TIBCO® Enterprise Runtime for R

Software Release 6.0.0

TIBCO®

Contents

Documentation and Support Services

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the TIBCO Product Documentation website, mainly in HTML and PDF formats.

The TIBCO Product Documentation website is updated frequently and is more current than any other documentation included with the product. To access the latest documentation, visit https://docs.tibco.com.

System Requirements for Spotfire Products

For information about the system requirements for Spotfire products, visit http://spotfi.re/sr.

TIBCO Enterprise Runtime for R documentation

You can find the following documents for TIBCO Enterprise Runtime for R in the TIBCO Documentation Library.

- *TIBCO® Enterprise Runtime for R Technical Documentation*
- *Language Reference* (HTML)
- *Differences Between TIBCO® Enterprise Runtime for R and Open-Source R* (HTML)
- *Release Notes* (PDF)
- *License Agreement* (PDF)

You can also find links to CRAN package compatibility reports for this release on TIBCO Cloud™ Spotfire®. See links in the TERR *Release Notes* for more information.

How to Contact TIBCO Support

You can contact TIBCO Support in the following ways:

- For an overview of TIBCO Support, visit http://www.tibco.com/services/support.
- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the TIBCO Support portal at https://support.tibco.com.
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to https://support.tibco.com. If you do not have a user name, you can request one by clicking Register on the website.

How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the TIBCO Ideas Portal. For a free registration, go to https://community.tibco.com.

For quick access to the TIBCO® Enterprise Runtime for R content, see https://community.tibco.com/products/terr.

TIBCO Enterprise Runtime for R System Requirements

TIBCO® Enterprise Runtime for R (TERR™) version 6.1.0 system requirements and third-party compatibility information are listed below.

Hard disk space
TERR™ requires 220 MB hard disk space

The following operating system versions have been tested with this version of TERR.

Tested operating systems	Tested version numbers
Microsoft® Windows	• 10 • 8 • 8.1 • 7
Microsoft Windows Server	• 2019 • 2016 • 2012 R2 • 2012
Red Hat® Enterprise Linux®	• 8.x 64-bit on x86-64 • 7.x 64-bit on x86-64 • 6.x 64-bit on x86-64
CentOS	• 8.x 64-bit on x86-64
SUSE® Enterprise Linux	• 15.x 64-bit on x86-64 • 12.x 64-bit on x86-64
Mac® OS X	Deprecated; no longer tested.

64-bit operating system strongly recommended.

The following third-party product versions have been tested with this version of TERR.

Third-party product	Tested version	Notes
Apache Tomcat	9.0.33	used by TIBCO Spotfire Statistics Services

Third-party product	Tested version	Notes
Java	11 through version 11.0.1	The following packages supplied with TERR require a bit-matching (32 bit or 64 bit) version of Java and must have the `JAVA_HOME` environment variable set to the location of a tested version of Java. • parallel • sjdbc • terrJava
Open-source R[1]	4.0.2	
OpenSSL	1.1.1g	
RStudio, RStudio Server[2]	1.3.1093	TERR has been tested for compatibility with the versions of the RStudio IDE listed here. Not all features are supported, and compatibility with future RStudio releases is not guaranteed. For more detailed compatibility information, type `help.start()` at the TERR command line, and in the resulting landing page, click the link *README for TIBCO Enterprise Runtime for R for RStudio*.
TIBCO Spotfire®	TIBCO Spotfire 11.1.0	

[1] Open-source R is available under separate open source software license terms and is not part of TERR. As such, open-source R is not within the scope of your license for TERR. Open-source R is not supported, maintained, or warranted in any way by TIBCO Software Inc. Download and use of open-source R is solely at your own discretion and subject to the free open source license terms applicable to open-source R.

[2] RStudio is available under separate open-source software license terms. TIBCO does not warrant, deliver, or support code or other material provided by RStudio, Inc., including but not limited to development tools and packages, and such code or other material does not constitute a part of the TIBCO Enterprise Runtime for R engine.

System Requirement Changes from the Previous Version

The following are changes in system requirements from version 5.1 to version 6.1.0 of TIBCO® Enterprise Runtime for R (TERR™).

Tested third-party software compatibility

Description	Version for this release	Status
Java	11	No change.
RStudio IDE compatibility	• RStudio Desktop 1.3.1093 • RStudio Server 1.3.1093	Updated from version 1.2.5042.
SUSE Enterprise Linux	• 15.x 64-bit on x86-64 • 12.x 64-bit on x86-64	No change.
open-source R	4.0.2	Updated from version 3.6.2.
Mac	Support for TERR on the Mac is deprecated and is no longer tested.	Deprecated.

Technical Guide

The TERR Technical Guide contains information on a variety of techncal aspects of working with TERR.

You could be trying to configure the startup behavior of your TERR installation, or you might be working with code across international boundaries. You might be dealing with heap size issues, or need to move code between open-source R and TERR. If you have a question about TERR, start in this guide.

Run the TIBCO Enterprise Runtime for R Console

Regardless from where you use an installation of TIBCO® Enterprise Runtime for R (TERR™), you can run a console interface for writing and testing functions scripts, or for running a batch process.

- If you have installed the stand-alone TERR™ console, for example, on Microsoft Windows, you can run the TERR console by double-clicking the program from the **Start** menu in Windows.
- You can run the engine's executable `TERR.exe` from its installation location. For example, for the 64-bit TERR 6.1 installation on 64-bit Windows, the engine is installed in `C:\Program Files\TIBCO \terr60\bin`.
- If you are running TERR in Spotfire, you can launch the TERR engine from the Spotfire menu by clicking **Tools** > **TERR Tools**, and then clicking **Launch TERR Console**. Also, you can find the executable in the same TERR Tools dialog by clicking **Copy TERR engine Path to Clipboard**.
- If you prefer to work in an integrated development environment, you can use the TERR engine with RStudio®. See *Technical Note: Configure RStudio® to use TIBCO® Enterprise Runtime for R* on https://docs.tibco.com/products/tibco-enterprise-runtime-for-r.

You can customize your TERR session by using the files `TERRenviron`, `TERRprofile`, `.TERRData`, and `.First`. See Customize the TERR environment at startup for more information.

Customize the TERR Environment at Startup

A user or an administrator can specify a variety of files, environment variables, and functions to customize a TERR session. These options are processed when the session starts. Changing them does not affect the current session of TERR.

Startup options are run in order and include the following.

1. `TERRenviron` files
2. `TERRprofile` files
3. `.TERRData` files
4. `.First` function

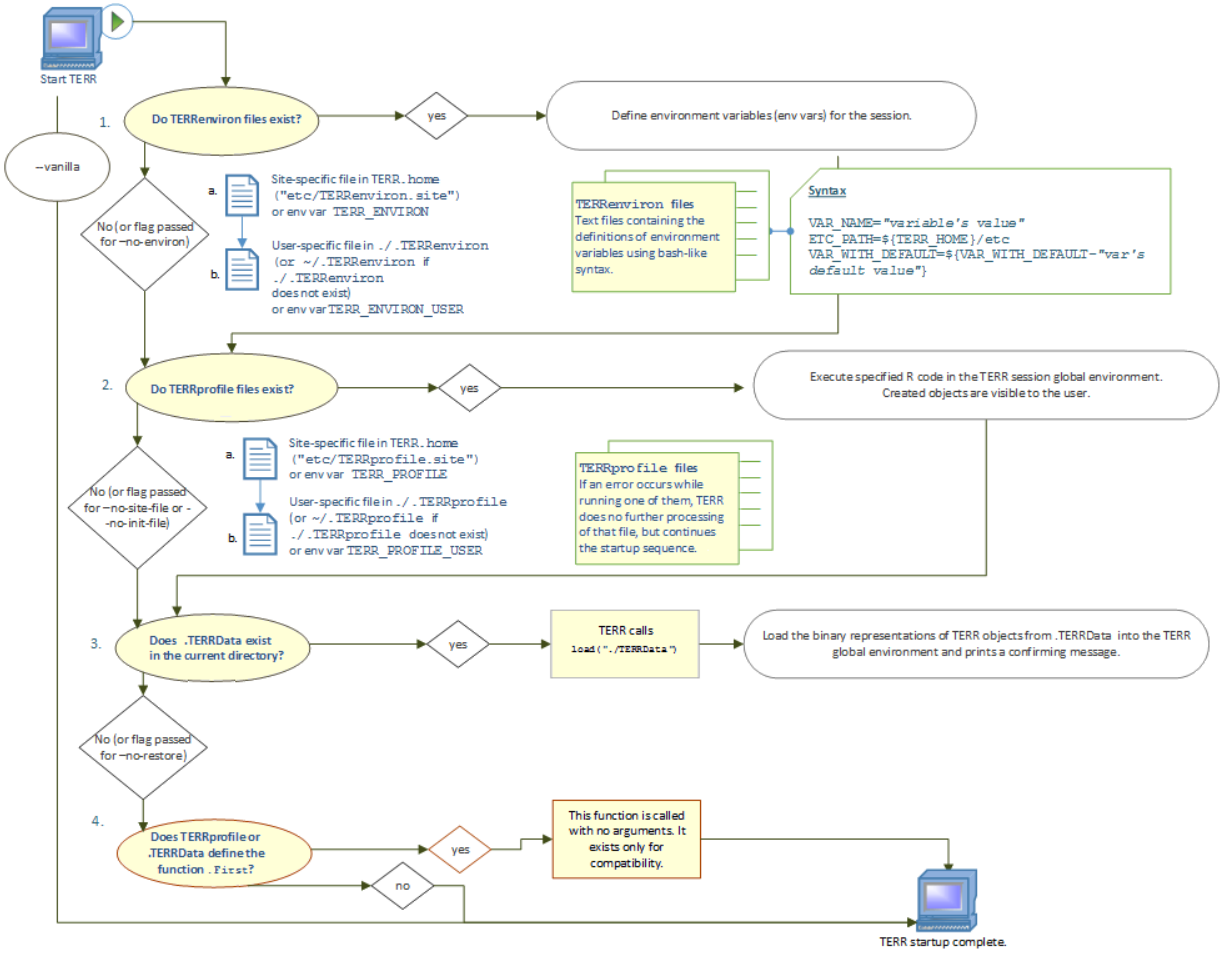
For both TERRenviron and TERRprofile, you can specify site-specific and user-specific files, which are processed in that order. Optionally, you can specify them as environment variables rather than files.

You can bypass running individual or all startup options by passing flags.

Optional startup flags	Description
`--vanilla`	All startup options ignored.
`--no-environ`	All TERRenviron files and environment variables are ignored.
`--no-site-file`	All site-specific profile files and environment variables are ignored.

Optional startup flags	Description
`--no-init-file`	All user-specific profile files and environment variables are ignored.

This flowchart shows you how these options are processed at startup.



TERRenviron

The optional `TERRenviron` files can contain the definitions of environment variables to specify such customizations as *`JAVA_HOME`* and the project directory for TERR. `TERRenviron` is the first file run in a customized TERR startup.

`TERRenviron` files can be defined in two places:

- The site-specific file in *`TERR_HOME`* (for example, in `etc/TERRenviron.site`)
- The user-specific file in `./.TERRenviron` (or in `~/.TERRenviron` if `./.TERRenviron` does not exist).

You can define environment variables for the TERR session using bash-like syntax in the `TERRenviron` files. Typical lines are as follows.

```
VAR_NAME="variable's value"
ETC_PATH=${TERR_HOME}/etc
VAR_WITH_DEFAULT=${VAR_WITH_DEFAULT-"var's default value"}
```

- The syntax `${VAR_NAME}` specifies that `${VAR_NAME}` should be replaced with the value of the environment variable `VAR_NAME`. If the environment variable is not defined, it should be replaced with an empty string.

- The syntax `${VAR_WITH_DEFAULT-"var's default value"}` specifies that `${VAR_NAME-"var's default value"}` should be replaced with the value of the environment variable `VAR_NAME` if `VAR_NAME` is defined. Otherwise, it should be replaced with "var's default value".

If you want to start a session and bypass running the `TERRenviron` files, you can use one of the following techniques.

- Use the command line option `--no-environ`.
- Use the command-line option `--vanilla`.

You can use nonstandard-named `TERRenviron` files by defining the environment variables `TERR_ENVIRON` and `TERR_ENVIRON_USER`, setting the name to the site-specific and user-specific environment files, respectively.

TERR does not report warnings or errors if these files do not exist.

Example

```
JAVA_HOME="C:/Program Files/Java/jre8"
```

TERRprofile

The optional `TERRprofile` files can contain R code that is run in the TERR session global enviornment so that objects created in them are visiable to the user.

`TERRprofile` files can be defined in two places:

- The site-specific file in *`TERR_HOME`* (for example, in `etc/TERRprofile`)
- The user-specific file in `./.TERRprofile` (or in `~/.TERRprofile` if `./.TERRprofile` does not exist).

If TERR encounters an error while running any code in `TERRprofile`, it does no further processing of the file but continues the startup sequence.

A profile file is run early in the startup sequence. You can depend on only the base package being loaded when they are run. You can use `loadNamespace("`*`pkg`*`")` or `library("`*`pkg`*`")` to load or attach other packages. Alternatively, you can use the double-colon syntax, such as *`pkg::func()`* to load the package *`pkg`* and run its function *`func`* as part of the startup sequence.

If you want to start a session and bypass running the `TERRprofile` files, you can use one of the following techniques.

- Use the command line options `--no-site-file` and `--no-init-file`, referring to the site-specific and user-specific profile files, respectively.
- Use the command-line option `--vanilla`.

You can use nonstandard-named `TERRprofile` files by defining the environment variables `TERR_PROFILE` and `TERR_PROFILE_USER`, setting the name to the site-specific and user-specific profile files, respectively.

TERR does not report warnings or errors if these files do not exist.

Example

Add the following line of code to the file *`TERR_HOME`*`\etc\TERRprofile.site`. When you next start TERR, the RinR package loads.

```
library("RinR")
```

.TERRData

The file `.TERRData` is created in the current directory by the function `save.image()` when you quit TERR and specify yes when prompted to save an image of the session.

`.TERRData` contains binary representations of TERR objects. On startup, if TERR finds this file in the current directory, it calls load("./TERRData") to load its objects into the global environment and displays a message that the previously saved workspace is restored.

.First

If you create a function called `.First` and put it in a `TERRprofile` file, or if it is saved in `.TERRData`, then it is run as part of the TERR startup.

the `.First` function is included in the startup options for backward compatibility only. We do not recommend using it.

If you override running `TERRprofile` using one of the command-line options, and you have defined `.First` in `TERRprofile`, it is not run.

If you include a `.First` function, be sure to test it to ensure that it works as expected to have it execute in subsequent sessions.

TIBCO Enterprise Runtime for R on macOS

Support for TERR on macOS has been deprecated and is no longer tested.

You can run install and run TERR in a Terminal window, or you can configure RStudio for the Mac to use the TERR engine, rather than the open-source R engine.

TERR installs only one package on macOS: `com.tibco.terr.framework`. (Open-source R installs four.)

Installing and running TIBCO Enterprise Runtime for R on a Mac

TERR provides a .dmg format file, which you can download and use to install TERR on an Apple macOS computer system.

As of TERR 6.0, support for TERR on the Mac is deprecated and is no longer tested.

Procedure

1. Download the file `TIB_terr_version.number_macosx_x86_64.dmg` to your Mac.
2. Find the file in your download folder, and then double-click it to display the installer icon.
3. Double-click the icon to run the installer.
4. Follow the installer steps, supplying a password if prompted to do so.
   TERR is installed on your Mac.
5. Open a Terminal window.
6. At the command prompt, type `terr`.
   A TERR session starts. You can now type TERR commands at the command prompt.
7. Optional: At the TERR prompt, type `help.start()`.
   A browser session is started, and the help page for TERR is displayed.

What to do next

If you use RStudio for Mac, you can configure it to use the TERR engine, also. For more information, see Configuring RStudio Mac desktop edition to run the TIBCO Enterprise Runtime for R engine.

Configuring RStudio Mac Desktop Edition to Run the TIBCO Enterprise Runtime for R Engine

You can use TERR in your RStudio installation on your Mac.

As of TERR 6.0, support for TERR on the Mac is deprecated and is no longer tested.

Prerequisites

- You must be running a supported version of the Mac operating system.
- You must have installed a supported RStudio version for the Mac desktop.
- You must have installed TERR version 6.1 on your Mac. See Installing and running TIBCO Enterprise Runtime for R on a Mac on page 15 for more information.

Procedure

1. On your Mac, open a Terminal session.
2. At the command prompt, type `RStudio-TERR`.
   RStudio launches with TERR as the language engine.

Running Multiple Versions of TIBCO Enterprise Runtime for R on Mac

When you run the TERR installer, it removes any previous versions of the TERR framework that it finds installed. You can work around this issue by running a package utility setting.

As of TERR 6.0, support for TERR on the Mac is deprecated and is no longer tested.

Perform this task in the Terminal on a Mac when you want to run more than one version of TERR.

Prerequisites

Close any running TERR session.

Procedure

1. At the Terminal command prompt, run the command `sudo pkgutil --forget com.tibco.terr.framework`.
2. Close the Terminal and run the installer for the new version.
   The newly-installed version now starts when you type `terr` at the command prompt.

   To run the previously-forgotten version of TERR, you must locate its framework (`com.tibco.terr.framework`) and double-click it to launch that version.

Uninstalling TIBCO Enterprise Runtime for R from a Mac operating system

You can remove TERR from your Mac operating system by running a command in the Terminal window.

As of TERR 6.0, support for TERR on the Mac is deprecated and is no longer tested.

Perform this task in a Terminal window on your macOS.

Prerequisites

A TERR installation on a supported version of macOS. See system requirements for more information.

Procedure

- At the Terminal command prompt, type the command to remove TERR, as follows.

```
sudo rm -rf /Library/Frameworks/TERR.framework /usr/local/bin/TERR /usr/local/bin/
TERRscript /usr/local/bin/RStudio-TERR
```

If you are prompted to supply a password, provide the administrator password for the computer.

The following three specified items are removed.

- The TERR framework package.
- Scripts for running TERR.
- A script to launch RStudio running TERR.

Result

TERR is removed from your macOS computer.

Troubleshooting Running Java and TIBCO Enterprise Runtime for R on a Mac

The terrJava package requires the Java Native Interface (JNI) capability, but this capability is not enabled by default in Oracle's JDK installation.

As of TERR 6.0, support for TERR on the Mac is deprecated and is no longer tested.

When you call `.JavaMethod` (or another terrJava function) on the Mac, if you receive the following error message, you must add the missing capability to the JDK `Info.plist`.

```
Java installation at /Library/Java/JavaVirtualMachines/jdk1.8.0_73.jdk/Contents/Home does not
 have required 'JNI' capability
```

Adding this missing capability to the Java configuration allows the JVM to sart as expected.

Procedure

- Type the following one-line command to add the JNI capability to the `Info.plist`.

```
sudo /usr/libexec/PlistBuddy -c "Add :JavaVM:JVMCapabilities: string JNI" $JAVA_HOME/../
Info.plist
```

Get Help with TIBCO Enterprise Runtime for R

You can get complete help for TERR from the console's command line.

The TERR help landing page, available in a web browser instance, provides general information and reference. From the TERR console command-line, type `help.start()`. From the resulting web page, you can select one of the following.

- Browse the language reference by package.
- Search the language reference for a specific function.
- Open and read product documentation and technical notes available on docs.tibco.com.
- Review the differences between TERR and open-source R.
- Review the latest information about implemented functions by category.

You can also get command-line help for TERR from the shell command line. At the command prompt, type `TERR --help`.

For this command to work, you must have the engine location in your path, or you must run the command from its `bin` directory.

Package Management in TIBCO Enterprise Runtime for R

You can develop custom packages, or you can use one of the packages customized to use specifically with TERR and posted on the TERR Archive Network, or you can use one of the many packages developed and posted on the Comprehensive R Archive Network (CRAN).

- You can use TERR to develop packages to share with other R developers.
- You can install packages directly on a Spotfire Statistics Services server using `install.packages()` from the engine on the server.
- You can put the packages on a Spotfire Statistics Services server using the Eclipse plugin designed to deploy packages to Spotfire Statistics Services, or you can put the packages on a Spotfire Statistics Services server using the spserverapi package function `administrationService.uploadPackageVersion`. (See its help for more information.)
- You can use the functions in your custom packages or in CRAN packages in data functions developed for Spotfire analytics, and then share the Spotfire analytics with others.

See the Package Management for the TIBCO Spotfire® Environment for more information.

Be sure you follow the recommended practices for maintaining package versions.

Installation Options for Packages

Use the function `install.packages()` to install packages to use in TERR either in the stand-alone console, or in Spotfire. You can find packages in a variety of locations, including repositories, on reliable

web sites, or stored locally. See the TERR help topic for `install.packages()` for more detail and examples.

Package Location	Description	Example
TERR Archive Network (TRAN) repository	The default location, `https://tran.tibco.com`, for installing packages using `install.packages( )`. Used for packages that have been customized to work specifically with TERR. Requires no further arguments.  Some packages customized and placed on TRAN require other packages not available on TRAN. Some of these packages cannot be installed using the TERR function `install.packages`, so the TRAN package cannot be successfully installed. If you encounter this situation, try building and installing the package using open-source R. For information on the installation differences between TERR and open-source R, see Specifying an older package from TRAN.	`# install R Datasets Package found on TRAN:` `install.packages("datasets")`
Microsoft R Application Network (MRAN) CRAN mirror repository	The second value in `options("repos")` is initialized to a URL to the MRAN repository, which contains snapshots of the CRAN packages on any given date. This URL specifies a fixed date on MRAN and will not access CRAN package versions after that date.	`#install package tidyr from MRAN:` `install.packages("tidyr")`
Comprehensive R Archive Network (CRAN) repository	CRAN is included in the default `options("repos")` to handle cases where the package is not available on TRAN or on MRAN (probably because it is a new package). If the default does not work, you can call `install.packages` setting the `repos` argument. One case where this option is useful is when accessing other repositories (such as Bioc).	`# install rpart package from CRAN:` `install.packages("rpart", repos="https://cloud.r-project.org"))`

In-house repository	You or someone in your organization has set up a CRAN-like repository (either on a network share or on a web server) using a tool like the drat package. All TERR or open-source R users in the organization can access the same package version from the repository.	`#download and install from a`
		`# local web service using a URL:`
		`URL <- "https://mycompanysvc/mypackage")`
		`install.packages(URL)`

Locally-available packages	If a trusted source gives you a package as a zip archive, you can put in on your computer and install it using `install.packages()`.	`# install local newtree package`
		`# from a zip file in the working directory:`
		`install.packages("newtree_1.2.zip")`

Trusted URL	If you are given a URL that contains a package you might want to use, and you trust the URL, you can pass the URL as the only argument to `install.packages()`.	`#download from a custom URL and install`
		`# a custom package`
		`URL <- "https://customurl/mypackage")`
		`install.packages(URL)`

The repositories contain binary packages (for Windows) and source packages (for Linux and Windows). You can easily install most binary and source packages in TERR. If you have problems building from source, then build the packages using open-source R before installing them into TERR. Note that TERR does not build binary packages from source packages that contain Java source code.

Platform	Package type	Notes
Linux, Windows	Binary	Call `install.packages(`*`pkgname`*`)`. TERR installs the binary package into your specified package directory.
Linux	Source; no Java code, no C/C++ or Fortran code	Call `install.packages(`*`pkgname`*`)`. TERR builds the source package into a binary package and installs it into your specified package directory.
Linux, Windows	Source; C/C++ or Fortran code (no Java code)	On Windows, first you must install the Rtools utilities package, which is maintained by Duncan Murdoch, and then update your PATH to specify the location of the utilities. 1. If you have not already done so, install the package rinclude by calling `install.packages(rinclude)` 2. Call `install.packages(`*`pkgname`*`)`. See Installation Options for Packages on page 18 for information on repositories accessed by `install.packages`. TERR builds the source package into a binary package and installs it into your specified package directory. If the package does not build and install, then try building it with open-source R, and then installing the binary as described here.
Linux	Source; Java code	1. Build the package using open-source R tools for building packages from source. The tools compile the source code to create the binary package. 2. Call `install.packages(`*`pkgname`*`)`.

See the help for `install.packages(`*`pkgname`*`)` for more information.

Due to changes in open-source R version 3.5 and resulting compatibility changes in TERR 5.0, packages that are built with a version of TERR prior to 5.0 must be rebuilt.

- To install a binary package from a repository, always call `install.packages(`*`pkgname`*`)` from TERR. The `install.packages` function finds the correct binary version in the repository for your version of TERR. Manually downloading the binary package from CRAN can result in errors when you use it with TERR.
- To install a package from source, try installing it first with TERR (with `install.packages` in TERR or with `TERR CMD INSTALL` from a command line).
- To install a package from source that you cannot build with TERR, install the package with the version of open-source R tested with TERR.

To get more information about the packages on TRAN, run the following code in TERR:

```
ap <- available.packages(contrib.url(getOption("repos")[1],
    getOption("pkgType")))
# to print the entire matrix
ap
# to print just the package names
row.names(ap)
```

Setting JAVA_HOME

Some packages that you use with TERR require access to Java on your system. If you call the TERR function `Sys.getenv("JAVA_HOME")` and it returns an empty string, you must set *JAVA_HOME* so the packages can access Java.

Perform this task on your Windows Or Linux system.

Prerequisites

The following list describes a few of the packages that are either provided with TERR or that you can use with TERR, but they require a bit-matching 32-bit or 64-bit version of Java, version 6 or later. (You might find other packages that require Java. These instructions can help you prepare your TERR session for those packages, too.)

Package name	Provided in your TERR installation
parallel	yes
sjdbc	yes
terrJava	yes
rJava	no  See Installing the rJava package for more information.

Procedure

1. Locate your Java installation and make a note of it.

   Your system can have more than one version of Java. Generally, use the latest version. On Windows, you can find this path in the registry. On Linux, you can usually find a link to it in the `\user\bin` directory.

   For example, on Windows, this path might be `C:/Program Files/jdk-11.0.1`.
2. Start a session of the TERR console.
3. At the TERR command prompt, type the command `Sys.setenv(JAVA_HOME="`*`path_to_ your_Java_installation`*`")` where *`path_to_ your_Java_installation`* is the path you noted in Step 1.
   For example, on Windows, this call might look like the following.

   ```
   > Sys.setenv(JAVA_HOME="C:/Program Files/jdk-11.0.1")
   ```

   On Linux, this call might look like the following.

   ```
   > Sys.setenv(JAVA_HOME="/usr/lib/jvm/java-11-sun/"
   ```

   Your system environment *JAVA_HOME* is now set to the specified Java installation.

4. Optional: Check the setting for *JAVA_HOME* from TERR by typing `Sys.getenv("JAVA_HOME")`.
   For example, on Windows, it might look like the following.

```
> Sys.getenv("JAVA_HOME")
[1] "C/Program Files/jdk-11.0.1"
```

What to do next

Install the package that requires setting *JAVA_HOME*. For an example, see Installing the rJava package.

Installing the rJava Package

The rJava package gives access to low-level R functions to the Java interface, but it is not provided with TERR. These instructions help you prepare your computer to use rJava.

Prerequisites

The rJava package requires the following.

- A bit-matching 32-bit or 64-bit version of Java, version 6 or later, is installed. (Tested with version 11.0.1.)
- The system variable *JAVA_HOME* is set. Follow the instructions for Setting JAVA_HOME if you are unsure.

These instructions are for installing the rJava package for use with TERR 4.2 or later. If you are using an earlier version, and you cannot update your version of TERR, see the release notes for more information for the verison of TERR you are running.

- For TERR version 3.1 and earlier, the rJava package does not work. To use rJava, update your version of TERR.
- For TERR version 3.2, you must use a build of rJava from TRAN. See that version's release notes for more information.

Perform this task in the TERR console or in TERR running under RStudio.

Procedure

1. At the command prompt, type `install.packages("rJava")`.
   The rJava package is installed from the package repository to the `site-library` directory.
2. At the command prompt, type `library(rJava)`.
   For example:

```
> library(rJava)
The following object(s) are masked _from_ 'package:utils':

    head, str, tail
The following object(s) are masked _from_ 'package:methods':

    new, show
The following object(s) are masked _from_ 'base':

    anyDuplicated, duplicated, rev, sort, unique
```

   The rJava package is now in your search path.
3. At the command prompt, type `searchpaths()`.
   For example:

```
> searchpaths()
[1] ".GlobalEnv"
[2] "C:/Program
    Files/TIBCO/terr60/site-library/rJava"
[3] "C:/Program
    Files/TIBCO/terr60/library/stats"
```

```
[4] "C:/Program
    Files/TIBCO/terr60/library/graphics"
[5] "C:/Program
    Files/TIBCO/terr60/library/grDevices"
[6] "C:/Program
    Files/TIBCO/terr60/library/utils"
[7] "C:/Program
    Files/TIBCO/terr60/library/methods"
[8] "C:/Program
    Files/TIBCO/terr60/library/base"
```

The absolute file path is returned for each package in the current environment. Note that by default, the newly-loaded rJava is listed second in the search path.

Recommendations for Using R Securely

The R Consortium, of which TIBCO is a proud member, has provided a summary of "Best Practices for Using R Securely."

We encourage anyone using open source R, whether with TIBCO products or not, to review those practices at the following site: https://www.r-consortium.org/blog/2015/08/17/best-practices-for-using-r-securely. This guidance essentially recommends that users who download R and R packages do so from a secure server using an encrypted HTTPS connection.

The following guidance provides information regarding how these recommendations do, or do not, apply to TERR.

Recommendation: If you download open-source R, always download it from a server using HTTPS

TERR is a commercial product, and you download it from our secure TIBCO Product Download site. This site use HTTPS.

Recommendation: If you download open-source R, check its MD5 checksums before you begin the installation

Customers downloading TERR from the TIBCO Product Download site should confirm the MD5 checksums following the same process as in detailed in the R Consortium blog post, cited in this topic.

Recommendation: If you have open-source R installed, configure it for secure file downloads

By default, TERR uses HTTPS for secure file download if a secure mirror is specified. There is no need to do any special configuration of TERR.

Recommendation: Always download CRAN packages from a secure mirror

We recommend TERR users follow this recommendation, and always download CRAN packages from a secure mirror. The Best Practices post includes a list of CRAN sites that use HTTPS.

By default, TERR installs packages from the Microsoft R Application Network (MRAN) snapshot with CRAN package versions available when this version of TERR was made available. This MRAN snapshot is a secure site.

Open-source R is available under separate open source software license terms and is not part of TERR. As such, open-source R is not within the scope of your license for TERR. Open-source R is not supported, maintained, or warranted in any way by TIBCO Software Inc. Download and use of open-source R is solely at your own discretion and subject to the free open source license terms applicable to open-source R.

Manage your Packages when You Install a New Version of TERR

When a new version of TERR is released, you might want to install it to take advantage of the changes. You can run older and newer versions of TERR on the same computer, or you can uninstall the older version(s). In either case, you probably want to make sure any custom-created packages or packages downloaded from a repository are kept available to the TERR version(s) you are running.

Uninstalling TERR does not remove the packages you installed. However, we recommend that you check for updates to any packages you have downloaded from package repositories after you install a new version of TERR. You can check for updated versions by calling `update.packages()`. See the help topic for `update.packages()` in the TERR *Language Reference* for more information.

The TERR installation includes the directory *`TERR_HOME`*`/site-library`, which is used by default. If you want to use another directory, you can define the environment variable *TERR_LIBS_SITE* and set it to the directory of your choice.

Initially, the `site-library` directory is empty. If you have permission to write to the *`TERR_HOME`* directory, any packages you create or download are installed in *`TERR_HOME`*`/site-library`.

Installing packages to the `site-library` directory provides the following advantages.

- It provides you with the means to protect and manage the packages you installed and want to keep, separate from the new installation.
- It separates the packages shipped with TERR so they can be updated with new releases, and so you do not accidentally change or remove them.

You should avoid changing any entries in the *`TERR_HOME`*`/library` directory. Doing so can cause TERR to behave in unexpected ways.

The directory *`TERR_HOME`*`/site-library` is added to the head of the search path, which is returned by `.libPaths()`. For example, on a Windows computer where you have permission to write to the *`TERR_HOME`* directory, this function call would appear as follows.

```
> .libPaths()
[1] "C:/Program Files/TIBCO/terr60/site-library"
[2] "C:/Program Files/TIBCO/terr60/library"
```

After installing the new version of TERR, you can just copy the packages from the older *`TERR_HOME`*`/site-library` directory to the new *`TERR_HOME`*`/site-library` directory.

If you are downloading packages to a computer where you do not have permission to write to the directory *`TERR_HOME`*`/site-library`, then packages are stored in the user directory. For example, on Windows, this directory is *`[My Documents]`*`/TERR/x86_64-pc-windows-library/<`*`version`*`>`, and, calling `.libPaths()` would appear as follows.

```
> .libPaths()
[1] "C:/users/jdoe/Documents/TERR/x86_64-pc-windows-library/terr6.0"
[2] "C:/Program Files/TIBCO/terr60/site-library"
[3] "C:/Program Files/TIBCO/terr60/library"
```

In this case, you can ignore the `site-library` directory (which remains empty) and manage your packages by copying them from the older *`<version>`* to the new *`<version>`* in the user directory location.

Use TIBCO Enterprise Runtime for R with TIBCO Spotfire

If you are developing and using data functions in Spotfire Analyst, you can select one of three possible TERR engines.

Spotfire provides three configuration options for using the TERR engine. You can set these options in the Spotfire Analyst user interface. On the Spotfire Analyst menu, click **Tools** > **Options**, and then scroll down and select the **Data Functions** option. In the resulting display, you can set one of the following:

- The locally-installed TERR engine, if you are running Spotfire Analyst.
- The default TIBCO Spotfire® Statistics Services deployed and configured with the TERR engine. If you are displaying data visualizations, you must have Spotfire Statistics Services deployed in your Spotfire Server.
- A custom Spotfire Statistics Services URL (such as the TIBCO Spotfire® Local Adapter).

Finding information	Resource
The default Spotfire Statistics Services URL.	See your Spotfire Server administrator.
Setting options for data functions.	*TIBCO Spotfire® User's Guide*, available from the Spotfire user interface.
Sharing Spotfire analyses that use TERR packages.	Package Management.
Creating functions that run the TERR engine in Spotfire Analyst.	Advanced Analytics in Spotfire.

Batch Processing

You can call the TERR engine non-interactively using a batch processing interface.

Batch processing is handled through standard input and standard output (`stdin`/`stdout`) redirection at the OS level, along with various optional arguments and settings.

For a list of commands that you can use for batch processing, see Command Line Options for TIBCO Enterprise Runtime for R on page 25.

Command Line Options for TIBCO Enterprise Runtime for R

From the shell command line, you can run the TERR command, passing in the following options, either interactively or in a batch process. You can also get help using the syntax and options described here.

Usage

```
TERR [options]
```

Description
Issue commands, either interactively or in a batch process, to TERR from the shell command line.

Options
`-o`, `--option=value` means that `-o value` and `--option=value` are synonymous.

Option	Description
`--args`	Do not process the rest of the arguments.
`--color`	Color input and output differently.
`--console-editor`	Use console line editor.

Option	Description
`--console-encoding ENC`, `--console-encoding= ENC`	Use character encoding `ENC` on `stdin` or `stdout` (defaults to `--encoding value`).
`--debug`	Enable more debug information.
`--disable-signal-handlers`	Disable signal handlers to catch OS-specific signals. The default behavior is to disable signal handlers.
`-e EXPR`	Runs `EXPR` and quits, unless `--interactive` is specified.
`--echo-console`	Always echo console input text in interactive mode (if neither is given, echoing is controlled by `options('echo')`).
`--enable-signal-handlers`	Enable signal handlers to catch OS-secific signals (the default is to disable signal handlers)
`--encoding ENC`, `--encoding=ENC`	This argument is always ignored: native encoding is always `'UTF-8'`.
`-f FILE, --file=FILE`	Runs `FILE` and quits unless `--interactive` is specified.
`-h, --help`	Print this help message and exit.
`-i, --interactive`	Force interactive mode. • If the engine is started with `-e` or `-f`, then the interactive command line is started after the specified expression or file is executed. • If `-e` or `-f` is not specified, this ensures that `interactive()` returns `TRUE`, even if `isatty(stdin())` is `FALSE`.
`--internet2`	Use Internet Explorer settings for proxies and other tasks. Microsoft Windows only.
`--no-console-editor`	Disable console line editor. This is also disabled if `-e` or `-f` is specified, or if `isatty(stdin())` or `isatty(stdout())` is `FALSE`.
`--no-echo-console`	Never echo console input text in interactive mode (if neither is given, echoing is controlled by `options('echo')`)
`--no-environ`	Do not read the site and user environment files.
`--no-init-file`	Do not read the user `.TERRProfile` file.
`--no-readline`	See `--no-console-editor`.
`--no-restore`	Do not restore saved workspace.
`--no-restore-data`	Do not restore saved workspace data.
`--no-restore-history`	Do not restore saved workspace history.
`--no-save`	Do not save workspace at end of session.
`--no-site-file`	Do not read the site `.TERRProfile` file.

Option	Description
`--profile=TYPE`	Enable profiling, where TYPE is one of the following: • `time` • `objects` • `memory` • `coverage`
`-q, --quiet`	Do not print startup message.
`--read-init-file`	Read the user `.TERRProfile` file.
`--read-site-file`	Read the site `.TERRProfile` file.
`--restore`	Restore saved workspace.
`RHOME`	Print path to the TERR home directory and exit.
`-s, --no-echo`	Run as quietly as possible.
`--save`	Save workspace at the end of the session.
`--silent`	Same as `--quiet`.
`--spotfire`	Used when launched by Spotfire.
`--vanilla`	Combine the following: • `--no-save` • `--no-restore` • `--no-site-file` • `--no-init-file` • `--no-environ`
`--verbose`	Print more information.
`--version`	Print version and copyright information, and then exit.
`--version`	Print version and copyright information and exit

CMD Commands in TIBCO Enterprise Runtime for R

You can issue a `CMD` command at the shell command line in TERR.

For these commands to work, you must have the engine location in your path, or you must run the command from its `bin` directory.

Command Line Options for the build Command

You can build packages from the command line by using the command `TERR CMD build` and specifying any of the following options to control the results.

Usage

```
TERR CMD build [options] pkgDirs
```

Description
Build packages specified by *pkgDirs*.

Options
-o, --option=value means that *-o value* and *--option=value* are synonymous.

Option	Description
-h, --help	Describe the arguments that can be passed to this command.
--binary	Build distributable archives of newly-installed packages.
--md5	Add MD5 sums.

Command Line Options for the check Command

You can check packages that you build for common errors from the command line by using the command `TERR CMD check` and specifying any of the following options to control the results.

Usage

```
TERR CMD check [options] pkgDirs
```

Description
Check packages specified by *pkgDirs* for common errors.

Options
-o, --option=value means that *-o value* and *--option=value* are synonymous.

Option	Description
-h, --help	Describe the arguments that can be passed to this command, and then exit.
-v--version	Print the package version information, and then exit.
-l--library=*LIB*	The library directory used for test installation of packages. (The default is `outdir`.
-o--outdir=*DIR*	The directory where log files, TERR output, and other information is written. The default is *pkg*`.TERRcheck` in the current directory, where *pkg* is the name of the package checked.

Command Line Options for the INSTALL Command

You can install packages from the command line by using the command `TERR CMD INSTALL` and specifying any of the following options to control the installation.

Usage

```
TERR CMD INSTALL [options] pkgs
```

Description
Install packages specified by *pkgs*, which can be the names of source package directories, source package `tar.gz` files, or binary (prebuilt) package `.zip` files.

Options
-o, --option=value means that *-o value* and *--option=value* are synonymous.

Other arguments that can be used with `R CMD INSTALL` are ignored.

Option	Description
`-h, --help`	Describe the arguments that can be passed to this command.
`-l, --library=LIB_DIR`	Install packages to the library directory `LIB_DIR`.
`--build`	Build distributable archives of newly-installed packages.
`--install-tests`	Install the specified package tests directory.
`-d, --debug`	Enable debugging messages.
`-v, --version`	Display the package installer version information.
`--no-R, --no-libs, --no-data, --no-help, --no-demo, --no-exec, --no-inst`	Suppress the installation of the specified part of the package (for testing).
`--no-configure`	Do not run the configuration scripts.
`--no-test-load`	Do not try to attach the package in a new TERR process after installing. Otherwise, an unattachable package is deleted.
`--clean`	After installation, remove from the source directory any files created during installation.
`--preclean`	Before installation, remove from the source directory any files made during a previous installation.
`--configure-args`	Command line arguments for the package configuration script.
`--configure-vars`	The environment variables set before running the package configuration script.
`--(with|without)-keep.source`	Keep or do not keep the source information with the R functions.
`--(with|without)-keep.parse.data`	Keep or do not keep the parse data with the R functions.

Command Line Options for the Rdconv Command

You can convert `.Rd` documentation to other formats by using the command `Rdconv` and specifying any of the following options to control the results.

Usage

```
<file-path-to>Rdconv.exe [OPTIONS] FILE
```

Description

Convert the file specified by *`FILE`* to the format specified by `--type=`*`arg`*.

Options

`-o, --option=value` means that *`-o value`* and *`--option=value`* are synonymous.

Option	Description
`-h, --help`	Print a help message for this command, and then exit.
`-v , --version`	Print file version information, and then exit.
`-d , --debug`	Enable debugging.
`-t , --type arg (=html)`	Convert to the format specified by *arg*.
`--encoding arg (=utf-8)`	Use *arg* as the output encoding.
`--package arg (=unknown)`	Use *arg* as the package name.
`-o , --output arg (=-)`	Use *arg* as the output file.
`--os arg (=windows)`	Assume the operating system *arg* (can be either UNIX or Windows).
`--run_dontrun`	Either comment out or do not comment out the `\dontrun` sections of examples.
`--run_donttest`	Either comment out or do not comment out `\donttest` sections of examples.

Embed the Supported TIBCO Enterprise Runtime for R Engine

The TERR engine is designed to be embedded in applications that require including a general computational statistical engine.

TERR has an embedding interface that implements a fully-compatible subset of the R embedding API, which is included in the open-source R engine developed by The R Project for Statistical Computing. Because TERR is targeted at adding computational functionality to applications, it includes none of the R embedding APIs providing GUI support.

Open-source R is available under separate open source software license terms and is not part of TERR. As such, open-source R is not within the scope of your license for TERR. Open-source R is not supported, maintained, or warranted in any way by TIBCO Software Inc. Download and use of open-source R is solely at your own discretion and subject to the free open source license terms applicable to open-source R.

- If you have an existing application that embeds the open-source R engine using the subset of the embedding APIs supported by TERR, you can embed the TERR engine by simply replacing the open-source R binaries with the TERR binaries.
- If you are developing an application and want to embed the TERR engine, you can do so by adding embedding support for the open-source R engine, and then including the TERR binaries instead of the open-source R binaries in the final application.

Internationalization

TERR supports many, but not all, localization features available in open-source R.

Open-source R has some support for internationalization and localization, representing strings with Unicode characters, and managing localized errors and messages in different languages.

Some of these differences include string representation, language parsing, locale, file input-output, localized messages, support for typing and displaying characters in the console in Japanese and other languages.

String Representation

TERR and open-source R vary in their string encodings.

Open-source R allows strings to be represented in any of four encodings:

- `"unknown"` (the default character encoding for the system)
- `"latin1"`
- `"UTF-8"`
- `"bytes"`

TERR currently creates all strings as `"unknown"` by default (note that `"unknown"` encoding is hard-wired to use UTF-8). The functions `Encoding` and `iconv` can be used for constructing strings with other encodings.

TERR allows adding Unicode characters into a string using an escape sequence such as `"\u30A4"` or `"\U{30A4}"` to create a string containing a single Japanese character. Alternatively, it is possible to add Unicode characters into a typed string by typing them, or copy-and-pasting them. Exactly which characters can be typed or printed depends on how the console is set up (described below).

The TERR string-manipulation functions (`substring`, `nchar`, `paste`, and so on) correctly handle UTF-8 strings. Functions for searching strings with regular expressions (`regexpr`, `grep`, `strsplit`, and so on) correctly handle UTF-8 strings as the data or the pattern strings.

TERR implements several functions for constructing strings from integers or raw bytes: `intToUtf8`, `utf8ToInt`, `charToRaw`, `rawToChar.`

S Language Parsing

The manner in which TERR parses character types aids in internationlization.

The TERR parser can process text containing Unicode characters. All S language keywords (`function`, `for`, and so on) are ASCII. Numbers are parsed only when written with ASCII digits using the US number format (such as "1234.56").

All non-ASCII Unicode characters are treated as possible name characters, except for space characters. Thus, it is possible to have a `data.frame` column name, a factor level, or a variable name containing Japanese characters.

Locale

Programmers using TERR in other locations need to understand its locale limitations.

TERR does not currently recognize the "locale" where it is running, and it has limited support for locale-specific string handling (such as performing sort according to the alphabetic order defined for different countries). For more information about supported locale settings, see the help file for the function `Sys.setlocale`.

File I O

Programmers need to understand character encoding arguments in TERR.

The functions for creating connections (such as `file` and `socketConnection`) have an encoding argument that specifies a character encoding for the connection. Characters are converted to or from that encoding when writing to or reading from the connection.

You can use the option `--console-encoding` and the function `terrUtils::setConsoleEncoding` to set the encoding for console reading or writing to `stdin` and `stdout`.

Localized Messages

Programmers need to understand the limitations of TERR

TERR does not currently do any locale-specific message handling. All messages and errors are in English.

Move Data Between TIBCO Enterprise Runtime for R and Open-Source R

You can transfer data objects between TERR and open-source R.

Both the TERR engine and the open-source R engine can read in plain ASCII data files (for example, `CSV` files) using the `scan()` and `read.table()` functions. To transfer basic data objects (vectors, matrices, factors, and data frames, for example) between TERR and open-source R, you must use other functions.

Transferring Data Objects from TIBCO Enterprise Runtime for R to Open-Source R (or Vice Versa)

You can use the `dump` and `source` functions to transfer data objects between TERR and open-source R.

You can perform this task from either the open-source R system or the TERR system. The advantage of using this technique of transferring data objects between TERR and open-source R is that `dump()` creates an ASCII file that can be viewed and edited outside of both systems.

Prerequisites

You have access to the consoles for both open-source R and TERR and their base packages. Review the help files for further information about these functions prior to using them.

Procedure

1. From the system to transfer from, call the `dump()` function.
   Provide the appopriate arguments, including a character vector giving the name(s) of the data object(s) and a file name to accept the output.

   You can dump multiple objects into a single dump file.

   An ASCII file containing the output is created.
2. From the system to transfer to, call the `source()` function.
   Provide the name of the ASCII file created from calling `dump()`.

Transferring Data Objects from Open-Source R to TIBCO Enterprise Runtime for R

You can use this alternative to the `dump` and `source` functions if you do not need to see or edit the R output before using it in TERR.

Prerequisites

You have access to the consoles for both open-source R and TERR and their base packages. Review the help files for further information about these functions prior to using them.

Procedure

1. From open-source R, call the `save()` function.
   A binary file containing the data object is created.
2. From TERR, call the `load()` function.
   Provide the name of the file created by calling `save()` in open-source R.

Transferring Data Objects from TIBCO Enterprise Runtime for R to Open-Source R

You can use this alternative to the `dump` and `source` functions if you do not need to see or edit the TERR output before using it in R.

Prerequisites

You have access to the consoles for both open-source R and TERR and their base packages. Review the help files for further information about these functions prior to using them.

Procedure

1. From TERR, call the `save()` function.

   You must set `RFormat=TRUE`. This option writes binary files in an open-source R-readable format.

   A binary file containing the data object is created.
2. From open-source R, call the `load()` function.
   Provide the name of the file created by calling `save()` in open-source R.

Manage Heap Size

TERR does not have built-in memory allocation size limits, except those imposed by the operating system (for example, the 2 gigabyte limit on 32-bit Windows). However, to prevent possible negative effects on the performance of other running applications, it is possible to set a limit on the total memory allocation by calling the `memory.limit` function.

The initial value of the `memory.limit` function is zero (meaning no limit), and it can be set to an integer for the number of megabytes. This limit might keep attempts at simultaneous large memory allocations (for example, those of large matrices) from succeeding.

The following example demonstrates increasing the total allocation limit to 2 gigabytes after hitting a 1 gigabyte limit with two large object allocations.

Each call to memory-limit returns the previous value of the limit.

```
memory.limit(1024) #set limit to 1GB
[1] 0
m <- 10000
n <- 10000
A <- matrix (runif (m*n),m,n)
# ~760 MegaByte - success
B <- matrix (runif (m*n),m,n)
# ~760 MegaByte more - failure due to initial 1GB limit
Error: out of memory
memory.limit(2*1024) # increase the limit to 2 GB
[1] 1024
memory.limit()
[1] 2048
B <- matrix (runif (m*n),m,n)
# ~760 MegaByte more - success after the limit increase
```

Signal Handlers and TIBCO Enterprise Runtime for R

TERR supports installing signal handlers.

Signal handlers are used to catch illegal operations (such as referencing an illegal memory location) that occur in foreign code called via the `.C()` or `.Call()` functions. If an illegal operation is caught by a signal handler, it will generate an error "Unhandled exception in foreign function" rather than crashing the process. Currently, these signal handlers are disabled by default, because they can interfere with signal handlers used by other software running in the same process. For example, we discovered problems when using Java and JDBC to access a database.

Sometimes when you need to investigate unexpected failures in foreign code, you might find it useful to enable the signal handlers during development to catch illegal operations. The signal handlers can be enabled when starting the TERR console application by specifying the option `--enable-signal-handlers`.

Working with the AsterDB Package

The TERR package AsterDB is provided in the Spotfire Statistics Services installation. If you are creating Spotfire Data Functions that use this package for a Teradata® Aster Database installation, see your system administrator.

The AsterDB package uses Java and requires special configuration before it is distributed for use. If it is distributed to you via a Spotfire update, make sure that you have installed a compatible version of Java, and that you have the JVM installed. Also, you must set the environment variable JAVA_HOME to that installation location.

Using TERR for Advanced Analytics in Spotfire

TERR is embedded in Spotfire and is used in various ways.

- You can use the predictive analytics tools built into Spotfire (which use the TERR engine "behind the scenes").
- You can access the engine directly using the TERR Tools in Spotfire to prototype, test, and debug functions.
- You can write ad hoc custom expressions that call TERR functions.
- You can write TERR data functions to store in the Spotfire library to reuse.

This documentation includes reference, information, example data sets, and walk-through tasks to help you learn how to use TERR in Spotfire.

Data Type Mapping

When you use TERR or open-source R expression functions or data functions in Spotfire, you must know how the supported data types are mapped between TERR (or open-source R) and Spotfire.

Spotfire data types	TERR or open-source R data types
Binary	`raw`
Boolean	`logical`
Currency	`numeric`
Date	`POSIXct` or `POSIXlt` with time zone UTC
DateTime	`POSIXct` or `POSIXlt` with time zone UTC
Integer	`integer`
LongInteger	`numeric`
Real	`numeric`
SingleReal	`numeric`
String	`character` (encoded as UTF-8)
Time	`POSIXct` or `POSIXlt` with zone UTC giving time on date 1/1/1970
TimeSpan	difftime

- All numeric invalid values in Spotfire are represented as NAs in TERR and open-source R engines. There is no special support for invalid values of other types. They become valid default values in TERR and open-source R engines.
- Spotfire converts TERR or open-source R factors to the data type String.
- Do not use the data type `date` when you use TERR or open-source R with Spotfire data types Date, DateTime, and Time. Spotfire converts the TERR data type `date` to the data type Real.
- Other data types than the ones included in this topic are not supported.

Open-source R is available under separate open source software license terms and is not part of TIBCO Spotfire Statistics Services. As such, open-source R is not within the scope of your license for TIBCO Spotfire Statistics Services. Open-source R is not supported, maintained, or warranted in any way by TIBCO Software Inc. Download and use of open-source R is solely at your own discretion and subject to the free open source license terms applicable to open-source R.

Data Dimension Mapping

When you use TERR or open-source R expression functions or data functions in Spotfire, you must know how the supported data dimensions are mapped between TERR (or open-source R) and Spotfire.

Spotfire data dimension	TERR and open-source R data dimension
Value	Vector of length 1
Column	Vector
Data table	data.frame

Open-source R is available under separate open source software license terms and is not part of TERR. As such, open-source R is not within the scope of your license for TERR. Open-source R is not supported, maintained, or warranted in any way by TIBCO Software Inc. Download and use of open-source R is solely at your own discretion and subject to the free open source license terms applicable to open-source R.

Accessing TIBCO Enterprise Runtime for R Directly from Spotfire

You can get direct access to TERR from your Spotfire installation. You can run the console, open an instance of th RStudio IDE that uses the TERR engine, or open the language reference. You can also find the path to the locally-installed TERR engine from Spotfire.

To start either the TERR console or the RStudio IDE, open Spotfire.

Prerequisites

You must have an installation of Spotfire 7.5 or later.

Procedure

1. From menu, click **Tools** > **TERR Tools**.
2. In the TERR Tools dialog box, click the option to perform.

Option	Description
Copy the TERR Engine Path to the Clipboard	Copy the contents of the text box **Path to Local TERR Engine**. You can paste this path into the **Start** command box to open the `engine` directory of the TERR installation in Spotfire.
Open the TERR Language Reference	Launch a web browser and display the Help landing page for TERR. From this page, you can access the language reference for the installed libraries and other technical documentation.
Launch TERR Console	Open the TERR console application in a separate window.
Launch RStudio IDE	Open an instance of RStudio that is configured to use the TERR engine installed with Spotfire.

Getting Help with TIBCO Enterprise Runtime for R

You can find help with the R language using the TERR language reference and other documentation directly from the Spotfire installation.

You can access help from the TERR console, or from Spotfire.

Procedure

1. From menu, click **Tools** > **TERR Tools**.
2. In the **TERR Tools** dialog box, click **Open TERR Language Reference**.
   A web browser opens to display the TERR landing page.
3. Optional: To find descriptions of data sets included in the Sdatasets package, click **Included Packages**, and then from the list, click **Sdatasets**.

Predictive Modeling

Your data and the kind of analysis you need to do determines the type of prediction to apply. Spotfire provides four types of predictive modeling tools.

Predictive Model	Description
Linear regression	Models the numeric response column as a weighted sum of the predictor columns. The weights, also known as the regression coefficients, are selected by the method of least squares, which minimizes the sum of the squared differences between the observed response and the predictions based on the weighted sum.
Regression tree	A nonparametric regression method that creates a binary tree by recursively splitting the data on the predictor values. The splits are selected so that the variation in the Response column is smaller in each child node than in the parent node. Various options are used to control how deep the tree is grown. Predictions are based on the mean of all the Response values in the terminal node for an observation.
Logistic regression	A classification method used when the Response column is categorical with only two possible values. The probability of the possible outcomes is modeled with a logistic transformation as a weighted sum of the Predictor columns. The weights or regression coefficients are selected to maximize the likelihood of the observed data.
Classification tree	A nonparametric classification method that creates a binary tree by recursively splitting the data on the predictor values. The splits are selected so that the variation in the Response column is smaller in each child mode than in the parent node. Various options are used to control how deep the tree is grown. Predictions are based on the mean of all the Response values in the terminal node for an observation.

Building a Regression Model in Spotfire

After you have determined which of the four Spotfire predictive models best suits your data and your desired analysis, use the predictive modeling options available to you from the **Tools** menu. This example task creates a regression model using sample data.

From Spotfire, you can build a predictive model that calls TERR for the statistical analysis. In this walkthrough task, build a linear regression model using the Spotfire predictive modeling tools.

For demonstration purposes, use the Baseball Player Statistics data example, available from the Spotfire Library, in `Demo/Analysis Files/Baseball`. Open the example DXP. (Optionally, use your own suitable data set.)

Prerequisites

A Spotfire license for advanced analytics.

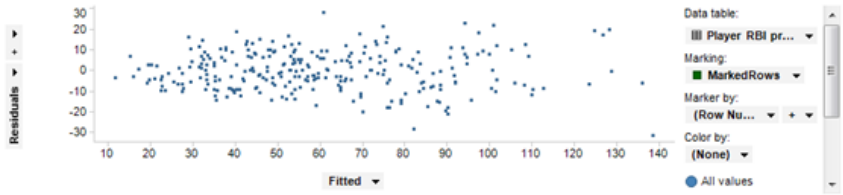
Procedure

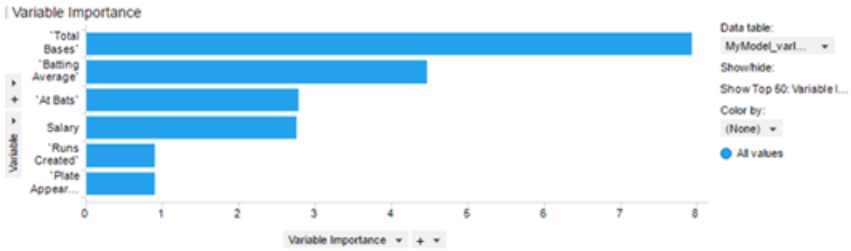
1. From the menu, click **Tools** > **Regression Modeling**.
2. Provide a suitable name and descriptive comment for the model.
3. From the **Model method** drop-down list, select **Linear Regression**, and then specify a **Data table**.
   For the example, specify **Baseball**.
4. For the **Response column**, select the response that you want to predict.
   For the example, to predict RBI (Runs Batted In), select RBI from the list.
5. From the **Predictor columns** box, select all of the variables to consider.
   You can select anything that is not a string. Select multiple predictor columns by holding down the control key as you click each one, or click **Add** for each predictor column you select.
   As you click **Add**, the predictor columns are added to the **Formula expression**. For the example, the formula expression to model for this example is as follows.

   Formula expression:

   ```
   `RBI` ~ `At Bats` + `Batting Average` + `Salary` + `Total Bases` + `Runs Created` + `Plate Appearences`
   ```

6. When you have the formula expression for the model, click **OK** to send the model specification to TERR and create the model.
   Spotfire displays the Model page based on your selections.
7. Review the Model page.

Display	Description
Model Summary	Provides the summary statistics appropriate for the particular model type. These statistics can give an indication of how well the model fits the data. It also displays an icon toolbar (), which you can use to edit the model, to create an evaluation model, to predict from the model, or to duplicate the model to manipulate.
Table of Coefficients	Provides the estimates of the coefficients, a measure of the variability or error of each estimate, and a test statistic (`t.value` or `z.value`) of the null hypothesis that the coefficients is zero (in other words, not needed in the model). It also provides a p-value for the statistical test.
Residuals vs. Fitted	Shows the residuals on the Y-axis and the fitted values on the X-axis. Values that have the residual 0 are those that would end up directly on the estimated regression surface. The residuals vs fit plot is commonly used to detect non-linearity, unequal error variances and outliers. When a linear regression model is suitable for a data set, then the residuals are more or less randomly distributed around the 0 line. The formula created in Spotfire creates the following pattern: 

Display	Description
Variable Importance	Shows a summary of the variables that are most relevant for determining the outline. If any of the variables has a very small relevancy, you might want to remove it from the model and rerun the analysis. 

8. Try duplicating the model and editing the copy to produce different results with different predictor columns.

What to do next

You can create an evaluation model, you can predict from the model, or you can export the model to share with others. See the Spotfire help for more information.

Sharing a Model

After you have built a predictive model in Spotfire, you can export it to a file on your desktop or on a network, or you can export it to the Spotfire library for use by others who have similar datasets.

The Analytic Models pane option is a toggle: To display it, or to hide it, click the option in the **View** menu. In this task, the Analytic Models pane is not yet displayed.

Prerequisites

You have a predictive model to share with others.

Procedure

1. On the menu, click **View** > **Analytic Models**.
2. From the Analytic Models pane, select the model to export, and then click the Export icon ( ) for that model.
   The Save As dialog box is displayed, and you are prompted to save the file as a Spotfire Analytic Model File (`.rds`).
3. Name the file, and then save it to the location of your choice.
   You can now share this analytic model with others, who can use it in various ways:
   - In TERR, running under a different application such as Streambase®.
   - In handling live events or in TIBCO BusinessEvents®.
   - In TERR running under TIBCO Spotfire® Statistics Services.
   - In another DXP file. For example, you could use this particular model with previous or subsequent years of baseball records.

Evaluating a Model

After you have created a predictive model in Spotfire, you can evaluate the model against data containing similar values to predict.

Perform this task in Spotfire, from the Model Summary pane.

Procedure

1. Click the icon for evaluating the model ( ).
2. In the Evaluate Analytic Model dialog box, specify the data table for which to apply the data, and then match the response columns and the predictor columns. Click **OK**.

   Typically, you evaluate a model against another data table that includes the values you are trying to predict using the model. For example, you can compare sales figures from data captured from month to month.

   Spotfire displays the evaluation.
3. Review the panes Evaluation Summary and Residuals vs. Predicted, and then apply any additional available diagnostic visualizations you want to review.

   For a list of available diagnostic visualizations, see the Spotfire help.

Expression Functions

You can call a TERR script directly from the Spotfire expression language. Using a TERR expression function, you can perform an ad hoc analysis quickly and easily.

For ad-hoc expressions, you can embed the contents of a script directly in your expression language call. Alternatively, you can write a TERR script, and then register it as an expression function for use anytime from the Custom Expression dialog box in Spotfire.

After you have created an expression function, you can reopen it, edit it, and rerun it.

Built-In TERR Expression Functions in Spotfire

In the Spotfire Custom Expressions or Insert Calculated Column dialog box, from the **Function** list, you can select one of the built-in TERR expression functions provided for advanced statistical analysis.

Each function invokes the TERR engine. The results depend on the expression function you select.

- For nonaggregated functions, the TERR script should set the variable `output` to a vector or a one-column data frame, which is the same length as the input column(s), of the specified TERR data type. Spotfire converts the data type to the corresponding Spotfire data type.
- For aggregated functions, the TERR script should set the variable `output` to a scalar value of the specified TERR data type, which Spotfire converts to the corresponding Spotfire data type. The TERR script is called once for each group of data to be aggregated. No special handling for aggregation is necessary in your TERR script.

Spotfire and TERR use the following naming convention for defining expression functions.

Expression function component	Description	Example
`TERR_<SpotfireDataType>` or `TERRAggregation_<SpotfireDataType>`	The built-in expression function.	`TERR_Real("output <- input1/input2", [Mileage], [Fuel])` The TERR returned value (specified by `output` is a vector of data type `numeric`, which Spotfire converts to a column of the Spotfire data type Real.

Expression function component	Description	Example
`input1`, `input2`, `input3`... `inputN` `[Colname]`, `[Colname]`, `[Colname]`, `[Colname]`	Each input (`inputN`) is a parameter that is passed in to the TERR expression function, and that corresponds to a column name (Colname) in the same numerical order. The column names corresponding to the inputs are added as the last component of the expression, immediately after the calculation's closing quotation marks and comma, and immediately before the closing parenthesis. Each column name is enclosed in square brackets. Column names are separated by commas.	`TERR_Real("output <- input1/input2", [Mileage], [Fuel])` • `[Mileage]` corresponds to `input1`. • `[Fuel]` corresponds to `input2`. The calculation specifies that, for each row, the value in the column `Mileage` is divided by the value for the column `Fuel`.
`output`	The returned value of the calculation in TERR. It can be added to the Spotfire visualization as a column or a single aggregated value.	`TERR_Real("output <- input1/input2", [Mileage], [Fuel])` `output` represents the column of `Real` values returned containing the calculation for each row of the mileage divided by the fuel.

Select one of the topics for the available built-in expression functions to learn more.

TERR_Binary

In the Spotfire Custom Expressions dialog box, you can select the pre-defined expression `TERR_Binary` from the **Function** list. This expression function invokes the TERR engine to return a vector or a single column data frame of the data type `raw`, which is converted to a Spotfire column of the corresponding Spotfire data type Binary.

The expression function has at least two arguments.

Argument	Argument description
A TERR script.	The TERR script contains the following. • A number of variables using the naming convention that Spotfire requires: `input1` to `inputN`, where `inputN` is the highest number of the specified inputs, numbered sequentially. • A TERR assignment operator (`<-`) that assigns the results of the TERR evaluation to an object named `output` (also using the naming convention that Spotfire requires).
Spotfire column names.	Passed as additional arguments, these are the data column names that `input1` to `inputN` represent. All columns must be the same length.

The output type is returned from TERR and converted by Spotfire.

Returned by TERR	Converted in Spotfire
A vector or a single column data frame of data type `raw`.	A column with the same number of rows as the input, and of the data type Binary.

TERR_Binary example

In this example, in Spotfire, create a one-column table called Measure containing the numbers between 0 and 5. Next, create a new column called Outliers, which displays the following, based on the number in the Measure column.

- If the number is below 3.5, display nothing.
- If the number is between 3.5 and 4.5, display a binary image of a small red question mark.
- If the number is over 4.5, display a binary image of a small red exclamation mark.

```
TERR_Binary("marks <- list(
    ok = NULL,
    # question mark
    hmm = as.raw(c(0x89, 0x50, 0x4e, 0x47, 0x0d, 0x0a, 0x1a,
 0x0a, 0x00,
          0x00, 0x00, 0x0d, 0x49, 0x48, 0x44, 0x52, 0x00, 0x00,
 0x00, 0x10,
          0x00, 0x00, 0x00, 0x10, 0x08, 0x03, 0x00, 0x00, 0x00,
 0x28, 0x2d,
          0x0f, 0x53, 0x00, 0x00, 0x00, 0x27, 0x50, 0x4c, 0x54,
 0x45, 0xff,
          0x00, 0x00, 0xff, 0x00, 0x66, 0xff, 0x3a, 0x00, 0xff,
 0x3a, 0x90,
          0xff, 0x66, 0x00, 0xff, 0x66, 0xb6, 0xff, 0x90, 0x3a,
 0xff, 0x90,
          0xdb, 0xff, 0xb6, 0x66, 0xff, 0xb6, 0xff, 0xff, 0xff,
 0xb6, 0xff,
          0xff, 0xdb, 0xff, 0xff, 0xff, 0xbb, 0xaf, 0xf5, 0x2c,
 0x00, 0x00,
          0x00, 0x3f, 0x49, 0x44, 0x41, 0x54, 0x18, 0x95, 0x63,
 0xe0, 0x41,
          0x03, 0x0c, 0x64, 0x0a, 0x70, 0x31, 0x30, 0x30, 0x30,
 0x22, 0x09,
          0x70, 0x33, 0xb1, 0xf2, 0xf0, 0xb0, 0x31, 0x22, 0x04,
 0x38, 0x18,
          0xc1, 0xa2, 0x9c, 0xa8, 0x66, 0x70, 0x31, 0xa0, 0x0a,
 0x70, 0x31,
          0xb0, 0xa2, 0xd8, 0x02, 0xe7, 0xc3, 0x04, 0xd8, 0x98,
 0xd1, 0xdc,
          0xc1, 0xc2, 0x4e, 0xc8, 0xa5, 0x84, 0x55, 0x20, 0x00,
 0x00, 0xe8,
          0x14, 0x0b, 0x2c, 0x7d, 0xc8, 0x15, 0x53, 0x00, 0x00,
 0x00, 0x00,
          0x49, 0x45, 0x4e, 0x44, 0xae, 0x42, 0x60, 0x82)),
    # exclamation mark
    oops = as.raw(c(0x89, 0x50, 0x4e, 0x47, 0x0d, 0x0a, 0x1a,
 0x0a, 0x00,
           0x00, 0x00, 0x0d, 0x49, 0x48, 0x44, 0x52, 0x00, 0x00,
 0x00, 0x10,
           0x00, 0x00, 0x00, 0x10, 0x08, 0x03, 0x00, 0x00, 0x00,
 0x28, 0x2d,
           0x0f, 0x53, 0x00, 0x00, 0x00, 0x21, 0x50, 0x4c, 0x54,
 0x45, 0xff,
           0x00, 0x00, 0xff, 0x00, 0x3a, 0xff, 0x00, 0x66, 0xff,
 0x3a, 0x66,
           0xff, 0xb6, 0x66, 0xff, 0xb6, 0xff, 0xff, 0xdb, 0x90,
 0xff, 0xdb,
           0xff, 0xff, 0xff, 0xb6, 0xff, 0xff, 0xdb, 0xff, 0xff,
 0xff, 0x10,
           0x12, 0xb4, 0x48, 0x00, 0x00, 0x00, 0x28, 0x49, 0x44,
 0x41, 0x54,
           0x18, 0x95, 0x63, 0xe0, 0x42, 0x03, 0x0c, 0xe4, 0x0a,
 0xb0, 0x30,
           0xb0, 0x12, 0x10, 0x60, 0xc3, 0x10, 0x60, 0x64, 0x27,
 0x20, 0xc0,
           0x81, 0x21, 0xc0, 0x84, 0x66, 0x2d, 0x27, 0x33, 0xc9,
 0x0e, 0xc3,
           0xe7, 0x17, 0x00, 0x05, 0x3d, 0x09, 0x6f, 0xa9, 0xc3,
 0xf3, 0x36,
           0x00, 0x00, 0x00, 0x00, 0x49, 0x45, 0x4e, 0x44, 0xae,
 0x42, 0x60,
```

```
            0x82)))
   # Use first item in input2 and input3 because they have been
extended to length of input1
   whichMark <- findInterval(input1, c(-Inf, input2[1],
input3[1]))
   # Wrap output list in I() so it remains one column in
data.frame
   output <- I(marks[whichMark])", [Measure], 3.5, 4.5)
```

The resulting table in Spotfire shows the following.

Measure	Outliers
0	
1	
2	
3	
4	?
5	!

Spotfire autocorrects the function case or name to that of built-in Spotfire function names. (For example, TERR contains the function `max`, and Spotfire contains the function `Max`.) You must overwrite this autocorrection manually to ensure that you use the TERR function case and name in your expression function.

See Embedding the contents of a script in an expression function for a detailed procedure for creating an expression function.

TERR_Boolean

In the Spotfire Custom Expressions dialog box, you can select the pre-defined expression `TERR_Boolean` from the **Function** list. This expression function invokes the TERR engine to return a vector or a single column data frame of the data type `logical` (`TRUE` or `FALSE`), which is converted to a Spotfire column of the corresponding Spotfire data type Boolean.

The expression function has at least two arguments.

Argument	Argument description
A TERR script.	The TERR script contains the following. • A number of variables using the naming convention that Spotfire requires: `input1` to `inputN`, where `inputN` is the highest number of the specified inputs, numbered sequentially. • A TERR assignment operator (`<-`) that assigns the results of the TERR evaluation to an object named `output` (also using the naming convention that Spotfire requires).
Spotfire column names.	Passed as additional arguments, these are the data column names that `input1` to `inputN` represent. All columns must be the same length.

The output type is returned from TERR and converted by Spotfire.

Returned by TERR	Converted in Spotfire
A vector or a single column data frame of data type `logical`.	A column with the same number of rows as the input, and of the data type Boolean.

TERR_Boolean example

This example uses the Sales and Marketing visualization in the Spotfire example library. The column created by the example is used as the **Color by** control.

```
TERR_Boolean("output <- input1 > .10",[Class Change Yr 1 to Yr
 2])
```

The resulting visualization in Spotfire shows the following.

Spotfire autocorrects the function case or name to that of built-in Spotfire function names. (For example, TERR contains the function `max`, and Spotfire contains the function `Max`.) You must overwrite this autocorrection manually to ensure that you use the TERR function case and name in your expression function.

See Embedding the contents of a script in an expression function for a detailed procedure for creating an expression function.

TERR_DateTime

In the Spotfire Custom Expressions dialog box, you can select the pre-defined expression `TERR_DateTime` from the **Function** list. This expression function invokes the TERR engine to return a vector or a single column data frame of the data type `POSIXct` or `POSIXlt`, which is converted to a Spotfire column of the corresponding Spotfire data type DateTime.

The expression function has at least two arguments.

Argument	Argument description
A TERR script.	The TERR script contains the following. • A number of variables using the naming convention that Spotfire requires: `input1` to `inputN`, where `inputN` is the highest number of the specified inputs, numbered sequentially. • A TERR assignment operator (`<-`) that assigns the results of the TERR evaluation to an object named `output` (also using the naming convention that Spotfire requires).
Spotfire column names.	Passed as additional arguments, these are the data column names that `input1` to `inputN` represent. All columns must be the same length.

The output type is returned from TERR and converted by Spotfire.

Returned by TERR	Converted in Spotfire
A vector or a single column data frame of data type `POSIXct` or `POSIXlt` with the time zone as UTC. Any data specified as NA in TERR maps to null in Spotfire. Do not use the `date` data type. (Spotfire converts `date` to data type Real.)	A column with the same number of rows as the input, and of the data type DateTime.

TERR_DateTime

In this example, in Spotfire, create a column that has the current date and time, and then create a calculated column to truncate the seconds.

```
TERR_DateTime("output <- trunc(input1,'mins')",[DateTimeNow()])
```

The resulting table in Spotfire shows the following.

DateTimeNow()	Show no seconds
7/19/2017 3:22:20 PM	7/19/2017 3:22:00 PM
7/19/2017 3:22:20 PM	7/19/2017 3:22:00 PM
7/19/2017 3:22:20 PM	7/19/2017 3:22:00 PM
7/19/2017 3:22:20 PM	7/19/2017 3:22:00 PM
7/19/2017 3:22:20 PM	7/19/2017 3:22:00 PM
7/19/2017 3:22:20 PM	7/19/2017 3:22:00 PM
7/19/2017 3:22:20 PM	7/19/2017 3:22:00 PM

`DateTimeNow()` is a built-in Spotfire function.

Spotfire autocorrects the function case or name to that of built-in Spotfire function names. (For example, TERR contains the function `max`, and Spotfire contains the function `Max`.) You must overwrite this autocorrection manually to ensure that you use the TERR function case and name in your expression function.

See Embedding the contents of a script in an expression function for a detailed procedure for creating an expression function.

TERR_Integer

In the Spotfire Custom Expressions dialog box, you can select the pre-defined expression `TERR_Integer` from the **Function** list. This expression function invokes the TERR engine to return a vector or a single column data frame of the data type `integer` , which is converted to a Spotfire column of the corresponding Spotfire data type Integer.

The expression function has at least two arguments.

Argument	Argument description
A TERR script.	The TERR script contains the following. • A number of variables using the naming convention that Spotfire requires: `input1` to `inputN`, where `inputN` is the highest number of the specified inputs, numbered sequentially. • A TERR assignment operator (`<-`) that assigns the results of the TERR evaluation to an object named `output` (also using the naming convention that Spotfire requires).
Spotfire column names.	Passed as additional arguments, these are the data column names that `input1` to `inputN` represent. All columns must be the same length.

The output type is returned from TERR and converted by Spotfire.

Returned by TERR	Converted in Spotfire
A vector or a single column data frame of data type `integer`.	A column with the same number of rows as the input, and of the data type Integer.

TERR_Integer example

This example data set gives information on the makes of cars taken from the April, 1990 issue of Consumer Reports (pages 235-255). This data set contains 6 columns for 61 cars (rows). You can find the sample data set in Car data set for Spotfire examples.

You can copy (CTRL+C) the contents of the sample data table and paste it (CTRL+V) into the Spotfire user interface.

In the expression function, multiply the columns Disp. and Mileage to create a new column.

```
TERR_Integer("output <- as.integer(input1*input2)",[Disp.],
[Mileage])
```

In the case where the output from the calculation might not be an integer, cast them as an integer by wrapping the expression in the TERR function `as.integer`.

The resulting table in Spotfire shows the following.

Column 1	Weight	Disp.	Mileage	Fuel	Disp times m...
Eagle Summit 4	2560	97	33	3.03	3201
Ford Escort 4	2345	114	33	3.03	3762
Ford Festiva 4	1845	81	37	2.70	2997
Honda Civic 4	2260	91	32	3.13	2912
Mazda Protege 4	2440	113	32	3.13	3616
Mercury Trace...	2285	97	26	3.85	2522
Nissan Sentra 4	2275	97	33	3.03	3201
Pontiac LeMan...	2350	98	28	3.57	2744
Subaru Loyale 4	2295	109	25	4.00	2725
Subaru Justy 3	1900	73	34	2.94	2482
Toyota Corolla 4	2390	97	29	3.45	2813
Toyota Tercel 4	2075	89	35	2.86	3115
Volkswagen J...	2330	109	26	3.85	2834
Chevrolet Ca...	3320	305	20	5.00	6100
Dodge Daytona	2885	153	27	3.70	4131

Spotfire autocorrects the function case or name to that of built-in Spotfire function names. (For example, TERR contains the function `max`, and Spotfire contains the function `Max`.) You must overwrite this autocorrection manually to ensure that you use the TERR function case and name in your expression function.

See Embedding the contents of a script in an expression function for a detailed procedure for creating an expression function.

TERR_Real

In the Spotfire Custom Expressions dialog box, you can select the pre-defined expression `TERR_Real` from the **Function** list. This expression function invokes the TERR engine to return a vector or a single column data frame of the data type `numeric` , which is converted to a Spotfire column of the corresponding Spotfire data type Real.

The expression function has at least two arguments.

Argument	Argument description
A TERR script.	The TERR script contains the following. • A number of variables using the naming convention that Spotfire requires: `input1` to `inputN`, where `inputN` is the highest number of the specified inputs, numbered sequentially. • A TERR assignment operator (`<-`) that assigns the results of the TERR evaluation to an object named `output` (also using the naming convention that Spotfire requires).
Spotfire column names.	Passed as additional arguments, these are the data column names that `input1` to `inputN` represent. All columns must be the same length.

The output type is returned from TERR and converted by Spotfire.

Returned by TERR	Converted in Spotfire
A vector or a single column data frame of data type `numeric`.	A column with the same number of rows as the input, and of the data type Real.

TERR_Real example

This example uses the TERR_PCA_Cars data sample from the Spotfire library. Select four columns (Name, Type, RetailPrice, and DealerCost), and then add a column that calculates a minimum of 5% profit above dealer cost.

```
TERR_Real("output <- input1*1.05",[DealerCost]))
```

The resulting table in Spotfire shows the following.

Name	Type	DealerCost	RetailPrice	5% above dealer
Chevrolet Aveo 4dr	Sedan	10965	11690	11513.25
Chevrolet Aveo LS 4dr hatch	Sedan	11802	12585	12392.10
Chevrolet Cavalier 2dr	Sedan	13697	14610	14381.85
Chevrolet Cavalier 4dr	Sedan	13884	14810	14578.20
Chevrolet Cavalier LS 2dr	Sedan	15357	16385	16124.85
Dodge Neon SE 4dr	Sedan	12849	13670	13491.45
Dodge Neon SXT 4dr	Sedan	14086	15040	14790.30
Ford Focus ZX3 2dr hatch	Sedan	12482	13270	13106.10
Ford Focus LX 4dr	Sedan	12906	13730	13551.30
Ford Focus SE 4dr	Sedan	14496	15460	15220.80
Ford Focus ZX5 5dr	Sedan	14607	15580	15337.35
Honda Civic DX 2dr	Sedan	12175	13270	12783.75
Honda Civic HX 2dr	Sedan	12996	14170	13645.80
Honda Civic LX 4dr	Sedan	14531	15850	15257.55
Hyundai Accent 2dr hatch	Sedan	10107	10539	10612.35
Hyundai Accent GL 4dr	Sedan	11116	11839	11671.80
Hyundai Accent GT 2dr	Sedan	11209	11939	11769.45

Spotfire autocorrects the function case or name to that of built-in Spotfire function names. (For example, TERR contains the function `max`, and Spotfire contains the function `Max`.) You must overwrite this autocorrection manually to ensure that you use the TERR function case and name in your expression function.

See Embedding the contents of a script in an expression function for a detailed procedure for creating an expression function.

TERR_String

In the Spotfire Custom Expressions dialog box, you can select the pre-defined expression `TERR_String` from the **Function** list. This expression function invokes the TERR engine to return a vector or a single column data frame of the data type `character` , which is converted to a Spotfire column of the corresponding Spotfire data type String data type.

The expression function has at least two arguments.

Argument	Argument description
A TERR script.	The TERR script contains the following. • A number of variables using the naming convention that Spotfire requires: `input1` to `inputN`, where `inputN` is the highest number of the specified inputs, numbered sequentially. • A TERR assignment operator (`<-`) that assigns the results of the TERR evaluation to an object named `output` (also using the naming convention that Spotfire requires).
Spotfire column names.	Passed as additional arguments, these are the data column names that `input1` to `inputN` represent. All columns must be the same length.

The output type is returned from TERR and converted by Spotfire.

Returned by TERR	Converted in Spotfire
A vector or a single column data frame of data type `character`. The returned output must be encoded as UTF-8.	A column with the same number of rows as the input, and of the data type String.

TERR_String example

This example data set gives information on the makes of cars taken from the April, 1990 issue of Consumer Reports (pages 235-255). This data set contains 6 columns for 61 cars (rows). You can find the sample data set in Car data set for Spotfire examples.

You can copy (CTRL+C) the contents of the sample data table and paste it (CTRL+V) into the Spotfire user interface.

In the expression function, create a new column that adds the character string "ish" to the end of each entry from Type.

```
TERR_String("output <- paste(input1,'ish',sep='')",[Type])
```

The resulting table in Spotfire shows the following.

Column 1	Mileage	Type	New Type
Eagle Summit 4	33	Small	Smallish
Ford Escort 4	33	Small	Smallish
Ford Festiva 4	37	Small	Smallish
Honda Civic 4	32	Small	Smallish
Mazda Protege 4	32	Small	Smallish
Mercury Trace...	26	Small	Smallish
Nissan Sentra 4	33	Small	Smallish
Pontiac LeMan...	28	Small	Smallish
Subaru Loyale 4	25	Small	Smallish
Subaru Justy 3	34	Small	Smallish
Toyota Corolla 4	29	Small	Smallish
Toyota Tercel 4	35	Small	Smallish
Volkswagen J...	26	Small	Smallish
Chevrolet Ca...	20	Sporty	Sportyish
Dodge Daytona	27	Sporty	Sportyish
Ford Mustang V8	19	Sporty	Sportyish
Ford Probe	30	Sporty	Sportyish
Honda Civic C...	33	Sporty	Sportyish
Honda Prelude...	27	Sporty	Sportyish
Nissan 240SX 4	24	Sporty	Sportyish
Plymouth Laser	26	Sporty	Sportyish
Subaru XT 4	28	Sporty	Sportyish
Audi 80 4	27	Compact	Compactish
Buick Skylark 4	23	Compact	Compactish
Chevrolet Ber...	26	Compact	Compactish
Chrysler Le B...	25	Compact	Compactish
Ford Tempo 4	24	Compact	Compactish
Honda Accord 4	26	Compact	Compactish

Spotfire autocorrects the function case or name to that of built-in Spotfire function names. (For example, TERR contains the function `max`, and Spotfire contains the function `Max`.) You must overwrite this autocorrection manually to ensure that you use the TERR function case and name in your expression function.

See Embedding the contents of a script in an expression function for a detailed procedure for creating an expression function.

TERR_TimeSpan

In the Spotfire Custom Expressions dialog box, you can select the pre-defined expression `TERR_TimeSpan` from the **Function** list. This expression function invokes the TERR engine to return a vector or a single column data frame of the data type `difftime` , which is converted to a Spotfire column of the corresponding Spotfire data type TimeSpan.

The expression function has at least two arguments.

Argument	Argument description
A TERR script.	The TERR script contains the following. • A number of variables using the naming convention that Spotfire requires: `input1` to `inputN`, where `inputN` is the highest number of the specified inputs, numbered sequentally. • A TERR assignment operator (`<-`) that assigns the results of the TERR evaluation to an object named `output` (also using the naming convention that Spotfire requires).
Spotfire column names.	Passed as additional arguments, these are the data column names that `input1` to `inputN` represent. All columns must be the same length.

The output type is returned from TERR and converted by Spotfire.

Returned by TERR	Converted in Spotfire
A vector or a single column data frame of data type `difftime`.	A column with the same number of rows as the input, and of the data type TimeSpan.

TERR_TimeSpan example

In this example, in Spotfire, take thirty observations with a start time and an end time. You can find the sample data set in Observation data set for Spotfire examples.

You can copy (CTRL+C) the contents of the sample data table and paste it (CTRL+V) into the Spotfire user interface.

Calculate the observation time for each entry.

```
TERR_TimeSpan("output <- difftime(input1, input2)",[End time],
[Start time])
```

In the example, we named the resulting column `Elapsed time` and added the column to a table visualization. The resulting table in Spotfire shows the following.

Date	Start time	End time	Observation	Elapsed time
3/6/2017	1:57:21 PM	2:16:14 PM	Positive	18:53.000
3/7/2017	1:58:26 PM	2:18:22 PM	Positive	19:56.000
3/8/2017	2:02:37 PM	2:20:54 PM	Negative	18:17.000
3/9/2017	2:03:36 PM	2:15:42 PM	Negative	12:06.000
3/10/2017	1:58:45 PM	2:23:46 PM	Positive	25:01.000
3/11/2017	1:55:45 PM	2:21:39 PM	Negative	25:54.000
3/12/2017	1:58:42 PM	2:17:21 PM	Negative	18:39.000
3/13/2017	2:01:27 PM	2:25:33 PM	Negative	24:06.000
3/14/2017	2:02:48 PM	2:17:27 PM	Positive	14:39.000
3/15/2017	2:04:47 PM	2:22:26 PM	Negative	17:39.000
3/16/2017	2:00:32 PM	2:20:43 PM	Negative	20:11.000
3/17/2017	2:03:16 PM	2:16:25 PM	Positive	13:09.000
3/18/2017	2:06:18 PM	2:25:52 PM	Negative	19:34.000
3/19/2017	2:01:48 PM	2:21:46 PM	Positive	19:58.000
3/20/2017	1:54:37 PM	2:15:48 PM	Negative	21:11.000
3/21/2017	1:56:38 PM	2:25:12 PM	Negative	28:34.000
3/22/2017	1:57:52 PM	2:23:16 PM	Positive	25:24.000
3/23/2017	2:04:22 PM	2:23:53 PM	Positive	19:31.000
3/24/2017	2:02:37 PM	2:18:19 PM	Positive	15:42.000
3/25/2017	1:55:52 PM	2:21:24 PM	Negative	25:32.000
3/26/2017	1:58:37 PM	2:17:28 PM	Positive	18:51.000
3/27/2017	2:01:47 PM	2:25:16 PM	Negative	23:29.000
3/28/2017	2:02:45 PM	2:17:55 PM	Positive	15:10.000
3/29/2017	2:04:29 PM	2:22:58 PM	Negative	18:29.000
3/30/2017	1:55:01 PM	2:17:22 PM	Positive	22:21.000
3/31/2017	1:54:50 PM	2:19:45 PM	Positive	24:55.000

Spotfire autocorrects the function case or name to that of built-in Spotfire function names. (For example, TERR contains the function `max`, and Spotfire contains the function `Max`.) You must overwrite this autocorrection manually to ensure that you use the TERR function case and name in your expression function.

See Embedding the contents of a script in an expression function for a detailed procedure for creating an expression function.

TERRAggregation_Binary

In the Spotfire Custom Expressions dialog box, you can select the pre-defined expression `TERRAggregation_Binary` from the **Function** list. This expression function sets the variable `output` to a scalar value of the TERR data type `raw`, which Spotfire converts to the corresponding Spotfire data type Binary. The TERR script is called once for each group of data to be aggregated. No special handling for aggregation is necessary in your TERR script.

The expression function has at least two arguments.

Argument	Argument description
A TERR script.	The TERR script contains the following. • A number of variables using the naming convention that Spotfire requires: `input1` to `inputN`, where `inputN` is the highest number of the specified inputs, numbered sequentially. • A TERR assignment operator (`<-`) that assigns the results of the TERR evaluation to an object named `output` (also using the naming convention that Spotfire requires).
Spotfire column names.	Passed as additional arguments, these are the data column names that `input1` to `inputN` represent. All columns must be the same length.

The output type is returned from TERR and converted by Spotfire.

Returned by TERR	Converted in Spotfire
A single aggregated value of data type `raw`.	A single aggregated value of data type Binary.

TERRAggregation_Binary example

```
TERRAggregation_Binary("output <- input2[input1==max(input1)]",
 [1])
```

Spotfire autocorrects the function case or name to that of built-in Spotfire function names. (For example, TERR contains the function `max`, and Spotfire contains the function `Max`.) You must overwrite this autocorrection manually to ensure that you use the TERR function case and name in your expression function.

See Embedding the contents of a script in an expression function for a detailed procedure for creating an expression function.

TERRAggregation_Boolean

In the Spotfire Custom Expressions dialog box, you can select the pre-defined expression `TERRAggregation_Boolean` from the **Function** list. This expression function sets the variable `output` to a scalar value of the TERR data type `logical` (`TRUE` or `FALSE`), which Spotfire converts to the corresponding Spotfire data type Boolean. The TERR script is called once for each group of data to be aggregated. No special handling for aggregation is necessary in your TERR script.

The expression function has at least two arguments.

Argument	Argument description
A TERR script.	The TERR script contains the following. • A number of variables using the naming convention that Spotfire requires: `input1` to `inputN`, where `inputN` is the highest number of the specified inputs, numbered sequentially. • A TERR assignment operator (`<-`) that assigns the results of the TERR evaluation to an object named `output` (also using the naming convention that Spotfire requires).
Spotfire column names.	Passed as additional arguments, these are the data column names that `input1` to `inputN` represent. All columns must be the same length.

The output type is returned from TERR and converted by Spotfire.

Returned by TERR	Converted in Spotfire
A single aggregated value of data type `logical`.	A single aggregated value of data type Boolean.

TERRAggregation_Boolean example

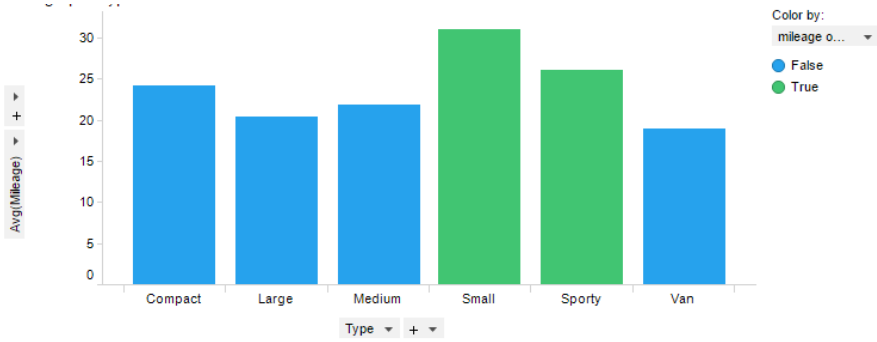
This example data set gives information on the makes of cars taken from the April, 1990 issue of Consumer Reports (pages 235-255). This data set contains 6 columns for 61 cars (rows). You can find the sample data set in Car data set for Spotfire examples.

You can copy (CTRL+C) the contents of the sample data table and paste it (CTRL+V) into the Spotfire user interface.

In the expression function, determine which types of cars have an aggregated value of mileage over 30.

```
TERRAggregation_Boolean("output <- any(input1 > 30)",[Mileage])
 OVER ([Type])
```

The resulting bar chart in Spotfire, with **Color by** set to the aggregation results, shows the following.



Spotfire autocorrects the function case or name to that of built-in Spotfire function names. (For example, TERR contains the function `max`, and Spotfire contains the function `Max`.) You must overwrite this autocorrection manually to ensure that you use the TERR function case and name in your expression function.

See Embedding the contents of a script in an expression function for a detailed procedure for creating an expression function.

TERRAggregation_DateTime

In the Spotfire Custom Expressions dialog box, you can select the pre-defined expression `TERRAggregation_DateTime` from the **Function** list. This expression function sets the variable `output` to a scalar value of the TERR data type `POSIXct` or `POSIXlt`, which Spotfire converts to the corresponding Spotfire data type DateTime. The TERR script is called once for each group of data to be aggregated. No special handling for aggregation is necessary in your TERR script.

The expression function has at least two arguments.

Argument	Argument description
A TERR script.	The TERR script contains the following. • A number of variables using the naming convention that Spotfire requires: `input1` to `inputN`, where `inputN` is the highest number of the specified inputs, numbered sequentially. • A TERR assignment operator (`<-`) that assigns the results of the TERR evaluation to an object named `output` (also using the naming convention that Spotfire requires).
Spotfire column names.	Passed as additional arguments, these are the data column names that `input1` to `inputN` represent. All columns must be the same length.

The output type is returned from TERR and converted by Spotfire.

Returned by TERR	Converted in Spotfire
A single aggregated value of data type `POSIXct` or `POSIXlt` with the time zone as UTC. Any data specified as NA in TERR maps to null in Spotfire. Do not use the `date` data type. (Spotfire converts `date` to data type Real.)	A single aggregated value of data type DateTime.

TERRAggregation_DateTime

```
TERRAggregation_DateTime("output <-max(input1, 'mins')", [X])
```

Spotfire autocorrects the function case or name to that of built-in Spotfire function names. (For example, TERR contains the function `max`, and Spotfire contains the function `Max`.) You must overwrite this autocorrection manually to ensure that you use the TERR function case and name in your expression function.

See Embedding the contents of a script in an expression function for a detailed procedure for creating an expression function.

TERRAggregation_Integer

In the Spotfire Custom Expressions dialog box, you can select the pre-defined expression `TERRAggregation_Integer` from the **Function** list. This expression function sets the variable `output` to a scalar value of the TERR data type `integer`, which Spotfire converts to the corresponding Spotfire data type Integer. The TERR script is called once for each group of data to be aggregated. No special handling for aggregation is necessary in your TERR script.

The expression function has at least two arguments.

Argument	Argument description
A TERR script.	The TERR script contains the following. • A number of variables using the naming convention that Spotfire requires: `input1` to `inputN`, where `inputN` is the highest number of the specified inputs, numbered sequentially. • A TERR assignment operator (`<-`) that assigns the results of the TERR evaluation to an object named `output` (also using the naming convention that Spotfire requires).
Spotfire column names.	Passed as additional arguments, these are the data column names that `input1` to `inputN` represent. All columns must be the same length.

The output type is returned from TERR and converted by Spotfire.

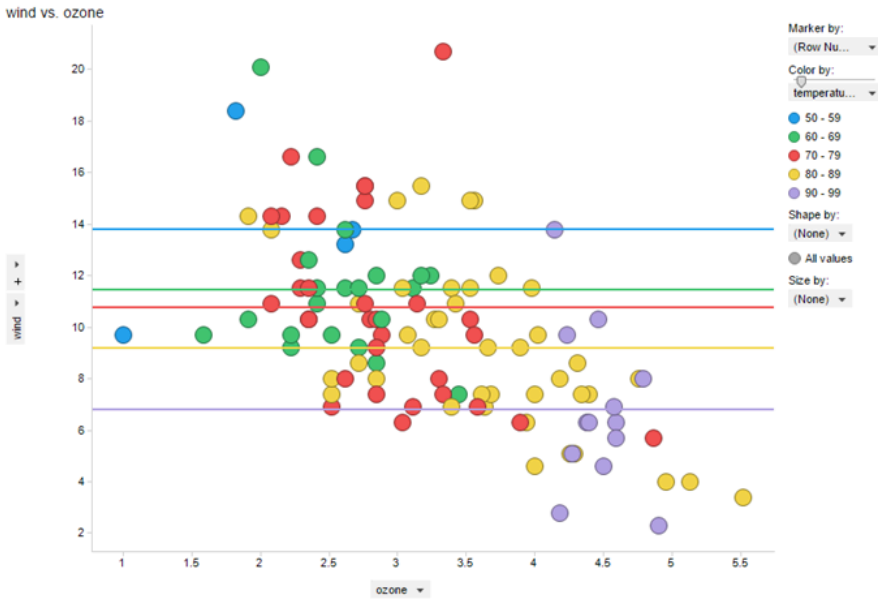
Returned by TERR	Converted in Spotfire
A single aggregated value of data type `integer`.	A single aggregated value of data type Integer.

TERRAggregation_Integer example

You can work through the example in Aggregating binned weather data using TERR in Spotfire, substituting `TERRAggregation_Integer` for `TERRAggregation_Real`. Make sure your inputs are integers, and that you wrap the expression in the TERR function `as.integer` to make sure the output data type is correct.

```
TERRAggregation_Integer("output <- as.integer(mean(input1))",
 [Y])
```

The resulting table in Spotfire shows the following.



Spotfire autocorrects the function case or name to that of built-in Spotfire function names. (For example, TERR contains the function `max`, and Spotfire contains the function `Max`.) You must overwrite this autocorrection manually to ensure that you use the TERR function case and name in your expression function.

See Embedding the contents of a script in an expression function for a detailed procedure for creating an expression function.

TERRAggregation_Real

In the Spotfire Custom Expressions dialog box, you can select the pre-defined expression `TERRAggregation_Real` from the **Function** list. This expression function sets the variable `output` to a scalar value of the TERR data type `numeric`, which Spotfire converts to the corresponding Spotfire data type Real. The TERR script is called once for each group of data to be aggregated. No special handling for aggregation is necessary in your TERR script.

The expression function has at least two arguments.

Argument	Argument description
A TERR script.	The TERR script contains the following. • A number of variables using the naming convention that Spotfire requires: `input1` to `inputN`, where `inputN` is the highest number of the specified inputs, numbered sequentially. • A TERR assignment operator (`<-`) that assigns the results of the TERR evaluation to an object named `output` (also using the naming convention that Spotfire requires).
Spotfire column names.	Passed as additional arguments, these are the data column names that `input1` to `inputN` represent. All columns must be the same length.

The output type is returned from TERR and converted by Spotfire.

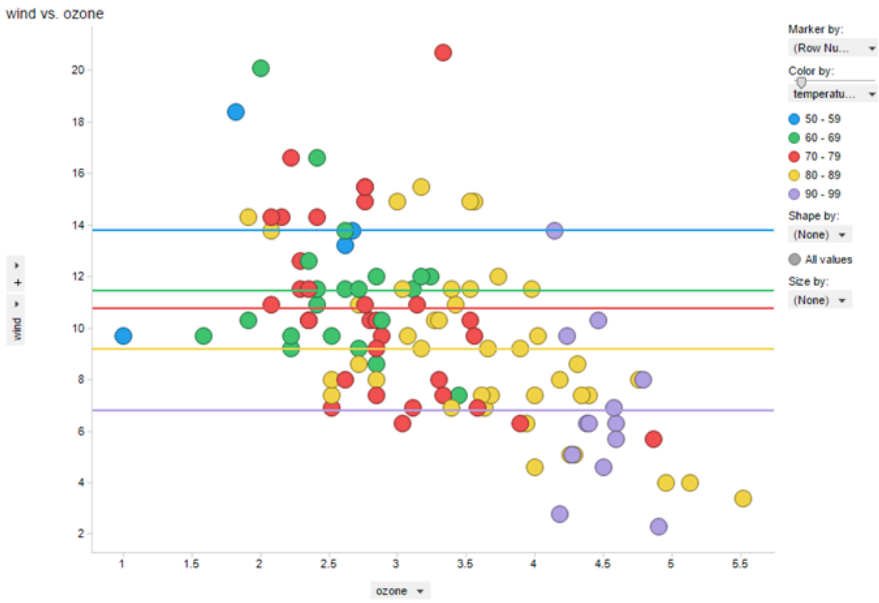
Returned by TERR	Converted in Spotfire
A single aggregated value of data type `numeric`.	A single aggregated value of data type Real.

TERRAggregation_Real example

You can work through the example in Aggregating binned weather data using TERR in Spotfire. That example adds a horizontal line on the the Y axis for data provided in air data set for Spotfire examples.

```
TERRAggregation_Real("output <- mean(input1)",[Y])
```

The resulting table in Spotfire shows the following.



TERRAggregation_String

In the Spotfire Custom Expressions dialog box, you can select the pre-defined expression `TERRAggregation_String` from the **Function** list. This expression function sets the variable `output` to a scalar value of the TERR data type `character`, which Spotfire converts to the corresponding Spotfire data type String. The TERR script is called once for each group of data to be aggregated. No special handling for aggregation is necessary in your TERR script.

The expression function has at least two arguments.

Argument	Argument description
A TERR script.	The TERR script contains the following. • A number of variables using the naming convention that Spotfire requires: `input1` to `inputN`, where `inputN` is the highest number of the specified inputs, numbered sequentially. • A TERR assignment operator (`<-`) that assigns the results of the TERR evaluation to an object named `output` (also using the naming convention that Spotfire requires).
Spotfire column names.	Passed as additional arguments, these are the data column names that `input1` to `inputN` represent. All columns must be the same length.

The output type is returned from TERR and converted by Spotfire.

Returned by TERR	Converted in Spotfire
A single aggregated value of data type `character`. The returned output must be encoded as UTF-8.	A single aggregated value of data type String.

TERRAggregation_String example

```
TERRAggregation_String("output <- input2[input1==max(input1)]",
 [1])
```

Spotfire autocorrects the function case or name to that of built-in Spotfire function names. (For example, TERR contains the function `max`, and Spotfire contains the function `Max`.) You must overwrite this autocorrection manually to ensure that you use the TERR function case and name in your expression function.

See Embedding the contents of a script in an expression function for a detailed procedure for creating an expression function.

TERRAggregation_TimeSpan

In the Spotfire Custom Expressions dialog box, you can select the pre-defined expression `TERRAggregation_TimeSpan` from the **Function** list. This expression function sets the variable `output` to a scalar value of the TERR data type `difftime`, which Spotfire converts to the corresponding Spotfire data type TimeSpan. The TERR script is called once for each group of data to be aggregated. No special handling for aggregation is necessary in your TERR script.

The expression function has at least two arguments.

Argument	Argument description
A TERR script.	The TERR script contains the following. • A number of variables using the naming convention that Spotfire requires: `input1` to `inputN`, where `inputN` is the highest number of the specified inputs, numbered sequentially. • A TERR assignment operator (`<-`) that assigns the results of the TERR evaluation to an object named `output` (also using the naming convention that Spotfire requires).

Argument	Argument description
Spotfire column names.	Passed as additional arguments, these are the data column names that `input1` to `inputN` represent. All columns must be the same length.

The output type is returned from TERR and converted by Spotfire.

Returned by TERR	Converted in Spotfire
A single aggregated value of data type `difftime` with units defined as seconds.	A single aggregated value of data type TimeSpan.

TERRAggregation_TimeSpan example

In this example, in Spotfire, take thirty observations with a start time and an end time. Calculate the observation time for each entry.

```
TERRAggregation_TimeSpan("output <- max(input2-input1)",[Start
 time],[End time])
```

Spotfire autocorrects the function case or name to that of built-in Spotfire function names. (For example, TERR contains the function `max`, and Spotfire contains the function `Max`.) You must overwrite this autocorrection manually to ensure that you use the TERR function case and name in your expression function.

See Embedding the contents of a script in an expression function for a detailed procedure for creating an expression function.

Registering the TERR Script as an Expression Function

You can create write a TERR function, register it in Spotfire to call as an expression, and then use it in any analysis from the Custom Expression and Insert Calculated Column dialog boxes. You can also edit the saved expression function from the Spotfire user interface.

This task uses as its example a data set consisting of 30 consecutive dates and their temperatures in Fahrenheit. The expression function to write and save adds a column that converts the temperature to Celsius. To work through this example, you can copy the data set from Temperature data set for Spotfire examples.

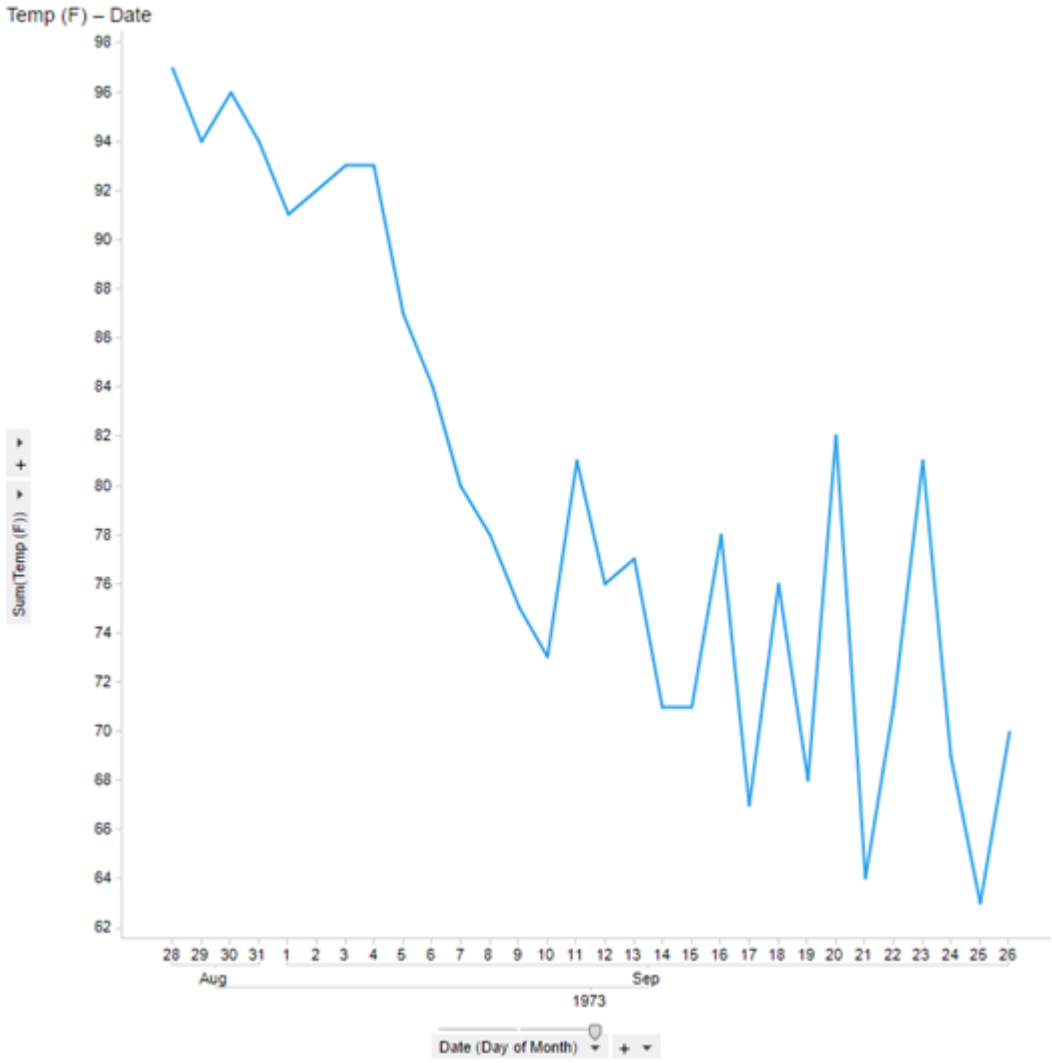
You can copy (CTRL+C) the contents of the sample data table and paste it (CTRL+V) into the Spotfire user interface.

Prerequisites

Copy the sample data set into Spotfire. (You can use any data set, and you can write any expression that works with your data type and visualization. This simple example expression is for demonstration only.)

Procedure

1. Create a visualization from the data. If you are using the data in this example, consider creating a line chart with the date on the X axis and the sum of the temperature on the Y axis.



2. From the menu, click **Edit** > **Data Function Properties**, and in the Data Function Properties dialog box, click the tab **Expression Functions**.
3. In the Expression Functions tab, click **New** to open the Expression Function dialog box.
4. Provide the metadata for the expression function.

Field	Description
Name	Spotfire displays this name in the function list for both the Custom Expression and Insert Calculated Column dialog boxes. For the example, call it `FahrenheitToCelsius`.
Description	If you plan to reuse this expression function, providing a description of its purpose, design, or use is a good practice. This description is displayed in the both the Custom Expression and Insert Calculated Column dialog boxes.  In the example, consider adding the text from the code comments (denoted by a hash mark, #) from the example data. These comments provide information and guidance about the function body for other users.

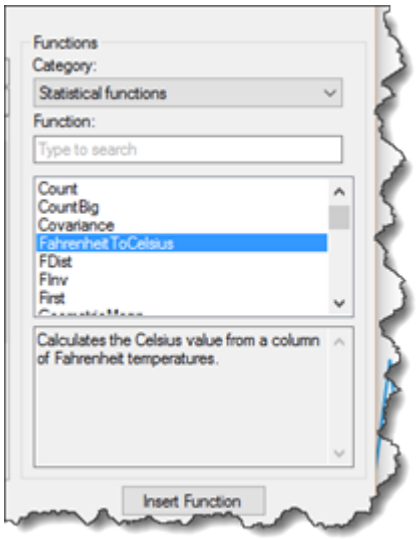
Field	Description
Function type	Specify whether the function adds a column or a single, aggregated value. This example specifies a column. A column specifies the same number of rows as the input.
Return type	Specify the Spotfire data type the expression function returns. For this function, set the data type to `Real`. See Data type mapping for more information.
Category	Specify the function list to contain the expression function. The function list is displayed in the Custom Expression and Insert Calculated Column dialog boxes. Categorizing the functions makes it easier to find different types of functions in the Spotfire user interface. By default, expression functions are listed under **Statistical functions**.

See the help available from the Spotfire dialog box for more information about these fields.

5. In the **Script** text box, write the function script.
   The script for the example data is written as follows.

```
# Define the FahrenheitToCelsius function.
FahrenheitToCelsius <- function(TempFahrenheit)
{
 TempCelsius <- (TempFahrenheit - 32) * (5/9)
 TempCelsius
 }
# Create and run the function as an expression to produce the output
# input1 specifies the column of the data to use in the evaluation. (In this case, input1
 is TempFahrenheit)
# output is the column containing the values calculated by the function.
output <- FahrenheitToCelsius(TempFahrenheit = input1)
```

6. Click **OK** to save the expression function and add it to the list of functions in the Custom Expression and Insert Calculated Column dialog boxes.
7. Return to the Spotfire visualization, and from the menu, click **Insert** > **Calculated Column**.
8. In the Insert Calculated Column dialog box, click the **Category** drop-down, and from the list, select the category you set in **Statistical functions**.
   The list displays the statistical functions in alphabetical order. For our example, the `FahrenheitToCelsius` is near the top of the list. Note that the description you provided when you created the expression function is displayed in the informational text.



9. Select the new expression function name, and then click **Insert Function**.
   The function is inserted in the **Expression** text box. The example shows `FahrenheitToCelsius()`.

10. From the **Available columns** list, click the column to which to apply the function, and then click **Insert Columns**.

    The example column is `Temp (F)`. This selection specifies that the calculation in the function is applied to each row of that column, and the new column contains the results of each calculation.

11. In **Column name**, provide a friendly name for the expression function.
    For example, type `Temp (C)`.

    If you do not perform this step, Spotfire uses the entire expression function as the column name.

12. Review the **Sample result**, and then click **OK** to run the expression function and return to the line chart visualization.

13. On the menu, click **Insert** > **Duplicate Visualization**.
    A duplicate of the line chart is displayed.

14. In the data panel, select the new calculated column and drag it to the Y axis.
    The Y axis shows the values as Temp (C), or Celsius, instead of Temp (F), or Fahrenheit.

15. Optional: Create a table visualization with all columns, including the new calculated column, and compare the values.

Result

The expression function you created is now registered in your installation of Spotfire Analyst. You can use the new expression function in other analyses with like data. Also, you can return to the Expression Functions dialog box and edit the expression function if you need to. (Any change you make to the expression function script is applied to any analysis that uses it.)

Embedding the Contents of a Script in an Expression Function

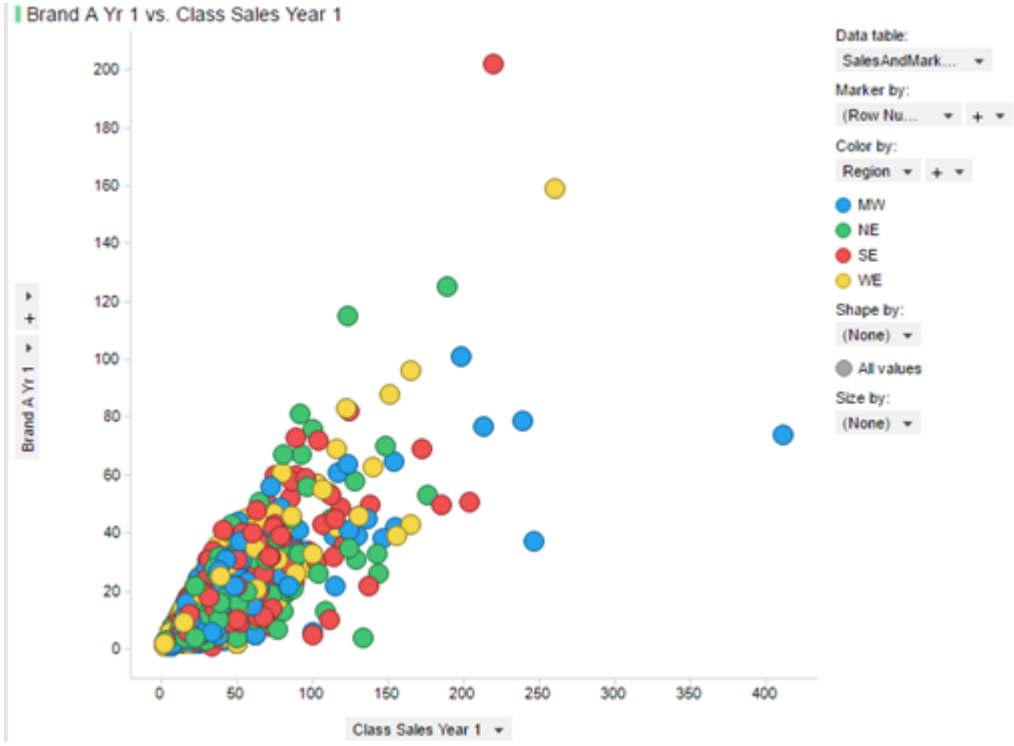
In the Insert Calculated Column or Custom Expression dialog box, you can use one of the Spotfire statistical functions, and embed a TERR script in the expression.

This task uses a sample data set Sales and Marketing, which is included in the Spotfire library, in the `Demo/Analysis Files/Sales and Marketing` folder. This simple example does not require statistical analysis. It is meant only to introduce concepts and workflow. This example demonstrates inserting a calculated column created from an expression function. You can try a similar technique using the Custom Expression dialog box, available by right-clicking the axis name.

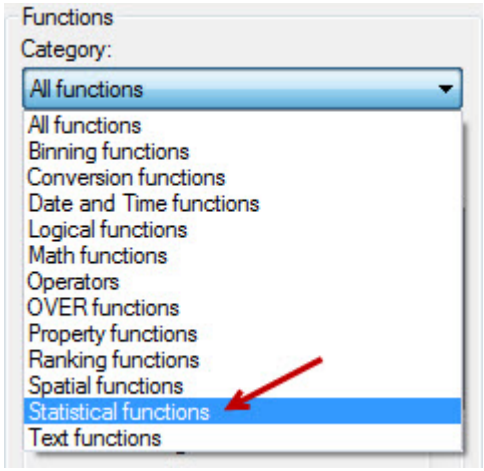
Procedure

1. Open the Sales and Marketing example data set and add a page.

2. Create a scatter plot visualization, setting the X axis to Class Sales Yr 1 and the Y axis as Brand A Share Yr 1.
   The resulting visualization appears as follows, with **Color by** set to Region.



3. From the menu, click **Insert** > **Calculated Column**.
   The Insert Calculated Column dialog box is displayed.
4. From the **Functions** list box, under **Category**, click the drop-down arrow.
5. From the list, select **Statistical Functions**.



   The **Function** list is filtered to show only the statistical functions.
6. In the list of statistical functions, find `TERR_Boolean`, and then double-click it.
   By selecting `TERR_Boolean`, you specify that the values returned by the call to the TERR engine should be returned as logical, which Spotfire converts to Boolean.
   The expression is inserted to the **Expression** text box, specifying the input name as `input1`, and the output name as `output`.

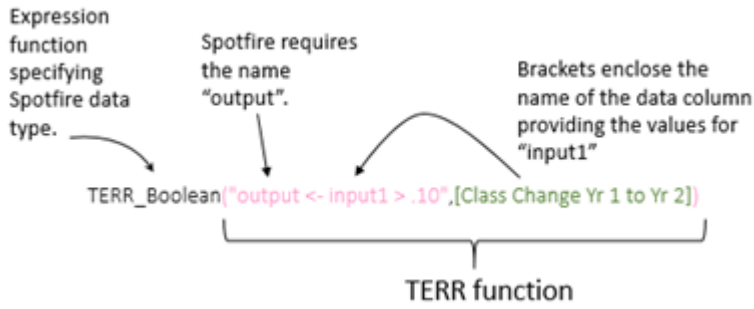
7. Place the cursor at the end of the expression, inside the closing parenthesis.
   This is where you add the column to which you apply the function.
8. From the **Available columns** list, double-click Class Change Yr 1 to Yr 2.
   The column `[Class Change Yr 1 to Yr 2]` is added to the expression.
9. Add the evaluation to perform in the expression.

   For this example, specify the following: `"output <- input1 > .10"`.

   This evaluation determines which of the entries in the column specified by the column Class Change Yr 1 to Yr 2 grew by more than 10%.

   > Spotfire autocorrects the function case or name to that of built-in Spotfire function names. (For example, TERR contains the function `max`, and Spotfire contains the function `Max`.) You must overwrite this autocorrection manually to ensure that you use the TERR function case and name in your expression function.

   The expression function is constructed, as follows.



   The **Sample result** box displays a valid sample of the result, such as `False`, and the **Type** box displays `Boolean`.
10. In the **Column name** text box, rename the column to a friendly name.

    If you do not rename it, the column name is the entire expression function.

    For the example, you could change the name to `Brand A change greater than 10%`.
11. Click **OK** to run the expression.
    The new column created by the expression function is displayed in the Data panel under **Categories**.
12. Select the **Color by** drop-down, and from the list, select te expression function name.

Result

The visualization displays two colors to indicate which Brand A products increased by over 10% (`True`) and which did not (`False`).

Aggregating Binned Weather Data Using TERR in Spotfire

Using a TERR expression function to aggregate data, you can create a Spotfire visualization that provides greater insight. This simple, generic example adds a custom expression, demonstrating how you might use these functions, and others like them, in your own analyses.

Perform this task in Spotfire. This task uses the data set air, which you can find in air data set for Spotfire examples.

This data set is a data frame with observations (rows) on four variables (columns), taken from an environmental study that measured ozone, solar radiation, temperature, and wind speed for 5 months in 1973 in the New York City area.

Prerequisites

In Spotfire, load the air data. You can copy the table from the air data set example.

You can copy (CTRL+C) the contents of the sample data table and paste it (CTRL+V) into the Spotfire user interface.

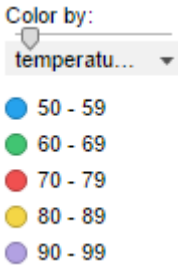
Procedure

1. Create a scatter plot, and set the axes as follows.
   - Set the X-axis to ozone.
   - Set the Y-axis to wind.

2. Click the **Color by** drop-down list, and in the resulting dialog box, select temperature, and select the **Auto-bin column** check box.

   By default, the expression results in the expression `AutoBinNumeric([temperature],5)`, which provides for 5 bins. You can change this value, but for the example, leave it a 5.

   The scatter plot points are colored by temperature, binned into five ranges, appearing as follows.



3. Right-click the visualization, and from the menu, click **Properties**.
4. In the Properties dialog box, select **Lines & Curves**.
   The Lines & Curves properties are displayed.
5. Click **Add** > **Horizontal Line** > **Straight Line**.
   The Horizontal Line dialog box is displayed.
6. Click **Custom expression**, and then click **Edit**.

   This location is just one of many ways you can access and create a custom expression in Spotfire.

7. In the Custom Expression dialog box, in the **Category** drop-down text box, select **Statistical functions**.
8. Scroll down the function list, and then from the list, double-click **TERRAggregation_Real**.

   `TERRAggregation_Real` is specified because the column used in the expression (wind) is the data type Real. You must always use the expression reflecting the data type of the column to which it is applied.

   The following expression, which specifies an aggregation of the data type Real, is displayed in the **Expression** text box.

```
TERRAggregation_Real("output <- input1[1]",)
```

9. Edit the entry in the **Expression** text box to calculate the mean of the value of the Y axis (wind) for each of the bins.

   The expression should read as follows.

```
TERRAggregation_Real("output <- mean(input1)",[Y])
```

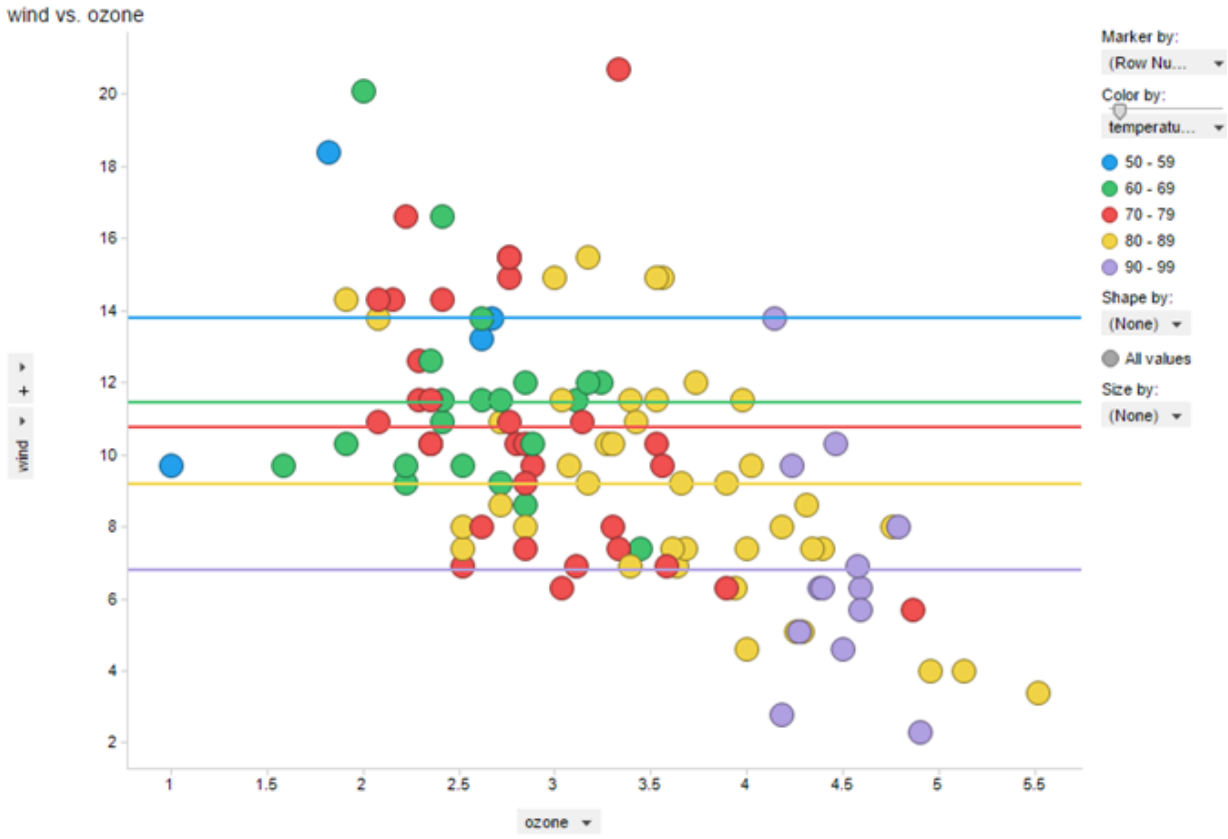
   Spotfire autocorrects the function case or name to that of built-in Spotfire function names. (For example, TERR contains the function `max`, and Spotfire contains the function `Max`.) You must overwrite this autocorrection manually to ensure that you use the TERR function case and name in your expression function.

10. After you have edited the expression, accept the changes to return to the Properties dialog box.
    The new entry, under **Visible lines and curves**, for the horizontal line based on the aggregation should be displayed, and its check box should be selected.
11. In the Properties dialog box, under **One per**, select the check box **Color**.

    This selection specifies that one horizontal line should be displayed for each of the colors used for the **Color by** temperature binning.

Result

The resulting visualization is displayed, showing the horizontal colored lines (one for each of the five bins), indicating the mean of each bin for the column on the Y axis (wind).



Expression Function Editing

After creating and saving an expression function in Spotfire, you can edit it. You can edit an expression function that is saved and registered in the function list, or that is an ad-hoc expression.

Editing a Registered Expression Function

You can edit the script or other details of an expression function that you have saved and registered in Spotfire Analyst.

Edit a saved and registered expression function in the Spotfire Analyst user interface.

Prerequisites

You must have a registered expression function in Spotfire Analyst. See Registering a TERR script as an expression function for more information.

Procedure

1. From the menu, click **Edit** > **Data Function Properties**.
2. In the Data Function Properties dialog, click the **Expression Functions** tab.
3. In the **Expression functions** list, select the expression function to edit, and then click **Edit**. The Expression Function dialog containing the expression function defintion is displayed.

4. Edit the **Script**, the **Function type**, the **Return type**, and the **Category**, as required.
   Be sure you specify the correct types, according to the script.
5. Click **OK**, and then click **Close** to save and rerun the data function.
   Review the results of the change in any visualization that uses the function.

Editing an Ad Hoc Expression Function

You can edit an ad hoc expression function that you have added as either a calculated column or custom expression.

You can change an expression function that you added as a calculated column. For information about editing an expression function that you added as a custom expression,

Prerequisites

You have added an ad hoc expression function and run it in your analysis. Spotfire Analyst is open, and the Data panel is displayed.

Procedure

1. In the Data panel, click the icon to expand the panel.
   The data table view is displayed.
2. From the data column list, select the expression function column name.
   The data panel displays the column view.
3. Click **Edit**.
   The Edit Calculated Column dialog is displayed.
4. Edit the **Expression** as required.
   Be sure you specify the correct types, according to the script.
5. Click **OK** to save and rerun the data function.
   Review the results of the change in any visualization that uses the function.

Data Functions

You can write data functions in R, S-PLUS, SAS, or MATLAB to perform statistical analyses and display the results in Spotfire.

Registering a Data Function in Spotfire

A data function is an embedded TERR script that you can save to the library and share with others. Spotfire Analyst includes several out-of-the-box data functions that you can study, edit and save, or run in example data sets. You can use this example task to learn to register a simple data function using a data set provided in the help.

Perform this task in Spotfire Analyst. Start by opening the data set contained in the help topic Aggregation Data for Spotfire Examples on page 81 and creating a table visualization of the three-column data set.

Prerequisites

You must have a license for advanced analytics in Spotfire Analyst. If you do not have access to the data function dialog box, see your Spotfire administrator.

Procedure

1. From the menu, click **Tools** > **Register Data Functions**.

2. In the Register Data Function dialog box, provide the data function basic information.

Option	Description
Name	Provide a name that is meaningful for the data function's intended functionality. For the example, name the data function aggregate.
Type	select the default, **R script - TIBCO Enterprise Runtime for R**. This selection specifies the type of statistical engine to use.
Packages	For this example, leave this text box blank. This exercise uses functionality in packages that are loaded at startup in TERR, so you do not need to install or load any additional packages.
Description	If you intend to share this data function for future use, you and other users can find additional information about the data function useful.
Allow caching	For the exercise, select the check box. (Clear this check box for a data function that evokes a random procedure, where you want the results to change each time it is run.)
Script	You can type the data function script directly in this text box, or you can copy and paste an R script you developed in RStudio or another development environment. See Step 3 for more detail about creating the script for the example. Alternatively, see one of the built-in data functions that ship with the TERR library examples of writing R code for a data function.

3. For the example, type the following script.

```
if (nrow(x) > 0) {
y <- aggregate(x[, c("Group", "x1", "x2")], by=list(x[, "Group"]), FUN=median)
} else {
y <- x
}
```

This example script body, including the output, the function, and the function arguments, includes the following.

`if... else`	Provided for error handling, the `if/else` statement ensures that if no rows are selected, the data function does not report an error.
`y`	The object containing the output of the function.
`aggregate`	The TERR function to run.
`x`	A `data.frame` (in TERR) that represents a data table in Spotfire Analyst. This object functions as the input paramter for the data function.
`"Group", "x1", "x2"`	The columns to aggregate.
`by`	An argument of the `aggregate` function that is specified as a list, which represents the vectors that we group by (in this case, the single vector, which is the `"Group"` column in the data set.)
`FUN`	An anonymous or in-line defined function, or a function available to use from one of the available packages in TERR, such as the base or stats package. This example specifies the `median` function from the stats package, which calculates the median of the input.

4. Click the **Input Parameter** tab, and then click **Add**.
The Input Parameters dialog box is displayed.

5. For the **Input parameter name**, assign x.

   A data function can have any number of input parameters. This example has only one: the table named x.

6. In the **Type** drop-down list, select **Table**.

   Remember in the script, you specified x as a data frame. The other choices are as follows.

Option	Corresponding type in TERR
Value	A vector of length 1.
Column	A vector.
Table	A data frame. (Select this option if you are working through the example.)

7. Select the **Allowed data types** for the script.

   For the example, select **String** and all numeric data types: **Integer**, **Real**, **SingleReal**, and **Currency**.

   Click **Numeric** to select all of the numeric data types.

8. Click **OK** to save the input parameter.
   The Input Parameters tab displays the single parameter.

9. Click the tab **Output Parameter**, and then click **Add**.
   The Output Parameter dialog box is displayed.

10. For the **Result parameter name**, designate y, and specify its **Type** as **Table**.

    For anything you create in the script that you then pass into Spotfire, you must designate as an output parameter.

11. Click **OK** to save the output parameter.
    The Output Parameters tab displays the single parameter.

What to do next

Edit the parameters for the data function.

Editing Data Function Parameters

After you have registered a data function in Spotfire Analyst, you can save it to the library to be used by others, or you can edit the parameters, and then embed it in your analysis.

Perform this task in Spotfire Analyst, from the Register Data Function dialog box.

Prerequisites

You must have completed the steps in Registering a Data Function in Spotfire on page 70.

Procedure

1. Click **Run** to save the data function and embed it into the analysis.

   Optionally, you can click **Save As** to save the analysis to the library, but for the example, continue testing the data function by running it.

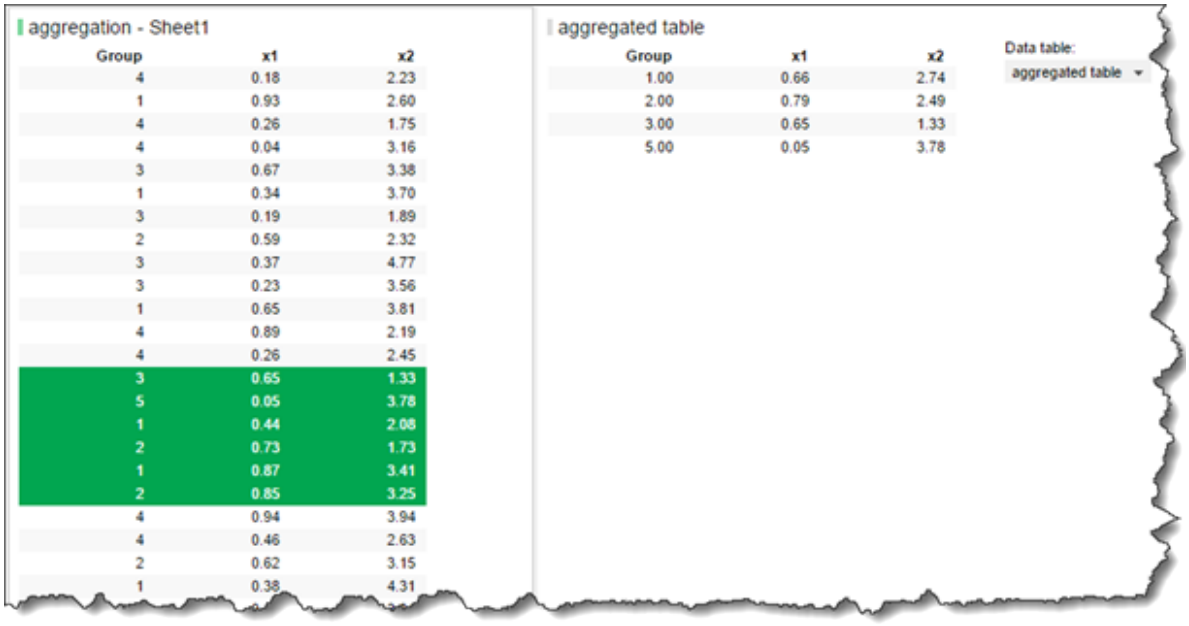
   The Edit Parameters dialog box is displayed.

2. In the Edit Parameters dialog box, in the Input tab, provide the following settings.

Option	Description
Refresh function automatically	For the example, select the check box to update the results from the data function automatically each time the input settings are changed. If the check box is cleared, you must perform a manual refresh for any updates to take effect.

Option	Description
Input parameters	Lists the input parameters for the data function. For the example, this table displays the example's one parameter, `x`.
Input handler	Lists all possible input handlers for the selected input parameter. The settings populating the dialog box depend on the selected input parameter. By default, the option **None** is selected. For the example, select **Columns**. In the resulting options, specify the following. • Click **Select Columns**, and then add all columns (`Group`, `x1`, and `x2`). • Scroll down and under **Limit by**, select **Marking**.

3. Click the Output tab and, for the example output parameter (`y`), set the **Output handler** to **Data table**.
4. Acccept the default **Create new data table**, provide a suitable table name, and then click **OK** to run the data function.
5. In the Register Data Functions dialog box, click **Close**.
   You are prompted to save the data function to the library. Click **Yes** to save it for reuse, or **No** to just continue running the example.
6. On the Spotfire Analyst toolbar, click the **Table** visualization icon to add a second table.
7. In the second table, from the **Data table** drop-down list box, select the table you created with the data function.
   The table visualization is blank.
8. In the first table, select a range of rows.
   The second table is populated with an aggregation from the selection in the first table.



Importing TERR Data Sets Using a Data Function

Several sample data sets are included in this help as HTML tables that you can copy and paste into Spotfire, but you can also write a data function to import data sets available in the TERR library Sdatasets.

Perform this task in Spotfire if you need sample data to model. You can find descriptions of the data sets in the Sdataset package included with TERR, and embedded in Spotfire. See Getting Help with TIBCO Enterprise Runtime for R.

Prerequisites

You must have a license for advanced analytics in Spotfire. If you do not have access to the data function dialog box, see your Spotfire administrator.

Procedure

1. From the menu, click **Tools** > **Register Data Functions** .
2. In the Register Data Functions dialog box, provide a name and a description.
   If you are working through the example, provide the following.

Option	Description
Name	air
Description	Observations (rows) on four variables (columns), taken from an environmental study that measured ozone, solar radiation, temperature, and wind speed for 5 months in 1973 in the New York City area.

3. In the Register Data Functions dialog box, in the Script tab, type the data function to import the data from the library.
   The name includes the name of the data object to create, the assignment operator, and the data set (including the library name) to import, in the form *`library::data`*. If you are working through the example, type the following.

```
airdata <- Sdatasets::air
```

4. Click the Output Parameters tab, and then click **Add**.
   The Output Parameter dialog box is displayed.
5. Provide the output name, and from the **Type** drop-down list box, select **Table**.
6. Click **OK**, and then in the Register Data Functions dialog box, click **Run** to run the data function.
   The Edit Parameters is displayed.
7. In the Edit Parameters dialog box, click the Output tab, and for the **Output handler**, select **Data table**.
8. Click **OK** to create the data table, and then in the Register Data Functions dialog box, click close.
   You are prompted to save the data function to the library. Click **Yes** to save it to the library, or click **No** to close without saving it.

Result

The data panel displays the column names, and the data is ready to use.

Testing Data Functions Inside and Outside of Spotfire

When you test a data function, to make sure you get consistent results in both the TERR engine in RStudio and the TERR engine in Spotfire, use the same data format and the same TERR engine in both environments.

When you read a `.csv` file containing your data, the TERR or open-source R function `read.table` performs a different data conversion than the conversion performed when you import the same data into Spotfire. The difference in these conversions can cause unwanted differences.

Perform this task in Spotfire Analyst.

Prerequisites

RStudio is installed on your computer.

Procedure

1. Import the file containing your data into Spotfire.
   Typically, this file is in `.csv` or a similar format.
   a) From the menu, click **File** > **Open**, and browse to the data file.
   b) In the Import Settings dialog, confirm the column data types, and then click **OK**.
   The data is imported, and a recommended visualization is displayed.
2. From the menu, click **File** > **Export** > **Data to File**.
   The Export Data dialog is displayed.
3. Select **Export all rows**, and then click **OK**.
4. Browse to the location to store the exported data, and in the **Save as type** drop-down list box, select **TIBCO Spotfire Binary Data Format (*.sbdf)**.
5. From the menu, click **Tools** > **TERR Tools**, and then click **Launch RStudio IDE**.
   RStudio opens, with the configuration to run the TERR engine that is included with Spotfire.
6. At the command prompt, read in the data using the following command.

   ```
   myData <- SpotfireData::importDataFromSBDF("myData.sbdf")
   ```

   where *myData* is the object name and *myData.sbdf* is the path and the name of the file you saved.

   The path to the file should use forward slashes instead of backslashes.

   The data file is imported into TERR in the same format as is used in Spotfire. This is the same process that is used when data is loaded in the data function.
7. In the RStudio IDE, create and test the function that you plan to use in Spotfire as your data function.

   If the data function is given zero rows of data (for example, if the data function is configured to be given only marked data, and initially there is often no marked data), in TERR, you can test the function for that possibility by specifying the source of the data as `myData[0,]`. Then testing shows all of the columns of the original data but no rows.
8. Copy the function to use in your data function, and then close RStudio.
9. In Spotfire, register and run the new data function.
10. If you see unexpected results when you run the data function in Spotfire, try the following.
    a) In Spotfire, at the top of the data function, add the following line of code, and then run it.
    `save.image("/winfolder/DataFunction.RData")`
    where *winfolder* is the Windows folder where to store the `.RData` file.
    b) In RStudio, at the top of the TERR function, add the following line of code, and then run through each line.
    `load("/winfolder/DataFunction.RData")`
    where *winfolder* is the Windows folder where to store the `.RData` file.
    (This loads the data that the data function got from Spotfire.)

Result

Identify differences; you should have the same results in both environments.

Enabling Debugging for Data Functions

In Spotfire Analyst, set the option to debug data function.

Perform this task in Spotfire Analyst.

Prerequisites

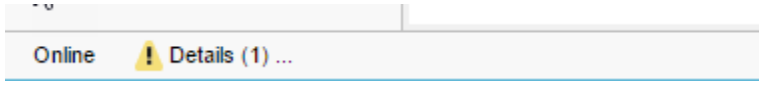
You must have the advanced analytics license to use the TERR toolset. If you do not have this license, see your Spotfire administrator.

Procedure

1. From the Spotfire Analyst menu, click **Tools** > **Options**.
2. From the list, click **Data Functions**.
3. In the Data Functions page, select **Enable Data function debugging**, and then click **OK** to accept the change.

Result

For any data function you run, Spotfire Analyst captures and displays the printed debugging output and any error that occurs. You can access the debugging output from the Details message displayed in the lower left corner of the Spotfire Analyst user interface.



Remember that when you enable debugging, performance can be affected, and a great deal of debugging information can be written, depending on the size of the data set and the complexity of the analysis. You should enable debugging only when it is needed.

Building a Spotfire Control to Check the Debugging Option

You can create a data function that uses the Spotfire debugging flag to check if data function debugging is enabled, and then use a Spotfire control to report the status of the debugging option.

Being able to determine quickly if you have set the debugging option for your data functions is a useful developing technique. Perform this task in Spotfire Analyst.

Prerequisites

- You must have the Advanced Analytics license in Spotfire Analyst to use the data function feature. If you do not see the menu item, contact the Spotfire administrator.
- Load a data set or an existing visualization to use for testing the debugging flag. If you are starting with a blank Spotfire Analyst session, you can use one of the data sets provided as examples. Follow the instructions in Importing TERR Data Sets Using a Data Function on page 73.

Procedure

1. From the Spotfire Analyst menu, click **Tools** > **Register Data Functions**.

2. Register the new data function.
   a) In the **Name** text box, provide a meaningful name, such as `DebugFlagTest`.
   b) Clear the check box **Allow caching**.
      You must ensure there is no caching so that the data function checks the option each time you send a request.
   c) In the Script tab, write the following script.

```
#create an object to check if the data function debuging option is set.
debug <- getOption("debug.spotfireConnector", FALSE)

#create an object to hold the output text reporting if the option is set.
x<- "Debugging is not enabled."
if (debug) x <- "Debugging is enabled."
```

      For more information about the `getOption` function, see the *TERR Language Reference*.
   d) Click the Output Parameters tab, and then click **Add** to specify the output parameter.
   e) For the **Result parameter name**, specify `x`, and for the **Type**, select **Value**.
   f) Click **OK**, and then click **Run**.
      The Edit Parameters dialog box is displayed.
   g) Clear the **Refresh function automatically check box**, and then select the Output tab.
   h) In the **Output parameters** list box, select the `x` output parameter, and for **Output handler**, select **Document property**.
   i) Click **New**, and in the New Property dialog box, for **Name**, type `DebugFlag`.
   j) Click **OK**, and then close the Register Data Functions dialog box.
      You are prompted to save the data function to the library.
   k) Click **Yes** to save the data function in the library so you can use it in all of your data-function-enabled visualizations.

3. Create a text area to provide the information returned from the data function.
   a) Add a Text Area to the visualization, and in the Text Area title bar, click the **Edit Text Area** button.
   b) Click the **Insert Property Control** button, and from the list, select **Label**.
   The Property Control dialog box is displayed.
   c) In the Document Properties tab, accept the default.
   The new property should be selected, and the Value should be the value you provided in the data function.
   d) Click OK to insert the label control.
   e) In the Edit Text Area window, click the **Insert Action Control** button.
   The Action Control dialog box is displayed.
   f) In the Actions pane, select **Data Function**.
   g) Provide **Display text**, such as `Refresh`.
   h) From the **Control type** drop-down list box, select **Button**.
   i) From the **Available data functions** list box, select the data function you created.
   j) Click **OK** to accept the changes.
   k) Move the button in the text area to an appropriate position in relation to the label control, and then close the Edit Text Area window, saving the changes.
   l) Optional: Provide a descriptive title for the text area (or remove the title), and resize the text area to an appropriate size.
4. Test the new controls.
   a) In the text box you created, click the button.
   The label control text should reflect the state of the data function debugging option.
   b) From the menu, click **Tools** > **Options**, and in the Options dialog box, scroll and select **Data Functions**, and then change the state of the **Enable Data Function debugging** check box, and save the change.
   c) In the text area, click the button again. The status should change to reflect the current state of the data function debugging option.

Debug a Simple Data Function

When you write a TERR data function to add to your Spotfire analysis, you can use the built-in debugging feature in Spotfire Analyst to examine the output debugging log, and you can find and fix problems with the data function directly from within the Spotfire Analyst environment.

The three tasks in this section introduce the debugging output, introduces a technique for checking your code at different points in its execution, and exporting the results of the data function to analyze outside of Spotfire Analyst.

Debugging a Simple Data Function

Create a simple data function and then examine the debugging output in Spotfire Analyst.

Create this example using any sample data set in Spotfire Analyst. The example described in this topic uses the geyser data set from the Sdatasets package in TERR.

Prerequisites

- You must have the advanced analytics license to use the TERR toolset. If you do not have this license, see your Spotfire administrator.
- You must have Enabling Debugging for Data Functions on page 76.

Procedure

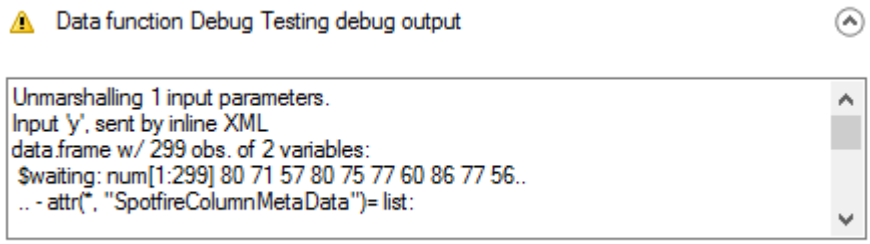
1. Open a data set in Spotfire Analyst.
   This example uses the TERR dataset `Sdatasets::geyser`. Follow the instructions in Importing TERR Data Sets Using a Data Function on page 73. Alternatively, you can use any data set.
2. Register a data function.
   This example shows a simple data function that assigns one table to another table.
   a) In the **Name** text box, provide a meaningful name, such as `Debug Testing`.
   b) In the Script tab, write a simple script to assign the existing table to a new table.

```
# This data function assigns the table y to the object x to create a new table.
x <- y
```

   c) On the Input Parameters tab, add the input parameter (in the example above, `y`) as a table, and allow all data types.
   d) On the Output Parameters tab, add the output parameter (in the example above, `x`) as a table.
   e) Click **OK**, and then click **Run**.
   The Edit Parameters dialog box is displayed.
   f) In the Edit Parameters dialog box, for the **Input parameter** `y`, specify the **Input handler** as an **Expression**.
   g) In the **Expression** text box, provide an expression.
   The example uses the following Spotfire expression, which uses property calls to create the new visualization. (See the topic *Properties in Expressions* in the Spotfire Help for property calls if you want more information on the property calls `$map`, `$esc`, and `$csearch`.)

```
$map("[geyser].$esc($csearch([geyser],"*"))",",")
```

   h) Click **OK** to run the data function.
   You should see the yellow notification triangle indicating that the data function ran and the results are in the debugger.
   i) Open the Notifications dialog box and review the debugging output.



   j) Clear the debugger, and then close the Notifications dialog box.

What to do next

Try writing a cat statement in the data function.

Adding a cat Statement to the Data Function

A `cat` or a `print` statement is useful for understanding your code's behavior. For example, you can use it to check the success of a condition, or to make a note in the output at a specific time when the code is running.

This example builds on the data function created in the example Debugging a Simple Data Function on page 78. You can add a `cat` statement to any data function.

Prerequisites

- You must have the advanced analytics license to use the TERR toolset. If you do not have this license, see your Spotfire administrator.
- You must have Enabling Debugging for Data Functions on page 76.
- You must have completed the example Debugging a Simple Data Function on page 78, or you must have your own data function that you want to debug.

Procedure

1. On the Spotfire Analyst menu, click **Edit** > **Data Function Properties**, and then in the Data Function Properties dialog box, click **Edit Script** to return to the data function script.
2. Add a `cat` statement to the script.

   If you are working through the example, add the `for` loop, shown below, and then save and close the script.

   ```
   # For the example, insert a script that simply prints an iteration.
   for (i in 1:10) {
     cat("iteration", i, "\n")
   }
   # The existing data function example
   x <- y
   ```
3. Click **Refresh**, and then close the dialog box.
4. Open the Notifications dialog box and review the debugging output.
   You should see the iteration in the debug output.
5. Close the Notifications dialog box.

What to do next

Try exporting the data function results to view in RStudio.

Exporting the Data Function Results

Being able to examine the data and the code outside of Spotfire Analyst is an important capability in developing robust data functions.

This example builds on the data function created in the example Debugging a Simple Data Function on page 78. You can export any data function using th code in this example.

Prerequisites

- You must have the advanced analytics license to use the TERR toolset. If you do not have this license, see your Spotfire administrator.
- You must have RStudio installed on your computer. for more information, see Configure RStudio to use TIBCO Enterprise Runtime for R on page 270.
- You must have Enabling Debugging for Data Functions on page 76.

- You must have completed the example Debugging a Simple Data Function on page 78, or you must have your own data function that you want to debug.

Procedure

1. On the Spotfire Analyst menu, click **Edit** > **Data Function Properties**, and then in the Data Function Properties dialog box, click **Edit Script** to return to the data function script.
2. Add the following code to the top of the data function.

```
z <- getOption("debug.spotfireConnector", FALSE)
if (z) save(list=ls(), file="C:/temp/debug.RData", RFormat=TRUE)
```

   This function determines whether the debugging option for TERR is set, and if it is, the data function debugging results are written to an `.RData` file. You can specify any directory, as long as you have write access to the directory.
3. Save the data function, and then click **Refresh** to run it.

   You can review the debugging results in the Notifications dialog box, if you want to.
4. From the **Tools** > **TERR Tools** menu, open TERR Tools, and then open RStudio.
   RStudio development environment opens, running TERR.
5. Open the `.RData` file you created by running the data function, and then examine the results.

Sample Data Sets

For your convenience, this help reference includes sample data sets. You can copy the contents of these tables to Spotfire to use in tasks and exercises to learn how to call TERR expression functions.

Alternatively, you can write a TERR data function that imports one or more data sets into Spotfire from TERR. See Importing TERR Data Sets Using a Data Function on page 73 for more information.

Aggregation Data for Spotfire Examples

Use this data set for the task for learning to write TERR data functions in Spotfire. This data set has one column of groups (1-5) and two columns of numeric variables. You can use this sample create an aggregated table based on selections you make in this table.

Use this data set to build the example in Registering a Data Function in Spotfire on page 70.

You can copy (CTRL+C) the contents of the sample data table and paste it (CTRL+V) into the Spotfire user interface.

groups data set

Group	X1	X2
3	0.59	2.54
5	0.93	1.11
4	0.78	4.10
1	0.52	2.86
5	0.92	2.00
5	0.65	3.25

Group	X1	X2
3	0.16	4.92
2	0.44	4.92
4	0.28	2.18
1	0.54	2.48
2	0.74	2.54
2	0.92	2.67
4	0.18	2.23
1	0.93	2.60
4	0.26	1.75
4	0.04	3.16
3	0.67	3.38
1	0.34	3.70
3	0.19	1.89
2	0.59	2.32
3	0.37	4.77
3	0.23	3.56
1	0.65	3.81
4	0.89	2.19
4	0.26	2.45
3	0.65	1.33
5	0.05	3.78
1	0.44	2.08
2	0.73	1.73
1	0.87	3.41
2	0.85	3.25
4	0.94	3.94
4	0.46	2.63
2	0.62	3.15

Group	X1	X2
1	0.38	4.31
5	0.42	3.31
3	0.69	3.17
1	0.53	0.69
5	0.28	5.38
4	0.97	3.35
4	0.97	1.57
3	0.65	3.26
1	0.79	3.80
2	0.71	4.63
2	0.2	1.33
1	1	1.83
2	0.1	0.79
5	0.49	3.62
3	0.86	2.70
4	0.08	3.14

Air Data Set for Spotfire Examples

Use this data set for the task for learning to write TERR expression functions in a custom expression in Spotfire. This data set has four columns taken from an environmental study that measured ozone, solar radiation, temperature, and wind speed for five months in 1973 in the New York City area.

Use this data set to build the example in Aggregating Binned Weather Data Using TERR in Spotfire on page 67.

You can copy (CTRL+C) the contents of the sample data table and paste it (CTRL+V) into the Spotfire user interface.

air data set

ozone	radiation	temperature	wind
3.44821724038273	190	67	7.4
3.30192724889463	118	72	8
2.28942848510666	149	74	12.6
2.6207413942089	313	62	11.5

ozone	radiation	temperature	wind
2.84386697985157	299	65	8.6
2.66840164872194	99	59	13.8
2	19	61	20.1
2.51984209978975	256	69	9.7
2.22398009056931	290	66	9.2
2.41014226417523	274	68	10.9
2.6207413942089	65	58	13.2
2.41014226417523	334	64	11.5
3.23961180127748	307	66	12
1.81712059283214	78	57	18.4
3.10723250595386	322	68	11.5
2.22398009056931	44	62	9.7
1	8	59	9.7
2.22398009056931	320	73	16.6
1.5874010519682	25	61	9.7
3.1748021039364	92	61	12
2.84386697985157	13	67	12
3.55689330449006	252	81	14.9
4.86294413109428	223	79	5.7
3.33222185164595	72	12.6	7.4
3.07231682568585	127	82	9.7
4.14081774942285	291	90	13.8
3.39121144301417	323	87	11.5
2.84386697985157	148	82	8
2.75892417638112	191	77	14.9
3.33222185164595	284	72	20.7
2.71441761659491	37	65	9.2
2.28942848510666	120	73	11.5

ozone	radiation	temperature	wind
2.35133468772076	137	76	10.3
5.12992784003009	269	84	4
3.65930571002297	248	85	9.2
3.1748021039364	236	81	9.2
4	175	83	4.6
3.41995189335339	314	83	10.9
4.25432086511501	276	88	5.1
4.59470089220704	267	92	6.3
4.59470089220704	272	92	5.7
4.39682967215818	175	89	7.4
2.15443469003188	264	73	14.3
3	175	81	14.9
1.91293118277239	48	80	14.3
3.63424118566428	260	81	6.9
3.27106631018859	274	82	10.3
3.93649718310217	285	84	6.3
4.29084042702621	187	87	5.1
3.97905720789639	220	85	11.5
2.51984209978975	7	74	6.9
4.30886938006377	294	86	8.6
4.7622031559046	223	85	8
2.71441761659491	81	82	8.6
3.73251115681725	82	86	12
4.34448148576861	213	88	7.4
3.68403149864039	275	86	7.4
4	253	83	7.4
3.89299641587326	254	81	9.2
3.39121144301417	83	81	6.9

ozone	radiation	temperature	wind
2.0800838230519	24	81	13.8
2.51984209978975	77	82	7.4
4.9596756638423	255	89	4
4.46474509558454	229	90	10.3
4.79141985706278	207	90	8
3.53034833532606	192	86	11.5
3.03658897187566	273	82	11.5
4.02072575858906	157	80	9.7
2.80203933065539	71	77	10.3
3.89299641587326	51	79	6.3
2.84386697985157	115	76	7.4
3.14138065239139	244	78	10.9
3.53034833532606	190	78	10.3
2.75892417638112	259	77	15.5
2.0800838230519	36	72	14.3
3.55689330449006	212	79	9.7
5.51784835276224	238	81	3.4
4.17933919638123	215	86	8
4.23582358425489	203	97	9.7
4.90486813152402	225	94	2.3
4.37951913988789	237	96	6.3
4.39682967215818	188	94	6.3
4.57885697021333	167	91	6.9
4.27265868169792	197	92	5.1
4.17933919638123	183	93	2.8
4.49794144527541	189	93	4.6
3.60882608013869	95	87	7.4
3.1748021039364	92	84	15.5

ozone	radiation	temperature	wind
2.71441761659491	252	80	10.9
2.84386697985157	220	78	10.3
2.75892417638112	230	75	10.9
2.88449914061482	259	73	9.7
3.53034833532606	236	81	14.9
2.75892417638112	259	76	15.5
3.03658897187566	238	77	6.3
2.0800838230519	24	71	10.9
2.35133468772076	112	71	11.5
3.58304787101595	237	78	6.9
2.6207413942089	224	67	13.8
2.35133468772076	27	76	10.3
2.88449914061482	238	68	10.3
2.51984209978975	201	82	8
2.35133468772076	238	64	12.6
2.84386697985157	14	71	9.2
3.30192724889463	139	81	10.3
1.91293118277239	49	69	10.3
2.41014226417523	20	63	16.6
3.10723250595386	193	70	6.9
2.41014226417523	191	75	14.3
2.6207413942089	131	76	8
2.71441761659491	223	68	11.5

Car Data Set for Spotfire Examples

Use this data set for the task for learning to write TERR expression functions in a custom expression in Spotfire. This data set gives information on makes of cars taken from the April, 1990 issue of Consumer Reports (pages 235-255). This data set contains 6 columns for 61 cars (rows).

Use this data set to build the example in TERR_Integer.

You can copy (CTRL+C) the contents of the sample data table and paste it (CTRL+V) into the Spotfire user interface.

car data set

Car	Weight	Disp	Mileage	Fuel	Type
Eagle Summit 4	2560	97	33	3.030303	Small
Ford Escort 4	2345	114	33	3.030303	Small
Ford Festiva 4	1845	81	37	2.702703	Small
Honda Civic 4	2260	91	32	3.125	Small
Mazda Protege 4	2440	113	32	3.125	Small
Mercury Tracer 4	2285	97	26	3.846154	Small
Nissan Sentra 4	2275	97	33	3.030303	Small
Pontiac LeMans 4	2350	98	28	3.571429	Small
Subaru Loyale 4	2295	109	25	4	Small
Subaru Justy 3	1900	73	34	2.941176	Small
Toyota Corolla 4	2390	97	29	3.448276	Small
Toyota Tercel 4	2075	89	35	2.857143	Small
Volkswagen Jetta 4	2330	109	26	3.846154	Small
Chevrolet Camaro V8	3320	305	20	5	Sporty
Dodge Daytona	2885	153	27	3.703704	Sporty
Ford Mustang V8	3310	302	19	5.263158	Sporty
Ford Probe	2695	133	30	3.333333	Sporty
Honda Civic CRX Si 4	2170	97	33	3.030303	Sporty
Honda Prelude Si 4WS 4	2710	125	27	3.703704	Sporty
Nissan 240SX 4	2775	146	24	4.166667	Sporty
Plymouth Laser	2840	107	26	3.846154	Sporty
Subaru XT 4	2485	109	28	3.571429	Sporty
Audi 80 4	2670	121	27	3.703704	Compact
Buick Skylark 4	2640	151	23	4.347826	Compact
Chevrolet Beretta 4	2655	133	26	3.846154	Compact

Car	Weight	Disp	Mileage	Fuel	Type
Chrysler Le Baron V6	3065	181	25	4	Compact
Ford Tempo 4	2750	141	24	4.166667	Compact
Honda Accord 4	2920	132	26	3.846154	Compact
Mazda 626 4	2780	133	24	4.166667	Compact
Mitsubishi Galant 4	2745	122	25	4	Compact
Mitsubishi Sigma V6	3110	181	21	4.761905	Compact
Nissan Stanza 4	2920	146	21	4.761905	Compact
Oldsmobile Calais 4	2645	151	23	4.347826	Compact
Peugeot 405 4	2575	116	24	4.166667	Compact
Subaru Legacy 4	2935	135	23	4.347826	Compact
Toyota Camry 4	2920	122	27	3.703704	Compact
Volvo 240 4	2985	141	23	4.347826	Compact
Acura Legend V6	3265	163	20	5	Medium
Buick Century 4	2880	151	21	4.761905	Medium
Chrysler Le Baron Coupe	2975	153	22	4.545455	Medium
Chrysler New Yorker V6	3450	202	22	4.545455	Medium
Eagle Premier V6	3145	180	22	4.545455	Medium
Ford Taurus V6	3190	182	22	4.545455	Medium
Ford Thunderbird V6	3610	232	23	4.347826	Medium
Hyundai Sonata 4	2885	143	23	4.347826	Medium
Mazda 929 V6	3480	180	21	4.761905	Medium
Nissan Maxima V6	3200	180	22	4.545455	Medium
Oldsmobile Cutlass Ciera 4	2765	151	21	4.761905	Medium

Car	Weight	Disp	Mileage	Fuel	Type
Oldsmobile Cutlass Supreme V6	3220	189	21	4.761905	Medium
Toyota Cressida 6	3480	180	23	4.347826	Medium
Buick Le Sabre V6	3325	231	23	4.347826	Large
Chevrolet Caprice V8	3855	305	18	5.555556	Large
Ford LTD Crown Victoria V8	3850	302	20	5	Large
Chevrolet Lumina APV V6	3195	151	18	5.555556	Van
Dodge Grand Caravan V6	3735	202	18	5.555556	Van
Ford Aerostar V6	3665	182	18	5.555556	Van
Mazda MPV V6	3735	181	19	5.263158	Van
Mitsubishi Wagon 4	3415	143	20	5	Van
Nissan Axxess 4	3185	146	20	5	Van
Nissan Van 4	3690	146	19	5.263158	Van

Observation Data Set for Spotfire Examples

Use this data set for the task for learning to write TERR expression functions in a custom expression in Spotfire. This data set has four columns for a fictional trial. Each row has a date, a start time, an end time, and the result of the observation, so it can be used with either a time span or a boolean expresssion.

Use this data set to build the example in TERR_TimeSpan.

You can copy (CTRL+C) the contents of the sample data table and paste it (CTRL+V) into the Spotfire user interface.

Observation data set

Date	Start time	End time	Observation
1-Mar-17	14:00:12	14:15:25	Positive
2-Mar-17	13:55:26	14:20:30	Negative
3-Mar-17	14:10:54	14:15:12	Positive
4-Mar-17	14:03:34	14:17:54	Negative
5-Mar-17	14:02:56	14:20:15	Positive

Date	Start time	End time	Observation
6-Mar-17	13:57:21	14:16:14	Positive
7-Mar-17	13:58:26	14:18:22	Positive
8-Mar-17	14:02:37	14:20:54	Negative
9-Mar-17	14:03:36	14:15:42	Negative
10-Mar-17	13:58:45	14:23:46	Positive
11-Mar-17	13:55:45	14:21:39	Negative
12-Mar-17	13:58:42	14:17:21	Negative
13-Mar-17	14:01:27	14:25:33	Negative
14-Mar-17	14:02:48	14:17:27	Positive
15-Mar-17	14:04:47	14:22:26	Negative
16-Mar-17	14:00:32	14:20:43	Negative
17-Mar-17	14:03:16	14:16:25	Positive
18-Mar-17	14:06:18	14:25:52	Negative
19-Mar-17	14:01:48	14:21:46	Positive
20-Mar-17	13:54:37	14:15:48	Negative
21-Mar-17	13:56:38	14:25:12	Negative
22-Mar-17	13:57:52	14:23:16	Positive
23-Mar-17	14:04:22	14:23:53	Positive
24-Mar-17	14:02:37	14:18:19	Positive
25-Mar-17	13:55:52	14:21:24	Negative
26-Mar-17	13:58:37	14:17:28	Positive

Date	Start time	End time	Observation
27-Mar-17	14:01:47	14:25:16	Negative
28-Mar-17	14:02:45	14:17:55	Positive
29-Mar-17	14:04:29	14:22:58	Negative
30-Mar-17	13:55:01	14:17:22	Positive
31-Mar-17	13:54:50	14:19:45	Positive

Temperature Data Set for Spotfire Examples

You can use this data set for the task for learning to write TERR expression functions in Spotfire. This data set is extracted from the a data frame with observations (rows) on five variables (columns), taken from an environmental study that measured ozone, solar radiation, temperature, and wind speed for 5 months in 1973 in the New York City area. (This sample set shows a date range between Auguest and September that have no missing values.)

Use this data set to build the example in Registering the TERR Script as an Expression Function on page 61.

You can copy (CTRL+C) the contents of the sample data table and paste it (CTRL+V) into the Spotfire user interface.

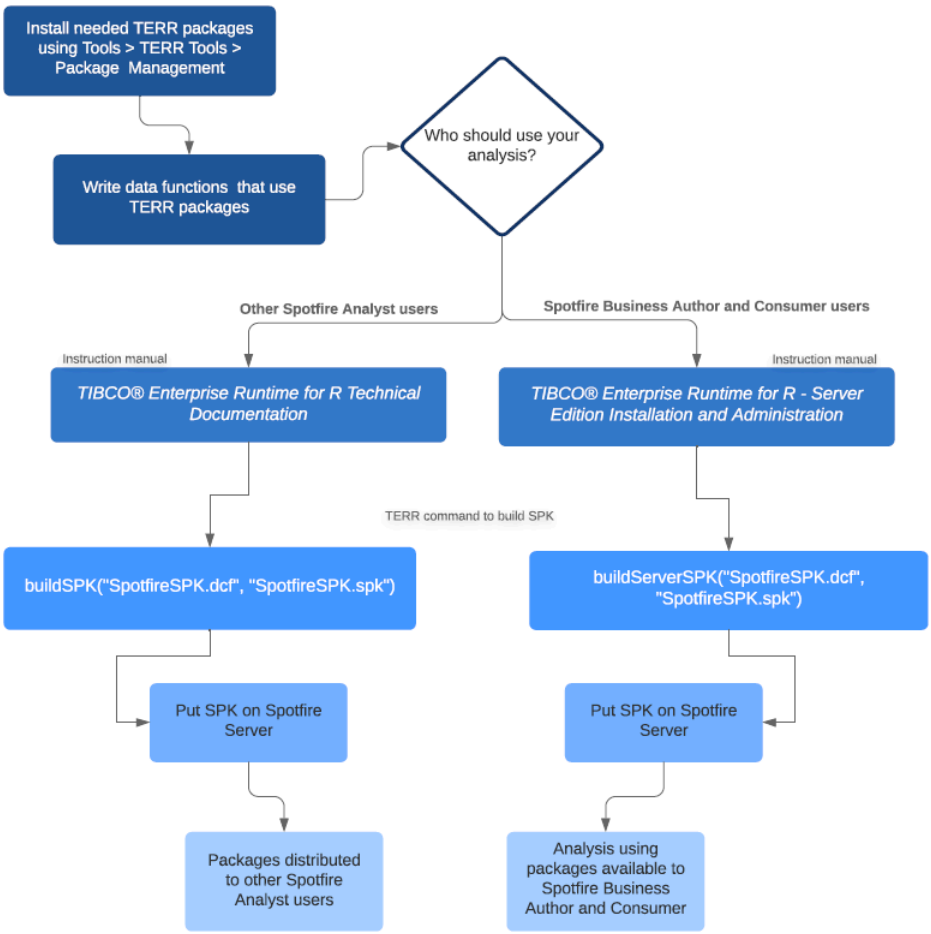
Date	Ozone	Radiation	Wind	Temp (F)
8/28/73	76	203	9.7	97
8/29/73	118	225	2.3	94
8/30/73	84	237	6.3	96
8/31/73	85	188	6.3	94
9/1/73	95.999	167	6.9	91
9/2/73	78	197	5.1	92
9/3/73	73	183	2.8	93
9/4/73	91	189	4.6	93
9/5/73	47	95	7.4	87
9/6/73	32	92	15.5	84
9/7/73	20	252	10.9	80
9/8/73	23	220	10.3	78
9/9/73	21	230	10.9	75

Date	Ozone	Radiation	Wind	Temp (F)
9/10/73	24	259	9.7	73
9/11/73	44	236	14.9	81
9/12/73	21	259	15.5	76
9/13/73	28	238	6.3	77
9/14/73	9	24	10.9	71
9/15/73	13	112	11.5	71
9/16/73	46	237	6.9	78
9/17/73	18	224	13.8	67
9/18/73	13	27	10.3	76
9/19/73	24	238	10.3	68
9/20/73	16	2.3	96	82
9/21/73	13	238	12.6	64
9/22/73	23	14	9.2	71
9/23/73	36	139	10.3	81
9/24/73	7	49	10.3	69
9/25/73	14	20	16.6	63
9/26/73	30	193	6.9	70

Package Management for the TIBCO Spotfire® Environment

This Package Management guide provides information about working with two kinds of packages in the Spotfire ecosystem: the R package, which you use with TERR, and the Spotfire package, or SPK, which you deploy from the Spotfire Server to client users.

This guide provides an orientation for both kinds of packages, including creating them, deploying and installing them, managing them, and troubleshooting them.



Package Management Orientation

If you are an R package developer but have no experience with Spotfire, or if your organization is new to advanced data analysis using TERR with Spotfire Analyst, review the topics in this section.

Find Help

Spotfire® includes many avenues to help with packages, whether they are R language packages to use with TERR™ or Spotfire packages (SPKs).

Task	Help resource
Building packages for open-source R.	R documentation  Open-source R is available under separate open source software license terms and is not part of TERR. As such, open-source R is not within the scope of your license for TERR. Open-source R is not supported, maintained, or warranted in any way by TIBCO Software Inc. Download and use of open-source R is solely at your own discretion and subject to the free open source license terms applicable to open-source R.
Building packages using TERR	*TIBCO® Enterprise Runtime for R (TERR™) Technical Documentation* at https://docs.tibco.com/products/tibco-enterprise-runtime-for-r
Creating a Spotfire SPK distribution.	*TIBCO Spotfire® Server Server and Environment Installation and Administration* at https://docs.tibco.com/products/tibco-spotfire-server
Learning about Spotfire Statistics Services architecture and server management, versus the TERR service management.	*TIBCO Spotfire® Statistics Services Installation and Administration* at https://docs.tibco.com/products/tibco-spotfire-statistics-services and *TIBCO® Enterprise Runtime for R - Server Edition Installation and Administration* at https://docs.tibco.com/products/tibco-enterprise-runtime-for-r-server-edition

R Language Primer

The R language has been developed into the open-source R engine and the TERR engine (among others), all developed from the legacy S-PLUS language.

Some differences exist between the open-source R engine and the TERR engine; however, they are highly compatible. Most scripts and functions that you write in open-source R run in the TERR engine.

For more information, see *Differences Between TIBCO Enterprise Runtime for R and Open-Source R* at https://docs.tibco.com/products/tibco-enterprise-runtime-for-r.

Open-source R is available under separate open source software license terms and is not part of TERR. As such, open-source R is not within the scope of your license for TERR. Open-source R is not supported, maintained, or warranted in any way by TIBCO Software Inc. Download and use of open-source R is solely at your own discretion and subject to the free open source license terms applicable to open-source R.

This documentation is not intended to teach programmers how to write scripts and functions or delve too deeply into creating R language packages. Rather, we address techniques for skilled package developers who must test and share their packages across an organization that has deployed Spotfire (and optionally Spotfire® Statistics Services), and that uses the TERR engine to run the package code.

See Recommendations for Using R Securely on page 23 for more guidelines for downloading R and packages.

Language Options

Whether you use TERR or open-source R, you can use the resources and tools in the Spotfire predictive analytics platform.

TIBCO Enterprise Runtime for R Packages

- Test your functions using the TERR engine and use TERR to build your packages. (See R Package Anatomy on page 105 for more information.)
- Use the SpotfireSPK package that is included in the TERR for Spotfire installation to create an SPK that you can put on a Spotfire® Server to distribute to other Spotfire Analyst users in your organization.

Open-source R packages

If you are an open-source R developer, you probably use either your own packages or those downloaded from a repository. You can test either by running them in a local TERR engine.

TIBCO maintains a report of tests run in TERR of CRAN packages' help examples. The report details the success of every expression in every help example provided by the package developer. TIBCO is not responsible for developing, testing, or supporting packages published to CRAN.

For a list of the CRAN packages for which we have run these tests, see the report for your server platform at https://docs.tibco.com/products/tibco-enterprise-runtime-for-r.

- In rare cases, we have provided different versions of packages tested to work with TERR, or we have customized popular CRAN packages to work with TERR. These package versions are loaded from a special TIBCO repository by default when you install them by calling `install.packages()`.
- For more information about testing your open-source R packages, see Testing Packages Locally on page 109.
- Check the list of known differences between the open-source R engine and the TERR engine by package.

Reviewing the List of Known Differences

In the installation TERR, you can find a list of known differences in function behavior between open-source R and TERR in the documentation.

You can access the list of known differences from Spotfire Analyst.

Procedure

1. From the menu, click **Tools** > **TERR Tools**.
2. Click **Open TERR Language Reference**.

   The language reference is included in the installation. The links to the technical guides and readme files open these documents on the TIBCO documentation website https://docs.tibco.com/products/tibco-enterprise-runtime-for-r.

   A web browser launches and displays the landing page with links to the documentation for TERR.
3. In the resulting browser window, click the link titled *Differences Between TERR and Open-Source R*.

Result

The resulting web page provides detailed information of known differences in function behavior between TERR and open-source R, sorted by their packages. (This list is compiled from like sections in the individual function help files.)

Spotfire Packages and R Binary Packages

Spotfire has two different types of packages: the Spotfire package (SPK), and the R language binary package.

The SPK is specific to Spotfire add-in development and deployment. You can create a Spotfire SPK that contains TERR engine-compatible R binary packages, and then deploy the SPK to a Spotfire Server to be distributed to other users in your organization.

Spotfire Analyst includes a license option for the TERR local engine. Your R binary packages, deployed using the SPK mechanism, can be used to create Spotfire analyses that use data functions.

R binary packages can run faster in a local TERR engine than one running in the TERR engine installed remotely on a Spotfire Statistics Services installation. For more information see Packages Running on a local TIBCO Enterprise Runtime for R engine in Spotfire.

Because you are creating or downloading a package to be distributed to Spotfire Analyst users, the package must be a Windows binary package. See Limitations and considerations for more information.

To create the SPK containing your R binary packages, you need the SpotfireSPK package. This package is provided with your TERR installation.

Manage Packages Through Roles

Working with packages in a deployment that includes Spotfire, Spotfire Server, and (optionally) Spotfire Statistics Services can add layers of complexity to management policies.

The job of synchronizing package versions among your development computers, your testing computers, and your servers is an important package management concern for an organization. You can reduce the risk of confusion and streamline your processes by defining roles in your organization for dealing with packages. Ensure processes and rules are established to manage packages.

Additionally, you can develop and use an in-house package repository from which all users install the same package versions. See Package Repositories on page 107 for additional guidance.

Developer Role

The developer is an R programmer or statistician who develops packages using TERR or open-source R.

The package developer accomplishes the following tasks using the Spotfire tools.

- Develop and test packages locally using the local TERR engine available from Spotfire Analyst.
- Create a Spotfire SPK containing the R code packages using the SpotfireSPK package, and then give them to the administrator who manages Spotfire distributions on Spotfire Server nodes. Packages uploaded to Spotfire Server nodes are loaded and run by the TERR service for users accessing advanced analytics from Spotfire Business Author.
- Upload packages for others to use in your organization's in-house repository. See Package Repositories on page 107 for more information.
- If your Spotfire environment includes Spotfire Statistics Services, and if you have the administrative permissions, upload the packages to Spotfire Statistics Services for users accessing advanced analytics from Spotfire Business Author. (Alternatively, give the package to the Spotfire Statistics Services administrator to upload.)

Curator Role

The curator maintains the standards and lists of officially-sanctioned packages. The curator keeps all of the package versions synchronized. The curator might be the same person who fills the developer role.

The approval process for adding a packaging is up to your organization, and might vary from minimal to extensive, depending on your usual practices. Designate a developer familiar with open-source R and TERR packages to be the package curator. The package curator works with package developers and server administrators to perform the following management tasks.

- Maintains the list of tested and sanctioned package versions (the gold standard), which would be the set of packages available for general use under Spotfire applications.
- Ensures that the SPK containing the gold standard package versions are placed on the Spotfire Server, distributed to Spotfire analysts who develop visualizations that use them, and available to

Spotfire Business Author users running data functions through the TERR service on the Spotfire Server.

- Creates and manages the organization's in-house repository from where all users can install packages.
- Ensures that the gold standard package versions are uploaded to Spotfire Statistics Services for use by Spotfire Business Author users.

Package versions used by these services must be kept synchronized.

Administrator Role

The Spotfire administrator manages packages on the Spotfire Server and on Spotfire Statistics Services.

The responsibilities for the administrator role include the following.

- Deploy the SPK containing CRAN packages to be called by TERR service or distributed to Spotfire Analyst users.
- Assign licenses for access to the Data Functions feature in Spotfire Analyst.
- Upload, maintain, and remove packages using the TERR console on Spotfire Statistics Services. (Might assign server permissions to the curator for this task.)

Package Installation Locations and Recommendations for Updating

You can use packages installed with TERR and with TERR in Spotfire Analyst, you can create and share your own packages, and you can download packages from a package repository. In all cases, you should know where they are installed and how they behave when you update your TERR or Spotfire installation.

Due to changes in open-source R version 3.5 and resulting compatibility changes in TERR 5.0, packages that are built with a version of TERR prior to 5.0 must be rebuilt.

- To install a binary package from a repository, always call `install.packages(`*`pkgname`*`)` from TERR. The `install.packages` function finds the correct binary version in the repository for your version of TERR. Manually downloading the binary package from CRAN can result in errors when you use it with TERR.
- To install a package from source, try installing it first with TERR (with `install.packages` in TERR or with `TERR CMD INSTALL` from a command line).
- To install a package from source that you cannot build with TERR, install the package with the version of open-source R tested with TERR.

This topic details where packages are installed by default with TERR, for those that you download and install from a package repository using the TERR console, those that you install using TERR Tools in Spotfire Analyst, and for those that you install from an SPK distributed by Spotfire Server.

From Spotfire Analyst TERR Tools, you can open the console and at the prompt, run the following commands to learn more about the location of installed packages.

- Run the function `.libPaths()` to discover where TERR finds packages. This is especially useful if you do not have write access to the `Program Files` directory on a Windows computer.
- Run the function `installed.packages()` to retrieve a list of all packages installed on the computer, and to discover other pertinent information, including where they are installed.

In Spotfire Analyst, you can use the TERR Tools Package Management tab to see a list of installed packages.

Installed by default with the TERR console application

Default Installation location	Description	Updating to a new version of TERR
TERR_HOME/library	Do not remove or change these packages. Doing so can cause unexpected behavior.	When you install a new version of TERR, the old library directory is removed and the packages are deleted. Updated versions of the packages are installed with the new version of TERR.

Installed by default with TERR in Spotfire Analyst

Default Installation location	Description	Updating to a new version of TERR
SPOTFIRE_HOME/Modules/*TERR_version*/engine/library	Do not remove or change these packages. Doing so can cause unexpected behavior.	When you install a new version of Spotfire Analyst, the old TERR library directory is removed and the packages are deleted. Updated versions of the TERR packages are installed with the new version of Spotfire Analyst.

Installed from a package repository using the console application

Default Installation location	Description	Updating to a new version of TERR
TERR_HOME/site-library	Packages downloaded from a repository are placed in this directory. On Windows, you must have write access to *TERR_HOME* for them to be installed at this location. If you do not have write access, packages you download are installed in the user directory. On Windows, this directory is [My Documents]/TERR/x86_64-pc-windows-library/*version*.	When you install a new version of TERR, the path to the older installation *TERR_HOME*/site-library is retained. You can take one of the following two steps. • Browse to the directory site-library for the older installation, and move the packages to the new installation directory *TERR_HOME*/site-library. (If you do not have write access, manage your packages by copying them from the older *version* to the new *version* in the user directory location.) • Download and reinstall the packages.

For more information, see Manage your packages when you install a new version of TIBCO Enterprise Runtime for R.

Downloaded and installed using TERR Tools in Spotfire Analyst

Default Installation location	Description	Updating to a new version of Spotfire
[My Documents]/TERR/x86_64-pc-windows-library/*version*	In all cases, packages downloaded from the repository using TERR Tools are placed in this directory.	When you install a new version of Spotfire Analyst, the path to the user library is retained. • Browse to the *version* directory for the older installation and move the packages to the new installation *version* directory . • Use TERR Tools to download and reinstall the packages. .

For more information, see Manage packages using TIBCO Spotfire and TIBCO Enterprise Runtime for R.

Installed by an SPK distributed in an update by Spotfire Server

Default Installation location	Description	Updating to a new version of Spotfire Analyst
SPOTFIRE_HOME/Modules/TERR Packages/library*version*	Custom packages provided to you through an update when you connect to Spotfire Server are placed in this directory.	The packages provided to you through an update by Spotfire Server should be reinstalled by the Spotfire Server.

For more information, see The Spofire SPK.

Installed on Spotfire Statistics Services

Default Installation location	Description	Updating to a new version of Spotfire Analyst
SPSERVER_HOME/data/appdata/library for a single server, or *SPSERVER_SHARE*/data/appdata/library for a cluster.	Packages that are uploaded to Spotfire Statistics Services for use by any client connecting to the Spotfire Statistics Services server. Your server administrator is responsible for setting the package location property, if necessary.	Your server administrator should preserve the data directory during updating.

Setting JAVA_HOME

Some packages that you use with TERR require access to Java on your system. If you call the TERR function `Sys.getenv("JAVA_HOME")` and it returns an empty string, you must set *JAVA_HOME* so the packages can access Java.

Perform this task on your Windows Or Linux system.

Prerequisites

The following list describes a few of the packages that are either provided with TERR or that you can use with TERR, but they require a bit-matching 32-bit or 64-bit version of Java, version 6 or later. (You might find other packages that require Java. These instructions can help you prepare your TERR session for those packages, too.)

Package name	Provided in your TERR installation
parallel	yes
sjdbc	yes
terrJava	yes
rJava	no  See Installing the rJava package for more information.

Procedure

1. Locate your Java installation and make a note of it.

   Your system can have more than one version of Java. Generally, use the latest version. On Windows, you can find this path in the registry. On Linux, you can usually find a link to it in the `\user\bin` directory.

   For example, on Windows, this path might be `C:/Program Files/jdk-11.0.1`.
2. Start a session of the TERR console.
3. At the TERR command prompt, type the command `Sys.setenv(JAVA_HOME="path_to_ your_Java_installation")` where *`path_to_ your_Java_installation`* is the path you noted in Step 1.
   For example, on Windows, this call might look like the following.

   ```
   > Sys.setenv(JAVA_HOME="C:/Program Files/jdk-11.0.1")
   ```

   On Linux, this call might look like the following.

   ```
   > Sys.setenv(JAVA_HOME="/usr/lib/jvm/java-11-sun/"
   ```

   Your system environment *JAVA_HOME* is now set to the specified Java installation.
4. Optional: Check the setting for *JAVA_HOME* from TERR by typing `Sys.getenv("JAVA_HOME")`.
   For example, on Windows, it might look like the following.

   ```
   > Sys.getenv("JAVA_HOME")
   [1] "C/Program Files/jdk-11.0.1"
   ```

What to do next

Install the package that requires setting *JAVA_HOME*. For an example, see Installing the rJava package.

Installing the rJava Package

The rJava package gives access to low-level R functions to the Java interface, but it is not provided with TERR. These instructions help you prepare your computer to use rJava.

Prerequisites

The rJava package requires the following.

- A bit-matching 32-bit or 64-bit version of Java, version 6 or later, is installed. (Tested with version 11.0.1.)
- The system variable *JAVA_HOME* is set. Follow the instructions for Setting JAVA_HOME if you are unsure.

These instructions are for installing the rJava package for use with TERR 4.2 or later. If you are using an earlier version, and you cannot update your version of TERR, see the release notes for more information for the verison of TERR you are running.

- For TERR version 3.1 and earlier, the rJava package does not work. To use rJava, update your version of TERR.
- For TERR version 3.2, you must use a build of rJava from TRAN. See that version's release notes for more information.

Perform this task in the TERR console or in TERR running under RStudio.

Procedure

1. At the command prompt, type `install.packages("rJava")`.
   The rJava package is installed from the package repository to the `site-library` directory.
2. At the command prompt, type `library(rJava)`.
   For example:

```
> library(rJava)
The following object(s) are masked _from_ 'package:utils':

    head, str, tail
The following object(s) are masked _from_ 'package:methods':

    new, show
The following object(s) are masked _from_ 'base':

    anyDuplicated, duplicated, rev, sort, unique
```

   The rJava package is now in your search path.
3. At the command prompt, type `searchpaths()`.
   For example:

```
> searchpaths()
[1] ".GlobalEnv"
[2] "C:/Program
    Files/TIBCO/terr60/site-library/rJava"
[3] "C:/Program
    Files/TIBCO/terr60/library/stats"
[4] "C:/Program
    Files/TIBCO/terr60/library/graphics"
[5] "C:/Program
    Files/TIBCO/terr60/library/grDevices"
[6] "C:/Program
    Files/TIBCO/terr60/library/utils"
[7] "C:/Program
    Files/TIBCO/terr60/library/methods"
[8] "C:/Program
    Files/TIBCO/terr60/library/base"
```

   The absolute file path is returned for each package in the current environment. Note that by default, the newly-loaded rJava is listed second in the search path.

Manage your Packages when You Install a New Version of TERR

When a new version of TERR is released, you might want to install it to take advantage of the changes. You can run older and newer versions of TERR on the same computer, or you can uninstall the older version(s). In either case, you probably want to make sure any custom-created packages or packages downloaded from a repository are kept available to the TERR version(s) you are running.

Uninstalling TERR does not remove the packages you installed. However, we recommend that you check for updates to any packages you have downloaded from package repositories after you install a new version of TERR. You can check for updated versions by calling `update.packages()`. See the help topic for `update.packages()` in the TERR *Language Reference* for more information.

The TERR installation includes the directory *`TERR_HOME`*`/site-library`, which is used by default. If you want to use another directory, you can define the environment variable *TERR_LIBS_SITE* and set it to the directory of your choice.

Initially, the `site-library` directory is empty. If you have permission to write to the *`TERR_HOME`* directory, any packages you create or download are installed in *`TERR_HOME`*`/site-library`.

Installing packages to the `site-library` directory provides the following advantages.

- It provides you with the means to protect and manage the packages you installed and want to keep, separate from the new installation.
- It separates the packages shipped with TERR so they can be updated with new releases, and so you do not accidentally change or remove them.

You should avoid changing any entries in the *TERR_HOME*`/library` directory. Doing so can cause TERR to behave in unexpected ways.

The directory *TERR_HOME*`/site-library` is added to the head of the search path, which is returned by `.libPaths()`. For example, on a Windows computer where you have permission to write to the *TERR_HOME* directory, this function call would appear as follows.

```
> .libPaths()
[1] "C:/Program Files/TIBCO/terr60/site-library"
[2] "C:/Program Files/TIBCO/terr60/library"
```

After installing the new version of TERR, you can just copy the packages from the older *TERR_HOME*`/site-library` directory to the new *TERR_HOME*`/site-library` directory.

If you are downloading packages to a computer where you do not have permission to write to the directory *TERR_HOME*`/site-library`, then packages are stored in the user directory. For example, on Windows, this directory is *[My Documents]*`/TERR/x86_64-pc-windows-library/<`*version*`>`, and, calling `.libPaths()` would appear as follows.

```
> .libPaths()
[1] "C:/users/jdoe/Documents/TERR/x86_64-pc-windows-library/terr6.0"
[2] "C:/Program Files/TIBCO/terr60/site-library"
[3] "C:/Program Files/TIBCO/terr60/library"
```

In this case, you can ignore the `site-library` directory (which remains empty) and manage your packages by copying them from the older `<`*version*`>` to the new `<`*version*`>` in the user directory location.

Manage Packages Using Spotfire and TERR

Analysts and data scientists can download existing packages, or they can build and test their own packages. They can wrap them in a Spotfire SPK for distribution to analysts in their organization. Spotfire Server administrators can distribute the packages using Spotfire Server, and Spotfire web client users can view Spotfire analyses that use the packages.

Because this documentation specifically addresses using TERR in Spotfire Analyst, the instructions demonstrate using the TERR tools available from the Spotfire Analyst user interface. Experienced R or TERR programmers, or programmers using TERR from the stand-alone console can perform many of these tasks directly from the console or from within RStudio. See the TERR *Technical Guide* and *Language Reference* for more information.

For the following tasks, Spotfire products each play a role in building, distributing, using, and maintaining packages in the Spotfire predictive analytics platform.

Package management tasks and tools

I want to...	Available tools
Discover the packages available in TERR.	TERR in Spotfire Analyst and the TERR stand-alone console.
Review the differences between TERR and open-source R.	TERR in Spotfire Analyst and the TERR stand-alone console.
Review the packages on CRAN that have been tested with TERR.	TERR in Spotfire Analyst and the TERR stand-alone console.

I want to...	Available tools
Find and install a package from a repository.	TERR in Spotfire Analyst and the TERR stand-alone console.
Create an SPK so the package can be shared with other Spotfire analysts and data scientists in the organization.	TERR in Spotfire Analyst and the TERR stand-alone console.
Remove a package from a Spotfire installation.	TERR in Spotfire Analyst
Change the version number for a package distributed to Spotfire analysts and data scientists.	Spotfire Server
Distribute a package to all Spotfire analysts and data scientists in your organization.	Spotfire Server
Distribute a package to a small group of Spotfire analysts and data scientists for testing.	Spotfire Server
Provide access to a package for Spotfire web client users who want to view data visualizations that rely on that package.	Spotfire Statistics Services

Development Tools for Creating Packages

Spotfire Analyst provides both a console application and access to an installation of RStudio so experienced R developers can create and test packages to use in Spotfire analyses.

Both the TERR console application an RStudio access are available from the Spotfire menu **Tools** > **TERR Tools**. Also you can find links to all of the TERR documentation from the **TERR Tools** dialog.

Before you develop your package, review the limitations and considerations, the list of components that make up a package, and the guidance for testing the package

Limitations and Considerations

Before you begin developing packages to use with TERR, you should familiarize yourself with some basic limitations.

Platform considerations

You can develop or download open-source R packages to run on either LINUX® or Microsoft Windows® platforms. Therefore, when you write your code to run on the server, take into account the platform you expect it to run on.

- If you plan to run packages locally on a TERR engine in Spotfire Analyst, your package must be a Windows binary package.
- If you plan to deploy a package to run TERR in Spotfire Statistics Services, the package type must be binary, and it must match the platform of the Spotfire Statistics Services deployment.

Graphical limitations

The TERR engine provides no graphical functions itself. However, it can run as a statistical engine in Spotfire Analyst, which provides data visualizations. Alternatively, you can use some CRAN packages (such as htmlwidgets,dygraph, or leaflet) that are compatible with TERR and that provide graphical displays in the browser, or you can use the RinR package, provided with your TERR installation, to call open-source R functions for graphic displays. You should always test package functions to make sure they run as expected. See *Graphics in TIBCO Enterprise Runtime for R* for more information.

Licensing limitations

To make a data function available in an analysis published to a Spotfire library, you must have the appropriate licenses. If you are not sure of your licenses, or if you do not see the **Tools** > **Register Data Functions** menu in the Spotfire Analyst installation, see your Spotfire Administrator for more information.

R Package Anatomy

Open-source R Packages running in the TERR engine are binary, and they must follow the standard package component design and contain version information.

The TERR engine is designed to be highly compatible with the open-source R engine. To develop packages using open-source R, see its documentation, available on the Comprehensive R Archive Network (CRAN).

Open-source R is available under separate open source software license terms and is not part of TERR. As such, open-source R is not within the scope of your license for TERR. Open-source R is not supported, maintained, or warranted in any way by TIBCO Software Inc. Download and use of open-source R is solely at your own discretion and subject to the free open source license terms applicable to open-source R.

A typical package is available as a binary file or as source code. You can use TERR to install a package from source. If the package has C/C++ code, you must first install the package rinclude (`install.packages(rinclude)`) before calling `install.packages(`*`packagename`*`)`. TERR does not support installing packages that have Java source code. See Installation Options for Packages on page 18 for more information.

Package components

A source package can contain any number of directories, including `html` (for the help index), `libs`, `help`, and so on. The simplest package requires the following files and directories:

File or Directory	Description
`mypkg`	The top-level directory name, which is also the package name (in this case, `mypkg`).
`mypkg/NAMESPACE`	Required. You must specify the NAMESPACE.
`mypkg/DESCRIPTION`	The file containing a description, the title, the author, date, the dialect, and version information, along with other information.
`mypkg/R`	The directory containing `*.R` files with R language functions as ASCII files.
`mypkg/R/mycode.R`	The R language code.

Your source package can also contain the following optional folders:

- `data` directory containing data files in a dump format.
- `man` directory containing help files in the `.Rd` help file format.
- `inst` directory contains files and directories to be copied, recursively, into the main package directory when the package is compiled. Any informational files that the end user should see should be included in the `inst` directory. For example, if you have a PDF containing a vignette, you can include it in the `inst/doc` directory.
- `src` directory, containing C, C++, or FORTRAN code.

- `tests` directory can contain package-specific tests. This directory can contain test code (that is, `.S`, `.ssc`, `.q`, and `.R`).

The TERR engine does not support the following:

- Packages using graphics devices or containing graphics functions. (However; we have implemented stub functions to allow the non-graphical portions of many packages to run without error.)
- Source packages with `src` directories that contain Java code.

Package versioning

Package version information is kept in the `DESCRIPTION` file. With every package revision, remember to revise the version number upward. This version number is an important part of your package management strategy.

Packages Running in a Local TIBCO Enterprise Runtime for R Engine in Spotfire

Running functions locally requires making the TERR engine available under Spotfire.

To run the functions in a package using the TERR engine that is installed with Spotfire Analyst, you must have the appropriate license enabled by your Spotfire administrator.

Remember that other people in your organization might be running analyses that depend on data functions stored in the library and that take advantage of the version of the package you are using.

In Spotfire, these scenarios are possible because Spotfire Analyst includes an embedded TERR engine. With enough resources, you can efficiently test and run analyses using data functions locally.

Although the package is in your Spotfire installation, it is not loaded into the engine. To use the package, you must load the library as part of your data function script or include it in the `Packages` field of your data function.

Opening the TIBCO Enterprise Runtime for R Console from Spotfire Analyst

TERR is provided in your installation of Spotfire Analyst so you can script and run data functions or create predictive models.

TERR Tools is provided to give you access to the TERR console to test scripts and functions, to launch the RStudio interactive development environment for script authoring, and to the TERR Language Reference for help with installed packages. TERR Tools also provides an interface to download and install packages from a package repository..

Perform this task from Spotfire Analyst.

Prerequisites

To work with TERR, you must have the appropriate license in Spotfire Analyst.

Procedure

1. From the menu, click **Tools** > **TERR Tools**.
2. Click **Launch TERR Console**.
   The TERR engine console that is included in the Spotfire Analyst `Modules` folder is displayed in a separate window.

Checking Installed Packages

You can find a list of packages included in the installation of TERR.

Check the packages that are installed and available to the TERR engine from TERR Tools in Spotfire Analyst.

Procedure

1. From the menu, click **Tools** > **TERR Tools**.
2. Click **Open TERR Language Reference**.

   The language reference is included in the installation. The links to the technical guides and readme files open these documents on the TIBCO documentation website https://docs.tibco.com/products/tibco-enterprise-runtime-for-r.

   A web browser launches and displays the landing page with links to the documentation for TERR.
3. In the resulting browser window, under **References**, click **Included Packages**.

Packages for Use with Spotfire

TERR includes a set of packages specifically for working with Spotfire.

Package Name	Description
Spotfire	This package is deprecated. See SpotfireData.
SpotfireConnector	This package contains functions used between Spotfire and Spotfire Statistics Services. You functions directly.
SpotfireData	This package contains functions for managing datasets between Spotfire and TERR. (In part reading and writing files in the Spotfire Binary Data format (SBDF).) Advanced users might Spotfire data directly in the TERR engine for debugging purposes.
SpotfireSPK	This package contains two functions: • `buildSPK` • `buildServerSPK` Use this package to create an SPK to contain your R language packages that you want distri users or on Spotfire Server. (The packages in your SPK can be automatically distributed to S working with data functions with local TERR engines.)
SpotfireStats	This package contains the helper functions used by Spotfire predictive analytics. You would functions directly.
SpotfireUtils	This package contains utility functions for interfacing between Spotfire and the TERR engin

Package Repositories

You can find or store packages in a variety of repositories. You can create your own in-house repository using the drat package.

You can download a package from a public repository, such as MRAN, CRAN, or TERR Archive Network (TRAN) repository, or you could use the drat package to create and maintain an in-house repository that is managed by your organization's package curator.

TIBCO does not warrant, deliver, or support code or other material provided by the R Project for Statistical Computing, including but not limited to development tools and packages, and such code and other material does not constitute a part of TERR. Such material therefore is not within the scope of your license for TERR. Download and use of such material is solely at your own discretion and subject to the free open source license terms applicable to such material. TIBCO recommends that you consult a legal professional concerning compliance with any free open source license terms applicable to such material, particularly if you plan to engage in redistribution of TERR and/or such material. (Please note that TERR may be redistributed solely pursuant to a license that expressly grants such redistribution rights.)

Public repositories

You can download open-source packages from a repository, and then use the functions in the package in your data functions in Spotfire. Be sure to review the list of differences between the TERR engine, available from the TERR landing page. Also review the limitations and considerations.

TRAN packages

A number of popular CRAN packages occasionally require customization to work with TERR. TIBCO hosts a package network, TRAN, where you can find these customized packages. Individual users can download these packages using the TERR console application by calling the function `install.packages()`. Optionally, these customized packages can be built into a Spotfire SPK and placed on a Spotfire Server to be available to Spotfire users throughout your organization.

Some packages customized and placed on TRAN require other packages not available on TRAN. Some of these packages cannot be installed using the TERR function `install.packages`, so the TRAN package cannot be successfully installed. If you encounter this situation, try building and installing the package using open-source R.

Company repositories

One skilled package developer in your organization should have the role of package curator to oversee the package version integrity within the company's package repository, whether these are packages downloaded and tested from a public repository or developed internally and kept in a CRAN-like in-house repository.

One easy way to establish and maintain a company repository is with the drat package, which is available in the MRAN and CRAN repositories. Using the drat package, you can create an in-house repository (on either a web server or a network share), you can upload packages to the in-house repository, and you can install packages from the in-house repository. For more information on using drat, install it into your TERR or open-source R console application, and then refer to its language reference and vignettes.

For more information about installing existing packages, see Installation options for packages.

Specifying an Older Package on TRAN

TERR and open-source R search for packages differently. By default, the repository search order for `options('repos')` is set to TRAN, MRAN, and then CRAN. If a package exists in all repositories, then TERR selects the version in TRAN, but open-source R selects the newest one based on the package version number.

This search-order difference is by design, because if a newer package on CRAN causes problems when tested with TERR, then TRAN contains an older version of a package that has been tested successfully with TERR. (The MRAN snapshot contains the version of the package available when this version of TERR was made available.)

In some cases, you must use open-source R to install a package to use with TERR. For example, under Linux, packages with source code for use in TERR often need to be installed using open-source R. If you encounter the search-difference issue with such a package, and an older version is available on TRAN, then you must take additional steps to make sure you get the working package version.

Procedure

1. From open-source R, run `install.packages(pkgname)`.
   The newest version of the package specified by *pkgname* is installed, along with its dependencies.
2. Set `options('repos')` to `c("https://tran.tibco.com/terr##")`.
   The repository search option is set to check only the TRAN site and the version of TERR specified by the version number ##.
3. Reinstall the needed package.
   The package is installed according to the TRAN site search option.

Testing Packages Locally

After you have either created a package or downloaded a package from a repository, you can test the package functions by running the example files in the TERR engine. Then you can write your own scripts using the package functions, and test them using Spotfire Analyst.

Prerequisites

Your Spotfire Analyst installation should be configured to use the local TERR engine. You can check this option from the Spotfire Analyst menu by clicking **Tools** > **Options**. In the Options dialog, click **Data Functions**, and make sure **Use locally-installed TIBCO Enterprise Runtime for R engine** is selected.

Perform this task from your Spotfire Analyst installation.

Procedure

1. From the menu, click **Tools** > **TERR Tools**.
2. Click **Launch TERR Console**.
   The TERR engine console that is included in the Spotfire Analyst `Modules` folder is displayed in a separate window.
3. At the prompt, type `library(packagename)` to load the package.
   The package is loaded to the package directory.
4. Type or paste the script or example using the package functions that you want to run.

Testing a package example

```
#install the package
install.packages("survival")
#load the package
library(survival)
#run the example
Surv(heart$start, heart$stop, heart$event)
```

Getting Package Help

Each package you download or build should have help files. If you are writing data functions or developing analyses that use the functions in a package, you might want to see Help topics associated with them.

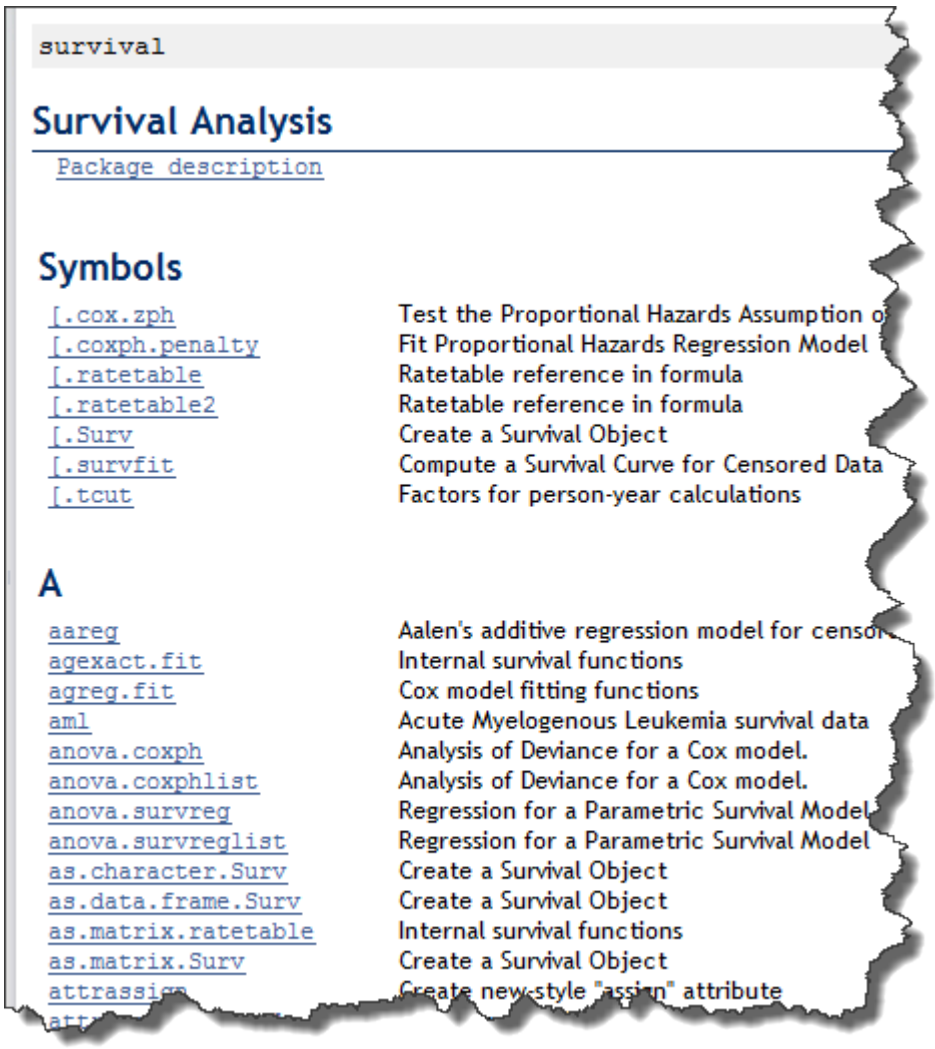
Procedure

1. In Spotfire, open a sample DXP file.
2. From the menu, click **Tools** > **TERR Tools**.
3. Click **Open TERR Language Reference**.

   The language reference is included in the installation. The links to the technical guides and readme files open these documents on the TIBCO documentation website https://docs.tibco.com/products/tibco-enterprise-runtime-for-r.

   A web browser launches and displays the landing page with links to the documentation for TERR.
4. Under **Reference**, click **Included Packages**.
5. In the resulting **Available Packages** page, note that your package is listed. Click its name.
6. From the resulting page, select the function for which you want help.



Removing a Package from a Spotfire Installation

You might decide that you no longer need a package in your Spotfire installation.

Perform this task using Spotfire Analyst.

Take care to remove only packages that you have downloaded and installed. If you remove a package that is installed with TERR, you could cause serious problems with the installation. Likewise, if you remove a package that was distributed by the Spotfire Server, any scripts or data functions that use the package will cease to work, and you will not be prompted to download and install it by the Spotfire Server distribution mechanism. See Troubleshooting TERR and Spotfire packages for more information.

Procedure

1. From the menu, click **Tools** > **TERR Tools**.
2. In the TERR Tools dialog, click the tab **Package Management**.
   If you are working through a proxy, select the check box Use **IE Proxy Settings**.
3. Review the list under **Installed Packages**.
4. Select the package to remove, and then click **Remove**.
   The package is removed from the list and from your installation.

The Spotfire SPK

A Spotfire SPK is usually created and tested by developers to package and deploy third-party extensions to the Spotfire Server, which can then be distributed to Spotfire Analyst users, or distributed to the Spotfire Server node for use by another service.

Even though they are both called “packages”, the R package and the Spotfire package (SPK) are different.

- The R package (usually downloaded from a repository) contains specialized R functions.
- The SPK is a means to deploy extensions to the Spotfire Server, which either distributes its contents to Spotfire Analyst users, or installs for a service, such as the TERR service, to use from the Spotfire Server node.

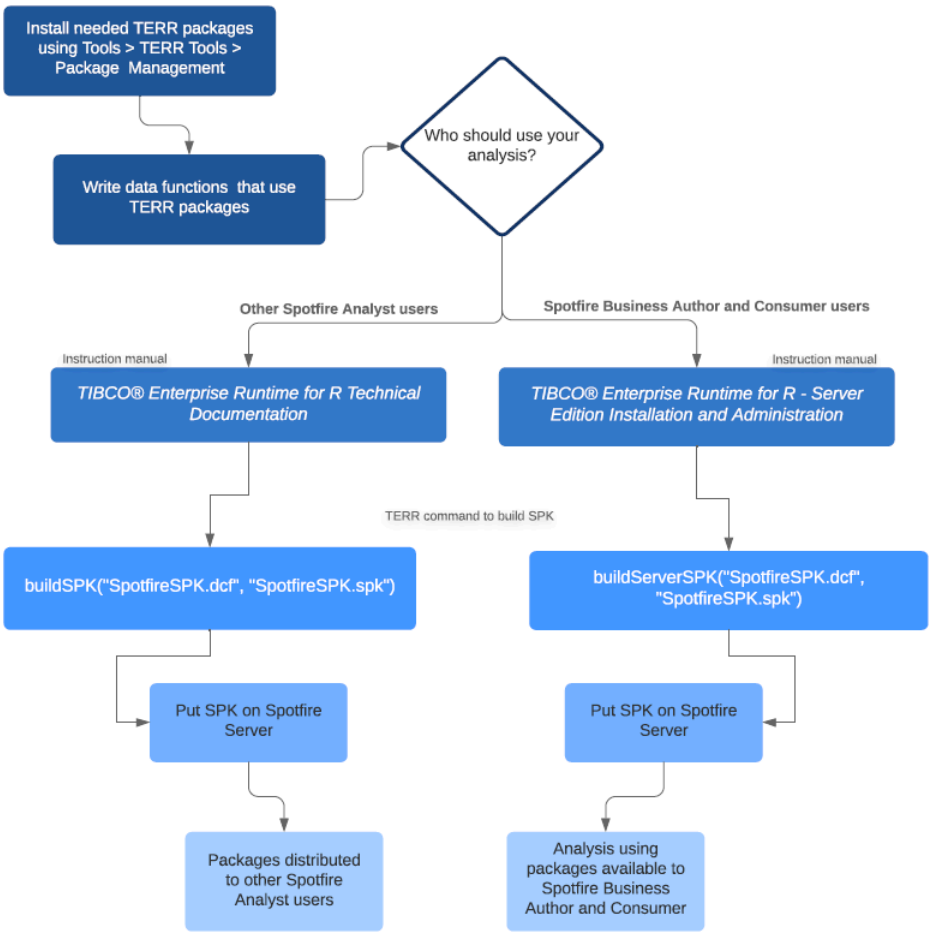
To make it easy to create SPK files containing R functions, TERR contains an R package, SpotfireSPK, that has two functions:

- `buildSPK`, which creates an SPK containing packages suitable for distribution to other Spotfire Analyst clients to create analyses using data functions.

  This use case applies only to TERR and not TERR service, so it is not covered in the TERR service documentation. For more information about creating packages to be distributed to your team members, see the "Package Management" section of the *TIBCO® Enterprise Runtime for R Technical Documentation*.
- `buildServerSPK`, which creates an SPK containing packages suitable for distribution to a Spotfire Server for use by the TERR service.

This documentation does not cover the Spotfire SPK in general, only those built using the SpotfireSPK package and containing R packages. For more general information about extending Spotfire using the SPK, see the Spotfire SDK documentation available at docs.tibco.com.



Package Workflow

Using packages with the TERR engine, Spotfire, and Spotfire Statistics Services is a two-part process.

1. Stage the packages (including developing or downloading, testing, and deploying).
2. Distribute and use the packages.

Stage the Packages

Displayed here is a high-level outline of the processes of staging packages, including deploying them to a Spotfire Server package repository and, optionally, to a Spotfire Statistics Services package repository.

To stage and deploy a package in the Spotfire Server cluster and optionally the Spotfire Statistics Services cluster, follow the pictured workflow. These steps describe the numbers in the image.

1. From your local TERR console, produce or download the R language packages to deploy and use. (Open the console from the Spotfire Analyst installation, from the **Tools** > **TERR Tools** menu.)
2. Using your local TERR engine, test the package.

3. From your local TERR engine, load the SpotfireSPK package, and then call the function to build the SPK (passing in the path to the Debian Control File (.dcf), the SPK package name, the SPK and any additional parameters you require).

   - To distribute packages to other Spotfire Analyst clients, call `buildSPK`.
   - To install packages on the Spotfire Server for TERR Service to use, call `buildServerSPK`.

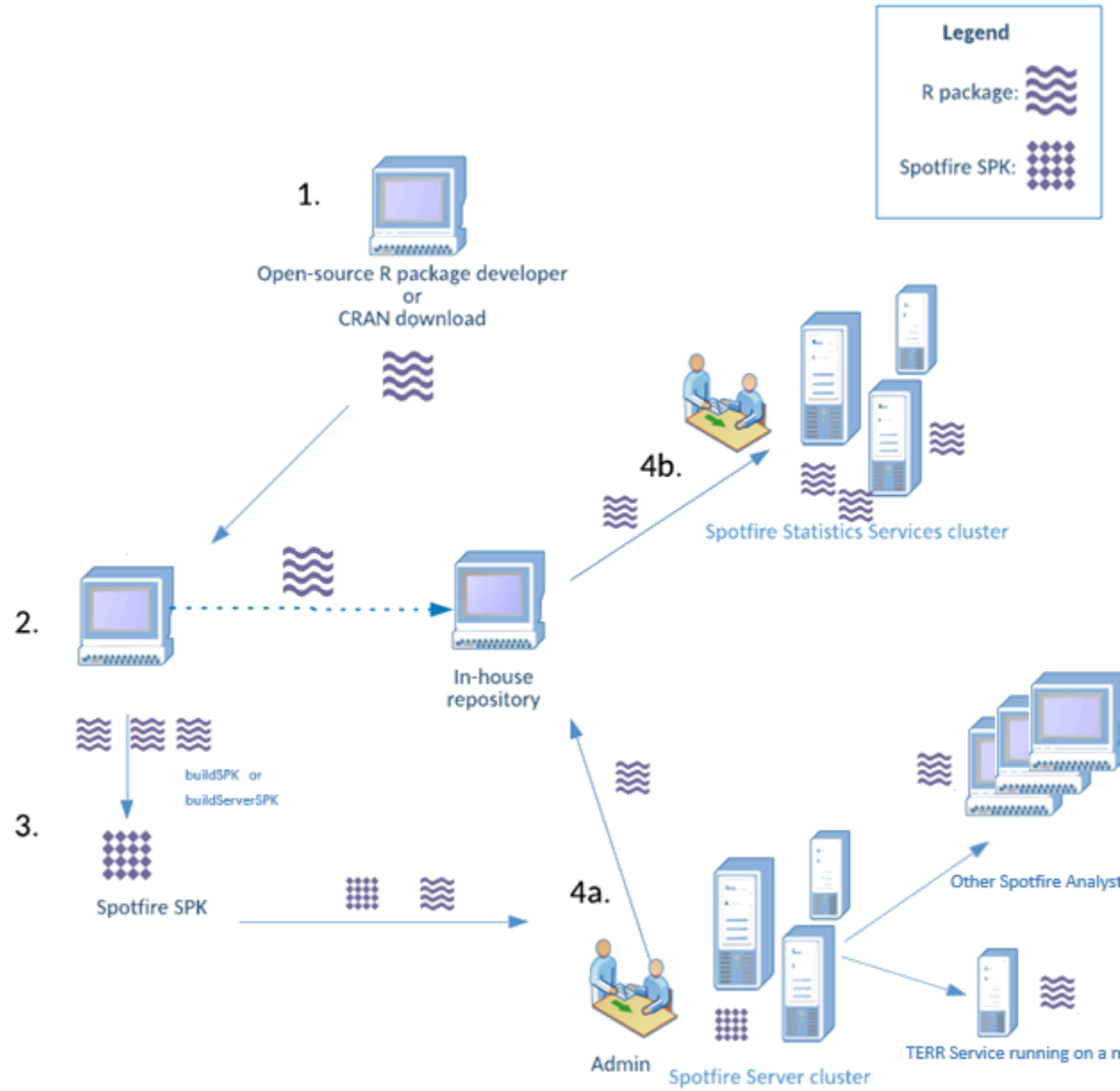
   For more information, see the section Creating the Spotfire SPK.

4. Hand off both the SPK and the R package to your Spotfire Server administrator. (Optionally, if you have the permissions, you can upload the R package to an in-house repository using the drat package.)

   a. The administrator places the SPK on the Spotfire Server for either distribution to other Spotfire Analyst clients, or to a node running TERR Service. If required, using the TERR console, the administrator can copy the package to the in-house repository.

   b. If you have Spotfire Statistics Services in your Spotfire Server deployment, then the administrator logs in to Spotfire Statistics Services and, using the TERR console, installs the R package from the in-house repository to Spotfire Statistics Services.

Optionally, the administrator can use the Eclipse plugin **TSSS Connector** to install the package to Spotfire Statistics Services. (For more information, see the section Uploading a package using TSSS Connector.)



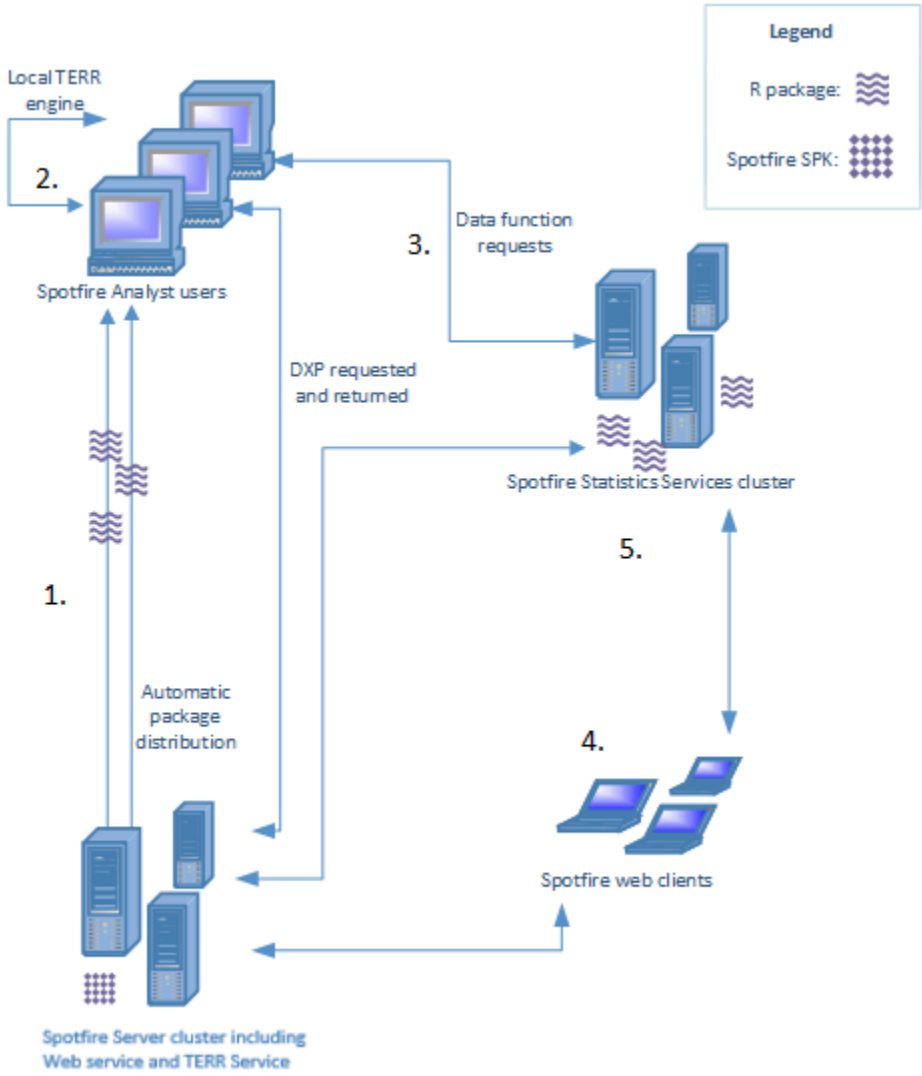
Distribute and Use the Packages

The following image provides a high-level picture of package distribution and use after it has been placed on TIBCO Spotfire® Server and TIBCO Spotfire® Statistics Services.

After the packages are placed on the Spotfire Server and, optionally, on TERR Service (or Spotfire Statistics Services), they are automatically distributed to Spotfire analysts for use in their advanced analytics, and they are available to Spotfire web client users who access Spotfire visualizations through a browser. These steps describe the numbers in the image.

1. When the Spotfire users launch Spotfire Analyst, they are notified by the server that a new distribution of the package is available. They accept the update.
2. If this Spotfire Analyst user has the Data Function license, he can run data functions using the packages and the local TERR engine.

3. Spotfire web client users can run TERR data functions remotely using the TERR engine through the TERR Service, or optionally, Spotfire Statistics Services.
4. Spotfire web client users access the Spotfire DXP (stored in the Spotfire Library) through the Spotfire Server.
5. If the Spotfire DXP includes data function(s), the Spotfire web client can access TERR Service or Spotfire Statistics Services to run the data function using the TERR engine, and then return the results to the Spotfire web client. The results are rendered in the Spotfire web client browser.



Obtaining the SpotfireSPK Toolset

Using the SpotfireSPK package, you can generate a Spotfire SPK containing your R or TERR packages, and then automatically distribute them to Spotfire Analyst installations in your organization.

You must have installed Spotfire Analyst. You must have a license to use TERR in your Spotfire Analyst installation.

Procedure

1. From the Spotfire Analyst menu, click **Tools** > **TERR Tools**.
2. On the **Engine** tab, click **Launch TERR Console**.
   The TERR console in the Spotfire installation is displayed.

3. At the console prompt, type `library("SpotfireSPK")`.
   The package is loaded in the TERR session, and you are now ready to build the SPK to contain the TERR-compatible binary packages.
4. Review the Help for the package functions, `buildSPK` and `buildServerSPK`, by typing the following in at the command-line in the TERR console:
   `?buildSPK` or `?buildServerSPK`
   - `buildSPK` builds an SPK that is distributed to other Spotfire Analyst users.
   - `buildServerSPK` builds an SPK that is installed on the Spotfire Server for the TERR Service to use.

Creating the Spotfire SPK for Other Spotfire Analyst Users

Practice building an SPK that contains two useful packages: boot and MASS, which you can install from the repository.

Prerequisites

You must have access to the TERR engine, either through your Spotfire installation, or the stand-alone TERR console.

This example builds an SPK for distributing a package to other Spotfire Analyst users. For an example of building an SPK for distributing a package for use by the TIBCO® Enterprise Runtime for R - Server Edition Installation and Administration Guide on the TIBCO Docsite.

The survival R package is compatible with TERR; however, note that some of its examples might use the `plot` function, which is not supported in version 6.1.

When you prepare an SPK to be deployed to the Spotfire Server, remember the following rules:

- The Spotfire SPK can contain as many R packages as you need.
- You can have as many SPK files in a deployment area as needed.
- You can have different SPK files in different deployment areas.
- To distribute an updated Spotfire SPK to the Spotfire Analyst installations using the server, the Spotfire SPK must have its `BuiltVersion` incremented.

This example walks you through installing a package, generating the list, installing your package. and making sure it works as expected.

Procedure

1. Load the Spotfire SPK package.
   `library(SpotfireSPK)`
2. Install a package to be in your Spotfire SPK file.
   `install.packages(c("boot","MASS"))`
   The package survival is downloaded from the repository.

   If you want to install a package from a different repository, specify it using the `repos` argument. See Installation Options for Packages on page 18.
3. Generate the Debian Control File (`DCF`).
   The DCF contains the package list to build into the Spotfire SPK.
   `writeLines("Packages: boot,MASS", "MySpotfireSPK1.dcf")`

   See Spotfire SPK versioning for information about creating the DCF and versioning the SPK.

4. Build the SPK.

In this example, you are building the SPK without passing any arguments for certificates or passwords, and the resulting output specifies that your resulting SPK is unsigned. If you do not include the arguments `certificate` and `password` in your `buildSPK` function, when the package is distributed, Spotfire Analyst users see a message warning of an unsigned file, and they are prompted to accept or reject the installation. This message appears for every update of the unsigned package. See your server administrator for a certificate and password to include.

```
buildSPK("MySpotfireSPK1.dcf", "MySpotfireSPK1.spk")
```

```
Note that "MySpotfireSPK1.spk" will not be signed.
Status information
     Done.
```

5. Print the new list file.

```
cat(readLines("MySpotfireSPK1.dcf"), sep="\n")
```

```
Packages: boot,MASS
Built: TERR 6.0.0; (includes date and time)
BuiltFile: MySpotfireSPK1.spk
BuiltName: TIBCO Enterprise Runtime for R Packages
BuiltId: F13B9A7E-783A-432a-8676-42FCD3022D70
BuiltVersion: 1.0.0.0
BuiltPackages: MASS (>=7.3-51.5),boot (>=1.3-24)
```

(Your output will vary.)

6. Browse your computer for the SPK file and the DCF file.

By default, both of these files are written to the your `Users\`*`user name`* directory.

As of TERR 5.1.0, you can have multiple SPK packages deployed to your Spotfire Server. Use the SPK's DCF list to keep track of the packages in the SPK. We recommend you keep the lists where you can update them as necessary. See Packages deployed for a small group for more information.

Spotfire SPK Versioning

You create the SPK (`.spk`) file containing the packages you want to distribute to others using Spotfire Analyst, or to install on a service running on a Spotfire Server node. You might need to change or update the packages that you distribute, which requires changing the version of the Spotfire `.spk` file.

You can create or change a Spotfire `.spk` file using TERR or from a text editor. The TERR functions `buildSPK` and `buildServerSPK` create the Spotfire `.spk` file using the versioning rule details for the following tasks.

The SPK property `BuiltVersion` is NOT the same as the package version. That information is stored in the package `DESCRIPTION` file. `BuiltVersion` is always specified as four components (*`N.n.n.n`*).

You can learn more about the TERR functions `buildSPK` and `buildServerSPK` by reading their help files. See Obtaining the SpotfireSPK toolset for more information.

As of TERR version 5.1, the SPK you create by calling `buildSPK` or `buildServerSPK` also includes Imports and Depends packages.

Task	Example	Notes and Results

Create a new DCF for a new SPK containing packages to put on the Spotfire Server.	From the TERR console, install the required packages, and then create the SPK.
	`#load the library containing`
	`#the tools`
	`library(SpotfireSPK)`
	`#install the packages to go`
	`# into the SPK`
	`install.packages(c("plyr","zoo"))`
	`#create the DCF containing`
	`# the package names`
	`writeLines("Packages: plyr,zoo", "`
	`ServerSPK.dcf")`
	`#create the SPK to upload to the`
	`# Spotfire Server`
	`buildServerSPK("ServerSPK.dcf", "`
	`ServerSPK.spk")`
		A DCF with the name `ServerSPK.dcf` is created containing the package names listed in the `text` argument of `writeLines`, along with their required and dependent packages. The specified packages are included in the SPK.
		The SPK `BuiltVersion` number is listed in the DCF as follows.
		`"BuiltVersion: 1.0.0.0"`
		 
		You can see the `BuiltVersion` by opening the DCF in a text editor, or by calling `readLines`:
		`readLines("ServerSPK.dcf")`

Add packages to an existing SPK to put on Spotfire Server.	Using a text editor, open the existing DCF, and add to the package names to the list of the (installed) packages to put in the SPK. Do not edit any other part of the DCF.
	`Packages: plyr,zoo,caret,forecast`
	From the TERR console, create the SPK.
	`#create the SPK to upload to the`
	`# Spotfire Server`
	`buildServerSPK("ServerSPK.dcf", "`
	`ServerSPK.spk",`
	`spkName = "Official Package List")`
		The DCF with the name `ServerSPK.dcf` is overwritten, and the packages you manually added to the DCF are added to the SPK.
		The SPK `BuiltVersion` is changed to the next minor version number and is listed in the DCF. For example:
		`"BuiltVersion: 1.1.0.0"`

Remove a package from the existing SPK (and subsequently from Spotfire Server).	Using a text editor, open the existing DCF, and remove the unwanted package names from the list the names of the packages to put in the SPK. Do not edit any other part of the DCF.
	`Packages: plyr`
	From the TERR console, create the SPK.
	`#create the SPK to upload to the`
	`# Spotfire Server`
	`buildServerSPK("ServerSPK.dcf", "`
	`ServerSPK.spk",`
	`spkName = "Official Package List")`
		The DCF with the name `ServerSPK.dcf` is overwritten, and only the packages remaining in the DCF are included in the SPK.
		The SPK `BuiltVersion` is changed to the next major version number and listed in the DCF. For example:
		`"BuiltVersion: 2.0.0.0"`

<table>
<tr><th>Task</th><th>Example</th><th>Notes and Results</th></tr>
<tr><td>Assign a specific <code>BuiltVersion</code> number to an SPK.</td><td>From the TERR console, install the required packages, and then create the SPK.<pre>#load the library containing
# the tools
library(SpotfireSPK)

#install the required packages
install.packages(c("plyr","zoo"))

#create the DCF containing
the packages
writeLines("Packages: plyr,zoo", "
ServerSPK.dcf")

#create the SPK to upload to the
Spotfire Server,
#passing in the version number.
#This argument is a character string
 or a
#numeric_version object containing
 four components
buildServerSPK("ServerSPK.dcf", "
ServerSPK.spk", version="1.2.3.4")</pre></td><td>The DCF with the name <code>ServerSPK.dcf</code> is created (or overwritten if it already existed) , and the packages specified in the DCF are included in the SPK.

The SPK <code>BuiltVersion</code> is set to the value passed in for the argument version. For example:<pre>"BuiltVersion: 1.2.3.4"</pre></td></tr>
<tr><td>Generate a new DCF with the same name.</td><td>From the TERR console, install the required packages, and then create the SPK.<pre>#load the library containing
# the tools
library(SpotfireSPK)

#install the packages to go into
the SPK
install.packages(c("forecast","caret"
))

#create the DCF containing the
names of the installed
packages
writeLines("Packages: forecast,caret"
, "ServerSPK.dcf")

#create the SPK to upload to the
Spotfire Server
buildServerSPK("ServerSPK.dcf", "
ServerSPK.spk")</pre></td><td>If you use this method to overwrite an existing DCF, the version number is still set to <code>1.0.0.0</code>; therefore, Spotfire Server does not register the package as a new one, so it does not distribute the new packages to the users.

The DCF with the name <code>ServerSPK.dcf</code> is overwritten, and the package names listed in the <code>text</code> argument of <code>writeLines</code>, along with their required and dependent packages, are included. The packages specified are included in the SPK.

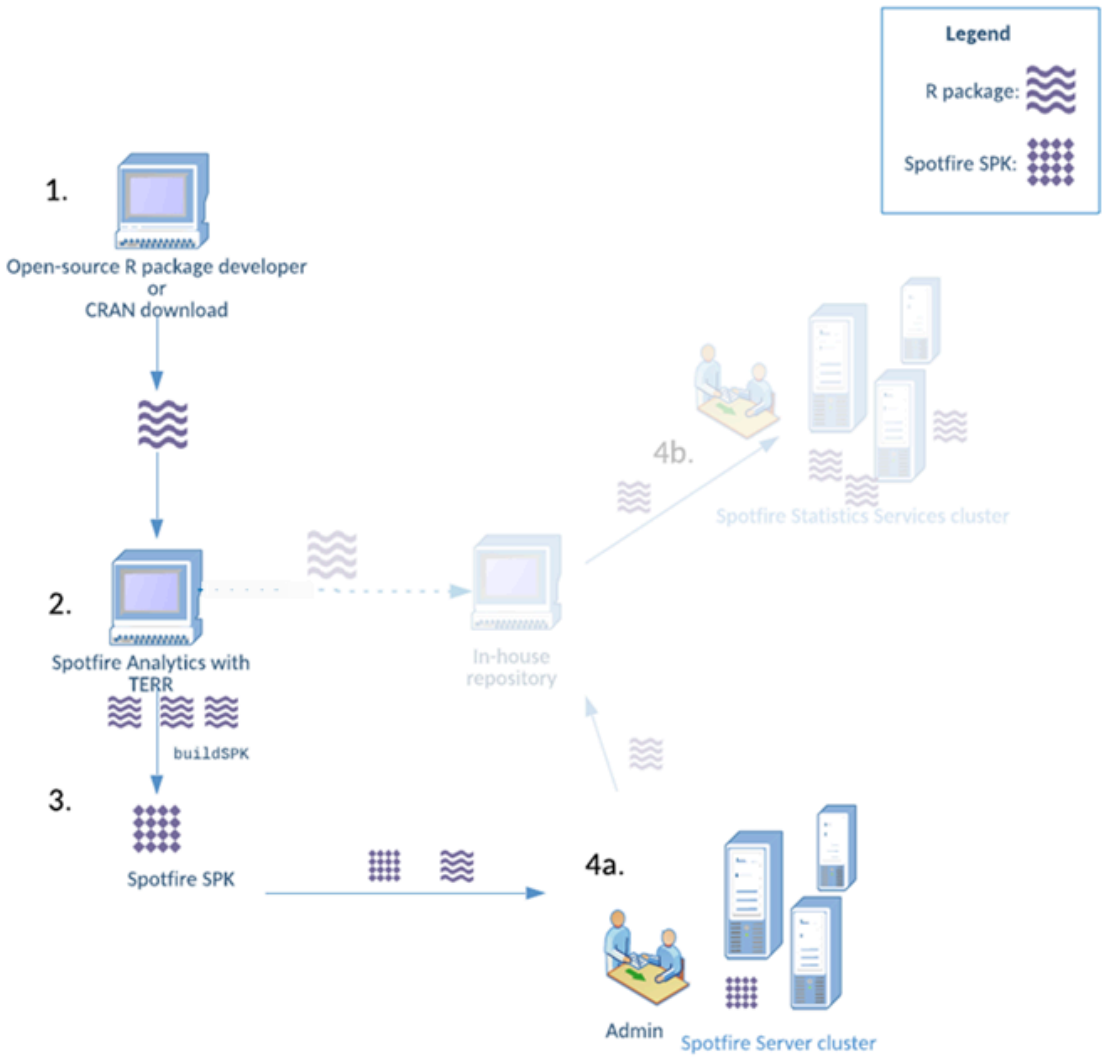
The SPK <code>BuiltVersion</code> number is listed in the DCF as follows:<pre>"BuiltVersion: 1.0.0.0"</pre></td></tr>
</table>

Deploy the SPK to the Spotfire Server

The SPK file created by `buildSPK` or `buildServerSPK` is deployed to the Spotfire Server the same way any SPK is deployed.

Using the Administration Console tools, the Spotfire Server administrator places the SPK in the Spotfire Server distribution area (4a).

- For packages built with `buildSPK` (for distribution to other users), when Spotfire Analyst users start the application, they are informed that an upgrade is available. (Alternatively, the server can be configured to force an upgrade so the new SPK is distributed automatically.)
- For packages built with `buildServerSPK`, the new packages are made available to the service.



If you did not provide a certificate when you build the SPK, the users are asked to accept the unsigned file. See your server administrator for more information, or to get a copy of a certificate and its password.

Packages Deployed for a Small Group

The package curator can deploy packages to a small group, rather than to an entire organization.

As of TERR version 5.1, a Spotfire Server deployment area can contain multiple SPK files. However, when you update or add an SPK to the primary deployment area, then the packages in that SPK are distributed to all users connected to the deployment area on the server. The package curator might want to create a small, contained package or set of packages for testing. To distribute a set of packages to a specific set of users, you can place the SPK file on a separate Spotfire Server deployment area. Members of a group connected to that deployment area have the packages deployed only to their Spotfire Analyst installations.

To deploy a package, or a set of packages in an SPK to a limited group, Ask the Spotfire Server administrator to create a new deployment area on the Spotfire Server and grant access to this deployment area to the designated group.

When the designated users open Spotfire and connect to the test deployment area, their installation is updated to the version of Spotfire with the package, and then they can use it with the local engine.

Even when you test a new package, make sure to follow your organization's rules to keep official package versions synchronized.

Spotfire Package Maintenance

Package maintenance includes updating package versions, removing obsolete packages, or adding new packages to the SPK distribution. It also includes ensuring that everyone using a package is using the same version.

We recommend that you assign a person in your organization the task of maintaining the R packages and versions distributed using the Spotfire SPK deployment mechanism, as well as those uploaded to Spotfire Statistics Services. (See Curator role for additional guidance.)

If you need to add one or more binary packages to the SPK, or if you need to update an existing package, you can recreate the `.spk` file as described in Creating the Spotfire SPK.

Redeploying an SPK of the same name overwrites the SPK currently in a deployment area. Spotfire Analyst users who start a session that connects to that deployment area are prompted to update.

If you have analysts running analyses using the local TERR in Spotfire Analyst, be sure to inform them of any changes to packages distributed to the team by Spotfire Server.

Install Packages on Spotfire Statistics Services

You can install packages onto the Spotfire Statistics Services server. If you want to use packages from a repository on the internet (such as MRAN or CRAN), you need a computer with a connection to the internet.

If you use this method in a cluster environment, you must upload the same package on each server in the cluster.

- If you have administrative privileges, then you can log on to the Spotfire Statistics Services server with administrative credentials, and then use the TERR function `install.packages` to upload packages from the repository. See Uploading a Package to Spotfire Statistics Services From a Repository on page 121 for more information.
- If you have administrative privileges, then you can copy a package from an in-house computer to the Spotfire Statistics Services server. See Uploading a Package to Spotfire Statistics Services from Another Computer on page 124 for more information.
- If you have Eclipse installed on your computer, and you can get access to the Spotfire Statistics Services server, then you can use the tool TSSS Connector to connect to Spotfire Statistics Services and install packages to either a single server deployment of Spotfire Statistics Services or a cluster. See Uploading a Package Using TSSS Connector on page 126 for more information.

After installing packages on Spotfire Statistics Services, you can validate the upload.

If you are using TERR with Spotfire Statistics Services, be aware that by default, certain Spotfire Statistics Services capabilities are disabled. If you created or used functions in a previous release, or if you install packages that contain functions that TERR deems to be potentially malicious, you might find that they do not work as expected. Any expression that TERR determines to be potentially malicious is disabled. For more information, see the *Spotfire Statistics Services User's Guide* and talk to your server administrator.

Uploading a Package to Spotfire Statistics Services From a Repository

If you have access to the internet from your installation of Spotfire Statistics Services, you can log on to the server and install packages directly.

Perform this task on the computer where Spotfire Statistics Services is installed.

If your Spotfire Statistics Services installation is a cluster, you must perform this task on every computer in the cluster.

Prerequisites

You must be able to log on with administrative credentials to the server where Spotfire Statistics Services is installed.

You must have built the package archive or downloaded a compatible package from a trusted web site or package repository.

- The repositories contain binary packages (for Windows) and source packages (for Linux and Windows). You can easily install most binary and source packages in TERR. If you have problems building from source, then build the packages using open-source R before installing them into TERR. Note that TERR does not build binary packages from source packages that contain Java source code.

Platform	Package type	Notes
Linux, Windows	Binary	Call `install.packages(pkgname)`. TERR installs the binary package into your specified package directory.
Linux	Source; no Java code, no C/C++ or Fortran code	Call `install.packages(pkgname)`. TERR builds the source package into a binary package and installs it into your specified package directory.
Linux, Windows	Source; C/C++ or Fortran code (no Java code)	On Windows, first you must install the Rtools utilities package, which is maintained by Duncan Murdoch, and then update your PATH to specify the location of the utilities. 1. If you have not already done so, install the package rinclude by calling `install.packages(rinclude)` 2. Call `install.packages(pkgname)`. See Installation Options for Packages on page 18 for information on repositories accessed by `install.packages`. TERR builds the source package into a binary package and installs it into your specified package directory. If the package does not build and install, then try building it with open-source R, and then installing the binary as described here.
Linux	Source; Java code	1. Build the package using open-source R tools for building packages from source. The tools compile the source code to create the binary package. 2. Call `install.packages(pkgname)`.

  See the help for `install.packages(pkgname)` for more information.

Due to changes in open-source R version 3.5 and resulting compatibility changes in TERR 5.0, packages that are built with a version of TERR prior to 5.0 must be rebuilt.

- To install a binary package from a repository, always call `install.packages(pkgname)` from TERR. The `install.packages` function finds the correct binary version in the repository for

your version of TERR. Manually downloading the binary package from CRAN can result in errors when you use it with TERR.

- To install a package from source, try installing it first with TERR (with `install.packages` in TERR or with `TERR CMD INSTALL` from a command line).
- To install a package from source that you cannot build with TERR, install the package with the version of open-source R tested with TERR.

Spotfire Statistics Services accepts `.zip` archives (Microsoft Windows servers only) or `tar.gz` archives (Linux servers only).

- If you plan to download an R package from CRAN to run on the TERR engine, we recommend that you check the package compatibility list for the version of TERR on your installation of Spotfire Statistics Services, and test the package with the version of TERR that is in your installation of Spotfire Statistics Services. See the Documentation page for TERR for more information.

Procedure

1. Log on with administrator credentials to the computer where Spotfire Statistics Services is installed.
   If you do not have administrative credentials, ask your server administrator to help you.
2. Open the `bin` directory of the server's engine.
   This path should look like *`SPSERVER_HOME`*`/engines/`*`eng`*`/bin` where *`SPSERVER_HOME`* is the installation and server context, and *`eng`* is the language engine, such as open-source R or TERR.
   Windows example: `C:\Program Files\TIBCO\statsvcs1010\TERRServer\engines\Terr\bin`
3. Right-click the language engine executable, and from the menu, click **Run as administrator**.
   A console for the language starts.
4. At the console command prompt, type the command `install.packages("`*`pkgname`*`")`, where *`pkgname`* is the package you want to install.
   If your installation of Spotfire Statistics Services is configured to be able to install packages from the internet, then `install.packages()` installs from MRAN. If you want to install a package from another repository, such as an in-house package repository, provide the path using the `repos` argument.

   By default, Spotfire Statistics Services is configured to restrict access to file I/O, downloading from the internet, and any other operation that can be considered a potential malicious action. If your installation of Spotfire Statistics Services is not configured to be able to install packages from the internet, then see your system administrator for guidance on how to get the packages you need.

   Example: `install.packages("h2o")`
   The current version of the package and all of the packages it requires to work are downloaded and installed into the directory *`SPSERVER_HOME`*`/engines/`*`eng`*`/library` from the specified repository.

What to do next

Validate the package upload.

Uploading a Package to Spotfire Statistics Services from Another Computer

If your installation of Spotfire Statistics Services is not connected to the internet, but you have administrative privileges to log on to the server, you can still install packages on it.

You can upload two types of packages by copying them to Spotfire Statistics Services.

- You can upload packages that have been created and built for your Spotfire Statistics Services installation.

- You can use another computer that is running the same operating system, and that is connected to the internet to download a package from a repository.

Prerequisites

You must have either a built binary package or access to a computer on the internet.

Binary (built) packages are required. Source packages must be built before they can be uploaded to Spotfire Statistics Services.

Spotfire Statistics Services accepts `.zip` archives (Microsoft Windows servers only) or `tar.gz` archives (Linux servers only).

You must have administrative credentials to log on to the server where Spotfire Statistics Services is installed (or you must give the package archive to a server administrator to upload them).

Procedure

1. From the internet-connected computer, make sure that TERR or open-source R (and optionally RStudio) is installed.

   The computer you use to download packages must be running the same operating system as the Spotfire Statistics Services server. (That is, Windows or Linux. If Linux, it must be the same version of Linux.)

2. Start a console or, if available, an RStudio session.

   If you are running the console from Microsoft Windows, from the **Start** menu, right-click the application, and then select **Run as administrator** so you can install the package into the engine's `site-library` directory. Otherwise, you are prompted to create a personal library. If you select that option, make a note of the location so you can find the package.

3. Open the directory where the package (and any dependent packages) downloaded, and using an archiving utility, create a `.zip` or a `.tar.gz` archive that contains all files and directories included with the download.
4. Copy the resulting archive to a medium (such as a flash drive) or a network location that your server can access.
5. If you have access, log on with administrator credentials to the server where Spotfire Statistics Services is installed.

   If you do not have administrative credentials, ask your server administrator to help you.
6. Insert the flash drive into a port on the server, or from the server, map a network drive to the package archive location.
7. Browse to the installation of Spotfire Statistics Services and open the `library` directory for the appropriate language (TERR or R).

   This path should look like *`SPSERVER_HOME`*`/engines/`*`eng`*`/library` where *`SPSERVER_HOME`* is the installation and server context, and *`eng`* is the language engine, such as R or TERR.

   Windows example: `C:\Program Files\TIBCO\statsvcs1010\TERRServer\engines\Terr\library`
8. Copy the zipped archive of the package(s) from the medium or network drive to the `library` directory, and then unzip them there.

What to do next

Validate the package.

Uploading a Package Using TSSS Connector

To upload a package to Spotfire Statistics Services, if you have server administrative privileges and an Eclipse installation, you can establish a connection to Spotfire Statistics Services and upload TERR or R packages using the TSSS Connector.

Prerequisites

To complete this task, you must have completed the following steps.

1. Built the package archive or downloaded a compatible package from a trusted web site or a package repository, such as CRAN or an in-house repository.

   Binary (built) packages are required. Source packages must be built before they can be uploaded to Spotfire Statistics Services.

   Spotfire Statistics Services accepts `.zip` archives (Microsoft Windows servers only) or `tar.gz` archives (Linux servers only).

2. Installed the Eclipse integrated development environment.

   We tested versions 3.6 to 4.2.2 (Juno) of Eclipse for the remote submission tools.

3. Downloaded the TSSS Connector Eclipse plug-in from the update page on Spotfire Statistics Services.

   For detailed information about updating or installing this plugin, browse to the landing page for the Spotfire Statistics Services installation available to you, and then see the update page.

   ```
   http://SName:P#/SC/update
   ```

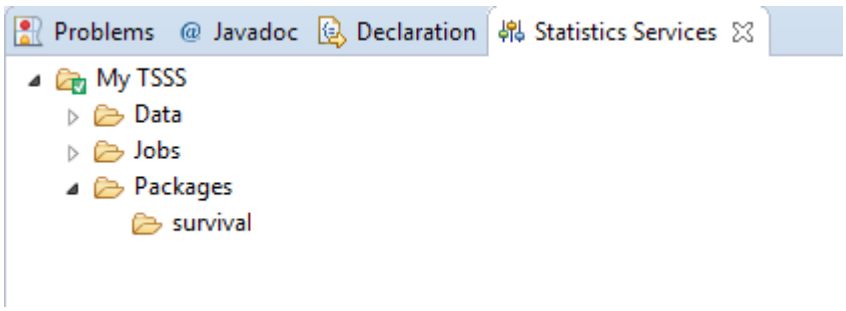
   SName is the server name, *P#* is the port number, and *SC* is the server context (for example, `http://CoTSSS:8080/TERRServer/update`).

If you are uploading a package that is also provided to Spotfire Analyst users from a Spotfire Server deployment, then the package version must match the version distributed in the SPK to the Spotfire Analyst users.

Perform this task using the **Statistics Services** plugin for Eclipse.

Procedure

1. Right-click the **Packages** folder.
2. From the menu, select **Upload Package**.
3. Supply the location of your package archive.



4. Add the archive. The package is now on Spotfire Statistics Services.

Although the package is on Spotfire Statistics Services, it is not loaded into the engine. With each call to Spotfire Statistics Services, the engine is started anew.

To use the package, you can either load the library as part of your function scripts, or you can include it in the `Packages` field of your Spotfire data function.

Maintaining a Package in TIBCO Spotfire Statistics Services

Keep the package versions in synch across your Spotfire Statistics Services deployments.

Prerequisites

You must have the Eclipse IDE and the TSSS Connector configured and installed.

After you add a package to the server, you can check its properties using the Statistics Services view in Eclipse.

Procedure

1. In the Statistics Services view, connect to the service if it is not already connected.
2. Expand the folder structure to see the packages listed in the `Packages` folder.
3. Right-click the package name, and from the menu, select **Package Properties**.

   The Properties dialog box appears, and information on the package, drawn from its `DESCRIPTION` file, is displayed.
4. Update the package as needed.
5. Follow the steps to upload the updated package.
   See Uploading a package to TIBCO Spotfire Statistics Services.

What to do next

After the package has been uploaded, follow the steps in Validating the package upload.

Validating the Package Upload

After you upload a package, run a quick validation to ensure that your package is on the server.

Prerequisites

You must have administrative privileges to perform this task.

Perform this task from the TERR console in your installation of Spotfire Statistics Services

Procedure

1. From the browser, type the following command.

   ```
   installed.packages()
   ```

   A list of installed packages is printed in the console.
2. Review the list (including the path) to ensure that the package you uploaded is installed.

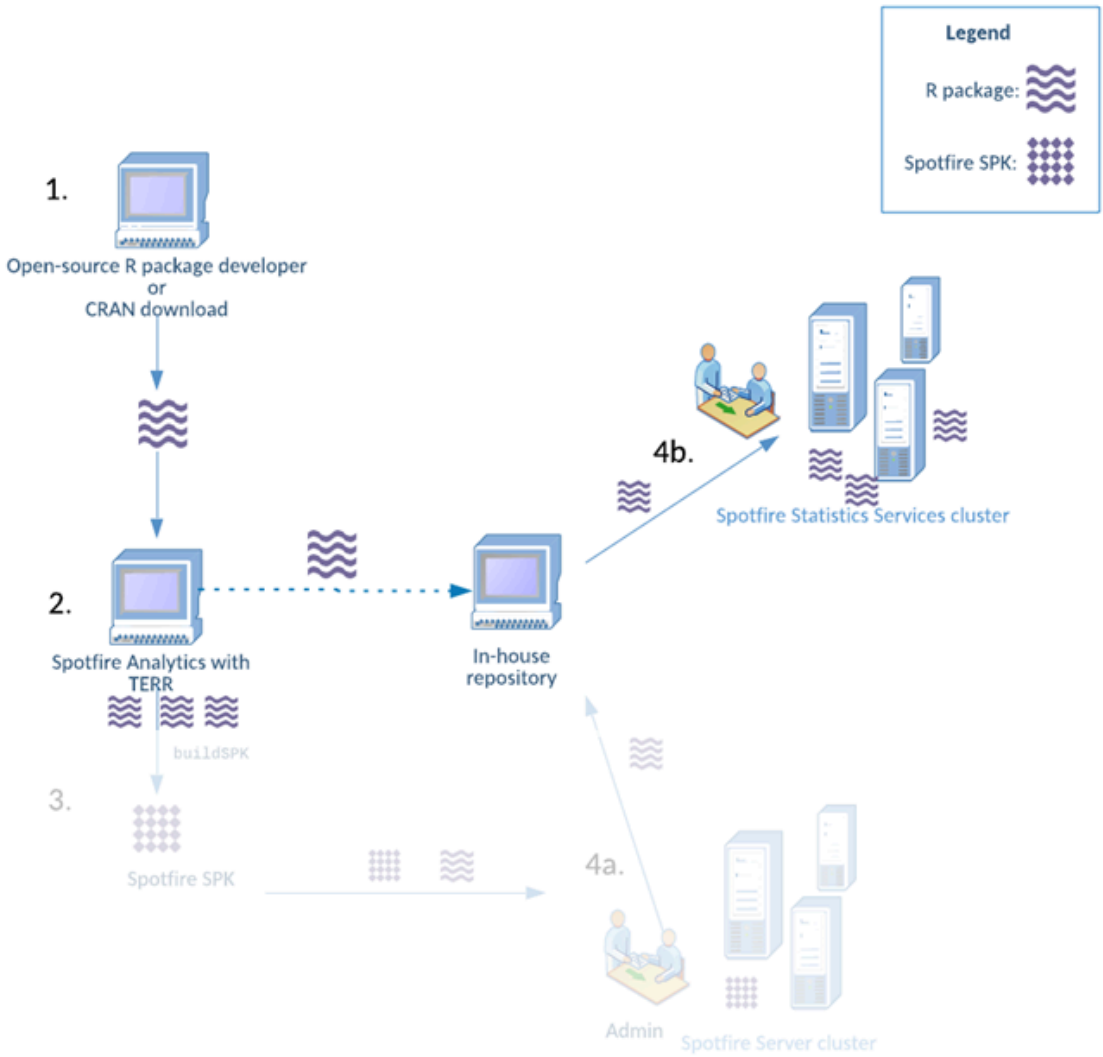
Although the package is on Spotfire Statistics Services, it is not loaded into the engine. With each call to Spotfire Statistics Services, the engine is started anew.

To use the package, you can either load the library as part of your function scripts, or you can include it in the `Packages` field of your data function.

Manage Packages Between Spotfire and Spotfire Statistics Services

You can share Spotfire visualizations that use R language packages. To share such visualizations widely in a web browser, your server configuration must include the Spotfire web client and Spotfire Statistics Services deployed and configured to work with Spotfire Server.

Spotfire Statistics Services includes a repository for the packages containing functions that can be used by Spotfire analyses. These packages must be identical to those packages distributed to the installations of Spotfire Analyst.



You can add packages to your Spotfire Statistics Services installation. For more information, see Install Packages on Spotfire Statistics Services.

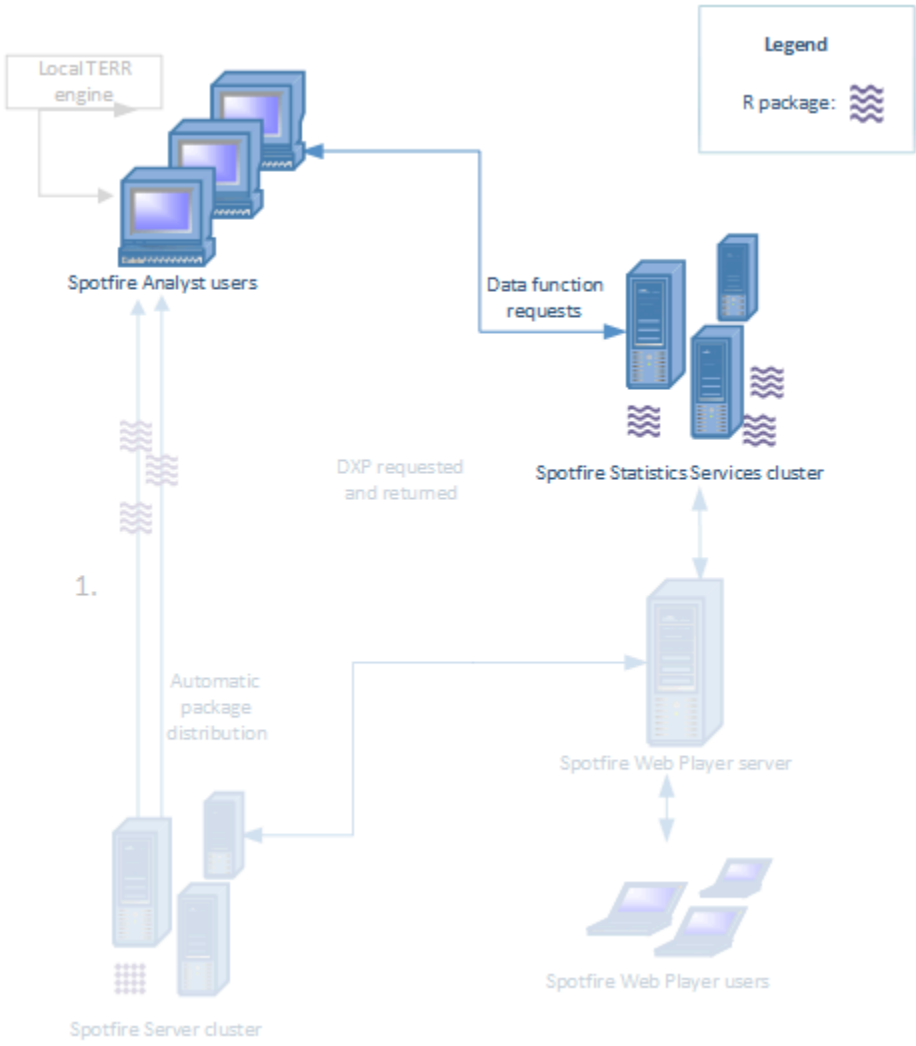
Changing the Local Engine Option

You can configure Spotfire Analyst to use the TERR engine that is installed in your organization's Spotfire Statistics Services deployment.

Prerequisites

Remember that you must have the same package version on your local installation and on the server. See your organization's package curator for help.

You can use Spotfire Statistics Services in deployments where web client users access Spotfire analyses. Changing from the local TERR engine to the one on Spotfire Statistics Services is useful for testing the analyses the Spotfire web client users access.



Procedure

1. In Spotfire Analyst, click **Tools** > **Options**.
2. In the left pane, scroll down and select **Data Functions**.
3. In the **Data Functions** pane, select **Custom URL**, and then provide the URL to the Spotfire Statistics Services (for example, `http://CoTSSS:8080/TERRServer`).
4. Clear the checkbox **Use locally installed TIBCO Enterprise Runtime for R**, and then click **OK** to accept.

The following advanced analytic tools available in the Spotfire Analyst installation always use the local engine, regardless of this setting.

- Classification modeling.
- Regression modeling (linear regression or regression tree).

Troubleshooting TERR and Spotfire Packages

If you remove a package accidentally, cannot install a package, get different results than you expect in your analysis, or cannot distribute an SPK, try these techniques to solve the problem.

I generated a new SPK using `writeLines` but it is not being distributed.

It is most likely that the SPK version number is not being properly revised. Review the rules for how the SPK version is determined in SpotfireSPK versioning for more information.

I have copied a package from CRAN but it is not working with TERR

Always make sure to use the version of open-source R that was tested with the version of TERR you are using.

Due to changes in open-source R version 3.5 and resulting compatibility changes in TERR 5.0, packages that are built with a version of TERR prior to 5.0 must be rebuilt.

- To install a binary package from a repository, always call `install.packages(`*`pkgname`*`)` from TERR. The `install.packages` function finds the correct binary version in the repository for your version of TERR. Manually downloading the binary package from CRAN can result in errors when you use it with TERR.
- To install a package from source, try installing it first with TERR (with `install.packages` in TERR or with `TERR CMD INSTALL` from a command line).
- To install a package from source that you cannot build with TERR, install the package with the version of open-source R tested with TERR.

I accidentally removed a package from my Spotfire installation that I need. How do I get it back?

We recommend never removing a package from Spotfire TERR Tools, unless you have downloaded and installed the package yourself from a repository. You must be very careful to not remove a package that is installed with TERR or distributed by the Spotfire Server using the SPK mechanism. However, should you accidentally remove a needed package, you can recover.

If you deleted a distributed package, follow these steps.

1. Open Windows Explorer and browse to *`%SPOTFIRE_HOME%`*`\Modules`.

   For example, `C:\Program Files (x86)\TIBCO\Spotfire\<`*`version#`*`>\Modules`.
2. Delete the folder `TIBCO Enterprise Runtime for R Packages_<`*`version#`*`>`.
3. Restart Spotfire and accept the prompt to update your installation.

All of the TERR packages contained in the SPK on the Spotfire Server are redistributed.

If you deleted a package required by TERR to operate, follow these steps.

1. Open Windows Explorer and browse to *`%SPOTFIRE_HOME%`*`\Modules`.

   For example, `C:\Program Files (x86)\TIBCO\Spotfire\<version#>\Modules`.
2. Delete the folder `TIBCO Enterprise Runtime for R_<`*`version#`*`>`.
3. Restart Spotfire and accept the prompt to update your installation.

The package containing TERR is redistributed.

I am generating a different result from those seen by others. What could cause this?

Check your package version numbers to make sure everyone is using the same package version. See your package curator for more information.

I am trying to use a package on Spotfire Statistics Services but it's not working.

Be aware that by default, certain functions are restricted by Spotfire Statistics Services. If you see the error `"Error: restricted call to Native[tempfile]"`, and execution of the expression is terminated, then the default restrictions have not been changed by your server administrator. Restricted behavior includes the following non-exhaustive list of operations.

- Performing any I/O to the file system or the internet.
- Loading new packages, except for the libraries included with TERR (stats, terrUtils, and so on).
- Spawning new OS processes (calling `system`).
- Calling `.Call`, which is used to call Rapi code in CRAN packages.
- Calling `.C` or `.Fortran`.
- Calling into Java using the terrJava package (which allows executing arbitrary Java methods).
- Calling any functions in the parallel package (which uses terrJava).
- Accessing any function environments in the stack above the call to `evalREX` using `sys.frame` or `parent.frame`. (This prevents malicious code from installing functions or expressions that could be executed after leaving restricted execution mode.)
- Changing the variable lookup path by setting `parent.env` of an environment, or reading or setting the environment of a closure.
- Defining S4 classes and methods using `setClass` or `setMethod`.

Be sure you are loading the package in your TERR script. Although the package is on Spotfire Statistics Services, it is not loaded into the engine. With each call to Spotfire Statistics Services, the engine is started anew. To use the package, you can either load the library as part of your function scripts, or you can include it in the `Packages` field of your data function.

I cannot install a package from TRAN on Linux because it requires another package that is not installing correctly.

Some packages customized and placed on TRAN require other packages not available on TRAN. Some of these packages cannot be installed using the TERR function `install.packages`, so the TRAN package cannot be successfully installed. If you encounter this situation, try building and installing the package using open-source R.

If you get a warning that the rinclude package is not installed, try installing that package from TRAN and trying again.

When you install a package using TERR, by default, TERR first checks for the package on TRAN, and then checks on MRAN, and then CRAN. TERR installs the first version it finds. This is different than open-source R, which installs packages according to the newest version number available on CRAN. This difference is by design, because occasionally a CRAN package update causes a break with TERR compatibility, so we make available a tested version of the package on TRAN. If you need to install one of these packages using open-source R, you can install the CRAN package from MRAN or CRAN, and then set `options()$repos` to install from only TRAN before reinstalling the package. See Specifying an older package on TRAN for more information.

Graphics in TIBCO Enterprise Runtime for R

This guide provides examples and suggestions for using TIBCO® Enterprise Runtime for R (TERR™) in Spotfire, with the RinR package, and with CRAN packages (such as htmlwidgets) that work with TERR™ to display graphics in a web browser.

TERR as a statistical programming language and runtime engine was developed to be compatible with open-source R, and to be integrated in other systems, such as KNIME, Hadoop, and RStudio. Historically, it was developed to be included with the analyst versions of Spotfire, so data visualizations using the statistical power of TERR could be created in Spotfire.

This graphical capability provided by Spotfire allowed TERR engineers to concentrate on open-source R language and package compatibility, power, and speed. However, the need to produce graphics exists, and you can take advantage of tools and packages to produce high-quality graphs with TERR.

Not all graphics functions from open-source R are supported in TERR; however, most have stub functions so calling them will not cause the program to fail. The RGB package produces an image, but because of unimplented functions, using this package causes TERR to fail. When you try your own code from the examples using the packages described here, you might see warnings about unimplemented functions.

JavaScript-Enabled Packages

You can use the TERR-compatible CRAN packages that take advantage of JavaScript to generate graphs, plots, and map visualizations, and then display them in a browser and save them as graphic images.

In this section, create a variety of graphics by loading compatible packages and calling functions.

- If your data has location information, such as a latitude column and a longitude column, and you want to create an interactive map, you can use the leaflet package.
- If your data has location information, such as latitude and longitude from around the world, and you want to create three-dimensional, interactive map graphics, you can use the threejs package.
- If your data has time and date columns, and you want to create a line graph, you can use the dygraphs package.

Creating a Plot with TERR and dygraphs

If you have time and date data, you can create a line plot showing trends with the TERR-compatible, JavaScript-enabled dygraphs package.

Perform this task from the TERR console. This example walks you through creating a line plot with data available on the internet, and then displays the results in a browser.

For information about the packages used in this example task, see the following links.

Package link	Package short description
http://www.quantmod.com/	The quantmod package provides quantitative modeling functions.
http://dygraphs.com/	The dygraphs package provides fast, flexible JavaScript charting functions.

You can perform this task from RStudio using the TERR engine. If you use RStudio, the results are displayed in the RStudio Viewer pane.

Prerequisites

TERR, access to the internet, and a browser.

Procedure

1. From the TERR console, install the quantmod and dygraphs packages.

   ```
   install.packages(c("quantmod", "dygraphs"))
   ```

   TERR checks the repository for the packages to install, and then installs them along with any packages they require.
2. Load the quantmod package.

   ```
   library("quantmod")
   ```

   The package and its required packages are loaded, and messages about any object masking are displayed, along with any warnings regarding TERR differences.
3. Call the quantmod function `getSymbols` to load the data and specify the source.

   ```
   getSymbols("ATLA013URN", src = "FRED", auto.assign=FALSE)
   ```

   In this case, the data is the Unemployment Rate in Atlanta-Sandy Springs-Roswell, GA. The source is the Federal Reserve Economic Data.

   To see more information about `getSymbols` (for example, other supported data sources and other arguments), see its help file in the quantmod package help.

   A message about the function is displayed and the data object is loaded.
4. Load the dygraphs package.

   ```
   library("dygraphs")
   ```

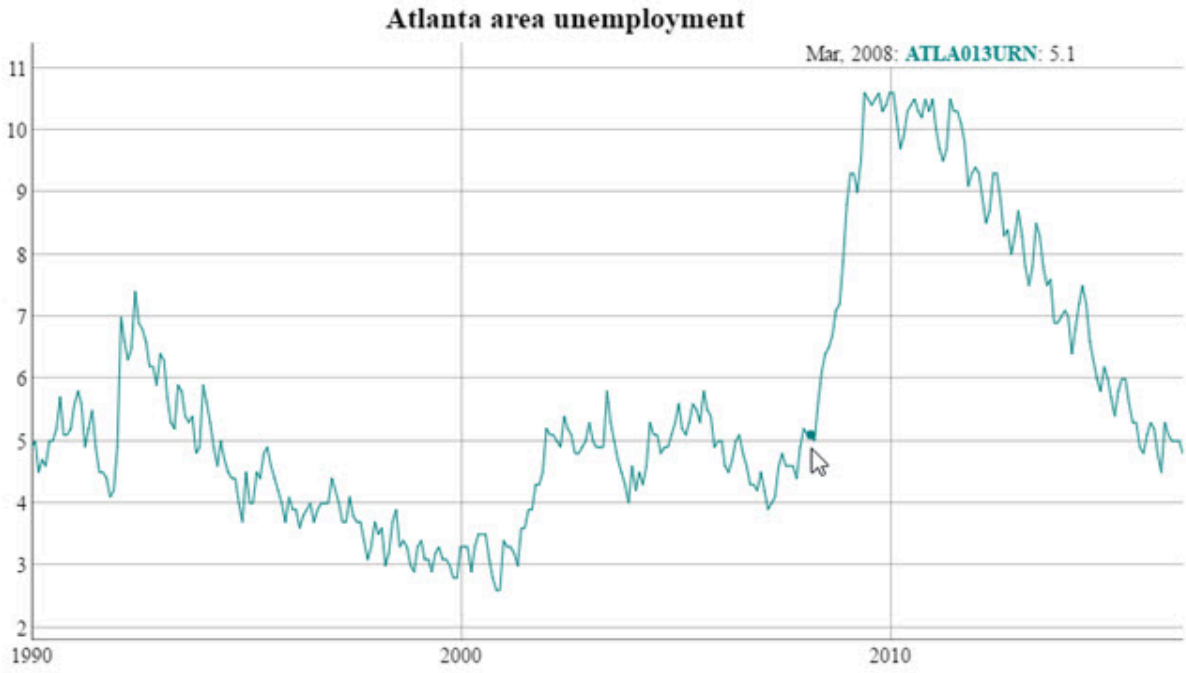
   The package and its required packages are loaded, and messages about any object masking are displayed, along with any warnings regarding TERR differences.
5. Call the `dygraph` function as follows to plot the data and display a plot title.

   ```
   dygraph(ATLA013URN, main = "Atlanta area unemployment")
   ```

Result

The default browser is launched, and a line plot showing the Atlanta area unemployment rate from January 1990 through January 2016 is displayed.

- You can resize the image. The X-axis label detail adjusts to the window size.
- You can highlight individual points on the line by hovering over the line with your mouse or pointer device. The row value for the highlighted point (in this case, the month and year, data source, and Y-axis value) is shown in the upper left of the browser window.



Mapping data with TERR and leaflet

If your data has location columns (latitude and longitude), you can retrieve data using the dplyr package, and then create a map chart using the TERR-compatible, Javascript-enabled leaflet package.

You can use RStudio to create this map chart, or you can use the TERR console and display the results in a browser. This example walks you through creating a map chart in the TERR console with data available on the internet, and then displaying the results in a browser.

The dplyr package is used to filter the original data, and the pipe operator (%>%) is used to avoid creating intermediate data objects.

If you search the internet, you can find several informative articles and samples for the packages we use in this example.

Package link	Package short description
https://dplyr.tidyverse.org/	Work with data frame-like objects, both in memory and out of memory.
https://rstudio.github.io/leaflet/	Create and customize interactive maps using the Leaflet JavaScript library.

Prerequisites

TERR, access to the internet, and a browser.

Procedure

1. From the TERR console, install the dplyr and leaflet packages.

   ```
   install.packages(c("dplyr", "leaflet"))
   ```

   TERR checks the repository for the packages to install, and then installs them along with any packages they require.

2. Call the `library` function to load the required packages.

```
library("dplyr")
library("leaflet")
```

   The packages and their required packages are loaded, and messages about any object masking are displayed, along with any warnings regarding TERR differences.

3. Load the data, and then restrict the data to Starbucks stores located in Washington state only.

```
file <- "https://opendata.socrata.com/api/views/ddym-zvjk/rows.csv"
starbucks <- read.csv(file) %>%
  filter(State == "WA")
```

   In this case, the data contains the location of Starbucks stores in the U.S. The source is the OpenData web site provided by Socrata. To see more information about `filter` (for example, filtering rows that match different conditions), see its help file in the dplyr package help.

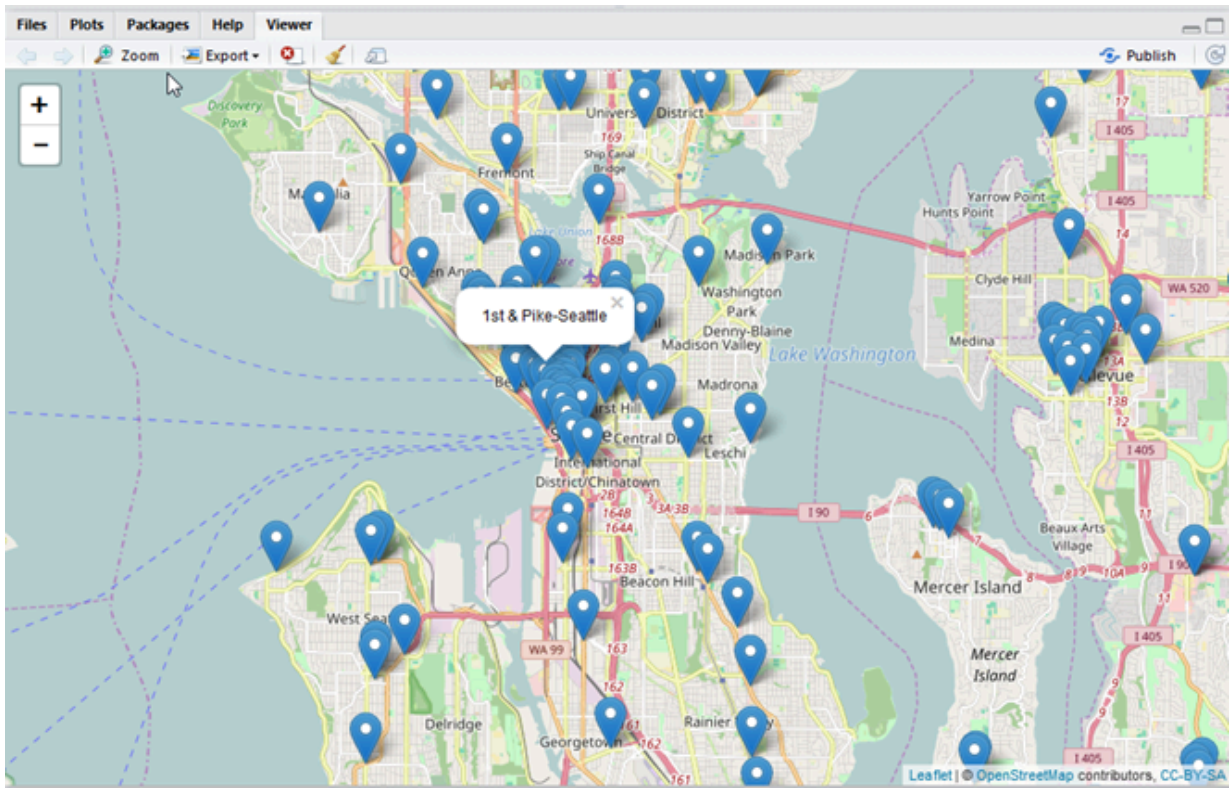
4. Use the leaflet package functions to draw a map, with the center in Seattle, and with the Starbucks store locations identified by markers.

```
leaflet() %>%
  addTiles() %>%
  setView(-122.32, 47.605, zoom = 12) %>%
  addMarkers(data = starbucks, lat = ~ Latitude, lng = ~ Longitude,
    popup = starbucks$Name)
```

The `addMarkers` function argument `popup` enables the `Name` column (containing the store name--usually named for its location) to be displayed when you select its marker.

Result

A browser opens, and a map with the center focused on the specified latitude and longitude is displayed. The `zoom` property of the map is set to the specified value. Each Starbucks store in the Seattle area is indicated by a blue marker. If you click a marker, the value of the `Name` column is displayed in a pop-up window.



What to do next

You can create create a formatted HTML document with graphed data using the rmarkdown package.

Creating a 3D Interactive Map with TERR and threejs

If your data has location columns (latitude and longitude) from around the globe, and you want the user to be able to interact with the globe, you can create a graphic using the TERR-compatible threejs package.

You can use RStudio to create this interactive globe map, or you can use the TERR console and display the results in a browser. Perform this task from either RStudio or the TERR console.

You can find samples for the threejs package used in this example on the internet. For more information, see http://www.htmlwidgets.org/showcase_threejs.html.

Prerequisites

TERR, access to the internet, and a browser.

Procedure

1. From the TERR console or RStudio prompt, install the threejs and maps packages.

   ```
   install.packages(c("threejs", "maps"))
   ```

   TERR checks the repository for the packages to install, and then installs them along with any packages they require.
2. Call the `library` function to load the required packages.

   ```
   library("threejs")
   library("maps")
   ```

   The packages and their required packages are loaded, and messages about any object masking are displayed, along with any warnings regarding TERR differences.
3. Call the utils function , passing in the data set `world.cities` contained in the maps package.

   ```
   data(world.cities, package="maps")
   ```
4. Sort the top thousand entries of the `pop` (population) column from world.cities into decreasing order, and assign the result to the object `cities`.

   ```
   cities <- world.cities[order(world.cities$pop,decreasing=TRUE)[1:1000],]
   ```
5. Create the scale for mapping the cities by calculating the percentage of size for each entry in `cities`, and then assign it to the object `value`.

   This step establishes a comparison among the largest to smallest values in the `cities` object that can be shown reasonably on the globe.

   ```
   value  <- 100 * cities$pop / max(cities$pop)
   ```
6. Call the grDevices package function `rainbow`.

   ```
   col <- rainbow(10,start=2.8/6,end=3.4/6)
   col <- col[floor(length(col)*(100-value)/100) + 1]
   ```

   This step sets the gradient of colors of the markers, and then calls the base package function `floor` to set the length.

7. Call the threejs function `globejs`, passing in the arguments.

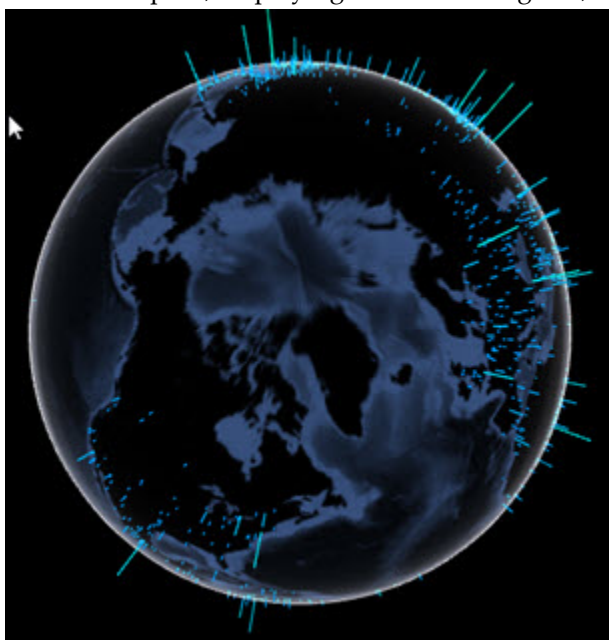
```
globejs(lat=cities$lat, long=cities$long, value=value, color=col,
    atmosphere=TRUE)
```

This step renders the globe and displays the markers for `cities`, as defined by `col`.

- `lat` is set to the lat column in the `cities` data.
- `long` is set to the long column of the `cities` data.
- `value` specifies using the percentage comparisons.
- `atmosphere` specifies to display the WebGL atmosphere effect.

Result

A browser opens, displaying an interactive globe, with markers showing cities by relative size.



Creating an Interactive Scatterplot Cloud with TERR and threejs

If you want to create an interactive scatterplot, you can use the `scatterplot3js` function from the threejs package from MRAN.

You can use RStudio to create this 100,000-point interactive scatter plot, or you can use the TERR console and display the results in a browser.

You can find samples for the threejs package used in this example on the internet. For more information, see http://www.htmlwidgets.org/showcase_threejs.html.

Prerequisites

TERR, access to the internet, and a browser.

Procedure

1. From the TERR console or RStudio prompt, install the threejs package.

   ```
   install.packages("threejs")
   ```

   TERR checks the repository for the packages to install, and then installs them along with any packages they require.
2. Call the `library` function to load the required packages.

   ```
   library("threejs")
   ```

3. Assign the value `10000` to `N1`, and the value `90000` to the name `N2`.

   ```
   N1 <- 10000
   N2 <- 90000
   ```

4. Assign the point distribution to the `x` axis.

   Set a random normal distribution for both `N1` and `N2`, with the standard deviation for `N1` of `.05`, and the standard deviation for `N2` of `2`.

   ```
   x <- c(rnorm(N1, sd=0.5), rnorm(N2, sd=2))
   ```

5. Assign the point distribution to the `y` axis.

   Set a random normal distribution for both `N1` and `N2`, with the standard deviation for `N1` set to `.05`, and the standard deviation for `N2` set to `2`.

   ```
   y <- c(rnorm(N1, sd=0.5), rnorm(N2, sd=2))
   ```

6. Assign the point distribution to the `z` axis.

   Set a random normal distribution for `N1`, with the standard deviation of `.05`, and a random Poisson distribution for `N2` with lambda of `20` to specify the means. Subtract 20 from the concatenation to center the points correctly on the `z` axis.

   ```
   z  <- c(rnorm(N1, sd=0.5), rpois(N2, lambda=20)-20)
   ```

7. Assign to `col` the color values for `N1` and `N2`.

   Set `N1` points to be yellow (`#ffff00`) and set the `N2` points to be blue (`#0000ff`).

   ```
   col <- c(rep("#ffff00",N1),rep("#0000ff",N2))
   ```

8. Call the threejs function `scatterplot3js` to create the three-dimensional plot.

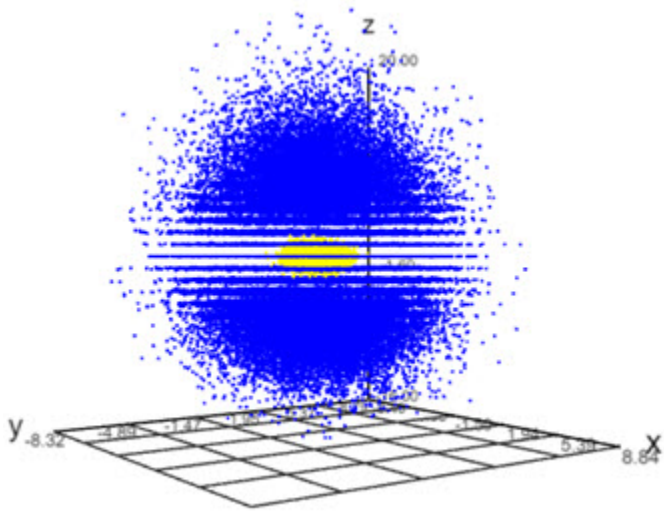
   Plot the points for the coordinate values `x`, `y`, and `z` axes in the three-dimensional graph, setting the colors to `col`, and the point size to `0.25`.

   ```
   scatterplot3js(x,y,z, color=col, size=0.25)
   ```

Result

A browser opens and shows the three-dimensional scatterplot, which you can reposition to see the point distribution.

Dragging the visualization vertically reveals the `N1` points centered in the cloud of `N2` points.



Displaying a Linear Model on a Scatterplot with ggvis

If you want to create an interactive graph that is similar to one you can create in ggplot2, you can install the ggvis package.

Perform this task from either RStudio or the TERR console. This example walks you through creating a scatter plot with a linear model prediction using data available on the internet, and then displays the results it in a browser. The ggvis package does not use JavaScript.

For information about the packages used in this example task, see http://ggvis.rstudio.com/.

If you use RStudio, the results are displayed in the RStudio Viewer pane.

Prerequisites

TERR, access to the internet, and a browser.

Procedure

1. From the TERR console or RStudio prompt, install the ggvis package.

   ```
   install.packages("ggvis")
   ```

   TERR checks the repository for the package to install, and then installs it along with any packages it requires.

2. Download the data from the internet, and assign it to `file`.

   The data in this example is in a `.csv` file on github.

   ```
   file <- "https://github.com/smach/NICAR15data/raw/master/testscores.csv"
   ```

3. Read in the `.csv` file using the utils package function `read.csv`, assigning it to the object `testdata`.

   ```
   testdata <- read.csv(file, stringsAsFactors = FALSE)
   ```

4. Load the ggvis package.

```
library("ggvis")
```

   The package and its required packages are loaded, and messages about any object masking are displayed, along with any warnings regarding TERR differences.

5. Call the ggvis package function `ggvis`, passing in the testdata object, and providing the property information and modeling for the graph.

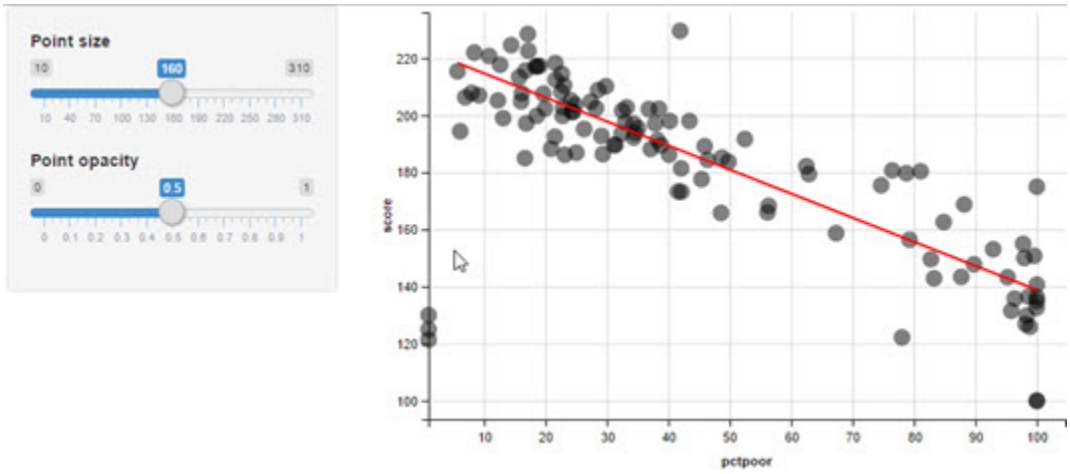
   In the example, these specify the following.

   - The x and y values to graph (`pctpoor` and `score` columns).
   - Point properties specifying size and opacity.
   - Sliders for interaction with point size and opacity, and which show default labels and scales for each point property.
   - A linear model prediction, specified as a red line.

```
ggvis(testdata, ~ pctpoor, ~ score) %>%
  layer_points(size := input_slider(10, 310, label = "Point size"),
    opacity := input_slider(0, 1, label = "Point opacity")) %>%
  layer_model_predictions(model = "lm", stroke := "red", fill := "red")
```

Result

The specified graph is opened in a browser window. You can manipulate the sliders to change point size and opacity.



Calling RGraph to Create an Image File with the TERR RinR Package

The RinR package provides functions that can start other versions of TERR or open-source R. Using the RinR function `RGraph`, you can create a graph in open-source R, and then just view the graph or save it as an image file to share with others.

Perform this task from the TERR console.

This example walks you through creating lattice graph, saved as an image (`.png`) file, using data available in the TERR library Sdatasets, and then displaying the results in a browser.

For information about RinR, load the package and then open the package help.

```
help(RinR)
```

Prerequisites

You must have open-source R and TERR installed on your computer. For this version of TERR, we tested with open-source R version 4.0.

Procedure

1. Load the RinR package in TERR.
   `library("RinR")`
2. Optional: If necessary, run the RinR function `configureREvaluator`, passing in the full path to your open-source R installation.
   This step might be required if you have installed open-source R in a non-standard location.

   ```
   configureREvaluator(REvaluator, FullPath="C:/R-4.0.0/bin/R")
   ```

   a) Run the RinR function `REvaluate(version$version.string, REvaluator)` to check the results.

   ```
   [1] "R version 4.0.0 (2020-04-24)"
   ```

3. Run the RinR function `RGraph`, passing in arguments to that create the graphic.

   - An expression, that specifies the type of graphic you want to produce. In the example, the example is an open-source R latticed histogram.
   - The data, that is specified in a character vector or in a list. In the example, the data, singer, is from TERR Sdatasets package and is specified in a list.
   - The argument `display=TRUE`, which opens the graph in your default image viewer.

   ```
   h <- RGraph(print(lattice::histogram(~height|voice.part,
   data=s)), data=list(s=Sdatasets::singer), display=TRUE)
   ```

If you want to save the resulting image as a file and share it with others, complete the following steps.

4. Optional: To save the resulting image as a file and share it with others, call the TERR base package function `tempfile` to create an object to contain the file extension.

   ```
   pngFile <- tempfile(fileext=".png")
   ```

5. Optional: Call the TERR base package function `writeBin`, passing in the two objects created in steps 3 and 4.
   This step creates the binary file containing the image and specifying the file type.

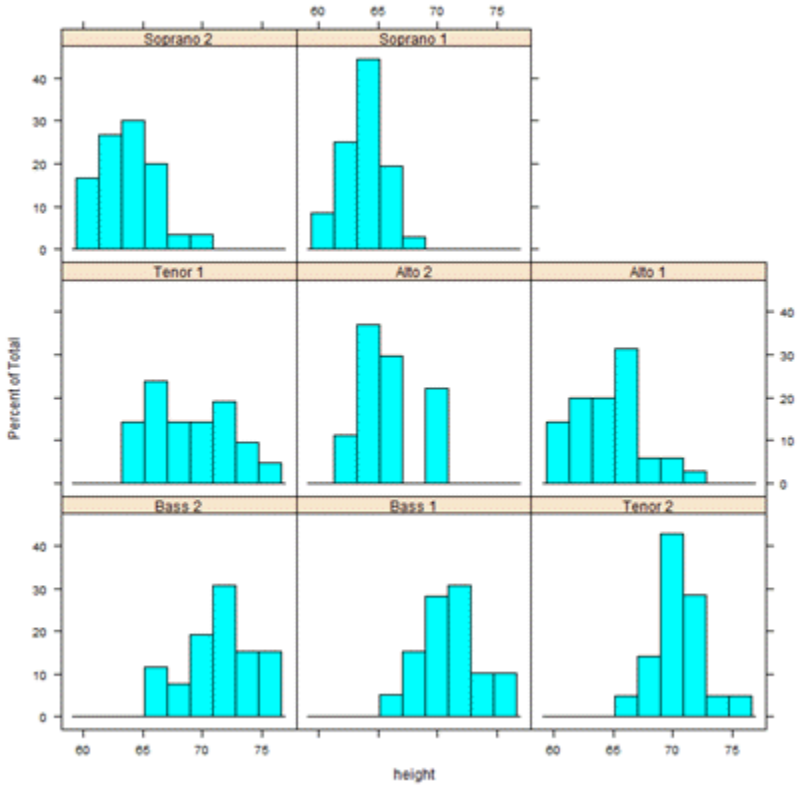
   ```
   writeBin(h, pngFile)
   ```

6. Call the TERR utils package function `browseURL`, passing in the image file.

   ```
   browseURL(pngFile)
   ```

   A default image viewer for the file type opens, displaying the image. In this example, the image shows a histogram of singer heights for each voice type in the choir.

Result



What to do next

From the TERR console, call the function `browseVignettes("RinR")` and review the additional examples for using RGraph with TERR.

Generating TERR Graphics in Spotfire Using Predictive Modeling

Spotfire supports using the TERR engine to create built-in predictive analytics, writing expression functions for ad hoc analysis, or writing more sophisticated scripts to call hand-crafted data functions. The type of prediction to undertake is determined by your data and the kind of an analysis you need to do.

Spotfire provides a range of complex statistical solutions. We address working just with the TERR engine. For more information on working with other statistical engines, see the Spotfire help.

- Access Spotfire predictive modeling from the **Tools** menu in the Spotfire user interface. This feature delivers visualizations that require no scripting. They are applicable to certain kinds of data sets, generally for common types of predictive statistical analyses. These include the following predictive models.
  - Linear regression
  - Regression tree
  - Logistic regression
  - Classification tree

  See Predictive modeling and the topic *What is Predictive Modeling?* in the Spotfire help for more information.

- Compute Holt-Winters forecasting in Spotfire from a line chart displaying time series. See the topic *Curve Fit Models* in the Spotfire help for more information.
- Access Spotfire expression functions that call the TERR engine from the **Insert** > **Calculated Column** menu or the Custom Expression dialog box. Use expression functions to call TERR scripts directly from the Spotfire expression language for ad hoc analysis. Alternatively, you can write and store an expression function for later use from the **Edit** > **Data Function Properties**, **Expression Functions** tab. See Expression functions for more information.
- You can access Spotfire data functions from several menus in Spotfire. Begin by finding it in the **Tools** > **Register Data Functions** menu in Spotfire. Using data functions, you can map input parameters to columns in your data table in Spotfire, and then use the data function output to create a data table to add to your visualization. You can develop data functions offline, edit an existing data function, or share a data function with others, and then use them in Spotfire to create visualizations. See the section Data functions for more information and example tasks. Also, in the Spotfire help, see the topic *How to Use Data Functions* for information about managing data functions in Spotfire.

Creating a Formatted HTML Document with Graphed Data with the rmarkdown Package

You can use the rmarkdown package with another Javascript-enabled package to create a formatted HTML document with graphed data.

This example uses the same data from the task described in the topic Mapping data with TERR and leaflet to create a map chart with data available on the internet. Additionally, we create a document with formatted text and an interactive map, and then display the results in a stand-alone HTML file.

- To install the required packages, call the `install.packages()` function from the TERR console.
- To run the script that creates and displays the results, call the functions from the command line.

Package link	Package short description
https://dplyr.tidyverse.org/	Work with data frame-like objects, both in memory and out of memory.
https://rstudio.github.io/leaflet/	Create and customize interactive maps using the 'Leaflet' JavaScript library and the 'htmlwidgets' package.
https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf	Convert R Markdown documents into a variety of formats.

Prerequisites

TERR, access to the internet, and a web browser.

Procedure

1. From the TERR console, install the dplyr and leaflet packages.

   ```
   install.packages(c("dplyr", "leaflet", "rmarkdown"))
   ```

   TERR checks the repository for the packages to install, and then installs them along with any packages they require.

2. Copy the code below to a text editor, and then save the file using the file name `leaflet_starbucks.Rmd`.

   ````
   ---
   title: "Starbucks in Seattle"
   author: "TIBCO Software Inc."
   date: "`rformat(Sys.time(), '%d %B %Y')`"
   output: html_document
   ---

   ```{r setup, include=FALSE}
   knitr::opts_chunk$set(echo = TRUE)
   ```

   ## Using TERR with Rmarkdown and leaflet

   We are creating an html document in TERR using the `rmarkdown` package.
   For graphics, we use the `leaflet` package to draw a map.

   For this example we use data on the location of Starbucks stores in the U.S.
   which is downloaded from the https://opendata.socrata.com website.

   This document can be processed by TERR from the command line with the command:
   ```{r eval=FALSE}
   TERR -e "rmarkdown::render('leaflet_starbucks.Rmd')"
   ```

   This will create the file `leaflet_starbucks.html` which we can view in a browser.
   ````

```
## Setup
First we load the required packages.
```{r packages, message=FALSE, warning=FALSE}
library("dplyr")
library("leaflet")
```

The Data

We restrict the data to stores in Washington state
(using `filter` from the `dplyr` package).

```{r data}
file <- "https://opendata.socrata.com/api/views/ddym-zvjk/rows.csv"
starbucks <- read.csv(file) %>%
  filter(State == "WA")
```

Starbucks in Seattle

We use the `leaflet` package functions to draw a map center in Seattle with the
Starbuck store locations identified.

```{r leaflet}
leaflet() %>%
  addTiles() %>%
  setView(-122.32, 47.605, zoom = 12) %>%
  addMarkers(data = starbucks, lat = ~ Latitude, lng = ~ Longitude,
    popup = starbucks$Name)
```
```

3. Open a command window, and navigate to the directory where you saved the file `leaflet_starbucks.Rmd`.
4. At the command prompt, type the following command.

```
TERR -e "rmarkdown::render('leaflet_starbucks.Rmd')"
```

You might have to supply the entire path to the `.Rmd` file, unless you run the command from the directory where the file is saved.

This command starts a TERR engine and calls the rmarkdown function `render`, passing in the file as the argument. This function renders the markdown text as HTML, and formats the code contained in the text, even as the code is run. (For information about the `render` function, see its help topic.)

a. The packages are loaded.
b. The `.csv` containing the data is loaded from the web site.
c. The data is filtered to just those values in Washington state.
d. The leaflet functions are called to do the following.

- Add the map tile service.
- Set the view to center and zoom on the city of Seattle.
- Add latitude and longitude markers for each entry in the filtered data.
- Add the ability to show the value in the Name column for the selected entry in a popup.

You might notice warning messages for functions that are not yet implemented in TERR. These warnings do not affect the output.

5. Open the directory where you saved the `.Rmd` file, and then locate the file that the command built: `leaflet_starbucks.html`.
6. Open the file and review the results in a browser.

Result

The resulting HTML file shows formatted text, containing a walkthrough for creating the map example.

Using TERR, the terrJava Package, and Java

The terrJava package is included in TIBCO® Enterprise Runtime for R (TERR™).

The terrJava package supports calls between Java code and TERR™ code in both directions:

- R code within TERR can call Java static methods using the `.JavaMethod` function.
- A Java application can start an embedded TERR engine and send expressions to be parsed and evaluated in TERR.
- A Java application can spawn a separate Java process with an embedded TERR engine, and control it just as if it were embedded in the same process.

Initially, we created this package for internal use only to support running TERR engines within the TIBCO Spotfire® Statistics Services server. However, others have expressed an interest to use terrJava to embed TERR in other Java applications. This document provides pointers so programmers can try this embedding using TERR and terrJava.

The terrJava package is subject to change. If the current package does not supply everything that programmers need, the TERR development group would like know, so we can improve and extend terrJava.

This document is divided into three main sections:

- Calling from TERR into Java.
- embedding TERR within Java.
- Spawning a separate Java process with an embedded TERR engine.

We suggest that a new user review these sections in order, testing that calling Java from TERR works correctly before trying the more complex task of embedding TERR within a Java process. This can be done by creating multiple Java processes with embedded TERR engines and managing communication between them. Alternatively, one can use the `TerrJavaRemote` class to spawn and control multiple processes with embedded TERR engines, as described in Setting up environment variables for a Java application to use TIBCO Enterprise Runtime for R.

To Call into Java from TIBCO Enterprise Runtime for R

terrJava and TERR work with Java in the same process.

When the terrJava package is loaded in TERR, a Java engine starts in the same process, if one is not already running. Using the `.JavaMethod` function, you can call any static Java method loaded in the Java engine, which can then run any Java code including Java UI operations (such as creating windows). You can use the `.JavaAttachClassPath` function to add new Java code at runtime, which you can then call via `.JavaMethod`. The `.JavaMethod` and `.JavaAttachClassPath` functions are both documented in the help pages for the terrJava package.

To Load the terrJava Package

Prepare your system before you load the terrJava package. After your system is prepared, you can load the package by calling `library()`.

Before loading the terrJava package, you must set the `JAVA_HOME` environment variable to the base of the Java installation. This can be read within TERR with:

```
Sys.getenv("JAVA_HOME")
```

If this is the empty string, set it to the directory where Java is installed with a call such as:

```
Sys.setenv(JAVA_HOME="/usr/lib/jvm/java-7-sun/")
```

Alternatively, you can set the `JAVA_HOME` environment variable before starting TERR.

The terrJava package is included with the TERR built-in libraries. You can load it with:

```
library(terrJava)
```

If the library loads without errors, you can test calling into Java by executing the following expressions, both of which should return the number 8:

```
.JavaMethod("java/lang/Math", "pow", "(DD)D", 2, 3)
.JavaMethod("com/tibco/terr/TerrJava", "testPow", "(DD)D", 2, 3)
```

Example

The following example demonstrates setting the `JAVA_HOME` environment variable, starting TERR, loading the terrJava package, and running the tests on Linux:

```
testlabl6406 ~% setenv JAVA_HOME /usr/lib/jvm/java-7-sun/
testlabl6406 ~% TERR
Cannot read termcap database; using dumb terminal settings.
TIBCO Software Inc. Confidential Information
Copyright (C) 2011-2016 TIBCO Software Inc. ALL RIGHTS RESERVED
TIBCO Enterprise Runtime for R version 4.2.0 for Linux 64-bit
Type 'help()' for help.
Type 'q()' to quit.
> library(terrJava)
> .JavaMethod("java/lang/Math", "pow", "(DD)D", 2, 3)
[1] 8
> .JavaMethod("com/tibco/terr/TerrJava", "testPow", "(DD)D", 2, 3)
[1] 8
```

Troubleshooting Running Java and TIBCO Enterprise Runtime for R on a Mac

The terrJava package requires the Java Native Interface (JNI) capability, but this capability is not enabled by default in Oracle's JDK installation.

As of TERR 6.0, support for TERR on the Mac is deprecated and is no longer tested.

When you call `.JavaMethod` (or another terrJava function) on the Mac, if you receive the following error message, you must add the missing capability to the JDK `Info.plist`.

```
Java installation at /Library/Java/JavaVirtualMachines/jdk1.8.0_73.jdk/Contents/Home does not
 have required 'JNI' capability
```

Adding this missing capability to the Java configuration allows the JVM to sart as expected.

Procedure

- Type the following one-line command to add the JNI capability to the `Info.plist`.

  ```
  sudo /usr/libexec/PlistBuddy -c "Add :JavaVM:JVMCapabilities: string JNI" $JAVA_HOME/../
  Info.plist
  ```

Other Useful Environment Variables

You can set other environment variables to affect the process of loading terrJava. After terrJava has been loaded, setting these variables does not affect it.

Environment Variable	Description
CLASSPATH	If this environment variable is set, it should contain one or more file names for Java .jar files or Java class directories. If there is more than one, they should be concatenated into a single string separated by ; (on Windows) or : (on Linux). These classes are added to Java's classpath when the Java engine is started. Setting this environment variable might be more convenient than calling .JavaAttachClassPath if you know the classes you want to load before starting Java.
JAVA_VERBOSE	If this environment variable is set to TRUE, status information is printed while the terrJava package is being loaded. This might print out useful information if errors occur when loading this package. Setting this environment variable to TRUE also prints the arguments passed to the Java engine on startup (which can be affected by the following environment variable).
JAVA_OPTIONS	If this environment variable is specified, it should contain a string of options to use when starting the Java engine. For example, the value -Xmx1000m specifies a maximum Java memory size of 1GB. One use for JAVA_OPTIONS is to start the Java engine so that it can be debugged remotely. For example, suppose you set JAVA_OPTIONS to -Xdebug -Xrunjdwp:transport=dt_socket,address=4444,server=y,suspend=n When the Java engine is started, it creates a server listening for connections from a debugger on port 4444. A Java debugger (like Eclipse) could connect remotely to the Java engine and set breakpoints in the Java code.

To Embed the TIBCO Enterprise Runtime for R Engine within a Java Application

The terrJava package supports starting a TERR engine from Java, and then sending expressions for TERR to evaluate.

Setting this up is more complicated than starting Java from TERR by executing `library(terrJava)`, but it has proven useful for Java applications that need to call TERR computations.

The TERR engine is single-threaded, so it is not possible to run more than one TERR engine at the same time within a given operating system process. If your Java application wants to create multiple TERR engines, the application must create the engines in multiple operating system processes. This can be done by creating multiple Java processes with embedded TERR engines, and managing communication between them. Alternatively, you can use the `TerrJavaRemote` class to spawn and control multiple processes with embedded TERR engines, as described in Setting up environment variables for a Java application to use TIBCO Enterprise Runtime for R.

Setting Up Environment Variables for a Java Application to Use TIBCO Enterprise Runtime for R

For a Java application to use the embedded TERR engine, you must prepare the environment.

Procedure

1. Set the `JAVA_HOME` environment variable to the base of the Java installation.

   This step is necessary because the embedded TERR engine automatically loads the terrJava package, which requires this environment variable.
2. Set the `TERR_HOME` environment variable to the base of the TERR installation (that is, the directory containing subdirectories `bin`, `library`, and so on).

3. Set the Java classpath to include the file *TERR_HOME*/library/java/terrJava.jar, which contains the Java code for starting and controlling an embedded TERR engine.

   The Java classpath can be set by setting the CLASSPATH environment variable, or giving the -classpath argument when starting Java.

4. Set the appropriate environment variable so Java can access the TERR engine libraries.

   - On Windows, you must add TERR_HOME/bin/x64 (if you are using 64-bit TERR and Java) or TERR_HOME/bin/i386 (if you are using 32-bit TERR and Java) to the PATH environment variable.
   - On Linux, you should add *TERR_HOME*/lib/x86_64-unknown-linux/ to the LD_LIBRARY_PATH environment variable.

What to do next

To test that these environment files are set correctly, the class com.tibco.terr.TerrJava (in terrJava.jar) contains a main method that implements a simple TERR console. See Example for setting Windows environment variables or Example for setting Linux environment variables.

Example for Setting Linux Environment Variables

This code example demonstrates setting environment variables on Linux.®

This example would be run on Linux in the tcsh shell.

Example

```
seaqal6406 ~% setenv JAVA_HOME /usr/lib/jvm/java-11-sun/
seaqal6406 ~% setenv TERR _HOME /opt/sw/snext/daily/intel
seaqal6406 ~% setenv CLASSPATH $TERR_HOME/library/terrJava/java/terrJava.jar
seaqal6406 ~% setenv LD_LIBRARY_PATH $TERR_HOME/lib/x86_64-unknown-
linux:$LD_LIBRARY_PATH
seaqal6406 ~% $JAVA_HOME/bin/java com.tibco.terr.TerrJava
**** starting engine ****
**** engine started ****
**** starting console loop ****
> 1:10
[1]  1  2  3  4  5  6  7  8  9 10
> .JavaMethod("java/lang/Math", "pow", "(DD)D", 2, 3)
[1] 8
> .JavaMethod("com/tibco/terr/TerrJava", "testPow", "(DD)D", 2, 3)
[1] 8
> q()
**** finished console loop ****
seaqal6406 ~%
```

Example for Setting Windows Environment Variables

This code example demonstrates setting environment variables on Microsoft Windows®.

This example would be run in a Windows command prompt window, using 64-bit TERR and Java.

Example

```
D:\> set JAVA_HOME=C:Program Files\Java\jdk-11.0.1
D:\> set TERR_HOME=D:\TERR
D:\> set CLASSPATH=%TERR_HOME%\library\terrJava\java\terrJava.jar
D:\> set PATH=%TERR_HOME%\bin\x64;%PATH%
D:\> %JAVA_HOME%\bin\java com.tibco.terr.TerrJava
**** starting engine ****
**** engine started ****
**** starting console loop ****
> 1:10
```

```
[1]  1  2  3  4  5  6  7  8  9 10
> .JavaMethod("java/lang/Math", "pow", "(DD)D", 2, 3)
[1] 8
> .JavaMethod("com/tibco/terr/TerrJava", "testPow", "(DD)D", 2, 3)
[1] 8
> q()
**** finished console loop ****
D:\>
```

Java API for Using an Embedded TIBCO Enterprise Runtime for R Engine

The class `com.tibco.terr.TerrJava` contains a set of static methods that a Java application can call to start an embedded TERR engine, send expressions to be evaluated, add hooks for text input and output, and interrupt a running computation.

You can find the class `com.tibco.terr.TerrJava` in `<TERR_HOME>/library/terrJava/java/terrJava.jar`. When you compile Java application code using these APIs, you must have the file `terrJava.jar` on the Java class path.

The reference for the static methods is in the javadoc documentation, available in the file `<TERR_HOME>/library/terrJava/doc/javadoc/index.html`.

To Pass Data Between Java and TIBCO Enterprise Runtime for R

TerrJava methods have limitations for evaluating expressions.

One such limitation is that the methods do not support sending data between Java and TERR, except for a single string value returned from `evaluateToString`. Simple values can be included in the expression string sent to TERR, and the string value returned from `evaluateToString` can include some data, but this is not reasonable for transferring large amounts of data.

The solution is to use methods for converting between TERR data objects and Java TerrData objects:

```
   public static void setVariable(String name, TerrData value)
   public static TerrData getVariable(String name)
   public static TerrData getVariable(String name, TerrData reuse)
```

- The first method converts a TerrData object into a TERR data object, and assigns it to the specified global variable.
- The second method retrieves the value of a global variable, and converts it to a TerrData object.
- The third method retrieves a value, but also allows reusing the storage of an existing TerrData object, reducing Java object allocation.

`TerrData` is the superclass of several specific classes for representing particular types of TERR data objects, including `TerrDouble`, `TerrString`, `TerrFactor`, `TerrList`, and `TerrDataFrame`. Here is some example code showing how to start an embedded engine, construct a `TerrDataFrame` object in Java, send it to TERR, evaluate a linear model on the data and retrieve the coefficients of the model as another `TerrData` object.

```
TerrJava.startEngine();
TerrDataFrame df = new TerrDataFrame(new String[] { "x", "y" },
   new TerrData[] {
      new TerrDouble(new double[] { 1,2,3,4,5,6 }),
      new TerrDouble(new double[] { 1,4,9,16,25,36 })
   });
TerrJava.setVariable("df", df);
TerrJava.evaluateInteractive("df.mod <- lm(y~x, data=df)");
TerrJava.evaluateInteractive("df.coef <- df.mod$coefficients");
TerrData coef = TerrJava.getVariable("df.coef");
for (int i=0; i<coef.getLength(); i++) {
   System.out.println(coef.names[i]+" : "+((TerrDouble)coef).data[i]);
}
// this prints:
//   (Intercept) : -9.333333333333345
//   x : 7.000000000000002
```

Before the development of the `TerrData` object and these methods for reading and writing variables, the best way to transfer data between Java and TERR was to use the `.JavaMethod` function to call static Java methods that read or write values from static Java fields. However, this only supported reading and writing simple vectors of doubles, strings, and so on. Large and complex objects such as `data.frames` could be transferred by breaking them down into simple vectors that can be transferred via `.JavaMethod`.

This works, but the code is complex and unreliable and slow.

In one test, transferring a data frame with 40 columns of doubles as a `TerrDataFrame` was 100 times as fast as transferring the columns individually via `.JavaMethod`.

Implement a Console Using the TerrJava API

We have provided the example Java code in both Windows and Linux for a Java application that implements a simple TERR console using the TerrJava API. This is similar to the console application in `TerrJava.main`, except that it accesses the TerrJava methods from another class.

An interesting feature of this code is that the main `read-eval` loop detects when the current line cannot be evaluated because it is an incomplete expression or string, and it accumulates multiple lines until it does evaluate (or another type of error occurs). It also calls into TERR with `TerrJava.evaluateToString` to retrieve the appropriate prompt string for normal input, incomplete expression, or incomplete string, which can be changed in TERR at any time.

A Console Application

This is the Java code for a simple console application using the TerrJava API.

```
import java.io.BufferedReader;
import java.io.InputStreamReader;

import com.tibco.terr.TerrJava;

public class TerrConsoleExample {

  public static void main(String[] args) throws Throwable {
    System.out.println("TerrConsoleExample:");

    // set up output/input handlers
    TerrJava.OutputHandler out = new TerrJava.OutputHandler() {
      public void write(String data, boolean prompted) {
        System.out.print(data);
      }
    };
    TerrJava.InputHandler in = new TerrJava.InputHandler() {
      public String readLine() {
        try {
          BufferedReader bin = new BufferedReader(
              new InputStreamReader(System.in));
          String cmd = bin.readLine();
          return cmd;
        } catch (Exception ex) {
          return "";
        }
      }
    };
    TerrJava.setOutputHandler(out);
    TerrJava.setInputHandler(in);

    // start TIBCO Enterprise Runtime for R engine
    TerrJava.startEngine();
    // repeatedly read input lines and evaluate them
    String resultCode = "";
    String prompt = "";
    StringBuffer expression = new StringBuffer();

    while (true) {
      // get prompt string from TIBCO Enterprise Runtime for R
      if (resultCode.equals("IncompleteString")) {
        prompt = TerrJava.evaluateToString("getOption('continueString')");
```

```
      } else if (resultCode.equals("IncompleteExpression")) {
        prompt = TerrJava.evaluateToString("getOption('continue')");
      } else {
        prompt = TerrJava.evaluateToString("getOption('prompt')");
        expression.setLength(0);
      }

      out.write(prompt, true);
      String line = in.readLine();
      // ignore empty input lines, except when we are reading a multi-line string
      if (line.isEmpty() && !resultCode.equals("IncompleteString")) {
        continue;
      }

      // accumulate multi-line expression, until we can parse it
      if (expression.length() > 0) {
        expression.append("\n");
      }
      expression.append(line);

      resultCode = TerrJava.evaluateInteractive(expression.toString());

      // exit if engine just evaluated q()
      if (resultCode.equals("Quit")) {
        resultCode = "Success";
        break;
      }

      // print error message if any
      if (resultCode.equals("EvaluationError") ||
        resultCode.equals("ParserError")
        ) {
        String error = TerrJava.getLastErrorMessage();
        if (!error.isEmpty()) {
          out.write(error + "\n", false);
        }
      }
    }
  }
}
```

Run the Console Application

After you create the console application, you can test it using example user input and text output on Linux.

```
seaqal6406 ~% setenv JAVA_HOME /usr/lib/jvm/java-6-sun/
seaqal6406 ~% setenv TERR_HOME /opt/sw/snext/daily/intel
seaqal6406 ~% setenv CLASSPATH
  $TERR_HOME/library/terrJava/java/terrJava.jar:/homes/sannella/TerrConsoleExample
seaqal6406 ~% setenv LD_LIBRARY_PATH
  $TERR_HOME/lib/x86_64-unknown-linux:$LD_LIBRARY_PATH
seaqal6406 ~% $JAVA_HOME/bin/java TerrConsoleExample
TerrConsoleExample:
> # simple comments do nothing
> 1:10
 [1]  1  2  3  4  5  6  7  8  9 10
>
> # terrJava is automatically loaded,
> # so you can call .JavaMethod
> .JavaMethod("java/lang/Math", "pow", "(DD)D", 2, 3)
[1] 8
>
> # call readline, which calls input handler
> # to read a line
> x <- readline()
typing in a new line
> x
[1] "typing in a new line"
>
> # change prompt string, then change it back
> options(prompt="newprompt--> ")
newprompt--> 1+2
[1] 3
newprompt--> options(prompt="> ")
>
> # handle incomplete expression
```

```
> # by accumulating multiple lines
> 1+(2
+ +3)
[1] 6
>
> # handle incomplete string
> # by accumulating multiple lines
> nchar("abcd
Continue string: efg")
[1] 8
>
> # error is printed out
> stop("foo")
Error: foo
> # traceback() gives stack from last error
> traceback()
1: stop("foo")
>
> # quit from engine
> q()
seaqal6406 ~%
```

To Spawn TIBCO Enterprise Runtime for R Engines in a Separate Process

The terrJava package has been extended with the Java class `com.tibco.terr.TerrJavaRemote`, which can be used to spawn a TERR engine running in a separate process, send commands to this engine, and get or set data objects to this engine.

The `com.tibco.terr.TerrJavaRemote` class contains methods similar in name and behavior to the methods of the `TerrJava` class that creates an embedded TERR engine within the Java process. One important difference is that the `TerrJava` methods are static class methods instead of object methods, because they control the (single) TERR engine within the process. In contrast, it is possible to create multiple `TerrJavaRemote` objects, each connected to and controlling a separate spawned TERR engine process.

Following are some reasons to use `TerrJavaRemote` objects rather than `TerrJava`.

- You can access more than one TERR engine at a time.
- You can use TERR from a Java application, where it is inconvenient or impossible to set the appropriate environment variables needed to run TERR within the Java process.
- You can spawn TERR within a Java JVM separate from the one containing the `TerrJavaRemote` object. For example, you can spawn a 64-bit Java and TERR process from a 32-bit Java application.

Like the `TerrJava` class, the `TerrJavaRemote` class is defined in *`TERR_HOME`*`/library/terrJava/java/terrJava.jar`, and documentation is available in *`TERR_HOME`*`/library/terrJava/doc/javadoc/index.html`. `TerrJavaRemote` includes methods for sending expressions to be evaluated in TERR, adding hooks for capturing TERR text input and output, and interrupting a running computation. There are also methods for transferring `TerrData` objects going to or coming from the TERR engine.

`TerrJavaRemote` methods are thread-safe, so a Java application could create and access multiple `TerrJavaRemote` objects in separate Java threads. These methods (except for `interrupt`) use Java synchronization so only one thread can manipulate a given engine at a time. Separate `TerrJavaRemote` objects can access separate TERR engines at once without any interaction between these engines.

A Console Application Using TerrJavaRemote

This is sample Java code for a simple console application using the `TerrJavaRemote` API.

This is very similar to the code in A console application, except that it uses `TerrJavaRemote` methods to create and control a spawned process with a TERR engine, rather than creating an embedded engine via `TerrJava`. Another difference is that the TERR and Java home paths are passed in as arguments, rather than being accessed via environment variables.

```
import java.io.BufferedReader;
```

```
import java.io.InputStreamReader;
import com.tibco.terr.TerrJava;
import com.tibco.terr.TerrJavaRemote;
public class TerrJavaRemoteConsoleExample {
    public static void main(String[] args) throws Throwable {
        System.out.println("TerrJavaRemoteConsoleExample:");
        // create TerrJavaRemote object
        TerrJavaRemote engine = new TerrJavaRemote();

        // set paths for spawned engine
        if (args.length<2) {
            throw new Exception("need two arguments: TERR home and Java home");
        }
        engine.setTerrHome(args[0]);
        engine.setJavaHome(args[1]);

        // set up output/input handlers
        TerrJava.OutputHandler out = new TerrJava.OutputHandler() {
                public void write(String data, boolean prompted) {
                    System.out.print(data);
                }
            };
        TerrJava.InputHandler in = new TerrJava.InputHandler() {
                public String readLine() {
                    try {
                        BufferedReader bin = new BufferedReader(
                                new InputStreamReader(System.in));
                        String cmd = bin.readLine();
                        return cmd;
                    } catch (Exception ex) {
                        return "";
                    }
                }
            };
        engine.setOutputHandler(out);
        engine.setInputHandler(in);
        // start TIBCO Enterprise Runtime for R engine
        engine.startEngine();
        // repeatedly read input lines and evaluate them
        String resultCode = "";
        String prompt = "";
        StringBuffer expression = new StringBuffer();
        while (true) {
            // get prompt string from TIBCO Enterprise Runtime for R
            if (resultCode.equals("IncompleteString")) {
                prompt = engine.evaluateToString("getOption('continueString')");
            } else if (resultCode.equals("IncompleteExpression")) {
                prompt = engine.evaluateToString("getOption('continue')");
            } else {
                prompt = engine.evaluateToString("getOption('prompt')");
                expression.setLength(0);
            }
            out.write(prompt, true);
            String line = in.readLine();
            // ignore empty input lines, except when we are reading a multi-line string
            if (line.isEmpty() && !resultCode.equals("IncompleteString")) {
                continue;
            }
            // accumulate multi-line expression, until we can parse it
            if (expression.length() > 0) {
                expression.append("\n");
            }
            expression.append(line);
            resultCode = engine.evaluateInteractive(expression.toString());
            // exit if engine just evaluated q()
            if (resultCode.equals("Quit")) {
                resultCode = "Success";
                break;
            }
            // print error message if any
            if (resultCode.equals("EvaluationError") ||
                resultCode.equals("ParserError")) {
                String error = engine.getLastErrorMessage();
                if (!error.isEmpty()) {
                    out.write(error + "\n", false);
                }
            }
        }
    }
```

```
}
```

Run the Console Application Using TerrJavaRemote

This topic shows example user input and text output when running the TerrJavaRemoteConsoleExample application on Linux.

This example includes the same input as the example from Run the console application, demonstrating that it is possible to do everything in the spawned TERR engine that was possible using the embedded engine, including calling readline to read a line from the console (which is returned to the spawned TERR process).

```
seaqal6406 TerrJavaRemoteConsoleExample% /opt2/users/sannella/jdk1.7.0_40/bin/
java -classpath .:/opt2/users/sannella/TERR.rev17575.lnx64.intel.release/library/
terrJava/java/terrJava.jar TerrJavaRemoteConsoleExample /opt2/users/sannella/
TERR.rev17575.lnx64.intel.release /opt2/users/sannella/jdk1.7.0_40
TerrJavaRemoteConsoleExample:
started engine node pid==24125 at Thu Jun 12 16:08:00 2014
> # simple comments do nothing
> 1:10
 [1]  1  2  3  4  5  6  7  8  9 10
>
> # terrJava is automatically loaded,
> # so you can call .JavaMethod
> .JavaMethod("java/lang/Math", "pow", "(DD)D", 2, 3)
[1] 8
>
> # call readline, which calls input handler
> # to read a line
> x <- readline()
typing in a new line
> x
[1] "typing in a new line"
>
> # change prompt string, then change it back
> options(prompt="newprompt--> ")
newprompt--> 1+2
[1] 3
newprompt--> options(prompt="> ")
>
> # handle incomplete expression
> # by accumulating multiple lines
> 1+(2
+ +3)
[1] 6
>
> # handle incomplete string
> # by accumulating multiple lines
> nchar("abcd
Continue string: efg")
[1] 8
>
> # error is printed out
> stop("foo")
Error: foo
> # traceback() gives stack from last error
> traceback()
1: stop("foo")
>
> # quit from engine
> q()
seaqal6406 TerrJavaRemoteConsoleExample%
```

Signal Handlers

TERR supports installing signal handlers to catch illegal operations that occur in foreign code called via the `.C()` or `.Call()` functions.

A signal handler can catch such illegal operations as referencing an illegal memory location. If an illegal operation is caught by a signal handler, it generates an error "Unhandled exception in foreign function"

rather than crashing the process. Currently, these signal handlers are disabled by default, because we found that they could possibly interfere with the signal handlers used by Java.

Sometimes when investigating unexpected failures, you might want to enable the signal handlers. You can enable signal handlers when starting the TIBCO Enterprise Runtime for R console application by specifying the option:

```
--enable-signal-handlers
```

Alternatively, you can enable them by starting the engine from Java with the following engine parameter:

```
TerrJava.startEngine("FFInterface.SignalHandlersEnabled=TRUE")
```

If you want to enable signal handling when using Java on Linux, it might be helpful to set the environment variable `LD_PRELOAD` so the signal handlers set up by TERR are "chained" after the Java signal handlers, as described in the URL http://www.oracle.com/technetwork/java/javase/signals-139944.html#gbzcz.

According to this URL, setting the environment variable `LD_PRELOAD` to `<libjvm.so-directory-in-java-tree>/libjsig.so` causes Java to link in the special libjsig.so library, which handles both Java signal handlers and native code with its own handlers. We have found several cases where this workaround solves the problem of embedding TERR within a complex Java application.

To Call Embedded TIBCO Enterprise Runtime for R from an IntelliJ Project

Using the IntelliJ IDE to debug your Java application could require a workaround.

In one case, a user had problems using embedded TERR when debugging a Java application within the IntelliJ IDE, even when the application worked when invoked from Java directly. Specifically, evaluating the following:

```
.JavaMethod("java/lang/Math", "pow", "(DD)D", 2, 3)
```

gave the correct answer, 8, whereas evaluating the following:

```
.JavaMethod("com/tibco/terr/TerrJava", "testPow", "(DD)D", 2, 3)
```

resulted in the following Java error:

```
java.lang.ClassNotFoundException: com.tibco.terr.TerrJava
```

Eventually, we discovered that IntelliJ uses a separate Java class loader when a project has a large classpath. This prevents `.JavaMethod` from finding the class `com/tibco/terr/TerrJava`. There is no clear way to work around this problem in TERR, because TERR cannot access IntelliJ's special class loader.

This problem is described in the article found at http://youtrack.jetbrains.com/issue/IDEA-48090?query=8, which also claims that this IntelliJ bug has been fixed.

In any case, this page also describes a workaround when using IntelliJ: remove the following line from the file `workspace.xml`:

```
<property name="dynamic.classpath" value="true"/>
```

Available Functions in TIBCO Enterprise Runtime for R

TERR strives to be entirely compatible with open-source R, and so it contains thousands of functions that you can use to run R code.

The lists of available functions provides you with a quick view into those functions that have been implemented as of version 6.1.0, and each release we add more to increase compatibility and enhance the performance and usability of TERR.

This table provides information about the numbers of functions implemented in open-source R that are also available in TIBCO Enterprise Runtime for R.

Package	Open-source R functions	TERR functions	Percent of open-source R functions implemented in TERR (%)
base	1359	1201	88.4
datasets	104	104	100
graphics*	88	86	97.7
grDevices*	113	104	92
methods	371	103	27.8
stats	456	394	86.4
utils	219	140	63.9

Basics

These basic functions are available in TERR. For help with a function, see its listing in the Language Reference.

Basic System Variables

These are the available TERR basic system variables. See each function's help topic in the TERR Language Reference for more information.

Variable name	Title description
`.Machine`	Machine Arithmetic Constants
`.Platform`	Platform Specific Variables
`.Random.seed`	Control Random Number Generator
`as.null`	The Null Object
`as.null.default`	The Null Object
`is.null`	The Null Object
`NULL`	The Null Object

Variable name	Title description
R.version	Version Information
R.Version	Version Information
R.version.string	Version Information
RNG	Control Random Number Generator
RNGkind	Control Random Number Generator
RNGversion	Control Random Number Generator
set.seed	Control Random Number Generator
version	Version Information

Categorical Data

These are the available functions for manipulating categorical data. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
.bincode	Create Factor Object from Numeric Vector
aggregate	Compute Summary Statistics of Subsets of Data
aggregate.data.frame	Compute Column-by-Column Summaries of Groups of Observations
aggregate.default	Compute Summary Statistics of Subsets of Data
aggregate.formula	Compute Summary Statistics of Subsets of Data
aggregate.ts	Compute Summary Statistics of Subsets of Data
as.data.frame.ftable	Flat Contingency Tables
as.factor	Create Factor Object
as.ordered	Create an Ordered factor Object
as.table.ftable	Flat Contingency Tables
asplit	Split Array into List of Subarrays
by	Split a Data Frame and Apply a Function to the Parts
by.data.frame	Split a Data Frame and Apply a Function to the Parts
by.default	Split a Data Frame and Apply a Function to the Parts
cut	Create Factor Object from Numeric Vector
cut.default	Create Factor Object from Numeric Vector

Function name	Title description
factor	Create Factor Object
format.ftable	Flat Contingency Tables
ftable	Flat Contingency Tables
ftable.default	Flat Contingency Tables
ftable.formula	Flat Contingency Tables
gl	Generate Patterned Factor
is.factor	Create Factor Object
is.ordered	Create an Ordered factor Object
Math.factor	Math Group Method for Factor Objects
nlevels	Number of Levels of a factor Object
Ops.data.frame	Ops Group Method for Data Frame Objects
Ops.factor	Operations for Factors and Ordered Factors
Ops.ordered	Operations for Factors and Ordered Factors
ordered	Create an Ordered factor Object
print.ftable	Flat Contingency Tables
print.summary.table	Summary of a table Object
rapply	Apply a Function Recursively
rowsum	Group Row Sums of a Matrix
rowsum.default	Group Row Sums of a Matrix
split	Split Data by Groups
split.data.frame	Split Data by Groups
split.Date	Split Data by Groups
split.default	Split Data by Groups
split.POSIXct	Split Data by Groups
split<-	Split Data by Groups
split<-.data.frame	Split Data by Groups
split<-.default	Split Data by Groups
summary.table	Summary of a table Object

Function name	Title description
tabulate	Count Entries in Bins
tapply	Apply a Function to a Ragged Array
unsplit	Split Data by Groups
valid.factor	Create Factor Object
xtabs	Cross Tabulation

Character Data ("String") Operations

These are the available functions for string operations.See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
agrep	Approximate String Matching (Fuzzy Matching)
agrepl	Approximate String Matching (Fuzzy Matching)
as.character	Character Objects
as.character.factor	Character Objects
basename	Manipulate File Paths
casefold	Convert Case of Character Strings
char.expand	Expand a String with Respect to a Target Table
character	Character Objects
charmatch	Partial Matching of Character Strings
dirname	Manipulate File Paths
Encoding	String Encodings of a Character Vector
Encoding<-	String Encodings of a Character Vector
endsWith	Match Patterns with Strings
format	Formatted Character Data
formatC	Formatting Using C-style Formats
format.data.frame	Formatted Character Data
format.default	Formatted Character Data
format.factor	Formatted Character Data
gettext	Translate Text Messages

Function name	Title description
gettextf	Generate C-Style Formatted Output
gregexpr	Match Patterns in Strings
grep	Search for a Pattern in Text
grepl	Search for a Pattern in Text
grepRaw	Search for Pattern in Text
gsub	Replace Part of a Character String
iconv	Convert Character Vector between Encodings
iconvlist	Convert Character Vector between Encodings
inverse.rle	Run Length Encoding and Decoding
is.character	Character Objects
make.unique	Make Character Strings Unique
match	Match Items against a Table
nchar	Lengths of Character Strings
ngettext	Translate Text Messages
nzchar	Lengths of Character Strings
paste	Concatenate Data to Make Character Data
pmatch	Partial Matching of Character Items in a Vector
prettyNum	Formatting Using C-style Formats
print.rle	Run Length Encoding and Decoding
regexec	Search for a Pattern in Text
regexpr	Match Patterns in Strings
regmatches	Extract or Replace Matched Substrings
regmatches<-	Extract or Replace Matched Substrings
rle	Run Length Encoding and Decoding
sort	Sort into Numeric or Alphabetic Order
sort.default	Sort into Numeric or Alphabetic Order
sort.int	Sort into Numeric or Alphabetic Order
sort.POSIXlt	Sort into Numeric or Alphabetic Order

Function name	Title description
sprintf	Generate C-Style Formatted Output
startsWith	Match Patterns with Strings
strrep	Replace Strings with Repeated Versions
strsplit	Split the Elements of a Character Vector
sub	Replace Part of a Character String
substr	Extract or Replace Portions of Character Strings
substr<-	Extract or Replace Portions of Character Strings
substring	Extract or Replace Portions of Character Strings
substring<-	Extract or Replace Portions of Character Strings
symnum	Symbolic Number Coding
tempdir	Create Unique Names for Files
tempfile	Create Unique Names for Files
tolower	Convert Case of Character Strings
toTitleCase	Convert Case of Character Strings
toupper	Convert Case of Character Strings
trimws	Remove leading or trailing white space

Complex Numbers

These are the available functions for complex numbers. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
%/%	Arithmetic Operators
%%	Arithmetic Operators
^	Arithmetic Operators
+	Arithmetic Operators
acos	Inverse Trigonometric Functions
acosh	Inverse Hyperbolic Trigonometric Functions
Arithmetic	Arithmetic Operators
as.complex	Complex Valued Objects

Function name	Title description
asin	Inverse Trigonometric Functions
asinh	Inverse Hyperbolic Trigonometric Functions
atan	Inverse Trigonometric Functions
atan2	Inverse Trigonometric Functions
atanh	Inverse Hyperbolic Trigonometric Functions
complex	Complex Valued Objects
cos	Trigonometric Functions
cosh	Hyperbolic Trigonometric Functions
cospi	Trigonometric Functions
digamma	Gamma Function (and Its Derivatives and Logarithm)
exp	Exponential and related Functions
expm1	Exponential and related Functions
fft	Fast Fourier Transform
gamma	Gamma Function (and Its Derivatives and Logarithm)
is.complex	Complex Valued Objects
lgamma	Gamma Function (and Its Derivatives and Logarithm)
log	Exponential and related Functions
log10	Exponential and related Functions
log1p	Exponential and related Functions
log2	Exponential and related Functions
logb	Exponential and related Functions
mvfft	Fast Fourier Transform
polyroot	Find the Roots of a Polynomial
psigamma	Gamma Function (and Its Derivatives and Logarithm)
sin	Trigonometric Functions
sinh	Hyperbolic Trigonometric Functions
sinpi	Trigonometric Functions
sqrt	Exponential and related Functions

Function name	Title description
tan	Trigonometric Functions
tanh	Hyperbolic Trigonometric Functions
tanpi	Trigonometric Functions
trig	Trigonometric Functions
trigamma	Gamma Function (and Its Derivatives and Logarithm)

Data Attributes

These are the available functions for data attributes. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
.col	Column and Row Identification in a Matrix
.row	Column and Row Identification in a Matrix
as.name	Names and Symbols
as.null	The Null Object
as.null.default	The Null Object
as.symbol	Names and Symbols
attr	Attribute of an Object
attributes	All Attributes of an Object
col	Column and Row Identification in a Matrix
dim	Dim Attribute of an Object
dim<-	Dim Attribute of an Object
dimnames	Dimnames Attribute of an Object
dimnames.data.frame	Dimnames Attribute of an Object
is.name	Names and Symbols
is.null	The Null Object
is.symbol	Names and Symbols
length	Length of a Vector or List
length.POSIXlt	Length of a Vector or List
levels	Get Or Set Levels Attribute

Function name	Title description
mode	Data Mode of the Values in a Vector
name	Names and Symbols
names	Names Attribute of an Object
names.POSIXlt	Names Attribute of an Object
names<-	Names Attribute of an Object
names<-.POSIXlt	Names Attribute of an Object
nlevels	Number of Levels of a factor Object
NULL	The Null Object
row	Column and Row Identification in a Matrix
storage.mode	Data Mode of the Values in a Vector
structure	An Object with Given Attributes
type	The Type of an Object
typeof	The Type of an Object

Data Manipulation

These are the available functions for data manipulation. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
->	Assign a Name to an Object
:	Generate a Sequence
@	Extract Slot from an S4 Object
<-	Assign a Name to an Object
<<-	Assign a Name to an Object
=	Assign a Name to an Object
.mapply	Apply a Function to Multiple List or Vector Arguments
.rowSums	Row and Column Summaries
.rowMeans	Row and Column Summaries
.colSums	Row and Column Summaries
.colMeans	Row and Column Summaries

Function name	Title description
abbreviate	Generate Abbreviations
addmargins	Puts Arbitrary Margins on Multidimensional Tables or Arrays
anyDuplicated	Determine Duplicate Elements
anyDuplicated.array	Determine Duplicate Elements
anyDuplicated.data.frame	Determine Duplicate Elements
anyDuplicated.default	Determine Duplicate Elements
anyDuplicated.matrix	Determine Duplicate Elements
append	Insert or Merge Data
arrayInd	Find TRUE Values
as.null	The Null Object
as.null.default	The Null Object
Assignment	Assign a Name to an Object
asplit	Split Array into List of Subarrays
c	Combine Values into a Vector or List
c.Date	Combine Values into a Vector or List
c.POSIXct	Combine Values into a Vector or List
c.POSIXlt	Combine Values into a Vector or List
casefold	Convert Case of Character Strings
cbind	Building a Matrix from Columns or Rows
charmatch	Partial Matching of Character Strings
colMeans	Row and Column Summaries
colnames	Column and Row Names
colnames<-	Column and Row Names
colSums	Row and Column Summaries
cut.Date	Create Factor by Cutting Date or POSIXt Object
cut.POSIXt	Create Factor by Cutting Date or POSIXt Object
duplicated	Determine Duplicate Elements
duplicated.array	Determine Duplicate Elements

Function name	Title description
duplicated.data.frame	Determine Duplicate Elements
duplicated.default	Determine Duplicate Elements
duplicated.matrix	Determine Duplicate Elements
duplicated.POSIXlt	Determine Duplicate Elements
gregexpr	Match Patterns in Strings
grep	Search for a Pattern in Text
grepl	Search for a Pattern in Text
grepRaw	Search for a Pattern in Text
grouping	Grouping Permutation
gsub	Replace Part of a Character String
head	Get the First or Last Part of an Object
ifelse	Conditional Data Selection
inverse.rle	Run Length Encoding and Decoding
is.na	Not Available / Missing Values
is.na.data.frame	Not Available / Missing Values
is.na.POSIXlt	Not Available / Missing Values
is.na<-	Not Available / Missing Values
is.na<-.default	Not Available / Missing Values
is.na<-.factor	Not Available / Missing Values
is.null	The Null Object
labels.dendrogram	ID Numbers or Labels of the Leaves in a Dendrogram
length	Length of a Vector or List
length.POSIXlt	Length of a Vector or List
list2DF	Quickly Make Data.frame From List of Vectors
mapply	Apply a Function to Multiple List or Vector Arguments
match	Match Items against a Table
merge	Merge Two Datasets and Match Columns
merge.data.frame	Merge Two Datasets and Match Columns

Function name	Title description
merge.default	Merge Two Datasets and Match Columns
mostattributes<-	All Attributes of an Object
NA	Not Available / Missing Values
NA_character_	Not Available / Missing Values
NA_complex_	Not Available / Missing Values
NA_integer_	Not Available / Missing Values
NA_real_	Not Available / Missing Values
na.action	Handle Missing Values in Objects
na.action.default	Handle Missing Values in Objects
na.exclude	Handle Missing Values in Objects
na.exclude.data.frame	Handle Missing Values in Objects
na.exclude.default	Handle Missing Values in Objects
na.fail	Handle Missing Values in Objects
na.fail.default	Handle Missing Values in Objects
na.omit	Handle Missing Values in Objects
na.omit.data.frame	Handle Missing Values in Objects
na.omit.default	Handle Missing Values in Objects
na.pass	Handle Missing Values in Objects
NaN	Not Available / Missing Values
napredict	Adjust for Missing Values
napredict.default	Adjust for Missing Values
napredict.exclude	Adjust for Missing Values
napredict.NULL	Adjust for Missing Values
naprint	Print Missing Value Information
naprint.default	Print Missing Value Information
naprint.exclude	Print Missing Value Information
naprint.omit	Print Missing Value Information
naresid	Adjust for Missing Values

Function name	Title description
naresid.default	Adjust for Missing Values
naresid.exclude	Adjust for Missing Values
naresid.NULL	Adjust for Missing Values
NLSstAsymptotic	Fit the Asymptotic Regression Model
NLSstAsymptotic.sortedXyData	Fit the Asymptotic Regression Model
NLSstClosestX	Inverse Interpolation
NLSstClosestX.sortedXyData	Inverse Interpolation
NLSstLfAsymptote	Horizontal Asymptote on the Left Side
NLSstLfAsymptote.sortedXyData	Horizontal Asymptote on the Left Side
NLSstRtAsymptote	Horizontal Asymptote on the Right Side
NLSstRtAsymptote.sortedXyData	Horizontal Asymptote on the Right Side
NULL	The Null Object
order	Vector of Indices That Sort Data
order.dendrogram	ID Numbers or Labels of the Leaves in a Dendrogram
paste	Concatenate Data to Make Character Data
pmatch	Partial Matching of Character Items in a Vector
print.rle	Run Length Encoding and Decoding
provideDimnames	Dimnames Attribute of an Object
rbind	Building a Matrix from Columns or Rows
regexec	Search for a Pattern in Text
regexpr	Match Patterns in Strings
reorder.dendrogram	Reorder a Dendrogram
rep	Replicate Data Values
rep_len	Replicate Data Values
rep.Date	Replicate Data Values
rep.default	Replicate Data Values
rep.factor	Replicate Data Values
rep.int	Replicate Data Values

Function name	Title description
rep.POSIXct	Replicate Data Values
rep.POSIXlt	Replicate Data Values
replace	Insert or Merge Data
reshape	Reshape Grouped Data
rev	Reverse the Order of an Object
rev.default	Reverse the Order of an Object
rle	Run Length Encoding and Decoding
row.names	Row Names Attribute
row.names.data.frame	Row Names Attribute
row.names<-	Row Names Attribute
row.names<-.data.frame	Row Names Attribute
rowMeans	Row and Column Summaries
rownames	Column and Row Names
rownames<-	Column and Row Names
rowsum	Group Row Sums of a Matrix
rowsum.default	Group Row Sums of a Matrix
rowSums	Row and Column Summaries
seq	Generate a Sequence
seq_along	Generate a Sequence
seq_len	Generate a Sequence
seq.Date	Sequences of Date-Times
seq.default	Generate a Sequence
seq.int	Generate a Sequence
seq.POSIXt	Sequences of Date-Times
sequence	Create A Vector of Sequences
sort	Sort into Numeric or Alphabetic Order
sort.default	Sort into Numeric or Alphabetic Order
sort.int	Sort into Numeric or Alphabetic Order

Function name	Title description
sort.list	Vector of Indices That Sort Data
sort.POSIXlt	Sort into Numeric or Alphabetic Order
split	Split Data by Groups
split.data.frame	Split Data by Groups
split.Date	Split Data by Groups
split.default	Split Data by Groups
split.POSIXct	Split Data by Groups
split<-	Split Data by Groups
split<-.data.frame	Split Data by Groups
split<-.default	Split Data by Groups
strsplit	Split the Elements of a Character Vector
structure	An Object with Given Attributes
sub	Replace Part of a Character String
subset	Subsetting Vectors, Matrices and Data Frames
subset.data.frame	Subsetting Vectors, Matrices and Data Frames
subset.default	Subsetting Vectors, Matrices and Data Frames
subset.matrix	Subsetting Vectors, Matrices and Data Frames
tail	Get the First or Last Part of an Object
tolower	Convert Case of Character Strings
toupper	Convert Case of Character Strings
transform	Transform an Object to a Data Frame Object
transform.data.frame	Transform an Object to a Data Frame Object
transform.default	Transform an Object to a Data Frame Object
unique	Unique Values
unique.array	Unique Values
unique.data.frame	Unique Values
unique.default	Unique Values
unique.matrix	Unique Values

Function name	Title description
unique.POSIXlt	Unique Values
unlist	Simplify the Structure of a List
unname	Remove "names" or "dimnames"
unsplit	Split Data by Groups
Vectorize	Apply a Function to Multiple List or Vector Arguments
which	Find TRUE Values
zapsmall	Coerce Small Numbers to Zero for Printing

Data Types (not OO)

These are the available functions for (non object-oriented) data types. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
~	Model Formula Objects
array	Multi-Way Arrays
as.array	Multi-Way Arrays
as.array.default	Multi-Way Arrays
as.character	Character Objects
as.character.factor	Character Objects
as.complex	Complex Valued Objects
as.data.frame	Construct a Data Frame Object
as.double	Double Precision Objects
as.double.difftime	Double Precision Objects
as.double.POSIXlt	Double Precision Objects
as.factor	Create Factor Object
as.function	Function Objects
as.integer	Integer Objects
as.list	List Objects
as.list.data.frame	List Objects
as.list.Date	List Objects

Function name	Title description
as.list.default	List Objects
as.list.environment	List Objects
as.list.factor	List Objects
as.list.function	List Objects
as.list.numeric_version	List Objects
as.list.POSIXct	List Objects
as.logical	Logical Objects
as.logical.factor	Logical Objects
as.matrix	Matrix Objects
as.matrix.data.frame	Matrix Objects
as.matrix.default	Matrix Objects
as.matrix.noquote	Matrix Objects
as.matrix.POSIXlt	Matrix Objects
as.numeric	Numeric Objects
as.raw	Create a Raw Vector
as.single	Single Precision Objects
as.single.default	Single Precision Objects
as.ts	Time Series Objects
as.vector	Vectors (Simple Objects)
binomial	Generate a Family Object
character	Character Objects
charToRaw	Convert to or from Raw Vectors
class	Class Attribute of an Object
complex	Complex Valued Objects
data.class	Class of an Object
data.frame	Construct a Data Frame Object
double	Double Precision Objects
factor	Create Factor Object

Function name	Title description
family	Generate a Family Object
family.object	Family of GLM Models
formula.object	Model Formula Objects
Gamma	Generate a Family Object
gaussian	Generate a Family Object
glm.object	Generalized Linear Model Object
inherits	Test Inheritance of an Object
integer	Integer Objects
intToBits	Convert to or from Raw Vectors
inverse.gaussian	Generate a Family Object
is.array	Multi-Way Arrays
is.atomic	Test for Atomic or Recursive Objects
is.character	Character Objects
is.complex	Complex Valued Objects
is.data.frame	Construct a Data Frame Object
is.double	Double Precision Objects
is.factor	Create Factor Object
is.function	Function Objects
is.integer	Integer Objects
is.language	Test for Atomic or Recursive Objects
is.list	List Objects
is.logical	Logical Objects
is.matrix	Matrix Objects
is.mts	Time Series Objects
is.numeric	Numeric Objects
is.numeric.Date	Numeric Objects
is.numeric.difftime	Numeric Objects
is.numeric.POSIXt	Numeric Objects

Function name	Title description
is.object	Test if an Object has a Class Attribute
is.raw	Create a Raw Vector
is.recursive	Test for Atomic or Recursive Objects
is.single	Single Precision Objects
is.ts	Time Series Objects
is.vector	Vectors (Simple Objects)
list	List Objects
lm.object	Linear Least Squares Model Object
loess.object	Loess Model Object
logical	Logical Objects
matrix	Matrix Objects
methods	List Methods of Old-Style (SV3) Generic Functions
mlm.object	Linear Least Squares Model Object
NextMethod	Methods Invoked from Functions
numeric	Numeric Objects
numToBits	Convert to or from Raw Vectors
numToInts	Convert to or from Raw Vectors
packBits	Convert to or from Raw Vectors
poisson	Generate a Family Object
quasi	Generate a Family Object
quasibinomial	Generate a Family Object
quasipoisson	Generate a Family Object
raw	Create a Raw Vector
rawShift	Convert to or from Raw Vectors
rawToBits	Convert to or from Raw Vectors
rawToChar	Convert to or from Raw Vectors
single	Single Precision Objects
terms.object	Class of Objects for Terms in a Model

Function name	Title description
ts	Time Series Objects
unclass	Class Attribute of an Object
UseMethod	Methods Invoked from Functions
vector	Vectors (Simple Objects)

Dates and Times

These are the available functions for dates and times. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
-.Date	Ops Group Method for Date/Time Objects
-.POSIXt	Ops Group Method for Date/Time Objects
.difftime	Time Intervals
.leap.seconds	Date-Time Classes
.POSIXct	Date-Time Classes
.POSIXlt	Date-Time Classes
[.Date	Date Class
[.difftime	Time Intervals
[.POSIXct	Date-Time Classes
[.POSIXlt	Date-Time Classes
[[.Date	Date Class
[[.POSIXct	Date-Time Classes
[<-.Date	Date Class
[<-.POSIXct	Date-Time Classes
[<-.POSIXlt	Date-Time Classes
*.difftime	Ops Group Method for Date/Time Objects
/.difftime	Ops Group Method for Date/Time Objects
+.Date	Ops Group Method for Date/Time Objects
+.POSIXt	Ops Group Method for Date/Time Objects
as.character.Date	Date-Time Formatting

Function name	Title description
as.character.POSIXt	Date-Time Formatting
as.Date	Date-Time Parsing
as.Date.character	Date-Time Parsing
as.Date.date	Date-Time Parsing
as.Date.dates	Date-Time Parsing
as.Date.default	Date-Time Parsing
as.Date.factor	Date-Time Parsing
as.Date.numeric	Date-Time Parsing
as.Date.POSIXct	Date-Time Parsing
as.Date.POSIXlt	Date-Time Parsing
as.difftime	Date-Time Parsing
as.POSIXct	Date-Time Parsing
as.POSIXct.date	Date-Time Parsing
as.POSIXct.Date	Date-Time Parsing
as.POSIXct.dates	Date-Time Parsing
as.POSIXct.default	Date-Time Parsing
as.POSIXct.numeric	Date-Time Parsing
as.POSIXct.POSIXlt	Date-Time Parsing
as.POSIXlt	Date-Time Parsing
as.POSIXlt.character	Date-Time Parsing
as.POSIXlt.date	Date-Time Parsing
as.POSIXlt.Date	Date-Time Parsing
as.POSIXlt.dates	Date-Time Parsing
as.POSIXlt.default	Date-Time Parsing
as.POSIXlt.factor	Date-Time Parsing
as.POSIXlt.numeric	Date-Time Parsing
as.POSIXlt.POSIXct	Date-Time Parsing
c.difftime	Time Intervals

Function name	Title description
check_tzones	Get the Current Date, Time, or Time Zone
cut.Date	Create Factor by Cutting Date or POSIXt Object
cut.POSIXt	Create Factor by Cutting Date or POSIXt Object
date	Get the Current Date, Time, or Time Zone
Date	Date Class
difftime	Time Intervals
format.Date	Date-Time Formatting
format.difftime	Date-Time Formatting
format.POSIXct	Date-Time Formatting
format.POSIXlt	Date-Time Formatting
ISOdate	Construct Date-time from Broken-down Time
ISOdatetime	Construct Date-time from Broken-down Time
julian	Extract Parts of a Date or POSIXt Object
julian.Date	Extract Parts of a Date or POSIXt Object
julian.POSIXt	Extract Parts of a Date or POSIXt Object
Math.Date	Math Group Method for Date/Time Objects
Math.difftime	Math Group Method for Date/Time Objects
Math.POSIXt	Math Group Method for Date/Time Objects
months	Extract Parts of a Date or POSIXt Object
months.Date	Extract Parts of a Date or POSIXt Object
months.POSIXt	Extract Parts of a Date or POSIXt Object
OlsonNames	Time Zones
Ops.Date	Ops Group Method for Date/Time Objects
Ops.difftime	Ops Group Method for Date/Time Objects
Ops.POSIXt	Ops Group Method for Date/Time Objects
POSIXct	Date-Time Classes
POSIXlt	Date-Time Classes
POSIXt	Date-Time Classes

Function name	Title description
print.Date	Date Class
print.difftime	Time Intervals
print.POSIXct	Date-Time Classes
print.POSIXlt	Date-Time Classes
quarters	Extract Parts of a Date or POSIXt Object
quarters.Date	Extract Parts of a Date or POSIXt Object
quarters.POSIXt	Extract Parts of a Date or POSIXt Object
round.Date	Math Group Method for Date/Time Objects
round.POSIXt	Math Group Method for Date/Time Objects
seq.Date	Sequences of Date-Times
seq.POSIXt	Sequences of Date-Times
strftime	Date-Time Formatting
strptime	Date-Time Parsing
Summary.Date	Summary Group Method for Date/Time Objects
Summary.difftime	Summary Group Method for Date/Time Objects
Summary.POSIXct	Summary Group Method for Date/Time Objects
Summary.POSIXlt	Summary Group Method for Date/Time Objects
Sys.Date	Get the Current Date, Time, or Time Zone
Sys.time	Get the Current Date, Time, or Time Zone
Sys.timezone	Get the Current Date, Time, or Time Zone
trunc.Date	Math Group Method for Date/Time Objects
trunc.POSIXt	Math Group Method for Date/Time Objects
units	Units for Time Intervals
units.difftime	Units for Time Intervals
units<-	Units for Time Intervals
units<-.difftime	Units for Time Intervals
weekdays	Extract Parts of a Date or POSIXt Object
weekdays.Date	Extract Parts of a Date or POSIXt Object

Function name	Title description
weekdays.POSIXt	Extract Parts of a Date or POSIXt Object

Environments, Scoping, and Packages

These are the available functions for environments, scoping, and packages. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
<<-	Assign a Name to an Object
<-	Assign a Name to an Object
->	Assign a Name to an Object
.GlobalEnv	Environment Access
.Library	Search Paths for Packages
.Library.site	Search Paths for Packages
.dynLibs	Loading DLLs from Packages
.expand_R_libs_env_var	Search Paths for Packages
.getNamespace	Get Namespace Environment Information
.libPaths	Search Paths for Packages
.onAttach	Load and List Packages
.onLoad	Load and List Packages
.onUnload	Load and List Packages
=	Assign a Name to an Object
Assignment	Assign a Name to an Object
R_LIBS	Search Paths for Packages
R_LIBS_SITE	Search Paths for Packages
R_LIBS_USER	Search Paths for Packages
apropos	Find Objects by (Partial) Name
as.environment	Coerce to an Environment Object
assign	Assign Object to Environment
attach	Attach a Set of Objects to the Search Path
attachNamespace	Loading and Unloading Namespaces

Function name	Title description
baseenv	Environment Access
cbind.data.frame	Build Data Frame from Columns
dependsOnPkgs	Find Reverse Dependencies
data	Data Sets
detach	Detach Data from the Search List
dget	Read or Write a Text Representation of an object
dput	Read or Write a Text Representation of an object
dump	Produce Text Representations of Objects
emptyenv	Environment Access
environment	Environment Access
environment<-	Environment Access
environmentName	Environment Access
evalOnLoad	Set Actions For Package Loading
evalqOnLoad	Set Actions For Package Loading
exists	Determine if an Object is Defined
extSoftVersion	Get Third Party Software Information
find	Find Packages that Contain an Object
get	Search for a Named Object
get0	Determine if an Object is Defined
getExportedValue	Get Namespace Environment Information
getLoadActions	Set Actions For Package Loading
getNamespace	Get Namespace Environment Information
getNamespaceExports	Get Namespace Environment Information
getNamespaceImports	Get Namespace Environment Information
getNamespaceInfo	Get Namespace Environment Information
getNamespaceName	Get Namespace Environment Information
getNamespaceUsers	Get Namespace Environment Information
getNamespaceVersion	Get Namespace Environment Information

Function name	Title description
globalenv	Environment Access
hasLoadAction	Set Actions For Package Loading
is.environment	Environment Access
isBaseNamespace	Check if an Environment is a (base) Namespace Environment
isNamespace	Check if an Environment is a (base) Namespace Environment
isNamespaceLoaded	Get Namespace Environment Information
library	Load and List Packages
library.dynam	Loading DLLs from Packages
library.dynam.unload	Loading DLLs from Packages
loadNamespace	Loading and Unloading Namespaces
loadedNamespaces	Loading and Unloading Namespaces
mget	Search for a Named Object
new.env	Environment Access
packageDate	Read a Package's DESCRIPTION File
parent.env	Environment Access
parent.env<-	Environment Access
print.packageIQR	Data Sets
rbind.data.frame	Create a Data Frame from Rows
require	Load and List Packages
requireNamespace	Loading and Unloading Namespaces
R_user_dir	R User Directories
search	Search List
searchpaths	Search List
setLoadAction	Set Actions For Package Loading
setLoadActions	Set Actions For Package Loading
topenv	Top-Level Environment
unloadNamespace	Loading and Unloading Namespaces

Lists

These are the available functions for lists. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
[	Extract or Replace Parts of an Object
[.data.frame	Extract or Replace Parts of an Object
[.factor	Extract or Replace Parts of an Object
[[	Extract or Replace Parts of an Object
[[.data.frame	Extract or Replace Parts of an Object
[[.factor	Extract or Replace Parts of an Object
[[<-	Extract or Replace Parts of an Object
[[<-.data.frame	Extract or Replace Parts of an Object
[<-	Extract or Replace Parts of an Object
[<-.data.frame	Extract or Replace Parts of an Object
[<-.factor	Extract or Replace Parts of an Object
$	Extract or Replace Parts of an Object
$<-	Extract or Replace Parts of an Object
as.list	List Objects
as.list.data.frame	List Objects
as.list.Date	List Objects
as.list.default	List Objects
as.list.environment	List Objects
as.list.factor	List Objects
as.list.function	List Objects
as.list.numeric_version	List Objects
as.list.POSIXct	List Objects
as.null	The Null Object
as.null.default	The Null Object
asplit	Split Array into List of Subarrays
c	Combine Values into a Vector or List

Function name	Title description
c.Date	Combine Values into a Vector or List
c.POSIXct	Combine Values into a Vector or List
c.POSIXlt	Combine Values into a Vector or List
eapply	Apply a Function Over Values in an Environment
Extract	Extract or Replace Parts of an Object
hasName	Names Attribute of an Object
is.list	List Objects
is.null	The Null Object
lapply	Apply a Function to Components of a List or Vector
length	Length of a Vector or List
length.POSIXlt	Length of a Vector or List
list	List Objects
names	Names Attribute of an Object
names.POSIXlt	Names Attribute of an Object
names<-	Names Attribute of an Object
names<-.POSIXlt	Names Attribute of an Object
NULL	The Null Object
rev	Reverse the Order of an Object
rev.default	Reverse the Order of an Object
sapply	Apply a Function to Components of a List or Vector
setNames	Attach a names attribute to an object
simplify2array	Apply a Function to Components of a List or Vector
split	Split Data by Groups
split.data.frame	Split Data by Groups
split.Date	Split Data by Groups
split.default	Split Data by Groups
split.POSIXct	Split Data by Groups
split<-	Split Data by Groups

Function name	Title description
split<-.data.frame	Split Data by Groups
split<-.default	Split Data by Groups
Subscript	Extract or Replace Parts of an Object
Subscript.data.frame	Extract or Replace Parts of an Object
Subscript.factor	Extract or Replace Parts of an Object
unlist	Simplify the Structure of a List
unsplit	Split Data by Groups
vapply	Apply a Function to Components of a List or Vector

Graphics

These graphics functions are available in TERR. For help with a function, see its listing in the TERR Language Reference.

Color

These are the available functions for color. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
adjustcolor	Alter a color specification
col2rgb	Interpret Color Names, Strings, and Numbers
colorRamp	Ramp/Interpolate Colors
colorRampPalette	Ramp/Interpolate Colors
colors	Get Color Names
extendrange	Get Color Names
gray, grey	Palette of Grays
gray.colors, grey.colors	Groups of Colors
hcl	Hue, Chroma, Luminance Color Model
hsv	Hue, Saturation, Value Color Specification
palette	Color Palette
rainbow	Rainbow Colors
rgb	Convert Numeric Color Values to Character Color Codes

Function name	Title description
rgb2hsv	Hue, Saturation, Value Color Model

Computations Related to Plotting (Graphics)

These are the available functions for computations related to plotting. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
acf	Auto- and Cross- Covariance or Correlation Estimation
approx	Interpolation Functions
approxfun	Interpolation Functions
ccf	Auto- and Cross- Covariance or Correlation Estimation
density	Kernel Estimate of Probability Density Function
density.default	Kernel Estimate of Probability Density Function
ecdf	Empirical Cumulative Distribution Function
extendedrange	Numbers a bit outside the range of a vector
pacf	Auto- and Cross- Covariance or Correlation Estimation
plot.ecdf	Empirical Cumulative Distribution Function
ppoints	Plotting Points for Quantile-Quantile Plots
print.ecdf	Empirical Cumulative Distribution Function
qqnorm	Normal Quantile-Quantile Plots
qqnorm.default	Normal Quantile-Quantile Plots
quantile.ecdf	Empirical Cumulative Distribution Function
range	Get the Range of Data
spline	Interpolating Splines
splinefun	Interpolating Splines
splinefunH	Interpolating Splines
summary.ecdf	Empirical Cumulative Distribution Function
xy.coords	X,Y Arguments
xyTable	Tabulate repeated x-y points

Devices

These are the available functions for interacting with devices. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
as.raster	Manipulate raster objects.
is.raster	Manipulate raster objects.
terrFakeDev	Fake Graphics Device for TERR

High-Level Plots

These are the available functions for high-level plots. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
[[.dendrogram	General Tree Structures
as.dendrogram	General Tree Structures
as.dendrogram.dendrogram	General Tree Structures
as.dendrogram.hclust	General Tree Structures
cut.dendrogram	General Tree Structures
dendrogram	General Tree Structures
ecdf	Empirical Cumulative Distribution Function
is.leaf	General Tree Structures
plot.dendrogram	General Tree Structures
plot.ecdf	Empirical Cumulative Distribution Function
print.dendrogram	General Tree Structures
print.ecdf	Empirical Cumulative Distribution Function
qqnorm	Normal Quantile-Quantile Plots
qqnorm.default	Normal Quantile-Quantile Plots
quantile.ecdf	Empirical Cumulative Distribution Function
str.dendrogram	General Tree Structures
summary.ecdf	Empirical Cumulative Distribution Function

Interacting with Plots

These are the available functions for interacting with plots. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
menu	Menu Interaction Function

Mathematics

These mathematic functions are available in TERR. For help with a function, see its listing in the Language Reference.

Basic Arithmetic and Sorting

These are the available functions for basic arithmetic. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
%*%	Matrix Multiplication
gl	Generate Patterned Factor
matmult	Matrix Multiplication
ppoints	Plotting Points for Quantile-Quantile Plots

Linear Algebra

These are the available functions for linear algebra. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
%c%	Matrix Cross Product
%o%	Generalized Outer Products
aperm	Array Permutations
aperm.default	Array Permutations
aperm.table	Array Permutations
apply	Apply a Function to Sections of an Array
backsolve	Backsolve Upper or Lower Triangular Equations
chol	Choleski Decomposition of Symmetric Matrix
chol.default	Choleski Decomposition of Symmetric Matrix
chol2inv	Invert a Matrix Given its Choleski Decomposition

Function name	Title description
colMeans	Row and Column Summaries
colSums	Row and Column Summaries
crossprod	Matrix Cross Product
det	Determinant of a Matrix
determinant	Determinant of a Matrix
diag	Diagonal Matrices
eigen	Eigenvalues and Eigenvectors of a Matrix
eigen.default	Eigenvalues and Eigenvectors of a Matrix
is.qr	QR Matrix Decomposition
La.svd	Singular Value Decomposition of a Matrix
outer	Generalized Outer Products
prcomp	Principal Components Analysis
prcomp.default	Principal Components Analysis
prcomp.formula	Principal Components Analysis
qr	QR Matrix Decomposition
qr.coef	Use a QR Matrix Decomposition
qr.default	QR Matrix Decomposition
qr.fitted	Use a QR Matrix Decomposition
qr.lm	QR Matrix Decomposition
qr.Q	Reconstruct the Q, R, or X Matrices from a QR Object
qr.qty	Use a QR Matrix Decomposition
qr.qy	Use a QR Matrix Decomposition
qr.R	Reconstruct the Q, R, or X Matrices from a QR Object
qr.resid	Use a QR Matrix Decomposition
qr.solve	Use a QR Matrix Decomposition
qr.X	Reconstruct the Q, R, or X Matrices from a QR Object
rowMeans	Row and Column Summaries
rowSums	Row and Column Summaries

Function name	Title description
scale	Scale Columns of a Matrix
scale.default	Scale Columns of a Matrix
solve	Solve Linear Equations and Invert Matrices
solve.default	Solve Linear Equations and Invert Matrices
solve.qr	Solve Linear Equations and Invert Matrices
svd	Singular Value Decomposition of a Matrix
t	Transpose a Matrix
t.default	Transpose a Matrix
tcrossprod	Matrix Cross Product

Logical Operators

These are the available functions for logical operators. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
!	Logical Operators
!=	Comparison Operators
&	Logical Operators
<	Comparison Operators
>	Comparison Operators
<=	Comparison Operators
>=	Comparison Operators
==	Comparison Operators
|	Logical Operators
||	Control Flow
all	Logical Sum and Product
all.equal	Test Two Objects for Full Equality
all.equal.factor	Test Two Objects for Full Equality
all.equal.numeric	Test Two Objects for Full Equality
all.equal.POSIXct	Test Two Objects for Full Equality

Function name	Title description
any	Logical Sum and Product
anyDuplicated	Determine Duplicate Elements
anyDuplicated.array	Determine Duplicate Elements
anyDuplicated.data.frame	Determine Duplicate Elements
anyDuplicated.default	Determine Duplicate Elements
anyDuplicated.matrix	Determine Duplicate Elements
anyNA	Quickly check for missing (NA) values
arrayInd	Find TRUE Values
as.logical	Logical Objects
as.logical.factor	Logical Objects
bitwAnd	Bitwise Logical Operations
bitwNot	Bitwise Logical Operations
bitwOr	Bitwise Logical Operations
bitwShiftL	Bitwise Logical Operations
bitwShiftR	Bitwise Logical Operations
bitwXor	Bitwise Logical Operations
break	Control Flow
Comparison	Comparison Operators
complete.cases	Find Complete Cases of Observations
Control	Control Flow
duplicated	Determine Duplicate Elements
duplicated.array	Determine Duplicate Elements
duplicated.data.frame	Determine Duplicate Elements
duplicated.default	Determine Duplicate Elements
duplicated.matrix	Determine Duplicate Elements
duplicated.POSIXlt	Determine Duplicate Elements
else	Control Flow
for	Control Flow

Function name	Title description
identical	Test for Complete Equality
if	Control Flow
ifelse	Conditional Data Selection
in	Control Flow
is.finite	Check IEEE Arithmetic Values
is.infinite	Check IEEE Arithmetic Values
is.logical	Logical Objects
is.na	Not Available / Missing Values
is.na.data.frame	Not Available / Missing Values
is.na.POSIXlt	Not Available / Missing Values
is.na<-	Not Available / Missing Values
is.na<-.default	Not Available / Missing Values
is.na<-.factor	Not Available / Missing Values
is.nan	Check IEEE Arithmetic Values
isTRUE	Test for the Value TRUE
Logic	Logical Operators
logical	Logical Objects
NA	Not Available / Missing Values
NA_character_	Not Available / Missing Values
NA_complex_	Not Available / Missing Values
NA_integer_	Not Available / Missing Values
NA_real_	Not Available / Missing Values
NaN	Not Available / Missing Values
next	Control Flow
repeat	Control Flow
sign	Signum Function
which	Find TRUE Values
which.max	Find the Index of the Minimum or Maximum Value

Function name	Title description
which.min	Find the Index of the Minimum or Maximum Value
while	Control Flow
xor	Logical Operators

Mathematical, Calculus, and Others

These are the available functions for math, calculus, and so on. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
!=	Comparison Operators
%/%	Arithmetic Operators
%%	Arithmetic Operators
^	Arithmetic Operators
+	Arithmetic Operators
<	Comparison Operators
<=	Comparison Operators
==	Comparison Operators
abs	Absolute Value
acos	Inverse Trigonometric Functions
acosh	Inverse Hyperbolic Trigonometric Functions
anyNA	Quickly check for missing (NA) values
approx	Interpolation Functions
approxfun	Interpolation Functions
Arithmetic	Arithmetic Operators
asin	Inverse Trigonometric Functions
asinh	Inverse Hyperbolic Trigonometric Functions
atan	Inverse Trigonometric Functions
atan2	Inverse Trigonometric Functions
atanh	Inverse Hyperbolic Trigonometric Functions
besselI	Bessel functions

Function name	Title description
besselJ	Bessel functions
besselK	Bessel functions
besselY	Bessel functions
ceiling	Integer Values
choose	Factorial, Combinations, and Permutations
colMeans	Row and Column Summaries
colSums	Row and Column Summaries
combn	Generate combinations of m elements out of x
Comparison	Comparison Operators
cos	Trigonometric Functions
cosh	Hyperbolic Trigonometric Functions
cospi	Trigonometric Functions
cummax	Cumulative Maxima and Minima
cummin	Cumulative Maxima and Minima
cumprod	Cumulative Sums and Products
cumsum	Cumulative Sums and Products
diff	Create an Object of Differences
diff.Date	Create an Object of Differences
diff.default	Create an Object of Differences
diff.POSIXt	Create an Object of Differences
diff.ts	Create an Object of Differences
digamma	Gamma Function (and Its Derivatives and Logarithm)
exp	Exponential and related Functions
expm1	Exponential and related Functions
factorial	Factorial, Combinations, and Permutations
floor	Integer Values
gamma	Gamma Function (and Its Derivatives and Logarithm)
is.finite	Check IEEE Arithmetic Values

Function name	Title description
is.infinite	Check IEEE Arithmetic Values
is.nan	Check IEEE Arithmetic Values
lchoose	Factorial, Combinations, and Permutations
lfactorial	Factorial, Combinations, and Permutations
lgamma	Gamma Function (and Its Derivatives and Logarithm)
log	Exponential and related Functions
log10	Exponential and related Functions
log1p	Exponential and related Functions
log2	Exponential and related Functions
logb	Exponential and related Functions
Math.data.frame	Math Group Method for Data Frame Objects
max	Extremes
mean	Mean Value (Arithmetic Average)
mean.data.frame	Mean Value (Arithmetic Average)
mean.Date	Mean Value (Arithmetic Average)
mean.default	Mean Value (Arithmetic Average)
mean.difftime	Mean Value (Arithmetic Average)
mean.POSIXct	Mean Value (Arithmetic Average)
mean.POSIXlt	Mean Value (Arithmetic Average)
median	Median
median.default	Median
min	Extremes
nextn	Highly Composite Numbers
optimHess	Numerically Estimate Hessian Matrix
pmax	Parallel Maximum or Minimum
pmax.int	Parallel Maximum or Minimum
pmin	Parallel Maximum or Minimum
pmin.int	Parallel Maximum or Minimum

Function name	Title description
polyroot	Find the Roots of a Polynomial
prod	Sums and Products
psigamma	Gamma Function (and Its Derivatives and Logarithm)
quantile	Empirical Quantiles
range	Get the Range of Data
rank	Ranks of Data
round	Rounding Functions
rowMeans	Row and Column Summaries
rowSums	Row and Column Summaries
signif	Rounding Functions
sin	Trigonometric Functions
sinh	Hyperbolic Trigonometric Functions
sinpi	Trigonometric Functions
spline	Interpolating Splines
splinefun	Interpolating Splines
splinefunH	Interpolating Splines
sqrt	Exponential and related Functions
sum	Sums and Products
tan	Trigonometric Functions
tanh	Hyperbolic Trigonometric Functions
tanpi	Trigonometric Functions
trig	Trigonometric Functions
trigamma	Gamma Function (and Its Derivatives and Logarithm)
trunc	Integer Values
which.max	Find the Index of the Minimum or Maximum Value
which.min	Find the Index of the Minimum or Maximum Value
xtfrm	Sorting and Ranking
zapsmall	Coerce Small Numbers to Zero for Printing

Matrices and Arrays

These are the available functions for matrices and arrays. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
[	Extract or Replace Parts of an Object
[.data.frame	Extract or Replace Parts of an Object
[.factor	Extract or Replace Parts of an Object
[[	Extract or Replace Parts of an Object
[[.data.frame	Extract or Replace Parts of an Object
[[.factor	Extract or Replace Parts of an Object
[[<-	Extract or Replace Parts of an Object
[[<-.data.frame	Extract or Replace Parts of an Object
[<-	Extract or Replace Parts of an Object
[<-.data.frame	Extract or Replace Parts of an Object
[<-.factor	Extract or Replace Parts of an Object
%*%	Matrix Multiplication
%c%	Matrix Cross Product
%x%	Generalized Kronecker Products
$	Extract or Replace Parts of an Object
$<-	Extract or Replace Parts of an Object
.col	Column and Row Identification in a Matrix
.row	Column and Row Identification in a Matrix
aggregate	Compute Summary Statistics of Subsets of Data
aggregate.data.frame	Compute Column-by-Column Summaries of Groups of Observations
aggregate.default	Compute Summary Statistics of Subsets of Data
aggregate.formula	Compute Summary Statistics of Subsets of Data
aggregate.ts	Compute Summary Statistics of Subsets of Data
aperm	Array Permutations
aperm.default	Array Permutations
aperm.table	Array Permutations

Function name	Title description
apply	Apply a Function to Sections of an Array
array	Multi-Way Arrays
as.array	Multi-Way Arrays
as.array.default	Multi-Way Arrays
as.matrix	Matrix Objects
as.matrix.data.frame	Matrix Objects
as.matrix.default	Matrix Objects
as.matrix.noquote	Matrix Objects
as.matrix.POSIXlt	Matrix Objects
backsolve	Backsolve Upper or Lower Triangular Equations
by	Split a Data Frame and Apply a Function to the Parts
by.data.frame	Split a Data Frame and Apply a Function to the Parts
by.default	Split a Data Frame and Apply a Function to the Parts
cbind	Building a Matrix from Columns or Rows
chol	Choleski Decomposition of Symmetric Matrix
chol.default	Choleski Decomposition of Symmetric Matrix
chol2inv	Invert a Matrix Given its Choleski Decomposition
col	Column and Row Identification in a Matrix
colMeans	Row and Column Summaries
colnames	Column and Row Names
colnames<-	Column and Row Names
colSums	Row and Column Summaries
cor	Correlation, Variance, and Covariance (Matrices)
cov	Correlation, Variance, and Covariance (Matrices)
cov2cor	Correlation, Variance, and Covariance (Matrices)
crossprod	Matrix Cross Product
data.matrix	Coerce Data Frame to Numeric Matrix
diag	Diagonal Matrices

Function name	Title description
dim	Dim Attribute of an Object
dim<-	Dim Attribute of an Object
dimnames	Dimnames Attribute of an Object
dimnames.data.frame	Dimnames Attribute of an Object
drop	Drop Length One Dimensions of an Array
eigen	Eigenvalues and Eigenvectors of a Matrix
eigen.default	Eigenvalues and Eigenvectors of a Matrix
Extract	Extract or Replace Parts of an Object
is.array	Multi-Way Arrays
is.matrix	Matrix Objects
isSymmetric	Test if an Object is Symmetric
isSymmetric.matrix	Test if an Object is Symmetric
kronecker	Generalized Kronecker Products
La.svd	Singular Value Decomposition of a Matrix
lower.tri	Logical Matrix of the Lower or Upper Triangle
margin.table	Compute Table Margin
mat.or.vec	Create a Matrix or a Vector
matmult	Matrix Multiplication
matrix	Matrix Objects
merge	Merge Two Datasets and Match Columns
merge.data.frame	Merge Two Datasets and Match Columns
merge.default	Merge Two Datasets and Match Columns
ncol	Get the Number of Rows or Columns of an Array or Matrix
NCOL	Get the Number of Rows or Columns of an Array or Matrix
nrow	Get the Number of Rows or Columns of an Array or Matrix
NROW	Get the Number of Rows or Columns of an Array or Matrix
prop.table	Express Table Entries as Fraction of Marginal Table
rapply	Apply a Function Recursively

Function name	Title description
rbind	Building a Matrix from Columns or Rows
row	Column and Row Identification in a Matrix
rowMeans	Row and Column Summaries
rownames	Column and Row Names
rownames<-	Column and Row Names
rowSums	Row and Column Summaries
scale	Scale Columns of a Matrix
scale.default	Scale Columns of a Matrix
solve	Solve Linear Equations and Invert Matrices
solve.default	Solve Linear Equations and Invert Matrices
solve.qr	Solve Linear Equations and Invert Matrices
Subscript	Extract or Replace Parts of an Object
Subscript.data.frame	Extract or Replace Parts of an Object
Subscript.factor	Extract or Replace Parts of an Object
svd	Singular Value Decomposition of a Matrix
sweep	Sweep Out Array Summaries
t	Transpose a Matrix
t.default	Transpose a Matrix
tapply	Apply a Function to a Ragged Array
tcrossprod	Matrix Cross Product
upper.tri	Logical Matrix of the Lower or Upper Triangle
var	Correlation, Variance, and Covariance (Matrices)

Optimization

These are the available functions for optimization. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
nlm	Nonlinear Minimization
nlminb	Nonlinear Minimization subject to Box Constraints

Function name	Title description
optim	General-purpose Optimization
optimHess	Numerically Estimate Hessian Matrix
optimise	Univariate Optimization of a Function
optimize	Univariate Optimization of a Function

Programming

These programming functions are available in TERR. For help with a function, see its listing in the TERR Language Reference.

Documentation

These are the available functions for documentation. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
?	Documentation Shortcuts
??	Search the Help System
apropos	Find Objects by (Partial) Name
args	Display the Argument List of a Function
browseVignettes	List Vignettes in an HTML Browser
demo	Demonstrations of Package Functionality
example	Run Examples Section from the Online Help
formals	Access and Manipulate the Formal Arguments
formals<-	Access and Manipulate the Formal Arguments
help	Online Documentation
help.search	Search the Help System
help.start	Hypertext Documentation
history	Print History of Evaluated Expressions
loadhistory	Print History of Evaluated Expressions
print.browseVignettes	List Vignettes in an HTML Browser
prompt	Generate a Skeleton Documentation File for an Object
promptData	Generate a Skeleton Documentation File for an Object

Function name	Title description
promptPackage	Make a skeleton help file for a package
Question	Documentation Shortcuts
savehistory	Print History of Evaluated Expressions
str	Compactly Display the Structure of an Object
str.data.frame	Compactly Display the Structure of an Object
str.default	Compactly Display the Structure of an Object
strOptions	Compactly Display the Structure of an Object
timestamp	Print History of Evaluated Expressions
vignetteInfo	Extract Metadata from Vignette Source File

Error Handling

These are the available functions for error handling. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
.Deprecated	Functions for Handling Unimplemented Functions and Arguments
.Defunct	Functions for Handling Unimplemented Functions and Arguments
.doTrace	Trace Calls to Functions
.handleSimpleError	Condition Handling and Recovery
.NotYetImplemented	Functions for Handling Unimplemented Functions and Arguments
.signalSimpleWarning	Condition Handling and Recovery
allowInterrupts	Disable or Enable User Interrupts
as.character.condition	Condition Handling and Recovery
as.character.error	Condition Handling and Recovery
assertCondition	Test Expected Errors and Warnings
assertError	Test Expected Errors and Warnings
assertWarning	Test Expected Errors and Warnings
browser	Browse Interactively in a Function's Frame
computeRestarts	Condition Handling and Recovery
condition	Condition Handling and Recovery

Function name	Title description
conditionCall	Condition Handling and Recovery
conditionCall.condition	Condition Handling and Recovery
conditionMessage	Condition Handling and Recovery
conditionMessage.condition	Condition Handling and Recovery
conditions	Condition Handling and Recovery
dump.frames	Save All Frames on Errors
findRestart	Condition Handling and Recovery
errorCondition	Condition Handling and Recovery
geterrmessage	Error and Warning Messages
getOption	Set or Return Options
globalCallingHandlers	Condition Handling and Recovery
invokeRestart	Condition Handling and Recovery
invokeRestartInteractively	Condition Handling and Recovery
isRestart	Condition Handling and Recovery
on.exit	Exit Expression For a Function
options	Set or Return Options
print.condition	Condition Handling and Recovery
print.restart	Condition Handling and Recovery
restartDescription	Condition Handling and Recovery
restartFormals	Condition Handling and Recovery
signalCondition	Condition Handling and Recovery
simpleCondition	Condition Handling and Recovery
simpleError	Condition Handling and Recovery
simpleMessage	Condition Handling and Recovery
simpleWarning	Condition Handling and Recovery
stop	Error and Warning Messages
stopifnot	Stop if Not All True
suppressWarnings	Error and Warning Messages

Function name	Title description
suspendInterrupts	Disable or Enable User Interrupts
trace	Trace Calls to Functions
traceback	Print Call Stack After Error
try	Continue after errors
tryCatch	Condition Handling and Recovery
tryInvokeRestart	Condition Handling and Recovery
unimplementedStop	Functions for Handling Unimplemented Functions and Arguments
unimplementedWarning	Functions for Handling Unimplemented Functions and Arguments
untrace	Trace Calls to Functions
warning	Error and Warning Messages
warningCondition	Condition Handling and Recovery
warnings	Saved Warning Messages
withCallingHandlers	Condition Handling and Recovery
withRestarts	Condition Handling and Recovery

Input and Output Connections

These are the available functions for input and output connections. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
bzfile	Functions to Manipulate Connections
clipboard	Functions to Manipulate Connections
close	Functions to Manipulate Connections
close.connection	Functions to Manipulate Connections
connection	Functions to Manipulate Connections
connections	Functions to Manipulate Connections
fifo	Functions to Manipulate Connections
file	Functions to Manipulate Connections
flush	Functions to Manipulate Connections
flush.connection	Functions to Manipulate Connections

Function name	Title description
gzfile	Functions to Manipulate Connections
isatty	Functions to Manipulate Connections
isIncomplete	Functions to Manipulate Connections
isOpen	Functions to Manipulate Connections
open	Functions to Manipulate Connections
open.connection	Functions to Manipulate Connections
nullfile	Name the Null File
pipe	Functions to Manipulate Connections
print.connection	Functions to Manipulate Connections
read.fwf	Read Fixed Width Format Files
readChar	Transfer Character Strings To and From Connections
readRDS	Serialization Interface for Single Objects
saveRDS	Serialization Interface for Single Objects
serialize	Simple Serialization Interface
socketConnection	Functions to Manipulate Connections
stderr	Functions to Manipulate Connections
stdin	Functions to Manipulate Connections
stdout	Functions to Manipulate Connections
summary.connection	Functions to Manipulate Connections
unserialize	Simple Serialization Interface
textConnection	Text Connections
textConnectionValue	Text Connections
unz	Functions to Manipulate Connections
url	Functions to Manipulate Connections
writeChar	Transfer Character Strings To and From Connections
xzfile	Functions to Manipulate Connections

Input and Output Files

These are the available functions for input and output files. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
basename	Manipulate File Paths
browseURL	Displays the contents of a URL in a Web Browser
bzfile	Functions to Manipulate Connections
cat	Print the Arguments
clipboard	Functions to Manipulate Connections
close	Functions to Manipulate Connections
close.connection	Functions to Manipulate Connections
connection	Functions to Manipulate Connections
connections	Functions to Manipulate Connections
count.fields	Count the Number of Fields per Line
dcf	Read and Write Data in DCF Format
dget	Read or Write a Text Representation of an object
dir	List the Files in a Directory
dir.create	File and Directory Manipulation
dirname	Manipulate File Paths
dput	Read or Write a Text Representation of an object
dump	Produce Text Representations of Objects
fifo	Functions to Manipulate Connections
file	Functions to Manipulate Connections
file.append	File and Directory Manipulation
file.copy	File and Directory Manipulation
file.create	File and Directory Manipulation
file.exists	File and Directory Manipulation
file.info	Extract File Information
file.link	File and Directory Manipulation
file.mode	Extract File Information
file.mtime	Extract File Information
file.path	Construct Path to File

Function name	Title description
file.remove	File and Directory Manipulation
file.rename	File and Directory Manipulation
file.size	Extract File Information
file.symlink	File and Directory Manipulation
file_test	Shell-style Tests of Files
flush	Functions to Manipulate Connections
flush.connection	Functions to Manipulate Connections
gzfile	Functions to Manipulate Connections
history	Print History of Evaluated Expressions
isatty	Functions to Manipulate Connections
isIncomplete	Functions to Manipulate Connections
isOpen	Functions to Manipulate Connections
list.dirs	List the Files in a Directory
list.files	List the Files in a Directory
load	Reload Saved Datasets
loadhistory	Print History of Evaluated Expressions
nullfile	Name the Null File
open	Functions to Manipulate Connections
open.connection	Functions to Manipulate Connections
package.skeleton	Create a Skeleton for a New Source Package
path.expand	Expand ~ in File Paths
pipe	Functions to Manipulate Connections
print.connection	Functions to Manipulate Connections
R.home	Paths to Files in the TIBCO Enterprise Runtime for R Installation
read.csv	Create a Data Frame by Reading a Table
read.csv2	Create a Data Frame by Reading a Table
read.dcf	Read and Write Data in DCF Format
read.delim	Create a Data Frame by Reading a Table

Function name	Title description
read.delim2	Create a Data Frame by Reading a Table
read.fwf	Read Fixed Width Format Files
read.table	Create a Data Frame by Reading a Table
readBin	Read and Write Binary Data
readChar	Transfer Character Strings To and From Connections
readline	Read a Line from the Terminal
readLines	Read or Write Multiple Lines from or to a Connection
readRDS	Serialization Interface for Single Objects
save	Save Objects
save.image	Save Objects
savehistory	Print History of Evaluated Expressions
saveRDS	Serialization Interface for Single Objects
scan	Input Data from a File or Connection
serialize	Simple Serialization Interface
shortPathName	Convert File Name to DOS 8.3 Format
sink	Send Output to a File
sink.number	Send Output to a File
socketConnection	Functions to Manipulate Connections
source	Read Expressions from a File or a Connection
stderr	Functions to Manipulate Connections
stdin	Functions to Manipulate Connections
stdout	Functions to Manipulate Connections
summary.connection	Functions to Manipulate Connections
Sys.getenv	Get Environment Variables
Sys.getlocale	Set or Get Locale-Specific Information
Sys.glob	File and Directory Manipulation
Sys.localeconv	Set or Get Locale-Specific Information
Sys.readlink	File and Directory Manipulation

Function name	Title description
Sys.setenv	Set or Unset Environment Variables
Sys.setlocale	Set or Get Locale-Specific Information
Sys.unsetenv	Set or Unset Environment Variables
textConnection	Text Connections
textConnectionValue	Text Connections
timestamp	Print History of Evaluated Expressions
unlink	Remove Files and Directories
unserialize	Simple Serialization Interface
unz	Functions to Manipulate Connections
url	Functions to Manipulate Connections
withAutoprint	Read Expressions from a File or a Connection
write	Write Data to ASCII File
write.csv	Write Matrix of Data to a File
write.csv2	Write Matrix of Data to a File
write.dcf	Read and Write Data in DCF Format
write.table	Write Matrix of Data to a File
writeBin	Read and Write Binary Data
writeChar	Transfer Character Strings To and From Connections
writeLines	Read or Write Multiple Lines from or to a Connection
xzfile	Functions to Manipulate Connections

Interfaces to Other Languages

These are the available functions for interfaces to other languages. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
.C	Call a Fortran or C Routine
.Call	Manipulate R objects from C code
.External	Manipulate R objects from C code
.External2	Manipulate R objects from C code

Function name	Title description
.Fortran	Call a Fortran or C Routine
as.double	Double Precision Objects
as.double.difftime	Double Precision Objects
as.double.POSIXlt	Double Precision Objects
as.single	Single Precision Objects
as.single.default	Single Precision Objects
double	Double Precision Objects
dyn.load	Foreign Function Interface
dyn.unload	Foreign Function Interface
getNativeSymbolInfo	Obtain a Description of one or more Native Symbols
is.double	Double Precision Objects
is.loaded	Foreign Function Interface
is.single	Single Precision Objects
NativeSymbol	Obtain a Description of one or more Native Symbols
NativeSymbolInfo	Obtain a Description of one or more Native Symbols
RegisteredNativeSymbol	Obtain a Description of one or more Native Symbols
single	Single Precision Objects
system	Invoke a System Command
system2	Invoke a System Command (Windows only at this time)

Looping and Iteration

These are the available functions for looping and iteration. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
(	The Structure of Expressions
{	The Structure of Expressions
||	Control Flow
aggregate	Compute Summary Statistics of Subsets of Data
aggregate.data.frame	Compute Column-by-Column Summaries of Groups of Observations

Function name	Title description
aggregate.default	Compute Summary Statistics of Subsets of Data
aggregate.formula	Compute Summary Statistics of Subsets of Data
aggregate.ts	Compute Summary Statistics of Subsets of Data
apply	Apply a Function to Sections of an Array
break	Control Flow
by	Split a Data Frame and Apply a Function to the Parts
by.data.frame	Split a Data Frame and Apply a Function to the Parts
by.default	Split a Data Frame and Apply a Function to the Parts
colMeans	Row and Column Summaries
colSums	Row and Column Summaries
Control	Control Flow
dendrapply	Apply a Function to All Nodes of a Dendrogram
eapply	Apply a Function Over Values in an Environment
else	Control Flow
for	Control Flow
function	The Structure of Expressions
if	Control Flow
in	Control Flow
lapply	Apply a Function to Components of a List or Vector
next	Control Flow
rapply	Apply a Function Recursively
repeat	Control Flow
return	The Structure of Expressions
rowMeans	Row and Column Summaries
rowsum	Group Row Sums of a Matrix
rowsum.default	Group Row Sums of a Matrix
rowSums	Row and Column Summaries
sapply	Apply a Function to Components of a List or Vector

Function name	Title description
simplify2array	Apply a Function to Components of a List or Vector
sweep	Sweep Out Array Summaries
Syntax	The Structure of Expressions
tapply	Apply a Function to a Ragged Array
vapply	Apply a Function to Components of a List or Vector
while	Control Flow

Methods and Generic Functions

These are the available methods and generic functions. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
!	Logical Operators
[	Extract or Replace Parts of an Object
[.data.frame	Extract or Replace Parts of an Object
[.factor	Extract or Replace Parts of an Object
[[	Extract or Replace Parts of an Object
[[.data.frame	Extract or Replace Parts of an Object
[[.factor	Extract or Replace Parts of an Object
[[<-	Extract or Replace Parts of an Object
[[<-.data.frame	Extract or Replace Parts of an Object
[<-	Extract or Replace Parts of an Object
[<-.data.frame	Extract or Replace Parts of an Object
[<-.factor	Extract or Replace Parts of an Object
&	Logical Operators
|	Logical Operators
~	Model Formula Objects
$	Extract or Replace Parts of an Object
$<-	Extract or Replace Parts of an Object
anova.glm	Analysis of Deviance for Generalized Linear Model Fits

Function name	Title description
anova.glmlist	Analysis of Deviance for Generalized Linear Model Fits
anova.lm	Anova Table for Linear Model Objects
anova.lmlist	Apply anova to a lmlist Object
as	Generic Coercion Function
as.data.frame	Construct a Data Frame Object
as.formula	Define or Extract a Model Formula
data.frame	Construct a Data Frame Object
deviance	Deviance of a Fitted Model
deviance.default	Deviance of a Fitted Model
deviance.glm	Deviance of a Fitted Model
deviance.lm	Deviance of a Fitted Model
deviance.mlm	Deviance of a Fitted Model
deviance.nls	Deviance of a Fitted Model
DF2formula	Define or Extract a Model Formula
el	Extract or Replace Part of an Object
el<-	Extract or Replace Part of an Object
expand.model.frame	Add New Variables to a Model Frame
Extract	Extract or Replace Parts of an Object
family.object	Family of GLM Models
formula	Define or Extract a Model Formula
formula.call	Define or Extract a Model Formula
formula.character	Define or Extract a Model Formula
formula.data.frame	Define or Extract a Model Formula
formula.default	Define or Extract a Model Formula
formula.formula	Define or Extract a Model Formula
formula.lm	Define or Extract a Model Formula
formula.nls	Define or Extract a Model Formula
formula.object	Model Formula Objects

Function name	Title description
formula.terms	Define or Extract a Model Formula
get_all_vars	Construct or Extract a Model Frame
getS3method	Get An S3 Method
glm.object	Generalized Linear Model Object
hasArg	Check for Argument Names
is.data.frame	Construct a Data Frame Object
is.object	Test if an Object has a Class Attribute
isS3stdGeneric	Identify S3 Generic Function
lm.object	Linear Least Squares Model Object
loess.object	Loess Model Object
Logic	Logical Operators
methods	List Methods of Old-Style (SV3) Generic Functions
mlm.object	Linear Least Squares Model Object
model.frame	Construct or Extract a Model Frame
model.frame.aovlist	Construct or Extract a Model Frame
model.frame.default	Construct or Extract a Model Frame
model.frame.lm	Construct or Extract a Model Frame
na.action	Handle Missing Values in Objects
na.action.default	Handle Missing Values in Objects
na.exclude	Handle Missing Values in Objects
na.exclude.data.frame	Handle Missing Values in Objects
na.exclude.default	Handle Missing Values in Objects
na.fail	Handle Missing Values in Objects
na.fail.default	Handle Missing Values in Objects
na.omit	Handle Missing Values in Objects
na.omit.data.frame	Handle Missing Values in Objects
na.omit.default	Handle Missing Values in Objects
na.pass	Handle Missing Values in Objects

Function name	Title description
NextMethod	Methods Invoked from Functions
noquote	Remove Quotation Marks from a String
predict.glm	Predict Method for a Generalized Linear Model
predict.lm	Predict Method for a Linear Model
predict.mlm	Predict Method for a Linear Model
print.anova	Print an anova Object
print.family	Use print() on a family Object
print.formula	Use print() on a formula Object
print.lm	Use print() on an lm Object
print.summary.table	Summary of a table Object
registerS3method	Register S3 methods
registerS3methods	Register S3 methods
removeGeneric	Remove all the Methods for a Function
removeMethod	Remove all the Methods for a Function
removeMethods	Remove all the Methods for a Function
residuals.glm	Compute Residuals for glm Objects
rowsum.data.frame	Use rowsum() on a data frame object
setAs	Generic Coercion Function
Subscript	Extract or Replace Parts of an Object
Subscript.data.frame	Extract or Replace Parts of an Object
Subscript.factor	Extract or Replace Parts of an Object
summary	Summarize an Object - Generic Function
Summary	Summary Group Generic Function and Group Method
summary.data.frame	Summary of a Data Frame Object
Summary.data.frame	Summary Group Generic Function and Group Method
summary.Date	Summarize an Object - Generic Function
summary.factor	Use summary() on a factor Object
Summary.factor	Summary Group Generic Function and Group Method

Function name	Title description
summary.matrix	Summary of a Data Frame Object
summary.nls	Summary of an nls Model Object
summary.POSIXct	Summarize an Object - Generic Function
summary.POSIXlt	Summarize an Object - Generic Function
summary.table	Summary of a table Object
terms.object	Class of Objects for Terms in a Model
UseMethod	Methods Invoked from Functions
xor	Logical Operators

Miscellaneous

These are the miscellaneous functions. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
as.person	Person Names and Contact Information
as.personList	Person Names and Contact Information
intersect	Set Operations
is.element	Set Operations
person	Person Names and Contact Information
personList	Person Names and Contact Information
setdiff	Set Operations
setequal	Set Operations
toBibtex.person	Person Names and Contact Information
union	Set Operations

Printing

These are the printing functions. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
[.hexmode	Display Integers in Octal or Hexadecimal
[.octmode	Display Integers in Octal or Hexadecimal

Function name	Title description
[.roman	Display Integers As Roman Numerals
as.character.hexmode	Display Integers in Octal or Hexadecimal
as.character.octmode	Display Integers in Octal or Hexadecimal
as.character.roman	Display Integers As Roman Numerals
as.hexmode	Display Integers in Octal or Hexadecimal
as.octmode	Display Integers in Octal or Hexadecimal
as.roman	Display Integers As Roman Numerals
cat	Print the Arguments
dcf	Read and Write Data in DCF Format
deparse	Turn a Parsed Expression into Character Form
deparse1	Turn a Parsed Expression into Character Form
dget	Read or Write a Text Representation of an object
dput	Read or Write a Text Representation of an object
format	Formatted Character Data
formatC	Formatting Using C-style Formats
format.data.frame	Formatted Character Data
format.default	Formatted Character Data
format.factor	Formatted Character Data
format.hexmode	Display Integers in Octal or Hexadecimal
format.octmode	Display Integers in Octal or Hexadecimal
format.roman	Display Integers As Roman Numerals
formatDL	Format Description Lists
formatOL	Format Unordered and Ordered Lists for Printing
formatUL	Format Unordered and Ordered Lists for Printing
gettextf	Generate C-Style Formatted Output
head	Get the First or Last Part of an Object
hexmode	Display Integers in Octal or Hexadecimal
labels	Labels for Printing

Function name	Title description
labels.default	Labels for Printing
labels.dist	Labels for Printing
labels.glm	Labels for Printing
labels.lm	Labels for Printing
labels.terms	Labels for Printing
ls.str	List Objects and their Structure
lsf.str	List Objects and their Structure
noquote	Remove Quotation Marks from a String
octmode	Display Integers in Octal or Hexadecimal
prettyNum	Formatting Using C-style Formats
print	Print Data - Generic function
print.data.frame	Print a Data Frame Object
print.default	Print Data
print.factor	Use print on a factor Object
print.hexmode	Display Integers in Octal or Hexadecimal
print.listof	Print a listof Object
print.loadings	Print a Loadings Matrix
print.ls_str	List Objects and their Structure
print.octmode	Display Integers in Octal or Hexadecimal
print.prcomp	Print a Principal Components Object
print.princomp	Print a Principal Components Object
print.roman	Display Integers As Roman Numerals
print.summary.prcomp	Print a Principal Component Summary
print.summary.princomp	Print a Principal Component Summary
print.table	Print a table Object
print.ts	Print a Time Series
printCoefmat	Print Coefficient Matrices
read.dcf	Read and Write Data in DCF Format

Function name	Title description
roman	Display Integers As Roman Numerals
sprintf	Generate C-Style Formatted Output
str	Compactly Display the Structure of an Object
str.data.frame	Compactly Display the Structure of an Object
str.default	Compactly Display the Structure of an Object
strOptions	Compactly Display the Structure of an Object
summary.default	Default Summary Method
tail	Get the First or Last Part of an Object
write.csv	Write Matrix of Data to a File
write.csv2	Write Matrix of Data to a File
write.dcf	Read and Write Data in DCF Format
write.table	Write Matrix of Data to a File
zapsmall	Coerce Small Numbers to Zero for Printing

Programming functions

These are the programming functions. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
...elt	Manipulate Ellipsis Arguments
...length	Manipulate Ellipsis Arguments
...names	Manipulate Ellipsis Arguments
.doTrace	Trace Calls to Functions
.GlobalEnv	Environment Access
.handleSimpleError	Condition Handling and Recovery
.getNamespace	Get Namespace Environment Information
.Last.value	Value of Last Evaluated Expression
.makeMessage	Diagnostic Messages
.packageStartupMessage	Diagnostic Messages
.signalSimpleWarning	Condition Handling and Recovery

Function name	Title description
`(`	The Structure of Expressions
`[.terms`	Modify Terms Objects
`{`	The Structure of Expressions
`||`	Control Flow
`addTaskCallback`	Manage Top-Level Task Callbacks
`all.names`	Find All Names in an Expression
`all.vars`	Find All Names in an Expression
`as`	Generic Coercion Function
`as.call`	Calling Functions
`as.character.condition`	Condition Handling and Recovery
`as.character.error`	Condition Handling and Recovery
`as.double`	Double Precision Objects
`as.double.difftime`	Double Precision Objects
`as.double.POSIXlt`	Double Precision Objects
`as.expression`	Expression Objects
`as.function`	Function Objects
`as.integer`	Integer Objects
`as.name`	Names and Symbols
`as.symbol`	Names and Symbols
`assign`	Assign Object to Environment
`attachNamespace`	Loading and Unloading Namespaces
`baseenv`	Environment Access
`bquote`	Partial Substitution in Expressions
`break`	Control Flow
`browser`	Browse Interactively in a Function's Frame
`call`	Calling Functions
`charmatch`	Partial Matching of Character Strings
`chkDots`	Manipulate Ellipsis Arguments

Function name	Title description
computeRestarts	Condition Handling and Recovery
condition	Condition Handling and Recovery
conditionCall	Condition Handling and Recovery
conditionCall.condition	Condition Handling and Recovery
conditionMessage	Condition Handling and Recovery
conditionMessage.condition	Condition Handling and Recovery
conditions	Condition Handling and Recovery
Control	Control Flow
delete.response	Modify Terms Objects
deparse	Turn a Parsed Expression into Character Form
deparse1	Turn a Parsed Expression into Character Form
dir	List the Files in a Directory
do.call	Execute a Function Call
dontCheck	Return Argument Unaltered
double	Double Precision Objects
drop.terms	Modify Terms Objects
else	Control Flow
emptyenv	Environment Access
environment	Environment Access
environment<-	Environment Access
environmentName	Environment Access
errorCondition	Condition Handling and Recovery
eval	Evaluate an Expression
eval.parent	Evaluate an Expression
evalq	Evaluate an Expression
existsFunction	Get Function from Environment
expression	Expression Objects
Filter	Common Higher-Order Functions

Function name	Title description
`Find`	Common Higher-Order Functions
`findRestart`	Condition Handling and Recovery
`for`	Control Flow
`forceAndCall`	Call a Function with Some Arguments Forced
`formals`	Access and Manipulate the Formal Arguments
`formals<-`	Access and Manipulate the Formal Arguments
`function`	The Structure of Expressions
`geterrmessage`	Error and Warning Messages
`getExportedValue`	Get Namespace Environment Information
`getFunction`	Get Function from Environment
`getNamespace`	Get Namespace Environment Information
`getNamespaceExports`	Get Namespace Environment Information
`getNamespaceImports`	Get Namespace Environment Information
`getNamespaceName`	Get Namespace Environment Information
`getNamespaceUsers`	Get Namespace Environment Information
`getNamespaceVersion`	Get Namespace Environment Information
`getTaskCallbackNames`	Manage Top-Level Task Callbacks
`globalenv`	Environment Access
`history`	Print History of Evaluated Expressions
`identity`	Return Argument Unaltered
`if`	Control Flow
`in`	Control Flow
`integer`	Integer Objects
`interactive`	Test For Interactive Execution
`invisible`	Mark Function as Non-Printing
`invokeRestart`	Condition Handling and Recovery
`invokeRestartInteractively`	Condition Handling and Recovery
`is.atomic`	Test for Atomic or Recursive Objects

Function name	Title description
is.call	Calling Functions
is.double	Double Precision Objects
is.environment	Environment Access
is.expression	Expression Objects
is.function	Function Objects
is.integer	Integer Objects
is.language	Test for Atomic or Recursive Objects
is.name	Names and Symbols
is.R	Test If Running Under an R Compatible Engine
is.recursive	Test for Atomic or Recursive Objects
is.symbol	Names and Symbols
isBaseNamespace	Check if an Environment is a (base) Namespace Environment
isNamespace	Check if an Environment is a (base) Namespace Environment
isNamespaceLoaded	Get Namespace Environment Information
isRestart	Condition Handling and Recovery
languageEl	Extract or Replace Parts of Expressions
list.dirs	List the Files in a Directory
list.files	List the Files in a Directory
loadedNamespaces	Loading and Unloading Namespaces
loadhistory	Print History of Evaluated Expressions
loadNamespace	Loading and Unloading Namespaces
local	Evaluate an Expression
make.names	Make Character Strings into Legal Names
make.unique	Make Character Strings Unique
Map	Common Higher-Order Functions
match.arg	Verify an Argument Using Partial Matching
match.call	Argument Matching
match.fun	Function Verification for "Function Variables"

Function name	Title description
message	Diagnostic Messages
missing	Check for Missing Arguments
mode	Data Mode of the Values in a Vector
name	Names and Symbols
nargs	Number of Arguments to Function
Negate	Common Higher-Order Functions
new.env	Environment Access
next	Control Flow
NextMethod	Methods Invoked from Functions
on.exit	Exit Expression For a Function
packageStartupMessage	Diagnostic Messages
parent.env	Environment Access
parent.env<-	Environment Access
parent.frame	Get the System Evaluator State
parse	Parse Expressions
Position	Common Higher-Order Functions
print.condition	Condition Handling and Recovery
print.restart	Condition Handling and Recovery
R.version	Version Information
R.Version	Version Information
R.version.string	Version Information
readline	Read a Line from the Terminal
readLines	Read or Write Multiple Lines from or to a Connection
Recall	Recursive Call of the Current Function
Reduce	Common Higher-Order Functions
reformulate	Modify Terms Objects
reg.finalizer	Finalization of Objects
removeTaskCallback	Manage Top-Level Task Callbacks

Function name	Title description
repeat	Control Flow
requireNamespace	Loading and Unloading Namespaces
restartDescription	Condition Handling and Recovery
restartFormals	Condition Handling and Recovery
return	The Structure of Expressions
S3Part	S4 Classes that Contain S3 Classes
savehistory	Print History of Evaluated Expressions
setAs	Generic Coercion Function
setNamespaceInfo	Access Namespace Environment Information
signalCondition	Condition Handling and Recovery
simpleCondition	Condition Handling and Recovery
simpleError	Condition Handling and Recovery
simpleMessage	Condition Handling and Recovery
simpleWarning	Condition Handling and Recovery
sleep	Sleep for a Specified Period
slice.index	Slice Identification in an Array
stop	Error and Warning Messages
stopifnot	Stop if Not All True
storage.mode	Data Mode of the Values in a Vector
str2expression	Parse Expressions
str2lang	Parse Expressions
substitute	Substitute in an Expression
substituteDirect	Substitute in an Expression
suppressMessages	Diagnostic Messages
suppressPackageStartupMessages	Diagnostic Messages
suppressWarnings	Error and Warning Messages
switch	Evaluate One of Several Expressions
Syntax	The Structure of Expressions

Function name	Title description
sys.call	Get the System Evaluator State
sys.calls	Get the System Evaluator State
sys.frame	Get the System Evaluator State
sys.frames	Get the System Evaluator State
sys.function	Get the System Evaluator State
sys.nframe	Get the System Evaluator State
sys.on.exit	Get the System Evaluator State
sys.parent	Get the System Evaluator State
sys.parents	Get the System Evaluator State
Sys.sleep	Sleep for a Specified Period
sys.status	Get the System Evaluator State
system	Invoke a System Command
taskCallbackManager	Create an R-level Task Callback Manager
tempdir	Create Unique Names for Files
tempfile	Create Unique Names for Files
timestamp	Print History of Evaluated Expressions
topenv	Top-Level Environment
trace	Trace Calls to Functions
traceback	Print Call Stack After Error
try	Continue after errors
tryCatch	Condition Handling and Recovery
unimplementedStop	Functions for Handling Unimplemented Functions and Arguments
unimplementedWarning	Functions for Handling Unimplemented Functions and Arguments
unlink	Remove Files and Directories
unloadNamespace	Loading and Unloading Namespaces
untrace	Trace Calls to Functions
UseMethod	Methods Invoked from Functions
version	Version Information

Function name	Title description
warning	Error and Warning Messages
warningCondition	Condition Handling and Recovery
while	Control Flow
with	Evaluates an expression in a given context
with.default	Evaluates an expression in a given context
withCallingHandlers	Condition Handling and Recovery
within	Evaluates an expression in a given context
within.data.frame	Evaluates an expression in a given context
within.list	Evaluates an expression in a given context
withRestarts	Condition Handling and Recovery
writeLines	Read or Write Multiple Lines from or to a Connection

Session Environment

These are the session environment functions. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
.Platform	Platform Specific Variables
apropos	Find Objects by (Partial) Name
as.environment	Coerce to an Environment Object
eapply	Apply a Function Over Values in an Environment
file.path	Construct Path to File
gc	Garbage Collection
gcinfo	Garbage Collection
gctorture	Garbage Collection
gctorture2	Garbage Collection (stub function does nothing)
getOption	Set or Return Options
getwd	Get or Set Current Working Directory
ls	Retrive a List of Objects
memory.limit	Memory Limit and Size

Function name	Title description
memory.size	Memory Limit and Size
object.size	Internal Size of an Object
objects	Retrive a List of Objects
options	Set or Return Options
parent.frame	Get the System Evaluator State
path.expand	Expand ~ in File Paths
proc.time	Running Time of TIBCO Enterprise Runtime for R
q	Quit From TIBCO Enterprise Runtime for R
quit	Quit From TIBCO Enterprise Runtime for R
R.home	Paths to Files in the TIBCO Enterprise Runtime for R Installation
R.version	Version Information
R.Version	Version Information
R.version.string	Version Information
reg.finalizer	Finalization of Objects
remove	Remove Objects from a Specified Environment
rm	Remove Objects from a Specified Environment
setwd	Get or Set Current Working Directory
sys.call	Get the System Evaluator State
sys.calls	Get the System Evaluator State
sys.frame	Get the System Evaluator State
sys.frames	Get the System Evaluator State
sys.function	Get the System Evaluator State
Sys.getenv	Get Environment Variables
sys.nframe	Get the System Evaluator State
sys.on.exit	Get the System Evaluator State
sys.parent	Get the System Evaluator State
sys.parents	Get the System Evaluator State
Sys.setenv	Set or Unset Environment Variables

Function name	Title description
sys.status	Get the System Evaluator State
Sys.unsetenv	Set or Unset Environment Variables
system	Invoke a System Command
version	Version Information

Utilities

These are the utility functions. See each function's help topic in the TERR Language Reference for more information.

summarize and replace errors in loops overFunction name	Title description
$.package_version	Numeric Versions
*.difftime	Ops Group Method for Date/Time Objects
+.Date	Ops Group Method for Date/Time Objects
+.POSIXt	Ops Group Method for Date/Time Objects
-.Date	Ops Group Method for Date/Time Objects
-.POSIXt	Ops Group Method for Date/Time Objects
.POSIXct	Date-Time Classes
.POSIXlt	Date-Time Classes
.decode_numeric_version	Numeric Versions
.difftime	Time Intervals
.encode_numeric_version	Numeric Versions
.leap.seconds	Date-Time Classes
.make_numeric_version	Numeric Versions
.mapply	Apply a Function to Multiple List or Vector Arguments
.userHooksEnv	Functions to Get and Set Hooks for Load, Attach, Detach and Unload
/.difftime	Ops Group Method for Date/Time Objects
activeBindingFunction	Binding and Environment Adjustments
argsAnywhere	Search Packages, Namespaces, and S3 Methods
askYesNo	Ask a Yes/No Question
BATCH	Batch Execution of TIBCO Enterprise Runtime for R

summarize and replace errors in loops overFunction name	Title description
c.difftime	Time Intervals
className	Class Name and Package
Date	Date Class
Encoding	String Encodings of a Character Vector
Encoding<-	String Encodings of a Character Vector
getAnywhere	Search Packages, Namespaces, and S3 Methods
getParseData	Get Detailed Parse Information from Object
getParseText	Get Detailed Parse Information from Object
ISOdate	Construct Date-time from Broken-down Time
ISOdatetime	Construct Date-time from Broken-down Time
Math.Date	Math Group Method for Date/Time Objects
Math.POSIXt	Math Group Method for Date/Time Objects
Math.difftime	Math Group Method for Date/Time Objects
OlsonNames	Time Zones
Ops.Date	Ops Group Method for Date/Time Objects
Ops.POSIXt	Ops Group Method for Date/Time Objects
Ops.difftime	Ops Group Method for Date/Time Objects
Ops.numeric_version	Numeric Versions
POSIXct	Date-Time Classes
POSIXlt	Date-Time Classes
POSIXt	Date-Time Classes
Rdiff	Compare Printed Output From TERR and R
R_system_version	Numeric Versions
Summary.Date	Summary Group Method for Date/Time Objects
Summary.POSIXct	Summary Group Method for Date/Time Objects
Summary.POSIXlt	Summary Group Method for Date/Time Objects
Summary.difftime	Summary Group Method for Date/Time Objects
Summary.numeric_version	Numeric Versions

summarize and replace errors in loops overFunction name	Title description
Sys.Date	Get the Current Date, Time, or Time Zone
Sys.time	Get the Current Date, Time, or Time Zone
Sys.timezone	Get the Current Date, Time, or Time Zone
Vectorize	Apply a Function to Multiple List or Vector Arguments
[<-.Date	Date Class
[<-.POSIXct	Date-Time Classes
[<-.POSIXlt	Date-Time Classes
[.Date	Date Class
[.POSIXct	Date-Time Classes
[.POSIXlt	Date-Time Classes
[.difftime	Time Intervals
[.numeric_version	Numeric Versions
[[<-.numeric_version	Numeric Versions
[[.Date	Date Class
[[.POSIXct	Date-Time Classes
[[.numeric_version	Numeric Versions
as.Date	Date-Time Parsing
as.Date.POSIXct	Date-Time Parsing
as.Date.POSIXlt	Date-Time Parsing
as.Date.character	Date-Time Parsing
as.Date.date	Date-Time Parsing
as.Date.dates	Date-Time Parsing
as.Date.default	Date-Time Parsing
as.Date.factor	Date-Time Parsing
as.Date.numeric	Date-Time Parsing
as.POSIXct	Date-Time Parsing
as.POSIXct.Date	Date-Time Parsing
as.POSIXct.POSIXlt	Date-Time Parsing

summarize and replace errors in loops overFunction name	Title description
as.POSIXct.date	Date-Time Parsing
as.POSIXct.dates	Date-Time Parsing
as.POSIXct.default	Date-Time Parsing
as.POSIXct.numeric	Date-Time Parsing
as.POSIXlt	Date-Time Parsing
as.POSIXlt.Date	Date-Time Parsing
as.POSIXlt.POSIXct	Date-Time Parsing
as.POSIXlt.character	Date-Time Parsing
as.POSIXlt.date	Date-Time Parsing
as.POSIXlt.dates	Date-Time Parsing
as.POSIXlt.default	Date-Time Parsing
as.POSIXlt.factor	Date-Time Parsing
as.POSIXlt.numeric	Date-Time Parsing
as.character.Date	Date-Time Formatting
as.character.POSIXt	Date-Time Formatting
as.character.numeric_version	Numeric Versions
as.data.frame.numeric_version	Numeric Versions
as.difftime	Date-Time Parsing
as.numeric_version	Numeric Versions
as.package_version	Numeric Versions
available.packages	List Available Packages at CRAN-like Repositories
base64decode	Decode a String
base64encode	Encode a String
bindenv	Binding and Environment Adjustments
bindingIsActive	Binding and Environment Adjustments
bindingIsLocked	Binding and Environment Adjustments
c.numeric_version	Numeric Versions
capabilities	Report Capabilities of This Engine

summarize and replace errors in loops overFunction name	Title description
capture.output	Send output to a character string or file
check_tzones	Get the Current Date, Time, or Time Zone
compareVersion	Compare Two Package Version Numbers
contrib.url	Find Appropriate Paths in CRAN-like Repositories
date	Get the Current Date, Time, or Time Zone
demo	Demonstrations of Package Functionality
difftime	Time Intervals
download.packages	Download Packages from a Repository
duplicated.numeric_version	Numeric Versions
encodeString	Encode Character Vector for Printing
environmentIsLocked	Binding and Environment Adjustments
evalREX	Restricted TERR Execution Mode
example	Run Examples Section from the Online Help
file_test	Shell-style Tests of Files
format.Date	Date-Time Formatting
format.POSIXct	Date-Time Formatting
format.POSIXlt	Date-Time Formatting
format.difftime	Date-Time Formatting
getCRANmirrors	Comrehensive R Archive Network
getFromNamespace	Utility functions for Developing Namespaces
getHook	Functions to Get and Set Hooks for Load, Attach, Detach and Unload
getREX	Restricted TERR Execution Mode
getRversion	Numeric Versions
getTERREdition	Determine the Edition of the TIBCO Enterprise Runtime for R Engine
iconv	Convert Character Vector between Encodings
iconvlist	Convert Character Vector between Encodings
infoRDS	RDS File Details
install.packages	Install Packages from CRAN-like Repositories

summarize and replace errors in loops overFunction name	Title description
installed.packages	Find Installed Packages
IntelMKLVersion	Find Library Version
is.numeric_version	Numeric Versions
is.package_version	Numeric Versions
isSymmetric	Test if an Object is Symmetric
isSymmetric.matrix	Test if an Object is Symmetric
l10n_info	Localization Information
localeToCharset	Localization Information
lockBinding	Binding and Environment Adjustments
lockEnvironment	Binding and Environment Adjustments
ls.str	List Objects and their Structure
lsf.str	List Objects and their Structure
makeActiveBinding	Binding and Environment Adjustments
mapply	Apply a Function to Multiple List or Vector Arguments
modifyList	Recursively Modify Elements of a List
new.packages	Compare Installed Packages with CRAN-like Repositories
noquote	Remove Quotation Marks from a String
normalizePath	Express File Paths in Canonical Form
numeric_version	Numeric Versions
old.packages	Compare Installed Packages with CRAN-like Repositories
package_native_routine_registration_skeleton	Register Native Routines
package.skeleton	Create a Skeleton for a New Source Package
packageEvent	Functions to Get and Set Hooks for Load, Attach, Detach and Unload
package_dependencies	Dependency Hierarchy
package_version	Numeric Versions
pos.to.env	Convert Positions in the Search Path to Environments
print.Date	Date Class

summarize and replace errors in loops overFunction name	Title description
print.POSIXct	Date-Time Classes
print.POSIXlt	Date-Time Classes
print.difftime	Time Intervals
print.ls_str	List Objects and their Structure
print.numeric_version	Numeric Versions
pskill	Kill a Process
psnice	Kill a Process
read.delim	Create a Data Frame by Reading a Table
read.delim2	Create a Data Frame by Reading a Table
regmatches	Extract or Replace Matched Substrings
regmatches<-	Extract or Replace Matched Substrings
relevel	Reorder Levels of Factor
relevel.default	Reorder Levels of Factor
relevel.factor	Reorder Levels of Factor
relevel.ordered	Reorder Levels of Factor
remove.packages	Remove Installed Packages
reorder	Reorder Levels of a Factor
reorder.default	Reorder Levels of a Factor
rep.numeric_version	Numeric Versions
round.Date	Math Group Method for Date/Time Objects
round.POSIXt	Math Group Method for Date/Time Objects
select.list	Select Items from a List
setHook	Functions to Get and Set Hooks for Load, Attach, Detach and Unload
setInternet2	Enable or Disable the Use of Internet Explorer Settings for Internet Access
setRepositories	Select Package Repositories
shQuote	Quote Strings for Use in OS Shells
shortPathName	Convert File Name to DOS 8.3 Format
SIG*	Kill a Process

summarize and replace errors in loops overFunction name	Title description
stopIfREX	Restricted TERR Execution Mode
str	Compactly Display the Structure of an Object
str.data.frame	Compactly Display the Structure of an Object
str.default	Compactly Display the Structure of an Object
strOptions	Compactly Display the Structure of an Object
strftime	Date-Time Formatting
strptime	Date-Time Parsing
suppressForeignCheck	Package Building from Source
symnum	Symbolic Number Coding
system.time	CPU Time Used
trunc.Date	Math Group Method for Date/Time Objects
trunc.POSIXt	Math Group Method for Date/Time Objects
unique.numeric_version	Numeric Versions
units	Units for Time Intervals
units<-	Units for Time Intervals
units<-.difftime	Units for Time Intervals
units.difftime	Units for Time Intervals
unix.time	CPU Time Used
unlockBinding	Binding and Environment Adjustments
update.packages	Compare Installed Packages with CRAN-like Repositories
uploadSBDF	Upload an SBDF File
warnErrList	Summarize and Replace Errors in Loops over tryCatch
xtfrm.numeric_version	Numeric Versions

Statistics

These statistics functions are available in TERR. For help with a function, see its listing in the TERR Language Reference.

Clustering

These are the available functions for clustering. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
acf	Auto- and Cross- Covariance or Correlation Estimation
aggregate	Compute Summary Statistics of Subsets of Data
aggregate.default	Compute Summary Statistics of Subsets of Data
aggregate.formula	Compute Summary Statistics of Subsets of Data
aggregate.ts	Compute Summary Statistics of Subsets of Data
ar	Fit Autoregressive Models to Time Series
ar.yw	Fit Autoregressive Models to Time Series
arima	ARIMA modeling of Time Series
arima.sim	Simulate a Univariate ARIMA Series
as.ts	Time Series Objects
bartlett.test	Bartlett Test of Homogeneity of Variances
Box.test	Box-Pierce and Ljung-Box Tests
cbind.ts	Union and Intersection of Time Series
ccf	Auto- and Cross- Covariance or Correlation Estimation
coef.Arima2	ARIMA modeling of Time Series
cycle	Create Time Vector or Index of Frequency
cycle.default	Create Time Vector or Index of Frequency
decompose	Classical Seasonal Decomposition by Moving Averages
diff	Create an Object of Differences
diff.Date	Create an Object of Differences
diff.default	Create an Object of Differences
diff.POSIXt	Create an Object of Differences
diff.ts	Create an Object of Differences
diffinv	Discrete Integration: inverse of diff
diffinv.default	Discrete Integration: inverse of diff
diffinv.ts	Discrete Integration: inverse of diff

Function name	Title description
fft	Fast Fourier Transform
filter	Apply a Filter to a Time Series
HoltWinters	Holt-Winters Filtering
is.mts	Time Series Objects
is.ts	Time Series Objects
lag	Create a Lagged Time Series
mvfft	Fast Fourier Transform
na.contiguous	Find Longest Contiguous Stretch of non-NAs
na.contiguous.default	Find Longest Contiguous Stretch of non-NAs
na.contiguous.ts	Find Longest Contiguous Stretch of non-NAs
pacf	Auto- and Cross- Covariance or Correlation Estimation
predict.ar	Fit Autoregressive Models to Time Series
predict.HoltWinters	Holt-Winters Filtering
print.ar	Fit Autoregressive Models to Time Series
print.Arima2	ARIMA modeling of Time Series
print.HoltWinters	Holt-Winters Filtering
print.ts	Print a Time Series
residuals.HoltWinters	Holt-Winters Filtering
time	Create Time Vector or Index of Frequency
time.default	Create Time Vector or Index of Frequency
ts	Time Series Objects
ts.intersect	Union and Intersection of Time Series
ts.union	Union and Intersection of Time Series
vcov.Arima2	ARIMA modeling of Time Series
window	Window a Time Series
window.default	Window a Time Series
window.ts	Window a Time Series
window<-	Window a Time Series

Function name	Title description
window<-.ts	Window a Time Series

Computations Related to Plotting (Statistics)

These are the available functions for computations related to plotting. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
is.stepfun	Computing a Step Function

Curve (and Surface) Smoothing

These are the available functions for curve (and surface) smoothing. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
density	Kernel Estimate of Probability Density Function
density.default	Kernel Estimate of Probability Density Function
lowess	Scatter Plot Smoothing
predict.smooth.spline	Smoothing Spline at New Data
smooth	Nonlinear Smoothing Using Running Medians
smooth.spline	Fit a Smoothing Spline
supsmu	Scatter Plot Smoothing Using Super Smoother

Designed Experiments

These are the available functions for designed experiments. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
anova	Anova Tables
aov	Fit an Analysis of Variance Model
C	Assign Contrasts to a Factor
contr.helmert	Contrast or Dummy Variable Matrix
contr.poly	Contrast or Dummy Variable Matrix
contr.SAS	Contrast or Dummy Variable Matrix
contr.sum	Contrast or Dummy Variable Matrix

Function name	Title description
contr.treatment	Contrast or Dummy Variable Matrix
contrasts	Contrasts Attribute
contrasts<-	Contrasts Attribute
eff.aovlist	Compute Efficiency Factors for aovlist Model Terms
friedman.test	Friedman Rank Sum Test
friedman.test.default	Friedman Rank Sum Test
friedman.test.formula	Friedman Rank Sum Test
interaction	Compute the Interaction of Several Factors
kruskal.test	Kruskal-Wallis Rank Sum Test
kruskal.test.default	Kruskal-Wallis Rank Sum Test
kruskal.test.formula	Kruskal-Wallis Rank Sum Test
loess	Fit a Local Regression Model
model.tables	Compute Tables of Estimates for Model Object
model.tables.aov	Compute Tables of Estimates for Model Object
model.tables.aovlist	Compute Tables of Estimates for Model Object
print.aov	Fit an Analysis of Variance Model
print.aovlist	Fit an Analysis of Variance Model
print.mtable	Compute Tables of Estimates for Model Object
print.summary.aov	Summary of an Analysis of Variance Object
print.summary.aovlist	Summary of an Analysis of Variance Object
print.tables_aov	Compute Tables of Estimates for Model Object
proj	Projection Matrix
proj.aov	Projection Matrix
proj.aovlist	Projection Matrix
proj.default	Projection Matrix
proj.lm	Projection Matrix
replications	Number of Replications of Terms
se.aov	Standard Error of AOV Objects

Function name	Title description
se.aovlist	Standard Error of AOV Objects
se.contrast.aov	Standard Errors for Contrasts between Means
se.contrast.aovlist	Standard Errors for Contrasts between Means
summary.aov	Summary of an Analysis of Variance Object
summary.aovlist	Summary of an Analysis of Variance Object

Loess Objects

These are the available functions for loess objects. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
expand.grid	Create a Data Frame from All Combinations of Factors
loess	Fit a Local Regression Model
loess.control	Computational Options for Loess Fitting
loess.object	Loess Model Object
loess.smooth	Smooth Loess Curve
predict.loess	Evaluation of Local Regression Surfaces

Multivariate Techniques

These are the available functions for multivariate techniques. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
[[.dendrogram	General Tree Structures
as.dendrogram	General Tree Structures
as.dendrogram.dendrogram	General Tree Structures
as.dendrogram.hclust	General Tree Structures
as.dist	Distance Matrix Calculation
as.dist.default	Distance Matrix Calculation
as.hclust	Converts Objects to Class hclust
as.hclust.default	Converts Objects to Class hclust
as.hclust.dendrogram	Converts Objects to Class hclust

Function name	Title description
as.hclust.twins	Converts Objects to Class hclust
as.matrix.dist	Distance Matrix Calculation
bartlett.test	Bartlett Test of Homogeneity of Variances
cancor	Canonical Correlation Analysis
cmdscale	Classical Metric Multi-Dimensional Scaling
cophenetic	Cophenetic Distances for a Hierarchical Clustering
cophenetic.default	Cophenetic Distances for a Hierarchical Clustering
cophenetic.dendrogram	Cophenetic Distances for a Hierarchical Clustering
cor	Correlation, Variance, and Covariance (Matrices)
cov	Correlation, Variance, and Covariance (Matrices)
cov.wt	Weighted Covariance Estimation
cov2cor	Correlation, Variance, and Covariance (Matrices)
cut.dendrogram	General Tree Structures
cutree	Create Groups from Hierarchical Clustering
dendrogram	General Tree Structures
dist	Distance Matrix Calculation
estVar	SSD Matrix and Estimated Variance Matrix in Multivariate Models
estVar.mlm	SSD Matrix and Estimated Variance Matrix in Multivariate Models
estVar.SSD	SSD Matrix and Estimated Variance Matrix in Multivariate Models
factanal	Estimate a Factor Analysis Model
fft	Fast Fourier Transform
fitted.kmeans	K-Means Clustering
format.dist	Distance Matrix Calculation
hclust	Hierarchical Clustering
is.leaf	General Tree Structures
kmeans	K-Means Clustering
labels.dist	Distance Matrix Calculation
loadings	Extract Loadings from an Object

Function name	Title description
loglin	Contingency Table Analysis
mahalanobis	Mahalanobis Distance
mvfft	Fast Fourier Transform
plot.dendrogram	General Tree Structures
prcomp	Principal Components Analysis
prcomp.default	Principal Components Analysis
prcomp.formula	Principal Components Analysis
predict.prcomp	Principal Component Scores
predict.princomp	Principal Component Scores
princomp	Principal Components Analysis
princomp.default	Principal Components Analysis
princomp.formula	Principal Components Analysis
print.dendrogram	General Tree Structures
print.dist	Distance Matrix Calculation
print.factanal	Estimate a Factor Analysis Model
print.hclust	Hierarchical Clustering
print.kmeans	K-Means Clustering
print.loadings	Print a Loadings Matrix
print.prcomp	Print a Principal Components Object
print.princomp	Print a Principal Components Object
print.summary.prcomp	Print a Principal Component Summary
print.summary.princomp	Print a Principal Component Summary
SSD	SSD Matrix and Estimated Variance Matrix in Multivariate Models
SSD.mlm	SSD Matrix and Estimated Variance Matrix in Multivariate Models
str.dendrogram	General Tree Structures
summary.prcomp	Summary of a Principal Components Object
summary.princomp	Summary of a Principal Components Object
var	Correlation, Variance, and Covariance (Matrices)

Non-Linear Regression

These are the available functions for non-linear regression. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
`integrate`	Integral of a Real-valued Function
`nlminb`	Nonlinear Minimization subject to Box Constraints
`nls.control`	Control the Iteration in nls()
`optim`	General-purpose Optimization
`optimHess`	Numerically Estimate Hessian Matrix
`optimise`	Univariate Optimization of a Function
`optimize`	Univariate Optimization of a Function
`predict.nls`	Predicting from Nonlinear Least Squares Fits
`uniroot`	Find a Root of a Univariate Function

Nonparametric Statistics

These are the available functions for nonparametric statistics. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
`cor.test`	Test for Correlation Between Paired Samples
`cor.test.default`	Test for Correlation Between Paired Samples
`cor.test.formula`	Test for Correlation Between Paired Samples
`dwilcox`	The Distribution of the Wilcoxon Rank Sum Statistic
`friedman.test`	Friedman Rank Sum Test
`friedman.test.default`	Friedman Rank Sum Test
`friedman.test.formula`	Friedman Rank Sum Test
`kruskal.test`	Kruskal-Wallis Rank Sum Test
`kruskal.test.default`	Kruskal-Wallis Rank Sum Test
`kruskal.test.formula`	Kruskal-Wallis Rank Sum Test
`predict.smooth.spline`	Smoothing Spline at New Data
`pwilcox`	The Distribution of the Wilcoxon Rank Sum Statistic
`qwilcox`	The Distribution of the Wilcoxon Rank Sum Statistic

Function name	Title description
rwilcox	The Distribution of the Wilcoxon Rank Sum Statistic
wilcox.test	Wilcoxon Rank Sum and Signed Rank Tests
wilcox.test.default	Wilcoxon Rank Sum and Signed Rank Tests
wilcox.test.formula	Wilcoxon Rank Sum and Signed Rank Tests
Wilcoxon	The Distribution of the Wilcoxon Rank Sum Statistic

Probability Distributions and Random Numbers

These are the available functions for probability distributions and random numbers. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
.Random.seed	Control Random Number Generator
Beta	The Beta Distribution
Binomial	The Binomial Distribution
Cauchy	The Cauchy Distribution
Chisquare	The Chi-square Distribution
dbeta	The Beta Distribution
dbinom	The Binomial Distribution
dcauchy	The Cauchy Distribution
dchisq	The Chi-square Distribution
density	Kernel Estimate of Probability Density Function
density.default	Kernel Estimate of Probability Density Function
dexp	The Exponential Distribution
df	The F Distribution
dgamma	The Gamma Distribution
dgeom	The Geometric Distribution
dhyper	The Hypergeometric Distribution
dlnorm	The Lognormal Distribution
dlogis	The Logistic Distribution
dmultinom	The Multinomial Distribution

Function name	Title description
dnbinom	The Negative Binomial Distribution
dnorm	The Normal Distribution
dpois	The Poisson Distribution
dsignrank	Distribution of the Wilcoxon Signed Rank Statistic
dt	The Student's t Distribution
dunif	The Uniform Distribution
dweibull	The Weibull Distribution
dwilcox	The Distribution of the Wilcoxon Rank Sum Statistic
Exponential	The Exponential Distribution
FDist	The F Distribution
fivenum	Tukey Five-Number Summaries
GammaDist	The Gamma Distribution
Geometric	The Geometric Distribution
Hypergeometric	The Hypergeometric Distribution
Logistic	The Logistic Distribution
Lognormal	The Lognormal Distribution
NegBinomial	The Negative Binomial Distribution
Normal	The Normal Distribution
pbeta	The Beta Distribution
pbinom	The Binomial Distribution
pcauchy	The Cauchy Distribution
pchisq	The Chi-square Distribution
pexp	The Exponential Distribution
pf	The F Distribution
pgamma	The Gamma Distribution
pgeom	The Geometric Distribution
phyper	The Hypergeometric Distribution
plnorm	The Lognormal Distribution

Function name	Title description
plogis	The Logistic Distribution
pnbinom	The Negative Binomial Distribution
pnorm	The Normal Distribution
Poisson	The Poisson Distribution
ppoints	Plotting Points for Quantile-Quantile Plots
ppois	The Poisson Distribution
psignrank	Distribution of the Wilcoxon Signed Rank Statistic
pt	The Student's t Distribution
ptukey	The Studentized Range Distribution
punif	The Uniform Distribution
pweibull	The Weibull Distribution
pwilcox	The Distribution of the Wilcoxon Rank Sum Statistic
qbeta	The Beta Distribution
qbinom	The Binomial Distribution
qcauchy	The Cauchy Distribution
qchisq	The Chi-square Distribution
qexp	The Exponential Distribution
qf	The F Distribution
qgamma	The Gamma Distribution
qgeom	The Geometric Distribution
qhyper	The Hypergeometric Distribution
qlnorm	The Lognormal Distribution
qlogis	The Logistic Distribution
qnbinom	The Negative Binomial Distribution
qnorm	The Normal Distribution
qpois	The Poisson Distribution
qqnorm	Normal Quantile-Quantile Plots
qqnorm.default	Normal Quantile-Quantile Plots

Function name	Title description
qsignrank	Distribution of the Wilcoxon Signed Rank Statistic
qt	The Student's t Distribution
qtukey	The Studentized Range Distribution
quantile	Empirical Quantiles
qunif	The Uniform Distribution
qweibull	The Weibull Distribution
qwilcox	The Distribution of the Wilcoxon Rank Sum Statistic
r2dtable	Random Two-way Tables with Given Marginals
rbeta	The Beta Distribution
rbinom	The Binomial Distribution
rcauchy	The Cauchy Distribution
rchisq	The Chi-square Distribution
rexp	The Exponential Distribution
rf	The F Distribution
rgamma	The Gamma Distribution
rgeom	The Geometric Distribution
rhyper	The Hypergeometric Distribution
rlnorm	The Lognormal Distribution
rlogis	The Logistic Distribution
rmultinom	The Multinomial Distribution
rnbinom	The Negative Binomial Distribution
RNG	Control Random Number Generator
RNGkind	Control Random Number Generator
RNGversion	Control Random Number Generator
rnorm	The Normal Distribution
rpois	The Poisson Distribution
rsignrank	Distribution of the Wilcoxon Signed Rank Statistic
rt	The Student's t Distribution

Function name	Title description
runif	The Uniform Distribution
rweibull	The Weibull Distribution
rwilcox	The Distribution of the Wilcoxon Rank Sum Statistic
sample	Generate Random Samples or Permutations of Data
sample.int	Generate Random Samples or Permutations of Data
set.seed	Control Random Number Generator
SignRank	Distribution of the Wilcoxon Signed Rank Statistic
TDist	The Student's t Distribution
Tukey	The Studentized Range Distribution
Uniform	The Uniform Distribution
Weibull	The Weibull Distribution
Wilcoxon	The Distribution of the Wilcoxon Rank Sum Statistic

Regression

These are the available functions for regression. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
.lm.fit	General fitting for linear (regression) models
.kappa_tri	Compute the Exact or Estimated Condition Number
add1.glm	Add a Single Term to a Linear Model
add1.lm	Add a Single Term to a Linear Model
anova.glm	Analysis of Deviance for Generalized Linear Model Fits
anova.glmlist	Analysis of Deviance for Generalized Linear Model Fits
cooks.distance	Regression Deletion Diagnostics
cooks.distance.glm	Regression Deletion Diagnostics
cooks.distance.lm	Regression Deletion Diagnostics
covratio	Regression Deletion Diagnostics
dfbeta	Regression Deletion Diagnostics
dfbeta.lm	Regression Deletion Diagnostics

Function name	Title description
dfbetas	Regression Deletion Diagnostics
dfbetas.lm	Regression Deletion Diagnostics
dffits	Regression Deletion Diagnostics
dummy.coef	Extract Original Coefficients from a Linear Model
dummy.coef.aovlist	Extract Original Coefficients from a Linear Model
dummy.coef.lm	Extract Original Coefficients from a Linear Model
effects	Single Degree-of-freedom Effects from a Fitted Model
effects.glm	Single Degree-of-freedom Effects from a Fitted Model
effects.lm	Single Degree-of-freedom Effects from a Fitted Model
glm	Fit a Generalized Linear Model
glm.control	Set Control Parameters for Generalized Linear Model
glm.fit	Fit a GLM without Computing the Model Matrix
glm.object	Generalized Linear Model Object
hat	Hat Diagonal Regression Diagnostic
hatvalues	Regression Deletion Diagnostics
hatvalues.lm	Regression Deletion Diagnostics
influence	Regression Diagnostics
influence.glm	Regression Diagnostics
influence.lm	Regression Diagnostics
influence.measures	Regression Deletion Diagnostics
isoreg	Isotonic / Monotone Regression
kappa	Compute the Exact or Estimated Condition Number
kappa.default	Compute the Exact or Estimated Condition Number
kappa.lm	Compute the Exact or Estimated Condition Number
kappa.qr	Compute the Exact or Estimated Condition Number
ksmooth	Scatter Plot Smoothing
lm	Fit Linear Regression Model
lm.fit	General fitting for linear (regression) models

Function name	Title description
lm.influence	Regression Diagnostics
lm.object	Linear Least Squares Model Object
lm.wfit	General fitting for linear (regression) models
lowess	Scatter Plot Smoothing
lsfit	Linear Least-Squares Fit
mlm.object	Linear Least Squares Model Object
poly	Compute Orthogonal Polynomials
polym	Compute Orthogonal Polynomials
predict.nls	Predicting from Nonlinear Least Squares Fits
predict.poly	Compute Orthogonal Polynomials
print.dummy_coef	Extract Original Coefficients from a Linear Model
print.dummy_coef_list	Extract Original Coefficients from a Linear Model
print.infl	Regression Deletion Diagnostics
print.summary.lm	Summary Method for Linear Models
print.summary.mlm	Summary Method for Linear Models
proj	Projection Matrix
proj.aov	Projection Matrix
proj.aovlist	Projection Matrix
proj.default	Projection Matrix
proj.lm	Projection Matrix
rstandard	Regression Deletion Diagnostics
rstandard.glm	Regression Deletion Diagnostics
rstandard.lm	Regression Deletion Diagnostics
rstudent	Regression Deletion Diagnostics
rstudent.glm	Regression Deletion Diagnostics
rstudent.lm	Regression Deletion Diagnostics
stat.anova	Add Statistics Columns to an Anova Table
summary.glm	Summary Method for Fitted Generalized Linear Models

Function name	Title description
summary.infl	Regression Deletion Diagnostics
summary.lm	Summary Method for Linear Models
summary.mlm	Summary Method for Linear Models

Regression and Classification Trees

These are the available functions for regression and classification trees . See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
[[.dendrogram	General Tree Structures
as.dendrogram	General Tree Structures
as.dendrogram.dendrogram	General Tree Structures
as.dendrogram.hclust	General Tree Structures
cut.dendrogram	General Tree Structures
dendrogram	General Tree Structures
is.leaf	General Tree Structures
plot.dendrogram	General Tree Structures
print.dendrogram	General Tree Structures
str.dendrogram	General Tree Structures

Robust and Resistant Techniques

These are the available functions for robust and resistant techniques. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
fivenum	Tukey Five-Number Summaries
ksmooth	Scatter Plot Smoothing
line	Robust Line Fitting
lowess	Scatter Plot Smoothing
mad	Robust Estimates of Scale
mean	Mean Value (Arithmetic Average)
mean.data.frame	Mean Value (Arithmetic Average)

Function name	Title description
mean.Date	Mean Value (Arithmetic Average)
mean.default	Mean Value (Arithmetic Average)
mean.difftime	Mean Value (Arithmetic Average)
mean.POSIXct	Mean Value (Arithmetic Average)
mean.POSIXlt	Mean Value (Arithmetic Average)
median	Median
median.default	Median
medpolish	Median Polish
smooth	Nonlinear Smoothing Using Running Medians

Simple Univariate Statistics

These are the available functions for simple univariate statistics. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
ave	Group Averages Over Level Combinations of Factors
cor	Correlation, Variance, and Covariance (Matrices)
cov	Correlation, Variance, and Covariance (Matrices)
cov2cor	Correlation, Variance, and Covariance (Matrices)
fivenum	Tukey Five-Number Summaries
sd	Compute Standard Deviation
var	Correlation, Variance, and Covariance (Matrices)
weighted.mean	Compute Weighted Mean
weighted.mean.Date	Compute Weighted Mean
weighted.mean.default	Compute Weighted Mean
weighted.mean.difftime	Compute Weighted Mean
weighted.mean.POSIXct	Compute Weighted Mean
weighted.mean.POSIXlt	Compute Weighted Mean

Statistical Inference

These are the available functions for statistical inference. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
binom.test	Exact Binomial Test
chisq.test	Pearson's Chi-square Test for Count Data
cor.test	Test for Correlation Between Paired Samples
cor.test.default	Test for Correlation Between Paired Samples
cor.test.formula	Test for Correlation Between Paired Samples
fisher.test	Fisher's Exact Test for Count Data
friedman.test	Friedman Rank Sum Test
friedman.test.default	Friedman Rank Sum Test
friedman.test.formula	Friedman Rank Sum Test
kruskal.test	Kruskal-Wallis Rank Sum Test
kruskal.test.default	Kruskal-Wallis Rank Sum Test
kruskal.test.formula	Kruskal-Wallis Rank Sum Test
ks.test	Kolmogorov-Smirnov Tests
mantelhaen.test	Mantel-Haenszel Chi-Square Test for Count Data
mcnemar.test	McNemar's Chi-Square Test for Count Data
oneway.test	Test for Equal Means in a One-Way Layout
p.adjust	Adjust P-values for Multiple Comparisons
p.adjust.methods	Adjust P-values for Multiple Comparisons
prop.test	Proportions Tests
shapiro.test	Shapiro-Wilk Test for Normality
t.test	Student's t-test
t.test.default	Student's t-test
t.test.formula	Student's t-test
var.test	F Test to Compare Two Variances
var.test.default	F Test to Compare Two Variances
var.test.formula	F Test to Compare Two Variances
wilcox.test	Wilcoxon Rank Sum and Signed Rank Tests
wilcox.test.default	Wilcoxon Rank Sum and Signed Rank Tests

Function name	Title description
wilcox.test.formula	Wilcoxon Rank Sum and Signed Rank Tests

Statistical Models

These are the available functions for statistical models. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
add.scope	Resolve Scopes for Formulas
add1	Compute Models by Adding One Term
add1.default	Compute Models by Adding One Term
AIC	Akaike's Information Criterion
anova	Anova Tables
anova.glm	Analysis of Deviance for Generalized Linear Model Fits
anova.glmlist	Analysis of Deviance for Generalized Linear Model Fits
aov	Fit an Analysis of Variance Model
ar	Fit Autoregressive Models to Time Series
ar.yw	Fit Autoregressive Models to Time Series
as.data.frame.array	Construct a Data Frame Object from an S object
as.data.frame.character	Construct a Data Frame Object from an S object
as.data.frame.complex	Construct a Data Frame Object from an S object
as.data.frame.data.frame	Construct a Data Frame Object from an S object
as.data.frame.Date	Construct a Data Frame Object from an S object
as.data.frame.default	Construct a Data Frame Object from an S object
as.data.frame.difftime	Construct a Data Frame Object from an S object
as.data.frame.factor	Construct a Data Frame Object from an S object
as.data.frame.integer	Construct a Data Frame Object from an S object
as.data.frame.list	Construct a Data Frame Object from an S object
as.data.frame.logical	Construct a Data Frame Object from an S object
as.data.frame.matrix	Construct a Data Frame Object from an S object
as.data.frame.model.matrix	Construct a Data Frame Object from an S object

Function name	Title description
as.data.frame.numeric	Construct a Data Frame Object from an S object
as.data.frame.ordered	Construct a Data Frame Object from an S object
as.data.frame.POSIXct	Construct a Data Frame Object from an S object
as.data.frame.POSIXlt	Construct a Data Frame Object from an S object
as.data.frame.raw	Construct a Data Frame Object from an S object
as.data.frame.table	Construct a Data Frame Object from an S object
as.data.frame.ts	Construct a Data Frame Object from an S object
as.data.frame.vector	Construct a Data Frame Object from an S object
as.terms	Create or Extract a Terms Object
as.vector.factor	Coerce a Factor Object into a Vector of a Given Mode
AsIs	Inhibit Interpretation/Conversion of Objects
asOneSidedFormula	Convert to One-Sided Formula
BIC	Akaike's Information Criterion
cmdscale	Classical Metric Multi-Dimensional Scaling
coef	Extract Information from a Model
coef.default	Extract Information from a Model
coef.listof	Extract Information from a Model
coefficients	Extract Information from a Model
confint	Confidence Intervals for Model Parameters
confint.default	Confidence Intervals for Model Parameters
drop.scope	Resolve Scopes for Formulas
drop1	Investigate models by dropping single terms
drop1.default	Investigate models by dropping single terms
drop1.glm	Investigate models by dropping single terms
drop1.lm	Investigate models by dropping single terms
dummy.coef	Extract Original Coefficients from a Linear Model
dummy.coef.aovlist	Extract Original Coefficients from a Linear Model
dummy.coef.lm	Extract Original Coefficients from a Linear Model

Function name	Title description
effects	Single Degree-of-freedom Effects from a Fitted Model
effects.glm	Single Degree-of-freedom Effects from a Fitted Model
effects.lm	Single Degree-of-freedom Effects from a Fitted Model
estVar	SSD Matrix and Estimated Variance Matrix in Multivariate Models
estVar.mlm	SSD Matrix and Estimated Variance Matrix in Multivariate Models
estVar.SSD	SSD Matrix and Estimated Variance Matrix in Multivariate Models
expand.model.frame	Add New Variables to a Model Frame
extractAIC	Extract AIC from a Fitted Model
extractAIC.glm	Extract AIC from a Fitted Model
extractAIC.lm	Extract AIC from a Fitted Model
factanal	Estimate a Factor Analysis Model
factor.scope	Resolve Scopes for Formulas
fitted	Extract Information from a Model
fitted.default	Extract Information from a Model
fitted.values	Extract Information from a Model
get_all_vars	Construct or Extract a Model Frame
getCall	Update a Fitted Model Object
glm	Fit a Generalized Linear Model
I	Inhibit Interpretation/Conversion of Objects
is.empty.model	Does a Model Contain any Predictors
lm	Fit Linear Regression Model
logLik	Extract Log-Likelihood
logLik.lm	Extract Log-Likelihood
logLik.nls	Extract Log-Likelihood
loglin	Contingency Table Analysis
lsfit	Linear Least-Squares Fit
make.link	Create a Link for GLM Families
makepredictcall	Utility Function for Safe Prediction

Function name	Title description
makepredictcall.default	Utility Function for Safe Prediction
makepredictcall.matrix	Utility Function for Safe Prediction
makepredictcall.poly	Utility Function for Safe Prediction
model.extract	Extract Special Information from Model Frame
model.frame	Construct or Extract a Model Frame
model.frame.aovlist	Construct or Extract a Model Frame
model.frame.default	Construct or Extract a Model Frame
model.frame.lm	Construct or Extract a Model Frame
model.matrix	Matrix of Predictors
model.matrix.default	Matrix of Predictors
model.matrix.object	Matrix of Predictors
model.offset	Extract Special Information from Model Frame
model.response	Extract Special Information from Model Frame
model.weights	Extract Special Information from Model Frame
nobs	Extract the Number of Observations from a Fit
nobs.default	Extract the Number of Observations from a Fit
nobs.glm	Extract the Number of Observations from a Fit
nobs.lm	Extract the Number of Observations from a Fit
nobs.nls	Extract the Number of Observations from a Fit
numericDeriv	Evaluate derivatives numerically
offset	Set an Offset Value in a Modeling Formula
power	Generate a Power Link Object
predict	Make Predictions from a Fitted Model Object
predict.ar	Fit Autoregressive Models to Time Series
predict.glm	Predict Method for a Generalized Linear Model
predict.lm	Predict Method for a Linear Model
predict.mlm	Predict Method for a Linear Model
predict.nls	Predicting from Nonlinear Least Squares Fits

Function name	Title description
princomp	Principal Components Analysis
princomp.default	Principal Components Analysis
princomp.formula	Principal Components Analysis
print.aov	Fit an Analysis of Variance Model
print.aovlist	Fit an Analysis of Variance Model
print.ar	Fit Autoregressive Models to Time Series
print.dummy_coef	Extract Original Coefficients from a Linear Model
print.dummy_coef_list	Extract Original Coefficients from a Linear Model
print.factanal	Estimate a Factor Analysis Model
print.logLik	Extract Log-Likelihood
relevel	Reorder Levels of Factor
relevel.default	Reorder Levels of Factor
relevel.factor	Reorder Levels of Factor
relevel.ordered	Reorder Levels of Factor
resid	Extract Information from a Model
residuals	Extract Information from a Model
residuals.default	Extract Information from a Model
residuals.glm	Compute Residuals for glm Objects
selfStart	Construct Self-starting Nonlinear Models
selfStart.default	Construct Self-starting Nonlinear Models
selfStart.formula	Construct Self-starting Nonlinear Models
sigma	Extract Residual Standard Deviation
SSasymp	Asymptotic Regression Model
SSasympOff	Asymptotic Regression Model
SSasympOrig	Asymptotic Regression Model
SSbiexp	Biexponential Model: The Sum of Two Exponentials
SSD	SSD Matrix and Estimated Variance Matrix in Multivariate Models
SSD.mlm	SSD Matrix and Estimated Variance Matrix in Multivariate Models

Function name	Title description
SSfol	First-order Compartment Model
SSfpl	Four-parameter Logistic Model
SSgompertz	Self-Starting Nls Gompertz Growth Model
SSlogis	Fitting a Logistic Curve
SSmicmen	Michaelis-Menten Model
SSweibull	Self-Starting Nls Weibull Growth Curve Model
stat.anova	Add Statistics Columns to an Anova Table
str.logLik	Extract Log-Likelihood
terms	Create or Extract a Terms Object
terms.aovlist	Create or Extract a Terms Object
terms.default	Create or Extract a Terms Object
terms.formula	Create or Extract a Terms Object
terms.terms	Create or Extract a Terms Object
update	Update a Fitted Model Object
update.default	Update a Fitted Model Object
update.formula	Update a Fitted Model Object
vcov	Variance-Covariance Matrix of the Estimated Coefficients
vcov.glm	Variance-Covariance Matrix of the Estimated Coefficients
vcov.lm	Variance-Covariance Matrix of the Estimated Coefficients
vcov.mlm	Variance-Covariance Matrix of the Estimated Coefficients
vcov.nls	Variance-Covariance Matrix of the Estimated Coefficients
vcov.summary.glm	Variance-Covariance Matrix of the Estimated Coefficients
vcov.summary.lm	Variance-Covariance Matrix of the Estimated Coefficients

Time Series

These are the available functions for time series. See each function's help topic in the TERR Language Reference for more information.

Function name	Title description
acf	Auto- and Cross- Covariance or Correlation Estimation

Function name	Title description
aggregate	Compute Summary Statistics of Subsets of Data
aggregate.default	Compute Summary Statistics of Subsets of Data
aggregate.formula	Compute Summary Statistics of Subsets of Data
aggregate.ts	Compute Summary Statistics of Subsets of Data
ar	Fit Autoregressive Models to Time Series
ar.yw	Fit Autoregressive Models to Time Series
arima	ARIMA modeling of Time Series
arima.sim	Simulate a Univariate ARIMA Series
as.ts	Time Series Objects
Box.test	Box-Pierce and Ljung-Box Tests
cbind.ts	Union and Intersection of Time Series
ccf	Auto- and Cross- Covariance or Correlation Estimation
coef.Arima2	ARIMA modeling of Time Series
cycle	Create Time Vector or Index of Frequency
cycle.default	Create Time Vector or Index of Frequency
decompose	Classical Seasonal Decomposition by Moving Averages
diff	Create an Object of Differences
diff.Date	Create an Object of Differences
diff.default	Create an Object of Differences
diff.POSIXt	Create an Object of Differences
diff.ts	Create an Object of Differences
diffinv	Discrete Integration: inverse of diff
diffinv.default	Discrete Integration: inverse of diff
diffinv.ts	Discrete Integration: inverse of diff
fft	Fast Fourier Transform
filter	Apply a Filter to a Time Series
HoltWinters	Holt-Winters Filtering
is.mts	Time Series Objects

Function name	Title description
is.ts	Time Series Objects
lag	Create a Lagged Time Series
mvfft	Fast Fourier Transform
na.contiguous	Find Longest Contiguous Stretch of non-NAs
na.contiguous.default	Find Longest Contiguous Stretch of non-NAs
na.contiguous.ts	Find Longest Contiguous Stretch of non-NAs
pacf	Auto- and Cross- Covariance or Correlation Estimation
predict.ar	Fit Autoregressive Models to Time Series
predict.HoltWinters	Holt-Winters Filtering
print.ar	Fit Autoregressive Models to Time Series
print.Arima2	ARIMA modeling of Time Series
print.HoltWinters	Holt-Winters Filtering
print.ts	Print a Time Series
residuals.HoltWinters	Holt-Winters Filtering
time	Create Time Vector or Index of Frequency
time.default	Create Time Vector or Index of Frequency
ts	Time Series Objects
ts.intersect	Union and Intersection of Time Series
ts.union	Union and Intersection of Time Series
vcov.Arima2	ARIMA modeling of Time Series
window	Window a Time Series
window.default	Window a Time Series
window.ts	Window a Time Series
window<-	Window a Time Series
window<-.ts	Window a Time Series

TERR

These specialized TERR functions are available in TERR. For help with a function, see its listing in the TERR Language Reference.

Functions Using the cURL Library for Access to URLs

These are the available functions for using the cURL library. See the function's help topic in the TERR Language Reference for more information.

Function name	Title description
`download.file`	Download a File from the Internet

Functions Not Available in TIBCO Enterprise Runtime for R

With each release, TIBCO® Enterprise Runtime for R (TERR™) strives for compatibility with open-source R. In version 6.1, we tested functionality against open-source R version 4.0. At this time, the functions and methods listed in this document are available in open-source R, but are not currently available in TERR™.

The functions are listed alphabetically, according to their packages. The statistics package is further categorized by types of functions.

- base
- utils
- methods
- statistics

This table provides information about the numbers of functions implemented in open-source R that are also available in TIBCO Enterprise Runtime for R.

Package	Open-source R functions	TERR functions	Percent of open-source R functions implemented in TERR (%)
base	1359	1201	88.4
datasets	104	104	100
graphics*	88	86	97.7
grDevices*	113	104	92
methods	371	103	27.8
stats	456	394	86.4
utils	219	140	63.9

*Because TERR is included with TIBCO Spotfire®, most of the functions related to directly producing graphics are not supported; however, TERR contains placeholders for many graphics functions for package compatibility purposes.

If you want to use the graphics functions available in R, you can use the `RGraph` function in the RinR package to generate graphics by passing an expression to open-source R and evaluating it there. You can also use a number of packages that use Javascript and htmlwidgets to create browser-based graphics. For more information, see Graphics in TIBCO Enterprise Runtime for R.

- For a list of available functions, see Available functions in TIBCO Enterprise Runtime for R..
- For more information about the differences between TERR, see the document *Differences Between TIBCO® Entrprise Runtime for R and Open-Source R*.

You can reach these documents from the TERR Landing Page. From the TERR console, type `help.start()`, and then click the link on the page.

To see a list of the functions included in TERR, from the TERR console, or from TERR running in RStudio, run the following commands:

```
objects("package:base")
objects("package:methods")
objects("package:utils")
objects("package:stats")
```

Base Functions Not Available in TIBCO Enterprise Runtime for R

The following functions in the open-source R base package are not implemented in TERR version 6.1.

[.Dlist	conflictRules	lazyLoadDBexec	print.srcfile
[.DLLInfoList	conflicts	length<-.Date	print.summary.warnings
[.simple.list	contributors	length<-.difftime	pushBack
[.warnings	Cstack_info	length<-.POSIXct	pushBackLength
[[<-.POSIXlt	debug	length<-.POSIXlt	retracemem
[<-.numeric_version	debuggingState	licence	returnValue
all.equal.environment	debugonce	license	sequence.default
all.equal.envRefClass	duplicated.warnings	loadingNamespaceInfo	serverSocket
anyNA.data.frame	dynGet	mem.maxNSize	socketAccept
anyNA.numeric_version	env.profile	mem.maxVSize	socketSelect
anyNA.POSIXlt	file.link	memory.profile	socketTimeout
as.list.difftime	findPackageEnv	namespaceImport	srcfilealias
asS3	format.AsIs	namespaceImportClasses	srcref
autoload	format.libraryIQR	namespaceImportFrom	summary.srcfile
autoloader	format.packageInfo	namespaceImportMethods	summary.srcref
bindtextdomain	format.summaryDefault	open.srcfile	summary.warnings
bodyOK	gc.time	open.srcfilealias	sys.save.image
browserCondition	getCallingDLL	open.srcfilecopy	Sys.setFileTime
browserSetDebug	getCallingDLLe	packageHasNamespace	tracemem
browserText	icuGetCollate	packageNotFoundError	tracingState
builtins	icuSetCollate	parseNamespaceFile	truncate.connection
c.warnings	importIntoEnv	pcre_config	undebug
callCC	is.na<-.numeric_version	print.Dlist	unique.warnings
clearPushBack	length<-.Date	pushBack	untracemem
close.srcfile	La_library	print.function	xpdrows.data.frame
close.srcfilealias	La_version	print.selfStart	xtfrm.AsIs

Methods Functions Not Available in TIBCO Enterprise Runtime for R

The following functions in the open-source R methods package are not implemented in TERR version 6.1.

addNextMethod	finalDefaultMethod	isXS3Class	promptClass
asMethodDefinition	findClass	linearizeMlist	promptMethods
assignMethodsMetaData	findMethod	listFromMethods	reconcilePropertiesAndPrototype

balanceMethodsList	findMethods	listFromMlist	registerImplicitGenerics
cacheGenericsMetaData	findMethodSignatures	loadMethod	rematchDefinition
cacheMetaData	findUnique	makeClassRepresentation	requireMethods
cacheMethod	fixPre1.8	makeExtends	resetGeneric
checkAtAssignment	generic.skeleton	makeGeneric	S3Class<-
checkSlotAssignment	getAllSuperClasses	makeMethodsList	sealClass
classesToAM	getGenerics	makePrototypeFromClassDef	setGroupGeneric
classMetaName	getGroup	makeStandardGeneric	setPrimitiveMethods
completeClassDefinition	getMethodsForDispatch	matchSignature	showDefault
completeExtends	getMethodsMetaData	mergeMethods	showExtends
completeSubclasses	getValidity	metaNameUndo	showMlist
conformMethod	hasMethods	method.skeleton	SignatureMethod
defaultDumpName	inheritedSlotNames	MethodAddCoerce	sigToEnv
defaultPrototype	initFieldArgs	methodSignatureMatrix	slotsFromS3
doPrimitiveMethod	insertClassMethods	MethodsList	substituteFunctionArgs
dumpMethod	insertMethod	MethodsListSelect	superClassDepth
dumpMethods	insertSource	multipleClasses	testInheritedMethods
elNamed	isClassDef	newBasic	testVirtual
elNamed<-	isGrammarSymbol	newClassRepresentation	tryNew
empty.dump	isGroup	newEmptyObject	unRematchDefinition
emptyMethodsList	linearizeMlist	reconcilePropertiesAndPrototype	validSlotNames
evalSource	isSealedClass	possibleExtends	
externalRefMethod	isSealedMethod	prohibitGeneric	

Utilities Functions Not Available in TIBCO Enterprise Runtime for R

The following functions in the open-source R utils package are not implemented in TERR version 6.1.

alarm	de.ncols	packageStatus	RweaveLatexFinish
aregexec	de.restore	page	RweaveLatexOptions
asDateBuilt	de.setup	process.events	RweaveLatexSetup
aspell	debugcall	promptImport	RweaveLatexWritedoc
aspell_package_C_files	debugger	read.DIF	RweaveTryStop
aspell_package_R_files	fileSnapshot	read.fortran	setBreakpoint
aspell_package_Rd_files	findLineNum	read.socket	Stangle
aspell_package_vignettes	fixInNamespace	recover	summaryRprof
aspell_write_personal_dictionary_file	hsearch_db	removeSource	Sweave
browseEnv	hsearch_db_concepts	Rprofmem	SweaveHooks
changedFiles	hsearch_db_keywords	RShowDoc	SweaveSyntaxLatex
checkCRAN	isS3method	RSiteSearch	SweaveSyntaxNoweb
chooseBioCmirror	limitedLabels	rtags	SweaveSyntConv
cite	make.packages.html	Rtangle	undebugcall
citeNatbib	make.socket	RtangleSetup	upgrade

close.socket	makeRweaveLatexCodeRunner	RtangleWritedoc	url.show
create.post	mirror2html	RweaveChunkPrefix	vignette
data.entry	news	RweaveEvalWithOpt	write.socket
dataentry	nsl	RweaveLatex	
de	osVersion		

Statistics Functions Not Available in TIBCO Enterprise Runtime for R

The following statistics functionality is not available in TERR version 6.1. The complete listings are in the related topics.

- Certain functionality for statistical modeling, including the following.

Statistical modeling	Functionality not available in TERR
Clustering	`plclust`, a cluster plotting function.
Density estimation	`kernel` and related functions.
Hypothesis testing	Some lesser-used tests.
Time series models	`spectrum`, `spec.pgram`, and other frequency domain functions.
Miscellaneous statistics models	`ppr`.

- The optimization function `constrOptim`.
- A few infrequently encountered statistical distribution-related functions.
- Most of the functions related to directly producing graphics.

Clustering Functions

The following clustering functions found in open-source R are not available in TERR version 6.1.

plclust

Density Functions

The following density functions found in open-source R are not available in TERR version 6.1.

bandwidth.kernel	bw.ucv
bw.bcv	df.kernel
kernapply	kernel

Distribution Functions

The following distribution functions found in open-source R are not available in TERR version 6.1.

pbirthday	ptukey
qbirthday	rWishart

factor.analysis Functions

The following `factor.analysis` functions open-source R are not available in TERR version 6.1.

promax
varimax

Graphics Functions

The following statistics graphics functions found in open-source R are not available in TERR version 6.1.

heatmap	lag.plot
plot.spec.coherency	plot.spec.phase
scatter.smooth	

htest Functions

The following `htest` functions found in open-source R are not available in TERR version 6.1.

fligner.test	power.anova.test
mauchly.test	power.prop.test
mood.test	power.t.test
pairwise.prop.test	PP.test
pairwise.t.test	prop.trend.test
pairwise.wilcox.test	quade.test
poisson.test	

Miscellaneous Model Functions

The following miscellaneous model functions found in open-source R are not available in TERR version 6.1.

ls.diag
manova
Pair
step
summary.manova
TukeyHSD

Miscellaneous stats Functions

The following miscellaneous stats functions found in open-source R are not available in TERR version 6.1.

ppr

Optimization Functions

The following optimization functions found in open-source R are not available in TERR version 6.1.

constrOptim

Time Series Functions

The following time series functions found in open-source R are not available in TERR version 6.1.

acf2AR	ar.burg
ar.mle	ar.ols
arima0	arima0.diag
ARMAacf	cpgram
is.tskernel	KalmanForecast
KalmanLike	KalmanRun
KalmanSmooth	spec.ar
spec.pgram	spec.taper
spectrum	StructTS
tsdiag	tsSmooth

Configure RStudio to use TIBCO Enterprise Runtime for R

You can use the RStudio™ integrated development environment (IDE) to write TIBCO® Enterprise Runtime for R (TERR™) scripts, expressions, and data functions.

RStudio is a full-featured, open-source IDE for working with R code. It is provided independently of TIBCO Software Inc. You can learn more about RStudio, and you can download and use RStudio for your development at www.rstudio.com.

RStudio is available under separate open-source software license terms. TIBCO does not warrant, deliver, or support code or other material provided by RStudio, Inc., including but not limited to development tools and packages, and such code or other material does not constitute a part of the TERR engine.

TERR has been tested with RStudio 1.3.1093 release on the following platforms:

- Using the Windows desktop version of the RStudio IDE.
- Using the Linux desktop version of th RStudio IDE.
- Accessing the RStudio IDE Server, and then executing code remotely via a web browser.

To download the version of RStudio that was tested with this release of TERR, see Older Versions of RStudio.

Configuring RStudio Windows Desktop Edition to Run the TIBCO Enterprise Runtime for R Engine

Configure the Windows desktop version of the RStudio IDE to work with TERR.

Perform this task from the Windows desktop.

Prerequisites

You must have installed RStudio version 0.99 or later for the Windows desktop. (See http://support.spotfire.com/sr.asp for tested versions.)

Due to a change in RStudio for Windows between versions 1.1.136 and 1.1.336, using TERR to print (for example, `print(1:1000)`) runs extremely slowly. If you experience this problem, then in your RStudio session, set `options(error=NULL)`.

Procedure

1. From the Windows **Start** menu, launch RStudio.
2. From within RStudio, on the menu, click **Tools** > **Global Options**.
3. In the **General** tab, set the R version to your installation of TERR.

   For example, `C:\Program Files\TIBCO\terr60`.
4. Click **Apply**, and then restart RStudio.

Configuring RStudio Mac Desktop Edition to Run the TIBCO Enterprise Runtime for R Engine

You can use TERR in your RStudio installation on your Mac.

As of TERR 6.0, support for TERR on the Mac is deprecated and is no longer tested.

Prerequisites

- You must be running a supported version of the Mac operating system.
- You must have installed a supported RStudio version for the Mac desktop.
- You must have installed TERR version 6.1 on your Mac. See Installing and running TIBCO Enterprise Runtime for R on a Mac on page 15 for more information.

Procedure

1. On your Mac, open a Terminal session.
2. At the command prompt, type `RStudio-TERR`.
   RStudio launches with TERR as the language engine.

Configuring RStudio Linux Desktop Edition to Run the TIBCO Enterprise Runtime for R Engine

Configure the Linux desktop version of the RStudio IDE to work with TERR.

Perform this task from the Linux desktop.

Prerequisites

You must have installed RStudio version 0.99 or later for the Linux desktop. (See http://support.spotfire.com/sr.asp for tested versions.)

Procedure

- Linux RStudio desktop users should follow the instructions for specifying the R version for Linux in the RStudio documentation, found at https://support.rstudio.com/hc/en-us/articles/200486138-Using-Different-Versions-of-R.

Configuring RStudio Linux Server Edition to Run the TIBCO Enterprise Runtime for R Engine

Configure the Linux server version of the RStudio IDE to work with TERR.

Perform this task from the Linux server.

Prerequisites

You must have installed RStudio version 0.99 or later for the Linux server. (See http://support.spotfire.com/sr.asp for tested versions.)

Procedure

- Linux Server administrators should follow the instructions under *Specifying R Version* in the RStudio Server configuration documentation, found at https://support.rstudio.com/hc/en-us/articles/200552316.

Unsupported Features for Using TERR with RStudio

Some features available for open-source R in RStudio are not supported for using TERR.

- On Linux, you cannot save **Command History** to a file.
- Plots are not supported in TERR. The **Plots** tab produces no plot.

- Rcpp integration is currently not supported.
- The following file types, listed in the RStudio **File** > **New** menu, are not supported if you are using TERR as your engine.
  - C++ File
  - R SWeave
  - R HTML
  - Shiny Web App
  - R Presentation

Troubleshooting RStudio with TIBCO Enterprise Runtime for R

If you have problems configuring RStudio with TERR, check for advice in this topic.

When I configured RStudio on Windows to point to TERR instead of open-source R, I see a blank GUI.
You might see a blank user interface with a version of TERR older than version 4.2. This can occur if the version of RStudio you have installed is not compatible with TERR. Follow these steps to fix the problem.

1. In Windows Explorer, go to `C:\Users\`*`currentUser`*`\AppData\Roaming\RStudio`.
2. Delete the file `Desktop.ini`.

   This technique forces RStudio to refresh and default to open-source R as the engine.
3. Go to the requirements page for your version of TERR and note the version of RStudio for which it was tested. See http://support.spotfire.com/sr.asp, and in the **All product system requirements** drop-down list box, select TIBCO Enterprise Runtime for R (TERR).
4. Download and install that version of RStudio.
5. After you have the version of RStudio that is compatible with your version of TERR, follow the steps to point RStudio to your installation of TERR.

When I try to use an RStudio feature with TERR that requires the rmarkdown or shiny packages, RStudio tries to download the packages but fails.
If the rmarkdown or the shiny packages are not already installed, and if you try to use an RStudio feature that requires one or both of these packages, RStudio prompts to install them automatically. This process currently fails if RStudio is configured with the TERR engine. To work around this problem, at the command prompt, call `install.packages()` to install the rmarkdown and shiny packages.

When I click Help > Check for Updates, nothing happens.
When RStudio is configured with TERR, the menu item **Help** > **Check for Updates** does nothing. To check for updates to RStudio from this menu item, you must change the RStudio configuration to run open-source R.

An easy way to select an engine for RStudio is to hold down Ctrl while you start RStudio. This action displays the dialog for choosing a different R engine. You can select TERR in this dialog.

Package Compatibility

As a standard part of each TERR release, we run all help examples provided in packages in the Comprehensive R Archive Network (CRAN) in the TERR engine from the Windows and Linux platforms.

Beginning with release 5.1, we also run all help examples provided in packages in the Bioconductor (BIOC) list.

Package loading improvements

- To see a list of issues fixed to improve package loading and performance, see Closed Issues.
- To see a list of Rapi C API functions and TERR functions implemented in this release to improve package loading and performance, see New features.
- To see a complete listing of the Rapi C API functions implemented in TERR so far, run the following code.

```
library(terrUtils)
implementedRapiEntries()
```

Package compatibility analysis in Spotfire

We report the results of these tests in visualizations that are available on TIBCO Cloud™ Spotfire® at the following links. You can browse and review the results for the packages you want to use and their Task Views. You can review the results run for every expression in every help file for every CRAN package or BioConductor package for the following platforms.

CRAN packages	BioConductor packages
Windows	Windows
Linux	Linux

The tests run against code examples in the help provide some guidance for determining rates of success. The accuracy of information collected depends on the number and quality of examples in the package reference topics. These analyses are not meant to be the definitive determination of exact compatibility.

For more information about CRAN packages, see https://cran.r-project.org/web/packages/

- For more information about BioConductor packages, see https://www.bioconductor.org/packages/release/bioc/

TIBCO does not warrant, deliver, or support code or other material provided by the R Project for Statistical Computing, including but not limited to development tools and packages, and such code and other material does not constitute a part of TERR. Such material therefore is not within the scope of your license for TERR. Download and use of such material is solely at your own discretion and subject to the free open source license terms applicable to such material. TIBCO recommends that you consult a legal professional concerning compliance with any free open source license terms applicable to such material, particularly if you plan to engage in redistribution of TERR and/or such material. (Please note that TERR may be redistributed solely pursuant to a license that expressly grants such redistribution rights.)

Package compatibility summary in TERR

Alternatively, we provide a summary of our testing results on the TIBCO documentation site in the following CSV-formatted files.

- `CRANonTERR-Linux.csv`
- `CRANonTERR-Win.csv`
- `BIOConTERR-Linux.csv`
- `BIOConTERR-Win.csv`

TERR console code example for CRAN

You can use the TERR console to quickly access a summary of the information. The following example returns the results for testing the help files for the CRAN package caret. You can change the example to match the platform and package of your choice.

1. Change the CSV file name to match the platform (Linux or Windows).
2. Remove the comment markers for the CSV file name.
3. Provide the name of the package you want to query.

```
# Example:  CRAN tests compatibility with this version of TERR
#
#packageCompat <- read.csv("https://docs.tibco.com/pub/enterprise-runtime-for-R/6.0.0/doc/
csv/CRANonTERR-Linux.csv",
#   stringsAsFactors=FALSE)
#
#packageCompat <- read.csv("https://docs.tibco.com/pub/enterprise-runtime-for-R/6.0.0/doc/
csv/CRANonTERR-Win.csv",
#   stringsAsFactors=FALSE)
#
subset(packageCompat, Package.Name=="caret")
## (update)
```

The returned results resemble the following for the above example.

```
    Package.Name Version            Status Percent.Successful Total.Executed
1608        caret  6.0-86 Mostly successful               98.5%            205
    Passed Failed Graphics Random.Numbers
1608    202      3       17             100
```

This table shows how to read the results for this example. (Shows sample results for running the example on Linux. Results for Windows can vary slightly.)

Column name	Result
Package.Name	caret
Version	6.0-86
Status	Mostly successful
Percent.Successful	98.5%
Total.Executed	205
Passed	202
Failed	3
Graphics	17

Column name	Result
Random.Numbers	100

TERR console code example for BioConductor

You can use the TERR console to quickly access a summary of the information. The following example returns the results for testing the help files for the BioConductor package limma. You can change the example to match the platform and package of your choice.

1. Change the CSV file name to match the platform (Linux or Windows).
2. Remove the comment markers for the CSV file name.
3. Provide the name of the package you want to query.

```
# Example:  BioConductor tests compatibility with this version of TERR
#
#packageCompat <- read.csv("https://docs.tibco.com/pub/enterprise-runtime-for-R/6.0.0/doc/
csv/BioConTERR-Linux.csv",
#   stringsAsFactors=FALSE)
#
#packageCompat <- read.csv("https://docs.tibco.com/pub/enterprise-runtime-for-R/6.0.0/doc/
csv/BioConTERR-Win.csv",
#   stringsAsFactors=FALSE)
#
subset(packageCompat, Package.Name=="limma")
## (update)
```

The returned results resemble the following for the above example.

```
Package.Name Version           Status Percent.Successful Total.Executed
901        limma  3.44.3 Mostly successful              96.9            391
    Passed Failed Graphics Random.Numbers
901    379     12       49             75
```

This table shows how to read the results for this example. (Shows sample results for running the example on Linux. Results for Windows can vary slightly.)

Column name	Result
Package.Name	limma
Version	3.44.3
Status	Mostly successful
Percent.Successful	96.9%
Total.Executed	391
Passed	379
Failed	12
Graphics	49
Random.Numbers	75

TIBCO Enterprise Runtime for R Release Notes

The Release Notes for this product version are provided to inform you of new features, known issues, and issues from previous releases that have been closed.

These release notes are for TIBCO® Enterprise Runtime for R (TERR™) version 6.1. They cover Linux® and Microsoft Windows® installations.

TERR™ is a high-performance statistical engine, which is compatible with open-source R. It can be embedded into a wide range of applications as an enterprise-grade alternative to open-source R, and can run a wide array of packages from CRAN.

This release of TERR focuses on improving compatibility with open-source R.

Open-source R is available under separate open source software license terms and is not part of TERR. As such, open-source R is not within the scope of your license for TERR. Open-source R is not supported, maintained, or warranted in any way by TIBCO Software Inc. Download and use of open-source R is solely at your own discretion and subject to the free open source license terms applicable to open-source R.

New Features

The following features have been added to version 6.1.0 of TERR.

TERR 6.1.0 has changed to accommodate the changes made to open-source R version 4.0.x.

New implementation for R 4.0.x compatiblity
TERR has new functions and changes to existing functions and behavior to match new functions and changes in open-source R 4.0.x.

New functions in included packages

Package	Name	Details

New R C API Entries

The following R C API entries are implemented to improve package loading and compatibility. For a complete listing of the Rapi C API entries implemented in TERR, run the following.

```
library(terrUtils)

implementedRapiEntries()
```

R C API entries	Description

Changes in Functionality, Features, and Compatibility

From release to release, we might change the functionality. In cases where product changes require migration procedures, we provide information for that purpose. The following changes have been made to TERR version 6.1.

Changes in functionality and compatibility

Version 6.1 of TERR was tested with open-source R version 4.0.2. The following changes to functions in TERR were made for compatibility with open-source R 4.0.2 or with Java 11.

Key	Description

Other version updates

Version updates and version compatibility testing include the following.

- OpenSSL updated to version 1.1.1g (Included with TERR).
- Tested with JAVA version 11; however, any compatible version of Java can be used.
- open-source R version 4.0.2.
- RStudio Desktop and Server version 1.3.1093. For more information, see *Configuring RStudio to use TIBCO® Enterprise Runtime for R* in the TERR documentation.

TERR is no longer tested with Spark, KNIME, or Hadoop. See the TERR articles on the TIBCO Community site for more information about using these tools.

The following packages included with TERR require a bit-matching 32-bit or 64-bit version Java.

- parallel
- sjdbc
- terrJava

Additionally, if you want to use the rJava package, or any other CRAN package that requires access to `JAVA_HOME`, this information applies.

To use these packages, you must set the `JAVA_HOME` environment variable to a valid Java installation before you load the packages. You can set `JAVA_HOME` using TERR by calling the following in the console:

```
Sys.setenv(JAVA_HOME="path_to_your_JRE_installation")
```

To check if the environment variable is set, call the following in the console:

```
Sys.getenv("JAVA_HOME")
```

For Windows installations, if the `JAVA_HOME` environment variable is not set, TERR uses system info to identify and load the latest Java installed on the system.

Deprecated and Removed Features

The following features have been deprecated or removed in TERR version 6.1.

Deprecated on macOS

Removed features

As of version 6.1, support for TERR is removed on macOS.

Update or Reinstall a Version of TERR

When you update to a more recent version of TERR, remove the previous version, and follow the guidelines to protect your installed packages.

Installing TERR over a previous TERR installation without removing it can cause unexpected failures. To remove TERR, follow the instructions for your operating system.

For a list of operating systems that TERR is supported on, see the TIBCO Enterprise Runtime for R system requirements.

When you update to a new version of TERR, follow the guidance in "Package installation locations and recommendations for updating" in the Package Management section of the *TIBCO® Enterprise Runtime for R Technical Documentation*.

On Linux

- From the command line, run the command `rm -rf <TERR_HOME>`, where *TERR_HOME* is the installation directory.

On Windows

- From the Settings app, click **Apps**, and then from the **Apps & features** list, double-click run the listing **TIBCO Enterprise Runtime for R <*version#*>**.

After uninstalling TERR, run the installer for the version of TERR you want to install.

Package Compatibility

As a standard part of each TERR release, we run all help examples provided in packages in the Comprehensive R Archive Network (CRAN) in the TERR engine from the Windows and Linux platforms.

Beginning with release 5.1, we also run all help examples provided in packages in the Bioconductor (BIOC) list.

Package loading improvements

- To see a list of issues fixed to improve package loading and performance, see Closed Issues.
- To see a list of Rapi C API functions and TERR functions implemented in this release to improve package loading and performance, see New features.
- To see a complete listing of the Rapi C API functions implemented in TERR so far, run the following code.

```
library(terrUtils)
implementedRapiEntries()
```

Package compatibility analysis in Spotfire

We report the results of these tests in visualizations that are available on TIBCO Cloud™ Spotfire® at the following links. You can browse and review the results for the packages you want to use and their Task Views. You can review the results run for every expression in every help file for every CRAN package or BioConductor package for the following platforms.

CRAN packages	BioConductor packages
Windows	Windows
Linux	Linux

The tests run against code examples in the help provide some guidance for determining rates of success. The accuracy of information collected depends on the number and quality of examples in the package reference topics. These analyses are not meant to be the definitive determination of exact compatibility.

For more information about CRAN packages, see https://cran.r-project.org/web/packages/

- For more information about BioConductor packages, see https://www.bioconductor.org/packages/release/bioc/

TIBCO does not warrant, deliver, or support code or other material provided by the R Project for Statistical Computing, including but not limited to development tools and packages, and such code and other material does not constitute a part of TERR. Such material therefore is not within the scope of your license for TERR. Download and use of such material is solely at your own discretion and subject to the free open source license terms applicable to such material. TIBCO recommends that you consult a legal professional concerning compliance with any free open source license terms applicable to such material, particularly if you plan to engage in redistribution of TERR and/or such material. (Please note that TERR may be redistributed solely pursuant to a license that expressly grants such redistribution rights.)

Package compatibility summary in TERR

Alternatively, we provide a summary of our testing results on the TIBCO documentation site in the following CSV-formatted files.

- `CRANonTERR-Linux.csv`
- `CRANonTERR-Win.csv`
- `BIOConTERR-Linux.csv`
- `BIOConTERR-Win.csv`

TERR console code example for CRAN

You can use the TERR console to quickly access a summary of the information. The following example returns the results for testing the help files for the CRAN package caret. You can change the example to match the platform and package of your choice.

1. Change the CSV file name to match the platform (Linux or Windows).
2. Remove the comment markers for the CSV file name.
3. Provide the name of the package you want to query.

```
# Example:  CRAN tests compatibility with this version of TERR
#
#packageCompat <- read.csv("https://docs.tibco.com/pub/enterprise-runtime-for-R/6.0.0/doc/
csv/CRANonTERR-Linux.csv",
#   stringsAsFactors=FALSE)
#
#packageCompat <- read.csv("https://docs.tibco.com/pub/enterprise-runtime-for-R/6.0.0/doc/
csv/CRANonTERR-Win.csv",
#   stringsAsFactors=FALSE)
#
subset(packageCompat, Package.Name=="caret")
## (update)
```

The returned results resemble the following for the above example.

```
    Package.Name Version           Status Percent.Successful Total.Executed
1608        caret  6.0-86 Mostly successful               98.5%            205
    Passed Failed Graphics Random.Numbers
1608    202      3       17             100
```

This table shows how to read the results for this example. (Shows sample results for running the example on Linux. Results for Windows can vary slightly.)

Column name	Result
Package.Name	caret
Version	6.0-86
Status	Mostly successful
Percent.Successful	98.5%
Total.Executed	205
Passed	202
Failed	3
Graphics	17
Random.Numbers	100

TERR console code example for BioConductor

You can use the TERR console to quickly access a summary of the information. The following example returns the results for testing the help files for the BioConductor package limma. You can change the example to match the platform and package of your choice.

1. Change the CSV file name to match the platform (Linux or Windows).
2. Remove the comment markers for the CSV file name.
3. Provide the name of the package you want to query.

```
# Example:  BioConductor tests compatibility with this version of TERR
#
#packageCompat <- read.csv("https://docs.tibco.com/pub/enterprise-runtime-for-R/6.0.0/doc/
csv/BioConTERR-Linux.csv",
#   stringsAsFactors=FALSE)
#
#packageCompat <- read.csv("https://docs.tibco.com/pub/enterprise-runtime-for-R/6.0.0/doc/
csv/BioConTERR-Win.csv",
#   stringsAsFactors=FALSE)
#
subset(packageCompat, Package.Name=="limma")
## (update)
```

The returned results resemble the following for the above example.

```
Package.Name Version           Status Percent.Successful Total.Executed
901        limma  3.44.3 Mostly successful               96.9            391
    Passed Failed Graphics Random.Numbers
901    379     12       49             75
```

This table shows how to read the results for this example. (Shows sample results for running the example on Linux. Results for Windows can vary slightly.)

Column name	Result
Package.Name	limma
Version	3.44.3
Status	Mostly successful
Percent.Successful	96.9%
Total.Executed	391
Passed	379
Failed	12
Graphics	49
Random.Numbers	75

Closed Issues

This table lists closed issues in version 6.1.0 of TERR. It reflects fixes for issues with package compatibility, open-source R compatibility, and general issues with TERR.

Key	Description
TERR-7929	The function `registerS3method` now registers the method in the generic function's environment, if possible. This change fixes problems with the tibble package (v3.0.4), which explicitly calls `registerS3method` to register functions like `format.tbl`.

Known Issues

This section lists known issues in version 6.1.0 of TERR.

In this release, some open-source R functionality is not available, including graphics devices, and some functions from the base and stats packages. Likewise, S4 is not entirely compatible. The following table lists additional known issues.

Issue	Description
TERR-4993	On Linux, you cannot save TERR Command History to a file.
TERR-5488	If you receive the message `"SSL certificate problem: unable to get local issuer certificate"` when accessing SSL protected URLs (for example, `https://`), make sure that your system has the latest version of the certificate authority database for your system. On RedHat 5, this should be `` `yum update openssl` ``. On RedHat 6, this should be `` `yum update ca-certificates` ``.
TERR-6422	Setting breakpoints in RStudio 0.99.903 with TERR on Windows can cause it to crash. When a script is sourced, and if it has breakpoints set, RStudio for Windows can crash. This crash does not occur when breakpoints are set inside a function and the function is called.
TERR-6576	If you try to use an RStudio feature that requires the rmarkdown package or the shiny package, and the required package is not already installed, then this process currently fails if RStudio is configured with the TERR engine. To work around this problem, at the command prompt, call `install.packages()` to install the rmarkdown and shiny packages.

Issue	Description
TERR-6812	Due to changes in open-source R version 3.5 and resulting compatibility changes in TERR 5.0, packages that are built with a version of TERR prior to 5.0 must be rebuilt. • To install a binary package from a repository, always call `install.packages(pkgname)` from TERR. The `install.packages` function finds the correct binary version in the repository for your version of TERR. Manually downloading the binary package from CRAN can result in errors when you use it with TERR. • To install a package from source, try installing it first with TERR (with `install.packages` in TERR or with `TERR CMD INSTALL` from a command line). • To install a package from source that you cannot build with TERR, install the package with the version of open-source R tested with TERR.
TERR-7077	Certain packages, such as rJava, cannot be installed with TERR from source under Centos. If you encounter a package that does not install with TERR from source, then you can build the package using open-source R, and then install the binary package in TERR.
TERR-7727	When you view help files in Rstudio, some text in the help (for example, the Arguments and the Value sections) might not be visible if you are using an Rstudio Editor theme with a dark background.
TERR-7728	When you run TERR under Rstudio Server, if you end the session using **File** > **Quit Session**, the following message is displayed: `Session Error, The previous R session was abnormally terminated due to an unexpected crash`. This problem can also occur when you open a project using the **File** > **New Project** or **File** > **Open Project** menu commands.

Package search order

When you install a package using TERR, by default, TERR first checks for the package on TRAN, and then checks on MRAN. TERR installs the first version it finds. This is different than open-source R, which installs packages according to the newest version number available on CRAN. This difference is by design, because occasionally a CRAN package update causes a break with TERR compatibility, so we make available a tested version of the package on TRAN.

If you need to install one of these packages using open-source R (for example, to get source code on Linux), you can install the CRAN package, and then set `options()$repos` to install from only TRAN before reinstalling the package.

See "Specifying an older package on TRAN" in the *TIBCO® Enterprise Runtime for R Technical Documentation* for more information.

When running on RedHat Linux, TIBCO Enterprise Runtime for R processes spawned by the parallel package may immediately crash

We have seen a problem when running TERR on RedHat Linux with versions of Java earlier than 1.7.0_40. If you call the `makeCluster` function in the parallel package to spawn new TERR processes, these processes may immediately crash with a fatal Java error. To test if this problem is occurring, try the following:

```
library(parallel)
c1 <- makeCluster(1, outfile="")
# create cluster with one spawned process
# specifying outfile="" to print all output
c1 <- makeCluster(1, outfile="")
clusterEvalQ(c1, 123)
```

If this problem is occurring, you see an error such as the following:

```
> library(parallel)
```

```
> # create cluster with one spawned process
> # specifying outfile="" to print all output from the process
> c1 <- makeCluster(1, outfile="")
Creating 1 TERR cluster nodes at Thu Aug 20 11:31:25 2020
> clusterEvalQ(c1, 123)
1: #
1: # A fatal error has been detected by the Java Runtime Environment:
1: #
1: # SIGSEGV (0xb) at pc=0x0000003ac2cbbfa5, pid=12649, tid=1075054912
1: #1: # JRE version: 7.0_13-b20
1: # Java VM: Java HotSpot(TM) 64-Bit Server VM (23.7-b01 mixed mode linux-amd64
compressed oops)
1: # Problematic frame:
1: # C [libstdc++.so.6+0xbbfa5) __cxa_allocate_exception+0x55
1: #
1: # Failed to write core dump. Core dumps have been disabled. To enable core
dumping, try "ulimit -c unlimited" before starting Java again
1: #
1: # An error report file with more information is saved as:
1: # /a/seafiler01.na.tibco.com/vol/vol2/users/jdoe/hs_err_pid12649.log
1: #
1: # If you would like to submit a bug report, please visit:
1: # http://bugreport.sun.com/bugreport/crash.jsp
1: #
Error in waitForClusterReady(cl) : some cluster nodes have crashed or stopped: all crashed
```

The workaround for this problem is to set the `LD_PRELOAD` environment variable to `libstdc++.so.6`. This can be done before TERR is started, or within TERR, before the parallel library has been loaded:

```
> Sys.setenv("LD_PRELOAD"="libstdc++.so.6")
> library(parallel)
> c1 <- makeCluster(1, outfile="")
Creating 1 TERR cluster nodes at Thu Aug 20 11:32:56 2020
> # create cluster with one spawned process
> # specifying outfile="" to print all output
> c1 <- makeCluster(1, outfile="")
Creating 1 TERR cluster nodes at Thu Aug 20 11:33:51 2020
> clusterEvalQ(c1, 123)
1: TIBCO Software Inc. Confidential Information
1: Copyright (C) 2011-2020 TIBCO Software Inc. ALL RIGHTS RESERVED
1: TIBCO Enterprise Runtime for R version 6.0.0 for Microsoft Windows 64-bit
1:
1: Type 'help()' for help.
1: Type 'q()' to quit.
1: started engine node pid==15788 at Thu Aug 20 11:33:54 2020
[[1]]
[1] 123
>
```

Legal and Third-Party Notices

Index

Special Characters

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z