

R Packages

FRE6871 & FRE7241, Spring 2023

Jerzy Pawlowski jp3900@nyu.edu

NYU Tandon School of Engineering

May 12, 2023



NYU

**TANDON SCHOOL
OF ENGINEERING**

R Packages

Types of R Packages

R can run libraries of functions called packages,

R packages can also contain data,

Most packages need to be *loaded* into R before they can be used,

R includes a number of base packages that are already installed and loaded,

There's also a special package called the base package, which is responsible for all the basic R functionality, `datasets` is a base package containing various datasets, for example `EuStockMarkets`,

The *base* Packages

R includes a number of packages that are pre-installed (often called *base* packages),

Some *base* packages:

- *base* - basic R functionality,
- *stats* - statistical functions and random number generation,
- *graphics* - basic graphics,
- *utils* - utility functions,
- *datasets* - popular datasets,
- *parallel* - support for parallel computation,

Very popular packages:

- *MASS* - functions and datasets for "Modern Applied Statistics with S",
- *ggplot2* - grammar of graphics plots,
- *shiny* - interactive web graphics from R,
- *slidify* - HTML5 slide shows from R,
- *devtools* - create R packages,
- *roxygen2* - document R packages,
- *Rcpp* - integrate C++ code with R,
- *RcppArmadillo* - interface to Armadillo linear algebra library,
- *forecast* - linear models and forecasting,
- *tseries* - time series analysis and computational finance,
- *zoo* - time series and ordered objects,
- *xts* - advanced time series objects,
- *quantmod* - quantitative financial modeling framework,
- *caTools* - moving window statistics for graphics and time series objects,

CRAN Package Views


CRAN view for package **AER**:

<http://cran.r-project.org/web/packages/AER/>

Note:

- Authors,
- Version number,
- Reference manual,
- Vignettes,
- Dependencies on other packages.

The package source code can be downloaded by clicking on the **package source** link,



The screenshot shows the CRAN web page for the 'AER' package. The browser address bar displays 'cran.us.r-project.org/web/packages/AER/'. The page title is 'AER: Applied Econometrics with R'. Below the title, it states 'Functions, data sets, examples, demos, and vignettes for the book Christian Kleiber and Achim Zeileis (2008), Applie'. The page lists various details about the package, including its version (1.2-1), dependencies (R (≥ 2.13.0), car (≥ 2.0-1), lme4, sandwich, survival, zoo), imports (stats, Formula (≥ 0.2-0)), suggests (boot, dplyr, effects, foreign, ineq, KernSmooth, lattice, MASS, mlogit, nlme, rnet, np, plm, pscl), published date (2013-11-07), author (Christian Kleiber [aut], Achim Zeileis [aut, cre]), maintainer (Achim Zeileis <Achim.Zeileis@R-project.org>), license (GPL-2), needs compilation (no), citation (AER citation info), materials (NEWS), in views (Econometrics, Survival, TimeSeries), and CRAN checks (AER results). There is a 'Downloads:' section with links for the reference manual (AER.pdf), vignettes (Applied Econometrics with R: Package Vignette and Errata, Sweave Example: Linear Regression for Economics Journals Data), package source (AER_1.2-1.tar.gz), MacOS X binary (AER_1.2-1.tgz), Windows binary (AER_1.2-1.zip), and old sources (AER archive). A 'Reverse dependencies:' section lists reverse depends (lpack, rdd) and reverse suggests (censReg, glmx, lme4, micEconCES, mlogit, plm, REEMtree, sandwich).

cran.us.r-project.org/web/packages/AER/

AER: Applied Econometrics with R

Functions, data sets, examples, demos, and vignettes for the book Christian Kleiber and Achim Zeileis (2008), Applie

Version: 1.2-1

Depends: R (≥ 2.13.0), [car](#) (≥ 2.0-1), [lme4](#), [sandwich](#), [survival](#), [zoo](#)

Imports: stats, [Formula](#) (≥ 0.2-0)

Suggests: [boot](#), [dplyr](#), [effects](#), [foreign](#), [ineq](#), [KernSmooth](#), [lattice](#), [MASS](#), [mlogit](#), [nlme](#), [rnet](#), [np](#), [plm](#), [pscl](#)

Published: 2013-11-07

Author: Christian Kleiber [aut], Achim Zeileis [aut, cre]

Maintainer: Achim Zeileis <Achim.Zeileis@R-project.org>

License: [GPL-2](#)

NeedsCompilation: no

Citation: [AER citation info](#)

Materials: [NEWS](#)

In views: [Econometrics](#), [Survival](#), [TimeSeries](#)

CRAN checks: [AER results](#)

Downloads:

Reference manual: [AER.pdf](#)

Vignettes: [Applied Econometrics with R: Package Vignette and Errata](#)
[Sweave Example: Linear Regression for Economics Journals Data](#)

Package source: [AER_1.2-1.tar.gz](#)

MacOS X binary: [AER_1.2-1.tgz](#)

Windows binary: [AER_1.2-1.zip](#)

Old sources: [AER archive](#)

Reverse dependencies:

Reverse depends: [lpack](#), [rdd](#)

Reverse suggests: [censReg](#), [glmx](#), [lme4](#), [micEconCES](#), [mlogit](#), [plm](#), [REEMtree](#), [sandwich](#)

CRAN Task Views

CRAN Finance Task View

<http://cran.r-project.org/>

Note:

- Maintainer,
- Topics,
- List of packages.

← → ↺ cran.us.r-project.org



CRAN

[Mirrors](#)

[What's new?](#)

[Task Views](#)

[Search](#)

About R

[R Homepage](#)

[The R Journal](#)

Software

[R Sources](#)

[R Binaries](#)

[Packages](#)

[Other](#)

Documentation

[Manuals](#)

[FAQs](#)

[Contributed](#)

CRAN Task View: Empirical Finance

Maintainer: Dirk Eddelbuettel

Contact: Dirk Eddelbuettel at R-project.org

Version: 2014-01-16

This CRAN Task View contains a list of packages useful for empirical work in Finance,

Besides these packages, a very wide variety of functions suitable for empirical work in Finance are available in R packages on the Comprehensive R Archive Network (CRAN). Consequently, several of the following packages are also available on the [CRAN Task Views](#) for [Optimization](#), [Robust](#), [SocialSciences](#) and [TimeSeries](#) Task Views.

Please send suggestions for additions and extensions for this task view to the [task view maintainer](#).

Standard regression models

- A detailed overview of the available regression methodologies is provided by the [lm4](#) package.
- Linear models such as ordinary least squares (OLS) can be estimated by `lm()` (from the [stats](#) package) or `glm()` (from the [stats](#) package). Many other suitable methods are available, such as `nlme()` from the [nlme](#) package.
- For the linear model, a variety of regression diagnostic tests are provided by the [car](#) package, which may be of interest as well.

Time series

- A detailed overview of tools for time series analysis can be found in the [TimeSeries](#) Task View.
- Classical time series functionality is provided by the [arima\(\)](#) and [KalmanLike\(\)](#) functions.
- The [dse](#) and [limsac](#) packages provides a variety of more advanced estimation methods.
- For volatility modeling, the standard GARCH(1,1) model can be estimated with the [rugarch](#) package. The [rugarch](#) package can be used to model a variety of univariate GARCH processes. The [rugarch](#) package provides methods for fit, forecast, simulation, inference and plotting are provided too. The [rugarch](#) package can also estimate and simulate the Beta-t-EGARCH model by Harvey. The [havesGARCH](#) package provides a variety of GARCH models. The [ccgarch](#) package can estimate (multivariate) Conditional Correlation GARCH models. The [AutoSEARCH](#) package provides automated general-to-specific model selection.
- Unit root and cointegration tests are provided by [series](#), and [urca](#). The [Rmetrics](#) package provides unit roots and more. The [CADTest](#) package implements the Hansen unit root tests.
- [MSBVAR](#) provides Bayesian estimation of vector autoregressive models. The [dlm](#) package provides a variety of dynamic linear models.
- The [vars](#) package offer estimation, diagnostics, forecasting and error decomposition for vector autoregressive models.
- The [dyn](#) and [dynlm](#) are suitable for dynamic (linear) regression models.
- Several packages provide wavelet analysis functionality: [rwt](#), [wavelets](#), [waveslim](#), [wavelet](#), [waveletComp](#), [waveletR](#), [waveletTools](#), [waveletVis](#), [waveletVis2](#), [waveletVis3](#), [waveletVis4](#), [waveletVis5](#), [waveletVis6](#), [waveletVis7](#), [waveletVis8](#), [waveletVis9](#), [waveletVis10](#), [waveletVis11](#), [waveletVis12](#), [waveletVis13](#), [waveletVis14](#), [waveletVis15](#), [waveletVis16](#), [waveletVis17](#), [waveletVis18](#), [waveletVis19](#), [waveletVis20](#), [waveletVis21](#), [waveletVis22](#), [waveletVis23](#), [waveletVis24](#), [waveletVis25](#), [waveletVis26](#), [waveletVis27](#), [waveletVis28](#), [waveletVis29](#), [waveletVis30](#), [waveletVis31](#), [waveletVis32](#), [waveletVis33](#), [waveletVis34](#), [waveletVis35](#), [waveletVis36](#), [waveletVis37](#), [waveletVis38](#), [waveletVis39](#), [waveletVis40](#), [waveletVis41](#), [waveletVis42](#), [waveletVis43](#), [waveletVis44](#), [waveletVis45](#), [waveletVis46](#), [waveletVis47](#), [waveletVis48](#), [waveletVis49](#), [waveletVis50](#), [waveletVis51](#), [waveletVis52](#), [waveletVis53](#), [waveletVis54](#), [waveletVis55](#), [waveletVis56](#), [waveletVis57](#), [waveletVis58](#), [waveletVis59](#), [waveletVis60](#), [waveletVis61](#), [waveletVis62](#), [waveletVis63](#), [waveletVis64](#), [waveletVis65](#), [waveletVis66](#), [waveletVis67](#), [waveletVis68](#), [waveletVis69](#), [waveletVis70](#), [waveletVis71](#), [waveletVis72](#), [waveletVis73](#), [waveletVis74](#), [waveletVis75](#), [waveletVis76](#), [waveletVis77](#), [waveletVis78](#), [waveletVis79](#), [waveletVis80](#), [waveletVis81](#), [waveletVis82](#), [waveletVis83](#), [waveletVis84](#), [waveletVis85](#), [waveletVis86](#), [waveletVis87](#), [waveletVis88](#), [waveletVis89](#), [waveletVis90](#), [waveletVis91](#), [waveletVis92](#), [waveletVis93](#), [waveletVis94](#), [waveletVis95](#), [waveletVis96](#), [waveletVis97](#), [waveletVis98](#), [waveletVis99](#), [waveletVis100](#).

Installing Packages

Most packages need to be *installed* before they can be loaded and used.

Some packages like *MASS* are installed with base R (but not loaded).

Installing a package means downloading and saving its files to a local computer directory (hard disk), so they can be *loaded* by the R system.

The function `install.packages()` installs packages from the R command line.

Most widely used packages are available on the *CRAN* repository:

<http://cran.r-project.org/web/packages/>

Or on *R-Forge* or *GitHub*:

<https://r-forge.r-project.org/>

<https://github.com/>

Packages can also be installed in *RStudio* from the menu (go to **Tools** and then **Install packages**),

Packages residing on GitHub can be installed using the devtools packages.

```
> getOption("repos") # get default package source
> .libPaths() # get package save directory
> install.packages("AER") # install "AER" from CRAN
> # install "PerformanceAnalytics" from R-Forge
> install.packages(
+   pkgs="PerformanceAnalytics", # name
+   lib="C:/Users/Jerzy/Downloads", # directory
+   repos="http://R-Forge.R-project.org") # source
> # install devtools from CRAN
> install.packages("devtools")
> # load devtools
> library(devtools)
> # install package "babynamesv" from GitHub
> install_github(repo="hadley/babynamesv")
```

Installing Packages From Source

Sometimes packages aren't available in compiled form, so it's necessary to install them from their source code.

To install a package from source, the user needs to first install compilers and development tools:

For Windows install Rtools:

<https://cran.r-project.org/bin/windows/Rtools/>

For Mac OSX install XCode developer tools:

<https://developer.apple.com/xcode/downloads/>

The function `install.packages()` with argument `type="source"` installs a package from source.

The function `download.packages()` downloads the package's installation files (compressed tar format) to a local directory.

The function `install.packages()` can then be used to install the package from the downloaded files.

```
> # install package "PortfolioAnalytics" from source
> install.packages("PortfolioAnalytics",
+   type="source",
+   repos="http://r-forge.r-project.org")
> # download files for package "PortfolioAnalytics"
> download.packages(pkgs = "PortfolioAnalytics",
+   destdir = ".", # download to cwd
+   type = "source",
+   repos="http://r-forge.r-project.org")
> # install "PortfolioAnalytics" from local tar source
> install.packages(
+   "C:/Users/Jerzy/Downloads/PortfolioAnalytics_0.9.3598.tar.gz",
+   repos=NULL, type="source")
```

Installed Packages

`defaultPackages` contains a list of packages loaded on startup by default.

The function `installed.packages()` returns a matrix of all packages installed on the system.

```
> getOption("defaultPackages")
> # matrix of installed package information
> pack_info <- installed.packages()
> dim(pack_info)
> # get all installed package names
> sort(unname(pack_info[, "Package"]))
> # get a few package names and their versions
> pack_info[sample(x=1:100, 5), c("Package", "Version")]
> # get info for package "xts"
> t(pack_info["xts", ])
```


Package Files and Directories

Package installation files are organized into multiple directories, including some of the following:

- `~/R` containing R source code files,
- `~/src` containing C++ and Fortran source code files,
- `~/data` containing datasets,
- `~/man` containing documentation files,

```
> # list directories in "PortfolioAnalytics" sub-directory
> gsub(
+   "C:/Users/Jerzy/Documents/R/win-library/3.1",
+   "~",
+   list.dirs(
+     file.path(
+       .libPaths()[1],
+       "PortfolioAnalytics")))
[1] "/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/1"
[2] "/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/2"
[3] "/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/3"
[4] "/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/4"
[5] "/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/5"
[6] "/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/6"
[7] "/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/7"
[8] "/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/8"
[9] "/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/9"
[10] "/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/10"
[11] "/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/11"
[12] "/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/12"
[13] "/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/13"
[14] "/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/14"
```

Loading Packages

Most packages need to be *loaded* before they can be used in an R session.

Loading a package means attaching the package *namespace* to the *search path*, which allows R to call the package functions and data.

The functions `library()` and `require()` load packages, but in slightly different ways.

`library()` produces an *error* (halts execution) if the package can't be loaded.

`require()` returns `TRUE` if the package is loaded successfully, and `FALSE` otherwise.

Therefore `library()` is usually used in script files that might be sourced, while `require()` is used inside functions.

```
> # load package, produce error if can't be loaded
> library(MASS)
> # load package, return TRUE if loaded successfully
> require(MASS)
> # load quietly
> library(MASS, quietly=TRUE)
> # load without any messages
> suppressMessages(library(MASS))
> # remove package from search path
> detach(MASS)
> # install package if it can't be loaded successfully
> if (!require("xts")) install.packages("xts")
```

Referencing Package Objects

After a package is *loaded*, the package functions and data can be accessed by name.

Package objects can also be accessed without *loading* the package, by using the double-colon ":" reference operator.

For example, `TTR::VWAP()` references the function `VWAP()` from the package `TTR`.

This way users don't have to load the package `TTR` (with `library(TTR)`) to use functions from the package `TTR`.

Using the ":" operator displays the source of objects, and makes R code easier to analyze.

```
> # calculate VTI volume-weighted average price
> vwapv <- TTR::VWAP(
+   price=quantmod::Cl(rutils::etfenv$VTI),
+   volume=quantmod::Vo(rutils::etfenv$VTI), n=10)
```

Exploring Packages

The package *Ecdat* contains data sets for econometric analysis.

The data frame *Garch* contains daily currency prices.

The function `data()` loads external data or listv data sets in a package.

Some packages provide *lazy loading* of their data sets, which means they automatically load their data sets when they're needed (when they are called by some operation).

The package's data isn't loaded into R memory when the package is *loaded*, so it's not listed using `ls()`, but the package data is available without calling the function `data()`.

The function `data()` isn't required to load data sets that are set up for *lazy loading*.

```
> library() # list all packages installed on the system
> search() # list all loaded packages on search path
>
> # get documentation for package "Ecdat"
> packageDescription("Ecdat") # get short description
> help(package="Ecdat") # load help page
> library(Ecdat) # load package "Ecdat"
> data(package="Ecdat") # list all datasets in "Ecdat"
> ls("package:Ecdat") # list all objects in "Ecdat"
> browseVignettes("Ecdat") # view package vignette
> detach("package:Ecdat") # remove Ecdat from search path
```

```
> library(Ecdat) # load econometric data sets
> class(Garch) # Garch is a data frame from "Ecdat"
> dim(Garch) # daily currency prices
> head(Garch[, -2]) # col 'dm' is Deutsch Mark
> detach("package:Ecdat") # remove Ecdat from search path
```

Package Namespaces

Package *namespaces*:

- Provide a mechanism for calling objects from a package,
- Hide functions and data internal to the package,
- Prevent naming conflicts between user and package names,

When a package is loaded using `library()` or `require()`, its *namespace* is attached to the search path.

```
> search() # get search path for R objects
> library(MASS) # load package "MASS"
> head(ls("package:MASS")) # list some objects in "MASS"
> detach("package:MASS") # remove "MASS" from search path
```

Package Namespaces and the Search Path

Packages may be loaded without their *namespace* being attached to the search path.

When packages are loaded, then packages they depend on are also loaded, but their *namespaces* aren't necessarily attached to the search path.

The function `loadedNamespaces()` lists all loaded *namespaces*, including those that aren't on the search path.

The function `search()` returns the current search path for R objects.

`search()` returns many package *namespaces*, but not all the loaded *namespaces*.

```
> loadedNamespaces() # get names of loaded namespaces
>
> search() # get search path for R objects
```

Not Attached Namespaces

The function `sessionInfo()` returns information about the current R session, including packages that are loaded, but *not attached* to the search path.

`sessionInfo()` lists those packages as "loaded via a *namespace* (and not attached)"

```
> # get session info,  
> # including packages not attached to the search path  
> sessionInfo()
```

Non-Visible Objects

Non-visible objects (variables or functions) are either:

- objects from *not attached namespaces*,
- objects *not exported* outside a package,

Objects from packages that aren't attached can be accessed using the double-colon ":" reference operator.

Objects that are *not exported* outside a package can be accessed using the triple-colon ":::" reference operator.

Colon operators automatically load the associated package.

Non-visible objects in namespaces often use the ".*" name syntax.

```
> plot.xts # package xts isn't loaded and attached
> head(xts::plot.xts, 3)
> methods("cbind") # get all methods for function "cbind"
> stats::cbind.ts # cbind isn't exported from package stats
> stats:::cbind.ts # view the non-visible function
> getAnywhere("cbind.ts")
> library(MASS) # load package 'MASS'
> select # code of primitive function from package 'MASS'
```


Exploring Namespaces and Non-Visible Objects

The function `getAnywhere()` displays information about R objects, including non-visible objects.

Objects referenced *within* packages have different search paths than other objects:

Their search path starts in the package *namespace*, then the global environment and then finally the regular search path.

This way references to objects from *within* a package are resolved to the package, and they're not masked by objects of the same name in other environments.

```
> getAnywhere("cbind.ts")
```