NCAR

Introduction to R

Siparcs

20 May 2015

Eric Gilleland

http://www.ral.ucar.edu/staff/ericg

National Center for Atmospheric Research

R's website

[Home]

Download

CRAN

R Project

About R
Contributors
What's New?
Mailing Lists
Bug Tracking
Conferences
Search

R Foundation

Foundation
Board
Members
Donors
Donate

Documentation

Manuals
FAQs

The R Project for Statistical Computing

Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To **download R**, please choose your preferred CRAN mirror.

If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

News

- **R version 3.2.0** (Full of Ingredients) has been released on 2015-04-16.
- **R version 3.1.3** (Smooth Sidewalk) has been released on 2015-03-09.
- **The R Journal Volume 6/2** is available.
- **useR! 2015**, will take place at the University of Aalborg, Denmark, June 30 - July 3, 2015.
- **useR! 2014**, took place at the University of California, Los Angeles, USA June 30 - July 3, 2014.

http://www.R-project.org

Some advanced topics relevant to Atmospheric Sciences

- Reading and writing Net CDF files

http://www.image.ucar.edu/Software/Netcdf

- A climate related precipitation example

http://www.image.ucar.edu/~nychka/FrontrangePrecip/

- Extreme Value Analysis Applied to Weather and Climate

http://www.ral.ucar.edu/staff/ericg/extRemes/extRemes2.pdf

- Forecast verification and CI's

http://opensky.library.ucar.edu/collections/TECH-NOTE-000-000-000-846

How R operates

- A function is implemented from the R prompt using the function name and () even if no arguments are passed.
  - For example, to see the current date and time:

```
Sys.time()
```

- A (visible) function's source code is listed with just the name at the R prompt.
  - For example, to see the source code for the Sys.time function:

```
Sys.time
```

How R operates

NCAR

- All work is performed in a “dot” file called .RData, referred to as the workspace.
- .RData exists in the working directory.
- getwd() will show the path for this directory (see also setwd).
- Use save.image() to save your work.
- q() to quit an R session (will prompt you to save the directory or not). Use q(“yes”) to avoid the prompt and save, or q(“no”) to avoid the prompt and not save.

Packages

NCAR

- Thousands of user-supplied packages freely available on CRAN, and accessible within an R session.
- Installing a package:
  - install.packages(“package name”)
- Updating already installed packages:
  - update.packages()
- Loading an installed package:
  - library(“package name”)

Some Atmospheric Science Oriented Packages

- Spatial Statistics
  - fields, LatticeKrig
  - spatstat
  - sp
  - inla
- Maps
  - maps
- Extreme Value Analysis
  - extRemes
  - SpatialExtremes
  - See also: http://www.ral.ucar.edu/staff/ericg/softextreme.php
- Forecast Verification
  - verification
  - SpatialVx
- Handling NetCDF files:
  - ncdf4, ncdf, netCDF
- Skew-T plots
  - RadioSonde

Citing R

- To find out how to cite R itself in a paper, from the R prompt, use:

citation()

- To find out how to cite a user-supplied package, e.g., the package called fields, use:

citation("fields")

Getting help

NCAR

- Various search engines and documentation found on the R project web site.
- Within an R session, you can use the help function, or the ? short cut to learn about most functions, packages and data sets. Examples:
  - ?fields
  - ?Sys.time
- To create a help file template for your own function, package or data set, use the prompt function.

Object oriented

NCAR

- Both S3 and S4 methods can be used.
- plot, summary, and print are three very standard *generic* functions that can be used for most R objects.
- methods(plot)
- methods(class = “lm”)

Logical Operators

- and: &, &&
  - use ?"&" to get help
- or: |, ||
- equal: ==
- >=, >, <, <=
- !=
- is.na
- is.finite
- is.numeric
- is.logical
- is.element
- many more

Other Operators

- Arithmetic operators:
  - +, -, *, /
  - ^, ** (power operators, x^2 or x**2 give x^2)
- Matrix operations:
  - multiplication: %*%
    - Use ?"%*%" to get help.
  - transpose: ?t
- Assignment operators
  - <-
    - e.g., tiid <- Sys.time()
  - ->
    - e.g., Sys.time() -> tiid
  - ?assign
  - =

R objects and classes

NCAR

- Atomic objects (all components of same mode: numeric, complex, logical, character or raw)
  - vectors
- Recursive objects
  - lists
  - functions
  - expressions
- Classes
  - data frames
  - matrices
  - arrays
  - other …

Examples

```
x <- numeric(0)
mode( x )
class( x )
length( x )
x
x[ 1 ]
any( is.na( x ) )
x <- c( x, 3 )
length( x )
x[ 1 ]
x <- c( x, rnorm( 99, mean = 3, sd = 25 ) )
x
length( x )
```

Examples

```
plot( x )

summary( x )

x <- c("Mary", x )
class( x )
mode( x )
length( x )

x[ x == "Mary" ] <- NA
mode( x )
x <- as.numeric( x )
x <- x[ !is.na( x ) ]
```

Examples

```
x[ 2:4 ]
x[ x < 0 ] <- 0

plot( x, type = “l”, col = “darkblue” )

x <- rexp( length( x ) )

y <- rgamma( 100, shape = 1 )
hist( y )

dat <- data.frame( x = x, y = y, z = rnorm( 100 ) )
```

Examples

```
names( dat )
colnames( dat )
rownames( dat )

class( dat )
mode( dat )

plot( y ~ x + z, data = dat )

par( mfrow = c(1, 2 ) )
plot( y ~ x + z, data = dat )

plot( dat )
```

Examples

```
dim( dat )
dim( t( dat ) )

summary( dat )

dat[1:3, 2] <- 0

dat$w <- dat$x * dat$y

plot( dat )

par( mfrow = c(1,1) )
boxplot( dat )
```

Examples

```
dat[[ “x” ]]

dat$x

dat[, 1]

dat[, “x” ]

dat[, -(2:dim( dat)[ 2 ]) ]

dat[1:10, ]

dat[ 1:10, “x”]
```

Examples

```
boxplot( dat, notch = TRUE, col = “darkred” )

dat$ind <- rep( 1:10, each = 10 )

X <- as.matrix( dat )
class( X )
mode( X )
dim( X )
names( X )
colnames( X )

X[ , “ind” ]

plot( X )
```

Examples

```
colnames( X ) <- NULL

plot( X )

diag( X )

Y <- cbind( 1:100, 0, rnorm( 100 ), NA, rexp( 100 ) )

X <- t( X ) %*% Y

dim( X )
X
summary( X )
```

Examples

```
Y <- X[ -c(2,3), -1]

sum( is.na( X ) )
colSums( X )
rowSums( X )

sum( X, na.rm = TRUE )
mean( X )
mean( X, na.rm = TRUE )

dat$lab <- rep( c("Moanne", "Tiiwusdei", "Woansdei",
    "Tongersdei", "Freed", "Sneon", "Snein", "Sko",
    "Sinne", "Hjerst" ), each = 10 )
```

Examples

NCAR

```
?par

plot( y ~ x, data = dat, xlab = “Example data”,
    ylab = “Response”, pch = 22, cex = 2, bg = “lightblue”,
    col = “darkblue”, cex.lab = 2,
    main = “Example”, cex.main = 2 )

plot( x ~ ind, data = dat, xaxt = “n”, pch = 23,
    bg = “lightblue”, col = “darkblue”, cex = 1.5 )

axis( 1, at = unique( dat$ind ), labels = unique( dat$lab ) )
```

Examples

NCAR

```
plot( x ~ ind, data = dat, xaxt = “n”, pch = 23,
    bg = “lightblue”, col = “darkblue”, cex = 1.5,
    xlab = “” )

axis( 1, at = unique( dat$ind ), labels = unique( dat$lab ),
    las = 2 )
```

More plotting features

NCAR

See also:

- ?lines, ?points (for adding to an existing plot)
  - use “lty” argument for different line types
  - use “lwd” argument for different line widths
- ?legend (add a legend to a plot)
- ?title, ?mtext
- Use “xlim” and “ylim” to set the plotting region.
- Use “mar”, “oma”, etc. in call to “par” to change inner and outer margin areas.
- To write a plot to a file, see ?pdf, ?png, ?jpeg
- ?dev.new, ?dev.off

Avoiding loops in R

NCAR

- Use of logical vectors
- colSums, colMeans, rowSums, rowMeans
- apply, lapply, sapply

Avoiding loops in R

```
laundry <- list()

laundry[[ 1 ]] <- rbind( 1:3, c(4, -1, 0) )
laundry$clean <- rexp( 100 )

laundry

lapply( laundry, summary )

summary( laundry )
```

Avoiding loops in R

```
laundry <- list()

laundry[[ 1 ]] <- rbind( 1:3, c(4, -1, 0) )
laundry$clean <- rexp( 100 )

laundry

lapply( laundry, summary )

summary( laundry )
```

Avoiding loops in R

NCAR

```
apply( X, 2, max )

apply( X, 2, max, na.rm = TRUE )

apply( X, 1, min )

apply( X, 1, min, na.rm = TRUE )

apply( X, 2, range )

apply( X, 2, range, finite = TRUE )
```

Avoiding loops in R

NCAR

```
Z <- array( NA, dim = c(3, 4, 2) )

class( Z )
mode( Z )
dim( Z )

Z[, , 1 ] <- cbind( c(1, 0, -1), rep(1, 3), 1:3, 4 )
Z[, , 2 ] <- rbind( 1:4, c(-1, 0, 1, 0), 0 )

Z[, , 1, drop = FALSE ]
```

Avoiding loops in R

NCAR

```
apply( Z, 1:2, sum )

apply( Z, 3, sum )

Z[1,2,1] <- NA

apply( Z, 1:2, sum )

apply( Z, 1:2, sum, na.rm = TRUE )
```

Writing Functions in R

NCAR

```
summy <- function( x, y ) (x + y) / (x – y)

summy( 4, 2 )

summy( 1:10, c(-1, -2, 0, 7:1) )

summy(1, 1)
```

Writing Functions in R

NCAR

```
summy <- function(x, y, … ) {

    cat("Hello, I am summy.\n")

    mx <- mean( x, … )
    my <- mean( y, … )
    n <- x^2 + y^2

    return( list( mx = mx, my = my, n = n ) )

}
```

Writing Functions in R

summy(4, 2)

summy(X, 1)

summy(X, 1, na.rm = TRUE)

Writing Functions in R

NCAR

Writing a generic function

```
summy <- function(x, … ) {
    UseMethod(“summy”)
}
summy( X )
```

Writing Functions in R

NCAR

Writing a method function

```
summy.default <- function(x, … ) {

    res <- sum( x, …)

    class( res ) <- “summied”

    invisible( res )
}
```

Writing Functions in R

NCAR

Writing a method function

```
methods( summy )

summy( X )

x2 <- summy( X, na.rm = TRUE )

x2

class( x2 )
mode( x2 )
```

Writing Functions in R

NCAR

Writing a method function

```
summy.matrix <- function(x, … ) {
    res <- colSums( x, … )
    class( res ) <- “summied”
    invisible( res )
}
```

Writing Functions in R

NCAR

Writing a method function

```
x3 <- summy( X, na.rm = TRUE )

x3
x2
sum( x3 )
```

Writing Functions in R

NCAR

Writing a method function

```
print.summied <- function( x, … ) {

    cat(“Hello, \n”)
    cat(“and thank you for using the summy function.\n\n”)

    print(c(x))

    invisible()

}
```

Writing Functions in R

NCAR

Writing a method function

x2

x3

Writing Functions in R

NCAR

```
summy.matrix <- function(x, … ) {

    theCall <- match.call()

    res <- colSums( x, … )

    class( res ) <- “summied”

    attr( res, “call” ) <- theCall

    invisible( res )

}
```

Writing Functions in R

NCAR

```
print.summied <- function( x, … ) {

    cat("Hello, \n")
    cat("and thank you for using the summy function.\n\n")

    a <- attributes( x )
    if( !is.null( a$call ) ) print( a$call )

    print(c(x))

    invisible()

}
```

Writing Functions in R

NCAR

```
x4 <- summy( rnorm( 10 ) )

x5 <- summy( X )

x4

x5
```

S4 Methods

Everything I've showed so far is an S3 method. I know less about S4 methods, but see:

http://cran.r-project.org/doc/contrib/Genolini-S4tutorialV0-5en.pdf

Statistical Models

NCAR

- Base/Stats packages
  - Linear Regression/ANOVA
    - ?lm
  - General linear models
    - ?glm
  - Time series (see also ?lm)
    - ?acf, ?pacf
    - arima, arima.sim
  - Nonlinear least squares
    - ?nls

Also note the method function for making predictions from these models.
?predict

Other

NCAR

- package: boot
  - ?boot, ?tsboot, ?boot.ci
- package: cluster, fastcluster
  - ?hclust, many more …
- package: fields, LatticeKrig
  - ?Krig, ?Tps, ?mKrig, ?LKrig
  - ?image.plot, ?poly.image
- package: ggplot2

Reading Data into R

NCAR

- ?scan
- ?read.table
- ?read.csv
- ?read.fwf
- ?source

Writing data from R to computer

NCAR

- ?write
- ?write.table
- ?dump

Attaching a workspace

Because .RData can become too large if you're using large data sets, it can be more efficient to attach other R workspaces that contain the data. See ?attach, ?detach and ?search to learn more.