# Course on R: Glossary

Elizabeth Ribble*

18-24 May 2017

*emcclel3@msudenver.edu

# Contents

# 1 Getting Help

```
?, help()       # Open the built-in html help file for a function
                # or operator (use quotes for special characters)
??              # Search the built-in html help files for a
                # keyword or phrase (in quotes)
RSiteSearch()   # Search the online R documentation for a keyword
                # or phrase (in quotes)
apropos()       # Return character vector of objects containing
                # searched character string
```

# 2 The R Workspace

```
objects(), ls()     # List the objects in the Workspace
rm()                # Remove objects from the Workspace
getwd()             # Determine the current working directory
setwd()             # Change the current working directory
save.image()        # Save all objects in session to a .RData file
savehistory()       # save all commands executed during current
                    # session to a .Rhistory file
save()              # save specified objects
load()              # load a .Rdata file into session
loadhistory()       # load a .Rhistory file into session
```

# 3 Reading and Writing Files

```
read.table(),   # Read data from a text file into a data frame, or
write.table()   # Write data from a data frame or matrix to a
                # text file
read.csv()      # Read data from a comma separated value
                # (.csv) file into a data frame
write.csv()     # Write data from a data frame or matrix to a
                # .csv file
source()        # Read and execute R commands from a file
```

# 4 Special Values, Operators, and Functions

```
<-                  # Assigns a value to a variable
#                   # Comment (not executed if ran)
NULL                # Represents an "empty" variable
args()              # View formal arguments of a function
Inf                 # Infinity
NaN                 # "Not a number"
NA                  # "Not available" (a missing value)
table()             # Create a contingency table of counts
                    # from a factor or character vector
class()             # Determines the class of an object
mode()              # determines the mode of an object
library()           # Loads a package into the current session
trace.back()        # Trace code to source of an error
set.seed()          # Fix random number seed (for reproducibility)
install.packages()  # Install specified package
data()              # Load pre-installed dataset
paste()             # Paste characters into a single string
letters             # character vector of lowercase letters
LETTERS             # character vector of uppercase letters
```

# 5 Arithmetic Operators

```
^                       # Exponentiation
%%                      # Modulo (i.e. remainder)
%/%                     # Integer divide
*  /                    # Multiplication, Division
+  -                    # Addition, Subtraction
```

# 6 A Few Built-in Mathematical Functions

```
sqrt()                  # Square root
abs()                   # Absolute value
sign()                  # Returns -1, 0, or +1 depending on
                        # whether its argument is negative,
                        # zero, or positive
round()                 # Round a value to a specified number
                        # of digits
signif()                # Express a value to a specified number of
                        # significant digits
floor()                 # Largest integer not greater than a value
ceiling()               # Smallest integer not less than a value
trunc()                 # Truncate a value toward 0
log(); log10()          # Natural logarithm, base 10 logarithm
exp()                   # Exponential function (e.g. exp(1) is
                        # the exponential constant e)
factorial()             # Factorial
choose()                # Number of ways to choose x objects from
                        # n objects
sin(); cos(); tan()     # Sine, cosine, tangent
beta(); gamma()         # Beta function, gamma function
```

# 7 Relational and Logical Operators

```
<               # Less than
>               # Greater than
==              # Equal to
!=              # Not equal to
<=              # Less than or equal to
>=              # Greater than or equal to
&&              # "And" for logical scalars
||              # "Or" for logical scalars
!               # "Not" (for logical scalars or vectors)
&               # "And" for logical vectors
|               # "Or" for logical vectors
```

# 8 User-Defined Functions

```
myFunc <- function(arg1, arg2, ..., argk) {
   statement1
   statement2
       .                                    # General form for a
       .                                    # function definition
       .
   statementq
   return(expression)
}
```

Above,

- `myFunc` is a name we choose for our function.
- `arg1`, `arg2`, ..., `argk` are names we choose for the $k$ formal arguments.
- `statement1`, `statement2`, ..., `statementq` are a sequence of $q$ statements (which may involve `arg1`, `arg2`, ..., `argk`).
- `expression` is evaluated and the result returned.

# 9 Vectors

```
c()           # Create a vector of values
length()      # Returns the number of elements in a vector
              # (sample size)
is.vector()   # Indicates whether or not an object is a vector
[ ]           # Access vector elements via their indices OR
              # extract a subset of vector elements that satisfy
              # a given condition (e.g. relational)
sort()        # Returns the data in sorted order
rev()         # Returns the data in reverse order
order()       # Returns a vector of indices such that x[order(x)]
              # returns the vector x in sorted order
seq()         # Create a sequence of values
:             # Create a sequence of integers
rep()         # Create a repeating pattern of values
any()         # Do any elements of a vector satisfy a
              # given condition?
all()         # Do all the elements of a vector satisfy a
              # given condition?
which()       # Returns the indices of the elements of a vector
              # that satisfy a given condition
which.min(),  # Returns the index of the minimum (or maximum) value
which.max()   # in a vector
%in%          # Return TRUE if given value(s) contained in
              # given vector
match()       # Return position of first argument if it appears in
              # second argument (only first match returned)
duplicated()  # Return TRUE if element appears in an
              # earlier position in vector
is.na()       # Returns TRUE or FALSE depending on
              # whether or not a value is NA
sample()      # Randomly permute or sample from a vector
cut()         # Convert a numeric vector into a factor based
              # on specified cut points
```

# 10 Factors

```
factor()          # Create a factor from a character vector
length()          # Returns the number of elements in a factor
                  # (sample size)
levels()          # Examine the levels of the factor
nlevels()         # Returns the number of levels of the factor
is.factor()       # Indicates whether or not an object is a factor
str()             # Describes the structure of a factor
as.character()    # Convert a factor to a character vector
as.numeric()      # Convert a factor to a numeric vector
```

# 11 Matrices (special vectors!)

```
matrix()            # Create a matrix, from a vector, with nrow
                    # rows and ncol columns
cbind(), rbind()    # Create a matrix from two or more vectors by
                    # "binding" them together in columns or rows
dim()               # Returns the dimensions (number of rows and
                    # columns) of a matrix
nrow(); ncol()      # Number of rows, number of columns of a matrix
is.matrix()         # Indicates whether an object is a matrix
[ , ]               # Access matrix elements via their row and
                    # column indices (separated by a comma)
rownames()          # View or change the row names of a matrix
colnames()          # View or change the column names of a matrix
apply()             # Apply a function separately to each row (or
                    # each column) of a matrix
```

# 12 Lists

```
list()          # Create a list
length()        # Returns the number of elements in a list
is.list()       # Indicates whether or not an object is a list
str()           # Describes the structure of a list
[[ ]]           # Retrieve a list element via its index or name
$               # Retrieve a list element via its name
[ ]             # Access a list element via its index or name,
                # returning a list
unlist()        # Convert the components of a list to a vector
lapply()        # Apply a function separately to each element of
                # a list, returning a list
sapply()        # Apply a function separately to each element of
                # a list, returning a vector
names()         # Examine or change the names of list elements
```

# 13 Data Frames (special lists!)

```
data.frame()      # Create a data frame from a set of vectors of
                  # the same length
read.table()      # Read data from a text file into a data frame
head()            # Prints the first 6 rows of a data frame
tail()            # Prints the last 6 rows of a data frame
names()           # Lists the names of the variables in the data
                  # frame
row.names()       # View or change row names of a data frame
nrow()            # Indicates the number of rows of a data frame
ncol()            # Indicates the number of columns of a data frame
dim()             # Gives the dimensions (number of rows and
                  # columns)
str()             # Gives the structure of a data frame
is.data.frame()   # Indicates whether or not an object is a data
                  # frame
[ , ]             # Access data frame elements, rows, or columns
                  # via their row and column indices (separated by
                  # a comma)
[[ ]]             # Retrieve a data frame variable (column) by
                  # specifying its index or name
$                 # Retrieve a data frame variable (column) by
                  # specifying its name
rbind()           # Create a new data frame by "binding" the
                  # rows of one data frame to those of another
cbind()           # Create a new data frame by "binding" the
                  # columns of one data frame to those of another
merge()           # Merge two data frames that share one or more
                  # variables in common
unique()          # Return only unique rows of a data frame
duplicated()      # Return TRUE if row appears in an
                  # earlier row in data frame
attach()          # Make the variables in a data frame directly
                  # accessible by name by "attaching" the
                  # data frame
detach()          # "Detach" a data frame that is "attached"
```

# 14 Plots

```
plot()              # Scatterplot, time-series plot
hist()              # Histogram
boxplot()           # Boxplot(s)
stripchart()        # Dot plot, individual value plot
qqnorm()            # Normal probability plot
                    # (quantile-quantile plot)
stem()              # Stem and leaf plot
barplot()           # Bar chart of given bar heights
pie()               # Pie chart of given pie areas
pdf(), png(),       # Save plot to file with given extension
jpeg(), etc.        # (follow plotting commands with dev.off())
par()               # Function to set graphical parameters
colors()            # View a list of names of available colors
```

# 15 Plotting Arguments

```
main           # Main title for the plot (in quotation marks)
sub            # Subtitle (in quotation marks)
xlab, ylab     # Labels for the x and y axes (in quotation marks)
xlim, ylim     # Limits for the x and y axes in the plot (in the
               # form c(lower, upper))
type           # Type of plot that should be drawn (e.g. points,
               # lines, etc.)
pch            # Plot character, or symbol type
cex            # Character expansion factor, i.e. size of plot
               # characters and/or text
lty, lwd       # Line type (e.g. "dashed" or "solid") and line
               # width (values greater than 1 increase the width)
col            # Color of the objects being plotted (in quotation
               # marks)
col.axis,
col.lab,       # Colors for the axes, axis labels, main title,
col.main       # and subtitle
col.sub
bg, fg         # Background and foreground colors (in quotation
               # marks)
mfrow, mfcol   # Multiple-figure plot arrangement as an nrow by
               # ncol array (a numerical vector of the form
               # c(nrow, ncol)) - only for par()
xaxt, yaxt     # x and y axis types (specify "n" for no axis)
bty            # Type of box drawn around the plot (specify "n"
               # for none)
cex.main,      # Character expansion factors, i.e. sizes of text,
cex.axis,      # for main title, axis annotations, and axis labels
cex.lab        # (values greater than 1 increase their sizes)
```

# 16 Add to Existing Plot

```
points()      # Add points to the plot at specified coordinates
symbols()     # Add various symbols to the plot (circles, squares,
              # etc.)
abline()      # Add a line to the plot with given intercept a and
              # slope b
lines()       # Add a line to the plot connecting specified
              # coordinates
segments()    # Add line segments to the plot between pairs of
              # points
curve()       # Add a curve to the plot (specify add = TRUE)
qqline()      # Add a line to a normal probability plot
arrows()      # Draw an arrow in the plot (with specified start
              # and end points)
rug()         # Adds a "rug" (tick marks for observations) to plot
title()       # Add a main title to the plot (if it does not
              # already have one).  Can also be used to add x
              # and y axis labels.
text()        # Add text to the plot at a specified set of
              # coordinates
mtext()       # Add text in a margin of the plot
legend()      # Add a legend to the plot
axis()        # Add an axis to the plot on a given side
box()         # Add a box around the plot (if one does not
              # already exist)
rect()        # Draw a rectangle in the plot at a given set
              # of coordinates
polygon()     # Draw a polygon in the plot with a given set
              # of vertices
```

# 17 Summary Statistics Functions

```
mean()          # The sample mean
median()        # Sample median
sd(); var()     # Sample standard deviation and variance
length()        # Number of observations (sample size)
sum()           # The sum of the values
min(); max()    # Smallest and largest values in the data set
range()         # Range (smallest and largest values)
                # of the data set
mad()           # Median absolute deviation
quantile()      # Sample quantile (percentile)
IQR()           # Interquartile range
summary()       # Five number summary (and sample mean)
lapply()        # Apply a function separately to each column of a
                # data frame, returning a list
sapply()        # Apply a function separately to each column of a
                # data frame, returning a vector
tapply()        # Compute a specified summary statistic separately
                # for each subset of a data frame via the arguments
                # X, INDEX, and FUN
aggregate()     # Compute a specified summary statistic separately
                # for each subset of a data frame via the arguments
                # x, by, and FUN
colMeans()      # Compute the mean separately for each column of a
                # data frame
colSums()       # Compute the sum separately for each column of a
                # data frame
rank()          # Assign ranks to values of a numeric vector
cor()           # Calculate Pearson"s correlation coefficient or
                # Spearman"s rho
```

# 18 Tests and Confidence Intervals

```
t.test()          # One- or two-sample (nonpooled or paired) t-test
                  # (assumes normality of each population or paired
                  # differences)
wilcox.test()     # One-sample signed rank test or two-sample
                  # Wilcoxon rank sum (a.k.a. Mann-Whitney U) test
                  # (no normality assumption)
cor.test()        # Test for the association between two variables
                  # (assumes normality of two populations)
aov()             # Perform ANOVA for differing means between two or
                  # more groups of a factor (assumes normality
                  # within groups)
summary()         # View the ANOVA table (F-test results from aov())
TukeyHSD()        # Calculate post-hoc pairwise confidence intervals
kruskal.test()    # Perform distribution-free test for differing
                  # locations between two or more groups of a factor
                  # (no normality assumption)
friedman.test()   # Perform distribution-free test for differing
                  # locations between two or more groups of a factor
                  # with a block variable (no normality assumption)
chisq.test()      # Perform chi-squared test of independence
fisher.test()     # Perform Fisher"s exact test of independence
```

# 19 Regression

```
lm()              # Fit a linear model (simple or multiple linear
                  # regression)
glm()             # Fit generalized linear models; fits a logistic
                  # regression model when family = binomial(logit)
summary()         # View results of models fit
predict()         # Predict from results of models fit
```

# 20 Programming

```
if()            # Used to execute a statement only if the given
                # condition is met
else            # Used to specify an alternative statement to be
                # executed if the condition given in if() is not met
ifelse()        # Returns a vector whose values depend on whether
                # or not a given condition is met by the elements
                # of another vector
for()           # Repeat set of statements a certain number of times
while()         # Repeat a set of statements as long as a
                # specified condition is met
repeat          # Repeat a set of statements until a break
                # command is encountered
break           # Terminate iterations of a loop
next            # Skip ahead to the next iteration
<<-             # Assign a value to a variable in the global
                # environment (Workspace).
assign()        # Assign a value to a variable in the global
                # environment (Workspace).
return()        # Terminate a function call and return a value.
stop()          # Terminate a function call; print error message.
warning()       # Print a warning message (without terminating the
                # function call).
source()        # Read R commands from a text file.
class()         # Determines the class of an object.  Can also be
                # used to assign a class to an object.
is.cname()      # Returns TRUE if an object belongs to the class
                # "cname" (e.g. is.numeric(), is.data.frame(),
                # etc.) and FALSE otherwise.
methods()       # Determine the S3 methods that are associated
                # with a given generic function
showMethods()   # Determine the S4 methods that are associated
                # with a given generic function
system.time()   # Returns the computation time required to
                # execute a chunk of R code (in seconds)
cmpfun()        # Translate/compile function from R code to bytecode
```