List of R Functions

September 2020 (R-Bootcamp FS20)

Pascal Himmelberger

Function	Description	Examples	Links
rbind()	rowbind, binds a list of rows together to form a matrix	rbind(c(1,2,3), c(1,2,3), c(1,2,3)) ==> 3x3 matrix with first column = 1, second column = 2, third column = 3	
cbind()	columnbind, a la rbind() but with columns	cbind(c(1,1,1), c(2,2,2), c(3,3,3)) ==> 3x3 matrix with first column = 1, second column = 2, third column = 3	
matrix(data=NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)	creates a matrix provided data and the expected dimensions. Default fills by column. If no data is provided, creates a 1x1 NA matrix.	matrix(letters[1:4], ncol=6, nrow=4) ==> 4x6 matrix with all 'a' in first row, all 'b' in second etc.	
ls()	lists all objects in the current environment	simply *ls()*	
rep(x, times = 1, length.out = NA, each = 1)	repeates value/list given as first argument. Output length, number of repetitions as well as how often each character is repeated can be specified.	rep(c('A','B','C','Z'), times=1:4) ==> "A" "B" "B" "C" "C" "C" "Z" "Z" "Z" "Z"	
diag(x = 'string'/number, nrow, ncol)	Creates a matrix with provided string/number 'x' on the diagonal.	diag(x = 999, nrow = 5, ncol = 5) [,1] [,2] [,3] [,4] [,5] [1,] 999 0 0 0 0 [2,] 0 999 0 0 0 [3,] 0 0 999 0 0 [4,] 0 0 0 999 0 [5,] 0 0 0 0 999	
data.frame(data, row.names = NULL, check.rows = FALSE, check.names = TRUE, fix.empty.names = TRUE, stringAsFactors = default.stringAsFactors())	creates a data frame based on the data provided. Data can be provided by specifying 'tag = value'. The tag will become the column name. Row names (Ids) can be specified using *row.names* argument	t.num <- 1:10 t.alph <- LETTERS[1:10] t.logical <- sample(x = c(TRUE, FALSE), size = 10, replace = TRUE) data.frame('Numbers' = t.num, 'Alphabest' = t.alph, 'Log' = t.logical, row.names = paste('case ', 1:length(t.alph)))	
colnames(data.frame)	Shows and allows assignment of column names of a data frame	colnames(d.test) colnames(d.test) [1] <- 'First Column'	
rownames(data.frame)	Shows and allows assignment of row names (default = Ids) of a data frame	row.names(d.2_again) = paste("case", 1:10)	
nrow()	Number of rows of a data object		
[R Integrated Data Set] LETTERS	Contains all letters of the alphabet	LETTER[1] # = 'A'	
View()	Allows to show tables/matrices/data.frames outside of the console	View(d.jobs) # given d.jobs is a compatible data structure	
seq(from = 1, to = 1, by = ((to - from)/(length.out - 1)), length.out = NULL)	Generates a sequence of numbers. Allows specification of lower (*from*), upper bounds (*to*) as well as steps (*by*) and output length (*length.out*)	round(seq(0.00, 10.00, length.out = 13), 2) [1] 0.00 0.83 1.67 2.50 3.33 4.17 5.00 5.83 6.67 7.50 8.33 9.17 10.00	
paste()	Concatenates provided strings	paste('hallo', 'welt') # 'hallo welt' # note that paste adds a space between strings per default	
rnorm(n, mean = 1, sd = 1)	Standard distribution function with specified mean and standard deviation	rnorm(n=10, mean=1:10)	
plot(x, y = NULL, type = 'p', ...)	Standard scatter (2D) scatter plot function. Allows usage of **common graphical parameters** such as: col (color) bg (background color) pch (plotting character) cex (character scaling factor) lty (line type) lwd (vector of line width)	plot(x=1:10, y = 101:110, main='first graph', type='b')	
hist()	Standard histogram plot	hist(x = swiss$Fertility)	
boxplot()	Standard boxplot	boxplot(Sepal.Length ~ Species, data = iris)	
gsub()	Replaces all occurences of a pattern/substring in a string.	gsub(',', '', iris$petallength) # would replace all occurences of string ',' with an empty string	
substring()	Extracts or replaces a substring in a character vector	substring(text, from, to)	
str()	stands for 'structure' - gives a string representation of the selected structure		
ggplot()	base high level function for ggplots		https://ggplot2-book.org/index.html
[ggplot] aes()	creates the plot space	mapping = aes (y = iris$Species, x = iris$SepalLength)	
[ggplot] geom_smooth()	adds smoother (lowe's)	arguments: method - specifies the smoothing method see docs	
[ggplot] geom_point()	adds data points		
[ggplot] geom_hline()	adds a horizontal line at the specified *yintercept. Accepts graphical formatting parameters (linetype, colors etc.)*		
[ggplot] geom_line()	adds a plot line (potentially connecting available point)		
[ggplot] theme_xxx()	theme options for ggplot2. See also 'ggthemes' package		
[ggplot] scale_color_brewer(type='qual', palette ='Dark2')	enables colour scale for points		
[ggplot] facet_wrap(~factor)	facet_wrap() wraps a 1d sequence of panels into 2d. This is generally a better use of screen space than facet_grid() because most displays are roughly rectangular.		
read.table()	Read table from a structured source (e.g. web sources)		
abline()	Plots a line specified by intercept and slope		
pairs()	visualise data when relatively few predictors are there. Gives a graphical overview of all variable pairs.	pairs(y~x, data=xx) pairs(y~x, data=x, upper.panel = panel.smooth) # adds smoothers to visuals	

par()	sets parameters for plot output. Common parameters include: mfrow=c(x,y) number of pannels vertically and horizontally mar=c(bottom,left,top,right) sets the margins in the plot output mai=c(b,l,t,r) same as *mar* but sets margins in *inches* *col.main = 'magenta' sets color of the main plot titles to 'magenta'*		https://bookdown.org/ndphillips/YaRrr/plot-margins.html
jpeg() / png()	Save a plot as jpeg or png image	jpeg(filename = "Rplot%03d.jpeg", width = 480, height = 480, pointsize = 12, quality = 75, bg = "white", res = NA, ...)	
xyplot()	alternative to native *plot()* function. part of library(lattice)	xyplot(Sepal.Length + Sepal.Width ~ Petal.Length + Petal.Width | Species, data = iris, scales = "free", layout = c(2, 2), auto.key = list(x = .6, y = .7, corner = c(0, 0)))	
here('subfolder','folder')	from the 'here' package. Can be used to dynamically set the working directory based on one or more listed folders. The packages uses a heuristic to set the WD (looking for Rproject files or .git repos etc.)	setwd(here('Group_Work'))	
sessionInfo()	Prints current session information such as OS, R version and attached packages		
find()	Returns the packages to which a method / object belongs		
apropos()	Finds help pages where the search term is listed		
t.test()	perform statistical t-test (t-test for equal var / Welch Two Sample for unequal var). Compares mean of two groups	t.test(Sepal.Length ~ Species, data = iris) --> Welch Two Sample t-test	
%in%	check for element in vector/list	old syntax: iris[iris$Species %in% c('versicolor', 'virginica') dyplr variante: iris %>% filter(Species == c('versicolor','virginica'))	
all.equal()	check for element -wise equality of two vectors. Gives you the difference if not equal. Related: *identical()*		
methods()	displays all methods for a given object	methods(class = 'matrix'), methods(object)	
con <- dbConnect(RSQLite::SQLite(), [dbfile], url='.....', username, password) *tables <- dbListTables(con)* *customers <- dbGetQuery(conn=con, statement="SELECT * FROM customers;")*	connect to an SQL db using *library(RSQLite), library(DBI)* , get available tables and query one of the tables		
resid()	Gets the residuals of a model		
coef()	Gets the coefficients of a model		
fitted()	Gets the fitted values of a model		
update()	To change an existing model		
anova()	To compare two models. Anova() in R is not limited to pure ANOVA tests. The function can perform different tasks such as model comparison	anova(m.iris2, m.iris1)	
is.na()	Finds missing (NA) values in dataset and returns boolean vector	#get all rows with missing values df.test[is.na(df.test),]	
mice::ampute()	Imputation function to generate missing values in a complete data frame		
apply()	Applies a function to a vector or matrix	apply(X, MARGIN, FUN)	
[dplyr] %>%	Pipe symbol in dplyr which transports output from one command to the input of another	see examples below	
[dplyr] %$%	Alternative pipe for when working with individual vectors rather than data frames. This "explodes" the values so they can be addressed individually --> takes individual element of the left side rather than the whole data frame		https://r4ds.had.co.nz/pipes.html
[dplyr] select()	Selects certain columns from a dataframe	airquality %>% select(Ozone, contains('Temp')) # selects only columns 'Ozone' and all columns containing 'Temp' in their name # alternatives to *contains()* : *starts_with()* , *ends_with()*	https://rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf
[dplyr] filter()	Filters certain observations from a dataframe	airquality %>% filter(Ozone >= 80) # filters for observations where value of column 'Ozone' is equal or greater than 80	
[dplyr] arrange()	Sorts by specified column	airquality %>% arrange(desc(Ozone), Temp) # sorts dataframe by Ozone in descending order and within Ozone by 'Temp'	
[dplyr] top_n()	Selects to n observations	iris %>% top_n(n=5, wt = Sepal.Width)	
[dplyr] group_by()	Group by specified column analog to SQL	iris %>% group_by(Species)	
[dplyr] print()	Specifies the output form of the tibble	iris %>% print(width = Inf) #all rows // print(height = Inf) #all columns	
[dplyr] summarise()	Summarises results into one nice dataframe	iris %>% group_by(Species) %>% summarise(mean(Sepal.Length)) #puts results into one dataframe, here grouped by 'Species'	
[dplyr] mutate()	Adds new columns to existing data frame	iris_2 <- iris %>% mutate(Sepal.Length.cm = cm(Sepal.Length))	
library(shiny)	package for interactive web dashboards using R		https://shiny.rstudio.com/

[plotly] plot_ly()	Creates an interactive plotly plot. See example for a interactive 3D plot and interactive animated plot	library(plotly)
		fig2 <- plot_ly(data=data,
		x=~Temperature,
		y=~Hour,
		z=~RentCount,
		type="scatter3d", mode="markers", color=~Season #here, we added the color argument and set it to the 'Season' factor
		)
		
		#This creates an animated plot, given data for each frame (frame = ~frame)
		plot_ly(
		x = ~Hour,
		y = ~SumRentCount,
		frame = ~frame,
		#color = ~Season,
		split = ~Season,
		type = 'scatter',
		#mode = 'lines',
		line = list(simplyfy = FALSE)
		)
		# NOTE: the dataframe needs to contain cumulative data for each frame in order to visualise a developing line
			https://plotly.com/r/getting-started/

[plotly] %>% layout(autosize = FALSE, width = pixels, height = pixels, margin = list(l=,r=,b=,t=,pad=))	Set plot space and margins with dplyr syntax	fig2 %>% layout(autosize = FALSE, width = 1000, height = 600, margin = list(l=50, r=10, b=10, t=10, pad=1))	