Getting started, getting the data in

Function	Source Package	Description	Example
Getting activated			
`install.packages()`	Base	Installs packages from CRAN	install.packages("ggplot2")
`library()`	Base	Packages must be loaded at the beginning of each R session.	library(ggplot2)
`file.path()`	Base	Codes a file path that can be called in subsequent functions.	src <- file.path("/users/filepath")
Generating R objects			
`c()`	Base	Combines elements into a vector	x <- c(1,2,3) y <- c("A","B","C")
`factor()`	Base	Generates a categorical factor variables	x <- factor(x=vector, levels=c(1,2,3), labels=c("A","B","C"))
`quantcut()`	gtools	Categorizes a continuous variable quantiles	x <- quantcut(x=vector, q=4)
`sample()`	Base	Randomly samples a set of numbers	x <- sample(x=vector, size=n, replace=TRUE)
`rnorm()`	Base	Generates a random normal distribution	x <- rnorm(n=n, mean=10, sd=5)
`list()`	Base	Generates a list	mylist <- list(a, b, c)
`data.frame()`	Base	Generates a data frame	df <- data.frame(a,b,c, stringsAsFactors=F)
Explore the data			
`summary()`	Base	Summarizes a continuous variable	summary(example_object)
`table()`	Base	Produces frequencies for a variable	table(example_object, useNA="always")
`prop.table()`	Base	Produces proportions for a table() object	prop.table(table(example_object, useNA="always"))
`mean() / median()`	Base	Calculates the mean / median of a continuous variable	mean(x, na.rm=TRUE) median(x, na.rm=TRUE)
`min() / max()`	Base	Calculates the minimum / maximum of a continuous variable	min(x, na.rm=TRUE) max(x, na.rm=TRUE)
`is.na()`	Base	TRUE/FALSE - is a value NA	is.na(x)
`str()`	Base	Returns the structure of an object	str(data.frame.example_object)
`nrow() / ncol()`	Base	Returns the number of rows/cols in a data frame	nrow(data.frame.example_object)
`head() / tail()`	Base	Prints the first/last 6 observations of a vector or data frame	head(data.frame.example_object) tail(data.frame.example_object)
`identical()`	Base	TRUE/FALSE - are two objects exactly identical	identical(example_object1, example_object2)
Working with external data			
`read.table()`	Base	Read a flat delimited file into R	df <- read.table(file.path, stringsAsFactors=F)
`read_sas()`	Haven	Read a .sas7bdat file into R	df <- read_sas(file.path)
`read.xlsx()`	openxls	Read an excel spreadsheet into R	df <- read.xlsx(spreadsheet.path,

	x		sheet=1)
save()	Base	Saves objects as an RDATA file	save(object, file="Object name.Rdata")
saveRDS()	Base	Save an object as an RDS file	saveRDS(data.frame, file="Object name.RDS")
load()	Base	Load's an RDATA file into R	load(file="Object name.Rdata")
readRDS()	Base	Reads an RDS file into R	df <- readRDS("Object name.RDS")

Subsetting and filtering functions

Function	Source Package	Description	Example
Variable selection			
x[*i*]	Base	Select the *i*-value of x	x[5]
df[*row_i*,col_i]	Base	select the i-row and i-column of df-data frame	df[5, 10]
df[,"var_name"]	Base	select the "var_name" variable in a data frame	summary(df[,"BMI"])
df$variable	Base	select the "var_name" variable in a data frame	summary(df$BMI)
names()	Base	Print the variable names in a data frame	names(df)
names() <- c()	Base	Rename the variable names in a data frame	names(df) <- c("SUBJID","DOB","BMI","AGE")
Operators			
<, <=	Base	Less than / less than or equal to	var1 < var2 var1 <= var2
>, >=	Base	Greater than/ greater than or equal to	var1 > var2 var1 >= var2
%in%	Base	Includes	var1 %in% var2
==	Base	Exactly equal to	var1 == var2
&	Base	And	df[df$x==1 & df$y==2,]
!	Base	Not	df[df$x != 1,]
|	Base	Or	df[df$x ==1 \| df$x==2,]
Manipulating datasets			
rbind()	Base	Concatenates rows of 2 data frames	df3 <- rbind(df1, df2)
cbind()	Base	Combines columns of 2 data frames	df3 <- cbind(df1, df2)
merge()	Base	Merges to data frames based on a merging variable	df3 <- merge(x=df1, y=df2, by= "mergevar", all.x=T, all.y=T)
order()	Base	Sorts / orders an object	df3 <- df4[order(df4$mergevar),]

Helpful cleaning functions

Function	Description	Example
Variable transformations		
ifelse(x, yes, no)	Basic if-else-then conditional function for binary logic	ifelse(x=="A", yes=1, no=0)

ifelse(x, yes, ifelse(x, yes, no))	Nested if-else-then function for multiple conditional logic expressions	ifelse(x=="A", 1, ifelse(x=="B", 2, 3))
toupper() / tolower()	Converts a character string to all upper- or lower-case	toupper("my name is dave) [1] "MY NAME IS DAVE"
quantile()	Generates quantile cutpoints	quantile(x, c(0.25, 0.5, 0.75), na.rm=TRUE)
gtools::quantcut()	Categorizes a continuous variable into quantiles	x <- quantcut(x=vector, q=4)
Arithmetic operations		
+	Addition	x + y
-	Subtraction	x - y
/	Division	x / y
*	Multiplication	x * y
** or ^	Exponentiation	x**2 or x^2
log() / exp()	Natural log and exponentiation	log(x); exp(x)
sqrt() / nthroot	Square-root or Nth-root	sqrt(100); nthroot(100, 3)
floor() / ceiling()	Round down / up to the nearest integer	floor(10.5)
round(x, digits=0)	Round to some decimal place	round(pi) [1] 3.141593
Dates		
as.Date()	Creates a date object	as.Date("2019-06-25") as.Date(x, origin="1970-01-01") R.date <- as.Date(SAS.date, origin="1960-01-01")
Sys.Date()	Today's date	Sys.Date()
Sys.time()	System date and time	Sys.time()

Renaming all types of things

Function	Description	Example
Renaming variables		
names()	Rename all the variables in a data frame	names(df) <- c("Var1", "Var2", "Var3")
dplyr::Rename(dat, newname=old)	Renames individual variables and dropping the old ones	Rename(df, Newvar1 = OldVar1, Newvar2 = Oldvar2)
Factor variables		
factor(x, levels, labels)	Create a factor variable. Order of the variable is set by **levels**, formats are set with **labels**	factor(x, levels=c(1,2,3), labels="Never", "Current", "Former")
levels()	Renames the formats in a factor variable	levels(x, c("Never smoker", "Current smoker", "Former smoker")
relevel(x, ref=)	Changes the reference group of a factor variable	relevel(x, "Current smoker")
Apply() functions		

`apply(x, margin, fun)`	Apply a function (fun) across variable X.
	Margin=1 applies the function across each observation
	Margin=2 applies the function for each column
		`# Sums 3 variables for each observation`
		`apply(df[,c("var1", "var2", "var3")], 1, sum)`
		
		`# Sums each of the three variables`
		`apply(df[,c("var1", "var2", "var3")], 2, sum)`
		
		`# Creates a frequency table for three variables`
		`lapply(df[,c("var1", "var2", "var3")], table)`

`lapply(x, fun)`	List-apply. Applies a function across heterogenous data	`# Converts all variables to numeric` `lapply(df[,names(df)], as.numeric)`
`tapply(continuous, categorical, function)`	Useful for getting summary statistics for each level of a categorical variable.	`# Get range of BMI for each level of category BMI` `tapply(df$BMI, df$BMICAT, summary)`

Handling duplicates

Function	Description	Example
`duplicated()`	TRUE/FALSE - is a value a duplicate of another	`duplicated(x)`
`!duplicated()`	TRUE/FALSE - is a value NOT duplicated of another	`!duplicated(x)`
`table(duplicated(x))`	Frequency table - how many duplicates?	`table(duplicated(x)`
`df[!duplicated(df$ID,]`	Remove the duplicate ID observations from data frame "df"	`nodup <- df[!duplicated(df$ID,]`

		`wide <- reshape(data = long,`
		`varying=c("var1", "var2"),`
		`vnames = "newvar",`
		`idvar = "SubjectID",`
		`direction = "wide")`

`reshape(data, varying, v.names, idvar, direction)`	Reshape a dataset from long-form to wide form, or vice versa	`long <- reshape(data = long,`
		`varying=c("var1", "var2"),`
		`vnames = "newvar",`
		`idvar = "SubjectID",`
		`direction = "long")`

Stay functional!