R BASICS

R	Description	Example	Example Explanation
setwd()	Function that tells R where to look for the files you are loading and/or saving	setwd("/Users/jpan/Downloads")	Tells R that any files I load or save will come from or go into my Downloads folder; function takes one argument, the path to the folder in quotes
getwd()	Function that tells you where R is looking for the files you are loading and/or saving	getwd()	Will return the current working directory; don't put anything into the function
c()	Function to create vectors	c("erica", "paulina", "lauren")	Creates a vector of 3 character strings

<-	Assignment operator used to save something as an object	backrow <- c("erica", "paulina", "lauren")
		mynumbers <- c(6,7,8,2)
			Save the vector of characters as an object called backrow
			Save vector of numbers as object called mynumbers

[]	Single square brackets allows you to select a subset of an object	backrow[1] backrow[c(1,2)]	Selects the first item in backrow, "erica" Select the 1st and 2nd items in backrow, "erica" "paulina")
==	Double equal sign to test to see what values meet a condition you set	backrow == "erica"	Looks at each element in object backrow, returns TRUE if the element equals "erica" return FALSE is not; return TRUE FALSE FALSE
read.csv()	Function that loads .csv files as dataframes, to then work with the dataframe, you must save the output of read.csv() as an object	shave <- read.csv("shave.csv")	Loads the numbers from shave.csv as a dataframe called shave
load()	Function that loads .RData objects, you do not need to save the output of load() as an object because whoever saved the object as a .Rdata file already named the object	load("UNpop.RData")	Loads the numbers from UNpop.Rdata
write.csv()	Function that saves objects as .csv files; you must provide 2 arguments: the name of the object, the name of the new file you are creating	write.csv(backrow, file="comm106students_backrow.csv")	Write the object backrow into the .csv file called "comm106students_backrow"
?function()	Way to get help on any function whose name you know	?setwd()	Opens a windows with more information about the setwd() function

ARITHMETIC AND STATISTICAL FUNCTIONS

R	Description	Example	Example Explanation
sum()	Function that adds elements together	sum(mynumbers)	Takes object mynumber, adds them (6+7+8+2), returns 23

min()	Function that finds the minimum value of an object; only needs 1 argument, unless your object has NAs, then needs an additional argument	min(mynumbers)
		min(mynumbers, na.rm=TRUE)
			Finds the smallest value in mynumbers, returns 2
			Finds smallest value in mynumbers, excludes NAs

max()	Function that finds the maximum value of an object; only needs 1 argument, unless your object has NAs, then needs an additional argument	max(mynumbers)
		max(mynumbers, na.rm=TRUE)
			Finds the largest value in mynumbers, returns 8
			Finds largest value in mynumbers, excludes NAs

mean()	Function that finds the mean of a vector	mean(UNpop$world.pop)	Mean of the world.pop vector in the dataframe UNpop, returns 4579529
median()	Function that finds the median of a vector	median(UNpop$world.pop)	Median of the world.pop vector in the dataframe UNpop, returns 4449049
sd()	Function that finds the standard deviation of a vector	sd(UNpop$world.pop)	Standard deviation of world.pop vector in the dataframe UNpop, returns 1625004

cor()	Function that finds the correlation between two vectors, order does not matter	cor(UNpop$world.pop, UNpop$year)
		cor(UNpop$year, UNpop$world.pop)
			Correlation between UNpop$world.pop and UNpop$year; order does not matter, returns 0.9972364

lm()	Function that estimates the ordinary least square linear regression line	lm(UNpop$world.pop ~ UNpop$year)
		lm(world.pop ~ year, data = UNpop)
			Estimates the alpha hat and beta hat (the linear regression line) where UNpop$world.pop is the Y (dependent variable), and UNpop$year is the X (independent variable); both examples return the exact same thing; the (Intercept) is the constant or alpha hat, and the UNpop$year coefficient is the slope or beta hat - these two numbers are all that's needed to write the equation for a linear regression line

sqrt()	Function that takes the square root	sqrt(4)	Takes square root of 4, returns 2
log()	Function that takes the natural log of a number	log(10)	Takes natural log of 10, returns 2.302585

FUNCTIONS TO EXPLORE DATA

R	Description	Example	Example Explanation
names()	Function that provides the column names	names(UNpop)	Gives column names of UNpop dataframe, returns "year" "world.pop"
colnames()	Function that provides the column names	colnames(UNpop)	Gives column names of UNpop dataframe, returns "year" "world.pop"
rownames()	Function that provides the row names	rownames(UNpop)	Gives row names of UNpop dataframe, returns "1" "2" "3" "4" "5" "6" "7"

dim()	Function that provides the number of rows and number of columns of objects with rows and columns	dim(backrow)
		dim(UNpop)
			Backrow does not have columns and rows, so dim(backrow) returns NULL
			Unpop does have columns and rows, so dim(UNpop) returns 7 (rows) 2 (cols)

nrow()	Function that provides the number of rows of objects with rows and columsn	nrow(backrow)
		nrow(UNpop)
			Backrow does not have rows, so nrow(backrow) returns NULL
			UNpop does have rows, so nrows(UNpop) returns 7 (rows)

ncol()	Function that provides the number of columns of objects with columns and rows	ncol(backrow)
		ncol(UNpop)
			Backrow does not have columns, so ncol(backrow) returns NULL
			UNpop does have columns, so ncol(UNpop) returns 2 (columns)

length()	Function that shows you the length of any object, for objects with rows and columns, length() returns the number of columns like ncol()	lengh(backrow)
		length(UNpop)
			backrow is an object with 3 elements, length(backrow) returns 3
			length(UNpop) returns 2 for 2 columns

FUNCTIONS TO EXPLORE DATA (continued)			
R	**Description**	**Example**	**Example Explanation**
View()	Function that opens a window where you can look at your data in a spreadsheet-style viewer	View(UNpop)	Opens a window where you can look at UNpop data

head()	Function that prints the first few rows of your data; by default shows 6 rows, can add argument to show more	head(UNpop)
		head(UNpop, 2)
			Looks at first 6 rows of data in UNpop (default)
			Looks at the first 2 rows of data in UNpop

summary()	Function that calculates the min, first quartile, median, mean, third quartile, and max values of an object	summary(mynumbers)
		summary(UNpop)
			Generates summary statistics for the vector mynumbers
			Generate summary statistics for each column of Unpop

class()	Function that shows you the class of the object, you put the name of the object inside the parentheses	class(backrow)	Tells you the class of the backrow object is "character"
subset()	Function to subset dataframe; 2 arguments: subet(yourdata, someconditionyourdatameets)	subset(UNpop, UNpop$year == 2010)	Subset the dataframe UNpop to only the row(s) where UNpop$year takes on the value 2010
ifelse()	Function to select (and do whatever you want with) variables that meet certain conditions; 3 arguments: ifelse(truefalsestatement, valueifstatementtrue, valueifstatefalse)	ifelse(shave$gender == "female", 1, 0)	If values in shave$gender is "female" then returns a 1, returns a 0 if shave$gender does not equal to "female" (this basically recodes shave$gender in 1s and 0s)

table()	Function that tabulates counts of unique values in your variables	table(shave$gender)
		table(shave$gender, shave$shaveface)
			shows that there are 29 female and 26 male in vector shave$gender
			shows the number of females and males by frequency of shaving; the vector you put first will be the columns, and the vector you put in second will be the rows

prop.table()	Function that calculates proportions of your table() output by row or column	mytable <- table(shave$shaveface, shave$gender)
		prop.table(mytable, 2)
		prop.table(mytable, 1)
			Save cross tabulation as object mytable
			Calculate proportion where each column sums to 1
			Calculate proportion where each row sums to 1

FUNCTIONS TO MAKE PLOTS			
R	**Description**	**Example**	**Example Explanation**

boxplot()	Function that creates plots where median, 1st and 3rd Qs are displayed as central box, and whiskers are quartile plus or minus 1.5 times IQR; take at minimum 2 arguments: boxplot(variableonyxaxis ~ variableonxaxis, data), but you can add more plot arguments like labels for axis	boxplot(altitude ~ village.surveyed,
		data = afghan,
		ylab = "Altitude (meters)",
		names = c("non-sampled", "sampled"))
			Plots altitude for the two values found in village.surveyed
			using the dataframe afghan
			labeling the y axis "Altitude (meters)"
			labeling each of the two values in village.surveyed

hist()	Function that plots the frequency of occurences of different values by bins	hist(afghan$population)	Frequency of population by bins

plot()	Function that plots X and Y values; take a minimum 2 arguments: plot(xvalues, yvalues) but you can add more plot arguments like lables for axis, title, color of points, type of points	plot(x = gini$year, y = gini$gini,
		pch=19,
		xlab="Year",
		cex.axis=2,
		cex.lab=2,
		cex=2,
		cex.main = 2,
		col = "hotpink2",
		ylab="Gini",
		main = "Income Inequality")
			Plots as x values the year, as y values the US gini coefficient
			make the points of pch = 19 (round solid dots)
			labels the x axis "Year"
			makes the font of the axis numbers bigger
			makes the font of the axis labels bigger
			makes the size of the dots bigger
			makes the size of the title bigger
			makes the color of the dots "hotpink2"
			labels the y axis "Gini"
			adds a title to the plot "Income Inequality"

par(mar =)	Function that goes before plot() and by default equals par(mar = c(5.1, 4.1, 4.1, 2.1)), change the values to change the margin on your plot, the four values refer to c(bottom, left, top, right) in lines	par(mar = c(5.1, 4.1, 4.1, 2.1))
		plot(y = gini$gini,x = gini$year, pch=19,
		xlab="Year",
		cex.axis=2, cex.lab=2, cex=2, cex.main = 2,
		col = "hotpink2", ylab="Gini",
		main = "Income Inequality")
			Makes the left margin of the plot described above one line wider (so the y axis label is not cut off)

abline()	Function that goes after plot() to add lines	plot(y = gini$gini,x = gini$year)
		abline(v=1980, lty="dashed")
		abline(h=.40, lty="dotted")
		abline(lm(gini$gini ~ gini$year), col="red")
			Plots as x values the year, as y values the US gini coefficient
			Add vertical line at y value 1980, line is dashed
			Add horizontal line at x value 0.40, lined is dotted
			Add OLS linear regression line of Y = gini$gini, X = gini$year, line is red colored

pdf() and dev.off()	Function that go before and after plot() to save everything in between as a pdf; for pdf() you must put the name of the pdf file in "", do not put anything in dev.off()	pdf("myginiplot.pdf")
		plot(y = gini$gini,x = gini$year)
		abline(v=1980, lty="dashed")
		abline(h=.40, lty="dotted")
		abline(lm(gini$gini ~ gini$year), col="red")
		dev.off()
			Save the plot described above as a PDF file called "myginiplot.pdf"