# # # # # # # # # # # # #

# TITLE: 2.2 Bivariate Statistics in R

# SOCI832: Advanced Research Methods

# Week 2: Linear Regression (2 of 3 script files)

# Author: Nicholas Harrigan

# Last updated: 5 Aug 2018

# # # # # # # # # # # # #

#

# WHAT DOES THIS SCRIPT DO?

# \* Introduces methods for calculating

# and presenting bivariate statistics in R

#

# WHAT DOES THIS TEACH ME ABOUT R?

# (1) Basic commands for bivariate statistics

# (2) Handy packages and tricks for getting your data in

# publishable formats (e.g. table out; summary stats; etc.)

#

# WHAT DOES THIS TEACH ME ABOUT STATISTICS?

# (1) Comparision of means

# (2) Correlation coefficients

# STUFF YOU NEED TO DO BEFORE STARTING

# 1. Change this to your working directory

setwd("C:/G/2018, SOCI832/Datasets/AES 2013/")

# 2. Put the file "elect\_2013.csv" into that folder

# This file can be found here:

# https://mqsociology.github.io/learn-r/soci832/elect\_2013.csv

# 3. Keep the codebook openned in a browser so you

# can refer to it when you need it. The codebook is here:

# https://mqsociology.github.io/learn-r/soci832/codebook%20aes%202013.html

# 4. Install summarytools package if you havne't already

install.packages("summarytools") # install the package

# (once, and then # it out)

library(summarytools) # load the library

# START HERE

# Import the data

elect\_2013 <- read.csv("elect\_2013.csv") # loads dataset

# The next command (below) gets rid of the first column

# which is not needed.

# FYI the command works by saying

# "copy all columns except the first".

# NOTE: Only run this command once after you run the

# 'read.csv' command. Each time you run it, it deletes

# the first variable.

elect\_2013 <- elect\_2013[,2:ncol(elect\_2013)]

######################################

# LESSON 1: CROSSTABS

######################################

# This section shows how to cross-

# tabulate the levels of two variables.

# This is often one of the simplest

# ways to get a quick look at the

# relationship between two variables

######################################

# - - - - - - - - - - - - - - - - - - - - - - - - - - - -

# Crosstabs Example 1: Pol Know vs Likelihood voting

# - - - - - - - - - - - - - - - - - - - - - - - - - - - -

# This is how the command 'ctable()' presents a cross

# tabulation with default settings:

print(ctable(elect\_2013$pol\_knowledge,

elect\_2013$likelihood\_vote),

method = "browser")

# Notice that each cell in the table has two numbers

# the first number is the number of survey respondents

# who had those two values of the two variables.

# For exampke, there are 51 people who got zero on the

# political knowledge quiz, and who also said their likelihood

# of voting if voting was not compulsory was 'definitely not'

# (i.e. a 1 on a 5 point scale).

#

# The second number in each cell (contained in brakets)

# is a percentage. In this default setting, this is the

# row percentage - the percentage of people with a political

# knowledge score of zero, who have a '1' on the

# 'likelihood\_vote' variable. You can see that this

# is 12.44%.

#

# Often when reading cross tabulations it can be difficult

# to work out whether the percentages are row or column

# percentages. One of the easiest ways to work this out

# is to simply look at the Total column, and Total row.

# Almost always you will find that one of the two Totals is

# equalt to 100% for all values of a variable. The total

# with the 100% is the denominator for all cells in it's

# row or column.

#

# So in this case, we can see that the Total column has

# 100% for all variables, telling you that the percentages

# are "percent of people with political knowledge of X, who

# had a likelihood of voting of Y."

#

# 'prop' ARGUMENT

# We can control the displaying of row and column percentages

# with the argument 'prop'. prop takes four different settings

# "r" for 'row percentages is the default (so you don't need

# to put this setting in).

# The other options are:

# "c" for column totals (cell count/column count)

# "t" for displaying percent of all cases (cell count/total count)

# "n" for none - no percentages

#

# So this command displays column totals

print(ctable(elect\_2013$pol\_knowledge,

elect\_2013$likelihood\_vote,

prop = "c"),

method = "browser", footnote = NA)

# Percentages of the total number of participants/cases

print(ctable(elect\_2013$pol\_knowledge,

elect\_2013$likelihood\_vote,

prop = "t"),

method = "browser", footnote = NA)

# No percentages

print(ctable(elect\_2013$pol\_knowledge,

elect\_2013$likelihood\_vote,

prop = "n"),

method = "browser", footnote = NA)

# As with the other summarytools functions, you can

# use various arguments to clean up the display of

# your table.

#

# Note that for some reason we use the argument

# 'useNA' rather than 'report.nas' to remove

# NAs from our table.

print(ctable(elect\_2013$pol\_knowledge,

elect\_2013$likelihood\_vote,

omit.headings = TRUE,

totals = FALSE,

useNA = "no",

prop = "n"),

method = "browser", footnote = NA)

######################################

# LESSON 2: COMPARISION OF MEANS

######################################

# We will learn about the basic ways

# to compare the difference in means

# of two groups - one where the groups

# are independent (e.g. height of men

# and women), and one where they are

# paired (e.g. height of the same people

# at age 15 and age 15.5 years).

######################################

# Let's say we want to compare the political knowledge

# of men and women in our dataset. We want to ask if

# the mean for men, and the mean for women is different

# the command to test this is 't.test'.

t.test(elect\_2013$pol\_knowledge ~ elect\_2013$female)

# In the console window you should see:

#

# Welch Two Sample t-test

#

# data: elect\_2013$pol\_knowledge by elect\_2013$female

# t = 11.64, df = 3838.6, p-value < 2.2e-16

# alternative hypothesis: true difference in means is not equal to 0

# 95 percent confidence interval:

# 0.8891745 1.2493867

# sample estimates:

# mean in group 0 mean in group 1

# 5.328302 4.259021

#

# >

#

# Now I know this looks like a mess, but it is actually not too

# difficult to understand.

# The fundamental rule I teach is that the first thing you

# do when you run a test is look at the p-value, and see

# if it is below 0.05. If it is not below 0.05, then

# there is generally not a need to interpret anything else

# because the test is not significant.

# [There is a complication to this instruction, because

# some statistical commands give multiple p-values - one

# for each variable, and one for the model overall - but we

# will deal with that later]

#

# So if we follow this rule of looking at the p-value first,

# What does it say?

#

# The p-value in this case is "p-value < 2.2e-16". What does

# that mean? it means the number 2.2 with 16 zeros in FRONT of

# it. i.e. p = 0.000000000000000022.

# Is that less than 0.05? Yes!

#

# So what is the next step for intepreting this output?

#

# Let's look at the last three line. They say:

# sample estimates:

# mean in group 0 mean in group 1

# 5.328302 4.259021

# this is telling us that the mean of the group with value "0"

# is 5.33, and the mean for the group with value "1" is 4.26.

# But what is group 0 and 1? Well we need to look at our data.

# The means are measure in "pol\_knowledge" units, and the variable

# for gender is 1 = female, and 0 = male.

# so this tells us that the mean political knowledge for men

# in our sample is 5.3, and for women is 4.3.

#

# We could stop interpreting our data here, but there is another

# useful part of the output to interpret. Look at these two lines:

# 95 percent confidence interval:

# 0.8891745 1.2493867

# This tells us that the 'difference of means' between men and

# women has a 95% confidence interval of 0.89 to 1.25. This says

# that the TRUE difference between men and women - the

# population parameter - is with 95% certainty between

# 0.89 and 1.25.

# the second type of comparison of means we are going to run

# is the paired test. In a paired test the two variables

# to be measured are measured on the same units of analysis

#

# The reason we need a different test for this is because

# when the same unit of analysis is used for the two variables

# the two variables are dependent on each other - they

# are not independent samples - as so the statistical test

# changes to account for this.

#

# In the next example, we are going to compare participants

# average score for 'following the election on TV' vs 'following

# the election in the newspaper'.

t.test(elect\_2013$election\_tv,

elect\_2013$election\_newspaper,

paired = TRUE)

# This can be read in the same way as the previous t-test

# except that in this case the last line reports the difference

# in means, not the two means.

#

# Intuitively we know that this means people followed the

# election more in the newspaper than on TV, but we can

# check this, by just running two means() to double

# check we are right:

mean(elect\_2013$election\_tv, na.rm = TRUE)

mean(elect\_2013$election\_newspaper, na.rm = TRUE)

# And you can see that what we thought was true is, with

# people having an average score of 2.01 for election\_tv, and

# 2.43 for election\_newspaper.

######################################

# LESSON 3: CORRELATION

######################################

# Coorelation coefficents are some

# of the most widely used and reported

# statistics in academic papers.

# In this section we learn about how

# to obtain the most common types of

# correlation coefficients.

# We also create our own function for

# making very attractive correlation

# matricies.

######################################

# We will be needing these two packages later,

# so make sure you have installed them, and also

# loaded them into this session of by calling them

# using the 'library' command.

install.packages("Hmisc") # hash this out after installing

# on your computer

install.packages("xtable") # and this.

library(xtable)

library(Hmisc)

# The third form of bivariate statistics we are learning

# about today is correlation, and in particular

# correlation coefficients.

#

# Correlation coefficients can roughly be defined as:

# \* a measure of the relationship between two variables

# \* that is standardised so that

# \* a perfect positive correlation is 1

# \* a perfect negative correlation is -1

# \* and no relationship is 0

#

# It turns out that there are lots of different

# correlation coefficients, and statisticians are

# probably still inventing new correlation coefficients

# today.

#

# When most people talk about correlation coefficients,

# or just correlation, they are talking about the

# most famous and widely used correlation coefficient

# called "Pearson correlation coefficient". This was

# invented by the famous statistician Karl Pearson.

# It is also called 'Pearson's R' and sometimes just 'r'.

# And that has no relationships with the statistical

# package 'R'!

#

# Each correlation coefficient is measured in a different

# way, and this is so that it takes account of the different

# types of data that the coefficient is working with.

#

# When we have approximately continuous data, Pearson's

# correlation coefficient is the best measure, and by

# learning about this measure, you will get a sense of

# what all the different correlation coefficients are

# trying to do.

#

# There is an excellent four page explaination of Pearson's

# r in Field et al. 2012 pages 206-209 (6.3.1- 6.3.2) which

# recommend everyone read. I will try to summarise it

# here, but my explanation will be overly simple.

#

# A correlation coefficient brings together two main

# ideas: (1) covariance; and (2) standardisation.

#

# Covariance is the idea that for two variables, when

# one is high on an individual, the other variable will

# also be high, and vice versa - when one is low,

# the other variable will be low.

#

# We could think of a very simple example of measuring

# height and weight on everyone in a school. Now we

# know that height and weight are not perfectly correlated

# because there are lots of other factors. But we also

# know that they tend to move together. This "moving

# together" is covariance.

#

# The way statisticians measure covariance is by looking

# at how much two measures move away from the mean of

# the whole sample. So if we have a school where the

# average height is 130cm and the average weight is 50kg,

# statisticians would measure the covariance by asking

# "When someone differs from the mean height, do they

# also differ by a similar amount in the same direction

# from the mean weight?"

#

# Mathematically the covariances of a person called Anne is:

# (Anne's weight - average school weight)\*

# (Anne's height - average school height)

#

# For the whole school, we can calcuate the average covariance

# which is called covariance by taking the average

# (actually n-1 is the denominator - which I won't explain)

# And this will give us a number that is positive

# if height and weight move together, and negative if

# they don't, and approximately zero if there is no relationship.

#

# However, we have a problem, which is that the units

# of covariance dependent on what we are measuring

# and the units we are measuring it.

#

# Statisticians solve this by using 'standardisation',

# which is actually used in many different settings.

#

# The formula for standardisation (in almost all settings)

# is "subtract the mean from each case, and then divide

# by the standard deviation". The reason you subtract

# the mean from each case, is that it causes the new

# mean of all cases to be zero.

#

# Now it turns out that for covariance, we have already

# subtracted the mean as part of the formula, so we

# don't need to worry about this.

#

# The second rule is to divide by the standard deviation

# of a variable (or in this case, the product of the

# standard deviation of each variable). Why? Because

# It transforms ALL our units of analysis - that is EVERY

# variable we divide by it's own standard deviation -

# so that the numbers "1" and "-1" mean exactly the same

# thing: they mean 1 standard deviation from the mean

# of the variable in the sample.

#

# This can be seen in Field et al. 2012 in equation

# 6.3.

#

# And the result is a magical number called Pearon's r.

#

# When a statistical packages such as R calculates

# a Pearons's r for you, there two questions you should

# be asking about the number:

# \* Is it 'significant'? - which we measure with p-value

# and/or confidence intervals.

# \* Is it 'important'? - which we call the effect size.

#

# Reading significance with p-values is pretty straight

# forward. Remember that we ask is the p-value < 0.05

#

# Reading confidence intervals is similarly easy. Generally

# we say that a Pearson's r is significant if the 95%

# confidence interval does not include zero (because

# if it did include zero, then there is a greater

# than 5% chance that the true parameter is zero).

#

# But how do we know 'importance'?

#

# There is a concept in statistics which has grown in

# importance over the last 30 years, and this is known

# as 'effect size'. The argument goes something like this:

# If you have a large enough sample size, then you can

# get a p-value less than 0.05 for something that is

# actually of only trivial importance. And for this

# reason, we can't just use p-values to tell us

# whether something really matters in the real world.

#

# As a result of this criticism, the notion of effect

# size's has developed.

#

# A variable has a large effect on another variable if,

# for example, a one standard deviation variation in

# one variable causes (or is correlated with) a one

# standard deviation variation in the second variable.

#

# We can see here that we are thinking about how variance,

# measured in standardised units, in one variable is

# related to variance in another.

#

# Anyway, we will be returning to effect size quite a

# few times in this course, for the moment, I just

# want you to know that in the case of Pearson's r,

# effect size is measured DIRECTLY in the coefficient.

#

# And the rule of thumb for interpreting effect sizes

# of a Pearson's r are:

# small effect size: +/- 0.1 - 0.2

# medium effect size: +/- 0.3 - 0.4

# large effect size: +/- 0.5+

#

# Now, while 'small', 'medium', and 'large' are quite

# vague terms, they are honestly used in much of statistics

#

# One way to get a grip on what these mean is to

# take the square of pearson's r, which is called

# 'R-square' or 'R-squared'.

#

# It turns out that R-squared can be read directly as

# the "Proportion of variation in one variable explained

# by the second and vice versa".

#

# In short, it can basically be read like a proportion

# (or if you like, percentage, just multiple by 100).

#

# So if we transform these Pearson's r into R-square,

# we can rewrite the rule of thumb for effect size

# measured in R-square as:

# small effect size: 0.01-0.04 (i.e 1%-4%)

# medium effect size: 0.09-0.16 (i.e.9%-16%)

# large effect size: 0.25+ (ie. 25%+)

#

# With these rules of thumb in hand, let's look

# at our data and analysis it with R.

#

# - - - - - - - - - - - - - - - - - - - - - - - - - - - -

# Correlation Method 1: cor.test()

# - - - - - - - - - - - - - - - - - - - - - - - - - - - -

# The two simplest commands to get a correlation

# coefficient are cor() and cor.test()

#

# cor() unfortuately doesn't give a lot of information

# so it's not that useful.

# cor.test() however is quite useful.

#

# Let's look at the correlation between five different

# variables and likelihood of voting.

#

# I'm going to move from those with little or no effect,

# through to those with a large effect size.

#

# This first correlation shows the relationship between

# urban/rural location and likelihood of voting

cor.test(elect\_2013$rural\_urban, elect\_2013$likelihood\_vote)

# You should see something like the following

# in your console window:

# Pearson's product-moment correlation

#

# data: elect\_2013$rural\_urban and elect\_2013$likelihood\_vote

# t = 1.3682, df = 3801, p-value = 0.1713

# alternative hypothesis: true correlation is not equal to 0

# 95 percent confidence interval:

# -0.009604256 0.053932694

# sample estimates:

# cor

# 0.02218662

# How do we read this? Remember the first thing to do is

# to look at the p-value. In this case it is 0.17, which

# is greater that 0.05, so we can safely say the

# correlation coefficient is not significantly different

# from zero.

#

# We can also see this when we look at the 95% confidence

# interval, which is from -0.0096 to 0.053. Notice that

# the 95% confidence interval includes zero, which

# means that we can't be confident the true parameter

# is not zero.

#

# We can also see that the correlation coefficient is

# 0.022 - which is tiny. Remember that a correlation of

# 0.1 is considered a small effect size, and this is

# one fifth of that.

#

# Let's next look at the effect of highest educational

# qualification:

cor.test(elect\_2013$highest\_qual, elect\_2013$likelihood\_vote)

# in this case, the p-value is below 0.05, so the correlation

# is significant, but when you look at the correlation

# coefficient - it is basically 0.04, which again is tiny.

# So this is a case of 'significant' but 'not very meaningful'.

#

# Next let's look at the impact of internet skills on

# the likelihood of voting:

cor.test(elect\_2013$internet\_skills, elect\_2013$likelihood\_vote)

# In this case, the correlation is highly significant

# and the effect size - basically 0.09 - is small but

# meaningful. So those people who have better internet

# skills tend to be more likely to vote.

#

# Next let's look at something with a much larger effect

# size: political knowledge.

cor.test(elect\_2013$pol\_knowledge, elect\_2013$likelihood\_vote)

# In this case the p-value is incredibly small, and the

# correlation coefficient is 0.37 - a medium sized

# effect.

#

# Last, let's look at the impact of interest in politics:

cor.test(elect\_2013$interest\_pol, elect\_2013$likelihood\_vote)

# In this case the relationship is highly significant

# and effect size is large, but the relationship is

# negative: -0.52.

#

# Does this mean that people who are interested in politics

# are less likely to vote? Are those interested in

# politics all anarchists?

#

# What do you think?

#

# When you get a result like this, you should check two

# things: your codebook, and your data-entry/data cleaning.

#

# If you look at the codebook, then you will find that

# "interest\_pol" is reverse coded. (1) = A good deal of

# interest in politics, while (4) = None (no interest in

# politics).

#

# Thus, the real meaning of this correlation is that

# those who are more interested in politics are much

# more likely to say they will vote.

#

# In fact, we can say that approximately 25% of the

# variance in likelihood of voting is explained by

# variance in interest in politics (by calculating

# the R-square).

#

# - - - - - - - - - - - - - - - - - - - - - - - - - - - -

# Correlation Method 2: corstar()

# - - - - - - - - - - - - - - - - - - - - - - - - - - - -

# AN EASY WAY TO MAKE A CORRELATION MATRIX

# - - - - - - - - - - - - - - - - - - - - - - - - - - - -

#

# The command I have taught you above - cor.test() -

# is fine for testing the relationship between

# just two variables. However, often you want to

# be able to look at the correlations between a

# large set of variables all the at the one time.

#

# Actually many journal articles expect you to

# produce a correlation matrix of all your variables

# as a prelude to then running a linear regression

# model.

#

# I have found a function on the internet which

# will allow you to easily make your own correlation

# matrix.

#

# The function is called 'corstar()'.

#

# NOTE: To use it, you first need to run the code below

# and create the function in the memory of your computer.

#

# You don't need to understand the code, but do

# feel free to take a look.

#

# When you are ready run the code from

# where I type #START HERE# to #END HERE#

#START HERE#

corstars <-function(x, method=c("pearson", "spearman"), removeTriangle=c("upper", "lower"),

result=c("none", "html", "latex")){

# SOURCES:

# \* http://www.sthda.com/english/wiki/elegant-correlation-table-using-xtable-r-package

# \* http://myowelt.blogspot.com/2008/04/beautiful-correlation-tables-in-r.html

# \* https://stat.ethz.ch/pipermail/r-help/2008-March/156583.html

#Compute correlation matrix

require(Hmisc)

x <- as.matrix(x)

correlation\_matrix<-rcorr(x, type=method[1])

R <- correlation\_matrix$r # Matrix of correlation coeficients

p <- correlation\_matrix$P # Matrix of p-value

## Define notions for significance levels; spacing is important.

mystars <- ifelse(p < .0001, "\*\*\*\*", ifelse(p < .001, "\*\*\* ", ifelse(p < .01, "\*\* ", ifelse(p < .05, "\* ", " "))))

## trunctuate the correlation matrix to two decimal

R <- format(round(cbind(rep(-1.11, ncol(x)), R), 2))[,-1]

## build a new matrix that includes the correlations with their apropriate stars

Rnew <- matrix(paste(R, mystars, sep=""), ncol=ncol(x))

diag(Rnew) <- paste(diag(R), " ", sep="")

rownames(Rnew) <- colnames(x)

colnames(Rnew) <- paste(colnames(x), "", sep="")

## remove upper triangle of correlation matrix

if(removeTriangle[1]=="upper"){

Rnew <- as.matrix(Rnew)

Rnew[upper.tri(Rnew, diag = TRUE)] <- ""

Rnew <- as.data.frame(Rnew)

}

## remove lower triangle of correlation matrix

else if(removeTriangle[1]=="lower"){

Rnew <- as.matrix(Rnew)

Rnew[lower.tri(Rnew, diag = TRUE)] <- ""

Rnew <- as.data.frame(Rnew)

}

## remove last column and return the correlation matrix

Rnew <- cbind(Rnew[1:length(Rnew)-1])

if (result[1]=="none") return(Rnew)

else{

if(result[1]=="html") print(xtable(Rnew), type="html")

else print(xtable(Rnew), type="latex")

}

}

#END HERE#

# if you have run that code, then the code should

# have appeared in your console window, but

# nothing else should have happened.

#

# You have loaded 'corstars()' into your computer

# you can now run the corstars command

# and generate a correlation matrix.

#

# I have created the code below to

# allow you to run, save, and view the

# output of corstars().

#

# The three lines of code below will:

# (1) create a correlation table and save

# it in variable x

# (2) save this to your computer as a file

# (3) open that file in a browser, so you can

# see the table.

#

# Feel free to run the following code:

x <- print(xtable(corstars(elect\_2013)), type="html")

dput(x, file = "output.html")

browseURL("output.html")

# The correlation matrix should open in your browser.

# Significance is indicated with stars:

# \*\*\*\* = p < .0001

# \*\*\* = p < .001

# \*\* = p < .01

# \* = p < .05

# This is the end of the Bivariate statistics

# R code/ R-script file. The class continues

# in the code "Week 2, Part 3, Linear Regression.R'