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

# ESM 206 LAB 1

# Allison Horst

# 9/27/17

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

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

# Step 1. Intro to RStudio Environment

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

### A - Open RStudio

### B - The environment (explain windows, scripts, files, environment, etc.)

### C - Working in the console (why it's quick and dirty, why it's a nightmare for research and collaboration)

### D - Why scripts are much better. Here, have them create a new script

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

# Part 2. Scripts - Good habits, organization and other things

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

### A - How scripts work (+ comments)

### B - How to run code in a script (+ shortcuts)

### C - They start creating their own script using great organization

### D - Have them create a couple of variables - note how they show up in the Environment window when stored (create so they can see how saving a script means they can quickly retrieve variables)

### E - Save the script they've started creating (make a point of emphasizing importance of organized folders starting NOW)

### \*\*\* Remember show them to wrap text: Tools > Global Options > Code > Soft Wrap R Source Files > Apply \*\*\*

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

# Part 3. Loading the package 'tidyverse'

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

### A - Explain briefly what packages are (and why they don't all load automatically)

### B - Show them how they would install packages if they weren't already

### C - Load package tidyverse

library(tidyverse)

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

# Part 4. Loading data from .csv files

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

### A - Describe how to load using buttons (Environment Tab > Import Dataset > From CSV > Browse > Import)

### B - Load dataset 'National Parks Visitation Data'

### C - Copy and paste the readr() code for automatically loading the dataset into RStudio, and explain why this can save time especially if working with multiple datasets (but requires working on the same computer)

National\_Parks\_Visitation\_Data <- read\_csv("~/Documents/Bren Courses/ESM 206/Fall 2017/Labs/Lab 1/National Parks Visitation Data.csv") # Make sure they get used to adding comments after each line of code!

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

# Part 5. Initial data exploration

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

NPDF <- National\_Parks\_Visitation\_Data # Make a copy that's called NPDF, which is easier to call as a variable

## ALWAYS. EXPLORE. YOUR. DATA. ##

names(NPDF) # Shows variable (column) names

dim(NPDF) # Dimensions of dataset

class(NPDF) # Data class

head(NPDF) # Shows first 6 lines of dataset

tail(NPDF) # Shows last 6 lines of dataset

# Want to know how a function works?

?names # Single question mark brings up R documentation for that function

??name # Will bring up every function that may contain this term

# Single columns can be referred to using a '$'

Reg <- NPDF$Region # A vector only containing information in column 'Region' from NPDF

MaxVis <- max(NPDF$Visitors) # Finds the maximum value from the 'Visitors' column in the NPDF df

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

# Part 6. Basic data wrangling (dplyr functions)

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

df1 <- select(NPDF, State:YearRaw) # Only select columns from State through YearRaw in NPDF data frame, store as a new data frame 'df1'

View(df1) # Remember to look at it!

df2 <- filter(df1, State == "CA" & Type == "National Park" & YearRaw >= 1950) # Filters so that only rows are retained if the State column is 'CA' and the Type column is 'National Park' AND only incudes if Year is 1950 or greater, stores as new subset data frame 'df2'

View(df2) # Always look!

df3 <- arrange(df2, Code, YearRaw) # Arranges data first by alphabetical order in 'Code' column, THEN by ascending value in YearRaw column

View(df3)

df4 <- mutate(df3, kVis = Visitors/1000) # Adds a NEW column called 'kVis' (for thousands of visitors) to 'df3' that is the Visitors column divided by 1000 (convert to units of 1000s of visitors)

View(df4)

df5 <- filter(df4, !YearRaw %in% "Total") # Remove any 'Total' values in the YearRaw column (this is a little advanced...tell them not to worry about it much). Basically %in% looks to match "Total" in YearRaw...putting the '!' before YearRaw makes it the opposite (look for anything NOT matching Total in the YearRaw column)

# Look at the class of each of the columns

summary(df5) # Uh oh. YearRaw is stored as a character (because it HAD those 'Total' characters in it). We want it to be stored as a number so that we can use it easily as our x-axis value.

df5$YearRaw <- as.numeric(df5$YearRaw) # Changes the YearRaw column of df5 to 'numeric' instead of 'character'

class(df5$YearRaw) # Check! Now it returns 'numeric.' Hooray!

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

# Part 7. Our first graph in ggplot

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

# We're going to make a graph of visitation to the parks (in thousands of visitors) over years, with the color dependent on what park we're showing.

# Three things you must tell R to make a graph in ggplot:

# (1) That you're using ggplot

# (2) What data you're using (including what should be x and what should be y)

# (3) What type of graph you want to create

# Everything after that is extra to make it beautiful

VisitorGraph <- ggplot(df5, aes(x = YearRaw, y = kVis)) +

geom\_point()

quartz() # will be windows() on a PC!

VisitorGraph

# So that's kind of cool for a first shot, but it is definitely not a finalized graph. Let's start adding levels, and changing the line type, so we can get a graph that's a little nicer.

VisitorGraph2 <- ggplot(df5, aes(x = YearRaw, y = kVis)) + # Tells R what data to use

geom\_point(aes(colour = Code)) + # Tells ggplot to make a scatterplot graph

geom\_line(aes(colour = Code)) + # ...then add to that a line graph

ggtitle("California National Parks Visitation\n1950 - 2016") + # Gives it a title

xlab("\nYear") + # Note that \n just adds a blank line before this label

ylab("Thousands of Visitors\n") + # Again, but after the label

theme(panel.grid.major = element\_blank(), panel.grid.minor = element\_blank(),

panel.background = element\_blank(), axis.line = element\_line(colour = "black")) + # These things all 'clean up' graphic area - remove gridlines, background, etc.

scale\_x\_continuous(breaks = c(1950,1960,1970,1980,1990,2000,2010,2020)) # Specify where tickmarks should be

quartz() # will be windows() on a PC!

VisitorGraph2

# Want to create an interactive graph? Use plotly!

library(plotly)

ggplotly(VisitorGraph2)

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

# Part 8. Another ggplot graph (decide if time to do this one too...otherwise code will be posted online)

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

VisBoxplot <- ggplot(df5, aes(x = Code, y = kVis)) +

geom\_boxplot(aes(fill = Code)) +

theme\_bw() +

ggtitle("CA National Park Visitation (1950 - 2016)") +

xlab("National Park") +

ylab("Thousands of Visitors") +

scale\_x\_discrete(breaks = c("CHIS","DEVA","JOTR","KICA","LAVO","PINN","REDW","SEQU","YOSE"), labels = c("Channel Islands","Death Valley","Joshua Tree","Kings Canyon","Lassen Volcanic","Pinnacles","Redwoods","Sequoia","Yosemite")) +

theme\_bw() +

theme(axis.text.x = element\_text(angle = 45))

quartz()

VisBoxplot

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

# Part 9. Using the datasets in 'datasets' package for practice

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

# Show them the datasets package for practicing

?datasets

library(help = "datasets")

?USArrests # 1973 crime rates by state (show them to check documentation)

View(USArrests) # View it

# They can do that with ANY of the datasets they want for practice!