Mick’s R notes

- Helpful links
  - R for data science textbook https://r4ds.hadley.nz/data-visualize.html
  - Cheatsheets https://posit.co/resources/cheatsheets/
- Notation
  - Red is a variable that has already been defined by you, like a dataset
  - Purple is a variable that is being defined by the current code, like in a <- function
  - Green is an object R recognizes, like geom type (boxplot,), variable type (factor, string, etc.), function name, package name, etc
  - # is a section, ## is a subsection
  - ** is bolded text **
- Basic functions
  - RUNNING COMMANDS
    - Type in console at bottom to run functions without adding to doc (like the shell in python)
    - ```{r} starts a block of code, ``` ends it, cmd opt I gives automatically
  - DATA
    - ADDING
      - data=read.csv("/users/mickscriven/documents/R/patients101.csv") saves a csv named “patients101.csv” saved in the R folder in documents as the variable data
      - If your data file is in your current working directory, you ONLY NEED data = read_csv(“filename.csv”)
      - If your data file is in a SUBFOLDER of your working directory, you ONLY NEED data <- read_csv(“subfolder/filename.csv”)
      - TO FIND PATH: on the right window, click import data, from text (base). Open the file, this will open it in a new R tab. At the bottom in the console, you can copy the command to read the data and paste it into your r file.
    - GETTING INFO
      - summary(csvName), glimpse(csvName), str(csvName), gives summary of data in a csv (summary can give wrong numbers in Q1/median/means, etc if NAs in data)
      - table(csvName$columnName) is useful for factors, where it will tell you how many of each level is in the table
      - nrow(csvName) and ncol(csvName) gives number of rows and columns, dim(csvName) gives dimensions of csv
      - names(csvName) gives names or columns in csv
      - head(csvName) and tail(csvName) give first/last few rows of csv
    - CLEAN DATA
      - No numbers

- No uppercase
- Fixing column names:
    - janitor package has clean_names() function, which removes spaces, special characters, uppercase letters only from column names. Can add to read.csv(“blah”) %>% clean_names()
- Fixing data
    - Make uppercase entries lowercase by
        - data %>% mutate(across(c("col1","col2"), tolower))
        - data %>% mutate(across(where(is.character), tolower))
    - Replace -999 with NA
        - data %>% mutate(col1New=ifelse(col1 == -999.00, NA, col1))
            - Makes new col without -999, leaves old column there
        - data %>% mutate(col1=ifelse(col1 == -999.00, NA, col1))
            - Replaces old col with new one without -999
        - data %>% mutate(across(c(col1, col2, col3), ~ifelse(.==-999.00, NA, .)))
            - Multiple columns
        - data %>% mutate(across(where(is.numeric),~ifelse(. == -999.00, NA, .)))
            - All numeric columns

- VARIABLES
    - Assign variables with <- or =
    - Change variable type with as. Function (ex., as.factor)
    - FACTORS
        - Factors are categorical variables that have a fixed number of possible values called levels
        - as.factor(variable) changes variable to be a factor (you need to save to be permanent, ex. variable <- as.factor(variable))
        - levels(variable) gives the levels of a factor variable
- PACKAGES
    - Install package with install.packages(“package_name”)
    - To use, run library(“package_name”) at beginning of code file

- OPERATORS
    - ARITHMETIC
        - + (addition)
        - - (subtraction)
        - * (multiplication)
        - / (division)
        - ^ or ** (exponentiation)
        - %% (modulus - returns the remainder of a division)
        - %/% (integer division - returns the integer quotient)
        - %*% (matrix multiplication)
        - %o% (outer product)
        - %/% (integer division, also an arithmetic operator)
        - %% (modulus, also an arithmetic operator)
    - COMPARISON (returns Boolean true/false)
        - == (equal to)
        - != (not equal to)
        - < (less than)
        - > (greater than)
        - <= (less than or equal to)
        - >= (greater than or equal to)
    - VECTORS
        - c() creates vector out of values (vector is a list), can be numbers or strings
        - x:y (sequence operator - creates a sequence of numbers from x to y, good for making vectors)
        - %in% (membership operator - checks if an element is present in a vector)
        - vector_name[i] gives the i-th value in the vector “vector_name”
        - vector_name<=10 gives TRUE or FALSE for each item in vector of whether it is less than or equal to ten
            - vector_name[vector_name<=10] gives identity of all values that are TRUE for less than or equal to ten
    - MISC
        - %>% (pipe), feeds left side into right side
            - var1 %>%
              function1() %>%
              function2()
                - Feeds var1 into function1, then that output goes into function2
- TUTORIALS
    - ?aggregate opens a tutorial on the aggregate function (very useful!!!)

- tidyverse
  - USING
    - install.packages("tidyverse")
    - library("tidyverse")
  - TIBBLE (data.frame equivalent)
    - CREATING
      - tibble(name1=vector1, name2=vector2, name3=vector3) gives a table where each column is a vector, titled by the given name (name1, unless not specified, then it is the variable of the vector). Ex: column 1 is titled vector1 and has values of vector1. Under vector name, type of variable in each vector is specified (character, float, etc…)
    - GETTING INFO
      - names(tibble_name) gives names of columns
      - Also works for data frames: str(
      - Also works for data frames: dim(tibble_name) gives dimensions of table ex, 3x3
      - tibble_name[1,] gives first row of tibble, tibble_name[,3] gives third column, tibble_name[1,3] gives row 1 column 3 value
      - $ to access a specific column, for ex mean(tibble_name$column_name) gives the mean of the data in column titled column_name in tibble saved as tibble_name
    - MISC
      - write.csv(tibble_name, "file_name_you_want.csv", row.names = FALSE) saves a csv file of your tibble to your working directory. row.names = FALSE means it doesn't give you numbered rows
- Stats functions
  - MEAN
    - mean(data$column) gives mean of "column" for csv saved as "data"
  - STANDARD DEVIATION
    - sd(data$column) gives standard deviation
  - AGGREGATED DATA (mean, sd… separated by a category)
    - aggregate(weight ~ gender,data = data,mean) gives two means of the weight separated by gender in the data set "data"
  - LENGTH
    - length(data) gives number of things
- Graphs
  - GENERAL
    - TYPES OF DATA
      - discrete quantitative data that only contains integers
      - continuous quantitative data that can take any numerical value
      - categorical qualitative data that can take on a limited number of values

  - COLORS
    - col=”color” changes color of graph to “color” (ex., “red”, “lightblue”)
    - col=c(“color1”,”color2”) changes color to be different for each graph there is. For example, if you have two bar graphs, they will be different colors.
      - If you put in too many colors, it will just use the first relevant number of colors (2 bar graphs=2 colors)
  - TITLES
    - main=”title” gives the graph a title
    - xlab=”label” or ylab=”label” will give a label to the x and y axes
  - AXIS RANGES
    - ylim=c(0,200) sets the axis to start at zero and end at 200
- dplyr
  - select()
    - Pulls specified columns out of large dataframe to make small dataframe. Needs to be saved.
    - include: newData <- select(data, col1, col2) gives dataframe with only col1, col2
    - Include set: select(data, colStart:colEnd) to give dataframe including cols between (incl) colStart and colEnd
    - Exclude: select(data, !col1) removes col1 from dataframe
    - Exclude many: select(data, !c(col1,col2,col3)) removes col1, col2, col3 from dataframe
    - start/end with: select(data, starts_with(“blah”)) or select(data, ends_with(“blah”)) gives data starting/ending with “blah”
    - Object type: select(data, where(is.objectType)) gives only data that is that type, like numeric
    - Rename: you can rename within the select command, select(data, col1Rename=col1)
  - filter()
    - Takes data that is true for a certain variable, for ex taking only data with bodyMass=1.5, and makes new dataframe
    - filter(data, col1== “variable”)
      - gives only data where col1 is variable
    - filter(data, col1 >= number)
      - gives only data where col1 is > number
    - filter(data, col1 %in% c(number1, number2)) or filter(data, col1 %in% c(“variable1”, “variable2”))
      - gives data where col is number1 or number2, or variable1 or variable2
    - filter(data, between(col1, number1, number2) or filter(data, col1 %in% c(number1:number2))

    - gives data where col1 is between (incl) number1 and number2
  - filter(data, near(col1, number, tol=toleranceNumber))
    - Gives data where col1 is within toleranceNumber of number. tol is automatically set to a small number
  - filter(data, col1== “variable” & col2> number)
    filter(data, col1== “variable | col2>number)
    - & gives data where both conditions are met, | gives data where either condition is met
- arrange()
  - Arranges data by specified column, automatically in ascending order. Add desc(col1) to make it descending
  -
- rename()
  - Renames specified columns
  - rename(data, col1Rename=col1)
- relocate()
  - Move specified columns, automatically to front
  - relocate(data, col1) moves col1 to be first column
- distinct()
  - distinct(data) removes duplicate rows
  - data %>%
    distinct(variable)
    - Gives you each distinct level for that variable
  - data %>%
    distinct(variable1, variable2)
    - Gives you each distinct combination of those two variables
  - Add .keep_all T to keep the variables not mentioned in the code
  - n_distinct() tells you how many distinct variables
- mutate()
  - Adds new mutated column to end of dataset
  - mutate(data, newVar=oldVar/1000)
    - Adds new column named newVar, contains oldVar/1000
- summarize()
  - Takes dataframe, makes new dataframe with selected stat functions (mean, min, max, sd, n_distinct, etc.)
    - data %>%
      summarize_all(mean, na.rm=T)
      - gives a new data frame with one row showing means for all columns (removes NA values to avoid errors)

- data %>%
  summarize(across(where(is.numeric),
  ~mean(.x, na.rm=T)))
    - Does the same thing
-

- GGPLOT BASICS
    - ggplot(data=dataName,
      mapping=aes(x=xVarName,y=yVarName))+
      geom_boxplot()
    - + adds a layer
    - labels
        - Labels labs(title= “”, x= “”,
    - geom_
        - changes type of graph
        - Format: geom_type(mapping=aes())
        - Options include: boxplot, point (scatterplot), smooth
            - (line of best fit, requires method=lm for linear model, se=T to show error bars),
        - na.rm=T in geom_blah(HERE) removes error messages from NA datapoints
    - Examples
        - ggplot(data=penguins,
          mapping = aes(x=body_mass_g,y=flipper_length_mm,
          color=species))+
          geom_point(na.rm=T)+
          geom_smooth(method=lm, se=T)
            - Above gives scatterplot comparing body mass and flipper length, giving each point a different color by species, and giving a line of best fit for each species
        - ggplot(data=penguins,
          mapping = aes(x=body_mass_g,y=flipper_length_mm))+
          geom_point(mapping=aes(color=species))+
          geom_smooth(method=lm, se=T)
            - Moving color=species to the scatterplot aes instead of the whole plot’s means the line of best fit is not separated by species, so we only have 1
- HISTOGRAM
    - hist(data$column, main=”title”,xlab=”xlab”,ylab=”ylab”,col=”red”)
        - for csv “data” makes a histogram of column “column” with title “title”, x axis labeled “xlab” and y axis labeled “ylab”, colored red

- BOXPLOT
    - boxplot(GSK$sbp ~ GSK$exercise, GSK=GSK,main="systolic blood pressure",xlab="exercise level",ylab="systolic blood pressure",col=c("yellow","orange","red"))
        - gives boxplot of sbp separated by exercise (gives a different plot for each value of exercise) in data “GSK” with labels and a different color for each plot
- TABLE
    - stress_count <- table(GSK$stress)
        - Creates a table called “stress_count” of “stress” from “GSK” data set
    - SE.table=table(GSK$educatn,GSK$stress)
        - 2 way table of “educatn” and “stress” columns/rows in GSK data saved as “SE.table”
- BARPLOT
    - barplot(stress_count, main="barplot of stress levels",xlab="stress level",ylab="count of subjects",col=c("lightpink","lightblue","lightyellow"),ylim=c(0,200))
        - Barplot from table “stress_count” with name “barplot of stress levels” and x and y labels, colors of each bar, y axis range from 0-200
- MOSAIC PLOT
    - mosaicplot(SE.table,main="Mosaic Plot of Stress and Education",col=c("red","blue","green"))
        - Mosaic plot from 2 way table “SE.table” with name and colors for boxes
- CURVE
    - curve(dnorm(x, 50, 10), from=0, to=100, main="Normal Distribution", ylab="frequency", xlab="value", add=TRUE, lwd=2)
        - Curve takes a function like dnorm and graphs a curve
        - from=, to=, changes x range for graph
        - add=T or F choose whether to add to an existing graph
        - lwd= changes line width
- PLOT
    - plot(x,y, type=)
        - Takes x,y to create scatterplot or more complex function for other graph
        - type="p" for points, "l" for lines, "b" for both, "o" for both ‘overplotted’, "h" for ‘histogram’ like (or ‘high-density’) vertical lines, "s" for stair steps, "n" for no plotting.

- Probability
  - Distributions
    - GENERAL
      - Four fundamental items can be calculated for a statistical distribution:
        - Density or point probability
        - Cumulated probability, distribution function
        - Quantiles
        - Pesudo-random numbers
      - x can be number or a sequence
    - NORMAL DIST
      - pnorm(60, 50, 10)
        - Probability of 60 or less in a normal distribution with mean 50 sd 10
      - pnorm(60)
        - Probability of 60 or less in standard normal distribution (mean=0 sd=1)
      - pnorm(60, 50, 10) - pnorm(40, 50, 10)
        - P(60>Z>40)
    - BINOM DIST
      - dbinom(x, size = 50, prob = 1/3)
        - x can be number or sequence
    - SIMULATIONS
      - Runif, rnorm, rbinom, rpois(), rexp(), rchisq()
        - generates random numbers from a distribution, returns a vector
        - runif(n, min = 0, max = 1)
        - n: The number of random values to generate.
        - min: The lower limit of the distribution (default is 0).
        - max: The upper limit of the distribution (default is 1).
      - replicate
        - replicate(5, mean(runif(10)))
        - generates five means of 10 random uniform numbers