R

Getwd()

Dir() (says what’s in directory)

Change working directory to be able to read files

Write in editor and paste functions to R,

or save .R in working dir, and do source(“filename.R”), hit ls() to see what functions are in the file

write more functions and source again, pick which function to use

**think in vectors, even number 1 is a vector**

entering input

* <- assignment operator
* message <- ‘hello’ character vector
* x <- 1:20 creates sequence
* based on one-based indexing

Data Types: **VECTOR**

* Everything is an object, most basic object is **a vector**, which can have multiple objects of the same kind in one vector (all integers, can’t mix classes)
  + The one exception is a list, which is a vector which can contain different classes
  + Create vector with **vector(),** first argument is class type, second argument is length
  + Vectors
    - Atomic vectors have one data type
    - Lists have multiple
* 5 basic classes
  + character
  + numeric objects(real numbers)
    - most numbers, have to specify if want an integer
    - Inf (infinity) is also a number
    - NAN not a number, undefined value, considered a number
  + integers
    - if want an integer have to use L suffix
    - 1 is numerinc, 1L is integer
  + complex numbers
  + logicals (True false)
    - used to compare numbers 1 < 3 returns TRUE
    - since T is 1 and F is 0, if we want to find out hw many T are in a list, just add everything up!
    - NA v. NAN
* object attributes
  + names, dimnames
  + dimensions
  + class
  + length
  + user-defined attributes
  + **attributes() function**  lets us set or modify
* **c() function to create vector**
  + x<- c(1, 2), vector of two elements
  + x<- c(T, F)
  + x<-c(‘a’, ‘b’)
  + x<- vector(‘numeric’, length = 10)
  + creating x<- c(12, ‘a’) won’t givean error **but will pick the least common denominator, character is least here COERSION**
    - result will turn 12 into ‘12’ string to match ‘a
  + c(T, 12) will convert T into 1
  + EXPLICIT COERSIONif I want to convert something, use as.numeric(x) as.character, etc
  + If x is (0, 1, 2) as.character(x) will put (‘0’, ‘1’, ‘2’)
  + vectors of NA result if trying to convert character to numeric
  + put vector inside vector
  + can also do operations on items inside vector x\*2 + 20
  + c(1, 2, 3, 4) + c(0,10) returns 1 12 3 14, the shorter vector is **recycled**, the 3, 4 are not ignored
  + *vectors have to be multiples of each other, can’t add vector of len 4 to one of len 3*
* matrices
  + can only have a single class of data, unlike dataframes
  + vectors with dimension attribute which itself is an integer vector of length 2 (r\*c)
  + r \* c
  + m<- matrix(nrow=2, ncol=3) initializes with NA values
  + dim(m) gives 2,3
  + 1. m<- matrix(1:6, nrow=2, ncol=3) puts in 1-6 starting top left column and going down
  + 2. if I already have a vector with numbers m <- 1:10
    - I can transform the vector into a matrix dim(m) <- c(2,5)
  + 3. Bind rows and columns
    - x<- 1:3
    - y<- 10:12
    - **cbind**(x,y) will put them into columns xy
    - **rbind**(x,y) will put it horizontally into rows x y
  + **dim()** tells dimensions and can set dimensions of anything including vectors **length()** for vectors
  + **attributes()**
  + convert vector into matrix by giving it dim(x) <- 4,5
  + work with matrix
    - **cbind()** can combine a vector into a matrix
      * **make sure matrix has only one class of data**
    - **data.frame()** will allow to have multiple classes of data (header and numbers)
    - **colnames()** sets header (names of column)
* lists
  + vector that can have elements of any class
  + x<- list(1, ‘a’, True)
  + prints out by each index
* sequences
  + in R they are inclusive so 1:20 prints all numbers
  + if I put pi:10, it will print 3.14… add one but until 9.14… because the next number is actually greater than 10
  + 15:1 also backwards
  + seq(0,10, by =0.5)
  + s eq(5, 10, length=30), gives dandom numbers between 5,10
* factions
  + rep()
    - function to replicate
    - rep(c(0,1,2), times =10) replicates 012 10 times
    - rep(c(0, 1, 2), each = 10) replicates each number 10 times
  + paste()
    - paste(my\_char, collapse = " ") creates “My name is”
    - paste("Hello", "world!", sep = " ")
    - paste(1:3, c("X", "Y", "Z"), sep = "") prints 1x 2y 3z
    - paste(LETTERS, 1:4, sep = "-") uses character recycling since 1:4 is shorter than 26 letters
  + is.na()
    - tells you where na are
* LETTERS is a predefined character with all letters in dict
* Indexing
  + With logical vectors
    - x[is.na(x)] gives all the nas in the vector
    - x[!is.na(x)] takes out nas
    - x [!is.na(x) & x > 0] gives me no nas and positive numbers
  + with integer vector
    - x[c(3, 5,7)] gets me the 3rd 5th and 7th element
    - x[c(-2, -10)] gives all the elements except the 2nd and 10th elements, can also write x[-c(2, 10)]
  + with names
    - no need to put ‘ ‘ as string names
    - names(x) gives the string names
    - can add names later by doing name(x) <- ….
    - Call the index x[“name”]

**Work with df**

**? in front of any command for help**

Import csv with df <- read.csv(“filename”)

**View data**

View(df) shows df

?df in R studio gives you more info about the df in the Help tab

str(df) gives more info

summary(df) mean, quartiles of columns

dim(df) gives dimensions of df

head(data)

names(df) tells you variable names (column names)

data$column gives only that column

**table(df$column)** aggregates all values

levels(df$column) gives levels

**subsetting:**

subsest(df, variable == something) only gives those variables with that something

df[df$column == something, ] #another way to subset, not having anything on the right means I want to return all columns, otherwise specify columns

subset(df, var1 > 30 & var2 > 100)

& and | or

Df[r,c] df[,1] gives first column

Df[df$var > 3,] row

Df[**which**df$var > …] takes out NAs for that logital statement

**save new subset output into variables**

mean(data$column) gives mean of that column

*mean(x, na.rm=TRUE) to get mean of df by taking out NAs*

*mean(df$Temp[df$Month == 6], na.rm = TRUE) mean of a column when another one is equalto something*

*max(df$Ozone[df$Month == 5], na.rm = TRUE)*

*max value of a column with respect to another one*

Print rows, first 4 df[1:4,]

Without comma prints columns

**Nrow(df)** gets numer of rows, Ncol n of columns

mean(df$Ozone, na.rm = TRUE) gets mean of df of column Ozone and takes out nas

**completes.cases()** takes out all rows with any NA

**create and drop vars**

df$var 🡨 something Creates new var

df <- subset(df, select = -var) to drop var, subset first

to create a new var using *ifelse*

data$new\_var 🡨 ifelse(cond, ‘what to call if I'm True’, ‘what to call if I’m False’)

**RMD file**

* Put R code along with other types of code
* Uses Markdown, easier than HTML
* Knit HTML button can let me publish my work
* Plot(df) and show graph, use Knit HTML to show it

**Plotting data, using qplot, can also use ggplot2, one layer at a time buildup**

* **Scatterplot** : qplot(x= var, y= var, data = mydata)
* Use names(df) to get all variable names
* **Ggplot(aes(x= var, y= car), data = mydata) + geom.\_histogram()**
* **Ggplot uses + to add new layers**
* *Qplot(data = mydata, x = column)*
* **qplot(x=var, data = mydata) +**

**scale\_x\_discrete(breaks=1:31)**

**scalex continuous** changes the numbers on the x axis, breaks say which days, can add a lim that limits x axis

* use + facet\_wrap(~var, ncol =#) to split a bin (multiple graphs within one)
* **facet\_wrap** takes in a formula inside
* + **facet\_grid**(vertical ~horizontal) better for 2 or more vars
* , **xlim** = c(0, 1000) *limits x axist to 1000*
* **coord\_cartesian(xlim = c()) is better than xlim bc it doesn’t take out data**
* , binwidth = 50
* **take na out**
* **qplot(data = subset(df,!is.na(gender)), x=friend\_count, binwidth = 25) +..**
* **change name of x-y axes**
  + after data = my data, xlab=”name of x”, ylab= ‘name of y”, color info…
* **taking log of my x variable will show me the log of x, but doing scale\_x\_log10 will log the data but have the x axis as the actual x values**
* ggplot … + geom\_point(alpha = 1/20) cleans many points, one point will represent 20
* **+ geom\_ point()** for scatterplots

**EXAMPLE**

**qplot(x=tenure/365, data = facebook, binwidth = .25, xlab= "Number of years using FB", ylab= "Number of users", color = I('black'), fill = I('#F79420')) + scale\_x\_continuous(breaks = seq(1,7,1), lim=c(0,7))**

create plot of tenure but divides it by 365 to get years not days, data is facebook df, binwidth is .25 to see less than 1 year, name of y and x axes chosen, color of bars, scale\_x\_continuous changes scale of x axes and limits it, goes from 1 to 7 and by a scale of 1

add binwidth = something to change how many ranges go into one bar

**ggplot(aes(x= www\_likes\_received, y = likes\_received), data = facebook) + geom\_point(alpha = 1/20) +**

**xlim(0, quantile(facebook$www\_likes\_received, 0.95)) +**

**ylim(0, quantile(facebook$likes\_received, 0.95)) +**

**geom\_smooth(method = "lm", color = "red")**

created a scatterplot of www likes v. likes, added line of best fit in red, limited axes to include only data up to 95th quartile

**scatterplot matrix**

* use GGally package
* and use ggpairs()

**Ordering factors**

* to make graphs in order
* if I have ranges of income, I want those ranges to be in order
* df$column**<- factor(df$column, levels = c('Under 18', '18-24', '25-34', '35-44', '45-54', '55-60', '65 or Above'))**

**xlim v. coord\_cartesian**

* use xlim when I want to limit axis and want to eliminate the data out of the range. So statistics will not include eliminated data
* coord\_cartesian limits axis but still calculates with data that is out of range

**coord\_trans v. coord\_cartesian**

* the last one applied will be the one that is used, can’t use both at once

Summary stats

* **by(variable I want something of, categorical variable to split first variable, function)**
  + **by(data$columnvar, data$categvar, summary)**
    - **by(df$friend\_count, df$gender, summary)**
    - this gives stats of friend\_count of each gender
  + funcdcation can be summary, sum,
* transform data by taking the **log** or **sqr** of the variable and getting the summary and qplot
* **taking log of my x variable will show me the log of x, but doing scale\_x\_log10 will log the data but have the x axis as the actual x values**
  + x=log(myvariable) or +scale\_x\_log10() after info

**create new Variables**

facebook$mobile\_checkin <-ifelse(facebook$mobile\_likes > 0, 1, 0)

#if their mobile\_likes were greater than 0, give value of 1, otherwise give value of 0

**rename variables**

names(agriculture)[1] <- "country"

renames first column to country

**subset data**

agriculture\_selected\_country <- subset(agriculture,agriculture$country %in% c("Australia","Brazil","China","Germany","Japan","United States"))

**dplyr package**

pairing raw data plots with summary data plots

* **Group by()**
  + Grouped\_df 🡨- Group\_by(df, variable) groups the dataframe by groups of that variable and subsequent operations on this new df are applied on each group
  + library(dplyr)
  + by\_clarity <- group\_by(diamonds, clarity)
  + diamondsByClarity <- summarize(by\_clarity,
  + n=n(),
  + mean\_price = mean(price),
  + median\_price = median(price),
  + min\_price = min(price),
  + max\_price = max(price))
* **Mutate()**
* **Fill()**
* **Arrange()**

**Statistics**

* It’s important to measure the correlation between two variables because high correlation between variables messes up regression analysis bc regression assumes no correlation, that they are independent variables
* **cor.test(df$var1, df$var2, method = "pearson")**
* visualize by adding line of best fit layer to ggplot
* **+ geom\_smooth(method = "lm", color = "red")**
* **create linear model using lm() function lm(y~x) (dependent , indep)**
  + **use the I() wrapper around each variable if need to transform vars, so R reads vars as they are**
  + [**http://www.theanalysisfactor.com/interpreting-regression-coefficients/**](http://www.theanalysisfactor.com/interpreting-regression-coefficients/)
  + **logistic regression**
    - mylogit <- glm(dependent ~ indep + indept + …, data = my data, family = “binomial”
    - if there are any categorical variables, convert them to a factor in R
    - summary(mylogit) to print it
    - **http://www.ats.ucla.edu/stat/r/dae/logit.htm**

**Wide-long format**

* **long:** many rows and the vars are repeated for each (year, countries are repeated for each year)
* **Wide**: one row per age, one column for male and female
* Use reshape2() package
* Use **dcast(old df, vars I want to keep ~ vars I want to have own column, value.var = “”)**
* **Melt()**

**Sampling**

* Create a sample using set.seed() to set the seed and sample(levels(df$var, number)

**Multiple dfs**

* Can merge dataframes if have variables in common
* Use the merge(df, df, by = c(“var”, “var”

**Can write my own functions in R**

[**https://www.youtube.com/watch?v=Z1wB1rHAYzQ&list=PLOU2XLYxmsIK9qQfztXeybpHvru-TrqAP**](https://www.youtube.com/watch?v=Z1wB1rHAYzQ&list=PLOU2XLYxmsIK9qQfztXeybpHvru-TrqAP)

**JSON IN R – converts to dataframe**

install.packages("**jsonlite**")

library(jsonlite)

setwd('/Users/Dana/Dropbox/dataScience/Pizza')

pizza <-fromJSON("train.json")

colnames(pizza)