Code Library

Ruhika Chatterjee

2024-12-07

Notes taken from Johns Hopkins University Coursera course series Data Science Specialization.

R Data Types

- Basic object is vector of same class except list.
- Atomic classes of objects: character, numeric (real), integer, complex, logical.
- Attributes can include names, dimnames, dimensions, class, length.

Atomic Data

```
# numeric Vector
x <- 5 # numeric vector of 1 element

# integer vector
x <- 5L # integer vector of len 1

x <- Inf # special number infinity, +/-
x <- NaN # special number undefined, usually hijacks operations

# character vector
msg <- "hello" # char vector of len 1

# logical vector
tf <- TRUE # logical vector of value true
# TRUE = 1 = T, FALSE = 0 = F, num > 0 = TRUE

# complex vector
x <- 1+4i # vector of complex num of len 1
```

complex Data Types

```
# vector
vector("numeric", length = 10) # create vector of one type, args: class, length
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0
```

```r
c(1,2,3,4) # creates vector of common denominator class with given values
```

```
## [1] 1 2 3 4
```

```r
1:20 # vector sequence of 20 elements 1-20
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```r
pi:10 # will not exceed 10, start from pi, increment by 1
```

```
## [1] 3.141593 4.141593 5.141593 6.141593 7.141593 8.141593 9.141593
```

```r
15:1 # increment -1
```

```
##  [1] 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
```

```r
seq(1,20) # same as :
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```r
seq(0,10,by=0.5) # to change increment
```

```
##  [1]  0.0  0.5  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5  6.0  6.5  7.0
## [16]  7.5  8.0  8.5  9.0  9.5 10.0
```

```r
seq(5,10,length=30) # to not set increment but number of numbers
```

```
##  [1]  5.000000  5.172414  5.344828  5.517241  5.689655  5.862069  6.034483
##  [8]  6.206897  6.379310  6.551724  6.724138  6.896552  7.068966  7.241379
## [15]  7.413793  7.586207  7.758621  7.931034  8.103448  8.275862  8.448276
## [22]  8.620690  8.793103  8.965517  9.137931  9.310345  9.482759  9.655172
## [29]  9.827586 10.000000
```

```r
seq_along(x) # vector of same length 1:length(x)
```

```
## [1] 1
```

```r
rep(10, times = 4) # repeats 10 4 times in vector
```

```
## [1] 10 10 10 10
```

```r
rep(c(0, 1, 2), times = 10) # repeats sequence of vector 10 times. Arg each can be used to repeat first
```

```
##  [1] 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2
```

```r
# lists
# vector capable of carrying different classes
x <- list(1, "a", TRUE, 1+4i) # vector of vectors

# Matrix
# vector of single class with rectangular dimensions (attribute of integer vector len 2)
x <- matrix(nrow=2,ncol=3) # empty matrix of given dimensions
x <- matrix(1:8, nrow = 4, ncol = 2) # creates matrix of given dimensions with values assigned, created
y <- matrix(rep(10,4),2,2) # creates matrix of 4 10s

x <- 1:10
dim(x) <- c(2,5) # creates matrix out of vector with dimension 2 rows x 5 columns

cbind(1:3,10:12) # creates matrix out of values in vector args, adding by column (1st arg = 1st col)
```

```
##      [,1] [,2]
## [1,]    1   10
## [2,]    2   11
## [3,]    3   12
```

```r
rbind(1:3,10:12) # same but using rows
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]   10   11   12
```

```r
# factors
# self-describing type of vector representing categorical data, ordered or unordered (labels)
x <- factor(c("male","female","female","female","male")) # character vector with specific linear modeli
f <- gl(3,10) # factor 3 levels, 10 times each
table(x) # prints counts of each factor
```

```
## x
## female   male
##      3      2
```

Data Frames

```r
# stores tabular/rectangular data, stored as lists of same length where each element is a column, lengt
x <- data.frame(foo=1:4, bar=c(T,T,F,F)) # creates data frame 2 columns foo and bar, 4 rows unnamed. Ca
x <- read.table(file = "hw1_data.csv", header = TRUE, sep = ",") # read in data from file
x <- read.csv("hw1_data.csv") # same
```

Date and Time Data Types

```r
# useful for time-series data (temporal changes) or other temporal info
# lubridate package by Hadley Wickham
```

```r
# Dates and Times
birthday <- as.Date("1970-01-01") # dates are date class defined by converting character string, year-m
today <- Sys.Date()

currentTime <- Sys.time()# time by POSIXct(large integer vector, useful in dataframe) or POSIXlt(list,
timedefined <- as.POSIXct("2012-10-25 06:00:00") # convert char vector, can define timezone
cTConvert <- as.POSIXlt(currentTime) # reclass, works other way
cTConvert$min # to subset list
```

```
## [1] 35
```

```r
datestring <- c("January 10, 2012 10:40", "December 9, 2011 9:10")
x <- strptime(datestring, "%B %d, %Y %H:%M") # Convert character vector to POSIXlt by defining format (
x
```

```
## [1] "2012-01-10 10:40:00 EST" "2011-12-09 09:10:00 EST"
```

```r
weekdays(birthday) # return day of week, date or time classes
```

```
## [1] "Thursday"
```

```r
months(birthday) # return month on date or time
```

```
## [1] "January"
```

```r
quarters(birthday) # return quarter of date or time
```

```
## [1] "Q1"
```

```r
# Operations
# CANNOT MIX CLASSES - convert
# add and subtract dates, compare dates
currentTime - timedefined # time difference, track of discrepancies (i.e. daylightsavings, timezones, l
```

```
## Time difference of 4471.816 days
```

```r
difftime(currentTime, timedefined, units = "days") # to specify unit
```

```
## Time difference of 4471.816 days
```

```r
rm(list=ls())
```

Basic R Functions

Functions and Operations

```r
# Input and Evaluation
x <- 1 # assignment operator, evaluates and returns
print(x) # print value as vector
```

```
## [1] 1
```

```r
x # auto-prints
```

```
## [1] 1
```

```r
# in console, press Tab for auto-completion
LETTERS # predefined character vector of capital letters
```

```
##  [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S"
## [20] "T" "U" "V" "W" "X" "Y" "Z"
```

```r
# <<- operator can be used to assign a value to an object in an environment that is different from the

# Mathematical and Statistical Functions
5 + 7 # basic arithmetic operations all work +, -, *, /, ^, %% (modulus). NA affects operation.
```

```
## [1] 12
```

```r
sqrt(4) # square root
```

```
## [1] 2
```

```r
abs(-1:2) # absolute value
```

```
## [1] 1 0 1 2
```

```r
mean(c(3,4,5,6,7)) # return mean of numeric vector
```

```
## [1] 5
```

```r
sd(c(3,4,5,6,7)) # returns standard deviation of numeric vector
```

```
## [1] 1.581139
```

```r
cor(c(3,4,5,6,7), c(61,47,18,18,5)) # correlation of x and y vectors make sure to set arg use for NAs
```

```
## [1] -0.9587623
```

```r
range(c(3,4,5,6,7)) # returns min and max as numeric vector of 2
```

```
## [1] 3 7
```

```r
quantile(c(3,4,5,6,7), probs = 0.25) # returns 25th percentile
```

```
## 25%
##   4
```

```r
-c(0.5,0.8,10) # distributes the negative to all elements of vector
```

```
## [1]  -0.5  -0.8 -10.0
```

```r
# vectorized operations
x <- 1:4; y <- 6:9 # different length vectors
x + y # add the elements of the vectors, all operators work
```

```
## [1]  7  9 11 13
```

```r
x > 2 # returns logical vector, >= or == or any of the logical expressions work
```

```
## [1] FALSE FALSE  TRUE  TRUE
```

```r
# Matrix Operations
x <- matrix(1:4,2,2); y <- matrix(rep(10,4),2,2)
x * y # element wise multiplication, for all operators
```

```
##      [,1] [,2]
## [1,]   10   30
## [2,]   20   40
```

```r
x %*% y # matrix multiplication
```

```
##      [,1] [,2]
## [1,]   40   40
## [2,]   60   60
```

```r
x <- matrix(rnorm(200), 20, 10)
rowSums(x) # vector of sum of rows
```

```
##  [1] -2.36607726  2.58132693 -1.58357390 -0.30170021  1.32333808 -5.40170402
##  [7]  1.82759094 -2.26658577  0.85118959 -3.18900482 -0.08864629  1.88340356
## [13] -1.12676539 -0.64100195 -0.90170208  4.16513759  0.20584043  0.48951395
## [19]  2.73265494 -6.11667994
```

```r
rowMeans(x) # vector of mean of rows
```

```
##  [1] -0.236607726  0.258132693 -0.158357390 -0.030170021  0.132333808
##  [6] -0.540170402  0.182759094 -0.226658577  0.085118959 -0.318900482
## [11] -0.008864629  0.188340356 -0.112676539 -0.064100195 -0.090170208
## [16]  0.416513759  0.020584043  0.048951395  0.273265494 -0.611667994
```

```r
colSums(x) # vector of sum of cols
```

```
##  [1] -4.64128373 -3.40717135  2.85371776 -1.91298829  0.54882800 -0.03673313
##  [7] -7.70075721  2.03744699  3.87043332  0.46506202
```

```r
colMeans(x) # vector of mean of cols
```

```
##  [1] -0.232064186 -0.170358568  0.142685888 -0.095649415  0.027441400
##  [6] -0.001836657 -0.385037861  0.101872350  0.193521666  0.023253101
```

```r
x <- matrix(rnorm(100), 10, 10)
solve(x) # returns inverse of matrix if invertible
```

```
##              [,1]        [,2]       [,3]        [,4]        [,5]          [,6]
##  [1,]  0.16606090  0.06286440 -0.3203517 -0.07571466  0.22933025 -0.109441248
##  [2,]  0.01438419 -0.01557975 -0.4293080  0.09997152 -0.16847085  0.140956417
##  [3,]  0.18523734  0.16111307 -0.5494279  0.00608573  0.08351094  0.080562824
##  [4,]  0.09920347 -0.17942066  0.1435977 -0.03218845  0.67558983  0.074493678
##  [5,] -0.32821829  0.28327191  0.1966371  0.24675930 -0.55214933  0.005863597
##  [6,] -0.22981251  0.54110445 -0.4452845  0.10940284 -0.42532043  0.214634769
##  [7,]  0.28816289  0.02769830 -0.3293377 -0.09844345 -0.06642845  0.175764506
##  [8,]  0.13326119 -0.07428055  0.1940233  0.18144522  0.27217310 -0.197173526
##  [9,]  0.07683688 -0.39693081 -0.2421137 -0.15487371  0.38259576 -0.115914528
## [10,] -0.40293935  0.85959609  0.1303149  0.43072910 -1.29819535  0.281413177
##              [,7]        [,8]        [,9]       [,10]
##  [1,] -0.22282661 -0.03187886 -0.71217658  0.12542443
##  [2,]  0.08361336 -0.06745472  0.07381881 -0.12521413
##  [3,] -0.05520660 -0.07268550  0.06129090  0.41693927
##  [4,] -0.29345918  0.09045872 -1.03671650  0.43806046
##  [5,]  0.11371267 -0.15774802  0.22499243 -0.31700861
##  [6,]  0.31262875 -0.19660489  0.91044356 -0.35268473
##  [7,]  0.36310037 -0.10535400  0.13884905  0.04248222
##  [8,]  0.07828105  0.11107000 -0.43213192  0.35235775
##  [9,] -0.18241960 -0.24547174  0.05005525  0.13375584
## [10,]  0.24477764 -0.03439356  1.81544938 -0.89914157
```

```r
# Logical operators
5 >= 2 # returns logical. <, >, <=, >=, ==, !=. NA in expression returns NA. Can also use to compare lo
```

```
## [1] TRUE
```

```r
TRUE | FALSE # OR A|B union, AND A&B intersection, NOT !A negation. & operates across vector, && evalua
```

```
## [1] TRUE
```

```r
isTRUE(6 > 4) # also evaluates logical expression
```

```
## [1] TRUE
```

```r
xor(5 == 6, !FALSE) # only returns TRUE if one is TRUE, one is FALSE
```

```
## [1] TRUE
```

```r
which(c(1,2,3,4,5,6) < 2) # returns indices of logical vector where element is TRUE
```

```
## [1] 1
```

```r
any(c(1,2,3,4,5,6) < 2) # returns TRUE if any of the logical index values are TRUE
```

```
## [1] TRUE
```

```r
all(c(1,2,3,4,5,6) < 2) # returns TRUE only if all the elements of vector are TRUE
```

```
## [1] FALSE
```

```r
# Character functions
paste(c("My","name","is"),collapse = " ") # join elements into one element, can join multiple vectors w
```

```
## [1] "My name is"
```

```r
c (c("My","name","is"), "Bob") # add to the vector
```

```
## [1] "My"   "name" "is"   "Bob"
```

```r
# Factors functions
x <- factor(c("male","female","female","female","male")) # can include levels argument to set order (ba
x # prints values in vector and levels
```

```
## [1] male   female female female male
## Levels: female male
```

```r
table(x) # prints labels and counts present
```

```
## x
## female   male
##      3      2
```

```r
unclass(x) # strips class to integer with levels of labels
```

```
## [1] 2 1 1 1 2
## attr(,"levels")
## [1] "female" "male"
```

```r
# Display Data Functions
print(data.frame(foo = 1:20, rar = 301:320)) # print whole object
```

```
##    foo rar
## 1    1 301
## 2    2 302
## 3    3 303
## 4    4 304
## 5    5 305
## 6    6 306
## 7    7 307
## 8    8 308
## 9    9 309
## 10  10 310
## 11  11 311
## 12  12 312
## 13  13 313
## 14  14 314
## 15  15 315
## 16  16 316
## 17  17 317
## 18  18 318
## 19  19 319
## 20  20 320
```

```r
head(data.frame(foo = 1:20, rar = 301:320)) # prints preview of first 6 lines
```

```
##   foo rar
## 1   1 301
## 2   2 302
## 3   3 303
## 4   4 304
## 5   5 305
## 6   6 306
```

```r
tail(data.frame(foo = 1:20, rar = 301:320)) # prints preview of last 6 lines
```

```
##    foo rar
## 15  15 315
## 16  16 316
## 17  17 317
## 18  18 318
## 19  19 319
## 20  20 320
```

```r
table(c(1,1,1,2,2,2,2,2,2,2,2,2,3,3,3,3,4,4,5)) # returns table of counts
```

```
##
## 1 2 3 4 5
## 3 9 4 2 1
```

```r
summary(c(3,4,5,6,7)) # result summaries of the results of various model fitting functions based on cla
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       3       4       5       5       6       7
```

```r
unique(c(3,4,5,6,7,3,3,5,7,2,8,3,5,6)) # returns only unique elements, duplicates removed
```

```
## [1] 3 4 5 6 7 2 8
```

```r
# str function - compactly display internal structure of R object (esp large lists). Diagnostic, alterna
str(unclass(as.POSIXlt(Sys.time()))) # prints list clearly
```

```
## List of 11
##  $ sec   : num 39.2
##  $ min   : int 35
##  $ hour  : int 0
##  $ mday  : int 22
##  $ mon   : int 0
##  $ year  : int 125
##  $ wday  : int 3
##  $ yday  : int 21
##  $ isdst : int 0
##  $ zone  : chr "EST"
##  $ gmtoff: int -18000
##  - attr(*, "tzone")= chr [1:3] "" "EST" "EDT"
##  - attr(*, "balanced")= logi TRUE
```

```r
str(lm) # list of function arguments
```

```
## function (formula, data, subset, weights, na.action, method = "qr", model = TRUE,
##     x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE, contrasts = NULL,
##     offset, ...)
```

```r
str(rnorm(100,2,4)) # type of vector, length, first 5 elements
```

```
##  num [1:100] 0.6163 2.058 4.3763 -0.199 -0.0949 ...
```

```r
str(gl(40,10)) # for factors
```

```
##  Factor w/ 40 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```r
# Missing Values
# represented as NA (missing, with specified class) or NaN (missing or undefined)
# NaN is NA but NA not always NaN
is.na(c (1,2,NA,5,6,NA, NA,3, NaN)) # output logical vector of length of input
```

```
## [1] FALSE FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE  TRUE
```

```r
is.nan(c (1,2,NaN,5,6,NA, NaN,3)) # output logical vector of length of input
```

```
## [1] FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE
```

Attributes of objects

```r
x <- c(0.5,105,10,0.1,2)
class(x) # determine class of object
```

```
## [1] "numeric"
```

```r
attributes(x) # function to return or modify attributes of object
```

```
## NULL
```

```r
identical(x,x) # returns logical for if two objects are identical
```

```
## [1] TRUE
```

```r
length(x) # to specifically get the length of vector
```

```
## [1] 5
```

```r
dim(x) # to get dimensions of matrix, data frame (row, column)
```

```
## NULL
```

```r
object.size(x) # return memory occupied in bytes
```

```
## 96 bytes
```

```r
as.numeric(0:6) # explicit coercion, works on all atomic classes, if not possible converts to NA and wa
```

```
## [1] 0 1 2 3 4 5 6
```

```r
# data frames
row.names(x) # get and set row names (attributes). Can also use rownames(x)
colnames(x) # get and set row names
```

```
## NULL
```

```r
nrow(x) # number of rows
```

```
## NULL
```

```r
ncol(x) # number of columns
```

```
## NULL
```

```r
data.matrix(x) # converts data frame to matrix, coercion
```

```
##       [,1]
## [1,]   0.5
## [2,] 105.0
## [3,]  10.0
## [4,]   0.1
## [5,]   2.0
```

```r
dim(x) # (row, column) dimensions of data frame
```

```
## NULL
```

```r
# names attribute
x <- 1:3
names(x) # is null
```

```
## NULL
```

```r
names(x) <- c("foo","bar","norf") #now not numbered vector but named, print x and names(x) with names
vect <- c(foo = 11, bar = 2, norf = NA) # adds elements with names to vector directly
# also for lists, names vectors not items
m <- matrix(1:4,nrow = 2, ncol = 2)
dimnames(m) <- list(c("a","b"),c("c","d")) # each dimension has a name for matrices, rows names then co
```

Indexing, Subsetting, and Dealing with NAs

```r
# Subsetting R Objects
x <- c("a","b","c","c","d","a")
x[1] # more than one element extracted, returns same class as the original, numeric/logical index
```

```
## [1] "a"
```

```r
x[1:4] # sequence of num index
```

```
## [1] "a" "b" "c" "c"
```

```r
x[x>"a"] # logical indexing, returns vector where logical is true
```

```
## [1] "b" "c" "c" "d"
```

```r
u <- x > "a" # create logical vector
x[u] # same as x[x>"a"]
```

```
## [1] "b" "c" "c" "d"
```

```r
x[!is.na(x) & x > 0] # returns only positive, non NA values
```

```
## [1] "a" "b" "c" "c" "d" "a"
```

```r
x[c(-2, -10)] # returns vector with 2nd and 10th elements removed
```

```
## [1] "a" "c" "c" "d" "a"
```

```r
x <- data.frame(foo = 1:6, bar = c("g","h","i","j","k","l"))
x[[which(x$bar == "h"), "foo"]] # get or set foo in the same row as bar of "h"
```

```
## [1] 2
```

```r
x <- list(foo = 1:4, bar = 0.6, baz = "hello")
x[1] # list containing first element
```

```
## $foo
## [1] 1 2 3 4
```

```r
x[[1]] # extract from list/data frame, single element, class can change. Ex, numerical vector returned
```

```
## [1] 1 2 3 4
```

```r
x$bar # like [[]] but by name. Ex, return num vector 0.6. Equivalent to x[["bar"]]. Expression x["bar"]
```

```
## [1] 0.6
```

```r
x[c(1,3)] # multiple object extraction from list, returns list
```

```
## $foo
## [1] 1 2 3 4
##
## $baz
## [1] "hello"
```

```r
name = "foo"
x[[name]] # must be used if using computed index
```

```
## [1] 1 2 3 4
```

```r
x[1][3] # return element in element in object
```

```
## $<NA>
## NULL
```

```r
x[[c(1,3)]]
```

```
## [1] 3
```

```r
# Subsetting Matrix
x <- matrix(1:6, 2, 3)
x [1,2] # returns vector len 1, different that x[2,1]. Get matrix using arg drop = FALSE.
```

```
## [1] 3
```

```r
x[1,] # get num vector of first row, can also get col x[,2]. drop = FALSE also works
```

```
## [1] 1 3 5
```

```r
# Removing NA values
x <- c(1,2,NA,4,NA,5)
bad <- is.na(x) # logical vector indicating presence of NA
x[!bad] # removes NA values
```

```
## [1] 1 2 4 5
```

```r
x[!is.na(x)] # simplified returns vector removing NA values
```

```
## [1] 1 2 4 5
```

```r
x <- c(1,2,NA,4,NA,5) # for two vectors
y <- c("a","b",NA,"d",NA,"f")
good <- complete.cases(x,y) # logical vectors where there is no NA in either list
x[good]
```

```
## [1] 1 2 4 5
```

```r
y[good]
```

```
## [1] "a" "b" "d" "f"
```

```r
# Sum of NA values
my_na <- is.na(x)
sum(my_na)
```

```
## [1] 2
```

```r
x <- read.csv("hw1_data.csv") # for data frames
goodVals <- complete.cases(x) # complete rows in the data frame
x[goodVals,]
```

```
##     Ozone Solar.R Wind Temp Month Day
## 1      41     190  7.4   67     5   1
## 2      36     118  8.0   72     5   2
## 3      12     149 12.6   74     5   3
## 4      18     313 11.5   62     5   4
## 7      23     299  8.6   65     5   7
## 8      19      99 13.8   59     5   8
## 9       8      19 20.1   61     5   9
## 12     16     256  9.7   69     5  12
## 13     11     290  9.2   66     5  13
## 14     14     274 10.9   68     5  14
## 15     18      65 13.2   58     5  15
## 16     14     334 11.5   64     5  16
## 17     34     307 12.0   66     5  17
## 18      6      78 18.4   57     5  18
## 19     30     322 11.5   68     5  19
## 20     11      44  9.7   62     5  20
## 21      1       8  9.7   59     5  21
## 22     11     320 16.6   73     5  22
## 23      4      25  9.7   61     5  23
## 24     32      92 12.0   61     5  24
## 28     23      13 12.0   67     5  28
## 29     45     252 14.9   81     5  29
## 30    115     223  5.7   79     5  30
## 31     37     279  7.4   76     5  31
## 38     29     127  9.7   82     6   7
## 40     71     291 13.8   90     6   9
## 41     39     323 11.5   87     6  10
## 44     23     148  8.0   82     6  13
## 47     21     191 14.9   77     6  16
## 48     37     284 20.7   72     6  17
## 49     20      37  9.2   65     6  18
## 50     12     120 11.5   73     6  19
## 51     13     137 10.3   76     6  20
## 62    135     269  4.1   84     7   1
## 63     49     248  9.2   85     7   2
## 64     32     236  9.2   81     7   3
## 66     64     175  4.6   83     7   5
## 67     40     314 10.9   83     7   6
## 68     77     276  5.1   88     7   7
## 69     97     267  6.3   92     7   8
## 70     97     272  5.7   92     7   9
## 71     85     175  7.4   89     7  10
## 73     10     264 14.3   73     7  12
## 74     27     175 14.9   81     7  13
## 76      7      48 14.3   80     7  15
## 77     48     260  6.9   81     7  16
## 78     35     274 10.3   82     7  17
## 79     61     285  6.3   84     7  18
```

```
## 80     79     187  5.1   87     7  19
## 81     63     220 11.5   85     7  20
## 82     16       7  6.9   74     7  21
## 85     80     294  8.6   86     7  24
## 86    108     223  8.0   85     7  25
## 87     20      81  8.6   82     7  26
## 88     52      82 12.0   86     7  27
## 89     82     213  7.4   88     7  28
## 90     50     275  7.4   86     7  29
## 91     64     253  7.4   83     7  30
## 92     59     254  9.2   81     7  31
## 93     39      83  6.9   81     8   1
## 94      9      24 13.8   81     8   2
## 95     16      77  7.4   82     8   3
## 99    122     255  4.0   89     8   7
## 100    89     229 10.3   90     8   8
## 101   110     207  8.0   90     8   9
## 104    44     192 11.5   86     8  12
## 105    28     273 11.5   82     8  13
## 106    65     157  9.7   80     8  14
## 108    22      71 10.3   77     8  16
## 109    59      51  6.3   79     8  17
## 110    23     115  7.4   76     8  18
## 111    31     244 10.9   78     8  19
## 112    44     190 10.3   78     8  20
## 113    21     259 15.5   77     8  21
## 114     9      36 14.3   72     8  22
## 116    45     212  9.7   79     8  24
## 117   168     238  3.4   81     8  25
## 118    73     215  8.0   86     8  26
## 120    76     203  9.7   97     8  28
## 121   118     225  2.3   94     8  29
## 122    84     237  6.3   96     8  30
## 123    85     188  6.3   94     8  31
## 124    96     167  6.9   91     9   1
## 125    78     197  5.1   92     9   2
## 126    73     183  2.8   93     9   3
## 127    91     189  4.6   93     9   4
## 128    47      95  7.4   87     9   5
## 129    32      92 15.5   84     9   6
## 130    20     252 10.9   80     9   7
## 131    23     220 10.3   78     9   8
## 132    21     230 10.9   75     9   9
## 133    24     259  9.7   73     9  10
## 134    44     236 14.9   81     9  11
## 135    21     259 15.5   76     9  12
## 136    28     238  6.3   77     9  13
## 137     9      24 10.9   71     9  14
## 138    13     112 11.5   71     9  15
## 139    46     237  6.9   78     9  16
## 140    18     224 13.8   67     9  17
## 141    13      27 10.3   76     9  18
## 142    24     238 10.3   68     9  19
## 143    16     201  8.0   82     9  20
```

```
## 144    13     238 12.6   64     9  21
## 145    23      14  9.2   71     9  22
## 146    36     139 10.3   81     9  23
## 147     7      49 10.3   69     9  24
## 148    14      20 16.6   63     9  25
## 149    30     193  6.9   70     9  26
## 151    14     191 14.3   75     9  28
## 152    18     131  8.0   76     9  29
## 153    20     223 11.5   68     9  30
```

Data Tables (not Frames)

- Package, faster and more memory efficient
- Inherets from data.frame (all functions), written in C, faster at sub-setting, grouping, and updating
- http://stackoverflow.com/questions/13618488/what-you-can-do-with-data-frame-that-you-cant-in-data-table
- https://github.com/Rdatatable/data.table

```
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 4.4.2
```

```
DF = data.frame(x=rnorm(9),y=rep(c("a","b","c"), each=3),z=rnorm(9))
head(DF,3)
```

```
##             x y         z
## 1 -0.04759896 a 0.3247651
## 2  2.49587691 a 0.4809967
## 3 -0.51806561 a 0.9213347
```

```
DT = data.table(x=rnorm(9),y=rep(c("a","b","c"), each=3),z=rnorm(9))
head(DT,3)
```

```
##             x      y          z
##         <num> <char>      <num>
## 1: -1.0363187      a -0.4888463
## 2:  0.8224796      a  1.9095918
## 3: -0.3295392      a -1.8000500
```

```
tables() # get all data tables in memory
```

```
##    NAME NROW NCOL MB  COLS    KEY
## 1:   DT    9    3  0 x,y,z [NULL]
## Total: 0MB using type_size
```

```r
# subsetting
DT[2,] # subset rows
```

```
##            x      y        z
##        <num> <char>    <num>
## 1: 0.8224796      a 1.909592
```

```r
DT[DT$y=="a",] # subset where y is "a"
```

```
##             x      y          z
##         <num> <char>      <num>
## 1: -1.0363187      a -0.4888463
## 2:  0.8224796      a  1.9095918
## 3: -0.3295392      a -1.8000500
```

```r
DT[c(2,3)] # subset rows 2 & 3, one variable is assigned to rows
```

```
##             x      y         z
##         <num> <char>     <num>
## 1:  0.8224796      a  1.909592
## 2: -0.3295392      a -1.800050
```

```r
# subset cols, DT[,c(2.3)] does not work bc uses expressions
DT[,list(mean(x),sum(z))] # pass list of functions applied by names of columns
```

```
##            V1         V2
##         <num>      <num>
## 1: 0.01608497 -0.6637091
```

```r
DT[,table(y)] # get table of y values
```

```
## y
## a b c
## 3 3 3
```

```r
DT[, w := z^2] # adds columns quickly
DT2 <- DT # does not make a copy in memory, change one changes all, pointing to same memory. Use copy f
DT[,m:= {tmp <- (x+z); log2(tmp+5)}] # multiple step function, returns last statement in evaluation
DT[,a:=x>0] # expression evaluates boolean for new variable
DT[,b:= mean(x+w),by=a] # grouping by boolean a into factors to evaluate expression
# special variable .N integer len 1 num times group appears
set.seed(123)
DT <- data.table(x=sample(letters[1:3], 1E5, TRUE))
DT[, .N, by=x] # count number of times grouped by x variable
```

```
##         x     N
##    <char> <int>
## 1:      c 33294
## 2:      b 33305
## 3:      a 33401
```

```r
# data.table contains keys
DT <- data.table(x=rep(c("a","b","c"),each=100), y=rnorm(300))
setkey(DT, x)
DT["a"] # subset based on key x, faster
```

```
## Key: <x>
##           x           y
##      <char>       <num>
##   1:      a  0.88631257
##   2:      a  2.82858132
##   3:      a  2.03145429
##   4:      a  1.90675413
##   5:      a  0.21490826
##   6:      a -0.86273413
##   7:      a -2.20493863
##   8:      a  0.24105923
##   9:      a  1.83832419
##  10:      a  0.79205468
##  11:      a  0.65053469
##  12:      a -1.53912061
##  13:      a -0.60830053
##  14:      a  0.38195644
##  15:      a -1.07500044
##  16:      a  0.21994264
##  17:      a -0.78288781
##  18:      a -1.11003346
##  19:      a -1.65871456
##  20:      a -0.50147343
##  21:      a  1.91636375
##  22:      a  1.41236645
##  23:      a  0.92260986
##  24:      a  1.01106201
##  25:      a  0.57213026
##  26:      a -0.62843126
##  27:      a -0.36316140
##  28:      a -1.05858811
##  29:      a -0.42935803
##  30:      a  0.86941467
##  31:      a -0.54001647
##  32:      a -1.14647747
##  33:      a -0.17151840
##  34:      a -0.56368340
##  35:      a -0.42994346
##  36:      a -1.23723779
##  37:      a  0.15901329
##  38:      a -1.16711067
##  39:      a -0.08111944
##  40:      a -0.51667953
##  41:      a  0.99540703
##  42:      a  0.79752142
##  43:      a  0.53895224
##  44:      a -1.40405605
##  45:      a  0.40144065
```

```
##  46:      a -0.52432237
##  47:      a -0.83952146
##  48:      a  0.47556591
##  49:      a -0.01194696
##  50:      a  0.10319780
##  51:      a -0.38575415
##  52:      a  1.11726438
##  53:      a -0.49961390
##  54:      a -0.44735091
##  55:      a -0.23784512
##  56:      a -0.86939374
##  57:      a  1.14887678
##  58:      a  0.53864996
##  59:      a -0.10680992
##  60:      a  0.60053649
##  61:      a -1.47499445
##  62:      a  0.98126964
##  63:      a -0.61118738
##  64:      a  0.08938648
##  65:      a -0.01327227
##  66:      a -0.97219341
##  67:      a -0.57946225
##  68:      a  0.14963144
##  69:      a  0.47640689
##  70:      a  0.44729682
##  71:      a -0.19180956
##  72:      a  0.51712710
##  73:      a  0.40338273
##  74:      a  1.78411385
##  75:      a  0.27775645
##  76:      a  0.77394978
##  77:      a -2.08081928
##  78:      a -0.35920889
##  79:      a -0.45932217
##  80:      a  0.20181947
##  81:      a  0.62401138
##  82:      a -0.25722981
##  83:      a  0.94414021
##  84:      a  0.25074808
##  85:      a -0.72784257
##  86:      a  0.36881323
##  87:      a  0.44415068
##  88:      a -1.00535422
##  89:      a -0.33152471
##  90:      a -0.37039325
##  91:      a -0.79701529
##  92:      a  0.28148559
##  93:      a  0.33307250
##  94:      a  0.52690325
##  95:      a -0.78168949
##  96:      a -0.02793948
##  97:      a -1.74492339
##  98:      a  0.65284209
##  99:      a -0.93830821
```

```
## 100:      a  0.62753159
##           x           y
```

```
DT1 <- data.table(x=c("a","a","b","dt1"), y=1:4)
DT2 <- data.table(x=c("a","b","dt2"), z=5:7)
setkey(DT1,x); setkey(DT2,x)
merge(DT1,DT2) # uses keys to merge
```

```
## Key: <x>
##         x     y     z
##    <char> <int> <int>
## 1:      a     1     5
## 2:      a     2     5
## 3:      b     3     6
```

```
# fast reading in data.table
big_df <- data.frame(x=rnorm(1E6),y=rnorm(1E6))
file <- tempfile()
write.table(big_df, file=file, row.names=FALSE, col.names=TRUE, sep="\t", quote=FALSE)
system.time(fread(file)) # basically read.table for csv
```

```
##    user  system elapsed
##    0.03    0.01    0.06
```

```
system.time(read.table(file,header=TRUE,sep="\t"))
```

```
##    user  system elapsed
##    3.46    0.11    3.95
```

```
rm(list=ls())
```

Random Numbers

```
# Random number generation
# Probability distribution functions have 4 functions associated: d- density, r- random number generati
set.seed(1) # set sequence of random number generation. set.seed(1); rnorm(5) always results in the sam
y <- rnorm(1000) # generate vector of 1000 numbers that are standard normal distribution. Agrs: n, mean
y <- dnorm(c(0.25,0.5,0.75)) # evaluate Normal probability density, (given mean,sd) at point or vector
y <- pnorm(0.5) # evaluate cumulative distribution function for normal distribution. Args: q, mean=0, s
y <- qnorm(0.5) # evaluates quantiles for normal distribution. Args: p, mean=0, sd=1, lower.tail=TRUE,
y <- sample(1:6,3) # random selection of 3 elements from array
ints <- sample(10) # random sample all integers from 1 to 10 without replacement. Permutation
nums <- sample(1:10, replace = TRUE) # with replacement
let <- sample(LETTERS) # sample all letters without replacement
flips <- sample(c(0,1), 100, replace = TRUE, prob = c(0.3,0.7)) # unfair coin
coin <- rbinom(1,1,0.5) # simulating coin flip
unfairflip <- rbinom(1, size = 100, prob = 0.7) # sum of flips above
flips2 <- rbinom(100,1,0.7) # flips above
```

```r
y <- rpois(10, 1) # generate random poisson variates with given rate. Args: n (count), rate (mean)
pois_mat <- replicate(100, rpois(5, 10))

# Simulate Linear Model Ex
# y = B(o) + B(1) * x + e
# e ~ N(0,2^2) assume x ~ N(0,1^2), B(0) = 0.5, B(1) = 2.
set.seed(20)
x <- rnorm(100)
e <- rnorm(100,0,2)
y <- 0.5 + 2 * x + e
# can combine different distributions
# Poisson: Y ~ Poisson(mu)
# log(mu) = B(0) + B(1)x
# B(0) = 0.5 and B(1) = 0.3
set.seed(1)
x <- rnorm(100)
log.mu <- 0.5 + 0.3 * x
y <- rpois(100, exp(log.mu))

rm(list=ls())
```

Control Functions and Loop Functions

Control Functions

```r
# control execution of program

x = 2
# if,else loops
y <- if(x > 3){ # testing condition
  10
} else if(x > 0 & x <= 3) { # can not have or multiple
  5
} else{ # can not have, at end
  0
}

if(x-5 == 0){
  y <- 0
} else{
  y <- 2
}

# for loops
for(i in 1:10) {# execute loop fixed number of times. Args iterator variable and vector(inc seq) or lis
  print(i)
}
```

```
## [1] 1
## [1] 2
```

```
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

```
x <- c("a","b","c","d")
for(i in 1:4){
  print(x[i])
}
```

```
## [1] "a"
## [1] "b"
## [1] "c"
## [1] "d"
```

```
for(i in seq_along(x)){
  print(x[i])
}
```

```
## [1] "a"
## [1] "b"
## [1] "c"
## [1] "d"
```

```
for(letter in x){
  print(letter)
}
```

```
## [1] "a"
## [1] "b"
## [1] "c"
## [1] "d"
```

```
for(i in 1:4) print(x[i])
```

```
## [1] "a"
## [1] "b"
## [1] "c"
## [1] "d"
```

```
x <- matrix(1:6,2,3)
for(i in seq_len(nrow(x))) { # nested, don't use more than 2-3 for readability
  for(j in seq_len(ncol(x))) {
    print(x[i,j])
  }
}
```

```
## [1] 1
## [1] 3
## [1] 5
## [1] 2
## [1] 4
## [1] 6
```

```
# while loops
count <- 0
while(count < 10){ # loop while condition is true
  print(count)
  count <- count + 1
} # be wary of infinite loops!! when condition cannot be true
```

```
## [1] 0
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
```

```
z <- 5
while(z >= 3 & z <= 10){
  print(z)
  coin <- rbinom(1,1,0.5)

  if (coin == 1) z <- z+1
  else z <- z-1
}
```

```
## [1] 5
## [1] 6
## [1] 5
## [1] 4
## [1] 3
```

```
# Repeat loop
x0 <- 0.01; tol <- 1e-3
repeat { # infinite loop
  x1 <- rnorm(1)
  if(abs(x1 - x0) < tol) {
    break # break execution of any loop
  }
  else x0 <- x1
}

# control a loop
for(i in 1:100) {
```

```
  if(i <= 20) next # skip next iteration of loop
  else {
    if (i > 50) break # exit for loop
  }
}

# return to exit a function, will end control structure inside function
```

Loop Functions

```
# Loop functions - useful for looping in the command line
# Hadley Wickham's Journal of Statistical Software paper titled 'The Split-Apply-Combine Strategy for D

# lapply - loop over a list and evaluate on each element. args: X (list or coercion), FUN (function or
x <- list(a = 1:5, b = rnorm(10))
lapply(x, mean) # returns list of 2 numerics
```

```
## $a
## [1] 3
##
## $b
## [1] 0.3985388
```

```
x <- 1:4
lapply(x, runif, min = 0, max = 10) # passes subsequent args to function
```

```
## [[1]]
## [1] 4.180447
##
## [[2]]
## [1] 5.3804163 0.7510495
##
## [[3]]
## [1] 3.049216 2.719333 8.182229
##
## [[4]]
## [1] 0.8832537 3.4918707 8.5187127 9.8035107
```

```
x <- list(a = matrix(1:4, 2, 2), b = matrix(1:6, 3, 2))
lapply(x, function(elt) elt[,1]) # define an anonymous function inside lapply
```

```
## $a
## [1] 1 2
##
## $b
## [1] 1 2 3
```

```
# sapply - same as lapply but simplify, i.e. will make list of 1 element vectors a vector, multiple ele
x <- list(a = 1:5, b = rnorm(10))
lapply(x, mean) # now returns vector length 2
```

```
## $a
## [1] 3
##
## $b
## [1] 0.3902621
```

```
# mean only operates on signle element numeric/logical, so need to use loop

# vapply - pre-specify type of return value, safer and faster. Args: X, FUN, FUN.VALUE (generalized vec
vapply(x, mean, numeric(1)) # same as sapply(x, mean)
```

```
##         a         b
## 3.0000000 0.3902621
```

```
# apply - apply function over margins of array (good for summary of matrices or higher level array). No
x <- matrix(rnorm(200), 20, 10)
apply(x, 2, mean) # mean of each column by collapsing 1st dimension, returns num vector length of ncol.
```

```
##  [1]  0.39576926  0.39693829 -0.29548099 -0.30587580  0.31690617 -0.24744022
##  [7]  0.26027330  0.07700510 -0.04652335 -0.23800285
```

```
rowSums(x) # equivalent to apply(x, 1, sum)
```

```
##  [1]  0.7609383 -4.1967248  5.0584592  0.5808195 -3.3859346  8.2206313
##  [7]  1.3595547 -0.1567391  2.1183256  2.2432819  3.0644650  1.3968116
## [13]  5.4715183 -2.0890661 -0.7462932  0.6588537 -2.3037316 -3.9577417
## [19] -4.5847842 -3.2412655
```

```
rowMeans(x) # equivalent to apply(x, 1, mean)
```

```
##  [1]  0.07609383 -0.41967248  0.50584592  0.05808195 -0.33859346  0.82206313
##  [7]  0.13595547 -0.01567391  0.21183256  0.22432819  0.30644650  0.13968116
## [13]  0.54715183 -0.20890661 -0.07462932  0.06588537 -0.23037316 -0.39577417
## [19] -0.45847842 -0.32412655
```

```
colSums(x) # apply(x, 2, sum)
```

```
##  [1]  7.9153853  7.9387658 -5.9096198 -6.1175160  6.3381234 -4.9488043
##  [7]  5.2054659  1.5401019 -0.9304669 -4.7600570
```

```
colMeans(x) #apply(x, 2, mean)
```

```
##  [1]  0.39576926  0.39693829 -0.29548099 -0.30587580  0.31690617 -0.24744022
##  [7]  0.26027330  0.07700510 -0.04652335 -0.23800285
```

```
apply(x, 1, quantile, probs = c(0.25, 0.75)) # runs quantile with 2 agrs for every element in list, ret
```

```
##           [,1]       [,2]       [,3]       [,4]       [,5]      [,6]      [,7]
## 25% -0.7571352 -0.7008243 -0.2473744 -0.6418233 -0.5553241 0.4407991 0.1330451
## 75%  1.0018439 -0.1354330  1.4629405  0.7853807  0.1146551 1.2413891 1.0496289
##           [,8]       [,9]      [,10]      [,11]      [,12]      [,13]
## 25% -0.7121101 -0.2189623 -0.1192721 -0.1532343 -0.1754369 -0.2377421
## 75%  0.2584542  0.7632128  1.0476237  0.7373560  0.4873948  1.1822437
##          [,14]      [,15]      [,16]      [,17]      [,18]     [,19]      [,20]
## 25% -0.7463529 -0.3586825 -0.7882050 -0.7254670 -0.8275423 -0.799203 -0.8441518
## 75%  0.2985215  0.1503075  0.7323559  0.3109353  0.7182965  0.290152 -0.3795045
```

```
a <- array(rnorm(2 * 2 * 10), c(2, 2, 10)) # array in 3D
apply(a, c(1,2), mean) # collapses only 3rd dimension, returns 2x2 matrix. Equivalent rowMeans(a, dims
```

```
##            [,1]       [,2]
## [1,] -0.3294688  0.1786066
## [2,] -0.1456898 -0.2877835
```

```
# tapply - apply function over subset of a vector. args: X is vector, INDEX is factor/list factors vect
x <- c(rnorm(10), runif(10), rnorm(10,1))
f <- gl(3,10) # factor 3 levels, 10 times each
tapply(x,f,mean)
```

```
##         1         2         3
## 0.2233750 0.3445618 0.4956007
```

```
# mapply - multivariate version of lapply. args: FUN as above, ... (arguments to apply over), MoreArgs
list(rep(1,4), rep(2,3), rep(3,2), rep(4,1))
```

```
## [[1]]
## [1] 1 1 1 1
##
## [[2]]
## [1] 2 2 2
##
## [[3]]
## [1] 3 3
##
## [[4]]
## [1] 4
```

```
mapply(rep, 1:4, 4:1) # equivalent
```

```
## [[1]]
## [1] 1 1 1 1
##
## [[2]]
## [1] 2 2 2
##
## [[3]]
## [1] 3 3
##
## [[4]]
## [1] 4
```

```r
noise <- function(n,mean,sd){rnorm(n,mean,sd)}
noise(1:5,1:5,2) # gives vector of 5, same as single num args
```

```
## [1] 0.5569244 1.6755360 3.6736906 6.1633545 7.0603864
```

```r
mapply(noise,1:5,1:5,2) # applies function for each pair, list of 5 of length i
```

```
## [[1]]
## [1] 0.2931417
##
## [[2]]
## [1] 4.150763 1.569355
##
## [[3]]
## [1] 3.367118 4.901136 5.167102
##
## [[4]]
## [1] 1.010141 2.712134 6.857595 3.225181
##
## [[5]]
## [1] 4.858346 4.861672 4.799808 5.668715 2.350646
```

```r
# split - in conjunction with lapply to split objects into subpieces. Args: x (any object), f (factor),
x <- c(rnorm(10), runif(10), rnorm(10,1))
f <- gl(3,10) # factor 3 levels, 10 times each
split(x,f) # tapply without function, sorts into list based on levels, can then use lapply or sapply.
```

```
## $`1`
##  [1]  0.78002347 -0.78709697 -0.58691682 -0.54546587  0.76247880  0.06403316
##  [7]  0.12819144  0.60560030  0.39492984 -0.53621606
##
## $`2`
##  [1] 0.61128364 0.50431157 0.49886556 0.15303652 0.58167801 0.05305581
##  [7] 0.08354486 0.19449867 0.50655472 0.80669924
##
## $`3`
##  [1]  1.1065169  1.2236401  0.7009779  1.8481351  2.4935228  0.3468278
##  [7] -0.3368842  1.8210809  0.5114539  1.2572268
```

```r
lapply(split(x,f), mean) # in this case can use tapply
```

```
## $`1`
## [1] 0.02795613
##
## $`2`
## [1] 0.3993529
##
## $`3`
## [1] 1.09725
```

```r
# can do data frames
data <- read.csv("hw1_data.csv")
s <- split(data, data$Month)
sapply(s, function(x) colMeans(x[,c("Ozone","Solar.R","Wind")], na.rm = TRUE)) # data$Month coerced int
```

```
##                 5         6          7          8         9
## Ozone    23.61538  29.44444  59.115385  59.961538  31.44828
## Solar.R 181.29630 190.16667 216.483871 171.857143 167.43333
## Wind     11.62258  10.26667   8.941935   8.793548  10.18000
```

```r
# Multi-level split
x <- rnorm(10)
f1 <- gl(2,5); f2 <- gl(5,2) # ex. race and gender 2 factors
interaction(f1,f1) # combine each pair, 10 factors
```

```
##  [1] 1.1 1.1 1.1 1.1 1.1 2.2 2.2 2.2 2.2 2.2
## Levels: 1.1 2.1 1.2 2.2
```

```r
split(x, list(f1,f2)) # interaction called, list returned for combination sort, drop = TRUE to remove u
```

```
## $`1.1`
## [1]  0.1165892 -0.1194990
##
## $`2.1`
## numeric(0)
##
## $`1.2`
## [1]  0.4679266 -1.4368877
##
## $`2.2`
## numeric(0)
##
## $`1.3`
## [1] 0.5310122
##
## $`2.3`
## [1] -0.8627139
##
## $`1.4`
## numeric(0)
##
## $`2.4`
## [1] -1.2451944  0.6457308
##
## $`1.5`
## numeric(0)
##
## $`2.5`
## [1] -0.3394378 -0.2064004
```

```
rm(list=ls())
```

Defining Functions

```
# stored in txt or R script, functions are R objects. Can pass functions as arguments for other functio

myfunction <- function(){ #create a function
  x <- rnorm(100)
  mean(x)
}
myfunction() #call created function
```

```
## [1] -0.1028367
```

```
myfunction # prints source code for function
```

```
## function ()
## {
##     x <- rnorm(100)
##     mean(x)
## }
```

```
args(myfunction) # returns arguments for passed function
```

```
## function ()
## NULL
```

```
myaddedfunction <- function(x,y){ #create a function with formal arguments x and y
  x + y + rnorm(100) # implicit return last expression
}
myaddedfunction(5,3)
```

```
##   [1]  7.647769  6.334089  7.593286  6.268142  9.548806  9.191841  8.190586
##   [8]  8.226173  8.766742  9.634012  9.245233  5.921075  6.841281  7.993894
##  [15]  6.722897  9.420619  8.915033  7.623340  8.032766  8.883314  9.142204
##  [22]  8.000106  7.991077  7.685731  6.878269  7.864682  7.372188  8.462985
##  [29]  8.260722  6.964953  8.243108  8.238265  7.603194  7.843892  8.568631
##  [36]  9.067935  7.573488  9.495201  7.325929  6.319661  7.007791  6.507580
##  [43]  9.482838  9.262762  9.473943  7.560676  7.166530  7.353693  9.219610
##  [50]  8.611367  6.644443  8.048528  6.539457  8.539169  7.676676  8.584478
##  [57]  8.233269  7.799053  7.504200  7.847320  6.365965 10.249351  9.269278
##  [64]  6.715166  8.497680  8.015868  6.417507  7.669606  7.323856  8.684615
##  [71]  6.486430  8.711168  8.226389  7.602351  7.960909  5.829723  6.282247
##  [78]  7.605290  7.147546  7.433600  7.803719  7.836636  8.618066  8.022089
##  [85]  8.608157  8.589619  8.760178  9.216551  6.242489  8.209841  8.268497
##  [92]  5.898325  7.066211  7.331588  8.012126  7.676726  9.810349  7.364054
##  [99]  6.261003  7.979133
```

```
myaddedfunction(4:10,2)
```

```
## Warning in x + y + rnorm(100): longer object length is not a multiple of
## shorter object length
```

```
##   [1]  6.409714  6.341150  8.912148  8.792755 11.154946 12.277935 12.926515
##   [8]  5.939344  4.986605  9.444333  8.014862 10.204588 10.538864 13.736550
##  [15]  4.563040  8.080383  8.684333  8.388580 11.043038 10.504557 12.941534
##  [22]  6.405706  5.874937  8.759100  8.274475  9.002475 10.189102 12.813112
##  [29]  8.025275  9.588654  7.999254  7.269448 10.570705 13.922480 13.380562
##  [36]  6.301785  7.219056  8.360819  9.090101 10.532460 10.865300 11.339198
##  [43]  5.243282  6.518864  7.321305  8.387538 10.510045 12.148739 11.287710
##  [50]  5.292170  5.165806  7.699683  8.664981  9.362315 10.495106 14.330201
##  [57]  6.509956  6.178560  7.865012  9.993201  9.207881  9.523909 12.187720
##  [64]  5.027184  7.545402  9.329583  9.364413  9.260397 10.757635 11.208893
##  [71]  6.239572  6.448273  9.943164  8.470355  8.911138 12.665069 11.722099
##  [78]  4.015922  5.826076  7.836202 10.036603  9.776626 11.550650 11.751905
##  [85]  7.533689  5.243682  7.060479  8.798539 10.764209 10.285979  9.854701
##  [92]  5.259725  8.486536  7.838495  9.715276  8.769817 11.769759 11.175956
##  [99]  7.670140  7.005200
```

```
# function with default argument if left unspecified, for common cases
above <- function(x, n = 10){
  use <- x > n
  x[use]
}
above(1:20) # n is default set to 10
```

```
##  [1] 11 12 13 14 15 16 17 18 19 20
```

```
above(1:20, 12) # n set at 12
```

```
## [1] 13 14 15 16 17 18 19 20
```

```
columnmean <- function(y, removeNA = TRUE) {
  nc <- ncol(y)
  means <- numeric(nc)
  for(i in 1:nc) means[i] <- mean(y[,i], na.rm = removeNA)
  invisible(means) # auto-return blocks auto-print
}

# Lazy Evaluation: R evaluated statements and arguments as they come
f <- function (a,b,c){
  print(a)
  #print(b) # error
}
f(3) # prints a, error for b, no rxn to not having c
```

```
## [1] 3
```

```r
# ways to call functions
# positional matching and naming can be mixed. Partial matching also allowed, if not found uses positio
# named helps for long arg list where most defaults are maintained or if order is hard to remember.
mydata <- rnorm(100)
sd(mydata) # default to first argument
```

```
## [1] 1.001767
```

```r
sd(x = mydata)
```

```
## [1] 1.001767
```

```r
sd(x = mydata, na.rm = FALSE)
```

```
## [1] 1.001767
```

```r
sd(na.rm = FALSE, x = mydata)
```

```
## [1] 1.001767
```

```r
sd(na.rm = FALSE, mydata) # remove argument from list, default works on first unspecified arg
```

```
## [1] 1.001767
```

```r
# Variable Arguments
# to extend another function without copying arg list of OG function
simon_says <- function(...){
  paste("Simon says:", ...)
}
# or for generic functions passed to methods
# unpacking an ellipses
mad_libs <- function(...){
  args <- list(...)
  place <- args$place
  adjective <- args$adjective
  noun <- args$noun
  paste("News from", place, "today where", adjective, "students took to the streets in protest of the n
}
# or when number of args unknown in advance (if at beginning, no positional or partial matching)
args(paste) # operates on unknown sets of character vectors
```

```
## function (..., sep = " ", collapse = NULL, recycle0 = FALSE)
## NULL
```

```r
# function as an argument
some_function <- function(func){
  func(2, 4) # returns result of function with 2,4 arguments
}
some_function(mean) # returns mean of 2,4
```

```
## [1] 2
```

```
# Anonymous function (chaos)
evaluate  <- function(func, dat){
  func(dat)
}
evaluate(function(x){x+1}, 6) # creates a function when calling evaluate to add 1
```

```
## [1] 7
```

```
# create a binary operation
"%mult_add_one%" <- function(left, right){
  left * right + 1
}
4 %mult_add_one% 5
```

```
## [1] 21
```

Lexical Scoping

```
make.power <- function(n) {
  pow <- function(x) {
    x^n
  }
  pow
}

cube <- make.power(3)
square <- make.power(2)
cube(3)
```

```
## [1] 27
```

```
square(3)
```

```
## [1] 9
```

```
# Scoping - environments
search()# provides list of environments
```

```
##  [1] ".GlobalEnv"         "package:data.table" "package:stats"
##  [4] "package:graphics"   "package:grDevices"  "package:utils"
##  [7] "package:datasets"   "package:methods"    "Autoloads"
## [10] "package:base"
```

```
ls(environment(cube)) # object names in function environment, same for square
```

```
## [1] "n"   "pow"
```

```
get("n",environment(cube)) # values in function environment, changes for square
```

```
## [1] 3
```

```
rm(list=ls())
```

R Packages

- Repositories: CRAN, BioConductor (bioinformatics), GitHub
- Search: https://www.rdocumentation.org/
- Base packages: utils, stats, datasets, graphics, grDevices, grid, methods, tools, parallel, compiler, splines, tcltk, stats4.
- Recommended packages: boot, class, cluster, codetools, foreign, KernSmooth, lattice, mgcv, nime, rpart, survival, MASS, spatial, nnet, Matrix.

```
# Install from CRAN:
#   install.packages("ggplot2", repos = "http://cran.us.r-project.org") #install
#   install.packages(c("labeling","tibble"), repos = "http://cran.us.r-project.org") #multiple

# Install from Bioconductor
#   install.packages("BiocManager", repos = "https://bioconductor.org/biocLite.R")
#   BiocManager::install(c("GenomicFeatures", "AnnotationDbi")) #install package

# Install from GitHub (need package, author name)
#   install.packages("devtools", repos = "http://cran.us.r-project.org") #only once
#   library(devtools)
#   install_github("author/package") #installs package

# library(ggplot2)# Load package, careful of dependencies
# installed.packages() #check installed packages
# library() #alternate
# old.packages(repos = "http://cran.us.r-project.org") #check packages to update
# update.packages(repos = "http://cran.us.r-project.org") #update all packages
# install.packages("ggplot2") #to update single package
# detach("package:ggplot2", unload=TRUE) #unload function
# remove.packages("ggtree") #remove package
# help(package = "ggplot2") #package info
# browseVignettes("ggplot2") #extended help files
```

Getting the Data

Defining Workspace or directory

```r
x <- getwd() # find working directory
x
```

```
## [1] "C:/Users/Owner/OneDrive/Documents/CompSci/CourseraDS/CourseraDataScience"
```

```r
dir.create("testdir") # create a directory if doesn't exist, args: dir name, for nested recursive = tru
```

```
## Warning in dir.create("testdir"): 'testdir' already exists
```

```r
setwd("testdir") # set working dir
file.create("mytest.R") # create file in wd
```

```
## [1] TRUE
```

```r
file.exists("mytest.R") # check if file or directory exists in wd
```

```
## [1] TRUE
```

```r
file.info("mytest.R") # file metadata, use $ operator to grab specific items
```

```
##          size isdir mode               mtime               ctime
## mytest.R    0 FALSE  666 2025-01-22 00:35:50 2025-01-22 00:35:50
##                        atime exe
## mytest.R 2025-01-22 00:35:50  no
```

```r
file.rename("mytest.R","mytest2.R") # rename
```

```
## [1] TRUE
```

```r
file.copy("mytest2.R","mytest3.R") # copy file
```

```
## [1] FALSE
```

```r
file.remove("mytest2.R") # remove file
```

```
## [1] TRUE
```

```r
file.path("mytest3.R") # relative path
```

```
## [1] "mytest3.R"
```

```r
setwd(x) # could use relative setwd("../") to move up one, setwd("./data"), or absolute path directly u

dir() # output files in directory. Also list.files()
```

```
##  [1] "API_auth_ex.R"              "Code-Library.pdf"
##  [3] "Code-Library.Rmd"           "Code Library.Rmd"
##  [5] "coded.R"                    "complete.R"
##  [7] "corr.R"                     "Course-Notes.pdf"
##  [9] "Course Notes.Rmd"           "CourseraDataScience.Rproj"
## [11] "example.h5"                 "getdata_data_ss06pid.csv"
## [13] "getdata_wksst8110.for"      "hw1_data.csv"
## [15] "pollutantmean.R"            "Programming 2.3"
## [17] "quiz_data_3.1.2.xlsx"       "quiz_data_3.1.csv"
## [19] "Rprof.out"                  "rprog_data_ProgAssignment3"
## [21] "specdata"                   "specdata.zip"
## [23] "testdir"
```

```
files_full <- list.files("specdata", full.names=TRUE) # pull all file names from a directory
ls() # prints the objects in work space
```

```
## [1] "files_full" "x"
```

```
rm(list=ls()) # clear workspace
rm(list=setdiff(ls(), "x")) # clear workspace except x
version #R info version
```

```
##                _
## platform       x86_64-w64-mingw32
## arch           x86_64
## os             mingw32
## crt            ucrt
## system         x86_64, mingw32
## status
## major          4
## minor          4.1
## year           2024
## month          06
## day            14
## svn rev        86737
## language       R
## version.string R version 4.4.1 (2024-06-14 ucrt)
## nickname       Race for Your Life
```

```
sessionInfo() #R info version, packages
```

```
## R version 4.4.1 (2024-06-14 ucrt)
## Platform: x86_64-w64-mingw32/x64
## Running under: Windows 11 x64 (build 22631)
##
## Matrix products: default
##
##
## locale:
## [1] LC_COLLATE=English_United States.utf8
## [2] LC_CTYPE=English_United States.utf8
## [3] LC_MONETARY=English_United States.utf8
```

```
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.utf8
##
## time zone: America/New_York
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] data.table_1.16.4
##
## loaded via a namespace (and not attached):
##  [1] compiler_4.4.1    fastmap_1.2.0     cli_3.6.3         tools_4.4.1
##  [5] htmltools_0.5.8.1 rstudioapi_0.17.1 yaml_2.3.10       rmarkdown_2.29
##  [9] knitr_1.49        xfun_0.49         digest_0.6.37     rlang_1.1.4
## [13] evaluate_1.0.3
```

```
source("coded.R") # load code into console
```

```
## [1] "Hello World"
```

```
args(ls()) # get arguments for a function
```

```
## NULL
```

```
help(ls) # access documentation on ls() function
```

```
## starting httpd help server ... done
```

```
?ls # same. for operator use ?`:`

# Interface to outside world
file(description = "hw1_data.csv") # open connection to standard, uncompressed file. Helps for partial
```

```
## A connection with
## description "hw1_data.csv"
## class       "file"
## mode        "r"
## text        "text"
## opened      "closed"
## can read    "yes"
## can write   "yes"
```

```
# gzfile() # connection to file w compression gzip
# bzfile() # connection to file w compression bzip2
jh <- url("http://www.jhsph.edu", "r") # connection to webpage
close(jh) # to end connection
```

Reading and Writing Data

```r
# Download data from website
fileUrl <- "https://stg-arcgisazurecdataprod1.az.arcgis.com/exportfiles-1506-24014/Part1_Crime_Beta_596
if(!file.exists("./testdir/crimedata.csv")) download.file(fileUrl, destfile = "./testdir/crimedata.csv"
dateDownloaded <- date() # Keep track of date downloaded for file

# Read table or csv data into R
x <- read.table("hw1_data.csv", header = TRUE, sep = ",") # reading tabular data from text files, retur
head(x)
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 5    NA      NA 14.3   56     5   5
## 6    28      NA 14.9   66     5   6
```

```r
x <- read.csv("hw1_data.csv") # Same but default separator is ", "  and header = TRUE
# write.table(x)

# help read.table with colClasses with smaller sample
initial <- read.table("hw1_data.csv", header = TRUE, sep = ",", nrows = 100)
classes <- sapply(initial, class)
tabAll <- read.table("hw1_data.csv", header = TRUE, sep = ",", colClasses = classes)

# Read Excel file into R
x <- "https://stg-arcgisazurecdataprod1.az.arcgis.com/exportfiles-1506-24014/Part1_Crime_Beta_383480701
if(!file.exists("./testdir/crimexl.xlsx")) {
  download.file(x, destfile = "./testdir/crimexl.xlsx", method = "curl") # improve
  dateXl <- date()
}
library(readxl) # xlsx, XLConnect other options
```

```
## Warning: package 'readxl' was built under R version 4.4.2
```

```r
cameraData <- read_excel("./testdir/crimexl.xlsx", sheet = 1, col_types = "text", n_max = 100)
head(cameraData)
```

```
## # A tibble: 6 x 23
##   RowID CCNumber CrimeDateTime CrimeCode Description Inside_Outside Weapon Post
##   <chr> <chr>    <chr>         <chr>     <chr>       <chr>          <chr>  <chr>
## 1 1     14I10336 41880.5       5A        BURGLARY    <NA>           NA     835
## 2 2     14H13075 41880.989583~ 4E        COMMON ASS~ <NA>           NA     515
## 3 3     14H13066 41880.965277~ 6G        LARCENY     <NA>           NA     114
## 4 4     14I00593 41880.291666~ 6J        LARCENY     <NA>           NA     732
## 5 5     14H13360 41880.979166~ 6G        LARCENY     <NA>           NA     213
## 6 6     14H12804 41880.475694~ 4E        COMMON ASS~ <NA>           NA     412
## # i 15 more variables: Gender <chr>, Age <chr>, Race <chr>, Ethnicity <chr>,
## #   Location <chr>, Old_District <chr>, New_District <chr>, Neighborhood <chr>,
## #   Latitude <chr>, Longitude <chr>, GeoLocation <chr>, PremiseType <chr>,
## #   Total_Incidents <chr>, x <chr>, y <chr>
```

```r
# write also works

lines <- readLines("coded.R") # reading lines of text file, return character vector
```

```
## Warning in readLines("coded.R"): incomplete final line found on 'coded.R'
```

```r
writeLines("coded.R")
```

```
## coded.R
```

```r
# editable textual format retains metadata, helpful for version control, corruption fixable, memory cos
dget("coded.R") # reading R objects deparsed into text files
```

```
## [1] "Hello World"
```

```r
dput("coded.R") # takes R object, create R code to reconstruct object saving attributes, names
```

```
## "coded.R"
```

```r
source("coded.R") # reading in R code files
```

```
## [1] "Hello World"
```

```r
#dump() # multiple R objects

# load() # read in saved workspace read binary objects into R
# save()
# unserialize() # read single R objects in binary form
# serialize()
```

Just XML stuff

- XML (extensible markup language)
- http://en.wikipedia.org/wiki/XML
- store structured data in internet applications. From web scraping, API
- Components: markup (labels for structure) and content (text)
- Tags (labels): start
  end
  empty
- Elements: example of tag Hello, world

- Attributes: component of label
- XPath language: http://www.stat.berkeley.edu/~statcur/Workshop2/Presentations/XML.pdf
- /node (top level node), //node (node any level), node[@attr-name] node w attr name, node[@attr-name='bob']
- XML Short Intro: https://www.omegahat.net/RSXML/shortIntro.pdf
- XML Long Intro: https://www.omegahat.net/RSXML/Tour.pdf

```
# Reading file
library(XML)
```

```
## Warning: package 'XML' was built under R version 4.4.2
```

```
library(httr)
```

```
## Warning: package 'httr' was built under R version 4.4.2
```

```
fileUrl <- "https://www.w3schools.com/xml/simple.xml"
doc <- xmlTreeParse(rawToChar(GET(fileUrl)$content),useInternalNodes = TRUE) # read file into R to pars
rootNode <- xmlRoot(doc) # wrapper for structured element

# Access Data
xmlName(rootNode) # name of xml
```

```
## [1] "breakfast_menu"
```

```
names(rootNode) # names of root node
```

```
##   food   food   food   food   food
## "food" "food" "food" "food" "food"
```

```
rootNode[[1]] # first food element, get first subelement of element using [[1]][[1]]
```

```
## <food>
##   <name>Belgian Waffles</name>
##   <price>$5.95</price>
##   <description>Two of our famous Belgian Waffles with plenty of real maple syrup</description>
##   <calories>650</calories>
## </food>
```

```
xmlSApply(rootNode,xmlValue) # give xml value of each root node
```

```
##
##                                "Belgian Waffles$5.95Two of our famous Belgian Waffles with plenty of
##
##                    "Strawberry Belgian Waffles$7.95Light Belgian waffles covered with strawberries a
```

```
##
## "Berry-Berry Belgian Waffles$8.95Light Belgian waffles covered with an assortment of fresh berries ar
##
##                                                  "French Toast$4.50Thick slices made from our homemade
##
##                          "Homestyle Breakfast$6.95Two eggs, bacon or sausage, toast, and our ever-popu
```

```
xpathSApply(rootNode,"//name",xmlValue) # get node xmlValue for each names
```

```
## [1] "Belgian Waffles"             "Strawberry Belgian Waffles"
## [3] "Berry-Berry Belgian Waffles" "French Toast"
## [5] "Homestyle Breakfast"
```

```
xpathSApply(rootNode,"//price",xmlValue)
```

```
## [1] "$5.95" "$7.95" "$8.95" "$4.50" "$6.95"
```

```
# Pokemon
library(rvest)
```

```
## Warning: package 'rvest' was built under R version 4.4.2
```

```
fileUrlBase <- "https://pokemondb.net/pokedex/stats/gen"
generations <- 1:8
# Scrape Data
pokemonList <- lapply(generations,function(x){
  html <- read_html(paste0(fileUrlBase,x)) # create URL
  tableNode <- html_node(html,xpath="//*[(@id = 'pokedex')]") # read HTML, extract node using xpath sel
  data <- html_table(html,header=TRUE)[[1]] # convert to data frame
  colnames(data)[1] <- "ID"
  data$Generation <- x # add generation ID
  data # return data frame to list
})
thePokemon <- do.call(rbind,pokemonList) # one data frame
# Cleaning
types <- strsplit(gsub('[a-z]\\K(?=[A-Z])', ' ',thePokemon$Type, perl=T)," ") # split Type column
thePokemon$Type1 <- unlist(lapply(types,function(x) x[1])) # Type1 column
thePokemon$Type2 <- unlist(lapply(types,function(x) ifelse(is.na(x[2])," ",x[2]))) # Type2 column
pokemonNames <- strsplit(gsub('[a-z]\\K(?=[A-Z])', '--', thePokemon$Name, perl=T),"--") # split Name
thePokemon$Name <- unlist(lapply(pokemonNames,function(x) x[1]))
thePokemon$Form <- unlist(lapply(pokemonNames,function(x) ifelse(is.na(x[2])," ",x[2])))
thePokemon <- thePokemon[,c(1,2,14,12,13,4,5,6,7,8,9,10,11)] # reorder and remove unneccesary
thePokemon[29,"Name"] <- "Nidoran" # cleaning gender
thePokemon[29,"Form"] <- "Female"
thePokemon[32,"Name"] <- "Nidoran"
thePokemon[32,"Form"] <- "Male"
```

Just JSON Stuff

- Javascript Object Notation- light weight data storage, common format from APIs, similar to XML but different syntax/format.

- Data stored as: number(double), strings(double quoted), boolean, array(ordered, comma sep, [] enclosed), objects(unordered, comma sep, {} enclosed, key:value pairs).
- http://en.wikipedia.org/wiki/JSON
- http://www.json.org/
- jsonlite vignette
- http://www.r-bloggers.com/new-package-jsonlite-a-smarter-json-encoderdecoder/

```
library(jsonlite)
```

```
## Warning: package 'jsonlite' was built under R version 4.4.2
```

```
jsonData <- fromJSON("https://api.github.com/users/jtleek/repos") # can pass any format
names(jsonData) # names of data frame
```

```
##  [1] "id"                          "node_id"
##  [3] "name"                        "full_name"
##  [5] "private"                     "owner"
##  [7] "html_url"                    "description"
##  [9] "fork"                        "url"
## [11] "forks_url"                   "keys_url"
## [13] "collaborators_url"           "teams_url"
## [15] "hooks_url"                   "issue_events_url"
## [17] "events_url"                  "assignees_url"
## [19] "branches_url"                "tags_url"
## [21] "blobs_url"                   "git_tags_url"
## [23] "git_refs_url"                "trees_url"
## [25] "statuses_url"                "languages_url"
## [27] "stargazers_url"              "contributors_url"
## [29] "subscribers_url"             "subscription_url"
## [31] "commits_url"                 "git_commits_url"
## [33] "comments_url"                "issue_comment_url"
## [35] "contents_url"                "compare_url"
## [37] "merges_url"                  "archive_url"
## [39] "downloads_url"               "issues_url"
## [41] "pulls_url"                   "milestones_url"
## [43] "notifications_url"           "labels_url"
## [45] "releases_url"                "deployments_url"
## [47] "created_at"                  "updated_at"
## [49] "pushed_at"                   "git_url"
## [51] "ssh_url"                     "clone_url"
## [53] "svn_url"                     "homepage"
## [55] "size"                        "stargazers_count"
## [57] "watchers_count"              "language"
## [59] "has_issues"                  "has_projects"
## [61] "has_downloads"               "has_wiki"
## [63] "has_pages"                   "has_discussions"
## [65] "forks_count"                 "mirror_url"
## [67] "archived"                    "disabled"
## [69] "open_issues_count"           "license"
```

```
## [71] "allow_forking"               "is_template"
## [73] "web_commit_signoff_required" "topics"
## [75] "visibility"                  "forks"
## [77] "open_issues"                 "watchers"
## [79] "default_branch"
```

```
names(jsonData$owner) # names of column
```

```
##  [1] "login"               "id"                  "node_id"
##  [4] "avatar_url"          "gravatar_id"         "url"
##  [7] "html_url"            "followers_url"       "following_url"
## [10] "gists_url"           "starred_url"         "subscriptions_url"
## [13] "organizations_url"   "repos_url"           "events_url"
## [16] "received_events_url" "type"                "user_view_type"
## [19] "site_admin"
```

```
jsonData$owner$login # who logged in
```

```
##  [1] "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek"
##  [9] "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek"
## [17] "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek"
## [25] "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek"
```

```
# write data frame to JSON
myjson <- toJSON(iris, pretty = TRUE)
cat(myjson)
```

```
## [
##   {
##     "Sepal.Length": 5.1,
##     "Sepal.Width": 3.5,
##     "Petal.Length": 1.4,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 4.9,
##     "Sepal.Width": 3,
##     "Petal.Length": 1.4,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 4.7,
##     "Sepal.Width": 3.2,
##     "Petal.Length": 1.3,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 4.6,
##     "Sepal.Width": 3.1,
```

```
##     "Petal.Length": 1.5,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5,
##     "Sepal.Width": 3.6,
##     "Petal.Length": 1.4,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5.4,
##     "Sepal.Width": 3.9,
##     "Petal.Length": 1.7,
##     "Petal.Width": 0.4,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 4.6,
##     "Sepal.Width": 3.4,
##     "Petal.Length": 1.4,
##     "Petal.Width": 0.3,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5,
##     "Sepal.Width": 3.4,
##     "Petal.Length": 1.5,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 4.4,
##     "Sepal.Width": 2.9,
##     "Petal.Length": 1.4,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 4.9,
##     "Sepal.Width": 3.1,
##     "Petal.Length": 1.5,
##     "Petal.Width": 0.1,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5.4,
##     "Sepal.Width": 3.7,
##     "Petal.Length": 1.5,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
```

```
##     "Sepal.Length": 4.8,
##     "Sepal.Width": 3.4,
##     "Petal.Length": 1.6,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 4.8,
##     "Sepal.Width": 3,
##     "Petal.Length": 1.4,
##     "Petal.Width": 0.1,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 4.3,
##     "Sepal.Width": 3,
##     "Petal.Length": 1.1,
##     "Petal.Width": 0.1,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5.8,
##     "Sepal.Width": 4,
##     "Petal.Length": 1.2,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5.7,
##     "Sepal.Width": 4.4,
##     "Petal.Length": 1.5,
##     "Petal.Width": 0.4,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5.4,
##     "Sepal.Width": 3.9,
##     "Petal.Length": 1.3,
##     "Petal.Width": 0.4,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5.1,
##     "Sepal.Width": 3.5,
##     "Petal.Length": 1.4,
##     "Petal.Width": 0.3,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5.7,
##     "Sepal.Width": 3.8,
##     "Petal.Length": 1.7,
##     "Petal.Width": 0.3,
##     "Species": "setosa"
```

```
##   },
##   {
##     "Sepal.Length": 5.1,
##     "Sepal.Width": 3.8,
##     "Petal.Length": 1.5,
##     "Petal.Width": 0.3,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5.4,
##     "Sepal.Width": 3.4,
##     "Petal.Length": 1.7,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5.1,
##     "Sepal.Width": 3.7,
##     "Petal.Length": 1.5,
##     "Petal.Width": 0.4,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 4.6,
##     "Sepal.Width": 3.6,
##     "Petal.Length": 1,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5.1,
##     "Sepal.Width": 3.3,
##     "Petal.Length": 1.7,
##     "Petal.Width": 0.5,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 4.8,
##     "Sepal.Width": 3.4,
##     "Petal.Length": 1.9,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5,
##     "Sepal.Width": 3,
##     "Petal.Length": 1.6,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5,
##     "Sepal.Width": 3.4,
##     "Petal.Length": 1.6,
```

```
##     "Petal.Width": 0.4,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5.2,
##     "Sepal.Width": 3.5,
##     "Petal.Length": 1.5,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5.2,
##     "Sepal.Width": 3.4,
##     "Petal.Length": 1.4,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 4.7,
##     "Sepal.Width": 3.2,
##     "Petal.Length": 1.6,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 4.8,
##     "Sepal.Width": 3.1,
##     "Petal.Length": 1.6,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5.4,
##     "Sepal.Width": 3.4,
##     "Petal.Length": 1.5,
##     "Petal.Width": 0.4,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5.2,
##     "Sepal.Width": 4.1,
##     "Petal.Length": 1.5,
##     "Petal.Width": 0.1,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5.5,
##     "Sepal.Width": 4.2,
##     "Petal.Length": 1.4,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 4.9,
```

```
##     "Sepal.Width": 3.1,
##     "Petal.Length": 1.5,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5,
##     "Sepal.Width": 3.2,
##     "Petal.Length": 1.2,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5.5,
##     "Sepal.Width": 3.5,
##     "Petal.Length": 1.3,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 4.9,
##     "Sepal.Width": 3.6,
##     "Petal.Length": 1.4,
##     "Petal.Width": 0.1,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 4.4,
##     "Sepal.Width": 3,
##     "Petal.Length": 1.3,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5.1,
##     "Sepal.Width": 3.4,
##     "Petal.Length": 1.5,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5,
##     "Sepal.Width": 3.5,
##     "Petal.Length": 1.3,
##     "Petal.Width": 0.3,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 4.5,
##     "Sepal.Width": 2.3,
##     "Petal.Length": 1.3,
##     "Petal.Width": 0.3,
##     "Species": "setosa"
##   },
```

```
##   {
##     "Sepal.Length": 4.4,
##     "Sepal.Width": 3.2,
##     "Petal.Length": 1.3,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5,
##     "Sepal.Width": 3.5,
##     "Petal.Length": 1.6,
##     "Petal.Width": 0.6,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5.1,
##     "Sepal.Width": 3.8,
##     "Petal.Length": 1.9,
##     "Petal.Width": 0.4,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 4.8,
##     "Sepal.Width": 3,
##     "Petal.Length": 1.4,
##     "Petal.Width": 0.3,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5.1,
##     "Sepal.Width": 3.8,
##     "Petal.Length": 1.6,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 4.6,
##     "Sepal.Width": 3.2,
##     "Petal.Length": 1.4,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5.3,
##     "Sepal.Width": 3.7,
##     "Petal.Length": 1.5,
##     "Petal.Width": 0.2,
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 5,
##     "Sepal.Width": 3.3,
##     "Petal.Length": 1.4,
##     "Petal.Width": 0.2,
```

```
##     "Species": "setosa"
##   },
##   {
##     "Sepal.Length": 7,
##     "Sepal.Width": 3.2,
##     "Petal.Length": 4.7,
##     "Petal.Width": 1.4,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 6.4,
##     "Sepal.Width": 3.2,
##     "Petal.Length": 4.5,
##     "Petal.Width": 1.5,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 6.9,
##     "Sepal.Width": 3.1,
##     "Petal.Length": 4.9,
##     "Petal.Width": 1.5,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 5.5,
##     "Sepal.Width": 2.3,
##     "Petal.Length": 4,
##     "Petal.Width": 1.3,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 6.5,
##     "Sepal.Width": 2.8,
##     "Petal.Length": 4.6,
##     "Petal.Width": 1.5,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 5.7,
##     "Sepal.Width": 2.8,
##     "Petal.Length": 4.5,
##     "Petal.Width": 1.3,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 6.3,
##     "Sepal.Width": 3.3,
##     "Petal.Length": 4.7,
##     "Petal.Width": 1.6,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 4.9,
##     "Sepal.Width": 2.4,
```

```
##     "Petal.Length": 3.3,
##     "Petal.Width": 1,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 6.6,
##     "Sepal.Width": 2.9,
##     "Petal.Length": 4.6,
##     "Petal.Width": 1.3,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 5.2,
##     "Sepal.Width": 2.7,
##     "Petal.Length": 3.9,
##     "Petal.Width": 1.4,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 5,
##     "Sepal.Width": 2,
##     "Petal.Length": 3.5,
##     "Petal.Width": 1,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 5.9,
##     "Sepal.Width": 3,
##     "Petal.Length": 4.2,
##     "Petal.Width": 1.5,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 6,
##     "Sepal.Width": 2.2,
##     "Petal.Length": 4,
##     "Petal.Width": 1,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 6.1,
##     "Sepal.Width": 2.9,
##     "Petal.Length": 4.7,
##     "Petal.Width": 1.4,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 5.6,
##     "Sepal.Width": 2.9,
##     "Petal.Length": 3.6,
##     "Petal.Width": 1.3,
##     "Species": "versicolor"
##   },
##   {
```

```
##     "Sepal.Length": 6.7,
##     "Sepal.Width": 3.1,
##     "Petal.Length": 4.4,
##     "Petal.Width": 1.4,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 5.6,
##     "Sepal.Width": 3,
##     "Petal.Length": 4.5,
##     "Petal.Width": 1.5,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 5.8,
##     "Sepal.Width": 2.7,
##     "Petal.Length": 4.1,
##     "Petal.Width": 1,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 6.2,
##     "Sepal.Width": 2.2,
##     "Petal.Length": 4.5,
##     "Petal.Width": 1.5,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 5.6,
##     "Sepal.Width": 2.5,
##     "Petal.Length": 3.9,
##     "Petal.Width": 1.1,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 5.9,
##     "Sepal.Width": 3.2,
##     "Petal.Length": 4.8,
##     "Petal.Width": 1.8,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 6.1,
##     "Sepal.Width": 2.8,
##     "Petal.Length": 4,
##     "Petal.Width": 1.3,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 6.3,
##     "Sepal.Width": 2.5,
##     "Petal.Length": 4.9,
##     "Petal.Width": 1.5,
##     "Species": "versicolor"
```

```
##   },
##   {
##     "Sepal.Length": 6.1,
##     "Sepal.Width": 2.8,
##     "Petal.Length": 4.7,
##     "Petal.Width": 1.2,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 6.4,
##     "Sepal.Width": 2.9,
##     "Petal.Length": 4.3,
##     "Petal.Width": 1.3,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 6.6,
##     "Sepal.Width": 3,
##     "Petal.Length": 4.4,
##     "Petal.Width": 1.4,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 6.8,
##     "Sepal.Width": 2.8,
##     "Petal.Length": 4.8,
##     "Petal.Width": 1.4,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 6.7,
##     "Sepal.Width": 3,
##     "Petal.Length": 5,
##     "Petal.Width": 1.7,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 6,
##     "Sepal.Width": 2.9,
##     "Petal.Length": 4.5,
##     "Petal.Width": 1.5,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 5.7,
##     "Sepal.Width": 2.6,
##     "Petal.Length": 3.5,
##     "Petal.Width": 1,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 5.5,
##     "Sepal.Width": 2.4,
##     "Petal.Length": 3.8,
```

```
##     "Petal.Width": 1.1,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 5.5,
##     "Sepal.Width": 2.4,
##     "Petal.Length": 3.7,
##     "Petal.Width": 1,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 5.8,
##     "Sepal.Width": 2.7,
##     "Petal.Length": 3.9,
##     "Petal.Width": 1.2,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 6,
##     "Sepal.Width": 2.7,
##     "Petal.Length": 5.1,
##     "Petal.Width": 1.6,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 5.4,
##     "Sepal.Width": 3,
##     "Petal.Length": 4.5,
##     "Petal.Width": 1.5,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 6,
##     "Sepal.Width": 3.4,
##     "Petal.Length": 4.5,
##     "Petal.Width": 1.6,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 6.7,
##     "Sepal.Width": 3.1,
##     "Petal.Length": 4.7,
##     "Petal.Width": 1.5,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 6.3,
##     "Sepal.Width": 2.3,
##     "Petal.Length": 4.4,
##     "Petal.Width": 1.3,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 5.6,
```

```
##     "Sepal.Width": 3,
##     "Petal.Length": 4.1,
##     "Petal.Width": 1.3,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 5.5,
##     "Sepal.Width": 2.5,
##     "Petal.Length": 4,
##     "Petal.Width": 1.3,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 5.5,
##     "Sepal.Width": 2.6,
##     "Petal.Length": 4.4,
##     "Petal.Width": 1.2,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 6.1,
##     "Sepal.Width": 3,
##     "Petal.Length": 4.6,
##     "Petal.Width": 1.4,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 5.8,
##     "Sepal.Width": 2.6,
##     "Petal.Length": 4,
##     "Petal.Width": 1.2,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 5,
##     "Sepal.Width": 2.3,
##     "Petal.Length": 3.3,
##     "Petal.Width": 1,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 5.6,
##     "Sepal.Width": 2.7,
##     "Petal.Length": 4.2,
##     "Petal.Width": 1.3,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 5.7,
##     "Sepal.Width": 3,
##     "Petal.Length": 4.2,
##     "Petal.Width": 1.2,
##     "Species": "versicolor"
##   },
```

```
##   {
##     "Sepal.Length": 5.7,
##     "Sepal.Width": 2.9,
##     "Petal.Length": 4.2,
##     "Petal.Width": 1.3,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 6.2,
##     "Sepal.Width": 2.9,
##     "Petal.Length": 4.3,
##     "Petal.Width": 1.3,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 5.1,
##     "Sepal.Width": 2.5,
##     "Petal.Length": 3,
##     "Petal.Width": 1.1,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 5.7,
##     "Sepal.Width": 2.8,
##     "Petal.Length": 4.1,
##     "Petal.Width": 1.3,
##     "Species": "versicolor"
##   },
##   {
##     "Sepal.Length": 6.3,
##     "Sepal.Width": 3.3,
##     "Petal.Length": 6,
##     "Petal.Width": 2.5,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 5.8,
##     "Sepal.Width": 2.7,
##     "Petal.Length": 5.1,
##     "Petal.Width": 1.9,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 7.1,
##     "Sepal.Width": 3,
##     "Petal.Length": 5.9,
##     "Petal.Width": 2.1,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.3,
##     "Sepal.Width": 2.9,
##     "Petal.Length": 5.6,
##     "Petal.Width": 1.8,
```

```
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.5,
##     "Sepal.Width": 3,
##     "Petal.Length": 5.8,
##     "Petal.Width": 2.2,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 7.6,
##     "Sepal.Width": 3,
##     "Petal.Length": 6.6,
##     "Petal.Width": 2.1,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 4.9,
##     "Sepal.Width": 2.5,
##     "Petal.Length": 4.5,
##     "Petal.Width": 1.7,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 7.3,
##     "Sepal.Width": 2.9,
##     "Petal.Length": 6.3,
##     "Petal.Width": 1.8,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.7,
##     "Sepal.Width": 2.5,
##     "Petal.Length": 5.8,
##     "Petal.Width": 1.8,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 7.2,
##     "Sepal.Width": 3.6,
##     "Petal.Length": 6.1,
##     "Petal.Width": 2.5,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.5,
##     "Sepal.Width": 3.2,
##     "Petal.Length": 5.1,
##     "Petal.Width": 2,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.4,
##     "Sepal.Width": 2.7,
```

```
##     "Petal.Length": 5.3,
##     "Petal.Width": 1.9,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.8,
##     "Sepal.Width": 3,
##     "Petal.Length": 5.5,
##     "Petal.Width": 2.1,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 5.7,
##     "Sepal.Width": 2.5,
##     "Petal.Length": 5,
##     "Petal.Width": 2,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 5.8,
##     "Sepal.Width": 2.8,
##     "Petal.Length": 5.1,
##     "Petal.Width": 2.4,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.4,
##     "Sepal.Width": 3.2,
##     "Petal.Length": 5.3,
##     "Petal.Width": 2.3,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.5,
##     "Sepal.Width": 3,
##     "Petal.Length": 5.5,
##     "Petal.Width": 1.8,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 7.7,
##     "Sepal.Width": 3.8,
##     "Petal.Length": 6.7,
##     "Petal.Width": 2.2,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 7.7,
##     "Sepal.Width": 2.6,
##     "Petal.Length": 6.9,
##     "Petal.Width": 2.3,
##     "Species": "virginica"
##   },
##   {
```

```
##     "Sepal.Length": 6,
##     "Sepal.Width": 2.2,
##     "Petal.Length": 5,
##     "Petal.Width": 1.5,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.9,
##     "Sepal.Width": 3.2,
##     "Petal.Length": 5.7,
##     "Petal.Width": 2.3,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 5.6,
##     "Sepal.Width": 2.8,
##     "Petal.Length": 4.9,
##     "Petal.Width": 2,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 7.7,
##     "Sepal.Width": 2.8,
##     "Petal.Length": 6.7,
##     "Petal.Width": 2,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.3,
##     "Sepal.Width": 2.7,
##     "Petal.Length": 4.9,
##     "Petal.Width": 1.8,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.7,
##     "Sepal.Width": 3.3,
##     "Petal.Length": 5.7,
##     "Petal.Width": 2.1,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 7.2,
##     "Sepal.Width": 3.2,
##     "Petal.Length": 6,
##     "Petal.Width": 1.8,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.2,
##     "Sepal.Width": 2.8,
##     "Petal.Length": 4.8,
##     "Petal.Width": 1.8,
##     "Species": "virginica"
```

```
##   },
##   {
##     "Sepal.Length": 6.1,
##     "Sepal.Width": 3,
##     "Petal.Length": 4.9,
##     "Petal.Width": 1.8,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.4,
##     "Sepal.Width": 2.8,
##     "Petal.Length": 5.6,
##     "Petal.Width": 2.1,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 7.2,
##     "Sepal.Width": 3,
##     "Petal.Length": 5.8,
##     "Petal.Width": 1.6,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 7.4,
##     "Sepal.Width": 2.8,
##     "Petal.Length": 6.1,
##     "Petal.Width": 1.9,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 7.9,
##     "Sepal.Width": 3.8,
##     "Petal.Length": 6.4,
##     "Petal.Width": 2,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.4,
##     "Sepal.Width": 2.8,
##     "Petal.Length": 5.6,
##     "Petal.Width": 2.2,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.3,
##     "Sepal.Width": 2.8,
##     "Petal.Length": 5.1,
##     "Petal.Width": 1.5,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.1,
##     "Sepal.Width": 2.6,
##     "Petal.Length": 5.6,
```

```
##     "Petal.Width": 1.4,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 7.7,
##     "Sepal.Width": 3,
##     "Petal.Length": 6.1,
##     "Petal.Width": 2.3,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.3,
##     "Sepal.Width": 3.4,
##     "Petal.Length": 5.6,
##     "Petal.Width": 2.4,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.4,
##     "Sepal.Width": 3.1,
##     "Petal.Length": 5.5,
##     "Petal.Width": 1.8,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6,
##     "Sepal.Width": 3,
##     "Petal.Length": 4.8,
##     "Petal.Width": 1.8,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.9,
##     "Sepal.Width": 3.1,
##     "Petal.Length": 5.4,
##     "Petal.Width": 2.1,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.7,
##     "Sepal.Width": 3.1,
##     "Petal.Length": 5.6,
##     "Petal.Width": 2.4,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.9,
##     "Sepal.Width": 3.1,
##     "Petal.Length": 5.1,
##     "Petal.Width": 2.3,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 5.8,
```

```
##     "Sepal.Width": 2.7,
##     "Petal.Length": 5.1,
##     "Petal.Width": 1.9,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.8,
##     "Sepal.Width": 3.2,
##     "Petal.Length": 5.9,
##     "Petal.Width": 2.3,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.7,
##     "Sepal.Width": 3.3,
##     "Petal.Length": 5.7,
##     "Petal.Width": 2.5,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.7,
##     "Sepal.Width": 3,
##     "Petal.Length": 5.2,
##     "Petal.Width": 2.3,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.3,
##     "Sepal.Width": 2.5,
##     "Petal.Length": 5,
##     "Petal.Width": 1.9,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.5,
##     "Sepal.Width": 3,
##     "Petal.Length": 5.2,
##     "Petal.Width": 2,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 6.2,
##     "Sepal.Width": 3.4,
##     "Petal.Length": 5.4,
##     "Petal.Width": 2.3,
##     "Species": "virginica"
##   },
##   {
##     "Sepal.Length": 5.9,
##     "Sepal.Width": 3,
##     "Petal.Length": 5.1,
##     "Petal.Width": 1.8,
##     "Species": "virginica"
##   }
```

```
## ]
```

```
# upload again
iris2 <- fromJSON(myjson)
head(iris2)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

mySQL Stuff

- Free, widely-used open source database, esp in internet-based applications.
- Data structured in databases, inside is tables(dataset), inside is fields(column), row is record. IDs link databases.
- http://en.wikipedia.org/wiki/MySQL and http://www.mysql.com/
- Install MySQL: https://dev.mysql.com/doc/refman/5.7/en/installing.html
- http://cran.r-project.org/web/packages/RMySQL/RMySQL.pdf
- Commands: http://www.pantz.org/software/mysql/mysqlcommands.html and http://www.r-bloggers.com/mysql-and-r
- Be careful with SQL commands, *DO NOT PUSH UNLESS REQUIRED!*

```
library (RMySQL)
```

```
## Warning: package 'RMySQL' was built under R version 4.4.2
```

```
## Loading required package: DBI
```

```
## Warning: package 'DBI' was built under R version 4.4.2
```

```
ucscDb <- dbConnect(MySQL(), user="genome",host="genome-mysql.cse.ucsc.edu") # open a connection to the
result <- dbGetQuery(ucscDb, "show databases;"); dbDisconnect(ucscDb); # run the mySQL command "show da
```

```
## [1] TRUE
```

```
result # print available databases
```

```
##               Database
## 1              acaChl1
## 2              ailMel1
```

```
## 3              allMis1
## 4              allSin1
## 5              amaVit1
## 6              anaPla1
## 7              ancCey1
## 8              angJap1
## 9              anoCar1
## 10             anoCar2
## 11             anoGam1
## 12             anoGam3
## 13             apaSpi1
## 14             apaVit1
## 15             apiMel1
## 16             apiMel2
## 17  apiMel2_justpushed
## 18             aplCal1
## 19             aptFor1
## 20             aptMan1
## 21             aquChr2
## 22             araMac1
## 23             ascSuu1
## 24             balAcu1
## 25             balPav1
## 26             bisBis1
## 27             bosTau2
## 28             bosTau3
## 29             bosTau4
## 30             bosTau5
## 31             bosTau6
## 32             bosTau7
## 33             bosTau8
## 34             bosTau9
## 35          bosTauMd3
## 36             braFlo1
## 37             bruMal2
## 38             bucRhi1
## 39             burXyl1
## 40             caeAng2
## 41             caeJap1
## 42             caeJap4
## 43              caePb1
## 44              caePb2
## 45              caePb3
## 46             caeRem2
## 47             caeRem3
## 48             caeRem4
## 49            caeSp111
## 50             caeSp51
## 51             calAnn1
## 52             calJac1
## 53             calJac3
## 54             calJac4
## 55             calMil1
## 56             canFam1
```

```
## 57            canFam2
## 58            canFam3
## 59            canFam4
## 60            canFam5
## 61            canFam6
## 62            capCar1
## 63            carCri1
## 64            cavPor3
## 65                cb1
## 66                cb3
## 67                cb4
## 68               ce10
## 69               ce11
## 70                ce2
## 71                ce4
## 72                ce6
## 73            cerSim1
## 74            chaVoc2
## 75            cheMyd1
## 76            chlSab2
## 77            chlUnd1
## 78            choHof1
## 79            chrPic1
## 80            chrPic2
## 81                ci1
## 82                ci2
## 83                ci3
## 84            colLiv1
## 85            colStr1
## 86            corBra1
## 87            corCor1
## 88            cotJap2
## 89            criGri1
## 90         criGriChoV1
## 91         criGriChoV2
## 92            cucCan1
## 93            danRer1
## 94           danRer10
## 95           danRer11
## 96            danRer2
## 97            danRer3
## 98            danRer4
## 99            danRer5
## 100           danRer6
## 101           danRer7
## 102           dasNov3
## 103           dipOrd1
## 104           dirImm1
## 105               dm1
## 106               dm2
## 107               dm3
## 108               dm6
## 109               dp2
## 110               dp3
```

```
## 111           droAna1
## 112           droAna2
## 113           droEre1
## 114           droGri1
## 115           droMoj1
## 116           droMoj2
## 117           droPer1
## 118           droSec1
## 119           droSim1
## 120           droSim2
## 121           droVir1
## 122           droVir2
## 123           droYak1
## 124           droYak2
## 125           eboVir3
## 126           echTel1
## 127           echTel2
## 128           egrGar1
## 129        enhLutNer1
## 130           equCab1
## 131           equCab2
## 132           equCab3
## 133           eriEur1
## 134           eriEur2
## 135           eurHel1
## 136           falChe1
## 137           falPer1
## 138           felCat3
## 139           felCat4
## 140           felCat5
## 141           felCat8
## 142           felCat9
## 143           ficAlb2
## 144               fr1
## 145               fr2
## 146               fr3
## 147           fulGla1
## 148           gadMor1
## 149           galGal2
## 150           galGal3
## 151           galGal4
## 152           galGal5
## 153           galGal6
## 154           galVar1
## 155           gasAcu1
## 156           gavSte1
## 157            gbMeta
## 158           geoFor1
## 159                go
## 160          go080130
## 161          go140213
## 162          go150121
## 163          go180426
## 164           gorGor3
```

```
## 165            gorGor4
## 166            gorGor5
## 167            gorGor6
## 168            haeCon2
## 169            halAlb1
## 170            halLeu1
## 171            hetBac1
## 172            hetGla1
## 173            hetGla2
## 174               hg16
## 175               hg17
## 176               hg18
## 177               hg19
## 178         hg19Patch10
## 179         hg19Patch13
## 180    hg19_justpushed
## 181               hg38
## 182         hg38Patch11
## 183    hg38_justpushed
## 184            hgFixed
## 185          hgcentral
## 186                hs1
## 187 information_schema
## 188  knownGeneKentHg19
## 189       knownGeneV39
## 190       knownGeneV43
## 191       knownGeneV44
## 192       knownGeneV45
## 193       knownGeneV46
## 194      knownGeneVM27
## 195      knownGeneVM30
## 196      knownGeneVM32
## 197            latCha1
## 198            lepDis1
## 199            letCam1
## 200            loaLoa1
## 201            loxAfr3
## 202            macEug1
## 203            macEug2
## 204            macFas5
## 205            manPen1
## 206            melGal1
## 207            melGal5
## 208            melHap1
## 209            melInc2
## 210            melUnd1
## 211            merNub1
## 212            mesUni1
## 213            micMur1
## 214            micMur2
## 215               mm10
## 216               mm39
## 217    mm39_justpushed
## 218                mm5
```

```
## 219                mm6
## 220                mm7
## 221                mm8
## 222                mm9
## 223            monDom1
## 224            monDom4
## 225            monDom5
## 226         mpxvRivers
## 227            musFur1
## 228            myoLuc2
## 229            nanPar1
## 230            nasLar1
## 231            necAme1
## 232            neoSch1
## 233            nipNip1
## 234            nomLeu1
## 235            nomLeu2
## 236            nomLeu3
## 237            ochPri2
## 238            ochPri3
## 239            oncVol1
## 240            opiHoa1
## 241            oreNil1
## 242            oreNil2
## 243            oreNil3
## 244            ornAna1
## 245            ornAna2
## 246            oryCun2
## 247            oryLat2
## 248            otoGar3
## 249            oviAri1
## 250            oviAri3
## 251            oviAri4
## 252            panPan1
## 253            panPan2
## 254            panPan3
## 255            panRed1
## 256            panTro1
## 257            panTro2
## 258            panTro3
## 259            panTro4
## 260            panTro5
## 261            panTro6
## 262            papAnu2
## 263            papAnu4
## 264            papHam1
## 265            pelCri1
## 266            pelSin1
## 267 performance_schema
## 268            petMar1
## 269            petMar2
## 270            petMar3
## 271            phaCar1
## 272            phaLep1
```

```
## 273           phoRub1
## 274           picPub1
## 275           ponAbe2
## 276           ponAbe3
## 277           priExs1
## 278           priPac1
## 279           priPac3
## 280           proCap1
## 281    proteins120806
## 282    proteins121210
## 283    proteins140122
## 284    proteins150225
## 285    proteins160229
## 286    proteins180404
## 287          proteome
## 288           pteGut1
## 289           pteVam1
## 290           pygAde1
## 291           pytBiv1
## 292           rheMac1
## 293          rheMac10
## 294           rheMac2
## 295           rheMac3
## 296           rheMac8
## 297           rhiRox1
## 298               rn3
## 299               rn4
## 300               rn5
## 301               rn6
## 302               rn7
## 303           sacCer1
## 304           sacCer2
## 305           sacCer3
## 306           saiBol1
## 307           sarHar1
## 308           serCan1
## 309           sorAra1
## 310           sorAra2
## 311          sp120323
## 312          sp121210
## 313          sp140122
## 314          sp150225
## 315          sp160229
## 316          sp180404
## 317           speTri2
## 318           strCam1
## 319           strPur1
## 320           strPur2
## 321           strRat2
## 322          susScr11
## 323           susScr2
## 324           susScr3
## 325           taeGut1
## 326           taeGut2
```

```
## 327            tarSyr1
## 328            tarSyr2
## 329            tauEry1
## 330            tetNig1
## 331            tetNig2
## 332            thaSir1
## 333            tinGut2
## 334            triMan1
## 335            triSpi1
## 336            triSui1
## 337            tupBel1
## 338            turTru2
## 339            tytAlb1
## 340            uniProt
## 341            vicPac1
## 342            vicPac2
## 343          visiGene
## 344            wuhCor1
## 345            xenLae2
## 346            xenTro1
## 347          xenTro10
## 348            xenTro2
## 349            xenTro3
## 350            xenTro7
## 351            xenTro9
## 352            zonAlb1
```

```
hg19 <- dbConnect(MySQL(), user="genome", db = "hg19", host="genome-mysql.cse.ucsc.edu") # connection t
allTables <- dbListTables(hg19) # extract names of all tables
length(allTables)
```

```
## [1] 12690
```

```
allTables[1:5]
```

```
## [1] "HInv"         "HInvGeneMrna" "acembly"      "acemblyClass" "acemblyPep"
```

```
dbListFields(hg19,"affyU133Plus2") # fields/columns of table affyU133Plus2 (is a microarray)
```

```
##  [1] "bin"         "matches"     "misMatches"  "repMatches"  "nCount"
##  [6] "qNumInsert"  "qBaseInsert" "tNumInsert"  "tBaseInsert" "strand"
## [11] "qName"       "qSize"       "qStart"      "qEnd"        "tName"
## [16] "tSize"       "tStart"      "tEnd"        "blockCount"  "blockSizes"
## [21] "qStarts"     "tStarts"
```

```
dbGetQuery(hg19, "select count(*) from affyU133Plus2") # how many rows/record. Pass mySQL command to co
```

```
##   count(*)
## 1    58463
```

```r
library(knitr) # to supress warnings
```

```
## Warning: package 'knitr' was built under R version 4.4.2
```

```r
suppressWarnings({ affyData <- dbReadTable(hg19, "affyU133Plus2") }) # import data from table in mySQL
head(affyData)
```

```
##   bin matches misMatches repMatches nCount qNumInsert qBaseInsert tNumInsert
## 1 585     530          4          0     23          3          41          3
## 2 585    3355         17          0    109          9          67          9
## 3 585    4156         14          0     83         16          18          2
## 4 585    4667          9          0     68         21          42          3
## 5 585    5180         14          0    167         10          38          1
## 6 585     468          5          0     14          0           0          0
##   tBaseInsert strand        qName qSize qStart qEnd tName     tSize tStart
## 1         898      -  225995_x_at   637      5  603  chr1 249250621  14361
## 2       11621      -  225035_x_at  3635      0 3548  chr1 249250621  14381
## 3          93      -  226340_x_at  4318      3 4274  chr1 249250621  14399
## 4        5743      - 1557034_s_at  4834     48 4834  chr1 249250621  14406
## 5          29      -    231811_at  5399      0 5399  chr1 249250621  19688
## 6           0      -    236841_at   487      0  487  chr1 249250621  27542
##    tEnd blockCount
## 1 15816          5
## 2 29483         17
## 3 18745         18
## 4 24893         23
## 5 25078         11
## 6 28029          1
##                                                                   blockSizes
## 1                                                      93,144,229,70,21,
## 2              73,375,71,165,303,360,198,661,201,1,260,250,74,73,98,155,163,
## 3                 690,10,32,33,376,4,5,15,5,11,7,41,277,859,141,51,443,1253,
## 4 99,352,286,24,49,14,6,5,8,149,14,44,98,12,10,355,837,59,8,1500,133,624,58,
## 5                                       131,26,1300,6,4,11,4,7,358,3359,155,
## 6                                                                       487,
##                                                                                                  qSta
## 1                                                                                    34,132,278,541,6
## 2                        87,165,540,647,818,1123,1484,1682,2343,2545,2546,2808,3058,3133,3206,3317,34
## 3                   44,735,746,779,813,1190,1195,1201,1217,1223,1235,1243,1285,1564,2423,2565,2617,30
## 4 0,99,452,739,764,814,829,836,842,851,1001,1016,1061,1160,1173,1184,1540,2381,2441,2450,3951,4103,47
## 5                                                           0,132,159,1460,1467,1472,1484,1489,1497,1856,52
## 6
##
## 1
## 2                                       14381,14454,14969,15075,15240,15543,15903,16104,16853,17054,172
## 3                                 14399,15089,15099,15131,15164,15540,15544,15549,15564,15569,15580,155
## 4 14406,20227,20579,20865,20889,20938,20952,20958,20963,20971,21120,21134,21178,21276,21288,21298,216
## 5                                                                                   19688,19819,19845,21145,211
## 6
```

```r
suppressWarnings({ query <- dbSendQuery(hg19, "select * from affyU133Plus2 where misMatches between 1 an
affyMis <- fetch(query); quantile(affyMis$misMatches) # fetch subset from database and run quantile on
```

```
##   0%  25%  50%  75% 100%
##    1    1    2    2    3
```

```
affyMisSmall <- fetch(query,n=10); dbClearResult(query);# fetch top 10 records
```

```
## [1] TRUE
```

```
dim(affyMisSmall)
```

```
## [1] 10 22
```

```
dbDisconnect(hg19) # CLOSE CONNECTION
```

```
## [1] TRUE
```

Just HDF5 Stuff

- Large data sets, store range of data types, Hierarchical Data Format.
- Groups of 0+ data sets and their metadata: group header (with name and list of attributes) and group symbol table (with list of objects in groups).
- Datasets are multi-dimensional array of data elements with their metadata: header (with name, datatype, dataspace, and storage layout) and data array (containing data).
- http://www.hdfgroup.org/
- Install: if (!require("BiocManager", quietly = TRUE)) install.packages("BiocManager"); BiocManager::install(version = "3.20"); BiocManager::install(pkgs=c("rhdf5"));
- https://www.bioconductor.org/packages/release/bioc/html/rhdf5.html

```
library(rhdf5)
```

```
## Warning: package 'rhdf5' was built under R version 4.4.2
```

```
file.remove("example.h5")
```

```
## [1] TRUE
```

```
created = h5createFile("example.h5") # create hdf5 file
created
```

```
## [1] TRUE
```

```
created = h5createGroup("example.h5","foo") # greate group with given name inside h5
created = h5createGroup("example.h5","baa")
created = h5createGroup("example.h5","foo/foobaa") # create subgroup
h5ls("example.h5") # list components of h5 file
```

```
##   group   name     otype dclass dim
## 0     /    baa H5I_GROUP
## 1     /    foo H5I_GROUP
## 2  /foo foobaa H5I_GROUP
```

```
A <- matrix(1:10,nrow=5,ncol=2)
h5write(A, "example.h5","foo/A") # write object to group
B <- array(seq(0.1,2.0,by=0.1),dim=c(5,2,2))
attr(B,"scale") <- "liter" # can include metadata
h5write(B, "example.h5","foo/foobaa/B")
h5ls("example.h5")
```

```
##         group   name       otype  dclass       dim
## 0           /    baa   H5I_GROUP
## 1           /    foo   H5I_GROUP
## 2        /foo      A H5I_DATASET INTEGER     5 x 2
## 3        /foo foobaa   H5I_GROUP
## 4 /foo/foobaa      B H5I_DATASET   FLOAT 5 x 2 x 2
```

```
df <- data.frame(1L:5L,seq(0,1,length.out=5), c("ab","cde","fghi","a","s"), stringsAsFactors = FALSE)
h5write(df, "example.h5","df") # write data set to top level group
h5ls("example.h5")
```

```
##         group   name       otype   dclass       dim
## 0           /    baa   H5I_GROUP
## 1           /     df H5I_DATASET COMPOUND         5
## 2           /    foo   H5I_GROUP
## 3        /foo      A H5I_DATASET  INTEGER     5 x 2
## 4        /foo foobaa   H5I_GROUP
## 5 /foo/foobaa      B H5I_DATASET    FLOAT 5 x 2 x 2
```

```
h5read("example.h5","foo/A") # read data from hf file
```

```
##      [,1] [,2]
## [1,]    1    6
## [2,]    2    7
## [3,]    3    8
## [4,]    4    9
## [5,]    5   10
```

```
h5write(c(12,13,14),"example.h5","foo/A",index=list(1:3,1)) # can read and write (in this case edit) in
h5read("example.h5","foo/A")
```

```
##      [,1] [,2]
## [1,]   12    6
## [2,]   13    7
## [3,]   14    8
## [4,]    4    9
## [5,]    5   10
```

```
rm(list=ls())
```

HTML and Webscraping

- Webscraping: programatically extracting data from HTML code of websites. Be careful of Terms of Service and speed of scraping...
- http://en.wikipedia.org/wiki/Web_scraping
- http://www.r-bloggers.com/?s=Web+Scraping
- http://cran.r-project.org/web/packages/httr/httr.pdf

```
con = url("https://scholar.google.com/citations?user=HI-I6C0AAAAJ&hl=en") # open connection to website
htmlCode = readLines(con) # extract file lines into a char vector
```

```
## Warning in readLines(con): incomplete final line found on
## 'https://scholar.google.com/citations?user=HI-I6C0AAAAJ&hl=en'
```

```
close(con) # REMEMBER
htmlCode[1] # Hard to read
```

```
## [1] "<!doctype html><html><head><title>Jeff Leek - Google Scholar</title><meta http-equiv=\"Content-
```

```
# alternate method: XML
library(XML)
url <- "http://web.archive.org/web/20130207021632/http://scholar.google.com:80/citations?user=HI-I6C0AA
html <- htmlTreeParse(url,useInternalNodes = TRUE) # parsing file using internal nodes
xpathSApply(html, "//title", xmlValue) # use sApply for clean vector. Accessing title
```

```
## [1] "Jeff Leek - Google Scholar Citations"
```

```
xpathSApply(html, "//td[@id='col-citedby']", xmlValue) # accessing elements of table
```

```
##  [1] "Cited by" "339"      "173"      "140"      "133"      "107"
##  [7] "95"       "79"       "79"       "54"       "17"       "10"
## [13] "9"        "8"        "8"        "8"        "8"        "6"
## [19] "6"        "5"        "3"
```

```
# alternate method: httr and GET
library(httr)
html2 <- GET("https://scholar.google.com/citations?user=HI-I6C0AAAAJ&hl=en")
content2 <- content(html2, as = "text") # extract content from HTML page as one text string
parsedHtml <- htmlParse(content2, asText = TRUE) # parse the text, same as xml package result
xpathSApply(parsedHtml, "//title", xmlValue)
```

```
## [1] "Jeff Leek - Google Scholar"
```

```
# passwords
url <- "http://httpbin.org/basic-auth/user/passwd"
pg1 <- GET(url)
pg1 # not authenticated, passes status 401
```

```
## Response [http://httpbin.org/basic-auth/user/passwd]
##   Date: 2025-01-22 05:36
##   Status: 401
##   Content-Type: <unknown>
## <EMPTY BODY>
```

```
pg2 <- GET(url, authenticate("user","passwd")) # sends authentication
pg2
```

```
## Response [http://httpbin.org/basic-auth/user/passwd]
##   Date: 2025-01-22 05:36
##   Status: 200
##   Content-Type: application/json
##   Size: 47 B
## {
##   "authenticated": true,
##   "user": "user"
## }
```

```
names(pg2)
```

```
##  [1] "url"         "status_code" "headers"     "all_headers" "cookies"
##  [6] "content"     "date"        "times"       "request"     "handle"
```

```
# use handles
google <- handle("http://google.com") # to save authentication across multiple accesses to website
pg1 <- GET(handle = google, path = "/")
pg2 <- GET(handle = google, path = "search")
```

APIs

- Application programming interfaces.
- Developed platform will have GET request URLs (and parameters), which you use in httr to get data. Need a API/dev account, need to submit a request (for each project!). Receive: consumer key, consumer secret, request token URL, and Authorization URL.
- Able to GET, POST, PUT, DELETE with httr if authorized.
- Need oauth in most cases, sometimes username/password allowed.
- Sites: Facebook, Google, Twitter, GitHub...

```
# in personal vault :)
```

Other Sources

Packages for Data Storage Mechanisms - Search Google: "data storage mechanism R package"
- ?connections to get info on creating connection (CLOSE)
- foreign package: loads data from Minitab, S, SAS, SPSS, Stata, Systat. read.lang. http://cran.r-project.org/web/packages/foreign/foreign.pdf
- RPostresSQL: https://code.google.com/p/rpostgresql/, http://cran.r-project.org/web/packages/RPostgreSQL/RPostgreSQL.pdf
- RODBC (PostgreQL, MySQL, Microsoft Access, SQLite): http://cran.r-project.org/web/packages/RODBC/vignettes/RODBC.pdf, http://cran.r-project.org/web/packages/RODBC/RODBC.pdf
- RMongo/rmongodb: http://cran.r-project.org/web/packages/RMongo/RMongo.pdf, http://www.r-bloggers.com/r-and-mongodb

Image Data - jpeg: http://cran.r-project.org/web/packages/jpeg/index.html
- readbitmap: http://cran.r-project.org/web/packages/readbitmap/index.html
- png: http://cran.r-project.org/web/packages/png/index.html
- EBImage (Bioconductor): http://www.bioconductor.org/packages/2.13/bioc/html/EBImage.html

GIS Data - raster: http://cran.r-project.org/web/packages/raster/index.html
- 'sf' and 'terra'

Musical Data - tuneR: http://cran.r-project.org/web/packages/tuneR/
- seewave: http://rug.mnhn.fr/seewave

Cleaning Data

- End of process generate: raw data, tidy data set, code book (metadata) describing each variable and its values in the tidy data set, explicit and exact recipe used to convert raw data to tidy data set and code book.

- Raw Data: original source of data i.e. no processing, editing, summarizing. Must process for data analysis (merging, sub-setting, transforming, etc.), be mindful of processing standards. Colloquially may be later step i.e. genome seq but must use rawest.

- Processed Data: ready for analysis, steps to reach stage must be recorded. Must: one variable per column, each observation in a separate row, different tables for different types of variables, if multiple tables allow for linking. Useful: top row of variable names which are human readable, save one file per table.

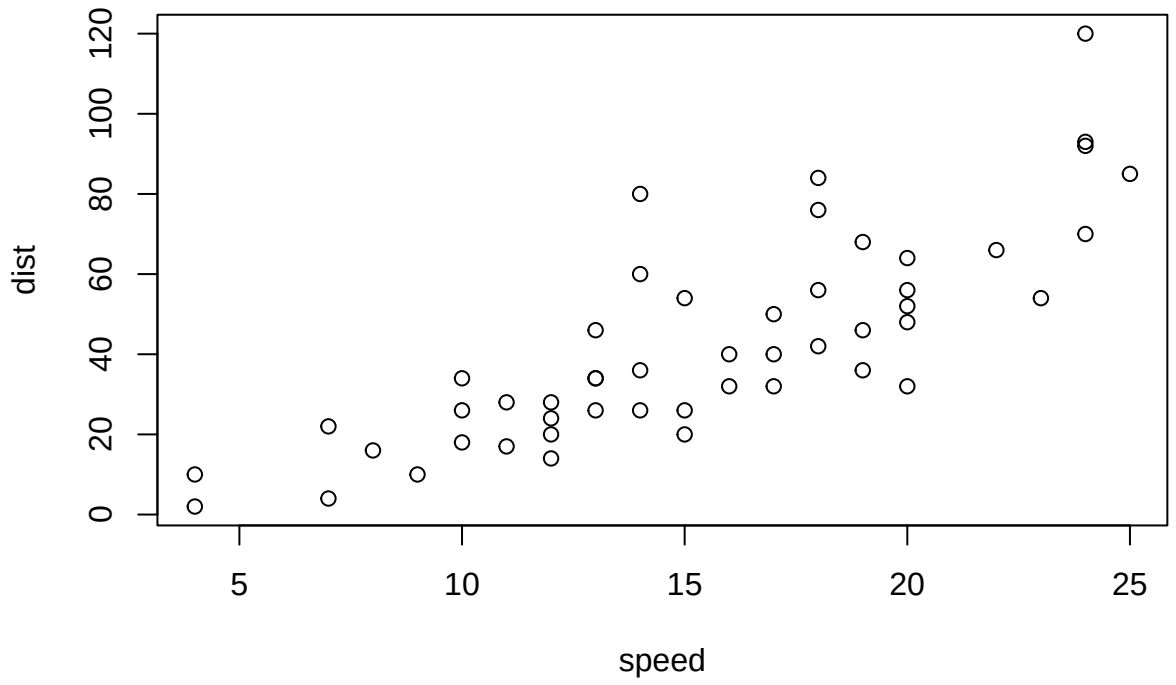
- Code book: info about variables not contained in data incl. units called *Code book*, info about summary choices, info about experimental study design called *Study design*. Often word/text file.

- Instruction list: in a computer script where input is raw data and output is tidy data with no parameters to the script. If not possible, provide instructions in steps (incl. parameters, software versions, how to use software).

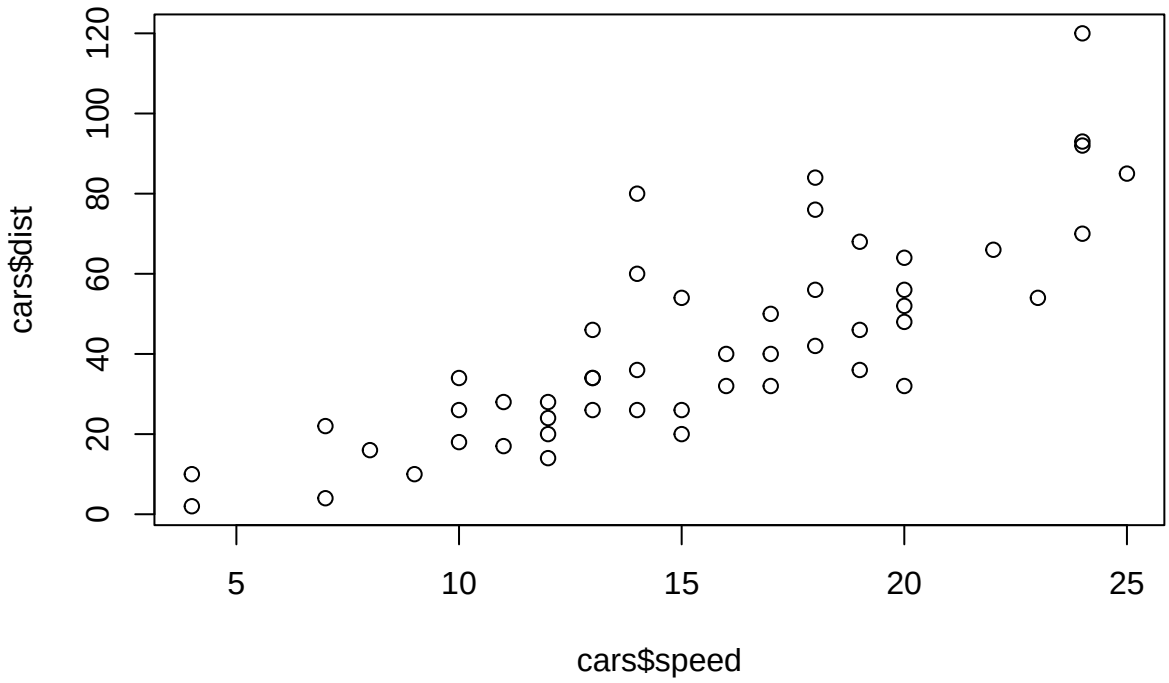
Graphics

http://www.ling.upenn.edu/~joseff/rstudy/week4.html

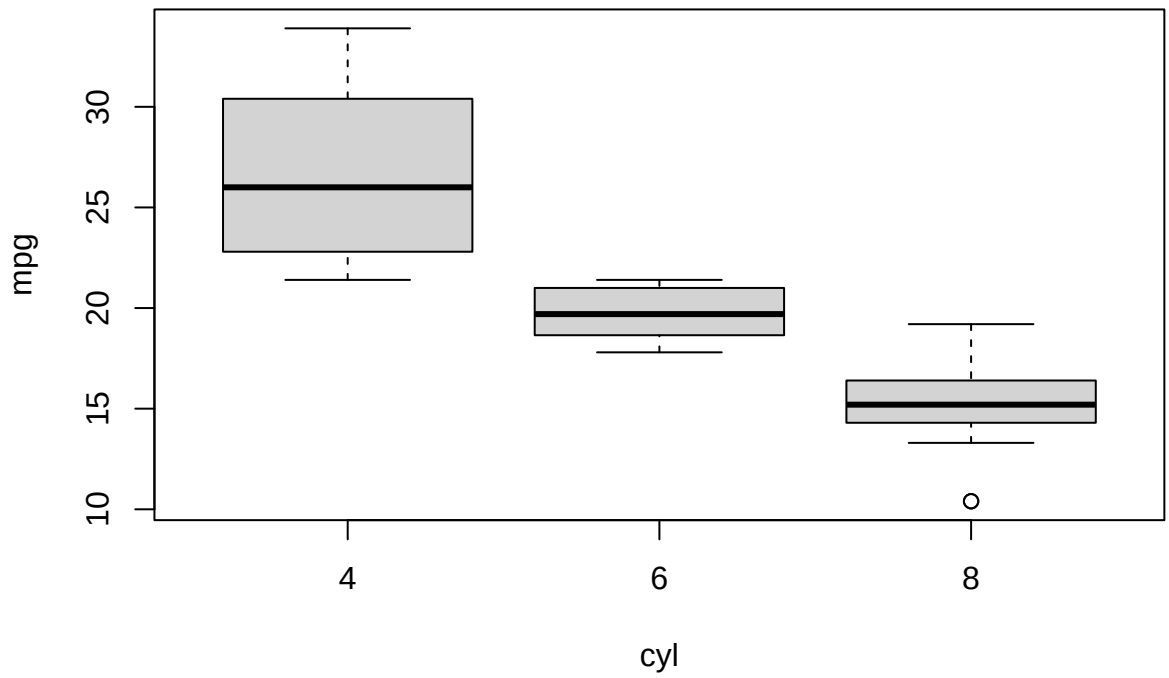
```r
# Start by getting sense of the data: dim(), names(), head(), tail() and summary().

# Scatterplot
data(cars)
plot(cars) # generates scatterplot with two columns against each other
```



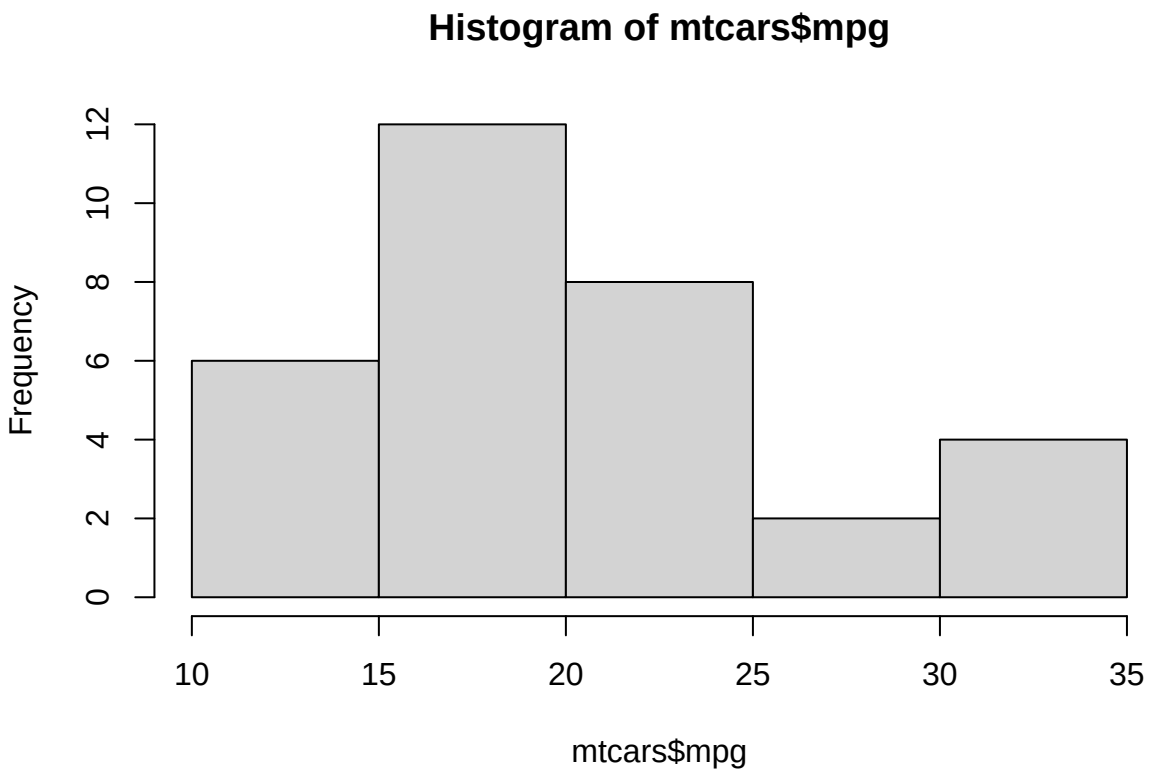

```r
plot(x = cars$speed, y = cars$dist) # explicit, can also pass plot(dist ~ speed, cars). Args: x, y = NU
```




```r
# Boxplot
data(mtcars)
boxplot(mpg ~ cyl, data = mtcars) # generate relationship between mpg(x) and cyl(y) from mtcars
```




```
# Histogram
hist(mtcars$mpg) # generate a histogram of vector
```



R Profiler and Optimization

- Systematic way to examine time spent in various part of the program. Useful to optimize the code.
- DON'T PREMATURELY OPTIMIZE
- Measure, not guess, data on what needs to be optimized.
- User time: computer experienced, may be greater if multiple cores/processors (accessible in multi-threaded BLAS libraries). Elapsed time: wall-clock time, may be greater if other computing tasks.

```
system.time(read.csv("hw1_data.csv")) # returns seconds to execute, if error then seconds to error. Wra
```

```
##    user  system elapsed
##       0       0       0
```

```
data(mtcars)
Rprof() # track function call stack at intervals (def = 0.02 sec), time spent in functions.

fit <- lm(mtcars$mpg ~ mtcars$cyl)

Rprof(NULL)

summaryRprof() # makes Rprof readable, tabluates, time in each function
```

```
## $by.self
## [1] self.time  self.pct   total.time total.pct
## <0 rows> (or 0-length row.names)
##
## $by.total
## [1] total.time total.pct  self.time  self.pct
## <0 rows> (or 0-length row.names)
##
## $sample.interval
## [1] 0.02
##
## $sampling.time
## [1] 0
```

```
# $by.total - divides time spent per function by total run time
# $by.self - same as by.total but first subtracts time spent in function above in call stack. Helps tar

rm(list=ls())
```