*Majority of effort is contained in exploring and preparing the data*

* R Data Strx
  + Vectors
    - Types: *integer*, *numeric* (decimal), *character*, *logical* (T/F)
    - Special Values: *NULL* – absence of value, *NA* – missing value
    - Create using c(), name using <-
      * > subject\_name <- c("John Doe", "Jane Doe", "Steve Graves")
    - Inherently ordered
    - 1-indexed, not 0-indexed
    - Can access ranges: names[2:3], and also exclusions: names[-2]
    - Specific accesses using names[c(TRUE, FALSE, TRUE, TRUE)]
      * Includes 1st,3rd,4th elements, not 2nd
  + Factors
    - Recall that features representing categorical characteristics are *nominal*
      * *E.g. gender, car color*
    - While we can use a character vector to store nominal data, we use factors
    - Special vector subtype
    - More efficient than character vectors
    - Needed for some ML algorithms using special routines
    - > colors <- factor(c(“**RED**”, “**RED**”, “**BLUE**”, “**RED**”))

> colors  
[1] **RED** **RED** **BLUE** **RED**

Levels: **BLUE** **RED**

* + - We can add levels
    - > blood <- factor(c(“**O**”, “**AB**”, “**A**”),

levels = c(“**A**”, “**B**”, “**AB**”, “**O**”))

> blood

[1] **O** **AB** **A**

Levels: **A B AB O**

* + Lists
    - We have the above vectors describing one persons name, car color, blood type
      * How do we access this all at once, instead of *n* accesses?
    - > subject1 <- list(fullname = subject\_name[1], temperature = temp[1], color = color[1], ….)
    - Calling just ‘subject1’ will print out all relevant data now
    - Specific accesses: subject[2] or subject$temperature
  + Data Frames
    - Similar to a spreadsheet
    - > pt\_data <- data.frame(subject\_name, temperature, flu\_status, gender,

blood, stringsAsFactors = FALSE)

* + - stringAsFactors needed, else R auto-converts ever character vector to a factor
    - This is innapropriate since names aren’t categorical data

> pt\_data   
subject\_name temperature flu\_status gender blood

1 John Doe 98.1 FALSE MALE O

2 Jane Doe 98.6 FALSE FEMALE AB

3 Steve Graves 101.4 TRUE MALE A

*Specific Columns*  
> pt\_data$subject\_name

[1] "John Doe" "Jane Doe" "Steve Graves"

*Specific Columns*

> pt\_data[c("temperature", "flu\_status")] //equivalent to pt\_data[2:3]

temperature flu\_status

1 98.1 FALSE

2 98.6 FALSE

3 101.4 TRUE

*Rows 1 and 3, and columns 2 and 4:*

> pt\_data[c(1, 3), c(2, 4)]

temperature gender

1 98.1 MALE   
3 101.4 MALE

*All rows of the first column*

> pt\_data[, 1]

[1] "John Doe" "Jane Doe" "Steve Graves"

*All columns of the first row*

> pt\_data[1, ]

subject\_name temperature flu\_status gender blood

1 John Doe 98.1 FALSE MALE O

*Everything:*

> pt\_data[ , ]

*Can access as lists, vectors:*

> pt\_data[c(1, 3), c("temperature", "gender")]

Is equivalent to:

> pt\_data[-2, c(-1, -3, -5)]

* + Matricies
    - > m <- matrix(c('a', 'b', 'c', 'd'), nrow = 2)  
      > m

[,1] [,2]

[1,] "a" "c"

[2,] "b" "d"

* + - > m <- matrix(c('a', 'b', 'c', 'd'), ncol = 2)   
      > m

[,1] [,2]

[1,] "a" "c"

[2,] "b" "d"

* + - *Column-major order*
  + Managing data with R
    - > save(x, y, z, file = "mydata.RData")
      * Where x,y,z are data strx
    - > load("mydata.RData")
    - To suspend – run save.image()
    - CSV Files
      * > pt\_data <- read.csv("/path/to/pt\_data.csv", stringsAsFactors = FALSE)
      * R by default assumes CSV includes header line
        + Else, use *‘header = FALSE’*

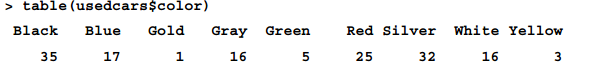
Creates default names *V1, V1, etc.*

* + - * *‘write.scv(pt\_data, file = “/path/to/pt\_data.csv”*
    - *‘read.csv()’* is special case of *‘read.table()’* which works on many forms, such as TSV
  + Exploring Data Strx
    - ‘*str()´* function provies data structure information
    - > str(usedcars) //can also check summary(usedcars$year)

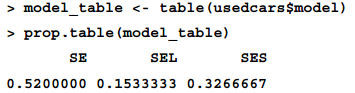
'data.frame': 150 obs. of 6 variables:

$ year : int 2011 2011 2011 2011 ...   
$ model : chr "SEL" "SEL" "SEL" "SEL" ...   
$ price : int 21992 20995 19995 17809 ...   
$ mileage : int 7413 10926 7351 11613 ...   
$ color : chr "Yellow" "Gray" "Silver" "Gray" ...   
$ transmission: chr "AUTO" "AUTO" "AUTO" "AUTO" ...

* + Other ways to Explore data
    - > mean(), median(), etc.
    - Range() // gives the two endpoints, to get invterval size: diff(range())
    - IQR() – interquartile range // Difference btween
    - quantile()
      * get arbitrary percentiles using ‘probs’:
      * quantile(usedcars$price, probs = c(0.01, 0.99))
    - boxplot(targetVar, options)
    - hist(targetVar, options)
    - plot(x, y, options) // can use plot(x = xTargetVar, y = yTargetVar, options)
    - var(x), sd(x)
  + Exploring categorical data







* + Data Visualization
    - Using crosstab, a two-way table
    - Often, use the gmodels package
    - We can split the used car data, suppose we only want to compare conservative vs. vibrant colors.
      * **
      * %in% asks the data, is each member of the data a member of this set? Returns True/False
      * > CrossTable(x = usedcars$model, y = usedcars$conservative)