First a few reminders…

• Pre-course survey (final due: Aug 28 by 11:59pm)
• Assignment 1 (due: Aug 30 by 6pm)
• R Tutorials: SPRK 339
  • Mon Aug 26, 3:10--4:30pm
  • Wed Aug 28, 3:10--4:30pm
  • Wed Sep 4, 3:10--4:30pm (likely Python tutorial)
• R Resources doc
  • Remember to check out a) the videos and b) the three interactive session notes
• Laptop/device use in class
Plan for today

• Left over topics from last lecture
• Graphics in R
Collections

Same types of data

Vector

<table>
<thead>
<tr>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>

Numeric

Character

Mixed types of data

List

<table>
<thead>
<tr>
<th>$name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1] &quot;Starbucks&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1] 5.25</td>
</tr>
</tbody>
</table>

1-Dimension

Matrix

<table>
<thead>
<tr>
<th>V1</th>
<th>V2</th>
<th>V3</th>
<th>V4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>7</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>8</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>9</td>
<td>14</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>10</td>
<td>15</td>
</tr>
</tbody>
</table>

Data frame

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Passed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>dog</td>
<td>TRUE</td>
</tr>
<tr>
<td>2</td>
<td>cat</td>
<td>FALSE</td>
</tr>
<tr>
<td>3</td>
<td>snake</td>
<td>FALSE</td>
</tr>
<tr>
<td>4</td>
<td>people</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

2-Dimension

https://trendct.org/2015/06/12/r-for-beginners-how-to-transition-from-excel-to-r/
Lists

• Lists are collections of items of any type
  • `l <- list("this", "is", 1, "list", TRUE)`

• Lists can also contain vectors, matrices, other lists, and data frames
Lists

- food <- list("potato", 163, 0.2, 37.0, 4.3)
- names(food) <- c("type", "calories", "fat", "carbs", "protein")
- food
  - $type
    - [1] "potato"
  - $calories
    - [1] 163
  - $fat
    - [1] 0.2
  - $carbs
    - [1] 37.0
  - $protein
    - [1] 4.3
- Alternatively
  - food <- list(type = "potato", calories = 163, fat = 0.2, carbs = 37.0, protein = 4.3)
Lists

- R indexing can be a bit confusing
  - `[[ ]]` Vs. `[]`
  - Generally, `[]` selects a *single element*, `[]` returns a *list of elements*
  - [https://cran.r-project.org/doc/manuals/R-lang.html#Indexing](https://cran.r-project.org/doc/manuals/R-lang.html#Indexing)

- `food[1]`
  - `$` type: “potato”

- `food[[1]] == food[[“type”]] == food$type`
  - “potato”
Lists

- `food <- list(type = “potato”, calories = 163, nutrition=c(fat=0.2, carbs = 37.0, protein = 4.3))`

- `food[1:3]`
  - `$ type: “potato”`
  - `$ calories:163`
  - `$ nutrition:
    - fat   carbs   protein
    - 0.2   37.0          4.3`

- `food[[3]][[2]] == food[[“nutrition”]][[“carbs”]] == food$nutrition$carbs`
  - 37.0

- `food$isTasty <- TRUE`
  - Adds a new boolean element called “isTasty” to food
Data Frames

- Data frame is essentially a database structure built into the language
  - Rows are observations (Joe, Mary, Sam, etc...)
  - Columns are variables (age, name, height, etc...)
  - Elements in a single column must be of the same type

- Example: declare a data frame for a roster:
  - `Roster <- data.frame(ID=integer(), FirstName=character(), LastName=character(), Grade=factor(levels = c("A", "A-", "B+", ... , "F")), stringAsFactors=FALSE)`
  - Can also read from file/web using `read.table()`
Data Frames

- head(Roster)
- summary(Roster)

To access a given column of a data frame:
- Roster$FirstName
- Roster[[“FirstName”]]

To access a single entry:
- Roster$Grade[2]
- Roster$ID[which(Roster$Grade == “A”)]
Extending Data Frames

• Add column
  • suppose you had a data frame called people, and you want to add weight info
  • weight <- c(132, 122, 184)
  • people$weight <- weight (or people[[“weight”]] <- weight)
  • Can also use cbind to add column
    • height <- c(60, 63, 67)
    • cbind(people, height)

• Add row
  • Create a new data frame with a single observation
  • Add the new data frame using rbind
  • E.g
    • john <- data.frame(ID=4, age=7, sex = 1, weight = 145, height = 59)
    • rbind(people, john)
Graphics in R
Graphics in R

• Both built in and third party libraries for graphics
• Popular examples include
  • ggplot2
  • ggvis
  • lattice
• We will start with built in graphics
• Basic functions to remember
  • plot()
  • hist()
Plotting Car Data

- str(cars)
  - 'data.frame': 38 obs. of 8 variables:
    - $ Country : Factor w/ 6 levels "France","Germany",...: 6 6 6 6 6 4 6 2 5 ...
    - $ Car : chr "Buick Estate Wagon", "Ford Country Squire Wagon", ...
    - $ MPG : num 16.9 15.5 19.2 18.5 30 27.5 27.2 30.9 20.3 17 ...
    - $ Weight: num 4.36 4.05 3.6 3.94 2.15 ...
    - $ Drive_Ratio: num 2.73 2.26 2.56 2.45 3.7 3.05 3.54 3.37 3.9 3.5 ...
    - $ Horsepower : int 155 142 125 150 68 95 97 75 103 125 ...
    - $ Displacement : int 350 351 267 360 98 134 119 105 131 163 ...
    - $ Cylinders : int 8 8 8 8 4 4 4 4 5 6 ...
- plot() is a generic plotting function
  - What it looks like depends on what you input
plot()

- plot(cars$Country)
- Country is a factor
- Thus, plot gives us a bar chart
- Shows the number of cars by country
plot()

- `plot(cars$MPG)`
- MPG is numeric (continuous)
- Thus, index plot
- Cars are plotted by index
- MPG is the Y axis
- Useful?
plot()

- plot(cars$Weight, cars$MPG)
- Two continuous variables
- Thus, a scatter plot
- Intuitive result
• To clarify the relationship, we can use the log function
• `plot(log(cars$Weight), log(cars$MPG))`
plot()

- `cars$Cylinders <- as.factor(cars$Cylinders)`
- `plot(cars$Cylinders, cars$Country)`
- Both categorical
- Thus, stacked bar chart
- First element is x, second is y
hist()

- Plot is great for getting a quick visualization for various types of data
- hist() is more suited to showing distributions
- Bins values for frequency

- Making a sub-dataframe
- American_cars <- cars[which(cars$Country == "U.S."), ]
- which() returns a list of indices where the condition is true
hist()

- `hist(American_cars$Horsepower)`
- R chooses the bin number
hist()

- `hist(American_cars$Horsepower, breaks = 10)`
- `hist(American_cars$Horsepower, breaks = 3)`
- Setting breaks may increase/decrease granularity
Many more options

- barplot()
- boxplot()
- pairs()

- You can see more about these in documentation, or just try them out.

- For now, we will go into more detail on the basic plots
Customizing Plots

- It is also possible to fully customize the basic plots
  - Color, sizes, Axes, labels, etc.

- `plot(cars$Weight, cars$MPG)
- plot(cars$Weight, cars$MPG,
  - `xlab = "Weight",
  - `ylab = "Miles per Gallon",
  - `main = "MPG vs Car Weight",
  - `col = "orange"`
Customizing Plots

- `par()`
  - Sets session wide graphical parameters
- `par(col = "blue")`
- `plot(cars$Weight, cars$MPG)`
Customizing Plots

- `plot(cars$Weight, cars$MPG,`
  - `xlab = "Weight",
  - `ylab = "Miles per Gallon",
  - `main = "MPG vs Car Weight",
  - `col = "orange",
  - `col.main = "darkgray",
  - `cex.axis = 0.6,
  - `pch = 4)`
- `cex` sets text sizes
- `pch` sets point type
Basic Trendlines

- `plot(cars$Weight ~ cars$MPG, data = cars,`
  - `xlab = "Weight",`
  - `ylab = "Miles per Gallon",`
  - `main = "MPG vs Car Weight",`
  - `col = "orange",`
  - `col.main = "darkgray",`
  - `cex.axis = 0.6,`
  - `pch = 4)`

- `abline(lm(cars$Weight ~ cars$MPG))`
  - `lm()` generates a linear model (regression) of the data
  - `~` is an R operator for formulas
ggplot2

- ggplot is another popular library for plotting
- Based around dataframes (rather than vectors)
- Basic “default” styles look a lot nicer than build in plot style

- install.packages("ggplot2")
- library(ggplot2)
ggplot() basics

- car_plot = ggplot(data = cars)
  - Can create a plot and modify as a variable
- car_plot + geom_point(aes(x = Weight, y = MPG))
  - Once you add a geometry type, you get a plot
- aes() modifies plot aesthetics
- Can use for plot to change the whole figure
  - Or can use for separate geometries
  - One plot can have many geometries
  - Similar to plot + abline
Customizing ggplot

- `car_plot(cars, aes(x=Weight, y=MPG, )) +`
  - `geom_point( aes( color = Country)) +`
  - `geom_smooth()+`
  - `labs(title="MPG vs Weight",`
    - `y="Miles per Gallon")`

- Many more options
- Check the cheat sheet at: