Making Sense of Data

Kadambari Devarajan

http://kadambarid.in
kadambari.devarajan@gmail.com

Data Science with R Workshop for Engineering Undergraduates

Outline

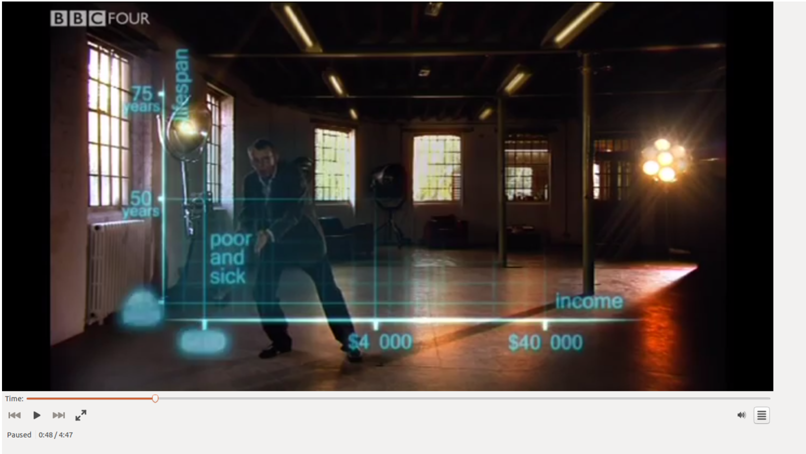
1. Data Science
   - Data, Datasets, Statistics
2. R + RStudio
   - The UI
   - R Basics
   - Visualization

Outline

Outline

1. Data Science
   - Data, Datasets, Statistics
2. R + RStudio
   - The UI
   - R Basics
   - Visualization

A World of Data



Hans Rosling's 200 Countries, 200 Years

A World of Data

Hans Rosling's River of Myths

Data

"Scientists seek to answer questions using rigorous methods and careful *observations*. These observations - collected from the likes of field notes, surveys, and experiments - form the backbone of a statistical investigation and are called **data**." - OpenIntro Statistics

Data

- **Shopping** - Amazon/Flipkart/BigBasket - when you buy something, when you don't; Recommendations
- **Food** - Zomato/Burrp; Recommendations, Suggestions, Popularity
- **Accommodation** - Housing, Hotels, Airbnb
- **Travel** - Makemytrip/Expedia/Goibibo/TripAdvisor; Redbus, IRCTC, Airlines; Maps, Traffic
- **Online** - Google, Facebook, Twitter, Blogs - when you say/post/tweet/like/delete/visit/search for something

Data

- **Shopping** - Amazon/Flipkart/BigBasket - when you buy something, when you don't; Recommendations
- **Food** - Zomato/Burrp; Recommendations, Suggestions, Popularity
- **Accommodation** - Housing, Hotels, Airbnb
- **Travel** - Makemytrip/Expedia/Goibibo/TripAdvisor; Redbus, IRCTC, Airlines; Maps, Traffic
- **Online** - Google, Facebook, Twitter, Blogs - when you say/post/tweet/like/delete/visit/search for something

Data

- **Entertainment** - IMDB/RottenTomatoes/YouTube, News
- **Science and Technology** - Publications, Books, Research, Journals, Experiments
- **Education** - EdX/Coursera/Udacity/KhanAcademy, Universities, Placements, Curriculum, Student/Staff info
- **Business and Industry** - Finance, Stock Market, Manufacturing; Sales
- **Agriculture** - Crop yields, Pesticide use, Sustainability, Organic farming
- **Health** - Hospitals, Diseases, Pharma, Medicines, Trials

Data

- **Entertainment** - IMDB/RottenTomatoes/YouTube, News
- **Science and Technology** - Publications, Books, Research, Journals, Experiments
- **Education** - EdX/Coursera/Udacity/KhanAcademy, Universities, Placements, Curriculum, Student/Staff info
- **Business and Industry** - Finance, Stock Market, Manufacturing; Sales
- **Agriculture** - Crop yields, Pesticide use, Sustainability, Organic farming
- **Health** - Hospitals, Diseases, Pharma, Medicines, Trials

Dataset

- A collection of related sets of information
- Composed of different elements, but can be operated as a single unit
- Contents of table or matrix of data, where every column denotes a variable, and each row corresponds an element or member of the collection of data

Data Science

Some combination of three related disciplines:

- **Data analysis** - Gathering, display, and summary of data
- **Probability** - Laws of chance
- **Statistical inference** - Science of drawing statistical conclusions from specific data using knowledge of probability

Data Science

Steps:

- Visualize/Analyze
- Infer
- Model
- Predict

Statistics

- The art and science of extracting meaning from data
  - Summarizing data
  - Visualizing data
  - Estimating and interpreting quantities
  - Making inferences

Statistics

- **Quantifying uncertainty** - “I am 95% confident that by the end of this class, between 82% and 87% of you will be able to make a plot!”

Vitalstatistix?!

Image Source: http://asterix.wikia.com/wiki/Vitalstatistix

- **-ve:** *Challenger Space Shuttle exploded in 1986, killing 7 astronauts.*
  - Decision to launch in cold weather, without analysis of performance data at low temp
- **+ve:** *Salk Polio Vaccine, trials on 4,00,00 children*
  - Strict controls against biased results + good statistical analysis $\rightarrow$ vaccine effectiveness + polio eradication

Exercise

Collect student weight in kg

- **Dot Plot**
- **Frequency Table** - Class Interval, Midpoint, Frequency, Relative Frequency
- **Histogram** - Bar graph where
  - each bar = interval
  - center = midpoint
  - bar height = no. of data points in interval
- **Stem-and-Leaf Diagram**

Class Intervals: Guidelines

- Intervals of equal length + midpoints at convenient round numbers
- Small data set $\rightarrow$ (Use) Few intervals
- Large data set $\rightarrow$ (Use) More intervals

Summary Statistics

Important properties of data/measurements:

- Central or typical value
- Spread around the value - wide, narrow

Summary Statistics: Center

An array or table of data: **Observation** (1, 2, ... n) and **Data Value** (x_1, x_2, ... x_n)

- **Mean** - average value; add all the data and divide by number of observations
  - $\bar{x} = (x_1 + x_2 + ... + x_n)/n$
- **Median** - midpoint of data after sorting in ascending order
  - Odd - 2 3 **9** 9 11
  - Even - 2 3 9 9 $\rightarrow$ (3+9)/2 = 6

Summary Statistics: Center

Why more than one measure of center?

- **Median** - not sensitive to outliers or extreme values
    - **Example:** No. of friends on Facebook
    - Data - 50, 50, 100, 100, 200
    - Median - 100; Mean - 100
    - If instead of 200, someone has 2000 friends:
    - Median - remains the same, while Mean = 460!

Summary Statistics: Spread

- Suppose all of you weigh exactly the same. What will the spread be?
- Suppose you had some wrestlers (Sakshi Malik, Phogat Sisters) and badminton players (PV Sindhu, Saina Nehwal) in your group. How will the histogram look then?

Summary Statistics: Spread

Interquartile Range:

- Order data numerically and find the median
- Divide the data into 2 groups at the median, say Highs and Lows
- Find the median of Lows (called the **first quartile** or **Q1**)
- Find the median of Highs (called the **third quartile** or **Q3**)
- Calculate the **interquartile range** using **IQR** = **Q3** - **Q1**

Summary Statistics: Spread

- In the group weight example, what does the IQR tell you?
  It gives the difference between a median heavy and median light student.

Summary Statistics: Spread

- In the group weight example, what does the IQR tell you?
  It gives the difference between a median heavy and median light student.

Summary Statistics: Spread

Box and Whiskers Plot

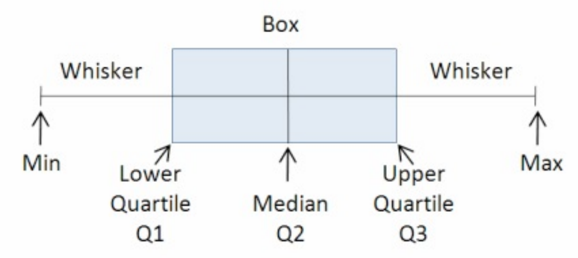
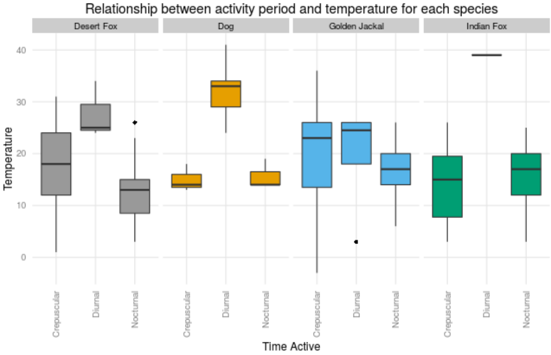


Image Source:
http://faculty.nps.edu/mjdixon/styled-11/styled-13/styled-18/files/pasted-graphic.jpg

Summary Statistics: Spread

Box and Whiskers Plot



- Outlier: Point more than 1.5 IQR from box ends
- Great for showing differences between groups

Summary Statistics: Standard Deviation

$$\text{Population Variance} = (\sigma x)^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2$$

$$\text{Sample Variance} = (Sx)^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2$$

Image Source: `https://mathbitsnotebook.com/Algebra1/StatisticsData/STSD.html`

- Measures spread from mean

Properties of Mean and SD

- Very good at summarizing symmetrical histograms without outliers
- For such bell-shaped data:
  - approx. 60% of the data is within 1 SD of the mean
  - approx. 95% of the data is within 2 SD of the mean

Hypothesis Testing

- Formulate hypotheses
  - Null - (usually) observations are a result of pure chance (i.e. random)
  - Alternate - observations are because of a real effect + chance variation
- Identify a test statistic
- p-value - If null hyp is true, then probability of observing a test statistic at least as extreme as that observed
- Compare p-value to a set significance level (if p-value <= alpha, then null hyp ruled out)

Hypothesis Testing

		Truth	
	Hypothesis Testing	**The Null Hypothesis Is True**	**The Alternative Hypothesis Is True**
Research	**The Null Hypothesis Is True**	Accurate	Type II Error
	The Alternative Hypothesis Is True	Type I Error	Accurate

Image Source: `http://cramster-image.s3.amazonaws.com/definitions/stat-1-img-1.png`

Hypothesis Testing

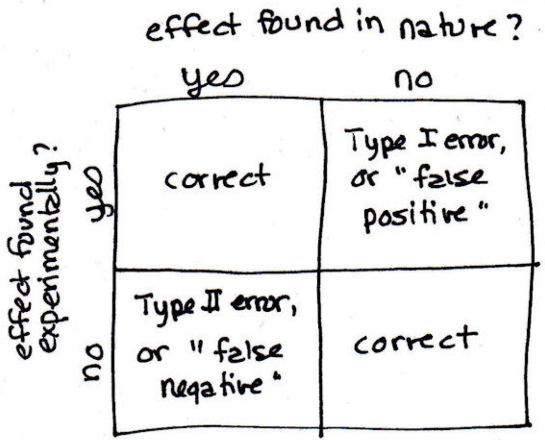


Image Source: `http://www.economistsdoitwithmodels.com/wp-content/uploads/2010/02/type-i-type-ii-error-1.jpg`

Sampling Design

- Quality vs. Quantity: both important in sampling
- Choosing a representative sample
- Types:
  - **Random** - Unbiased and Independent; Simple
  - **Stratified** - Divide population units into homogeneous groups and then draw SRS from each group
  - **Cluster** - Group the population into small clusters, draw SRS of clusters, and observe everything in sampled cluster
  - **Systematic** - Start with randomly chosen unit, then select every n-th unit

Sampling Exercise

- Pull 5 candies out of the bag
- Put the candies back in the bag!
- The candy weights are on the board
- Multiply the weights of the 5 candies you picked by 20
- Make a note of your estimate
- Tell me the weight of the bag
- The best estimate gets the full bag of candies :)

The Candy Exercise

- Let us tally the estimates made by every individual in the class
- Estimate Value : Tally
  - 880: ||||
- This is a dataset!

The Candy Exercise

- What are we studying?
  - Estimating weight of the bag.
- What are we sampling?
- What is our sample size?
- What type of sampling did we do?
- Were we able to estimate the weight of the bag accurately?
- Why or why not?
  - Sampling bias; Small candy tend to go to the bottom of the bag
  - Erroneous values
- What is the moral of this exercise?

The Candy Exercise

- What are we studying?
  - Estimating weight of the bag.
- What are we sampling?
- What is our sample size?
- What type of sampling did we do?
- Were we able to estimate the weight of the bag accurately?
- Why or why not?
  - Sampling bias; Small candy tend to go to the bottom of the bag
  - Erroneous values
- What is the moral of this exercise?

The Candy Exercise

- What are we studying?
  - Estimating weight of the bag.
- What are we sampling?
- What is our sample size?
- What type of sampling did we do?
- Were we able to estimate the weight of the bag accurately?
- Why or why not?
  - Sampling bias; Small candy tend to go to the bottom of the bag
  - Erroneous values
- What is the moral of this exercise?

The Candy Exercise

Using this data, make:

- List: Estimate, Tally, Total Count
- Dot Table
- Stem-and-leaf Plot
- Frequency Table - Class Interval, Midpoint, Frequency, Relative Frequency
- Histograms

The Candy Exercise: Quiz

Using the Candy dataset collected in class,

- Calculate the Mean, Median, and Mode.
- Calculate Q1, Q2, Q3, and IQR.
- Make a box-and-whiskers plot

Variables

Types:

- Numerical
  - Continuous
  - Discrete
- Categorical
  - Regular Categorical
  - Ordinal

Variables: Numerical

- Observations can take any value in a set of real numbers
- Can add, subtract, take averages
- Example:
  - **Discrete** - Numerical values with jumps; can take only certain number of values (finite or countably infinite)
    - No. of items bought at a market, Population counts, Census
  - **Continuous** - Opposite of discrete; infinite possible values
    - Height, Weight, Time, Government spending, Fluid measurements (milk, water)

Variables: Categorical

- Observations that form categories
- CanNOT add, subtract, take averages
- Example:
  - **Regular Categorical** - One or more categories, no order
    - States, Countries, Gender
  - **Ordinal** - Levels have a natural order
    - Economic status, Education

Distributions

Figure 6A.15: Distributional Choices

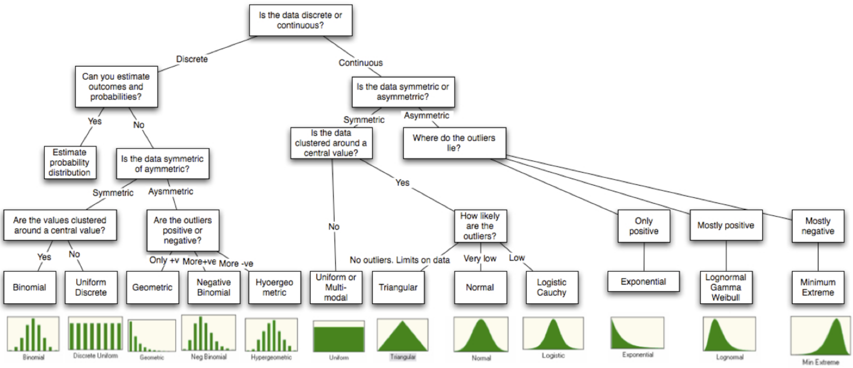


Image Source: `https://www.r-bloggers.com/fitting-distributions-with-r/`

Study Design

Studies can be classified into:

- **Observational** - The researcher studies a system, but does not influence the outcome.
- **Experimental** - The researcher influences the system, then observes what happens.

Observational Studies

Types:

- **Cross-sectional**: Census
- **Longitudinal**: Aging and health-related studies
- **Cohort**: HIV and cancer incidence

Experimental Studies

Types:

- **Controlled**: Drug trials
- **Natural**:
  - Cholera outbreak
  - Smoking ban
  - Nuclear weapons testing
  - Many studies in meteorology, astronomy, geology, and ecology
- **Field**:
  - Clinical trials
  - Product prototypes
  - Many studies in anthropology, ecology, social sciences, economics, pharmaceuticals

Experiments and Causality

- Is chocolate good for you?
- Does demonetization act as a deterrant to lavish weddings?
- What causes breast cancer?

What do these questions have in common?

Experiments and Causality

- All of them attempt to assign a cause to an effect!
- Data + Statistics can help answer such questions!

Lies, Damned Lies, and Statistics

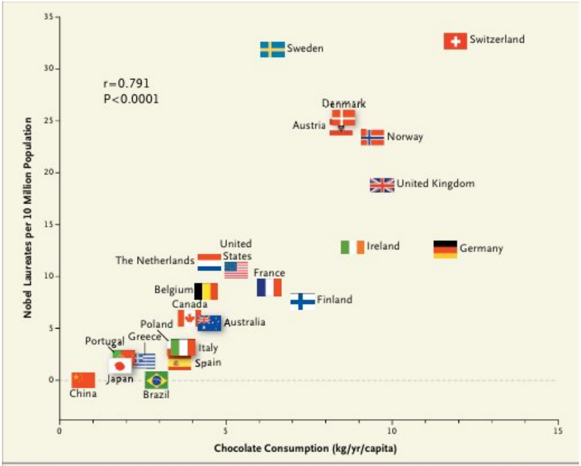


Figure 1. Correlation between Countries' Annual Per Capita Chocolate Consumption and the Number of Nobel Laureates per 10 Million Population.

Image Source: `http://www.nejm.org/doi/full/10.1056/NEJMon1211064`

Lies, Damned Lies, and Statistics

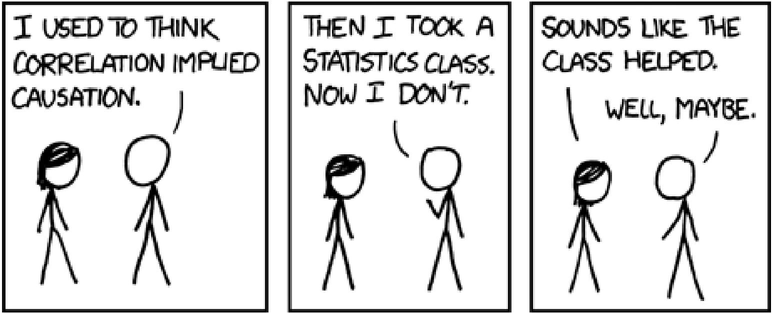


Image Source: `http://imgs.xkcd.com/comics/correlation.png`

Gotchas

- Never convert a categorical variable to a number and then use it in analysis!
- When trying to interpret or analyse data, always look for gaps!
- When trying to explain missing places/anomalies in the data - if it's not convenient, don't discard!
- Outliers matter!
- Treat data analysis like a detective story! The why's and how's matter!

Outline

1. Data Science
   - Data, Datasets, Statistics
2. R + RStudio
   - The UI
   - R Basics
   - Visualization

20000 feet view of R

- Programming language - not just a statistics package!
- Object-oriented
  - data/information stored as objects
  - operations on objects
- Flexible and powerful

Why R Rocks

- One of the most powerful environments for statistics, currently
  - Interactive
  - Data structures
  - Functions as objects
  - Missing data
- Command-line = Clarity!
- Avoiding the dangers of button-clicking

Why R Rocks

- Safety with scripts
- Pretty pictures - graphics and visualization
- Free (as in “free beer” AND “freedom”)
  - Packages
  - Community

Installation: Troubleshooting

In your terminal:

```
# Add R repository to
# /etc/apt/sources.list file
$ sudo echo
  "deb http://cran.rstudio.com/bin/
  linux/ubuntu xenial/" | sudo tee -a
  /etc/apt/sources.list
# Change 'ubuntu' and 'xenial' based on
# your Linux distribution and version
```

Installation: Troubleshooting

```
# Add R to Ubuntu keyring
$  gpg --keyserver keyserver.ubuntu.
com --recv-key E084DAB9
$ gpg -a --export E084DAB9 |
sudo apt-key add -
```

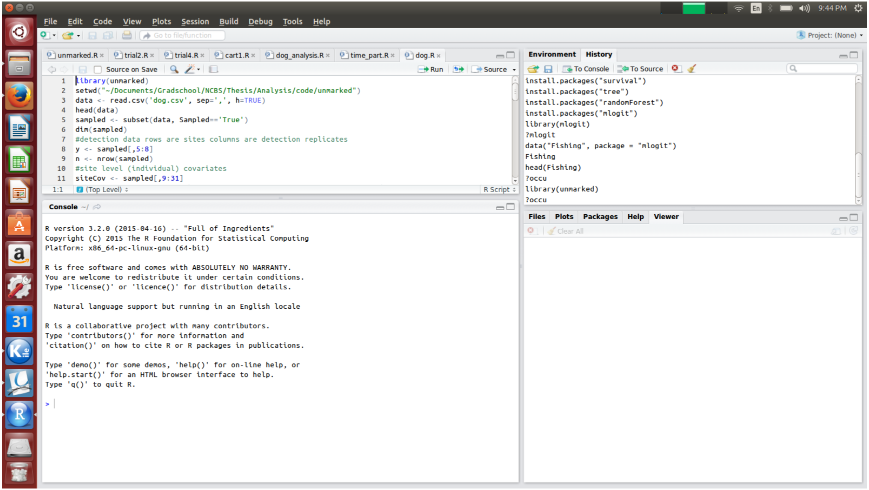
Installation: Troubleshooting

```
# Install R-base
$ sudo apt-get update
$ sudo apt-get install
r-base r-base-dev
```

Installation: Troubleshooting

```
# Install RStudio
$ sudo apt-get install gdebi-core
$ wget https://download1.rstudio.
org/rstudio-1.0.44-amd64.deb
$ sudo gdebi -n
rstudio-1.0.44-amd64.deb
$ rm rstudio-1.0.44-amd64.deb
```

RStudio


File Edit Code View Plots Session Build Debug Tools Help
Go to file/function
Project: (None)
unmarked.R trial2.R trial4.R cart1.R dog_analysis.R time_part.R dog.R
Source on Save
Run
Source
library(unmarked)
setwd("~/Documents/Gradschool/NCBS/Thesis/Analysis/code/unmarked")
data <- read.csv('dog.csv', sep=',', h=TRUE)
head(data)
sampled <- subset(data, Sampled=='True')
dim(sampled)
#detection data rows are sites columns are detection replicates
y <- sampled[,5:8]
n <- nrow(sampled)
#site level (individual) covariates
siteCov <- sampled[,9:31]
1:1
(Top Level)
R Script
Console ~/
R version 3.2.0 (2015-04-16) -- "Full of Ingredients"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
Environment History
To Console
To Source
install.packages("survival")
install.packages("tree")
install.packages("randomForest")
install.packages("mlogit")
library(mlogit)
?mlogit
data("Fishing", package = "mlogit")
Fishing
head(Fishing)
?occu
library(unmarked)
?occu
Files Plots Packages Help Viewer
Clear All
9:44 PM

Outline

1. Data Science
   - Data, Datasets, Statistics
2. R + RStudio
   - The UI
   - R Basics
   - Visualization

Getting to Know the UI

- Console
- Help
- File editing
- File browser
- Plots
- Menus

RStudio Basics

- Files - Open, Save
- Executing

If typing directly in the console, just pressing ‘Enter’ will suffice for the command to be executed. However, if typing in the source (recommended), ‘Ctrl-Enter’ will do the trick.

- Executing a block of commands
- Multiline commands and the ‘+’ symbol

Some Ground Rules

- Everyone must type along
- Any text following the command prompt (“>”) has to be typed into your Source or Console
- The output has not been given in the slides

Some Tips

- Navigation on console
  - Arrow keys
  - Tab completion
- Help
- Help using functions
  - ?- calls help file for builtin function
  - ?? - searches all builtin functions for the word

```
> ?sin
> ?mean
> ??mean
```

Outline

1. Data Science
   - Data, Datasets, Statistics

2. R + RStudio
   - The UI
   - R Basics
   - Visualization

Let's Get Rolling with R

Basic Arithmetic

```
> 23 + 79 # Evaluates expression
# and prints the result
```

- The '>' symbol is called the 'prompt'
- Anything following the '#' symbol is a 'comment'

Let's Get Rolling with R

Expressions

```
> 12/4 + 2 # Operator precedence
```

- 12/(4 + 2) is different from (12/4) + 2
- Use parantheses

R as a Calculator

Try these:

```
> 17 + 24
> 1.23456*42
> 47/6
> 4.567^54
> 2/4 + 1
> 2/(4 + 1)
```

R as a Scientific Calculator

Try these:

```
> sqrt(3)
> sin(pi/2)
> asin(0.5)
> asin(0.5)*180/pi
> log(2)
> log10(2)
```

Assignment

Assignment binds information to an object
"<-" is the assignment symbol
"=" can also be used

```
> x <- 5
# Assign 5 to the variable x
> y <- 4
> x
> x + y
> wt = 50
> val1 <- x - y
```

Assignment

```
> x + 4
> val1*30
> x^3
```

Objects

- Everything is an object
- Objects can be of different types
- Objects contain data
- We can perform operations on objects

Naming Objects

- Names must always start with a character (never a numeral)
- Names are case sensitive (wt is different from WT and wT)
- Names can be separated by an underscore (eg. female_wt) or a period (eg. female.wt)
  - Never use a space for separating compound names (eg. female wt is invalid)

Object Types

```
> wt <- 60.3
# Object type - Numeric
> x <- "hello"
# Object type - String
#(or Character)
> z <- TRUE
# Object type - Logical
#(TRUE or FALSE)
# Double precision float
```

Object Types

- Vectors
  - Simplest
  - Series of elements of a single data type
  - Similar to a column of values in a spreadsheet
- Matrices
- Data frames

Object Types: Vectors

Create using ‘concatenate’ - c(<comma separated list>)

```
> data <- c(1,4,3,2,1)
# c() stands for concatenate
# Values put into the same vector
> data*2
# Simultaneous operations - Useful
# functionality of vectors
# Operations on a vector are
# carried out one element at a time
> alphabet <- c("a","b","c","d")
# Vector of type ‘character’
```

Object Types: Vectors

Exercise:

```
> x <- 5
> x <- x + 1
> a <- c("x", "y", "z")
> a <- c(1, 2, "c")
> x <- c(1, 2, 3, 4)
> a[3]
> x[2]
> x[c(1, 3, 4)]
```

Find the type of an object:

```
> typeof(x)
```

Accessing Elements of a Vector

```
> b <- c("a", "b", "c", "d")
# Vector b of type 'character'
> b
> b[1]
# Value of the first element of b
> b[c(2,4)]
# Value of 2nd and 4th
# elements of b
> d <- b[-1]
# Assign all of vector b
# except the first element to d
```

Relational Operations

Operators - <, <=, >, >=, ==, !=

Try these:

```
> x <- 2
> x > 4
> x < 5
> a <- c(1, 2, 3, 4)
> a != 3
```

Logical Vectors

```
> a <- c(1,3,4,5)
> a[a<3]
```

This is the same as:

```
> a[c(TRUE, FALSE, FALSE, FALSE)]
```

Now try:

```
> which(a<3)
```

Exercise

- Create a vector ‘vec’ containing the values 10 through 60 in increments of 10
- Display the elements of ‘vec’
- Increase every element of the vector by 5 and assign these values to a new vector ‘vec1’

Exercise

- Display the elements of ‘vec1’
  - In how many ways can the 3rd and 5th elements of ‘vec1’ be displayed?
  - Display the values of the 3rd and 5th (of ‘vec’) using the methods discussed
- Display all elements of ‘vec1’ that are less than 35
  - less than and equal to 35

Logical Operations

Operators - !, &, |

```
> x <- c(1, 2, 3, 4, 5, 6)
> x > 3 & x < 6
```

- Exercise: Display elements of ‘vec1’ greater than 20 and less than and including 65

Do not confuse

- The relational operator $>$ and the command prompt $>$
- $<-$, =, and ==
  - Assign - Assignment Operator - $<-$ and =
  - Check/Verify - Relational Operator - ==

Do not confuse

- The different brackets used:
    - Parantheses - () - eg. functions
    - Square brackets - [] - eg. vector operations
    - Curly braces - {} - eg. expressions (*enclosing an expression that already uses parantheses*)
      Note - () and {} can be use interchangeably for most part

Object Types: Functions

- Functions have a name and a variable number of arguments
- Built-in functions
- User defined functions

```
> ?log
> log(x=100, base=10)
# The arguments x and base are
# passed to the function log()
> log(100,10)
# Arguments can be passed in right
# order without naming them
```

Object Types: Functions

Generating a sequence of numbers:

```
> seq(from=2, to=20, by=2)
# Function to generate regular
# sequence of nos.
```

Generating random numbers:

```
> runif(n=10)
# Default - random numbers
# from 0 to 1
> runif(n=100, min=0, max=100)
```

Summarizing Data

```
> a <- runif(n=100)
> b <- a[a<0.5]
> length(b)
# Count of no. of elements in b
> sum(b)
# Sum of the elements in vector b
> mean(b)
> sd(b)
> median(b)
> summary(b)
```

Plotting Data

```
> b_seq <- 1:length(b)
> plot(x=b_seq, y=b)
```

Matrices

```
> x <- matrix(c(5,7,9,6,3,4),nrow=3)
> y <- matrix(c(5,7,9,6), ncol=2)
> dim(x)
> x[1,1]
> x[2,]
> x[,2]
> x[2]?
> x%*%y
> t(x)
> solve(y)
```

Readily Available Data

R comes bundled with ready datasets that one can play around with.

```
# Load the package MASS
# (Modern Applied Statistics with S)
> library(MASS)

# List all the datasets in
# loaded packages
> data()
> data(iris)
# will load the dataset Iris
# into memory
```

Popular Built-in Datasets

- iris
- trees
- orange
- cars
- islands
- mtcars
- sleep
- titanic
- women

Built-in Datasets

```
> data()
> trees
# Also try ?trees
> summary(trees)
> head(trees)
> tail(trees)
```

This is a dataframe!

Accessing Elements of a Dataframe

```
> trees[1:5,]
> trees[,2]

> names(trees)
> trees$Girth
> trees$Volume
```

Accessing Elements: ‘attach’ function

Attach to a dataset:

```
> attach(trees)
> mean(Height)
> mean(Girth)

> detach(trees) # When finished
```

Accessing Elements: 'with' function

with can be conveniently used instead of **attach**

```
> plot(iris$Petal.Length ~
+   iris$Species)
> with(iris, plot(Petal.Length ~
+   Species)) # Same thing!
```

Notice, no need to **detach**

I recommend using 'with' instead of 'attach'!

Plotting Data

```
# Histogram
> hist(trees$Height)

# Boxplot
> attach(trees)
> boxplot(Height)

# Scatterplot
> plot(Height, Girth)
> detach(trees)
```

Exercise: Rewrite these to use ‘with’ instead of ‘attach’

Plotting Data

Multiple plots:

```
> attach(trees)

> par(mfrow=c(2,2))
> hist(Height); boxplot(Height)
> hist(Volume); boxplot(Volume)

> detach(trees)
> par(mfrow=c(1,1))
```

Exercise: Rewrite these to use ‘with’ instead of ‘attach’

Plotting Data

Other plots:

```
> barplot(1:10)
> ?pie
```

Categorical Data

Explore the **iris** dataset (as was shown for **trees**).

```
> iris$Species
> pie(iris$Species)

# Using 'table'
> table(iris$Species)
> pie(table(iris$Species))
```

Plotting Data: Using Formulae

- Very convenient with categorical data
- Use ~ to create a formula

```
> boxplot(iris$Petal.Length ~
+ iris$Species)
> plot(iris$Petal.Length ~
+ iris$Species) # Same thing!
```

```
> plot(iris$Petal.Length ~
+ iris$Species, col=c("red", "blue",
+ "green"))
```

Subsetting

Subsets of vectors/data frames

```
> subset(iris,
+ iris$Species=="setosa")
> subset(iris, Species=="setosa")
# Also works!
> subset(iris, Species="setosa")
# Wrong!
> subset(iris, select=
+ c(Petal.Width, Petal.Length))
# Check docs for more options
```

Creating Your Own Dataframes

```
> x <- 1:20
> y <- x*x
> z <- y + 10
> df <- data.frame(x=x, y=y, z=z)
> df
> df <- data.frame(a=x, b=y, c=z)
# Changes name of column
> names(df)
```

Add/Remove Columns

```
> df$total <- df$x + df$y
# Error!
> df$total <- df$a + df$b
# Adds a column called 'total'
> names(df)
# To remove this column:
> df$total <- NULL
> names(df)
```

Add/Remove Rows

```
> rbind(df, c(-1, -2, -3))
> df
# Also check the cbind function
```

Using ifelse

```
> x <- 1:10
> ifelse(x < 6, "blue", "green")
> ifelse(x < 4, "blue",
+ ifelse(x < 7, "green", "red"))
# Color values in scatterplot
> with(iris, plot(
+     Petal.Length, Petal.Width,
+     col=ifelse(
+         Species=="setosa", "red",
+           ifelse(
+           Species=="virginica",
+           "blue", "green"))))
```

Missing Values

- Indicated by NA
- Typically automatically handled
- Use **is.na** to find the NA values

```
> x <- 1:5; y <- x*x
> plot(x, y)
> x[3] <- NA
> is.na(x)
[1] FALSE FALSE  TRUE FALSE FALSE
> plot(x, y)
```

Type-along Exercise

```
> wt <- c(69, 73, 70, 69, 90, 48,48)
> mean(wt)
> summary(wt)
> plot(wt)
> hist(wt*20)
> hist(wt, breaks=4)
> hist(wt*20, breaks=7,
+  xlim=c(500,2500), col="blue")
# Vertical line on existing graph
> abline(v=930)
```

R Scripts

- Use RStudio to create the code
- Save it to filename.R
- Run it directly using menus/shortcut
- Run it on console using:

```
> source("file.R")
```

Working Directory

Set working directory

- If R can't find relative files

```
> getwd()
> setwd()
> setwd("~/Documents/Gradschool
+ /Analysis/code/unmarked")
```

- Using the UI
  - Menu: Session ⇒ Set Working Directory
  - Shortcut: Ctrl+Shift+H

Reading a .csv file

Reading CSV data

```
> read.csv("file.csv")
> read.csv("http://www.ats.ucla.
edu/stat/data/hsb2.csv")
> data <- read.csv('pop.csv',
sep=',', h=TRUE)
```

Explore the data, summarize and visualize.

Writing a .csv file

```
#Create a data frame
> data <- read.table(header=TRUE,
text='
subject sex size
      1   M    7
      2   F   NA
      3   F    9
      4   M   11
')
```

Writing a .csv file

```
# Write to a file, suppress row names
> write.csv(data, "data.csv", row.names=FALSE)

# Same, except that instead of "NA", output
#blank cells
> write.csv(data, "data.csv",
row.names=FALSE, na="")

# Use tabs, suppress row names and column names
> write.table(data, "data.csv", sep="\t",
row.names=FALSE, col.names=FALSE)
```

The Candy Exercise: Sampling

- Create another dataset
- **Large**, **Small**, **Bag_Wt**, where:
  - **Large** - No. of large candy
  - **Small** - No. of small candy
  - **Bag_Wt** - Estimated bag weight
- Add columns for total weight of large candy, total weight of small candy - every individual
- Add a column for the total weight of the bag
- **Tot_Candy**, **Wt_Large**, **Wt_Small**
- Formula: Weight of the bag: [(x*L)+((5-x)*S)]*20
- **Tot_Wt = [(Large*Wt_Large) + (Small*Wt_Small)]*20**

Loops in R

```
# Counter
> for(i in 1:10) {
# Task for each iteration
print(i)
}
```

Loops in R

```
> for(movie.actor in
c("Rajnikanth","Kamal Haasan",
"Shahrukh Khan","Aamir Khan",
"Nagarjuna","Venkatesh")) {
cat(sprintf("Hello %s...\n",
movie.actor))
}
```

Loops in R

```
> x <- 1:10
> y <- rep(NA,length(x))
> for(i in 1:length(x)) {
  y[i] <- x[i] ^2
  }
> print(y)
```

Working with Lists

```
> str(iris)
> head(iris)
> sp.l <- split(iris,iris$Species)
> str(sp.l)
> for(sp in sp.l) {
  print(head(sp))
  }
```

Working with Lists

Calculate the statistics of each species:

```
> res <- list()
> for(n in names(sp.l)){
 dat <- sp.l[[n]]
 #Extract the data from the list
 res[[n]] <- data.frame(species=n,
                        mean.sepal.length=
                        mean(dat$Sepal.Length),
                        sd.sepal.length=
                        sd(dat$Sepal.Length),
                        n.samples=
                        nrow(dat))
}
> print(res)
```

Working with Lists

Converting the list obtained into a data frame:

```
> res.df <- do.call(rbind,res)
> print(res.df)
```

lapply()

lapply() takes 2 arguments - a list and a function, and returns a list.

```
# Continuing with the Iris dataset
> lapply(sp.l,nrow)
# #Arguments can also be
# provided to the function
> lapply(sp.l,head,n=2)
```

sapply()

lapply() returns a list, sapply() *simplifies* the list to a vector.

```
> sapply(sp.l,nrow)
```

sapply()

Doesn't always work the way you expect!

```
> res <- sapply(sp.l,function(dat){
r <- data.frame(mean.sepal.length=
                mean(dat$Sepal.Length),
                sd.sepal.length=
                sd(dat$Sepal.Length),
                n.samples=nrow(dat))
return(r)
})
> print(res)
```

tapply()

Takes care of subsetting too.

```
> tapply(iris$Sepal.Length,
iris$Species,mean)
```

tapply()

Does in one line, while sapply() takes many!

```
> sp.l <- split(iris,iris$Species)
> res <- sapply(sp.l, function(d){
 mean(d$Sepal.Length)
 })
> print(res)
```

tapply()

Can only work on a single vector at a time. Bypassed by:

```
> traits <- data.frame(sepal.len.mean=
tapply(iris$Sepal.Length,iris$Species,
mean),sepal.len.sd=
tapply(iris$Sepal.Length,
iris$Species,sd),
n.obs=tapply(iris$Sepal.Length,
iris$Species,length))
> print(traits)
```

replicate()

Repeats an expression many times. Useful for bootstrapping or sampling from distributions.

```
> reps <- replicate(1e5,
max(rnorm(100)))
> hist(reps,breaks=100)
```

Bootstrapping

Steps:

- Resample a given data set a specified number of times
- Calculate a specific statistic from each sample
- Find the SD of the distribution of that statistic

```
# sample(x, size, replace, prob)
```

User-defined Function for Bootstrapping

```
# Function to bootstrap the
# standard error of the median
> b.median <- function(data, num) {
  resamples <- lapply(1:num,
  function(i) sample(data, replace=T))
  r.median <- sapply(resamples,
  median)
  std.err <- sqrt(var(r.median))
  list(std.err=std.err,
  resamples=resamples, medians=
  r.median)     }
```

User-defined Function for Bootstrapping

```
# Generate the data to be used
> data1 <- round(rnorm(100, 5, 3))

# Save the results of the
# function b.median in the object b1
> b1 <- b.median(data1, 30)

# Display the first of the 30
# bootstrap samples
> b1$resamples[1]
```

User-defined Function for Bootstrapping

```
# Display the standard error
> b1$std.err

# Display the histogram of
# the distribution of medians
> hist(b1$medians)
```

User-defined Function for Bootstrapping

```
# Display standard error
# in one loc
> b.median(rnorm(100, 5, 2),
50)$std.err

# Display the histogram of the
# distribution of medians
> hist(b.median(rnorm(100, 5, 2),
50)$medians)
```

Outline

1. Data Science
   - Data, Datasets, Statistics
2. R + RStudio
   - The UI
   - R Basics
   - Visualization

Plots Great and Small

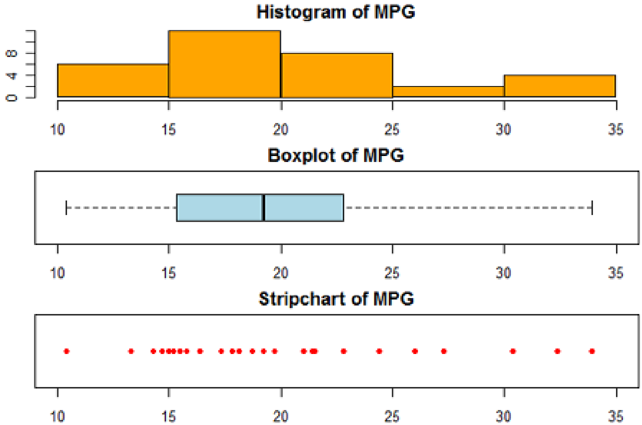


Image Source: `http://www.statmethods.net/graphs/`

Advanced Plots

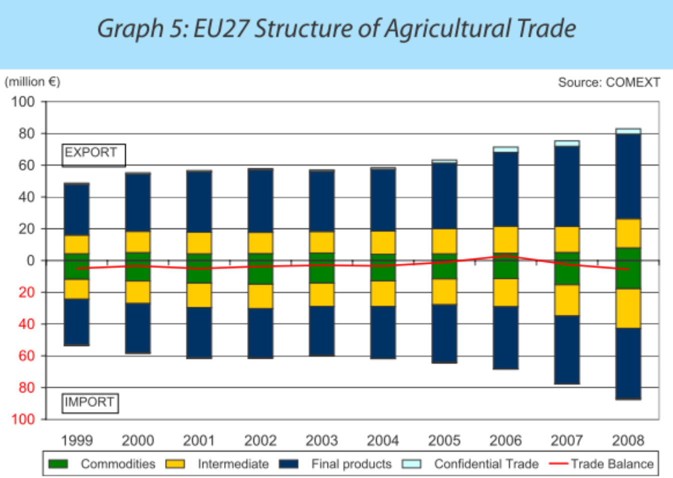
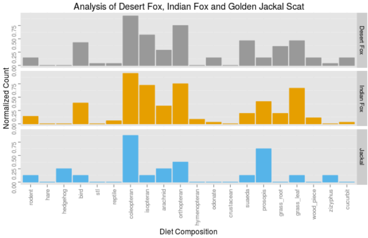
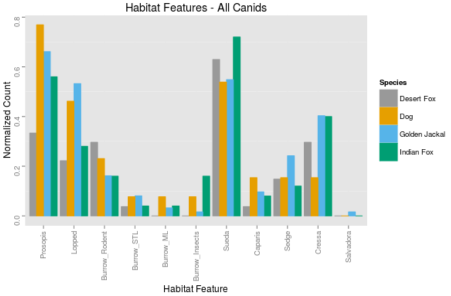


Image Source: `https://learnr.wordpress.com/`

Advanced Plots

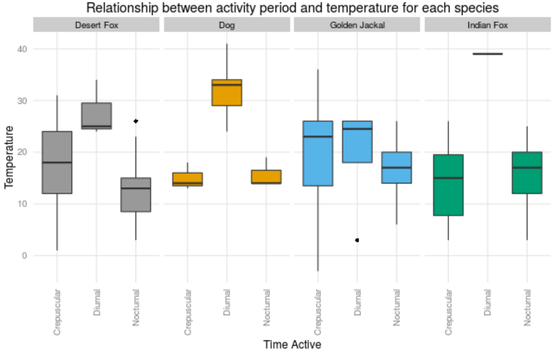
ggplot2 - Grammar of Graphics





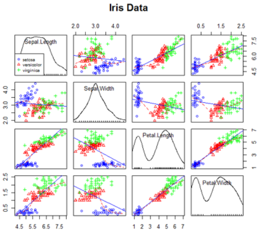
Advanced Plots

ggplot2 - Grammar of Graphics



Advanced Plots

- 3D plots
- Interactive graphs



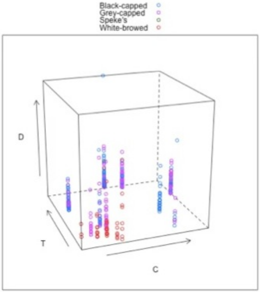
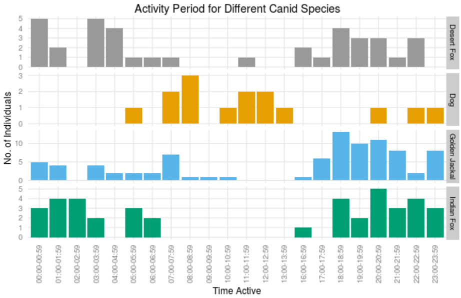
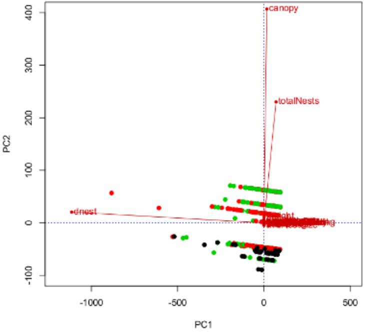


Image Source 1: `http://www.statmethods.net/advgraphs/index.html`
Image Source 3: `https://learnr.wordpress.com/2010/08/16/consultants-chart-in-ggplot2/`

Advanced Analysis





Packages in R

- Installing packages
- The Comprehensive R Archive Network - CRAN
  - `https://cran.r-project.org/`
- Some important packages:
  - **Data wrangling, data analysis**: dplyr, plyr, tidyr, reshape2, janitor
  - **Data import, web scraping**: rvest, scrapeR
  - **Data visualization**: ggplot2, shiny
  - **Engineering and Science**: randomForest, tree
  - **Social science**: demography, survey, sampling

Installing Packages

- Through UI
- Through command-line
  - With root access on Linux:

```
# install.packages("packagename")
> install.packages("ggplot2")
> library(ggplot2)
```

Installing Packages

- Without root access on Linux:

```
# Create a directory to install
# packages
> install.packages("ggplot2",
lib="/data/RPackages/")
# Where /data/RPackages is an
# example directory
> library(ggplot2,
lib.loc="/data/Rpackages/")
```

Installing Packages

- To avoid typing /data/RPackages everytime:
  - Create file .Renviron in your home area
  - Add the line R_LIBS=/data/Rpackages/ to it
- Setting repository:
  - Create a file .Rprofile in your home area
  - Add these lines to it:

```
cat(".Rprofile: Setting UK
repository")
r = getOption("repos") # hard
# code the UK repo for CRAN
r["CRAN"] =
"http://cran.uk.r-project.org"
options(repos = r)
rm(r)
```

Scatterplots with qplot

```
> install.packages(ggplot2)
> library(ggplot2)

> head(iris)
# By default, head displays the first
# 6 rows
> head(iris, n = 10)
# We can also explicitly set the
# number of rows to display
```

Scatterplots with qplot

```
> qplot(Sepal.Length, Petal.Length,
data = iris)

# Plot Sepal.Length vs. Petal.Length,
# using data from the `iris`
# data frame.

> qplot(Sepal.Length, Petal.Length,
data = iris, color = Species)
```

Scatterplots with qplot

```
> qplot(Sepal.Length, Petal.Length,
data = iris, color = Species,
size = Petal.Width)

# We see that Iris setosa flowers
# have the narrowest petals.
```

Scatterplots with qplot

```
> qplot(Sepal.Length, Petal.Length,
data = iris, color = Species,
size = Petal.Width, alpha = I(0.7))

# By setting the alpha of each
# point to 0.7, we reduce the
# effects of overplotting.
```

Scatterplots with qplot

```
> qplot(Sepal.Length, Petal.Length,
data = iris, color = Species,
xlab = "Sepal Length",
ylab = "Petal Length",
main = "Sepal vs. Petal Length in
Fisher's Iris data")
```

Line charts

```
> qplot(Sepal.Length,
Petal.Length, data = iris,
geom = "line", color = Species)
 # Using a line geom doesn't
 # really make sense here.
```

Line charts

```
# 'Orange' is another built-in
# data frame that describes
# the growth of orange trees.
> qplot(age, circumference,
data = Orange, geom = "line",
colour = Tree, main = "How does
orange tree circumference
vary with age?")
```

Line charts

```
# We can also plot
# both points and lines.
> qplot(age, circumference,
data = Orange,
geom = c("point", "line"),
colour = Tree)
```

Regression

- Statistical tool to establish relationship between two (or more) variables
  - Predictor variable - Value obtained through experiments
  - Response variable - Value derived from predictor variable

Regression

Of many kinds:

- Linear - Simple, Multiple
- Logistic
- Polynomial
- Ridge
- ...

Linear Regression

- Two (or more) variables related through an equation
- Exponent of both variables is 1
- Linear relationship is mathematically represented by a straight line when plotted as a graph (in 2D; changes for more dimensions)
- Non-linear relation - exponent of any variable not 1 - curve

Linear Regression

- y = ax + b
  - y - response variable
  - x - predictor variable
  - a and b - constants called coefficients

Example of Regression

Predicting weight of a person given her height

- Experiment - sample of observed values of height and corresponding weight
- Create a relationship model using lm() function in R
- Find the coefficients from the model
- Create a mathematical equation using the coefficients
- Get summary of the relationship model to know the average error in prediction (called Residuals).
- Predict the weight of new individuals using the predict() function in R

lm() function in R

Basic syntax is: lm(formula,data)

- formula - symbol presenting relation between x and y
- data - vector on which formula will be applied

Example of Regression

```
> x <- c(151, 174, 138, 186, 128, 136,
179, 163, 152, 131)
> y <- c(63, 81, 56, 91, 47, 57,
76, 72, 62, 48)

# Apply the lm() function.
> relation <- lm(y~x)

> print(relation)
> print(summary(relation))
```

predict() function in R

Basic syntax is: predict(object, newdata)

- object - formula that was created using lm()
- newdata - vector with new value for predictor variable

Example of Regression

```
# Find weight of a person with
# height 170
> a <- data.frame(x = 170)
> result <-  predict(relation,a)
> print(result)
```

Example of Regression

Visualizing the regression graphically:

```
# Give the chart file a name
> png(file = "linear_regression.png")

# Plot the chart
> plot(x,y,col = "red",main =
"Height & Weight - Regression",
 abline(lm(y~x)),cex = 1.3,pch = 16,
 xlab = "Height in cm",
 ylab = "Weight in kg")

# Save the file.
> dev.off()
```

Class Exercise

- Collect gender, height and weight data in class
- Create a spreadsheet
- Save as a .csv file
- Run a linear regression on this dataset
- Predict the weight for height = 160 cm

Resources

- StackOverflow
- https://cran.r-project.org/manuals.html
- https://www.r-project.org/other-docs.html
- Blog aggregator for posts about R: http://www.r-bloggers.com/
- Courses on EdX, Coursera, Udacity ...

Resources

- R basics: `http://cran.r-project.org/doc/contrib/usingR.pdf`
- Quick-R: `http://www.statmethods.net/`
- Interactive introduction to R: `https://www.datacamp.com/courses/introduction-to-r`
- In-browser learning of R: `http://tryr.codeschool.com/`

Resources

- Data Mining tool in R: `http://rattle.togaware.com/`
- Online e-book for Data Mining with R: `http://www.liaad.up.pt/~ltorgo/DataMiningWithR/`
- Introduction to the Text Mining package in R: `http://cran.r-project.org/web/packages/tm/vignettes/tm.pdf`

Interesting Resources

- https://deedy.quora.com/Hacking-into-the-Indian-Education-System
- https://www.r-bloggers.com/datasets-to-practice-your-data-mining/
- UC Irvine Machine Learning Repository: http://archive.ics.uci.edu/ml/