Section 1<

- **Online Course Layout:**
  - **Section 1 – Big Data and R**
    - **Class overview**
    - **Big Data Overview**
    - **Installing R**
    - **The IDE**
    - **Researching and best practices**
    - **R help**
    - **R Packages**
    - **Review Exercise 1**
    - **Hands on exercise**
  - **Section 2 – Data Wrangling**
  - **Section 3 – Data Visualization**
  - **Section 4 – R Markdown**
  - **Section 5 – Exploratory Data Analysis**
    - **Diamond Exercise**
    - **Bank Marketing Exercise**
  - **Section 6 – Introduction to Regression**
    - **Car Make Regression**
    - **Orange Juice Exercise**
  - **Section 7 – Introduction to Machine Learning**
    - **Titanic Kaggle Competition**
  - **Section 8 – Strategy**
    - **Big box store competitors**

https://stat.ethz.ch/R-manual/R-devel/library/base/html/Memory-limits.html

Background<

- 10+ years industry experience
  - Banking / Finance / Media / Tech
  - Cyber Security
  - Analytics
- MSBA ‘15 Stern School of Business NYU
  - Capstone: Predicted churn for members of a non-profit Arboretum

❑ DataKind

✓ Predictive Analytics to personalize Health Care

✓ 2016 Bloomberg’s Data for Good Exchange

getwd()<

- ✓ **I think, therefore I am…**

Agenda for Session 1

- **BIG DATA**
- **Industry leaders in Big Data**
- **Data Science**
- **Roles in Data Science**
- **Data Science Use Cases**
- **Data'isms**
- **Class Format**
- **R Background**
- **R Intro**
- **R Studio**

Big Data <

Big Data Leaders & Disrupters<

Company	Theme	Initiative	Launch Date
Google	Payments	Google Wallet app	2011
	Payments	Google Compare	2012
	Payments	Android Pay	2015
	Payments	Apple Wallet (previously Passbook)	2012
	Payments	Apple Pay	2015
	Payments	Messenger Payments	2015
amazon	Lending	Amazon SMB Lending	2012
	Digital Currency	Amazon Coins	2014
	Lending	Amazon Store Card & Card Comparison	2015
	Insurance	Amazon Protect (Product insurance)	2016
	Payments	Amazon Payments	2017
	Payments	Check-out by Amazon (B2B e-commerce solution)	NC

https://drive.google.com/file/d/0B18Jj7sicGWDY1VjMGM3clpNRk0/view

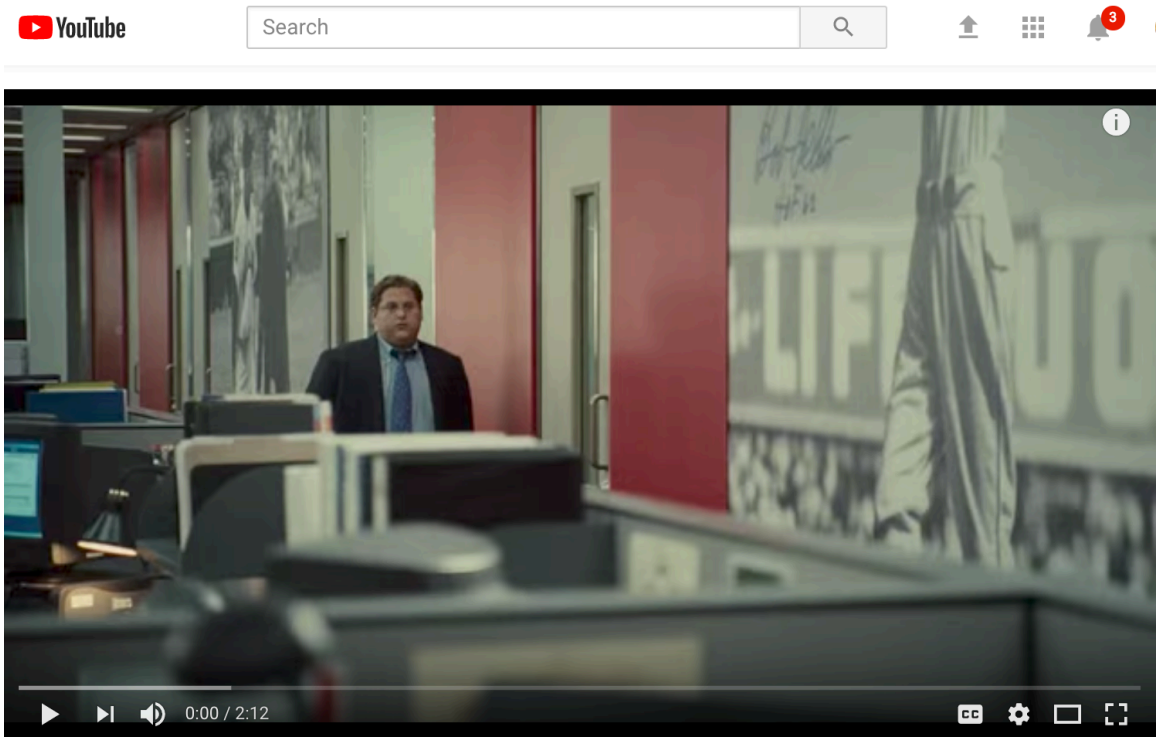
Data Science <

Data Science is the study of the generalizable extraction of knowledge from data

Business Analyst to Data Scientist <

What’s next for data science? <


MONEYBALL
Moneyball: The Art of Winning an Unfair Game is a book by Michael Lewis, published in 2003, about the Oakland Athletics baseball team and its general manager Billy Beane. Its focus is the team's analytical, evidence-based, sabermetric approach to assembling a competitive baseball team, despite Oakland's disadvantaged revenue situation. A film based on the book starring Brad Pitt and Jonah Hill was released in 2011.



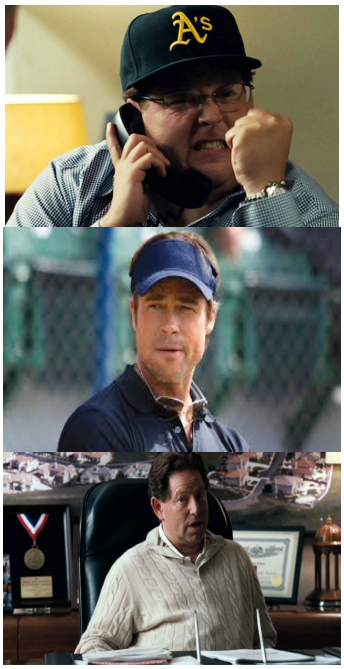
https://www.youtube.com/watch?v=TpBcwGOvO80

Roles in Data Science <

- "Data Scientist"
  - –can do the actual modeling
  - –applied statistician X computer scientist
- Collaborator in a data-centric project
  - –can transform from business problem to execution

- Manager in a data-centric company
  - –Ability to envision opportunities
  - –Ability to evaluate proposals
  - –Ability to evaluate execution
  - –Ability to interface with a broad variety of people

- Strategist, Investor, …
  - –Can envision opportunities, come up with novel ideas, design data science projects/companies conceptually, can evaluate the promise of new ideas

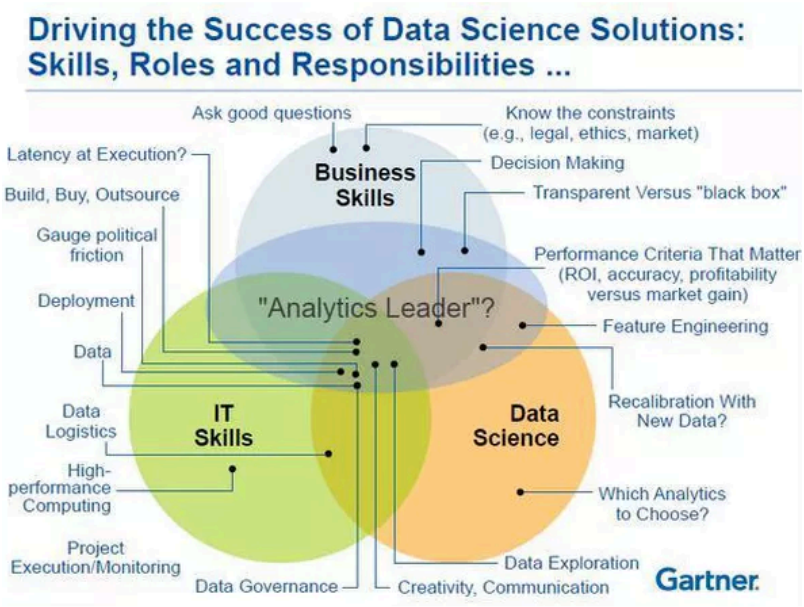


What's the secret sauce to a Data Scientist?<

Science + hacker
+ Business Knowledge
+ PATIENCE^10 =

DATA SCIENTIST

Responsibilities of a Data Scientist <

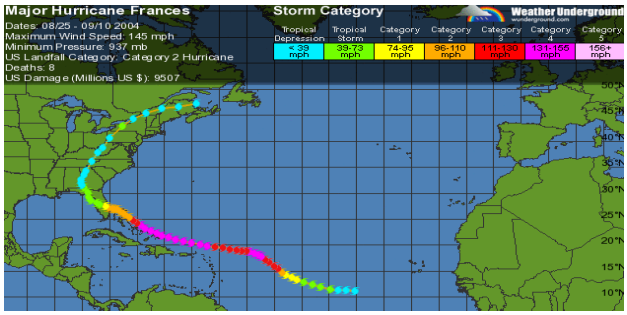


Business use cases for data science?<

- **Marketing**
  - Targeted marketing
  - online advertising
  - Recommendations for cross-selling
- **Finance**
  - Credit scoring
  - Trading
  - Fraud
  - Workforce management
  - Minimize operational expenses
- **Retail**
  - Supply chain management
- **Others????**

Business use case 1 – Walmart <

Winds: 145 mph
Date: August 24, 2004 – September 10, 2004



Walmart
Save money. Live better.

- **How can Walmart use data science to prepare the effected areas from the hurricane?**

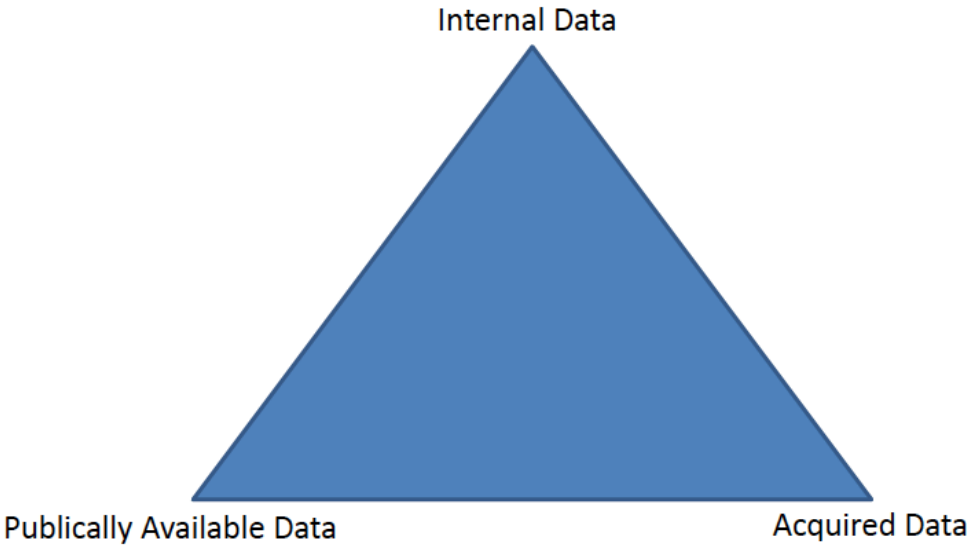
http://www.nytimes.com/2004/11/14/business/yourmoney/what-walmart-knows-about-customers-habits.html?_r=0

Business use case 2 – Target <

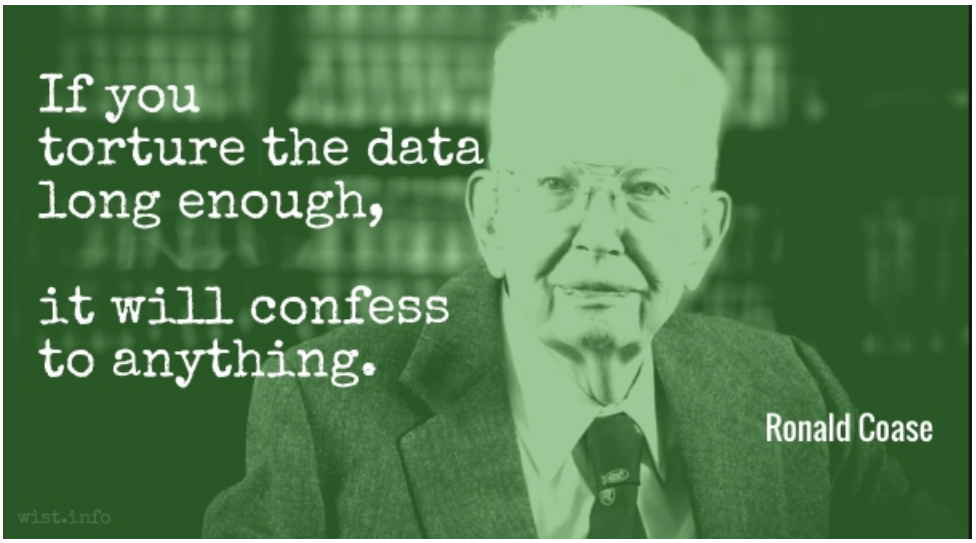
http://www.nytimes.com/2012/02/19/magazine/shopping-habits.html

- How can Target use data science to identify pregnant customers?
- Is there a risk of predicting *pregnant* customer?

The trifecta of data and knowledge extraction <



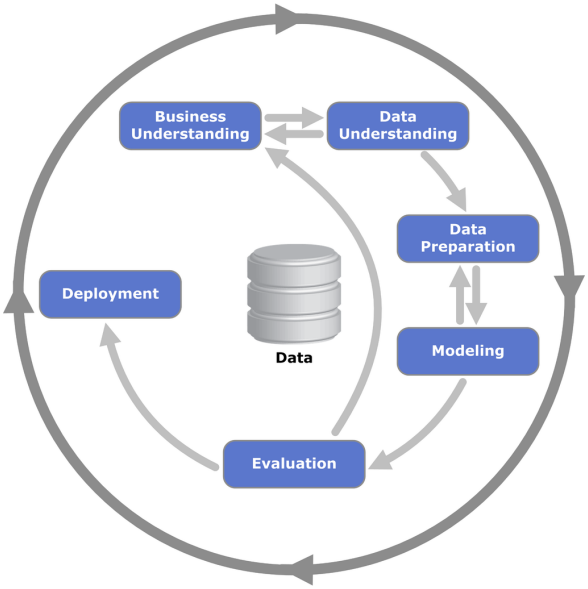
Data'isms <


If you
torture the data
long enough,
it will confess
to anything.
Ronald Coase
wist.info

How to start and finish a data science problem<

- Start with a data science problem
- Then finish with an answer to the data science problem
- Data handling is *not* the first step
- Consult a domain expert

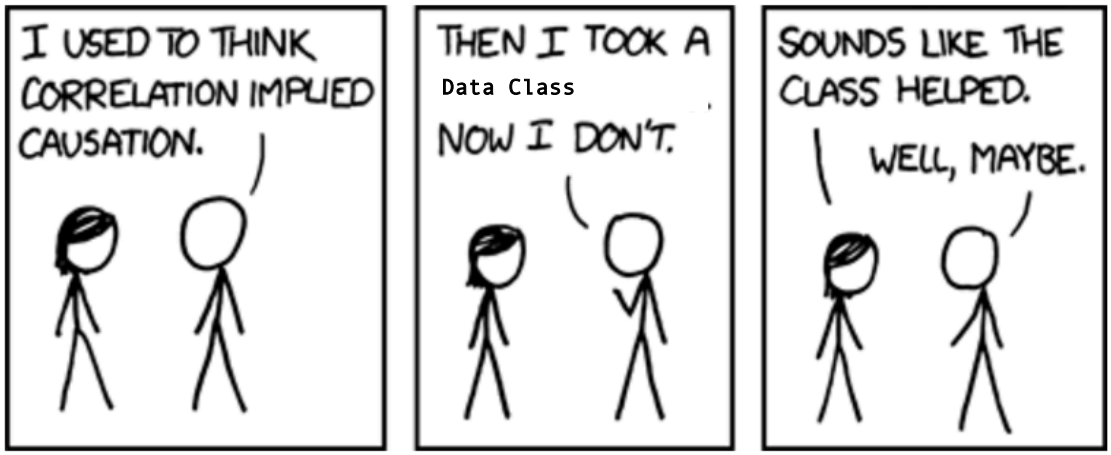
CRISP <


Business Understanding
Data Understanding
Data Preparation
Modeling
Evaluation
Deployment
Data

Correlation vs correlation <

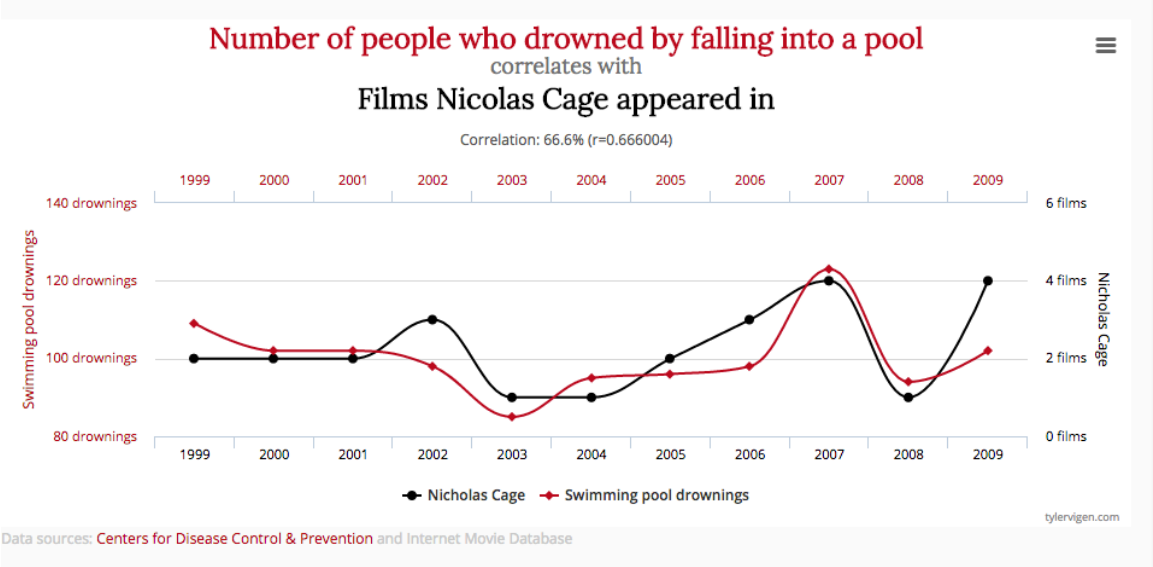
Correlation between two variables does **not** imply that one causes the other.

Correlation does not imply correlation <



One must always be wary when drawing conclusions from data! Randall Munroe, CC BY-NC

Correlation does not imply correlation Example <



R for Data Science <

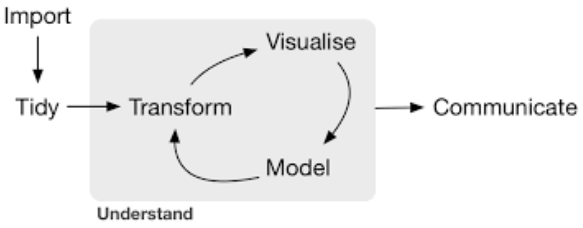
The goal for this class is to empower you to turn raw data into understanding, insight, and knowledge.

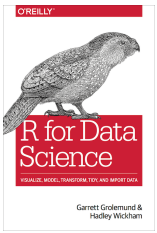
R is simply a tool that will allow you to do data science

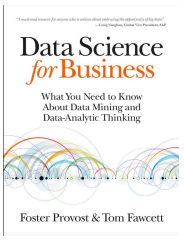
KNOWLEDGE IS POWER

Format going forward<

- **The format of the course onward will be broken out by the typical data analysis process explained by Hadley Wickham, author of "R for Data Science"**
- **Each data "problem" should be answered in this model**
- **80% of the heavy lifting in this model will be within the following sections Import->Tidy-> Transform**







http://r4ds.had.co.nz/intro.html

Import <

- **Take the data store in a file, data frame or web api and load it into a data.frame in R.**

- **If you cant get the data into R, then you cant do data science**

Tidy <

- **Cleaning the data and storing it in a consistent form that matches the semantics of the dataset with the way its stored**

Transform <

- **Narrow in on observations of interest, creating new variables that provide calculations or summary statistics**

Visualization <

- **Require human interpretation and are fundamental in describing data**

Models <

- At the surface they are a mathematical or computational tool
- Scaling the model can be difficult

Communication <

- **The ability to articulate and provide value to the end product of your data analysis**

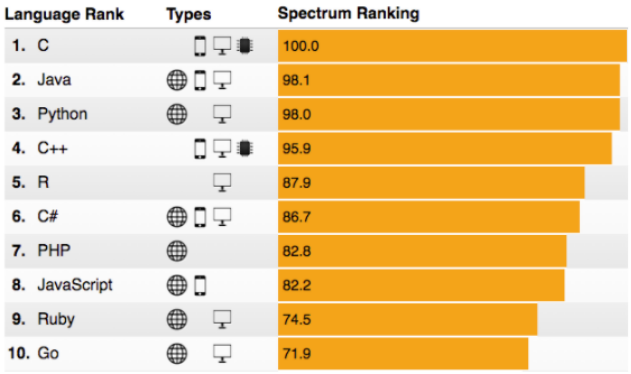
Not Data Science <

- **Data Warehousing – collection and storage of data**
- **Querying – SQL and any GUI clicking**
- **Statistical Analysis – hypothesis generation or testing. “A” “B” testing**
- **No Microsoft Excel here**

> R

History<

- The R statistical programming language is a free open source package based on the S language developed by Bell Labs. Evolved to R in 1997 with source code and packages available on CRAN and is a GNU project
- **R is** interpreted computer language used for **data manipulation, statistics, and graphics.**
- **In April of 2015 Microsoft acquired Revolution Analytics, the commercial provider of software and services for the R programming language**



R ⬆ 1 spot from 2015

http://www.datasciencecentral.com/profiles/blogs/r-moves-up-to-5th-place-in-ieee-language-rankings

R explained <

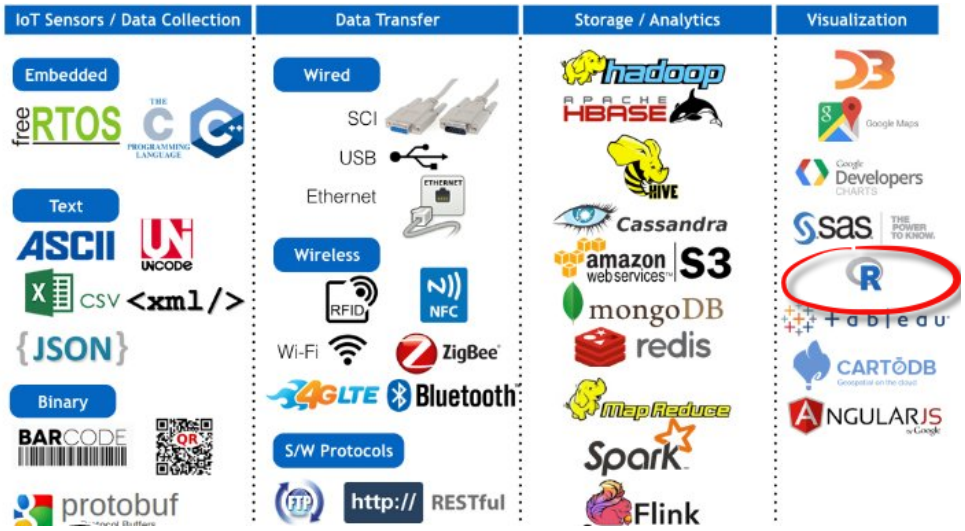
- R stores all objects in memory
- R can process 8TB of RAM on 64 bit operating systems
- R Rules of the Road
  - Can process up to 1 million records with ease
  - For > 1 million && < 1 billion records, R may struggle
  - > than 1 billion records, its best to use big data platforms that leverage map reduce and processed on big data platforms like Hadoop

https://stat.ethz.ch/R-manual/R-devel/library/base/html/Memory-limits.html

R explained part duex

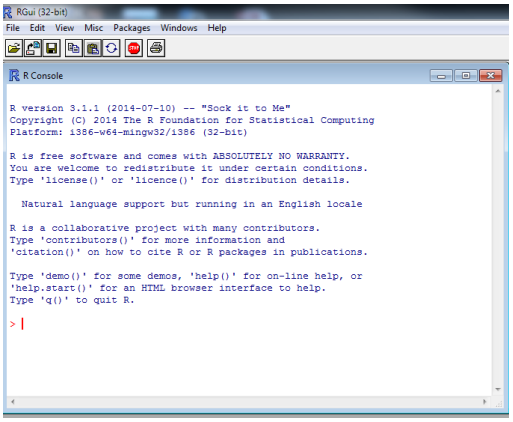
https://stat.ethz.ch/R-manual/R-devel/library/base/html/Memory-limits.html

Gartner IoT to Big Data Analytics Open Source Tech Stack



R Studio<

- For instructional purposes we will be using R Studio IDE
- Once the underlying R software is installed, RStudio can then be installed



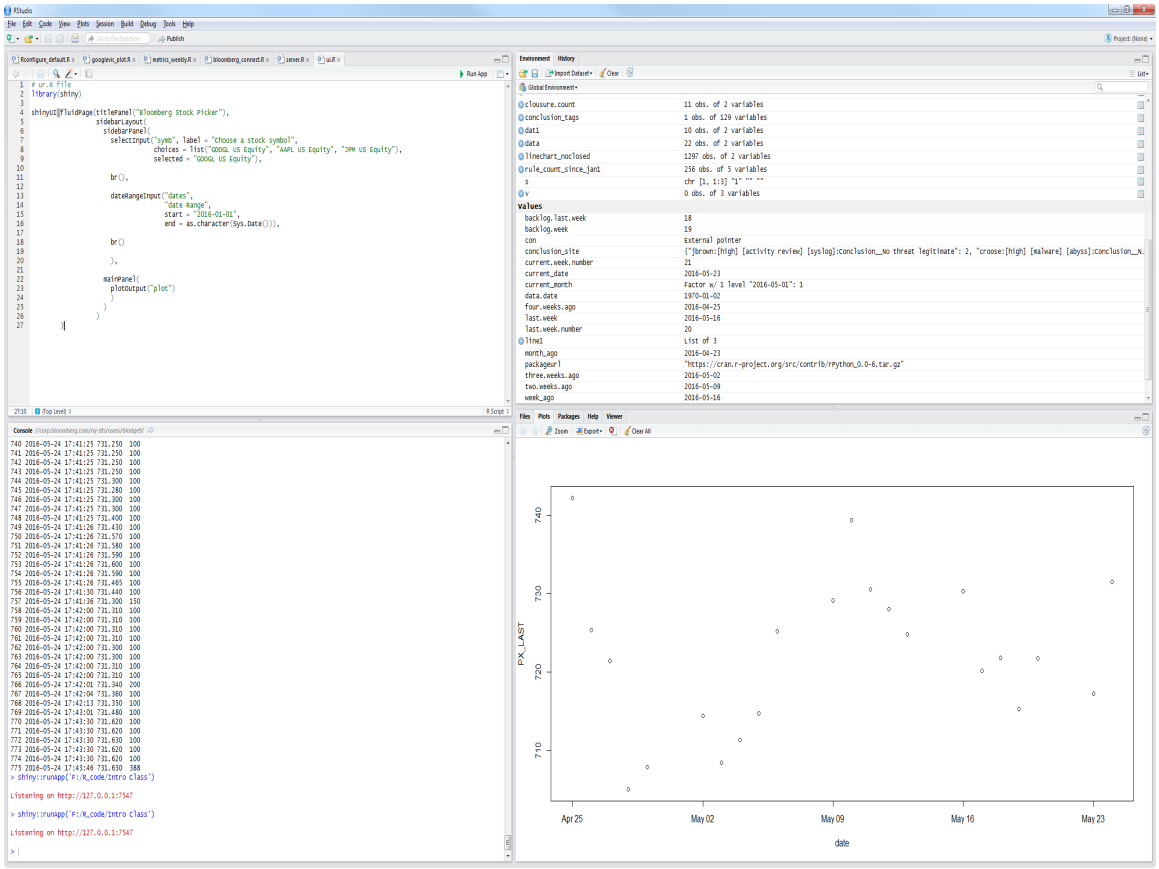
R Studio Install<

- **R Class Installation Guide:**
- **Install R and R Studio – RStudio requires R 2.11.1 (or higher)**
- **You'll need admin rights to perform the install.**

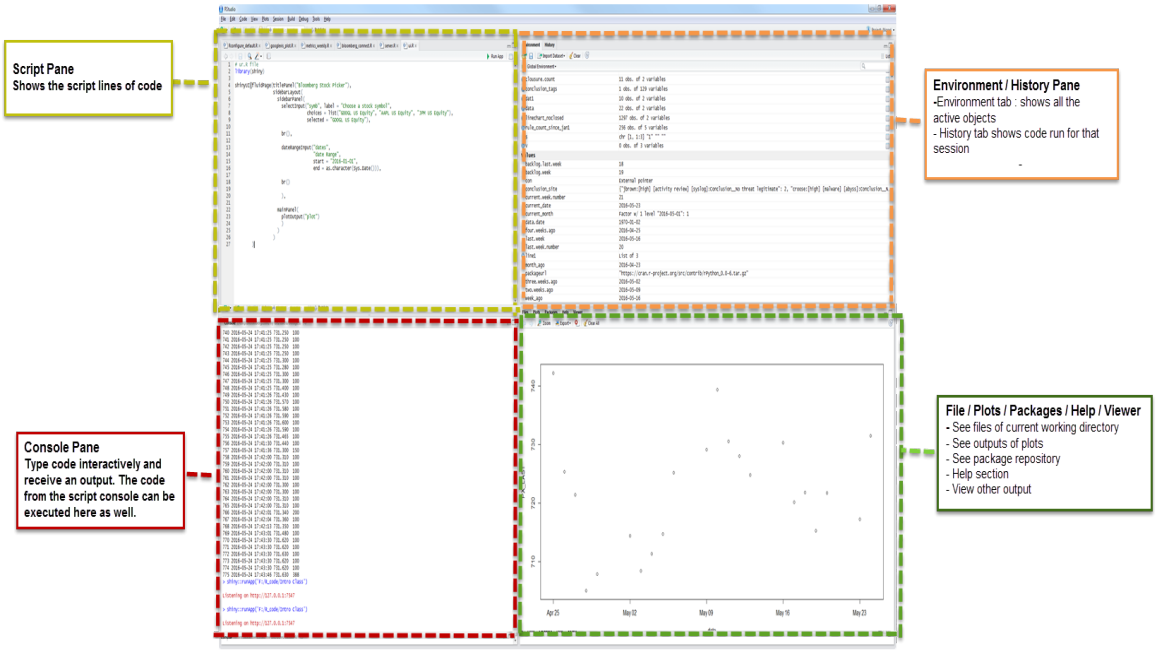
- **2 Step install R > RStudio1.**

- **1. Install R here –**
  - **https://cran.rstudio.com/**
  - **Download for your OS (Linux, Mac, Windows)**

- **2. Install R Studio- Install RStudio here -**
  https://www.rstudio.com/products/rstudio/download/
- **- Download the installer for your OS (Linux, Mac, Windows)**


RStudio
File Edit Code View Plots Session Build Debug Tools Help
Publish
Project: (None)
Reconfigure_default.R
googleis_plot.R
metrics_weekly.R
bloomberg_connect.R
server.R
ui.R
Run App
1 # ui.R File
2 library(shiny)
3
4 shinyUI(fluidPage(titlePanel("Bloomberg Stock Picker"),
5 sidebarLayout(
6 sidebarPanel(
7 selectInput("symb", label = "choose a stock symbol",
8 choices = list("GOOGL US Equity", "AAPL US Equity", "JPM US Equity"),
9 selected = "GOOGL US Equity"),
10
11 br(),
12
13 dateRangeInput("dates",
14 "date Range",
15 start = "2016-01-01",
16 end = as.character(Sys.Date())),
17
18 br()
19
20),
21
22 mainPanel(
23 plotoutput("plot")
24)
25)
26)
27)
27:10
(Top Level)
R Script
Console
740 2016-05-24 17:41:25 731.250 100
741 2016-05-24 17:41:25 731.250 100
742 2016-05-24 17:41:25 731.250 100
743 2016-05-24 17:41:25 731.250 100
744 2016-05-24 17:41:25 731.300 100
745 2016-05-24 17:41:25 731.280 100
746 2016-05-24 17:41:25 731.300 100
747 2016-05-24 17:41:25 731.300 100
748 2016-05-24 17:41:25 731.400 100
749 2016-05-24 17:41:26 731.430 100
750 2016-05-24 17:41:26 731.570 100
751 2016-05-24 17:41:26 731.580 100
752 2016-05-24 17:41:26 731.590 100
753 2016-05-24 17:41:26 731.600 100
754 2016-05-24 17:41:26 731.590 100
755 2016-05-24 17:41:26 731.465 100
756 2016-05-24 17:41:30 731.440 100
757 2016-05-24 17:41:36 731.300 150
758 2016-05-24 17:42:00 731.310 100
759 2016-05-24 17:42:00 731.310 100
760 2016-05-24 17:42:00 731.310 100
761 2016-05-24 17:42:00 731.310 100
762 2016-05-24 17:42:00 731.300 100
763 2016-05-24 17:42:00 731.300 100
764 2016-05-24 17:42:00 731.310 100
765 2016-05-24 17:42:00 731.310 100
766 2016-05-24 17:42:01 731.340 200
767 2016-05-24 17:42:04 731.360 100
768 2016-05-24 17:42:13 731.350 100
769 2016-05-24 17:43:01 731.480 100
770 2016-05-24 17:43:30 731.620 100
771 2016-05-24 17:43:30 731.620 100
772 2016-05-24 17:43:30 731.630 100
773 2016-05-24 17:43:30 731.620 100
774 2016-05-24 17:43:30 731.620 100
775 2016-05-24 17:43:46 731.630 388
> shiny::runapp('F:/R_code/Intro Class')
Listening on http://127.0.0.1:7547
> shiny::runapp('F:/R_code/Intro Class')
Listening on http://127.0.0.1:7547
>
Environment History
Import Dataset
Clear
List
Global Environment
closure.count 11 obs. of 2 variables
conclusion_tags 1 obs. of 129 variables
dat1 10 obs. of 2 variables
data 22 obs. of 2 variables
linechart_noclosed 1297 obs. of 2 variables
rule_count_since_jan1 256 obs. of 5 variables
s chr [1, 1:3] "1" "" ""
v 0 obs. of 3 variables
Values
backlog.last.week 18
backlog.week 19
con External pointer
conclusion_site {"jbrown:[high] [activity review] [syslog]:conclusion__no threat legitimate": 2, "croose:[high] [malware] [abyss]:conclusion__N..
current.week.number 21
current_date 2016-05-23
current_month Factor w/ 1 level "2016-05-01": 1
data.date 1970-01-02
four.weeks.ago 2016-04-25
last.week 2016-05-16
last.week.number 20
line1 List of 3
month_ago 2016-04-23
packageurl "https://cran.r-project.org/src/contrib/rPython_0.0-6.tar.gz"
three.weeks.ago 2016-05-02
two.weeks.ago 2016-05-09
week_ago 2016-05-16
Files Plots Packages Help Viewer
Zoom
Export
Clear All
PX_LAST
740
730
720
710
Apr 25
May 02
May 09
May 16
May 23
date

R Studio IDE breakdown



Starting with R

- R is in fact scripting or programming
- I am expecting that you have some familiarity with scripting (Bash, Visual Basic, Perl, Python, Ruby, etc)
- Or have some experience with programming (C++, C#, Java, etc)
- We don't intend to write thousand line programs in this class but we'll show how to issue R commands

Online R Resources

- GOOGLE
- CRAN - The main R site: http://cran.r-project.org
- Stack Overflow R section

Tag Info

info newest 4 featured frequent votes active unanswered

173,698
questions tagged
r

Ask Question

About r

R is a free, open-source programming language and software environment for statistical computing, bioinformatics, and graphics. Please supplement your question with a minimal reproducible example. Use dput() for data and specify all non-base packages with library calls. For statistical questions please use http://stats.stackexchange.com.

R Programming Language

R is a free, open-source programming language and software environment for statistical computing, bioinformatics, and graphics. It is a multi-paradigm language and dynamically typed. R is an implementation of the S programming language combined with lexical scoping semantics inspired by Scheme. R was created by Ross Ihaka and Robert Gentleman and is now developed by the R Development Core Team. The R environment is easily extended through a packaging system on CRAN, the Comprehensive R Archive Network.

Scope of questions

This tag should be used for programming-related questions about R. Including a minimal reproducible example in your question will increase your chances of getting a timely, useful answer. Questions should *not* use the rstudio tag unless they relate specifically to the RStudio interface and not just the R language.

If your question is more focused on statistics or data science, use Cross Validated or Data Science, respectively. Bioinformatics-specific questions may be better received on Bioconductor Support or Biostars. General questions about R (such as requests for off-site resources or discussion questions) are unsuitable for StackOverflow and may be appropriate for one of the general, or special-interest, R mailing lists.

Please do not cross-post across multiple venues. Do research (read tag wikis, look at existing questions, or search online) to determine the most appropriate venue so that you have a better chance of receiving solutions to your question. Your question may be automatically migrated to a more appropriate StackOverflow site. If you receive **no** response to your questions after a few days, or if your question is put on-hold for being off-topic, it is then OK to post to another venue, giving a link to your StackOverflow question - but don't cross-post just because your question is down-voted or put on hold for being unclear. *Instead, work on improving your question.*

Synonyms

rstats r-language

Stats

created 8 years ago by David Locke
viewed 50017 times
active 1 month ago
editors 69

Top Answerers


akrun
242k ● 9 ● 71 ● 122


Dirk Eddelbuettel
222k ● 24 ● 405 ● 509


42-
175k ● 10 ● 173 ● 305


A5C1D2H2I1M1N2O1R2T1
124k ● 15 ● 179 ● 267

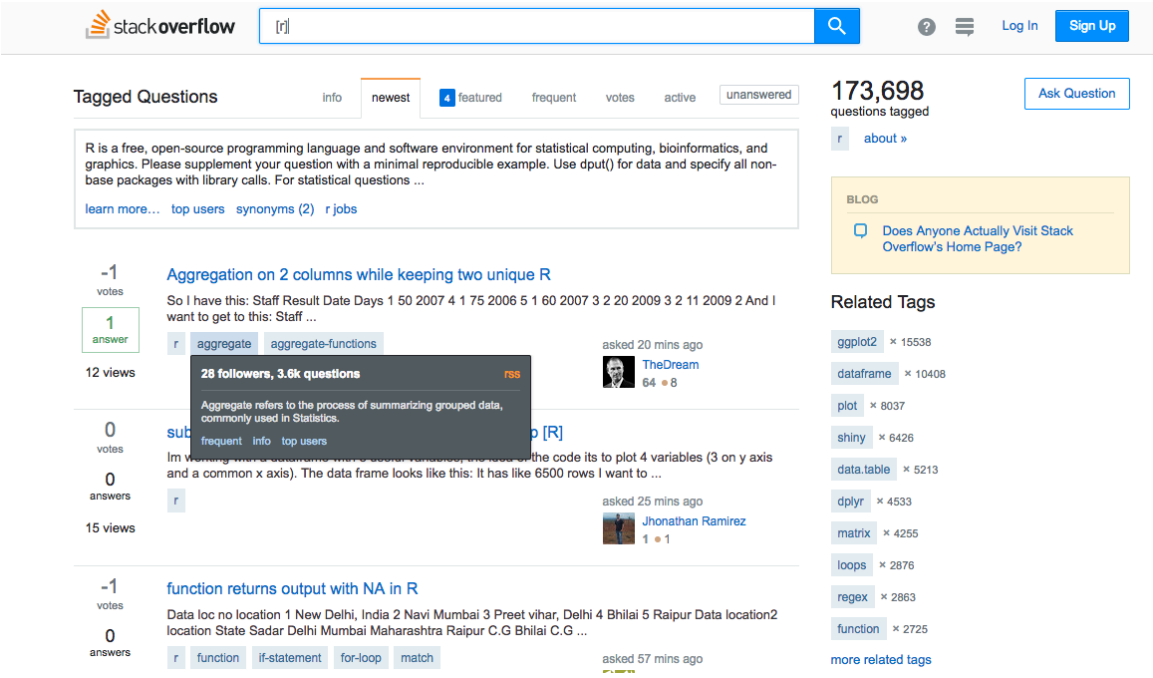

Gavin Simpson
111k ● 14 ● 230 ● 319

more »

Recent Hot Answers

R: possible truncation of >= 4GB file

How to save row names when selecting each column independently instead of row number?


stackoverflow
[r]
Log In
Sign Up
Tagged Questions
info
newest
featured
frequent
votes
active
unanswered
173,698
questions tagged
Ask Question
r
about »
R is a free, open-source programming language and software environment for statistical computing, bioinformatics, and graphics. Please supplement your question with a minimal reproducible example. Use dput() for data and specify all non-base packages with library calls. For statistical questions ...
learn more… top users synonyms (2) r jobs
BLOG
Does Anyone Actually Visit Stack Overflow's Home Page?
-1
votes
1
answer
12 views
Aggregation on 2 columns while keeping two unique R
So I have this: Staff Result Date Days 1 50 2007 4 1 75 2006 5 1 60 2007 3 2 20 2009 3 2 11 2009 2 And I want to get to this: Staff ...
r
aggregate
aggregate-functions
asked 20 mins ago
TheDream
64 ● 8
28 followers, 3.6k questions
rss
Aggregate refers to the process of summarizing grouped data, commonly used in Statistics.
frequent info top users
0
votes
0
answers
15 views
p [R]
the code its to plot 4 variables (3 on y axis and a common x axis). The data frame looks like this: It has like 6500 rows I want to ...
r
asked 25 mins ago
Jhonathan Ramirez
1 ● 1
-1
votes
0
answers
function returns output with NA in R
Data loc no location 1 New Delhi, India 2 Navi Mumbai 3 Preet vihar, Delhi 4 Bhilai 5 Raipur Data location2 location State Sadar Delhi Mumbai Maharashtra Raipur C.G Bhilai C.G ...
r
function
if-statement
for-loop
match
asked 57 mins ago
Related Tags
ggplot2 × 15538
dataframe × 10408
plot × 8037
shiny × 6426
data.table × 5213
dplyr × 4533
matrix × 4255
loops × 2876
regex × 2863
function × 2725
more related tags

R best practices

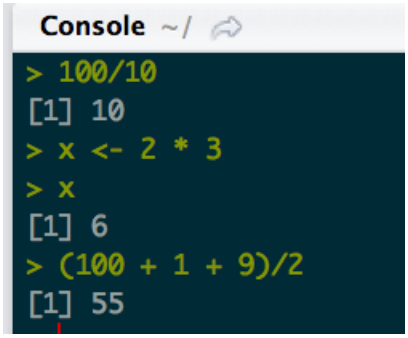
- Ensure your code is easy to read for yourself and others
  - Use spaces
  - Variable names should be succinct, yet informative
  - Use #comments
  - Remove any code that is unnecessary or trial code
  - The more concise your code, the easier it is to understand and easier to fix

R help

- R has a very good help system built in
- If you need help on a package, function, etc use the Help section in the bottom right pane

R as a calculator

- The R console or standard out functions much like a calculator
- Assignment statements
  - Object_name < value
  - You can mentally read this aloud as “object name gets (“<”) value”
- Can use “=“ instead of “<” but it gets confusing when you begin using math formulas



R tips and tricks

- “>” the prompt
- Keyboard Shortcut [ALT-] = “<-”
- Objects
  - Must start with a letter and can only contain letters, numbers, “_” and “.”
- Code completion using [Tab] for variables, functions, packages, etc
- “+” after code generation means you are missing some syntax
  - Comma
  - Quote
  - Parenthesis
  - Press [ESC] to try again

R packages

- The fundamental units of reproducible code.
- They include
  - Reusable functions
  - Documentation

Comparisons & Boolean & NA

- <
- >
- <=
- >=
- !=
- ==
- & ”and”
- | “or”
- ! “not”
- NA “not available”
- is.na – determine if a value is missing or NA

- How do find those values that are not NA?

R - Objects

- Must start with a letter
- Can only contain letters, numbers, '_', '.'
- You'll want your object name to be explanatory and you'll need a naming convention for multiple words

```
I_like_to_use_this <- 1
OtherPeopleDoThis <- 1
Some.People.Do.This < 1
And_a.Few.People_will.do.THIS

This_is_a_really_long_data_object < 2.5
```

You can use RStudio's auto completion facility by typing "this"+[tab]

Press Cmd+CTRL+ ↑ and you'll see the last commands you've typed

R – Objects continued

- There are many types of R-objects.
  The frequently used ones are:
  - Vectors
  - Lists
  - Matrices
  - Arrays
  - Factors
  - Data Frames

Vector Object

- 6 data types of vectors
  - Logical
  - Numeric
  - Integer
  - Complex
  - Character
  - Raw

Data types

- int – integer - as.integer(3.14) = 3
- dbl – double or real number - 1, 2.5, 4.5
- chr – character vector/ string - character
- dttm – date-time – 2017-01-01
- lgl – logical – TRUE or FALSE
- fctr – factors R uses to represent categorical variables with fixed possible values
- date – dates - as.Date(as.POSIXct("2013-01-01 07:00", 'GMT')

  [1]

  "2013-01-01"

Data Type	Example	Verify

Logical	TRUE, FALSE	`v <- TRUE`
		`print(class(v))`
		it produces the following result —
		`[1] "logical"`

Numeric	12.3, 5, 999	`v <- 23.5`
		`print(class(v))`
		it produces the following result —
		`[1] "numeric"`

Integer	2L, 34L, 0L	`v <- 2L`
		`print(class(v))`
		it produces the following result —
		`[1] "integer"`

Complex	3 + 2i	`v <- 2+5i`
		`print(class(v))`
		it produces the following result —
		`[1] "complex"`

Character	'a' , '"good", "TRUE", '23.4'	`v <- "TRUE"`
		`print(class(v))`
		it produces the following result —
		`[1] "character"`

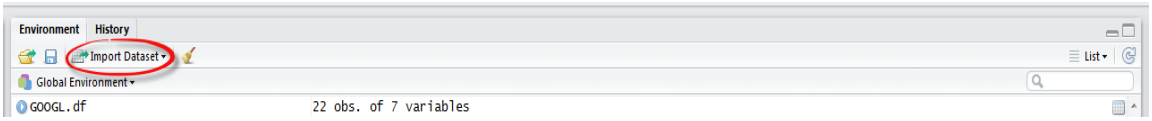
Raw	"Hello" is stored as 48 65 6c 6c 6f	`v <- charToRaw("Hello")`
		`print(class(v))`
		it produces the following result —
		`[1] "raw"`

Export & Import

Step 1: Run the code to pull googles stock price for the last 31 days (missing weekends because stocks are not traded then)

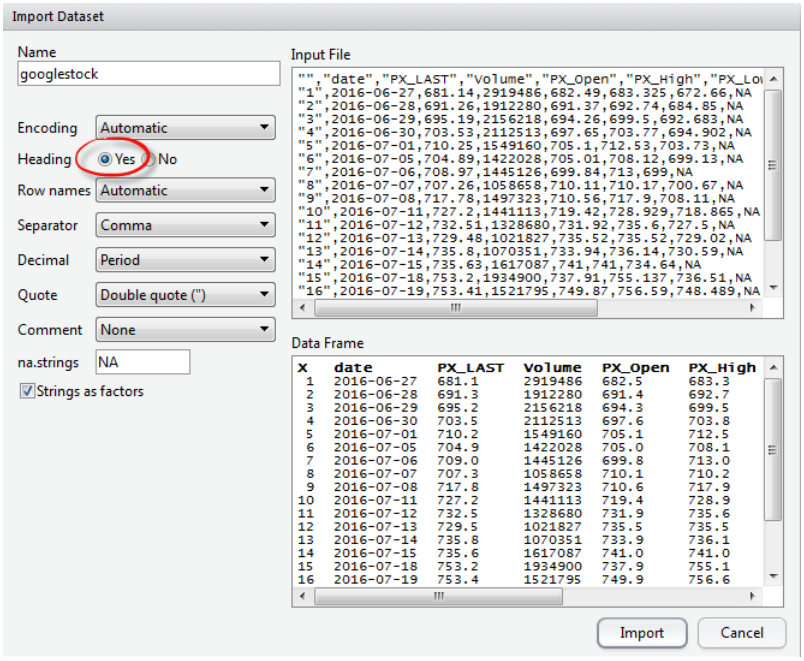
```
1  library(quantmod)
2  getSymbols("GOOGL")
3  GOOGL.2016 <- GOOGL['2016']
4  GOOGL.2016.df <- as.data.frame(GOOGL.2016)
5
6  write.csv(GOOGL.2016.df, "googlestock.csv")
```

Step 2: View the GOOGL.df data frame
Step 3: write the data frame to a csv output



Step 4: Import Dataset you just sent to csv file

Import


Import Dataset
Name
googlestock
Encoding
Automatic
Heading
Yes
No
Row names
Automatic
Separator
Comma
Decimal
Period
Quote
Double quote (")
Comment
None
na.strings
NA
Strings as factors
Input File
"","date","PX_LAST","Volume","PX_Open","PX_High","PX_Lo
"1",2016-06-27,681.14,2919486,682.49,683.325,672.66,NA
"2",2016-06-28,691.26,1912280,691.37,692.74,684.85,NA
"3",2016-06-29,695.19,2156218,694.26,699.5,692.683,NA
"4",2016-06-30,703.53,2112513,697.65,703.77,694.902,NA
"5",2016-07-01,710.25,1549160,705.1,712.53,703.73,NA
"6",2016-07-05,704.89,1422028,705.01,708.12,699.13,NA
"7",2016-07-06,708.97,1445126,699.84,713,699,NA
"8",2016-07-07,707.26,1058658,710.11,710.17,700.67,NA
"9",2016-07-08,717.78,1497323,710.56,717.9,708.11,NA
"10",2016-07-11,727.2,1441113,719.42,728.929,718.865,NA
"11",2016-07-12,732.51,1328680,731.92,735.6,727.5,NA
"12",2016-07-13,729.48,1021827,735.52,735.52,729.02,NA
"13",2016-07-14,735.8,1070351,733.94,736.14,730.59,NA
"14",2016-07-15,735.63,1617087,741,741,734.64,NA
"15",2016-07-18,753.2,1934900,737.91,755.137,736.51,NA
"16",2016-07-19,753.41,1521795,749.87,756.59,748.489,NA
Data Frame
X date PX_LAST Volume PX_Open PX_High
1 2016-06-27 681.1 2919486 682.5 683.3
2 2016-06-28 691.3 1912280 691.4 692.7
3 2016-06-29 695.2 2156218 694.3 699.5
4 2016-06-30 703.5 2112513 697.6 703.8
5 2016-07-01 710.2 1549160 705.1 712.5
6 2016-07-05 704.9 1422028 705.0 708.1
7 2016-07-06 709.0 1445126 699.8 713.0
8 2016-07-07 707.3 1058658 710.1 710.2
9 2016-07-08 717.8 1497323 710.6 717.9
10 2016-07-11 727.2 1441113 719.4 728.9
11 2016-07-12 732.5 1328680 731.9 735.6
12 2016-07-13 729.5 1021827 735.5 735.5
13 2016-07-14 735.8 1070351 733.9 736.1
14 2016-07-15 735.6 1617087 741.0 741.0
15 2016-07-18 753.2 1934900 737.9 755.1
16 2016-07-19 753.4 1521795 749.9 756.6
Import
Cancel

Hands on challenge

- **Using the library(quantmod), pick your own stock to answer the following questions:**

1. **Pick 3 fields**
2. **Write your new data frame to csv**
3. **Import that data frame to R**
4. **View the data frame in R**

Bonus

- **Remove a column of data from your data frame (use help to find remove column function)**

Hands on challenge

- **Using the library(quantmod), pick your own stock to answer the following questions:**

1. **Remove 3 fields**
2. **Write your new data frame to csv**
3. **Import that data frame to R**
4. **View the data frame in R**

Bonus

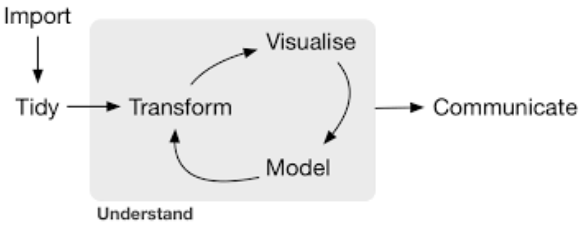
- **Find an additional column that does not come with the original dataframe.**
  - **Check out http://www.quantmod.com/examples/data/**

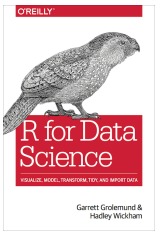
Section 2

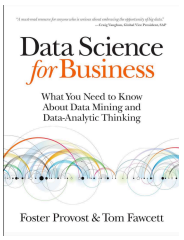
- **Online Course Layout:**
  - **Section 1 -Familiarity with R**
  - **Section 2 – Data Wrangling**
    - **API's**
    - **Data store types**
    - **Tidying**
    - **Dplyr**
    - **Data wrangling stock data**
    - **Stats wrangling**
    - **Hands-on exercise with a stock of your choice**
    - **Regular expressions**
  - **Section 3 – Data Visualization**
  - **Section 4 – R Markdown**
  - **Section 5 – Exploratory Data Analysis**
    - **Diamond Exercise**
    - **Bank Marketing Exercise**
  - **Section 6 – Introduction to Regression**
  - **Section 7 – Introduction to Machine Learning**
    - **Titanic Kaggle Competition**
  - **Section 8 – Strategy**
    - **Big box store competitors**

Format going forward<

- **The format of the course onward will be broken out by the typical data analysis process explained by Hadley Wickham, author of "R for Data Science"**
- **Each data "problem" should be answered in this model**
- **80% of the heavy lifting in this model will be within the following sections Import->Tidy-> Transform**



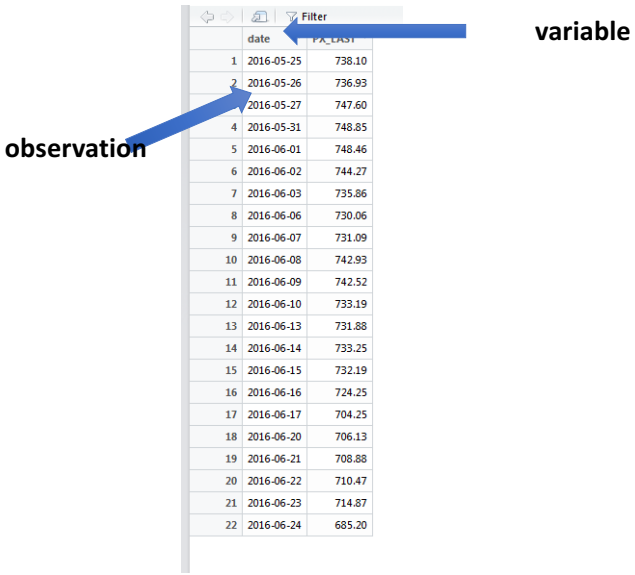




http://r4ds.had.co.nz/intro.html

Tidying

- **Getting it prepped in a standard format with variables (columns) and observations (rows)**



Dplyr Tidying

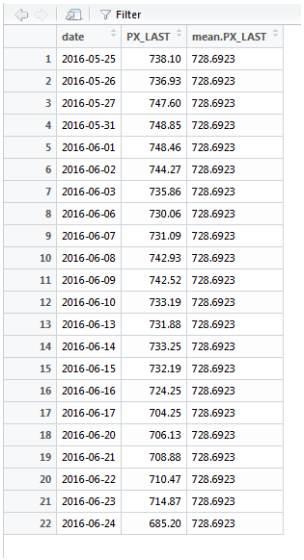
- **Filter() - pick observations by their values**
- **Arrange() – reorder the rows**
- **Select() – pick variables by their names**
- **Mutate() – create new variables with functions of existing variables**
- **Summarize() – collapse many variables down to a single summary**
- **group_by() – changes the scope of each function from operating on the entire data set to operating on it group-by-group**

Dplyr Tidying

- **Dplyr Fishery Example code**

Transformation

- **Data summaries provide overviews of key properties of the data**
- **Goal is to describe important properties of the distribution of the values across observations that were measured**
- **Implying knowledge generation against the dataset**
  - **Subset**
  - **Mathematical functions**
  - **Calculations**
  - **Summary statistics**
  - **sorting**



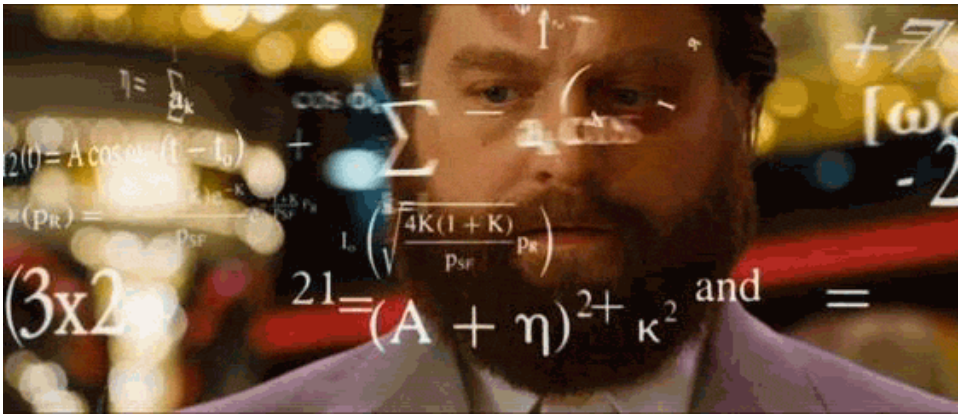
Filter

	date	PX_LAST	mean.PX_LAST
1	2016-05-25	738.10	728.6923
2	2016-05-26	736.93	728.6923
3	2016-05-27	747.60	728.6923
4	2016-05-31	748.85	728.6923
5	2016-06-01	748.46	728.6923
6	2016-06-02	744.27	728.6923
7	2016-06-03	735.86	728.6923
8	2016-06-06	730.06	728.6923
9	2016-06-07	731.09	728.6923
10	2016-06-08	742.93	728.6923
11	2016-06-09	742.52	728.6923
12	2016-06-10	733.19	728.6923
13	2016-06-13	731.88	728.6923
14	2016-06-14	733.25	728.6923
15	2016-06-15	732.19	728.6923
16	2016-06-16	724.25	728.6923
17	2016-06-17	704.25	728.6923
18	2016-06-20	706.13	728.6923
19	2016-06-21	708.88	728.6923
20	2016-06-22	710.47	728.6923
21	2016-06-23	714.87	728.6923
22	2016-06-24	685.20	728.6923

Transformation continued…

- **Mean – average**
- **Median – middle value**
- **Mode – most common value**
- **Range – difference between the largest and the smallest value**
- **Variance – numerical measure of how the data values are dispersed around the mean. The average of the squared distances from the mean**
- **Standard deviation – a measure of how spread out the numbers are**

```
subset(GOOGL.df, PX_LAST > 738)

median(GOOGL.df$PX_LAST)

centralValue(GOOGL.df$date)

max(GOOGL.df$PX_LAST)

var(GOOGL.df$PX_LAST)

sd(GOOGL.df$PX_LAST)

describe(GOOGL.df)

summary(GOOGL.df)

range(GOOGL.df$PX_LAST)

#mathisfun.com/data/standard-deviation.html
```


21=(A + η)² + κ² and =

Cheat Sheet

- **Mean – average**
- **Median – middle value**
- **Mode – most common value**
- **Range – difference between the largest and the smallest value**
- **Variance – numerical measure of how the data values are dispersed around the mean. The average of the squared distances from the mean**
- **Standard deviation – a measure of how spread out the numbers are**

```
subset(GOOGL.df, PX_LAST > 738)

median(GOOGL.df$PX_LAST)

centralvalue(GOOGL.df$date)

max(GOOGL.df$PX_LAST)

var(GOOGL.df$PX_LAST)

sd(GOOGL.df$PX_LAST)

describe(GOOGL.df)

summary(GOOGL.df)

range(GOOGL.df$PX_LAST)

#mathisfun.com/data/standard-deviation.html
```

Hands on challenge

- **Using the librcary(quantmod), pick your own stock to answer the following questions:**

1. **Select stock prices for your stock of choice for the last 31 days - library(quantmod)**
2. **Subset the data to the last 10 days and turn it into a data frame**
3. **What is the highest value and lowest value of your stock within the last 31 days?**
4. **What was the average price of your stock of choice for the last 31 days**
5. **Find the average of the squared distances from the mean for the stock prices in the last 31 days. Hint: this is a single function**
6. **Would you buy or sell your stock tomorrow? Explain why or why not.**

API's

- **API** – application Programming Interface
  - It's a means of accessing the functionality of a program from inside another program
  - Allows for flexibility and customization of the data scientist and not depend on the application for exporting on an ad-hoc basis
- API's are the driving force behind data wrangling
- They allow machines to access data programmatically through specific formatting, keys and API calls

Wrangling an API

- **Consist of a URL to a domain and a path**
  - **Example: https://api.census.gov/data/2015/acs5?get=…**
    - **Api.census.gov is the URL**
    - **After the get= is the data to retrieve**
  - **HTTP – hyper text protocol in which the web is built on**
    - **GET – command request from a client to query a server and recieves an answer**
    - **POST – sends a data payload to a server (from a client)**

API types

- **JSON – JavaScript Object Notation**
  - **Emerging as the go-to standard for API format**
  - **Less characters and no tags but brackets**

- **XML – eXtended Markup Language**
  - **Legacy format and slower with higher processing power to display characters and tags but still used**
  - **Includes HTML tags like <id> or <item> and </id> or </item> to close the tag**

JSON Example


```
"product" : {
    "id" : 15,
    "name" : "Widgets",
    "description" : "These widgets are the finest widgets ever made by anyone.",
    "options" : [
        {
            "type" : "color",
            "items" : [
                "Purple",
                "Green",
                "Orange"
            ]
        }
    ]
}
```

XML Example

```
<product>

   <id>15</id>

   <name>Widgets</name>

   <description>These widgets are the finest widgets ever made by anyone.
</description>

   <options type="color">

      <item>Purple</item>

      <item>Green</item>

      <item>Orange</item>

   </options>

</product>
```

API Exercise

- We'll now pull data from the census bureau
- The API site is here
- http://proximityone.com/zipcode_data_analytics.htm#option3
- Leverage the data dictionary provided to assign the column names
- Use transformation techniques to find interesting figures
  - Average House Hold income?
  - Sum the population of the US

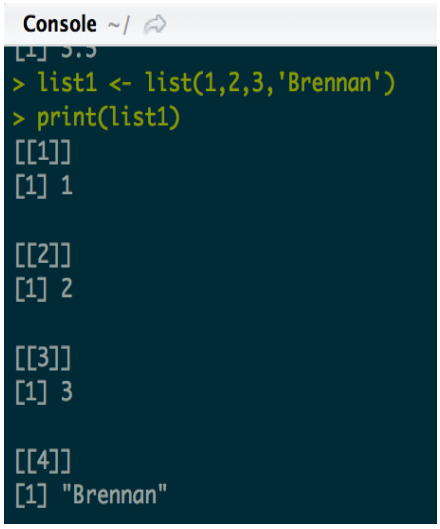
Creating multiple vectors

- When you want to create a vector with more than one element, you must use the ‘c()’ function which allows you to combine the elements into a vector

```
# Create a vector.
apple <- c('red','green',"yellow")
print(apple)

# Get the class of the vector.
print(class(apple))
```

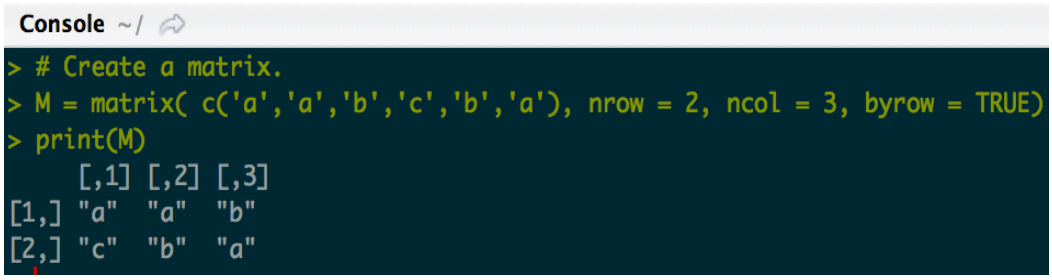
Lists

- In addition to vectors (created with the c() operator), A list is an R-object which can contain many different types of elements inside it like vectors, functions and even another list inside it
- The basic R *list* is created as the list() operator



Matrices

- Matrices are two-dimensional structures addresses by rows and columns
- It can be created using a vector input to the matrix function



Arrays

- While matrices are confined to two dimensions, arrays can be of any number of dimensions. The array functions takes a ‘dim’ attribute which creates the require number of dimensions. We’ll create an array with two elements which are 3x3 matrices each

```
> array1 <- array(c('red','orange','yellow'), dim = c(3,3,2))
> print(array1)
, , 1

     [,1]     [,2]     [,3]
[1,] "red"    "red"    "red"
[2,] "orange" "orange" "orange"
[3,] "yellow" "yellow" "yellow"

, , 2

     [,1]     [,2]     [,3]
[1,] "red"    "red"    "red"
[2,] "orange" "orange" "orange"
[3,] "yellow" "yellow" "yellow"
```

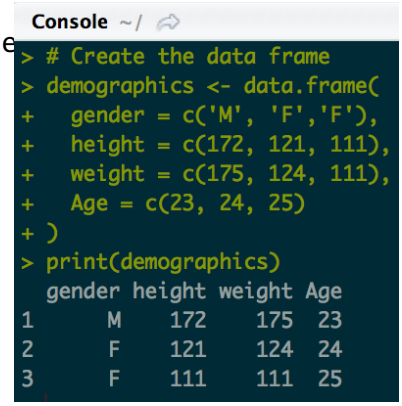
Factors

- Factors are the r-objects which are created using a vector. It stores the vector along with the distinct values of the elements in the vector as labels. The labels are always character irrespective of whether it is numeric or character or Boolean in the input vector. They are useful in statistical modeling.
- Factors are created using the **factor()** function. The **nlevels** function gives the count of levels.

```
> crayons <- c('red', 'blue', 'green', 'yellow',
+              'orange', 'violet', 'brown', 'black',
+              'red','red','yellow','brown','orange','blue')
> factor_crayons <- factor(crayons)
> print(factor_crayons)
 [1] red    blue   green  yellow orange violet brown  black  red    red
[11] yellow brown  orange blue
Levels: black blue brown green orange red violet yellow
> print(nlevels(factor_crayons))
[1] 8
>
```

data.frame

- R's central data structure is called the *data frame.* A data frame is organized into rows and columns. A data frame is a list of columns of different types. Each row has a value for each column. An R data frame is much like a database table: the column types and names are the schema and the rows are the data. In R, you can quickly create a data frame using **data.frame()** command
- Other useful functions with the data.frame
  - colnames()
  - summary()
  - dim()
- 80% of a data scientists work is figuring out how to transform data into this form



Example code of data structures

Regular Expressions

- The R for statistical computing provides multiple regular expression functions in its **base** package

- We can also use the **stringr** package to provide string operations

- The **grep** function takes your regex as the first argument and the input vector as the second argument.
  - If you use 'value=FALSE' then **grep** returns a new vector
  - If you use 'value = TRUE' it will return the actual matches
- Using **grepl** is similar to **grep** but it returns a logical vector (TRUE or FALSE)
- **sub** and **gsub** will perform replacement of a found regex pattern

Regular Expression syntax

- Regex's will use metacharacters that have specific meaning
  - **$ * + . ? [ ] ^ { } | ( ) **
    - **$** matches the end of the string
    - ***** matches at least 0 times
    - . matches any single character
    - **+** matches at least 1 time
    - **?** matches at most 1 time
    - […] a character list, matches any one of the characters inside the square brackets
    - **^** matches the start of string
    - **{n}** matches exactly n times
    - **| or**
    - **(…)** grouping to allow you to retrieve matches or capturing
  - **\n = newline**
  - **\r carriage return** or a control character or mechanism used to reset a device's position to the beginning of a line of text.
  - **\t** tab
  - **\b** backspace
  - **\\ backslash** or suppress the special meaning of the metacharacters

Basic Regular Expressions in R
Cheat Sheet

Character Classes

[[:digit:]] or \\d	Digits; `[0-9]`
\\D	Non-digits; `[^0-9]`
[[:lower:]]	Lower-case letters; `[a-z]`
[[:upper:]]	Upper-case letters; `[A-Z]`
[[:alpha:]]	Alphabetic characters; `[A-z]`
[[:alnum:]]	Alphanumeric characters `[A-z0-9]`
\\w	Word characters; `[A-z0-9_]`
\\W	Non-word characters
[[:xdigit:]] or \\x	Hexadec. digits; `[0-9A-Fa-f]`
[[:blank:]]	Space and tab
[[:space:]] or \\s	Space, tab, vertical tab, newline, form feed, carriage return
\\S	Not space; `[^[:space:]]`
[[:punct:]]	Punctuation characters; `!"#$%&'()*+,-./:;<=>?@[]^_`{|}~`
[[:graph:]]	Graphical char.; `[[:alnum:][:punct:]]`
[[:print:]]	Printable characters; `[[:alnum:][:punct:]\\s]`
[[:cntrl:]] or \\c	Control characters; `\n`, `\r` etc.

Special Metacharacters

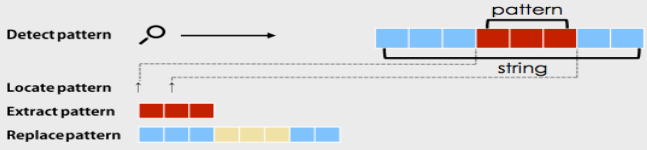
\n	New line
\r	Carriage return
\t	Tab
\v	Vertical tab
\f	Form feed

Lookaraounds and Conditionals*

(?=)	Lookahead (requires PERL = TRUE), e.g. `(?=yx)`: position followed by 'xy'
(?!)	Negative lookahead (PERL = TRUE); position NOT followed by pattern
(?<=)	Lookbehind (PERL = TRUE), e.g. `(?<=yx)`: position following 'xy'
(?<!)	Negative lookbehind (PERL = TRUE); position NOT following pattern
?(if)then	If-then-condition (PERL = TRUE); use lookaheads, optional char. etc in if-clause
?(if)then|else	If-then-else-condition (PERL = TRUE)

*see, e.g. http://www.regular-expressions.info/lookaround.html http://www.regular-expressions.info/conditional.html

Functions for Pattern Matching




```
> string <- c("Hiphopopotamus", "Rhymenoceros", "time for bottomless lyrics")
> pattern <- "t.m"
```

Detect Patterns

grep(pattern, string)
`[1] 1 3`

grep(pattern, string, value = TRUE)
`[1] "Hiphopopotamus"`
`[2] "time for bottomless lyrics"`

grepl(pattern, string)
`[1] TRUE FALSE TRUE`

stringr::str_detect(string, pattern)
`[1] TRUE FALSE TRUE`

Locate Patterns

regexpr(pattern, string)
find starting position and length of first match

gregexpr(pattern, string)
find starting position and length of all matches

stringr::**str_locate(string, pattern)**
find starting and end position of first match

stringr::**str_locate_all(string, pattern)**
find starting and end position of all matches

Split a String using a Pattern

strsplit(string, pattern) or stringr::**str_split(string, pattern)**

Extract Patterns

regmatches(string, regexpr(pattern, string))
extract first match `[1] "tam" "tim"`

regmatches(string, gregexpr(pattern, string))
extracts all matches, outputs a list
`[[1]] "tam" [[2]] character(0) [[3]] "tim" "tom"`

stringr::**str_extract(string, pattern)**
extract first match `[1] "tam" NA "tim"`

stringr::**str_extract_all(string, pattern)**
extract all matches, outputs a list

stringr::**str_extract_all(string, pattern, simplify = TRUE)**
extract all matches, outputs a matrix

stringr::**str_match(string, pattern)**
extract first match + individual character groups

stringr::**str_match_all(string, pattern)**
extract all matches + individual character groups

Replace Patterns

sub(pattern, replacement, string)
replace first match

gsub(pattern, replacement, string)
replace all matches

stringr::**str_replace(string, pattern, replacement)**
replace first match

stringr::**str_replace_all(string, pattern, replacement)**
replace all matches

Character Classes and Groups

.	Any character except \n
|	Or, e.g. (a|b)
[...]	List permitted characters, e.g. [abc]
[a-z]	Specify character ranges
[^...]	List excluded characters
(...)	Grouping, enables back referencing using \\N where N is an integer

Anchors

^	Start of the string
$	End of the string
\\b	Empty string at either edge of a word
\\B	NOT the edge of a word
\\<	Beginning of a word
\\>	End of a word

Quantifiers

*	Matches at least 0 times
+	Matches at least 1 time
?	Matches at most 1 time; optional string
{n}	Matches exactly n times
{n,}	Matches at least n times
{,n}	Matches at most n times
{n,m}	Matches between n and m times

General Modes

By default R uses *POSIX extended regular expressions*. You can switch to *PCRE regular expressions* using `PERL = TRUE` for base or by wrapping patterns with `perl()` for stringr.

All functions can be used with literal searches using `fixed = TRUE` for base or by wrapping patterns with `fixed()` for stringr.

All base functions can be made case insensitive by specifying `ignore.cases = TRUE`.

Escaping Characters

Metacharacters (`. * +` etc.) can be used as literal characters by escaping them. Characters can be escaped using `\\` or by enclosing them in `\\Q...\\E`.

Case Conversions

Regular expressions can be made case insensitive using `(?i)`. In backreferences, the strings can be converted to lower or upper case using `\\L` or `\\U` (e.g. `\\L\\1`). This requires `PERL = TRUE`.

Greedy Matching

By default the asterisk * is greedy, i.e. it always matches the longest possible string. It can be used in lazy mode by adding `?`, i.e. `*?`.

Greedy mode can be turned off using `(?U)`. This switches the syntax, so that `(?U)a*` is lazy and `(?U)a*?` is greedy.

Note

Regular expressions can conveniently be created using rex::**rex()**.

https://www.rstudio.com/wp-content/uploads/2016/09/RegExCheatsheet.pdf

Section 2 Exercise

Section 3

- **Online Course Layout:**
  - **Section 1 -Familiarity with R**
  - **Section 2 – Data Wrangling**
  - **Section 3 – Exploratory Data Analysis**
    - **Asking the right questions as a BA**
    - **Missing values**
    - **Diamond Exercise**
    - **Bank Marketing Exercise**
  - **Section 6 – Introduction to Regression**
  - **Section 7 – Introduction to Machine Learning**
    - **Titanic Kaggle Competition**

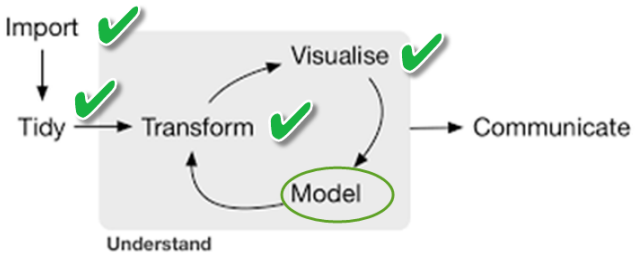
https://stat.ethz.ch/R-manual/R-devel/library/base/html/Memory-limits.html

Exploratory Data Analysis with R

Data Analysis objectives

1. Generate questions about the data

2. Search for answers by performing the following
   - Visualize
   - Transform
   - Model



Data Analysis start

- How to begin
  - You've got your data in a data.frame !
    - YAY!!!
  - Wait there's a lot more to do …
    - Resist the temptation to dive head first into your data pool
    - NO DATASET IS PERFECT
      - You may be missing data "NA"
      - Some data will be dirty
      - Some data will be inconsistent
  - Address your data issues early and often
  - We've learned in previous sessions on how to do this
    - How do we address them ?

Data Analysis considerations from the start

- This is the first step in the process
- Its exploratory
- More research and development than engineering, modeling or prediction
- This initial approach is more on strategy to get you started rather than choosing software, models or outcomes
- Big difference between data engineering and data scientist
- GOALS
    - As a business analyst – recognize what sort of analytic technique is appropriate for addressing a particular problem
    - Scope
    - Reduce uncertainty

The Human Factor

- AI will *not* rule us all!
  - The human factor in data science includes the ability to decompose a data analytics problem logically
    - Break into pieces
    - Recognize familiar problems and solutions
    - No need to recreate the wheel
  - All cannot be automated
- Data Science Analysis requires the following *human* traits
  - Creativity
  - Intelligence
  - Past Experiences
  - Critical Thinking
  - Proven and working methods

Business Understanding

- Understanding the data also means understanding…
  - The problem
    - What are you trying to solve
    - How long will it take
    - What's the $budget$?
  - The data
    - $Cost$
    - reliability
  - The business
    - Politics
    - Subject Matter Experts
  - The Stakeholders
    - RACI
      - Responsible
      - Accountable
      - Consulted
      - Informed
  - The impact
    - psychological
    - environmental
    - ethical

Exploratory Data Analysis and beginning to understand your data

- Data Exploration
  - Summary Statistics
    - summary()
    - str()
    - nrow()
    - Means / medians
    - Variances
    - Counts
  - Visualization
    - Histograms to get distributions
    - Finding those gotcha’s
    - Finding outliers
  - Joining Data
    - merge()
    - Adding additional data sources
    - Featurization

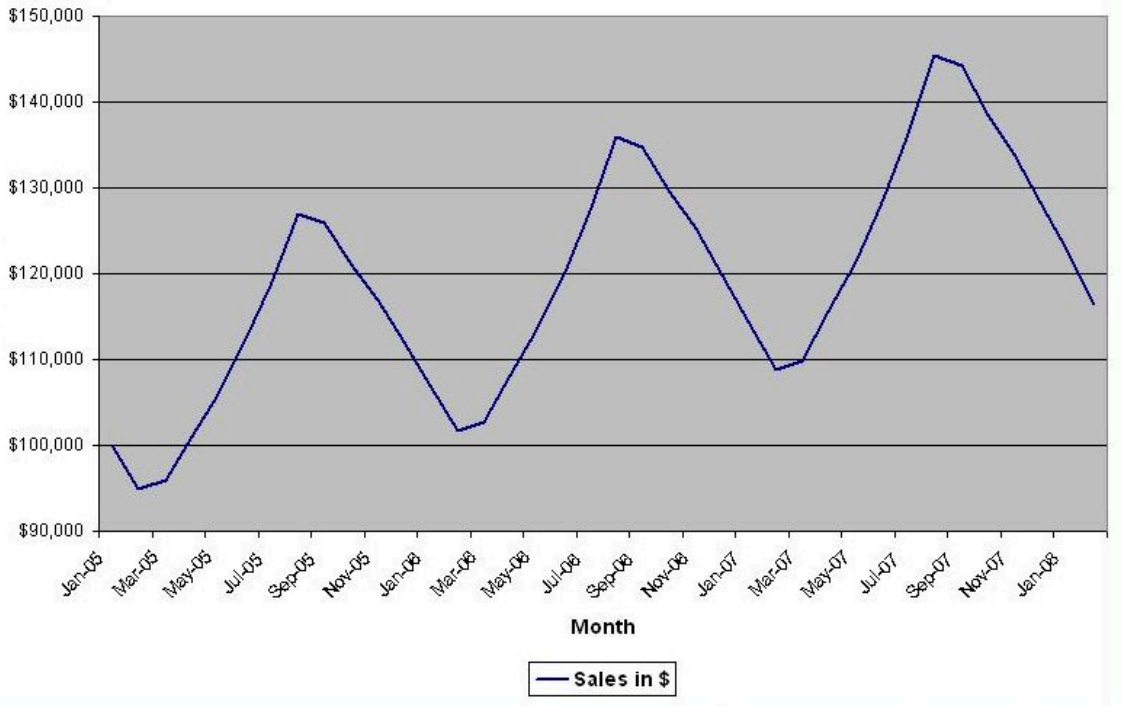
Strange and Missing Values

- If a particular field is immensely unpopulated then its worth finding WHY
  - You may want to drop these fields / variables / rows all together
  - Fill the NA's with zeros
  - Or take an average and use the average number smooth out the distribution

- Are there negative numbers that are throwing off the data?
  - Should a negative income be present
  - Are there outliers that shouldn't't be there
  - Are these outliers data entry errors ?
  - Are temperatures in both Celsius and Fahrenheit?
  - What is the unit of measurement?
  - Is there seasonality involved?
  - NORMALIZE, NORMALIZE, NORMALIZE

- Understand your range of values

Seasonality Example



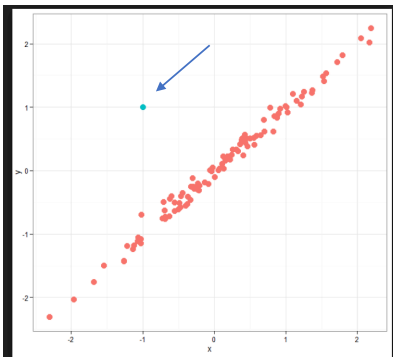
Data Analysis objectives

- Example questions to ask?

    1. Which values are the most common?
    2. Which values are rare
    3. Can you see any unusual patterns?
    4. Could this pattern be due to coincidence (random)
    5. How can you describe the relationship implied by the pattern?
    6. How strongly is the relationship implied by the pattern?
    7. What other variables might help the relationship?
    8. Clusters of similar values can suggest that subgroups, trends, commonalities or information from your data may exist

Data Analysis prep

- *Outliers* – observations that are unusual or data that stands out among the "crowd" or cluster of norm
  - Outliers could be
    - Data entry errors
    - New findings
  - Example
    - IP mapping data error by Maxmind
    - Zip codes 90210, 12345



Data Analysis prep

- *Missing values* – in R they are referred to "NA" or "not available." NA marks an unknown value

- The generic function is.na indicates which elements are missing.

- "NaN" means not a number and you may come across this in your data sets. If you see NaN as a value it could mean that there is a mix of categorical values and numerical values and R could not properly identify the value

Data Analysis prep

- Data Noise to Signal question
  - How can we segment the population with respect to something that we would like to estimate or even predict?

- *Data Mining* – finding or selecting important, informative attributes or variables of the entity described by the data

- *Information* – is a quantity that reduces uncertainty of the data

Data Analysis prep

- *Target Variable* – the attribute that will be the focus of the data problem to be solved. This manifests our perception of finding informative attributes.

    - This will lead to identifying one or more variables that reduces our uncertainty about the value of the target
    - Finding strongly related attributes that correlate with the target of interest will reduce uncertainty

- *Descriptive Modeling* – where the primary purpose of the model is not to estimate a value but instead to gain insight into the underlying real-world relationships between factors

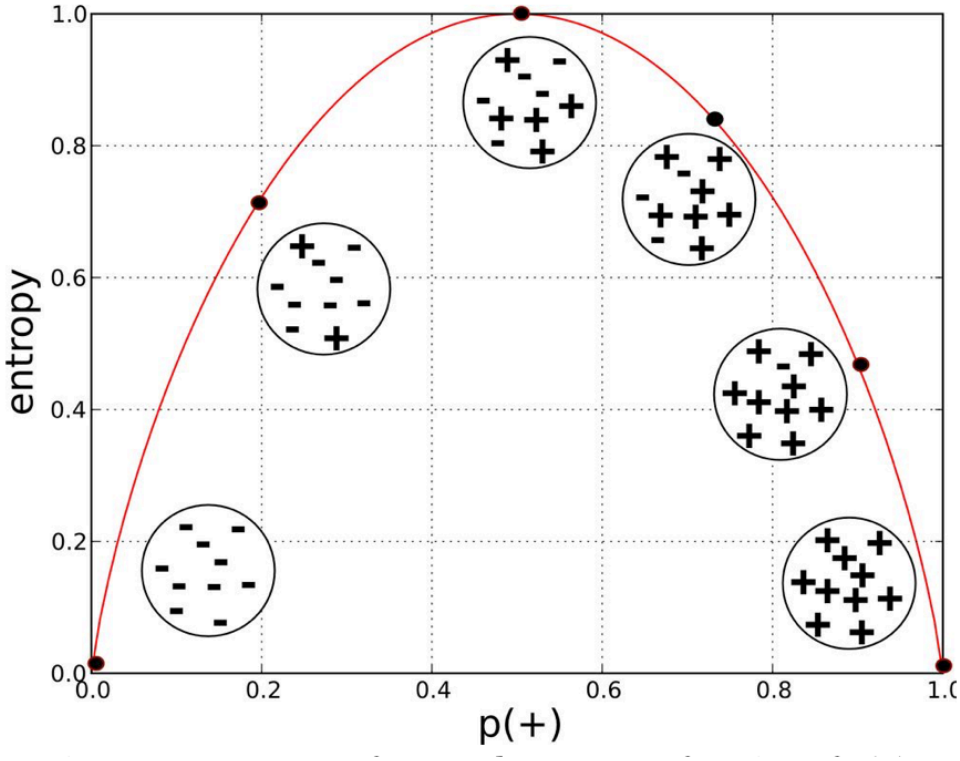
Data Analysis prep

- *Entropy* – measure of disorder that can be applied to a data set such as one of our individual segments

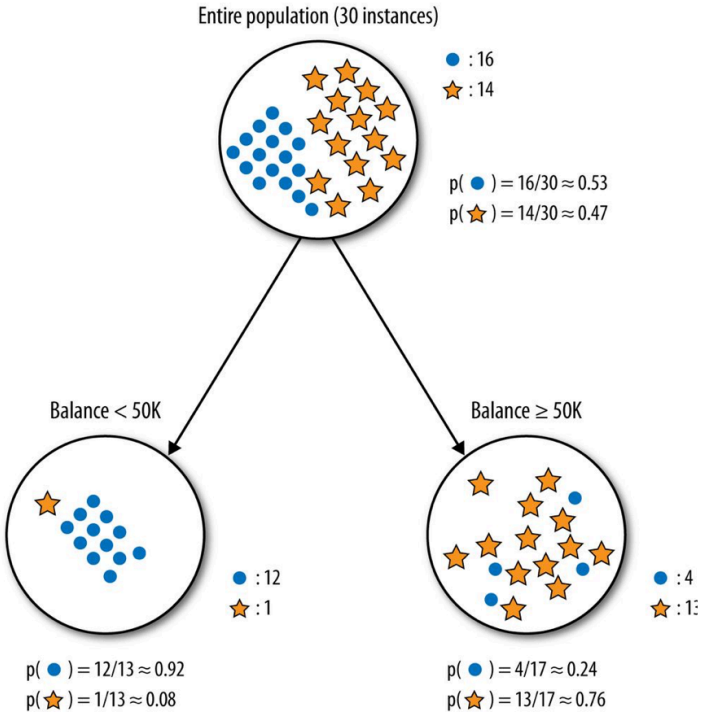
    - Disorder – corresponds to how mixed (impure) the segment is with respect to the properties of interest
    - 1 = pure
    - 0 = impure

$$entropy = -p_1 \log(p_1) - p_2 \log(p_2) - \cdots$$

Data Analysis prep



Data Analysis prep



Data Analysis prep

- *Information Gain* – how much an attribute improves (decreases) entropy over the whole segmentation it creates
  - Measures the change in entropy due to any count of new information being added

$$IG(parent, children) = entropy(parent) - [p(c_1) \times entropy(c_1) + p(c_2) \times entropy(c_2) + \cdots$$

Data Analysis prep

- *Variance* – the natural measure of impurity for numeric values
  - If the data set has all the same values for the numeric target variable then the data set is pure and the variance is 0
  - If the numeric target values in the set are very different then the set will have high variance

Formula - The **variance** (σ^2), is defined as the sum of the squared distances of each term in the distribution from the mean (μ), divided by the number of terms in the distribution (N).

Data Science In-class Discussion Exercise

- Consider the following set of hypothetical data science project questions to determine what the approach should be?

    - What is the problem?
    - How do we solve it?

Data Science Discussion Question 1

- Who are the most profitable customers?

Data Science Discussion Question 2

- What is the difference between the most profitable customers and the average customer?

Data Science Discussion Question 3

- Who are these customers? Can we characterize them?

Data Science Discussion Question 4

- Given a set criteria of customer attributes can we determine if this new customer will be profitable? How much revenue should I expect this customer to generate?

Key Takeaways

- Take the time to examine your data before diving head first into your data pool
- Summarize and understand your data first.
  - Identify data anomalies 1st
- Visualization gives you a sense of your data's distribution, relationships, outliers among all values and variables
- Visualization is an iterative process and helps answer questions about your data.
- Time spent in the data analysis phase is not time wasted.
- Data Analysis helps answer questions about the data

Business Analyst takeaways

- For a given project, the data analysis phase is a useful framework for analyzing a project or proposal
    - Understand whether the project is well conceived or is fundamentally flawed
    - What would make it successful from the start?
    - What would be an example of project pitfalls

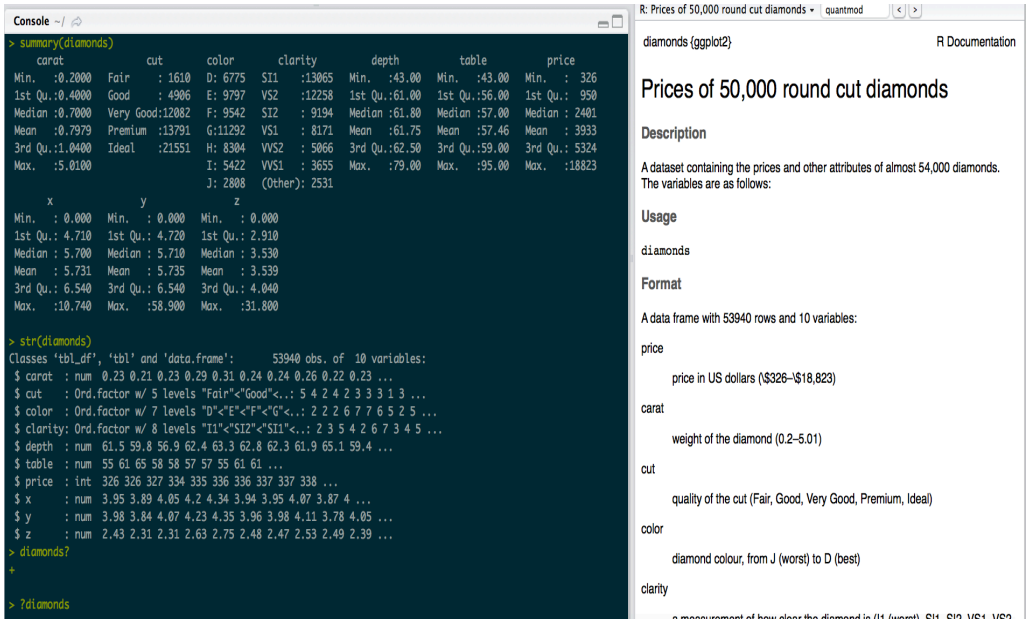
Data(diamond)

- Generate questions about the data

    - What questions do we have about the diamond data set?

        - What's the summary?
        - What are the features?
        - What type of data are we dealing with?
        - Are there any transformative features?
        - What is the structure of the data?

Data(diamond)



```
Console ~/
> summary(diamonds)
     carat               cut        color        clarity          depth      
 Min.   :0.2000   Fair      : 1610   D: 6775   SI1    :13065   Min.   :43.00  
 1st Qu.:0.4000   Good      : 4906   E: 9797   VS2    :12258   1st Qu.:61.00  
 Median :0.7000   Very Good:12082   F: 9542   SI2    : 9194   Median :61.80  
 Mean   :0.7979   Premium   :13791   G:11292   VS1    : 8171   Mean   :61.75  
 3rd Qu.:1.0400   Ideal     :21551   H: 8304   VVS2   : 5066   3rd Qu.:62.50  
 Max.   :5.0100                     I: 5422   VVS1   : 3655   Max.   :79.00  
                                    J: 2808   (Other): 2531                  
     table           price      
 Min.   :43.00   Min.   :  326  
 1st Qu.:56.00   1st Qu.:  950  
 Median :57.00   Median : 2401  
 Mean   :57.46   Mean   : 3933  
 3rd Qu.:59.00   3rd Qu.: 5324  
 Max.   :95.00   Max.   :18823  
                                
       x                y               z         
 Min.   : 0.000   Min.   : 0.000   Min.   : 0.000  
 1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.910  
 Median : 5.700   Median : 5.710   Median : 3.530  
 Mean   : 5.731   Mean   : 5.735   Mean   : 3.539  
 3rd Qu.: 6.540   3rd Qu.: 6.540   3rd Qu.: 4.040  
 Max.   :10.740   Max.   :58.900   Max.   :31.800  

> str(diamonds)
Classes 'tbl_df', 'tbl' and 'data.frame':	53940 obs. of  10 variables:
 $ carat  : num  0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
 $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
 $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...
 $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 ...
 $ depth  : num  61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
 $ table  : num  55 61 65 58 58 57 57 55 61 61 ...
 $ price  : int  326 326 327 334 335 336 336 337 337 338 ...
 $ x      : num  3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
 $ y      : num  3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
 $ z      : num  2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
> diamonds?
+

> ?diamonds
```

R: Prices of 50,000 round cut diamonds | quantmod

diamonds {ggplot2} R Documentation

Prices of 50,000 round cut diamonds

Description

A dataset containing the prices and other attributes of almost 54,000 diamonds. The variables are as follows:

Usage

```
diamonds
```

Format

A data frame with 53940 rows and 10 variables:

price

: price in US dollars (\$326–\$18,823)

carat

: weight of the diamond (0.2–5.01)

cut

: quality of the cut (Fair, Good, Very Good, Premium, Ideal)

color

: diamond colour, from J (worst) to D (best)

clarity

: a measurement of how clear the diamond is (I1 (worst), SI1, SI2, VS1, VS2

Exploratory Data Analysis in class exercise

- What diamond attributes have the strongest impact on price?

We'll explore the Diamond Data set to answer this question..and more

Hands on challenge

- Bank Marketing Data Set

- What are the strongest variables for the potential banking customer to become a client?
- Identify the following:
  - Percentage of success with telemarketing currently (yes)
  - Are there NA's? How would you fill the Nas?
  - summary statistics
  - descriptive statistics
  - Outliers
- What are the ranges in education type?
- How can we improve the business given the data set

http://archive.ics.uci.edu/ml/datasets/Bank+Marketing

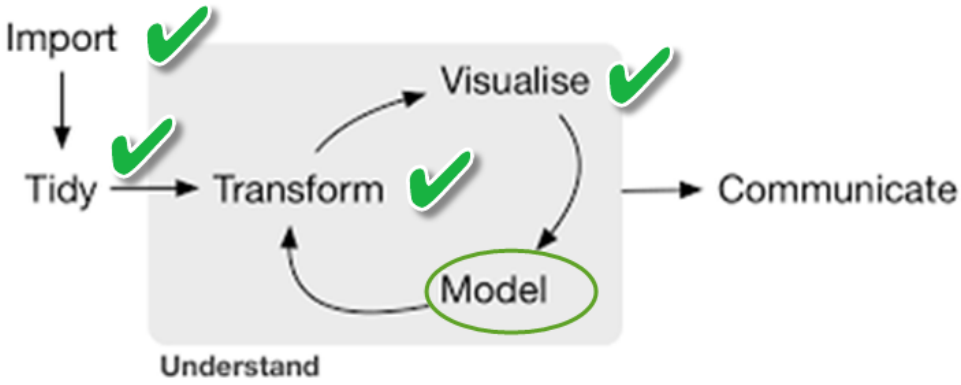
[Moro et al., 2014] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014

Section 6 Introduction to Regression

- Online Course Layout:
  - Section 1 -Familiarity with R
  - Section 2 – Data Wrangling
  - Section 3 – Data Visualization
  - Section 4 – R Markdown
  - Section 5 – Exploratory Data Analysis
  - Section 6 – Introduction to Regression
    - Definition
    - The regression line
    - The formula
    - Auto Example of regression
    - Progresso soup exercise
  - Section 7 – Introduction to Machine Learning
    - Titanic Kaggle Competition
  - Section 8 – Strategy
    - Big box store competitors

https://stat.ethz.ch/R-manual/R-devel/library/base/html/Memory-limits.html

Next up…model



http://r4ds.had.co.nz/intro.html

*Hadley Wickham is from New Zealand and that's how they spell "visualise"

Introduction to Regression with R

Regression definition

- Provides a statistical analysis of the relationship between a dependent variable and a set of independent or exploratory variables

    - Used often to describe or predict a numerical value

Regression examples

- Business Use case for Regression

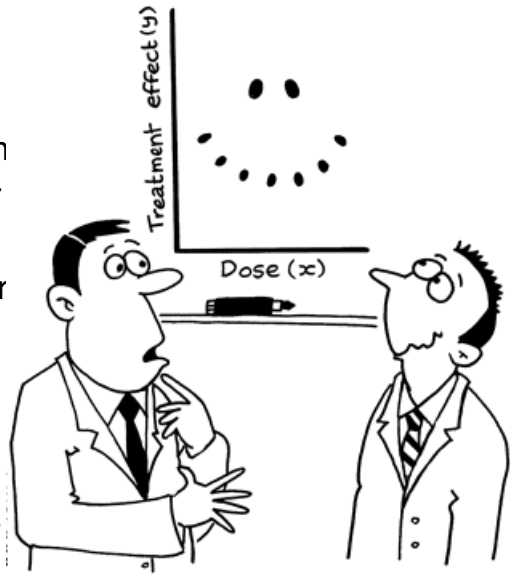
- used in targeted marketing

    - How much will a given customer use this service
    - List of consumers ranked by their predicted likelihood of responding positively by an offer, coupon, discount, survey, advertisement

    - Goal = obtain exact probability of response

Regression is similar to polling

- If we have a large representative sample and solid methodology then the relationship we observe for our sample data is not likely to deviate wildly from the true relationship for the whole population
- We can isolate the statistical relationship that we care about that take into account other factors that might confuse the relationship



"It's a non-linear pattern with outliers.....but for some reason I'm very happy with the data."

Regression Definition:

A regression is a statistical analysis assessing the association between two variables. In simple linear regression, a single independent variable is used to predict the value of a dependent variable.

Regression Formula:

Regression Equation(y) = a + bx

Slope(b) = $(N\Sigma XY - (\Sigma X)(\Sigma Y)) / (N\Sigma X^2 - (\Sigma X)^2)$

Intercept(a) = $(\Sigma Y - b(\Sigma X)) / N$

Where,

x and y are the variables.

b = The slope of the regression line

a = The intercept point of the regression line and the y axis.

N = Number of values or elements

X = First Score

Y = Second Score

ΣXY = Sum of the product of first and Second Scores

ΣX = Sum of First Scores

ΣY = Sum of Second Scores

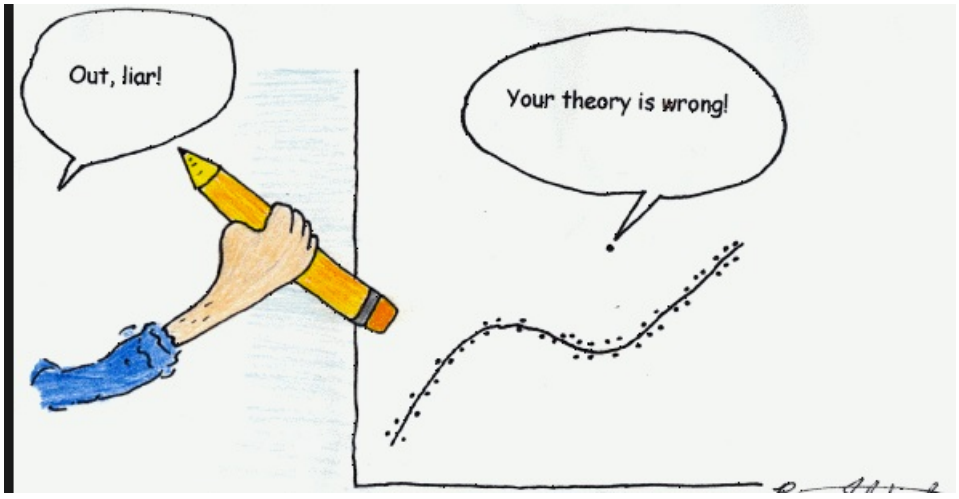
ΣX^2 = Sum of square First Scores

Regression mathematical definition

- Objective for the computer running the mathematical regression equation:
  - Find the minimum sum of errors
    - If I predict 10 and the actual value is 12 or 8 then the error (or disturbance) is 2

Regression drawback

- Very sensitive to the data
- Erroneous or otherwise outlier data points can severely skew the result of the linear function
- Relatively easy to use but hard to use well – and potentially dangerous when used improperly



Variables

- Dependent variable - a variable (denoted by y in the regression formula) whose values depend on that of another

- Independent Variable – a variable (denoted by x) who variation does not depend on that of another

- Dummy variable – a variable (usually manually) created that takes a value of 0 or 1 to indicate the presence or absence of a categorical effect that may have changes to the outcome

Training vs Test

- Training set is used to build the model
- Test set is used to estimate the models predictive performance
- Vocabulary Clarification
- Predictor Variables = Independent Variables

hypothesis and using regression

- People who are taller tend to weigh more

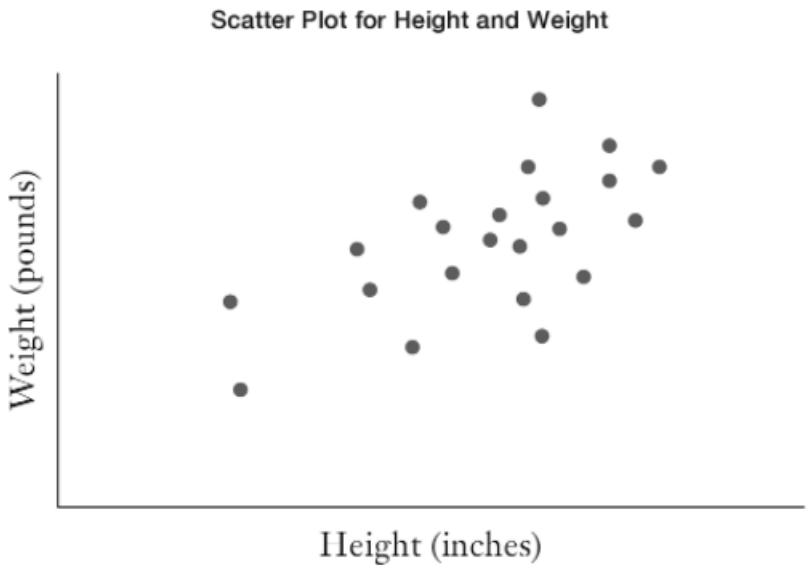
Pattern

- Weight seems to increase with height

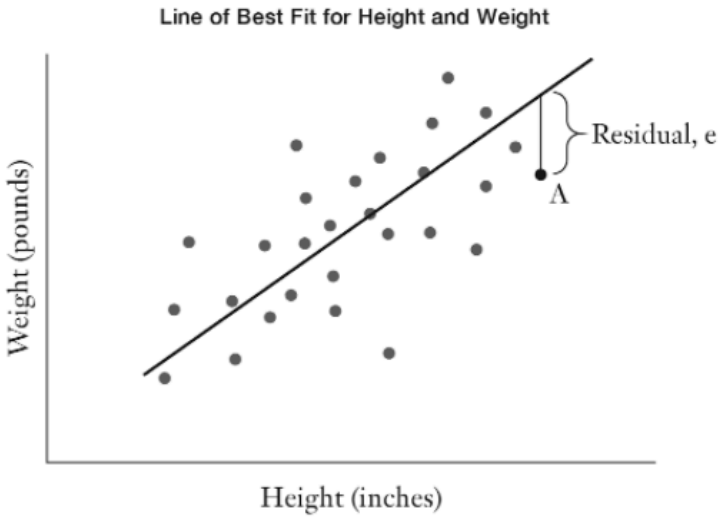
Regression

- Fit a line that best describes a linear relationship between two variables
    - Weight (dependent variable)
    - Height (independent or explanatory variable)

Scatter Plot of Weight (y) and Height (x)



Scatter Plot of regression line

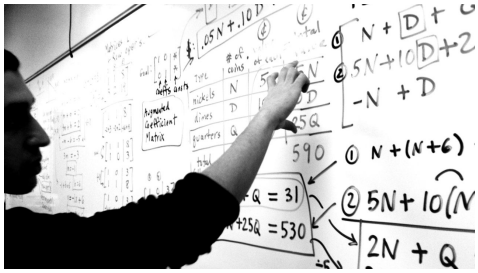


But how did you get that mysterious line?

- Regression uses a methodology called "**O**rdinary **L**east **S**quares"
  - **OLS –** fits the line that minimizes the sum of the square residuals

- **Residuals –** vertical distance from the regression line except for those observations that directly lie on the line and those residuals = 0.
  - The larger the sum of residuals overall the worse fit of the line

Back to the formula …



- Regression equation:
- Y = a + bX

- Y = weight in pounds
- a = intercept at the line
- b = slope of the line
- x = height in inches

- b = describes the best linear relationship between height and weight as defined by ordinary least squares
- Weight = a + b(height) + e
- e – residual that catches the variation in weight for each individual that is not explained by height

Fit

- **Fitting Linear Models**
- **Description**
  - lm is used to fit linear models. It can be used to carry out regression, single stratum analysis of variance and analysis of covariance
- **Usage**
  - lm(formula, data, subset, weights, na.action, method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE, contrasts = NULL, offset, ...)

Best guess or ….prediction?

- Best guess for weight of a person
- = a + b (height)
- Weight is known as the dependent variable because it depends on other factors

- Explanatory variables or independent variables
  - Variables used to explain our dependent variable since they explain the outcome we care about

Best guess or ….prediction?

- Weight in pounds = -350.73719 + (7.71729) x 71 in inches
- Y = a + bx

- a = -350.73719 or the intercept
  - No particular meaning on its own
- b = 7.71729 or the regression coefficient
  - Gives the best estimate or the relationship between height and weight
  - A one unit increase in the independent variable (height) is associated with an increase of units in 7.71729 the dependent variable (weight)
- 71 inches is the example we will use

Simple Regression

- When there is only one predictor variable, we refer to the regression model as **SIMPLE LINEAR REGRESSION MODEL**

Put the formula to a test

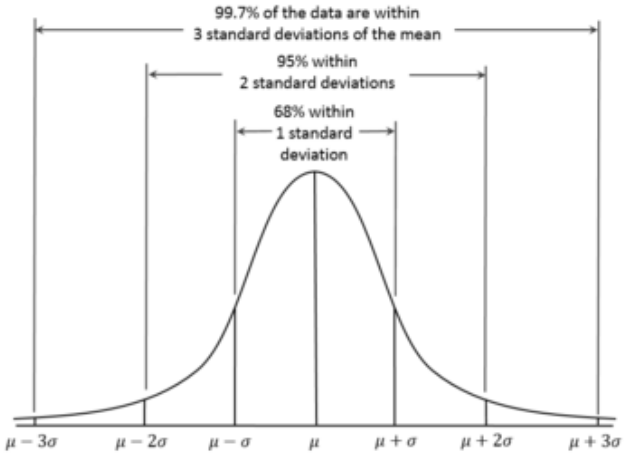
- Weight in pounds = # + (#) x height in inches
- Example = is 5 feet 11 (71 inches)
- # + #(EXAMPLE) = ?lbs

Breaking down the coefficient

- Sign – (positive or negative)
  - Tells us the direction of its associate with the dependent variable
    - The outcome we are trying to explain
      - Taller people weigh more
- Significance
  - Does the value reflect a meaningful association that is likely to be observed for the population as a whole ?
  - Standard Error – measure of the likely dispersion we would observe in the coefficient if we were to conduct the regression analysis on repeated samples drawn from the same population

Breaking down the coefficient

- Distribution is more dispersed than the normal distribution and therefore has fatter tails
  - Smaller samples tend to generate more dispersion in outcomes



More on the standard error

- If we were to do this analysis repeatedly – say 100 different samples – then we expect our observed regression coefficient to be within 2 standard errors of the true population parameters or roughly 95 times out of 100

- It is a measurement of correlation and not accuracy and is dependent on the variation in the outcome

- R square = 0.75 implies that the model can explain 3 quarters of the variation in the outcome

- Rule of Thumb

    - Weight example:
    - Standard error .13
    - Coefficient 4.5
        - The coefficient is likely to be statistically significant when the coefficient is at least 2x the size of the standard error

R squared

- Measure of the total amount of variation explained by the regression equation
  - Tells us how much of that variation around the mean is associated with the differences in height alone
  - R-squared is always between 0 and 100%:
    - 0% indicates that the model explains none of the variability of the response data around its mean.
    - 100% indicates that the model explains all the variability of the response data around its mean.

T-Test

- In order to compare two samples to see if they have different means its best to use the t.test function.
- Feed R a formula of data to compare
- In R this can be done by using the ‘~’ (tilde) or “explained by”
- Do the predicted value depend on the dependent variable

P-value

- The P value shows the probability that this apparent difference between the two variables (groups) could appear by chance
  - A low p-value (<%5) means we can be fairly confident that there is actual difference between the variables

Root Mean Square Error

- When the outcome is a number, the most common method for characterizing a models predictive capabilities is to use Root Mean Square Error (RMSE)

$$R = \sqrt{\frac{(P-W)^2}{N}}$$

RMSE is: where P is the predicted weight and W is the verification weight.

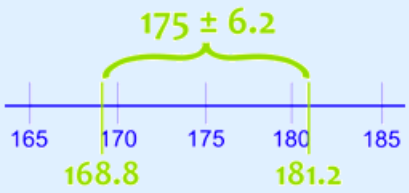
  - Calculated by taking the square root of the Mean Square Error so that it is the same units as the original value
  - The value is usually interpreted as either how far (on average) the residuals are from zero or the average distance between the observed values and the model predictors

Weight and Height Code

Confidence Interval

- A range of values we are fairly sure our true value lies in
  - Example:
    - We want to measure the heights of 40 randomly chosen men
    - Mean height of 175cm
    - Standard Deviation of 20cm
      - If we use a 95% Confidence Interval
      - Then the true mean of ALL men (if we could measure their heights) is likely to be between 168.8cm and 181.2cm
      - The 95% confidence interval means that 95% of the experiments like we did will include the true mean %5 will not be included
      - So there is a 1-in-20 chance (%5) that our Confidence Interval does NOT include the true mean

175cm ± 6.2cm



- Number of samples: **n = 40**
- Mean: $\bar{X}$ **= 175**
- Standard Deviation: **s = 20**

	z
95%	1.960

$$175 \pm 1.960 \times \frac{20}{\sqrt{40}}$$

Which is:

175cm ± 6.20cm

In other words: from 168.8cm to 181.2cm

Correlation

- Looking at data relation is a sensible step to understand how your different variable interact together.
- Correlation looks at trends shared between two variables, and regression looks at causal relation between a predictor (independent variable) and a response (dependent) variable.
- Correlations look at global movement shared between two variables, for example when one variable increases and the other increases as well, then these two variables are said to be positively correlated.

Correlation formula

In the formula below,

- **x** and **y** are two vectors of length **n**
- m_x and m_y corresponds to the means of x and y, respectively.

Pearson correlation formula

$$r = \frac{\sum (x - m_x)(y - m_y)}{\sqrt{\sum (x - m_x)^2 \sum (y - m_y)^2}}$$

m_x and m_y are the means of x and y variables.

The p-value (significance level) of the correlation can be determined :

1. by using the correlation coefficient table for the degrees of freedom : $df = n - 2$, where n is the number of observation in x and y variables.
2. or by calculating the **t value** as follow:

$$t = \frac{r}{\sqrt{1 - r^2}} \sqrt{n - 2}$$

In the case 2) the corresponding p-value is determined using **t distribution table** for $df = n - 2$

✔ If the p-value is < 5%, then the correlation between x and y is significant.

Correlation function

```r
# Load data
data("mtcars")
my_data <- mtcars[, c(1,3,4,5,6,7)]
# print the first 6 rows
head(my_data, 6)
```

```
                   mpg disp  hp drat    wt  qsec
Mazda RX4         21.0  160 110 3.90 2.620 16.46
Mazda RX4 Wag     21.0  160 110 3.90 2.875 17.02
Datsun 710        22.8  108  93 3.85 2.320 18.61
Hornet 4 Drive    21.4  258 110 3.08 3.215 19.44
Hornet Sportabout 18.7  360 175 3.15 3.440 17.02
Valiant           18.1  225 105 2.76 3.460 20.22
```

Correlation function

Compute correlation matrix

```
res <- cor(my_data)
round(res, 2)
```

```
       mpg  disp    hp  drat    wt  qsec
mpg   1.00 -0.85 -0.78  0.68 -0.87  0.42
disp -0.85  1.00  0.79 -0.71  0.89 -0.43
hp   -0.78  0.79  1.00 -0.45  0.66 -0.71
drat  0.68 -0.71 -0.45  1.00 -0.71  0.09
wt   -0.87  0.89  0.66 -0.71  1.00 -0.17
qsec  0.42 -0.43 -0.71  0.09 -0.17  1.00
```

In the table above **correlations coefficients** between the possible pairs of variables are shown.

Note that, if your data contain missing values, use the following R code to handle missing values by case-wise deletion.

Correlation Plots

Use corrplot() function: Draw a correlogram

The function **corrplot**(), in the package of the same name, creates a **graphical** display of a correlation matrix, highlighting the most correlated variables in a data table.

In this plot, correlation coefficients are colored according to the value. Correlation matrix can be also reordered according to the degree of association between variables.

- **Install corrplot**:

```
install.packages("corrplot")
```

- **Use corrplot**() to create a **correlogram**:

The function **corrplot()** takes the **correlation matrix** as the first argument. The second argument (type="upper") is used to display only the upper triangular of the **correlation matrix**.

```
library(corrplot)
corrplot(res, type = "upper", order = "hclust",
         tl.col = "black", tl.srt = 45)
```

Multiple regression analysis

- Now we decide to add an additional explanatory variable for weight
  - GENDER

- New formula
  - Weight = -# + # X (height in inches) + # X (age in years)
    - .1 = coefficient
      - For each Gender type is associated additional pounds in weight, holding height constant

  - Test out the formula

In class example

- Data("mtcars")
  - Which transmission type gets better miles per gallon efficiency?
    - Manual?
    - Automatic?
  - Explain with a linear regression model

- Identify a business problem to solve with the mtcars data set
  - Build a cor plot
  - Which features are strongest in relation to your business problem?

?mtcars

mtcars {datasets} R Documentation

Motor Trend Car Road Tests

Description

The data was extracted from the 1974 *Motor Trend* US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models).

Usage

```
mtcars
```

Format

A data frame with 32 observations on 11 variables

Progresso soup exercise

Lets take the perspective of Progresso

- Start by summarizing the data
- Demographics
- Marketing Mix
- (Shares/Price/Promotion)
- Seasonality
- Strong Markets/Periods
- Regression Based Models to understand elasticity
- Recommendations for growth

```r
library(coefplot)
library(ggplot2)

setwd("~/Progresso")
PS<-read.csv(file="Progresso_Soup.csv", header=TRUE, sep=",")

# Create Dummy Variables for Winter Months
PS$Winter <- PS$Month_Jan_Dec
PS$Winter<- ifelse(PS$Winter > 2 & PS$Winter < 10, c("notwinter"), c("winter"))


PS$Incomelvl <- ifelse(PS$High_Income > 0, c("highincome"), c("lowincome"))


# market share of Progresso for each store-month
PS$progmk <- PS$sales_prog / PS$Category_sales

# Box plots of market shares across all stores

ggplot(PS, aes(x=Winter, y=progmk, fill=Winter))+geom_boxplot ()

ggplot(PS, aes(x=Region, y=progmk, fill=Winter))+geom_boxplot ()
```

```r
# Aggregate sales to get market shares
cat<-aggregate(Category_sales~Winter, data=PS, FUN=sum)
prog1<-aggregate(sales_prog~Winter, data=PS, FUN=sum)
prog_sales<-prog1[,2]
agshare<-cbind(cat,prog_sales)
agshare$progmk <- agshare$prog_sales / agshare$Category_sales
agshare



# Linear Regression with just prices
reg1 <- lm(Log_sales.Progresso~Log_Price.Progresso+Log_Price.Campbell +Log_Price.PL, data=PS)
summary(reg1)
coefplot(reg1)


# Add control for Seasonality
reg2 <- lm(Log_sales.Progresso~Log_Price.Progresso+Log_Price.Campbell +Log_Price.PL+Winter, data=PS)
summary(reg2)
coefplot(reg2)


# Add control for REGIONS
reg3 <- lm(Log_sales.Progresso~Log_Price.Progresso+Log_Price.Campbell +Log_Price.PL+Winter+Region, data=PS)
summary(reg3)
coefplot(reg3)
```

-Summary-

-Seasonal demand for category as a whole

-Market shares for Progresso strong seasonal patterns as well

-Mostly driven by Price promotions in Winter months

-Able to make substantial gains in some markets but not others

Hands on exercise

- Business Use Case Studies

Section 7 ML

- **Online Course Layout:**
  - **Section 1 -Familiarity with R**
  - **Section 2 – Data Wrangling**
  - **Section 3 – Data Visualization**
  - **Section 4 – R Markdown**
  - **Section 5 – Exploratory Data Analysis**
  - **Section 6 – Introduction to Regression**
    - **Definition**
    - **The regression line**
    - **The formula**
    - **Auto Example of regression**
    - **Progresso soup exercise**
  - **Section 7 – Introduction to Machine Learning**
    - **Types of ML**
    - **CRISP Modeling**
    - **Evaluation**
    - **Kaggle Titanic**
    - **Decision Trees Prediction**
    - **German Credit Exercise**

https://stat.ethz.ch/R-manual/R-devel/library/base/html/Memory-limits.html

Introduction to Machine Learning

Agenda

- Understand the machines learning
- Types of ML
- CRISP
- Modeling
- Evaluation
- Kaggle Titanic
- Exploratory analysis
- Dplyr
- Decision Trees
- Prediction

Understanding the Machines and Learning

- In order for us to imply machine learning, we must learn how the machines learn

- Machine Learning definition:
  - Explores the study and construction of algorithms that can learn from and make predictors on data

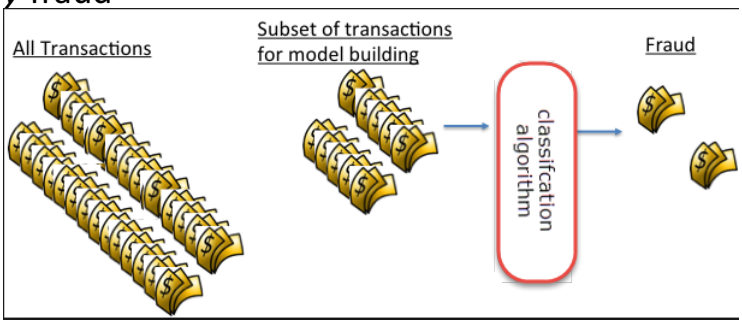
Supervised vs Unsupervised Machine Learning

- Supervised
  - Based on a well-defined target variable and the test data include labels for the target variable

- Unsupervised
  - May apply when one cant satisfy a target variable
  - Unsupervised data mining is often used for exploratory data analysis

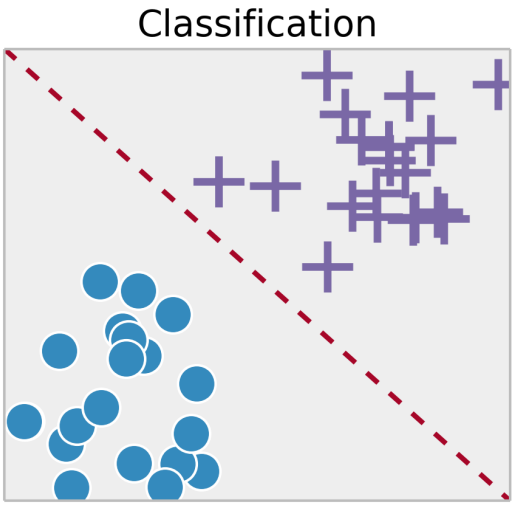
Types of ML – Classification

- Classification
  - Has a categorical target or target variable
  - The output variables takes class labels
  - Example
    - Fraud detection – credit card transactions come through and one comes in as fraudulent. Use ML to *classify* fraud

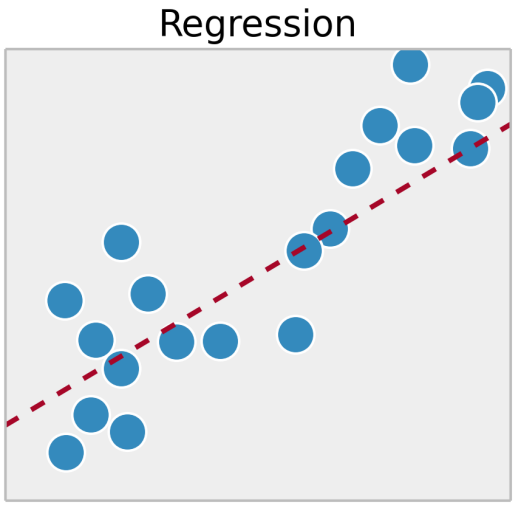


Types of ML – Regression

- Regression
  - An estimation used to observe the relationship between variables as their values change
  - The output variable takes continuous values
  - Example
    - Fraud detection – what is the percentage likelihood that this transaction is fraud



FRAUD vs NOT FRAUD



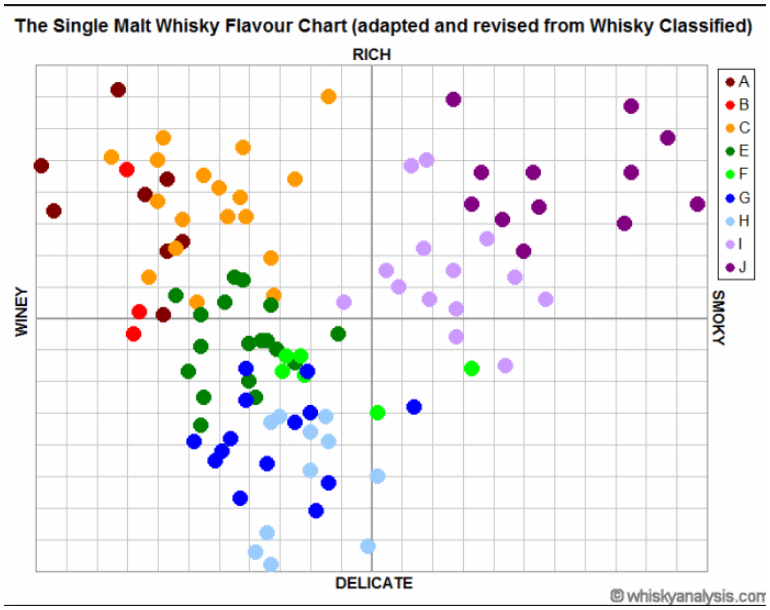
All transactions with a linear discriminant to identify the likelihood of fraud (on the line)

Types of ML – Clustering

- Clustering
  - Used to find groupings of similar feature vectors
  - Defines a similarity/distance metric between objects
  - Popular in exploratory analysis

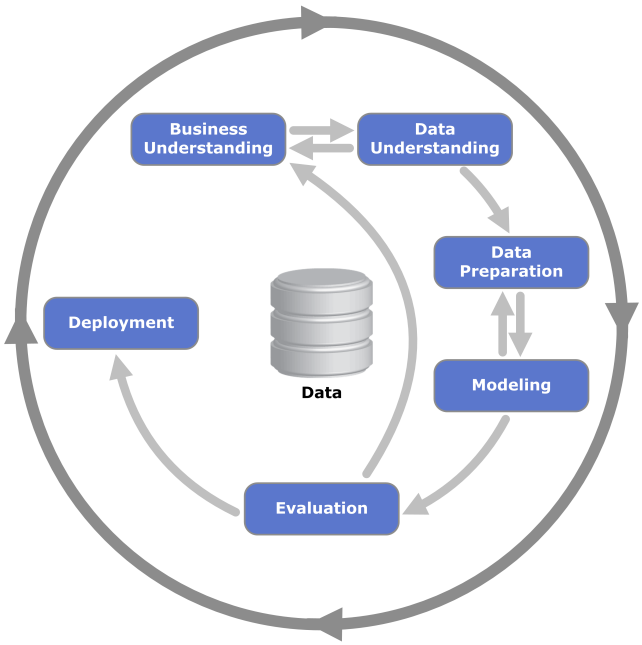
    - Example:
      - If I like a particular malt whisky, what other whiskies might I also enjoy?

Types of ML – Clustering

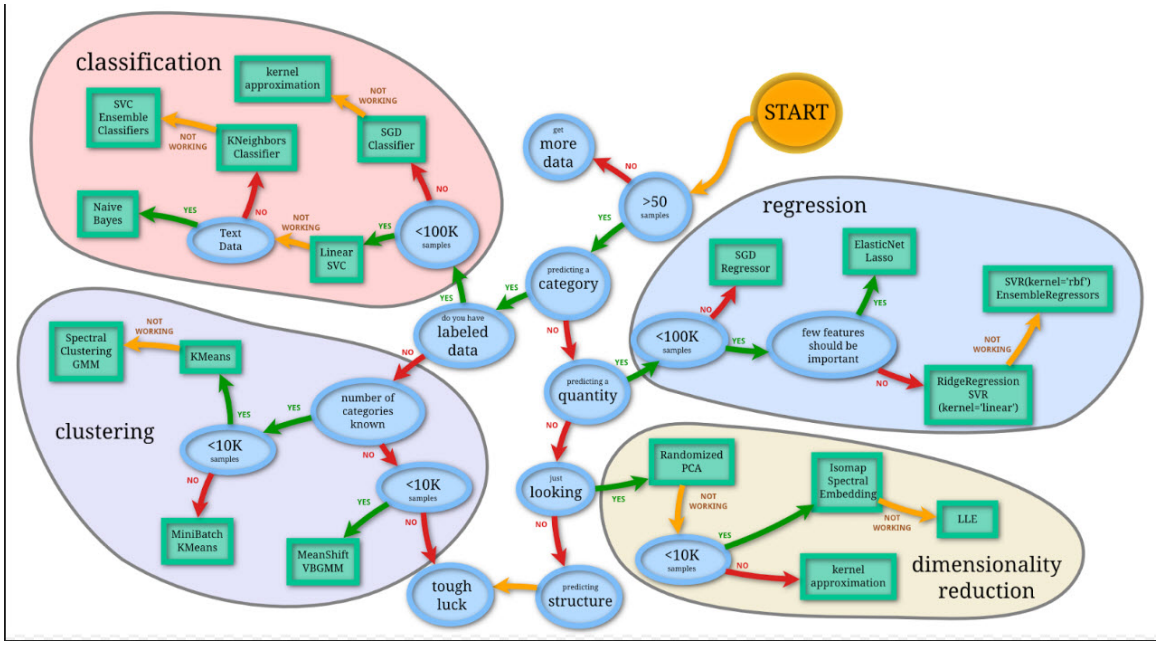


CRISP-DM

Cross Industry Standard Process for Data Mining



Model selection guide



List of Common Machine Learning Algorithms

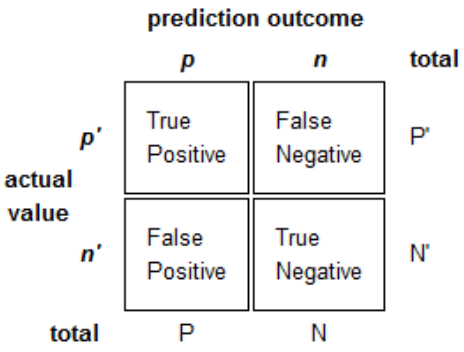
Here is the list of commonly used machine learning algorithms. These algorithms can be applied to almost any data problem:

1. Linear Regression
2. Logistic Regression
3. Decision Tree
4. SVM
5. Naive Bayes
6. KNN
7. K-Means
8. Random Forest
9. Dimensionality Reduction Algorithms
10. Gradient Boosting algorithms
    1. GBM
    2. XGBoost
    3. LightGBM
    4. CatBoost

https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/

Model Evaluation

- Cross Validation with the holdout method –
  - The data set is separated into two sets, called the training set and the testing set.
  - The function approximator fits a function using the training set only.
  - Then the function approximator is asked to predict the output values for the data in the testing set
  - The errors it makes are accumulated as before to give the mean absolute test set error, which is used to evaluate the model

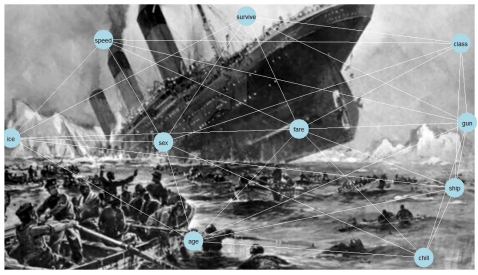


https://www.cs.cmu.edu/~schneide/tut5/node42.html

The Titanic Kaggle Challenge!

https://www.kaggle.com/c/titanic

- In the proceeding slides we will be using the Introductory Kaggle Titanic example
- Can we predict who survived or perished the maiden voyage of the Titanic that sank after colliding with an iceberg and killing 1502 out of 2224 passengers and crew?
- Some groups of people, like women, children and those with upper class tickets.
- Our challenge is to complete the analysis using machine learning and prediction algorithms to predict which passengers survived the tragedy.



Lets begin our voyage

- In machine learning there are two types of datasets to use when predicting
  - Training Data – complete with the outcome (or target variable). In this prediction problem our data will be the group of passengers as well as a collection of other features such as the following:
  - Test Data or test set –
    - Which you must predict the now unknown target variable based on the other passenger attributes that are provided for both datasets

Variable Name	Description
Survived	Survived (1) or died (0)
Pclass	Passenger's class
Name	Passenger's name
Sex	Passenger's sex
Age	Passenger's age
SibSp	Number of siblings/spouses aboard
Parch	Number of parents/children aboard
Ticket	Ticket number
Fare	Fare
Cabin	Cabin
Embarked	Port of embarkation

More on the variables

```
VARIABLE DESCRIPTIONS:
survival        Survival
                (0 = No; 1 = Yes)
pclass          Passenger Class
                (1 = 1st; 2 = 2nd; 3 = 3rd)
name            Name
sex             Sex
age             Age
sibsp           Number of Siblings/Spouses Aboard
parch           Number of Parents/Children Aboard
ticket          Ticket Number
fare            Passenger Fare
cabin           Cabin
embarked        Port of Embarkation
                (C = Cherbourg; Q = Queenstown; S = Southampton)

SPECIAL NOTES:
Pclass is a proxy for socio-economic status (SES)
 1st ~ Upper; 2nd ~ Middle; 3rd ~ Lower

Age is in Years; Fractional if Age less than One (1)
 If the Age is Estimated, it is in the form xx.5

With respect to the family relation variables (i.e. sibsp and parch)
some relations were ignored.  The following are the definitions used
for sibsp and parch.

Sibling:  Brother, Sister, Stepbrother, or Stepsister of Passenger Aboard Titanic
Spouse:   Husband or Wife of Passenger Aboard Titanic (Mistresses and Fiances Ignored)
Parent:   Mother or Father of Passenger Aboard Titanic
Child:    Son, Daughter, Stepson, or Stepdaughter of Passenger Aboard Titanic

Other family relatives excluded from this study include cousins,
nephews/nieces, aunts/uncles, and in-laws.  Some children travelled
only with a nanny, therefore parch=0 for them.  As well, some
travelled with very close friends or neighbors in a village, however,
the definitions do not support such relations.
```

Objective 1

- Perform EDA on Titanic Data Set

    1. Combine the test and data set into one separate file
    2. Identify NA's or missing values
        - Substitute values
        - Average values
        - NA values
    3. Perform EDA on the data set
    4. What is the proportion of survived to perished?
    5. What is the proportion of males to females?
    6. What is the proportion of children to adults?
        1. You will have to create a new column for this feature (call it Child)
    7. What are the Important features?
        - Hint: Correlation & corrplot

- Extra Credit –
    - Create a new feature
        - Hint:
            - AGE
            - FARE

Examine the training data

```
#prediction and titanic example
#In this competition, you must predict the fate of the passengers aboard the RMS Titanic
#Did the theory of "women and children first" actually happen on the Titanic ?
#https://www.kaggle.com/c/titanic/data?train.csv


#set working diriectory for where titanic files training and test sets are stored
setwd("F:/R_code/Intro Class/titanic")


# By default R imports all text strings as factors but we dont want all of them as factors right now
train <- read.csv("train.csv", stringsAsFactors=FALSE)
test <- read.csv("test.csv", stringsAsFactors=FALSE)

#find the structure of the import
str(train)
str(test)

#isolate survivors [1 is survived]
table(train$Survived)

#proportion of survivors
prop.table(table(train$Survived))

#everyone dies prediction only 418 rows in the test set
nrow(test)
test$Survived <- rep(0, 418)


#everyone dies prediction
test$Survived <- rep(0, 418)

#submit a csv file with the PassengerId as well as our Surived predictions
submit <- data.frame(PassengerId = test$PassengerId, Survived = test$Survived)
#write.csv(submit, file = "theyallperish.csv", row.names = FALSE)
```

Using dplyr to explore the Titanic

```r
#dplyr explore the data
library(dplyr)

#filter on female and create new dataframe "females"
#filter() allows you to select a subset of rows in a data frame. The first argument is the name of the data frame.
#The second and subsequent arguments are the expressions that filter the data frame
females <- filter(train, Sex == "female")

#select first 5 rows using slice
slice5 <- slice(train, 1:5)

#arrange passengers by fare price (acending order)
fare_arranged <- arrange(train, Fare)

#arrange by descending order
fare_desc <- arrange(train, desc(Fare))

#get distinct Pclass types
distinct(train, Pclass)

# avearge Pclass of all the passengers
summarise(train, average.class = mean(Pclass, na.rm=TRUE))

# max price for a fare
summarise(train, max_fare = max(Fare, na.rm=TRUE))

# max age
summarise(train, max_age = max(Age, na.rm=TRUE))
```

Looking for further variables

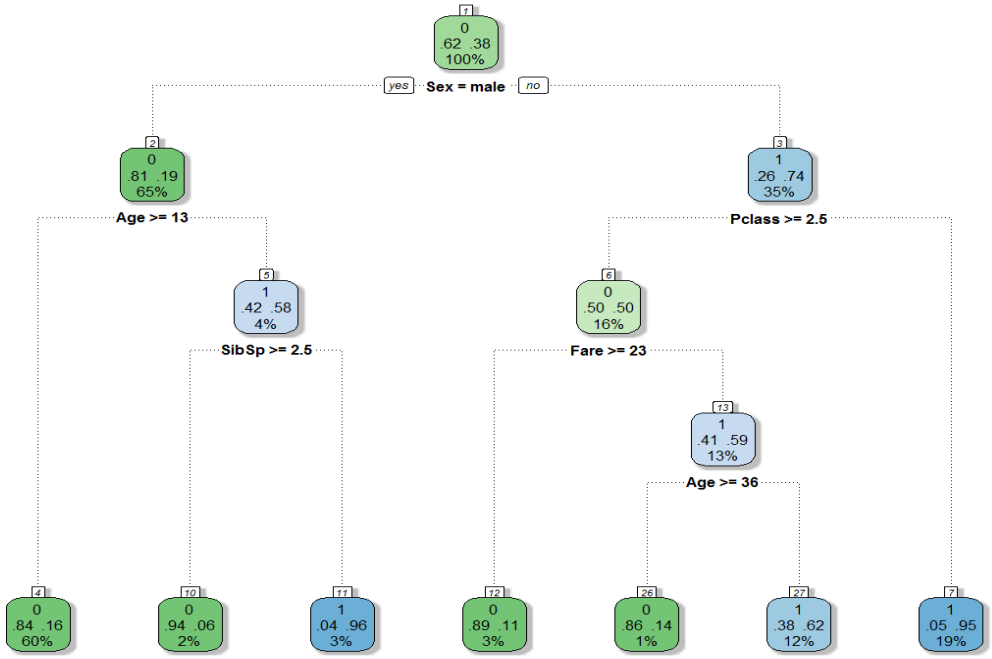
```
#disaster was famous for saving "women and children first"
summary(train$Sex)

#proportion of sex and survivors 1 is survived
#we need to tell the command to give us proportions in the 1st dimension which stands for the rows
prop.table(table(train$Sex, train$Survived),1)

#update our prediction to all females suriving
test$Survived <- 0
test$Survived[test$Sex == 'female'] <- 1

#now on to test to see if all the children survived as well
summary(train$Age)

#label the children with a binary of 0 not child and 1 as child
train$Child <- 0
train$Child[train$Age < 18] <- 1

#find the number of survivors for the different subsets
#The aggregate command takes a formula with the target variable on the left hand side
# of the tilde symbol and the variables to subset over on the right
aggregate(Survived ~ Child + Sex, data=train, FUN=sum)

#total number of people in each subset
#length of the survived vector for each subset and output the result
aggregate(Survived ~ Child + Sex, data=train, FUN=length)

#We need to create a function that takes the subset vector as input and applies
#both the sum and length commands to it, and then does the division to give us a proportion
aggregate(Survived ~ Child + Sex, data=train, FUN=function(x) {sum(x)/length(x)})

#another feature variable is costs of ticket
# we'll bin the fares
train$Fare2 <- '30+'
train$Fare2[train$Fare < 30 & train$Fare >= 20] <- '20-30'
train$Fare2[train$Fare < 20 & train$Fare >= 10] <- '10-20'
train$Fare2[train$Fare < 10] <- '<10'
```

```
#longer aggregate function to see if there's anything interesting to work with here
aggregate(Survived ~ Fare2 + Pclass + Sex, data=train, FUN=function(x) {sum(x)/length(x)})

test$Survived <- 0
test$Survived[test$Sex == 'female'] <- 1
test$Survived[test$Sex == 'female' & test$Pclass == 3 & testFare >= 20] <- 0
```

Prediction code with decision trees

```r
#decsion trees
library(rpart)


#rpart command works similarly to the aggregate function
#You feed it the equation, headed up by the variable of interest and followed by the variables used for prediction
fit <- rpart(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked,
             data=train,
             method="class")
#boring but plot of the decision tree
plot(fit)
text(fit)

#new and exciting packages to make this plot pop
library(RGtk2)
library(rattle)
library(rpart.plot)
library(RColorBrewer)

fancyRpartPlot(fit)

#new prediction with decision tree @ 78%
Prediction <- predict(fit, test, type = "class")
submit <- data.frame(PassengerId = test$PassengerId, Survived = Prediction)
```

Machine learning with Decision Trees



Feature engineering

- definition of **feature engineering** is to chop, and combine different attributes that we were given to squeeze a little bit more value from them
- We can create additional features to better understand and predict our data
- For example we can see if titles of people like Capt, Dr, Lady, Miss or Sir the Countess may have an impact on our prediction

Feature engineering code

```
#feature engineering
# definition of feature engineering is to chop,
#and combine different attributes that we were given to squeeze a little bit more value from them


#See if name field could be a viable feature
train$Name[1]
#we see that there are titles for the people, "Master" and "Countess"

#extract the titles of people to make new variables

#reload train and test and clear variables (top right)
test <- read.csv("test.csv", stringsAsFactors=FALSE)
train <- read.csv("train.csv", stringsAsFactors=FALSE)

#remove Surivived from train to have equal number of columns
train$Survived <- NULL

#combine train and test set for feature engineering
combi <- rbind(train, test)
#test$Survived <- NA
#train$Survived <- NA
#new dataframe called "combi" with all the same rows as the original two datasets
#stacked in the order in which we specified: train first, and test second.

#change name field to a character
combi$Name <- as.character(combi$Name)
combi$Name[1]

#break apart title from persons name
strsplit(combi$Name[1], split='[,.]')

#^Those symbols in the square brackets are called regular expressions
#isolate the title
strsplit(combi$Name[1], split='[,.]')[[1]][2]
```

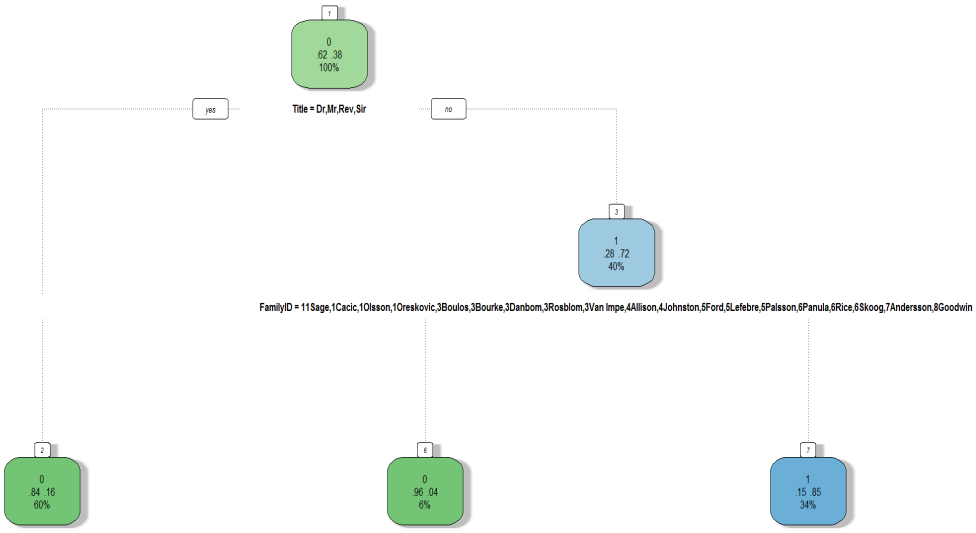
Feature engineering code 2

```
#^Those symbols in the square brackets are called regular expressions
#isolate the title
strsplit(combi$Name[1], split='[,.]')[[1]][2]

#We feed sapply our vector of names and our function that we just came up with.
#It runs through the rows of the vector of names, and sends each name to the function
combi$Title <- sapply(combi$Name, FUN=function(x) {strsplit(x, split='[,.]')[[1]][2]})

# strip off those spaces from the beginning of the titles
combi$Title <- sub(' ', '', combi$Title)

#view the titles of everyone
table(combi$Title)

#Mademoiselle and Madame are pretty similar (so long as you don't mind offending) so let's combine them into a single category
combi$Title[combi$Title %in% c('Mme', 'Mlle')] <- 'Mlle'

#%in% operator checks to see if a value is part of the vector we're comparing it to.
#So here we are combining two titles, "Mme" and "Mlle",
#into a new temporary vector using the c() operator and seeing if any of the existing titles
#in the entire Title column match either of them. We then replace any match with "Mlle"

#reducing the outliers to more simple titles
combi$Title[combi$Title %in% c('Capt', 'Don', 'Major', 'Sir')] <- 'Sir'
combi$Title[combi$Title %in% c('Dona', 'Lady', 'the Countess', 'Jonkheer')] <- 'Lady'

#change the variable back to a factor
combi$Title <- factor(combi$Title)

#two variables SibSb and Parch that indicate the number of family members the passenger is travelling with
combi$FamilySize <- combi$SibSp + combi$Parch + 1

#Pretty simple! We just add the number of siblings, spouses, parents and children the passenger had with them,
#and plus one for their own existence of course,
#and have a new variable indicating the size of the family they travelled with.

#ensure the families with the same naems are combined together correctly
combi$Surname <- sapply(combi$Name, FUN=function(x) {strsplit(x, split='[,.]')[[1]][1]})
```

Feature engineering code 3

```
#convert the FamilySize variable temporarily to a string and combine it with the Surname to get our new FamilyID variable
combi$FamilyID <- paste(as.character(combi$FamilySize), combi$Surname, sep="")

#the new family groups test
table(combi$FamilyID)


#Perhaps some families had different last names, but whatever the case, all these one or two people groups is what we sought to avoid with the three person cut-off.
#clean this up
famIDs <- data.frame(table(combi$FamilyID))

#subset this dataframe to show only those unexpectedly small FamilyID groups
famIDs <- famIDs[famIDs$Freq <= 2,]

#We then need to overwrite any family IDs in our dataset for groups that were not correctly identified and finally convert it to a factor
combi$FamilyID[combi$FamilyID %in% famIDs$Var1] <- 'Small'
combi$FamilyID <- factor(combi$FamilyID)

#break apart the comi dataframe into train and test sets again
train <- combi[1:891,]
test <- combi[892:1309,]

#new decision tree with newly engineering features
fit <- rpart(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked + Title + FamilySize + FamilyID,
             data=train, method="class")
fancyRpartPlot(fit)

Prediction <- predict(fit, test, type = "class")
submit <- data.frame(PassengerId = test$PassengerId, Survived = Prediction)
```

New tree, new accuracy, but less leaves



How accurate are our features in predicting? Part 1

```
# Set working directory and import datafiles

train <- read.csv("train.csv")
test <- read.csv("test.csv")

# Install and load required packages for decision trees and forests


# Join together the test and train sets for easier feature engineering
test$Survived <- NA
combi <- rbind(train, test)

# Convert to a string
combi$Name <- as.character(combi$Name)

# Engineered variable: Title
combi$Title <- sapply(combi$Name, FUN=function(x) {strsplit(x, split='[,.]')[[1]][2]})
combi$Title <- sub(' ', '', combi$Title)
# Combine small title groups
combi$Title[combi$Title %in% c('Mme', 'Mlle')] <- 'Mlle'
combi$Title[combi$Title %in% c('Capt', 'Don', 'Major', 'Sir')] <- 'Sir'
combi$Title[combi$Title %in% c('Dona', 'Lady', 'the Countess', 'Jonkheer')] <- 'Lady'
# Convert to a factor
combi$Title <- factor(combi$Title)

# Engineered variable: Family size
combi$FamilySize <- combi$SibSp + combi$Parch + 1

# Engineered variable: Family
combi$Surname <- sapply(combi$Name, FUN=function(x) {strsplit(x, split='[,.]')[[1]][1]})
```

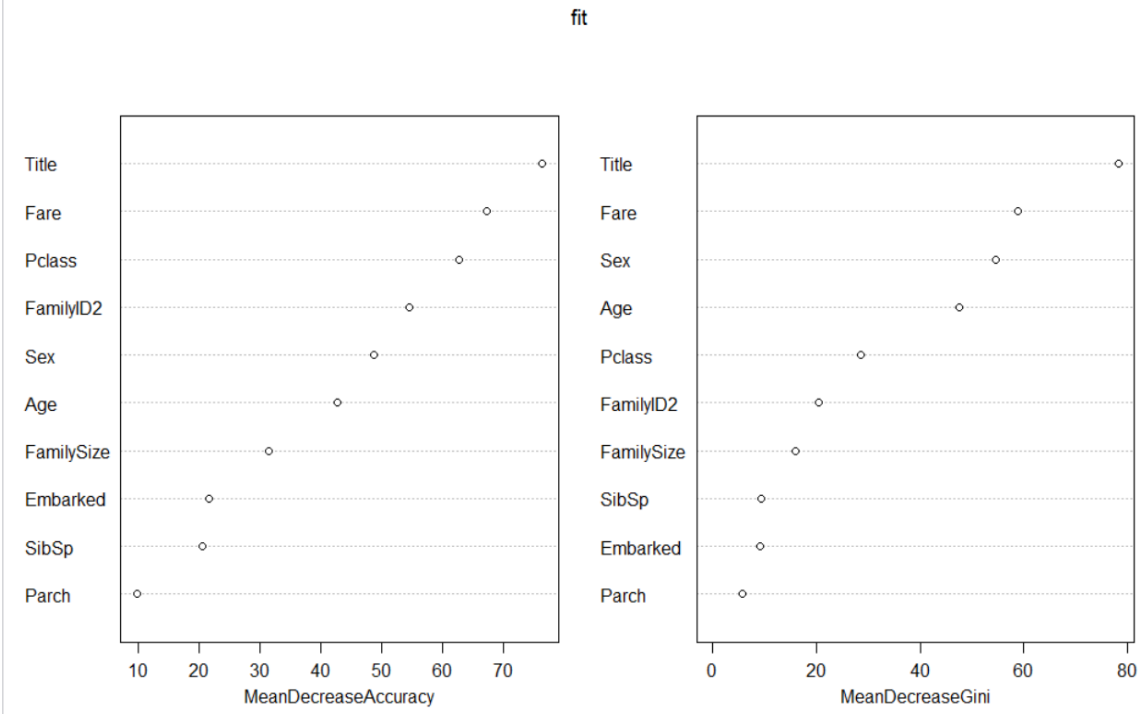
How accurate are our features in predicting? Part 2

```
combi$Surname <- sapply(combi$Name, FUN=function(x) {strsplit(x, split='[,.]')[[1]][1]})
combi$FamilyID <- paste(as.character(combi$FamilySize), combi$Surname, sep="")
combi$FamilyID[combi$FamilySize <= 2] <- 'Small'

# Delete erroneous family IDs
famIDs <- data.frame(table(combi$FamilyID))
famIDs <- famIDs[famIDs$Freq <= 2,]
combi$FamilyID[combi$FamilyID %in% famIDs$Var1] <- 'Small'

# Convert to a factor
combi$FamilyID <- factor(combi$FamilyID)

# Fill in Age NAs
summary(combi$Age)
Agefit <- rpart(Age ~ Pclass + Sex + SibSp + Parch + Fare + Embarked + Title + FamilySize,
                data=combi[!is.na(combi$Age),], method="anova")
combi$Age[is.na(combi$Age)] <- predict(Agefit, combi[is.na(combi$Age),])

# Check what else might be missing
summary(combi)
# Fill in Embarked blanks
summary(combi$Embarked)
which(combi$Embarked == '')
combi$Embarked[c(62,830)] = "S"
combi$Embarked <- factor(combi$Embarked)


# Fill in Fare NAs
summary(combi$Fare)
which(is.na(combi$Fare))
combi$Fare[1044] <- median(combi$Fare, na.rm=TRUE)
```

How accurate are our features in predicting? Part 3

```
combi$Fare[1044] <- median(combi$Fare, na.rm=TRUE)

# New factor. only allowed <32 levels, so reduce number
combi$FamilyID2 <- combi$FamilyID
# Convert back to string
combi$FamilyID2 <- as.character(combi$FamilyID2)
combi$FamilyID2[combi$FamilySize <= 3] <- 'Small'
# And convert back to factor
combi$FamilyID2 <- factor(combi$FamilyID2)

# Split back into test and train sets
train <- combi[1:891,]
test <- combi[892:1309,]

# Build Random Forest Ensemble
set.seed(1234)
fit <- randomForest(as.factor(Survived) ~ Pclass + Sex + Age + SibSp + Parch + Fare +
                      Embarked + Title + FamilySize + FamilyID2,
                    data=train, importance=TRUE, ntree=2000)
# Look at variable importance
varImpPlot(fit)


# Now let's make a prediction and write a submission file
Prediction <- predict(fit, test)
submit <- data.frame(PassengerId = test$PassengerId, Survived = Prediction)
View(submit)
```

Variable Accuracy



Business Use Case: bank.csv

Using the bank.csv file in the resources section of Piazza perform the following

1. **Find the correlation relationships by plotting a corrplot**
2. **Make sure to change variables to numeric for modeling**
3. **Create 2 new features**
4. **Split the banks.csv into a data set and training set with 75% training and 25% test set**
5. **Choose an algorithm to predict on y**
6. **What's your accuracy on your prediction?**