CS 660: Mathematical Foundations for Analytics

Dr. Francis Parisi

Pace University

Spring 2017

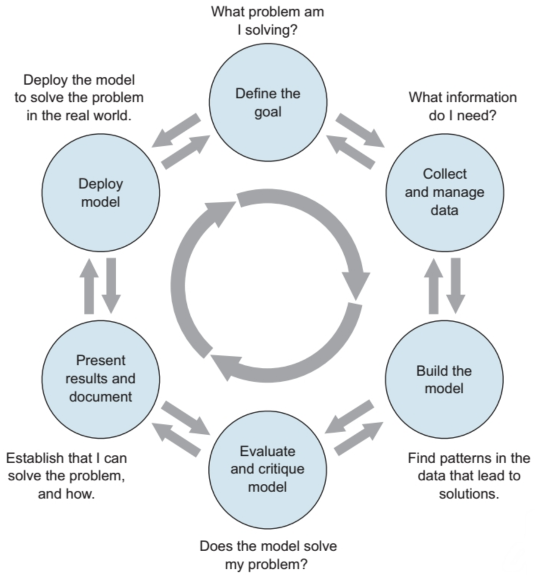
Course Overview

1. Part I: Data Science Fundamentals
    - Data Science Concepts and Process
    - The R Language
    - Exploratory Data Analysis
    - Cleaning & Manipulating Data
    - Presenting Results
2. Part II: Graphs & Statistical Methods
    - Basic Graphics
    - Advanced Graphics
    - Probability & Statistical Methods
3. Part III: Modeling Methods
    - Linear and Logistic Regression
    - Model Selection and Evaluation
    - Unsupervised Methods
    - Advanced Modeling Methods

Data Science Concepts and Process

- Data science is more than statistical analysis
- Emphasis on collaboration and project definition
- Project roles
  - Project sponsor
  - Client or SME
  - Data scientist
  - Data architect
  - Operations
- Data science project life cycle ...

Data Science Concepts and Process



Data Science Concepts and Process

- Project goal – why are we doing this?
- Data collection, quality, sufficiency, and management
- Model development
- Model evaluation and sufficiency
- Presentation to stakeholders, project documentation, and reproducibility

Data Science Concepts and Process

Communicating Results

- You're telling a story
- What are the questions you're seeking to answer?
- Why are these interesting questions?

Data Science Concepts and Process

General structure of a paper

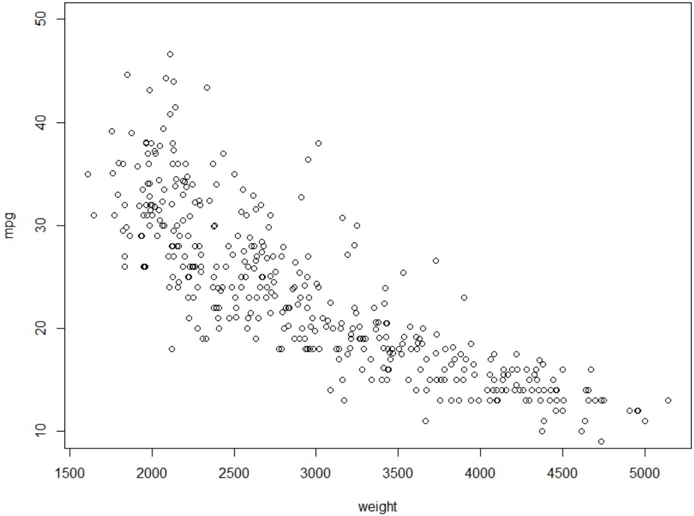
1. Abstract – Brief summary of the question(s) of interest, the methodology and the results
2. Introduction – Clear statement of the scientific question, objectives of the study, background information to put the question and research into perspective
3. Methods/Methodology – Describe the design of the study and the analytical methodology you used
4. Results – Describe the key results of your data analysis and interpret the results with respect to the objectives of the study and the question(s) of interest

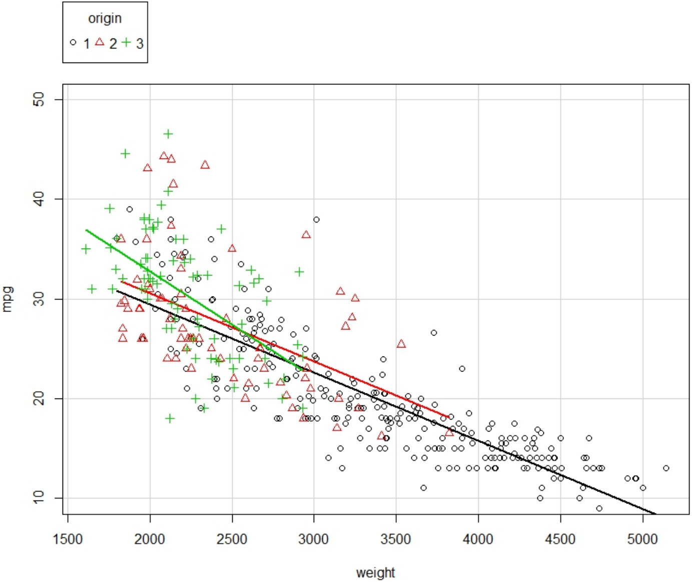
General structure of a paper (cont'd)

1. Conclusions – Summary of the main findings and a discussion of how these findings relate to the question(s) of interest, comparison to results from other studies, and ideas for future research
2. References – Citations for literature you used
3. Technical Appendices (if necessary) – Describe more complicated analyses, or derivations and proofs, or any other material that would disrupt the flow of the paper if it were included in the body

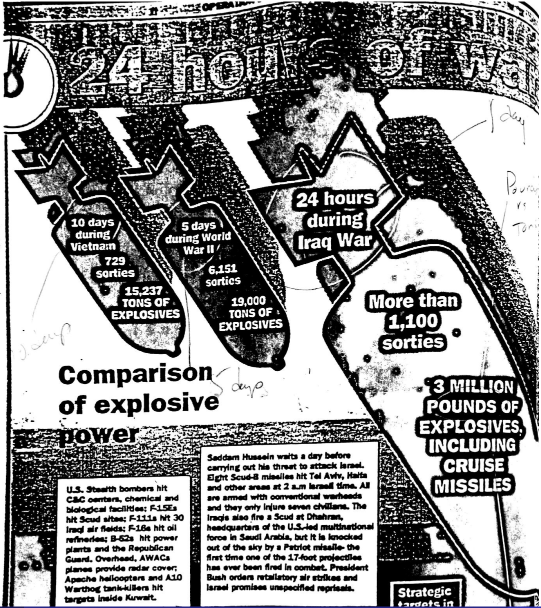
Use graphs to clarify not confuse or mis-lead

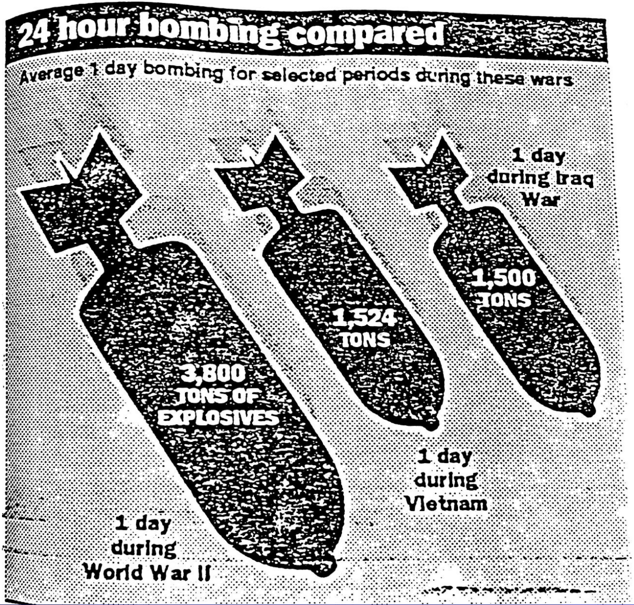
Data Science Concepts and Process




origin
1 2 3
50
40
30
20
10
mpg
1500
2000
2500
3000
3500
4000
4500
5000
weight

Data Science Concepts and Process




24 hour bombing compared
Average 1 day bombing for selected periods during these wars
3,800 TONS OF EXPLOSIVES
1 day during World War II
1,524 TONS
1 day during Vietnam
1,500 TONS
1 day during Iraq War

The R Language

- R is great tool for statistical analysis, visualization and reporting
- Installing R from CRAN
  `http://cran.r-project.org/`
- Installing RStudio
  `https://www.rstudio.com/products/rstudio/#Desktop`
- Installing R packages
  `http://cran.r-project.org/packages/`

R Basics – Data Types

- Numeric data
- Character data (“string”)
- Dates and times
- Logical data
- Factors

R Basics - Math

```
> 1 + 1 # Spaces between operators are not necessary
[1] 2
```

```
> 3 * 7 * 2
[1] 42
```

```
> 4/3
[1] 1.333333
```

```
> sqrt(2)
[1] 1.414214
```

The number of *digits* displayed is set using `options(digit =` n`)`

```
> options(digits = 4)
> 37/3
[1] 12.33 # notice there are only 4 digits displayed
```

R Basics – Variables

- Variable Assignment <- or =

```
> x <- 2
> x
[1] 2
```

```
> y = 5
> y
[1] 5
```

- The arrow operator is preferred, and can also point in the other direction

```
> 3 -> z
> z
[1] 3
```

R Basics – Variables

- We can assign a value to multiple variables simultaneously

```
> a <- b <- 7
> a
[1] 7
```

```
> b
[1] 7
```

- Sometimes it is necessary to use the `assign()` function

```
> assign("j", 4)
> j
[1] 4
```

R Basics – Variables

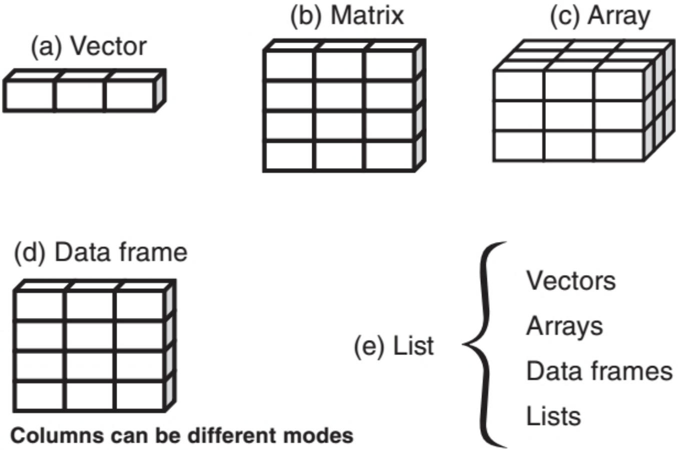
- The `rm()` function allows use to remove objects

```
> rm(j) # removes object j
> j
Error:  object 'j' not found
```

- Variable names ARE case sensitive

```
> myVariable <- 75
> myVariable
[1] 75
> MYVARIABLE
Error:  object 'MYVARIABLE' not found
```

R Basics – Data Structures



Source: Kabacoff (2015)

R Basics – Data Structures

- Vectors – the combine function `c()` creates vectors

```
> a <- c(4, 6, 7, 2, -3.5, 19)
> b <- c("hello", "world", "cats", "dogs")
> c <- c(TRUE, TRUE, FALSE, TRUE, FALSE)
```

- Vector operations

```
> a[3]
[1] 7
> a[c(2,4,5)]
[1] 6.0 2.0 -3.5
> a[2:4]
[1] 6 7 2
> a * 4 # multiply every element at once
[1] 16 24 28 8 -14 76
```

- Matrices

```
> y <- matrix(1:20, nrow=5, ncol=4)
> y
      [,1] [,2] [,3] [,4]
 [1,]    1    6   11   16
 [2,]    2    7   12   17
 [3,]    3    8   13   18
 [4,]    4    9   14   19
 [5,]    5   10   15   20
```

- If the number of dimensions exceeds the number of elements R will recycle . . . try this

```
> matrix(1:20, nrow=5, ncol=5)
```

- Creating and subscripting matrices

```
> x <- matrix(1:15, nrow=3)
> x
       [,1] [,2] [,3] [,4] [,5]
  [1,]    1    4    7   10   13
  [2,]    2    5    8   11   14
  [3,]    3    6    9   12   15
```

- `matrix()` fills columnwise by default
- This behavior is easily modified

R Basics – Data Structures

```
> x <- matrix(1:15, nrow=3, byrow = TRUE)
> x # fill the matrix rowwise
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
[3,]   11   12   13   14   15
> x[2,]
[1] 6 7 8 9 10
> x[,2]
[1] 2 7 12
> x[ , 2, drop = FALSE]
     [,1]
[1,]    2
[2,]    7
[3,]   12
```

R Basics – Data Structures

- **Arrays** are like matrices but have more then two dimensions
- Think of an array as an Excel workbook with multiple sheets

  ```
  > z <- array(1:18, c(2,3,3))
  ```

  An array with two rows, three columns and three “sheets”
- **Data frames** are the most commonly used R structure in data modeling
- Similar to a matrix but columns can have different modes of data

R Basics – Data Structures

- Creating a data frame

```
> # vector of patient ID nums
> patientID <- c(1, 2, 3, 4)
> # vector of ages
> age <- c(25, 34, 28, 52)
> # vector of type
> diabetes <- c("Type1", "Type2", "Type1",
"Type1")
> # vector of health status
> status <- c("Poor", "Improved", "Excellent",
"Poor")
Create the data frame
> patientdata <- data.frame(patientID, age,
diabetes, status)
```

R Basics – Data Structures

- Examining the structure of data frame

  `str(patientdata)`

- Looking at the first few observations of a data frame

  `head(patientdata)`

- Initially `diabetes` and `status` are character data
- But these really represent distinct categories
- For modeling purposes we should make them factors

  ```
  diabetes <- factor(diabetes)
  ```

  ```
  status <- factor(status, order=TRUE,
  levels=c("Poor", "Improved", "Excellent"))
  ```

- Pass these to `data.frame()`

R Basics – Data Structures

- **Lists** are the most complex of R's data structures but offer great flexibility
- Lists may contain any combination of R structures . . . the columns of a `data.frame` must be vectors

```
> list1 <- list(c(1,2,3), c("Bob", "Joe",
"Ellen"), patientdata, list(c(5,6,7), "Mike"))
```

- We can name the elements of a list

```
> list2 <- list(num=c(1,2,3), names=c("Bob",
"Joe", "Ellen"), df=patientdata,
daList=list(c(5,6,7), "Mike"))
```

- And we can embed lists within lists

```
list3 <- list(list1, list2)
```

- Functions
  - Built-in functions
    `mean(), str(), head(), cos(), read.table(), nchar(), length(), lm(), summary(), anova(), ggplot(), install.packages(), etc...`
  - User defined

```
hello.user <- function(yourname)
{
  print(sprintf("Hello %s", yourname))
}
```

```
Doubleit <- function(x)
{
  return(x * 2)
}
```

- Control Flow
  - `if` and `else`
  - `switch`
  - `ifelse`
  - `for` loops
  - `while` loops
- Avoid loops where you can – take advantage of R vectorized operations, and the `apply` family of functions (more later)
- Compare times for the following code

```
> xx = 1:1000
> o1 <- system.time(for(i in 1:length(xx))
print(xx[i]))
> o2 <- system.time(print(xx))
> o1
> o2
```

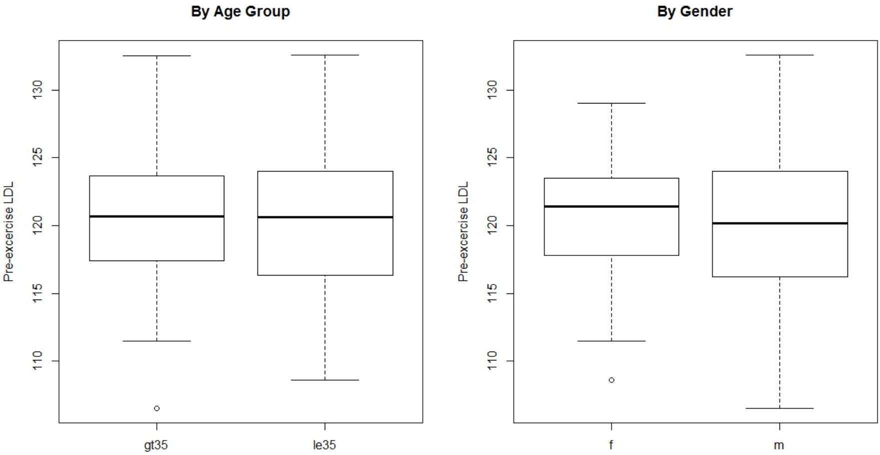
Exploratory Data Analysis

- EDA allows us to get a sense for what data we have if/and how they are related
- Summary statistics

```
> summary(ldl)
     age               gender              ldl.pre          ldl.post    
 Length:100         Length:100         Min.   :106.5   Min.   :102.9  
 Class :character   Class :character   1st Qu.:116.9   1st Qu.:112.8  
 Mode  :character   Mode  :character   Median :120.7   Median :117.2  
                                       Mean   :120.4   Mean   :116.7  
                                       3rd Qu.:123.9   3rd Qu.:119.9  
                                       Max.   :132.6   Max.   :130.6  
> ldl$fgender <- factor(ldl$gender)
> ldl$fage <- factor(ldl$age)
```

```
> summary(ldl[,-c(1,2)])
 ldl.pre           ldl.post          fgender   fage
 Min.    :106.5   Min.    :102.9   f:50     gt35:51
 1st Qu. :116.9   1st Qu. :112.8   m:50     le35:49
 Median  :120.7   Median  :117.2
 Mean    :120.4   Mean    :116.7
 3rd Qu. :123.9   3rd Qu. :119.9
 Max.    :132.6   Max.    :130.6
```

- Visualizing data



Exploratory Data Analysis

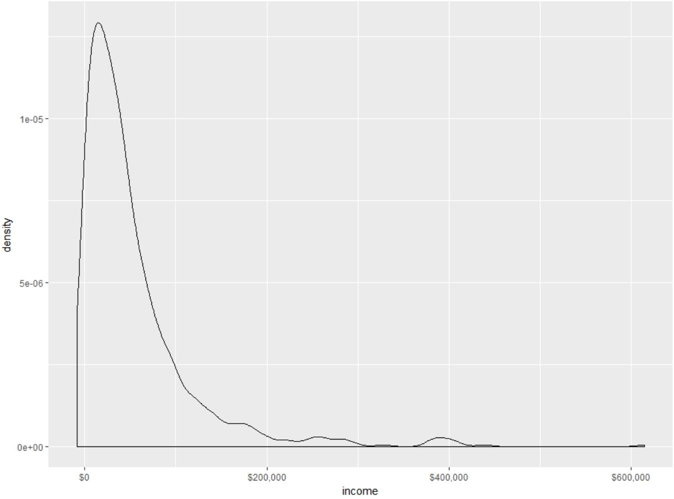
- Identifying data problems through data summaries and visualization
  - Missing data
  - Invalid data and outliers
  - Data ranges and comparable units

Exploratory Data Analysis

```
> # create a random sample of 100 numbers (size)
from 1 through 100, and allow repeated values
> x <- sample(x = 1:100, size = 100, replace = TRUE)
> # calculate the mean()
> mean(x)
[1] 48.78
> # make a copy of x
> y <- x
> # Draw a random sample of size 20 from our vector
y and set those values to NA
> y[sample(x = 1:100, size = 20, replace = FALSE)]
<- NA
> mean(y)
[1] NA
> mean(y, na.rm = TRUE)
[1] 49.225
```

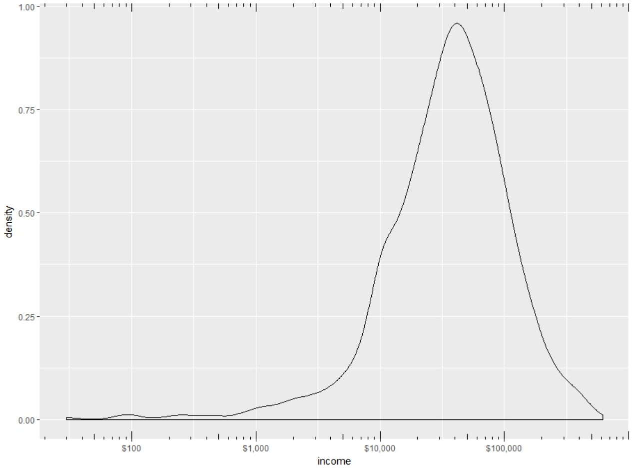
Exploratory Data Analysis

```
> library(scales)
> ggplot(custdata) + geom_density(aes(x=income)) +
scale_x_continuous(labels = dollar)
```



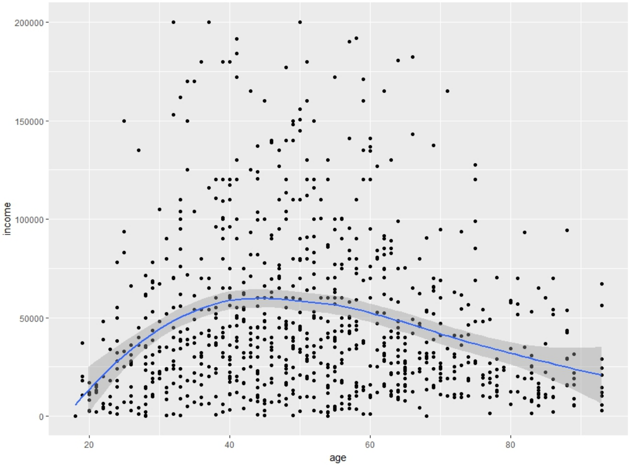
Exploratory Data Analysis

```
> ggplot(custdata) + geom_density(aes(x=income)) +
scale_x_log10(breaks = c(100,1000,10000,100000), labels=dollar)
+ annotation_logticks(sides = "bt")
```



Exploratory Data Analysis

```
> custdata2 <- subset(custdata, (custdata$age > 0 &
custdata$age < 100 & custdata$income > 0))
> ggplot(custdata2, aes(x=age, y=income)) + geom_point() +
geom_smooth() + ylim(0, 200000)
```



Cleaning & Manipulating Data

- Data transformations
  - It is often helpful/necessary to transform data for analysis
  - As we saw earlier, the transformed income data helped in visualization
  - When fitting a linear model a $\log$ transform will reduce nonlinearity
- Invalid data values
- Data ranges and units

Cleaning & Manipulating Data

- Functions for data manipulation
  - `apply`
  - `lapply` and `sapply`
  - `mapply`
  - `aggregate`
  - `plyr` package by Hadley Wickham
    - `ddply`
    - `llply`
    - `plyr` helper functions
  - `data.table` package
    - extends and enhances the functionality of `data.frame`
    - uses an index like databases
    - increased speed, group operations and joins
    - we can set keys using the `setkey` function
    - faster aggregation using the built-in functionality
    - the only drawback is the syntax for `data.table` is different and will take some getting used to it

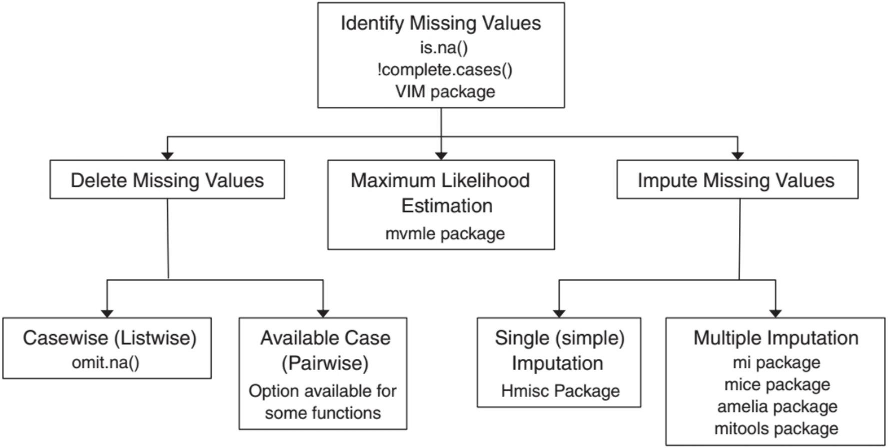
Cleaning & Manipulating Data

- Handling missing data - just deleting is not always good
- Data may be missing for a number of reasons
- It is important to understand what data are missing and why
  - Identifying missing data
  - Visualizing missing data patterns
- We can remove the observations with missing data (avoid if possible)
  - Complete-case analysis – listwise deletion
  - Pair-wise deletion
- We can replace the missing values
  - Simple imputation
  - Multiple imputation
- R packages `VIM`, `mice`, `Hmisc`, `mi` and more

Classifying Missing Data

- Missing completely at random (MCAR)
- Missing at Random (MAR)
- Not Missing at Random (NMAR)

Cleaning & Manipulating Data



Source: Kabacoff (2015)

Presenting Results

- Clear communication of results
- Target presentation to your audience
  - Project sponsor or company executives
  - End-users
  - Other data scientists, analysts or researchers

Presenting Results

Project sponsor or company executives

- Summarize the project's goals motivation for doing it
- State the results
- Provide details as needed
- Discuss recommendations or future work

Think Journalistic Style . . .

Presenting Results

End-users

- Summarize the project's goals and motivation for doing it
- Show how the model fits into the users' workflow and *improves* the workflow
- Show how to use the model

Presenting Results

Other data scientists, analysts or researchers

- Introduce the problem
- Discuss related work
- Discuss your approach
- Results and findings
- Discuss future work

This reflects the structure of a research article in a journal

References

Kabacoff, R. I. (2015).
R in Action.
Manning, Shelter Island, NY, second edition.

Lander, J. P. (2014).
R for Everyone.
Addison-Wesley, Upper Saddle River.

Zumel, N. and Mount, J. (2014).
Practical Data Science with R.
Manning, Shelter Island, NY, second edition.