Northeastern University

INFO 6105
Data Sci Eng Methods & Tools
Lecture 2 LABS *12 January 2023*

Programming =

- ..working with many numbers at the same time
- ..storing intermediate computations in variables (like M+)


NORTHEASTERN UNIVERSITY
LVX VERITAS VIRTVS
FOUNDED AD 1898
BOSTON, MASSACHUSETTS

Part 1
R

R vs Python

- Ross Ihaka and Robert Gentleman created the open-source language R in 1995 as an implementation of the S programming language
  - Purpose was to develop a language that focused on delivering a better and more user-friendly way to do data analysis, statistics and graphical models
  - CRAN, a huge repository of curated R packages to which users can easily contribute
    - https://cran.r-project.org/

Preliminaries

- Install *R*, *RStudio*, and the *dplyr* and *ggplot2* packages
  - In Rstudio console:
    ```
    install.packages("dplyr")
    install.packages("ggplot2")
    ```
- Set the working directory to the directory of the RHandsOn zip file on blackboard, as follows:
  - Session -> Set Working Directory -> Choose Directory...
  - Navigate and Open the /programs directory
- *Focus console after executing from source* option: Moves the focus to the console after executing a line or selection of code within the source editor (you *will* thank me)
  - Tools -> Global Options -> Code Editing...
  - Check "Focus console after executing from Source"

Topics

1. **Data manipulation, including package `dplyr`**
2. **Graphics, including package `ggplot2`**
3. **Basic statistical models: linear and logistic regression**

0-intro.R

- Open the file 0-intro.R [File - Open file...]
- If the file appears empty, then..
  - `File | Reopen with Encoding | UTF-8`
- Learn programming by reading and executing instructions one by one to familiarize yourself with R commands..

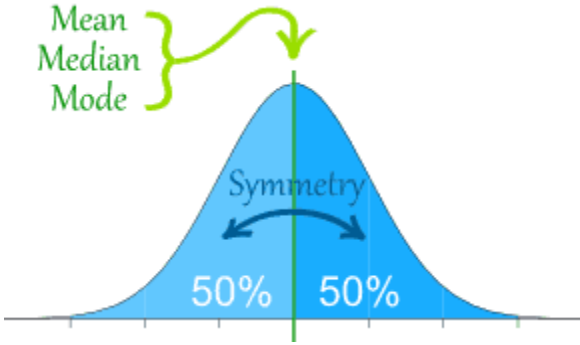
1-data.R

- Open the file 1-data.R [File - Open file...]
- *CTRL-L to clear the Rstudio console*
- college <- read.csv("../data/College.csv")
  - College.csv is a dataset of 777 different universities and colleges in the US, and contains information stored in a number of variables
- What do you think would happen if you try to load a file that is bigger than the amount of RAM on your laptop?

Normal distribution (rnorm)

- **We say the data is "normally distributed“ when:**
  - **The probability density of the normal distribution is:**

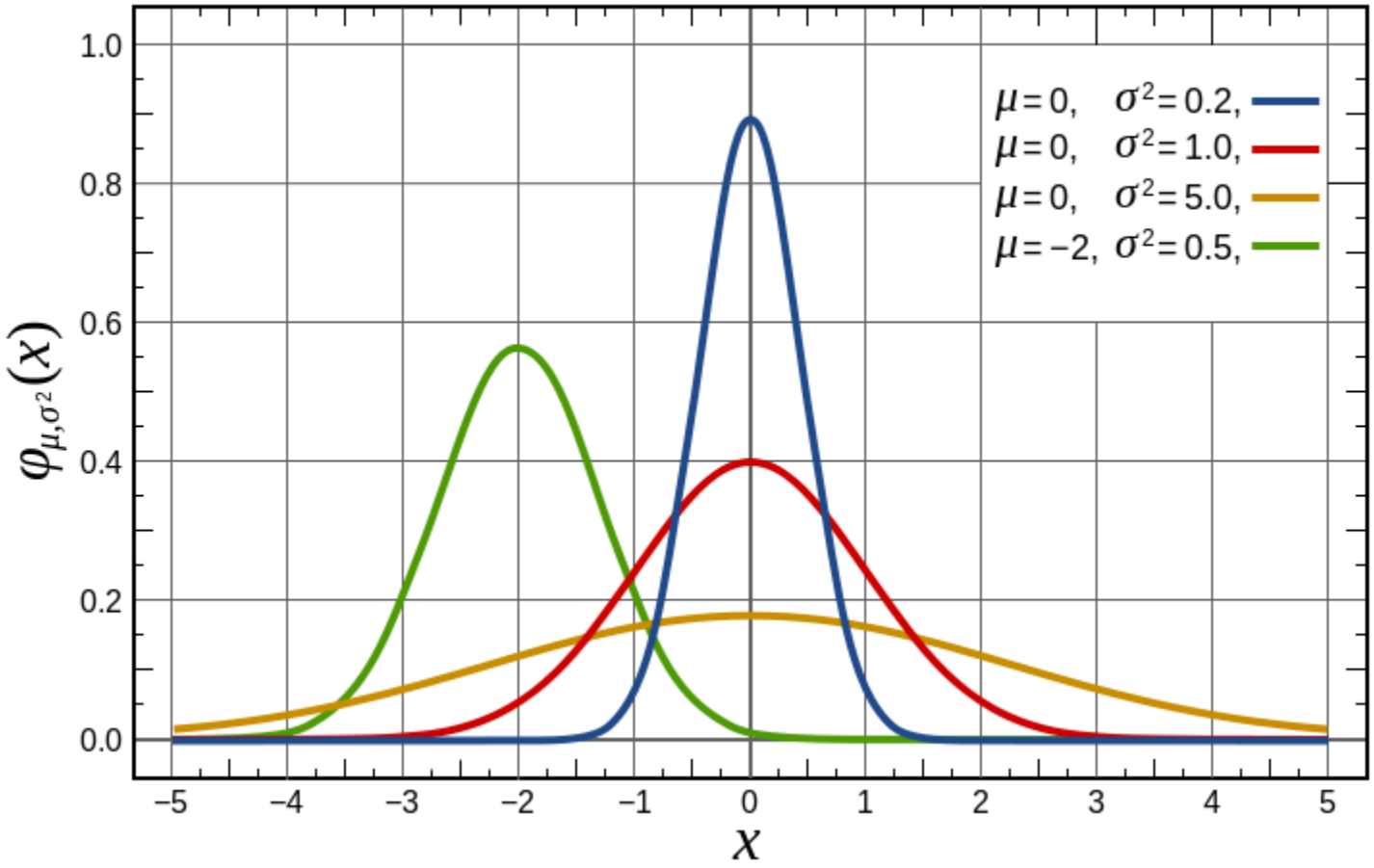
$$f(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



  - **Here, μ is the mean or expectation of the distribution (and also its median and mode). The parameter σ is its standard deviation; its variance is then σ^2**
  - **If μ = 0 and σ = 1 the distribution is called the *standard normal distribution* or the *unit normal distribution***

symmetry about the center
50% of values less
than the
mean and 50% greater
than
the mean

Normal (a.k.a. Gaussian) Distributions



dplyr

- Solves most common data manipulation operations, so that your options are helpfully constrained when thinking about how to tackle a problem (replaces plyr)
- Does split-apply-combine (SAC) logic
  - Best analogy is the `GROUP BY` statement in SQL
  - http://www.r-bloggers.com/do-your-data-janitor-work-like-a-boss-with-dplyr/
- Provides simple functions that correspond to the most common data manipulation verbs, so that you can easily translate your thoughts into code
- Uses efficient data storage back-ends, so that you spend as little time waiting for the computer as possible

dplyr (more)

- Abstracts away how your data is stored, so that you can work with data frames, data tables and remote databases using the same functions
  - This lets you think about what you want to achieve, not the logistics of data storage
- Provides a thoughtful default `print()` method so you don't accidentally print pages of data to the screen
- Compared to base functions, dplyr is more consistent: Functions have the same interface so that once you've mastered one, you can easily pick up the others

dplyr:filter()

- filter() allows you to select a subset of the rows of a data frame
  - The first argument is the name of the data frame, and the second and subsequent are filtering expressions evaluated in the context of that data frame
  - filter(flights, month == 1, day == 1) is equivalent to:
  - flights[flights$month == 1 & flights$day == 1,]

dplyr:select()

- **Allows you to rapidly zoom in on a useful subset using operations that usually only work on numeric variable positions**
  - **# Select columns by name**
    select(flights, year, month, day)

dplyr: summarize()

- **You use summarize() with aggregate functions, which take a vector of values, and return a single number**
  - **There are many useful functions in base R like *min(), max(), mean(), sum(), sd(), median(), n()* (number of observations in the current group), *n_distinct(x)* (count the number of unique values in x)**

Grouped Ops

- **select() now begins to look a lot like SQL's SELECT operator, and group_by() like SQL's GROUP BY operator..**

Chaining

- The *dplyr* API is ***functional*** in the sense that function calls don't have side-effects
- If you want to do many operations at once, you either have to do it step-by-step, or
- If you don't want to save intermediate results, you need to wrap the function calls inside each other

Step by step


```
a1 <- group_by(flights, year, month, day)
a2 <- select(a1, arr_delay, dep_delay)
a3 <- summarise(a2,
  arr = mean(arr_delay, na.rm = TRUE),
  dep = mean(dep_delay, na.rm = TRUE))
a4 <- filter(a3, arr > 30 | dep > 30)
```

Wrapping function calls

```
filter(
  summarise(
    select(
      group_by(flights, year, month, day),
      arr_delay, dep_delay
    ),
    arr = mean(arr_delay, na.rm = TRUE),
    dep = mean(dep_delay, na.rm = TRUE)
  ),
  arr > 30 | dep > 30
)
#> Source: local data frame [49 x 5]
#> Groups: year, month
#>
#>    year month day      arr      dep
#> 1  2013     1  16 34.24736 24.61287
#> 2  2013     1  31 32.60285 28.65836
#> 3  2013     2  11 36.29009 39.07360
#> 4  2013     2  27 31.25249 37.76327
#> ..  ...   ... ...      ...      ...
```

- **Difficult to read because the order of the operations is from inside to out, and the arguments are a long way away from the function..**

The pipe (%>%) operator

Option: Using %>%

- `x %>% f(y)` turns into `f(x, y)`
- Use it to rewrite multiple operations so you can read from left-to-right, top-to-bottom:

```
flights %>%
  group_by(year, month, day) %>%
  select(arr_delay, dep_delay) %>%
  summarise(
    arr = mean(arr_delay, na.rm = TRUE),
    dep = mean(dep_delay, na.rm = TRUE)
  ) %>%
  filter(arr > 30 | dep > 30)
```

dplyr:mutate()

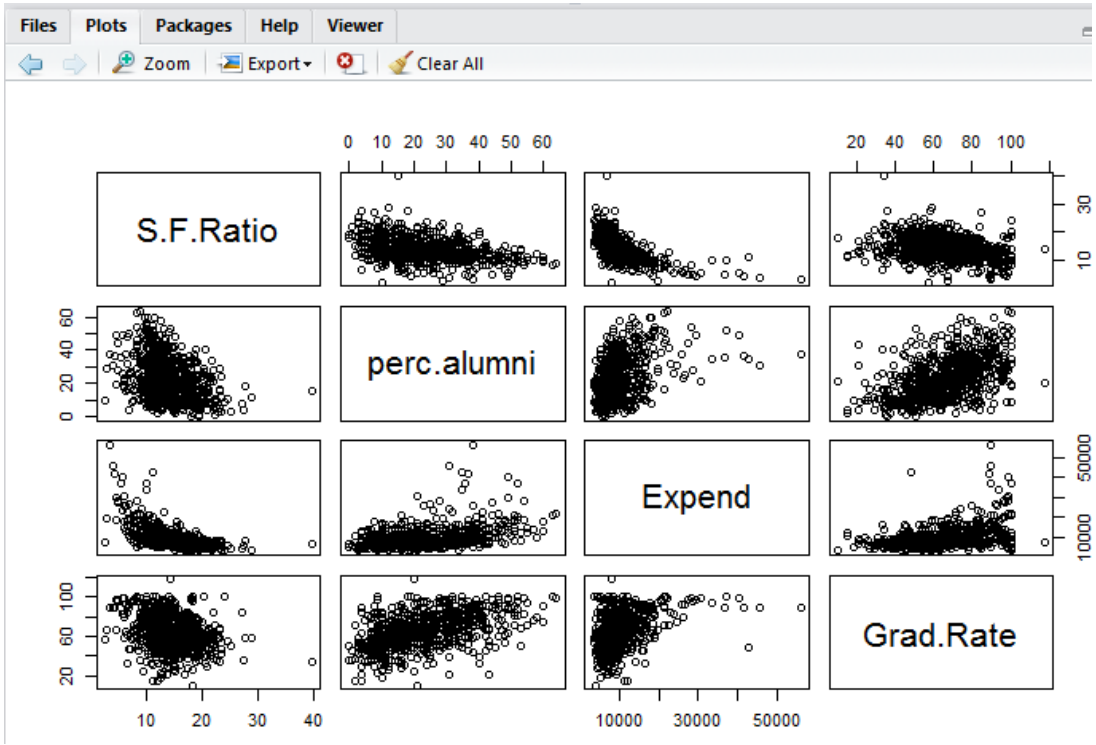
- Adds new columns that are functions of existing columns
  - mutate(flights,
    gain = arr_delay - dep_delay,
    speed = distance / air_time * 60)

2-graphics.R

- **Open the file 2-graphics.R [File - Open file...]**
- ***CTRL-L to clear the RStudio console***

Base R

- **Plot() is pretty decent!**
  - **pairs(college[,c(16:19)]):**



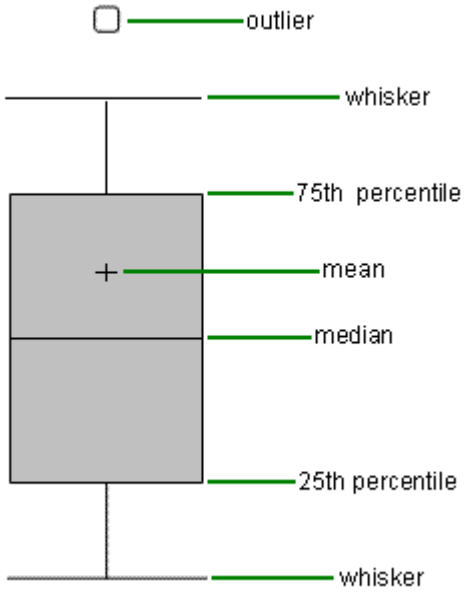
ggplot2

- **Designed to work in a layered fashion**
  - **Example:**

```
## aes = aesthetics
p <- ggplot(college, aes(x=S.F.Ratio, y=Grad.Rate))
p + geom_point()
p + geom_point(aes(colour = Private))
```

Box plot

- **The box plot (a.k.a. box and whisker diagram) is a standardized way of displaying the distribution of data based on the five number summary: *minimum, first quartile, median, third quartile, and maximum***



Facets

- Faceting approach supported by ggplot2 partitions a plot into a matrix of panels
  - Each panel shows a different subset of the data
  - facet_wrap(~cell) - univariate: create a 1-d strip of panels, based on one factor, and wrap the strip into a 2D matrix
  - facet_grid(row~col) - (usually) bivariate: create a 2D matrix of panels, based on two factors
- Good example of faceting data analysis:
  - http://sape.inf.usi.ch/quick-reference/ggplot2/facet

3-stats.R

- **Open the file 3-stats.R [File - Open file...]**
- ***CTRL-L to clear the Rstudio console***

What for

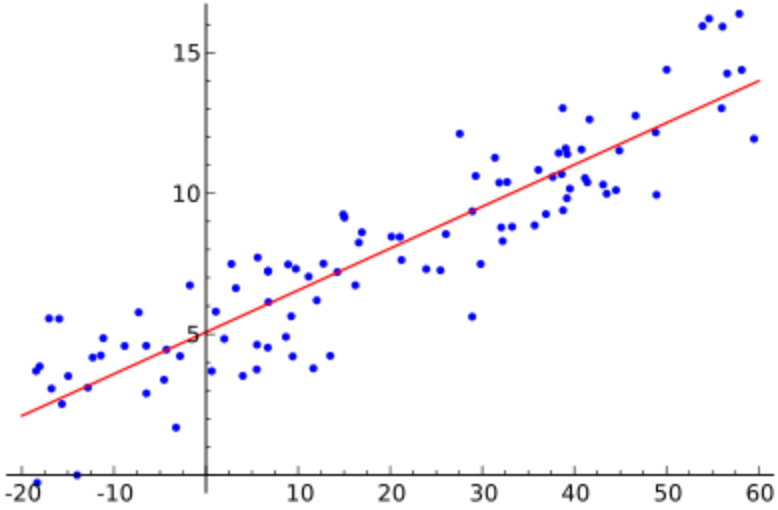
- **We use statistical analysis for:**
  - ***Inference* - making conclusions based on data**
  - ***Prediction* - what will happen when I observe new data?**
  - **And we create *models* to do both of those things**
- ***"All models are wrong - some are useful"* - George E. P. Box**

It’s been said that..

- **Our brain simply consists of a bunch of *predictors*, based on *models* we build for ourselves in our lifetimes..**

Linear Regression

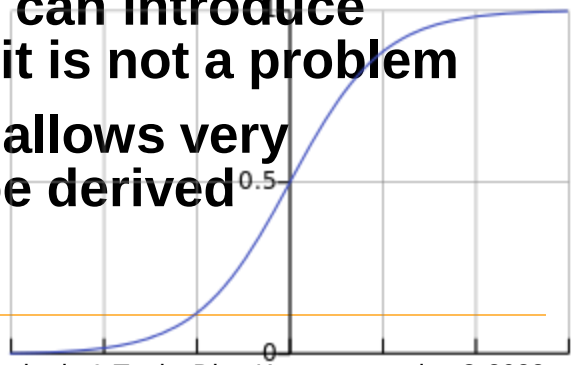
- **Regression is an approach for modeling the relationship between a scalar *dependent* variable y and one or more explanatory variable (*independent* variable) x**
  - **In linear regression, data are modeled using linear predictor functions**
  - **Linear regression models are often fitted using the least squares approach, but they may also be fitted in other ways..**



$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

Logistic Regression

- Logistic regression describes a kind of classification model in which predictor variables are combined with linear weights and then passed through a soft-limit function that limits the output to the range [0, 1]
- Logistic regression is closely related to other models such as:
  - Perceptron (where the soft limit is replaced by a hard limit)
  - *Neural networks* (where multiple layers of linear combination and soft limiting are used)
  - *Naive Bayes* (where linear weights are determined strictly by feature frequencies assuming independence)
- Logistic regression can't separate all possible classes, but in very high dimensional problems or where you can introduce new variables by combining other predictors, it is not a problem
- Mathematical simplicity of logistic regression allows very efficient and effective learning algorithms to be derived



7-oop.R

- Open the file 7-oop.R [File - Open file...]
- *CTRL-L to clear the Rstudio console*
- Although the latest built-in OOP model, RC ("reference classes"), is superior in many ways to S4 and its immediate predecessor model S3, both S3 and S4 are still widely used
  - Quite a few common R language functions were created using the S4 model
  - The RC model is much more like the C# OOP model you;re used to, but most R programmers come from a strictly R programming background and feel more comfortable with S3 or S4
  - One of the significant improvements in the S4 OOP model compared to the S3 model is that you can specify types for the fields. In addition to the atomic types "integer," "character," and "numeric" used in the demo, you can specify composite types such as "vector," "matrix," "array," and "data.frame"

Class Methods

- An S4 class encapsulates data fields inside a `setClass` function, but doesn't encapsulate class methods
- Instead, S4 class methods are defined by pairs of special R functions named `setMethod` and `setGeneric`
- There's a special initialize function that's defined only by `setMethod` but not with `setGeneric`
  - The special initialize function corresponds to a C++ constructor

Equivalents

- **C#**

```
Person p1 = new Person; // Default values for fields
p1.empID = 65565;
p1.lastName = "Adams"; // Change name
p1.hireDate = DateTime.Parse("2010/09/15");
p1.payRate = 43.21;
p1.Display;
int tenure = p1.YearsService;
```

- **R**

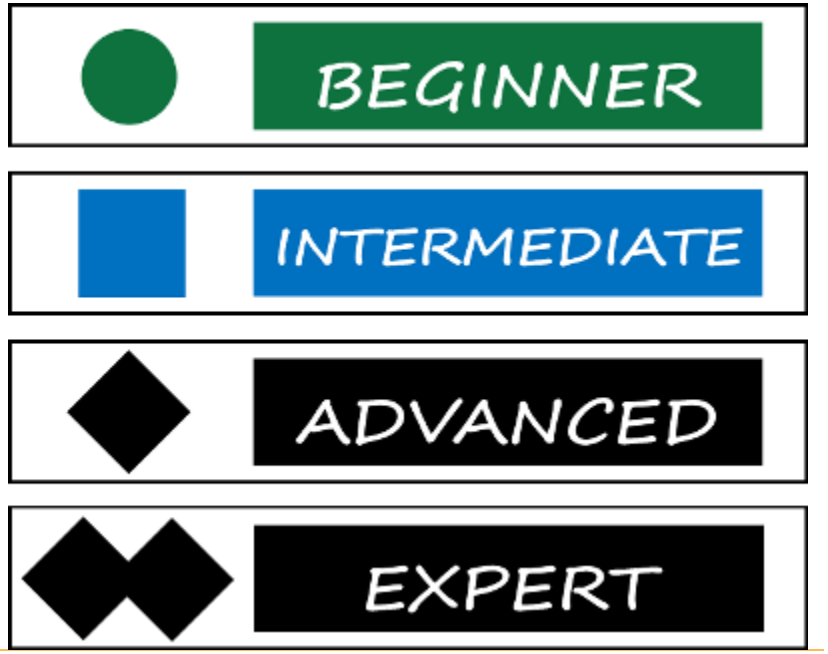
```
p1 <- new("Person") # default values for fields
p1@empID <- as.integer(65565)
p1@lastName <- "Adams"  # change name
p1@hireDate <- "2010-09-15"
p1@payRate <- 43.21
display(p1)
tenure <- yearsService(p1)
```

Resources

- **Quick-R**
  http://www.statmethods.net
- ***An Introduction to Statistical Learning with Applications in R*, by James et al (ISLR)**
- **Our *Data Wrangling* handout**

Homework: 4 levels

- Beginner programmers (***everybody*** does this first):
  - Fill in the missing code in the R files *we studied in class*
- Intermediate programmers (tm):
  - Do the Iliad + Odyssey ***Easy*** word cloud lab
- Advanced programmers (udpipe):
  - ***Advanced*** NLP lab Australia 2008
- Expert programmers (udpipe):
  - Wordcloud with your own corpus *in your own language*!
- Due: next week
- Groups: of 2 with independent submissions




CLASS
DISMISSED
iStockphoto