University of Chicago Professional Education

MSCA 37010 01/02
Programming for Analytics
Week 1 Lecture Notes

Autumn 2020

Welcome to the Class of Programming for Analytics!

❑ Introduction

Instructor: Mei Najim
Email: mnajim@uchicago.edu
Tel: 847-800-9979 (C)
Class Meeting Time: 6:00 - 9:00pm, Mondays (01 Section)
1:30 - 4:30pm Saturdays (02 Section)
Tentative Office Hour: Wednesdays (5:30pm – 7:00pm or until last students)
Saturdays (4:30pm – 6:00pm or until last students)
Sundays (1:00pm – 2:00pm or until last students)

Notes: 1) First ten-minute quiz; Two 10-minute breaks
2) Set up a discussion group on canvas, breakout groups in zoom; allow 24 hours to respond
3) Use Google Doc to share lectures notes
4) Try to email questions first and then we can set up a time if needed;
5) If it is urgent, feel free to text me directly (847-800-9979)

❑ Introduction

Week 1 Class Agenda

- Course Introduction
- Overview of Analytics
- Introduction to a General Predictive Analytics Process
  (Optional Reading: ReturnToWorkModel_CSPConference02162019_MeiYuNajim.pdf)
- Introduction to R, RStudio, and R Basics
  (Session Management, Arithmetic in R, Comparison Operators, Variables)
- Data Structures - Vector
  (Vector Basics, Operations, Indexing, and Slicing)

Week 1 Class Agenda

- Course Introduction
- Overview of Analytics
- Introduction to a General Predictive Analytics Process
  (Optional Reading: ReturnToWorkModel_CSPConference02162019_MeiYuNajim.pdf)
- Introduction to R, RStudio, and R Basics
  (Session Management, Arithmetic in R, Comparison Operators, Variables)
- Data Structures - Vector
  (Vector Basics, Operations, Indexing, and Slicing)

❑ Course Introduction

- **Goals and Expectation:**
  - Introductory programming for analytics class which you will gain some basic programming skills in R and Python in order to prepare you for analytics classes in the MScA program
  - This course syllabus lists the following course specific goals:
    - Use R and Python comfortably
    - Write your own codes when the existing off-the-shelf functionality is insufficient
    - Read and learn more about the advanced aspects of R and Python languages and programming **on your own confidently**
  - My personal goals: Get a good sense about R vs. Python; Share more about solving business problems (fish vs. Fishing), analytical thinking; career development, etc.

❑ Course Introduction

- **What is needed to excel in this class:**
  - ✓ A Laptop
  - ✓ **Anaconda Navigator (Optional), RStudio and Jupyter Notebook (installed)**
  - ✓ 10 workshop style classes + 10 quizzes + 10 assignments to gain hands-on programming experience
  - ✓ **Turn on your video**, **Engaging** during the class, and **Practicing** the codes after the class
  - ✓ If you will miss a class or have a late assignment, please **email** me
  - ✓ If you have any questions, feedback, or suggestion, please talk to me

  Note: Academic Integrity, Don't copy other's work

❑ Course Introduction

- **For some students who would like to have more challenges**
  - ✓ Share more data science/analytics references via Canvas
  - ✓ Incorporate and share some of my industry analytics experience throughout the lectures, time allows
  - ✓ Weekly assignments and quizzes to help you to form your own structured practicing time
  - ✓ Assign some bonus problems in some assignments
  - ✓ Consider inviting some speakers from industry during some breaks, time allows **(ideas/topics?)**

❑ Course Introduction

R Programming Outline

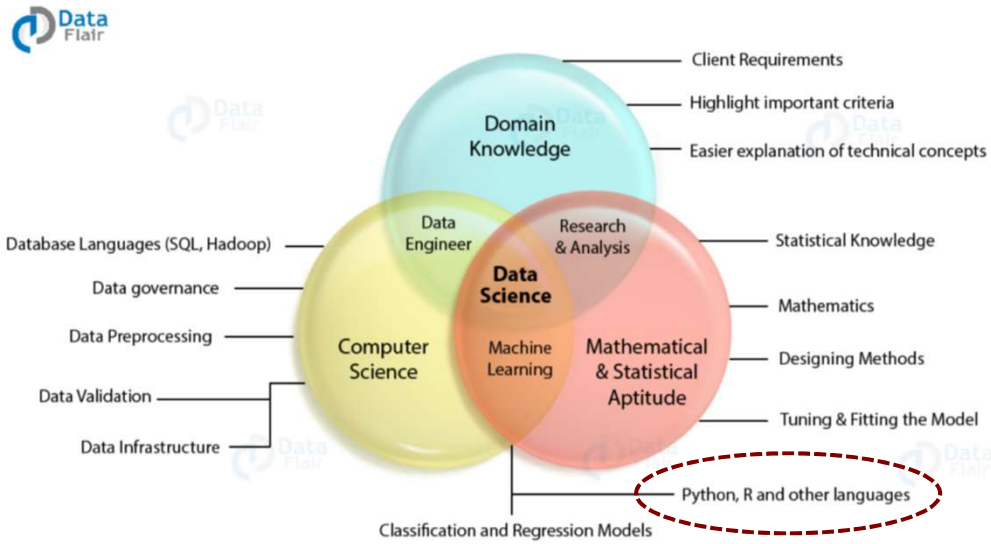
- Week 1 Overview of Analytics and its General Process; Introduction to RStudio; Introduction to R Basics (Session Management, Arithmetic in R, Comparison Operators, Variables); Data Structures - Vector (Vector Basics, Operations, Indexing and Slicing)
- Week 2 Data Structures (Matrixes, Arrays, Lists, Data Frames, Input and Outputs)
- Week 3 Introduction to R Programming Basics (Functions, Conditional Execution, FOR and WHILE Loop)
- Week 4 Data Manipulation (Dplyr, Tidyr, Pipe Operator)
- Week 5 Data Visualization (ggplot2); High Level Introduction to Machine Learning in R; Memory Management

Week 1 Class Agenda

- Course Introduction
- Overview of Analytics
- Introduction to a General Predictive Analytics Process
  (Optional Reading: ReturnToWorkModel_CSPConference02162019_MeiYuNajim.pdf)
- Introduction to RStudio and R Basics
  (Session Management, Arithmetic in R, Comparison Operators, Variables)
- Data Structures - Vector
  (Vector Basics, Operations, Indexing, and Slicing)

❑ Overview of Analytics

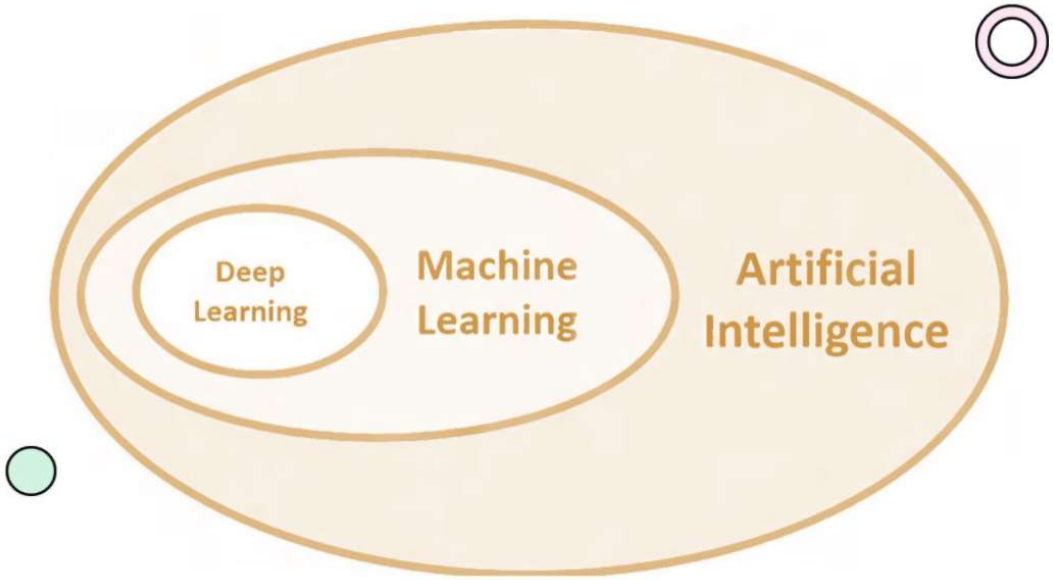
What is Data Science?



Source: Data Flair

❑ Overview of Analytics

The Subsets of Data Science?



Source: https://www.how2shout.com/technology/difference-between-ai-machine-learning-deep-learning.html

❑ Overview of Analytics

Data Science Demand Insights by IBM

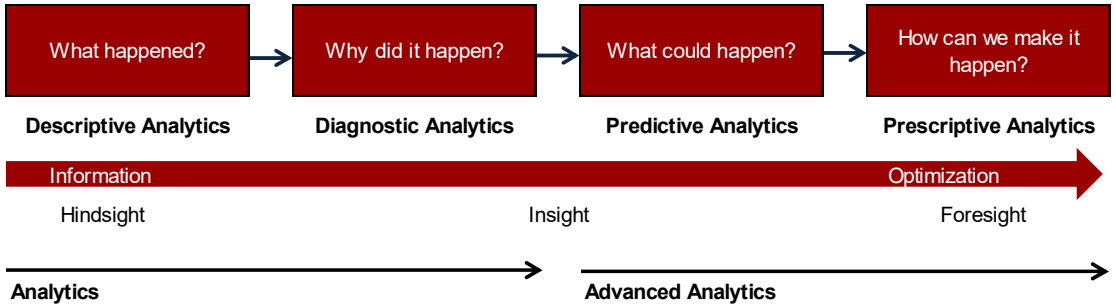
Finance and professional services are the main demand zones for data scientists. The total data scientist & advanced analytics demand in these fields is 54%, which is the highest among all the industrial sectors. Finance industries account for 19% of the openings whereas professional services account for 18% and IT sector holds 17% of the total data science openings

DSA Framework Category	Professional Services	Finance & Insurance	Manufacturing	Information	Health Care & Social Assistance	Retail Trade
Data-Driven Decision Makers	23%	17%	16%	10%	6%	6%
Functional Analysts	23%	34%	9%	5%	8%	4%
Data Systems Developers	41%	14%	14%	10%	5%	3%
Data Analysts	34%	25%	9%	6%	7%	3%
Data Scientists & Advanced Analysts	31%	23%	12%	10%	6%	4%
Analytics Managers	21%	41%	9%	9%	6%	3%

❑ Overview of Analytics

Different Types of Analytics

- While the traditional analytics that comprise business intelligence (BI) examine historical data, advanced analytics focus on forecasting future events and behaviors, allowing businesses to conduct what-if analyses to predict the effects of potential changes in business strategies
- Advanced Analytics != advanced & complicated programming



❑ Overview of Analytics

Some Commonly Referred Types of Data Science/Analytics

- **Data Analysis** is an interactive process of a person tackling a problem, finding the data required to get an answer, analyzing that data to gain useful insights from the data , and interpreting the results in order to provide a recommendation for action or better decision making process.
- **Reporting:** the process of organizing and summarizing data in an easily readable format to communicate important information. Reports help organizations in monitoring different areas of performance and improving results (e.g.: customer satisfaction).
- **Business Intelligence** operations provide various data analysis capabilities that rely on data aggregation as well as focus on the domain expertise of businesses. In Statistical applications, business analytics can be divided into **Exploratory Data Analysis (EDA) and Confirmatory Data Analysis (CDA)**. EDA focuses on discovering new features in the data and CDA focuses on confirming or falsifying existing hypotheses.
- **Data Mining** is a popular type of data analysis technique to carry out data modeling as well as knowledge discovery that is geared towards predictive purposes.
- **Predictive Analytics** forecasts or classification by focusing on statistical or structural models while in text analytics, statistical, linguistic and structural techniques are applied to extract and classify information from textual sources, a species of unstructured data. All these are varieties of data analysis.

❑ Overview of Analytics

Some Commonly Referred Types of Data Science/Analytics

- **Data mining** also called data or knowledge discovery means analyzing data from different perspectives and summarizing it into useful information which we can use to make important decisions. It is the technique of *exploring, analyzing, and detecting patterns* in large amounts of data. The goal of data mining is either classification or prediction.

  1. Classification of Trees: The various tree-shaped structures denote the set of executable decisions.

  2. Logistic Regression: It predicts the probability of an outcome that can only have two values.

  3. Neural Networks: These are non-linear predictive models that resemble biological **neural networks** in structure and are learned through training.

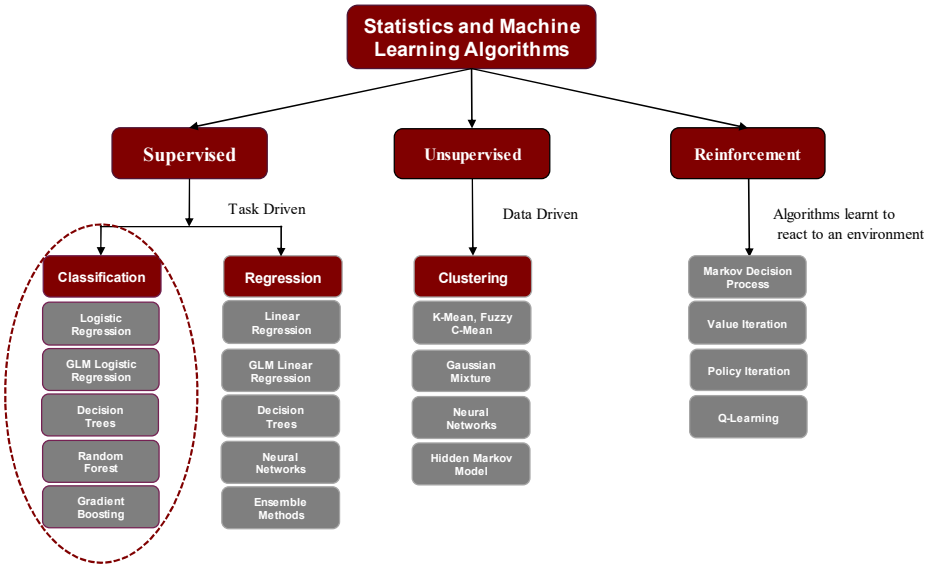
  4. Clustering Techniques like the K-nearest Neighbors: This is the technique that classifies each record in a dataset based on a combination of classes of the *k* record(s) that are most similar to it in a historical dataset (where *k* 1). Sometimes we call it the *k*-nearest neighbor technique.

  5. Anomaly Detection: The identification of items, events and other observations that do not observe a standard pattern in the dataset.

❑ Overview of Analytics

Common Statistical Methods and Machine Learning Algorithms

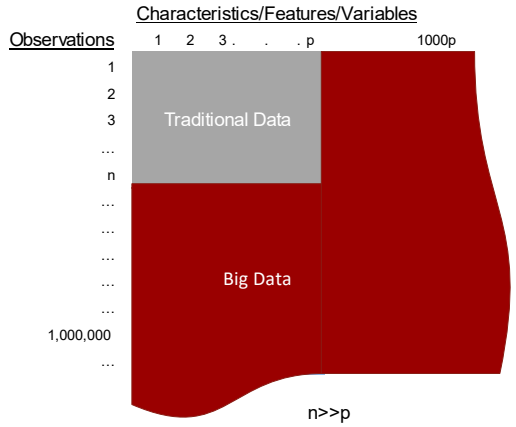
Machine learning is developed based on statistical methods. It is a subset of artificial intelligence that describes systems that can "learn" without human intervention – this is not the focus of this course.



❑ Overview of Analytics

Traditional Data vs. Big Data

- Big Data (Volume, Variety, and Velocity) → Big Data Strategy (Value)
  - Volume: MB→ GB→ TB → PB
  - Variety: Structure, Semistructure, Unstructure, Photo, Web, Audio, Social, Video, Mobile
  - Velocity: Batch → Periodic → Near Real Time → Real Time → Streams



❑ Overview of Analytics

Big Data Analytics

- Big Data Analytics has transformed the way industries perceived data. Traditionally, companies made use of IT data capturing systems, statistical tools and surveying to gather data and perform analysis on the limited amount of information. Most of the times, the deductions and inferences that were produced based on the information were not adequate and did not lead to positive results. Because of this, companies had to incur losses.
- With the advancements in technology and a massive increase in the computational capabilities contributed by High-Performance Computing, industries are able to expand their domain of knowledge. What comprised of a few gigabytes in the past is now in the size of quintillions. This is contributed by the massive expanse in mobile phones, IoT devices and other internet services. To make sense of this, industries have resorted to Big Data Analytics.
- **A Big Data Analytics platform** is a comprehensive platform that provides both **the analytical capabilities** as well as **massive storage capacity**. Some popular Big Data tools like **Hadoop**, **Spark**, **Flink** and **Kafka** have the capability to not only store massive bulk of data but also perform analysis on the data. As a result, they provide comprehensive solutions to companies with their big data needs.

❑ Overview of Analytics

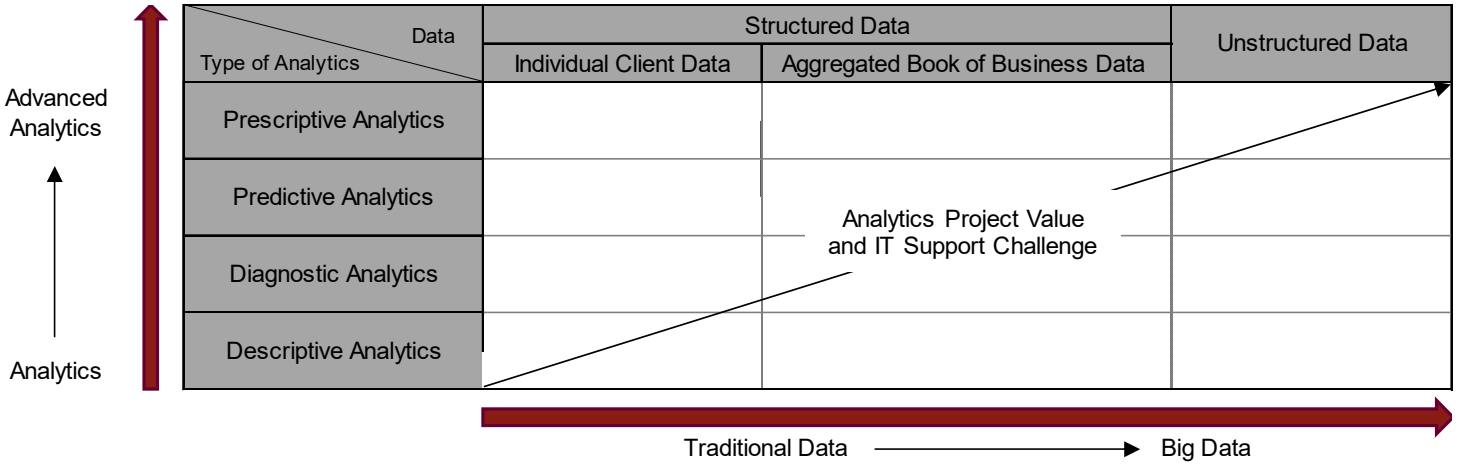
Big Data Examples

- Facebook: stores 300 **PB** data, with an incoming daily rate of about 600 **TB**. (as of 2016)
- YouTube: 1000 PB video storage, 100 M views/day
- Google: 4M searches/minutes, stores 10 EB data(estimation)
- AT&T: 1.9 T phone call records, 70,000 calls/second
- US Credit cards: 1.4 B cards, 20 B transactions/year

❑ Overview of Analytics

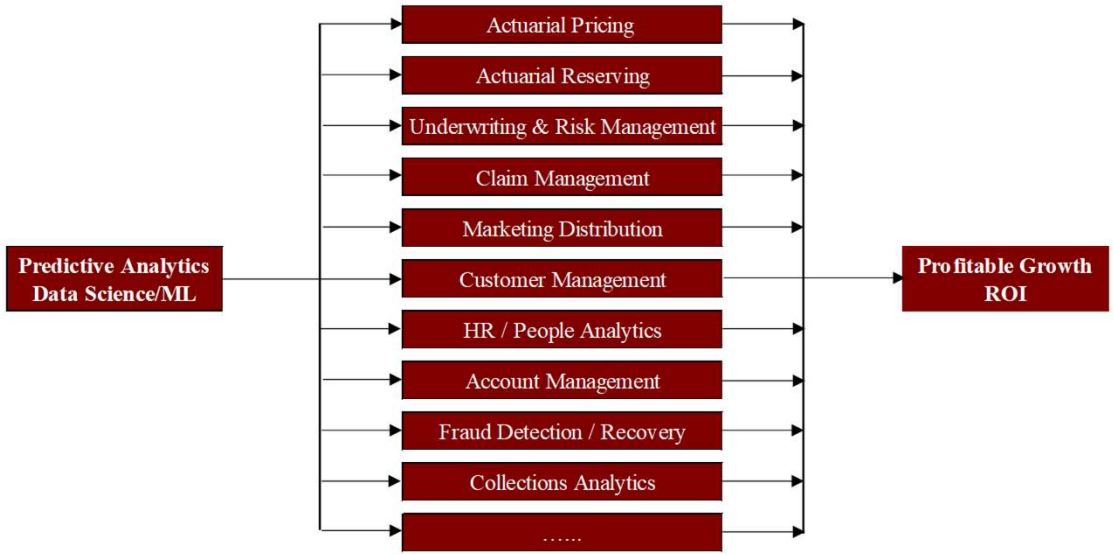
Analytics Project Value and IT Support Challenge

A super data scientist != strong advanced analytics/data science capability



❑ Overview of Analytics

Predictive Analytics Applications In Insurance and Banking

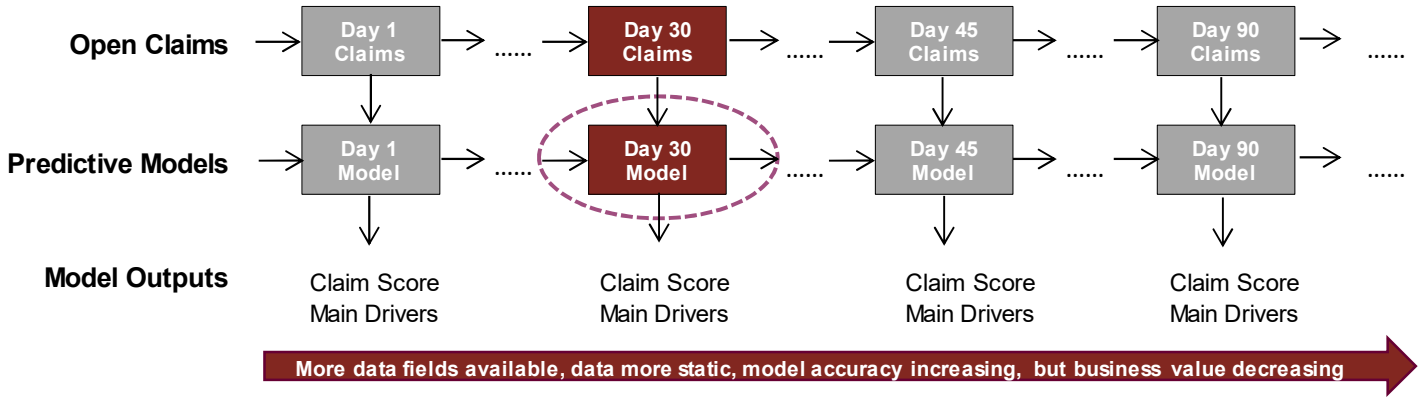


Note: In the Property & Casualty Insurance industry, predictive analytics and machine learning have increasingly penetrated into each of the core business operations – marketing, underwriting, actuarial pricing, actuarial reserving, claims, and risk management, etc. The banking industry is similar. Advanced analytics is becoming a more important way to grow revenue and increase profit in the financial industry.

❑ Overview of Analytics

Near Real-Time Example 1:

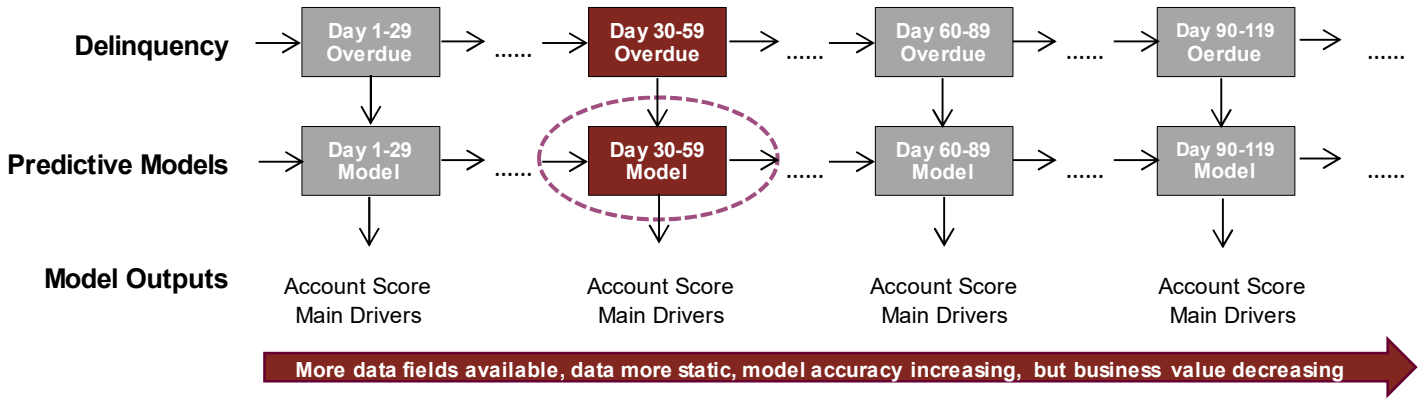
Insurance Claim Predictive Models



❑ Overview of Analytics

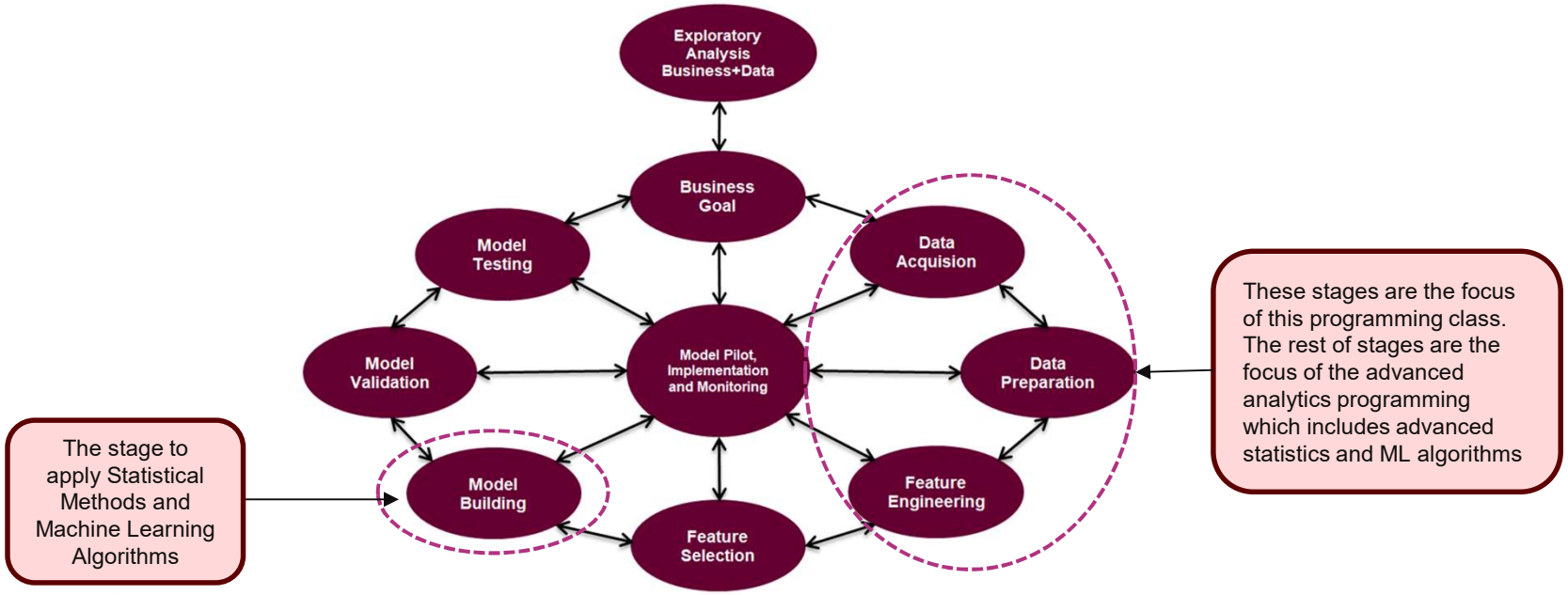
Near Real-Time Example 2:

Banking Delinquency Predictive Models



❑ Overview of Analytics

A Life Cycle Predictive Analytics Process Overview



❑ Overview of Analytics

Technical & Business Skills for Data Science & Analytics

- Technical skills: 1. Packages and Statistical Methods
  2. BI Platform and Data Warehousing
  3. Base Design of Data
  4. Data Visualization and Munging/Wrangling – part of this course
  5. Reporting Methods
  6. Knowledge of Hadoop and MapReduce
  7. Data Mining
- Business Skills: 1. Effective communication skills
  2. Creative thinking
  3. Industry knowledge
  4. Analytic problem solving

❑ Overview of Analytics

Popular Analytics Tools

- Data Manipulation Tools: SQL, R, Python, or SAS
- Statistics and Machine Learning Tools: R, Python, or SAS

→ Powerful Analytics Open Source Tools: R + Python

Week 1 Class Agenda

- Course Introduction
- Overview of Analytics
- Introduction to a General Predictive Analytics Process
  (Optional Reading: ReturnToWorkModel_CSPConference02162019_MeiYuNajim.pdf)
- Introduction to R, RStudio, and R Basics
  (Session Management, Arithmetic in R, Comparison Operators, Variables)
- Data Structures - Vector
  (Vector Basics, Operations, Indexing, and Slicing)

❑ Introduction to R

R Programming Definition and Background

- R is a high-level computer language and environment for statistics and graphs
  - Performs a variety of simple and advanced statistical methods
  - Produces high quality graphics
  - R is a computer language so we can write new functions that extends R's uses
- R was initially written by **R**oss Ihaka and **R**obert Gentleman at the Department of Statistics of the University of Auckland in Auckland, New Zealand (hence the name)
- R is an implementation of the **S** programming language which was created by John Chambers. R is a free open source software maintained by several contributors. Including an \R Core Team" of 17 programmers who are responsible for modifying the R source code. The official R home page: http://www.r-project.org

❑ Introduction to R

R Free Sources

- R is open source so please use the following free resources**:**
  For help with R code: https://support.rstudio.com/hc/en-us/articles/200552336
  For Open Source product questions: community.rstudio.com
  For support with RStudio professional products: Priority Email Support
  To report a bug in the RStudio IDE: https://github.com/rstudio/rstudio/issues
  Quick-R: Intro to R for SAS/SPSS users: https://www.statmethods.net/
  Other Helpful Links: search.r-project.org
  Rseek.org

❑ Introduction to R

R System and Installation

- The R system can be installed on, Windows, Mac or Linux. You will want to install the base system by visiting the R website (https://www.r-project.org/) to follow the installation directions.
- RStudio is an Integrated Development Environment (**IDE**) for R. RStudio is available in open source and commercial editions and runs on the desktop. It has better user-friendly interface
- **To run RStudio, you need to download R from CRAN first** (Download RStudio: www.rstudio.com)
- Anaconda is recommended to be downloaded first (optional), then download Rstudio and Python. Once Anaconda is downloaded, you can access the RStudio through the Anaconda Navigator. For more information on downloading Anaconda, visit: https://docs.anaconda.com/anaconda/install/

  Or, please refer to the Anacoda installation guides I have provided for this course

R Package

- Packages are a collection of functions, examples and documentation that usually focus on a special task and is designed to be reusable by other developers.
- The base system contains some packages. To install a package, from the Comprehensive R Archive Network (CRAN) >install.packages("gplots")
- To get an overview of an installed package and its contents: >help(package="gplots")
- Before using the contents of the package we need to load it, see the R website for a complete list of contributed packages >library(gplots)

❑ Introduction to R

Example - Package Documentation

- Example: We are interested in making a flow chart. After browsing the list of contributed packages on the R website we decide to use diagram.

  Step 1: Download, install and load the package diagram

  >install.packages("readr")

  >library(readr)

  Step 2: List of functions with brief descriptions in the diagram package

  >help(package="readr") – not working in RStudio

  >help(package="readr") – working in RStudio

  Note: Pay attention to the slightly different quotation marks between two cases

❑ Introduction to R

Getting Help

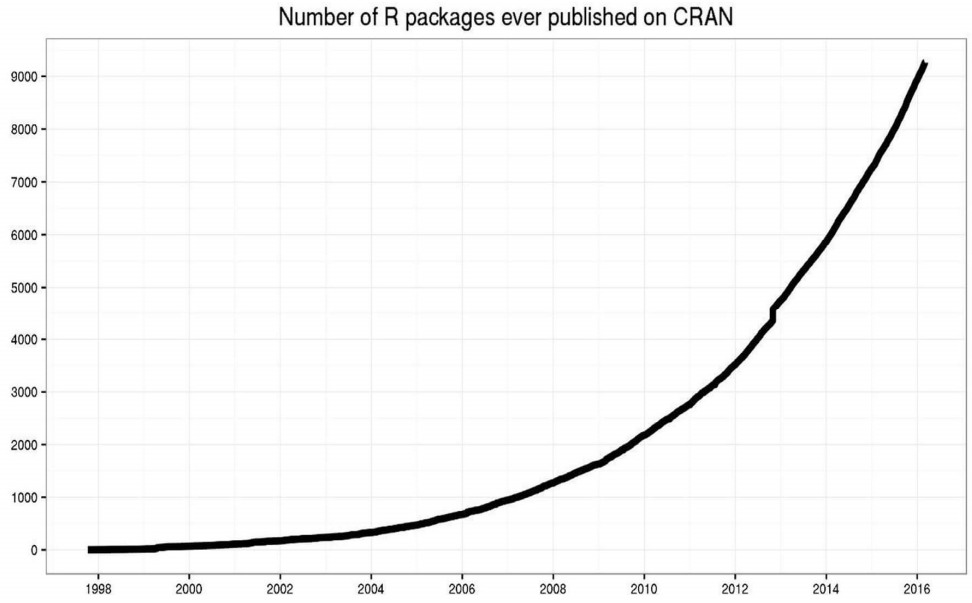
- General help: >help.start()
- To look for a specific function from a loaded package: >?function name or help(function name)
- To look for a symbol put quotations around the symbol, i.e. >? ":" or >help(var) or >help("+")
- To search all installed packages for key words, >??"key words" or >help.search("key words")
- To run the example included with the documentation, >example(function name)
- There are also demos included with the base system and some packages. To see a complete list of demos in the base system, >demo() To run a particular demo, >demo("topic")
- Know the name of the linear regression function, lm, but have a question about the syntax, >?lm
- On the other hand if we don't know what function to use, >help.search("regression")
- To find out the arguments of a function, use the args() function: >args(var)
- Search a function that only know part of the function name: >apropos("len")

R Documentation

- R function documentation contains:
  - A general description of the function
  - A list and description of the function's formal arguments and default settings
  - Details about the function
  - A list and description of values returned by the function
  - References
  - An executable example
- Sometimes the documentation is very complete and other times it is vague. It is more of a quick reference and not intended for instruction.
- Better educational resources: R manuals, contributed manuals, FAQs available on the R website and mailing lists.
- Start to create your own documentation while you are learning

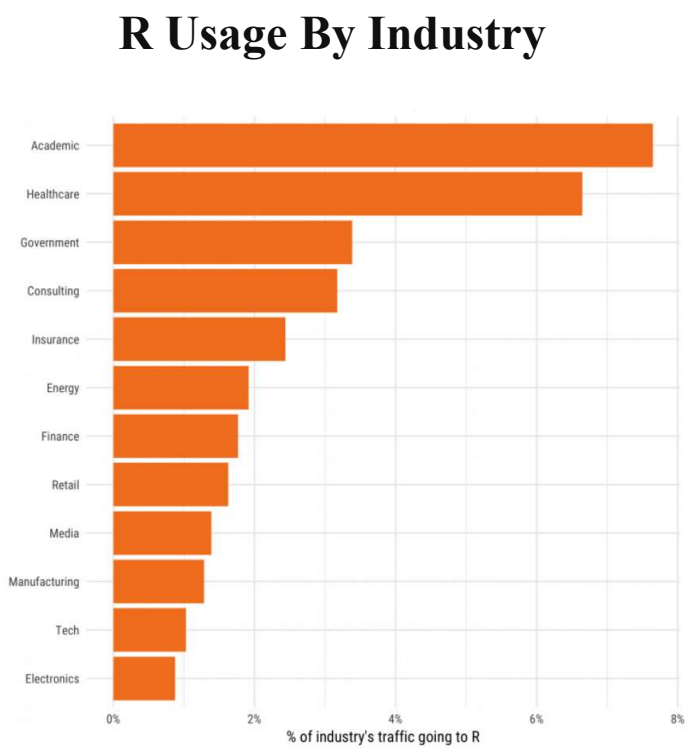
❑ Introduction to R

R is an Open Source Project



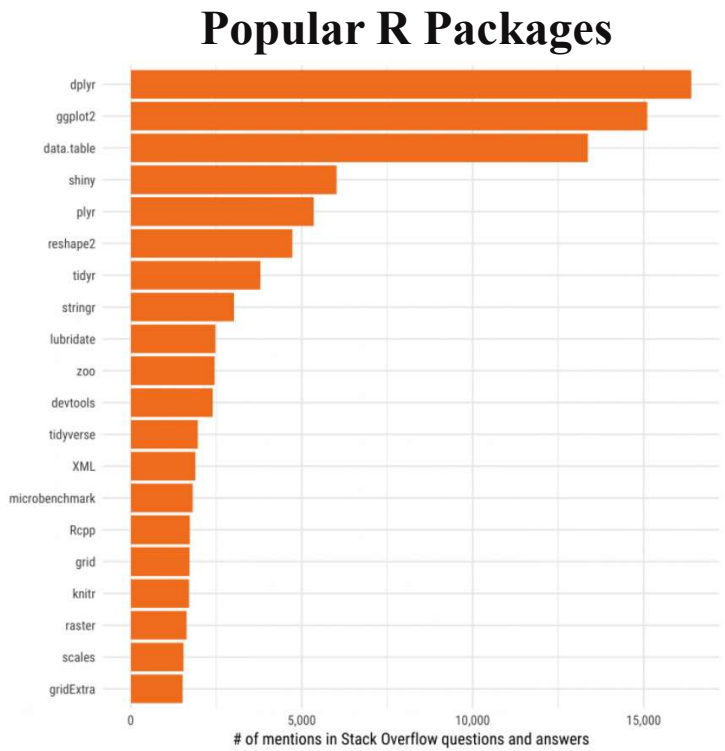
source

❑ Introduction to R



Note:
1) Visits to R By industry: based on visits to Stack Overflow questions from the US/UK in Jan-Aug 2017. The denominator is the total traffic from that industry
2) source

❑ Introduction to R



Note:
1) Most mentioned R Packages in Stack Overflow Q&A in non-deleted questions and answers up to September 2017
2) source

❑ Introduction to R

Some Popular R Libraries/Packages for Data Science

- **dplyr** - Data wrangling and data analysis for data frame
- **tidyr** - Data cleaning and tidying
- **stringr** - Data preparation and cleaning with strings (Text mining)
- **ggplot2** - Data visualization
- **plotly** - Data visualization for building interactive graphs and embed them on web applications
- **tidyverse** – Including packages: ggplot2 and dplyr, etc.
- **MICE** (Multivariate Imputation via Chained Sequences) – Dealing with missing values
- **rpart** – Decision trees
- **caret** (*Classification and Regression Training)* - model complex regression and classification problems
- **randomForest** – Creating a large number of decision trees to building random forest

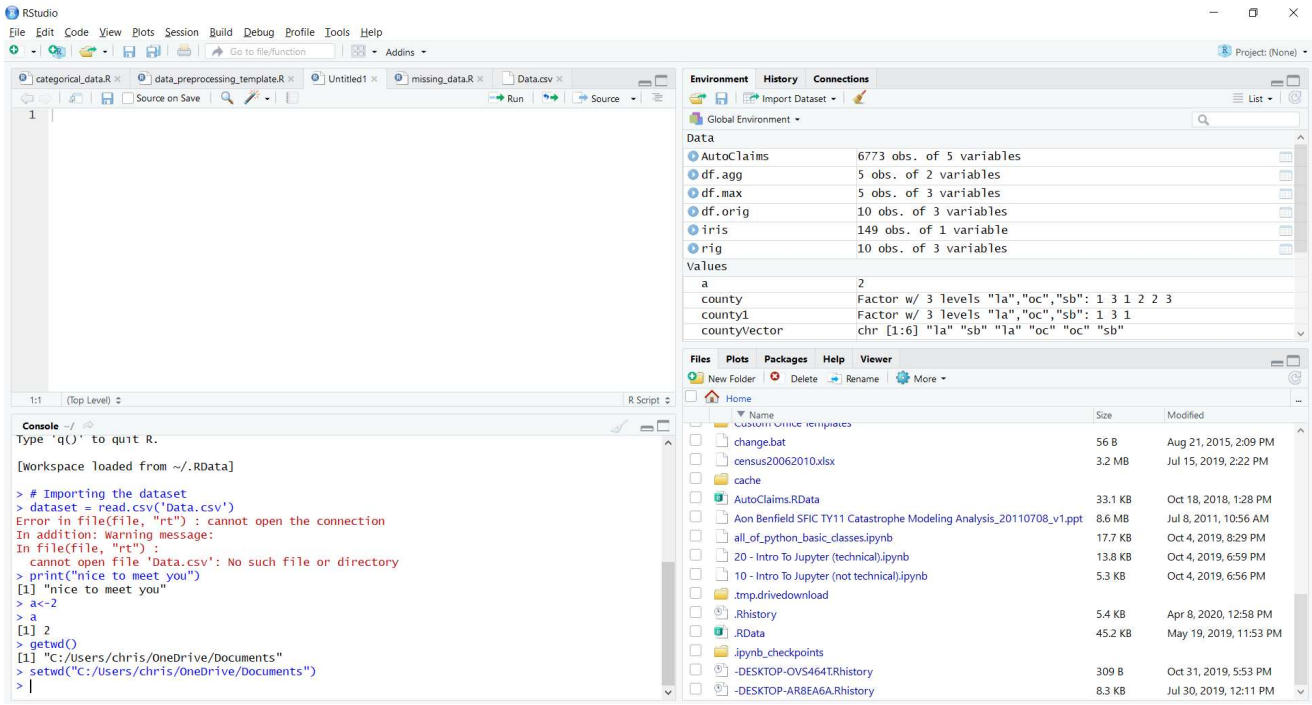
❑ Introduction to RStudio

What is RStudio

- RStudio is an open-source Integrated Development Environment (IDE) that facilitates statistical modeling as well as graphical capabilities for R. It makes use of the QT framework for its GUI features.
- There are two versions of RStudio – **RStudio Desktop** and **RStudio Server**.
- RStudio desktop provides facilities for working on the local desktop environment, whereas RStudio Server provides access through a web browser.

❑ Introduction to RStudio

RStudio Introduction



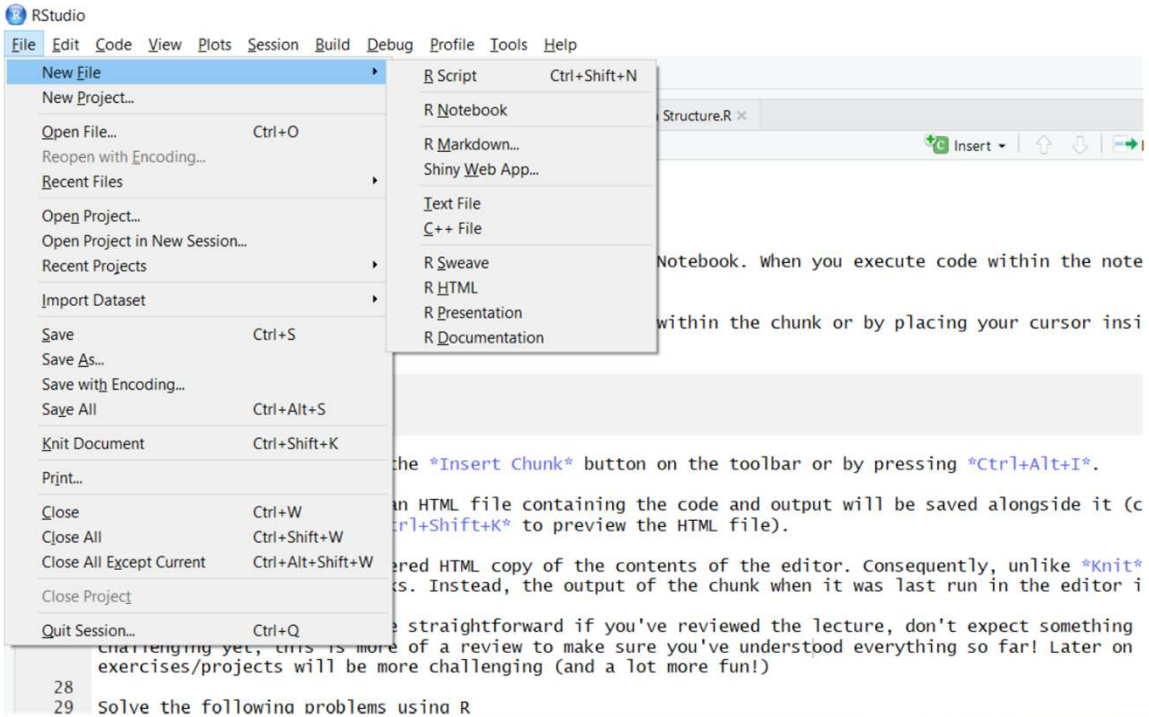
❑ Introduction to RStudio

R Console

- The R Console is where computations are performed. An expression is inputted into the console and the expression is evaluated. Depending on the expression, the system may respond by outputting results to the console or creating a graph in a new window. Then another expression is inputted and evaluated.
- An R session is this interaction between you and the system
- To get the last expression entered use the up arrow
- To get the value of the last evaluated expression type .Last.value
- Press Esc to stop evaluating the current expression

❑ Introduction to RStudio

RStudio Introduction



❑ Introduction to R Basics

Arithmetic Operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
^ or **	Exponentiation

❑ Introduction to R Basics

Calculator

- Enter a math expression and the result is outputted to the console

Binary Operators	+	-	*	/	^	%%
Math Functions	abs	sqrt	log	exp	log10	factorial
Trig Functions	sin	cos	tan	asin	acos	atan
Rounding	round	ceiling	floor	trunc	signif	zapsmall

```
> 5 %% 4          [1] 1      # %% is for modular arithmetic
> log(2) + exp(5) [1] 0.6931472
> ceiling(3.2)    [1] 4
```

Note: See Week 1 in class exercises for examples

Logical Operators

Operator	Description
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Exactly equal to
!=	Not equal to
!x	Not x
x&y	x and y
isTRUE(x)	Test if X is TRUE

Note: See Week 1 in class exercises for examples

❑ Introduction to R Basics

Objects and Functions

- What we normally do is create objects and then perform functions on those objects (functions are also considered objects, more on this later)
- Call a function by ***function name*** (list of arguments separated by commas)

- Each function has a set of formal arguments some with default values; these can be found in the function's documentation **>mean()**
- A function call can include any subset of the complete argument list
- To specify a particular argument use the argument's name
- **Arguments do not have to be named** if they are entered in the same order as the function's formal argument list. However, to make your code easier to understand it is usually a good idea to name your arguments
- When specifying values for an argument use an "**=**"

❑ Introduction to R Basics

Expressions and Assignments

- Commands in R are either expressions or assignments
- Commands are separated by either a semi-colon or a new line
- An expression command is evaluated and normally printed

  - **Assign an object a name** "x" using either "x **<-** object" or "x **=** object"

  - Use ***print***() function or typing the name of an object to print this object or an object summary

- An assignment command evaluates an expression and passes the value onto a variable but the result is not printed.

  - Example:

```
> a<-5*pi
> a              # call variable a to print
  [1] 15.70796
> print(a)       # use print() function to print
  [1] 15.70796
```

❑ Introduction to R Basics

Naming Conventions

- R names are **CASE SENSITIVE**.
- R names for objects can be made up from

  - Upper and/or lowercase letters; Digits: 0 to 9, in any non-initial position

  - Period and underscores usually used to separate words in names

  - Object names cannot start with an underscore
- **Avoid using system names** such as: c, q,s,t, C, D, F, I, T, diff, length, mean, pi, range, rank, time, tree, and var
- You are **not allowed to use the following reserved words**: FALSE, Inf, NA, NaN, TRUE, break, else, for, function, if, in, next, repeat, return, and, while

❑ Introduction to R Basics

Examples - Assigning Objects & Function Calls

- Example 1: Find the mean of a set of numbers: First assign the vector of numbers a name ***x*** and then call the function ***mean***().

```
> x = c(0,5,7,9,1,2,8);  >x     [1] 0 5 7 9 1 2 8
> mean(x)                       [1] 4.571429
> X                             Error: object 'X' not found
```

- Example 2: Sort a vector y so that the numbers are descending. By default R will sort ascending so I need to change the formal argument decreasing to TRUE (the default value for decreasing is FALSE)

```
> y <- c(4,2,0,9,5,3,10);   >y
  [1] 4 2 0 9 5 3 10
> sort(y);   sort(y, decreasing=TRUE)
  [1] 0 2 3 4 5 9 10          [1] 10 9 5 4 3 2 0
```

❑ Introduction to R Basics

options() Function

- The function options() sets several global options that affect how R computes and displays results.
- To look at the value of a single option, getOption("option name")
- Options reset to the defaults with each new R session, even if you save the workspace.
- Example: we want to change the maximum number of digits printed from 7 (default) to 15.

```
> defaults <- options()     # Save all default options
> getOption("digits")       [1] 7
> pi                        [1] 3.141593
> options(digits=15)        # Change to 15 digits
> pi                        [1] 3.14159265358979
> options(defaults)         # Restore all default options
```

❑ Introduction to R Basics - Session Management

Work Directory

- The working directory is the directory in which R will, by default, look for and save files. The default choice of a working directory, usually an R installation directory, is not a good choice for storing data. When we load/save datasets, load source files or save graphs **we will need to specify the file path**. To avoid typing the path every time we can specify a working directory
- To know the current working directory, type: >getwd()
- To see the contents of the working directory, type: >dir()

  To set the working directory click File > Change dir... or type:
  >setwd("C:/Current/TeachingUChicago/Spring2020")
- **Best Practice**: Use a separate working directory for each different project and to save the important objects into an image file in the working directory

❑ Introduction to R Basics - Session Management

Workspace

- The workspace is the collection of R objects that are listed upon typing ls() or objects() where all the objects you create during a session are located
- Depending on what you are working on, it is usually wiser to write a script and then run the entire script with each new session. This way you know exactly what objects you have created. If a particular object is very important then save it to a file.
- Managing the Workspace:

  - To list what variables you have created in the current session, ls()
  - To see what libraries and dataframes are loaded, search()
  - To see what libraries have been installed, library()
  - To remove an object, rm(object names)
  - To remove all objects, rm(list=ls())

❑ Introduction to R Basics - Session Management

Script Editor

- All the R commands are typed after the R prompt
- Most of time, we write the R code in a script file
- To create a script file, click on ***File → New Script***. Then you can type all your R commands in the script file
- You can run a segment of your script file by highlighting the R codes and clicking on the run icon
- To save your script file, click on ***File → Save*** or ***File → Save as***…and use *.R as the file extension
  Example: my-Rcode.R

❑ Introduction to R Basics - Session Management

Saving Objects

- You can save the workspace to a file at any time by using the ***save.image***() function. All the objects from the workspace will be saved to the file in your working directory.
  Example: >save.image("Week1.Rdata")?

- If you only want to save only a few selected objects, you can use the ***save***() function
  Example: >save(a1, b2, file="practice.Rdata")

- The .RData file can be loaded by default if you double click on the file
  Example: >load("practice.Rdata")
  >ls()
- When you close R, you will be prompted to \Save workspace image?" If you say yes, the next time you open R from the same working directory the workspace will be restored. That is all of the objects you created during your last session will still be available. Also see ***save.image***()

❑ Introduction to R Basics - Session Management

Sample Script

```
# Calculate the mean of x
  >x = c(0,5,7,9,1,2,8);  >mean(x)
# R views the first line as a complete expression. Thus the code is treated as two separate expressions
  instead of one long expression
  > long.answer.name <- 500*factorial(long.variable.name)
                                + sqrt(really.long.variable.name)

# Here the first line is not a complete expression (trailing + sign) so R continues reading lines of code until
  the expression is complete
  > long.answer.name <- 500*factorial(long.variable.name) +
                                sqrt(really.long.variable.name)
# Writing two expressions on the same line requires a ;
  > mean(x); var(x)
```

Week 1 Class Agenda

- Course Introduction
- Overview of Analytics
- Introduction to a General Predictive Analytics Process
  (Optional Reading: ReturnToWorkModel_CSPConference02162019_MeiYuNajim.pdf)
- Introduction to RStudio and R Basics
  (Session Management, Arithmetic in R, Comparison Operators, Variables)
- Data Structures - Vector
  (Vector Basics, Operations, Indexing, and Slicing)

❑ Data Structures

Data Structures

- **Vectors**: A vector is a one-dimensional array and an ordered collection of objects of the **same type**
- **Matrices**: A matrix is just a two-dimensional generalization of a vector
- **Arrays:** An array is a multi-dimensional generalization of a vector
- **Lists**: A list is a general form of a vector, where the elements don't need to be of the same type or dimension.
- **Dataframes**: R refers to datasets as dataframes

❑ Data Structures - Vector

- A vector is an ordered collection of objects of **the same type**. We can create a vector with all the basic data types. The simplest way to create a vector in R, is to use the **c()** command
- The function **c**(...) concatenates its arguments to form a vector
- Vector Names - use the **names()** function to assign names to each element in our vector
- To create a patterned vector
  - "**:**" sequence of integers
  - **seq()** general sequence
  - **rep()** vector of replicated elements

```
> v1 <- c(2.5, 4, 7.3, 0.1); v1
[1] 2.5 4.0 7.3 0.1
> v2 <- c("A", "B", "C", "D"); v2
[1] "A" "B" "C" "D"
> #: Sequence of integers
> v3 <- -3:3; v3
[1] -3 -2 -1  0  1  2  3
> # seq() General sequence
> seq(0, 2, by=0.5); seq(0, 2, len=6)
[1] 0.0 0.5 1.0 1.5 2.0
[1] 0.0 0.4 0.8 1.2 1.6 2.0
> # rep() Vector of replicated elements
> rep(1:5, each=2); rep(1:5, times=2)
 [1] 1 1 2 2 3 3 4 4 5 5
 [1] 1 2 3 4 5 1 2 3 4 5
```

❑ Data Structure – Reference Elements of a Vector

- Use bracket notation **"[ ]"** with a vector/scalar of positions to index and or access individual elements from a vector/scalar of positions to reference elements
- A **minus sign** before the vector/scalar to **remove** elements
- Slicing - use a colon (**:**) to indicate a slice of a vector. The format is: vector[start_index:stop_index]

Example:

```
> x <- c(4, 7, 2, 10, 1, 0)
> x[4]
[1] 10
> x[1:3]
[1] 4 7 2
> x[c(2,5,6)]
[1] 7 1 0
> x[-3]
[1]  4  7 10  1  0
> x[-c(4,5)]
[1] 4 7 2 0
> x[x>4]
[1]  7 10
> x[3] <- 99
> x
[1]  4  7 99 10  1  0
```

❑ Data Structure – which() and match()

- Additional functions that will return the indices of a vector

  - ***which***() Indices of a logical vector where the condition is TRUE

  - ***which.max***() Location of the (rst) maximum element of a numeric vector

  - ***which.min***() Location of the (rst) minimum element of a numeric vector

  - ***match***() First position of an element in a vector

```
> x <- c(4, 7, 2, 10, 1, 0)
> x>=4
[1]  TRUE  TRUE FALSE  TRUE FALSE FALSE
> which(x>=4)
[1] 1 2 4
> which.max(x)
[1] 4
> x[which.max(x)]
[1] 10
> max(x)
[1] 10
```

```
> y <- rep(1:5, times=5:1);y
 [1] 1 1 1 1 1 2 2 2 2 3 3 3 4 4 5
> match(1:5, y)
[1]  1  6 10 13 15
> match(unique(y), y)
[1]  1  6 10 13 15
> ?match
```

❑ Data Structure – Vector Operations

- When vectors are used in math expressions, the operations are performed **element by element**

  Example:

```
> x <- c(4,7,2,10,1,0)
> y <- x^2 + 1;
> y
[1]  17  50   5 101   2   1
> x*y
[1]   68  350   10 1010    2    0
```

❑ Data Structure – Useful Vector Functions

`sum(x)`	`prod(x)`	Sum/product of the elements of `x`
`cumsum(x)`	`cumprod(x)`	Cumulative sum/product of the elements of `x`
`min(x)`	`max(x)`	Minimum/Maximum element of `x`
`mean(x)`	`median(x)`	Mean/median of `x`
`var(x)`	`sd(x)`	Variance/standard deviation of `x`
`cov(x,y)`	`cor(x,y)`	Covariance/correlation of `x` and `y`
`range(x)`		Range of `x`
`quantile(x)`		Quantiles of `x` for the given probabilities
`fivenum(x)`		Five number summary of `x`
`length(x)`		Number of elements in `x`
`unique(x)`		Unique elements of `x`
`rev(x)`		Reverse the elements of `x`
`sort(x)`		Sort the elements of `x`
`which()`		Indices of `TRUE`s in a logical vector
`which.max(x)`	`which.min(x)`	Index of the max/min element of `x`
`match()`		First position of an element in a vector
`union(x, y)`		Union of `x` and `y`
`intersect(x, y)`		Intersection of `x` and `y`
`setdiff(x, y)`		Elements of `x` that are not in `y`
`setequal(x, y)`		Do `x` and `y` contain the same elements?

Q & A

Thank You

Contact Information:

Your comments and questions are valued and encouraged.

Mei Najim
E-mail: mnajim@uchicago.edu

LinkedIn: https://www.linkedin.com/in/meinajim/