Introduction to Data Science

Daniel Gutierrez, Data Scientist

Los Angeles, Calif.

Course Outcomes

- Provide an overview of Data Science
- Discuss the *Data Science Venn Diagram*
- Define the *Data Science Process*.
- Present an *Introduction to R Programming Part I*
- Start filling your data science toolbox with useful R programming techniques

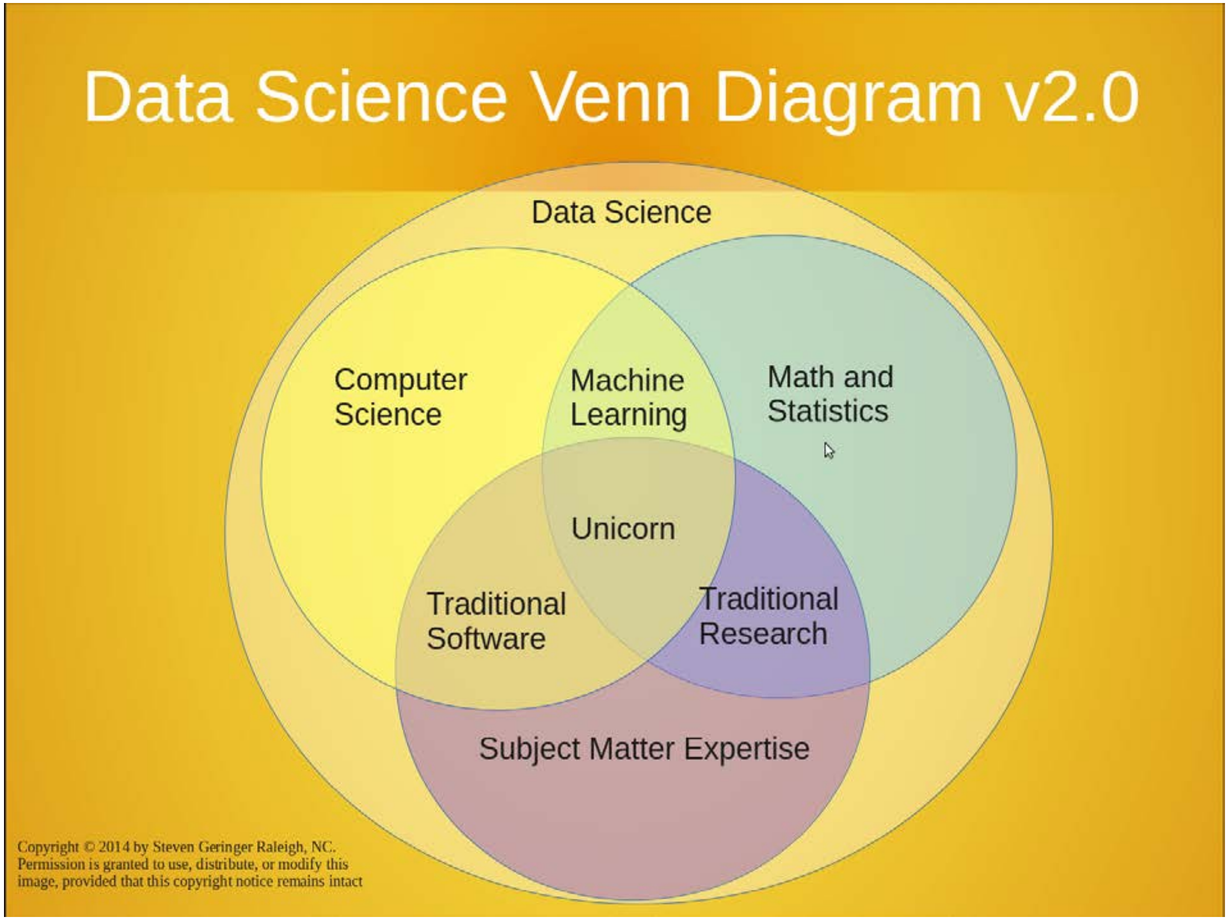
Lesson Objectives

- Fully understand the *Data Science Process*
- Provide a brief history of the R statistical environment
- Discuss installing, configuring and using R and Rstudio
- Talk about writing R scripts using basic language constructs and data types
- Understand the atomic classes in R
- Show how to code assignment statements
- Define a number of useful R objects: vectors, lists, matrices, factors, data frames, arrays
- Show how to create sequences
- Review object attributes such as names and dimensionality
- Discuss the importance of commenting your R code
- Give examples of coercion
- Show how to handle missing values: NA (not available), NaN (not a number) and NULL

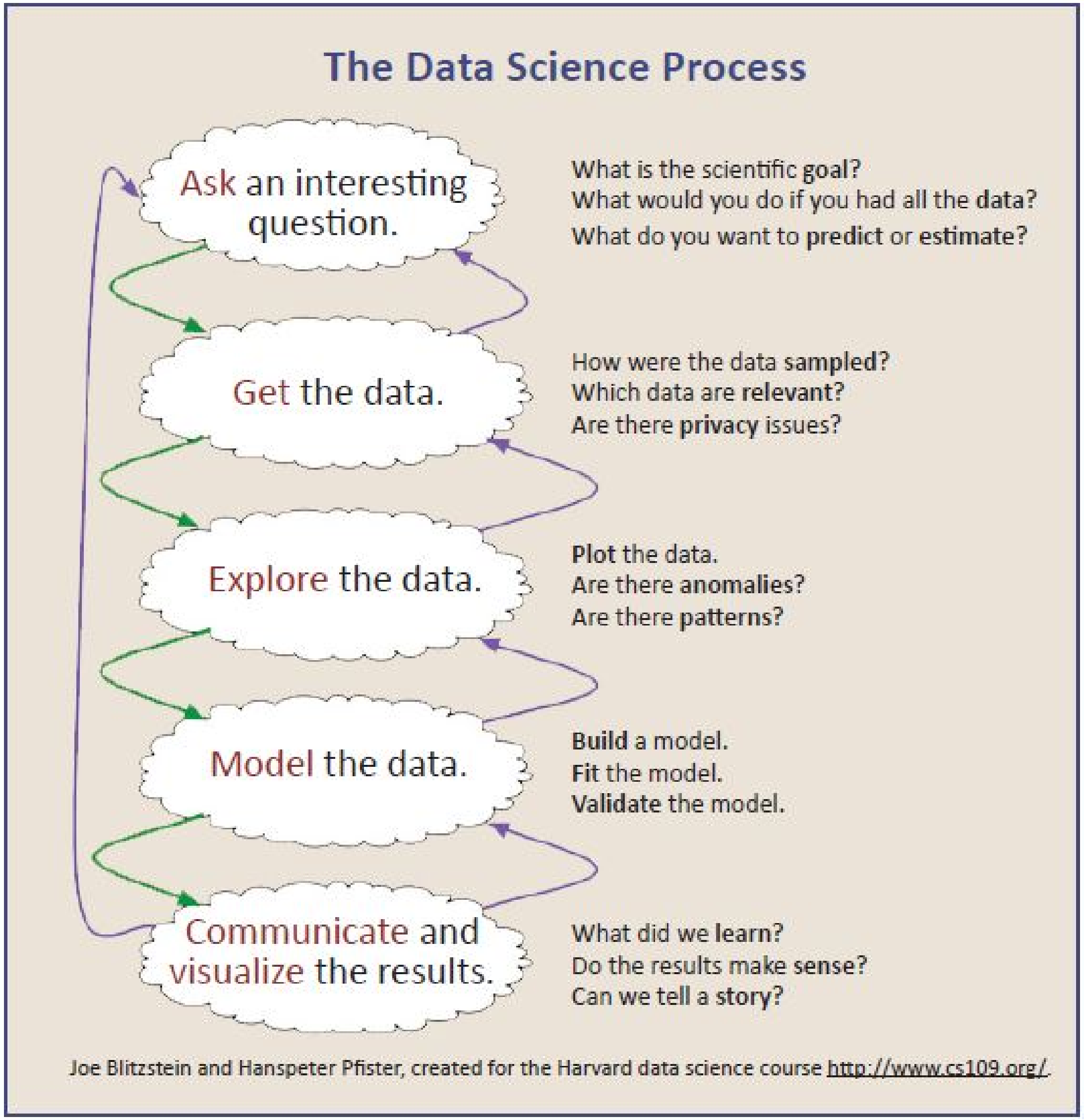
Data Science vs. Data Engineering

- A big confusion exists in our industry. What are the data scientist skillsets?
- Many companies advertise for “data scientist” but expect a vast array of skills
- Many companies actually seek a “unicorn”
- Realistically, these companies need a data science “team” consisting of both data scientists plus data engineers
- This class is for data scientists

The Data Science Venn Diagram



Data Science Process



The Data Science Process

Ask an interesting question.

What is the scientific **goal**?
What would you do if you had all the **data**?
What do you want to **predict** or **estimate**?

Get the data.

How were the data **sampled**?
Which data are **relevant**?
Are there **privacy** issues?

Explore the data.

Plot the data.
Are there **anomalies**?
Are there **patterns**?

Model the data.

Build a model.
Fit the model.
Validate the model.

Communicate and visualize the results.

What did we **learn**?
Do the results make **sense**?
Can we tell a **story**?

Joe Blitzstein and Hanspeter Pfister, created for the Harvard data science course http://www.cs109.org/.

Not So Brief History of R

- What is R?
- Historically, R is a dialect of the S language
- S is a language that was originally developed by John Chambers and others at Bell labs
- S was initiated in 1976 as an internal statistical analysis environment – originally implemented as Fortran libraries
- Early versions of the language did not contain functions for statistical modeling
- In 1988 the system was rewritten in C and began to resemble the system that we have today (this was Version 3 of the language). The book *Statistical Models in S* by Chambers and Hastie (the white book) documents the statistical analysis functionality.

Not so Brief History of R (continued)

- Version 4 of the S language was released in 1998. The book *Programming with Data* by John Chambers (the green book) documents this version of the language
- In 1993 Bell Labs gave StatSci (now Insightful Corp.) an exclusive license to develop and sell the S language
- In 2004 Insightful purchased the S language from Lucent for $2 million
- Insightful sells its implementation of the S language under the product name S-PLUS
- In 2008 Insightful is acquired by TIBCO for $25 million (TIBCO is a prominent player in the R market today)

Not so Brief History of R (continued)

- 1991: Created in New Zealand by Ross Ihaka and Robert Gentleman. Their experience developing R is documented in a 1996 JCGS paper
- 1993: First announcement of R to the public
- 1995: Martin Mächler convinces Ross and Robert to use the GNU General Public License to make R free software
- 1997: The R Core Group is formed (containing some people associated with S-PLUS). The core group controls the source code for R
- 2000: R Version 1.0.0 is released
- 2019: R version 3.5.3 is released on March 11, 2019

Not so Brief History of R (continued)

- Quite lean, as far as software goes; functionality is divided into modular “packages”
- Graphics capabilities very sophisticated and better than most stat packages.
- Useful for interactive work, but contains a powerful programming language for developing new tools (user → programmer)
- Very active and vibrant user community; R-bloggers (www.r-bloggers.com) and Stack Overflow (www.stackoverflow.com)

The R Environment

- The R system is divided into 2 conceptual parts:
    1. The “base” R system that you download from the CRAN website
    2. Everything else!
- The “everything else” consists of many other packages:
    - There are more than 13,914 packages on CRAN that have been developed by users and programmers around the world.
    - People often make special purpose packages available on their personal websites; there is no reliable way to keep track of how many packages are available in this fashion.

The R Environment

- In order to use an R package you must first determine if it exists
- Find the package via Google e.g. “time series in R”
- Use CRAN page such as: http://cran.r-project.org/web/packages/timeSeries/index.html
- Download “Reference Manual” and vignettes (if any)

```
> install.packages(“timeSeries”)
> library(timeSeries)
```

The R Environment

- Available from CRAN (http://cran.r-project.org)
  - The Comprehensive R Archive Network
  - An introduction to R
  - Writing R packages
  - R data import/export
  - R installation and administration (mostly for building R from source code)
  - R internals (not for the faint of heart)

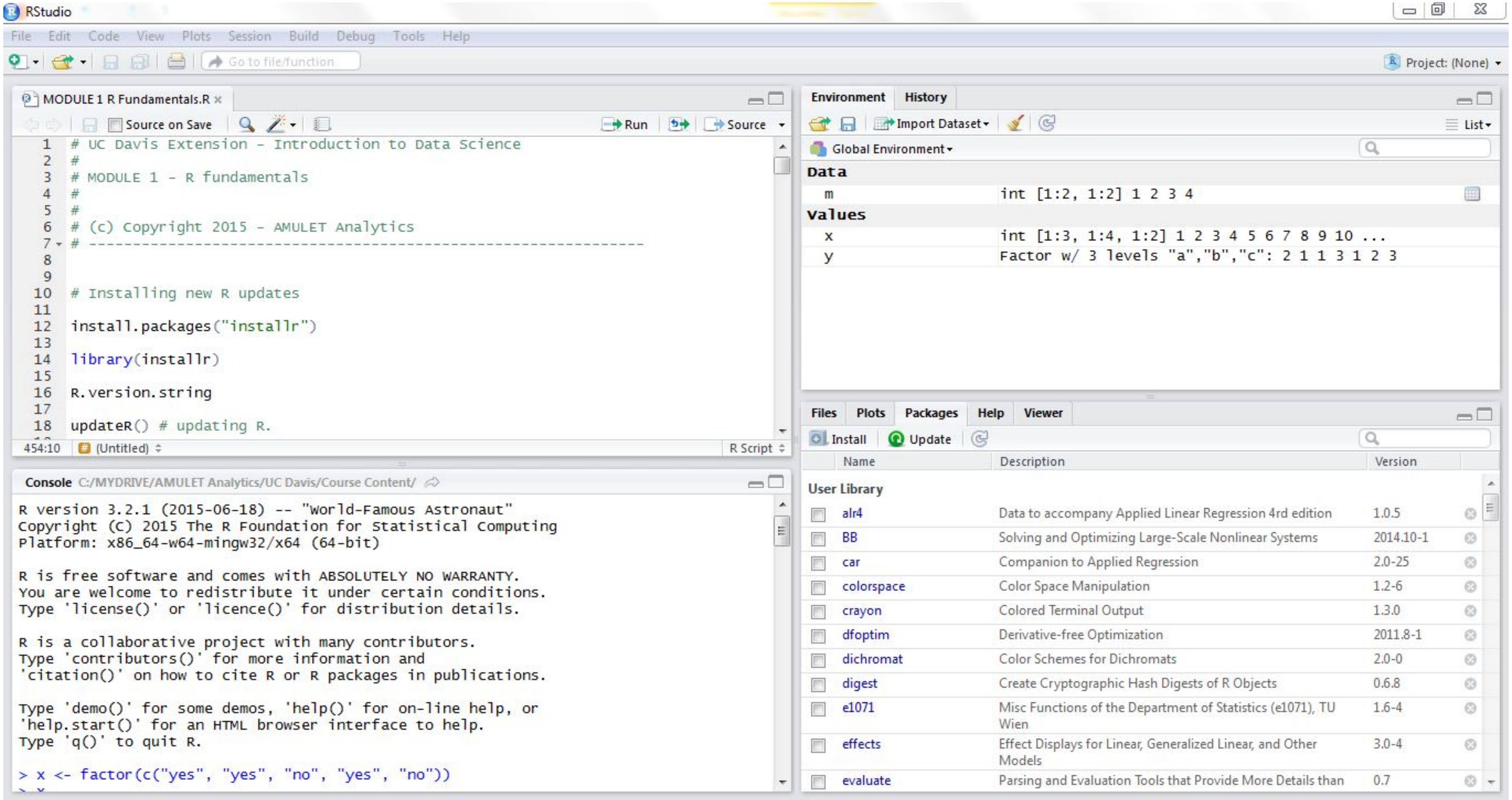
Installing R

- Download R from CRAN (http://cran.r-project.org)
- Select your environment: Windows, Mac, or Linux
- Click on “Install R for the first time”
- Click on “Download R 3.5.3 for Windows” for example
- Follow the useful installation prompts
- NOTE: you must install R before RStudio

Installing RStudio

- Download the RStudio IDE from http://www.rstudio.com/
- Click on “Download RStudio” button
- Click on “RStudio Desktop”
- Click on “Download RStudio Desktop” button
- Click on the installer that matches your system
- Follow the useful installation prompts
- You should now see both the R and RStudio icons on your desktop

R Fundamentals – Part 1



```r
# UC Davis Extension - Introduction to Data Science
#
# MODULE 1 - R fundamentals
#
#
# (c) Copyright 2015 - AMULET Analytics
# ---------------------------------------------------------------


# Installing new R updates

install.packages("installr")

library(installr)

R.version.string

updateR() # updating R.
```

Console C:/MYDRIVE/AMULET Analytics/UC Davis/Course Content/

```
R version 3.2.1 (2015-06-18) -- "World-Famous Astronaut"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> x <- factor(c("yes", "yes", "no", "yes", "no"))
> x
```

```
Data
  m     int [1:2, 1:2] 1 2 3 4
Values
  x     int [1:3, 1:4, 1:2] 1 2 3 4 5 6 7 8 9 10 ...
  y     Factor w/ 3 levels "a","b","c": 2 1 1 3 1 2 3
```

	Name	Description	Version
	User Library		
□	alr4	Data to accompany Applied Linear Regression 4rd edition	1.0.5
□	BB	Solving and Optimizing Large-Scale Nonlinear Systems	2014.10-1
□	car	Companion to Applied Regression	2.0-25
□	colorspace	Color Space Manipulation	1.2-6
□	crayon	Colored Terminal Output	1.3.0
□	dfoptim	Derivative-free Optimization	2011.8-1
□	dichromat	Color Schemes for Dichromats	2.0-0
□	digest	Create Cryptographic Hash Digests of R Objects	0.6.8
□	e1071	Misc Functions of the Department of Statistics (e1071), TU Wien	1.6-4
□	effects	Effect Displays for Linear, Generalized Linear, and Other Models	3.0-4
□	evaluate	Parsing and Evaluation Tools that Provide More Details than	0.7

R Fundamentals – Part 1

- Tips on Developing with R
- On your local machine, set up a distinct folder (directory) for each R project. Don’t mix R projects in the same folder
- Use `Session -> Set Working Directory` in RStudio to point to your project folder
- Use `File -> New File -> R Script` to open up a new R script in the code editor
- Start coding and using the R command line
- Use `Session -> Save Workspace As` to create an environment file containing all your variables. This will be named with the name you provide plus an .RDATA extension
- Next time, start up Rstudio with the .RDATA file to resume

R Fundamentals – Part 1

- R has five basic or “atomic” classes of objects:
    - Character
    - Numeric (real numbers)
    - Integer
    - Complex
    - Logical (True / False)
- The most basic object is a vector
    - A vector can only contain objects of the same class
    - BUT: The one exception is a *list*, which is represented as a vector but can contain objects of different classes (indeed, that’s usually why we use them)

R Fundamentals – Part 1

- At the R command prompt we generally type what are called *expressions*. You type the expression and R evaluates it.
- The `<-` symbol is the assignment operator (very similar in function to = assignment operator found in other programming environments) which assigns what's on the right to what's on the left
- The # character indicates a *comment*. Anything to the right of the # (including the # itself) is ignored as part of the R expression. Commenting your R scripts is highly recommended!

R Fundamentals – Part 1

- Numbers in R are generally treated as numeric objects (i.e., double precision real numbers)
- If you explicitly want an integer, you need to specify the `L` suffix
- Ex: Entering 1 gives you a numeric object; entering `1L` explicitly gives you an integer.
- There is also a special number `Inf` which represents infinity; e.g. 1 / 0; Inf can be used in ordinary calculations; e.g. `1 / Inf` is 0
- The value `NaN` represents as undefined value (“not a number”); e.g. 0 / 0; NaN can also be thought of as a missing value (more on that later)

R Fundamentals – Part 1

- In addition to the atomic types, here are some additional object types available in R:
  - Vectors
  - Lists
  - Matrices
  - Factors
  - Data frames
  - Arrays

R Fundamentals – Part 1

- R objects also can have attributes
  - Names
  - dimnames
  - Dimensions (e.g., matrices, arrays)
  - Class
  - Length
  - Other user-defined attributes/metadata

R Fundamentals – Part 1

- Matrices are vectors with a *dimension* attribute. The dimension attribute is itself an integer vector of length 2 (nrow, ncol)
- Matrices are constructed *column-wise*, so entries can be thought of starting in the “upper left” corner and running down the columns.
- Lists are a special type of vector that can contain elements of different classes. The List is a very important data type in R and you should get to know them well.

R Fundamentals – Part 1

- Factors are used to represent categorical data. Factors can be unordered or ordered. One can think of a factor as an integer vector where each integer has a *label*.
  - Factors are treated specifically by modelling functions like `lm()` and `glm()`
  - Using factors with labels is *better* than using integers because factors are self-describing; having a variable that has values “Male” and “Female” is better than a variable that has values 1 and 2.

R Fundamentals – Part 1

- Missing values are denoted by NA or NaN for undefined mathematical operations (e.g. division by zero).
  - `is.na()` is used to test objects if they are NA
  - `is.nan()` is used to test for NaN
  - NA values have a class also, so there are integer NA, character NA, etc.
  - A NaN value is also NA but the converse is not true

R Fundamentals – Part 1

- Data frames are used to store tabular data
  - They are represented as a special type of list where every element of the list has to have the same length
  - Each element of the list can be thought of as a column and the length of each element of the list is the number of rows
  - Unlike matrices, data frames can store different classes of objects in each column (just like lists); matrices must have every element be the same class
  - Data frames also have a special attribute called `row.names`
  - Data frames are usually created by calling `read.table()` or `read.csv()`
  - Can be converted to a matrix by calling `data.matrix()`

R Fundamentals – Part 1

- Our goal for course Weeks 1 through 4 is to present a collection of R fundamentals and start filling your data science toolbox with a collection of programming techniques

Code Modules

- WEEK 1-1 Code Module – updating R
- WEEK 1-2 Code Module – vectors
- WEEK 1-3 Code Module – coercion
- WEEK 1-4 Code Module – matrices
- WEEK 1-5 Code Module – lists
- WEEK 1-6 Code Module – factors
- WEEK 1-7 Code Module – missing values
- WEEK 1-8 Code Module – data frames
- WEEK 1-9 Code Module – names
- WEEK 1-10 Code Module - arrays

Summary

- In WEEK 1 of Introduction to Data Science, we presented all of the introductory material for the course including the Data Science Venn diagram, and the Data Science Process.
- We also introduced R by giving a short history of the language as well as instructions for installing both R and RStudio.
- Next, we start our journey through the R language by learning some basics.