SIM Global Education

Block 1
Introduction to Data Science Ecosystem and Programming Tools

Learning Outcomes

After completing this topic and the recommended reading, you should be able to:

- Introduce yourself to Data Science and review real-world data examples.
- Gain experience using basic tools and technology for programming, such as notebooks, IDEs and version control using Git.

1. Introduction to Data Science

Data (definition)

- "Facts and statistics collected together for reference or analysis."

  [Oxford English Dictionary]

- "Information, especially facts or numbers, collected to be examined and considered and used to help decision-making, or information in an electronic form that can be stored and used by a computer."

  [Cambridge Dictionary]

- "Factual information (such as measurements or statistics) used as a basis for reasoning, discussion, or calculation."

  [Merriam-Webster]

- "Individual facts, statistics, or items of information."

  [Dictionary.com]

- "Data are individual facts, statistics, or items of information, often numeric, that are collected through observation. In a more technical sense, data are a set of values of qualitative or quantitative variables about one or more persons or objects."

  [Wikipedia]

Data Science (definition)

- "Data science encompasses preparing data for analysis, including cleansing, aggregating, and manipulating the data to perform advanced data analysis. Analytic applications and data scientists can then review the results to uncover patterns and enable business leaders to draw informed insights."

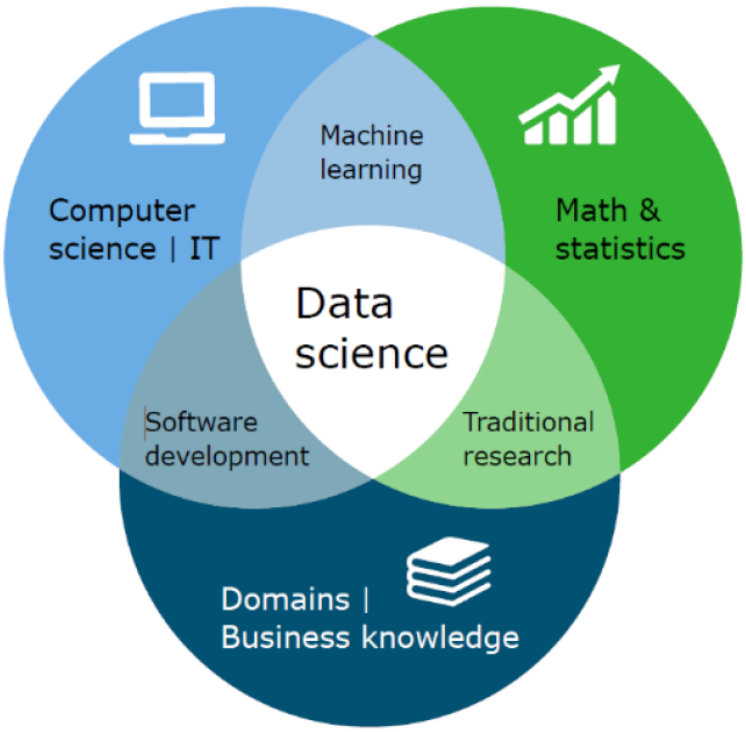
  [Oracle]

- "Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from noisy, structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains."

  [Wikipedia]

Data Science

- ***Data science*** is the application of **computational** and **statistical** techniques on data to address or gain insight into some problem in the **real world**.
- Combination of various techniques
    - Data collection
    - Data pre-processing / processing
    - Big data
    - Scientific hypotheses
    - Business insights
    - Visualisation
    - Machine learning
    - Statistics
    - Etc.



[Source: *https://www.wur.nl/*]

Information (definition)

- "Facts provided or learned about something or someone."

  [Oxford English Dictionary]

- "Facts or details about a situation, person, event, etc."

  [Cambridge Dictionary]

- "Knowledge obtained from investigation, study, or instruction."

  [Merriam-Webster]

- "Knowledge communicated or received concerning a particular fact or circumstance; knowledge gained through study, communication, research, instruction, etc."

  [Dictionary.com]

Data vs. Information

- ***Data***
  - Raw, unorganised facts that need to be processed.
  - Unusable until it is organised.
- ***Information***
  - Created when data is processed, organised, and structured.
  - Needs to be put in an appropriate context in order to become useful.

Data Science Tasks

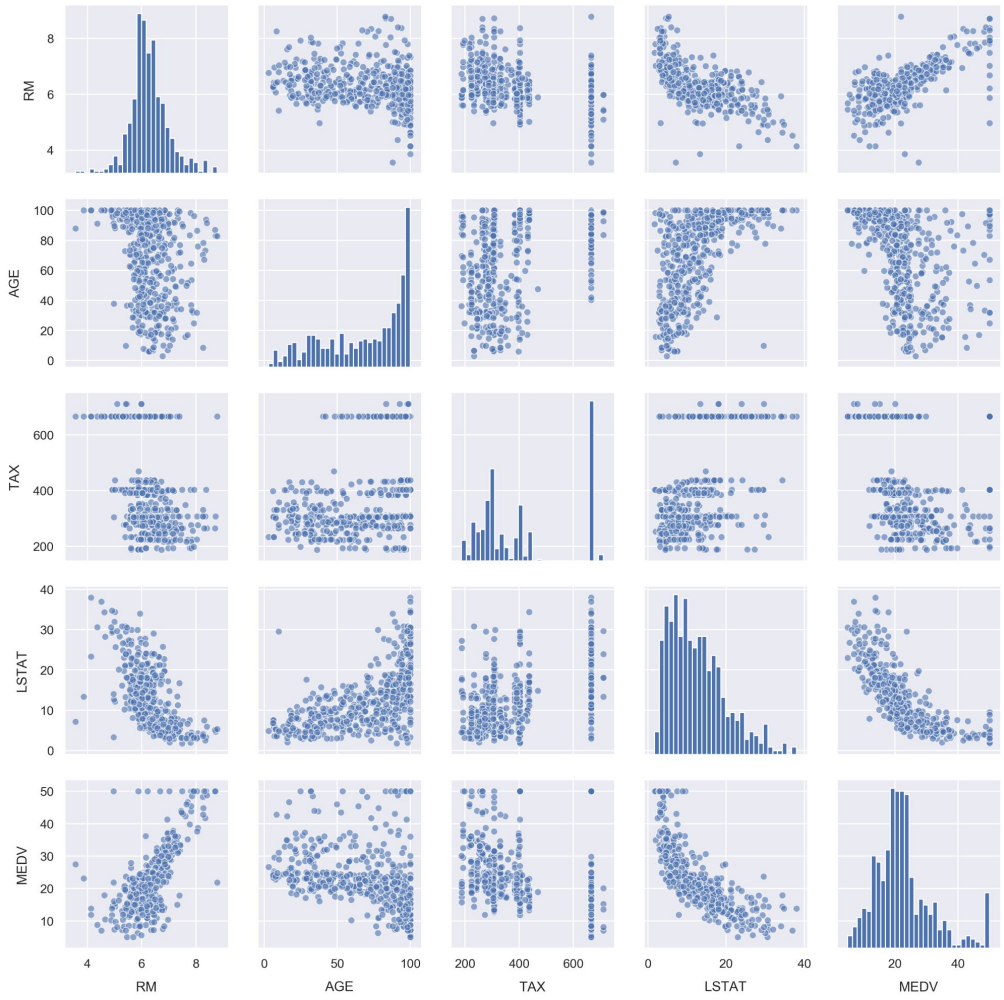
- Example 1: Email Spam
  - 4601 email messages were stored and labelled as spam or not.
  - The relative frequency of the 57 most common words are available.
  - Aim is to design automatic spam detector that could filter out spam before clogging the users mailboxes.

TABLE 1.1. *Average percentage of words or characters in an email message equal to the indicated word or character. We have chosen the words and characters showing the largest difference between* `spam` *and* `email`.

	george	you	your	hp	free	hpl	!	our	re	edu	remove
spam	0.00	2.26	1.38	0.02	0.52	0.01	0.51	0.51	0.13	0.01	0.28
email	1.27	1.27	0.44	0.90	0.07	0.43	0.11	0.18	0.42	0.29	0.01

[*Hastie et al. 2001 © Springer*]

- Example 2: Real Estate
  - Determine neighbourhood characteristics that drive house prices.
  - Matrix scatter plot of the Boston Housing dataset variables.



[*scikit learn Python library*]

- Example 3: Handwritten Digits
    - Construct a machine that identifies the numbers in a handwritten ZIP code from digitised images.

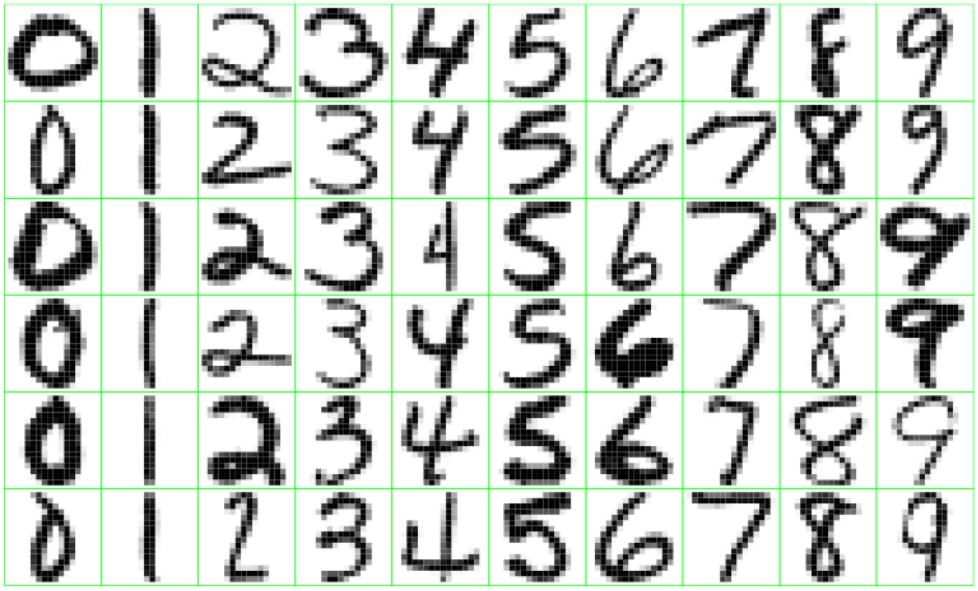


FIGURE 1.2. *Examples of handwritten digits from U.S. postal envelopes.*

[*Hastie et al. 2001 © Springer*]

Programming and Data Science

- Tasks to undertake for data science
    - Data collection
    - Data processing (wrangling)
    - Data visualisation
    - Train and apply algorithms from fields such as machine learning, statistics, data mining, optimisation, image processing, etc.

- ***Programming***
  - The process of producing an executable computer program that performs a specific task.
  - The purpose is to find a sequence of instructions that automate the implementation of the task for solving a given problem.

- ***Programming Language***
  - The source code of a program is written in one or more languages that are intelligible to humans, rather than machine code, which is directly executed by the CPU.
  - R 
    - https://www.r-project.org/
  - Python 
    - https://www.python.org/

- ***Open Source Software***
  - Free computer software which the user can modify and distribute within the terms of a licence.
  - Collaborative development has created diverse and very powerful software ecosystems.
  - Modular structure permits users to build an environment exactly suited to their needs.

2. Source-Code Editors and IDEs for R and Python

Source-code Editors

- Visual Studio Code
  - https://code.visualstudio.com/
- Notepad++
  - Windows only
  - https://notepad-plus-plus.org/
- Vim
  - https://www.vim.org/
- Sublime Text
  - Not open source
  - https://www.sublimetext.com/
- Atom
  - https://atom.io/
- Jupyter
  - https://jupyter.org/
- Emacs
  - https://www.gnu.org/software/emacs/
- TextMate
  - Macs only
  - https://macromates.com/

Integrated Development Environments (IDEs)

- Spyder
  - https://www.spyder-ide.org/
- RStudio

  - https://rstudio.com
- Eclipse
  - https://www.eclipse.org/
- Microsoft Visual Studio
  - https://visualstudio.microsoft.com/vs/

Version Control

- Git 
  - https://git-scm.com/
- GitHub 
  - https://github.com/

Markdown / Markup Languages

- HTML
- XML
- LaTeX

3. Installing and Interacting with R 

Installing R

- Go to *Comprehensive R Archive Network* (CRAN)
  - https://cran.r-project.org/
- Click at *0-Cloud*
  - https://cloud.r-project.org/
- Click at *Download R for macOS* or *Download R for Windows*
  - https://cloud.r-project.org/bin/macosx/
    - Download *R-XXX.pkg* and install
  - https://cloud.r-project.org/bin/windows/
    - Download *base* and install

Using R: Interface

- *Terminal* on macOS or *Command Prompt* on Windows
  - Type "*R*" into the prompt, hit enter, to open R.
  - For Windows, path to the *R bin* need to be added.

- Default *R Console* or *R GUI*
  - Double-click at *R* icon

- RStudio Desktop IDE
  - Script file with *.r* suffix

- R Prompt
  - ">"
  - Scripting language run commands as soon as it is entered.

Using R: Environment

- Get working directory
  - getwd()
- Set working directory
  - setwd("/path/to/directory")
- Installing packages
  - install.packages("packagename")
- Getting help
  - help(function) or ?function
  - help.start()
    - open the html version of R's online documentation

R Operations (prelude)

- Assignment Operator
  - "<-"
  - Example:
    - a <- 67890/12345

      # compute the ratio, store the result in ram, assign to a

      # the value of a is 5.499392
    - b <- a

      # binding the value bound to a to b
- Output
  - "print()"
  - Example:
    - print('Hello World!') *# print the string literals*

- print(a, 5)

  # print the value bound to a in 5 significant digits

  # the output is 5.4994

4. R Markdown and R Notebooks 

Markdown

- ***Markdown*** is a markup language that consists of a set of rules for adding formatting elements to plain text documents
    - Boldface, italics, headers, paragraphs, lists, code blocks, images, etc.
    - https://www.markdownguide.org/

- Invented by *John Gruber*
    - The overriding design goal for Markdown's formatting syntax is to make it as readable as possible.
    - The idea is that a Markdown-formatted document should be publishable as-is, as plain text, without looking like it's been marked up with tags or formatting instructions

- Pandoc's Markdown
    - Allows effortless conversion between a wide range of markup formats to others and to various file types

R Markdown

- ***R Markdown*** combines the strengths of Pandoc's Markdown and R to produce an authoring framework for data science that allows the combination of text, code and results in a single document, which can in turn, be converted to a wide range of formats using Pandoc.
    - https://rmarkdown.rstudio.com/

- R Markdown file has the ".rmd" extension, hosts 3 types of content
    - YAML metadata
        - Yet Another Markup Language
        - Where some configuration options are provided to guide the R Markdown build process
        - Enclosed between "---"
    - Text
        - The narrative part of the R Markdown file, written in Markdown syntax
    - Code chunks
        - Where R code is placed and options are provided to determine what happens with the code and its outputs
        - Start with "```{r}" and end with "```"

- R Markdown Package
    - install.packages("rmarkdown")
        - *# install the rmarkdown package*
    - library("rmarkdown")
    - render("/path/to/hello.rmd")
        - *# save the contents of R Markdown file*

- R Markdown Sample

```
---
title: "Hello Rmd"
author: "[Luke Skywalker](https://en.wikipedia.org/wiki/Luke_Skywalker)"
date: 21 December 2112
---
```

```
## My First R Markdown File

After a hard training day with Yoda, I decided to author my first [R
Markdown](https://rmarkdown.rstudio.com) file. This is a text chunk
written in *Markdown syntax*. I can write **bold** and *italics*, and
even record quotes I want to remember like

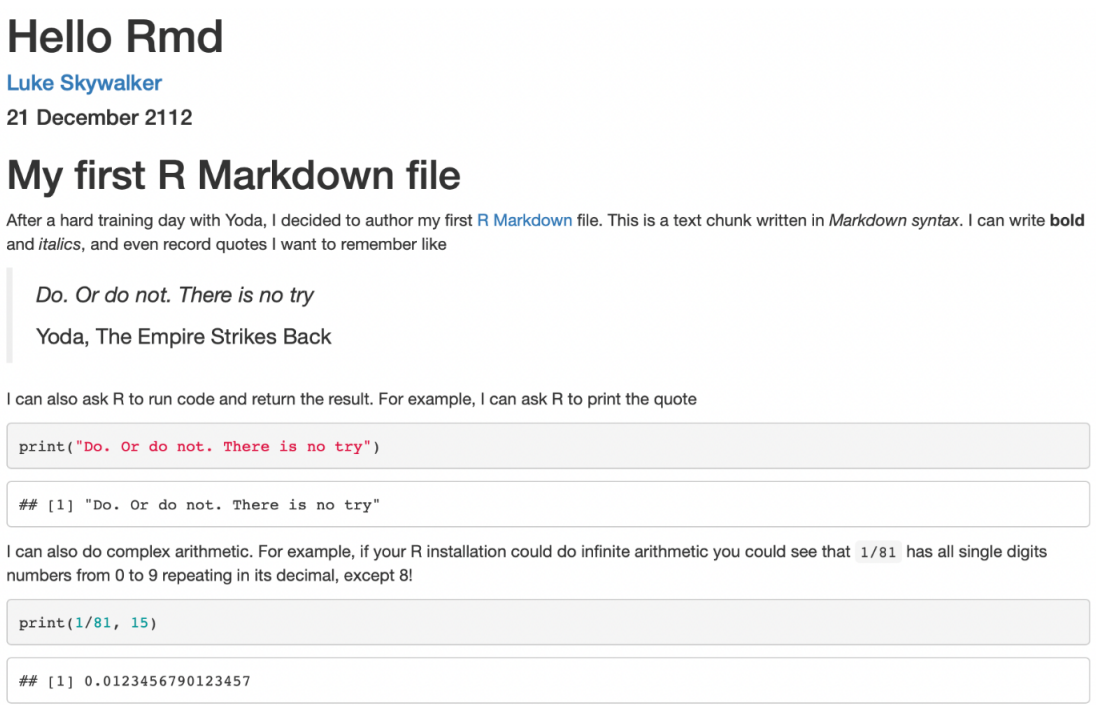
> *Do. Or do not. There is no try*
>
> Yoda, The Empire Strikes Back

I can also ask R to run code and return the results. For example, I can
ask R to print the quote

```{r quote}
print("Do. Or do not. There is no try")
```

I can also do complex arithmetic. For example, if your R installation
could do infinite arithmetic you could see that `1/81` has all single
digits numbers from 0 to 9 repeating in its decimal, except 8!

```{r arithmetic}
print(1/81, 15)
```
```



Hello Rmd

Luke Skywalker

21 December 2112

My first R Markdown file

After a hard training day with Yoda, I decided to author my first R Markdown file. This is a text chunk written in *Markdown syntax*. I can write **bold** and *italics*, and even record quotes I want to remember like

> *Do. Or do not. There is no try*
>
> Yoda, The Empire Strikes Back

I can also ask R to run code and return the result. For example, I can ask R to print the quote

```
print("Do. Or do not. There is no try")
```

```
## [1] "Do. Or do not. There is no try"
```

I can also do complex arithmetic. For example, if your R installation could do infinite arithmetic you could see that `1/81` has all single digits numbers from 0 to 9 repeating in its decimal, except 8!

```
print(1/81, 15)
```

```
## [1] 0.0123456790123457
```

R Notebooks

- R Notebook mode is a special way to working with R Markdown files.
  - Code chunks can be executed independently and interactively.
  - Chunk output shown immediately under the code chunks.

````
---
title: "Hello Rmd"
author: '[Luke Skywalker](https://en.wikipedia.org/wiki/Luke_Skywalker)'
date: "21 December 2112"
---

# My first R Markdown file

After a hard training day with Yoda, I decided to author my first [R
Markdown](https://rmarkdown.rstudio.com) file. This is a text chunk
written in *Markdown syntax*. I can write **bold** and *italics*, and
even record quotes I want to remember like

> *Do. Or do not. There is no try*
>
> Yoda, The Empire Strikes Back

I can also ask R to run code and return the result. For example, I can
ask R to print the quote

```{r quote}
print("Do. Or do not. There is no try")
```

I can also do complex arithmetic. For example, if your R installation
could do infinite arithmetic you could see that `1/81` has all single
digits numbers from 0 to 9 repeating in its decimal, except 8!

```{r arithmetic}
print(1/81, 15)
```
````

5. Installing and Working with Python

Installing Python

- Go to *Anaconda*, download *Anaconda Individual Edition*
  - https://www.anaconda.com/products/distribution
- Packages include
  - *conda*
    - package management system
  - *pandas, scikit-learn, nltk*
    - packages for data science
  - *Anaconda Navigator*
    - a graphical user interface
  - *QtConsole*
    - an interactive Python environment
  - *Spyder*
    - a standard cross-platform IDE for Python
    - https://www.spyder-ide.org/
  - *Jupyter Notebook*
    - an interactive web-browser based application for creating and sharing code
    - https://jupyter.org/

QtConsole

- Launch "QtConsole" from "Anaconda Navigator"

- Python Prompt
  - "In [x]:"

    - Scripting language run command as soon as it's entered
    - Common commands:
        - "ls": list the contents of a directory
        - "cd xname": change to a directory named xname
        - "cd ..": go up a directory
        - "run example.py": run the program example.py
- Python Scripts
    - Script file with ".py" suffix

Spyder

- Launch "Spyder" from "Anaconda Navigator"

- IDE for scientific computing
    - Editor: write code
    - Object Inspector: access variables, plots and files
    - Console: evaluate code and view the results
        - runfile('example.py')

- Create project
    - Select "Projects" → "New Project"
    - Specify the project name and working directory
    - Click green "play" button to execute the program

Jupyter Notebooks

- Launch "Jupyter Notebook" from "Anaconda Navigator"

- "Julia" + "Python" + "R"
- Integrate code and output into a single document contains:

  - Live code, mathematical equations, visualisations, and explanatory/narrative text
- Can be easily shared
  - Notebook files have ".ipynb" extension

- Create new notebook
  - "File" → "New Notebook" → "Python 3"
- Exporting notebook
  - "File" → "Download as" → "HTML (.html)"
  - "File" → "Print Preview" (for PDF)
- Shutting Down Jupyter
  - "File" → "Close and Halt"
  - Quit

Python Operations (prelude)

- Assignment Operator
  - "="
  - Example:
    - a = 67890/12345
      # compute the ratio, store the result in ram, assign to a
      # the value of a is 5.499392
    - b = a
      # b pointing to value of a
- Output
  - "print()"
  - Example:
    - print('Hello World!') *# print the string literals*
    - print(a) *# print the value of a*

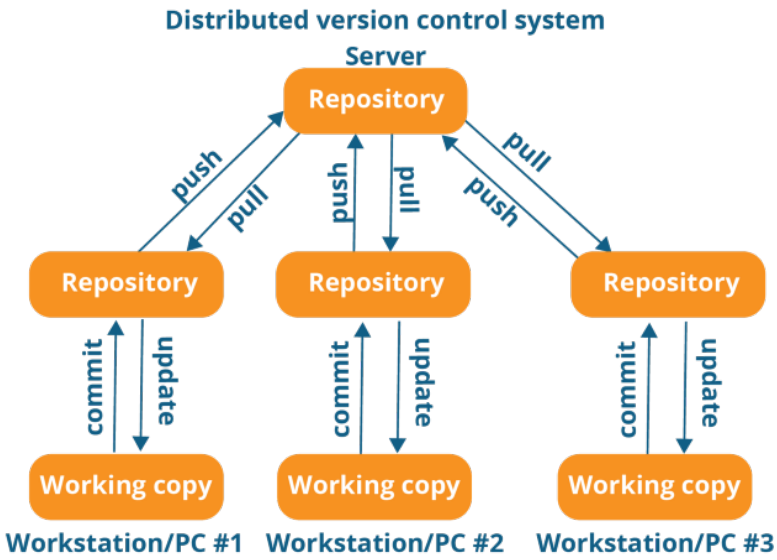
6. Version Control Systems

- ***Version Control*** is a class of systems responsible for managing changes to computer programs, documents, large websites, or other collections of information.

- ***Version Control Systems*** (VCS) are software tools that help software teams manage changes to source code over time.
    - Undertakes the tedious task of keeping track of the changes to all project's files and who made them
    - Allows users to recover any previous version at any given time
    - Examples:
        - **Subversion** (used to develop R)
        - **Git** (used to develop RStudio)

Git 

- ***Git*** is a distributed version control system (DVCS).
- Originally created by *Linus Torvalds* in 2005 for Linux kernel development.
- Distributed version control system (DVCS)
    - All project files and their histories are present both remotely and in the computers of all developers contributing to the project.
    - Developers can work offline and asynchronously without a constant connection to a central repository.



[Source: *https://medium.com/@sahoosunilkumar/how-does-git-works-5cc8444ea383*]

Installing Git

- Go to "Git", click at "Download for Mac" or "Download for Windows"
  - https://git-scm.com/

More about Git

- ***Git repository*** contains the collection of the files and directories, as well as the history of changes made to those files.
  - A local directory holding a project's files and folders that is not under version control is turned into a Git repository
  - A remote Git repository is cloned into your computer from elsewhere
- ***Git commit*** captures a snapshot or milestone along the timeline of a Git project. Commits are created to capture the project's currently staged changes
- ***Git branch*** is a pointer to a snapshot of changes. Git stores a branch as a reference to a commit, instead of copying files from directory to directory.

Basic Git Commands

- Setting up Git Credentials
  - git --version
    - *# show the git version*
  - git config --global user.name "HANDSOME"
  - git config --global user.email "HAND@SOME.KOH"

- Setting up Git Repository
  - cd ~/directory/gitKOH/
    - *# change to gitKOH directory*
  - git init
    - *# initialise empty Git repository in gitKOH*
    - *# ~/directory/gitKOH/.git/ created*

- Staging
  - git add example.r
    - *# stage example.r for version control*

- Check repository current status
  - git status

- Commit changes
  - git commit –m "first commitment"

- Show complete log of all changes
  - git log

- Branching
    - git branch first-branch
        - # *create a new branch name first-branch*
    - git branch
        - # *checking the branch currently at*Staging

- Switch Branch
    - git checkout first-branch
        - # *switch to branch first-branch*

- Compare the state
    - git diff master first-branch
        - # *show differences between both branches*

- Merging
    - git checkout master
        - # *switch to master branch*
    - git merge first-branch
        - # *merge first-branch into master*

7. GitHub

- Remote Git repository hosting service
  - GitHub [https://github.com/]
  - GitLab [https://about.gitlab.com/]
  - BitBucket [https://bitbucket.org/product/]

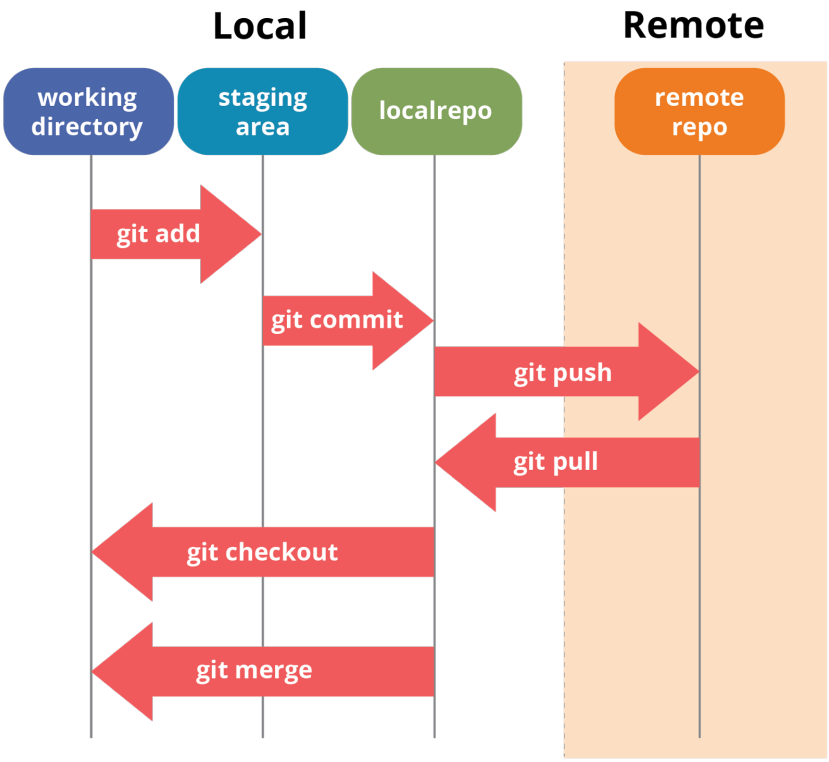
- Create GitHub account
  - Using same email credentials as local repository

- Upload local repository content to remote repository
  - git push ‘remote_name’ ‘branch_name’

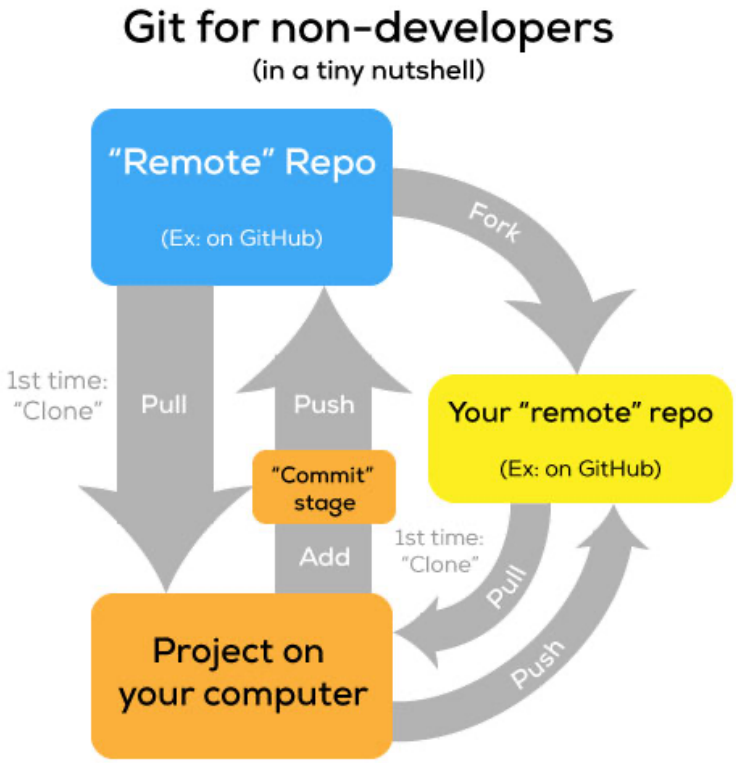
- Update local repository content from remote repository
  - git pull

- Clone a remote repository into your computer
  - git clone https://github.com/pandas-dev/pandas.git

Git Workflow



[Source: *https://medium.com/@sahoosunilkumar/how-does-git-works-5cc8444ea383*]



[Source: *http://anitacheng.com/git-for-non-developers*]

Useful Resources

- 50 years of Data Science (Article)
  - http://courses.csail.mit.edu/18.337/2015/docs/50YearsDataScience.pdf
- The Joy of Stats (Video)
  - https://www.gapminder.org/videos/the-joy-of-stats/
- Git for non-developers (Blog)
  - http://anitacheng.com/git-for-non-developers
- Programming language (Wikipedia)
  - https://en.wikipedia.org/wiki/Programming_language
- Source-code editor (Wikipedia)
  - https://en.wikipedia.org/wiki/Source-code_editor
- Integrated development environment (Wikipedia)
  - https://en.wikipedia.org/wiki/Integrated_development_environment
- R programming for beginners (Youtube)
  - https://youtube.com/playlist?list=PLtL57Fdbwb_Chn-dNR0qBjH3esKS2MXY3
- Install R and RStudio on Mac (Youtube)
  - https://www.youtube.com/watch?v=Y20P3u3c_1c
- Install R and RSudio on Windows (Youtube)
  - https://www.youtube.com/watch?v=GAGUDL-4aVw
- The Markdown Guide (Website)
  - https://www.markdownguide.org/
- R Markdown Quick Tour (Video)
  - https://rmarkdown.rstudio.com/authoring_quick_tour.html
- R Markdown Cheat Sheet

  - https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf
- R Markdown: The Definitive Guide (Book)
  - https://bookdown.org/yihui/rmarkdown/
- R for Data Science (Book)
  - https://r4ds.had.co.nz/index.html
- Jupyter Notebook: An Introduction (Website)
  - https://realpython.com/jupyter-notebook-introduction/#getting-up-and-running-with-jupyter-notebook
- How to Use Jupyter Notebook: A Beginner's Tutorial (Blog)
  - https://www.dataquest.io/blog/jupyter-notebook-tutorial/
- A Beginner's Tutorial to Jupyter Notebooks (Blog)
  - https://towardsdatascience.com/a-beginners-tutorial-to-jupyter-notebooks-1b2f8705888a
- Happy Git and GitHub for the user (Book)
  - https://happygitwithr.com/
- Git and GitHub Handbook (Website)
  - https://docs.github.com/en/get-started/using-git/about-git
- Git/GitHub Guide (Website)
  - https://kbroman.org/github_tutorial/
- Learn Git Branching (Interactive)
  - https://learngitbranching.js.org/
- Git Cheat Sheets
  - https://training.github.com/