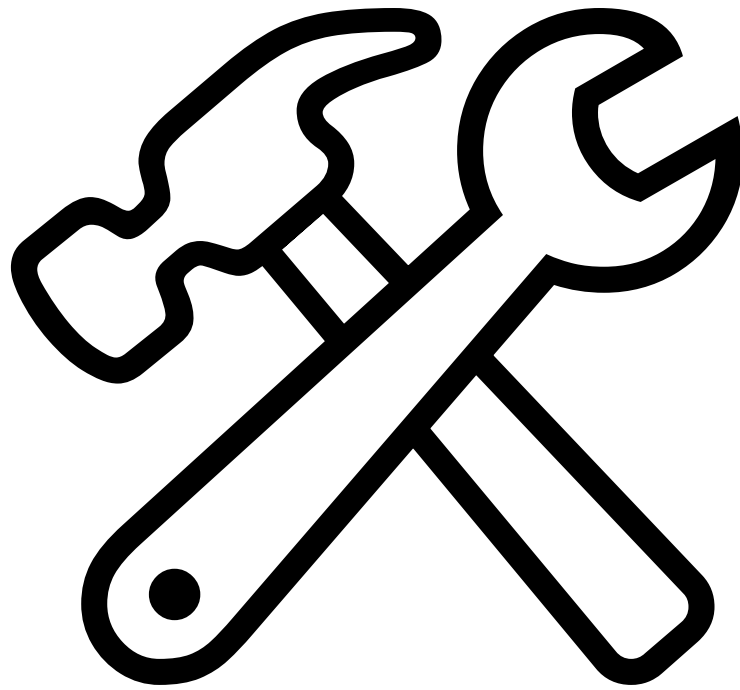


# Exercise 1 | Introduction & Logistics

Max Pellert

IS 616: Large Scale Data Analysis and Visualization

Let's get your basic setups  
running!



# R

## Install R

```
https://cran.r-project.org/
```

## Use RStudio

```
https://posit.co/products/open-source/rstudio/
```

## Install R packages

```
install.packages()
```



# R packages

```
data.table  
ggplot2  
tidyverse  
quanteda  
...
```

Usually, functions in R are well-documented, just run any function name prefixed with `?` to get help if you are stuck.

# Visualization and R



## ggplot2

### Overview

ggplot2 is a system for declaratively creating graphics, based on [The Grammar of Graphics](#). You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.

# Data Analysis in R

## Introduction to data.table

2023-02-16

This vignette introduces the `data.table` syntax, its general form, how to *subset rows*, *select and compute* on columns, and perform aggregations *by group*. Familiarity with `data.frame` data structure from base R is useful, but not essential to follow this vignette.

### Data analysis using `data.table`

Data manipulation operations such as *subset*, *group*, *update*, *join* etc., are all inherently related. Keeping these *related operations together* allows for:

- *concise* and *consistent* syntax irrespective of the set of operations you would like to perform to achieve your end goal.
- performing analysis *fluidly* without the cognitive burden of having to map each operation to a particular function from a potentially huge set of functions available before performing the analysis.
- *automatically* optimising operations internally, and very effectively, by knowing precisely the data required for each operation, leading to very fast and memory efficient code.

Briefly, if you are interested in reducing *programming* and *compute* time tremendously, then this package is for you. The philosophy that `data.table` adheres to makes this possible. Our goal is to illustrate it through this series of vignettes.

<https://cran.r-project.org/web/packages/data.table/vignettes/datatable-intro.html>

# Python

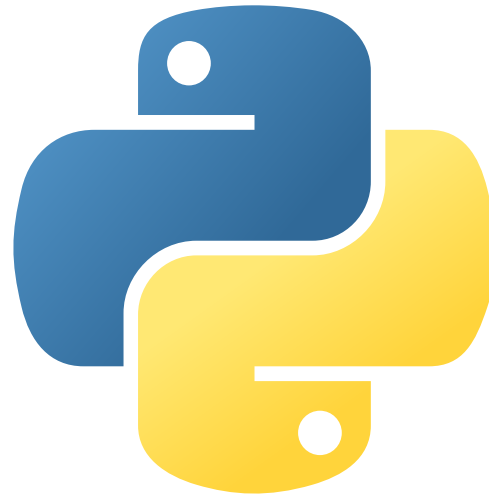
## Install Spyder

```
https://www.spyder-ide.org/
```

## Use pip as package manager

```
https://pip.pypa.io/en/stable/installation/
```

```
after installing pip, install packages with `pip install`
```



# Python packages

```
wordshiftgraphs  
  
matplotlib  
  
seaborn  
  
altair  
  
nltk  
  
spacy  
  
pytorch  
  
transformers  
  
...
```

Python Package Index (PyPI) (<https://pypi.org/>) usually also provides links to package documentations

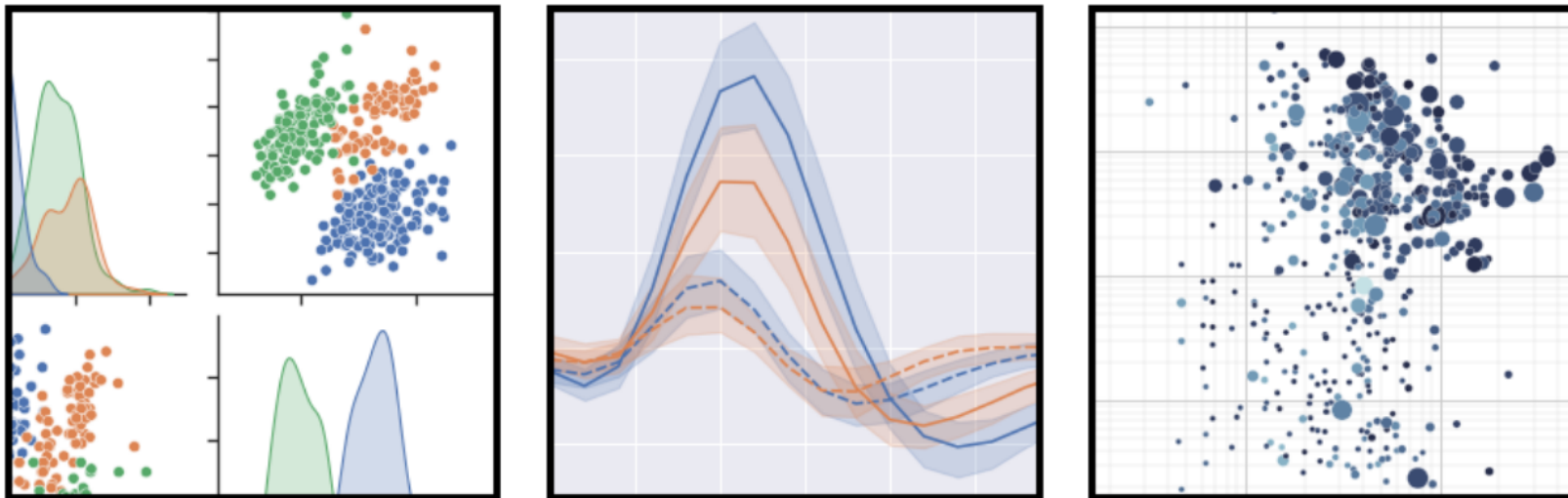


# Visualization and Python

## Matplotlib: Visualization with Python

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

## seaborn: statistical data visualization



# Visualization and Python

## Vega-Altair: Declarative Visualization in Python



**Vega-Altair** is a declarative visualization library for Python. Its simple, friendly and consistent API, built on top of the powerful **Vega-Lite** grammar, empowers you to spend less time writing code and more time exploring your data.

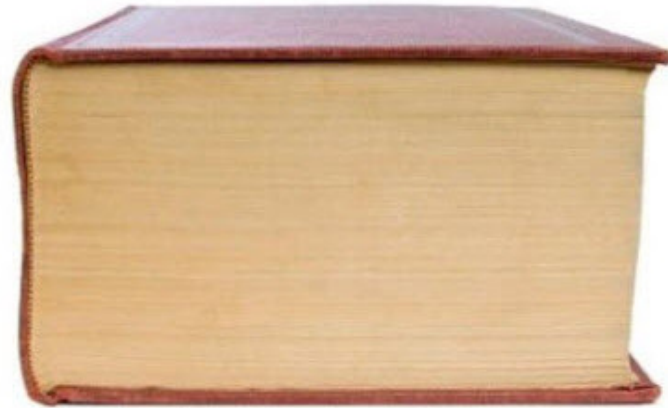
# Data Analysis in Python

## pandas

**pandas** is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

Install pandas now!

<https://pandas.pydata.org/>



A Data Analysis Workflow  
in Python



The same workflow in R

# R vs. Python?

In this course, you are free to use either

The popularity of one over the other is currently largely determined by disciplinary tastes and traditions (econ more towards R, computer science more towards Python) and this course has a interdisciplinary audience

They are both non-commercial and have dedicated communities

R may still have an edge in concise statistical computing and also visualization, but Python caught up a lot

Python is a general purpose language and the de-facto standard in deep learning

# To do

Catch up on using your favorite visualization package

Take special care to check out all ways to customize your plots, e.g.

- How to change the theme of a plot
- How to set custom axis limits
- How to set custom axis ticks and labels

...

You will need those skills later in the course

# To do

Also start refreshing your data wrangling skills

How to load data in

How to handle most common preprocessing steps

...

It is obvious that you need those skills to be able to do **data visualization**

# git

## git - the simple guide

just a simple guide for getting started with git. no deep shit ;)

 Tweet



download the  
cheat sheet  
now. it's free!

by Roger Dudler

credits to @tfnico, @fhd and Namics

this guide in deutsch, español, français, indonesian, italiano, nederlands, polski, portugûês, русский, türkçe,

မြန်မာ, 日本語, 中文, 한국어 Vietnamese

please report issues on [github](#)



want a simple  
but powerful  
git client for  
your mac?



<https://rogerdudler.github.io/git-guide/>





Search or jump to...



[Pull requests](#) [Issues](#) [Codespaces](#) [Marketplace](#) [Explore](#)




## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

Owner \*

Repository name \*

 maxpel ▾

/

Great repository names are short and memorable. Need inspiration? How about **cautious-barnacle**?

Description (optional)

 **Public**

Anyone on the internet can see this repository. You choose who can commit.

 **Private**

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

**Add a README file**

This is where you can write a long description for your project. [Learn more.](#)

**Add .gitignore**

Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None ▾

If you work alone on your repository, the following commands are usually all you need

```
git clone https://github.com/USERNAME/REPONAME.git  
  
git add .  
  
git commit -m "add first files"  
  
git push
```

Versioning tools are an excellent way to do backups of your code and to share it with other people systematically

If you work together with others, who also push to the same repo, you will need commands like

```
git pull
```

too, to make your local repo up-to-date before pushing to the remote one.

# LaTeX

To write your document in LaTeX, create a free account on [overleaf.com](https://www.overleaf.com)

You can start writing from a template, for example the one provided by Overleaf for submissions to [Nature Scientific Reports](#)

Overleaf also offers a [good introduction to LaTeX](#), if you have not used it before

# Regular Expressions

If you ever used a line like `find *.txt`, you performed pattern matching

Much more complex patterns are possible with regex

For a quick introduction:

<https://www.codemag.com/article/0305041/Getting-Started-With-Regular-Expressions>

Also used by many other useful basic tools like `awk`, `grep`, `sub`

Those are standalone command line tools but they are such classics that their functionality is also mimicked in other programming languages (for example in R there is `grep1`,

`gsub`, ...)

# Regular Expressions

You will often stumble over regex in many contexts (for example in this course)

There is an enormous amount of small variations (“flavours”) among them, for a comparison see for example:

<https://gist.github.com/CMCDragonkai/6c933f4a7d713ef712145c5eb94a1816>

This (and other factors) can often make writing regex a frustrating experience

Tools like ChatGPT can help by fixing dysfunctional regex or by explaining them to you!

# Learning by doing

## Teach Yourself Programming in Ten Years

Peter Norvig

### Why is everyone in such a rush?

Walk into any bookstore, and you'll see how to *Teach Yourself Java in 24 Hours* alongside endless variations offering to teach C, SQL, Ruby, Algorithms, and so on in a few days or hours. The Amazon advanced search for [[title: teach, yourself, hours, since: 2000](#)] and found 512 such books. Of the top ten, nine are programming books (the other is about bookkeeping). Similar results come from replacing "teach yourself" with "learn" or "hours" with "days."

The conclusion is that either people are in a big rush to learn about programming, or that programming is somehow fabulously easier to learn than anything else. Felleisen *et al.* give a nod to this trend in their book [How to Design Programs](#), when they say "Bad programming is easy. *Idiots* can learn it in *21 days*, even if they are *dummies*." The Abtruse Goose comic also had [their take](#).

Let's analyze what a title like [Teach Yourself C++ in 24 Hours](#) could mean:

<https://norvig.com/21-days.html>

# Questions?

