Session overview

1	Fri, Sep 6, 2024	**VSC, virtual environments**
2	Mon, Sep 9, 2024	**Version control**
3	Tue, Sep 10, 2024	**EDA & feature creation in python**
4	Thu, Sep 12, 2024	**EDA & feature creation in python (continued)**
5	Fri, Sep 13, 2024	**RStudio, importing and exploring data (EDA) in R**
6	Tue, Sep 17, 2024	**EDA live challenge**

Truths and lies trackers

Alejandro	Delgado Tello
Aleksandr	Smolin
Anastasiia	Chernavskaia
Angad Singh	Sahota
Blanca	Jimenez
Deepak	Malik
Denis	Shadrin
Enzo	Infantes
Ferran	Boada Bergadà
Hannes	Schiemann
Julián	Romero
Lucia	Sauer
Maria	Simakova Mariukha

Maria Jose	Aleman Hernandez
Marta	Sala
Matias	Borrell
Moritz	Peist
Nicolas	Rauth
Noemi	Lucchi
Pablo	Fernández
Simon	Vellin
Soledad	Monge
Tarang	Kadyan
Viktoria	Gagua
Wei	Sun

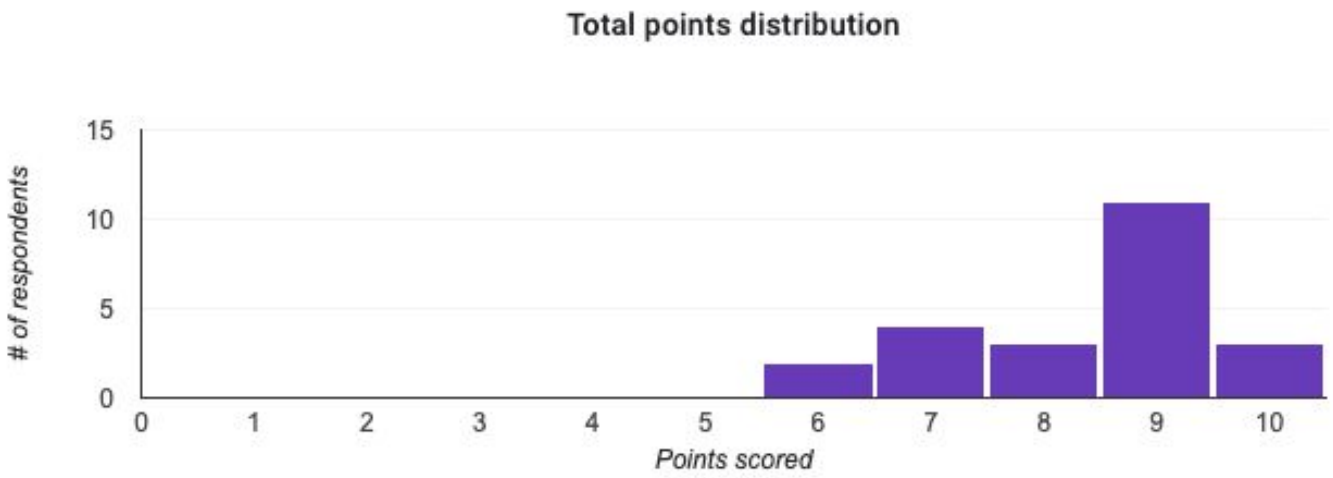
Sessions 5:
R and loop station

DSDM Brushup Course - Coding - September 2024

Margherita Philipp

Nicely done!

Average	Median	Range
8.39 / 10 points	9 / 10 points	6 - 10 points



Truths and lies interlude

Python

- "Do everything" generalist language - hybrid functional and object oriented language.
- More intuitive at the beginning.
- Better suited for writing production-grade code.

R

- Functional language, tightly optimized for data science work and statistics.
- Treats data as vectors and manipulates it with linear algebra like operations.
- Not great for low level programing.
  - It can be easy to do complicated things in R, but surprisingly difficult to do relatively simple things.
- Publication-level visuals and graphics.
- Great for time series data.
- Some highly specific stats packages
  - Mixed data sampling (MIDAS) regressions are now commonly used to deal with time series data sampled at different frequencies.

SYNTAX	Python	R
Importing Libraries	import numpy as np import pandas as pd	library(ggplot2) library (dplyr)
Import files	pd.read_csv('file.csv')	read_csv("file.csv")
Assignment	=	<- or ->
Data types	Numeric, Set, Dictionary, Boolean, Sequence Type	Numeric, Logical, Integer, Complex, Character , Raw
Data Structures	Dictionary, Set, Tuple, List	Vector, List, Dataframe, Matrice, Factor
Indexing	starts from [0]	starts from [1]
Arithmetic Operators	+, -, *, /, **	+, -, *, /, ^
Relational Operators	>, <, ==, <=, >=, !=	>, <, ==, <=, >=, !=
Logical Operators	AND, OR, NOT	&, |, &&, ||, !

R	VS.	PYTHON
OVERVIEW		
Open source statistical computing programming language		General purpose programming language
TYPE OF USER		
Knowledge users with a background in statistics		Diverse users ranging from beginners to software developers
SYNTAX		

`numbers <- c(1, 2, 3)`
`total <- sum(numbers)`
`print(total)`
		`numbers = [1, 2, 3]`
		`total = sum(numbers)`
		`print(total)`

LEARNING CURVE		
Steeper due to specific syntax and vectorized operations		Simple syntax and easy to learn for beginners
LIBRARIES		
dplyr, data.table, ggplot2, lm, glm, survival, caret, xgboost, randomForest		pandas, Matplotlib, seaborn, SciPy, scikit-learn, TensorFlow, PyTorch
VISUALIZATION		
High quality, publication level visuals and graphics		Diverse visualization and graphing options
INDUSTRY		
Scientific industries, healthcare, academia, social sciences		Data analytics, web development, machine learning

brushup - mph - RStudio

Go to file/function | Addins | brushup

EDA_equivalent_analysis.R | df_og

Source on Save | Run | Source

RStudio

Replace | Replace | All

In selection | Match case | Whole word | Regex | Wrap

```r
# Load necessary libraries
library(dplyr)
library(tidyr)
library(ggplot2)

##############################################################################
# 1. Importing data and explore the basics
##############################################################################

# Assuming the relative path to the data file is available, we'll load the CSV file.
df_og <- read.csv('./data/WB_more_data.csv')

# dimension of data set (shape)
dim(df_og)

# show structure
str(df_og)

```

20:1 | (Untitled) | R Script

Console | Terminal | Background Jobs

R 4.3.0 · ~/Documents/GitHub/brushup/

```
[1] 1085   13
> # show structure
> str(df_og)
'data.frame':   1085 obs. of  13 variables:
 $ Series.Name : chr  "Population, total" "Population, total" "Population, total" "Population, total" ...
 $ Series.Code : chr  "SP.POP.TOTL" "SP.POP.TOTL" "SP.POP.TOTL" "SP.POP.TOTL" ...
 $ Country.Name: chr  "Afghanistan" "Albania" "Algeria" "American Samoa" ...
 $ Country.Code: chr  "AFG" "ALB" "DZA" "ASM" ...
 $ X2001       : chr  "19688632" "3060173" "31200985" "58324" ...
 $ X2002       : chr  "21000256" "3051010" "31624696" "58177" ...
 $ X2003       : chr  "22645130" "3039616" "32055883" "57941" ...
 $ X2011       : chr  "29249157" "2905195" "36543541" "54310" ...
 $ X2012       : chr  "30466479" "2900401" "37260563" "53691" ...
 $ X2013       : chr  "31541209" "2895092" "38000626" "52995" ...
 $ X2021       : chr  "40099462" "2811666" "44177969" "45035" ...
 $ X2022       : chr  "41128771" "2777689" "44903225" "44273" ...
 $ X2023       : chr  "42239854" "2745972" "45606480" "43914" ...
> View(df_og)
```

Environment | History | Connections | Git | Tutorial

Import Dataset | 123 MiB | List

R | Global Environment

Data

df_og	1085 obs. of 13 variables

Files | Plots | Packages | Help | Viewer | Presentation

New Folder | New Blank File | Delete | Rename | More

Home > Documents > GitHub > brushup > data

Name	Size	Modified
..		
P_Data_Extract_From_World_Development_I...		
P_Data_Extract_From_World_Development_I...	1.2 MB	Sep 1, 2024, 10:55 AM
sample_data.csv	113 B	Sep 8, 2024, 11:12 PM
sample_data.txt	249 B	Sep 8, 2024, 11:12 PM
sample_data.xlsx	5.4 KB	Sep 8, 2024, 11:12 PM
text_data.csv	4.4 KB	Sep 10, 2024, 9:47 AM
WB_full.csv	2.5 MB	Sep 1, 2024, 8:55 AM
WB_metadata.csv	2.6 MB	Sep 1, 2024, 8:55 AM
WB_more_data.csv	151.2 KB	Sep 11, 2024, 6:33 PM
WB_pop_clean.csv	25.5 KB	Sep 10, 2024, 9:37 AM
WB_reshaped_nomissing.csv	69.5 KB	Sep 11, 2024, 11:08 PM

20m Task: R

1. R has slightly changed the column names.
   - How have they changed?
   - Why might that be?
2. Assigning missing values
   - How does R assign?
   - Do you remember/ can you check how “NA” is assigned in python? (hint: you need to specify a package)
3. What are the equivalents in R to these python commands:
   - to select a column: df[‘column_name]
   - to select a subset where column name is a certain value: df.loc[df[‘column_name] == ‘value’]
4. Why is there a comma before the closing bracket?
   - pop_data <- df_og[df_og$Series.Name == "Population, total",]
5. What does x <- c(‘item1’, ‘item2’, ‘item3’) do?
   - What is the equivalent in python?
6. Look at the reshaping (melting and pivoting) part of the code
   - R works with the pipe operator: it takes the output of the expression on its left and passes it as the first argument to the function on its right.
   - In the terminal, type: ??pivot_wider - what does that do?
7. What do gsub and rename do?
8. What are the python equivalents of
   - df[df$Year %in% c(2003, 2013, 2023),]
   - df %>% na.omit()
   - df <- df %>%
   - mutate(
   - ChildrenOutOfSchoolPrimary = as.numeric(ChildrenOutOfSchoolPrimary))

Truths and lies interlude

Google Colab

Collab demo.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text

```
[1] import pandas as pd
```

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

+ Code + Text

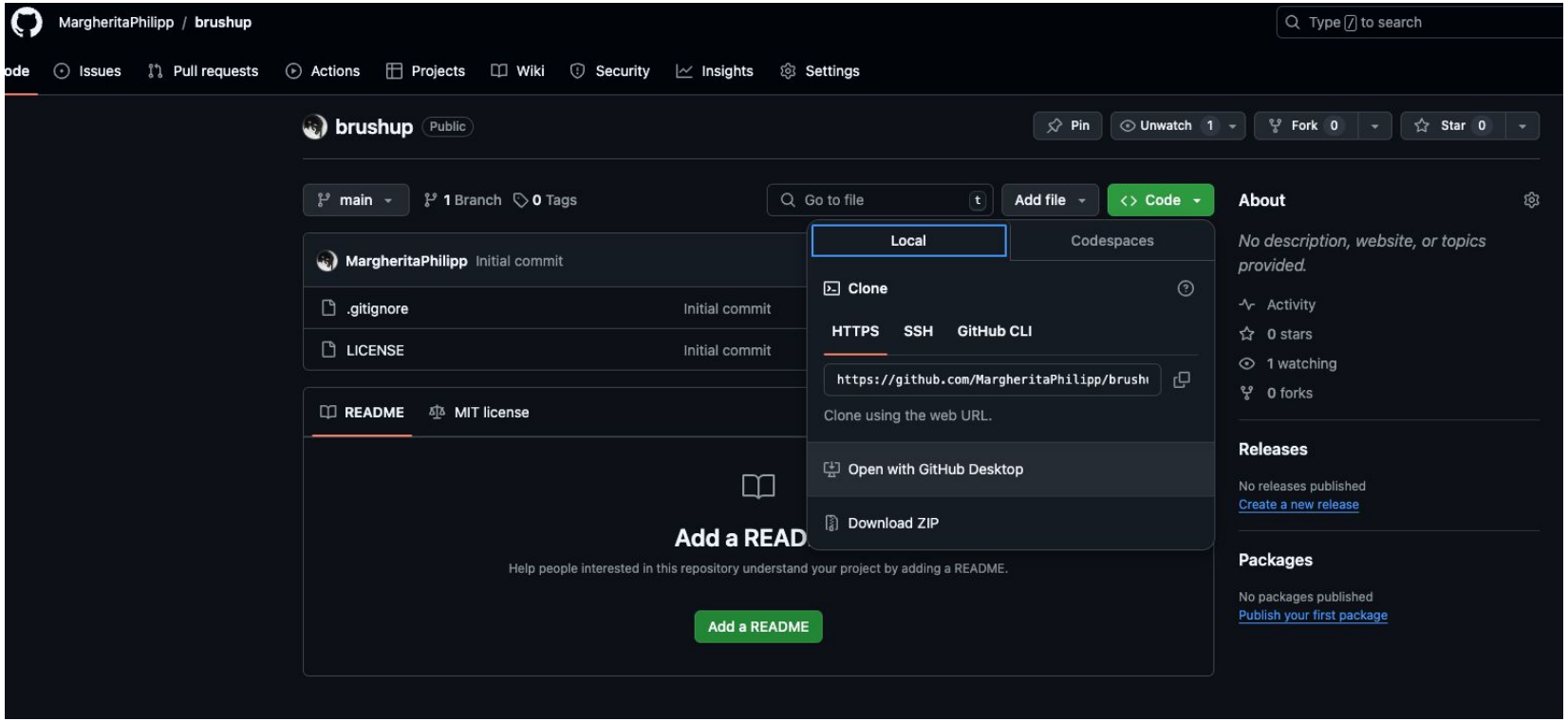
```
[3] df = pd.read_csv('/content/drive/MyDrive/EconAI_private/Teaching/Brushup_DSDM/data/WB_pop_clean.csv')

    #df = pd.read_csv('/content/drive/MyDrive/WB_pop_clean.csv')
```

```
df.head()
```

	Series Name	Series Code	Country Name	Country Code	2001	2002	2003	2011	2012	2013	2021	2022	2023
0	Population, total	SP.POP.TOTL	Afghanistan	AFG	19688632	21000256	22645130	29249157	30466479	31541209	40099462	41128771	42239854
1	Population, total	SP.POP.TOTL	Albania	ALB	3060173	3051010	3039616	2905195	2900401	2895092	2811666	2777689	2745972
2	Population, total	SP.POP.TOTL	Algeria	DZA	31200985	31624696	32055883	36543541	37260563	38000626	44177969	44903225	45606480
3	Population, total	SP.POP.TOTL	American Samoa	ASM	58324	58177	57941	54310	53691	52995	45035	44273	43914
4	Population, total	SP.POP.TOTL	Andorra	AND	67820	70849	73907	70567	71013	71367	79034	79824	80088

Next steps: Generate code with df | View recommended plots | New interactive sheet

Using GitHub desktop



Loops

You can loop over anything that is an iterable:

An **iterable** is any Python object capable of returning its members one at a time, permitting it to be iterated over in a for-loop.

What are examples of iterables?

Loops

You can loop over anything that is an iterable:

An **iterable** is any Python object capable of returning its members one at a time, permitting it to be iterated over in a for-loop.

Examples: lists, tuples, strings, dictionaries, sets

20m Task: Loopdiloop

For each loop, create a new code cell underneath to make your own altered version!

Briefly head down to modules and functions to see what you can find out about pandas within your notebook!

Done?

- Move on to file handling - which also uses loops!
- Move on to classes and do the exercise.
- Can you replicate the planets class from the youtube video? Or can you make your own?

For Monday

1. Push your new files to your github and merge your branch (submit screenshot on classroom)
2. Review the notebooks and R script
3. Think about what data you might want to use for the live coding challenge