Integrating R & Python into a Data Science program

Tiffany Timbers & Ian Flores Siaca

University of British Columbia

UBC Master of Data Science

FALL
SEP - DEC

Block 1 (4 weeks)

- 511 - Programming for Data Science
- 521 - Computing Platforms for Data Science
- 542 - Communication and Argumentation
- 551 - Descriptive Statistics and Probability for Data Science

Block 2 (4 weeks)

- 523 - Data Wrangling
- 531 - Data Visualization I
- 512 - Algorithms and Data Structures
- 552 - Statistical Inference and Computation I

Block 3 (4 weeks)

- 561 - Regression I
- 532 - Data Visualization II
- 571 - Supervised Learning I
- 513 - Databases and Data Retrieval

WINTER
JAN - APR

Block 4 (4 weeks)

- 562 - Regression II
- 573 - Feature and Model Selection
- 572 - Supervised Learning II
- 522 - Data Science Workflows

Block 5 (4 weeks)

- 563 - Unsupervised Learning
- 553 - Statistical Inference and Computation II
- 524 - Collaborative Software Development
- 574 - Spatial and Temporal Models

Block 6 (4 weeks)

- 575 - Advanced Machine Learning
- 541 - Privacy, Ethics and Security
- 554 - Experimentation and Causal Inference
- 525 - Web and Cloud Computing

SPRING
MAY - JUN

CAPSTONE PROJECT

(8 weeks)

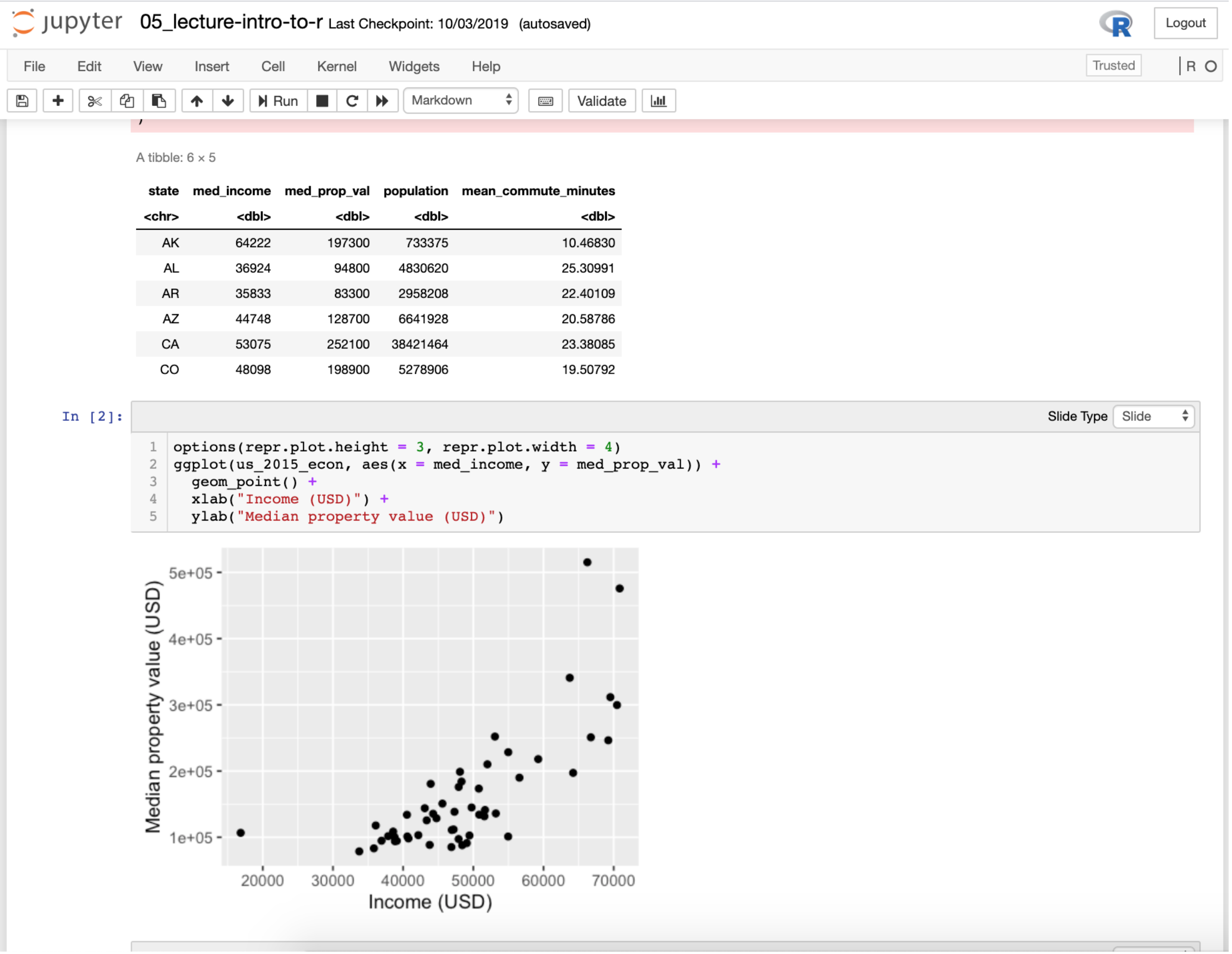
Languages used: R, Python, R & Python

UBC Master of Data Science program

Tools we teach for R & Python harmony

- RStudio
- Jupyter
- knitr & R Markdown
- feather file format
- reticulate
- Make
- Docker
- plotly Dash

Example 1: R in Jupyter!



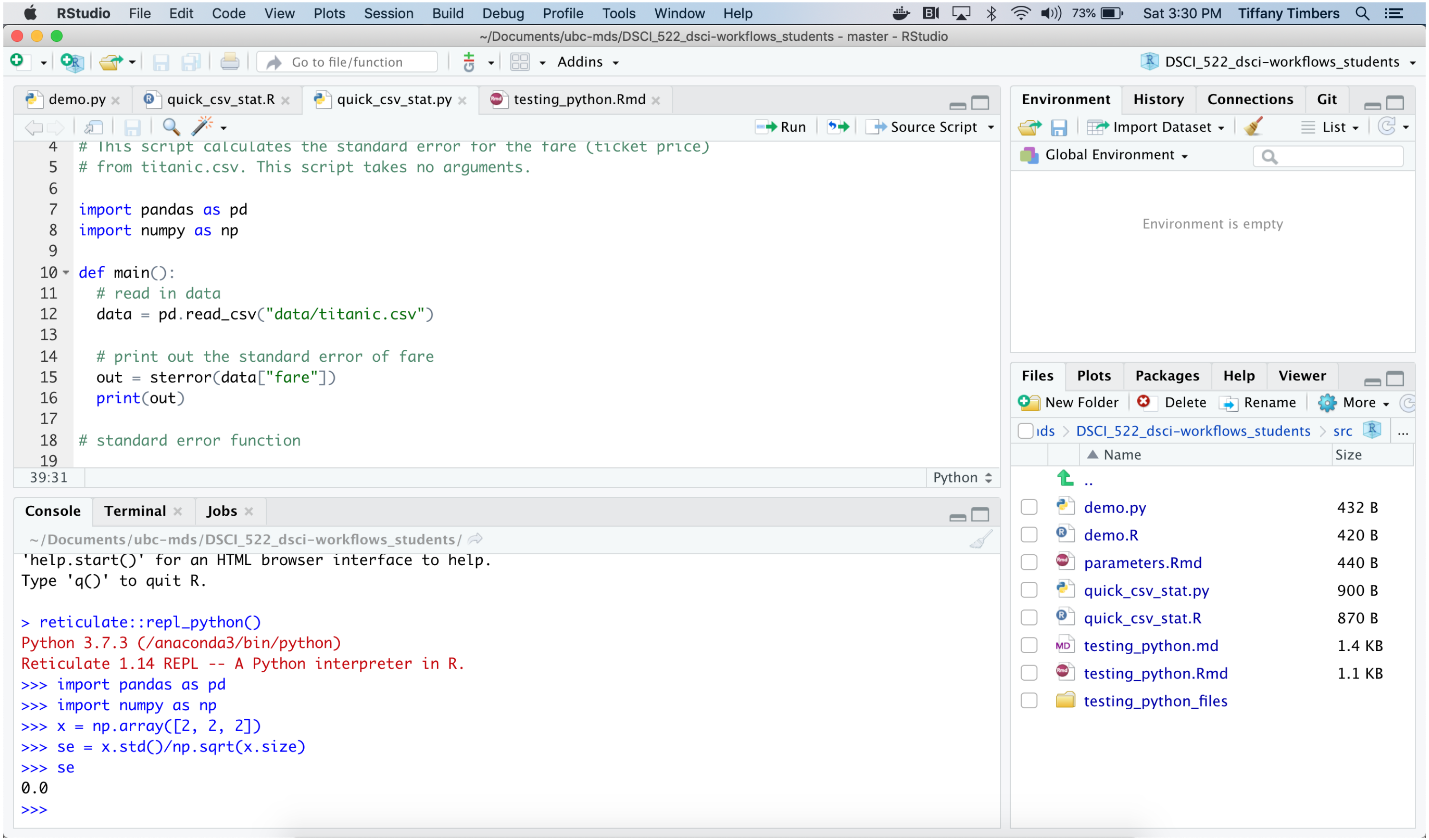
A tibble: 6 × 5

state	med_income	med_prop_val	population	mean_commute_minutes
<chr>	<dbl>	<dbl>	<dbl>	<dbl>
AK	64222	197300	733375	10.46830
AL	36924	94800	4830620	25.30991
AR	35833	83300	2958208	22.40109
AZ	44748	128700	6641928	20.58786
CA	53075	252100	38421464	23.38085
CO	48098	198900	5278906	19.50792

In [2]:

```r
options(repr.plot.height = 3, repr.plot.width = 4)
ggplot(us_2015_econ, aes(x = med_income, y = med_prop_val)) +
  geom_point() +
  xlab("Income (USD)") +
  ylab("Median property value (USD)")
```

Example 2: RStudio as a Python IDE!

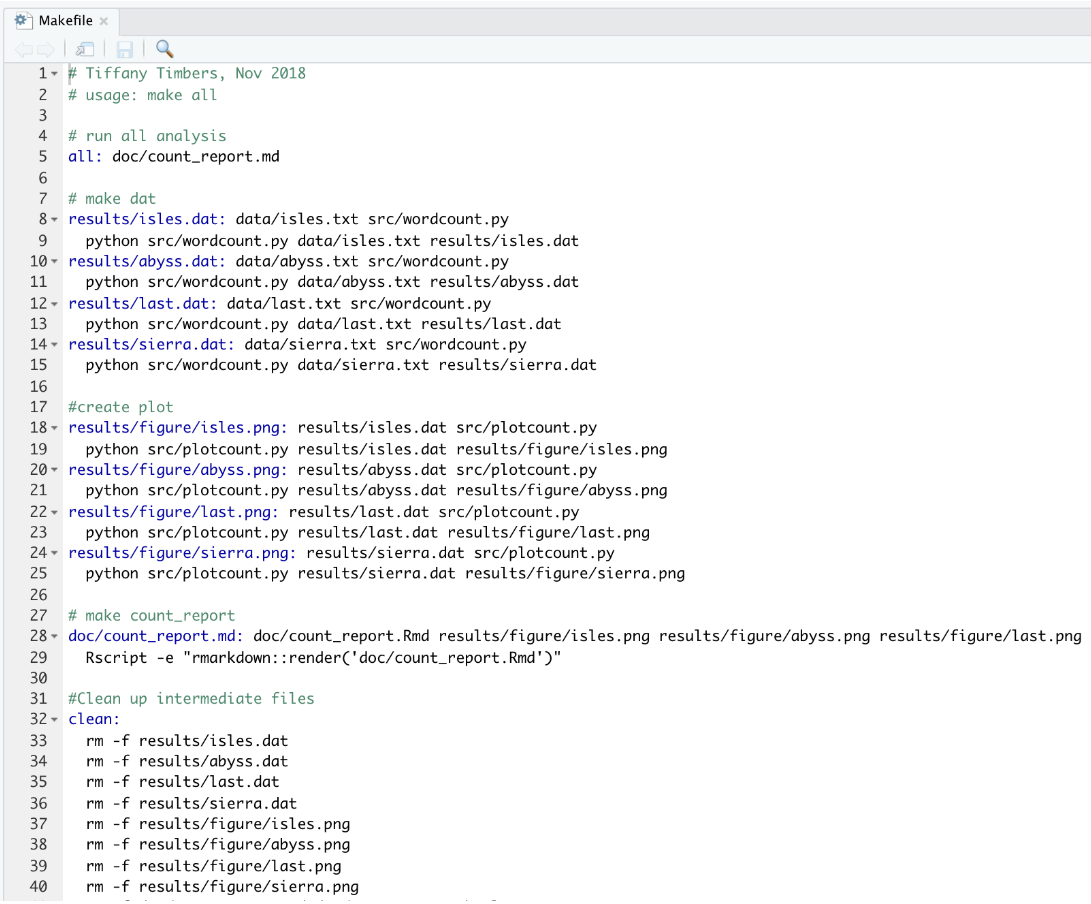


```
# This script calculates the standard error for the fare (ticket price)
# from titanic.csv. This script takes no arguments.

import pandas as pd
import numpy as np

def main():
  # read in data
  data = pd.read_csv("data/titanic.csv")

  # print out the standard error of fare
  out = sterror(data["fare"])
  print(out)

# standard error function

```

```
~/Documents/ubc-mds/DSCI_522_dsci-workflows_students/
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> reticulate::repl_python()
Python 3.7.3 (/anaconda3/bin/python)
Reticulate 1.14 REPL -- A Python interpreter in R.
>>> import pandas as pd
>>> import numpy as np
>>> x = np.array([2, 2, 2])
>>> se = x.std()/np.sqrt(x.size)
>>> se
0.0
>>>
```

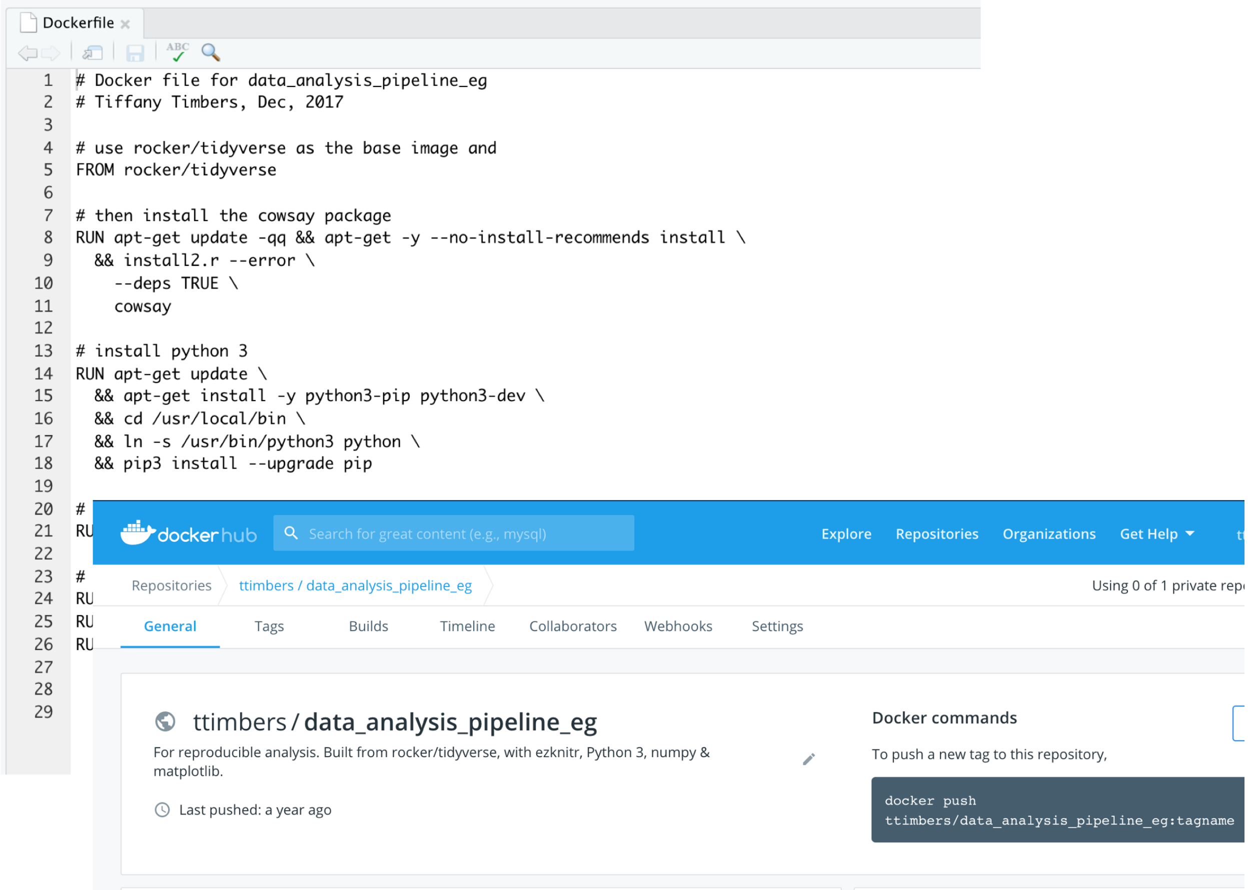
▲ Name	Size
..	
□ demo.py	432 B
□ demo.R	420 B
□ parameters.Rmd	440 B
□ quick_csv_stat.py	900 B
□ quick_csv_stat.R	870 B
□ testing_python.md	1.4 KB
□ testing_python.Rmd	1.1 KB
□ testing_python_files	

Example 3: GNU Make for polyglot automation!



```
# Tiffany Timbers, Nov 2018
# usage: make all

# run all analysis
all: doc/count_report.md

# make dat
results/isles.dat: data/isles.txt src/wordcount.py
  python src/wordcount.py data/isles.txt results/isles.dat
results/abyss.dat: data/abyss.txt src/wordcount.py
  python src/wordcount.py data/abyss.txt results/abyss.dat
results/last.dat: data/last.txt src/wordcount.py
  python src/wordcount.py data/last.txt results/last.dat
results/sierra.dat: data/sierra.txt src/wordcount.py
  python src/wordcount.py data/sierra.txt results/sierra.dat

#create plot
results/figure/isles.png: results/isles.dat src/plotcount.py
  python src/plotcount.py results/isles.dat results/figure/isles.png
results/figure/abyss.png: results/abyss.dat src/plotcount.py
  python src/plotcount.py results/abyss.dat results/figure/abyss.png
results/figure/last.png: results/last.dat src/plotcount.py
  python src/plotcount.py results/last.dat results/figure/last.png
results/figure/sierra.png: results/sierra.dat src/plotcount.py
  python src/plotcount.py results/sierra.dat results/figure/sierra.png

# make count_report
doc/count_report.md: doc/count_report.Rmd results/figure/isles.png results/figure/abyss.png results/figure/last.png
  Rscript -e "rmarkdown::render('doc/count_report.Rmd')"

#Clean up intermediate files
clean:
  rm -f results/isles.dat
  rm -f results/abyss.dat
  rm -f results/last.dat
  rm -f results/sierra.dat
  rm -f results/figure/isles.png
  rm -f results/figure/abyss.png
  rm -f results/figure/last.png
  rm -f results/figure/sierra.png
```

Example 4: Docker for polyglot reproducibility!



```
# Docker file for data_analysis_pipeline_eg
# Tiffany Timbers, Dec, 2017

# use rocker/tidyverse as the base image and
FROM rocker/tidyverse

# then install the cowsay package
RUN apt-get update -qq && apt-get -y --no-install-recommends install \
  && install2.r --error \
    --deps TRUE \
    cowsay

# install python 3
RUN apt-get update \
  && apt-get install -y python3-pip python3-dev \
  && cd /usr/local/bin \
  && ln -s /usr/bin/python3 python \
  && pip3 install --upgrade pip

#
RU

#
RU
RU
RU
```

ttimbers / **data_analysis_pipeline_eg**

For reproducible analysis. Built from rocker/tidyverse, with ezknitr, Python 3, numpy & matplotlib.

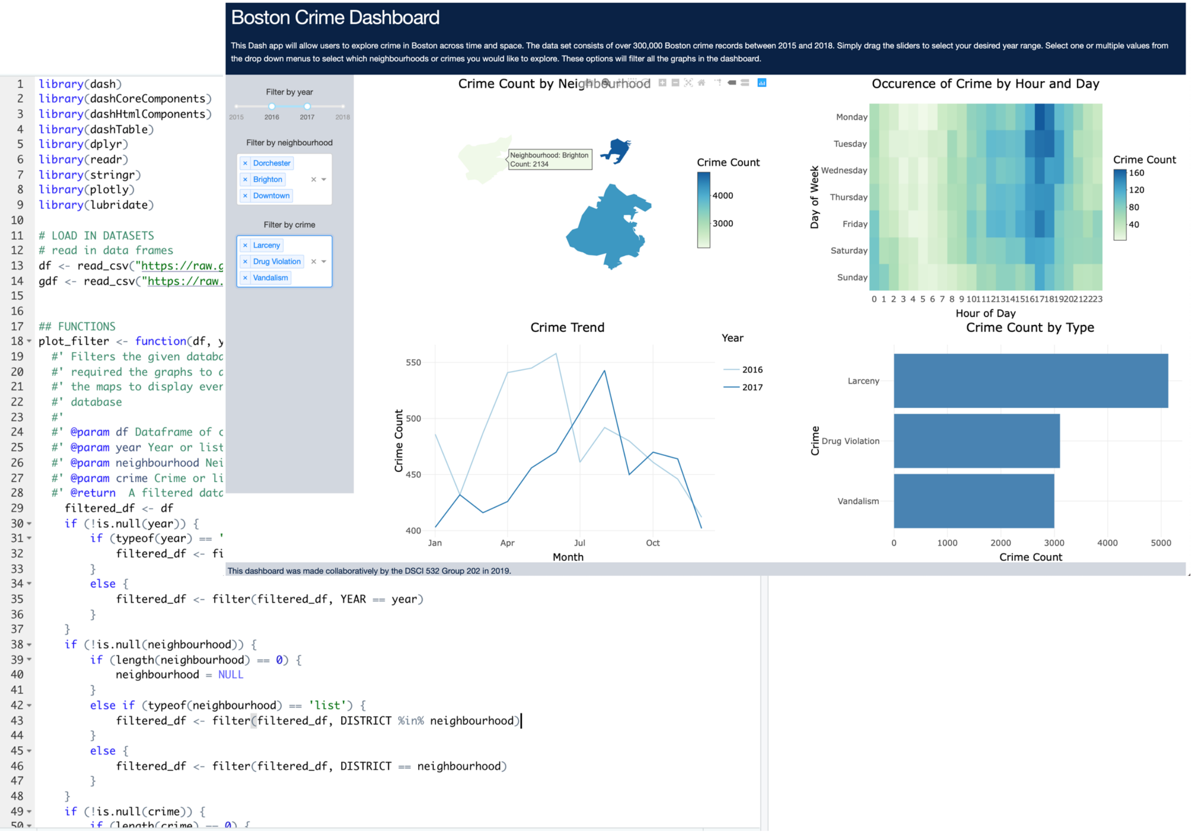
Last pushed: a year ago

Docker commands

To push a new tag to this repository,

```
docker push
ttimbers/data_analysis_pipeline_eg:tagname
```

Example 5: Dashboards in R with plotly Dash!



```r
library(dash)
library(dashCoreComponents)
library(dashHtmlComponents)
library(dashTable)
library(dplyr)
library(readr)
library(stringr)
library(plotly)
library(lubridate)

# LOAD IN DATASETS
# read in data frames
df <- read_csv("https://raw.g
gdf <- read_csv("https://raw.


## FUNCTIONS
plot_filter <- function(df, y
  #' Filters the given databc
  #' required the graphs to c
  #' the maps to display ever
  #' database
  #'
  #' @param df Dataframe of c
  #' @param year Year or list
  #' @param neighbourhood Nei
  #' @param crime Crime or li
  #' @return  A filtered datc
    filtered_df <- df
    if (!is.null(year)) {
        if (typeof(year) == '
            filtered_df <- fi
        }
        else {
            filtered_df <- filter(filtered_df, YEAR == year)
        }
    }
    if (!is.null(neighbourhood)) {
        if (length(neighbourhood) == 0) {
            neighbourhood = NULL
        }
        else if (typeof(neighbourhood) == 'list') {
            filtered_df <- filter(filtered_df, DISTRICT %in% neighbourhood)
        }
        else {
            filtered_df <- filter(filtered_df, DISTRICT == neighbourhood)
        }
    }
    if (!is.null(crime)) {
        if (length(crime) == 0) {
```

Boston Crime Dashboard

This Dash app will allow users to explore crime in Boston across time and space. The data set consists of over 300,000 Boston crime records between 2015 and 2018. Simply drag the sliders to select your desired year range. Select one or multiple values from the drop down menus to select which neighbourhoods or crimes you would like to explore. These options will filter all the graphs in the dashboard.

This dashboard was made collaboratively by the DSCI 532 Group 202 in 2019.

Pedagogical challenges (and solutions!) for teaching both R & Python

Problem 1: Mixed proficiencies of previous R & Python programming skills between students

Solution 1: Optional questions to challenge more advanced students, and extra practice questions with feedback to support novices.

Pedagogical challenges (and solutions!) for teaching both R & Python

Problem 2: Dual task interference

Solution 2: Learning outcomes in the program include comparing and contrasting the diffences between the languages (i.e., we spend a lot of time teaching and assessing whether the students know this).

Pedagogical challenges (and solutions!) for teaching both R & Python

Problem 3: Memory decay during breaks in practice

Solution 3: All blocks in the program have courses that require students to use R & Python

Take homes:

Tips for integrating R & Python into a Data Science program:

- Carefully choose tools that work well with both languages, and skip the ones that don't.
- Expect students to have a heterogeneous knowledge base that may differ between languages, and design exercises to address this.
- Teach the R'isms and the Python'isms and have the students compare and contrast them. Also, asses them on this!
- Structure the program so students repeatedly practice both languages, avoid gaps in one language if possible!

Thanks!

UBC MDS public resources: https://github.com/UBC-MDS/public