CDS Visual Analytics Assignment #4: Newspaper Face Detection Across Decades

What is it?

This assignment features 2 scripts, whose purpose is to process folders of images, scan them for faces, group them by decade, and produce a statistics `.csv` and a plot for each folder of images.
It features the main `face_detection.py` script, which processes each folder in the `/in` folder and scans each image inside for faces, generating statistics by decade, and saving them as `.csv` files.
This is followed by the `plot.py` script, which then generates line charts based on the generated `.csv` files.

Setup

1. Make sure to have python and Git Bash installed!
2. Open a Git Bash terminal and use Git to download the repository:

```
git clone https://github.com/missingusername/cds-vis-git.git
```

3. Navigate to the project folder for this assignment:

```
cd assignments/assignment4
```

4. Before doing anything else, you first need to get the data. You can download the `images.zip` manually from Zenodo, or simply by clicking here. When downloaded, unzip the folder and place the `GDL` , `IMP` & `JDG` folder inside the `in` folder of the `assignment4` directoy.
   This should leave you with a file structure like this:

```
assignment4
│
├──in
│   ├──GDL
│   ├──IMP
│   └──JDG
│
├──out
│   ├──GDL plot.png
│   ├──GDL stats.csv
│   ├──IMP plot.png
│   ├──IMP stats.csv
│   ├──JDG plot.png
│   └──JDG stats.csv
│
├──src
│   ├──face_detection.py
│   └──plot.py
│
├──README.md
├──requirements.txt
├──unix_run.sh
├──unix_setup.sh
├──win_run.sh
└──win_setup.sh
```

5. Before you can run the scripts, make sure you have the required libraries in the `requirements.txt` . This can be done by simply running the OS-appropriate setup script from inside the `assignment4` folder, which will set up a virtual environment and get the required libraries. Again, using Git Bash:

For Unix:

```
bash unix_setup.sh
```

For Windows:

```
bash win_setup.sh
```

6. The scripts feature an optional command line argument:

name	flag	description	is required
scale	-s	Scale images before running face detection. Can grant greatly improve computing speeds at the cost of accuracy. Takes floats as input. For example, `-s 0.5` would scale all images to 50% of their original size.	OPTIONAL

7. To finally execute the scripts (`face_detection.py` & `plot.py`), simply run the OS-appropriate `run.sh` script in the same Git Bash terminal:

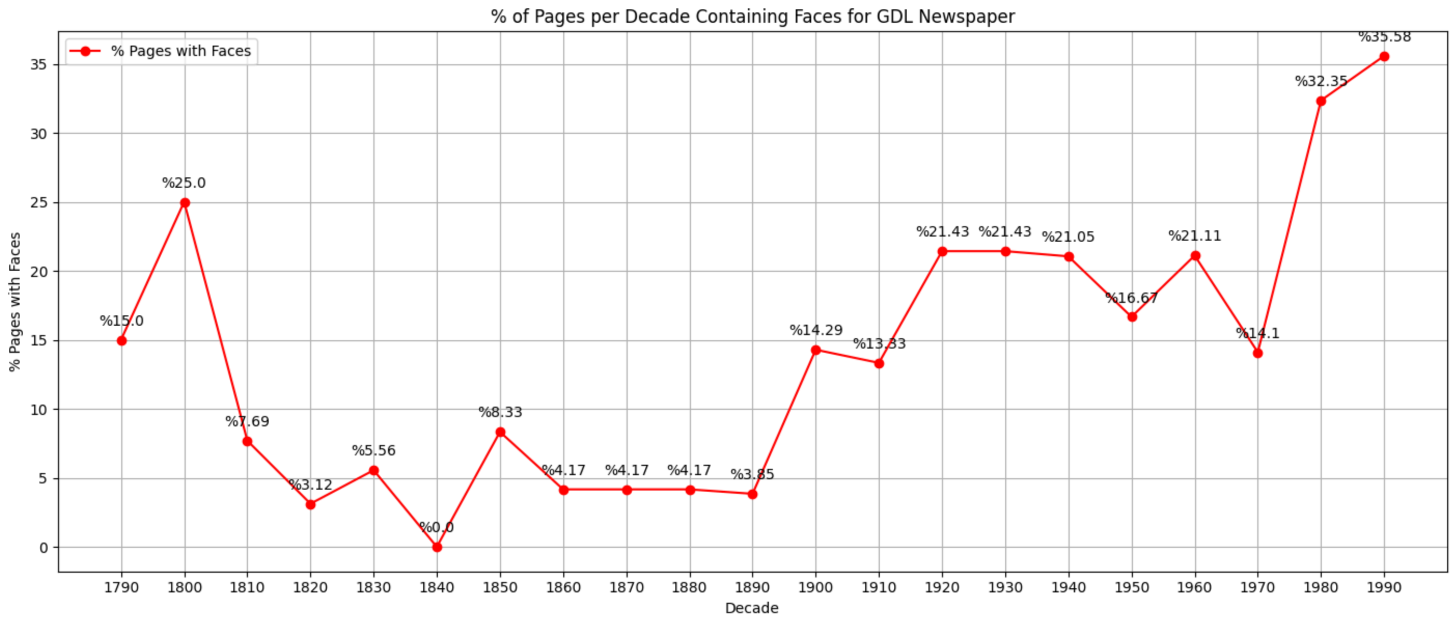
Unix: Running basic script

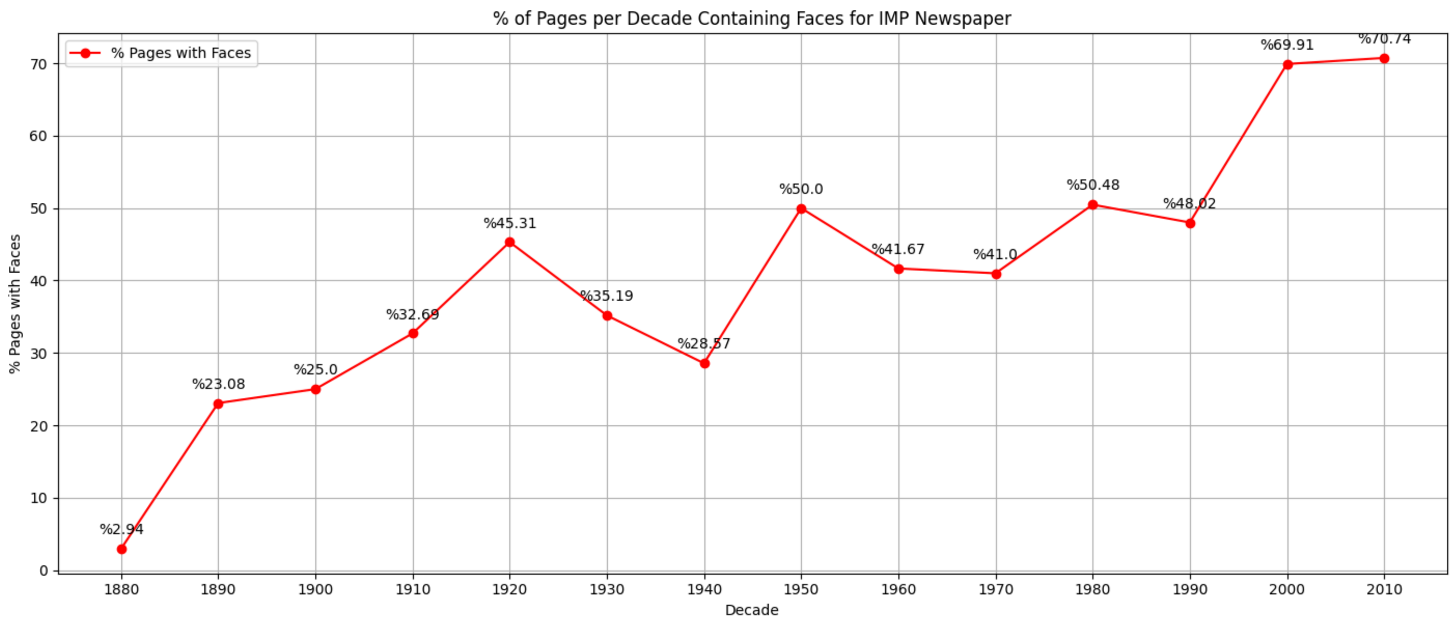
```
bash unix_run.sh
```

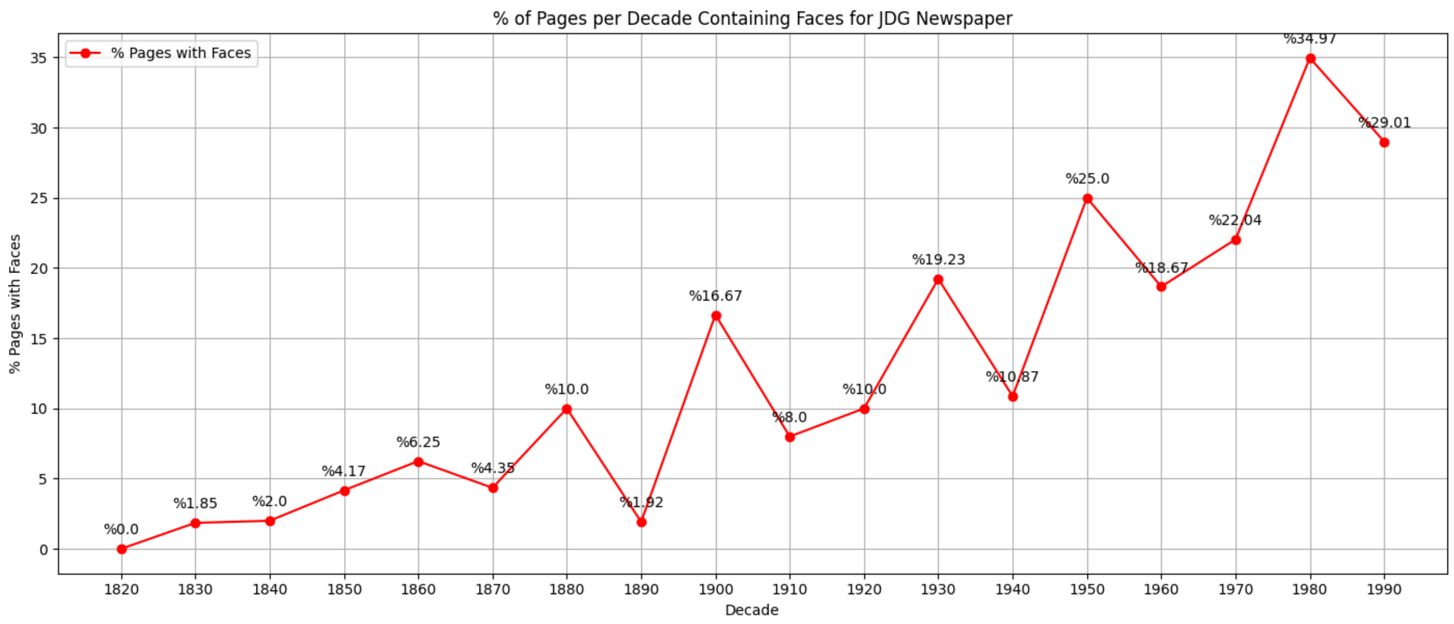
Windows: Using optional scale flag, to scale images to 50% size.

```
bash win_run.sh -s 0.5
```

Takeaways from output







By looking at the plots generated from the different newspapers, we can see that all the papers in general seem to include a larger proportional amount of faces relative to the amount of pages, as the decades increase. The exception being the GDL paper, which seemingly started off with realtively high amount of pages containing faces, which then dropped off, and the nrose again in tandem with the other papers.

The Code

Face_detection.py

The script goes through each folder in the `in` folder, and processes each image inside that folder.

The script extracts the decade information from the filenames, all following this name scheme `XYZ-1234-12-34-a-p000x`. This is then split into a list `['XYZ','1234','12','34','a','p000x']`, which we can use to extract the year at index position 1, `1234`.
We then calculate the appropriate decade using integer division: `(1234//10)*10=1230`

The script processes images within a specified, detecting faces and computing statistics on a per-decade basis. It uses the MTCNN (Multi-task Cascaded Convolutional Networks) from the facenet_pytorch library for face detection.

The script then counts the number of faces detected per image/"page", and categorizes these counts by decade. The script calculates the total number of pages, pages with faces, and the total number of faces per decade. It also calculates the percentage of pages with faces.

Each image can optionally be resized prior to procseeing, based on a specified scale factor to optimize the face detection process. This is done using `argparse` command line arguments, as explained in the section regarding code setup.

Once the processing is complete, the script generates a CSV file for each folder, containing the calculated statistics sorted by decade. This allows for an organized overview of the face detection results, providing insights into the distribution of faces over time. The CSV files are saved in the output directory.

Plot.py

This script processes the `.CSV` files generated by the previous face detection script, and creates line charts to visualizing the percentage of pages containing faces over different decades. The script operates by scanning a specified directory (`/out`) for CSV files, reading the data as a pandas dataframe, and then generating line charts using the matplotlib pyplot library.

The main function sets the folder path where the CSV files are located and calls the `process_csv_files_in_folder()` function.
The `process_csv_files_in_folder()` function iterates through all CSV files in a given folder, extracting the name of each newspaper from the filename. It reads the data into a DataFrame and calls the `line_chart()` function to generate a plot for each newspaper.
The `line_chart()` function takes a DataFrame and the column names to be used as the x and y axes, along with a title for the plot. It extracts the data from the specified columns in the dataframe and plots it, adding markers and annotations to highlight each data point's value.
The resulting plots are saved as PNG files in the same folder as the CSV files, named according to the respective newspapers.

Once all plots are generated and saved, it prints a confirmation message indicating the completion of the process.