Project Brief

AwesomeBabyNames is a program designed to analyze baby names using the US Government's census data that includes all newborns' names registered in the US from 1910 to 2018.

Instructions

To run the AwesomeBabyNames program, simply run:

```
>> python src/main.py
```

A few notes:

- Please choose to create the database if you are entering the program for the first time. The program will quit automatically with an error message if the database is not found.
- A minimal width of 750px is required for the terminal window. Plots will not be rendered correctly otherwise.
- Note that the performance for loading the data from SQLite3 is quite poor as of this version. It can take up to 10 seconds depending on the query.

Suggested User Inputs (Just for Fun)

- For Analysis [1]:
    - Enter "Mary" and "CA"
    - Result: People use the name, Mary, for both baby boys and baby girls in California.
- For Analysis [1]:
    - Enter "Gunnar" and some random states
    - Result: Gunnar is a "trendy" baby name that becomes very popular after 2010.

Project Directory

The project directory is structured by following the Python development best practices that I have learned through research. Only a few folders, however, are used for the project:

- **demo.m4v** is a 2-min demo video.
- **requirements.txt** includes a list of the required third-party packages.
- **data/** contains all the raw csv files from data.gov.
- **src/** contains all the source codes.
    - **src/abn/** contains all the modules and classes.
    - **src/main.py** is the script to be executed in Bash.

Third Party Packages Required

As specified in requirements.txt, the following third-party packages are required:

- pandas 1.0.0
- termplotlib 0.2.4

Note that termplotlib requires gnuplot, a command-line for generating plots.

Completed Features

I managed to complete two features:

- Analyze the popularity trend
- Analyze the geographic distribution

I would like to finish all other features specified in the design doc in the future. This is a tool that I personally wished to have when I had my first baby (and when I had to come up with my own English name way back in high school). I would love to convert it to an app or website for other people to take advantage of this highly underutilized dataset from the US government.

Just for fun, I generated a report of source code line count for the project:

```
-------------------------------------------------------------------------------
File                                     blank        comment           code
-------------------------------------------------------------------------------
./abn/analyzers.py                          48             70            152
./abn/models.py                             51             91            132
./abn/menus.py                              14             24             74
./abn/utils/mvc.py                          24             55             48
./abn/events.py                             10             18             27
./abn/controllers.py                        14             27             26
./abn/configs.py                             9              5             19
./abn/utils/performance.py                   9             15             19
./abn/constants.py                           4              4             17
./abn/awesomebabynames.py                    6             16             16
./main.py                                    2              3              3
-------------------------------------------------------------------------------
SUM:                                       191            328            533
-------------------------------------------------------------------------------
```

Challenges

There are three main challenges I encountered during the development:

- Software engineering
- Data processing
- Package and module design

Software Engineering

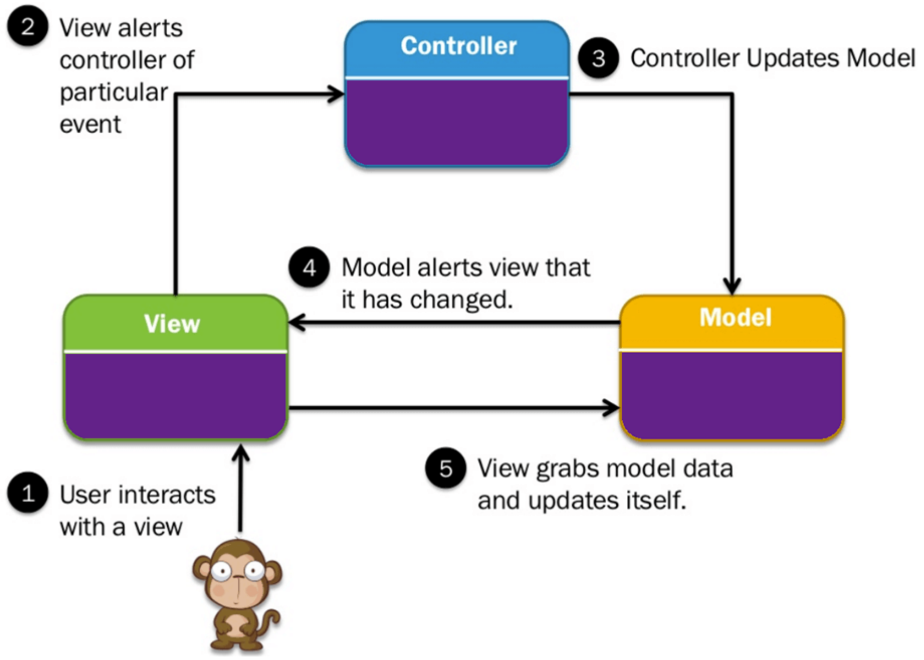
This is the first time I attempt to write a full Python program from the ground up. One goal I have is to learn how to apply design patterns using Python both for my own learning and the ease of future extensions. Specifically, I have applied the following four design patterns:

- Model-View-Controller (MVC)
- Service Locator
- Observer
- Singleton

Of the four design patterns, MVC is the most difficult to apply, but very rewarding once the foundation is laid out. MVC strictly separates and encapsulates the software into three large component groups:

- Models handle all the data processing and state management
- Views are responsible for rendering the plots and user prompts
- Controllers receive and process all the events (e.g. user input, app init, etc.)

A diagram (from Guru909) is attached below as a reference of the MVC architecture that I follow for the project:


2 View alerts controller of particular event
Controller
3 Controller Updates Model
4 Model alerts view that it has changed.
View
Model
1 User interacts with a view
5 View grabs model data and updates itself.

Data Loading & Processing

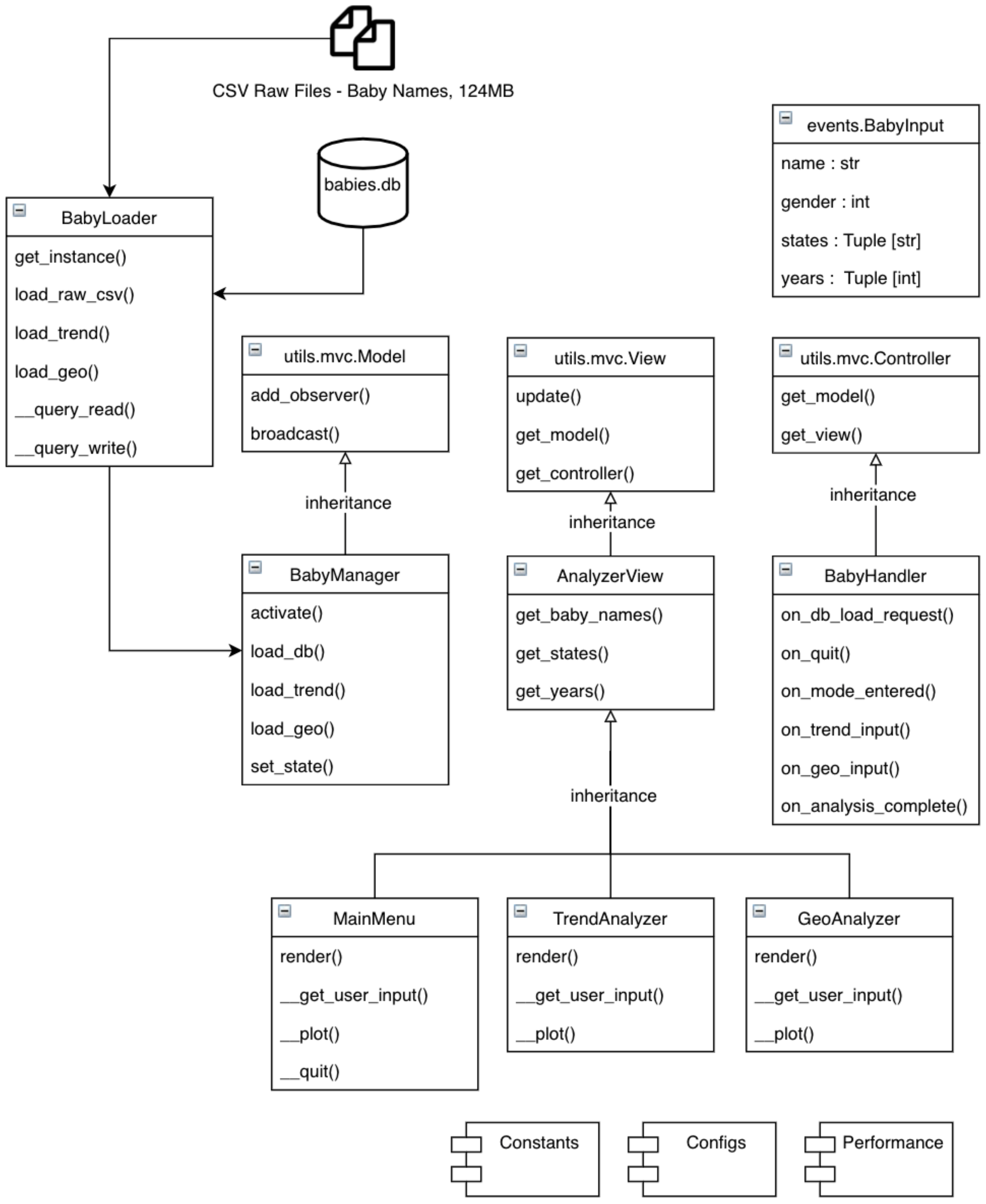
There is a total of 124MB worth of baby names data, separated into 51 different CSV files, that I need to load, transform, and render. Unfortunately, I have zero experience using NumPy, Pandas, or any package for plotting prior to the project. I also have very limited knowledge on SQLite 3.

I have no choice but to learn everything from the ground up, which is one reason why I am unable to complete all features given the time I have. I find the learning curve quite steep and time-consuming. The third-party package, termplotlib, in particular, comes with limited documentation so I have to dig into the source codes to find what I need.

Package & Module Design

The concept of modules in Python is new to me. I am familiar with the concept of packages from my prior experience with ActionScript/JavaScript/Java, but have difficulty determining if module or package is better for structing my classes. This is one question I plan to take to the office hours for further discussion.

Appendix: Architecture UML


CSV Raw Files - Baby Names, 124MB
babies.db
events.BabyInput
name : str
gender : int
states : Tuple [str]
years : Tuple [int]
BabyLoader
get_instance()
load_raw_csv()
load_trend()
load_geo()
__query_read()
__query_write()
utils.mvc.Model
add_observer()
broadcast()
utils.mvc.View
update()
get_model()
get_controller()
utils.mvc.Controller
get_model()
get_view()
inheritance
inheritance
inheritance
BabyManager
activate()
load_db()
load_trend()
load_geo()
set_state()
AnalyzerView
get_baby_names()
get_states()
get_years()
BabyHandler
on_db_load_request()
on_quit()
on_mode_entered()
on_trend_input()
on_geo_input()
on_analysis_complete()
inheritance
MainMenu
render()
__get_user_input()
__plot()
__quit()
TrendAnalyzer
render()
__get_user_input()
__plot()
GeoAnalyzer
render()
__get_user_input()
__plot()
Constants
Configs
Performance