covid

This is a project to download and play with the data collated by the Johns Hopkins CSSE folks.

This should work for you if you clone the project. The file structure depends on a `base` path, which is established by an environmental variable.

In my `~/.zshrc` I have the line:

```
covid_base=$HOME'/Dropbox/Github/covid'
export covid_base
alias cov='cd $covid_base && pwd'
```

You could do this as a one-off in Terminal with just the first line

```
covid_base=$HOME'/path/to/covid'
```

Then, in the scripts, one of the first things that usually happens is:

```
import sys, os

base = os.environ.get('covid_base')
sys.path.insert(0,base)
```

From there, it all just works.

Run this at home

I'm using Python3 for this project. I installed it with Homebrew.

Most things should work with Python2. The tests module uses `subprocess.run`, which requires Python3.

Also, I installed `plotly` and `pandas` with pip3. These are needed for the choropleth maps. I don't know how well they work with Python2.

To test, I used the web interface to Github to clone the project by downloading a zipfile and then opened it. The directory name is covid-master. Then I set the environmental variable

```
covid_base=$HOME"/Desktop/covid-master"
```

and everything seems to work.

More

This is all Python3 code now, after a recent update.

My version of the database is constructed from their database files by **update.py**. This checks the 'csv.source' directory and if it's not up-to-date, downloads the appropriate data files from their data.

As of this morning, there are two databases: one with just the data from this month, and one with all the data. You can construct the latter by passing the argument `--max` to `build/update.py`.

The scripts use the standard database. The mega version is for some not-yet-written stuff to plot the entire course of the pandemic.

My database looks like

```
2020-03-22
2020-04-28

Autauga;Alabama;01001;US
0,0,1,4 ...
0,0,0,0 ...

...
```

Rather than mark each data point with the date, we just track the first and last dates for the database as a whole. This means we have to watch for when updates don't happen properly, if a new key appears, or if a county decides not to report after a while.

Options for scripts are:

- the US broken down by states: `all_states.py`
- the US broken down by counties: `us_by_counties.py`
- a given state broken down by counties: `one_state.py`

Features that are currently supported are given by the `--help` flag:

```
> python scripts/one_state.py --help

flags
-h  --help    help
-n   <int>    display the last n values, default: 7
-N   <int>    display N rows of data: default: 50

-c  --delta   change or delta, display day over day rise
-d  --deaths  display deaths rather than cases (default)
-r  --rate    compute statistics
-s  --sort    (only if stats are asked for)

to do:
-u   <int>    data slice ends this many days before yesterday
-p  --pop     normalize to population

example:
python scripts/one_state.py  -n 10 -sdr

>
```

The statistic is the slope of a linear regression, divided by the mean of the values.

So, for example, if a 10-day series goes smoothly from 100 to 110, then the slope is about 10/10 = 1 and the statistic is a bit less than 0.01. If the series goes from 1000 to 1100, then the slope is about 100/10 = 10, but the statistic is still approximately 0.01.

More recently, I have started making what are called choropleth plots, geographic plots where the fill color is based on the statistic for case growth (or whatever else you want). These can be found in `geo`, and examples are in `results`.