Melbourne Bioinformatics Seminar

Hatching a plot on the command line

Bernard Pope
Lead Bioinformatician, Cancer and Clinical Genomics
Victorian Health and Medical Research Fellow
Melbourne Bioinformatics
The University of Melbourne, Australia

Plotting tabular data

- Like most bioinformaticians and data analysts I spend a lot of time dealing with tabular data:
    - VCF
    - CSV, TSV
    - BED
- Often this data is quite large: lots of rows, lots of columns, lots of files.
- Unsurprisingly, I also do a lot of plotting of this data.

Plotting tabular data

- Of course you can plot the data in R, Python, spreadsheets etc.
- However, most of the plotting I do is the same kind of thing, e.g:
    - Plot the distribution of values by group.
    - Make a scatter plot comparing two variables.
    - … line plots, bar plots, heat maps and so on.
- While Python + Pandas + Seaborn can do what I want, writing custom plotting code is tedious and repetitive.

Reinventing the wheel

- Previously I used to use Gnuplot, but I find it rather fiddly.
- Unsurprisingly, there's plenty of other similar tools online, but:
    - They are controlled by other people.
    - Their tastes and needs are often different to mine.
- So, in the grand tradition of Bioinformatics, I reinvented the wheel.

hatch

- https://github.com/bjpop/hatch
- 2D plots from tabular data in (wide) CSV/TSV, output to PNG:
    - Histograms (regular and cumulative)
    - Distributions by group (box and violin)
    - Scatter plots
    - Line plots
    - Heatmaps
    - Counts (bar plots)
- Row filtering.
- Essentially a command line wrapper around Python + Pandas + Seaborn (and thus Numpy and Matplotlib).

installation

```
$ git clone https://github.com/bjpop/hatch
$ python3 -m venv hatch_dev
$ source hatch_dev/bin/activate
$ pip install -U /path/to/hatch
```

Plan to eventually support PyPI, conda, Docker *etcetera* installs.

Aside: repo was started with Bionitio

- The hatch repository was created using Bionitio!
- Easy to get started.
- Has (some) batteries included.

Getting help

```
$ hatch -h
usage: hatch [-h] [-v] {hist,dist,scatter,line,count,heatmap} ...

Generate plots of tabular data

optional arguments:
  -h, --help            show this help message and exit
  -v, --version         show program's version number and exit

Plot type:
  {hist,dist,scatter,line,count,heatmap}
                        sub-command help
    hist                Histograms of numerical data
    dist                Distributions of numerical data
    scatter             Scatter plots of numerical data
    line                Line plots of numerical data
    count               Counts (bar plots) of categorical data
    heatmap             Heatmap of two categories with numerical values
```

Getting help for a specific plot type

```
$ hatch scatter -h
usage: hatch scatter [-h] [--outdir DIR] [--filetype FILETYPE] [--name NAME]
                     [--logfile LOG_FILE] [--nolegend] [--filter EXPR]
                     [--navalues STR] [--title STR] [--width SIZE]
                     [--height SIZE] --xy X,Y [X,Y ...] [--logx] [--logy]
                     [--xlim LOW HIGH LOW HIGH] [--ylim LOW HIGH LOW HIGH]
                     [--hue FEATURE] [--size FEATURE] [--alpha ALPHA]
                     [--linewidth WIDTH]
                     DATA

positional arguments:
  DATA                  Filepaths of input CSV/TSV file

optional arguments:
  -h, --help            show this help message and exit
  --outdir DIR          Name of optional output directory.
  --filetype FILETYPE   Type of input file
  --name NAME           Name prefix for output files
 … etc etc …
```

Output truncated to fit on slide

Simple example

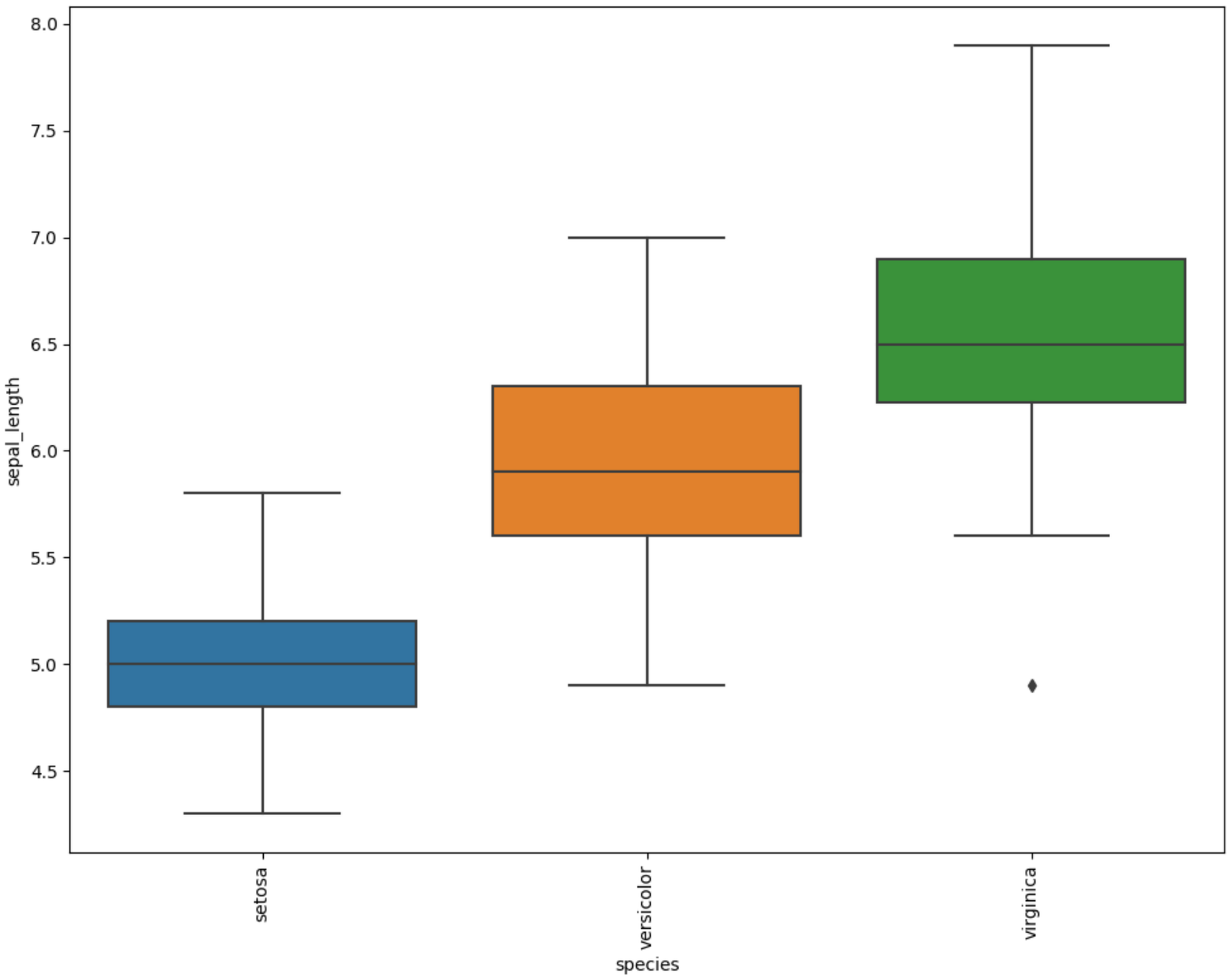
```
$ head -1 iris.csv

sepal_length,sepal_width,petal_length,petal_width,species
```

```
$ hatch dist --columns sepal_length --group species -- iris.csv
```

Output goes to:
`iris.sepal_length.species.dist.png`

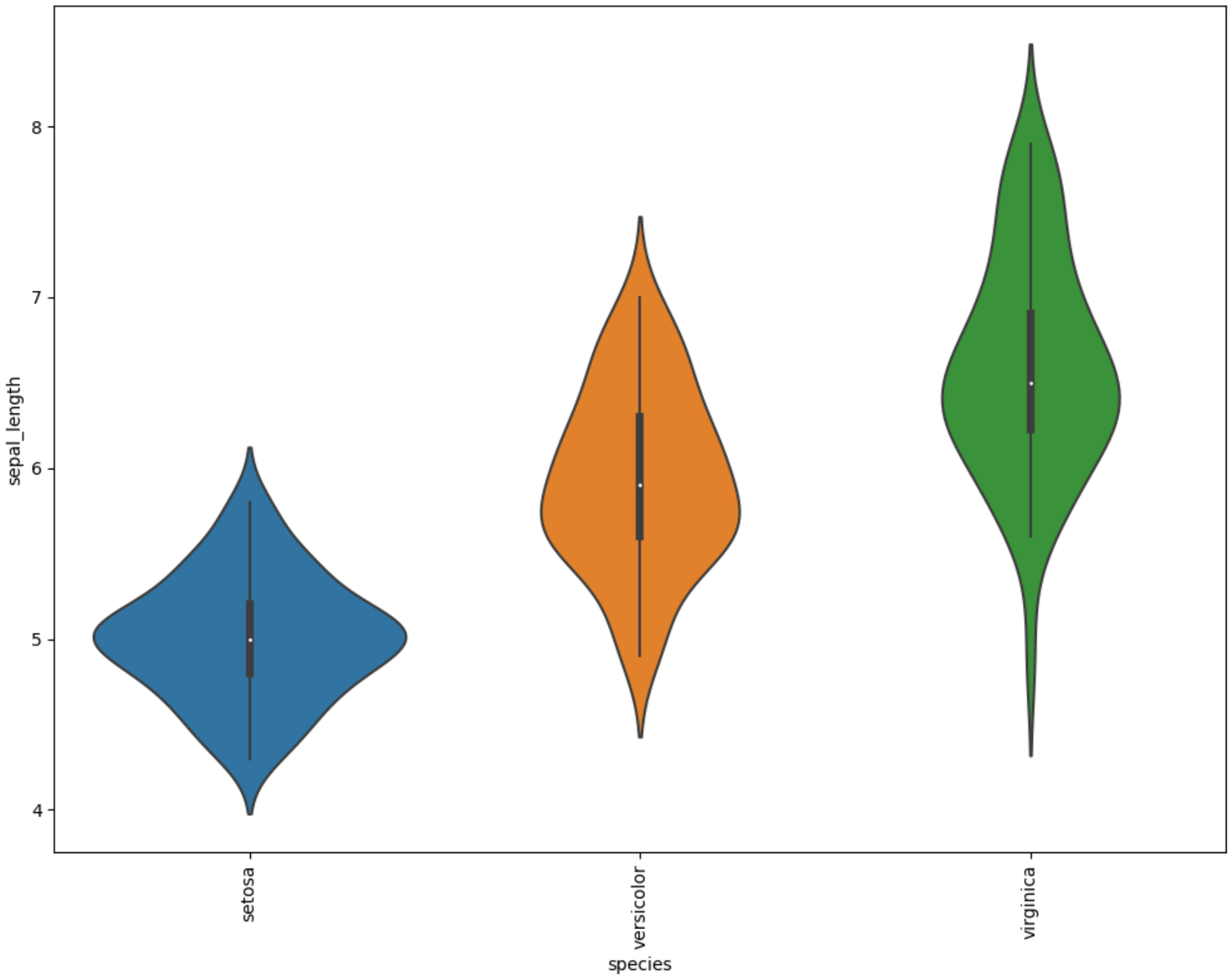
Simple example



Simple example

```
$ hatch dist --columns sepal_length --group species \
             --type violin -- iris.csv
```

Simple example



Filtering rows

- The `--filter` flag takes a Python expression as its argument.
- The expression uses Pandas data frame query notation to filter rows.
- Rows which make the expression true are retained, all others are discarded.

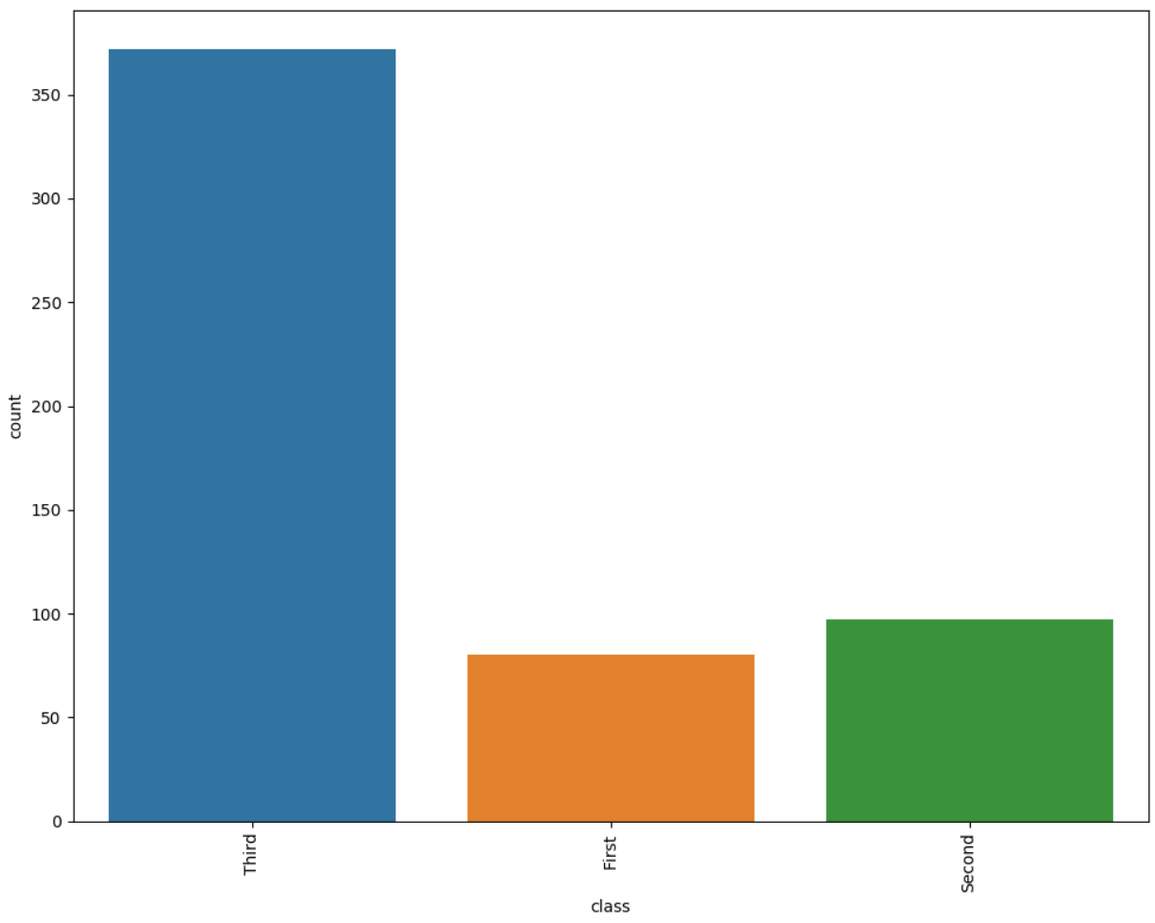
More complex example

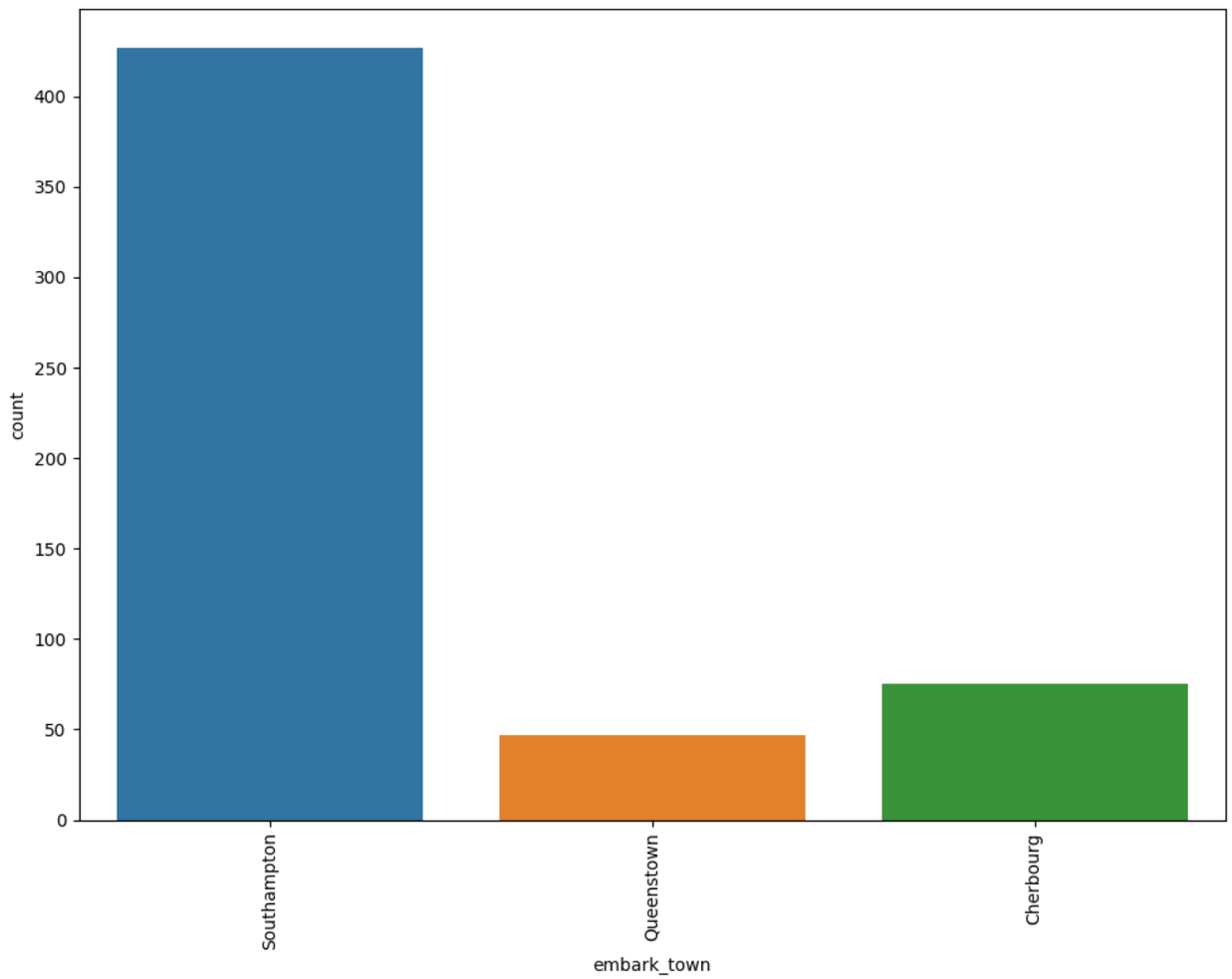
```
$ head -1 titanic.csv
survived,pclass,sex,age,sibsp,parch,fare,embarked,class,who,adult_m
ale,deck,embark_town,alive,alone
```

```
$ hatch count --columns class embark_town \
    --filter "survived == 0" -- titanic.csv
```

Outputs go to:
`titanic.class.count.png`
`titanic.embark_town.count.png`

More complex example





Performance on large(ish) datasets

- The test file `varlap.csv` has 11 columns and ~ 1.4 million rows, and is 82 MB in size.
- That is larger than the maximum number of rows that Excel can handle.
- While it is not HUGE, it is big enough to be a problem for many tools.

Performance on large(ish) datasets

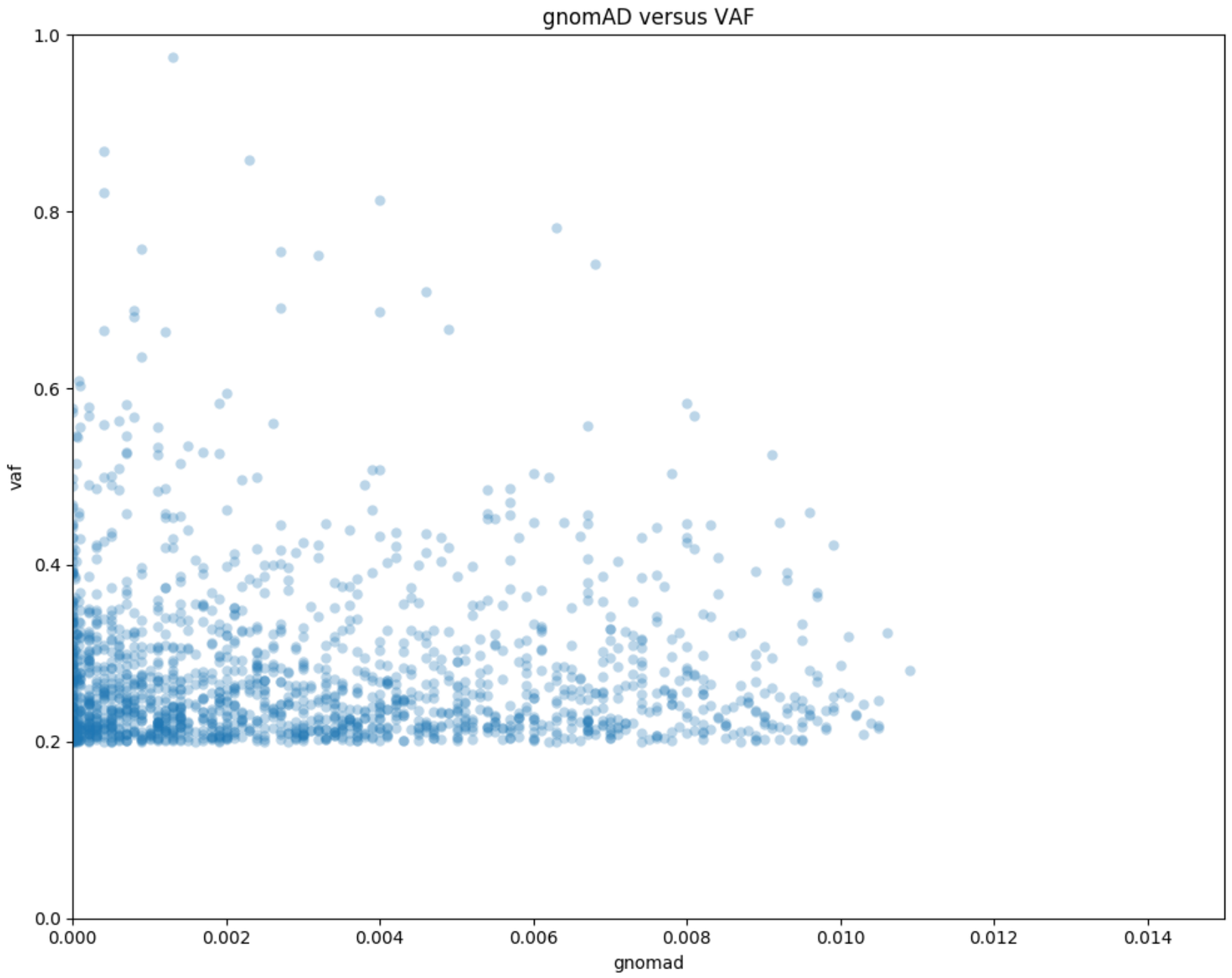
```
$ time \
  hatch scatter --xy gnomad,vaf \
  --filter "callers == 'Sanger' and vaf >= 0.2" \
  --xlim 0 0.015 --ylim 0 1 --title "gnomAD versus VAF" -- \
  varlap.csv

real   0m5.701s
user   0m4.132s
sys    0m1.101s
```

Only 5 seconds to plot the graph!

Output goes to:
`varlap.gnomad.vaf.scatter.png`

Performance on large(ish) datasets



Data transformations?

- Sometimes CSV data comes in an inconvenient format, which can be a pain for plotting.
- Maybe `hatch` should handle some basic data transformations, e.g. reshape?
- Generally I've decided against this (except for filtering) because there are some good tools that can already do it:
    - xsv (really fast!)
    - miller, csvkit
- But I might change my mind about this.

Future work

- Features are added as I need them.
- However, I'd really like some other users:
    - Test it out, find bugs, suggest/add features.
        - Keep me honest, and motivate me to polish the sharp corners.
- More plot types.
- Better error messages and sanity checking.
- Some statistical calculations, e.g. ANOVA etc.
- Different plotting backends, e.g. Plotly for interactivity.