Hatching a plot (and more) on the command line

Bernard Pope
Victorian Health and Medical Research Fellow
Melbourne Bioinformatics
The University of Melbourne, Australia

Bioinformatician's Shed

Overview

- Hatch is a **command line** tool for **analysing** and **visualising** data.
- Input is tabular data in CSV or TSV format.
- Output is (one or both of):
    - plot(s)
    - transformed tabular data in CSV or TSV format
- Built on top of pandas, scikit-learn, numpy, matplotlib, seaborn.

Simple example

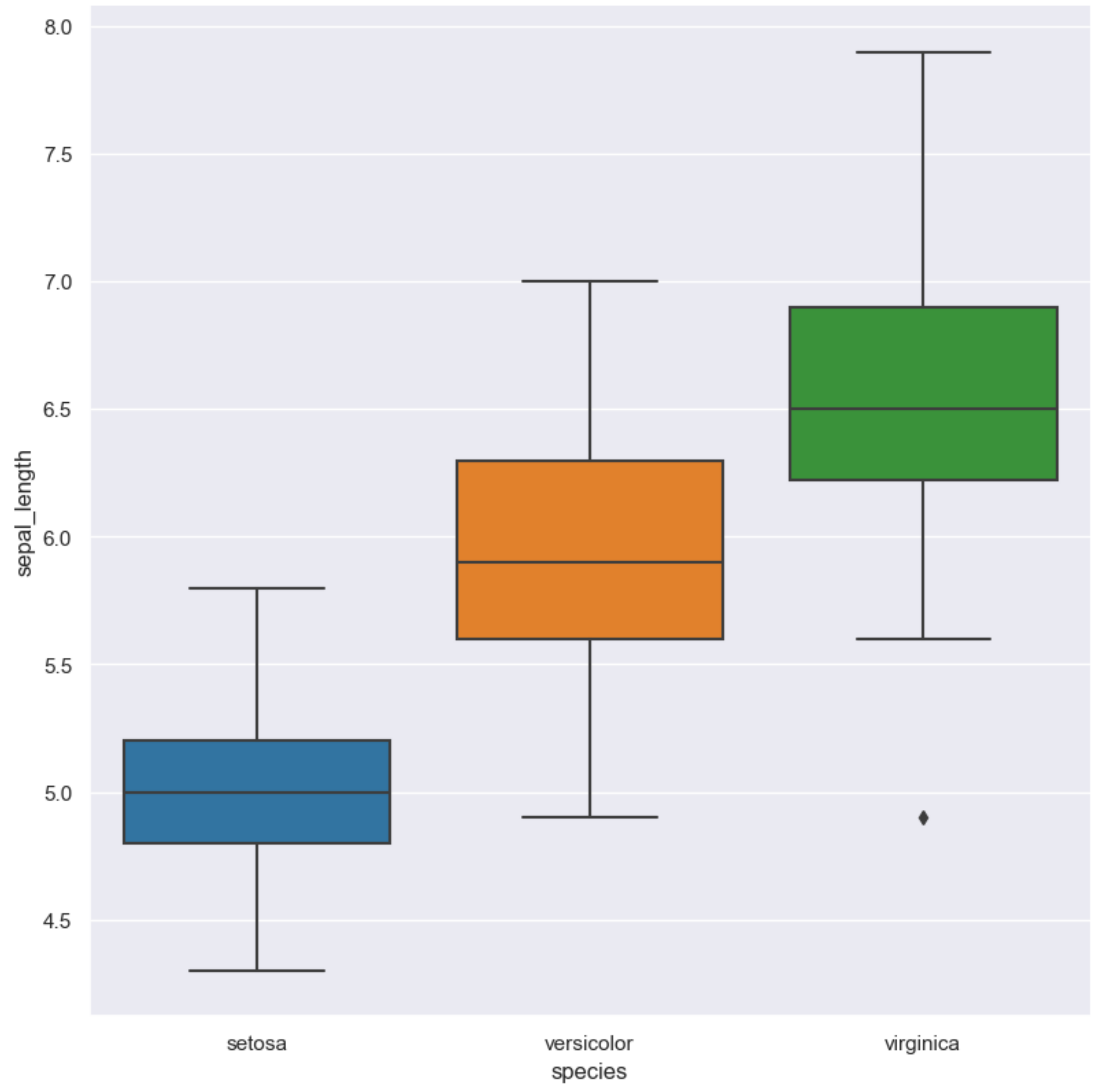
```
cat iris.csv | hatch pretty
 sepal_length  sepal_width  petal_length  petal_width    species
          5.1          3.5           1.4          0.2     setosa
          4.9          3.0           1.4          0.2     setosa
          4.7          3.2           1.3          0.2     setosa
          4.6          3.1           1.5          0.2     setosa
          5.0          3.6           1.4          0.2     setosa
...                    ...           ...          ...        ...
          6.7          3.0           5.2          2.3  virginica
          6.3          2.5           5.0          1.9  virginica
          6.5          3.0           5.2          2.0  virginica
          6.2          3.4           5.4          2.3  virginica
          5.9          3.0           5.1          1.8  virginica

[150 rows x 5 columns]
```

Simple example

```
cat iris.csv | hatch describe

        sepal_length  sepal_width  petal_length  petal_width     species
count     150.000000   150.000000    150.000000   150.000000         150
unique           NaN          NaN           NaN          NaN           3
top              NaN          NaN           NaN          NaN  versicolor
freq             NaN          NaN           NaN          NaN          50
mean        5.843333     3.054000      3.758667     1.198667         NaN
std         0.828066     0.433594      1.764420     0.763161         NaN
min         4.300000     2.000000      1.000000     0.100000         NaN
25%         5.100000     2.800000      1.600000     0.300000         NaN
50%         5.800000     3.000000      4.350000     1.300000         NaN
75%         6.400000     3.300000      5.100000     1.800000         NaN
max         7.900000     4.400000      6.900000     2.500000         NaN
```

Simple example

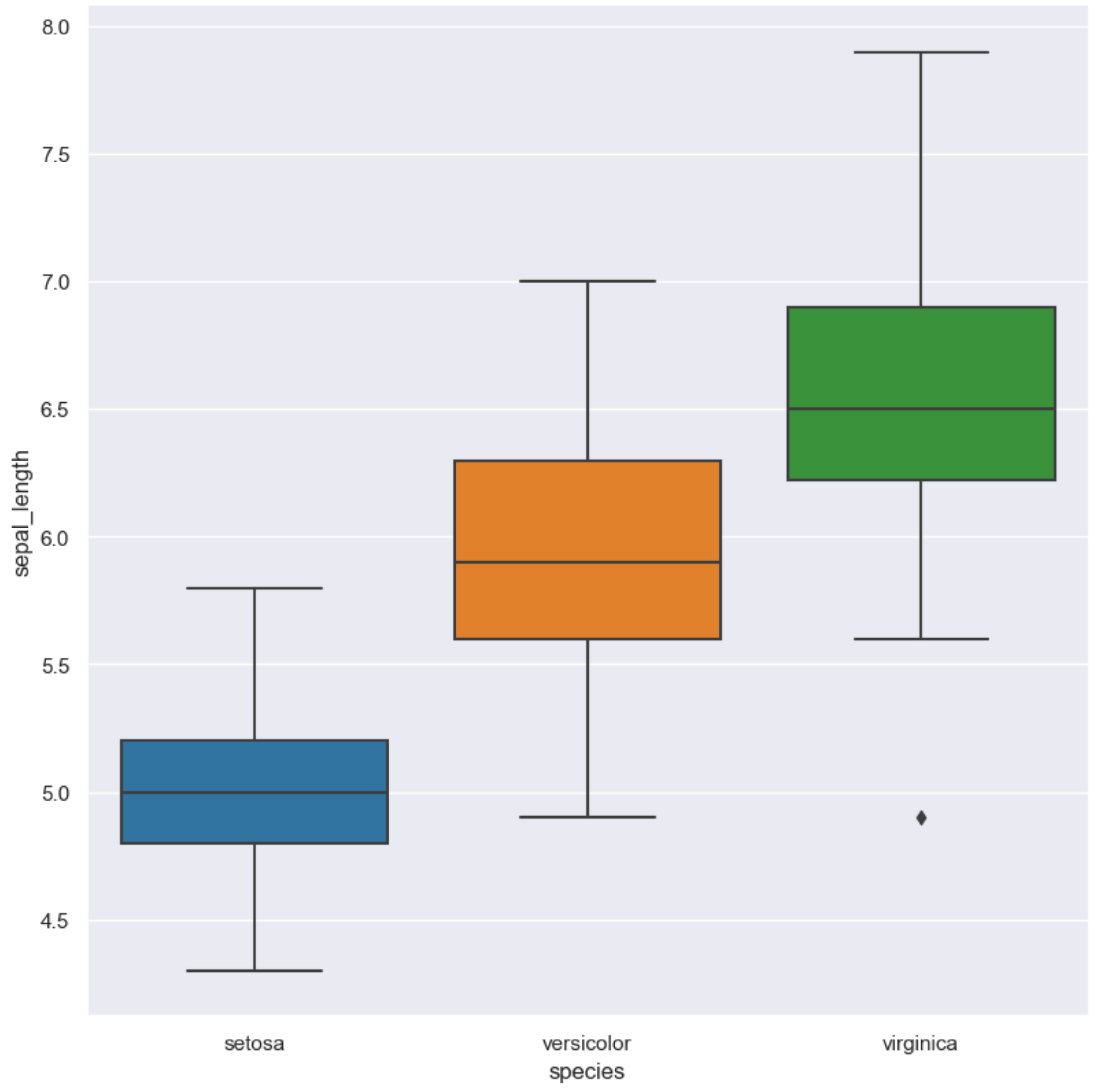
```
cat iris.csv | hatch box -x species -y sepal_length
```



output is written to
`hatch.species.sepal_length.box.png`

Equivalently

```
hatch box -x species -y sepal_length < iris.csv
```

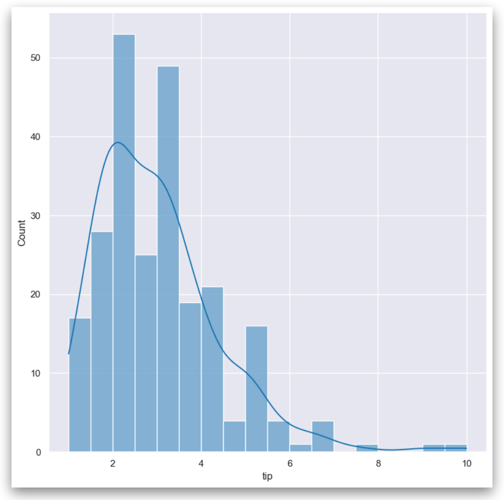


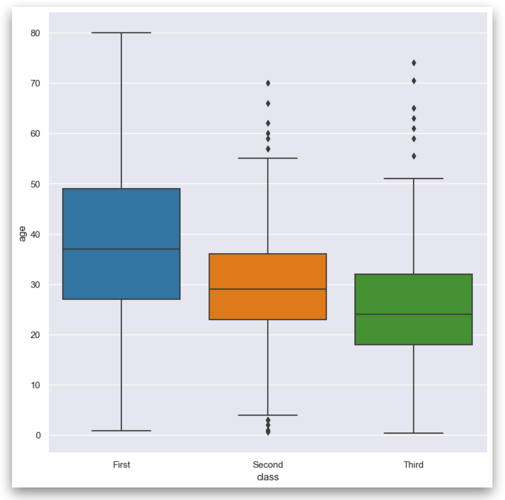
output is written to
`hatch.species.sepal_length.box.png`

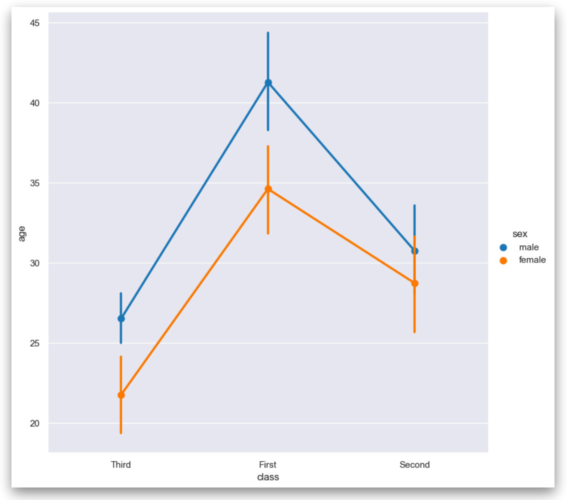
Purpose and philosophy

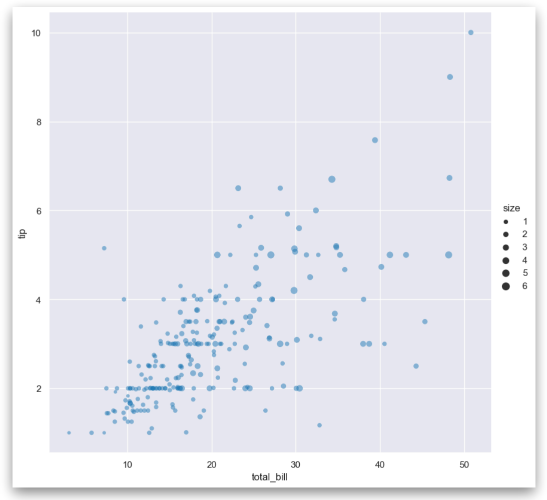
- **Fast**, **convenient** and **flexible** data analytics on the command line.
- Large input data sets are supported (> millions rows).
- Simple tasks should be easy. Complex tasks should be possible.
    - Highly customisable, but sensible defaults used extensively.

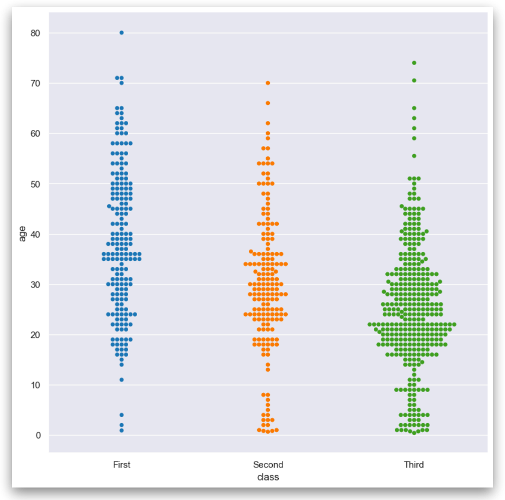
Supported basic plot types

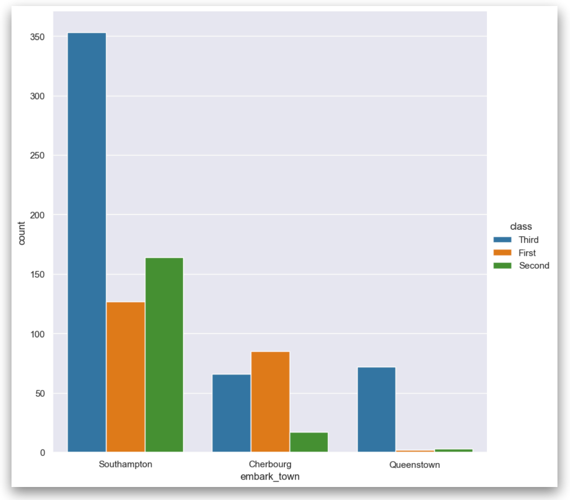

histogram

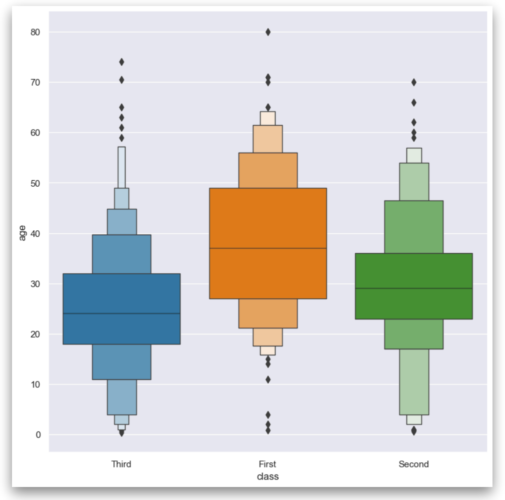

box

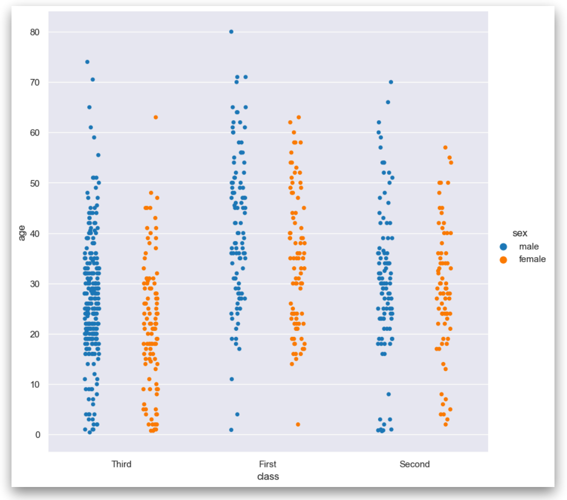

point

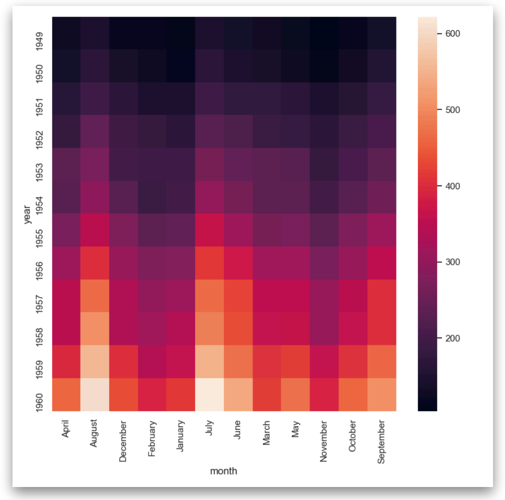

scatter

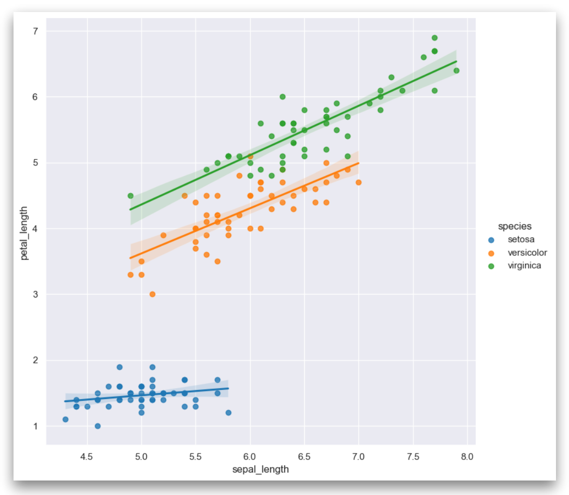

swarm

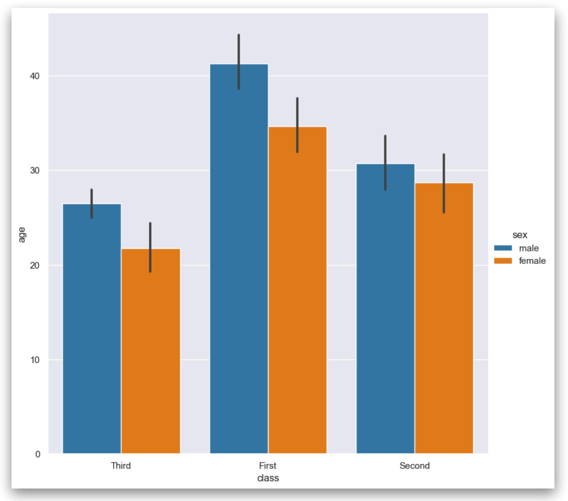

count

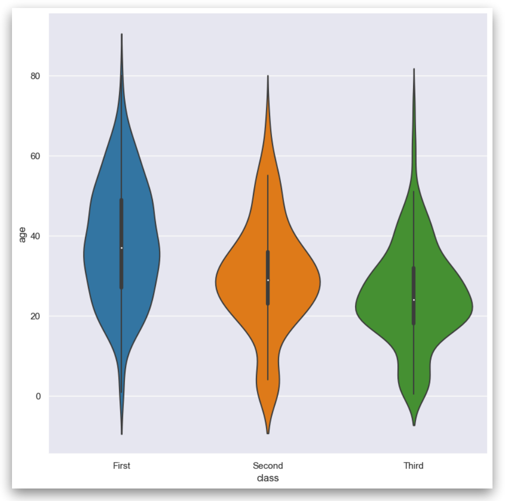

boxen

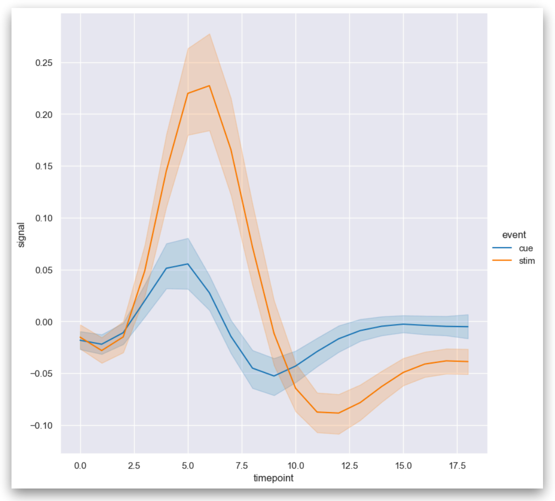

strip

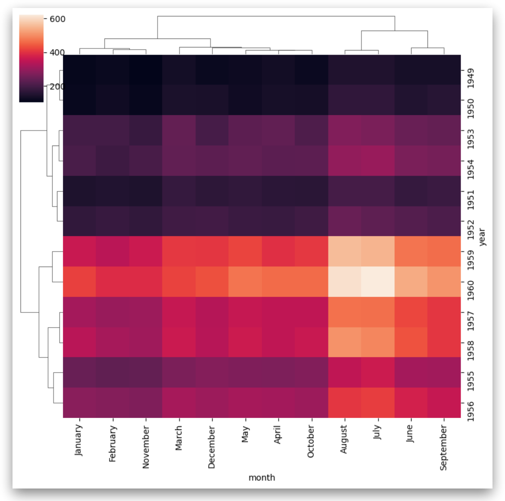

heat map

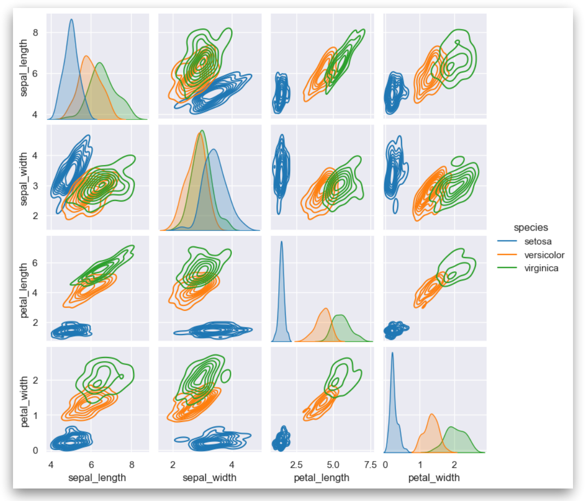

regression


bar


violin

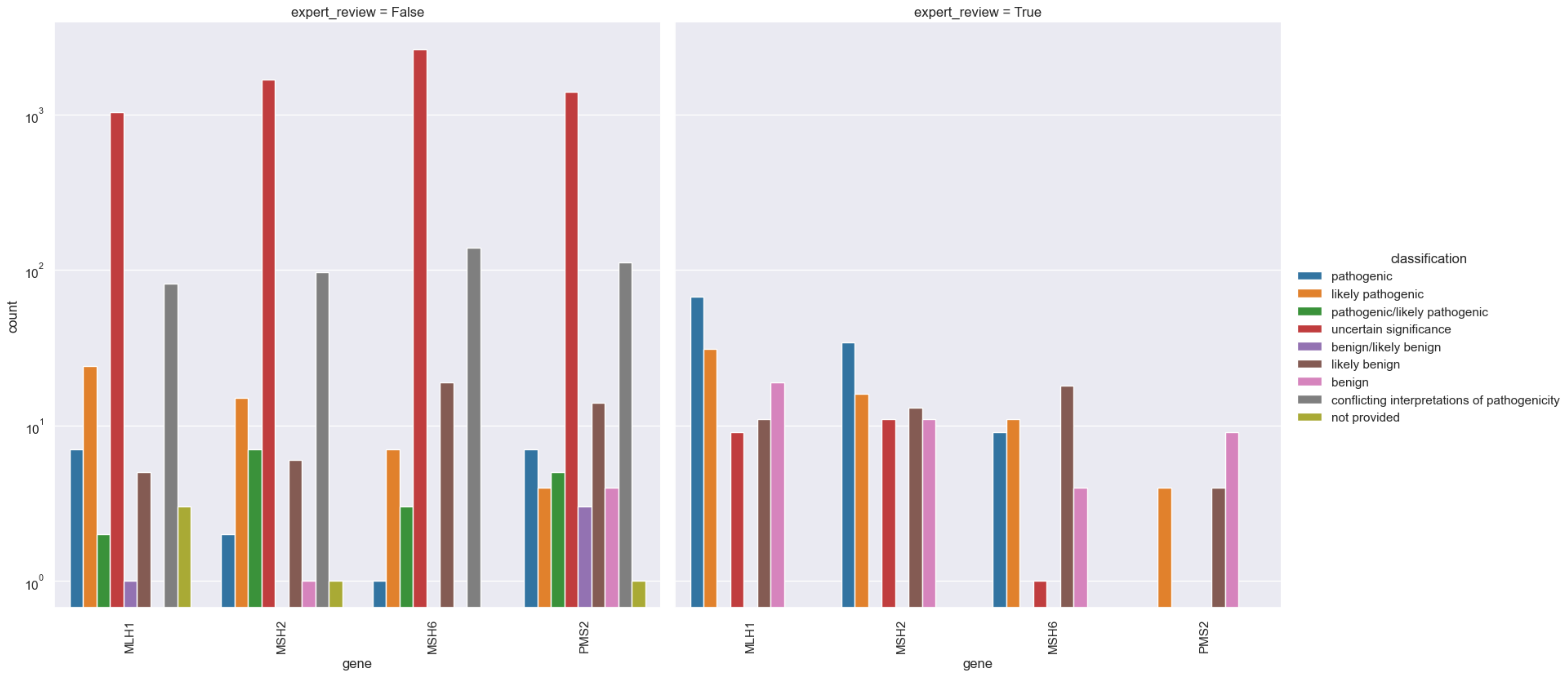

line


cluster map


pair plot

Facets

```
cat mmr.csv | hatch count -x gene --hue classification --col expert_review
```



Supported analysis and data transformations

name	description
corr	pairwise correlation
cut	column selection
drop	drop rows or columns containing missing values
eval	compute new columns from existing data
filter	filter rows using a logical expression
gmm	Gaussian mixture model clustering
head	select the first N rows in the data
isnorm	test if numerical features follow normal distribution
kmeans	k-means clustering

name	description
melt	reshape wide format into long format
outlier	detect outliers using interquartile range
pca	principal component analysis
pivot	reshape long format into wide format
sample	randomly sample rows
sort	sort rows
tail	select the last N rows in the data
score	compute z-score for numerical columns

Data summary commands

name	description
describe	summary statistics for each column
pretty	pretty print the first few and last few rows

Simple tasks should be easy. Complex tasks should be possible.

- Hatch provides a suite of commands, each carrying out a analytics or plotting task.
- Commands can be chained together in a **modular** fashion for more complex tasks.

Syntax for command chain

```
hatch command₁ + command₂ + . . . + commandₙ
```

Syntax for command chain

```
hatch command1 + command2 + . . . + commandn
```

multiple commands are separated by +

Syntax for command chain

$\xrightarrow{\text{data}_0}$ $\xrightarrow{\text{data}_1}$ $\xrightarrow{\text{data}_2}$ $\xrightarrow{\text{data}_{n-1}}$ $\xrightarrow{\text{data}_n}$

$\texttt{hatch command}_1 + \texttt{command}_2 + \,.\;.\;.\, + \texttt{command}_n$

data flows from left to right
and may be transformed
along the way

Syntax for command chain

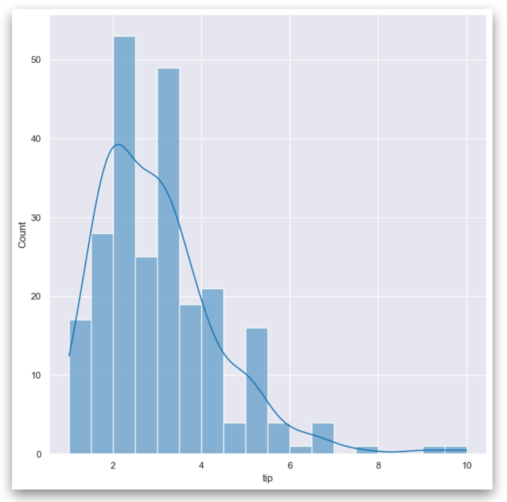
$\xrightarrow{\text{data}_\alpha}$ $\xrightarrow{\text{data}_\alpha}$

plot commands pass data on unchanged

hatch command$_1$ + histogram + . . . + command$_n$

$\big\downarrow$ image

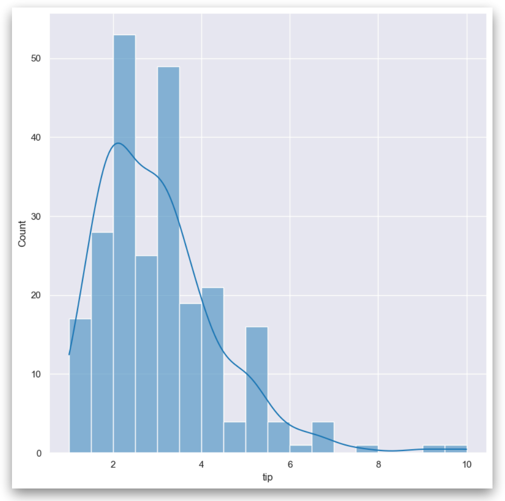
plot commands generate images

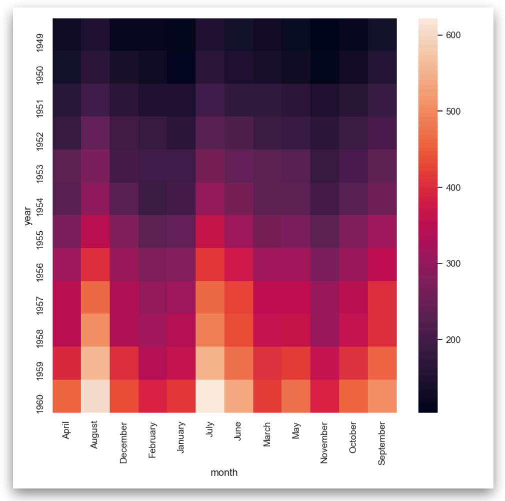


Syntax for command chain

complex chains can be formed and multiple plots are possible

hatch command$_1$ + plot$_1$ + command$_2$ + plot$_2$ + . . .





Commands have optional arguments

```
hatch command₁ [args] + . . . + commandₙ [args]
```

Input

- By default Hatch reads input from stdin and assumes that the data is in CSV format.
- This behaviour can be modified if needed:
    - Reading from stdin but in TSV format:

      ```
      hatch stdin --format tsv + ...
      ```

    - Reading from a named file:

      ```
      hatch in iris.tsv + ...
      ```

file format guessed from filename extension where possible

Output

- If the last command in a chain is not a plot or an explicit output command Hatch will write the (possibly transformed) data to stdout in CSV format.
- This behaviour can be modified if needed:
    - Writing to stdout but in TSV format:

```
hatch ... + stdout --format tsv
```

    - Writing to a named file:

```
hatch ... + out iris.tsv
```

file format guessed from filename extension where possible

Help

- Overall help:

```
hatch -h
```

- Help for a specific command:

```
hatch command -h
```

  - For example:

```
hatch scatter -h
```

Data transformation example

```
cat iris.csv | hatch sort -c sepal_length + tail 10
sepal_length,sepal_width,petal_length,petal_width,species
7.2,3.2,6.0,1.8,virginica
7.2,3.6,6.1,2.5,virginica
7.3,2.9,6.3,1.8,virginica
7.4,2.8,6.1,1.9,virginica
7.6,3.0,6.6,2.1,virginica
7.7,2.8,6.7,2.0,virginica
7.7,2.6,6.9,2.3,virginica
7.7,3.8,6.7,2.2,virginica
7.7,3.0,6.1,2.3,virginica
7.9,3.8,6.4,2.0,virginica
```

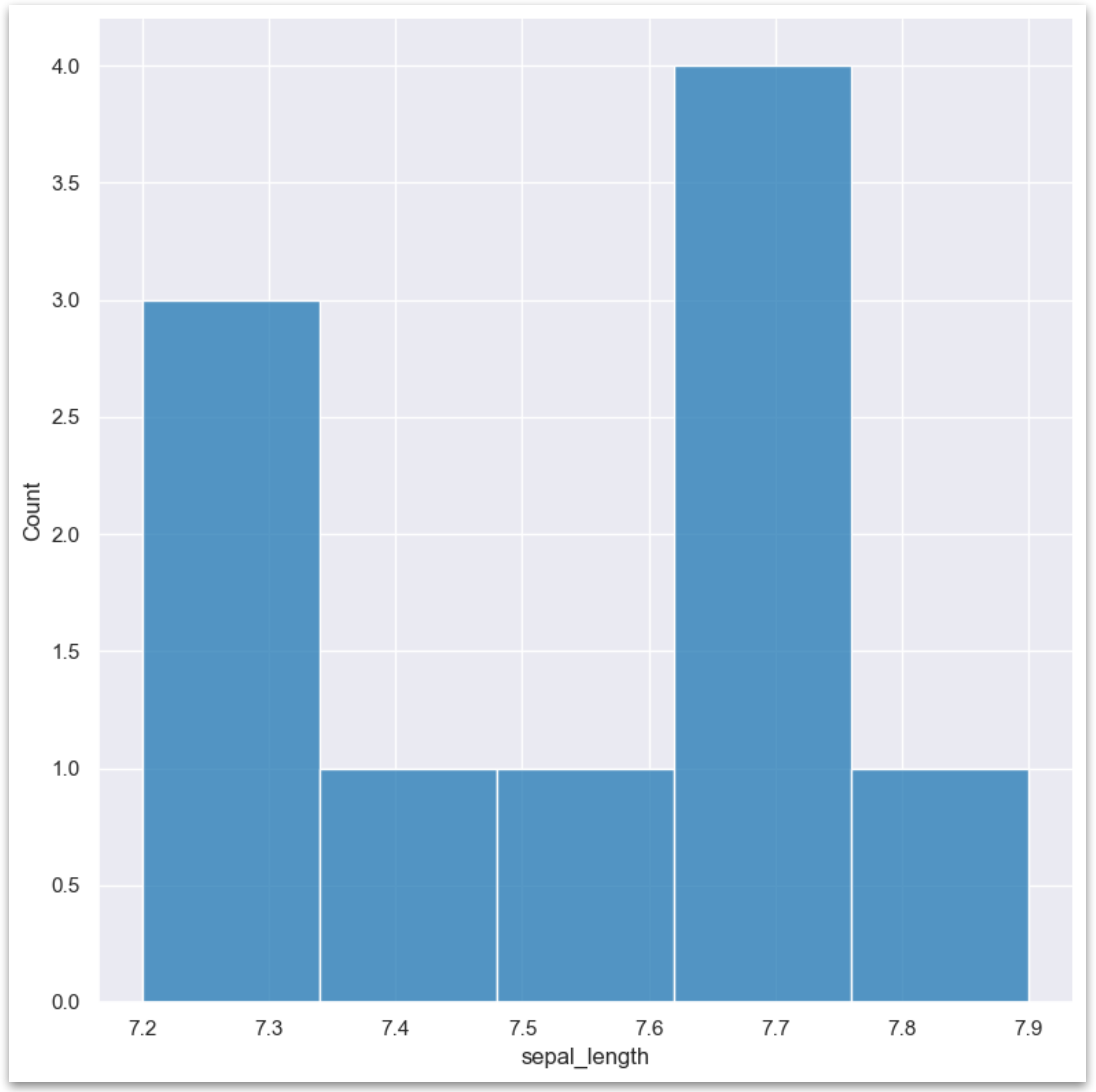
Data transformation example

```
cat iris.csv | hatch sort -c sepal_length + tail 10
sepal_length,sepal_width,petal_length,petal_width,species
7.2,3.2,6.0,1.8,virginica
7.2,3.6,6.1,2.5,virginica
7.3,2.9,6.3,1.8,virginica
7.4,2.8,6.1,1.9,virginica
7.6,3.0,6.6,2.1,virginica
7.7,2.8,6.7,2.0,virginica
7.7,2.6,6.9,2.3,virginica
7.7,3.8,6.7,2.2,virginica
7.7,3.0,6.1,2.3,virginica
7.9,3.8,6.4,2.0,virginica
```

input is read from `stdin`, then sorted (ascending) on the `sepal_length` column, then the last 10 rows are selected and output to `stdout`

Data transformation and plotting example

```
cat iris.csv | hatch sort -c sepal_length + tail 10 + histogram -x sepal_length
```

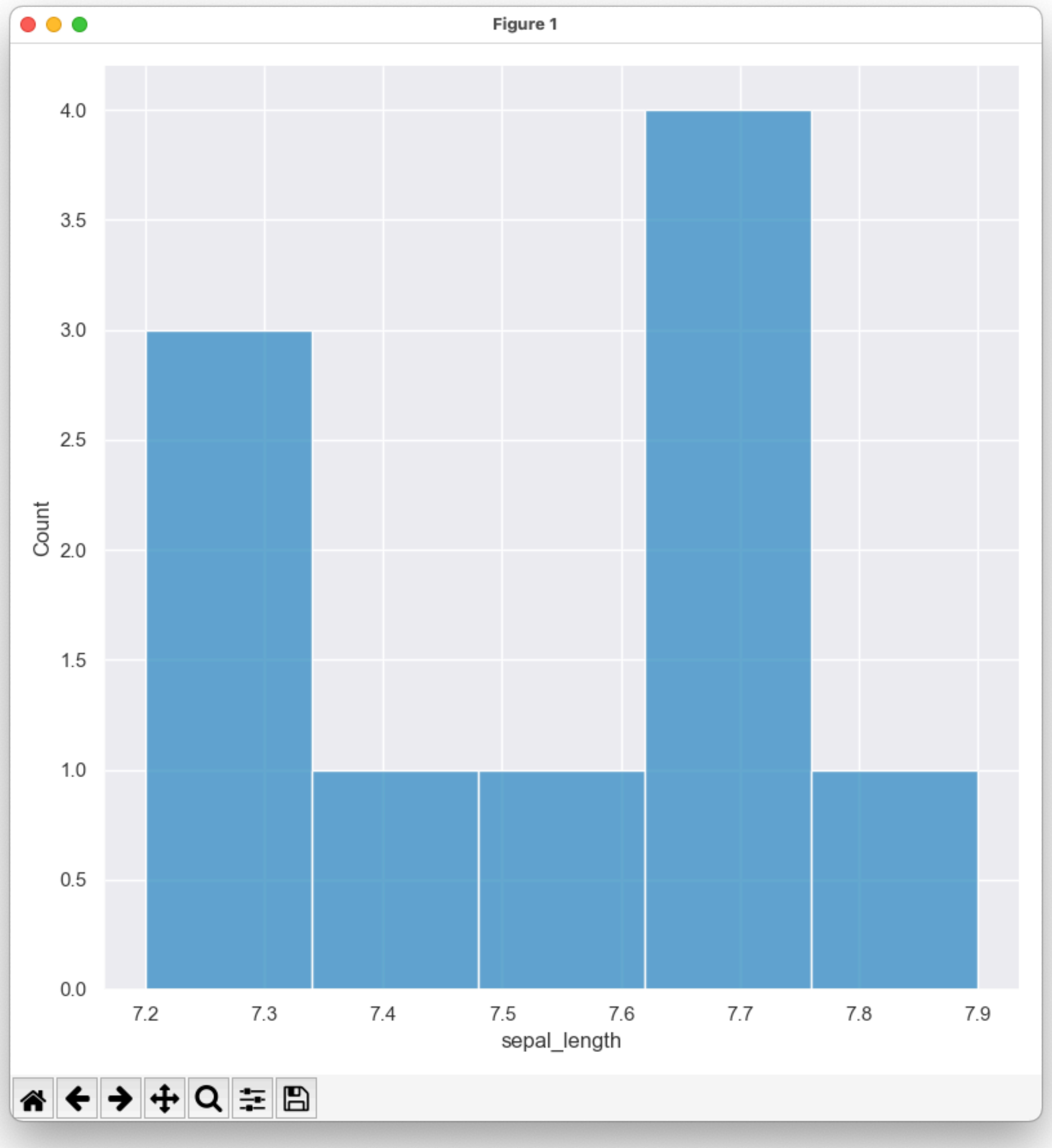


output is written to `hatch.sepal_length.histogram.png`

*nothing is printed on stdout because the last command is a plot

Data transformation and interactive plotting example

```
cat iris.csv | hatch sort -c sepal_length + tail 10 + histogram -x sepal_length --show
```



An interactive window displays the current plot, no file is generated.

PCA example

```
cat iris.csv | hatch pca + pretty
 sepal_length  sepal_width  petal_length  petal_width    species       pc1       pc2
          5.1          3.5           1.4          0.2     setosa -2.264542  0.505704
          4.9          3.0           1.4          0.2     setosa -2.086426 -0.655405
          4.7          3.2           1.3          0.2     setosa -2.367950 -0.318477
          4.6          3.1           1.5          0.2     setosa -2.304197 -0.575368
          5.0          3.6           1.4          0.2     setosa -2.388777  0.674767
...                    ...           ...          ...        ...       ...       ...
          6.7          3.0           5.2          2.3 virginica   1.870522  0.382822
          6.3          2.5           5.0          1.9 virginica   1.558492 -0.905314
          6.5          3.0           5.2          2.0 virginica   1.520845  0.266795
          6.2          3.4           5.4          2.3 virginica   1.376391  1.016362
          5.9          3.0           5.1          1.8 virginica   0.959299 -0.022284

[150 rows x 7 columns]
```

PCA example

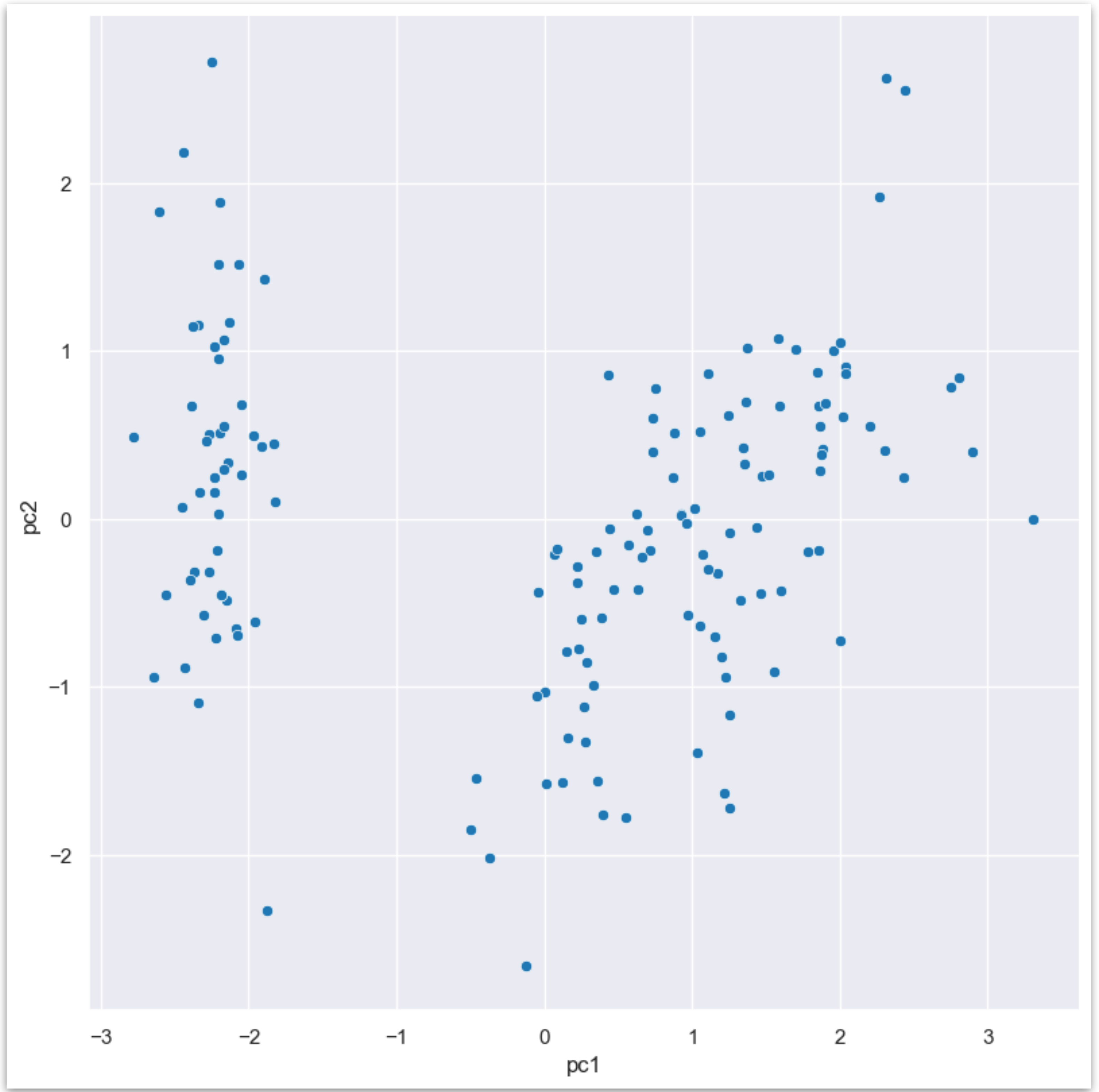
```
cat iris.csv | hatch pca + pretty
 sepal_length  sepal_width  petal_length  petal_width    species       pc1       pc2
                                                  0.2     setosa -2.264542  0.505704
                                                  0.2     setosa -2.086426 -0.655405
                                                  0.2     setosa -2.367950 -0.318477
                                                  0.2     setosa -2.304197 -0.575368
          5.0          3.6           1.4          0.2     setosa -2.388777  0.674767
...                    ...           ...          ...        ...       ...       ...
          6.7          3.0           5.2          2.3  virginica  1.870522  0.382822
          6.3          2.5           5.0          1.9  virginica  1.558492 -0.905314
          6.5          3.0           5.2          2.0  virginica  1.520845  0.266795
          6.2          3.4           5.4          2.3  virginica  1.376391  1.016362
          5.9          3.0           5.1          1.8  virginica  0.959299 -0.022284

[150 rows x 7 columns]
```

`pca` defaults to 2 components, but you can specify more with the `-n` argument. Unless otherwise specified all numerical features are taken as inputs.

in this case 2 extra columns (`pc1`, `pc2`) are added to the data

PCA example with scatter plot

```
cat iris.csv | hatch pca + scatter -x pc1 -y pc2
```



output is written to
`hatch.pc1.pc2.scatter.png`

note that the scatter command refers to the new `pc1` and `pc2` columns

PCA example with k-means clustering

```
cat iris.csv | hatch pca + kmeans -n 2 -c pc1 pc2 + pretty
 sepal_length  sepal_width  petal_length  petal_width    species       pc1       pc2  cluster
          5.1          3.5           1.4          0.2     setosa -2.264542  0.505704        0
          4.9          3.0           1.4          0.2     setosa -2.086426 -0.655405        0
          4.7          3.2           1.3          0.2     setosa -2.367950 -0.318477        0
          4.6          3.1           1.5          0.2     setosa -2.304197 -0.575368        0
          5.0          3.6           1.4          0.2     setosa -2.388777  0.674767        0
...                    ...           ...          ...        ...       ...       ...      ...
          6.7          3.0           5.2          2.3  virginica  1.870522  0.382822        1
          6.3          2.5           5.0          1.9  virginica  1.558492 -0.905314        1
          6.5          3.0           5.2          2.0  virginica  1.520845  0.266795        1
          6.2          3.4           5.4          2.3  virginica  1.376391  1.016362        1
          5.9          3.0           5.1          1.8  virginica  0.959299 -0.022284        1

[150 rows x 8 columns]
```

PCA example with k-means clustering

```
cat iris.csv | hatch pca + kmeans -n 2 -c pc1 pc2 + pretty
```

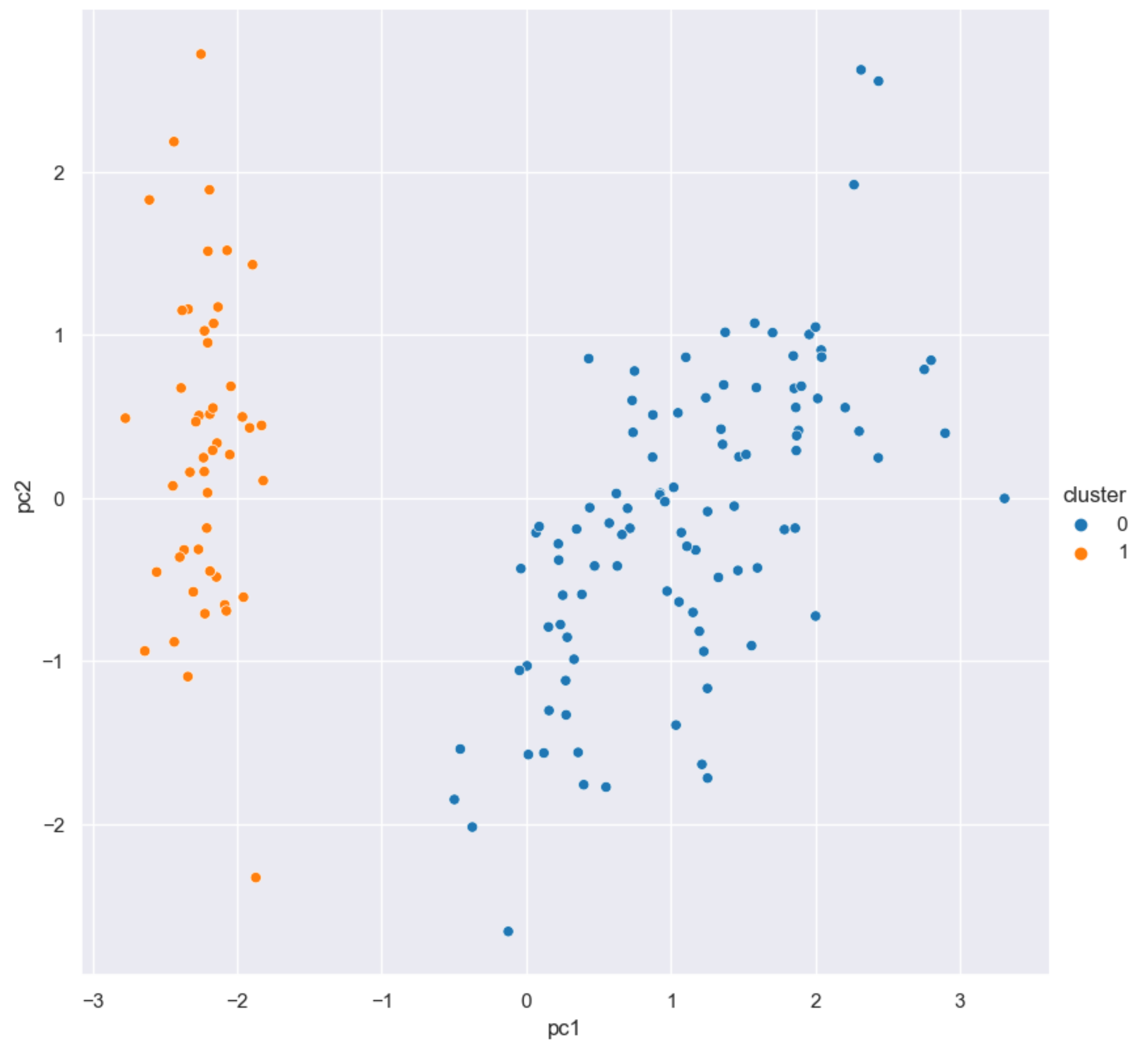
use kmeans to find 2 clusters in the data using only the columns pc1 and pc2

```
                                          width    species       pc1       pc2  cluster
                                            0.2     setosa -2.264542  0.505704        0
                                            0.2     setosa -2.086426 -0.655405        0
         4.7         3.2          1.3       0.2     setosa -2.367950 -0.318477        0
         4.6         3.1          1.5       0.2     setosa -2.304197 -0.575368        0
         5.0         3.6          1.4       0.2     setosa -2.388777  0.674767        0
...                  ...          ...       ...        ...       ...       ...      ...
         6.7         3.0          5.2       2.3  virginica  1.870522  0.382822        1
         6.3         2.5          5.0       1.9  virginica  1.558492 -0.905314        1
         6.5         3.0          5.2       2.0  virginica  1.520845  0.266795        1
         6.2         3.4          5.4       2.3  virginica  1.376391  1.016362        1
         5.9         3.0          5.1       1.8  virginica  0.959299 -0.022284        1

[150 rows x 8 columns]
```

1 extra column (cluster) is added to the data

PCA example with k-means clustering and scatter plot

```
cat iris.csv | hatch pca + kmeans -n 2 -c pc1 pc2 + scatter -x pc1 -y pc2 --hue cluster
```



output is written to `hatch.pc1.pc2.cluster.scatter.png`

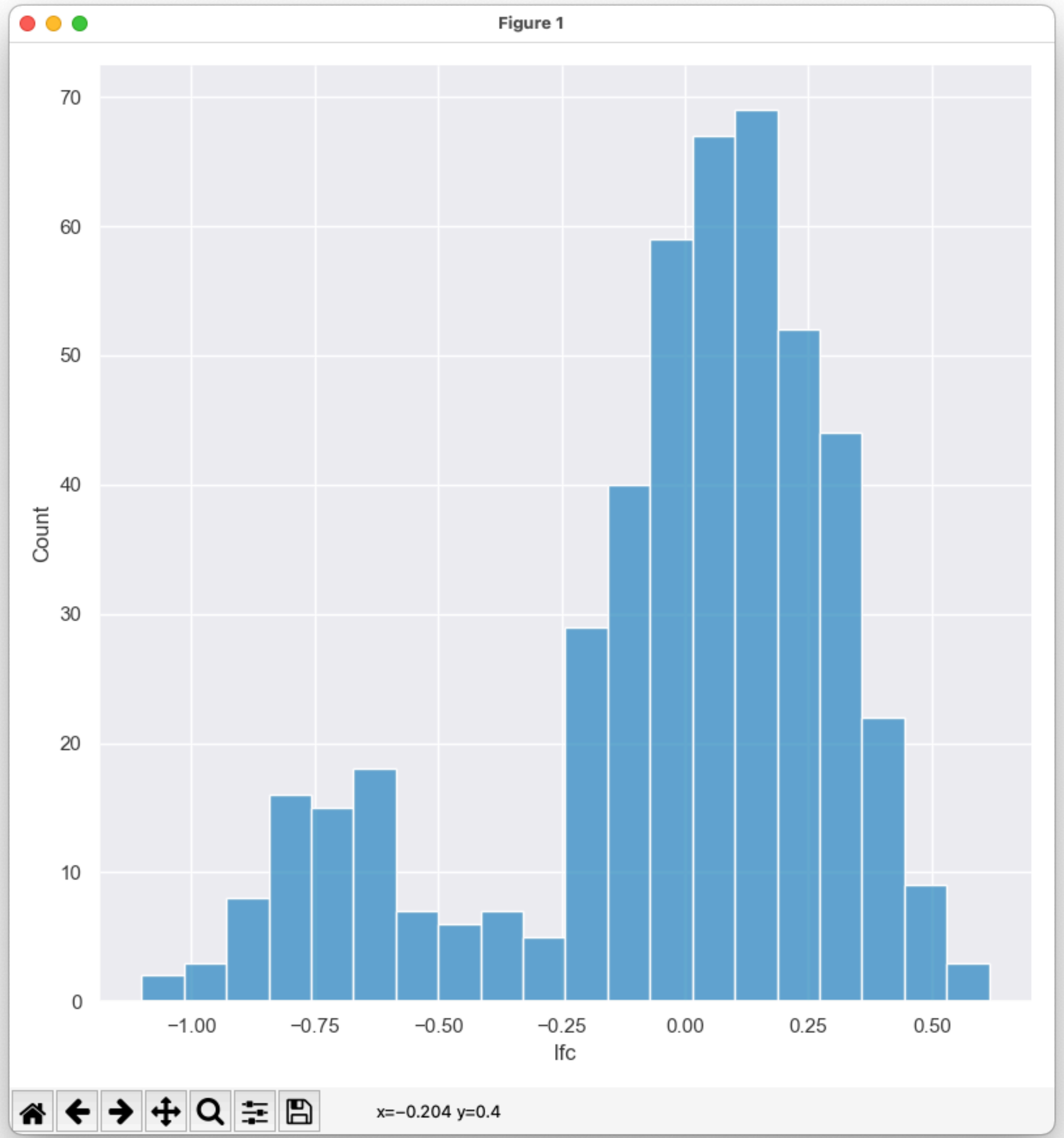
note that the scatter command refers to the new `cluster` column

Bioinformatics example: copy number analysis

```
cat TSO500.lfc.csv | hatch pretty -c chrom start end gene lfc gene_ordinal sample_code
chrom        start            end    gene        lfc  gene_ordinal sample_code
 chr1   8013207.0    8025060.0 ERRFI1   0.391513             0          A1
 chr1   9710453.0    9727048.0 PIK3CD -0.215299             1          A1
 chr1  11107482.0   11259411.0   MTOR -0.135017             2          A1
 chr1  15848065.0   15939429.0   SPEN   0.210534             3          A1
...          ...          ...    ...        ...           ...         ...
chr22  28687894.0   28741491.0  CHEK2 -0.385767           483          J9
chr22  29268006.0   29300527.0  EWSR1 -0.382720           484          J9
chr22  29603996.0   29694804.0    NF2 -0.109701           485          J9
chr22  37973492.0   37983786.0  SOX10 -0.922735           486          J9
chr22  41093002.0   41178958.0  EP300   0.180778           487          J9

[24023 rows x 27 columns]
```

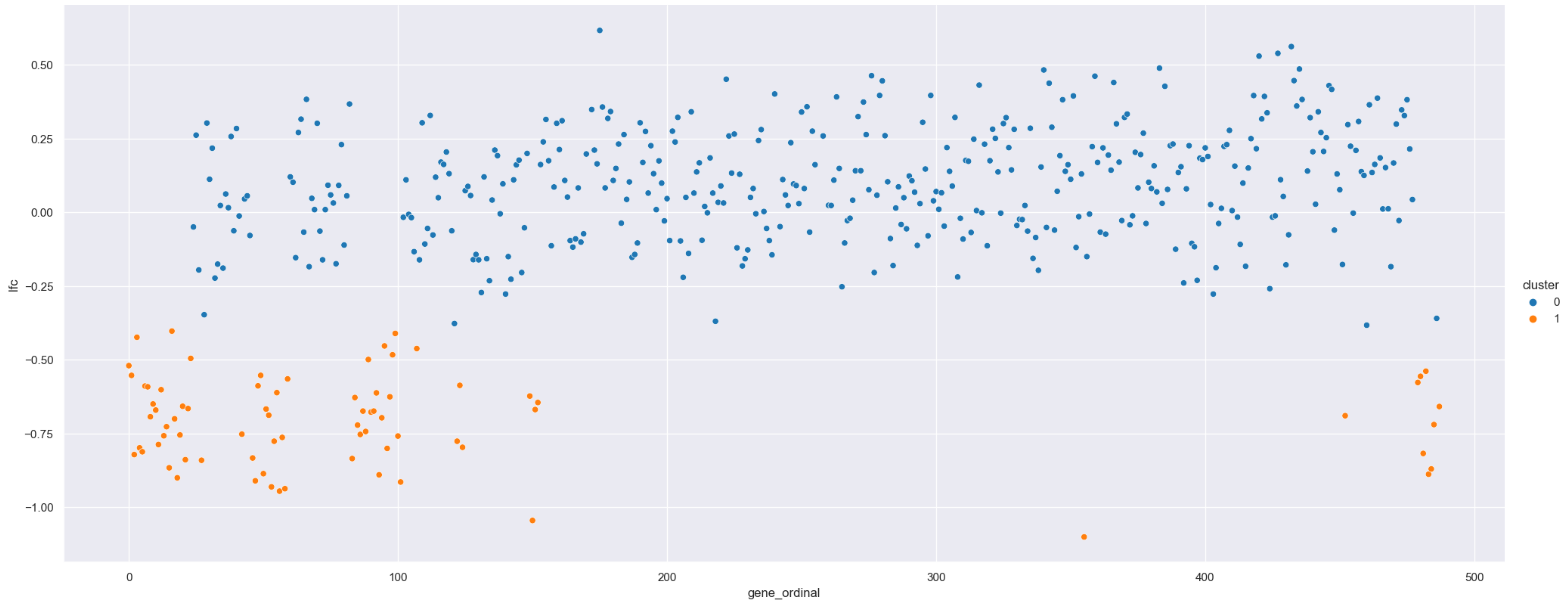
Bioinformatics example: copy number analysis

```
cat TSO500.lfc.csv | hatch filter 'sample_code == "B1"' + histogram -x lfc --show
```



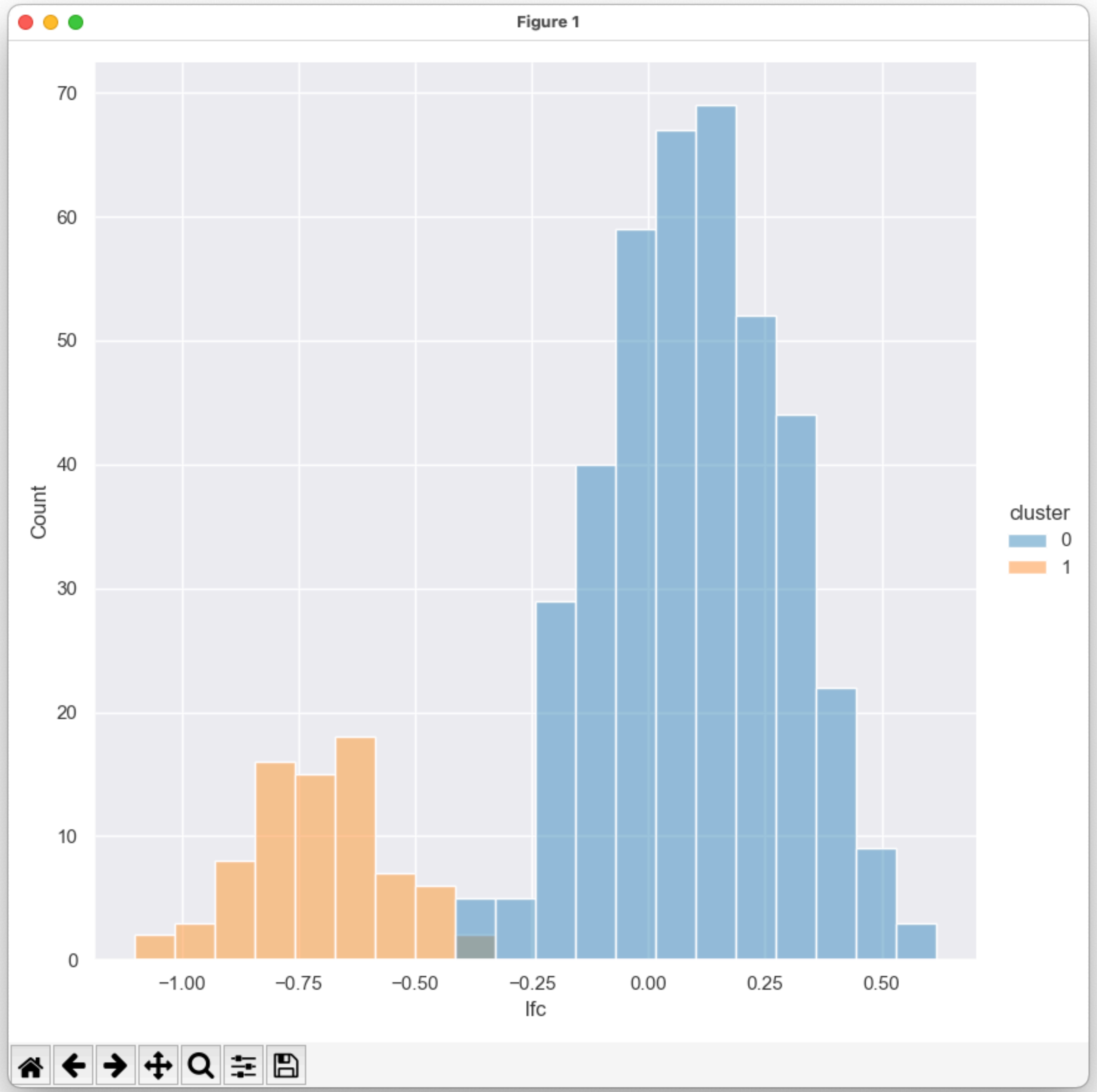
Bioinformatics example: copy number analysis

```
cat TSO500.lfc.csv | hatch filter 'sample_code == "B1"' + \
                           gmm -n 2 -c lfc + \
                           scatter -x gene_ordinal -y lfc --hue cluster --width 20
```



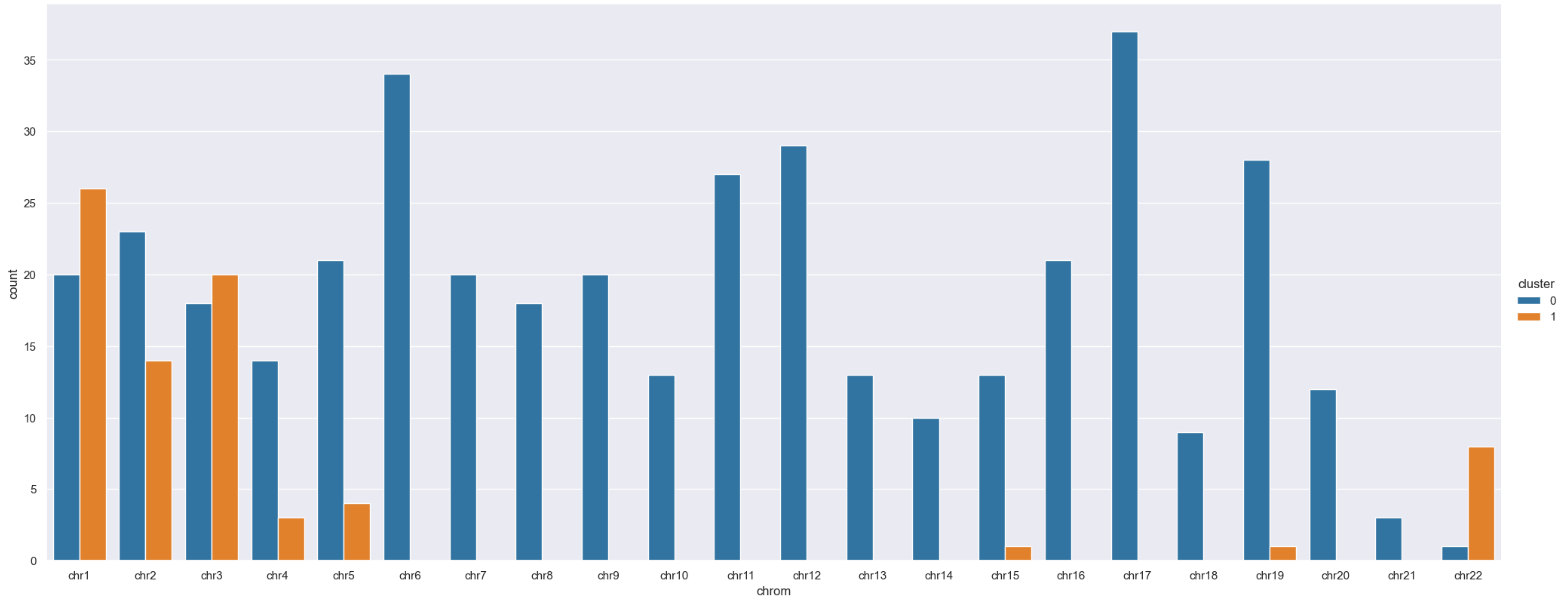
Bioinformatics example: copy number analysis

```
cat TSO500.lfc.csv | hatch filter 'sample_code == "B1"' + gmm -n 2 -c lfc + histogram -x lfc --hue cluster --show
```



Bioinformatics example: copy number analysis

```
cat TSO500.lfc.csv | hatch filter 'sample_code == "B1"' + \
                         gmm -n 2 -c lfc + \
                         count -x chrom --hue cluster --width 20
```



Code and documentation

- code:

  github.com/bjpop/hatch

- documentation:

  bjpop.github.io/hatch/

work in progress, actively being updated

Installation

```
git clone https://github.com/bjpop/hatch

python3 -m venv hatch_dev

source hatch_dev/bin/activate

pip install -U /path/to/hatch
```

Python 3.6 or greater is required.

PyPI and conda *etc* packages will soon be available.