

Getting started with csvkit

DATA PROCESSING IN SHELL



Susan Sun
Data Person

What is csvkit?

`csvkit` :

- is a suite of command-line tools
- is developed in Python by Wireservice
- offers data processing and cleaning capabilities on CSV files
- has data capabilities that rival Python, R, and SQL
- documentation: <https://csvkit.readthedocs.io/en/latest/>

csvkit installation

Install `csvkit` using Python package manager `pip` :

```
pip install csvkit
```

Upgrade `csvkit` to the latest version:

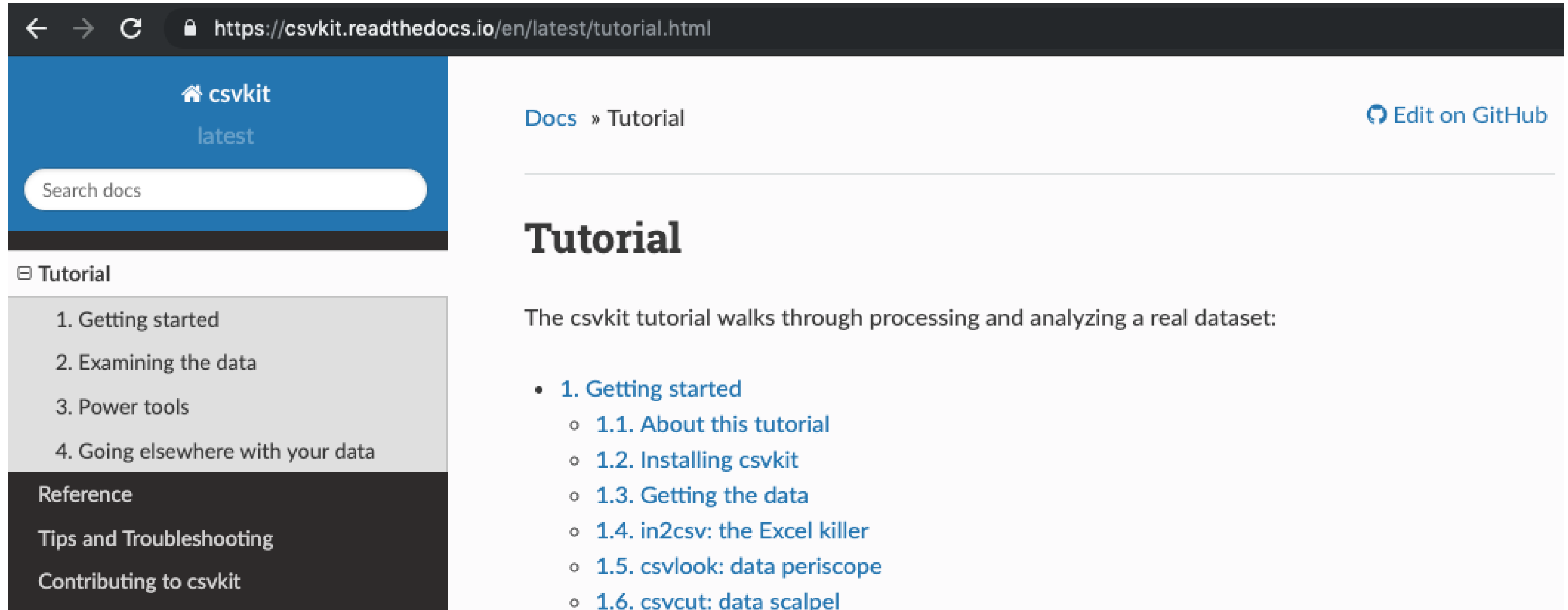
```
pip install --upgrade csvkit
```

Full instructions:

<https://csvkit.readthedocs.io/en/latest/tutorial.html>.

Browsing the csvkit manual

Web-based documentation: <https://csvkit.readthedocs.io/en/latest/tutorial.html>



The screenshot shows a web browser window with the URL <https://csvkit.readthedocs.io/en/latest/tutorial.html>. The page has a blue header with the 'csvkit latest' logo and a search bar. A left sidebar contains a table of contents with 'Tutorial' expanded, showing sections 1 through 4. The main content area has a breadcrumb 'Docs » Tutorial', a link to 'Edit on GitHub', and a large 'Tutorial' heading. Below the heading, it states 'The csvkit tutorial walks through processing and analyzing a real dataset:' followed by a bulleted list of sections: '1. Getting started' (which is expanded to show sub-sections 1.1 through 1.6), '2. Examining the data', '3. Power tools', and '4. Going elsewhere with your data'.

← → ↻ 🔒 <https://csvkit.readthedocs.io/en/latest/tutorial.html>

🏠 csvkit
latest

Search docs

☐ Tutorial

- 1. Getting started
- 2. Examining the data
- 3. Power tools
- 4. Going elsewhere with your data

Reference

Tips and Troubleshooting

Contributing to csvkit

Docs » Tutorial [Edit on GitHub](#)

Tutorial

The csvkit tutorial walks through processing and analyzing a real dataset:

- 1. Getting started
 - 1.1. About this tutorial
 - 1.2. Installing csvkit
 - 1.3. Getting the data
 - 1.4. in2csv: the Excel killer
 - 1.5. csvlook: data periscope
 - 1.6. csvcut: data scalpel

in2csv: converting files to CSV

Web-based documentation:

<https://csvkit.readthedocs.io/en/latest/scripts/in2csv.html>

Command line-based documentation:

```
in2csv --help
in2csv -h
```

```
usage: in2csv [-h] [-d DELIMITER] [-t] [-q QUOTECHAR] [-u {0,1,2,3}] [-b]
             [-p ESCAPECHAR] [-z FIELD_SIZE_LIMIT] [-e ENCODING] [-L LOCALE]
             [-S] [--blanks] [--date-format DATE_FORMAT]
             [--datetime-format DATETIME_FORMAT] [-H] [-K SKIP_LINES] [-v]
```

in2csv: converting files to CSV

Syntax:

```
in2csv SpotifyData.xlsx > SpotifyData.csv
```

Prints the first sheet in Excel to console and does not save

```
in2csv SpotifyData.xlsx
```

> redirects the output and saves it as a new file `SpotifyData.csv`

```
> SpotifyData.csv
```

in2csv: converting files to CSV

Use `--names` or `-n` option to print all sheet names in `SpotifyData.xlsx`.

```
in2csv -n SpotifyData.xlsx
```

```
Worksheet1_Popularity  
Worksheet2_MusicAttributes
```

Use `--sheet` option followed by the sheet `"Worksheet1_Popularity"` to be converted.

```
in2csv SpotifyData.xlsx --sheet "Worksheet1_Popularity" > Spotify_Popularity.csv
```

in2csv: converting files to CSV

`in2csv` does not print logs to console.

```
in2csv SpotifyData.xlsx --sheet "Worksheet1_Popularity" > Spotify_Popularity.csv
```

Sanity check:

```
ls
```

```
SpotifyData.xlsx  Spotify_Popularity.csv  backup  bin
```


csvlook: data preview on the command line

`csvlook` : renders a CSV to the command line in a Markdown-compatible, fixed-width format

Documentation:

```
csvlook -h
```

```
usage: csvlook [-h] [-d DELIMITER] [-t] [-q QUOTECHAR] [-u {0,1,2,3}] [-b]
              [-p ESCAPECHAR] [-z FIELD_SIZE_LIMIT] [-e ENCODING] [-L LOCALE]
              [-S] [--blanks] [--date-format DATE_FORMAT]
```

csvlook: data preview on the command line

Syntax:

```
csvlook Spotify_Popularity.csv
```

```
| track_id          | popularity |
| -----|
| 118GQ70Sp6pMqn6w1oKuki | 7 |
| 6S7cr72a7a8RVAXzDCRj6m | 7 |
| 7h2qWpMJzIVtiP30E8VDW4 | 7 |
| 3KVQFxJ5CW0cbxdpPYdi4o | 7 |
| 0JjNrI1xmsTfhaiU1R60Vc | 7 |
| 3HjTcZt29JUHg5m60QhLMw | 7 |
```

csvstat: descriptive stats on CSV data files

`csvstat` : prints descriptive summary statistics on all columns in CSV (e.g. mean, median, unique values counts)

Documentation:

```
csvstat -h
```

```
usage: csvstat [-h] [-d DELIMITER] [-t] [-q QUOTECHAR] [-u {0,1,2,3}] [-b]
               [-p ESCAPECHAR] [-z FIELD_SIZE_LIMIT] [-e ENCODING] [-S] [-H]
               [-K SKIP_LINES] [-v] [-l] [--zero] [-V] [--csv] [-n]
```

csvstat: descriptive stats on CSV data files

Syntax:

```
csvstat Spotify_Popularity.csv
```

1. "track_id"

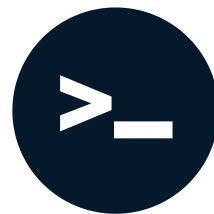
Type of data:	Text
Contains null values:	False
Unique values:	24
Longest value:	22 characters
Most common values:	118GQ70Sp6pMqn6w1oKuki (1x) 6S7cr72a7a8RVAXzDCRj6m (1x)

Let's try some csvkit!

DATA PROCESSING IN SHELL

Filtering data using csvkit

DATA PROCESSING IN SHELL



Susan Sun
Data Person

What does it mean to filter data?

We can create a subset of the original data file by:

1. Filtering the data by column
2. Filtering the data by row

csvcut : filters data using **column** name or position

csvgrep : filters data by **row** value through exact match, pattern matching, or even regex

csvcut: filtering data by column

`csvcut` : filters and truncates CSV files by **column name** or **column position**

Documentation:

```
csvcut -h
```

```
usage: csvcut [-h] [-d DELIMITER] [-t] [-q QUOTECHAR] [-u {0,1,2,3}] [-b]
              [-p ESCAPECHAR] [-z FIELD_SIZE_LIMIT] [-e ENCODING] [-S] [-H]
              [-K SKIP_LINES] [-v] [-l] [--zero] [-V] [-n] [-c COLUMNS]
```


csvcut: filtering data by column

Use `--names` or `-n` option to print all column names in `Spotify_MusicAttributes.csv`.

```
csvcut -n Spotify_MusicAttributes.csv
```

```
1: track_id  
2: danceability  
3: duration_ms
```

csvcut: filtering data by column

```
1: track_id  
2: danceability  
3: duration_ms
```

Returns the first column in the data, by **position**:

```
csvcut -c 1 Spotify_MusicAttributes.csv
```

```
track_id  
118GQ70Sp6pMqn6w1oKuki  
6S7cr72a7a8RVAXzDCRj6m
```

csvcut: filtering data by column

```
1: track_id  
2: danceability  
3: duration_ms
```

Returns only the first column in the data, by **name**:

```
csvcut -c "track_id" Spotify_MusicAttributes.csv
```

```
track_id  
118GQ70Sp6pMqn6w1oKuki  
6S7cr72a7a8RVAXzDCRj6m
```

csvcut: filtering data by column

```
1: track_id  
2: danceability  
3: duration_ms
```

Returns the second and third column in the data, by **position**:

```
csvcut -c 2,3 Spotify_MusicAttributes.csv
```

```
danceability,duration_ms  
0.787,124016.0  
0.777,128016.0  
0.7959999999999999,132742.0
```

csvcut: filtering data by column

```
1: track_id  
2: danceability  
3: duration_ms
```

Returns the second and third column in the data, by **name**:

```
csvcut -c "danceability","duration_ms" Spotify_MusicAttributes.csv
```

```
danceability,duration_ms  
0.787,124016.0  
0.777,128016.0  
0.7959999999999999,132742.0
```

csvgrep: filtering data by row value

csvgrep :

- filters by **row** using exact match or regex fuzzy matching
- must be paired with one of these options:

-m : followed by the exact row value to filter

-r : followed with a regex pattern

-f : followed by the path to a file

Documentation:

```
csvgrep -h
```

csvgrep: filtering data by row value

Find in `Spotify_Popularity.csv` where `track_id` = `5RCPsfzmEpTXMCTNk7wEfQ`

```
csvgrep -c "track_id" -m 5RCPsfzmEpTXMCTNk7wEfQ Spotify_Popularity.csv
```

```
track_id,popularity  
5RCPsfzmEpTXMCTNk7wEfQ,7.0
```

```
csvgrep -c 1 -m 5RCPsfzmEpTXMCTNk7wEfQ Spotify_Popularity.csv
```

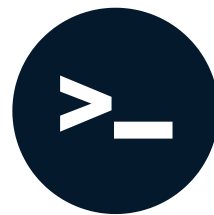
```
track_id,popularity  
5RCPsfzmEpTXMCTNk7wEfQ,7.0
```

Let's do data filtering with csvkit!

DATA PROCESSING IN SHELL

Stacking data and chaining commands with csvkit

DATA PROCESSING IN SHELL



Susan Sun
Data Person

csvstack: stacking multiple CSV files

`csvstack` : stacks up the rows from two or more CSV files

Documentation:

```
csvstack -h
```

```
usage: csvstack [-h] [-d DELIMITER] [-t] [-q QUOTECHAR] [-u {0,1,2,3}] [-b]
               [-p ESCAPECHAR] [-z FIELD_SIZE_LIMIT] [-e ENCODING] [-S] [-H]
               [-n GROUP_NAME] [--filenames]
```

csvstack: stacking multiple CSV files

Stack two similar files `Spotify_Rank6.csv` and `Spotify_Rank7.csv` into one file.

Preview the data to check schema:

```
csvlook Spotify_Rank6.csv
```

```
| track_id                | popularity |
| -----|
| 7JYCpIzpoIdD0nnmxmHwtj |          6 |
| 0mmFibEg5NuULMwTVN2tRU |          6 |
```

csvstack: stacking multiple CSV files

```
csvlook Spotify_Rank7.csv
```

track_id	popularity
-----	-----
118GQ70Sp6pMqn6w1oKuki	7
6S7cr72a7a8RVAXzDCRj6m	7

csvstack: stacking multiple CSV files

Syntax:

```
csvstack Spotify_Rank6.csv Spotify_Rank7.csv > Spotify_AllRanks.csv
```

```
csvlook Spotify_AllRanks.csv
```

track_id	popularity
-----	-----
7JYCpIzpoIdD0nnmxmHwtj	6
0mmFibEg5NuULMwTVN2tRU	6
118GQ70Sp6pMqn6w1oKuki	7
6S7cr72a7a8RVAXzDCRj6m	7

csvstack: stacking multiple CSV files

```
csvstack -g "Rank6","Rank7" \  
Spotify_Rank6.csv Spotify_Rank7.csv > Spotify_AllRanks.csv
```

```
csvlook Spotify_AllRanks.csv
```

group	track_id	popularity
-----	-----	-----
Rank6	7JYCpIzpoIdD0nnmxmHwtj	6
Rank6	0mmFibEg5NuULMwTVN2tRU	6
Rank7	118GQ70Sp6pMqn6w1oKuki	7
Rank7	6S7cr72a7a8RVAXzDCRj6m	7

csvstack: stacking multiple CSV files

```
csvstack -g "Rank6","Rank7" -n "source" \  
Spotify_Rank6.csv Spotify_Rank7.csv > Spotify_AllRanks.csv
```

```
csvlook Spotify_AllRanks.csv
```

source	track_id	popularity
-----	-----	-----
Rank6	7JYCpIzpoIdD0nnmxmHwtj	6
Rank6	0mmFibEg5NuULMwTVN2tRU	6
Rank7	118GQ70Sp6pMqn6w1oKuki	7
Rank7	6S7cr72a7a8RVAXzDCRj6m	7

chaining command-line commands

; links commands together and runs sequentially

```
csvlook SpotifyData_All.csv; csvstat SpotifyData_All.csv
```

&& links commands together, but only runs the 2nd command if the 1st succeeds

```
csvlook SpotifyData_All.csv && csvstat SpotifyData_All data.csv
```


chaining command-line commands

> re-directs the output from the 1st command to the location indicated as the 2nd

```
in2csv SpotifyData.xlsx > SpotifyData.csv
```

chaining command-line commands

| uses the output of the 1st command as input to the 2nd

Example:

Output of `csvcut` is not well formatted:

```
csvcut -c "track_id","danceability" Spotify_MusicAttributes.csv
```

```
track_id,danceability
118GQ70Sp6pMqn6w1oKuki,0.787
6S7cr72a7a8RVAXzDCRj6m,0.777
7h2qWpMJzIVtiP30E8VDW4,0.795
3KVQFxJ5CW0cbxdpPYdi4o,0.815
```

chaining command-line commands

Example (continued):

Re-format `csvcut` 's output by piping the output as input to `csvlook` :

```
csvcut -c "track_id","danceability" Spotify_Popularity.csv | csvlook
```

```
| track_id | danceability |
| ----- | ----- |
| 118GQ70Sp6pMqn6w1oKuki | 0.787 |
| 6S7cr72a7a8RVAXzDCRj6m | 0.777 |
| 7h2qWpMJzIVtiP30E8VDW4 | 0.796 |
| 3KVQFxJ5CW0cbxdpPYdi4o | 0.815 |
```

Let's put everything together!

DATA PROCESSING IN SHELL