csvmedkit Documentation

Release 0.0.9.13

Dan Nguyen

Nov 11, 2020

The utilities:

The unofficial extended family of csvkit, i.e. even more tools for command-line data parsing and wrangling

CHAPTER 1

csvflatten

csvflatten is a command for producing "flattened" records. Useful for quickly getting a view of records with numerous fields, and for documenting data examples in Markdown-compatible format.

For example, given a `data.csv` containing this:

```
id,product,price,description
001,apples,1.50,"An apple is an edible fruit produced by an apple tree (Malus␣
↪domestica)"
002,oranges,2.25,"An orange is a type of citrus fruit that people often eat. Oranges␣
↪are a very good source of vitamin C."
```

— **csvflatten** can be used to transform the data into a "narrow" 2-column denormalized format:

```
$ csvflatten data.csv
field,value
id,001
product,apples
price,1.50
description,An apple is an edible fruit produced by an apple tree (Malus domestica)
~~~~~~~~~~,
id,002
product,oranges
price,2.25
description,An orange is a type of citrus fruit that people often eat. Oranges are a␣
↪very good source of vitamin C.
```

TK jump to prettify section

Table of contents

1.1 Usage reference

1.1.1 `-P, --prettify`

Print output in Markdown tabular format instead of CSV

TK

A common use case is to produce "pretty" output — e.g. Markdown-table format — which can be easier to skim than a spreadsheet, especially for very *wide* data:

```
$ csvflatten -P data.csv
| field       | value                                                 |
| ----------- | ----------------------------------------------------- |
| id          | 001                                                   |
| product     | apples                                                |
| price       | 1.50                                                  |
| description | An apple is an edible fruit produced by an apple tree |
|             | (Malus domestica)                                     |
| ~~~~~~~~~~ |                                                       |
| id          | 002                                                   |
| product     | oranges                                               |
| price       | 2.25                                                  |
```

(continues on next page)

(continued from previous page)

```
| description | An orange is a type of citrus fruit that people often |
|             | eat. Oranges are a very good source of vitamin C.     |
```

1.1.2 -L, --max-length <max_length_of_field>

TK Split up values longer than [max_field_length] into multiple row-values as needed.

1.1.3 -R, --rec-id

TK Include a *_recid_* column for each row, for easier tracking the 0-based index of each record

1.1.4 -B, --label-chunks

When a long value is split into multiple "chunks", the *field* (i.e. first column) is left blank after the first chunk.

Setting the –chunk-labels flag will fill the *field* column with: "field~n", where *n* indicates the n-th chunk of a chopped value

1.1.5 -E, --eor <end_of_record_divider>

TK end of record; When flattening multiple records, separate each records with a row w/ fieldname of [marker]. Set to '' or 'none' to disable.

By default, the EOR marker is a series of tildes (~~~~~).

Note: this setting defaults to 'none' if *-R/–rowid* flag is used

1.2 High level overview

For every input record, **csvflatten**'s output will contain 2-column rows — `field,value` — for each of the record's key-value pairs. This is useful for viewing records one at a time, especially if each row contains many columns.

It's a concept similar to xsv flatten, though the structure of "flattened" output differs.

For example, given the following table:

id	product	price
001	apples	1.50
002	oranges	2.25

The "flattened" view of its 2 records would look like this:

field	value
id	001
product	apples
price	1.50
id	002
product	oranges
price	2.25

1.2.1 Basic example

Given a data file, *ids.csv*:

TK given *hamlet.csv*: hey

By default, **`csvflatten`** produces CSV output without word-wrapping long fields (such as the `lines` field in the example below). But the most common use-case is to produce *pretty* tabular output, including word-wrapping long fields to the width of the terminal. This is done using the `-P/--prettify` flag:

```
$ csvflatten examples/hamlet.csv -P

| fieldname | value                                          |
| --------- | ---------------------------------------------- |
| act       | 1                                              |
| scene     | 5                                              |
| speaker   | Horatio                                        |
| lines     | Propose the oath, my lord.                     |
| ~~~~~~~~~ |                                                |
| act       | 1                                              |
| scene     | 5                                              |
| speaker   | Hamlet                                         |
| lines     | Never to speak of this that you have seen,     |
|           | Swear by my sword.                             |
| ~~~~~~~~~ |                                                |
| act       | 1                                              |
| scene     | 5                                              |
| speaker   | Ghost                                          |
| lines     | [Beneath] Swear.                               |
| ~~~~~~~~~ |                                                |
| act       | 3                                              |
| scene     | 4                                              |
| speaker   | Gertrude                                       |
| lines     | O, speak to me no more;                        |
|           | These words, like daggers, enter in mine ears; |
|           | No more, sweet Hamlet!                         |
| ~~~~~~~~~ |                                                |
| act       | 4                                              |
| scene     | 7                                              |
| speaker   | Laertes                                        |
| lines     | Know you the hand?                             |
```

This output is suitable for pasting into a Markdown file to produce a formatted HTML table:

TK mention how hamlet.csv has new lines

fieldname	value
act	1
scene	5
speaker	Horatio
lines	Propose the oath, my lord.
~~	
act	1
scene	5
speaker	Hamlet
lines	Never to speak of this that you have seen,
	Swear by my sword.
~~	
act	1
scene	5

```
act,scene,speaker,lines
1,5,Horatio,"Propose the oath, my lord."
1,5,Hamlet,"Never to speak of this that you have seen,
Swear by my sword."
1,5,Ghost,[Beneath] Swear.
3,4,Gertrude,"O, speak to me no more;
These words, like daggers, enter in mine ears;
No more, sweet Hamlet!"
4,7,Laertes,Know you the hand?
```

1.3 How it compares to existing tools

1.3.1 Compared to csvkit's"csvlook"

csvlook doesn't pretty-format multi-line fields, and can also result in very wide tables without `--max-column-width`:

```
$ csvlook examples/hamlet.csv --max-column-width 50

| act | scene | speaker  | lines                                              |
| --- | ----- | -------- | -------------------------------------------------- |
|   1 |     5 | Horatio  | Propose the oath, my lord.                         |
|   1 |     5 | Hamlet   | Never to speak of this that you have seen,
Swea... |
|   1 |     5 | Ghost    | [Beneath] Swear.                                   |
|   3 |     4 | Gertrude | O, speak to me no more;
These words, like dagge... |
|   4 |     7 | Laertes  | Know you the hand?                                 |
```

1.3.2 Compared to `xsv flatten`

xsv flatten does do auto-wrapping of long entries, but doesn't produce tableized output:

```
$ xsv flatten examples/hamlet.csv

act      1
scene    5
speaker  Horatio
lines    Propose the oath, my lord.
#
act      1
scene    5
speaker  Hamlet
lines    Never to speak of this that you have seen,
Swear by my sword.
#
act      1
scene    5
speaker  Ghost
lines    [Beneath] Swear.
#
act      3
```

(continues on next page)

(continued from previous page)

```
scene    4
speaker  Gertrude
lines    O, speak to me no more;
These words, like daggers, enter in mine ears;
No more, sweet Hamlet!
#
act      4
scene    7
speaker  Laertes
lines    Know you the hand?
```

1.3.3 Compared to `tabulate`

python-tabulate is a command-line tool for producing a variety of tabular outputs, including `rst`, `grid`, and `html` formats. However, it does not handle multi-line fields well. Nor does it natively handle the CSV format, e.g. double-quoted values that contain commas, hence, the use of csvkit's csvformat to change delimiters to `\t` in the example below:

```
$ csvformat -T examples/hamlet.csv | tabulate -f grid -1 -s '\t'

+-----------------------------------------------+---------+-----------+--------------
↪-------------------------------+
| act                                           |   scene | speaker   | lines       ␣
↪                               |
+===============================================+=========+===========+==============================
| 1                                             |       5 | Horatio   | Propose the␣
↪oath, my lord.                  |
+-----------------------------------------------+---------+-----------+--------------
↪-------------------------------+
| 1                                             |       5 | Hamlet    | "Never to␣
↪speak of this that you have seen, |
+-----------------------------------------------+---------+-----------+--------------
↪-------------------------------+
| Swear by my sword."                           |         |           |             ␣
↪                               |
+-----------------------------------------------+---------+-----------+--------------
↪-------------------------------+
| 1                                             |       5 | Ghost     | [Beneath]␣
↪Swear.                          |
+-----------------------------------------------+---------+-----------+--------------
↪-------------------------------+
| 3                                             |       4 | Gertrude  | "O, speak to␣
↪me no more;                     |
+-----------------------------------------------+---------+-----------+--------------
↪-------------------------------+
| These words, like daggers, enter in mine ears; |         |           |             ␣
↪                               |
+-----------------------------------------------+---------+-----------+--------------
↪-------------------------------+
| No more, sweet Hamlet!"                       |         |           |             ␣
↪                               |
+-----------------------------------------------+---------+-----------+--------------
↪-------------------------------+
| 4                                             |       7 | Laertes   | Know you the␣
↪hand?                           |
```

(continues on next page)

(continued from previous page)

```
+-----------------------------------------------+---------+-----------+--------------
↪-------------------------------+
```

That said, if you like `tabulate`'s table-formatting options, such as `-f grid`, you can pipe **`csvflatten`** (and **`csvformat`** to convert to tab-delimiters) into `tabulate` like so:

```
$ csvflatten --eor 'none' examples/hamlet.csv | csvformat -T \
    tabulate -f grid -1 -s '\t'

+---------+-------------------------------------------------+
| field   | value                                           |
+=========+=================================================+
| act     | 1                                               |
+---------+-------------------------------------------------+
| scene   | 5                                               |
+---------+-------------------------------------------------+
| speaker | Horatio                                         |
+---------+-------------------------------------------------+
| lines   | Propose the oath, my lord.                      |
+---------+-------------------------------------------------+
| act     | 1                                               |
+---------+-------------------------------------------------+
| scene   | 5                                               |
+---------+-------------------------------------------------+
| speaker | Hamlet                                          |
+---------+-------------------------------------------------+
| lines   | Never to speak of this that you have seen,      |
+---------+-------------------------------------------------+
|         | Swear by my sword.                              |
+---------+-------------------------------------------------+
| act     | 1                                               |
+---------+-------------------------------------------------+
| scene   | 5                                               |
+---------+-------------------------------------------------+
| speaker | Ghost                                           |
+---------+-------------------------------------------------+
| lines   | [Beneath] Swear.                                |
+---------+-------------------------------------------------+
| act     | 3                                               |
+---------+-------------------------------------------------+
| scene   | 4                                               |
+---------+-------------------------------------------------+
| speaker | Gertrude                                        |
+---------+-------------------------------------------------+
| lines   | O, speak to me no more;                         |
+---------+-------------------------------------------------+
|         | These words, like daggers, enter in mine ears;  |
+---------+-------------------------------------------------+
|         | No more, sweet Hamlet!                          |
+---------+-------------------------------------------------+
| act     | 4                                               |
+---------+-------------------------------------------------+
| scene   | 7                                               |
+---------+-------------------------------------------------+
| speaker | Laertes                                         |
+---------+-------------------------------------------------+
```

(continues on next page)

(continued from previous page)

```
| lines   | Know you the hand?                              |
+---------+-------------------------------------------------+
```

1.4 Reference: Options and usage

1.4.1 -P/–prettify

Print output in tabular format instead of CSV. Unless `-L/--max-length` is explicitly specified, long values are split into multiple rows based on the current terminal width.

1.4.2 -L/–max-length [integer]

Specify a max character length for field values; values that exceed this length are split into multiple rows. This is useful for producing output easier to view in a spreadsheet:

```
$ csvflatten -L 5 examples/abc123.csv

field,value
code,alfa
blob,01234
,56789
~~~~~,
code,beta
blob,ABCDE
,FGHIJ
```

Or combining with the `-P/--prettify` option:

```
$ csvflatten -P -L 5 examples/abc123.csv

| field | value |
| ----- | ----- |
| code  | alfa  |
| blob  | 01234 |
|       | 56789 |
| ~~~~~ |       |
| code  | beta  |
| blob  | ABCDE |
|       | FGHIJ |
```

The default behavior is as follows:

- *Without* `--prettify` mode, this value is set to `0`, i.e. no splitting of long values.
- *With* `--prettify` mode, this value is automatically set to the width of the terminal. To disable this behavior, you can explicitly set `--max-length 0`

1.4.3 -B/–chunk-labels

When a value is chopped into chunks across multiple rows, by default, the `field` (i.e. first column) is filled in for the value's *first* row, then left blank for its subsequent rows:

```
| field |  value  |
+-------+---------+
| id    | 001     |
| title | this is |
|       | a story |
|       | of love |
```

If the `--chunk-labels` flag is set, each subsequent `field` will be filled with an incremental label, e.g.:

```
|  field   |  value  |
+----------+---------+
| id       | 001     |
| title    | this is |
| title__1 | a story |
| title__2 | of love |
```

1.4.4 -E/–eor [END_OF_RECORD_MARKER (string)]

By default, each record is separated by having a string of *tildes* in `field`, e.g.:

```
| field |  value  |
+-------+---------+
| id    | 001     |
| title | this is |
|       | a story |
|       | of love |
| ~~~~~ |         |
| id    | 002     |
| title | Book 2  |
```

Set `'none'` to disable:

```
| field |  value  |
+-------+---------+
| id    | 001     |
| title | this is |
|       | a story |
|       | of love |
| id    | 002     |
| title | Book 2  |
```

Or to a value of your choosing:

```
$ csvflatten -E 'NEW-RECORD' data.csv

|   field    |  value  |
+------------+---------+
| id         | 001     |
| title      | this is |
|            | a story |
|            | of love |
| NEW-RECORD |         |
| id         | 002     |
| title      | Book 2  |
```

1.4.5 -R/–rec-id

Include a `_recid_` column that contains the 0-based index for the respective record:

```
| _recid_ | field |  value   |
+---------+-------+----------+
|       0 | id    | 001      |
|       0 | title | this is  |
|       0 |       | a story  |
|       0 |       | of love  |
|       1 | id    | 002      |
|       1 | title | Book 2   |
```

Note that `--rec-id` by default disables the end-of-record separator

1.5 Common scenarios and use cases

1.5.1 Making multiline tweets easier to read

about the data TK

TK Raw data comes from Twitter's API and using twarc's flattened CSV version:

- https://github.com/DocNow/twarc#timeline

```
$ twarc timeline POTUS --format csv
```

Reference file: `tweets-newlines.csv`

TK: image of data in excel

```
$ csvflatten -P examples/tweets-newlines.csv
```

First two tweets, in flattened format:

```
.. code-block:: text
```

field | value |
——————————- | ——————————————————————— |
id | 1196869405629702148 |
tweet_url | https://twitter.com/POTUS/status/1196869405629702148 |
created_at | Tue Nov 19 19:14:37 +0000 2019 |
parsed_created_at | 2019-11-19 19:14:37+00:00 |
user_screen_name | POTUS |
text | Since the day President @realDonaldTrump took office, |
| House Democrats have ignored working Americans to |
| focus on their only real agenda item: impeachment. |
| Meanwhile, our country created $11 trillion in worth. |
| https://t.co/BwCBFMU2Tn |
tweet_type | retweet |
coordinates | |
hashtags | |

media | |
urls | |
favorite_count | 40704 |
in_reply_to_screen_name | |
in_reply_to_status_id | |
in_reply_to_user_id | |
lang | en |
place | |
possibly_sensitive | |
retweet_count | 12547 |
retweet_or_quote_id | 1196864730226401280 |
retweet_or_quote_screen_name | WhiteHouse |
retweet_or_quote_user_id | 822215673812119553 |
source | <a href="https://www.sprinklr.com" |
| rel="nofollow">TheWhiteHouse |
user_id | 822215679726100480 |
user_created_at | Thu Jan 19 22:54:28 +0000 2017 |
user_default_profile_image | False |
user_description | 45th President of the United States of America, |
| @realDonaldTrump. Tweets archived: |
| https://t.co/eVVzoBb3Zr |
user_favourites_count | 104 |
user_followers_count | 32630758 |
user_friends_count | 39 |
user_listed_count | 29517 |
user_location | Washington, D.C. |
user_name | President Trump |
user_statuses_count | 10928 |
user_time_zone | |
user_urls | http://WhiteHouse.gov |
user_verified | True |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ | |
id | 1240016437248438276 |
tweet_url | https://twitter.com/POTUS/status/1240016437248438276 |
created_at | Tue Mar 17 20:45:31 +0000 2020 |
parsed_created_at | 2020-03-17 20:45:31+00:00 |
user_screen_name | POTUS |
text | It's time for all Americans to do their part to stop |
| the transmission of the Coronavirus. We will come |
| out stronger than ever before! https://t.co/u0SHHpRJ0l |
tweet_type | retweet |
coordinates | |
hashtags | |
media | |
urls | |
favorite_count | 15363 |
in_reply_to_screen_name | |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

in_reply_to_status_id | |
in_reply_to_user_id | |
lang | en |
place | |
possibly_sensitive | |
retweet_count | 3779 |
retweet_or_quote_id | 1240016322920091655 |
retweet_or_quote_screen_name | WhiteHouse |
retweet_or_quote_user_id | 822215673812119553 |
source | <a href="https://www.sprinklr.com" |
| rel="nofollow">TheWhiteHouse |
user_id | 822215679726100480 |
user_created_at | Thu Jan 19 22:54:28 +0000 2017 |
user_default_profile_image | False |
user_description | 45th President of the United States of America, |
| @realDonaldTrump. Tweets archived: |
| https://t.co/eVVzoBb3Zr |
user_favourites_count | 104 |
user_followers_count | 32630764 |
user_friends_count | 39 |
user_listed_count | 29517 |
user_location | Washington, D.C. |
user_name | President Trump |
user_statuses_count | 10928 |
user_time_zone | |
user_urls | http://WhiteHouse.gov |
user_verified | True |

CHAPTER 2

csvheader

csvheader is a command for listing and changing the headers of CSV-formatted data.

For example, given a `data.csv` containing this

```
Case #,X,Y,I.D.
1,2,3,4
5,6,7,8
```

You can *slugify* the headers:

```
$ csvheader -S data.csv
case,x,y,i_d
1,2,3,4
5,6,7,8
```

And/or selectively rename them:

```
$ csvheader -R '1|Case Num,4|ID,X|lat,Y|lng' data.csv
Case Num,lat,lng,ID
1,2,3,4
5,6,7,8
```

Table of contents

2.1 Options TK reference

2.1.1 -A, –add

Add a header row of generic, numbered column names, starting from 1, e.g. field_1, field_2, and so on.

2.1.2 -B, –bash

Bash (i.e. completely replace) the current header row with generic column names, e.g. field_1, field_2.

2.1.3 -C, –create <column_names>

Similar to `--add`, but specify column names with a comma-delimited string, e.g. `'ID,cost,"Name, proper"'`

2.1.4 -R, –rename <renamed_header_pairs>

Rename individual columns. The required argument is a comma-delimited string of pipe-delimited pairs — column id/name and the new name.

For example, to rename the “a” column to “Apples”; and also, the 2nd and 3rd columns to “hello” and “world”, respectively, the quoted argument string would be:

```
'a|Apples,2|hello,3|world'
```

2.1.5 -S, –slugify

Converts the existing column names to snake_case style. For example, `APPLES` and `'Date - Time '` are converted, respectively, to `'apples'` and `'date_time'`.

2.1.6 -X, –regex <pattern> <replacement>

In the existing column names, replace all occurrences of a regular expression <*pattern*> with <*replacement*>.

2.1.7 -P, –preview

When no options are invoked, only the existing header is printed as a comma-delimited list. Invoking any of the aforementioned options prints the transformed header *and* the data. In the latter case, use the `--preview` flag to see only what the transformed headers look like.

2.2 High level overview TK

In its most basic invocation, **`csvheader`** simply produces a list of column names in CSV format:

```
$ csvheader examples/heady.csv

index,field
1,A
2, B Sharps
3,"SEA, shells!"
```

However, enabling any of its renaming options, such as `--slug`, will reproduce the input data with its headers renamed:

```
$ csvheader --slugify examples/heady.csv

a,b_sharps,sea_shells
100,cats,Iowa
200,dogs,Ohio
```

Examples and stuff TK

2.3 How csvheader compares to existing tools TK

2.3.1 Compared to adding a header row with `csvformat --no-header-row`

```
$ echo '1,2,3,4' | csvformat --no-header-row
a,b,c,d
1,2,3,4
```

```
$ echo '1,2,3,4' | csvheader --add
field_1,field_2,field_3,field_4
1,2,3,4
```

Note: csvformat 1.0.6 bug

In the latest official release of csvkit — 1.0.6 — csvformat's `-H/--no-header-row` does not work as expected. See issue/pull request here. (TODO: update this if csvkit master is patched)

2.3.2 Compared to listing column names with `csvcut --names`

TK Lorem ipsum dolor sit amet, consectetur adipisicing elit

```
$ echo 'a,b, c ,d  ' | csvcut --names
  1: a
  2: b
  3:  c
  4: d
```

In contrast, because `csvheader` outputs the header as CSV, its output can be piped into, say, `csvformat`, which, if you want, *can* produce quoted values to make the whitespace more obvious:

```
$ echo 'a,b, c ,d  ' | csvheader | csvformat -U 1
"index","field"
"1","a"
"2","b"
"3"," c "
"4","d  "
```

2.3.3 Compared to listing column names with `xsv headers`

```
$ echo 'a,b, c ,d  ' | xsv headers
1   a
2   b
3    c
4   d
```

2.3.4 Compared to replacing the first line of data with `sed`

It's possible to use **`sed`** to replace the entire first line of input:

```
$ sed '1s/.*/alpha,bravo,charlie/' examples/heady.csv

alpha,bravo,charlie
100,cats,Iowa
200,dogs,Ohio
```

However, this invocation of **`sed`** will not work on multi-line headers (which is admittedly, an edge-case).

But `sed` can't be used to selectively rename headers — it can only do string replacement. For example, to rename *only* the 1st column requires tailoring a specific regex:

```
$ sed '1s/^A/alpha/' examples/heady.csv

alpha, B Sharps ,"SEA, shells!"
100,cats,Iowa
200,dogs,Ohio
```

Renaming only the 1st *and* 3rd columns gets very messy:

```
$ sed -e '1s/^A/alpha/' -e '1s/"SEA.*/charlie/' examples/heady.csv

alpha, B Sharps ,charlie
100,cats,Iowa
200,dogs,Ohio
```

In contrast, `csvheader --rename` allows for renaming columns by (1-based) index:

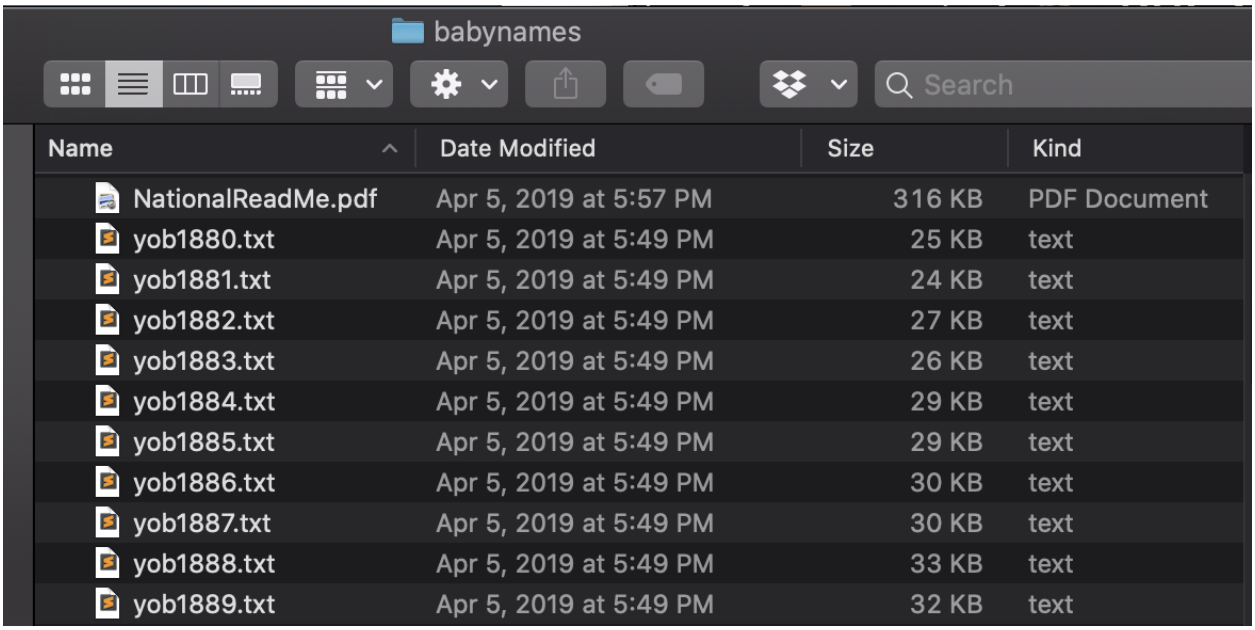
```
$ csvheader --rename '1|alpha,3|charlie' examples/heady.csv


alpha, B Sharps ,charlie
100,cats,Iowa
200,dogs,Ohio
```

2.4 Real-world use cases TK

2.4.1 Adding a header to the Social Security babynames data

The nationwide baby names data comes as a zip file of comma-delimited text files, one for each year, e.g. `yob1880.txt` and `yob2015.txt`:



Each file contains the same 3 columns — name, sex, and count — but *sans* header row. Here are the first 3 rows in `yob1880.txt`:

```
Mary,F,7065
Anna,F,2604
Emma,F,2003
```

Invoking `csvheader` with the `-G/--generic` flag will add generic column names to the data of each individual file, e.g.:

```
$ csvheader -G yob1880.txt
field_1,field_2,field_3
Mary,F,7065
Anna,F,2604
Emma,F,2003

...
```

But it's not much more work to add our own useful column names using the `-A/--add` option:

```
$ csvheader yob1880.txt -A 'name,sex,count'
name,sex,count
Mary,F,7065
Anna,F,2604
Emma,F,2003
```

Of course, doing that for every file would be extremely tedious. You should be using `csvstack` with the `-H/--no-header-row` option to collate all the files into a single file and header:

```
$ csvstack *.txt -H yob*.txt > babynames.csv
```

The more important reason to use `csvstack` is its `--filenames` option, which prepends a 'group' column to the data that contains the *filename* for each record:

```
$ csvstack *.txt -H --filenames yob.txt > babynames.csv
```

This is *absolutely* critical, because the rows in each `yob****.txt` file don't include the year of the data file — which makes the compiled `babynames.csv` completely useless.

However, with `csvstack --filenames`, that vital year context is included in the compiled `babynames.csv`:

```
group,a,b,c
yob1880.txt,Mary,F,7065
yob1880.txt,Anna,F,2604
yob1880.txt,Emma,F,2003
...
yob2018.txt,Zyrie,M,5
yob2018.txt,Zyron,M,5
yob2018.txt,Zzyzx,M,5
```

So combining that with `csvhheader --AX/--add-x`:

```
$ csvstack *.txt -H --filenames | csvheader --AX 'year,name,sex,count' > babynames.csv
```

— results in `babynames.csv` looking like:

```
year,name,sex,count
yob1880.txt,Mary,F,7065
yob1880.txt,Anna,F,2604
yob1880.txt,Emma,F,2003
...
yob2018.txt,Zyrie,M,5
yob2018.txt,Zyron,M,5
yob2018.txt,Zzyzx,M,5
```

TK make test

To see how to clean up the `year` column — e.g. change `'yob2018.txt'` to `'2018'`, see: *Using csvsed to clean up the SSA babynames data*

CHAPTER 3

csvslice

csvslice is a command for selecting rows by 0-based index and/or inclusive ranges.

Given a data file, *ids.csv*:

```
$ csvslice -i 0,2-3 ids.csv
```

The output:

```
source
id,val
a,0
c,2
d,3
```

Table of contents

3.1 Options and flags

csvslice has 3 unique options, `--indexes, --head, and --tail`. Each of these specify a *mode*, and one and only one mode can be specified.

3.1.1 -i, –indexes <values>

<*values*> is a comma-delimited list of values representing individual indexes or ranges of indexes to be sliced and included from the 0-indexed dataset.

An index value can take these forms:

- An individual row: `-i 0` returns the very first row
- A range of rows: `-i 2-4` returns rows in the inclusive index range of 2 through 4
- An open-ended range: `-i 3-` returns all rows starting from row index 3

Multiple interval values can be passed into `-i/--indexes`, e.g.

```
csvslice -i '0-5,12,15-18,42-' data.csv
```

3.1.2 –head <int>

Return the header and the *first* <*int*> rows; <*int*> must be greater than 0.

3.1.3 –tail <int>

Return the header and the *last* <*int*> rows; <*int*> must be greater than 0.

3.2 Usage overview and examples

These examples refer to data as found in `ids.csv`

3.2.1 Get the first n rows with –head

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolor

3.2.2 Get the last n rows with –tail

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

3.2.3 Slicing individual rows with –index

You can specify rows to be sliced by 0-based index:

```
csvslice -i 1 ids.csv
id,val
1,b
```

You can also specify a series of individual indexes as a comma-delimited string:

```
csvslice -i 0,5 ids.csv
id,val
0,a
5,f
```

3.2.4 Slicing rows by an index range

Rows can be specified by using a range syntax: `start-end`

The range is *inclusive*:

```
$ csvslice -i 1-3 ids.csv
id,val
1,b
2,c
3,d
```

Omitting the right-side *end* value returns an open range of values:

```
$ csvslice -i 3- ids.csv
id,val
3,d
4,e
5,f
```

Like indexes, a series of ranges can be specified as a comma-delimited string:

```
$ csvslice -i 0-1,3- ids.csv
id,val
0,a
1,b
3,d
4,e
5,f
```

And you can combine ranges with individual indexes:

```
$ csvslice -i 0,2-3,5 ids.csv
id,val
0,a
2,c
3,d
5,f
```

3.2.5 Troubleshooting

Even though `3-1` and is technically a valid range, **`csvslice`** will throw an error if the `end` value is smaller than the `start` value:

```
$ csvslice -i 3-1 examples/ids.csv
InvalidRange: Invalid range specified: 3-1
```

For the most part, though, **`csvslice`** isn't too whiny (in terms of warning you) about messy or otherwise nonsensical values for `-i/--indexes`.

No matter what order you specify the indexes and ranges, it will always return rows in sequential order:

```
$ csvslice -i 4,0,2 ids.csv
id,val
0,a
2,c
4,e
```

```
$ csvslice -i 4,0-2,3 ids.csv
id,val
0,a
1,b
2,c
3,d
4,e
```

If you pass in repeated indexes and/or overlapping ranges, **`csvslice`** will still only return the original, sequential data, i.e. it will *not* return duplicates of rows:

```
$ csvslice -i 3,1,3,1,1 ids.csv
id,val
1,b
3,d
```

```
$ csvslice -i 1,0-2,1-3 ids.csv
id,val
0,a
```

(continues on next page)

(continued from previous page)

```
1,b
2,c
3,d
```

If you pass in references to non-existent row indexes, such as out-of-bounds numbers — those too are ignored:

```
csvslice -i 5,42 ids.csv
id,val
5,f
```

3.3 How csvslice compares to existing tools

Given `ids.csv` (TK: not sure I need to repeat this reference?)

3.3.1 head: Get the first n rows

The issue with **`head`** is that it only understands text and newline characters. Thus, specifying `-n 3` returns 3 lines, one of which is the header line.

```
$ head -n 3 ids.csv
id,val
0,a
1,b
```

In contrast, **`csvslice`** has a notion of comma-delimited *data*, in which the first line is **not** data, but the header. Thus, an argument equivalent to `-n 3` would return *4 lines* in total: 1 header line and 3 data lines.

```
$ csvslice --head 3 ids.csv
id,val
0,a
1,b
2,c
```

3.3.2 tail: Get the last n rows

TK lorem with **`tail`**

```
$ tail -n 2 ids.csv
4,e
5,f
```

TK lorem with **`csvslice`**

```
$ csvslice --tail 2 ids.csv
id,val
4,e
5,f
```

3.3.3 csvformat: Skip the first n lines

```
$ TKTK ids.csv
```

3.3.4 xsv slice

csvslice is so much slower than xsv that the main reason to use csvslice for common functionality is that you just didn't get around to installing xsv. lorem ipsum

TODO: section on benchmarks - make this its own include file

xsv slice a single record

```
$ xsv slice -i 2 ids.csv
id,val
2,c
```

lorem ipsum

```
$ csvslice -i 2 ids.csv
id,val
2,c
```

However, performance is *much* slower! Lorem ipsum TK.

xsv slice the first n records

```
$ xsv slice -l 3 ids.csv
0,a
1,b
2,c
```

```
$ csvslice ^-head 3 ids.csv
```

Note that **`xsv slice`** has no `tail`-like functionality, i.e. returning the last *n* rows.

xsv slice a range of records

```
$ xsv slice -s 1 -e 3 ids.csv
id,val
1,b
2,c
```

lorem **`csvslice`** uses inclusive ranges

```
$ csvslice -i 1-2 ids.csv
id,val
1,b
2,c
```

3.3.5 The agate library

TK Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut

3.3.6 The pandas library

TK Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

3.4 Real-world use cases

3.4.1 Skipping the meta-header in Census data TK

About the data

You can view the data on the Census's Data site

You can download a copy of the exported zip file: `ACSDT1Y2019.B01003_2020-11-10T165412.zip`

Given file `acs-race.csv`

TK use csvflatten to show file, or include screenshot of data in Excel

As you can see the first row of "data" is not really data, but another row of meta header, which we do not need.

TK: Using sed

Keep all rows but the second row, which is unneeded metadata TK

```
$ sed '2d' acs-pop.csv
"GEO_ID","NAME","B01003_001E","B01003_001M"
"0200000US1","Northeast Region","55982803","*****"
"0200000US2","Midwest Region","68329004","*****"
"0200000US3","South Region","125580448","*****"
"0200000US4","West Region","78347268","*****"
```

using csvsed

```
$ csvslice -i '1-' acs-pop.csv
GEO_ID,NAME,B01003_001E,B01003_001M
0200000US1,Northeast Region,55982803,*****
0200000US2,Midwest Region,68329004,*****
0200000US3,South Region,125580448,*****
0200000US4,West Region,78347268,*****
```

CHAPTER 4

csvnorm

csvnorm TKTK

TK TK TK Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod

Table of contents

4.1 Usage reference

4.1.1 -c, –columns <COLUMNS>

A list of columns to apply csvnorm's effects.

<COLUMNS> should be a comma separated list of column indices, names or ranges to be extracted, e.g. "1,id,3-5". Defaults to all columns.

4.1.2 -S, –slugify

Convert values to snake_case style

4.1.3 -L, –lowercase

Transform letters into lowercase

4.1.4 -U, –uppercase

Transform letters into uppercase

4.1.5 –keep-lines

Do not convert newline characters to regular whitespace

4.2 High level overview

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

4.3 How csvnorm compares to existing tools

4.3.1 csvformat

TK

4.3.2 csvsed

TK

4.3.3 Agate

TK

4.3.4 Excel/Google Sheets

TK

4.4 Usecases

TK

CHAPTER 5

csvpivot

csvpivot is a command for producing simple pivot tables.

TK TK TK Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod

Table of contents

5.1 Usage reference

5.1.1 --list-aggs

List the available aggregate functions.

The available aggregate functions are a subset of those implemented in Agate's Aggregations API

- count
- max
- maxlength
- min
- mean
- median
- mode
- stdev
- sum

5.1.2 -r, --pivot-rows PIVOT_ROWNAMES

The column name(s) on which to use as pivot rows. Should be either one name (or index) or a comma- separated list with one name (or index)

5.1.3 -c, --pivot-column PIVOT_COLNAME

Optionally, a column name/id to use as a pivot column. Only one is allowed

5.1.4 -a, --agg AGGREGATES_LIST

The name of an aggregation to perform on each group of data in the pivot table. For aggregations that require an argument (i.e. a column name), pass in the aggregation name, followed by a colon, followed by comma-delimited arguments, e.g. *-a "sum:age"* and *-a "count:name,hello" To see a list, run 'csvpivot –list-aggs*

5.2 High level overview

Pivot Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Example of TK (what's the word for this) rows given a single column name, and counting the frequency of values:

```
$ csvpivot -r 'race' examples/peeps.csv
race,count_of
white,1
asian,2
black,2
latino,1
```

Example of pivoting (TK whats the word) given multiple rows and a column, and counting the combinations:

```
$ csvpivot -r 'race' -c 'gender' examples/peeps.csv
race,female,male
white,1,0
asian,1,1
black,2,0
latino,0,1
```

5.3 How it compares to existing tools

5.3.1 Excel/Google Sheets

TK

5.3.2 pandas.pivot_table()

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.pivot_table.html

5.3.3 agate.Table.pivot()

https://agate.readthedocs.io/en/1.6.1/api/table.html#agate.Table.pivot

https://agate.readthedocs.io/en/1.6.1/cookbook/transform.html?highlight=pivot#pivot-by-a-single-column

https://agate.readthedocs.io/en/1.6.1/cookbook/transform.html?highlight=pivot#pivot-by-multiple-columns

https://agate.readthedocs.io/en/1.6.1/cookbook/transform.html?highlight=pivot#pivot-to-sum

(can't do this) https://agate.readthedocs.io/en/1.6.1/cookbook/transform.html?highlight=pivot#pivot-to-percent-of-total

5.4 Usecases

Counting Congress demographics:

```
$ csvpivot -r party -c gender  examples/congress.csv  | csvlook

| party       |   M |   F |
| ----------- | --- | --- |
| Democrat    | 174 | 107 |
| Independent |   2 |   0 |
| Republican  | 229 |  24 |
| Libertarian |   1 |   0 |
```

Federal judges

https://www.pewresearch.org/fact-tank/2020/07/15/how-trump-compares-with-other-recent-presidents-in-appointing-federal-judges/

```
$ csvpivot examples/real/fed-judges-service.csv -r 'Appointing President' -c 'ABA␣
↪Rating' \
    | csvheader -R '1|President' \
    | csvcut -c 1,3,2,5,6 \
```

(continues on next page)

(continued from previous page)

```
    | csvlook

| President           | Well Qualified | Qualified | None | Not Qualified |
| ------------------- | -------------- | --------- | ---- | ------------- |
| Barack Obama        |            206 |       124 |    0 |             0 |
| Ronald Reagan       |            175 |       182 |   25 |             0 |
| Jimmy Carter        |            130 |       110 |   18 |             3 |
| Gerald Ford         |             27 |        37 |    3 |             0 |
| William J. Clinton  |            237 |       143 |    0 |             3 |
| George W. Bush      |            230 |        93 |    0 |             4 |
| Richard M. Nixon    |             87 |       105 |   17 |             0 |
| Donald J. Trump     |            158 |        56 |    0 |             7 |
| George H.W. Bush    |            113 |        80 |    1 |             0 |
```

5.5 Limitations/future fixes

If there are any NULL or irregular values in a column that is being summed/max/min/most aggregations, agate.Table will throw an error.

See more info about that issue here: https://github.com/wireservice/agate/issues/714#issuecomment-681176978

Assuming that agate's behavior can't/won't be changed, a possible solution is filling a to-be-aggregated column with non-null values (i.e. `0`). However, we should give the user the option of specifying that value. Also, it should probably require explicit enabling, so users who aren't aware their data contains non-null/numeric values are noisily informed.

CHAPTER 6

csvsed

csvsed is a command to do find-and-replace on a per-column basis.

For example, given a `data.csv` like this:

```
id,name
1,Mrs. Adams
2,Miss Miller
3,Mrs Smith
```

Replace by patternTKTK:

```
$ csvsed '(Mrs|Miss|Ms)\.?' 'Ms.' data.csv
id,name
1,Ms. Adams
2,Ms. Miller
3,Ms. Smith
```

Table of contents

6.1 Usage reference

6.1.1 -c, –columns <columns_list>

A comma separated list of column indices, names or ranges to be affected, e.g. “1,id,3-5”. Defaults to all columns.

6.1.2 -m, –match-literal

By default, [PATTERN] is assumed to be a regular expression. Set this flag to do a literal match and replacement.

6.1.3 -F, –filter

Only return rows that matched [PATTERN]. This has the same effect as operating on data filtered and piped from `csvgrep -r (or -m) [PATTERN]`

The main reason to use this is for terseness.

Given `data.csv` like this:

```
id,name,val
1,2,3x
3,4,5
6,7y,8z
```

And TK:

```
$ csvsed -F '[a-z]' '%' data.csv
id,name,val
1,2,3%
6,7%,8%
```

```
$ csvsed -F -c 1-20  'pattern' 'replace' data.csv
```

Versus TKTK:

```
$ csvgrep --any-match -c 1-20 -r pattern data.csv | csvsed -c 1-20 'pattern' 'replace'
```

6.2 High level description

Like `sed`, but on a per-column basis

Example:

```
$ csvsed "Ab[bi].+" "Abby" -E "(B|R)ob.*" "\1ob" -E "(?:Jack|John).*" "John" ␣
↪examples/aliases.csv


id,to,from
1,Abby,Bob
2,Bob,John
3,Abby,John
4,John,Abner
```

(continues on next page)

(continued from previous page)

```
5,Rob,John
6,Jon,Abby
7,Rob,Abby
```

6.3 Real-world use cases

6.3.1 Using csvsed to clean up the SSA babynames data

Continued from: *Adding a header to the Social Security babynames data*

Given file `babynames-yob.csv`

```
year,name,sex,count
yob1880.txt,Mary,F,7065
yob1880.txt,Anna,F,2604
yob1880.txt,Emma,F,2003
...
yob2018.txt,Zyrie,M,5
yob2018.txt,Zyron,M,5
yob2018.txt,Zzyzx,M,5
```

TK you can do:

TK make test

```
$ csvsed 'yob(\d{4}).txt' '\1' babynames-yob.csv
  year,name,sex,count
  1880,Mary,F,7065
  1880,James,F,22
  1880,Leslie,F,8
  1880,James,M,5927
```

CHAPTER 7

FAQ

Q. How is this related to wireservice/csvkit?

A. **csvmedkit** is an extension of *csvkit* (and thus has csvkit and agate as dependencies) that adds a bunch of new command-line utilities for data-wrangling convenience.

Q. What are the point of these new utilities?

A. As useful as core csvkit is, there are still a bunch of common data-wrangling tasks that are cumbersome to perform even when the data is in a spreadsheet or SQL database. "Cumbersome", in the sense that you'd basically have to write a custom Python script to do them.

CHAPTER 8

Indices and tables

- search

8.1 Credits

8.1.1 Development Lead

- Dan Nguyen <dansonguyen@gmail.com>

8.1.2 Contributors

None yet. Why not be the first?

8.2 History

8.2.1 0.0.0.1 (2020-10-02)

- Just grabbing the PyPI name for now

8.3 Cookbook of real-world CSV wrangling TKTK

A list of real-world use cases for **csvmedkit** command-line wrangling.

8.3.1 Browsing/understanding messy data

Figuring out what's in the NHTSA's safety-related defect complaint database

- Landing page: https://www-odi.nhtsa.dot.gov/downloads/
- README: https://www-odi.nhtsa.dot.gov/downloads/folders/Complaints/CMPL.txt
- Direct download (250MB+): https://www-odi.nhtsa.dot.gov/downloads/folders/Complaints/FLAT_CMPL.zip

Skimming the structure

TKTKTK

Looking at the first record:

```
source
    $ head -n 1 examples/real/nhtsa-complaints.txt | csvformat -t | csvheaders --HM |␣
↪csvflatten -P


    | field    | value                                                      |
    | -------- | ---------------------------------------------------------- |
    | field_1  | 1                                                          |
    | field_2  | 958173                                                     |
    | field_3  | Ford Motor Company                                         |
    | field_4  | LINCOLN                                                    |
    | field_5  | TOWN CAR                                                   |
    | field_6  | 1994                                                       |
    | field_7  | Y                                                          |
    | field_8  | 19941222                                                   |
    | field_9  | N                                                          |
    | field_10 | 0                                                          |
    | field_11 | 0                                                          |
    | field_12 | SERVICE BRAKES, HYDRAULIC:PEDALS AND LINKAGES              |
    | field_13 | HIGH LAND PA                                               |
    | field_14 | MI                                                         |
    | field_15 | 1LNLM82W8RY                                                |
    | field_16 | 19950103                                                   |
    | field_17 | 19950103                                                   |
    | field_18 |                                                            |
    | field_19 | 1                                                          |
    | field_20 | BRAKE PEDAL PUSH ROD RETAINER WAS NOT PROPERLY INSTALLED,  |
    |          | CAUSING BRAKES TO FAIL, RESULTING IN AN ACCIDENT AFTER     |
    |          | RECALL REPAIRS (94V-129). *AK                              |
    | field_21 | EVOQ                                                       |
    | field_22 |                                                            |
    | field_23 |                                                            |
    | field_24 |                                                            |
    | field_25 |                                                            |
    | field_26 |                                                            |
    | field_27 |                                                            |
    | field_28 |                                                            |
    | field_29 |                                                            |
    | field_30 |                                                            |
    | field_31 |                                                            |
    | field_32 |                                                            |
```

(continues on next page)

(continued from previous page)

```
    | field_33 |                                                                 |
    | field_34 |                                                                 |
    | field_35 |                                                                 |
    | field_36 |                                                                 |
    | field_37 |                                                                 |
    | field_38 |                                                                 |
    | field_39 |                                                                 |
    | field_40 |                                                                 |
    | field_41 |                                                                 |
    | field_42 |                                                                 |
    | field_43 |                                                                 |
    | field_44 |                                                                 |
    | field_45 |                                                                 |
    | field_46 | V                                                               |
    | field_47 |                                                                 |
    | field_48 |                                                                 |
    | field_49 |                                                                 |
```

8.4 Data samples

Table of contents

8.4.1 ids.csv

`Download`

Listing 1: ids.csv

```
id,val
0,a
1,b
2,c
3,d
4,e
5,f
```

8.4.2 hamlet.csv

`Download`

Listing 2: hamlet.csv

```
act,scene,speaker,lines
1,5,Horatio,"Propose the oath, my lord."
1,5,Hamlet,"Never to speak of this that you have seen,
```

(continues on next page)

(continued from previous page)

```
Swear by my sword."
1,5,Ghost,[Beneath] Swear.
3,4,Gertrude,"O, speak to me no more;
These words, like daggers, enter in mine ears;
No more, sweet Hamlet!"
4,7,Laertes,Know you the hand?
```