csvdiff Documentation

Release 0.3.1

Lars Yencken

Jul 20, 2017

Contents

Contents:

CHAPTER 1

csvdiff

Overview

Generate a diff between two CSV files on the command-line.

`csvdiff` allows you to compare the semantic contents of two CSV files, ignoring things like row and column ordering in order to get to what's actually changed. This is useful if you're comparing the output of an automatic system from one day to the next, so that you can look at just what's changed.

It's also useful for maintaining patches to third-party data. Diffs generated by `csvdiff` are a subset of JSON and can be stored and applied using the matching `csvpatch` command. If upstream data changes, you can fetch the new version and re-apply your changes to it easily.

Installing

You'll firstly need Python and pip. Then run:

```
pip install csvdiff
```

Examples

For example, suppose we have `a.csv`:

```
id,name,amount
1,bob,20
2,eva,63
3,sarah,7
4,jeff,19
6,fred,10
```

After some changes and corrections to the data, we now have `b.csv`:

```
id,name,amount
1,bob,23       <--- changed
3,sarah,7
4,jeff,19
5,mira,81      <--- added
6,fred,13      <--- changed
```

Now we can ask for a summary of differences:

```
$ csvdiff --style=summary id a.csv b.csv
1 rows removed (20.0%)
1 rows added (20.0%)
2 rows changed (40.0%)
```

Or look at the full diff pretty printed, to make it more readable:

```
$ csvdiff --style=pretty --output=diff.json id a.csv b.csv
$ cat diff.json
{
  "_index": [
    "id"
  ],
  "added": [
    {
      "amount": "81",
      "id": "5",
      "name": "mira"
    }
  ],
  "changed": [
    {
      "fields": {
        "amount": {
          "from": "20",
          "to": "23"
        }
      },
      "key": [
        "1"
      ]
    },
    {
      "fields": {
        "amount": {
          "from": "10",
          "to": "13"
        }
      },
      "key": [
        "6"
      ]
    }
  ],
  "removed": [
    {
      "amount": "63",
      "id": "2",
```

```
      "name": "eva"
    }
  ]
}
```

If you want to ignore a column from the comparison then you can do so by specifying a comma seperated list of column names to ignore. For example:

```
$ csvdiff --style=summary --ignore-columns=amount id a.csv b.csv
1 rows removed (20.0%)
1 rows added (20.0%)
0 rows changed (0%)
```

You can also choose to compare numeric fields only up to a certain number of significant figures. Use negative significant figures for orders of magnitude:

```
$ csvdiff --style=summary id a.csv c.csv
0 rows removed (0.0%)
0 rows added (0.0%)
2 rows changed (40.0%)
$ csvdiff --style=summary id --significance=-1 a.csv c.csv
files are identical
```

Diffs generated this way contain all the data that's changed, and can be reapplied later if the original data changes. For example, suppose more data gets added to `a.csv`, giving us `a-plus.csv`:

```
id,name,amount
1,bob,20
2,eva,63
3,sarah,7
4,jeff,19
6,fred,10
8,henry,9
```

We can reapply our changes with the `csvpatch` command:

```
$ csvpatch --input=diff.json --output=b-plus.csv a-plus.csv
$ cat b-plus.csv
id,name,amount
1,bob,23
3,sarah,7
4,jeff,19
5,mira,81
6,fred,13
8,henry,9
```

This can be useful if you're using csvdiff to transform data that's outside your control. In this case, you maintain the patch file and simply reapply it when the upstream data provider gives you a fresh file.

For more usage options, run `csvdiff --help` or `csvpatch --help`.

License

BSD license

CHAPTER 2

Installation

At the command line:

```
$ easy_install csvdiff
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv csvdiff
$ pip install csvdiff
```

CHAPTER 3

Usage

To use csvdiff in a project:

```
import csvdiff
```

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

Types of Contributions

Report Bugs

Report bugs at https://github.com/larsyencken/csvdiff/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

Write Documentation

csvdiff could always use more documentation, whether as part of the official csvdiff docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/larsyencken/csvdiff/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Get Started!

Ready to contribute? Here's how to set up *csvdiff* for local development.

1. Fork the *csvdiff* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/csvdiff.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv csvdiff
$ cd csvdiff/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 csvdiff tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/larsyencken/csvdiff/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ python -m unittest tests.test_csvdiff
```

CHAPTER 5

Credits

Development Lead

- Lars Yencken <lars@yencken.org>

Contributors

- Jason Marshall <j.j.marshall@kent.ac.uk>

CHAPTER 6

History

dev

- Add the –significance option to limit to significant figures.

0.3.2 (2017-07-20)

- Add the –sep option for different delimiters.
- Fix a bug when a patched document becomes empty (#29).

0.3.1 (2016-04-20)

- Fix a bug in summary mode.
- Check for rows bleeding into one another.

0.3.0 (2015-01-07)

- Standardise patch format with a JSON schema.
- Provide a matching csvpatch command applying diffs.
- Add a man page and docs for csvpatch.
- Use exit codes to indicate difference.
- Add a –quiet option to csvdiff.

0.2.0 (2014-12-30)

- Uses click for the command-line interface.
- Drop YAML support in favour of pretty-printed JSON.
- Uses –style option to change output style.
- Provides a full man page.

0.1.0 (2014-03-15)

- First release on PyPI.
- Generates a JSON or YAML difference between two CSV files
- Specify multiple key components with `-k`
- Can provide a difference summary
- Assumes files use standard comma-separation, double-quoting and a header row with field names

CHAPTER 7

Indices and tables

- genindex
- modindex
- search