csverve Documentation

Release 0.3.5+0.g6f1be9f.dirty

Shah Lab

Jun 02, 2023

CONTENTS:

CHAPTER

ONE

CSVERVE

Csverve, pronounced like “swerve” with a “v”, is a package for manipulating tabular data.

- Free software: MIT license
- Documentation: https://csverve.readthedocs.io.

1.1 Features

- Take in a regular gzipped CSV file and convert it to *csverve* format
- Merge gzipped CSZ files
- Concatenate gzipped CSV files (handles large datasets)
- Rewrite a gzipped CSV file (delete headers etc.)
- Annotate - add a column based on provided dictionary
- Write pandas DataFrame to *csverve* CSV
- Read a *csverve* CSV

1.2 Requirements

Every gzipped CSV file must be accompanied by a meta YAML file. The meta yaml file must have the exact name as the gzipped CSV file, with the addition of a *.yaml* ending.

1.2.1 csv.gz.yaml must contain:

- column names
- dtypes for each column
- separator
- header (bool) to specify if file has header or not

Example:

```
columns:
 - dtype: int
   name: prediction_id
 - dtype: str
   name: chromosome_1
 - dtype: str
   name: strand_1
 header: true
 sep: "\t"
```

1.3 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install csverve, run this command in your terminal:

```
$ pip install csverve
```

This is the preferred method to install csverve, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for csverve can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/mondrian-scwgs/csverve
```

Or download the tarball:

```
$ curl -OJL https://github.com/mondrian-scwgs/csverve/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER

THREE

USAGE

To use csverve in a project:

```
import csverve
```

CHAPTER

FOUR

CSVERVE

4.1 csverve package

4.1.1 Subpackages

csverve.api package

Submodules

csverve.api.api module

csverve.api.api.**add_col_from_dict**(*infile*, *col_data*, *outfile*, *dtypes*, *skip_header=False*, ***kwargs*)

TODO: fill this in Add column to gzipped CSV.

Parameters

- **infile** –
- **col_data** –
- **outfile** –
- **dtypes** –
- **skip_header** –

Returns

csverve.api.api.**annotate_csv**(*infile: str*, *annotation_df: DataFrame*, *outfile*, *annotation_dtypes*, *on='cell_id'*, *skip_header: bool = False*, ***kwargs*)

TODO: fill this in :param infile: :param annotation_df: :param outfile: :param annotation_dtypes: :param on: :param skip_header: :return:

csverve.api.api.**concatenate_csv**(*inputfiles: List[str]*, *output: str*, *skip_header: bool = False*, *drop_duplicates: bool = False*, ***kwargs*) → None

Concatenate gzipped CSV files, dtypes in meta YAML files must be the same.

Parameters

- **inputfiles** – List of gzipped CSV file paths, or a dictionary where the keys are file paths.
- **output** – Path of resulting concatenated gzipped CSV file and meta YAML.
- **skip_header** – boolean, True = write header, False = don't write header.

Returns

csverve.api.api.**concatenate_csv_files_pandas**(*in_filenames: Union[List[str], Dict[str, str]]*, *out_filename: str*, *dtypes: Dict[str, str]*, *skip_header: bool = False*, *drop_duplicates: bool = False*, ***kwargs*) → None

Concatenate gzipped CSV files.

Parameters

- **in_filenames** – List of gzipped CSV file paths, or a dictionary where the keys are file paths.
- **out_filename** – Path of resulting concatenated gzipped CSV file and meta YAML.
- **dtypes** – Dictionary of pandas dtypes, where key = column name, value = dtype.
- **skip_header** – boolean, True = write header, False = don't write header.

Returns

csverve.api.api.**concatenate_csv_files_quick_lowmem**(*inputfiles: List[str]*, *output: str*, *dtypes: Dict[str, str]*, *columns: List[str]*, *skip_header: bool = False*, ***kwargs*) → None

Concatenate gzipped CSV files.

Parameters

- **inputfiles** – List of gzipped CSV file paths.
- **output** – Path of resulting concatenated gzipped CSV file and meta YAML.
- **dtypes** – Dictionary of pandas dtypes, where key = column name, value = dtype.
- **columns** – List of column names for newly concatenated gzipped CSV file.
- **skip_header** – boolean, True = write header, False = don't write header.

Returns

csverve.api.api.**get_columns**(*infile*)

csverve.api.api.**get_dtypes**(*infile*)

csverve.api.api.**merge_csv**(*in_filenames: Union[List[str], Dict[str, str]]*, *out_filename: str*, *how: str*, *on: List[str]*, *skip_header: bool = False*, ***kwargs*) → None

Create one gzipped CSV out of multiple gzipped CSVs.

Parameters

- **in_filenames** – Dictionary containing file paths as keys
- **out_filename** – Path to newly merged CSV
- **how** – How to join DataFrames (inner, outer, left, right).
- **on** – Column(s) to join on, comma separated if multiple.
- **skip_header** – boolean, True = write header, False = don't write header

Returns

csverve.api.api.**read_csv**(*infile: str*, *chunksize: Optional[int] = None*, *usecols=None*, *dtype=None*) → DataFrame

Read in CSV file and return as a pandas DataFrame.

Assumes a YAML meta file in the same path with the same name, with a .yaml extension. YAML file structure is atop this file.

Parameters

- **infile** – Path to CSV file.
- **chunksize** – Number of rows to read at a time (optional, applies to large datasets).
- **usecols** – Restrict to specific columns (optional).
- **dtype** – Override the dtypes on specific columns (optional).

Returns

pandas DataFrame.

csverve.api.api.**remove_duplicates**(*filepath: str*, *outputfile: str*, *skip_header: bool = False*) → None

remove duplicate rows

Assumes a YAML meta file in the same path with the same name, with a .yaml extension. YAML file structure is atop this file.

Parameters

- **filepath** – Path to CSV file.
- **outputfile** – Path to CSV file.

csverve.api.api.**rewrite_csv_file**(*filepath: str*, *outputfile: str*, *skip_header: bool = False*, *dtypes: Optional[Dict[str, str]] = None*, ***kwargs*) → None

Generate header less csv files.

Parameters

- **filepath** – File path of CSV.
- **outputfile** – File path of header less CSV to be generated.
- **skip_header** – boolean, True = write header, False = don't write header.
- **dtypes** – Dictionary of pandas dtypes, where key = column name, value = dtype.

Returns

csverve.api.api.**simple_annotate_csv**(*in_f: str*, *out_f: str*, *col_name: str*, *col_val: str*, *col_dtype: str*, *skip_header: bool = False*, ***kwargs*) → None

Simplified version of the annotate_csv method. Add column with the same value for all rows.

Parameters

- **in_f** –
- **out_f** –
- **col_name** –
- **col_val** –
- **col_dtype** –
- **skip_header** –

Returns

csverve.api.api.**write_dataframe_to_csv_and_yaml**(*df: DataFrame*, *outfile: str*, *dtypes: Dict[str, str]*, *skip_header: bool = False*, ***kwargs*) → None

Output pandas dataframe to a CSV and meta YAML files.

Parameters

- **df** – pandas DataFrame.
- **outfile** – Path of CSV & YAML file to be written to.
- **dtypes** – dictionary of pandas dtypes by column, keys = column name, value = dtype.
- **skip_header** – boolean, True = skip writing header, False = write header

Returns

Module contents

csverve.core package

Submodules

csverve.core.csverve_input module

class csverve.core.csverve_input.**CsverveInput**(*filepath: str*)

Bases: object

property columns: List[str]

get the list of columns

Returns
separator

property dtypes: Dict[str, str]

get the data types

Returns
dtypes

property header: bool

True if file has header

Returns
header

read_csv(*chunksize: Optional[int] = None*, *usecols=None*, *dtype=None*) → DataFrame

Read CSV.

Parameters

- **chunksize** – Number of rows to read at a time (optional, applies to large datasets).
- **usecols** – Restrict to specific columns (optional).
- **dtype** – Override the dtypes on specific columns (optional).

Returns
pandas DataFrame.

property separator: str

get the separator used

Returns
separator

property yaml_file: str

Append '.yaml' to CSV path.

Returns

YAML metadata path.

csverve.core.csverve_output module

class csverve.core.csverve_output.**CsverveOutput**(*filepath: str*, *dtypes: Dict[str, str]*, *columns: List[str]*, *skip_header: bool = False*, *na_rep: str = 'NaN'*, *sep: str = ','*)

Bases: object

write_yaml() → None

Write .yaml file.

Returns

property yaml_file: str

Append '.yaml' to CSV path.

Returns

YAML metadata path.

csverve.core.csverve_output_data_frame module

class csverve.core.csverve_output_data_frame.**CsverveOutputDataFrame**(*df: DataFrame*, *filepath: str*, *dtypes: Dict[str, str]*, *skip_header: bool = False*, *na_rep: str = 'NaN'*, *sep: str = ','*)

Bases: *CsverveOutput*

write_df() → None

Write out dataframe to CSV.

Parameters

- **df** – Pandas DataFrames.
- **chunks** – bool.

Returns

csverve.core.csverve_output_file_stream module

class csverve.core.csverve_output_file_stream.**CsverveOutputFileStream**(*filepath: str*, *dtypes: Dict[str, str]*, *columns: List[str]*, *skip_header: bool = False*, *na_rep: str = 'NaN'*, *sep: str = ','*)

Bases: *CsverveOutput*

rewrite_csv(*csvfile: str*) → None

Rewrite CSV. :param csvfile: Filepath of CSV file. :return:

write_data_streams(*csvfiles: List[str]*) → None

Write data streams. :param csvfiles: List of CSV files paths. :return:

csverve.core.irregular_csv_input module

class csverve.core.irregular_csv_input.**IrregularCsverveInput**(*filepath: str*, *dtypes: Dict[str, str]*, *sep=','*)

Bases: object

get_columns() → List[str]

Detect whether file is tab or comma separated from header. :return: ' ', or ',', or raise error if unable to detect separator.

read_csv(*chunksize: Optional[int] = None*) → DataFrame

Read CSV.

Parameters
chunksize – Number of rows to read at a time (optional, applies to large datasets).

Returns
pandas DataFrame.

property yaml_file: str

Append '.yaml' to CSV path.

Returns
YAML metadata path.

Module contents

csverve.errors package

Submodules

csverve.errors.errors module

exception csverve.errors.errors.**CsverveAnnotateError**

Bases: Exception

exception csverve.errors.errors.**CsverveConcatException**

Bases: Exception

exception csverve.errors.errors.**CsverveDtypeError**

Bases: Exception

exception csverve.errors.errors.**CsverveInputError**

Bases: Exception

exception csverve.errors.errors.**CsverveMergeColumnMismatchException**

Bases: Exception

exception csverve.errors.errors.**CsverveMergeCommonColException**

Bases: Exception

exception csverve.errors.errors.**CsverveMergeDtypesEmptyMergeSet**

Bases: Exception

exception csverve.errors.errors.**CsverveMergeException**

Bases: Exception

exception csverve.errors.errors.**CsverveParseError**

Bases: Exception

exception csverve.errors.errors.**CsverveWriterError**

Bases: Exception

exception csverve.errors.errors.**DtypesMergeException**

Bases: Exception

Module contents

csverve.utils package

Submodules

csverve.utils.utils module

csverve.utils.utils.**merge_dtypes**(*dtypes_all: List[Dict[str, str]]*) → Dict[str, str]

Merge pandas dtypes.

Parameters
dtypes_all – List of dtypes dictionaries, where key = column name, value = pandas dtype.

Returns
Merged dtypes dictionary.

csverve.utils.utils.**merge_frames**(*frames: List[DataFrame]*, *how: str*, *on: List[str]*) → DataFrame

Takes in a list of pandas DataFrames, and merges into a single DataFrame. #TODO: add handling if empty list is given

Parameters

- **frames** – List of pandas DataFrames.
- **how** – How to join DataFrames (inner, outer, left, right).
- **on** – Column(s) to join on, comma separated if multiple.

Returns
merged pandas DataFrame.

csverve.utils.utils.**pandas_to_std_types**(*dtype: Any*) → str

Module contents

4.1.2 Submodules

4.1.3 csverve.cli module

Console script for csverve.

4.1.4 Module contents

CHAPTER

FIVE

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:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/mondrian-scwgs/csverve/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.

5.1.2 Fix Bugs

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

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

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

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/mondrian-scwgs/csverve/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 :)

5.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/csverve.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 csverve
$ cd csverve/
$ 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 csverve tests
$ python setup.py test or pytest
$ 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.

5.3 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 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/mondrian-scwgs/csverve/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

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

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER

SIX

CREDITS

6.1 Development Lead

- Shah Lab <todo@todo.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER

SEVEN

HISTORY

7.1 0.1.0 (2020-12-16)

- First release on PyPI.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

C

INDEX

P

R

S

W

Y