pandas-validation Documentation

Release 0.5.0

Markus Englund

Jun 13, 2019

Contents

pandas-validation is a small Python package for casting and validating data handled with pandas.

CHAPTER 1

Installing pandas-validation

For most users, the easiest way is probably to install the latest version hosted on PyPI:

```
$ pip install pandas-validation
```

The project is hosted at https://github.com/jmenglund/pandas-validation and can also be installed using git:

```
$ git clone https://github.com/jmenglund/pandas-validation.git
$ cd pandas-validation
$ python setup.py install
```

Tip: You may consider installing pandas-validation and its required Python packages within a virtual environment in order to avoid cluttering your system's Python path. See for example the environment management system conda or the package virtualenv.

CHAPTER 2

Quickstart

This guide gives you a brief introduction on how to use pandas-validation. The library contains four core functions that let you validate values in a pandas Series (or a DataFrame column). The examples below will help you get started. If you want to know more, I suggest that you have a look at the *API reference*.

- *Validate dates*
- *Validate timestamps*
- *Validate numeric values*
- *Validate strings*

The code examples below assume that you first do the following imports:

```
>>> import numpy as np
>>> import pandas as pd
>>> import pandasvalidation as pv
```

2.1 Validate dates

Our first example shows how to validate a pandas Series with a few dates specified with Python's *datetime.date* data type. Values of other types are replaced with `NaT` ("not a time") prior to the validation. Warnings then inform the user if any of the values are invalid. If *return_type* is set to `values`, a pandas Series will be returned with only the valid dates.

```
>>> from datetime import date
>>> s1 = pd.Series(
...     [
            date(2010, 10, 7),
            date(2018, 3, 15),
            date(2018, 3, 15),
            np.nan
        ], name='My dates')
```

(continues on next page)

(continued from previous page)

```
>>> pv.validate_date(
...     s1,
...     nullable=False,
...     unique=True,
...     min_date=date(2014, 1, 5),
...     max_date=date(2015, 2, 15),
...     return_type=None)
ValidationWarning: 'My dates': NaT value(s); duplicates; date(s) too early; date(s)␣
↪too late.
```

2.2 Validate timestamps

Validation of timestamps works in the same way as date validation. The major difference is that only values of type *pandas.Timestamp* are taken into account. Values of other types are replaced by `NaT`. If *return_type* is set to `values`, a pandas Series will be returned with only the valid timestamps.

```
>>> s2 = pd.Series(
...     [
...         pd.Timestamp(2018, 2, 7, 12, 31, 0),
...         pd.Timestamp(2018, 2, 7, 13, 6, 0),
...         pd.Timestamp(2018, 2, 7, 13, 6, 0),
...         np.nan
...     ], name='My timestamps')
>>> pv.validate_timestamp(
...     s2,
...     nullable=False,
...     unique=True,
...     min_timestamp=pd.Timestamp(2014, 1, 5, 0, 0, 0),
...     max_timestamp=pd.Timestamp(2018, 2, 7, 13, 0, 0),
...     return_type=None)
ValidationWarning: 'My timestamps': NaT value(s); duplicates; timestamp(s) too late.
```

2.3 Validate numeric values

Validation of numeric values (e.g. floats and integers) follows the same general principles as the validation of dates and timestamps. Non-numeric values are treated as `NaN`, and warnings are issued to indicate invalid values to the user. If *return_type* is set to `values`, a pandas Series will be returned with only the valid numeric values.

Note: Prior to version 0.5.0, some non-numeric data types were automatically converted numeric types before the validation. This was often convenient but could also lead to unexpected behaviour. The current implementation is cleaner and gives the user more control over the data types.

```
>>> s3 = pd.Series(
...     [1, 1, 2.3, np.nan],
...     name='My numeric values')
>>> pv.validate_numeric(
...     s3,
...     nullable=False,
...     unique=True,
```

(continues on next page)

(continued from previous page)

```
...     integer=True,
...     min_value=2,
...     max_value=2,
...     return_type=None)
ValidationWarning: 'My numeric values': NaN value(s); duplicates; non-integer(s);␣
↪value(s) too low; values(s) too high.
```

2.4 Validate strings

String validation works in the same way as the other validations, but concerns only strings. Values of other types, like numbers and timestamps, are simply replaced with `NaN` values before the validation takes place. If *return_type* is set to `values`, a pandas Series will be returned with only the valid strings.

Note: Prior to version 0.5.0, some non-string data types were automatically converted to strings before the validation. This was often convenient but could also lead to unexpected behaviour. The current implementation is cleaner and gives the user more control over the data types.

```
>>> s4 = pd.Series(
...     ['1', 'ab\n', 'Ab', 'AB', np.nan],
...     name='My strings')
>>> pv.validate_string(
...     s4,
...     nullable=False,
...     unique=True,
...     min_length=2,
...     max_length=2,
...     case='lower',
...     newlines=False,
...     whitespace=False,
...     return_type=None)
ValidationWarning: 'My strings': NaN value(s); string(s) too short; string(s) too␣
↪long; wrong case letter(s); newline character(s); whitespace.
```

CHAPTER 3

API Reference

This document describes the API of the *pandasvalidation* module.

Module for validating data with the library pandas.

exception pandasvalidation.**ValidationWarning**
Bases: Warning

pandasvalidation.**mask_nonconvertible**(*series*, *to_datatype*, *datetime_format=None*, *exact_date=True*)
Return a boolean same-sized object indicating whether values cannot be converted.

Parameters

- **series** (*pandas.Series*) – Values to check.
- **to_datatype** (*str*) – Datatype to which values should be converted. Available values are 'numeric' and 'datetime'.
- **datetime_format** (*str*) – strftime to parse time, eg '%d/%m/%Y', note that '%f' will parse all the way up to nanoseconds. Optional.
- **exact_date** (*bool*) –
  - If True (default), require an exact format match.
  - If False, allow the format to match anywhere in the target string.

pandasvalidation.**to_datetime**(*arg*, *dayfirst=False*, *yearfirst=False*, *utc=None*, *box=True*, *format=None*, *exact=True*, *coerce=None*, *unit='ns'*, *infer_datetime_format=False*)
Convert argument to datetime and set nonconvertible values to NaT.

This function calls `to_datetime()` with `errors='coerce'` and issues a warning if values cannot be converted.

pandasvalidation.**to_numeric**(*arg*)
Convert argument to numeric type and set nonconvertible values to NaN.

This function calls `to_numeric()` with `errors='coerce'` and issues a warning if values cannot be converted.

pandasvalidation.**to_string**(*series*, *float_format='%g'*, *datetime_format='%Y-%m-%d'*)
Convert values in a pandas Series to strings.

Parameters

- **series** (*pandas.Series*) – Values to convert.
- **float_format** (*str*) – Format string for floating point number. Default: '%g'.
- **datetime_format** (*str*) – Format string for datetime type. Default: '%Y-%m-%d'

Returns converted

Return type pandas.Series

pandasvalidation.**validate_date**(*series*, *nullable=True*, *unique=False*, *min_date=None*, *max_date=None*, *return_type=None*)
Validate a pandas Series with values of type *datetime.date*. Values of a different data type will be replaced with NaN prior to the validataion.

Parameters

- **series** (*pandas.Series*) – Values to validate.
- **nullable** (*bool*) – If False, check for NaN values. Default: True.
- **unique** (*bool*) – If True, check that values are unique. Default: False
- **min_date** (*datetime.date*) – If defined, check for values before min_date. Optional.
- **max_date** (*datetime.date*) – If defined, check for value later than max_date. Optional.
- **return_type** (*str*) – Kind of data object to return; 'mask_series', 'mask_frame' or 'values'. Default: None.

pandasvalidation.**validate_datetime**(*series*, *nullable=True*, *unique=False*, *min_datetime=None*, *max_datetime=None*, *return_type=None*)
Validate a pandas Series containing datetimes.

Deprecated since version 0.5.0: *validate_datetime()* will be removed in version 0.7.0. Use *validate_date()* or *validate_timestamp()* instead.

Parameters

- **series** (*pandas.Series*) – Values to validate.
- **nullable** (*bool*) – If False, check for NaN values. Default: True.
- **unique** (*bool*) – If True, check that values are unique. Default: False
- **min_datetime** (*str*) – If defined, check for values before min_datetime. Optional.
- **max_datetime** (*str*) – If defined, check for value later than max_datetime. Optional.
- **return_type** (*str*) – Kind of data object to return; 'mask_series', 'mask_frame' or 'values'. Default: None.

pandasvalidation.**validate_numeric**(*series*, *nullable=True*, *unique=False*, *integer=False*, *min_value=None*, *max_value=None*, *return_type=None*)
Validate a pandas Series containing numeric values.

Parameters

- **series** (*pandas.Series*) – Values to validate.
- **nullable** (*bool*) – If False, check for NaN values. Default: True

- **unique** (*bool*) – If True, check that values are unique. Default: False
- **integer** (*bool*) – If True, check that values are integers. Default: False
- **min_value** (*int*) – If defined, check for values below minimum. Optional.
- **max_value** (*int*) – If defined, check for value above maximum. Optional.
- **return_type** (*str*) – Kind of data object to return; 'mask_series', 'mask_frame' or 'values'. Default: None.

pandasvalidation.**validate_string**(*series*, *nullable=True*, *unique=False*, *min_length=None*, *max_length=None*, *case=None*, *newlines=True*, *trailing_whitespace=True*, *whitespace=True*, *matching_regex=None*, *non_matching_regex=None*, *whitelist=None*, *blacklist=None*, *return_type=None*)

Validate a pandas Series with strings. Non-string values will be converted to strings prior to validation.

Parameters

- **series** (*pandas.Series*) – Values to validate.
- **nullable** (*bool*) – If False, check for NaN values. Default: True.
- **unique** (*bool*) – If True, check that values are unique. Default: False.
- **min_length** (*int*) – If defined, check for strings shorter than minimum length. Optional.
- **max_length** (*int*) – If defined, check for strings longer than maximum length. Optional.
- **case** (*str*) – Check for a character case constraint. Available values are 'lower', 'upper' and 'title'. Optional.
- **newlines** (*bool*) – If False, check for newline characters. Default: True.
- **trailing_whitespace** (*bool*) – If False, check for trailing whitespace. Default: True.
- **whitespace** (*bool*) – If False, check for whitespace. Default: True.
- **matching_regex** (*str*) – Check that strings matches some regular expression. Optional.
- **non_matching_regex** (*str*) – Check that strings do not match some regular expression. Optional.
- **whitelist** (*list*) – Check that values are in *whitelist*. Optional.
- **blacklist** (*list*) – Check that values are not in *blacklist*. Optional.
- **return_type** (*str*) – Kind of data object to return; 'mask_series', 'mask_frame' or 'values'. Default: None.

pandasvalidation.**validate_timestamp**(*series*, *nullable=True*, *unique=False*, *min_timestamp=None*, *max_timestamp=None*, *return_type=None*)

Validate a pandas Series with values of type *pandas.Timestamp*. Values of a different data type will be replaced with NaT prior to the validataion.

Parameters

- **series** (*pandas.Series*) – Values to validate.
- **nullable** (*bool*) – If False, check for NaN values. Default: True.
- **unique** (*bool*) – If True, check that values are unique. Default: False

- **min_timestamp** (*pandas.Timestamp*) – If defined, check for values before min_timestamp. Optional.
- **max_timestamp** (*pandas.Timestamp*) – If defined, check for value later than max_timestamp. Optional.
- **return_type** (*str*) – Kind of data object to return; 'mask_series', 'mask_frame' or 'values'. Default: None.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index