Timedelta

- Bug in `Timedelta.__mul__()` where multiplying by `NaT` returned `NaT` instead of raising a `TypeError` (GH19819)
- Bug in *`Series`* with `dtype='timedelta64[ns]'` where addition or subtraction of `TimedeltaIndex` had results cast to `dtype='int64'` (GH17250)
- Bug in *`Series`* with `dtype='timedelta64[ns]'` where addition or subtraction of `TimedeltaIndex` could return a `Series` with an incorrect name (GH19043)
- Bug in `Timedelta.__floordiv__()` and `Timedelta.__rfloordiv__()` dividing by many incompatible numpy objects was incorrectly allowed (GH18846)
- Bug where dividing a scalar timedelta-like object with *`TimedeltaIndex`* performed the reciprocal operation (GH19125)
- Bug in *`TimedeltaIndex`* where division by a `Series` would return a `TimedeltaIndex` instead of a `Series` (GH19042)
- Bug in `Timedelta.__add__()`, `Timedelta.__sub__()` where adding or subtracting a `np.timedelta64` object would return another `np.timedelta64` instead of a `Timedelta` (GH19738)
- Bug in `Timedelta.__floordiv__()`, `Timedelta.__rfloordiv__()` where operating with a `Tick` object would raise a `TypeError` instead of returning a numeric value (GH19738)
- Bug in *`Period.asfreq()`* where periods near `datetime(1, 1, 1)` could be converted incorrectly (GH19643, GH19834)
- Bug in *`Timedelta.total_seconds()`* causing precision errors, for example `Timedelta('30S').total_seconds()==30.000000000000004` (GH19458)
- Bug in `Timedelta.__rmod__()` where operating with a `numpy.timedelta64` returned a `timedelta64` object instead of a `Timedelta` (GH19820)
- Multiplication of *`TimedeltaIndex`* by `TimedeltaIndex` will now raise `TypeError` instead of raising `ValueError` in cases of length mis-match (GH19333)
- Bug in indexing a *`TimedeltaIndex`* with a `np.timedelta64` object which was raising a `TypeError` (GH20393)

Timezones

- Bug in creating a `Series` from an array that contains both tz-naive and tz-aware values will result in a `Series` whose dtype is tz-aware instead of object (GH16406)
- Bug in comparison of timezone-aware *`DatetimeIndex`* against `NaT` incorrectly raising `TypeError` (GH19276)
- Bug in `DatetimeIndex.astype()` when converting between timezone aware dtypes, and converting from timezone aware to naive (GH18951)
- Bug in comparing *`DatetimeIndex`*, which failed to raise `TypeError` when attempting to compare timezone-aware and timezone-naive datetimelike objects (GH18162)
- Bug in localization of a naive, datetime string in a `Series` constructor with a `datetime64[ns, tz]` dtype (GH174151)
- *`Timestamp.replace()`* will now handle Daylight Savings transitions gracefully (GH18319)
- Bug in tz-aware *`DatetimeIndex`* where addition/subtraction with a *`TimedeltaIndex`* or array with `dtype='timedelta64[ns]'` was incorrect (GH17558)

- Bug in `DatetimeIndex.insert()` where inserting `NaT` into a timezone-aware index incorrectly raised (GH16357)
- Bug in *`DataFrame`* constructor, where tz-aware Datetimeindex and a given column name will result in an empty `DataFrame` (GH19157)
- Bug in *`Timestamp.tz_localize()`* where localizing a timestamp near the minimum or maximum valid values could overflow and return a timestamp with an incorrect nanosecond value (GH12677)
- Bug when iterating over *`DatetimeIndex`* that was localized with fixed timezone offset that rounded nanosecond precision to microseconds (GH19603)
- Bug in *`DataFrame.diff()`* that raised an `IndexError` with tz-aware values (GH18578)
- Bug in *`melt()`* that converted tz-aware dtypes to tz-naive (GH15785)
- Bug in `Dataframe.count()` that raised an `ValueError`, if `Dataframe.dropna()` was called for a single column with timezone-aware values. (GH13407)

Offsets

- Bug in `WeekOfMonth` and `Week` where addition and subtraction did not roll correctly (GH18510, GH18672, GH18864)
- Bug in `WeekOfMonth` and `LastWeekOfMonth` where default keyword arguments for constructor raised `ValueError` (GH19142)
- Bug in `FY5253Quarter`, `LastWeekOfMonth` where rollback and rollforward behavior was inconsistent with addition and subtraction behavior (GH18854)
- Bug in `FY5253` where `datetime` addition and subtraction incremented incorrectly for dates on the year-end but not normalized to midnight (GH18854)
- Bug in `FY5253` where date offsets could incorrectly raise an `AssertionError` in arithmetic operations (GH14774)

Numeric

- Bug in *`Series`* constructor with an int or float list where specifying `dtype=str`, `dtype='str'` or `dtype='U'` failed to convert the data elements to strings (GH16605)
- Bug in *`Index`* multiplication and division methods where operating with a `Series` would return an `Index` object instead of a `Series` object (GH19042)
- Bug in the *`DataFrame`* constructor in which data containing very large positive or very large negative numbers was causing `OverflowError` (GH18584)
- Bug in *`Index`* constructor with `dtype='uint64'` where int-like floats were not coerced to *`UInt64Index`* (GH18400)
- Bug in *`DataFrame`* flex arithmetic (e.g. `df.add(other, fill_value=foo)`) with a `fill_value` other than `None` failed to raise `NotImplementedError` in corner cases where either the frame or `other` has length zero (GH19522)
- Multiplication and division of numeric-dtyped *`Index`* objects with timedelta-like scalars returns `TimedeltaIndex` instead of raising `TypeError` (GH19333)
- Bug where `NaN` was returned instead of 0 by *`Series.pct_change()`* and *`DataFrame.pct_change()`* when `fill_method` is not `None` (GH19873)

Strings

- Bug in `Series.str.get()` with a dictionary in the values and the index not in the keys, raising *KeyError* (GH20671)

Indexing

- Bug in `Index` construction from list of mixed type tuples (GH18505)
- Bug in `Index.drop()` when passing a list of both tuples and non-tuples (GH18304)
- Bug in `DataFrame.drop()`, `Panel.drop()`, `Series.drop()`, `Index.drop()` where no `KeyError` is raised when dropping a non-existent element from an axis that contains duplicates (GH19186)
- Bug in indexing a datetimelike `Index` that raised `ValueError` instead of `IndexError` (GH18386).
- `Index.to_series()` now accepts `index` and `name` kwargs (GH18699)
- `DatetimeIndex.to_series()` now accepts `index` and `name` kwargs (GH18699)
- Bug in indexing non-scalar value from `Series` having non-unique `Index` will return value flattened (GH17610)
- Bug in indexing with iterator containing only missing keys, which raised no error (GH20748)
- Fixed inconsistency in `.ix` between list and scalar keys when the index has integer dtype and does not include the desired keys (GH20753)
- Bug in `__setitem__` when indexing a `DataFrame` with a 2-d boolean ndarray (GH18582)
- Bug in `str.extractall` when there were no matches empty `Index` was returned instead of appropriate `MultiIndex` (GH19034)
- Bug in `IntervalIndex` where empty and purely NA data was constructed inconsistently depending on the construction method (GH18421)
- Bug in `IntervalIndex.symmetric_difference()` where the symmetric difference with a non-`IntervalIndex` did not raise (GH18475)
- Bug in `IntervalIndex` where set operations that returned an empty `IntervalIndex` had the wrong dtype (GH19101)
- Bug in `DataFrame.drop_duplicates()` where no `KeyError` is raised when passing in columns that don't exist on the `DataFrame` (GH19726)
- Bug in `Index` subclasses constructors that ignore unexpected keyword arguments (GH19348)
- Bug in `Index.difference()` when taking difference of an `Index` with itself (GH20040)
- Bug in `DataFrame.first_valid_index()` and `DataFrame.last_valid_index()` in presence of entire rows of NaNs in the middle of values (GH20499).
- Bug in `IntervalIndex` where some indexing operations were not supported for overlapping or non-monotonic `uint64` data (GH20636)
- Bug in `Series.is_unique` where extraneous output in stderr is shown if Series contains objects with `__ne__` defined (GH20661)
- Bug in `.loc` assignment with a single-element list-like incorrectly assigns as a list (GH19474)
- Bug in partial string indexing on a `Series/DataFrame` with a monotonic decreasing `DatetimeIndex` (GH19362)
- Bug in performing in-place operations on a `DataFrame` with a duplicate `Index` (GH17105)

- Bug in `IntervalIndex.get_loc()` and `IntervalIndex.get_indexer()` when used with an `IntervalIndex` containing a single interval (GH17284, GH20921)
- Bug in `.loc` with a `uint64` indexer (GH20722)

MultiIndex

- Bug in `MultiIndex.__contains__()` where non-tuple keys would return `True` even if they had been dropped (GH19027)
- Bug in `MultiIndex.set_labels()` which would cause casting (and potentially clipping) of the new labels if the `level` argument is not 0 or a list like [0, 1, ...] (GH19057)
- Bug in `MultiIndex.get_level_values()` which would return an invalid index on level of ints with missing values (GH17924)
- Bug in `MultiIndex.unique()` when called on empty `MultiIndex` (GH20568)
- Bug in `MultiIndex.unique()` which would not preserve level names (GH20570)
- Bug in `MultiIndex.remove_unused_levels()` which would fill nan values (GH18417)
- Bug in `MultiIndex.from_tuples()` which would fail to take zipped tuples in python3 (GH18434)
- Bug in `MultiIndex.get_loc()` which would fail to automatically cast values between float and int (GH18818, GH15994)
- Bug in `MultiIndex.get_loc()` which would cast boolean to integer labels (GH19086)
- Bug in `MultiIndex.get_loc()` which would fail to locate keys containing `NaN` (GH18485)
- Bug in `MultiIndex.get_loc()` in large `MultiIndex`, would fail when levels had different dtypes (GH18520)
- Bug in indexing where nested indexers having only numpy arrays are handled incorrectly (GH19686)

I/O

- `read_html()` now rewinds seekable IO objects after parse failure, before attempting to parse with a new parser. If a parser errors and the object is non-seekable, an informative error is raised suggesting the use of a different parser (GH17975)
- `DataFrame.to_html()` now has an option to add an id to the leading *<table>* tag (GH8496)
- Bug in `read_msgpack()` with a non existent file is passed in Python 2 (GH15296)
- Bug in `read_csv()` where a `MultiIndex` with duplicate columns was not being mangled appropriately (GH18062)
- Bug in `read_csv()` where missing values were not being handled properly when `keep_default_na=False` with dictionary `na_values` (GH19227)
- Bug in `read_csv()` causing heap corruption on 32-bit, big-endian architectures (GH20785)
- Bug in `read_sas()` where a file with 0 variables gave an `AttributeError` incorrectly. Now it gives an `EmptyDataError` (GH18184)
- Bug in `DataFrame.to_latex()` where pairs of braces meant to serve as invisible placeholders were escaped (GH18667)
- Bug in `DataFrame.to_latex()` where a `NaN` in a `MultiIndex` would cause an `IndexError` or incorrect output (GH14249)

- Bug in `DataFrame.to_latex()` where a non-string index-level name would result in an `AttributeError` (GH19981)
- Bug in `DataFrame.to_latex()` where the combination of an index name and the *index_names=False* option would result in incorrect output (GH18326)
- Bug in `DataFrame.to_latex()` where a `MultiIndex` with an empty string as its name would result in incorrect output (GH18669)
- Bug in `DataFrame.to_latex()` where missing space characters caused wrong escaping and produced non-valid latex in some cases (GH20859)
- Bug in `read_json()` where large numeric values were causing an `OverflowError` (GH18842)
- Bug in `DataFrame.to_parquet()` where an exception was raised if the write destination is S3 (GH19134)
- `Interval` now supported in `DataFrame.to_excel()` for all Excel file types (GH19242)
- `Timedelta` now supported in `DataFrame.to_excel()` for all Excel file types (GH19242, GH9155, GH19900)
- Bug in `pandas.io.stata.StataReader.value_labels()` raising an `AttributeError` when called on very old files. Now returns an empty dict (GH19417)
- Bug in `read_pickle()` when unpickling objects with `TimedeltaIndex` or `Float64Index` created with pandas prior to version 0.20 (GH19939)
- Bug in `pandas.io.json.json_normalize()` where sub-records are not properly normalized if any sub-records values are NoneType (GH20030)
- Bug in `usecols` parameter in `read_csv()` where error is not raised correctly when passing a string. (GH20529)
- Bug in `HDFStore.keys()` when reading a file with a soft link causes exception (GH20523)
- Bug in `HDFStore.select_column()` where a key which is not a valid store raised an `AttributeError` instead of a `KeyError` (GH17912)

Plotting

- Better error message when attempting to plot but matplotlib is not installed (GH19810).
- `DataFrame.plot()` now raises a `ValueError` when the `x` or `y` argument is improperly formed (GH18671)
- Bug in `DataFrame.plot()` when `x` and `y` arguments given as positions caused incorrect referenced columns for line, bar and area plots (GH20056)
- Bug in formatting tick labels with `datetime.time()` and fractional seconds (GH18478).
- `Series.plot.kde()` has exposed the args `ind` and `bw_method` in the docstring (GH18461). The argument `ind` may now also be an integer (number of sample points).
- `DataFrame.plot()` now supports multiple columns to the `y` argument (GH19699)

Groupby/resample/rolling

- Bug when grouping by a single column and aggregating with a class like `list` or `tuple` (GH18079)
- Fixed regression in `DataFrame.groupby()` which would not emit an error when called with a tuple key not in the index (GH18798)
- Bug in `DataFrame.resample()` which silently ignored unsupported (or mistyped) options for `label`, `closed` and `convention` (GH19303)
- Bug in `DataFrame.groupby()` where tuples were interpreted as lists of keys rather than as keys (GH17979, GH18249)
- Bug in `DataFrame.groupby()` where aggregation by `first`/`last`/`min`/`max` was causing timestamps to lose precision (GH19526)
- Bug in `DataFrame.transform()` where particular aggregation functions were being incorrectly cast to match the dtype(s) of the grouped data (GH19200)
- Bug in `DataFrame.groupby()` passing the *on*= kwarg, and subsequently using `.apply()` (GH17813)
- Bug in `DataFrame.resample().aggregate` not raising a `KeyError` when aggregating a non-existent column (GH16766, GH19566)
- Bug in `DataFrameGroupBy.cumsum()` and `DataFrameGroupBy.cumprod()` when `skipna` was passed (GH19806)
- Bug in `DataFrame.resample()` that dropped timezone information (GH13238)
- Bug in `DataFrame.groupby()` where transformations using `np.all` and `np.any` were raising a `ValueError` (GH20653)
- Bug in `DataFrame.resample()` where `ffill`, `bfill`, `pad`, `backfill`, `fillna`, `interpolate`, and `asfreq` were ignoring `loffset`. (GH20744)
- Bug in `DataFrame.groupby()` when applying a function that has mixed data types and the user supplied function can fail on the grouping column (GH20949)
- Bug in `DataFrameGroupBy.rolling().apply()` where operations performed against the associated `DataFrameGroupBy` object could impact the inclusion of the grouped item(s) in the result (GH14013)

Sparse

- Bug in which creating a `SparseDataFrame` from a dense `Series` or an unsupported type raised an uncontrolled exception (GH19374)
- Bug in `SparseDataFrame.to_csv` causing exception (GH19384)
- Bug in `SparseSeries.memory_usage` which caused segfault by accessing non sparse elements (GH19368)
- Bug in constructing a `SparseArray`: if `data` is a scalar and `index` is defined it will coerce to `float64` regardless of scalar's dtype. (GH19163)

Reshaping

- Bug in `DataFrame.merge()` where referencing a `CategoricalIndex` by name, where the `by` kwarg would `KeyError` (GH20777)
- Bug in `DataFrame.stack()` which fails trying to sort mixed type levels under Python 3 (GH18310)
- Bug in `DataFrame.unstack()` which casts int to float if `columns` is a `MultiIndex` with unused levels (GH17845)
- Bug in `DataFrame.unstack()` which raises an error if `index` is a `MultiIndex` with unused labels on the unstacked level (GH18562)
- Fixed construction of a `Series` from a `dict` containing `NaN` as key (GH18480)
- Fixed construction of a `DataFrame` from a `dict` containing `NaN` as key (GH18455)
- Disabled construction of a `Series` where len(index) > len(data) = 1, which previously would broadcast the data item, and now raises a `ValueError` (GH18819)
- Suppressed error in the construction of a `DataFrame` from a `dict` containing scalar values when the corresponding keys are not included in the passed index (GH18600)
- Fixed (changed from `object` to `float64`) dtype of `DataFrame` initialized with axes, no data, and `dtype=int` (GH19646)
- Bug in `Series.rank()` where `Series` containing `NaT` modifies the `Series` inplace (GH18521)
- Bug in `cut()` which fails when using readonly arrays (GH18773)
- Bug in `DataFrame.pivot_table()` which fails when the `aggfunc` arg is of type string. The behavior is now consistent with other methods like `agg` and `apply` (GH18713)
- Bug in `DataFrame.merge()` in which merging using `Index` objects as vectors raised an Exception (GH19038)
- Bug in `DataFrame.stack()`, `DataFrame.unstack()`, `Series.unstack()` which were not returning subclasses (GH15563)
- Bug in timezone comparisons, manifesting as a conversion of the index to UTC in `.concat()` (GH18523)
- Bug in `concat()` when concatenating sparse and dense series it returns only a `SparseDataFrame`. Should be a `DataFrame`. (GH18914, GH18686, and GH16874)
- Improved error message for `DataFrame.merge()` when there is no common merge key (GH19427)
- Bug in `DataFrame.join()` which does an `outer` instead of a `left` join when being called with multiple DataFrames and some have non-unique indices (GH19624)
- `Series.rename()` now accepts `axis` as a kwarg (GH18589)
- Bug in `rename()` where an Index of same-length tuples was converted to a MultiIndex (GH19497)
- Comparisons between `Series` and `Index` would return a `Series` with an incorrect name, ignoring the `Index`'s name attribute (GH19582)
- Bug in `qcut()` where datetime and timedelta data with `NaT` present raised a `ValueError` (GH19768)
- Bug in `DataFrame.iterrows()`, which would infers strings not compliant to ISO8601 to datetimes (GH19671)
- Bug in `Series` constructor with `Categorical` where a `ValueError` is not raised when an index of different length is given (GH19342)
- Bug in `DataFrame.astype()` where column metadata is lost when converting to categorical or a dictionary of dtypes (GH19920)

- Bug in `cut()` and `qcut()` where timezone information was dropped (GH19872)
- Bug in `Series` constructor with a `dtype=str`, previously raised in some cases (GH19853)
- Bug in `get_dummies()`, and `select_dtypes()`, where duplicate column names caused incorrect behavior (GH20848)
- Bug in `isna()`, which cannot handle ambiguous typed lists (GH20675)
- Bug in `concat()` which raises an error when concatenating TZ-aware dataframes and all-NaT dataframes (GH12396)
- Bug in `concat()` which raises an error when concatenating empty TZ-aware series (GH18447)

Other

- Improved error message when attempting to use a Python keyword as an identifier in a `numexpr` backed query (GH18221)
- Bug in accessing a `pandas.get_option()`, which raised `KeyError` rather than `OptionError` when looking up a non-existent option key in some cases (GH19789)
- Bug in `testing.assert_series_equal()` and `testing.assert_frame_equal()` for Series or DataFrames with differing unicode data (GH20503)

Contributors

A total of 328 people contributed patches to this release. People with a "+" by their names contributed a patch for the first time.

- Aaron Critchley
- AbdealiJK +
- Adam Hooper +
- Albert Villanova del Moral
- Alejandro Giacometti +
- Alejandro Hohmann +
- Alex Rychyk
- Alexander Buchkovsky
- Alexander Lenail +
- Alexander Michael Schade
- Aly Sivji +
- Andreas Költringer +
- Andrew
- Andrew Bui +
- András Novoszáth +
- Andy Craze +
- Andy R. Terrel
- Anh Le +

- Anil Kumar Pallekonda +
- Antoine Pitrou +
- Antonio Linde +
- Antonio Molina +
- Antonio Quinonez +
- Armin Varshokar +
- Artem Bogachev +
- Avi Sen +
- Azeez Oluwafemi +
- Ben Auffarth +
- Bernhard Thiel +
- Bhavesh Poddar +
- BielStela +
- Blair +
- Bob Haffner
- Brett Naul +
- Brock Mendel
- Bryce Guinta +
- Carlos Eduardo Moreira dos Santos +
- Carlos García Márquez +
- Carol Willing
- Cheuk Ting Ho +
- Chitrank Dixit +
- Chris
- Chris Burr +
- Chris Catalfo +
- Chris Mazzullo
- Christian Chwala +
- Cihan Ceyhan +
- Clemens Brunner
- Colin +
- Cornelius Riemenschneider
- Crystal Gong +
- DaanVanHauwermeiren
- Dan Dixey +
- Daniel Frank +

- Daniel Garrido +
- Daniel Sakuma +
- DataOmbudsman +
- Dave Hirschfeld
- Dave Lewis +
- David Adrián Cañones Castellano +
- David Arcos +
- David C Hall +
- David Fischer
- David Hoese +
- David Lutz +
- David Polo +
- David Stansby
- Dennis Kamau +
- Dillon Niederhut
- Dimitri +
- Dr. Irv
- Dror Atariah
- Eric Chea +
- Eric Kisslinger
- Eric O. LEBIGOT (EOL) +
- FAN-GOD +
- Fabian Retkowski +
- Fer Sar +
- Gabriel de Maeztu +
- Gianpaolo Macario +
- Giftlin Rajaiah
- Gilberto Olimpio +
- Gina +
- Gjelt +
- Graham Inggs +
- Grant Roch
- Grant Smith +
- Grzegorz Konefał +
- Guilherme Beltramini
- HagaiHargil +

- Hamish Pitkeathly +
- Hammad Mashkoor +
- Hannah Ferchland +
- Hans
- Haochen Wu +
- Hissashi Rocha +
- Iain Barr +
- Ibrahim Sharaf ElDen +
- Ignasi Fosch +
- Igor Conrado Alves de Lima +
- Igor Shelvinskyi +
- Imanflow +
- Ingolf Becker
- Israel Saeta Pérez
- Iva Koevska +
- Jakub Nowacki +
- Jan F-F +
- Jan Koch +
- Jan Werkmann
- Janelle Zoutkamp +
- Jason Bandlow +
- Jaume Bonet +
- Jay Alammar +
- Jeff Reback
- JennaVergeynst
- Jimmy Woo +
- Jing Qiang Goh +
- Joachim Wagner +
- Joan Martin Miralles +
- Joel Nothman
- Joeun Park +
- John Cant +
- Johnny Metz +
- Jon Mease
- Jonas Schulze +
- Jongwony +

- Jordi Contestí +
- Joris Van den Bossche
- José F. R. Fonseca +
- Jovixe +
- Julio Martinez +
- Jörg Döpfert
- KOBAYASHI Ittoku +
- Kate Surta +
- Kenneth +
- Kevin Kuhl
- Kevin Sheppard
- Krzysztof Chomski
- Ksenia +
- Ksenia Bobrova +
- Kunal Gosar +
- Kurtis Kerstein +
- Kyle Barron +
- Laksh Arora +
- Laurens Geffert +
- Leif Walsh
- Liam Marshall +
- Liam3851 +
- Licht Takeuchi
- Liudmila +
- Ludovico Russo +
- Mabel Villalba +
- Manan Pal Singh +
- Manraj Singh
- Marc +
- Marc Garcia
- Marco Hemken +
- Maria del Mar Bibiloni +
- Mario Corchero +
- Mark Woodbridge +
- Martin Journois +
- Mason Gallo +

- Matias Heikkilä +
- Matt Braymer-Hayes
- Matt Kirk +
- Matt Maybeno +
- Matthew Kirk +
- Matthew Rocklin +
- Matthew Roeschke
- Matthias Bussonnier +
- Max Mikhaylov +
- Maxim Veksler +
- Maximilian Roos
- Maximiliano Greco +
- Michael Penkov
- Michael Röttger +
- Michael Selik +
- Michael Waskom
- Mie~~~
- Mike Kutzma +
- Ming Li +
- Mitar +
- Mitch Negus +
- Montana Low +
- Moritz Münst +
- Mortada Mehyar
- Myles Braithwaite +
- Nate Yoder
- Nicholas Ursa +
- Nick Chmura
- Nikos Karagiannakis +
- Nipun Sadvilkar +
- Nis Martensen +
- Noah +
- Noémi Éltető +
- Olivier Bilodeau +
- Ondrej Kokes +
- Onno Eberhard +

- Paul Ganssle +
- Paul Mannino +
- Paul Reidy
- Paulo Roberto de Oliveira Castro +
- Pepe Flores +
- Peter Hoffmann
- Phil Ngo +
- Pietro Battiston
- Pranav Suri +
- Priyanka Ojha +
- Pulkit Maloo +
- README Bot +
- Ray Bell +
- Riccardo Magliocchetti +
- Ridhwan Luthra +
- Robert Meyer
- Robin
- Robin Kiplang’at +
- Rohan Pandit +
- Rok Mihevc +
- Rouz Azari
- Ryszard T. Kaleta +
- Sam Cohan
- Sam Foo
- Samir Musali +
- Samuel Sinayoko +
- Sangwoong Yoon
- SarahJessica +
- Sharad Vijalapuram +
- Shubham Chaudhary +
- SiYoungOh +
- Sietse Brouwer
- Simone Basso +
- Stefania Delprete +
- Stefano Cianciulli +
- Stephen Childs +

- StephenVoland +
- Stijn Van Hoey +
- Sven
- Talitha Pumar +
- Tarbo Fukazawa +
- Ted Petrou +
- Thomas A Caswell
- Tim Hoffmann +
- Tim Swast
- Tom Augspurger
- Tommy +
- Tulio Casagrande +
- Tushar Gupta +
- Tushar Mittal +
- Upkar Lidder +
- Victor Villas +
- Vince W +
- Vinícius Figueiredo +
- Vipin Kumar +
- WBare
- Wenhuan +
- Wes Turner
- William Ayd
- Wilson Lin +
- Xbar
- Yaroslav Halchenko
- Yee Mey
- Yeongseon Choe +
- Yian +
- Yimeng Zhang
- ZhuBaohe +
- Zihao Zhao +
- adatasetaday +
- akielbowicz +
- akosel +
- alinde1 +

- amuta +
- bolkedebruin
- cbertinato
- cgohlke
- charlie0389 +
- chris-b1
- csfarkas +
- dajcs +
- deflatSOCO +
- derestle-htwg
- discort
- dmanikowski-reef +
- donK23 +
- elrubio +
- fivemok +
- fjdiod
- fjetter +
- froessler +
- gabrielclow
- gfyoung
- ghasemnaddaf
- h-vetinari +
- himanshu awasthi +
- ignamv +
- jayfoad +
- jazzmuesli +
- jbrockmendel
- jen w +
- jjames34 +
- joaoavf +
- joders +
- jschendel
- juan huguet +
- l736x +
- luzpaz +
- mdeboc +

- miguelmorin +
- miker985
- miquelcamprodon +
- orereta +
- ottiP +
- peterpanmj +
- rafarui +
- raph-m +
- readyready15728 +
- rmihael +
- samghelms +
- scriptomation +
- sfoo +
- stefansimik +
- stonebig
- tmnhat2001 +
- tomneep +
- topper-123
- tv3141 +
- verakai +
- xpvpc +
- zhanghui +

5.5 Version 0.22

5.5.1 v0.22.0 (December 29, 2017)

This is a major release from 0.21.1 and includes a single, API-breaking change. We recommend that all users upgrade to this version after carefully reading the release note (singular!).

Backwards incompatible API changes

Pandas 0.22.0 changes the handling of empty and all-*NA* sums and products. The summary is that

- The sum of an empty or all-*NA* `Series` is now `0`
- The product of an empty or all-*NA* `Series` is now `1`
- We've added a `min_count` parameter to `.sum()` and `.prod()` controlling the minimum number of valid values for the result to be valid. If fewer than `min_count` non-*NA* values are present, the result is *NA*. The default is `0`. To return `NaN`, the 0.21 behavior, use `min_count=1`.

Some background: In pandas 0.21, we fixed a long-standing inconsistency in the return value of all-*NA* series depending on whether or not bottleneck was installed. See *Sum/Prod of all-NaN or empty Series/DataFrames is now consistently NaN*. At the same time, we changed the sum and prod of an empty `Series` to also be `NaN`.

Based on feedback, we've partially reverted those changes.

Arithmetic operations

The default sum for empty or all-*NA* `Series` is now `0`.

pandas 0.21.x

```
In [1]: pd.Series([]).sum()
Out[1]: nan

In [2]: pd.Series([np.nan]).sum()
Out[2]: nan
```

pandas 0.22.0

```
In [1]: pd.Series([]).sum()
Out[1]: 0.0

In [2]: pd.Series([np.nan]).sum()
Out[2]: 0.0
```

The default behavior is the same as pandas 0.20.3 with bottleneck installed. It also matches the behavior of NumPy's `np.nansum` on empty and all-*NA* arrays.

To have the sum of an empty series return `NaN` (the default behavior of pandas 0.20.3 without bottleneck, or pandas 0.21.x), use the `min_count` keyword.

```
In [3]: pd.Series([]).sum(min_count=1)
Out[3]: nan
```

Thanks to the `skipna` parameter, the `.sum` on an all-*NA* series is conceptually the same as the `.sum` of an empty one with `skipna=True` (the default).

```
In [4]: pd.Series([np.nan]).sum(min_count=1)  # skipna=True by default
Out[4]: nan
```

The `min_count` parameter refers to the minimum number of *non-null* values required for a non-NA sum or product.

`Series.prod()` has been updated to behave the same as *`Series.sum()`*, returning `1` instead.

```
In [5]: pd.Series([]).prod()
Out[5]: 1.0

In [6]: pd.Series([np.nan]).prod()
Out[6]: 1.0

In [7]: pd.Series([]).prod(min_count=1)
Out[7]: nan
```

These changes affect *`DataFrame.sum()`* and *`DataFrame.prod()`* as well. Finally, a few less obvious places in pandas are affected by this change.

Grouping by a categorical

Grouping by a `Categorical` and summing now returns `0` instead of `NaN` for categories with no observations. The product now returns `1` instead of `NaN`.

pandas 0.21.x

```
In [8]: grouper = pd.Categorical(['a', 'a'], categories=['a', 'b'])

In [9]: pd.Series([1, 2]).groupby(grouper).sum()
Out[9]:
a    3.0
b    NaN
dtype: float64
```

pandas 0.22

```
In [8]: grouper = pd.Categorical(['a', 'a'], categories=['a', 'b'])

In [9]: pd.Series([1, 2]).groupby(grouper).sum()
Out[9]:
a    3
b    0
Length: 2, dtype: int64
```

To restore the 0.21 behavior of returning `NaN` for unobserved groups, use `min_count>=1`.

```
In [10]: pd.Series([1, 2]).groupby(grouper).sum(min_count=1)
Out[10]:
a    3.0
b    NaN
Length: 2, dtype: float64
```

Resample

The sum and product of all-*NA* bins has changed from `NaN` to `0` for sum and `1` for product.

pandas 0.21.x

```
In [11]: s = pd.Series([1, 1, np.nan, np.nan],
   ....:               index=pd.date_range('2017', periods=4))
   ....: s
Out[11]:
2017-01-01    1.0
2017-01-02    1.0
2017-01-03    NaN
2017-01-04    NaN
Freq: D, dtype: float64

In [12]: s.resample('2d').sum()
Out[12]:
2017-01-01    2.0
2017-01-03    NaN
Freq: 2D, dtype: float64
```

pandas 0.22.0

```
In [11]: s = pd.Series([1, 1, np.nan, np.nan],
   ....:               index=pd.date_range('2017', periods=4))
   ....:

In [12]: s.resample('2d').sum()
Out[12]:
2017-01-01    2.0
2017-01-03    0.0
Freq: 2D, Length: 2, dtype: float64
```

To restore the 0.21 behavior of returning `NaN`, use `min_count>=1`.

```
In [13]: s.resample('2d').sum(min_count=1)
Out[13]:
2017-01-01    2.0
2017-01-03    NaN
Freq: 2D, Length: 2, dtype: float64
```

In particular, upsampling and taking the sum or product is affected, as upsampling introduces missing values even if the original series was entirely valid.

pandas 0.21.x

```
In [14]: idx = pd.DatetimeIndex(['2017-01-01', '2017-01-02'])

In [15]: pd.Series([1, 2], index=idx).resample('12H').sum()
Out[15]:
2017-01-01 00:00:00    1.0
2017-01-01 12:00:00    NaN
2017-01-02 00:00:00    2.0
Freq: 12H, dtype: float64
```

pandas 0.22.0

```
In [14]: idx = pd.DatetimeIndex(['2017-01-01', '2017-01-02'])

In [15]: pd.Series([1, 2], index=idx).resample("12H").sum()
Out[15]:
2017-01-01 00:00:00    1
2017-01-01 12:00:00    0
2017-01-02 00:00:00    2
Freq: 12H, Length: 3, dtype: int64
```

Once again, the `min_count` keyword is available to restore the 0.21 behavior.

```
In [16]: pd.Series([1, 2], index=idx).resample("12H").sum(min_count=1)
Out[16]:
2017-01-01 00:00:00    1.0
2017-01-01 12:00:00    NaN
2017-01-02 00:00:00    2.0
Freq: 12H, Length: 3, dtype: float64
```

Rolling and expanding

Rolling and expanding already have a `min_periods` keyword that behaves similar to `min_count`. The only case that changes is when doing a rolling or expanding sum with `min_periods=0`. Previously this returned `NaN`, when fewer than `min_periods` non-*NA* values were in the window. Now it returns `0`.

pandas 0.21.1

```
In [17]: s = pd.Series([np.nan, np.nan])

In [18]: s.rolling(2, min_periods=0).sum()
Out[18]:
0   NaN
1   NaN
dtype: float64
```

pandas 0.22.0

```
In [17]: s = pd.Series([np.nan, np.nan])

In [18]: s.rolling(2, min_periods=0).sum()
Out[18]:
0    0.0
1    0.0
Length: 2, dtype: float64
```

The default behavior of `min_periods=None`, implying that `min_periods` equals the window size, is unchanged.

Compatibility

If you maintain a library that should work across pandas versions, it may be easiest to exclude pandas 0.21 from your requirements. Otherwise, all your `sum()` calls would need to check if the `Series` is empty before summing.

With setuptools, in your `setup.py` use:

```
install_requires=['pandas!=0.21.*', ...]
```

With conda, use

```
requirements:
  run:
    - pandas !=0.21.0,!=0.21.1
```

Note that the inconsistency in the return value for all-*NA* series is still there for pandas 0.20.3 and earlier. Avoiding pandas 0.21 will only help with the empty case.

Contributors

A total of 1 people contributed patches to this release. People with a "+" by their names contributed a patch for the first time.

- Tom Augspurger

5.6 Version 0.21

5.6.1 v0.21.1 (December 12, 2017)

This is a minor bug-fix release in the 0.21.x series and includes some small regression fixes, bug fixes and performance improvements. We recommend that all users upgrade to this version.

Highlights include:

- Temporarily restore matplotlib datetime plotting functionality. This should resolve issues for users who implicitly relied on pandas to plot datetimes with matplotlib. See *here*.
- Improvements to the Parquet IO functions introduced in 0.21.0. See *here*.

What's new in v0.21.1

Restore Matplotlib datetime converter registration

Pandas implements some matplotlib converters for nicely formatting the axis labels on plots with `datetime` or `Period` values. Prior to pandas 0.21.0, these were implicitly registered with matplotlib, as a side effect of `import pandas`.

In pandas 0.21.0, we required users to explicitly register the converter. This caused problems for some users who relied on those converters being present for regular `matplotlib.pyplot` plotting methods, so we're temporarily reverting that change; pandas 0.21.1 again registers the converters on import, just like before 0.21.0.

We've added a new option to control the converters: `pd.options.plotting.matplotlib.register_converters`. By default, they are registered. Toggling this to `False` removes pandas' formatters and restore any converters we overwrote when registering them (GH18301).

We're working with the matplotlib developers to make this easier. We're trying to balance user convenience (automatically registering the converters) with import performance and best practices (importing pandas shouldn't have the side effect of overwriting any custom converters you've already set). In the future we hope to have most of the datetime formatting functionality in matplotlib, with just the pandas-specific converters in pandas. We'll then gracefully deprecate the automatic registration of converters in favor of users explicitly registering them when they want them.

New features

Improvements to the Parquet IO functionality

- `DataFrame.to_parquet()` will now write non-default indexes when the underlying engine supports it. The indexes will be preserved when reading back in with `read_parquet()` (GH18581).
- `read_parquet()` now allows to specify the columns to read from a parquet file (GH18154)
- `read_parquet()` now allows to specify kwargs which are passed to the respective engine (GH18216)

Other enhancements

- `Timestamp.timestamp()` is now available in Python 2.7. (GH17329)
- `Grouper` and `TimeGrouper` now have a friendly repr output (GH18203).

Deprecations

- `pandas.tseries.register` has been renamed to `pandas.plotting.register_matplotlib_converters()` (GH18301)

Performance improvements

- Improved performance of plotting large series/dataframes (GH18236).

Bug fixes

Conversion

- Bug in *TimedeltaIndex* subtraction could incorrectly overflow when NaT is present (GH17791)
- Bug in *DatetimeIndex* subtracting datetimelike from DatetimeIndex could fail to overflow (GH18020)
- Bug in IntervalIndex.copy() when copying and IntervalIndex with non-default closed (GH18339)
- Bug in *DataFrame.to_dict()* where columns of datetime that are tz-aware were not converted to required arrays when used with orient='records', raising TypeError (GH18372)
- Bug in DateTimeIndex and *date_range()* where mismatching tz-aware start and end timezones would not raise an err if end.tzinfo is None (GH18431)
- Bug in *Series.fillna()* which raised when passed a long integer on Python 2 (GH18159).

Indexing

- Bug in a boolean comparison of a datetime.datetime and a datetime64[ns] dtype Series (GH17965)
- Bug where a MultiIndex with more than a million records was not raising AttributeError when trying to access a missing attribute (GH18165)
- Bug in *IntervalIndex* constructor when a list of intervals is passed with non-default closed (GH18334)
- Bug in Index.putmask when an invalid mask passed (GH18368)
- Bug in masked assignment of a timedelta64[ns] dtype Series, incorrectly coerced to float (GH18493)

I/O

- Bug in class:~*pandas.io.stata.StataReader* not converting date/time columns with display formatting addressed (GH17990). Previously columns with display formatting were normally left as ordinal numbers and not converted to datetime objects.
- Bug in *read_csv()* when reading a compressed UTF-16 encoded file (GH18071)
- Bug in *read_csv()* for handling null values in index columns when specifying na_filter=False (GH5239)
- Bug in *read_csv()* when reading numeric category fields with high cardinality (GH18186)
- Bug in *DataFrame.to_csv()* when the table had MultiIndex columns, and a list of strings was passed in for header (GH5539)
- Bug in parsing integer datetime-like columns with specified format in read_sql (GH17855).
- Bug in DataFrame.to_msgpack() when serializing data of the numpy.bool_ datatype (GH18390)
- Bug in *read_json()* not decoding when reading line delimited JSON from S3 (GH17200)
- Bug in pandas.io.json.json_normalize() to avoid modification of meta (GH18610)
- Bug in to_latex() where repeated MultiIndex values were not printed even though a higher level index differed from the previous row (GH14484)
- Bug when reading NaN-only categorical columns in HDFStore (GH18413)

- Bug in *DataFrame.to_latex()* with `longtable=True` where a latex multicolumn always spanned over three columns (GH17959)

Plotting

- Bug in `DataFrame.plot()` and `Series.plot()` with *DatetimeIndex* where a figure generated by them is not pickleable in Python 3 (GH18439)

Groupby/resample/rolling

- Bug in `DataFrame.resample(...).apply(...)` when there is a callable that returns different columns (GH15169)
- Bug in `DataFrame.resample(...)` when there is a time change (DST) and resampling frequency is 12h or higher (GH15549)
- Bug in `pd.DataFrameGroupBy.count()` when counting over a datetimelike column (GH13393)
- Bug in `rolling.var` where calculation is inaccurate with a zero-valued array (GH18430)

Reshaping

- Error message in `pd.merge_asof()` for key datatype mismatch now includes datatype of left and right key (GH18068)
- Bug in `pd.concat` when empty and non-empty DataFrames or Series are concatenated (GH18178 GH18187)
- Bug in `DataFrame.filter(...)` when `unicode` is passed as a condition in Python 2 (GH13101)
- Bug when merging empty DataFrames when `np.seterr(divide='raise')` is set (GH17776)

Numeric

- Bug in `pd.Series.rolling.skew()` and `rolling.kurt()` with all equal values has floating issue (GH18044)

Categorical

- Bug in *DataFrame.astype()* where casting to 'category' on an empty `DataFrame` causes a segmentation fault (GH18004)
- Error messages in the testing module have been improved when items have different `CategoricalDtype` (GH18069)
- `CategoricalIndex` can now correctly take a `pd.api.types.CategoricalDtype` as its dtype (GH18116)
- Bug in `Categorical.unique()` returning read-only `codes` array when all categories were `NaN` (GH18051)
- Bug in `DataFrame.groupby(axis=1)` with a `CategoricalIndex` (GH18432)