- Bug in *read_csv()* when called with a single-element list `header` would return a `DataFrame` of all NaN values (GH7757)
- Bug in *DataFrame.to_csv()* defaulting to 'ascii' encoding in Python 3, instead of 'utf-8' (GH17097)
- Bug in *read_stata()* where value labels could not be read when using an iterator (GH16923)
- Bug in *read_stata()* where the index was not set (GH16342)
- Bug in *read_html()* where import check fails when run in multiple threads (GH16928)
- Bug in *read_csv()* where automatic delimiter detection caused a `TypeError` to be thrown when a bad line was encountered rather than the correct error message (GH13374)
- Bug in *DataFrame.to_html()* with `notebook=True` where DataFrames with named indices or non-MultiIndex indices had undesired horizontal or vertical alignment for column or row labels, respectively (GH16792)
- Bug in *DataFrame.to_html()* in which there was no validation of the `justify` parameter (GH17527)
- Bug in *HDFStore.select()* when reading a contiguous mixed-data table featuring VLArray (GH17021)
- Bug in `to_json()` where several conditions (including objects with unprintable symbols, objects with deep recursion, overlong labels) caused segfaults instead of raising the appropriate exception (GH14256)

Plotting

- Bug in plotting methods using `secondary_y` and `fontsize` not setting secondary axis font size (GH12565)
- Bug when plotting `timedelta` and `datetime` dtypes on y-axis (GH16953)
- Line plots no longer assume monotonic x data when calculating xlims, they show the entire lines now even for unsorted x data. (GH11310, GH11471)
- With matplotlib 2.0.0 and above, calculation of x limits for line plots is left to matplotlib, so that its new default settings are applied. (GH15495)
- Bug in `Series.plot.bar` or `DataFrame.plot.bar` with `y` not respecting user-passed `color` (GH16822)
- Bug causing `plotting.parallel_coordinates` to reset the random seed when using random colors (GH17525)

Groupby/resample/rolling

- Bug in `DataFrame.resample(...).size()` where an empty `DataFrame` did not return a `Series` (GH14962)
- Bug in *infer_freq()* causing indices with 2-day gaps during the working week to be wrongly inferred as business daily (GH16624)
- Bug in `.rolling(...).quantile()` which incorrectly used different defaults than *Series.quantile()* and *DataFrame.quantile()* (GH9413, GH16211)
- Bug in `groupby.transform()` that would coerce boolean dtypes back to float (GH16875)
- Bug in `Series.resample(...).apply()` where an empty `Series` modified the source index and did not return the name of a `Series` (GH14313)
- Bug in `.rolling(...).apply(...)` with a `DataFrame` with a `DatetimeIndex`, a `window` of a timedelta-convertible and `min_periods >= 1` (GH15305)

- Bug in `DataFrame.groupby` where index and column keys were not recognized correctly when the number of keys equaled the number of elements on the groupby axis (GH16859)
- Bug in `groupby.nunique()` with `TimeGrouper` which cannot handle `NaT` correctly (GH17575)
- Bug in `DataFrame.groupby` where a single level selection from a `MultiIndex` unexpectedly sorts (GH17537)
- Bug in `DataFrame.groupby` where spurious warning is raised when `Grouper` object is used to override ambiguous column name (GH17383)
- Bug in `TimeGrouper` differs when passes as a list and as a scalar (GH17530)

Sparse

- Bug in `SparseSeries` raises `AttributeError` when a dictionary is passed in as data (GH16905)
- Bug in `SparseDataFrame.fillna()` not filling all NaNs when frame was instantiated from SciPy sparse matrix (GH16112)
- Bug in `SparseSeries.unstack()` and `SparseDataFrame.stack()` (GH16614, GH15045)
- Bug in `make_sparse()` treating two numeric/boolean data, which have same bits, as same when array `dtype` is `object` (GH17574)
- `SparseArray.all()` and `SparseArray.any()` are now implemented to handle `SparseArray`, these were used but not implemented (GH17570)

Reshaping

- Joining/Merging with a non unique `PeriodIndex` raised a `TypeError` (GH16871)
- Bug in *`crosstab()`* where non-aligned series of integers were casted to float (GH17005)
- Bug in merging with categorical dtypes with datetimelikes incorrectly raised a `TypeError` (GH16900)
- Bug when using `isin()` on a large object series and large comparison array (GH16012)
- Fixes regression from 0.20, *`Series.aggregate()`* and *`DataFrame.aggregate()`* allow dictionaries as return values again (GH16741)
- Fixes dtype of result with integer dtype input, from *`pivot_table()`* when called with `margins=True` (GH17013)
- Bug in *`crosstab()`* where passing two `Series` with the same name raised a `KeyError` (GH13279)
- *`Series.argmin()`*, *`Series.argmax()`*, and their counterparts on `DataFrame` and groupby objects work correctly with floating point data that contains infinite values (GH13595).
- Bug in *`unique()`* where checking a tuple of strings raised a `TypeError` (GH17108)
- Bug in *`concat()`* where order of result index was unpredictable if it contained non-comparable elements (GH17344)
- Fixes regression when sorting by multiple columns on a `datetime64` dtype `Series` with `NaT` values (GH16836)
- Bug in *`pivot_table()`* where the result's columns did not preserve the categorical dtype of `columns` when `dropna` was `False` (GH17842)
- Bug in `DataFrame.drop_duplicates` where dropping with non-unique column names raised a `ValueError` (GH17836)

- Bug in `unstack()` which, when called on a list of levels, would discard the `fillna` argument (GH13971)
- Bug in the alignment of `range` objects and other list-likes with `DataFrame` leading to operations being performed row-wise instead of column-wise (GH17901)

Numeric

- Bug in `.clip()` with `axis=1` and a list-like for `threshold` is passed; previously this raised `ValueError` (GH15390)
- *`Series.clip()`* and *`DataFrame.clip()`* now treat NA values for upper and lower arguments as `None` instead of raising `ValueError` (GH17276).

Categorical

- Bug in *`Series.isin()`* when called with a categorical (GH16639)
- Bug in the categorical constructor with empty values and categories causing the `.categories` to be an empty `Float64Index` rather than an empty `Index` with object dtype (GH17248)
- Bug in categorical operations with *Series.cat* not preserving the original Series' name (GH17509)
- Bug in *`DataFrame.merge()`* failing for categorical columns with boolean/int data types (GH17187)
- Bug in constructing a `Categorical`/`CategoricalDtype` when the specified `categories` are of categorical type (GH17884).

PyPy

- Compatibility with PyPy in *`read_csv()`* with `usecols=[<unsorted ints>]` and *`read_json()`* (GH17351)
- Split tests into cases for CPython and PyPy where needed, which highlights the fragility of index matching with `float('nan')`, `np.nan` and `NAT` (GH17351)
- Fix *`DataFrame.memory_usage()`* to support PyPy. Objects on PyPy do not have a fixed size, so an approximation is used instead (GH17228)

Other

- Bug where some inplace operators were not being wrapped and produced a copy when invoked (GH12962)
- Bug in *`eval()`* where the `inplace` parameter was being incorrectly handled (GH16732)

Contributors

A total of 206 people contributed patches to this release. People with a "+" by their names contributed a patch for the first time.

- 3553x +
- Aaron Barber
- Adam Gleave +
- Adam Smith +

- AdamShamlian +
- Adrian Liaw +
- Alan Velasco +
- Alan Yee +
- Alex B +
- Alex Lubbock +
- Alex Marchenko +
- Alex Rychyk +
- Amol K +
- Andreas Winkler
- Andrew +
- Andrew
- André Jonasson +
- Becky Sweger
- Berkay +
- Bob Haffner +
- Bran Yang
- Brian Tu +
- Brock Mendel +
- Carol Willing +
- Carter Green +
- Chankey Pathak +
- Chris
- Chris Billington
- Chris Filo Gorgolewski +
- Chris Kerr
- Chris M +
- Chris Mazzullo +
- Christian Prinoth
- Christian Stade-Schuldt
- Christoph Moehl +
- DSM
- Daniel Chen +
- Daniel Grady
- Daniel Himmelstein
- Dave Willmer

- David Cook
- David Gwynne
- David Read +
- Dillon Niederhut +
- Douglas Rudd
- Eric Stein +
- Eric Wieser +
- Erik Fredriksen
- Florian Wilhelm +
- Floris Kint +
- Forbidden Donut
- Gabe F +
- Giftlin +
- Giftlin Rajaiah +
- Giulio Pepe +
- Guilherme Beltramini
- Guillem Borrell +
- Hanmin Qin +
- Hendrik Makait +
- Hugues Valois
- Hussain Tamboli +
- Iva Miholic +
- Jan Novotný +
- Jan Rudolph
- Jean Helie +
- Jean-Baptiste Schiratti +
- Jean-Mathieu Deschenes
- Jeff Knupp +
- Jeff Reback
- Jeff Tratner
- JennaVergeynst
- JimStearns206
- Joel Nothman
- John W. O'Brien
- Jon Crall +
- Jon Mease

- Jonathan J. Helmus +
- Joris Van den Bossche
- JosephWagner
- Juarez Bochi
- Julian Kuhlmann +
- Karel De Brabandere
- Kassandra Keeton +
- Keiron Pizzey +
- Keith Webber
- Kernc
- Kevin Sheppard
- Kirk Hansen +
- Licht Takeuchi +
- Lucas Kushner +
- Mahdi Ben Jelloul +
- Makarov Andrey +
- Malgorzata Turzanska +
- Marc Garcia +
- Margaret Sy +
- MarsGuy +
- Matt Bark +
- Matthew Roeschke
- Matti Picus
- Mehmet Ali “Mali” Akmanalp
- Michael Gasvoda +
- Michael Penkov +
- Milo +
- Morgan Stuart +
- Morgan243 +
- Nathan Ford +
- Nick Eubank
- Nick Garvey +
- Oleg Shteynbuk +
- P-Tillmann +
- Pankaj Pandey
- Patrick Luo

- Patrick O'Melveny
- Paul Reidy +
- Paula +
- Peter Quackenbush
- Peter Yanovich +
- Phillip Cloud
- Pierre Haessig
- Pietro Battiston
- Pradyumna Reddy Chinthala
- Prasanjit Prakash
- RobinFiveWords
- Ryan Hendrickson
- Sam Foo
- Sangwoong Yoon +
- Simon Gibbons +
- SimonBaron
- Steven Cutting +
- Sudeep +
- Sylvia +
- T N +
- Telt
- Thomas A Caswell
- Tim Swast +
- Tom Augspurger
- Tong SHEN
- Tuan +
- Utkarsh Upadhyay +
- Vincent La +
- Vivek +
- WANG Aiyong
- WBare
- Wes McKinney
- XF +
- Yi Liu +
- Yosuke Nakabayashi +
- aaron315 +

- abarber4gh +
- aernlund +
- agustín méndez +
- andymaheshw +
- ante328 +
- aviolov +
- bpraggastis
- cbertinato +
- cclauss +
- chernrick
- chris-b1
- dkamm +
- dwkenefick
- economy
- faic +
- fding253 +
- gfyoung
- guygoldberg +
- hhuuggoo +
- huashuai +
- ian
- iulia +
- jaredsnyder
- jbrockmendel +
- jdeschenes
- jebob +
- jschendel +
- keitakurita
- kernc +
- kiwirob +
- kjford
- linebp
- lloydkirk
- louispotok +
- majiang +
- manikbhandari +

- matthiashuschle +
- mattip
- maxwasserman +
- mjlove12 +
- nmartensen +
- pandas-docs-bot +
- parchd-1 +
- philipphanemann +
- rdk1024 +
- reidy-p +
- ri938
- ruiann +
- rvernica +
- s-weigand +
- scotthavard92 +
- skwbc +
- step4me +
- tobycheese +
- topper-123 +
- tsdlovell
- ysau +
- zzgao +

5.7 Version 0.20

5.7.1 v0.20.3 (July 7, 2017)

This is a minor bug-fix release in the 0.20.x series and includes some small regression fixes and bug fixes. We recommend that all users upgrade to this version.

What's new in v0.20.3

- *Bug fixes*
  - *Conversion*
  - *Indexing*
  - *I/O*
  - *Plotting*
  - *Reshaping*

Bug fixes

- Fixed a bug in failing to compute rolling computations of a column-MultiIndexed `DataFrame` (GH16789, GH16825)
- Fixed a pytest marker failing downstream packages' tests suites (GH16680)

Conversion

- Bug in pickle compat prior to the v0.20.x series, when `UTC` is a timezone in a Series/DataFrame/Index (GH16608)
- Bug in `Series` construction when passing a `Series` with `dtype='category'` (GH16524).
- Bug in *`DataFrame.astype()`* when passing a `Series` as the `dtype` kwarg. (GH16717).

Indexing

- Bug in `Float64Index` causing an empty array instead of `None` to be returned from `.get(np.nan)` on a Series whose index did not contain any `NaN` s (GH8569)
- Bug in `MultiIndex.isin` causing an error when passing an empty iterable (GH16777)
- Fixed a bug in a slicing DataFrame/Series that have a `TimedeltaIndex` (GH16637)

I/O

- Bug in *`read_csv()`* in which files weren't opened as binary files by the C engine on Windows, causing EOF characters mid-field, which would fail (GH16039, GH16559, GH16675)
- Bug in *`read_hdf()`* in which reading a `Series` saved to an HDF file in 'fixed' format fails when an explicit `mode='r'` argument is supplied (GH16583)
- Bug in *`DataFrame.to_latex()`* where `bold_rows` was wrongly specified to be `True` by default, whereas in reality row labels remained non-bold whatever parameter provided. (GH16707)
- Fixed an issue with *`DataFrame.style()`* where generated element ids were not unique (GH16780)
- Fixed loading a `DataFrame` with a `PeriodIndex`, from a `format='fixed'` HDFStore, in Python 3, that was written in Python 2 (GH16781)

Plotting

- Fixed regression that prevented RGB and RGBA tuples from being used as color arguments (GH16233)
- Fixed an issue with *DataFrame.plot.scatter()* that incorrectly raised a `KeyError` when categorical data is used for plotting (GH16199)

Reshaping

- `PeriodIndex` / `TimedeltaIndex.join` was missing the `sort=` kwarg (GH16541)
- Bug in joining on a `MultiIndex` with a `category` dtype for a level (GH16627).
- Bug in *merge()* when merging/joining with multiple categorical columns (GH16767)

Categorical

- Bug in `DataFrame.sort_values` not respecting the `kind` parameter with categorical data (GH16793)

Contributors

A total of 20 people contributed patches to this release. People with a "+" by their names contributed a patch for the first time.

- Bran Yang
- Chris
- Chris Kerr +
- DSM
- David Gwynne
- Douglas Rudd
- Forbidden Donut +
- Jeff Reback
- Joris Van den Bossche
- Karel De Brabandere +
- Peter Quackenbush +
- Pradyumna Reddy Chinthala +
- Telt +
- Tom Augspurger
- chris-b1
- gfyoung
- ian +
- jdeschenes +
- kjford +
- ri938 +

5.7.2 v0.20.2 (June 4, 2017)

This is a minor bug-fix release in the 0.20.x series and includes some small regression fixes, bug fixes and performance improvements. We recommend that all users upgrade to this version.

What's new in v0.20.2

Enhancements

- Unblocked access to additional compression types supported in pytables: 'blosc:blosclz, 'blosc:lz4', 'blosc:lz4hc', 'blosc:snappy', 'blosc:zlib', 'blosc:zstd' (GH14478)
- `Series` provides a `to_latex` method (GH16180)
- A new groupby method *`ngroup()`*, parallel to the existing *`cumcount()`*, has been added to return the group order (GH11642); see *here*.

Performance improvements

- Performance regression fix when indexing with a list-like (GH16285)
- Performance regression fix for MultiIndexes (GH16319, GH16346)
- Improved performance of `.clip()` with scalar arguments (GH15400)
- Improved performance of groupby with categorical groupers (GH16413)
- Improved performance of `MultiIndex.remove_unused_levels()` (GH16556)

Bug fixes

- Silenced a warning on some Windows environments about "tput: terminal attributes: No such device or address" when detecting the terminal size. This fix only applies to python 3 (GH16496)
- Bug in using `pathlib.Path` or `py.path.local` objects with io functions (GH16291)
- Bug in `Index.symmetric_difference()` on two equal MultiIndex's, results in a `TypeError` (GH13490)
- Bug in `DataFrame.update()` with `overwrite=False` and `NaN values` (GH15593)
- Passing an invalid engine to *`read_csv()`* now raises an informative `ValueError` rather than `UnboundLocalError`. (GH16511)
- Bug in *`unique()`* on an array of tuples (GH16519)
- Bug in *`cut()`* when `labels` are set, resulting in incorrect label ordering (GH16459)
- Fixed a compatibility issue with IPython 6.0's tab completion showing deprecation warnings on `Categoricals` (GH16409)

Conversion

- Bug in *`to_numeric()`* in which empty data inputs were causing a segfault of the interpreter (GH16302)
- Silence numpy warnings when broadcasting `DataFrame` to `Series` with comparison ops (GH16378, GH16306)

Indexing

- Bug in `DataFrame.reset_index(level=)` with single level index (GH16263)
- Bug in partial string indexing with a monotonic, but not strictly-monotonic, index incorrectly reversing the slice bounds (GH16515)
- Bug in `MultiIndex.remove_unused_levels()` that would not return a `MultiIndex` equal to the original. (GH16556)

I/O

- Bug in *`read_csv()`* when `comment` is passed in a space delimited text file (GH16472)
- Bug in *`read_csv()`* not raising an exception with nonexistent columns in `usecols` when it had the correct length (GH14671)
- Bug that would force importing of the clipboard routines unnecessarily, potentially causing an import error on startup (GH16288)
- Bug that raised `IndexError` when HTML-rendering an empty `DataFrame` (GH15953)
- Bug in *`read_csv()`* in which tarfile object inputs were raising an error in Python 2.x for the C engine (GH16530)
- Bug where `DataFrame.to_html()` ignored the `index_names` parameter (GH16493)
- Bug where `pd.read_hdf()` returns numpy strings for index names (GH13492)
- Bug in `HDFStore.select_as_multiple()` where start/stop arguments were not respected (GH16209)

Plotting

- Bug in `DataFrame.plot` with a single column and a list-like `color` (GH3486)
- Bug in `plot` where `NaT` in `DatetimeIndex` results in `Timestamp.min` (GH12405)
- Bug in `DataFrame.boxplot` where `figsize` keyword was not respected for non-grouped boxplots (GH11959)

Groupby/resample/rolling

- Bug in creating a time-based rolling window on an empty `DataFrame` (GH15819)
- Bug in `rolling.cov()` with offset window (GH16058)
- Bug in `.resample()` and `.groupby()` when aggregating on integers (GH16361)

Sparse

- Bug in construction of `SparseDataFrame` from `scipy.sparse.dok_matrix` (GH16179)

Reshaping

- Bug in `DataFrame.stack` with unsorted levels in `MultiIndex` columns (GH16323)
- Bug in `pd.wide_to_long()` where no error was raised when `i` was not a unique identifier (GH16382)
- Bug in `Series.isin(..)` with a list of tuples (GH16394)
- Bug in construction of a `DataFrame` with mixed dtypes including an all-NaT column. (GH16395)
- Bug in `DataFrame.agg()` and `Series.agg()` with aggregating on non-callable attributes (GH16405)

Numeric

- Bug in `.interpolate()`, where `limit_direction` was not respected when `limit=None` (default) was passed (GH16282)

Categorical

- Fixed comparison operations considering the order of the categories when both categoricals are unordered (GH16014)

Other

- Bug in `DataFrame.drop()` with an empty-list with non-unique indices (GH16270)

Contributors

A total of 34 people contributed patches to this release. People with a "+" by their names contributed a patch for the first time.

- Aaron Barber +
- Andrew +
- Becky Sweger +
- Christian Prinoth +
- Christian Stade-Schuldt +
- DSM
- Erik Fredriksen +
- Hugues Valois +
- Jeff Reback
- Jeff Tratner
- JimStearns206 +
- John W. O'Brien
- Joris Van den Bossche
- JosephWagner +
- Keith Webber +
- Mehmet Ali "Mali" Akmanalp +
- Pankaj Pandey
- Patrick Luo +
- Patrick O'Melveny +
- Pietro Battiston
- RobinFiveWords +
- Ryan Hendrickson +
- SimonBaron +
- Tom Augspurger
- WBare +
- bpraggastis +
- chernrick +
- chris-b1
- economy +
- gfyoung

- jaredsnyder +
- keitakurita +
- linebp
- lloydkirk +

5.7.3 v0.20.1 (May 5, 2017)

This is a major release from 0.19.2 and includes a number of API changes, deprecations, new features, enhancements, and performance improvements along with a large number of bug fixes. We recommend that all users upgrade to this version.

Highlights include:

- New `.agg()` API for Series/DataFrame similar to the groupby-rolling-resample API's, see *here*
- Integration with the `feather-format`, including a new top-level `pd.read_feather()` and `DataFrame.to_feather()` method, see *here*.
- The `.ix` indexer has been deprecated, see *here*
- `Panel` has been deprecated, see *here*
- Addition of an `IntervalIndex` and `Interval` scalar type, see *here*
- Improved user API when grouping by index levels in `.groupby()`, see *here*
- Improved support for `UInt64` dtypes, see *here*
- A new orient for JSON serialization, `orient='table'`, that uses the Table Schema spec and that gives the possibility for a more interactive repr in the Jupyter Notebook, see *here*
- Experimental support for exporting styled DataFrames (`DataFrame.style`) to Excel, see *here*
- Window binary corr/cov operations now return a MultiIndexed `DataFrame` rather than a `Panel`, as `Panel` is now deprecated, see *here*
- Support for S3 handling now uses `s3fs`, see *here*
- Google BigQuery support now uses the `pandas-gbq` library, see *here*

Warning: Pandas has changed the internal structure and layout of the code base. This can affect imports that are not from the top-level `pandas.*` namespace, please see the changes *here*.

Check the *API Changes* and *deprecations* before updating.

Note: This is a combined release for 0.20.0 and 0.20.1. Version 0.20.1 contains one additional change for backwards-compatibility with downstream projects using pandas' `utils` routines. (GH16250)

What's new in v0.20.0

- *New features*
  - *`agg` API for DataFrame/Series*
  - *`dtype` keyword for data IO*

New features

`agg` API for DataFrame/Series

Series & DataFrame have been enhanced to support the aggregation API. This is a familiar API from groupby, window operations, and resampling. This allows aggregation operations in a concise way by using *`agg()`* and *`transform()`*. The full documentation is *here* (GH1623).

Here is a sample

```
In [1]: df = pd.DataFrame(np.random.randn(10, 3), columns=['A', 'B', 'C'],
   ...:                   index=pd.date_range('1/1/2000', periods=10))
   ...:

In [2]: df.iloc[3:7] = np.nan

In [3]: df
Out[3]:
                   A         B         C
2000-01-01  0.469112 -0.282863 -1.509059
2000-01-02 -1.135632  1.212112 -0.173215
2000-01-03  0.119209 -1.044236 -0.861849
2000-01-04       NaN       NaN       NaN
2000-01-05       NaN       NaN       NaN
2000-01-06       NaN       NaN       NaN
2000-01-07       NaN       NaN       NaN
```

(continues on next page)

(continued from previous page)

```
2000-01-08  0.113648 -1.478427  0.524988
2000-01-09  0.404705  0.577046 -1.715002
2000-01-10 -1.039268 -0.370647 -1.157892

[10 rows x 3 columns]
```

One can operate using string function names, callables, lists, or dictionaries of these.

Using a single function is equivalent to `.apply`.

```
In [4]: df.agg('sum')
Out[4]:
A   -1.068226
B   -1.387015
C   -4.892029
Length: 3, dtype: float64
```

Multiple aggregations with a list of functions.

```
In [5]: df.agg(['sum', 'min'])
Out[5]:
            A         B         C
sum -1.068226 -1.387015 -4.892029
min -1.135632 -1.478427 -1.715002

[2 rows x 3 columns]
```

Using a dict provides the ability to apply specific aggregations per column. You will get a matrix-like output of all of the aggregators. The output has one column per unique function. Those functions applied to a particular column will be `NaN`:

```
In [6]: df.agg({'A': ['sum', 'min'], 'B': ['min', 'max']})
Out[6]:
            A         B
max       NaN  1.212112
min -1.135632 -1.478427
sum -1.068226       NaN

[3 rows x 2 columns]
```

The API also supports a `.transform()` function for broadcasting results.

```
In [7]: df.transform(['abs', lambda x: x - x.min()])
Out[7]:
                   A                     B                     C
                 abs  <lambda>       abs  <lambda>       abs  <lambda>
2000-01-01  0.469112  1.604745  0.282863  1.195563  1.509059  0.205944
2000-01-02  1.135632  0.000000  1.212112  2.690539  0.173215  1.541787
2000-01-03  0.119209  1.254841  1.044236  0.434191  0.861849  0.853153
2000-01-04       NaN       NaN       NaN       NaN       NaN       NaN
2000-01-05       NaN       NaN       NaN       NaN       NaN       NaN
2000-01-06       NaN       NaN       NaN       NaN       NaN       NaN
2000-01-07       NaN       NaN       NaN       NaN       NaN       NaN
2000-01-08  0.113648  1.249281  1.478427  0.000000  0.524988  2.239990
2000-01-09  0.404705  1.540338  0.577046  2.055473  1.715002  0.000000
2000-01-10  1.039268  0.096364  0.370647  1.107780  1.157892  0.557110
```

(continues on next page)

(continued from previous page)

```
[10 rows x 6 columns]
```

When presented with mixed dtypes that cannot be aggregated, `.agg()` will only take the valid aggregations. This is similar to how groupby `.agg()` works. (GH15015)

```
In [8]: df = pd.DataFrame({'A': [1, 2, 3],
   ...:                    'B': [1., 2., 3.],
   ...:                    'C': ['foo', 'bar', 'baz'],
   ...:                    'D': pd.date_range('20130101', periods=3)})
   ...:

In [9]: df.dtypes
Out[9]:
A             int64
B           float64
C            object
D    datetime64[ns]
Length: 4, dtype: object
```

```
In [10]: df.agg(['min', 'sum'])
Out[10]:
     A    B          C          D
min  1  1.0        bar 2013-01-01
sum  6  6.0  foobarbaz        NaT

[2 rows x 4 columns]
```

`dtype` keyword for data IO

The `'python'` engine for *`read_csv()`*, as well as the *`read_fwf()`* function for parsing fixed-width text files and *`read_excel()`* for parsing Excel files, now accept the `dtype` keyword argument for specifying the types of specific columns (GH14295). See the *io docs* for more information.

```
In [11]: data = "a  b\n1  2\n3  4"

In [12]: pd.read_fwf(StringIO(data)).dtypes
Out[12]:
a    int64
b    int64
Length: 2, dtype: object

In [13]: pd.read_fwf(StringIO(data), dtype={'a': 'float64', 'b': 'object'}).dtypes
Out[13]:
a    float64
b     object
Length: 2, dtype: object
```

`.to_datetime()` has gained an `origin` parameter

to_datetime() has gained a new parameter, `origin`, to define a reference date from where to compute the resulting timestamps when parsing numerical values with a specific `unit` specified. (GH11276, GH11745)

For example, with 1960-01-01 as the starting date:

```
In [14]: pd.to_datetime([1, 2, 3], unit='D', origin=pd.Timestamp('1960-01-01'))
Out[14]: DatetimeIndex(['1960-01-02', '1960-01-03', '1960-01-04'], dtype=
↪'datetime64[ns]', freq=None)
```

The default is set at `origin='unix'`, which defaults to `1970-01-01 00:00:00`, which is commonly called 'unix epoch' or POSIX time. This was the previous default, so this is a backward compatible change.

```
In [15]: pd.to_datetime([1, 2, 3], unit='D')
Out[15]: DatetimeIndex(['1970-01-02', '1970-01-03', '1970-01-04'], dtype=
↪'datetime64[ns]', freq=None)
```

Groupby enhancements

Strings passed to `DataFrame.groupby()` as the `by` parameter may now reference either column names or index level names. Previously, only column names could be referenced. This allows to easily group by a column and index level at the same time. (GH5677)

```
In [16]: arrays = [['bar', 'bar', 'baz', 'baz', 'foo', 'foo', 'qux', 'qux'],
   ....:           ['one', 'two', 'one', 'two', 'one', 'two', 'one', 'two']]
   ....:

In [17]: index = pd.MultiIndex.from_arrays(arrays, names=['first', 'second'])

In [18]: df = pd.DataFrame({'A': [1, 1, 1, 1, 2, 2, 3, 3],
   ....:                    'B': np.arange(8)},
   ....:                   index=index)
   ....:

In [19]: df
Out[19]:
              A  B
first second
bar   one     1  0
      two     1  1
baz   one     1  2
      two     1  3
foo   one     2  4
      two     2  5
qux   one     3  6
      two     3  7

[8 rows x 2 columns]

In [20]: df.groupby(['second', 'A']).sum()
Out[20]:
          B
second A
one    1  2
       2  4
```

(continues on next page)

(continued from previous page)

```
       3  6
two    1  4
       2  5
       3  7

[6 rows x 1 columns]
```

Better support for compressed URLs in `read_csv`

The compression code was refactored (GH12688). As a result, reading dataframes from URLs in *read_csv()* or *read_table()* now supports additional compression methods: `xz`, `bz2`, and `zip` (GH14570). Previously, only `gzip` compression was supported. By default, compression of URLs and paths are now inferred using their file extensions. Additionally, support for bz2 compression in the python 2 C-engine improved (GH14874).

```
In [21]: url = ('https://github.com/{repo}/raw/{branch}/{path}'
   ....:        .format(repo='pandas-dev/pandas',
   ....:                branch='master',
   ....:                path='pandas/tests/io/parser/data/salaries.csv.bz2'))
   ....:

# default, infer compression
In [22]: df = pd.read_csv(url, sep='\t', compression='infer')

# explicitly specify compression
In [23]: df = pd.read_csv(url, sep='\t', compression='bz2')

In [24]: df.head(2)
Out[24]:
       S  X  E  M
0  13876  1  1  1
1  11608  1  3  0

[2 rows x 4 columns]
```

Pickle file I/O now supports compression

read_pickle(), *DataFrame.to_pickle()* and *Series.to_pickle()* can now read from and write to compressed pickle files. Compression methods can be an explicit parameter or be inferred from the file extension. See *the docs here.*

```
In [25]: df = pd.DataFrame({'A': np.random.randn(1000),
   ....:                    'B': 'foo',
   ....:                    'C': pd.date_range('20130101', periods=1000, freq='s')})
   ....:
```

Using an explicit compression type

```
In [26]: df.to_pickle("data.pkl.compress", compression="gzip")

In [27]: rt = pd.read_pickle("data.pkl.compress", compression="gzip")

In [28]: rt.head()
```

(continues on next page)

(continued from previous page)

```
Out[28]:
          A    B                   C
0 -1.344312  foo 2013-01-01 00:00:00
1  0.844885  foo 2013-01-01 00:00:01
2  1.075770  foo 2013-01-01 00:00:02
3 -0.109050  foo 2013-01-01 00:00:03
4  1.643563  foo 2013-01-01 00:00:04

[5 rows x 3 columns]
```

The default is to infer the compression type from the extension (`compression='infer'`):

```
In [29]: df.to_pickle("data.pkl.gz")

In [30]: rt = pd.read_pickle("data.pkl.gz")

In [31]: rt.head()
Out[31]:
          A    B                   C
0 -1.344312  foo 2013-01-01 00:00:00
1  0.844885  foo 2013-01-01 00:00:01
2  1.075770  foo 2013-01-01 00:00:02
3 -0.109050  foo 2013-01-01 00:00:03
4  1.643563  foo 2013-01-01 00:00:04

[5 rows x 3 columns]

In [32]: df["A"].to_pickle("s1.pkl.bz2")

In [33]: rt = pd.read_pickle("s1.pkl.bz2")

In [34]: rt.head()
Out[34]:
0   -1.344312
1    0.844885
2    1.075770
3   -0.109050
4    1.643563
Name: A, Length: 5, dtype: float64
```

UInt64 support improved

Pandas has significantly improved support for operations involving unsigned, or purely non-negative, integers. Previously, handling these integers would result in improper rounding or data-type casting, leading to incorrect results. Notably, a new numerical index, `UInt64Index`, has been created (GH14937)

```
In [35]: idx = pd.UInt64Index([1, 2, 3])

In [36]: df = pd.DataFrame({'A': ['a', 'b', 'c']}, index=idx)

In [37]: df.index
Out[37]: UInt64Index([1, 2, 3], dtype='uint64')
```

- Bug in converting object elements of array-like objects to unsigned 64-bit integers (GH4471, GH14982)
- Bug in `Series.unique()` in which unsigned 64-bit integers were causing overflow (GH14721)

- Bug in `DataFrame` construction in which unsigned 64-bit integer elements were being converted to objects (GH14881)
- Bug in `pd.read_csv()` in which unsigned 64-bit integer elements were being improperly converted to the wrong data types (GH14983)
- Bug in `pd.unique()` in which unsigned 64-bit integers were causing overflow (GH14915)
- Bug in `pd.value_counts()` in which unsigned 64-bit integers were being erroneously truncated in the output (GH14934)

GroupBy on categoricals

In previous versions, `.groupby(..., sort=False)` would fail with a `ValueError` when grouping on a categorical series with some categories not appearing in the data. (GH13179)

```
In [38]: chromosomes = np.r_[np.arange(1, 23).astype(str), ['X', 'Y']]

In [39]: df = pd.DataFrame({
   ....:     'A': np.random.randint(100),
   ....:     'B': np.random.randint(100),
   ....:     'C': np.random.randint(100),
   ....:     'chromosomes': pd.Categorical(np.random.choice(chromosomes, 100),
   ....:                                   categories=chromosomes,
   ....:                                   ordered=True)})
   ....:

In [40]: df
Out[40]:
     A   B   C chromosomes
0   87  22  81           4
1   87  22  81          13
2   87  22  81          22
3   87  22  81           2
4   87  22  81           6
..  ..  ..  ..         ...
95  87  22  81           8
96  87  22  81          11
97  87  22  81           X
98  87  22  81           1
99  87  22  81          19

[100 rows x 4 columns]
```

Previous behavior:

```
In [3]: df[df.chromosomes != '1'].groupby('chromosomes', sort=False).sum()
---------------------------------------------------------------------------
ValueError: items in new_categories are not the same as in old categories
```

New behavior:

```
In [41]: df[df.chromosomes != '1'].groupby('chromosomes', sort=False).sum()
Out[41]:
                A    B    C
chromosomes
2             348   88  324
```

(continues on next page)

(continued from previous page)

```
3             348   88  324
4             348   88  324
5             261   66  243
6             174   44  162
...           ...  ...  ...
22            348   88  324
X             348   88  324
Y             435  110  405
1               0    0    0
21              0    0    0

[24 rows x 3 columns]
```

Table schema output

The new orient `'table'` for *`DataFrame.to_json()`* will generate a Table Schema compatible string representation of the data.

```
In [42]: df = pd.DataFrame(
   ....:     {'A': [1, 2, 3],
   ....:      'B': ['a', 'b', 'c'],
   ....:      'C': pd.date_range('2016-01-01', freq='d', periods=3)},
   ....:     index=pd.Index(range(3), name='idx'))
   ....:

In [43]: df
Out[43]:
     A  B          C
idx
0    1  a 2016-01-01
1    2  b 2016-01-02
2    3  c 2016-01-03

[3 rows x 3 columns]

In [44]: df.to_json(orient='table')
Out[44]: '{"schema":{"fields":[{"name":"idx","type":"integer"},{"name":"A","type":
↪"integer"},{"name":"B","type":"string"},{"name":"C","type":"datetime"}],"primaryKey
↪":["idx"],"pandas_version":"0.20.0"},"data":[{"idx":0,"A":1,"B":"a","C":"2016-01-
↪01T00:00:00.000Z"},{"idx":1,"A":2,"B":"b","C":"2016-01-02T00:00:00.000Z"},{"idx":2,
↪"A":3,"B":"c","C":"2016-01-03T00:00:00.000Z"}]}'
```

See *IO: Table Schema for more information*.

Additionally, the repr for `DataFrame` and `Series` can now publish this JSON Table schema representation of the Series or DataFrame if you are using IPython (or another frontend like nteract using the Jupyter messaging protocol). This gives frontends like the Jupyter notebook and nteract more flexibility in how they display pandas objects, since they have more information about the data. You must enable this by setting the `display.html.table_schema` option to `True`.