```
In [73]: d = pd.Timestamp('2014-02-01')

In [74]: d
Out[74]: Timestamp('2014-02-01 00:00:00')

In [75]: d + pd.offsets.QuarterBegin(n=0, startingMonth=2)
Out[75]: Timestamp('2014-02-01 00:00:00')

In [76]: d + pd.offsets.QuarterBegin(n=0, startingMonth=1)
Out[76]: Timestamp('2014-04-01 00:00:00')
```

For the `QuarterBegin` offset in previous versions, the date would be rolled *backwards* if date was in the same month as the quarter start date.

```
In [3]: d = pd.Timestamp('2014-02-15')

In [4]: d + pd.offsets.QuarterBegin(n=0, startingMonth=2)
Out[4]: Timestamp('2014-02-01 00:00:00')
```

This behavior has been corrected in version 0.18.0, which is consistent with other anchored offsets like `MonthBegin` and `YearBegin`.

```
In [77]: d = pd.Timestamp('2014-02-15')

In [78]: d + pd.offsets.QuarterBegin(n=0, startingMonth=2)
Out[78]: Timestamp('2014-05-01 00:00:00')
```

Resample API

Like the change in the window functions API *above*, `.resample(...)` is changing to have a more groupby-like API. (GH11732, GH12702, GH12202, GH12332, GH12334, GH12348, GH12448).

```
In [79]: np.random.seed(1234)

In [80]: df = pd.DataFrame(np.random.rand(10,4),
   ....:                   columns=list('ABCD'),
   ....:                   index=pd.date_range('2010-01-01 09:00:00',
   ....:                                       periods=10, freq='s'))
   ....:

In [81]: df
Out[81]:
                            A         B         C         D
2010-01-01 09:00:00  0.191519  0.622109  0.437728  0.785359
2010-01-01 09:00:01  0.779976  0.272593  0.276464  0.801872
2010-01-01 09:00:02  0.958139  0.875933  0.357817  0.500995
2010-01-01 09:00:03  0.683463  0.712702  0.370251  0.561196
2010-01-01 09:00:04  0.503083  0.013768  0.772827  0.882641
2010-01-01 09:00:05  0.364886  0.615396  0.075381  0.368824
2010-01-01 09:00:06  0.933140  0.651378  0.397203  0.788730
2010-01-01 09:00:07  0.316836  0.568099  0.869127  0.436173
2010-01-01 09:00:08  0.802148  0.143767  0.704261  0.704581
2010-01-01 09:00:09  0.218792  0.924868  0.442141  0.909316

[10 rows x 4 columns]
```

Previous API:

You would write a resampling operation that immediately evaluates. If a `how` parameter was not provided, it would default to `how='mean'`.

```
In [6]: df.resample('2s')
Out[6]:
                           A         B         C         D
2010-01-01 09:00:00  0.485748  0.447351  0.357096  0.793615
2010-01-01 09:00:02  0.820801  0.794317  0.364034  0.531096
2010-01-01 09:00:04  0.433985  0.314582  0.424104  0.625733
2010-01-01 09:00:06  0.624988  0.609738  0.633165  0.612452
2010-01-01 09:00:08  0.510470  0.534317  0.573201  0.806949
```

You could also specify a `how` directly

```
In [7]: df.resample('2s', how='sum')
Out[7]:
                           A         B         C         D
2010-01-01 09:00:00  0.971495  0.894701  0.714192  1.587231
2010-01-01 09:00:02  1.641602  1.588635  0.728068  1.062191
2010-01-01 09:00:04  0.867969  0.629165  0.848208  1.251465
2010-01-01 09:00:06  1.249976  1.219477  1.266330  1.224904
2010-01-01 09:00:08  1.020940  1.068634  1.146402  1.613897
```

New API:

Now, you can write `.resample(..)` as a 2-stage operation like `.groupby(...)`, which yields a `Resampler`.

```
In [82]: r = df.resample('2s')

In [83]: r
Out[83]: <pandas.core.resample.DatetimeIndexResampler object at 0x7f53438ebb50>
```

Downsampling

You can then use this object to perform operations. These are downsampling operations (going from a higher frequency to a lower one).

```
In [84]: r.mean()
Out[84]:
                           A         B         C         D
2010-01-01 09:00:00  0.485748  0.447351  0.357096  0.793615
2010-01-01 09:00:02  0.820801  0.794317  0.364034  0.531096
2010-01-01 09:00:04  0.433985  0.314582  0.424104  0.625733
2010-01-01 09:00:06  0.624988  0.609738  0.633165  0.612452
2010-01-01 09:00:08  0.510470  0.534317  0.573201  0.806949

[5 rows x 4 columns]
```

```
In [85]: r.sum()
Out[85]:
                           A         B         C         D
2010-01-01 09:00:00  0.971495  0.894701  0.714192  1.587231
2010-01-01 09:00:02  1.641602  1.588635  0.728068  1.062191
2010-01-01 09:00:04  0.867969  0.629165  0.848208  1.251465
```

(continues on next page)

(continued from previous page)

```
2010-01-01 09:00:06  1.249976  1.219477  1.266330  1.224904
2010-01-01 09:00:08  1.020940  1.068634  1.146402  1.613897

[5 rows x 4 columns]
```

Furthermore, resample now supports `getitem` operations to perform the resample on specific columns.

```
In [86]: r[['A','C']].mean()
Out[86]:
                            A         C
2010-01-01 09:00:00  0.485748  0.357096
2010-01-01 09:00:02  0.820801  0.364034
2010-01-01 09:00:04  0.433985  0.424104
2010-01-01 09:00:06  0.624988  0.633165
2010-01-01 09:00:08  0.510470  0.573201

[5 rows x 2 columns]
```

and `.aggregate` type operations.

```
In [87]: r.agg({'A' : 'mean', 'B' : 'sum'})
Out[87]:
                            A         B
2010-01-01 09:00:00  0.485748  0.894701
2010-01-01 09:00:02  0.820801  1.588635
2010-01-01 09:00:04  0.433985  0.629165
2010-01-01 09:00:06  0.624988  1.219477
2010-01-01 09:00:08  0.510470  1.068634

[5 rows x 2 columns]
```

These accessors can of course, be combined

```
In [88]: r[['A','B']].agg(['mean','sum'])
Out[88]:
                            A                   B
                         mean       sum      mean       sum
2010-01-01 09:00:00  0.485748  0.971495  0.447351  0.894701
2010-01-01 09:00:02  0.820801  1.641602  0.794317  1.588635
2010-01-01 09:00:04  0.433985  0.867969  0.314582  0.629165
2010-01-01 09:00:06  0.624988  1.249976  0.609738  1.219477
2010-01-01 09:00:08  0.510470  1.020940  0.534317  1.068634

[5 rows x 4 columns]
```

Upsampling

Upsampling operations take you from a lower frequency to a higher frequency. These are now performed with the `Resampler` objects with `backfill()`, `ffill()`, `fillna()` and `asfreq()` methods.

```
In [89]: s = pd.Series(np.arange(5, dtype='int64'),
   ....:               index=pd.date_range('2010-01-01', periods=5, freq='Q'))
   ....:

In [90]: s
```

(continues on next page)

(continued from previous page)

```
Out[90]:
2010-03-31    0
2010-06-30    1
2010-09-30    2
2010-12-31    3
2011-03-31    4
Freq: Q-DEC, Length: 5, dtype: int64
```

Previously

```
In [6]: s.resample('M', fill_method='ffill')
Out[6]:
2010-03-31    0
2010-04-30    0
2010-05-31    0
2010-06-30    1
2010-07-31    1
2010-08-31    1
2010-09-30    2
2010-10-31    2
2010-11-30    2
2010-12-31    3
2011-01-31    3
2011-02-28    3
2011-03-31    4
Freq: M, dtype: int64
```

New API

```
In [91]: s.resample('M').ffill()
Out[91]:
2010-03-31    0
2010-04-30    0
2010-05-31    0
2010-06-30    1
2010-07-31    1
2010-08-31    1
2010-09-30    2
2010-10-31    2
2010-11-30    2
2010-12-31    3
2011-01-31    3
2011-02-28    3
2011-03-31    4
Freq: M, Length: 13, dtype: int64
```

Note: In the new API, you can either downsample OR upsample. The prior implementation would allow you to pass an aggregator function (like `mean`) even though you were upsampling, providing a bit of confusion.

Previous API will work but with deprecations

Warning: This new API for resample includes some internal changes for the prior-to-0.18.0 API, to work with a deprecation warning in most cases, as the resample operation returns a deferred object. We can intercept operations and just do what the (pre 0.18.0) API did (with a warning). Here is a typical use case:

```
In [4]: r = df.resample('2s')

In [6]: r*10
pandas/tseries/resample.py:80: FutureWarning: .resample() is now a deferred␣
↪operation
use .resample(...).mean() instead of .resample(...)

Out[6]:
                            A         B         C         D
2010-01-01 09:00:00  4.857476  4.473507  3.570960  7.936154
2010-01-01 09:00:02  8.208011  7.943173  3.640340  5.310957
2010-01-01 09:00:04  4.339846  3.145823  4.241039  6.257326
2010-01-01 09:00:06  6.249881  6.097384  6.331650  6.124518
2010-01-01 09:00:08  5.104699  5.343172  5.732009  8.069486
```

However, getting and assignment operations directly on a `Resampler` will raise a `ValueError`:

```
In [7]: r.iloc[0] = 5
ValueError: .resample() is now a deferred operation
use .resample(...).mean() instead of .resample(...)
```

There is a situation where the new API can not perform all the operations when using original code. This code is intending to resample every 2s, take the `mean` AND then take the `min` of those results.

```
In [4]: df.resample('2s').min()
Out[4]:
A    0.433985
B    0.314582
C    0.357096
D    0.531096
dtype: float64
```

The new API will:

```
In [92]: df.resample('2s').min()
Out[92]:
                               A         B         C         D
2010-01-01 09:00:00  0.191519  0.272593  0.276464  0.785359
2010-01-01 09:00:02  0.683463  0.712702  0.357817  0.500995
2010-01-01 09:00:04  0.364886  0.013768  0.075381  0.368824
2010-01-01 09:00:06  0.316836  0.568099  0.397203  0.436173
2010-01-01 09:00:08  0.218792  0.143767  0.442141  0.704581

[5 rows x 4 columns]
```

The good news is the return dimensions will differ between the new API and the old API, so this should loudly raise an exception.

To replicate the original operation

```
In [93]: df.resample('2s').mean().min()
Out[93]:
A    0.433985
B    0.314582
C    0.357096
D    0.531096
Length: 4, dtype: float64
```

Changes to eval

In prior versions, new columns assignments in an `eval` expression resulted in an inplace change to the `DataFrame`. (GH9297, GH8664, GH10486)

```
In [94]: df = pd.DataFrame({'a': np.linspace(0, 10, 5), 'b': range(5)})

In [95]: df
Out[95]:
      a  b
0   0.0  0
1   2.5  1
2   5.0  2
3   7.5  3
4  10.0  4

[5 rows x 2 columns]
```

```
In [12]: df.eval('c = a + b')
FutureWarning: eval expressions containing an assignment currentlydefault to␣
↪operating inplace.
This will change in a future version of pandas, use inplace=True to avoid this␣
↪warning.

In [13]: df
Out[13]:
      a  b     c
0   0.0  0   0.0
1   2.5  1   3.5
2   5.0  2   7.0
3   7.5  3  10.5
4  10.0  4  14.0
```

In version 0.18.0, a new `inplace` keyword was added to choose whether the assignment should be done inplace or return a copy.

```
In [96]: df
Out[96]:
      a  b     c
0   0.0  0   0.0
1   2.5  1   3.5
2   5.0  2   7.0
3   7.5  3  10.5
4  10.0  4  14.0

[5 rows x 3 columns]

In [97]: df.eval('d = c - b', inplace=False)
Out[97]:
      a  b     c     d
0   0.0  0   0.0   0.0
1   2.5  1   3.5   2.5
2   5.0  2   7.0   5.0
```

(continues on next page)

(continued from previous page)

```
3   7.5  3  10.5   7.5
4  10.0  4  14.0  10.0

[5 rows x 4 columns]

In [98]: df
Out[98]:
      a  b     c
0   0.0  0   0.0
1   2.5  1   3.5
2   5.0  2   7.0
3   7.5  3  10.5
4  10.0  4  14.0

[5 rows x 3 columns]

In [99]: df.eval('d = c - b', inplace=True)

In [100]: df
Out[100]:
      a  b     c     d
0   0.0  0   0.0   0.0
1   2.5  1   3.5   2.5
2   5.0  2   7.0   5.0
3   7.5  3  10.5   7.5
4  10.0  4  14.0  10.0

[5 rows x 4 columns]
```

Warning: For backwards compatibility, `inplace` defaults to `True` if not specified. This will change in a future version of pandas. If your code depends on an inplace assignment you should update to explicitly set `inplace=True`

The `inplace` keyword parameter was also added the `query` method.

```
In [101]: df.query('a > 5')
Out[101]:
      a  b     c     d
3   7.5  3  10.5   7.5
4  10.0  4  14.0  10.0

[2 rows x 4 columns]

In [102]: df.query('a > 5', inplace=True)

In [103]: df
Out[103]:
      a  b     c     d
3   7.5  3  10.5   7.5
4  10.0  4  14.0  10.0

[2 rows x 4 columns]
```

Warning: Note that the default value for `inplace` in a `query` is `False`, which is consistent with prior versions.

`eval` has also been updated to allow multi-line expressions for multiple assignments. These expressions will be evaluated one at a time in order. Only assignments are valid for multi-line expressions.

```
In [104]: df
Out[104]:
      a  b     c     d
3   7.5  3  10.5   7.5
4  10.0  4  14.0  10.0

[2 rows x 4 columns]

In [105]: df.eval("""
   .....: e = d + a
   .....: f = e - 22
   .....: g = f / 2.0""", inplace=True)
   .....:

In [106]: df
Out[106]:
      a  b     c     d     e    f    g
3   7.5  3  10.5   7.5  15.0 -7.0 -3.5
4  10.0  4  14.0  10.0  20.0 -2.0 -1.0

[2 rows x 7 columns]
```

Other API changes

- `DataFrame.between_time` and `Series.between_time` now only parse a fixed set of time strings. Parsing of date strings is no longer supported and raises a `ValueError`. (GH11818)

```
In [107]: s = pd.Series(range(10), pd.date_range('2015-01-01', freq='H',␣
↪periods=10))

In [108]: s.between_time("7:00am", "9:00am")
Out[108]:
2015-01-01 07:00:00    7
2015-01-01 08:00:00    8
2015-01-01 09:00:00    9
Freq: H, Length: 3, dtype: int64
```

  This will now raise.

```
In [2]: s.between_time('20150101 07:00:00','20150101 09:00:00')
ValueError: Cannot convert arg ['20150101 07:00:00'] to a time.
```

- `.memory_usage()` now includes values in the index, as does memory_usage in `.info()` (GH11597)
- `DataFrame.to_latex()` now supports non-ascii encodings (eg `utf-8`) in Python 2 with the parameter `encoding` (GH7061)
- `pandas.merge()` and `DataFrame.merge()` will show a specific error message when trying to merge with an object that is not of type `DataFrame` or a subclass (GH12081)

- `DataFrame.unstack` and `Series.unstack` now take `fill_value` keyword to allow direct replacement of missing values when an unstack results in missing values in the resulting `DataFrame`. As an added benefit, specifying `fill_value` will preserve the data type of the original stacked data. (GH9746)
- As part of the new API for *window functions* and *resampling*, aggregation functions have been clarified, raising more informative error messages on invalid aggregations. (GH9052). A full set of examples are presented in *groupby*.
- Statistical functions for `NDFrame` objects (like `sum(), mean(), min()`) will now raise if non-numpy-compatible arguments are passed in for `**kwargs` (GH12301)
- `.to_latex` and `.to_html` gain a `decimal` parameter like `.to_csv`; the default is `'.'` (GH12031)
- More helpful error message when constructing a `DataFrame` with empty data but with indices (GH8020)
- `.describe()` will now properly handle bool dtype as a categorical (GH6625)
- More helpful error message with an invalid `.transform` with user defined input (GH10165)
- Exponentially weighted functions now allow specifying alpha directly (GH10789) and raise `ValueError` if parameters violate `0 < alpha <= 1` (GH12492)

Deprecations

- The functions `pd.rolling_*`, `pd.expanding_*`, and `pd.ewm*` are deprecated and replaced by the corresponding method call. Note that the new suggested syntax includes all of the arguments (even if default) (GH11603)

```
In [1]: s = pd.Series(range(3))

In [2]: pd.rolling_mean(s,window=2,min_periods=1)
        FutureWarning: pd.rolling_mean is deprecated for Series and
             will be removed in a future version, replace with
             Series.rolling(min_periods=1,window=2,center=False).mean()
Out[2]:
        0    0.0
        1    0.5
        2    1.5
        dtype: float64

In [3]: pd.rolling_cov(s, s, window=2)
        FutureWarning: pd.rolling_cov is deprecated for Series and
             will be removed in a future version, replace with
             Series.rolling(window=2).cov(other=<Series>)
Out[3]:
        0    NaN
        1    0.5
        2    0.5
        dtype: float64
```

- The `freq` and `how` arguments to the `.rolling`, `.expanding`, and `.ewm` (new) functions are deprecated, and will be removed in a future version. You can simply resample the input prior to creating a window function. (GH11603).

  For example, instead of `s.rolling(window=5,freq='D').max()` to get the max value on a rolling 5 Day window, one could use `s.resample('D').mean().rolling(window=5).max()`, which first resamples the data to daily data, then provides a rolling 5 day window.

- `pd.tseries.frequencies.get_offset_name` function is deprecated. Use offset's `.freqstr` property as alternative (GH11192)
- `pandas.stats.fama_macbeth` routines are deprecated and will be removed in a future version (GH6077)
- `pandas.stats.ols`, `pandas.stats.plm` and `pandas.stats.var` routines are deprecated and will be removed in a future version (GH6077)
- show a `FutureWarning` rather than a `DeprecationWarning` on using long-time deprecated syntax in `HDFStore.select`, where the `where` clause is not a string-like (GH12027)
- The `pandas.options.display.mpl_style` configuration has been deprecated and will be removed in a future version of pandas. This functionality is better handled by matplotlib's style sheets (GH11783).

Removal of deprecated float indexers

In GH4892 indexing with floating point numbers on a non-`Float64Index` was deprecated (in version 0.14.0). In 0.18.0, this deprecation warning is removed and these will now raise a `TypeError`. (GH12165, GH12333)

```
In [109]: s = pd.Series([1, 2, 3], index=[4, 5, 6])

In [110]: s
Out[110]:
4    1
5    2
6    3
Length: 3, dtype: int64

In [111]: s2 = pd.Series([1, 2, 3], index=list('abc'))

In [112]: s2
Out[112]:
a    1
b    2
c    3
Length: 3, dtype: int64
```

Previous behavior:

```
# this is label indexing
In [2]: s[5.0]
FutureWarning: scalar indexers for index type Int64Index should be integers and not␣
↪floating point
Out[2]: 2

# this is positional indexing
In [3]: s.iloc[1.0]
FutureWarning: scalar indexers for index type Int64Index should be integers and not␣
↪floating point
Out[3]: 2

# this is label indexing
In [4]: s.loc[5.0]
FutureWarning: scalar indexers for index type Int64Index should be integers and not␣
↪floating point
Out[4]: 2

# .ix would coerce 1.0 to the positional 1, and index
```

(continues on next page)

(continued from previous page)

```
In [5]: s2.ix[1.0] = 10
FutureWarning: scalar indexers for index type Index should be integers and not␣
↪floating point

In [6]: s2
Out[6]:
a     1
b    10
c     3
dtype: int64
```

New behavior:

For iloc, getting & setting via a float scalar will always raise.

```
In [3]: s.iloc[2.0]
TypeError: cannot do label indexing on <class 'pandas.indexes.numeric.Int64Index'>␣
↪with these indexers [2.0] of <type 'float'>
```

Other indexers will coerce to a like integer for both getting and setting. The `FutureWarning` has been dropped for `.loc`, `.ix` and `[]`.

```
In [113]: s[5.0]
Out[113]: 2

In [114]: s.loc[5.0]
Out[114]: 2
```

and setting

```
In [115]: s_copy = s.copy()

In [116]: s_copy[5.0] = 10

In [117]: s_copy
Out[117]:
4     1
5    10
6     3
Length: 3, dtype: int64

In [118]: s_copy = s.copy()

In [119]: s_copy.loc[5.0] = 10

In [120]: s_copy
Out[120]:
4     1
5    10
6     3
Length: 3, dtype: int64
```

Positional setting with `.ix` and a float indexer will ADD this value to the index, rather than previously setting the value by position.

```
In [3]: s2.ix[1.0] = 10
In [4]: s2
```

(continues on next page)

(continued from previous page)

```
Out[4]:
a       1
b       2
c       3
1.0    10
dtype: int64
```

Slicing will also coerce integer-like floats to integers for a non-`Float64Index`.

```
In [121]: s.loc[5.0:6]
Out[121]:
5    2
6    3
Length: 2, dtype: int64
```

Note that for floats that are NOT coercible to ints, the label based bounds will be excluded

```
In [122]: s.loc[5.1:6]
Out[122]:
6    3
Length: 1, dtype: int64
```

Float indexing on a `Float64Index` is unchanged.

```
In [123]: s = pd.Series([1, 2, 3], index=np.arange(3.))

In [124]: s[1.0]
Out[124]: 2

In [125]: s[1.0:2.5]
Out[125]:
1.0    2
2.0    3
Length: 2, dtype: int64
```

Removal of prior version deprecations/changes

- Removal of `rolling_corr_pairwise` in favor of `.rolling().corr(pairwise=True)` (GH4950)
- Removal of `expanding_corr_pairwise` in favor of `.expanding().corr(pairwise=True)` (GH4950)
- Removal of `DataMatrix` module. This was not imported into the pandas namespace in any event (GH12111)
- Removal of `cols` keyword in favor of `subset` in `DataFrame.duplicated()` and `DataFrame.drop_duplicates()` (GH6680)
- Removal of the `read_frame` and `frame_query` (both aliases for `pd.read_sql`) and `write_frame` (alias of `to_sql`) functions in the `pd.io.sql` namespace, deprecated since 0.14.0 (GH6292).
- Removal of the `order` keyword from `.factorize()` (GH6930)

Performance improvements

- Improved performance of `andrews_curves` (GH11534)
- Improved huge `DatetimeIndex`, `PeriodIndex` and `TimedeltaIndex`'s ops performance including `NaT` (GH10277)
- Improved performance of `pandas.concat` (GH11958)
- Improved performance of `StataReader` (GH11591)
- Improved performance in construction of `Categoricals` with `Series` of datetimes containing `NaT` (GH12077)
- Improved performance of ISO 8601 date parsing for dates without separators (GH11899), leading zeros (GH11871) and with white space preceding the time zone (GH9714)

Bug Fixes

- Bug in `GroupBy.size` when data-frame is empty. (GH11699)
- Bug in `Period.end_time` when a multiple of time period is requested (GH11738)
- Regression in `.clip` with tz-aware datetimes (GH11838)
- Bug in `date_range` when the boundaries fell on the frequency (GH11804, GH12409)
- Bug in consistency of passing nested dicts to `.groupby(...).agg(...)` (GH9052)
- Accept unicode in `Timedelta` constructor (GH11995)
- Bug in value label reading for `StataReader` when reading incrementally (GH12014)
- Bug in vectorized `DateOffset` when `n` parameter is `0` (GH11370)
- Compat for numpy 1.11 w.r.t. `NaT` comparison changes (GH12049)
- Bug in `read_csv` when reading from a `StringIO` in threads (GH11790)
- Bug in not treating `NaT` as a missing value in datetimelikes when factorizing & with `Categoricals` (GH12077)
- Bug in getitem when the values of a `Series` were tz-aware (GH12089)
- Bug in `Series.str.get_dummies` when one of the variables was 'name' (GH12180)
- Bug in `pd.concat` while concatenating tz-aware NaT series. (GH11693, GH11755, GH12217)
- Bug in `pd.read_stata` with version <= 108 files (GH12232)
- Bug in `Series.resample` using a frequency of `Nano` when the index is a `DatetimeIndex` and contains non-zero nanosecond parts (GH12037)
- Bug in resampling with `.nunique` and a sparse index (GH12352)
- Removed some compiler warnings (GH12471)
- Work around compat issues with `boto` in python 3.5 (GH11915)
- Bug in `NaT` subtraction from `Timestamp` or `DatetimeIndex` with timezones (GH11718)
- Bug in subtraction of `Series` of a single tz-aware `Timestamp` (GH12290)
- Use compat iterators in PY2 to support `.next()` (GH12299)
- Bug in `Timedelta.round` with negative values (GH11690)

- Bug in `.loc` against `CategoricalIndex` may result in normal `Index` (GH11586)
- Bug in `DataFrame.info` when duplicated column names exist (GH11761)
- Bug in `.copy` of datetime tz-aware objects (GH11794)
- Bug in `Series.apply` and `Series.map` where `timedelta64` was not boxed (GH11349)
- Bug in `DataFrame.set_index()` with tz-aware `Series` (GH12358)
- Bug in subclasses of `DataFrame` where `AttributeError` did not propagate (GH11808)
- Bug groupby on tz-aware data where selection not returning `Timestamp` (GH11616)
- Bug in `pd.read_clipboard` and `pd.to_clipboard` functions not supporting Unicode; upgrade included `pyperclip` to v1.5.15 (GH9263)
- Bug in `DataFrame.query` containing an assignment (GH8664)
- Bug in `from_msgpack` where `__contains__()` fails for columns of the unpacked `DataFrame`, if the `DataFrame` has object columns. (GH11880)
- Bug in `.resample` on categorical data with `TimedeltaIndex` (GH12169)
- Bug in timezone info lost when broadcasting scalar datetime to `DataFrame` (GH11682)
- Bug in `Index` creation from `Timestamp` with mixed tz coerces to UTC (GH11488)
- Bug in `to_numeric` where it does not raise if input is more than one dimension (GH11776)
- Bug in parsing timezone offset strings with non-zero minutes (GH11708)
- Bug in `df.plot` using incorrect colors for bar plots under matplotlib 1.5+ (GH11614)
- Bug in the `groupby plot` method when using keyword arguments (GH11805).
- Bug in `DataFrame.duplicated` and `drop_duplicates` causing spurious matches when setting `keep=False` (GH11864)
- Bug in `.loc` result with duplicated key may have `Index` with incorrect dtype (GH11497)
- Bug in `pd.rolling_median` where memory allocation failed even with sufficient memory (GH11696)
- Bug in `DataFrame.style` with spurious zeros (GH12134)
- Bug in `DataFrame.style` with integer columns not starting at 0 (GH12125)
- Bug in `.style.bar` may not rendered properly using specific browser (GH11678)
- Bug in rich comparison of `Timedelta` with a `numpy.array` of `Timedelta` that caused an infinite recursion (GH11835)
- Bug in `DataFrame.round` dropping column index name (GH11986)
- Bug in `df.replace` while replacing value in mixed dtype `Dataframe` (GH11698)
- Bug in `Index` prevents copying name of passed `Index`, when a new name is not provided (GH11193)
- Bug in `read_excel` failing to read any non-empty sheets when empty sheets exist and `sheetname=None` (GH11711)
- Bug in `read_excel` failing to raise `NotImplemented` error when keywords `parse_dates` and `date_parser` are provided (GH11544)
- Bug in `read_sql` with `pymysql` connections failing to return chunked data (GH11522)
- Bug in `.to_csv` ignoring formatting parameters `decimal`, `na_rep`, `float_format` for float indexes (GH11553)

- Bug in `Int64Index` and `Float64Index` preventing the use of the modulo operator (GH9244)
- Bug in `MultiIndex.drop` for not lexsorted MultiIndexes (GH12078)
- Bug in `DataFrame` when masking an empty `DataFrame` (GH11859)
- Bug in `.plot` potentially modifying the `colors` input when the number of columns didn't match the number of series provided (GH12039).
- Bug in `Series.plot` failing when index has a `CustomBusinessDay` frequency (GH7222).
- Bug in `.to_sql` for `datetime.time` values with sqlite fallback (GH8341)
- Bug in `read_excel` failing to read data with one column when `squeeze=True` (GH12157)
- Bug in `read_excel` failing to read one empty column (GH12292, GH9002)
- Bug in `.groupby` where a `KeyError` was not raised for a wrong column if there was only one row in the dataframe (GH11741)
- Bug in `.read_csv` with dtype specified on empty data producing an error (GH12048)
- Bug in `.read_csv` where strings like `'2E'` are treated as valid floats (GH12237)
- Bug in building *pandas* with debugging symbols (GH12123)
- Removed `millisecond` property of `DatetimeIndex`. This would always raise a `ValueError` (GH12019).
- Bug in `Series` constructor with read-only data (GH11502)
- Removed `pandas._testing.choice()`. Should use `np.random.choice()`, instead. (GH12386)
- Bug in `.loc` setitem indexer preventing the use of a TZ-aware DatetimeIndex (GH12050)
- Bug in `.style` indexes and MultiIndexes not appearing (GH11655)
- Bug in `to_msgpack` and `from_msgpack` which did not correctly serialize or deserialize `NaT` (GH12307).
- Bug in `.skew` and `.kurt` due to roundoff error for highly similar values (GH11974)
- Bug in `Timestamp` constructor where microsecond resolution was lost if HHMMSS were not separated with ':' (GH10041)
- Bug in `buffer_rd_bytes` src->buffer could be freed more than once if reading failed, causing a segfault (GH12098)
- Bug in `crosstab` where arguments with non-overlapping indexes would return a `KeyError` (GH10291)
- Bug in `DataFrame.apply` in which reduction was not being prevented for cases in which `dtype` was not a numpy dtype (GH12244)
- Bug when initializing categorical series with a scalar value. (GH12336)
- Bug when specifying a UTC `DatetimeIndex` by setting `utc=True` in `.to_datetime` (GH11934)
- Bug when increasing the buffer size of CSV reader in `read_csv` (GH12494)
- Bug when setting columns of a `DataFrame` with duplicate column names (GH12344)

Contributors

A total of 101 people contributed patches to this release. People with a "+" by their names contributed a patch for the first time.

- ARF +
- Alex Alekseyev +
- Andrew McPherson +
- Andrew Rosenfeld
- Andy Hayden
- Anthonios Partheniou
- Anton I. Sipos
- Ben +
- Ben North +
- Bran Yang +
- Chris
- Chris Carroux +
- Christopher C. Aycock +
- Christopher Scanlin +
- Cody +
- Da Wang +
- Daniel Grady +
- Dorozhko Anton +
- Dr-Irv +
- Erik M. Bray +
- Evan Wright
- Francis T. O'Donovan +
- Frank Cleary +
- Gianluca Rossi
- Graham Jeffries +
- Guillaume Horel
- Henry Hammond +
- Isaac Schwabacher +
- Jean-Mathieu Deschenes
- Jeff Reback
- Joe Jevnik +
- John Freeman +
- John Fremlin +

- Jonas Hoersch +
- Joris Van den Bossche
- Joris Vankerschaver
- Justin Lecher
- Justin Lin +
- Ka Wo Chen
- Keming Zhang +
- Kerby Shedden
- Kyle +
- Marco Farrugia +
- MasonGallo +
- MattRijk +
- Matthew Lurie +
- Maximilian Roos
- Mayank Asthana +
- Mortada Mehyar
- Moussa Taifi +
- Navreet Gill +
- Nicolas Bonnotte
- Paul Reiners +
- Philip Gura +
- Pietro Battiston
- RahulHP +
- Randy Carnevale
- Rinoc Johnson
- Rishipuri +
- Sangmin Park +
- Scott E Lasley
- Sereger13 +
- Shannon Wang +
- Skipper Seabold
- Thierry Moisan
- Thomas A Caswell
- Toby Dylan Hocking +
- Tom Augspurger
- Travis +

- Trent Hauck
- Tux1
- Varun
- Wes McKinney
- Will Thompson +
- Yoav Ram
- Yoong Kang Lim +
- Yoshiki Vázquez Baeza
- Young Joong Kim +
- Younggun Kim
- Yuval Langer +
- alex argunov +
- behzad nouri
- boombard +
- brian-pantano +
- chromy +
- daniel +
- dgram0 +
- gfyoung +
- hack-c +
- hcontrast +
- jfoo +
- kaustuv deolal +
- lllllllll
- ranarag +
- rockg
- scls19fr
- seales +
- sinhrks
- srib +
- surveymedia.ca +
- tworec +

5.10 Version 0.17

5.10.1 v0.17.1 (November 21, 2015)

Note: We are proud to announce that *pandas* has become a sponsored project of the (NumFOCUS organization). This will help ensure the success of development of *pandas* as a world-class open-source project.

This is a minor bug-fix release from 0.17.0 and includes a large number of bug fixes along several new features, enhancements, and performance improvements. We recommend that all users upgrade to this version.

Highlights include:

- Support for Conditional HTML Formatting, see *here*
- Releasing the GIL on the csv reader & other ops, see *here*
- Fixed regression in `DataFrame.drop_duplicates` from 0.16.2, causing incorrect results on integer values (GH11376)

What's new in v0.17.1

- *New features*
  - *Conditional HTML formatting*
- *Enhancements*
- *API changes*
  - *Deprecations*
- *Performance improvements*
- *Bug fixes*
- *Contributors*

New features

Conditional HTML formatting

Warning: This is a new feature and is under active development. We'll be adding features an possibly making breaking changes in future releases. Feedback is welcome.

We've added *experimental* support for conditional HTML formatting: the visual styling of a DataFrame based on the data. The styling is accomplished with HTML and CSS. Accesses the styler class with the *`pandas.DataFrame.style`*, attribute, an instance of `Styler` with your data attached.

Here's a quick example:

```
In [1]: np.random.seed(123)

In [2]: df = pd.DataFrame(np.random.randn(10, 5), columns=list('abcde'))
```

(continues on next page)

(continued from previous page)

```
In [3]: html = df.style.background_gradient(cmap='viridis', low=.5)
```

We can render the HTML to get the following table.

`Styler` interacts nicely with the Jupyter Notebook. See the *documentation* for more.

Enhancements

- `DatetimeIndex` now supports conversion to strings with `astype(str)` (GH10442)
- Support for `compression` (gzip/bz2) in *`pandas.DataFrame.to_csv()`* (GH7615)
- `pd.read_*` functions can now also accept `pathlib.Path`, or `py._path.local.LocalPath` objects for the `filepath_or_buffer` argument. (GH11033) - The `DataFrame` and `Series` functions `.to_csv()`, `.to_html()` and `.to_latex()` can now handle paths beginning with tildes (e.g. `~/Documents/`) (GH11438)
- `DataFrame` now uses the fields of a `namedtuple` as columns, if columns are not supplied (GH11181)
- `DataFrame.itertuples()` now returns `namedtuple` objects, when possible. (GH11269, GH11625)
- Added `axvlines_kwds` to parallel coordinates plot (GH10709)
- Option to `.info()` and `.memory_usage()` to provide for deep introspection of memory consumption. Note that this can be expensive to compute and therefore is an optional parameter. (GH11595)

```
In [4]: df = pd.DataFrame({'A': ['foo'] * 1000})  # noqa: F821

In [5]: df['B'] = df['A'].astype('category')

# shows the '+' as we have object dtypes
In [6]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   A       1000 non-null   object
 1   B       1000 non-null   category
dtypes: category(1), object(1)
memory usage: 9.0+ KB

# we have an accurate memory assessment (but can be expensive to compute this)
In [7]: df.info(memory_usage='deep')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   A       1000 non-null   object
 1   B       1000 non-null   category
dtypes: category(1), object(1)
memory usage: 75.5 KB
```

- `Index` now has a `fillna` method (GH10089)

```
In [8]: pd.Index([1, np.nan, 3]).fillna(2)
Out[8]: Float64Index([1.0, 2.0, 3.0], dtype='float64')
```

- Series of type `category` now make `.str.<...>` and `.dt.<...>` accessor methods / properties available, if the categories are of that type. (GH10661)

```
In [9]: s = pd.Series(list('aabb')).astype('category')

In [10]: s
Out[10]:
0    a
1    a
2    b
3    b
Length: 4, dtype: category
Categories (2, object): [a, b]

In [11]: s.str.contains("a")
Out[11]:
0     True
1     True
2    False
3    False
Length: 4, dtype: bool

In [12]: date = pd.Series(pd.date_range('1/1/2015', periods=5)).astype('category')

In [13]: date
Out[13]:
0   2015-01-01
1   2015-01-02
2   2015-01-03
3   2015-01-04
4   2015-01-05
Length: 5, dtype: category
Categories (5, datetime64[ns]): [2015-01-01, 2015-01-02, 2015-01-03, 2015-01-04,␣
↪2015-01-05]

In [14]: date.dt.day
Out[14]:
0    1
1    2
2    3
3    4
4    5
Length: 5, dtype: int64
```

- `pivot_table` now has a `margins_name` argument so you can use something other than the default of 'All' (GH3335)
- Implement export of `datetime64[ns, tz]` dtypes with a fixed HDF5 store (GH11411)
- Pretty printing sets (e.g. in DataFrame cells) now uses set literal syntax (`{x, y}`) instead of Legacy Python syntax (`set([x, y])`) (GH11215)
- Improve the error message in `pandas.io.gbq.to_gbq()` when a streaming insert fails (GH11285) and when the DataFrame does not match the schema of the destination table (GH11359)

API changes

- raise `NotImplementedError` in `Index.shift` for non-supported index types (GH8038)
- `min` and `max` reductions on `datetime64` and `timedelta64` dtyped series now result in `NaT` and not `nan` (GH11245).
- Indexing with a null key will raise a `TypeError`, instead of a `ValueError` (GH11356)
- `Series.ptp` will now ignore missing values by default (GH11163)

Deprecations

- The `pandas.io.ga` module which implements `google-analytics` support is deprecated and will be removed in a future version (GH11308)
- Deprecate the `engine` keyword in `.to_csv()`, which will be removed in a future version (GH11274)

Performance improvements

- Checking monotonic-ness before sorting on an index (GH11080)
- `Series.dropna` performance improvement when its dtype can't contain `NaN` (GH11159)
- Release the GIL on most datetime field operations (e.g. `DatetimeIndex.year`, `Series.dt.year`), normalization, and conversion to and from `Period`, `DatetimeIndex.to_period` and `PeriodIndex.to_timestamp` (GH11263)
- Release the GIL on some rolling algos: `rolling_median`, `rolling_mean`, `rolling_max`, `rolling_min`, `rolling_var`, `rolling_kurt`, `rolling_skew` (GH11450)
- Release the GIL when reading and parsing text files in `read_csv`, `read_table` (GH11272)
- Improved performance of `rolling_median` (GH11450)
- Improved performance of `to_excel` (GH11352)
- Performance bug in repr of `Categorical` categories, which was rendering the strings before chopping them for display (GH11305)
- Performance improvement in `Categorical.remove_unused_categories`, (GH11643).
- Improved performance of `Series` constructor with no data and `DatetimeIndex` (GH11433)
- Improved performance of `shift`, `cumprod`, and `cumsum` with groupby (GH4095)

Bug fixes

- `SparseArray.__iter__()` now does not cause `PendingDeprecationWarning` in Python 3.5 (GH11622)
- Regression from 0.16.2 for output formatting of long floats/nan, restored in (GH11302)
- `Series.sort_index()` now correctly handles the `inplace` option (GH11402)
- Incorrectly distributed .c file in the build on `PyPi` when reading a csv of floats and passing `na_values=<a scalar>` would show an exception (GH11374)
- Bug in `.to_latex()` output broken when the index has a name (GH10660)

- Bug in `HDFStore.append` with strings whose encoded length exceeded the max unencoded length (GH11234)
- Bug in merging `datetime64[ns, tz]` dtypes (GH11405)
- Bug in `HDFStore.select` when comparing with a numpy scalar in a where clause (GH11283)
- Bug in using `DataFrame.ix` with a MultiIndex indexer (GH11372)
- Bug in `date_range` with ambiguous endpoints (GH11626)
- Prevent adding new attributes to the accessors `.str`, `.dt` and `.cat`. Retrieving such a value was not possible, so error out on setting it. (GH10673)
- Bug in tz-conversions with an ambiguous time and `.dt` accessors (GH11295)
- Bug in output formatting when using an index of ambiguous times (GH11619)
- Bug in comparisons of Series vs list-likes (GH11339)
- Bug in `DataFrame.replace` with a `datetime64[ns, tz]` and a non-compat to_replace (GH11326, GH11153)
- Bug in `isnull` where `numpy.datetime64('NaT')` in a `numpy.array` was not determined to be null(GH11206)
- Bug in list-like indexing with a mixed-integer Index (GH11320)
- Bug in `pivot_table` with `margins=True` when indexes are of `Categorical` dtype (GH10993)
- Bug in `DataFrame.plot` cannot use hex strings colors (GH10299)
- Regression in `DataFrame.drop_duplicates` from 0.16.2, causing incorrect results on integer values (GH11376)
- Bug in `pd.eval` where unary ops in a list error (GH11235)
- Bug in `squeeze()` with zero length arrays (GH11230, GH8999)
- Bug in `describe()` dropping column names for hierarchical indexes (GH11517)
- Bug in `DataFrame.pct_change()` not propagating `axis` keyword on `.fillna` method (GH11150)
- Bug in `.to_csv()` when a mix of integer and string column names are passed as the `columns` parameter (GH11637)
- Bug in indexing with a `range`, (GH11652)
- Bug in inference of numpy scalars and preserving dtype when setting columns (GH11638)
- Bug in `to_sql` using unicode column names giving UnicodeEncodeError with (GH11431).
- Fix regression in setting of `xticks` in `plot` (GH11529).
- Bug in `holiday.dates` where observance rules could not be applied to holiday and doc enhancement (GH11477, GH11533)
- Fix plotting issues when having plain `Axes` instances instead of `SubplotAxes` (GH11520, GH11556).
- Bug in `DataFrame.to_latex()` produces an extra rule when `header=False` (GH7124)
- Bug in `df.groupby(...).apply(func)` when a func returns a `Series` containing a new datetimelike column (GH11324)
- Bug in `pandas.json` when file to load is big (GH11344)
- Bugs in `to_excel` with duplicate columns (GH11007, GH10982, GH10970)
- Fixed a bug that prevented the construction of an empty series of dtype `datetime64[ns, tz]` (GH11245).

- Bug in `read_excel` with MultiIndex containing integers (GH11317)
- Bug in `to_excel` with openpyxl 2.2+ and merging (GH11408)
- Bug in `DataFrame.to_dict()` produces a `np.datetime64` object instead of `Timestamp` when only datetime is present in data (GH11327)
- Bug in `DataFrame.corr()` raises exception when computes Kendall correlation for DataFrames with boolean and not boolean columns (GH11560)
- Bug in the link-time error caused by C `inline` functions on FreeBSD 10+ (with `clang`) (GH10510)
- Bug in `DataFrame.to_csv` in passing through arguments for formatting `MultiIndexes`, including `date_format` (GH7791)
- Bug in `DataFrame.join()` with `how='right'` producing a `TypeError` (GH11519)
- Bug in `Series.quantile` with empty list results has `Index` with `object` dtype (GH11588)
- Bug in `pd.merge` results in empty `Int64Index` rather than `Index(dtype=object)` when the merge result is empty (GH11588)
- Bug in `Categorical.remove_unused_categories` when having `NaN` values (GH11599)
- Bug in `DataFrame.to_sparse()` loses column names for MultiIndexes (GH11600)
- Bug in `DataFrame.round()` with non-unique column index producing a Fatal Python error (GH11611)
- Bug in `DataFrame.round()` with `decimals` being a non-unique indexed Series producing extra columns (GH11618)

Contributors

A total of 63 people contributed patches to this release. People with a "+" by their names contributed a patch for the first time.

- Aleksandr Drozd +
- Alex Chase +
- Anthonios Partheniou
- BrenBarn +
- Brian J. McGuirk +
- Chris
- Christian Berendt +
- Christian Perez +
- Cody Piersall +
- Data & Code Expert Experimenting with Code on Data
- DrIrv +
- Evan Wright
- Guillaume Gay
- Hamed Saljooghinejad +
- Iblis Lin +
- Jake VanderPlas

- Jan Schulz
- Jean-Mathieu Deschenes +
- Jeff Reback
- Jimmy Callin +
- Joris Van den Bossche
- K.-Michael Aye
- Ka Wo Chen
- Loïc Séguin-C +
- Luo Yicheng +
- Magnus Jöud +
- Manuel Leonhardt +
- Matthew Gilbert
- Maximilian Roos
- Michael +
- Nicholas Stahl +
- Nicolas Bonnotte +
- Pastafarianist +
- Petra Chong +
- Phil Schaf +
- Philipp A +
- Rob deCarvalho +
- Roman Khomenko +
- Rémy Léone +
- Sebastian Bank +
- Sinhrks
- Stephan Hoyer
- Thierry Moisan
- Tom Augspurger
- Tux1 +
- Varun +
- Wieland Hoffmann +
- Winterflower
- Yoav Ram +
- Younggun Kim
- Zeke +
- ajcr