Deliverable 2

Team_01 (GitHub):
Abel Debalkew
Carl Alvares
Jaya Thirugnanasampanthan
Kemar Harris
Lan Yao
Seemin Syed

Table of Contents

Bugs

Team FOSSkateers investigated 5 bugs reported in the pandas project. They have implemented fixes for bugs 1 and 2 (Issues #14467 and 13094).

Bug 1 (Fixed Issue #14467)

Link: Dataframe constructor misinterprets `columns` argument if nested list is passed in as the `data` parameter. · Issue #14467 · pandas-dev/pandas

Labels: Bug, Reshaping

Version:

Development: 1.1.0.dev0+578.ge88629f0c
Production: 1.0.1

Reproducible example:

```
>>>import pandas as pd
>>>df = pd.DataFrame([[1, 2, 3], [4, 5, 6]],
        index=[['gibberish']*2, [0, 1]],
        columns=[['baldersash']*3, [10, 20, 30]])
```

Output:

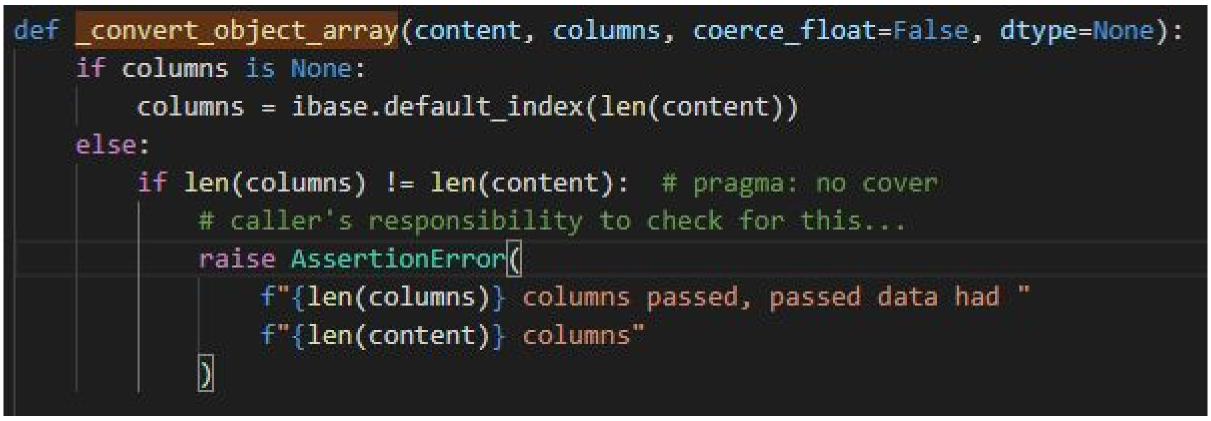
```
PS C:\Users\kemar\Documents\School\CSCD01\Project\repo\team_01-project\pandas
repo\pandas> python columnBug.py
Traceback (most recent call last):
  File "C:\Users\kemar\Documents\School\CSCD01\Project\repo\team_01-project\pandas
repo\pandas\pandas\core\internals\construction.py", line 495, in _list_to_arrays
    result = _convert_object_array(
  File "C:\Users\kemar\Documents\School\CSCD01\Project\repo\team_01-project\pandas
repo\pandas\pandas\core\internals\construction.py", line 579, in _convert_object_array
    raise AssertionError(
AssertionError: 2 columns passed, passed data had 3 columns

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "columnBug.py", line 3, in <module>
    df = pd.DataFrame([[1, 2, 3], [4, 5, 6]],
  File "C:\Users\kemar\Documents\School\CSCD01\Project\repo\team_01-project\pandas
repo\pandas\pandas\core\frame.py", line 480, in __init__
    arrays, columns = to_arrays(data, columns, dtype=dtype)
  File "C:\Users\kemar\Documents\School\CSCD01\Project\repo\team_01-project\pandas
repo\pandas\pandas\core\internals\construction.py", line 460, in to_arrays
    return _list_to_arrays(data, columns, coerce_float=coerce_float, dtype=dtype)
  File "C:\Users\kemar\Documents\School\CSCD01\Project\repo\team_01-project\pandas
repo\pandas\pandas\core\internals\construction.py", line 499, in _list_to_arrays
    raise ValueError(e) from e
ValueError: 2 columns passed, passed data had 3 columns
```

Expected:

```
             baldersash
             10 20 30
gibberish  0  1  2  3
           1  4  5  6
```

Root of bug:

File to edit: pandas/core/internals/construction.py



```python
def _convert_object_array(content, columns, coerce_float=False, dtype=None):
    if columns is None:
        columns = ibase.default_index(len(content))
    else:
        if len(columns) != len(content):  # pragma: no cover
            # caller's responsibility to check for this...
            raise AssertionError(
                f"{len(columns)} columns passed, passed data had "
                f"{len(content)} columns"
            )
```

In pandas, the user is able to pass a columns parameter to specify the columns they want for their dataframe. On line 578 of construction.py, in _convert_object_array(), the function checks the length of the data passed in and makes sure it matches the length of the columns passed in. If it doesn't, a ValueError is raised. The issue occurs because the code doesn't check if the column passed in was intended to be a MultiIndex. To fix this, we need to add code that checks if the columns were intended to be a MultiIndex before continuing with this check.

Estimate of work:

Again, to try to fix this issue, we would have to add an if statement that checks if the columns passed were a MultiIndex. It would take 3 or 4 members to fix this bug because this code is tight coupled to the rest of the codebase. It would take roughly 8 hours to finish. The first hour would be spent understanding the code flow works with MultiIndex. The next hour would be spent looking at how the numpy.array() version of this implements the solution. The next hour will be spent implementing the fix, and ensuring it matches the code-flow of the numpy.array() version. The rest of the time will be used for debugging and making sure the implemented solution does not break any existing functionality.

Arguments:

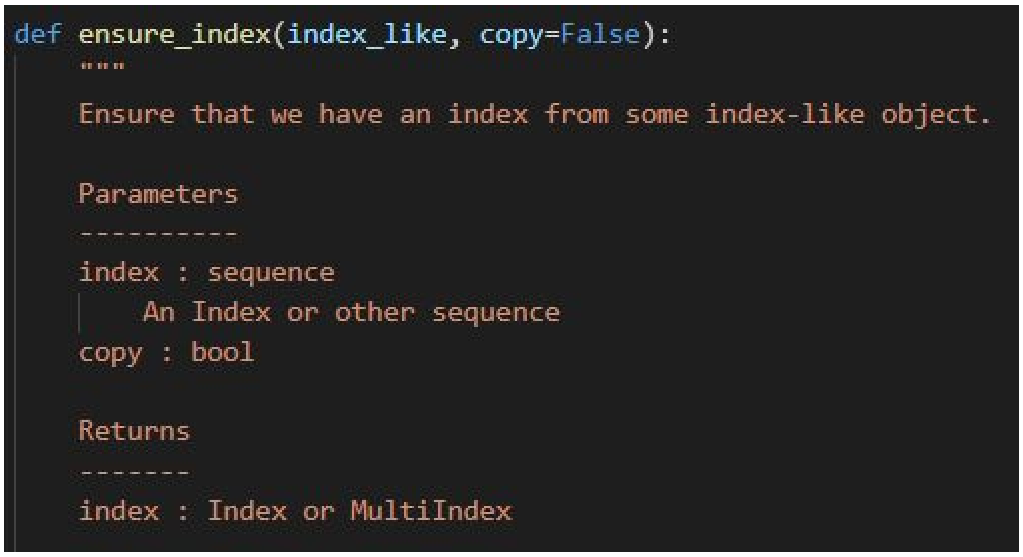
Because it works with numpy.array (but not nested list), we basically have an example somewhere in code with a working solution. This will let us follow the reuse-oriented model by reusing the parts of the components that have already been implemented to fix this bug. Also, it deals with multiple levels of pandas as we have to investigate how the numpy.array constructor for dataframe works and the control flow for the constructor itself. This will give us an in-depth understanding of how pandas works, and more experience in working with pandas' low level code.

In addition, as we will be reusing code that works reliably, we can anticipate that our bug fix will work reliably too. This is important for us as we want to deliver value to users.

Anticipated risks:

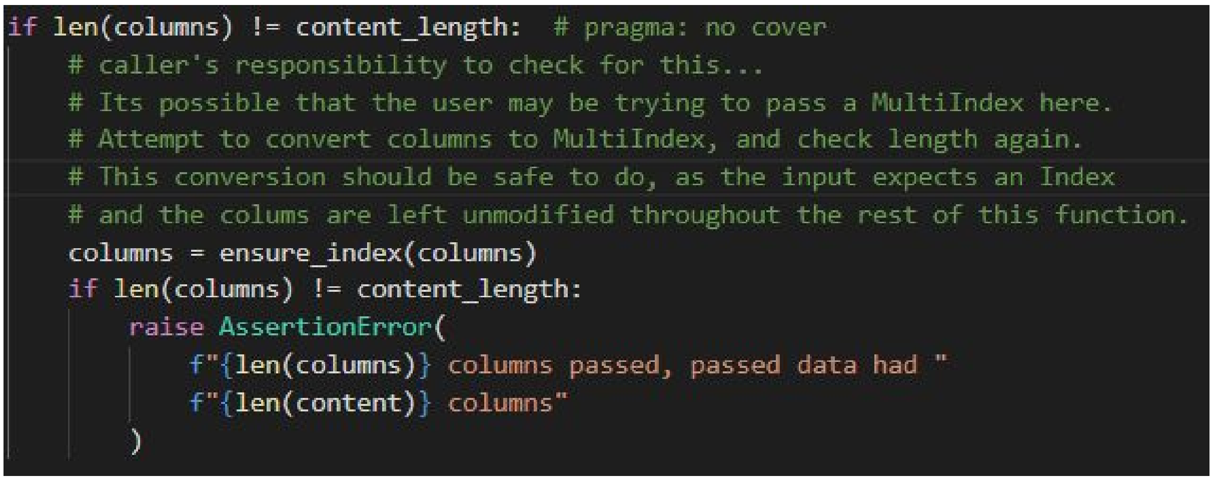
It is possible that our fix unintentionally introduces new bugs. To mitigate this risk, we will (and are required to if merging to the pandas codebase) run the pandas test suite after implementing our fix (i.e., regression testing). We acknowledge that the test suite itself carries a risk as it may not necessarily be exhaustive. Thus, in addition to the pandas test suite, we will brainstorm specific tests and run these new tests against our fix.

Code to reuse:
pandas/core/indexes/base.py, line: 5468



```python
def ensure_index(index_like, copy=False):
    """
    Ensure that we have an index from some index-like object.

    Parameters
    ----------
    index : sequence
        An Index or other sequence
    copy : bool

    Returns
    -------
    index : Index or MultiIndex
```

New reused code:
pandas/core/internals/construction.py, line 578



```python
if len(columns) != content_length:  # pragma: no cover
    # caller's responsibility to check for this...
    # Its possible that the user may be trying to pass a MultiIndex here.
    # Attempt to convert columns to MultiIndex, and check length again.
    # This conversion should be safe to do, as the input expects an Index
    # and the colums are left unmodified throughout the rest of this function.
    columns = ensure_index(columns)
    if len(columns) != content_length:
        raise AssertionError(
            f"{len(columns)} columns passed, passed data had "
            f"{len(content)} columns"
        )
```

Bug 2 (Fixed Issue #13094)

Link: ERR: warning on merging on unequal levels for an Index · Issue #13094 · pandas-dev/pandas

Labels: Bug, Error Reporting, Reshaping, good first issue

Version:

Identified in older releases but present in:

Development: 1.1.0.dev0+689.gbdb4a0830.dirty
Production: 1.0.1

Reproducible example:

```
>>>import pandas as pd
>>>X = pd.DataFrame([[2, 3], [5, 7]], columns=['a','p']).set_index('a')
#    p
# a
# 2  3
#5  7
>>>Y = pd.DataFrame([[1, 2, 3], [3, 4, 8], [5, 6, 9]],
columns=['a','b','c']).set_index(['a','b'])
#      c
# a b
# 1 2  3
# 3 4  8
# 5 6  9
>>>X.join(Y, how='left')
```

Actual:

```
#    p  c
# a b
# 5 6  7  9
```

Expected:

```
~\pandas\pandas\core\reshape\merge.py:623: UserWarning: merging between
different levels can give an unintended result (1 levels on the left,2 on the right)
  warnings.warn(msg, UserWarning)
#    p  c
# a b
# 5 6  7  9
```

Root of bug:

File to edit: ~pandas\pandas\core\reshape\merge.py

Merging on unequal index levels should trigger a warning that the user is not merging on all levels, but is not triggered in this specific case as the warning triggers on level comparison on the columns only rather than columns and indexes.

Estimate of work:

The difficulty in this bug fix is not in writing new code, as the trigger and warning are already present, but to find out why this edge case is not accounted for, and to integrate checks for it into the code base by tracing the program. As such, the time taken to implement the fix would be under an hour to trace the program and write the check, as well as another hour to add test cases and insure the change did not break other functionality.

Arguments:

The warning and checks for the general case of this bug are already implemented in the code, so the fix is less about modifying the code and more about understanding the code base, specifically the interactions behind the index and reshaping packages. Once the original checks have been analysed, said code will be available for modification to solve the current issue as well as future issues related to this, resulting in reliable behaviour that keep users informed during their use of the related merge function.

Anticipated risks:

Because of a lack of familiarity with the code base, and the standard behaviours pandas had adopted, there may be inconsistencies between the developers' understanding of what is required and what is actually required. This risk can be mitigated by internal discussion, as well as communication with the original posters of the issue.

Another possible risk is if the fix causes a change in expected behaviour in other use cases, in which case the already present test suites and the tests added by the team will help diminish said risk.

Reused component:

Original check in merge.py on whether two DataFrames have different column.nlevels and issue a warning if so.

Code to reuse:

```
        # warn user when merging between different levels
        if _left.columns.nlevels != _right.columns.nlevels:
            msg = (
                "merging between different levels can give an unintended "
                f"result ({left.columns.nlevels} levels on the left,"
                f"{right.columns.nlevels} on the right)"
            )
            warnings.warn(msg, UserWarning)
```

New reused code:

```
        elif _left.index.nlevels != _right.index.nlevels:
            msg = (
                "merging between different levels can give an unintended "
                f"result ({left.index.nlevels} levels on the left,"
                f"{right.index.nlevels} on the right)"
            )
            warnings.warn(msg, UserWarning)
```

Bug 3

Link: BUG: merge raises for how='outer'/'right' when duplicate suffixes are specified · Issue #29697 · pandas-dev/pandas

Labels: Bug, Reshaping

Reproducible example:

- Input:
  >> import pandas as pd

```
>> df1 = pd.DataFrame({'A': list('ab'), 'B': [0, 1]})
>> df2 = pd.DataFrame({'A':list('ac'), 'B': [100, 200]})
>> pd.merge(df1, df2, on="A", how="outer", suffixes=("_x", "_x"))
```

- Output:
```
ValueError: Buffer has wrong number of dimensions (expected 1, got 0)
```

- Expected:
```
  A  B_x    B_x
0  a  0.0  100.0
1  b  1.0    NaN
2  c  NaN  200.0
```

Root of bug:

File to edit: ~/pandas/core/reshape/merge.py

There is a difference in how duplicate suffixes are handled in the case when the "how" parameter of the the "merge" method is set to "outer" or "right" compared to when it is set as "inner" or "left".

Estimate of work:

It will take a team of 2 people around 4 hours to fix this bug. It will require us to replicate the handling of duplicate suffixes from the 2 cases which have the expected output into the 2 cases which throw the ValueError incorrectly. After this, we will need to budget around 2 hours for unit testing and acceptance testing, in order to ensure that these changes to the merge method do not affect any other expected behaviour.

Bug 4

Link: SparseDataFrame.to_coo does not convert the default fill value when is not 0 · Issue #24817 · pandas-dev/pandas

Labels: Missing data, sparse, good first issue

Version: not specified (0.24.0rc1)

Reproducible example:

- Input:
```
>> import pandas as pd
>> df = pd.DataFrame({"A": pd.SparseArray([1, 1, 1, 2], fill_value=1)})

>> df.sparse.to_coo().todense()
```

- Output:
```
matrix([[0],
        [0],
        [0],
        [2]])
```

- Expected

  Should instead raise a ValueError with an informative message

Root of bug:

File to edit: ~/pandas/core/arrays/sparse/accessor.py

Root lies in the fact that the fill value of Data Frame is not checked, since to_coo() method depends on the SciPy library, and SciPy only supports it when the fill value of Data Frame is 0. Therefore, an if statement should be included on line 147 and check the value of the array. Then, a message should be displayed to the user, pointing them to change the fill value before calling this method.

Estimate of work:

It will take a team of 2 or 3 people to finish in about 6 hours. About 4 hours will be implementation and the remaining few hours for creating test cases like unit tests and acceptance tests. Since it will be a fairly simple fix, but there are 2 things that might need more time to explore for actual implementation:

1. How to check the fill value of data frame in the file to edit?
2. to_coo() method also impacts Series and not just Data Frame, will Series be affected?

Bug 5

Link: round method throws error if DataFrame includes Int64Dtype · Issue #31478 · pandas-dev/pandas

Labels: Algos, NA - Masked Arrays

Version:

Identified in Pandas 1.0.0

Reproducible example:

```
>>> import pandas as pd
>>> s = pd.Series([1, 2, None], dtype=”Int64”)
>>> s.round(1)
```

Output:

```
AttributeError: 'float' object has no attribute 'rint'
The above exception was the direct cause of the following exception:
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/local/lib/python3.7/site-packages/pandas/core/series.py", line 2145, in round
    result = com.values_from_object(self).round(decimals)
TypeError: loop of ufunc does not support argument 0 of type float which has no callable rint
method
```

Problem description:

The current behaviour is to throw an error as above but it should behave similar to setting the dtype=”int64” which ignores the NaN value.

```
>>> import pandas as pd
>>> s = pd.Series([1, 2, None], dtype=”int64”)
>>> s.round(1)
```

0   1
1   2
dtype: int64

Expected output:
0   1
1   2
2   <Na>
Dtype: Int64

Root of bug:

File to edit: ~/pandas/core/series.py

The bug lies in ~/pandas/core/series.py on line 2096 where the round function is called. This is due to the function being built on Numpy which has a data type “int64”, but “Int64” is a Pandas data type. Therefore, there needs to be a change that would allow the rounding to work for the pandas data type.

```
2094        """
2095        nv.validate_round(args, kwargs)
2096        result = com.values_from_object(self).round(decimals)
2097        result = self._constructor(result, index=self.index).__finalize__(self)
2098
2099        return result
2100
```

Estimate of work:

A possible solution for this issue is to create a rounding method for ExtensionArrays, an abstract base class for 1-D arrays. The solution would take 2 or 3 members about 1 or 2 days to finish. It would first require 2-3 hours to review interactions between relevant parts of the code base and explore reusable components like library functions. Design, testing, and implementation of the solution should take 4 or 5 hours.

Customer Acceptance Tests

Bug 1 (Issue #14467)

In a terminal, open Python and import the pandas library (i.e, import pandas as pd).

```
import pandas as pd
```

Initialize a DataFrame with a nested list passed as the ‘data’ argument (e.g., [[1, 2, 3], [4, 5, 6]]), along with the ‘index’ and ‘columns’ arguments.

DataFrame example:

```
df = pd.DataFrame([[1, 2, 3], [4, 5, 6]],
                  index=[['gibberish']*2, [0, 1]],
                  columns=[['baldersash']*3, [10, 20, 30]])
```

Check the output of the newly created DataFrame (by printing df, if the example above is used).

The expected output for the example given above is:

```
             baldersash
                     10 20 30
gibberish 0           1  2  3
          1           4  5  6
```

Bug 2 (Issue #13094)

In a terminal, open Python and import the pandas library (i.e, import pandas as pd).

```
import pandas as pd
```

Initialize two DataFrames with different columns and indexes, such that some indexes are shared and some are not.

```
X = pd.DataFrame([[2, 3], [5, 7]], columns=['a','p']).set_index('a')
#    p
# a
# 2  3
# 5  7

Y = pd.DataFrame([[1, 2, 3], [3, 4, 8], [5, 6, 9]],
        columns=['a','b','c']).set_index(['a','b'])
#      c
# a b
# 1 2  3
# 3 4  8
# 5 6  9
```

Join or Merge the two on a shared index.

```
X.join(Y, how='left')
```

As you are intending to merge on unequal index levels, a console warning is triggered, although the merge is executed as well.

```
~\pandas\pandas\core\reshape\merge.py:623:
  UserWarning: merging between different levels can give an unintended
  result (1 levels on the left,2 on the right)
  warnings.warn(msg, UserWarning)

#      p c
# a b
# 5 6  7 9
```

Technical Commentary

Bug 1 (Issue #14467)

Files looked at:
- pandas/core/frame.py
- pandas/core/internals/construction.py
- pandas/core/indexes/base.py

Files modified:
- pandas/core/internals/construction.py
- pandas/tests/indexes/multi/test_nested_list_data.py

The team first traced the bug to where the error occurred. They found that the bug occured when the length of the data passed to the columns parameter did not match the length of the data passed to the data parameter. Then they followed the numpy.array version of the code, which had a working example. In the code, they realized that the numpy.array version converted the columns of a DataFrame to an Index before continuing, while the nested-list version did not. An Index takes into consideration the possibility of a nested list, and uses the lengths of the inner list to calculate its length as opposed to the length of the outer list. To fix this bug, in the case that the length of the data did not match the length of the columns, the team re-used the ensure_index() function, which converts a set of array data to an Index. Then, they checked the length of the generated Index, and if it did not match the length of the data, the previous error was thrown.

In the function outside of where the bug occurs, the code calls ensure_index() on the next line on the columns parameter. In our fix, in the function where the bug occurs _convert_object_array(), we modify the return value of columns and change it to an Index. We looked into ensure_index(), and in the code if the parameter passed is already an index, it is returned as normal. Also, beforehand, _convert_object_array didn't change the value of columns, and returned an Index if the value of columns was None, implying that converting columns to an Index is a safe operation.

Bug 2 (Issue #13094)

Files modified:
- pandas/core/reshape/merge.py
- pandas/tests/frame/test_axis_select_reindex.py

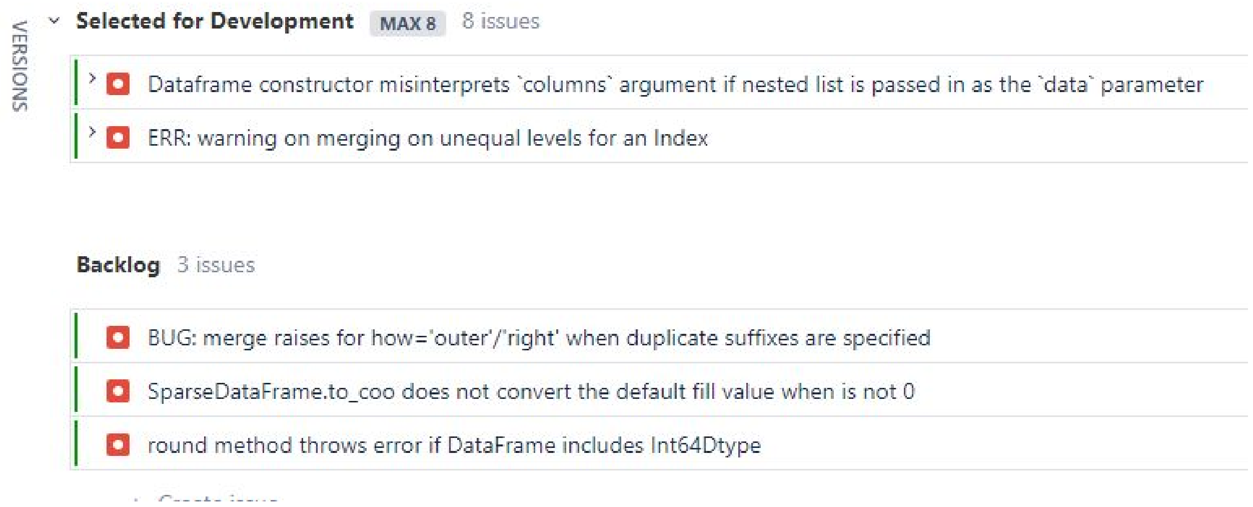
After looking at how merge operation is implemented, the team found out that the reusable component was how the warning was triggered when merging two DataFrames with different levels of columns in merge.py. This bug required us to check the different index levels of two DataFrames, which followed a similar pattern. Therefore, we re-used that component but made modifications like changing the accessed fields and the wording of the warning to solve this issue.

Software Development Process

As discussed in our previous deliverable, we will be using an agile development process. This process involves using Kanban and the reuse-oriented model.

Kanban

The team investigated and documented 5 bugs in the backlog. After having a meeting (which was a blend of the Delivery Planning and Replenishment meetings), the team selected 2 bugs to fix and moved their respective cards to the "Selected for Development" column.


VERSIONS

Selected for Development MAX 8 8 issues

- Dataframe constructor misinterprets `columns` argument if nested list is passed in as the `data` parameter
- ERR: warning on merging on unequal levels for an Index

Backlog 3 issues

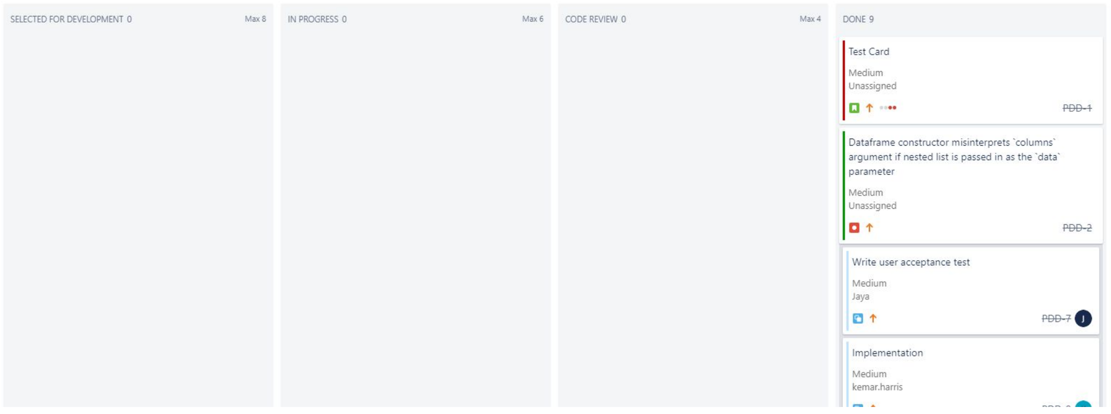
- BUG: merge raises for how='outer'/'right' when duplicate suffixes are specified
- SparseDataFrame.to_coo does not convert the default fill value when is not 0
- round method throws error if DataFrame includes Int64Dtype

We had fewer Daily meetings (approximately 10 minute discussions via Google Hangouts with the Kanban board guiding the meeting) than expected as the actual deliverable did not take more than 2 days to complete.

These bug cards were then broken down into subtasks, prioritized, and rearranged on the board, with the top cards having the highest importance. Team members were reminded that new tasks should be pulled only if the WIP limit in any column hadn't been reached and tasks should be pulled from the top (due to having higher importance).

In regards to the WIP limits, we found that we needed to modify our WIP limits as the project progressed as we found that the team could take on further work without issue.

(L to R): Selected for Development (WIP limit: 8),
In Progress (WIP limit: 6), Code Review (WIP limit: 4), Done


SELECTED FOR DEVELOPMENT 0 Max 8

IN PROGRESS 0 Max 6

CODE REVIEW 0 Max 4

DONE 9

Test Card
Medium
Unassigned
~~PDD-1~~

Dataframe constructor misinterprets `columns` argument if nested list is passed in as the `data` parameter
Medium
Unassigned
~~PDD-2~~

Write user acceptance test
Medium
Jaya
~~PDD-7~~ J

Implementation
Medium
kemar.harris

Reuse-Oriented Model

The process activities in our software development process were:

- Requirements Specification: Here, the team gathered functional requirements for the two bugs chosen to be fixed. This involved reproducing the bug locally using the data given by the bug reporter. In addition, the team clarified requirements with pandas members and/or bug reporters.

- Component Analysis: Here, the team investigated the code to see if there were reusable components to be used in our solution. These reusable components were discussed earlier in this document.

- System Design With Reuse: Here, the team discussed how the fixes would be designed with the reusable components. This is discussed in the “Technical Commentary” section.

- Development and Integration: Here, the team implemented the fixes and wrote unit tests verifying that the fixes were implemented according to the requirement specifications. The fixes were implemented in separate branches (following best practices in the pandas code guidelines) and then merged to the master branch after code review. Code review was done by other team members to make sure the specifications were met and the fix was implemented with reusable components if applicable.

- Validation: Here, the team ran acceptance tests (designed with the data provided in the bug ticket) to check that the team’s bug fix actually addressed what the customer wanted.

These process activities were interleaved as requirements were clarified throughout the development process.

Reflection of the Software Development Process

The Kanban board helped the team be efficient because the team had fewer things to document and the board kept the team updated on what everyone was doing. A drawback that was anticipated with the Kanban board is that since there are fewer meetings held, everyone relies heavily on the board being always up to date. However, we found that the team was diligent in keeping the board up to date. In addition, the WIP limits ensured that unreviewed tasks did not pile up, which helped the team focus on finishing current tasks rather than start new ones.

The reuse-oriented model also helped the team be efficient since the team was able to reuse code to address the bugs. Consequently, the team spent less time coding. However, this meant that the team actually spent more time on the “Component Analysis” process activity. This was fine with the team since looking for reusable components increased our overall understanding of the code base. In addition, since the reusable components have already been tested, they would be reliable to include as part of our own bug fixes, which in turn increases the chances of our bug fixes being merged to the pandas code base.