Getting Started with pandas

Part 7

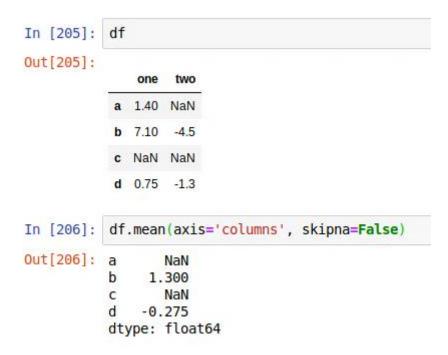
Summarizing and Computing Descriptive Statistics

Part 1

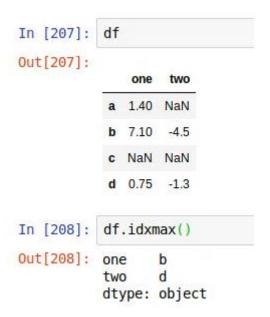
- pandas objects are equipped with a set of common mathematical and statistical methods.
- Most of these fall into the category of reductions or summary statistics, methods that extract a single value (like the sum or mean) from a Series or a Series of values from the rows or columns of a DataFrame.
- Compared with the similar methods found on NumPy arrays, they have built-in handling for missing data.

```
In [201]: df = pd.DataFrame([[1.4, np.nan], [7.1, -4.5],
                                [np.nan, np.nan], [0.75, -1.3]],
index=['a', 'b', 'c', 'd'],
columns=['one', 'two'])
            df
Out[201]:
                one two
             a 1.40 NaN
             b 7.10 -4.5
            c NaN NaN
             d 0.75 -1.3
In [202]: df.sum()
Out[202]: one
                   9.25
            two -5.80
            dtype: float64
In [203]: df.sum(axis='columns')
Out[203]: a
                 1.40
                 2.60
                 0.00
                -0.55
            dtype: float64
```

- NA values are excluded unless the entire slice (row or column in this case) is NA.
- This can be disabled with the skipna option:



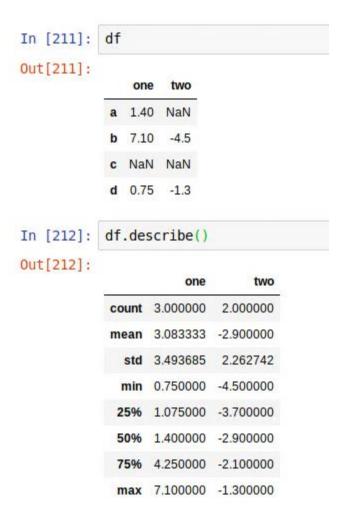
• Some methods, like idxmin and idxmax, return indirect statistics like the index value where the minimum or maximum values are attained:



• Other methods are accumulations:

```
In [209]: df
Out[209]:
              one two
           a 1.40 NaN
           b 7.10 -4.5
           c NaN NaN
           d 0.75 -1.3
In [210]: df.cumsum()
Out[210]:
           a 1.40 NaN
           b 8.50 -4.5
           c NaN NaN
           d 9.25 -5.8
```

- Another type of method is neither a reduction nor an accumulation.
- describe is one such example, producing multiple summary statistics in one shot:



• On non-numeric data, describe produces alternative summary statistics:

```
In [213]: obj = pd.Series(['a', 'a', 'b', 'c'] * 4)
In [214]: obj
Out[214]: 0
          10
          11
          12
          13
          14
          dtype: object
In [215]: obj.describe()
Out[215]: count
                    16
          unique
                     3
          top
                     a
          freq
          dtype: object
```

Correlation and Covariance

- Some summary statistics, like correlation and covariance, are computed from pairs of arguments.
- Let's consider some DataFrames of stock prices and volumes obtained from Yahoo! Finance using the add-on pandas-datareader package.
- If you don't have it installed already, it can be obtained via conda or pip:

```
(base) joshua@joshua-Virtual-Machine:~$ conda install pandas-datareader
Collecting package metadata: done
Solving environment: done

## Package Plan ##

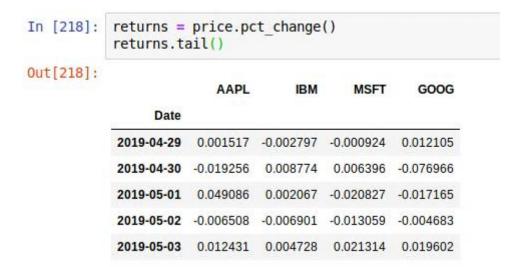
environment location: /home/joshua/anaconda3

added / updated specs:
- pandas-datareader
```

```
(base) joshua@joshua-Virtual-Machine:~$ pip install fix yahoo finance --upgrade --no-cache-dir
Collecting fix yahoo finance
  Downloading https://files.pythonhosted.org/packages/0c/b3/55468a37787c0a32d0e26155bdd2c712469d9
5071a125884fc1ff149b75c/fix-yahoo-finance-0.1.33.tar.gz
Requirement already satisfied, skipping upgrade: pandas>=0.24 in ./anaconda3/lib/python3.7/site-p
ackages (from fix yahoo finance) (0.24.2)
Requirement already satisfied, skipping upgrade: numpy>1.15 in ./anaconda3/lib/python3.7/site-pac
kages (from fix yahoo finance) (1.16.2)
Collecting requests<2.20 (from fix yahoo finance)
  Downloading https://files.pythonhosted.org/packages/65/47/7e02164a2a3db50ed6d8a6ab1d6d60b69c4c3
fdf57a284257925dfc12bda/requests-2.19.1-py2.py3-none-any.whl (91kB)
    100%
                                           92kB 297kB/s
Collecting multitasking>=0.0.7 (from fix yahoo finance)
 Downloading https://files.pythonhosted.org/packages/ac/1a/0750416c5e3683d170757e423f097fdf78ceb
9ccdc65658b24341664e53e/multitasking-0.0.7.tar.gz
Requirement already satisfied, skipping upgrade: python-dateutil>=2.5.0 in ./anaconda3/lib/python
3.7/site-packages (from pandas>=0.24->fix vahoo finance) (2.8.0)
```

• The pandas_datareader module (together with fix-yahoo-finance) can be used to download some data for a few stock tickers:

• We can now compute percent changes of the prices.



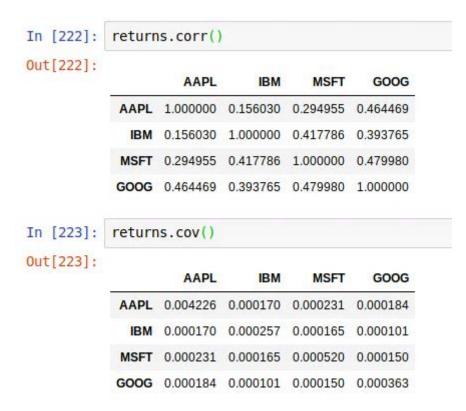
- The corr method of Series computes the correlation of the overlapping, non-NA, aligned-by-index values in two Series.
- Relatedly, COV computes the covariance:

```
In [219]: returns['MSFT'].corr(returns['IBM'])
Out[219]: 0.4177864219010938
In [220]: returns['MSFT'].cov(returns['IBM'])
Out[220]: 0.00016455420082779368
```

• Since MSFT is a valid Python attribute, we can also select these columns using more concise syntax:

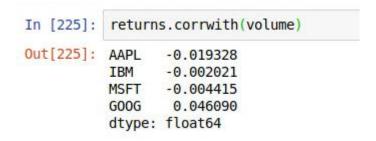
```
In [221]: returns.MSFT.corr(returns.IBM)
Out[221]: 0.4177864219010938
```

 DataFrame's corr and cov methods, on the other hand, return a full correlation or covariance matrix as a DataFrame, respectively:



- Using DataFrame's corrwith method, you can compute pairwise correlations between a DataFrame's columns or rows with another Series or DataFrame.
- Passing a Series returns a Series with the correlation value computed for each column:

- Passing a DataFrame computes the correlations of matching column names.
- Here I compute correlations of percent changes with volume:



- Passing axis='columns' does things row-by-row instead.
- In all cases, the data points are aligned by label before the correlation is computed.

Unique Values, Value Counts, and Membership

- Another class of related methods extracts information about the values contained in a one-dimensional Series.
- To illustrate these, consider this example:

```
In [226]: obj = pd.Series(['c', 'a', 'd', 'a', 'a', 'b', 'b', 'c', 'c'])
```

 The first function is unique, which gives you an array of the unique values in a Series:

```
In [227]: uniques = obj.unique()
uniques

Out[227]: array(['c', 'a', 'd', 'b'], dtype=object)
```

• The unique values are not necessarily returned in sorted order, but could be sorted after the fact if needed (uniques.sort()).

• value counts computes a Series containing value frequencies:

 value_counts is also available as a top-level pandas method that can be used with any array or sequence:

• isin performs a vectorized set membership check and can be useful in filtering a dataset down to a subset of values in a Series or column in a DataFrame:

```
In [230]: obj
Out[230]:
          dtype: object
In [231]: mask = obj.isin(['b', 'c'])
          mask
Out[231]: 0
                True
               False
               False
               False
               False
                True
                True
                True
                True
          dtype: bool
In [232]:
          obj[mask]
Out[232]: 0
          dtype: object
```

• Related to isin is the Index.get_indexer method, which gives you an index array from an array of possibly non-distinct values into another array of distinct values:

```
In [233]: to_match = pd.Series(['c', 'a', 'b', 'b', 'c', 'a'])
    unique_vals = pd.Series(['c', 'b', 'a'])
    pd.Index(unique_vals).get_indexer(to_match)

Out[233]: array([0, 2, 1, 1, 0, 2])
```

• In some cases, you may want to compute a histogram on multiple related columns in a DataFrame.

```
In [234]: data = pd.DataFrame({'Qul': [1, 3, 4, 3, 4],
                               'Qu2': [2, 3, 1, 2, 3],
                               'Qu3': [1, 5, 2, 4, 4]})
          data
Out[234]:
             Qu1 Qu2 Qu3
In [235]: result = data.apply(pd.value counts).fillna(0)
          result
Out[235]:
             Qu1 Qu2 Qu3
           1 1.0 1.0 1.0
             0.0 2.0 1.0
             2.0 2.0 0.0
             2.0 0.0 2.0
           5 0.0 0.0 1.0
```