ANL251 Python Programming

Study Unit 6

Data Analysis with Python

Learning Outcomes and Learning Resources

1. Use appropriate Pandas functions to import datasets into DataFrame structures
   - SU6 Chapters 1.1 and 1.2
   - https://vimeo.com/59324550
   - http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html
   - http://pandas.pydata.org/pandas-docs/stable/generated/pandas.merge.html
2. Apply indexing, sorting and selection to DataFrames, and inspect missing values in DataFrames
   - SU6 Chapters 1.2, 1.3, 2 and 3
   - http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.set_index.html
   - http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.sort_values.html
3. Analyse data by grouping and aggregation
   - SU6 Chapter 4
   - http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.groupby.html

Learning Outcomes and Learning Resources

4. Prepare and examine time series data
   - SU6 Chapters 5.1 and 5.2
   - http://pandas.pydata.org/pandas-docs/stable/generated/pandas.to_datetime.html
   - http://pandas.pydata.org/pandas-docs/stable/timeseries.html#offset-aliases
   - http://pandas.pydata.org/pandas-docs/stable/timeseries.html#partial-string-indexing
5. Construct basic statistical charts.
   - SU6 Chapter 5.3
   - http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.plot.html
   - http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.rolling.html

Seminars: discussion and activities to reinforce students' understanding

1. Reading CSV File into DataFrame

Discussion

Why pandas?

NumPy arrays can only have data of the same type. In practice, you work with data of different types. The pandas package is a high-level data manipulation tool to efficiently handle this.

pandas library handles time-series data via native methods it provides to ingest, transform, and analyze time-series data effectively.

If you're looking for a functionality to perform some data transformation, chances are pandas already has it. It is actively supported by developer community and constantly increasing in functionality. (http://pandas.pydata.org/pandas-docs/stable/api.html)

Recap

Reading CSV file into DataFrame

FAO.csv

```
Area Abbreviation,Area Code,Area,Item Code,Item,Element Code,Element,Unit,latitude,longitude,Year,Production
AFG,2,Afghanistan,2511,Wheat and products,5142,Food,1000 tonnes,33.94,67.71,1961,1928.0
AFG,2,Afghanistan,2511,Wheat and products,5142,Food,1000 tonnes,33.94,67.71,1962,1904.0
AFG,2,Afghanistan,2511,Wheat and products,5142,Food,1000 tonnes,33.94,67.71,1963,1666.0
AFG,2,Afghanistan,2511,Wheat and products,5142,Food,1000 tonnes,33.94,67.71,1964,1950.0
AFG,2,Afghanistan,2511,Wheat and products,5142,Food,1000 tonnes,33.94,67.71,1965,2001.0
AFG,2,Afghanistan,2511,Wheat and products,5142,Food,1000 tonnes,33.94,67.71,1966,1808.0
```

Figure 6.2 FAO.csv opened in a text editor

Note: One of the biggest advantages of using pandas is its ability to ingest data in a variety of data types and formats. (http://pandas.pydata.org/pandas-docs/stable/io.html)

Recap

Reading CSV file into DataFrame (SU6 Chapters 1.1 and 1.2, http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html)

Note: Open the command line and execute *pip install pandas* if you haven't installed the pandas library.

```
>>> import pandas as pd
>>> df = pd.read_csv('FAO.csv')
>>> df.head()
  Area Abbreviation  Area Code         Area  Item Code                Item  \
0               AFG          2  Afghanistan       2511  Wheat and products
1               AFG          2  Afghanistan       2511  Wheat and products
2               AFG          2  Afghanistan       2511  Wheat and products
3               AFG          2  Afghanistan       2511  Wheat and products
4               AFG          2  Afghanistan       2511  Wheat and products

   Element Code Element         Unit  latitude  longitude  Year  Production
0          5142    Food  1000 tonnes     33.94      67.71  1961      1928.0
1          5142    Food  1000 tonnes     33.94      67.71  1962      1904.0
2          5142    Food  1000 tonnes     33.94      67.71  1963      1666.0
3          5142    Food  1000 tonnes     33.94      67.71  1964      1950.0
4          5142    Food  1000 tonnes     33.94      67.71  1965      2001.0
```

Figure 6.3 Getting CSV data into a DataFrame

Discussion

Merging DataFrames

- Study the exemples at http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.merge.html.
- Discuss the output if we assign other values from *{'left', 'right', 'outer', 'inner'}* to the parameter *how* ?

DataFrame.merge(right, how='inner', on=None, left_on=None, right_on=None, left_index=False, right_index=False, sort=False, suffixes=('_x', '_y'), copy=True, indicator=False, validate=None)

2. Indexing, Sorting, Selection and Missing Values

Recap

Indexing DataFrame (SU6 Chapter 1.2, http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.set_index.html)

```
>>> df.set_index('Item',inplace=True)
>>> df.head()
                    Area Abbreviation  Area Code         Area  Item Code  \
Item
Wheat and products               AFG          2  Afghanistan       2511
Wheat and products               AFG          2  Afghanistan       2511
Wheat and products               AFG          2  Afghanistan       2511
Wheat and products               AFG          2  Afghanistan       2511
Wheat and products               AFG          2  Afghanistan       2511

                     Element Code Element         Unit  latitude  longitude  \
Item
Wheat and products          5142    Food  1000 tonnes     33.94      67.71
Wheat and products          5142    Food  1000 tonnes     33.94      67.71
Wheat and products          5142    Food  1000 tonnes     33.94      67.71
Wheat and products          5142    Food  1000 tonnes     33.94      67.71
Wheat and products          5142    Food  1000 tonnes     33.94      67.71

                     Year  Production
Item
Wheat and products  1961      1928.0
Wheat and products  1962      1904.0
Wheat and products  1963      1666.0
Wheat and products  1964      1950.0
Wheat and products  1965      2001.0
```

Figure 6.4 Setting index for a DataFrame

Recap

Reseting index

```
>>> df.reset_index(inplace=True)
>>> df.head()
                 Item Area Abbreviation  Area Code         Area  Item Code  \
0  Wheat and products               AFG          2  Afghanistan       2511
1  Wheat and products               AFG          2  Afghanistan       2511
2  Wheat and products               AFG          2  Afghanistan       2511
3  Wheat and products               AFG          2  Afghanistan       2511
4  Wheat and products               AFG          2  Afghanistan       2511

   Element Code Element         Unit  latitude  longitude  Year  Production
0          5142    Food  1000 tonnes     33.94      67.71  1961      1928.0
1          5142    Food  1000 tonnes     33.94      67.71  1962      1904.0
2          5142    Food  1000 tonnes     33.94      67.71  1963      1666.0
3          5142    Food  1000 tonnes     33.94      67.71  1964      1950.0
4          5142    Food  1000 tonnes     33.94      67.71  1965      2001.0
```

Figure 6.16 Resetting the DataFrame index

Recap

Sorting DataFrame (SU6 Chapter 1.3, http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.sort_values.html)

```
>>> df.sort_values(by='Production',ascending=False).head()
           Area Abbreviation  Area Code             Area  Item Code  \
Item
Vegetables               CHN         41  China, mainland       2918
Vegetables               CHN         41  China, mainland       2918
Vegetables               CHN         41  China, mainland       2918
Vegetables               CHN         41  China, mainland       2918
Vegetables               CHN         41  China, mainland       2918

            Element Code Element         Unit  latitude  longitude  Year  \
Item
Vegetables          5142    Food  1000 tonnes     35.86      104.2  2013
Vegetables          5142    Food  1000 tonnes     35.86      104.2  2012
Vegetables          5142    Food  1000 tonnes     35.86      104.2  2011
Vegetables          5142    Food  1000 tonnes     35.86      104.2  2010
Vegetables          5142    Food  1000 tonnes     35.86      104.2  2009

            Production
Item
Vegetables    489299.0
Vegetables    479028.0
Vegetables    462696.0
Vegetables    451838.0
Vegetables    434724.0
```

Figure 6.5 Sorting the column 'Production' in a descending order

Recap

Selection (SU6 Chapter 2)

Note: While Numpy expressions for selecting and setting are intuitive, for production code, we recommend the optimized pandas data access methods, .loc and .iloc.

```
>>> df['Area']
Item
Wheat and products    Afghanistan
Wheat and products    Afghanistan
Wheat and products    Afghanistan
Wheat and products    Afghanistan
Wheat and products    Afghanistan
Wheat and products    Afghanistan
```

Figure 6.6 Selecting one column using indexing operator

Recap

Selection (SU6 Chapter 2)

```
>>> df['Raw Production'] = df['Production'] * 1000
>>> df.head()
                    Area Abbreviation  Area Code         Area  Item Code  \
Item
Wheat and products               AFG          2  Afghanistan       2511
Wheat and products               AFG          2  Afghanistan       2511
Wheat and products               AFG          2  Afghanistan       2511
Wheat and products               AFG          2  Afghanistan       2511
Wheat and products               AFG          2  Afghanistan       2511

                     Element Code Element         Unit  latitude  longitude  \
Item
Wheat and products          5142    Food  1000 tonnes     33.94      67.71
Wheat and products          5142    Food  1000 tonnes     33.94      67.71
Wheat and products          5142    Food  1000 tonnes     33.94      67.71
Wheat and products          5142    Food  1000 tonnes     33.94      67.71
Wheat and products          5142    Food  1000 tonnes     33.94      67.71

                     Year  Production  Raw Production
Item
Wheat and products  1961      1928.0       1928000.0
Wheat and products  1962      1904.0       1904000.0
Wheat and products  1963      1666.0       1666000.0
Wheat and products  1964      1950.0       1950000.0
Wheat and products  1965      2001.0       2001000.0
```

Figure 6.9 Adding a new column

Recap

Selection (SU6 Chapter 2)

Note: While Numpy expressions for selecting and setting are intuitive, for production code, we recommend the optimized pandas data access methods, .loc and .iloc.

Note: The position starts at 0.

```
>>> df.iloc[2]
Area Abbreviation            AFG
Area Code                      2
Area                 Afghanistan
Item Code                   2511
Element Code                5142
Element                     Food
Unit                 1000 tonnes
latitude                   33.94
longitude                  67.71
Year                        1963
Production                  1666
Raw Production         1.666e+06
Name: Wheat and products, dtype: object
```

Figure 6.11 Selecting a row by position

Recap

Selection (SU6 Chapter 2)

Note: Similar to the column access, the .loc attribute is also used to add new rows or update existing rows. If the row index label passed in doesn't exist, a new entry is added. Otherwise, the existing row is updated.

```
>>> df.loc['Coffee and products'].sort_values(by='Production',ascending=False).head()
                     Area Abbreviation  Area Code                      Area  \
Item
Coffee and products               USA        231  United States of America
Coffee and products               USA        231  United States of America
Coffee and products               USA        231  United States of America
Coffee and products               USA        231  United States of America
Coffee and products               USA        231  United States of America

                     Item Code  Element Code Element         Unit  latitude  \
Item
Coffee and products       2630          5142    Food  1000 tonnes     37.09
Coffee and products       2630          5142    Food  1000 tonnes     37.09
Coffee and products       2630          5142    Food  1000 tonnes     37.09
Coffee and products       2630          5142    Food  1000 tonnes     37.09
Coffee and products       2630          5142    Food  1000 tonnes     37.09

                     longitude  Year  Production  Raw Production
Item
Coffee and products     -95.71  2013      1367.0       1367000.0
Coffee and products     -95.71  1962      1356.0       1356000.0
Coffee and products     -95.71  1968      1351.0       1351000.0
Coffee and products     -95.71  1963      1340.0       1340000.0
Coffee and products     -95.71  1964      1338.0       1338000.0
```

Figure 6.10 Selecting rows by label

Recap

Selection (SU6 Chapter 2)

```
>>> df.loc['Coffee and products',['Area','Production','Year']].sort_values(by='Production',ascending=False)
                                        Area  Production  Year
Item
Coffee and products  United States of America      1367.0  2013
Coffee and products  United States of America      1356.0  1962
Coffee and products  United States of America      1351.0  1968
Coffee and products  United States of America      1340.0  1963
Coffee and products  United States of America      1338.0  1964
```

Figure 6.12 Selecting elements by specifying both column and row labels

Recap

Boolean masking (SU6 Chapter 2.4)

```
>>> df[df['Production']>400000]
                    Area Abbreviation  Area Code             Area  Item Code  \
Item
Vegetables, Other                 CHN         41  China, mainland       2605
Vegetables, Other                 CHN         41  China, mainland       2605
Vegetables, Other                 CHN         41  China, mainland       2605
Vegetables                        CHN         41  China, mainland       2918
Vegetables                        CHN         41  China, mainland       2918
Vegetables                        CHN         41  China, mainland       2918
Vegetables                        CHN         41  China, mainland       2918
```

```
Vegetables                5142    Food  1000 tonnes     35.86      104.2
Vegetables                5142    Food  1000 tonnes     35.86      104.2

                     Year  Production  Raw Production
Item
Vegetables, Other    2011    402338.0     402338000.0
Vegetables, Other    2012    419262.0     419262000.0
Vegetables, Other    2013    426850.0     426850000.0
Vegetables           2007    402975.0     402975000.0
Vegetables           2008    425537.0     425537000.0
Vegetables           2009    434724.0     434724000.0
Vegetables           2010    451838.0     451838000.0
Vegetables           2011    462696.0     462696000.0
Vegetables           2012    479028.0     479028000.0
Vegetables           2013    489299.0     489299000.0
```

Figure 6.13 Using one column for Boolean masking

Recap

Boolean masking
(SU6 Chapter 2.4)

Note: Two Boolean masks being compared with bitwise logical operator

```
>>> df[(df['Production']>400000) & (df['Year']>=2010)]
                    Area Abbreviation  Area Code             Area  Item Code  \
Item
Vegetables, Other                CHN         41  China, mainland       2605
Vegetables, Other                CHN         41  China, mainland       2605
Vegetables, Other                CHN         41  China, mainland       2605
Vegetables                       CHN         41  China, mainland       2918
Vegetables                       CHN         41  China, mainland       2918
Vegetables                       CHN         41  China, mainland       2918
Vegetables                       CHN         41  China, mainland       2918

                     Element Code Element         Unit  latitude  longitude  \
Item
Vegetables, Other          5142    Food  1000 tonnes     35.86      104.2
Vegetables, Other          5142    Food  1000 tonnes     35.86      104.2
Vegetables, Other          5142    Food  1000 tonnes     35.86      104.2
Vegetables                 5142    Food  1000 tonnes     35.86      104.2
Vegetables                 5142    Food  1000 tonnes     35.86      104.2
Vegetables                 5142    Food  1000 tonnes     35.86      104.2
Vegetables                 5142    Food  1000 tonnes     35.86      104.2

                     Year  Production  Raw Production
Item
Vegetables, Other  2011    402338.0     402338000.0
Vegetables, Other  2012    419262.0     419262000.0
Vegetables, Other  2013    426850.0     426850000.0
Vegetables         2010    451838.0     451838000.0
Vegetables         2011    462696.0     462696000.0
Vegetables         2012    479028.0     479028000.0
Vegetables         2013    489299.0     489299000.0
```

Figure 6.14 Using multiple columns for Boolean masking

Recap

Missing Values (SU6 Chapter 3, http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html)

Note: pandas primarily uses the value *np.nan* to represent missing data. When we use statistical functions on DataFrames, these functions typically ignore missing values. This is usually what we want but we should be aware that values are being excluded.

The function read_csv() provides control for missing values using two function parameters.

- the **na_values** parameter: It is to indicate other strings which could refer to missing values. By default, the following values are interpreted as NaN: ‘’, ‘#N/A’, ‘#N/A N/A’, ‘#NA’, ‘-1.#IND’, ‘-1.#QNAN’, ‘-NaN’, ‘-nan’, ‘1.#IND’, ‘1.#QNAN’, ‘N/A’, ‘NA’, ‘NULL’, ‘NaN’, ‘n/a’, ‘nan’, ‘null’.
- the **na_filter** parameter: It is to turn off white space filtering, if white space is an actual value of interest. The default is True.

Recap

Study the read_csv() function (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html).

pandas.read_csv(filepath_or_buffer, sep=', ', delimiter=None, header='infer', names=None, index_col=None, usecols=None, squeeze=False, prefix=None, mangle_dupe_cols=True, dtype=None, engine=None, converters=None, true_values=None, false_values=None, skipinitialspace=False, skiprows=None, nrows=None, na_values=None, keep_default_na=True, na_filter=True, verbose=False, skip_blank_lines=True, parse_dates=False, infer_datetime_format=False, keep_date_col=False, date_parser=None, dayfirst=False, iterator=False, chunksize=None, compression='infer', thousands=None, decimal=b'.', lineterminator=None, quotechar='"', quoting=0, escapechar=None, comment=None, encoding=None, dialect=None, tupleize_cols=None, error_bad_lines=True, warn_bad_lines=True, skipfooter=0, doublequote=True, delim_whitespace=False, low_memory=True, memory_map=False, float_precision=None)

Discussion

To get the boolean mask where values are NumPy nan.

- Study the examples at http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.isna.html.
- Write code to check whether any missing value exists in the DataFrame.

```
>>> df.head()
                    Area Abbreviation  Area Code         Area  Item Code  \
Item
Wheat and products               AFG          2  Afghanistan       2511
Wheat and products               AFG          2  Afghanistan       2511
Wheat and products               AFG          2  Afghanistan       2511
Wheat and products               AFG          2  Afghanistan       2511
Wheat and products               AFG          2  Afghanistan       2511

                     Element Code Element         Unit  latitude  longitude  \
Item
Wheat and products          5142    Food  1000 tonnes     33.94      67.71
Wheat and products          5142    Food  1000 tonnes     33.94      67.71
Wheat and products          5142    Food  1000 tonnes     33.94      67.71
Wheat and products          5142    Food  1000 tonnes     33.94      67.71
Wheat and products          5142    Food  1000 tonnes     33.94      67.71

                     Year  Production  Raw Production
Item
Wheat and products  1961      1928.0       1928000.0
Wheat and products  1962      1904.0       1904000.0
Wheat and products  1963      1666.0       1666000.0
Wheat and products  1964      1950.0       1950000.0
Wheat and products  1965      2001.0       2001000.0
```

Quiz

The variable **df** refers to the DataFrame below

	one	two
apple	100.0	111.0
ball	200.0	222.0
cerill	NaN	333.0
clock	300.0	NaN
dancy	NaN	4444.0

Write code to

- print the first column ‘one’.
- print the first two values in the first column ‘one’.
- print the first row
- print the first and third rows
- print the index
- add a new column ‘three’, with its values equal to the product of the first two columns’ values.
- add a new column ‘flag’ to check whether the first column’s value is greater than 250.
- add a new row with the values 150, 300 for the two columns

Discussion

Methods for Descriptive Stats

```
>>> df.head()
                    Area Abbreviation  Area Code         Area  Item Code  \
Item
Wheat and products               AFG          2  Afghanistan       2511
Wheat and products               AFG          2  Afghanistan       2511
Wheat and products               AFG          2  Afghanistan       2511
Wheat and products               AFG          2  Afghanistan       2511
Wheat and products               AFG          2  Afghanistan       2511

                     Element Code Element         Unit  latitude  longitude  \
Item
Wheat and products          5142    Food  1000 tonnes     33.94      67.71
Wheat and products          5142    Food  1000 tonnes     33.94      67.71
Wheat and products          5142    Food  1000 tonnes     33.94      67.71
Wheat and products          5142    Food  1000 tonnes     33.94      67.71
Wheat and products          5142    Food  1000 tonnes     33.94      67.71

                     Year  Production  Raw Production
Item
Wheat and products  1961      1928.0       1928000.0
Wheat and products  1962      1904.0       1904000.0
Wheat and products  1963      1666.0       1666000.0
Wheat and products  1964      1950.0       1950000.0
Wheat and products  1965      2001.0       2001000.0
```

- Write code to compute the descriptive stats for Column ‘Production’ using the describe() method http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.describe.html
- Understand other methods for descriptive stats http://pandas.pydata.org/pandas-docs/stable/api.html#computations-descriptive-stats

3. Grouping and Aggregation

Recap

Grouping and aggregation (SU6 Chapter 4, http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.groupby.html)

```
>>> df.groupby('Area').agg({'Production': np.average})
                                          Production
Area
Afghanistan                               156.663333
Albania                                    36.386256
Algeria                                   232.899117
Angola                                    122.211528
Antigua and Barbuda                         0.716981
Argentina                                 503.171192
Armenia                                    40.240260
Australia                                 335.213149
Austria                                   173.193566
```

Figure 6.15 Computing average production per area using the agg method

Discussion

Grouping and aggregation

- Any other ways for the calculation in Figure 6.15?
- Count the number of records for each area.

4. Time Series

Recap

Time series data(SU6 Chapters 5.1 and 5.2)

```
>>> df.set_index('Year',inplace=True)
>>> df.head()
                    Item Area Abbreviation  Area Code         Area  Item Code  \
Year
1961  Wheat and products               AFG          2  Afghanistan       2511
1962  Wheat and products               AFG          2  Afghanistan       2511
1963  Wheat and products               AFG          2  Afghanistan       2511
1964  Wheat and products               AFG          2  Afghanistan       2511
1965  Wheat and products               AFG          2  Afghanistan       2511

      Element Code Element         Unit  latitude  longitude  Production
Year
1961          5142    Food  1000 tonnes     33.94      67.71      1928.0
1962          5142    Food  1000 tonnes     33.94      67.71      1904.0
1963          5142    Food  1000 tonnes     33.94      67.71      1666.0
1964          5142    Food  1000 tonnes     33.94      67.71      1950.0
1965          5142    Food  1000 tonnes     33.94      67.71      2001.0
```

Figure 6.17 Setting the index using the Year column

Recap

Time series data(SU6 Chapters 5.1 and 5.2, http://pandas.pydata.org/pandas-docs/stable/generated/pandas.to_datetime.html, http://pandas.pydata.org/pandas-docs/stable/timeseries.html#offset-aliases)

```
>>> df.index=pd.to_datetime(df.index,format='%Y').to_period('A')
>>> df.head()
     Area Abbreviation  Area Code         Area  Item Code                Item  \
Year
1961               AFG          2  Afghanistan       2511  Wheat and products
1962               AFG          2  Afghanistan       2511  Wheat and products
1963               AFG          2  Afghanistan       2511  Wheat and products
1964               AFG          2  Afghanistan       2511  Wheat and products
1965               AFG          2  Afghanistan       2511  Wheat and products

      Element Code Element         Unit  latitude  longitude  Production
Year
1961          5142    Food  1000 tonnes     33.94      67.71      1928.0
1962          5142    Food  1000 tonnes     33.94      67.71      1904.0
1963          5142    Food  1000 tonnes     33.94      67.71      1666.0
1964          5142    Food  1000 tonnes     33.94      67.71      1950.0
1965          5142    Food  1000 tonnes     33.94      67.71      2001.0
>>> type(df.index)
<class 'pandas.core.indexes.period.PeriodIndex'>
```

Figure 6.18 Converting the index to PeriodIndex

Recap

Time series data(SU6 Chapters 5.1 and 5.2)

```
>>> import pandas as pd
>>> df = pd.read_csv('GOOGL.csv', index_col='Date', parse_dates=True)
>>> df.head()
              Open    High     Low   Close    Volume   Name
Date
2006-01-03  211.47  218.05  209.32  217.83  13137450  GOOGL
2006-01-04  222.17  224.70  220.09  222.84  15292353  GOOGL
2006-01-05  223.22  226.00  220.97  225.85  10815661  GOOGL
2006-01-06  228.66  235.49  226.85  233.06  17759521  GOOGL
2006-01-09  233.44  236.94  230.70  233.68  12795837  GOOGL
>>> type(df.index)
<class 'pandas.core.indexes.datetimes.DatetimeIndex'>
```

Figure 6.19 Importing data with the index as DatetimeIndex

Discussion

Study the read_csv() function (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html).

pandas.read_csv(filepath_or_buffer, sep=', ', delimiter=None, header='infer', names=None, index_col=None, usecols=None, squeeze=False, prefix=None, mangle_dupe_cols=True, dtype=None, engine=None, converters=None, true_values=None, false_values=None, skipinitialspace=False, skiprows=None, nrows=None, na_values=None, keep_default_na=True, na_filter=True, verbose=False, skip_blank_lines=True, parse_dates=False, infer_datetime_format=False, keep_date_col=False, date_parser=None, dayfirst=False, iterator=False, chunksize=None, compression='infer', thousands=None, decimal=b'.', lineterminator=None, quotechar='"', quoting=0, escapechar=None, comment=None, encoding=None, dialect=None, tupleize_cols=None, error_bad_lines=True, warn_bad_lines=True, skipfooter=0, doublequote=True, delim_whitespace=False, low_memory=True, memory_map=False, float_precision=None)

In Figure 6.19, if we decide to set index later, what values should be given to *parse_dates*?

df = pd.read_csv(‘GOOGL.csv’, parse_dates = ???)

Discussion

DatetimeIndex slicing

Study the examples at http://pandas.pydata.org/pandas-docs/stable/timeseries.html#partial-string-indexing

Refer to Figure 6.19. Write code to slice

- rows from 2006-01-05 to 2006-01-09
- all rows in the month 2006-01

5. Plotting with pandas

Recap

Construct statistical charts using pandas (SU6 Chapter 5.3, http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.plot.html, http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.rolling.html)

```
>>> import matplotlib.pyplot as plt
>>> df['Close 30 Moving Ave']=df['Close'].rolling(30).mean()
>>> df.head()
              Open    High     Low   Close    Volume   Name  \
Date
2006-01-03  211.47  218.05  209.32  217.83  13137450  GOOGL
2006-01-04  222.17  224.70  220.09  222.84  15292353  GOOGL
2006-01-05  223.22  226.00  220.97  225.85  10815661  GOOGL
2006-01-06  228.66  235.49  226.85  233.06  17759521  GOOGL
2006-01-09  233.44  236.94  230.70  233.68  12795837  GOOGL

            Close 30 Moving Ave
Date
2006-01-03                  NaN
2006-01-04                  NaN
2006-01-05                  NaN
2006-01-06                  NaN
2006-01-09                  NaN
>>> df[['Close','Close 30 Moving Ave']].plot();plt.show()
<matplotlib.axes._subplots.AxesSubplot object at 0x10e400e10>
```

Figure 6.20 Plotting the close price and its 30-day moving average

Note: The plot() method is a wrapper of plt.plot().

Discussion

Bar plot

Study the examples at http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.plot.bar.html

Write code to draw a bar plot for the top 10 production areas, from the output below.

```
>>> df.groupby('Area').agg({'Production': np.average})
                                          Production
Area
Afghanistan                               156.663333
Albania                                    36.386256
Algeria                                   232.899117
Angola                                    122.211528
Antigua and Barbuda                         0.716981
Argentina                                 503.171192
Armenia                                    40.240260
Australia                                 335.213149
Austria                                   173.193566
```

Discussion

Box plot

Study the examples at http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.boxplot.html

Write code to draw a box plot of wheat and products production, for the two areas France and Germany.

```
>>> df.head()
     Area Abbreviation  Area Code         Area  Item Code                Item  \
Year
1961               AFG          2  Afghanistan       2511  Wheat and products
1962               AFG          2  Afghanistan       2511  Wheat and products
1963               AFG          2  Afghanistan       2511  Wheat and products
1964               AFG          2  Afghanistan       2511  Wheat and products
1965               AFG          2  Afghanistan       2511  Wheat and products

      Element Code Element         Unit  latitude  longitude  Production
Year
1961          5142    Food  1000 tonnes     33.94      67.71      1928.0
1962          5142    Food  1000 tonnes     33.94      67.71      1904.0
1963          5142    Food  1000 tonnes     33.94      67.71      1666.0
1964          5142    Food  1000 tonnes     33.94      67.71      1950.0
1965          5142    Food  1000 tonnes     33.94      67.71      2001.0
```

Discussion

Histogram

Study the examples at http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.hist.html

Write code to draw histograms of wheat and products production, for the two areas France and Germany.

```
>>> df.head()
     Area Abbreviation  Area Code         Area  Item Code                Item  \
Year
1961               AFG          2  Afghanistan       2511  Wheat and products
1962               AFG          2  Afghanistan       2511  Wheat and products
1963               AFG          2  Afghanistan       2511  Wheat and products
1964               AFG          2  Afghanistan       2511  Wheat and products
1965               AFG          2  Afghanistan       2511  Wheat and products

      Element Code Element         Unit  latitude  longitude  Production
Year
1961          5142    Food  1000 tonnes     33.94      67.71      1928.0
1962          5142    Food  1000 tonnes     33.94      67.71      1904.0
1963          5142    Food  1000 tonnes     33.94      67.71      1666.0
1964          5142    Food  1000 tonnes     33.94      67.71      1950.0
1965          5142    Food  1000 tonnes     33.94      67.71      2001.0
```