Pandas

Pandas Data Types

Pandas dtype	Python type	Usage
object	str	Text
int64	int	Integer numbers
float64	float	Floating point numbers
bool	bool	True/False values
datetime64	NA	Date and time values
timedelta[ns]	NA	Differences between two datetimes
category	NA	Finite list of text values

Pandas Data Structure: Series

One-dimensional labeled array

In [5]:
```
pd.Series([1.0, 2.4, 3.7])
```

Out[5]:
```
0    1.0
1    2.4
2    3.7
dtype: float64
```

In [6]:
```
pd.Series({'a' : 1.0, 'b' : 2.4, 'c' : 3.7}, dtype="int64")
```

Out[6]:
```
a    1
b    2
c    3
dtype: int64
```

Pandas Data Structure: DataFrame

- Two-dimensional, like a spreadsheet or SQL table
- Rows have an *index*
- Columns have a *label* and a *data type*
- Commonly used variable name: `df`

In [9]:
```
df = pd.DataFrame({'somecat' : ['a', 'b', 'b', 'a'],
  'somedate': [pd.datetime.utcnow(), pd.datetime(2000,1,1), None, None],
  'somefloat' : [1.4, 1.23456789e-5, None, np.pi],
  'someint' : [-1, 0, 42, 1000],
  'sometext': ['foo', 'bar', 'baz', None]})
df
```

Out[9]:

	somecat	somedate	somefloat	someint	sometext
0	a	2018-08-18 06:09:04.811527	1.400000	-1	foo
1	b	2000-01-01 00:00:00.000000	0.000012	0	bar
2	b	NaT	NaN	42	baz
3	a	NaT	3.141593	1000	None

DataFrame: data types

In [15]:
```
df.dtypes
```

Out[15]:
```
somecat              object
somedate     datetime64[ns]
somefloat           float64
someint               int64
sometext             object
dtype: object
```

DataFrame: head, tail, sample

In [17]:
```
df.head(2)
```

Out[17]:

	somecat	somedate	somefloat	someint	sometext
0	a	2018-08-15 06:43:48.751782	1.400000	-1	foo
1	b	2000-01-01 00:00:00.000000	0.000012	0	bar

In [56]:
```
df.tail(2)
```

Out[56]:

	somecat	somedate	somefloat	someint	sometext
2	b	NaT	NaN	42	baz
3	a	NaT	3.141593	1000	None

In [57]:
```
df.sample(2)
```

Out[57]:

	somecat	somedate	somefloat	someint	sometext
3	a	NaT	3.141593	1000	None
0	a	2018-05-06 17:58:40.283808	1.400000	-1	foo

DataFrame: dimensions

- `shape`: number of rows and columns
- `count()`: number of values per column (excluding `None`)

In [58]:
```
df.shape
```

Out[58]:
```
(4, 5)
```

In [59]:
```
df.count()
```

Out[59]:
```
somecat      4
somedate     2
somefloat    3
someint      4
sometext     3
dtype: int64
```

Data Import and Export

Data Export

- Pandas DataFrame has many `to_*` functions for exporting data
- Writes to file if path is provided
- Many options to specify format:
  - JSON: orient, date formatting, ...
  - CSV: separator, column order, ...
  - SQL: replace, append, data type mapping, ...

In [11]:
```
df.to_json('data/demo.json', orient='columns')
df.to_json()
```

Out[11]:
```
'{"somecat":{"0":"a","1":"b","2":"b","3":"a"},"somedate":{"0":153457254481
1,"1":946684800000,"2":null,"3":null},"somefloat":{"0":1.4,"1":0.000012345
7,"2":null,"3":3.1415926536},"someint":{"0":-1,"1":0,"2":42,"3":1000},"sometex
t":{"0":"foo","1":"bar","2":"baz","3":null}}'
```

In [30]:
```
df.to_csv('data/demo.csv', index=False)
df.to_csv()
```

Out[30]:
```
',somecat,somedate,somefloat,someint,sometext\n0,a,2018-08-15 06:43:48.751782,
1.4,-1,foo\n1,b,2000-01-01 00:00:00.000000,1.23456789e-05,0,bar\n2,b,,,42,baz
\n3,a,,3.141592653589793,1000,\n'
```

Data Import

- Pandas has many `read_*` functions for importing data into a DataFrame
  - CSV, SQL (query or table), JSON, XML
- CSV:
  - Tries to automagically detect: separators, column names, data types
  - Fails sometimes, but can be defined explicitely
  - Parsing of dates needs to be defined explicitely
- SQL:
  - Requires connection to database

In [36]:
```
df = pd.read_csv('data/demo.csv', parse_dates=['somedate'])
```

Exercise 3 - Pandas overview & Basic functions

Goals:

- Be able to read data from into a Pandas DataFrame
- Get a brief overview about the data

Tasks:

- Open and inspect the Rossmann sample data with the Jupyter web interface
- Import store.csv and sales.csv data as Data Frame
  - Observe the warning when reading *sales* data
- Explore the data briefly:
  - Print the ten first/last/random stores and sales
  - What are the data types of the columns?
  - How many rows and colums are in the data sets?
  - How many null/NaN values are in the data sets?
- Write *store* data to a JSON file

Exercise 3 - Bonus Tasks

- Fix the warning when reading *sales* data
- Write to and read from SQL (sqlite)
- Why are some numeric values converted to int64 while others are converted to float64?

Explore Data

Selecting columns

If only some columns are required for further processing.

In [15]:
```
df[['somecat','someint']]
```

Out[15]:

	somecat	someint
0	a	-1
1	b	0

In [16]:
```
df[df.columns[2:4]].head(2)
```

Out[16]:

	somefloat	someint
0	1.400000	-1
1	0.000012	0

Filter Rows by Boolean Indexing

https://pandas.pydata.org/pandas-docs/stable/indexing.html#boolean-indexing (https://pandas.pydata.org/pandas-docs/stable/indexing.html#boolean-indexing)

In [64]:
```
df.loc[(df.somecat == 'b') | (df.someint > 100)]
```

Out[64]:

	somecat	somedate	somefloat	someint	sometext
1	b	2000-01-01	0.000012	0	bar
2	b	NaT	NaN	42	baz
3	a	NaT	3.141593	1000	NaN

Filter Rows by Query

https://pandas.pydata.org/pandas-docs/stable/indexing.html#the-query-method (https://pandas.pydata.org/pandas-docs/stable/indexing.html#the-query-method)

In [65]:
```
df.query('somecat == "b" or someint > 100')
```

Out[65]:

	somecat	somedate	somefloat	someint	sometext
1	b	2000-01-01	0.000012	0	bar
2	b	NaT	NaN	42	baz
3	a	NaT	3.141593	1000	NaN

N largest / smallest

- Get the rows with the **n** largest or smallest values of a column.

In [66]:
```
df.nlargest(2, 'someint')
```

Out[66]:

	somecat	somedate	somefloat	someint	sometext
3	a	NaT	3.141593	1000	NaN
2	b	NaT	NaN	42	baz

In [67]:
```
df.nsmallest(2, 'somefloat')
```

Out[67]:

	somecat	somedate	somefloat	someint	sometext
1	b	2000-01-01 00:00:00.000000	0.000012	0	bar
0	a	2018-05-06 17:58:40.283808	1.400000	-1	foo

Descriptive Statistics

Measures

- Location
  - Mean: Sum up all values and divide them by the number of values, $\bar{x} = 1/n \sum_{i=1}^{n} x_i$
  - Median: Order values and take the one from the middle
- Variability
  - Variance: $\frac{\sum (x-\bar{x})^2}{n-1}$
  - Standard Deviation: $\sqrt{variance}$
- Example: [1,2,3,10,100]
  - mean: (1+2+3+10+100)/5=23.2, median: 3
  - variance: $\frac{(1-23.2)^2+(2-23.2)^2+(3-23.2)^2+(10-23.2)^2+(100-23.2)^2}{4}$, std: 43

$$= \frac{493+449+408+174+5898}{4} = \frac{7422}{4} = 1855,5$$

Describe

- Generates various summary statistics, excluding NaN.
- By default numeric columns only
- Can be changed by defining `include`

In [68]:
```
df.describe()
```

Out[68]:

	somefloat	someint
count	3.000000	4.000000
mean	1.513868	260.250000
std	1.573883	493.573618
min	0.000012	-1.000000
25%	0.700006	-0.250000
50%	1.400000	21.000000
75%	2.270796	281.500000
max	3.141593	1000.000000

Describe

- Generates various summary statistics, excluding NaN.
- By default numeric columns only
- Can be changed by defining `include`

In [69]:
```
df.describe(include=[np.object,np.bool,np.datetime64])
```

Out[69]:

	somecat	somedate	sometext
count	4	2	3
unique	2	2	3
top	a	2018-05-06 17:58:40.283808	bar
freq	2	1	1
first	NaN	2000-01-01 00:00:00	NaN
last	NaN	2018-05-06 17:58:40.283808	NaN

Single Summaries

- Many functions to calculate summaries
- mean, min, max, median, quantiles, std, var, count, sum, ...

In [70]:
```
df.sum()
```

Out[70]:
```
somefloat       4.541605
someint      1041.000000
dtype: float64
```

Grouped Summaries

- Works also for grouped data
- mean, min, max, median, quantiles, std, var, count, sum, ...

In [71]:
```
df.groupby('somecat').sum()
```

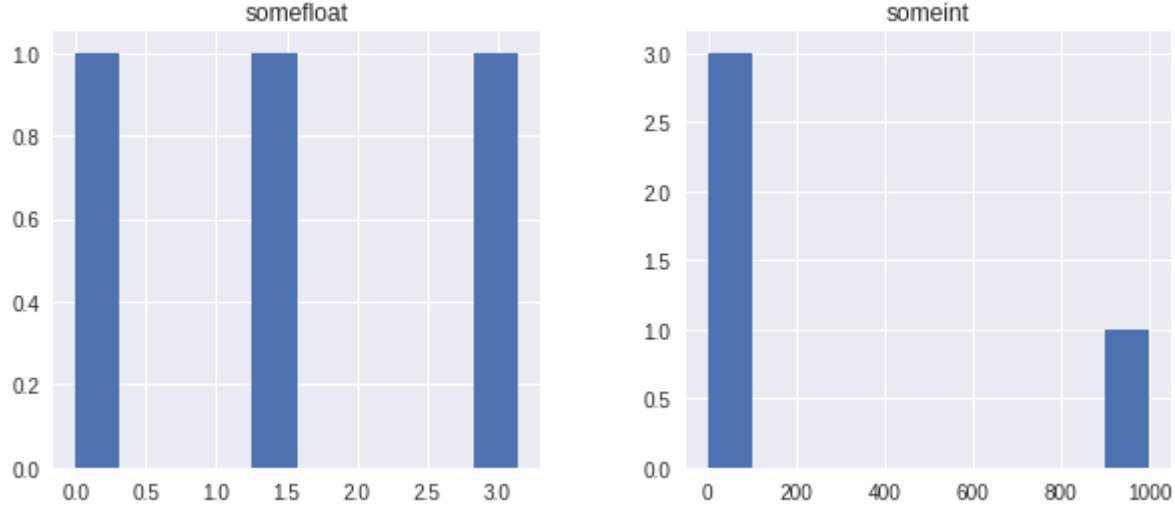
Out[71]:

	somefloat	someint
somecat		
a	4.541593	999
b	0.000012	42

Histograms

- Draw histogram of the DataFrame's series using matplotlib / pylab.
- By default 10 bins.
- Frequency of values within the dataset
- Hint: trailing semicolon supresses output

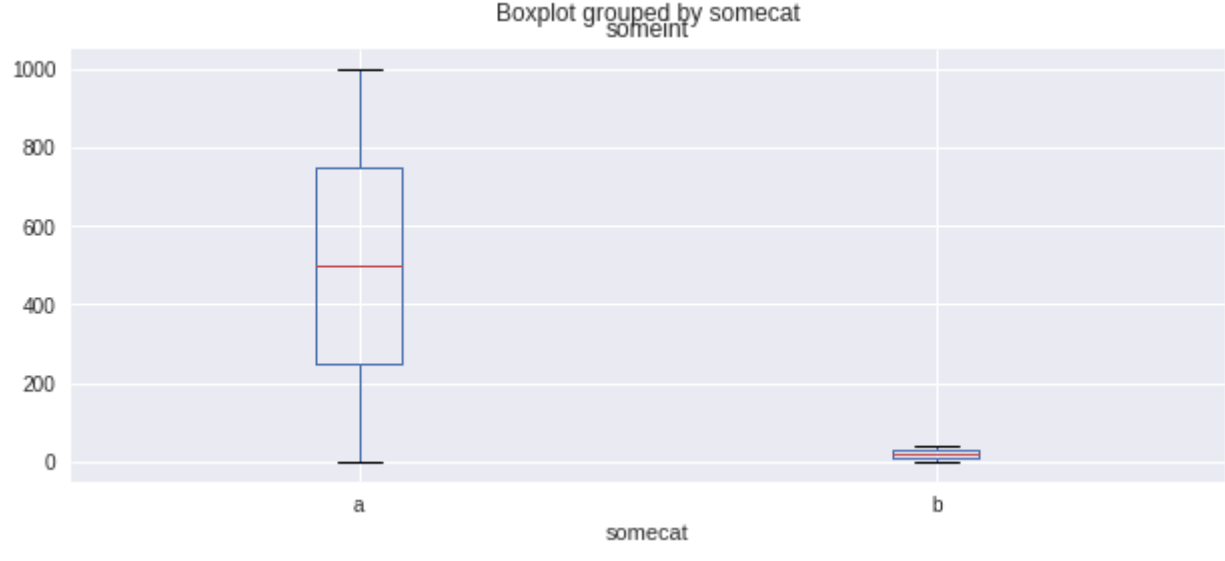
In [72]:
```
df.hist();
```



Boxplot

- Visualizes the distribution of values
- Median and first and third quantile

In [73]:
```
df.boxplot(column='someint', by='somecat');
```



Exercise 4 - Descriptive Statistics

Goals:

- Explore a data set
- Determine descritive statistics

Tasks:

- Get the number of stores per store type
- Get the number of stores per assortment
- Determine the number of stores which have a competitor within 5 km
- Calculate min, max, and mean distance to the nearest competitor
- Plot a histogram of the distance to competitors
- Plot sales and number of customers per weekday
- Boxplot with sales per weekday

Exercise 4 - Bonus Tasks

- Sales per store: min, avg, max, median
- Plot a histogram with sales per store, using 50 bins
- Play with the describe() function

Modifying a DataFrame

Modifying a DataFrame

- Selection often return a *view*
  - When modifying such a *view* a warning is printed
  - Explicit `copy()` possible
- Functions that modify data don't modify the DataFrame but return a copy
  - Many functions have an `inplace` parameter which is `False` by default

Data Cleaning

- Fill gaps:
  - Interpolate
  - Padding or backfill
  - Fill with default vlaue
  - Drop row
- Drop duplicates

In [17]:
```
df = pd.DataFrame({'somecat' : ['a', 'b', 'b', 'a'],
  'somedate': [pd.datetime.utcnow(), pd.datetime(2000,1,1), None, None],
  'somefloat' : [1.4, 1.23456789e-5, None, np.pi],
  'someint' : [-1, 0, 42, 1000], 'sometext': ['foo', 'bar', 'baz', None]})
df
```

Out[17]:

	somecat	somedate	somefloat	someint	sometext
0	a	2018-08-18 13:45:43.329279	1.400000	-1	foo
1	b	2000-01-01 00:00:00.000000	0.000012	0	bar
2	b	NaT	NaN	42	baz
3	a	NaT	3.141593	1000	None

In [18]:
```
df2 = df.copy()
df2.somefloat.interpolate(inplace=True)
df2.sometext.fillna(method='ffill', inplace=True)
df2
```

Out[18]:

	somecat	somedate	somefloat	someint	sometext
0	a	2018-08-18 13:45:43.329279	1.400000	-1	foo
1	b	2000-01-01 00:00:00.000000	0.000012	0	bar
2	b	NaT	1.570802	42	baz
3	a	NaT	3.141593	1000	baz

Data Cleaning

In [19]:
```
df2
```

Out[19]:

	somecat	somedate	somefloat	someint	sometext
0	a	2018-08-18 13:45:43.329279	1.400000	-1	foo
1	b	2000-01-01 00:00:00.000000	0.000012	0	bar
2	b	NaT	1.570802	42	baz
3	a	NaT	3.141593	1000	baz

In [20]:
```
df2.fillna(0, inplace=True)
df2.drop_duplicates('somedate', inplace=True)
df2
```

Out[20]:

	somecat	somedate	somefloat	someint	sometext
0	a	2018-08-18 13:45:43.329279	1.400000	-1	foo
1	b	2000-01-01 00:00:00	0.000012	0	bar
2	b	0	1.570802	42	baz

Add columns

In [22]:
```
df3 = df2.copy()
df3['newtext'] = 'abc'
df3['newint'] = df3['somefloat'].pow(df3.someint).astype(np.int64)
df3
```

Out[22]:

	somecat	somedate	somefloat	someint	sometext	newtext	
0	a	2018-08-18 13:45:43.329279	1.400000	-1	foo	abc	0
1	b	2000-01-01 00:00:00	0.000012	0	bar	abc	1
2	b	0	1.570802	42	baz	abc	1726

Combine DataFrames

In [27]:
```
df3
```

Out[27]:

	somecat	somedate	somefloat	someint	sometext	newtext	
0	a	2018-08-18 13:45:43.329279	1.400000	-1	foo	abc	0
1	b	2000-01-01 00:00:00	0.000012	0	bar	abc	1
2	b	0	1.570802	42	baz	abc	1726

In [26]:
```
other_df = pd.DataFrame({
  'date': [pd.datetime(2000,1,1), pd.datetime(2000,1,2), pd.datetime(2000,1,3),
    pd.datetime(2000,1,1), pd.datetime(2000,1,2), pd.datetime(2000,1,3),
    pd.datetime(2000,1,1), pd.datetime(2000,1,2), pd.datetime(2000,1,3),],
  'key' : ['foo', 'foo', 'foo', 'bar', 'bar', 'bar', 'baz', 'baz', 'baz']})
other_df
```

Out[26]:

	date	key
0	2000-01-01	foo
1	2000-01-02	foo
2	2000-01-03	foo
3	2000-01-01	bar
4	2000-01-02	bar
5	2000-01-03	bar
6	2000-01-01	baz
7	2000-01-02	baz
8	2000-01-03	baz

Combine Data Frames

In [28]:
```
merged_df = pd.merge(left=other_df, right=df3[['somecat','somefloat','someint',
'sometext']], left_on='key', right_on='sometext')
merged_df
```

Out[28]:

	date	key	somecat	somefloat	someint	sometext
0	2000-01-01	foo	a	1.400000	-1	foo
1	2000-01-02	foo	a	1.400000	-1	foo
2	2000-01-03	foo	a	1.400000	-1	foo
3	2000-01-01	bar	b	0.000012	0	bar
4	2000-01-02	bar	b	0.000012	0	bar
5	2000-01-03	bar	b	0.000012	0	bar
6	2000-01-01	baz	b	1.570802	42	baz
7	2000-01-02	baz	b	1.570802	42	baz
8	2000-01-03	baz	b	1.570802	42	baz

Exercise 5 - Modifying a DataFrame

Goals:

- Data cleaning
- Add columns
- Combine data frames

Tasks:

- Read `date` column in *sales* data as date (datetime64)
- Calculate CompetitionOpenSince date from month and year columns (datetime64)
- Fill missing CompetitionDistance with average value of all distances
- Fill missing Competition date with today's date
- Replace StateHoliday categories a, b, c with 1 and convert to int32
- Join store type, assortment, distance and open since from store df to sales df
- Plot average sales and number of customers per store type

Exercise 5 - Bonus Task

- Find correlations and plot a heat map