Python Workshop:
Handling data with Pandas

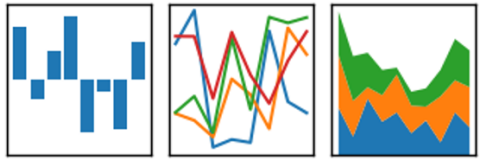
Ties de Kok

Tilburg University

Pandas Library

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



- data structures
- data analysis tools

Panda by George Lu (2011) https://www.flickr.com/photos/59328895@N00/7708872342/

Pandas Library

DataFrame

	make	price	mpg	rep78	headroom	trunk	weight	length	turn	displacement	gear_ratio	foreign
0	AMC Concord	4099	22	3.0	2.5	11	2930	186	40	121	3.58	Domestic
1	AMC Pacer	4749	17	3.0	3.0	11	3350	173	40	258	2.53	Domestic
2	AMC Spirit	3799	22	NaN	3.0	12	2640	168	35	121	3.08	Domestic
3	Buick Century	4816	20	3.0	4.5	16	3250	196	40	196	2.93	Domestic
4	Buick Electra	7827	15	4.0	4.0	20	4080	222	43	350	2.41	Domestic
5	Buick LeSabre	5788	18	3.0	4.0	21	3670	218	43	231	2.73	Domestic
6	Buick Opel	4453	26	NaN	3.0	10	2230	170	34	304	2.87	Domestic

Pandas Library

DataFrame

	make	price	mpg	rep78	headroom	trunk	weight	length	turn	displacement	gear_ratio	foreign
0	AMC Concord	4099	22	3.0	2.5	11	2930	186	40	121	3.58	Domestic
1	AMC Pacer	4749	17	3.0	3.0	11	3350	173	40	258	2.53	Domestic
2	AMC Spirit	3799	22	NaN	3.0	12	2640	168	35	121	3.08	Domestic
3	Buick Century	4816	20	3.0	4.5	16	3250	196	40	196	2.93	Domestic
4	Buick Electra	7827	15	4.0	4.0	20	4080	222	43	350	2.41	Domestic
5	Buick LeSabre	5788	18	3.0	4.0	21	3670	218	43	231	2.73	Domestic
6	Buick Opel	4453	26	NaN	3.0	10	2230	170	34	304	2.87	Domestic

```
df_auto[(df_auto['price'] < 3800) & (df_auto['foreign'] == 'Foreign')]
```

	make	price	mpg	rep78	headroom	trunk	weight	length	turn	displacement	gear_ratio	foreign
65	Subaru	3798	35	5.0	2.5	11	2050	164	36	97	3.81	Foreign
67	Toyota Corolla	3748	31	5.0	3.0	9	2200	165	35	97	3.21	Foreign

Pandas Library

Agenda

What are we going to do this session

1. Terminology
2. Specific topics:
    - Open files
    - Saving files
    - Navigating dataframe
    - Select data
    - Create new columns
    - Merge data
    - Groupby operation
    - Plotting with Pandas
    - Plotting with Seaborn

Terminology

Pandas vs. Numpy

Numpy provides a powerful N-dimensional array object.

- Pandas builds upon the Numpy functionality.

Terminology

Pandas vs. Numpy

Numpy provides a powerful N-dimensional array object.

- Pandas builds upon the Numpy functionality.

pd.DataFrame vs. pd.Series

A Pandas Series is a 1D data structure (like a vector)
A Pandas DataFrame is a 2D data structure (like a matrix)

- Columns and rows in a DataFrame are Series.

Open data

Opening data

Pandas can open pretty much any data file!

Open Excel file

```
excel_file = pd.read_excel(join(data_path, 'excel_sample.xlsx'))
```

Open CSV file

```
csv_file = pd.read_csv(join(data_path, 'csv_sample.csv'), sep=',')
```

Open Stata file

```
stata_file = pd.read_stata(join(data_path, 'stata_sample.dta'))
```

▶ Opening and Saving files with Pandas notebook

Saving data

Pandas can save to pretty much any data file! (except SAS)

Save Excel file

```
excel_file.to_excel(join(data_path, 'excel_sample.xlsx'))
```

Save CSV file

```
csv_file.to_csv(join(data_path, 'csv_sample.csv'), sep=',')
```

Save Stata file

```
stata_file.to_stata(join(data_path, 'stata_sample.dta'))
```

▶ Opening and Saving files with Pandas notebook

HDF files

- Tip: HDF files are awesome!

HDF vs CSV

```
%timeit test_hdf_table_write(df)
1 loops, best of 3: 901 ms per loop

%timeit test_csv_write(df)
1 loops, best of 3: 3.44 s per loop
```

HDF is a lot faster!

Read HDF files using Pandas

```
hdf_df = pd.read_hdf(join(data_path, 'hdf_sample.h5'), 'hdf_sample')
```

Write HDF files using Pandas

```
hdf_df.to_hdf(join(data_path, 'hdf_sample.h5'), 'hdf_sample')
```

Open data

Save data

Navigate

How to inspect your data?

- There is no standard data browser for DataFrames

My recommendation

Use basic operations to view parts of the data in the notebook:

```
df_auto.head(3)
```

	make	price	mpg	rep78	headroom	trunk	weight	length	turn	displacement	gear_ratio	foreign
0	AMC Concord	4099	22	3.0	2.5	11	2930	186	40	121	3.58	Domestic
1	AMC Pacer	4749	17	3.0	3.0	11	3350	173	40	258	2.53	Domestic
2	AMC Spirit	3799	22	NaN	3.0	12	2640	168	35	121	3.08	Domestic

```
df_auto.tail(3)
```

	make	price	mpg	rep78	headroom	trunk	weight	length	turn	displacement	gear_ratio	foreign
71	VW Rabbit	4697	25	4.0	3.0	15	1930	155	35	89	3.78	Foreign
72	VW Scirocco	6850	25	4.0	2.0	16	1990	156	36	97	3.78	Foreign
73	Volvo 260	11995	17	5.0	2.5	14	3170	193	37	163	2.98	Foreign

Open data

Save data

Navigate

Alternative, use the QGrid extension

```
In [2]: import qgrid
qgrid_widget = qgrid.show_grid(df, show_toolbar=True)
qgrid_widget
```

Add Row | Remove Row

index	A	B	C	D	E
0	2013-01-01	-2.0773	washington	foo	✔
1	2013-01-02	-0.55236	adams	bar	
2	2013-01-03	-1.46247	washington	buzz	
3	2013-01-04	0.85559	madison	bippity	
4	2013-01-05	-1.66466	lincoln	boppity	
5	2013-01-06	0.85659	jefferson	foo	✔
6	2013-01-07	1.23665	hamilton	foo	✔
7	2013-01-08	-0.36515	roosevelt	bar	
8	2013-01-09	0.89955	kennedy	zoo	

```
In [3]: qgrid_widget.get_changed_df()
```

Out[3]:

	A	B	C	D	E
0	2013-01-01	-2.077295	washington	foo	True
1	2013-01-02	-0.552359	adams	bar	False
2	2013-01-03	-1.462471	washington	buzz	False
3	2013-01-04	0.855593	madison	bippity	False
4	2013-01-05	-1.664660	lincoln	boppity	False
5	2013-01-06	0.856594	jefferson	foo	True
6	2013-01-07	1.236655	hamilton	foo	True
7	2013-01-08	-0.365152	roosevelt	bar	False
8	2013-01-09	0.899548	kennedy	zoo	False

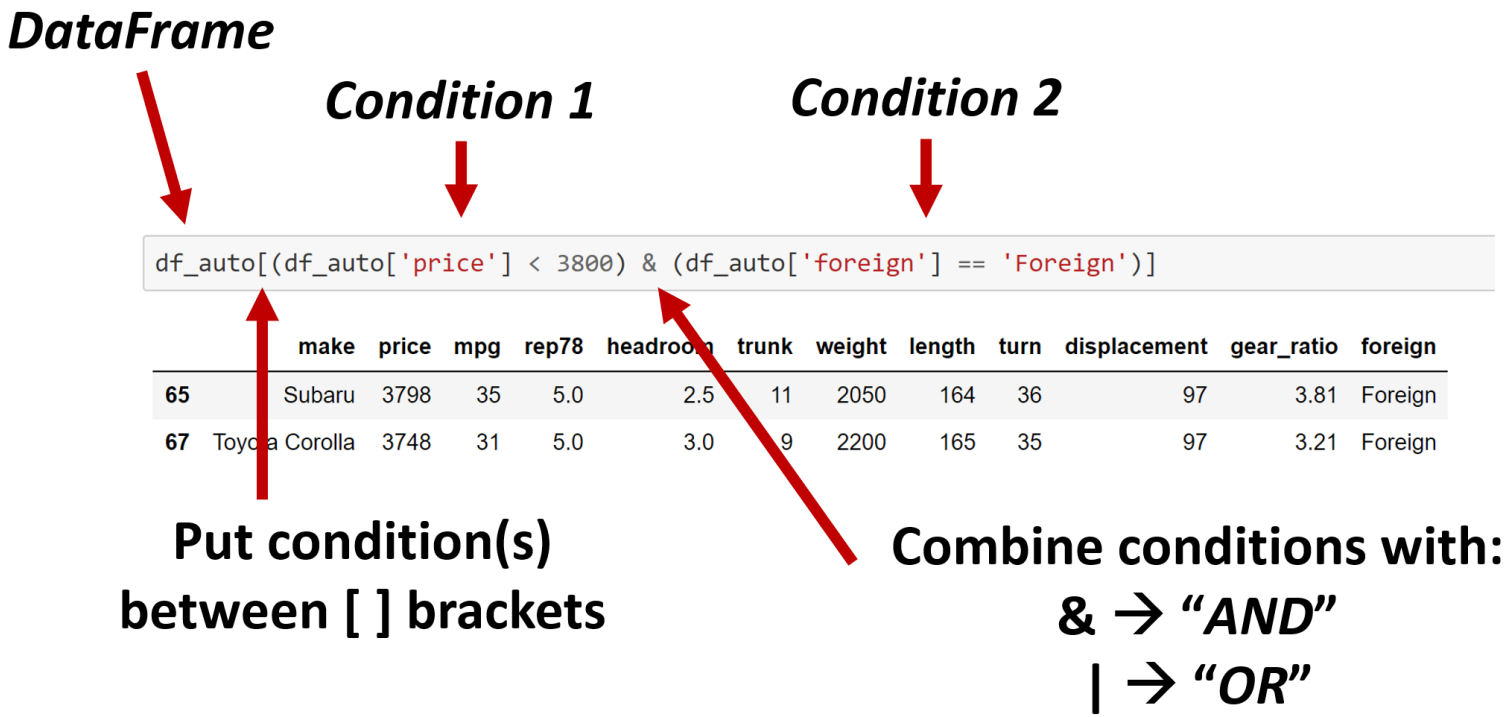
Open data

Save data

Navigate

Select data

Selecting data



- Selecting data based on a condition, Jupyter Notebook

Open data

Save data

Navigate

Select data

Create Columns

Creating columns

```
df_auto['price_trunk_ratio'] = df_auto.price / df_auto.trunk
df_auto[['price', 'trunk', 'price_trunk_ratio']].head()
```

Simple operations

	price	trunk	price_trunk_ratio
55	6229	6	1038.166667
47	4934	7	704.857143
44	6486	8	810.750000
23	4389	9	487.666667
17	3667	7	523.857143

Advanced operations

```
def new_price_function(x):
    if x.foreign == 'Foreign':
        return x.price * 1.5
    else:
        return x.price
```

```
df_auto['new_price'] = df_auto.apply(new_price_function, axis=1)
df_auto[['make', 'price', 'foreign', 'new_price']].head()
```

	make	price	foreign	new_price
55	Datsun 200	6229	Foreign	9343.5
47	Pont. Firebird	4934	Domestic	4934.0
44	Plym. Sapporo	6486	Domestic	6486.0
23	Ford Fiesta	4389	Domestic	4389.0
17	Chev. Monza	3667	Domestic	3667.0

- Various methods to create columns, Jupyter Notebook

Merging DataFrames

```
df_auto_p1.head(3)
```

	make	price	mpg
55	Datsun 200	6229	23
47	Pont. Firebird	4934	18
44	Plym. Sapporo	6486	26

+

```
df_auto_p2.head(3)
```

	make	headroom	trunk
55	Datsun 200	1.5	6
47	Pont. Firebird	1.5	7
44	Plym. Sapporo	1.5	8

=

```
merged_auto = pd.merge(df_auto_p1, df_auto_p2, how='left', on='make')
merged_auto.head(3)
```

	make	price	mpg	headroom	trunk
0	Datsun 200	6229	23	1.5	6
1	Pont. Firebird	4934	18	1.5	7
2	Plym. Sapporo	6486	26	1.5	8

Identifier

- Various methods to merge, join, and append, Jupyter Notebook

GroupBy Operations

```
grouped = df_auto.groupby(['foreign'])
```

```
grouped.mean()
```

foreign	price	mpg	headroom	trunk	weight	length
Domestic	6072.423077	19.826923	3.153846	14.750000	3317.115385	196.134615
Foreign	6384.681818	24.772727	2.613636	11.409091	2315.909091	168.545455

```
grouped.first()
```

foreign	price	mpg	headroom	trunk	weight	length
Domestic	4934	18	1.5	7	3470	198
Foreign	6229	23	1.5	6	2370	170

→ ***And a lot more!***

- Various methods to merge, join, and append, Jupyter Notebook

Save data

Navigate

Select data

Create Columns

Merge data

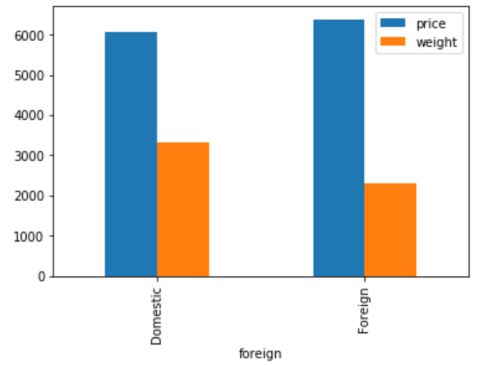
Groupby Operation

Plotting

Plotting data (Pandas and Seaborn)

```
df_auto.groupby(['foreign']).mean()[['price', 'weight']].plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x16a5fa5e668>
```



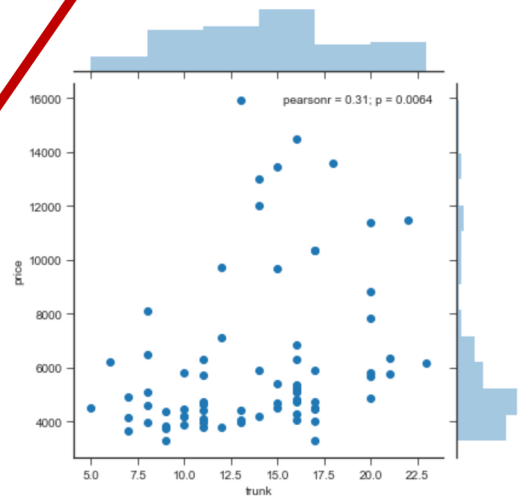
Manual transformation

Automatic transformation

```
import seaborn as sns
```

```
sns.jointplot(x="trunk", y="price", data=df_auto)
```

```
<seaborn.axisgrid.JointGrid at 0x16a61e77a90>
```



- Comprehensive notebook for plotting with Pandas

Closing remarks

Questions?

Mini-Task Instructions

Mini Task

Goal: Get hands-on experience with Pandas on a real-life dataset.

Instructions

1. Open (start) a Jupyter Notebook in the `Materials` folder
2. Solve the mini-tasks in: `day_1 > mini_task > pandas_mini_task.ipynb`

Feel free to also work on: `day_1 > mini_task > basic_python_tasks.ipynb`

For help:

- Python tutorial
- Python Basics Notebook
- Opening files with Python / Pandas
- Data handling with Pandas