Fourth Industrial Summer School

Day 3: Data Analysis Foundations

Time Series

Session Objectives

- ✓ Time Series Basics
- ✓ Resampling
- ✓ Examples
  - Air Passengers
  - Germany power consumption
- ✓ Feature Engineering using tsfresh

Time series

- Set of data points indexed in time order
  - Common to have equally spaced points

- Example
  - Day temperature
  - Wind speed
  - Stock market
  - Think of examples in your field?

- Focus
  - Trend (increase /decrease)
  - Seasonal patterns

Time series

- The **objective** of time series analysis is to uncover a pattern in the time series and then extrapolate the pattern to forecast the future.

- The measurements may be taken every
  - Hour/Day/Week/Month/Year
  - or at any other regular interval

- Sometimes, dates are given or expected as strings, so a **conversion** from strings to dates is necessary.
  - https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html

Example – Exploring time series data

- Consider the provided dataset (Passengers.csv)
  - Holds passengers data captured over a number of years
  - Time series data

- Objective
  - Time series analysis

Time series - Passengers example

- Passengers.csv
  - Contains the volume of international passengers over a period of years.

```
import pandas as pd

data = pd.read_csv('Passengers.csv')
data.head()
```

	Month	Passengers
0	1949-01-01	112
1	1949-02-01	118
2	1949-03-01	132
3	1949-04-01	129
4	1949-05-01	121

Time series - Passengers example

- Check data types

```
1   data.dtypes
```

```
Month          object
Passengers      int64
dtype: object
```

- Think about appropriate data type for a time series.

Time series - Passengers example

- Convert Month column from String to Date

```
data['Month'] = pd.to_datetime(data['Month'])
data.dtypes
```

```
Month         datetime64[ns]
Passengers             int64
dtype: object
```

```
data['Month'].dt.month.head()
```

```
0    1
1    2
2    3
3    4
4    5
Name: Month, dtype: int64
```

Time series - Passengers example

- Make datetime an index to the pandas dataframe

```
data = data.set_index('Month')
data.head()
```

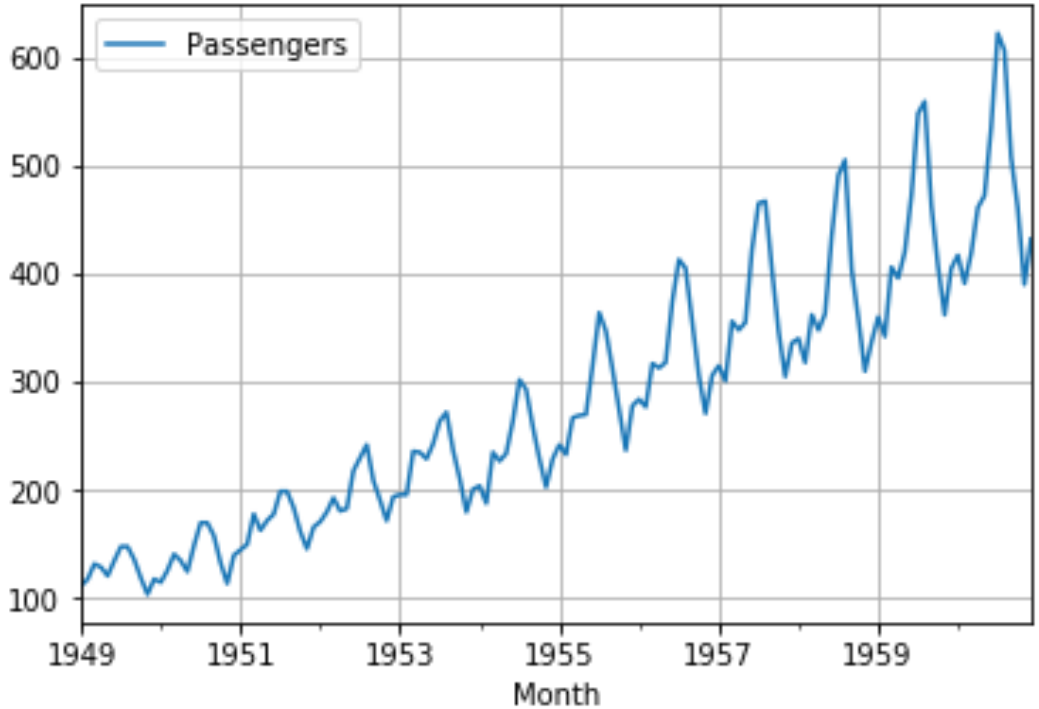
	Passengers
Month	
1949-01-01	112
1949-02-01	118
1949-03-01	132
1949-04-01	129
1949-05-01	121

Time series - Passengers example

- Plot the time series

```
data.plot(grid='on')
```

- Observations?



Time series - Passengers example

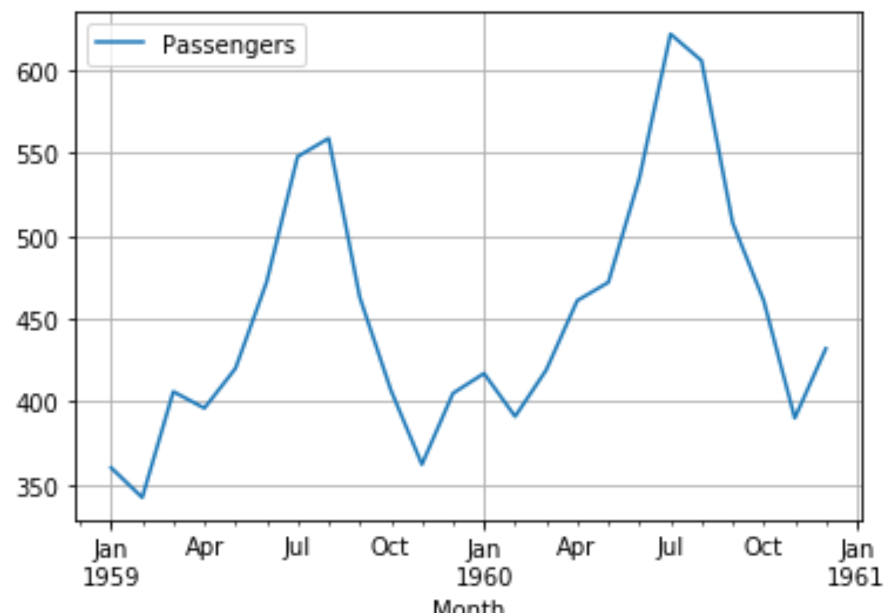
- Narrow window to a specified time frame

```python
from datetime import datetime

start = datetime(1959,1,1)
end = datetime(1960,12,30)

shortData = data[ (start <= data.index) & ( data.index <= end)]
# ----------- Option2. shortData = data['1/1/1959':'12/30/1960']
shortData.plot(grid='on')
```

- Observations?



Time series - Passengers example

- Descriptive statistics
  - Generate descriptive statistics that summarize the central tendency, dispersion and shape of a dataset’s distribution, excluding null values.

```
1   data.describe()
```

	Passengers
count	144.000000
mean	280.298611
std	119.966317
min	104.000000
25%	180.000000
50%	265.500000
75%	360.500000
max	622.000000

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.describe.html

Time series - Passengers example

- Resample

```
data.resample('Y').sum()
```

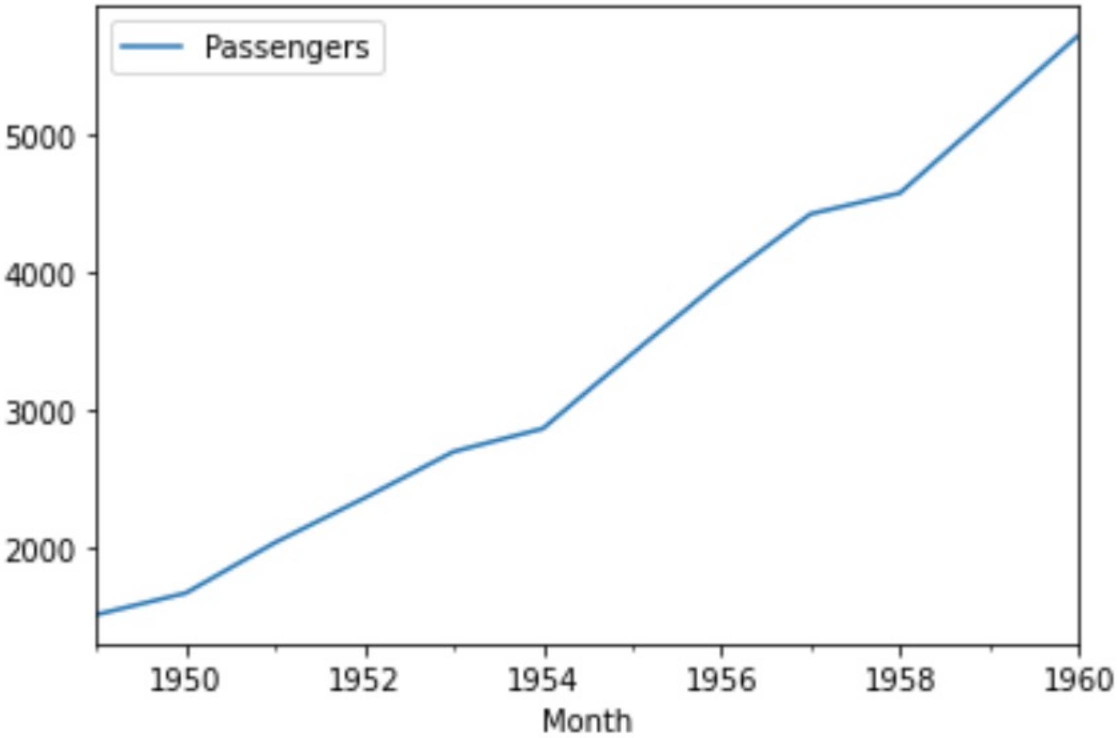
	Passengers
Month	
1949-12-31	1520
1950-12-31	1676
1951-12-31	2042
1952-12-31	2364
1953-12-31	2700
1954-12-31	2867
1955-12-31	3408
1956-12-31	3939
1957-12-31	4421
1958-12-31	4572
1959-12-31	5140
1960-12-31	5714

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.resample.html

Time series - Passengers example

- Resample

```
data.resample('Y').sum().plot()
```



<u>Example</u>

Germany Open Power Systems Data

Time series – Germany power example

- Germany.csv
  - Electricity production and consumption are reported as daily totals in gigawatt-hours (GWh).

- The columns of the data file are:
  - Date
    - The date (yyyy-mm-dd format)
  - Consumption
    - Electricity consumption in GWh
  - Wind
    - Wind power production in GWh
  - Solar
    - Solar power production in GWh
  - Wind+Solar
    - Sum of wind and solar power production in GWh

Time series – Germany power example

- Read data. New way of reading data:
  - Parsing dates
  - Setting first column (Time) as index

```
df = pd.read_csv('Germany.csv',index_col=0, parse_dates=True)
df.dtypes
```

```
Consumption    float64
Wind           float64
Solar          float64
Wind+Solar     float64
dtype: object
```

Time series – Germany power example

- Make sure time was set as an index

```
1   df.index
```

```
DatetimeIndex(['2006-01-01', '2006-01-02', '2006-01-03', '2006-01-04',
               '2006-01-05', '2006-01-06', '2006-01-07', '2006-01-08',
               '2006-01-09', '2006-01-10',
               ...
               '2017-12-22', '2017-12-23', '2017-12-24', '2017-12-25',
               '2017-12-26', '2017-12-27', '2017-12-28', '2017-12-29',
               '2017-12-30', '2017-12-31'],
              dtype='datetime64[ns]', name='Date', length=4383, freq=None)
```

Time series – Germany power example

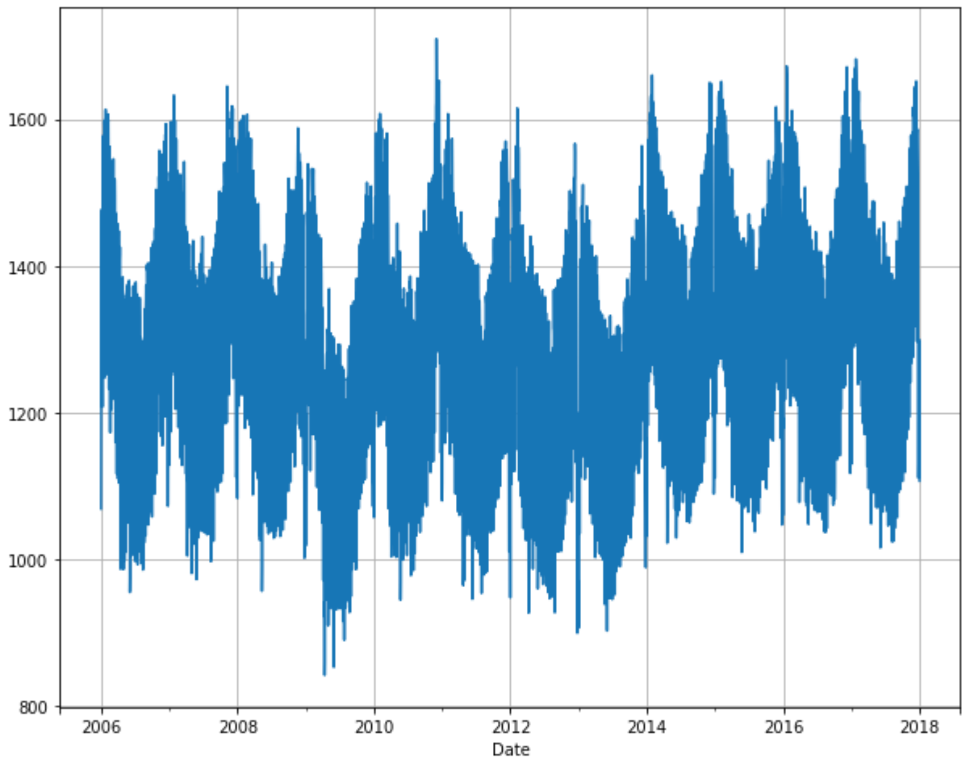
- Beauty of having datetime object
  - Ability to have individual date/time components as attributes
  - Created new columns
    - Year
    - Month
    - Day of the week

```
df['Year'] = df.index.year
df['Month'] = df.index.month
df['Weekday Name'] = df.index.day_name()
```

Time series – Germany power example

- Line plot of Germany daily electricity consumption

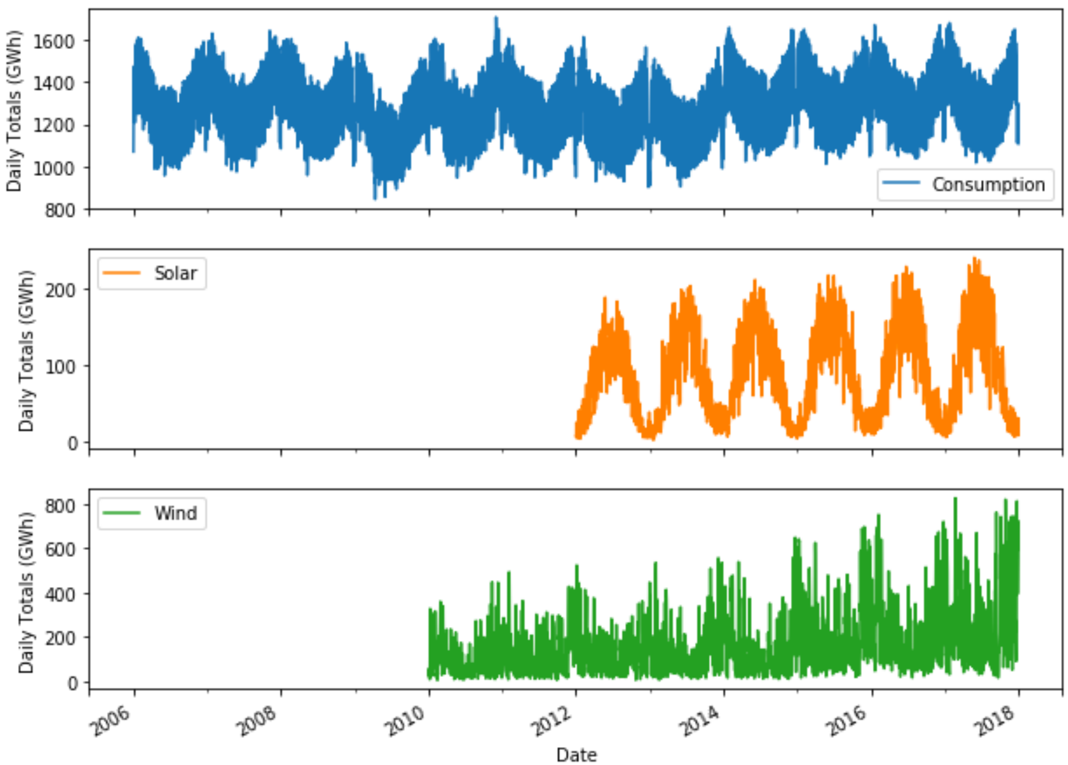
```
1  df['Consumption'].plot(grid='on')
```



Time series – Germany power example

- Plot all columns information
  - Electricity Consumption
  - Wind production
  - Solar production

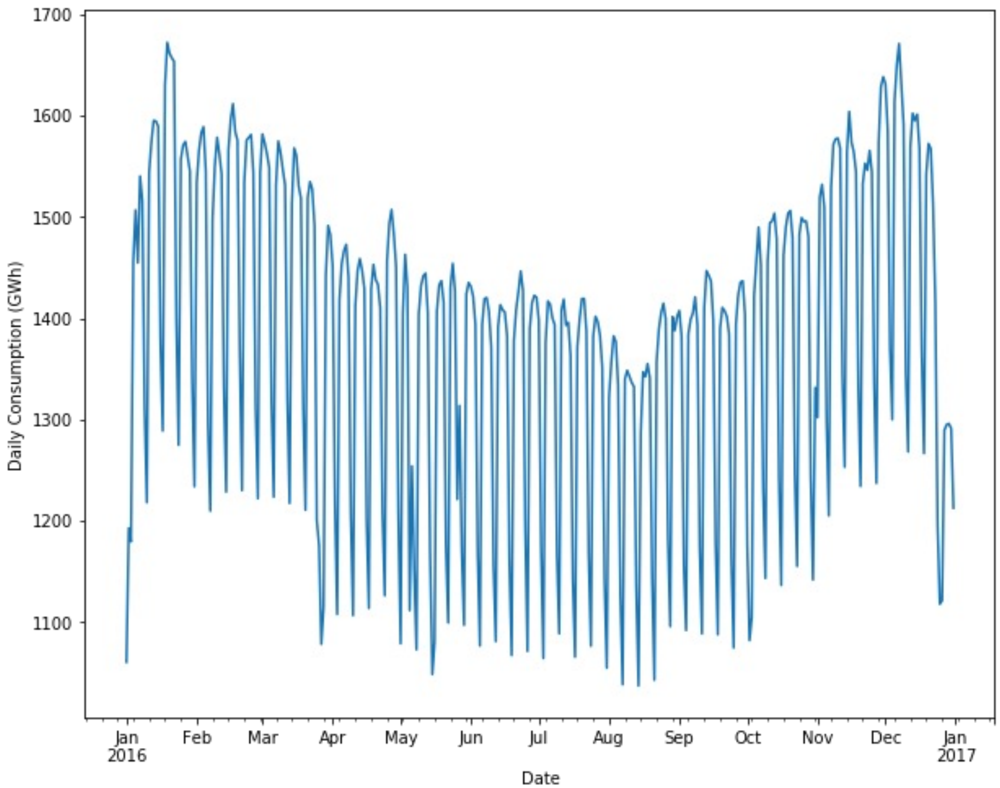
```
cols_plot = ['Consumption', 'Solar', 'Wind']
axes = df[cols_plot].plot(subplots=True,figsize=(10,10))
for ax in axes:
    ax.set_ylabel('Daily Totals (GWh)')
```



Time series – Germany power example

- Study consumption for a specific year (e.g. 2016)

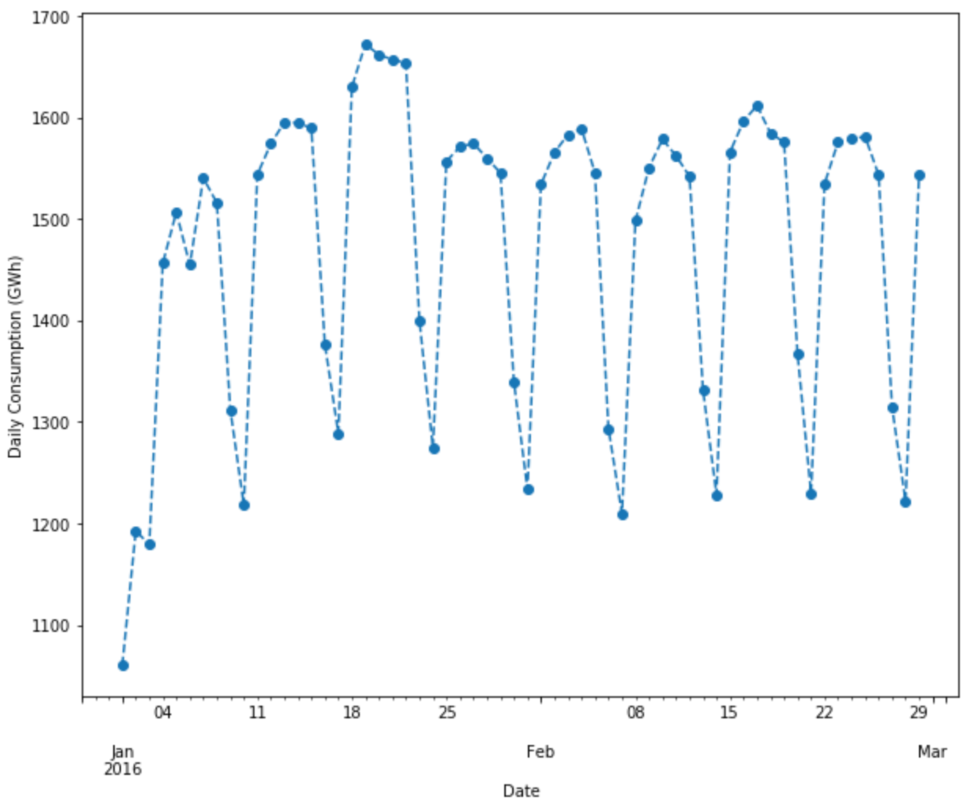
```
ax = df.loc['2016', 'Consumption'].plot()
ax.set_ylabel('Daily Consumption (GWh)')
```



Time series – Germany power example

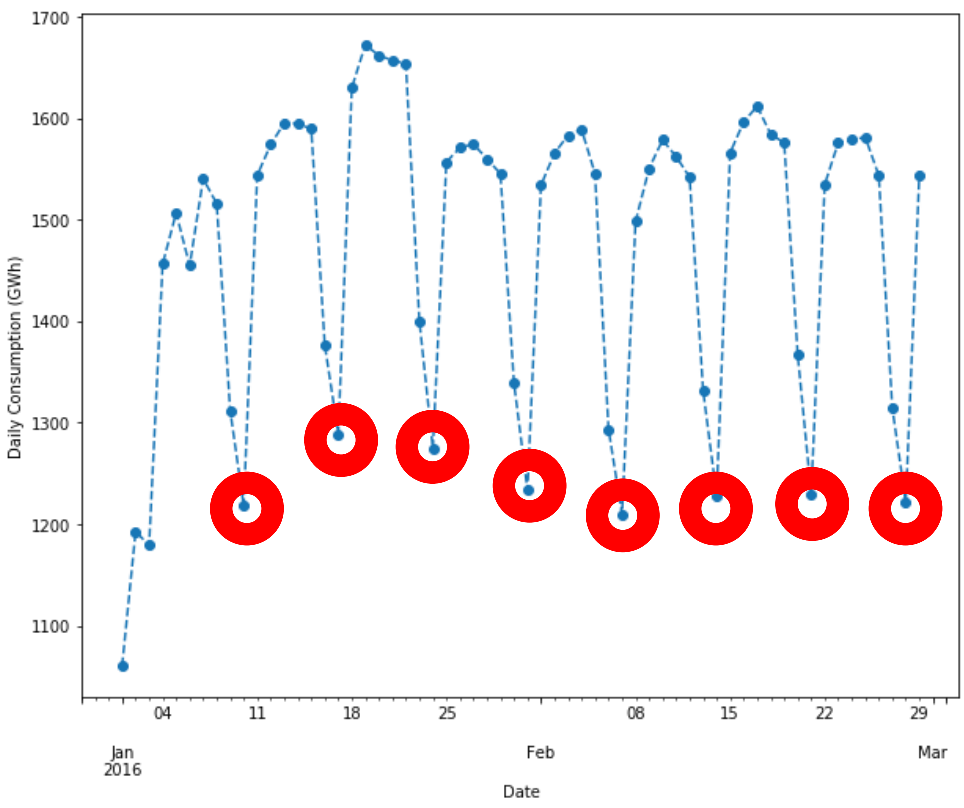
- Narrow the window to 2 month in 2016

```
1   ax = df.loc['2016-01':'2016-02', 'Consumption'].plot(marker='o',linestyle='--')
2   ax.set_ylabel('Daily Consumption (GWh)')
```



Time series – Germany power example

- Investigate why we have those low points?



Time series – Germany power example

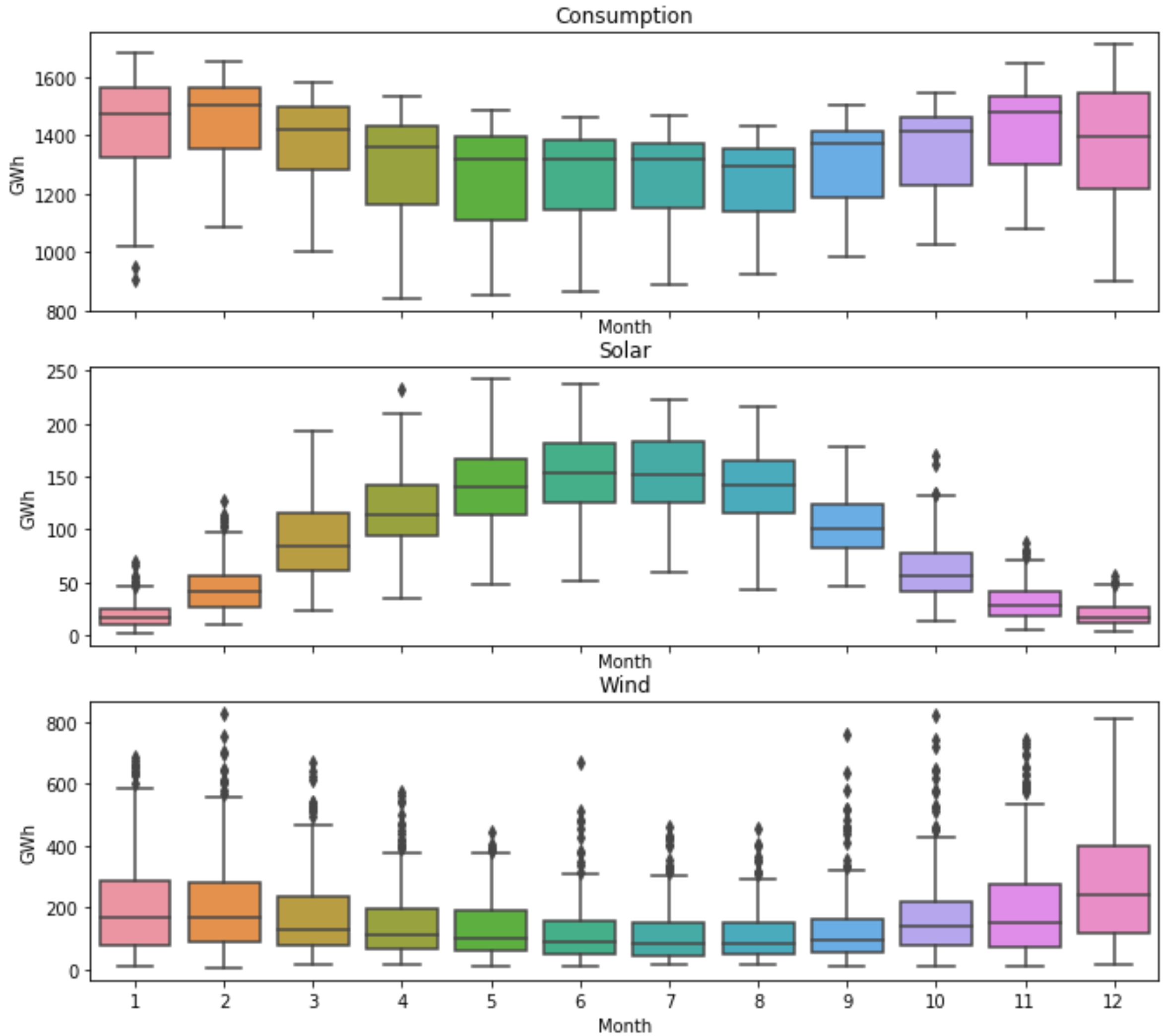
- Seasonality
  - Using boxplots group the data by months and display the distributions for each group
    - To visualize yearly seasonality

```
fig, axes = plt.subplots(3, 1, figsize=(10,10), sharex=True)
for name, ax in zip(['Consumption', 'Solar', 'Wind'], axes):
    sns.boxplot(data=df, x='Month', y=name, ax=ax)
    ax.set_ylabel('GWh')
    ax.set_title(name)
```

https://docs.python.org/3.3/library/functions.html#zip

Time series – Germany power example

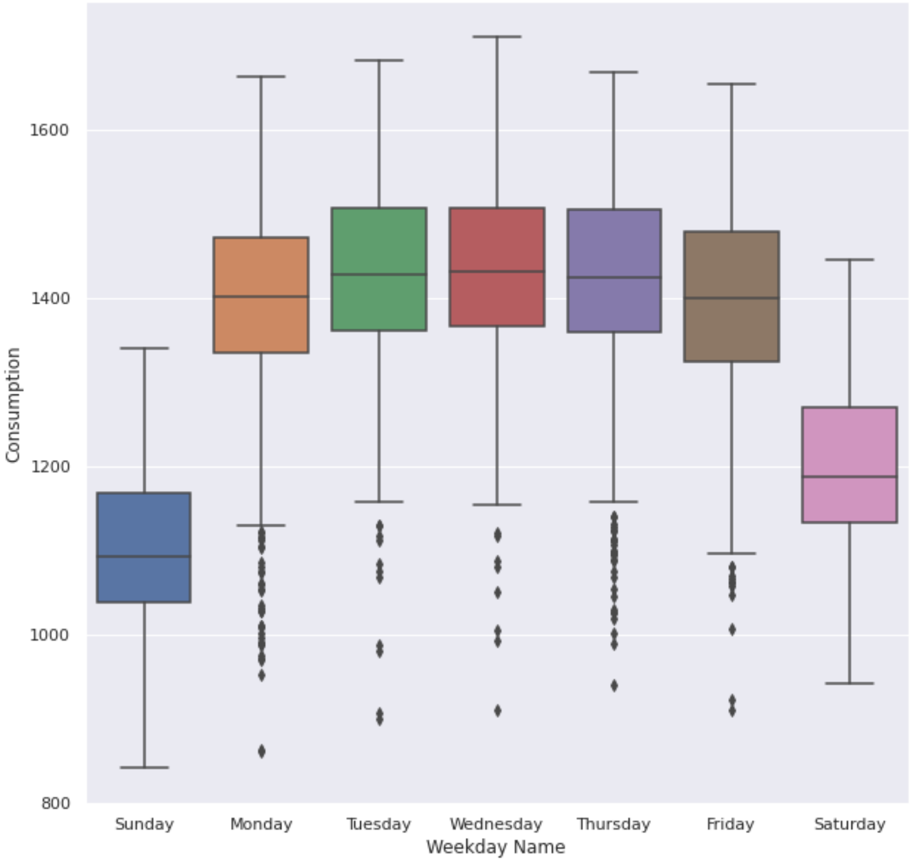
- Observations?



Time series – Germany power example

- Seasonality
  - Group the electricity consumption time series by day of the week
    - To visualize weekly seasonality

```
plt.figure(figsize=(10, 10))
sns.boxplot(data=df, x='Weekday Name', y='Consumption')
```

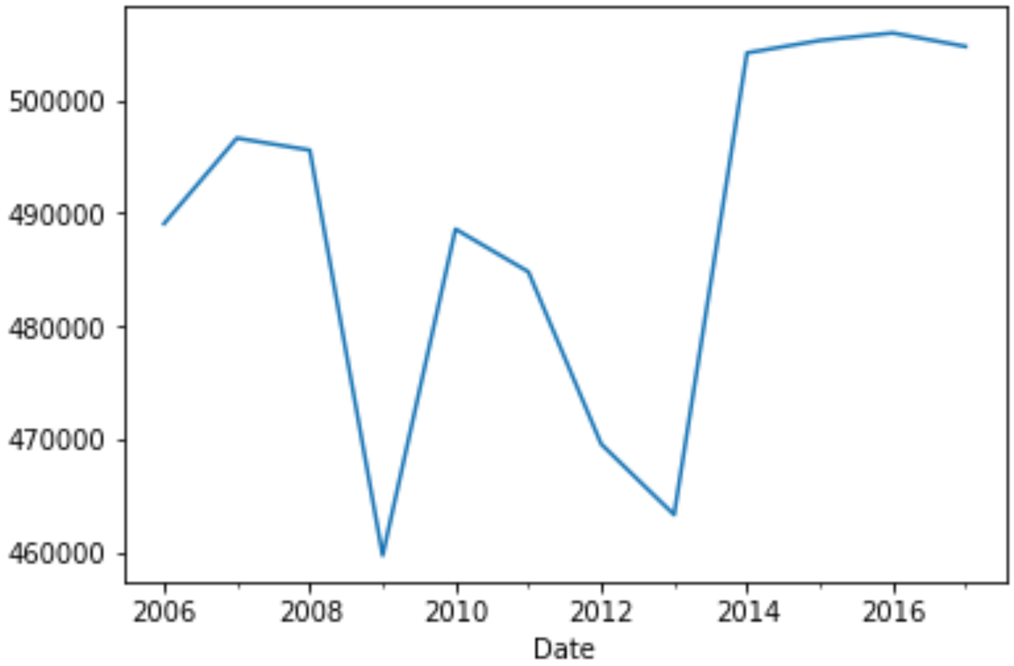


- Observations?

Time series – Germany power example

- Resample (year)

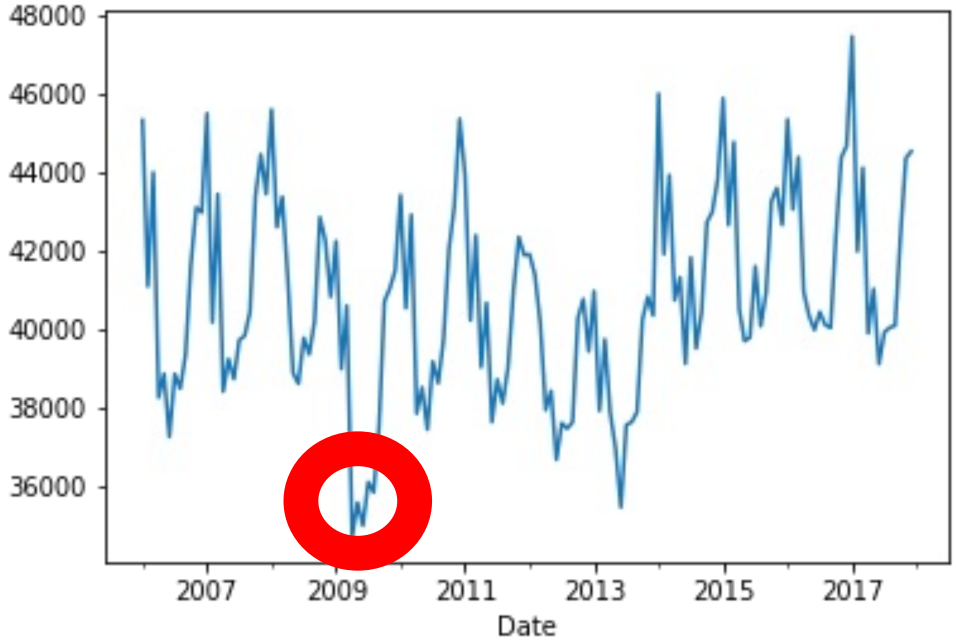
```
1   df['Consumption'].resample('Y').sum().plot()
```



Time series – Germany power example

- Resample (Month)

```
1    df['Consumption'].resample('M').sum().plot()
```



- Investigate this low point.

<u>Break</u>

15 Minutes

Part

tsfresh

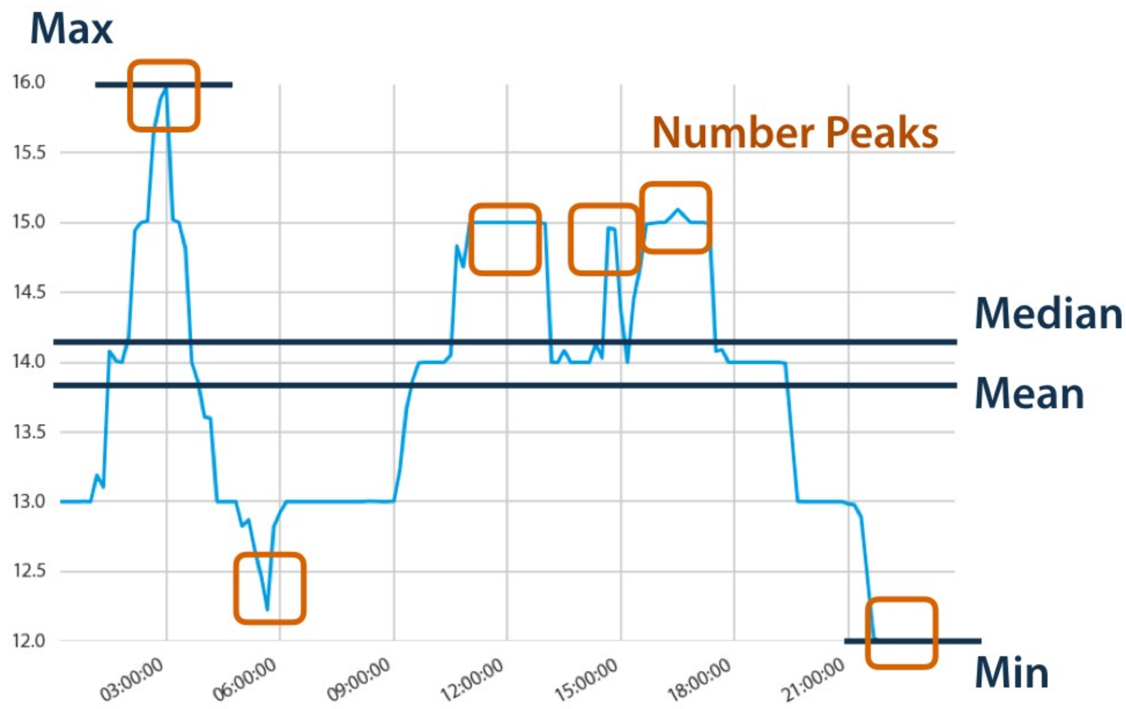
tsfresh

- Time Series Feature extraction based on scalable hypothesis tests
  - https://www.sciencedirect.com/science/article/pii/S0925231218304843

- Python package for time series analysis that contains
  - feature extraction methods
  - feature selection algorithm

tsfresh

- Automatically extracts 100s features from time series data that describe both basic and complex characteristics of a time series
  - number of peaks
  - average value
  - maximum value
  - etc …



- Those features can be used to build regression or classification based machine learning models.

tsfresh

- Objective: Time series often contain noise, redundancies or irrelevant information.
  - As a result most of the extracted features will not be useful for the machine learning task

- Solution
  - To avoid extracting irrelevant features, the TSFRESH package has a built-in filtering procedure.
  - It is based on the well developed theory of hypothesis testing and uses a multiple test procedure.

tsfresh - example

- **Robot Execution Failures** dataset
  - Contains time series data recorded by 6 sensors from 88 robots.
    - Each sensor will produce a time series
    - Each feature is numeric, representing a force or a torque measured after failure detection; each failure instance is characterized in terms of 15 force/torque samples collected at regular time intervals starting immediately after failure detection; The total observation window for each failure instance was of 315 ms.
  - For each sample denoted by a different id we are going to classify if the robot reports a failure or not.

- Failure due to
  - Failures in approach to grasp position.
  - Failures in transfer of a part.
  - Position of part after a transfer failure.
  - Failures in approach to ungrasp position.
  - Failures in motion with part

tsfresh - example

- Install tsfresh in your colab environment

```
!pip install tsfresh
```

- Don’t forget to **restart your runtime**

- Import the dataset from tsfresh
- Load dataset

```
from tsfresh.examples.robot_execution_failures import download_robot_execution_failures, \
                                                      load_robot_execution_failures
download_robot_execution_failures()
timeseries, y = load_robot_execution_failures()
```

tsfresh - example

- Check the dataframe

```
1   timeseries.head()
```

	id	time	F_x	F_y	F_z	T_x	T_y	T_z
0	1	0	-1	-1	63	-3	-1	0
1	1	1	0	0	62	-3	-1	0
2	1	2	-1	-1	61	-3	0	0
3	1	3	-1	-1	63	-2	-1	0
4	1	4	-1	-1	63	-3	-1	0

```
1   y.head()
```

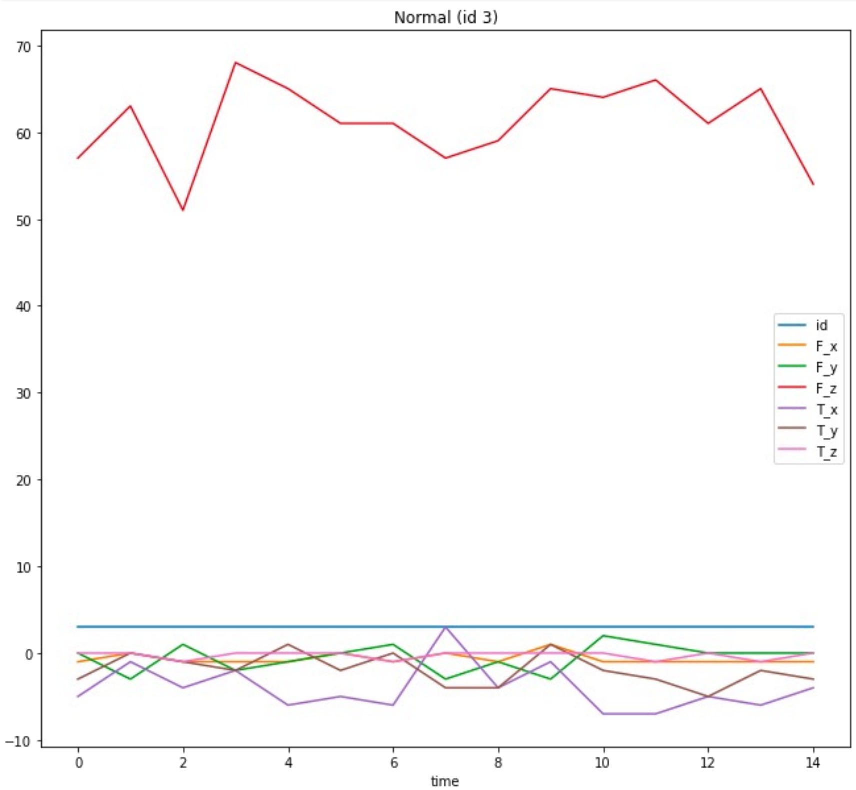
```
1      True
2      True
3      True
4      True
5      True
dtype: bool
```

- The first column is the DataFrame index
- There are six different time series for the different sensors.
- The different robots are denoted by the ids column.

tsfresh - example

- Plot a robot with no reported failure

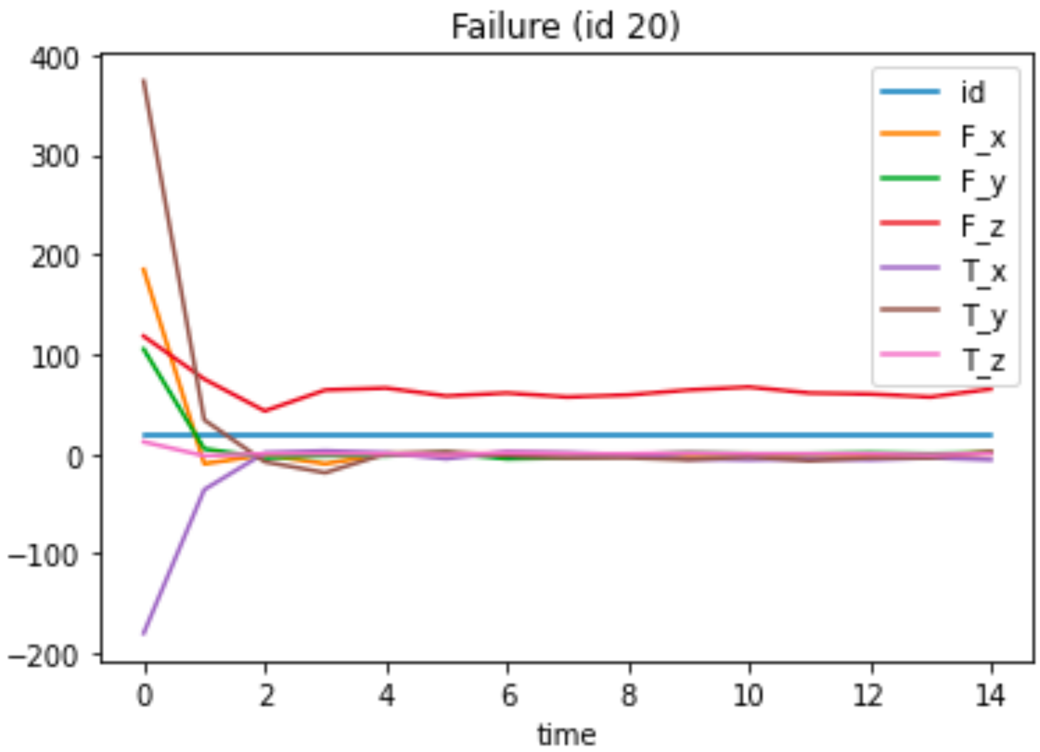
```
import matplotlib.pyplot as plt
normal = timeseries[timeseries.id == 3]
normal.plot(x="time", kind="line", figsize=(10, 10))
plt.title('Normal (id 3)')
```



tsfresh - example

- Plot a robot report failure

```
failure = timeseries[timeseries.id == 20]
failure.plot(x="time", kind="line")
plt.title('Failure (id 20)')
```



tsfresh - example

- Extract relevant feature set from the 6 different time-series
  - extract_features function

```
from tsfresh import extract_features
extracted_features = extract_features(timeseries, column_id='id', column_sort='time')
extracted_features.head()
```

- Features extracted
  - Features > 4000
  - Some of the features include range counts, standard deviation, variance, auto-correlation, linear-trends, quantiles and change in quantiles.

```
extracted_features.head()
```

tsfresh - example

- Select only the relevant features
  - select_features function

```python
from tsfresh import select_features
from tsfresh.utilities.dataframe_functions import impute
extracted_features = impute(extracted_features)
features_filtered = select_features(extracted_features, y)
```

- ~ 600 features were classified as relevant enough.

```python
features_filtered.shape
```

```
(88, 682)
```

tsfresh - example

- Perform the extraction, imputing and filtering at the same time
  - **extract_relevant_features** function

```
# ------------  All in one
from tsfresh import extract_relevant_features
features_direct = extract_relevant_features(timeseries,y,column_id='id',column_sort='time')
```

- ~ 600 features were classified as relevant enough.

- Do we get the same number of selected features?

<u>Hands on session</u>

Problem Solving

Module Topics Covered

- Predictive modeling
- Hypothesis testing
- Parametric and non-parametric tests
- Correlation
- Exploratory Data Analysis (EDA)
- Data plotting and visualization
- Estimating correlation
- Feature Engineering
- Feature Extraction with PCA
- Time series
- Feature extraction in time series