# Конспект по теме "Анализ временных рядов"

## Временные ряды

**Временные ряды** (англ. *time series*) — это последовательности чисел на оси времени. Интервал между значениями ряда постоянный.

Для того, чтобы корректно работать с датой и временем в pandas, нужно корректно изменить тип данных соответствующего столбца с *object* на *datetime64*. Один из вариантов — сделать это прямо при чтении с помощью параметра `parse_dates`. Также, необходимо установить индекс датафрейма, указав в параметре `index_col` список нужных столбцов:

```
# значения параметра parse_dates - список номеров столбцов или названий столбцов
data = pd.read_csv('filename.csv', index_col=[0], parse_dates=[0])
```

Чтобы работать с временными рядами, необходимо проверить, в хронологическом ли порядке расположены даты и время. Для этого нужно посмотреть атрибут индекса таблицы **is_monotonic**. Если порядок соблюдён, атрибут вернёт *True*, если нет — *False:*

```
print(data.index.is_monotonic)
```

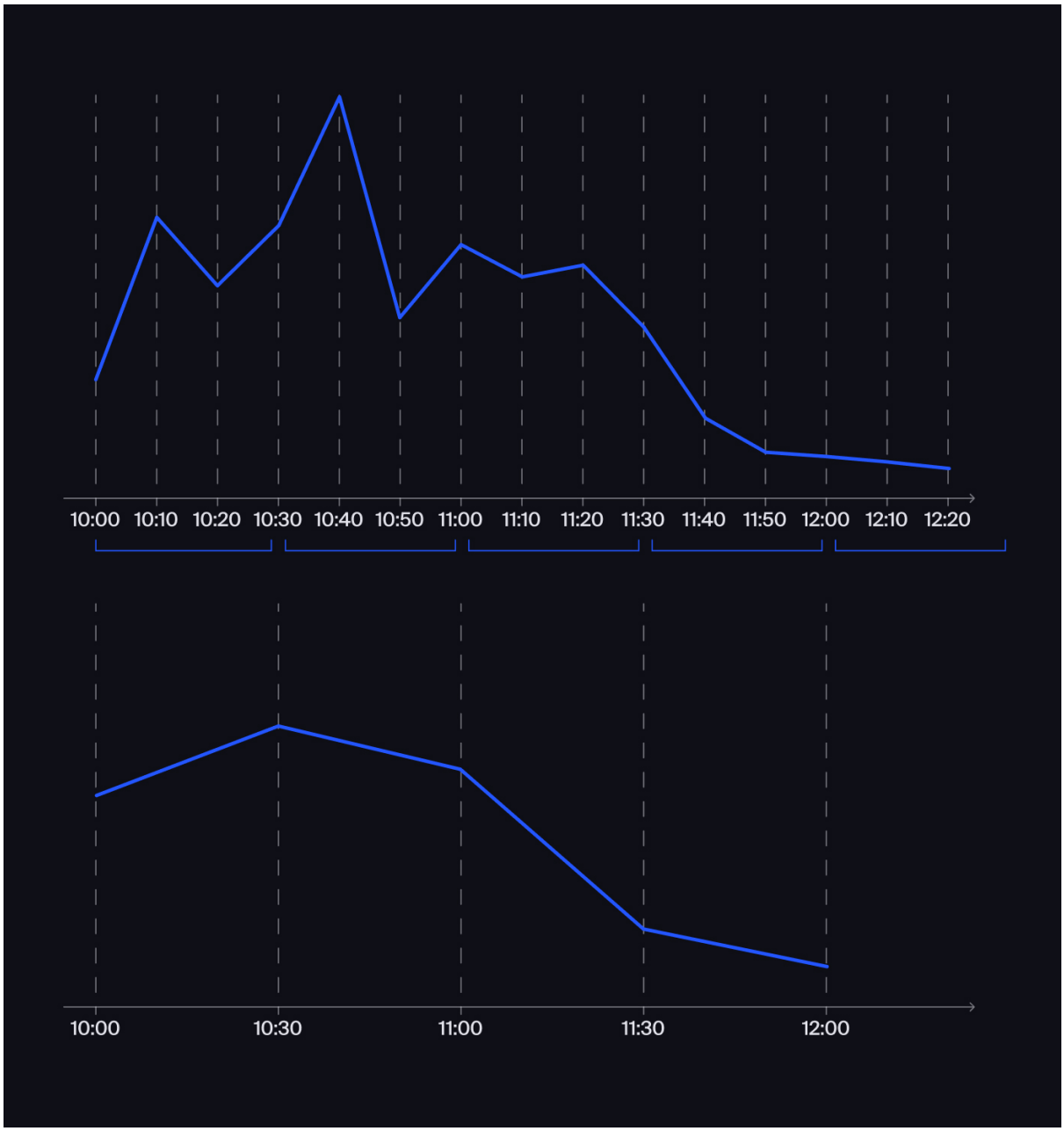
Указав дату и время в качестве индекса, можно производить выборку данных по дате:

```
data = data['2016':'2017']
data = data['2016-01':'2016-06']
data = data['2016-01-01':'2016-01-10']
```

# Ресемплирование

**Ресемплирование**, или **ресемплинг** (англ. *resample*) — это изменение интервала со значениями ряда. Его выполняют в два этапа:

1. Выбирают новую длину интервала. Причём значения из текущего интервала группируются.
2. В каждой группе вычисляется агрегированное значение ряда. Это может быть медиана, среднее, максимум или минимум.



Чтобы поменять интервал и сгруппировать значения, вызовем функцию *resample()*, в аргументе укажем новый интервал:

```
# 1H англ. hour, 1 час
data.resample('1H')

# 2W англ. week, 2 недели
data.resample('2W')
```

Функция *resample()* похожа на *groupby()*. После группировки вызовем функции *mean()* и *max()* для агрегации значений:

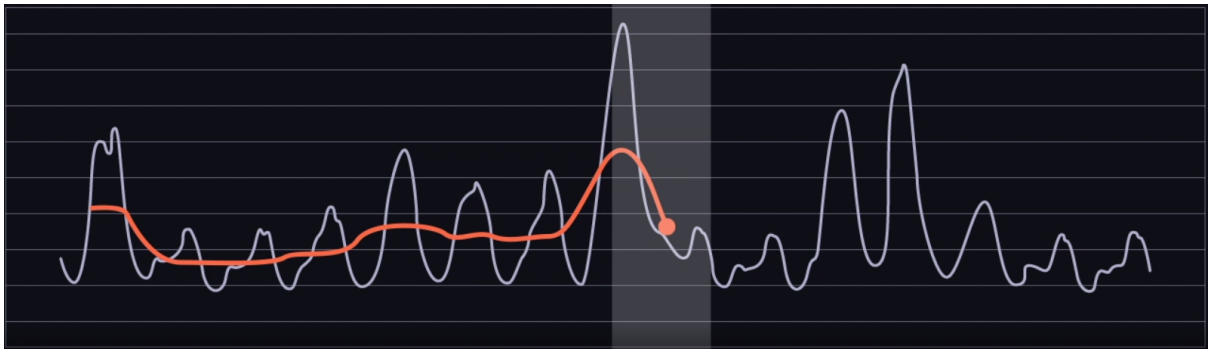
```
# среднее по каждому часу
data.resample('1H').mean()

# максимум по каждым двум неделям
data.resample('2W').max()
```

## Скользящее среднее

**Скользящее среднее** (англ. *rolling mean*), или **движущееся среднее** (англ. *moving average*), — метод сглаживания временных рядов. Его суть заключается в поиске значений, которые меньше всего подвержены колебаниям, то есть средних арифметических.

Метод работает так: экспериментально подбирается **размер окна** (англ. *window size*) — интервал, в котором выполняют усреднение. Чем интервал больше, тем сильнее сглаживание. Затем окно начинает «скользить» почти от начала ряда к его концу, в каждой точке вычисляя среднее значение ряда и тем самым сглаживая его.

В скользящем среднем окна «наслаиваются» друг на друга и не могут выходить за пределы ряда. Поэтому средних будет чуть меньше, чем исходных значений ряда.



В Pandas скользящее среднее вычисляют в два этапа:

1. Вызовом функции *rolling()* создают скользящее окно. В аргументе указывают его размер:

```
# размер окна 7
data.rolling(7)
```

2. Для агрегации значений вызывают функцию *mean()*:

```
# скользящее среднее с окном размером 7
data.rolling(7).mean()
```

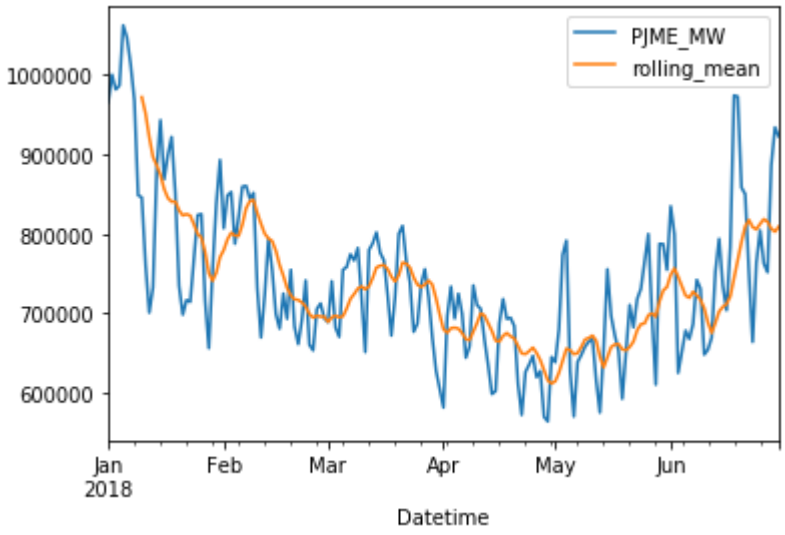
## Тренды и сезонность

**Тренд** (англ. *trend*) — плавное изменение среднего значения ряда без повторяющихся закономерностей. Например, ежегодное увеличение объёма продаж авиабилетов.

**Сезонность** (англ. *seasonality*) — циклично повторяющиеся закономерности во временном ряду. Допустим, рост продаж авиабилетов летом.

Тренды и сезонность зависят от масштаба данных. Нельзя увидеть закономерности, повторяющиеся каждое лето, если есть данные только за год.

Посмотрим на график *rolling_mean*. Увеличение энергопотребления зимой и летом — это тренд.




Если эти данные анализировать в масштабе нескольких лет, рост энергопотребления зимой и летом — это уже сезонные изменения.

В модуле **tsa.seasonal** (от англ. *time series analysis*) библиотеки **statsmodels** есть функция **seasonal_decompose()** . Она раскладывает временной ряд на три составляющие: тренд, сезонность и **остаток** (англ. *residuals*). Это компонента, которая не объясняется трендом и сезонностью, это шум.

```
from statsmodels.tsa.seasonal import seasonal_decompose

decomposed = seasonal_decompose(data)
```

Функция *seasonal_decompose()* принимает временной ряд, а возвращает объект структуры **DecomposeResult**. В нём есть нужные атрибуты:

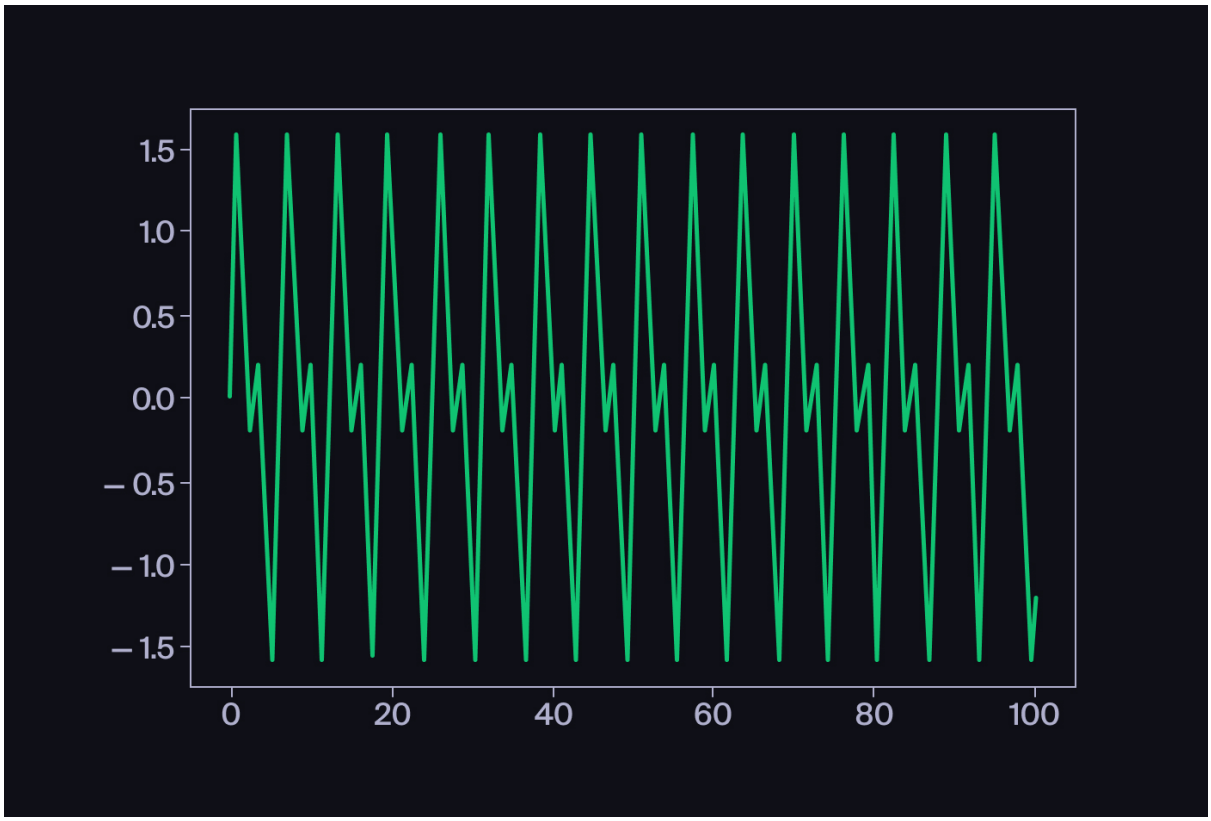
- *decomposed.trend* — тренд;
- *decomposed.seasonal* — сезонная составляющая;
- *decomposed.resid* — остаток декомпозиции.

## Стационарные ряды

В статистике временной ряд описывается **стохастическим процессом** (англ. *stochastic process*). Это случайная величина, у которой со временем

меняется её распределение. У этой величины есть среднее и дисперсия, которые тоже меняются.

Стохастический процесс **стационарный** (англ. *stationary stochastic process*), если его распределение со временем не меняется. Например, к такому процессу относятся периодические колебания значений.




Если распределение меняется, то процесс называется **нестационарным** (англ. *nonstationary stochastic process*).

Узнать распределение временного ряда нельзя. Поэтому **стационарные временные ряды** (англ. *stationary time series*) — это ряды, у которых среднее и стандартное отклонение не меняются. Когда среднее и стандартное отклонение первого ряда меняется медленнее второго, то первый ряд «более стационарный», чем второй.

**Нестационарные ряды** (англ. *nonstationary time series*) прогнозировать сложнее: их свойства меняются слишком быстро.

## Разности временного ряда

**Разности временного ряда** (англ. *time series difference*) — это набор разностей между соседними элементами временного ряда, т. е. из каждого значения вычитается предыдущее.

Для поиска разностей временного ряда применяется метод **shift()** (англ. «сдвиг»). Все значения он сдвигает вдоль временной оси на один шаг вперёд:

```
data = pd.Series([0.5, 0.7, 2.4, 3.2])
print(data)
print(data.shift())
```

```
0    0.5
1    0.7
2    2.4
3    3.2
dtype: float64
0    NaN
1    0.5
2    0.7
3    2.4
dtype: float64
```

Последнее значение ряда пропадает: его сдвигать некуда. На месте нулевого — *NaN*, потому что для него значения нет. Добавим аргумент, чтобы заполнить недостающие значения:

```
import pandas as pd

data = pd.Series([0.5, 0.7, 2.4, 3.2])
print(data)
# англ. заполнить значением
print(data.shift(fill_value=0))
```

```
0    0.5
1    0.7
2    2.4
3    3.2
dtype: float64
0    0.0
1    0.5
2    0.7
3    2.4
dtype: float64
```

Разности временного ряда более стационарны, чем сам ряд. Например, нелинейный тренд преобразуется в линейный: