# Визуализация временных рядов

Проведем импорт необходимых библиотек

try:  
 import pandas as pd   
except:  
 !pip install pandas  
finally:  
 import pandas as pd

import matplotlib.pyplot as plt  
import matplotlib.dates as mdates#Date Parser  
  
import seaborn as sns  
sns.set\_style('white')  
sns.set(rc={'figure.figsize':(11, 4)})

## Загрузка данных в Pandas

урок вдохновлён данным репозиторием <https://github.com/jenfly/opsd>

В качестве набор данных для практики рассмотрим часть набор данных [открытые данные энергетических систем](https://open-power-system-data.org/). Мы будем работать с данным относящимися ко временным рядам (<https://data.open-power-system-data.org/time_series/>). По приведенной ссылке можно найти описание набора данных. Также набор можно найти на странице официального репозитория: <https://github.com/Open-Power-System-Data/time_series>.

Для начала давайте попробуем загрузить последнюю версию набора данных.

# Download hourly data from OPSD website  
url = 'https://data.open-power-system-data.org/time\_series/2020-10-06/'  
datafile = url + 'time\_series\_60min\_singleindex.csv'  
df\_all = pd.read\_csv(datafile, index\_col='utc\_timestamp', parse\_dates=True, low\_memory=False)  
df\_all.head()

[5 rows x 299 columns]

Из всего набора выделим только данные относящиеся к целевой стране. В качестве примера рассмотрим германию. В качестве периода анализа возьмём период с 2015 по 2019 годы.

def extract\_country(df\_all, country\_code, year\_min=None, year\_max=None):  
 """Extract data for a single country"""  
   
 # List of columns to extract  
 columns = [col for col in df\_all.columns if col.startswith(country\_code)]  
   
 # Extract columns and remove country codes from column labels  
 columns\_map = {col : col[3:] for col in columns}  
 df\_out = df\_all[columns].rename(columns=columns\_map)  
   
 # Exclude years outside of specified range, if any  
 if year\_min is not None:  
 df\_out = df\_out[df\_out.index.year >= year\_min]  
 if year\_max is not None:  
 df\_out = df\_out[df\_out.index.year <= year\_max]  
   
 return df\_out  
df\_hrly = extract\_country(df\_all, country\_code='DE', year\_min=2015, year\_max=2019)  
df\_hrly

[43824 rows x 41 columns]

Нам понадобятся не все колонки, поэтому выделим необходимые, кроме того приведем колонки к более интерпретируемому виду

def transform\_dataframe(df, cols\_map):  
 # Rename columns for convenience  
 df = df[list(cols\_map.keys())].rename(columns=cols\_map)  
 # Convert from MW to GW  
 df = df / 1000  
 df = df.resample('D').sum(min\_count=24)  
 df = df.rename\_axis('Date')  
 df.index = df.index.strftime('%Y-%m-%d')  
 return df  
  
cols\_map = {'load\_actual\_entsoe\_transparency' : 'Consumption',  
 'wind\_generation\_actual' : 'Wind',  
 'solar\_generation\_actual' : 'Solar'}  
df\_daily = transform\_dataframe(df\_hrly, cols\_map)  
  
# Compute wind + solar generation  
df\_daily['Wind+Solar'] = df\_daily[['Wind', 'Solar']].sum(axis=1, skipna=False)  
df\_daily.to\_csv('de\_data.csv')  
df\_daily.head()

Consumption Wind Solar Wind+Solar  
Date   
2015-01-01 1088.317 325.165 NaN NaN  
2015-01-02 1246.588 603.554 7.757 611.311  
2015-01-03 1117.554 462.955 7.237 470.192  
2015-01-04 1081.980 385.023 19.982 405.005  
2015-01-05 1325.920 216.540 26.522 243.062

Теперь набор представляет собой ежедневное потребление электричества (в гига-Ваттах в час) в Германии. Набор включает следующие временные ряды в виде колонок:

* Date — дата в формате гггг-мм-дд;
* Consumption — Общее потребление, ГВт/ч;
* Wind — Потребление ветряной энергии, ГВт/ч;
* Solar — Потребление солнечной энергии, ГВт/ч;
* Wind+Solar — Потребление энергии из альтернативных источников, ГВт/ч.

Проведем анализ сформированного набора данных

path\_ts = 'de\_data.csv'  
  
df = pd.read\_csv(path\_ts)  
  
df.sample(5, random\_state=0)

Date Consumption Wind Solar Wind+Solar  
793 2017-03-04 1224.314 253.250 99.948 353.198  
789 2017-02-28 1509.014 602.221 49.109 651.330  
118 2015-04-29 1365.365 153.288 188.720 342.008  
318 2015-11-15 1176.237 530.791 21.279 552.070  
891 2017-06-10 1132.742 89.475 214.438 303.913

Размер набора данных

print(df.shape)

(1826, 5)

Информация о колонках в наборе

df.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 1826 entries, 0 to 1825  
Data columns (total 5 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 Date 1826 non-null object   
 1 Consumption 1826 non-null float64  
 2 Wind 1819 non-null float64  
 3 Solar 1818 non-null float64  
 4 Wind+Solar 1817 non-null float64  
dtypes: float64(4), object(1)  
memory usage: 71.5+ KB

Описание суммарных статистик данных

df.describe()

Consumption Wind Solar Wind+Solar  
count 1826.000000 1819.000000 1818.000000 1817.000000  
mean 1340.623081 267.842699 103.386051 371.257950  
std 156.544793 192.922094 66.797470 178.690531  
min 934.864000 16.482000 4.989000 27.529000  
25% 1213.262000 120.906500 38.559500 242.188000  
50% 1374.256000 211.989000 101.705500 338.443000  
75% 1455.366500 364.868000 159.751250 470.921000  
max 1636.406000 998.899000 264.538000 1034.494000

типы данных

df.dtypes

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

Введение индексов-дат

df.Date = pd.to\_datetime(df.Date)  
df.set\_index('Date', inplace=True)  
df.sample(15, random\_state=0)

Consumption Wind Solar Wind+Solar  
Date   
2017-03-04 1224.314 253.250 99.948 353.198  
2017-02-28 1509.014 602.221 49.109 651.330  
2015-04-29 1365.365 153.288 188.720 342.008  
2015-11-15 1176.237 530.791 21.279 552.070  
2017-06-10 1132.742 89.475 214.438 303.913  
2017-06-03 1126.704 97.750 165.064 262.814  
2015-12-14 1479.403 110.010 27.409 137.419  
2019-10-15 1441.266 214.059 101.408 315.467  
2016-04-27 1456.925 390.802 106.074 496.876  
2015-01-07 1442.712 227.204 17.114 244.318  
2017-01-14 1345.572 487.188 16.765 503.953  
2019-08-17 1100.024 391.408 98.425 489.833  
2019-02-06 1573.666 362.081 54.640 416.721  
2015-07-11 1122.407 50.020 193.951 243.971  
2017-06-23 1407.411 427.695 169.412 597.107

df.info()

<class 'pandas.core.frame.DataFrame'>  
DatetimeIndex: 1826 entries, 2015-01-01 to 2019-12-31  
Data columns (total 4 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 Consumption 1826 non-null float64  
 1 Wind 1819 non-null float64  
 2 Solar 1818 non-null float64  
 3 Wind+Solar 1817 non-null float64  
dtypes: float64(4)  
memory usage: 71.3 KB

Отметим, что на самом деле можно было сразу загрузить данные в таком виде, чтобы индексы были датами

df = pd.read\_csv(path\_ts, parse\_dates=['Date'], index\_col="Date")  
df.head()

Consumption Wind Solar Wind+Solar  
Date   
2015-01-01 1088.317 325.165 NaN NaN  
2015-01-02 1246.588 603.554 7.757 611.311  
2015-01-03 1117.554 462.955 7.237 470.192  
2015-01-04 1081.980 385.023 19.982 405.005  
2015-01-05 1325.920 216.540 26.522 243.062

Данные обработанные в форме дат DateTimeIndex позволяют работать с индексом как с датой

print(df.index.day)  
print(df.index.weekday)  
print(df.index.year)

Int64Index([ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,  
 ...  
 22, 23, 24, 25, 26, 27, 28, 29, 30, 31],  
 dtype='int64', name='Date', length=1826)  
Int64Index([3, 4, 5, 6, 0, 1, 2, 3, 4, 5,  
 ...  
 6, 0, 1, 2, 3, 4, 5, 6, 0, 1],  
 dtype='int64', name='Date', length=1826)  
Int64Index([2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015,  
 ...  
 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019],  
 dtype='int64', name='Date', length=1826)

Теперь мы можем обращаться к данным по дате

df.loc['2017-08-10']

Consumption 1342.278  
Wind 100.274  
Solar 71.162  
Wind+Solar 171.436  
Name: 2017-08-10 00:00:00, dtype: float64

или по выборке дат

df.loc['2014-12-31':'2015-01-22']

Consumption Wind Solar Wind+Solar  
Date   
2015-01-01 1088.317 325.165 NaN NaN  
2015-01-02 1246.588 603.554 7.757 611.311  
2015-01-03 1117.554 462.955 7.237 470.192  
2015-01-04 1081.980 385.023 19.982 405.005  
2015-01-05 1325.920 216.540 26.522 243.062  
2015-01-06 1331.916 117.227 32.888 150.115  
2015-01-07 1442.712 227.204 17.114 244.318  
2015-01-08 1474.375 440.318 8.599 448.917  
2015-01-09 1459.409 641.729 6.822 648.551  
2015-01-10 1242.817 634.677 20.473 655.150  
2015-01-11 1162.956 630.378 19.811 650.189  
2015-01-12 1485.927 631.832 11.025 642.857  
2015-01-13 1462.804 511.733 50.283 562.016  
2015-01-14 1462.715 406.982 18.106 425.088  
2015-01-15 1479.245 556.471 17.155 573.626  
2015-01-16 1431.414 249.323 22.818 272.141  
2015-01-17 1227.210 137.717 21.405 159.122  
2015-01-18 1139.695 127.215 31.259 158.474  
2015-01-19 1457.299 31.620 23.681 55.301  
2015-01-20 1489.253 16.738 10.791 27.529  
2015-01-21 1486.707 50.390 22.120 72.510  
2015-01-22 1496.905 62.295 12.119 74.414

а также обращаться к конкретной колонке по дате

df.loc['2015-12-20':'2015-12-25', 'Wind']

Date  
2015-12-20 370.889  
2015-12-21 583.385  
2015-12-22 638.509  
2015-12-23 552.048  
2015-12-24 450.480  
2015-12-25 420.503  
Name: Wind, dtype: float64

Или использовать колонку как переменную

df.Wind.loc['2015-12-20':'2015-12-25']

Date  
2015-12-20 370.889  
2015-12-21 583.385  
2015-12-22 638.509  
2015-12-23 552.048  
2015-12-24 450.480  
2015-12-25 420.503  
Name: Wind, dtype: float64

а также к каждой колонке можно обращаться по ключу

df[['Wind']].loc['2015-12-20':'2015-12-25']

Wind  
Date   
2015-12-20 370.889  
2015-12-21 583.385  
2015-12-22 638.509  
2015-12-23 552.048  
2015-12-24 450.480  
2015-12-25 420.503

df['Wind'].loc['2015-12-20':'2015-12-25']

Date  
2015-12-20 370.889  
2015-12-21 583.385  
2015-12-22 638.509  
2015-12-23 552.048  
2015-12-24 450.480  
2015-12-25 420.503  
Name: Wind, dtype: float64

также можно обращаться по индексу чрез метод iloc

df.iloc[0:2,0:3]

Consumption Wind Solar  
Date   
2015-01-01 1088.317 325.165 NaN  
2015-01-02 1246.588 603.554 7.757

Данные можно представлять с нужной частотой при помощи метода asfreq, например с частотой D - день, W,M,Y для недели, месяца и года соответственно.

df[['Wind']].loc['2015-10-20':'2015-12-25'].asfreq('W')

Wind  
Date   
2015-10-25 107.528  
2015-11-01 77.941  
2015-11-08 317.087  
2015-11-15 530.791  
2015-11-22 295.045  
2015-11-29 696.636  
2015-12-06 589.628  
2015-12-13 288.092  
2015-12-20 370.889

можно сделать обращение по месяцу с периодом 1 неделя

df.loc['2016-02'].asfreq('W')

Consumption Wind Solar Wind+Solar  
Date   
2016-02-07 1158.086 515.965 38.073 554.038  
2016-02-14 1179.309 280.405 26.907 307.312  
2016-02-21 1192.219 641.776 22.521 664.297  
2016-02-28 1155.177 187.644 71.204 258.848

или точно также с использованием открытой нотации обращения к массиву

df.loc['2015':].asfreq('Y')

Consumption Wind Solar Wind+Solar  
Date   
2015-12-31 1122.732 306.450 20.226 326.676  
2016-12-31 1146.908 315.059 34.580 349.639  
2017-12-31 1061.793 721.172 19.979 741.151  
2018-12-31 1148.098 245.130 9.327 254.457  
2019-12-31 1124.933 425.225 36.562 461.787

Полагаем, что в данном случае будет визуально правильней поменять индексы на значение года

df.loc['2012':].asfreq('Y').set\_index(df.loc['2012':].asfreq('Y').index.year)

Consumption Wind Solar Wind+Solar  
Date   
2015 1122.732 306.450 20.226 326.676  
2016 1146.908 315.059 34.580 349.639  
2017 1061.793 721.172 19.979 741.151  
2018 1148.098 245.130 9.327 254.457  
2019 1124.933 425.225 36.562 461.787

Для индексов в формате дат также доступно группирование методом groupby. Группирование groupby происходит по заданным периодам, например W, 'Y' или A (год), '2y' (по 2 года) и т.д.

Часто после использования методов groupby, asfreq, а также groupby используется некоторая функция итога, например, sum, mean, median or std.

df.groupby(pd.Grouper(freq='1y')).sum()

Consumption Wind Solar Wind+Solar  
Date   
2015-12-31 479496.047 77172.286 34779.771 111419.348  
2016-12-31 486995.370 74996.242 34146.371 109089.308  
2017-12-31 492116.437 102670.983 35882.978 138553.961  
2018-12-31 498895.226 108564.492 41231.973 149796.465  
2019-12-31 490474.666 123801.867 41914.747 165716.614

df.resample('1w').median().head(3)

Consumption Wind Solar Wind+Solar  
Date   
2015-01-04 1102.9355 423.989 7.757 470.192  
2015-01-11 1331.9160 440.318 19.811 448.917  
2015-01-18 1462.7150 406.982 21.405 425.088

df.asfreq('1w').head(3)

Consumption Wind Solar Wind+Solar  
Date   
2015-01-04 1081.980 385.023 19.982 405.005  
2015-01-11 1162.956 630.378 19.811 650.189  
2015-01-18 1139.695 127.215 31.259 158.474

The full list of frequencies with its description can be find in this book <https://jakevdp.github.io/PythonDataScienceHandbook/03.11-working-with-time-series.html>

Также, полагаем, что в наборе достаточно много значений NaN. Давайте их уберем. Для этого в Pandas есть несколько инструментов, в том числе ffill, bfill для заполнения пропусков соответственно следующими или предыдущими значениями, например можно использовать метод так .asfreq('D', method='ffill'). Также возможны использования методов удаления пропусков dropna или заполнения заданными значениями filna. Если используется метод dropna, то из данных будет удалена вся строка с пропуском.

Давайте для начала посмотрим сколько у нас NaN значений. Для этого можно использовать или метод isnull или isna .

df.isna().sum()

Consumption 0  
Wind 7  
Solar 8  
Wind+Solar 9  
dtype: int64

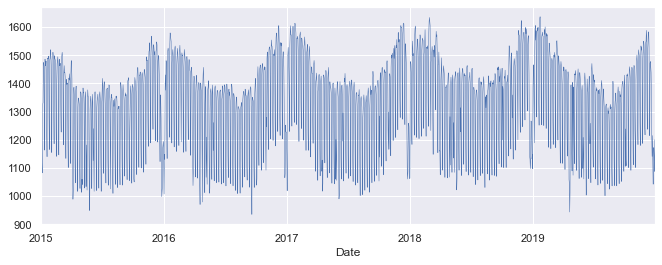
df.fillna(0, inplace=True)  
df.head(3)

Consumption Wind Solar Wind+Solar  
Date   
2015-01-01 1088.317 325.165 0.000 0.000  
2015-01-02 1246.588 603.554 7.757 611.311  
2015-01-03 1117.554 462.955 7.237 470.192

## Визуализация временного ряда

Для начала давайте посмотрим на наиболее простой тип визуализации для одного из столбцов наших данных

df['Consumption'].plot(linewidth=0.5);

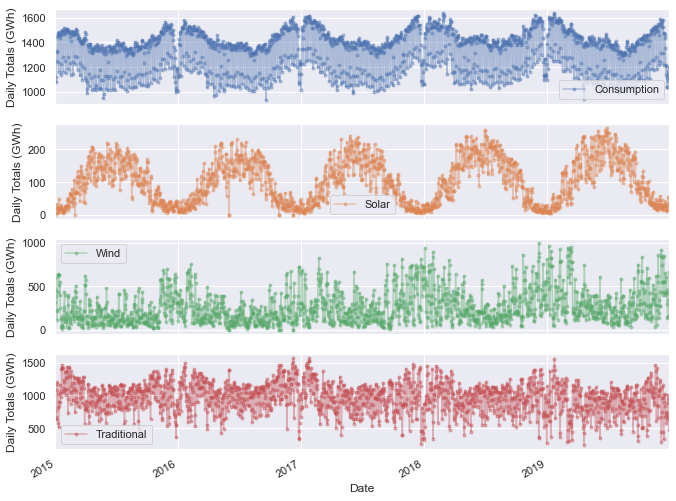


В наших данных есть составляющее общего потребления, солнечной и ветряной энергии. Полагаем, что нам может потребоваться столбец, соответствующий другим источникам (не альтернативным).

df['Traditional'] = df['Consumption'] - df['Solar'] - df['Wind']

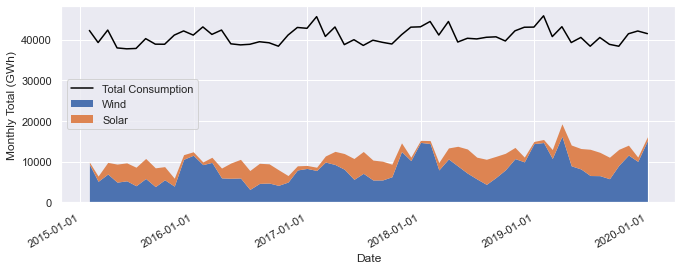
Также давайте попробуем визуализировать и остальные столбцы

cols\_plot = ['Consumption', 'Solar', 'Wind','Traditional']  
axes = df[cols\_plot].plot(marker='.', alpha=0.4, linestyle='-', figsize=(11, 9), subplots=True)  
for ax in axes:  
 ax.set\_ylabel('Daily Totals (GWh)')



Теперь попробуем провести визуализацию на одном графике, с периодом 1 месяц

df\_monthly = df.resample('M').sum(min\_count=7)  
  
  
fig, ax = plt.subplots()  
ax.plot(df\_monthly['Consumption'], color='black', label='Total Consumption')  
  
df\_monthly[['Wind', 'Solar']].plot.area(ax=ax, linewidth=0)  
  
ax.xaxis.set\_major\_locator(mdates.YearLocator())  
ax.legend()  
ax.set\_ylabel('Monthly Total (GWh)');



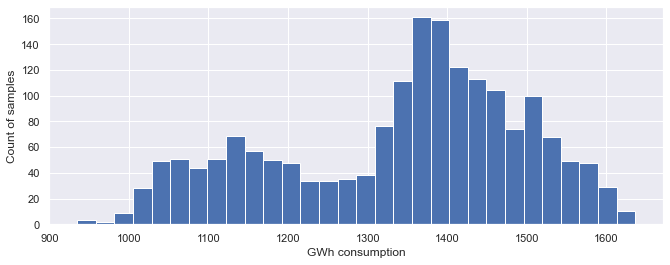
Анализ полученных графиков показывает следующее:

* Все три графика осциллируют во времени в течение года. Рискнем предположить, что это связано с сезонностью и изменением погоды.
* Потребление электроэнергии выше зимой и ниже летом.
* Потребление энергии имеет 2 сезонных составляющих:
  + основная, с примерным диапазоном значений 1300-1500 ГВт
  + дополнительная со значениями порядка 1100 ГВт, которая предположительно связана с изменением потребления в течение неделе. Это также подтверждается заметным снижением потребления в начале каждого года.
* Пик производства солнечной энергии приходится на лето.
* Пик производства ветряной энергии приходится на зиму, причем колебания этого ряда куда более подвержены дисперсии. Полагаем, что это связано с погодным фактором.
* Значение альтернативных источников энергии растет, но очень медленно.
* Общее электропотребление, а также потребление из альтернативных источников имеют растущий тренд, тогда как тренд традиционных источников - спадающий.

Анализ общего потребления

Давайте подробней изучить сезонность по подробнее. Для анализа гипотезы о наличии двух составляющих в общем потреблении давайте посмотрим на распределение значений.

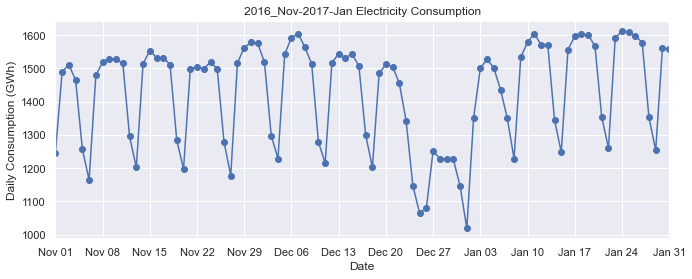
ax = df.Consumption.hist(bins=30)  
# ax = df.Consumption.plot(kind='kde' )  
ax.set\_ylabel('Count of samples')  
ax.set\_xlabel('GWh consumption')  
plt.show()



В распределении мы видим две составляющих, предположительно соответствующих двум периодам в данных. Пики этих составляющих приходятся на 1100 и 1400 ГВт.

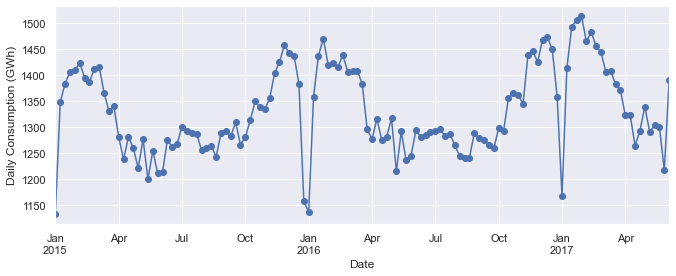
построим график потребления для периода в 2 месяца, включая начала и конец года

ax = df.loc['2016-11':'2017-01', 'Consumption'].plot(marker='o', linestyle='-')  
ax.set\_ylabel('Daily Consumption (GWh)')  
ax.set\_title('2016\_Nov-2017-Jan Electricity Consumption')  
  
# For more convinient ticks (week ticks)  
ax.xaxis.set\_major\_locator(mdates.WeekdayLocator())  
# Format 3-letter month name and day number  
ax.xaxis.set\_major\_formatter(mdates.DateFormatter('%b %d'))  
plt.show()



Действительно у нас имеется две сезонных компоненты. Спад потребления связан с выходными). Однако, Потребление во время смены годов можно считать или аномалией или третьей составляющей (что возможно правильней). Проверим гипотезу о 3 составляющей. Для этого используем скользящее среднее.

ax = df.loc['2013-10':'2017-05', 'Consumption'].\  
 resample('W').mean().plot(marker='o', linestyle='-',linewidth=1.5)  
ax.set\_ylabel('Daily Consumption (GWh)')  
plt.show()

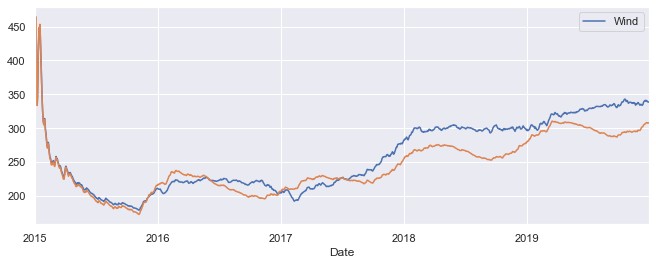


Анализ потребления ветровой энергии

Также давайте проверим наличие тренда у ветровой составляющей (полагаем, что тут тренд не очевиден). Для этого используем скользящее вреднее при помощи метода rolling. Данный метод можно выполнить с заданным периодом, в частях года или днях. Отметим, что также можно воспользоваться методом скользящего среднего

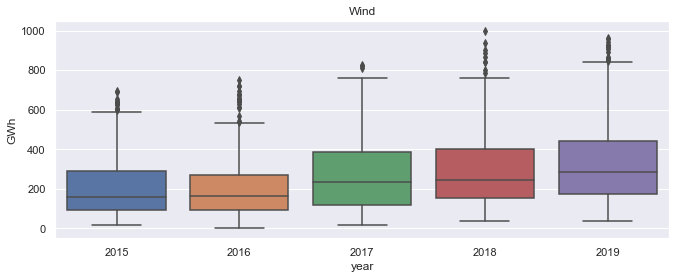
df[['Wind']].rolling('365d').mean().plot( linewidth=1.5, );  
df.Wind.ewm(halflife=365, min\_periods=0,adjust=True).mean().plot()

<AxesSubplot:xlabel='Date'>



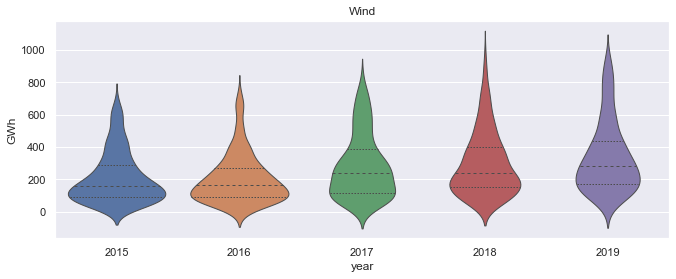
Также проверим тренд при помощи построения BBox

ax = sns.boxplot(data=df, x=df.index.year, y='Wind')  
ax.set\_ylabel('GWh')  
ax.set\_xlabel('year')  
ax.set\_title('Wind')  
plt.show()



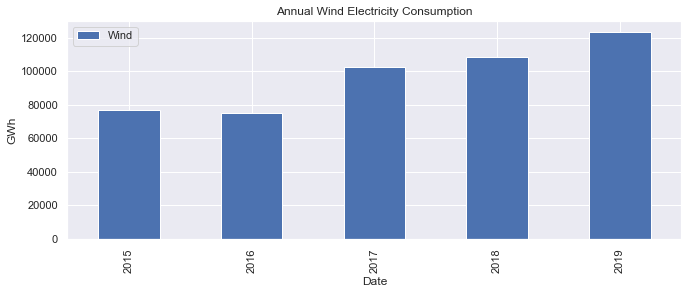
Отметим, что по мимо BBox можно также воспользоваться альтернативным отображением

ax=sns.violinplot(data=df, x=df.index.year, y='Wind',  
 split=True, inner="quart", linewidth=1, )  
ax.set\_ylabel('GWh')  
ax.set\_xlabel('year')  
ax.set\_title('Wind')  
plt.show()



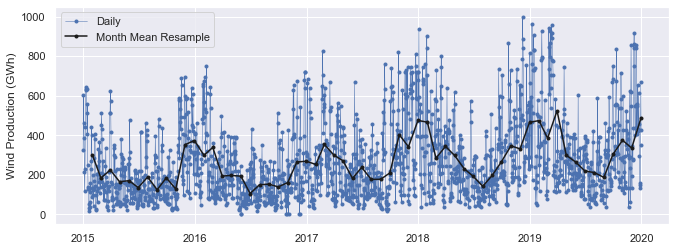
Однако, также можно проверить гипотезу при помощи bar-plot

df\_test = df[['Wind']].resample('Y').sum()  
  
ax = df\_test.set\_index(df\_test.index.year).plot.bar()  
ax.set\_title('Annual Wind Electricity Consumption')  
ax.set\_ylabel('GWh');



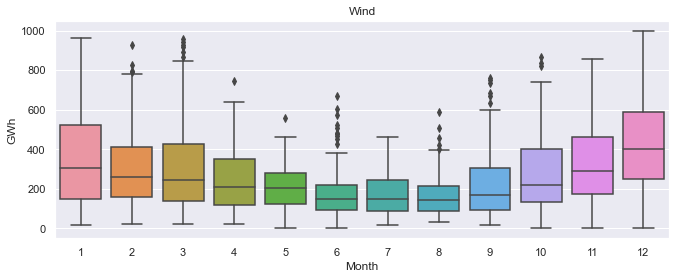
Также давайте проверим наличие сезонных составляющих у ряда с ветряной энергией.

start, end = '2015-01', '2019-12'  
  
fig, ax = plt.subplots()  
ax.plot(df.loc[start:end, 'Wind'],marker='.', linestyle='-', linewidth=0.5, label='Daily')  
ax.plot(df.resample('M').mean().loc[start:end, 'Wind'], marker='o', markersize=3, linestyle='-', label='Month Mean Resample', color='k')  
ax.set\_ylabel('Wind Production (GWh)')  
ax.legend();



Вероятно, сезонность нестабильна, чтобы это проверить давайте построим boxplot месячных значений

ax = sns.boxplot(data=df, x=df.index.month, y='Wind')  
ax.set\_ylabel('GWh')  
ax.set\_xlabel('Month')  
ax.set\_title('Wind')  
plt.show()

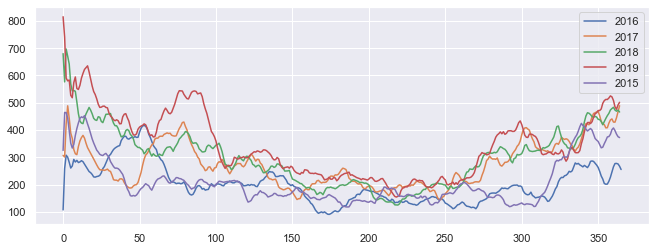


На boxplot видно большое количество выброс и изменение среднего значения и дисперсии.

В дополнение мы можем проверить нестационарность сезонности ветровой составляющей с использованием скользящего среднего по годам. Пример показан ниже.

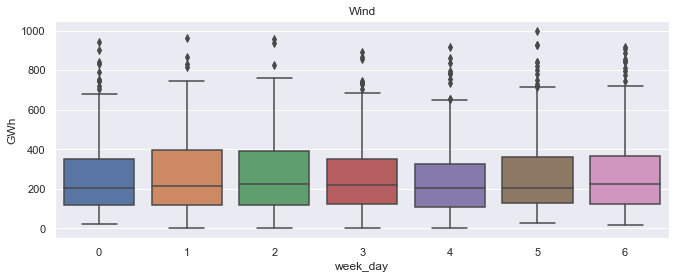
for year in list(set(df.index.year)):  
 plt.plot(df.loc[str(year):str(year), 'Wind'].rolling('30d').mean().values, label=year)  
plt.legend()

<matplotlib.legend.Legend at 0x1a014397eb8>



Вероятно, нестационарность сезонности увеличивается от года к году. Также оценим сезонность дней недели.

ax = sns.boxplot(data=df, x=df.index.weekday, y='Wind');  
ax.set\_ylabel('GWh')  
ax.set\_xlabel('week\_day')  
ax.set\_title('Wind')  
plt.show()



на графике не видно явной сезонности или нестационарности, однако число выбросов достаточно большое.

В качестве упражнения слушателю предлагается провести анализ солнечной энергии и традиционной энергии

Вопросы:

1 Укажите на то, как вычислить столбец традиционные источники для наших данных

**Ответ**: df['Consumption'] - df['Solar'] - df['Wind']

Или df['Consumption'] - df['Solar+Wind']

Мы взяли шаго данных 1 день, сколько сезонных составляющих мы обнаружили в данных для общего потребления:

**Ответ**: 2

2 Взглянув на данные для солнечного потребления, какая часть ряда там преобладает: тренд, сезонность или шумы.

**Ответ**: сезонность.

3 Мы оценили наличие тренда в ветряной энергии, как можно охарактеризовать этот тренд:

**А линейный, возрастающий**

Б монотонный с насыщением

В тренд с точками перегиба

Г тренд имеет некоторую цикличность

**Ответ** : А