# prediction

March 28, 2021

## 1 Prediction of the number of bicycle passing between 00:01 AM and 09:00 AM on Friday, April 2nd

### 1.1 The required libraries

[144]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib
import math
import itertools
import warnings
from statsmodels import api as sm
from statsmodels.graphics.tsaplots import plot_pacf
from statsmodels.graphics.tsaplots import plot_acf
from matplotlib import pyplot as plt
warnings.filterwarnings("ignore")
sns.set()
sns.set_style('whitegrid')
%matplotlib inline
```

### 1.2 The dataset

[332]:

```
data = pd.read_csv("data_bike.csv")
#data
```

[333]:

```
data_clean = data.copy()
data_clean = data_clean.drop([0, 8], axis = 0)
data_clean = data_clean.drop(columns = ['Vélos depuis le 1er janvier / Grand␣
 ↪total', 'Unnamed: 4', 'Remarque'])
data_clean = data_clean.dropna(how = 'any')
data_clean = data_clean.rename(columns = {"Vélos ce jour / Today's total":␣
 ↪'Bicycle'})
data_clean['Bicycle'] = data_clean['Bicycle'].astype(int)
#data_clean
```

[334]:

```
data_clean['Date'] = data_clean['Date'] + " " + data_clean['Heure / Time']
data_clean = data_clean.set_index('Date')
data_clean = data_clean.drop('Heure / Time', 1)
data_clean.index = pd.to_datetime(data_clean.index,format="%d/%m/%Y %H:%M:%S")
data_clean = data_clean.sort_index()
#data_clean.head(20)
```

[335]:

```
serie_bike = data_clean.copy()
serie_bike = serie_bike[(serie_bike.index.time >= pd.to_datetime("00:01:00").
 ↪time()) &
                  (serie_bike.index.time <= pd.to_datetime("09:01:00").time())]
serie_bike = serie_bike.loc['2021-01-16':'2021-03-25']
#serie_bike
```

## 1.3 Handling missing data

[336]:

```
data_week = data_clean.copy()
data_week = data_week[(data_week.index.time >= pd.to_datetime("00:01:00").
 ↪time()) &
                  (data_week.index.time <= pd.to_datetime("09:15:00").time())]
data_week = data_week.loc['2021-01-16':'2021-03-25']
data_week['Weekday'] = pd.to_datetime(data_week.index)
data_week['Weekday'] = data_week['Weekday'].dt.day_name()
data_week['Weekday'] = data_week['Weekday'].apply(str)
data_week = data_week.query(' Weekday == ["Monday", "Tuesday", "Wednesday",␣
 ↪"Thursday", "Friday"]')
data_week = data_week.drop([data_week.index[5], data_week.index[13],data_week.
 ↪index[14],data_week.index[17],data_week.index[22],data_week.
 ↪index[23],data_week.index[24],data_week.index[25],data_week.index[27]])
print('mean week', data_week.mean().apply(np.floor), 'median week', data_week.
 ↪median().apply(np.floor), data_week_end)
```

[337]:

```
data_week_end = data_clean.copy()
data_week_end = data_week_end[(data_week_end.index.time >= pd.to_datetime("00:
 ↪01:00").time()) &
                  (data_week_end.index.time <= pd.to_datetime("10:00:00").
 ↪time())]
data_week_end = data_week_end.loc['2021-01-16':'2021-03-25']
data_week_end['Weekday'] = pd.to_datetime(data_week_end.index)
data_week_end['Weekday'] = data_week_end['Weekday'].dt.day_name()
data_week_end['Weekday'] = data_week_end['Weekday'].apply(str)
data_week_end = data_week_end.query(' Weekday == ["Saturday", "Sunday"]')
data_week_end = data_week_end.drop([data_week_end.index[2], data_week_end.
 ↪index[6]])
print('mean week-end', data_week_end.mean(), data_week_end)
```

[363]:

```
serie_bike = serie_bike.drop([serie_bike.index[7],serie_bike.index[15],␣
 ↪serie_bike.index[16], serie_bike.index[20], serie_bike.index[25], serie_bike.
 ↪index[26], serie_bike.index[27], serie_bike.index[28], serie_bike.index[30]])
serie_bike = serie_bike.resample("1D").sum() #1D for 1 day : day per day
serie_bike['Weekday'] = pd.to_datetime(serie_bike.index)
serie_bike['Weekday'] = serie_bike['Weekday'].dt.day_name()
serie_bike['Weekday'] = serie_bike['Weekday'].apply(str)
serie_bike.head(50)
```

[370]:

```
for i in range(0,len(serie_bike)):
    if serie_bike.iat[i,0]==0 and (serie_bike.iat[i,1]=="Saturday" or␣
 ↪serie_bike.iat[i,1]=="Sunday"):
        serie_bike.loc[serie_bike.index[i],'Bicycle'] = 55
    elif serie_bike.iat[i,0]==0 and (serie_bike.iat[i,1]!="Saturday" and␣
 ↪serie_bike.iat[i,1]!="Sunday"):
            serie_bike.loc[serie_bike.index[i],'Bicycle'] = 182

serie_bike.head(50)
```

[370]:

```
            Bicycle    Weekday
Date
2021-01-17       15     Sunday
2021-01-18      188     Monday
2021-01-19      182    Tuesday
2021-01-20      182  Wednesday
2021-01-21      186   Thursday
2021-01-22      182     Friday
2021-01-23       55   Saturday
2021-01-24       55     Sunday
2021-01-25      189     Monday
2021-01-26      182    Tuesday
2021-01-27      182  Wednesday
2021-01-28      178   Thursday
2021-01-29       35     Friday
2021-01-30       55   Saturday
2021-01-31       55     Sunday
2021-02-01      182     Monday
2021-02-02      182    Tuesday
2021-02-03      182  Wednesday
2021-02-04      183   Thursday
2021-02-05      151     Friday
2021-02-06       55   Saturday
2021-02-07       55     Sunday
2021-02-08      170     Monday
2021-02-09      182    Tuesday
2021-02-10      182  Wednesday
2021-02-11      182   Thursday
```

```
2021-02-12      182     Friday
2021-02-13       55   Saturday
2021-02-14       55     Sunday
2021-02-15      182     Monday
2021-02-16      182    Tuesday
2021-02-17      182  Wednesday
2021-02-18      182   Thursday
2021-02-19      186     Friday
2021-02-20       55   Saturday
2021-02-21       55     Sunday
2021-02-22      182     Monday
2021-02-23      182    Tuesday
2021-02-24      182  Wednesday
2021-02-25      182   Thursday
2021-02-26      182     Friday
2021-02-27       15   Saturday
2021-02-28       55     Sunday
2021-03-01      182     Monday
2021-03-02      182    Tuesday
2021-03-03      182  Wednesday
2021-03-04      187   Thursday
2021-03-05      182     Friday
2021-03-06       55   Saturday
2021-03-07       55     Sunday
```

[373]:

```
serie_bike = serie_bike.drop('Weekday',1)
#serie_bike
```

[373]:

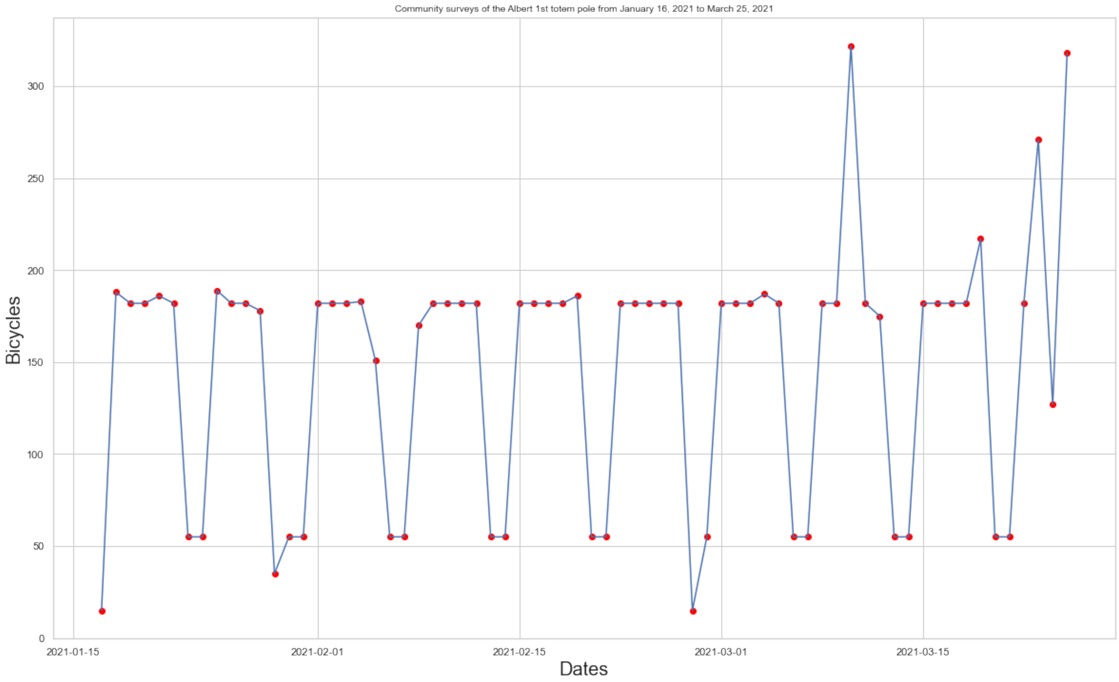
```
            Bicycle
Date
2021-01-17       15
2021-01-18      188
2021-01-19      182
2021-01-20      182
2021-01-21      186
…                 …
2021-03-21       55
2021-03-22      182
2021-03-23      271
2021-03-24      127
2021-03-25      318

[68 rows x 1 columns]
```

## 1.4 Data visualization

[375]:

```
plt.figure(figsize=(20,12))
plt.plot(serie_bike)
plt.scatter(serie_bike.index,serie_bike, color='red')
plt.xlabel('Dates', size=20)
plt.ylabel('Bicycles', size=20)
graph = plt.title('Community surveys of the Albert 1st totem pole from January␣
 ↪16 to March 25, 2021')
graph.set_fontsize(10)
```



## 1.5 Creating a commonly used method for time-series, SARIMA

### 1.5.1 parameters set

[442]:

```
p = d = q = range(0,2)
pdq = list(itertools.product(p,d,q))
ses = [(x[0], x[1], x[2], 7) for x in pdq]
```

[443]:

```
for param in pdq:
    for ses_param in ses:
        mod1 = sm.tsa.statespace.
 ↪SARIMAX(serie_bike,order=param,seasonal_order=ses_param,enforce_stationarity=False,␣
 ↪enforce_invertibility=False)
        results = mod1.fit()
    print(f'AIC:{results.aic}')
print("Done")
```

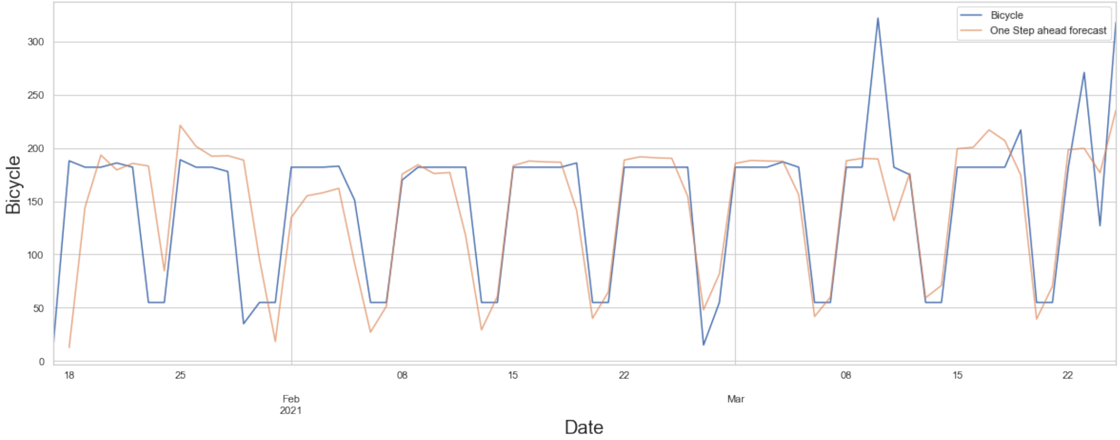
```
AIC:536.6855262864601
AIC:525.1787976162552
AIC:570.7886661888169
AIC:521.8523795973554
AIC:531.9164008427946
AIC:523.9759732606194
AIC:539.0700370425398
AIC:514.6300121719221
Done
```

### 1.5.2 Predicting observed values

```
[444]: pred = results.get_prediction(start=1, dynamic=False)
       pred_conf_int = pred.conf_int()
```

```
[447]: ax = serie_bike.plot(label='Observed')
       pred.predicted_mean.plot(ax=ax, label='One Step ahead forecast', alpha=.7,␣
        ↪figsize=(20, 7))
       #ax.fill_between(pred_conf_int.index,
       #                 pred_conf_int.iloc[:, 0],
       #                 pred_conf_int.iloc[:, 1], color='k', alpha=.2)
       ax.set_xlabel('Date', size=20)
       ax.set_ylabel('Bicycle',size=20)
       plt.legend()
```

```
[447]: <matplotlib.legend.Legend at 0x7fe364d5b4c0>
```



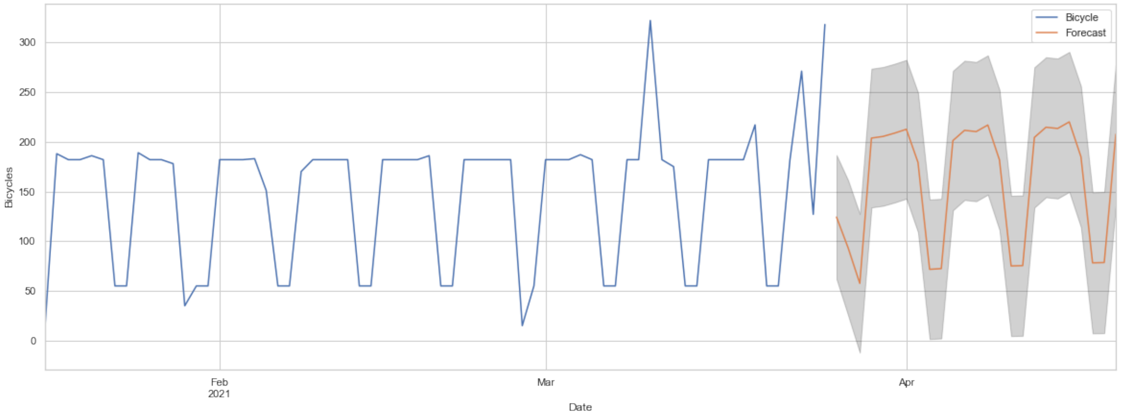
### 1.5.3 Forecasting

```
[448]: predict = results.get_forecast(steps=25)
       predict_conf_int = predict.conf_int()
```

```
[449]: predict.predicted_mean.head()
```

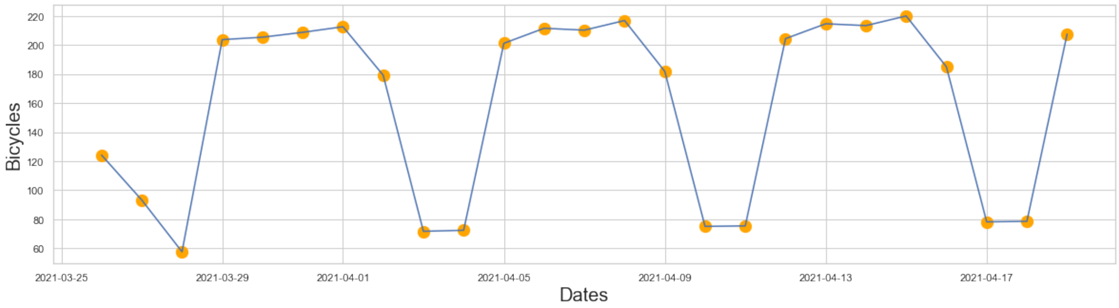
```
[449]: 2021-03-26    124.201971
        2021-03-27     93.210020
        2021-03-28     57.573932
        2021-03-29    203.723651
        2021-03-30    205.392620
        Freq: D, Name: predicted_mean, dtype: float64
```

```
[451]: ax = serie_bike.plot(label='observed', figsize=(20, 7))
       predict.predicted_mean.plot(ax=ax, label='Forecast')
       ax.fill_between(predict_conf_int.index,
                       predict_conf_int.iloc[:, 0],
                       predict_conf_int.iloc[:, 1], color='k', alpha=.2)
       ax.set_xlabel('Date')
       ax.set_ylabel('Bicycles')
       plt.legend()
       plt.show()
```



```
[452]: plt.figure(figsize=(20,5))
       plt.plot(predict.predicted_mean)
       plt.scatter(predict.predicted_mean.index, predict.predicted_mean.values,␣
        ↪color="orange", s=150)
       plt.xlabel("Dates", size=20)
       plt.ylabel("Bicycles", size=20)
```

```
[452]: Text(0, 0.5, 'Bicycles')
```



### 1.6 Bicycles passing between 00:01 - 09:00 on April 2, 2021

[453]:
```
april_2_9AM = predict.predicted_mean['2021-04-02']
```

[454]:
```
print(f"Predicted number: {round(april_2_9AM)}")
```

```
Predicted number: 179
```