What is a time series?

According to the wikipedia, A time series is a series of data points indexed (or listed or graphed) in time order. Most commonly, a time series is a sequence taken at successive equally spaced points in time. Thus it is a sequence of discrete-time data. For example, stock prices over a fixed period of time, hotel bookings, ecommerce sales, waether cycle reports etc.

Time series analysis comprises methods for analyzing time series data in order to extract meaningful statistics and other characteristics of the data. Time series forecasting is the use of a model to predict future values based on previously observed values.

Let's discuss a few definitions related to time series first.

Definitions

- **Level:** Level is the average of the values of the series.
- **Trend:** Trend shows a pattern in the data. For example, whether the stock prices are increasing with time(uptrend) or are they decreasing with time(downtrend) or time doesn't have that much effect on the prices(Horizontal trend)

Image Courtsey: Financial Hub

- **Seasonality:** When the data shows a repetative pattern for over an year, it can be termed as seasonal pattern. For example, the sale of airconditioners will increase every year during summer and the sale will decrease during winter.
- **Cyclic Patterns:** These are the repetative patterns shown over a longer period of time(more than one year). For example, after every five year the share market has some fluctuations due to the general elections.
- **Noise:** The variations which do not show any pattern.

Let's now take an example to see what was done before the advent of Time Series Analysis.

Let's say that we have a problem at hand where we have been asked to predict the sales of skiing products for a sports manufacturer. You can do the predictions using the following methods:

Old Methods

- **Using Average:** You might give the prediction as the average of all the prevous values.
- **Using Moving Average:** This is the average of the previous values over a fixed period. For example you might predict the sales in November based on the average of past 3 months. The past three months will be August, September and October. If you are predicting the sales for December, the past three months will be September, October

and November. Although the number of months considered are same but the window moved from one set of months to another. Hence the name Moving Average.

- **Using the Naive Method:** The Naive method says that the prediction will be same as the last figure. For example, the prediction for November will be the sales for October.
- **Using the Seasonal Naive Method:** Seasonal naive method is similar to naive method. Here, the new prediction is equal to the sales for the previous season.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

ARIMA

In [24]:
```
import numpy as np
from scipy import stats
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm

from statsmodels.graphics.api import qqplot
%matplotlib inline
```

In [3]:
```
female_birth_data=pd.read_csv("daily-total-female-births-CA.csv") # This is
```

In [4]:
```
female_birth_data.head()
```

Out[4]:

	date	births
0	1959-01-01	35
1	1959-01-02	32
2	1959-01-03	30
3	1959-01-04	31
4	1959-01-05	44

In [6]:
```
birth_data=pd.read_csv("daily-total-female-births-CA.csv", index_col=[0], pa
```

In [7]:
```
birth_data.head()
```

Out[7]:

	births
date	
1959-01-01	35
1959-01-02	32
1959-01-03	30
1959-01-04	31
1959-01-05	44

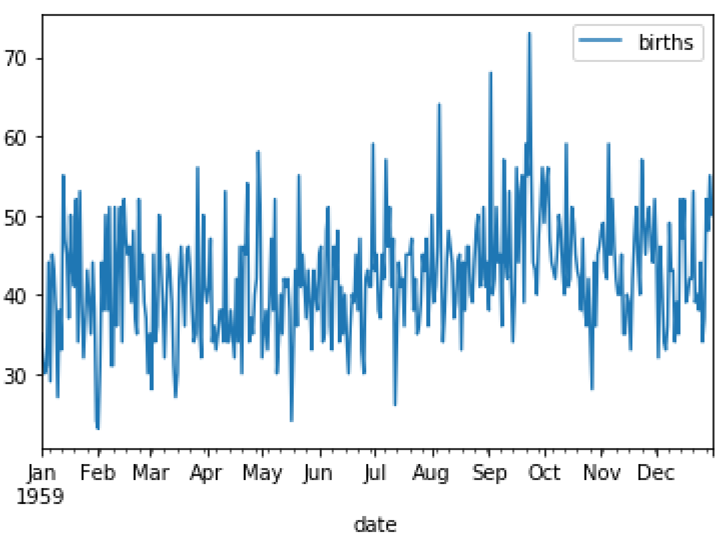
In [8]:
```
birth_data.describe()
```

Out[8]:

	births
count	365.000000
mean	41.980822
std	7.348257
min	23.000000
25%	37.000000
50%	42.000000
75%	46.000000
max	73.000000

In [9]:
```
birth_data.plot() #almost a stationary series
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x1f14fd95198>



In [21]:
```
# also called as smoothing
moving_average_birth=birth_data.rolling(window=20).mean()  # window: This is
```

Loading [MathJax]/extensions/Safe.js

In [22]:
```
moving_average_birth
```

Out[22]:

date	births
1959-01-01	NaN
1959-01-02	NaN
1959-01-03	NaN
1959-01-04	NaN
1959-01-05	NaN
1959-01-06	NaN
1959-01-07	NaN
1959-01-08	NaN
1959-01-09	NaN
1959-01-10	NaN
1959-01-11	NaN
1959-01-12	NaN
1959-01-13	NaN
1959-01-14	NaN
1959-01-15	NaN
1959-01-16	NaN
1959-01-17	NaN
1959-01-18	NaN
1959-01-19	NaN
1959-01-20	39.75
1959-01-21	39.70
1959-01-22	40.75
1959-01-23	41.20
1959-01-24	41.25
1959-01-25	40.90
1959-01-26	41.60
1959-01-27	41.30
1959-01-28	40.90
1959-01-29	41.20
1959-01-30	41.75
...	...
1959-12-02	42.80

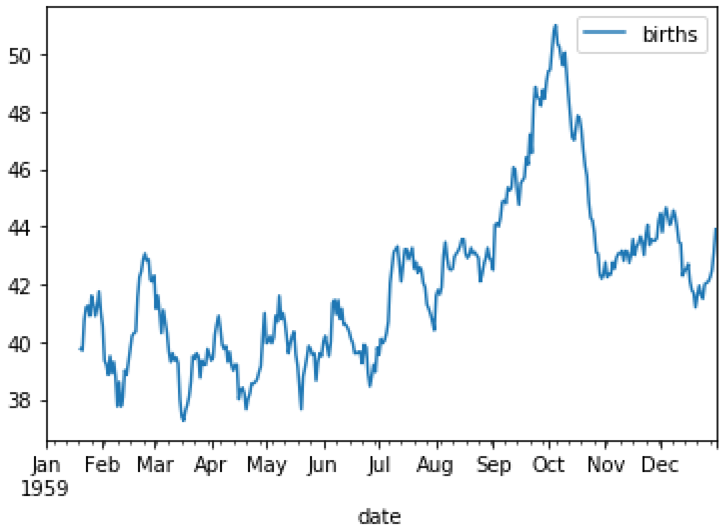
Loading [MathJax]/extensions/Safe.js

	births
date	
1959-12-03	44.35
1959-12-04	44.65
1959-12-05	44.35
1959-12-06	44.05
1959-12-07	44.20
1959-12-08	44.55
1959-12-09	44.35
1959-12-10	43.95
1959-12-11	43.45
1959-12-12	43.40
1959-12-13	42.30
1959-12-14	42.45
1959-12-15	42.55
1959-12-16	42.70
1959-12-17	42.10
1959-12-18	41.80
1959-12-19	41.70
1959-12-20	41.20
1959-12-21	41.60
1959-12-22	41.95
1959-12-23	41.65
1959-12-24	41.50
1959-12-25	42.00
1959-12-26	42.05
1959-12-27	42.10
1959-12-28	42.25
1959-12-29	42.50
1959-12-30	43.10
1959-12-31	43.90

365 rows × 1 columns

In [23]:
```
moving_average_birth.plot() # we can see that there is a peak in the month o
```

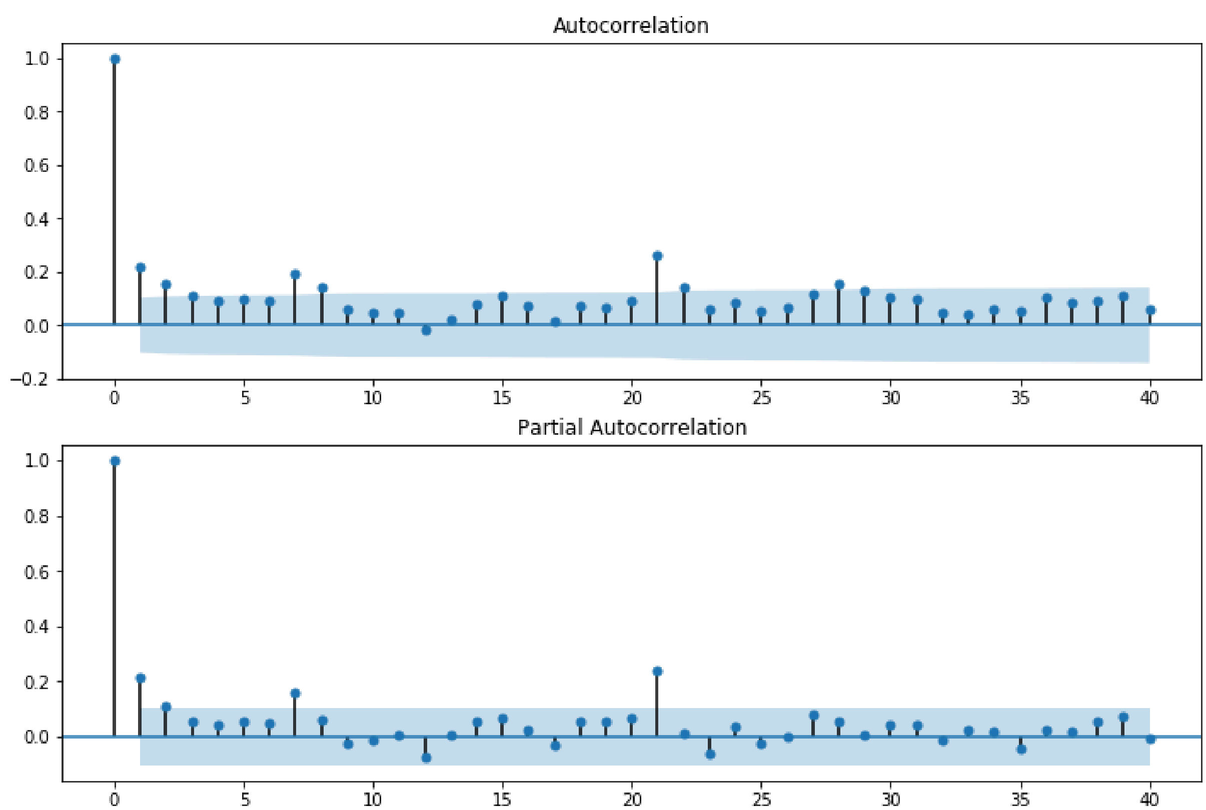
Out[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1f150907b00>
```



In [25]:
```python
sm.stats.durbin_watson(birth_data) # very less correlation
```

Out[25]:
```
array([0.04624491])
```

In [26]:
```python
# show plots in the notebook
%matplotlib inline
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(birth_data.values.squeeze(), lags=40, ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(birth_data, lags=40, ax=ax2)
```



Loading [MathJax]/extensions/Safe.js

In [27]:
```
training_data=birth_data[0:320]
test_data=birth_data[320:]
```

In [31]:
```
"""
from sklearn.model_selection import train_test_split
t_x,t=train_test_split(birth_data)
""""""
```

In [1]:
```
from statsmodels.tsa.arima_model import ARIMA
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [1], in <cell line: 1>()
----> 1 from statsmodels.tsa.arima_model import ARIMA

File ~\Anaconda3\lib\site-packages\statsmodels\tsa\__init__.py:1, in <module>
----> 1 from statsmodels.tools._testing import PytestTester
      3 test = PytestTester()

File ~\Anaconda3\lib\site-packages\statsmodels\tools\__init__.py:1, in <modul
e>
----> 1 from .tools import add_constant, categorical
      2 from statsmodels.tools._testing import PytestTester
      4 __all__ = ['test', 'add_constant', 'categorical']

File ~\Anaconda3\lib\site-packages\statsmodels\tools\tools.py:7, in <module>
      5 import numpy.lib.recfunctions as nprf
      6 import pandas as pd
----> 7 import scipy.linalg
      9 from statsmodels.compat.python import lzip, lmap
     11 from statsmodels.tools.data import _is_using_pandas, _is_recarray

File ~\Anaconda3\lib\site-packages\scipy\linalg\__init__.py:213, in <module>
    211 from ._procrustes import *
    212 from ._decomp_update import *
--> 213 from ._sketches import *
    215 __all__ = [s for s in dir() if not s.startswith('_')]
    217 from numpy.dual import register_func

File ~\Anaconda3\lib\site-packages\scipy\linalg\_sketches.py:11, in <module>
      8 import numpy as np
     10 from scipy._lib._util import check_random_state
---> 11 from scipy.sparse import csc_matrix
     13 __all__ = ['clarkson_woodruff_transform']
     16 def cwt_matrix(n_rows, n_columns, seed=None):

File ~\Anaconda3\lib\site-packages\scipy\sparse\__init__.py:229, in <module>
    223 # Original code by Travis Oliphant.
    224 # Modified and extended by Ed Schofield, Robert Cimrman,
    225 # Nathan Bell, and Jake Vanderplas.
    227 import warnings as _warnings
--> 229 from .base import *
    230 from .csr import *
    231 from .csc import *

File ~\Anaconda3\lib\site-packages\scipy\sparse\base.py:8, in <module>
      6 from scipy._lib.six import xrange
      7 from scipy._lib._numpy_compat import broadcast_to
----> 8 from .sputils import (isdense, isscalarlike, isintlike,
      9                       get_sum_dtype, validateaxis, check_reshape_kwar
gs,
     10                       check_shape, asmatrix)
     12 __all__ = ['spmatrix', 'isspmatrix', 'issparse',
     13            'SparseWarning', 'SparseEfficiencyWarning']
     16 class SparseWarning(Warning):
```

Loading [MathJax]/extensions/Safe.js

```
File ~\Anaconda3\lib\site-packages\scipy\sparse\sputils.py:17, in <module>
     11 __all__ = ['upcast', 'getdtype', 'isscalarlike', 'isintlike',
     12            'isshape', 'issequence', 'isdense', 'ismatrix', 'get_sum_d
type']
     14 supported_dtypes = ['bool', 'int8', 'uint8', 'short', 'ushort', 'int
c',
     15                     'uintc', 'longlong', 'ulonglong', 'single', 'doub
le',
     16                     'longdouble', 'csingle', 'cdouble', 'clongdoubl
e']
---> 17 supported_dtypes = [np.typeDict[x] for x in supported_dtypes]
     19 _upcast_memo = {}
     22 def upcast(*args):

File ~\Anaconda3\lib\site-packages\scipy\sparse\sputils.py:17, in <listcomp>
(.0)
     11 __all__ = ['upcast', 'getdtype', 'isscalarlike', 'isintlike',
     12            'isshape', 'issequence', 'isdense', 'ismatrix', 'get_sum_d
type']
     14 supported_dtypes = ['bool', 'int8', 'uint8', 'short', 'ushort', 'int
c',
     15                     'uintc', 'longlong', 'ulonglong', 'single', 'doub
le',
     16                     'longdouble', 'csingle', 'cdouble', 'clongdoubl
e']
---> 17 supported_dtypes = [np.typeDict[x] for x in supported_dtypes]
     19 _upcast_memo = {}
     22 def upcast(*args):

File ~\Anaconda3\lib\site-packages\numpy\__init__.py:320, in __getattr__(att
r)
    317     from .testing import Tester
    318     return Tester
--> 320 raise AttributeError("module {!r} has no attribute "
    321                      "{!r}".format(__name__, attr))

AttributeError: module 'numpy' has no attribute 'typeDict'
```

In [ ]:

In [45]:
```
arima= ARIMA(training_data,order=(2,1,3))
```

```
C:\Users\virat\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:
171: ValueWarning: No frequency information was provided, so inferred frequen
cy D will be used.
  % freq, ValueWarning)
C:\Users\virat\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:
191: FutureWarning: Creating a DatetimeIndex by passing range endpoints is de
precated.  Use `pandas.date_range` instead.
  start=index[0], end=index[-1], freq=freq)
C:\Users\virat\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:
171: ValueWarning: No frequency information was provided, so inferred frequen
cy D will be used.
  % freq, ValueWarning)
```

In [46]:
```
model=arima.fit()
```

In [47]:
```
model.aic
```

Out[47]:
```
2159.076974912458
```

In [48]:
```
pred= model.forecast(steps=45)[0]
```

In [49]:
```
pred
```

Out[49]:
```
array([44.20336376, 44.64661409, 43.40243639, 43.18642407, 42.46860908,
       43.44949502, 43.80021262, 44.81159761, 44.23852903, 43.97639274,
       42.83728932, 43.04815868, 43.14699235, 44.33687764, 44.50162656,
       44.71946592, 43.69098356, 43.32064023, 42.82208716, 43.6233574 ,
       44.11878344, 44.92162375, 44.49438888, 44.08981115, 43.14260896,
       43.23608944, 43.4795044 , 44.49993718, 44.76125302, 44.83139948,
       43.94563728, 43.49084713, 43.1407838 , 43.82776885, 44.4038833 ,
       45.06229125, 44.72230914, 44.2341347 , 43.42110948, 43.44996156,
       43.78382252, 44.68646475, 44.99650633, 44.96865567, 44.17958866])
```

In [50]:
```
from sklearn.metrics import mean_squared_error
```

In [51]:
```
np.sqrt(mean_squared_error(test_data,pred))
```

Out[51]:
```
6.419420721712673
```