Educational Materials (Assignments) for Introductory Course in Data Science and AI

ALEXANDER STOTSKY
Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg SE - 412 96
SWEDEN
alexander.stotsky2@telia.com

Programming Assignment 2: Cleaning & Processing of Atmospheric Measurements

ALEXANDER STOTSKY
Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg SE - 412 96
SWEDEN
alexander.stotsky2@telia.com

Description

Under the National Oceanic and Atmospheric Administration, the National Weather Service provides daily weather reports for cities across USA. This is done through the use of 122 different Weather Forcast Offices throughout the country. These WFOs are responsible for the daily weather reports for serveral cities throughout their region of coverage. This data set takes the information from these WFO reports for cities across the country and summarizes it at the weekly level for all of 2016.
In this assignment you will import and clean the dataset which contains atmospheric measurements across USA. You will pickup the measurements acquired in California during summer time of 2016, convert the Fahrenheit scale to the Celsius temperature scale, present the temperature distribution in the form of histogram and calculate the mean value and standard deviation of the temperature, which will be the final result of your data processing.
The assignment will be graded automatically by the codegrade software. The codegrade will check your final result i.e., the mean value and standard deviation of the temperature in California during summer time of 2016. This is a simple and rapid (not questionwise) procedure which is easily perfomed by the codegrade. Automatic grading tolerates possible roundoff errors, which are always present in finite digit calculations. If your calculations are not correct, the codegrade will check intermediate variables and verify relations providing you with the guidance for improvement of your code aiming to correct calculation of the output.
The procedure of finding correct solution associated with the feedback provided by automatic grading can also be seen as stepwise interactive learning.
The assignment should be accomplished (as usual) as the answers to the questions presented in the form of the Python code. Notebook template, which contains questions to be answered, please find in the file notebook template. The codegrade converts automatically your Jupyter notebook to executable Python code.
Please read attentively the questions in the template which include recommendations associated with the names of variables and functions. Please follow these recommendations which are needed for automatic evaluation and grading of your code.
The original data which is needed for processing, please find in the file data and the list of measured variables and more information about the measurements, please find in the external link to CORGIS Dataset Project .

Good luck !

Figure 1: Processing of Atmospheric Measurements

Programming Assignment 2: Data Cleaning

Under the National Oceanic and Atmospheric Administration, the National Weather Service provides daily weather reports for cities across USA. This is done through the use of 122 different Weather Forcast Offices throughout the country. These WFOs are responsible for the daily weather reports for serveral cities throughout their region of coverage. This data set takes the information from these WFO reports for cities across the country and summarizes it at the weekly level for all of 2016.

Note about the data

The data was provided by the CORGIS Dataset Project, Weather dataset

Question 1: Import libs and read file with specified delimiter

Import pandas, numpy and matplotlib using standard aliases. Read the file `weather.txt` and save the result in a variable called `df`. Use `,""` as delimiter and the following column names 'Precipitation','Full','Month','Week of','Year','Station City','Station Code','Station Location','State','Taverage','Temperature Max Temp','Temperature Min Temp','Wind Direction','Wind Speed'. To supress warning specify `engine=python` in the command `pd.read_csv`. Examine your input file by entering command `df`. In addition and for further convenience you can ignore warnings by importing lib `warnings` and entering command `warnings.filterwarnings("ignore")`

In [1]:
```
# Codegrade Tag Question1
# Write all your code for this answer in this cell
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
df = pd.read_csv('weather.txt',delimiter= ',""', names=[  'Precipitation','Date'
df
```

Out[1]:

	Precipitation	Date	Month	Week of	Year	Station City	Station Code	Statior Locatior
0	"0.0	2016-01-03""	1""	3""	2016""	Birmingham""	BHM""	Birmingham AL"'
1	"0.0	2016-01-03""	1""	3""	2016""	Huntsville""	HSV""	Huntsville AL"'
2	"0.16	2016-01-03""	1""	3""	2016""	Mobile""	MOB""	Mobile, AL"'
3	"0.0	2016-01-03""	1""	3""	2016""	Montgomery""	MGM""	Montgomery AL"'
4	"0.01	2016-01-03""	1""	3""	2016""	Anchorage""	ANC""	Anchorage AK"'
...	...	...	...	...	...	...	...	..
16738	"0.08	2017-01-01""	1""	1""	2017""	Casper""	CPR""	Casper, WY"'
16739	"0.0	2017-01-01""	1""	1""	2017""	Cheyenne""	CYS""	Cheyenne WY"'
16740	"0.0	2017-01-01""	1""	1""	2017""	Lander""	LND""	Lander, WY"'
16741	"0.06	2017-01-01""	1""	1""	2017""	Rawlins""	RWL""	Rawlins WY"'
16742	"0.1	2017-01-01""	1""	1""	2017""	Sheridan""	SHR""	Sheridan WY"'

16743 rows × 14 columns

Question 2: Cleaning dataframe df

Examine your dataframe `df` and delete symbols which are not needed. You can use attribute replace associated with `df`. Examine your dataframe after clearning. Hint: delete `"` in whole `df`.

In [2]:
```
# Codegrade Tag Question2
# Write all your code for this answer in this cell
df = df.replace('"','',regex=True)
df
```

Out[2]:

	Precipitation	Date	Month	Week of	Year	Station City	Station Code	Station Location
0	0.0	2016-01-03	1	3	2016	Birmingham	BHM	Birmingham, AL
1	0.0	2016-01-03	1	3	2016	Huntsville	HSV	Huntsville, AL
2	0.16	2016-01-03	1	3	2016	Mobile	MOB	Mobile, AL
3	0.0	2016-01-03	1	3	2016	Montgomery	MGM	Montgomery, AL
4	0.01	2016-01-03	1	3	2016	Anchorage	ANC	Anchorage, AK
...	...	...	...	...	...	...	...	...
16738	0.08	2017-01-01	1	1	2017	Casper	CPR	Casper, WY
16739	0.0	2017-01-01	1	1	2017	Cheyenne	CYS	Cheyenne, WY
16740	0.0	2017-01-01	1	1	2017	Lander	LND	Lander, WY
16741	0.06	2017-01-01	1	1	2017	Rawlins	RWL	Rawlins, WY
16742	0.1	2017-01-01	1	1	2017	Sheridan	SHR	Sheridan, WY

16743 rows × 14 columns

Question 3: Data type conversion & selection of subset of dataframe with respect to time

Identify data type for the column `Date` in the dataframe using attribute `dtypes` . Convert the column `Date` to `datetime64[ns]` format using `Pandas.to_datetime()` function which is used to convert different data types into datetime objects. Check the data type of column `Date` after conversion. Select the subset `df_summer` of dataframe `df` for summer time between `2016-06-01` and `2016-08-30` using `datetime64` attribute in `numpy` . Examine the subset by entering the command `df_summer`

In [3]:
```
# Codegrade Tag Question3
# Write all your code for this answer in this cell
df['Date'] = pd.to_datetime(df['Date'])
df_summer = df[(df['Date'] >= np.datetime64('2016-08-01')) & \
(df['Date'] <= np.datetime64('2016-08-30')) ]
df_summer
```

Out[3]:

	Precipitation	Date	Month	Week of	Year	Station City	Station Code	Station Location	
6947	0.44	2016-06-05	6	5	2016	Birmingham	BHM	Birmingham, AL	
6948	2.56	2016-06-05	6	5	2016	Huntsville	HSV	Huntsville, AL	
6949	4.03	2016-06-05	6	5	2016	Mobile	MOB	Mobile, AL	
6950	0.39	2016-06-05	6	5	2016	Montgomery	MGM	Montgomery, AL	
6951	0.0	2016-06-05	6	5	2016	Anchorage	ANC	Anchorage, AK	
...	...	...	...	...	...	...	...	...	
11044	0.78	2016-08-28	8	28	2016	Casper	CPR	Casper, WY	\
11045	0.21	2016-08-28	8	28	2016	Cheyenne	CYS	Cheyenne, WY	\
11046	0.01	2016-08-28	8	28	2016	Lander	LND	Lander, WY	\
11047	0.0	2016-08-28	8	28	2016	Rawlins	RWL	Rawlins, WY	\
11048	0.0	2016-08-28	8	28	2016	Sheridan	SHR	Sheridan, WY	\

4102 rows × 14 columns

Question 4: Further selection of subset in dataframe

Create dataframe with measurements in state California during summer time in `df_summer`. Name the dataframe as `df_California_summer`. Examine new dataframe by entering command `df_California_summer`.

In [4]:
```
# Codegrade Tag Question4
# Write all your code for this answer in this cell
df_California_summer = df_summer[(df_summer['State'] == 'California') ]
df_California_summer
```

Out[4]:

	Precipitation	Date	Month	Week of	Year	Station City	Station Code	Station Location	S
6992	0.0	2016-06-05	6	5	2016	Bakersfield	BFL	Bakersfield, CA	Califc
6993	0.0	2016-06-05	6	5	2016	Bishop	BIH	Bishop, CA	Califc
6994	0.0	2016-06-05	6	5	2016	China Lake	NID	China Lake, CA	Califc
6995	0.0	2016-06-05	6	5	2016	Concord	CCR	Concord, CA	Califc
6996	0.0	2016-06-05	6	5	2016	Eureka	EKA	Eureka, CA	Califc
...	...	...	...	...	...	...	...	...	
10791	0.0	2016-08-28	8	28	2016	San Francisco	SFO	San Francisco, CA	Califc
10792	0.0	2016-08-28	8	28	2016	Sandberg	SDB	Sandberg, CA	Califc
10793	0.0	2016-08-28	8	28	2016	Santa Barbara	SBA	Santa Barbara, CA	Califc
10794	0.0	2016-08-28	8	28	2016	Santa Maria	SMX	Santa Maria, CA	Califc
10795	0.0	2016-08-28	8	28	2016	Stockton	SCK	Stockton, CA	Califc

247 rows × 14 columns

Question 5: Conversion of Fahrenheit temperature scale to Celsius scale

Create the function which converts the Fahrenheit scale to the Celsius temperature scale. Notice that temperature in Celsius scale, `T_c` can be calcuated using measurements in Fahrenheit scale, `T_f` as follows: `T_c = (T_f - 32) * 5 / 9` Name this function as `convert_f_to_c` and specify temperature in Fahrenheit scale as input.

In [5]:
```
#Codegrade Tag Question5
# Write all your code for this answer in this cell
def convert_f_to_c(temp_in_fahrenheit):
    return (temp_in_fahrenheit - 32) * 5 / 9 + 0.5
```

Question 6: Conversion of summer temperature in California

Convert measurements of average summer temperature in California `Taverage` to dataframe degrees of Celsius using dataframe `df_California_summer` and user defined function `convert_f_to_c`. Insert the result of conversion in the column `Taverage` of the dataframe `df_California_summer`

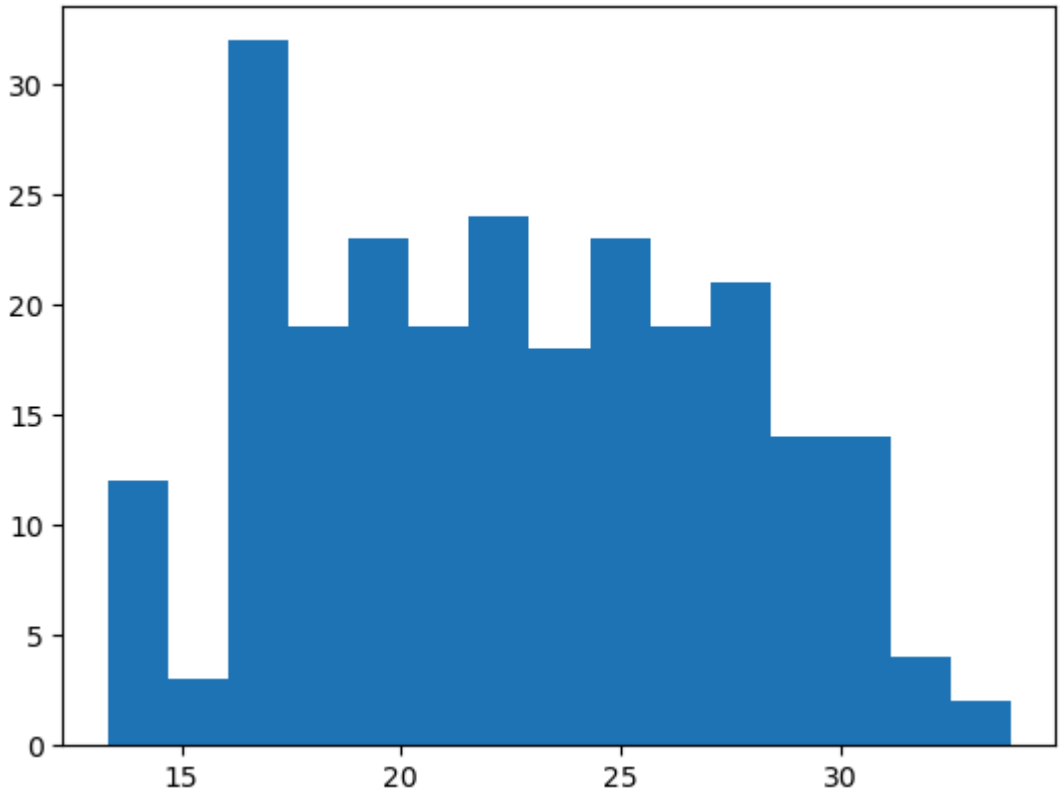
In [6]:
```
#Codegrade Tag Question6
# Write all your code for this answer in this cell
for i in range(0,len(df_California_summer['Taverage'])-1):
 df_California_summer['Taverage']    =  [convert_f_to_c(pd.to_numeric(df['Tavera
```

Question 7: Plotting histogram and calculation of mean value and standard deviation

Plot histogram of the distribution of the temperature (variable `df_California_summer['Taverage']`) in California during summer time using Matplotlib in th next code cell. Calculate mean value and standard deviation of the temperature and report them in one vector (array) called `California_summer_temp_vector` using `np.array` in the second cell. The first component of this vector should represent mean value and standard deviation of the temperature respectively.

In [7]:
```
#Codegrade Tag Question7
# Write all your code for this answer in this cel
#df_California_summer['Taverage']
plt.clf()
plt.hist(df_California_summer['Taverage'],bins=15)
```

Out[7]:
```
(array([12.,  3., 32., 19., 23., 19., 24., 18., 23., 19., 21., 14., 14.,
         4.,  2.]),
 array([13.33333333, 14.7037037 , 16.07407407, 17.44444444, 18.81481481,
        20.18518519, 21.55555556, 22.92592593, 24.2962963 , 25.66666667,
        27.03703704, 28.40740741, 29.77777778, 31.14814815, 32.51851852,
        33.88888889]),
 <BarContainer object of 15 artists>)
```



In [8]:
```
California_mean = np.average(df_California_summer['Taverage'])
California_std = np.std(df_California_summer['Taverage'])
California_summer_temp_vector = np.array([California_mean , California_std])
California_summer_temp_vector
```

Out[8]:
```
array([22.60683761,  4.87774967])
```

In [ ]:
```

```

Programming Assignment 3: Predictions of Beer Production with ARIMA Model

ALEXANDER STOTSKY
Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg SE - 412 96
SWEDEN
alexander.stotsky2@telia.com

Description

Accurate prediction of future demand of products allows optimization of the factory capacity, supply and storage, personnel and seasonal hiring which reduces lead times, minimizes costs, boosts customer satisfaction and finally increases sales performance. In this assignment you will anticipate beer production in Austria using monthly measurements.
Time series of beer production measured in million liter from January 1956 untill August 1995 has low frequency component associated with annual growth and seasonal high frequency wave form component which reaches maximum values during winter and minimum values in summer time each year. For estimation of low frequency trend you will use decomposition method which separates trend and seasonality.
You will use the Auto Regressive (AR) Integrated (I) Moving Average (MA), ARIMA model for predictions of high frequency oscillations in beer production. The ARIMA model is extenstion of well-known ARMA model with an integration component, which compensates nonstationary trend via differencing. The ARIMA Model in Python can be presented in the following form: ARIMA(p,d,q), where p and q are the orders of AR and MA parts respectively and d in the number of differences of the time series. Notice that d = 0 in this assignment since nonstationary low frequency trend will be removed. The ARMA model for estimation of the production output y_k can be written in the form:

$$y_k = c + \sum_{i=1}^{p} \phi_i \, y_{k-i} + \sum_{j=1}^{q} \theta_j \, \varepsilon_{k-j} + \varepsilon_k \tag{1}$$

and can be seen as ARIMA(p,0,q) model with two unknown orders p and q and corresponding coefficients ϕ_i and θ_j, where ε_k is the noise and k is the step number. ARMA model is well suited for stationary time series (where constant c corresponds to offset) and can be applied after subtraction of the low frequency trend. You will use goodness of fit statistical test for evaluation of the estimation performance.
The assignment should be accomplished (as usual) as the answers to the questions submitted in the form of the Python code in Jupyter notebook. Notebook template, which contains questions to be answered, please find in the file notebook template . The codegrade converts automatically your Jupyter notebook to executable Python code.
Please read attentively the questions in the template which include scaffolding and recommendations associated with the names of variables and functions. Please follow these recommendations which are needed for automatic evaluation and grading of your code.
Explanatory video for this assignment can be found in the video link.

Programming Assignment 3: Predictions of beer production with ARIMA model

The assignment is associated with anticipation of beer production in Austria using monthly measurements. Time series of beer production measured in million liter from January 1956 untill August 1995 has low frequency component associated with annual growth and seasonal high frequency wave form component which reaches maximum values during winter and minimum values in summer time each year. For estimation of low frequency trend you will use `seasonal_decompose` method imported from the library `statsmodels.tsa.seasonal` which separates trend and seasonality.
You will use the Auto Regressive (AR) Integrated (I) Moving Average (MA), ARIMA model for predictions of high frequency oscillations in beer production. The ARIMA model is extenstion of well-known ARMA model with an integration component, which compensates nonstationary trend via differencing. The ARIMA Model in Python can be presented in the following form: ARIMA(p,d,q), where p and q are the orders of AR and MA parts respectively and d in the number of differences of the time series. Notice that d = 0 in this assignment since nonstationary low frequency trend estimated via `seasonal_decompose` method will be removed. The ARMA model for high frequency component can be written in the form: $y_k = c + \sum_{i=1}^{p} \phi_i y_{k-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{k-j} + \varepsilon_k$, where y_k is an estimate of the production yield and ε_k is the noise. The model can be seen as ARIMA(p,0,q) model with two unknown orders p and q and corresponding coefficients ϕ_i and θ_j, where k is the step number. ARMA model is well suited for stationary time series (where constant c corresponds to offset) and can be applied after subtraction of the low frequency trend.

Question 1: Import libs and read the file with beer production

Import standard libraries using the commands presented in this cell. Read the file `data.csv` and save the result in a variable called `df`. Examine your input file by entering command `df`. In addition and for further convenience you can ignore warnings by importing lib `warnings` and entering command `warnings.filterwarnings("ignore")`

In [1]:
```
# Codegrade Tag Question1
# Write all your code for this answer in this cell
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima.model import ARIMA
import scipy.stats as stats
#from scikit-learn import mean_squared_error
# from sklearn.metrics import mean_squared_error
```

```
from statsmodels.graphics.gofplots import qqplot
from scipy.stats import shapiro
from matplotlib import pyplot
import warnings
warnings.filterwarnings("ignore")

df = pd.read_csv('data.csv')
df
```

Out[1]:

	Month	Monthly beer production
0	1956-01	93.2
1	1956-02	96.0
2	1956-03	95.2
3	1956-04	77.1
4	1956-05	70.9
...	...	...
471	1995-04	127.0
472	1995-05	151.0
473	1995-06	130.0
474	1995-07	119.0
475	1995-08	153.0

476 rows × 2 columns

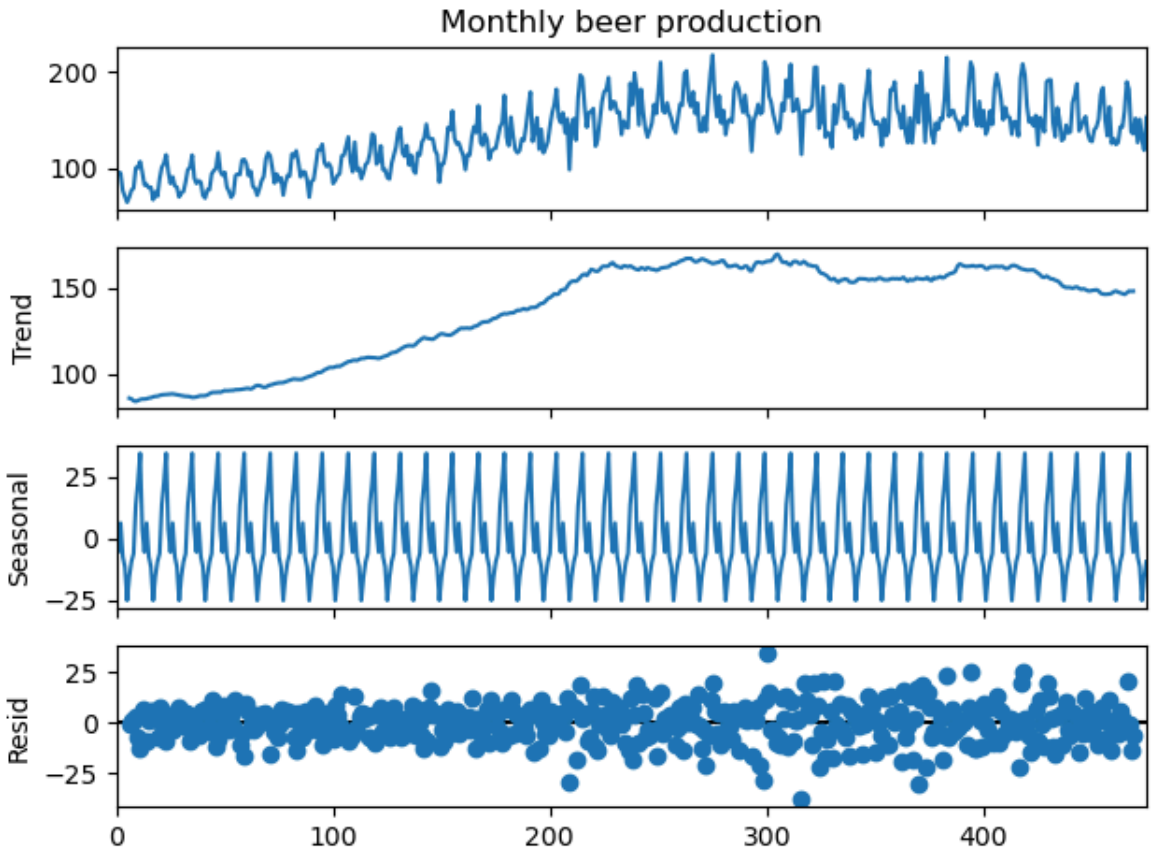
Question 2: Decomposition of time series and correction of low frequency trend

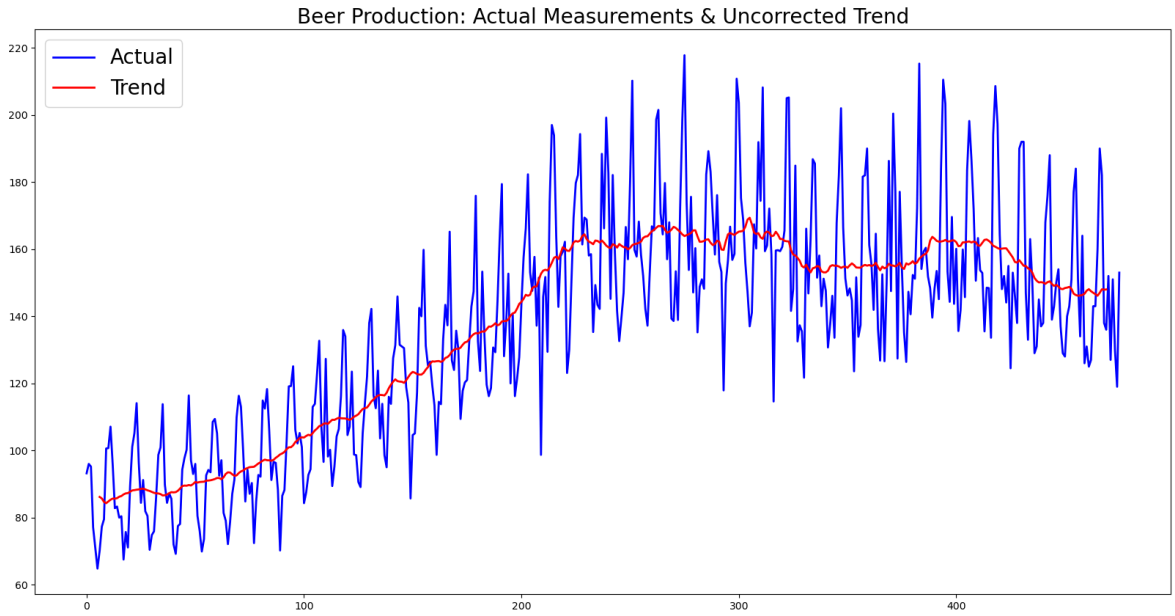
Apply `seasonal_decompose` function for the variable `decomposition = seasonal_decompose(y, model='additive', period = 12)` which corresponds to one year seasonal component (variable `y` is defined below). Plot your decomposition using attribute plot.
Plot times series in Cartesian coordinates where `x = np.arange(0,df.shape[0])` and `y = df['Monthly beer production']` together with low frequency trend obtained from `decomposition.trend` . Create and examine variable `trend = decomposition.trend` . This variable contains NaN components, which are visible on the plot. Remove these components using interpolation function as follows: `nan_index = np.isnan(trend)` `trend[nan_index] = np.interp(np.flatnonzero(nan_index), np.flatnonzero(~nan_index), trend[~nan_index])` . Examine once more the variable `trend` to be sure that all NaN components have been removed. Plot the corrected low frequency `trend` and `x,y` variables. Compare the plots before and after correction.

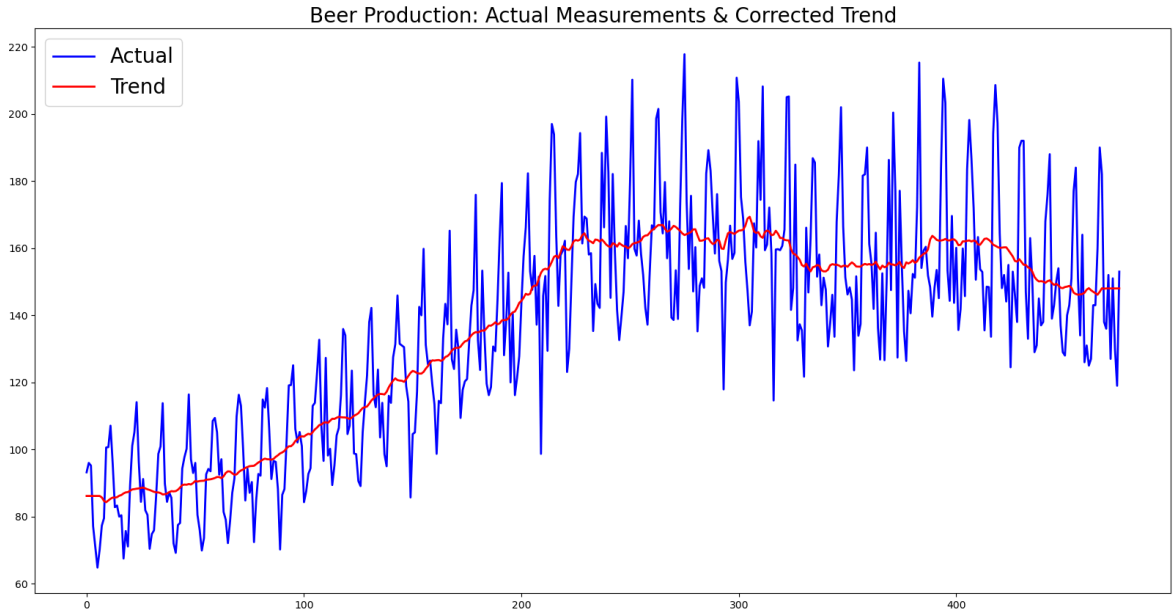
In [2]:
```
# Codegrade Tag Question2
# Write all your code for this answer in this cell
```

```
y = df['Monthly beer production']
x = np.arange(0,df.shape[0] , 1)
decomposition = seasonal_decompose(y, model='additive', period = 12)
decomposition.plot()
fig = plt.figure(figsize=(20, 10))
plt.title('Beer Production: Actual Measurements & Uncorrected Trend', fontsize=2
plt.plot(x,y,color='b',linewidth = 2,label='Actual')
plt.plot(x,decomposition.trend,color='r',linewidth = 2,label='Trend')
plt.legend(fontsize =20, loc='upper left')
trend = decomposition.trend
trend
nan_index = np.isnan(trend)
trend[nan_index] = np.interp(np.flatnonzero(nan_index), np.flatnonzero(~nan_inde
trend
fig = plt.figure(figsize=(20, 10))
plt.title('Beer Production: Actual Measurements & Corrected Trend', fontsize=20)
plt.plot(x,y,color='b',linewidth = 2,label='Actual')
plt.plot(x,trend,color='r',linewidth = 2,label='Trend')
plt.legend(fontsize =20, loc='upper left')
```

Out[2]: <matplotlib.legend.Legend at 0x26af264b740>





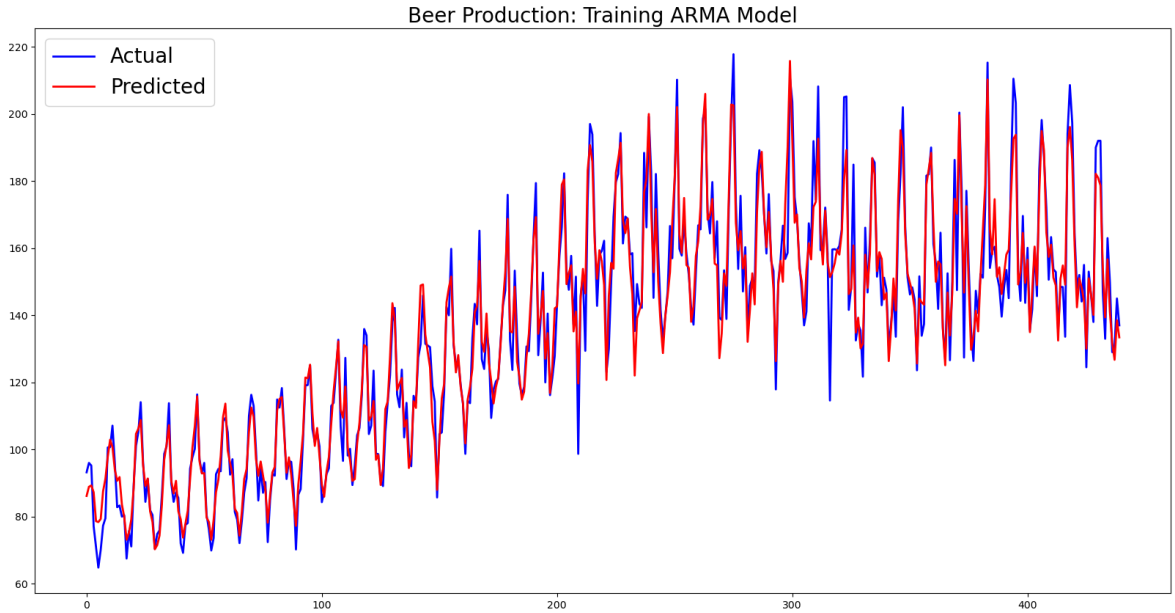


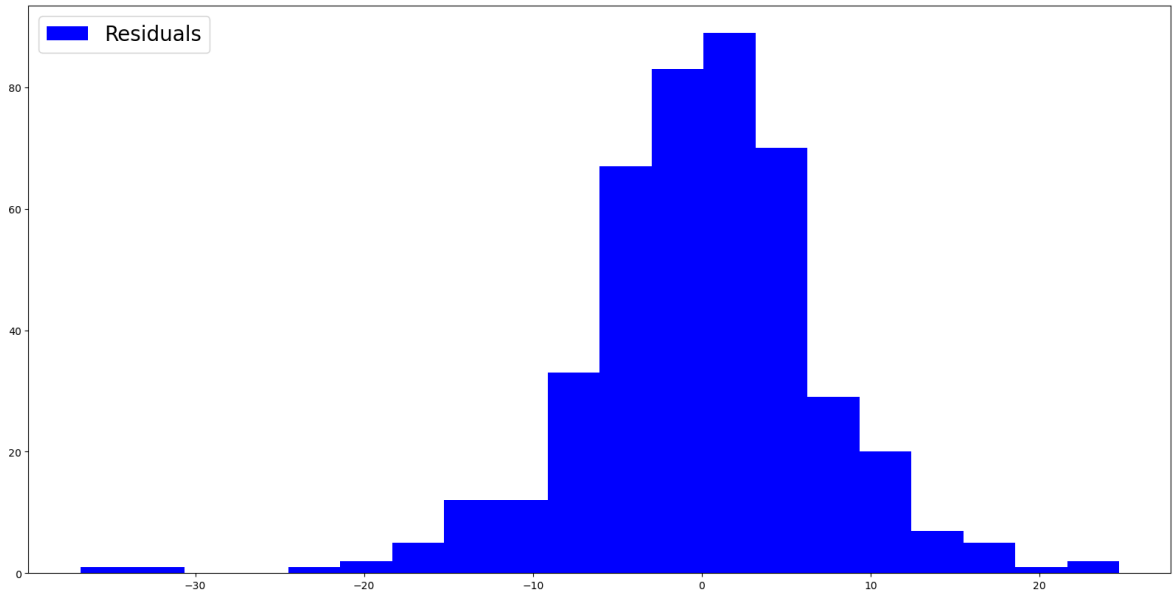
Question 3: Separation of the time series in training and prediction set and fitting ARIMA model

For training of ARMA model define the variable `training_data` as the difference between variables `y` and `trend` for `x[:ts]`, where `ts = 440` is the time point which separates training and test data. Therefore the test sequence is defined for `x[ts:]`. Fit ARMA model as `ARMA_model = ARIMA(training_data, order=(12, 0, 17)).fit()` with recommended AR order, `p = 12` and recommended MA order, `q = 17` where the `training_data` is the periodic high frequency training sequence. Use attribute `predict` associated to `ARMA_model` and create variable `predictions`. Plot measured time series and predictions with the added trend in one plot.
Calculate mean squared error between `training_data` and `predictions` from the ARMA model using function `mean_squared_error` imported from `sklearn.metrics`. Finally, plot histogram of the residual error between `training_data` and `predictions`. Residuals should be normally distributed.

In [3]:
```
# Codegrade Tag Question3
# Write all your code for this answer in this cell
ts = 440
xs = x[:ts]
training_data = y.loc[:ts-1] - trend[:ts]
ARMA_model = ARIMA(training_data, order=(12, 0, 17)).fit()
predictions = ARMA_model.predict()
fig = plt.figure(figsize=(20, 10))
plt.title('Beer Production: Training ARMA Model', fontsize=20)
plt.plot(xs,y.loc[:ts-1],color='b',linewidth = 2,label='Actual')
plt.plot(xs,predictions +  trend[:ts],color='r',linewidth = 2,label='Predicted')
plt.legend(fontsize =20, loc='upper left')
residual = training_data - predictions
fig = plt.figure(figsize=(20, 10))
plt.hist(residual,bins = 20,color='b',linewidth = 2,label='Residuals')
plt.legend(fontsize =20, loc='upper left')
np.square(trend-y).mean()
```

Out[3]: 342.67317536910605

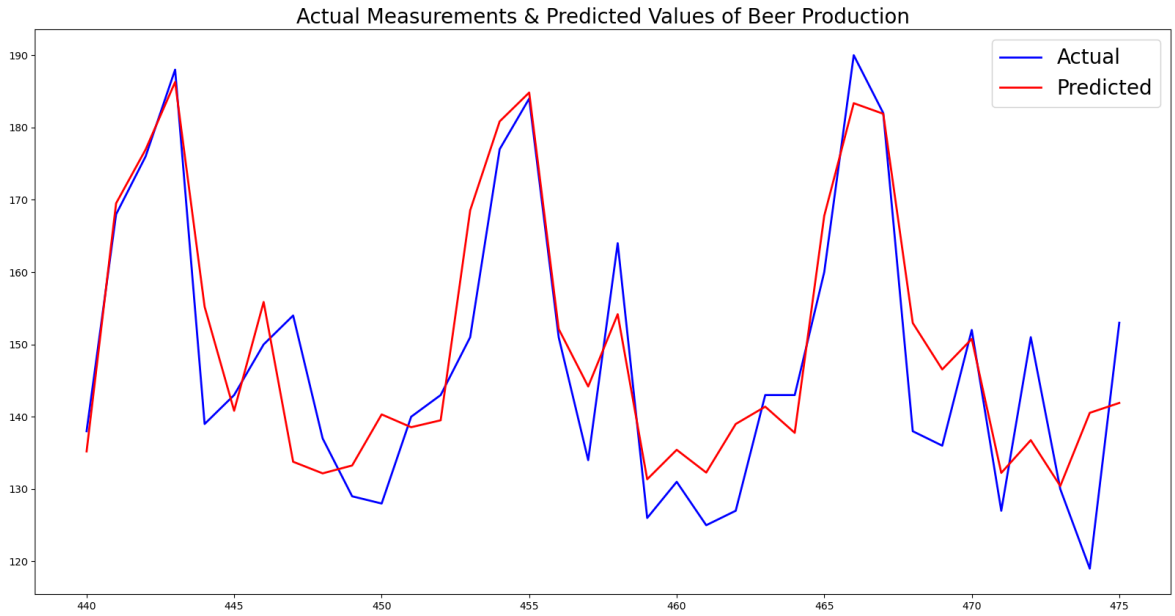


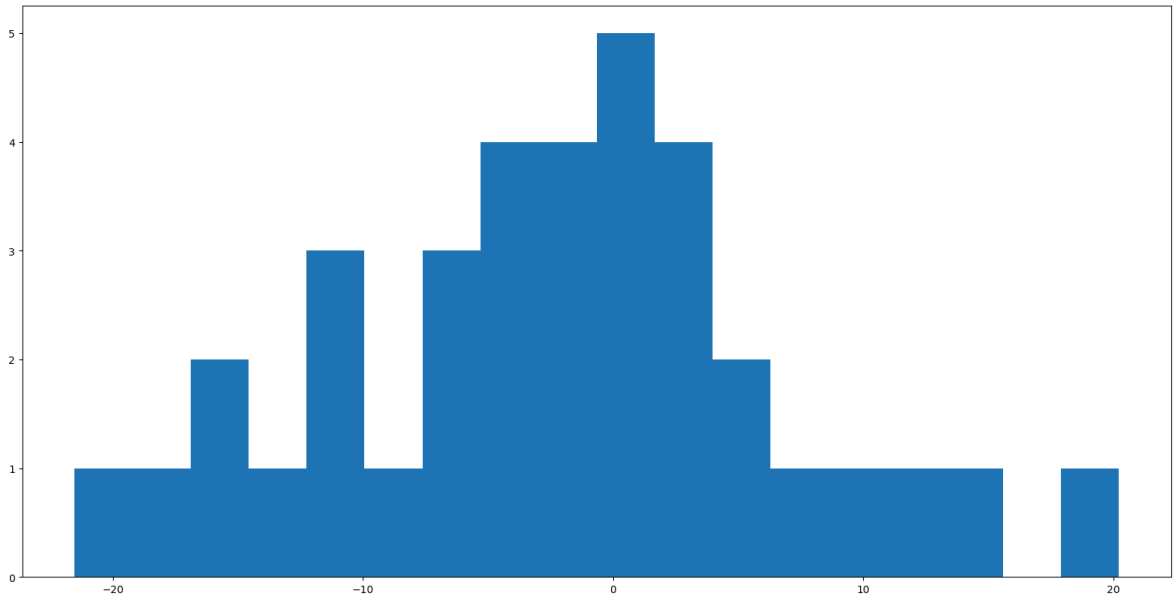


Question 4: Prediction of beer production and normality test for prediction error

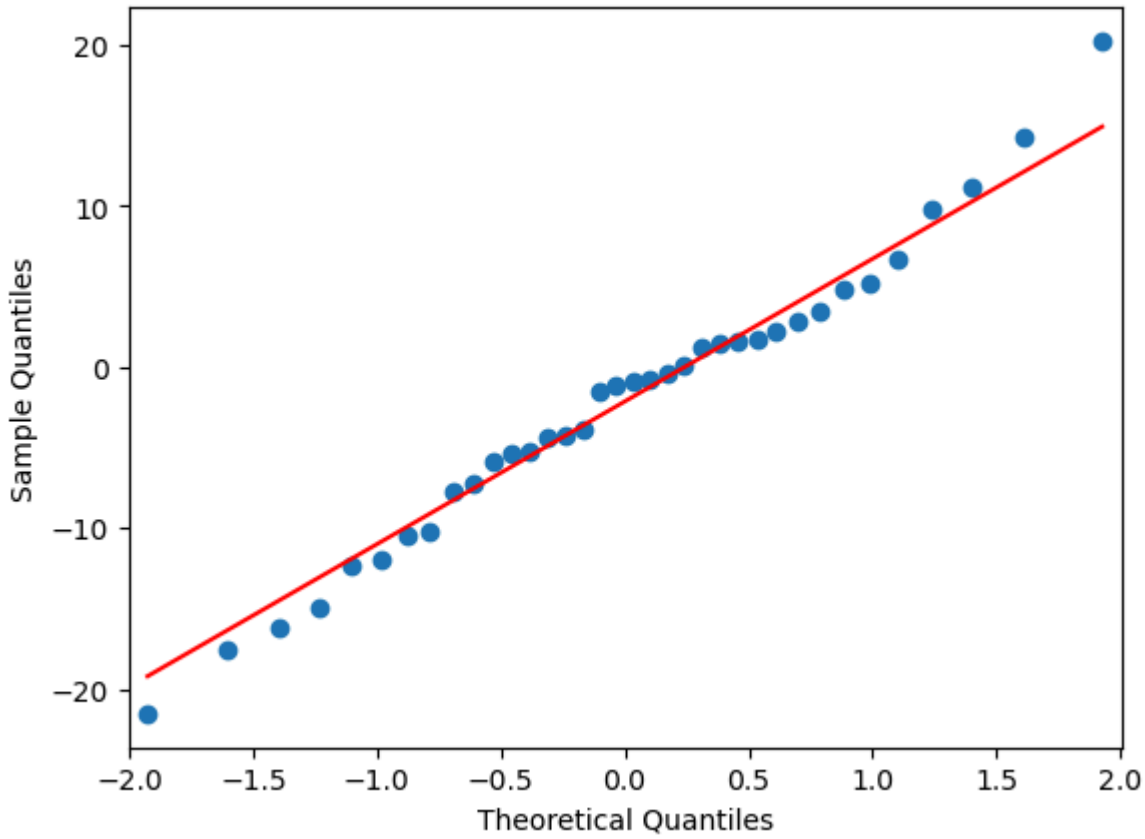
Predict time series for the test sequence using ARMA model and the nearest (to the test sequence) value of the trend using model as follows: `test_predictions = ARMA_model.predict(start=ts, end=475) + trend[ts-1]`. Plot measured test series for `x[ts:]` and `test_predictions` in the same plot. Plot histogram of the residual error between measured time series and `test_predictions` for `x[ts:]`. Residuals should be normally distributed. Departures from normality can be identified using `qqplot` improted from the library `statsmodels.graphics.gofplots`. The residuals should approximately follow the straight line on the `qqplot` which indicates the goodness of fit, if the residuals are normally distributed. Plot `qqplot` and examine the result. Quanitification of the goodness of fit is associated with Shapiro–Wilk test, which tests the null hypothesis that the data is drawn from a normal distribution. The test is recommended for small sample sizes. For large samples the Kolmogorov-Smirnov test is recommended for goodness of fit. Create two output variables `stat_sh` and `p_sh` for statistic and p-value of the function `shapiro` for residuals associated with Shapiro–Wilk test. The hypothesis that the residuals are normally distributed is taken as the null hypothesis, which is tested against the alternative hypothesis that the residuals are not normally distributed indicating inaccurate prediction. Choose the significance level of `5 %` (five percent) and decide based on the output of the function `shapiro` to reject the null hypothesis in favor of the alternative hypothesis or not. Print `Sample is Gaussian (fail to reject H0)` or `Sample is not Gaussian (reject H0)` depending on the test result.

In [4]:
```
# Codegrade Tag Question4
# Write all your code for this answer in this cell
test_predictions = ARMA_model.predict(start=ts, end=475) + trend[ts-1]
fig = plt.figure(figsize=(20, 10))
plt.title('Actual Measurements & Predicted Values of Beer Production', fontsize=
plt.plot(x[ts:],y.loc[ts:],color='b',linewidth = 2,label='Actual')
plt.plot(x[ts:],test_predictions,color='r',linewidth = 2,label='Predicted')
plt.legend(fontsize =20, loc='upper right')
fig = plt.figure(figsize=(20, 10))
plt.hist(y.loc[ts:] - test_predictions ,bins = 18)
fig = plt.figure(figsize=(20, 10))
qqplot(y.loc[ts:] - test_predictions, line='s')
pyplot.show()
stat_sh, p_sh = shapiro(y.loc[ts:] - test_predictions)
alpha = 0.05
if p_sh > alpha:
        print('Sample is Gaussian (fail to reject H0)')
else:
        print('Sample is not  Gaussian (reject H0)')
```





<Figure size 2000x1000 with 0 Axes>




```
Sample is Gaussian (fail to reject H0)
```

In [ ]:

Programming Assignment 4: Short Term Forecasting with Linear Regression Model

ALEXANDER STOTSKY
Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg SE - 412 96
SWEDEN
alexander.stotsky2@telia.com

Description

Short-term forecasting helps organizations in optimization of day-to-day operations, efficiency improvement and satisfaction of the customer demands. Short-term forecasts predict future events (revenues for example) within the weeks via application of the regression analysis. The forecasting performance is based on the prediction accuracy associated with regression analysis.
In this assignment you will predict total revenue of the company for the next 17 days of operation and compare the result with actual sales result for the same period. Two datasets are provided in the links. The first dataset consists of historical data of daily revenues in USD (United States Dollars) of the company as a function of the temperature in degrees celsius. You will design linear regression with three different methods based on historical data from this dataset. In the notebook you will also find relevant reference, provided for deeper understanding of linear regression analysis.
The manager of the company gets the temperature forecast for the next 17 days. Using the temperature data you will make the prediction of the total revenue of the company for these days using linear regression model. Actual revenues for these days are provided in the second file for comparison and evaluation of the error of short term forecasting. You will see that short term forecasting which helps in optimization of operations of the companies can be very accurate with linear regression model.
The assignment should be accomplished (as usual) as the answers to the questions submitted in the form of the Python code in Jupyter notebook. Notebook template, which contains questions to be answered, please find in the file notebook template. The codegrade converts automatically your Jupyter notebook to executable Python code.
Please read attentively the questions in the template which include scaffolding and recommendations associated with the names of variables and functions. Please follow these recommendations which are needed for automatic evaluation and grading of your code. The questions include even the references to the relevant literature.
Good luck !

Figure 1: Icecream

Programming Assignment 4: Short Term Forecasting with Linear Regression Model

In this assignment you will predict total revenue of the company for the next 17 days and compare the result with actual sales data for the same period. Two datasets are provided. The first dataset consists of historical data of daily revenues in USD (United States Dollars) of the company as a function of the temperature in degrees celsius. Ice cream sales tend to increase as the temperature outside rises. The temperature is presented as independent variable and the revenues are associated with dependent variable. You will design linear regression with three different methods based on historical data. Suppose that the manager of the company gets the temperature forecast for the next 17 days. Using the temperature data you will make the prediction of the total revenue of the company for these days using linear regression model. Actual revenues for these days are provided for comparison and evaluation of the error of short term forecasting.

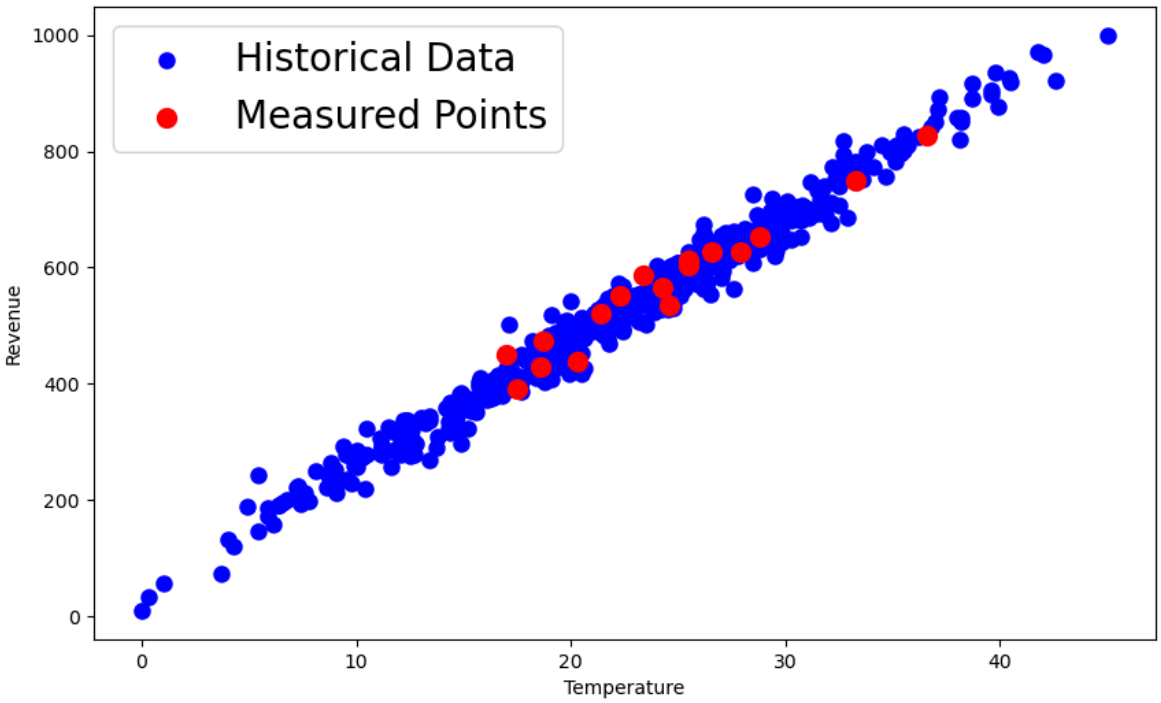
Question 1: Import libs and read the files with sales data

Import standard libraries using the commands presented in this cell. Read the file `Icecream1.csv` which consists of historical data of the revenues of the company as a function of the temperature and save the result as dataframe `df`. Icecream sales (variable `df['Revenue']`) tend to increase as the temperature `df['Temperature']` outside rises. Examine your input file by entering command `df`. Read the file `Icecream2.csv` ansd save the result in dataframe `df_measured`, where the temperature forecast for the next 17 days you find in variable `df_measured['Temperature']`. Actual revenues for these days you can find in the variable `df_measured['Revenue']` for comparison and evaluation of the error of short term forecasting. For later convenience introduce the following variables `x = df['Temperature']`, `y = df['Revenue']` and `xm = df_measured['Temperature']`, `ym = df_measured['Revenue']`. Plot `x, y` and `xm, ym` with different colors on the same scatter plot. In addition and for further convenience you can ignore warnings by importing lib `warnings` and entering command `warnings.filterwarnings("ignore").

In [2]:
```
# Codegrade Tag Question1
# Write all your code for this answer in this cell
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as stats
from scipy.optimize import curve_fit
#from statsmodels.graphics.gofplots import qqplot
#from matplotlib import pyplot
import warnings
warnings.filterwarnings("ignore")
```

```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

df = pd.read_csv("Icecream1.csv")
df_measured = pd.read_csv("Icecream2.csv")
x = df['Temperature']
y = df['Revenue']
xm = df_measured['Temperature']
ym = df_measured['Revenue']
fig = plt.figure(figsize=(10, 6))
plt.scatter(x,y,color='b',label='Historical Data',linewidth = 3)
plt.scatter(xm,ym, color='r',label='Measured Points',linewidth = 5)
plt.legend(fontsize =20, loc='upper left')
plt.ylabel("Revenue")
plt.xlabel("Temperature")
plt.show()
```



Question 2: Design linear regression using different techniques

You will apply three methods for estimation of the revenue $\hat{y}$ as linear function of `x` for dataset `df` as follows: $\hat{y}_i = \alpha_i + \beta_i\, x$, where α_i and β_i are regression coefficients. Define the coefficients as `alpha_i` and `beta_i` in the program, $i = 0, . . , 2$.

Method 1. Calculate Pearson correlation coefficient (and define it as `r_0`) associated with correlation of `x` and `y` variables using `scipy.stats` library. Calculate the regression coefficients as follows: $\beta_0 = r_0 \frac{s_y}{s_x}$ and $\alpha_0 = \overline{y} - \beta_0\, \overline{x}$, where s_y, s_x and $\overline{y}$, $\overline{x}$ are standard deviations and mean values of y and x variables respectively.

Method 2. Calculate the slope as follows: $\beta_1 = \dfrac{\sum_{j=1}^{n} x_j y_j - n \overline{x}\,\overline{y}}{\sum_{j=1}^{n} x_j^2 - n \overline{x}^2}$, where n is the sample size. The offset $\alpha_0 = \overline{y} - \beta_1 \overline{x}$ is calculated similar to method 1, see above.

Method 3. Apply least squares fitting by using function `curve_fit()` imported from `scipy.optimize`. The function `curve_fit()` requires specification of the linear function to be optimized. The form of this function `linfunc` is defined in the next cell. Save the slope in variable `beta_2` and the offset in `alpha_2`. Examine the covariance matrix calculated using function `curve_fit()`. What shows covariance matrix in this case ? The answer you will be able to find in the book by Sheldon Ross, `"Introduction to Probability and Statistics for Engineers and Scientists"`, see section 9.10 on pages 394-402.

Compare estimated coefficients `alpha_i` and `beta_i`, $i = 0, . . , 2$ obtained via three methods. The coefficients should be the same. If the coefficients are not the same, find the errors and perform recalculations.

Plot measured points from the dataset `df` as a scatter plot and regression line `hy = alpha_0 + beta_0 x` (which is your model) on the same plot. Plot histogram of the residuals for verification of the goodness of the fit. The residuals should be approximately normally distributed.

In [3]:
```
# Codegrade Tag Question2
# Write all your code for this answer in this cell
def linfunc(x, beta_2, alpha_2):
      y = beta_2 * x + alpha_2
      return y
# method 1
r_0 = stats.pearsonr(x, y).statistic
s_x = np.std(x)
s_y = np.std(y)
beta_0 = r_0 * s_y / s_x
alpha_0 = np.mean(y) - beta_0 * np.mean(x)
# method 2
beta_1   = (sum(x*y) - len(y) * np.mean(x)*np.mean(y)) / (sum(x*x) - len(x)*(np.
alpha_1 = np.mean(y) - beta_1 * np.mean(x)
#method 3
(beta_2, alpha_2),covariance = curve_fit(linfunc, xdata = x, ydata = y)
beta_0, alpha_0
beta_1, alpha_1
beta_2, alpha_2
hy = alpha_0 + beta_0 * x

fig = plt.figure(figsize=(10, 6))
plt.scatter(x,y,color='b',label='Historical Data',linewidth = 3)
plt.plot(x,hy,color='r',label='Linear Regression',linewidth = 5)
plt.legend(fontsize =20, loc='upper left')
plt.ylabel("Revenue")
plt.xlabel("Temperature")
plt.show()
fig = plt.figure(figsize=(20, 10))
```

```
plt.hist(y - hy,bins = 40)
plt.ylabel("Frequency")
plt.xlabel("Residual Error")
```

```
Covariance [[ 0.01937453 -0.43035123]
 [-0.43035123 10.85009772]]
```

Out[3]: (21.44697629199806, 44.0789127356083)

Out[3]: (21.44697629199809, 44.07891273560756)

Out[3]: (21.446976285357458, 44.07891290494358)

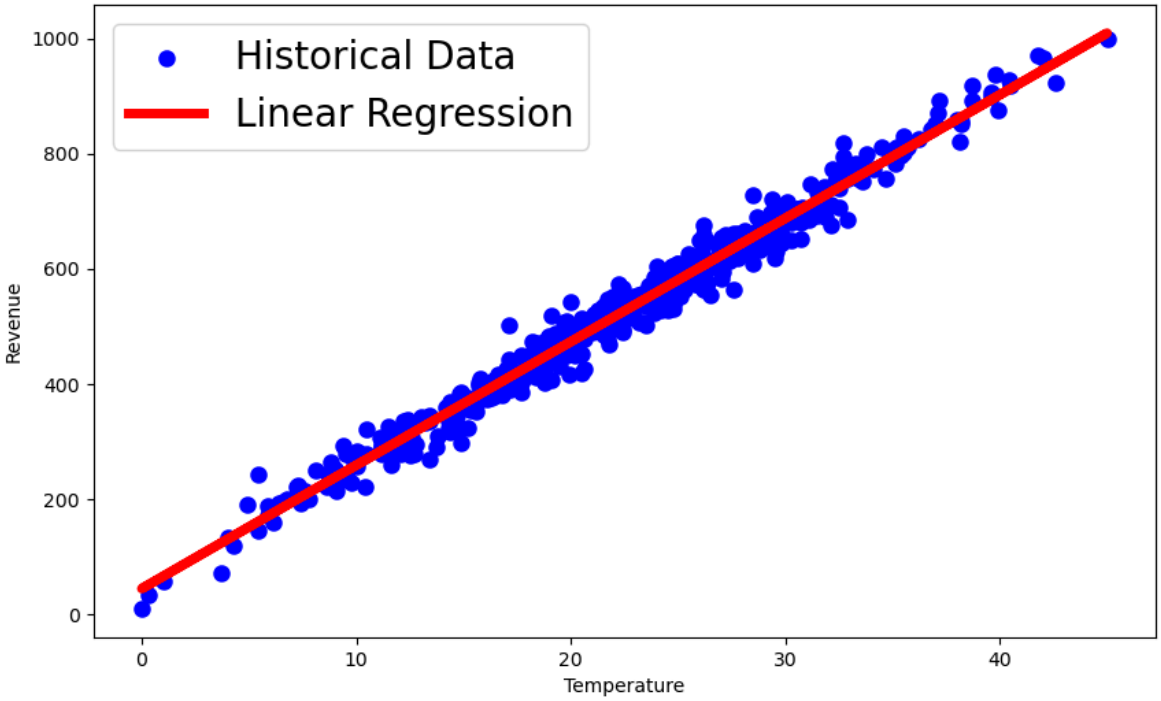
Out[3]: <matplotlib.collections.PathCollection at 0x288774ba600>

Out[3]: [<matplotlib.lines.Line2D at 0x28878188d70>]

Out[3]: <matplotlib.legend.Legend at 0x2887746cbc0>

Out[3]: Text(0, 0.5, 'Revenue')

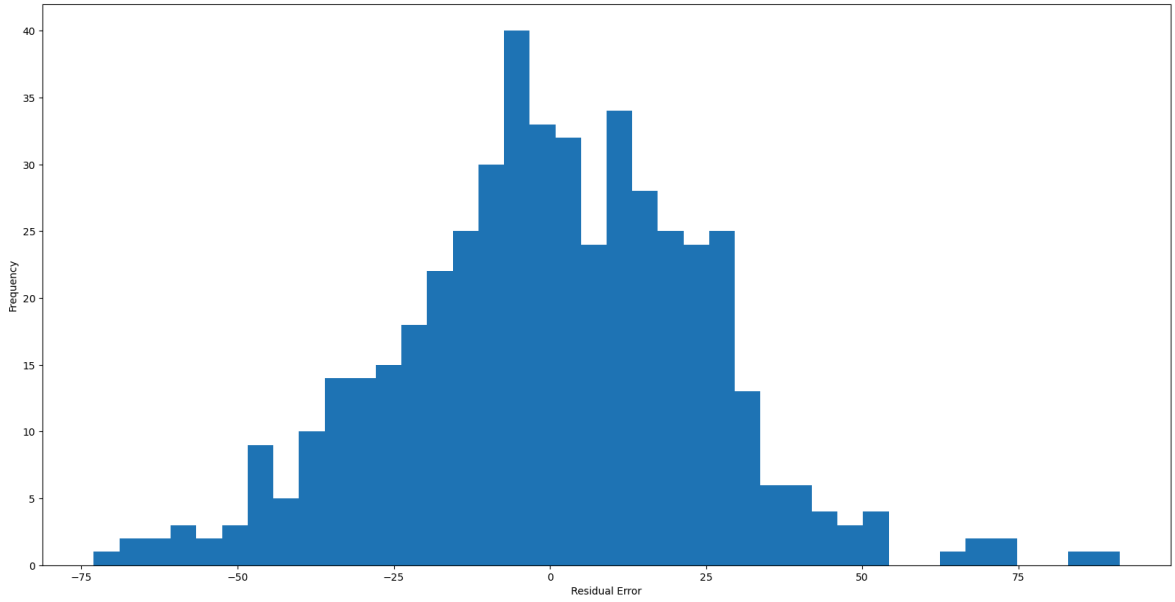
Out[3]: Text(0.5, 0, 'Temperature')



Out[3]:
```
(array([ 1.,  2.,  2.,  3.,  2.,  3.,  9.,  5., 10., 14., 14., 15., 18.,
        22., 25., 30., 40., 33., 32., 24., 34., 28., 25., 24., 25., 13.,
         6.,  6.,  4.,  3.,  4.,  0.,  0.,  1.,  2.,  2.,  0.,  0.,  1.,
         1.]),
 array([-73.01545839, -68.91062712, -64.80579584, -60.70096456,
        -56.59613329, -52.49130201, -48.38647073, -44.28163946,
        -40.17680818, -36.0719769 , -31.96714563, -27.86231435,
        -23.75748307, -19.6526518 , -15.54782052, -11.44298925,
         -7.33815797,  -3.23332669,   0.87150458,   4.97633586,
          9.08116714,  13.18599841,  17.29082969,  21.39566097,
         25.50049224,  29.60532352,  33.7101548 ,  37.81498607,
         41.91981735,  46.02464863,  50.1294799 ,  54.23431118,
         58.33914246,  62.44397373,  66.54880501,  70.65363629,
         74.75846756,  78.86329884,  82.96813012,  87.07296139,
         91.17779267]),
 <BarContainer object of 40 artists>)
```

Out[3]: Text(0, 0.5, 'Frequency')

Out[3]: Text(0.5, 0, 'Residual Error')



Question 3: Short term forecast of the revenues

Suppose that the manager of the company gets the temperature forecast for the next 17 days in variable `xm`. Calculate revenues for these days using linear regression model (save the result in vector `yf`) and plot measured `ym` and predicted revenues `yf` on the same plot. The manager wants to know how much money will be available by the end of this period `Sf` = $\sum_{i=0}^{16} y_{fi}$. During this period the manager gets the data of actual revenues, `Sa` = $\sum_{i=0}^{16} y_{mi}$. Compare actual and predicted values of revenues and calculate `Sd` = `Sa` - `Sf`. Calculate total forecast error in percent as follows: `error` = $\frac{\sum_{i=0}^{16} y_{mi} - y_{fi}}{\sum_{i=0}^{16} y_{mi}}$ 100 %.

In [4]:
```
# Codegrade Tag Question3
# Write all your code for this answer in this cell
yf =  alpha_0 + beta_0 * xm
fig = plt.figure(figsize=(20, 10))
plt.plot(xm.index, yf, color='r',label='Forecast',linewidth = 5)
plt.plot(xm.index,ym, color='b',label='Measured',linewidth = 5)
plt.legend(fontsize =20, loc='upper right')
plt.xlabel("Days")
plt.ylabel("Revenues")
plt.show()
Sf = yf.sum()
Sa = ym.sum()
Sd = Sa - Sf
error = Sd / Sa * 100
```

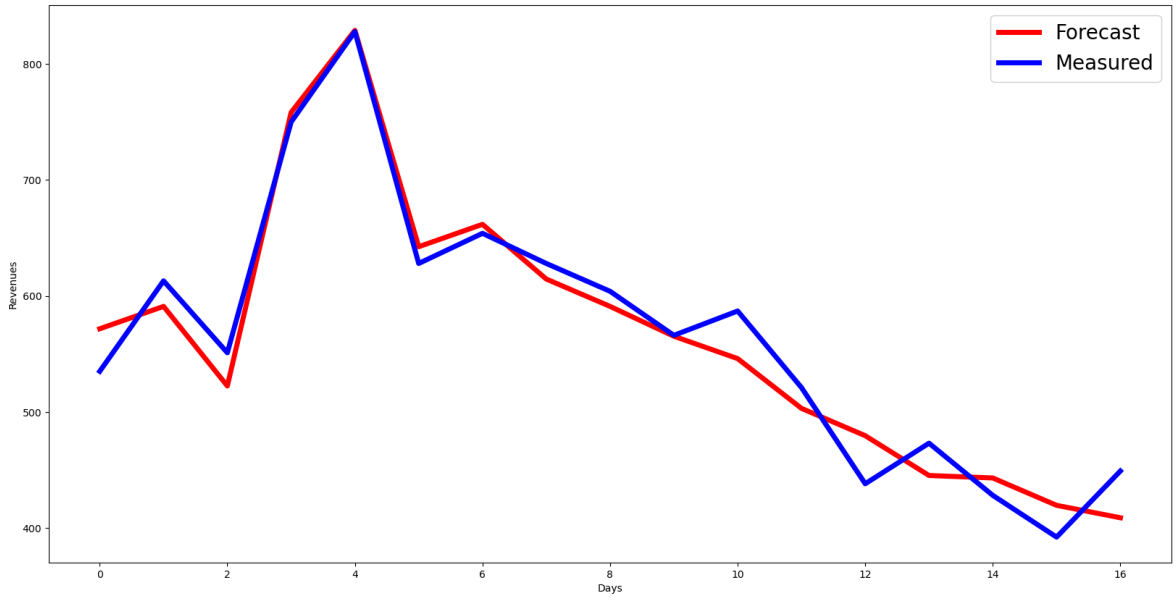
Out[4]: [<matplotlib.lines.Line2D at 0x288781e4170>]

Out[4]: [<matplotlib.lines.Line2D at 0x2887800bcb0>]

Out[4]: <matplotlib.legend.Legend at 0x288781ff560>

Out[4]: Text(0.5, 0, 'Days')

Out[4]: Text(0, 0.5, 'Revenues')



In [ ]: