Carlos J. Costa

REGRESSION

Learning Goals

- Students should use the main libraries to create and fit regression model.
- Students should use the main libraries to analyse regression model.

statsmodels

- Python module
- provides classes and functions
- estimation statistical models,
- statistical tests,
- statistical data exploration.
- open source Modified BSD (3-clause) license.
- https://www.statsmodels.org/

- Regression and Linear Models
- Time Series Analysis
- Other Models (e.g. Non parametric models)
- Statistics and Tools
- Data Sets

statsmodels

- **statsmodels.api:**
  - Cross-sectional models and methods.
  - Canonically imported using

```
–import statsmodels.api as sm
```

- **statsmodels.tsa.api:**
  - Time-series models and methods.
  - Canonically imported using:

```
–import statsmodels.tsa.api as tsa
```

- **statsmodels.formula.api:**
  - A convenience interface for specifying models using formula strings and DataFrames.
  - Canonically imported using:

```
–import statsmodels.formula.api as smf
```

The main statsmodels API is split into models:

Regression and Linear Models

- Linear Regression
- Generalized Linear Models
- Generalized Estimating Equations
- Generalized Additive Models (GAM)
- Robust Linear Models
- Linear Mixed Effects Models
- Regression with Discrete Dependent Variable
- Generalized Linear Mixed Effects Models
- ANOVA

Linear Regression

- Linear models with independently and identically distributed errors, and for errors with heteroscedasticity or autocorrelation.
- This module allows estimation by:
  - ordinary least squares (OLS),
  - weighted least squares (WLS),
  - generalized least squares (GLS), and
  - ..

Regression and Linear Models

- OLS Assumptions

1. Linearity
2. No Multicollinearity
3. errors are normally distributed
4. Non Autocorrelation (Autocorrelation - correlation between the error values)
5. Homoscedasticity (variance of the error terms should be constant with respect to the independent variable)

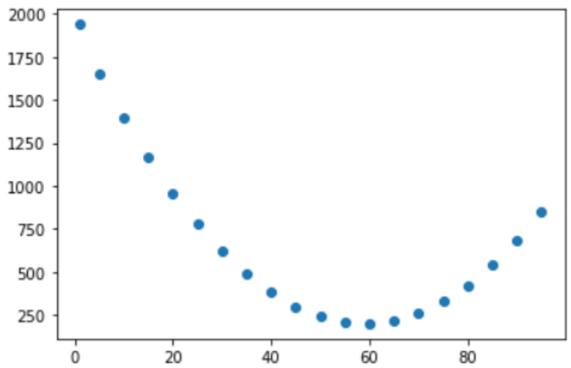
https://aiaspirant.com/ols-assumptions

A Data Set

```python
import matplotlib.pyplot as plt
```

```python
x=[1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95]
y=[1940,1650,1400,1170,960,780,620,490,380,300,240,210,200,220,260,330,420,540,680,850]
```

```python
plt.plot(x,y,'o')
plt.draw()
```



- What is the best model to explain

  - `Y=f(X)` ?

- A linear Model?

Create and Fit

```
import statsmodels.api as sm
x1 = sm.add_constant(x)
model=sm.OLS(y, x1)
result=model.fit()
z=result.predict(x1)
result.summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.441
Model:	OLS	**Adj. R-squared:**	0.410
Method:	Least Squares	**F-statistic:**	14.21
Date:	Wed, 03 Mar 2021	**Prob (F-statistic):**	0.00140
Time:	14:32:02	**Log-Likelihood:**	-146.69
No. Observations:	20	**AIC:**	297.4
Df Residuals:	18	**BIC:**	299.4
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1226.9153	168.921	7.263	0.000	872.025	1581.805
x1	-11.4598	3.040	-3.770	0.001	-17.847	-5.073

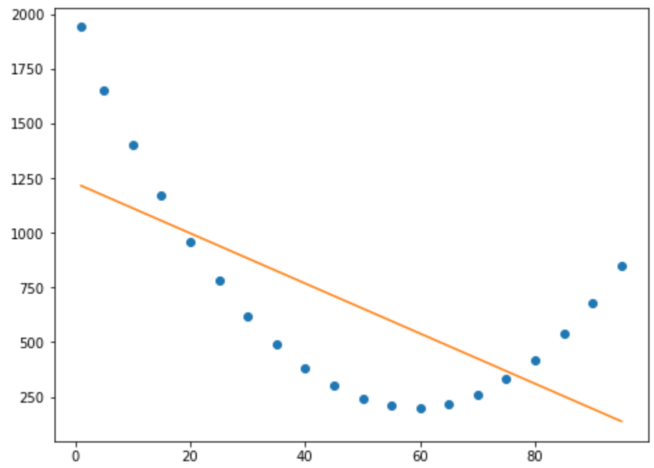
Omnibus:	2.599	**Durbin-Watson:**	0.132
Prob(Omnibus):	0.273	**Jarque-Bera (JB):**	1.974
Skew:	0.625	**Prob(JB):**	0.373
Kurtosis:	2.101	**Cond. No.**	107.

- Import modules
- Add constant
- Create model
- Fit model
- Use model to predict
- Use model and show summary

Show results

```
fig, ax = plt.subplots(figsize=(8,6))
ax.plot(x,y,'o')
ax.plot(x,z,'-')
```

```
[<matplotlib.lines.Line2D at 0x208ef0e25b0>]
```



- Graphical analysis
- Estimation vs. effective data

- What is the best model to explain
  - `Y=f(X)` ?
- A quadratic model?

Create Model

```
import pandas as pd
xy=[x,y]
df=pd.DataFrame(xy)
df1=df.T
df1.columns = ['x','y']
```

```
df1['x2']=df1['x']**2
y=df1['y']
xx=df1[['x','x2']]
```

```
xx1=sm.add_constant(xx)
model=sm.OLS(y, xx1)
result=model.fit()
result.summary()
```

- Create data frame
- Add columns
- Identify X and Y
- Create model
- Fit model

Results

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.999
Model:	OLS	**Adj. R-squared:**	0.999
Method:	Least Squares	**F-statistic:**	1.663e+04
Date:	Wed, 03 Mar 2021	**Prob (F-statistic):**	1.05e-28
Time:	14:49:02	**Log-Likelihood:**	-76.719
No. Observations:	20	**AIC:**	159.4
Df Residuals:	17	**BIC:**	162.4
Df Model:	2		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	1959.4584	7.518	260.650	0.000	1943.598	1975.319
x	-59.7517	0.367	-162.928	0.000	-60.525	-58.978
x2	0.5065	0.004	136.291	0.000	0.499	0.514

Omnibus:	22.125	**Durbin-Watson:**	1.693
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	39.243
Skew:	1.691	**Prob(JB):**	3.01e-09
Kurtosis:	8.971	**Cond. No.**	1.16e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.16e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

- Bad statistics
- But…

Results

- A graphical analysis allows to verify that we obtained a good fit.

```
z=result.predict(xx1)
```

```
fig, ax = plt.subplots(figsize=(8,6))
ax.plot(x,y,'o')
ax.plot(x,z,'-')
```

```
[<matplotlib.lines.Line2D at 0x208ef20d100>]
```