

How can I perform quantile regression in Python?

Authored by
stats writer

April 24, 2024

RECOMMENDED CITATION

stats writer (2024). *How can I perform quantile regression in Python?*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=138628>

Quantile regression is a statistical method used to estimate the relationship between a set of independent variables and a continuous dependent variable, by identifying the conditional quantiles of the dependent variable at different levels of the independent variables. This method is particularly useful when the data is not normally distributed or when the focus is on extreme values. In order to perform quantile regression in Python, one can use the "statsmodels" library, which provides a comprehensive set of tools for regression analysis. The steps involved include importing the necessary libraries, loading the data, fitting the regression model, and interpreting the results. With the availability of various functions and methods, Python offers a convenient and efficient way to perform quantile regression and obtain valuable insights from the data.

Perform Quantile Regression in Python

Linear regression is a method we can use to understand the relationship between one or more predictor variables and a .

Typically when we perform linear regression, we're interested in estimating the mean value of the response variable.

However, we could instead use a method known as quantile regression to estimate *any* quantile or percentile value of the response value such as the 70th percentile, 90th percentile, 98th percentile, etc.

This tutorial provides a step-by-step example of how to use this function to perform quantile regression in Python.

Step 1: Load the Necessary Packages

First, we'll load the necessary packages and functions:

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import statsmodels.formula.api as smf
import matplotlib.pyplot as plt
```

Step 2: Create the Data

For this example we'll create a dataset that contains the hours studied and the exam score received for 100 students at some university:

```
#make this example reproducible
np.random.seed(0)
```

```
#create dataset
obs = 100
```

```
hours = np.random.uniform(1, 10, obs)
score = 60 + 2*hours + np.random.normal(loc=0,
scale=.45*hours, size=100)
```

```
df = pd.DataFrame({'hours': hours, 'score': score})
```

```
#view first five rows  
df.head()
```

```
hours score
```

```
0 5.939322 68.764553
```

```
1 7.436704 77.888040
```

```
2 6.424870 74.196060
```

```
3 5.903949 67.726441
```

```
4 4.812893 72.849046
```

Step 3: Perform Quantile Regression

Next, we'll fit a quantile regression model using hours studied as the predictor variable and exam score as the response variable.

We'll use the model to predict the expected 90th percentile of exam scores based on the number of hours studied:

```
#fit the model
```

```
model = smf.quantreg('score ~ hours', df).fit(q=0.9)
```

```
#view model summary
```

```
print(model.summary())
```

QuantReg Regression Results

```

=====
=====
Dep. Variable: score Pseudo R-squared: 0.6057
Model: QuantReg Bandwidth: 3.822
Method: Least Squares Sparsity: 10.85
Date: Tue, 29 Dec 2020 No. Observations: 100
Time: 15:41:44 Df Residuals: 98
Df Model: 1
=====
=====
coef std err t P>|t|
-----
Intercept 59.6104 0.748 79.702 0.000 58.126 61.095
hours 2.8495 0.128 22.303 0.000 2.596 3.103
=====
=====

```

From the output, we can see the estimated regression equation:

90th percentile of exam score = 59.6104 + 2.8495*(hours)

For example, the 90th percentile of scores for all students who study 8 hours is expected to be 82.4:

The output also displays the upper and lower confidence limits for the intercept and the predictor variable hours.

Step 4: Visualize the Results

We can also visualize the results of the regression by creating a with the fitted quantile regression equation overlaid on the plot:

```
#define figure and axis
```

```
fig, ax = plt.subplots(figsize=(8, 6))
```

```
#get y values
```

```
get_y = lambda a, b: a + b * hours
```

```
y = get_y(model.params, model.params)
```

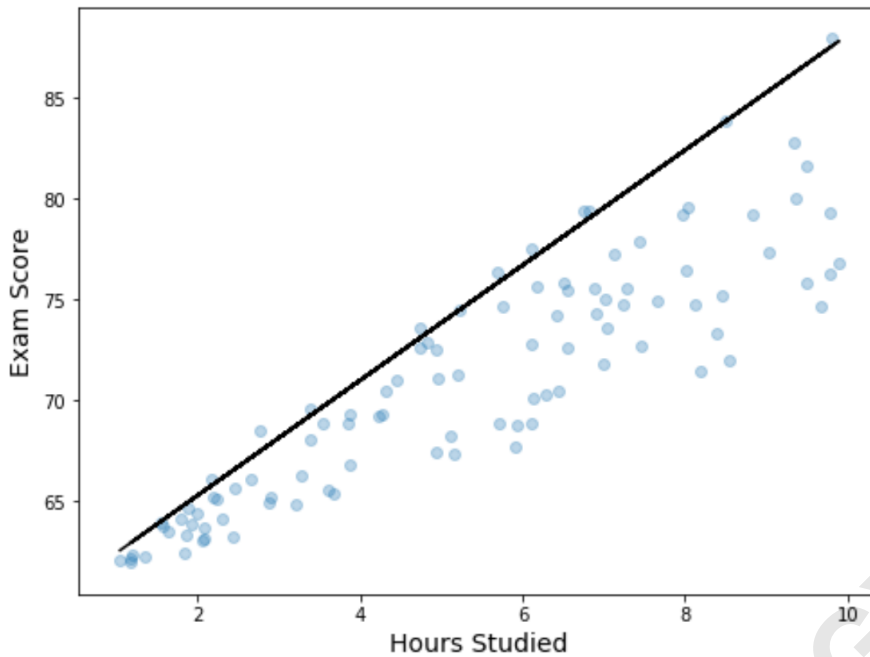
```
#plot data points with quantile regression equation overlaid
```

```
ax.plot(hours, y, color='black')
```

```
ax.scatter(hours, score, alpha=.3)
```

```
ax.set_xlabel('Hours Studied', fontsize=14)
```

```
ax.set_ylabel('Exam Score', fontsize=14)
```



Unlike a simple linear regression line, notice that this fitted line doesn't represent the "line of best fit" for the data. Instead, it goes through the estimated 90th percentile at each level of the predictor variable.