

the Tarzan

[R] + applied economics.

About

ECNS 561

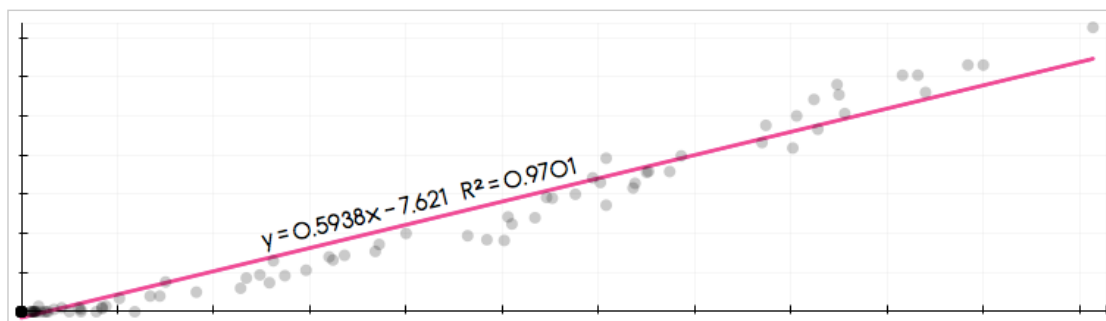
Nuts'n Bolts

Resources

« Parallel computing with package 'snowfall' |

Calculate an OLS regression using matrices in Python using Numpy

The following code will attempt to replicate the results of the `numpy.linalg.lstsq()` function in Numpy. For this exercise, we will be using a cross sectional data set provided by me in .csv format called "cdd.ny.csv", that has monthly cooling degree data for New York state. The data is available [here](#) (File -> Download).



The OLS regression equation:

$$Y = X\beta + \varepsilon$$

where ε = a white noise error term. For this example Y = the population-weighted Cooling Degree Days (CDD) (CDD.pop.weighted), and X = CDD measured at La Guardia airport (CDD.LGA). *Note: this is a meaningless regression used solely for illustrative purposes.*

Recall that the following matrix equation is used to calculate the vector of estimated coefficients $\hat{\beta}$ of an OLS regression:

$$\hat{\beta} = (X'X)^{-1}X'Y$$

where X = the matrix of regressor data (the first column is all 1's for the intercept), and Y = the vector of the dependent variable data.

Matrix operators in Numpy

- `matrix()` coerces an object into the matrix class.
- `.T` transposes a matrix.
- `*` or `dot(X, Y)` is the operator for matrix multiplication (when matrices are 2-dimensional; [see here](#)).
- `.I` takes the inverse of a matrix. Note: the matrix must be invertible.

Back to OLS

The following code calculates the 2 x 1 matrix of coefficients, $\hat{\beta}$:

```

1  ## load in required modules:
2  import numpy as np
3  import csv
4
5  ## read data into a Numpy array
6  df1 = csv.reader(open('/your/file/path/cdd.ny.csv', 'rb'), delimiter=',')
7  b1 = np.array(list(df1)[1:,3:5].astype('float'))
8
9  nrow = b1.shape[0]
10
11  intercept = np.ones( (nrow,1) )
12  b2 = b1[:,0].reshape(-1, 1)
13
14  X = np.concatenate((intercept, b2), axis=1)
15  Y = b1[:,1].T
16
17  ## X and Y arrays must have the same number of columns for the matrix multiplication to work:
18  print(X.shape)
19  print(Y.shape)
20
21  ## Use the equation above (X'X)^(-1)X'Y to calculate OLS coefficient estimates:
22  bh = np.dot(np.linalg.inv(np.dot(X.T,X)), np.dot(X.T,Y))
23  print bh
24
25  ## check your work with Numpy's built in OLS function:
26  z,resid,rank,sigma = np.linalg.lstsq(X,Y)
27  print(z)

```

Calculating Standard Errors

To calculate the standard errors, you must first calculate the variance-covariance (VCV) matrix, as follows:

Search this blog

Search...

Contributors



Goulding Kevin

Categories

Econometrics
Econometrics with R
Numpy
Python
R tips & tricks
Surviving Graduate
Econometrics with R
TikZ for Economists
Visualizing Data with R
White Papers

Twitterfeed

RT @gappy3000: This post, apparently about #julialang and #pydata, explains why #rstats has become the standard of data analysis [http:// ... 3 years ago](#)

RT @justinwolffers: "If prediction markets are really as valuable as economists think, then...more experimentation could prove worthwhile. ... 3 years ago

RT @vsbuffalo: For me the biggest victory is for statistics and empiricism. Go Nate Silver and @fivethirtyeight for a brilliant forecast ... 3 years ago

Follow @baha_kev

Tag Cloud

cluster-robust
Econometrics
heteroskedasticity

LaTeX Numpy

Parallel Computing plots

Python R STATA
tex TikZ

$$\text{Var}(\hat{\beta}|X) = \frac{1}{n-k} \hat{\epsilon}' \hat{\epsilon} (X'X)^{-1}$$

The VCV matrix will be a square $k \times k$ matrix. Standard errors for the estimated coefficients $\hat{\beta}$ are found by taking the square root of the diagonal elements of the VCV matrix.

```

1  ## Calculate vector of residuals
2  res = as.matrix(women$weight-bh[1]-bh[2]*women$height)
3  res = Y-(bh[0]+X[:,1]*bh[1])
4  ## Define n and k parameters
5  n = nrow
6  k = X.shape[1]
7
8  ## Calculate Variance-Covariance Matrix
9  VCV = np.true_divide(1,n-k)*np.dot(np.dot(res.T,res),np.linalg.inv(np.dot(X.T,X)))
10
11 ## Standard errors of the estimated coefficients
12 stderr = np.sqrt(np.diagonal(VCV))

```

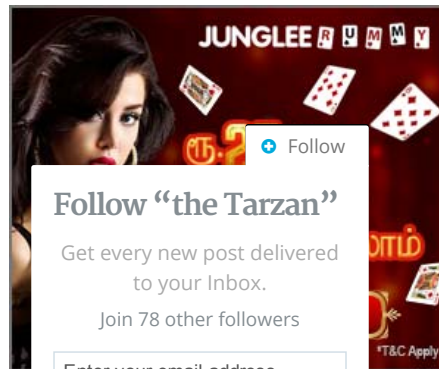
Now you can check the results above using the `lm()` function in R:

```

1  df1 = read.csv('/your/file/path/cdd.ny.csv',header=T)
2  coef(lm(CDD.pop.weighted ~ CDD.LGA,data=df1))
3
4  ## (Intercept) CDD.LGA
5  ## -7.6210191 0.5937734
6
7  summary(lm(CDD.pop.weighted ~ CDD.LGA,data=df1))

```

About these ads



Follow "the Tarzan"

Get every new post delivered to your Inbox.

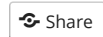
Join 78 other followers

Enter your email address

Sign me up

*T&C Apply

Share this:



Be the first to like this.

Related

[Calculate OLS regression manually using matrix algebra in R](#)
In "Econometrics with R"

[Surviving Graduate Econometrics with R: Fixed Effects Estimation -- 3 of 8](#)
In "Surviving Graduate Econometrics with R"

[Clustered Standard Errors in R](#)
In "Econometrics with R"

Posted on October 27, 2012 at 2:30 pm in [Econometrics](#), [Numpy](#), [Python](#) | [RSS feed](#) | [Reply](#) | [Trackback URL](#)

Tags: [Econometrics](#), [Numpy](#), [Python](#)

One Comment to "Calculate an OLS regression using matrices in Python using Numpy"



Julia

October 31, 2012 at 9:37 am

Hello fellow master student,

First of all your blog is amazing, we loved your section about HCCME. Very helpful and thorough. You did a great job !

Would you have any ideas on how to approach FGLS method (calculating the parameters) with autoregressive errors residuals or anything about instrumental variables?

Thanks a lot and best of luck.

Reply

Leave a Reply

Enter your comment here...

Tags

cluster-robust *econometrics* *heteroskedasticity*
latex *numpy* *parallel**computing* *plots*
python *r* *stata* *tex* *tikz*

Calendar

October 2012						
M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				
« Feb						

Archives

October 2012
February 2012
July 2011
June 2011
May 2011

Blogroll

Documentation
Plugins
Suggest Ideas
Support Forum
Themes
WordPress Blog
WordPress Planet