Bio 723: Class Session 3
Bivariate Regression

September 9, 2014

```
In []: %matplotlib inline
```

```
In []: import numpy as np
       from matplotlib import pyplot as plt
       import pandas as pd
```

```
In []: # np.random.seed seeds the random number generator
       # giving a specific seeds allows us to generate pseudo-random numbers
       # deterministically so that our results are reproducible

       np.random.seed(20140909)
```

1 Bivariate Regression

Last week we discussed how bivariate regression of an outcome variable y on a predictor variable x is equivalent to finding the variable, $\hat{\vec{y}}$, in the subspace defined by $\vec{x}$, that is closest to $\vec{y}$. We showed that $\hat{\vec{y}}$ could be obtained by projecting $\vec{y}$ onto $\hat{x}$.

For **mean centered vectors** the regression of y on x is given by is given by the following formula:

$$\hat{\vec{y}} = b\vec{x}$$

where

$$b = \frac{\vec{x} \cdot \vec{y}}{\vec{x} \cdot \vec{x}}$$

The **more general formula** for the regression of y on x is given by:

$$\hat{\vec{y}} = a + b\vec{x}$$

where a is the intercept. To solve the general case, let $\vec{x_c}$ and $\vec{y_c}$ be the mean centered vectors derived from x and y (i.e. $\vec{x_c} = \vec{x} - \bar{x}\vec{1}$), then:

$$b = \frac{\vec{x_c} \cdot \vec{y_c}}{\vec{x_c} \cdot \vec{x_c}}$$

and

$$a = \bar{y} - b\bar{x}$$

2 Regression functions in the StatsModels library

Least-squares regression functions are available from two Python libraries – StatsModels (http://statsmodels.sourceforge.net/) and SciKit-Learn (http://scikit-learn.org/stable/). For this class session we'll use the functions implemented in StatsModels.

```
In []: ## Import StatsModel modules we'll need
       import statsmodels.api as sm

       # the formula API allow us to write R-like formulas for models
       import statsmodels.formula.api as smf
```

2.0.1 Tribolium data set

We'll illustrate the least-squares regression functions in StatsModels using a simple bivariate data set from Sokal and Rohlf's textbook, Biometry. This data set, describes the weight-loss of nine batches of 25 Tribolium beetles, after six days of starvation at nine different humidities. is Table 14.1 in Biometry, 4th ed.

```
In []: # read the tribolium data set from the course repository
       tribolium = pd.read_csv('https://github.com/pmagwene/Bio723/raw/master/datasets/tribolium.csv')
       tribolium
```

```
In []: # rename the columns of the data frame for convenience
       tribolium.columns = ["humidity","wtloss"]
       tribolium
```

```
In []: # create a scatter plot for the tribolium data set
       ax = plt.axes()
       plt.plot(tribolium.humidity, tribolium.wtloss, 'ko')
       plt.xlabel('Percent Relative Humidity')
       plt.ylabel('Weight Loss (mg)')
       plt.xlim(-5, 100)
       plt.ylim(-1,10)

       # draw only left and bottom axes
       ax.spines['right'].set_visible(False)
       ax.spines['top'].set_visible(False)
       ax.xaxis.set_ticks_position('bottom')
       ax.yaxis.set_ticks_position('left')
```

2.1 Basic Least-Squares Regression in StatsModels

The basic regression function in StatsModels in `OLS` (short for "ordinary least-squares"). The `OLS` function can be used as so:

```
In []: # add_constant appends a column of ones to the matrix
       # this allows us to estimate the intercept as well as the slope
       X = sm.add_constant(tribolium.humidity)
       y = tribolium.wtloss

       # specify the model
       lm_tribolium = sm.OLS(y, X)

       # fit the model
       fit_tribolium = lm_tribolium.fit()
```

```
In []: # print a summary table
       fit_tribolium.summary()
```

```
In []: # Draw the bivarate plot with the regression line superimposed

       ax = plt.axes()
       plt.plot(tribolium.humidity, tribolium.wtloss, 'ko')
       plt.xlabel('Percent Relative Humidity')
       plt.ylabel('Weight Loss (mg)')
       plt.xlim(-5, 100)
       plt.ylim(-1,10)

       # draw only left and bottom axes
       ax.spines['right'].set_visible(False)
       ax.spines['top'].set_visible(False)
       ax.xaxis.set_ticks_position('bottom')
       ax.yaxis.set_ticks_position('left')

       # add regression line
       plt.plot(tribolium.humidity, fit_tribolium.predict(), 'r--', alpha=0.5, linewidth=2)
```

2.2 Examining the Residuals from the Regression Model

When fit a regression model, it's always good practice to plot the residuals from the regression. A model that is appropriate for the data in hand should exhibit approximately uniform scatter of residual values.

```
In []: # plot residuals vs x

       plt.plot(tribolium.humidity, fit_tribolium.resid, 'ko')

       plt.xlabel('Relative Humidity')
       plt.ylabel('Residuals')
       plt.xlim(-5,100)
       plt.hlines(0, -5, 100, linestyle='dashed', color='r', linewidth=2)
```

2.3 Using StatsModels' Formula Interface

The StatsModels package also allows one to specify statistical models using "formula strings", similar to those found in R (as we'll see in a few weeks).

The StatsModels documentation provides a brief introduction to formulas, here: http://statsmodels.sourceforge.net/devel/example_formulas.html. The underlying package that does the heavy lifting for StatsModels is called Patsy. Extensive documentation of the Patsy package can be found at http://patsy.readthedocs.org/en/latest/.

```
In []: # for this next example we'll generate some random data

       x = np.random.uniform(-2,2,size=50)
       y = 3*x + x**2 + np.random.normal(0,0.5,50)

       df = pd.DataFrame({"x":x, "y":y})
```

```
In []: df.describe()
```

```
In []: plt.plot(df.x, df.y, 'ko')
       plt.xlabel("x")
       plt.ylabel("y")
```

2.3.1 Specifying the model using a formula

```
In []: # The formula based equivalent of the sm.OLS function is smf.ols

       # here we specify and fit the model simultaneously
       xyfit = smf.ols('y ~ x', data = df).fit()
```

```
In []: xyfit.summary()
```

2.3.2 Question

Examine the regression summary above, especially the cells corresponding to R-squared, the F-statistic, and associated p-values. What do you conclude about the appropriateness of the regression model we fit?

```
In []: # use the plot() methods associated with Pandas data frames
       ax = df.plot(x = "x", y = "y", kind="scatter")

       # add regression line
       # the predict function can also be used to predict Y values for x's that
       # are not in the original data set. The call below illusrates how I did
       # this for values just outside the min- and max- x-range


       # because we used the formula notation above
       # the predict function expects a DataFrame (or dictionary)
       # with the column (key) "x"

       xextremes = pd.DataFrame({"x":[-2,2]})
       ax.plot(xextremes.x, xyfit.predict(xextremes), 'r--', alpha=0.5, linewidth=2)
```

```
In []: # As before we examine the residuals

       plt.plot(x, xyfit.resid, 'bo')
       plt.xlabel('x')
       plt.ylabel('Residuals')
       plt.hlines(0, -3, 3,linestyle='dashed', color='r', linewidth=2)
       plt.xlim(-2,2)
```

2.3.3 Question

What do you conclude about the appropriateness of the linear regression of y on x, given the residual plot above? Does your assesment gel with what you concluded based on the summary table?