Bio 723: Class Session 3
One-way ANOVA

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 One-way ANOVA, Single Factor

```
In []: ## Import StatsModels modules
       import statsmodels.api as sm
       import statsmodels.formula.api as smf

       # include the anova model
       from statsmodels.stats.anova import anova_lm

       # we'll also use the scipy.stats module
       import scipy.stats
```

1.0.1 Read the Iris data

A nice feature of Pandas is you can read data sets directly from the web by specifying a URL instead of a file name.

```
In []: iris = pd.read_csv('https://github.com/pmagwene/Bio723/raw/master/datasets/iris.csv')
```

```
In []: # examine column names
       iris.columns
```

For convenience, we'll rename the columns of the iris data set with shorter names.

```
In []: iris.columns = ["SLength", "SWidth", "PLength", "PWidth", "Species"]
```

```
In []: # let's remind ourselves of the species names in the iris data set
       iris.Species.unique()
```

1.0.2 Subset the Iris data

For today's exercises we're doing a simple one-way anova between a two groups, so we'll exclude the setosa specimens.

```
In []: # exclude the setosa data
       # see the pandas documentation on the query method
       # http://pandas.pydata.org/pandas-docs/stable/indexing.html#the-query-method-experimental

       sub_iris = iris.query("Species != 'setosa'")
```

```
In []: sub_iris.Species.unique()
```

```
In []: sub_iris.boxplot("SWidth", "Species")
```

1.0.3 Fitting the ANOVA model

As before we use the `smf.ols` function to fit the linear model. The difference here is that Species is a categorical variable. Below I show the default way to fit the model.

In the first example, I fit the model using the default "Contrast Coding" associated with the Patsy module. The default coding scheme as applied to this data it to assign to assign the first group ("versicolor") the value 0, and the other group ("virginica") the value 1.

In the second example, I explicitly make species a categorical variable using the "C()" syntax, and in addition I specifying the type of contrast coding to use, in this case a scheme called "Helmert coding". This conform to the type of "dummy coding" I discussed in lecture.

For more examples of different contrast coding schemes see: http://statsmodels.sourceforge.net/devel/contrasts.html

```
In []: lm_sub_default = smf.ols('SWidth ~ Species)', data=sub_iris).fit()
```

```
In []: # now fit the model using Helmert coding
       lm_sub = smf.ols('SWidth ~ C(Species, Helmert)', data=sub_iris).fit()
       lm_sub.summary()
```

```
In []: anova_sub =  anova_lm(lm_sub)
```

```
In []: anova_sub
```

If you're curious you can compare the so-called "design matrices" created by these two different contrast coding schemes as so

```
In []: from patsy import dmatrix

       contrast1 = dmatrix('Species', sub_iris)
       contrast2 = dmatrix('C(Species, Helmert)', sub_iris)
```

```
In []: # the first column is the intercept
       # second column represents the group coding
       print contrast1
```

```
In []: # the first column is the intercept
       # second column represents the group coding
       print contrast2
```

1.1 Carrying out ANOVA by hand using vector operations

It's instructive to carry out the equivalent one-way ANOVA "by hand" using dot products so you can understand the relationship between the values returned to by a standard ANOVA function and the equivalent regression.

```
In []: y = sub_iris.SWidth.values

       # read help for np.where
       dummy_coding = np.where(sub_iris.Species == 'versicolor', -1, 1)
```

```
In []: # examine our dummy coding of group membership
       dummy_coding
```

```
In []: # mean-center the variables
       ctr_y = y - np.mean(y)
       ctr_dummy = dummy_coding - np.mean(dummy_coding)

       # calculate coefficient and intercept
       b = np.dot(ctr_y, ctr_dummy) / np.dot(ctr_dummy, ctr_dummy)
       intercept = np.mean(y) - b * np.mean(dummy_coding)

       # calculate yhat
       yhat = b * ctr_dummy
       len_yhat = np.sqrt(np.dot(yhat, yhat))
       len_y = np.sqrt(np.dot(ctr_y, ctr_y))
       df_yhat = 1

       # calculate error term
       error = ctr_y - yhat
       len_error = np.sqrt(np.dot(error, error))
       df_error = len(y) - 2

       # coefficient of determination is R**2
       R = len_yhat/len_y

       # corresponding F-statistic
       F = (len_yhat**2/df_yhat) / (len_error**2/df_error)
```

```
In []: print "Intercept: ", intercept
       print "Coefficient, b: ", b
       print "R^2: ", R**2
       print "F: ", F

       # call the "survival function" sf() associated with the f-distribution
       # to calculate probability that F would be as large or greater than
       # our calculated value under the null hypothesis
       print "P(> F): ", scipy.stats.f.sf(F, 1, 98)
```

1.2 ANOVA with randomly generated data

For completeness, let's do an ANOVA for an artificial data set where know that there should be no underlying differences in group means.

```
In []: X = np.random.normal(loc=5, scale=1, size=100)
       groups = ['A']*50 + ['B']*50
       df = pd.DataFrame({'X':X, 'group':groups})
```

```
In []: df.boxplot('X','group')

In []: lm_df = smf.ols('X ~ C(group, Helmert)', data=df).fit()
       table_df = anova_lm(lm_df)

In []: table_df

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