logistic-regression

September 23, 2014

```
In []: import numpy as np
       import pandas as pd
       import statsmodels.api as sm
       import statsmodels.formula.api as smf
```

```
In []: %matplotlib inline
       from matplotlib import pyplot as plt
```

0.1 Titanic data set

The titanic.txt data set contains information on the survival of passengers of the ship RMS Titanic, following it's sinking in the North Atlantic in 1912. In addition to survival information, this data set includes information such as passengers age, the passenger class (1st, 2nd, 3rd), etc.

The titanic data set was obtained from http://lib.stat.cmu.edu/S/Harrell/data/descriptions/titanic.html. That site also includes a more detailed description of the data's provenance.

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

```
In []: titanic.columns
```

```
In []: # read the documentation on the drop attribute of DataFrames
       help(pd.DataFrame.drop)
```

```
In []: # drop the useless "row.names" columns
       titanic.drop('row.names', axis=1, inplace=True)
```

```
In []: males = titanic[titanic.sex == 'male']
```

```
In []: # alternate way to subset data, using query method
       females = titanic.query('sex == "female"')
```

```
In []: # the query method allow for complex queries
       adult_males = titanic.query('sex == "male" & age > 18')
       young_males = titanic.query('sex == "male" & age < 18')

       # isnull() returns true for missing values
       noage_males = males[males['age'].isnull()]
```

```
In []: print "Adult male data set dimensions: ", adult_males.shape
       print "Young male data set dimensions: ", young_males.shape
       print "Data set dimensions for males where age is not known: ", noage_males.shape
```

1 Logistic regression using StatsModels

As with least squares regression, StatsModels provides a formula based interface for carrying logistic regression from the `statsmodels.formula.api`.

```
In []: # fit a logistic regression of survival on age
       survival = smf.logit("survived ~ age", data = titanic).fit()

In []: print survival.summary()

In []: # generate a plot showing predicting survival as a function of age
       age = np.linspace(0, 80, 100)
       pred_survival = survival.predict(pd.DataFrame({'age':age}))
       plt.plot(age, pred_survival, 'r-')

       # overplot the observed data
       plt.plot(titanic.age, titanic.survived, 'ko', alpha=0.5)
       plt.ylim(-0.05, 1.05)
       plt.xlabel('Passenger Age')
       plt.ylabel('Probability of Survival')
```

2 The Seaborn Statistical Plotting Library

Seaborn is a plotting library, built on top of Matplotlib, that specializes in generating nice statistical figures.

In addition to providing a set of useful functions, Seaborn changes the default plotting and background colors for for Matplotlib plots to provide better color discrimination and more readable plots. If you don't like the changes that Seaborn makes you can change the defaults (see the Seaborn documentation)

2.1 Logistic Regression Plots in Seaborn

Seaborn makes it very easy to fit regression models. The basic plotting function for all regression models is `regplot()`. If we wish to fit a logistic regression we can simply set the `logistic` argument to `True`.

```
In []: import seaborn as sns
       g = sns.regplot("age", "survived", data=titanic, y_jitter=.02, logistic=True)
```

2.2 Subsetting on categorical variables

Another powerful feature of Seaborn is that it makes it very easy to subset data on categorical variables. To do this we use the `lmplot()` function which combines the `regplot()` with a class for handling faceted data called a `FacetGrid` (see the Seaborn docs).

```
In []: # gender stereotypes!

       # you can specify colors using HTML hex codes
       # see for example http://html-color-codes.info/
       colors = dict(male='#6495ED',   # blue hue
                     female='#F08080')   # pink/red hue

       g = sns.lmplot("age", "survived", hue='sex', palette=colors,
                      data=titanic, y_jitter=.02, logistic=True)
```

In the plot above, the overlapping observations are a bit messy, so let's break out the logistic regressions for the separate sexes into two side by side panels.

```
In []: # as before, but now we specify a col(umn) variable to define a grid of subplots
       g = sns.lmplot("age", "survived", hue='sex', col='sex',
                      palette=colors, data=titanic, y_jitter=.02, logistic=True)

       # simultaneously set limits for all the subplots
       g.set(xlim=(0, 80), ylim=(-.05, 1.05))
```

Breaking the data set down by sex reveals an important trend that was masked in the data as a whole – probability of survival increases with age in women, but decreases with age in men!

2.3 Exploring survival as a function of passenger class

Now let's explore how the passengers ticket class affected the probability of survival.

2.3.1 Grouping in Pandas

When working with categorical variables in Pandas, the `groupby` function is very useful.

```
In []: # create the grouping
       class_grouping = titanic.groupby('pclass')

In []: # we can use the grouping to get counts across groups
       class_grouping.pclass.count()

In []: # grouping allows use to get sophisticated with functions applied to groups
       # here we calculate the median age across each passenger class
       class_grouping.age.median()

In []: # groupings can be defined by multiple categorical variables
       grouping2 = titanic.groupby(['survived','pclass'])
```

2.3.2 Creating tables with groupings

```
In []: # create a table giving the total number of passengers who survived or died
       # broken down across class
       grouping2.survived.count()

In []: # express the above table in terms of the number of passengers in each class
       grouping2.survived.count() / class_grouping.pclass.count()
```

2.3.3 Interpretation and Logistic Regression on Class

Our table above suggests that approximately 40% of first class passengers died, while >80% of 3rd class passengers died! Let's create a complementary logistic regression for the data broken down by class and sex.

```
In []: # logistic regression requires that our predictor variables be quantitative
       # create a new variable, mapping class name to a quantitative value
       titanic['class'] = titanic.pclass.map({'1st':1, '2nd':2, '3rd':3})

In []: # as before, but now we specify a col(umn) variable to define a grid of subplots
       g = sns.lmplot("class", "survived", hue='sex', col='sex',
                      palette=colors, data=titanic, x_jitter = .02, y_jitter=.02, logistic=True)
```

The logistic regression shows us that the effect of passenger class on survival is particularly prominent for the female passengers!

```
In []:
```