Practical 14: Logistic Regression

A O Majekodunmi

IBI1, 2018/19

1 Learning objectives

- Name and describe an example of a public health informatic project
- Name and describe common databases for nucleotide sequences, protein structures, biomedical literature, public health data
- Analyse a public health dataset (e.g. from fluview) in Python

2 Introduction

The Centre for Disease Control compiles data on influenza activity year-round in the United States and produces FluView, a weekly influenza surveillance report, and FluView Interactive, which allows for more in-depth exploration of influenza surveillance data.

Surveillance includes detection and reporting of **Novel Influenza A Virus Strains** which are a notifiable condition. This includes all human infections with influenza A viruses that are different from currently circulating human seasonal influenza H1 and H3 viruses, including those with a non-human origin. These infections pose a high risk of severe infections and pandemic spread as there is little to no pre-existing immunity in the population. Rapid detection and reporting of human infections with these is important to accelerate the implementation of effective public health responses.

In this practical we will work on the Fluview data on Novel Influenza A viruses from 2010 – 2018. The task is to analyse the available data using logistic regression to determine risk factors for infection with these Novel Influenza A viruses.

2 Importing Excel data in Python

We will use `pandas` (`https://pandas.pydata.org`) to manage the data importation and prepossessing into Python. The `pandas` module has rapidly become a standard means for the Python data scientist to process time series and tabular data sets, providing natural ("pythonic") and high-performance data structures and analysis tools. From a conceptual viewpoint, `pandas` provides a data-frame based interface to tabular data in a manner similar to a computer spreadsheet or the fundamental data-type employed by the the R language (`https://www.r-project.org`). Many conveniences are provided to handle the ugly realities faced when processing "real data," for example: missing entries, multiple data-types, and numerous file storage formats.

The following serves as the most basic introduction to data processing with `pandas`.

- Let us first make sure that `pandas` has been included in your installation.

  ```
  import pandas as pd
  ```

  Your Anaconda Python installation should include `pandas` and the above will produce no errors.

- Load the data set (`Final_Fluview_Practical_dataset.xlsx`) for this practical into Excel. Inspect the data set.

- You can load data from an Excel spreadsheet with the following:

  ```
  df = pd.read_excel('Final_Fluview_Practical_dataset.xlsx')
  ```

- With any new data set, it is critical to inspect and understand it prior to any analysis. Gather some information first by calling the `info` method:

  ```
  df.info()
  ```

  Does the printed information agree with what you expect to see given the raw data displayed in Excel?

- Print the first 5 rows of the tabular data using the `head` method:

  ```
  df.head(5)
  ```

  Do the data from the first 5 rows agree with the first 5 rows displayed in Excel?

3 Data exploration

Before analyzing a new data set, it is always wise to first inspect the values and other factors. It is always possible that there is something unusual in the data set or the way that is formatted that can confuse the software written to import the data. As always in computation: garbage in yields garbage out, i.e., the analysis and its proper interpretation are useless (even less than useless, because the result is incorrect and you may not realize this fact) if the data are incorrect.

- Use `pandas` to get a sense of the values that appear in each column of the data set by using the `describe` method:

  ```
  df.describe()
  ```

  What is the `count` and `unique` for the values in each column? Is this what you expect?

- Let us display the unique values found in each column of the data set:

```
for column in df:
    print(column)
    print(df[column].unique())
    print('')
```

What do you see? Does this make sense given the values you would expect to see in a given column?

4 Data preparation

The data must be formatted in a manner consistent with the expectations of the logistic regression analysis package. The following will guide you though the process for this data set.

- Only a subset data columns are relevant to this exercise. We now *slice* those columns from the full data set.

  ```
  df_regress = df[['Virus Strain',
                   'Age',
                   'Gender',
                   'Hospitalized?',
                   'Swine Contact?',
                   'Attended Agricultural Event?']]
  ```

  Inspect the first 5 rows of the sliced data frame with the `head` method. Is this what you expect?

- It is possible that some values are missing in the data set. By default, `pandas` will insert a special value called `NaN` (not a number) in these cases. Calculations involving `NaN` will typically produce `NaN` to indicate to the user that a problem involving missing data has been encountered. Let us first query the data set and display those rows, if any, that have missing values.

  ```
  df_regress[df_regress.isna().any(axis=1)]
  ```

  How many rows have missing values?

  The above command invocation likely seems quite magical. If you are interested, a full explanation is provided here: `https://stackoverflow.com/questions/43424199/display-rows-with-one-or-more-nan-values-in-pandas-dataframe`. The basic idea is to select a slice of the data frame that has missing values given by calling the `isna` method.

- Let us remove all rows with missing values. This is easily done with the `dropna` method.

  ```
  df_regress = df_regress.dropna()
  ```

  Confirm that the rows with missing value have been removed.

- The logistic regression analysis package does not understand how to interpret values such as `'Yes'` or `'No'`. Categorical data must be converted to integers, e.g., `'Yes'` replaced by `1`.

  Before making any categorical value substitutions, let us first understand the values that appear in each column of the data set. The following will display the set of values that appear across each column

```
for column in df_regress:
    print(column, df_regress[column].unique())
```

In words, the above states: iterate over each column name of the data frame, select that column, form the set of values in that column using the `unique` method, and print the column name and unique value set.

- The categorical value substitution must be done column by column. The following is an appropriate substitution for the age column:

```
df_regress['Age'] = df_regress['Age'].map({'<18 Years': 0,
                                           '>=18 Years': 1})
```

Determine suitable substitutions for the remaining columns. Confirm the substitutions have been successful.

5 Logistic regression

We will use the `statsmodels` module (`https://www.statsmodels.org`) to calculate logistic regressions.

- The module should have been included as part of your Anaconda Python installation. Confirm this by attempting to import the module with the following:

```
import statsmodels.api as sm
import statsmodels.formula.api as smf
```

The above should produce no errors.

- We must split the data between endogenous (dependent) and exogenous (independent) variables:

```
endog = df['Virus Strain']
exog = df[['Age',
           'Gender',
           'Hospitalized?',
           'Swine Contact?',
           'Attended Agricultural Event?']]
```

- To fit a y-intercept parameter, we must add a data column with all values equal to 1.

```
exog = sm.add_constant(exog)
```

- A logistic regression can them be executed with the following:

```
logit = smf.Logit(endog, exog)
result = logit.fit()
```

A successful result will be accompanied by the message `Optimization terminated successfully`.

- An exhaustive report of the regression is provided by the `summary` method:

```
result = summary()
```

For the H3N2 influenza strain, the correct result is:

Dep. Variable:	H3N2v	**No. Observations:**	462
Model:	Logit	**Df Residuals:**	456
Method:	MLE	**Df Model:**	5
Date:	Tue, 21 May 2019	**Pseudo R-squ.:**	0.05986
Time:	16:53:47	**Log-Likelihood:**	-116.53
converged:	True	**LL-Null:**	-123.95

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	0.9578	0.640	1.496	0.135	-0.297	2.213
Age	-0.3496	0.492	-0.711	0.477	-1.314	0.615
Gender	0.6460	0.370	1.744	0.081	-0.080	1.372
Hospitalized?	1.6116	1.066	1.511	0.131	-0.478	3.702
Swine Contact?	-0.0094	0.580	-0.016	0.987	-1.146	1.128
Attended Agricultural Event?	1.4649	0.447	3.279	0.001	0.589	2.341

- The odds ratios can be calculated with the following:

```
import numpy as np

print('Odds ratios:')
print(np.exp(result.params))
```

The odds ratios are reported in the same order as the exogenous variables are ordered in the regression report.