Analysing data on Midwife Led Units (MLUs)

This notebook outlines a basic process for analysing a dataset based on FOI requests to hospital trusts, resulting in this story (http://www.bbc.co.uk/news/uk-england-37471091). The GitHub repo for this story can be found here (https://github.com/BBC-Data-Unit/midwife-led-units).

First, import the `pandas` library (with an alias of `pd`), and import the data into a variable called `mludata`. Use the `info()` method to find out what columns it has:

In [3]:
```
import pandas as pd
mludata = pd.read_csv("https://raw.githubusercontent.com/BBC-Data-Unit/midwife-led-units/master/Midwife-led%20units%20BBC%20investigation.csv")
mludata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52 entries, 0 to 51
Data columns (total 11 columns):
NHS Trust                                             52 non-null object
When the midwife-led unit opened                      45 non-null object
Midwife-unit                                          52 non-null object
Nearest consultant-led unit                           48 non-null object
Distance to consultant in miles                       52 non-null object
Alternative shorter distance in miles                 1 non-null float64
Deliveries commenced at midwife-led unit 2015-16      52 non-null int64
Deliveries completed at midwife-led unit 2015-16      52 non-null int64
Transferred from midwife unit to consultant 2015-16   52 non-null int64
% transferred 2015-16                                 52 non-null object
Deliveries commenced in consultant-led unit 2015-16   36 non-null float64
dtypes: float64(2), int64(3), object(6)
memory usage: 4.5+ KB
```

The key question is how many patients end up being transferred from the MLU to a consultant. That information is stored in the "Transferred from midwife unit to consultant 2015-16" column as a whole number, and "% transferred 2015-16" as a percentage.

Adding up or averaging percentages isn't a good idea, unless you're going to weight those percentages by the size of the MLU that it represents (you don't want to treat one MLU with a 90% transferral rate but only 10 patients as just the same as one with 10% transferral rate but 100 patients)

Instead it's best to calculate new overall percentages based on the numbers.

Let's get just the column names using the `.columns` method:

In [5]:
```
mludata.columns
```

Out[5]:
```
Index([u'NHS Trust', u'When the midwife-led unit opened', u'Midwife-unit',
       u'Nearest consultant-led unit ', u'Distance to consultant in miles',
       u'Alternative shorter distance in miles',
       u'Deliveries commenced at midwife-led unit 2015-16',
       u'Deliveries completed at midwife-led unit 2015-16',
       u'Transferred from midwife unit to consultant 2015-16',
       u'% transferred 2015-16',
       u'Deliveries commenced in consultant-led unit 2015-16'],
      dtype='object')
```

Because the names have spaces, they are best accessed using square brackets like so:

In [8]:
```
mludata['Transferred from midwife unit to consultant 2015-16'].head()
```

Out[8]:
```
0    230
1    569
2     78
3    560
4     74
Name: Transferred from midwife unit to consultant 2015-16, dtype: int64
```

But it's easier to rename using `.rename()` (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.rename.html)

In [12]:
```
mludata = mludata.rename(columns={'Transferred from midwife unit to consultant 2015-16':'transferred1516'})
mludata.columns
```

Out[12]:
```
Index([u'NHS Trust', u'When the midwife-led unit opened', u'Midwife-unit',
       u'Nearest consultant-led unit ', u'Distance to consultant in miles',
       u'Alternative shorter distance in miles',
       u'Deliveries commenced at midwife-led unit 2015-16',
       u'Deliveries completed at midwife-led unit 2015-16', u'transferred151
6',
       u'% transferred 2015-16',
       u'Deliveries commenced in consultant-led unit 2015-16'],
      dtype='object')
```

Now to find out the total number of patients transferred in 2015/16, using `.sum()` on that column:

In [13]:
```
mludata.transferred1516.sum()
```

Out[13]:
```
10489
```

To work out what that is as a percentage of all deliveries, we need to divide it by the total. First, let's rename the column showing completed deliveries in each MLU and then calculate a total of all completed deliveries in all trusts:

In [16]:
```
mludata = mludata.rename(columns={'Deliveries completed at midwife-led unit 2015-16':'deliveriescompleted1516'})
mludata.deliveriescompleted1516.sum()
```

Out[16]: 28060

Based on those two figures, then - 10,489 transferrals to consultant led units against 28,060 deliveries completed in MLUs - we can guess that the proportion is going to be somewhere around a quarter. But a simple division doesn't give us that:

In [39]:
```
mludata.transferred1516.sum() / (mludata.transferred1516.sum()+mludata.deliveriescompleted1516.sum())
```

Out[39]: 0

Why is that? Well, it's to do with the *type* of data which is being stored in that column: it's an *integer*, or whole number.

A percentage, on the other hand, almost always has decimal places: it is a *float* (50%, for example, is expressed as 0.5, 75% would be 0.75 and so on. Only 100% - 1 - would be an integer).

You can see this in action by trying to divide 2 by 4. 2 out of 4 should come out as 50%, but instead we get 0 again:

Note: this isn't a problem in Python 3, where two integers divided by each other can produce a float. In Python 2, however, we need to solve it

In [34]:
```
2/4
```

Out[34]: 0

Adding a decimal place to those two numbers, however, changes the result:

In [35]:
```
2.0/4.0
```

Out[35]: 0.5

We have two options to solve our problem: add decimal places to the column values (in other words, change the data type for those numbers from integers to floats); or add decimal places to the totals themselves. Here's how to do the latter by using the `float()` function:

In [40]:
```
float(mludata.transferred1516.sum()) / (float(mludata.transferred1516.sum())+float(mludata.deliveriescompleted1516.sum()))
```

Out[40]: 0.27209525538924484

We could also divide the transferrals by all deliveries *commenced* at the MLU (in other words, what proportion of those deliveries which began in an MLU ended up transferring to a consultant led unit) - which gives us the same result.

In [42]:
```
float(mludata.transferred1516.sum()) / float(mludata['Deliveries commenced at
 midwife-led unit 2015-16'].sum())
```

Out[42]: 0.27209525538924484

And we can work out what the split is between pregnancies that commence in a consultant led unit, and those which begin in an MLU. This time I've added a couple of print commands so you can see the parts of the calculation:

In [53]:
```
print 'total Deliveries commenced at midwife-led unit 2015-16:', mludata['Deli
veries commenced at midwife-led unit 2015-16'].sum()
print 'total Deliveries commenced at consultant-led unit 2015-16:', mludata['D
eliveries commenced in consultant-led unit 2015-16'].sum()
print 'total Deliveries commenced in either unit 2015-16:', mludata['Deliverie
s commenced at midwife-led unit 2015-16'].sum()+mludata['Deliveries commenced
 in consultant-led unit 2015-16'].sum()
float(mludata['Deliveries commenced at midwife-led unit 2015-16'].sum())/(floa
t(mludata['Deliveries commenced at midwife-led unit 2015-16'].sum())+float(mlu
data['Deliveries commenced in consultant-led unit 2015-16'].sum()))
```

```
total Deliveries commenced at midwife-led unit 2015-16: 38549
total Deliveries commenced at consultant-led unit 2015-16: 141961.0
total Deliveries commenced in either unit 2015-16: 180510.0
```

Out[53]: 0.21355603567669382

Note that `float` is applied to the *results* of the `sum` *before* it is then divided by the other number. If you try to apply `float` *after* the division you still get zero, only with a decimal place added, because the result is still zero *before* the conversion to float is applied.

In [37]:
```
float(mludata.transferred1516.sum() / mludata.deliveriescompleted1516.sum())
```

Out[37]: 0.0

While we're on data types, notice that the percentages column actually contains text, not numbers, because the percentage symbol is interpreted as a text character rather than a number.

In [38]:
```
mludata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52 entries, 0 to 51
Data columns (total 11 columns):
NHS Trust                                            52 non-null object
When the midwife-led unit opened                     45 non-null object
Midwife-unit                                         52 non-null object
Nearest consultant-led unit                          48 non-null object
Distance to consultant in miles                      52 non-null object
Alternative shorter distance in miles                1 non-null float64
Deliveries commenced at midwife-led unit 2015-16     52 non-null int64
deliveriescompleted1516                              52 non-null int64
transferred1516                                      52 non-null int64
% transferred 2015-16                                52 non-null object
Deliveries commenced in consultant-led unit 2015-16  36 non-null float64
dtypes: float64(2), int64(3), object(6)
memory usage: 4.5+ KB
```

So if you try to add those numbers you'll get a result like this:

In [78]:
```
mludata['% transferred 2015-16'].sum()
```

Out[78]:
```
'27%34%17%58%22%29%35%20%18%28%21%30%37%23%30%37%20%19%17%20%21%32%31%23%17%2
1%25%31%22%16%22%23%25%23%22%19%29%12%20%17%19%0%27%40%34%19%27%12%32%31%32%2
4%'
```

Cleaning a percentages column from strings to numbers

Here's how to sort that column. First, we need to replace all the % symbols (which is what causes Python to interpret the column as a series of strings) using `.str.replace()` - and create a new column in the dataset containing the results:

In [68]:
```
print 'Original column looks like this:\n', mludata['% transferred 2015-16'].head()
mludata['%transferred1516'] = mludata['% transferred 2015-16'].str.replace('%','')
print '---'
print 'After replacing all % symbols it looks like this:\n', mludata['%transferred1516'].head()
```

```
Original column looks like this:
0    27%
1    34%
2    17%
3    58%
4    22%
Name: % transferred 2015-16, dtype: object
---
After replacing all % symbols it looks like this:
0    27
1    34
2    17
3    58
4    22
Name: %transferred1516, dtype: object
```

To convert a column to a number, use `pd.to_numeric()` :

In [75]:
```
mludata['%transferred1516'] = pd.to_numeric(mludata['%transferred1516'])
#Check if it's worked - among the columns you should see that %transferred1516 is "52 non-null int64", meaning an integer
mludata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52 entries, 0 to 51
Data columns (total 12 columns):
NHS Trust                                               52 non-null object
When the midwife-led unit opened                        45 non-null object
Midwife-unit                                            52 non-null object
Nearest consultant-led unit                             48 non-null object
Distance to consultant in miles                         52 non-null object
Alternative shorter distance in miles                   1 non-null float64
Deliveries commenced at midwife-led unit 2015-16        52 non-null int64
deliveriescompleted1516                                 52 non-null int64
transferred1516                                         52 non-null int64
% transferred 2015-16                                   52 non-null object
Deliveries commenced in consultant-led unit 2015-16     36 non-null float64
%transferred1516                                        52 non-null int64
dtypes: float64(2), int64(4), object(6)
memory usage: 4.9+ KB
None
```

Now that that column is numeric, we can calculate the average percentage of transferrals across all units

In [76]:
```
mludata['%transferred1516'].mean()
```

Out[76]:
```
24.807692307692307
```

Note that this is different from the percent we got when we divided the total transferrals by the total deliveries. This is because some units will be bigger than others, so as explained earlier, this figure is less reliable and should not be used. If we were to describe it we might say something like "The average midwife led unit transferred 25% of deliveries to a consultant led unit", whereas the earlier figure can be described as "27% of patients in midwife led units were transferred to a consultant led unit" - much simpler and clearer.