Reshaping the Data

This lesson explains the possible ways to reshape and arrange the data using Pandas.

WE'LL COVER THE FOLLOWING

Pivot Table

Somewhat like Excel, we can pivot our data using pandas pivot_table functionality. To do so, we will use the `pivot_table()` function.

The `values` parameter is the column being used for aggregation, the `index` parameter is for the index values that creates multiple rows, and the `columns` parameter is for the value on which you want to have multiple columns created.

You can also use the `aggfunc` parameter to pass a function with which to aggregate your pivots.

Let's look at an example:

```
import numpy as np
import pandas as pd
names = ['age', 'workclass', 'fnlwgt', 'education', 'educationnum', 'maritalstatus', 'occupat
        'sex', 'capitalgain', 'capitalloss', 'hoursperweek', 'nativecountry', 'label']
train_df = pd.read_csv("adult.data", header=None, names=names)

# Pivot the data frame to show by relationship, workclass (rows) and label (columns) the aver
print(pd.pivot_table(train_df, values='hoursperweek', index=['relationship','workclass'],
               columns=['label'], aggfunc=np.mean).round(2))
```

Now we have a table of the average hours per week for a given `relationship`, `workclass`, and `label`.

Cross Tab

`Crosstab` is a nice way to get frequency tables. What you do is pass two columns to the function and you will get the frequency of all the pair-wise combinations of those two variables.

Let's look at an example using **label** and **relationship** as our columns:

```
import numpy as np
import pandas as pd
names = ['age', 'workclass', 'fnlwgt', 'education', 'educationnum', 'maritalstatus', 'occupat
        'sex', 'capitalgain', 'capitalloss', 'hoursperweek', 'nativecountry', 'label']
train_df = pd.read_csv("adult.data", header=None, names=names)

# Calculate the frequencies between label and relationship
print(pd.crosstab(train_df['label'], train_df.relationship))
```

We now have counts broken down by **relationship** and **label**. The first parameter is for the *rows*, and the second is for the *columns*. We can also normalize the results using the `normalize=True` parameter.

```
import numpy as np
import pandas as pd
names = ['age', 'workclass', 'fnlwgt', 'education', 'educationnum', 'maritalstatus', 'occupat
        'sex', 'capitalgain', 'capitalloss', 'hoursperweek', 'nativecountry', 'label']
train_df = pd.read_csv("adult.data", header=None, names=names)

# Crosstab with normalized outputs
print(pd.crosstab(train_df['label'], train_df.relationship, normalize=True))
```

Reshape

With Pandas, you can use `pivot()` to reshape your data. To illustrate this concept, I will use code from this post to create a dataframe in a long format.

```
import pandas.util.testing as tm; tm.N = 3
import numpy as np
import pandas as pd

def unpivot(frame):
    N, K = frame.shape
    data = {'value' : frame.values.ravel('F'),
            'variable' : np.asarray(frame.columns).repeat(N),
            'date' : np.tile(np.asarray(frame.index), K)}
    return pd.DataFrame(data, columns=['date', 'variable', 'value'])
df = unpivot(tm.makeTimeDataFrame())
print(df)
```

As you can see from the output above, a long format is where you have a variable which acts like a potential column. In this example, **variable** has values *A*, *B*, and *C*. This is a long format. To make it a wide format, we would make columns *A*, *B*, and *C* and remove the *variable* column.

Long to wide format

Here is how we would covert this long format dataframe to a wide format:

```
import pandas.util.testing as tm; tm.N = 3
import numpy as np
import pandas as pd

def unpivot(frame):
    N, K = frame.shape
    data = {'value' : frame.values.ravel('F'),
            'variable' : np.asarray(frame.columns).repeat(N),
            'date' : np.tile(np.asarray(frame.index), K)}
    return pd.DataFrame(data, columns=['date', 'variable', 'value'])
df = unpivot(tm.makeTimeDataFrame())

# Use pivot to keep date as the index and value as the values, but use the vaiable column to
df_pivot = df.pivot(index='date', columns='variable', values='value')
print(df_pivot)
```

We use the `pivot()` function. The `index` parameter is the column we want for the index of our new dataframe. The `columns` variable is for the column for which we want to use the unique values to create new columns. The `values` parameter is which column we want to use to populate the values of these new columns.

Wide to long format

To convert the format form wide back to long, Pandas provide us with the functionality of unstacking via function `unstack()`. This function (in this situation) will stack our columns back into a single variable.

Let's see an example below:

```
import pandas.util.testing as tm; tm.N = 3
import numpy as np
import pandas as pd

# Create long dataframe
def unpivot(frame):
    N, K = frame.shape
    data = {'value' : frame.values.ravel('F'),
            'variable' : np.asarray(frame.columns).repeat(N),
            'date' : np.tile(np.asarray(frame.index), K)}
    return pd.DataFrame(data, columns=['date', 'variable', 'value'])
df = unpivot(tm.makeTimeDataFrame())

# Convert to wide format
df_pivot = df.pivot(index='date', columns='variable', values='value')

# Convert back to long format
print(df_pivot.unstack())
```

Now that you are familiar with techniques to gather statistics on data and to arrange it the way you want to, the next lesson brings a challenge for you to solve.