Coastal vs Non-Coastal Statistics

May 9, 2019

1 Benchmark 1 Companion Notebook

1.1 Discrete Scores of Coastal vs Non-Coastal Districts

This notebook is meant to be a companion to the benchmark 1 analysis in the Mathematics of Redistricting textbook (authored by the Mathematics majoring seniors of Willamette University as a part of the Mathematics Senior Seminar course). Throughout the notebook, we will perform basic two-tailed t-tests to see whether or not the mean scores of coastal and non-coastal districts significantly differ.

First, it is important to have this file saved in the same location as the two .csv files that were also in the GitHub repository (sourced from https://github.com/mggg/DiscreteCompactness); All of our analysis will be done on information from those tables.

We encourage the reader to read through the comments of each cell to understand the code better.

1.2 Imports

Before we begin analysis, we must import a few packages:

In [2]:
```
# Handle imports first!

# This will be used to create dataframes and read in .csv files.
import pandas as pd

# This has many quality of life methods for analysis.
import numpy as np

# This package contains many functions that will basically perform the tests for us.
from scipy import stats
```

1.3 Reading in the tables

Now we read in the tables so that we may look at the data. To see what the tables look like, one can either open the .csv file, or uncomment line 8 or 9 and run the cell again.

In [3]:
```
# Read in the large tables. Ensure that the tables are saved
# in the same location (working directory) as this notebook is!
```

```
bigTable = pd.read_csv('final_big_table.csv', header = 0)
zoomTable = pd.read_csv('final_zoom_table.csv', header = 0)

# Uncomment these to see what they look like!
#bigTable
#zoomTable
```

1.4 Defining coastal districts

As mentioned in the textbook, there is no variable or delimiters that separate the coastal districts from the non-coastal districts (as far as we could find). As such we manually look at publically available maps and create lists that denote which districts are coastal (along the ocean).

While the example in the textbook focuses on the example of California, we have created lists for some other states and their coastal districts. Note, however, that these lists only consider ocean-incident districts, and not those bounded by lakes, mountain ranges, or other natural boundaries that may also cause coastline effects.

The districts are denoted by their Geo-ID, as covered in the textbook.

In [4]:
```
# Lists of coastal districts in ocean-incident states

# If you wish to add more states manually, use the following
# template:

# state = ['State Abbreviation', [CoastalDistrict1, CoastalDistrict2,...]]

california = ['CA',[602,603,605,609,611,612,613,614,617,618,620,624,
                    626,633,644,647,648,649,651,652,653]]

florida = ['FL',[1201,1202,1204,1206,1208,1211,1212,1213,1216,1217,
                 1218,1219,1221,1222,1223,1226,1227]]

louisiana = ['LA',[2202,2203,2207]]

oregon = ['OR', [4101,4103,4104,4105]]

texas = ['TX',[4827,4814,4822,4809]]

washington = ['WA', [5301,5302,5303,5306]]
```

1.5 Function definition

Here we define all of our functions so that we may use them modularly in future cells. The comments in the code go over what each specific function does.

Recall that the Welch t-test is a parametric test (i.e. requires the source distributions to be normal). As such, we use the Shapiro-Wilk test to test for normality before the Welch t-test.

If the Shapiro-Wilk test indicates that the data is not a part of a normal distribution, we use the rank transformation on the data before performing the Welch t-test.

Special Note: Although numpy and scipy/stats do majority of the work, the degrees of freedom must be found manually. Code for this was sourced from https://pythonfordatascience.org/welch-t-test-python-pandas/.

In [9]:
```
# This method performs the rank transformation that we need
# when we "fail" the Shapiro-Wilk test.
# Note that we wont use this function directly--only through other methods
def createRank(array1,array2):
    rankingArea = []
    rankedArray1 = []
    rankedArray2 = []
    for x in range(0,len(array1)):
        rankingArea.append(array1[x])
    for y in range(0,len(array2)):
        rankingArea.append(array2[y])
    rankingArea.sort()
    for z in range(0,len(rankingArea)):
        if(rankingArea[z] in array1):
            rankedArray1.append(z+1)
        else:
            rankedArray2.append(z+1)
    print('Creating rankings...')
    return [rankedArray1, rankedArray2]


# Python has all the methods built in for the two tailed
# t-test. Yay! Assumptions include:
    # Normally distributed
    # Categorical independent variable
    # Numeric dependent variable
# Note that we wont use this function directly--only through other methods.
def welch_ttest(x, y):
    ## Welch-Satterthwaite Degrees of Freedom ##
    dof = (np.var(x)/len(x) + np.var(y)/len(y))**2 / ((np.var(x)/len(x))**2 / (len(x)-1

    t, p = stats.ttest_ind(x, y, equal_var = False)

    print("\n",
          f"Welch's t-test= {t:.4f}", "\n",
          f"p-value = {p:.4f}", "\n",
          f"Welch-Satterthwaite Degrees of Freedom= {dof:.4f}")

# Python also has the function that performs the Shapiro-Wilk
# test! Basically tests if your sample comes from a normal
# distribution.
def testNormality(array1, array2):
    arrTest1 = stats.shapiro(array1)
    arrTest2 = stats.shapiro(array2)
```

```
    if(arrTest1[1]<0.05 or arrTest2[1]<0.05):
        print('May violate normal distribution assumption, or too small sample size!')
        if(arrTest1[1]<0.05):
            print(arrTest1)
            print('The values of the non-coastal districts are to blame!')
        if(arrTest2[1]<0.05):
            print(arrTest2)
            print('The values of the coastal districts are to blame!')

# The main code that we'll be using a lot of. It uses
# all of the methods above to perform the Welch t-test
# on a set of coastal/non-coastal districts. The states
# must be defined as they are in the cell before this one.
def welchtTest(state, ppType, ranked):
    selectedType = 5

    # The 'ppType' parameter chooses what data it will use.
    # 0 for weighted, 1 for unweighted
    if(ppType ==0 ):
        selectedType = 19
        print('Discrete Weighted Score Selected!')
    elif(ppType ==1):
        selectedType = 25
        print('Discrete Unweighted Score Selected!')
    else:
        selectedType = 25
        print('Discrete Unweighted Score Selected!')

    # Creates a smaller table/dataframe with just the data
    # concerning the inputted state.
    stateTable = bigTable[bigTable['state']==state[0]]
    coastalList = []
    nonCoastalList = []
    counter = stateTable.shape[0]

    # Creates two lists based on the defined coastal district
    # and the selected data type
    for x in range(0,counter):
        if(stateTable.iloc[x,1] in state[1]):
            coastalList.append(stateTable.iloc[x,selectedType])
        else:
            nonCoastalList.append(stateTable.iloc[x,selectedType])

    # Makes sure that the numbers are floating points so that
    # we can do math with them!
    nclFloat = [float(j) for j in nonCoastalList]
    clFloat = [float(i) for i in coastalList]
```

```
    # The 'ranked' parameter lets you use the rank transformation
    # on the data before performing the Welch t-test.
    if (ranked):
        rankedArrays = createRank(nclFloat,clFloat)
        testNormality(rankedArrays[0],rankedArrays[1])
        welch_ttest(rankedArrays[0],rankedArrays[1])
    else:
        testNormality(nclFloat,clFloat)
        welch_ttest(nclFloat,clFloat)
    print("")

# This does nearly the same thing as the above
# code, but uses the other large table to compare
# continuous Poslby-Popper scores
def welchtTestZoom(state, ppType, ranked):
    selectedType = 5
    if(ppType ==0 ):
        selectedType = 7
        print('TIGER/Line Score Selected!')
    elif(ppType ==1):
        selectedType = 8
        print('1:500,000 Score Selected!')
    elif(ppType==2):
        selectedType = 9
        print('1:5,000,000 Score Selected!')
    else:
        selectedType = 10
        print('1:20,000,000 Score Selected!')

    stateTable = zoomTable[zoomTable['state']==state[0]]
    coastalList = []
    nonCoastalList = []
    counter = stateTable.shape[0]
    for x in range(0,counter):
        if(float(stateTable.iloc[x,1]) in state[1]):
            coastalList.append(stateTable.iloc[x,selectedType])
        else:
            nonCoastalList.append(stateTable.iloc[x,selectedType])
    nclFloat = [float(j) for j in nonCoastalList]
    clFloat = [float(i) for i in coastalList]
    if (ranked):
        rankedArrays = createRank(nclFloat,clFloat)
        testNormality(rankedArrays[0],rankedArrays[1])
        welch_ttest(rankedArrays[0],rankedArrays[1])
    else:
        testNormality(nclFloat,clFloat)
        welch_ttest(nclFloat,clFloat)
    print("")
```

1.6 Performing analysis (on California)

Now we can finally use these functions to analyze the data. The parameters of the welchtTest and welchtTestZoom functions denote what census unit we use to create the dual-graph, or what type of map we want to analyze respectively.

From here, we walk through the example given in the textbook by performing the tests on the continuous scores, and then on the discrete scores.

1.6.1 Tests for continuous scores:

```
In [8]: tigerLine = welchtTestZoom(california, 0, False)
        fiveHundredK = welchtTestZoom(california, 1, False)
        fiveM = welchtTestZoom(california, 2, False)
        twentyM = welchtTestZoom(california, 3, False)

TIGER/Line Score Selected!

 Welch's t-test= 0.2621
 p-value = 0.7945
 Welch-Satterthwaite Degrees of Freedom= 42.2170

1:500,000 Score Selected!

 Welch's t-test= 1.9053
 p-value = 0.0630
 Welch-Satterthwaite Degrees of Freedom= 45.9869

1:5,000,000 Score Selected!

 Welch's t-test= 1.6570
 p-value = 0.1048
 Welch-Satterthwaite Degrees of Freedom= 43.0175

1:20,000,000 Score Selected!

 Welch's t-test= 2.2568
 p-value = 0.0292
 Welch-Satterthwaite Degrees of Freedom= 43.1056
```

We see that there are no warnings concerning the Shapiro-Wilk test for normality, so we don't need to run the continuous scores through the rank transform. The data above is also in textbook as a table.

1.6.2 Tests for discrete scores:

```
In [10]: weighted = welchtTest(california, 0, False)
         unweighted = welchtTest(california, 1, False)
```

```
Discrete Weighted Score Selected!
May violate normal distribution assumption, or too small sample size!
(0.5803228616714478, 2.1440062170086094e-08)
The values of the non-coastal districts are to blame!
(0.5573885440826416, 7.203976792879985e-07)
The values of the coastal districts are to blame!

 Welch's t-test= -1.3322
 p-value = 0.1949
 Welch-Satterthwaite Degrees of Freedom= 24.9196

Discrete Unweighted Score Selected!
May violate normal distribution assumption, or too small sample size!
(0.7946770191192627, 3.242444290663116e-05)
The values of the non-coastal districts are to blame!
(0.7693633437156677, 0.00023136104573495686)
The values of the coastal districts are to blame!

 Welch's t-test= -2.1229
 p-value = 0.0438
 Welch-Satterthwaite Degrees of Freedom= 25.1432
```

As mentioned in the textbook, we see that the p-values for the Shapiro-Wilk test indicate that the scores do not have an underlying normal distribution. As such, we perform the tests again, this time including the rank transform before performing the Welch t-test.

1.6.3 Tests for discrete scores with rank transform:

```
In [11]: weightedRanked = welchtTest(california, 0, True)
         unweightedRanked = welchtTest(california, 1, True)

Discrete Weighted Score Selected!
Creating rankings...

 Welch's t-test= -1.2831
 p-value = 0.2067
 Welch-Satterthwaite Degrees of Freedom= 41.3223

Discrete Unweighted Score Selected!
Creating rankings...

 Welch's t-test= -2.3142
 p-value = 0.0256
 Welch-Satterthwaite Degrees of Freedom= 42.2055
```

As we see, the Welch t-test returns negative values for both the weighted and unweighted scores, with a significant p-value for the unweighted scores.

1.7 Conclusion and Future Testing

This notebook was created to assist with simple exploratory data analysis concerning potential discrete analogs of coastal effects. One may use the following empty cells to perform their own tests with the other predefined states to continue the EDA. More comprehensive tests with more manual categorization of districts based on natural boundaries should be done to assure that the analog effect exists and to find the underlying reasons for them.