D207 PA v2

September 12, 2021

[1]:
```
import pandas as pd
import numpy as np # not used
from scipy.stats import ttest_ind
import seaborn as sns
import matplotlib.pyplot as plt
```

[2]:
```
pd.options.mode.chained_assignment = None  # default='warn' ---- ignores false␣
↪warning for database writing
```

[3]:
```
# write csv into datafile
df = pd.read_csv('medical_clean.csv')
```

[4]:
```
# create datafile of selected columns
newdf = df[['Area', 'Complication_risk', 'Initial_days', 'TotalCharge',␣
↪'Additional_charges']].copy()
```

[5]:
```
# check for duplicates
data = newdf.loc[newdf.duplicated()]
print(data)
```

```
Empty DataFrame
Columns: [Area, Complication_risk, Initial_days, TotalCharge,
Additional_charges]
Index: []
```

[6]:
```
#check for null values in each column
print(newdf.isnull().sum())
```

```
Area                 0
Complication_risk    0
Initial_days         0
TotalCharge          0
Additional_charges   0
dtype: int64
```

[7]:
```
# Total charge is cost per day as stated in the pdf file, so total cost is the␣
↪amount of days in hospital, multiplied by the total charge per day, and added␣
↪to the additional charges.
```

```
newdf['Total_cost'] = newdf['Initial_days'] * newdf['TotalCharge'] +␣
↪newdf['Additional_charges']
```

[8]:
```
print(newdf.head())
print(newdf['Area'].head())
```

```
       Area Complication_risk  Initial_days  TotalCharge  Additional_charges  \
0  Suburban             Medium     10.585770  3726.702860         17939.403420
1     Urban               High     15.129562  4193.190458         17612.998120
2  Suburban             Medium      4.772177  2434.234222         17505.192460
3  Suburban             Medium      1.714879  2127.830423         12993.437350
4     Rural                Low      1.254807  2113.073274          3716.525786

     Total_cost
0  57389.421674
1  81054.134013
2  29121.789533
3  16642.409430
4   6368.025351
0    Suburban
1       Urban
2    Suburban
3    Suburban
4       Rural
Name: Area, dtype: object
```

[9]:
```
# create 3 tables that are suburban, urban, and rural that contain the costs
rural_df = newdf.groupby(newdf['Area']).get_group('Rural')
suburban_df = newdf.groupby(newdf['Area']).get_group('Suburban')
urban_df = newdf.groupby(newdf['Area']).get_group('Urban')
```

[10]:
```
print(rural_df.describe())
print(suburban_df.describe())
print(urban_df.describe())
```

```
       Initial_days  TotalCharge  Additional_charges     Total_cost
count   3369.000000  3369.000000         3369.000000    3369.000000
mean      34.064556  5275.513171        12861.865881  249565.561693
std       26.407198  2186.098037         6560.279403  221013.454263
min        1.009143  1957.445547         3132.259990    6305.195646
25%        7.795208  3171.353946         7935.568053   39046.761561
50%       24.498128  4619.773928        11478.667310  125272.797554
75%       61.317820  7476.964000        15512.552780  472706.120543
max       71.981490  9169.248000        30566.070000  665793.196261
       Initial_days  TotalCharge  Additional_charges     Total_cost
count   3328.000000  3328.000000         3328.000000    3328.000000
mean      34.443709  5310.764071        13090.248254  252677.948135
```

```
std       26.322122  2180.012708         6595.586519  220510.852254
min        1.001981  1938.312067         3125.703000    5857.219178
25%        7.883798  3167.008282         8109.785987   38189.742528
50%       36.467510  5243.429000        11659.204390  206711.866906
75%       61.031492  7442.691000        16223.397577  468996.115253
max       71.964150  9067.605000        30395.025240  666043.264215
       Initial_days  TotalCharge  Additional_charges     Total_cost
count   3303.000000  3303.000000         3303.000000    3303.000000
mean      34.865529  5350.984253        12851.744938  255668.239594
std       26.198032  2174.945084         6469.715801  219776.153800
min        1.012481  2022.650007         3139.049369    6666.580482
25%        8.028876  3187.025909         7915.033978   39272.405808
50%       38.656840  5470.604000        11594.081770  228389.853451
75%       61.098875  7472.365500        15354.995000  467660.273764
max       71.961640  9180.728000        30466.930000  673944.437621
```

[11]:
```
print(rural_df.head())
print(suburban_df.head())
print(urban_df.head())
```

```
     Area Complication_risk  Initial_days  TotalCharge  Additional_charges  \
4   Rural               Low      1.254807  2113.073274         3716.525786
6   Rural               Low      9.058210  3694.627161        16815.513600
18  Rural            Medium      7.302395  2698.883482        23453.358310
23  Rural            Medium     11.644075  3715.033085        12876.791960
24  Rural               Low     18.874241  4300.035326         3624.049387

      Total_cost
4    6368.025351
6   50282.223534
18  43161.670240
23  56134.914380
24  84783.953427
        Area Complication_risk  Initial_days  TotalCharge  Additional_charges  \
0   Suburban            Medium     10.585770  3726.702860        17939.403420
2   Suburban            Medium      4.772177  2434.234222        17505.192460
3   Suburban            Medium      1.714879  2127.830423        12993.437350
11  Suburban              High      7.075083  3166.627638        21480.886130
13  Suburban              High      2.020142  3186.814113         8454.613363

      Total_cost
0   57389.421674
2   29121.789533
3   16642.409430
11  43885.040452
13  14892.431074
    Area Complication_risk  Initial_days  TotalCharge  Additional_charges  \
1  Urban              High     15.129562  4193.190458        17612.998120
```

```
5  Urban            Medium      5.957250  2636.691180        12742.589910
7  Urban            Medium     14.228019  3021.499039         6930.572138
8  Urban               Low      6.180339  2968.402860         8363.187290
9  Urban              High      1.632554  3147.855813        26225.989910

     Total_cost
1  81054.134013
5  28450.018018
7  49920.518115
8  26708.924310
9  31365.033788
```

[12]:
```
# note: newdf is whole datafile, rural_df, suburban_df, and urban_df are sorted␣
 ↪by area
# use t test to compare averages between all 3 rural-suburban, rural-urban,␣
 ↪suburban-urban
ruralAverage = rural_df['Total_cost'].mean()
suburbanAverage = suburban_df['Total_cost'].mean()
urbanAverage = urban_df['Total_cost'].mean()
```

[13]:
```
# print averages, turns out they are all pretty close
print(ruralAverage, suburbanAverage, urbanAverage)
```

```
249565.56169311932 252677.94813519763 255668.23959423244
```

[14]:
```
# calculates t test values of the areas total cost ratios.

resRuralSuburban = ttest_ind(rural_df['Total_cost'], suburban_df['Total_cost'])
resSuburbanUrban = ttest_ind(suburban_df['Total_cost'], urban_df['Total_cost'])
resUrbanRural = ttest_ind(urban_df['Total_cost'], rural_df['Total_cost'])
print('Rural/Suburban t-test:', resRuralSuburban)
print('Suburban/Urban t-test:', resSuburbanUrban)
print('Urban/Rural t-test:', resUrbanRural)
```

```
Rural/Suburban t-test: Ttest_indResult(statistic=-0.5768562459594408,
pvalue=0.5640559644891201)
Suburban/Urban t-test: Ttest_indResult(statistic=-0.5530453453925396,
pvalue=0.580251009402408)
Urban/Rural t-test: Ttest_indResult(statistic=1.130790359515346,
pvalue=0.25818399599366887)
```

[15]:
```
# 2 continuous and 2 categorical: Total Cost and Initial Days as continuous,␣
 ↪Area and Complication risk as categorical

di = {'Rural': 1, 'Suburban': 2, 'Urban': 3}
di2 = {'Low': 1, 'Medium': 2, 'High': 3}
newdf_numeric = newdf.replace({'Area': di, 'Complication_risk': di2})
```

```
print(newdf_numeric.head())
```

```
   Area  Complication_risk  Initial_days  TotalCharge  Additional_charges  \
0     2                  2     10.585770  3726.702860        17939.403420
1     3                  3     15.129562  4193.190458        17612.998120
2     2                  2      4.772177  2434.234222        17505.192460
3     2                  2      1.714879  2127.830423        12993.437350
4     1                  1      1.254807  2113.073274         3716.525786

     Total_cost
0  57389.421674
1  81054.134013
2  29121.789533
3  16642.409430
4   6368.025351
```

[16]:
```
analysis_df = newdf_numeric[['Area', 'Complication_risk', 'Initial_days',␣
 ↪'Total_cost']]
analysis_df.hist()
plt.savefig('hospital_pyplot.jpg')
plt.close()
```

[17]:
```
sns.heatmap(analysis_df.corr(), annot=True)
plt.savefig('heatmap.jpg', bbox_inches='tight')
plt.close()
```