schulzdLab3

January 4, 2021

1 Lab 3 - Exploratory Data Analysis with Statistical Testing

David Schulz

1.1 Introduction

In the last two labs, we cleaned a data set of real estate transactions and then used visualizations to explore the relationships between the independent (input) and dependent (output) variables for two machine learning tasks. In this lab, we are going to test for correlation and association between the independent and dependent variables using statistical tests.

1.2 Part I: Review of Statistical Tests

- Hypothesis: Students who play video games regularly have a lower GPA than students who do not play video games.
- The t-test is used when there is one nominal variable and one measurement variable and you want to compare the means of the measurement variable. The nominal variable must also only have two possible values. This situation does meet those criteria. The assumption that these data sets are normally distributed must be made. The test also assumes that the two total populations from the groups have the same variance.
- Null Hypothesis: The mean GPA for students who play video games is equal to the mean GPA for students who don't.
- Alternative Hypothesis: The mean GPA for students who play video games is less than the mean GPA for students who don't.

[1]:
```
from scipy.stats import ttest_ind_from_stats

stat, pvalue = ttest_ind_from_stats(3.4, 1.2, 68, 3.3, 1.1, 32,␣
 ↪alternative='less')
print("p-value: " + str(pvalue))
```

```
p-value: 0.6545968708463726
```

- Since the p-value is not less than 0.01, we cannot reject the null hypothesis, so the differences are not statistically significant.
- The evidence from the statistical test fails to prove that students who play video games regularly have a lower GPA than students who don't.

1.3 Part II: Exploring Additional Statistical Tests

- Test for Correlation Based on Linear Regression
  - Variables: Two measurements
  - Assumptions: Normality, homoscedasticity, linearity, independence
  - Null Hypothesis: The slope of the best-fit line is equal to zero.
  - Alternative Hypothesis: The slope of the best-fit line is (greater than/less than/not equal to) zero.
  - If the test indicated statistical significance, the measurement variables are associated with each other.
  - If the test did not indicate statistical significance, the measurement variables may or may not be associated with each other.
- Kruskal-Wallis Test
  - Variables: One nominal and one measurement
  - Assumptions: The observations in each group come from populations with the same shape of distribution
  - Null Hypothesis: The mean ranks of the groups are the same.
  - Alternative Hypothesis: The mean ranks of the groups are not the same.
  - If the test indicated statistical significance, the mean ranks are not the same in all the groups.
  - If the test did not indicate statistical significance, the mean ranks might or might not be the same in all the groups.
- Chi-squared Test of Goodness of Fit
  - Variables: One nominal with two or more possible values
  - Assumptions: Independence
  - Null Hypothesis: The number of observations in each category is equal to a predicted value.
  - Alternative Hypothesis: The number of observations in each category is not equal to a predicted value.
  - If the test indicated statistical significance, the number of observations in each category is not equal to a predicted value.
  - If the test did not indicate statistical significance, the number of observations in each category might or might not be equal to a predicted value.

1.4 Part III: Regression on Price

[2]:
```
import pandas as pd

data = pd.read_csv('Sacramentorealestatetransactions.csv')
price = data['price']
data = data.drop('price', axis=1)

print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 985 entries, 0 to 984
Data columns (total 13 columns):
```

```
street          985 non-null object
city            985 non-null object
zip             985 non-null int64
state           985 non-null object
beds            985 non-null int64
baths           985 non-null int64
sq__ft          985 non-null int64
type            985 non-null object
sale_date       985 non-null object
latitude        985 non-null float64
longitude       985 non-null float64
empty_lot       985 non-null bool
street_type     985 non-null object
dtypes: bool(1), float64(2), int64(4), object(6)
memory usage: 93.4+ KB
None
```

[3]:
```
import scipy.stats as stats

slope, intercept, r, p, stderr = stats.linregress(price, data['sq__ft'])
print("Square Feet: " + str(p))
slope, intercept, r, p, stderr = stats.linregress(price, data['latitude'])
print("Latitude: " + str(p))
slope, intercept, r, p, stderr = stats.linregress(price, data['longitude'])
print("Longitude: " + str(p))
```

```
Square Feet: 4.433056844561304e-27
Latitude: 0.2146410657697643
Longitude: 8.552356644184264e-20
```

Variable	p-value	Statistically Significant?
Square Feet	4.433 * 10^-27	Yes
Latitude	0.215	No
Longitude	8.552 * 10^-20	Yes

[5]:
```
samples_by_group = []
for value in set(data["street"]):
    mask = data["street"] == value
    samples_by_group.append(price[mask])
stat, p = stats.kruskal(*samples_by_group)
print("Street: " + str(p))

samples_by_group = []
for value in set(data["city"]):
    mask = data["city"] == value
    samples_by_group.append(price[mask])
```

```
stat, p = stats.kruskal(*samples_by_group)
print("City: " + str(p))

samples_by_group = []
for value in set(data["zip"]):
    mask = data["zip"] == value
    samples_by_group.append(price[mask])
stat, p = stats.kruskal(*samples_by_group)
print("Zip Code: " + str(p))

# Can't do state because it's only one group

samples_by_group = []
for value in set(data["beds"]):
    mask = data["beds"] == value
    samples_by_group.append(price[mask])
stat, p = stats.kruskal(*samples_by_group)
print("Beds: " + str(p))

samples_by_group = []
for value in set(data["baths"]):
    mask = data["baths"] == value
    samples_by_group.append(price[mask])
stat, p = stats.kruskal(*samples_by_group)
print("Baths: " + str(p))

samples_by_group = []
for value in set(data["type"]):
    mask = data["type"] == value
    samples_by_group.append(price[mask])
stat, p = stats.kruskal(*samples_by_group)
print("Type: " + str(p))

samples_by_group = []
for value in set(data["sale_date"]):
    mask = data["sale_date"] == value
    samples_by_group.append(price[mask])
stat, p = stats.kruskal(*samples_by_group)
print("Sale Date: " + str(p))

samples_by_group = []
for value in set(data["empty_lot"]):
    mask = data["empty_lot"] == value
    samples_by_group.append(price[mask])
stat, p = stats.kruskal(*samples_by_group)
print("Empty Lot: " + str(p))
```

```
samples_by_group = []
for value in set(data["street_type"]):
    mask = data["street_type"] == value
    samples_by_group.append(price[mask])
stat, p = stats.kruskal(*samples_by_group)
print("Street Type: " + str(p))
```

```
Street: 0.4580759804602128
City: 8.588143361577209e-49
Zip Code: 2.2206936511391717e-65
Beds: 9.323012566322206e-38
Baths: 1.1109110373679834e-50
Type: 1.1531292565464816e-06
Sale Date: 1.3914349696353547e-14
Empty Lot: 0.03803941698702659
Street Type: 3.1544959263053637e-16
```

Variable	p-value	Statistically Significant?
Street	0.458	No
City	8.588 * 10^-49	Yes
Zip Code	2.221 * 10^-65	Yes
State	N/A	No
Beds	9.323 * 10^-38	Yes
Baths	1.111 * 10^-50	Yes
Type	1.153 * 10^-06	Yes
Sale Date	1.391 * 10^-14	Yes
Empty Lot	0.038	No
Street Type	3.154 * 10^-16	Yes

Apparently I didn't work with some of the categorical variables in lab 2, possibly because there were just too many groups for them to be considered categorical or too many groups to be able to read the visualization when they were all graphed. For the variables I did use, my responses for square footage, latitude, zip code, state, beds, and baths were the same as the test results, and my responses for longitude, city, and type were different.

1.5 Part IV: Classification on Property Type

[6]:
```
samples_by_group = []
for value in set(data["type"]):
    mask = data["type"] == value
    samples_by_group.append(data["sq__ft"][mask])
stat, p = stats.kruskal(*samples_by_group)
print("Square Feet: " + str(p))

samples_by_group = []
```

```
for value in set(data["type"]):
    mask = data["type"] == value
    samples_by_group.append(data["latitude"][mask])
stat, p = stats.kruskal(*samples_by_group)
print("Latitude: " + str(p))

samples_by_group = []
for value in set(data["type"]):
    mask = data["type"] == value
    samples_by_group.append(data["longitude"][mask])
stat, p = stats.kruskal(*samples_by_group)
print("Longitude: " + str(p))
```

```
Square Feet: 3.849635133463987e-12
Latitude: 0.493369613879038
Longitude: 0.3574295072205312
```

Variable	p-value	Statistically Significant?
Square Feet	3.850 * 10^-12	Yes
Latitude	0.493	No
Longitude	0.357	No

[11]:
```
combination_counts = data[["type","street"]].groupby(by=["type","street"]).
 ↪size().unstack(level=0).fillna(0)
chi2, p, _, _ = stats.chi2_contingency(combination_counts)
print("Street: " + str(p))

combination_counts = data[["type","city"]].groupby(by=["type","city"]).size().
 ↪unstack(level=0).fillna(0)
chi2, p, _, _ = stats.chi2_contingency(combination_counts)
print("City: " + str(p))

combination_counts = data[["type","zip"]].groupby(by=["type","zip"]).size().
 ↪unstack(level=0).fillna(0)
chi2, p, _, _ = stats.chi2_contingency(combination_counts)
print("Zip Code: " + str(p))

combination_counts = data[["type","state"]].groupby(by=["type","state"]).size().
 ↪unstack(level=0).fillna(0)
chi2, p, _, _ = stats.chi2_contingency(combination_counts)
print("State: " + str(p))

combination_counts = data[["type","beds"]].groupby(by=["type","beds"]).size().
 ↪unstack(level=0).fillna(0)
chi2, p, _, _ = stats.chi2_contingency(combination_counts)
```

```
print("Beds: " + str(p))

combination_counts = data[["type","baths"]].groupby(by=["type","baths"]).size().
 ↪unstack(level=0).fillna(0)
chi2, p, _, _ = stats.chi2_contingency(combination_counts)
print("Baths: " + str(p))

combination_counts = data[["type","sale_date"]].
 ↪groupby(by=["type","sale_date"]).size().unstack(level=0).fillna(0)
chi2, p, _, _ = stats.chi2_contingency(combination_counts)
print("Sale Date: " + str(p))

combination_counts = data[["type","empty_lot"]].
 ↪groupby(by=["type","empty_lot"]).size().unstack(level=0).fillna(0)
chi2, p, _, _ = stats.chi2_contingency(combination_counts)
print("Empty Lot: " + str(p))

combination_counts = data[["type","street_type"]].
 ↪groupby(by=["type","street_type"]).size().unstack(level=0).fillna(0)
chi2, p, _, _ = stats.chi2_contingency(combination_counts)
print("Street Type: " + str(p))
```

```
Street: 0.41918692100156474
City: 6.884142190454073e-149
Zip Code: 6.195871832483889e-06
State: 1.0
Beds: 1.6090894896930032e-64
Baths: 2.225837221388893e-41
Sale Date: 0.20367568979378245
Empty Lot: 0.03857273017762252
Street Type: 2.906579353314612e-15
```

Variable	p-value	Statistically Significant?
Street	0.419	No
City	6.884 * 10^-149	Yes
Zip Code	6.196 * 10^-06	Yes
State	1.0	No
Beds	1.609 * 10^-64	Yes
Baths	2.226 * 10^-41	Yes
Sale Date	0.204	No
Empty Lot	0.039	No
Street Type	2.907 * 10^-15	Yes

Again, I didn't work with some of the categorical variables in lab 2, probably for the same reasons. For the variables I did use, my responses for square footage, latitude, longitude, city, zip code, beds, and baths were the same as the test results. My response for state was different, but only

because I misunderstood what was being predicted in lab 2.