Test a Perceptual Phenomenon

February 10, 2019

0.0.1 Analyzing the Stroop Effect

Perform the analysis in the space below. Remember to follow the instructions and review the project rubric before submitting. Once you've completed the analysis and write-up, download this file as a PDF or HTML file, upload that PDF/HTML into the workspace here (click on the orange Jupyter icon in the upper left then Upload), then use the Submit Project button at the bottom of this page. This will create a zip file containing both this .ipynb doc and the PDF/HTML doc that will be submitted for your project.

(1) What is the independent variable? What is the dependent variable?

Independent Variable: matching font color and word name or different.
Dependent Variable: The duration of time that participant take to indicate the color.

In [1]:
```
import  pandas as  pd
path  =  r'stroopdata.csv'
data = pd.read_csv(path)
data
```

Out[1]:

	Congruent	Incongruent
0	12.079	19.278
1	16.791	18.741
2	9.564	21.214
3	8.630	15.687
4	14.669	22.803
5	12.238	20.878
6	14.692	24.572
7	8.987	17.394
8	9.401	20.762
9	14.480	26.282
10	22.328	24.524
11	15.298	18.644
12	15.073	17.510
13	16.929	20.330
14	18.200	35.255
15	12.130	22.158
16	18.495	25.139
17	10.639	20.429

```
18      11.344        17.425
19      12.369        34.288
20      12.944        23.894
21      14.233        17.960
22      19.710        22.058
23      16.004        21.157
```

(2) What is an appropriate set of hypotheses for this task? Specify your null and alternative hypotheses, and clearly define any notation used. Justify your choices.

The sample has n=24 with the recognition times of congruent and incongruent data. The null hypothesis is tested that the true mean difference is zero between the two data sets.

Null hypotheses (H_0): There is no difference in time needed to read a congruent displayed text to a incongruent displayed text.

Alternative hypotheses (H_A): It takes a different time to read an incongruent displayed text than to read a congruent displayed text.

$$\mu_{congruent} - \mu_{incongruent} = \mu_{difference}$$
$$H_0 : \mu_{difference} = 0$$
$$H_A : \mu_{difference} \neq 0$$
$$\mu = populationaverage$$
$$\bar{d} = sampleaverage$$

t-test is performed because our sample size is smaller than 30 and the population standard deviation is unknown. Two-tailed paired t-test performed, to test if there is a difference in the reaction time.

(3) Report some descriptive statistics regarding this dataset. Include at least one measure of central tendency and at least one measure of variability. The name of the data file is 'stroop-data.csv'.

Measures of Central

```
In [2]: data['Congruent'].mean()
```

```
Out[2]: 14.051124999999999
```

```
In [3]: data['Congruent'].median()
```

```
Out[3]: 14.3565
```

```
In [4]: data['Congruent'].mode()
```

```
Out[4]: 0      8.630
        1      8.987
        2      9.401
        3      9.564
        4     10.639
```

```
5      11.344
6      12.079
7      12.130
8      12.238
9      12.369
10     12.944
11     14.233
12     14.480
13     14.669
14     14.692
15     15.073
16     15.298
17     16.004
18     16.791
19     16.929
20     18.200
21     18.495
22     19.710
23     22.328
dtype: float64
```

```
In [5]: data['Incongruent'].mean()
```

```
Out[5]: 22.015916666666666
```

```
In [6]: data['Incongruent'].median()
```

```
Out[6]: 21.0175
```

```
In [7]: data['Incongruent'].mode()
```

```
Out[7]: 0      15.687
        1      17.394
        2      17.425
        3      17.510
        4      17.960
        5      18.644
        6      18.741
        7      19.278
        8      20.330
        9      20.429
        10     20.762
        11     20.878
        12     21.157
        13     21.214
        14     22.058
        15     22.158
        16     22.803
        17     23.894
```

```
18    24.524
19    24.572
20    25.139
21    26.282
22    34.288
23    35.255
dtype: float64
```

Measures of variability

In [8]:
```
#calculate the standard deviation
data.std()
```

Out[8]:
```
Congruent      3.559358
Incongruent    4.797057
dtype: float64
```

In [9]:
```
# calculate outlieres
# make a copy of original dataframe
newdata = data.copy()

newdata['CongruentDeviationFromMean'] = abs(newdata['Congruent'] - newdata['Congruent'].
newdata['IsCongruentOutlier'] = abs(newdata['Congruent'] - newdata['Congruent'].mean())
newdata['CongruentSquareDevation'] = abs(newdata['Congruent'] - newdata['Congruent'].mea

newdata['IncongruentDeviationFromMean'] = abs(newdata['Incongruent'] - newdata['Incongru
newdata['IsIncongruentOutlier'] = abs(newdata['Incongruent'] - newdata['Incongruent'].me
newdata['IncongruentSquareDevation'] = abs(newdata['Incongruent'] - newdata['Incongruent

newdata
```

Out[9]:

	Congruent	Incongruent	CongruentDeviationFromMean	IsCongruentOutlier \
0	12.079	19.278	1.972125	False
1	16.791	18.741	2.739875	False
2	9.564	21.214	4.487125	False
3	8.630	15.687	5.421125	False
4	14.669	22.803	0.617875	False
5	12.238	20.878	1.813125	False
6	14.692	24.572	0.640875	False
7	8.987	17.394	5.064125	False
8	9.401	20.762	4.650125	False
9	14.480	26.282	0.428875	False
10	22.328	24.524	8.276875	True
11	15.298	18.644	1.246875	False
12	15.073	17.510	1.021875	False
13	16.929	20.330	2.877875	False
14	18.200	35.255	4.148875	False
15	12.130	22.158	1.921125	False
16	18.495	25.139	4.443875	False

17	10.639	20.429	3.412125	False
18	11.344	17.425	2.707125	False
19	12.369	34.288	1.682125	False
20	12.944	23.894	1.107125	False
21	14.233	17.960	0.181875	False
22	19.710	22.058	5.658875	False
23	16.004	21.157	1.952875	False

	CongruentSquareDevation	IncongruentDeviationFromMean	\
0	3.889277	2.737917	
1	7.506915	3.274917	
2	20.134291	0.801917	
3	29.388596	6.328917	
4	0.381770	0.787083	
5	3.287422	1.137917	
6	0.410721	2.556083	
7	25.645362	4.621917	
8	21.623663	1.253917	
9	0.183934	4.266083	
10	68.506660	2.508083	
11	1.554697	3.371917	
12	1.044229	4.505917	
13	8.282165	1.685917	
14	17.213164	13.239083	
15	3.690721	0.142083	
16	19.748025	3.123083	
17	11.642597	1.586917	
18	7.328526	4.590917	
19	2.829545	12.272083	
20	1.225726	1.878083	
21	0.033079	4.055917	
22	32.022866	0.042083	
23	3.813721	0.858917	

	IsIncongruentOutlier	IncongruentSquareDevation
0	False	7.496188
1	False	10.725079
2	False	0.643070
3	False	40.055186
4	False	0.619500
5	False	1.294854
6	False	6.533562
7	False	21.362114
8	False	1.572307
9	False	18.199467
10	False	6.290482
11	False	11.369822
12	False	20.303285

```
13                  False                   2.842315
14                   True                 175.273328
15                  False                   0.020188
16                  False                   9.753650
17                  False                   2.518305
18                  False                  21.076516
19                   True                 150.604029
20                  False                   3.527197
21                  False                  16.450460
22                  False                   0.001771
23                  False                   0.737738
```

```
In [10]: print(data.describe())
```

```
       Congruent  Incongruent
count  24.000000    24.000000
mean   14.051125    22.015917
std     3.559358     4.797057
min     8.630000    15.687000
25%    11.895250    18.716750
50%    14.356500    21.017500
75%    16.200750    24.051500
max    22.328000    35.255000
```

```
In [11]: congruentvariance = ((newdata.CongruentSquareDevation).sum())/(newdata['Congruent'].cou
         congruentvariance
```

```
Out[11]: 12.669029070652176
```

```
In [12]: incongruentvariance = ((newdata.IncongruentSquareDevation).sum())/(newdata['Incongruent
         incongruentvariance
```

```
Out[12]: 23.011757036231884
```

```
In [13]: #calc IQR for Congruent Data
         dfc = pd.DataFrame({'Congruent': data['Congruent']})
         dfc.sort_values('Congruent', inplace=True)

         Q1 = dfc['Congruent'].quantile(0.25)
         Q3 = dfc['Congruent'].quantile(0.75)
         IQR_Congruent = Q3 - Q1
         print ("Congruent Data")
         print ("Q1:", Q1)
         print ("Q3:", Q3)
         print ("IQR:", IQR_Congruent)
         Outlier = Q1-(1.5*IQR_Congruent)
         print (Outlier)
         Outlierabove = Q3 + (1.5*IQR_Congruent)
         print (Outlierabove)
```

```
Congruent Data
Q1: 11.89525
Q3: 16.20075
IQR: 4.3054999999999986
5.437000000000003
22.659
```

```
In [14]: #calc IQR for Incongruent Data
         dfi = pd.DataFrame({'Incongruent': data['Incongruent']})
         dfi.sort_values('Incongruent', inplace=True)

         Q1i = dfi['Incongruent'].quantile(0.25)
         Q3i = dfi['Incongruent'].quantile(0.75)
         IQR_Incongruent = Q3i - Q1i

         print ("Incongruent Data")
         print ("Q1:", Q1i)
         print ("Q3:", Q3i)
         print ("IQR:", IQR_Incongruent)
         OutlierI = Q1i-(1.5*IQR_Incongruent)
         print (OutlierI)
         OutlieraboveI = Q3i+ (1.5*IQR_Incongruent)
         print (OutlieraboveI)
```

```
Incongruent Data
Q1: 18.71675
Q3: 24.0515
IQR: 5.33475
10.714625000000002
32.053625
```

Standard Deviation

$$\sigma_{congruent} = 3.55$$
$$\sigma_{incongruent} = 4.79$$

Variance

$$s^2 = \frac{\sum(x_i - \overline{x})^2}{n-1}$$
$$s^2_{congruent} = 12.66$$
$$s^2_{incongruent} = 23.01$$

Range

The range including outliers shows that the quickest reader for incongruent data needed double the time as the quickest reader from the congruent data, in Congruent Data range start from 8.63 to 22.32 seconds, where as the Incroguent Data from 15.69 to 35.26.

Inter-quartile Range

$$Q11.5\dot{I}QR \text{ or } Q3 + 1.5\dot{I}QR$$

$$IQR_{congruent} = 4.3055$$

$$IQR_{incongruent} = 5.3347$$

50% of the congruent Data differ 4.3055 seconds and the lie between the marks Q1: 11.89525 and Q3: 16.20075. While the incongruent Data 50% differ 5.3347 seconds and the lie between the marks 18.71675 Q3: 24.0515. This shows that the incongruent data takes more time to read then the congruent data, because the 25% (Q1) and the 75%(Q3) are higher and the spread is also bigger.

(4) Provide one or two visualizations that show the distribution of the sample data. Write one or two sentences noting what you observe about the plot or plots.

```
In [15]: #visualizations
         %pylab inline

         import matplotlib.pyplot as plt
         matplotlib.style.use('ggplot')

         #boxplot
         color = dict(boxes='grey', whiskers='blue', medians="DarkBlue", caps="green" )
         data.plot.box(color=color, sym="r+");
         ylabel('Time in Seconds')
         title('Boxplots for Stroop Data')
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
Out[15]: Text(0.5,1,'Boxplots for Stroop Data')
```