Name: Frances Aneth Rosales
Student Number: 2021044631
FA7

Click to direct line.

I. Introduction

Independent samples t-test for Eye Movementsdata set

Showing the Data

In [46]:

```
import pandas as pd

file_path = "Eye_Movements.xlsx"
xls = pd.ExcelFile(file_path)

data_sheet = pd.read_excel(file_path, sheet_name="Eye_Movements")

datframe_c = pd.DataFrame(data_sheet)
print(datframe_c)
```

	ParticipantNumber	Condition	CriticalRecall
0	1	Horizontal	4
1	3	Fixation	14
2	4	Horizontal	12
3	6	Fixation	4
4	7	Horizontal	11
5	9	Fixation	23
6	10	Horizontal	16
7	12	Fixation	22
8	13	Horizontal	9
9	15	Fixation	16
10	16	Horizontal	12
11	19	Horizontal	3
12	21	Fixation	10
13	22	Horizontal	11
14	24	Fixation	16
15	25	Horizontal	10
16	27	Fixation	18
17	28	Horizontal	8
18	30	Fixation	23
19	31	Horizontal	12
20	33	Fixation	10
21	34	Horizontal	6
22	36	Fixation	13
23	39	Fixation	23
24	42	Fixation	4
25	43	Horizontal	11
26	45	Fixation	23
27	46	Horizontal	16
28	48	Fixation	10
29	49	Horizontal	11
30	51	Fixation	18
31	52	Horizontal	8
32	54	Fixation	11
33	55	Horizontal	15
34	57	Fixation	11
35	58	Horizontal	10
36	60	Fixation	25
37	61	Horizontal	15
38	63	Fixation	9
39	64	Horizontal	14
40	66	Fixation	18
41	67	Horizontal	9
42	70	Horizontal	11
43	73	Horizontal	13
44	75	Fixation	11
45	76	Horizontal	3
46	78	Fixation	11
47	79	Horizontal	22
48	81	Fixation	24

a. CriticalRecall Summary Data

In [49]:

```
import numpy as np
from scipy import stats

data_sheet = pd.read_excel(file_path, sheet_name="Eye_Movements")
```

```
critical_recall = data_sheet["CriticalRecall"]

mean = np.mean(critical_recall)

mode = stats.mode(critical_recall).mode[0]

median = np.median(critical_recall)

std_dev = np.std(critical_recall)

variance = np.var(critical_recall)

skewness = stats.skew(critical_recall)
se_skewness = stats.sem(critical_recall)

kurtosis = stats.kurtosis(critical_recall)

q1 = np.percentile(critical_recall, 25)
q2 = np.percentile(critical_recall, 50)
q3 = np.percentile(critical_recall, 75)

d9 = np.percentile(critical_recall, 90)
p95 = np.percentile(critical_recall, 95)

ress = pd.DataFrame({
    "Summary Data": ["", "", "", "", "", "", "", "", "", "", "", "", "", ""],
    "Data": ["Valid", "Mode", "Mean", "Median", "Standard Deviation", "Variance"
    "Score": [len(critical_recall), mode, mean, median, std_dev, variance, skewn
})

ress["Score"] = ress["Score"].apply(lambda x: "{:.3f}".format(x))

print(ress)
```

```
   Summary Data                         Data   Score
0                                      Valid  49.000
1                                       Mode  11.000
2                                       Mean  13.041
3                                     Median  11.000
4                         Standard Deviation   5.753
5                                   Variance  33.100
6                                   Skewness   0.390
7               Standard Error of Skewness   0.830
8                                   Kurtosis  -0.490
9                                         Q1  10.000
10                                        Q2  11.000
11                                        Q3  16.000
12                                        D9  23.000
13                                       P95  23.000
```

```
C:\Users\asus\AppData\Local\Temp\ipykernel_15380\3632994806.py:10: FutureWarning:
Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behavior
of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this behav
ior will change: the default value of `keepdims` will become False, the `axis` ov
er which the statistic is taken will be eliminated, and the value None will no lo
nger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode = stats.mode(critical_recall).mode[0]
```

b. Horizontal & Fixation Summary Data

In [48]:

```
critical_recall = data_sheet["CriticalRecall"]
condition_col = data_sheet["Condition"]
horizontal_data = critical_recall[condition_col == "Horizontal"]
fixation_data = critical_recall[condition_col == "Fixation"]

def calculate_stats(data):
    return {
        "Valid": len(data),
        "Mode": stats.mode(data).mode[0],
        "Mean": np.mean(data),
        "Median": np.median(data),
        "Standard Deviation": np.std(data),
        "Variance": np.var(data),
        "Skewness": stats.skew(data),
        "Standard Error of Skewness": stats.sem(data),
        "Kurtosis": stats.kurtosis(data),
        "Q1": np.percentile(data, 25),
        "Q2": np.percentile(data, 50),
        "Q3": np.percentile(data, 75),
        "D9": np.percentile(data, 90),
        "P95": np.percentile(data, 95)
    }

horizontal_stats = calculate_stats(horizontal_data)
fixation_stats = calculate_stats(fixation_data)

results_df_horizontal = pd.DataFrame({
    "Summary Data": ["", "", "", "", "", "", "", "", "", "", "", "", "", ""],
    "Data": ["Valid", "Mode", "Mean", "Median", "Standard Deviation", "Variance"
    "Score": list(horizontal_stats.values())
})

results_df_fixation = pd.DataFrame({
    "Summary Data": ["", "", "", "", "", "", "", "", "", "", "", "", "", ""],
    "Data": ["Valid", "Mode", "Mean", "Median", "Standard Deviation", "Variance"
    "Score": list(fixation_stats.values())
})

results_df_horizontal["Score"] = results_df_horizontal["Score"].apply(lambda x:
results_df_fixation["Score"] = results_df_fixation["Score"].apply(lambda x: "{:.

print("Horizontal Condition Statistics:")
print(results_df_horizontal)

print("\nFixation Condition Statistics:")
print(results_df_fixation)
```

```
Horizontal Condition Statistics:
   Summary Data                          Data   Score
0                                       Valid  25.000
1                                        Mode  11.000
2                                        Mean  10.880
3                                      Median  11.000
4                          Standard Deviation   4.236
5                                    Variance  17.946
6                                    Skewness   0.181
7                  Standard Error of Skewness   0.865
8                                    Kurtosis   0.483
9                                          Q1   9.000
10                                         Q2  11.000
11                                         Q3  13.000
12                                         D9  15.600
13                                        P95  16.000

Fixation Condition Statistics:
   Summary Data                          Data   Score
0                                       Valid  24.000
1                                        Mode  11.000
2                                        Mean  15.292
3                                      Median  15.000
4                          Standard Deviation   6.242
5                                    Variance  38.957
6                                    Skewness  -0.023
7                  Standard Error of Skewness   1.301
8                                    Kurtosis  -1.112
9                                          Q1  10.750
10                                         Q2  15.000
11                                         Q3  22.250
12                                         D9  23.000
13                                        P95  23.850
```

```
C:\Users\asus\AppData\Local\Temp\ipykernel_15380\3234557549.py:9: FutureWarning:
Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behavior
of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this behav
ior will change: the default value of `keepdims` will become False, the `axis` ov
er which the statistic is taken will be eliminated, and the value None will no lo
nger be accepted. Set `keepdims` to True or False to avoid this warning.
  "Mode": stats.mode(data).mode[0],
```

Assumption Checks:

Assumption 1.

This makes the assumption that the data in the "CriticalRecall" column is numerical, enabling useful computations of central tendency and dispersion.

Assumption 2.

This is presuming that there are two separate categories in the "Condition" column, and that each participant belongs to either the "Horizontal" or the "Fixation" group. These classifications ought to stand for conditions that are exhaustive and mutually exclusive.

Assumption 3.

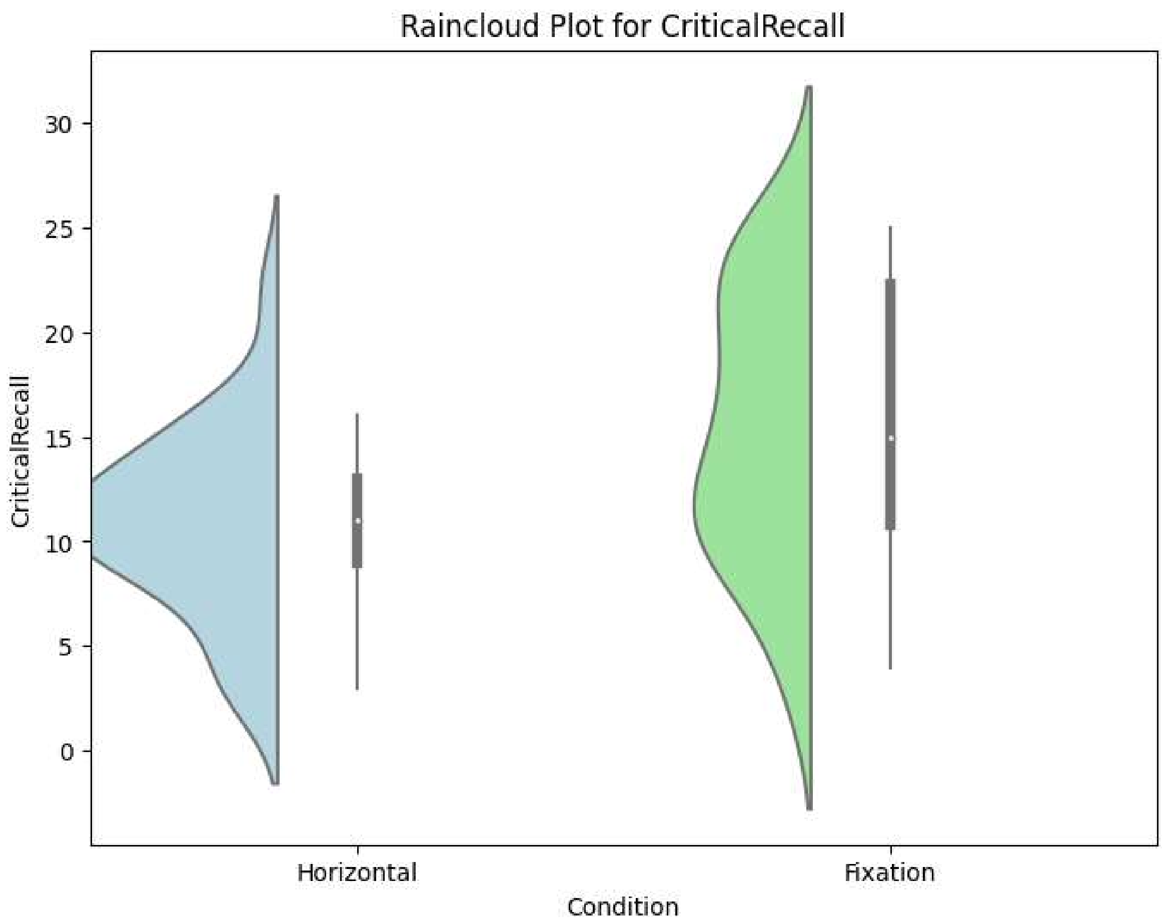
This is presuming that each research subject is appropriately categorized as belonging to the "Horizontal" or "Fixation" groups. Group assignments should be clear and unambiguous, guaranteeing that each participant's data is connected to a single condition.

In [51]:

```
import pandas as pd
import numpy as np
import ptitprince as pt
import matplotlib.pyplot as plt


fig, ax = plt.subplots(figsize=(8, 6))
pt.half_violinplot(
    x="Condition", y="CriticalRecall", data=data_sheet, palette={"Horizontal": "
)
ax.set_title("Raincloud Plot for CriticalRecall")
ax.set_ylabel("CriticalRecall")
ax.set_xlabel("Condition")

plt.show()
```



The density of the data distribution is shown by the Raincloud plot's width. Higher variability or dispersion in the CriticalRecall values for the Fixation group is indicated by a broader range for the Fixation condition relative to the Horizontal condition, which suggests that the data points in the Fixation condition are more dispersed.

In other words:

Wider span: Shows a wider range of values, indicating that participants' CriticalRecall outcomes or responses were more varied under the Fixation condition.

Assumption 4: Normality of Residuals

The residuals—differences between actual and expected values—are distributed normally, according to the null hypothesis.

Test: Shapiro-Wilk normalcy test for every circumstance. Interpretation: The assumption of normality is supported by a non-significant result indicating that the residuals have a normal distribution.

In [52]:
```
from scipy.stats import shapiro

condition1 = "Horizontal"
condition2 = "Fixation"

list_num = data_sheet[data_sheet["Condition"] == condition1]["CriticalRecall"]
list_num2 = data_sheet[data_sheet["Condition"] == condition2]["CriticalRecall"]


statistic1, p_value1 = shapiro(list_num)
statistic2, p_value2 = shapiro(list_num2)


print(f"Shapiro-Wilk Test for {condition1}: W = {statistic1:.3f}, p = {p_value1:
print(f"Shapiro-Wilk Test for {condition2}: W = {statistic2:.3f}, p = {p_value2:
```

```
Shapiro-Wilk Test for Horizontal: W = 0.959, p = 0.396
Shapiro-Wilk Test for Fixation: W = 0.926, p = 0.079
```

Assumption 5: Homogeneity of Variances

The two conditions' variances are equal, which is the null hypothesis. Levene's test of equality of variances is the test.

Interpretation: The notion of homogeneity is supported by a non-significant result, which shows that the variances are homogenous across situations.

In [53]:
```
from scipy.stats import levene

statistic, p_value = levene(list_num, list_num2)

print(f"Levene's Test for Homogeneity of Variances: W = {statistic:.3f}, p = {p_
```

```
Levene's Test for Homogeneity of Variances: W = 7.504, p = 0.009
```