Reproducing Findings PeerJ Review #97425

April 1, 2024

[1]:
```
import numpy as np
import pandas as pd

file = "peerj-97425-1-supplemental/peerj-97425/supplemental/
 ↪peerj-97425-Mastcatory_15_Feb_2024_DR_Fayad^_V2.xlsx"
```

0.1 Preparation

[2]:
```
cols = ["PTTno", "Gender", "AgeGroups", "Age", "Con1", "Con6", "impl1", "imp6"]

df = pd.read_excel(file, usecols=cols, nrows=45)

# df.head()
```

[3]:
```
gender_dict = {1:"Male", 2:"Female"}
agegroup_dict = {1:"<47", 2:"47-52", 3:">52"}
```

[4]:
```
def bin_age(age):

    agegroup = None

    if age >= 44:
        if age < 47:
            agegroup = 1
        elif age <= 52:
            agegroup = 2
        elif age <= 59:
            agegroup = 3

    assert agegroup != None

    return agegroup
```

0.2 Descriptive Statistics

[5]:
```
df.Age.describe().round(2)
```

[5]:
```
count    45.00
mean     50.47
std       4.78
min      44.00
25%      46.00
50%      50.00
75%      54.00
max      59.00
Name: Age, dtype: float64
```

0.3 Regarding Table (1): Gender Frequency

[6]:
```
df["Gender_decoded"] = df.Gender.map(gender_dict)
```

[7]:
```
df.Gender_decoded.value_counts()
```

[7]:
```
Female    24
Male      21
Name: Gender_decoded, dtype: int64
```

[8]:
```
df.Gender_decoded.value_counts()/len(df)
```

[8]:
```
Female    0.533333
Male      0.466667
Name: Gender_decoded, dtype: float64
```

0.4 Regarding Table (2) The mean and standard deviation of the perceived masticatory ability measurements at different intervals

[9]:
```
df.Con1.mean().round(2), df.Con1.std().round(1)
```

[9]:
```
(37.89, 10.6)
```

[10]:
```
df.Con6.mean().round(1), df.Con6.std().round(1)
```

[10]:
```
(36.4, 10.4)
```

[11]:
```
df.impl1.mean().round(1), df.impl1.std().round(1)
```

[11]:
```
(28.7, 8.4)
```

[12]:
```
df.imp6.mean().round(1), df.imp6.std().round(1)
```

[12]:
```
(27.0, 8.5)
```

[13]:
```
df.AgeGroups.map(agegroup_dict).value_counts()
```

[13]:
```
>52      18
47-52    14
<47      13
Name: AgeGroups, dtype: int64
```

0.5 Table (6) Relation between gender and PrMA

[14]:
```
(df.Age.map(bin_age) != df.AgeGroups).sum()
```

[14]:
```
0
```

[15]:
```
df.groupby(by="Gender_decoded").Con1.describe()
```

[15]:

Gender_decoded	count	mean	std	min	25%	50%	75%	max
Female	24.0	37.000000	10.400669	15.0	30.0	40.0	45.0	50.0
Male	21.0	38.904762	10.954016	15.0	35.0	40.0	45.0	55.0

[16]:
```
df.Con1.describe()
```

[16]:
```
count    45.000000
mean     37.888889
std      10.583482
min      15.000000
25%      30.000000
50%      40.000000
75%      45.000000
max      55.000000
Name: Con1, dtype: float64
```

[17]:
```
df.groupby(by="Gender_decoded").Con6.describe()
```

[17]:

Gender_decoded	count	mean	std	min	25%	50%	75%	max
Female	24.0	35.625000	10.664090	15.0	33.75	40.0	40.0	50.0
Male	21.0	37.190476	10.264595	15.0	35.00	40.0	45.0	53.0

[18]:
```
df.Con6.describe()
```

[18]:
```
count    45.000000
mean     36.355556
std      10.390458
min      15.000000
25%      35.000000
50%      40.000000
75%      45.000000
max      53.000000
Name: Con6, dtype: float64
```

[19]:
```
df.groupby(by="Gender_decoded").impl1.describe()
```

[19]:
```
                count       mean       std   min   25%   50%   75%   max
Gender_decoded
Female           24.0  28.083333  8.586727  10.0  25.0  30.0  35.0  39.0
Male             21.0  29.333333  8.392457  15.0  23.0  30.0  35.0  50.0
```

[20]:
```
df.impl1.describe()
```

[20]:
```
count    45.000000
mean     28.666667
std       8.423452
min      10.000000
25%      25.000000
50%      30.000000
75%      35.000000
max      50.000000
Name: impl1, dtype: float64
```

[21]:
```
df.groupby(by="Gender_decoded").imp6.describe()
```

[21]:
```
                count       mean       std   min   25%   50%   75%   max
Gender_decoded
Female           24.0  26.458333  8.214405   9.0  23.0  28.0  31.5  38.0
Male             21.0  27.523810  9.003438  13.0  21.0  28.0  33.0  43.0
```

[22]:
```
df.imp6.describe()
```

[22]:
```
count    45.000000
mean     26.955556
std       8.509234
min       9.000000
25%      23.000000
50%      28.000000
75%      33.000000
max      43.000000
Name: imp6, dtype: float64
```

0.6 Regarding Table (7) Relation between different age groups and PrMA

[23]:
```
df["AgeGroups_decoded"] = df.AgeGroups.map(agegroup_dict)
```

[24]:
```
df.groupby(by="AgeGroups_decoded").Con1.describe()
```

[24]:
```
                   count       mean        std   min   25%   50%   75%   max
AgeGroups_decoded
47-52               14.0  34.071429  11.874111  15.0  25.0  39.0  40.0  54.0
<47                 13.0  36.923077  10.711843  15.0  35.0  40.0  40.0  55.0
```

```
>52                 18.0  41.555556   8.610679  20.0  38.5  45.0  45.0  55.0
```

[25]:
```
df.groupby(by="AgeGroups_decoded").Con6.describe()
```

[25]:
```
                    count       mean        std   min   25%   50%   75%   max
AgeGroups_decoded
47-52                14.0  32.142857  11.883131  15.0  20.0  37.5  40.0  50.0
<47                  13.0  35.615385  11.064496  15.0  30.0  35.0  40.0  53.0
>52                  18.0  40.166667   7.390375  15.0  40.0  40.0  45.0  50.0
```

[26]:
```
df.groupby(by="AgeGroups_decoded").impl1.describe()
```

[26]:
```
                    count       mean       std   min    25%   50%   75%   max
AgeGroups_decoded
47-52                14.0  26.428571  9.362070  12.0  15.75  30.0  35.0  35.0
<47                  13.0  27.384615  7.297699  15.0  20.00  30.0  30.0  40.0
>52                  18.0  31.333333  8.131276  10.0  30.00  30.0  35.0  50.0
```

[27]:
```
df.groupby(by="AgeGroups_decoded").imp6.describe()
```

[27]:
```
                    count       mean       std   min    25%   50%   75%   max
AgeGroups_decoded
47-52                14.0  24.642857  8.889122  11.0  17.25  28.0  28.0  38.0
<47                  13.0  25.615385  9.251473  13.0  20.00  23.0  33.0  43.0
>52                  18.0  29.722222  7.258306   9.0  28.00  29.0  33.0  43.0
```

0.7 Regarding Table (3) Tests of Normality.

[28]:
```
from scipy.stats import shapiro

shapiro(df[["Con1", "Con6", "impl1", "imp6"]])
```

[28]:
```
ShapiroResult(statistic=0.9741241335868835, pvalue=0.001976399915292859)
```

[29]:
```
# from scipy.stats import kstest
from statsmodels.stats.diagnostic import lilliefors

# print(kstest(df["Con1"], 'norm'))
# print(kstest(df["Con6"], 'norm'))
# print(kstest(df["impl1"], 'norm'))
# print(kstest(df["imp6"], 'norm'))
print(lilliefors(df["Con1"], dist='norm'))
print(lilliefors(df["Con6"], dist='norm'))
print(lilliefors(df["impl1"], dist='norm'))
print(lilliefors(df["imp6"], dist='norm'))
```

```
(0.179053327685899, 0.0010380044895450715)
(0.23711173118527018, 0.0009999999999998899)
```

```
(0.2073296036721995, 0.0009999999999998899)
(0.1932889281023809, 0.0009999999999998899)
```

0.8 Regarding Table (4) Friedman Test

[30]:
```
from scipy.stats import friedmanchisquare

friedmanchisquare(df["Con1"], df["Con6"], df["impl1"], df["imp6"])
```

[30]:
```
FriedmanchisquareResult(statistic=96.60598503740653,
pvalue=8.339628106377734e-21)
```

0.9 Regarding Table (5): Perceived masticatory ability means comparison at different intervals

[31]:
```
values = df.Con1.to_list() + df.Con6.to_list() + df.impl1.to_list() + df.imp6.
  ↪to_list()
groups = ["Con1"]*45 + ["Con6"]*45 + ["impl1"]*45 + ["imp6"]*45
```

[32]:
```
from statsmodels.stats.multicomp import pairwise_tukeyhsd

tukey_results = pairwise_tukeyhsd(values, groups)
```

[33]:
```
print(tukey_results.summary())
```

```
 Multiple Comparison of Means - Tukey HSD, FWER=0.05
====================================================
group1 group2 meandiff p-adj   lower    upper  reject
----------------------------------------------------
  Con1   Con6  -1.5333 0.8709  -6.7448  3.6781  False
  Con1   imp6 -10.9333    0.0 -16.1448 -5.7219   True
  Con1  impl1  -9.2222    0.0 -14.4337 -4.0108   True
  Con6   imp6     -9.4    0.0 -14.6115 -4.1885   True
  Con6  impl1  -7.6889  0.001 -12.9003 -2.4774   True
  imp6  impl1   1.7111 0.8296  -3.5003  6.9226  False
----------------------------------------------------
```

[34]:
```
from scipy.stats import ttest_ind

data = pd.DataFrame()
data["values"] = values
data["groups"] = groups

group_column = "groups"
value_column = "values"

groups = data[group_column].unique()
```

```
for i in range(len(groups)):
    for j in range(i+1, len(groups)):
        group1 = groups[i]
        group2 = groups[j]
        subset1 = data[data[group_column] == group1][value_column]
        subset2 = data[data[group_column] == group2][value_column]
        t_stat, p_val = ttest_ind(subset1, subset2)
        bonferroni_corrected_p_val = p_val / len(groups)
        print(f"Comparison between {group1} and {group2}: p-value =␣
 ↪{bonferroni_corrected_p_val}")
```

```
Comparison between Con1 and Con6: p-value = 0.12245234624753637
Comparison between Con1 and impl1: p-value = 3.901190721196135e-06
Comparison between Con1 and imp6: p-value = 1.3943216869538753e-07
Comparison between Con6 and impl1: p-value = 5.471133863474832e-05
Comparison between Con6 and imp6: p-value = 2.4332412227230405e-06
Comparison between impl1 and imp6: p-value = 0.08508819281019667
```

0.10 Table (6): Relation between gender and PrMA

[35]:
```
from scipy.stats import mannwhitneyu

group1_data = df[df["Gender_decoded"] == 'Female'][["Con1", "Con6", "impl1",␣
 ↪"imp6"]]
group2_data = df[df["Gender_decoded"] == 'Male'][["Con1", "Con6", "impl1",␣
 ↪"imp6"]]

# Perform Mann-Whitney U test
statistic, p_value = mannwhitneyu(group1_data, group2_data)

# Print results
print("Mann-Whitney U test results:")
print(f"U-statistic: {statistic}")
print(f"P-value: {p_value:}")
```

```
Mann-Whitney U test results:
U-statistic: [238.  233.  249.5 238. ]
P-value: [0.75558393 0.66646657 0.96287327 0.75507799]
```

0.11 Regarding Table (7): Relation between different age group and PrMA

[36]:
```
from scipy.stats import kruskal

# Extract data for each group
groups = [group_data for _, group_data in df.
 ↪groupby("AgeGroups_decoded")[["Con1", "Con6", "impl1", "imp6"]]]

# Perform Kruskal-Wallis test
```

```
statistic, p_value = kruskal(*groups)

# Print results
print("Kruskal-Wallis test results:")
print(f"H-statistic: {statistic}")
print(f"P-value: {p_value}")
```

```
Kruskal-Wallis test results:
H-statistic: [4.3772598  4.39621397 2.1095534  3.62307223]
P-value: [0.11207019 0.11101311 0.34827019 0.16340294]
```