```
import pandas as pd
import numpy as np

import statistics as st
from statsmodels.stats import weightstats as stests
from scipy.stats import ttest_ind
from scipy.stats import f_oneway
import matplotlib.pyplot as plt

import altair as alt
```

```
# Read the data and display 5 rows
df = pd.read_excel('/content/WorldUniversity.xlsx')
df.head()
```

| | world_rank | institution | country | national_rank | quality_of_education | alumni_employment | qual |
|---|---|---|---|---|---|---|---|
| **0** | 1 | Harvard University | USA | 1 | 7 | 9 | |
| **1** | 2 | Massachusetts Institute of Technology | USA | 2 | 9 | 17 | |
| **2** | 3 | Stanford University | USA | 3 | 17 | 11 | |
| **3** | 4 | University of Cambridge | United Kingdom | 1 | 10 | 24 | |
| **4** | 5 | California Institute of Technology | USA | 4 | 2 | 29 | |

```
# Getting column insights

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2200 entries, 0 to 2199
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   world_rank            2200 non-null   int64  
 1   institution           2200 non-null   object 
 2   country               2200 non-null   object 
 3   national_rank         2200 non-null   int64  
 4   quality_of_education  2200 non-null   int64  
 5   alumni_employment     2200 non-null   int64  
 6   quality_of_faculty    2200 non-null   int64  
 7   publications          2200 non-null   int64  
 8   influence             2200 non-null   int64  
 9   citations             2200 non-null   int64  
 10  broad_impact          2000 non-null   float64
 11  patents               2200 non-null   int64  
 12  score                 2200 non-null   float64
 13  year                  2200 non-null   int64  
dtypes: float64(2), int64(10), object(2)
memory usage: 240.8+ KB
```

```
# Checking for null values

df.isnull().sum()
```

```
world_rank                0
institution               0
country                   0
national_rank             0
quality_of_education      0
alumni_employment         0
quality_of_faculty        0
publications              0
influence                 0
citations                 0
broad_impact            200
patents                   0
score                     0
year                      0
dtype: int64
```

```
# counting year column

len(df['year'])
```

```
2200
```

```
# finding statistical description

df.describe()
```

| | world_rank | national_rank | quality_of_education | alumni_employment | quality_of_faculty | pub |
|---|---|---|---|---|---|---|
| **count** | 2200.000000 | 2200.000000 | 2200.000000 | 2200.000000 | 2200.000000 | 22 |
| **mean** | 459.590909 | 40.278182 | 275.100455 | 357.116818 | 178.888182 | 4 |
| **std** | 304.320363 | 51.740870 | 121.935100 | 186.779252 | 64.050885 | 3 |
| **min** | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | |
| **25%** | 175.750000 | 6.000000 | 175.750000 | 175.750000 | 175.750000 | 1 |
| **50%** | 450.500000 | 21.000000 | 355.000000 | 450.500000 | 210.000000 | 4 |
| **75%** | 725.250000 | 49.000000 | 367.000000 | 478.000000 | 218.000000 | 7 |
| **max** | 1000.000000 | 229.000000 | 367.000000 | 567.000000 | 218.000000 | 1( |

```
# Z-Test

alm_emp = df['alumni_employment']
print('Data =')
print(alm_emp)
print('\n')

emp_mean = st.mean(alm_emp)
print('Mean data = ', emp_mean)

emp_stdv = st.stdev(alm_emp)
print('Standard Deviation :', emp_stdv)

ztest ,pval = stests.ztest(alm_emp, value=30)
print("Z-Test Score :", ztest)
print("P-Value :", pval)

if pval<0.05:
  print("Reject Null Hypothesis")
else:
  print("Accept the Null Hypothesis")
```

```
Data =
0         9
1        17
2        11
3        24
4        29
       ...
2195    567
2196    566
2197    549
2198    567
2199    567
Name: alumni_employment, Length: 2200, dtype: int64


Mean data =  357.1168181818182
Standard Deviation : 186.77925165404082
Z-Test Score : 82.145841452733
P-Value : 0.0
Reject Null Hypothesis
```

```
# T-test


top3 = df[df['world_rank'] <= 3]
rest = df[df['world_rank'] > 3]

t_test ,pval2 = ttest_ind(top3['quality_of_education'], rest['quality_of_education'])

print("T-Test Score :", t_test)
print("P-Value :", pval2)

if pval2<0.05:
  print("Reject Null Hypothesis")
else:
  print("Accept the Null Hypothesis")
```

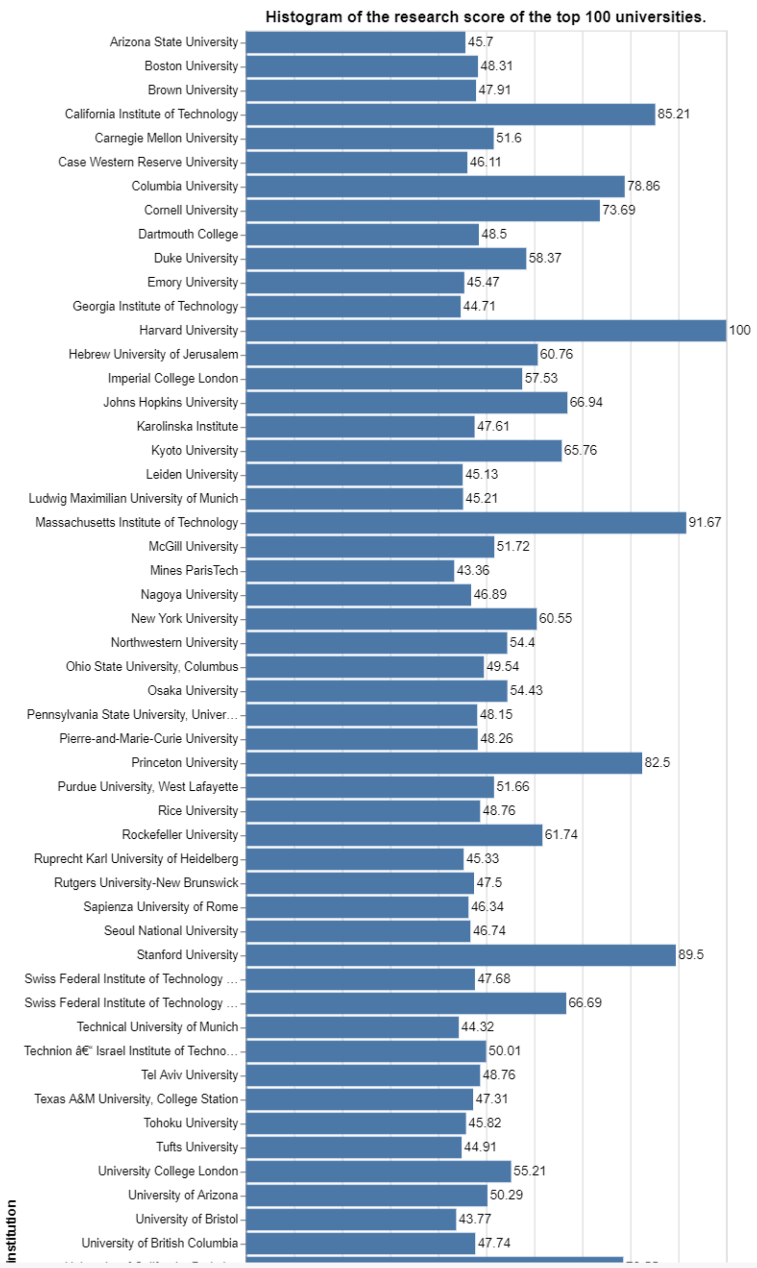
```
T-Test Score : -7.748907591220499
P-Value : 1.4052202544891618e-14
Reject Null Hypothesis
```

```
# ANNOVA Test

groups = [df[df['country'] == country]['quality_of_education'] for country in df['country'].unique()]

f_statistic, p_value_anova = f_oneway(*groups)

print("F-Statistic: ",f_statistic)
print("P-Value (ANOVA): ",p_value_anova)
```

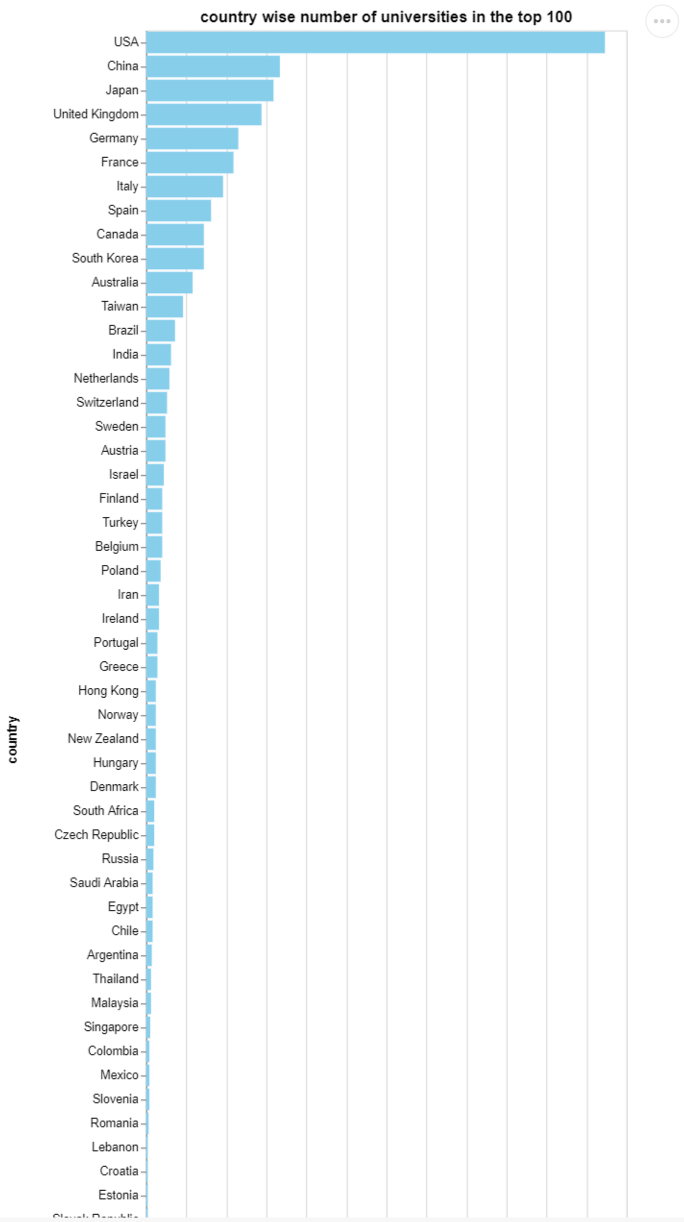
```
F-Statistic:  9.934212491053064
P-Value (ANOVA):  7.812029724331615e-75
```

## Visualization

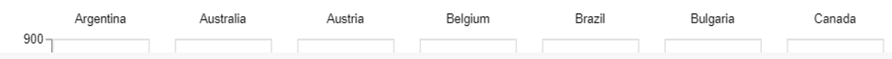
```
# Histogram



top100 = df.head(100)

base = alt.Chart(top100).mark_bar(cornerRadiusBottomRight=3,
    cornerRadiusTopRight=3).encode(
    alt.X('score'),
    alt.Y('institution'),
    text = 'score'
).properties(
    title = 'Histogram of the research score of the top 100 universities. '
)
base.mark_bar() + base.mark_text(align='left', dx=2)
```



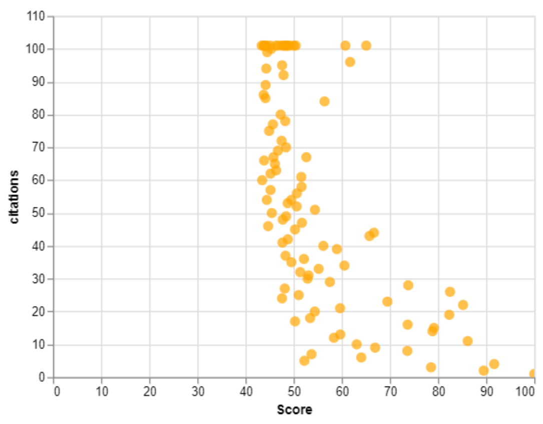
```
# Bar Chart

df_counts = df['country'].value_counts().reset_index()
df_counts.columns = ['country', 'count']


bar_chart = alt.Chart(df_counts).mark_bar(color='skyblue').encode(
    x='count:Q',
    y=alt.Y('country:N', sort='-x'),
    tooltip=['country:N', 'count:Q']
).properties(
    title='country wise number of universities in the top 100'
)

alt.renderers.enable('colab')
bar_chart
```



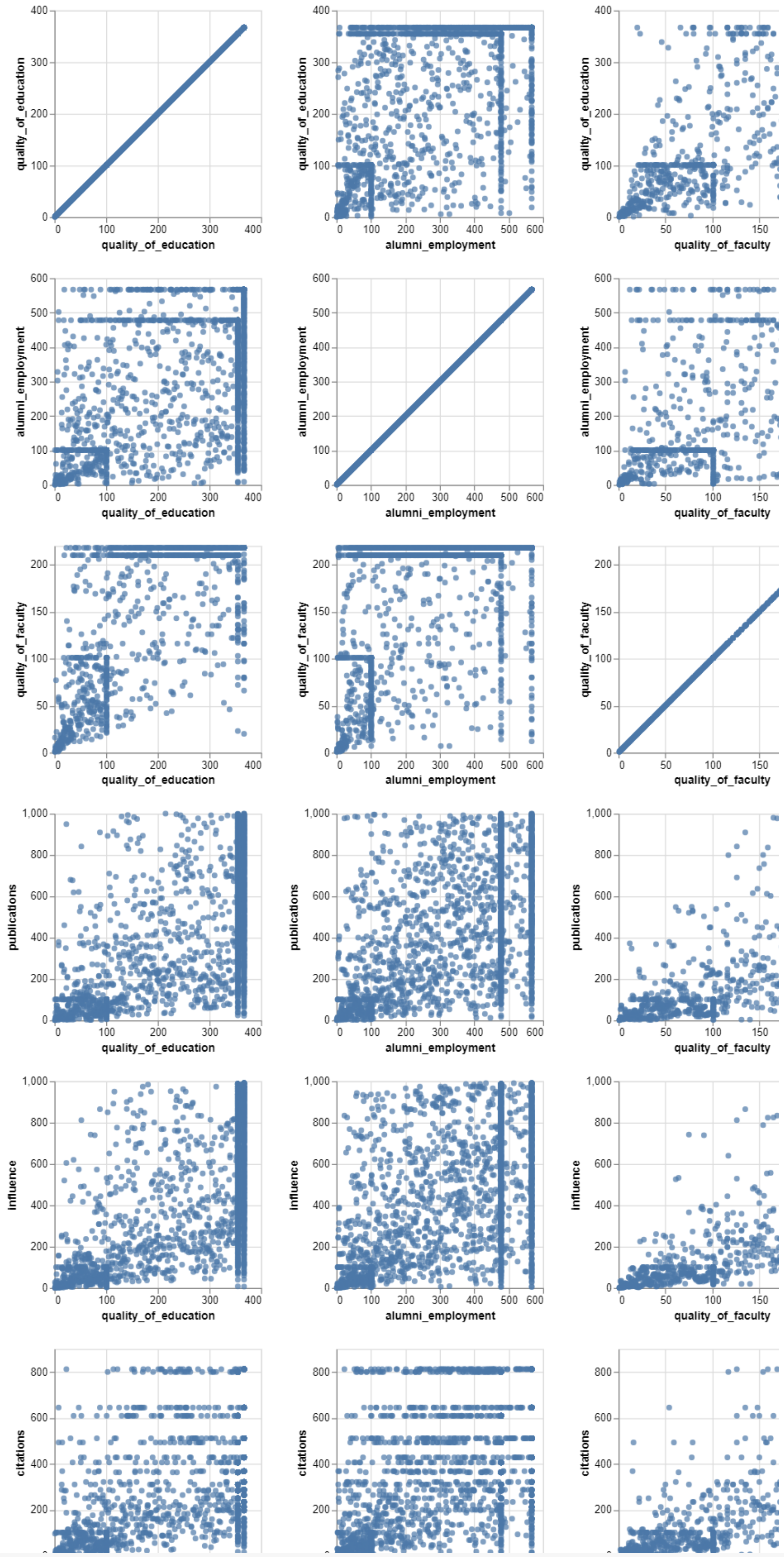
```
# Grouped Bar chart

bar_chart = alt.Chart(df).mark_bar().encode(
    x = 'year:O',
    y = alt.Y('mean(citations):Q', title='Mean Citation'),
    color = 'year:N'
)

alt.layer(bar_chart).facet(
    column = 'country'
)
```



```
# Scatter Chart
#(since we dont have Teaching Score column in dataset so i have used Score column)

scatterc = alt.Chart(top100).mark_circle(size = 70, color = 'orange').encode(
    x=alt.Y('score', title = "Score"),
    y=alt.Y('citations', title = "citations")
).interactive()

alt.renderers.enable('colab')
scatterc
```

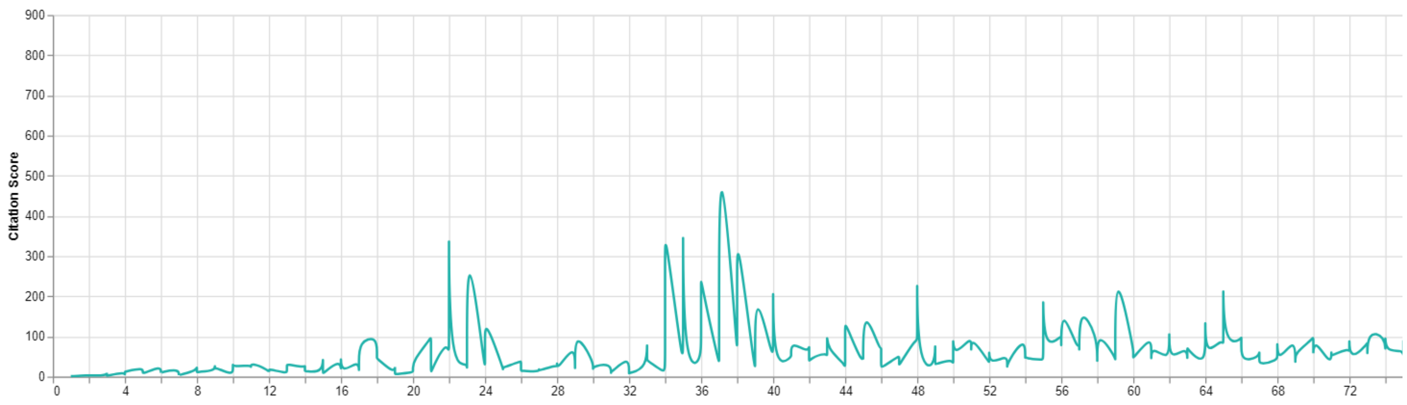


```
# Scatter Matrix

nos_columns = ['quality_of_education', 'alumni_employment', 'quality_of_faculty','publications',
               'influence', 'citations', 'patents', 'score']


matrix = alt.Chart(df).mark_circle().encode(
    alt.X(alt.repeat("column"), type = 'quantitative'),
    alt.Y(alt.repeat("row"), type = 'quantitative')
).properties(width=200,
    height=200).repeat(
    row = nos_columns,
    column= nos_columns
)


alt.renderers.enable('colab')
matrix
```

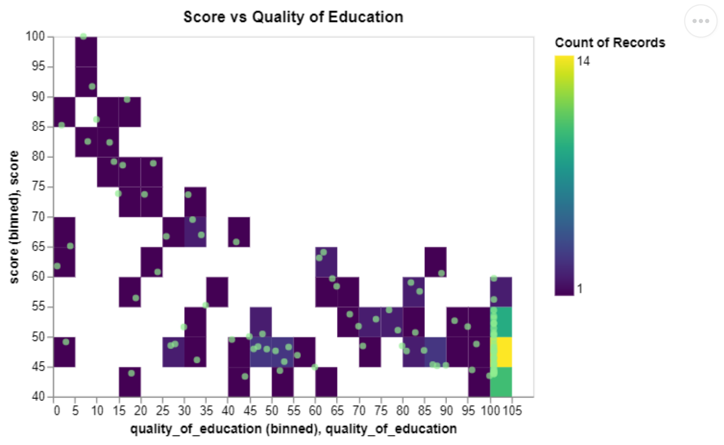


```
# Line chart

linec = alt.Chart(df).mark_line(interpolate='basis', color='lightseagreen').encode(
    x = alt.X('world_rank:Q', title = "University Rank"),
    y = alt.Y('citations:Q', title = "Citation Score")
).properties(width=15000)
```

```
alt.renderers.enable('colab')
linec
```



```
# Hybrid Chart
# (we dont have Teaching score and Research score column so i am using quality_of_education vs score columns)

heatm = alt.Chart(top100).mark_rect().encode(
    x=alt.X('quality_of_education:Q', bin=alt.Bin(maxbins=20)),
    y=alt.Y('score:Q', bin=alt.Bin(maxbins=20)),
    color=alt.Color('count():Q', scale=alt.Scale(scheme='viridis')),
    tooltip=['count():Q']
).properties(
    title='Score vs Quality of Education'
)


scatter = alt.Chart(top100).mark_circle(size=30, color='lightgreen').encode(
    x='quality_of_education:Q',
    y='score:Q',
    tooltip=['institution:N']
)

res = heatm + scatter
res
```


Score vs Quality of Education
score (binned), score
quality_of_education (binned), quality_of_education
Count of Records