# pyStatQuest

Yao-Chung Chen

2022/8/19

# Table of contents

# Preface

# 1 PCA (Singular Value Decomposition)

StatQuest 154 singular value decomposition SVD PCA:

python

### 1.0.1

PCA clustering Sum of square of distances / n-1 (variance) PC variance scree plot PCA PC1, PC2 scree plot. PCA Unsupervised dimensionality reduction

- cocktail recipe, linear combination
- singular vector, Eigenvector, loading score

### 1.0.2

PCA Singular Value Decomposition (SVD) Variances
PCA **linear dimensionality reduction**

- Dimension feature
-
-
-
-

### 1.0.3 PCA

- Factor Analysis
- Independent Component Analysis (ICA)

### 1.0.4 1

#### 1.0.4.1 iris data

plotly iris features

```python
import plotly.express as px
import pandas as pd

# load iris data
df = px.data.iris()
df.head()
```

```
/Users/yaochung41/.virtualenvs/statquest/lib/python3.9/site-packages/IPython/core/formatters

In future versions `DataFrame.to_latex` is expected to utilise the base implementation of `S
```

| | sepal_length | sepal_width | petal_length | petal_width | species | species_id |
|---|---|---|---|---|---|---|
| 0 | 5.1 | 3.5 | 1.4 | 0.2 | setosa | 1 |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 | setosa | 1 |
| 2 | 4.7 | 3.2 | 1.3 | 0.2 | setosa | 1 |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 | setosa | 1 |
| 4 | 5.0 | 3.6 | 1.4 | 0.2 | setosa | 1 |

```python
# create scatter plot using plotly
features = ["sepal_width", "sepal_length", "petal_width", "petal_length"]

fig = px.scatter_matrix(
    df,
    dimensions=features,
    color="species"
)
fig.update_traces(diagonal_visible=False)
fig.show()
```

```
Unable to display output for mime type(s): text/html
```

```
Unable to display output for mime type(s): text/html
```

#### 1.0.4.2 sklearn PCA

```
from sklearn.decomposition import PCA

pca = PCA()
components = pca.fit_transform(df[features])
labels = {
    str(i): f"PC {i+1} ({var:.1f}%)"
    for i, var in enumerate(pca.explained_variance_ratio_ * 100)
}

fig = px.scatter_matrix(
    components,
    labels=labels,
    dimensions=range(4),
    color=df["species"]
)
fig.update_traces(diagonal_visible=False)
fig.show()
```

```
Unable to display output for mime type(s): text/html
```

PCA xy PC1 (92.5%), PC2 (5.3%)

### 1.0.5 2

1 scale features PCA scale 2 seaborn diamonds PCA

```
# load packages
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn import decomposition

diamond = sns.load_dataset("diamonds")
diamond.head()
```

```
/Users/yaochung41/.virtualenvs/statquest/lib/python3.9/site-packages/IPython/core/formatters

In future versions `DataFrame.to_latex` is expected to utilise the base implementation of `S
```

| | carat | cut | color | clarity | depth | table | price | x | y | z |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.23 | Ideal | E | SI2 | 61.5 | 55.0 | 326 | 3.95 | 3.98 | 2.43 |
| 1 | 0.21 | Premium | E | SI1 | 59.8 | 61.0 | 326 | 3.89 | 3.84 | 2.31 |
| 2 | 0.23 | Good | E | VS1 | 56.9 | 65.0 | 327 | 4.05 | 4.07 | 2.31 |
| 3 | 0.29 | Premium | I | VS2 | 62.4 | 58.0 | 334 | 4.20 | 4.23 | 2.63 |
| 4 | 0.31 | Good | J | SI2 | 63.3 | 58.0 | 335 | 4.34 | 4.35 | 2.75 |

features carat, cut, color, clarity, depth, table, price, x, y, z
PCA `dummy variables`

```
# get dummies and store it in a variable
dummies = pd.get_dummies(diamond[["cut", "clarity"]])

# concat dummies to original dataframe and drop values
merged = pd.concat([diamond, dummies], axis='columns')
merged.drop(['cut', 'clarity'], axis='columns', inplace=True)

#random select rows
merged = merged.sample(n=500)

print(merged.describe())
```

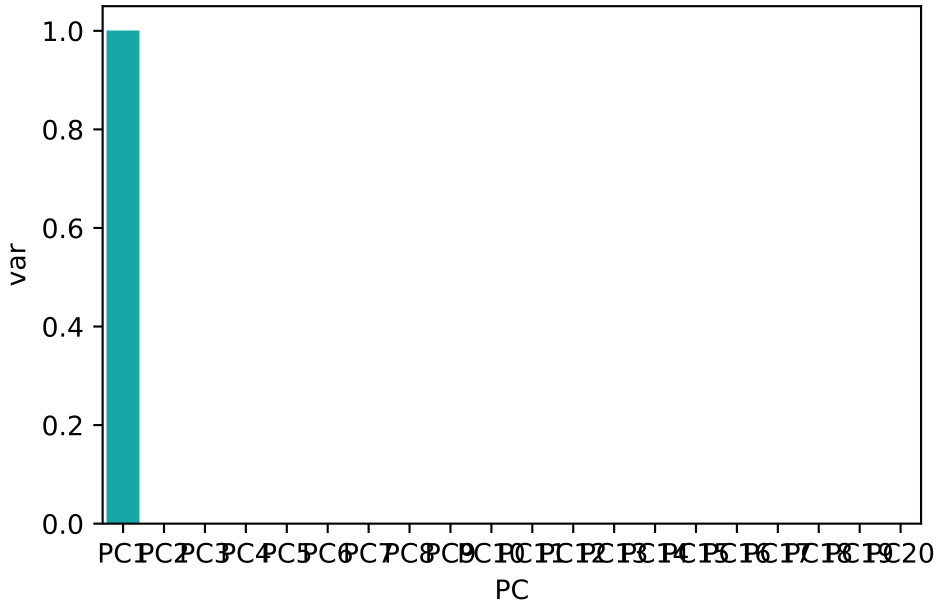
```
            carat       depth       table         price           x  \
count  500.000000  500.000000  500.000000    500.000000  500.000000
mean     0.813920   61.807200   57.560600   4121.034000    5.763260
std      0.489151    1.466072    2.345009   4136.904937    1.126839
min      0.230000   55.200000   52.000000    391.000000    3.870000
25%      0.410000   61.100000   56.000000   1008.500000    4.740000
50%      0.700000   61.900000   57.000000   2411.000000    5.700000
75%      1.030000   62.600000   59.000000   5656.500000    6.520000
max      2.670000   69.800000   67.000000  18686.000000    8.690000

                y           z   cut_Ideal  cut_Premium  cut_Very Good  \
count  500.000000  500.000000  500.000000   500.000000     500.000000
mean     5.764320    3.562820    0.372000     0.250000       0.242000
std      1.119366    0.700083    0.483822     0.433446       0.428723
min      3.850000    2.430000    0.000000     0.000000       0.000000
```

```
25%      4.760000    2.930000    0.000000    0.000000       0.000000
50%      5.725000    3.540000    0.000000    0.000000       0.000000
75%      6.510000    4.032500    1.000000    0.250000       0.000000
max      8.640000    5.540000    1.000000    1.000000       1.000000

         cut_Good    cut_Fair  clarity_IF  clarity_VVS1  clarity_VVS2  \
count  500.000000  500.000000  500.000000    500.000000    500.000000
mean     0.106000    0.030000    0.030000      0.080000      0.094000
std      0.308146    0.170758    0.170758      0.271565      0.292121
min      0.000000    0.000000    0.000000      0.000000      0.000000
25%      0.000000    0.000000    0.000000      0.000000      0.000000
50%      0.000000    0.000000    0.000000      0.000000      0.000000
75%      0.000000    0.000000    0.000000      0.000000      0.000000
max      1.000000    1.000000    1.000000      1.000000      1.000000

       clarity_VS1  clarity_VS2  clarity_SI1  clarity_SI2  clarity_I1
count   500.000000   500.000000   500.000000   500.000000  500.000000
mean      0.168000     0.220000     0.230000     0.168000    0.010000
std       0.374241     0.414661     0.421254     0.374241    0.099598
min       0.000000     0.000000     0.000000     0.000000    0.000000
25%       0.000000     0.000000     0.000000     0.000000    0.000000
50%       0.000000     0.000000     0.000000     0.000000    0.000000
75%       0.000000     0.000000     0.000000     0.000000    0.000000
max       1.000000     1.000000     1.000000     1.000000    1.000000
```

color min, max, std PCA

```
pca = decomposition.PCA()
pc = pca.fit_transform(merged.loc[:, merged.columns!='color'])
pc_df = pd.DataFrame(data=pc)
pc_df.head()

df = pd.DataFrame({
  'var': pca.explained_variance_ratio_,
  'PC':["PC" + str(i) for i in list(range(1,21))]
  })

sns.barplot(x = 'PC', y='var', data=df, color="c")
```

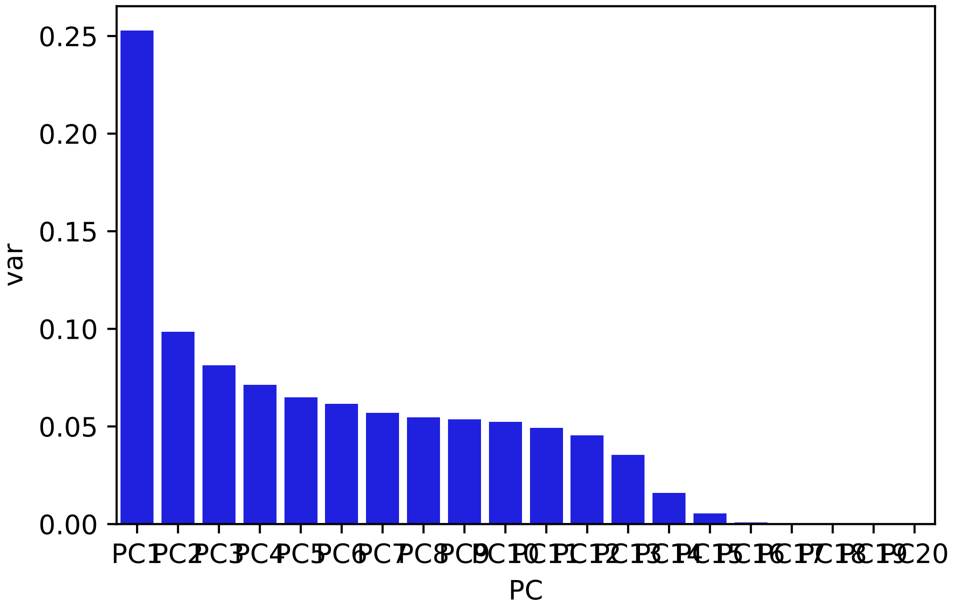
```
<AxesSubplot:xlabel='PC', ylabel='var'>
```



variance

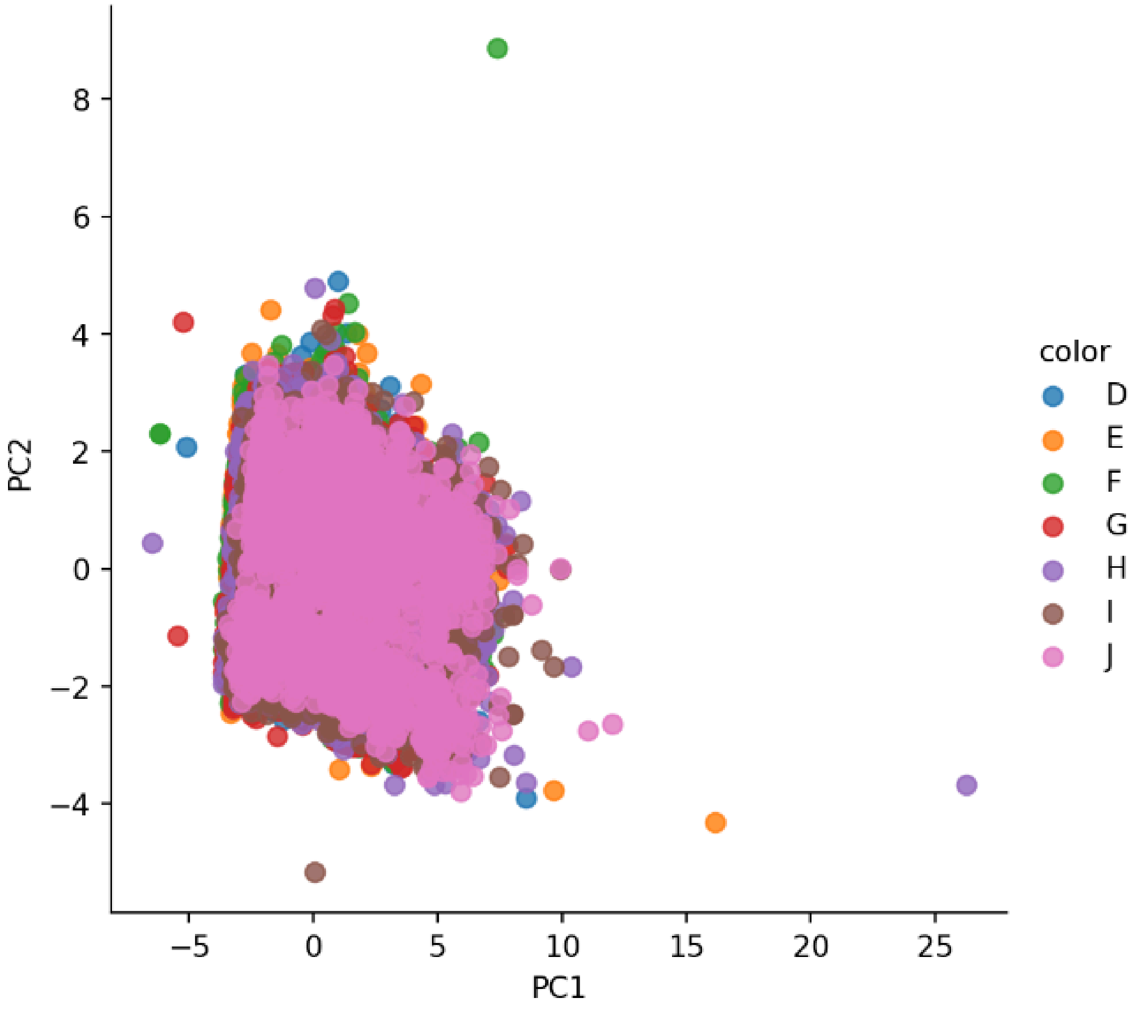
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
merged_scale = scaler.fit_transform(merged.loc[:, merged.columns!='color'])

pca = decomposition.PCA()
pc_scale = pca.fit_transform(merged_scale)
pc_df_scale = pd.DataFrame(pc_scale, columns = ["PC" + str(i) for i in list(range(1,21))])
pc_df_scale['color'] = merged.color

df_scale = pd.DataFrame({
  'var': pca.explained_variance_ratio_,
  'PC': ["PC" + str(i) for i in list(range(1,21))]
})

sns.barplot(x = 'PC', y='var', data=df_scale, color="b")
```

```
<AxesSubplot:xlabel='PC', ylabel='var'>
```



```
standard scaler
```

```
sns.lmplot(
  x="PC1",
  y="PC2",
  data=pc_df_scale,
  hue="color",
  fit_reg = False,
  legend=True,
  scatter_kws={"s": 40}
)
```

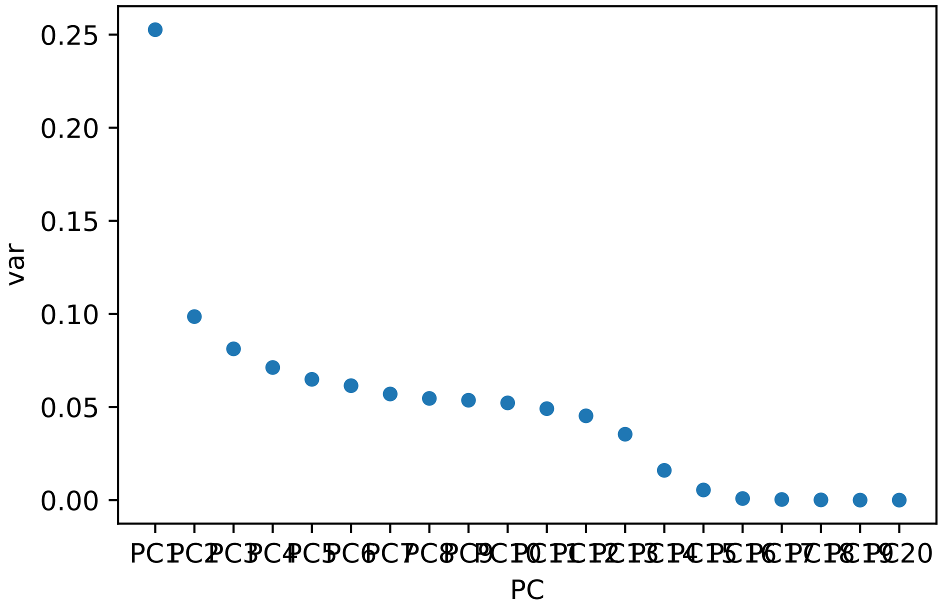


PCA clustering scree plot variance

scree plot using `sns.scatterplot`

```
pc_value = np.arange(pca.n_components_) + 1
sns.scatterplot(
  x='PC',
  y='var',
  data=df_scale
)
```

```
<AxesSubplot:xlabel='PC', ylabel='var'>
```



### 1.0.6

- PCA
- `normalization, standardization)` PCA
- (1) xy PC1, PC2 (2) Scree plot
- PCs Variance (Ex: 70%)

### 1.0.7

PCA
Plotly PCA
PCA
Python and R tips

# 2 Logistic Regression

StatQuest Logistic Regression

- 
- True/ False
- ( Walid Test)
- fit the line with Maximum-likelihood

python

# 3 Maximum likelihood

StatQuest likelihood maximum likelihood

Probability is not likelihood!

Maximum likelihood, clearly explained!

-

# 4 p-value

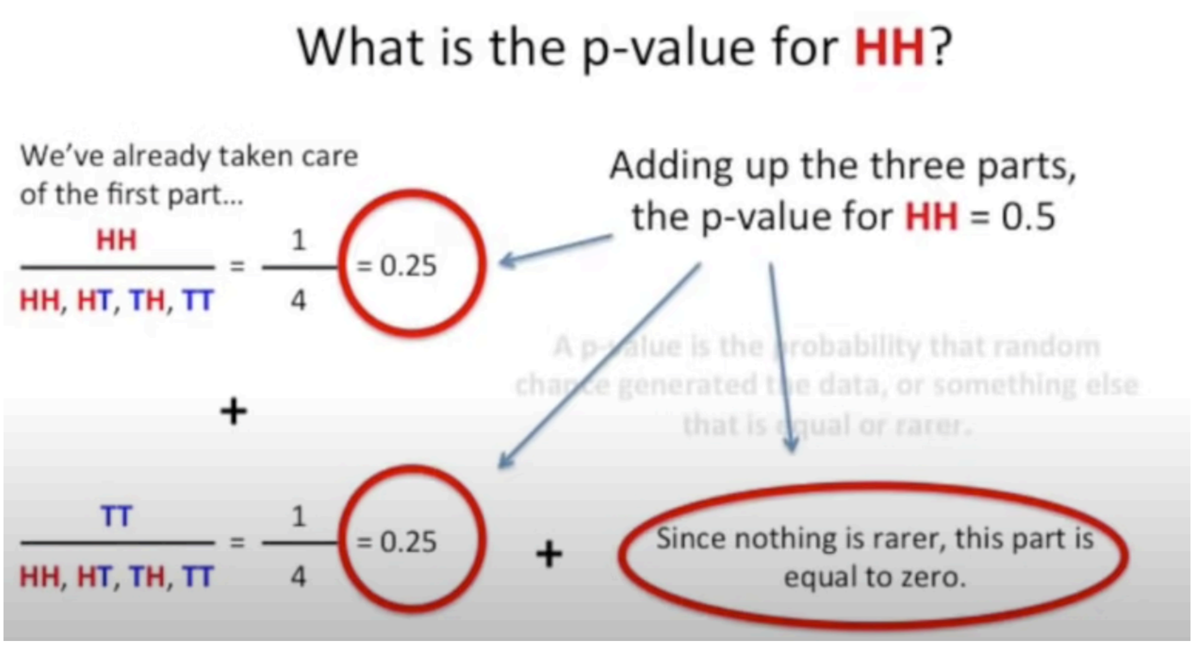
StatQuest p-value

p-value

ℹ Note

A *p-value* is the probabillity that random chance generated the data, or something else that is equal or rarer.

p-value data Part1 data (density) Part2 p-value



What is the p-value for HH?

We've already taken care of the first part…

$$\frac{HH}{HH, HT, TH, TT} = \frac{1}{4} = 0.25$$

+

$$\frac{TT}{HH, HT, TH, TT} = \frac{1}{4} = 0.25$$

+

Since nothing is rarer, this part is equal to zero.

Adding up the three parts, the p-value for HH = 0.5

A p-value is the probability that random chance generated the data, or something else that is equal or rarer.

HH p-value HH TT 0.25 part1: 0.25 + 0.25 = 0.5 part2: 0 HH p-value 0.5

# 5 Introduction

This is a book created from markdown and executable code.

See Knuth (1984) for additional discussion of literate programming.

# 6 Summary

In summary, this book has no content whatsoever.

# References

Knuth, Donald E. 1984. "Literate Programming." *Comput. J.* 27 (2): 97–111. https://doi.org/10.1093/comjnl/27.2.97.