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

```
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

```
# 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`

```python
# 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())
```

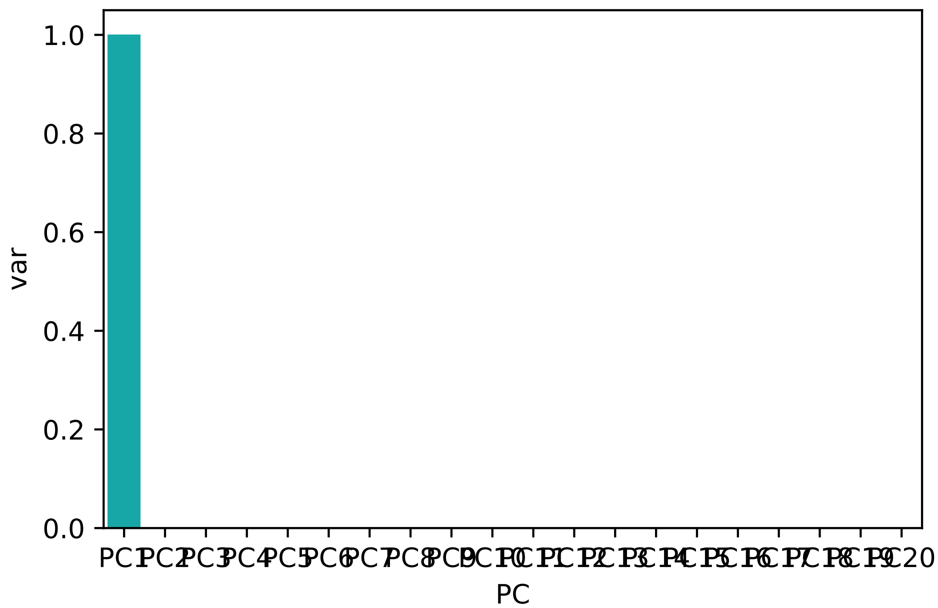
```
            carat       depth       table         price           x  \
count  500.000000  500.000000  500.000000    500.000000  500.000000
mean     0.794720   61.721800   57.508600   3891.870000    5.729640
std      0.462839    1.451799    2.260089   4013.002554    1.120256
min      0.230000   53.300000   52.000000    413.000000    3.900000
25%      0.400000   61.000000   56.000000    951.000000    4.757500
50%      0.700000   61.800000   57.000000   2367.500000    5.680000
75%      1.062500   62.400000   59.000000   5369.250000    6.585000
max      2.540000   67.300000   67.000000  18797.000000    8.800000

                y           z   cut_Ideal  cut_Premium  cut_Very Good  \
count  500.000000  500.000000  500.000000   500.000000     500.000000
mean     5.730760    3.535200    0.422000     0.256000       0.188000
std      1.112807    0.685153    0.494373     0.436859       0.391103
min      3.930000    2.380000    0.000000     0.000000       0.000000
```

```
25%       4.747500    2.910000    0.000000     0.000000       0.000000
50%       5.695000    3.535000    0.000000     0.000000       0.000000
75%       6.600000    4.060000    1.000000     1.000000       0.000000
max       8.700000    5.360000    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      0.096000    0.038000    0.032000      0.062000      0.092000
std       0.294886    0.191388    0.176176      0.241397      0.289315
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   500.00000
mean       0.148000     0.220000     0.244000     0.182000     0.02000
std        0.355456     0.414661     0.429923     0.386231     0.14014
min        0.000000     0.000000     0.000000     0.000000     0.00000
25%        0.000000     0.000000     0.000000     0.000000     0.00000
50%        0.000000     0.000000     0.000000     0.000000     0.00000
75%        0.000000     0.000000     0.000000     0.000000     0.00000
max        1.000000     1.000000     1.000000     1.000000     1.00000
```

color min, max, std PCA

```python
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")
```

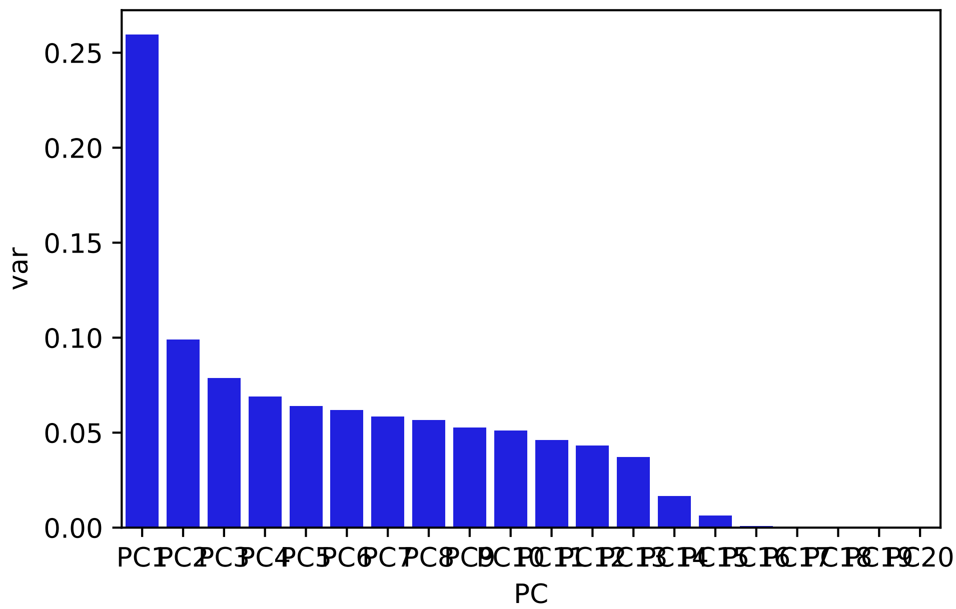
```
<AxesSubplot:xlabel='PC', ylabel='var'>
```



variance

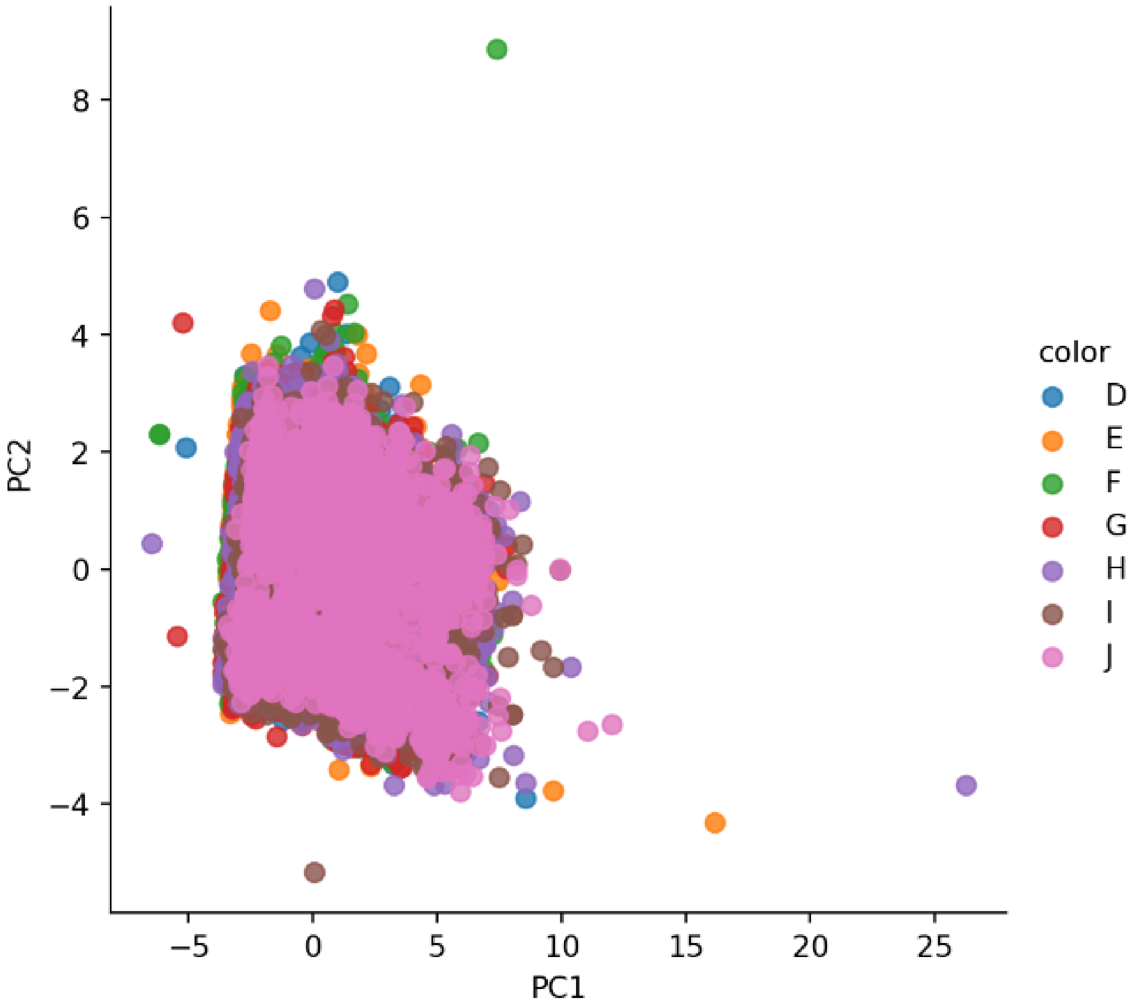
```
from sklearn.preprocessing import StandardScaler

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}
)
```

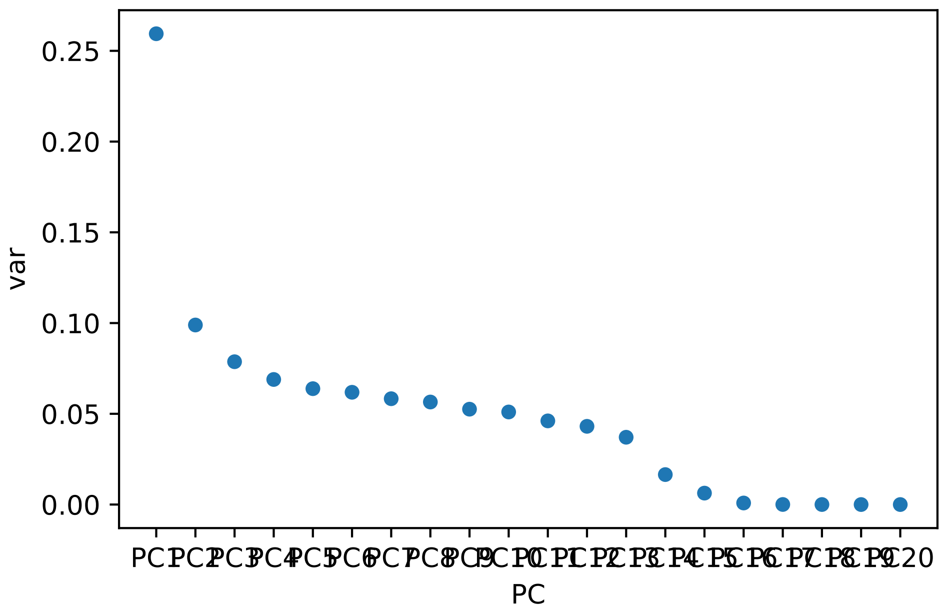


PCA clustering scree plot variance

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 Introduction

This is a book created from markdown and executable code.

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

4 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.