prince

Many variables, a few dimensions: PCA for numbers, FAMD for numbers and categories

Python cheatsheet · MATH60033A · Quantitative Methods in IB · used in session 13

```
import prince
```

fits like scikit-learn: .fit(df), then read attributes

1 · Which method?

All four turn many correlated columns into a few dimensions.

your columns are…	use	
only numbers	**prince.PCA**	or sklearn PCA
numbers + categories	**prince.FAMD**	session 13's wines
only categories	**prince.MCA**	survey answers
latent factors, rotated	**statsmodels**	Factor + varimax

Session 13 runs PCA with **scikit-learn** and factor analysis with **statsmodels**: both have their own cheatsheet. prince does the same PCA, and FAMD for mixed data.

2 · PCA — fit

```
df = movies[["budget", "popularity",
...]].dropna()
```

numeric columns only, no gaps.

```
pca = prince.PCA(n_components=4,
rescale_with_mean=True, rescale_with_std=True)
```

standardise (the two `rescale_` options): otherwise budget in dollars outweighs a rating out of 10.

```
pca = pca.fit(df)
```

PCA on unstandardised columns ranks variables by their units, not their information.

3 · The observations on the map

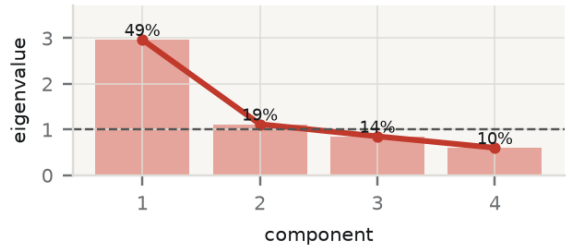
```
pca.row_coordinates(df)
```

every film's position on the components — a method: pass the data.

```
pca.row_contributions_
```

which observations pull a component most.

4 · How many components?




```
pca.eigenvalues_summary
```

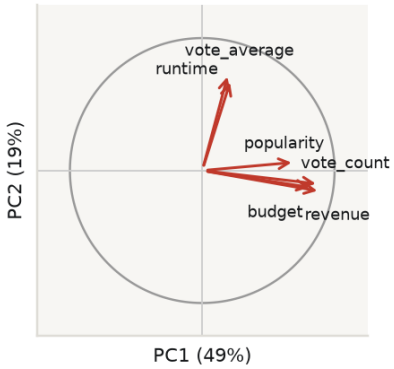
eigenvalue, % of variance and cumulative % per component — the table behind the scree plot.

```
pca.percentage_of_variance_
```

the same percentages, as numbers.

Keep components with eigenvalue above 1 (the dashed line), or up to the elbow of the scree plot; here two (68% of the variance).

5 · What the components mean




```
pca.column_correlations
```

each variable's correlation with each component: the arrows of the circle.

```
pca.column_contributions_
```

how much each variable builds each component (shares).

PC1 here is **size** (budget, revenue, votes); PC2 is **quality and length** (rating, runtime).

6 · FAMD — numbers and categories together

Session 13: the wines' chemistry (numbers) and their verdict (a category).

```
wine["quality_label"] = np.where(wine["good"] ==
1, "good", "ordinary")
```

the category as **text**: FAMD treats object columns as categorical.

```
famd = prince.FAMD(n_components=5,
random_state=0).fit(wine)
```

no rescaling options: FAMD standardises the numbers and weights the categories itself.

```
famd.eigenvalues_summary
```

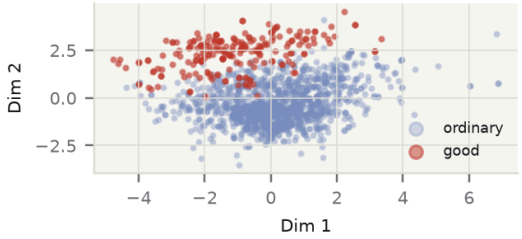
as for PCA.

```
famd.column_coordinates_
```

how much each variable — number or category — carries each dimension. An **attribute**: no brackets.

```
famd.row_coordinates(wine)
```

each wine on the dimensions, as plotted below.



Drop the 0/1 column once you have made it a label, or the same information enters twice.

7 · Pitfalls

```
column_coordinates_ vs row_coordinates(df)
```

the first is an attribute, the second a method that takes the data.

```
components are 0, 1, 2 …
```

prince counts from 0: column `0` is the first component.

```
a component is not a cause
```

it summarises correlation; naming it is your interpretation, to defend.