

Chapter 8

WORKING WITH LARGE DATASETS: DIMENSION REDUCTION

DATA ANALYSIS IN R

Outline

- Large Datasets: Related Problems & Solutions
- Dimensionality Reduction: Principle Components Analysis

Large Dataset

- **Data is often too large; reducing data can improve performance**
- **Data reduction consists of reducing the representation of the data set while producing the same (or almost the same) results**
- **Data reduction includes:**
 - Dimensionality reduction
 - Discretization
 -

Dimensionality Reduction

- **Curse of dimensionality**

When dimensionality increases data becomes increasingly sparse

Density and distance between points, which is critical to clustering outlier analysis becomes less meaningful

The possible combinations of subspaces will grow exponentially

- **Dimensionality reduction**

Avoid the curse of dimensionality

Help eliminate irrelevant features and reduce noise

Reduce time and space required in data mining

Allow easier visualization

- **Dimensionality reduction techniques**

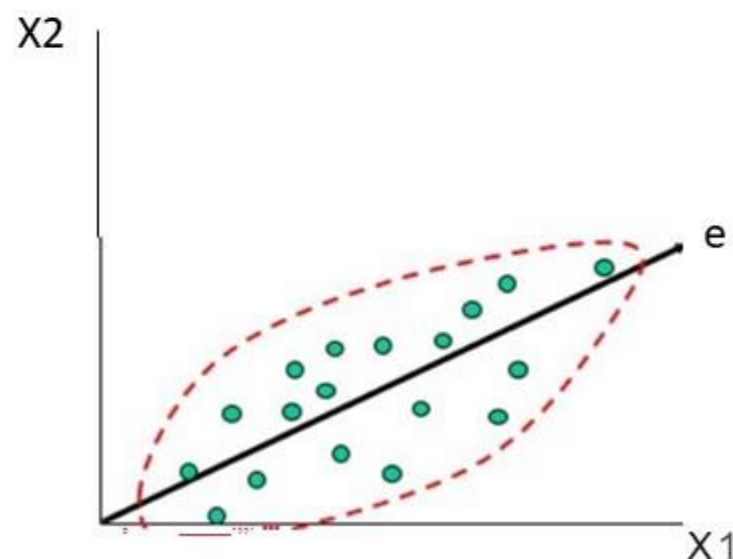
Principal Component Analysis

Attribute subset selection

Dimensionality Reduction: Principal Component Analysis (PCA)

- Find a projection that captures the largest amount of variation in data
- The original data are projected onto a much smaller space, resulting in dimensionality reduction

Done by finding the eigenvectors of the covariance matrix, and these eigenvectors define the new space. The eigenvector with the largest eigenvalue is the direction along which the data set has the maximum variance



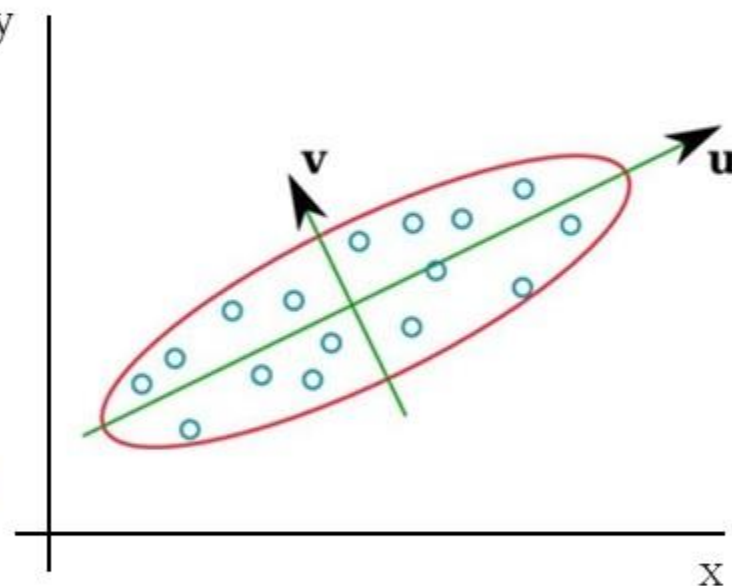
Dimensionality Reduction: Principal Component Analysis (PCA)

- **Principal Components are the underlying structure in the data.**
They are the directions where there is the most variance, the directions where the data is most spread out. We try to find the straight line that best spreads the data out when it is projected along it. This is the first principal component, the straight line that shows the most substantial variance in the data.

- **PCA is a type of linear transformation on a given data set that has values for a certain number of variables (coordinates) for a certain amount of spaces.**

This linear transformation fits this dataset to a new coordinate system in such a way that the most significant variance is found on the first coordinate and each subsequent coordinate is orthogonal to the last and has a lesser variance. In this way, you transform a set of x correlated variables over y samples to a set of p uncorrelated principal components over the same samples.

An **eigenvector** is a direction, such as "Vertical" or "45 degrees", while an **eigenvalue** is a number telling you how much variance there is in the data in that direction. The eigenvector with the highest eigenvalue is, therefore, the first principal component.



Dimensionality Reduction: Principal Component Analysis (PCA)

More eigenvalues and eigenvectors to be found in one dataset?

- That's correct!
- The number of eigenvalues and eigenvectors that exists is equal to the number of dimensions the data set has.
- We can reframe a dataset in terms of these eigenvectors and eigenvalues without changing the underlying information.
- Note that reframing a dataset regarding a set of eigenvalues and eigenvectors does not entail changing the data itself, you're just looking at it from a different angle, which should represent the data better.

PCA in R

- **mtcars dataset, which is built into R.**

This dataset consists of data on 32 models of car taken from an American motoring magazine (1974 Motor Trend magazine). For each car, you have 11 features, expressed in varying units (US units)

- **PCA works with numerical data, you'll exclude the two categorical variables (vs and am).**

You are left with a matrix of 9 columns and 32 rows which you pass to the `prcomp()` function, assigning your output to `mtcars.pca`. You will also set two arguments `center` and `scale`, to be `TRUE`. Then you can have a peek at your PCA object with `summary()`.

PCA in R

```
> mtcars.pca <- prcomp(mtcars[,c(1:7, 10, 11)], center = TRUE, scale = TRUE)
```

```
> summary(mtcars.pca)
```

Importance of components:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9
Standard deviation	2.3782	1.4429	0.71008	0.51481	0.42797	0.35184	0.32413	0.2419	0.14896
Proportion of Variance	0.6284	0.2313	0.05602	0.02945	0.02035	0.01375	0.01167	0.0065	0.00247
Cumulative Proportion	0.6284	0.8598	0.91581	0.94525	0.96560	0.97936	0.99103	0.9975	1.00000

You obtain 9 PC: PC1 explains 63% of the total variance, which means that nearly two thirds of the information in the dataset (9 variables) can be encapsulated by just that one Principal Component.

PC2 explains 23% of the variance. So, by knowing the position of a sample in relation to just PC1 and PC2, you can get a very accurate view on where it stands in relation to other samples, as Just PC1 and PC2 can explain 86% of the variance.

Plotting PCA

- **Make a biplot, which includes both the position of each sample in terms of PC1 and PC2 and also will show you how the initial variables map onto this.**

A biplot is a type of plot that will allow you to visualize how the samples relate to one another in our PCA (which samples are similar and which are different) and will simultaneously reveal how each variable contributes to each principal component.

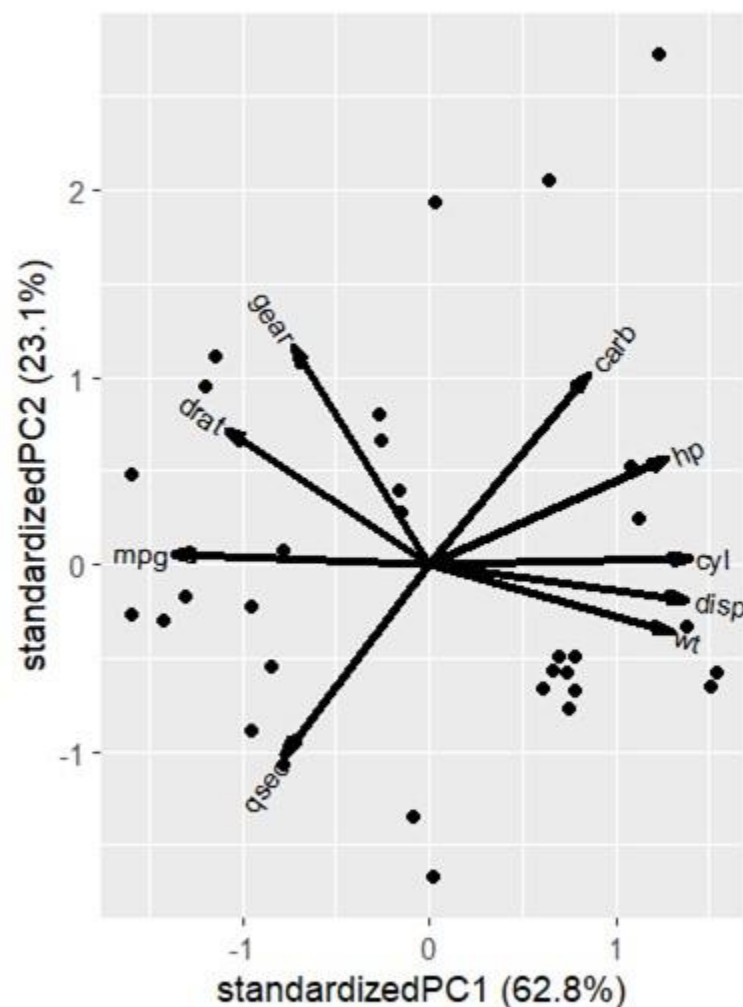
- **Use the ggbiplot package, which offers a user-friendly and pretty function to plot biplots.**
- **Install ggbiplot!**

```
library(devtools)  
install.packages("vqv/ggbiplot")  
library(ggbiplot)
```

Call ggbiplot on your PCA

```
ggbiplot(mtcars.pca)
```

- The axes are seen as arrows originating from the center point.
- Variables `hp`, `cyl`, and `disp` all contribute to PC1, with higher values in those variables moving the samples to the right on this plot.
- This lets you see how the data points relate to the axes, but it's not very informative without knowing which point corresponds to which sample (car).



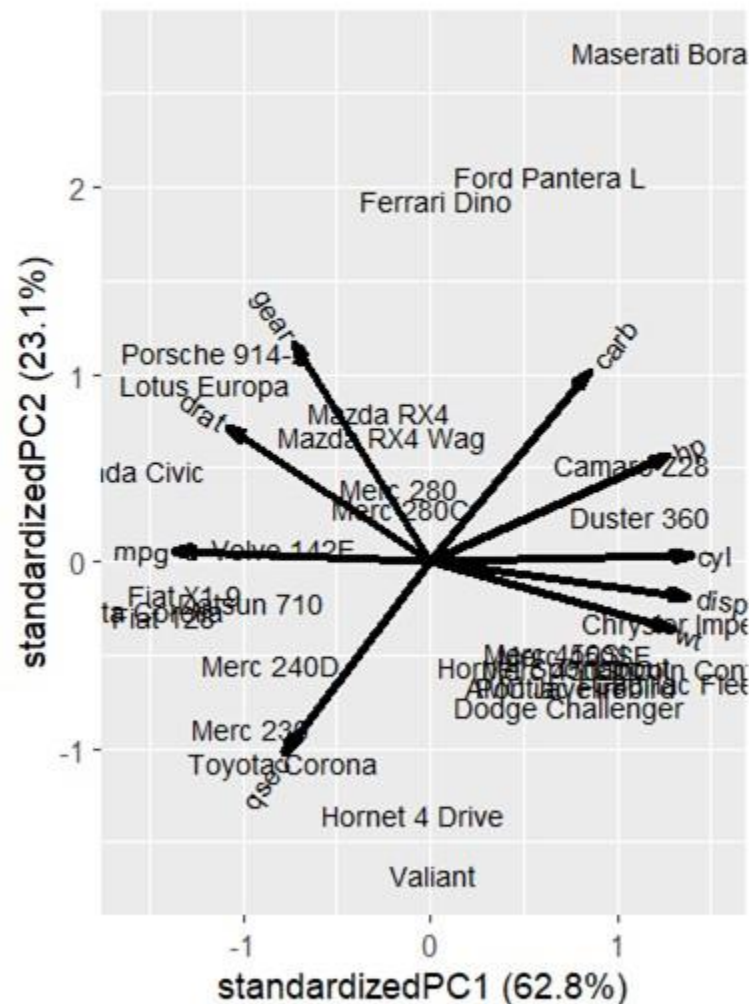
Call ggbiplot on your PCA

- Give to ggbiplot the rownames of mtcars as labels.

▶ This will name each point with the name of the car in question:

```
ggbiplot(mtcars.pca, labels = rownames(mtcars))
```

- Now you can see which cars are similar to one another.
- For example, the Maserati Bora, Ferrari Dino and Ford Pantera L all cluster together at the top. This makes sense, as all of these are sports cars.



Interpreting the results PC1 & PC2

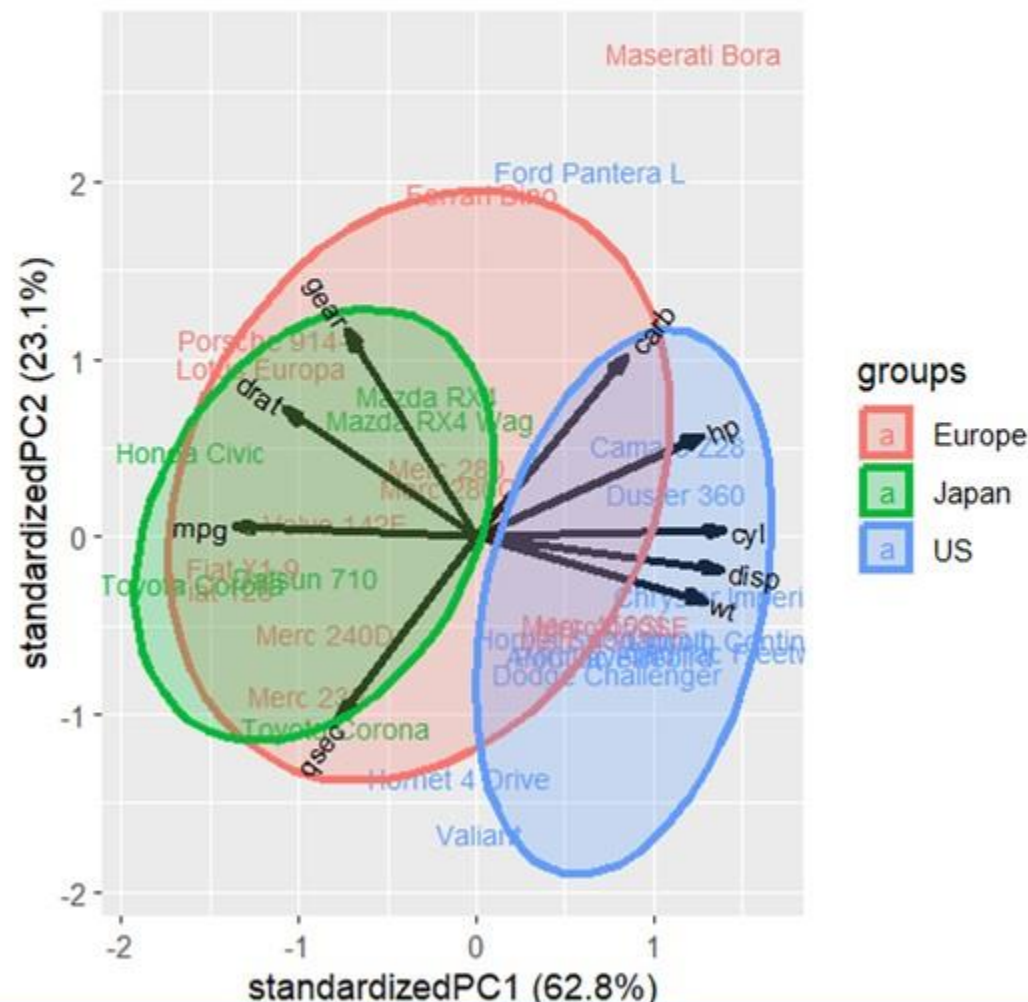
```
mtcars.country <- c(rep("Japan", 3), rep("US", 4), rep("Europe", 7), rep("US", 3), "Europe", rep("Japan", 3),  
rep("US", 4), rep("Europe", 3), "US", rep("Europe", 3))  
ggbiplot(mtcars.pca, ellipse=TRUE, labels=rownames(mtcars), groups=mtcars.country)
```

- **Look at the origin of each of the cars.**

- ▶ put them into one of three categories (categories?), one each for the US, Japanese and European cars.
- ▶ make a list for this info, then pass it to the groups argument of ggbiplot.
- ▶ set the ellipse argument to be TRUE, which will draw an ellipse around each group.

The American cars form a distinct cluster to the right. Looking at the axes, you see that the American cars are characterized by high values for cyl, disp, and wt.

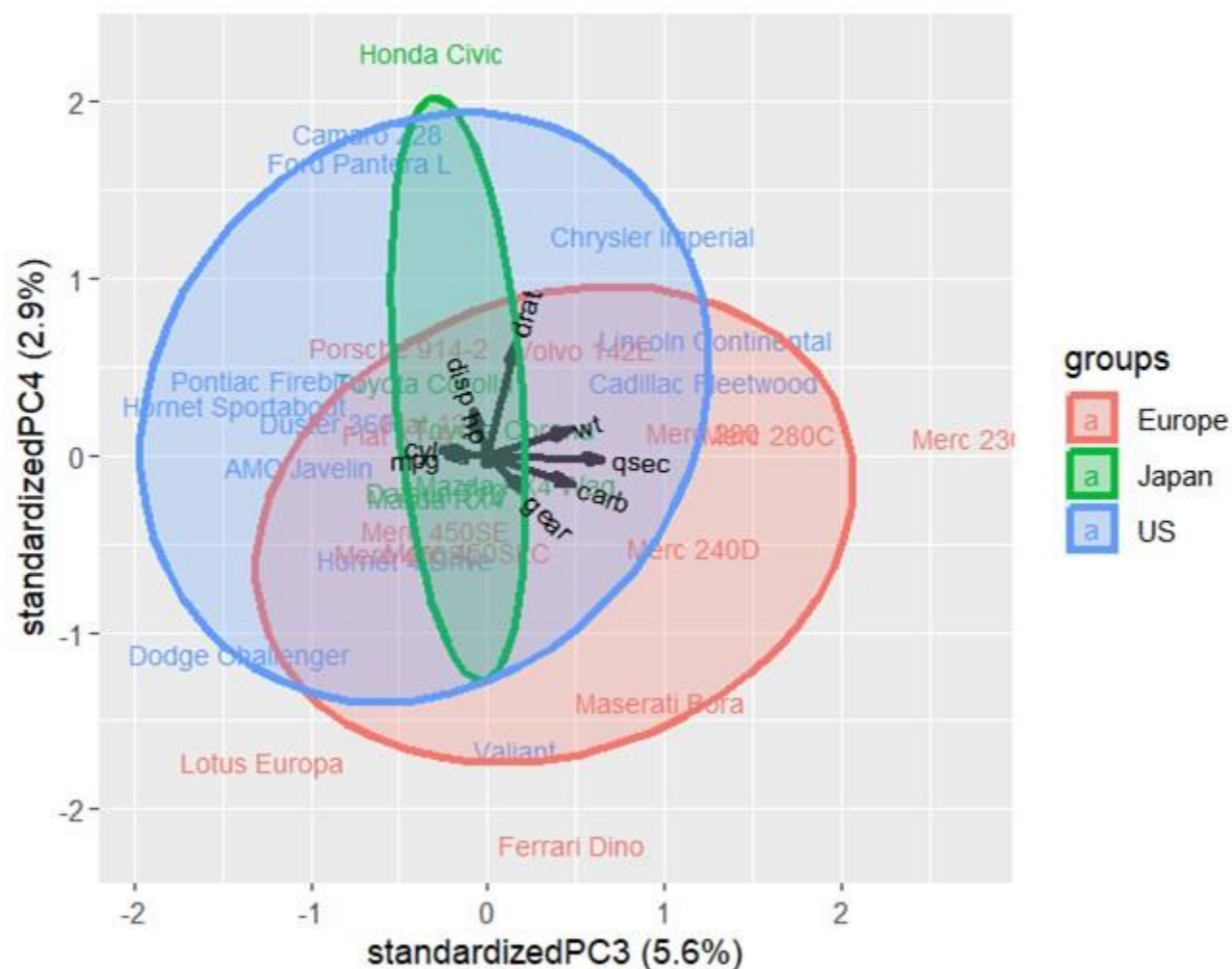
Japanese cars, on the other hand, are characterized by high mpg. European cars are somewhat in the middle and less tightly clustered than either group.



Interpreting the results PC3 & PC4

```
ggbiplot(mtcars.pca,ellipse=TRUE,choices=c(3,4), labels=rownames(mtcars), groups=mtcars.country)
```

- You don't see much here, but this isn't too surprising. PC3 and PC4 explain very small percentages of the total variation, so it would be surprising if you found that they were very informative and separated the groups or revealed apparent patterns.



Questions