Exploring Multivariate Data with Clustering and Dimensionality Reduction

Marco Baroni

Practical Statistics in R

Outline

Introduction

Clustering

Clustering in R

Dimensionality reduction

Dimensionality reduction in R

Outline

Introduction

Clustering

Clustering in R

Dimensionality reduction

Dimensionality reduction in R

Clustering and dimensionality reduction

- Techniques that are typically appropriate when:
  - You do not have an obvious dependent variable
  - You have many, possibly correlated variables
- Clustering:
  - Group the observations into *n* groups based on how they pattern with respect to the measured variables
- Dimensionality reduction
  - Find fewer "latent" variables with a more general interpretation based on the patterns of correlation among the measured variables

Outline

Introduction

Clustering

- k-means

Clustering in R

Dimensionality reduction

Dimensionality reduction in R

(Hard partitional) clustering

- We only explore here:
  - **Hard** clustering: an observation can belong to one cluster only, no distribution of a single observation across clusters
    - PCA below can be interpreted as a form of soft clustering
  - **Partitional** clustering: “flat” clustering into *n* classes, no hierarchical structure
    - Look at `?hclust` for a basic R implementation of the hierarchical alternative
- Hard partitional clustering has many drawbacks, but it leads to clear-cut, straightforwardly interpretable results (which is part of what causes the drawbacks)

Why clustering?

- Perhaps you really do not know what are the underlying classes in which your observations should be grouped
  - E.g., which areas of the brain have similar patterns of activation in response to a stimulus?
  - Do children cluster according to different developmental patterns?
- You know the “true” classes, but you want to see whether the distinction between them would emerge from the variables you measured
  - Will a distinction between natural and artificial entities arise simply on the basis of color and hue features?
  - Is the distinction between nouns, verbs and adjectives robust enough to emerge from simple contextual cues alone?
- When you do not know the true classes, interpretation of the results will obviously be very tricky, and possibly circular

Logistic regression and clustering

Supervised and unsupervised learning

- In (binomial or multinomial) logistic regression (*supervised learning*), you are given the labels (classes) of the observations, and you use them to tune the features (independent variables) so that they will maximize the distinction between observations belonging to different classes
  - You go from the classes to the optimal feature combination
  - The dependent variable is given and you tune the independent variables
- In clustering (*unsupervised learning*), you are *not* given the labels, and you must use some goodness-of-fit criterion that does not rely on the labels to reconstruct them
  - You go from the features to the optimal class assignment
  - The independent variables are fixed and you tune the dependent variable
    - Although as part of this process you can also reweight the independent variables, of course!

Logistic regression and clustering

Supervised and unsupervised learning

- Unsupervised learning might be a more realistic model of what children do when acquiring language and other cognitive skills
- . . . although the majority of work in machine learning focuses on the supervised setting
  - Better theoretical models, better quality criteria, better empirical results

Outline

Introduction

Clustering

- k-means

Clustering in R

Dimensionality reduction

Dimensionality reduction in R

k-means

- One of the simplest and most widely used hard partitional clustering algorithms
- For more sophisticated options, see the *cluster* and *e1071* packages

k-means

- The basic algorithm
  1. Start from *k* random points as cluster centers
  2. Assign points in dataset to cluster of closest center
  3. Re-compute centers (means) from points in each cluster
  4. Iterate cluster assignment and center update steps until configuration converges (e.g., centers stop moving around)
- Given random nature of initialization, it pays off to repeat procedure multiple times (or to start from “reasonable” initialization)

Illustration of the k-means algorithm

See `?iris` for more information about the data set used

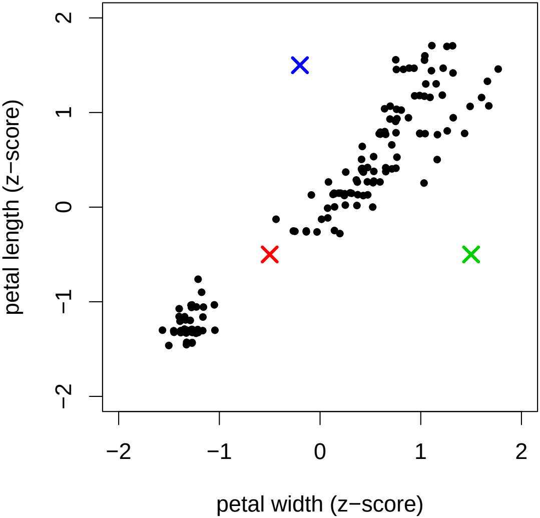


Illustration of the k-means algorithm

See `?iris` for more information about the data set used

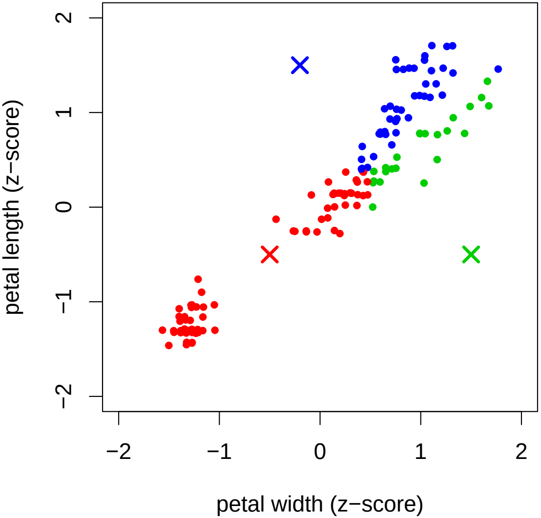


Illustration of the k-means algorithm

See `?iris` for more information about the data set used

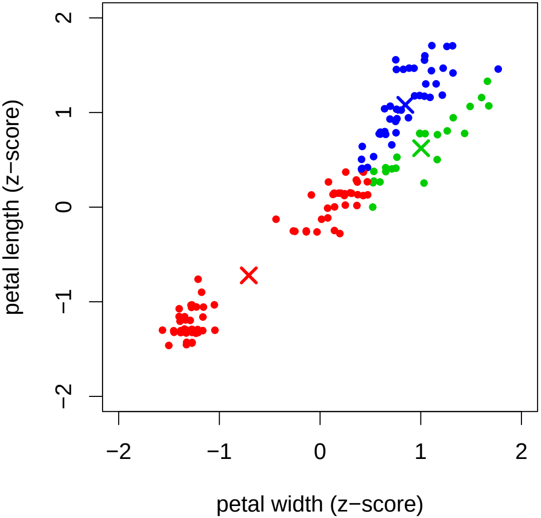


Illustration of the k-means algorithm

See `?iris` for more information about the data set used

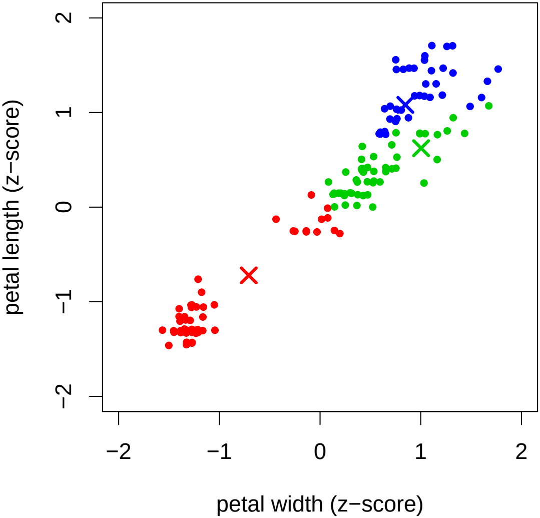


Illustration of the k-means algorithm

See `?iris` for more information about the data set used

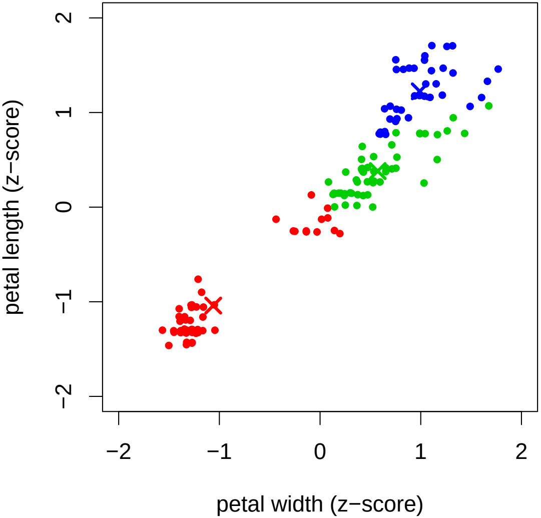


Illustration of the k-means algorithm

See `?iris` for more information about the data set used

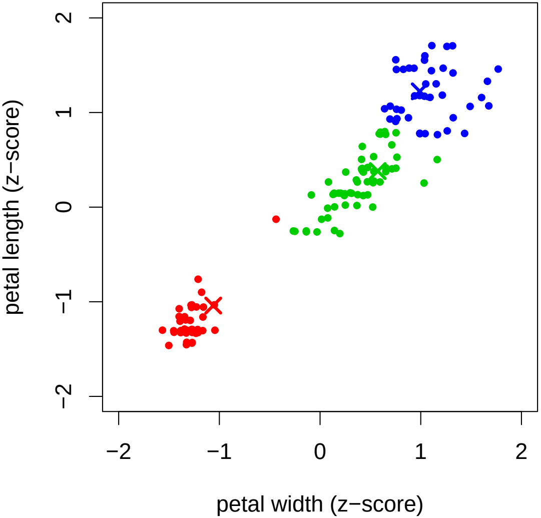


Illustration of the k-means algorithm

See `?iris` for more information about the data set used

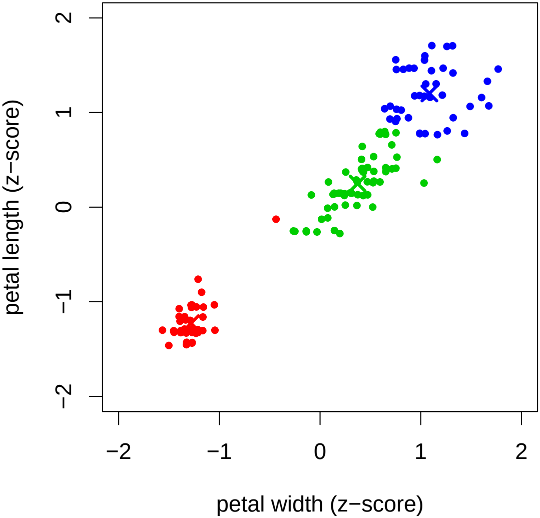


Illustration of the k-means algorithm

See `?iris` for more information about the data set used

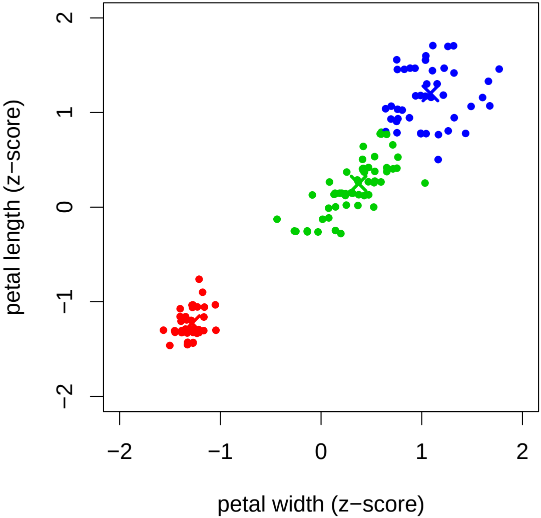


Illustration of the k-means algorithm

See `?iris` for more information about the data set used

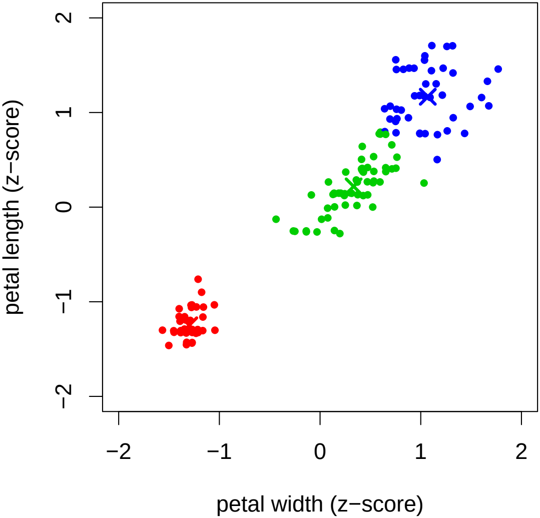


Illustration of the k-means algorithm

See `?iris` for more information about the data set used

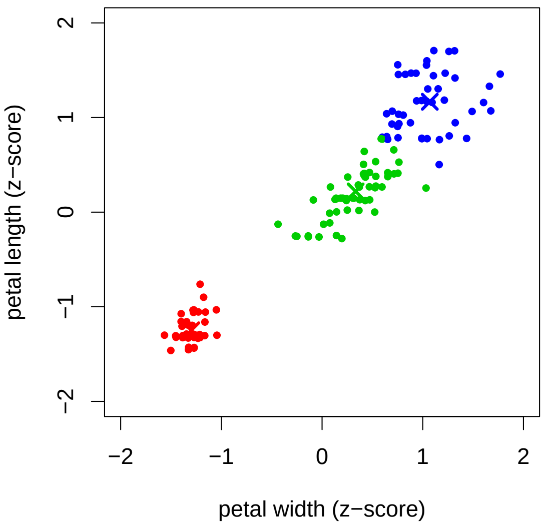
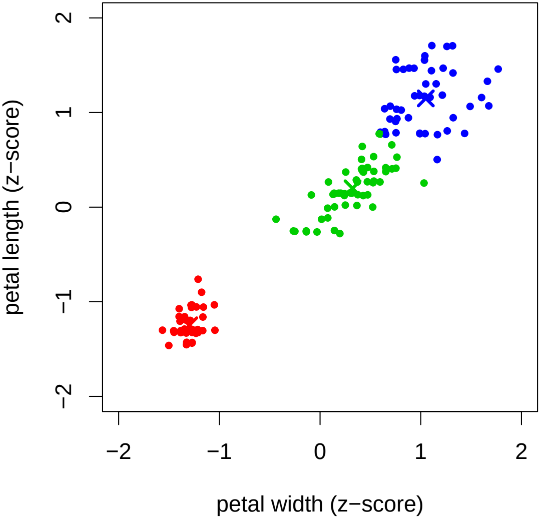


Illustration of the k-means algorithm

See `?iris` for more information about the data set used



How many clusters?

- When clustering is exploratory (we do not want to reconstruct the labels we know, we want to see which classes emerge from the data) setting k is a big issue
- Classic approach to find optimal k, given measure of clustering fit (typically measuring intra-cluster similarity):
  - Try clustering with a range of ks
  - Pick k that optimizes fit

Outline

Introduction

Clustering

Clustering in R

Dimensionality reduction

Dimensionality reduction in R

The concrete concept dataset

- 43 concrete concepts from the subject-generated norms of:
  - McRae, K., Cree, G., Seidenberg, M., & McNorgan, C. (2005). Semantic feature production norms for a large set of living and nonliving things. *Behavior Research Methods*, 37, 547–559.
- Macro-categories of properties from ongoing work with Gerhard Kremer and Alessandro Lenci
- Download and unpack `r-data-2.zip`
- Load `concrete-concepts.txt`, attach data and take a quick look at them

The concrete concept dataset

CONCEPT	elephants, lettuce, ship...
CLASS6	bird, fruit, green, groundAnimal, tool, vehicle
CLASS3	animal, artifact, vegetable
CLASS2	natural, manMade
FEATURE	an_animal, is_edible, made_of_metal...
TYPE	(of feature) behaviour, category, context, function, part, quality, related

Creating a concept by feature matrix

```
# table() will count the number of times each feature was produced
# for each concept

# The columns of the resulting matrix are the
# variables we will use for clustering

> concept_by_feature<-table(CONCEPT,FEATURE)

> concept_by_feature[1:4,1:4]
```

Clustering into 6 classes on the basis of the feature distribution

```
# [,] forces R to treat our table as a matrix

> partition6<-kmeans(concept_by_feature[,],6)

> partition6$cluster
```

Exploring the solution

```
# Unique concept-class6 pairs

> c6<-unique(cbind(as.character(CONCEPT),
  as.character(CLASS6)))
> head(c6)

# Class by cluster

> table(c6[,2],partition6$cluster)

                 1  2  3  4  5  6
  bird           0  0  0  0  0  7
  fruit          4  0  0  0  0  0
  green          0  0  4  0  0  0
  groundAnimal   0  1  0  0  7  0
  tool           0 13  0  0  0  0
  vehicle        0  3  0  4  0  0
```

Exploring the solution

```
# The ground animal in the tool cluster

> c6[,1][c6[,2]=="groundAnimal" &
  partition6$cluster==2]
[1] "snail"

# The features with the highest values on the centroids

> head(partition6$centers[1,][order(
  partition6$center[1,],decreasing=TRUE)],20)
> head(partition6$centers[2,][order(
  partition6$center[2,],decreasing=TRUE)],20)

# etc. (I wished there was an easier way to sort in R!)
```

Trying multiple starts

```
> partition6_100starts<-kmeans(
  concept_by_feature[,],6,nstart=100)

# The clusters got more tight, as shown by the total within-cluster
# sum of squares:

> sum(partition6$withinss)
[1] 61715.47
> sum(partition6_100starts$withinss)
[1] 61498.13

# However, no obvious improvement in clustering quality:

> table(c6[,2],partition6$cluster)
> table(c6[,2],partition6_100$cluster)
```

Practice

- Try clustering into the 3- and 2-way superordinate classes
  - Repeat the same analyses, but remember to compare the results against CLASS3 and CLASS2
- Try clustering by feature TYPEs
  - Can the simple information that, e.g., a concept has many functional features reveal that the concept is a tool?

Outline

Introduction

Clustering

Clustering in R

Dimensionality reduction

- PCA

Dimensionality reduction in R

Dimensionality reduction

- We measured *n* variables, but we reduce them to *k* “latent” variables
- From a $m \times n$ matrix to a $m \times k$ matrix, where $k << n$
- Typically, latent variables can be interpreted as generalizations of the patterns in the observed variables
- Why?
    - To be able to visually inspect trends in the data (especially if $k = 2$)
    - Hope that latent dimensions will capture “deeper” patterns of correlation
    - Efficiency/storage
        - Resulting matrix will be easier to store and process, but most dimensionality reduction procedures require the full matrix as input and are computationally intensive!

Outline

Introduction

Clustering

Clustering in R

Dimensionality reduction

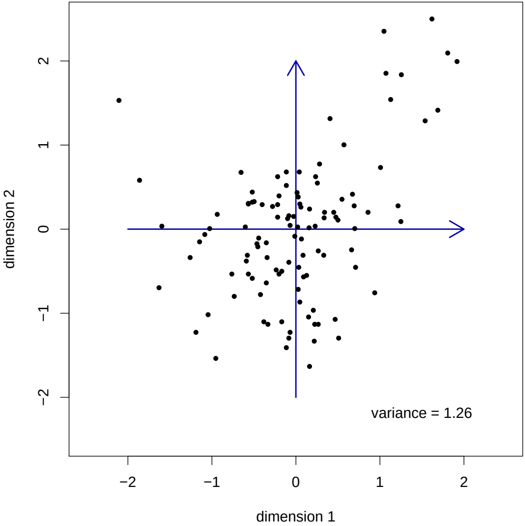
- PCA

Dimensionality reduction in R

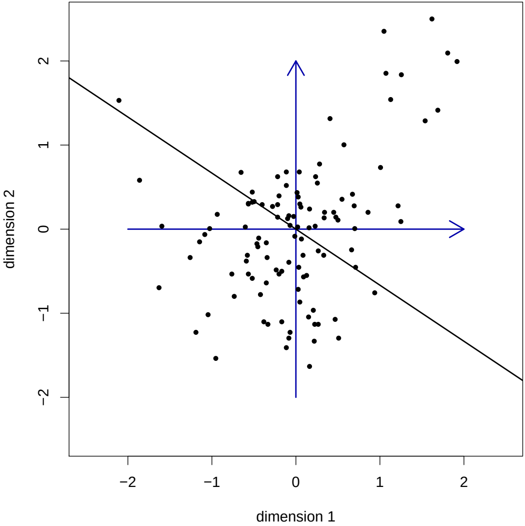
Principal component analysis (PCA)

- One of the oldest and most commonly used dimensionality reduction techniques
- Find a set of orthogonal dimensions such that the first dimension “accounts” for the most *variance* in the original dataset, the second dimension accounts for as much as possible of the remaining variance, etc.
- The top *k* dimensions (principal components) are the best subset of *k* dimensions to approximate the spread in the original dataset
  - I.e., they are the *k* orthogonal dimensions in which the projections of the original data-points (observations) have the largest variance
- Correlation of original variables to principal components might reveal interesting underlying factors

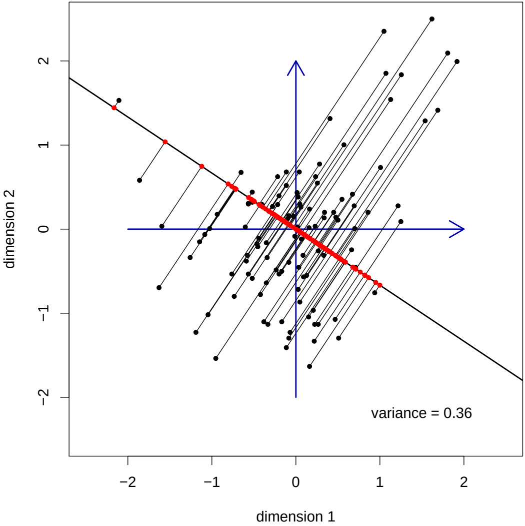
Preserved variance: examples



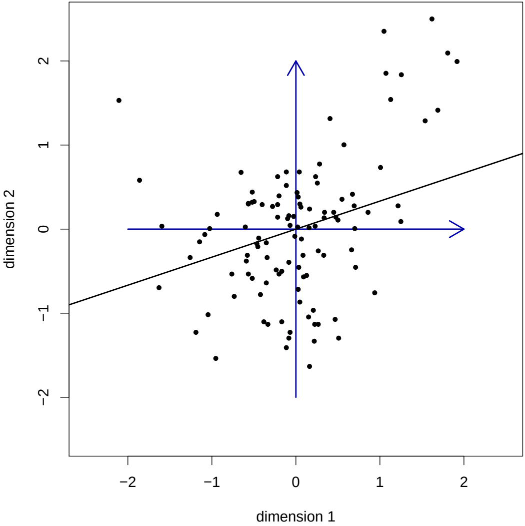
Preserved variance: examples



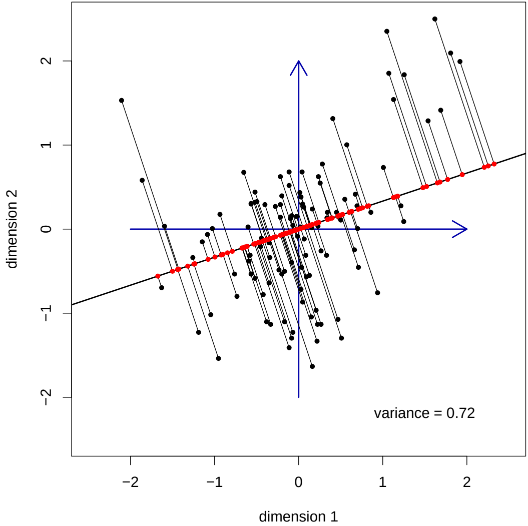
Preserved variance: examples



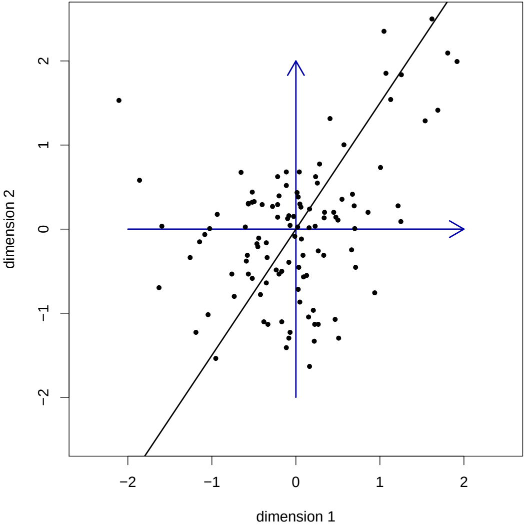
Preserved variance: examples



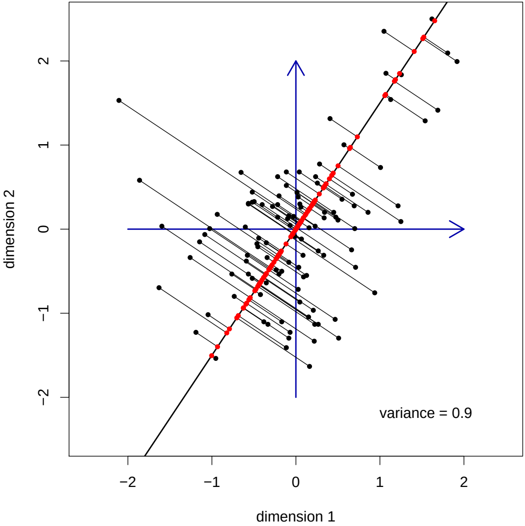
Preserved variance: examples



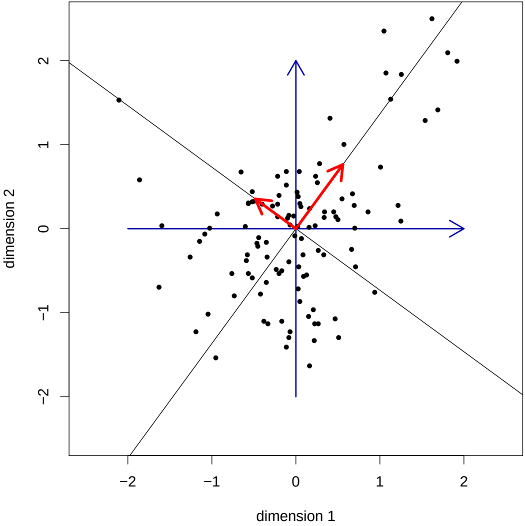
Preserved variance: examples



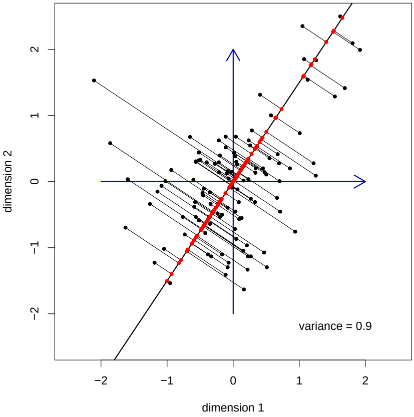
Preserved variance: examples

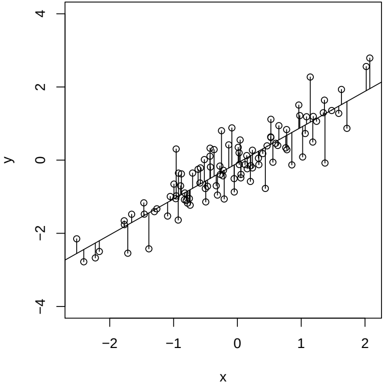


Adding an orthogonal dimension



NB: PCA vs. least squares line fitting

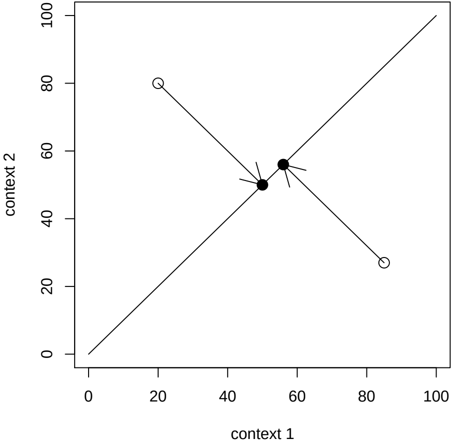




Dimensionality reduction as generalization

- (Simplifying somewhat,) correlated variables will be (partially) collapsed onto same dimensions in reduced space
    - In the concept description norms, “has feathers” and “flies” might be both highly correlated with a reduced space “birdness” dimension (but “has feathers” might also be correlated to a “part” dimension)
    - Pattern of co-activation of voxels might reveal larger functionally correlated areas that are mapped to same reduced dimensions

Dimension reduction as generalization



Dimensionality reduction as soft clustering

- In some lucky cases, the reduced space dimensions can be interpreted as categories (the “birdness” dimension, the “toolness” dimension)
- Then, the principal components (reduced space orthogonal dimensions) can be seen as clusters, and the values of the original points when projected in the new dimensions can be interpreted as the “degree of membership” of the points in each cluster
  - E.g., a horse might have high values on both the “animal” and the “vehicle” dimensions
- Of course, you can also run standard hard clustering using the reduced dimensions as features!

PCA and SVD

- Principal components are typically extracted using a technique called Singular Value Decomposition
- Given original observation matrix M, SVD decomposes it into:

$$M = U\Sigma V^T$$

- First k columns of $U\Sigma$ give projections of target words into reduced space
- V is the eigenvector matrix, specifying the correlation of each original variable with each principal component
- In R, it is instructive to reproduce the `rotation` and `x` contents of an object created by `prcomp()` with the matrices created by `svd()`

How many *k*s?

- If purpose is plotting, we will use top 3 or 2 principal components
    - It might make sense to look at multiple 2-dimensional plots: first vs. second component, second vs. third, etc.
- Heuristic criteria to choose *k*:
    - Pick minimum number of dimensions that have *n*% of original variance (e.g., 90%)
    - Look at histogram of variance on each dimension, cut where you see a sharp decrease

Beyond PCA

- Loads of other dimensionality reduction techniques
- Two trendy ones: Independent Component Analysis and Positive Matrix Factorization
- When the issue is scaling up, consider Random Indexing

Outline

Introduction

Clustering

Clustering in R

Dimensionality reduction

Dimensionality reduction in R

Back to the concrete noun dataset

- If you haven't already, load `concrete-concepts.txt` and attach
- Create a concept by feature table as above:

```
> concept_by_feature<-table(CONCEPT,FEATURE)
```

PCA in R

```
# Centering is important (but prcomp() does it by default)

> c_by_f.pca<-prcomp(concept_by_feature,
  center=TRUE,scale=TRUE)

# Variance of the top principal components

> summary(c_by_f.pca)
Importance of components:
                          PC1    PC2    PC3   ...
Standard deviation     4.1610 4.0027 3.9239   ...
Proportion of Variance 0.0465 0.0431 0.0414   ...
Cumulative Proportion  0.0465 0.0896 0.1310   ...
# NB: 43 components because we have 43 observations (concepts)
# and thus maximally 43 orthogonal dimensions

# Variance plot

> plot(c_by_f.pca)
```

Looking inside the PCA object

```
> head(c_by_f.pca$sdev)
[1] 4.161032 4.002719 3.923918 3.746610 3.705787 3.628330

> c_by_f.pca$rotation[1:3,1:3]
                           PC1         PC2         PC3
a_baby_is_a_kitten -0.04963951 0.034148523 -0.02587115
a_baby_is_a_piglet -0.10091012 0.090070350 -0.08258978
a_bird             -0.04648581 0.007532426  0.01030939

> c_by_f.pca$x[1:3,1:2]
CONCEPT        PC1        PC2
  banana 0.2973792 -1.4367816
  boat   0.2261732 -0.7561464
  bottle 1.9571371 -3.1795953
```

Looking inside the PCA object

```
# Original variables most associated with the third component

> head(sort(c_by_f.pca$rotation[,3],decreasing=TRUE),3)
     a_vegetable grows_in_gardens        is_edible
       0.1357469        0.1252715        0.1203748

# Concepts most associated with the third component

> head(sort(c_by_f.pca$x[,3],decreasing=TRUE),5)
    lettuce      potato   pineapple    mushroom screwdriver
  10.075616    7.208782    6.725311    3.889524    3.363768

# Don’t ask me why here sort() works as I would like it to...
```

Visualizing the reduced space

```
# In principle biplots are very useful, but with so many
# original variables we just get a beautiful mess:

> biplot(c_by_f.pca)

# Manual plots of the points on PC1 vs. PC2
# and PC1 vs. PC3 dimensions

> c6<-unique(cbind(as.character(CONCEPT),
  as.character(CLASS6)))

> plot(c_by_f.pca$x[,1], c_by_f.pca$x[,2],type="n")
> text(c_by_f.pca$x[,1],c_by_f.pca$x[,2],labels=c6[,1],
  col=rank(c6[,2]))

> plot(c_by_f.pca$x[,1],c_by_f.pca$x[,3],type="n")
> text(c_by_f.pca$x[,1],c_by_f.pca$x[,3],labels=c6[,1],
  col=rank(c6[,2]))

# Try also adding a few of the original features
```

Clustering in reduced space

```
> partition6<-kmeans(c_by_f.pca$x[,1:30],6)

> table(c6[,2],partition6$cluster)
```

# Compare to results obtained with the full matrix

Concept by type PCA

Just for the sake of producing a readable biplot

```
> concept_by_type<-table(CONCEPT,TYPE)

> concept_by_type<-prcomp(concept_by_type,
  center=TRUE,scale=TRUE)

> biplot(concept_by_type)
```

The preschoolers' dataset

- Data provided by Alessandro Chinello, from ongoing work with Cattani, Bonfiglioli and Piazza
- The development of parietal lobe in preschoolers
- One research question: what are the patterns of correlations between various cognitive ability indices measured on preschoolers? Do they group into sets corresponding to broader functional (and neural) classes?
- Clean up workspace, detach, load `preschoolers.txt` dataset, take a look at it, create version without NAs:

```
nona<-na.omit(d)
```

The preschoolers’ dataset

SUBJECT	subject id
AGE	age in months
FINGER	error count in finger discrimination
SPAN	max number of visual elements that can be memorized
ATTENTION	difference between RTs in congruent vs. incongruent cue-target conditions
DFACE	D prime measure of face sensitivity
DOBJ	D prime measure of object sensitivity
NUMBER	Weber fraction in point quantity discrimination task (the lower, the better the discrimination)
GRASPING	Maximum thumb-index distance when grasping objects

PCA of the preschoolers’ data

```
> ps.pca<-prcomp(nona[,3:9],center=TRUE,scale=TRUE)

> summary(ps.pca)

> plot(ps.pca)

> biplot(ps.pca)

# More meaningful to plot subject age

> biplot(ps.pca,xlabs=nona$AGE)

# Do it yourself:

> plot(ps.pca$x[,1],ps.pca$x[,2],type="n")
> text(ps.pca$rotation[,1]*4,ps.pca$rotation[,2]*4,
  labels=names(nona)[3:9],col="grey",cex=1.5)
> text(ps.pca$x[,1],ps.pca$x[,2],labels=nona$AGE)
```

Our last practice: multidimensional scaling

- *Multi-dimensional scaling* (MDS) is another popular dimensionality reduction technique, that attempts to preserve the distances between points as faithfully as possible in the reduced space
- It is mostly used for visualization purposes
- Perform an MDS analysis of the concrete concept data, based on the sets of cues we described

MDS Practice

1. MDS operates on a *distance* matrix, a symmetric matrix of distances between each point in the dataset and each other point
   - Look at the documentation for the `dist()` function, and generate distance matrices from the original concept-by-feature table, using two different methods to compute distance
2. Perform MDS with the `cmdscale()` function: take a look at its documentation, and run MDS on each of your distance matrices
   - For further exploration of MDS, take a look at `sammon()` and `isoMDS()` from the `MASS` package
3. Plot the concepts in the first two dimensions produced by the MDS analyses, using different colours for different classes