Exercises: Principal Component Analysis

In this exercise, you will know

- How to use PCA
- How to use PCA for mixed data
- How to use neural network for dimension reduction

Don't forget to change your working directory!

1 Principal component analysis

We perform PCA on the `USArrests` data set, which is part of the base R package. The rows of the data set contain the 50 states, in alphabetical order.

```
states = row.names(USArrests)
states
```

```
##  [1] "Alabama"        "Alaska"         "Arizona"        "Arkansas"
##  [5] "California"     "Colorado"       "Connecticut"    "Delaware"
##  [9] "Florida"        "Georgia"        "Hawaii"         "Idaho"
## [13] "Illinois"       "Indiana"        "Iowa"           "Kansas"
## [17] "Kentucky"       "Louisiana"      "Maine"          "Maryland"
## [21] "Massachusetts"  "Michigan"       "Minnesota"      "Mississippi"
## [25] "Missouri"       "Montana"        "Nebraska"       "Nevada"
## [29] "New Hampshire"  "New Jersey"     "New Mexico"     "New York"
## [33] "North Carolina" "North Dakota"   "Ohio"           "Oklahoma"
## [37] "Oregon"         "Pennsylvania"   "Rhode Island"   "South Carolina"
## [41] "South Dakota"   "Tennessee"      "Texas"          "Utah"
## [45] "Vermont"        "Virginia"       "Washington"     "West Virginia"
## [49] "Wisconsin"      "Wyoming"
```

The columns of the data set contain the four variables.

```
names(USArrests)
```

```
## [1] "Murder"   "Assault"  "UrbanPop" "Rape"
```

We first briefly examine the data. We notice that the variables have vastly different means.

```
apply(USArrests, 2, mean)
```

```
##   Murder  Assault UrbanPop     Rape
##    7.788  170.760   65.540   21.232
```

Note that the `apply()` function allows us to apply a function - in this case, the `mean()` function - to each row or column of the data set. The second input here denotes whether we wish to compute the mean of the rows, 1, or the columns, 2. We see that there are on average three times as many rapes as murders, and more than eight times as many assaults as rapes. We can also examine the variances of the four variables using the `apply()` function.

```
apply(USArrests, 2, var)
```

```
##     Murder    Assault   UrbanPop       Rape
##   18.97047 6945.16571  209.51878   87.72916
```

Not surprisingly, the variables also have vastly different variances: the `UrbanPop` variable measures the percentage of the population in each state living in an urban area, which is not a comparable number to the number of rapes in each state per 100,000 individuals. If we failed to scale the variables before performing PCA, then most of the principal components that we observed would be driven by the `Assault` variable, since it has by far the largest mean and variance. Thus, it is important to standardize the variables to have mean zero and standard deviation one before performing PCA.

We now perform principal components analysis using the `prcomp()` function, which is one of several functions in R that perform PCA.

```
pr.out = prcomp(USArrests, scale = TRUE)
```

By default, the `prcomp()` function centres the variables to have mean zero. By using the option `scale = TRUE`, we scale the variables to have standard deviation one. The output from `prcomp()` contains a number of useful quantities.

```
names(pr.out)
```

```
## [1] "sdev"     "rotation" "center"   "scale"    "x"
```

The center and scale components correspond to the means and standard deviations of the variables that were used for scaling prior to implementing PCA.

```
pr.out$center
```

```
##   Murder  Assault UrbanPop     Rape
##    7.788  170.760   65.540   21.232
```

```
pr.out$scale
```

```
##    Murder   Assault  UrbanPop      Rape
##  4.355510 83.337661 14.474763  9.366385
```

The rotation matrix provides the principal component loadings; each column of `pr.out$rotation` contains the corresponding principal component loading vector.

```
pr.out$rotation
```

```
##                  PC1        PC2        PC3         PC4
## Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
## Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
## UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
## Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
```

We see that there are four distinct principal components. This is to be expected because there are in general $\min(n-1, p)$ informative principal components in a data set with n observations and p variables.

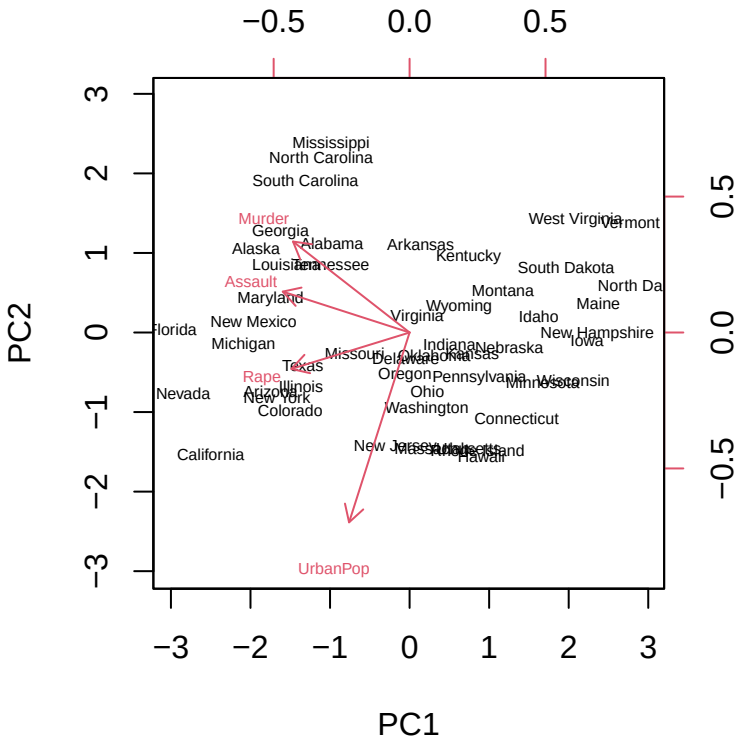
Using the `prcomp()` function, we do not need to explicitly multiply the data by the principal component loading vectors in order to obtain the principal component score vectors. Rather the 50×4 matrix `x` has as its columns the principal component score vectors. That is, the k-th column is the k-th principal component score vector.

```
dim(pr.out$x)
```

```
## [1] 50  4
```

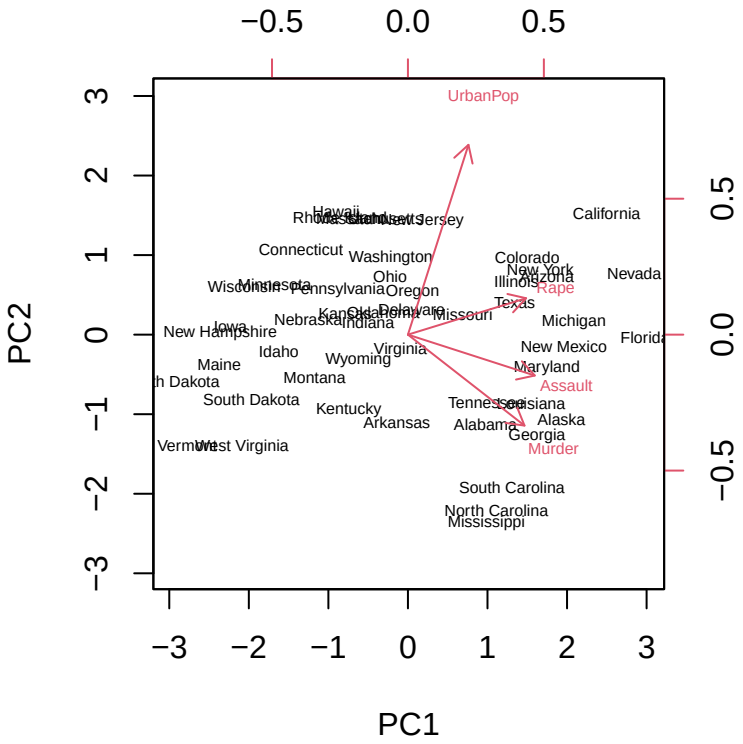
We can plot the first two principal components as follows:

```
biplot(pr.out, scale = 0,cex=0.5)
```



The `scale=0` argument to `biplot()` ensures that the arrows are scaled to represent the loadings; other values for scale give slightly different biplots with different interpretations. Recall that the principal components are only unique up to a sign change, so we can produce the figure by making a few small changes:

```
pr.out$rotation = -pr.out$rotation
pr.out$x = -pr.out$x
biplot(pr.out, scale =0,cex=0.5)
```



The `prcomp()` function also outputs the standard deviation of each principal component. For instance, on the `USArrests` data set, we can access these standard deviations as follows:

```
pr.out$sdev
```

```
## [1] 1.5748783 0.9948694 0.5971291 0.4164494
```

The variance explained by each principal component is obtained by squaring these:

```
pr.var = pr.out$sdev ^2
pr.var
```

```
## [1] 2.4802416 0.9897652 0.3565632 0.1734301
```

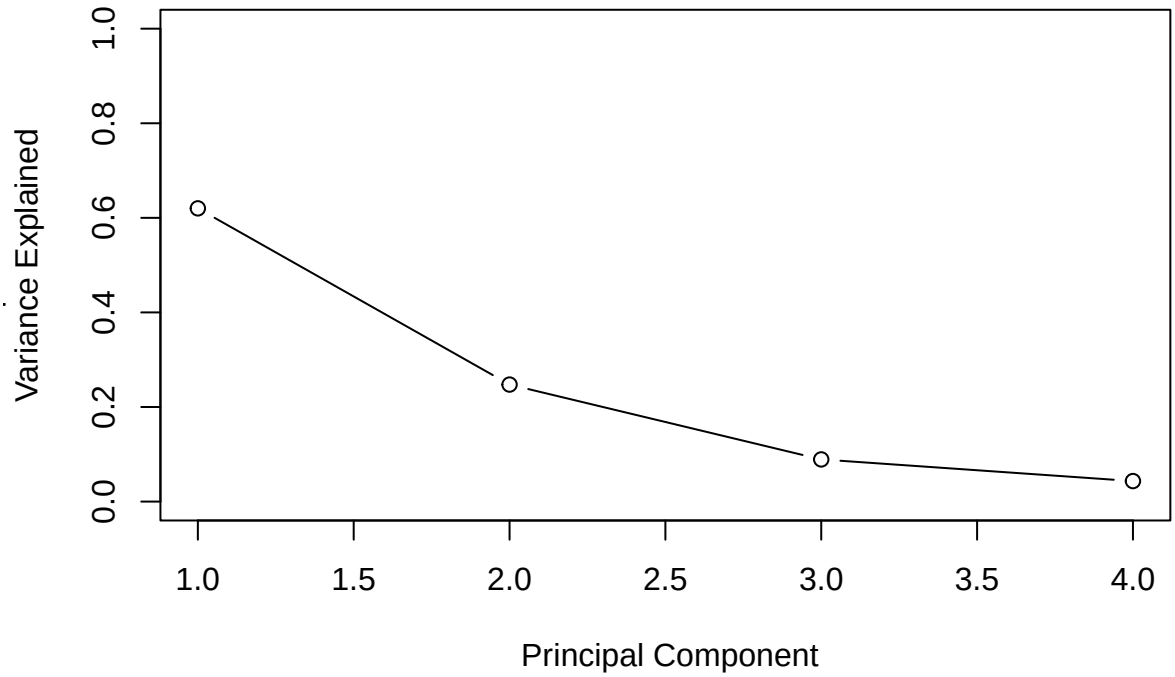
To compute the proportion of variance explained by each principal component, we simply divide the variance explained by each principal component by the total variance explained by all four principal components:

```
pve = pr.var/sum(pr.var)
pve
```

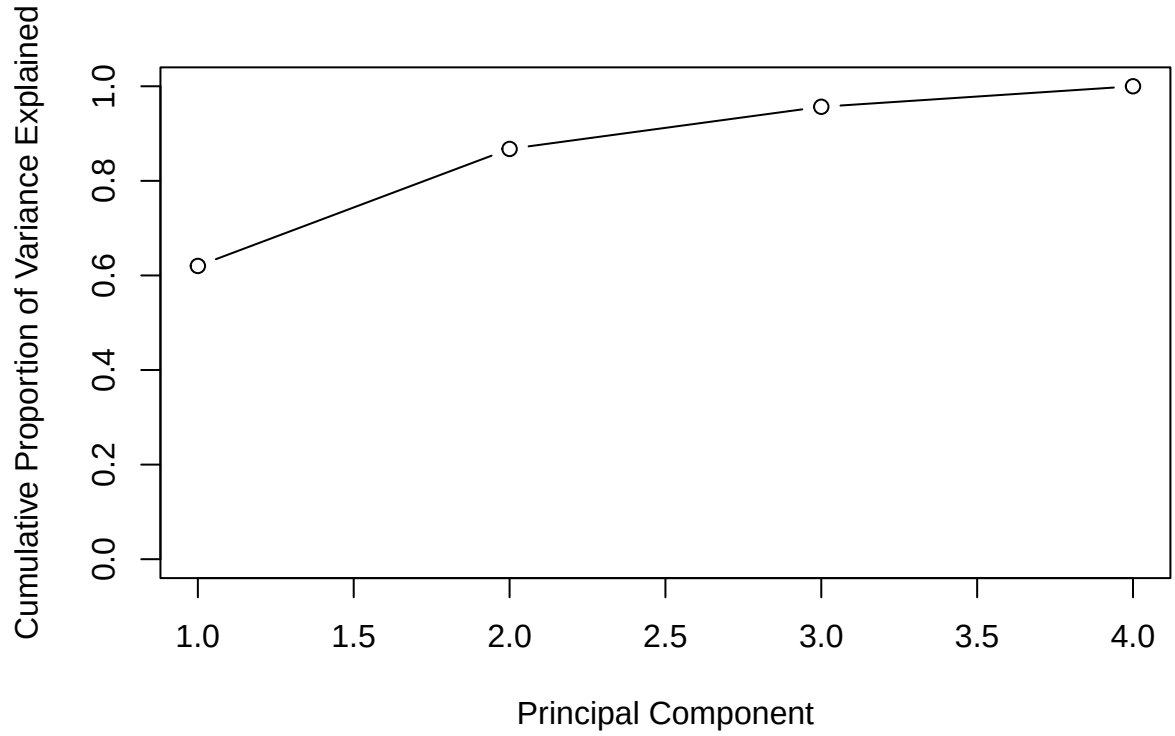
```
## [1] 0.62006039 0.24744129 0.08914080 0.04335752
```

We see that the first principal component explains 62.0% of the variance in the data, the next principal component explains 24.7% of the variance, and so forth. We can plot the PVE explained by each component, as well as the cumulative PVE, as follows:

```
plot(pve, xlab = " Principal Component", ylab = "Proportion of
Variance Explained", ylim = c(0,1), type = "b")
```




```
plot(cumsum(pve), xlab = "Principal Component", ylab ="
Cumulative Proportion of Variance Explained", ylim = c(0,1),
type = "b")
```



2 PCA for mixed data

As you have seen PCA handles numerical variables whereas multiple correspondence analysis (which we have not covered) handles categorical variables. Often data sets contain a mixture of both categorical and mixture

data.

PCA methods dealing with a mixture of numerical and categorical variables already exist and have been implemented in the `R` packages `ade4` and `FactoMineR`. The `R` package we will be looking at is `PCAmixdata`, where the function `PCAmix()` implements an algorithm presented as a single PCA with metrics, i.e., based on a generalized singular value decomposition of pre-processed data. This algorithm includes naturally standard PCA and standard MCA as special cases.

Let's assume that the dataset to be analyzed comprises n observations described by p_1 numerical variables and p_2 categorical variables. The dataset is represented by the $n \times p_1$ quantitative matrix $\mathbf{X}_1$ and the $n \times p_2$ qualitative matrix $\mathbf{X}_2$. Let m denote the total number of levels of the p_2 categorical variables. The algorithm merges PCA and MCA thanks to the general framework. The two first steps of the algorithm (pre-processing and factor coordinates processing) mimic this general framework with the numerical data matrix $\mathbf{X}_1$ and the qualitative data matrix $\mathbf{X}_2$ as inputs. The third step is dedicated to squared loading processing where squared loadings are defined as squared correlations for numerical variables and correlation ratios for categorical variables.

- Step 1
  - Build the real matrix $\mathbf{Z} = [\mathbf{Z}_1; \mathbf{Z}_2]$ of dimension $n \times (p_1 + m)$ where $\mathbf{Z}_1$ is the standardized version of $\mathbf{X}_1$ and $\mathbf{Z}_2$ the centered indicator matrix of the levels of $\mathbf{X}_2$.
  - Build the diagonal matrix $\mathbf{N}$ of the weights of the rows. The n rows are weighted by $1/n$.
  - Build the diagonal matrix $\mathbf{M}$ of the weights of the columns.

  The p_1 first columns are weighted by 1. The m last columns are weighted by n/n_s, with n_s being the number of observations with levels. The metric $M = \text{diag}(1, \ldots, 1, \frac{n}{n_s}, \ldots, \frac{n}{n_m})$ indicates that the distance between two rows of $\mathbf{Z}$ is a mixture of the simple euclidean distance used in PCA (for the first p_1 columns) and the weighted distance in the spirit of the χ^2 distance used in MCA (for the last m columns). It can be shown that the total variance is $p_1 + m - p_2$.
- Step 2 This step is rather technical and beyond the scope of this course. In this step a generalized singular value decomposition is performed on $\mathbf{Z}$ and eigenvalues and eigenvectors obtained.
- Step 3 In this step the principal component scores are computed.

Let us now illustrate the procedure `PCAmix` with the data `housing` of the dataset `gironde`. This data set contains $n = 542$ municipalities with 3 numerical variables and 2 categorical with a total of 4 levels.

```
library(PCAmixdata)
data("gironde")
head(gironde$housing)
```

```
##                     density primaryres   houses owners council
## ABZAC                131.70      88.77  inf 90%  64.23  sup 5%
## AILLAS                21.21      87.52  sup 90%  77.12  inf 5%
## AMBARES-ET-LAGRAVE   531.99      94.90  inf 90%  65.74  sup 5%
## AMBES                101.21      93.79  sup 90%  66.54  sup 5%
## ANDERNOS-LES-BAINS   551.87      62.14  inf 90%  71.54  inf 5%
## ANGLADE               63.82      81.02  sup 90%  80.54  inf 5%
```

Edit `?gironde` for a description of the `housing` data. A principal component analysis is performed using the function `PCAmix` on the `housing` data.

```
split = splitmix(gironde$housing)
X1 = split$X.quanti
X2 = split$X.quali
res.pcamix = PCAmix(X.quanti = X1, X.quali = X2, rename.level = TRUE, graph = FALSE)
res.pcamix$eig
```

```
##       Eigenvalue Proportion Cumulative
```

```
## dim 1  2.5268771  50.537541   50.53754
## dim 2  1.0692777  21.385553   71.92309
## dim 3  0.6303253  12.606505   84.52960
## dim 4  0.4230216   8.460432   92.99003
## dim 5  0.3504984   7.009968  100.00000
```

Note that the function `splitmix` splits a mixed data matrix into two datasets: one with the numerical variables and one with the categorical variables.

The sum of the eigenvalues is equal to $p_1 + m - p_2 = 5$ and the first two dimensions retrieve 72% of the total variation. Let us visualize on these two dimensions the 4 different plots.

```
par(mfrow=c(2,2))
plot(res.pcamix, choice = "ind", coloring.ind = X2$houses, label = FALSE,
     posleg = "bottomright", main ="(a) Observations")
plot(res.pcamix, choice = "levels", xlim = c(-1.5,2.5), main = "(b) Levels")
plot(res.pcamix, choice = "cor", main = "(c) Numerical variables")
plot(res.pcamix, choice = "sqload", coloring.var = T, leg = TRUE,
     posleg = "topright", main = "(d) All variables")
```

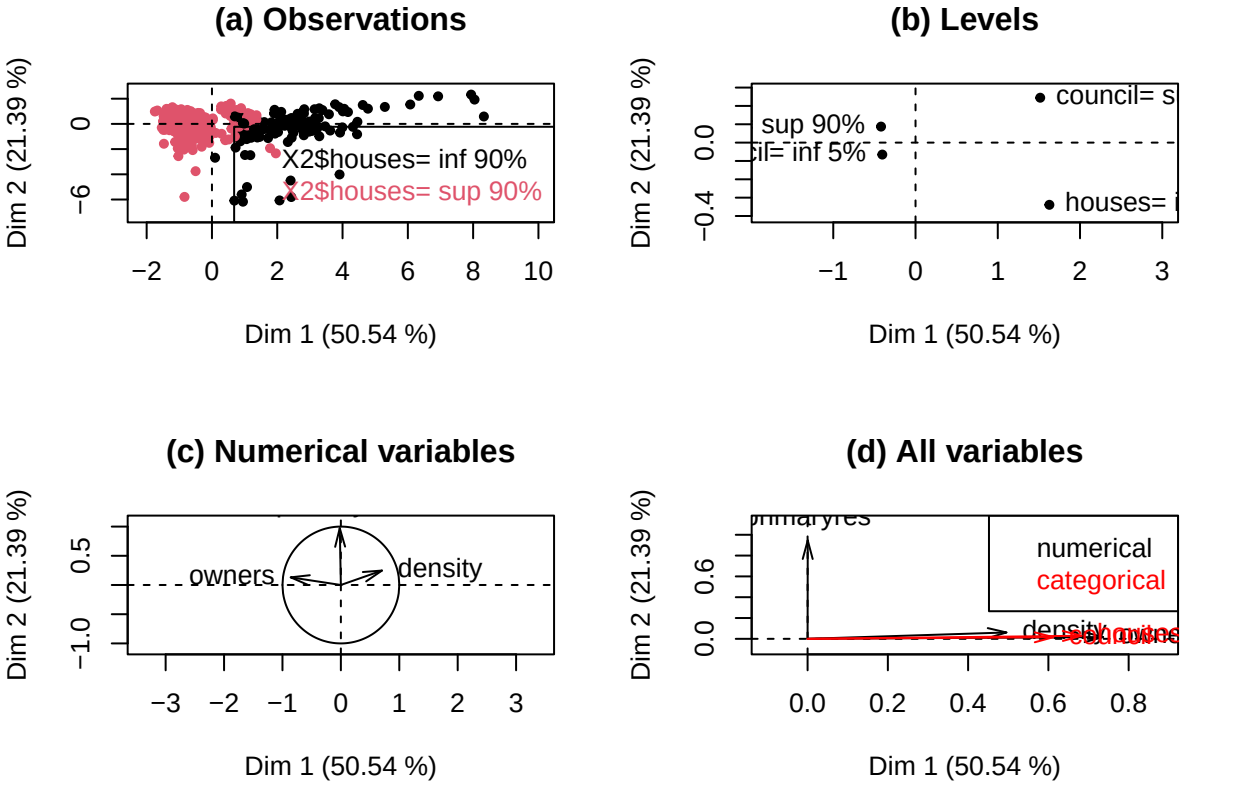


Figure (a) shows the principal component map where the municipalities (the observations) are colored by their percentage of houses (less than 90%, more than 90%). The first dimension (left hand side) highlights municipalities with large proportions of privately-owned properties. The level map in Figure (b) confirms this interpretation and suggests that municipalities with a high proportion of houses (on the left) have a low percentage of council housing. The correlation circle in Figure (c) indicates that population density is negatively correlated with the percentage of home owners and that these two variables discriminate the municipalities on the first dimension. Figure (d) plots the variables (categorical or numerical) using squared loadings as coordinates. For numerical variables, squared loadings are squared correlations and for categorical variables squared loadings are correlation ratios. In both cases, they measure the link between the variables and the principal components. One observes that the two numerical variables `density` and `owners` and the two categorical variables `houses` and `council` are linked to the first component. On the contrary, the

variable `primaryres` is clearly orthogonal to these variables and associated to the second component. %Note that these links show neither a positive nor a negative association, and %the maps Figure (b) and (c) are necessary for more precise interpretation.

In summary, municipalities on the right of the principal component map have a relatively high proportion of council housing and a small percentage of privately-owned houses, with most accommodation being rented. On the other hand, municipalities on the left hand side are mostly composed of home owners living in their primary residence. The percentage of primary residences also has a structuring role in the characterization of municipalities in this region of France by defining clearly the second dimension. Indeed the municipalities at the bottom of the map (those with small values on the second dimension) are sea resorts with many secondary residences. For instance the 10 municipalities with the smallest coordinates in the second dimension are well-known resorts on France's Atlantic coast:

```
sort(res.pcamix$ind$coord[,2])[1:10]
```

```
##  VENDAYS-MONTALIVET             CARCANS             LACANAU      SOULAC-SUR-MER
##           -6.171971           -6.087304           -6.070451           -5.802359
## GRAYAN-ET-L'HOPITAL     LEGE-CAP-FERRET      VERDON-SUR-MER             HOURTIN
##           -5.791642           -5.596315           -5.008545           -4.493259
##            ARCACHON               PORGE
##           -4.013374           -3.751233
```

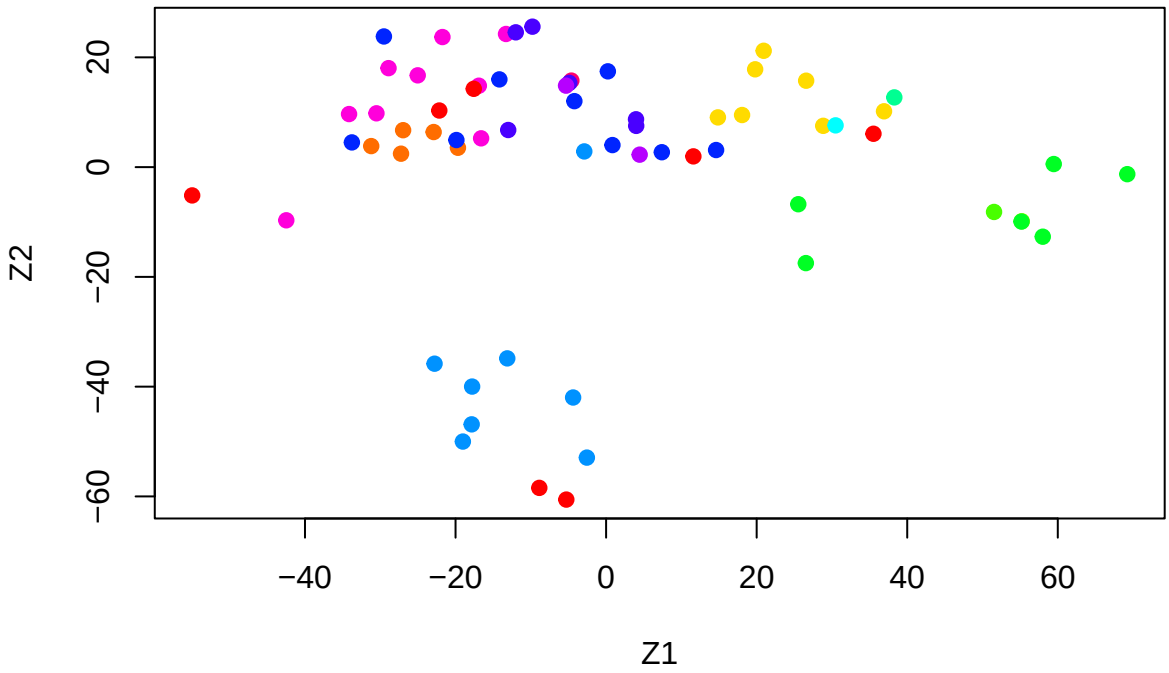
3 Neural networks for dimension reduction

In this section, we will use the gene expression data in `ISLR` package.

```
library(ISLR)
library(keras)
library(tensorflow)
######get gene data
nci.labs=NCI60$labs
nci.data=NCI60$data
```

We first find the principal components of the data based on PCA and get the PC plots. We label each instance with their known class labels, i.e. the samples from the same class have the same color.

```
####################################
######PCA
pr.out=prcomp(nci.data, scale=TRUE)
######PC plots
Cols=function(vec){
  cols=rainbow(length(unique(vec)))
  return(cols[as.numeric(as.factor(vec))])
}
plot(pr.out$x[,1:2], col=Cols(nci.labs), pch=19,
       xlab="Z1",ylab="Z2")
```



We then use a neural network with one hidden layer and linear activations functions for dimension reduction. The point is to set the input and output layers with the same size.

```
#############################################################
######Neural network with one hidden layer
model = keras_model_sequential()
```

```
## Loaded Tensorflow version 2.8.0
```

```
model %>%
  layer_dense(units = 2, activation = 'linear', input_shape = ncol(nci.data),name = "subspace") %>%
  layer_dense(units = ncol(nci.data), activation = 'linear')
summary(model)
```

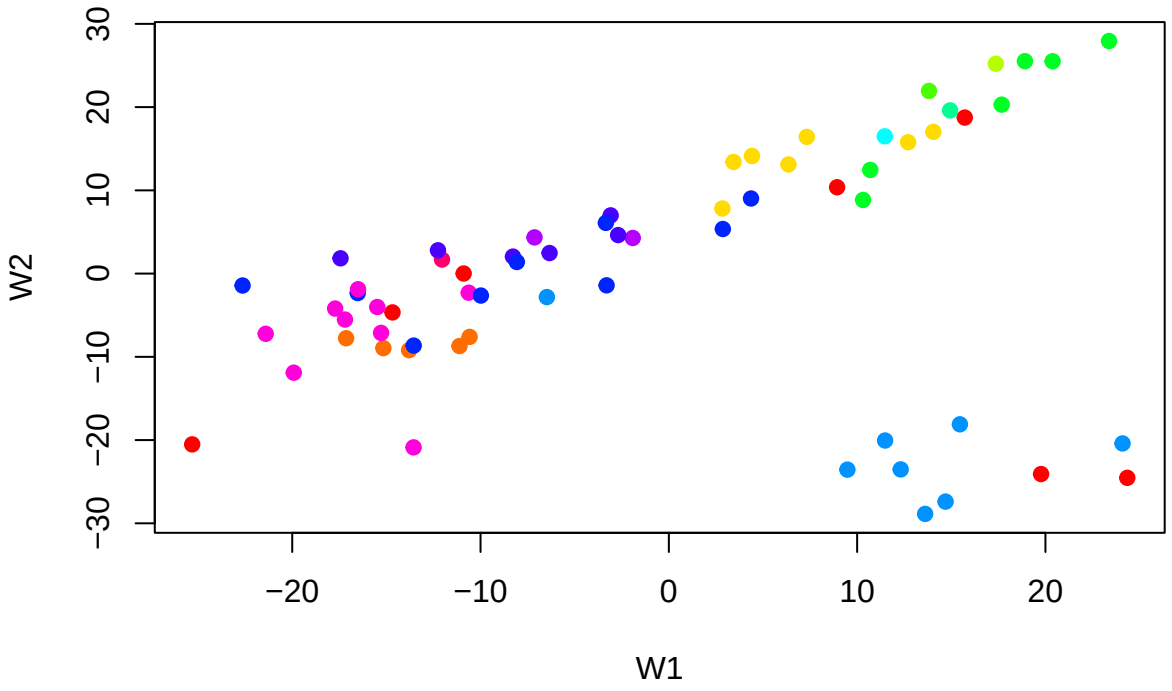
```
## Model: "sequential"
## _________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## =========================================================================
##  subspace (Dense)                   (None, 2)                       13662
##
##  dense (Dense)                      (None, 6830)                    20490
##
## =========================================================================
## Total params: 34,152
## Trainable params: 34,152
## Non-trainable params: 0
## _________________________________________________________________________
```

```
model %>% compile(
  loss = 'mean_squared_error',
  optimizer = "adam"
)
history = model %>% fit(
  as.matrix(nci.data), as.matrix(nci.data),
```

```
  epochs = 80
)
```

We can then get the middle hidden layer out and project the training instances to the subspace spanned by the two hidden layers.

```
############################################################
######Neural network with one hidden layer
subspace = keras_model(inputs = model$input, outputs = get_layer(model, "subspace")$output)
projection1 = predict(subspace, nci.data)
plot(projection1, col=Cols(nci.labs), pch=19,
     xlab="W1",ylab="W2")
```



We obtain the two graphs for PCA and neural network. It seems that the two subspaces are quite similar. Here we only set `epochs = 80`, but you can set it to a larger number to further reduce the reconstruction error and see the outputs.