dimension_reduction_PHS597_hw2

Jingyu Xu

Dimension reduction, including VQ, PCA and NMF. All three algorithms could lead to dimension reduction. In this HW, we will explore how these algorithms can preserve information for eQTL analysis. eQTL analysis, in simple terms, is to regress gene expression level over genotypes. For HW, please apply VQ, PCA and NMF to reduce the dimension of gene expression levels to a given dimension (which will vary)

Glance at the data

```r
## reading gene expression data
expression_raw <- read.delim("GEUVADIS_normalized_expression_chr20")
## reading genotype data
genotype <- read.delim("GEUVADIS_chr20_processed.traw")
## glance at the data
## transpose the matrix
expression = t(expression_raw[,5:ncol(expression_raw)])
dim(expression)
```

```
## [1] 358 545
```

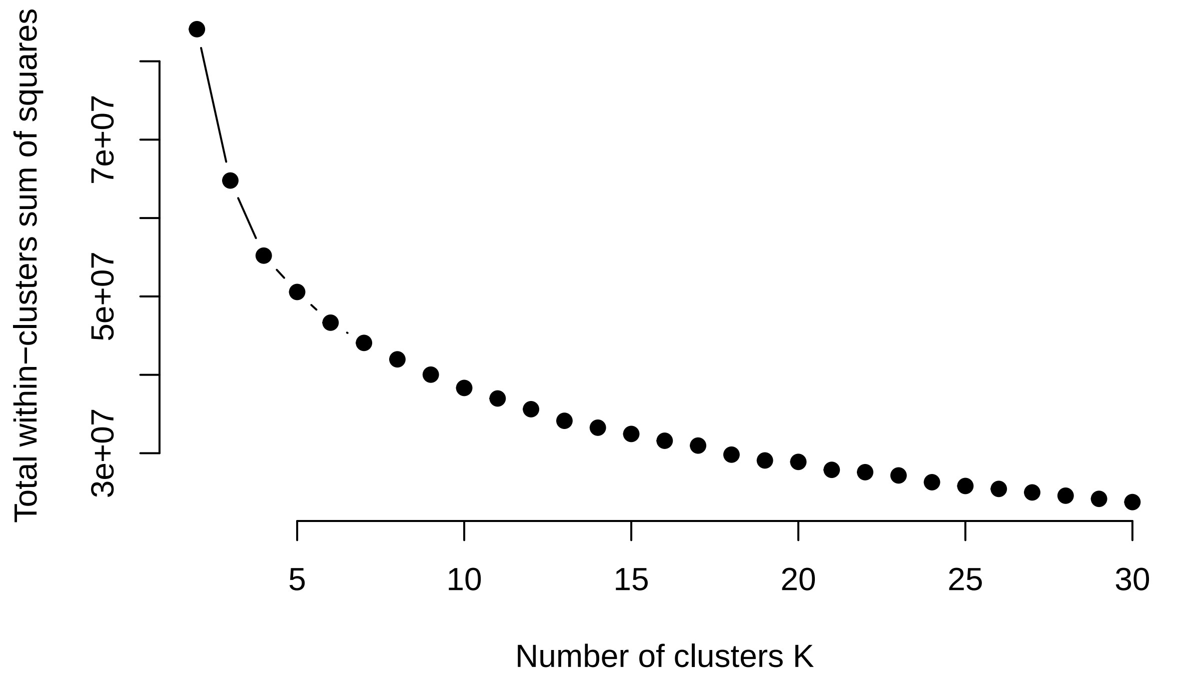
Three dimension reduction methods

Vector Quantization

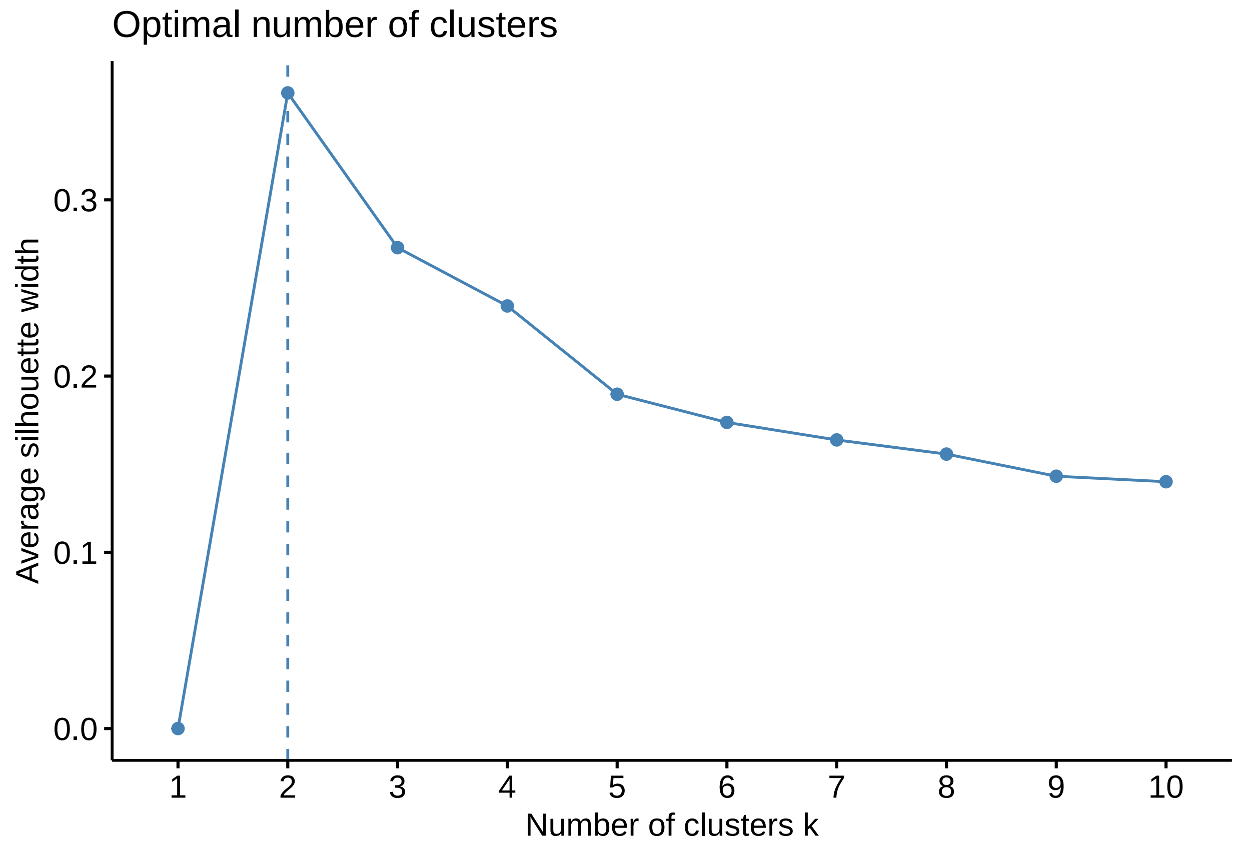
Vector quantization is an implementation of K-mean clustering method. The general idea is to implement K-means clustering to compress the data into K clusters and apply them to the centroid of the cluster. In this scenario, our main goal is to do the dimension reduction rather than do a perfect classification. We have to consider two aspects:1) information retaining 2) classification performance(within cluster variation). We might not need to do choose optimal K as we normally did in clustering analysis. It will depend on to which degree we hope to compress the data as well. But I still attach the procedure of choosing best clusters here for a mini practice. It seems like that 2 or 3 clusters already acheive good clustering performace, however, here I choose an arbitary criteria that we can classify them into 20 clusters to retain more information.

```r
set.seed(58)

# function to compute total within-cluster sum of square
wss <- function(k) {
  kmeans(expression, k, nstart = 10 )$tot.withinss
}
k.values <- c(2:30)
# extract wss for 2-30 clusters
wss_values <- map_dbl(k.values, wss)
plot(k.values, wss_values,
       type="b", pch = 19, frame = FALSE,
       xlab="Number of clusters K",
       ylab="Total within-clusters sum of squares")
```



```
## directly use silhoette method
fviz_nbclust(expression, kmeans, method = "silhouette")
```



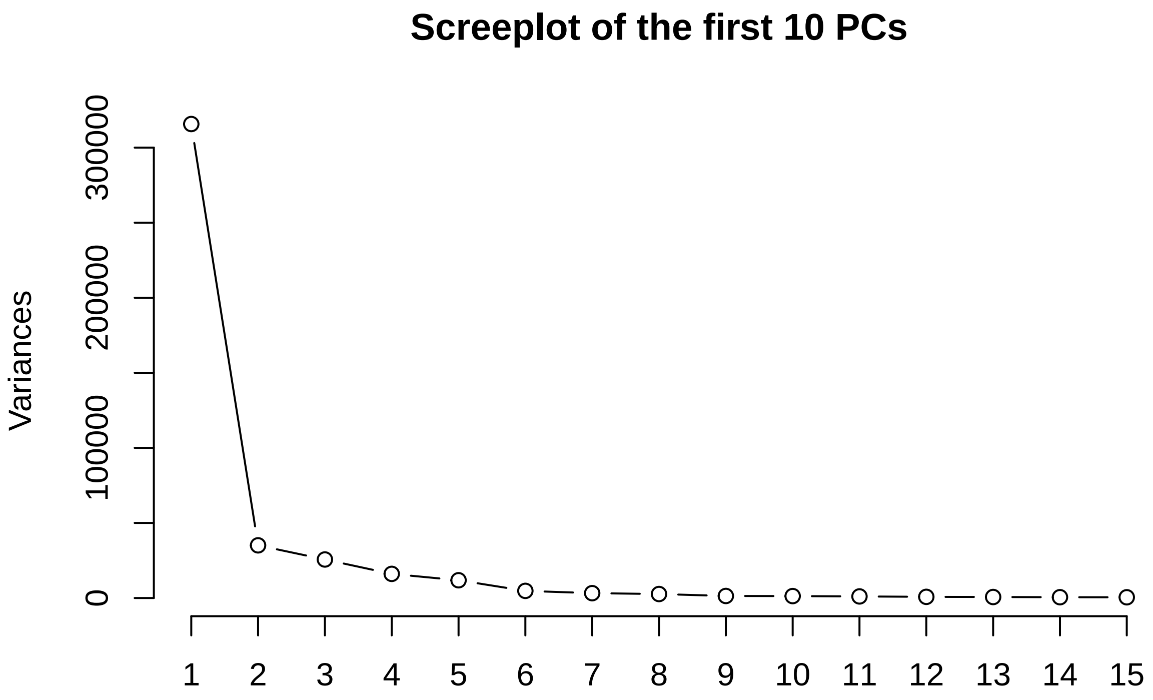
```
##
vq = kmeans(expression, centers=20)
## assign centroid to each gene_id for vector quantification
```

```r
index = vq$cluster
center = vq$centers
compression_vq = as.matrix(center)[index,]
saveRDS(compression_vq,file="vq.rds")
```

Principle Component Analysis(PCA)

There is a small trick which might be of concern in Principle component analysis. Usually we centered the data before doing principle component analysis. Therefore, after reconstucting the matrix, we have

```r
mu = colMeans(expression)
pca = prcomp(expression)# default center=TRUE scale=FALSE
screeplot(pca, type = "l", npcs = 15, main = "Screeplot of the first 10 PCs")
```



```r
sum(summary(pca)$importance[2,1:10])
```

```
## [1] 0.97313
```

```r
ncomp = 10
compression_pca = pca$x[,1:ncomp] %*% t(pca$rotation[,1:ncomp])
compression_pca = scale(compression_pca, center = -mu, scale = FALSE)
saveRDS(compression_pca,file="pca.rds")
```

Nonnegative Matrix Factorization

In that nonnegative matrix factorization require that the all the elements are non-negative. Therefore, we could manipulate the dataset by substracing the minimum value from the matrix

```r
substr_min = function(x){
  return(x-min(x))
}
expression_nmf = apply(expression,substr_min,MARGIN = 2)
min_value = apply(expression,min,MARGIN = 2)
nmf1 = NMF::nmf(expression_nmf,20)
```

```
##retreive the estiamted target matrix
compression_nmf = fitted(nmf1)
## to convert back to orginal one, we may have to add min value again
## store two version(removed the later one)
#compression_nmf_add = t(t(compression_nmf)+min_value)
saveRDS(compression_nmf,"nmf.rds")
```

Comparison between compressed and uncompressed data

Regression gene expression on genotypes to find eQTL

To find eQTL, I regress gene expression on genotype via univariate regression and adjust multiple comparison via FDR control.

```
#### link

### show the code running on cluster in parallel

no_cores <- detectCores() - 1
cl <- makeCluster(4)
doParallel::registerDoParallel(cl)
element = c("uncompressed","vq","pca","nmf")
expression = element[i] ## read corresponding dataset

cl <- makeCluster(no_cores)
doParallel::registerDoParallel(cl)
regression = function(x,y){
  A = summary(lm(y~x))
  return(A$coefficients[,4][2])
}

foreach (i = 1:545) %do%{
  start = expression_raw[i,3]-500000
  end = expression_raw[i,4]+500000
  snp_selected = genotype%>%filter(POS>=start&POS<=end)
  snp_index = snp_selected$SNP
  snp_selected = t(snp_selected[,-c(1:6)])
  colnames(snp_selected) = snp_index
  p_value = apply(snp_selected,regression,y=expression[,i],MARGIN = 2)
  p_value_adj = p.adjust(p_value, method="fdr")
  return(p_value_adj)
}
```

Compare the eQTL identified in terms of eQTL identified

In that it takes long time to knit directly, I recall the saved results

```
### read saved results
uncompressed = readRDS("~/Documents/PHD-2022-Spring/PHS597/Unsupervised-learning-for-dimension-reductio
vq_p <- readRDS("~/Documents/PHD-2022-Spring/PHS597/Unsupervised-learning-for-dimension-reduction-with-
pca_p = readRDS("~/Documents/PHD-2022-Spring/PHS597/Unsupervised-learning-for-dimension-reduction-with-
nmf_p = readRDS("~/Documents/PHD-2022-Spring/PHS597/Unsupervised-learning-for-dimension-reduction-with-



##checked the identified eqtl in uncompressed and compressed data
```

```r
original = nrow(uncompressed[which(uncompressed$pval<0.05),])
vq_eqtl = nrow(vq_p[which(vq_p$pval<0.05),])
pca_eqtl = nrow(pca_p[which(pca_p$pval<0.05),])
nmf_eqtl = nrow(nmf_p[which(nmf_p$pval<0.05),])

## check how many eqtl identified in original data is identified in the dimension reduction method
vq_lap = sum((vq_p[which(uncompressed$pval<0.05),][,3]<0.05))/nrow(uncompressed)
pca_lap = sum((pca_p[which(uncompressed$pval<0.05),][,3]<0.05))/nrow(uncompressed)
nmf_lap = sum((nmf_p[which(uncompressed$pval<0.05),][,3]<0.05))/nrow(uncompressed)
data.frame(method=c("Vector Quantization","PCA","Nonnegative Matrix Factorization"),recovered_eqtl = c(
```

method	recovered_eqtl
Vector Quantization	0.0001225
PCA	0.0001061
Nonnegative Matrix Factorization	0.0000016

It seems that the recovered eqtl from compressed data is of small proportion. Among three methods, nmf has the worst result. I think the initial matrix manipulation might have influence on the biased result.