Dimension_reduction

Xinying Fang

2/27/2022

```
library(factoextra)
```

```
## Loading required package: ggplot2
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
library(purrr)
library(NMF)
```

```
## Loading required package: pkgmaker
```

```
## Loading required package: registry
```

```
## Loading required package: rngtools
```

```
## Loading required package: cluster
```

```
## NMF - BioConductor layer [OK] | Shared memory capabilities [NO: bigmemory] | Cores 11/12
```

```
##   To enable shared memory capabilities, try: install.extras('
## NMF
## ')
```

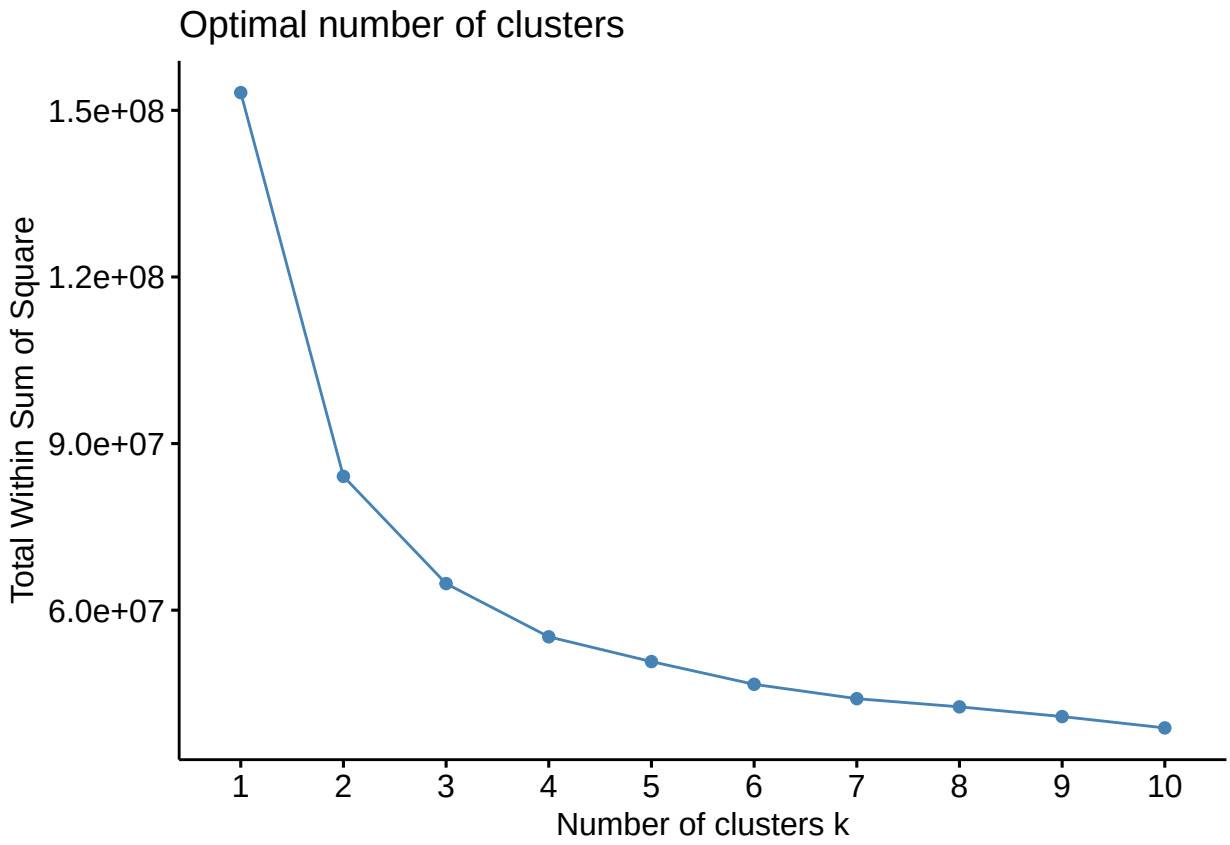
load data

```
gene <- read.delim("/Users/rrrrrita/Documents/GitHub/PHS597DeepLearning/gene_expression_sample/GEUVADIS_
geneexp <- gene[,c(1,5:ncol(gene))]
genotype <- read.delim("/Users/rrrrrita/Documents/GitHub/PHS597DeepLearning/gene_expression_sample/GEUV
```

Vector quantization (VQ)

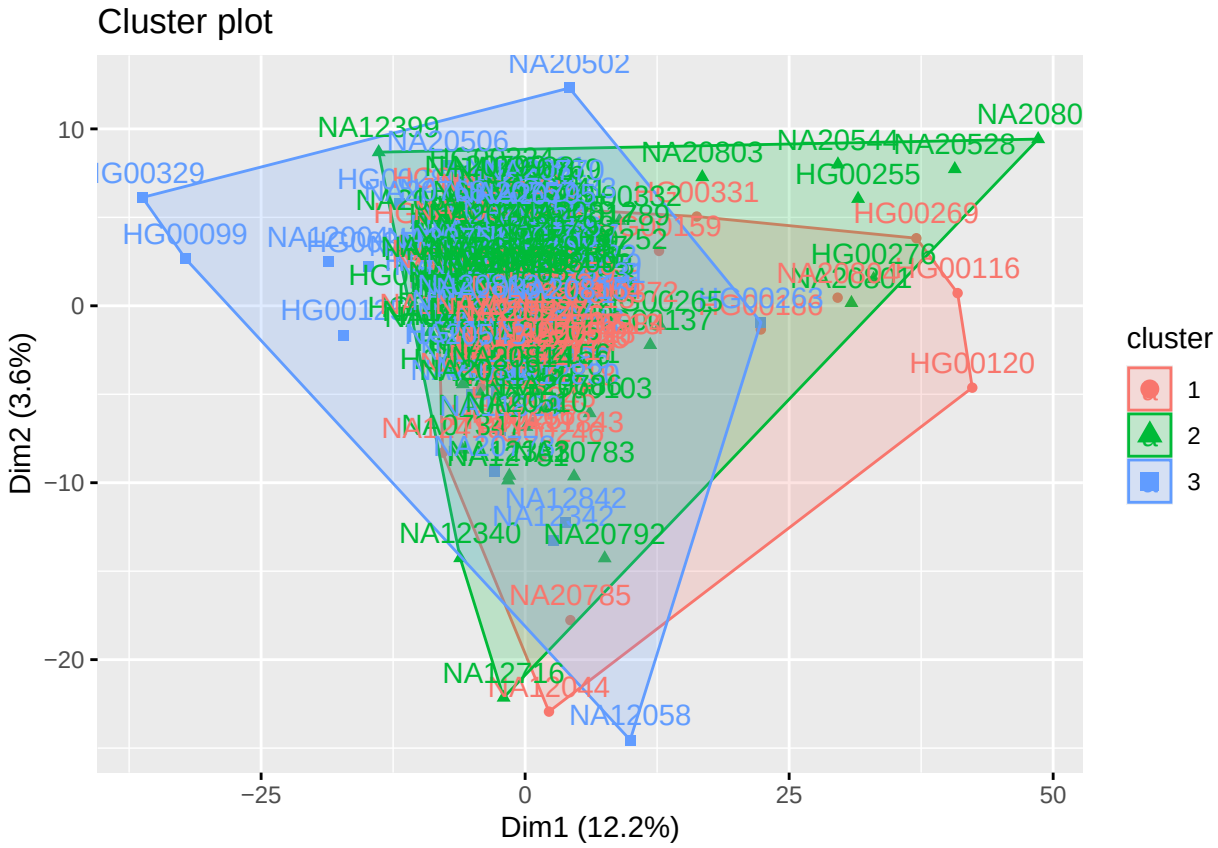
VQ is one type of k means. First I determined the optimal k based on total within-cluster sum of square:

```
wss <- function(k) {
  kmeans(t(geneexp[,-1]), k, nstart = 10 )$tot.withinss
}

fviz_nbclust(t(geneexp[-1]), kmeans, method = "wss")
```



The elbow point happens at k=2 or 3. Here I chose k=3:

```
kres <- kmeans(t(geneexp[-1]), 3, nstart = 1)
fviz_cluster(kres, data = t(geneexp[-1]))
```



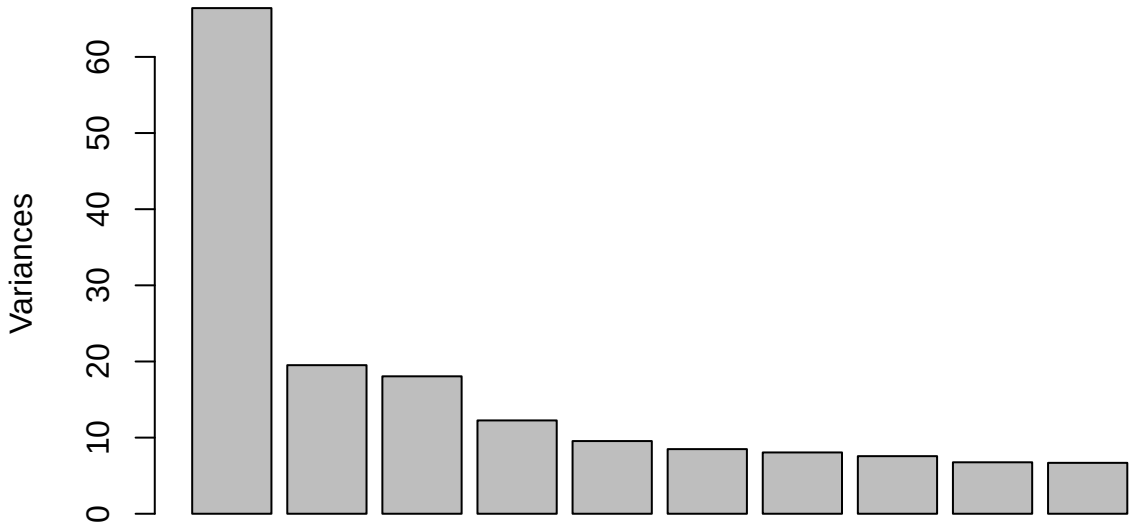
The cluster plot tells that it is hard to discern these samples based on gene expression.

I used the centroid values of each cluster to represent samples within each cluster:

```
y_vq <- kres$centers[kres$cluster,]
rownames(y_vq) <- colnames(geneexp[,-1])
```

PCA

```
pca <- prcomp(t(geneexp[-1]), center=TRUE, scale = TRUE)
plot(pca, npcs=10)
```



As we can tell, first 10 PCs have represented most of the total variances, so I used the first 10 pcs to recover the uncompressed gene expression:

(It is also reasonable to choose first 3 PCs)

```
pca <- prcomp(t(geneexp[-1]), center=TRUE, scale = FALSE)
pc <- pca$x
rotation <- pca$rotation

npc <- 10
colm <- colMeans(t(geneexp[,-1]))

y_pca <- pc[,1:npc] %*% t(rotation[,1:npc]) + colm
```

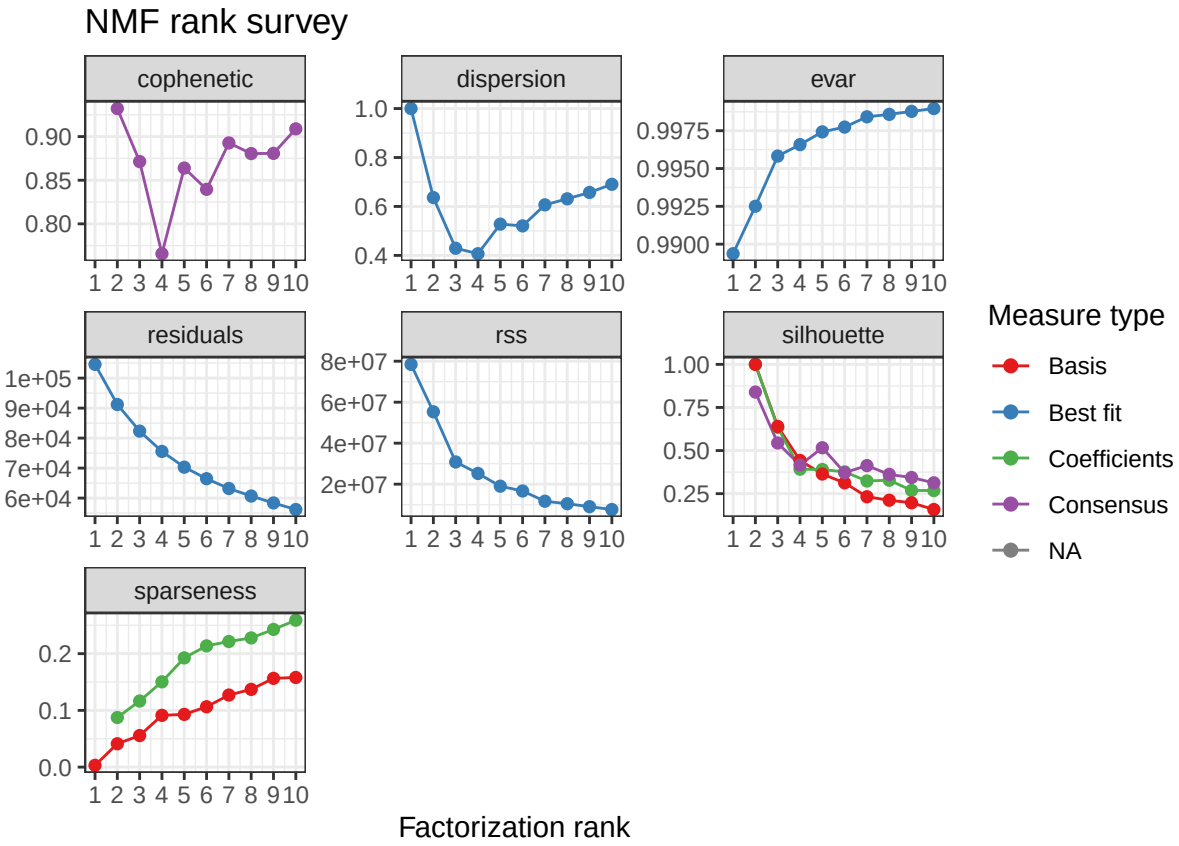
NMF

First, run several experiments to select the optimal rank:

```
geneexp_nmf <- t(geneexp[-1])

# transfer into nonnegative data
geneexp_nmf <- nneg(geneexp_nmf)
estim.r <- nmf(geneexp_nmf, 1:10, nrun=20, seed=123456)
plot(estim.r)
estim.r
```

```
## Warning: Removed 3 row(s) containing missing values (geom_path).
```

```
## Warning: Removed 5 rows containing missing values (geom_point).
```



For optimal rank, (Frigyesi 2008) considered the smallest value at which the decrease in the RSS is lower than the decrease of the *RSS* obtained from random data. From the above RSS plot, the decrease from 3 to 4 is lower than the decrease from 1 to 2 and 2 to 3. Therefore, I chose rank = 4.

```
geneexp_nmf <- t(geneexp[-1])
# transfer into nonnegative data
geneexp_nmf <- nneg(geneexp_nmf)

nmffit <- nmf(geneexp_nmf, rank=4, "brunet", seed=123456) # higher rank -> better results
w <- nmffit@fit@W
h <- nmffit@fit@H
y_nmf <- w %*% h
```

Results comparison

eQTL analysis

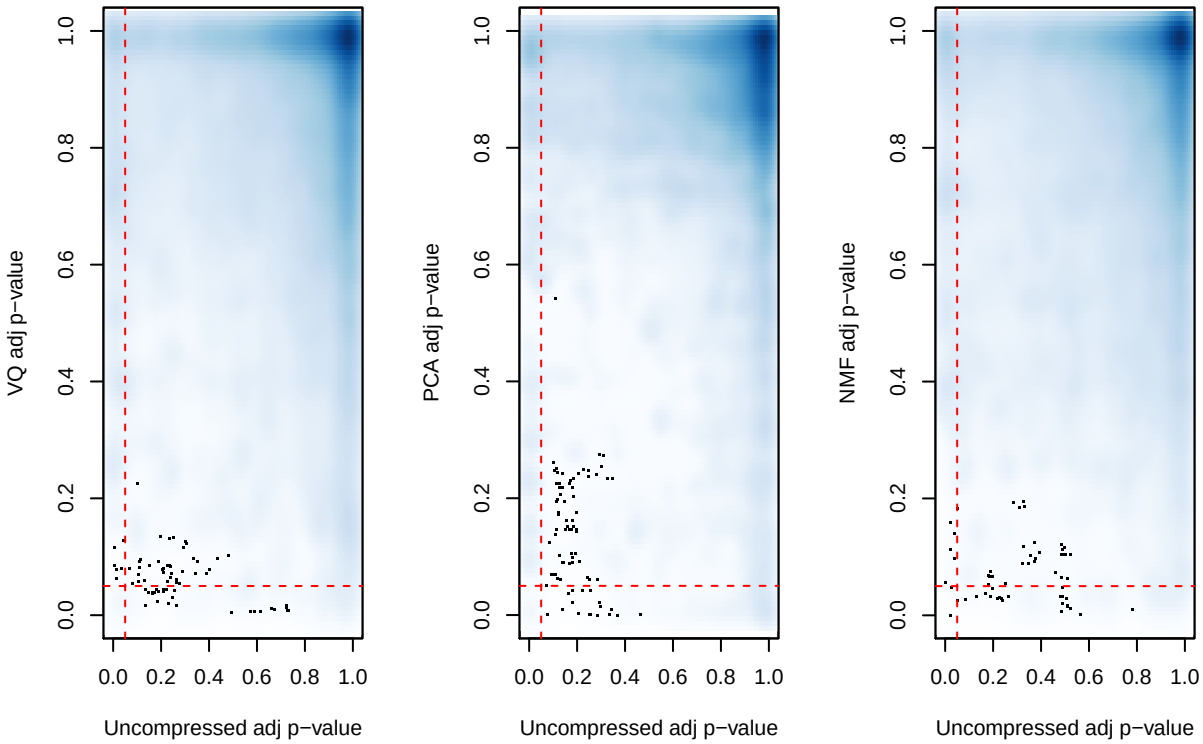
To conduct the eQTL analysis, I filtered the data within 500000 downstream and upstream of gene start and end sites. I regressed each gene expression (SNP) on each genotype. The Benjamini-Hochberg method was applied for multiple testing adjustment.

```
resdf <- c()
eqtl <- function(gexp){

  for ( i in 1:nrow(gene)){
    gtype <- subset(genotype, POS >= (gene$start[i] - 500000) & POS <= (gene$end[i] + 500000))
```

```
    gtype <- gtype[,-(1:6)]
    y <- gexp[,i]

    resdf1 <- c()
    for ( j in 1:nrow(gtype)){
      x <- t(gtype[j,])
      lmfit <- lm(y ~ x)

      out <- data.frame(gene=gene$gene_id[i],
                        genotype = genotype$SNP[j],
                        pval = summary(lmfit)$coefficients[2,4])
      resdf1 <- rbind(resdf1, out)
    }
    resdf1$pval <- p.adjust(resdf1$pval, method = "BH")
    resdf <- rbind(resdf, resdf1)
  }

  return(resdf)
}
```

```
# Read saved results
vq <- readRDS("vq_dopar.rds")
pca <- readRDS("pca_dopar.rds")
nmf <- readRDS("nmf_dopar.rds")
uncomp <- readRDS("uncomp_dopar.rds")
```

```
par(mfrow=c(1,3))
smoothScatter(x=uncomp$pval, y=vq$pval,
              xlab = "Uncompressed adj p-value", ylab = "VQ adj p-value")
abline(v = 0.05, h=0.05, lty = 2, col = "red")
smoothScatter(x=uncomp$pval, y=pca$pval,
              xlab = "Uncompressed adj p-value", ylab = "PCA adj p-value")
abline(v = 0.05, h=0.05, lty = 2, col = "red")
smoothScatter(x=uncomp$pval, y=nmf$pval,
              xlab = "Uncompressed adj p-value", ylab = "NMF adj p-value")
abline(v = 0.05, h=0.05, lty = 2, col = "red")
```



It's not easy to see a linear relationship between adjusted p-values for uncomparessed dataset and compressed dataset with three dimension reduction methods. Then we further evaluate the relationship with confusion matrices:

Confusion matrix

```
binary.pval <- function(x){
  y <- ifelse(x<0.05, 1, 0)
  y <- factor(y, levels = c(0,1))
  return(y)
}
vq_cm <- caret::confusionMatrix(data = binary.pval(vq$pval),
                        reference = binary.pval(uncomp$pval))
```

```
## Registered S3 methods overwritten by 'proxy':
##   method               from
##   print.registry_field registry
##   print.registry_entry registry
```

```
pca_cm <- caret::confusionMatrix(data = binary.pval(pca$pval),
                        reference = binary.pval(uncomp$pval))

nmf_cm <- caret::confusionMatrix(data = binary.pval(nmf$pval),
                        reference = binary.pval(uncomp$pval))
```

```
byClass <- rbind(VQ =round(vq_cm$byClass, 3),
                 PCA=round(pca_cm$byClass, 3),
```

```
                NMF = round(nmf_cm$byClass, 3))

overall <- rbind(VQ =round(vq_cm$overall, 3),
                PCA=round(pca_cm$overall, 3),
                NMF = round(nmf_cm$overall, 3))
byClass
```

```
##     Sensitivity Specificity Pos Pred Value Neg Pred Value Precision Recall
## VQ        0.998       0.006          0.979          0.074     0.979  0.998
## PCA       0.998       0.005          0.979          0.044     0.979  0.998
## NMF       0.999       0.000          0.979          0.001     0.979  0.999
##        F1 Prevalence Detection Rate Detection Prevalence Balanced Accuracy
## VQ  0.989      0.979          0.978                0.998             0.502
## PCA 0.988      0.979          0.977                0.998             0.501
## NMF 0.989      0.979          0.978                0.999             0.499
```

```
overall
```

```
##     Accuracy  Kappa AccuracyLower AccuracyUpper AccuracyNull AccuracyPValue
## VQ     0.978  0.008         0.978         0.978        0.979              1
## PCA    0.977  0.005         0.977         0.977        0.979              1
## NMF    0.978 -0.002         0.978         0.978        0.979              1
##     McnemarPValue
## VQ              0
## PCA             0
## NMF             0
```

All three methods have high sensitivity but low specificity. NMF has the lowest negative predictive value among the three. For other measurements, these three methods are pretty much the same.