PHS597 HW2, Spring 2022

Havell Markus

Description

- In this homework I will apply VQ (as k-means), PCA and NMF to reduce the dimension of gene expression levels to a given dimension, and examine how many eQTLs can be recovered from compressed data as compared to the analysis of uncompressed gene expression level.
- The data consists of 545 genes and from 358 samples.

```
library("dplyr")
library("data.table")
library("foreach")
library("reshape2")
library("ggplot2")
library("muHVT")
library("NMF")

## load data
genotype <- fread("/gpfs/group/dxl146/default/private/hmarkus/stat_555/gene_expression_sample/GEUVADIS_c
genotype[1:10,1:10]
```

```
##     CHR                       SNP (C)M   POS COUNTED         ALT
##  1:  20          20_61098_C_T_b37    0 61098       C           T
##  2:  20         20_61138_C_CT_b37    0 61138       C          CT
##  3:  20          20_61795_G_T_b37    0 61795       G           T
##  4:  20          20_62731_C_A_b37    0 62731       C           A
##  5:  20          20_63244_A_C_b37    0 63244       A           C
##  6:  20          20_63799_C_T_b37    0 63799       C           T
##  7:  20          20_65288_G_T_b37    0 65288       G           T
##  8:  20          20_65900_G_A_b37    0 65900       G           A
##  9:  20          20_66370_G_A_b37    0 66370       G           A
## 10:  20 20_67500_T_TTGGTATCTAG_b37    0 67500       T TTGGTATCTAG
##     HG00096_HG00096 HG00097_HG00097 HG00099_HG00099 HG00100_HG00100
##  1:               2               2               2               2
##  2:               2               1               2               2
##  3:               1               2               2               2
##  4:               1               2               2               2
##  5:               2               2               2               2
##  6:               1               2               2               2
##  7:               1               2               2               2
##  8:               0               0               0               0
##  9:               0               0               0               0
## 10:               0               0               0               1
```

```
expression <- fread("/gpfs/group/dxl46/default/private/hmarkus/stat_555/gene_expression_sample/GEUVADIS_
expression[1:10,1:10]
```

```
##             gene_id chromosome    start      end     HG00096     HG00097
##  1: ENSG00000000419         20 49551404 49575092 28.29986678 23.46322572
##  2: ENSG00000020256         20 50668202 50820847 10.40688641 10.55414231
##  3: ENSG00000022277         20 55043647 55093943 48.91700036 48.59283957
##  4: ENSG00000025293         20 34359896 34538303 13.68761088 17.04665137
##  5: ENSG00000025772         20 43570771 43589127 17.16384282 15.37322398
##  6: ENSG00000026036         20 62290653 62330037  0.94725636  1.78671629
##  7: ENSG00000026559         20 49620193 49639666  0.02809289  0.00278352
##  8: ENSG00000042062         20 49202645 49308065  0.03717368  0.11045699
##  9: ENSG00000053438         20 36149617 36152092  0.21680385  0.32822860
## 10: ENSG00000054793         20 50213053 50385173  0.04902307  0.00598812
##         HG00099      HG00100     HG00101      HG00102
##  1: 17.72964416 25.844493272 27.56897062 25.976945042
##  2:  4.43153757  8.562853550 11.27501356  7.649552947
##  3: 31.76922233 55.296140084 53.43624565 43.149508809
##  4:  7.64845057  9.999433386 12.04703137 10.041512573
##  5:  9.12561464 19.737871083 17.85708602 18.208971505
##  6:  1.16090884  1.680259103  1.30697770  1.187033191
##  7:  0.02458499  0.017009740  0.01348924  0.019711201
##  8:  0.02882870  0.027182935  0.15638962  0.145630248
##  9:  0.13815770  0.383013929  0.31430508  0.196532672
## 10:  0.05399513  0.004547755  0.04600689  0.009611866
```

Principal component analysis (PCA)

- I will use the first 50 PCs and its respective loading matrix to reconstruct the original expression matrix

```
exprs.mat.pca <- t(exps.mat) %>% as.data.frame()
exprs.pca <- prcomp(exprs.mat.pca, center = FALSE, scale = FALSE, rank. = ncol(exprs.mat.pca))

exprs.mat.pca <- exprs.pca$x[, 1:50] %*% t(exprs.pca$rotation[,1:50])
exprs.mat.pca <- data.frame(exprs.mat.pca)
```

Non-negative matrix factorization

- Since the expression values are normalized, and some values are negative, I will substract them by the minimum expression value to get positive expression values.
- I will again decompose the original matrix into 50 features and use the respective weight matrix to reconstuct the original expression matrix

```
exprs.mat.nmf <- (exps.mat) %>% as.data.frame()
exprs.mat.nmf <- exprs.mat.nmf - min(exprs.mat.nmf)
exprs.mat.nmf <- as.matrix(exprs.mat.nmf)

exprs.nmf <- nmf(exprs.mat.nmf, 50, 'brunet', seed=123456)
exprs.mat.nmf <- fitted(exprs.nmf)
exprs.mat.nmf <- t(exprs.mat.nmf)
```

Vector quantization

- For vector quantization I will use k-means to cluster the samples into 50 clusters, and assign each sample the compressed-expression value associated to the closest centroid.

```
set.seed(123)

exprs.mat.kmeans <- t(exps.mat) %>% as.data.frame()
km.res <- kmeans(exprs.mat.kmeans, 50, nstart = 25)
exprs.mat.kmeans <- km.res$centers[km.res$cluster,]
```

Code for eQTL analysis

- To identidy eQTL, I will look at SNPs that lie 1mb downstream and upstream of gene start and end site respectively
- I will then regress each SNP against the expression value
- I will also do a multiple testing correction using Benjamini & Hochberg p-value correction to identify significant eQTLs
- I have already ran the code below and saved the results for all methods

```
cl <- parallel::makeForkCluster(8)
doParallel::registerDoParallel(cl)

eqtl.analysis <- foreach(i = 1:nrow(expression), .combine = "rbind") %dopar% {
  # get variants 1mb around gene start site
  iGenotype <- genotype %>% filter(CHR == expression$chromosome[i] & POS >= (expression$start[i] - 1e6)
                                   (POS <= (expression$end[i] + 1e6)))

  # get variants 1mb around gene start site
  # iexpression <- t(expression[i,-c(1:4)])  %>% as.data.frame()
  # colnames(iexpression) <- "exprs.value"
  # iexpression$sample <- row.names(iexpression)
  # row.names(iexpression) <- NULL
  # iexpression$sample <- paste0(iexpression$sample, "_", iexpression$sample)

  iexpression <- exprs.mat.pca[,i]  %>% as.data.frame()
  colnames(iexpression) <- "exprs.value"
  iexpression$sample <- colnames(expression)[-c(1:4)]
  iexpression$sample <- paste0(iexpression$sample, "_", iexpression$sample)

  # regress all variants to the gene expression
  all.i.eqtl <- foreach(j = 1:nrow(iGenotype), .combine = "rbind") %do% {
    jGenotype <- t(iGenotype[j,-c(1:6)])  %>% as.data.frame()
    colnames(jGenotype) <- "geno"
    jGenotype$sample <- row.names(jGenotype)
    row.names(jGenotype) <- NULL

    train.data <- plyr::join(iexpression, jGenotype)

    jLM.model <- lm(exprs.value ~ geno, data = train.data)
    jLM.model <- summary(jLM.model)

    if(nrow(jLM.model$coefficients) > 1){
```

```
      return(data.frame(gene_id = expression$gene_id[i], SNP = iGenotype$SNP[j],
                        pvalue = jLM.model$coefficients[2,4]))
    } else {
      return(data.frame(gene_id = expression$gene_id[i], SNP = iGenotype$SNP[j],
                        pvalue = NA))
    }

  }

  # Benjamini & Hochberg p-value correction
  all.i.eqtl$pvalue.adj <- p.adjust(all.i.eqtl$pvalue, method = "BH")
  return(all.i.eqtl)
}
parallel::stopCluster(cl)

# save(eqtl.analysis, file = "/gpfs/group/dxl46/default/private/hmarkus/stat_555/gene_expression_sample/
# save(eqtl.analysis, file = "/gpfs/group/dxl46/default/private/hmarkus/stat_555/gene_expression_sample/
# save(eqtl.analysis, file = "/gpfs/group/dxl46/default/private/hmarkus/stat_555/gene_expression_sample/
# save(eqtl.analysis, file = "/gpfs/group/dxl46/default/private/hmarkus/stat_555/gene_expression_sample/
```

- load saved results

```
load("/gpfs/group/dxl46/default/private/hmarkus/stat_555/gene_expression_sample/eqtl.analysis.RData")
eqtl.analysis.uncompressed <- eqtl.analysis

load("/gpfs/group/dxl46/default/private/hmarkus/stat_555/gene_expression_sample/eqtl.analysis.pca50.RDa
eqtl.analysis.pca <- eqtl.analysis

load("/gpfs/group/dxl46/default/private/hmarkus/stat_555/gene_expression_sample/eqtl.analysis.nmf50.RDa
eqtl.analysis.nmf <- eqtl.analysis

load("/gpfs/group/dxl46/default/private/hmarkus/stat_555/gene_expression_sample/eqtl.analysis.kmeans.RDa
eqtl.analysis.kmeans <- eqtl.analysis
```

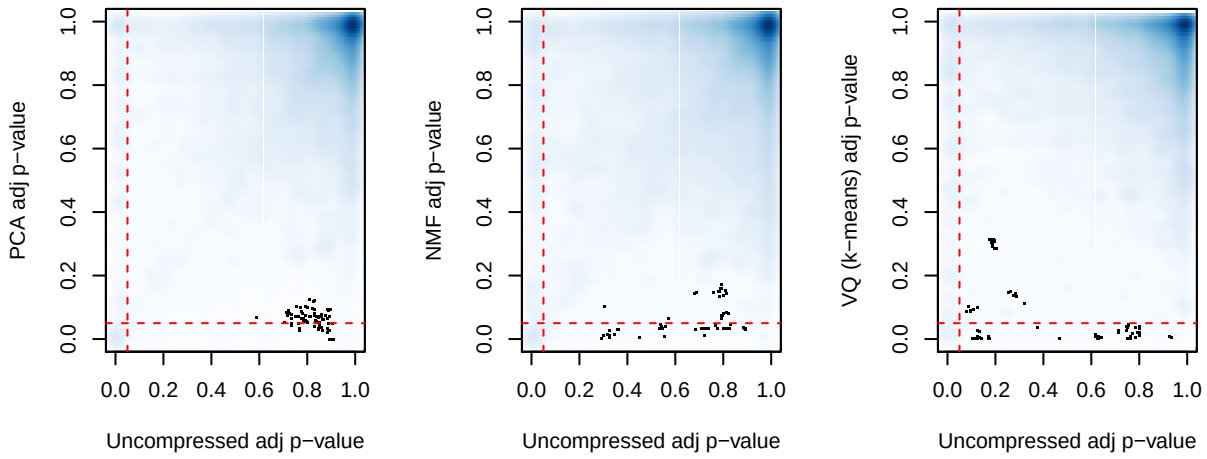
Compared adjusted p-value between uncompressed and compressed methods

- Although there is a linear relationship, there are still p-values that are off the 1-1 diagnogal line

```
par(mfrow=c(1,3))
smoothScatter(x=eqtl.analysis.uncompressed$pvalue.adj, y=eqtl.analysis.pca$pvalue.adj,
              xlab = "Uncompressed adj p-value", ylab = "PCA adj p-value")
abline(v = 0.05, h=0.05, lty = 2, col = "red")

smoothScatter(x=eqtl.analysis.uncompressed$pvalue.adj, y=eqtl.analysis.nmf$pvalue.adj,
              xlab = "Uncompressed adj p-value", ylab = "NMF adj p-value")
abline(v = 0.05, h=0.05, lty = 2, col = "red")

smoothScatter(x=eqtl.analysis.uncompressed$pvalue.adj, y=eqtl.analysis.kmeans$pvalue.adj,
              xlab = "Uncompressed adj p-value", ylab = "VQ (k-means) adj p-value")
abline(v = 0.05, h=0.05, lty = 2, col = "red")
```




```
par(mfrow=c(1,1))
```

Overall performance measure

- All methods have high accuracy, but poor specificity
- NMF and PCA had the highest specificity, when compared to VQ (k-means).

```
pvalue.to.binary <- function(x){
  y <- ifelse(x < 0.05, 1, 0)
  y <- factor(y, levels = c(0,1))
}

#Creating confusion matrix: PCA
pca.confusion.mat <- caret::confusionMatrix(data = pvalue.to.binary(eqtl.analysis.pca$pvalue.adj),
                                            reference = pvalue.to.binary(eqtl.analysis.uncompressed$pvalue.adj))
pca.confusion.mat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       0       1
##          0 2331218   20964
##          1    2866    4359
##
##                Accuracy : 0.9899
##                  95% CI : (0.9898, 0.99)
##     No Information Rate : 0.9893
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2643
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
##             Sensitivity : 0.9988
##             Specificity : 0.1721
##          Pos Pred Value : 0.9911
##          Neg Pred Value : 0.6033
##              Prevalence : 0.9893
##          Detection Rate : 0.9881
##    Detection Prevalence : 0.9969
##       Balanced Accuracy : 0.5855
##
##        'Positive' Class : 0
##
```

```
#Creating confusion matrix: NMF
nmf.confusion.mat <- caret::confusionMatrix(data = pvalue.to.binary(eqtl.analysis.nmf$pvalue.adj),
                                            reference = pvalue.to.binary(eqtl.analysis.uncompressed$pval
nmf.confusion.mat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       0       1
##          0 2332000   22019
##          1    2084    3304
##
##                Accuracy : 0.9898
##                  95% CI : (0.9897, 0.9899)
##     No Information Rate : 0.9893
##     P-Value [Acc > NIR] : 4.04e-15
##
##                   Kappa : 0.2122
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9991
##             Specificity : 0.1305
##          Pos Pred Value : 0.9906
##          Neg Pred Value : 0.6132
##              Prevalence : 0.9893
##          Detection Rate : 0.9884
##    Detection Prevalence : 0.9977
##       Balanced Accuracy : 0.5648
##
##        'Positive' Class : 0
##
```

```
#Creating confusion matrix: VQ
vq.confusion.mat <- caret::confusionMatrix(data = pvalue.to.binary(eqtl.analysis.kmeans$pvalue.adj),
                                            reference = pvalue.to.binary(eqtl.analysis.uncompressed$pval
vq.confusion.mat
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction       0       1
##          0 2332543   23895
##          1    1541    1428
##
##                Accuracy : 0.9892
##                  95% CI : (0.9891, 0.9894)
##     No Information Rate : 0.9893
##     P-Value [Acc > NIR] : 0.7635
##
##                   Kappa : 0.0989
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99934
##             Specificity : 0.05639
##          Pos Pred Value : 0.98986
##          Neg Pred Value : 0.48097
##              Prevalence : 0.98927
##          Detection Rate : 0.98861
##    Detection Prevalence : 0.99874
##       Balanced Accuracy : 0.52787
##
##        'Positive' Class : 0
##
```