Code Appendix

Jared Yu

June 8, 2019

```r
# Libraries
library(MVN); library(gridExtra); library(ztable); library(ICSNP); library(data.table)
library(devtools); library(ggbiplot); library(ggfortify); library(rrcov); library(MASS)
# Load data
# https://data.world/craigkelly/usda-national-nutrient-db/workspace/file?filename=nndb_flat.csv
nutrition <- read.csv('nndb_flat.csv')

# Filter out some food groups, columns
nutrition <- subset(nutrition, nutrition[,2] %in% c("Baked Products",
  "Cereal Grains and Pasta", "Finfish and Shellfish Products",
  "Lamb, Veal, and Game Products", "Pork Products", "Vegetables and Vegetable Products",
  "Beef Products", "Dairy and Egg Products", "Fats and Oils", "Fruits and Fruit Juices",
  "Legumes and Legume Products", "Nut and Seed Products", "Poultry Products"))

nutrition$FoodGroup <- droplevels(nutrition$FoodGroup)

# Multivariate Normality Check
# https://cran.r-project.org/web/packages/MVN/vignettes/MVN.pdf
mardia <- mvn(data = nutrition[nutrition$FoodGroup %in%
  c('Beef Products', 'Vegetables and Vegetable Products'), c(8:30)], mvnTest = "mardia")

hz <- mvn(data = nutrition[nutrition$FoodGroup %in%
  c('Beef Products', 'Vegetables and Vegetable Products'), c(8:30)], mvnTest = "hz")

royston <- mvn(data = nutrition[nutrition$FoodGroup %in%
  c('Beef Products', 'Vegetables and Vegetable Products'), c(8:30)], mvnTest = "royston")

# Choose 2 populations: Beef, Vegetables
beef_names <- nutrition[nutrition$FoodGroup %in%
  c('Beef Products', 'Vegetables and Vegetable Products'),]
nut_clean <- nutrition[nutrition$FoodGroup %in%
  c('Beef Products', 'Vegetables and Vegetable Products'),]
nut_clean$FoodGroup <- droplevels(nut_clean$FoodGroup)
beef_names$FoodGroup <- droplevels(beef_names$FoodGroup)
nut_clean <- nut_clean[order(nut_clean$FoodGroup),]
beef_names <- beef_names[order(beef_names$FoodGroup),]
nut_clean <- nut_clean[,c(2,8:30)]
beef_names <- beef_names[,c(2,4,8:30)]

# no NA's
any(is.na(nut_clean))
nrow(nut_clean)
sort(sapply(nut_clean[,-1], function(x) sum(x == 0)))

# Sample estimates
x <- nut_clean[,-1]
y <- factor(nut_clean[,1])
```

```r
comb_s_mean <- colMeans(x)
pop_2_s_mean <- sapply(x, function(w) tapply(w, y, mean))
sample_var_cov <- lapply(split(x, y), var)

# https://stackoverflow.com/questions/42860716/export-dataframe-to-pdf-png-in-r
comb_s_mean <- data.frame(comb_s_mean)
colnames(comb_s_mean) <- c("Sample Mean")
comb_s_mean$Nutrients <- rownames(comb_s_mean)
rownames(comb_s_mean) <- NULL
comb_s_mean$Nutrients <- gsub("_", " ", comb_s_mean$Nutrients)
comb_s_mean$`Sample Mean` <- round(comb_s_mean$`Sample Mean`, 4)
grid.table(comb_s_mean)

colnames(pop_2_s_mean) <- gsub("_", " ", colnames(pop_2_s_mean))
pop_2_s_mean <- t(pop_2_s_mean)
colnames(pop_2_s_mean) <- c("Beef", "Vegetables")
grid.table(round(pop_2_s_mean, 4))

png('var_cov_mat.png', height = 1200, width = 4000)
grid.table(sample_var_cov$`Beef Products`)
dev.off()

# Barplot
# https://stackoverflow.com/questions/29639680/r-table-function-how-to-remove-0-counts
# https://www.r-graph-gallery.com/213-rotating-x-axis-labels-on-barplot/
par(mar = c(10,3,3,3))
nut_table <- sort(table(nutrition[,2]), decreasing = TRUE)
food_group_barplot <- barplot(nut_table, names.arg="",
                              las = 1, main = 'Barplot of Food Groups Frequency')
xlab_names <- names(nut_table)
xlab_names <- gsub(" Products", "", xlab_names)
xlab_names[2] <- "Vegetables"
text(food_group_barplot[,1], -3.7, srt = 60, adj= 1, xpd = TRUE, labels = xlab_names, cex=0.7)

# Distribution table
# https://www.guru99.com/r-sort-data-frame.html
# https://stackoverflow.com/questions/24428051/removing-display-of-row-names-from-data-frame
food_group_dist <- cbind(as.data.frame(prop.table(table(droplevels(nutrition[,2]))))*100),
                         as.data.frame(table(droplevels(nutrition[,2])))[,2])
colnames(food_group_dist) <- c('Food Groups', 'Percentage (%)', 'Count')
food_group_dist <- food_group_dist[order(food_group_dist$Count, decreasing = TRUE),]
rownames(food_group_dist) <- NULL
ztable(food_group_dist, caption = 'Percentage and Count of Food Groups')

# Numerical summary
# https://stackoverflow.com/questions/11346880/r-plot-multiple-box-plots-using-columns-from-data-frame
# https://stackoverflow.com/questions/23050928/error-in-plot-new-figure-margins-too-large-scatter-plot
# https://www.r-bloggers.com/setting-graph-margins-in-r-using-the-par-function-and-lots-of-cow-milk/
X_list <- split(nut_clean[,-1], nut_clean$FoodGroup)
colnames(X_list[[1]]) <- gsub("_", " ", colnames(X_list[[1]]))
colnames(X_list[[2]]) <- gsub("_", " ", colnames(X_list[[2]]))
X_list_summary <- lapply(X_list, function(x) sapply(x, summary))
```

```r
grid.table(t(round(X_list_summary$`Beef Products`, 4)))
grid.table(t(round(X_list_summary$`Vegetables and Vegetable Products`, 4)))

# Boxplots
par(mfrow=c(5,5))
par(mar=c(2,2,5,2))
nut_clean_bar <- subset(nut_clean, nut_clean$Fiber_g < 15 & nut_clean$VitA_mcg < 2000 &
       nut_clean$VitB12_mcg < 40 & nut_clean$VitC_mg < 500 & nut_clean$VitE_mg < 3 &
       nut_clean$Folate_mcg < 200 & nut_clean$Niacin_mg < 20 & nut_clean$Riboflavin_mg < 1.5 &
       nut_clean$Thiamin_mg < 5 & nut_clean$Calcium_mg < 400 & nut_clean$Copper_mcg < 5 &
       nut_clean$Iron_mg < 20 & nut_clean$Magnesium_mg < 200 & nut_clean$Manganese_mg < 3 &
       nut_clean$Selenium_mcg < 100)
beef_names <- subset(beef_names, beef_names$Fiber_g < 15 & beef_names$VitA_mcg < 2000 &
       beef_names$VitB12_mcg < 40 & beef_names$VitC_mg < 500 & beef_names$VitE_mg < 3 &
       beef_names$Folate_mcg < 200 & beef_names$Niacin_mg < 20 & beef_names$Riboflavin_mg < 1.5 &
       beef_names$Thiamin_mg < 5 & beef_names$Calcium_mg < 400 & beef_names$Copper_mcg < 5 &
       beef_names$Iron_mg < 20 & beef_names$Magnesium_mg < 200 & beef_names$Manganese_mg < 3 &
       beef_names$Selenium_mcg < 100)

levels(nut_clean_bar$FoodGroup) = c("Beef", "Vegs")
levels(beef_names$FoodGroup) = c("Beef", "Vegs")
colnames(nut_clean_bar) <- gsub("_", " ", colnames(nut_clean_bar))
colnames(beef_names) <- gsub("_", " ", colnames(beef_names))
# create boxplot
for (i in 2:24) {
  boxplot(nut_clean_bar[,i] ~ FoodGroup,
          data = nut_clean_bar, main = names(nut_clean_bar)[i])
}
mtext("Boxplot of Nutrients for Beef and Vegetable Products",
      side = 3, line = -1, outer = TRUE, cex = 0.8)
dev.off()

### Confidence Interval

# One-sample inference about a mean  vector
#  Compute sample mean vector and
#  sample covariance matrix
# https://stackoverflow.com/questions/3369959/moving-columns-within-a-data-frame-without-retyping
# filter data
beef_filter <- nut_clean_bar[nut_clean_bar$FoodGroup == 'Beef', c(-1,-6)]
beef_names <- beef_names[beef_names$FoodGroup == 'Beef', 2]
vege_filter <- nut_clean_bar[nut_clean_bar$FoodGroup == 'Vegs', c(-1,-6)]
xbar <- colMeans(beef_filter)
xvar <- var(beef_filter)

# One-at-a-time confidence interval for mu_1,...,mu_p
univariate_test <- sapply(beef_filter, function(x) t.test(x, mu = round(mean(x))))
uni_test_pval <- as.data.frame(unlist(univariate_test[3,]))
colnames(uni_test_pval) <- c('p-value')
uni_test_pval$Nutrient <- rownames(uni_test_pval)
uni_test_pval <- subset(uni_test_pval, select = c('Nutrient', 'p-value'))
rownames(uni_test_pval) <- NULL
uni_test_pval[,2] <- round(uni_test_pval[,2], 4)
```

```r
uni_test_pval[uni_test_pval[,2] > 0.05,]
grid.table(uni_test_pval)

uni_ci <- round(t(as.data.frame(univariate_test[4,])), 4)
colnames(uni_ci) <- c("Lower Bound", "Upper Bound")
grid.table(uni_ci)

# Bonferroni confidence interval for mu_1,...,mu_p
bonf_test <- sapply(beef_filter, function(x) t.test(x,
  conf.level = (1-(0.05)/ncol(beef_filter)), mu = round(mean(x))))
bonf_ci <- round(t(as.data.frame(bonf_test[4,])), 4)
colnames(bonf_ci) <- c("Lower Bound", "Upper Bound")
grid.table(bonf_ci)
bonf_test_pval <- min(as.data.frame(unlist(bonf_test[3,])))

# Hotelling confidence interval for mu_1,...,mu_p
p_hotelling <- ncol(beef_filter)
n_hotelling <- nrow(beef_filter)
hotelling_crit_value <- ((p_hotelling*(n_hotelling - 1))/(n_hotelling - p_hotelling)) *
       qf(p = .95, df1 = p_hotelling, df2 = (n_hotelling - p_hotelling))
hotelling_lb <- xbar - sqrt(hotelling_crit_value * diag(xvar))
hotelling_ub <- xbar + sqrt(hotelling_crit_value * diag(xvar))
hotelling_ci <- round(as.data.frame(cbind(hotelling_lb, hotelling_ub)), 4)
colnames(hotelling_ci) <- c("Lower Bound", "Upper Bound")
grid.table(hotelling_ci)

# Compute Hotelling statistic
p <- length(colnames(beef_filter))
n <- nrow(beef_filter)
nullmean <- round(xbar)
d <- xbar-nullmean
t2 <- n*t(d)%*%solve(xvar)%*%d
cval <- (n-1)*p/(n-p)*qf(0.95,p,n-p)

t2mod <- (n-p)*t2/(p*(n-1))
pval <- 1- pf(t2mod,p,n-p)

cat("Hotelling T-squared statistic", fill=T)
t2

cat("p-value", fill=T)
pval

# alternative using the function in the ISCP package

HotellingsT2(X = beef_filter, mu = nullmean)

# Confidence Region
# https://stackoverflow.com/questions/35805555/return-max-correlation-and-row-name-from-corr-matrix
beef_cor <- cor(beef_filter[,-1])
beef_cor_triangle <- setDT(melt(beef_cor))[Var1 != Var2, .SD[which.max(value)], keyby=Var1]
max(beef_cor_triangle$value)
# Protein and Fat
```

```r
datatemp<-beef_filter[,c(2, 3)]
colnames(datatemp)<-c("Protein","Fat")

conf.reg<-function(xdata,alpha){
  if(ncol(xdata)!=2) stop("Only for bivariate normal")
  n<-nrow(xdata)
  xbar<-colMeans(xdata)
  S<-cov(xdata)
  es<-eigen(S)
  e1<-es$vec %*% diag(sqrt(es$val))
  r1<-sqrt(qf(alpha,2,n-2))*sqrt(2*(n-1)/(n*(n-2)))
  theta<-seq(0,2*pi,len=250)
  v1<-cbind(r1*cos(theta), r1*sin(theta))
  pts<-t(xbar-(e1%*%t(v1)))
  plot(pts,type="l",main="Confidence Region for Bivariate Normal",xlab=colnames(xdata)[1],ylab=colnames
  segments(0,xbar[2],xbar[1],xbar[2],lty=2) # highlight the center
  segments(xbar[1],0,xbar[1],xbar[2],lty=2)

  th2<-c(0,pi/2,pi,3*pi/2,2*pi)   #adding the axis
  v2<-cbind(r1*cos(th2), r1*sin(th2))
  pts2<-t(xbar-(e1%*%t(v2)))
  segments(pts2[3,1],pts2[3,2],pts2[1,1],pts2[1,2],lty=3)
  segments(pts2[2,1],pts2[2,2],pts2[4,1],pts2[4,2],lty=3)

}

conf.reg(datatemp,alpha=0.95)


# Compute 95% simultaneous confidence intervals for the two mean values
p<-ncol(datatemp)
n<-nrow(datatemp)
S<-cov(datatemp)
xbar<-colMeans(datatemp)
mu1.L=xbar[1]-sqrt(((n-1)*p/(n-p))*qf(0.95,p,n-p))*sqrt(S[1,1]/n)
mu1.U=xbar[1]+sqrt(((n-1)*p/(n-p))*qf(0.95,p,n-p))*sqrt(S[1,1]/n)
mu2.L=xbar[2]-sqrt(((n-1)*p/(n-p))*qf(0.95,p,n-p))*sqrt(S[2,2]/n)
mu2.U=xbar[2]+sqrt(((n-1)*p/(n-p))*qf(0.95,p,n-p))*sqrt(S[2,2]/n)
c(mu1.L,mu1.U)
c(mu2.L,mu2.U)

lines(c(mu1.L,mu1.L),c(0.53,mu2.U),lty=2,col=2,lwd=2)
lines(c(mu1.U,mu1.U),c(0.53,mu2.U),lty=2,col=2,lwd=2)
lines(c(0.49,mu1.U),c(mu2.L,mu2.L),lty=2,col=2,lwd=2)
lines(c(0.49,mu1.U),c(mu2.U,mu2.U),lty=2,col=2,lwd=2)

# Compute 95% Bonferroni confidence intervals for the two mean values
mu1.LB=xbar[1]-qt(0.05/(2*p),n-1,lower.tail=F)*sqrt(S[1,1]/n)
mu1.UB=xbar[1]+qt(0.05/(2*p),n-1,lower.tail=F)*sqrt(S[1,1]/n)
mu2.LB=xbar[2]-qt(0.05/(2*p),n-1,lower.tail=F)*sqrt(S[2,2]/n)
mu2.UB=xbar[2]+qt(0.05/(2*p),n-1,lower.tail=F)*sqrt(S[2,2]/n)
c(mu1.LB,mu1.UB)
c(mu2.LB,mu2.UB)
```

```r
# Plot the confidence intervals together with the confidence ellipse:

lines(c(mu1.LB,mu1.LB),c(0.53,mu2.UB),lty=3,col=3,lwd=2)
lines(c(mu1.UB,mu1.UB),c(0.53,mu2.UB),lty=3,col=3,lwd=2)
lines(c(0.49,mu1.UB),c(mu2.LB,mu2.LB),lty=3,col=3,lwd=2)
lines(c(0.49,mu1.UB),c(mu2.UB,mu2.UB),lty=3,col=3,lwd=2)
legend("topright",
       legend = c("95% Confidence Interval T^2", "95% Confidence Interval Bonferroni"),
       col = c("red", "green"), lty = 2, cex = 0.7)


#### two-sample Hotelling's T2 test  -------
# now we perform the two-sample Hotelling T^2-test
HotellingsT2(beef_filter, vege_filter)

n<-c(nrow(beef_filter),nrow(vege_filter))
p<-ncol(beef_filter)
xmean1<-colMeans(beef_filter)
xmean2<-colMeans(vege_filter)
d<-xmean1-xmean2
S1<-var(beef_filter)
S2<-var(vege_filter)
Sp<-((n[1]-1)*S1+(n[2]-1)*S2)/(sum(n)-2)
t2 <- t(d)%*%solve(sum(1/n)*Sp)%*%d
t2
alpha<-0.05
cval <- (sum(n)-2)*p/(sum(n)-p-1)*qf(1-alpha,p,sum(n)-p-1)
cval

# since we reject the null, we use the simultaneous confidence intervals
# to check the significant components

# simultaneous confidence intervals
alpha<-0.05
wd<-sqrt(((n[1]+n[2]-2)*p/(n[1]+n[2]-p-1))*qf(1-alpha,p,n[1]+n[2]-p-1))*sqrt(diag(Sp)*sum(1/n))
Cis<-cbind(d-wd,d+wd)
cat("95% simultaneous confidence interval","\n")
Cis <- as.data.frame(Cis)
colnames(Cis) <- c("Lower Bound", "Upper Bound")
Cis <- round(Cis, 4)
grid.table(Cis)
Cis[Cis[,1] < 0 & Cis[,2] > 0,]

#Bonferroni simultaneous confidence intervals
wd.b<- qt(1-alpha/(2*p),n[1]+n[2]-2) *sqrt(diag(Sp)*sum(1/n))
Cis.b<-cbind(d-wd.b,d+wd.b)
cat("95% Bonferroni simultaneous confidence interval","\n")
Cis.b <- as.data.frame(Cis.b)
colnames(Cis.b) <- c("Lower Bound", "Upper Bound")
Cis.b <- round(Cis.b, 4)
grid.table(Cis.b)
Cis.b[Cis.b[,1] < 0 & Cis.b[,2] > 0,]
```

```r
# both component-wise simultaneous confidence intervals
# do not contain 0, so they have significant differences.

### PCA
# subset only beef, remove sugar and labels
beef.pc <- princomp(beef_filter, cor=T)

# Showing the coefficients of the components:
summary(beef.pc,loadings=T)

# Showing the eigenvalues of the correlation matrix:
beef_lambda <- (beef.pc$sdev)^2
lambda_df <- as.data.frame(beef_lambda)
colnames(lambda_df) <- c("Lambda")
lambda_df$PC <- rownames(lambda_df)
rownames(lambda_df) <- NULL
lambda_df <- subset(lambda_df, select = c('PC', 'Lambda'))
grid.table(lambda_df)

# A scree plot:
par(mfrow=c(1,2))
plot(1:(length(beef.pc$sdev)),  (beef.pc$sdev)^2, type='b',
     main="Scree Plot", xlab="Number of Components", ylab="Eigenvalue Size")
### plot proportion version
plot(cumsum(beef_lambda/sum(beef_lambda)),
     main = 'Proportion of Variance from Cumulative Eigenvalues',
     ylab = 'Proportion of variance',
     xlab = 'Eigenvalue Index')
abline(h = .8)
legend("bottomright", legend = "80% Cutoff", lty = 1)
dev.off()
beef_loadings <- round(beef.pc$loadings[,1:3], 4)
grid.table(beef_loadings)

# What seems to be a reasonable number of PCs to use?

# Plotting the PC scores for the sample data in the space of the first two principal components:
set.seed(135)
beef_pc_index <- sample(1:nrow(beef.pc$scores))[1:50]
par(pty="s")
plot(beef.pc$scores[beef_pc_index,1],
     beef.pc$scores[beef_pc_index,2],
     xlab="PC 1", ylab="PC 2", type ='n', lwd=2)
text(beef.pc$scores[beef_pc_index,1],
     beef.pc$scores[beef_pc_index,2], cex=0.7, lwd=2)
biplot(beef.pc)

## explore the subgroups
ggbiplot(beef.pc, ellipse = TRUE, varname.adjust = 2) +
  ggtitle("Beef Biplot")
score1 = beef.pc$scores[,1]
score2 = beef.pc$scores[,2]
std.score1 = (score1 - mean(score1))/sd(score1)
```

```r
std.score2 = (score2 - mean(score2))/sd(score2)

grp1 = which((std.score1 > -3  & std.score1 < -1) & (std.score2 > 3 & std.score2 < 4.5))
beef_names[grp1]

grp2 = which((std.score1 > 4))
beef_names[grp2]
beef_names[grepl("Beef, Australian, Wagyu", beef_names)]

grp3 = which((std.score1 > 2.5  & std.score1 < 4) & (std.score2 > -1 & std.score2 < 1.5))
beef_names[grp3]

grp4 = which((std.score1 > -1  & std.score1 < 0) & (std.score2 > 2.5 & std.score2 < 3.1))
beef_names[grp4]

grp5 = which((std.score1 > -2  & std.score1 < 2) & (std.score2 > -3 & std.score2 < 2.5))
sort(beef_names[grp5])

grp6 = which((std.score2 > 4.5))
sort(beef_names[grp6])

a = apply(beef.pc$scores[grp6,], 2, median)
b = apply(beef.pc$scores[-grp6,], 2, median)
colnames(nutrition)
names(nut_clean)[-c(1, 6)]

plot(a, type = 'b')
points(x = 1:22, y = b)

# https://stackoverflow.com/questions/2370515/how-to-get-row-index-number-in-r
beef_names[which(as.numeric(rownames(beef.pc$scores)) == '7516')]
beef_names[which(as.numeric(rownames(beef.pc$scores)) == '7519')]

### data visualization
### beef and vegetable biplot
X <- nut_clean[,c(-1, -6)]
groupid <-nut_clean[,1]
X.pca <- princomp(~.,data=X)
summary(X.pca,loadings=TRUE)
plot(X.pca$scores[,1],X.pca$scores[,2],xlab="PC1",ylab="PC2",
     pch=rep(1:2,n),col=groupid,main="Iris data")
ggbiplot(X.pca, alpha = 0.1, ellipse = TRUE, groups = groupid) + xlim(-2, 0.5) + ylim(-2, 1)
legend("bottomleft",legend=levels(groupid),pch=1:3,col=1:3,cex=0.7)

pc1 = X.pca$scores[,1]
pc2 = X.pca$scores[,2]
spc1 = (pc1-mean(pc1))/sd(pc1)
spc2 = (pc2-mean(pc2))/sd(pc2)
outlier1 = which.min(spc2)
outlier2 = which(spc1 < -1)

# redo pc without outliers
new_groupid <-nut_clean[-c(outlier1, outlier2),1]
```

```r
levels(new_groupid) = c("Beef", "Vegetable")
new_X.pca <- princomp(~.,data=X[-c(outlier1, outlier2),])
ggbiplot(new_X.pca, ellipse = TRUE, groups = new_groupid) +
  ggtitle("Beef and Vegetable Biplot")

### LDA
par(mar=c(4,4,2,1))
# http://www.cookbook-r.com/Manipulating_data/Randomizing_order/
set.seed(135)
comb_filter <- rbind(beef_filter[sample(1:nrow(beef_filter))[1:50], c(1,18)],
                     vege_filter[sample(1:nrow(vege_filter))[1:50], c(1,18)])
comb_filter$Label <- c(rep('Beef', 50), rep('Vegetables', 50))
lda.obj<-lda(Label~.,data=comb_filter,prior=c(1,1)/2)
plda<-predict(object=lda.obj,newdata=comb_filter)

#plot the decision line
gmean <- lda.obj$prior %*% lda.obj$means
const <- as.numeric(gmean %*%lda.obj$scaling)
slope <- - lda.obj$scaling[1] / lda.obj$scaling[2]
intercept <- const / lda.obj$scaling[2]
#Plot decision boundary
plot(comb_filter[,1:2],pch=rep(c(18,20),each=50),col=rep(c(2,4),each=50),
     main = 'LDA of Energy and Magnesium from Beef and Vegetables')
# plot(salmon[,2:3],pch=rep(c(18,20),each=50),col=rep(c(2,4),each=50))
abline(intercept, slope)
# legend("topright",legend=c("Alaskan","Canadian"),pch=c(18,20),col=c(2,4))
legend("topright",legend=c("Beef","Vegetables"),pch=c(18,20),col=c(2,4))
#A Stacked Histogram of the LDA Values
ldahist(data = plda$x[,1], g=comb_filter$Label, main = 'Stacked Histogram of LDA Values')
# Confusion matrix
table(comb_filter$Label,plda$class)
```