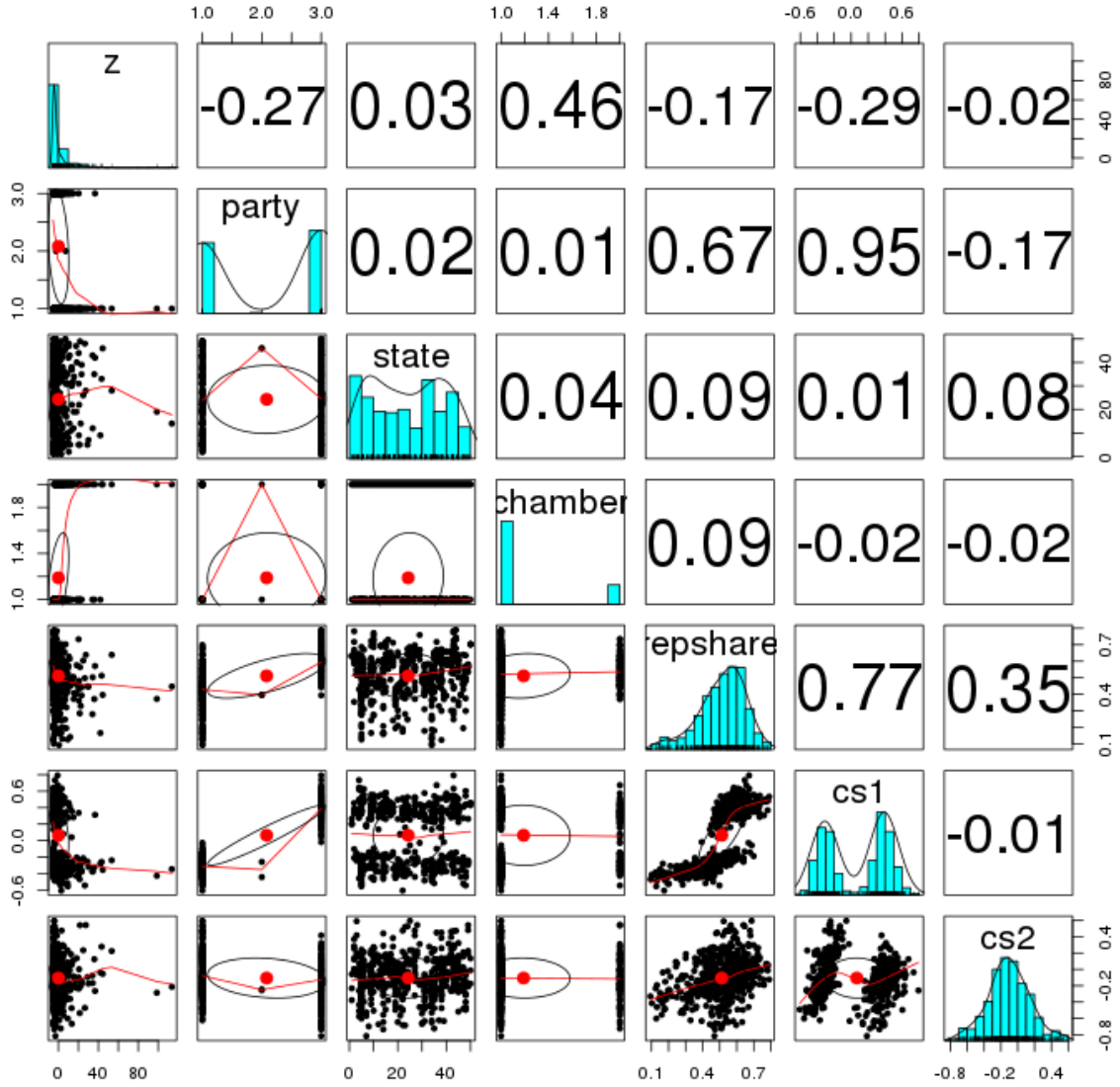
StatMod2 - Exercises 5 - PCA - Question 3 - Congress Data

Maurice Diesendruck

May 5, 2015

3.1 Evaluation of First Principal Component

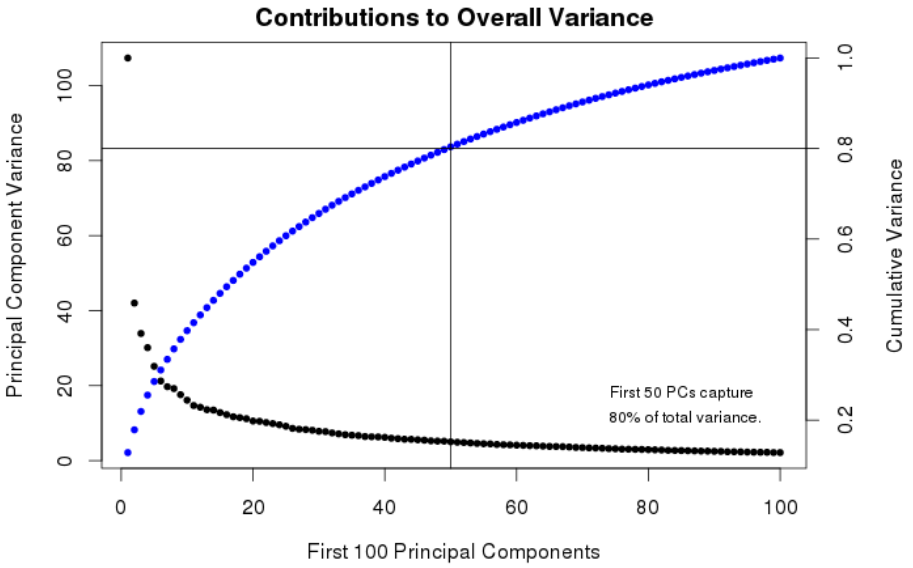
Using "congress109.csv" and "congress109members.csv", project data onto first principal component (i.e. get vector $Z = Yv_1$, where v_1 is the first column of the right-singular matrix of $Y = U\Sigma V'$). Then, merge with data about people, and identify relationships. A `pairs.panels` output is displayed.



The first principal component appears to be most strongly related to `chamber`, where a higher z is associated with being in the smaller chamber (presumed to be the "senate", versus the "house").

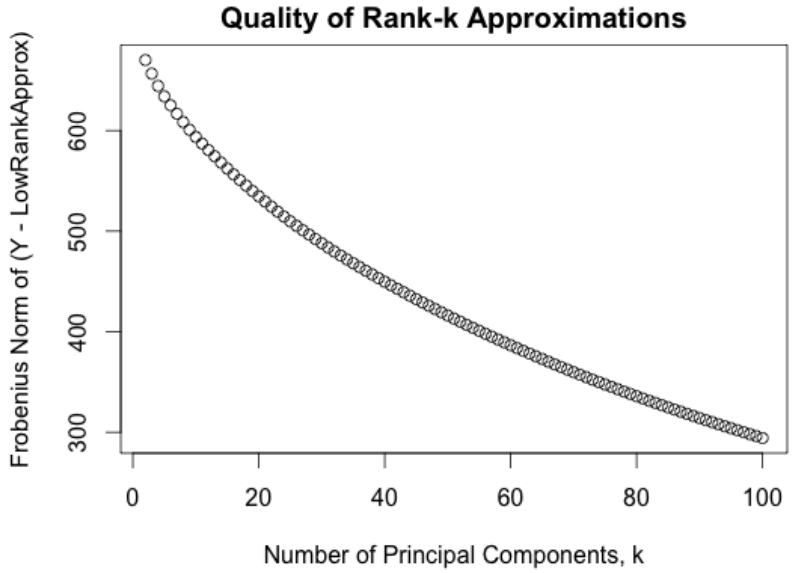
3.2 Rank k Approximation. Which k?

Each principal component accounts for a portion of the overall variance. What should k be for a rank-k approximation? One heuristic is to choose k so that the cumulative sum of those k variances is 80% of the total variance. The following graph shows that for this data set, $k = 50$.



3.3 Comparing Lower-Rank Approximations Using Frobenius Norm

The Frobenius norm can measure how close the rank-k approximation is to the original data matrix. Results are shown below, demonstrating that including more principal components generally improves "closeness" to the original data.



3.4 Full R Code

```
# StatMod2 - Latent Feature Models (PCA)

library(Matrix)
library(psych)

data <- read.csv("congress109.csv")
data2 <- read.csv("congress109members.csv")

# Remove column of names.
Y <- as.matrix(data[,-1])
n <- dim(Y)[1]
p <- dim(Y)[2]
names(Y) <- seq(1:dim(Y)[2]) # Enables compact printing.
# Standardize columns of Y to be mean=0, sd=1.
for (c in 1:p) {
  Y[,c] <- (Y[,c] - mean(Y[,c]))/sqrt(var(Y[,c]))
}

# Do SVD of Y.
decomp.y <- svd(Y)
U.y <- as.matrix(decomp.y$u)
D.y <- as.matrix(diag(decomp.y$d))
V.y <- as.matrix(decomp.y$v)

# Do eigenvalue decomposition of S=(1/n)*t(Y)*Y
S1 <- cov(Y)
S <- (1/n)*t(Y)%*%Y
colnames(S) <- seq(1:dim(S)[2]) # Enables compact printing.
rownames(S) <- seq(1:dim(S)[2])
decomp.s <- eigen(S)
vals.s <- decomp.s$values
vecs.s <- decomp.s$vectors

# Project all values onto 1st column of V.
z <- Y%*%vecs.s[,1]
pc1.in.context <- cbind(z, data2[2:7])
pairs.panels(pc1.in.context)

# Test variance of projections using first 100 columns of V.
range <- 1:100
t <- NULL
cumulative.vars <- NULL
for (i in range) {
  t <- c(t, var(Y%*%vecs.s[,i]))
  cumulative.vars[i] <- sum(t[1:i])
```

```
}
cumulative.vars.scaled <- cumulative.vars/max(cumulative.vars)
pc.cutoff <- min(which(cumulative.vars.scaled>0.8))

# Plot cumulative variance.
par(mar = c(5,5,2,5))
plot(range, t, ylab="Principal Component Variance",
     xlab=bquote("First"~.(length(range))~"Principal Components"),
     main="Contributions to Overall Variance")
text(85, 15, cex=.75,
     bquote(atop("First"~.(pc.cutoff)~"PCs capture",
                 "  80% of total variance.")))
par(new = T)
plot(range, cumulative.vars.scaled,
     col = "blue", axes = F, xlab = NA, ylab = NA)
axis(side = 4)
mtext(side = 4, line = 3, "Cumulative Variance")
abline(h=0.8)
abline(v=pc.cutoff)

# Get a lower-rank approximation of Y, using the top eigenvalues that describe
# 80% of the total variance.
LowRankApprox <- function(U.y, D.y, V.y, pc.cutoff) {
  lr.U <- U.y[,1:pc.cutoff]
  lr.D <- D.y[1:pc.cutoff,1:pc.cutoff]
  lr.V <- V.y[,1:pc.cutoff]
  lr.Y <- lr.U%*%lr.D%*%t(lr.V)
  return (lr.Y)
}

# Compute difference between original and low-rank approximation of Y.
frob.dist <- NULL
for (c in 2:100) {
  lr.approx <- LowRankApprox(U.y, D.y, V.y, c)
  frob.dist[c] <- norm(Y-lr.approx)
}
plot(2:100, frob.dist[2:100], xlab="Number of Principal Components, k",
     ylab="Frobenius Norm of (Y - LowRankApprox)",
     main="Quality of Rank-k Approximations")
```