HW13

Answers are in **black** or red text.

**Question 1)** This time, let’s try to capture as much variance of all these independent variables as possible. Let’s start by recreating the cars\_log dataset, which log-transforms all variables except model year and origin. Important: remove any rows that have missing values.

auto <- read.table("auto-data.txt", header=FALSE, na.strings = "?", stringsAsFactors = F)  
names(auto) <- c("mpg", "cylinders", "displacement", "horsepower", "weight",  
"acceleration", "model\_year", "origin", "car\_name")  
cars\_log <- with(auto, data.frame(log(mpg), log(cylinders), log(displacement), log(horsepower), log(weight), log(acceleration), model\_year, origin))  
cars\_log <- na.omit(cars\_log)

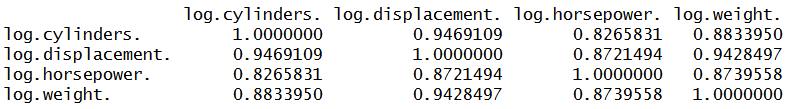
**a).** Create a new data.frame of the four log-transformed independent variables with multicollinearity

1. Give this smaller data frame an appropriate name (think what they jointly mean)

colinear\_var <- cars\_log[,c("log.cylinders.","log.displacement.","log.horsepower.","log.weight.")]

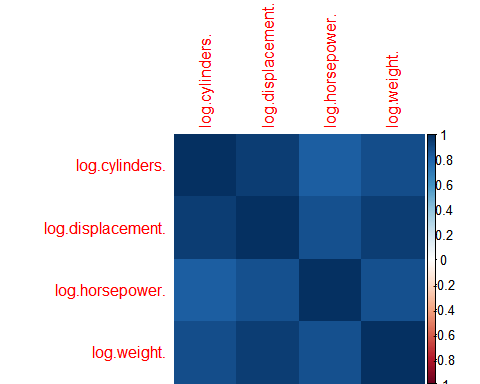
1. Check the correlation table of these four variables to confirm they are indeed collinear

cor(colinear\_var)

****

library(corrplot)

corrplot(cor(colinear\_var),method="color")



The above figure shows all blue, which means they're collinear.

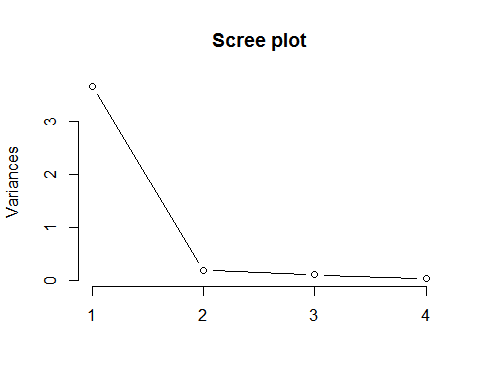
**b).** Let’s analyze the principal components of the four collinear variables

1. How many principal components are needed to summarize these four variables? (use the eigenvalues and scree plot criteria we discussed in class)

eigenvalue <- eigen(cor(colinear\_var))$values  
princi\_com <- eigen(cor(colinear\_var))$vectors  
eigenvalue

## [1] 3.67425879 0.18762771 0.10392787 0.03418563

screeplot(prcomp(colinear\_var,scale.=TRUE),type = "line",main = "Scree plot")



According to the "eigenvalue > 1 criteria" and "screeplot criteria", I think we should take one principal component.

1. How much variance of the four variables is explained by their first principal component? (a summary of the pca reports it, but try computing this from the eigenvalues alone)

#use summary function  
summary(prcomp(colinear\_var,scale. = T))

## Importance of components:  
## PC1 PC2 PC3 PC4  
## Standard deviation 1.9168 0.43316 0.32238 0.18489  
## Proportion of Variance 0.9186 0.04691 0.02598 0.00855  
## Cumulative Proportion 0.9186 0.96547 0.99145 1.00000

#computing proportion from the eigenvalues  
eigenvalue[1]/sum(eigenvalue)

## [1] 0.9185647

About 91 percent of the variance was is explained by the first principal component.

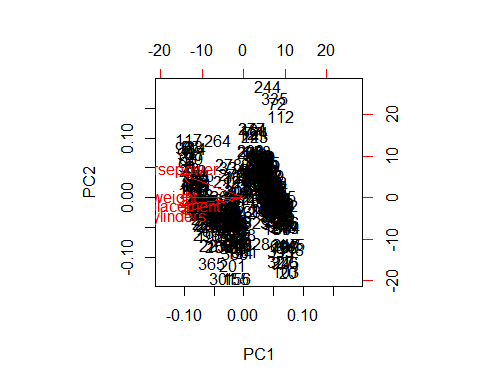
1. Looking at the values and valence (positive/negative) of the first principal component’s eigenvector, what would you call the information captured by this component? (i.e., think what the first principal component means)

prcomp(colinear\_var,scale. = T)

## Standard deviations:  
## [1] 1.9168356 0.4331601 0.3223785 0.1848936  
##   
## Rotation:  
## PC1 PC2 PC3 PC4  
## log.cylinders. -0.4979145 -0.53580374 0.52633608 0.4335503  
## log.displacement. -0.5122968 -0.25665246 -0.07354139 -0.8162556  
## log.horsepower. -0.4856159 0.80424467 0.34193949 0.0210980  
## log.weight. -0.5037960 0.01530917 -0.77500928 0.3812031

The first principal component might represent the average of those four variables. Since those four variable are highly correlated, the first principal component summarizes their characteristics "equally". Therefore, the values in eigenvectors are close to each other.

biplot(prcomp(colinear\_var,scale. = T))



**c).** Let’s reduce the four collinear variables into one new variable!

1. Store the scores of the first principal component as a new column of cars\_log
2. Name this column appropriately based on the meaning of this first principal component

cars\_log$average\_abi <- prcomp(colinear\_var,scale. = T)$x[,1]

**d).** Let’s revisit our regression analysis on cars\_log: (HINT: to compare variables across models, it helps to conduct fully standardized regression)

1. Regress mpg over weight, acceleration, model\_year and origin

regr1 <- lm(data = cars\_log, log.mpg. ~ log.weight. + log.acceleration. + model\_year + factor(origin))  
summary(regr1)

##   
## Call:  
## lm(formula = log.mpg. ~ log.weight. + log.acceleration. + model\_year +   
## factor(origin), data = cars\_log)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.38259 -0.07054 0.00401 0.06696 0.39798   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 7.410974 0.316806 23.393 < 2e-16 \*\*\*  
## log.weight. -0.875499 0.029086 -30.101 < 2e-16 \*\*\*  
## log.acceleration. 0.054377 0.037132 1.464 0.14389   
## model\_year 0.032787 0.001731 18.937 < 2e-16 \*\*\*  
## factor(origin)2 0.056111 0.018241 3.076 0.00225 \*\*   
## factor(origin)3 0.031937 0.018506 1.726 0.08519 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.1163 on 386 degrees of freedom  
## Multiple R-squared: 0.8845, Adjusted R-squared: 0.883   
## F-statistic: 591.1 on 5 and 386 DF, p-value: < 2.2e-16

1. Repeat the regression, but replace weight with the factor scores of the 1st principal component of our collinear independent variables

regr2 <- lm(data = cars\_log, log.mpg. ~ average\_abi + log.acceleration. + model\_year + factor(origin))  
summary(regr2)

##   
## Call:  
## lm(formula = log.mpg. ~ average\_abi + log.acceleration. + model\_year +   
## factor(origin), data = cars\_log)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.51137 -0.06050 -0.00183 0.06322 0.46792   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.398114 0.166554 8.394 8.99e-16 \*\*\*  
## average\_abi 0.145663 0.005057 28.804 < 2e-16 \*\*\*  
## log.acceleration. -0.191482 0.041722 -4.589 6.02e-06 \*\*\*  
## model\_year 0.029180 0.001810 16.122 < 2e-16 \*\*\*  
## factor(origin)2 0.008272 0.019636 0.421 0.674   
## factor(origin)3 0.019687 0.019395 1.015 0.311   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.1199 on 386 degrees of freedom  
## Multiple R-squared: 0.8772, Adjusted R-squared: 0.8756   
## F-statistic: 551.6 on 5 and 386 DF, p-value: < 2.2e-16

1. Use VIF scores to check whether the either regression suffers from multicollinearity

library(car)

## Warning: package 'car' was built under R version 3.3.3

cat("Origin regression")

## Origin regression

vif(regr1)

## GVIF Df GVIF^(1/(2\*Df))  
## log.weight. 1.933208 1 1.390398  
## log.acceleration. 1.304761 1 1.142261  
## model\_year 1.175545 1 1.084225  
## factor(origin) 1.710178 2 1.143564

cat("\nPCA regression")

##   
## PCA regression

vif(regr2)

## GVIF Df GVIF^(1/(2\*Df))  
## average\_abi 2.555002 1 1.598437  
## log.acceleration. 1.549953 1 1.244971  
## model\_year 1.208800 1 1.099454  
## factor(origin) 1.845979 2 1.165619

According to the VIF scores, neither of them suffer from multicollinearity.

1. (ungraded) Comparing the two regressions, how has the story changed?

log.acceleration. become significant in regression including PC1.

**Question 2)** An online marketing firm is studying how customers who shop on e-commerce websites over the winter holiday season perceive the security of e-commerce sites. Based on feedback from experts, the company has created eighteen questions (see ‘questions’ tab of excel file) regarding important security considerations at e-commerce websites. Over 400 customers responded to these questions (see ‘data’ tab of Excel file). Respondents were asked to consider a shopping site they were familiar with when answering questions (site was chosen randomly from those each subject has recently visited). The company now wants to use the results of these eighteen questions to reveal if there are some underlying dimensions of people’s perception of online security that effectively capture the variance of these eighteen questions. Let’s analyze the principal components of the eighteen items.

sec\_q <- read.csv("security\_questions.csv")

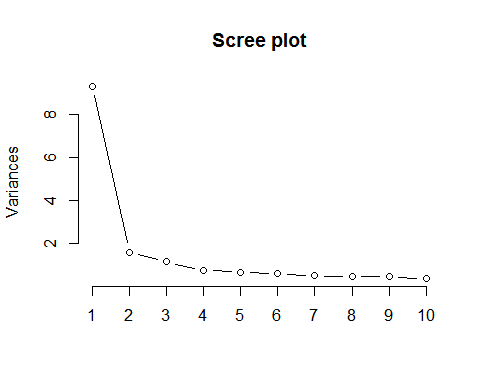
**a).** How much variance did each extracted factor explain?

summary(prcomp(sec\_q,scale. = T))

## Importance of components:  
## PC1 PC2 PC3 PC4 PC5 PC6  
## Standard deviation 3.0514 1.26346 1.07217 0.87291 0.82167 0.78209  
## Proportion of Variance 0.5173 0.08869 0.06386 0.04233 0.03751 0.03398  
## Cumulative Proportion 0.5173 0.60596 0.66982 0.71216 0.74966 0.78365  
## PC7 PC8 PC9 PC10 PC11 PC12  
## Standard deviation 0.70921 0.68431 0.67229 0.6206 0.59572 0.54891  
## Proportion of Variance 0.02794 0.02602 0.02511 0.0214 0.01972 0.01674  
## Cumulative Proportion 0.81159 0.83760 0.86271 0.8841 0.90383 0.92057  
## PC13 PC14 PC15 PC16 PC17 PC18  
## Standard deviation 0.54063 0.51200 0.48433 0.4801 0.4569 0.4489  
## Proportion of Variance 0.01624 0.01456 0.01303 0.0128 0.0116 0.0112  
## Cumulative Proportion 0.93681 0.95137 0.96440 0.9772 0.9888 1.0000

**b).** Show a scree plot of factors extracted

screeplot(prcomp(sec\_q,scale.=TRUE),type = "line",main = "Scree plot")



**c).** How many factors should we retain in our analysis? (judge using the criteria we’ve discussed)

eigen(cor(sec\_q))$values

## [1] 9.3109533 1.5963320 1.1495582 0.7619759 0.6751412 0.6116636 0.5029855  
## [8] 0.4682788 0.4519711 0.3851964 0.3548816 0.3013071 0.2922773 0.2621437  
## [15] 0.2345788 0.2304642 0.2087471 0.2015441

From the screeplot and eigenvalues, we could retain the first three principal components.

**d).** (ungraded) Can you interpret what any of the principal components mean? Try guessing the meaning of the first few principal components

prcomp(sec\_q,scale. = T)

## Standard deviations:  
## [1] 3.0513855 1.2634603 1.0721745 0.8729123 0.8216697 0.7820893 0.7092147  
## [8] 0.6843090 0.6722880 0.6206419 0.5957194 0.5489145 0.5406268 0.5119997  
## [15] 0.4843334 0.4800669 0.4568885 0.4489367  
##   
## Rotation:  
## PC1 PC2 PC3 PC4 PC5  
## Q1 -0.2677422 0.110341691 -0.001973491 0.126220668 -0.048468417  
## Q2 -0.2204272 0.010886972 0.083171536 0.258122218 0.093887919  
## Q3 -0.2508767 0.025878543 0.083648794 -0.399268076 -0.061766335  
## Q4 -0.2042919 -0.508981768 0.100759585 0.040690031 -0.072913141  
## Q5 -0.2261544 0.024745268 -0.505845415 0.052574743 -0.193207848  
## Q6 -0.2237681 0.082805088 0.193281966 -0.004209098 0.611348765  
## Q7 -0.2151891 0.251398450 0.302354487 0.327318232 0.008596733  
## Q8 -0.2576225 -0.033526840 -0.320109219 0.076017162 0.209097752  
## Q9 -0.2369512 0.183342667 0.189853454 -0.124795087 0.025138160  
## Q10 -0.2248660 0.078103267 -0.496820932 -0.034236123 -0.249119125  
## Q11 -0.2467645 0.206580870 0.160903091 0.264607608 -0.210724202  
## Q12 -0.2065785 -0.504591429 0.113342400 0.060346524 0.052819352  
## Q13 -0.2333066 0.051159791 0.078658760 -0.602543012 -0.030357718  
## Q14 -0.2659342 0.078910404 0.146232765 -0.362581586 -0.086718158  
## Q15 -0.2307289 -0.008373326 -0.310161141 0.069411508 0.513508897  
## Q16 -0.2482681 0.160524168 0.170839887 0.204337585 -0.342722070  
## Q17 -0.2023781 -0.525747030 0.102652280 0.080754652 -0.157376900  
## Q18 -0.2643810 0.089915229 -0.060800871 0.051492827 -0.024214541  
## PC6 PC7 PC8 PC9 PC10  
## Q1 0.1826730451 -0.47564502 0.011877666 -0.158945743 0.02559547  
## Q2 0.7972988590 0.10381142 0.370484027 0.018906337 -0.01758985  
## Q3 0.1343170710 0.29794768 -0.045361944 0.046160967 0.62920376  
## Q4 -0.0683434170 0.07323286 -0.082718228 0.034011814 0.13146697  
## Q5 0.1493338250 0.19273010 -0.188948821 0.218690034 -0.09878156  
## Q6 0.0551361412 -0.06503361 -0.538423059 0.331476460 0.04348905  
## Q7 -0.0562329401 0.45399251 -0.229822767 -0.236185029 -0.31439194  
## Q8 -0.2005009349 -0.06635056 0.204619876 -0.232217507 -0.08234563  
## Q9 -0.2696485391 0.12766155 0.452229009 0.595761520 -0.25923949  
## Q10 0.0232597277 0.15613131 -0.250158309 0.141066357 -0.09604999  
## Q11 -0.1928970917 -0.01757216 -0.170741343 -0.289466716 0.12972901  
## Q12 -0.0454546580 -0.03110171 0.005586284 0.007633808 -0.16822370  
## Q13 0.0949114194 -0.03589479 -0.013028375 -0.281562536 -0.49131061  
## Q14 -0.0006735609 -0.07224998 0.032286752 -0.224017714 0.12173004  
## Q15 -0.2572918341 0.15806779 0.305772284 -0.250812042 0.19230189  
## Q16 -0.2189544787 -0.03885431 0.186064954 0.134618480 0.21266262  
## Q17 -0.0527365890 0.02827931 -0.038609734 0.023978170 -0.09198523  
## Q18 -0.0327588454 -0.58413134 -0.079484842 0.184214340 0.01232082  
## PC11 PC12 PC13 PC14 PC15  
## Q1 -0.261433547 0.3655136121 -0.09437152 0.21538278 0.107191422  
## Q2 0.141511628 -0.1423173350 -0.01439656 -0.14151031 -0.124321587  
## Q3 -0.215411545 0.0711375730 0.07897104 0.38275058 -0.173199162  
## Q4 -0.182772484 0.0001075882 0.32083974 -0.53718169 -0.009053271  
## Q5 0.090154465 0.0962621836 0.41176540 0.13779948 0.420108616  
## Q6 0.230188841 0.1679270706 -0.06866003 -0.12229591 -0.076584623  
## Q7 -0.441121206 0.0404427953 -0.01046519 0.03486607 0.164646045  
## Q8 -0.218910615 0.3074295739 0.08286262 -0.07220809 -0.517381497  
## Q9 -0.125837984 -0.1387657899 0.06167134 0.06636535 -0.103891809  
## Q10 -0.006787801 -0.1568738426 -0.54451920 -0.17543121 -0.275471410  
## Q11 0.395639123 -0.4128696157 0.22239835 0.14404891 -0.308218564  
## Q12 0.072388580 -0.1181594259 -0.39416050 0.46427132 0.147423769  
## Q13 0.306206763 0.1388173302 0.19909498 0.01118762 -0.042881369  
## Q14 -0.134853427 -0.2306763906 -0.29401321 -0.38305994 0.322075542  
## Q15 0.178156051 -0.1589461038 -0.01621655 0.01470750 0.336177176  
## Q16 0.383866578 0.4817217034 -0.17169894 -0.17403268 0.168614520  
## Q17 0.083760590 0.0503178068 0.03431935 0.09260499 -0.096523523  
## Q18 -0.229097907 -0.3832085961 0.19580495 0.02702597 0.077981920  
## PC16 PC17 PC18  
## Q1 -0.26663363 -0.15892454 0.49709414  
## Q2 0.04539846 0.01378516 -0.07954338  
## Q3 0.10905667 -0.08731092 -0.07451547  
## Q4 -0.26266355 -0.39030988 0.02091260  
## Q5 -0.20508811 0.26389562 -0.07356419  
## Q6 -0.04426883 0.11718533 0.02443898  
## Q7 0.19302912 -0.07574440 -0.08656284  
## Q8 -0.08324463 0.31696165 -0.32212598  
## Q9 -0.19386537 0.01929777 0.22424357  
## Q10 0.07402245 -0.24996841 0.14445897  
## Q11 -0.28230295 0.05599291 0.11746105  
## Q12 -0.29758805 -0.08367724 -0.38027121  
## Q13 0.11740772 -0.26739129 -0.04166051  
## Q14 -0.16553236 0.50553644 -0.01188146  
## Q15 0.18191811 -0.22010115 0.21302663  
## Q16 0.17538230 -0.09232084 -0.26436304  
## Q17 0.51310849 0.39101042 0.42651093  
## Q18 0.42203495 -0.12287014 -0.30773331

The first component seems to summarizes the average score of questions.