Renel Chesak
CSC 424

Homework 2

5. Insurance Data

a. Linear model made from training set:

```
> summary(lmFit)

Call:
lm(formula = newpol ~ ., data = insuranceNumeric[1:30, ])

Residuals:
    Min      1Q  Median      3Q     Max 
-3.3334 -1.5451 -0.1276  1.0662  3.8301 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) 15.4056703  3.7444455   4.114 0.000395 ***
pctmin      -0.0868654  0.0220759  -3.935 0.000621 ***
fires       -0.0748638  0.0664806  -1.126 0.271258    
thefts       0.0138674  0.0182911   0.758 0.455741    
pctold      -0.0770536  0.0233160  -3.305 0.002977 ** 
income      -0.0001124  0.0002153  -0.522 0.606330    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.931 on 24 degrees of freedom
Multiple R-squared:  0.8045,	Adjusted R-squared:  0.7637 
F-statistic: 19.75 on 5 and 24 DF,  p-value: 8.406e-08


> rse = sqrt(sum((yTest - yHat)^2) / 24) #compute predicted rmse using degrees of freedom 
specified
> rse
[1] 1.903018
```

The linear model has performs well on the training set (RSE = 1.9), but performs no better when predicting y-values for the test set (RSE = 1.9).

Code:

```
################
#Problem 6:
################
library(glmnet)
library(corrplot)
library(MASS)

insurancedata = read.table("chicinsur.txt", header=T)
head(insurancedata)

#pull out non-numeric fields and put the newpol column in front because it will
#be our "Y". This makes it easier to interpret the correlation matrices
insuranceNumeric = insurancedata[,c(6, 2:5, 8)]
head(insuranceNumeric)

#########Normal Linear Regression:
#[on a training set (30 out of 47 obs)]
lmFit = lm(newpol ~ ., data=insuranceNumeric[1:30, ])  # train on the first 30
summary(lmFit)
plot(lmFit) #goes through several plots that test model assumptions

#explicitly create the training and test sets:
xTrain = insuranceNumeric[1:30, 2:6]
yTrain = insuranceNumeric[1:30, 1]
xTest = insuranceNumeric[31:47, 2:6]
yTest = insuranceNumeric[31:47, 1]

#How well does this model predict on the test data?
yHat = predict(lmFit, xTest) #feed xTest data into lmFit model to get predicted y values
yHat #confirm that you only have predicted y values for the test set (obs. 31:47)
rse = sqrt(sum((yTest - yHat)^2) / 24) #compute predicted rmse using degrees of freedom
#specified in lm (lmFit)
rse
#visually compare a few of the predicted and actual values:
head(yHat)
head(yTrain)
```

b. Lasso regression with cross-validation:

We are using this to curb overfitting because the number of samples is so small. We are sacrificing the minimization of RSE on the training set in order to further minimize the RSE on the test set.

```
> cvFit <- cv.glmnet(as.matrix(xTrain), yTrain)
> lassoFit = glmnet(as.matrix(xTrain), yTrain, lambda = cvFit$lambda.min) #glmnet on the
> lassoFit

Call:  glmnet(x = as.matrix(xTrain), y = yTrain, lambda = cvFit$lambda.min)

     Df  %Dev Lambda
[1,]  3 0.788 0.3108


> coef(lassoFit, s=0.3108) #this gives us the coefficients for the variables included in
the
6 x 1 sparse Matrix of class "dgCMatrix"
                      1
(Intercept) 12.58419485
pctmin      -0.07785502
fires       -0.03486817
thefts       .
pctold      -0.05735031
income       .
```

The best lambda = 0.3108, and it produces the model:

newpol = 12.6 -0.08*pctmin -0.03*fires -0.06*pctold

```
> yHat = predict(lassoFit, as.matrix(xTrain), s=0.3108)
> rse = sqrt(sum((yTrain - yHat)^2) / 24)
> rse
[1] 2.010149
```

Note that this model has a significantly larger RSE for the training set. In the next part, we will see if it does its job by significantly lowering the RES on the test set.

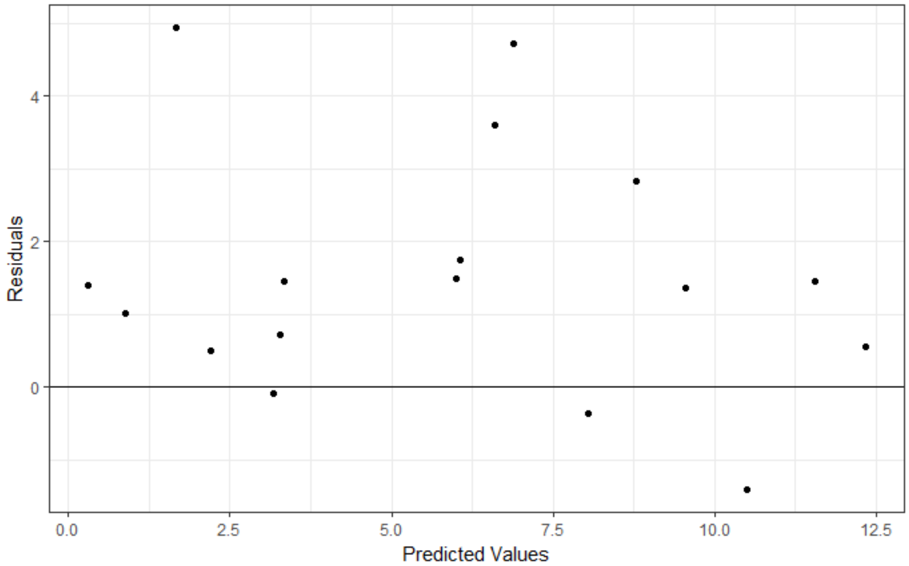
c. Using Lasso model to predict test set:

```
> yPredict = predict(lassoFit, as.matrix(xTest), s=0.3108)
> rsePredict = sqrt(sum((yTest - yPredict)^2) / 24)
> rsePredict
[1] 1.885905
```

The RSE for the test set is slightly lower when using a model optimized by lasso regularized regression (RSE = 1.886), compared to normal linear regression (RSE = 1.903).

c. Residuals vs. the predicted values

```
resid = yTest - yPredict
yep<- data.frame(resid, yPredict)
yep
library(ggplot2)
library(gcookbook)
graph <- ggplot(yep, aes(x=yPredict, y=resid)) + geom_point() + theme_bw()
graph + xlab("Predicted Values") + ylab("Residuals") + geom_hline(yintercept=0)
```



Here we can see that there is some bias in the residuals because they do not appear to have a mean of zero. This means that overall, we are largely underestimating the predicted values. However, it is commonly accepted that bias in the predicted values can be sacrificed for a lower RSE in regularized regression.

6. Employment Data

a. PCA:

```
> p = princomp(employNumeric)
> summary(p)
Importance of components:
                          Comp.1    Comp.2     Comp.3     Comp.4      Comp.5      Comp.6
Standard deviation     17.0817636 6.4823470 3.82393204 2.32861792 1.532782553 1.002896265
Proportion of Variance  0.8157836 0.1174827 0.04088179 0.01516024 0.006568567 0.002811041
Cumulative Proportion   0.8157836 0.9332663 0.97414811 0.98930835 0.995876918 0.998688959
                           Comp.7       Comp.8       Comp.9
Standard deviation     0.63612956 0.2498589145 4.287707e-02
Proportion of Variance 0.00113136 0.0001745417 5.139960e-06
Cumulative Proportion  0.99982032 0.9999948600 1.000000e+00
```

We can see in the summary above that it takes only 2 PCs to explain about 93% of the variance in the data.

b. Meaning of PCs:

```
> p$loadings #this is a matrix of the eigen vectors, and it tries to hide the things

Loadings:
    Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9
Agr  0.892         0.118         0.180 -0.153                0.335
Min                                     0.456 -0.766  0.290  0.324
Man -0.271  0.770  0.185         0.336 -0.201  0.162         0.337
PS                                      0.231        -0.909  0.340
Con                             -0.724 -0.558 -0.194         0.325
SI  -0.192 -0.234 -0.580 -0.608  0.266                0.104  0.337
Fin        -0.130 -0.470  0.781  0.121                0.123  0.334
SPS -0.298 -0.567  0.598         0.236 -0.248                0.332
TC                 0.159        -0.435  0.546  0.567  0.224  0.334

               Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9
SS loadings     1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000
Proportion Var  0.111  0.111  0.111  0.111  0.111  0.111  0.111  0.111  0.111
Cumulative Var  0.111  0.222  0.333  0.444  0.556  0.667  0.778  0.889  1.000
```

PC1 = 0.892*Agr – 0.271*Man – 0.192*SI – 0.298*SPS

PC2 = 0.770*Man – 0.234*SI – 0.12*Fin – 0.567*SPS

NOTE: These PCs are **not standardized**, which means everything can be interpreted in terms of the original units, and all differences in variance across variables has been preserved. This is what we want for this dataset, since all data are already in the same units (percent), and thus any differences in variance are inherent. It looks like PC1 is primarily Arg, while slightly and negatively Man, SI, and SPS. PC2 is primarily Man, significantly and negatively SPS, and slightly and negatively SI and Fin. This means that, for example, if a country's score for Man increased, its score for PC1 would decrease and PC2 would increase. We can see that there is some overlap in the interpretation of these PCs; both contain Man, SI, and SPS (overlap highlighted in yellow). Overlap such as this makes these components difficult to comprehend and interpret.

```
> p2 = psych::principal(employNumeric, rotate="varimax", nfactors=2, scores=TRUE)
> print(p2$loadings, cutoff=.4, sort=F) #the goal of varimax is to get the loadings to be either

Loadings:
    RC1    RC2   
Agr -0.905       
Min        -0.858
Man  0.778       
PS   0.572       
Con  0.600       
SI   0.514  0.706
Fin         0.673
SPS  0.587  0.532
TC   0.743       

                 RC1   RC2
SS loadings    3.356 2.261
Proportion Var 0.373 0.251
Cumulative Var 0.373 0.624
```

RC1 = 0.905*Agr – 0.778*Man - 0.572*PS - 0.600*Con - .0514*SI – 0.587*SPS – 0.743*TC

RC2 = 0.858*Min – 0.706*SI – 0.673*Fin – 0.532*SPS

NOTE: the principal() function **standardizes** the vectors by default. This means that the loadings are in terms of SDs, and any new data plugged into the regression formula must be standardized by the same means and SDs as the data the model was built on. This is not ideal for this dataset, since the original units are all the same (percent), and thus any differences in variance across variables is inherent. After rotating the components using PCA Factor Analysis techniques, we can see that the overlap between the two RCs (rotated principle components) has been slightly reduced (overlap highlighted in yellow); this makes the RCs a little easier to interpret. RC1 is primarily Agr, followed by negatively Man and TC, and finally negatively by Con, SPS, PS, and SI. You would have to speak with a domain expert to know whether these fields of employment could be considered interrelated subgroups of a larger group. You will notice that there are several extra contributors to RC1, which makes it less parsimonious and succinct, and thus may be more difficult to explain to someone. This may be aided by adding more RCs to the model and allowing the variance to spread out among the components.

RC2 is primarily Min, followed negatively by SI, Fin, and SPS. Both components contain a negative and mild relationship to SPS; this makes it slightly more difficult to interpret, but depending on the level of variance we want to capture and the cutoff we want to make for 'significant' contributors, we could remove it altogether.

You may notice that the signs in the output are opposite; this is trivial and I have reversed them in order to facilitate the comparison of the interpretations. Finally, note that the variance explained has been significantly reduced by standardization and rotation of the components (from 0.933 to 0.624).

c. Scores:

```
p2$scores #look at new scores (how each sample scales along each PC). NOTE: these scores are
#centered => they will always be around a mean of zero (so don't try to compare them directly
#to the data multiplied by the rotation matirx)
#The rows are in the same order as the original data so you can compare the original values to
#their new scores on the PCs

#create a data frame that contains the countries and their new scores:
x = as.matrix(p2$scores) #first start with a matrix so you can reverse the signs
g = x*-1 #I am doing this because the sign is trivial, and it makes better interpretive sense
#if they are opposite of what R has defaulted them to
s = as.data.frame(g)
s$Country <-employ[, 1] #attach the country names as a field called 'Country'
s = s[, c(3, 1, 2)] #reorder them to put the country name first
s
attach(s) #allows you to call variables without naming the dataset first (e.g. RC1 vs. s$RC1)
#(adds the variable names to R's current session)
s[order(RC1),] #sort the dataframe rows by highest and lowest values of PC1
s[order(RC2),] #sort the dataframe rows by highest and lowest values of PC2
```

```
> s[order(RC1),] #sort the dataframe rows
          Country          RC1        RC2
21     E. Germany -1.47699154  1.5066273
22        Hungary -0.94550866  1.9127303
20 Czechoslovakia -0.76938198  1.6318345
9  United Kingdom -0.73572258 -0.5045475
7      Luxembourg -0.68667078  0.3095863
10        Austria -0.61702380 -0.1007688
13         Norway -0.61687550 -0.9623251
1         Belgium -0.59832877 -1.0595183
8     Netherlands -0.42357623 -1.5558836
4      W. Germany -0.42309897 -0.1465071
17    Switzerland -0.37293535 -0.6468584
11        Finland -0.33866749 -0.6398245
25           USSR -0.28983489  0.8004834
16         Sweden -0.21102466 -1.1849486
3          France -0.14231031 -0.8391757
2         Denmark -0.03091821 -1.5142897
6           Italy -0.02654030 -0.5526932
19       Bulgaria  0.04934314  1.0742403
5         Ireland  0.13816381 -0.2474394
23         Poland  0.15767562  1.3984567
15          Spain  0.34631705 -0.3223641
14       Portugal  0.66489194 -0.3175746
24        Rumania  0.68842513  1.3599235
12         Greece  1.12858168  0.1194751
26     Yugoslavia  2.14140737  0.1254186
18         Turkey  3.39060427  0.3559428
```

```
#See how original data values rank according to Agr, the primary component of RC1
head(employ)
attach(employ)
employ[order(Agr),]
```

```
> employ[order(Agr),]
          Country  Agr Min  Man  PS  Con   SI  Fin  SPS  TC
9  United Kingdom  2.7 1.4 30.2 1.4  6.9 16.9  5.7 28.3 6.4
1         Belgium  3.3 0.9 27.6 0.9  8.2 19.1  6.2 26.6 7.2
21     E. Germany  4.2 2.9 41.2 1.3  7.6 11.2  1.2 22.1 8.4
16         Sweden  6.1 0.4 25.9 0.8  7.2 14.4  6.0 32.4 6.8
8     Netherlands  6.3 0.1 22.5 1.0  9.9 18.0  6.8 28.5 6.8
4      W. Germany  6.7 1.3 35.8 0.9  7.3 14.4  5.0 22.3 6.1
7      Luxembourg  7.7 3.1 30.8 0.8  9.2 18.5  4.6 19.2 6.2
17    Switzerland  7.7 0.2 37.8 0.8  9.5 17.5  5.3 15.4 5.7
13         Norway  9.0 0.5 22.4 0.8  8.6 16.9  4.7 27.6 9.4
2         Denmark  9.2 0.1 21.8 0.6  8.3 14.6  6.5 32.2 7.1
3          France 10.8 0.8 27.5 0.9  8.9 16.8  6.0 22.6 5.7
10        Austria 12.7 1.1 30.2 1.4  9.0 16.8  4.9 16.8 7.0
11        Finland 13.0 0.4 25.9 1.3  7.4 14.7  5.5 24.3 7.6
6           Italy 15.9 0.6 27.6 0.5 10.0 18.1  1.6 20.1 5.7
20 Czechoslovakia 16.5 2.9 35.5 1.2  8.7  9.2  0.9 17.9 7.0
22        Hungary 21.7 3.1 29.6 1.9  8.2  9.4  0.9 17.2 8.0
15          Spain 22.9 0.8 28.5 0.7 11.5  9.7  8.5 11.8 5.5
5         Ireland 23.2 1.0 20.7 1.3  7.5 16.8  2.8 20.8 6.1
19       Bulgaria 23.6 1.9 32.3 0.6  7.9  8.0  0.7 18.2 6.7
25           USSR 23.7 1.4 25.8 0.6  9.2  6.1  0.5 23.6 9.3
14       Portugal 27.8 0.3 24.5 0.6  8.4 13.3  2.7 16.7 5.7
23         Poland 31.1 2.5 25.7 0.9  8.4  7.5  0.9 16.1 6.9
24        Rumania 34.7 2.1 30.1 0.6  8.7  5.9  1.3 11.7 5.0
12         Greece 41.4 0.6 17.6 0.6  8.1 11.5  2.4 11.0 6.7
26     Yugoslavia 48.7 1.5 16.8 1.1  4.9  6.4 11.3  5.3 4.0
18         Turkey 66.8 0.7  7.9 0.1  2.8  5.2  1.1 11.9 3.2
```

In the above left, we can see the countries ordered by their scores on RC1, and on the right, on their scores on Agr. Turkey has the highest score for RC1, which makes sense because RC1 is primarily positively Agr, and Turkey has the highest percentage of people employed in agriculture. Above, we can also see that E. Germany has the lowest score for RC1, which makes sense because it has one of the lowest scores for Agr. We can see how the other components of RC1 play a role, because if Agr were the only component of RC1, we would expect to see the United Kingdom have the lowest score on RC1.

```
> s[order(RC2),] #sort the dataframe rows
           Country         RC1        RC2
8      Netherlands -0.42357623 -1.5558836
2          Denmark -0.03091821 -1.5142897
16          Sweden -0.21102466 -1.1849486
1          Belgium -0.59832877 -1.0595183
13          Norway -0.61687550 -0.9623251
3           France -0.14231031 -0.8391757
17     Switzerland -0.37293535 -0.6468584
11         Finland -0.33866749 -0.6398245
6            Italy -0.02654030 -0.5526932
9   United Kingdom -0.73572258 -0.5045475
15           Spain  0.34631705 -0.3223641
14        Portugal  0.66489194 -0.3175746
5          Ireland  0.13816381 -0.2474394
4       W. Germany -0.42309897 -0.1465071
10         Austria -0.61702380 -0.1007688
12          Greece  1.12858168  0.1194751
26      Yugoslavia  2.14140737  0.1254186
7       Luxembourg -0.68667078  0.3095863
18          Turkey  3.39060427  0.3559428
25            USSR -0.28983489  0.8004834
19        Bulgaria  0.04934314  1.0742403
24         Rumania  0.68842513  1.3599235
23          Poland  0.15767562  1.3984567
21      E. Germany -1.47699154  1.5066273
20  Czechoslovakia -0.76938198  1.6318345
22         Hungary -0.94550866  1.9127303
```

```
> employ[order(Min), c(1, 3, 2, 4:10)] #just reordered the col
          Country Min  Agr  Man  PS  Con   SI  Fin  SPS  TC
2         Denmark 0.1  9.2 21.8 0.6  8.3 14.6  6.5 32.2 7.1
8     Netherlands 0.1  6.3 22.5 1.0  9.9 18.0  6.8 28.5 6.8
17    Switzerland 0.2  7.7 37.8 0.8  9.5 17.5  5.3 15.4 5.7
14       Portugal 0.3 27.8 24.5 0.6  8.4 13.3  2.7 16.7 5.7
11        Finland 0.4 13.0 25.9 1.3  7.4 14.7  5.5 24.3 7.6
16         Sweden 0.4  6.1 25.9 0.8  7.2 14.4  6.0 32.4 6.8
13         Norway 0.5  9.0 22.4 0.8  8.6 16.9  4.7 27.6 9.4
6           Italy 0.6 15.9 27.6 0.5 10.0 18.1  1.6 20.1 5.7
12         Greece 0.6 41.4 17.6 0.6  8.1 11.5  2.4 11.0 6.7
18         Turkey 0.7 66.8  7.9 0.1  2.8  5.2  1.1 11.9 3.2
3          France 0.8 10.8 27.5 0.9  8.9 16.8  6.0 22.6 5.7
15          Spain 0.8 22.9 28.5 0.7 11.5  9.7  8.5 11.8 5.5
1         Belgium 0.9  3.3 27.6 0.9  8.2 19.1  6.2 26.6 7.2
5         Ireland 1.0 23.2 20.7 1.3  7.5 16.8  2.8 20.8 6.1
10        Austria 1.1 12.7 30.2 1.4  9.0 16.8  4.9 16.8 7.0
4      W. Germany 1.3  6.7 35.8 0.9  7.3 14.4  5.0 22.3 6.1
9  United Kingdom 1.4  2.7 30.2 1.4  6.9 16.9  5.7 28.3 6.4
25           USSR 1.4 23.7 25.8 0.6  9.2  6.1  0.5 23.6 9.3
26     Yugoslavia 1.5 48.7 16.8 1.1  4.9  6.4 11.3  5.3 4.0
19       Bulgaria 1.9 23.6 32.3 0.6  7.9  8.0  0.7 18.2 6.7
24        Rumania 2.1 34.7 30.1 0.6  8.7  5.9  1.3 11.7 5.0
23         Poland 2.5 31.1 25.7 0.9  8.4  7.5  0.9 16.1 6.9
20 Czechoslovakia 2.9 16.5 35.5 1.2  8.7  9.2  0.9 17.9 7.0
21     E. Germany 2.9  4.2 41.2 1.3  7.6 11.2  1.2 22.1 8.4
7      Luxembourg 3.1  7.7 30.8 0.8  9.2 18.5  4.6 19.2 6.2
22        Hungary 3.1 21.7 29.6 1.9  8.2  9.4  0.9 17.2 8.0
```

In the above left, we can see the countries ordered by their scores on RC2, and on the right, on their scores on Min. We can also see that the Netherlands has the lowest score on RC2. RC2 is primarily positively Min, but also has strong contributions from other variables. On the left, we can see that the Netherlands has one of the lowest percentage of people in Min, which explains its score on RC2. We can also see that Hungary has the highest score on RC1, which makes sense because it also has the highest percentage of people working in Min.

d. Significance of Correlations:

```
# Run a correlation test to see which correlations are significant
library(psych)
options("scipen"=100, "digits"=5)
round(cor(employNumeric), 2) #gives you the correlation matrix; rounds it off to 2 decimals
MCorrTest = corr.test(employNumeric, adjust="none") #this tests correlations for
#statistical significance. gives you a set of P values. adjust="none" => makes it so that the
# correlation matrix is perfectly symmetric
MCorrTest #remember, P < 0.1 is significant because we were asked to use a 90% C.L.

M = MCorrTest$p #this shows you the p values without rounding
M #need to use the un-rounded version for the MTest below:

# Now, for each element, see if it is < .1 (or whatever significance) and set the entry to
# true = significant or false. keep in mind, we are running a massive number
#of T-tests here, and the chances of making a type I error are high, so it is better to be
# a bit more stringent and choose a high C.L.
MTest = ifelse(M < .1, T, F)
MTest

# Now lets see how many significant correlations there are for each variable.  We can do
# this by summing the columns of the matrix
colSums(MTest) - 1  # We have to subtract 1 for the diagonal elements (self-correlation)
#we can use this to see if there are any variables that are overcorrelated (correlated with a
#ton of other variables at a statistically significant level)
```

```
> MTest = ifelse(M < .1, T, F)
> MTest
      Agr   Min   Man    PS   Con    SI   Fin   SPS    TC
Agr  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
Min FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE
Man  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE
PS   TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE
Con  TRUE FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE
SI   TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE
Fin FALSE  TRUE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE
SPS  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE
TC   TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE
> colSums(MTest) - 1  # We have to subtract 1 for the diagonal elements (self-correlation)
Agr Min Man  PS Con  SI Fin SPS  TC 
  6   4   5   4   4   5   2   3   5 
```

We have set our cutoff for being overcorrelated at 75% correlation with other predictors, which is equivalent to being correlated with (9 * 0.75) ≈ 7 other variables. There are no variables in this dataset that pass this threshold, however Agr gets close with 6 other correlations. Let's try removing it from the analysis and see if this helps the RCs become more easily interpretable:

```
> p3 = psych::principal(employNumeric[, 2:9], rotate="varimax", nfactors=2, scores=TRUE)
> print(p3$loadings, cutoff=.4, sort=F)

Loadings:
    RC1    RC2   
Min        -0.834
Man  0.769       
PS   0.602       
Con  0.602       
SI   0.429  0.751
Fin         0.649
SPS  0.523  0.589
TC   0.781       

                 RC1   RC2
SS loadings    2.533 2.140
Proportion Var 0.317 0.267
Cumulative Var 0.317 0.584
```

For side-by-side comparison, below is the model with Agr included:

```
> p2 = psych::principal(employNumeric, rotate="varimax", nfactors=2, scores=TRUE)
> print(p2$loadings, cutoff=.4, sort=F) #the goal of varimax is to get the loadings to be either

Loadings:
    RC1    RC2   
Agr -0.905       
Min        -0.858
Man  0.778       
PS   0.572       
Con  0.600       
SI   0.514  0.706
Fin         0.673
SPS  0.587  0.532
TC   0.743       

                 RC1   RC2
SS loadings    3.356 2.261
Proportion Var 0.373 0.251
Cumulative Var 0.373 0.624
```

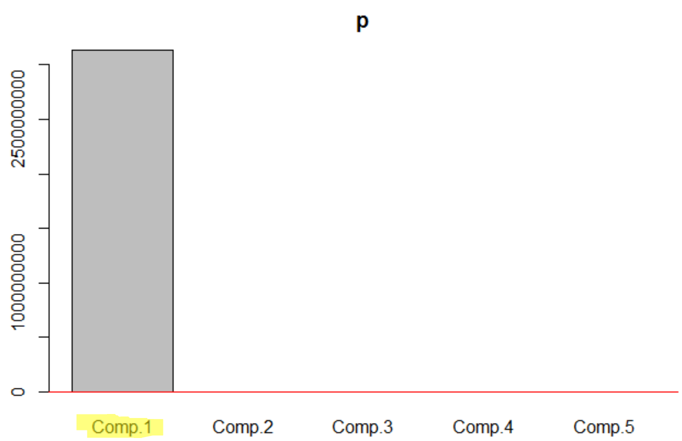
We can see that between models, the weights have shifted slightly, but not much. The signs have all remained the same, along with the variables that contribute to each component. The issue with overlapping variables remains the same.

Altogether, removing agriculture did not improve the model's interpretability, and resulted in a significant decrease in variance explained (from 0.624 to 0.584). Given this, and the fact that Agr is only collinear with 60% of the other variables (thus its variance may not be well accounted for in the rest of the predictors), I conclude that Agr would be better left in the model. However, if we were to have a variable that was truly overcorrelated, its removal from the model would likely result in a more parsimonious and interpretable model.

7. Census Data

a. PCA using the covariance matrix:

```
> p = princomp(census)
> summary(p)
Importance of components:
                       Comp.1          Comp.2         Comp.3          Comp.4           Comp.5
Standard deviation      55982 10.122848425033 6.167701460206 2.2282148623406 1.54698471591050
Proportion of Variance      1  0.000000032697 0.000000012138 0.0000000015842 0.00000000076361
Cumulative Proportion       1  0.999999985514 0.999999997652 0.9999999992364 1.00000000000000
```



From the summary above and the plot to the left, we can see that PC1 accounts for, essentially, 100% of the variance.

```
> p$loadings

Loadings:
              Comp.1 Comp.2 Comp.3 Comp.4 Comp.5
Population                         -0.483  0.872
Professional                        0.871  0.480
Employed             -0.513 -0.854
Government            0.855 -0.510
MedianHomeVal  1.000

               Comp.1 Comp.2 Comp.3 Comp.4 Comp.5
SS loadings       1.0    1.0    1.0    1.0    1.0
Proportion Var    0.2    0.2    0.2    0.2    0.2
Cumulative Var    0.2    0.4    0.6    0.8    1.0
```

From the loadings (left), we can see that PC1 is made up completely and positively by median home value.

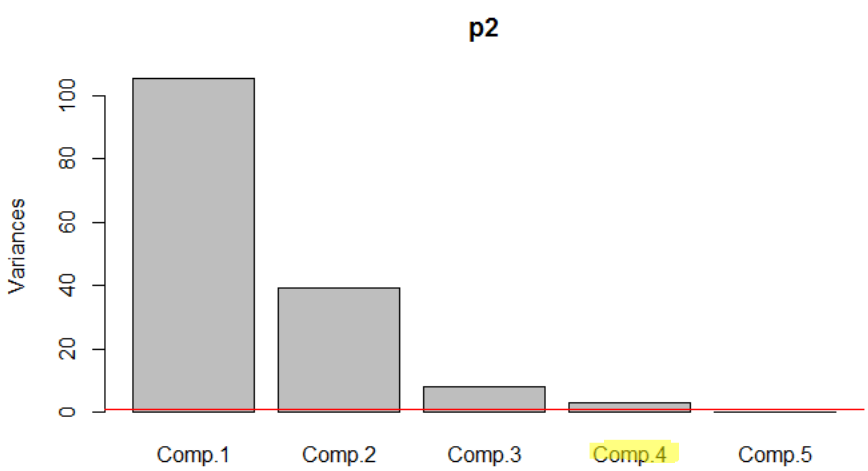
```
> census =read.csv("Census2.csv")
> head(census)
  Population Professional Employed Government MedianHomeVal
1       2.67         5.71    69.02       30.3        148000
2       2.25         4.37    72.98       43.3        144000
3       3.12        10.27    64.94       32.0        211000
4       5.14         7.44    71.29       24.5        185000
5       5.54         9.25    74.94       31.0        223000
6       5.04         4.84    53.61       48.2        160000
```

From the sample of the original data (left), we can see that median home value has scores which are significantly greater than scores for the other predictors. This is causing the relatively small amounts of variance seen in the other predictors to be drown out by the relatively enormous variance in median home value. In order to correct this, we need to standardize either the entire dataset, or just median home value. We also have the option of scaling median home value.

b. PCA with scaled median home value:

```
census =read.csv("Census2.csv")
head(census)

################ PCA:
p = princomp(census)
summary(p)
plot(p)
abline(1,0, col="red") #allows you to notice which PCs have a SD above 1 for the cut-off
p$loadings

#create a dataset with a scaled median home value:
census2 <- as.matrix(census)
smhv <- (census2[, 5] * 1/100000)
census2 = as.data.frame(census2)
census2$SMedianHomeVal <- smhv
census2 = census2[, c(1:4, 6)]
head(census2)

#PCA with scaled median home value:
p2 = princomp(census2)
summary(p2)
plot(p2)
abline(1,0, col="red") #allows you to notice which PCs have a SD above 1 for the cut-off
p2$loadings
```

```
> p2 = princomp(census2)
> summary(p2)
Importance of components:
                          Comp.1    Comp.2     Comp.3     Comp.4       Comp.5
Standard deviation     10.2596737 6.2467410 2.86943177 1.67954149 0.3900732431
Proportion of Variance  0.6769654 0.2509611 0.05295308 0.01814182 0.0009785696
Cumulative Proportion   0.6769654 0.9279265 0.98087961 0.99902143 1.0000000000
```



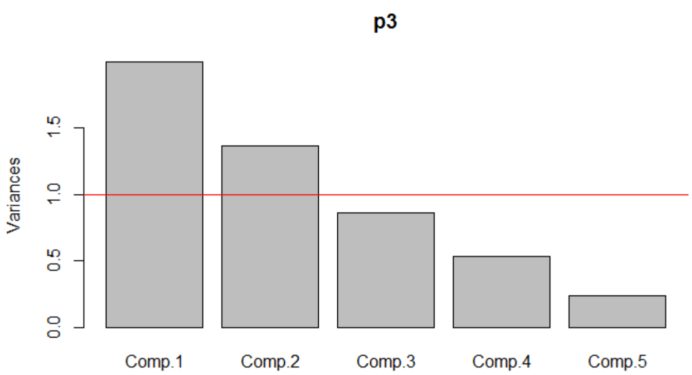

```
> p2$loadings

Loadings:
               Comp.1 Comp.2 Comp.3 Comp.4 Comp.5
Population                   -0.188  0.977       
Professional   -0.105 -0.130  0.961  0.171 -0.139
Employed        0.492 -0.864                     
Government     -0.863 -0.480 -0.153              
SMedianHomeVal                0.125         0.989

               Comp.1 Comp.2 Comp.3 Comp.4 Comp.5
SS loadings       1.0    1.0    1.0    1.0    1.0
Proportion Var    0.2    0.2    0.2    0.2    0.2
Cumulative Var    0.2    0.4    0.6    0.8    1.0
```

Scaling the median home value puts its values into a range that is comparable to the other predictors, which allows the variance in the other predictors to become relatively significant. We can see now that in order to capture 99% of the variance, we need to include 4 components. From the loadings, we can also see that PC1 is now primarily, negatively composed of government (loading = -0.836), and that median home value isn't a big contributor unless you include PC5 (loading = 0.989), which doesn't add much in terms of variance explained.

c. PCA with correlation matrix (standardized values):

```
#PCA with correlation matrix (standardized values):
p3 = princomp(census, cor=T)
summary(p3)
plot(p3)
abline(1,0, col="red")
p3$loadings
```

```
> p3 = princomp(census, cor=T)
> summary(p3)
Importance of components:
                          Comp.1    Comp.2    Comp.3    Comp.4     Comp.5
Standard deviation     1.4113534 1.1694129 0.9296006 0.7314787 0.49126036
Proportion of Variance 0.3983837 0.2735053 0.1728315 0.1070122 0.04826735
Cumulative Proportion  0.3983837 0.6718890 0.8447204 0.9517327 1.00000000
```



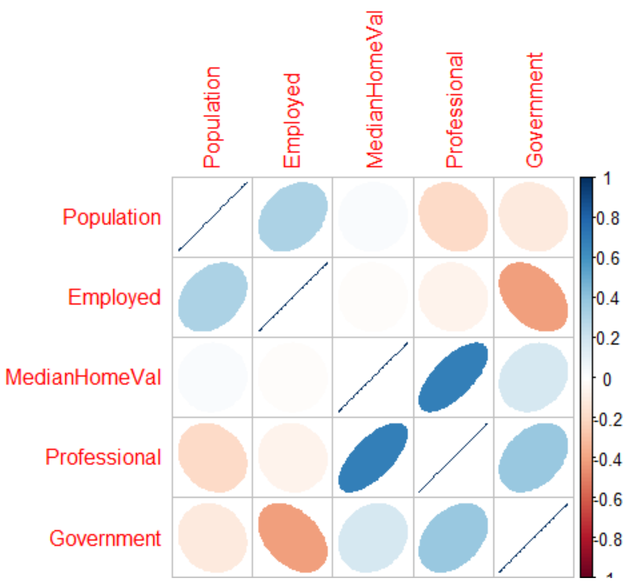

```
> p3$loadings

Loadings:
             Comp.1 Comp.2 Comp.3 Comp.4 Comp.5
Population    0.263  0.463  0.784  0.217  0.235
Professional -0.593  0.326 -0.164 -0.145  0.703
Employed      0.326  0.605 -0.225 -0.663 -0.194
Government   -0.479 -0.252  0.551 -0.572 -0.277
MedianHomeVal -0.493  0.500         0.407 -0.580

               Comp.1 Comp.2 Comp.3 Comp.4 Comp.5
SS loadings       1.0    1.0    1.0    1.0    1.0
Proportion Var    0.2    0.2    0.2    0.2    0.2
Cumulative Var    0.2    0.4    0.6    0.8    1.0
```

Standardizing the entire dataset allows us to capture only 95% of the variance in the first for PCs, as opposed 99% when we only scaled median home value. The scree plot shows that only the first two variables have SDs above 1, and the “knee” of the plot agrees with this. However, only including 2 PCs significantly reduces the variance captured to 67%.

Compared to simply scaling median home value, standardizing the entire dataset changed the weightings of the predictors’ contributions significantly. The latter tells a different story about what in the data explains the most variance; we can interpret the loadings to say that PC1 is largely a mix of all the predictors, however it is largely composed negatively of Professional (loading = -0.593). This contrasts starkly with the above model, which insists that Professional only played a small role (loading = -0.105) in PC1, while Government played the largest role (loading = -0.863). Between these two options, there isn’t much difference in terms of interpretability; they are both difficult to interpret. We can try rotation the components in order to fix this. However, the only important factor in deciding whether to only scale median home value or standardize the entire dataset is whether one way or the other makes more sense based upon the data. This is where a domain expert comes in to tell you whether the differences in the variances between the non-standardized predictors are inherent (important) or whether standardizing all of the data is wise. For the purposes of demonstration, you can see in the analysis above that the implications of such a decision are large, as the results are quite different.

d. Overcorrelated predictors:

```
# Run a correlation test to see which correlations are significant
library(psych)
options("scipen"=100, "digits"=5)
round(cor(census), 2) #gives you the correlation matrix; rounds it off to 2 decimals
MCorrTest = corr.test(census, adjust="none") #this tests correlations for
#statistical significance. gives you a set of P values. adjust="none" => makes it so that the
# correlation matrix is perfectly symmetric
MCorrTest #remember, P < 0.05 is significant because we were asked to use a 95% C.L.

M = MCorrTest$p #this shows you the p values without rounding
M #need to use the un-rounded version for the MTest below:

# Now, for each element, see if it is < .05 (or whatever significance) and set the entry to
# true = significant or false. keep in mind, we are running a massive number
#of T-tests here, and the chances of making a type I error are high, so it is better to be
# a bit more stringent and choose a high C.L.
MTest = ifelse(M < .05, T, F)
MTest

# Now lets see how many significant correlations there are for each variable.  We can do
# this by summing the columns of the matrix
colSums(MTest) - 1  # We have to subtract 1 for the diagonal elements (self-correlation)
#we can use this to see if there are any variables that are overcorrelated (correlated with a
#ton of other variables at a statistically significant level)
```




```
> MCorrTest #remember, P < 0.05 is significant because we were asked to use a 95% C.L.
Call:corr.test(x = census, adjust = "none")
Correlation matrix
              Population Professional Employed Government MedianHomeVal
Population          1.00        -0.19     0.31      -0.12          0.03
Professional       -0.19         1.00    -0.07       0.37          0.69
Employed            0.31        -0.07     1.00      -0.41         -0.01
Government         -0.12         0.37    -0.41       1.00          0.18
MedianHomeVal       0.03         0.69    -0.01       0.18          1.00
Sample Size
[1] 61
Probability values (Entries above the diagonal are adjusted for multiple tests.)
              Population Professional Employed Government MedianHomeVal
Population          0.00         0.14     0.01       0.36          0.84
Professional        0.14         0.00     0.62       0.00          0.00
Employed            0.01         0.62     0.00       0.00          0.94
Government          0.36         0.00     0.00       0.00          0.17
MedianHomeVal       0.84         0.00     0.94       0.17          0.00
```

Above, we can see that there aren't any extreme multicollinearity issues. There is a moderate correlation between Professional and median home value, but all other correlations are weak.

```
> MTest = ifelse(M < .05, T, F)
> MTest
                Population Professional Employed Government MedianHomeVal
Population            TRUE        FALSE     TRUE      FALSE         FALSE
Professional         FALSE         TRUE    FALSE       TRUE          TRUE
Employed              TRUE        FALSE     TRUE       TRUE         FALSE
Government           FALSE         TRUE     TRUE       TRUE         FALSE
MedianHomeVal        FALSE         TRUE    FALSE      FALSE          TRUE
> colSums(MTest) - 1  # We have to subtract 1 for the diagonal elements (self-correlation)
   Population  Professional      Employed    Government MedianHomeVal 
            1             2             2             2             1 
```

Above, we can see which correlations are statistically significant at the 95% confidence level. Note that the correlation between Professional and median home value is highly significant. With 5 variables, the most any one variable could be correlated to is 4 other variables. Given this, the fact that 2 is the highest number of correlations any variable has, and also the fact that the correlations themselves are weak or moderate, one might deduce that there isn't strong evidence of overcorrelation. However, there could be an argument for overcorrelation if you consider the fact that 3/5 variables are correlated to 2 other variables at a significant level. If there is any negative effect from this, it is likely to cause the variable contributions to the PCs to overlap, which in turn makes them difficult to interpret. We could test this by removing a variable, such as Professional, and checking to see how much this affects overlapping of contributions to PCs as well as the effect on variance captured.

e. Meaning of Correlation Matrix

The point of PCA is to detect meaningful covariances among the variables, and thus illuminate the latent variable(s) that they all help to describe. The correlation matrix attempts to detect these same relationships, but it standardizes the data. When standardizing, such underlying relationships will still be detected, unless they are very weak and highly dependent on the sizes of the original variances. However, this is not usually the case, and thus scaling is extremely common in factor analysis.

Within this dataset, several of the variables are recorded in different units. The problem with data being in different units is that they lead to different ranges for the variables, which in turn gives certain variables more or less weight/effect. The correlation matrix standardizes all of the values by dividing each vector (variable column) by its standard deviation, which essentially makes the vectors unit-less. This forces all the data to fall in a common range, which theoretically gives them an equal weight. In rare cases, this also has the power to take away any differences in variance that may or may not be inherent (permanent, essential, characteristic) in the data, and thus may or may not be important, so it must be used with caution. In a case where the variance is inherent, the question becomes how do we detect and preserve this while also evening out any other differences that are only due to variable size (e.g. large values vs. small values).

The only important factor in deciding whether or not to standardize is whether one way or the other makes more sense based upon your specific data. This is where domain expertise comes in to decide whether the differences in the variances between the non-standardized predictors are inherent (important) or whether standardizing all of the data is wise because there might be small-scale, important effects that would otherwise be drown out.

For example, given that three of the variables in this dataset are in the same unit (percent), any differences in their variation is inherent and should be preserved if possible (don't scale). On the other hand, having another variable, in different units and in the hundreds of thousands, creates a range of variation that definitely drowns out the variation seen in the smaller-valued variables (do scale).

For the purposes of demonstration, you can see in the analysis above that the implications of such a decision are large, as the results are quite different. This emphasizes the importance of knowing your data and standardizing only when appropriate. Scaling also has implications for prediction with new data, as any new data will have to be standardized with the same means and SDs that the model was built on. When interpreting the regression formula, the meaning and scale of the PCs, as well as how they affect the response variable, must be considered in order to understand the model and how changing certain aspects can improve outcomes for your application. For example, it is important to understand how increasing or decreasing the PC contributors (original variables that you can measure) will affect the PC (the latent variable), and by extension, the response variable (the outcome). In this way, you can make the model work for you.

8. Track Record Data

a. PCA

The original data comes in different units (seconds and minutes), and also in widely different ranges (tens vs. hundreds), as we can see below:

```
> head(track)
    Country  m100  m200  m400 m800 m1500 m5000 m10000 Marathon
1 Argentina 10.23 20.37 46.18 1.77  3.68 13.33  27.65   129.57
2 Australia  9.93 20.06 44.38 1.74  3.53 12.93  27.53   127.51
3   Austria 10.15 20.45 45.80 1.77  3.58 13.26  27.72   132.22
4   Belgium 10.14 20.19 45.02 1.73  3.57 12.83  26.87   127.20
5   Bermuda 10.27 20.30 45.26 1.79  3.70 14.64  30.49   146.37
6    Brazil 10.00 19.89 44.29 1.70  3.57 13.48  28.13   126.05
```

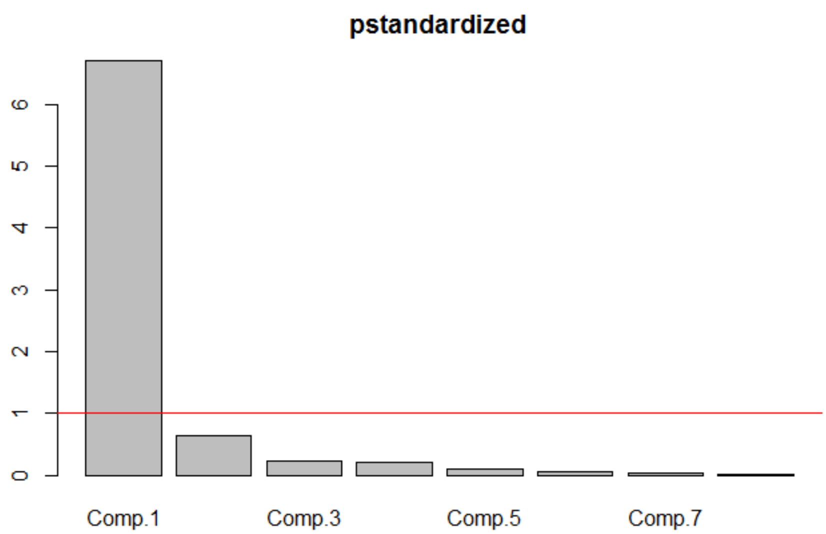
Using the data as is will surely lead to Marathon describing the majority of the data, since its values are the largest. Converting all of the data to the same units (seconds) intensifies this problem (tens vs. thousands):

```
> head(trackseconds)
    Country  m100  m200  m400  m800 m1500 m5000 m10000 Marathon
1 Argentina 10.23 20.37 46.18 106.2 220.8 799.8 1659.0   7774.2
2 Australia  9.93 20.06 44.38 104.4 211.8 775.8 1651.8   7650.6
3   Austria 10.15 20.45 45.80 106.2 214.8 795.6 1663.2   7933.2
4   Belgium 10.14 20.19 45.02 103.8 214.2 769.8 1612.2   7632.0
5   Bermuda 10.27 20.30 45.26 107.4 222.0 878.4 1829.4   8782.2
6    Brazil 10.00 19.89 44.29 102.0 214.2 808.8 1687.8   7563.0
```

This is a situation where standardization comes in handy, as long as none of the covariances are very weak and highly dependent on the sizes of the original variances. Given that this is not common, and lacking domain expertise, we will move forward with standardized vectors:

Renel Chesak
CSC 424

```
> library(psych)
> pstandardized = princomp(tracknumeric, cor=T)
> summary(pstandardized)
Importance of components:
                          Comp.1    Comp.2    Comp.3    Comp.4    Comp.5    Comp.6     Comp.7     Comp.8
Standard deviation     2.58907 0.799006 0.476995 0.453706 0.312374 0.265872 0.2166611 0.0985843
Proportion of Variance 0.83791 0.079801 0.028441 0.025731 0.012197 0.008836 0.0058678 0.0012149
Cumulative Proportion  0.83791 0.917713 0.946153 0.971884 0.984081 0.992917 0.9987851 1.0000000
> plot(pstandardized)
> abline(1,0, col="red")
> pstandardized$loadings

Loadings:
         Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8
m100     -0.332 -0.529  0.344  0.381 -0.300 -0.362  0.348
m200     -0.346 -0.470         0.217  0.541  0.349 -0.440
m400     -0.339 -0.345        -0.851 -0.133         0.114
m800     -0.353        -0.783  0.134  0.227 -0.341  0.259
m1500    -0.366  0.154 -0.244  0.233 -0.652  0.530 -0.147
m5000    -0.370  0.295  0.183               -0.359 -0.328  0.706
m10000   -0.366  0.334  0.244               -0.273 -0.351 -0.697
Marathon -0.354  0.387  0.335         0.338  0.375  0.594

               Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8
SS loadings     1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000
Proportion Var  0.125  0.125  0.125  0.125  0.125  0.125  0.125  0.125
Cumulative Var  0.125  0.250  0.375  0.500  0.625  0.750  0.875  1.000
```



Above, we can see that about **92%** of the variance is captured in the first 2 PCs. The plot on the left shows that the vast majority of the variance is captured by PC1. The loadings (above) indicate a large amount of overlap, and **extremely even contributions to PC1**. This is not helpful for interpretation, so we will attempt to rotate the components to clear things up.

```
> pstandardizedROTATED1 = psych::principal(tracknumeric, covar=FALSE, r
otate="varimax", nfactors=1,
+                                          scores=TRUE)
> print(pstandardizedROTATED1$loadings, cutoff=.4, sort=T) #the goal of
 varimax is to get the loadings to be either

Loadings:
[1] 0.861 0.896 0.878 0.914 0.948 0.957 0.947 0.917

                 PC1
SS loadings    6.703
Proportion Var 0.838
```

With 1 rotated PC, we capture 84% of the variance, yet there are 8 variable contributions that are still very even. There is not much improvement here from the last model.

Renel Chesak
CSC 424

```
> pstandardizedROTATED2 = psych::principal(tracknumeric, covar=FALSE, r
otate="varimax", nfactors=2,
+                                           scores=TRUE)
> print(pstandardizedROTATED2$loadings, cutoff=.4, sort=T) #the goal of
 varimax is to get the loadings to be either

Loadings:
         RC1   RC2  
m800     0.735 0.547
m1500    0.794 0.531
m5000    0.876 0.453
m10000   0.889 0.423
Marathon 0.894      
m100           0.885
m200     0.427 0.872
m400     0.480 0.785

                 RC1   RC2
SS loadings    4.078 3.263
Proportion Var 0.510 0.408
Cumulative Var 0.510 0.918
```

With 2 rotated PCs, there is a little improvement in that the largest contributors to RC1 are narrowed down to 5 variables (loadings > 0.7). There is a lot of overlap in contributions between RC1 and RC2.

```
> pstandardizedROTATED3 = psych::principal(tracknumeric, covar=FALSE, r
otate="varimax", nfactors=3,
+                                           scores=TRUE)
> print(pstandardizedROTATED3$loadings, cutoff=.4, sort=T) #the goal of
 varimax is to get the loadings to be either

Loadings:
         RC1   RC2   RC3  
m1500    0.682 0.468 0.492
m5000    0.833 0.430      
m10000   0.856 0.406      
Marathon 0.878            
m100           0.886      
m200           0.839      
m400     0.407 0.746      
m800     0.536 0.435 0.709

                 RC1   RC2   RC3
SS loadings    3.387 2.934 1.248
Proportion Var 0.423 0.367 0.156
Cumulative Var 0.423 0.790 0.946
```

With 3 rotated PCs, there is not much improvement, and it appears that the major difference is that m1500 and m800 split off a bit and contributed to RC3. However, now there are only 3 variables with loadings > 0.7 for RC1 and RC2.

```
> pstandardizedROTATED4 = psych::principal(tracknumeric, covar=FALSE, r
otate="varimax", nfactors=4,
+                                           scores=TRUE)
> print(pstandardizedROTATED4$loadings, cutoff=.4, sort=T) #the goal of
 varimax is to get the loadings to be either

Loadings:
         RC1   RC2   RC3   RC4  
m1500    0.680 0.435 0.499      
m5000    0.829                  
m10000   0.852                  
Marathon 0.875                  
m100           0.873            
m200           0.788            
m800     0.533       0.702      
m400           0.521       0.709

                 RC1   RC2   RC3   RC4
SS loadings    3.343 2.334 1.175 0.923
Proportion Var 0.418 0.292 0.147 0.115
Cumulative Var 0.418 0.710 0.857 0.972
```

With 4 rotated PCs, there is not much improvement.

Renel Chesak
CSC 424

```
> pstandardizedROTATED5 = psych::principal(tracknumeric, covar=FALSE, r
otate="varimax", nfactors=5,
+                                              scores=TRUE)
> print(pstandardizedROTATED5$loadings, cutoff=.4, sort=T) #the goal of
 varimax is to get the loadings to be either

Loadings:
         RC1    RC2    RC3    RC4    RC5
m1500     0.669  0.422  0.481
m5000     0.827
m10000    0.853
Marathon  0.880
m100             0.865
m200             0.791
m800      0.537         0.705
m400             0.513         0.717

                 RC1   RC2   RC3   RC4   RC5
SS loadings    3.342 2.298 1.163 0.950 0.119
Proportion Var 0.418 0.287 0.145 0.119 0.015
Cumulative Var 0.418 0.705 0.850 0.969 0.984
```

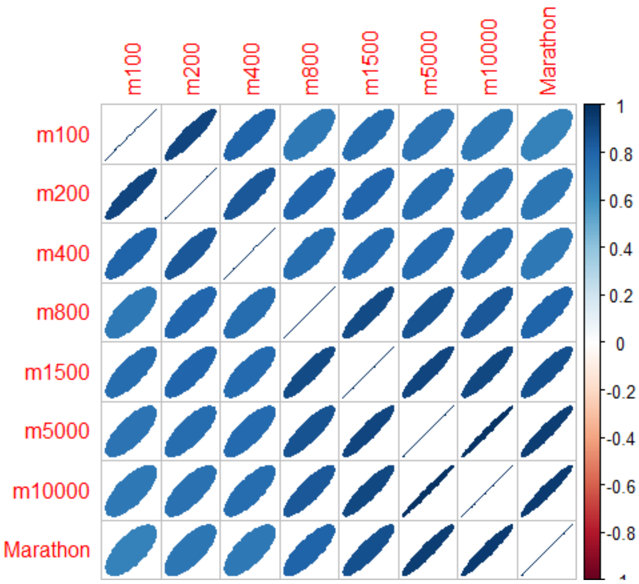
Including a 5th rotated PC does not give us any significant improvement. You can see that there aren't even any contributions to RC5.

Final model selected by PCA Factor Analysis (with all 8 variables):

```
> pstandardizedROTATED3b = psych::principal(tracknumeric, covar=FALSE, rotat
e="varimax", nfactors=3,
+                                              scores=TRUE)
> print(pstandardizedROTATED3b$loadings, cutoff=.7, sort=T) #use a cutoff of
 0.7 instead

Loadings:
         RC1   RC2   RC3
m1500
m5000    0.833
m10000   0.856
Marathon 0.878
m100           0.886
m200           0.839
m400           0.746
m800                 0.709

                 RC1   RC2   RC3
SS loadings    3.387 2.934 1.248
Proportion Var 0.423 0.367 0.156
Cumulative Var 0.423 0.790 0.946
```

Based upon these models, it appears that the best separation of contributions is achieved using a loadings cutoff of 0.7. Including 3 rotated PCs with a loadings cutoff of 0.7 achieves a balance that affords a high percentage of variance explained (95%).

Let's look at the correlation matrix to check for overcorrelation:



We can see that there is a high degree of strong multicollinearity among all of the variables. This is likely causing the multiple contributions to our PCs as well as the high degree of overlap between PCs.

```
> library(psych)
> options("scipen"=100, "digits"=5)
> round(cor(tracknumeric), 2) #gives you the correlation
         m100 m200 m400 m800 m1500 m5000 m10000 Marathon
m100     1.00 0.91 0.80 0.71  0.77  0.74   0.71     0.68
m200     0.91 1.00 0.84 0.80  0.80  0.76   0.75     0.72
m400     0.80 0.84 1.00 0.77  0.77  0.78   0.77     0.71
m800     0.71 0.80 0.77 1.00  0.90  0.86   0.84     0.81
m1500    0.77 0.80 0.77 0.90  1.00  0.92   0.90     0.88
m5000    0.74 0.76 0.78 0.86  0.92  1.00   0.99     0.94
m10000   0.71 0.75 0.77 0.84  0.90  0.99   1.00     0.95
Marathon 0.68 0.72 0.71 0.81  0.88  0.94   0.95     1.00
```

Above I have flagged correlations > 0.85 as having the most extreme correlations, though all correlations in this table are strong.

```
> MTest = ifelse(M < .01, T, F)
> MTest
         m100 m200 m400 m800 m1500 m5000 m10000 Marathon
m100     TRUE TRUE TRUE TRUE  TRUE  TRUE   TRUE     TRUE
m200     TRUE TRUE TRUE TRUE  TRUE  TRUE   TRUE     TRUE
m400     TRUE TRUE TRUE TRUE  TRUE  TRUE   TRUE     TRUE
m800     TRUE TRUE TRUE TRUE  TRUE  TRUE   TRUE     TRUE
m1500    TRUE TRUE TRUE TRUE  TRUE  TRUE   TRUE     TRUE
m5000    TRUE TRUE TRUE TRUE  TRUE  TRUE   TRUE     TRUE
m10000   TRUE TRUE TRUE TRUE  TRUE  TRUE   TRUE     TRUE
Marathon TRUE TRUE TRUE TRUE  TRUE  TRUE   TRUE     TRUE
> colSums(MTest) - 1  # We have to subtract 1 for the diagonal elements
    m100     m200     m400     m800    m1500    m5000   m10000 Marathon
       7        7        7        7        7        7        7        7
```

On the left we can see that all the variables in this dataset are correlated with each other at a statistically significant level (P < 0.01). This strongly indicates that several variables can be removed from this dataset without losing much variance.

After removing m200, m5000, m10000, and Marathon from the dataset, we get the following array of models:

```
> pstandardizedROTATEDa = psych::principal(track2, covar=FALSE, rotat
e="varimax", nfactors=1,
+                                         scores=TRUE)
> print(pstandardizedROTATEDa$loadings, cutoff=.7, sort=T)

Loadings:
[1] 0.893 0.911 0.922 0.938

                 PC1
SS loadings    3.359
Proportion Var 0.840
```

Given that there are now only 4 original variables in this model, it would be wise to choose a PCA model with 1-2 PCs, otherwise we haven't reduced dimensionality enough.

```
> pstandardizedROTATEDb = psych::principal(track2, covar=FALSE, rotat
e="varimax", nfactors=2,
+                                         scores=TRUE)
> print(pstandardizedROTATEDb$loadings, cutoff=.7, sort=T)

Loadings:
      RC1   RC2
m800  0.885
m1500 0.838
m100        0.883
m400        0.803

                 RC1   RC2
SS loadings    1.868 1.838
Proportion Var 0.467 0.459
Cumulative Var 0.467 0.926
```

Including only 2 rotated PCs allows us to explain 92.6% of the variance, with RC1 primarily, equally, and positively composed of m800 and m1500. RC2 is primarily, equally, and positively composed of m100 and m400. This **model** gives us a high degree of parsimony and a high percentage of variance accounted for, and thus is preferable to the model chosen by PCA factor analysis with all 8 variables.

```
> pstandardizedROTATEDc = psych::principal(track2, covar=FALSE, rotat
e="varimax", nfactors=3,
+                                         scores=TRUE)
> print(pstandardizedROTATEDc$loadings, cutoff=.7, sort=T)

Loadings:
      RC1   RC2   RC3
m800  0.863
m1500 0.828
m100        0.845
m400              0.808

                 RC1   RC2   RC3
SS loadings    1.751 1.133 1.019
Proportion Var 0.438 0.283 0.255
Cumulative Var 0.438 0.721 0.976
```

b. Common Factor Analysis:

Note: though there are high degrees of overcorrelation in the original dataset, we will keep all original variables because common factor analysis requires a particular ratio of original variables to PCs. Failure to stay within the bounds of this ratio results in an error:

```
> cfa2 = factanal(track2, 2) #using 2 components
Error in factanal(track2, 2) : 2 factors are too many for 4 variables
```

Due to this limitation, comparisons between common factor analysis and PCA factor analysis will be made only between models that were built using the entire dataset, with all 8 variables.

```
> cfa2 = factanal(tracknumeric, 2) #using
> print(cfa2$loadings, cutoff=.7, sort=T)
ferently

Loadings:
         Factor1 Factor2
m800
m1500     0.745
m5000     0.883
m10000    0.897
Marathon 0.863
m100             0.841
m200             0.894
m400             0.714

               Factor1 Factor2
SS loadings      3.912   3.231
Proportion Var   0.489   0.404
Cumulative Var   0.489   0.893
> cfa3 = factanal(tracknumeric, 3)
> print(cfa3$loadings, cutoff=.7, sort=T)

Loadings:
         Factor1 Factor2 Factor3
m1500
m5000     0.842
m10000    0.870
Marathon 0.837
m100             0.866
m200             0.829
m400
m800                     0.715

               Factor1 Factor2 Factor3
SS loadings      3.403   2.816   1.179
Proportion Var   0.425   0.352   0.147
Cumulative Var   0.425   0.777   0.925
> cfa4 = factanal(tracknumeric, 4)
> print(cfa4$loadings, cutoff=.7, sort=T)

Loadings:
         Factor1 Factor2 Factor3 Factor4
m1500
m5000     0.843
m10000    0.871
Marathon  0.837
m100              0.849
m200              0.867
m400
m800                      0.715

               Factor1 Factor2 Factor3 Factor4
SS loadings      3.405   2.838   1.177   0.045
Proportion Var   0.426   0.355   0.147   0.006
Cumulative Var   0.426   0.780   0.927   0.933
> cfa5 = factanal(tracknumeric, 5)
Error in factanal(tracknumeric, 5) :
  5 factors are too many for 8 variables
```

In the array of models to the left, we can see that common factor analysis (CFA) has given us results that are very similar to PCA factor analysis (with all 8 variables). The small differences that I notice include:

1. There are slightly fewer factor contributions for some PCs in common factor analysis (e.g. with 3 PCs, PCA has 3 contributors to RC2 while CFA has 2).
2. The variance explained is slightly lower for the CFA models (e.g. with 3 PCs, the PCA model accounts for 94.6% of the variance, while the CFA model accounts for 92.5%).

Final thoughts:

Because common factor analysis imposes a specific ratio of original variables to PCs, it also limits our ability to remove highly overcorrelated variables from the model. This, in turn, leads to PCs that are made of multiple variable contributors, unnecessarily. Due to this limitation and subsequent lack of parsimony, it appears that PCA factor analysis produced the best model (with only 4 original variables and 2 PCs).