Exercise 1: Here are some data where the observations are divided into an *x* group and a *y* group.

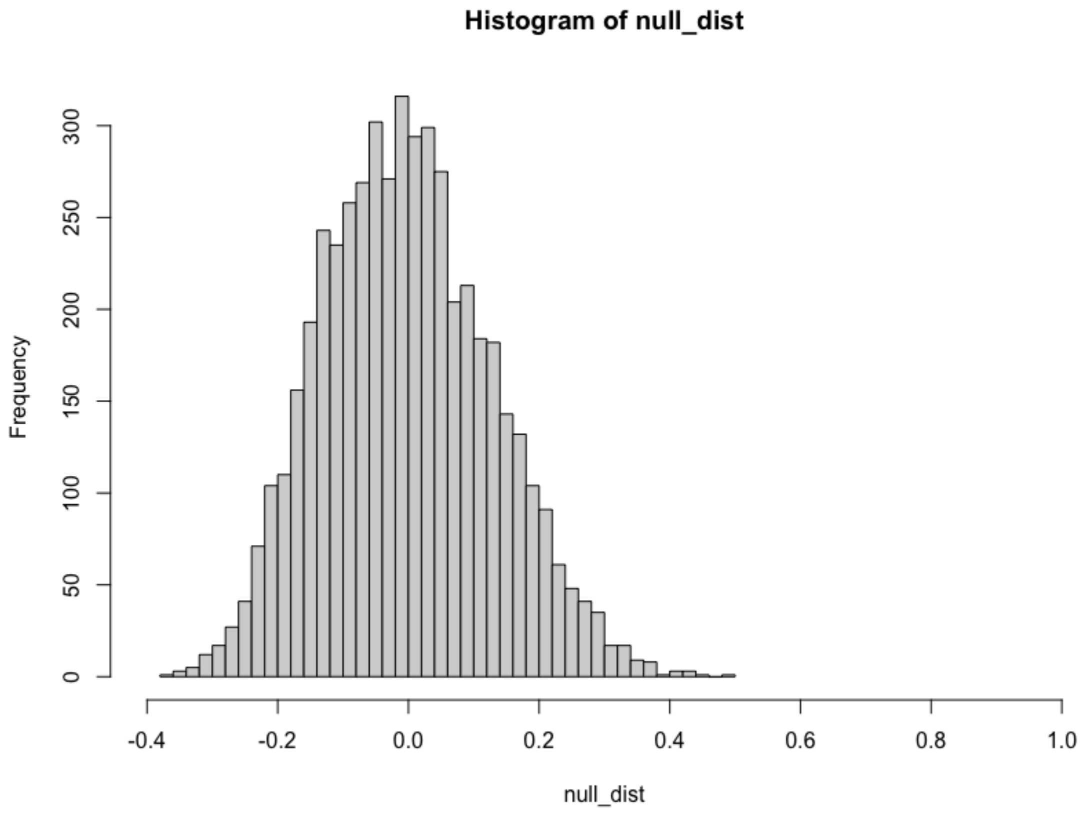
```
data_mat <-structure(c(4, 5, 9, 10, 6, 4, 13, 4, 16, 14, 9, 2, 7, 6,
9, 16, 7, 7, 7, 9, 1, 15, 11, 20, 8, 5, 19, 16, 17, 18, 18, 18, 17,
20, 15, 29, 16, 14, 17, 17, 25, 17, 1, 12, 33, 26, 16, 3, 25, 7, 2,
30, 16, 24, 6, 3, 23, 24, 24, 15, 2.02, 3.54, 2.27, 4.97, 4.2, 3.15,
4.07, 3.3, 4.2, 4.55, 5.3, 3.86, 4.44, 5.65, 3.23, 6.4, 3.61, 4.63,
4.34, 3.58, 2.13, 5.1, 4.43, 5, 2.46, 2.77, 3.67, 5.67, 5.26, 5.03,
3.25, 5.21, 2.85, 3.87, 4.57, 4.79, 4.17, 4.91, 3.5, 3.42, 3.86, 3.45,
4.73, 2.77, 2.76, 2.67, 4.09, 3.46, 4.03, 3.23, 2.64, 5.92, 4.54, 3.55,
3.43, 4.2, 3.88, 3.59, 3.48, 5.66, 2.48, 6.57, 7.52, 11.47, 6.26, 4.83,
7.6, 9.54, 11.35, 9.96, 10, 7.4, 5.58, 8.91, 8.64, 10.77, 6.04, 9.28, 8.47,
5.72), .Dim = c(20L, 7L), .Dimnames = list(NULL, c("y1", "y2", "y3", "x1", "x2", "x3", "x4")))
```

a. Using R, run the Mantel test to see if the distance structure in y1, y2, y3 (count data) is the same as the distance structure for the x1, x2, x3, x4 data.

Solution:
Before we can compute a Mantel test, we must first obtain a distance measure for each group. In the examples that were shown in class we standardized each variable then computed the euclidean distance between each observation (We will also test the count data with the Bray-Curtis measure using the vegan package). Proceeding similarly we get a p-value of 0.0002, so at the $\alpha = .05$ level we would reject the null and conclude that the distance measures for each group are correlated, ie they cluster in a similar way. Plotting the null distribution, and double checking our mantel test with the mantel() function from the vegan package we get the same conclusion,
Code:

```
## Scaling each variable to mean = 0 sd = 1
data_mat_scale <- scale(data_mat)

## Computing the distance matrices for each group.
Y <- dist(data_mat_scale[,1:3], method="euclidean", diag=TRUE, upper=TRUE)
X <- dist(data_mat_scale[,4:7], method="euclidean", diag=TRUE, upper=TRUE)

## Computing the test statistic for the our given data.
T <- cor(cbind(as.vector(Y), as.vector(X)))[1,2]
    [1] 0.468862



## Computing the mantel test, generating approx.
## null dist. histogram and computing test statistic.
Nsim <- 5000
null_dist <- rep(NA,Nsim)
Y <- data_mat[,1:3]


for(i in 1:Nsim){
  #Finding a permutation of the 20 observations in dat_mat_scale
  ind <- sample(1:20, size=20, replace=FALSE)
  #Applying permutation to Y
  Y_rand <- Y[ind,]
  #Computing permuted distance
  Y_rand_dist <- dist(Y_rand, method="", diag=TRUE, upper=TRUE)
  #Computing test statistic with permuted Y dist matrix
  null_dist[i] = cor(cbind(as.vector(X), as.vector(Y_rand_dist)))[1,2]
}

hist(null_dist, n=40, xlim=c(-0.4,1.0)) # Plotting null dist.
## Computing P-value
p_value <- mean((null_dist< -abs(T))|(null_dist> abs(T)))
    [1] 2e-04



## Double checking with mantel test in Vegan library
## Also using the Bray-Curtis distance for the count data
## instead.
load(vegan)
Ymantel <- vegdist(data_mat[,1:3], method = 'bray')
Xmantel <- vegdist(data_mat_scale[,4:7], method = 'euclidean')

mantel(Xmantel, Ymantel, method = 'pearson', permutations = 5000)
Mantel statistic based on Pearson's product-moment correlation

Call:
mantel(xdis = Xmantel, ydis = Ymantel, method = "pearson", permutations = 5000)
Mantel statistic r: 0.4251
      Significance: 0.0009998
Upper quantiles of permutations (null model):
  90%   95% 97.5%   99%
0.164 0.210 0.254 0.303
Permutation: free
Number of permutations: 5000
```

Figure 1: Distribution of test statistic under the null.



b. What are your conclusions? what does the Mantel test actually test? What is the null hypothesis?

Solution:
Like we previously stated, we conclude that the groups have correlated distance measures we can also sat that the groups cluster in a similar way. The null of the Mantel test assumes that the distance measures are uncorrelated, the alternative states that they are correlated. When we do a randomization test, we sample the distribution of the test statistic under the assumption of the null hypothesis since distances should be uncorrelated when the order of observations in one of the groups is randomized. With that sample distribution and our original test statistic we can compute a p-value.

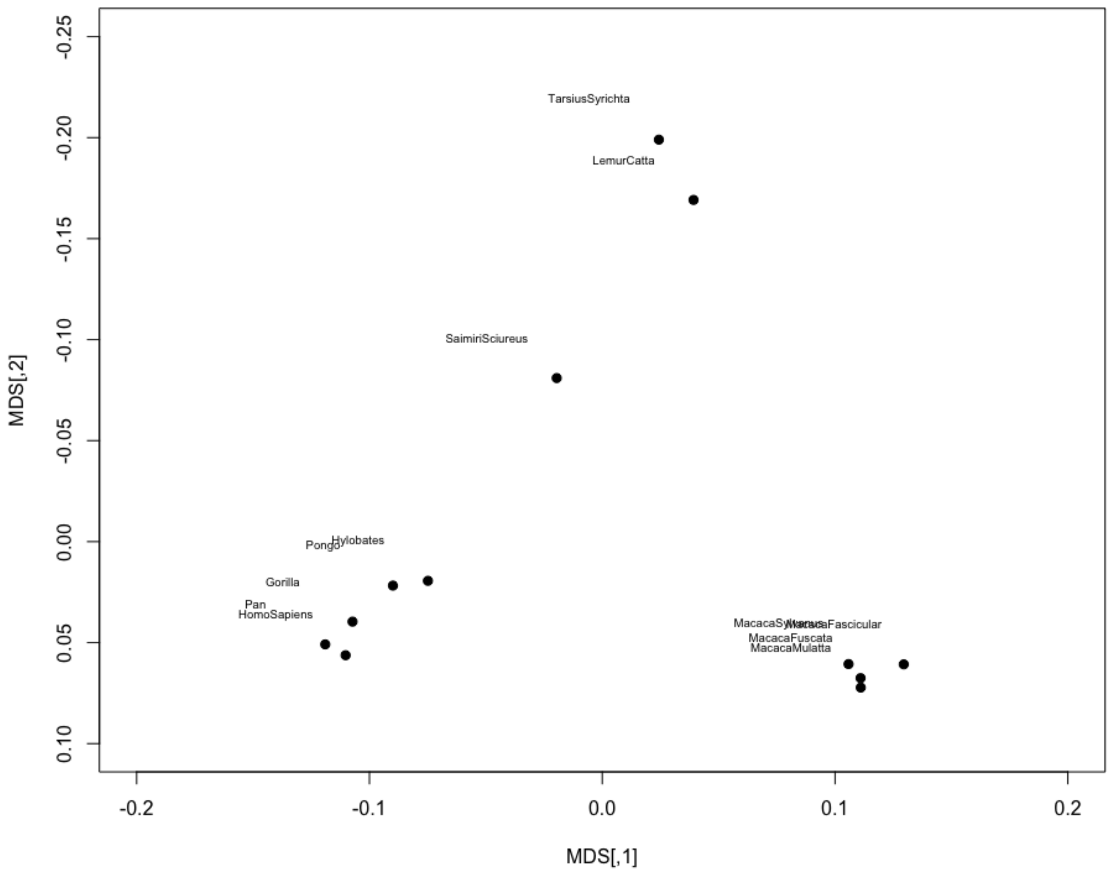
Exercise 2: The following is a phylogenetic distance matrix. Use multidimensional scaling (metric, that is , cmdscale) to make a map of these species. What does it mean?

HomoSapiens	0.000	0.089	0.104	0.161	0.182	0.232	0.233	0.249	0.256	0.273	0.322	0.308
Pan	0.089	0.000	0.106	0.171	0.189	0.243	0.251	0.268	0.249	0.284	0.321	0.309
Gorilla	0.104	0.106	0.000	0.166	0.189	0.237	0.235	0.262	0.244	0.271	0.314	0.293
Pongo	0.161	0.171	0.166	0.000	0.188	0.244	0.247	0.262	0.241	0.284	0.303	0.293
Hylobates	0.182	0.189	0.189	0.188	0.000	0.247	0.239	0.257	0.242	0.269	0.309	0.296
MacacaFuscata	0.232	0.243	0.237	0.244	0.247	0.000	0.036	0.084	0.124	0.289	0.314	0.282
MacacaMulatta	0.233	0.251	0.235	0.247	0.239	0.036	0.000	0.093	0.120	0.293	0.316	0.289
MacacaFascicular	0.249	0.268	0.262	0.262	0.257	0.084	0.093	0.000	0.123	0.287	0.311	0.298
MacacaSylvanus	0.256	0.249	0.244	0.241	0.242	0.124	0.120	0.123	0.000	0.287	0.319	0.287
SaimiriSciureus	0.273	0.284	0.271	0.284	0.269	0.289	0.293	0.287	0.287	0.000	0.320	0.285
TarsiusSyrichta	0.322	0.321	0.314	0.303	0.309	0.314	0.316	0.311	0.319	0.320	0.000	0.252
LemurCatta	0.308	0.309	0.293	0.293	0.296	0.282	0.289	0.298	0.287	0.285	0.252	0.000

Solution:
Running the multidimensional scaling function cmd scale we get the following plot of each of the observations.
Code:

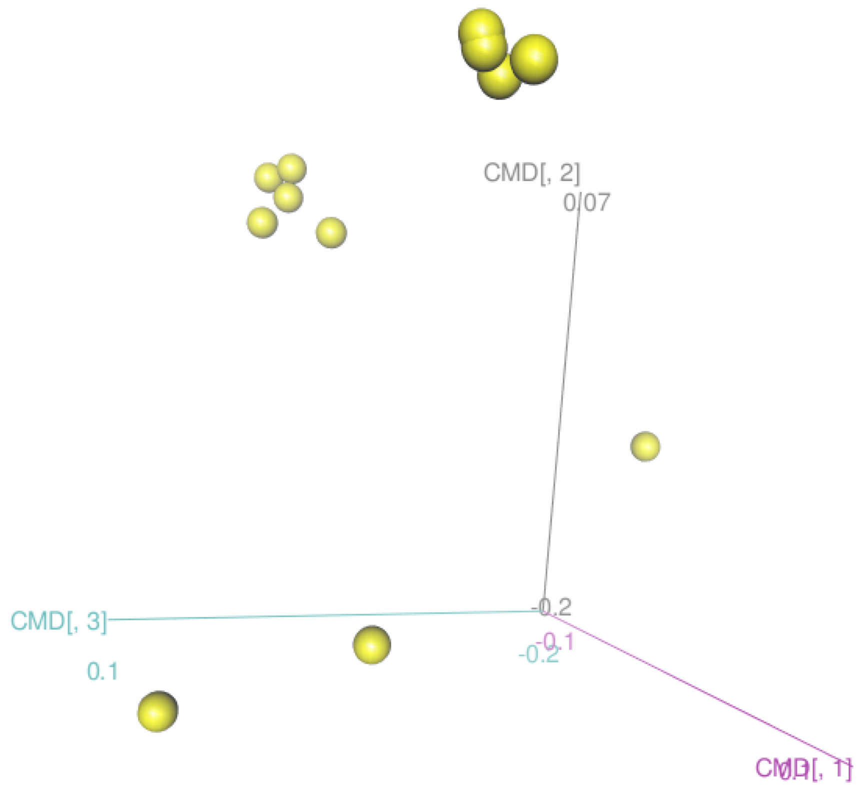
```
CMD <- cmdscale(as.dist(D), k = 2)

CMDGOF <- cmdscale(as.dist(D), list=TRUE, k = 2)$GOF
    [1] 0.5939294 0.5942756

cmdscale(as.dist(D), list=TRUE, k = 3)$GOF
    [1] 0.7328829 0.7333100

plot(MDS, pch=19, xlim = c(-.20, .20), ylim = c(.10, -.25))
text(MDS[,1]-.03, MDS[,2]-.02, rownames(D), cex=.6)

scatter3d(x = CMD[,1], y = CMD[,2], z = CMD[,3], surface = FALSE)
```

Figure 2: MDS plot of phylogenetic distance matrix.



We can see some obvious clustering going on in 2-dimensions. The bottom left corner contains larger primates like Gorilla, Homosapians, Pan and Pongo. the bottom right we can see a clustering of different species in the Macaca genus. The upper cluster seem to be smaller primates like Lemurs and Trasius. When we look at the GOF of our multidimensional scaling we see that we kept around 59% of the variance after compressing the dimension. It might be worthwhile to explore visualizing the data in three dimensions, because when we look at the GOF in three dimensions we get around 73% of the variance. Doing so we get the following plot,

Figure 3: MDS plot of phylogenetic distance matrix with 3 dimensions.



This shows us that the lower group of observations isn't really a cluster. It's possible that in higher dimensions there is a greater distance among the group of large primates, I would assume because of the relation in genus the other cluster stays relatively close together in higher dimensions.

Exercise 3: Using the R base function princomp() does principal component analysis. Run the following.

```
X <- data_mat[,4:7]
```

```
summary(X)
pairs(X)
#
#
tmp <- princomp(X)
plot(tmp)
tmp
summary(tmp)
#
tmp2 <- princomp(X, cor=TRUE)
plot(tmp2)tmp2summary(tmp2)
```

a. Does using cor=TRUE make a difference? In general, when should you not use cor=TRUE? When should you use cor=TRUE? Try multiplying column x1 by 100 and run princomp() both ways. Does this support your thoughts about correlation vs covariance matrices in principal components?

Solution:
Yes, using the correlation matrix for principal component analysis removes the effect of scale for all of the variables, essentially standardizing each variable. If the scale matters in our analysis then we would want to use the covariance matrix. If scale doesn't matter then the correlation matrix is suitable and even better conditioned for the eigenvalue problem. Beyond that if you have a constant variable, the variance will be 0 and the related correlations will be undefined.

Running the following code we see this in action. When we scale the first variable by 100 our covariance matrix PCA attributes all the variance in the data to that first variable. While the correlation matrix PCA is invariant to change in scale.
Code:

```
--------------------------------------------------
#### PCA with X and Covariance Matrix
summary(princomp(X))
Importance of components:
                          Comp.1    Comp.2     Comp.3     Comp.4
Standard deviation     2.5105558 1.0157902 0.79133114 0.43315887
Proportion of Variance 0.7734982 0.1266274 0.07684863 0.02302576
Cumulative Proportion  0.7734982 0.9001256 0.97697424 1.00000000
--------------------------------------------------
#### PCA with X and Correlation Matrix
summary(princomp(X, cor = TRUE))
Importance of components:
                          Comp.1    Comp.2    Comp.3     Comp.4
Standard deviation     1.4408041 1.0741502 0.8147347 0.32633153
Proportion of Variance 0.5189791 0.2884497 0.1659481 0.02662307
Cumulative Proportion  0.5189791 0.8074288 0.9733769 1.00000000
--------------------------------------------------
--------------------------------------------------
#### Scaling the first variable by 100
```

```
XScaledUP <- cbind(X[,1]*100, X[,(2:4)])

#### PCA with Scaled X and Covariance Matrix
summary(princomp(XScaledUP))
Importance of components:
                           Comp.1       Comp.2       Comp.3       Comp.4
Standard deviation     103.4506348 1.9300487517 9.157441e-01 4.780835e-01
Proportion of Variance   0.9995524 0.0003479171 7.832267e-05 2.134749e-05
Cumulative Proportion    0.9995524 0.9999003298 9.999787e-01 1.000000e+00
--------------------------------------------------
#### PCA with Scaled X and Correlation Matrix
summary(princomp(XScaledUP, cor = TRUE))
Importance of components:
                          Comp.1    Comp.2    Comp.3     Comp.4
Standard deviation     1.4408041 1.0741502 0.8147347 0.32633153
Proportion of Variance 0.5189791 0.2884497 0.1659481 0.02662307
Cumulative Proportion  0.5189791 0.8074288 0.9733769 1.00000000
--------------------------------------------------
```

b. Looking at the variances (for instance, when cor=TRUE), how many principal components do you think you should use? Why? Does it look like principal components was a good approach to follow in this case, why or why not?

Solution:
It seems as though there are several rule of thumb test to decide how many components we should retain. I've hear we include them from largest to smallest until we've explained 80% of the variance. I've heard the same rule for 90% and I've also heard that you should drop any components that contribute less than 10% to the variance. I think that looking at our data it seems as though we only need to retain 3 of the 4 components.

I like to think of the components as the magnitude of the semi-axis of the hyperellipse which bounds the data. When we have a small value for a component there exists a hyperellipse in a smaller dimension which is a good approximation for the larger dimension. Generally when we have small components in our data, we are adding extra unneeded complexity(both computationally, and parsimoniously) in the form of another dimension. Whenever we can effectively reduce the dimension of our data with PCA, it is worthwhile.

c. What does the following set of loadings and scores tell you (in general, what are loadings and scores and what are they used for)?

```
X <- data_mat[,4:7]
hold = princomp(X, cor=TRUE, scores=TRUE)
names(hold)
```

```
hold$scores
hold$loadings
```

Solution:
Running the given code we get the following loadings and scores.
Code:

```
> hold$scores
          Comp.1      Comp.2     Comp.3      Comp.4
 [1,] -3.59837512  0.46592358 -0.3087188 -0.04902597
 [2,] -0.10501162 -0.32382941 -0.9637492  0.81770366
 [3,] -1.03858755 -1.50611264 -0.7762246 -0.57497050
 [4,]  2.13004445  0.68264826 -0.5335500 -0.12447069
 [5,] -1.02326106  1.85187203  0.2233541 -0.04203984
 [6,] -1.80092901  1.50535704 -0.6359655  0.19873641
 [7,] -0.35930163 -0.04003031  0.3851416 -0.17866740
 [8,]  0.92084766 -0.72621391 -1.4723743  0.06481649
 [9,]  1.59340467 -0.81261185 -0.3868939 -0.50560097
[10,]  1.40325871  0.14930224 -0.4962171  0.12098110
[11,]  1.03431067  1.81051677  0.4264689 -0.39770549
[12,] -0.12086602 -2.56378551  0.7111325  0.21298274
[13,] -1.25216137  0.09466807  1.3447405  0.09458907
[14,]  1.02000282  0.76562254  0.9965436  0.21105188
[15,]  0.08973018 -0.14030215 -1.0270956 -0.22983631
[16,]  2.26714084 -0.12109288  1.4461451  0.23233223
[17,] -0.76243448 -0.21277685 -0.2243883  0.44816422
[18,]  1.11665484 -0.08876730 -0.1424104  0.23257586
[19,]  0.06495144  0.66935439  0.2354269 -0.31574799
[20,] -1.57941842 -1.45974212  1.1986344 -0.21586849



> hold$loadings
Loadings:
   Comp.1 Comp.2 Comp.3 Comp.4
x1  0.520  0.281  0.706  0.389
x2  0.500 -0.527 -0.447  0.523
x3 -0.168 -0.802  0.544 -0.183
x4  0.672               -0.736

               Comp.1 Comp.2 Comp.3 Comp.4
SS loadings      1.00   1.00   1.00   1.00
Proportion Var   0.25   0.25   0.25   0.25
Cumulative Var   0.25   0.50   0.75   1.00
```

The loadings tell us how much of each variable should be weighted when computing the component, and the scores tell us the weight of each component in a given observation.

d. Repeat a (correlation-based) principal component analysis on the following. What conclusions do you draw? Do the variances of the principal components add up to 4 (which is the number of variables)?

```
M <-structure(c(-1.4, -1.09, -0.19, -0.09, 1.04, -0.5, -0.22, -1.65, -1.02,
0.71, 0.23, -0.48, -1.4, -1.2, -0.5, 0.19, 0.9, -0.59, -0.23, -1.9, -0.73,
0.84, 0.21, -0.46, -1.62, -1.02, 0.03, 0.18, 1.11, -0.62, -0.12, -1.45, -1.13,
0.93, 0.4, -0.52, -1.52, -1.11, -0.24, -0.9, 0.99, -0.38, -0.04, -1.54, -1.15,
0.37, 0.48, -0.35),
.Dim = c(12L, 4L), .Dimnames = list(NULL, c("X1", "X2", "X3", "X4")))
```

Solution:
Running the correlation-based PCA we get the following,
Code:

```
> summary(princomp(M, cor = TRUE))
Importance of components:
                          Comp.1     Comp.2      Comp.3       Comp.4
Standard deviation     1.9551976 0.36853202 0.194175456 0.0606830718
Proportion of Variance 0.9556994 0.03395396 0.009426027 0.0009206088
Cumulative Proportion  0.9556994 0.98965336 0.999079391 1.0000000000

> c(0.9556994*4, 0.03395396*4, 0.009426027*4, 0.0009206088*4)
    [1] 3.822797600 0.135815840 0.037704108 0.003682435

> sum(c(0.9556994*4, 0.03395396*4, 0.009426027*4, 0.0009206088*4))
    [1] 4
```

Looking at the proportion of variance explained it seems clear the the data are highly one dimensional since we used the correlation based approach we can be sure that this behavior is not due to variable scaling. I would likely conclude that the data could be successfully reduced to a single dimension with a very little drop in variance.

Exercise 4: Find any multivariate data set of interest to you and perform PCA. The data should have at least three columns. Did PCA help to reduce dimensionality? Looking at the loadings of PC1, what does the PC seem to represent? Try to find a data set that (as far as you can tell) has not already had PCA performed on it online?

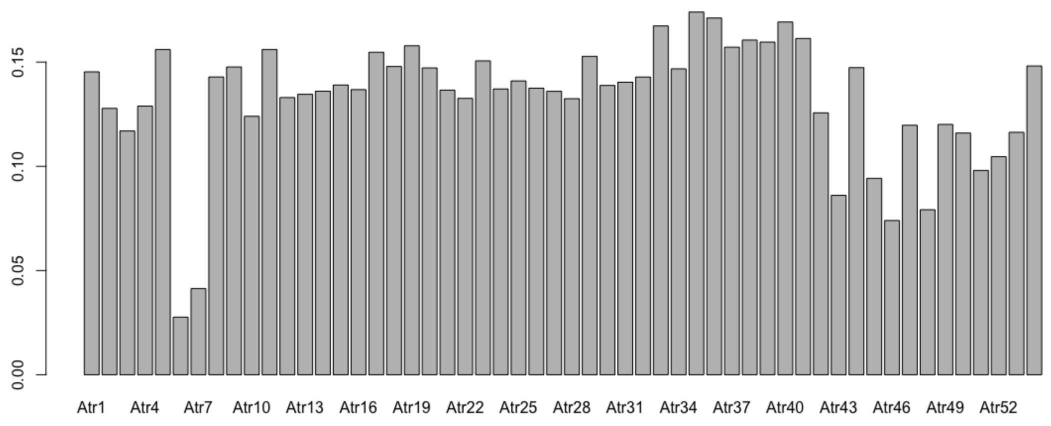
Solution:
The data set I used comes from a survey of couples on the subject of their relationship

with the goal building a model to predict if the couple is divorced or not. The data set has 54 predictors which represent survey questions with participants responding on an integer scale from 0 to 4. The data was downloaded from UC Irvine open source machine learning data repository. Performing a PCA(covariance based) using princomp() we find that almost 77% of the variance is explained by a single component. Looking at the loadings for that component it seems to represent a fairly even weighted sum, with only survey questions 6 and 7 contributing noticeably less than the others.
Code:

```
> divorce <- read_delim("divorce.csv", delim = ";",
+      escape_double = FALSE, trim_ws = TRUE)


> head(divorce)
   Atr1  Atr2  Atr3  Atr4  Atr5  Atr6  Atr7  Atr8  Atr9   ...
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1     2     2     4     1     0     0     0     0     0
2     4     4     4     4     4     0     0     4     4
3     2     2     2     2     1     3     2     1     1
4     3     2     3     2     3     3     3     3     3
5     2     2     1     1     1     1     0     0     0
6     0     0     1     0     0     2     0     0     0


> PCA <- princomp(divorce[,(1:54)])

> summary(PCA)
Importance of components:
                          Comp.1     Comp.2     Comp.3     Comp.4     Comp.5
Comp.6
Standard deviation     9.9135049 2.11850488 1.80288198 1.37424785 1.30018960 1.22798122
Proportion of Variance 0.7722181 0.03526505 0.02553995 0.01483938 0.01328308 0.01184865
Cumulative Proportion  0.7722181 0.80748315 0.83302310 0.84786248 0.86114556 0.87299421
                           Comp.7      Comp.8      Comp.9     Comp.10
Comp.11    Comp.12
Standard deviation     1.19411314 1.096292351 1.066951587 1.044745300 0.974016988 0.942
Proportion of Variance 0.01120409 0.009443615 0.008944889 0.008576427 0.007454501 0.006
Cumulative Proportion  0.88419830 0.893641917 0.902586806 0.911163233 0.918617734 0.925
                           Comp.13     Comp.14     Comp.15     Comp.16
Comp.17     Comp.18 ...
Standard deviation     0.886145774 0.831871787 0.805137992 0.769054831 0.762101073 0.70
Proportion of Variance 0.006170152 0.005437488 0.005093616 0.004647294 0.004563633 0.00
Cumulative Proportion  0.931764066 0.937201554 0.942295169 0.946942463 0.951506096 0.95



> PCA$loadings[,1]
      Atr1       Atr2       Atr3       Atr4       Atr5       Atr6       Atr7
Atr8
0.14532355 0.12780166 0.11695844 0.12890181 0.15604871 0.02763893 0.04136702 0.14287002
      Atr9      Atr10      Atr11      Atr12      Atr13      Atr14      Atr15
Atr16
```

```
0.14766593 0.12400143 0.15605523 0.13299777 0.13456220 0.13602599 0.13901268 0.13681112
     Atr17      Atr18      Atr19      Atr20      Atr21      Atr22      Atr23
Atr24
0.15471569 0.14788853 0.15783402 0.14721578 0.13656669 0.13260662 0.15059101 0.13711374
     Atr25      Atr26      Atr27      Atr28      Atr29      Atr30      Atr31
Atr32
0.14097043 0.13749577 0.13598832 0.13245854 0.15275977 0.13880466 0.14034432 0.14282389
     Atr33      Atr34      Atr35      Atr36      Atr37      Atr38      Atr39
Atr40
0.16736545 0.14678003 0.17401198 0.17115496 0.15714922 0.16058318 0.15960030 0.16923167
     Atr41      Atr42      Atr43      Atr44      Atr45      Atr46      Atr47
Atr48
0.16133608 0.12565843 0.08606940 0.14735120 0.09421994 0.07399709 0.11971012 0.07918526
     Atr49      Atr50      Atr51      Atr52      Atr53      Atr54
0.12011115 0.11596136 0.09802381 0.10463189 0.11631775 0.14813410

> barplot(PCA$loadings[,1])
```

Figure 4: Bar plot of Loadings for each question in the survey.

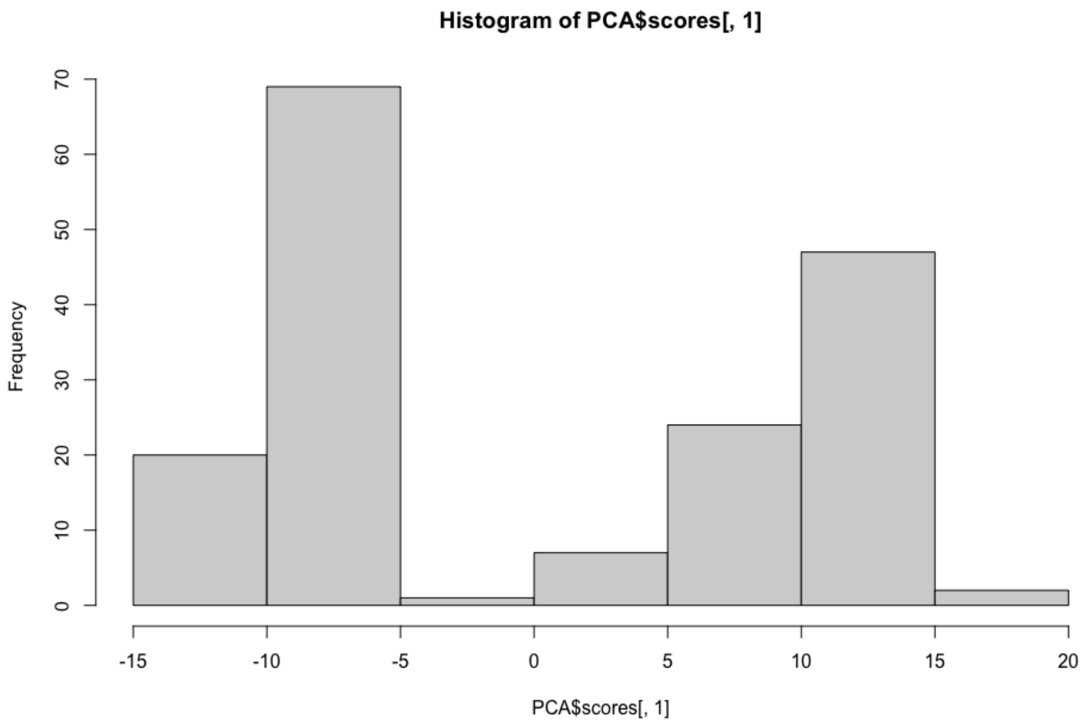


Looking at the scores of the observations in the first component we see they exhibit clear bimodalness, where the boundary seems to designate the divorced and non-divorced observations (around observation 85).

Code:

```
> PCA$scores[,1]
  [1]  -5.034834   6.082489   2.112230   4.765050  -6.048864  -6.125519
  [7]   4.636226   2.866737   1.245790  -3.994374  15.013718  15.013718
 [13]  13.684812  13.684812  13.537460  13.682746  13.682746  13.582610
 [19]  13.140429  13.582610  13.116951  13.116951  12.835676  12.464237
 [25]  12.551155  12.464237  12.391429  12.370017  12.375253  12.462171
 [31]  12.375253  12.362035  12.340623  12.259620  12.275117  11.809457
 [37]  11.867411  11.634894  11.634894  11.685843  11.149442  11.241596
 [43]  11.281387  11.154678  11.241596  11.133265  11.154678  11.133265
 [49]  11.241596  11.154678  11.241596  11.060458  11.009651  10.715591
 [55]  10.936844  10.692113  10.505059  10.505059  10.378350   9.934103
 [61]   9.934103   9.912690   9.934103   9.934103   9.960676   9.723570
 [67]   9.702158   9.789076   9.629350   8.642267   9.406032   8.500951
 [73]   8.629705   8.255912   8.059799   7.111691   5.882899   5.454844
 [79]   5.650616   5.044036   4.252657   9.973884   7.241175   2.845738
 [85] -11.725144 -11.659492 -11.261800 -11.376719 -10.998180 -10.725802
 [91] -11.025546 -10.991772 -10.130046 -10.071233  -9.919028 -10.279158
 [97] -10.105848 -10.118556 -10.541153 -10.041946 -10.130887 -10.133866
[103] -10.147190  -9.969850 -10.152863  -9.652144 -10.127179  -9.651935
[109]  -9.552364  -9.510380  -9.488157  -9.383214  -9.519640  -9.028921
[115]  -9.080701  -9.678057  -9.672411  -9.054363  -9.425221  -9.493400
[121]  -9.396816  -8.885112  -9.122728  -9.453767  -9.472082  -8.762338
[127]  -8.762338  -9.297598  -8.279844  -9.216059  -8.830888  -8.791266
[133]  -9.323096  -8.888367  -9.008702  -9.027225  -8.669607  -9.186244
[139]  -9.186244  -8.736300  -8.841863  -8.533205  -8.887203  -8.802613
[145]  -8.689610  -9.051975  -8.557702  -8.789391  -8.568272  -8.918316
[151]  -8.917761  -8.381054  -8.689880  -8.442909  -8.197338  -8.298847
[157]  -8.291527  -8.337184  -8.112729  -8.214733  -8.122544  -8.010257
[163]  -8.883499  -8.182890  -8.299792  -7.834202  -8.253722  -8.121778
[169]  -8.167597  -8.319674
```

Figure 5: Histogram of Scores in the first component for each participant in the survey.



Not only can PCA reduce the dimensionality of the data significantly, but the loadings of the first component already give us a good model to classify the data (when we consider the scores). The loadings also tell us which survey questions we might want to question the efficacy of (questions 6 and 7).