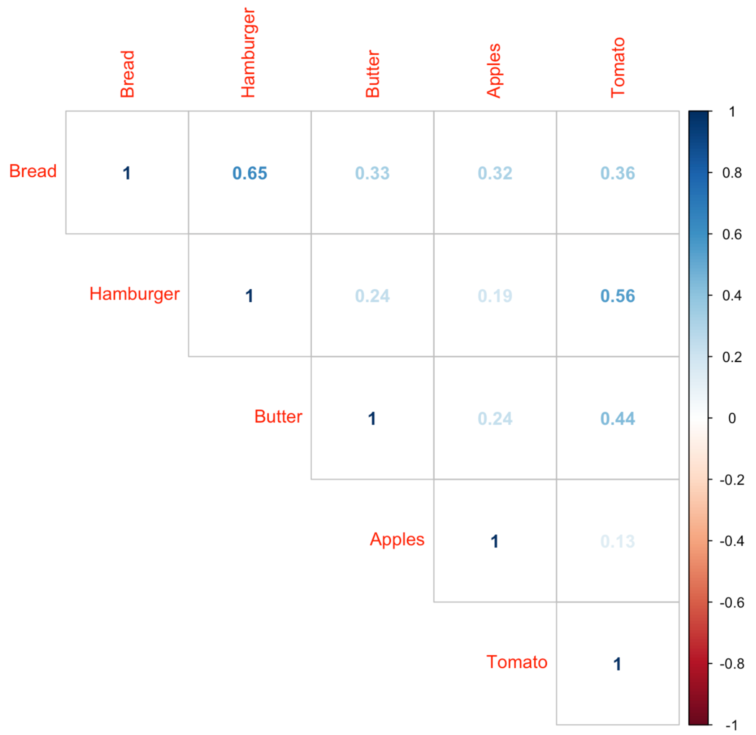
PCA on Food Prices

PCA - ***capture maximum variance, Reduce number of features, group similar features***

Data - Data manipulation / exploration

The data set consist of food prices for 24 cities. Our features (columns/Food) are Bread, Hamburger, Butter, Apples, and Tomato. The mean and median are fairly close for each feature, which means are data is fairly balanced (not skewed). There are no missing values in the data set. Our highest variance is Hamburger prices, thus meaning highest standard deviation. This just means the prices for Hamburger are father away from the mean (more spread apart). Hamburger prices accounts for 32.6%.

There is no negative correlation; This means above average price in a city for food product (feature x) will be associated with an above average price for another food product (feature y).This means that if feature x increases y increases as well and vice versa. So as the price of Bread increase so does the price of Hamburger. This is Pearson correlation; correlation function will convert to z-score at first step. We can see that Hamburger and Bread is highly correlated (0.649) and Hamburger and Tomato is highly correlated (0. 555). This makes sense you because you usually buy these items together when making sandwiches.



Co variance - how much above average x and y are and multiply them together and divide by (n-1). From Co-variance matrix we get the variances diagonally and co-variance for the rest. We see the same features with the highest values that we saw with our correlation graph. Main difference from Correlation and co variance is that co variance gives you the direction of two random variables whereas correlation gives you direction and magnitude. Covariance can take any value while correlation is bounded by 1 and -1.

Process/Analysis

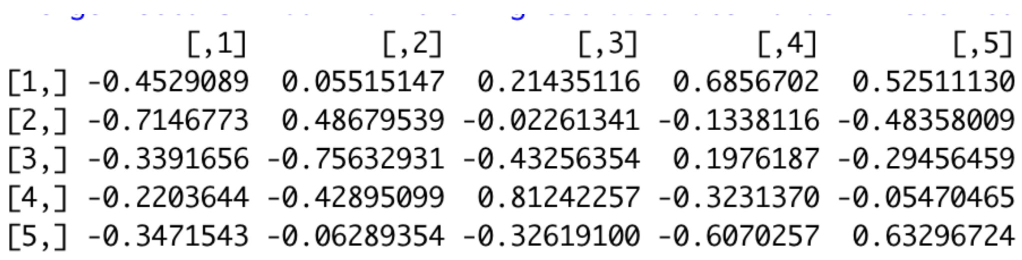
X1 = Bread X2= Hamburger ….. all the feature names

Y⍰1⍰ = a⍰11⍰bread + a⍰12⍰hamburger + ⋯ + a⍰15⍰tomato

Above equation a11 is equal to the first eigen vector and first row (-0.45). A12 = first eigen vector and 2nd row (-0.71)……

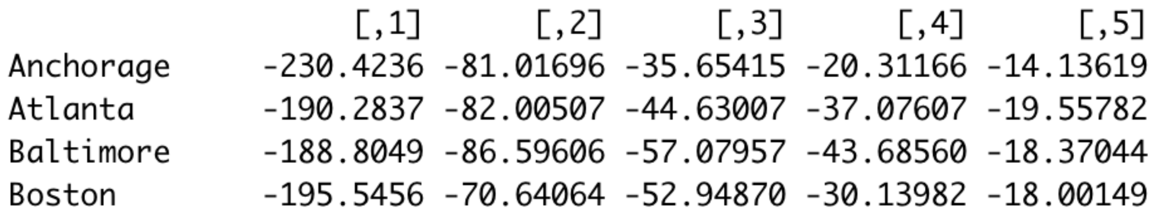
We have 5 eigenvectors created, first vector has all negative values. Keep values that are greater than 0.5 and less than -0.5 (expectations can be made). That means for the first vector we have row2 (Hamburger), and row1(Bread) with the highest eigen value. High price of Hamburger is associated with a low value of principle component 1. High price of butter is associated with a low value of principle component 2. High price of Apple is associated with a high value of principle component 3. If a city has a low value for pc1 it means that the city will be one of the most expensive.

Bread and Hamburger influence PC1 the most.
Butter and Hamburger influence PC2 the most (could argue Apples).
Below is a table of values of the principle components.



```
             [,1]        [,2]        [,3]       [,4]        [,5]
[1,] -0.4529089  0.05515147  0.21435116  0.6856702  0.52511130
[2,] -0.7146773  0.48679539 -0.02261341 -0.1338116 -0.48358009
[3,] -0.3391656 -0.75632931 -0.43256354  0.1976187 -0.29456459
[4,] -0.2203644 -0.42895099  0.81242257 -0.3231370 -0.05470465
[5,] -0.3471543 -0.06289354 -0.32619100 -0.6070257  0.63296724
```

T_1



```
                [,1]      [,2]      [,3]      [,4]      [,5]
Anchorage  -230.4236 -81.01696 -35.65415 -20.31166 -14.13619
Atlanta    -190.2837 -82.00507 -44.63007 -37.07607 -19.55782
Baltimore  -188.8049 -86.59606 -57.07957 -43.68560 -18.37044
Boston     -195.5456 -70.64064 -52.94870 -30.13982 -18.00149
```

T_2

P1 = highest variance, P2,P3……Pn = lowest variance.
To calculate Anchorage value for p1 we use the formula above (y1):

(-0.45*70.9)+(-0.71*135.6)+(-0.33*155)+(-0.22*63.9)+(-0.34*100.1) = -230

As you can see from T2 for column 1 we have Anchorage equal to -230, which is also the highest negative value which means Anchorage is the most expensive when it comes to Hamburger prices. San Diego has the greatest value for y1 which means it has the cheapest price for Hamburger. For y2 Buffalo has the greatest value which means it has the cheapest price for Butter and y3 San Francisco has the highest price for Apples.

Eigen values = 216.79440, 79.12794, 62.26846, 34.67047, 22.22005
First principle component (216.79) explains about 3 times the amount of variance than principle component 2 (79.12). Eigen value explains the importance of principle components.

```
> percentvars
    Bread Hamburger    Butter    Apples    Tomato
0.1686394 0.3260335 0.2051058 0.1683354 0.1318860
> percentvars_pc
[1] 0.52229379 0.19063238 0.15001509 0.08352694 0.05353180
```

T3

Above you see percent of variance in the original data and principle components. We see that the first two pc components have a total roughly variance of 0.71 which explains most of the data. I would not consider pc3 because 4/5 of the original data has a greater variance and pc3 is heavily influenced by apples but pc2 is almost at 0.5 cutoff. We can also argue to consider pc3 because we want to get total of 80% of total variance from the principle components, but not always necessary. Our goal is to reduce dimensions and 70% is still a significant amount of total variance.

```
> cor(y1,data)
          Bread  Hamburger     Butter     Apples     Tomato
[1,] -0.7970559 -0.9045578 -0.5412281 -0.3881604 -0.6908451
```

T4

The above table shows us the correlation of y1 from T1 and our original data. Hamburger is highly correlated which is expected because in T1 we see that y1 is mostly influenced by Hamburger.

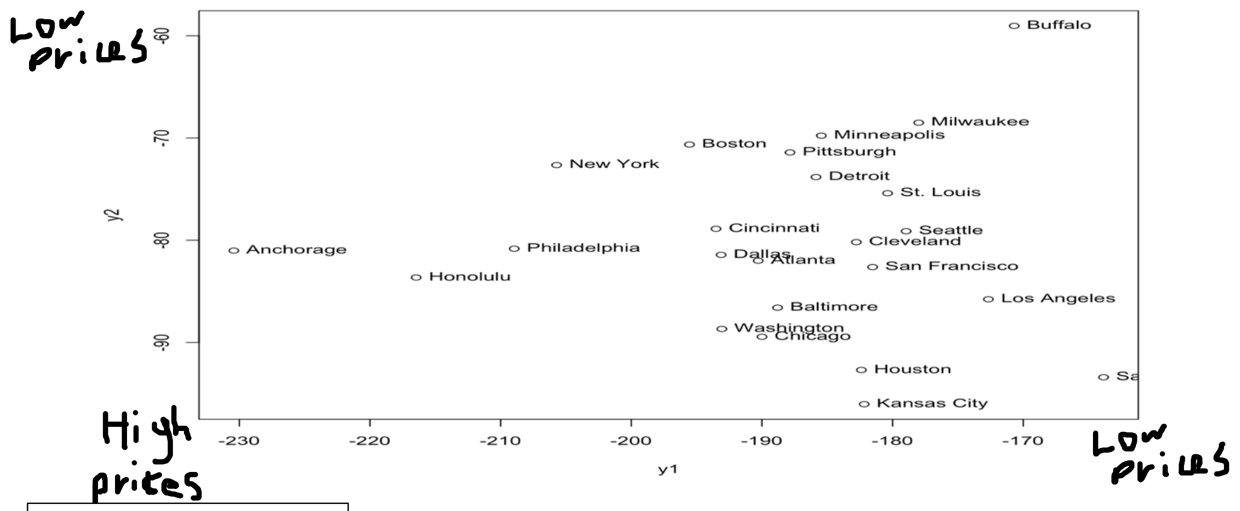
Self Checks

Sum of original variance is 415 and PC (principle component) sum of variance is 415.

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    0
[2,]    0    1    0    0    0
[3,]    0    0    1    0    0
[4,]    0    0    0    1    0
[5,]    0    0    0    0    1
```

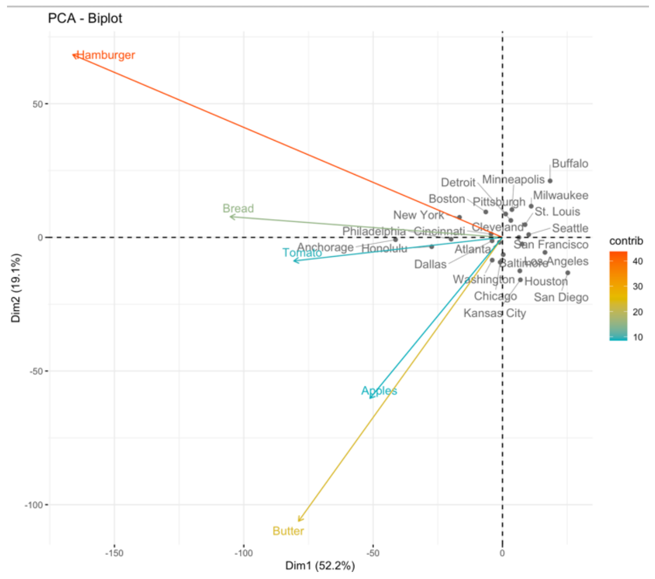
As you can see here the correlation and co variance is 0. So they are not related.

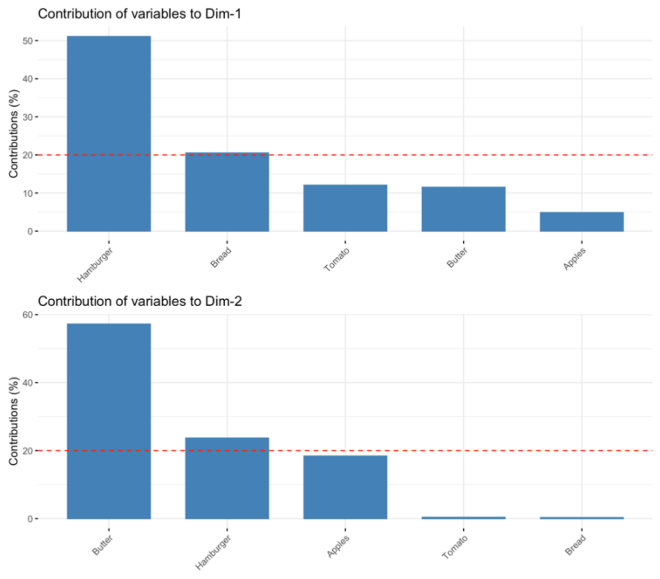
Visuals



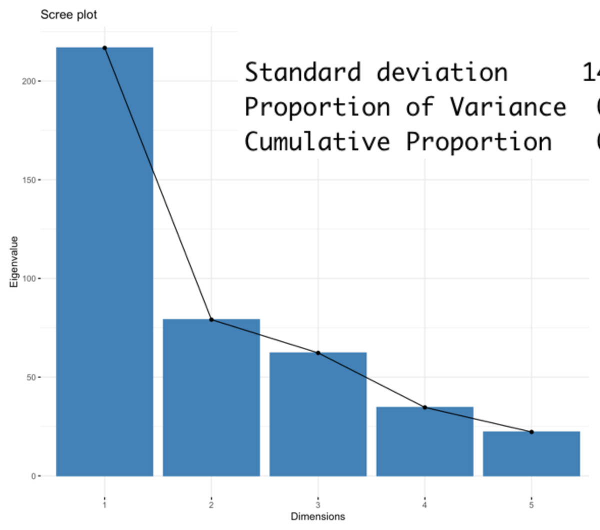
Butter & Hamburger

From the above graph we see that Anchorage is to the left most near -230 which means it has the highest hamburger prices. Buffalo has the lowest butter prices and second cheapest prices for hamburger.





From the biplot graph we vectors pointed in the direction with highest squared multiple correlation with the principal components. It is showing score and loading of each case on the first two principal components.
From the other graph pc1 (Dim-1) shows Hamburger and Bread having at least a 20% variance. For pc2 it shows that Butter and Hamburger have 20% variance with Apple just missing the cutoff.



```
                          PC1    PC2    PC3     PC4     PC5
Standard deviation     14.7239 8.8954 7.8910 5.88816 4.71381
Proportion of Variance  0.5223 0.1906 0.1500 0.08353 0.05353
Cumulative Proportion   0.5223 0.7129 0.8629 0.94647 1.00000
```

In this scree plot we look for the elbow just like knn clusters to see how many pc we should use. ***Note that we have y-axis as eigenvalue****, we can change that to proportion variance as well.*

Regression

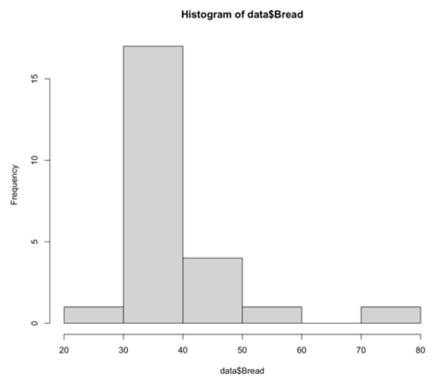
Running regression for all the data and all the principle components we get a R^2 value of 93%. Which means we have the same predictive power. We can explain 93% of the dependent variable from the independent variables.

We then create a correlation matrix with the random data created with the original dataset. We see that Hamburger and Tomato are the two highly correlated. Then run a regression with just those two variables and another regression with pc1 and pc2. We get a R^2 value of 0.76 with our original data and R^2 of 0.91 with our principle component value. This shows us the power of PCA as our first two pc inputs explains 91% of the data. When we used all of our pc it explained 93%, this means we have almost the same amount of predictive power just from the first 2 pc.

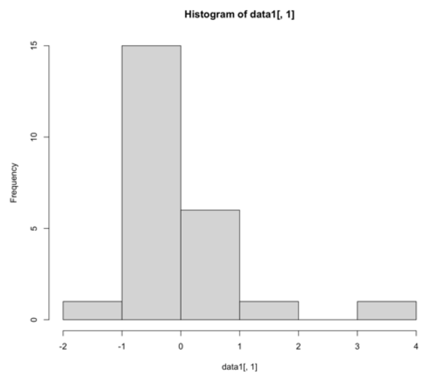
Re-Run on Standardized Data

From the below table we see that the values are a scaled now. They will have a mean of 0 and std of 1 thus variance is 1 as well. Now each variable is equally important. Anchorage has a z-score of 3.87 which means that the price of Bread is 3.87 std (standard deviations) more than the mean price. Also Anchorage has all positive values which means every variable has an above average price. This matches up with our conclusions for the unstandardized data as Anchorage had the most expensive prices.

```
                 Bread  Hamburger      Butter      Apples      Tomato
Anchorage  3.879538966  2.0075514  1.16909152  1.45501855  1.39773432
Atlanta   -0.244027483 -0.0641127  0.00943799  0.25870320  0.83008074
Baltimore -1.140454972 -0.2962078  0.73557618 -0.50693863  1.99241903
Boston     0.568733440  0.6063844 -0.23983333 -1.27258046  0.91117411
Buffalo   -0.471122447 -0.2016506 -2.10394929 -1.93055390 -1.87303157
```



Important to note that standardizing data does not change the distribution of your data.



The correlation matrix will the same as the standardized data, because the first step is to convert the data to z scores. The co-variance matrix is the same as the correlation matrix.

```
> eigenvalues #keep the columns that capture majority of the da
[1] 2.4393555 0.9295911 0.8332378 0.5328728 0.2649429
> eigenvectors #look for the highest absolute value in each col
           [,1]        [,2]       [,3]        [,4]        [,5]
[1,] -0.5099267 -0.05649609  0.4017162  0.53197875  0.54074549
[2,] -0.5200718  0.27761601  0.4074371 -0.07148075 -0.69378685
[3,] -0.3973106 -0.09940133 -0.7684773  0.43041438 -0.23759156
[4,] -0.2909422 -0.87675286  0.0713631 -0.37257819 -0.05243928
[5,] -0.4764421  0.37571444 -0.2774329 -0.62275040  0.40872301
```

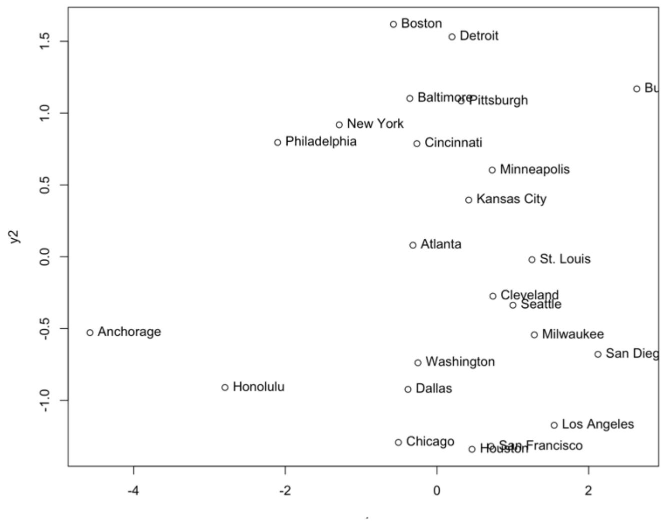
The first eigen value is the most important but less that half. The others are less than 1 which means that they don't tell us more than the original data set. Since we are doing PCA we need at least 2, so I will included pc2.
We have Bread and Hamburger as the most influential for pc1. For pc2 Apple is most influential, this is different from unstandardized data where pc2 Butter was most influential.

Self Checks
Variance is equal to 5 for both pc and regular data, each feature has a variance of 1
Percent variance is 0.2 (1/5) – They are equally important
PC1 accounts for 48% (2.43/5) variance, and PC2 accounts for 18% (0.92/5).

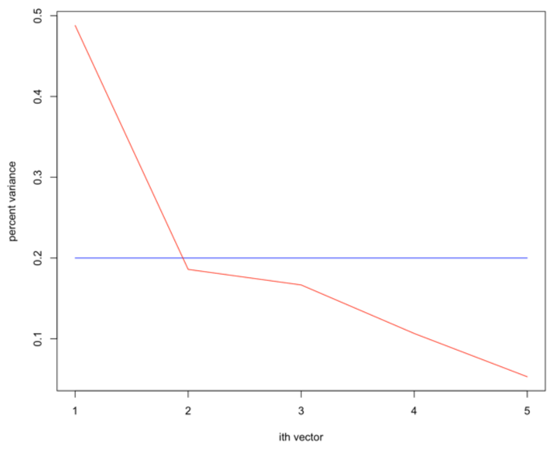
```
cor(y1,data)
        Bread  Hamburger     Butter     Apples     Tomato
,] -0.7964257 -0.8122708 -0.6205371 -0.4544061 -0.7441281
```

Hamburger is still the highest correlated, followed by bread and tomato.
They are still all uncorrelated.

Visuals



Almost the same graph as before but Buffalo if more to the right now. This means Buffalo should have the cheapest prices for Hamburger and Apples.



Graph shows us the percent of variance in each pc in red. In blue we have the variance of each feature which is a straight line because we standardize the data causing each feature it have a variance of 1. Same thing as Scree plot, look for the elbow to determine how many pc we may want to include.

Regression

dv=rowSums(data)+rnorm(24,mean=0,**sd=1) ---- changed from 10 to 1 because its standardized data.** Mean = 0 sd=1

We have the same R^2 value 0.93 for the original data and the pc data. This is the same value for the unstandardized data.

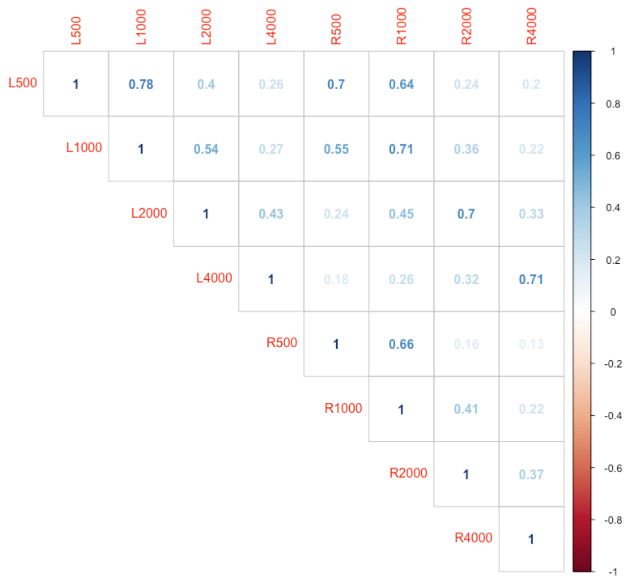
When running regression with the two best correlations (Hamburger & Tomato) we get an R^2 of 0.69 for the original data. We get an R^2 value of 0.93 when running our first two principle components. This gave us a better R^2 value than the one from the unstandardized data by 2%. More importantly the R^2 we get here is the same as the R^2 for all the pc & unstandardized data but in this case we are only using 2 principle components.

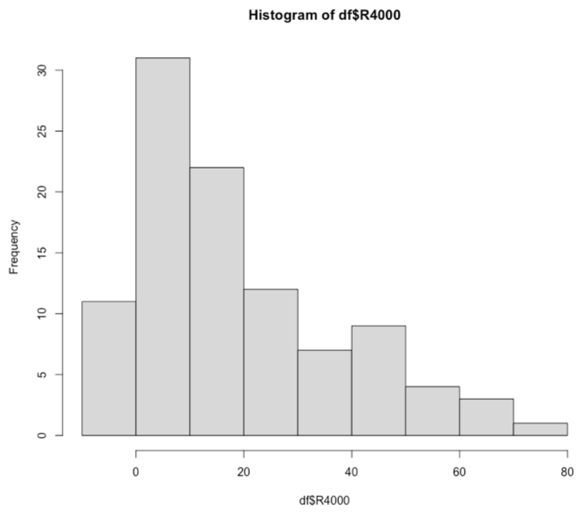
PCA on Audio

Step 1

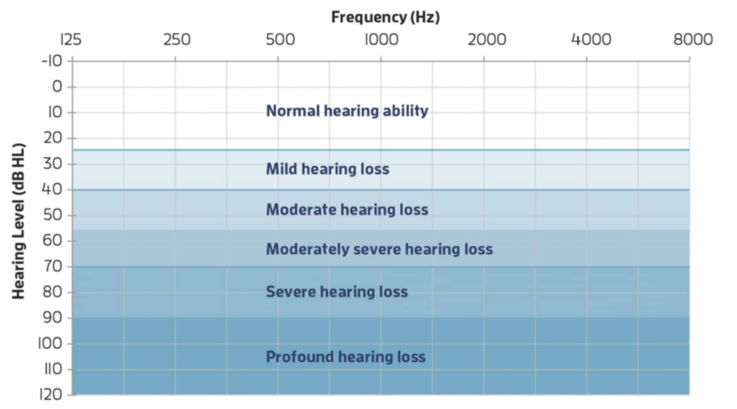
There is no missing values in the data set. The mean and medium for R4000 is significantly different (medium = 15, mean = 21.25). From the histogram below you can see that our data is skewed to the right. The minimum value for each feature is -10. We have 384.77 as our highest variance for L4000 feature, and R4000 almost has the same amount coming in second. From the below figure we see that our highest correlation is 0.78 with feature L500 and L1000. However important to note that R1000 is correlated with 3 other variables but not by much. We have all positive correlations as well. Same logic as the food data if one variable increase the other increases and vice versa.

Decibel is how loud the sounds is. **L4000 means that at least one of the men in left ear received a decibel of 70 (which is at the max value for L4000) at 4000hertz frequency to be able to hear it.** We can also tell from the summary of the dataset the **no one has hearing loss** for frequencies 500 and 1000 in Left ear and Right ear. We determine this by looking at the min and max values.



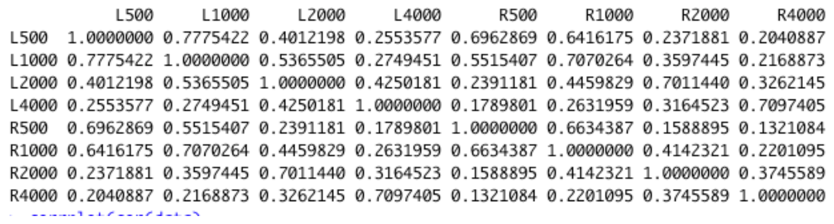


The data was collected on 100 men age 39. Features are in Hertz and values in decibel.
https://www.healthyhearing.com/report/52516-The-abc-s-of-audiograms
Using the above website I learned that for all frequency the normal hearing is around -10 through +25 decibels. **This means our data should be skewed to the right**. For all of our features which are frequencies the men should fall in the normal hearing ability category.



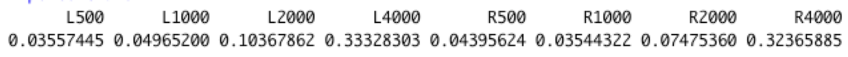
Disclaimer I don't know how accurate this site is because it is not .org but I will use this for our assumptions on the dataset and time.

Step 2
We calculate the co variance matrix. From the below image we see that the vairance is diagonal and the rest are covariance. Our highest value is L500 and L1000 and no negative values which is expected as that was our highest correlation.



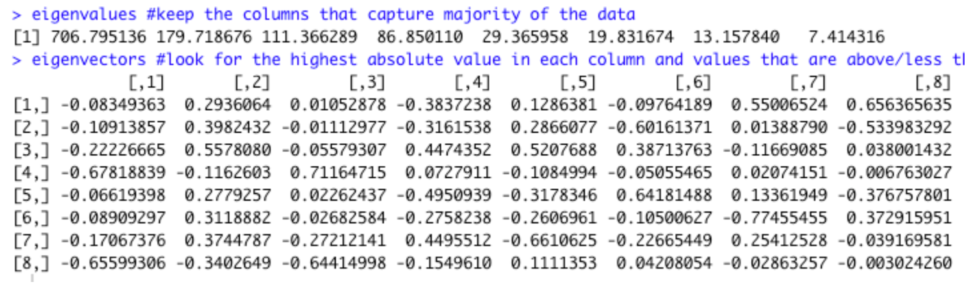
```
          L500     L1000     L2000     L4000      R500     R1000     R2000     R4000
L500  1.0000000 0.7775422 0.4012198 0.2553577 0.6962869 0.6416175 0.2371881 0.2040887
L1000 0.7775422 1.0000000 0.5365505 0.2749451 0.5515407 0.7070264 0.3597445 0.2168873
L2000 0.4012198 0.5365505 1.0000000 0.4250181 0.2391181 0.4459829 0.7011440 0.3262145
L4000 0.2553577 0.2749451 0.4250181 1.0000000 0.1789801 0.2631959 0.3164523 0.7097405
R500  0.6962869 0.5515407 0.2391181 0.1789801 1.0000000 0.6634387 0.1588895 0.1321084
R1000 0.6416175 0.7070264 0.4459829 0.2631959 0.6634387 1.0000000 0.4142321 0.2201095
R2000 0.2371881 0.3597445 0.7011440 0.3164523 0.1588895 0.4142321 1.0000000 0.3745589
R4000 0.2040887 0.2168873 0.3262145 0.7097405 0.1321084 0.2201095 0.3745589 1.0000000
```

Below is table of the percent variance, which means how much variance each variable accounts for. L4000 has the highest and the second highest is R4000. We should expect this because the max value for L4000 is 70 and R4000 is 75 which is significantly higher than the max of the rest of the variables. Also we know that L500,L1000 and R500, R1000 should have the lowest variance because there min max is the small.



```
      L500      L1000      L2000      L4000       R500      R1000      R2000      R4000
0.03557445 0.04965200 0.10367862 0.33328303 0.04395624 0.03544322 0.07475360 0.32365885
```

Step 3
We have



```
> eigenvalues #keep the columns that capture majority of the data
[1] 706.795136 179.718676 111.366289  86.850110  29.365958  19.831674  13.157840   7.414316
> eigenvectors #look for the highest absolute value in each column and values that are above/less tl
            [,1]       [,2]        [,3]       [,4]       [,5]        [,6]        [,7]         [,8]
[1,] -0.08349363  0.2936064  0.01052878 -0.3837238  0.1286381 -0.09764189  0.55006524  0.656365635
[2,] -0.10913857  0.3982432 -0.01112977 -0.3161538  0.2866077 -0.60161371  0.01388790 -0.533983292
[3,] -0.22226665  0.5578080 -0.05579307  0.4474352  0.5207688  0.38713763 -0.11669085  0.038001432
[4,] -0.67818839 -0.1162603  0.71164715  0.0727911 -0.1084994 -0.05055465  0.02074151 -0.006763027
[5,] -0.06619398  0.2779257  0.02262437 -0.4950939 -0.3178346  0.64181488  0.13361949 -0.376757801
[6,] -0.08909297  0.3118882 -0.02682584 -0.2758238 -0.2606961 -0.10500627 -0.77455455  0.372915951
[7,] -0.17067376  0.3744787 -0.27212141  0.4495512 -0.6610625 -0.22665449  0.25412528 -0.039169581
[8,] -0.65599306 -0.3402649 -0.64414998 -0.1549610  0.1111353  0.04208054 -0.02863257 -0.003024260
```

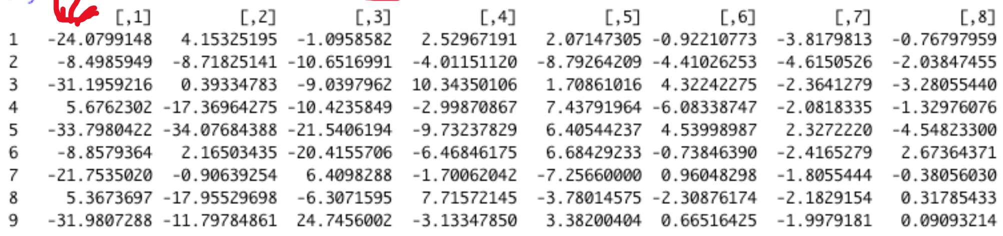
X1 = L500 X2= L1000 ….. all the feature names

$$Y_1 = a_{11}L500 + a_{12}L1000 + \cdots + a_{18}R4000$$

Above equation a11 is equal to the first eigen vector and first row (-0.08). A12 = first eigen vector and 2nd row (-0.10)……

The first eigen value explains 4 times the amount than the second amount.
We have 8 eigenvectors created, first vector has all negative values. Keep values that are greater than 0.5 and less than -0.5 (expectations can be made). That means for the first vector we have row4 (L4000), and row8(R4000) with the highest eigen value. High decibel value for frequency L4000 is associated with a low value of principle component 1. High decibel value for L2000 is associated with a High value of principle component 2.

L4000 and R4000 are the most influential on pc1
L2000 is the most influential on pc2. ----why isn't R2000 influential

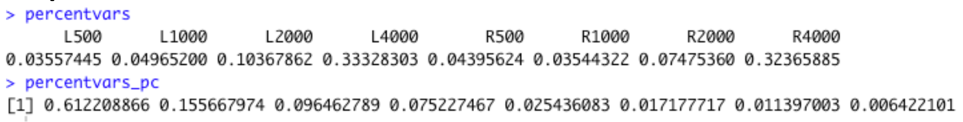


```
         [,1]         [,2]        [,3]        [,4]        [,5]        [,6]       [,7]        [,8]
1  -24.0799148   4.15325195  -1.0958582  2.52967191  2.07147305 -0.92210773 -3.8179813 -0.76797959
2   -8.4985949  -8.71825141 -10.6516991 -4.01151120 -8.79264209 -4.41026253 -4.6150526 -2.03847455
3  -31.1959216   0.39334783  -9.0397962 10.34350106  1.70861016  4.32242275 -2.3641279 -3.28055440
4    5.6762302 -17.36964275 -10.4235849 -2.99870867  7.43791964 -6.08338747 -2.0818335 -1.32976076
5  -33.7980422 -34.07684388 -21.5406194 -9.73237829  6.40544237  4.53998987  2.3272220 -4.54823300
6   -8.8579364   2.16503435 -20.4155706 -6.46846175  6.68429233 -0.73846390 -2.4165279  2.67364371
7  -21.7535020  -0.90639254   6.4098288 -1.70062042 -7.25660000  0.96048298 -1.8055444 -0.38056030
8    5.3673697 -17.95529698  -6.3071595  7.71572145 -3.78014575 -2.30876174 -2.1829154  0.31785433
9  -31.9807288 -11.79784861  24.7456002 -3.13347850  3.38200404  0.66516425 -1.9979181  0.09093214
```

To calculate the first persons value for p1 we use the formula above (y1):
(-0.08*0)+(-0.10*5)+(-0.22*10)+(-0.67 *15)+(-0.06*0)+(-0.08*5)+(-0.17*5)+(-0.65*15) = -24

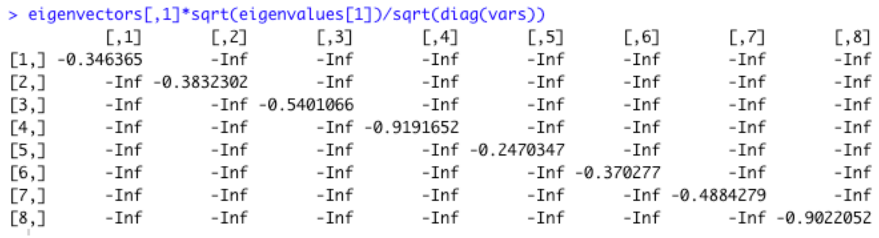
For pc1 row 98 (98th person) we have a value of -92.61, which is the highest negative value which means that he will have the worst hearing in the Left ear at a frequency of 4000hertz.

Step 4



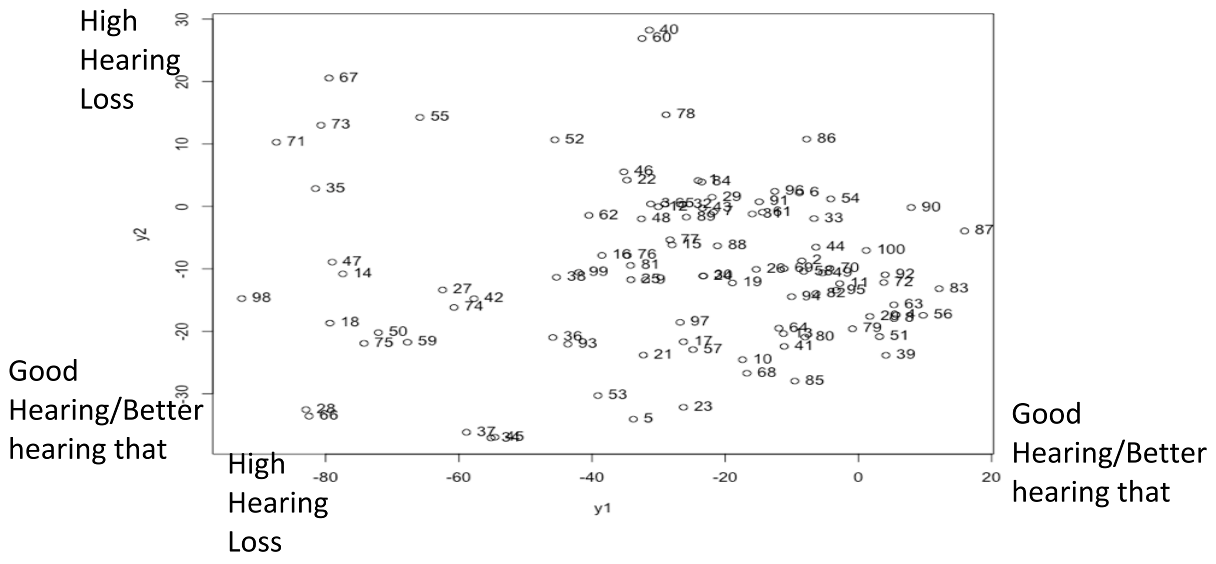
```
> percentvars
      L500      L1000      L2000      L4000       R500      R1000      R2000      R4000 
0.03557445 0.04965200 0.10367862 0.33328303 0.04395624 0.03544322 0.07475360 0.32365885 
> percentvars_pc
[1] 0.612208866 0.155667974 0.096462789 0.075227467 0.025436083 0.017177717 0.011397003 0.006422101
```

We can see how much variance each feature has in the original data and the second table shows how much variance each pc has. We will select the first 2 pc because they account for most of the data. Once you get to pc3 we get a variance that is lower than the original data. The sum of variances for the original data and the principle components is 1154.5.
Below we see a correlation matrix between our pc1 and original data. We can see that L4000 and R4000 are the highly correlated. Also the pc are uncorrelated.

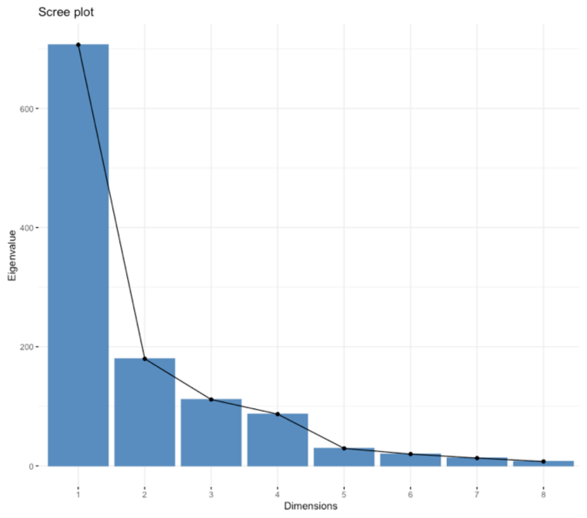


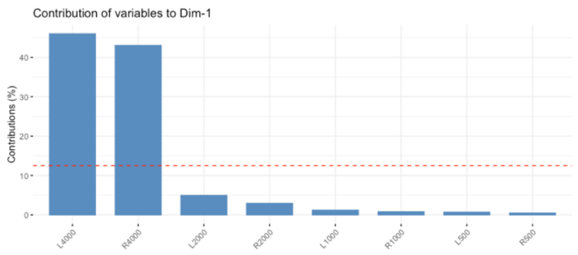
```
> eigenvectors[,1]*sqrt(eigenvalues[1])/sqrt(diag(vars))
          [,1]       [,2]       [,3]       [,4]       [,5]      [,6]       [,7]       [,8]
[1,] -0.346365       -Inf       -Inf       -Inf       -Inf      -Inf       -Inf       -Inf
[2,]      -Inf -0.3832302       -Inf       -Inf       -Inf      -Inf       -Inf       -Inf
[3,]      -Inf       -Inf -0.5401066       -Inf       -Inf      -Inf       -Inf       -Inf
[4,]      -Inf       -Inf       -Inf -0.9191652       -Inf      -Inf       -Inf       -Inf
[5,]      -Inf       -Inf       -Inf       -Inf -0.2470347      -Inf       -Inf       -Inf
[6,]      -Inf       -Inf       -Inf       -Inf       -Inf -0.370277       -Inf       -Inf
[7,]      -Inf       -Inf       -Inf       -Inf       -Inf      -Inf -0.4884279       -Inf
[8,]      -Inf       -Inf       -Inf       -Inf       -Inf      -Inf       -Inf -0.9022052
```

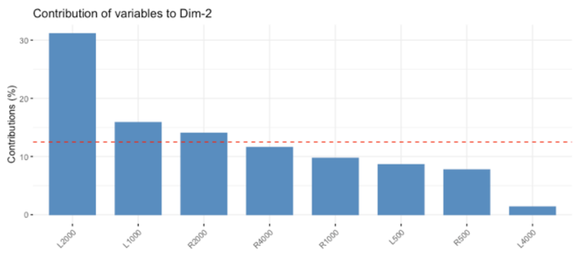
Step 5

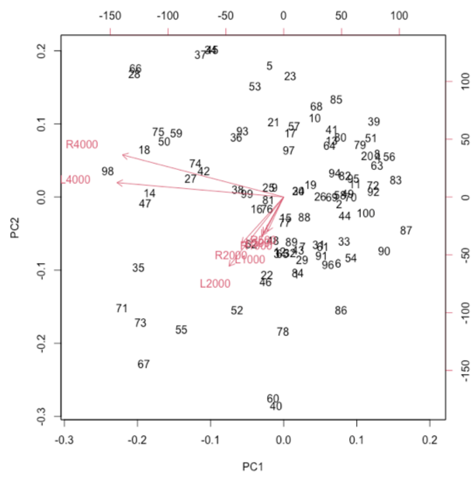


From the above graph we can see that person 98 has the worst hearing. But when we look at the data he has better hearing in his left ear than others. He received a decibel of 55 while person 55 had received 60. This means that R4000 also play a role because person 98 had the worst hearing receiving 75 decibels. We see that person 67, 60 and 40 have the worst hearing for L2000. Below graphs are the same concept as Food PCA









The Biplot (graph to left) shows us that magnitude and direction of the vector. This shows us that R4000 and L4000 is heavily influenced and we see our person 98 is there. We see that in the bottom left that we have a cluster of people who have bad hearing for L2000.

Step 6
dv=rowSums(df)+rnorm(100,mean=0,sd=10) – just changed to 100 because we have 100 rows in original data set.
We received a R^2 value of 97.69% for both the original data and the principle components.
When picking our two best we pick L4000 and R4000 which gives us a R^2 value of 76.62%. When picking pc1 and pc2 we get R^2 = 96.37. This looks too good to be true, but it looks like our first two principle components can explain almost the same amount of data as 5 pc/all data.
Doesn't make sense to me to run a PCA on this type of data.
Consider these as ordinal data.
We can manipulate the data to where -10 through +25 is equal to 0.
26 – 40 is Mild hearing loss so we assign the value 1….. for all values.

PCA on Audio – Standardized Data

Step 1
We create the same correlation matrix and covariant matrix will be the same. Showing a variance of 1 now though.

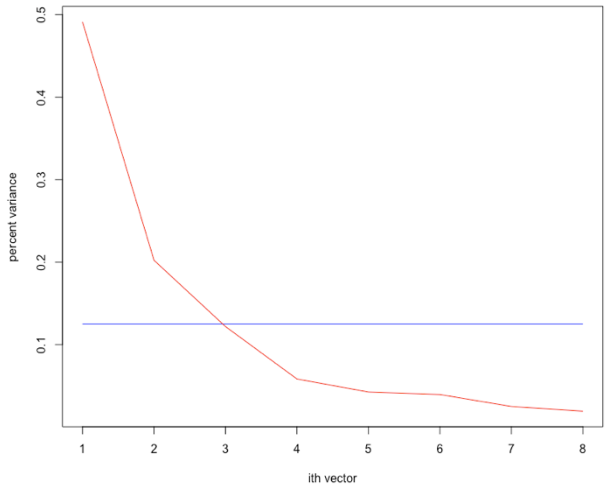
Step 2
Variance is now 0.125 (1/8) for each variable.

Step 3
```
> eigenvalues #keep the columns that capture majority of the data
[1] 3.9290053 1.6183218 0.9753248 0.4667822 0.3400900 0.3158912 0.2001111 0.1544736
> eigenvectors #look for the highest absolute value in each column and values that are above/less than 0.5
           [,1]       [,2]        [,3]       [,4]        [,5]        [,6]        [,7]        [,8]
[1,] -0.4010948  0.3169638 -0.15815686 -0.3277576 -0.02313643  0.44590406 -0.32925525  0.54629986
[2,] -0.4209908  0.2254640  0.05196134 -0.4816310  0.37922678 -0.06745818  0.03312121 -0.62273890
[3,] -0.3663748 -0.2385933  0.47029298 -0.2824293 -0.43924664 -0.06379987  0.52551666  0.18634686
[4,] -0.2808559 -0.4741545 -0.42950248 -0.1610807 -0.35031958 -0.41692698 -0.42694396 -0.08393472
[5,] -0.3432510  0.3860197 -0.25931925  0.4876003 -0.49750307  0.19477711  0.15935065 -0.34253018
[6,] -0.4114209  0.2317725  0.02885395  0.3723164  0.35131760 -0.61363774  0.08367787  0.36136545
[7,] -0.3115483 -0.3170590  0.56293305  0.3914171  0.11078565  0.26503010 -0.47781584 -0.14658750
[8,] -0.2542212 -0.5135121 -0.42622285  0.1590982  0.39595899  0.36604656  0.41393534  0.05082062
```
We see that pc1 is about 2 times greater than pc2. Pc1 is influenced by L1000, R1000, and L500. Pc2 is influenced by R4000, and L4000. The Pc2 we get here is our PC1 for our unstandardized data.

Step 4
```
> percentvars_pc
[1] 0.49112566 0.20229022 0.12191560 0.05834777 0.04251125 0.03948640 0.02501389 0.01930921
```
We see here that pc1 accounts for 49% of the variance and pc2 accounts for 20%. We could consider adding pc3 as that would give us 81% total variance of our data. Our sum of variance is 8. Below we see pc1 correlation with our data. It matches with our table in step 3. We see that L1000 is the most correlated followed by R1000 and the L500. They are uncorrelated.

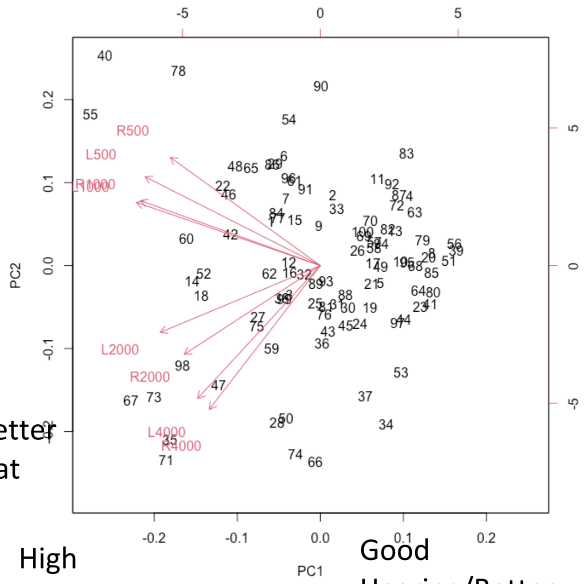
```
           L500     L1000      L2000      L4000       R500      R1000      R2000      R4000
[1,] -0.7950389 -0.834476 -0.7262179 -0.5567047 -0.6803825 -0.8155068 -0.6175423 -0.5039101
```

Step 5



We can see here that variance for original data is at 0.125 and from the red line (pc variance) we determine to have 2 factors. Elbow rule knn.

High Hearing Loss



Good Hearing/Better hearing that

High Hearing Loss

Good Hearing/Better hearing that

From this biplot we see that we can clearly group the liked features of our data. We have L500,L1000,R500, and R1000 as one cluster, and the rest of the data as another cluster. So the lower frequencies and higher frequencies are grouped together.

That means person 40, 55 and 75 have the worst hearing compared to the rest of the men at low frequencies. Note this does not mean that they have hearing loss.

Person 67,71 and 73 have the worst hearing compared to the rest of the men for higher frequencies.

Step 6

We get an R^2 value of 96.83 using all of our data and pc values.
Picking our best two values which is L1000 and R1000 we get a R^2 value of 69.55% and using pc1 and pc2 we get R^2 = 96.67.

FA Audio

Step 1

We get the same results from Step 1 in PCA Audio.

Step 2

Compute eigen values using correlation matrix this mean we are using standardized data. We have the same eigen values as before. So we will take y1 and y2 because they are greater than 1 variance. M is equal to 2 because of this (we also know his from PCA).

```
> eigenvalues
[1] 3.9290053 1.6183218 0.9753248 0.4667822 0.3400900 0.3158912 0.2001111 0.1544736
> (eigenvalues)>1
[1]  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

(3.9290053+1.6183218)/8 = 69% of our total data is explained from the first two.

Step 3

Below is the loading Matrix by taking the sqrt(Eigenvalues*Eigenvectors). We have 2 factors and 8 variables, so 16 total loadings.

```
           [,1]       [,2]
[1,] -0.7950389  0.4032200
[2,] -0.8344760  0.2868202
[3,] -0.7262179 -0.3035224
[4,] -0.5567047 -0.6031874
[5,] -0.6803825  0.4910683
[6,] -0.8155068  0.2948454
[7,] -0.6175423 -0.4033411
[8,] -0.5039101 -0.6532555
```

Step 4

To get the first common value we look at the first row of our loading matrix. So we do (0.795^2)+(0.4032^2) which gives us 0.79. Because this is standardized data we have a variance of 1. So to find the unique variances all we do is subtract 1 from each common value.

1 - 0.79 = 0.21

Can think about it as common correlation and unique correlation

```
> common
[1] 0.7946732 0.7786161 0.6195183 0.6737553 0.7040685 0.7519852 0.5440425 0.6806681
> unique
[1] 0.2053268 0.2213839 0.3804817 0.3262447 0.2959315 0.2480148 0.4559575 0.3193319
```

Step 5

Our recreated matrix is very similar to the original correlation matrix so we can use 2 factors.

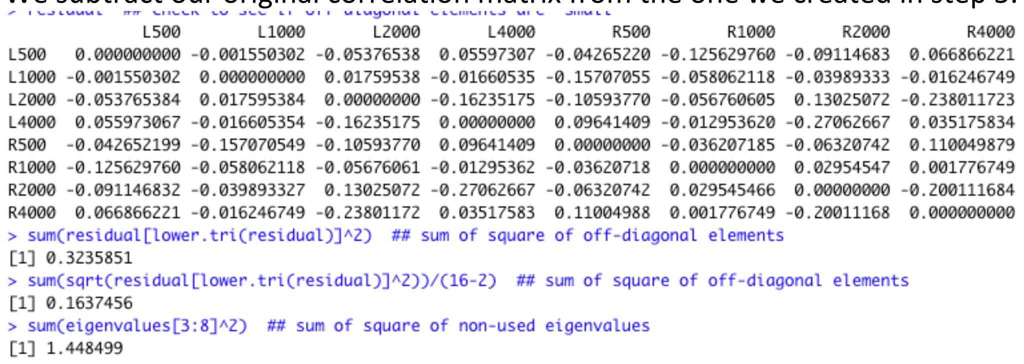
```
          [,1]      [,2]      [,3]       [,4]       [,5]      [,6]      [,7]       [,8]
[1,] 1.0000000 0.7790925 0.4549852 0.19938468 0.73893912 0.7672472 0.3283349 0.13722243
[2,] 0.7790925 1.0000000 0.5189551 0.29155042 0.70861123 0.7650885 0.3996378 0.23313401
[3,] 0.4549852 0.5189551 1.0000000 0.58736984 0.34505576 0.5027435 0.5708933 0.56422619
[4,] 0.1993847 0.2915504 0.5873698 1.00000000 0.08256597 0.2761495 0.5870790 0.67456465
[5,] 0.7389391 0.7086112 0.3450558 0.08256597 1.00000000 0.6996459 0.2220970 0.02205857
[6,] 0.7672472 0.7650885 0.5027435 0.27614948 0.69964585 1.0000000 0.3846867 0.21833274
[7,] 0.3283349 0.3996378 0.5708933 0.58707897 0.22209696 0.3846867 1.0000000 0.57467055
[8,] 0.1372224 0.2331340 0.5642262 0.67456465 0.02205857 0.2183327 0.5746706 1.00000000
```

Recreated

```
           L500     L1000     L2000     L4000      R500     R1000     R2000     R4000
L500  1.0000000 0.7775422 0.4012198 0.2553577 0.6962869 0.6416175 0.2371881 0.2040887
L1000 0.7775422 1.0000000 0.5365505 0.2749451 0.5515407 0.7070264 0.3597445 0.2168873
L2000 0.4012198 0.5365505 1.0000000 0.4250181 0.2391181 0.4459829 0.7011440 0.3262145
L4000 0.2553577 0.2749451 0.4250181 1.0000000 0.1789801 0.2631959 0.3164523 0.7097405
R500  0.6962869 0.5515407 0.2391181 0.1789801 1.0000000 0.6634387 0.1588895 0.1321084
R1000 0.6416175 0.7070264 0.4459829 0.2631959 0.6634387 1.0000000 0.4142321 0.2201095
R2000 0.2371881 0.3597445 0.7011440 0.3164523 0.1588895 0.4142321 1.0000000 0.3745589
R4000 0.2040887 0.2168873 0.3262145 0.7097405 0.1321084 0.2201095 0.3745589 1.0000000
```

Original

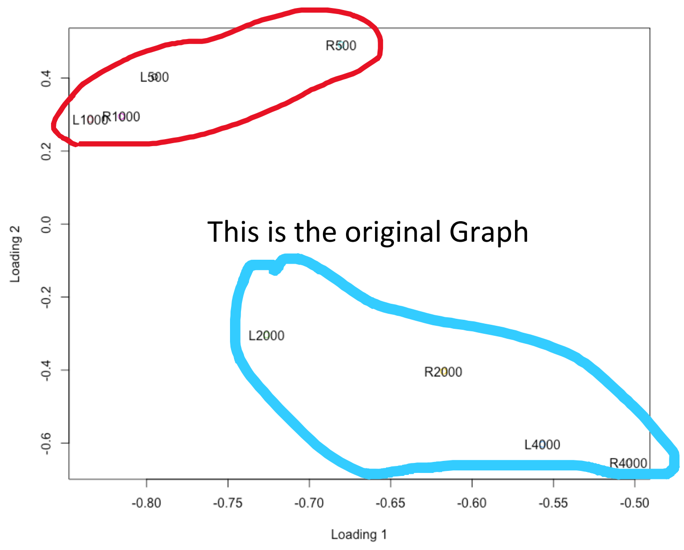
Step 6
We subtract our original correlation matrix from the one we created in step 5.

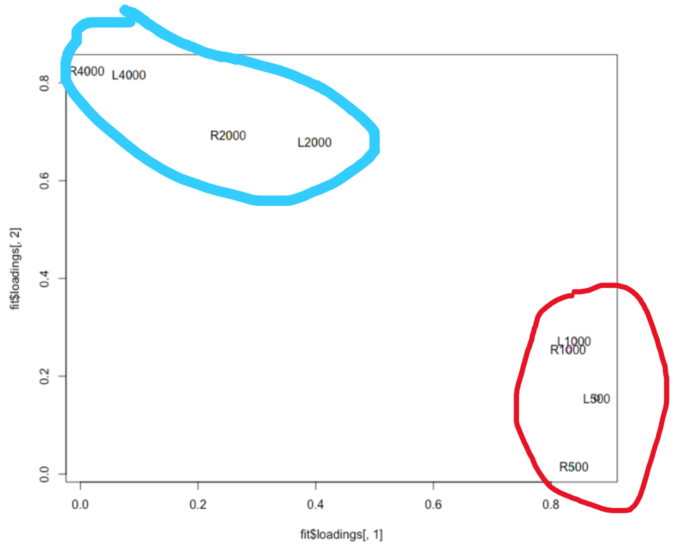


```
> residual  ## check to see if off diagonal elements are small
             L500         L1000        L2000       L4000        R500         R1000        R2000        R4000
L500   0.000000000 -0.001550302 -0.05376538  0.05597307 -0.04265220 -0.125629760 -0.09114683  0.066866221
L1000 -0.001550302  0.000000000  0.01759538 -0.01660535 -0.15707055 -0.058062118 -0.03989333 -0.016246749
L2000 -0.053765384  0.017595384  0.00000000 -0.16235175 -0.10593770 -0.056760605  0.13025072 -0.238011723
L4000  0.055973067 -0.016605354 -0.16235175  0.00000000  0.09641409 -0.012953620 -0.27062667  0.035175834
R500  -0.042652199 -0.157070549 -0.10593770  0.09641409  0.00000000 -0.036207185 -0.06320742  0.110049879
R1000 -0.125629760 -0.058062118 -0.05676061 -0.01295362 -0.03620718  0.000000000  0.02954547  0.001776749
R2000 -0.091146832 -0.039893327  0.13025072 -0.27062667 -0.06320742  0.029545466  0.00000000 -0.200111684
R4000  0.066866221 -0.016246749 -0.23801172  0.03517583  0.11004988  0.001776749 -0.20011168  0.000000000
> sum(residual[lower.tri(residual)]^2)  ## sum of square of off-diagonal elements
[1] 0.3235851
> sum(sqrt(residual[lower.tri(residual)]^2))/(16-2)  ## sum of square of off-diagonal elements
[1] 0.1637456
> sum(eigenvalues[3:8]^2)  ## sum of square of non-used eigenvalues
[1] 1.448499
```

So for L500 and L1000 we have a value of -0.00155. To get this we take the original value of 0.7775422 and subtract it from 0.7790925 which is the recreated matrix. We have

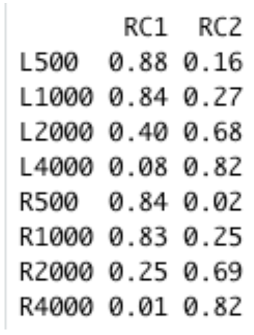
Step 7
From below we can see that this is related to our pc standardized data biplot. We can group together the same features. So L500,L1000, R500 and R1000 will be one group. We have low frequency group and a high frequency group. R500 and L2000 are messing with our grouping a little. So we want to make the low frequency group 0 on one factor and extreme on another.





Since we have some variables intervening with the loadings, we will rotate the graph. We use varimax to maximize the sum of variances of the loadings. This means in simple term that we will get one group to near zero and the other one with large value so we can easily group them together. We do not need to use it because you can see from the above graph that we know what will happen when we rotate it. We also can know that L500,L1000, R500 and R1000 should be grouped together because we know that everyone from our dataset has a max decibel value of 15-25. This means that everyone has good hearing/no hearing loss for those frequencies.

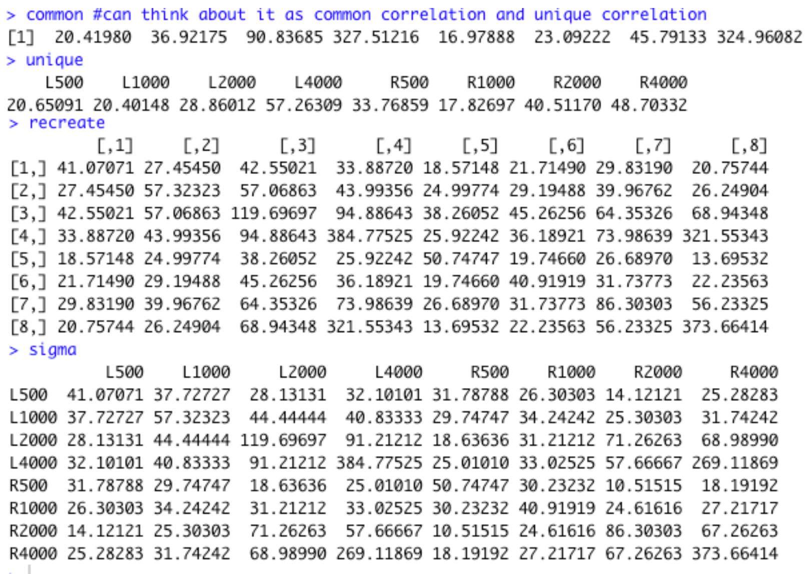
Below is the matrix of what are rotated variables look like. This is how the above graph is made.



```
       RC1  RC2
L500   0.88 0.16
L1000 0.84 0.27
L2000 0.40 0.68
L4000 0.08 0.82
R500   0.84 0.02
R1000 0.83 0.25
R2000 0.25 0.69
R4000 0.01 0.82
```

Using non-Standardized data on Audio FA

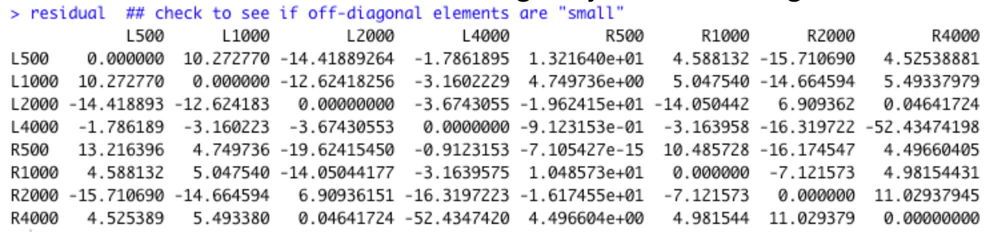
We rerun the whole process but now we use the covariance matrix/variance matrix to compute our eigen values. We will have the same eigen values as the PCA non standardized data. Instead of subtracting from 1 we subtract from the diagonal of the covariance matrix.



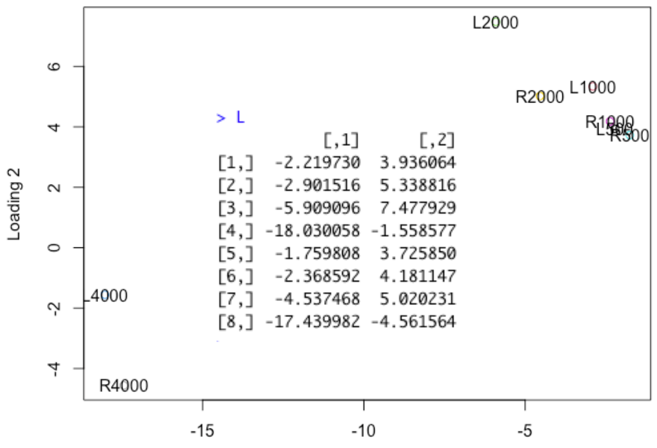
```
> common #can think about it as common correlation and unique correlation
[1]  20.41980  36.92175  90.83685 327.51216  16.97888  23.09222  45.79133 324.96082
> unique
    L500    L1000    L2000    L4000     R500    R1000    R2000    R4000 
20.65091 20.40148 28.86012 57.26309 33.76859 17.82697 40.51170 48.70332 
> recreate
         [,1]     [,2]      [,3]      [,4]     [,5]     [,6]     [,7]      [,8]
[1,] 41.07071 27.45450  42.55021  33.88720 18.57148 21.71490 29.83190  20.75744
[2,] 27.45450 57.32323  57.06863  43.99356 24.99774 29.19488 39.96762  26.24904
[3,] 42.55021 57.06863 119.69697  94.88643 38.26052 45.26256 64.35326  68.94348
[4,] 33.88720 43.99356  94.88643 384.77525 25.92242 36.18921 73.98639 321.55343
[5,] 18.57148 24.99774  38.26052  25.92242 50.74747 19.74660 26.68970  13.69532
[6,] 21.71490 29.19488  45.26256  36.18921 19.74660 40.91919 31.73773  22.23563
[7,] 29.83190 39.96762  64.35326  73.98639 26.68970 31.73773 86.30303  56.23325
[8,] 20.75744 26.24904  68.94348 321.55343 13.69532 22.23563 56.23325 373.66414
> sigma
          L500    L1000     L2000     L4000     R500    R1000    R2000     R4000
L500  41.07071 37.72727  28.13131  32.10101 31.78788 26.30303 14.12121  25.28283
L1000 37.72727 57.32323  44.44444  40.83333 29.74747 34.24242 25.30303  31.74242
L2000 28.13131 44.44444 119.69697  91.21212 18.63636 31.21212 71.26263  68.98990
L4000 32.10101 40.83333  91.21212 384.77525 25.01010 33.02525 57.66667 269.11869
R500  31.78788 29.74747  18.63636  25.01010 50.74747 30.23232 10.51515  18.19192
R1000 26.30303 34.24242  31.21212  33.02525 30.23232 40.91919 24.61616  27.21717
R2000 14.12121 25.30303  71.26263  57.66667 10.51515 24.61616 86.30303  67.26263
R4000 25.28283 31.74242  68.98990 269.11869 18.19192 27.21717 67.26263 373.66414
```

Here we are the common and unique where we add them together to get the variance(diagonal) values. 20.419+20.65 which gives us 41.07 which is our first value

From the residual we see that we did a good job in recreating the matrix.

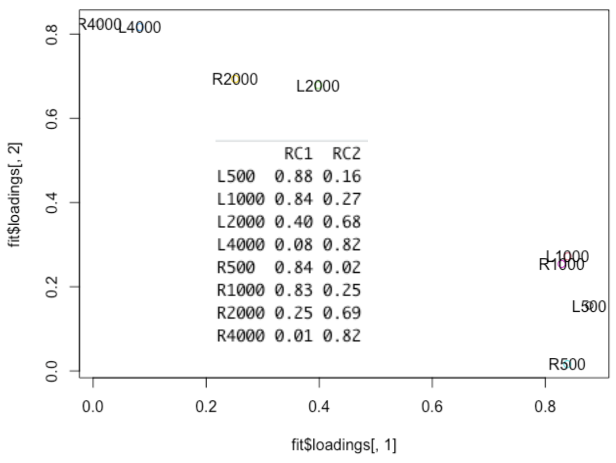


```
> residual  ## check to see if off-diagonal elements are "small"
           L500      L1000        L2000       L4000          R500      R1000      R2000        R4000
L500   0.000000  10.272770 -14.41889264  -1.7861895  1.321640e+01   4.588132 -15.710690   4.52538881
L1000  10.272770  0.000000 -12.62418256  -3.1602229  4.749736e+00   5.047540 -14.664594   5.49337979
L2000 -14.418893 -12.624183   0.00000000  -3.6743055 -1.962415e+01 -14.050442   6.909362   0.04641724
L4000  -1.786189  -3.160223  -3.67430553   0.0000000 -9.123153e-01  -3.163958 -16.319722 -52.43474198
R500   13.216396   4.749736 -19.62415450  -0.9123153 -7.105427e-15  10.485728 -16.174547   4.49660405
R1000   4.588132   5.047540 -14.05044177  -3.1639575  1.048573e+01   0.000000  -7.121573   4.98154431
R2000 -15.710690 -14.664594   6.90936151 -16.3197223 -1.617455e+01  -7.121573   0.000000  11.02937945
R4000   4.525389   5.493380   0.04641724 -52.4347420  4.496604e+00   4.981544  11.029379   0.00000000
```



```
> L
          [,1]      [,2]
[1,]  -2.219730  3.936064
[2,]  -2.901516  5.338816
[3,]  -5.909096  7.477929
[4,] -18.030058 -1.558577
[5,]  -1.759808  3.725850
[6,]  -2.368592  4.181147
[7,]  -4.537468  5.020231
[8,] -17.439982 -4.561564
```

This is the unrotated graph we get and we seem to have a very good cluster already made. L4000 and R4000 are grouped and the rest of the datapoints are grouped. **This makes sense because from the biplot in PCA we had L4000 and R4000 vectors together**.



```
       RC1  RC2
L500  0.88 0.16
L1000 0.84 0.27
L2000 0.40 0.68
L4000 0.08 0.82
R500  0.84 0.02
R1000 0.83 0.25
R2000 0.25 0.69
R4000 0.01 0.82
```

This is the graph we get when we rotate it using varimax. I know this graph is wrong.

fit <- principal(df, nfactors=2, rotate="varimax", covar = TRUE)

I changed covar to equal to TRUE so we wouldn't use standardized data. Also instead of using the correlation matrix I'm using the covariance matrix. But its not working….

library("fortunes")
fortune(108)

Funny quote for R.

FA on Masst

Step 0

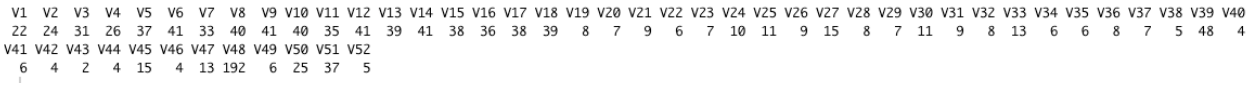
Read in Data

data <- read.csv("masst.csv", na.strings = c("", " "))

data <- as.data.frame(apply(data, 2, as.numeric))
data=data[,-1] #remove first column
data=data[,-53]

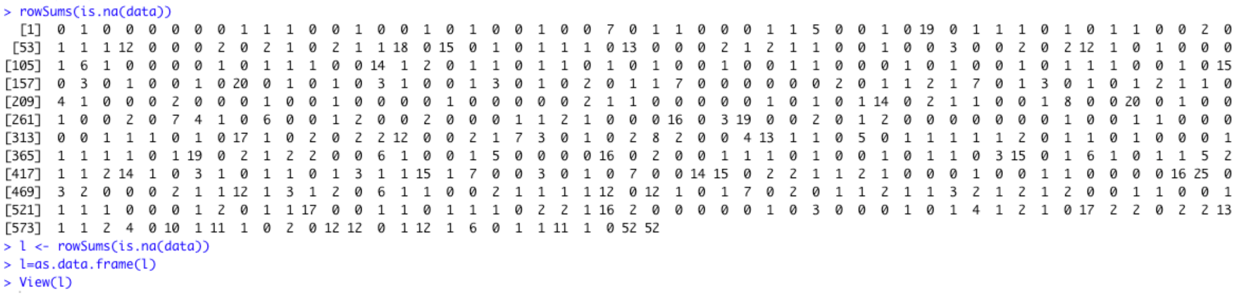
Step 1

We have 1177 total missing values in the dataset. From the table below we see that feature V48 has the most missing variables (192). We could just drop the column. 192/599 = 32% of data missing.



```
 V1  V2  V3  V4  V5  V6  V7  V8  V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 V29 V30 V31 V32 V33 V34 V35 V36 V37 V38 V39 V40
 22  24  31  26  37  41  33  40  41  40  35  41  39  41  38  36  38  39   8   7   9   6   7  10  11   9  15   8   7  11   9   8  13   6   6   8   7   5  48   4
V41 V42 V43 V44 V45 V46 V47 V48 V49 V50 V51 V52
  6   4   2   4  15   4  13 192   6  25  37   5
```

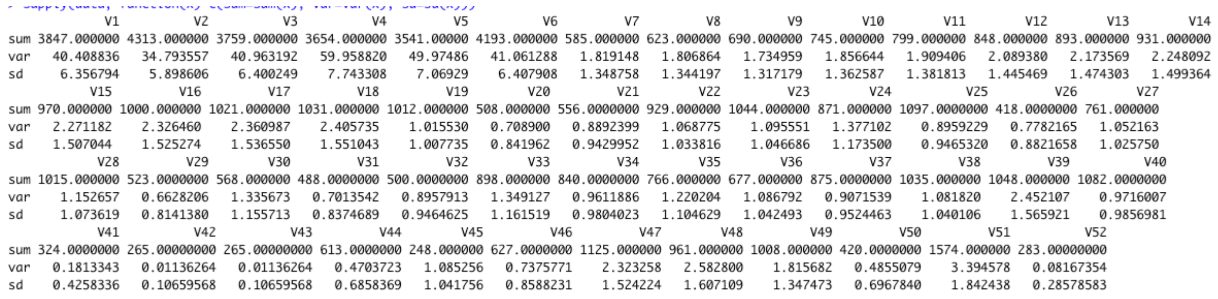
Below is an image of how many values we are missing from each row. Interesting we see that the last two people (598 & 599) did not answer any of the questions. We will drop these rows for sure.



```
> rowSums(is.na(data))
  [1]  0  1  0  0  0  0  0  0  1  1  1  0  0  1  0  0  1  0  1  0  0  1  0  0  7  0  1  1  0  0  0  1  1  5  0  0  1  0 19  0  1  1  1  0  1  0  1  1  0  0  2  0
 [53]  1  1  1 12  0  0  0  2  0  2  1  0  2  1  1 18  0 15  0  1  0  1  1  1  0 13  0  0  0  2  1  2  1  1  0  0  1  0  0  3  0  0  2  0  2 12  1  0  1  0  0  0
[105]  1  6  1  0  0  0  0  1  0  1  1  1  0  0 14  1  2  0  1  1  0  1  1  0  1  0  1  0  0  1  0  0  1  1  0  0  0  1  0  1  0  0  1  0  1  1  1  0  0  1  0 15
[157]  0  3  0  1  0  0  1  0 20  0  1  0  1  0  3  1  0  0  1  3  0  1  0  2  0  1  1  7  0  0  0  0  0  0  2  0  1  1  2  1  7  0  1  3  0  1  0  1  2  1  1  0
[209]  4  1  0  0  0  2  0  0  0  1  0  0  1  0  0  0  0  1  0  0  0  0  0  2  1  1  0  0  0  0  0  1  0  1  0  1 14  0  2  1  1  0  0  1  8  0  0 20  0  1  0  0
[261]  1  0  0  2  0  7  4  1  0  6  0  0  1  2  0  0  2  0  0  0  1  1  2  1  0  0  0 16  0  3 19  0  0  2  0  1  2  0  0  0  0  0  0  0  1  0  0  1  1  0  0  0
[313]  0  0  1  1  1  0  1  0 17  1  0  2  0  2  2 12  0  0  2  1  7  3  0  1  0  2  8  2  0  0  4 13  1  1  0  5  0  1  1  1  1  1  2  0  1  1  0  1  0  0  0  1
[365]  1  1  1  1  0  1 19  0  2  1  2  2  0  0  6  1  0  0  1  5  0  0  0  0 16  0  2  0  0  1  1  1  0  1  0  0  1  0  1  1  0  3 15  0  1  6  1  0  1  1  5  2
[417]  1  1  2 14  1  0  3  1  0  1  1  0  1  3  1  1 15  1  7  0  0  3  0  1  0  7  0  0 14 15  0  2  2  1  1  2  1  0  0  0  1  0  0  1  1  0  0  0  0 16 25  0
[469]  3  2  0  0  0  2  1  1 12  1  3  1  2  0  6  1  1  0  0  2  1  1  1  1 12  0 12  1  0  1  7  0  2  0  1  1  2  1  1  3  2  1  2  1  2  0  0  1  1  0  0  1
[521]  1  1  1  0  0  0  1  2  0  1  1 17  0  0  1  1  0  1  1  1  0  2  2  1 16  2  0  0  0  0  0  1  0  3  0  0  0  1  0  1  4  1  2  1  0 17  2  2  0  2  2 13
[573]  1  1  2  4  0 10  1 11  1  0  2  0 12 12  0  1 12  1  6  0  1  1 11  1  0 52 52
> l <- rowSums(is.na(data))
> l=as.data.frame(l)
> View(l)
```

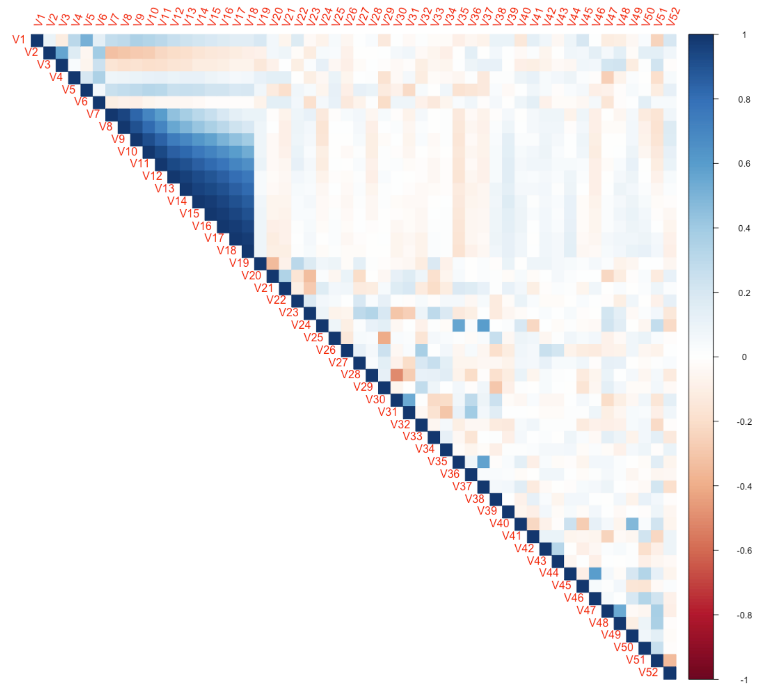
Once removing all NA values and last two rows we are left with 262 rows/people and 52 variables. We could impute the data using Knn imputation, wouldn't use mean or median as these are categorical data.

Below are all the variances for each column. V4 has the highest variance followed by V5 with a variance of 49.97



```
             V1          V2          V3          V4         V5          V6         V7         V8         V9        V10        V11        V12        V13        V14
sum 3847.000000 4313.000000 3759.000000 3654.000000 3541.00000 4193.000000 585.000000 623.000000 690.000000 745.000000 799.000000 848.000000 893.000000 931.000000
var   40.408836   34.793557   40.963192   59.958820   49.97486   41.061288   1.819148   1.806864   1.734959   1.856644   1.909406   2.089380   2.173569   2.248092
sd     6.356794    5.898606    6.400249    7.743308    7.06929    6.407908   1.348758   1.344197   1.317179   1.362587   1.381813   1.445469   1.474303   1.499364
           V15         V16         V17         V18         V19        V20         V21        V22         V23        V24          V25         V26        V27
sum 970.000000 1000.000000 1021.000000 1031.000000 1012.000000 508.000000 556.0000000 929.000000 1044.000000 871.000000 1097.0000000 418.0000000 761.000000
var   2.271182    2.326460    2.360987    2.405735    1.015530   0.708900   0.8892399   1.068775    1.095551   1.377102    0.8959229   0.7782165   1.052163
sd    1.507044    1.525274    1.536550    1.551043    1.007735   0.841962   0.9429952   1.033816    1.046686   1.173500    0.9465320   0.8821658   1.025750
            V28         V29        V30         V31         V32        V33         V34        V35        V36         V37         V38         V39          V40
sum 1015.000000 523.0000000 568.000000 488.0000000 500.0000000 898.000000 840.0000000 766.000000 677.000000 875.0000000 1035.000000 1048.000000 1082.0000000
var    1.152657   0.6628206   1.335673   0.7013542   0.8957913   1.349127   0.9611886   1.220204   1.086792   0.9071539    1.081820    2.452107    0.9716007
sd     1.073619   0.8141380   1.155713   0.8374689   0.9464625   1.161519   0.9804023   1.104629   1.042493   0.9524463    1.040106    1.565921    0.9856981
            V41          V42          V43         V44        V45         V46         V47        V48         V49         V50         V51          V52
sum 324.0000000 265.00000000 265.00000000 613.0000000 248.000000 627.0000000 1125.000000 961.000000 1008.000000 420.0000000 1574.000000 283.00000000
var   0.1813343   0.01136264   0.01136264   0.4703723   1.085256   0.7375771    2.323258   2.582800    1.815682   0.4855079    3.394578   0.08167354
sd    0.4258336   0.10659568   0.10659568   0.6858369   1.041756   0.8588231    1.524224   1.607109    1.347473   0.6967840    1.842438   0.28578583
```

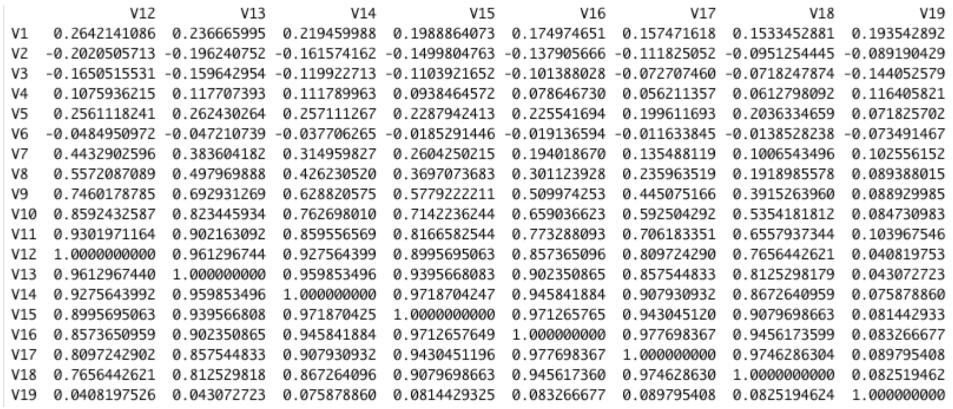
The correlation matrix below the darker the color the more correlated it is. Blue is positively correlated and red is negatively correlated. Looks like column V8 through V19 seem to be correlated with rows V7-V17. Also looks like we have some correlations around the 0.6.



The highest correlation is V16 and V17 with a correlation of 97.769%. V17 and V18 have a correlation of 97.4628%

From the data we can see that variables 7-18 can be assigned a value from 1-5 (5= will use mass transport). This means that people who put a score of 5 for an increase in gas price by 10% then the rest of the values for him will be 5 as well.

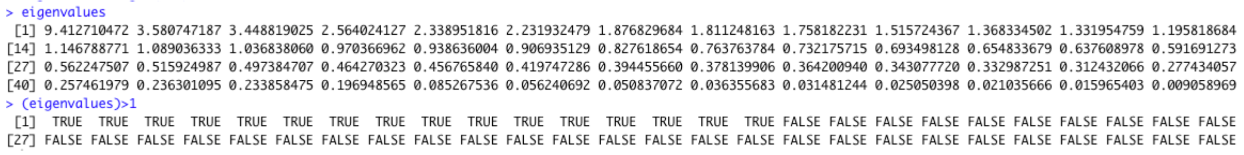
Sample of numerical values to correlation matrix



```
            V12          V13          V14          V15          V16          V17          V18          V19
V1   0.2642141086  0.236665995  0.219459988  0.1988864073  0.174974651  0.157471618  0.1533452881  0.193542892
V2  -0.2020505713 -0.196240752 -0.161574162 -0.1499804763 -0.137905666 -0.111825052 -0.0951254445 -0.089190429
V3  -0.1650515531 -0.159642954 -0.119922713 -0.1103921652 -0.101388028 -0.072707460 -0.0718247874 -0.144052579
V4   0.1075936215  0.117707393  0.111789963  0.0938464572  0.078646730  0.056211357  0.0612798092  0.116405821
V5   0.2561118241  0.262430264  0.257111267  0.2287942413  0.225541694  0.199611693  0.2036334659  0.071825702
V6  -0.0484950972 -0.047210739 -0.037706265 -0.0185291446 -0.019136594 -0.011633845 -0.0138528238 -0.073491467
V7   0.4432902596  0.383604182  0.314959827  0.2604250215  0.194018670  0.135488119  0.1006543496  0.102556152
V8   0.5572087089  0.497969888  0.426230520  0.3697073683  0.301123928  0.235963519  0.1918985578  0.089388015
V9   0.7460178785  0.692931269  0.628820575  0.5779222211  0.509974253  0.445075166  0.3915263960  0.088929985
V10  0.8592432587  0.823445934  0.762698010  0.7142236244  0.659036623  0.592504292  0.5354181812  0.084730983
V11  0.9301971164  0.902163092  0.859556569  0.8166582544  0.773288093  0.706183351  0.6557937344  0.103967546
V12  1.0000000000  0.961296744  0.927564399  0.8995695063  0.857365096  0.809724290  0.7656442621  0.040819753
V13  0.9612967440  1.000000000  0.959853496  0.9395668083  0.902350865  0.857544833  0.8125298179  0.043072723
V14  0.9275643992  0.959853496  1.000000000  0.9718704247  0.945841884  0.907930932  0.8672640959  0.075878860
V15  0.8995695063  0.939566808  0.971870425  1.0000000000  0.971265765  0.943045120  0.9079698663  0.081442933
V16  0.8573650959  0.902350865  0.945841884  0.9712657649  1.000000000  0.977698367  0.9456173599  0.083266677
V17  0.8097242902  0.857544833  0.907930932  0.9430451196  0.977698367  1.000000000  0.9746286304  0.089795408
V18  0.7656442621  0.812529818  0.867264096  0.9079698663  0.945617360  0.974628630  1.0000000000  0.082519462
V19  0.0408197526  0.043072723  0.075878860  0.0814429325  0.083266677  0.089795408  0.0825194624  1.000000000
```

Step 2

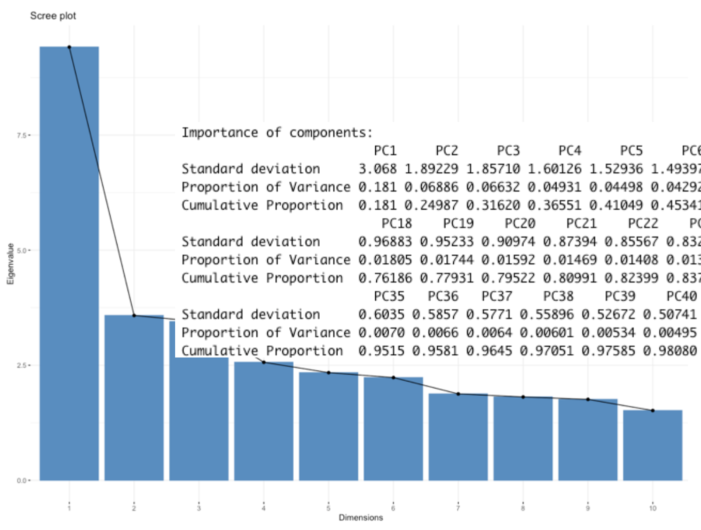
These are the eigenvalues we get, and we have 16 that are above the value of 1.



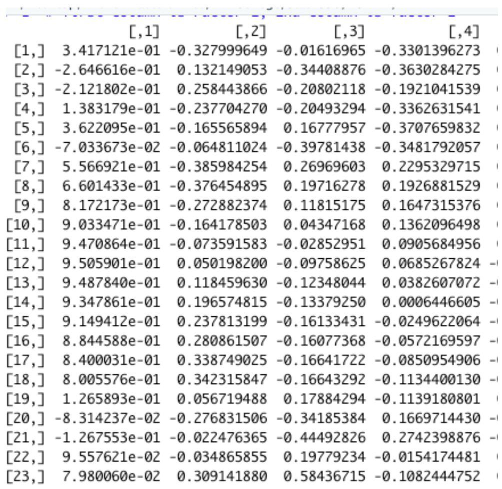
```
> eigenvalues
 [1] 9.412710472 3.580747187 3.448819025 2.564024127 2.338951816 2.231932479 1.876829684 1.811248163 1.758182231 1.515724367 1.368334502 1.331954759 1.195818684
[14] 1.146788771 1.089036333 1.036838060 0.970366962 0.938636004 0.906935129 0.827618654 0.763763784 0.732175715 0.693498128 0.654833679 0.637608978 0.591691273
[27] 0.562247507 0.515924987 0.497384707 0.464270323 0.456765840 0.419747286 0.394455660 0.378139906 0.364200940 0.343077720 0.332987251 0.312432066 0.277434057
[40] 0.257461979 0.236301095 0.233858475 0.196948565 0.085267536 0.056240692 0.050837072 0.036355683 0.031481244 0.025050398 0.021035666 0.015965403 0.009058969
> (eigenvalues)>1
 [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[27] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

That means we can set our M to 16 but we still may be able to get away with using less.

We can simply make a Scree plot and see how many factors we want. We also can see the total cumulative proportion for each pc component. We want to get a decent amount of the data so I will pick 15 factors (m = 15). We will capture 70% of the data.



```
Importance of components:
                         PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9    PC10    PC11    PC12   PC13    PC14    PC15    PC16    PC17
Standard deviation     3.068 1.89229 1.85710 1.60126 1.52936 1.49397 1.36997 1.34583 1.32596 1.23115 1.16976 1.15410 1.0935 1.07088 1.04357 1.01825 0.98507
Proportion of Variance 0.181 0.06886 0.06632 0.04931 0.04498 0.04292 0.03609 0.03483 0.03381 0.02915 0.02631 0.02561 0.0230 0.02205 0.02094 0.01994 0.01866
Cumulative Proportion  0.181 0.24987 0.31620 0.36551 0.41049 0.45341 0.48950 0.52433 0.55814 0.58729 0.61361 0.63922 0.6622 0.68427 0.70521 0.72515 0.74381
                          PC18    PC19    PC20    PC21    PC22    PC23    PC24    PC25    PC26    PC27    PC28    PC29    PC30    PC31    PC32    PC33    PC34
Standard deviation     0.96883 0.95233 0.90974 0.87394 0.85567 0.83277 0.80922 0.79850 0.76921 0.74983 0.71828 0.70526 0.68137 0.67584 0.64788 0.62806 0.61493
Proportion of Variance 0.01805 0.01744 0.01592 0.01469 0.01408 0.01334 0.01259 0.01226 0.01138 0.01081 0.00992 0.00957 0.00893 0.00878 0.00807 0.00759 0.00727
Cumulative Proportion  0.76186 0.77931 0.79522 0.80991 0.82399 0.83733 0.84992 0.86218 0.87356 0.88437 0.89429 0.90386 0.91279 0.92157 0.92964 0.93723 0.94450
                         PC35   PC36   PC37    PC38    PC39    PC40    PC41   PC42    PC43    PC44    PC45    PC46   PC47    PC48    PC49   PC50    PC51    PC52
Standard deviation     0.6035 0.5857 0.5771 0.55896 0.52672 0.50741 0.48611 0.4836 0.44379 0.29201 0.23715 0.22547 0.1907 0.17743 0.15827 0.1450 0.12635 0.09518
Proportion of Variance 0.0070 0.0066 0.0064 0.00601 0.00534 0.00495 0.00454 0.0045 0.00379 0.00164 0.00108 0.00098 0.0007 0.00061 0.00048 0.0004 0.00031 0.00017
Cumulative Proportion  0.9515 0.9581 0.9645 0.97051 0.97585 0.98080 0.98534 0.9898 0.99363 0.99527 0.99635 0.99733 0.9980 0.99863 0.99911 0.9995 0.99983 1.00000
```



```
               [,1]         [,2]        [,3]          [,4]
 [1,]  3.417121e-01 -0.327999649 -0.01616965 -0.3301396273
 [2,] -2.646616e-01  0.132149053 -0.34408876 -0.3630284275
 [3,] -2.121802e-01  0.258443866 -0.20802118 -0.1921041539
 [4,]  1.383179e-01 -0.237704270 -0.20493294 -0.3362631541
 [5,]  3.622095e-01 -0.165565894  0.16777957 -0.3707659832
 [6,] -7.033673e-02 -0.064811024 -0.39781438 -0.3481792057
 [7,]  5.566921e-01 -0.385984254  0.26969603  0.2295329715
 [8,]  6.601433e-01 -0.376454895  0.19716278  0.1926881529
 [9,]  8.172173e-01 -0.272882374  0.11815175  0.1647315376
[10,]  9.033471e-01 -0.164178503  0.04347168  0.1362096498
[11,]  9.470864e-01 -0.073591583 -0.02852951  0.0905684956
[12,]  9.505901e-01  0.050198200 -0.09758625  0.0685267824
[13,]  9.487840e-01  0.118459630 -0.12348044  0.0382607072
[14,]  9.347861e-01  0.196574815 -0.13379250  0.0006446605
[15,]  9.149412e-01  0.237813199 -0.16133431 -0.0249622064
[16,]  8.844588e-01  0.280861507 -0.16077368 -0.0572169597
[17,]  8.400031e-01  0.338749025 -0.16641722 -0.0850954906
[18,]  8.005576e-01  0.342315847 -0.16643292 -0.1134400130
[19,]  1.265893e-01  0.056719488  0.17884294 -0.1139180801
[20,] -8.314237e-02 -0.276831506 -0.34185384  0.1669714430
[21,] -1.267553e-01 -0.022476365 -0.44492826  0.2742398876
[22,]  9.557621e-02 -0.034865855  0.19779234 -0.0154174481
[23,]  7.980060e-02  0.309141880  0.58436715 -0.1082444752
```

Step 3

These are our loading components.

Step 4

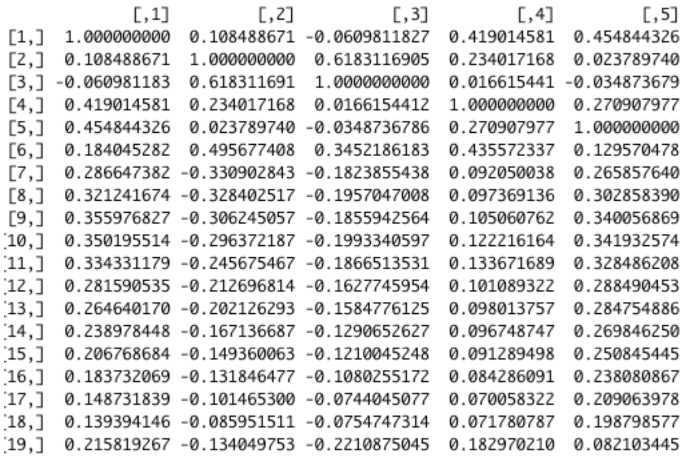
To get the first common value we do (3.417^2)+(0.327^2)+(0.0161^2)….. which gives us 0.67

To find the unique variances all we do is subtract 1 from each common value.

1 - 0.67 = 0.33

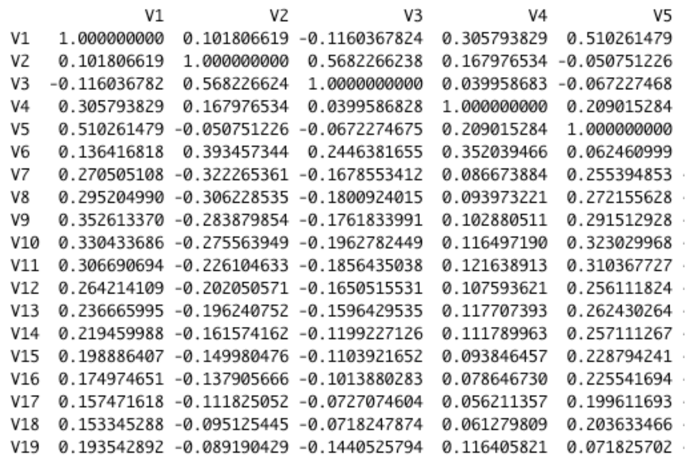
```
> common
 [1] 0.6713460 0.7335749 0.7208371 0.6104870 0.4863770 0.6375862 0.9168016 0.9466260 0.9405295 0.9294951 0.9307163 0.9301825 0.9491527 0.9604116 0.9686219
[16] 0.9715297 0.9563912 0.9181530 0.6984342 0.6785832 0.5831071 0.5924384 0.5534754 0.7674115 0.5716960 0.5585732 0.5096847 0.5476569 0.6757075 0.7091524
[31] 0.6306638 0.6297044 0.4488702 0.5893098 0.7063158 0.4739178 0.7705025 0.5397362 0.4952608 0.7563440 0.6662261 0.6346285 0.5378191 0.7265698 0.5754243
[46] 0.7433968 0.6884371 0.7185115 0.6058525 0.7302105 0.6359192 0.7427422
> unique
 [1] 0.32865405 0.26642510 0.27916291 0.38951299 0.51362303 0.36241382 0.08319842 0.05337398 0.05947049 0.07050487 0.06928369 0.06981750 0.05084726 0.03958842
[15] 0.03137815 0.02847029 0.04360881 0.08184704 0.30156577 0.32141679 0.41689294 0.40756160 0.44652457 0.23258848 0.42830398 0.44142679 0.49031525 0.45234310
[29] 0.32429252 0.29084765 0.36933624 0.37029558 0.55112981 0.41069018 0.29368421 0.52608223 0.22949753 0.46026377 0.50473922 0.24365604 0.33377385 0.36537146
[43] 0.46218094 0.27343025 0.42457569 0.25660317 0.31156291 0.28148848 0.39414752 0.26978950 0.36408077 0.25725779
```

Step 5



```
             [,1]         [,2]          [,3]        [,4]         [,5]
[1,]  1.000000000  0.108488671 -0.0609811827 0.419014581  0.454844326
[2,]  0.108488671  1.000000000  0.6183116905 0.234017168  0.023789740
[3,] -0.060981183  0.618311691  1.0000000000 0.016615441 -0.034873679
[4,]  0.419014581  0.234017168  0.0166154412 1.000000000  0.270907977
[5,]  0.454844326  0.023789740 -0.0348736786 0.270907977  1.000000000
[6,]  0.184045282  0.495677408  0.3452186183 0.435572337  0.129570478
[7,]  0.286647382 -0.330902843 -0.1823855438 0.092050038  0.265857640
[8,]  0.321241674 -0.328402517 -0.1957047008 0.097369136  0.302858390
[9,]  0.355976827 -0.306245057 -0.1855942564 0.105060762  0.340056869
10,]  0.350195514 -0.296372187 -0.1993340597 0.122216164  0.341932574
11,]  0.334331179 -0.245675467 -0.1866513531 0.133671689  0.328486208
12,]  0.281590535 -0.212696814 -0.1627745954 0.101089322  0.288490453
13,]  0.264640170 -0.202126293 -0.1584776125 0.098013757  0.284754886
14,]  0.238978448 -0.167136687 -0.1290652627 0.096748747  0.269846250
15,]  0.206768684 -0.149360063 -0.1210045248 0.091289498  0.250845445
16,]  0.183732069 -0.131846477 -0.1080255172 0.084286091  0.238080867
17,]  0.148731839 -0.101465300 -0.0744045077 0.070058322  0.209063978
18,]  0.139394146 -0.085951511 -0.0754747314 0.071780787  0.198798577
19,]  0.215819267 -0.134049753 -0.2210875045 0.182970210  0.082103445
```

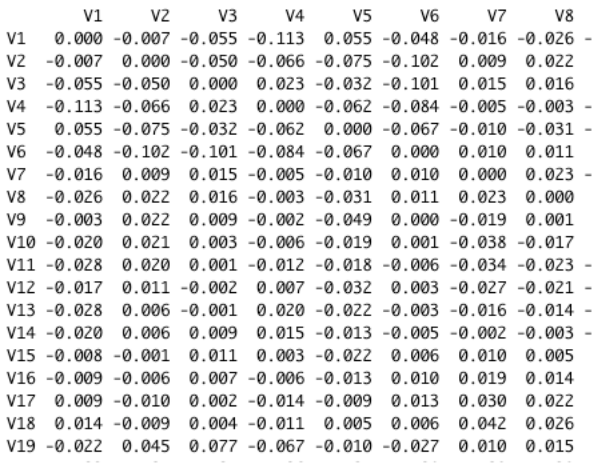
Recreated correlation matrix



```
              V1           V2            V3          V4           V5
V1   1.000000000  0.101806619 -0.1160367824 0.305793829  0.510261479
V2   0.101806619  1.000000000  0.5682266238 0.167976534 -0.050751226
V3  -0.116036782  0.568226624  1.0000000000 0.039958683 -0.067227468
V4   0.305793829  0.167976534  0.0399586828 1.000000000  0.209015284
V5   0.510261479 -0.050751226 -0.0672274675 0.209015284  1.000000000
V6   0.136416818  0.393457344  0.2446381655 0.352039466  0.062460999
V7   0.270505108 -0.322265361 -0.1678553412 0.086673884  0.255394853
V8   0.295204990 -0.306228535 -0.1800924015 0.093973221  0.272155628
V9   0.352613370 -0.283879854 -0.1761833991 0.102880511  0.291512928
V10  0.330433686 -0.275563949 -0.1962782449 0.116497190  0.323029968
V11  0.306690694 -0.226104633 -0.1856435038 0.121638913  0.310367727
V12  0.264214109 -0.202050571 -0.1650515531 0.107593621  0.256111824
V13  0.236665995 -0.196240752 -0.1596429535 0.117707393  0.262430264
V14  0.219459988 -0.161574162 -0.1199227126 0.111789963  0.257111267
V15  0.198886407 -0.149980476 -0.1103921652 0.093846457  0.228794241
V16  0.174974651 -0.137905666 -0.1013880283 0.078646730  0.225541694
V17  0.157471618 -0.111825052 -0.0727074604 0.056211357  0.199611693
V18  0.153345288 -0.095125445 -0.0718247874 0.061279809  0.203633466
V19  0.193542892 -0.089190429 -0.1440525794 0.116405821  0.071825702
```

This is the original matrix

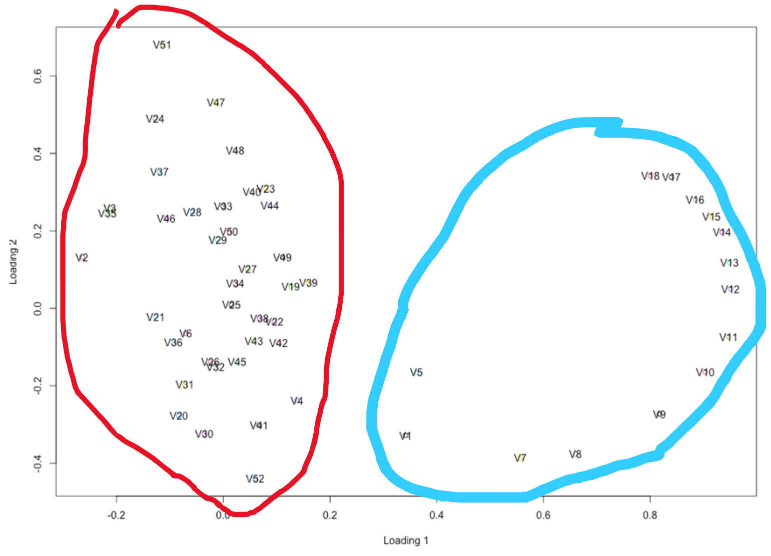
Step 6



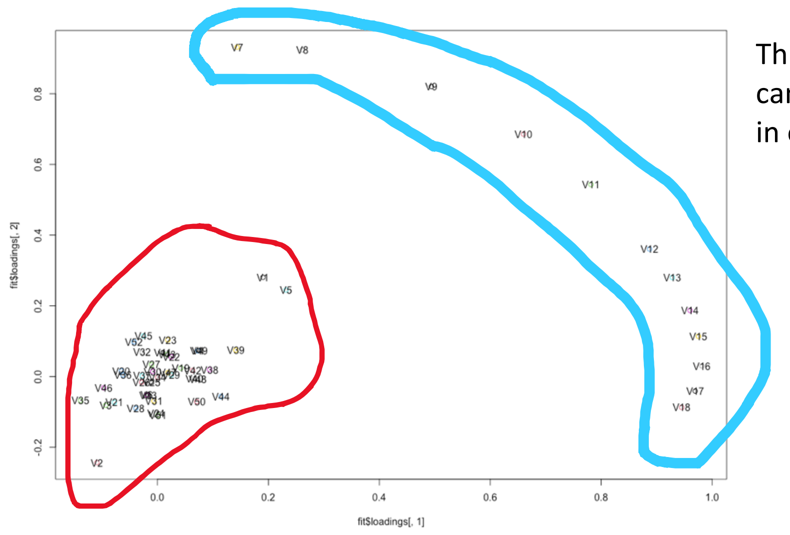
```
        V1     V2     V3     V4     V5     V6     V7     V8
V1   0.000 -0.007 -0.055 -0.113  0.055 -0.048 -0.016 -0.026 -
V2  -0.007  0.000 -0.050 -0.066 -0.075 -0.102  0.009  0.022
V3  -0.055 -0.050  0.000  0.023 -0.032 -0.101  0.015  0.016
V4  -0.113 -0.066  0.023  0.000 -0.062 -0.084 -0.005 -0.003 -
V5   0.055 -0.075 -0.032 -0.062  0.000 -0.067 -0.010 -0.031 -
V6  -0.048 -0.102 -0.101 -0.084 -0.067  0.000  0.010  0.011
V7  -0.016  0.009  0.015 -0.005 -0.010  0.010  0.000  0.023 -
V8  -0.026  0.022  0.016 -0.003 -0.031  0.011  0.023  0.000
V9  -0.003  0.022  0.009 -0.002 -0.049  0.000 -0.019  0.001
V10 -0.020  0.021  0.003 -0.006 -0.019  0.001 -0.038 -0.017
V11 -0.028  0.020  0.001 -0.012 -0.018 -0.006 -0.034 -0.023 -
V12 -0.017  0.011 -0.002  0.007 -0.032  0.003 -0.027 -0.021 -
V13 -0.028  0.006 -0.001  0.020 -0.022 -0.003 -0.016 -0.014 -
V14 -0.020  0.006  0.009  0.015 -0.013 -0.005 -0.002 -0.003 -
V15 -0.008 -0.001  0.011  0.003 -0.022  0.006  0.010  0.005
V16 -0.009 -0.006  0.007 -0.006 -0.013  0.010  0.019  0.014
V17  0.009 -0.010  0.002 -0.014 -0.009  0.013  0.030  0.022
V18  0.014 -0.009  0.004 -0.011  0.005  0.006  0.042  0.026
V19 -0.022  0.045  0.077 -0.067 -0.010 -0.027  0.010  0.015
```

This correlation matrix shows us the difference from the one we attempted to create. We did a pretty good job at it.

Step 7

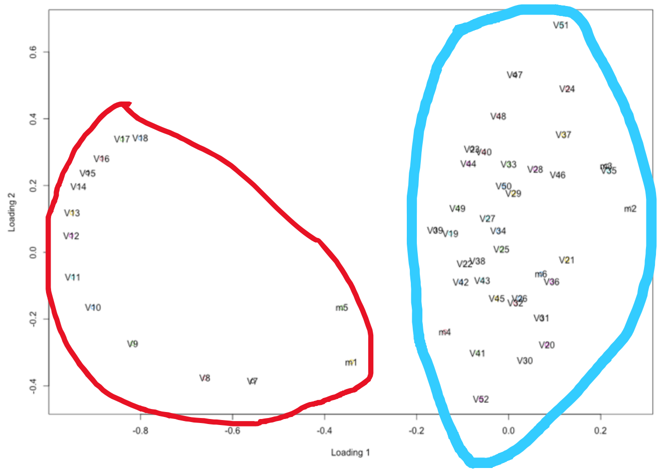


We see that we have to two groups made. The group in blue is made up of one question which is how likely that are to use public transportation if gas prices goes up a certain percentage. I know that I need to deal with the variance for V1-V6 by dividing the entire column by 5. These will scale it down to the other variables because it has 25 levels as a categorical variable.

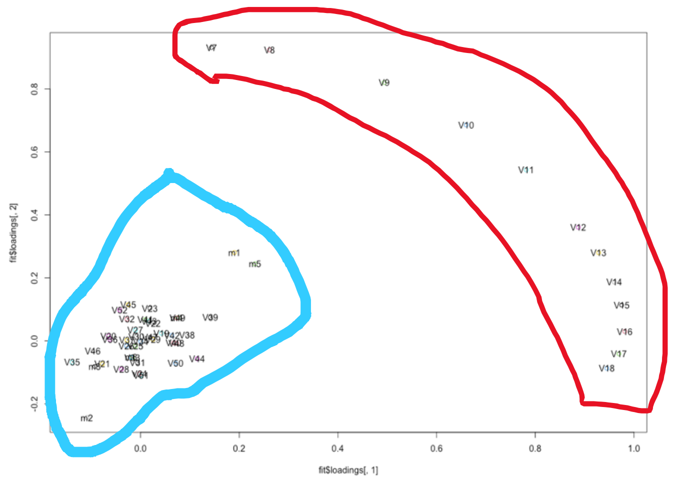


This is when we use varimax function. As you can you can all of the data is clumped together in one pile except V7-V18.

Running through the entire script except this time for V1-V6 I will be dividing each column by 5.



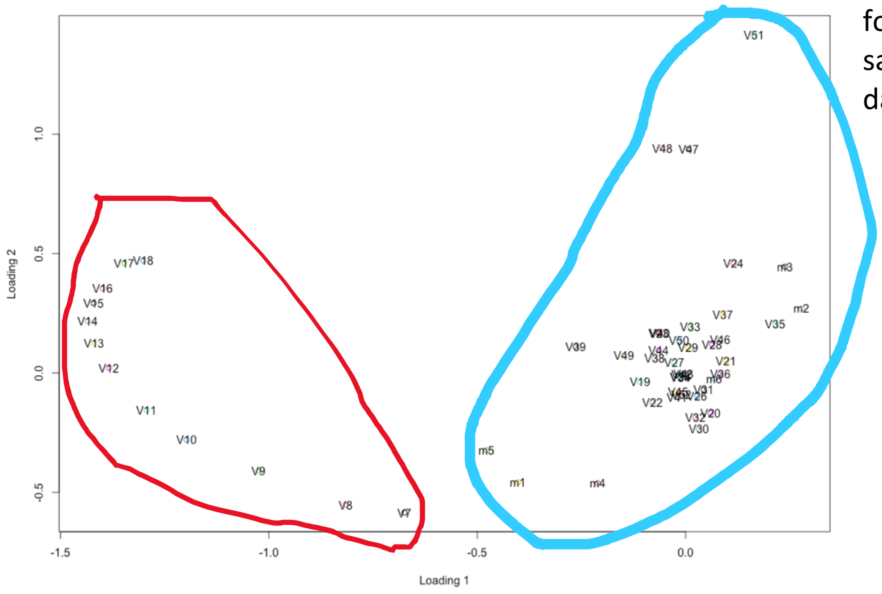
Looks like not much happened. We still get the same grouping. The only thing is now they are switch positions.



This is the graph when using varimax and it is the same as before.

Un-standardize data

All we do is replace the correlation matrix with the covariance matrix. We produce this graph below for without any rotations. I ran into the same problem when setting covar = False. It doesn't seem to be working for me because I produce the same plot as standardized data.


Loading 1
Loading 2
V51
V48
V47
V24
m3
m2
V37
V35
V33
V46
V28
V29
V50
V39
V49
V44
V38
V27
V21
V36
m6
V19
V31
V26
V22
V41
V20
V32
V30
m5
m1
m4
V17
V18
V16
V15
V14
V13
V12
V11
V10
V9
V8
V7