FACTOR ANALYSIS

Read in the Big5 Personality test dataset, which contains likert scale responses (five point scale where 1=Disagree, 3=Neutral, 5=Agree. 0 = missing) on 50 different questions in columns 8 through 57. The questions, labeled E1-E10 (E=extroversion), N1-N10 (N=neuroticism), A1-A10 (A=agreeableness), C1-C10 (C=conscientiousness), and O1-O10 (O=openness) all attempt to measure 5 key angles of human personality. The first 7 columns contain demographic information coded as follows:

- **Race** Chosen from a drop down menu.
  - 1=Mixed Race
  - 2=Arctic (Siberian, Eskimo)
  - 3=Caucasian (European)
  - 4=Caucasian (Indian)
  - 5=Caucasian (Middle East)
  - 6=Caucasian (North African, Other)
  - 7=Indigenous Australian
  - 8=Native American
  - 9=North East Asian (Mongol, Tibetan, Korean Japanese, etc)
  - 10=Pacific (Polynesian, Micronesian, etc)
  - 11=South East Asian (Chinese, Thai, Malay, Filipino, etc)
  - 12=West African, Bushmen, Ethiopian
  - 13=Other (0=missed)
- **Age** Entered as text (individuals reporting age < 13 were not recorded)
- **Engnat** Response to "is English your native language?"
  - 1=yes
  - 2=no
  - 0=missing
- **Gender** Chosen from a drop down menu
  - 1=Male

  - 2=Female
  - 3=Other
  - 0=missing

- **Hand** "What hand do you use to write with?"
  - 1=Right
  - 2=Left
  - 3=Both
  - 0=missing

```
> options(digits=2)
> load("big5.Rdata")
```

To perform the same analysis we did in SAS, we want to use Correlation PCA and rotate the axes with a varimax transformation. We will start by performing the PCA. We need to set the option scale=T to perform PCA on the correlation matrix rather than the default covariance matrix. We will only compute the first 5 principal components because we have 5 personality traits we are trying to measure. We could also compute more than 5 and take the number of components with eigenvalues >1 to match the default output in SAS (without n=5 option).

```
> options(digits=5)
> pca.out = prcomp(big5[,8:57], rank = 5, scale = T)
```

Remember the only difference between the default PROC PRINCOMP output and the default PROC FACTOR output in SAS was the fact that the eigenvectors in PROC PRINCOMP were normalized to be unit vectors and the factor vectors in PROC FACTOR were those same eigenvectors scaled by the square roots of the eigenvalues. So we want to multiply each eigenvector column output in pca.out$rotation (recall this is the loading matrix or matrix of eigenvectors) by the square root of the corresponding eigenvalue given in pca.out$sdev. You'll recall that multiplying a matrix by a diagonal matrix on the right has the effect of scaling the columns of the matrix. So we'll just make a diagonal matrix, **S** with diagonal elements from the pca.out$sdev vector and scale the columns of the pca.out$rotation matrix. Similarly, the coordinates of the data along each component then need to be *divided* by the standard deviation to cancel out this effect of lengthening the axis. So again we will multiply by a diagonal matrix to perform this scaling, but this time, we use the diagonal matrix $\mathbf{S}^{-1}$ =diag(1/(pca.out$sdev)).

Matrix multiplication in R is performed with the %% operator.

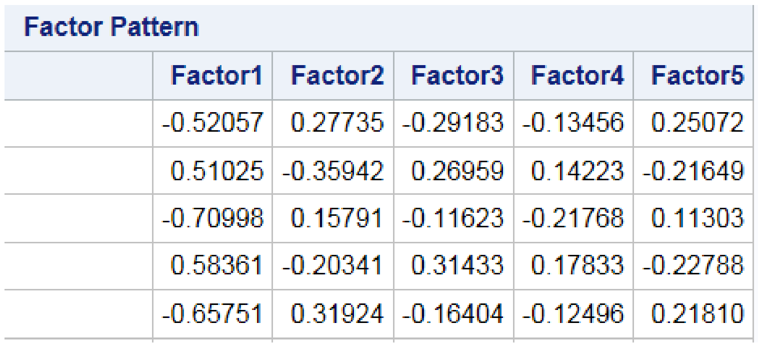
```
> fact.loadings = pca.out$rotation[,1:5] %*% diag(pca.out$sdev[1:5])
> fact.scores = pca.out$x[,1:5] %*%diag(1/pca.out$sdev[1:5])
> # PRINT OUT THE FIRST 5 ROWS OF EACH MATRIX FOR CONFIRMATION.
> fact.loadings[1:5,1:5]
```

```
        [,1]     [,2]     [,3]     [,4]     [,5]
E1 -0.52057  0.27735 -0.29183  0.13456 -0.25072
E2  0.51025 -0.35942  0.26959 -0.14223  0.21649
E3 -0.70998  0.15791 -0.11623  0.21768 -0.11303
E4  0.58361 -0.20341  0.31433 -0.17833  0.22788
E5 -0.65751  0.31924 -0.16404  0.12496 -0.21810
```

```
> fact.scores[1:5,1:5]
```

```
        [,1]     [,2]     [,3]      [,4]      [,5]
[1,] -2.53286 -1.16617 0.276244  0.043229 -0.069518
[2,]  0.70216 -1.22761 1.095383  1.615919 -0.562371
[3,] -0.12575  1.33180 1.525208 -1.163062 -2.949501
[4,]  1.29926  1.17736 0.044168 -0.784411  0.148903
[5,] -0.37359  0.47716 0.292680  1.233652  0.406582
```

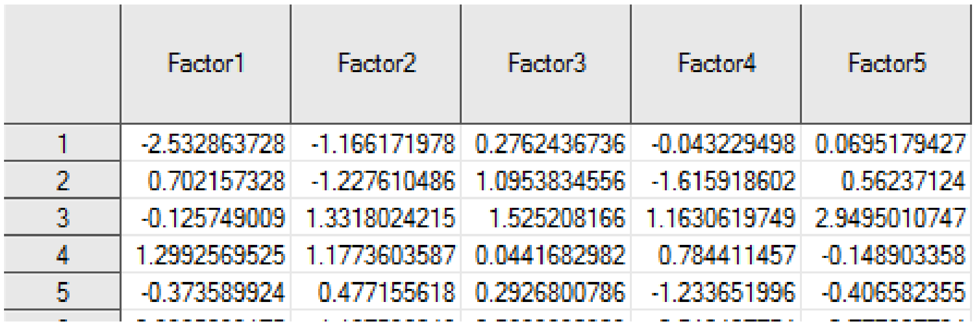
This should match the output from SAS and it does. Remember these columns are unique up to a sign, so you'll see factor 4 does not have the same sign in both software outputs. This is not cause for concern.



Factor Pattern

	Factor1	Factor2	Factor3	Factor4	Factor5
	-0.52057	0.27735	-0.29183	-0.13456	0.25072
	0.51025	-0.35942	0.26959	0.14223	-0.21649
	-0.70998	0.15791	-0.11623	-0.21768	0.11303
	0.58361	-0.20341	0.31433	0.17833	-0.22788
	-0.65751	0.31924	-0.16404	-0.12496	0.21810

Figure 1: Default (Unrotated) Factor Loadings Output by SAS



	Factor1	Factor2	Factor3	Factor4	Factor5
1	-2.532863728	-1.166171978	0.2762436736	-0.043229498	0.0695179427
2	0.702157328	-1.227610486	1.0953834556	-1.615918602	0.56237124
3	-0.125749009	1.3318024215	1.525208166	1.1630619749	2.9495010747
4	1.2992569525	1.1773603587	0.0441682982	0.784411457	-0.148903358
5	-0.373589924	0.477155618	0.2926800786	-1.233651996	-0.406582355

Figure 2: Default (Unrotated) Factor Scores Output by SAS

The next task we may want to undertake is a rotation of the factor axes according to the varimax procedure. The most simple way to go about this is to use the varimax() function to find the optimal rotation of the eigenvectors in the matrix pca.out$rotation. The varimax() function outputs both the new set of axes in the matrix called loadings and the rotation matrix (rotmat) which performs the rotation from the original principal component axes to the new axes. (i.e. if $\mathbf{V}$ contains the old axes as columns and $\hat{\mathbf{V}}$ contains the new axes and $\mathbf{R}$ is the rotation matrix then $\hat{\mathbf{V}} = \mathbf{VR}$.) That rotation matrix can be used to perform the same rotation on the scores of the observations. If the matrix $\mathbf{U}$ contains the scores for each observation, then the rotated scores $\hat{\mathbf{U}}$ are found by $\hat{\mathbf{U}} = \mathbf{UR}$

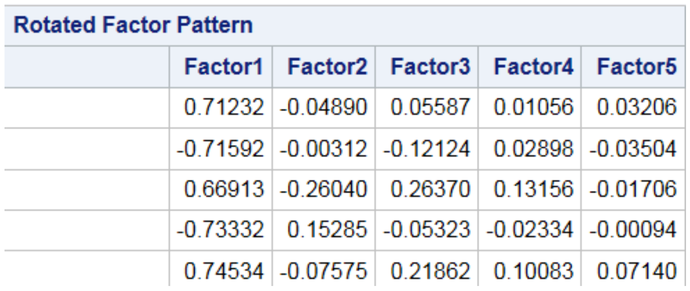
```
> varimax.out = varimax(fact.loadings)
> rotated.fact.loadings = fact.loadings %*% varimax.out$rotmat
> rotated.fact.scores = fact.scores %*% varimax.out$rotmat
> # PRINT OUT THE FIRST 5 ROWS OF EACH MATRIX FOR CONFIRMATION.
> rotated.fact.loadings[1:5,]
```

```
       [,1]       [,2]      [,3]        [,4]      [,5]
E1 -0.71232 -0.0489043  0.010596 -0.03206926  0.055858
E2  0.71592 -0.0031185  0.028946  0.03504236 -0.121241
E3 -0.66912 -0.2604049  0.131609  0.01704690  0.263679
E4  0.73332  0.1528552 -0.023367  0.00094685 -0.053219
E5 -0.74534 -0.0757539  0.100875 -0.07140722  0.218602
```

```
> rotated.fact.scores[1:5,]
```

```
         [,1]     [,2]     [,3]     [,4]       [,5]
[1,] -1.09083 -2.04516  1.40699 -0.38254  0.5998386
[2,]  0.85718 -0.19268  1.07708  2.03665 -0.2178616
[3,] -0.92344  2.58761  2.43566 -0.80840 -0.1833138
[4,]  0.61935  1.53087 -0.79225 -0.59901 -0.0064665
[5,] -0.39495 -0.10893 -0.24892  0.99744  0.9567712
```

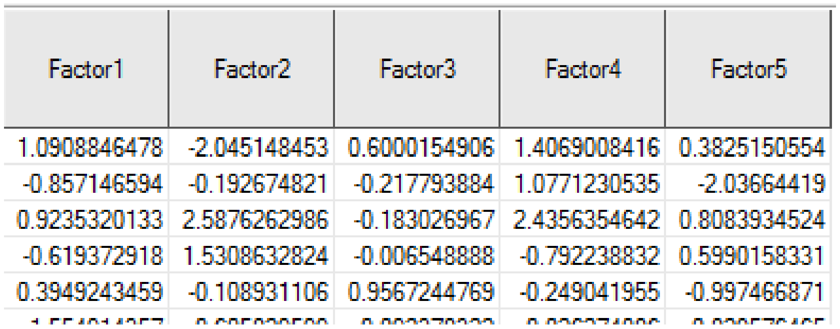
And again we can see that these line up with our SAS Rotated output, **however** the order does not have to be the same! SAS conveniently reorders the columns according to the variance of the data along that new direction. Since we have not done that in R, the order of the columns is not the same! Factors 1 and 2 are the same in both outputs, but SAS Factor 3 = R Factor 4 and SAS Factor 5 = (-1)* R Factor 4. The coordinates are switched too so nothing changes in our interpretation. Remember, when you rotate factors, you no longer keep the notion that the "first vector" explains the most variance unless you reorder them so that is true (like SAS does).



Rotated Factor Pattern

	Factor1	Factor2	Factor3	Factor4	Factor5
	0.71232	-0.04890	0.05587	0.01056	0.03206
	-0.71592	-0.00312	-0.12124	0.02898	-0.03504
	0.66913	-0.26040	0.26370	0.13156	-0.01706
	-0.73332	0.15285	-0.05323	-0.02334	-0.00094
	0.74534	-0.07575	0.21862	0.10083	0.07140

Figure 3: Rotated Factor Loadings Output by SAS



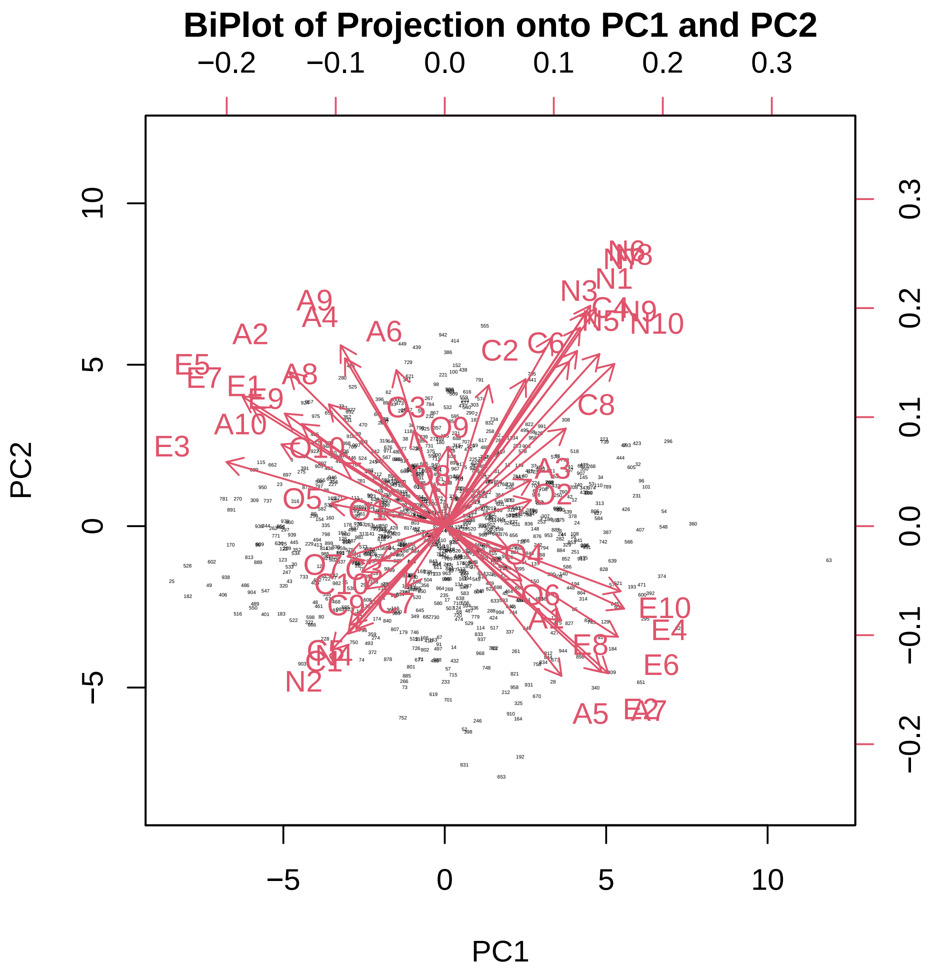
Factor1	Factor2	Factor3	Factor4	Factor5
1.0908846478	-2.045148453	0.6000154906	1.4069008416	0.3825150554
-0.857146594	-0.192674821	-0.217793884	1.0771230535	-2.03664419
0.9235320133	2.5876262986	-0.183026967	2.4356354642	0.8083934524
-0.619372918	1.5308632824	-0.006548888	-0.792238832	0.5990158331
0.3949243459	-0.108931106	0.9567244769	-0.249041955	-0.997466871
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Figure 4: Rotated Factor Scores Output by SAS

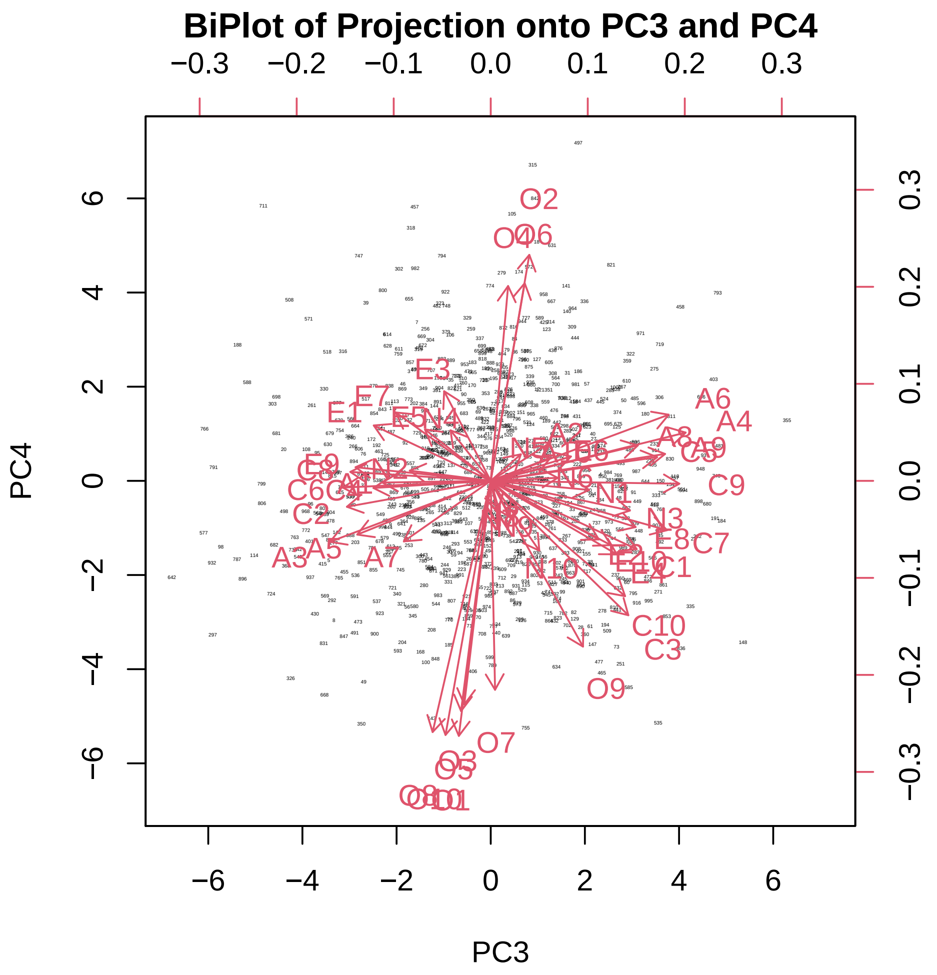
Visualizing Rotation in BiPlots

Let's start with a peek at BiPlots of the first 2 *pairs* of principal component loadings, prior to rotation. Notice that here I'm not going to bother with any scaling of the factor loadings as I'm not interested in forcing my output to look like SAS's output. I'm also downsampling the observations because 20,000 is far to many to plot.

```
> biplot(pca.out$x[sample(1:19719,1000),1:2],
+        pca.out$rotation[,1:2],
+        cex=c(0.2,1),
+        main = 'BiPlot of Projection onto PC1 and PC2')
```

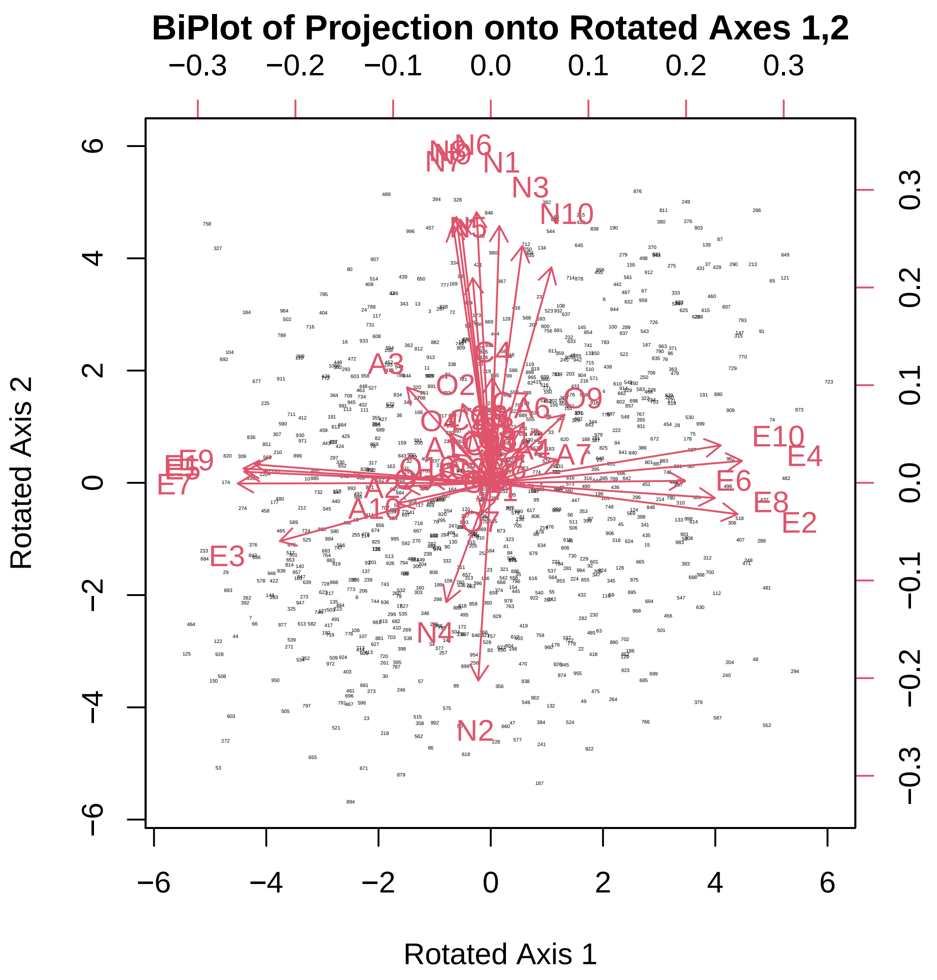


```
> biplot(pca.out$x[sample(1:19719,1000),3:4],
+        pca.out$rotation[,3:4],
+        cex=c(0.2,1),
+        main = 'BiPlot of Projection onto PC3 and PC4')
```

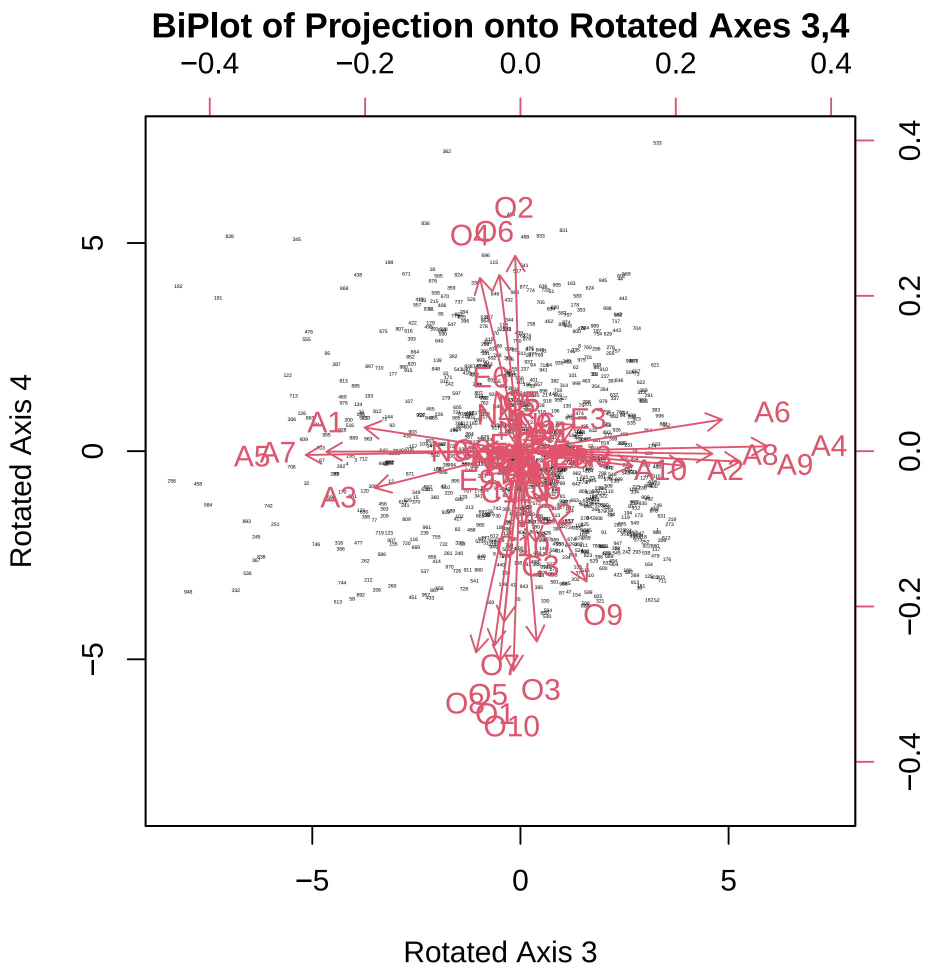


Let's see what happens to these biplots after rotation:

```
> vmax = varimax(pca.out$rotation)
> newscores = pca.out$x%*%vmax$rotmat
> biplot(newscores[sample(1:19719,1000),1:2],
+        vmax$loadings[,1:2],
+        cex=c(0.2,1),
+        main = 'BiPlot of Projection onto Rotated Axes 1,2',
+        xlab = 'Rotated Axis 1',
+        ylab = 'Rotated Axis 2')
```



```
> biplot(newscores[sample(1:19719,1000),3:4],
+        vmax$loadings[,3:4],
+        cex=c(0.2,1),
+        main = 'BiPlot of Projection onto Rotated Axes 3,4',
+        xlab = 'Rotated Axis 3',
+        ylab = 'Rotated Axis 4')
```



After the rotation, we can see the BiPlots tell a more distinct story. The extroversion questions line up along rotated axes 1, neuroticism along rotated axes 2, and agreeableness and openness are reflected in rotated axes 3 and 4 respectively.