Factor Analysis with R

Code

This notebook shows how to perform Factor Analysis with R and how to use the output to construct a perceptual map. To perform this analysis you need to install these R packages: 1. nFactors 2. ggplot2 3. ggrepel 4. psych

Reading and outputing data

The dataset includes data on ten students' performance on six tests, graded on a 100point scale. The aim of factor analysis is to reduce the dimensionality of the 10 by 6 data matrix of test scores into a smaller dimensionality (e.g., 10 by 2). The rows in this dataset are 10 students and the columns (ie., the variables) are test scores for six subject areas: Grammar, Spelling, Composition, Algebra, Geometry, and Trigonometry. The first analysis decision in factor analysis is to determine the number of factors to retain. The second decision is to understand the factor structure of the data.

Hide

```
test_scores = read.csv(file = "testscoresdata.csv")
test_scores
```

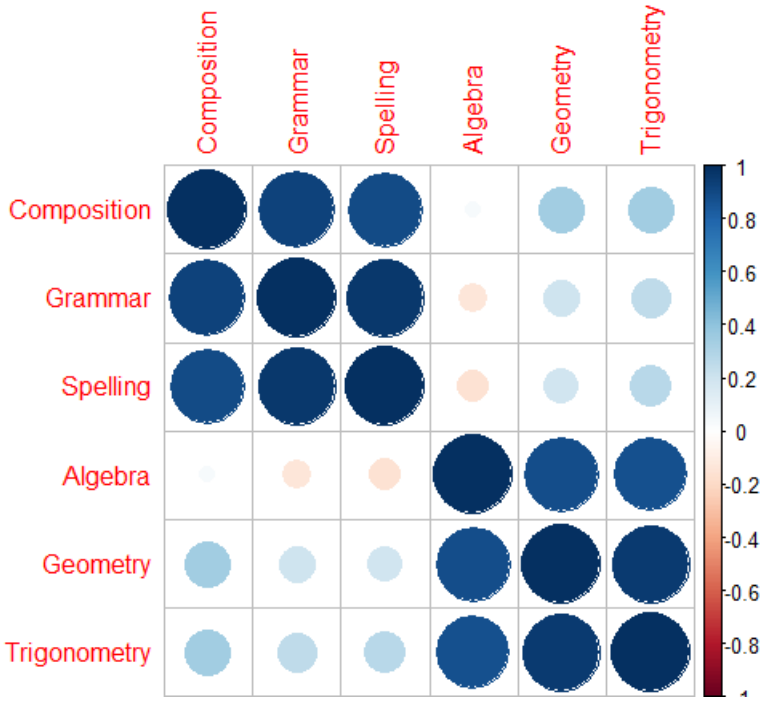
Determining the Number of Factors

A simple way to determine the number of factors underlying a dataset is to inspect the correlation matrix.

Inspecting the correlation matrix

Hide

```
library(corrplot)
corrplot(cor(test_scores[,2:7]), order = "hclust")
```



We see from this plot that the test scores group into two clusters of highly correlated variables: verbal tests (Composition, Grammar, Spelling) and math tests (Algebra, Geometry, and Trigonometry). The size of the circles in the graph above represents strength of correlation between pairs of variables.

The eigenvalue > 1 criterion

This criterion suggests two factors. In the output below, the first two eigenvalues are greater than 1: the first is 3.33 and the second is 2.43. The third eigenvalue is 0.137 is less than 1. Thus we can retain two factors. Note that with six variables we can extract a maximum of six factors.

Hide

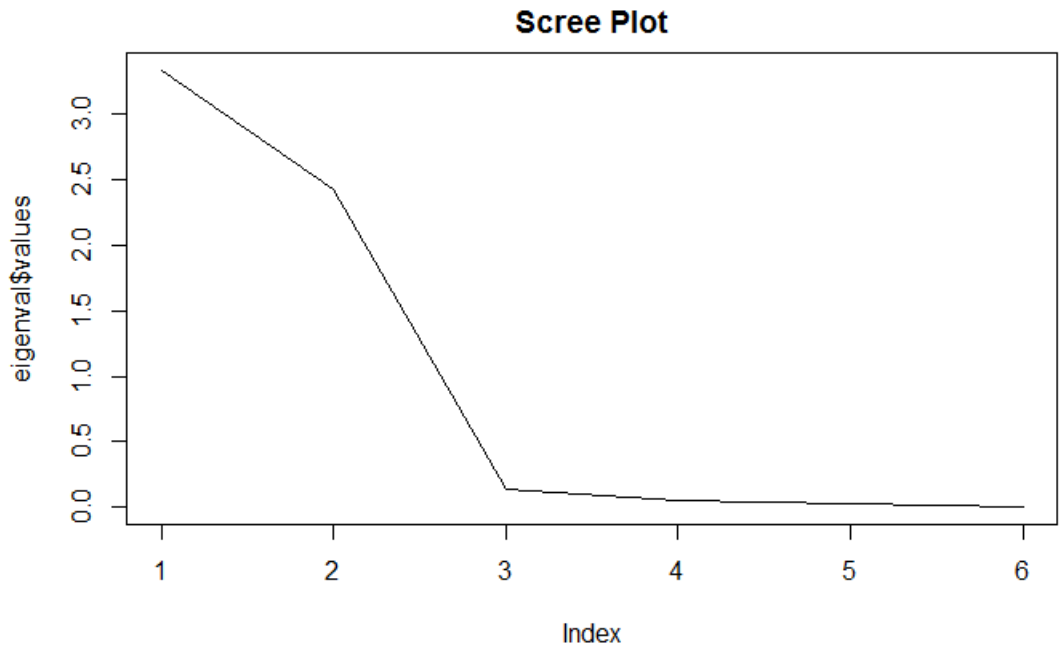
```
eigenval=eigen(cor(test_scores[,2:7]))
eigenval$values
```

```
[1] 3.334873852 2.431236985 0.137298334 0.057348762 0.031841202 0.007400866
```

The scree plot

Hide

```
plot(eigenval$values, main="Scree Plot", type="l")
```



The scree plot also suggests two factors. (This is a plot of the eigenvalues that are associated with factors 1 to 6.) The figure indicates where additional factors do not add much to explained variance. There is an elbow in the figure at factor 3. This indicates that two factors are sufficient in capturing the variation in the data.

R can perform the eigenvalue and scree plot analyses automatically using nScree() along with two other methods. In the output below, all of these methods suggest 2 factors. Now we are ready to use the nFactor package to determine the number of factors.

Hide

```
library(nFactors)
nScree(test_scores[, 2:7], cor=TRUE) #This function help you determine the number of factors
```

```
  noc naf nparallel nkaiser
1   2   2         2       2
```

The Kaiser criterion (nkaiser) corresponds to the eigenvalue > 1 criterion when factor analysis is applied on the correlation matrix. The acceleration factor criterion (naf) is based on the elbow criterion in the scree plot. Parallel analysis (nparallel) and optimal coordinates (noc) are two other methods. As you can see all four criteria indicated two factors to retain. Now we are ready to run a factor analysis with two factors using the Varimax Rotated Principal Components method.

Hide

```
library(psych)
fit <- principal(test_scores[, 2:7],nfactors=2, rotate="varimax")
```

There are several statistics reported in the output, all are contained in the fit object. For example, fit$values reports the eigenvalues of the correlation matrix.

Hide

```
fit$values #print the eigen values of the correlation matrix
```

```
[1] 3.334873852 2.431236985 0.137298334 0.057348762 0.031841202 0.007400866
```

These are the same eigenvalues discussed above.

Interpreting the two factors

We use the factor loadings to interpret the factors. The factor loadings are the correlations between the variables (i.e., the tests scores) and the factors. In interpreting each factor, underline the variables that load high (i.e., has a high correlation) on the factor and ask what's common between these variables.

Hide

```
fit$loadings # print the factor loading results
```

```
Loadings:
             RC1    RC2
Grammar       0.989
Spelling      0.981
Composition   0.947  0.174
Algebra      -0.166  0.969
Geometry      0.184  0.965
Trigonometry  0.227  0.956

                 RC1   RC2
SS loadings    2.951 2.816
Proportion Var 0.492 0.469
Cumulative Var 0.492 0.961
```

The top panel of the table above reports the factor loading matrix. We can see that Grammar, Spelling, and Composition all load high (have high correlation with) on RC1 (i.e., Factor 1). All of these tests are verbal tests. Thus, we could call Factor 1 Verbal Ability. Similarly, Algebra, Geometry, and Trigonometry all load high on RC2 (i.e., Factor 2). Thus we could name Factor 2 Mathematical Ability. Note that the missing loadings of Grammar and Spelling on RC2 indicate values close to zero.

The bottom panel reports summary fit statistics for the two-factor solution. In this analysis, the variance explained by the first factor is 2.951 and that of the second factor is 2.816. As the sum of the variances of the six variables is six (each variable is normalized to have variance equal to 1), the percent of total variance in the data that is explained by Factor 1 is 49.2% (=2.951/6) and the percent of total variance explained by Factor 2 is 46.9%. Thus the two-factor solution explains 96.1% of the variability of the data.

Factor weights

A factor is a linear (i.e., weighted) combination of the variables. The table below reports the weights used to construct the factor scores.

Hide

```
colnames(fit$weights) = c("Verbal", "Math") # renaming RC1 as "Verbal" factor and RC2 as "Math" factor
fit$weights # print the factor weights
```

```
                 Verbal        Math
Grammar       0.34191828 -0.04456066
Spelling      0.33963570 -0.04646619
Composition   0.31925112  0.01117554
Algebra      -0.11086057  0.36157268
Geometry      0.01083039  0.34104688
Trigonometry  0.02604433  0.33554399
```

Thus the verbal ability (factor) score is computed as follows for each student: Verbal= 0.44 Grammar + 0.33Spelling + 0.31 Composition + … + 0.02Trigonometry.

The math ability score is computed as: Math=-0.04Grammar + …+0.36Algebra + 0.34Geometry + 0.33Trigonometry.

In R, these factor scores are output in the table below.

Factor Scores

Hide

```
colnames(fit$scores) = c("Verbal", "Math") # renaming RC1 as "Verbal" factor and RC2 as "Math" factor
fit$scores #print the factor scores
```

```
          Verbal       Math
 [1,]  0.9153038 -0.5302432
 [2,] -1.4485777  0.9167256
 [3,] -0.2393435 -0.3530964
 [4,]  0.9443561  1.2404663
 [5,] -1.4692481 -1.4411465
 [6,]  0.5122126  0.8352546
 [7,]  1.2581872 -0.5666709
 [8,] -0.9580358  1.2334432
 [9,]  0.0487361 -1.2620155
[10,]  0.4364093 -0.0727173
```

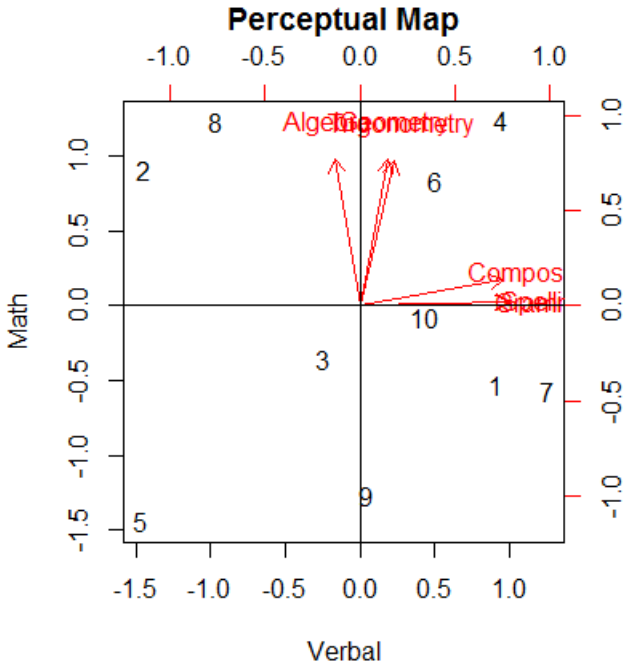
Now we are ready to summarize the whole output in a map.

Hide

```
colnames(fit$scores) = c("Verbal", "Math") # renaming RC1 as "Verbal" factor and RC2 as "Math" factor
biplot(fit$scores, fit$loadings, xlabs=test_scores[,1], main = "Perceptual Map\n\n") # Perceptual map
abline(h=0) #add a horizontal line in the graph
```

Hide

```
abline(v=0) #add a vertical line in the graph
```



The graph above is a joint plot of the factor loading matrix, where the loading of each variable on Verbal and Math factors are represented by red vectors and the factor scores of students by their id numbers.

To produce a better map, you may use this extensive R program.

Hide

```
# Varimax Rotated Principal Components
# retaining 2 factors
library(psych)
fit <- principal(test_scores[, 2:7],nfactors=2, rotate="varimax")
#extracting scores and loadings info from "fit" object, and binding them together in order to plot together
ty = rbind(as.data.frame(unclass(fit$scores)),
           (as.data.frame(unclass(fit$loadings))*3)) #multiplied by scalar for plot
colnames(ty) = c("Verbal", "Math")
#Credit for the plotting function goes to Ben Levine, Ph.D student at Columbia Business School
library(ggplot2) #library for creating visualizations
```

```
Attaching package: <U+393C><U+3E31>ggplot2<U+393C><U+3E32>

The following objects are masked from <U+393C><U+3E31>package:psych<U+393C><U+3E32>:

    %+%, alpha
```

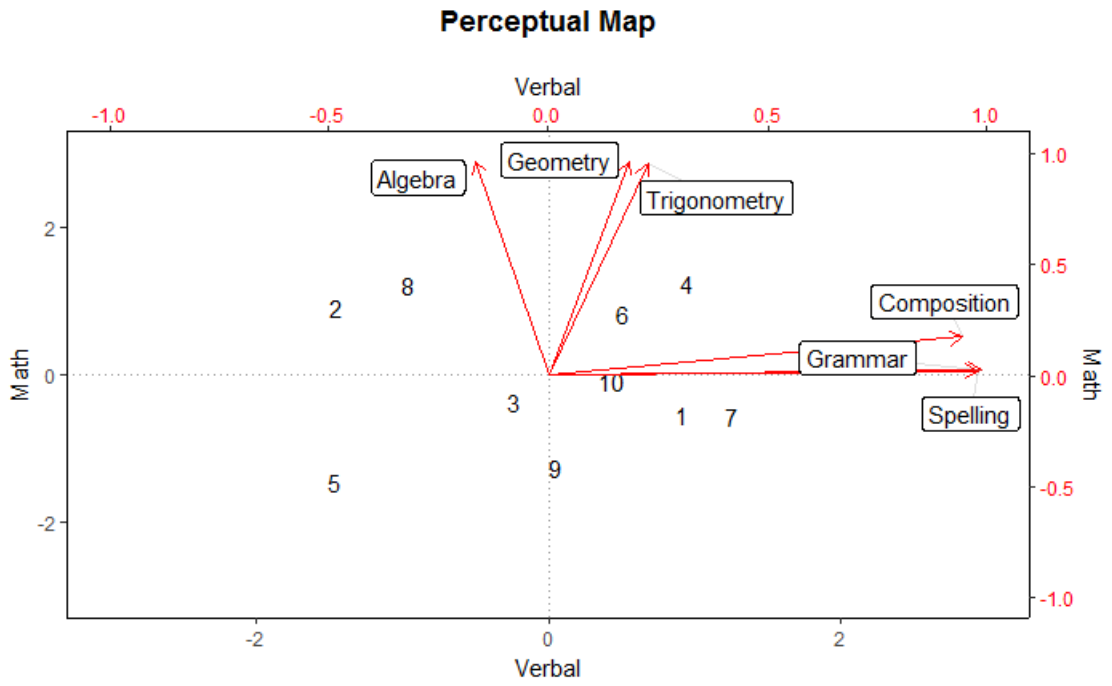
Hide

```
#PCA plot!
#ggplot takes at least two arguments: what dataframe it is referencing (ty),
#and the aesthetic mappings of the plot it will produce, i.e. the names of the columns it will use for the x
and y axes.
#the default x and y axes titles are the variable names (in this case, RC1 and RC2). if you want to change t
hem just uncomment the below:
# colnames(ty) = c("yournew_x_name", "yournew_y_name")
library(ggplot2)
library(ggrepel)
ggplot(ty, aes(Verbal, Math)) +
  #first, the scores

  #for the students, we will restrict the referenced data to only the first 10 rows of ty (the last 6 are cl
asses).
  #rather than plotting points ('geom_point()'), we are plotting the text, hence the use of the geom_text fu
nction
  geom_text(data = ty[1:10,],
            aes(x = ty[1:10,1],
                y = ty[1:10,2],
                label = rownames(ty[1:10,]))) + #assigning the row names as the text labels, but could be a
ny list of 10 character elements
  #the 'sec.axis' arguments below create a second x and y axis to mimic the BI plot above,
  #the creators of R and the ggplot package intentionally try to discourage plots with more than 2 axes on t
he same plane
  #so we must make the "second" axes linear transformations of the first. here, i divide by 3 to mimic the B
I plot above
  #i had to multiply the loadings data by 3 when i joined it to the scores data,
  #because plotting the raw loadings data on the 'scores' scale made the arrows too small. again, this is ma
de intentionally difficult
  #we can change the second axes titles by changing what's given to the 'name' argument
  scale_y_continuous(limits = c(-3,3),
                     sec.axis = sec_axis(~./3, name = "Math")) +
  scale_x_continuous(limits = c(-3,3),
                     sec.axis = sec_axis(~./3, name = "Verbal")) +
  #the below is purely to make the plot look better, setting colors, typeface, etc.
  theme_classic() +
  theme(axis.text.x.top = element_text(color = "red"),
        axis.text.y.right =  element_text(color = "red"),
        plot.title = element_text(face = "bold",hjust = 0.5)) +
  geom_hline(yintercept = 0, linetype = "dotted", alpha = .4) +
  geom_vline(xintercept = 0, linetype = "dotted", alpha = .4) +

  #now we do loadings! referencing a subset of row vectors that correspond to the loadings data

  #plotting 'segments' as the line one can draw between the origin and the point
  geom_segment(data=ty[11:16,],
               aes(x = ty[11:16,1],
                   y =ty[11:16,2]),
               xend=0,
               yend=0,
               arrow = arrow(length = unit(0.03, "npc"),
                             ends = "first"),
               color = "red") +
  #'smart labels' that won't overlap with each other
  geom_label_repel(data=ty[11:16,],
                   aes(x = ty[11:16,1],
                       y = ty[11:16,2],
                       label = rownames(ty[11:16,])),
                   label = rownames(ty[11:16,]),
                   segment.alpha = .25,
                   box.padding = unit(0.35, "lines"),
                   segment.color = "grey50") +
  #setting a title
  ggtitle("Perceptual Map\n")
#and, finally, saving to your working directory
ggsave("pca_plot_example.png", height = 8, width = 8)
```



We can see from this map that factor 1 (the x-axis) is associated with the verbal tests and factor 2 is associated with the math tests. We can also see that student 5 is poor on both verbal and math abilities whereas student 4 is strong on both. As the olds saying goes, a picture is worth more than 1000 words.