**Function:** Hornscurve()

**Descriptive name:** Horn’s Parallel Analysis

**Function Explanation:** Returns a vector of the average eigenvalues produced by a Monte Carlo simulation that randomly generates a large number of matrices of size *N x p*, where each element is drawn from a standard normal probability distribution. The values can then be plotted or compared to the true eigenvalues of a dataset for a dimensionality assessment. [1]

**Parameters:**

* N: integer value
* p: integer value

**Examples:**

##Graph the scree line for a dimensionality assessment

x = matrix(rnorm(200\*3), ncol = 10)

N = nrow(x)

M = ncol(x)

curvepoints = Hornscurve(N,M)

plot(curvepoints)

**Function:** Factor\_Analysis()

**Descriptive name:** Factor Analysis with Varimax rotation

**Function Explanation:** Reduce the structure of the data by relating the correlation between variables to a set of factors, using the eigen-decomposition of the correlation matrix.

[2]

**Parameters:**

* data: matrix of data
* HC\_points: vector of eigenvalues [designed to use output from Hornscurve() function]

**Values:**

* fa\_loadings: numerical matrix with the original factor loadings
* fa\_scores: numerical matrix with the row scores for each factor
* fa\_loadings\_rotated: numerical matrix with the varimax rotated factor loadings
* fa\_scores\_rotated: numerical matrix with the row scores for each varimax rotated factor

**Examples:**

##Perform factor analysis on a numerical matrix

x = matrix(rnorm(200\*3), ncol = 10)

N = nrow(x)

M = ncol(x)

curvepoints = Hornscurve(N,M)

FA\_output = Factor\_Analysis(x,curvepoints)

#plot the factor loadings in correlation heat map

heatmap.2(FA\_output$fa\_loadings\_rotated,Rowv = FALSE, Colv = FALSE, dendrogram = “none”, trace = “none”, density.info = “none”, breaks = c(-1,-0.75,-0.5,-0.25,0,0.25,0.5,0.75,1), col = brewer.pal(8, “RdBu”))

#plot the factor scores against one another to determine outliers

plot(FA\_output$fa\_scores\_rotated[,1], FA\_output$fa\_scores\_rotated[,2]

text(FA\_output$fa\_scores\_rotated[,1], FA\_output$fa\_scores\_rotated[,2],labels = 1:N)

**Function:** IFS()

**Descriptive name:** Kaiser’s Index of Factorial Simplicity

**Function Explanation:** Developed by Henry Kaiser [3], this function returns a score designed to assess the quality of a factor analysis solution. It measures the tendency towards unifactoriality for both a given row and the entire matrix as a whole. Kaiser proposed the evaluations of the score shown in the table below. Use as basis for selecting original or rotated loadings/scores in Factor\_Analysis().

|  |  |
| --- | --- |
| **IFS Level** | **Evaluation** |
| In the .90s | Marvelous |
| In the .80s | Meritorious |
| In the .70s | Middling |
| In the .60s | Mediocre |
| In the .50s | Miserable |
| Below .50s | Unacceptable |

**Parameters:**

* L: numerical matrix of the factor loadings

**Examples:**

## Find the original and rotated IFS scores

x = matrix(rnorm(200\*3), ncol = 10)

N = nrow(x)

M = ncol(x)

curvepoints = Hornscurve(N,M)

FA\_output = Factor\_Analysis(x, curvepoints)

IFS\_unrotated = IFS(FA\_output$fa\_loadings)

IFS\_rotated = IFS(FA\_output$fa\_loadings\_rotated)

#usually, the rotated score will be higher

There is already a function in the stats package called Mahalanobis

**Function:** mahalanobis3()

**Descriptive name:** Mahalanobis Distance

**Function Explanation:** Calculates the distance between the elements in *data* and the mean vector of the data for outlier detection. Values are independent of the scale between variables. [4]

**Parameters:**

* data: matrix of data

**Values:**

* md: vector of distances, one for each row
* md\_sort: sorted vector of distances
* md\_index: The ordering index vector
* bd: matrix of the absolute values of the breakdown distances. Used to see which columns drive the Mahalanobis distance

**Examples:**

## Create a Histogram Matrix [5] using the Mahalanobis distance and breakdown distance

x = matrix(rnorm(200\*3), ncol = 3)

colnames(x) = c(“C1”, “C2”, “C3”)

N = nrow(x)

M = ncol(x)

MD = mahalanobis3(x)

#the next step normalizes the columns since each variable has its own scale

temp = MD$bd %\*% diag(1/colSums(MD$bd))

colnames(temp) = colnames(x)

temp = data.frame(cbind(c(1:N),MD$md,temp)

colnames(DF1)[1:2] = c(“Row”, “MD”)

DF1 = temp %>% gather(Col\_Name,BD,3:(M+2))

DF1$Col\_Name <- factor(DF1$Col\_Name, levels=unique(DF1$Col\_Name))

ggplot(DF1,aes(x=Col\_Name,y=Row,color=MD,size=BD)) + geom\_point()

**Function:** bd\_row()

**Descriptive name:** Breakdown for Mahalanobis Distance

**Function Explanation:** Returns a vector that indicates which variables in *data* are driving the Mahalanobis distance for a specific row *r*, relative to the mean vector of the data*.*

**Parameters:**

* data: matrix of data
* r: row of interest in data

**Values:**

* bd: vector of the breakdown distances for each column (values could be +/-)
* bd\_mag: vector of the absolute values of the breakdown distances
* bd\_sort: sorted absolute values
* bd\_index: The ordering index vector of the sorted values

**Examples:**

##Find the breakdown distance of row 5 and show a barplot of the top 3 columns

x = matrix(rnorm(200\*3), ncol = 10)

colnames(x) = paste(“C”,1:ncol(x))

BD\_5 = bd\_row(x,5)

Top3 = BD\_5%bd\_index[1:3]

barplot(x[,Top3],col=rainbow(3),legend = colnames(x)[top3])

The function below is something I found on stackoverflow as a solution posted by Richie Cotton

**Function:** get\_all\_factors()

**Descriptive name:** Find All Factors

**Function Explanation:** This function finds all factor pairs of a given number

**Parameters:**

* n: number to be factored

**Examples:**

##Find all the factors of 39304

get\_all\_factors(39304)

[1] 1 2 4 8 17 34 68 136 289 578 1156 2312 4913 9826 19652 39304

# Bibliography

|  |  |
| --- | --- |
| [1] | J. L. Horn, "A rationale and test for the number of factors in factor analysis," *Psychometrika,* vol. 30, no. 2, pp. 179-185, 1965. |
| [2] | H. F. Kaiser, "The Application of Electronic Computers to Factor Analysis," Educational and Psychological Measurement, 1960. |
| [3] | H. F. Kaiser, "An index of factorial simplicity," *Psychometrika,* vol. 39, no. 1, pp. 31-36, 1974. |
| [4] | W. Wang and R. Battiti, "Identifying Intrusions in Computer Networks with Principal Component Analysis," in *First International Conference on Availability, Reliability and Security*, 2006. |
| [5] | A. Frei and M. Rennhard, "Histogram Matrix: Log file visualization for anomaly detection," in *Third International Conference on Availability, Reliability and Security*, 2008. |