Principal Components and Exploratory Factor Analysis Tutorial

Professor Christopher J. Schmank

2025-04-24

Install and Load R Packages for Principal Component and Factor Analysis

```r
# Load Packages Into R Environment

library(tidyverse)
library(jmv)
library(psych)
library(EGAnet)
```

We will conduct PCA and FA on a data frame called `wiscsem.csv`. This data represents 11 subscales of the Wechsler Intelligence Scale for Children (WISC) for 175 children. You can find the data set on Canvas. For more information about the WISC Tasks here is a brief description.

WISC-III Selected Tasks

Verbal Ability

`Information` (`info`) – general knowledge questions

`Comprehension` (`comp`) – questions about social situations or common concepts

`Arithmetic` (`arith`) – orally administered arithmetic word problems (timed test)

`Similarities` (`simil`)– asking how two words are alike/similar

`Vocabulary` (`vocab`) – examinee is asked to define a provided word

`Digit Span` (`digit`) - examinee listens to sequences of numbers orally and repeat them in reverse order and in ascending order

Performance Ability

`Picture Completion` (`pictcomp`) - examinees are shown images and tasked with describing what is missing from the image

`Picture Arrangement` (`parang`) - examinee presented with series of cards in incorrect order that when placed in correct order tells lucid story

`Block Design` (`block`) – examinee forms red-and-white blocks into pattern according to displayed model (timed test with bonuses for completing more difficult puzzles)

`Object Assembly` (`object`) - examinee views an image and attempts to reproduce the image with predetermined pieces

`Coding` (`coding`) – either the youngest examinees (< 8 years old) mark rows of shapes with different lines according to a pre determined code OR the older examinees (> 8 years old) transcribe a digit-symbol code using a key (time limited test)

Load `wiscsem.sav` into R

```
wisc_data <- read.csv("wiscsem.csv")

head(wisc_data)
```

```
##   client agemate info comp arith simil vocab digit pictcomp parang block object
## 1      3       3    8    7    13     9    12     9        6     11    12      7
## 2      4       3    9    6     8     7    11    12        6      8     7     12
## 3      5       3   13   18    11    16    15     6       18      8    11     12
## 4      6       3    8   11     6    12     9     7       13      4     7     12
## 5      7       2   10    3     8     9    12     9        7      7    11      4
## 6      8       3   11    7    15    12    10    12        6     12    10      5
##   coding
## 1      9
## 2     14
## 3      9
## 4     11
## 5     10
## 6     10
```

Notice that our first two variables reflect an ID variable (`client`) and a categorical age variable (`agemate`), when it comes to running our PCA or FAs we will **NOT** want these variables in our data frame. We will start by removing these columns using the subsetting features available in R.

Subsetting the Data Frame

The following command retains all rows, but deletes the collection of columns 1 and 2. We have seen this coding before, but remember this is very important when it comes to creating a data frame with ONLY THE DATA WE WANT TO FACTOR ANALYZE!

```
# Can make a call to remove extraneous variables

wisc_dat <- wisc_data[,-c(1,2)]
head(wisc_dat)
```

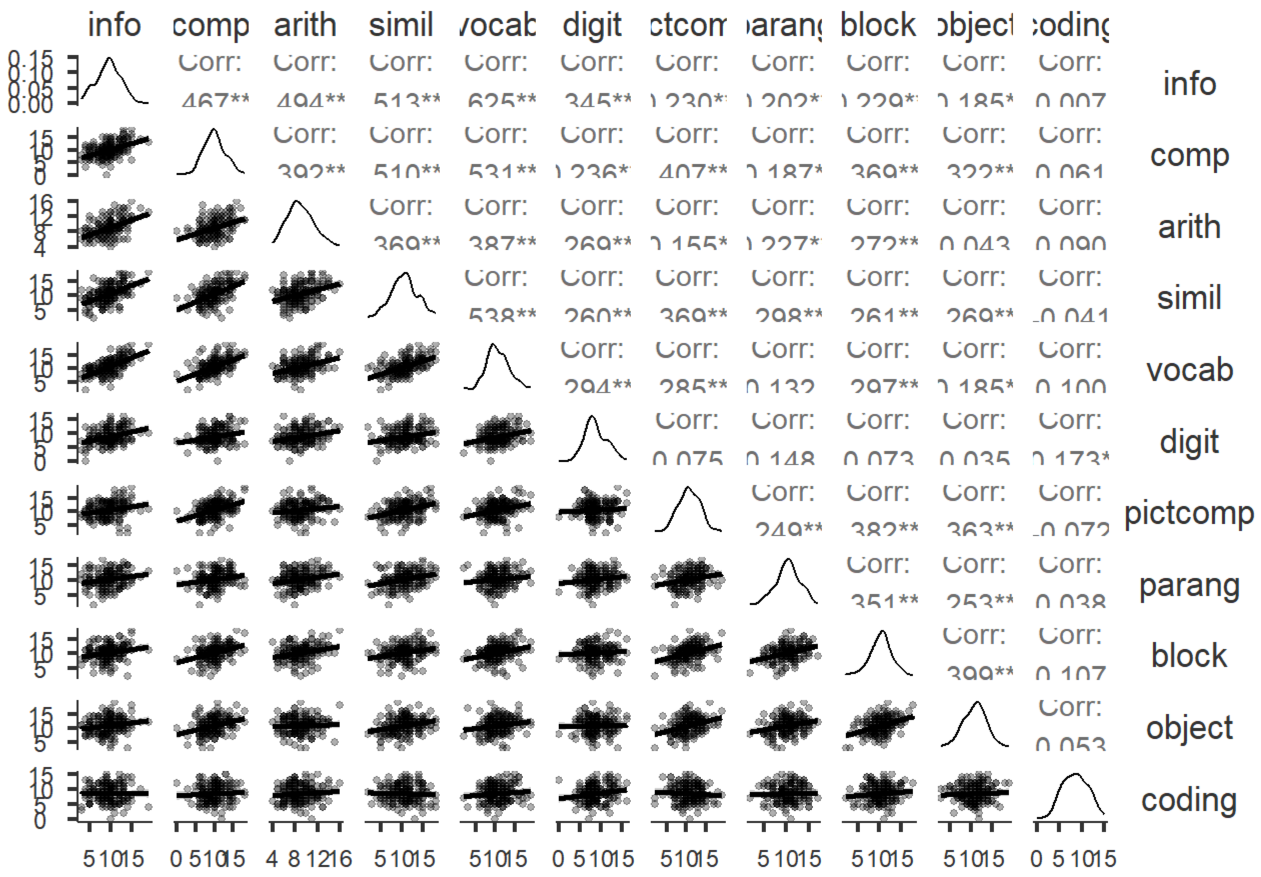
```
##   info comp arith simil vocab digit pictcomp parang block object coding
## 1    8    7    13     9    12     9        6     11    12      7      9
## 2    9    6     8     7    11    12        6      8     7     12     14
## 3   13   18    11    16    15     6       18      8    11     12      9
## 4    8   11     6    12     9     7       13      4     7     12     11
## 5   10    3     8     9    12     9        7      7    11      4     10
## 6   11    7    15    12    10    12        6     12    10      5     10
```

```
# OR can call only the variables to retain
# wisc_dat <- wisc_data[3:13]
```

Correlation Matrix

Next, let's generate a correlation matrix. **Remember**: We ONLY need the correlation matrix when using PCA or FA and the sample size that the correlation matrix was generated from. So, we can use the base R `cor()` function OR the psych package `corr.test()` function to create this data object.

```
# Base R: `cor()`
wisc_cor <- cor(wisc_dat)

# psych package: `corr.test()`
wisc_corr <- corr.test(wisc_dat)$r

# Optional: Visualization from `corrMatrix()`
corrMatrix(wisc_dat,
           plots = TRUE,
           plotDens = TRUE,
           plotStats = TRUE)$plot
```



Now that we have our correlation matrix, we must run some diagnostics/assumption checks on the factorability of the correlation matrix. **Remember**: Since this is a correlational/regression technique, we would have ALREADY run regression diagnostics as a means of screening and cleaning the data set!

Bartlett's Test of Sphericity

This initial assumption test is to make sure that the correlations in our matrix (as a unit) show devition from an Identity matrix (where 1's are along the diagonal and 0's are everywhere else):

```
# cortest.bartlett() requires two arguments:
# 1. The correlation matrix object (Can also use a data set--will receive error message about needing to compute square R ma
trix, i.e., correlation matrix)
# 2. Sample size (N) of the data set

cortest.bartlett(wisc_cor, n = 175)
```

```
## $chisq
## [1] 502.8861
##
## $p.value
## [1] 1.380904e-73
##
## $df
## [1] 55
```

OK, so *p* is VERY small (*p* < .001), so we can be confident that our sample correlation matrix is statistically significantly different from an identity matrix–i.e., correlations are substantial enough for factorization.

The *degrees of freedom (df)* for Bartlett's Test here is the number of correlations within the correlation matrix; which is also the `number of sampling moments` minus the `number of items in the data set` :

To calculate *degrees of freedom* for this analysis and any other using Bartlett's Test you can use the following formula:

$$\text{Sampling Moments} = [\text{Number of Variables} * (\text{Number of Variables } + 1)]/2$$

$$(11 * 12)/2 = 66 \text{ Sampling Moments}$$

From these 66 `sampling moments` we subtract the number of variables we estimate correlations for (i.e., 11 variables in `wisc_dat`) and end up with our Bartlett's *df* of 55—also the number of correlations in a matrix with 11 variables! This is a good sanity check to make sure you are including the correct number of variables in your analyses before you get to the PCA or EFA sections!

Kaiser-Meyer-Olkin (KMO) Test of Sampling Adequacy

```
# KMO() requires one argument:
# 1. Correlation Matrix

KMO(wisc_cor)
```

```
## Kaiser-Meyer-Olkin factor adequacy
## Call: KMO(r = wisc_cor)
## Overall MSA =  0.83
## MSA for each item = 
##     info     comp    arith    simil    vocab    digit pictcomp   parang 
##     0.82     0.89     0.83     0.87     0.82     0.85     0.85     0.78 
##    block   object   coding 
##     0.80     0.77     0.43
```

Important Note: `Sampling Adequacy` refers to the adequacy that variables were chosen for the analysis, it has NOTHING to do with the sample of subjects or cases collected!!

For factor analysis it is desirable that each factor (or component) be measured by at least three items. A larger KMO is desirable, with 1.00 the maximum possible. Values between .80-.89 are viewed as ‘meritorious’ and greater than .90 as ‘marvelous.’

Metrics Used to Assess KMO Values

Value	Judgment
0.00 – 0.50	Unacceptable / Bad
0.50 – 0.60	Miserable / Bad
0.60 – 0.70	Mediocre / OK
0.70 – 0.80	Middling / OK
0.80 – 0.90	Meritorious / Good
0.90 – 1.00	Marvelous / Great

If all variables are independent, KMO = .50. `MSA for each item` is a 'Measure of Sampling Adequacy' for each item relative to the other items in the data set. If an item has at least two 'friends' with mutually high correlations, MSA is large. Here we see that the overall MSA (i.e., KMO) looks 'meritorious' and all items have good MSA values except for `coding` which doesn't have at least two good friends. We will address this later.

Determinant

```
# det() requires one argument:
# 1. Correlation Matrix

det(wisc_cor)
```

```
## [1] 0.05146363
```

If the `determinant` of a matrix is zero, that is an indication of *singularity* where at least one variable can be perfectly predicted from the other variables. Thus, when the `determinant` is zero, there is less information in the correlation matrix than first meets the eye, and some matrix algebra calculations cannot be performed.

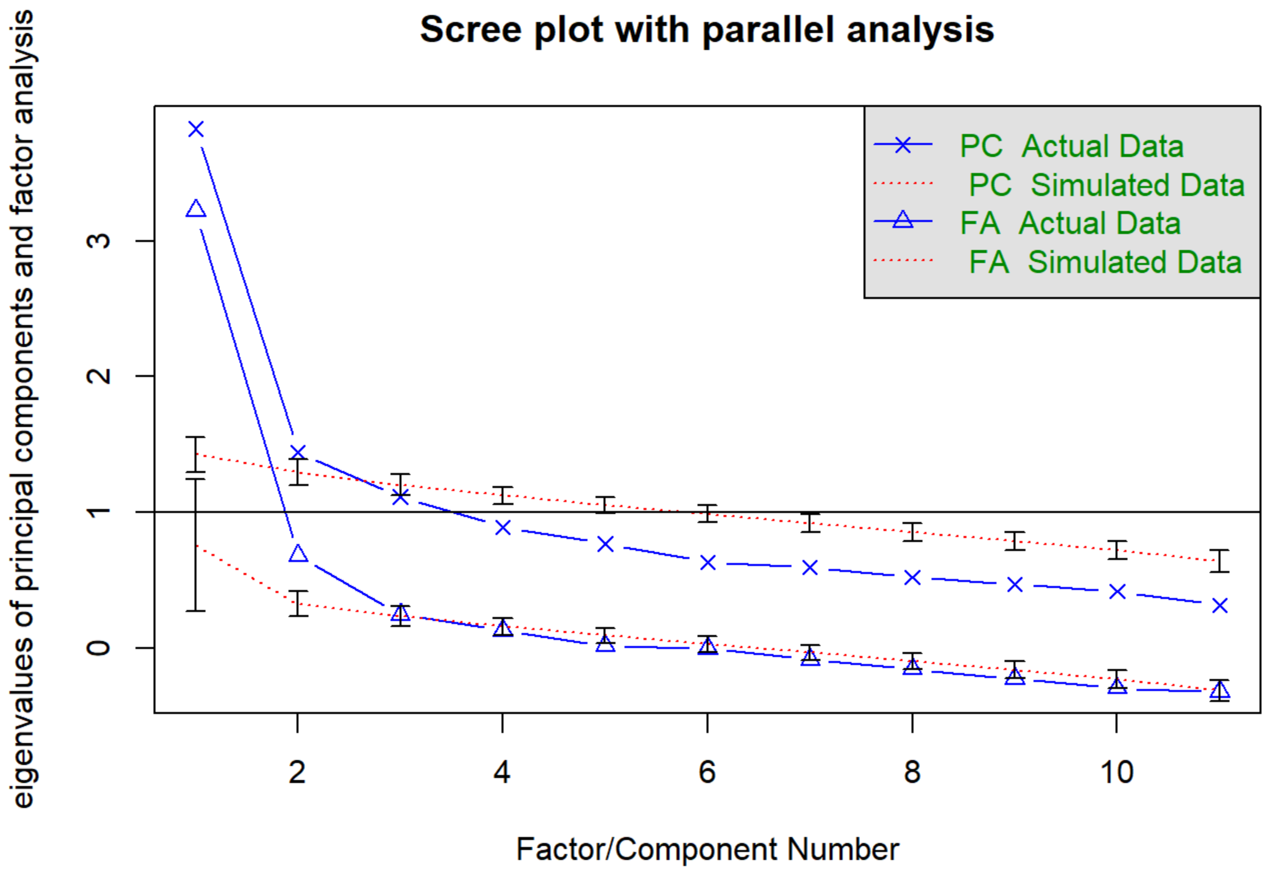
If the correlation is an identity matrix with all correlations equal to zero, then the `determinant` is equal to 1.00. So, we do expect to have a small `determinant`, but just not zero.

`Determinant` values less than .00001 are problematic. When this occurs you should assess your correlation matrix for values greater than .8 and remove any and all variables that correlate this highly as they are statistically redundant.

Parallel Analysis (How many Components/Factors to Extract?)

Traditionally, Scree plots have been used to show the pattern of the Eigenvalues to make an assessment of how many factors/components should be extracted based on a correlation matrix or data set. Our coding will differ whether we are using PCA or FA so the next few chunks are representative of these differences.

```r
# fa.parallel() requires several arguments:
# 1. Data set or Correlation Matrix
# 2. Number of observations (Use when input is correlation matrix)
# 3. fa - For both we use "both"
# 4. fm - allows for a factor method to be specified IFF using factor analysis, e.g. maximum likelihood is "ml", principal a
xis is “pa”
# 5. n.iter - How many bootstrapped iterations do you want to use --- the more you use the more computing power is required
# 6. error.bars = TRUE or FALSE (Do you want error bars on your plot?)
# 7. show.legend = TRUE or FALSE (Do you need a legend?)
# 8. main = Title of plot

fa.parallel(wisc_cor,
            n.obs = 175,
            fa="both",
            fm="ml",
            n.iter=500,
            error.bars=TRUE,
            show.legend=TRUE,
            main="Scree plot with parallel analysis")
```



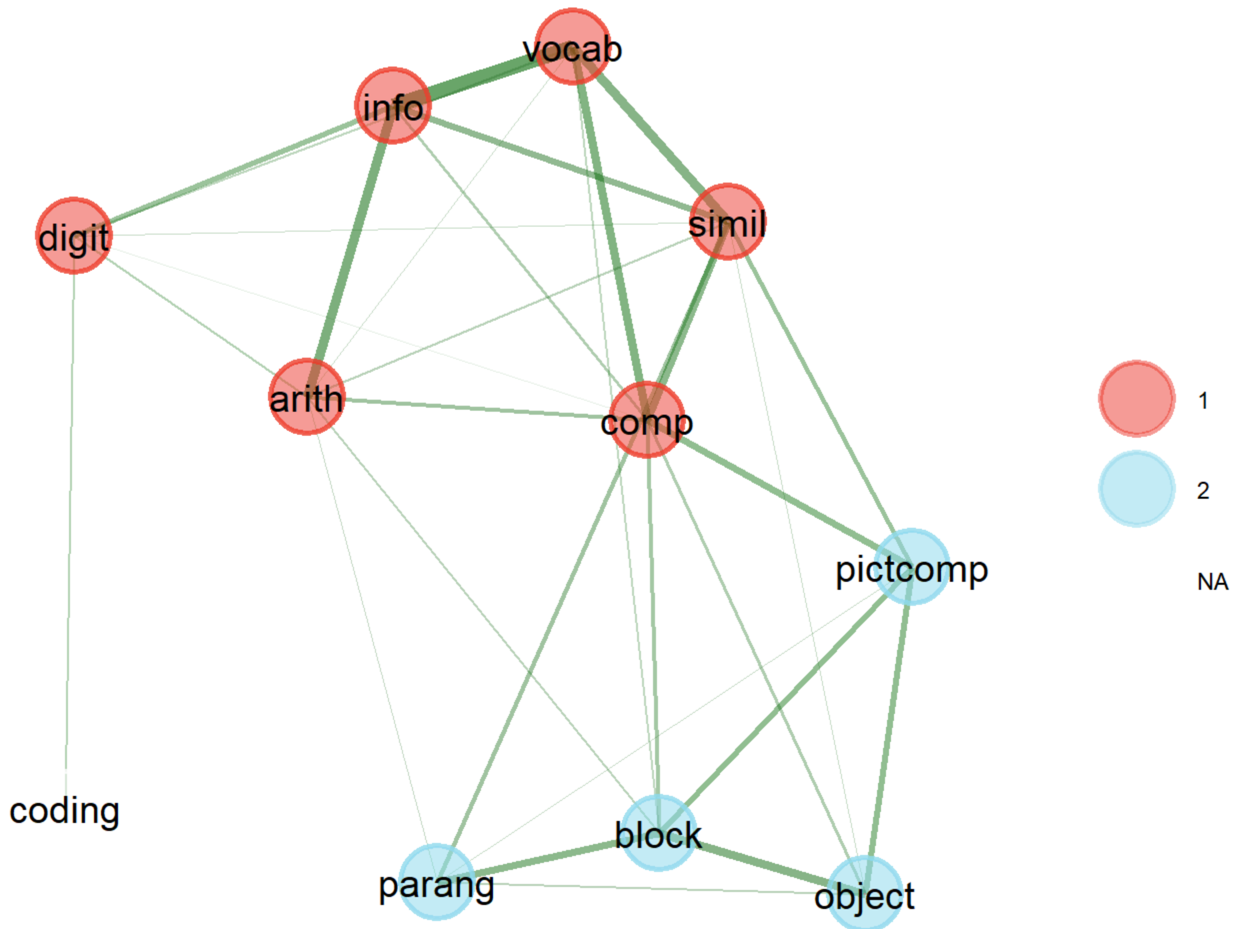
```
## Parallel analysis suggests that the number of factors =  2  and the number of components =  2
```

```
# Optional Code: fa.parallel() using raw data

# fa.parallel(wisc_dat,
#             fa="both",
#             fm="ml",
#             n.iter=500,
#             error.bars=TRUE,
#             show.legend=TRUE,
#             main="Scree plot with parallel analysis")
```

Something else that I have been using lately is called an `Exploratory Graph Analysis` which conducts a similar assessment of the number of dimensions of a correlation matrix or raw data set using `psychometric network analysis` :

```
# Remember to copy and paste install.packages("EGAnet", dependencies = TRUE) into your console if you have not installed thi
s into your R Environment!

EGA(wisc_cor,
    n = 175)
```

```
## Warning: Some variables did not belong to a dimension: coding
##
## Use caution: These variables have been removed from the TEFI calculation
```



```
## Model: GLASSO (EBIC with gamma = 0.5)
## Correlations: auto
## Lambda: 0.106713138166304 (n = 100, ratio = 0.1)
##
## Number of nodes: 11
## Number of edges: 31
## Edge density: 0.564
##
## Non-zero edge weights:
##      M    SD   Min   Max
##  0.113 0.076 0.007 0.339
##
## ----
##
## Algorithm:  Walktrap
##
## Number of communities:  2
##
##     info     comp    arith    simil    vocab    digit pictcomp   parang
##        1        1        1        1        1        1        2        2
##    block   object   coding
##        2        2       NA
##
## ----
##
## Unidimensional Method: Louvain
## Unidimensional: No
##
## ----
##
## TEFI: -5.408
```

```
# Optional Code: EGA() for raw data

# EGA(wisc_dat)
```

Although this does not provide us with any information about how PCA and EFA differ, it does provide us with another indication of how many dimensions (i.e., components OR factors) the correlational space could be divided into. Additionally, it focuses on the partial correlations among the manifest variables assessed by the raw data instead of attempting to generate unobserved latent variables—a very popular technique these days!

Takeaway Message: Our parallel analysis and the exploratory graph analysis ALL indicate that 2 components or factors (i.e., dimensions) are appropriate for our subsequent analyses, so we will proceed with this number in mind.

Principal Components Analysis (PCA)

```
# principal() requires several arguments:
# 1. Data set or Correlation Matrix
# 2. nfactors - how many components to extract (default is 1)
# 3. rotate - allows for specification of a particular rotation method (default is varimax)
# 4. scores - allows component scores to be obtained (default is FALSE)
# 5. residuals - can assess residuals to see how well our PCA fits data (residuals closer to 0 imply better fit)

# Full Component Model using Raw Data
# Model will extract the same number of components as variables without rotation

wisc_pca <- principal(wisc_dat,
                      nfactors = 11,
                      rotate = "none")

print.psych(wisc_pca, cut=.3,sort = TRUE)
```

```
## Principal Components Analysis
## Call: principal(r = wisc_dat, nfactors = 11, rotate = "none")
## Standardized loadings (pattern matrix) based upon correlation matrix
##          item  PC1   PC2   PC3   PC4   PC5   PC6   PC7   PC8   PC9  PC10  PC11
## comp        2 0.75                                           -0.45 -0.31
## simil       4 0.74                               0.30              0.45
## vocab       5 0.74                                    -0.34              0.36
## info        1 0.74 -0.35                                      0.31       -0.34
## arith       3 0.60 -0.35             -0.46              0.37
## block       9 0.57  0.45             -0.30             -0.38
## pictcomp    7 0.56  0.47                    0.49              0.30
## object     10 0.46  0.58                   -0.36 -0.32
## coding     11              0.86 -0.31
## parang      8 0.45  0.32        0.70
## digit       6 0.42 -0.47              0.60
##          h2       u2 com
## comp      1 -8.9e-16 2.6
## simil     1 -1.8e-15 2.8
## vocab     1 -8.9e-16 2.9
## info      1 -1.8e-15 3.0
## arith     1 -6.7e-16 4.6
## block     1 -1.1e-15 5.2
## pictcomp  1  5.6e-16 4.6
## object    1 -4.4e-16 5.0
## coding    1 -4.4e-16 1.7
## parang    1  3.3e-16 3.3
## digit     1  0.0e+00 4.5
##
##                        PC1  PC2  PC3  PC4  PC5  PC6  PC7  PC8  PC9 PC10 PC11
## SS loadings           3.83 1.44 1.12 0.89 0.77 0.63 0.60 0.52 0.47 0.42 0.31
## Proportion Var        0.35 0.13 0.10 0.08 0.07 0.06 0.05 0.05 0.04 0.04 0.03
## Cumulative Var        0.35 0.48 0.58 0.66 0.73 0.79 0.84 0.89 0.93 0.97 1.00
## Proportion Explained  0.35 0.13 0.10 0.08 0.07 0.06 0.05 0.05 0.04 0.04 0.03
## Cumulative Proportion 0.35 0.48 0.58 0.66 0.73 0.79 0.84 0.89 0.93 0.97 1.00
##
## Mean item complexity =  3.7
## Test of the hypothesis that 11 components are sufficient.
##
## The root mean square of the residuals (RMSR) is  0
##  with the empirical chi square  0  with prob <  NA
##
## Fit based upon off diagonal values = 1
```

```
# Before parallel analysis we could use the SS loadings (i.e., Eigenvalues) to assess the number of components that should b
e extracted--based on Kaiser cutoff values of 1.00 for Eigenvalues we could have 3 components based on this correlation matr
ix
```

Reading the output

Component Loadings - Values Under **PC#** Columns of the pattern matrix

Values associated with individual items and the component they are associated with Squaring these loadings assesses the amount of variance accounted for in an item by the component

Communality - h^2

This is the proportion of the variance of the variable that is captured by the component(s). In this case we have 11 components to capture variance of 11 variables, so it is no surprise that we can capture ALL of the variance for each variable.

Uniqueness - u^2 or $1 – h^2$

The variance that is NOT shared with the components. The values shown are zero within computational error because PCA assumes that ALL the communality within an item is explained by the component (This is a disadvantage!!)

Eigenvalues - *SS loadings*

Used to use these values to produce a scree plot of the Eigenvalues to clarify how many components to extract. The first three Eigen values are greater than 1.00, so the Kaiser rule suggests three components to be extracted.

Accounted for Variance - *Proportion Var* and *Cumulative Var*

The proportion of variance of the original items/variables that are captured by each of the components. With 11 variables, chance is 1/11 or .091– the first three components exceed this proportion substantially implying that these components account for 35%, 13%, and 10% (respectively) of the variance in the items that load on these components highly. The cumulative variance column adds these values up to assess exactly how much variance was accounted for (in the full component model this will always be 1.00!)

PCA Based on Parallel Analysis and Eigenvalues

```
# Model will extract 2 components based on parallel analysis without rotation

wisc_pca2 <- principal(wisc_dat,
                      nfactors = 2,
                      rotate = "none")

print.psych(wisc_pca2, cut = .3, sort = TRUE)
```

```
## Principal Components Analysis
## Call: principal(r = wisc_dat, nfactors = 2, rotate = "none")
## Standardized loadings (pattern matrix) based upon correlation matrix
##          item  PC1   PC2    h2   u2 com
## comp        2 0.75       0.569 0.43 1.0
## simil       4 0.74       0.558 0.44 1.0
## vocab       5 0.74       0.631 0.37 1.3
## info        1 0.74 -0.35 0.664 0.34 1.4
## arith       3 0.60 -0.35 0.485 0.51 1.6
## block       9 0.57  0.45 0.531 0.47 1.9
## pictcomp    7 0.56  0.47 0.531 0.47 1.9
## parang      8 0.45  0.32 0.304 0.70 1.8
## object     10 0.46  0.58 0.554 0.45 1.9
## digit       6 0.42 -0.47 0.398 0.60 2.0
## coding     11            0.045 0.95 1.5
##
##                        PC1  PC2
## SS loadings           3.83 1.44
## Proportion Var        0.35 0.13
## Cumulative Var        0.35 0.48
## Proportion Explained  0.73 0.27
## Cumulative Proportion 0.73 1.00
##
## Mean item complexity =  1.6
## Test of the hypothesis that 2 components are sufficient.
##
## The root mean square of the residuals (RMSR) is  0.09
##  with the empirical chi square  150.46  with prob <  1.4e-16
##
## Fit based upon off diagonal values = 0.91
```

```
# Model will extract 3 components based on Eigenvalues above 1.00 without rotation

wisc_pca3 <- principal(wisc_dat,
                       nfactors = 3,
                       rotate = "none")

print.psych(wisc_pca3, cut = .3, sort = TRUE)
```

```
## Principal Components Analysis
## Call: principal(r = wisc_dat, nfactors = 3, rotate = "none")
## Standardized loadings (pattern matrix) based upon correlation matrix
##          item  PC1   PC2   PC3   h2   u2 com
## comp        2 0.75             0.58 0.42 1.0
## simil       4 0.74             0.62 0.38 1.2
## vocab       5 0.74             0.64 0.36 1.3
## info        1 0.74 -0.35       0.69 0.31 1.6
## arith       3 0.60 -0.35       0.49 0.51 1.6
## block       9 0.57  0.45       0.60 0.40 2.3
## pictcomp    7 0.56  0.47       0.56 0.44 2.2
## parang      8 0.45  0.32       0.37 0.63 2.4
## object     10 0.46  0.58       0.57 0.43 2.0
## digit       6 0.42 -0.47       0.47 0.53 2.6
## coding     11             0.86 0.79 0.21 1.1
##
##                        PC1  PC2  PC3
## SS loadings           3.83 1.44 1.12
## Proportion Var        0.35 0.13 0.10
## Cumulative Var        0.35 0.48 0.58
## Proportion Explained  0.60 0.23 0.17
## Cumulative Proportion 0.60 0.83 1.00
##
## Mean item complexity =  1.8
## Test of the hypothesis that 3 components are sufficient.
##
## The root mean square of the residuals (RMSR) is  0.09
##  with the empirical chi square  147.47  with prob <  2.5e-19
##
## Fit based upon off diagonal values = 0.91
```

Exploring Various PCA Solutions

Based on theory, we might expect two factors, such as verbal and performance abilities as bolstered by the parallel analysis. However, the Kaiser rule suggests the potential of three factors.

Let’s reassess our 3 component model using an Orthogonal rotation (i.e., `Varimax`)–**REMEMBER** this treats our components as completely uncorrelated latent variables:

```
wisc_pca3V <- principal(wisc_dat,
                        nfactors = 3,
                        rotate = "varimax",
                        residuals = TRUE)

print.psych(wisc_pca3, cut = .3, sort = TRUE)
```

```
## Principal Components Analysis
## Call: principal(r = wisc_dat, nfactors = 3, rotate = "none")
## Standardized loadings (pattern matrix) based upon correlation matrix
##          item  PC1   PC2   PC3   h2   u2 com
## comp        2 0.75             0.58 0.42 1.0
## simil       4 0.74             0.62 0.38 1.2
## vocab       5 0.74             0.64 0.36 1.3
## info        1 0.74 -0.35       0.69 0.31 1.6
## arith       3 0.60 -0.35       0.49 0.51 1.6
## block       9 0.57  0.45       0.60 0.40 2.3
## pictcomp    7 0.56  0.47       0.56 0.44 2.2
## parang      8 0.45  0.32       0.37 0.63 2.4
## object     10 0.46  0.58       0.57 0.43 2.0
## digit       6 0.42 -0.47       0.47 0.53 2.6
## coding     11             0.86 0.79 0.21 1.1
##
##                        PC1  PC2  PC3
## SS loadings           3.83 1.44 1.12
## Proportion Var        0.35 0.13 0.10
## Cumulative Var        0.35 0.48 0.58
## Proportion Explained  0.60 0.23 0.17
## Cumulative Proportion 0.60 0.83 1.00
##
## Mean item complexity =  1.8
## Test of the hypothesis that 3 components are sufficient.
##
## The root mean square of the residuals (RMSR) is  0.09
##  with the empirical chi square  147.47  with prob <  2.5e-19
##
## Fit based upon off diagonal values = 0.91
```

```
print.psych(wisc_pca3V, cut = .3, sort = TRUE)
```

```
## Principal Components Analysis
## Call: principal(r = wisc_dat, nfactors = 3, residuals = TRUE, rotate = "varimax")
## Standardized loadings (pattern matrix) based upon correlation matrix
##          item  RC1  RC2  RC3   h2   u2 com
## info        1 0.83            0.69 0.31 1.0
## vocab       5 0.78            0.64 0.36 1.1
## simil       4 0.69 0.32       0.62 0.38 1.6
## arith       3 0.67            0.49 0.51 1.2
## comp        2 0.63 0.42       0.58 0.42 1.7
## digit       6 0.53      0.43 0.47 0.53 1.9
## object     10      0.76       0.57 0.43 1.0
## block       9      0.74       0.60 0.40 1.2
## pictcomp    7      0.65       0.56 0.44 1.7
## parang      8      0.57       0.37 0.63 1.3
## coding     11           0.88 0.79 0.21 1.0
## 
##                        RC1  RC2  RC3
## SS loadings           3.02 2.21 1.15
## Proportion Var        0.27 0.20 0.10
## Cumulative Var        0.27 0.48 0.58
## Proportion Explained  0.47 0.35 0.18
## Cumulative Proportion 0.47 0.82 1.00
## 
## Mean item complexity =  1.3
## Test of the hypothesis that 3 components are sufficient.
## 
## The root mean square of the residuals (RMSR) is  0.09 
##  with the empirical chi square  147.47  with prob <  2.5e-19 
## 
## Fit based upon off diagonal values = 0.91
```

Now the h^2 communality values are more “interesting” when we extract FEWER components than items (not all 1). For instance, 37% of the variance in `parang` is captured by the three components (meaning 63% is left unaccounted for!). But if we look at the factor loadings for `parang` in the rotated model, we have a much easier time determining which factor explains it. **Remember**: This is just an artifact of running PCA with fewer components than variables, PCA ALWAYS ACCOUNTS FOR 100% OF THE VARIANCE!

Can also see that the majority of the variance in the `coding` variable has been accounted for by a single component–79% of the variance. When we take a look at the loadings for each component, `coding` has a loading of .88 on the third component (RC3), but it is the only variable with a loading greater than .43 on that component.

It **DOES NOT** make sense to have a component devoted to a single variable. Based on this and the KMO inadequacy it could be required/we might be justified in removing `coding` as a variable in our analysis–remember good PCA or EFAs require 3 indicators/items per latent variable. Additionally, if we remove `coding` our component/factor structure matches our parallel analysis!

Finetuning Before EFA (or PCA Reanalysis)

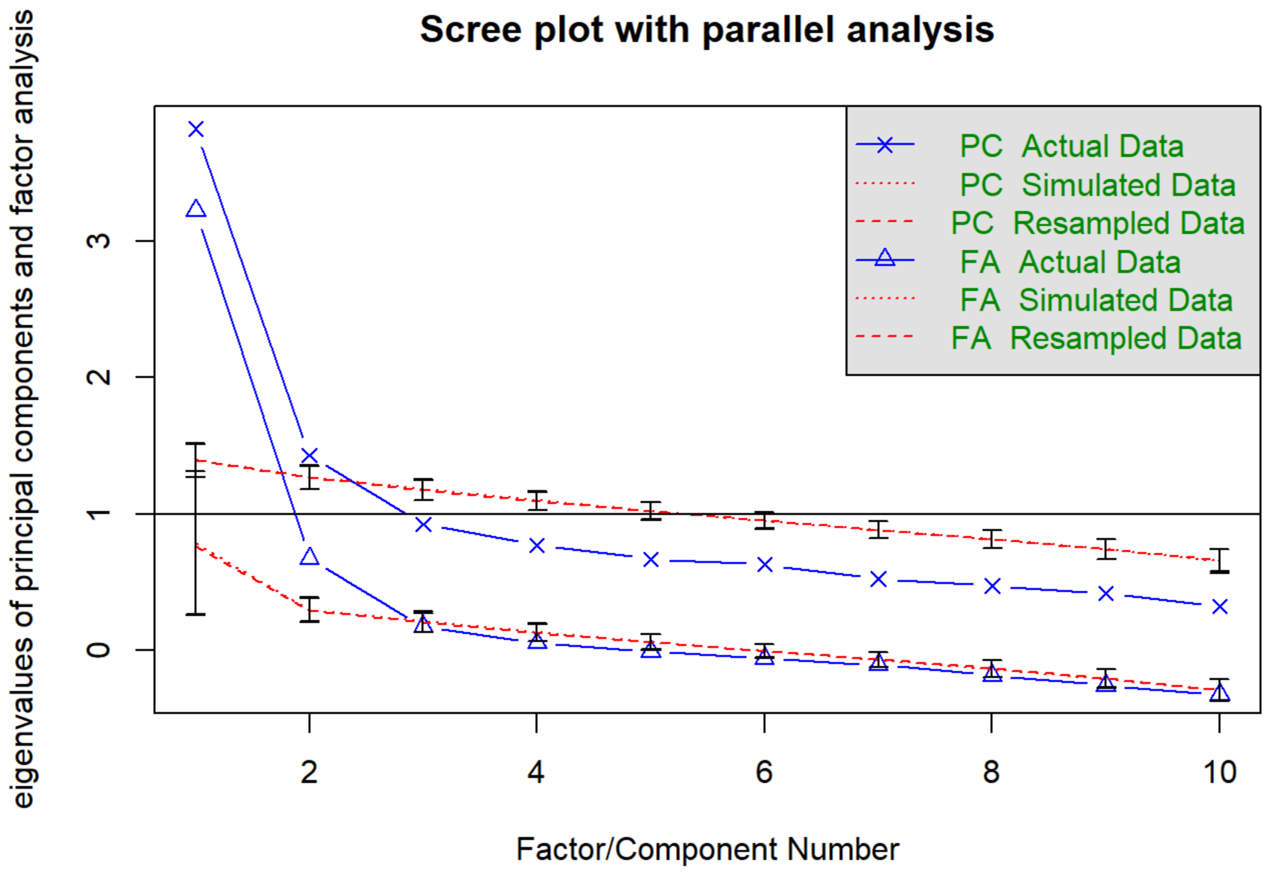
Can use a similar coding from before to remove coding from our data matrix:

```
wisc_dat_2 <- wisc_dat[,-11] #retains all rows, removes column 11
head(wisc_dat_2)
```

```
##   info comp arith simil vocab digit pictcomp parang block object
## 1    8    7    13     9    12     9        6     11    12      7
## 2    9    6     8     7    11    12        6      8     7     12
## 3   13   18    11    16    15     6       18      8    11     12
## 4    8   11     6    12     9     7       13      4     7     12
## 5   10    3     8     9    12     9        7      7    11      4
## 6   11    7    15    12    10    12        6     12    10      5
```

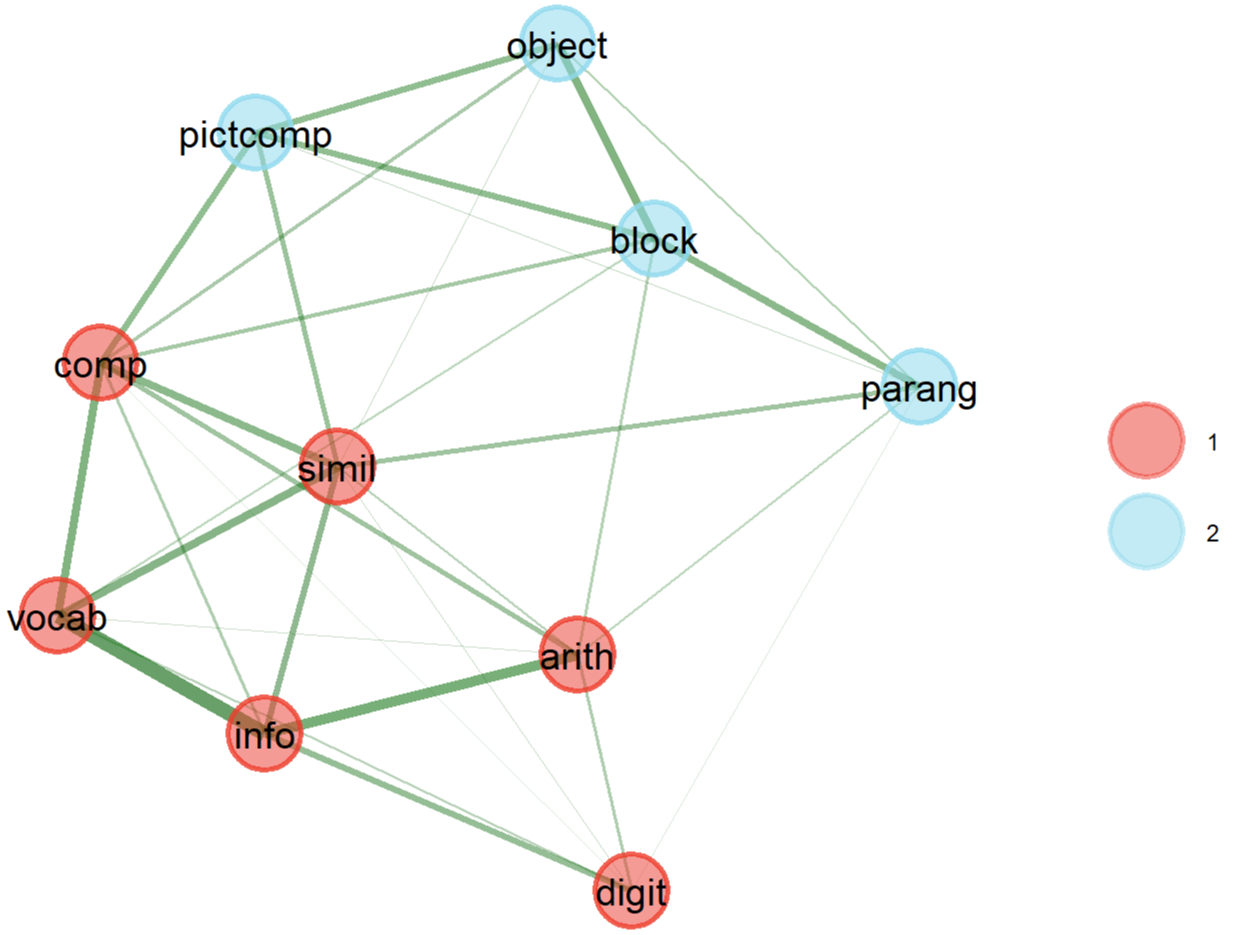
```
wisc_cor2 <- cor(wisc_dat_2)
```

```
# fa.parallel() requires several arguments:
# 1. Data set or Correlation Matrix
# 2. Number of observations (Use when input is correlation matrix)
# 3. fa - For both we use "both"
# 4. fm - allows for a factor method to be specified IFF using factor analysis, e.g. maximum likelihood is "ml", principal a
xis is “pa”
# 5. n.iter - How many bootstrapped iterations do you want to use --- the more you use the more computing power is required
# 6. error.bars = TRUE or FALSE (Do you want error bars on your plot?)
# 7. show.legend = TRUE or FALSE (Do you need a legend?)
# 8. main = Title of plot
fa.parallel(wisc_dat_2,
            fa="both",
            fm="ml",
            n.iter=500,
            error.bars=TRUE,
            show.legend=TRUE,
            main="Scree plot with parallel analysis")
```



```
## Parallel analysis suggests that the number of factors =  2  and the number of components =  2
```

```
EGA(wisc_dat_2)
```


object
pictcomp
block
comp
parang
simil
vocab
arith
info
digit
1
2

```
## Model: GLASSO (EBIC with gamma = 0.5)
## Correlations: auto
## Lambda: 0.0770555190451461 (n = 100, ratio = 0.1)
##
## Number of nodes: 10
## Number of edges: 31
## Edge density: 0.689
##
## Non-zero edge weights:
##      M    SD   Min   Max
##  0.116 0.079 0.012 0.351
##
## ----
##
## Algorithm:  Walktrap
##
## Number of communities:  2
##
##     info     comp    arith    simil    vocab    digit pictcomp   parang
##        1        1        1        1        1        1        2        2
##    block   object
##        2        2
##
## ----
##
## Unidimensional Method: Louvain
## Unidimensional: No
##
## ----
##
## TEFI: -5.408
```

Factor Analysis (Common Factor Analysis, FA)

Let’s start with looking at the finalized 10 variable data set assuming the 2 factor solution proposed by our parallel analysis and exploratory graph analysis. Lets also assume that our latent variables are correlated (i.e., oblique rotation) and lets use several extraction methods (just for good measure—always use “best” factor extraction method based on data quality).

```
# create factor analysis model as object
# form: fa(dataframe or R matrix, nfactors = number of factors, rotate = "method of rotation", scores = TRUE or FALSE, fm =
"factor method")

# nfactors - how many factors/components to extract; default is 1
# rotate - allows for specification of a particular rotation method; default is oblimin
# scores - allows factors scores to be obtained or not; default is FALSE (no scores)
# fm - allows for a factor method to be specified, e.g. maximum likelihood is "ml", principal axis is “pa”

wisc_fa <- fa(wisc_dat_2,
              nfactors = 2,
              fm = "ml")
```

```
## Loading required namespace: GPArotation
```

```
print.psych(wisc_fa, cut = .3, sort = TRUE)
```

```
## Factor Analysis using method =  ml
## Call: fa(r = wisc_dat_2, nfactors = 2, fm = "ml")
## Standardized loadings (pattern matrix) based upon correlation matrix
##          item   ML1   ML2   h2   u2 com
## info        1  0.84       0.65 0.35 1.0
## vocab       5  0.75       0.59 0.41 1.0
## arith       3  0.59       0.34 0.66 1.0
## simil       4  0.57       0.49 0.51 1.3
## comp        2  0.49  0.33 0.51 0.49 1.8
## digit       6  0.47       0.18 0.82 1.1
## object     10        0.65 0.37 0.63 1.0
## block       9        0.62 0.41 0.59 1.0
## pictcomp    7        0.60 0.39 0.61 1.0
## parang      8        0.40 0.19 0.81 1.1
##
##                        ML1  ML2
## SS loadings           2.52 1.60
## Proportion Var        0.25 0.16
## Cumulative Var        0.25 0.41
## Proportion Explained  0.61 0.39
## Cumulative Proportion 0.61 1.00
##
##  With factor correlations of
##      ML1  ML2
## ML1 1.00 0.49
## ML2 0.49 1.00
##
## Mean item complexity =  1.1
## Test of the hypothesis that 2 factors are sufficient.
##
## df null model =  45  with the objective function =  2.88 with Chi Square =  488.37
## df of  the model are 26  and the objective function was  0.19
##
## The root mean square of the residuals (RMSR) is  0.04
## The df corrected root mean square of the residuals is  0.05
##
## The harmonic n.obs is  175 with the empirical chi square  24.38  with prob <  0.55
## The total n.obs was  175  with Likelihood Chi Square =  32.66  with prob <  0.17
##
```

```
## Tucker Lewis Index of factoring reliability =  0.974
## RMSEA index =  0.038  and the 90 % confidence intervals are  0 0.075
## BIC =  -101.63
## Fit based upon off diagonal values = 0.99
## Measures of factor score adequacy             
##                                                    ML1  ML2
## Correlation of (regression) scores with factors   0.92 0.86
## Multiple R square of scores with factors          0.85 0.73
## Minimum correlation of possible factor scores     0.70 0.47
```

```
# Alternative for using correlation matrix as input

# wisc_fa2 <- fa(wisc_cor2,
#                nfactors = 2,
#                n.obs = 175,
#                fm = "ml")

# print.psych(wisc_fa2, cut = .3, sort = TRUE)
```

```
wisc_faB <- fa(wisc_dat_2,
               nfactors = 2,
               fm = "pa")

print.psych(wisc_faB, cut = .3, sort = TRUE)
```

```
## Factor Analysis using method =  pa
## Call: fa(r = wisc_dat_2, nfactors = 2, fm = "pa")
## Standardized loadings (pattern matrix) based upon correlation matrix
##          item   PA1   PA2   h2   u2 com
## info        1  0.84       0.65 0.35 1.0
## vocab       5  0.74       0.57 0.43 1.0
## arith       3  0.60       0.34 0.66 1.0
## simil       4  0.57       0.49 0.51 1.3
## comp        2  0.50  0.31 0.50 0.50 1.7
## digit       6  0.47       0.19 0.81 1.1
## object     10        0.66 0.38 0.62 1.0
## block       9        0.62 0.42 0.58 1.0
## pictcomp    7        0.58 0.38 0.62 1.0
## parang      8        0.38 0.19 0.81 1.1
##
##                        PA1  PA2
## SS loadings           2.54 1.57
## Proportion Var        0.25 0.16
## Cumulative Var        0.25 0.41
## Proportion Explained  0.62 0.38
## Cumulative Proportion 0.62 1.00
##
##  With factor correlations of
##      PA1  PA2
## PA1 1.00 0.48
## PA2 0.48 1.00
##
## Mean item complexity =  1.1
## Test of the hypothesis that 2 factors are sufficient.
##
## df null model =  45  with the objective function =  2.88 with Chi Square =  488.37
## df of  the model are 26  and the objective function was  0.2
##
## The root mean square of the residuals (RMSR) is  0.04
## The df corrected root mean square of the residuals is  0.05
##
## The harmonic n.obs is  175 with the empirical chi square  23.81  with prob <  0.59
## The total n.obs was  175  with Likelihood Chi Square =  32.97  with prob <  0.16
##
```

```
## Tucker Lewis Index of factoring reliability =  0.973
## RMSEA index =  0.039  and the 90 % confidence intervals are  0 0.076
## BIC =  -101.32
## Fit based upon off diagonal values = 0.99
## Measures of factor score adequacy             
##                                                    PA1  PA2
## Correlation of (regression) scores with factors   0.92 0.86
## Multiple R square of scores with factors          0.85 0.73
## Minimum correlation of possible factor scores     0.69 0.46
```

```
wisc_faC <- fa(wisc_dat_2,
               nfactors = 2,
               fm = "gls")

print.psych(wisc_faC, cut = .3, sort = TRUE)
```

```
## Factor Analysis using method =  gls
## Call: fa(r = wisc_dat_2, nfactors = 2, fm = "gls")
## Standardized loadings (pattern matrix) based upon correlation matrix
##          item  GLS1  GLS2   h2   u2 com
## info        1  0.84       0.64 0.36 1.0
## vocab       5  0.75       0.58 0.42 1.0
## arith       3  0.60       0.35 0.65 1.0
## simil       4  0.57       0.49 0.51 1.3
## comp        2  0.51  0.31 0.50 0.50 1.7
## digit       6  0.48       0.19 0.81 1.1
## object     10        0.67 0.40 0.60 1.0
## block       9        0.61 0.41 0.59 1.0
## pictcomp    7        0.58 0.38 0.62 1.0
## parang      8        0.41 0.21 0.79 1.1
##
##                        GLS1 GLS2
## SS loadings            2.57 1.59
## Proportion Var         0.26 0.16
## Cumulative Var         0.26 0.42
## Proportion Explained  0.62 0.38
## Cumulative Proportion 0.62 1.00
##
##  With factor correlations of
##      GLS1 GLS2
## GLS1 1.00 0.47
## GLS2 0.47 1.00
##
## Mean item complexity =  1.1
## Test of the hypothesis that 2 factors are sufficient.
##
## df null model =  45  with the objective function =  2.88 with Chi Square =  488.37
## df of  the model are 26  and the objective function was  0.2
##
## The root mean square of the residuals (RMSR) is  0.04
## The df corrected root mean square of the residuals is  0.05
##
## The harmonic n.obs is  175 with the empirical chi square  24.22  with prob <  0.56
## The total n.obs was  175  with Likelihood Chi Square =  33.03  with prob <  0.16
##
```

```
## Tucker Lewis Index of factoring reliability =  0.972
## RMSEA index =  0.039  and the 90 % confidence intervals are  0 0.076
## BIC =  -101.25
## Fit based upon off diagonal values = 0.99
## Measures of factor score adequacy
##                                                    GLS1 GLS2
## Correlation of (regression) scores with factors   0.92 0.86
## Multiple R square of scores with factors          0.85 0.74
## Minimum correlation of possible factor scores     0.70 0.47
```

Examining residuals

So, how do we determine which model is "best"? Can assess how well our "model" fits the data by looking at the residuals (similar to MR when we assessed residuals to see how well our line of best fit "fit" the data!). To examine the residuals, we can create our residual matrix using the actual correlation matrix and our factor loadings that have been extracted from our models.

Mathematically, the residual matrix is calculated by taking the difference of the actual correlation matrix and the correlation matrix reproduced by the model. So let's isolate those from our data.

Step 1: We already have the observed or `actual` correlation matrix for our data set saved as `wisc_cor2` so let's rename this object `wisc_cor_actual` for our residual analysis

```
wisc_cor_actual <- wisc_cor2
wisc_cor_actual
```

```
##               info      comp      arith     simil     vocab      digit
## info     1.0000000 0.4671810 0.49444292 0.5131272 0.6250211 0.34544800
## comp     0.4671810 1.0000000 0.39234279 0.5101537 0.5313307 0.23579916
## arith    0.4944429 0.3923428 1.00000000 0.3693510 0.3873593 0.26901062
## simil    0.5131272 0.5101537 0.36935100 1.0000000 0.5378567 0.25950808
## vocab    0.6250211 0.5313307 0.38735926 0.5378567 1.0000000 0.29424519
## digit    0.3454480 0.2357992 0.26901062 0.2595081 0.2942452 1.00000000
## pictcomp 0.2299054 0.4068811 0.15537391 0.3692838 0.2854177 0.07529928
## parang   0.2015843 0.1865479 0.22667626 0.2981383 0.1321191 0.14819441
## block    0.2291472 0.3690516 0.27212556 0.2613866 0.2974690 0.07275946
## object   0.1848890 0.3223403 0.04292297 0.2687490 0.1853203 0.03470004
##            pictcomp    parang      block     object
## info     0.22990541 0.2015843 0.22914719 0.18488898
## comp     0.40688108 0.1865479 0.36905162 0.32234029
## arith    0.15537391 0.2266763 0.27212556 0.04292297
## simil    0.36928375 0.2981383 0.26138657 0.26874904
## vocab    0.28541774 0.1321191 0.29746897 0.18532030
## digit    0.07529928 0.1481944 0.07275946 0.03470004
## pictcomp 1.00000000 0.2487645 0.38206986 0.36333189
## parang   0.24876445 1.0000000 0.35131414 0.25324989
## block    0.38206986 0.3513141 1.00000000 0.39922185
## object   0.36333189 0.2532499 0.39922185 1.00000000
```

Step 2: Create reproduced correlation matrix using factor loadings from various exploratory models

```
wisc_cor_reproducedA <- factor.model(wisc_fa$loadings)
wisc_cor_reproducedB <- factor.model(wisc_faB$loadings)
wisc_cor_reproducedC <- factor.model(wisc_faC$loadings)
```

Step 3: Create residual correlation matrix by subtracting the reproduced correlation matrices from our actual correlation matrix OR using the `factor.residuals()` function from the `psych` package

```
# Can literally subtract our correlation matrices we have saved as R objects

resid1 <- wisc_cor_actual-wisc_cor_reproducedA
resid2 <- wisc_cor_actual-wisc_cor_reproducedB
resid3 <- wisc_cor_actual-wisc_cor_reproducedC

# Or we can use the `factor.residuals()` function from the `psych` package
# This function requires the actual correlation matrix and the factor/component loadings from your "best" factor model

fresid1 <- factor.residuals(wisc_cor_actual, wisc_fa$loadings)
fresid2 <- factor.residuals(wisc_cor_actual, wisc_faB$loadings)
fresid3 <- factor.residuals(wisc_cor_actual, wisc_faC$loadings)

# Can uncomment these lines of code below to ensure that the SAME data is present across either residual calculation above--
-resulting output/matrix should contain only zeros.
# resid1 - fresid1
# resid2 - fresid2
# resid3 - fresid3
```

With this data, we can assess the proportion of residuals that are greater than +/- 0.05. The “best” models OR models that fit the data “best” ideally would have very few residuals at this level, as better fitting models result in residuals closer and closer to zero.

Now, let’s calculate how many residuals are greater than +/- 0.05 for each of our factor models:

Step 1: Isolate one half of the residual matrices for assessment (remember, these residual matrices are redundant like correlation matrices)

```
# To extract the values from one half of our residual correlation matrix object we use the following code

# ML Extraction
resid1_upper <- as.matrix(fresid1[upper.tri(fresid1)])
resid1_upper
```

```
##                [,1]
##  [1,]  0.085768042
##  [2,] -0.003458659
##  [3,]  0.112753062
##  [4,]  0.056388836
##  [5,]  0.160957446
##  [6,]  0.040575882
##  [7,] -0.004856381
##  [8,]  0.154498241
##  [9,] -0.056674147
## [10,]  0.105093348
## [11,] -0.061436405
## [12,]  0.045795892
## [13,] -0.013042736
## [14,]  0.018845651
## [15,] -0.058061381
## [16,]  0.229131666
## [17,]  0.179632088
## [18,]  0.134533271
## [19,]  0.203647042
## [20,]  0.222497147
## [21,]  0.119070143
## [22,]  0.176397327
## [23,]  0.021618569
## [24,]  0.195509511
## [25,]  0.171731403
## [26,]  0.068286314
## [27,]  0.163316554
## [28,]  0.009436685
## [29,]  0.243465497
## [30,]  0.143048012
## [31,]  0.261227538
## [32,]  0.100452348
## [33,]  0.246024122
## [34,]  0.126488820
## [35,]  0.012830646
## [36,]  0.105304437
## [37,]  0.305243283
## [38,]  0.145520485
## [39,]  0.105499671
## [40,]  0.170290919
## [41,]  0.225597126
## [42,]  0.150516272
## [43,] -0.017838759
## [44,]  0.002949098
## [45,]  0.004002117
```

```
# PAF Extraction
resid2_upper <- as.matrix(fresid2[upper.tri(fresid2)])
resid2_upper
```

```
##              [,1]
##  [1,]  0.070709523
##  [2,] -0.010136095
##  [3,]  0.099793146
##  [4,]  0.050915085
##  [5,]  0.154273959
##  [6,]  0.032865321
##  [7,]  0.005995246
##  [8,]  0.149768405
##  [9,] -0.055219032
## [10,]  0.108006142
## [11,] -0.062348159
## [12,]  0.034379145
## [13,] -0.018309100
## [14,]  0.013904453
## [15,] -0.053708624
## [16,]  0.215025875
## [17,]  0.188347003
## [18,]  0.126969733
## [19,]  0.201456767
## [20,]  0.214015372
## [21,]  0.108837808
## [22,]  0.154371987
## [23,]  0.019064110
## [24,]  0.181178598
## [25,]  0.161254714
## [26,]  0.051759291
## [27,]  0.149352593
## [28,]  0.020053090
## [29,]  0.237423907
## [30,]  0.147690986
## [31,]  0.258826120
## [32,]  0.096775726
## [33,]  0.241859602
## [34,]  0.122736497
## [35,]  0.017651485
## [36,]  0.107086397
## [37,]  0.311501938
## [38,]  0.159908538
## [39,]  0.113040054
## [40,]  0.175769609
## [41,]  0.230815221
## [42,]  0.153998279
## [43,] -0.010044654
## [44,]  0.009205838
## [45,] -0.006441748
```

```
# GLS Extraction
resid3_upper <- as.matrix(fresid3[upper.tri(fresid3)])
resid3_upper
```

```
##              [,1]
##  [1,]  0.067006551
##  [2,] -0.009452291
##  [3,]  0.096420430
##  [4,]  0.052408695
##  [5,]  0.152929073
##  [6,]  0.032587771
##  [7,] -0.002985038
##  [8,]  0.142868249
##  [9,] -0.063720097
## [10,]  0.103128906
## [11,] -0.064415086
## [12,]  0.030164350
## [13,] -0.021509720
## [14,]  0.012384445
## [15,] -0.063634799
## [16,]  0.208961222
## [17,]  0.188848121
## [18,]  0.123746064
## [19,]  0.198070284
## [20,]  0.213728255
## [21,]  0.107151658
## [22,]  0.163621147
## [23,]  0.016875738
## [24,]  0.187586305
## [25,]  0.160144514
## [26,]  0.060210895
## [27,]  0.157774532
## [28,]  0.004384099
## [29,]  0.224996658
## [30,]  0.149323039
## [31,]  0.251582962
## [32,]  0.092564286
## [33,]  0.237838735
## [34,]  0.116762047
## [35,]  0.025012092
## [36,]  0.094598626
## [37,]  0.313617588
## [38,]  0.161447437
## [39,]  0.115934796
## [40,]  0.174060260
## [41,]  0.239700743
## [42,]  0.159413565
## [43,] -0.016912614
## [44,] -0.015812752
## [45,] -0.004988488
```

Step 2: Assess absolute value of each residual against 0.05 and sum to assess number of residuals greater than 0.05:

```
# abs() calculates the absolute value for object in argument
resid1_magnitude <- abs(resid1_upper) > 0.05
resid2_magnitude <- abs(resid2_upper) > 0.05
resid3_magnitude <- abs(resid3_upper) > 0.05

# Gives us a count of how many residuals are greater than 0.05
sum(resid1_magnitude)
```

```
## [1] 33
```

```
sum(resid2_magnitude)
```

```
## [1] 33
```

```
sum(resid3_magnitude)
```

```
## [1] 33
```

Step 3: Calculate proportion of residuals > 0.05 with the following code:

```
# nrow calculates the number of rows in a data frame or matrix

sum(resid1_magnitude)/nrow(resid1_upper)
```

```
## [1] 0.7333333
```

```
sum(resid2_magnitude)/nrow(resid2_upper)
```

```
## [1] 0.7333333
```

```
sum(resid3_magnitude)/nrow(resid3_upper)
```

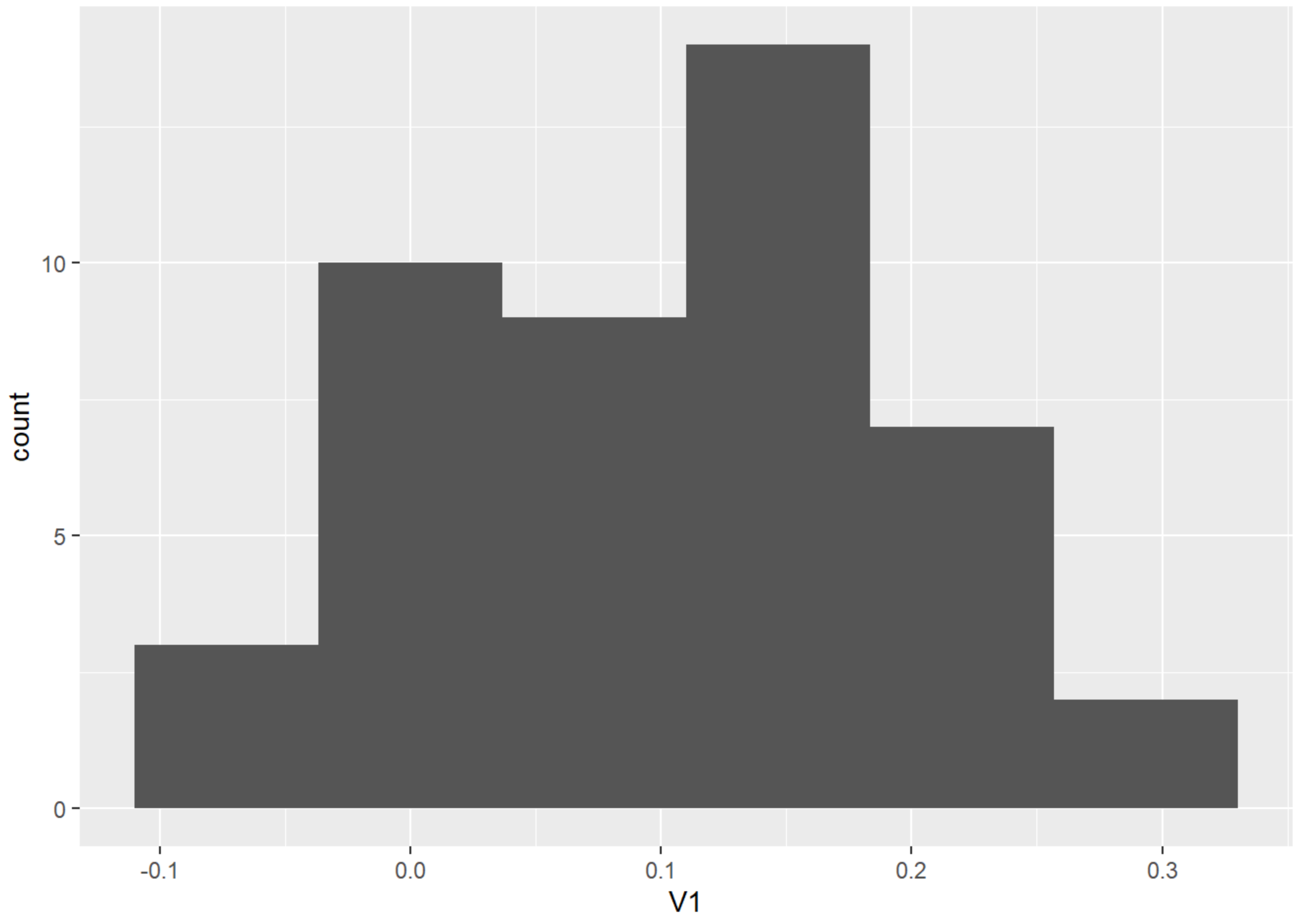
```
## [1] 0.7333333
```

Step 4: Finally, we can calculate the root mean square residual (RMSR) and plot a histogram of the residuals:

```
# ML Extraction
sqrt(mean(resid1_upper^2))
```

```
## [1] 0.1423376
```

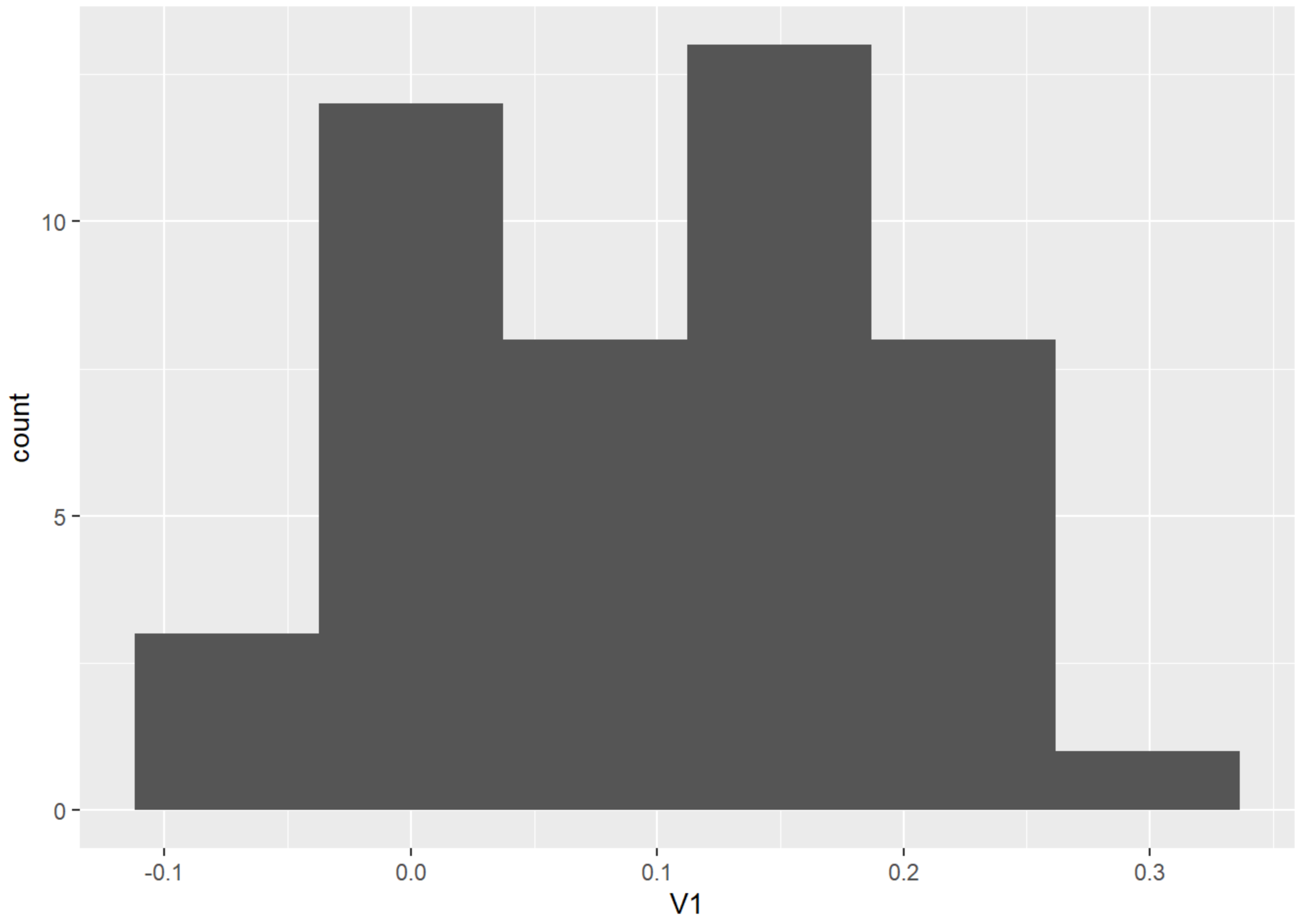
```
as.data.frame(resid1_upper) %>%
ggplot(aes(x = V1)) +
  geom_histogram(bins = 6)
```



```
# PAF Extraction
sqrt(mean(resid2_upper^2))
```

```
## [1] 0.1395267
```

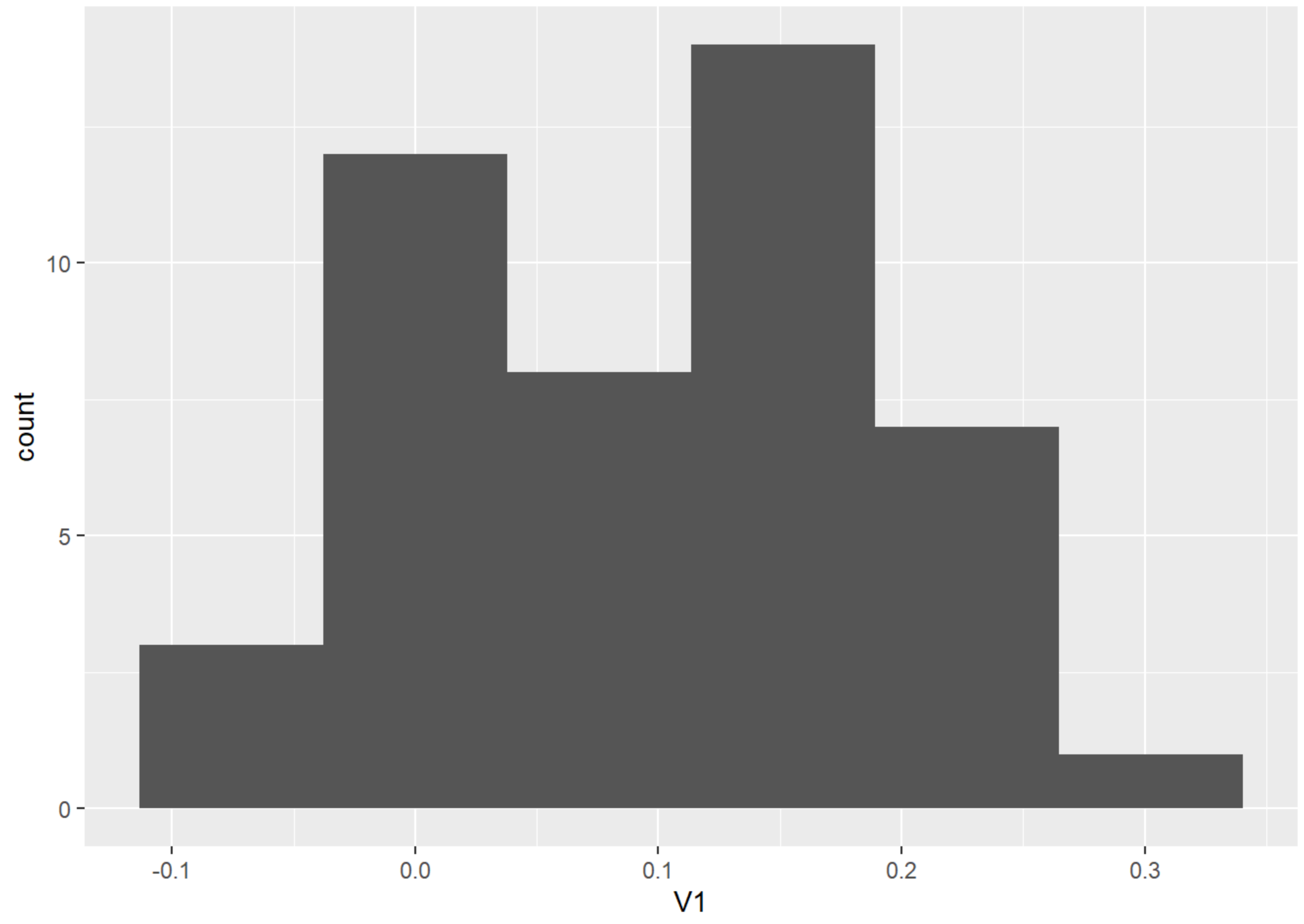
```
as.data.frame(resid2_upper) %>%
ggplot(aes(x = V1)) +
  geom_histogram(bins = 6)
```



```
# GLS Extraction
sqrt(mean(resid3_upper^2))
```

```
## [1] 0.1390063
```

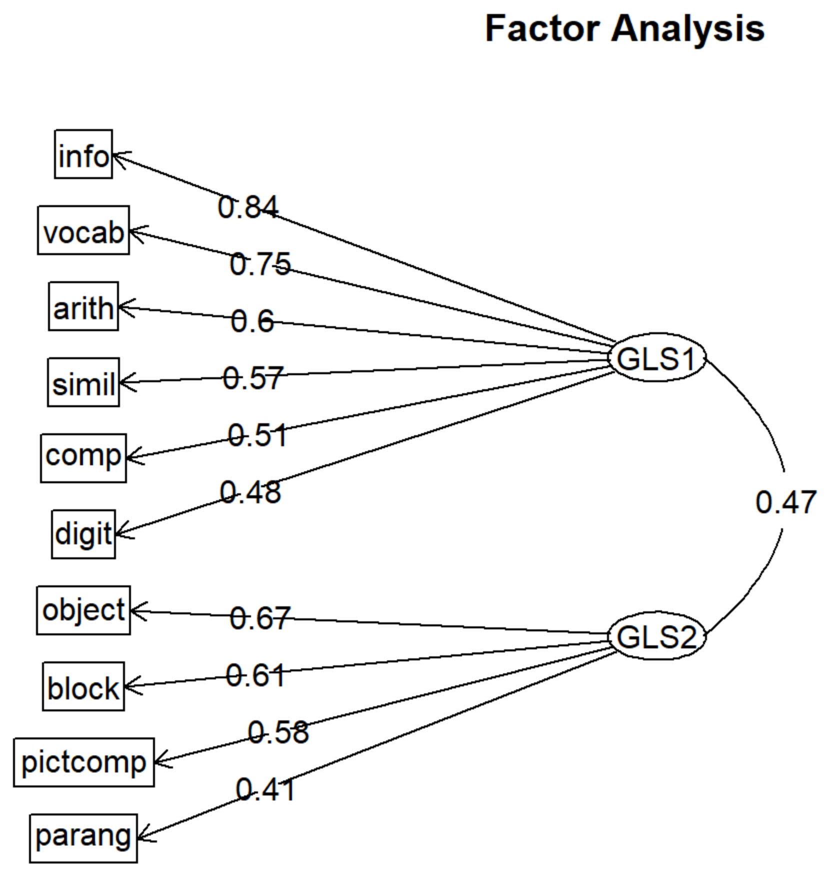
```
as.data.frame(resid3_upper) %>%
ggplot(aes(x = V1)) +
  geom_histogram(bins = 6)
```



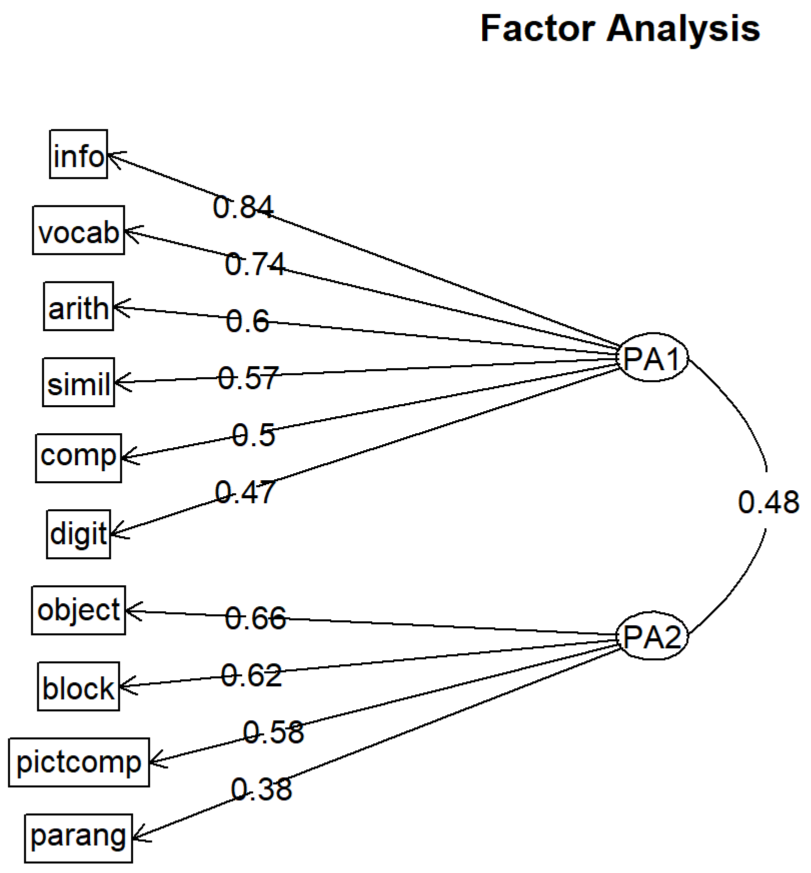
The plot of the residuals is not great. Some of the residuals are in the -.1 to .3 range, however, the majority of the residuals are greater than .05.

Something new that uses exploratory graph analysis–a complimentary alternative to exploratory factor analysis

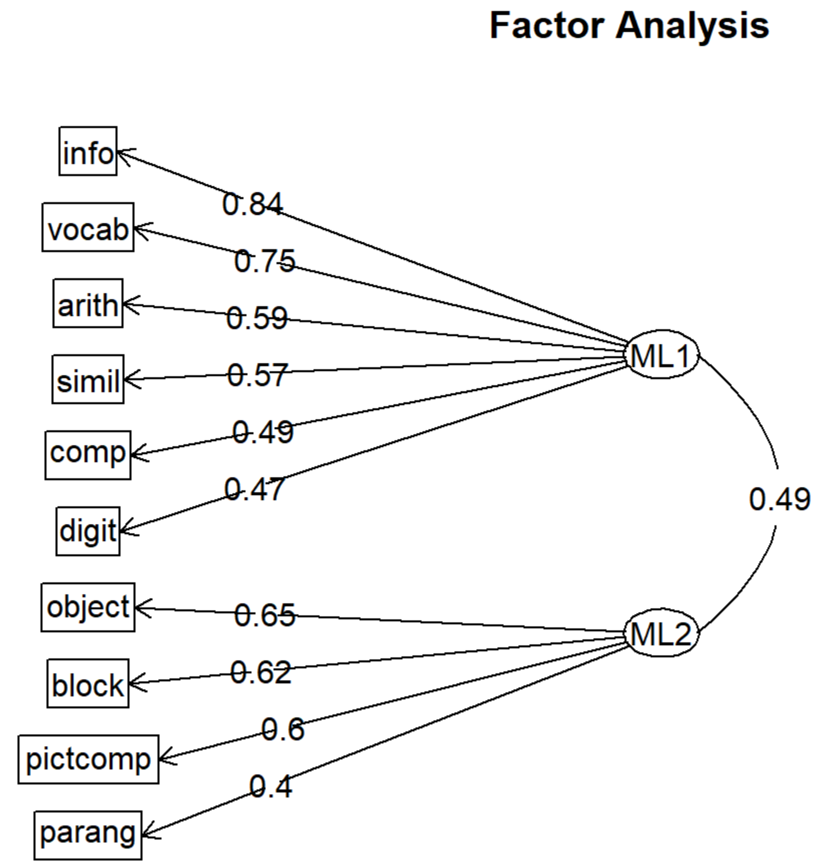
```
# Best Model == GLS Model
fa.diagram(wisc_faC,
           digits = 2,
           cut = .3,
           sort = TRUE)
```



```
# Additional Models == PAF Model
fa.diagram(wisc_faB,
           digits = 2,
           cut = .3,
           sort = TRUE)
```



```
# Additional Models == ML Model
fa.diagram(wisc_fa,
           digits = 2,
           cut = .3,
           sort = TRUE)
```


Factor Analysis
info
vocab
arith
simil
comp
digit
object
block
pictcomp
parang
ML1
ML2
0.84
0.75
0.59
0.57
0.49
0.47
0.65
0.62
0.6
0.4
0.49