Principal Components Analysis; Exploratory Factor Analysis; Conducting Exploratory Analyses with CFA (or…Why I Hate EFA)

EPSY 905: Multivariate Analysis

Spring 2016

Lecture #13 – April 27, 2016

Today's Class

- Methods for exploratory factor analysis (EFA)
  - Principal Components-based (TERRIBLE)
  - Maximum Likelihood-based Exploratory Factor Analysis (BAD)
  - Exploratory Structural Equation Modeling (ALSO BAD)

- Comparisons of CFA and EFA

- How to do exploratory analyses with CFA
  - Structure of no items known
  - Structure of some items known

The Logic of Exploratory Analyses

- Exploratory analyses attempt to discover hidden structure in data with little to no user input
  - Aside from the selection of analysis and estimation

- The results from exploratory analyses can be misleading
  - If data do not meet assumptions of model or method selected
  - If data have quirks that are idiosyncratic to the sample selected
  - If some cases are extreme relative to others
  - If constraints made by analysis are implausible

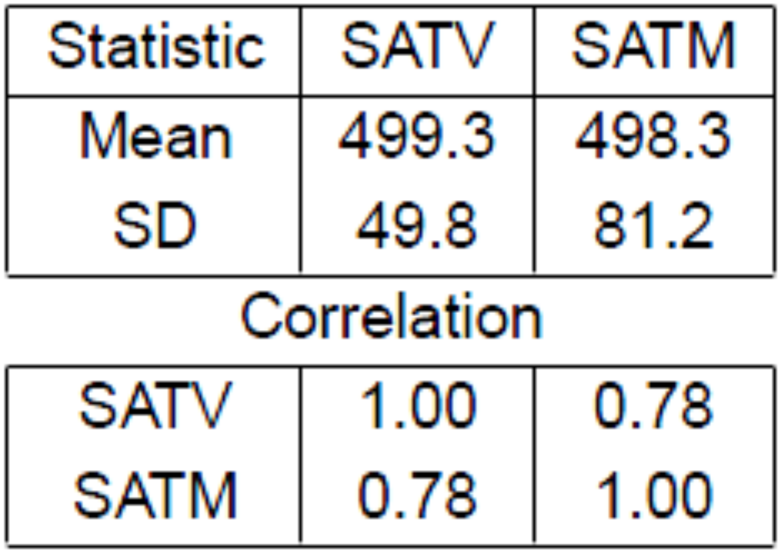
- Sometimes, exploratory analyses are needed
  - Must construct an analysis that capitalizes on the known features of data
  - There are better ways to conduct such analyses

- Often, exploratory analyses are not needed
  - But are conducted anyway – see a lot of reports of scale development that start with the idea that a construct has a certain number of dimensions

ADVANCED MATRIX OPERATIONS

A Guiding Example

- To demonstrate some advanced matrix algebra, we will make use of data
- I collected data SAT test scores for both the Math (SATM) and Verbal (SATV) sections of 1,000 students
- The descriptive statistics of this data set are given below:



Statistic	SATV	SATM
Mean	499.3	498.3
SD	49.8	81.2
Correlation		
SATV	1.00	0.78
SATM	0.78	1.00

Matrix Trace

- For a square matrix $\mathbf{\Sigma}$ with *p* rows/columns, the trace is the sum of the diagonal elements:

$$tr\mathbf{\Sigma} = \sum_{i=1}^{p} a_{ii}$$

- For our data, the trace of the correlation matrix is 2
  - For all correlation matrices, the trace is equal to the number of variables because all diagonal elements are 1

- The trace will be considered the total variance in principal components analysis
  - Used as a target to recover when applying statistical models

Matrix Determinants

- A square matrix can be characterized by a scalar value called a determinant:

$$\det \boldsymbol{\Sigma} = |\boldsymbol{\Sigma}|$$

- Calculation of the determinant by hand is tedious
  - Our determinant was 0.3916
  - Computers can have difficulties with this calculation (unstable in cases)
- The determinant is useful in statistics:
  - Shows up in multivariate statistical distributions
  - Is a measure of “generalized” variance of multiple variables
- If the determinant is positive, the matrix is called **positive definite**
  - Is invertable
- If the determinant is not positive, the matrix is called **non-positive definite**
  - Not invertable

Matrix Orthogonality

- A square matrix $\mathbf{\Lambda}$ is said to be orthogonal if:

$$\mathbf{\Lambda}\mathbf{\Lambda}^T = \mathbf{\Lambda}^T\mathbf{\Lambda} = \mathbf{I}$$

- Orthogonal matrices are characterized by two properties:
  1. The dot product of all row vector multiples is the zero vector
     - Meaning vectors are orthogonal (or uncorrelated)
  2. For each row vector, the sum of all elements is one
     - Meaning vectors are "normalized"

- The matrix above is also called **orthonormal**
  - **The diagonal is equal to 1 (each vector has a unit length)**

- Orthonormal matrices are used in principal components and exploratory factor analysis

Eigenvalues and Eigenvectors

- A square matrix $\boldsymbol{\Sigma}$ can be decomposed into a set of eigenvalues $\boldsymbol{\lambda}$ and a set of eigenvectors $\mathbf{e}$

$$\boldsymbol{\Sigma}\mathbf{e} = \lambda\mathbf{e}$$

- Each eigenvalue has a corresponding eigenvector
  - The number equal to the number of rows/columns of $\boldsymbol{\Sigma}$
  - The eigenvectors are all orthogonal

- Principal components analysis uses eigenvalues and eigenvectors to reconfigure data

Eigenvalues and Eigenvectors Example

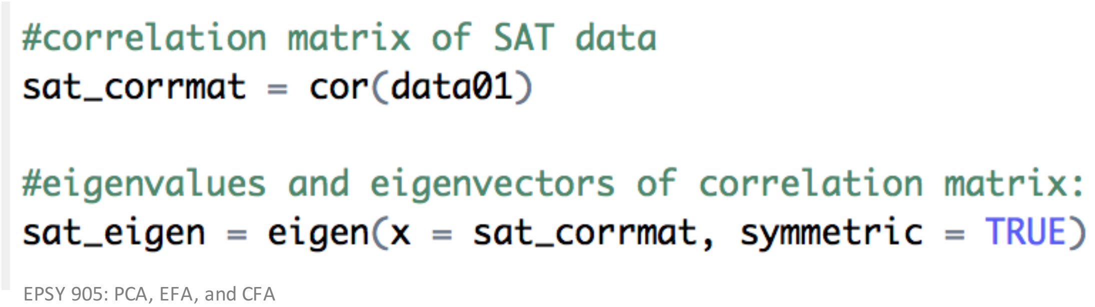
- In our SAT example, the two eigenvalues obtained were:

$$\lambda_1 = 1.78$$
$$\lambda_2 = 0.22$$

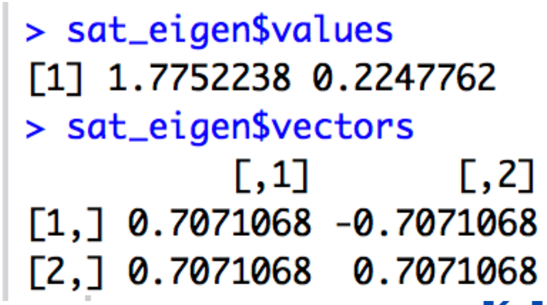
- The two eigenvectors obtained were:

$$\mathbf{e}_1 = \begin{bmatrix} 0.71 \\ 0.71 \end{bmatrix}; \mathbf{e}_2 = \begin{bmatrix} 0.71 \\ -0.71 \end{bmatrix}$$

- These terms will have much greater meaning principal components analysis



```
#correlation matrix of SAT data
sat_corrmat = cor(data01)

#eigenvalues and eigenvectors of correlation matrix:
sat_eigen = eigen(x = sat_corrmat, symmetric = TRUE)
```



```
> sat_eigen$values
[1] 1.7752238 0.2247762
> sat_eigen$vectors
          [,1]       [,2]
[1,] 0.7071068 -0.7071068
[2,] 0.7071068  0.7071068
```

Spectral Decomposition

- Using the eigenvalues and eigenvectors, we can reconstruct the original matrix using a spectral decomposition:

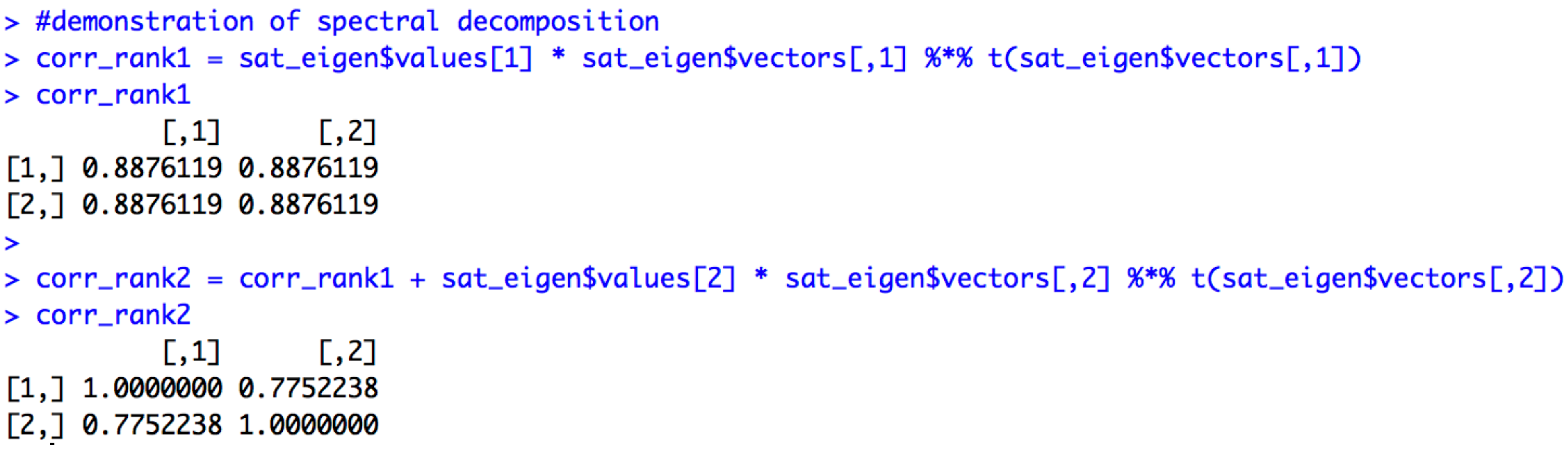
$$\mathbf{\Sigma} = \sum_{i=1}^{p} \lambda_i \mathbf{e}_i \mathbf{e}_i^T$$

- For our example, we can get back to our original matrix:

$$\mathbf{R}_1 = \lambda_1 \mathbf{e}_1 \mathbf{e}_1^T = 1.78 \begin{bmatrix} .71 \\ .71 \end{bmatrix} \begin{bmatrix} .71 & .71 \end{bmatrix} = \begin{bmatrix} .89 & .89 \\ .89 & .89 \end{bmatrix}$$

$$\mathbf{R}_2 = \mathbf{R}_1 + \lambda_2 \mathbf{e}_2 \mathbf{e}_2^T$$
$$= \begin{bmatrix} .89 & .89 \\ .89 & .89 \end{bmatrix} + 0.22 \begin{bmatrix} .71 \\ -.71 \end{bmatrix} \begin{bmatrix} .71 & -.71 \end{bmatrix} = \begin{bmatrix} 1.00 & 0.78 \\ 0.78 & 1.00 \end{bmatrix}$$

Spectral Decomposition in R



```
> #demonstration of spectral decomposition
> corr_rank1 = sat_eigen$values[1] * sat_eigen$vectors[,1] %*% t(sat_eigen$vectors[,1])
> corr_rank1
          [,1]      [,2]
[1,] 0.8876119 0.8876119
[2,] 0.8876119 0.8876119
> 
> corr_rank2 = corr_rank1 + sat_eigen$values[2] * sat_eigen$vectors[,2] %*% t(sat_eigen$vectors[,2])
> corr_rank2
          [,1]      [,2]
[1,] 1.0000000 0.7752238
[2,] 0.7752238 1.0000000
```

Additional Eigenvalue Properties

- The matrix trace is the sum of the eigenvalues:

$$tr\boldsymbol{\Sigma} = \sum_{i=1}^{p} \lambda_i$$

  - In our example, the $tr\mathbf{R} = 1.78 + .22 = 2$

- The matrix determinant can be found by the product of the eigenvalues

$$|\boldsymbol{\Sigma}| = \prod_{i=1}^{p} \lambda_i$$

  - In our example $|\mathbf{R}| = 1.78 * .22 = .3916$

AN INTRODUCTION TO PRINCIPAL COMPONENTS ANALYSIS

PCA Overview

- Principal Components Analysis (PCA) is a method for re-expressing the covariance (or often correlation) between a set of variables
  - ➢ The re-expression comes from creating a set of new variables (linear combinations) of the original variables

- PCA has two objectives:
  1. Data reduction
     - ◆ Moving from many original variables down to a few “components”
  2. Interpretation
     - ◆ Determining which original variables contribute most to the new “components”

Goals of PCA

- The goal of PCA is to find a set of *k* principal components (composite variables) that:
  - Is much smaller in number than the original set of *V* variables
  - Accounts for nearly all of the total variance
    - Total variance = trace of covariance/correlation matrix

- If these two goals can be accomplished, then the set of *k* principal components contains almost as much information as the original *V* variables
  - Meaning – the components can now replace the original variables in any subsequent analyses

Questions when using PCA

- PCA analyses proceed by seeking the answers to two questions:

1. How many components (new variables) are needed to "adequately" represent the original data?
   - ➢ The term adequately is fuzzy (and will be in the analysis)

2. (once #1 has been answered): What does each component represent?
   - ➢ The term "represent" is also fuzzy

PCA Features

- PCA often reveals relationships between variables that were not previously suspected
  - New interpretations of data and variables often stem from PCA

- PCA usually serves as more of a means to an end rather than an end it itself
  - Components (the new variables) are often used in other statistical techniques
    - Multiple regression/ANOVA
    - Cluster analysis

- Unfortunately, PCA is often intermixed with Exploratory Factor Analysis
  - Don’t. Please don’t. Please make it stop.

PCA Details

- Notation: Z are our new components and $\mathbf{Y}$ is our original data matrix (with *N* observations and *V* variables)
  - We will let *p* be our index for a subject

- The new components are linear combinations:

$$
\begin{gathered}
Z_{p1} = \mathbf{e}_1^T\mathbf{Y} = e_{11}Y_{p1} + e_{21}Y_{p2} + \cdots + e_{V1}Y_{pV} \\
Z_{p2} = \mathbf{e}_2^T\mathbf{Y} = e_{12}Y_{p1} + e_{22}Y_{p2} + \cdots + e_{V2}Y_{pV} \\
\vdots \\
Z_{pV} = \mathbf{e}_V^T\mathbf{Y} = e_{1V}Y_{p1} + e_{2V}Y_{p2} + \cdots + e_{VV}Y_{pV}
\end{gathered}
$$

- The weights of the components (e_{jk}) come from the eigenvectors of the covariance or correlation matrix for component k and variable j

Details About the Components

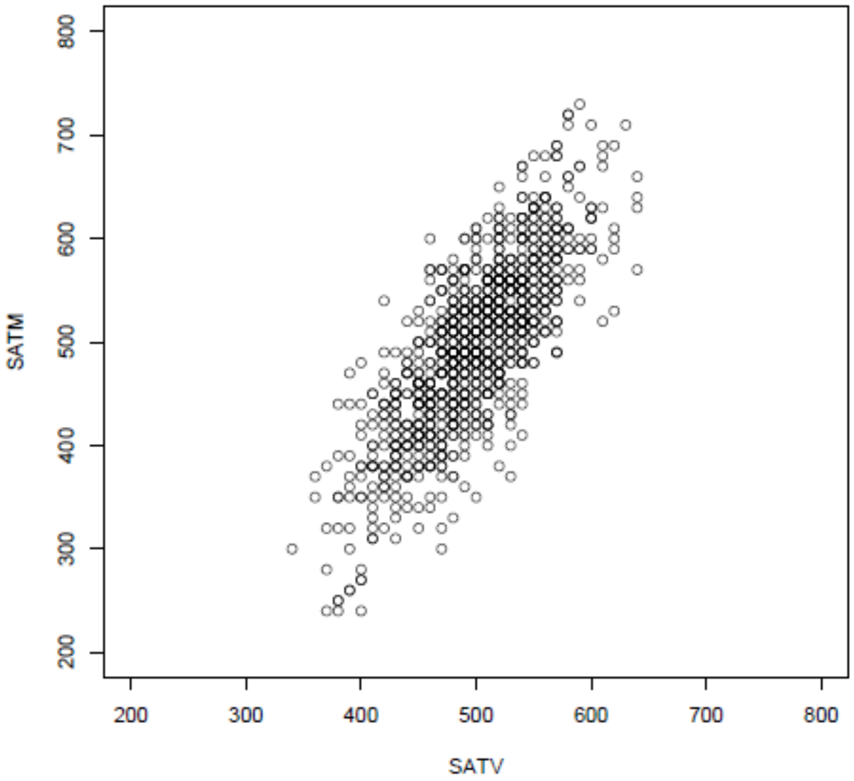
- The components (Z) are formed by the weights of the eigenvectors of the covariance or correlation matrix of the original data
  - The variance of a component is given by the eigenvalue associated with the eigenvector for the component

- Using the eigenvalue and eigenvectors means:
  - Each successive component has lower variance
    - $\text{Var}(Z_1) > \text{Var}(Z_2) > \ldots > \text{Var}(Z_v)$
  - All components are uncorrelated
  - The sum of the variances of the principal components is equal to the total variance:

$$\sum_{v=1}^{V} Var(Z_v) = tr\mathbf{\Sigma} = \sum_{v=1}^{V} \lambda_v$$

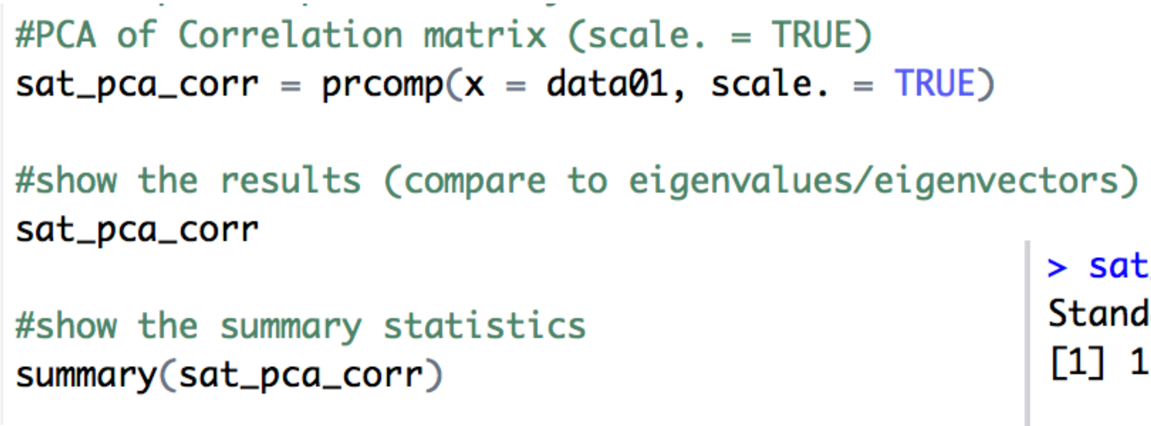
PCA on our Example

- We will now conduct a PCA on the **correlation matrix** of our sample data
  - ➢ This example is given for demonstration purposes – typically we will not do PCA on small numbers of variables

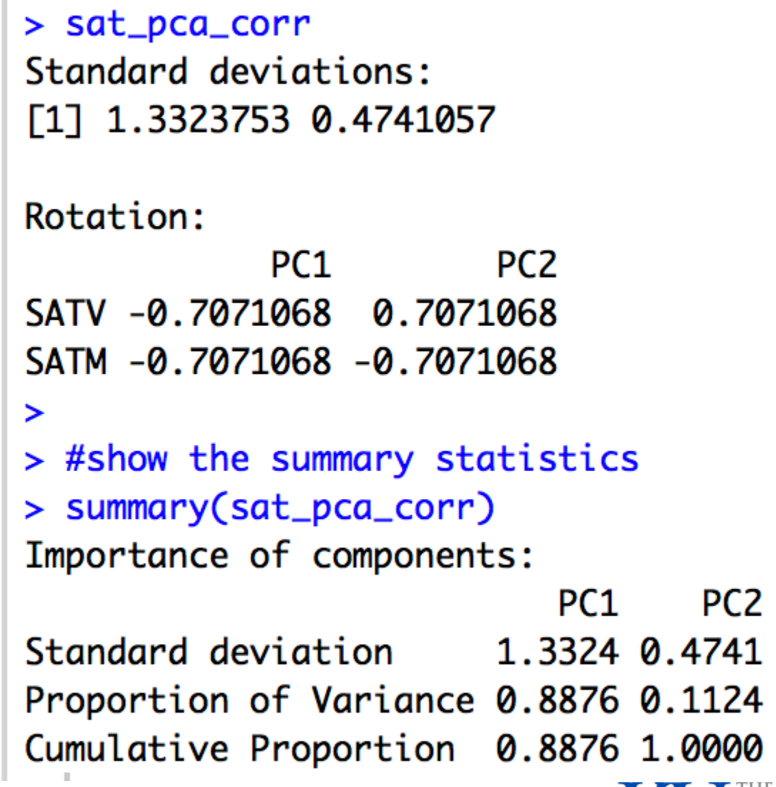


PCA in R

- The R function that does principal components is called prcomp()



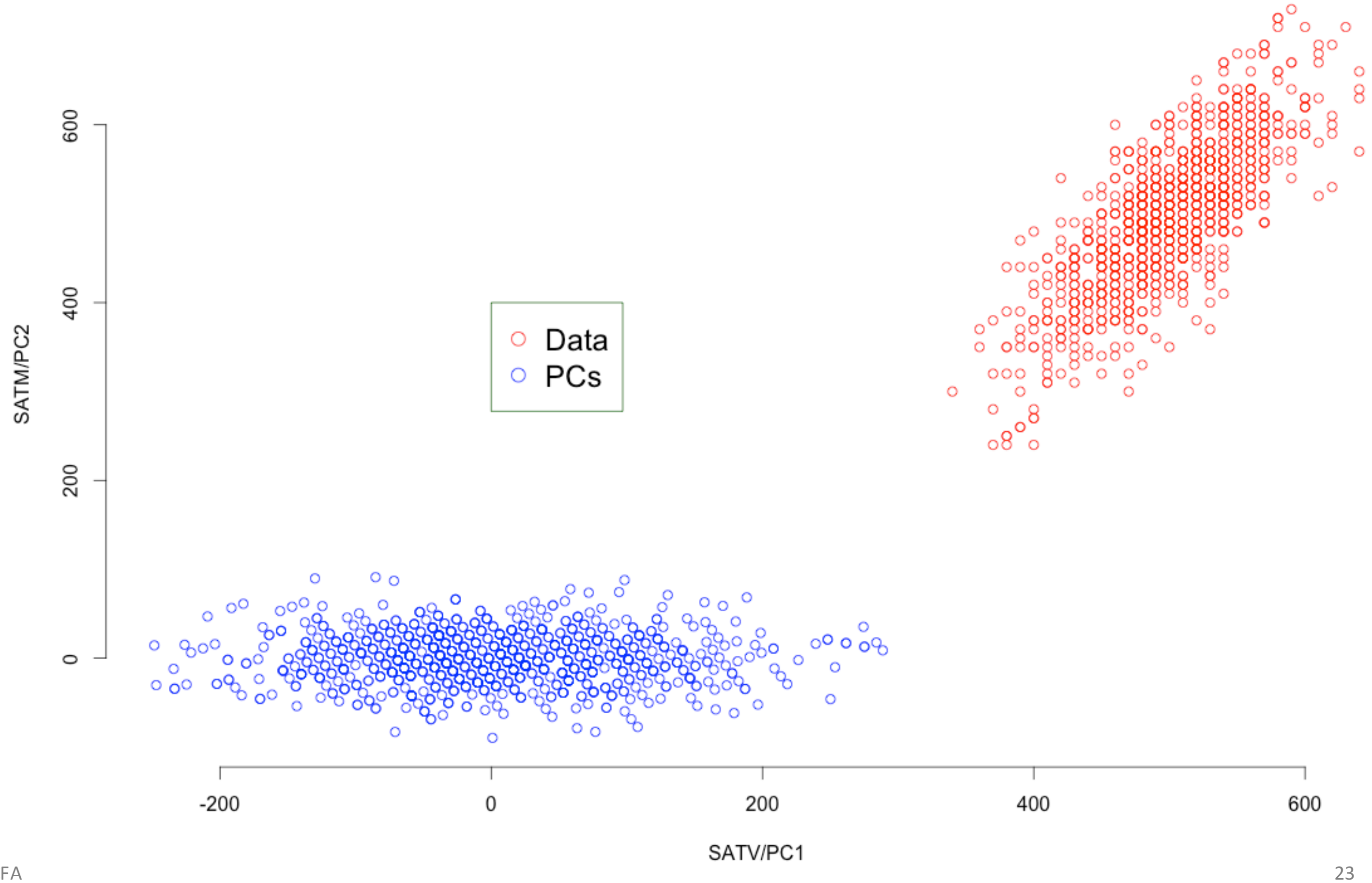
```
#PCA of Correlation matrix (scale. = TRUE)
sat_pca_corr = prcomp(x = data01, scale. = TRUE)

#show the results (compare to eigenvalues/eigenvectors)
sat_pca_corr

#show the summary statistics
summary(sat_pca_corr)
```



```
> sat_pca_corr
Standard deviations:
[1] 1.3323753 0.4741057

Rotation:
            PC1        PC2
SATV -0.7071068  0.7071068
SATM -0.7071068 -0.7071068
>
> #show the summary statistics
> summary(sat_pca_corr)
Importance of components:
                          PC1    PC2
Standard deviation     1.3324 0.4741
Proportion of Variance 0.8876 0.1124
Cumulative Proportion  0.8876 1.0000
```

Graphical Representation

- Plotting the components and the original data side by side reveals the nature of PCA:
  - Shown from PCA of covariance matrix



The Growth of Gambling Access

- In past 25 years:
  - An exponential increase in the accessibility of gambling
  - An increased rate of with problem or pathological gambling (Volberg, 2002, Welte et al., 2009)

- Hence, there is a need to better:
  - Understand the underlying causes of the disorder
  - Reliably identify potential pathological gamblers
  - Provide effective treatment interventions

Pathological Gambling: DSM Definition

- To be diagnosed as a pathological gambler, an individual must meet 5 of 10 defined criteria:

1. Is preoccupied with gambling
2. Needs to gamble with increasing amounts of money in order to achieve the desired excitement
3. Has repeated unsuccessful efforts to control, cut back, or stop gambling
4. Is restless or irritable when attempting to cut down or stop gambling
5. Gambles as a way of escaping from problems or relieving a dysphoric mood
6. After losing money gambling, often returns another day to get even
7. Lies to family members, therapist, or others to conceal the extent of involvement with gambling
8. Has committed illegal acts such as forgery, fraud, theft, or embezzlement to finance gambling
9. Has jeopardized or lost a significant relationship, job, educational, or career opportunity because of gambling
10. Relies on others to provide money to relieve a desperate financial situation caused by gambling

Research on Pathological Gambling

- In order to study the etiology of pathological gambling, more variability in responses was needed

- The Gambling Research Instrument (Feasel, Henson, & Jones, 2002) was created with 41 Likert-type items
  - Items were developed to measure each criterion

- Example items (ratings: Strongly Disagree to Strongly Agree):
  - I worry that I am spending too much money on gambling (C3)
  - There are few things I would rather do than gamble (C1)

- The instrument was used on a sample of experienced gamblers from a riverboat casino in a Flat Midwestern State
  - Casino patrons were solicited after playing roulette

The GRI Items

- The GRI used a 6-point Likert scale
  - 1: Strongly Disagree
  - 2: Disagree
  - 3: Slightly Disagree
  - 4: Slightly Agree
  - 5: Agree
  - 6: Strongly Agree

- To meet the assumptions of factor analysis, we will treat these responses as being continuous
  - This is tenuous at best, but often is the case in factor analysis
  - Categorical items would be better….but you'd need another course for how to do that
    - Hint: Item Response Models

The Sample

- Data were collected from two sources:
  - 112 “experienced” gamblers
    - Many from an actual casino
  - 1192 college students from a “rectangular” midwestern state
    - Many never gambled before

- Today, we will combine both samples and treat them as homogenous – one sample of 1304 subjects

Final 10 Items on the Scale

Item	Criterion	Question
GRI1	3	I would like to cut back on my gambling.
GRI3	6	If I lost a lot of money gambling one day, I would be more likely to want to play again the following day.
GRI5	2	I find it necessary to gamble with larger amounts of money (than when I first gambled) for gambling to be exciting.
GRI9	4	I feel restless when I try to cut down or stop gambling.
GRI10	1	It bothers me when I have no money to gamble.
GRI13	3	I find it difficult to stop gambling.
GRI14	2	I am drawn more by the thrill of gambling than by the money I could win.
GRI18	9	My family, coworkers, or others who are close to me disapprove of my gambling.
GRI21	1	It is hard to get my mind off gambling.
GRI23	5	I gamble to improve my mood.

PCA with Gambling Items

- To show how PCA works with a larger set of items, we will examine the 10 GRI items (the ones that fit a one-factor CFA model)

- **TO DO THIS YOU MUST IMAGINE:**
  - THESE WERE THE ONLY 10 ITEMS YOU HAD
  - YOU WANTED TO REDUCE THE 10 ITEMS INTO 1 OR 2 COMPONENT VARIABLES

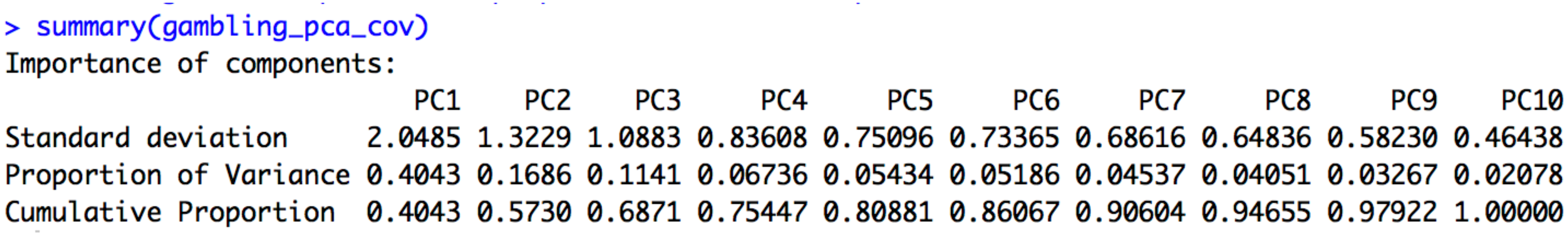
- CAPITAL LETTERS ARE USED AS YOU SHOULD NEVER DO A PCA AFTER RUNNING A CFA – THEY ARE FOR DIFFERENT PURPOSES!

Question #1: How Many Components?

- To answer the question of how many components, two methods are used:
  - Scree plot of eigenvalues (looking for the “elbow”)
  - Variance accounted for (should be > 70%)

- We will go with 4 components: (variance accounted for VAC = 75%)

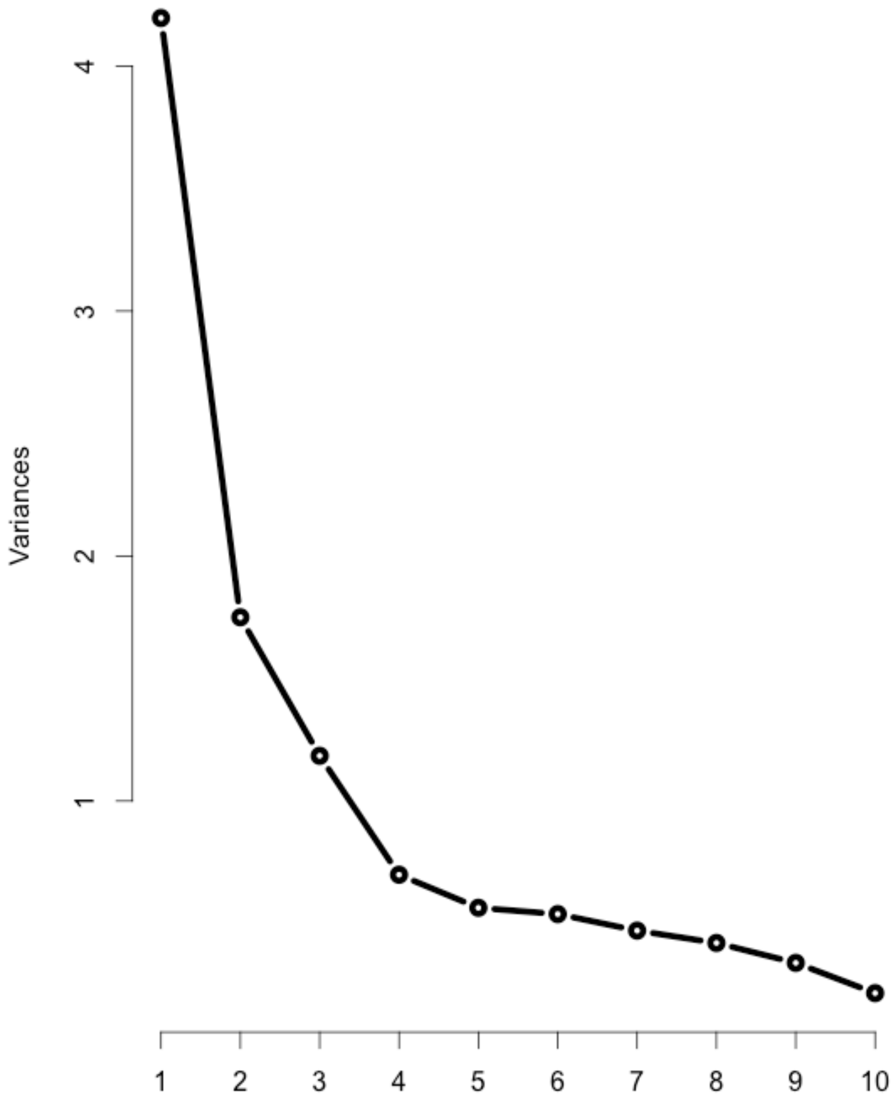
- Variance accounted for is for the **total sample variance**



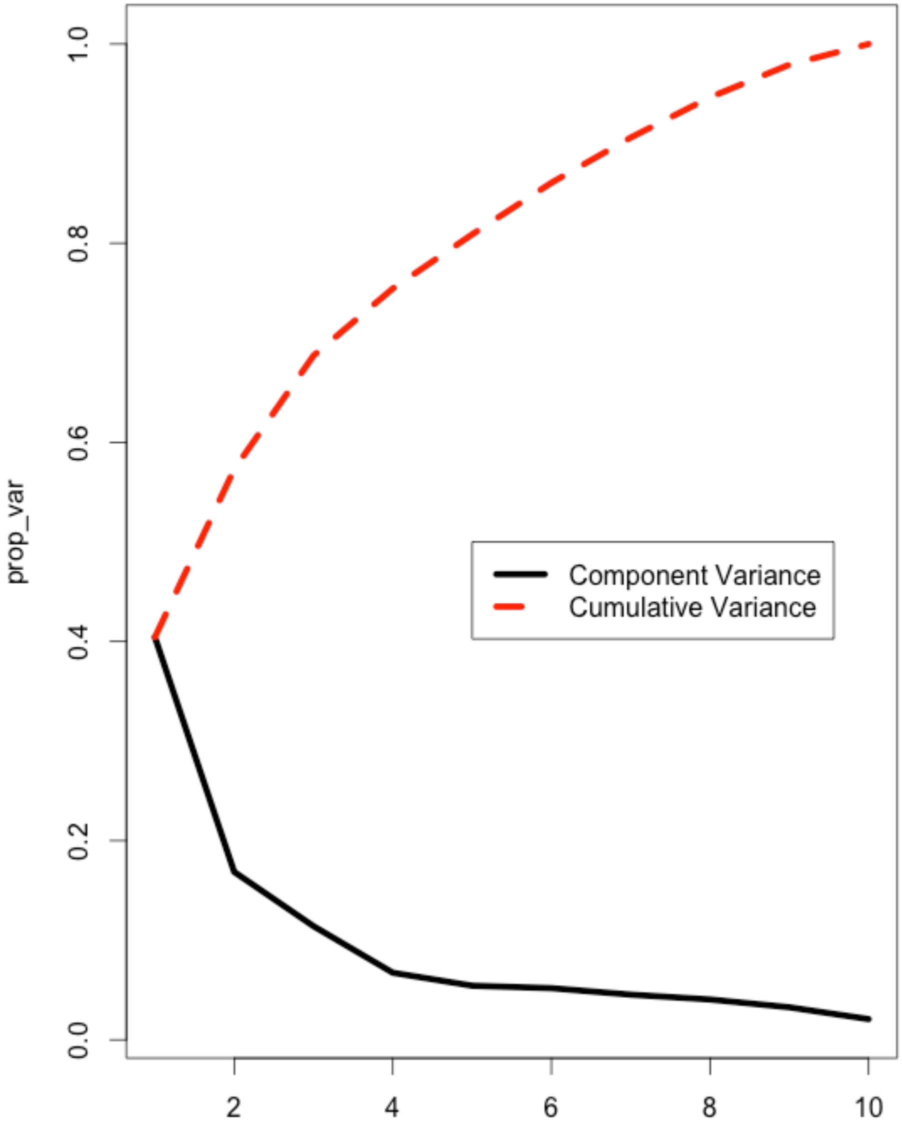
```
> summary(gambling_pca_cov)
Importance of components:
                          PC1    PC2    PC3     PC4     PC5     PC6     PC7     PC8     PC9    PC10
Standard deviation     2.0485 1.3229 1.0883 0.83608 0.75096 0.73365 0.68616 0.64836 0.58230 0.46438
Proportion of Variance 0.4043 0.1686 0.1141 0.06736 0.05434 0.05186 0.04537 0.04051 0.03267 0.02078
Cumulative Proportion  0.4043 0.5730 0.6871 0.75447 0.80881 0.86067 0.90604 0.94655 0.97922 1.00000
```

Plots to Answer How Many Components

Scree Plot of PCA Eigenvalues

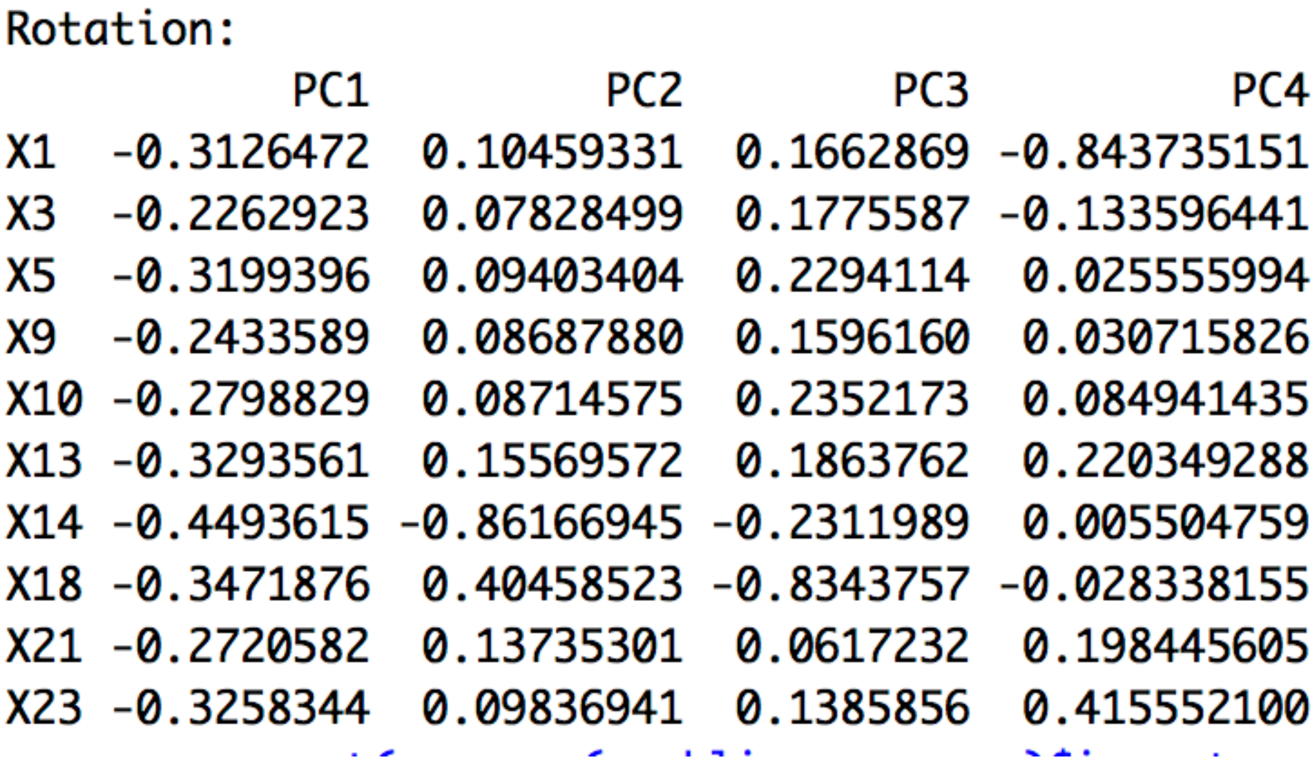


Proportion of Variance Explained by Component



Question #2:
What Does Each Component Represent?

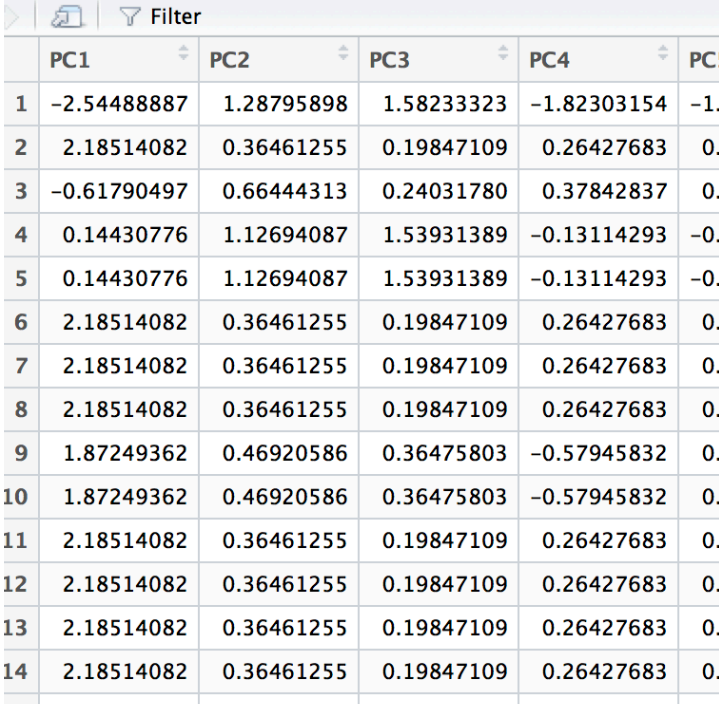
- To answer question #2 – we look at the weights of the eigenvectors (here is the unrotated solution)



```
Rotation:
           PC1         PC2        PC3          PC4
X1  -0.3126472  0.10459331  0.1662869 -0.843735151
X3  -0.2262923  0.07828499  0.1775587 -0.133596441
X5  -0.3199396  0.09403404  0.2294114  0.025555994
X9  -0.2433589  0.08687880  0.1596160  0.030715826
X10 -0.2798829  0.08714575  0.2352173  0.084941435
X13 -0.3293561  0.15569572  0.1863762  0.220349288
X14 -0.4493615 -0.86166945 -0.2311989  0.005504759
X18 -0.3471876  0.40458523 -0.8343757 -0.028338155
X21 -0.2720582  0.13735301  0.0617232  0.198445605
X23 -0.3258344  0.09836941  0.1385856  0.415552100
```

Final Result: Four Principal Components

- Using the weights of the eigenvectors, we can create four new variables – the four principal components



	PC1	PC2	PC3	PC4	PC
1	-2.54488887	1.28795898	1.58233323	-1.82303154	-1.
2	2.18514082	0.36461255	0.19847109	0.26427683	0.
3	-0.61790497	0.66444313	0.24031780	0.37842837	0.
4	0.14430776	1.12694087	1.53931389	-0.13114293	-0.
5	0.14430776	1.12694087	1.53931389	-0.13114293	-0.
6	2.18514082	0.36461255	0.19847109	0.26427683	0.
7	2.18514082	0.36461255	0.19847109	0.26427683	0.
8	2.18514082	0.36461255	0.19847109	0.26427683	0.
9	1.87249362	0.46920586	0.36475803	-0.57945832	0.
10	1.87249362	0.46920586	0.36475803	-0.57945832	0.
11	2.18514082	0.36461255	0.19847109	0.26427683	0.
12	2.18514082	0.36461255	0.19847109	0.26427683	0.
13	2.18514082	0.36461255	0.19847109	0.26427683	0.
14	2.18514082	0.36461255	0.19847109	0.26427683	0.

- Each of these is uncorrelated with each other
  - ➢ The variance of each is equal to the corresponding eigenvalue

- We would then use these in subsequent analyses

PCA Summary

- PCA is a **data reduction** technique that relies on the mathematical properties of eigenvalues and eigenvectors
  - **Used to create new variables (small number) out of the old data (lots of variables)**
  - The new variables are principal components (they are not factor scores)

- PCA appeared first in the psychometric literature
  - Many “factor analysis” methods used variants of PCA before likelihood-based statistics were available

- Currently, PCA (or variants) methods are the default option in SPSS and SAS (PROC FACTOR)

Potentially Solvable Statistical Issues in PCA

- The typical PCA analysis also has a few statistical concerns
  - Some of these can be solved if you know what you are doing
  - The typical analysis (using program defaults) does not solve these

- Missing data is omitted using listwise deletion – biases possible
  - Could use ML to estimate covariance matrix, but then would have to assume multivariate normality
  - Could use MI to impute data

- The distributions of variables can be anything…but variables with much larger variances will look like they contribute more to each component
  - Could standardize variables – but some can't be standardized easily (think gender)

- The lack of standard errors makes the component weights (eigenvector elements)
  hard to interpret
  - Can use a resampling/bootstrap analysis to get SEs (but not easy to do)

My (Unsolvable) Issues with PCA

- My issues with PCA involve the two questions in need of answers for any use of PCA:

1. The number of components needed is not based on a statistical hypothesis test and hence is subjective
   - Variance accounted for is a descriptive measure
   - No statistical test for whether an additional component significantly accounts for more variance

2. The relative meaning of each component is questionable at best and hence is subjective
   - Typical packages provide no standard errors for each eigenvector weight (can be obtained in bootstrap analyses)
   - No definitive answer for component composition

- In sum, I feel it is very easy to be misled (or purposefully mislead) with PCA

EXPLORATORY FACTOR ANALYSIS

Primary Purpose of EFA

- **EFA**: "Determine nature and number of latent variables that account for observed variation and covariation among set of observed indicators (≈ items or variables)"
  - In other words, what causes these observed responses?
  - Summarize patterns of correlation among indicators
  - Solution is an end (i.e., is of interest) in and of itself

- Compared with **PCA**: "Reduce multiple observed variables into fewer components that summarize their variance"
  - In other words, how can I abbreviate this set of variables?
  - Solution is usually a means to an end

Methods for EFA

- You will see many different types of methods for “extraction” of factors in EFA
  - Many are PCA-based
  - Most were developed before computers became relevant or likelihood theory was developed

- You can ignore all of them and focus on one:

Only Use Maximum Likelihood for EFA

- The maximum likelihood method of EFA extraction:
  - Uses the same log-likelihood as confirmatory factor analyses/SEM
    - Default assumption: multivariate normal distribution of data
  - Provides consistent estimates with good statistical properties (assuming you have a large enough sample)
  - Missing data using all the data that was observed (MAR)
  - Is consistent with modern statistical practices

Questions when using EFA

- EFAs proceed by seeking the answers to two questions:

(the same questions posed in PCA; but with different terms)

1. How many latent factors are needed to “adequately” represent the original data?
   - “Adequately” = does a given EFA model fit well?

2. (once #1 has been answered): What does each factor represent?
   - The term “represent” is fuzzy

The Syntax of Factor Analysis

- Factor analysis works by hypothesizing that a set of latent factors helps to determine a person's response to a set of variables
  - This can be explained by a system of simultaneous linear models
  - Here Y = observed data, p = person, v = variable, F = factor score (Q factors)

$$
\begin{gathered}
Y_{p1} = \mu_{y_1} + \lambda_{11}F_{p1} + \lambda_{12}F_{p2} + \cdots + \lambda_{1Q}F_{pQ} + e_{p1} \\
Y_{p2} = \mu_{y_2} + \lambda_{21}F_{p1} + \lambda_{22}F_{p2} + \cdots + \lambda_{2Q}F_{pQ} + e_{p2} \\
\vdots \\
Y_{pV} = \mu_{y_V} + \lambda_{V1}F_{p1} + \lambda_{V2}F_{p2} + \cdots + \lambda_{VQ}F_{pQ} + e_{pV}
\end{gathered}
$$

- μ_{y_v} = mean for variable v
- λ_{vf} = factor loading for variable v onto factor f (regression slope)
  - Factors are assumed distributed MVN with zero mean and (for EFA) identity covariance matrix (uncorrelated factors – to start)
- e_{pv} = residual for person p and variable v
  - Residuals are assumed distributed MVN (across items) with a zero mean and a diagonal covariance matrix $\mathbf{\Psi}$ containing the unique variances
- Often, this gets shortened into matrix form:

$$\mathbf{Y}_p = \boldsymbol{\mu}_Y + \mathbf{\Lambda}\mathbf{F}_p^T + \mathbf{e_p}$$

How Maximum Likelihood EFA Works

- Maximum likelihood EFA assumes the data follow a multivariate normal distribution
  - The basis for the log-likelihood function (same log-likelihood we have used in every analysis to this point)

- The log-likelihood function depends on two sets of parameters: the mean vector and the covariance matrix
  - Mean vector is saturated (just uses the item means for item intercepts) – so it is often not thought of in analysis
    - Denoted as $\boldsymbol{\mu}_Y = \boldsymbol{\mu}_I$
  - Covariance matrix is what gives "factor structure"
    - EFA models provide a structure for the covariance matrix

The EFA Model for the Covariance Matrix

- The covariance matrix is modeled based on how it would look if a set of hypothetical (latent) factors had caused the data

- For an analysis measuring F factors, each item in the EFA:
  - Has 1 unique variance parameter
  - Has F factor loadings

- The initial estimation of factor loadings is conducted based on the assumption of uncorrelated factors
  - Assumption is dubious at best – yet is the cornerstone of the analysis

Model Implied Covariance Matrix

- The factor model implied covariance matrix is $\mathbf{\Sigma}_Y = \mathbf{\Lambda}\mathbf{\Phi}\mathbf{\Lambda}^T + \mathbf{\Psi}$
  - Where:
    - $\mathbf{\Sigma}_Y$ = model implied covariance matrix of the observed data (size I x I)
    - $\mathbf{\Lambda}$ = matrix of factor loadings (size I x F)
      - In EFA: all terms in $\mathbf{\Lambda}$ are estimated
    - $\mathbf{\Phi}$ = factor covariance matrix (size F x F)
      - In EFA: $\mathbf{\Phi} = \mathbf{I}$ (all factors have variances of 1 and covariances of 0)
      - In CFA: this is estimated
    - $\mathbf{\Psi}$ = matrix of unique (residual) variances (size I x I)
      - In EFA: $\mathbf{\Psi}$ is diagonal by default (no residual covariances)

- Therefore, the EFA model-implied covariance matrix is:

$$\mathbf{\Sigma}_Y = \mathbf{\Lambda}\mathbf{\Lambda}^T + \mathbf{\Psi}$$

EFA Model Identifiability

- Under the ML method for EFA, the same rules of identification apply to EFA as to Path Analysis
  - T-rule: Total number of EFA model parameters must not exceed unique elements in saturated covariance matrix of data
    - For an analysis with a number of factors F and a set number of items I there are $F^{*}I + I = I(F+1)$ EFA model parameters
    - As we will see, there must be $\frac{F(F-1)}{2}$ constraints for the model to work
    - Therefore, $I(F+1) - \frac{F(F-1)}{2} < \frac{I(I+1)}{2}$
  - Local-identification: each portion of the model must be locally identified
    - With all factor loadings estimated local identification fails
      - No way of differentiating factors without constraints

Constraints to Make EFA in ML Identified

- The EFA model imposes the following constraint:

$$\boldsymbol{\Lambda}^{T}\boldsymbol{\Psi}\boldsymbol{\Lambda} = \boldsymbol{\Delta}$$

such that Δ is a diagonal matrix

- This puts $\frac{F(F-1)}{2}$ constraints on the model (that many fewer parameters to estimate)

- This constraint is not well known – and how it functions is hard to describe
  - For a 1-factor model, the results of EFA and CFA will match

- Note: the other methods of EFA "extraction" avoid this constraint by not being statistical models in the first place
  - PCA-based routines rely on matrix properties to resolve identification

The Nature of the Constraints in EFA

- The EFA constraints provide some detailed assumptions about the nature of the factor model and how it pertains to the data

- For example, take a 2-factor model (one constraint):

$$\sum_{v=1}^{V} \psi_v^2 \prod_{f=1}^{Q=2} \lambda_{vf} = 0$$

- In short, some combinations of factor loadings and unique variances (across and within items) cannot happen
  - This goes against most of our statistical constraints – which must be justifiable and understandable (therefore testable)
  - This constraint is not testable in CFA

The Log-Likelihood Function

- Given the model parameters, the EFA model is estimated by maximizing the multivariate normal log-likelihood
  - For the data

$$\log L = \log\left[= (2\pi)^{-\frac{NV}{2}} \left|\boldsymbol{\Sigma}\right|^{-\frac{N}{2}} \exp\left[\sum_{p=1}^{N} -\frac{\left(\boldsymbol{Y}_p - \boldsymbol{\mu}_y\right)^T \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{Y}_p - \boldsymbol{\mu}_y\right)}{2}\right]\right] =$$

$$-\frac{NV}{2}\log(2\pi) - \frac{N}{2}\log(|\boldsymbol{\Sigma}|) - \sum_{p=1}^{N} \frac{\left(\boldsymbol{Y}_p - \boldsymbol{\mu}_y\right)^T \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{Y}_p - \boldsymbol{\mu}_y\right)}{2}$$

- Under EFA, this becomes:

$$\log L = -\frac{NV}{2}\log(2\pi) - \frac{N}{2}\log\left(\left|\boldsymbol{\Lambda}\boldsymbol{\Lambda}^T + \boldsymbol{\Psi}\right|\right) - \sum_{p=1}^{N} \frac{\left(\boldsymbol{Y}_p - \boldsymbol{\mu}_{\boldsymbol{I}}\right)^T \left(\boldsymbol{\Lambda}\boldsymbol{\Lambda}^T + \boldsymbol{\Psi}\right)^{-1}\left(\boldsymbol{Y}_{\boldsymbol{p}} - \boldsymbol{\mu}_{\boldsymbol{I}}\right)}{2}$$

Benefits and Consequences of EFA with ML

- The parameters of the EFA model under ML retain the same benefits and consequences of any model (i.e., CFA)
  - Asymptotically (large N) they are consistent, normal, and efficient
  - Missing data are "skipped" in the likelihood, allowing for incomplete observations to contribute (assumed MAR)

- Furthermore, the same types of model fit indices are available in EFA as are in CFA

- As with CFA, though, an EFA model must be a close approximation to the saturated model covariance matrix if the parameters are to be believed
  - This is a marked difference between EFA in ML and EFA with other methods – quality of fit is statistically rigorous

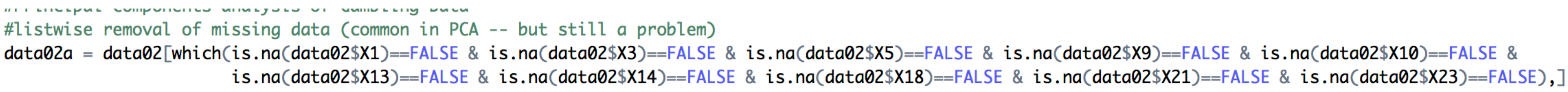
ML EFA WITH BASE R FUNCTION FACTANAL (THE BAD WAY)

ML EFA Using the factanal() Function

- The base R program has the factanal() function that conducts ML-based EFA
    - But it is very limited

- Although the function use ML, you still cannot have missing data in the analysis
    - BAD R!

- We will remove cases with any missing data (listwise deletion) and proceed



```
#listwise removal of missing data (common in PCA -- but still a problem)
data02a = data02[which(is.na(data02$X1)==FALSE & is.na(data02$X3)==FALSE & is.na(data02$X5)==FALSE & is.na(data02$X9)==FALSE & is.na(data02$X10)==FALSE &
                       is.na(data02$X13)==FALSE & is.na(data02$X14)==FALSE & is.na(data02$X18)==FALSE & is.na(data02$X21)==FALSE & is.na(data02$X23)==FALSE),]
```

- We will also not use a rotation method at first as to show how default constraints in EFA with ML are ridiculous

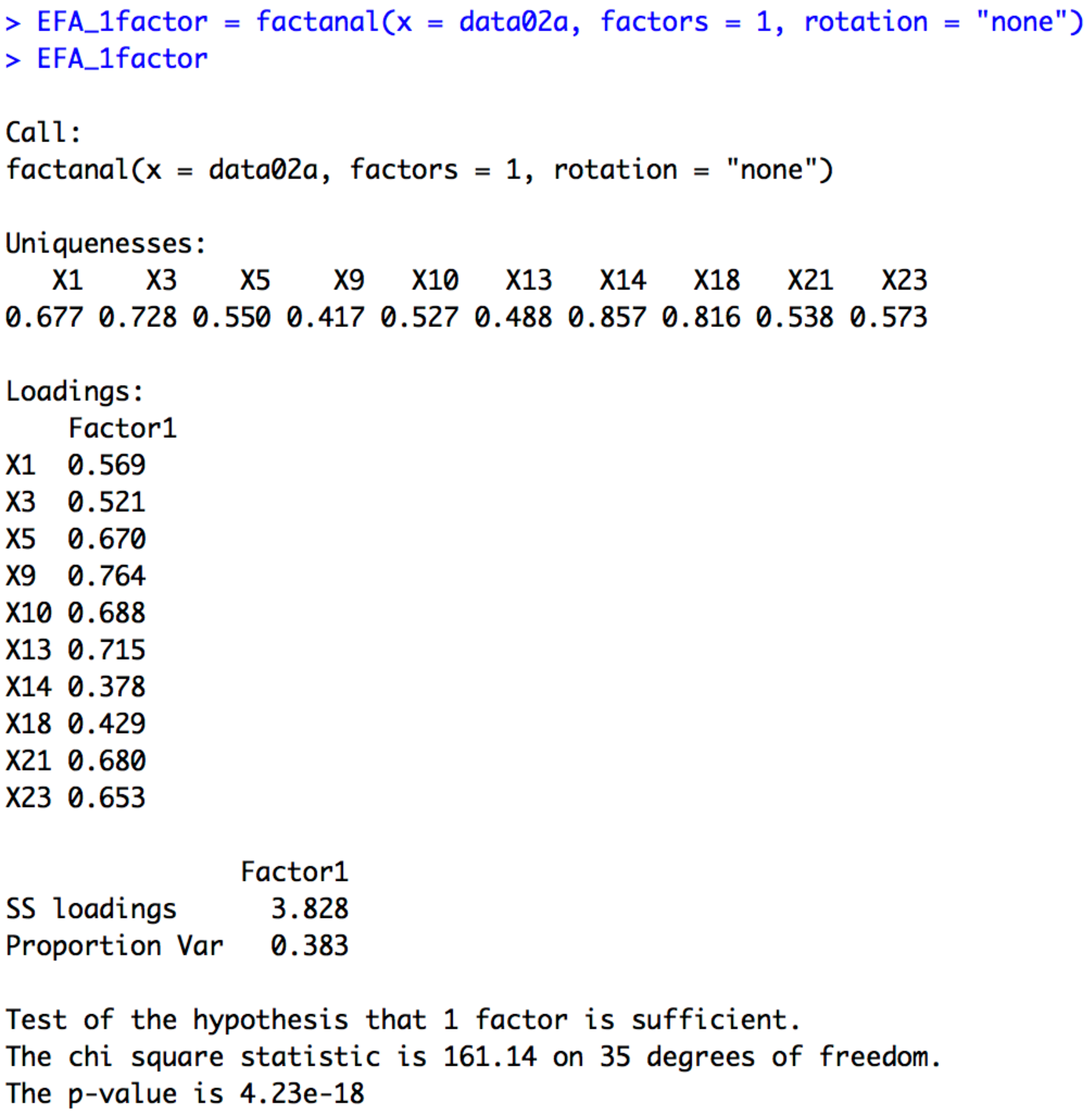
Step #1: Determine Number of Factors

- The EFA factanal() function provides a rudimentary test for model fit

- Remember the saturated model from path analysis?
  - All covariances estimated

- The model fit tests the solution from EFA vs the saturated model
  - EFA 1-factor model shown

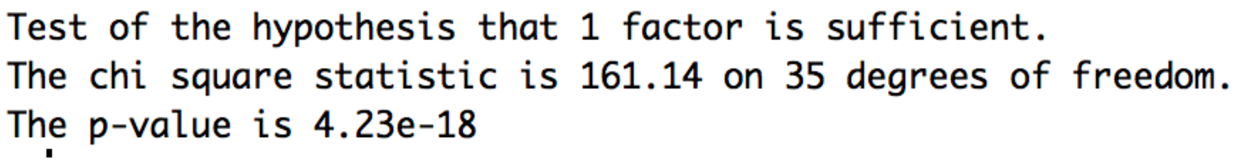
- The goal is to find a model that fits well



```
> EFA_1factor = factanal(x = data02a, factors = 1, rotation = "none")
> EFA_1factor

Call:
factanal(x = data02a, factors = 1, rotation = "none")

Uniquenesses:
   X1    X3    X5    X9   X10   X13   X14   X18   X21   X23 
0.677 0.728 0.550 0.417 0.527 0.488 0.857 0.816 0.538 0.573 

Loadings:
    Factor1
X1  0.569  
X3  0.521  
X5  0.670  
X9  0.764  
X10 0.688  
X13 0.715  
X14 0.378  
X18 0.429  
X21 0.680  
X23 0.653  

               Factor1
SS loadings      3.828
Proportion Var   0.383

Test of the hypothesis that 1 factor is sufficient.
The chi square statistic is 161.14 on 35 degrees of freedom.
The p-value is 4.23e-18 
```

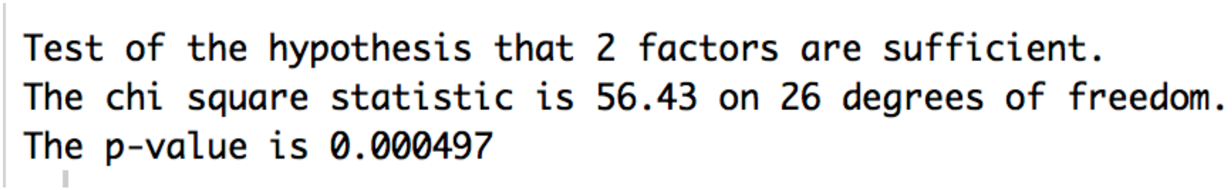
Step #1 in R: Model Fit Tests

- One factor:



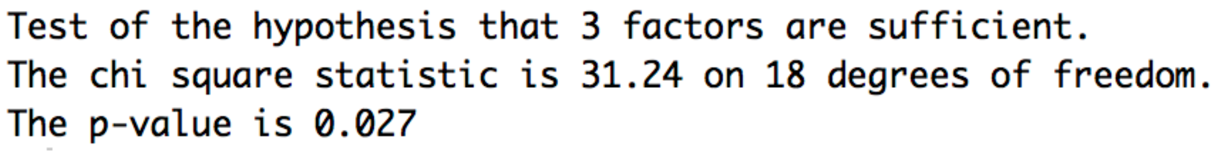
```
Test of the hypothesis that 1 factor is sufficient.
The chi square statistic is 161.14 on 35 degrees of freedom.
The p-value is 4.23e-18
```

- Two factors:



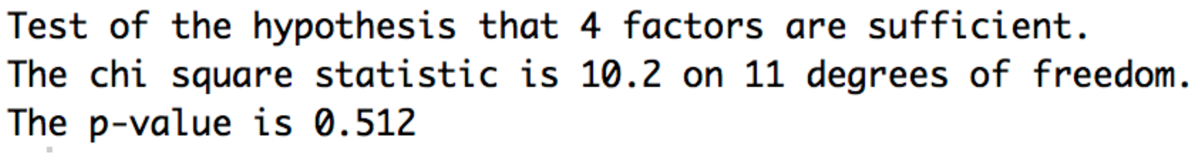
```
Test of the hypothesis that 2 factors are sufficient.
The chi square statistic is 56.43 on 26 degrees of freedom.
The p-value is 0.000497
```

- Three factors:



```
Test of the hypothesis that 3 factors are sufficient.
The chi square statistic is 31.24 on 18 degrees of freedom.
The p-value is 0.027
```

- Four factors:



```
Test of the hypothesis that 4 factors are sufficient.
The chi square statistic is 10.2 on 11 degrees of freedom.
The p-value is 0.512
```

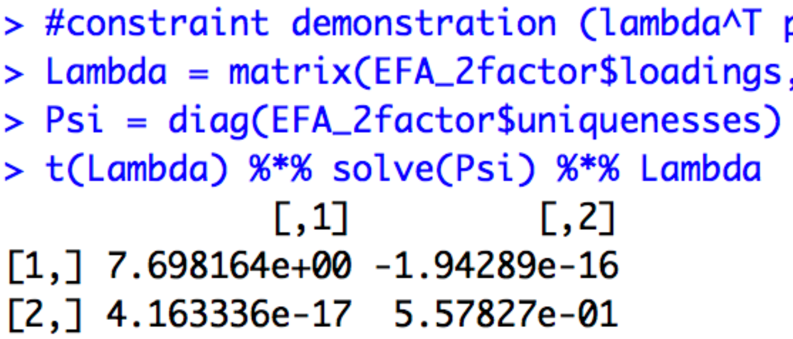
Side Note: Default Constraints in ML EFA

- The EFA model imposes the following constraint:

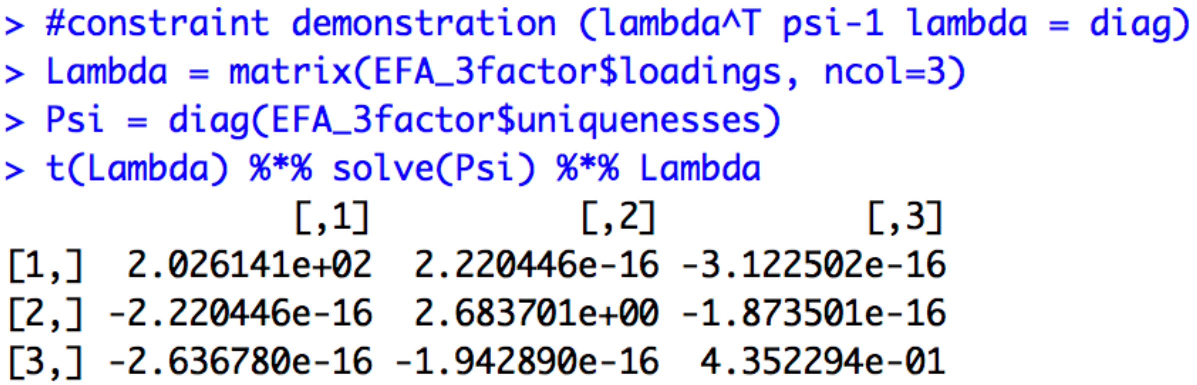
$$\boldsymbol{\Lambda}^T\boldsymbol{\Psi}\boldsymbol{\Lambda} = \boldsymbol{\Delta}$$

such that $\boldsymbol{\Delta}$ is a diagonal matrix

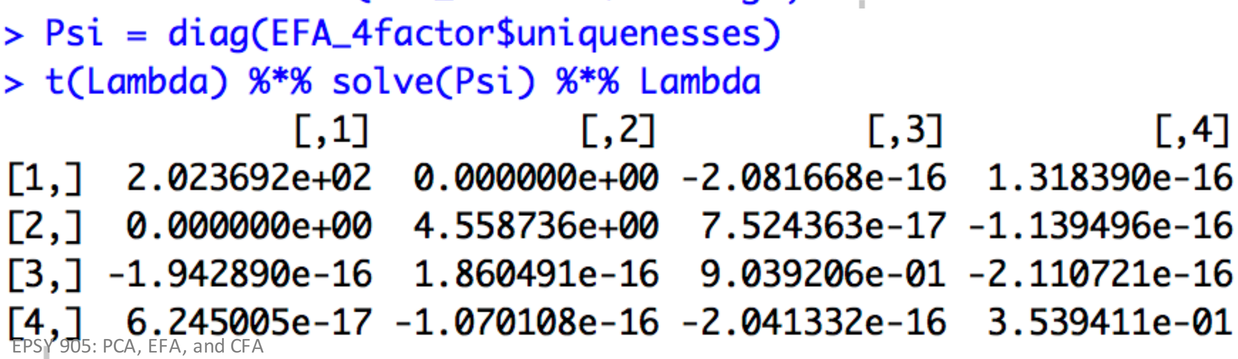
- Here are the $\boldsymbol{\Delta}$ matrices from each analysis:



```
> #constraint demonstration (lambda^T psi-1 lambda = diag)
> Lambda = matrix(EFA_2factor$loadings, ncol=2)
> Psi = diag(EFA_2factor$uniquenesses)
> t(Lambda) %*% solve(Psi) %*% Lambda
             [,1]         [,2]
[1,] 7.698164e+00 -1.94289e-16
[2,] 4.163336e-17  5.57827e-01
```



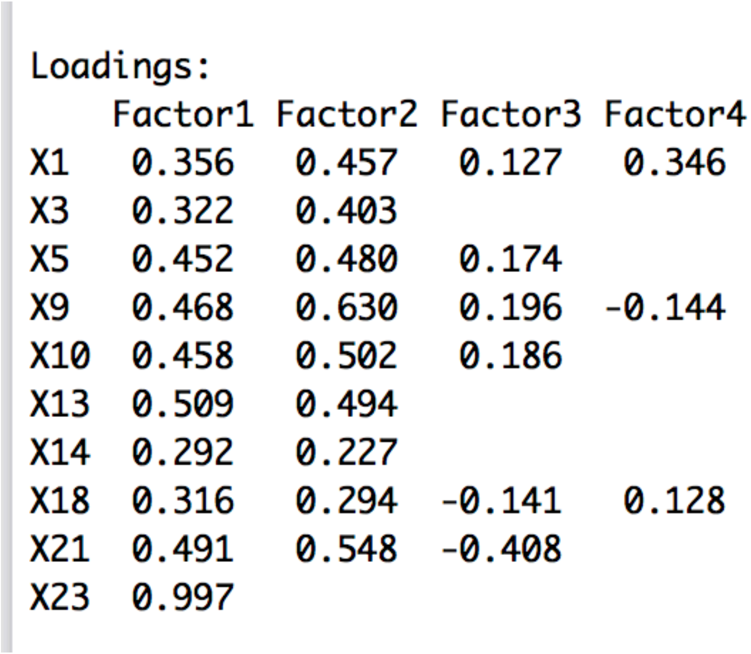
```
> #constraint demonstration (lambda^T psi-1 lambda = diag)
> Lambda = matrix(EFA_3factor$loadings, ncol=3)
> Psi = diag(EFA_3factor$uniquenesses)
> t(Lambda) %*% solve(Psi) %*% Lambda
              [,1]          [,2]          [,3]
[1,]  2.026141e+02  2.220446e-16 -3.122502e-16
[2,] -2.220446e-16  2.683701e+00 -1.873501e-16
[3,] -2.636780e-16 -1.942890e-16  4.352294e-01
```



```
> Lambda = matrix(EFA_4factor$loadings,
> Psi = diag(EFA_4factor$uniquenesses)
> t(Lambda) %*% solve(Psi) %*% Lambda
              [,1]          [,2]          [,3]          [,4]
[1,]  2.023692e+02  0.000000e+00 -2.081668e-16  1.318390e-16
[2,]  0.000000e+00  4.558736e+00  7.524363e-17 -1.139496e-16
[3,] -1.942890e-16  1.860491e-16  9.039206e-01 -2.110721e-16
[4,]  6.245005e-17 -1.070108e-16 -2.041332e-16  3.539411e-01
```

Step #2: Interpreting the Best Model

- As the four-factor solution fit best, we will interpret it
- Unrotated solution of factor loadings:



```
Loadings:
    Factor1 Factor2 Factor3 Factor4
X1   0.356   0.457   0.127   0.346
X3   0.322   0.403
X5   0.452   0.480   0.174
X9   0.468   0.630   0.196  -0.144
X10  0.458   0.502   0.186
X13  0.509   0.494
X14  0.292   0.227
X18  0.316   0.294  -0.141   0.128
X21  0.491   0.548  -0.408
X23  0.997
```

What???

FACTOR LOADING ROTATIONS IN EFA

Rotations of Factor Loadings in EFA

- Transformations of the factor loadings are possible as the matrix of factor loadings is only unique up to an orthogonal transformation
  - Don’t like the solution? Rotate!

- Historically, rotations use the properties of matrix algebra to adjust the factor loadings to more interpretable numbers

- Modern versions of rotations/transformations rely on “target functions” that specify what a “good” solution should look like
  - The details of the modern approach are lacking in most texts

Types of Classical Rotated Solutions

- Multiple types of rotations exist but two broad categories seem to dominate how they are discussed:

- **Orthogonal rotations:** rotations that force the factor correlation to zero (orthogonal factors). The name orthogonal relates to the angle between axes of factor solutions being 90 degrees. The most prevalent is the varimax rotation.

- **Oblique rotations:** rotations that allow for non-zero factor correlations. The name orthogonal relates to the angle between axes of factor solutions not being 90 degrees. The most prevalent is the promax rotation.
  - These rotations provide an estimate of "factor correlation"

How Classical Orthogonal Rotation Works

- Classical orthogonal rotation algorithms work by defining a new rotated set of factor loadings $\mathbf{\Lambda}^*$ as a function of the original (non-rotated) loadings $\mathbf{\Lambda}$ and an orthogonal rotation matrix $\mathbf{T}$

$$\mathbf{\Lambda}^* = \mathbf{\Lambda T}$$

where: $\mathbf{TT}^T = \mathbf{T}^T\mathbf{T} = \mathbf{I}$

- These rotations do not alter the fit of the model as

$$\mathbf{\Sigma}_Y = \mathbf{\Lambda}^*\mathbf{\Lambda}^{*T} + \mathbf{\Psi} = \mathbf{\Lambda T}(\mathbf{\Lambda T}\,)^T + \mathbf{\Psi} = \mathbf{\Lambda TT}^T\mathbf{\Lambda}^T + \mathbf{\Psi}$$
$$= \mathbf{\Lambda\Lambda}^T + \mathbf{\Psi}$$

Modern Versions of Rotation

- Most studies using EFA use the classical rotation mechanisms, likely due to insufficient training

- Modern methods for rotations rely on the use of a target function for how an optimal loading solution should look

$$\mathbf{L} = \begin{bmatrix} x & 0 & 0 \\ x & 0 & 0 \\ x & 0 & 0 \\ x & 0 & 0 \\ 0 & x & 0 \\ 0 & x & 0 \\ 0 & 0 & x \\ 0 & 0 & x \\ 0 & 0 & x \end{bmatrix}$$

From Browne (2001)

Rotation Algorithms

- Given a target function, rotation algorithms seek to find a rotated solution that simultaneously:

    1. Minimizes the distance between the rotated solution and the original factor loadings

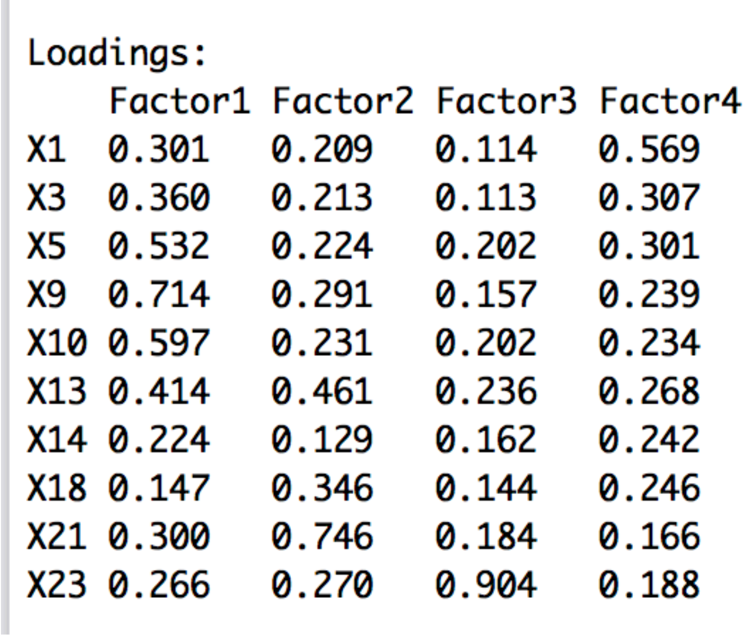
    2. Fits best to the target function

- Rotation algorithms are typically iterative – meaning they can fail to converge

- Rotation searches typically have multiple optimal values
    - Need many restarts

Rotated Factor Loadings: Orthogonal Rotation via Varimax

- The Varimax rotation brought about the following loadings



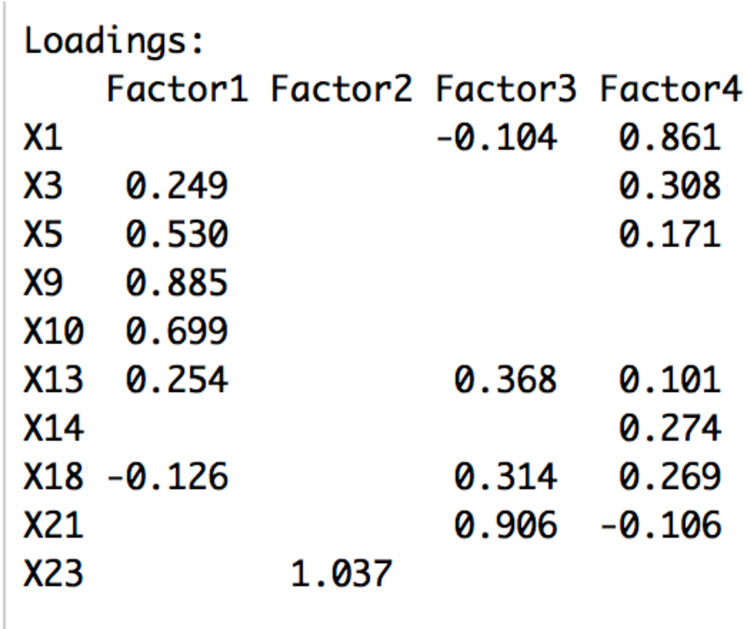
```
Loadings:
    Factor1 Factor2 Factor3 Factor4
X1  0.301   0.209   0.114   0.569
X3  0.360   0.213   0.113   0.307
X5  0.532   0.224   0.202   0.301
X9  0.714   0.291   0.157   0.239
X10 0.597   0.231   0.202   0.234
X13 0.414   0.461   0.236   0.268
X14 0.224   0.129   0.162   0.242
X18 0.147   0.346   0.144   0.246
X21 0.300   0.746   0.184   0.166
X23 0.266   0.270   0.904   0.188
```

- Are these better for interpretation?

- Also note: no factor correlation

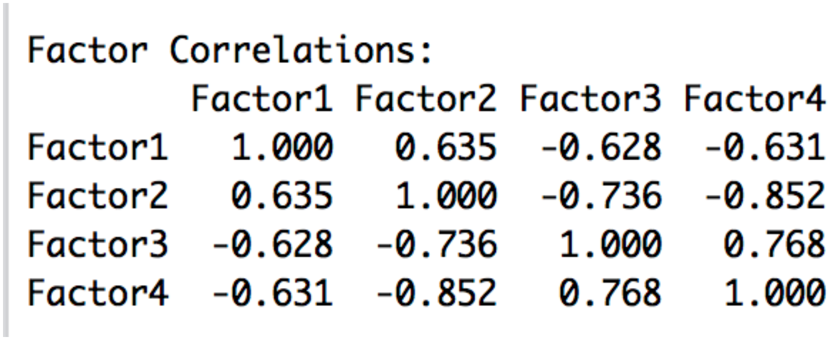
Rotated Factor Loadings: Oblique Rotation via Promax

- The Promax rotation brought about the following loadings:



```
Loadings:
    Factor1 Factor2 Factor3 Factor4
X1                  -0.104   0.861
X3   0.249                   0.308
X5   0.530                   0.171
X9   0.885
X10  0.699
X13  0.254           0.368   0.101
X14                          0.274
X18 -0.126           0.314   0.269
X21                  0.906  -0.106
X23          1.037
```

- It also brought about the following factor correlations:



```
Factor Correlations:
        Factor1 Factor2 Factor3 Factor4
Factor1   1.000   0.635  -0.628  -0.631
Factor2   0.635   1.000  -0.736  -0.852
Factor3  -0.628  -0.736   1.000   0.768
Factor4  -0.631  -0.852   0.768   1.000
```

EFA VIA CFA

CFA Approaches to EFA

- We can conduct exploratory analysis using a CFA model
  - Need to set the right number of constraints for identification
  - We set the value of factor loadings for a few items on a few of the factors
    - Typically to zero (my usual thought)
    - Sometimes to one (Brown, 2002)
  - We keep the factor covariance matrix as an identity
    - Uncorrelated factors (as in EFA) with variances of one

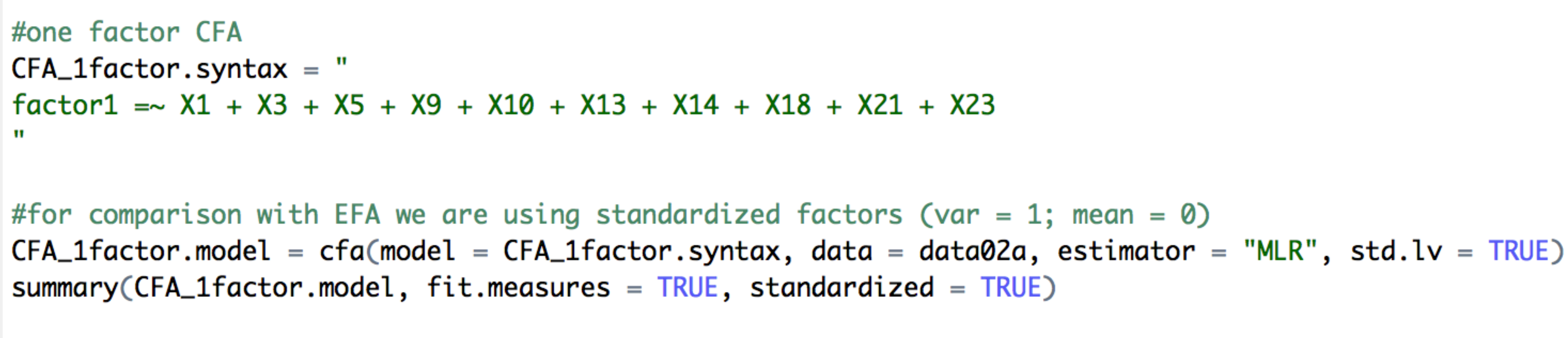
- Benefits of using CFA for exploratory analyses:
  - CFA constraints remove rotational indeterminacy of factor loadings – no rotating is needed (or possible)
  - Defines factors with *potentially* less ambiguity
    - Constraints are easy to see
  - For some software (SAS and SPSS), we get much more model fit information

EFA with CFA Constraints

- To do EFA with CFA, you must:
  - Fix factor loadings (set to either zero or one)
    - Use “row echelon” form :
    - One item has only one factor loading estimated
    - One item has only two factor loadings estimated
    - One item has only three factor loadings estimated
  - Fix factor covariances
    - Set all to 0
  - Fix factor variances
    - Set all to 1

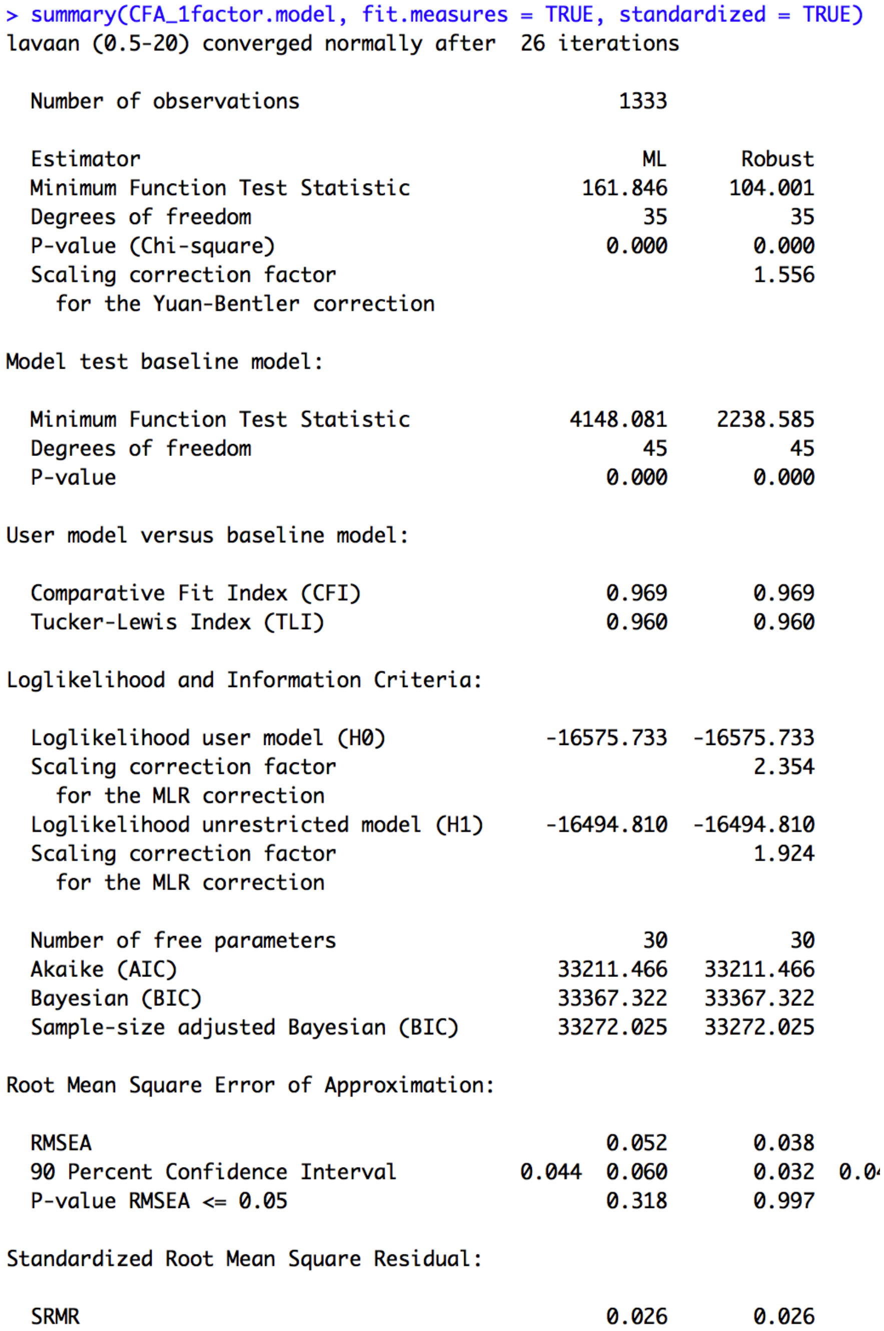
EFA Via CFA Example

- We can use lavaan to do CFA…here is the syntax for the one factor model
  - The ~= is the key → Factor name to the left, items measuring it to the right

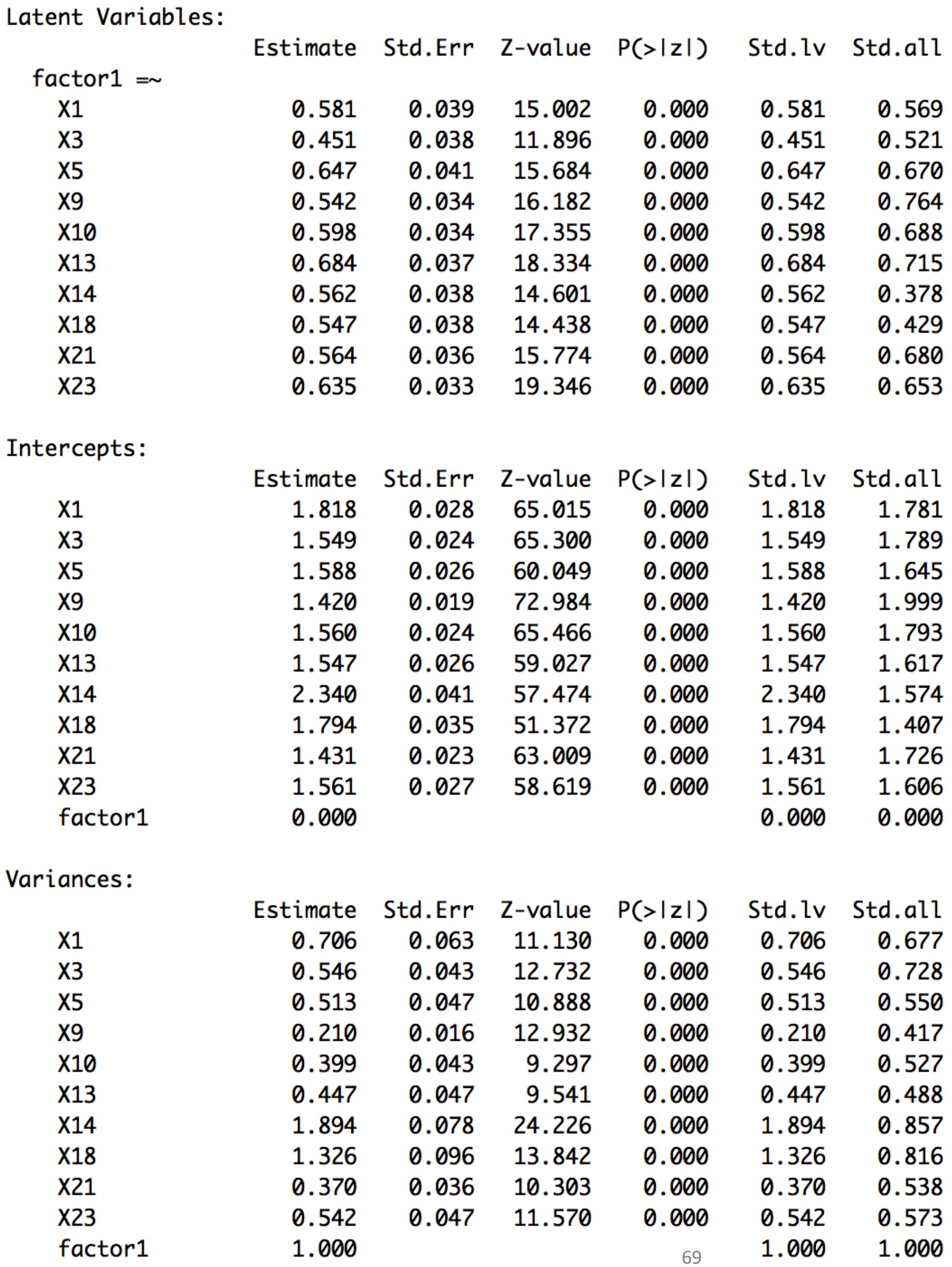


```
#one factor CFA
CFA_1factor.syntax = "
factor1 =~ X1 + X3 + X5 + X9 + X10 + X13 + X14 + X18 + X21 + X23
"

#for comparison with EFA we are using standardized factors (var = 1; mean = 0)
CFA_1factor.model = cfa(model = CFA_1factor.syntax, data = data02a, estimator = "MLR", std.lv = TRUE)
summary(CFA_1factor.model, fit.measures = TRUE, standardized = TRUE)
```

One-Factor Results from lavaan

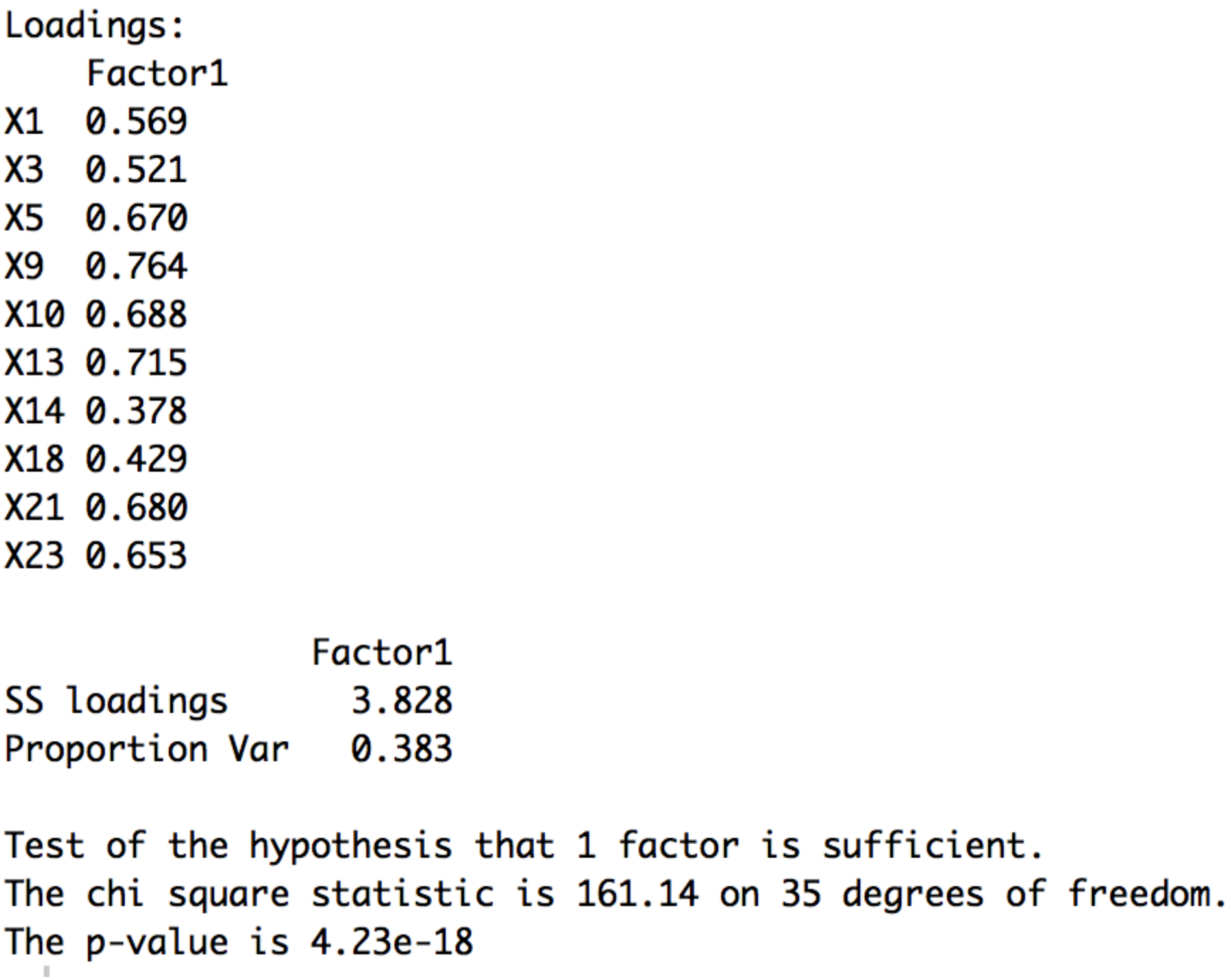


```
> summary(CFA_1factor.model, fit.measures = TRUE, standardized = TRUE)
lavaan (0.5-20) converged normally after  26 iterations

  Number of observations                          1333

  Estimator                                         ML      Robust
  Minimum Function Test Statistic              161.846     104.001
  Degrees of freedom                                35          35
  P-value (Chi-square)                           0.000       0.000
  Scaling correction factor                                  1.556
    for the Yuan-Bentler correction

Model test baseline model:

  Minimum Function Test Statistic             4148.081    2238.585
  Degrees of freedom                                45          45
  P-value                                        0.000       0.000

User model versus baseline model:

  Comparative Fit Index (CFI)                    0.969       0.969
  Tucker-Lewis Index (TLI)                       0.960       0.960

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)             -16575.733  -16575.733
  Scaling correction factor                                  2.354
    for the MLR correction
  Loglikelihood unrestricted model (H1)     -16494.810  -16494.810
  Scaling correction factor                                  1.924
    for the MLR correction

  Number of free parameters                         30          30
  Akaike (AIC)                               33211.466   33211.466
  Bayesian (BIC)                             33367.322   33367.322
  Sample-size adjusted Bayesian (BIC)        33272.025   33272.025

Root Mean Square Error of Approximation:

  RMSEA                                          0.052       0.038
  90 Percent Confidence Interval          0.044  0.060       0.032  0.0
  P-value RMSEA <= 0.05                          0.318       0.997

Standardized Root Mean Square Residual:

  SRMR                                           0.026       0.026
```



```
Latent Variables:
                   Estimate  Std.Err  Z-value  P(>|z|)   Std.lv  Std.all
  factor1 =~
    X1                0.581    0.039   15.002    0.000    0.581    0.569
    X3                0.451    0.038   11.896    0.000    0.451    0.521
    X5                0.647    0.041   15.684    0.000    0.647    0.670
    X9                0.542    0.034   16.182    0.000    0.542    0.764
    X10               0.598    0.034   17.355    0.000    0.598    0.688
    X13               0.684    0.037   18.334    0.000    0.684    0.715
    X14               0.562    0.038   14.601    0.000    0.562    0.378
    X18               0.547    0.038   14.438    0.000    0.547    0.429
    X21               0.564    0.036   15.774    0.000    0.564    0.680
    X23               0.635    0.033   19.346    0.000    0.635    0.653

Intercepts:
                   Estimate  Std.Err  Z-value  P(>|z|)   Std.lv  Std.all
    X1                1.818    0.028   65.015    0.000    1.818    1.781
    X3                1.549    0.024   65.300    0.000    1.549    1.789
    X5                1.588    0.026   60.049    0.000    1.588    1.645
    X9                1.420    0.019   72.984    0.000    1.420    1.999
    X10               1.560    0.024   65.466    0.000    1.560    1.793
    X13               1.547    0.026   59.027    0.000    1.547    1.617
    X14               2.340    0.041   57.474    0.000    2.340    1.574
    X18               1.794    0.035   51.372    0.000    1.794    1.407
    X21               1.431    0.023   63.009    0.000    1.431    1.726
    X23               1.561    0.027   58.619    0.000    1.561    1.606
    factor1           0.000                               0.000    0.000

Variances:
                   Estimate  Std.Err  Z-value  P(>|z|)   Std.lv  Std.all
    X1                0.706    0.063   11.130    0.000    0.706    0.677
    X3                0.546    0.043   12.732    0.000    0.546    0.728
    X5                0.513    0.047   10.888    0.000    0.513    0.550
    X9                0.210    0.016   12.932    0.000    0.210    0.417
    X10               0.399    0.043    9.297    0.000    0.399    0.527
    X13               0.447    0.047    9.541    0.000    0.447    0.488
    X14               1.894    0.078   24.226    0.000    1.894    0.857
    X18               1.326    0.096   13.842    0.000    1.326    0.816
    X21               0.370    0.036   10.303    0.000    0.370    0.538
    X23               0.542    0.047   11.570    0.000    0.542    0.573
    factor1           1.000                               1.000    1.000
```

Look Familiar? They are Identical to the One-Factor EFA from factanal()



```
Loadings:
    Factor1
X1  0.569
X3  0.521
X5  0.670
X9  0.764
X10 0.688
X13 0.715
X14 0.378
X18 0.429
X21 0.680
X23 0.653

               Factor1
SS loadings      3.828
Proportion Var   0.383

Test of the hypothesis that 1 factor is sufficient.
The chi square statistic is 161.14 on 35 degrees of freedom.
The p-value is 4.23e-18
```

CFA Logic…Applied to EFA

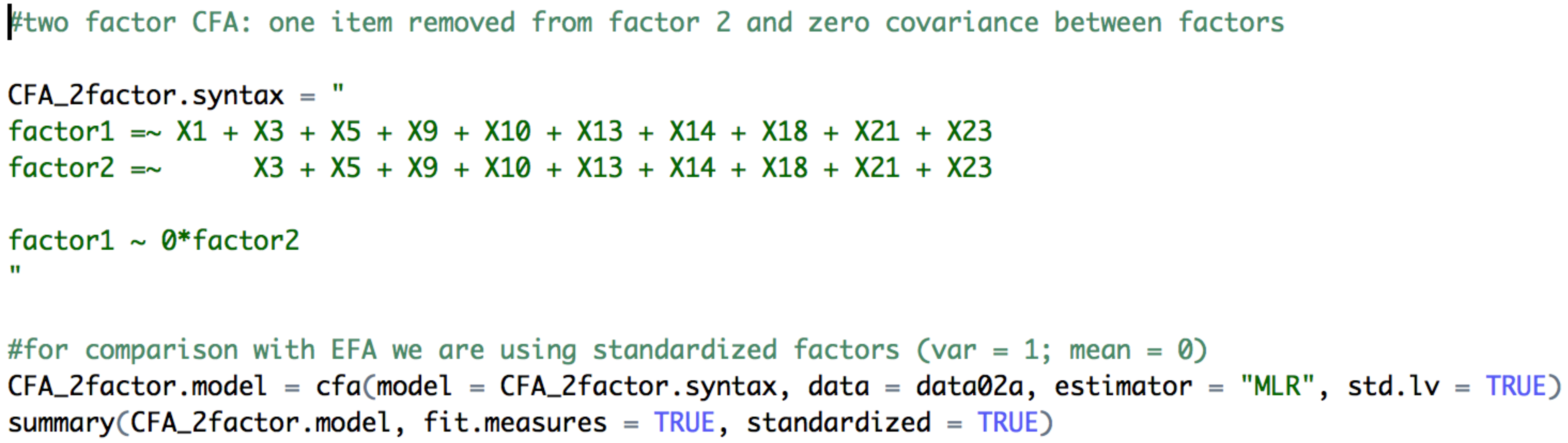
- Because our one-factor model fit well, we can stop!

- CFA has more indices of model fit – which can make finding an appropriate solution easier

- CFA also gives you the standard errors for each factor loading, leading to a Wald test to see if it is non-zero
  - No need to use arbitrary .3 cutoff
  - Small note: Although most EFA routines (like factanal) don't give SEs they are certainly attainable under ML theory

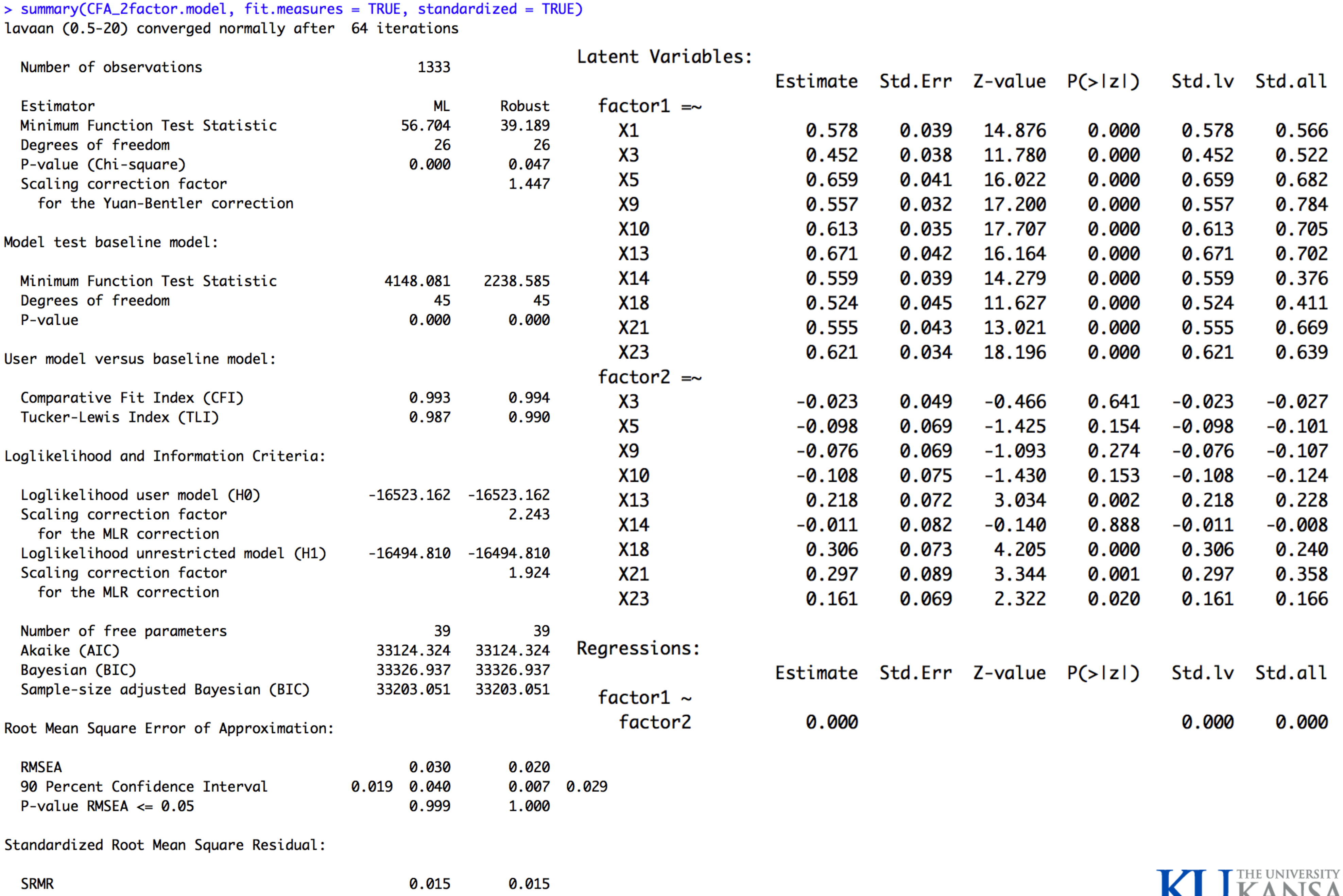
- Although we should stop here…We'll continue with the two- and three-factor versions to compare with EFA

Two-Factor Syntax



```
#two factor CFA: one item removed from factor 2 and zero covariance between factors

CFA_2factor.syntax = "
factor1 =~ X1 + X3 + X5 + X9 + X10 + X13 + X14 + X18 + X21 + X23
factor2 =~      X3 + X5 + X9 + X10 + X13 + X14 + X18 + X21 + X23

factor1 ~ 0*factor2
"

#for comparison with EFA we are using standardized factors (var = 1; mean = 0)
CFA_2factor.model = cfa(model = CFA_2factor.syntax, data = data02a, estimator = "MLR", std.lv = TRUE)
summary(CFA_2factor.model, fit.measures = TRUE, standardized = TRUE)
```

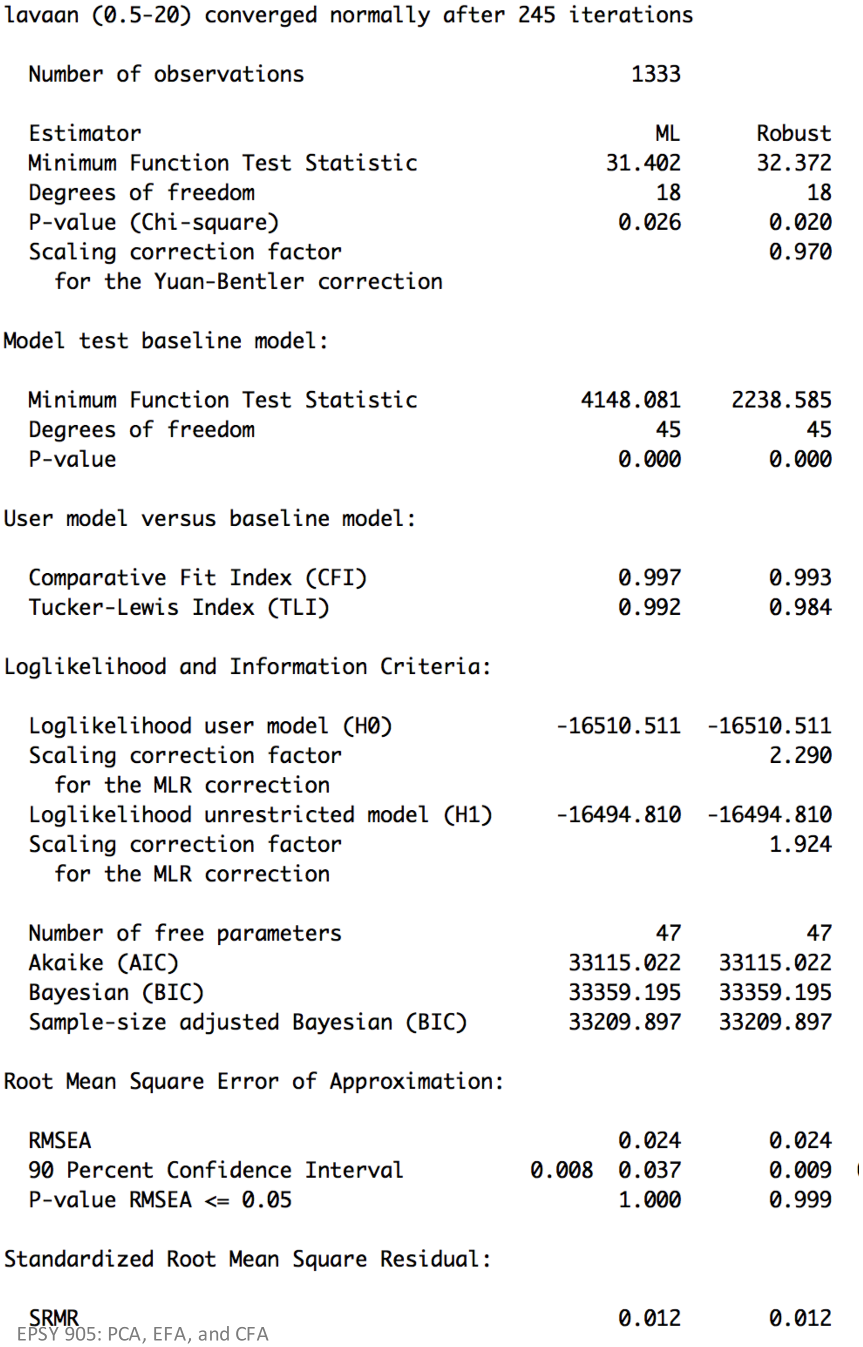
CFA Two-Factor Results



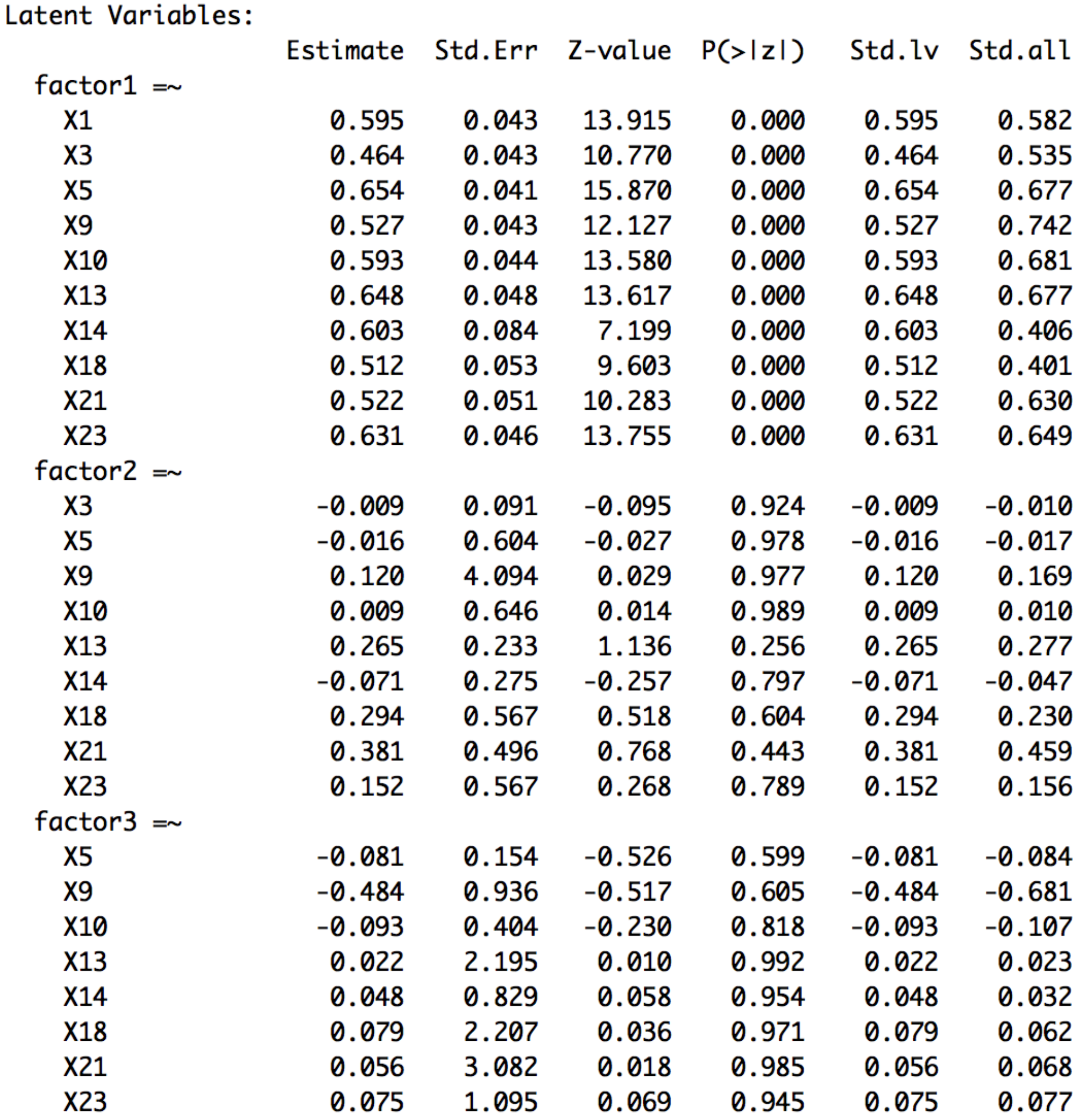
```
> summary(CFA_2factor.model, fit.measures = TRUE, standardized = TRUE)
lavaan (0.5-20) converged normally after  64 iterations

  Number of observations                          1333

  Estimator                                         ML      Robust
  Minimum Function Test Statistic               56.704      39.189
  Degrees of freedom                                26          26
  P-value (Chi-square)                           0.000       0.047
  Scaling correction factor                                  1.447
    for the Yuan-Bentler correction

Model test baseline model:

  Minimum Function Test Statistic             4148.081    2238.585
  Degrees of freedom                                45          45
  P-value                                        0.000       0.000

User model versus baseline model:

  Comparative Fit Index (CFI)                    0.993       0.994
  Tucker-Lewis Index (TLI)                       0.987       0.990

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)             -16523.162  -16523.162
  Scaling correction factor                                  2.243
    for the MLR correction
  Loglikelihood unrestricted model (H1)     -16494.810  -16494.810
  Scaling correction factor                                  1.924
    for the MLR correction

  Number of free parameters                         39          39
  Akaike (AIC)                               33124.324   33124.324
  Bayesian (BIC)                             33326.937   33326.937
  Sample-size adjusted Bayesian (BIC)        33203.051   33203.051

Root Mean Square Error of Approximation:

  RMSEA                                          0.030       0.020
  90 Percent Confidence Interval          0.019  0.040       0.007  0.029
  P-value RMSEA <= 0.05                          0.999       1.000

Standardized Root Mean Square Residual:

  SRMR                                           0.015       0.015
```

```
Latent Variables:
                   Estimate  Std.Err  Z-value  P(>|z|)   Std.lv  Std.all
  factor1 =~
    X1                0.578    0.039   14.876    0.000    0.578    0.566
    X3                0.452    0.038   11.780    0.000    0.452    0.522
    X5                0.659    0.041   16.022    0.000    0.659    0.682
    X9                0.557    0.032   17.200    0.000    0.557    0.784
    X10               0.613    0.035   17.707    0.000    0.613    0.705
    X13               0.671    0.042   16.164    0.000    0.671    0.702
    X14               0.559    0.039   14.279    0.000    0.559    0.376
    X18               0.524    0.045   11.627    0.000    0.524    0.411
    X21               0.555    0.043   13.021    0.000    0.555    0.669
    X23               0.621    0.034   18.196    0.000    0.621    0.639
  factor2 =~
    X3               -0.023    0.049   -0.466    0.641   -0.023   -0.027
    X5               -0.098    0.069   -1.425    0.154   -0.098   -0.101
    X9               -0.076    0.069   -1.093    0.274   -0.076   -0.107
    X10              -0.108    0.075   -1.430    0.153   -0.108   -0.124
    X13               0.218    0.072    3.034    0.002    0.218    0.228
    X14              -0.011    0.082   -0.140    0.888   -0.011   -0.008
    X18               0.306    0.073    4.205    0.000    0.306    0.240
    X21               0.297    0.089    3.344    0.001    0.297    0.358
    X23               0.161    0.069    2.322    0.020    0.161    0.166

Regressions:
                   Estimate  Std.Err  Z-value  P(>|z|)   Std.lv  Std.all
  factor1 ~
    factor2           0.000                               0.000    0.000
```

CFA Three-Factor Syntax



```
lavaan (0.5-20) converged normally after 245 iterations

  Number of observations                          1333

  Estimator                                         ML      Robust
  Minimum Function Test Statistic               31.402      32.372
  Degrees of freedom                                18          18
  P-value (Chi-square)                           0.026       0.020
  Scaling correction factor                                  0.970
    for the Yuan-Bentler correction

Model test baseline model:

  Minimum Function Test Statistic             4148.081    2238.585
  Degrees of freedom                                45          45
  P-value                                        0.000       0.000

User model versus baseline model:

  Comparative Fit Index (CFI)                    0.997       0.993
  Tucker-Lewis Index (TLI)                       0.992       0.984

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)             -16510.511  -16510.511
  Scaling correction factor                                  2.290
    for the MLR correction
  Loglikelihood unrestricted model (H1)     -16494.810  -16494.810
  Scaling correction factor                                  1.924
    for the MLR correction

  Number of free parameters                         47          47
  Akaike (AIC)                               33115.022   33115.022
  Bayesian (BIC)                             33359.195   33359.195
  Sample-size adjusted Bayesian (BIC)        33209.897   33209.897

Root Mean Square Error of Approximation:

  RMSEA                                          0.024       0.024
  90 Percent Confidence Interval          0.008  0.037       0.009
  P-value RMSEA <= 0.05                          1.000       0.999

Standardized Root Mean Square Residual:

  SRMR                                           0.012       0.012
```



```
Latent Variables:
                   Estimate  Std.Err  Z-value  P(>|z|)   Std.lv  Std.all
  factor1 =~
    X1                0.595    0.043   13.915    0.000    0.595    0.582
    X3                0.464    0.043   10.770    0.000    0.464    0.535
    X5                0.654    0.041   15.870    0.000    0.654    0.677
    X9                0.527    0.043   12.127    0.000    0.527    0.742
    X10               0.593    0.044   13.580    0.000    0.593    0.681
    X13               0.648    0.048   13.617    0.000    0.648    0.677
    X14               0.603    0.084    7.199    0.000    0.603    0.406
    X18               0.512    0.053    9.603    0.000    0.512    0.401
    X21               0.522    0.051   10.283    0.000    0.522    0.630
    X23               0.631    0.046   13.755    0.000    0.631    0.649
  factor2 =~
    X3               -0.009    0.091   -0.095    0.924   -0.009   -0.010
    X5               -0.016    0.604   -0.027    0.978   -0.016   -0.017
    X9                0.120    4.094    0.029    0.977    0.120    0.169
    X10               0.009    0.646    0.014    0.989    0.009    0.010
    X13               0.265    0.233    1.136    0.256    0.265    0.277
    X14              -0.071    0.275   -0.257    0.797   -0.071   -0.047
    X18               0.294    0.567    0.518    0.604    0.294    0.230
    X21               0.381    0.496    0.768    0.443    0.381    0.459
    X23               0.152    0.567    0.268    0.789    0.152    0.156
  factor3 =~
    X5               -0.081    0.154   -0.526    0.599   -0.081   -0.084
    X9               -0.484    0.936   -0.517    0.605   -0.484   -0.681
    X10              -0.093    0.404   -0.230    0.818   -0.093   -0.107
    X13               0.022    2.195    0.010    0.992    0.022    0.023
    X14               0.048    0.829    0.058    0.954    0.048    0.032
    X18               0.079    2.207    0.036    0.971    0.079    0.062
    X21               0.056    3.082    0.018    0.985    0.056    0.068
    X23               0.075    1.095    0.069    0.945    0.075    0.077
```

CONCLUDING REMARKS

Wrapping Up

- Today we discussed the world of exploratory factor analysis and found the following:
  - PCA is what people typically run when they are after EFA
  - ML EFA is a better option to pick (likelihood based)
    - Constraints employed are hidden!
    - Rotations can break without you realizing they do
  - ML EFA can be shown to be equal to CFA for certain models
  - Overall, CFA is still your best bet