Principal Components Analysis using R

Francis Huang / huangf@missouri.edu

November 2, 2016

Principal components analysis (PCA) is a convenient way to reduce high dimensional data into a smaller number number of 'components.' PCA has been referred to as a data reduction/compression technique (i.e., dimensionality reduction). PCA is often used as a means to an end and is not the end in itself. For example, instead of performing a regression with six independent (and highly correlated) variables, we may be able to compress the data into one or two meaningful components instead and use these in our models instead of the original six variables. Using less variables reduces possible problems associated with multicollinearity. This decreases the problems of redundancy.

Note that PCA has often been called exploratory factor analysis (EFA) probably due to the fact that many software programs list PCA under the factor analysis heading (e.g., in SPSS, analyze–> dimension reduction–> factor; PROC FACTOR in SAS). They are NOT the same. Often though– both procedures will produce similar results (depending on the struture of the dataset) and both can be used to reduce high dimensional data into a smaller number of components or factors. However, the causal structure of both procedures differ. Note the directions of the arrows in both procedures. For a FA, the underlying assumption is that the covariation among observed variables is *caused* by a common factor (or factors or latent variables). Both procedures can be used for exploratory analysis. EFA (not PCA) is often used as a precursor to Confirmatory Factor Analysis.

1. Manually running a principal components analysis

The following example uses sample classroom literacy data (n = 120). We are interested in six variables (rhyme awareness, beginning sound awareness, alphabet recognition, letter sound knowledge, spelling, and concept of word) and will remove the first variable from the dataset (gender). The six variables of interest are subtasks from the **Phonological Awareness Literacy Screening (PALS)** assessment. The raw data is imported and then a correlation matrix is generated (this is using simulated data based on the original correlation matrix). The correlation matrix will then be used to run our PCA.

```
dat<-read.csv("http://faculty.missouri.edu/huangf/data/mvnotes/READING120n.csv")
str(dat) #just checking on our data
```

```
## 'data.frame':    120 obs. of  7 variables:
##  $ ï..GEN  : Factor w/ 2 levels "F","M": 2 1 2 1 1 2 2 2 1 1 ...
##  $ rhyme   : int  10 10 9 5 2 5 8 4 3 9 ...
##  $ Begsnd  : int  10 10 10 10 10 6 5 3 7 10 ...
##  $ ABC     : int  6 22 23 10 4 22 25 26 18 26 ...
##  $ LS      : int  7 19 15 3 0 8 20 16 8 17 ...
##  $ Spelling: int  4 9 5 2 0 17 12 3 3 15 ...
##  $ COW     : int  7 15 6 3 2 6 4 0 0 15 ...
```

```
options(digits=3) #just so we don't get so many digits in our results
dat<-dat[,-1] #removing the first variable which is gender
p<-ncol(dat) #no of variables
R<-cor(dat) #saving the correlation matrix
R #displaying it-- note: if you put a parenthesis around your statement, it will also print the output
```

```
##          rhyme Begsnd   ABC    LS Spelling   COW
## rhyme    1.000  0.616 0.499 0.677    0.668 0.693
## Begsnd   0.616  1.000 0.285 0.347    0.469 0.469
## ABC      0.499  0.285 1.000 0.796    0.589 0.598
## LS       0.677  0.347 0.796 1.000    0.758 0.749
## Spelling 0.668  0.469 0.589 0.758    1.000 0.767
## COW      0.693  0.469 0.598 0.749    0.767 1.000
```

For some, the first stop is to check if the data can be reduced. This relies on variables being correlated with each other and allows them to be combined. If not (i.e., if the variables were all orthogonal), there would be no way to combine the variables as factors or components. One basic test is Bartlett's test of sphericity (as it is called in SPSS)– the null hypothesis of the test is that the correlation matrix is an identity matrix– or that the matrix has one's on the diagonal and zeroes on all the off diagonals. The test statistic follows a chi square distribution and to proceed, we would want to see statistically significant results.

```
#Highlight from the function name to the ending bracket and run. After, a new function called bart will
bart<-function(dat){ #dat is your raw data
   R<-cor(dat)
   p<-ncol(dat)
   n<-nrow(dat)
   chi2<- -((n-1)-((2*p)+5)/6 ) * log(det(R)) #this is the formula
   df<-(p*(p-1)/2)
   crit<-qchisq(.95,df) #critical value
   p<-pchisq(chi2,df,lower.tail=F) #pvalue
   cat("Bartlett's test of sphericity: X2(",
    df,")=",chi2,", p=",
   round(p,3),sep="" )
}
```

The above lines are a function that we just created. Now we have a homemade `bart` function and we pass it the raw data we want to analyze.

```
bart(dat)
```

```
## Bartlett's test of sphericity: X2(15)=497, p=0
```

Results indicate that the p value is $< .001$ (not really 0!) and is statistically significant. PCA can be done. Note: I don't often use this test but others tend to use this. For one, when I have data, I do end up checking, inspecting the correlation matrix or use items in a way where I know they will be correlated with each other to some extent. But, in any case, good to know this. Now, moving on…

```
e<-eigen(R) #solving for the eigenvalues and eigenvectors from the correlation matrix
str(e)
```

```
## List of 2
##  $ values : num [1:6] 4.042 0.873 0.42 0.299 0.232 ...
##  $ vectors: num [1:6, 1:6] -0.42 -0.307 -0.385 -0.446 -0.436 ...
```

NOTE: `e` is a list of two important sets of values. The first set contains the eigenvalues and the second is the set of eigenvectors for the corresponding eigenvalues. We will store them separately and use them in our analyses. The eigenvalue equation is an important equation that is used regularly in MV stats– though fortunately, computers will solve those for us (as we have solved this before using a 2 x 2 matrix and it took a while).

```
L<-e$values #placing the eigenvalues in L
Vm<-matrix(0,nrow=p,ncol=p) #creating a p x p matrix with zeroes.
#Vm is an orthogonal matrix since all correlations between variable are 0.
diag(Vm)<-L #putting the eigenvalues in the diagonals
Vm #check-- matrix with eigenvalues on the diagonals
```

```
##      [,1]  [,2] [,3]  [,4]  [,5]  [,6]
## [1,] 4.04 0.000 0.00 0.000 0.000 0.000
## [2,] 0.00 0.873 0.00 0.000 0.000 0.000
## [3,] 0.00 0.000 0.42 0.000 0.000 0.000
## [4,] 0.00 0.000 0.00 0.299 0.000 0.000
## [5,] 0.00 0.000 0.00 0.000 0.232 0.000
## [6,] 0.00 0.000 0.00 0.000 0.000 0.134
```

```
e$vectors #these are the eigenvectors-- these are the standardized regression weights
```

```
##        [,1]    [,2]    [,3]    [,4]    [,5]    [,6]
## [1,] -0.420  0.2993 -0.0927  0.8002 -0.1228  0.2642
## [2,] -0.307  0.7597  0.4314 -0.3329  0.0299 -0.1754
## [3,] -0.385 -0.4678  0.6571 -0.0846  0.0660  0.4354
## [4,] -0.446 -0.3346  0.0653  0.1441 -0.1308 -0.8044
## [5,] -0.436 -0.0389 -0.4390 -0.4379 -0.6046  0.2420
## [6,] -0.439 -0.0290 -0.4201 -0.1709  0.7727  0.0647
```

```
loadings<-e$vectors %*% sqrt(Vm) #these are the loadings
#or the correlation of the component variables with the original variables-- sometimes referred to as t
#SPSS refers to this as the component matrix
```

To reproduce the original correlation matrix (just shown again below):

```
cor(dat) #original correlation matrix
```

```
##          rhyme Begsnd   ABC    LS Spelling   COW
## rhyme    1.000  0.616 0.499 0.677    0.668 0.693
## Begsnd   0.616  1.000 0.285 0.347    0.469 0.469
## ABC      0.499  0.285 1.000 0.796    0.589 0.598
## LS       0.677  0.347 0.796 1.000    0.758 0.749
## Spelling 0.668  0.469 0.589 0.758    1.000 0.767
## COW      0.693  0.469 0.598 0.749    0.767 1.000
```

We can use the eigenvalues and the eigenvectors.

```
e$vectors %*% Vm %*% t(e$vectors) # V L V`
```

```
##       [,1]  [,2]  [,3]  [,4]  [,5]  [,6]
## [1,] 1.000 0.616 0.499 0.677 0.668 0.693
## [2,] 0.616 1.000 0.285 0.347 0.469 0.469
## [3,] 0.499 0.285 1.000 0.796 0.589 0.598
## [4,] 0.677 0.347 0.796 1.000 0.758 0.749
## [5,] 0.668 0.469 0.589 0.758 1.000 0.767
## [6,] 0.693 0.469 0.598 0.749 0.767 1.000
```

```
#NOTE: I use L above in the comment but L is really a diagonal matrix with the L on the diagonals

#This is the proportion of variance accounted for by each PC
L/length(L)
```

```
## [1] 0.6736 0.1454 0.0700 0.0498 0.0387 0.0224
```

So the original correlation matrix can be reproduced as: the eigenvectors times the diagonal matrix of eigenvalues (has to be a matrix or will be nonconformable) times the tranpose of the matrix of eigenvectors.

To compute component scores– PCA1 = a1z1 + a2z2 + … + a6z6– we need the *weights* (the eigenvectors) and the *standardized* values of the original data (z1, z2, etc.). You can have as many component scores as you have variables (but not all will be useful, need to decide how many to retain– if you end up with 6 components from your 6 variables, we haven't reduced/compressed anything!). To compute the PCA scores (using matrices), zValues x eigenvectors:

```
zdat<-scale(dat) #this is just to standardize the original data, M = 0, SD =1
pca.scores<- zdat %*% e$vectors #scaled values x vectors
colnames(pca.scores)<-c('pca1','pca2','pca3','pca4','pca5','pca6') #just quickly naming the columns
head(pca.scores) #just to show some component scores
```

```
##        pca1   pca2     pca3   pca4     pca5    pca6
## [1,]  1.119  2.226 -0.80240  0.849 -0.07816 -0.2028
## [2,] -1.345  0.536 -0.00557  0.327  0.21459 -0.2122
## [3,] -0.181  0.610  0.90468  0.477 -0.22323 -0.0487
## [4,]  2.227  1.663  0.07935 -0.374  0.00992 -0.0769
## [5,]  3.370  1.922 -0.22066 -0.990  0.22398 -0.4874
## [6,]  0.427 -0.597 -0.64273 -1.111 -1.20665  1.0341
```

```
#NOTE: these scores are scaled such that they have a mean of zero and the variance comes out to the eig
round(colMeans(pca.scores),2) #each pca score has a mean of zero
```

```
## pca1 pca2 pca3 pca4 pca5 pca6
##    0    0    0    0    0    0
```

```
apply(pca.scores,2,var) #if you get the variance PER column...
```

```
##  pca1  pca2  pca3  pca4  pca5  pca6
## 4.042 0.873 0.420 0.299 0.232 0.134
```

```
e$values #...it will be the same as the eigenvalues too
```

```
## [1] 4.042 0.873 0.420 0.299 0.232 0.134
```

The component scores will differ from what you have with SPSS which scales the components so that it has a mean of zero and an SD of 1. To get the same results (the +/- may just be switched but that does not matter):

```
head(scale(pca.scores)[,1]) #check this with SPSS output- just the first 6 component scores
```

```
## [1]  0.557 -0.669 -0.090  1.108  1.676  0.212
```

The first component accounts for the most variance (4.042/6). To show the percentage of variance accounted for by each variable– divide the eigenvalues by the number of variables since each scaled variable has a variance of 1. First component accounts for 67% of the variance, second 15%, etc.

```
e$values/p
```

```
## [1] 0.6736 0.1454 0.0700 0.0498 0.0387 0.0224
```

Also note, a property of the principal component scores is that they are not correlated with each other– they are completely orthogonal. To see this, generate a correlation matrix based on the pca.scores dataset. So we can see why using PC scores also reduces multicollinearity when these components, if ever, are used in a regression.

```
round(cor(pca.scores),2) #that is not an error, I had it round to 2 decimal places to make it clearer.
```

```
##      pca1 pca2 pca3 pca4 pca5 pca6
## pca1    1    0    0    0    0    0
## pca2    0    1    0    0    0    0
## pca3    0    0    1    0    0    0
## pca4    0    0    0    1    0    0
## pca5    0    0    0    0    1    0
## pca6    0    0    0    0    0    1
```

To see the structure matrix– which is the correlation of the component scores with the original variables– we can get the correlation between the original values and the newly created component scores. We just correlate the first PC here since that's the one that accounts for the most variance and the second one accounts for much less.

```
cor(dat[,1:6],pca.scores[,1]) #this is correlating the six original variables with the first PC.
```

```
##             [,1]
## rhyme    -0.845
## Begsnd   -0.617
## ABC      -0.774
## LS       -0.896
## Spelling -0.876
## COW      -0.882
```

This is the same as (without having to compute the component scores first):

```
comp.matrix<-e$vectors %*% sqrt(Vm) #sometimes referred to as P matrix
#or eigenvectors x sqrt(Vm): P %*% t(P) is equal to the R matrix.
comp.matrix
```

```
##        [,1]    [,2]    [,3]    [,4]    [,5]    [,6]
## [1,] -0.845  0.2796 -0.0601  0.4376 -0.0592  0.0968
## [2,] -0.617  0.7097  0.2796 -0.1821  0.0144 -0.0643
## [3,] -0.774 -0.4370  0.4259 -0.0463  0.0318  0.1596
## [4,] -0.896 -0.3126  0.0423  0.0788 -0.0630 -0.2949
## [5,] -0.876 -0.0364 -0.2845 -0.2394 -0.2913  0.0887
## [6,] -0.882 -0.0271 -0.2722 -0.0935  0.3723  0.0237
```

If you square the correlations, this will give you how much variance the PC accounts for in the original variables. These are referred to as the **communalities**.

```
comp.matrix[,1]^2 #comp 1 accounts can account for 71% of the variance of the first variable
```

```
## [1] 0.714 0.381 0.599 0.803 0.768 0.777
```

2. Using a function for running a principal components analysis

You can compare our results above (the loadings and the eigenvectors) to what you would get if done in SPSS (or in this case, in `R` which is done with the `psych` package– install it if you have not already done so).

```
#install.packages("psych")
library(psych)
```

You can run the Bartlett test as we did and you can also run a Kaiser, Meyer, Olkin (KMO) Measure of Sampling Adequacy (MSA, see Word doc class notes for interpretation).

```
cortest.bartlett(R,n=nrow(dat)) #indicating that n equals the number of rows in our dataset. If not, it
```

```
## $chisq
## [1] 497
##
## $p.value
## [1] 2.06e-96
##
## $df
## [1] 15
```

```r
KMO(R)
```

```
## Kaiser-Meyer-Olkin factor adequacy
## Call: KMO(r = R)
## Overall MSA =  0.83
## MSA for each item =
##    rhyme   Begsnd      ABC       LS Spelling      COW
##     0.84     0.77     0.80     0.77     0.89     0.90
```

The following lines perform the actual PCA.

```r
pca<-principal(dat,nfactor=p,rotate="none") #forcing to extract p=6 components
pca
```

```
## Principal Components Analysis
## Call: principal(r = dat, nfactors = p, rotate = "none")
## Standardized loadings (pattern matrix) based upon correlation matrix
##           PC1   PC2   PC3   PC4   PC5   PC6 h2       u2 com
## rhyme    0.84  0.28 -0.06 -0.44 -0.06  0.10  1  1.1e-16 1.8
## Begsnd   0.62  0.71  0.28  0.18  0.01 -0.06  1 -2.2e-16 2.5
## ABC      0.77 -0.44  0.43  0.05  0.03  0.16  1  5.6e-16 2.3
## LS       0.90 -0.31  0.04 -0.08 -0.06 -0.29  1 -2.2e-16 1.5
## Spelling 0.88 -0.04 -0.28  0.24 -0.29  0.09  1 -4.4e-16 1.6
## COW      0.88 -0.03 -0.27  0.09  0.37  0.02  1 -4.4e-16 1.6
##
##                        PC1  PC2  PC3  PC4  PC5  PC6
## SS loadings           4.04 0.87 0.42 0.30 0.23 0.13
## Proportion Var        0.67 0.15 0.07 0.05 0.04 0.02
## Cumulative Var        0.67 0.82 0.89 0.94 0.98 1.00
## Proportion Explained  0.67 0.15 0.07 0.05 0.04 0.02
## Cumulative Proportion 0.67 0.82 0.89 0.94 0.98 1.00
##
## Mean item complexity =  1.9
## Test of the hypothesis that 6 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
```

NOTE: What is being shown is more commonly referred to as the structure matrix (SPSS calls it the component matrix)– which contain the zero-order correlations between the factor and the original variables. You can compare these to the results generated using the eigenvectors x the square root of the matrix of eigenvalues (really a diagonal matrix with a the square root of the eigenvalues on the diagonal).

```r
loadings<-e$vectors %*% sqrt(Vm) #these are the correlations as we have shown earlier
loadings #signs are just different
```

```
##        [,1]    [,2]    [,3]    [,4]    [,5]    [,6]
## [1,] -0.845  0.2796 -0.0601  0.4376 -0.0592  0.0968
## [2,] -0.617  0.7097  0.2796 -0.1821  0.0144 -0.0643
## [3,] -0.774 -0.4370  0.4259 -0.0463  0.0318  0.1596
## [4,] -0.896 -0.3126  0.0423  0.0788 -0.0630 -0.2949
## [5,] -0.876 -0.0364 -0.2845 -0.2394 -0.2913  0.0887
## [6,] -0.882 -0.0271 -0.2722 -0.0935  0.3723  0.0237
```

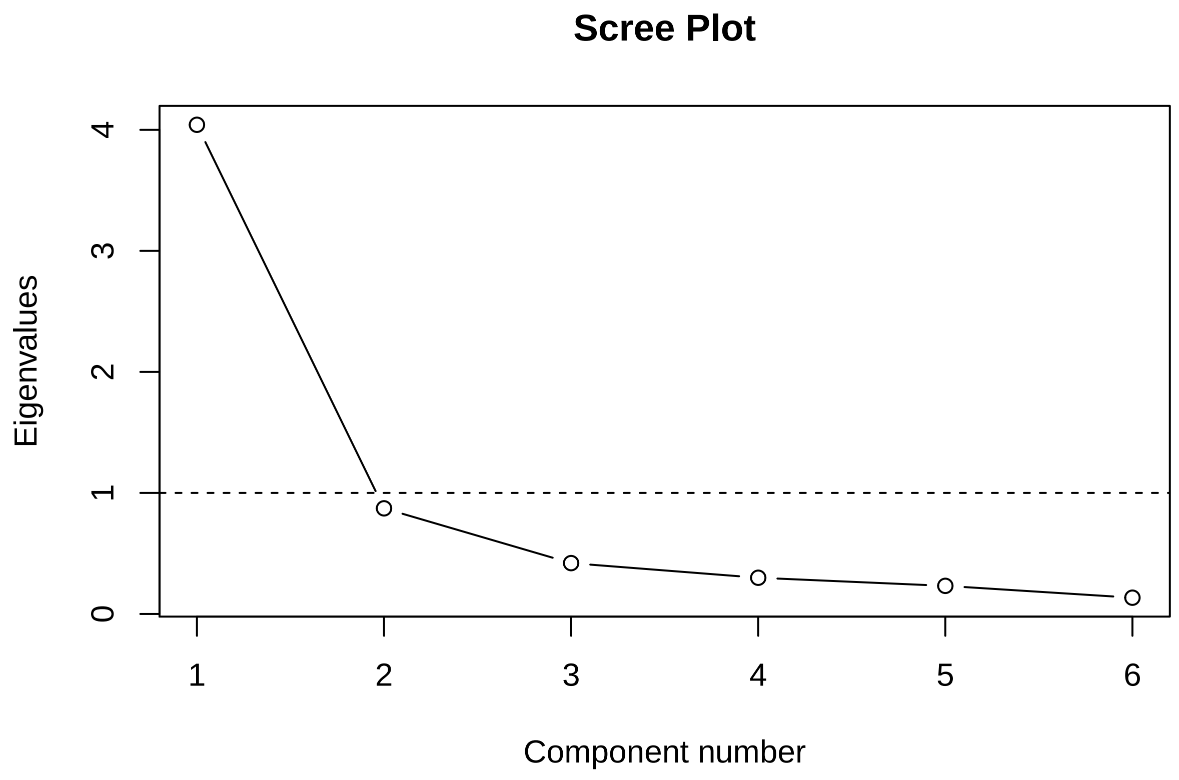
NOTE: the correlations are the same. Except that eigenvectors 1 and 4 have the opposite signs in their loadings compared to the output from SPSS and R (using the principal function in base R). To get the exact same output, the syntax below can be applied to change the signs (if this bothers you).

```
sign <- vector(mode = "numeric", length = p)
sign <- sign(colSums(loadings))
#generates a -1 if total is a negative number and keeps it at +1 if > 0.
#This is then applied to the loadings to flip the signs. See below:
loadings2<-loadings %*% diag(sign)
loadings2
```

```
##       [,1]    [,2]    [,3]    [,4]    [,5]    [,6]
## [1,] 0.845  0.2796 -0.0601 -0.4376 -0.0592  0.0968
## [2,] 0.617  0.7097  0.2796  0.1821  0.0144 -0.0643
## [3,] 0.774 -0.4370  0.4259  0.0463  0.0318  0.1596
## [4,] 0.896 -0.3126  0.0423 -0.0788 -0.0630 -0.2949
## [5,] 0.876 -0.0364 -0.2845  0.2394 -0.2913  0.0887
## [6,] 0.882 -0.0271 -0.2722  0.0935  0.3723  0.0237
```

You can also use the eigenvalues to draw a scree plot like in SPSS (in case you want to refer to something like Kaiser's rule later on– which many studies have shown to not be very exact– but it is simple):

```
#remember, L here refers to the eigenvalues we created earlier
plot(L,main="Scree Plot",ylab="Eigenvalues",xlab="Component number",type='b')
abline(h=1, lty=2)
```



```
L/p #this shows you what percent of overall variance each component accounts for, the first
```

```
## [1] 0.6736 0.1454 0.0700 0.0498 0.0387 0.0224
```

```
#component accounts for 63% of the variance, the second, 15%--> have shown this already twice before.
```

3. Number of components to retain

Several methods have been suggested to decide how many components/factors to retain. There are several methods, some of which are better than others. Kaiser's rule suggests eigenvalues > 1 but that is not always a good way to decide (the thought being that if the L < 1 then the component accounts for less variance than one of the original variables). Others may suggest using the % of variance accounted for (as long as it is 'meaningful'). One of the better recognized ways to empirically decide how many factors to retain is to run a parallel analysis (Horn, 1965).

From the `hornpa` function: The program generates a specified number of **random** datasets based on the number of variables entered by the user. As the correlation matrices have an eigenvalue based on random noise, these eigenvalues are computed for the each dataset and collected. The mean and the specified percentile (95th is the default) are computed. The output table shows how large eigenvalues can be as a result of merely using randomly generated datasets. If the user's own dataset has an actual eigenvalue greater than the generated eigenvalue (which is based on random noise), that lends support to retain that factor. In other words, if the i(th) eigenvalue from the actual data was larger than the percentile of the (i)th eigenvalue generated using randomly generated data, empirical support is provided to retain that factor. Install the `hornpa` package if you have not already done so.

We specify how many variables are in the original dataset (k = 6), how big our sample is (size = 120), how many repetitions to run (reps = 500), and we set an optional seed so we get the same results if we run it again (seed = 1234).

```
library(hornpa)
hornpa(k=6,size=120,reps=500,seed=1234)
```

```
##
##  Parallel Analysis Results
##
## Method: pca
## Number of variables: 6
## Sample size: 120
## Number of correlation matrices: 500
## Seed: 1234
## Percentile: 0.95
##
## Compare your observed eigenvalues from your original dataset to the 95 percentile in the table below
##
##  Component  Mean  0.95
##          1 1.312 1.448
##          2 1.159 1.247
##          3 1.042 1.109
##          4 0.939 1.007
##          5 0.836 0.912
##          6 0.712 0.804
```

```
L #our original eigenvalues again
```

```
## [1] 4.042 0.873 0.420 0.299 0.232 0.134
```

Since our first eigenvalue (4.04) is > 1.448, we have support to retain the first component. The second eigenvalue (0.87) is < 1.247– so we may opt to drop the second component and just retain the first one.

Try to find a consensus among the different methods. For the above example, this satisfies the eigenvalue > 1 rule AND more importantly, is suggested by parallel analysis. Although we only account for 60%+ of the variance, that may be acceptable.

References

Bartlett, M. S. (1951). The effect of standardization on a chi square approximation in factor analysis, Biometrika, 38, 337-344.

Horn, J. (1965). A rationale and test for the number of factors in factor analysis. Psychometrika, 32, 179-185