Multivariate Statistics with R

Principal Components Analysis

Aja Murray, Aja.Murray@ed.ac.uk

Overview

- Week 1: Dimension Reduction (PCA and EFA)
- Week 2: Confirmatory Factor Analysis
- Week 3: Path Analysis
- Week 4: Structural Equation Modeling I
- Week 5: Structural Equation Modeling II

This Week

- Techniques
  - *Principal Components Analysis (PCA)*
  - *Exploratory Factor Analysis (EFA)*
- Key Functions
  - *vss( )*
  - *fa.parallel( )*
  - *principal( )*
  - *fa( )*
- Reading: *Principal Components Analysis* and *Exploratory Factor Analysis* Chapters (on *Learn* under ‘Reading’)

Learning Outcomes

- Understand the principles of dimension reduction
- Understand thd difference between PCA and EFA
- Know how to perform and interpret PCA and EFA in R

Dimension Reduction

- Summarise a set of variables in terms of a smaller number of dimensions
  - *e.g., can 10 aggression items summarised in terms of ‘physical’ and ‘verbal’ aggression dimensions?*
    1. *I hit someone*
    2. *I kicked someone*
    3. *I shoved someone*
    4. *I battered someone*
    5. *I physically hurt someone on purpose*
    6. *I deliberately insulted someone*
    7. *I swore at someone*
    8. *I threatened to hurt someone*
    9. *I called someone a nasty name to their face*
    10. *I shouted mean things at someone*

Uses of dimension reduction techniques

- Theory testing
  - *What are the number and nature of dimensions that best describe a theoretical construct?*
- Test construction
  - *How should I group my items into subscales?*
  - *Which items are the best measures of my constructs?*
- Pragmatic
  - *I have multicollinearity issues/too many variables, how can I defensibly combine my variables?*

Our running example

- A researcher has collected n=1000 responses to our 10 aggression items
- We'll use this data to illustrate dimension reduction techniques

```
library(psych)
describe(agg.items)
```

```
##        vars    n  mean   sd median trimmed  mad   min  max range  skew kurtosis
## item1     1 1000  0.02 0.99   0.02    0.02 1.00 -3.83 2.99  6.82  0.00    -0.14
## item2     2 1000  0.02 1.01   0.02    0.03 0.98 -3.57 3.02  6.59 -0.01    -0.02
## item3     3 1000  0.02 1.00   0.03    0.02 1.02 -3.38 2.91  6.30 -0.02    -0.14
## item4     4 1000  0.04 1.01   0.08    0.05 1.02 -3.02 3.31  6.34 -0.06    -0.14
## item5     5 1000  0.02 0.98   0.03    0.02 1.00 -3.40 2.97  6.37 -0.05     0.02
## item6     6 1000  0.03 1.01   0.02    0.02 1.00 -2.93 3.28  6.21  0.14     0.06
## item7     7 1000  0.02 0.96   0.05    0.02 0.97 -2.81 3.40  6.21  0.08     0.18
## item8     8 1000  0.02 0.96   0.00    0.01 0.96 -2.84 3.39  6.23  0.15    -0.06
## item9     9 1000  0.04 0.99   0.02    0.04 0.98 -2.84 3.49  6.34  0.04     0.09
## item10   10 1000 -0.03 0.96  -0.03   -0.05 0.90 -2.81 3.30  6.11  0.16     0.05
##          se
## item1  0.03
## item2  0.03
## item3  0.03
## item4  0.03
## item5  0.03
## item6  0.03
## item7  0.03
## item8  0.03
## item9  0.03
## item10 0.03
```

PCA

- Starts with a correlation matrix

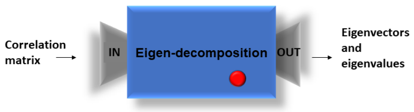
```
#compute the correlation matrix for the aggression items
round(cor(agg.items),2)
```

```
##        item1 item2 item3 item4 item5 item6 item7 item8 item9 item10
## item1   1.00  0.57  0.50  0.44  0.57  0.06  0.15  0.11  0.13   0.07
## item2   0.57  1.00  0.56  0.51  0.67  0.07  0.18  0.11  0.13   0.04
## item3   0.50  0.56  1.00  0.46  0.61  0.05  0.13  0.09  0.13   0.03
## item4   0.44  0.51  0.46  1.00  0.52  0.13  0.21  0.18  0.20   0.12
## item5   0.57  0.67  0.61  0.52  1.00  0.01  0.13  0.03  0.07   0.01
## item6   0.06  0.07  0.05  0.13  0.01  1.00  0.57  0.58  0.41   0.46
## item7   0.15  0.18  0.13  0.21  0.13  0.57  1.00  0.77  0.59   0.60
## item8   0.11  0.11  0.09  0.18  0.03  0.58  0.77  1.00  0.61   0.63
## item9   0.13  0.13  0.13  0.20  0.07  0.41  0.59  0.61  1.00   0.50
## item10  0.07  0.04  0.03  0.12  0.01  0.46  0.60  0.63  0.50   1.00
```

What PCA does

- Repackages the variance from the correlation matrix into a set of **components**
    - *components= orthogonal (i.e.,uncorrelated) linear combinations of the original variables*
- the first component is the linear combination that accounts for the most possible variance
- the second accounts for second-largest after the variance accounted for by the first is removed etc.
- each component accounts for as much remaining variance as possible
- there are as many components are there were variables in original correlation matrix

Eigendecomposition



- Components are formed using an **eigendecomposition** of the correlation matrix
- Eigendecomposition is a transformation of the correlation matrix to re-express it in terms of **eigenvectors** and **eigenvalues**

Eigenvectors and eigenvalues

- There is one eigenvector and one eigenvalue for each component
    - *Eigenvectors are sets of weights (one weight per variable in original correlation matrix)*
        - e.g., if we had 10 variables each eigenvector would contain 10 weights
        - Larger weights mean a variable makes a bigger contribution to the component
    - *Eigenvalues are a measure of the size of the variance packaged into a component*
        - Larger eigenvalues mean that the component accounts for a large proportion of the variance in the original correlation matrix

Eigendecomposition of aggression item correlation matrix

- We can use the eigen() function to conduct an eigendecomposition for our 10 aggression items

```
eigen(cor(agg.items))
```

```
## eigen() decomposition
## $values
##  [1] 3.7684958 2.7356804 0.5953319 0.5716398 0.5125667 0.4763628 0.4305519
##  [8] 0.3697766 0.3200951 0.2194991
##
## $vectors
##             [,1]       [,2]        [,3]        [,4]         [,5]         [,6]
##  [1,] -0.2870177  0.3199446  0.07143109 -0.43175730  0.563399577 -0.394183123
##  [2,] -0.3106275  0.3539678 -0.08503881 -0.09909968 -0.001258554 -0.031939338
##  [3,] -0.2826611  0.3434128  0.02420767 -0.14612253 -0.616734464  0.282948549
##  [4,] -0.3071276  0.2517745 -0.04400815  0.86277821  0.237590135 -0.002733387
##  [5,] -0.2873526  0.3965675 -0.02902837 -0.09222646 -0.077814049  0.190005840
##  [6,] -0.2898209 -0.2887244 -0.78294062 -0.01816988 -0.097574218 -0.219838183
##  [7,] -0.3821881 -0.2823866 -0.01734628 -0.08471720 -0.063206183  0.014687355
##  [8,] -0.3652672 -0.3289337  0.02308076 -0.03205596 -0.026148774 -0.019219419
##  [9,] -0.3298340 -0.2499686  0.58197751  0.10882278 -0.279335864 -0.462173009
## [10,] -0.3034151 -0.3161771  0.17825030 -0.09562427  0.384038803  0.681844667
##              [,7]       [,8]        [,9]       [,10]
##  [1,] -0.35347092 -0.1567593  0.00958351 -0.03651226
##  [2,]  0.57029316  0.2854264 -0.59385552 -0.02786398
##  [3,] -0.49368762 -0.1821845 -0.19950563 -0.05558836
##  [4,] -0.14278153 -0.1281945 -0.02388620 -0.04595395
##  [5,]  0.28097835  0.1827296  0.74414831  0.20630427
##  [6,] -0.21718997  0.3327602  0.05858465 -0.01154841
##  [7,]  0.31835488 -0.4740413  0.15556865 -0.64205513
##  [8,]  0.13737585 -0.4362779 -0.11390590  0.73046829
##  [9,] -0.08239528  0.4163793  0.07027187 -0.03812767
## [10,] -0.18049584  0.3343479 -0.08699177 -0.05216665
```

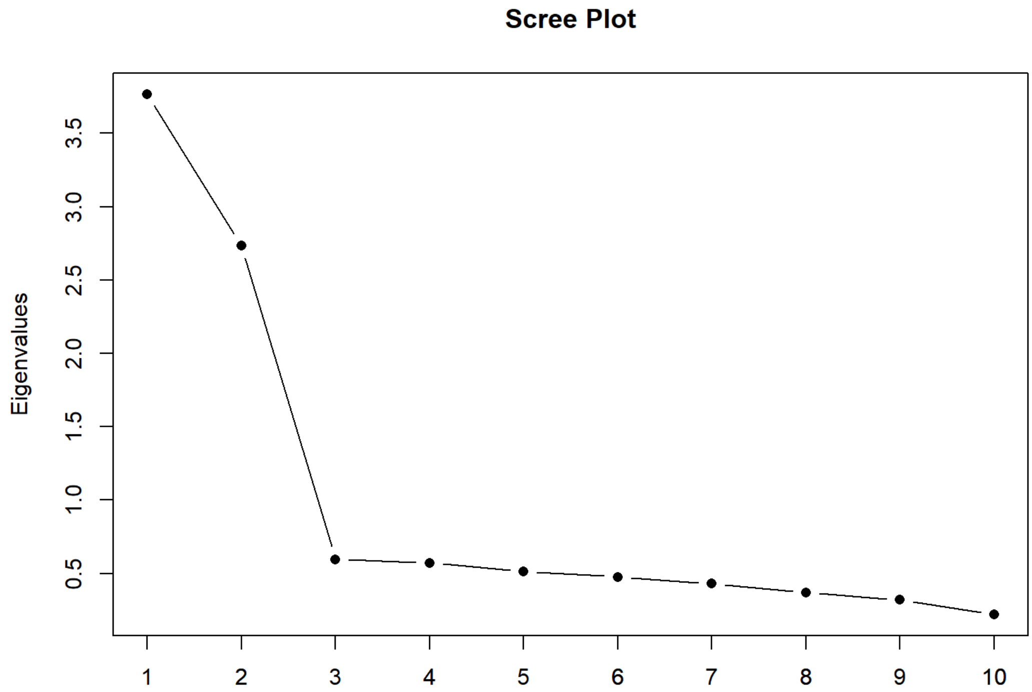
How many components to keep?

- Eigendecomposition repackages the variance but does not reduce our dimensions
- Dimension reduction comes from keeping only the largest components
- It is assumed the others can be dropped with little loss of information
- Our decisions on how many components to keep can be guided by several methods
  - *Scree plot*
  - *Minimum average partial test (MAP)*
  - *Parallel analysis*
- Our decision should also be based on substantive considerations
  - *Do the selected components make theoretical sense given our background knowledge of the construct?*

Kaiser criterion

- Keeps number of components with eigenvalue >1
- DO NOT USE Kaiser criterion
- Often suggests keeping far too many components

Scree plot

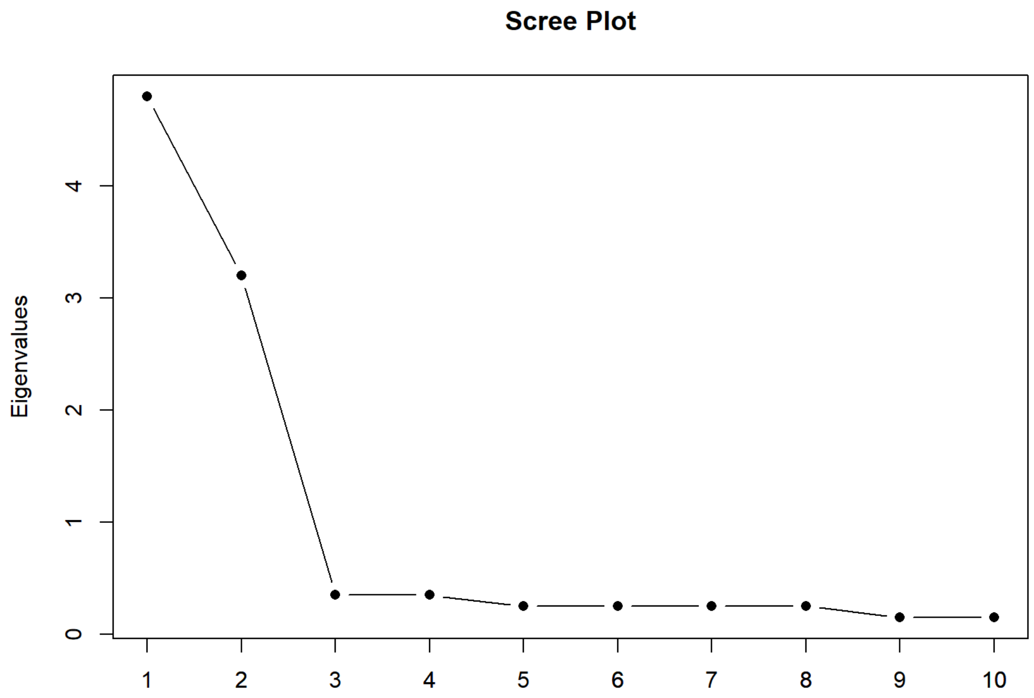


- Plots the eigenvalues
  - *x-axis is component number*
  - *y-axis is eigenvalue*
- Keep the components with eigenvalues above a kink in the plot

Further scree plot examples

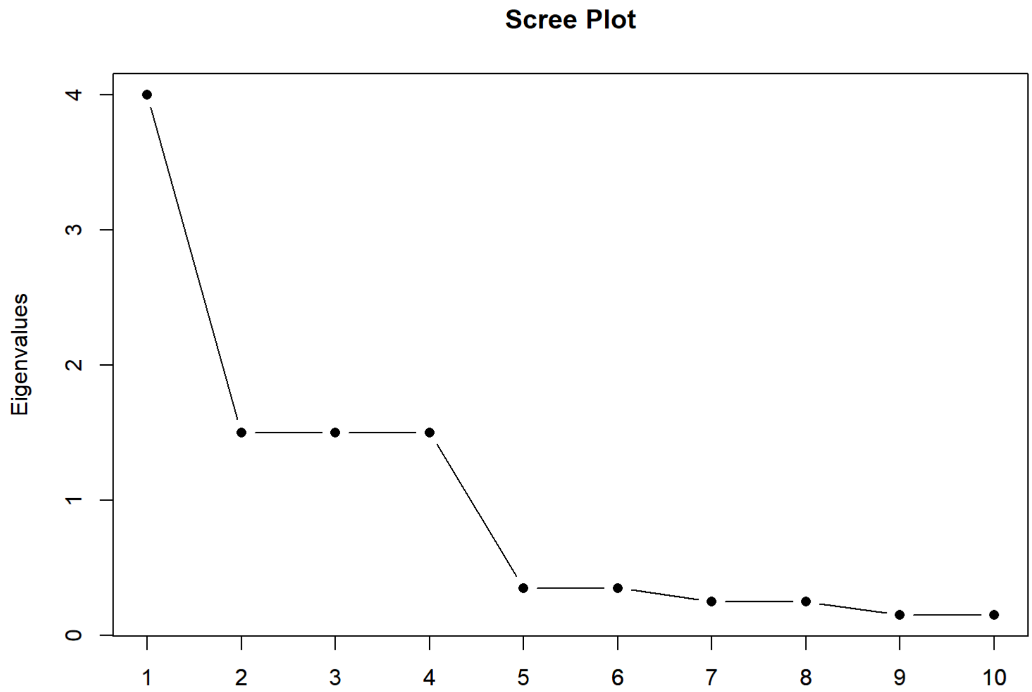
- Scree plots vary in how easy it is to interpret them

```
## [1] 10
```



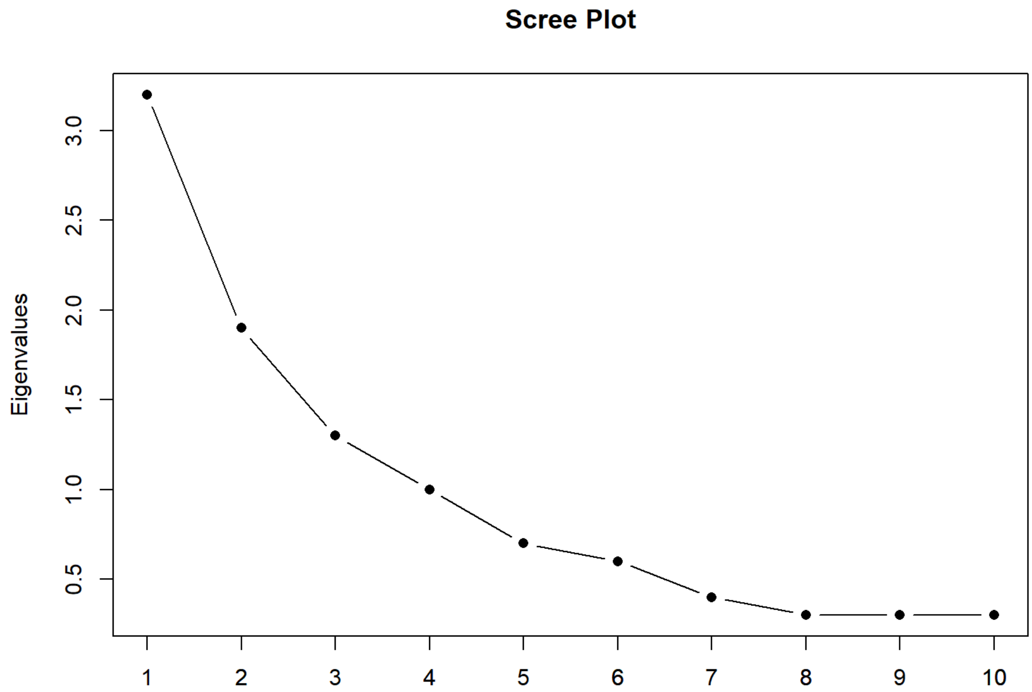
Further scree plot examples

```
## [1] 10
```



Further scree plot examples

```
## [1] 10
```



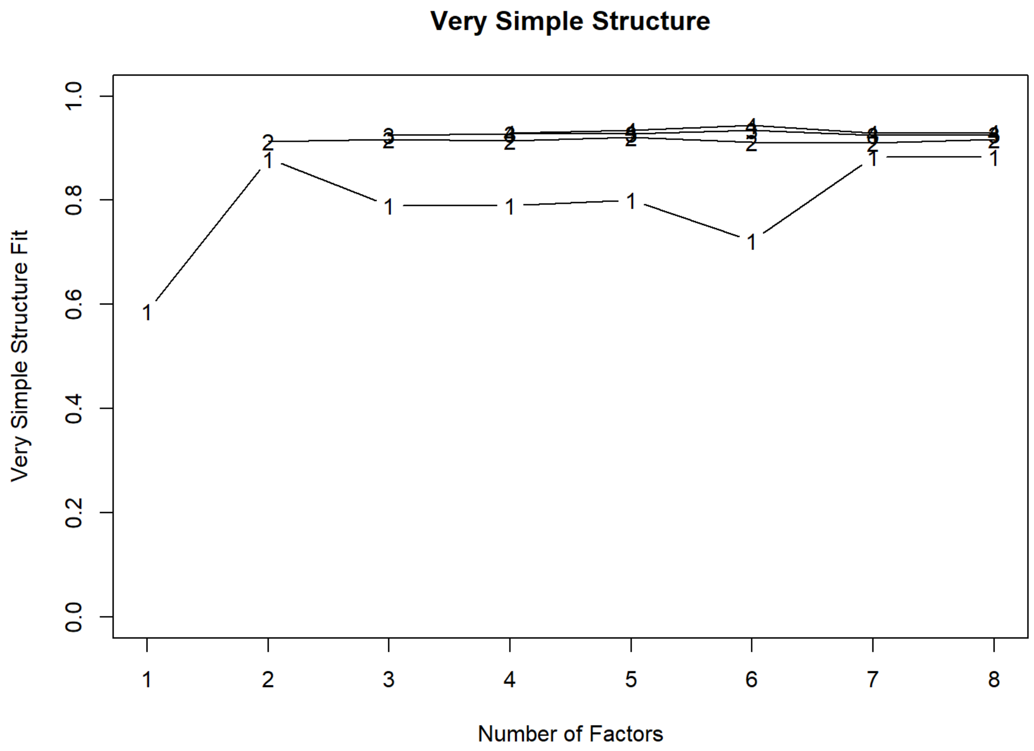
Minimum average partial test (MAP)

- Extracts components iteratively from the correlation matrix
- Computes the average squared partial correlation after each extraction
- At first this quantity goes down with each component extracted but then it starts to increase again
- MAP keeps the components from point at which the average squared partial correlation is at its smallest

MAP test for the aggression items

- We can obtain the results of the MAP test via the vss( ) function from the psych package

```
library(psych)
vss(agg.items)
```




```
##
## Very Simple Structure
## Call: vss(x = agg.items)
## Although the VSS complexity 1 shows  7  factors, it is probably more reasonable to think about  2  factors
## VSS complexity 2 achieves a maximimum of 0.92  with  5  factors
##
## The Velicer MAP achieves a minimum of 0.03  with  2  factors
## BIC achieves a minimum of  NA  with  2  factors
## Sample Size adjusted BIC achieves a minimum of  NA  with  2  factors
##
## Statistics by number of factors
##   vss1 vss2   map dof   chisq prob sqresid  fit  RMSEA  BIC SABIC complex
## 1 0.59 0.00 0.153  35 2.5e+03 0.00     9.6 0.59 0.2627 2209  2320     1.0
## 2 0.88 0.91 0.030  26 3.1e+01 0.22     2.0 0.91 0.0140 -148   -66     1.0
## 3 0.79 0.92 0.063  18 1.9e+01 0.40     1.7 0.93 0.0067 -106   -48     1.1
## 4 0.79 0.91 0.099  11 9.2e+00 0.61     1.7 0.93 0.0000  -67   -32     1.2
## 5 0.80 0.92 0.147   5 3.4e+00 0.64     1.5 0.93 0.0000  -31   -15     1.2
## 6 0.72 0.91 0.242   0 2.6e-01   NA     1.3 0.95     NA   NA    NA     1.3
## 7 0.88 0.91 0.422  -4 8.1e-07   NA     1.6 0.93     NA   NA    NA     1.2
## 8 0.88 0.92 0.466  -7 2.2e-08   NA     1.6 0.93     NA   NA    NA     1.2
##    eChisq    SRMR eCRMS eBIC
## 1 4.7e+03 2.3e-01 0.260 4489
## 2 1.1e+01 1.1e-02 0.014 -169
## 3 5.2e+00 7.6e-03 0.012 -119
## 4 3.1e+00 5.8e-03 0.012  -73
## 5 1.4e+00 4.0e-03 0.012  -33
## 6 9.9e-02 1.0e-03    NA   NA
## 7 1.9e-07 1.5e-06    NA   NA
## 8 5.1e-09 2.4e-07    NA   NA
```

The MAP values

- The averaged squared partial correlation values

```r
VSS$map
```

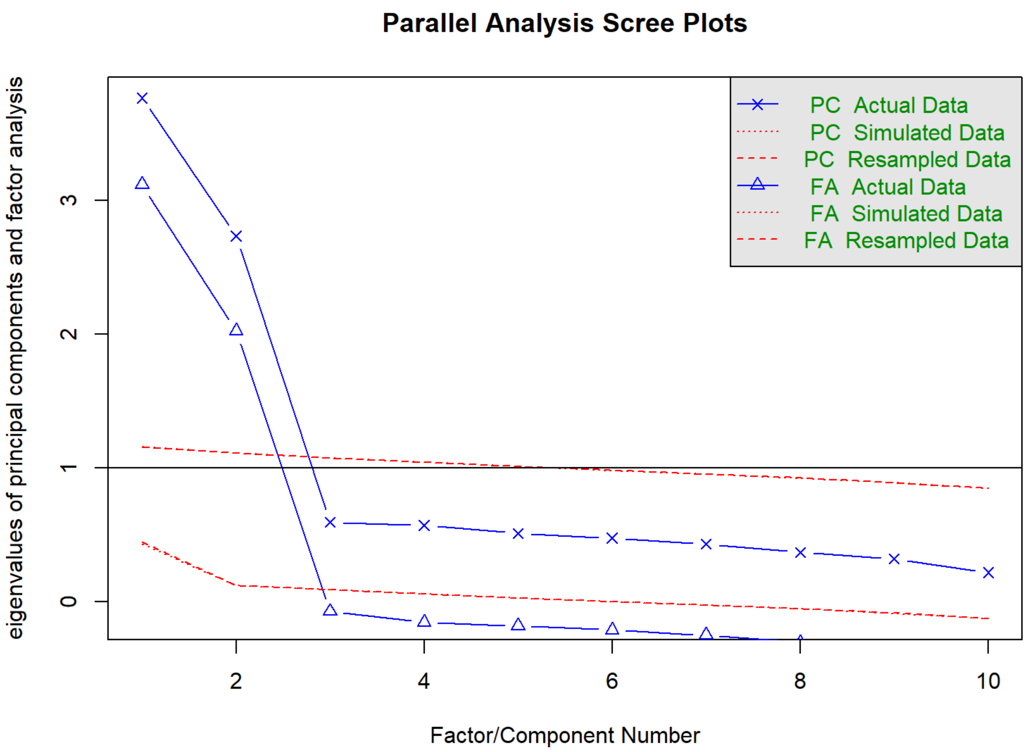
```
## [1] 0.15264717 0.02951950 0.06285463 0.09897900 0.14661427 0.24158082 0.42246750
## [8] 0.46642588
```

Parallel analysis

- Simulates datasets with same number of participants and variables but no correlations
- Computes an eigendecomposition for the simulated datasets
- Compares the average eigenvalue across the simulated datasets for each component
- If a real eigenvalue exceeds the corresponding average eigenvalue from the simulated datasets it is retained
- We can also use alternative methods to compare our real versus gthe simulated eigenvalues
  - *e.g. 95% percentile of the simulated eigenvalue distributions*

Parallel analysis for the aggression items

```
fa.parallel(agg.items, n.iter=500)
```

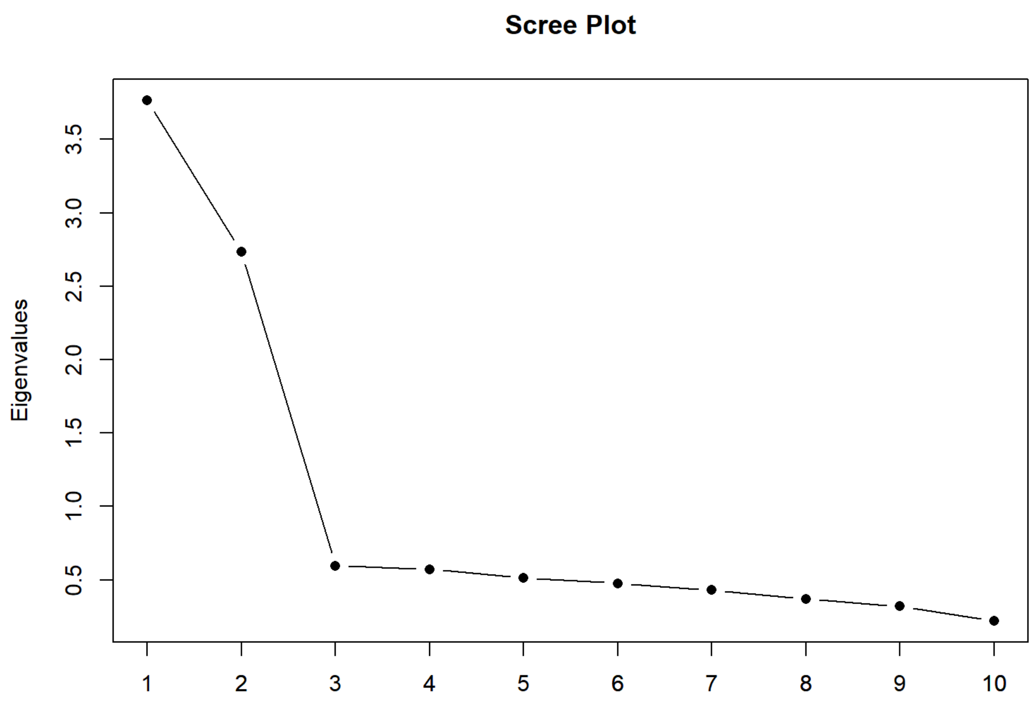



```
## Parallel analysis suggests that the number of factors =  2  and the number of components =  2
```

The fa.parallel( ) function

- Notice the function also gives us a scree plot
- We can use this to find a point of inflection
    - *Use the 'PC Actual Data' datapoints*
- However, if we want to include a scree plot in a report we should construct our own, e.g.:

```
eigenvalues<-eigen(cor(agg.items))$values
plot(eigenvalues, type = 'b', pch = 16,
     main = "Scree Plot", xlab="", ylab="Eigenvalues")
axis(1, at = 1:10, labels = 1:10)
```



Limitations of scree, MAP, and parallel analysis

- There is no one right answer about the number of components to retain
- Scree plot, MAP and parallel analysis frequently disagree
- Each method has weaknesses
  - *Scree plots are subjective and may have multiple or no obvious kinks*
  - *Parallel analysis sometimes suggest too many components*
  - *MAP sometimes suggests too few components*
- Examining the PCA solutions keeping different numbers of components should also form part of the decision

Interpreting the components

- Once we have decided how many components to keep (or to help us decide) we examine the PCA solution
- We do this based on the component loadings
    - *Component loadings are calculated from the values in the eigenvectors*
    - *They can be interpreted as the correlations between variables and components*

The component loadings

- Component loading matrix
- RC1 and RC2 columns show the component loadings

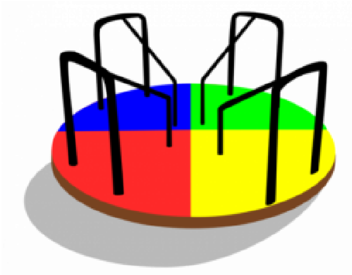
```
PC2<-principal(r=agg.items, nfactors=2)
PC2$loadings
```

```
##
## Loadings:
##        RC1    RC2
## item1         0.765
## item2         0.838
## item3         0.789
## item4   0.175  0.706
## item5         0.861
## item6   0.738
## item7   0.866  0.137
## item8   0.892
## item9   0.754  0.110
## item10  0.788
##
##                  RC1   RC2
## SS loadings    3.321 3.183
## Proportion Var 0.332 0.318
## Cumulative Var 0.332 0.650
```

Interpreting the components

1. I hit someone
2. I kicked someone
3. I shoved someone
4. I battered someone
5. I physically hurt someone on purpose
6. I deliberately insulted someone
7. I swore at someone
8. I threatened to hurt someone
9. I called someone a nasty name to their face
10. I shouted mean things at someone

Rotation of components



- Rotation takes an initial PCA solution and transforms it to make it more interpretable
- An initial PCA solution typically has:
  - *has high loadings on the first component*
  - *has a mix of positive and negative loadings on subsequent components*
  - *is difficult to interpret*
- We typically try to achieve *simple structure* with a rotation
  - *each item has a high loading on one component and close to zero loading on all others*

Initial PCA solution for the aggression items

```
PC_initial<-principal(r=agg.items, nfactors=2, rotate='none')
PC_initial$loadings
```

```
##
## Loadings:
##         PC1    PC2
## item1   0.557  0.529
## item2   0.603  0.585
## item3   0.549  0.568
## item4   0.596  0.416
## item5   0.558  0.656
## item6   0.563 -0.478
## item7   0.742 -0.467
## item8   0.709 -0.544
## item9   0.640 -0.413
## item10  0.589 -0.523
##
##                  PC1   PC2
## SS loadings    3.768 2.736
## Proportion Var 0.377 0.274
## Cumulative Var 0.377 0.650
```

Different types of rotation

- The initial (unrotated) loading matrix is transformed by multiplication by a *transformation matrix*
- Different transformation matrices are used to achieve different transformations
- The most important distinction is between *orthogonal* versus *oblique* rotations
    - *Orthogonal rotations force the components to remain uncorrelated*
        - They include varimax, quartimax and equamax
    - *Oblique rotations allow the components to be correlated*
        - They include oblimin, promax, direct oblimin, and quartimin

Choosing a rotation

- Orthogonal rotations are useful for e.g. reducing multicollinearity in regression
- Oblique rotations better reflect the reality that psychological constructs tend to be correlated
- Advice: use an oblique rotation and switch to orthogonal if correlation is very low
  - *Oblimin is a good choice for oblique rotation*
  - *Varimax is a good choice for orthogonal rotation*
  - *… but trying a few and comparing is a good idea*

Interpreting an oblique rotation

- When an orthogonal rotation is used only one loading matrix is produced
- When an oblique rotation is used two loading matrices are produced:
  - *structure matrix (correlations between the components and the variables)*
  - *pattern matrix (regression weights from the components to the variables)*
- Pattern is likely to be most useful for interpreting the components

PCA solution for the aggression items using an oblique rotation

```
PC2<-principal(r=agg.items, nfactors=2, rotate='oblimin')
```

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

```
PC2$loadings
```

```
##
## Loadings:
##        TC1    TC2
## item1          0.765
## item2          0.839
## item3          0.792
## item4   0.127  0.698
## item5          0.869
## item6   0.744
## item7   0.864
## item8   0.896
## item9   0.753
## item10  0.795
##
##                  TC1   TC2
## SS loadings    3.322 3.172
## Proportion Var 0.332 0.317
## Cumulative Var 0.332 0.649
```

How good is my PCA solution?

- A good PCA solution explains the variance of the original correlation matrix in as few components as possible

```
principal(r=agg.items, nfactors=2, rotate='oblimin')
```

```
## Principal Components Analysis
## Call: principal(r = agg.items, nfactors = 2, rotate = "oblimin")
## Standardized loadings (pattern matrix) based upon correlation matrix
##          TC1   TC2   h2   u2 com
## item1   0.02  0.77 0.59 0.41 1.0
## item2   0.01  0.84 0.71 0.29 1.0
## item3  -0.02  0.79 0.62 0.38 1.0
## item4   0.13  0.70 0.53 0.47 1.1
## item5  -0.07  0.87 0.74 0.26 1.0
## item6   0.74 -0.05 0.54 0.46 1.0
## item7   0.86  0.07 0.77 0.23 1.0
## item8   0.90 -0.01 0.80 0.20 1.0
## item9   0.75  0.05 0.58 0.42 1.0
## item10  0.79 -0.07 0.62 0.38 1.0
##
##                        TC1  TC2
## SS loadings           3.33 3.18
## Proportion Var        0.33 0.32
## Cumulative Var        0.33 0.65
## Proportion Explained  0.51 0.49
## Cumulative Proportion 0.51 1.00
##
##  With component correlations of
##     TC1  TC2
## TC1 1.00 0.15
## TC2 0.15 1.00
##
## Mean item complexity =  1
## Test of the hypothesis that 2 components are sufficient.
##
## The root mean square of the residuals (RMSR) is  0.06
##  with the empirical chi square  338.11  with prob <  4.6e-56
##
## Fit based upon off diagonal values = 0.97
```

Computing scores for the components

- After conducting a PCA you may want to create scores for the new dimensions
  - *e.g., to use in a regression*
- Simplest method is to sum the scores for all items with loadings >|.3|
- Better method is to compute them taking into account the weights

Computing component scores in R

```
PC<-principal(r=agg.items, nfactors=2, rotate='oblimin')
scores<-PC$scores
head(scores)
```

```
##              TC1        TC2
## [1,] -0.3533007  0.7753172
## [2,]  0.6259317 -0.7284820
## [3,]  0.2482317 -0.2948437
## [4,]  0.5860367 -0.8585294
## [5,]  0.3506143  1.2707699
## [6,] -0.6130446  1.9424753
```

Reporting a PCA

- Method
  - *Methods used to decide on number of factors*
  - *Rotation method*
- Results
  - *Results of MAP, parallel analysis, scree test (& any other considerations in choice of number of components)*
  - *How many components were retained*
  - *The loading matrix for the chosen solution (pattern for oblique rotations)*
  - *Correlations between components (for oblique rotations)*
  - *Variance expained by components*
  - *Labelling and interpretation of the components*

PCA Summary

- PCA is a common dimension reduction technique
- Steps are:
  - *Decide how many components to keep (scree plot, parallel analysis, MAP test)*
  - *Rotate (orthogonal versus oblique)*
  - *Interpret loadings*
- There are many subjective decision points - critical thinking is needed
- Number of components is arguably most important decision