WEEK 3
Principal Component Analysis

Data Analysis for Psychology in R 3

dapR3 Team

Department of Psychology
The University of Edinburgh

Learning Objectives

1. Understand the core principle of data reduction methods and their use in psychology
2. Understand the core goals of principal components analysis (PCA)
3. Run and interpret PCA analysis in R
4. Extract PCA scores from analyses in R

Part 1: Introduction to data reduction

Part 2: Purpose of PCA

Part 3: Eigenvalues & Eigenvectors

Part 4: Running & Interpreting PCA

Part 5: PCA scores

What's data/dimension reduction?

- Mathematical and statistical procedures
  - Reduce large set of variables to a smaller set
  - Several forms of data reduction
    - **Principal components analysis**
    - **Factor analysis**
    - Image analysis
    - Correspondence analysis
    - K-means clustering
    - Multidimensional scaling
    - Latent class analysis

When might you use data reduction?

- You work with observational data and many variables
  - Psychology (differential, industrial/organizational)
  - Genetics
  - Epidemiology

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 sub-scales?
    - Which items are the best measures of my constructs?
- Pragmatic
    - I have multicollinearity issues/too many variables, how can I defensibly combine my variables?

Questions to ask before you start

- Why are your variables correlated?
    - Agnostic/don't care
    - Believe there *are* underlying "causes" of these correlations
- What are your goals?
    - Just reduce the number of variables
    - Reduce your variables and learn about/model their underlying (latent) causes

Questions to ask before you start

- Why are your variables correlated?
  - **Agnostic/don't care**
  - Believe there *are* underlying "causes" of these correlations
- What are your goals?
  - **Just reduce the number of variables**
  - Reduce your variables and learn about/model their underlying (latent) causes

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

Our running example

- A researcher has collected n=1000 responses to our 10 aggregation 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 1.04  -0.05   -0.03 1.06 -3.01 3.33  6.34  0.09    -0.09
## item2     2 1000 -0.03 0.98  -0.07   -0.04 0.96 -2.43 2.99  5.43  0.13    -0.15
## item3     3 1000 -0.02 1.02  -0.06   -0.03 1.06 -2.85 3.38  6.23  0.10    -0.28
## item4     4 1000 -0.03 0.99  -0.06   -0.03 1.03 -3.60 3.05  6.66 -0.01    -0.19
## item5     5 1000 -0.02 0.99  -0.03   -0.03 1.00 -3.12 3.24  6.35  0.09     0.02
## item6     6 1000  0.03 1.05   0.06    0.05 1.05 -3.37 3.34  6.71 -0.13    -0.13
## item7     7 1000  0.00 0.98  -0.01   -0.01 0.98 -3.34 3.68  7.02  0.02     0.18
## item8     8 1000  0.01 1.03   0.02    0.01 1.03 -2.92 3.32  6.23  0.07    -0.04
## item9     9 1000 -0.01 1.01  -0.01   -0.01 1.01 -2.90 3.16  6.06  0.06    -0.04
## item10   10 1000 -0.02 1.02  -0.05   -0.03 1.03 -3.10 4.19  7.29  0.11     0.06
##          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
```

End of Part 1

Part 1: Introduction to data reduction

Part 2: Purpose of PCA

Part 3: Eigenvalues & Eigenvectors

Part 4: Running & Interpreting PCA

Part 5: PCA scores

Principal components analysis

- Goal is explaining as much of the total variance in a data set as possible
  - Starts with original data
  - Calculates covariances (correlations) between variables
  - Applies procedure called **eigendecomposition** to calculate a set of linear composites of the original variables

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  0.58  0.51  0.45  0.58  0.09  0.12  0.10  0.12   0.10
## item2   0.58  1.00  0.59  0.51  0.67  0.07  0.13  0.12  0.12   0.09
## item3   0.51  0.59  1.00  0.49  0.62  0.05  0.12  0.10  0.11   0.12
## item4   0.45  0.51  0.49  1.00  0.55  0.08  0.13  0.10  0.14   0.08
## item5   0.58  0.67  0.62  0.55  1.00  0.03  0.10  0.07  0.09   0.07
## item6   0.09  0.07  0.05  0.08  0.03  1.00  0.60  0.63  0.46   0.48
## item7   0.12  0.13  0.12  0.13  0.10  0.60  1.00  0.79  0.59   0.62
## item8   0.10  0.12  0.10  0.10  0.07  0.63  0.79  1.00  0.61   0.62
## item9   0.12  0.12  0.11  0.14  0.09  0.46  0.59  0.61  1.00   0.46
## item10  0.10  0.09  0.12  0.08  0.07  0.48  0.62  0.62  0.46   1.00
```

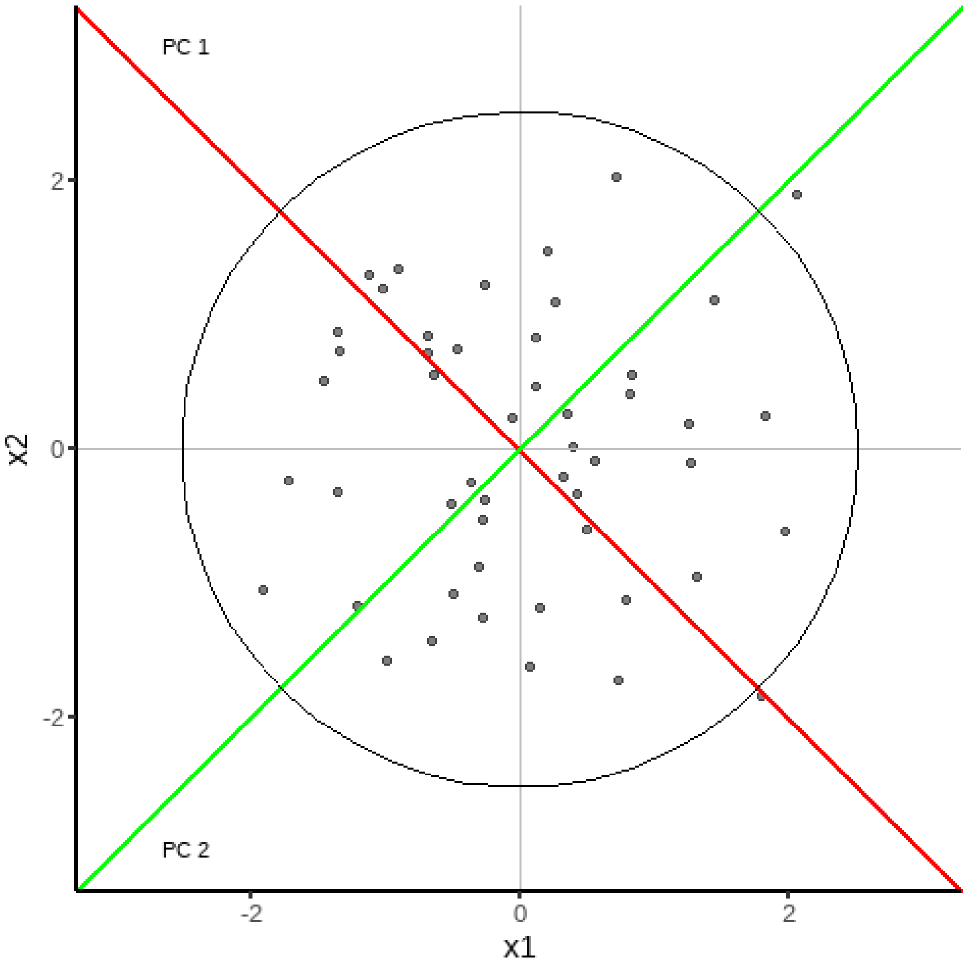
What PCA does do?

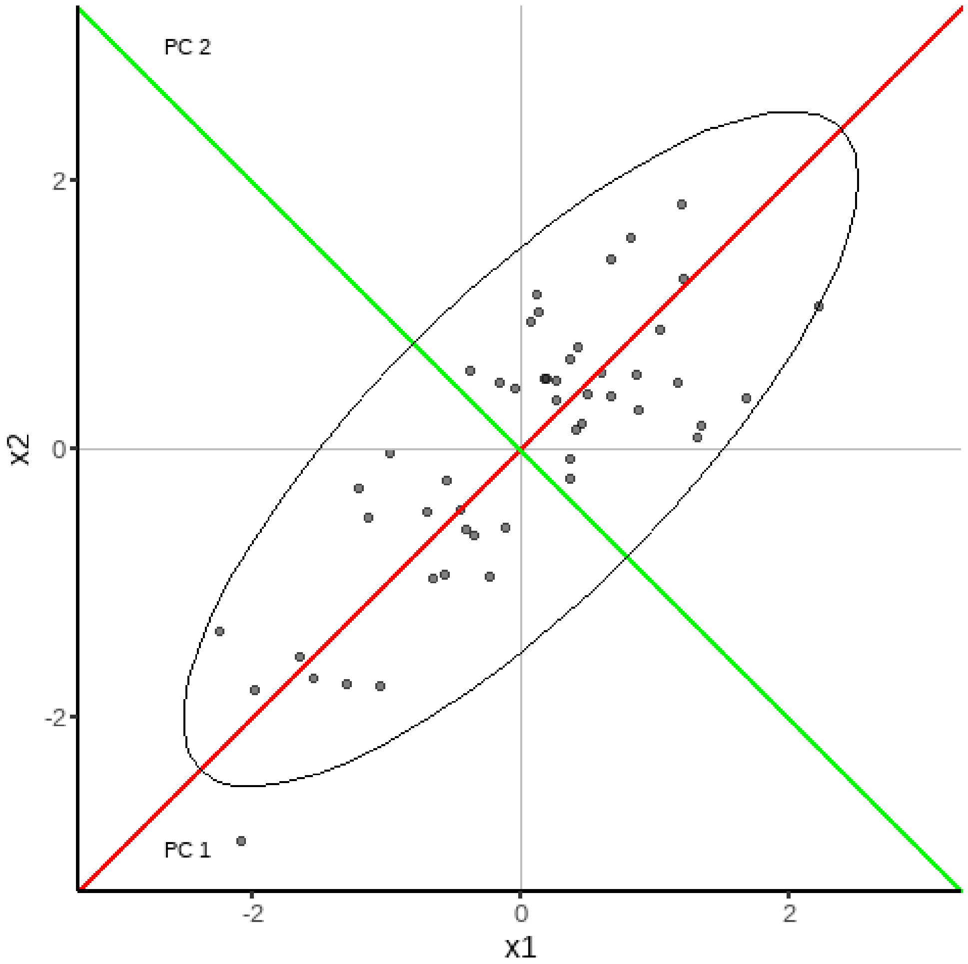
- Repackages the variance from the correlation matrix into a set of **components**
- Components = orthogonal (i.e.,uncorrelated) linear combinations of the original variables
  - 1st component is the linear combination that accounts for the most possible variance
  - 2nd accounts for second-largest after the variance accounted for by the first is removed
  - 3rd...etc...
- Each component accounts for as much remaining variance as possible

What PCA does do?

- If variables are very closely related (large correlations), then we can represent them by fewer composites.
- If variables are not very closely related (small correlations), then we will need more composites to adequately represent them.
- In the extreme, if variables are entirely uncorrelated, we will need as many components as there were variables in original correlation matrix.

Thinking about dimensions





Eigendecomposition

- Components are formed using an **eigen-decomposition** of the correlation matrix
- Eigen-decomposition is a transformation of the correlation matrix to re-express it in terms of **eigenvalues** and **eigenvectors**
- There is one eigenvector and one eigenvalue for each 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.
  - Visually (previous slide) eigenvalues are the length of the line
- Eigenvectors provide information on the relationship of each variable to each component.
  - Visually, eigenvectors provide the direction of the line.

End of Part 2

Part 1: Introduction to data reduction

Part 2: Purpose of PCA

Part 3: Eigenvalues & Eigenvectors

Part 4: Running & Interpreting PCA

Part 5: PCA scores

Eigenvalues and eigenvectors

```
## [1] "e1" "e2" "e3" "e4" "e5"
```

```
##       component1 component2 component3 component4 component5
## item1 "w11"      "w12"      "w13"      "w14"      "w15"
## item2 "w21"      "w22"      "w23"      "w24"      "w25"
## item3 "w31"      "w32"      "w33"      "w34"      "w35"
## item4 "w41"      "w42"      "w43"      "w44"      "w45"
## item5 "w51"      "w52"      "w53"      "w54"      "w55"
```

- Eigenvectors are sets of **weights** (one weight per variable in original correlation matrix)
  - e.g., if we had 5 variables each eigenvector would contain 5 weights
  - Larger weights mean a variable makes a bigger contribution to the component

Eigen-decomposition of aggression item correlation matrix

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

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

Eigen-decomposition of aggression item correlation matrix

- Eigenvalues:

```
##  [1] 3.789 2.797 0.578 0.538 0.533 0.471 0.407 0.351 0.326 0.210
```

- Eigenvectors

```
##         [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]   [,8]   [,9]  [,10]
##  [1,] -0.290  0.316  0.375  0.242 -0.435 -0.529  0.295 -0.242 -0.038 -0.029
##  [2,] -0.311  0.349  0.146  0.039 -0.092  0.177 -0.563  0.268 -0.572  0.054
##  [3,] -0.295  0.332  0.106 -0.326  0.162  0.556  0.544 -0.201 -0.116  0.007
##  [4,] -0.279  0.299 -0.707  0.164  0.405 -0.340  0.080 -0.067 -0.119 -0.040
##  [5,] -0.299  0.379  0.055 -0.046  0.019  0.086 -0.278  0.207  0.796 -0.026
##  [6,] -0.303 -0.297  0.125  0.731  0.169  0.254  0.249  0.332  0.033  0.070
##  [7,] -0.370 -0.310  0.004 -0.040  0.057  0.013 -0.252 -0.521  0.071  0.650
##  [8,] -0.365 -0.330  0.020 -0.007  0.013  0.087 -0.223 -0.369  0.002 -0.751
##  [9,] -0.320 -0.253 -0.463 -0.233 -0.658  0.077  0.164  0.310  0.004  0.054
## [10,] -0.318 -0.279  0.307 -0.464  0.384 -0.422  0.116  0.411 -0.064  0.011
```

Eigenvalues and variance

- It is important to understand some basic rules about eigenvalues and variance.

- The sum of the eigenvalues will equal the number of variables in the data set.

    - The covariance of an item with itself is 1 (think the diagonal in a correlation matrix)
    - Adding these up = total variance.
    - A full eigendecomposition accounts for all variance distributed across eigenvalues.
    - So in our example, the sum of the eigenvalues must = 10.

```
sum(eigen_res$values)
```

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

Eigenvalues and variance

- Given this, if we want to know the variance accounted for my a given component:

$$\frac{eigenvalue}{totalvariance}$$

- or

$$\frac{eigenvalue}{p}$$

- where p = number of items.

Eigenvalues and variance

```
(eigen_res$values/sum(eigen_res$values))*100
```

```
##  [1] 37.894 27.972  5.782  5.376  5.332  4.707  4.073  3.506  3.259  2.099
```

- and if we sum this

```
sum((eigen_res$values/sum(eigen_res$values))*100)
```

```
## [1] 100
```

Eigenvectors & PCA Loadings

- Whereas we use eigenvalues to think about variance, we use eigenvectors to think about the nature of components.
- To do so, we convert eigenvectors to PCA loadings.
    - A PCA loading gives the strength of the relationship between the item and the component.
    - Range from -1 to 1
    - The higher the absolute value, the stronger the relationship.
- The sum of the squared loadings for any variable on all components will equal 1.
    - That is all the variance in the item is explained by the full decomposition.

Eigenvectors & PCA Loadings

- We get the loadings by:

$$a_{ij}^{*} = a_{ij}\sqrt{\lambda_j}$$

- where
  - a_{ij}^{*} = the component loading for item i on component j
  - a_{ij} = the associated eigenvector value
  - λ_j is the eigenvalue for component j
- Essentially we are scaling the eigenvectors by the eigenvalues such that the components with the largest eigenvalues have the largest loadings.

End of Part 3

Part 1: Introduction to data reduction

Part 2: Purpose of PCA

Part 3: Eigenvalues & Eigenvectors

Part 4: Running & Interpreting PCA

Part 5: PCA scores

How many components to keep?

- Eigen-decomposition repackages the variance but does not reduce our dimensions
- Dimension reduction comes from keeping only the largest components
- Assume the others can be dropped with little loss of information
- Our decisions on how many components to keep can be guided by several methods
  - Set a amount of variance you wish to account for
  - Scree plot
  - Minimum average partial test (MAP)
  - Parallel analysis

Variance accounted for

- As has been noted, each component accounts for some proportion of the variance in our original data.
- The simplest method we can use to select a number of components is simply to state a minimum variance we wish to account for.
    - We then select the number of components above this value.

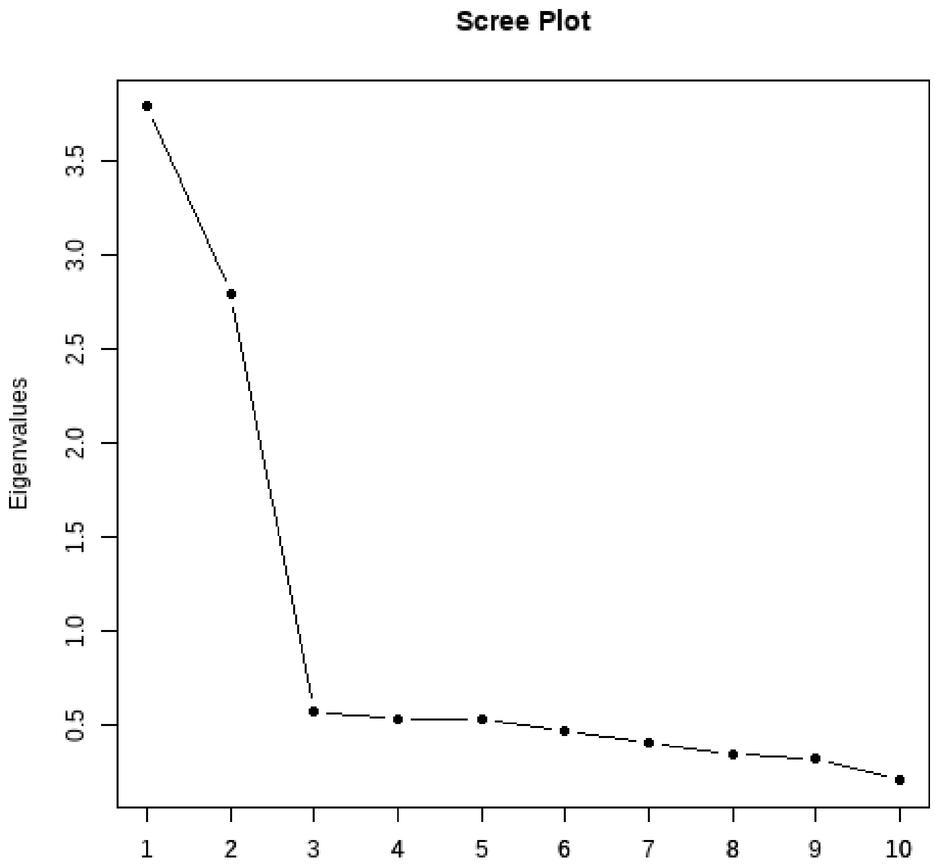
Scree plot

- Based on plotting the eigenvalues
  - Remember our eigenvalues are representing variance.
- Looking for a sudden change of slope
- Assumed to potentially reflect point at which components become substantively unimportant
  - As the slope flattens, each subsequent component is not explaining much additional variance.

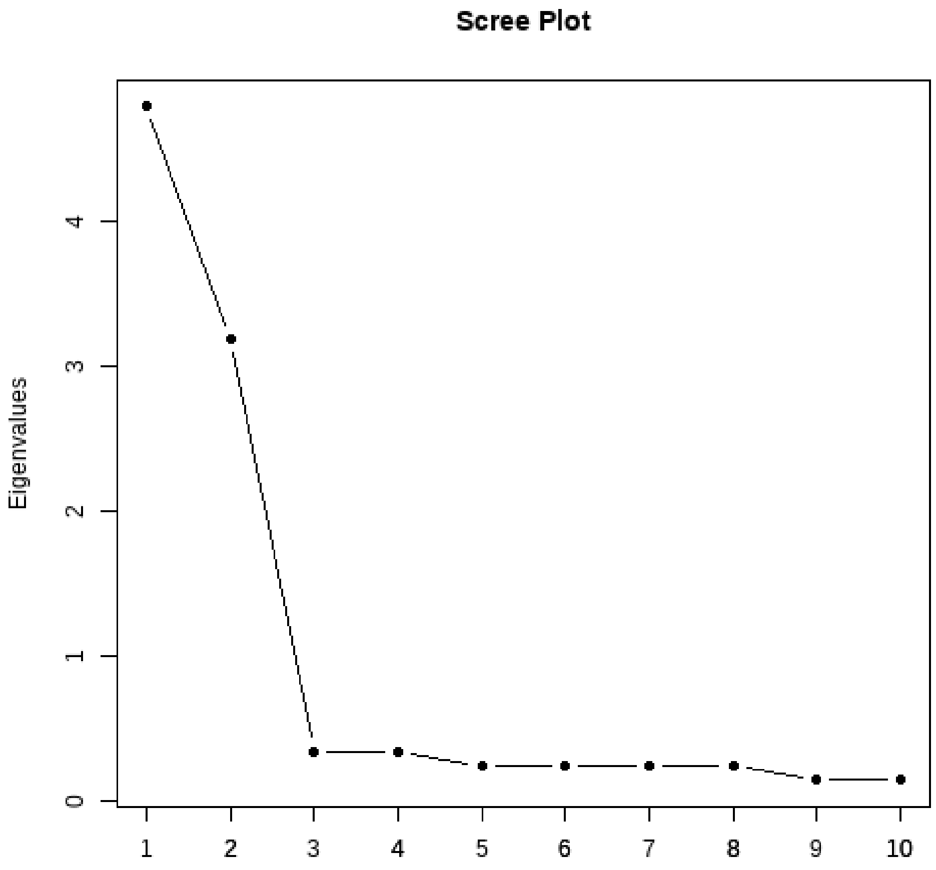
Constructing a scree plot

```
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)
```

- Eigenvalue plot
    - x-axis is component number
    - y-axis is eigenvalue for each component
- Keep the components with eigenvalues above a kink in the plot

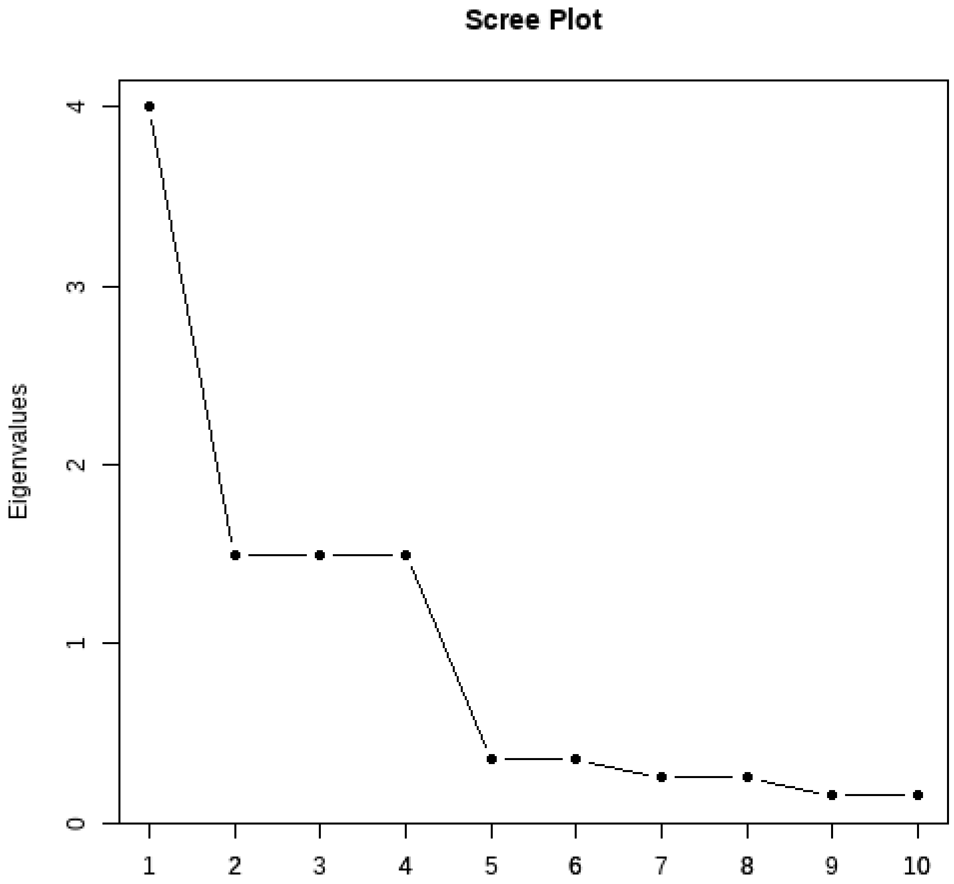


Further scree plot examples

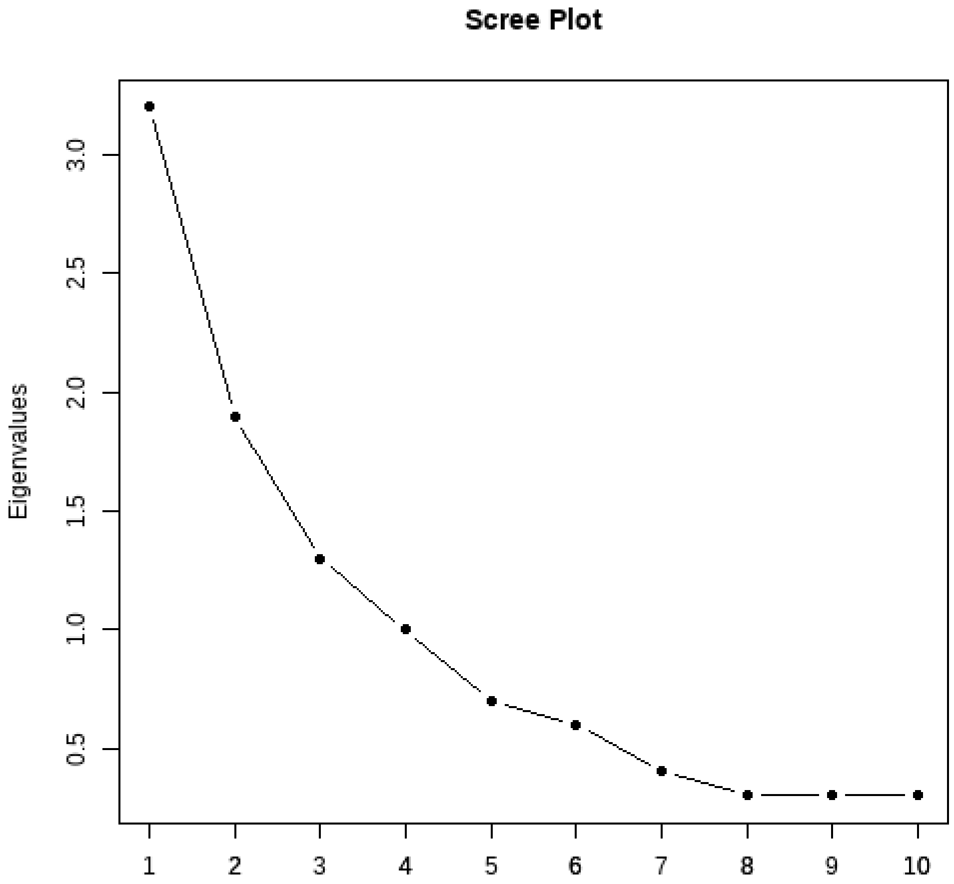


- Scree plots vary in how easy it is to interpret them

Further scree plot examples



Further scree plot examples



Minimum average partial test (MAP)

- Extracts components iteratively from the correlation matrix
- Computes the average squared partial correlation after each extraction
    - This is the MAP value.
- 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)
```

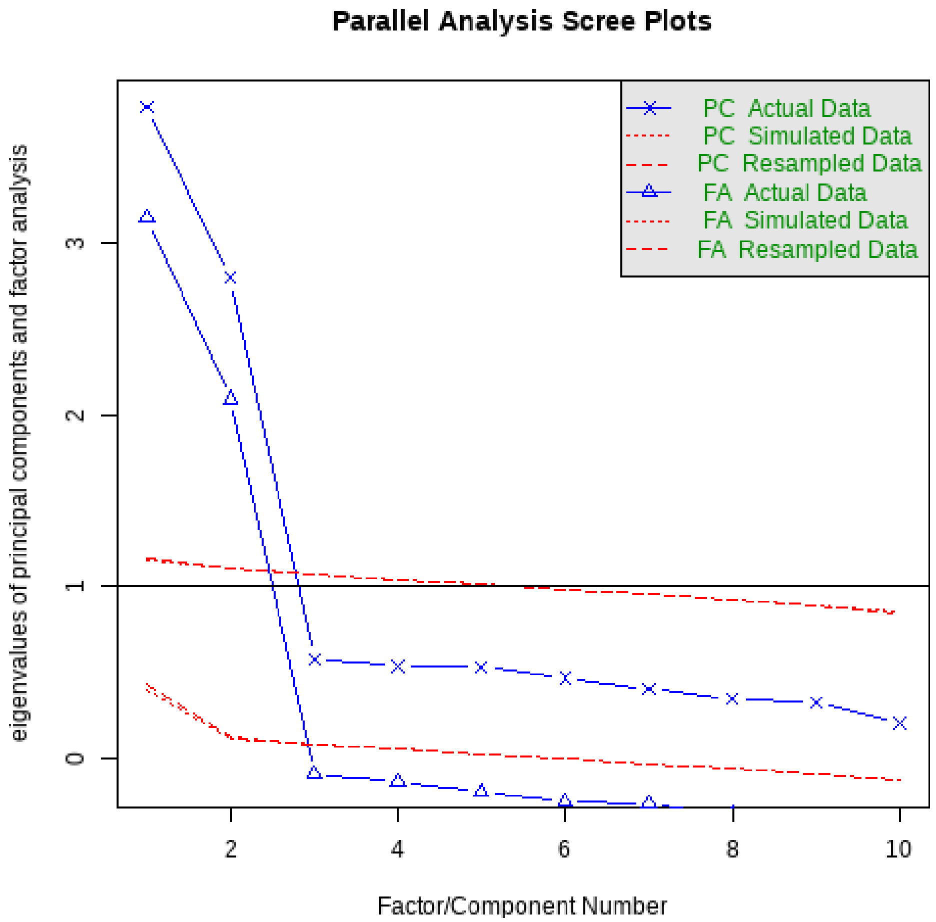
- In this output:
  - Plot that associates with a different test (Very Simple Structure - one for office hours)
  - A text output explaining what different methods indicate are the best number of componenets
  - And the numerical results for the tests.

Parallel analysis

- Simulates datasets with same number of participants and variables but no correlations
- Computes an eigen-decomposition 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 simulated eigenvalues
    - e.g. 95% percentile of the simulated eigenvalue distributions

Parallel analysis for the aggression items

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



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 (over-extraction)
  - MAP sometimes suggests too few components (under-extraction)
- Examining the PCA solutions should also form part of the decision
  - Do components make practical sense given purpose?
  - Do components make substantive sense?

Running a PCA with a reduced number of components

- We can run a PCA keeping just a selected number of components
- We do this using the `principal()` function from then psych package
- We supply the dataframe or correlation matrix as the first argument
- We specify the number of components to retain with the `nfactors=` argument
- It can be useful to compare and contrast the solutions with different numbers of components
  - Allows us to check which solutions make most sense based on substantive/practical considerations

```
PC2<-principal(agg.items, nfactors=2)
PC3<-principal(agg.items, nfactors=3)
```

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

  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

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

```
## 
## Loadings:
##        RC1   RC2  
## item1        0.771
## item2        0.838
## item3        0.797
## item4        0.736
## item5        0.860
## item6  0.770      
## item7  0.881 0.102
## item8  0.897      
## item9  0.746 0.108
## item10 0.772      
## 
##                  RC1   RC2
## SS loadings    3.339 3.247
## Proportion Var 0.334 0.325
## Cumulative Var 0.334 0.659
```

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 Components Analysis
## Call: principal(r = agg.items, nfactors = 2)
## Standardized loadings (pattern matrix) based upon correlation matrix
##         RC1  RC2   h2   u2 com
## item1  0.06 0.77 0.60 0.40   1
## item2  0.05 0.84 0.71 0.29   1
## item3  0.05 0.80 0.64 0.36   1
## item4  0.07 0.74 0.55 0.45   1
## item5  0.00 0.86 0.74 0.26   1
## item6  0.77 0.03 0.59 0.41   1
## item7  0.88 0.10 0.79 0.21   1
## item8  0.90 0.07 0.81 0.19   1
## item9  0.75 0.11 0.57 0.43   1
## item10 0.77 0.07 0.60 0.40   1
##
##                        RC1  RC2
## SS loadings           3.34 3.25
## Proportion Var        0.33 0.32
## Cumulative Var        0.33 0.66
## Proportion Explained  0.51 0.49
## Cumulative Proportion 0.51 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  322.6  with prob <  6.2e-53
##
```

End of Part 4

Part 1: Introduction to data reduction

Part 2: Purpose of PCA

Part 3: Eigenvalues & Eigenvectors

Part 4: Running & Interpreting PCA

Part 5: PCA scores

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 that are deemed to "belong" to a component.
    - This idea is usually on the size of the component loadings
    - A loading of >|.3| is typically used.
- Better method is to compute them taking into account the weights
    - i.e. based on the eigenvalues and vectors

Computing component scores in R

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

```
##              RC1      RC2
## [1,] -0.475241  0.60172
## [2,]  0.003577  0.05011
## [3,]  0.414221  1.33385
## [4,] -0.999129 -1.02670
## [5,] -0.663171 -0.84120
## [6,]  0.727095 -0.26428
```

Reporting a PCA

- Main principles: transparency and reproducibility
- Method
  - Methods used to decide on number of components
- Results
  - Scree test (& any other considerations in choice of number of components)
  - How many components were retained
  - The loading matrix for the chosen solution
  - Variance explained by components

End