Week 5, Lecture 10
Advanced statistical methods, part I: Ecological analyses, ordinal data, and dimensionality reduction

Richard E.W. Berl

Spring 2019

Dimensionality reduction

Let's load our Bob Marshall onsite survey data back in:

```
bm = read.csv("./data/BMWC2004_onsitedata.csv", header=T, na.strings="88",
              stringsAsFactors=F)
bm$recent_f[bm$recent_f == 0] = NA

bmLik = bm
bmLik[,36:45] = lapply(bmLik[,36:45], function(x) ordered(x))
```

And recreate our correlation matrix:

```
library(lavaan)
```

```
## This is lavaan 0.6-3
```

```
## lavaan is BETA software! Please report any bugs.
```

```
bmLikCor = lavCor(bmLik[,36:45])
```

Let's also load our previously saved "best" Christmas Bird Count data:

```
best = read.csv("./data/fcbirdbest.csv", header=T, row.names=1)
```

And finally, we'll also bring back our old Indo-European folktale data from Lecture 03:

```
library(readxl)
```

```
folktales = as.data.frame(read_xlsx(path="./data/rsos150645supp1.xlsx",
                                    sheet=1, range="A2:JP52"))
colnames(folktales)[1] = "society"
```

Multidimensional scaling

Primarily used as a way to visualize a distance matrix

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

```
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following object is masked from 'package:lavaan':
##
##     cor2cov
```

```
?cor2dist
```

```
bmLikDist = as.dist(cor2dist(bmLikCor))
bmLikDist
```

```
##            natural  remotnes  scenic_b   hunting   fishing  recent_f
## remotnes 0.5704977
## scenic_b 1.0515567 0.9863890
## hunting  1.4615125 1.4393572 1.5125929
## fishing  1.3603630 1.3329079 1.4377701 0.8566035
## recent_f 1.3198146 1.3881493 1.1623764 1.2596373 1.4117703
## test_ski 1.2981759 1.4014717 1.2798294 1.2405682 1.4147664 1.1854113
## familiar 1.3609819 1.3644849 1.2606143 1.2996073 1.3103081 1.2660455
## variety  1.3184390 1.2984087 1.2842120 1.4259006 1.4726524 1.4263578
## friend_s 1.3824247 1.3641166 1.2151582 1.4267610 1.4266808 1.3921892
##           test_ski  familiar   variety
## remotnes
## scenic_b
## hunting
## fishing
## recent_f
## test_ski
## familiar 1.2210226
## variety  1.2355076 1.5686837
## friend_s 1.3660988 1.5218809 1.1418339
```

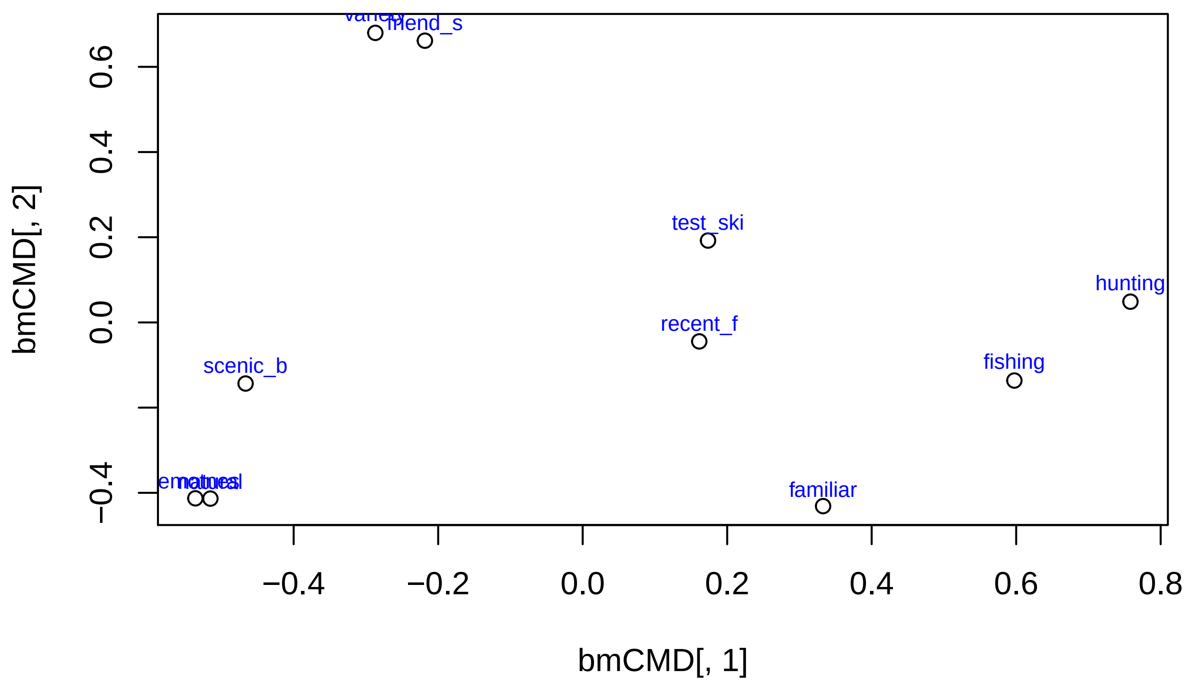
Classical

```
?cmdscale
```

```
bmCMD = cmdscale(bmLikDist)
bmCMD
```

```
##                [,1]        [,2]
## natural  -0.5152544 -0.41355292
## remotnes -0.5360831 -0.41294618
## scenic_b -0.4665433 -0.14344422
## hunting   0.7581483  0.04870982
## fishing   0.5974677 -0.13641546
## recent_f  0.1613884 -0.04454814
## test_ski  0.1734981  0.19224084
## familiar  0.3327288 -0.43118845
## variety  -0.2869680  0.67972386
## friend_s -0.2183826  0.66142085
```

```
plot(bmCMD[,1], bmCMD[,2])
text(bmCMD[,1], bmCMD[,2] + 0.04,
     labels=rownames(bmCMD), col="blue", cex=0.7)
```



Nonmetric

Tries to reproduce ranks of distances rather than distance values themselves

```
library(MASS)
```

```
?isoMDS
```

```
bmNMD = isoMDS(bmLikDist)
```

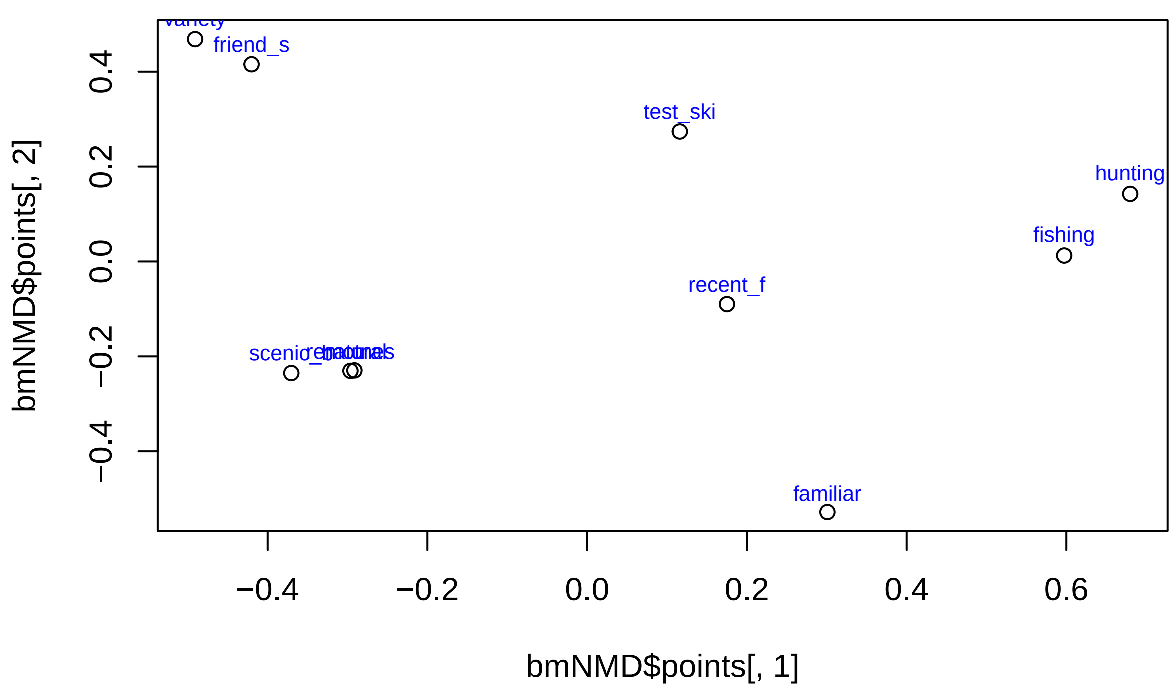
```
## initial  value 19.805863
## iter   5 value 14.937231
## iter  10 value 14.361126
## iter  15 value 13.937601
## final  value 13.767700
## converged
```

```
bmNMD
```

```
## $points
##                [,1]        [,2]
## natural  -0.2914076 -0.22953015
## remotnes -0.2963235 -0.23046987
## scenic_b -0.3703879 -0.23513955
## hunting   0.6799271  0.14245565
## fishing   0.5972598  0.01257943
## recent_f  0.1750672 -0.08985301
## test_ski  0.1160045  0.27390459
```

```
## familiar  0.3008428 -0.52798807
## variety  -0.4908231  0.46848012
## friend_s -0.4201593  0.41556086
##
## $stress
## [1] 13.7677
```

```r
plot(bmNMD$points[,1], bmNMD$points[,2])
text(bmNMD$points[,1], bmNMD$points[,2] + 0.04,
     labels=rownames(bmNMD$points), col="blue", cex=0.7)
```




```r
library(vegan)
```

```
## Loading required package: permute
```

```
## Loading required package: lattice
```

```
## This is vegan 2.5-4
```

```r
?metaMDS
```

```r
head(best)
```

```
##      American.Crow American.Dipper American.Goldfinch American.Kestrel
## 1952           353              12                 16                2
## 1956             5               2                 36                2
## 1957             3               3                  7                3
## 1958           168               8                  3                7
## 1960             2               5                  3                6
## 1962           590              20                  6                2
```

```
bestNMD = metaMDS(best)
```

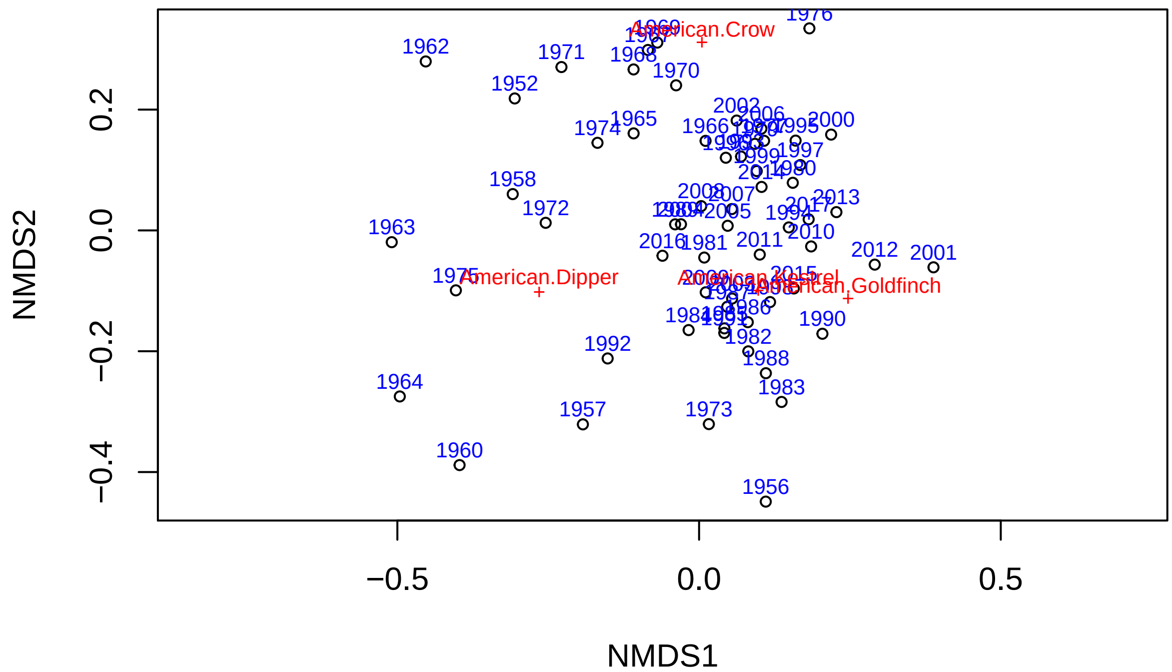
```
## Square root transformation
## Wisconsin double standardization
## Run 0 stress 0.08931323
## Run 1 stress 0.08928892
## ... New best solution
## ... Procrustes: rmse 0.001500627  max resid 0.008328678
## ... Similar to previous best
## Run 2 stress 0.08928881
## ... New best solution
## ... Procrustes: rmse 9.122812e-05  max resid 0.0006093033
## ... Similar to previous best
## Run 3 stress 0.08928881
## ... New best solution
## ... Procrustes: rmse 2.142381e-06  max resid 1.006372e-05
## ... Similar to previous best
## Run 4 stress 0.08931322
## ... Procrustes: rmse 0.001494668  max resid 0.00833698
## ... Similar to previous best
## Run 5 stress 0.08928881
## ... New best solution
## ... Procrustes: rmse 1.210555e-05  max resid 7.555205e-05
## ... Similar to previous best
## Run 6 stress 0.08928889
## ... Procrustes: rmse 5.33297e-05  max resid 0.0003422892
## ... Similar to previous best
## Run 7 stress 0.08928885
## ... Procrustes: rmse 3.369827e-05  max resid 0.0002128477
## ... Similar to previous best
## Run 8 stress 0.08931325
## ... Procrustes: rmse 0.001499033  max resid 0.008325851
## ... Similar to previous best
## Run 9 stress 0.08928881
## ... Procrustes: rmse 7.93628e-06  max resid 3.297777e-05
## ... Similar to previous best
## Run 10 stress 0.08928892
## ... Procrustes: rmse 6.219826e-05  max resid 0.0003919981
## ... Similar to previous best
## Run 11 stress 0.08928899
## ... Procrustes: rmse 7.604608e-05  max resid 0.0004769528
## ... Similar to previous best
## Run 12 stress 0.08928881
## ... New best solution
## ... Procrustes: rmse 7.634703e-06  max resid 4.314968e-05
## ... Similar to previous best
## Run 13 stress 0.08931326
## ... Procrustes: rmse 0.001492973  max resid 0.008342409
## ... Similar to previous best
## Run 14 stress 0.08928887
## ... Procrustes: rmse 2.457438e-05  max resid 0.0001375928
## ... Similar to previous best
## Run 15 stress 0.08928881
## ... Procrustes: rmse 7.501627e-06  max resid 4.994405e-05
```

```
## ... Similar to previous best
## Run 16 stress 0.2111758
## Run 17 stress 0.08931322
## ... Procrustes: rmse 0.00149561  max resid 0.008340165
## ... Similar to previous best
## Run 18 stress 0.4042937
## Run 19 stress 0.08931327
## ... Procrustes: rmse 0.001498969  max resid 0.008327518
## ... Similar to previous best
## Run 20 stress 0.08928955
## ... Procrustes: rmse 0.0002181771  max resid 0.001463275
## ... Similar to previous best
## *** Solution reached
```

```
bestNMD
```

```
##
## Call:
## metaMDS(comm = best)
##
## global Multidimensional Scaling using monoMDS
##
## Data:     wisconsin(sqrt(best))
## Distance: bray
##
## Dimensions: 2
## Stress:     0.08928881
## Stress type 1, weak ties
## Two convergent solutions found after 20 tries
## Scaling: centring, PC rotation, halfchange scaling
## Species: expanded scores based on 'wisconsin(sqrt(best))'
```

```
plot(bestNMD)
text(bestNMD$points[,1], bestNMD$points[,2] + 0.025,
     labels=rownames(bestNMD$points), col="blue", cex=0.7)
text(bestNMD$species[,1], bestNMD$species[,2] + 0.025,
     labels=rownames(bestNMD$species), col="red", cex=0.7)
```



Principal components analysis

```r
?princomp
```

```r
?psych::principal
```

```r
?psych::tetrachoric
```

```r
folk = as.data.frame(t(folktales[,-1]))
colnames(folk) = folktales$society
folk[1:5,1:10]
```

```
##      Italian Ladin Sardinian Walloon French Spanish Portuguese Catalan
## 300        1     1         1       1      1       1          1       1
## 300A       0     0         0       0      0       0          0       0
## 301        1     1         1       1      1       1          1       1
## 301D       0     0         0       0      0       0          0       0
## 302        1     1         1       0      1       1          1       1
##      Romanian Welsh
## 300         1     0
## 300A        1     0
## 301         1     0
## 301D        0     0
## 302         1     0
```

```r
str(folk)
```

```
## 'data.frame':    275 obs. of  50 variables:
```

```
## $ Italian      : num  1 0 1 0 1 0 0 1 0 0 ...
## $ Ladin        : num  1 0 1 0 1 0 0 1 0 1 ...
## $ Sardinian    : num  1 0 1 0 1 0 0 1 0 0 ...
## $ Walloon      : num  1 0 1 0 0 0 0 1 0 0 ...
## $ French       : num  1 0 1 0 1 0 0 1 1 1 ...
## $ Spanish      : num  1 0 1 0 1 0 0 1 0 1 ...
## $ Portuguese   : num  1 0 1 0 1 0 0 1 0 1 ...
## $ Catalan      : num  1 0 1 0 1 0 0 1 0 0 ...
## $ Romanian     : num  1 1 1 0 1 0 1 1 1 1 ...
## $ Welsh        : num  0 0 0 0 0 0 0 0 0 0 ...
## $ Irish        : num  1 0 1 0 1 0 0 1 0 1 ...
## $ Scottish     : num  1 0 1 0 1 0 0 1 0 0 ...
## $ Luxembourgish: num  0 0 0 0 0 0 0 0 0 0 ...
## $ German       : num  1 1 1 0 1 0 1 1 1 1 ...
## $ Austrian     : num  1 0 1 0 1 0 0 1 0 1 ...
## $ Flemish      : num  1 0 1 0 1 0 0 1 0 1 ...
## $ Dutch        : num  1 0 1 0 0 0 0 1 0 0 ...
## $ Frisian      : num  1 0 1 0 1 0 0 1 0 1 ...
## $ English      : num  1 0 1 0 0 0 0 0 0 0 ...
## $ Swedish      : num  1 0 1 0 1 1 0 1 0 1 ...
## $ Norwegian    : num  1 0 1 0 1 0 0 1 1 1 ...
## $ Danish       : num  1 0 1 0 1 0 0 1 1 1 ...
## $ Faroese      : num  1 0 1 0 0 0 0 1 0 0 ...
## $ Icelandic    : num  0 0 1 0 1 0 0 0 0 0 ...
## $ Czech        : num  1 1 1 0 1 0 1 1 0 1 ...
## $ Slovak       : num  1 1 1 0 1 0 1 1 1 1 ...
## $ Lusatian     : num  1 0 1 0 1 0 0 0 0 0 ...
## $ Polish       : num  1 1 1 0 1 0 1 1 0 1 ...
## $ Byelorussian : num  1 1 1 1 1 0 1 1 1 0 ...
## $ Ukrainian    : num  1 1 1 1 1 0 0 1 0 1 ...
## $ Russian      : num  1 1 1 1 1 0 1 1 0 1 ...
## $ Bulgarian    : num  1 0 1 1 1 1 0 1 0 1 ...
## $ Macedonian   : num  0 0 0 0 0 1 0 0 0 0 ...
## $ Serbian      : num  1 0 1 0 1 0 0 0 1 1 ...
## $ Croation     : num  0 0 1 0 1 0 0 0 1 0 ...
## $ Slovenenian  : num  1 0 1 0 1 0 0 1 1 0 ...
## $ Latvian      : num  1 1 1 0 1 0 0 1 0 0 ...
## $ Lithuanian   : num  1 1 1 1 1 1 1 1 0 1 ...
## $ Pakistani    : num  1 0 0 0 1 0 0 0 0 0 ...
## $ Indian       : num  1 0 1 0 1 0 0 1 0 0 ...
## $ Nepali       : num  0 0 0 0 0 0 0 0 0 0 ...
## $ Gypsy        : num  1 1 1 0 1 0 1 1 1 1 ...
## $ Tadzhik      : num  0 0 0 0 0 0 0 1 1 0 ...
## $ Iranian      : num  0 0 1 0 0 0 0 1 0 0 ...
## $ Kurdish      : num  0 0 0 0 0 1 0 0 1 0 ...
## $ Afghan       : num  1 0 1 0 0 0 0 0 0 0 ...
## $ Ossetian     : num  1 1 1 0 1 0 1 1 1 1 ...
## $ Albanian     : num  0 0 0 0 0 0 0 0 0 1 ...
## $ Greek        : num  1 0 1 0 1 1 0 1 1 1 ...
## $ Armenian     : num  0 0 1 1 1 1 0 1 0 1 ...
```

Tetrachoric correlation

Binary data

```
folkCor = tetrachoric(folk)$rho

## For i = 10 j = 5  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 10 j = 6  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 10 j = 7  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 10 j = 8  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 11 j = 10  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 13 j = 1  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 14 j = 4  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 14 j = 10  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 16 j = 10  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 18 j = 10  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 22 j = 10  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 26 j = 10  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 28 j = 10  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 30 j = 10  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 31 j = 10  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 33 j = 10  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 33 j = 13  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 37 j = 4  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 37 j = 10  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 38 j = 10  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 39 j = 10  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 39 j = 13  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 41 j = 10  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 41 j = 12  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 41 j = 13  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 42 j = 10  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 43 j = 10  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 43 j = 13  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 45 j = 10  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 45 j = 13  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 46 j = 10  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 46 j = 13  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 47 j = 10  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 48 j = 13  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
## For i = 49 j = 10  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
```

```
## For i = 50 j = 13  A cell entry of 0 was replaced with correct =  0.5.  Check your data!
```

```
## Warning in cor.smooth(mat): Matrix was not positive definite, smoothing was
## done
```

```
folkCor[1:10,1:10]
```

```
##              Italian     Ladin Sardinian   Walloon    French   Spanish
## Italian    1.0000000 0.6143107 0.7672081 0.5822875 0.7623941 0.7019760
## Ladin      0.6143107 1.0000000 0.5082589 0.4529498 0.6912242 0.5869226
## Sardinian  0.7672081 0.5082589 1.0000000 0.4697491 0.6411595 0.5570139
## Walloon    0.5822875 0.4529498 0.4697491 1.0000000 0.5475553 0.3977685
## French     0.7623941 0.6912242 0.6411595 0.5475553 1.0000000 0.7574446
## Spanish    0.7019760 0.5869226 0.5570139 0.3977685 0.7574446 1.0000000
## Portuguese 0.6514227 0.4780249 0.5945318 0.3558462 0.6855161 0.7875559
## Catalan    0.7661892 0.5715129 0.5199610 0.5004013 0.7938338 0.8457141
## Romanian   0.4675313 0.5093527 0.5261116 0.3636709 0.5062839 0.4811890
## Welsh      0.3112035 0.3171770 0.1855564 0.2100189 0.4494361 0.4542234
##            Portuguese   Catalan  Romanian     Welsh
## Italian     0.6514227 0.7661892 0.4675313 0.3112035
## Ladin       0.4780249 0.5715129 0.5093527 0.3171770
## Sardinian   0.5945318 0.5199610 0.5261116 0.1855564
## Walloon     0.3558462 0.5004013 0.3636709 0.2100189
## French      0.6855161 0.7938338 0.5062839 0.4494361
## Spanish     0.7875559 0.8457141 0.4811890 0.4542234
## Portuguese  1.0000000 0.6911430 0.4440923 0.4418054
## Catalan     0.6911430 1.0000000 0.4819166 0.4794101
## Romanian    0.4440923 0.4819166 1.0000000 0.3790717
## Welsh       0.4418054 0.4794101 0.3790717 1.0000000
```

Note warnings about bivariate cell corrections. Okay to leave as 0.5. Source:

Savalei, V. (2011). What to do about zero frequency cells when estimating polychoric correlations. Structural Equation Modeling, 18(2), 253-273. doi: 10.1080/10705511.2011.557339

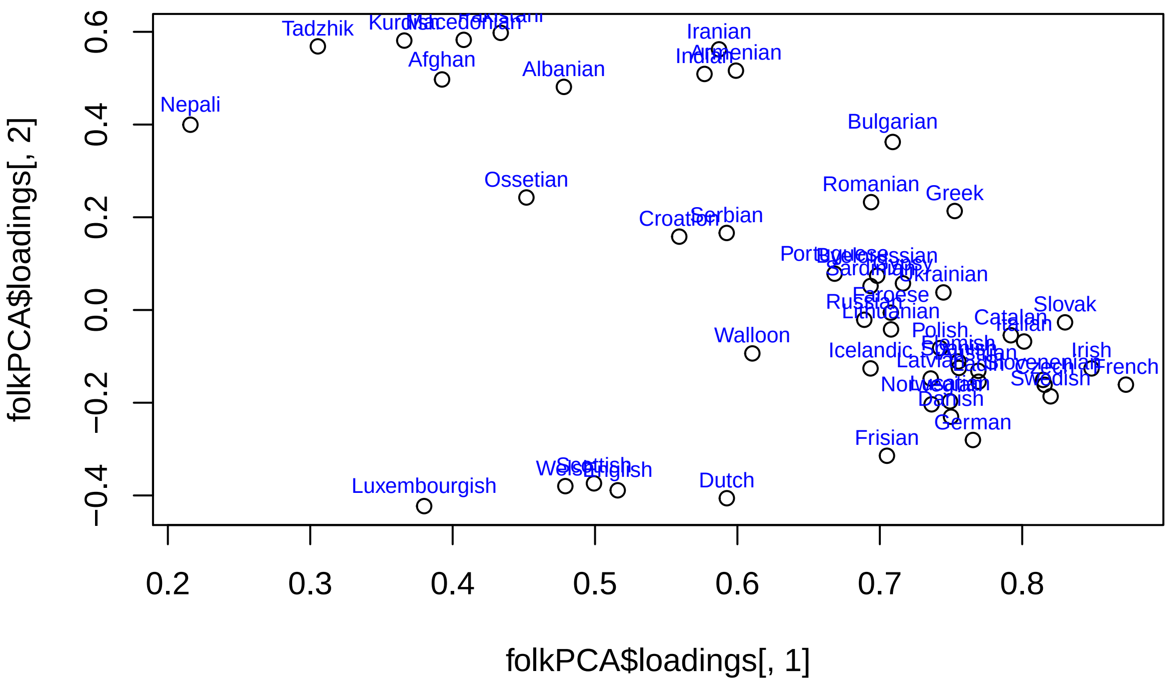
Smoothing ensures the matrix isn't singular and is usually also okay.

```
folkPCA = principal(folkCor, 2, rotate="none")
folkPCA
```

```
## Principal Components Analysis
## Call: principal(r = folkCor, nfactors = 2, rotate = "none")
## Standardized loadings (pattern matrix) based upon correlation matrix
##                PC1   PC2   h2   u2 com
## Italian       0.80 -0.07 0.65 0.35 1.0
## Ladin         0.77 -0.15 0.62 0.38 1.1
## Sardinian     0.69  0.05 0.48 0.52 1.0
## Walloon       0.61 -0.09 0.38 0.62 1.0
## French        0.87 -0.16 0.79 0.21 1.1
## Spanish       0.76 -0.13 0.59 0.41 1.1
## Portuguese    0.67  0.08 0.45 0.55 1.0
## Catalan       0.79 -0.05 0.63 0.37 1.0
## Romanian      0.69  0.23 0.54 0.46 1.2
## Welsh         0.48 -0.38 0.37 0.63 1.9
## Irish         0.85 -0.13 0.74 0.26 1.0
## Scottish      0.50 -0.37 0.39 0.61 1.9
## Luxembourgish 0.38 -0.42 0.32 0.68 2.0
## German        0.77 -0.28 0.66 0.34 1.3
```

```
## Austrian      0.77 -0.13 0.61 0.39 1.1
## Flemish       0.76 -0.11 0.58 0.42 1.0
## Dutch         0.59 -0.41 0.52 0.48 1.8
## Frisian       0.71 -0.31 0.60 0.40 1.4
## English       0.52 -0.39 0.42 0.58 1.9
## Swedish       0.82 -0.19 0.71 0.29 1.1
## Norwegian     0.74 -0.20 0.58 0.42 1.2
## Danish        0.75 -0.23 0.62 0.38 1.2
## Faroese       0.71 -0.01 0.50 0.50 1.0
## Icelandic     0.69 -0.13 0.50 0.50 1.1
## Czech         0.82 -0.16 0.69 0.31 1.1
## Slovak        0.83 -0.03 0.69 0.31 1.0
## Lusatian      0.75 -0.20 0.60 0.40 1.1
## Polish        0.74 -0.08 0.56 0.44 1.0
## Byelorussian  0.70  0.07 0.49 0.51 1.0
## Ukrainian     0.74  0.04 0.56 0.44 1.0
## Russian       0.69 -0.02 0.48 0.52 1.0
## Bulgarian     0.71  0.36 0.63 0.37 1.5
## Macedonian    0.41  0.58 0.51 0.49 1.8
## Serbian       0.59  0.17 0.38 0.62 1.2
## Croation      0.56  0.16 0.34 0.66 1.2
## Slovenenian   0.81 -0.15 0.69 0.31 1.1
## Latvian       0.74 -0.15 0.56 0.44 1.1
## Lithuanian    0.71 -0.04 0.50 0.50 1.0
## Pakistani     0.43  0.60 0.55 0.45 1.8
## Indian        0.58  0.51 0.59 0.41 2.0
## Nepali        0.22  0.40 0.21 0.79 1.5
## Gypsy         0.72  0.06 0.52 0.48 1.0
## Tadzhik       0.31  0.57 0.42 0.58 1.5
## Iranian       0.59  0.56 0.66 0.34 2.0
## Kurdish       0.37  0.58 0.47 0.53 1.7
## Afghan        0.39  0.50 0.40 0.60 1.9
## Ossetian      0.45  0.24 0.26 0.74 1.5
## Albanian      0.48  0.48 0.46 0.54 2.0
## Greek         0.75  0.21 0.61 0.39 1.2
## Armenian      0.60  0.52 0.63 0.37 2.0
##
##                         PC1  PC2
## SS loadings           22.13 4.55
## Proportion Var         0.44 0.09
## Cumulative Var         0.44 0.53
## Proportion Explained   0.83 0.17
## Cumulative Proportion  0.83 1.00
##
## Mean item complexity =  1.3
## Test of the hypothesis that 2 components are sufficient.
##
## The root mean square of the residuals (RMSR) is  0.08
##
## Fit based upon off diagonal values = 0.97
```

```
plot(folkPCA$loadings[,1], folkPCA$loadings[,2])
text(folkPCA$loadings[,1], folkPCA$loadings[,2] + 0.04,
     labels=rownames(folkPCA$loadings), col="blue", cex=0.7)
```



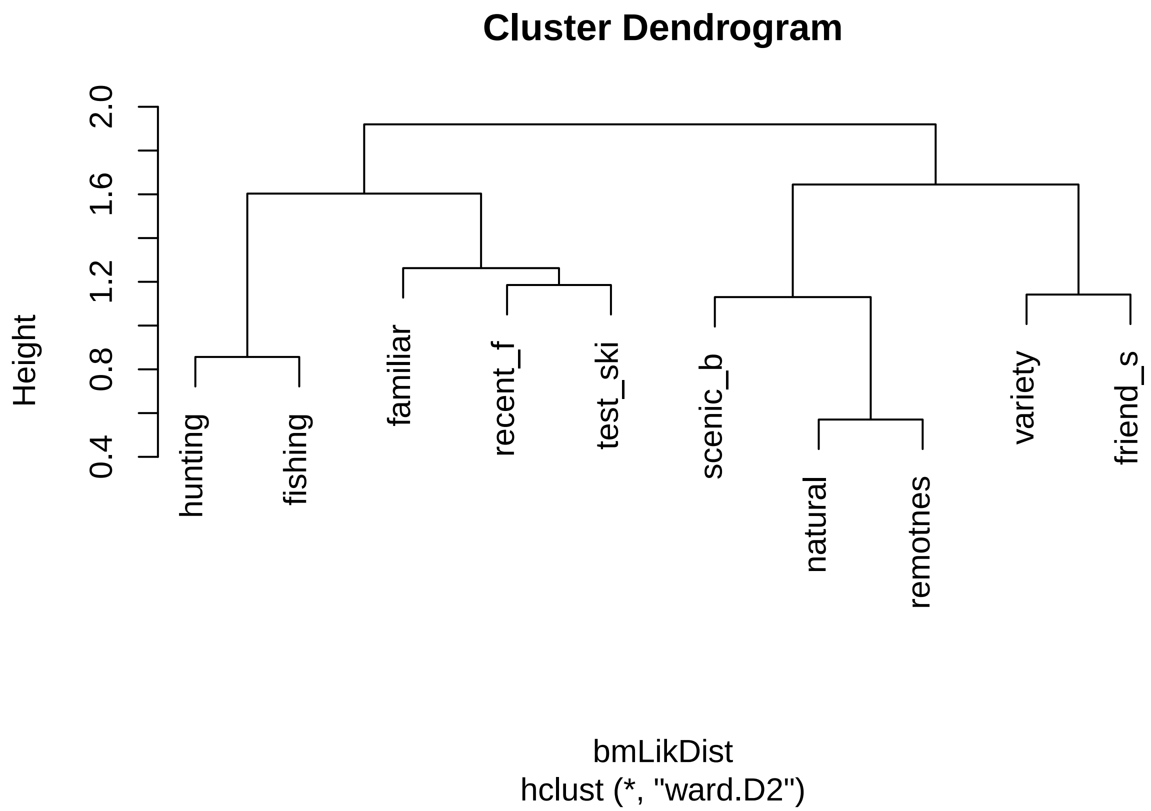
Cluster analysis

Hierarchical clustering

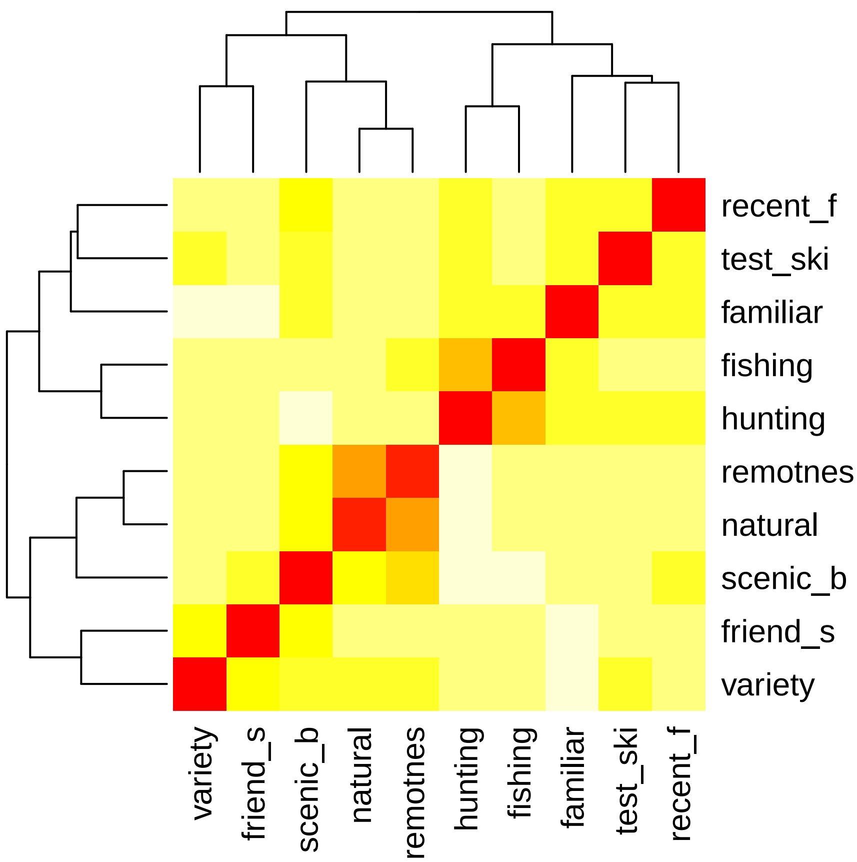
```
?hclust
```

```
bmHC = hclust(bmLikDist, method="ward.D2")
```

```
plot(bmHC)
```




```
heatmap(as.matrix(bmLikDist),
        hclustfun=function(x) hclust(x, method="ward.D2"))
```




```r
install.packages("pvclust")
```

```r
library(pvclust)
```

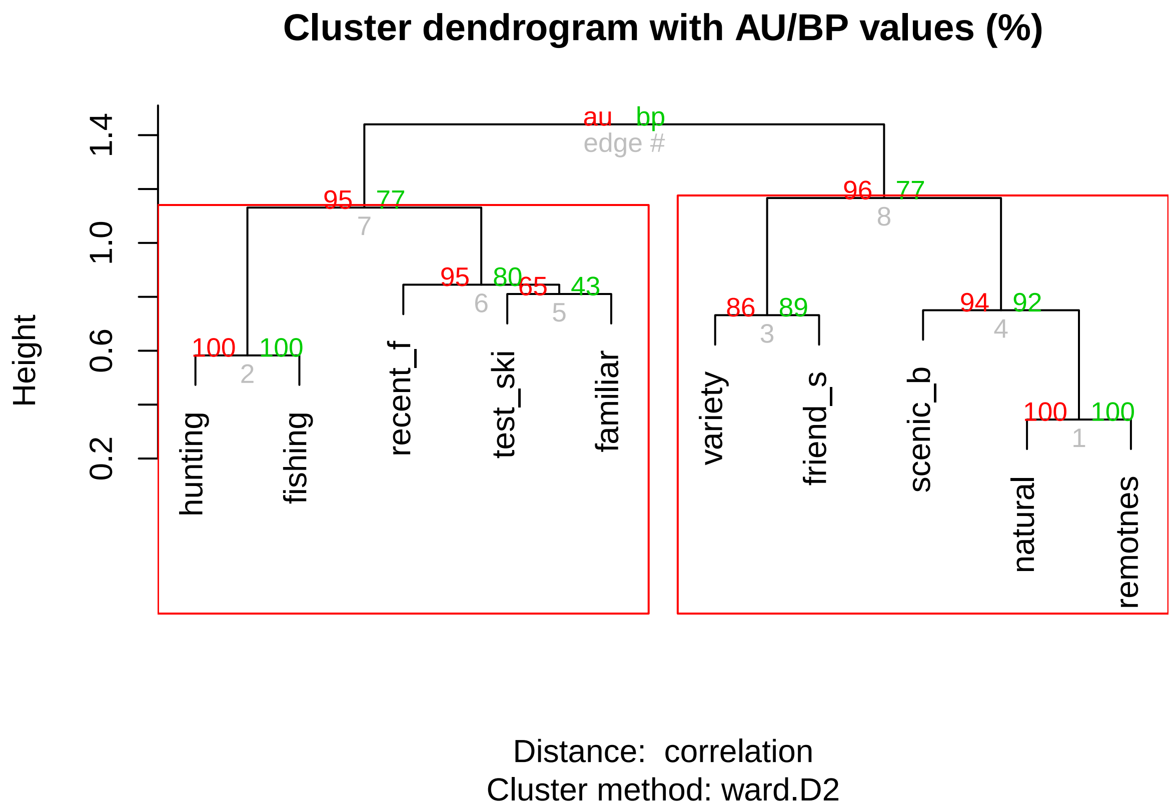
```r
?pvclust
```

Needs raw numeric data; does not allow distance matrix as input

```r
# Note: Takes some time to run
bmPVHC = pvclust(bm[,36:45], method.hclust="ward.D2")
```

```
## Bootstrap (r = 0.5)... Done.
## Bootstrap (r = 0.6)... Done.
## Bootstrap (r = 0.7)... Done.
## Bootstrap (r = 0.8)... Done.
## Bootstrap (r = 0.9)... Done.
## Bootstrap (r = 1.0)... Done.
## Bootstrap (r = 1.1)... Done.
## Bootstrap (r = 1.2)... Done.
## Bootstrap (r = 1.3)... Done.
## Bootstrap (r = 1.4)... Done.
```

```r
plot(bmPVHC)
pvrect(bmPVHC)
```



Red = "AU" (Approximately Unbiased) _p_value: 1 - *p*-value (>95 is "significant")

Green = "BP" (Bootstrap Probability): percent of times the tree-building algorithm produced that branch

K-means clustering

```
?kmeans
```

Does not work with missing values

"Elbow method"

```
install.packages("factoextra")
```

```
library(factoextra)
```

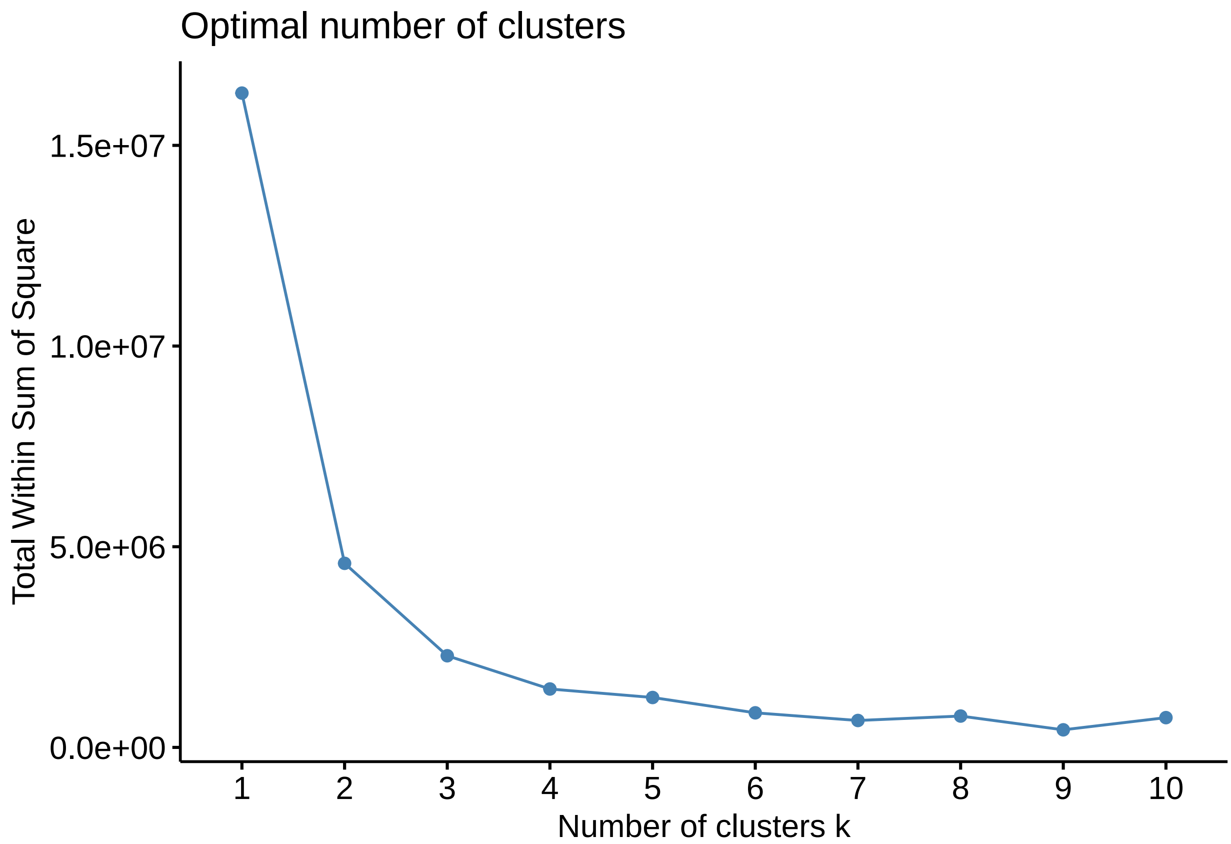
```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

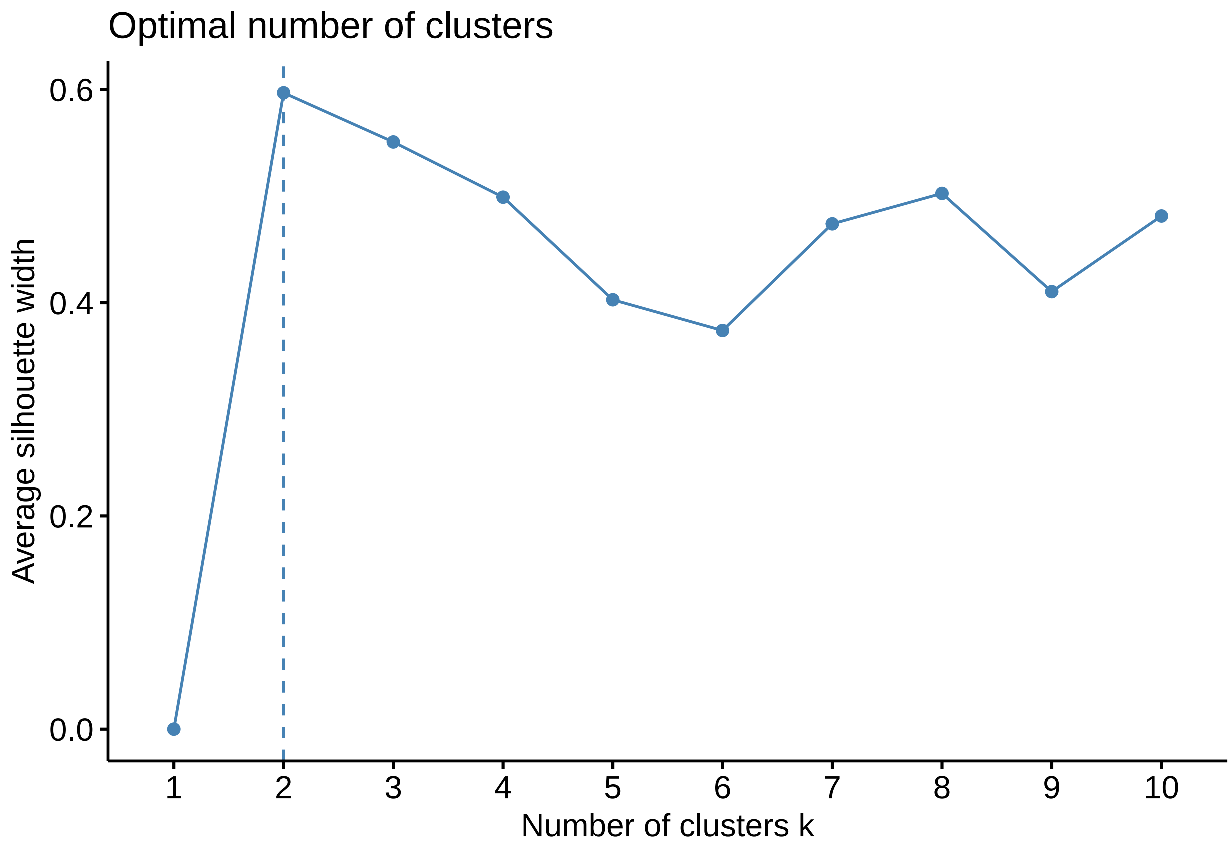
```
## The following objects are masked from 'package:psych':
##
##     %+%, alpha
```

```
## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ
```

```
fviz_nbclust(best, kmeans, "wss")
```




```
fviz_nbclust(best, kmeans, "silhouette")
```

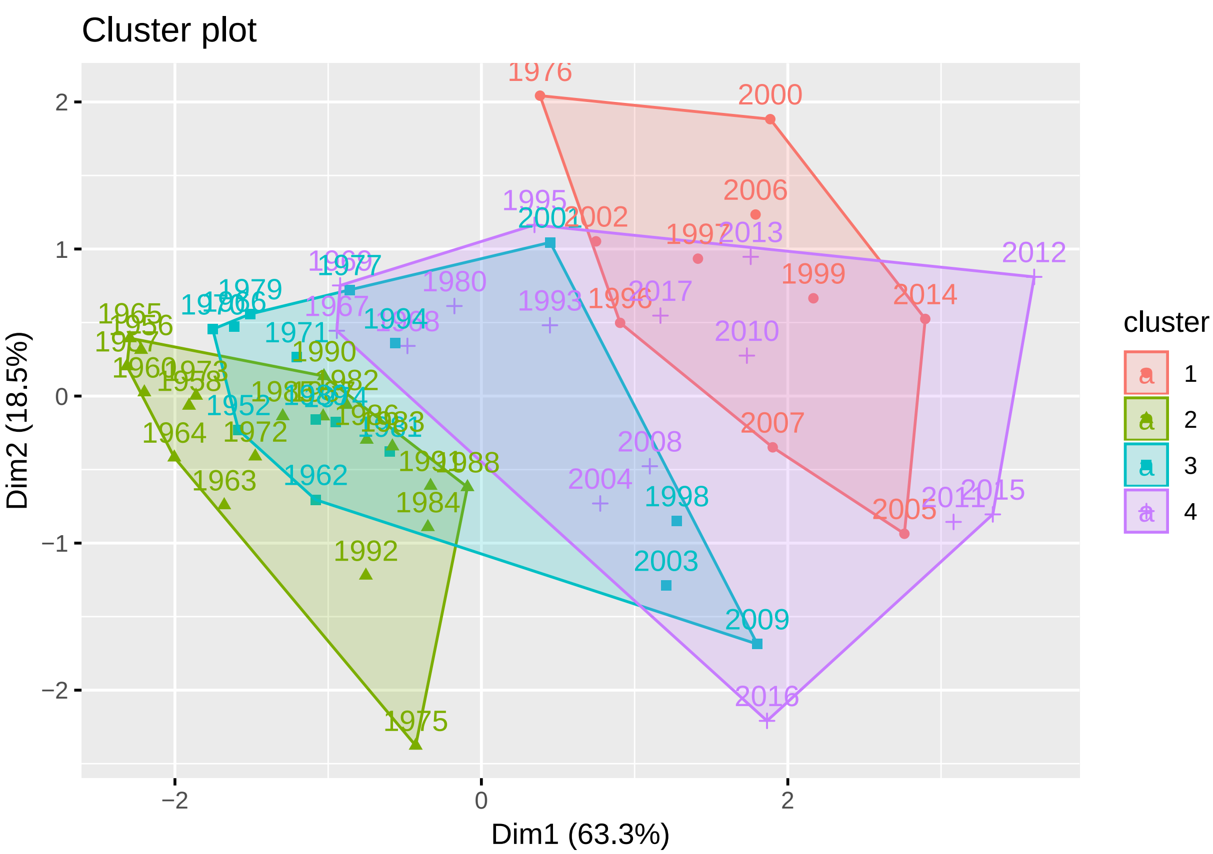



```
bestKM4 = kmeans(best, 4)
bestKM4
```

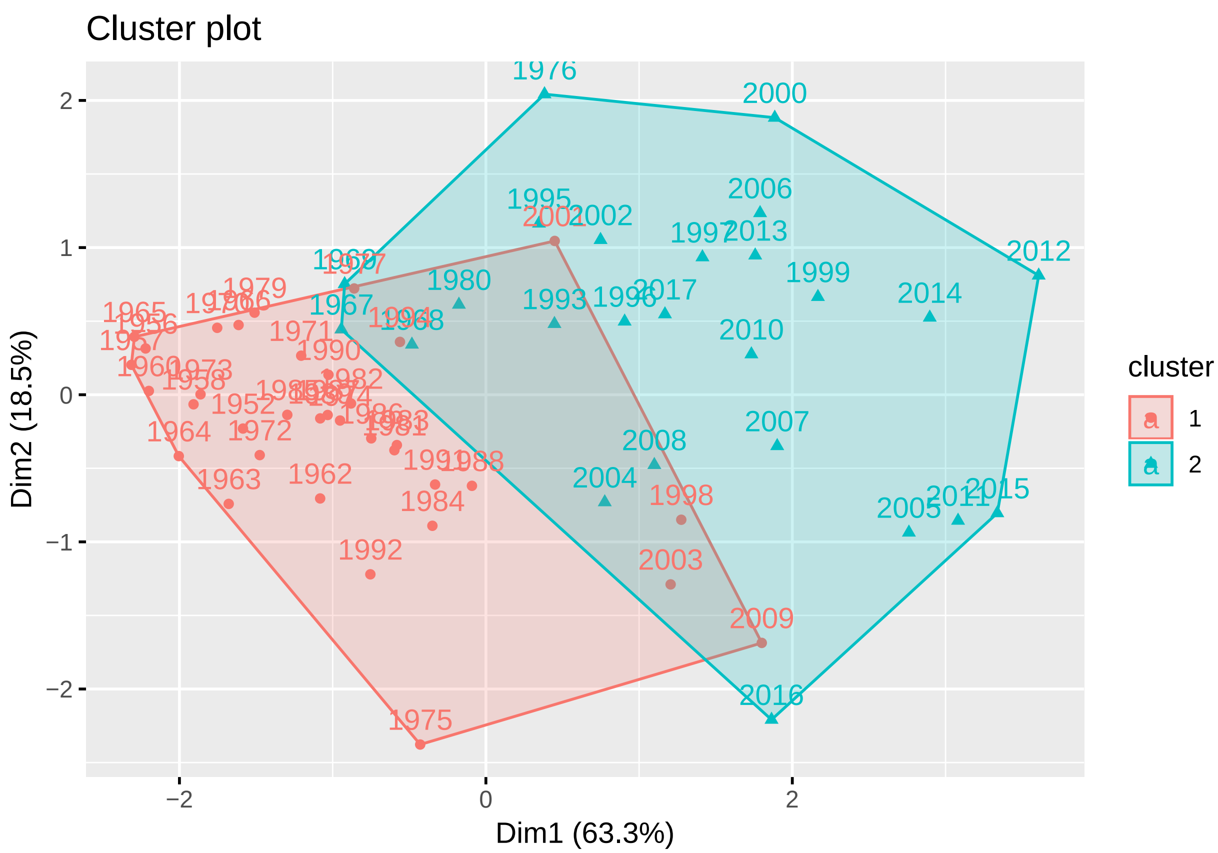
```
## K-means clustering with 4 clusters of sizes 10, 20, 15, 15
##
## Cluster means:
##   American.Crow American.Dipper American.Goldfinch American.Kestrel
## 1     1504.0000        17.80000           199.4000         67.80000
## 2      115.8500        11.50000            59.5500         21.45000
## 3      478.6667        13.46667            95.2000         30.66667
## 4      928.0000        18.26667           195.2667         63.53333
##
## Clustering vector:
## 1952 1956 1957 1958 1960 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971
##    3    2    2    2    2    3    2    2    2    3    4    4    4    3    3
## 1972 1973 1974 1975 1976 1977 1979 1980 1981 1982 1983 1984 1985 1986 1987
##    2    2    3    2    1    3    3    4    3    2    2    2    2    2    2
## 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002
##    2    3    2    2    2    4    3    4    1    1    3    1    1    3    1
## 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017
##    3    4    1    1    1    4    3    4    4    4    4    1    4    4    4
##
## Within cluster sum of squares by cluster:
## [1] 412681.6 186471.5 359440.8 496603.6
##  (between_SS / total_SS =  91.1 %)
##
## Available components:
```

```
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```
fviz_cluster(bestKM4, data=best, show.clust.cent=F)
```

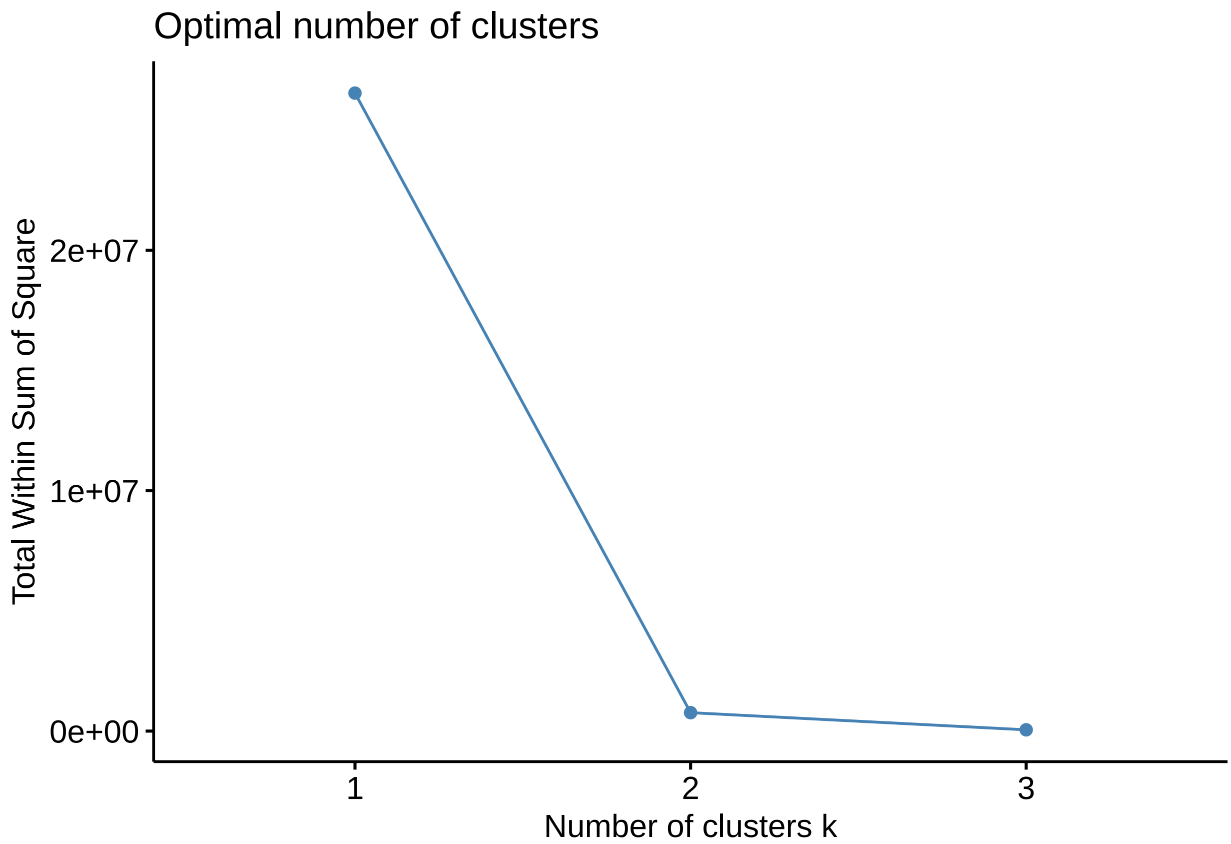



```
bestKM2 = kmeans(best, 2)
fviz_cluster(bestKM2, data=best, show.clust.cent=F)
```

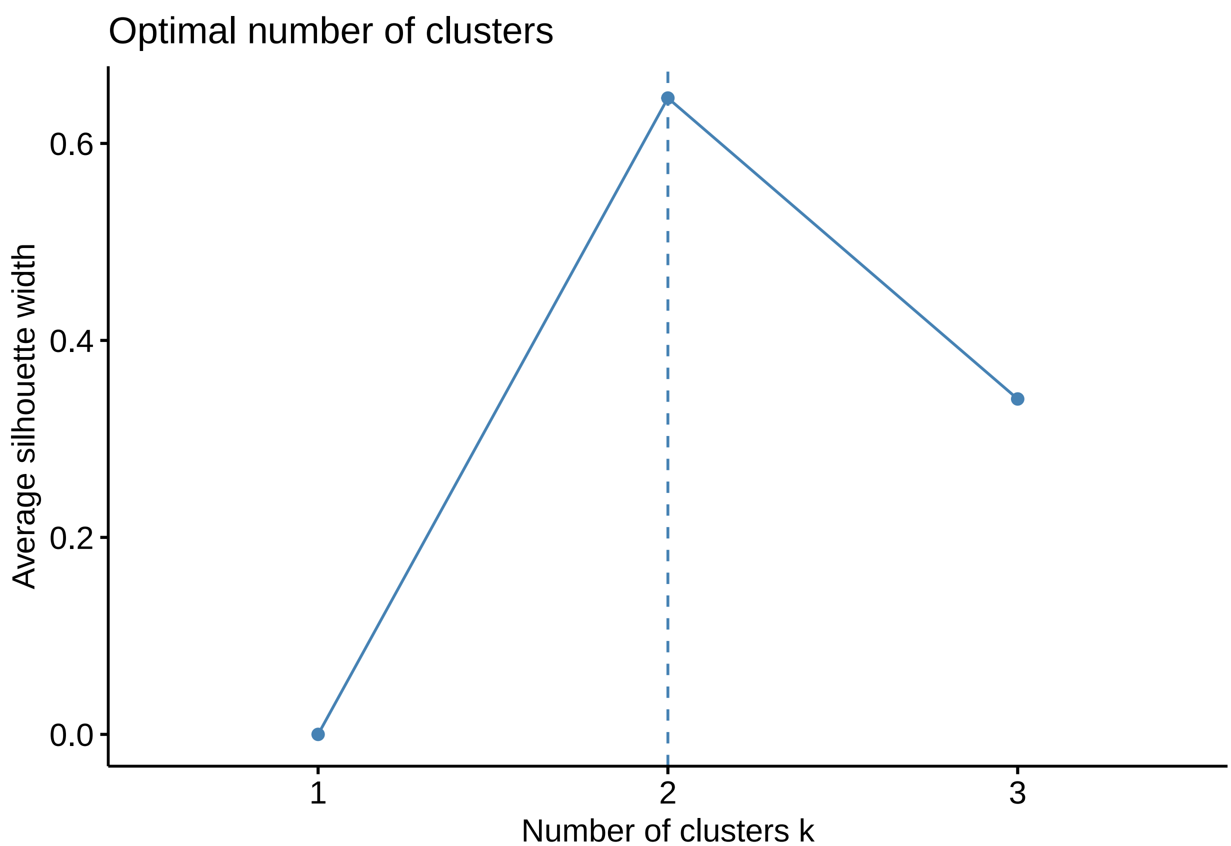


This clustered by row (year). What if we want to cluster by column (species) instead?

```
fviz_nbclust(t(best), kmeans, "wss", k.max=nrow(t(best)) - 1)
```




```
fviz_nbclust(t(best), kmeans, "silhouette", k.max=nrow(t(best)) - 1)
```

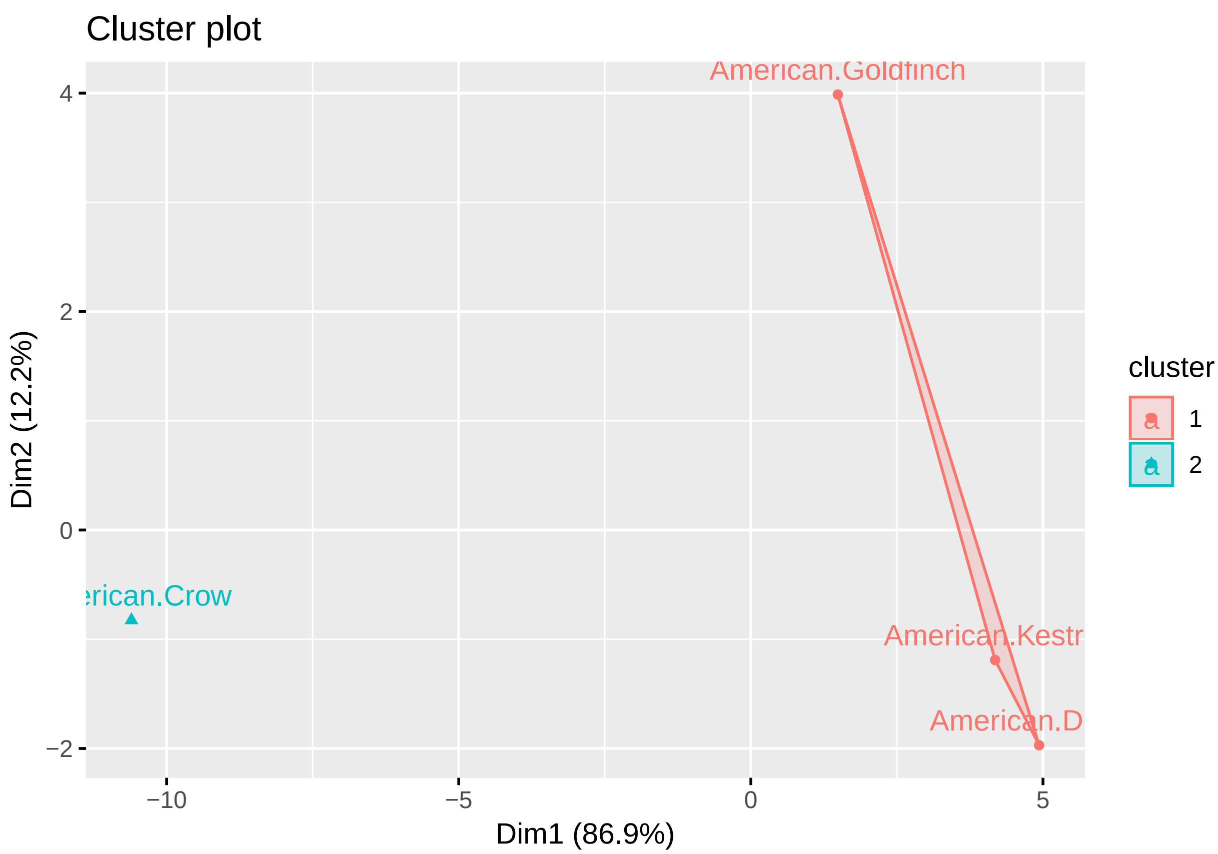



```
bestTKM2 = kmeans(t(best), 2)
bestTKM2
```

```
## K-means clustering with 2 clusters of sizes 3, 1
##
## Cluster means:
##   1952     1956     1957 1958     1960       1962 1963      1964
## 1   10 13.33333 4.333333    6 4.666667   9.333333    7  5.666667
## 2  353  5.00000 3.000000  168 2.000000 590.000000  130 13.000000
##         1965      1966      1967       1968      1969 1970 1971 1972
## 1   3.333333  16.33333  23.33333   26.66667  17.33333   13   10   12
## 2 100.000000 390.00000 870.00000 1100.00000 990.00000  390  720  248
##       1973      1974      1975       1976      1977      1979      1980
## 1 22.33333  18.33333  21.33333   60.33333  33.33333  19.33333  66.33333
## 2 29.00000 623.00000 250.00000 1756.00000 644.00000 362.00000 731.00000
##        1981 1982 1983      1984      1985 1986 1987 1988      1989
## 1  43.33333   41   74  51.33333  32.33333   50   37   82  27.66667
## 2 357.00000  125   85 184.00000 104.00000  159  166  127 352.00000
##        1990      1991     1992 1993      1994 1995       1996      1997
## 1  57.66667  57.66667 33.66667   71  50.33333   67   64.33333  109.6667
## 2 148.00000 173.00000 98.00000 1111 419.00000 1137 1266.00000 1346.0000
##       1998 1999      2000 2001       2002     2003 2004      2005
## 1 122.3333  109  106.3333  116   57.66667 104.3333   61  132.6667
## 2 495.0000 1663 1672.0000  486 1469.00000 448.0000  799 1304.0000
##         2006 2007       2008 2009     2010      2011     2012      2013
## 1   81.33333  110   73.66667  103 132.3333  157.3333 187.3333  120.3333
## 2 1713.00000 1239 1019.00000  551 805.0000 1037.0000 894.0000 1005.0000
```

```
##        2014 2015 2016     2017
## 1  118.6667  184   88 109.6667
## 2 1612.0000  748  803 871.0000
##
## Clustering vector:
##      American.Crow    American.Dipper American.Goldfinch
##                  2                  1                  1
##   American.Kestrel
##                  1
##
## Within cluster sum of squares by cluster:
## [1] 766650.7      0.0
##  (between_SS / total_SS =  97.1 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```
fviz_cluster(bestTKM2, data=t(best), show.clust.cent=F)
```



K-medoids clustering

```
install.packages("cluster")
```

```
library(cluster)
```

```
?pam
```

```
install.packages("fpc")
```

```
library(fpc)
```

```
?pamk
```

```
folkDist = as.dist(cor2dist(folkCor))
as.matrix(folkDist)[1:10,1:10]
```

```
##              Italian     Ladin Sardinian   Walloon    French   Spanish
## Italian    0.0000000 0.8782816 0.6823370 0.9140159 0.6893560 0.7720415
## Ladin      0.8782816 0.0000000 0.9917067 1.0459926 0.7858445 0.9089306
## Sardinian  0.6823370 0.9917067 0.0000000 1.0298067 0.8471606 0.9412610
## Walloon    0.9140159 1.0459926 1.0298067 0.0000000 0.9512568 1.0974803
## French     0.6893560 0.7858445 0.8471606 0.9512568 0.0000000 0.6964989
## Spanish    0.7720415 0.9089306 0.9412610 1.0974803 0.6964989 0.0000000
## Portuguese 0.8349579 1.0217388 0.9005201 1.1350364 0.7930749 0.6518344
## Catalan    0.6838287 0.9257290 0.9798357 0.9995986 0.6421311 0.5554924
## Romanian   1.0319581 0.9906031 0.9735383 1.1281215 0.9936963 1.0186373
## Welsh      1.1737091 1.1686086 1.2762787 1.2569655 1.0493463 1.0447742
##            Portuguese   Catalan  Romanian    Welsh
## Italian     0.8349579 0.6838287 1.0319581 1.173709
## Ladin       1.0217388 0.9257290 0.9906031 1.168609
## Sardinian   0.9005201 0.9798357 0.9735383 1.276279
## Walloon     1.1350364 0.9995986 1.1281215 1.256965
## French      0.7930749 0.6421311 0.9936963 1.049346
## Spanish     0.6518344 0.5554924 1.0186373 1.044774
## Portuguese  0.0000000 0.7859478 1.0544266 1.056593
## Catalan     0.7859478 0.0000000 1.0179228 1.020382
## Romanian    1.0544266 1.0179228 0.0000000 1.114386
## Welsh       1.0565932 1.0203822 1.1143862 0.000000
```

```
dim(folkDist)
```

```
## NULL
```

```
nrow(folkDist)
```

```
## NULL
```

```
ncol(folkDist)
```

```
## NULL
```

```
class(folkDist)
```

```
## [1] "dist"
```

```
attributes(folkDist)
```

```
## $Labels
##  [1] "Italian"       "Ladin"         "Sardinian"     "Walloon"
##  [5] "French"        "Spanish"       "Portuguese"    "Catalan"
##  [9] "Romanian"      "Welsh"         "Irish"         "Scottish"
## [13] "Luxembourgish" "German"        "Austrian"      "Flemish"
## [17] "Dutch"         "Frisian"       "English"       "Swedish"
## [21] "Norwegian"     "Danish"        "Faroese"       "Icelandic"
## [25] "Czech"         "Slovak"        "Lusatian"      "Polish"
## [29] "Byelorussian"  "Ukrainian"     "Russian"       "Bulgarian"
```

```
## [33] "Macedonian"    "Serbian"       "Croation"      "Slovenenian"
## [37] "Latvian"       "Lithuanian"    "Pakistani"     "Indian"
## [41] "Nepali"        "Gypsy"         "Tadzhik"       "Iranian"
## [45] "Kurdish"       "Afghan"        "Ossetian"      "Albanian"
## [49] "Greek"         "Armenian"
##
## $Size
## [1] 50
##
## $call
## as.dist.default(m = cor2dist(folkCor))
##
## $class
## [1] "dist"
##
## $Diag
## [1] FALSE
##
## $Upper
## [1] FALSE
```

```
folkPAMK = pamk(folkDist, diss=T, krange=2:attributes(folkDist)$Size-1)
folkPAMK
```

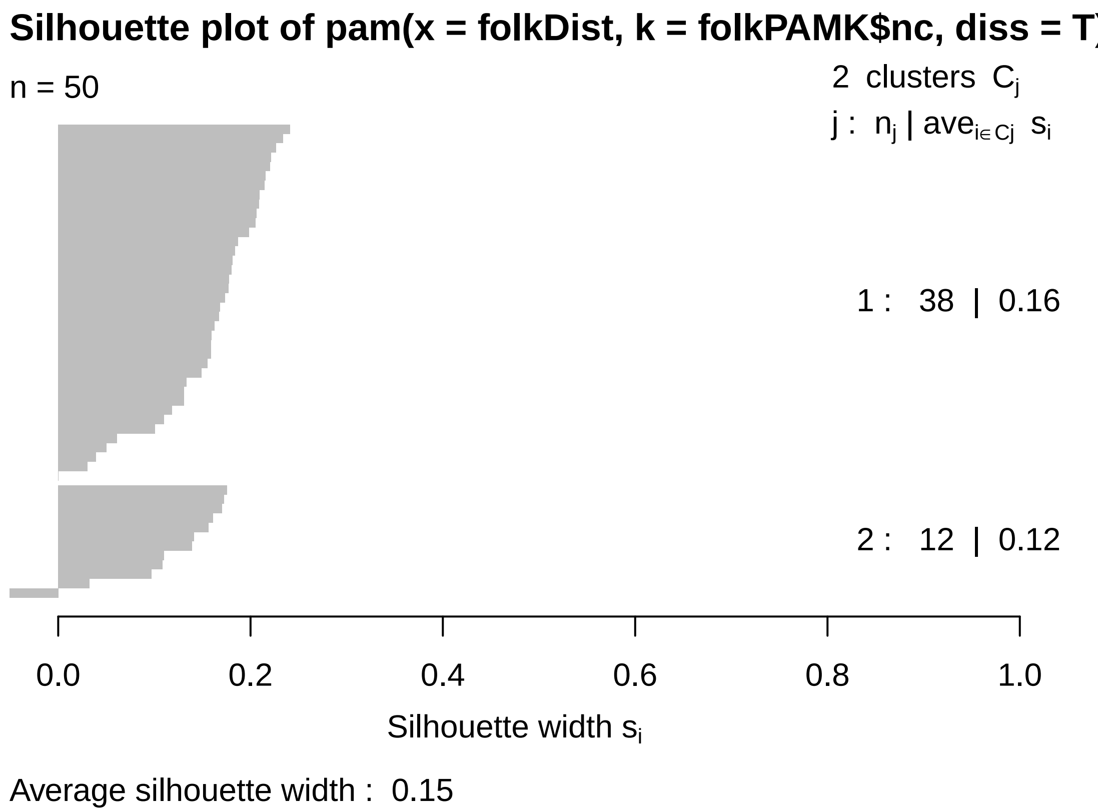
```
## $pamobject
## Medoids:
##      ID
## [1,] "5"  "French"
## [2,] "44" "Iranian"
## Clustering vector:
##       Italian         Ladin     Sardinian       Walloon        French
##             1             1             1             1             1
##       Spanish    Portuguese       Catalan      Romanian         Welsh
##             1             1             1             2             1
##         Irish      Scottish Luxembourgish        German      Austrian
##             1             1             1             1             1
##       Flemish         Dutch       Frisian       English       Swedish
##             1             1             1             1             1
##     Norwegian        Danish       Faroese     Icelandic         Czech
##             1             1             1             1             1
##        Slovak      Lusatian        Polish  Byelorussian     Ukrainian
##             1             1             1             1             1
##       Russian     Bulgarian    Macedonian       Serbian      Croation
##             1             2             2             1             1
##   Slovenenian       Latvian    Lithuanian     Pakistani        Indian
##             1             1             1             2             2
##        Nepali         Gypsy       Tadzhik       Iranian       Kurdish
##             2             1             2             2             2
##        Afghan      Ossetian      Albanian         Greek      Armenian
##             2             1             2             1             2
## Objective function:
##     build      swap
## 0.8496261 0.8496261
##
## Available components:
```

```
## [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
## [6] "clusinfo"   "silinfo"    "diss"       "call"
##
## $nc
## [1] 2
##
## $crit
##  [1] 0.000000000 0.149179002 0.102074871 0.080441029 0.084323994
##  [6] 0.080504114 0.073613609 0.081312922 0.087193120 0.085950584
## [11] 0.086882719 0.085815507 0.082688501 0.079179050 0.081001586
## [16] 0.084154537 0.084107761 0.087905382 0.082774597 0.079421444
## [21] 0.076192212 0.080882142 0.082893335 0.086114439 0.084555630
## [26] 0.079142232 0.082129217 0.082401490 0.079769015 0.078175093
## [31] 0.076887800 0.076728420 0.078670137 0.073632584 0.067214642
## [36] 0.052717717 0.049145717 0.043758800 0.040458898 0.040388866
## [41] 0.032240344 0.028562676 0.026081553 0.025265582 0.023046323
## [46] 0.018532269 0.013992138 0.009393337 0.003738835
```

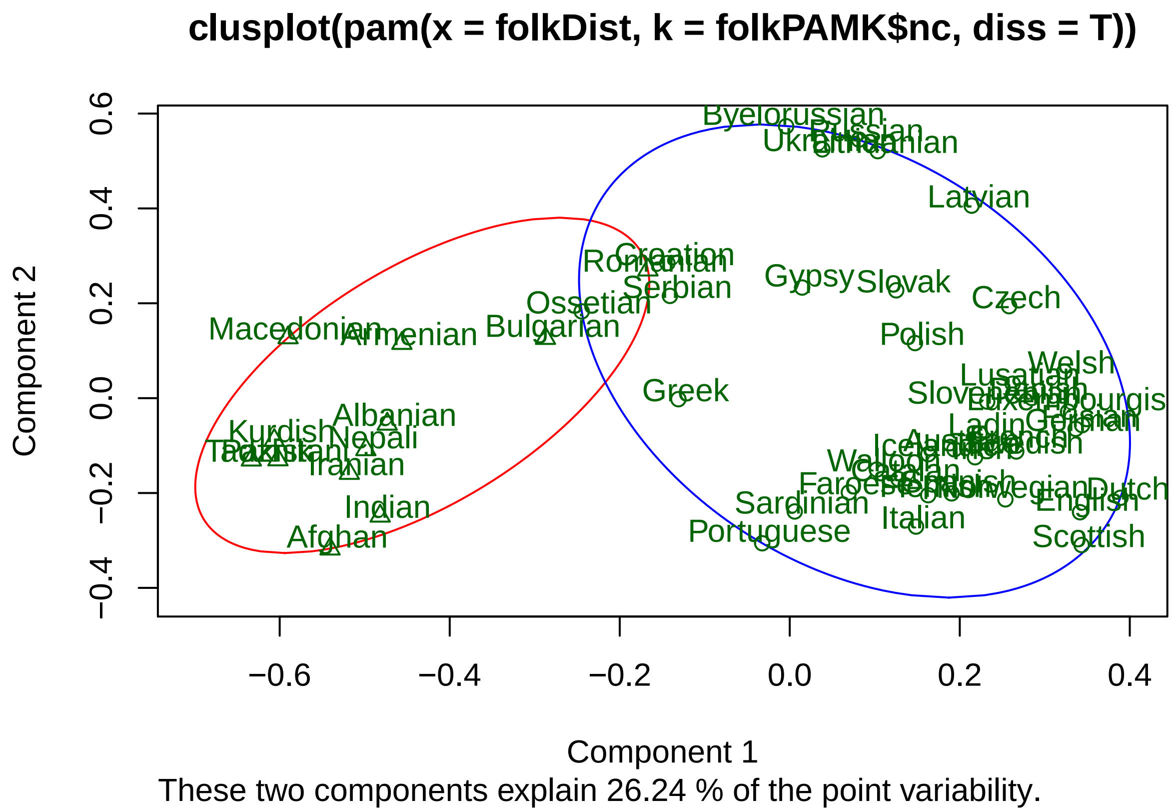
```
folkPAM = pam(folkDist, diss=T, k=folkPAMK$nc)
folkPAM
```

```
## Medoids:
##      ID
## [1,] "5"  "French"
## [2,] "44" "Iranian"
## Clustering vector:
##       Italian         Ladin     Sardinian       Walloon        French
##             1             1             1             1             1
##       Spanish    Portuguese       Catalan      Romanian         Welsh
##             1             1             1             2             1
##         Irish      Scottish Luxembourgish        German      Austrian
##             1             1             1             1             1
##       Flemish         Dutch       Frisian       English       Swedish
##             1             1             1             1             1
##     Norwegian        Danish       Faroese     Icelandic         Czech
##             1             1             1             1             1
##        Slovak      Lusatian        Polish  Byelorussian     Ukrainian
##             1             1             1             1             1
##       Russian     Bulgarian    Macedonian       Serbian      Croation
##             1             2             2             1             1
##    Slovenenian       Latvian    Lithuanian     Pakistani        Indian
##             1             1             1             2             2
##        Nepali         Gypsy       Tadzhik       Iranian       Kurdish
##             2             1             2             2             2
##        Afghan      Ossetian      Albanian         Greek      Armenian
##             2             1             2             1             2
## Objective function:
##     build      swap
## 0.8496261 0.8496261
##
## Available components:
## [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
## [6] "clusinfo"   "silinfo"    "diss"       "call"
```

```
plot(folkPAM)
```




```
?clusplot
clusplot(folkPAM, lines=0, color=T, labels=3)
```



Exploratory factor analysis

```
?factanal
```

```
?psych::fa
```

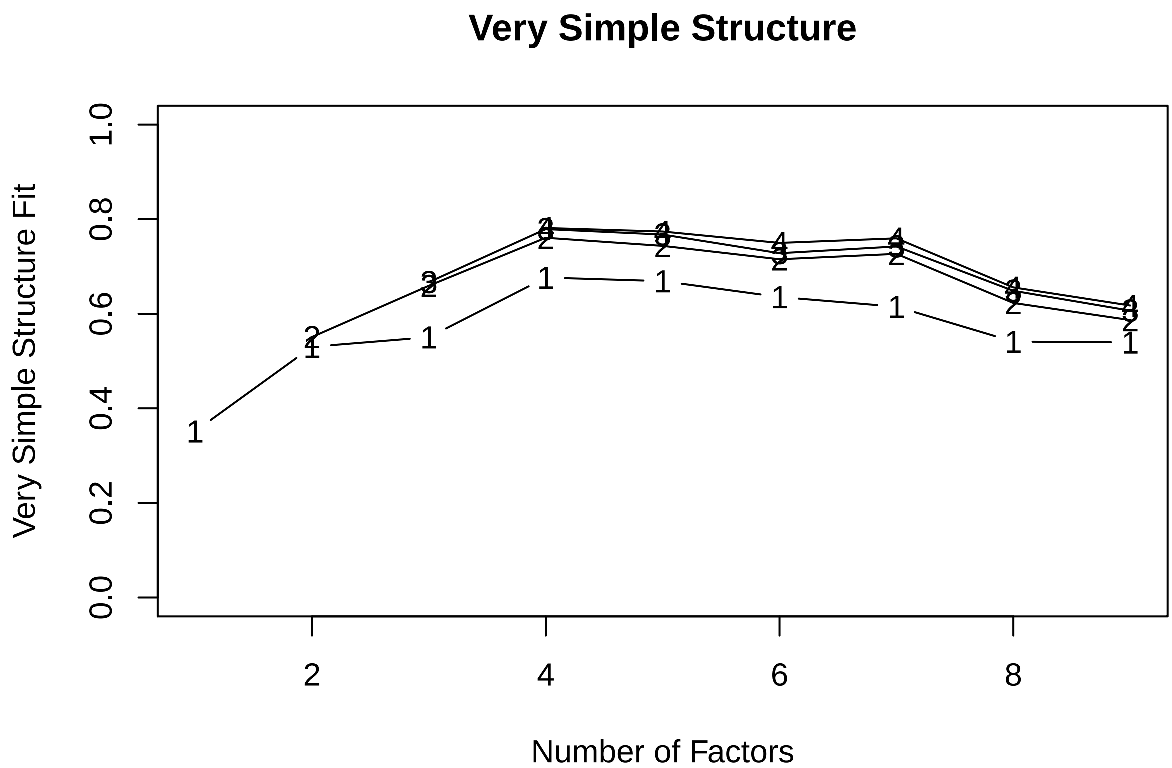
```
install.packages("GPArotation")
```

```
?psych::vss
?psych::fa.parallel
```

```
vss(bmLikCor, n=nrow(bmLikCor)-1, rotate="oblimin", fm="ml", n.obs=nrow(bmLik))
```

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

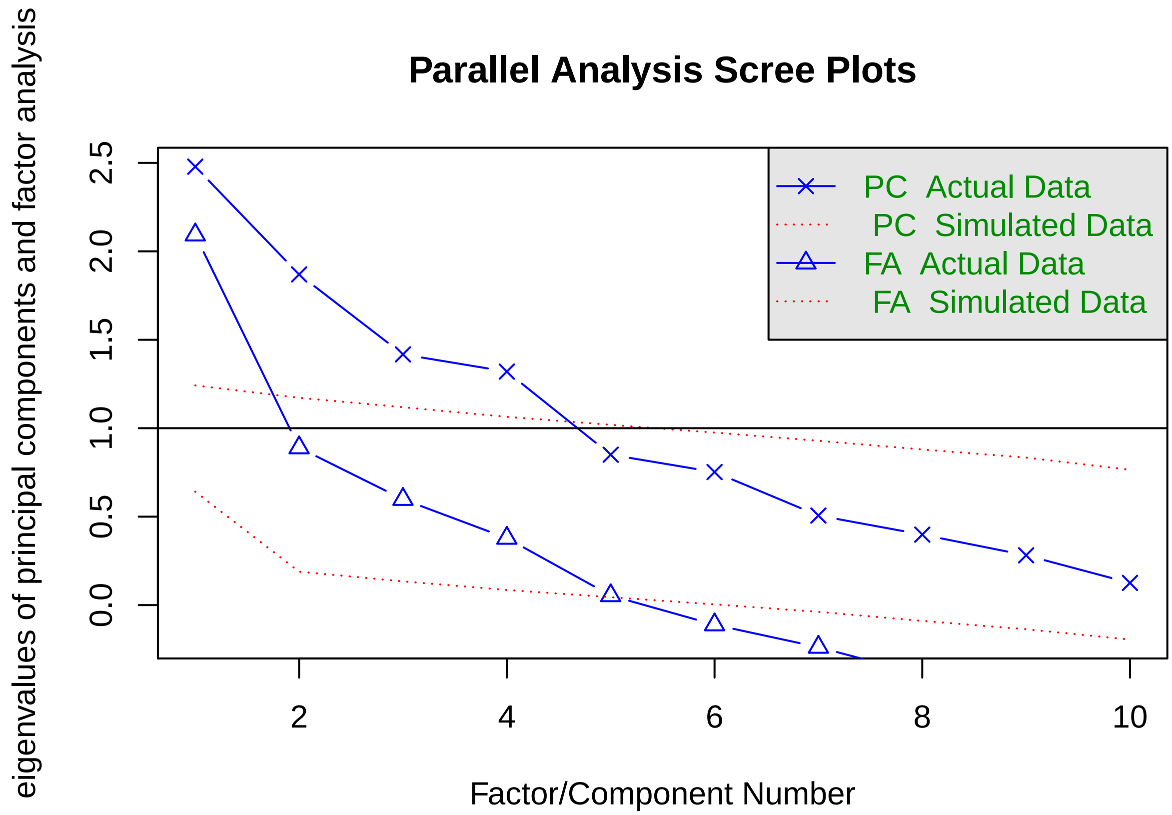
```
## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate =
## rotate, : A loading greater than abs(1) was detected. Examine the loadings
## carefully.

## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate =
## rotate, : A loading greater than abs(1) was detected. Examine the loadings
## carefully.
```




```
##
## Very Simple Structure
## Call: vss(x = bmLikCor, n = nrow(bmLikCor) - 1, rotate = "oblimin",
##     fm = "ml", n.obs = nrow(bmLik))
## VSS complexity 1 achieves a maximimum of 0.68  with  4  factors
## VSS complexity 2 achieves a maximimum of 0.76  with  4  factors
##
## The Velicer MAP achieves a minimum of 0.07  with  1  factors
## BIC achieves a minimum of  NA  with  5  factors
## Sample Size adjusted BIC achieves a minimum of  NA  with  5  factors
##
## Statistics by number of factors
##   vss1 vss2   map dof   chisq     prob sqresid  fit RMSEA    BIC SABIC
## 1 0.35 0.00 0.066  35 6.7e+02 5.6e-118     9.9 0.35  0.21 457.19   568
## 2 0.53 0.55 0.084  26 3.9e+02  7.6e-67     6.8 0.55  0.19 234.92   317
## 3 0.55 0.66 0.106  18 2.2e+02  4.9e-37     5.1 0.67  0.17 113.21   170
## 4 0.68 0.76 0.099  11 1.1e+02  1.2e-17     3.3 0.78  0.15  39.79    75
## 5 0.67 0.74 0.131   5 3.0e+01  1.7e-05     3.4 0.77  0.11  -0.39    15
## 6 0.64 0.72 0.198   0 1.4e+01       NA     3.8 0.75    NA     NA    NA
## 7 0.61 0.73 0.379  -4 1.1e-07       NA     3.4 0.77    NA     NA    NA
## 8 0.54 0.62 0.546  -7 2.3e-09       NA     5.1 0.66    NA     NA    NA
## 9 0.54 0.59 1.000  -9 0.0e+00       NA     5.7 0.62    NA     NA    NA
##   complex  eChisq    SRMR eCRMS  eBIC
## 1     1.0 9.9e+02 1.6e-01 0.186 781.0
## 2     1.1 5.4e+02 1.2e-01 0.159 380.5
## 3     1.3 2.7e+02 8.6e-02 0.136 162.6
```

```
## 4      1.3 7.0e+01 4.4e-02 0.088   4.2
## 5      1.4 1.7e+01 2.1e-02 0.064 -13.6
## 6      1.4 5.5e+00 1.2e-02    NA    NA
## 7      1.6 8.1e-08 1.5e-06    NA    NA
## 8      1.3 1.6e-09 2.1e-07    NA    NA
## 9      1.3 1.2e-17 1.8e-11    NA    NA
```

```
fa.parallel(bmLikCor, fm="ml", n.obs=nrow(bmLik))
```




```
## Parallel analysis suggests that the number of factors =  5  and the number of components =  4
```

```
bmEFA = fa(bmLikCor, nfactors=4, rotate="oblimin", fm="ml", n.obs=nrow(bmLik))
```

```
## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate =
## rotate, : A loading greater than abs(1) was detected. Examine the loadings
## carefully.
```

```
bmEFA
```

```
## Factor Analysis using method =  ml
## Call: fa(r = bmLikCor, nfactors = 4, n.obs = nrow(bmLik), rotate = "oblimin",
##     fm = "ml")
##
##  Warning: A Heywood case was detected.
## Standardized loadings (pattern matrix) based upon correlation matrix
##            ML1   ML2   ML3   ML4   h2    u2 com
## natural   0.83 -0.03  0.00  0.09 0.73 0.274 1.0
## remotnes  1.01  0.04  0.01 -0.07 1.00 0.005 1.0
## scenic_b  0.44 -0.19  0.06  0.44 0.51 0.490 2.4
```

```
## hunting  -0.03  0.99  0.02  0.06 1.00 0.005 1.0
## fishing   0.15  0.66 -0.07 -0.11 0.45 0.551 1.2
## recent_f -0.03  0.09 -0.07  0.64 0.42 0.583 1.1
## test_ski -0.08  0.14  0.21  0.51 0.35 0.648 1.5
## familiar  0.06  0.09 -0.28  0.38 0.23 0.770 2.0
## variety   0.01  0.01  0.96 -0.01 0.93 0.067 1.0
## friend_s  0.00 -0.03  0.35  0.09 0.14 0.859 1.1
##
##                        ML1  ML2  ML3  ML4
## SS loadings           1.97 1.50 1.20 1.08
## Proportion Var        0.20 0.15 0.12 0.11
## Cumulative Var        0.20 0.35 0.47 0.57
## Proportion Explained  0.34 0.26 0.21 0.19
## Cumulative Proportion 0.34 0.60 0.81 1.00
##
##  With factor correlations of
##       ML1   ML2   ML3  ML4
## ML1  1.00 -0.06  0.14 0.19
## ML2 -0.06  1.00 -0.05 0.12
## ML3  0.14 -0.05  1.00 0.10
## ML4  0.19  0.12  0.10 1.00
##
## Mean item complexity =  1.3
## Test of the hypothesis that 4 factors are sufficient.
##
## The degrees of freedom for the null model are  45  and the objective function was  3.23 with Chi Squa
## The degrees of freedom for the model are 11  and the objective function was  0.26
##
## The root mean square of the residuals (RMSR) is  0.04
## The df corrected root mean square of the residuals is  0.09
##
## The harmonic number of observations is  409 with the empirical chi square  70.36  with prob <  1e-10
## The total number of observations was  409  with Likelihood Chi Square =  105.94  with prob <  1.2e-17
##
## Tucker Lewis Index of factoring reliability =  0.689
## RMSEA index =  0.147  and the 90 % confidence intervals are  0.121 0.171
## BIC =  39.79
## Fit based upon off diagonal values = 0.97
## Measures of factor score adequacy
##                                                   ML1  ML2  ML3  ML4
## Correlation of (regression) scores with factors   1.00 1.00 0.97 0.81
## Multiple R square of scores with factors          0.99 0.99 0.93 0.66
## Minimum correlation of possible factor scores     0.99 0.99 0.87 0.31
```

```
bmEFA$loadings
```

```
##
## Loadings:
##          ML1    ML2    ML3    ML4
## natural   0.830
## remotnes  1.009
## scenic_b  0.444 -0.191         0.440
## hunting          0.988
## fishing   0.152  0.664        -0.110
## recent_f                       0.636
```

```
## test_ski         0.139  0.208  0.515
## familiar               -0.283  0.384
## variety                 0.965
## friend_s                0.352
##
##                  ML1   ML2   ML3   ML4
## SS loadings    1.939 1.492 1.193 1.047
## Proportion Var 0.194 0.149 0.119 0.105
## Cumulative Var 0.194 0.343 0.462 0.567
```

(pdf / Rmd)