Factor analysis

Vs. principal components

- Principal components:
  - Purely mathematical.
  - Find eigenvalues, eigenvectors of correlation matrix.
  - No testing whether observed components reproducible, or even probability model behind it.
- Factor analysis:
  - some way towards fixing this (get test of appropriateness)
  - In factor analysis, each variable modelled as: “common factor” (eg. verbal ability) and “specific factor” (left over).
  - Choose the common factors to “best” reproduce pattern seen in correlation matrix.
  - Iterative procedure, different answer from principal components.

Packages

```
library(ggbiplot)
library(tidyverse)
library(conflicted)
conflict_prefer("mutate", "dplyr")
conflict_prefer("select", "dplyr")
conflict_prefer("filter", "dplyr")
conflict_prefer("arrange", "dplyr")
```

Example

- 145 children given 5 tests, called PARA, SENT, WORD, ADD and DOTS. 3 linguistic tasks (paragraph comprehension, sentence completion and word meaning), 2 mathematical ones (addition and counting dots).
- Correlation matrix of scores on the tests:

para	1	0.722	0.714	0.203	0.095
sent	0.722	1	0.685	0.246	0.181
word	0.714	0.685	1	0.170	0.113
add	0.203	0.246	0.170	1	0.585
dots	0.095	0.181	0.113	0.585	1

- Is there small number of underlying "constructs" (unobservable) that explains this pattern of correlations?

To start: principal components

Using correlation matrix. Read that first:

```
my_url <- "http://ritsokiguess.site/datafiles/rex2.txt"
kids <- read_delim(my_url, " ")
kids
```

```
# A tibble: 5 x 6
  test   para  sent  word   add  dots
  <chr> <dbl> <dbl> <dbl> <dbl> <dbl>
1 para  1     0.722 0.714 0.203 0.095
2 sent  0.722 1     0.685 0.246 0.181
3 word  0.714 0.685 1     0.17  0.113
4 add   0.203 0.246 0.17  1     0.585
5 dots  0.095 0.181 0.113 0.585 1
```

Principal components on correlation matrix

Turn into R `matrix`, using column `test` as column names:

```
kids %>%
column_to_rownames("test") %>%
as.matrix() -> m
```

Principal components:

```
kids.0 <- princomp(covmat = m)
```

I used `kids.0` here since I want `kids.1` and `kids.2` later.

Scree plot

```
# ggscreeplot(kids.0)
```

Principal component results

- Need 2 components. Loadings:

```
kids.0$loadings
```

```
Loadings:
     Comp.1 Comp.2 Comp.3 Comp.4 Comp.5
para  0.534  0.245  0.114         0.795
sent  0.542  0.164         0.660 -0.489
word  0.523  0.247 -0.144 -0.738 -0.316
add   0.297 -0.627  0.707              
dots  0.241 -0.678 -0.680         0.143

               Comp.1 Comp.2 Comp.3 Comp.4 Comp.5
SS loadings       1.0    1.0    1.0    1.0    1.0
Proportion Var    0.2    0.2    0.2    0.2    0.2
Cumulative Var    0.2    0.4    0.6    0.8    1.0
```

Comments

- First component has a bit of everything, though especially the first three tests.
- Second component rather more clearly `add` and `dots`.
- No scores, plots since no actual data.
- See how factor analysis compares on these data.

Factor analysis

- Specify number of factors first, get solution with exactly that many factors.
- Includes hypothesis test, need to specify how many children wrote the tests.
- Works from correlation matrix via `covmat` or actual data, like `princomp`.
- Introduces extra feature, *rotation*, to make interpretation of loadings (factor-variable relation) easier.

Factor analysis for the kids data

- Create "covariance list" to include number of children who wrote the tests.
- Feed this into `factanal`, specifying how many factors (2).
- Start with the matrix we made before.

```
m
```

```
     para  sent  word   add  dots
para 1.000 0.722 0.714 0.203 0.095
sent 0.722 1.000 0.685 0.246 0.181
word 0.714 0.685 1.000 0.170 0.113
add  0.203 0.246 0.170 1.000 0.585
dots 0.095 0.181 0.113 0.585 1.000
```

```
ml <- list(cov = m, n.obs = 145)
kids.2 <- factanal(factors = 2, covmat = ml)
```

Uniquenesses

```r
kids.2$uniquenesses
```

```
     para      sent      word       add      dots
0.2424457 0.2997349 0.3272312 0.5743568 0.1554076
```

- Uniquenesses say how "unique" a variable is (size of specific factor). Small uniqueness means that the variable is summarized by a factor (good).
- Very large uniquenesses are bad; `add`'s uniqueness is largest but not large enough to be worried about.
- Also see "communality" for this idea, where *large* is good and *small* is bad.

Loadings

```
kids.2$loadings
```

```
Loadings:
     Factor1 Factor2
para 0.867
sent 0.820   0.166
word 0.816
add  0.167   0.631
dots         0.918

               Factor1 Factor2
SS loadings      2.119   1.282
Proportion Var   0.424   0.256
Cumulative Var   0.424   0.680
```

- Loadings show how each factor depends on variables. Blanks indicate "small", less than 0.1.

Comments

- Factor 1 clearly the “linguistic” tasks, factor 2 clearly the “mathematical” ones.
- Two factors together explain 68% of variability (like regression R-squared).
- Which variables belong to which factor is *much* clearer than with principal components.

Are 2 factors enough?

```r
kids.2$STATISTIC
```

```
objective
0.5810578
```

```r
kids.2$dof
```

```
[1] 1
```

```r
kids.2$PVAL
```

```
objective
 0.445898
```

P-value not small, so 2 factors OK.

1 factor

```
kids.1 <- factanal(factors = 1, covmat = ml)
kids.1$STATISTIC
```

```
objective
 58.16534
```

```
kids.1$dof
```

```
[1] 5
```

```
kids.1$PVAL
```

```
   objective
2.907856e-11
```

1 factor rejected (P-value small). Definitely need more than 1.

Places rated, again

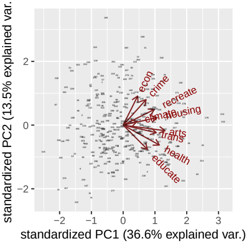
- Read data, transform, rerun principal components, get biplot:

```
my_url <- "http://ritsokiguess.site/datafiles/places.txt"
places0 <- read_table(my_url)
places0 %>%
mutate(across(-id, \(x) log(x))) -> places
places %>% select(-id) -> places_numeric
places.1 <- princomp(places_numeric, cor = TRUE)
g <- ggbiplot(places.1, labels = places$id,
       labels.size = 0.8)
```

- This is all exactly as for principal components (nothing new here).

The biplot

```
g
```



standardized PC2 (13.5% explained var.)
standardized PC1 (36.6% explained var.)
econ
crime
recreate
climate
housing
arts
trans
health
educate

Comments

- Most of the criteria are part of components 1 *and* 2.
- If we can rotate the arrows counterclockwise:
  - economy and crime would point straight up
    - part of component 2 only
  - health and education would point to the right
    - part of component 1 only
- would be easier to see which variables belong to which component.
- Factor analysis includes a rotation to help with interpretation.

Factor analysis

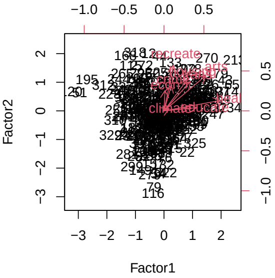
- Have to pick a number of factors *first*.
- Do this by running principal components and looking at scree plot.
- In this case, 3 factors seemed good (revisit later):

```
places.3 <- factanal(places_numeric, 3, scores = "r")
```

- There are different ways to get factor scores. These called "regression" scores.

A bad biplot

```r
biplot(places.3$scores, places.3$loadings,
xlabs = places$id)
```



Comments

- I have to find a way to make a better biplot!
- Some of the variables now point straight up and some straight across (if you look carefully for the red arrows among the black points).
- This should make the factors more interpretable than the components were.

Factor loadings

```r
places.3$loadings
```

```

Loadings:
         Factor1 Factor2 Factor3
climate                   0.994
housing   0.360   0.482   0.229
health    0.884   0.164
crime     0.115   0.400   0.205
trans     0.414   0.460
educate   0.511
arts      0.655   0.552   0.102
recreate  0.148   0.714
econ              0.318  -0.114

               Factor1 Factor2 Factor3
SS loadings      1.814   1.551   1.120
Proportion Var   0.202   0.172   0.124
Cumulative Var   0.202   0.374   0.498
```

Comments on loadings

- These are at least somewhat clearer than for the principal components:
- Factor 1: health, education, arts: “well-being”
- Factor 2: housing, transportation, arts (again), recreation: “places to be”
- Factor 3: climate (only): “climate”
- In this analysis, economic factors don’t seem to be important.

Factor scores

- Make a dataframe with the city IDs and factor scores:

```
cbind(id = places$id, places.3$scores) %>%
as_tibble() -> places_scores
```

- Make percentile ranks again (for checking):

```
places %>%
mutate(across(-id, \(x) percent_rank(x))) -> places_pr
```

Highest scores on factor 1, “well-being”:

- for the top 4 places:

```
places_scores %>%
slice_max(Factor1, n = 4)
```

```
# A tibble: 4 x 4
     id Factor1 Factor2 Factor3
  <dbl>   <dbl>   <dbl>   <dbl>
1   213    2.47   1.78    0.506
2    65    2.39   0.925  -0.287
3   234    2.32   0.122   0.524
4   314    2.22   0.671   0.521
```

Check percentile ranks for factor 1

```
places_pr %>%
select(id, health, educate, arts) %>%
filter(id %in% c(213, 65, 234, 314))
```

```
# A tibble: 4 x 4
     id health educate  arts
  <dbl>  <dbl>   <dbl> <dbl>
1    65  0.997   0.963 0.997
2   213  1       0.723 1
3   234  0.991   1     0.985
4   314  0.985   0.994 0.991
```

- These are definitely high on the well-being variables.
- City #213 is not so high on education, but is highest of all on the others.

Highest scores on factor 2, “places to be”:

```
places_scores %>%
slice_max(Factor2, n = 4)
```

```
# A tibble: 4 x 4
     id Factor1 Factor2 Factor3
  <dbl>   <dbl>   <dbl>   <dbl>
1   318  -1.01     2.05 -0.0957
2    12  -0.540    2.02 -3.80
3   168  -1.35     1.94  0.273
4    44  -0.149    1.92 -0.556
```

Check percentile ranks for factor 2

```
places_pr %>%
select(id, housing, trans, arts, recreate) %>%
filter(id %in% c(318, 12, 168, 44))
```

```
# A tibble: 4 x 5
     id housing trans  arts recreate
  <dbl>   <dbl> <dbl> <dbl>    <dbl>
1    12   0.933 0.729 0.604    0.896
2    44   0.927 0.963 0.735    0.988
3   168   0.832 0.872 0.442    0.979
4   318   0.881 0.744 0.668    0.963
```

- These are definitely high on housing and recreation.
- Some are (very) high on transportation, but not so much on arts.
- Could look at more cities to see if #168 being low on arts is a fluke.

Highest scores on factor 3, "climate":

```
places_scores %>%
slice_max(Factor3, n = 4)
```

```
# A tibble: 4 x 4
     id Factor1 Factor2 Factor3
  <dbl>   <dbl>   <dbl>   <dbl>
1   227  -0.184   0.385    2.04
2   218   0.881   0.897    2.02
3   269   0.932   1.19     1.98
4   270   1.50    1.84     1.94
```

Check percentile ranks for factor 3

```
places_pr %>%
select(id, climate) %>%
filter(id %in% c(227, 218, 269, 270))
```

```
# A tibble: 4 x 2
     id climate
  <dbl>   <dbl>
1   218   0.997
2   227   0.991
3   269   0.994
4   270   0.997
```

This is very clear.

Uniquenesses

- We said earlier that the economy was not part of any of our factors:

```
places.3$uniquenesses
```

```
  climate   housing    health     crime     trans   educate
0.0050000 0.5859175 0.1854084 0.7842407 0.6165449 0.735192
     econ
0.8856382
```

- The higher the uniqueness, the less the variable concerned is part of any of our factors (and that maybe another factor is needed to accommodate it).
- This includes economy and maybe crime.

Test of significance

We can test whether the three factors that we have is enough, or whether we need more to describe our data:

```r
places.3$PVAL
```

```
   objective
1.453217e-14
```

- 3 factors are not enough.
- What would 5 factors look like?

Five factors

```r
places.5 <- factanal(places_numeric, 5, scores = "r")
places.5$loadings
```

```
Loadings:
         Factor1 Factor2 Factor3 Factor4 Factor5
climate                           0.131   0.559 
housing   0.286   0.505   0.289  -0.113   0.475 
health    0.847   0.214                   0.187 
crime             0.196   0.143   0.948   0.181 
trans     0.389   0.515           0.175         
educate   0.534                                 
arts      0.611   0.564           0.172   0.145 
recreate          0.705           0.115   0.136 
econ                      0.978   0.135         

               Factor1 Factor2 Factor3 Factor4 Factor5
SS loadings      1.628   1.436   1.087   1.023   0.658
Proportion Var   0.181   0.160   0.121   0.114   0.073
Cumulative Var   0.181   0.340   0.461   0.575   0.648
```

Comments 1/2

- On (new) 5 factors:
- Factor 1 is health, education, arts: same as factor 1 before.
- Factor 2 is housing, transportation, arts, recreation: as factor 2 before.
- Factor 3 is economy.
- Factor 4 is crime.
- Factor 5 is climate and housing: like factor 3 before.

Comments 2/2

- The two added factors include the two “missing” variables.
- Is this now enough?

```
places.5$PVAL
```

```
   objective
0.0009741394
```

- No. My guess is that the authors of Places Rated chose their 9 criteria to capture different aspects of what makes a city good or bad to live in, and so it was too much to hope that a small number of factors would come out of these.

A bigger example: BEM sex role inventory

- 369 women asked to rate themselves on 60 traits, like "self-reliant" or "shy".
- Rating 1 "never or almost never true of me" to 7 "always or almost always true of me' '.
- 60 personality traits is a lot. Can we find a smaller number of factors that capture aspects of personality?
- The whole BEM sex role inventory on next page.

The whole inventory

1. self reliant 2. yielding 3. helpful 4. defends own beliefs 5. cheerful 6. moody 7. independent 8. shy 9. conscientious 10. athletic 11. affectionate 12. theatrical 13. assertive 14. flatterable 15. happy 16. strong personality 17. loyal 18. unpredictable 19. forceful 20. feminine	21. reliable 22. analytical 23. sympathetic 24. jealous 25. leadership ability 26. sensitive to other's needs 27. truthful 28. willing to take risks 29. understanding 30. secretive 31. makes decisions easily 32. compassionate 33. sincere 34. self-sufficient 35. eager to soothe hurt feelings 36. conceited 37. dominant 38. soft spoken 39. likable 40. masculine	41. warm 42. solemn 43. willing to take a stand 44. tender 45. friendly 46. aggressive 47. gullible 48. inefficient 49. acts as a leader 50. childlike 51. adaptable 52. individualistic 53. does not use harsh language 54. unsystematic 55. competitive 56. loves children 57. tactful 58. ambitious 59. gentle 60. conventional

Some of the data

```
my_url <- "http://ritsokiguess.site/datafiles/factor.txt"
bem <- read_tsv(my_url)
bem
```

```
# A tibble: 369 x 45
   subno helpful reliant defbel yielding cheerful indpt athlet   shy assert
   <dbl>   <dbl>   <dbl>  <dbl>    <dbl>    <dbl> <dbl>  <dbl> <dbl>  <dbl>
 1     1       7       7      5        5        7     7      7     1      7
 2     2       5       6      6        6        2     3      3     3      4
 3     3       7       6      4        4        5     5      2     3      4
 4     4       6       6      7        4        6     6      3     4      4
 5     5       6       6      7        4        7     7      7     2      7
 6     7       5       6      7        4        6     6      2     4      4
 7     8       6       4      6        6        6     3      1     3      3
 8     9       7       6      7        5        6     7      5     2      5
 9    10       7       6      6        4        4     5      2     2      5
10    11       7       4      7        4        7     5      2     1      5
# i 359 more rows
# i 35 more variables: strpers <dbl>, forceful <dbl>, affect <dbl>,
#   flatter <dbl>, loyal <dbl>, analyt <dbl>, feminine <dbl>, sympathy <dbl>,
#   moody <dbl>, sensitiv <dbl>, undstand <dbl>, compass <dbl>, leaderab <dbl>,
#   soothe <dbl>, risk <dbl>, decide <dbl>, selfsuff <dbl>, conscien <dbl>,
#   dominant <dbl>, masculin <dbl>, stand <dbl>, happy <dbl>, softspok <dbl>,
#   warm <dbl>, truthful <dbl>, tender <dbl>, gullible <dbl>, ...
```

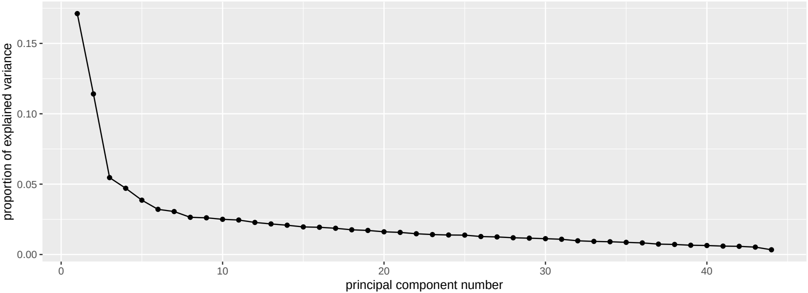
Principal components first

…to decide on number of factors:

```
bem.pc <- bem %>%
select(-subno) %>%
princomp(cor = T)
```

The scree plot

```r
(g <- ggscreeplot(bem.pc))
```

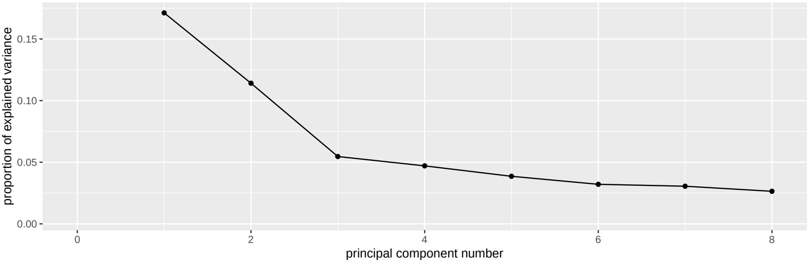


- No obvious elbow.

Zoom in to search for elbow

Possible elbows at 3 (2 factors) and 6 (5):

```
g + scale_x_continuous(limits = c(0, 8))
```



but is 2 really good?

```r
summary(bem.pc)
```

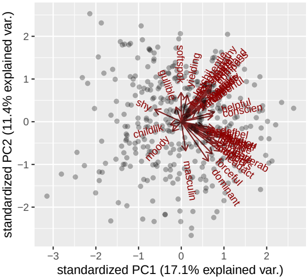
```
Importance of components:
                          Comp.1    Comp.2     Comp.3     Comp.4     Comp.5
Standard deviation     2.7444993 2.2405789 1.55049106 1.43886350 1.30318840
Proportion of Variance 0.1711881 0.1140953 0.05463688 0.04705291 0.03859773
Cumulative Proportion  0.1711881 0.2852834 0.33992029 0.38697320 0.42557093
                           Comp.6     Comp.7     Comp.8     Comp.9    Comp.10
Standard deviation     1.18837867 1.15919129 1.07838912 1.07120568 1.04901318
Proportion of Variance 0.03209645 0.03053919 0.02643007 0.02607913 0.02500974
Cumulative Proportion  0.45766738 0.48820657 0.51463664 0.54071577 0.56572551
                          Comp.11    Comp.12    Comp.13    Comp.14   Comp.15
Standard deviation     1.03848656 1.00152287 0.97753974 0.95697572 0.9287543
Proportion of Variance 0.02451033 0.02279655 0.02171782 0.02081369 0.0196042
Cumulative Proportion  0.59023584 0.61303238 0.63475020 0.65556390 0.6751681
                          Comp.16    Comp.17   Comp.18    Comp.19    Comp.20
Standard deviation     0.92262649 0.90585705 0.8788668 0.86757525 0.84269120
Proportion of Variance 0.01934636 0.01864948 0.0175547 0.01710652 0.01613928
Cumulative Proportion  0.69451445 0.71316392 0.7307186 0.74782514 0.76396443
                          Comp.21    Comp.22    Comp.23    Comp.24    Comp.25
Standard deviation     0.83124925 0.80564654 0.78975423 0.78100835 0.77852606
Proportion of Variance 0.01570398 0.01475151 0.01417527 0.01386305 0.01377506
Cumulative Proportion  0.77966841 0.79441992 0.80859519 0.82245823 0.83623330
                          Comp.26    Comp.27    Comp.28    Comp.29    Comp.30
Standard deviation     0.74969868 0.74137885 0.72343693 0.71457305 0.70358645
Proportion of Variance 0.01277382 0.01249188 0.01189457 0.01160488 0.01125077
```

Comments

- Want overall fraction of variance explained ("cumulative proportion' ') to be reasonably high.
- 2 factors, 28.5%. Terrible!
- Even 56% (10 factors) not that good!
- Have to live with that.

Biplot

```r
ggbiplot(bem.pc, alpha = 0.3)
```



Comments

- Ignore individuals for now.
- Most variables point to 1 o'clock or 4 o'clock.
- Suggests factor analysis with rotation will get interpretable factors (rotate to 12 o'clock and 3 o'clock, for example).
- Try for 2-factor solution (rough interpretation, will be bad):

```
bem %>%
select(-subno) %>%
factanal(factors = 2) -> bem.2
```

- Show output in pieces (just print `bem.2` to see all of it).

Uniquenesses, sorted

```r
sort(bem.2$uniquenesses)
```

```
 leaderab   leadact      warm    tender  dominant    gentle
0.4091894 0.4166153 0.4764762 0.4928919 0.4942909 0.5064551
 forceful   strpers   compass     stand  undstand    assert
0.5631857 0.5679398 0.5937073 0.6024001 0.6194392 0.6329347
   soothe    affect    decide  selfsuff  sympathy     indpt
0.6596103 0.6616625 0.6938578 0.7210246 0.7231450 0.7282742
  helpful    defbel      risk   reliant   individ   compete
0.7598223 0.7748448 0.7789761 0.7808058 0.7941998 0.7942910
 conscien     happy  sensitiv     loyal  ambitiou       shy
0.7974820 0.8008966 0.8018851 0.8035264 0.8101599 0.8239496
 softspok  cheerful  masculin  yielding  feminine  truthful
0.8339058 0.8394916 0.8453368 0.8688473 0.8829927 0.8889983
  lovchil    analyt    athlet   flatter  gullible     moody
0.8924392 0.8968744 0.9229702 0.9409500 0.9583435 0.9730607
 childlik  foullang
0.9800360 0.9821662
```

Comments

- Mostly high or very high (bad).
- Some smaller, eg.: Leadership ability (0.409), Acts like leader (0.417), Warm (0.476), Tender (0.493).
- Smaller uniquenesses captured by one of our two factors.
- Larger uniquenesses are not: need more factors to capture them.

Factor loadings, some

```r
bem.2$loadings
```

```
Loadings:
         Factor1 Factor2
helpful   0.314   0.376
reliant   0.453   0.117
defbel    0.434   0.193
yielding -0.131   0.338
cheerful  0.152   0.371
indpt     0.521
athlet    0.267
shy      -0.414
assert    0.605
strpers   0.657
forceful  0.649  -0.126
affect    0.178   0.554
flatter           0.223
loyal     0.151   0.417
analyt    0.295   0.127
feminine  0.113   0.323
sympathy          0.526
moody            -0.162
sensitiv  0.135   0.424
undstand          0.610
compass   0.114   0.627
```

Making a data frame

There are too many to read easily, so make a data frame. A bit tricky:

```
bem.2$loadings %>%
unclass() %>%
as_tibble() %>%
mutate(trait = rownames(bem.2$loadings)) -> loadings
loadings %>% slice(1:8)
```

```
# A tibble: 8 x 3
  Factor1  Factor2 trait
    <dbl>    <dbl> <chr>
1   0.314  0.376   helpful
2   0.453  0.117   reliant
3   0.434  0.193   defbel
4  -0.131  0.338   yielding
5   0.152  0.371   cheerful
6   0.521  0.00587 indpt
7   0.267  0.0755  athlet
8  -0.414 -0.0654  shy
```

Pick out the big ones on factor 1

Arbitrarily defining > 0.4 or < -0.4 as "big":

```
loadings %>% filter(abs(Factor1) > 0.4)
```

```
# A tibble: 17 x 3
   Factor1  Factor2 trait
     <dbl>    <dbl> <chr>
 1   0.453  0.117   reliant
 2   0.434  0.193   defbel
 3   0.521  0.00587 indpt
 4  -0.414 -0.0654  shy
 5   0.605  0.0330  assert
 6   0.657  0.0208  strpers
 7   0.649 -0.126   forceful
 8   0.765  0.0695  leaderab
 9   0.442  0.161   risk
10   0.542  0.113   decide
11   0.511  0.134   selfsuff
12   0.668 -0.245   dominant
13   0.607  0.172   stand
14   0.763 -0.0407  leadact
15   0.445  0.0891  individ
16   0.450  0.0532  compete
17   0.414  0.137   ambitiou
```

Factor 2, the big ones

```
loadings %>% filter(abs(Factor2) > 0.4)
```

```
# A tibble: 11 x 3
   Factor1 Factor2 trait
     <dbl>   <dbl> <chr>
 1  0.178    0.554 affect
 2  0.151    0.417 loyal
 3  0.0230   0.526 sympathy
 4  0.135    0.424 sensitiv
 5  0.0911   0.610 undstand
 6  0.114    0.627 compass
 7  0.0606   0.580 soothe
 8  0.119    0.430 happy
 9  0.0796   0.719 warm
10  0.0511   0.710 tender
11 -0.0187   0.702 gentle
```

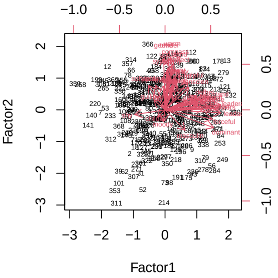
Plotting the two factors

- A bi-plot, this time with the variables reduced in size. Looking for unusual individuals.
- Have to run `factanal` again to get factor scores for plotting.

```
bem %>% select(-subno) %>%
factanal(factors = 2, scores = "r") -> bem.2a
biplot(bem.2a$scores, bem.2a$loadings, cex = c(0.5, 0.5))
```

- Numbers on plot are row numbers of `bem` data frame.

The (awful) biplot



Comments

- Variables mostly up ("feminine") and right ("masculine"), accomplished by rotation.
- Some unusual individuals: 311, 214 (low on factor 2), 366 (high on factor 2), 359, 258 (low on factor 1), 230 (high on factor 1).

Individual 366

```r
bem %>% slice(366) %>% glimpse()
```

```
Rows: 1
Columns: 45
$ subno    <dbl> 755
$ helpful  <dbl> 7
$ reliant  <dbl> 7
$ defbel   <dbl> 5
$ yielding <dbl> 7
$ cheerful <dbl> 7
$ indpt    <dbl> 7
$ athlet   <dbl> 7
$ shy      <dbl> 2
$ assert   <dbl> 1
$ strpers  <dbl> 3
$ forceful <dbl> 1
$ affect   <dbl> 7
$ flatter  <dbl> 9
$ loyal    <dbl> 7
$ analyt   <dbl> 7
$ feminine <dbl> 7
$ sympathy <dbl> 7
$ moody    <dbl> 1
$ sensitiv <dbl> 7
$ undstand <dbl> 7
$ compass  <dbl> 6
$ leaderab <dbl> 3
$ soothe   <dbl> 7
$ risk     <dbl> 7
$ decide   <dbl> 7
$ selfsuff <dbl> 7
$ conscien <dbl> 7
$ dominant <dbl> 1
$ masculin <dbl> 1
$ stand    <dbl> 7
```

Comments

- Individual 366 high on factor 2, but hard to see which traits should have high scores (unless we remember).
- Idea 1: use percentile ranks as before.
- Idea 2: Rating scale is easy to interpret. So *tidy* original data frame to make easier to look things up.

Tidying original data

```r
bem %>%
ungroup() %>%
mutate(row = row_number()) %>%
pivot_longer(c(-subno, -row), names_to="trait",
             values_to="score") -> bem_tidy
bem_tidy
```

```
# A tibble: 16,236 x 4
   subno   row trait    score
   <dbl> <int> <chr>    <dbl>
 1     1     1 helpful      7
 2     1     1 reliant      7
 3     1     1 defbel       5
 4     1     1 yielding     5
 5     1     1 cheerful     7
 6     1     1 indpt        7
 7     1     1 athlet       7
 8     1     1 shy          1
 9     1     1 assert       7
10     1     1 strpers      7
# i 16,226 more rows
```

Recall data frame of loadings

```
loadings %>% slice(1:10)
```

```
# A tibble: 10 x 3
   Factor1  Factor2 trait
     <dbl>    <dbl> <chr>
 1   0.314  0.376   helpful
 2   0.453  0.117   reliant
 3   0.434  0.193   defbel
 4  -0.131  0.338   yielding
 5   0.152  0.371   cheerful
 6   0.521  0.00587 indpt
 7   0.267  0.0755  athlet
 8  -0.414 -0.0654  shy
 9   0.605  0.0330  assert
10   0.657  0.0208  strpers
```

Want to add the factor scores for each trait to our tidy data frame `bem_tidy`. This is a left-join (over), matching on the column `trait` that is in both data frames (thus, the default):

Looking up loadings

```
bem_tidy %>% left_join(loadings) -> bem_tidy
bem_tidy %>% sample_n(12)
```

```
# A tibble: 12 x 6
   subno   row trait    score  Factor1  Factor2
   <dbl> <int> <chr>    <dbl>    <dbl>    <dbl>
 1   555   324 compass      5  0.114    0.627
 2   142    91 cheerful     6  0.152    0.371
 3   529   308 indpt        7  0.521    0.00587
 4   460   259 truthful     7  0.109    0.315
 5   232   130 flatter      1  0.0964   0.223
 6   104    64 yielding     1 -0.131    0.338
 7    83    49 sympathy     7  0.0230   0.526
 8    28    18 sympathy     6  0.0230   0.526
 9   592   348 foullang     5 -0.00493  0.133
10    38    27 yielding     4 -0.131    0.338
11    29    19 decide       4  0.542    0.113
12    66    37 leadact      5  0.763   -0.0407
```

Individual 366, high on Factor 2

So now pick out the rows of the tidy data frame that belong to individual 366 (`row=366`) and for which the `Factor2` score exceeds 0.4 in absolute value (our "big" from before):

```
bem_tidy %>% filter(row == 366, abs(Factor2) > 0.4)
```

```
# A tibble: 11 x 6
   subno   row trait    score Factor1 Factor2
   <dbl> <int> <chr>    <dbl>   <dbl>   <dbl>
 1   755   366 affect       7  0.178    0.554
 2   755   366 loyal        7  0.151    0.417
 3   755   366 sympathy     7  0.0230   0.526
 4   755   366 sensitiv     7  0.135    0.424
 5   755   366 undstand     7  0.0911   0.610
 6   755   366 compass      6  0.114    0.627
 7   755   366 soothe       7  0.0606   0.580
 8   755   366 happy        7  0.119    0.430
 9   755   366 warm         7  0.0796   0.719
10   755   366 tender       7  0.0511   0.710
11   755   366 gentle       7 -0.0187   0.702
```

As expected, high scorer on these.

Several individuals

Rows 311 and 214 were *low* on Factor 2, so their scores should be low. Can we do them all at once?

```
bem_tidy %>% filter(
row %in% c(366, 311, 214),
abs(Factor2) > 0.4
)
```

```
# A tibble: 33 x 6
   subno   row trait    score Factor1 Factor2
   <dbl> <int> <chr>    <dbl>   <dbl>   <dbl>
 1   369   214 affect       1  0.178    0.554
 2   369   214 loyal        7  0.151    0.417
 3   369   214 sympathy     4  0.0230   0.526
 4   369   214 sensitiv     7  0.135    0.424
 5   369   214 undstand     5  0.0911   0.610
 6   369   214 compass      5  0.114    0.627
 7   369   214 soothe       3  0.0606   0.580
 8   369   214 happy        4  0.119    0.430
 9   369   214 warm         1  0.0796   0.719
10   369   214 tender       3  0.0511   0.710
# i 23 more rows
```

Can we display each individual in own column?

Individual by column

Un-tidy, that is, `pivot_wider`:

```
bem_tidy %>%
filter(
  row %in% c(366, 311, 214),
  abs(Factor2) > 0.4
) %>%
select(-subno, -Factor1, -Factor2) %>%
pivot_wider(names_from=row, values_from=score)
```

```
# A tibble: 11 x 4
   trait    `214` `311` `366`
   <chr>    <dbl> <dbl> <dbl>
 1 affect       1     5     7
 2 loyal        7     4     7
 3 sympathy     4     4     7
 4 sensitiv     7     4     7
 5 undstand     5     3     7
 6 compass      5     4     6
 7 soothe       3     4     7
 8 happy        4     3     7
 9 warm         1     3     7
10 tender       3     4     7
11 gentle       2     3     7
```

366 high, 311 middling, 214 (sometimes) low.

Individuals 230, 258, 359

These were high, low, low on factor 1. Adapt code:

```
bem_tidy %>%
filter(row %in% c(359, 258, 230), abs(Factor1) > 0.4) %>%
select(-subno, -Factor1, -Factor2) %>%
pivot_wider(names_from=row, values_from=score)
```

```
# A tibble: 17 x 4
   trait    `230` `258` `359`
   <chr>    <dbl> <dbl> <dbl>
 1 reliant      7     4     1
 2 defbel       7     1     1
 3 indpt        7     7     1
 4 shy          2     7     5
 5 assert       7     3     1
 6 strpers      7     1     3
 7 forceful     7     1     1
 8 leaderab     7     1     1
 9 risk         7     5     7
10 decide       7     1     2
11 selfsuff     7     4     1
12 dominant     7     1     1
13 stand        7     1     6
14 leadact      7     1     1
15 individ      7     3     3
16 compete      6     2     1
17 ambitiou     7     2     4
```

Is 2 factors enough?

Suspect not:

```
bem.2$PVAL
```

```
    objective
1.458183e-150
```

2 factors resoundingly rejected. Need more. Have to go all the way to 15 factors to not reject:

```
bem %>%
select(-subno) %>%
factanal(factors = 15) -> bem.15
bem.15$PVAL
```

```
objective
 0.132617
```

Even then, only just over 50% of variability explained.

What's important in 15 factors?

- Let's take a look at the important things in those 15 factors.
- Get 15-factor loadings into a data frame, as before:

```
bem.15$loadings %>%
unclass() %>%
as_tibble() %>%
mutate(trait = rownames(bem.15$loadings)) -> loadings
```

- then show the highest few loadings on each factor.

Factor 1 (of 15)

```
loadings %>%
arrange(desc(abs(Factor1))) %>%
select(Factor1, trait) %>%
slice(1:10)
```

```
# A tibble: 10 x 2
   Factor1 trait
     <dbl> <chr>
 1   0.813 compass
 2   0.676 undstand
 3   0.661 sympathy
 4   0.641 sensitiv
 5   0.597 soothe
 6   0.348 warm
 7   0.280 gentle
 8   0.279 tender
 9   0.250 helpful
10   0.234 conscien
```

Compassionate, understanding, sympathetic, soothing: thoughtful of others.

Factor 2

```
loadings %>%
arrange(desc(abs(Factor2))) %>%
select(Factor2, trait) %>%
slice(1:10)
```

```
# A tibble: 10 x 2
   Factor2 trait
     <dbl> <chr>
 1   0.762 strpers
 2   0.716 forceful
 3   0.698 assert
 4   0.504 dominant
 5   0.393 leaderab
 6   0.367 stand
 7   0.351 leadact
 8  -0.313 softspok
 9  -0.287 shy
10   0.260 analyt
```

Strong personality, forceful, assertive, dominant: getting ahead.

Factor 3

```
loadings %>%
arrange(desc(abs(Factor3))) %>%
select(Factor3, trait) %>%
slice(1:10)
```

```
# A tibble: 10 x 2
   Factor3 trait
     <dbl> <chr>
 1   0.670 reliant
 2   0.648 selfsuff
 3   0.620 indpt
 4   0.390 helpful
 5  -0.339 gullible
 6   0.333 individ
 7   0.332 decide
 8   0.329 conscien
 9   0.288 leaderab
10   0.280 defbel
```

Self-reliant, self-sufficient, independent: going it alone.

Factor 4

```
loadings %>%
arrange(desc(abs(Factor4))) %>%
select(Factor4, trait) %>%
slice(1:10)
```

```
# A tibble: 10 x 2
   Factor4 trait
     <dbl> <chr>
 1   0.696 gentle
 2   0.692 tender
 3   0.599 warm
 4   0.447 affect
 5   0.394 softspok
 6   0.278 lovchil
 7   0.244 undstand
 8   0.244 happy
 9   0.213 loyal
10   0.202 soothe
```

Gentle, tender, warm (affectionate): caring for others.

Factor 5

```
loadings %>%
arrange(desc(abs(Factor5))) %>%
select(Factor5, trait) %>%
slice(1:10)
```

```
# A tibble: 10 x 2
   Factor5 trait
     <dbl> <chr>
 1   0.696 compete
 2   0.674 ambitiou
 3   0.345 risk
 4   0.342 individ
 5   0.281 athlet
 6   0.270 leaderab
 7   0.245 decide
 8   0.206 dominant
 9   0.193 leadact
10   0.185 strpers
```

Ambitious, competitive (with a bit of risk-taking and individualism): Being the best.

Factor 6

```
loadings %>%
arrange(desc(abs(Factor6))) %>%
select(Factor6, trait) %>%
slice(1:10)
```

```
# A tibble: 10 x 2
   Factor6 trait
     <dbl> <chr>
 1   0.868 leadact
 2   0.608 leaderab
 3   0.338 dominant
 4   0.201 forceful
 5  -0.192 shy
 6   0.179 risk
 7   0.170 masculin
 8   0.164 decide
 9   0.159 compete
10   0.147 athlet
```

Acts like a leader, leadership ability (with a bit of Dominant): Taking charge.

Factor 7

```
loadings %>%
arrange(desc(abs(Factor7))) %>%
select(Factor7, trait) %>%
slice(1:10)
```

```
# A tibble: 10 x 2
   Factor7 trait
     <dbl> <chr>
 1   0.670 happy
 2   0.667 cheerful
 3  -0.522 moody
 4   0.219 athlet
 5   0.213 warm
 6   0.172 gentle
 7  -0.164 masculin
 8   0.160 reliant
 9   0.147 yielding
10   0.141 lovchil
```

Happy and cheerful.

Factor 8

```
loadings %>%
arrange(desc(abs(Factor8))) %>%
select(Factor8, trait) %>%
slice(1:10)
```

```
# A tibble: 10 x 2
   Factor8 trait
     <dbl> <chr>
 1   0.630 affect
 2   0.516 flatter
 3  -0.251 softspok
 4   0.221 warm
 5   0.188 tender
 6   0.185 strpers
 7  -0.180 shy
 8   0.180 compete
 9   0.166 loyal
10   0.155 helpful
```

Affectionate, flattering: Making others feel good.

Factor 9

```
loadings %>%
arrange(desc(abs(Factor9))) %>%
select(Factor9, trait) %>%
slice(1:10)
```

```
# A tibble: 10 x 2
   Factor9 trait
     <dbl> <chr>
 1   0.863 stand
 2   0.340 defbel
 3   0.245 individ
 4   0.194 risk
 5  -0.172 shy
 6   0.171 decide
 7   0.120 assert
 8   0.116 conscien
 9   0.112 analyt
10  -0.112 gullible
```

Taking a stand.

Factor 10

```
loadings %>%
arrange(desc(abs(Factor10))) %>%
select(Factor10, trait) %>%
slice(1:10)
```

```
# A tibble: 10 x 2
   Factor10 trait
      <dbl> <chr>
 1    0.808 feminine
 2   -0.264 masculin
 3    0.245 softspok
 4    0.232 conscien
 5    0.202 selfsuff
 6    0.176 yielding
 7    0.141 gentle
 8    0.113 flatter
 9    0.109 decide
10   -0.0941 lovchil
```

Feminine. (A little bit of not-masculine!)

Factor 11

```
loadings %>%
arrange(desc(abs(Factor11))) %>%
select(Factor11, trait) %>%
slice(1:10)
```

```
# A tibble: 10 x 2
   Factor11 trait
      <dbl> <chr>
 1   0.916  loyal
 2   0.189  affect
 3   0.159  truthful
 4   0.125  helpful
 5   0.104  analyt
 6   0.101  tender
 7   0.0972 lovchil
 8   0.0964 gullible
 9   0.0935 cheerful
10   0.0821 conscien
```

Loyal.

Factor 12

```
loadings %>%
arrange(desc(abs(Factor12))) %>%
select(Factor12, trait) %>%
slice(1:10)
```

```
# A tibble: 10 x 2
   Factor12 trait
      <dbl> <chr>
 1    0.611 childlik
 2   -0.285 selfsuff
 3   -0.279 conscien
 4    0.259 moody
 5    0.201 shy
 6   -0.167 decide
 7    0.154 masculin
 8    0.146 dominant
 9    0.138 compass
10   -0.130 leaderab
```

Childlike. (With a bit of moody, shy, not-self-sufficient, not-conscientious.)

Factor 13

```
loadings %>%
arrange(desc(abs(Factor13))) %>%
select(Factor13, trait) %>%
slice(1:10)
```

```
# A tibble: 10 x 2
   Factor13 trait
      <dbl> <chr>
 1    0.573 truthful
 2   -0.278 gullible
 3    0.263 happy
 4    0.189 warm
 5   -0.167 shy
 6    0.165 loyal
 7   -0.144 yielding
 8   -0.130 assert
 9    0.114 defbel
10   -0.111 lovchil
```

Truthful. (With a bit of happy and not-gullible.)

Factor 14

```
loadings %>%
arrange(desc(abs(Factor14))) %>%
select(Factor14, trait) %>%
slice(1:10)
```

```
# A tibble: 10 x 2
   Factor14 trait
      <dbl> <chr>
 1    0.443 decide
 2    0.237 selfsuff
 3    0.195 forceful
 4   -0.186 softspok
 5    0.160 risk
 6   -0.148 strpers
 7    0.146 dominant
 8    0.128 happy
 9    0.115 compass
10    0.105 masculin
```

Decisive. (With a bit of self-sufficient and not-soft-spoken.)

Factor 15

```
loadings %>%
arrange(desc(abs(Factor15))) %>%
select(Factor15, trait) %>%
slice(1:10)
```

```
# A tibble: 10 x 2
   Factor15 trait
      <dbl> <chr>
 1   -0.324 compass
 2    0.247 athlet
 3    0.229 sensitiv
 4    0.199 risk
 5   -0.164 affect
 6    0.163 moody
 7   -0.112 individ
 8    0.110 warm
 9    0.105 cheerful
10    0.101 reliant
```

Not-compassionate, athletic, sensitive: A mixed bag. (“Cares about self”?)

Anything left out? Uniquenesses

```
enframe(bem.15$uniquenesses, name="quality", value="uniq") %>%
  slice_max(uniq, n = 10)
```

```
# A tibble: 10 x 2
   quality   uniq
   <chr>    <dbl>
 1 foullang 0.914
 2 lovchil  0.824
 3 analyt   0.812
 4 yielding 0.791
 5 masculin 0.723
 6 athlet   0.722
 7 shy      0.703
 8 gullible 0.700
 9 flatter  0.663
10 helpful  0.652
```

Uses foul language especially, also loves children and analytical. So could use even more factors.