Week 6, Lecture 11 Advanced statistical methods, part II: Structural equation modeling, social network analysis, and geospatial analyses

Richard E.W. Berl

Spring 2019

Structural equation modeling

One common use of structural equation modeling (SEM) is confirmatory factor analysis (CFA), which allows us to test how well our data fit in the factors that we think they should in our model. Naturally, a good way to find this model in the first place is through EFA, and doing this process all together in an SEM framework is sometimes called E/CFA. There are other types of SEM analyses, including path analysis. Here we'll focus on EFA and CFA, but the methods and tools are similar across other types of SEM.

We'll use the items from the "Nerdy Personality Attributes Scale" to build and evaluate our model. Download the data here: https://openpsychometrics.org/_rawdata/

On the right side, click "NPAS-data-16December2018.zip" to download the ZIP file with the data and codebook. More info is available on the page for the scale itself (and the link below on its development): https://openpsychometrics.org/tests/NPAS/

After extracting the CSV file, place it in your `/data` folder and rename it to `npas.txt` (since it's actually tab delimited, not comma delimited).

Then load it in, using the `readr` package due to a few messed up rows:

```
library(readr)
```

```
nerd = as.data.frame(read_tsv("./data/npas.txt"))
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   country = col_character(),
##   major = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
## Warning: 14 parsing failures.
##  row   col           expected actual              file
## 2290 major delimiter or quote      , './data/npas.txt'
## 2290 major delimiter or quote      h './data/npas.txt'
## 2405 major delimiter or quote      , './data/npas.txt'
## 2405 major delimiter or quote      b './data/npas.txt'
## 4267 major delimiter or quote        './data/npas.txt'
## .... ..... ................. ...... ................
## See problems(...) for more details.
```

```r
head(nerd)
```

```
##   Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 Q9 Q10 Q11 Q12 Q13 Q14 Q15 Q16 Q17 Q18 Q19 Q20
## 1  3  5  3  3  5  5  5  3  5   5   4   5   5   5   3   5   4   5   5   5
## 2  4  4  4  3  5  2  5  1  4   4   1   5   4   4   1   3   3   3   1   3
## 3  5  5  5  5  5  5  5  5  5   5   5   4   5   5   4   5   5   4   5   5
## 4  5  5  5  5  5  5  5  3  5   5   5   5   4   4   4   2   5   5   5   5
## 5  4  4  4  4  4  4  4  4  4   5   4   4   5   5   1   1   5   5   3   5
## 6  4  4  4  4  4  4  5  3  3   4   4   4   3   5   3   4   3   4   5   4
##   Q21 Q22 Q23 Q24 Q25 Q26 country introelapse testelapse surveyelapse
## 1   5   5   5   5   5   5      US          41         93          125
## 2   3   3   4   4   4   5      GB          13        131          161
## 3   5   5   5   5   3   5      PL           2         49           87
## 4   4   1   5   5   5   5      US          26        710          228
## 5   5   4   4   5   4   0      US           6        195          424
## 6   4   2   5   5   3   4      US         410        117          113
##   TIPI1 TIPI2 TIPI3 TIPI4 TIPI5 TIPI6 TIPI7 TIPI8 TIPI9 TIPI10 VCL1 VCL2
## 1     1     1     7     1     6     7     7     1     7      1    1    1
## 2     5     3     5     4     5     5     5     5     6      1    1    1
## 3     1     6     3     7     6     7     2     6     2      1    1    1
## 4     7     2     7     4     7     2     6     1     7      1    1    1
## 5     2     5     4     3     7     6     5     5     6      2    1    1
## 6     2     3     5     5     5     5     4     3     4      4    1    1
##   VCL3 VCL4 VCL5 VCL6 VCL7 VCL8 VCL9 VCL10 VCL11 VCL12 VCL13 VCL14 VCL15
## 1    1    1    1    0    1    1    0     1     1     0     1     1     1
## 2    1    1    1    0    0    1    0     1     1     0     1     1     1
## 3    1    1    1    1    0    0    0     1     1     1     1     1     1
## 4    1    1    1    0    1    1    0     1     1     0     1     1     1
## 5    0    1    1    0    0    0    0     1     0     0     1     1     1
## 6    1    1    1    1    0    0    0     1     0     1     1     1     1
##   VCL16 education urban gender engnat age screenw screenh hand religion
## 1     1         3     2      2      1  69    2520    1418    1        6
## 2     1         4     2      2      1  50     854     534    1        1
## 3     1         3     1      2      2  22    1366     768    1        1
## 4     1         4     3      1      1  44     768    1024    1        2
## 5     1         1     1      2      1  17    1366     768    1        7
## 6     1         2     3      1      1  18    1366     768    1        1
##   orientation voted married familysize       major race_arab race_asian
## 1           1     1       3          4  Studio Art         0          0
## 2           1     1       1          3  biophysics         0          0
## 3           2     1       1          2     biology         0          0
## 4           1     2       3          4 Mathematics         0          0
## 5           4     2       1          1        <NA>         0          0
## 6           2     2       1          3     Geology         0          0
##   race_black race_white race_hispanic race_nativeam race_nativeau
## 1          0          1             0             0             0
## 2          0          1             0             0             0
## 3          0          1             0             0             0
## 4          0          1             0             0             0
## 5          0          1             0             0             0
## 6          0          1             0             0             0
##   race_other nerdy ASD
## 1          0     7   2
## 2          0     6   2
## 3          0     7   2
## 4          0     7   2
## 5          0     6   2
## 6          0     5   2
```

```
str(nerd, give.attr=F)
```

```
## 'data.frame':    19749 obs. of  80 variables:
##  $ Q1          : num  3 4 5 5 4 4 5 5 5 4 ...
##  $ Q2          : num  5 4 5 5 4 4 3 3 5 5 ...
##  $ Q3          : num  3 4 5 5 4 4 4 4 5 5 ...
##  $ Q4          : num  3 3 5 5 4 4 3 5 4 5 ...
##  $ Q5          : num  5 5 5 5 4 4 5 5 2 4 ...
##  $ Q6          : num  5 2 5 5 4 4 4 3 3 3 ...
##  $ Q7          : num  5 5 5 5 4 5 4 3 0 5 ...
##  $ Q8          : num  3 1 5 3 4 3 5 5 5 4 ...
##  $ Q9          : num  5 4 5 5 4 3 5 3 4 4 ...
##  $ Q10         : num  5 4 5 5 5 4 5 5 4 5 ...
##  $ Q11         : num  4 1 5 5 4 4 3 4 4 1 ...
##  $ Q12         : num  5 5 4 5 4 4 3 4 4 4 ...
##  $ Q13         : num  5 4 5 4 5 3 4 3 5 4 ...
##  $ Q14         : num  5 4 5 4 5 5 4 5 3 5 ...
##  $ Q15         : num  3 1 4 4 1 3 1 3 5 5 ...
##  $ Q16         : num  5 3 5 2 1 4 1 5 4 4 ...
##  $ Q17         : num  4 3 5 5 5 3 3 4 5 5 ...
##  $ Q18         : num  5 3 4 5 5 4 1 4 5 5 ...
##  $ Q19         : num  5 1 5 5 3 5 3 3 4 2 ...
##  $ Q20         : num  5 3 5 5 5 4 1 3 3 5 ...
##  $ Q21         : num  5 3 5 4 5 4 5 4 5 2 ...
##  $ Q22         : num  5 3 5 1 4 2 1 4 2 3 ...
##  $ Q23         : num  5 4 5 5 4 5 4 3 5 5 ...
##  $ Q24         : num  5 4 5 5 5 5 5 5 4 5 ...
##  $ Q25         : num  5 4 3 5 4 3 3 2 2 2 ...
##  $ Q26         : num  5 5 5 5 0 4 5 5 3 5 ...
##  $ country     : chr  "US" "GB" "PL" "US" ...
##  $ introelapse : num  41 13 2 26 6 410 7 418 37 144 ...
##  $ testelapse  : num  93 131 49 710 195 117 107 64 114 112 ...
##  $ surveyelapse : num  125 161 87 228 424 113 145 97 568 128 ...
##  $ TIPI1       : num  1 5 1 7 2 2 3 3 1 5 ...
##  $ TIPI2       : num  1 3 6 2 5 3 1 6 1 6 ...
##  $ TIPI3       : num  7 5 3 7 4 5 2 5 5 3 ...
##  $ TIPI4       : num  1 4 7 4 3 5 1 7 5 6 ...
##  $ TIPI5       : num  6 5 6 7 7 5 7 6 5 6 ...
##  $ TIPI6       : num  7 5 7 2 6 5 5 5 7 7 ...
##  $ TIPI7       : num  7 5 2 6 5 4 5 6 7 4 ...
##  $ TIPI8       : num  1 5 6 1 5 3 5 7 7 7 ...
##  $ TIPI9       : num  7 6 2 7 6 4 6 1 3 5 ...
##  $ TIPI10      : num  1 1 1 1 2 4 1 4 1 2 ...
##  $ VCL1        : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ VCL2        : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ VCL3        : num  1 1 1 1 0 1 1 1 0 1 ...
##  $ VCL4        : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ VCL5        : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ VCL6          : num  0 0 1 0 0 1 0 0 0 0 ...
##  $ VCL7          : num  1 0 0 1 0 0 0 0 0 0 ...
##  $ VCL8          : num  1 1 0 1 0 0 1 0 0 1 ...
##  $ VCL9          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ VCL10         : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ VCL11         : num  1 1 1 1 0 0 0 0 0 1 ...
##  $ VCL12         : num  0 0 1 0 0 1 0 0 0 0 ...
##  $ VCL13         : num  1 1 1 1 1 1 0 1 1 1 ...
##  $ VCL14         : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ VCL15         : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ VCL16         : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ education     : num  3 4 3 4 1 2 2 2 3 1 ...
##  $ urban         : num  2 2 1 3 1 3 2 2 2 2 ...
##  $ gender        : num  2 2 2 1 2 1 1 2 2 3 ...
##  $ engnat        : num  1 1 2 1 1 1 1 1 1 2 ...
##  $ age           : num  69 50 22 44 17 18 18 21 25 17 ...
##  $ screenw       : num  2520 854 1366 768 1366 ...
##  $ screenh       : num  1418 534 768 1024 768 ...
##  $ hand          : num  1 1 1 1 1 1 2 1 1 1 ...
##  $ religion      : num  6 1 1 2 7 1 1 12 7 1 ...
##  $ orientation   : num  1 1 2 1 4 2 1 5 1 2 ...
##  $ voted         : num  1 1 1 2 2 2 2 1 2 2 ...
##  $ married       : num  3 1 1 3 1 1 1 1 1 1 ...
##  $ familysize    : num  4 3 2 4 1 3 2 3 4 3 ...
##  $ major         : chr  "Studio Art" "biophysics" "biology" "Mathematics" ...
##  $ race_arab     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ race_asian    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ race_black    : num  0 0 0 0 0 0 1 0 1 0 ...
##  $ race_white    : num  1 1 1 1 1 1 0 1 0 0 ...
##  $ race_hispanic: num  0 0 0 0 0 0 0 0 0 1 ...
##  $ race_nativeam: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ race_nativeau: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ race_other    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ nerdy         : num  7 6 7 7 6 5 6 5 7 7 ...
##  $ ASD           : num  2 2 2 2 2 2 2 2 2 2 ...
```

We can see that, along with the "nerdiness" questions, there are a number of other scale items and demographic variables represented. Let's change each of the Likert variables (and education) to ordered, using `lapply()`.

```
colnames(nerd)
```

```
##  [1] "Q1"           "Q2"           "Q3"           "Q4"
##  [5] "Q5"           "Q6"           "Q7"           "Q8"
##  [9] "Q9"           "Q10"          "Q11"          "Q12"
## [13] "Q13"          "Q14"          "Q15"          "Q16"
## [17] "Q17"          "Q18"          "Q19"          "Q20"
## [21] "Q21"          "Q22"          "Q23"          "Q24"
## [25] "Q25"          "Q26"          "country"      "introelapse"
## [29] "testelapse"   "surveyelapse" "TIPI1"        "TIPI2"
## [33] "TIPI3"        "TIPI4"        "TIPI5"        "TIPI6"
## [37] "TIPI7"        "TIPI8"        "TIPI9"        "TIPI10"
## [41] "VCL1"         "VCL2"         "VCL3"         "VCL4"
## [45] "VCL5"         "VCL6"         "VCL7"         "VCL8"
## [49] "VCL9"         "VCL10"        "VCL11"        "VCL12"
## [53] "VCL13"        "VCL14"        "VCL15"        "VCL16"
## [57] "education"     "urban"         "gender"        "engnat"
## [61] "age"           "screenw"       "screenh"       "hand"
## [65] "religion"      "orientation"   "voted"         "married"
## [69] "familysize"    "major"         "race_arab"     "race_asian"
## [73] "race_black"    "race_white"    "race_hispanic" "race_nativeam"
## [77] "race_nativeau" "race_other"    "nerdy"         "ASD"
```

```
nerd[,c(1:26,31:40,57,79)] = lapply(nerd[,c(1:26,31:40,57,79)], function(x) ordered(x))
```

Exploratory factor analysis

First we'll load up the packages we'll need: `psych` for parallel analysis to determine our number of factors, and `lavaan` and `semTools` for the EFA and CFA.

```
library(psych)
```

```
library(lavaan)
```

```
## This is lavaan 0.5-23.1097

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

##
## Attaching package: 'lavaan'

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

```
install.packages("semTools")
```

```
library(semTools)
```

```
##

## ###############################################################################

## This is semTools 0.4-14

## All users of R (or SEM) are invited to submit functions or ideas for functions.

## ###############################################################################

##
## Attaching package: 'semTools'

## The following object is masked from 'package:psych':
##
##     skew

## The following object is masked from 'package:readr':
##
##     clipboard
```

One important point in the E/CFA process is that we can't build our model and evaluate our model using the same data. If we did, we would risk building a model that is only valid for that particular data set. The best way to counteract this would be to collect two separate data sets and use one for EFA an the other for CFA. Since here we only have one data set to work with, we'll do the next best thing, which is to split it randomly into two sets that we use for the two steps of E/CFA. Following convention in machine learning, in which this kind of splitting is common for training and validating predictive models, we'll call them the "train" and "test" sets, respectively.

To split the data, we'll use the `sample()` function to randomly select half of the rows in the data frame (rounded to an even digit). We use those rows are our train set, and the rest of the rows for our test set. Before sampling, we set the seed for the random number generator in R, so that if someone else goes to run our code they will get the same result. I tend to use the current date as my seed, but you could use any number.

```
set.seed(20190430)
trainSet = sample(nrow(nerd), round(nrow(nerd)/2))
nerdTrain = nerd[trainSet,]
nerdTest = nerd[-trainSet,]
head(nerdTrain)
```

```
##       Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 Q9 Q10 Q11 Q12 Q13 Q14 Q15 Q16 Q17 Q18 Q19
## 12801  1  5  2  4  4  4  5  4  4   1   1   5   2   4   1   1   1   5   1
## 178    2  5  2  2  4  3  5  5  2   4   5   5   4   4   3   3   3   2   5
## 15165  4  4  5  5  4  4  5  3  4   3   1   4   2   4   1   1   4   5   2
## 7646   5  4  5  5  5  4  4  4  4   5   1   4   3   3   4   3   4   5   1
## 3162   5  1  5  5  4  4  3  1  5   4   1   3   3   5   1   5   5   1   1
## 10896  4  5  4  2  1  5  4  4  4   4   3   4   4   4   4   2   3   5   3
##       Q20 Q21 Q22 Q23 Q24 Q25 Q26 country introelapse testelapse
## 12801   4   1   1   1   4   1   5      IE        1103         83
## 178     3   4   2   2   3   3   3      CA          12         84
## 15165   4   1   2   1   4   3   5      US          21        131
## 7646    4   5   3   4   5   2   4    NONE          64        173
## 3162    4   3   5   5   5   1   2      US           5        107
## 10896   3   2   3   5   4   4   4      US          11        100
##       surveyelapse TIPI1 TIPI2 TIPI3 TIPI4 TIPI5 TIPI6 TIPI7 TIPI8 TIPI9
## 12801           80     7     1     5     1     6     3     7     5     7
## 178            133     4     5     6     3     5     5     3     5     6
## 15165          213     3     5     2     6     2     4     3     7     5
## 7646           273     6     5     6     6     5     5     4     5     3
## 3162           126     1     7     2     7     6     6     1     7     1
## 10896          131     5     5     6     3     7     5     6     6     5
##       TIPI10 VCL1 VCL2 VCL3 VCL4 VCL5 VCL6 VCL7 VCL8 VCL9 VCL10 VCL11
## 12801      2    1    1    1    1    1    0    1    1    1     1     0
## 178        5    1    1    0    1    1    0    0    0    0     1     0
## 15165      2    1    1    1    1    1    0    1    1    0     1     0
## 7646       2    1    1    1    0    1    0    0    0    0     1     0
## 3162       2    1    1    0    1    1    0    0    1    0     1     0
## 10896      2    1    1    1    1    1    0    0    0    0     1     0
##       VCL12 VCL13 VCL14 VCL15 VCL16 education urban gender engnat age
## 12801     0     1     1     1     1         4     1      2      1  28
## 178       0     1     1     1     1         2     1      1      1  17
## 15165     0     1     1     1     1         4     3      1      1  52
## 7646      0     1     1     1     1         1     1      2      1  18
## 3162      0     0     1     1     1         1     1      2      1  15
## 10896     0     1     1     1     1         2     2      2      1  15
##       screenw screenh hand religion orientation voted married familysize
## 12801    1280    1024    1        2           1     1       1          4
## 178      1920    1080    1        2           1     2       1          3
## 15165     360     640    2        9           1     1       2          2
## 7646      320     534    3        5           1     2       1          6
## 3162     1366     768    1        2           1     2       1          1
## 10896     375     667    1        6           3     2       1          2
##         major race_arab race_asian race_black race_white race_hispanic
## 12801 science          0          0          0          1            0
## 178      <NA>          0          0          0          1            0
## 15165 Geology          0          0          0          1            0
## 7646     <NA>          0          0          0          1            0
## 3162     <NA>          0          0          0          1            0
## 10896    <NA>          0          0          0          1            0
##       race_nativeam race_nativeau race_other nerdy ASD
## 12801             0             0          0     6   2
## 178               0             0          0     3   2
## 15165             0             0          0     5   2
## 7646              0             0          0     5   2
## 3162              0             0          0     2   2
## 10896             0             0          0     6   2
```

```
head(nerdTest)
```

```
##   Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 Q9 Q10 Q11 Q12 Q13 Q14 Q15 Q16 Q17 Q18 Q19 Q20
## 1  3  5  3  3  5  5  5  3  5   5   4   5   5   5   3   5   4   5   5   5
## 2  4  4  4  3  5  2  5  1  4   4   1   5   4   4   1   3   3   3   1   3
## 3  5  5  5  5  5  5  5  5  5   5   5   4   5   5   4   5   5   4   5   5
## 4  5  5  5  5  5  5  5  3  5   5   5   5   4   4   4   2   5   5   5   5
## 6  4  4  4  4  4  4  5  3  3   4   4   4   3   5   3   4   3   4   5   4
## 7  5  3  4  3  5  4  4  5  5   5   3   3   4   4   1   1   3   1   3   1
##   Q21 Q22 Q23 Q24 Q25 Q26 country introelapse testelapse surveyelapse
## 1   5   5   5   5   5   5      US          41         93          125
## 2   3   3   4   4   4   5      GB          13        131          161
## 3   5   5   5   5   3   5      PL           2         49           87
## 4   4   1   5   5   5   5      US          26        710          228
## 6   4   2   5   5   3   4      US         410        117          113
## 7   5   1   4   5   3   5      US           7        107          145
##   TIPI1 TIPI2 TIPI3 TIPI4 TIPI5 TIPI6 TIPI7 TIPI8 TIPI9 TIPI10 VCL1 VCL2
## 1     1     1     7     1     6     7     7     1     7      1    1    1
## 2     5     3     5     4     5     5     5     5     6      1    1    1
## 3     1     6     3     7     6     7     2     6     2      1    1    1
## 4     7     2     7     4     7     2     6     1     7      1    1    1
## 6     2     3     5     5     5     5     4     3     4      4    1    1
## 7     3     1     2     1     7     5     5     5     6      1    1    1
##   VCL3 VCL4 VCL5 VCL6 VCL7 VCL8 VCL9 VCL10 VCL11 VCL12 VCL13 VCL14 VCL15
## 1    1    1    1    0    1    1    0     1     1     0     1     1     1
## 2    1    1    1    0    0    1    0     1     1     0     1     1     1
## 3    1    1    1    1    0    0    0     1     1     1     1     1     1
## 4    1    1    1    0    1    1    0     1     1     0     1     1     1
## 6    1    1    1    1    0    0    0     1     0     1     1     1     1
## 7    1    1    1    0    0    1    0     1     0     0     0     1     1
##   VCL16 education urban gender engnat age screenw screenh hand religion
## 1     1         3     2      2      1  69    2520    1418    1        6
## 2     1         4     2      2      1  50     854     534    1        1
## 3     1         3     1      2      2  22    1366     768    1        1
## 4     1         4     3      1      1  44     768    1024    1        2
## 6     1         2     3      1      1  18    1366     768    1        1
## 7     1         2     2      1      1  18    1280     720    2        1
##   orientation voted married familysize       major race_arab race_asian
## 1           1     1       3          4  Studio Art         0          0
## 2           1     1       1          3  biophysics         0          0
## 3           2     1       1          2     biology         0          0
## 4           1     2       3          4 Mathematics         0          0
## 6           2     2       1          3     Geology         0          0
## 7           1     2       1          2        <NA>         0          0
##   race_black race_white race_hispanic race_nativeam race_nativeau
## 1          0          1             0             0             0
## 2          0          1             0             0             0
## 3          0          1             0             0             0
## 4          0          1             0             0             0
## 6          0          1             0             0             0
## 7          1          0             0             0             0
##   race_other nerdy ASD
## 1          0     7   2
## 2          0     6   2
## 3          0     7   2
## 4          0     7   2
## 6          0     5   2
## 7          0     6   2
```

Then, to proceed with our EFA, we build a correlation matrix of the "nerdiness" scale items.

```
nerdCor = lavCor(nerdTrain[,c(1:26)])
```

```
dim(nerdCor)
```

```
## [1] 26 26
```

```
nerdCor[1:10,1:10]
```

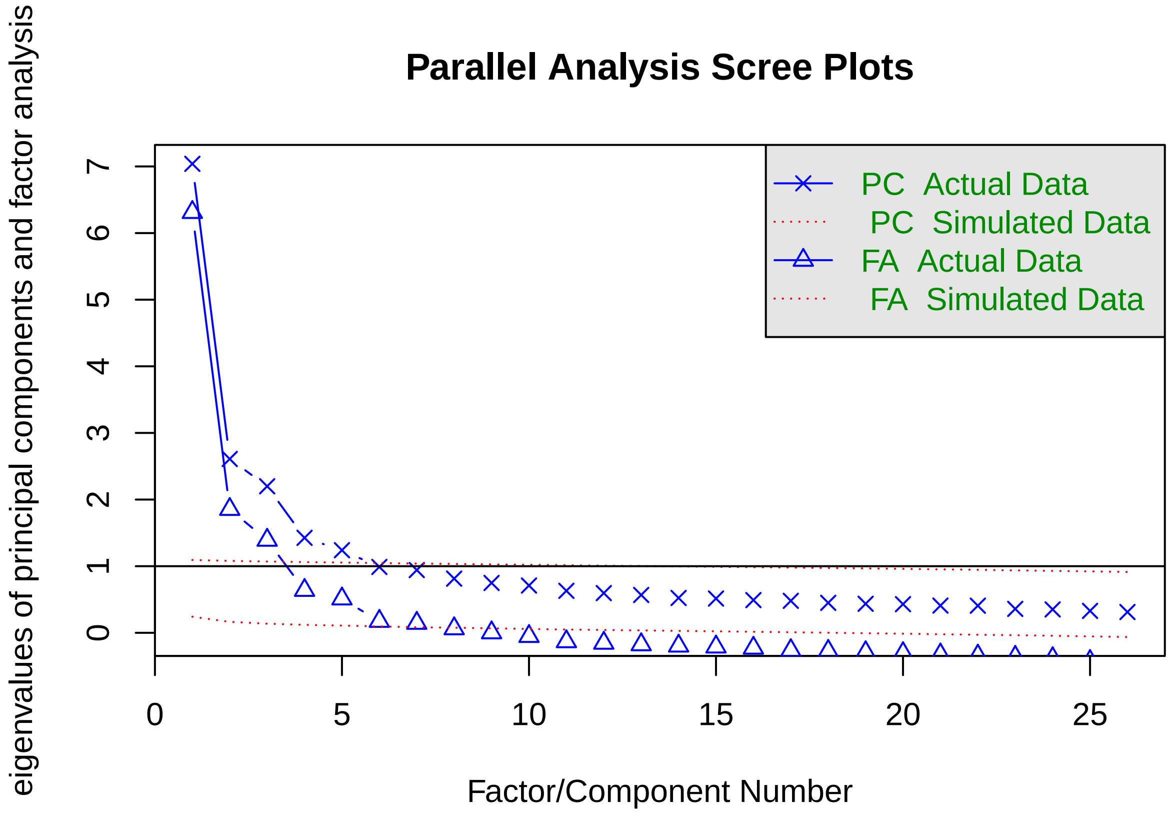
```
##            Q1        Q2        Q3        Q4        Q5        Q6        Q7
## Q1  1.0000000 0.2136586 0.4296777 0.6261649 0.2710970 0.2127082 0.1573807
## Q2  0.2136586 1.0000000 0.1826554 0.2220183 0.2216742 0.2021222 0.3988995
## Q3  0.4296777 0.1826554 1.0000000 0.3755428 0.1768708 0.2290472 0.1014630
## Q4  0.6261649 0.2220183 0.3755428 1.0000000 0.3002433 0.2415370 0.1656149
## Q5  0.2710970 0.2216742 0.1768708 0.3002433 1.0000000 0.3501039 0.3710673
## Q6  0.2127082 0.2021222 0.2290472 0.2415370 0.3501039 1.0000000 0.2492080
## Q7  0.1573807 0.3988995 0.1014630 0.1656149 0.3710673 0.2492080 1.0000000
## Q8  0.3799838 0.3032137 0.4246551 0.3222260 0.1638154 0.2601828 0.0573631
## Q9  0.2235768 0.2555981 0.1163953 0.2475506 0.4768827 0.4460943 0.3729157
## Q10 0.2593911 0.1776734 0.3222602 0.2636585 0.3251740 0.3110858 0.2065492
##            Q8        Q9       Q10
## Q1  0.3799838 0.2235768 0.2593911
## Q2  0.3032137 0.2555981 0.1776734
## Q3  0.4246551 0.1163953 0.3222602
## Q4  0.3222260 0.2475506 0.2636585
## Q5  0.1638154 0.4768827 0.3251740
## Q6  0.2601828 0.4460943 0.3110858
## Q7  0.0573631 0.3729157 0.2065492
## Q8  1.0000000 0.1785132 0.2282437
## Q9  0.1785132 1.0000000 0.2980757
## Q10 0.2282437 0.2980757 1.0000000
```

To stick with a consistent set of tools, we'll use `efaUnrotate()` in the `semTools` package to run our EFA this time.

```
?semTools::efaUnrotate
```

Before we do, we need to determine the appropriate number of factors. Instead of maximum likelihood estimation (`ml`), in this case we'll use weighted least squares methods (`wls`) because they tend to be more appropriate for ordinal data.

```
fa.parallel(nerdCor, fm="wls", n.obs=nrow(nerdTrain))
```



```
## Parallel analysis suggests that the number of factors =  7  and the number of components =  5
```

Looks like 7. Let's give it a try. `efaUnrotate()` takes raw data, but knows what to do with our ordinal variables, so it's okay to use them.

Here we use `WLSMV` as our estimator, which is a statistically robust variant (meaning: good for both normal and non-normal data) of WLS available in `lavaan` and `semTools`. A list of options are available on the `lavaan` "Estimators" webpage: http://lavaan.ugent.be/tutorial/est.html

```
# Note: long run time
nerdEFA = efaUnrotate(nerdTrain[,c(1:26)],
                      nf=7, estimator="WLSMV")
```

Then, just like our previous work with EFA, we rotate the result. Here I use `oblimin` rotation again, because we expect our different factors may be correlated (since they all measure "nerdiness").

```
nerdEFA.O = oblqRotate(nerdEFA, method="oblimin")
```

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

```
nerdEFA.O
```

```
## Standardized Rotated Factor Loadings
##     factor1 factor2 factor3 factor4 factor5 factor6 factor7
## Q17 -0.761*
## Q13 -0.629*          0.104*                  -0.147* -0.236*
## Q16 -0.597*                  -0.140*
## Q3  -0.499* -0.111* -0.337*
## Q8  -0.436*         -0.171*                 -0.333*
## Q10 -0.292*  0.276*
## Q22 -0.285*  0.189*          0.113*         -0.271* -0.189*
## Q5           0.682*                  0.104*
## Q24          0.579* -0.121*          0.116*
## Q26          0.507*                  0.160*         -0.205*
## Q9           0.495*                  0.107* -0.123* -0.222*
## Q1                  -0.778*
## Q4                  -0.756*
## Q19                         -0.866*
## Q11                         -0.750*         -0.115*
## Q7                                   0.751*  0.115* -0.122*
## Q12                                   0.728*
## Q2                          -0.179*  0.534* -0.358*
## Q25          0.248*         -0.145*  0.502*
## Q15         -0.168*         -0.134*  0.206* -0.536*
## Q23          0.246*          0.110*         -0.470* -0.276*
## Q21          0.250* -0.152*         -0.217* -0.400*
## Q6           0.141*                         -0.144* -0.513*
## Q18  0.195*         -0.111*                         -0.491*
## Q20 -0.359*                          0.178*  0.130* -0.436*
## Q14 -0.201*                          0.139*  0.109* -0.320*
##
## Factor Correlation
##            factor1     factor2    factor3     factor4     factor5
## factor1  1.0000000 -0.19992781  0.4353482  0.20338370 -0.12370052
## factor2 -0.1999278  1.00000000 -0.2843631 -0.02436977  0.42249015
## factor3  0.4353482 -0.28436309  1.0000000  0.30746979 -0.21769809
## factor4  0.2033837 -0.02436977  0.3074698  1.00000000 -0.29241720
## factor5 -0.1237005  0.42249015 -0.2176981 -0.29241720  1.00000000
## factor6  0.2430809 -0.14202300  0.2115626  0.13406027 -0.07220745
## factor7  0.3011301 -0.51064846  0.1433126 -0.12776643 -0.23331675
##             factor6    factor7
## factor1  0.24308089  0.3011301
## factor2 -0.14202300 -0.5106485
## factor3  0.21156265  0.1433126
## factor4  0.13406027 -0.1277664
## factor5 -0.07220745 -0.2333167
## factor6  1.00000000  0.2314242
## factor7  0.23142418  1.0000000
##
## Method of rotation:  Oblimin Quartimin
## [1] "The standard errors are close but do not match with other packages. Be mindful when using it."
```

And we get our 7 factors with the loadings for each item. It conveniently organizes them from highest-to-lowest loading (positive or negative) for the first factor, then for the next factor, and so on. It's okay to see high negative loadings for some items on some factors–it means that those items are strongly opposed to the positive items in that factor.

However, this immediately alerts us to a problem with this model, in that we don't have *any* high positive loadings on the first factor. The highest is Q18 ("I love to read challenging material"), but it's below 0.2, which means it really doesn't fit there either.

The problem here is that we have too many factors, even though parallel analysis said this was the best fit.

This is called "overdimensionalization" and sometimes happens with ordinal data like Likert surveys. Here is a reference to read more about it:

- van der Eijk, C., & Rose, J. (2015). Risky business: Factor analysis of survey data – Assessing the probability of incorrect dimensionalisation. *PLOS ONE, 10*(3), e0118900. doi: 10.1371/journal.pone.0118900

The solution is just to drop a factor. So we do:

```
# Note: long run time
nerdEFA = efaUnrotate(nerdTrain[,c(1:26)],
                      nf=6, estimator="WLSMV")

nerdEFA.O = oblqRotate(nerdEFA, method="oblimin")
nerdEFA.O
```

```
## Standardized Rotated Factor Loadings
##     factor1 factor2 factor3 factor4 factor5 factor6
## Q9   0.636*          0.144*                        
## Q26  0.598*          0.207*         -0.119*        
## Q24  0.539*          0.160*                  0.188*
## Q5   0.536*          0.172*         -0.203*  0.159*
## Q23  0.528*  0.139*         -0.104*  0.393*        
## Q6   0.514*  0.202*                  0.121*        
## Q18  0.380*          0.107*                        
## Q21  0.369*         -0.220*  0.104*  0.313*  0.184*
## Q22  0.346*  0.313*         -0.102*  0.213*        
## Q17          0.724*                          0.125*
## Q13  0.229*  0.690*                  0.111* -0.111*
## Q16 -0.171*  0.597*          0.155*          0.106*
## Q3  -0.139*  0.525*                          0.328*
## Q20  0.249*  0.488*  0.187*                        
## Q8           0.436*                  0.331*  0.179*
## Q14  0.254*  0.305*  0.157*                        
## Q10  0.265*  0.303*                 -0.117*  0.121*
## Q7                   0.779*                        
## Q12                  0.756*                        
## Q25  0.101*          0.539*  0.171*                
## Q2                   0.531*  0.186*  0.383*        
## Q19                          0.835*                
## Q11                          0.777*                
## Q15                  0.176*  0.145*  0.557*  0.103*
## Q1                                           0.765*
## Q4                                           0.762*
##
## Factor Correlation
##             factor1   factor2     factor3     factor4     factor5
## factor1  1.00000000 0.3074398  0.40102180 -0.04290244  0.14204093
## factor2  0.30743985 1.0000000  0.14376513  0.17124531  0.24477692
## factor3  0.40102180 0.1437651  1.00000000  0.28591339 -0.01310584
## factor4 -0.04290244 0.1712453  0.28591339  1.00000000  0.12865805
## factor5  0.14204093 0.2447769 -0.01310584  0.12865805  1.00000000
## factor6  0.23008408 0.4180401  0.23344705  0.32800642  0.13975687
##           factor6
## factor1 0.2300841
## factor2 0.4180401
## factor3 0.2334470
## factor4 0.3280064
## factor5 0.1397569
## factor6 1.0000000
##
## Method of rotation:  Oblimin Quartimin
## [1] "The standard errors are close but do not match with other packages. Be mindful when using it."
```

Looks better, but we have only negative loadings on factor 5 now (and spoiler: 5 factors doesn't work well either). At this point the better approach would probably be for us to drop some of the more problematic items and re-run analyses from the start. For this lecture, I'm just going to cut it down to 4 factors so we have something to use for our CFA.

```
# Note: long run time
nerdEFA = efaUnrotate(nerdTrain[,c(1:26)],
                      nf=4, estimator="WLSMV")
```

```
## Warning in lav_samplestats_from_data(lavdata = lavdata, missing = lavoptions$missing, : lavaan WARNI
##                     lavInspect(fit, "zero.cell.tables")

## Warning in lav_samplestats_from_data(lavdata = lavdata, missing = lavoptions$missing, : lavaan WARNI
##                     lavInspect(fit, "zero.cell.tables")
```

```
nerdEFA.O = oblqRotate(nerdEFA, method="oblimin")
nerdEFA.O
```

```
## Standardized Rotated Factor Loadings
##     factor1 factor2 factor3 factor4
## Q7   0.760*                  0.178*
## Q12  0.649*                  0.254*
## Q26  0.625*  0.151* -0.123* -0.186*
## Q5   0.568*         -0.244* -0.103*
## Q25  0.560* -0.107*          0.259*
## Q24  0.557*  0.122* -0.181*
## Q9   0.544*  0.304*         -0.114*
## Q18  0.356*  0.159*
## Q14  0.308*  0.213* -0.175*
## Q23  0.170*  0.686*  0.115*
## Q13          0.646* -0.177*
## Q22  0.105*  0.548*
## Q8  -0.163*  0.511* -0.256*  0.252*
## Q6   0.306*  0.494*         -0.153*
## Q21          0.411*
## Q20  0.244*  0.362* -0.162*
## Q1   0.101*         -0.753*
## Q4   0.160*         -0.703*
## Q3  -0.139*  0.236* -0.536*  0.105*
## Q17 -0.102*  0.360* -0.482*  0.162*
## Q16 -0.168*  0.253* -0.405*  0.230*
## Q10  0.219*  0.193* -0.304*
## Q19         -0.135* -0.102*  0.752*
## Q11                          0.716*
## Q2   0.319*  0.234*  0.107*  0.494*
## Q15          0.362*          0.429*
##
## Factor Correlation
##             factor1     factor2    factor3     factor4
## factor1  1.0000000  0.29674975 -0.1988301  0.11859358
## factor2  0.2967497  1.00000000 -0.4175340  0.05306424
## factor3 -0.1988301 -0.41753403  1.0000000 -0.26056564
## factor4  0.1185936  0.05306424 -0.2605656  1.00000000
##
## Method of rotation:  Oblimin Quartimin
## [1] "The standard errors are close but do not match with other packages. Be mindful when using it."
```

If we want more details, like standard error values and confidence intervals, we could run:

```r
summary(nerdEFA.O)
```

We can use `fitMeasures()` to look at a number of different criteria used to assess the fit of the EFA. We can also look at the r squared values ("communalities") for each item to see how well it fits. We'll come back to these for the CFA.

```r
fitMeasures(nerdEFA, c("chisq","df","pvalue","cfi","tli","rmsea","srmr"))
```

```
##    chisq       df   pvalue      cfi      tli    rmsea     srmr
## 6155.099  227.000    0.000    0.978    0.968    0.051    0.042
```

```r
inspect(nerdEFA, "rsquare")
```

```
##    Q1    Q2    Q3    Q4    Q5    Q6    Q7    Q8    Q9   Q10   Q11   Q12
## 0.569 0.444 0.458 0.518 0.462 0.449 0.599 0.498 0.494 0.267 0.569 0.520
##   Q13   Q14   Q15   Q16   Q17   Q18   Q19   Q20   Q21   Q22   Q23   Q24
## 0.535 0.262 0.291 0.388 0.546 0.182 0.618 0.327 0.206 0.340 0.509 0.445
##   Q25   Q26
## 0.396 0.523
```

Here are how each of our factors breaks down. Sometimes it's helpful to assign labels to your constructs to help make sense of them (though the labels themselves are arbitrary).

Factor 1 ("Geek"):

```
Q7  I watch science related shows.
Q12 I spend more time at the library than any other public place.
Q26 I can be socially awkward at times.
Q5  I collect books.
Q25 I care about super heroes.
Q9  I like science fiction.
Q24 I am a strange person.
Q18 I love to read challenging material.
Q14 I like to read technology news reports.
```

Factor 2 ("Reader"):

```
Q13 I would describe my smarts as bookish.
Q23 I get excited about my ideas and research.
Q22 I enjoy learning more than I need to.
Q8  I spend recreational time researching topics others might find dry or overly rigorous.
Q17 I am more comfortable interacting online than in person.
Q6  I prefer academic success to social success.
Q20 I was a very odd child.
Q21 I sometimes prefer fictional people to real ones.
Q16 I gravitate towards introspection.
Q10 I would rather read a book than go to a party.
```

Factor 3 ("Know-it-all"):

```
Q1  I am interested in science.
```

```
Q4  My appearance is not as important as my intelligence.
Q3  I like to play RPGs. (Ex. D&D)
```

Factor 4 ("Gamer"):

```
Q19 I have played a lot of video games.
Q11 I am more comfortable with my hobbies than I am with other people.
Q2  I was in advanced classes.
Q15 I have started writing a novel.
```

Then, for our subsequent analysis, we'll need to define this model in a way that `lavaan` can understand. There are details on the syntax in their tutorial here (part 1) and here (part 2).

```
nerdModel =  "nerd =~ geek + reader + knowitall + gamer
              geek =~ Q7 + Q12 + Q26 +Q5 + Q25 + Q9 + Q24 + Q18 + Q14
              reader =~ Q13 + Q23 + Q22 + Q8 + Q17 + Q6 + Q20 + Q21 + Q16 + Q10
              knowitall =~ Q1 + Q4 + Q3
              gamer =~ Q19 + Q11 + Q2 + Q15"
```

Essentially we're telling it that `geek`, `reader`, `knowitall` and `gamer` are our 4 factors, and that they all contribute to a second-order factor called `nerd`.

Confirmatory factor analysis

Note: The most recent versions of the `lavaan` and `semTools` packages refused to fit the model for me. I had to roll back to older versions, using the commands below. If you're trying to replicate it and can't get it to work, try doing the same.

```
remove.packages(c("lavaan","semTools"))
install.packages("devtools")
devtools::install_version("lavaan", version="0.6.3")
devtools::install_version("semTools", version="0.4-14")
```

Now that we have our model, built using our train subset, we can test the fit of our test subset to that model using CFA. It's fairly straightforward here (but more details are available on the `lavaan` webpage):

```
nerdFit = cfa(nerdModel, nerdTest[,c(1:26)], estimator="WLSMV")
```

```
## Warning in lav_samplestats_from_data(lavdata = lavdata, missing = lavoptions$missing, : lavaan WARNI
##                     lavInspect(fit, "zero.cell.tables")
```

We get the warning again about empty bivariate cells, which we're okay to ignore.

```
summary(nerdFit)
```

```
## lavaan (0.5-23.1097) converged normally after  34 iterations
##
##   Number of observations                          9875
##
##   Estimator                                       DWLS      Robust
##   Minimum Function Test Statistic            27890.631   29727.068
##   Degrees of freedom                               295         295
##   P-value (Chi-square)                           0.000       0.000
##   Scaling correction factor                                  0.941
##   Shift parameter                                           76.852
##     for simple second-order correction (Mplus variant)
##
## Parameter Estimates:
##
```

```
##   Information                                 Expected
##   Standard Errors                           Robust.sem
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   nerd =~
##     geek              1.000
##     reader            1.356    0.033   40.870    0.000
##     knowitall         1.267    0.029   43.733    0.000
##     gamer             0.739    0.022   33.384    0.000
##   geek =~
##     Q7                1.000
##     Q12               1.009    0.015   67.554    0.000
##     Q26               1.088    0.015   70.779    0.000
##     Q5                1.048    0.015   68.615    0.000
##     Q25               0.825    0.015   56.517    0.000
##     Q9                1.059    0.016   68.226    0.000
##     Q24               1.037    0.016   66.033    0.000
##     Q18               0.576    0.017   33.949    0.000
##     Q14               0.831    0.016   51.277    0.000
##   reader =~
##     Q13               1.000
##     Q23               0.833    0.014   60.354    0.000
##     Q22               0.795    0.014   57.554    0.000
##     Q8                0.901    0.014   65.948    0.000
##     Q17               1.012    0.013   76.021    0.000
##     Q6                0.911    0.014   66.350    0.000
##     Q20               0.861    0.014   62.427    0.000
##     Q21               0.625    0.015   41.026    0.000
##     Q16               0.775    0.014   55.468    0.000
##     Q10               0.759    0.015   50.779    0.000
##   knowitall =~
##     Q1                1.000
##     Q4                0.993    0.015   65.947    0.000
##     Q3                0.901    0.016   56.565    0.000
##   gamer =~
##     Q19               1.000
##     Q11               1.197    0.024   48.902    0.000
##     Q2                1.200    0.025   48.867    0.000
##     Q15               0.858    0.021   40.772    0.000
##
## Intercepts:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .Q7                0.000
##    .Q12               0.000
##    .Q26               0.000
##    .Q5                0.000
##    .Q25               0.000
##    .Q9                0.000
##    .Q24               0.000
##    .Q18               0.000
##    .Q14               0.000
##    .Q13               0.000
##    .Q23               0.000
##    .Q22              0.000
##    .Q8               0.000
##    .Q17              0.000
##    .Q6               0.000
##    .Q20              0.000
##    .Q21              0.000
##    .Q16              0.000
##    .Q10              0.000
##    .Q1               0.000
##    .Q4               0.000
##    .Q3               0.000
##    .Q19              0.000
##    .Q11              0.000
##    .Q2               0.000
##    .Q15              0.000
##     nerd             0.000
##     geek             0.000
##     reader           0.000
##     knowitall        0.000
##     gamer            0.000
##
## Thresholds:
##                    Estimate  Std.Err  z-value  P(>|z|)
##     Q7|t1            -2.615    0.051  -50.912    0.000
##     Q7|t2            -1.738    0.023  -76.640    0.000
##     Q7|t3            -1.296    0.017  -74.797    0.000
##     Q7|t4            -0.837    0.014  -58.293    0.000
##     Q7|t5            -0.016    0.013   -1.298    0.194
##     Q12|t1           -2.631    0.052  -50.285    0.000
##     Q12|t2           -1.427    0.019  -76.748    0.000
##     Q12|t3           -0.933    0.015  -62.952    0.000
##     Q12|t4           -0.469    0.013  -35.747    0.000
##     Q12|t5            0.344    0.013   26.703    0.000
##     Q26|t1           -2.722    0.058  -46.680    0.000
##     Q26|t2           -1.896    0.026  -74.250    0.000
##     Q26|t3           -1.370    0.018  -76.057    0.000
##     Q26|t4           -0.797    0.014  -56.171    0.000
##     Q26|t5            0.010    0.013    0.755    0.450
##     Q5|t1            -2.925    0.075  -38.790    0.000
##     Q5|t2            -1.599    0.021  -77.489    0.000
##     Q5|t3            -1.030    0.015  -67.003    0.000
##     Q5|t4            -0.528    0.013  -39.783    0.000
##     Q5|t5             0.313    0.013   24.342    0.000
##     Q25|t1           -2.755    0.061  -45.402    0.000
##     Q25|t2           -0.969    0.015  -64.497    0.000
##     Q25|t3           -0.385    0.013  -29.679    0.000
##     Q25|t4            0.096    0.013    7.616    0.000
##     Q25|t5            0.799    0.014   56.284    0.000
##     Q9|t1            -2.744    0.060  -45.840    0.000
##     Q9|t2            -1.711    0.022  -76.900    0.000
##     Q9|t3            -1.183    0.016  -72.102    0.000
##     Q9|t4            -0.568    0.013  -42.440    0.000
##     Q9|t5             0.275    0.013   21.497    0.000
##     Q24|t1           -2.766    0.062  -44.951    0.000
##     Q24|t2          -2.046    0.029  -70.773    0.000
##     Q24|t3          -1.600    0.021  -77.487    0.000
##     Q24|t4          -1.023    0.015  -66.736    0.000
##     Q24|t5          -0.059    0.013   -4.659    0.000
##     Q18|t1          -2.639    0.053  -49.962    0.000
##     Q18|t2          -1.268    0.017  -74.211    0.000
##     Q18|t3          -0.927    0.015  -62.650    0.000
##     Q18|t4          -0.516    0.013  -38.974    0.000
##     Q18|t5           0.077    0.013    6.107    0.000
##     Q14|t1          -2.778    0.062  -44.485    0.000
##     Q14|t2          -1.667    0.022  -77.230    0.000
##     Q14|t3          -1.007    0.015  -66.078    0.000
##     Q14|t4          -0.298    0.013  -23.241    0.000
##     Q14|t5           0.447    0.013   34.199    0.000
##     Q13|t1          -2.790    0.063  -44.004    0.000
##     Q13|t2          -1.357    0.018  -75.857    0.000
##     Q13|t3          -0.840    0.014  -58.441    0.000
##     Q13|t4          -0.274    0.013  -21.417    0.000
##     Q13|t5           0.240    0.013   18.810    0.000
##     Q23|t1          -2.744    0.060  -45.840    0.000
##     Q23|t2          -1.306    0.017  -74.991    0.000
##     Q23|t3          -0.873    0.015  -60.111    0.000
##     Q23|t4          -0.520    0.013  -39.211    0.000
##     Q23|t5           0.093    0.013    7.335    0.000
##     Q22|t1          -2.830    0.067  -42.462    0.000
##     Q22|t2          -0.389    0.013  -29.998    0.000
##     Q22|t3           0.202    0.013   15.899    0.000
##     Q22|t4           0.693    0.014   50.318    0.000
##     Q22|t5           1.190    0.016   72.301    0.000
##     Q8|t1           -2.712    0.058  -47.083    0.000
##     Q8|t2           -1.340    0.018  -75.586    0.000
##     Q8|t3           -0.936    0.015  -63.076    0.000
##     Q8|t4           -0.556    0.013  -41.654    0.000
##     Q8|t5            0.131    0.013   10.392    0.000
##     Q17|t1          -2.907    0.074  -39.469    0.000
##     Q17|t2          -1.759    0.023  -76.405    0.000
##     Q17|t3          -1.124    0.016  -70.325    0.000
##     Q17|t4          -0.539    0.013  -40.552    0.000
##     Q17|t5           0.202    0.013   15.859    0.000
##     Q6|t1           -2.733    0.059  -46.266    0.000
##     Q6|t2           -1.586    0.020  -77.504    0.000
##     Q6|t3           -0.982    0.015  -65.069    0.000
##     Q6|t4           -0.315    0.013  -24.522    0.000
##     Q6|t5            0.520    0.013   39.211    0.000
##     Q20|t1          -2.790    0.063  -44.004    0.000
##     Q20|t2          -1.612    0.021  -77.462    0.000
##     Q20|t3          -0.957    0.015  -63.991    0.000
##     Q20|t4          -0.194    0.013  -15.276    0.000
##     Q20|t5           0.477    0.013   36.283    0.000
##     Q21|t1          -2.665    0.054  -48.947    0.000
##     Q21|t2          -0.534    0.013  -40.197    0.000
##     Q21|t3          -0.177    0.013  -13.950    0.000
##     Q21|t4           0.092    0.013    7.315    0.000
##     Q21|t5           0.617    0.014   45.591    0.000
##     Q16|t1          -2.925    0.075  -38.790    0.000
##     Q16|t2          -1.156    0.016  -71.333    0.000
##     Q16|t3          -0.616    0.014  -45.532    0.000
##     Q16|t4          -0.089    0.013   -7.013    0.000
##     Q16|t5           0.441    0.013   33.742    0.000
##     Q10|t1          -2.571    0.049  -52.643    0.000
##     Q10|t2          -1.825    0.024  -75.506    0.000
##     Q10|t3          -1.420    0.019  -76.674    0.000
##     Q10|t4          -0.680    0.014  -49.490    0.000
##     Q10|t5           0.071    0.013    5.604    0.000
##     Q1|t1           -2.803    0.064  -43.508    0.000
##     Q1|t2           -1.754    0.023  -76.460    0.000
##     Q1|t3           -1.260    0.017  -74.024    0.000
##     Q1|t4           -0.623    0.014  -46.020    0.000
##     Q1|t5            0.242    0.013   18.970    0.000
##     Q4|t1           -2.683    0.056  -48.229    0.000
##     Q4|t2           -1.548    0.020  -77.483    0.000
##     Q4|t3           -0.965    0.015  -64.358    0.000
##     Q4|t4           -0.425    0.013  -32.588    0.000
##     Q4|t5            0.328    0.013   25.503    0.000
##     Q3|t1           -2.639    0.053  -49.962    0.000
##     Q3|t2           -1.796    0.024  -75.930    0.000
##     Q3|t3           -1.368    0.018  -76.020    0.000
##     Q3|t4           -0.960    0.015  -64.131    0.000
##     Q3|t5           -0.034    0.013   -2.667    0.008
##     Q19|t1          -2.778    0.062  -44.485    0.000
##     Q19|t2          -0.852    0.014  -59.067    0.000
##     Q19|t3          -0.428    0.013  -32.787    0.000
##     Q19|t4          -0.103    0.013   -8.180    0.000
##     Q19|t5           0.352    0.013   27.283    0.000
##     Q11|t1          -2.693    0.056  -47.857    0.000
##     Q11|t2          -0.653    0.014  -47.866    0.000
##     Q11|t3          -0.320    0.013  -24.903    0.000
##     Q11|t4           0.082    0.013    6.470    0.000
##     Q11|t5           0.575    0.013   42.911    0.000
##     Q2|t1           -2.600    0.050  -51.512    0.000
##     Q2|t2           -1.538    0.020  -77.463    0.000
##     Q2|t3           -1.121    0.016  -70.218    0.000
##     Q2|t4           -0.713    0.014  -51.489    0.000
##     Q2|t5            0.032    0.013    2.506    0.012
##     Q15|t1          -2.693    0.056  -47.857    0.000
##     Q15|t2          -0.669    0.014  -48.872    0.000
##     Q15|t3          -0.266    0.013  -20.836    0.000
##     Q15|t4           0.147    0.013   11.598    0.000
##     Q15|t5           0.732    0.014   52.598    0.000
##
## Variances:
##                   Estimate  Std.Err  z-value  P(>|z|)
##    .Q7               0.585
##    .Q12              0.578
##    .Q26              0.509
##    .Q5               0.544
##    .Q25              0.718
##    .Q9               0.535
##    .Q24              0.554
##    .Q18              0.862
##    .Q14              0.714
##    .Q13              0.552
##    .Q23              0.689
##    .Q22              0.717
##    .Q8               0.636
##    .Q17              0.541
##    .Q6               0.628
##    .Q20              0.668
##    .Q21              0.825
##    .Q16              0.731
##    .Q10              0.742
##    .Q1               0.449
##    .Q4               0.457
##    .Q3               0.553
##    .Q19              0.615
##    .Q11              0.448
##    .Q2               0.445
##    .Q15              0.716
##     nerd             0.184    0.007   26.676    0.000
##     geek             0.231    0.007   34.294    0.000
##     reader           0.109    0.007   15.940    0.000
##     knowitall        0.255    0.009   29.752    0.000
##     gamer            0.285    0.008   33.629    0.000
##
## Scales y*:
##                   Estimate  Std.Err  z-value  P(>|z|)
##     Q7               1.000
##     Q12              1.000
##     Q26              1.000
##     Q5               1.000
##     Q25              1.000
##     Q9               1.000
##     Q24              1.000
##     Q18              1.000
##     Q14              1.000
##     Q13              1.000
##     Q23              1.000
##     Q22              1.000
##     Q8               1.000
##     Q17              1.000
##     Q6               1.000
##     Q20              1.000
##     Q21              1.000
##     Q16              1.000
##     Q10              1.000
##     Q1               1.000
##     Q4               1.000
##     Q3               1.000
##     Q19              1.000
##     Q11              1.000
##     Q2               1.000
##     Q15              1.000
```

We see the results for all 4 factors and the items that they're constructed from, as well as a bunch of other information. But how well does the model fit the data?

```
fitMeasures(nerdFit, c("chisq","df","pvalue","cfi","tli","rmsea","srmr"))
```

```
##     chisq        df    pvalue       cfi       tli     rmsea      srmr
## 27890.631   295.000     0.000     0.891     0.880     0.097     0.088
```

Chi-square statistics (including degrees of freedom and p-value) are commonly reported in the literature and are expected by some reviewers, but really mean nothing in this context as they're almost always significant given a decent sample size. The ones to pay attention to are the last 4: CFI, TLI, RMSEA, and (sometimes) SRMR. Good criteria to use to evaluate them are:

Comparative Fit Index ("CFI") > 0.95
Tucker-Lewis Index ("TLI") > 0.96
Root Mean Square Error of Approximation ("RMSEA") < 0.05
Standardized Root Mean Square Residual ("SRMR") < 0.07

So, does our model fit? No, not well at all–all of our fit indices are outside the criteria. This means we'd have to go back, probably trim down some of the less well-fitting items, and see if we can build a better model.

Note: These values come from the following sources (of which, Hu and Bentler is the classic source and Yu updated a couple of the scores based on additional analyses):

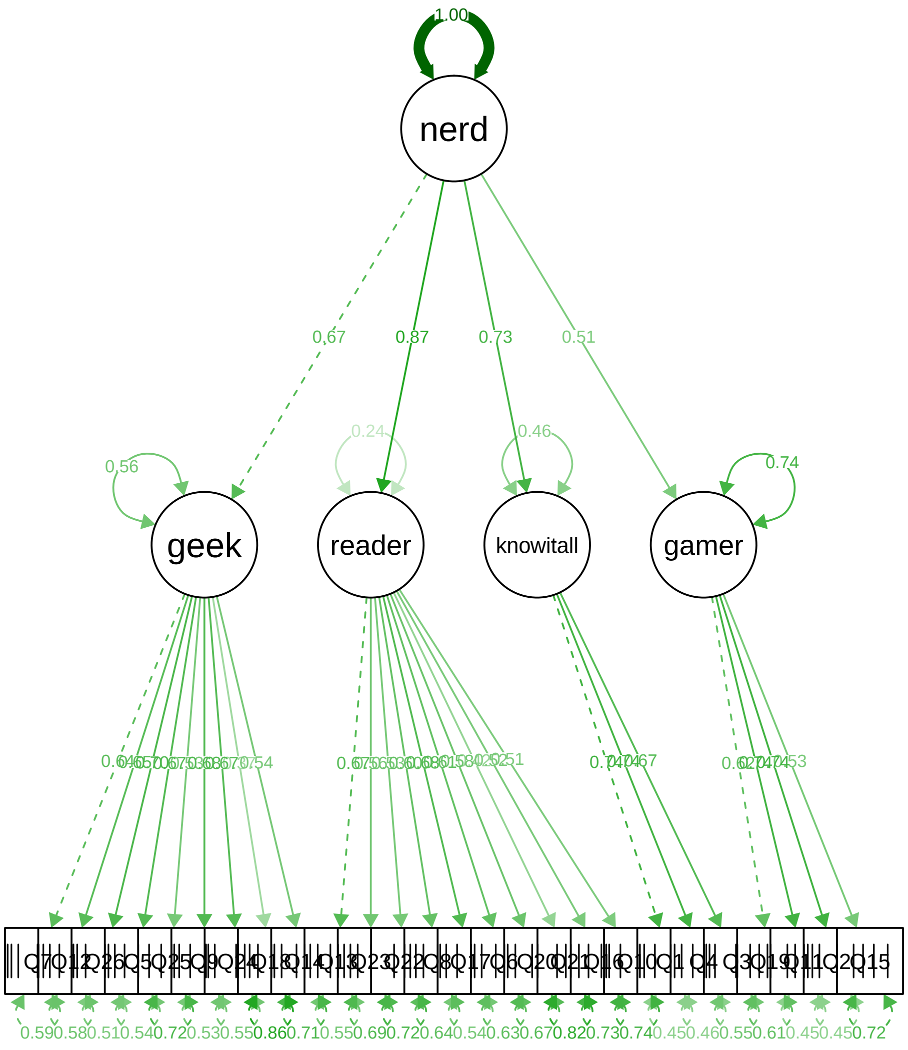
- Hu L, Bentler PM (1999) Cutoff criteria for fit indexes in covariance structure analysis: Conventional criteria versus new alternatives. Structural Equation Modeling 6(1):1–55. doi: 10.1080/10705519909540118
- Yu C-Y (2002) Evaluating cutoff criteria of model fit indices for latent variable models with binary and continuous outcomes. Dissertation (University of California, Los Angeles). Available: https://www.statmodel.com/download/Yudissertation.pdf

Before we go, let's plot our model using the `semPlot` package.

```
install.packages("semPlot")
# devtools::install_version("semPlot", version="1.1")
```

```
library(semPlot)
```

```
semPaths(nerdFit, what="std", nCharNodes=0, intercepts=F)
```



You'd need to know a bit more about CFA and SEM to be able to interpret everything properly, but the lines from latent factors to lower factors and items represent loadings, so you can see the relative strength of each.

The figure could use some cleaning up, using additional parameters (see `?semPaths`), to make everything a bit more visible. But here, we'll leave it as-is and move on.

CFA is just a specialized instance of SEM, so there are many more ways to do SEM using the `lavaan` package. More details are available here: http://lavaan.ugent.be/tutorial/sem.html

The general function is `sem()`:

```
?lavaan::sem
```

Social network analysis

To have a data set to explore as a first look into social networks, we'll use data on a Northern Alaskan community helping network from the following paper:

Lee, H. W., et al. (2018). Mapping the structure of perceptions in helping networks of Alaska Natives. PLOS ONE, 13(11), e0204343. doi: 10.1371/journal.pone.0204343

In the Supporting Information, download "S1 File" and extract "Edge data.csv" to your `/data` folder: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0204343#sec016

A little network data terminology: networks are made up of nodes and edges. Nodes are the individual points in the network, usually people, and are often represented as numerical IDs. Edges are the connections between nodes, and can represent any kind of association or interaction (time spent together, citations, etc.). When we read in the data, we can have separate lists of nodes and edges, and edge lists often contain other variables that relate to the links between nodes.

There are two types of networks: directed and undirected. In directed networks, the edges have a direction, e.g. A does something to B. In undirected networks, A is connected to B, but there is no direction to the connection, they're just associated.

Here, all we have is an edge list and, from the paper, it's directed (A helps B). We read it in:

```
helpnet = read.csv("./data/Edge data.csv")
```

```
head(helpnet)
```

```
##   NodeA.01. NodeB.02. Saw_AB.15. Rec_AB.16. Diff_AB.17. Same_AB
## 1       144       210          9          6           5       1
## 2       210       144          9          6           5       1
## 3       283       385          9          8           6       2
## 4       385       283          9          8           6       2
## 5      -170        92          8          7           5       2
## 6      -143       144          8          3           1       2
##   Avg_Dist.18. Std_Dist.19.
## 1    0.4906949    0.1013380
## 2    0.4906949    0.1013380
## 3    0.3279706    0.2976857
## 4    0.3279706    0.2976857
## 5    0.5479487    0.3182149
## 6    0.3696195    0.0000000
```

We can see the network has some negative values for the nodes.

```
summary(helpnet$NodeA.01.)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -172.00  -49.00   85.00   97.74  250.00  395.00
```

```
summary(helpnet$NodeB.02.)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -172.00  -46.00   86.00   99.31  250.00  395.00
```

This is actually what's called a "signed" network, in that it has edges with positive and negative signs. Positive values represent positive interactions (from the paper: "e.g. liking, loving, supporting") and negative interactions ("disliking, hating, opposing"). To simplify things here, we're just going to look at the positive edges.

```
helpnetP = helpnet[helpnet$NodeA.01. > 0 & helpnet$NodeB.02. > 0,]
dim(helpnetP)
```

```
## [1] 64262     8
```

```
length(unique(unname(unlist(helpnetP[,1:2]))))
```

```
## [1] 254
```

Looks like it's still a very big network, with 64,262 connections between 254 nodes.

There are a number of social network packages in R, the main ones being `network`, `sna`, and `igraph`. Here we'll use `igraph` because it seems to be the most commonly used package, and is also cross-platform with other programming languages such as Python. But depending on your analyses and your needs, feel free to check the others out too.

```
install.packages("igraph")
```

```
library(igraph)
```

We'll use the `graph_from_data_frame()` function to make our data frame into an igraph object.

```
?graph_from_data_frame
```

```
helpnetG = graph_from_data_frame(helpnetP)
helpnetG
```

```
## IGRAPH f271f4c DN-- 254 64262 --
## + attr: name (v/c), Saw_AB.15. (e/n), Rec_AB.16. (e/n),
## | Diff_AB.17. (e/n), Same_AB (e/n), Avg_Dist.18. (e/n),
## | Std_Dist.19. (e/n)
## + edges from f271f4c (vertex names):
##  [1] 144->210 210->144 283->385 385->283 20 ->335 132->283 199->269
##  [8] 223->362 269->199 283->132 335->20  362->223 5  ->258 53 ->310
## [15] 74 ->202 74 ->208 146->199 151->302 181->283 199->146 202->74
## [22] 208->74  210->286 223->315 223->331 234->301 258->5   259->385
## [29] 283->181 286->210 290->392 301->234 302->151 310->53  315->223
## [36] 320->331 331->223 331->320 385->259 392->290 5  ->84  5  ->98
## + ... omitted several edges
```

Looks like it loaded in correctly. We can see the other variables from the data frame as attributes, and the beginning of the edge list, with direction indicated. We can get vectors of the nodes (in `igraph`, called "vertices") with `V()` and the edges with `E()`.

```
V(helpnetG)
```

```
## + 254/254 vertices, named, from f271f4c:
##   [1] 144 210 283 385 20  132 199 223 269 335 362 5   53  74  146 151 181
##  [18] 202 208 234 258 259 286 290 301 302 310 315 320 331 392 21  33  35
##  [35] 36  38  41  42  46  55  62  63  66  68  71  73  80  83  84  85  92
##  [52] 97  98  101 107 108 109 110 111 117 119 124 127 130 155 157 163 172
##  [69] 173 174 176 180 185 187 190 197 198 203 206 207 220 224 227 228 241
##  [86] 243 245 247 253 255 256 257 261 264 265 266 273 274 275 276 278 292
## [103] 293 296 309 311 314 326 336 337 339 344 349 368 378 384 388 393 395
## [120] 1   2   7   8   9   14  19  23  24  26  31  32  34  44  45  54  58
## [137] 59  65  69  72  76  77  78  86  89  93  99  100 106 112 126 128 131
## [154] 134 135 141 142 145 152 154 158 159 165 166 167 168 169 170 171 175
## + ... omitted several vertices
```

```
E(helpnetG)
```

```
## + 64262/64262 edges from f271f4c (vertex names):
##  [1] 144->210 210->144 283->385 385->283 20 ->335 132->283 199->269
##  [8] 223->362 269->199 283->132 335->20  362->223 5  ->258 53 ->310
## [15] 74 ->202 74 ->208 146->199 151->302 181->283 199->146 202->74
## [22] 208->74  210->286 223->315 223->331 234->301 258->5   259->385
## [29] 283->181 286->210 290->392 301->234 302->151 310->53  315->223
## [36] 320->331 331->223 331->320 385->259 392->290 5  ->84  5  ->98
## [43] 21 ->385 33 ->227 35 ->132 36 ->174 38 ->185 38 ->187 41 ->84
## [50] 41 ->92  42 ->302 42 ->310 42 ->344 46 ->293 53 ->55  53 ->220
## [57] 53 ->265 55 ->53  55 ->111 55 ->220 55 ->253 55 ->259 55 ->362
## [64] 62 ->326 62 ->368 63 ->243 66 ->80  68 ->180 68 ->368 71 ->296
## + ... omitted several edges
```

If we want to pull out any of the other attributes from our original edge list, we can use the `edge_attr()` function:

```
edge_attr(helpnetG, "Avg_Dist.18.")[1:10]
```

```
##  [1] 0.4906949 0.4906949 0.3279706 0.3279706 0.5044079 0.4257790 0.2104445
##  [8] 0.5632773 0.2104445 0.4257790
```

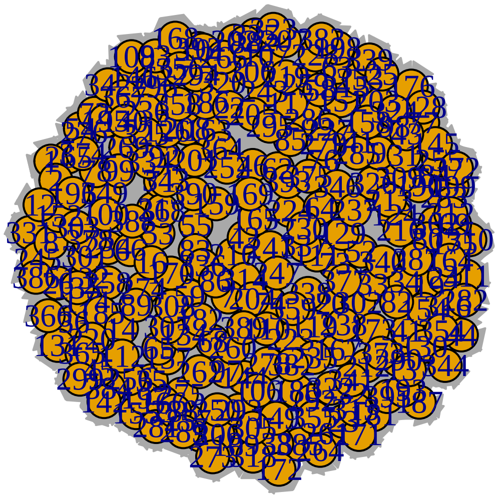
If we reference it like a matrix, we can see the connections between nodes:

```
helpnetG[1,]
```

```
## 144 210 283 385  20 132 199 223 269 335 362   5  53  74 146 151 181 202
##   0   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 208 234 258 259 286 290 301 302 310 315 320 331 392  21  33  35  36  38
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  41  42  46  55  62  63  66  68  71  73  80  83  84  85  92  97  98 101
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 107 108 109 110 111 117 119 124 127 130 155 157 163 172 173 174 176 180
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 185 187 190 197 198 203 206 207 220 224 227 228 241 243 245 247 253 255
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 256 257 261 264 265 266 273 274 275 276 278 292 293 296 309 311 314 326
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 336 337 339 344 349 368 378 384 388 393 395   1   2   7   8   9  14  19
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  23  24  26  31  32  34  44  45  54  58  59  65  69  72  76  77  78  86
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  89  93  99 100 106 112 126 128 131 134 135 141 142 145 152 154 158 159
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 165 166 167 168 169 170 171 175 186 193 195 201 204 212 213 216 235 236
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 250 260 279 285 294 303 306 307 308 316 318 322 324 325 338 342 343 346
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 355 359 364 365 366 367 373 374 377 382 383 386 389  16  22  25  43  52
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  64  82  88 129 137 140 149 164 196 205 230 237 238 252 262 281 287 288
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 295 299 305 312 313 323 334 341 354 356 358 375  12 182 219 282 289 340
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 372 390
##   1   1
```

It looks like every node is connected to every other node. This may not make for a very informative graph, but we can still plot it to take a look:

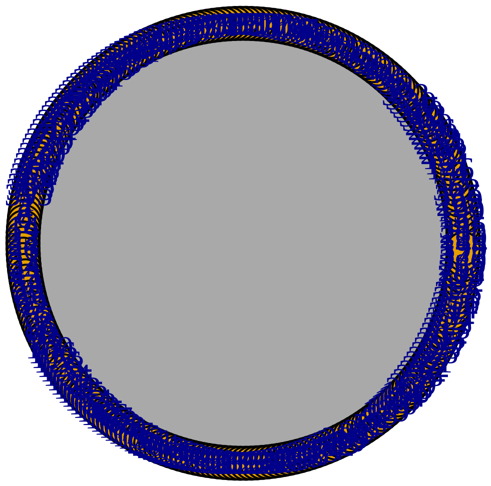
```
plot(helpnetG)
```



Yep, just a total mess. This can often be a downfall of social network diagrams, in that they can be very pretty, but contain such a high density of information that they don't really communicate anything (refer back to our resources on information design).

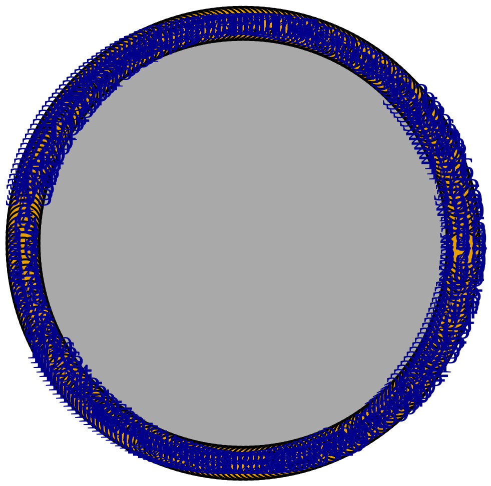
We can try to remedy it by changing the layout of the data:

```
plot(helpnetG, layout=layout_in_circle)
```



Nope. Still a mess. If we had less data, perhaps by looking at meaningful subsets, it could be better. Since all nodes are connected, it would be good to use those attributes to, for instance, display different line weights for our edges. Though we won't notice a difference here, we could do it like so:

```
plot(helpnetG, layout=layout_in_circle, edge.width=E(helpnetG)$Avg_Dist.18.)
```



We'll do more with plotting next time, with a more tractable data set.

(pdf / Rmd)