Computational Statistics 11.2: Factor Analysis

Factor analysis

Factor analysis
- **Latent dimensions in space**
- **Projection of observations**
- **Projection of variables**

Estimation and derivation
- **Estimation via eigenvectors**
- **Example estimation**
- **Example 2**
- **Example 3**
- **Example 4**

Other estimation methods
- **Estimation bonus**

How many factors to keep?
- **Scree plot**
- **Cumulative variance**

Factor analysis

Factor analysis
- **Latent dimensions in space**
- **Projection of observations**
- **Projection of variables**

Estimation and derivation
- **Estimation via eigenvectors**
- **Example estimation**
- **Example 2**
- **Example 3**
- **Example 4**

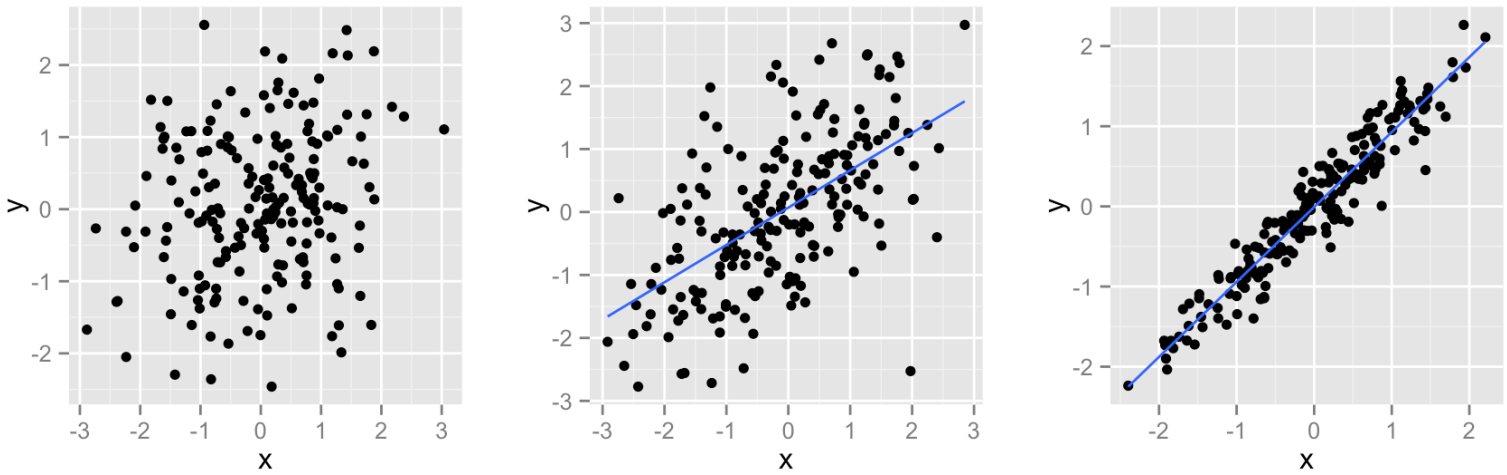
Other estimation methods
- **Estimation bonus**

How many factors to keep?
- **Scree plot**

Factor analysis

Factor analysis is the search for these underlying dimensions, or factors, underneath a larger set of variables. Factor analysis actually long predates the era of big data and machine learning, and has a long history of different methods, most of which produce similar results.

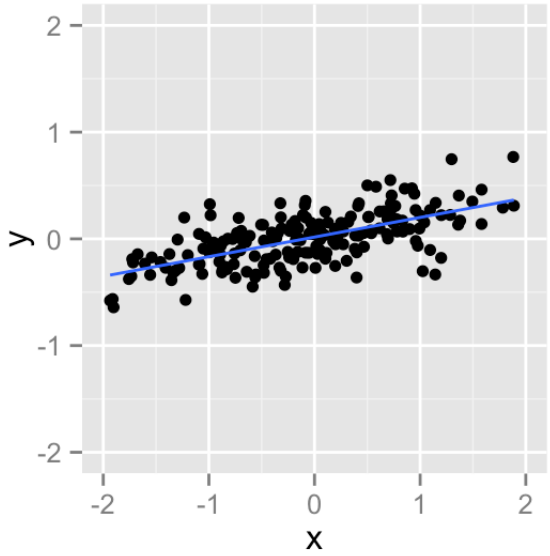
Usually we find strong underlying factors when the variables we measure are correlated with each other. For instance, when x and y values are well-correlated (as in the previous figure) creating a diagonal (American) football shape, then we can speculate that there is actually one underlying factor that explains them both.



In the figure on the right, there is mainly a single underlying factor explaining the data, and x and y are just two measurements of basically the same thing (eg, x and y could be length of the right leg and length of the left leg, and the underlying factor is a person's height). In the figure on the left, x and y are uncorrelated, and thus there are truly two different variables at play here. And in the middle is something inbetween: there is a strong diagional latent variable, but also a lot of variation around that diagional line, which may constitute a second latent dimension.

Projection of variables

The second way to interpret the latent dimenion is in terms not of the observations (what each individual score is on these new axes), but in terms of the variables themselves. Consider the following data:



Here, although there is clearly an underlying factor that explains much of the variation in the data, that dimension is fairly similar to x, and fairly orthogonal (perpendicular) to y. That is, the projection of the principal component on x is large, while the projection on y is small. If these were political opinions again, we would conclude that the political world is mainly determined by one's social views, and only slightly affected by one's economic views; while there is still basically just one main political dimension (the first principal component, the blue line), that main political dimension isn't all that different from the x variable, and the scores individuals would have on x would be similar to the scores they have on the first component.

So when we have found our principal components, then, we can ask two things: First, how do the observations project onto the components? That is, what are the individual scores not in terms of the original variables, but in terms of the new, transformed dimensions. Second, how do the original variables project onto the components: that is (roughly speaking) what proportion of each of the original variables make the new one? Is it 50% x and 30% y and 20% z (ie, is it pointing mainly in the x direction), or is it 10% x, 10% y, and 80% z (ie, pointing almost entirely in the z direction, and not very different from z)? Etc.

As we will see, when we do our factor analysis, we get both. But in terms of understanding the system we are examining, we are mainly interested in the second: how do our new variables (factors or components; I use the terms interchangeably here) explain and relate to the old ones? Only in some applications do we really care about the projections of the observations. One example of the latter is IQ. IQ is perhaps the most famous factor analysis, which takes scores on a bunch of different types of mental tests, and finds that they are all strongly correlated with each other and thus can be "explained" by a single underlying factor; when you take an IQ test, you actually get scores on all these separate variables, and then your IQ is the underlying factor that combines these scores into one.

Estimation and derivation

As you might imagine from the fact that the components are straight lines, the equation for each component in terms of the existing variables X_1, X_2, etc, is a simple linear equation

$$C_p = \phi_{p1} X_1 + \phi_{p2} X_2 + \phi_{p3} X_2 \ldots \phi_{pp} X_p$$

Where C_p is the pth component, and as said before, there are always as many components as there are variables (p). The first component, C_1, is defined by the values $\{\phi_{11}, \phi_{12}, \dots\}$ (which like β_i, are just numbers), and in this equation, just as X_1 is a vector of n observations, C_p is also a vector of size n: the projection of the observations onto that component. What we want is to find those values $\{\phi_{11}, \phi_{12}, \dots\}$ such that the variance of C_1 is maximized – ie, we want to find the latent dimension (line in space) that goes through the longest axis of the "football" of data. After we've found the first component C_1, we can then find C_2, the dimension through the next longest axis of the data football, and so on. And values $\{\phi_{11}, \phi_{12}, \dots\}$ are of course the projections of the original variables onto C_1 – the larger ϕ_{12} is, for instance, the larger the projection of X_2 on C_1 – ie, the more like X_2 C_1 is, or the more similar their directions in space are.

So how do we find these values $\{\phi_{p1}, \phi_{p2}, \dots\}$? There are a number of methods, and what is powerful about factor analysis (FA) or principal component analysis (PCA) is that all these methods tend to produce very similar results. In fact, factor analysis and principal component analysis aren't quite the same thing, but rather than worrying about that, we can trust in the fact that fundamentally they tend to produce the same dimensions.

Estimation via eigenvectors

The most common estimation of principal components is using the covariance matrix of the variables. That is, if X_1 and X_3 are highly correlated (and thus have a high covariance), then since an observation's scores X_1 and X_3 are very similar, we would likewise expect X_1 and X_3 to have similar scores on any given component. The covariance matrix presents us with all the covariances between every pair of variables, and what we want to do is use this matrix to find which variables are jointly both highly correlated, and have a high degree of variance. One flaw with this approach as described is that the variance of a variable can depend in its units: measured in $10K units, the variance of incomes in the US is one thing, but measured in $1 units, the variance is much larger. So generally we need to standardize (rescale) everything to have a mean of 0 and sd of 1 before we conduct our analysis – just as one does in standardized regression to allow comparison of β sizes between X variables.

Once we've got our covariance matrix, we then need to find the component (direction) with the highest variance. This is often does using the eigenvectors of the covariance matrix. Explaining this in perfect detail would take us too far afield, but the eigenvectors of a matrix are much like the principal components of some data set: a p x p matrix has exactly p eigenvectors, where each eigenvector is a vector of length p. An eigenvector is defined as that vector such that, when multiplied by its associated matrix (using matrix multiplication; see Module 1.2), gives as a result the exact same vector, except stretched or shrunk by some value known as the eigenvalue. The equation is

$$Mv_p = \lambda_p v_p$$

That is, matrix M times vector v_p equals that vector v_p times some real number λ_p. Each eigenvector v_p has its own associated eigenvalue λ_p that defines how streched or shrunk it is when multiplied by M. And crucially, there are *no* other vectors besides the eigenvectors that, when multiplied by M, give you a vector pointing in the same direction; every other vector ends up pointing in some different direction after multiplying by M.

So why do we care about eigenvectors? Well, it turns out the eigenvectors turn up all over mathematics and science, from quantum physics to social science to image recognition and self-driving cars. Like principal components, they give you a reduced version of the data that contains much of the original data, but in a smaller form. And for temporal processes, they often describe where complex temporal structures with lots of feedback end up in equilibrium over time. If you haven't heard of them, they are probably the most important mathematical construct you've never heard of.

So getting back to the math, every matrix M has p eigenvectors and p eigenvalues associated with it: $\{v_1, \lambda_1\}, \{v_2, \lambda_2\}, \{v_3, \lambda_3\} \ldots$. We order them by the size of λ_p: the eigenvector with the largest λ_p is the first eigenvector (and thus the first principal component), and so on for all the rest. And because eigenvector-based analysis is so ubiquitous, there are a lot of standard methods for finding the p pairs $\{v_1, \lambda_1\}, \{v_2, \lambda_2\}, \{v_3, \lambda_3\} \ldots$ for any given matrix M.

Example estimation

So to summarize, one classic procedure for finding the principal components (ie, the values $\{\phi_{p1}, \phi_{p2}, \ldots\}$ and the projections $C_1, C_2 \ldots$) is:

1. Standardize the variables
2. Create the covariance matrix
3. Find the eigenvectors and eigenvalues for that matrix.
4. Choose how many of them we want to keep and analyze.

To put this into more concrete terms, lets look at some real data. Let's use the `psych` package, a package of useful functions and example datasets for analyzing psychological data. Inside, it has the "Motivational State Questionnaire" dataset, a set of 3896 individual responses to a battery of mood questions. The subset we're interested in here is 71 questions that were asked of each subject about their mood at the time of the survey: how happy, anxious, alert, sleepy, surprised, etc, etc, they were at that moment on a 0-3 scale. The subsantive question is, there are a lot of mood questions out there, but do people really have 71 different moods at any given time? Or are there a few underlying emotional factors that explain most of how you are at any given moment, and the myriad specific moods are just different ways of measuring the simpler, underlying factors? For instance, in the study that collected these data, the researchers started with the common assumption that one of the dominant underlying factors is whether you are happy or sad at any given moment; this level of happiness vs sadness shapes many of the other moods people report at any given moment, and is sort of like a basic thermometer reading for anyone at any time. At least, that is the common theory these scientists were testing.

So let's get the data, clean it up a bit, and take a quick look at it:

```r
#install.packages("psych")
library(psych)
data(msq)
msq1 <- msq[,2:72]
msq1[msq1=="9"] = NA
msq1 <- data.frame(msq1)
msq2 <- na.omit(msq1)
names(msq2)
```

```
 [1] "afraid"       "alert"        "angry"        "anxious"
 [5] "aroused"      "ashamed"      "astonished"   "at.ease"
 [9] "at.rest"      "attentive"    "blue"         "bored"
[13] "calm"         "cheerful"     "clutched.up"  "confident"
[17] "content"      "delighted"    "depressed"    "determined"
[21] "distressed"   "drowsy"       "dull"         "elated"
[25] "energetic"    "enthusiastic" "excited"      "fearful"
[29] "frustrated"   "full.of.pep"  "gloomy"       "grouchy"
[33] "guilty"       "happy"        "hostile"      "idle"
[37] "inactive"     "inspired"     "intense"      "interested"
[41] "irritable"    "jittery"      "lively"       "lonely"
[45] "nervous"      "placid"       "pleased"      "proud"
[49] "quiescent"    "quiet"        "relaxed"      "sad"
[53] "satisfied"    "scared"       "serene"       "sleepy"
[57] "sluggish"     "sociable"     "sorry"        "still"
[61] "strong"       "surprised"    "tense"        "tired"
[65] "tranquil"     "unhappy"      "upset"        "vigorous"
[69] "wakeful"      "warmhearted"  "wide.awake"
```

```r
dim(msq2)
```

```
[1] 1747   71
```

```r
msq2[1:5,1:5]
```

```
    afraid alert angry anxious aroused
1        1     1     0       1       1
2        0     1     0       0       0
3        0     0     0       0       0
4        0     1     0       1       1
103      0     0     1       0       0
```

That's a lot of columns and a lot of variables. Let's do some factor analysis on it to see what, if any, are the underlying dimensions.

Example 2

The `psych` package also has a very robust factor analysis function, `fa()`. It actually does much more than simply finding the eigenvectors of the covariance matrix. One of the most important tweaks is that it rotates the principal components a little bit to nudge things so that each component loads only on a few variables, and not on any of the rest. Without the rotation, we might get for the first principal component that it is 40% X1, 30% X2, 20% X3, and 10% X4; after the rotation, we might get 60% X1 and 40% X2, and none of the rest. But as we will see, often this extra rotation doesn't make much difference, and ultimately we end up with the same eigenvectors as ever.

```
fact <- fa(msq2,nfactors=2)
fact1 <- fact$loadings[,1]
```

Here we are only calculating the first two factors, but that's enough for our purposes for now. The second line extracts from the output the *loadings*, or the projections of the variables onto the factors, $\{\phi_{11}, \phi_{12}, \dots\}$. These are essential for interpreting the factors: if the first factor loads highly positive on all the happy variables and highly negative on all the unhappy variables, then we can know this factor is probably a happiness-unhappiness dimension. But if it loads on some other set of variables, maybe some other underlying factor explains the biggest variance in mood.

Example 3

The best way to read the factor, then, is to look not at the loadings as ordered by the variables (ie, $\{\phi_{11}, \phi_{12}, \dots\}$), but rather to order them from largest to smallest, so we can see what variables load most positively, and what variables load most negatively: ie, what are the two poles along which the first latent dimension varies.

```
fact1[order(fact1)]
```

```
   sluggish       tired    inactive      sleepy      drowsy
-0.503894778 -0.499009020 -0.479337232 -0.453446700 -0.452279643
       dull       bored        idle       quiet      grouchy
-0.451339402 -0.330342109 -0.282364229 -0.269818502 -0.234994186
     gloomy       still   irritable   depressed     unhappy
-0.230250716 -0.212145514 -0.195875465 -0.183939118 -0.163483072
       blue      placid      lonely     hostile         sad
-0.159041691 -0.107262862 -0.090452613 -0.086442759 -0.079305286
      upset  frustrated       angry  distressed    tranquil
-0.057779239 -0.038305419 -0.002635468  0.011474854  0.025122472
      sorry     ashamed        calm      guilty   quiescent
 0.039702915  0.057750178  0.080191644  0.088639846  0.089099944
    fearful      serene      afraid  clutched.up      scared
 0.105974727  0.114540925  0.114735548  0.115292799  0.146308753
      tense     at.rest     relaxed     nervous     at.ease
 0.146348183  0.173458620  0.192944774  0.215194555  0.279960120
    anxious     jittery  astonished    surprised     intense
 0.298239271  0.321695437  0.339180797  0.391128880  0.479498021
    content    satisfied  warmhearted    confident       proud
 0.556118415  0.567897327  0.570353331  0.591782989  0.616313051
     strong   interested    delighted     sociable   determined
 0.621646642  0.630228830  0.649276754  0.651824470  0.659586367
    wakeful     inspired     pleased     aroused    attentive
 0.682711025  0.696233314  0.698687364  0.704613580  0.707349417
      happy   wide.awake       alert      elated     vigorous
 0.718339485  0.718801402  0.736859382  0.741358314  0.755506112
   cheerful enthusiastic     excited  full.of.pep      lively
 0.777665930  0.791972877  0.813047207  0.838803443  0.850925710
  energetic
 0.862065244
```

This is a lot to look at, but we can immediately tell what variables go into the first factor by looking at the lowest- and highest-scoring variables (the sign, ie, which direction is positive vs negative, is arbitrary and meaningless). What we see is that while one end is happy, enthusiastic, cheerful, full.of.pep, lively, and energetic, the other side is not unhappy so much as sluggish, tired, dull, sleepy, etc. Ie, the most important factor underlying all these 71 moods is not happy vs unhappy, but more like energetic (and happy) vs tired, sleepy, and depressed. An important discovery!

Example 4

So what is the second factor? Is it happy/unhappy? Or some other dimension of mood? We can again examine it from the `fa()` output:

```
fact2 <- fact$loadings[,2]
fact2[order(fact2)]
```

```
      at.ease       relaxed          calm       content      tranquil 
-4.266090e-01 -4.116543e-01 -3.831604e-01 -3.038816e-01 -2.980951e-01 
      at.rest        serene     satisfied         happy   warmhearted 
-2.792488e-01 -2.730150e-01 -2.711652e-01 -2.154788e-01 -1.645754e-01 
    confident       pleased      sociable      cheerful         still 
-1.457839e-01 -1.438760e-01 -1.326938e-01 -1.281305e-01 -1.094916e-01 
       placid     delighted  enthusiastic         proud       wakeful 
-9.851339e-02 -8.727590e-02 -3.046087e-02 -9.626961e-03 -8.524701e-03 
   interested          idle   full.of.pep     attentive     quiescent 
-3.748504e-03  3.752201e-05  7.299197e-03  9.072171e-03  1.207486e-02 
   wide.awake        lively         alert      inactive        elated 
 1.295675e-02  1.299289e-02  2.011185e-02  2.257429e-02  2.919448e-02 
    energetic        strong         quiet      vigorous         bored 
 3.581410e-02  7.626629e-02  9.558207e-02  1.040518e-01  1.094262e-01 
      excited        drowsy        sleepy         tired       aroused 
 1.140680e-01  1.162290e-01  1.283373e-01  1.329017e-01  1.435096e-01 
     inspired      sluggish          dull    determined     surprised 
 1.722510e-01  1.738016e-01  2.329041e-01  2.500367e-01  2.789269e-01 
   astonished       jittery       intense        guilty       anxious 
 3.533054e-01  3.964455e-01  4.352757e-01  5.340407e-01  5.376114e-01 
       lonely       hostile       grouchy     irritable       ashamed 
 5.387408e-01  5.535441e-01  5.569419e-01  5.814694e-01  5.866371e-01 
  clutched.up       nervous         sorry        afraid       fearful 
 6.040513e-01  6.096402e-01  6.219696e-01  6.327035e-01  6.341214e-01 
       gloomy        scared          blue     depressed         angry 
 6.372932e-01  6.378026e-01  6.442801e-01  6.628803e-01  6.708130e-01 
      unhappy         tense           sad         upset    distressed 
 6.870229e-01  6.932115e-01  6.974707e-01  7.356378e-01  7.410766e-01 
   frustrated                                                        
 7.443217e-01                                                        
```

Now we see a second dimension that is interestingly different from the first: not energetic vs sleepy, but relaxed and calm versus tense and frustrated. Here too, the factor is both not quite what we expected, but at the same time familiar and plausible.

Within the `fa()` output are also the projections of individuals onto each of these dimensions (ie, how energetic-vs-tired someone seems to be, and how calm-vs-tense they seem to be, and so on for all the other factors we might estimate), and a large number of other diagnostics and outputs (too many to show here!). There's a whole world to explore in factor analysis, but the basic truth is that the fundamental factors are the same however we estimate them.

Other estimation methods

For instance, here are two other methods for estimating principal components, both built into the base R package, but using different approaches. The first uses *singular value decomposition* (another unsupervised approach similar to factor analysis), and the second uses the covariance/eigenvector approach.

```
# prcomp method using SVD
pcaA <- prcomp(msq2)
pcaA1 <- pcaA$rotation[,1]

# Princomp method using eigen of cov
pcaB <- princomp(msq2)
pcaB1 <- pcaB$loadings[,1]
```

Finally, let's do it manually ourselves using `cov()` to estimate the covariance matrix and `eigen()` to get the eigenvectors:

```
# Direct eigen of cov
covm <- cov(msq2)
eigenm <- eigen(covm)
eigen1 <- eigenm$vectors[,1]
```

How similar are these methods?

```
f1mat <- cbind(fact1,pcaA1,pcaB1,eigen1)
cor(f1mat)
```

```
            fact1      pcaA1      pcaB1     eigen1
fact1   1.0000000  0.9733082 -0.9733082 -0.9733082
pcaA1   0.9733082  1.0000000 -1.0000000 -1.0000000
pcaB1  -0.9733082 -1.0000000  1.0000000  1.0000000
eigen1 -0.9733082 -1.0000000  1.0000000  1.0000000
```

Essentially identical, with `fa()` being slightly different due to the rotation – but not very different.

Estimation bonus

One final method for estimating the components directly from the data is perhaps a bit surprising. Because eigenvectors are, in a sense, both the underlying truth of the data and the equilibrium of any process that the data describes, one additional way to find the first eigenvector / principal component is to take any random vector v_a of length p and multiply it by the data matrix; then take the resulting vector v_b (now of length n) and multiply it back through the matrix to get another vector v'_a, and so on back and forth. If you keep doing this a few times, ultimately the process converges so that the final v_a is

almost identical to the first eigenvector. (This is basically how singular value decomposition works.) It's a crude but very effective way to get the first component (and can be adapted to get the others), and works both quickly and well on even very large datasets.

```
tm.msq2 <- t(as.matrix(msq2))
m.msq2 <- as.matrix(msq2)
va <- rnorm(ncol(msq2))
for(i in 1:10){
  vb <- m.msq2 %*% scale(va)
  va <- tm.msq2 %*% scale(vb)
}
```

(Note how we have to rescale va and vb each time, otherwise they increase because the eigenvalue stretches the eigenvector each time it's multiplied through the matrix. We only care about the direction of v (ie, the relative sizes of the loadings), not the absolute size, so we rescale it to keep it from getting unwieldly big.)

So does the final va match the first component?

```
cor(va,pcaA1)
```

```
           [,1]
[1,] -0.9994921
```

Yes it does.

How many factors to keep?

Since factor analysis theoretically can produces as many factors as there were variables originally (p), but one of the purposes of factor analysis is to reduce a high-p dataset to a smaller set of factors, one might naturally wonder how many factors do we have to keep? Can a 100-variable dataset be reduced to just three factors that explain most of the variation? How do we know how many to keep?

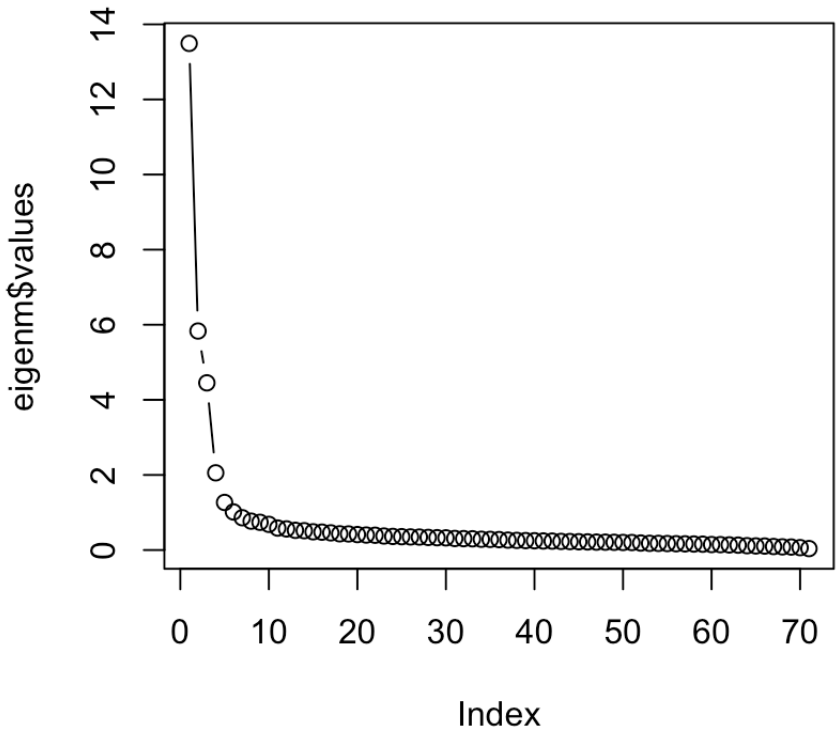
It turns out this question is a vexed and ongoing one, and has been for decades. Ideally we would want a nice statistical test to tell us that the first k factors are statistically significant in some sense, and the remaining ones are just picking up on random noise. But despite many suggested such tests, the community has not really settled on a single best test.

Perhaps one of the oldest and most established methods is looking at the eigenvalues, which in a sense determine the relative importance of each of the factors: the bigger the eigenvalue, the more of the variation in the original data that eigenvector/factor explains.

Scree plot

One common way of examining these values is to plot them, ordered by size, in something called a "scree plot" (because it looks the the scree on the side of a mountain). All of our estimation methods above include the values as well as the factors, and here is a simple scree plot of them from our manual eigenvector method.

```
plot(eigenm$values,type="b")
```

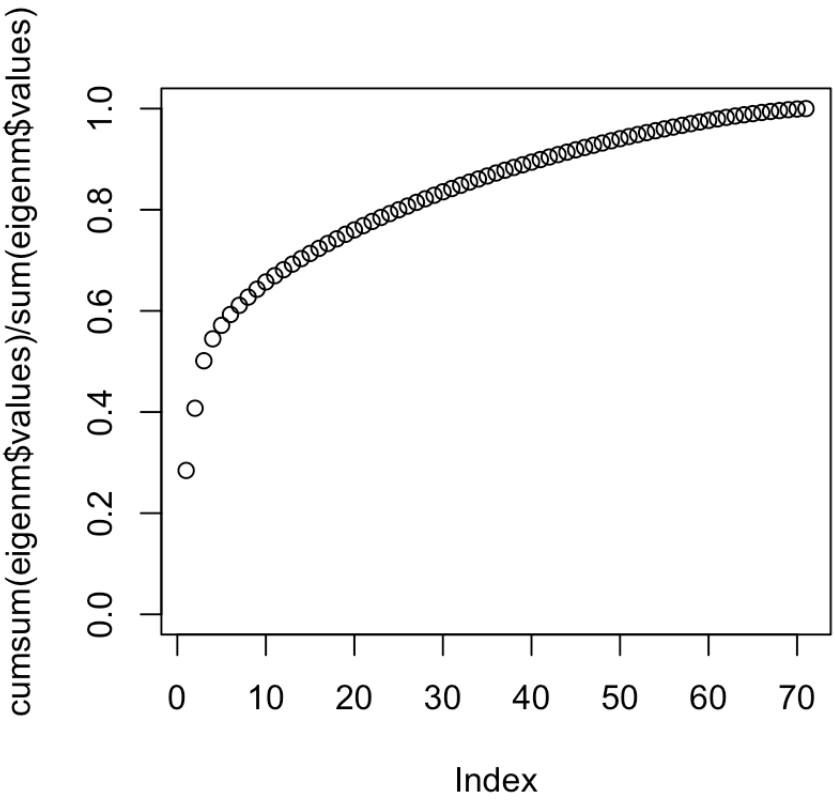


There are two common criteria to use for choosing the number of factors based on an examination of these values. First, one looks for the "elbow" in the curve – where it goes from the steep decline, the the flat area, where the presumption is the flat are is all the factors that are just noise. So in this figure, counting from the left, it might be 5 or so factors that are significant, and the rest are just noise. Another method is to only keep those factors with values above 1 (because that distinguishes eigenvectors that grow vs those that shrink), which in this case yields 6 factors to retain – so the two methods are in fairly close agreement.

Cumulative variance

One final way to see this is the plot the proportion of the total variance in the data explained by the factors as you add them together. Since the eigenvalues are equivalent to the amount of variance each component explains in the original data, we can just do the same plot as before, but use the `cumsum` function to plot the cumulative amount (`cumsum` just turns a vector v of i values in a vector c where each value c_i is the sum of all the v_i up to the ith), and normalize it by dividing by the total.

```
plot(cumsum(eigenm$values)/sum(eigenm$values),ylim=c(0,1))
```



We can see that this is a fairly smooth curve, with a bit of an elbow at around the fourth or fifth value – about the same as before. More interestingly, over 50% of the total variance in the data is explained by just those first 4 factors, so only 4 out 71 factors are already explaining over half of the variation. This plus the previous tests suggests that 4-6 factors do explain quite a lot of what's going on in the variation of mood among people over time. We may not be 70-dimensional emotional thinkers, but 4-6 seems like a plausibly complex psychological model.