Lab: Scale Development and Factor Analysis

PHS2000B

1 May 2023

Contents

Acknowledgment: Thanks to Louisa Smith for extensive work on an earlier version of the exploratory data analysis and factor analysis materials.

1 The Centers for Epidemiologic Studies Depression inventory (CES-D)

Today we will be using data from the CES-D (Centers for Epidemiologic Studies Depression inventory)[1] from the National Health and Nutrition Examination Survey I (1971-1974). This self-report depression scale was developed to assess depressive symptoms in the general population, and has found to be both reliable and valid in a nubmer of populations.[2]

However, there are a lot of ways that depression can manifest, and two people who have equally strong depression could have symptoms along different dimensions, such as loss of interest in activities, feeling sad, or experiencing psychosomatic pain. We will be using responses to the 20 questions that make up the CES-D to illustrate the exploratory analyses you will be conducting for your Measurement Project, including using factor analysis to explore the underlying dimensions of the scale.

We'll restrict our analysis to white women and use only those observation with complete data on all 20 of the items. These data are included in the file `nhanes1_cesd.csv`.

The 20 questions ask about the subject's experiences in the past week and are scored on a four point scale:

- 0 = Rarely or none of the time (less than 1 day)
- 1 = Some or a little of the time (1-2 days)
- 2 = Occasionally or a moderate amount of time (3-4 days)
- 3 = Most or all of the time (5-7 days)

The items include:

1. I was bothered by things that usually don't bother me.
2. I did not feel like eating; my appetite was poor.
3. I felt that I could not shake off the blues even with help from my family or friends.
4. I felt that I was just as good as other people.
5. I had trouble keeping my mind on what I was doing.
6. I felt depressed.
7. I felt that everything I did was an effort.
8. I felt hopeful about the future.
9. I thought my life had been a failure.
10. I felt fearful.
11. My sleep was restless.
12. I was happy.
13. I talked less than usual.
14. I felt lonely.
15. People were unfriendly.

[1] Radloff LS. The CES-D scale: a self-report depression scale for research in the general population. Applied Psychological Measurement. 1977;1:385-401.

[2] A lot of information and references about the CES-D can be found here: https://www.sralab.org/rehabilitation-measures/center-epidemiological-studies-depression-scale-ces-d

16. I enjoyed life.
17. I had crying spells.
18. I felt sad.
19. I felt that people disliked me.
20. I could not get going.

2 Scale Development

Our goal for the measurement project is to develop a scale that measures a latent construct. To accomplish this, you have generated an item pool, reviewed the items for content, and will be collecting empirical data from a sample of respondents.

Once you have the data collected, we've asked you to compute:

1. item means
2. item variances (or standard deviations)
3. the correlation matrix of all of the items
4. run an exploratory factor analysis to explore the underlying factor structure of your instrument
5. reliability (internal consistency)
6. uncorrected and corrected item-scale correlations

We'll illustrate each of these steps below, making use of the functions in the `psych` package. Note that there is a lot of useful information and code in this document and we don't expect you to absorb it all in one sitting. As you work on your measurement project data analysis, refer back to this document and feel free to bring your questions to class or office hours or to post on the discussion board.

2.1 Exploring item means, standard deviations, and distributions

```
# Here is a quick function to output nobs, mean, std, min, and max
f.summarize <- function(x){
data.frame(obs=apply(x,2,length),mean=apply(x,2,mean),
           std=apply(x,2,sd), min=apply(x,2,min), max=apply(x,2,max))
}

kable(f.summarize(cesd), digits = 3,
      caption = "Mean, standard deviation, minimum, and maximum of CES-D items",
      booktabs = T, longtable = T)
```

Table 1: Mean, standard deviation, minimum, and maximum of CES-D items

	obs	mean	std	min	max
bothered	1427	0.396	0.699	0	3
appetite	1427	0.298	0.687	0	3
blues	1427	0.280	0.673	0	3
good	1427	0.735	1.186	0	3
mind	1427	0.479	0.779	0	3
depressed	1427	0.531	0.770	0	3
effort	1427	0.587	0.889	0	3
hopeful	1427	0.845	1.148	0	3
failure	1427	0.201	0.550	0	3
fearful	1427	0.311	0.652	0	3
sleep	1427	0.739	0.932	0	3
happy	1427	0.633	0.949	0	3
talk	1427	0.488	0.801	0	3
lonely	1427	0.458	0.796	0	3
unfriendly	1427	0.180	0.538	0	3
enjoy	1427	0.585	0.984	0	3
cry	1427	0.245	0.590	0	3
sad	1427	0.477	0.709	0	3
dislike	1427	0.153	0.474	0	3
getgoing	1427	0.603	0.807	0	3

- Why might we want to look at item means and standard deviations?

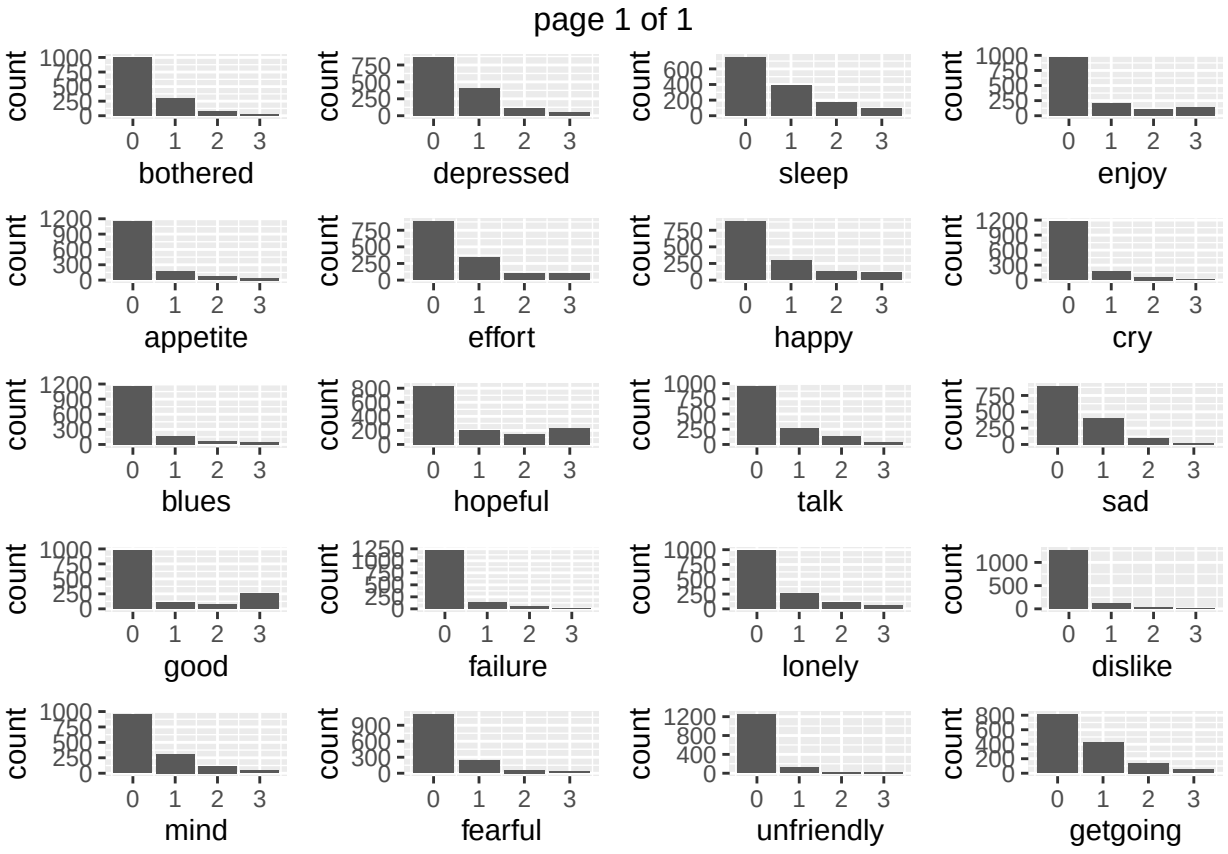
It may also be helpful to visualize the distribution of responses to each item.

```
# use ggplot to output summary histograms of all CES-D items.
# note: lapply repeatedly calls ggplot to generate the plots
# marrangeGrob from the gridExtra package lays out the plots
```

```
plot.items <- lapply(names(cesd)[1:20],
       function(n)
         ggplot(data=cesd, aes_string(x=n)) +
         geom_bar())
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation ideoms with `aes()`
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
ml <- marrangeGrob(plot.items, nrow=5, ncol=4)
ml
```



2.2 Computing and visualizing the correlation matrix

Now let's look at the correlation matrix for the items. This should give you some idea of whether your items are measuring the same thing (though it won't tell you if you're measuring the right thing!).

```
cor_mat <- cor(cesd)
rownames(cor_mat) <- colnames(cor_mat) <- names(cesd)
```

```
panderOptions("round",3)
pander(cor_mat, caption="CES-D item correlation matrix")
```

Table 2: CES-D item correlation matrix (continued below)

	bothered	appetite	blues	good	mind	depressed
bothered	1	0.299	0.45	0.024	0.312	0.48
appetite	0.299	1	0.375	0.001	0.265	0.349
blues	0.45	0.375	1	-0.014	0.363	0.592
good	0.024	0.001	-0.014	1	-0.021	0.024
mind	0.312	0.265	0.363	-0.021	1	0.429
depressed	0.48	0.349	0.592	0.024	0.429	1
effort	0.324	0.281	0.388	0.024	0.327	0.437
hopeful	0.145	0.101	0.161	0.433	0.102	0.166
failure	0.296	0.293	0.447	0.088	0.38	0.465
fearful	0.366	0.283	0.432	-0.002	0.401	0.47
sleep	0.265	0.277	0.327	0.011	0.34	0.396
happy	0.242	0.165	0.277	0.343	0.185	0.324
talk	0.27	0.257	0.306	0.021	0.334	0.334
lonely	0.344	0.273	0.477	0.004	0.338	0.551
unfriendly	0.241	0.134	0.221	0.078	0.177	0.218
enjoy	0.182	0.085	0.22	0.343	0.148	0.244
cry	0.35	0.262	0.471	0.075	0.311	0.491
sad	0.451	0.318	0.549	0.028	0.385	0.642
dislike	0.255	0.168	0.266	0.078	0.284	0.331
getgoing	0.376	0.267	0.372	0.023	0.421	0.466

Table 3: Table continues below

	effort	hopeful	failure	fearful	sleep	happy	talk
bothered	0.324	0.145	0.296	0.366	0.265	0.242	0.27
appetite	0.281	0.101	0.293	0.283	0.277	0.165	0.257
blues	0.388	0.161	0.447	0.432	0.327	0.277	0.306
good	0.024	0.433	0.088	-0.002	0.011	0.343	0.021
mind	0.327	0.102	0.38	0.401	0.34	0.185	0.334
depressed	0.437	0.166	0.465	0.47	0.396	0.324	0.334
effort	1	0.099	0.345	0.308	0.385	0.185	0.249
hopeful	0.099	1	0.174	0.122	0.099	0.499	0.074
failure	0.345	0.174	1	0.524	0.298	0.261	0.275
fearful	0.308	0.122	0.524	1	0.349	0.27	0.236
sleep	0.385	0.099	0.298	0.349	1	0.204	0.282
happy	0.185	0.499	0.261	0.27	0.204	1	0.061

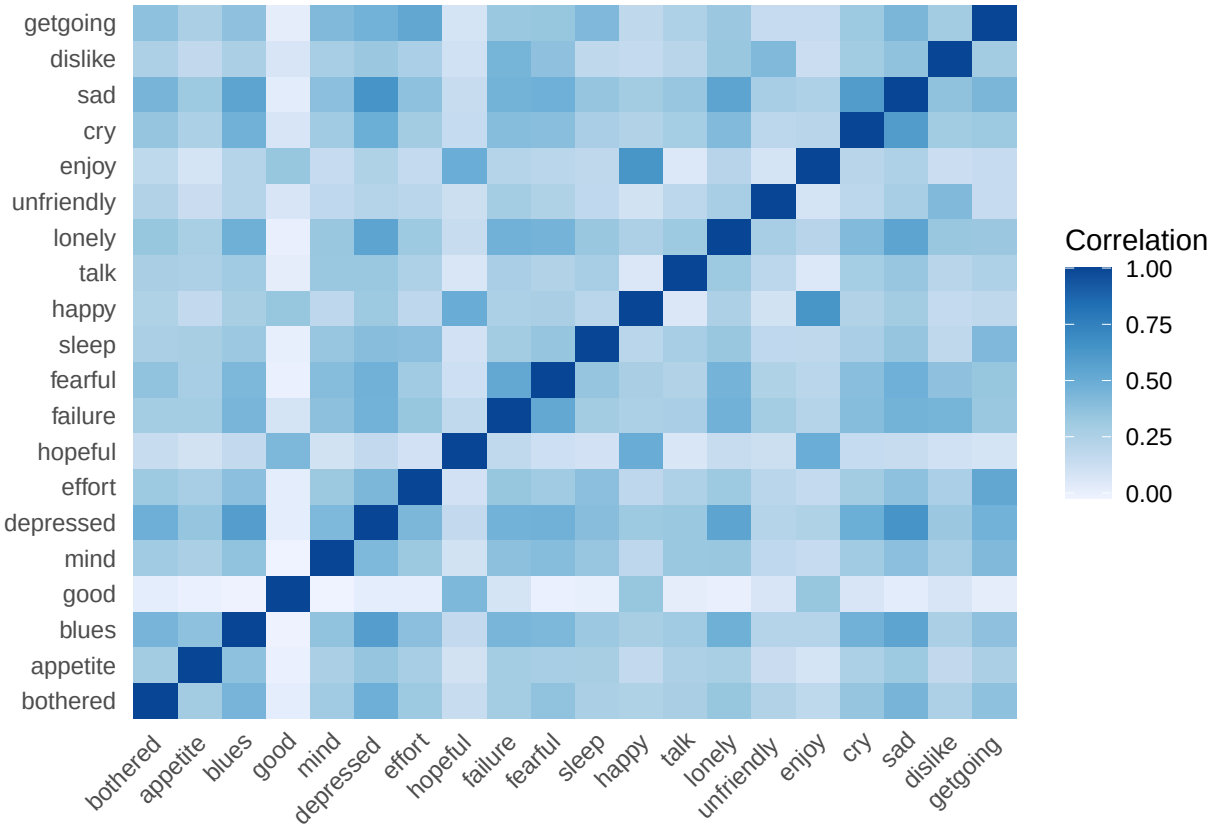
	effort	hopeful	failure	fearful	sleep	happy	talk
talk	0.249	0.074	0.275	0.236	0.282	0.061	1
lonely	0.323	0.141	0.472	0.456	0.335	0.254	0.318
unfriendly	0.201	0.124	0.295	0.242	0.176	0.102	0.193
enjoy	0.156	0.494	0.218	0.2	0.18	0.639	0.059
cry	0.298	0.153	0.402	0.391	0.275	0.236	0.292
sad	0.376	0.144	0.461	0.479	0.347	0.3	0.34
dislike	0.264	0.107	0.455	0.372	0.181	0.156	0.206
getgoing	0.534	0.088	0.333	0.346	0.426	0.181	0.251

	lonely	unfriendly	enjoy	cry	sad	dislike	getgoing
bothered	0.344	0.241	0.182	0.35	0.451	0.255	0.376
appetite	0.273	0.134	0.085	0.262	0.318	0.168	0.267
blues	0.477	0.221	0.22	0.471	0.549	0.266	0.372
good	0.004	0.078	0.343	0.075	0.028	0.078	0.023
mind	0.338	0.177	0.148	0.311	0.385	0.284	0.421
depressed	0.551	0.218	0.244	0.491	0.642	0.331	0.466
effort	0.323	0.201	0.156	0.298	0.376	0.264	0.534
hopeful	0.141	0.124	0.494	0.153	0.144	0.107	0.088
failure	0.472	0.295	0.218	0.402	0.461	0.455	0.333
fearful	0.456	0.242	0.2	0.391	0.479	0.372	0.346
sleep	0.335	0.176	0.18	0.275	0.347	0.181	0.426
happy	0.254	0.102	0.639	0.236	0.3	0.156	0.181
talk	0.318	0.193	0.059	0.292	0.34	0.206	0.251
lonely	1	0.28	0.214	0.415	0.549	0.334	0.329
unfriendly	0.28	1	0.088	0.193	0.281	0.42	0.15
enjoy	0.214	0.088	1	0.209	0.252	0.133	0.147
cry	0.415	0.193	0.209	1	0.597	0.303	0.321
sad	0.549	0.281	0.252	0.597	1	0.368	0.439
dislike	0.334	0.42	0.133	0.303	0.368	1	0.3
getgoing	0.329	0.15	0.147	0.321	0.439	0.3	1

The full correlation matrix of all 20 items is hard to absorb. Sometimes it's more informative to look at a graphical depiction of the correlation matrix. There are many ways to do this; here's one:

```
# This function turns the correlation matrix into a "long" dataset for use with ggplot
cor_dat <- melt(cor_mat, value.name = "Correlation")

# We are going to use the cor_dat dataframe, with Var1 on the x-axis and
# Var2 on the y-axis, and we are going to fill in the colors according
# to the correlation values
```

```
ggplot(cor_dat, aes(Var1, Var2, fill = Correlation)) +
# Each data point is going to be portrayed as a colored tile
  geom_tile() +
# This function helps make pretty colors
  scale_fill_distiller(type = "seq", palette = "Blues", direction=1) +
# A bunch of formatting things I always have to Google
  theme(axis.title = element_blank(), panel.background = element_blank(),
        panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        axis.text.x = element_text(angle = 45, hjust = 1.1, vjust = 1.1),
        axis.ticks = element_blank())
```



- What patterns can you see in the visual display of the correlation matrix?

3 Factor analysis

We can think of factor analysis as giving us a formal way to characterize the patterns in the correlation matrix.

3.1 Factor analytic model

We'll define $\mathbf{X}_i$ as the vector of manifest variables: a set of p observable items corresponding to individual i.[3]

$$\mathbf{X}_i = (X_{i1}, X_{i2}, \ldots, X_{ip})^T$$

In our example, these are the values assigned to the answers "most or all of the time" (3), "occasionally" (2), and "some or a little of the time" (1), and "rarely" (0) in response to the series of $p = 20$ questions asked of each participant.

Each of the X_{ij} has a mean value, μ_j, which we don't really care about (we only care about the *relationship* between our manifest variables and the underlying factors). We can center $\mathbf{X}_i$ by subtracting the item-specific means, so that we have $\mathbf{X}_i - \boldsymbol{\mu}$, which is still a vector of length p:

$$\mathbf{X}_i - \boldsymbol{\mu} = (X_{i1} - \mu_1, X_{i2} - \mu_2, \ldots, X_{ip} - \mu_p)^T$$

On the left-hand side we now have the item scores relative to their means – in other words, whether a person's response was exceptionally high, low, or close to average.

Now let's assume that each of these centered, observable random variables is some linear combination of unobservable random variables. In particular, each contains some (possibly zero-valued) contribution from each of the q latent factors ($q < p$), along with some added random error.

In our example, those factors may reflect latent constructs such as depressed affect, interpersonal functioning, etc. We may or may not know what we expect these factors should be.[4]

For each item X_{ij}, $j = 1, \ldots, p$, for individual i (that is, for each item in the above vector), this looks like:

$$X_{ij} - \mu_j = \lambda_{j1}F_{i1} + \lambda_{j2}F_{i2} + \ldots + \lambda_{jq}F_{ij} + e_{ij}$$

So we assume that the jth (centered) item for person i is made up of a linear combination of their q factors $\mathbf{F}_i$ (a vector of length q, specific to individual i) and the associated loadings $\boldsymbol{\lambda}_j$ (also a vector of length q, but it will be the same for every i – it's a constant), plus a random error.

Now you can see why matrix notation might be useful! We can represent this model for each of the p items in the following form, where each of those mean-centered items is a row:

$$\begin{bmatrix} X_{i1} \\ X_{i2} \\ \vdots \\ X_{ip} \end{bmatrix} - \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_q \end{bmatrix} = \begin{bmatrix} \lambda_{11} & \lambda_{12} & \ldots & \lambda_{1q} \\ \lambda_{21} & \lambda_{22} & \ldots & \lambda_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{p1} & \lambda_{p2} & \ldots & \lambda_{pq} \end{bmatrix} \begin{bmatrix} F_{i1} \\ F_{i2} \\ \vdots \\ F_{iq} \end{bmatrix} + \begin{bmatrix} e_{i1} \\ e_{i2} \\ \vdots \\ e_{ip} \end{bmatrix}$$

[3] For quick reference: i will index individuals: $i = 1, \ldots, n$; j will index items: $j = 1, \ldots, p$; k will index factors: $k = 1, \ldots, q$

[4] *Exploratory* factor analysis involves trying to determine the factor structure from the observed data, while *confirmatory* factor analysis involves testing how well a pre-defined factor structure fits the data. We're focusing on exploratory factor analysis here.

which corresponds to:

$$\mathbf{X}_{p\times 1} - \boldsymbol{\mu}_{p\times 1} = \boldsymbol{\Lambda}_{p\times q}\mathbf{F}_{q\times 1} + \mathbf{e}_{p\times 1}$$

Note that on both sides of the equation we end up with a column vector of length p.[5]

Writing the model out like this helps us see which components of the model are random variables and which are fixed constants:

$$\mathbf{X}_i - \boldsymbol{\mu} = \boldsymbol{\Lambda}\mathbf{F}_i + \mathbf{e}_i$$

This looks somewhat like the setup for other regression models you may have seen. However, note that we don't observe *anything* on the right-hand side of this model! And yet our goal is to estimate the parameters $\boldsymbol{\Lambda}$ – the loadings that link each factor to each item.

We can take the covariance of both sides:

$$\text{Cov}(\mathbf{X}_i - \boldsymbol{\mu}) = \text{Cov}(\boldsymbol{\Lambda}\mathbf{F}_i + \mathbf{e})$$

First we can replace $\text{Cov}(\mathbf{X}_i - \boldsymbol{\mu})$ with $\boldsymbol{\Sigma}$, the covariance matrix of $\mathbf{X}_i$, since $\boldsymbol{\mu}$ is a constant vector.

Since we are assuming $\mathbf{F} \perp\!\!\!\perp \mathbf{e}$, we can split the sum on the right-hand side. Then we can pull out the loadings from $\text{Cov}(\boldsymbol{\Lambda}\mathbf{F}_i)$, since it is a matrix of constants, not random variables.[6]

Now we have

$$\boldsymbol{\Sigma} = \boldsymbol{\Lambda}\text{Cov}(\mathbf{F}_i)\boldsymbol{\Lambda}^T + \text{Cov}(\mathbf{e}_i)$$

The next assumption we need to invoke is that $\text{Cov}(\mathbf{F}_i) = \mathbf{I}$. That is, we assume that the factors are uncorrelated (we will loosen this assumption when we do *oblique rotation* below). This means that $\text{Cov}(\mathbf{F}_i)$ falls out – very conveniently, since we can't observe $\mathbf{F}_i$! We also assume that the errors are independent with a diagonal variance-covariance matrix $\boldsymbol{\Psi}$, leaving us with:

$$\boldsymbol{\Sigma} = \boldsymbol{\Lambda}\boldsymbol{\Lambda}^T + \boldsymbol{\Psi}$$

We can observe values that allow us to estimate the left-hand side. Factor analysis is all about trying to partition the observed variance-covariance matrix of the items into the matrix for the loadings, and the diagonal matrix representing the contributions of the random error.

[5] If we wanted to represent this model in terms of the observed data, with its n observations, we could instead have a $p \times n$ matrix on each side of the equation. Recall that we can add and subtract matrices as long as their dimensions match, and we can multiply matrices as long as their inner dimensions match, which produces a matrix with dimension equal to their outer dimensions. If you would like some matrix algebra practice, I encourage you to write out the matrix operations above and show that each entry of the matrix we would have on the right-hand side is of the form above: $\lambda_{j1}F_{i1} + \lambda_{j2}F_{i2} + \ldots + \lambda_{jq}F_{iq} + e_{ij}$.

[6] We could also write $\text{Cov}(\boldsymbol{\Lambda}\mathbf{F}_i)$ as $\text{Var}(\boldsymbol{\Lambda}\mathbf{F}_i)$; they are equivalent. In both cases, since we are working with a multivariable $\mathbf{F}_i$, we are referring to its variance-covariance matrix. Recall that we can pull out a constant from the variance of a single random variable, $\text{Var}(aX) = a^2\text{Var}(X)$, and from the covariance $\text{Cov}(aX, bY) = ab\text{Cov}(X, Y)$. When we're dealing with multivariable $\mathbf{F}_i$, we have both situations (the diagonal of a covariance matrix is composed of the variances of each individual F). But we don't observe $\mathbf{F}_i$ directly; instead, we observe p linear combinations of the q-dimensional $\mathbf{F}_i$. We are representing that transformation with the $p \times q$ matrix $\boldsymbol{\Lambda}$. In such a situation, we can factor out the constants from the variance-covariance matrix of a multivariable random variable $\mathbf{X}$ by writing $\text{Cov}(\mathbf{AX}) = \mathbf{A}\text{Cov}(\mathbf{X})\mathbf{A}^T$.

3.2 Model fitting: The principal axis approach

We're going to walk through one way to fit this model using the principal axis approach (also called principal factor analysis). This is primarily of interest as a heuristic for helping us understand how we go about decomposing the correlation matrix of the manifest variables.

3.2.1 Eigendecomposition

Let's assume all the observed variables are standardized to have variance 1, so the covariance matrix is equal to the correlation matrix ($\text{corr}(\mathbf{X}_i) = \text{Cov}(\mathbf{X}_i) = \mathbf{\Sigma}$). We'll be using the correlation matrix to illustrate how the $\mathbf{\Lambda}$ matrix can be estimated.

When we have a symmetric matrix like the correlation matrix (remember that $\text{corr}(X, Y) = \text{corr}(Y, X)$, so all elements on opposite sides of the diagonal match), something called the spectral theorem tells us that we can always *diagonalize* it.

What this means for the correlation matrix $\mathbf{\Sigma}$ is that there is some matrix $\mathbf{A}$ and some diagonal matrix (which means that the only non-zero elements are on the diagonal) $\mathbf{\Delta}$ such that we can write

$$\mathbf{\Sigma} = \mathbf{A}\mathbf{\Delta}\mathbf{A}^T$$

This is called the *eigendecomposition*: the *eigenvectors* make up the columns of matrix $\mathbf{A}$ and the *eigenvalues* are the elements on the diagonal of $\mathbf{\Delta}$.

What is an eigenvalue? When we multiply a vector by a matrix, the vector is transformed in some way. For a given matrix, some vectors are just scaled by a factor. Those pairs of vectors and their scaling factors are called eigenvectors and eigenvalues. For our purposes, think of a matrix as having a certain amount of information in it. When we do eigendecomposition, an eigenvalue represent how much information is explained by its corresponding eigenvector – larger ones explain more; those near 0 mean there's little left over to explain after taking into account the other eigenvectors. If an eigenvalue is 0, that means its eigenvector is redundant: it's just a linear combination of the other eigenvectors.

Since $\mathbf{\Delta}$ is a diagonal matrix, it looks like this:

$$\mathbf{\Delta} = \begin{bmatrix} \delta_1 & 0 & \dots & 0 \\ 0 & \delta_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \delta_p \end{bmatrix} = \begin{bmatrix} \sqrt{\delta_1} & 0 & \dots & 0 \\ 0 & \sqrt{\delta_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sqrt{\delta_p} \end{bmatrix} \begin{bmatrix} \sqrt{\delta_1} & 0 & \dots & 0 \\ 0 & \sqrt{\delta_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sqrt{\delta_p} \end{bmatrix}$$

That is, we can split a diagonal matrix into the product of the diagonal matrix of the square roots of its elements – the eigenvalues.

Therefore, we can rewrite

$$\mathbf{\Sigma} = \mathbf{A}\mathbf{\Delta}\mathbf{A}^T = \mathbf{A}\sqrt{\mathbf{\Delta}}\sqrt{\mathbf{\Delta}}\mathbf{A}^T$$

Since we are post-multiplying $\mathbf{A}$ with $\sqrt{\mathbf{\Delta}}$ and pre-multiplying $\mathbf{A}^T$ with $\sqrt{\mathbf{\Delta}}$, we conveniently end up with a matrix and its transpose, which we'll call $\mathbf{\Lambda}^*$:

$$\mathbf{\Sigma} = \mathbf{A}\sqrt{\mathbf{\Delta}}\sqrt{\mathbf{\Delta}}\mathbf{A}^T = \mathbf{\Lambda}^*\mathbf{\Lambda}^{*T}$$

This looks almost exactly like what we have on the right-hand side of the factor analytic model! All that's left to do is pull out the uniquenesses.

3.2.2 Moving from $\mathbf{\Lambda}^*\mathbf{\Lambda}^{*T}$ to $\mathbf{\Lambda}\mathbf{\Lambda}^T + \mathbf{\Psi}$

We'll walk through an iterative approach to estimate the factor loadings. Our goal is to move from $\mathbf{\Lambda}^*\mathbf{\Lambda}^{*T}$ to $\mathbf{\Lambda}\mathbf{\Lambda}^T + \mathbf{\Psi}$. To do this, first note that $\mathbf{\Lambda}^*$ is of dimension p, because we perform the eigendecomposition on the correlation matrix of the observed variables, of which there are p. However, our goal is to reduce its dimensionality to q: we assume that there are fewer factors than there are items, so that the same factor contributes to multiple items.

We'll discuss later how to choose the number of factors, but right now assume $q = 4$.

First, we'll use the `eigen()` function in `R` to perform the eigendecomposition.

```
sigma <- cor(cesd)
eigenv <- eigen(sigma, symmetric = TRUE)
```

The resulting object contains the eigenvectors (`eigenv$vectors`) and eigenvalues (`eigenv$values`).

We can create the $\mathbf{\Lambda}^*$ matrix by multiplying the matrix of eigenvectors by the square root of the eigenvalues (that is, $\mathbf{\Lambda}^* = \mathbf{A}\sqrt{\mathbf{\Delta}}$).

```
lambda_star <- eigenv$vectors %*% sqrt(diag(eigenv$values))
```

We can confirm that $\mathbf{\Lambda}^*\mathbf{\Lambda}^{*T} = \mathbf{\Sigma}$:

```
(lambda_star %*% t(lambda_star))[1:5, 1:5]
```

```
##            [,1]        [,2]        [,3]         [,4]        [,5]
## [1,] 1.00000000 0.298982581  0.45019708  0.024236559  0.31162741
## [2,] 0.29898258 1.000000000  0.37518196  0.001359453  0.26506366
## [3,] 0.45019708 0.375181964  1.00000000 -0.014332813  0.36345124
## [4,] 0.02423656 0.001359453 -0.01433281  1.000000000 -0.02110194
## [5,] 0.31162741 0.265063660  0.36345124 -0.021101944  1.00000000
```

```
sigma[1:5, 1:5]
```

```
##            bothered    appetite       blues         good        mind
## bothered 1.00000000 0.298982581  0.45019708  0.024236559  0.31162741
## appetite 0.29898258 1.000000000  0.37518196  0.001359453  0.26506366
## blues    0.45019708 0.375181964  1.00000000 -0.014332813  0.36345124
## good     0.02423656 0.001359453 -0.01433281  1.000000000 -0.02110194
```

```
## mind      0.31162741 0.265063660  0.36345124 -0.021101944  1.00000000
```

However, this doesn't really help us: we need to take our first step toward pulling out $\mathbf{\Psi}$ by reducing the dimensionality of $\mathbf{\Lambda}^*$. Instead of using all of the eigenvalues to create $\mathbf{\Lambda}^*$, we will only use the first q. The eigenvalues are ordered by size – in essence, how much of the variability in the data they represent – so the first q are the most important. We won't be losing that much information about the data by representing it with only the top q eigenvalues (this is also the idea behind the dimension-reduction method of principal components analysis).

```
q <- 4
lambda_star <- eigenv$vectors[,1:q] %*% sqrt(diag(eigenv$values[1:q]))
```

Note that $\mathbf{\Lambda}^*$ is now a $p \times q$ matrix, whereas before it was $p \times p$ (this is why we need $q < p$: we can't *add* information to our observed data, only take it away).

Not only are we reducing the dimension of $\mathbf{\Lambda}^*$, we also need to get rid of the part that belongs to $\mathbf{\Psi}$. Recall that $\mathbf{\Psi}$ is a diagonal matrix.

We call the diagonal elements of $\mathbf{\Lambda\Lambda}^T$ the **communalities**. The communality for a given item, h_j^2, represents the total proportion of the variance of each x_j that is explained by the factors (you can think of this as something like the α we estimated for tests for single latent variable, or R^2 in linear regression).

We calculate the communality as the sum of the squared loadings for that item:

$$h_j^2 = \sum_{k=1}^{q} \lambda_{jk}^2$$

The rest of the variance of the item is assumed to be random error: the part contributed by $\mathbf{\Psi}$. We call the diagonal elements of $\mathbf{\Psi}$ the **uniquenesses**. The uniquenesses must be what we get when we subtract the communalities from the diagonal of the correlation matrix, of which each element is 1:

$$u_j^2 = 1 - h_j^2$$

To estimate the communalities, we are going to "recreate" the correlation matrix as the product of our new $\mathbf{\Lambda}^*$ and its transpose:

$$\mathbf{\Sigma}^* = \mathbf{\Lambda}^* \mathbf{\Lambda}^{*T}$$

```
sigma_star <- lambda_star %*% t(lambda_star)
dim(sigma_star)
```

```
## [1] 20 20
```

Like the original correlation matrix $\mathbf{\Sigma}$, $\mathbf{\Sigma}^*$ is still $p \times p$. However, since we're only calculating it from q eigenvalues, we haven't explained all of the variability in the data – we are assuming

the rest, which is not explained by the q factors, is the random error. Therefore, the diagonal elements of this new matrix $\mathbf{\Sigma}^*$ represent the communalities.[7]

We are going to replace the diagonal of the old correlation matrix $\mathbf{\Sigma}$ (which used to be all 1's) with the communalities from the new $\mathbf{\Sigma}^*$ matrix:

```
diag(sigma) <- diag(sigma_star)
```

That way, the difference between the real correlation matrix and the one we created with $\mathbf{\Lambda}^*\mathbf{\Lambda}^{*T}$ will be our estimate of $\mathbf{\Psi}$.

3.2.3 The iterative process

We're not done. We need to iterate through this process until in converges. We'll use `R` to repeat the process (starting from the beginning) until the communalities stay consistent from one iteration to the next.

```
# start off with the original correlation matrix
sigma_star_old <- cor(cesd)
# arbitrary starting values for the communalities
h_new <- 100
h_old <- 0

# this tells R to repeat what's in the brackets until there's a very, very small
# difference in between the communalities from one iteration (h_old) and the next (h_new
while(abs(h_new) - abs(h_old) > 0.00001){

  # first we calculate the communalities from the current sigma_star
  h_old <- sum(diag(sigma_star_old))

  # we do an eigendecomposition
  eigen_old <- eigen(sigma_star_old, symmetric = TRUE)

  # and calculate new values for lambda from the first q eigenvalues
  lambda <- eigen_old$vectors[,1:q] %*% sqrt(diag(eigen_old$values[1:q]))

  # use the new lambda to calculate a new sigma_star
  sigma_star_new <- lambda %*% t(lambda)

  # replace the old diagonal of sigma_star with the new diagonal
  diag(sigma_star_old) <- diag(sigma_star_new)

  # calculate the new communalities
```

[7] Try writing out the matrix multiplication of the loading matrix $\mathbf{\Lambda}^*\mathbf{\Lambda}^{*T}$ to show that each element of the diagonal corresponds to h_j^2.

```
  h_new <- sum(diag(sigma_star_new))
}
```

Now we should have a stable estimate of $\mathbf{\Lambda\Lambda}^T$, saved as `sigma_star_old`. We can confirm that the 1's on the diagonal of the correlation matrix (the total variance of each item) have been replaced with values <1 – the communalities – representing the proportion of the variance explained by the factors. Whatever is left over is our estimate of $\mathbf{\Psi}$!

```
cor(cesd)[1:5, 1:5]
```

```
##            bothered    appetite       blues         good        mind
## bothered 1.00000000 0.298982581  0.45019708  0.024236559  0.31162741
## appetite 0.29898258 1.000000000  0.37518196  0.001359453  0.26506366
## blues    0.45019708 0.375181964  1.00000000 -0.014332813  0.36345124
## good     0.02423656 0.001359453 -0.01433281  1.000000000 -0.02110194
## mind     0.31162741 0.265063660  0.36345124 -0.021101944  1.00000000
```

```
sigma_star_old[1:5, 1:5]
```

```
##            bothered    appetite       blues         good        mind
## bothered 0.39037380 0.298982581  0.45019708  0.024236559  0.31162741
## appetite 0.29898258 0.297631150  0.37518196  0.001359453  0.26506366
## blues    0.45019708 0.375181964  0.60543878 -0.014332813  0.36345124
## good     0.02423656 0.001359453 -0.01433281  0.546105077 -0.02110194
## mind     0.31162741 0.265063660  0.36345124 -0.021101944  0.42342831
```

However, we don't really care about $\mathbf{\Psi}$. We can look at the values of `lambda`, which is the $p \times q$ matrix of the factor loadings:

```
dim(lambda)
```

```
## [1] 20  4
```

```
lambda[1:5,]
```

```
##             [,1]        [,2]        [,3]        [,4]
## [1,] -0.6077886  0.06160526 -0.09144989  0.09385394
## [2,] -0.4867159  0.12083252 -0.21321902  0.02599854
## [3,] -0.7157665  0.09008332 -0.11555101  0.26767526
## [4,] -0.1212748 -0.68295334  0.11270656 -0.22862513
## [5,] -0.5932657  0.15540030 -0.10026868 -0.19303116
```

These aren't very interpretable as they are (we'll use built-in `R` functions below to fit rotated models, which are easier to interpret), but each value λ_{jk} is the "amount" that a factor k contributes to an item j.

3.3 What just happened?

To review, we just used the following algorithm to estimate the factor loadings:

Using m to index iterations of this algorithm, we:

1. Begin with the eigendecomposition of the original correlation matrix $\mathbf{\Sigma}$, which we will designate $\mathbf{\Sigma}_{(0)}$.
2. Obtain current estimates of the factor loadings, $\mathbf{\Lambda}_{(m)}$ from the first q eigenvectors and eigenvalues.
3. Use the current estimates of the factor loadings to predict the correlation matrix

$$\mathbf{\Sigma}^*_{(m)} = \mathbf{\Lambda}_{(m)}\mathbf{\Lambda}^T_{(m)}$$

4. Replace the diagonal of $\mathbf{\Sigma}_{(0)}$ with the diagonal of $\mathbf{\Sigma}^*_{(m)}$ to get a new $\mathbf{\Sigma}_{(m)}$.
5. Perform the eigendecomposition of $\mathbf{\Sigma}_{(m)}$.
6. Repeat steps 2-5 until the change in the sum of the communalities $\left[\text{trace}(\mathbf{\Sigma}^*_{(m)}) - \text{trace}(\mathbf{\Sigma}^*_{(m-1)})\right]$ is negligible.[8]

4 Model fitting: other approaches

There are other approaches to fitting factor analytic models – we don't always need to do it by hand (and we probably don't want to)! These approaches also make it easy to calculate the best rotation for the factor loadings (see below).

4.1 Maximum likelihood factor analysis

The maximum likelihood solution finds those communality values that minimize the chi square goodness of fit test. The `fa()` function in the `psych` package can perform maximum likelihood factor analysis if we request `fm = "ml"`. Note that there is also a function called `factanal` included in the `stats` package in base `R` that fits maximum likelihood factor analysis.

The code below shows three different types of rotation using the maximum likelihood approach. (What we did by hand corresponds to `rotate = "none"`.) We'll discuss what rotation does in a bit.

```
# FA by maximum likelihood, no rotation (factors are orthogonal)
cesd.ml.none <- fa(cor(cesd), nfactors = q, n.obs = nrow(cesd),
                   fm = "ml", rotate = "none")
```

[8] The trace of a square matrix is the sum of its diagonal elements.

```
# FA by maximum likelihood, varimax rotation (factors are orthogonal)
cesd.ml.varimax <- fa(cor(cesd), nfactors = q, n.obs = nrow(cesd),
                      fm = "ml", rotate = "varimax")

# FA by maximum likelihood, oblimin rotation (factors are oblique)
cesd.ml.oblimin <- fa(cor(cesd), nfactors = q, n.obs = nrow(cesd),
                      fm = "ml", rotate = "oblimin")
```

4.2 Factor analysis with the `minres` solution

The minimum residual (`minres`) solution is an unweighted least squares solution that uses the `optim` function and adjusts the diagonal elements of the correlation matrix to minimize the squared residual when the factor model is the eigenvalue decomposition of the reduced matrix. The `minres` solution is similar to principal axis factor analysis and both methods will often work when maximum likelihood does not.

The code below shows the three main types of rotation with this solution.

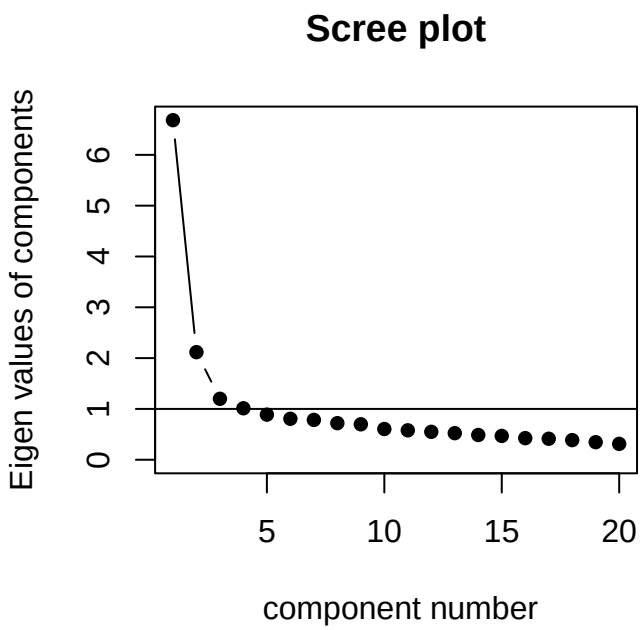
```
# FA by minres, no rotation (factors are orthogonal)
cesd.minres.none <- fa(cor(cesd), nfactors = q, fm = "minres", rotate = "none")

# FA by minres, varimax rotation (factors are orthogonal)
cesd.minres.varimax <- fa(cor(cesd), nfactors = q, fm = "minres", rotate = "varimax")

# FA by minres, varimax rotation (factors are orthogonal)
cesd.minres.oblimin <- fa(cor(cesd),nfactors = q, fm = "minres", rotate = "oblimin")
```

5 Modeling choices

5.1 Choosing the number of factors

One way to get a sense of how many factors to use is to look at the scree plot, the plot of the eigenvalues.

```r
scree(cesd, factors = F)
```



A somewhat old-fashioned way to determine the number of factors is to use the Kaiser rule, which drops all components with eigenvalues under 1.0. The idea is that we don't want factors that have less information than we would expect the average single item to have, because we are trying to reduce our data.[9]). The Kaiser criterion is the default in SPSS and most statistical software but is not recommended when used as the sole cut-off criterion for estimating the number of factors, as it tends to over-extract factors.

Another criterion is to look for the "elbow" in the scree plot and to exclude all components after the component where the "elbow" starts.

More modern criteria, include Horn's Parallel Analysis (1965) and Velicer's Minimum Average Partial (MAP) criterion, are implemented in the `psych` package.

```r
# Horn's Parallel analysis suggests 7 factors
fa.parallel(cor(cesd), n.obs = nrow(cesd), fm = "ml", fa = "fa")

# Velicer's MAP criterion suggests 2 factors
vss(cor(cesd), n.obs = nrow(cesd), fm = "ml")
```

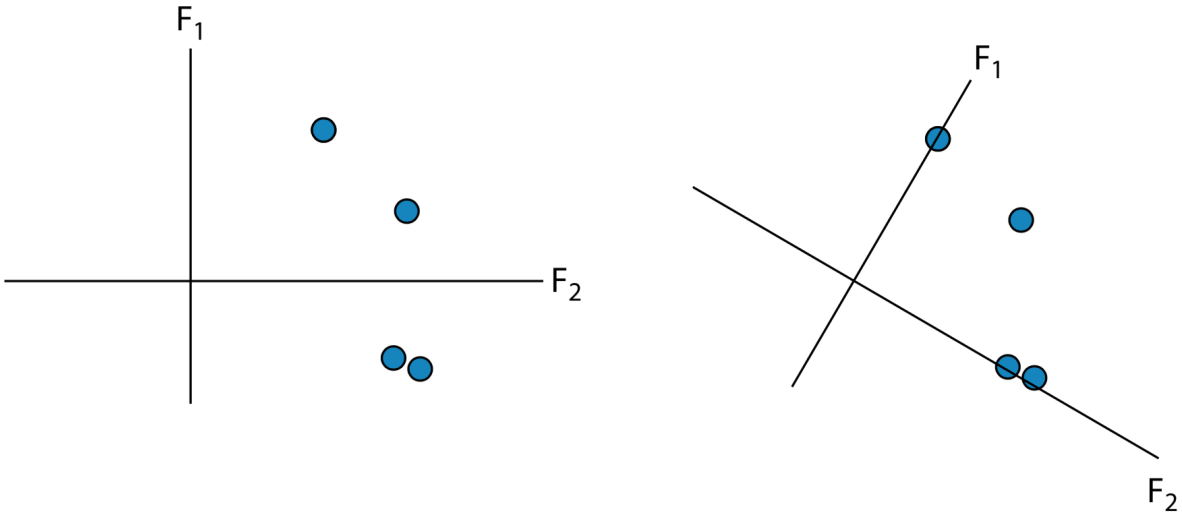
Choosing the number of factors is somewhat of an art ....

[9]The sum of the eigenvalues of a matrix is equal to the trace of the matrix. We started with the correlation matrix, which has 1 as each of its diagonal elements, so its trace is p, the number of items. Therefore the eigenvalues will sum to p.

5.2 Rotation

5.2.1 Orthogonal rotation

We noted earlier that the factor loadings were not particularly interpretable. When fitting the model as we did above, we have forced the first factor to explain most of the variability in the items, then the second factor to explain as much as possible of what's left over, and so on. However, that's not really what we expect the factors to do. We still want them to explain as much of the variability in the items as possible, we just don't feel the need to insist that the first factor be so ambitious in its explanatory power. Instead, we can "rotate" the loading matrix so that the items are as closely aligned to a single factor as possible. When we do an *orthogonal* rotation, we maintain the lack of correlation between the factors that we assumed in the model above. We're still explaining the same amount of variation, it's just spread across the factors in a way that minimizes the distance between the factors and the items. This means that the loadings for a given item will be as close to 0 for as many of the factors as possible (and hopefully large for just a single factor).



	Factor 1	Factor 2		Factor 1	Factor 2
x_1	0.4	0.69	x_1	-0.8	0
x_2	0.4	0.69	x_2	-0.8	0
x_3	0.65	0.32	x_3	-0.6	0.4
x_4	0.69	-0.4	x_4	0	0.8
x_5	0.61	-0.35	x_5	0	0.7

Let's look at the factor loadings before rotation. We'll (somewhat arbitrarily) only print off the loadings above 0.35 to make it easier to read:

```
# before rotation
print(cesd.ml.none$loadings, cut = .35)
```

```
##
## Loadings:
##             ML1    ML2    ML3    ML4
## bothered     0.571
## appetite     0.438
## blues        0.691
## good                0.493
## mind         0.548
## depressed    0.776
## effort       0.561
## hopeful             0.583
## failure      0.645
## fearful      0.632
## sleep        0.512
## happy        0.486  0.635
## talk         0.422
## lonely       0.659
## unfriendly   0.364
## enjoy        0.417  0.653
## cry          0.622
## sad          0.764
## dislike      0.504         0.428
## getgoing     0.606
##
##                  ML1   ML2   ML3   ML4
## SS loadings    6.148 1.614 0.645 0.510
## Proportion Var 0.307 0.081 0.032 0.026
## Cumulative Var 0.307 0.388 0.420 0.446
```

Note that most items are loading only on the first factor – we've forced them to do so. Note also the `SS loadings` row: these are the eigenvalues from the $\mathbf{\Lambda}\mathbf{\Lambda}^T$ matrix. These are clearly ordered so that the eigenvalue corresponding to the first factor explains by far the most variation in the data (`Proportion Var` row).

Now we'll look at the factor loadings from a solution after an orthogonal rotation:

```
# orthogonal rotation
print(cesd.ml.varimax$loadings, cut = .35)
```

```
##
## Loadings:
##             ML1    ML2    ML4    ML3
## bothered     0.483
```

```
## appetite    0.382
## blues       0.672
## good               0.518
## mind        0.380         0.383
## depressed   0.712
## effort                    0.583
## hopeful            0.653
## failure     0.465                 0.455
## fearful     0.506
## sleep                     0.451
## happy              0.751
## talk        0.364
## lonely      0.608
## unfriendly                        0.495
## enjoy              0.749
## cry         0.604
## sad         0.723
## dislike                           0.676
## getgoing                  0.709
##
##                   ML1   ML2   ML4   ML3
## SS loadings     3.839 1.955 1.774 1.349
## Proportion Var  0.192 0.098 0.089 0.067
## Cumulative Var  0.192 0.290 0.378 0.446
```

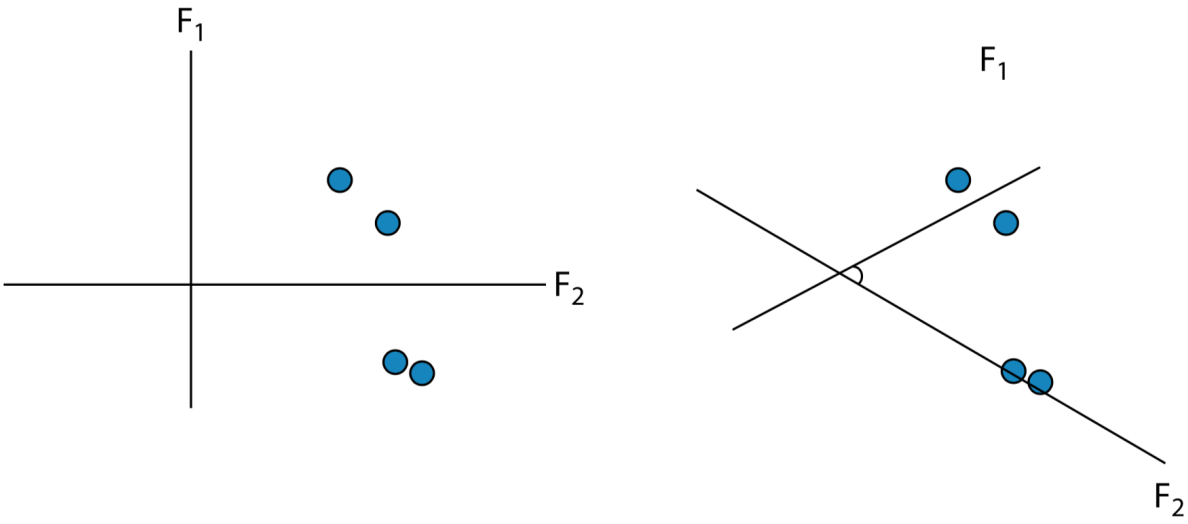
Now we see that the number of items corresponding to each factor is more evenly distributed across the factors. This is also reflected in the fact that each explains a somewhat more equal proportion of the variance.

Now we can start to interpret them. Factor 1 looks like it corresponds to what was described in the original paper as "depressed affect", factor 2 to "positive affect", factor 3 to "somatic and retarded activity", and factor 4 to "interpersonal".[10]

5.2.2 Oblique rotation

We could instead assume that the factors are correlated in some way. This may be more realistic – for example, in the CES-D, we may expect that "positive affect" and "depressed affect" are correlated. However, it also means the covariance matrix for the factors, $\text{Cov}(\mathbf{F}_i)$, doesn't drop out like it did when we worked through the model above.

[10]However, Radloff (1977) noted the high internal consistency of the scale as a whole, and recommended against "undue emphasis on separate factors" for the use of the total score "as an estimate of the degree of depressive symptomatology".



That's ok, though! We just have another matrix to estimate: the correlation matrix for the factors. Let's look at the solution with an oblique rotation:

```
# oblique rotation
print(cesd.ml.oblimin$loadings, cut = .35)
```

```
##
## Loadings:
##             ML1    ML2    ML4    ML3
## bothered    0.440
## appetite    0.353
## blues       0.720
## good               0.548
## mind
## depressed   0.714
## effort                    0.630
## hopeful            0.673
## failure     0.366                0.384
## fearful     0.450
## sleep                     0.452
## happy              0.759
## talk
## lonely      0.637
## unfriendly                       0.507
## enjoy              0.766
## cry         0.661
```

```
## sad         0.764
## dislike                          0.693
## getgoing                 0.797
##
##                  ML1   ML2   ML4   ML3
## SS loadings    3.40 1.937 1.460 1.012
## Proportion Var 0.17 0.097 0.073 0.051
## Cumulative Var 0.17 0.267 0.340 0.390
```

These item-factor pairs correspond almost exactly to what we saw on the lecture slides.

We can also look at the estimated correlation matrix for the factors:

```
cesd.ml.oblimin$Phi
```

```
##            ML1       ML2       ML4       ML3
## ML1 1.0000000 0.3392740 0.6731728 0.4949523
## ML2 0.3392740 1.0000000 0.2259520 0.2106451
## ML4 0.6731728 0.2259520 1.0000000 0.3851314
## ML3 0.4949523 0.2106451 0.3851314 1.0000000
```

It looks like the factors corresponding to "depressed affect" and "somatic and retarded activity" are highly correlated, while "positive affect" is not very correlated with either "somatic and retarded activity" or "interpersonal".

6 Cronbach's alpha: internal consistency reliability

Based on Radloff's (1977) recommendation, let's proceed as though we can treat the total CES-D score as a valid measurement of a single latent construct (depressive symptomatology). We asked you to compute Cronbach's alpha as a measure of internal consistency reliability.

Recall that we can calculate Cronbach's alpha from the covariance matrix by hand. If Y is a scale consisting of k items, $X_1, X_2, \ldots, X_k$, then we calculate

$$\alpha = \frac{k}{k-1}\left[1 - \frac{\sum_{i=1}^{k}\sigma_i^2}{\sigma_y^2}\right]$$

where σ_i^2 are the item variances and σ_y^2 is the total variance of the scale (which we can obtain by summing all the elements of the covariance matrix).

Instead of calculating this by hand for the measurement project, however, we can use the `alpha()` function in the `psych` package to compute Cronbach's alpha for us. The `alpha()` function produces a lot of output, so first we'll pull out the Cronbach's alpha statistic that we are interested in.

```
# note that we have to invoke the alpha function using psych::alpha
# because R has other functions called alpha
psych::alpha(cesd)$total[1]
```

```
##  raw_alpha
##  0.8679446
```

- How do you interpret this Cronbach's alpha?

A few helpful notes about the behavior of the `alpha()` function:

- `alpha()` requires non-missing data, so if you should exclude respondents with missing data (or use multiple imputation to fill in the missing data)
- You will generally want to recode reverse-coded items so that they are going the same direction as the other items. You can check the correlation matrix for negative correlations to see whether any items are behaving this way.
- `alpha()` does have an option `check.keys=TRUE` which will "find the first principal component and reverse key items with negative loadings.'' It will give a warning of this happens.
- While Cronbach's alpha is the most widely reported measure of internal consistency reliability (probably because it is easy to calculate and is implemented in most software packages) it has been noted that it "underestimates the reliability of a test and over estimates the first factor saturation.'' There are other measure of reliability that have better properties for different situations (see Revelle Chapter 7 for more information).[11]

7 Raw and corrected item correlations

The `alpha()` function actually produces a lot of useful output that we can examine to see how the individual items are performing.

The raw and corrected item correlations appear under `Item Statistics` in the output. These are correlations of the individual items with the full scale. The raw item correlations (`raw.r` in the output below) are correlations between individual items and the full scale **including** the individual item, i.e. not correcting for item overlap. The corrected item correlations (`r.drop`) are the correlations of the item with the scale composed of the remaining items.

Also take a look at the output under `Reliability if an item is dropped`. The `raw_alpha` column shows Cronbach's alpha computed after dropping the item.

```
psych::alpha(cesd)
```

```
##
## Reliability analysis
## Call: psych::alpha(x = cesd)
##
##   raw_alpha std.alpha G6(smc) average_r S/N    ase mean   sd median_r
##       0.87      0.89     0.9      0.28 7.7 0.0051 0.46 0.43     0.28
##
```

[11] Revelle W, An introduction to psychometric theory with applications in R (Under development). Chapter 7: Reliability. Available at: https://www.personality-project.org/r/book/

```
##     95% confidence boundaries
##          lower alpha upper
## Feldt     0.86  0.87  0.88
## Duhachek  0.86  0.87  0.88
##
##  Reliability if an item is dropped:
##            raw_alpha std.alpha G6(smc) average_r S/N alpha se var.r med.r
## bothered        0.86      0.88    0.90      0.28 7.3   0.0054 0.021  0.28
## appetite        0.86      0.88    0.90      0.28 7.5   0.0053 0.021  0.29
## blues           0.86      0.88    0.89      0.27 7.0   0.0055 0.020  0.28
## good            0.88      0.89    0.90      0.30 8.2   0.0046 0.016  0.30
## mind            0.86      0.88    0.90      0.28 7.3   0.0054 0.021  0.28
## depressed       0.85      0.87    0.89      0.27 6.9   0.0056 0.018  0.28
## effort          0.86      0.88    0.90      0.28 7.3   0.0054 0.021  0.28
## hopeful         0.87      0.89    0.90      0.29 7.7   0.0050 0.019  0.30
## failure         0.86      0.88    0.89      0.27 7.1   0.0054 0.021  0.28
## fearful         0.86      0.88    0.90      0.27 7.1   0.0054 0.020  0.28
## sleep           0.86      0.88    0.90      0.28 7.4   0.0053 0.021  0.28
## happy           0.86      0.88    0.90      0.28 7.4   0.0054 0.021  0.29
## talk            0.86      0.88    0.90      0.28 7.5   0.0052 0.021  0.29
## lonely          0.86      0.88    0.89      0.27 7.1   0.0055 0.020  0.28
## unfriendly      0.87      0.88    0.90      0.29 7.7   0.0052 0.021  0.30
## enjoy           0.86      0.88    0.90      0.28 7.6   0.0053 0.020  0.30
## cry             0.86      0.88    0.90      0.27 7.2   0.0054 0.021  0.28
## sad             0.86      0.87    0.89      0.27 6.9   0.0056 0.019  0.27
## dislike         0.86      0.88    0.90      0.28 7.4   0.0053 0.021  0.28
## getgoing        0.86      0.88    0.90      0.28 7.2   0.0054 0.020  0.28
##
##  Item statistics
##                n raw.r std.r r.cor r.drop mean   sd
## bothered    1427  0.58  0.59  0.56   0.52 0.40 0.70
## appetite    1427  0.47  0.49  0.44   0.40 0.30 0.69
## blues       1427  0.66  0.68  0.67   0.61 0.28 0.67
## good        1427  0.31  0.23  0.17   0.17 0.74 1.19
## mind        1427  0.56  0.58  0.54   0.49 0.48 0.78
## depressed   1427  0.73  0.75  0.75   0.68 0.53 0.77
## effort      1427  0.58  0.58  0.55   0.50 0.59 0.89
## hopeful     1427  0.47  0.39  0.36   0.35 0.85 1.15
## failure     1427  0.63  0.67  0.65   0.59 0.20 0.55
## fearful     1427  0.61  0.65  0.63   0.56 0.31 0.65
## sleep       1427  0.55  0.55  0.51   0.47 0.74 0.93
## happy       1427  0.58  0.52  0.51   0.50 0.63 0.95
## talk        1427  0.46  0.48  0.43   0.39 0.49 0.80
## lonely      1427  0.64  0.66  0.64   0.58 0.46 0.80
## unfriendly 1427  0.39  0.43  0.38   0.34 0.18 0.54
```

```
## enjoy      1427  0.53  0.46  0.44   0.44 0.59 0.98
## cry        1427  0.60  0.63  0.61   0.55 0.24 0.59
## sad        1427  0.71  0.74  0.74   0.67 0.48 0.71
## dislike    1427  0.48  0.53  0.50   0.44 0.15 0.47
## getgoing   1427  0.60  0.60  0.58   0.53 0.60 0.81
##
## Non missing response frequency for each item
##                0    1    2    3 miss
## bothered   0.71 0.21 0.06 0.02    0
## appetite   0.81 0.12 0.05 0.03    0
## blues      0.82 0.11 0.04 0.03    0
## good       0.69 0.07 0.06 0.18    0
## mind       0.67 0.22 0.08 0.03    0
## depressed  0.61 0.28 0.08 0.03    0
## effort     0.62 0.24 0.07 0.07    0
## hopeful    0.59 0.15 0.11 0.16    0
## failure    0.86 0.10 0.04 0.01    0
## fearful    0.77 0.16 0.04 0.02    0
## sleep      0.53 0.28 0.12 0.07    0
## happy      0.62 0.21 0.09 0.08    0
## talk       0.68 0.19 0.10 0.03    0
## lonely     0.70 0.19 0.08 0.04    0
## unfriendly 0.87 0.09 0.02 0.02    0
## enjoy      0.68 0.14 0.08 0.09    0
## cry        0.82 0.12 0.04 0.01    0
## sad        0.63 0.28 0.07 0.02    0
## dislike    0.89 0.08 0.02 0.01    0
## getgoing   0.57 0.30 0.10 0.04    0
```

- What happens to Cronbach's alpha when leaving out an item that has a low item correlation?
- How would you use the raw and corrected item correlations to help you decide what items to include in your final scale?