Factor Analysis

Max Turgeon

STAT 4690–Applied Multivariate Analysis

Latent variable models

- With PCA, we saw how we could reduce the dimension of data using the eigenvectors of the sample covariance matrix.
- Conversely, we could construe PCA has a generative model, where the principal components give rise to the observed data.
- **Latent Variable Models** formalise this idea:
  - Latent (i.e. unobserved) variables $\mathbf{F}$ give rise to observed data $\mathbf{Y}$ through a *specified* model.

Factor Analysis i

- **Factor Analysis** is a special kind of latent variable model.
- Let $\mathbf{Y}$ be a p-dimensional vector with mean μ and covariance matrix Σ.
- Let $\mathbf{F}$ be a m-dimensional *latent* vector.
- The *orthogonal factor model* is given by

$$\mathbf{Y} - \mu = L\mathbf{F} + \mathbf{E},$$

where L is a $p \times m$ *matrix of factor loadings*, and $\mathbf{E}$ is a p-dimensional vector of *errors*.

Factor Analysis ii

- $\mathbf{F}$ are also called *common factors*; $\mathbf{E}$ are also called *specific factors*.
- **Note**: This is essentially a multivariate regression model, but where the covariates are unobserved.

Assumptions i

- The model above is generally not identifiable, since there are too many parameters.
- We therefore need to impose further restrictions:
  - $E(\mathbf{F}) = 0$
  - $\mathrm{Cov}(\mathbf{F}) = I$
  - $E(\mathbf{E}) = 0$
  - $\mathrm{Cov}(\mathbf{E}) = \Psi = \begin{pmatrix} \psi_1 & 0 & \cdots & 0 \\ 0 & \psi_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \psi_p \end{pmatrix}$
  - $\mathrm{Cov}(\mathbf{F}, \mathbf{E}) = 0$

Assumptions ii

- In other words:
- Both common and specific factors have mean zero;
- They are uncorrelated;
- The common factors are mutually uncorrelated and standardised;
- The specific factors each affect only one observed variable.

Structured Covariance i

- As a consequence of these assumptions, we can derive an assumption on the structure of $\Sigma = \mathrm{Cov}(\mathbf{Y})$:

$$
\begin{aligned}
\Sigma &= E\left((\mathbf{Y}-\mu)(\mathbf{Y}-\mu)^T\right) \\
&= E\left((L\mathbf{F}+\mathbf{E})(L\mathbf{F}+\mathbf{E})^T\right) \\
&= LE(\mathbf{F}\mathbf{F}^T)L + E(\mathbf{E}\mathbf{F}^T)L^T + LE(\mathbf{F}\mathbf{E}^T) + E(\mathbf{E}\mathbf{E}^T) \\
&= LIL^T + 0L^T + L0 + \Psi \\
&= LL^T + \Psi.
\end{aligned}
$$

Structured Covariance ii

- Similarly, we can show that

$$\mathrm{Cov}(\mathbf{Y}, \mathbf{F}) = L.$$

- If we write ℓ_{ij} for the (i, j)-th element of L, we see that

$$\mathrm{Var}(Y_i) = \sum_{k=1}^{m} \ell_{ik}^2 + \psi_i.$$

- Crucially, these assumptions are **testable**. In other words, we can check whether they are reasonable for our data.

Example i

- Let's look at an example where there is no solution.
- Assume $p = 3$, $m = 1$, with

$$\Sigma = \begin{pmatrix} 1 & 0.9 & 0.7 \\ 0.9 & 1 & 0.4 \\ 0.7 & 0.4 & 1 \end{pmatrix}.$$

- From our assumptions on the covariance structure, we derive a system of equations

$$\begin{aligned} 1 = \ell_{11}^2 + \psi_1 \quad & 0.9 = \ell_{11}\ell_{21} \quad & 0.7 = \ell_{11}\ell_{31} \\ 1 = \ell_{22}^2 + \psi_2 \quad & 0.4 = \ell_{21}\ell_{31} \\ & 1 = \ell_{33}^2 + \psi_3 \end{aligned}$$

Example ii

- From $0.7 = \ell_{11}\ell_{31}$ and $0.4 = \ell_{21}\ell_{31}$, we get

$$\ell_{21} = \frac{0.4}{0.7}\ell_{11}.$$

- But since $0.9 = \ell_{11}\ell_{21}$, we can conclude that

$$\ell_{11} = \pm 1.255.$$

- However, since the first component Y_1 has unit variance, $\ell_{11} = \mathrm{Corr}(Y_1, F_1)$, and therefore the correlation is out of bounds.
- Similarly, we get

$$\psi_1 = 1 - \ell_{11}^2 = 1 - 1.575 = -0.575.$$

Example iii

- But since ψ_1 is the variance of the first error term, we once again get a non-sensical solution.

Factor Rotation i

- Even with our assumptions above, our model is still not uniquely identified.
- Let T be an $m \times m$ orthogonal matrix. We have

$$\begin{aligned}\mathbf{Y} - \mu &= L\mathbf{F} + \mathbf{E} \\ &= LTT^T\mathbf{F} + \mathbf{E} \\ &= \tilde{L}\tilde{\mathbf{F}} + \mathbf{E},\end{aligned}$$

where $\tilde{L} = LT$ and $\tilde{\mathbf{F}} = T^T\mathbf{F}$.

Factor Rotation ii

- Both models lead to the same covariance matrix:

$$\Sigma = LL^T + \Psi = LTT^T L^T + \Psi = \tilde{L}\tilde{L}^T + \Psi.$$

- As we will see, this will turn out to be a blessing in disguise:
  - We will impose a uniqueness condition to get one solution.
  - Then we will rotate our solution using T to improve interpretation.

Estimation–Principal Component Method i

- Recall the spectral decomposition of the covariance matrix:

$$\Sigma = \sum_{i=1}^{p} \lambda_i w_i w_i^T,$$

with $\lambda_1 \geq \cdots \geq \lambda_p$.

- If we let W be the matrix whose i-th column is $\sqrt{\lambda_i} w_i$, we can rewrite the spectral decomposition as

$$\Sigma = WW^T.$$

- In other words, if we let $m = p$ and $\Psi = 0$, we see that we recover the orthogonal factor model with $L = W$.

Estimation–Principal Component Method ii

- Of course, this is not very satisfactory, as the dimension of the common factors is the same as that of the original data.
- Instead, we select $m < p$ using one of the methods we discussed with PCA and we approximate

$$\Sigma \approx \sum_{i=1}^{m} \lambda_i w_i w_i^T.$$

- If we let L be the $p \times m$ matrix whose i-th column is $\sqrt{\lambda_i} w_i$, we can estimate Ψ as follows:

$$\psi_i = \sigma_{ii} - \sum_{j=1}^{m} \ell_{ij}^2.$$

Estimation–Principal Component Method iii

Algorithm

1. Let $\hat{\lambda}_1 \cdots > \hat{\lambda}_p$ and $\hat{w}_1, \ldots, \hat{w}_p$ be the eigenvalues and eigenvectors of the covariance matrix S_n.
2. Select m using one of the PCA criteria.
3. Estimate $\hat{L}$ with the $p \times m$ matrix whose i-th column is $\sqrt{\hat{\lambda}_i}\hat{w}_i$.
4. Estimate $\hat{\Psi}$ with the diagonal elements of $S_n - \hat{L}\hat{L}^T$.

Example i

- We are going to use the `bfi` dataset in the `psych` package.
- Contains data on 25 personality items grouped in 5 categories:
  - A (Agreeableness)
  - C (Conscientiousness)
  - E (Extraversion)
  - N (Neuroticism)
  - O (Openness)

Example ii

```
library(psych)
```

```
dim(bfi)
```

```
## [1] 2800   28
```

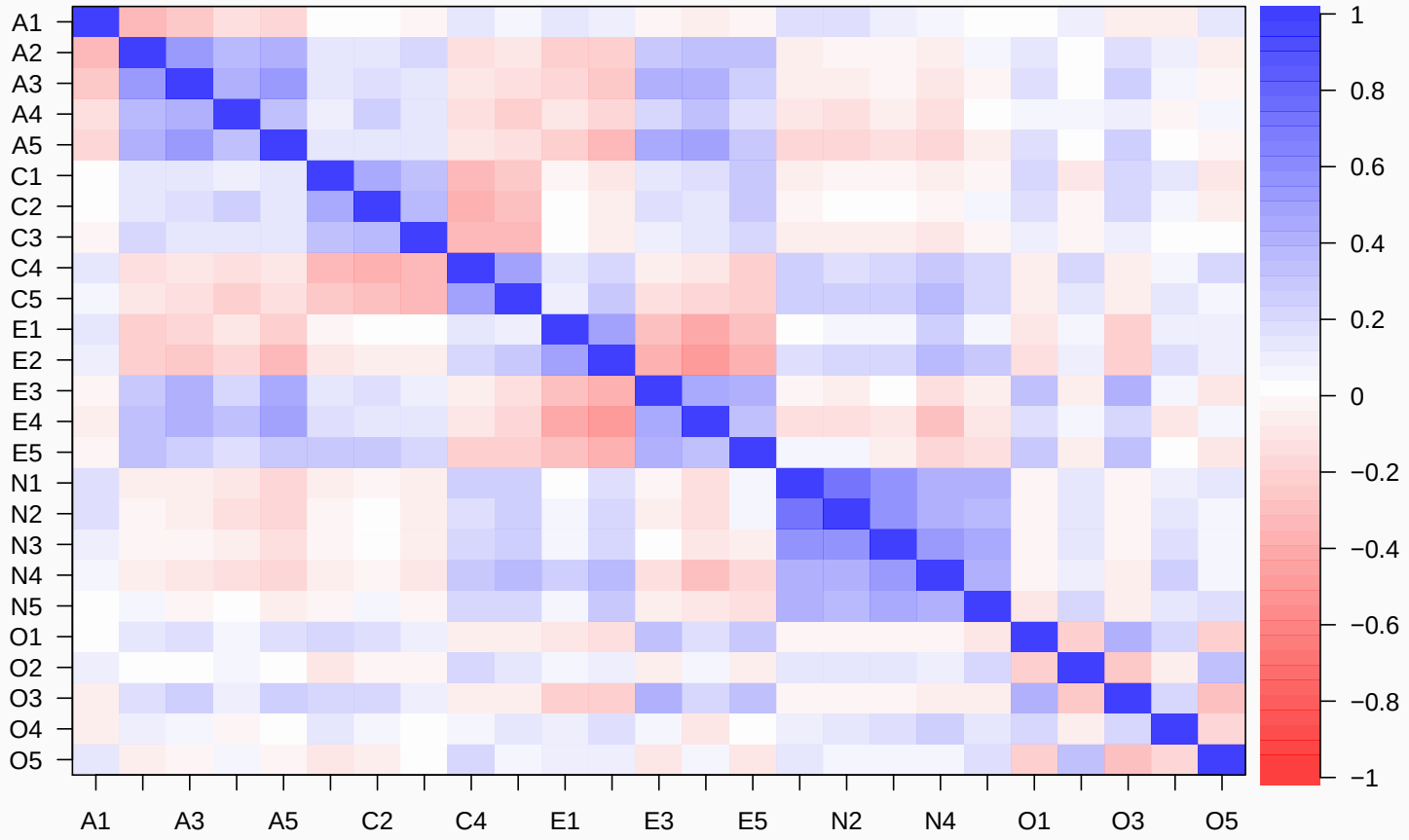
```
tail(names(bfi), n = 3)
```

```
## [1] "gender"    "education" "age"
```

Example iii

```
library(tidyverse)

# Remove demographic variable and keep complete data
data <- bfi %>%
  select(-gender, -education, -age) %>%
  filter(complete.cases(.))

cor.plot(cor(data))
```

Example iv


A1
A2
A3
A4
A5
C1
C2
C3
C4
C5
E1
E2
E3
E4
E5
N1
N2
N3
N4
N5
O1
O2
O3
O4
O5
A1
A3
A5
C2
C4
E1
E3
E5
N2
N4
O1
O3
O5
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1

Example v

```
decomp <- prcomp(data)
summary(decomp)$importance[,1:3]
```

```
##                             PC1      PC2      PC3
## Standard deviation     3.291635 2.451538 2.030393
## Proportion of Variance 0.215650 0.119620 0.082050
## Cumulative Proportion  0.215650 0.335270 0.417320
```

Example vi

```
cum_prop <- decomp %>%
  summary %>%
  .[["importance"]] %>%
  .["Cumulative Proportion",]

which(cum_prop > 0.8)[1]
```

```
## PC14
##   14
```

Example vii

```
Lhat <- decomp$rotation[,1:14] %*%
  diag(decomp$sdev[1:14])
Psi_hat <- diag(cov(data) - tcrossprod(Lhat))

# Sum squared error
sum((cov(data) - tcrossprod(Lhat) - diag(Psi_hat))^2)

## [1] 3.645694

# Compare to the total variance
sum(diag(cov(data)))
```

Example viii

```
## [1] 50.24287
```

```
# Our FA model explains:
sum(colSums(Lhat^2)/sum(diag(cov(data))))
```

```
## [1] 0.8160488
```

Comments

- In our example above, we saw that 14 factors explained 82% of the total variance.
- Two common default values for m in statistical softwares:
  - Number of positive eigenvalues of the sample covariance matrix.
  - Number of eigenvalues greater than one for the sample correlation matrix.
- In our example, the first criterion would lead to $m = p$, which is not very helpful

Example (cont'd) i

```
(m <- sum(eigen(cor(data))$values > 1))
```

```
## [1] 6
```

```
Lhat <- decomp$rotation[,seq_len(m)] %*%
  diag(decomp$sdev[seq_len(m)])
Psi_hat <- diag(cov(data) - tcrossprod(Lhat))

# Sum squared error
sum((cov(data) - tcrossprod(Lhat) - diag(Psi_hat))^2)
```

Example (cont'd) ii

```
## [1] 6.999035
```

```
# Compare to the total variance
sum(diag(cov(data)))
```

```
## [1] 50.24287
```

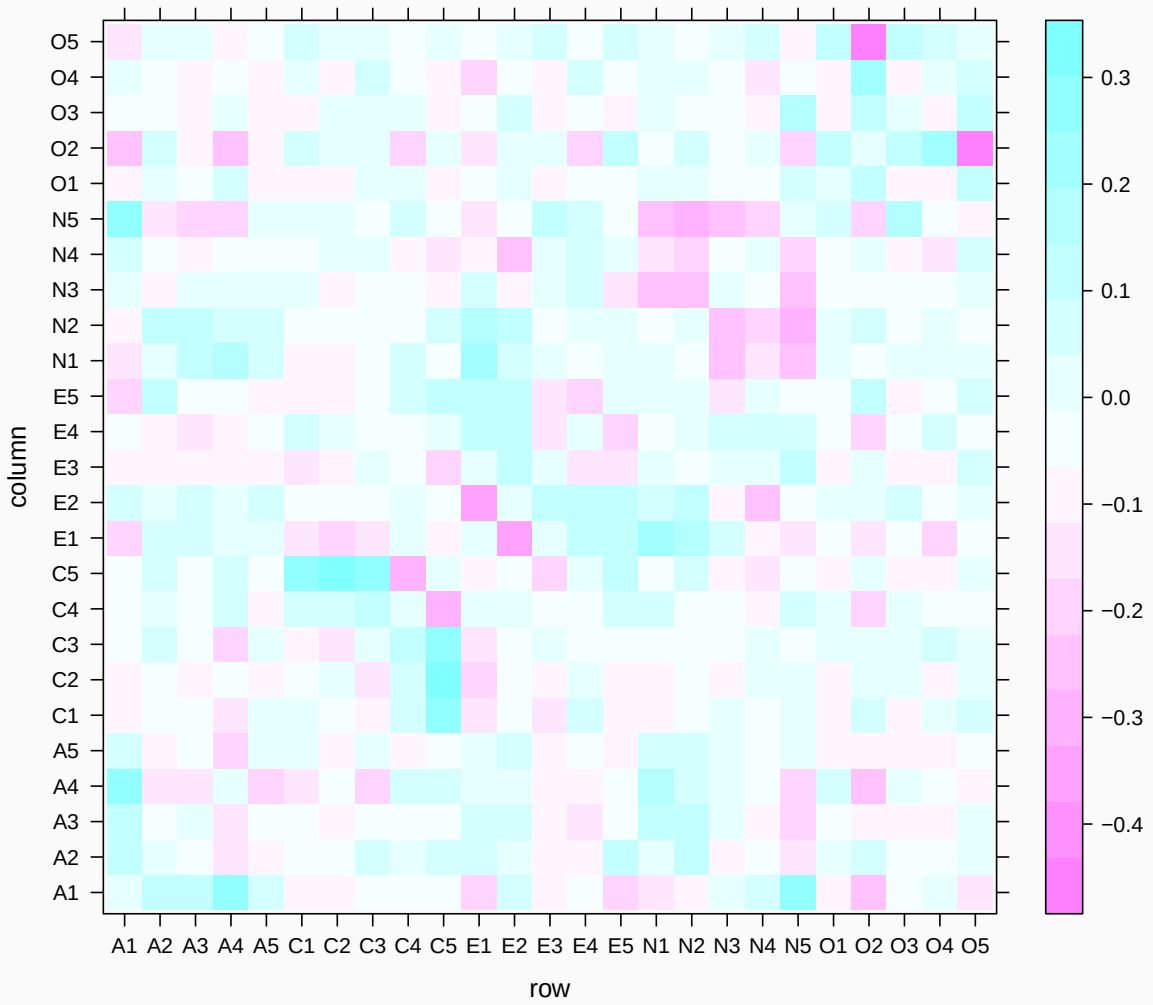
```
# Our FA model explains:
sum(colSums(Lhat^2)/sum(diag(cov(data))))
```

```
## [1] 0.5910586
```

Example (cont'd) iii

```
# We can also  visualize the fit
Sn <- cov(data)
Sn_fit <- tcrossprod(Lhat) + diag(Psi_hat)

library(lattice)
levelplot(Sn - Sn_fit)
```

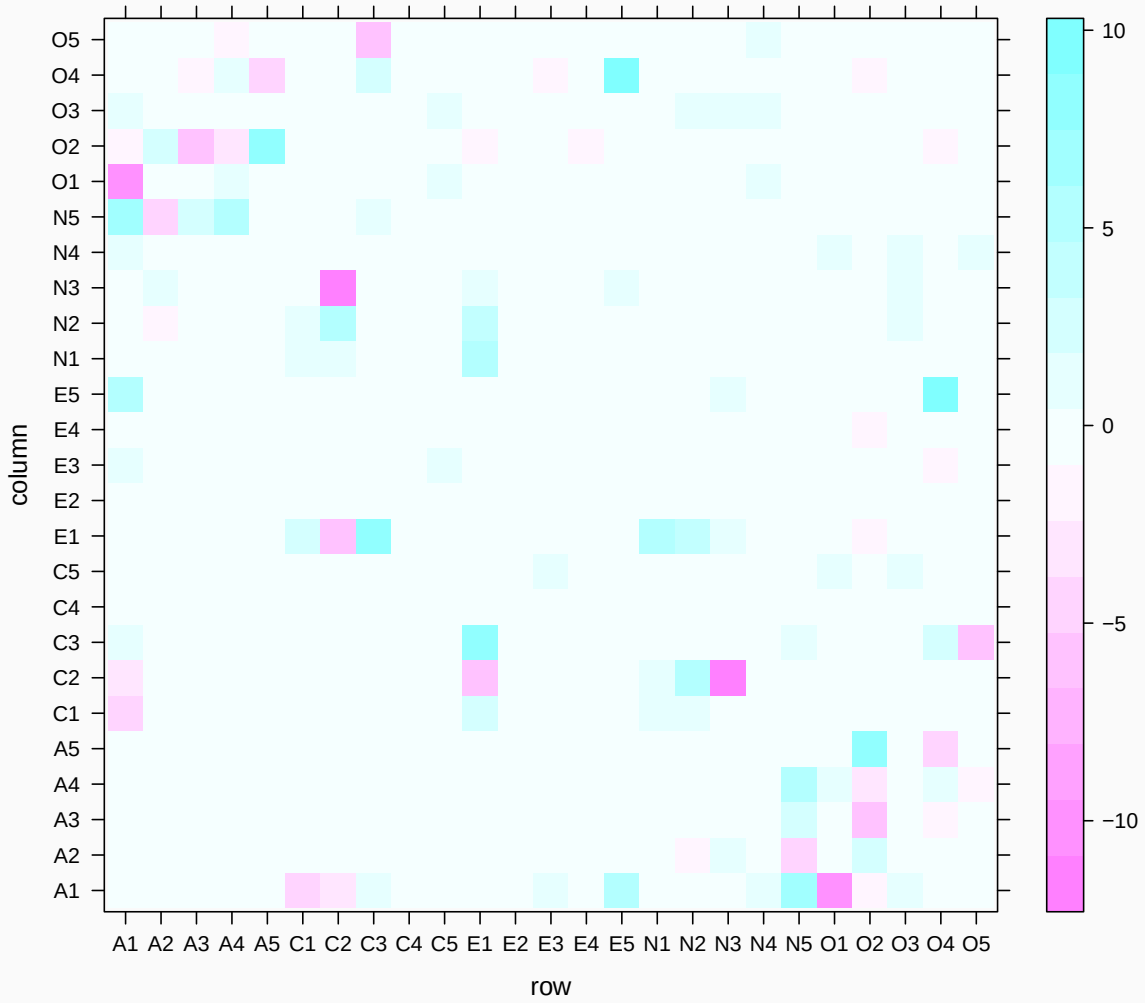
Example (cont'd) iv



Example (cont'd) v

```
# Or if you prefer % difference
levelplot((Sn - Sn_fit)/Sn)
```

Example (cont'd) vi



Estimation–Maximum Likelihood Method i

- This estimation method assumes that both common factors $\mathbf{F}$ and specific factors $\mathbf{E}$ follow a multivariate normal distribution
  - $\mathbf{F} \sim N_m(0, I)$
  - $\mathbf{E} \sim N_p(0, \Psi)$
- From this assumption, it follows that $\mathbf{Y}$ also follows a multivariate normal distribution
  - $\mathbf{Y} \sim N_p(\mu, LL^T + \Psi)$
- Therefore, we can write down the likelihood in terms of both L and Ψ.

Estimation–Maximum Likelihood Method ii

- However, because of the factor rotation problem, we need to impose a constraint in order to obtain a unique solution:
  - $L^T \Psi^{-1} L = \Delta$ is diagonal.
- Given this assumption, the maximum likelihood estimates $\hat{L}$ and $\hat{\Psi}$ can be found using an iterative algorithm.
- We will not go into the details of the algorithm (but see Johnson & Wichern, Supplement 9A if interested).
  - Instead, we will rely on the `R` function `stats::factanal`.

Example (cont'd) i

```
# CAREFUL: uses correlation matrix
fa_decomp <- factanal(data, factors = m,
                      rotation = 'none')

# Uniquenesses are the diagonal elements
# of the matrix Psi
Psi_mle <- fa_decomp$uniquenesses
Lmle <- fa_decomp$loadings
```

Example (cont'd) ii

```r
# We get an estimate of the correlation
R_mle <- tcrossprod(Lmle) + diag(Psi_mle)

sum((cor(data) - R_mle)^2)
```

```
## [1] 0.2357231
```

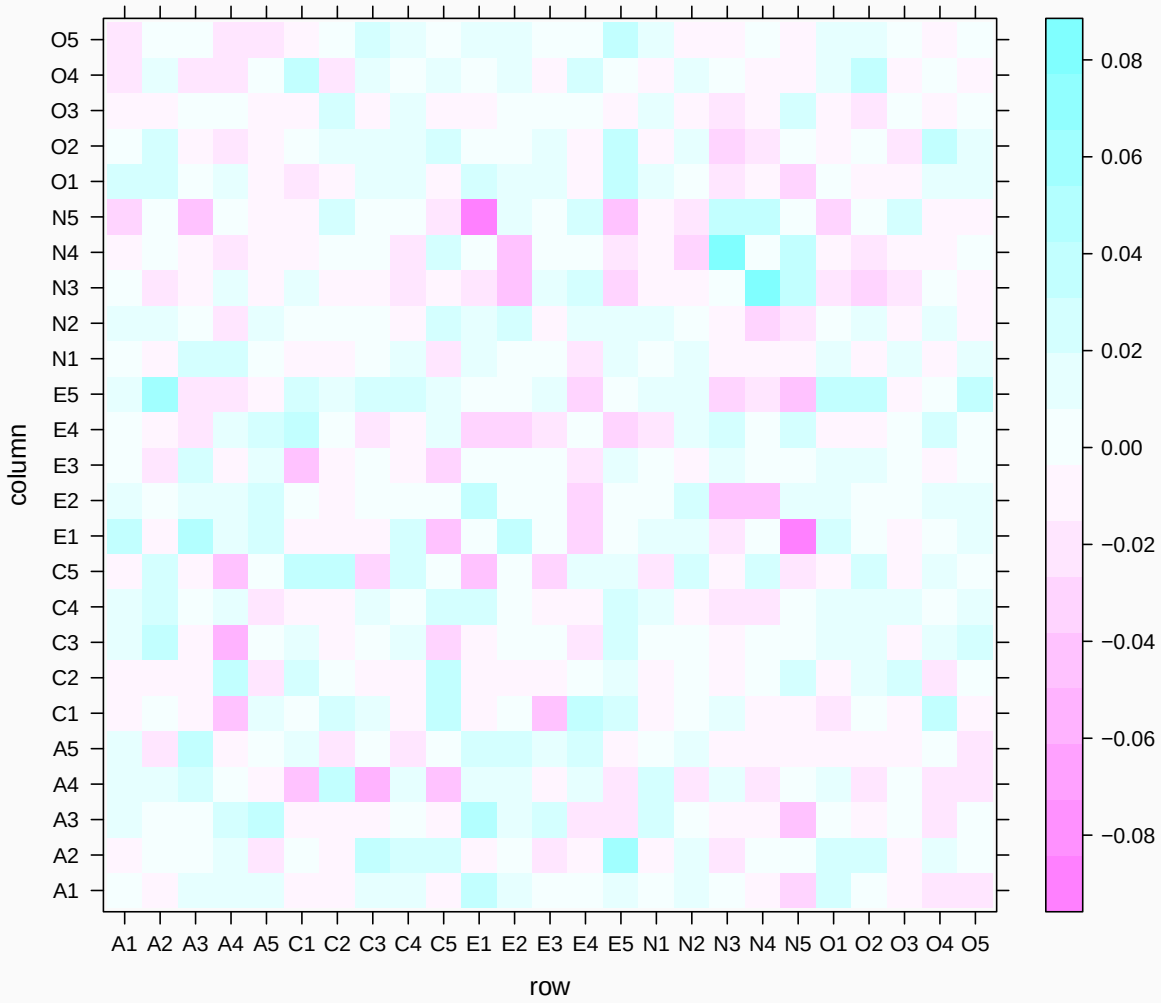
```r
# Our FA model explains:
sum(colSums(Lmle^2)/ncol(data))
```

```
## [1] 0.4498342
```

Example (cont'd) iii

```
levelplot(cor(data) - R_mle)
```

Example (cont'd) iv



Example (cont'd) v

```r
# To factor the covariance matrix
# Use psych::fa
fa_decomp <- psych::fa(data, nfactors = m,
                       rotate = "none",
                       covar = TRUE,
                       fm = "ml")

# Extract estimates
Psi_mle <- fa_decomp$uniquenesses
Lmle <- fa_decomp$loadings
```

Example (cont'd) vi

```
# We get an estimate of the covariance
Sn_mle <- tcrossprod(Lmle) + diag(Psi_mle)

sum((Sn - Sn_mle)^2)
```

```
## [1] 1.068995
```

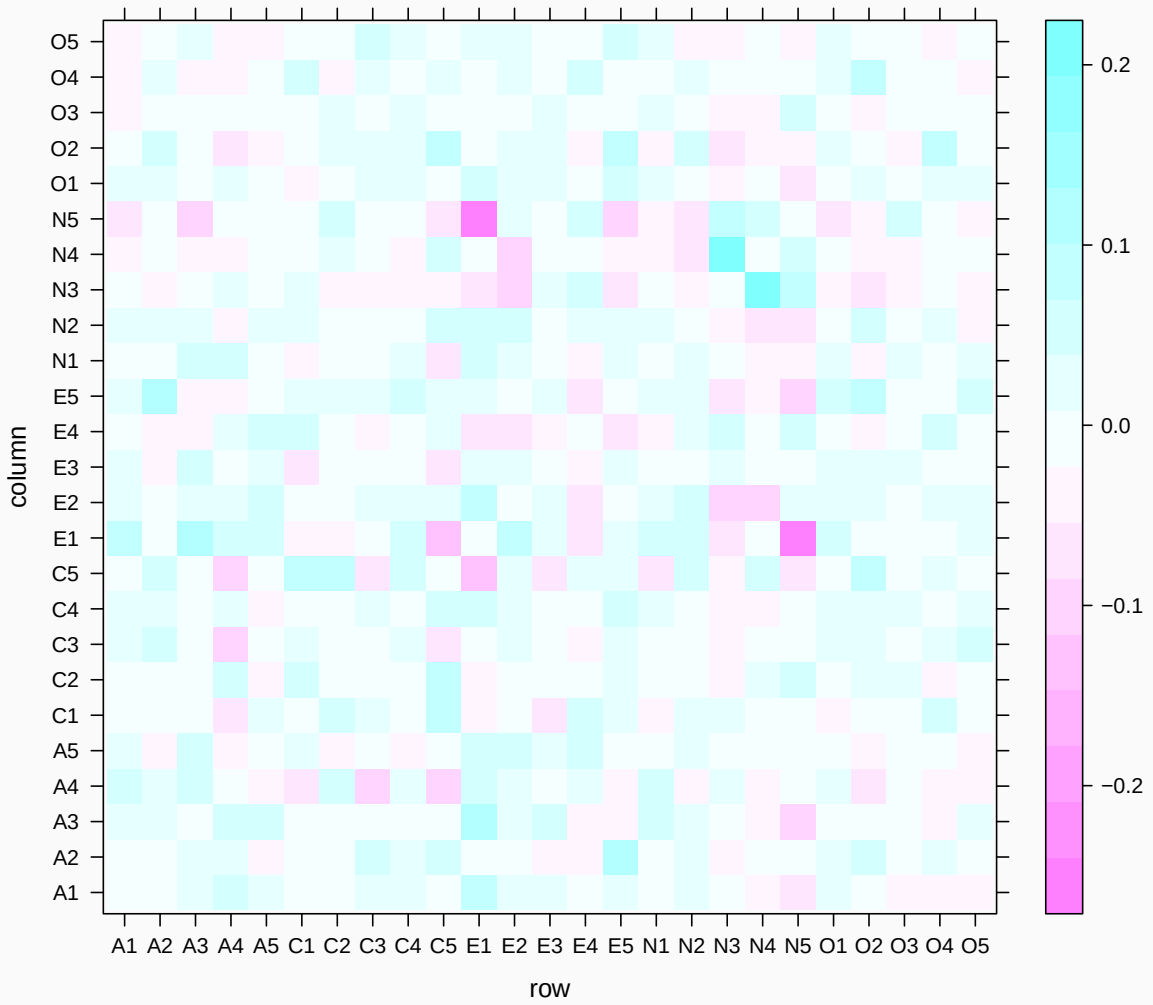
```
# Our FA model explains:
sum(colSums(Lmle^2)/sum(diag(Sn)))
```

```
## [1] 0.459665
```

Example (cont'd) vii

```
levelplot(Sn - Sn_mle)
```

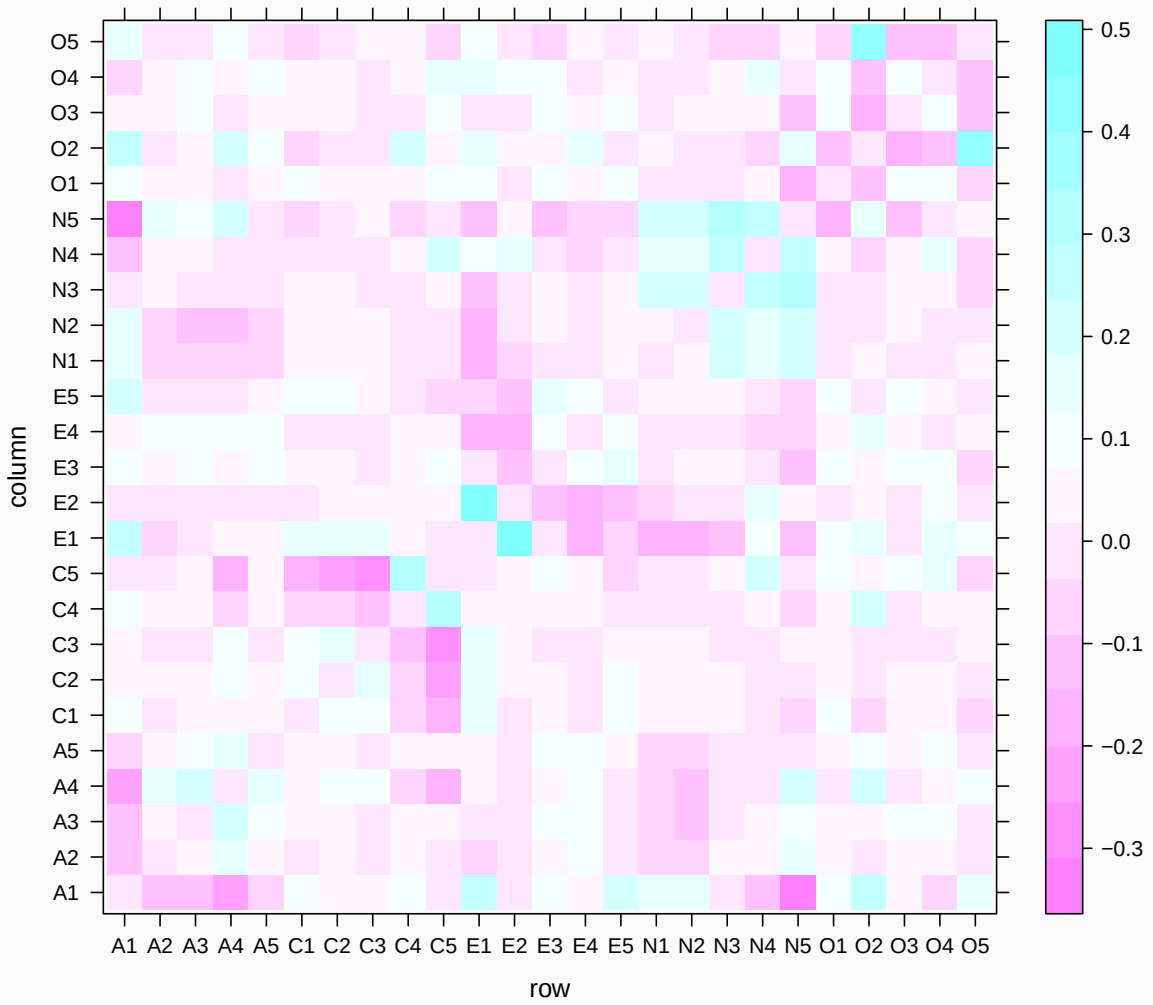
Example (cont'd) viii



Example (cont'd) ix

```
# Compare MLE with PC estimate
levelplot(Sn_fit - Sn_mle)
```

Example (cont'd) x



Comments about estimation i

- There are other methods of estimating the loadings and the "uniquenesses"
  - Ordinary Least Squares
  - Weighted OLS
  - Principal factor
- With so many choices of estimation methods, it can be hard to compare statistical softwares
  - And you also have to read the manual in order to know what is going on...
- You also always have the choice between factoring the covariance or the correlation matrix

Comments about estimation ii

    - Which one you choose depends on the commensurability of your variables (just like in PCA)
- Finally, it's always a good idea to compare the output of multiple estimation strategies
    - If your model is a good fit, you should get a similar answer regardless of the method.

Factor Rotation Redux i

- As we saw earlier, any orthogonal matrix T gives rise to the same factor analysis model

$$\Sigma = LL^T + \Psi = LTT^TL^T + \Psi = \tilde{L}\tilde{L}^T + \Psi.$$

- In other words, we cannot choose T to maximise the goodness of fit.
  - We need another criterion
- Intuitively, to ease interpretation, we want each variable to have large loadings for one factor and negligible loadings for the other ones.

Factor Rotation Redux ii

  - `https://maxturgeon.ca/f19-stat4690/factor_rotation.gif`
- One common analytic criterion that formalises this idea is the **varimax criterion**.
  - Resulting loadings are called *varimax loadings*
- We have

$$\text{VARIMAX} \propto \sum_{j=1}^{m} \begin{pmatrix} \text{Variance of squares of scales loadings} \\ \text{for } j\text{-th factor} \end{pmatrix}$$

- More precisely:
  - Let $\tilde{\ell}_{ij}$ be the (i, j)-th entry of the matrix $\tilde{L} = LT$. In other words, $\tilde{\ell}_{ij}$ depends on T.

Factor Rotation Redux iii

- Let $\tilde{h}_i^2 = \sum_{j=1}^m \tilde{\ell}_{ij}^2$.
- Define the scaled loadings $\tilde{\ell}_{ij}^* = \tilde{\ell}_{ij}/h_i$.
- The varimax criterion V is given as

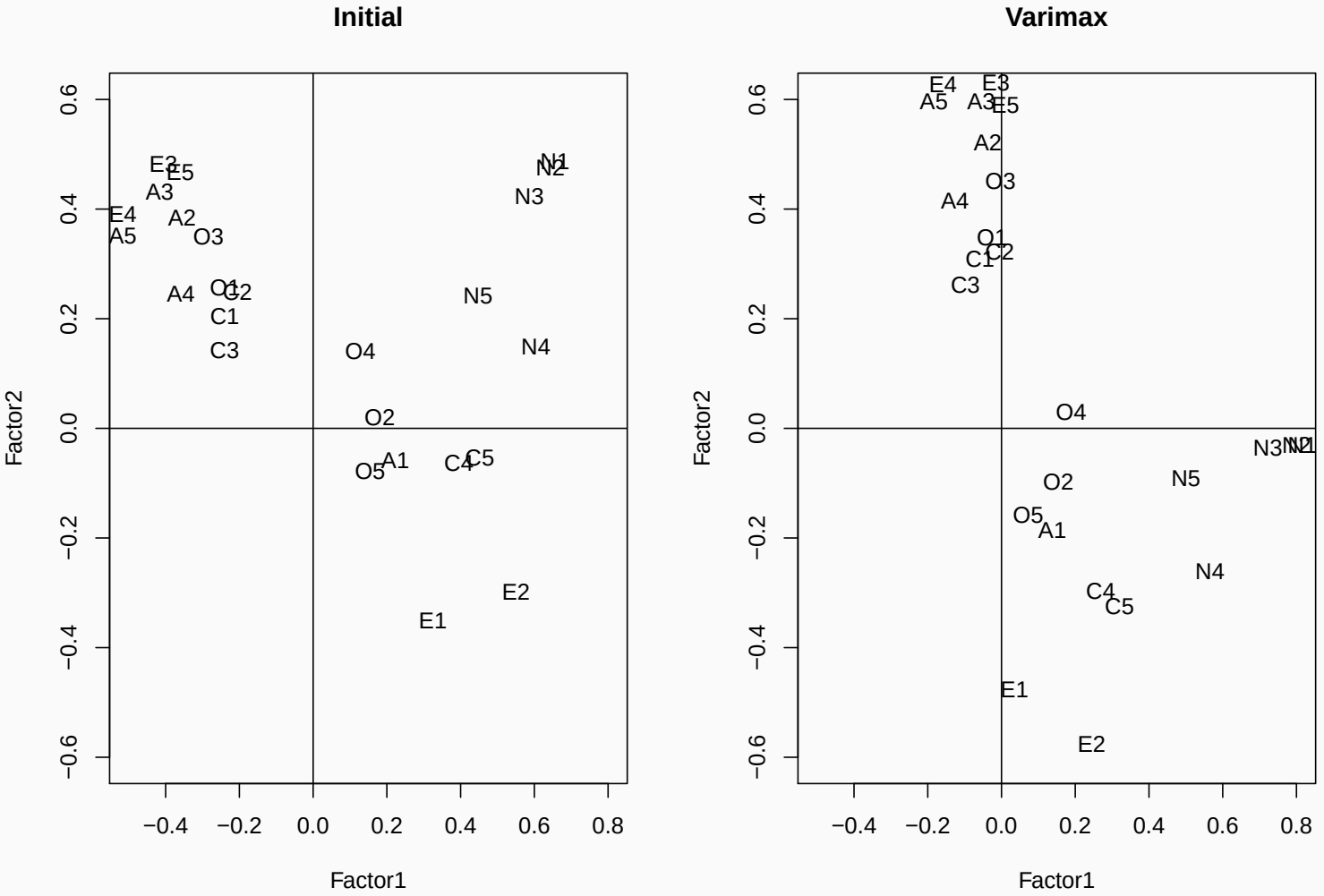
$$V = \frac{1}{p}\sum_{j=1}^{m}\left(\sum_{i=1}^{p}\tilde{\ell}_{ij}^{*4} - \frac{1}{p}\left(\sum_{i=1}^{p}\tilde{\ell}_{ij}^{*2}\right)^2\right).$$

- In `R`, you can compute the rotated loadings using the `stats::varimax` function. Alternatively, the function `stats::factanal` can compute the rotation for you as part of the factor analysis (and so does `psych::fa`).

Example (cont'd) i

```
# Let's start with m=2 for visualization
fa_decomp <- factanal(data, factors = 2,
                      rotation = 'none')

initial_loadings <- fa_decomp$loadings
varimax_loadings <- varimax(initial_loadings)
```

Example (cont'd) ii



Example (cont'd) iii

```
# You can extract the matrix T
varimax_loadings$rotmat
```

```
##             [,1]       [,2]
## [1,] 0.7810310 -0.6244923
## [2,] 0.6244923  0.7810310
```

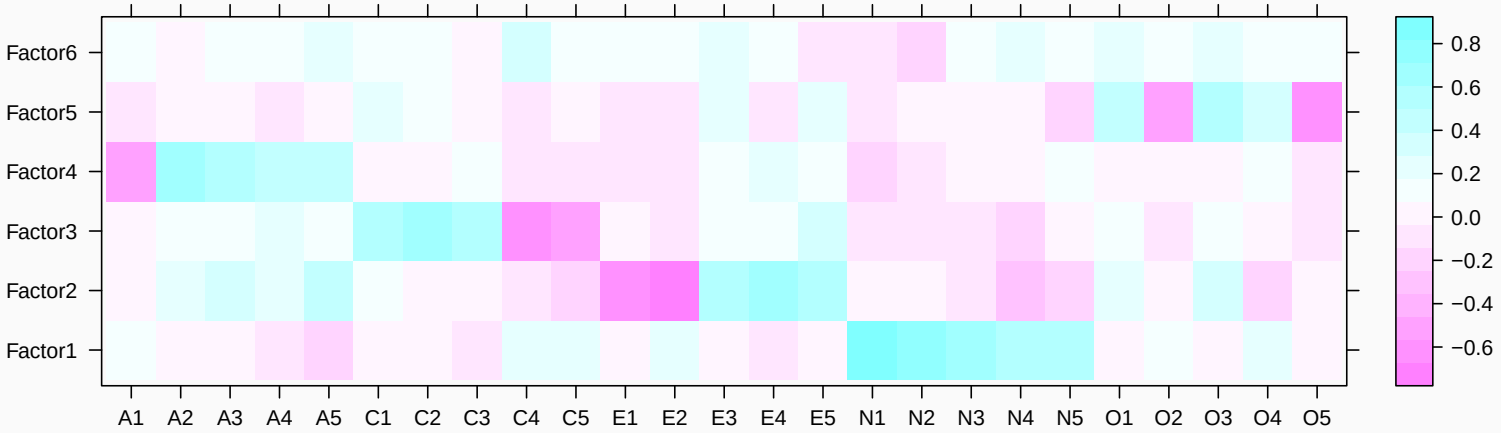
```
# We can also get the angle of rotation
acos(varimax_loadings$rotmat[1,1])
```

```
## [1] 0.6744813
```

Example (cont'd) iv

```
# In more dimensions
fa_decomp <- factanal(data, factors = m,
                      rotation = 'varimax')

levelplot(unclass(fa_decomp$loadings),
          xlab = "", ylab = "")
```

Example (cont'd) v



Comments

- As with estimation, there are many more rotation methods.
  - See for example the help page `?GPArotation::rotations`
- One particular class of rotations are called *oblique*
  - The matrix T is no longer constrained to be orthogonal.
- Factor rotation is especially useful with loadings obtained through MLE
  - Recall the constraint on $L^T \Psi^{-1} L$ being diagonal
- Factor rotations are also sometimes used with PCA loadings.

Selecting the number of factors i

- We have discussed some of these strategies already.
- We could select the minimum number of factors m that explains a certain proportion of variance.
  - Let L be the matrix of loadings and for each factor j we let

$$\frac{\sum_{i=1}^{p} \ell_{ij}^2}{\operatorname{tr}(S_n)}$$

    be the proportion of total sample variance due to the j-th factor.
  - In other words, when computing the proportion of variance due to the factor model, we do not include the variance coming from the error term.

Selecting the number of factors ii

- We could use a scree plot approach:
  - Fit different factor analysis models with a varying value of m and plot the total proportion of variance explained as a function of m.
  - You could also select m using a scree plot based on the eigenvalues of S_n (just as in PCA).
- We could also select m as the number of eigenvalues of S_n that are larger than the average eigenvalue.
- We could also select only factors that can be explained by the researcher.
  - This requires a lot of domain-knowledge expertise.

Example (cont'd) i

```
# First, let's look at the average eigenvalue
decomp <- eigen(Sn, symmetric = TRUE,
                only.values = TRUE)

mean(decomp$values)
```

```
## [1] 2.009715
```

```
sum(decomp$values > mean(decomp$values))
```

```
## [1] 6
```

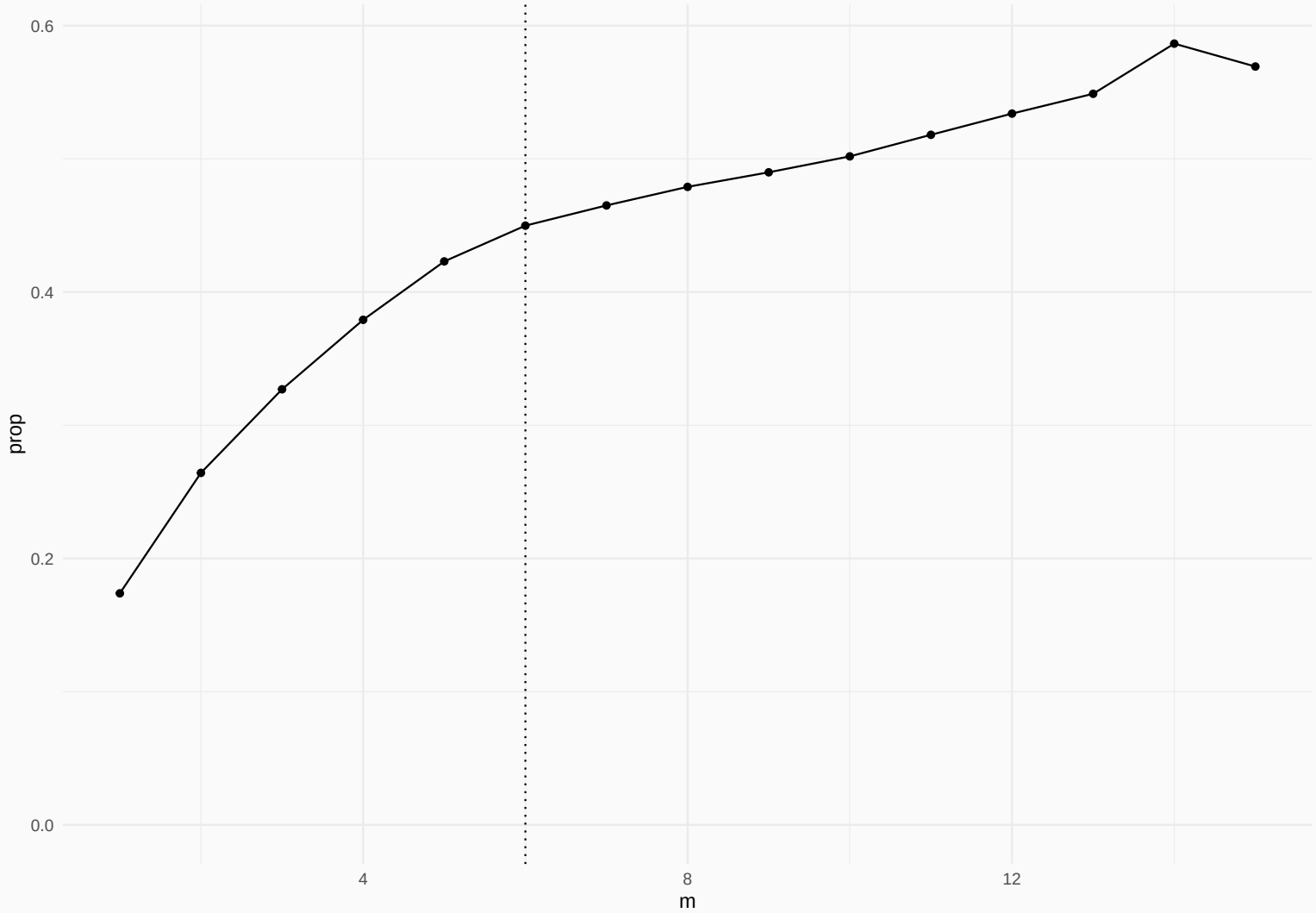
Example (cont'd) ii

```
# We will go from 1 factor to 15
prop_var_explained <- purrr::map_df(
  seq(1, 15),  function(m) {
    fa_decomp <- factanal(data, factors = m)
    Lmle <- fa_decomp$loadings
    prop <- sum(colSums(Lmle^2))/ncol(data)

    data.frame(
      prop = prop,
      m = m
    )})
```

Example (cont'd) iii

```
prop_var_explained %>%
  ggplot(aes(m, prop)) +
  geom_point() +
  geom_line() +
  theme_minimal() +
  expand_limits(y = 0) +
  geom_vline(xintercept = 6,
             linetype = 'dotted')
```

Example (cont'd) iv



Information criteria

- We will discuss two more approaches.
- First, if we use the MLE estimation method, we can also use information criteria.
  - Bayesian Information Criterion (BIC)
  - empirically-derived BIC (eBIC)
  - sample-size adjusted BIC (saBIC)
- These 3 criteria have been implemented in `psych::fa`.
- To use these criteria, fit the model with different values of m and select the model with the smallest value of the criterion.

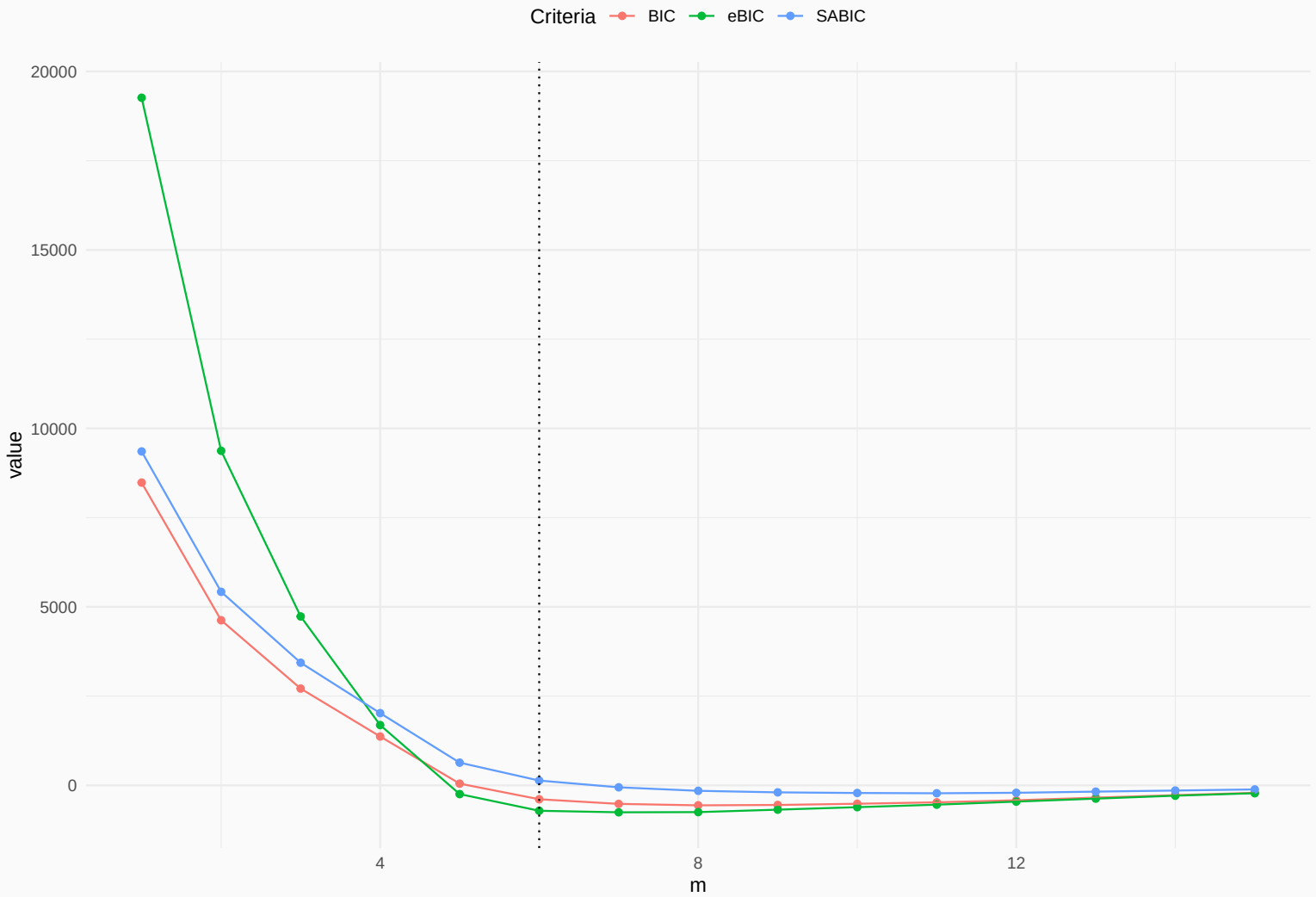
Example (cont'd) i

```
inform_crit <- purrr::map_df(
  seq(1, 15),  function(m) {
    fa_decomp <- psych::fa(data, nfactors = m,
                           fm = 'ml')
    data.frame(
      BIC = fa_decomp$BIC, eBIC = fa_decomp$EBIC,
      SABIC = fa_decomp$SABIC, m = m
    )})
```

Example (cont'd) ii

```
inform_crit %>%
  gather(Criteria, value, -m) %>%
  ggplot(aes(m, value, colour = Criteria)) +
  geom_point() +
  geom_line() +
  theme_minimal() +
  theme(legend.position = 'top') +
  geom_vline(xintercept = 6,
             linetype = 'dotted')
```

Example (cont'd) iii



Example (cont'd) iv

```
inform_crit %>%
  gather(Criteria, value, -m) %>%
  group_by(Criteria) %>%
  filter(value == min(value)) %>%
  select(-value) %>%
  spread(Criteria, m)
```

```
## # A tibble: 1 x 3
##     BIC  eBIC SABIC
##   <int> <int> <int>
## 1     8     7    11
```

Large sample test i

- The second alternative approach is based on hypothesis testing.
- We will perform a different test for each value of m.
- For a fixed m, the *null hypothesis* is that the corresponding factor model

$$\Sigma = LL^T + \Psi$$

  is correct.
- The alternative hypothesis is that Σ is *unstructured*.
  - Note that the different factor models with different m are not nested.

Large sample test ii

- Under H_0, the likelihood of the model is proportional to

$$|\hat{L}\hat{L}^T + \hat{\Psi}|^{-n/2} \exp\left(-\frac{1}{2} n \mathrm{tr}\left((\hat{L}\hat{L}^T + \hat{\Psi})^{-1} S_n\right)\right).$$

- Therefore, the likelihood ratio statistic is

$$-2 \log \Lambda = n \log\left(\frac{|\hat{L}\hat{L}^T + \hat{\Psi}|}{|S_n|}\right).$$

- Under H_0, this follows a χ^2 distribution with ν degrees of freedom, where

$$\nu = \frac{1}{2}\left((p-m)^2 - p - m\right).$$

Example (cont'd) i

```
m <- 6
fa_decomp <- psych::fa(data, nfactors = m,
                       covar = TRUE,
                       fm = 'ml')
```

Example (cont'd) ii

```
Sn <- cov(data)
Sn_mle <- tcrossprod(fa_decomp$loadings) +
  diag(fa_decomp$uniquenesses)

test_stat <- nrow(data) * (
  log(det(Sn_mle)) -
    log(det(Sn))
)
```

Example (cont'd) iii

```
p <- ncol(data)
nu <- 0.5*((p-m)^2 - p - m)

test_stat > qchisq(0.95, df = nu)
```

```
## [1] TRUE
```

Comments

- The hypothesis test approach is nor really recommended:
  - It tends to select too many factors, especially when the sample size is large.
- If a factor model is good model for a given dataset, the "average eigenvalue" method should given a similar answer to the scree plot approach.

Factor scores i

- Recall the original model:

$$\mathbf{Y} - \mu = L\mathbf{F} + \mathbf{E}.$$

- We have discussed ways to estimate L and the covariance matrix of $\mathbf{E}$, but we have not covered how to estimate $\mathbf{F}$ yet.
- In a typical analysis, we may be mainly interested in estimating and inspecting the loadings.
- But it may be useful to estimate the common factors for diagnostic purposes (e.g. model fit) or as input into a second analysis (e.g. two-stage modeling).

Factor scores ii

- We will discuss two estimation strategies:
  1. Weighted Least Squares
  2. Regression

Weighted Least Squares Method i

- First, assume that we know L, μ, Ψ.
- If we regard $\mathbf{E}$ as error terms, we would want to minimise

$$(\mathbf{Y} - \mu - L\mathbf{F})^T(\mathbf{Y} - \mu - L\mathbf{F}).$$

- However, since the errors terms have unequal variance (i.e. heteroscedasticity), we may have a better performance by weighting the j-th term by the inverse of ψ_j. We therefore have a different criterion to minimise:

$$(\mathbf{Y} - \mu - L\mathbf{F})^T\Psi^{-1}(\mathbf{Y} - \mu - L\mathbf{F}).$$

Weighted Least Squares Method ii

- Minimising with respect to $\mathbf{F}$, we get

$$\hat{\mathbf{F}} = (L^T \Psi^{-1} L)^{-1} L^T \Psi^{-1} (\mathbf{Y} - \mu).$$

- Using this computation as a heuristic, we can replace L, Ψ, μ by $\hat{L}, \hat{\Psi}, \bar{\mathbf{Y}}$, and for each observation $\mathbf{Y}_i$, we get an estimate

$$\hat{\mathbf{F}}_j = (\hat{L}^T \hat{\Psi}^{-1} \hat{L})^{-1} \hat{L}^T \hat{\Psi}^{-1} (\mathbf{Y}_j - \bar{\mathbf{Y}}).$$

- **Note**: When $\hat{L}, \hat{\Psi}$ are obtained using the principal component approach, it is customary to use the *unweighted* approach, in which case the estimates $\hat{\mathbf{F}}_j$ are simply the principal components of $\mathbb{Y}$.

Example (cont'd) i

```
m <- 6
fa_decomp <- psych::fa(data, nfactors = m,
                       covar = TRUE,
                       fm = 'ml')
```

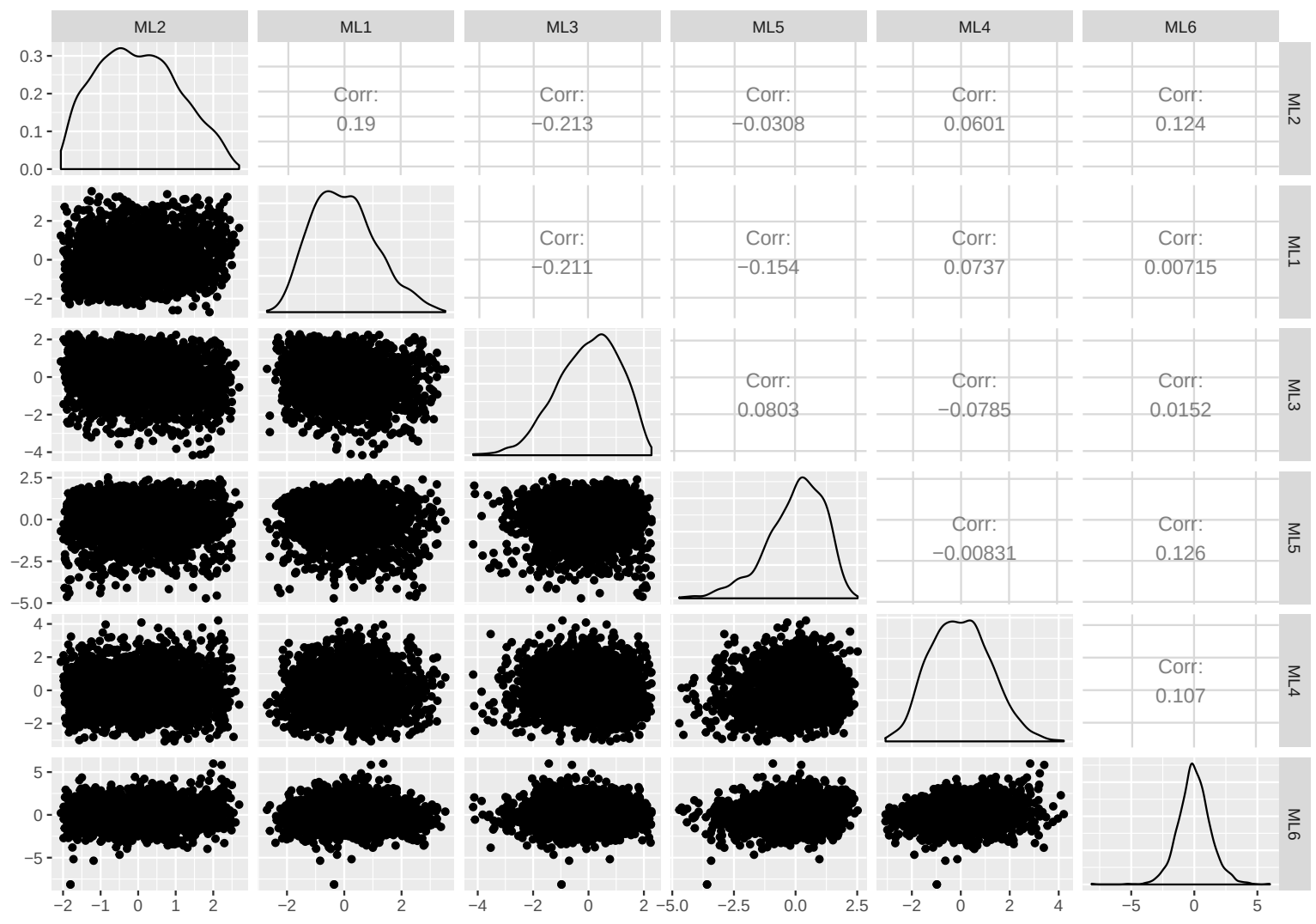
Example (cont'd) ii

```
# Extract estimates
InvPsi <- diag(fa_decomp$uniquenesses^-1)
Lhat <- fa_decomp$loadings

hat_mat <- solve(t(Lhat) %*% InvPsi %*% Lhat) %*%
  t(Lhat) %*% InvPsi

scores <- scale(data, center = TRUE,
                scale = FALSE) %*% t(hat_mat)
```

Example (cont'd) iii

```
GGally::ggpairs(as.data.frame(scores))
```

Example (cont'd) iv



Regression Method i

- As before, assume that we know L, μ, Ψ.
- Under the normality assumptions $\mathbf{F} \sim N_m(0, I)$ and $\mathbf{E} \sim N_p(0, \Psi)$, we can conclude that $(\mathbf{F}, \mathbf{E})$ and therefore $(\mathbf{Y} - \mu, \mathbf{F})$ are also jointly normal.
- More precisely, $(\mathbf{Y} - \mu, \mathbf{F})$ is $N_{m+p}(0, \Sigma^*)$, where

$$\Sigma^* = \begin{pmatrix} LL^T + \Psi & L \\ L^T & I \end{pmatrix}.$$

- Using our result on conditional distributions (see slides on Multivariate Normal Distribution), we know that the distribution of $\mathbf{F}$ given $\mathbf{Y}$ has:

Regression Method ii

  - Mean $L^T(LL^T + \Psi)^{-1}(\mathbf{Y} - \mu)$;
  - Covariance $I - L^T(LL^T + \Psi)^{-1}L$.
- Therefore, given estimates $\hat{L}, \hat{\Psi}$, we get an estimate of the scores as:

$$\hat{\mathbf{F}}_j = \hat{L}^T \left(\hat{L}\hat{L}^T + \hat{\Psi}\right)^{-1} (\mathbf{Y}_j - \bar{\mathbf{Y}}).$$

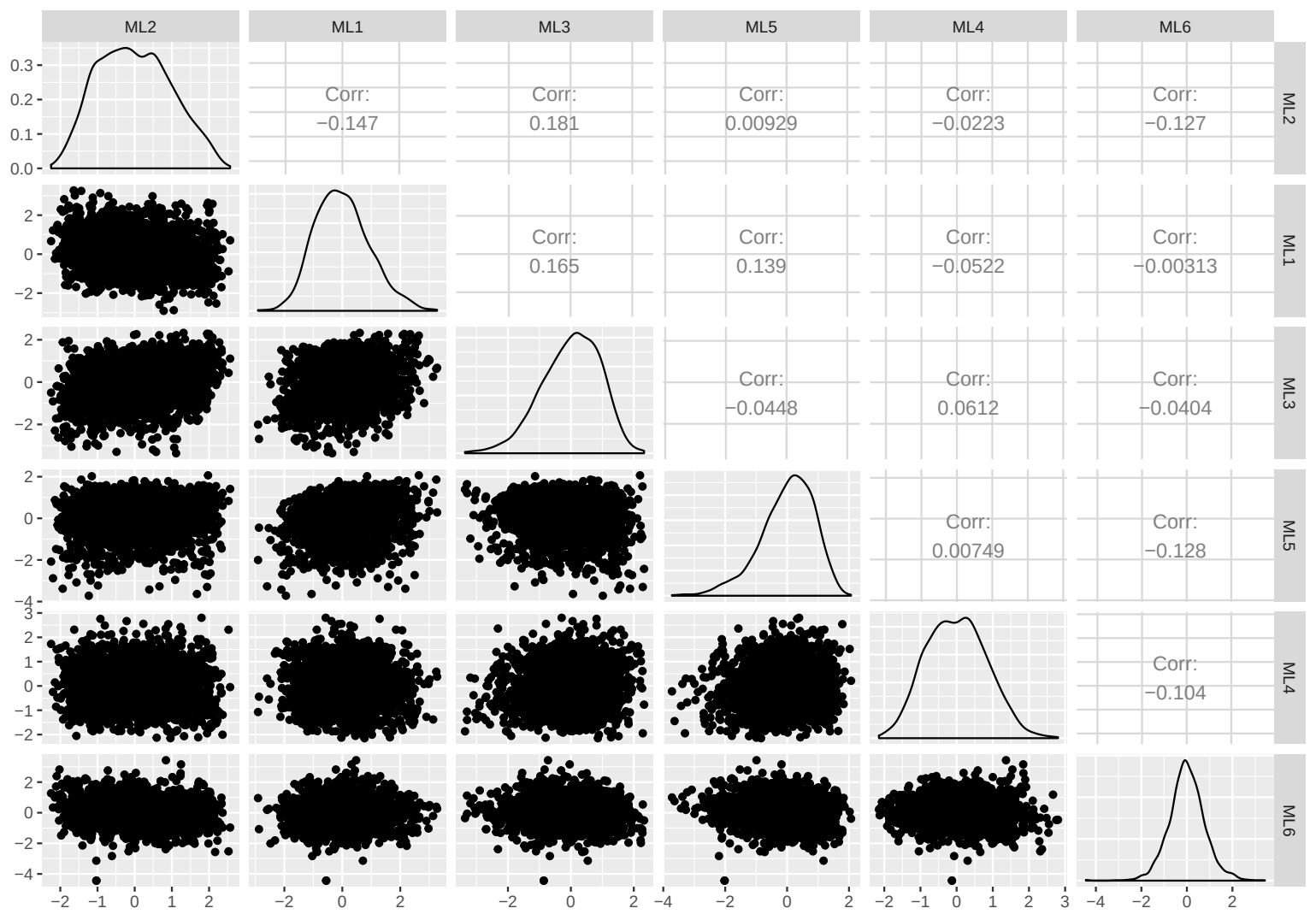
- To mitigate the effects of model misspecification, it is common to replace $\hat{L}\hat{L}^T + \hat{\Psi}$ by the sample covariance matrix S_n.

Example (cont'd) i

```
scores_reg <- scale(data, center = TRUE,
                          scale = FALSE) %*%
  solve(Sn) %*% Lhat
```

```
GGally::ggpairs(as.data.frame(scores_reg))
```

Example (cont'd) ii



Example (cont'd) iii

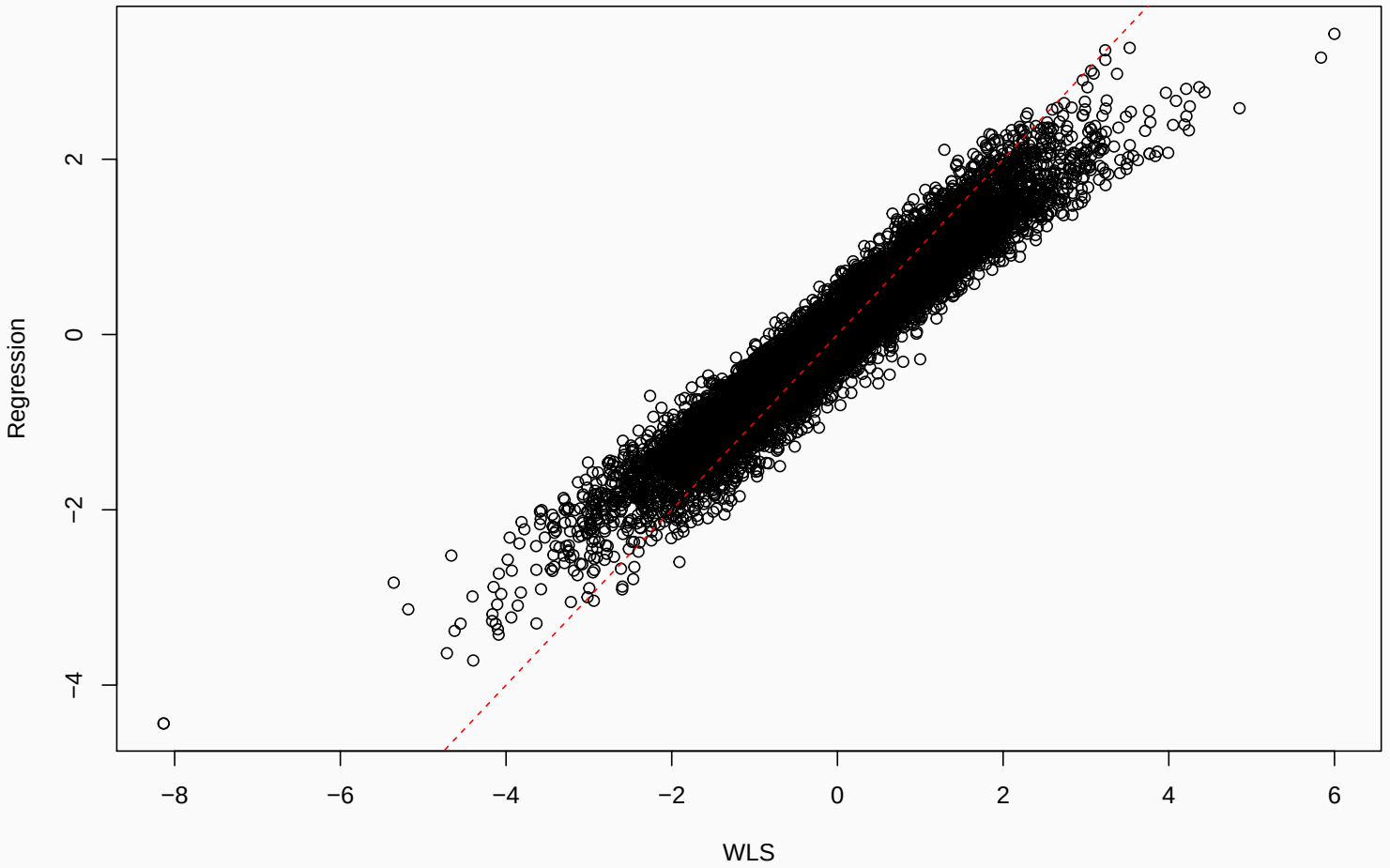
```
# Let's look at agreement
round(cor(scores, scores_reg), 2)
```

```
##       ML2  ML1  ML3  ML5  ML4  ML6
## ML2 0.96 0.00 0.00 0.00 0.00 0.00
## ML1 0.00 0.96 0.00 0.00 0.00 0.00
## ML3 0.00 0.00 0.96 0.00 0.00 0.00
## ML5 0.00 0.00 0.00 0.98 0.00 0.00
## ML4 0.00 0.00 0.00 0.00 0.99 0.00
## ML6 0.00 0.00 0.00 0.00 0.00 0.98
```

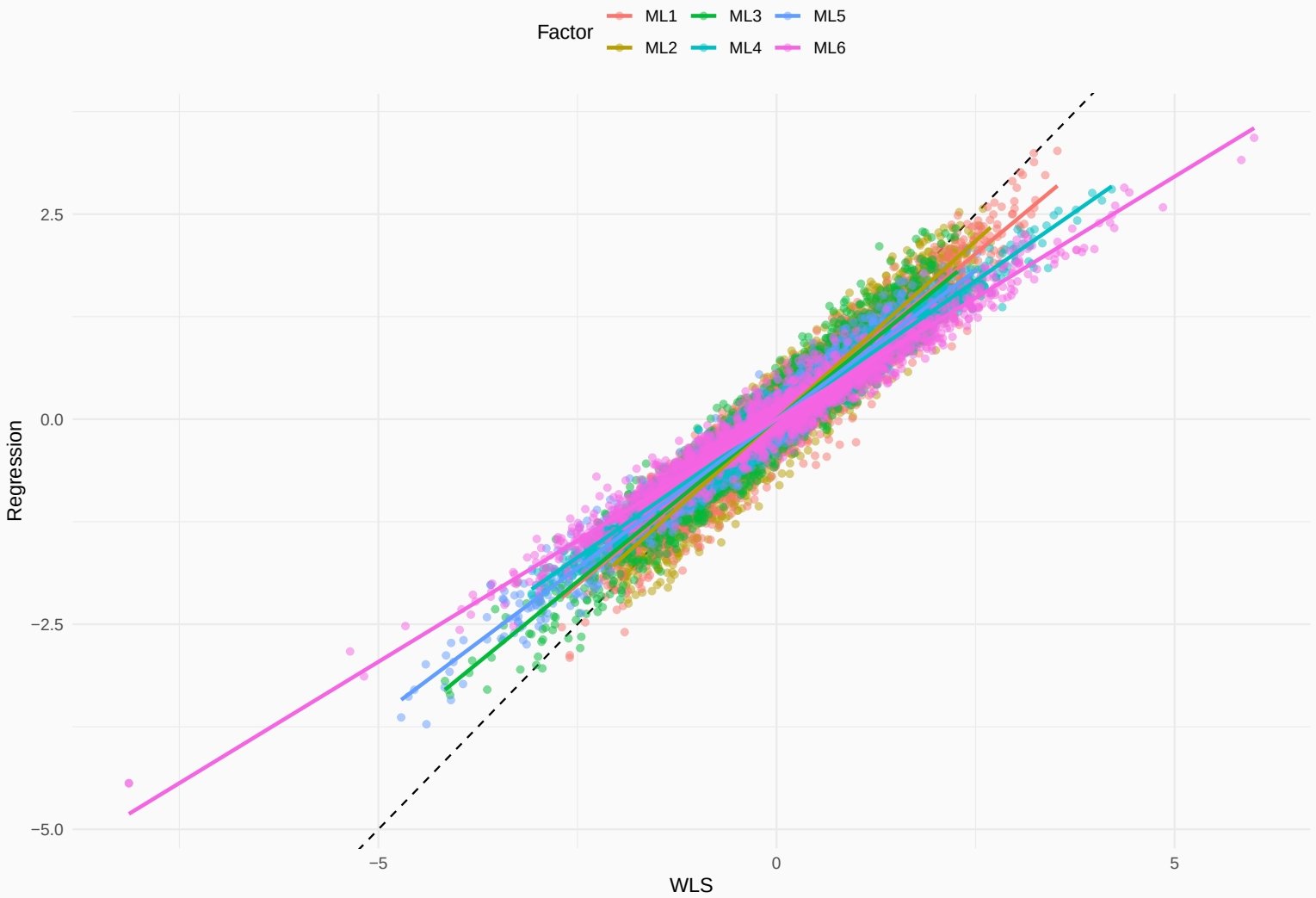
Example (cont'd) iv

```
# Or graphically
plot(as.vector(scores),
     as.vector(scores_reg),
     xlab = "WLS",
     ylab = "Regression")
abline(a = 0, b = 1,
       col='red',
       lty = 2)
```

Example (cont'd) v



Example (cont'd) vi



Comments

- Neither method is uniformly superior.
  - On the other hand, the regression method uses normal theory to derive its heuristic.
- As for other topics we discussed earlier, there are other methods for estimating the scores
  - Harman's method
  - Anderson's method
  - ten Berge's method
- What is important to remember is that there is no best way to estimate the scores, and different methods optimise different criteria
  - E.g. Anderson's method looks for *uncorrelated* scores.

Final comments on Factor Analysis i

- The type of Factor Analysis we discussed is called **Exploratory**.
  - **Confirmatory** FA would make strong assumptions about the nature of the latent factors and perform statistical inference.
- At every stage of FA, there are choices to make: estimation method, number of factors, factor rotation, score estimation.
  - This makes FA more of an art than a science.
- But there are still some general principles you can follow.

Final comments on Factor Analysis ii

General Strategy for FA

1. Perform a *Principal Component* Factor Analysis.
   - It is simple to run.
   - It will help you find potential outliers.
2. Perform a *Maximum Likelihood* Factor Analysis.
   - Try a varimax rotation to see if it makes sense.
3. Compare the solution of both methods to see if they generally agree.
4. Repeat for different values of m and check if adding more factors improve interpretation of the results.
5. For large datasets, you can split your data, run the same model on both subsets, and compare the loadings to see if they generally agree.

A different example i

- The dataset: `state.x77`. It contains general information about all 50 states.
  - Population
  - Income per capita
  - Illiteracy rate
  - Life Expectancy
  - Murder rate
  - High-School Graduation rate
  - Average number of days below 0C
  - Total area

A different example ii

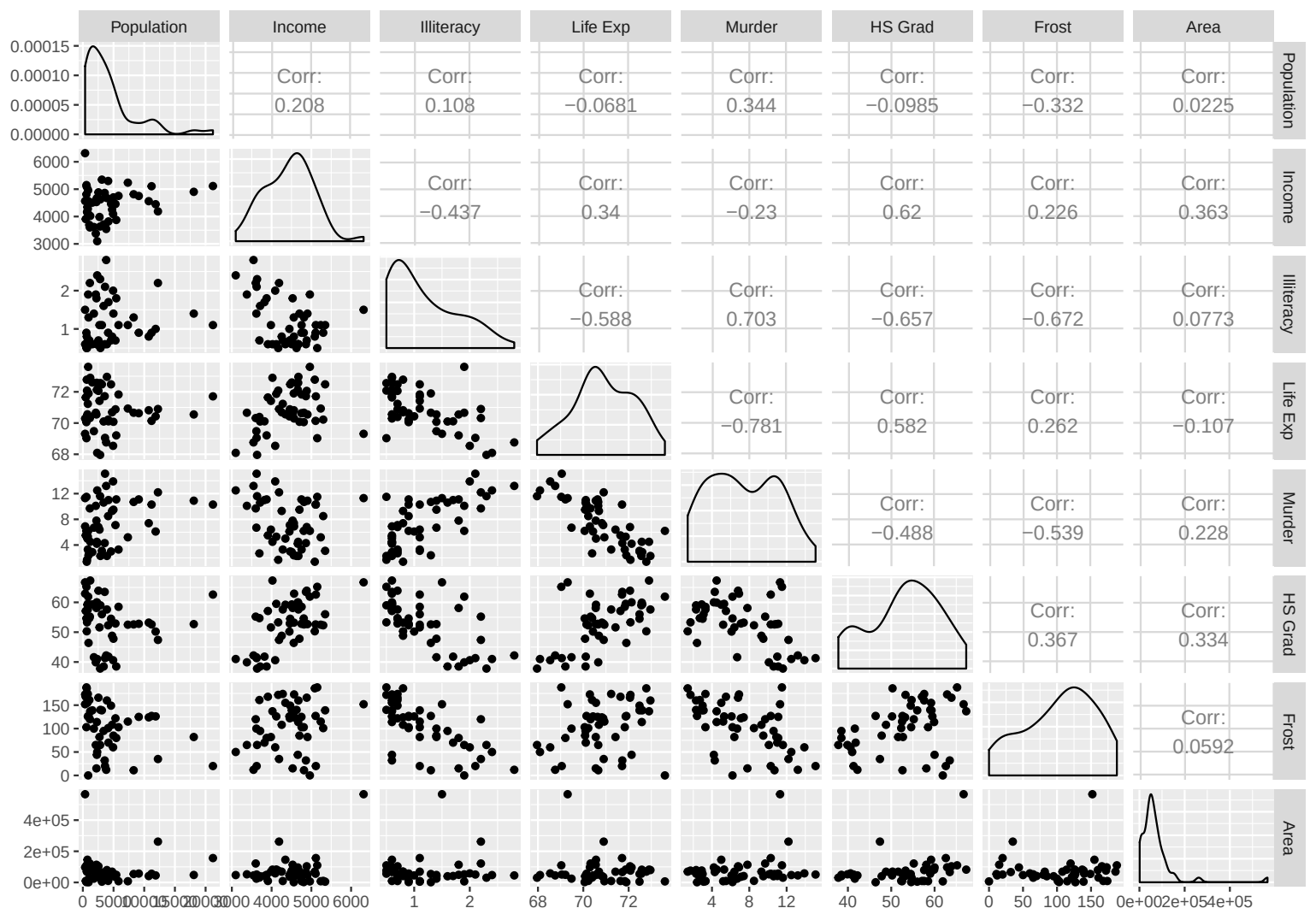
```
state.x77[1:5,1:4]
```

```
##            Population Income Illiteracy Life Exp
## Alabama          3615   3624        2.1    69.05
## Alaska            365   6315        1.5    69.31
## Arizona          2212   4530        1.8    70.55
## Arkansas         2110   3378        1.9    70.66
## California      21198   5114        1.1    71.71
```

A different example iii

```
library(GGally)
data <- as.data.frame(state.x77)
ggpairs(data)
```

A different example iv



A different example v

```
# Potential outliers?
data %>%
  rownames_to_column('State') %>%
  top_n(Population, n = 2)
```

```
##         State Population Income Illiteracy Life Exp M
## 1 California      21198   5114        1.1    71.71
## 2   New York      18076   4903        1.4    70.55
##     Area
## 1 156361
## 2  47831
```

A different example vi

```
data %>%
  rownames_to_column('State') %>%
  top_n(Area, n = 2)
```

```
##    State Population Income Illiteracy Life Exp Murde
## 1 Alaska        365   6315        1.5    69.31    11
## 2  Texas      12237   4188        2.2    70.90    12
```

```
# 1. Principal Component Factor analysis
decomp <- princomp(data, cor = TRUE)
decomp
```

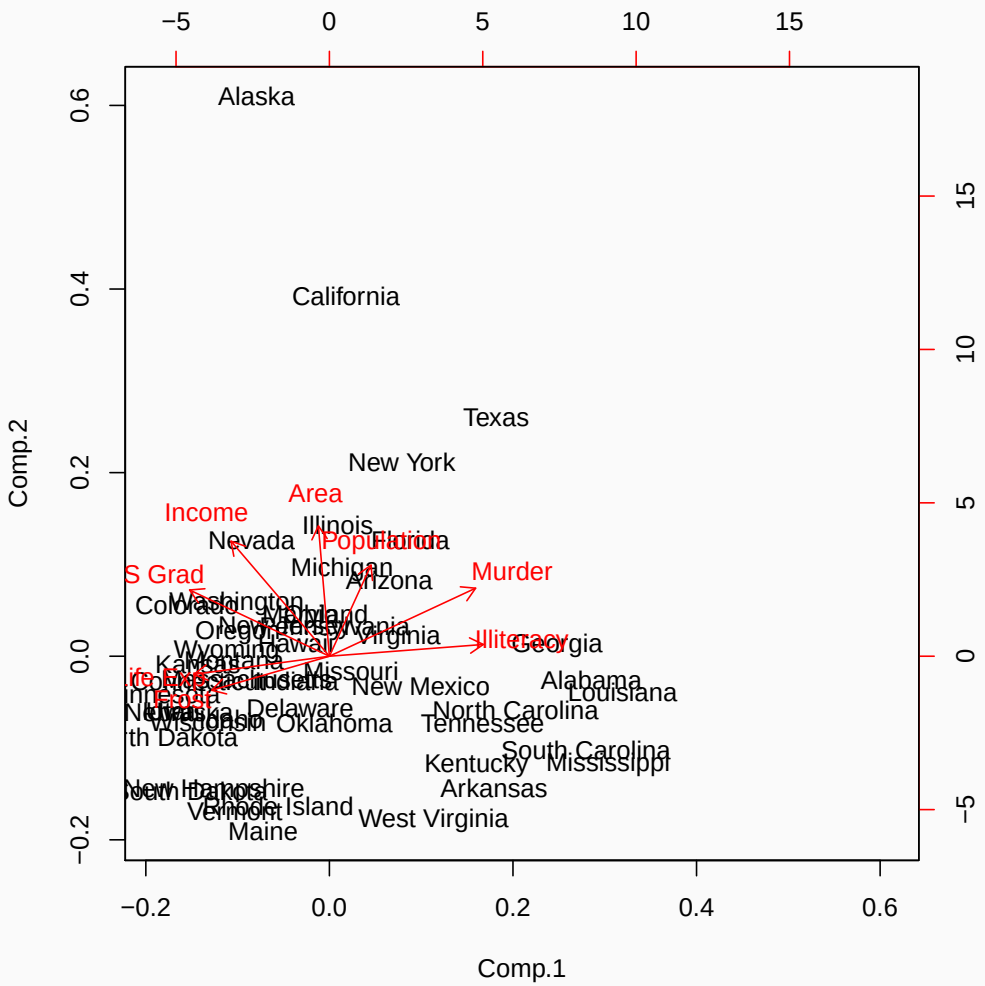
A different example vii

```
## Call:
## princomp(x = data, cor = TRUE)
##
## Standard deviations:
##    Comp.1    Comp.2    Comp.3    Comp.4    Comp.5
## 1.8970755 1.2774659 1.0544862 0.8411327 0.6201949 0
##    Comp.8
## 0.3364338
##
##  8  variables and  50 observations.
```

A different example viii

```
biplot(decomp)
```

A different example ix



A different example x

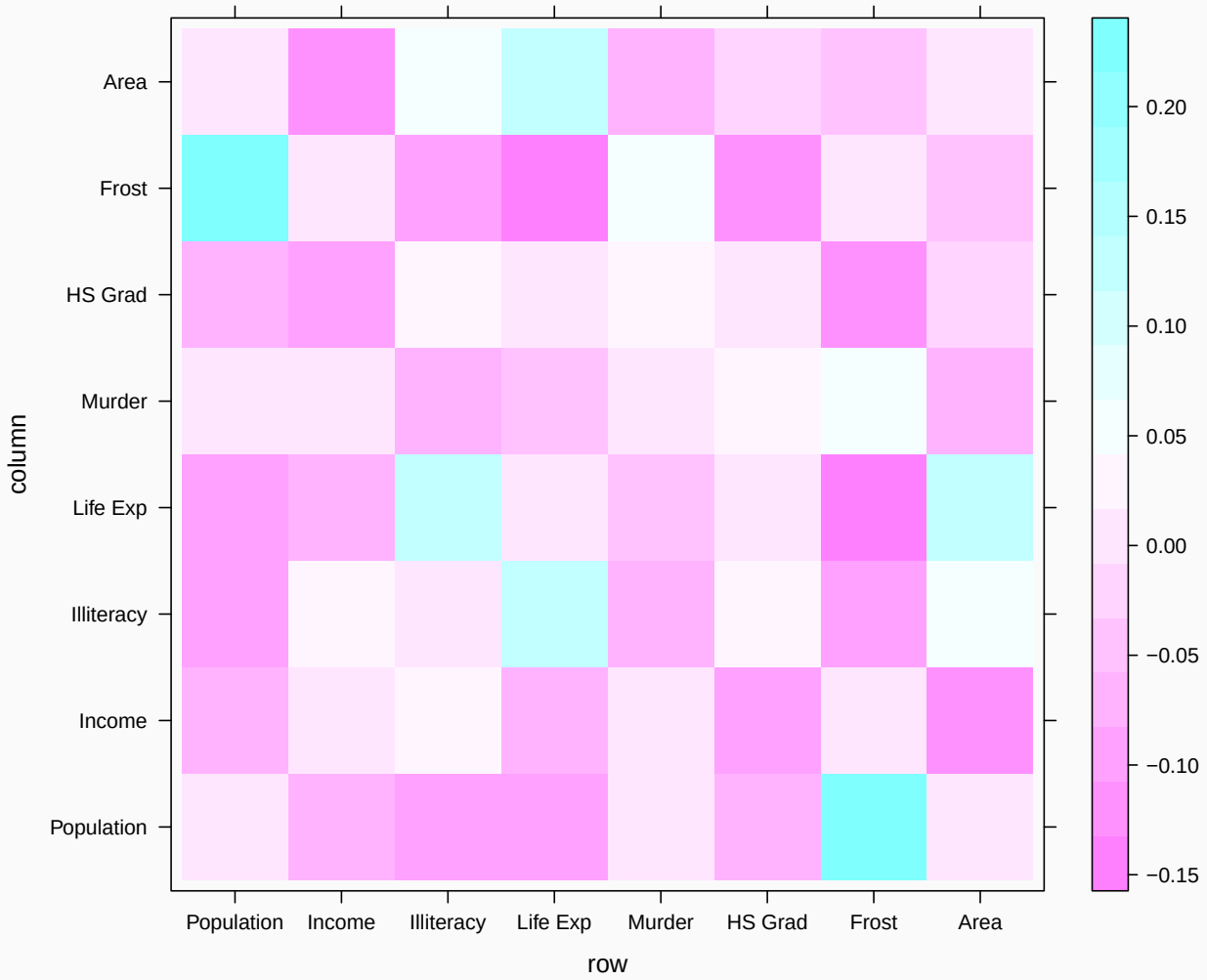
```
m <- 3
Lhat <- decomp$loadings[,seq_len(m)] %*%
  diag(decomp$sdev[seq_len(m)])
colnames(Lhat) <- paste0("PC", 1:3)
Psi_hat <- diag(cor(data) - tcrossprod(Lhat))

# Our FA model explains:
sum(colSums(Lhat^2))/ncol(data)
```

```
## [1] 0.7928445
```

A different example xi

```
# We can also  visualize the fit
Rn <- cor(data)
Rn_fit <- tcrossprod(Lhat) + diag(Psi_hat)

levelplot(Rn - Rn_fit)
```

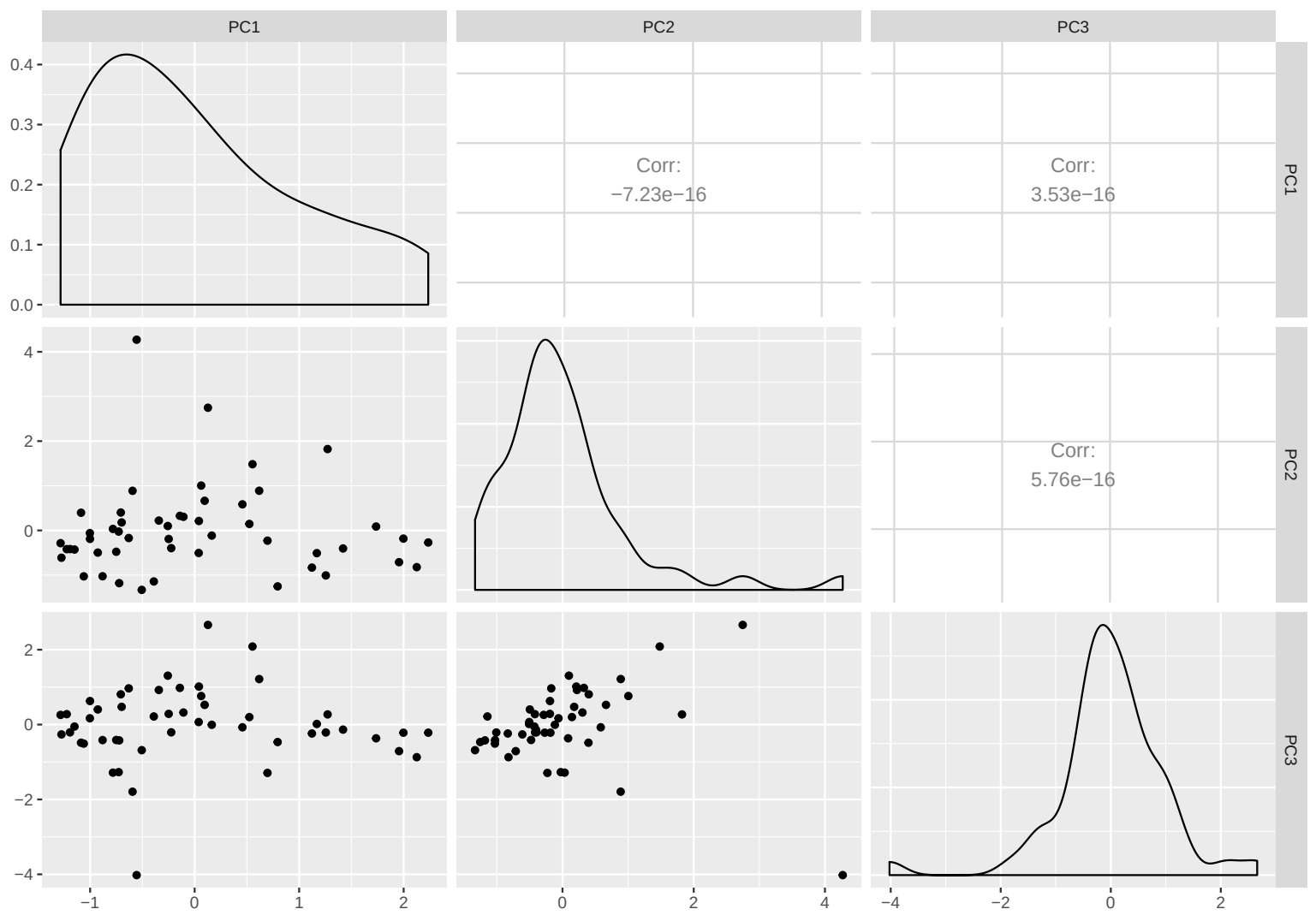
A different example xii



A different example xiii

```
scores_pc <- scale(data, center = TRUE,
                         scale = TRUE) %*%
  solve(Rn) %*% Lhat

ggpairs(as.data.frame(scores_pc))
```

A different example xiv



A different example xv

```r
# What state has the outlying values?
scores_pc %>%
  as.data.frame %>%
  rownames_to_column('State') %>%
  filter(PC2 > 4 | PC3 < -4)
```

```
##    State        PC1      PC2       PC3
## 1 Alaska -0.5551363 4.271092 -4.021475
```

A different example xvi

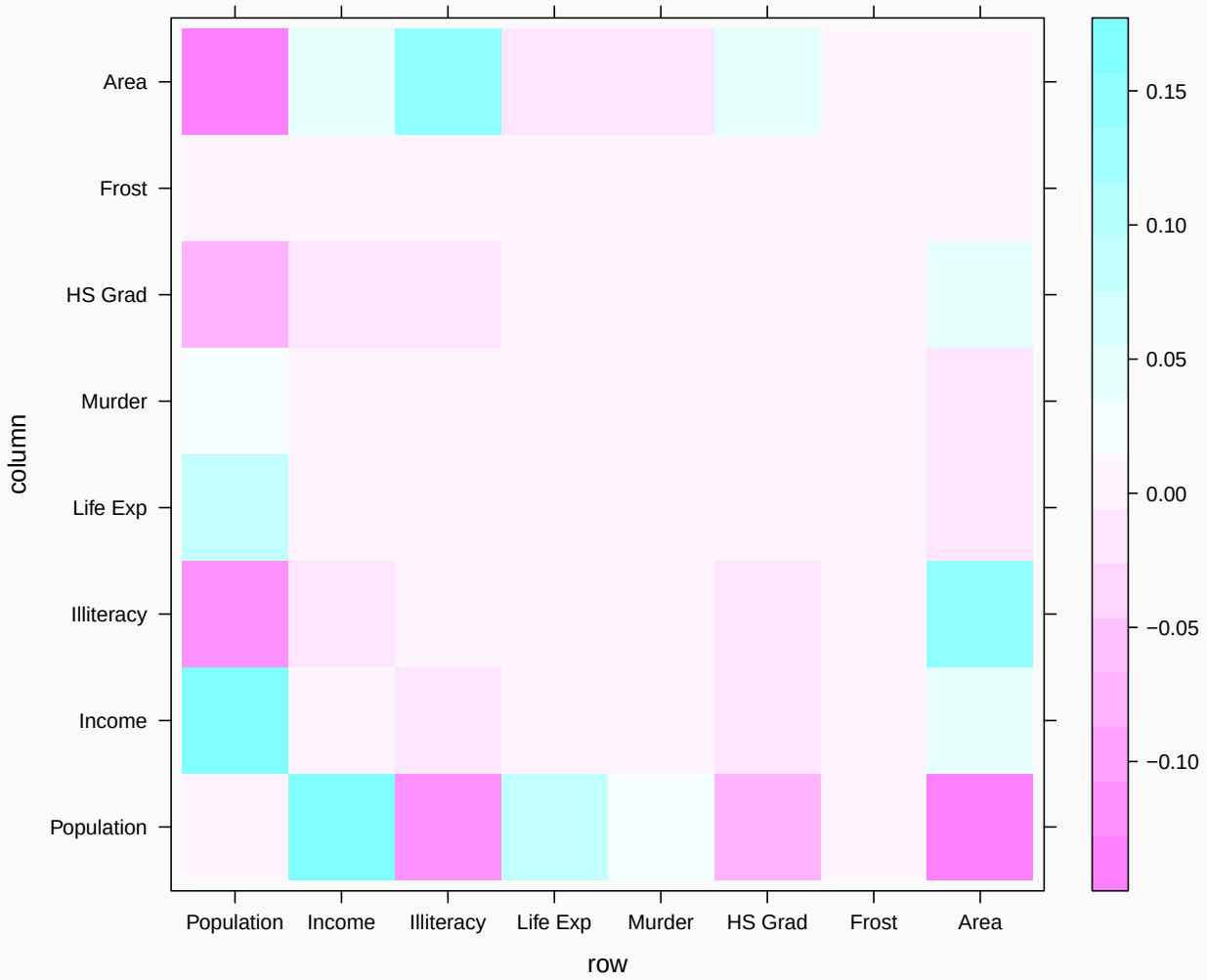
```r
# 2. Maximum Likelihood Factor analysis
fa_decomp <- factanal(data, factors = m)

# Extract estimates
Psi_mle <- fa_decomp$uniquenesses
Lmle <- fa_decomp$loadings

# Our FA model explains:
sum(colSums(Lmle^2))/ncol(data)
```

```
## [1] 0.671468
```

A different example xvii

```
# We can also  visualize the fit
Rn_mle <- tcrossprod(Lmle) + diag(Psi_mle)

levelplot(Rn - Rn_mle)
```

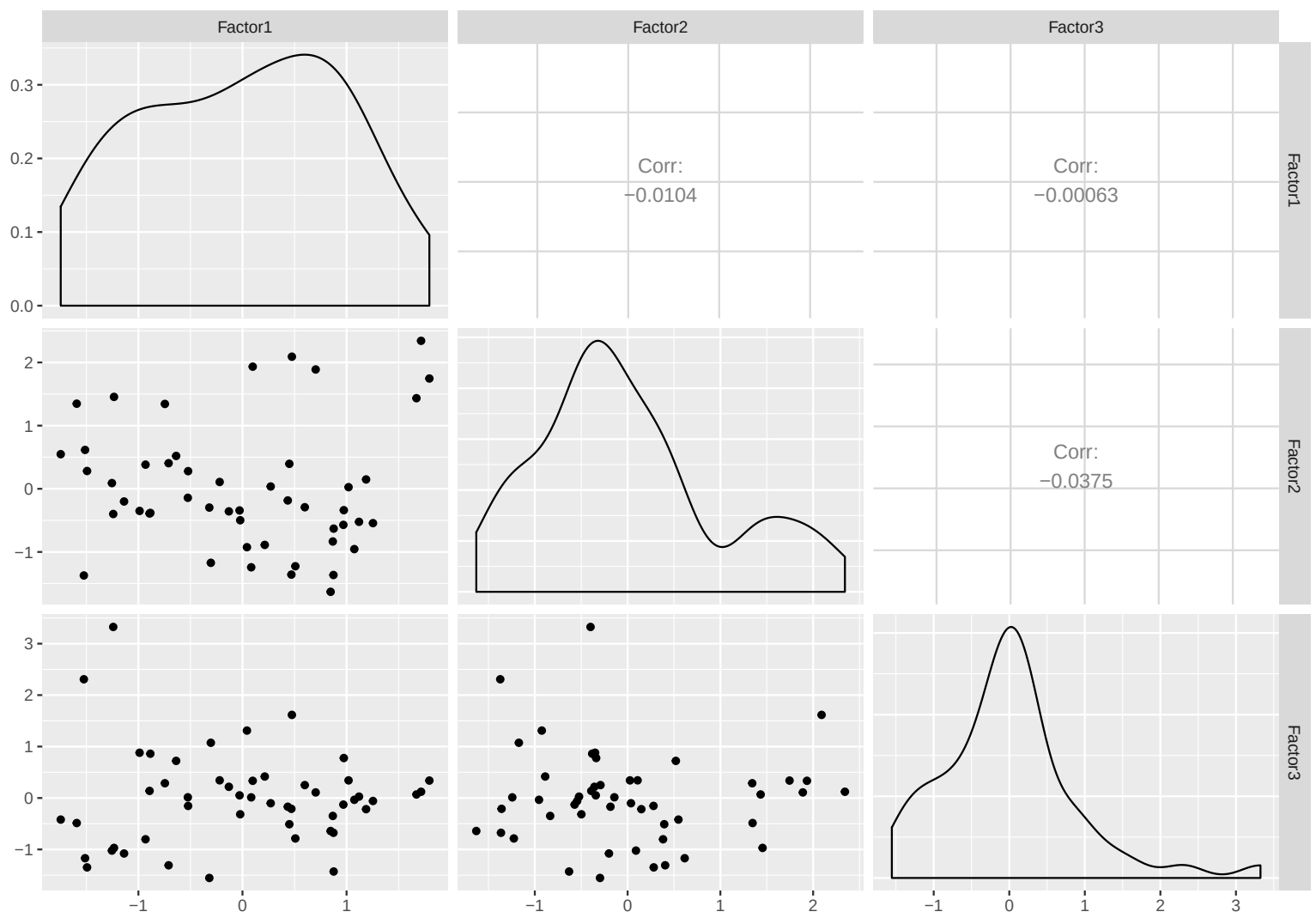
A different example xviii



A different example xix

```
scores_mle <- scale(data, center = TRUE,
                    scale = TRUE) %*%
  solve(Rn) %*% Lmle

ggpairs(as.data.frame(scores_mle))
```

A different example xx



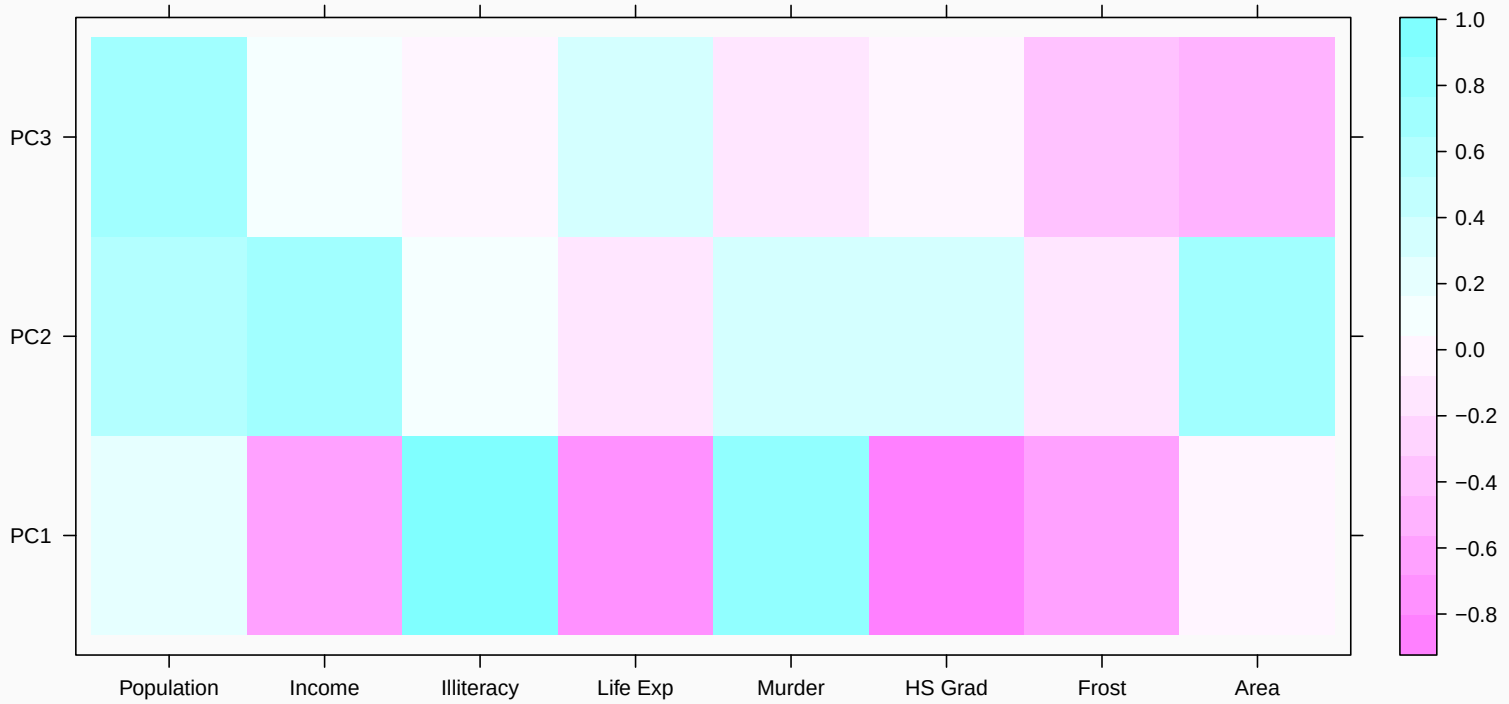
A different example xxi

```
# 3. Compare both loadings
round(cor(scores_pc, scores_mle), 2)
```

```
##     Factor1 Factor2 Factor3
## PC1   -0.76    0.52   -0.41
## PC2   -0.18    0.34    0.80
## PC3    0.37    0.44   -0.19
```

```
levelplot(Lhat,
          xlab = "", ylab = "")
```

A different example xxii



A different example xxiii

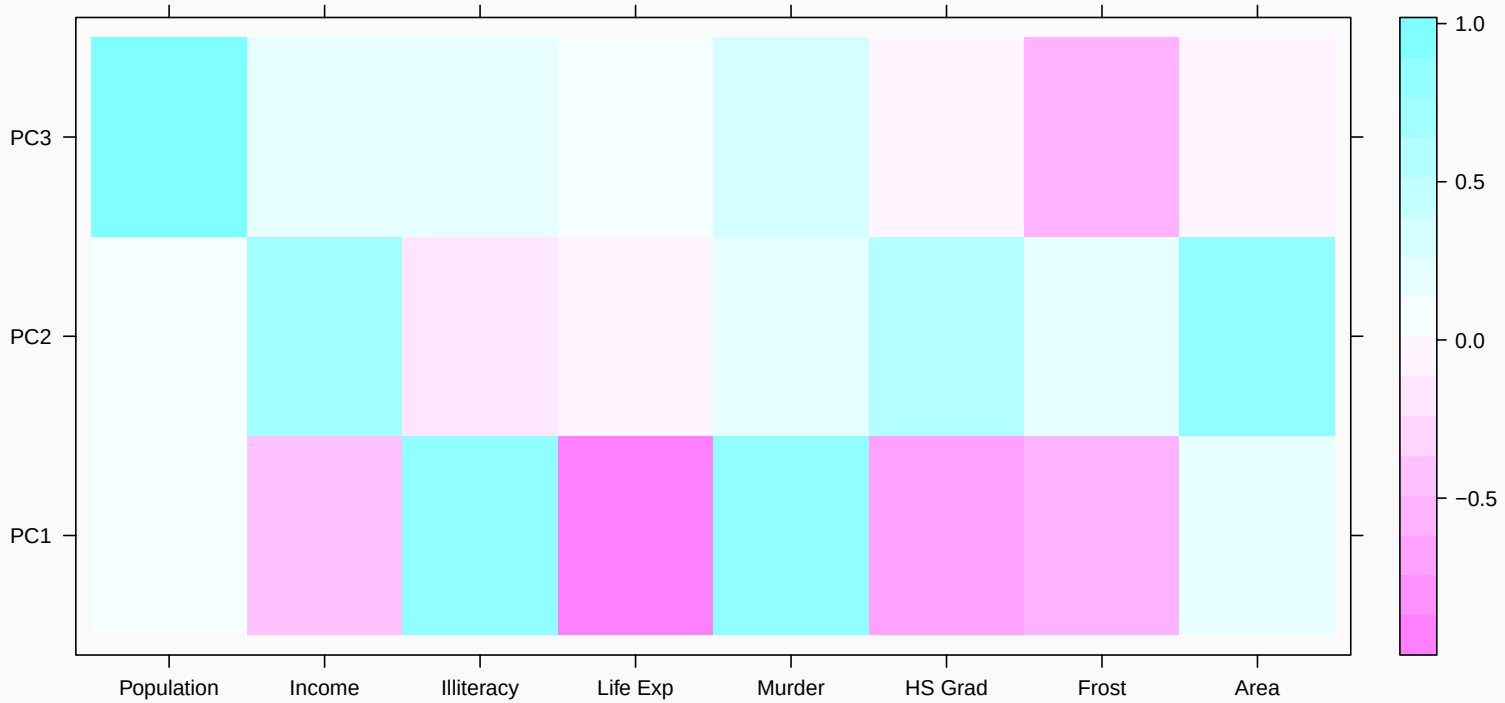
```
# Let's rotate the PC loadings
Lhat <- varimax(Lhat)$loadings
scores_pc <- scale(data, center = TRUE,
                         scale = TRUE) %*%
  solve(Rn) %*% Lhat

# Compare both loadings again
round(cor(scores_pc, scores_mle), 2)
```

A different example xxiv

```
##      Factor1 Factor2 Factor3
## PC1    -0.86    0.36   -0.20
## PC2    -0.09   -0.04    0.89
## PC3     0.02    0.67    0.13
```

```
levelplot(unclass(Lhat),
          xlab = "", ylab = "")
```

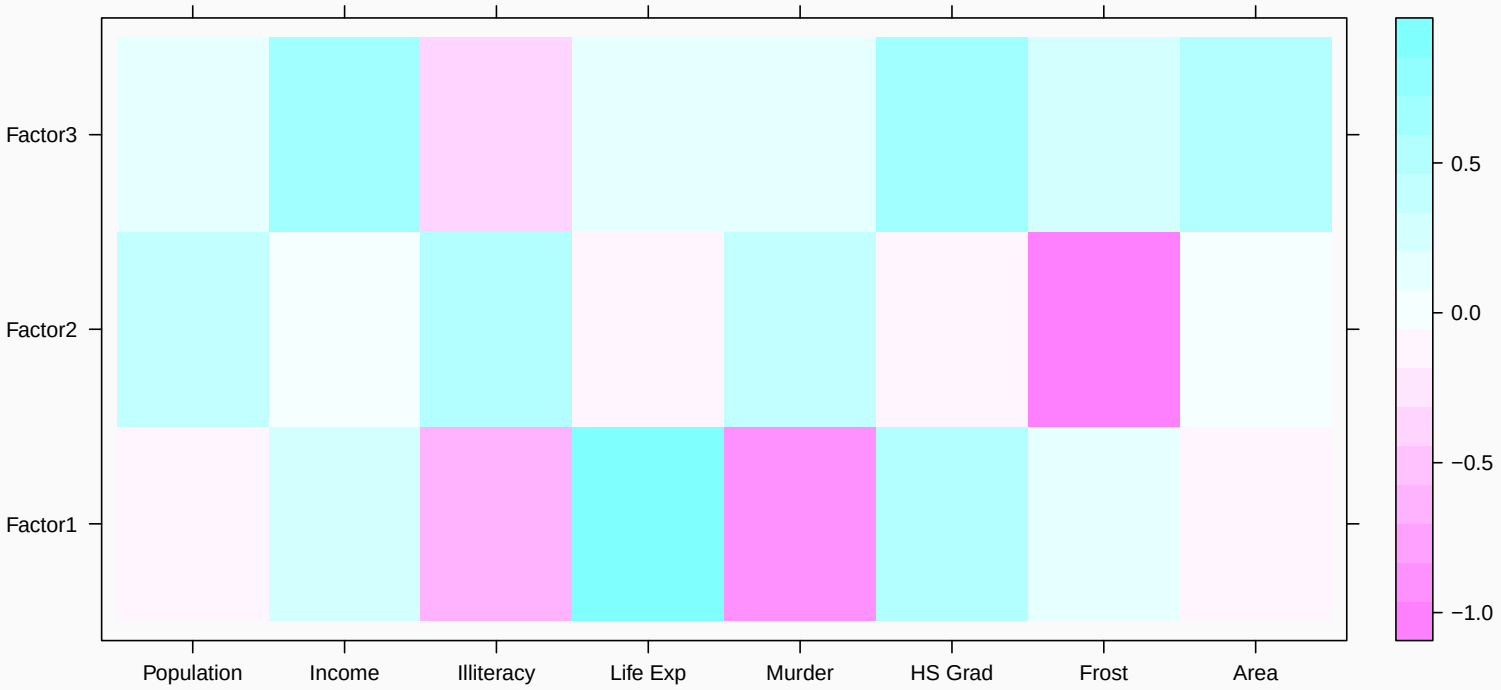
A different example xxv



A different example xxvi

```
levelplot(unclass(Lmle),
          xlab = "", ylab = "")
```

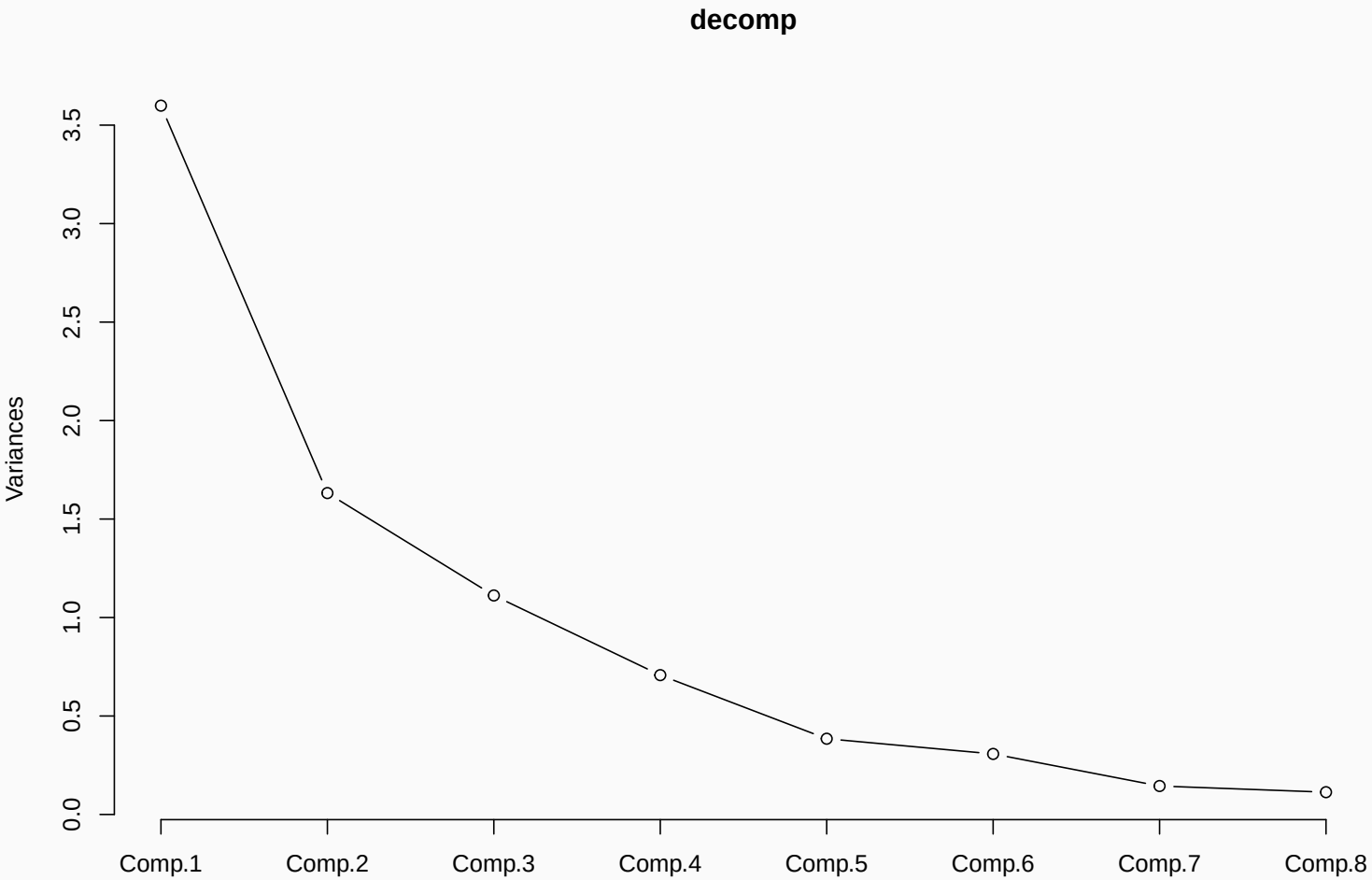
A different example xxvii



A different example xxviii

```r
# 4. Compare different m
# Let's start with scree plot
screeplot(decomp, type = 'l')
```

A different example xxix



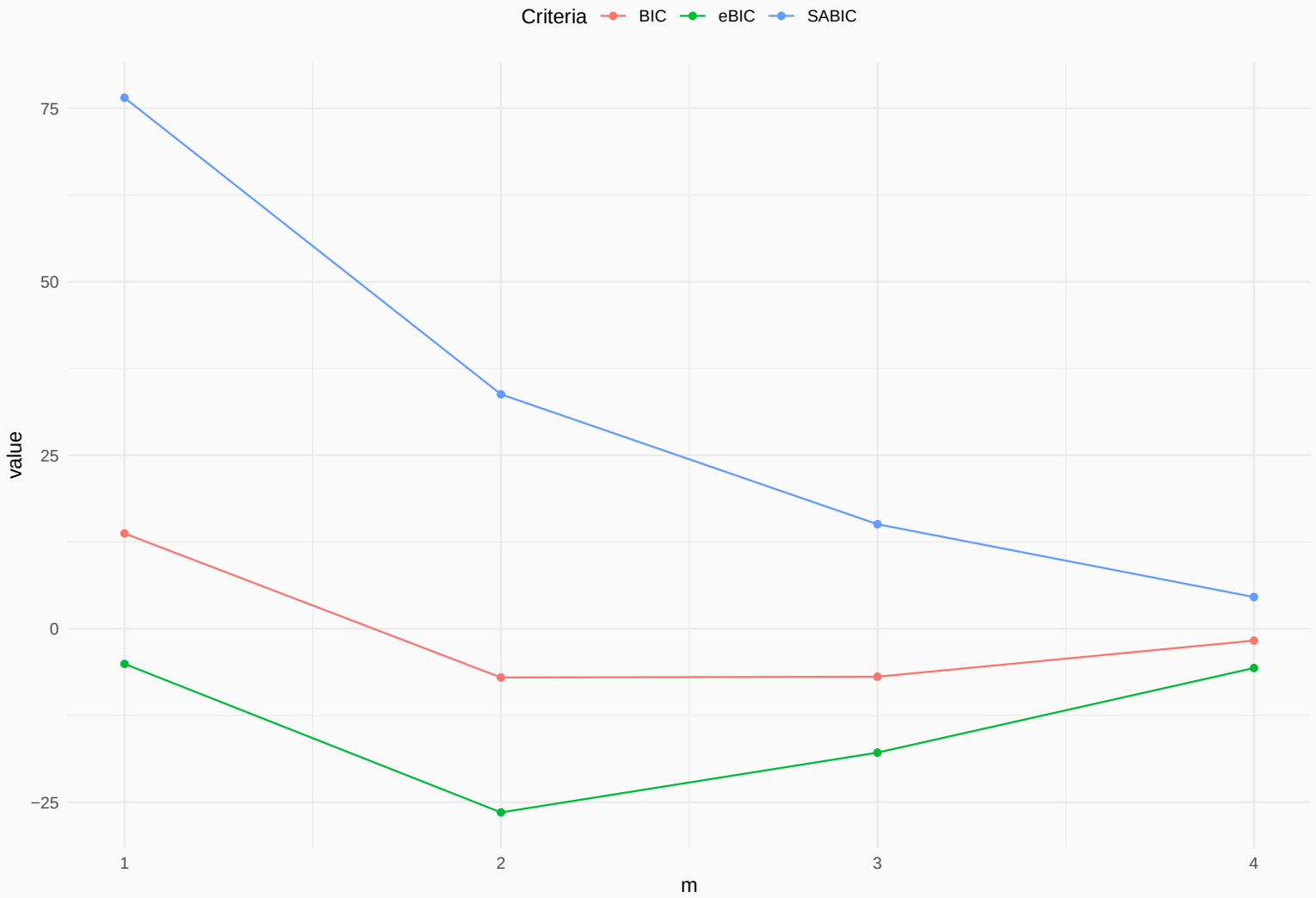
A different example xxx

```
# Then let's look at information criteria
inform_crit <- purrr::map_df(
  seq_len(4),  function(m) {
    fa_decomp <- psych::fa(data, nfactors = m,
                           fm = 'ml')
    data.frame(
      BIC = fa_decomp$BIC, eBIC = fa_decomp$EBIC,
      SABIC = fa_decomp$SABIC, m = m
    )})
```

A different example xxxi

```
inform_crit %>%
  gather(Criteria, value, -m) %>%
  ggplot(aes(m, value, colour = Criteria)) +
  geom_point() +
  geom_line() +
  theme_minimal() +
  theme(legend.position = 'top')
```

A different example xxxii



A different example xxxiii

```
inform_crit %>%
  gather(Criteria, value, -m) %>%
  group_by(Criteria) %>%
  filter(value == min(value)) %>%
  select(-value) %>%
  spread(Criteria, m)
```

```
## # A tibble: 1 x 3
##     BIC  eBIC SABIC
##   <int> <int> <int>
## 1     2     2     4
```

A different example xxxiv

```
m <- 2
Lhat <- decomp$loadings[,seq_len(m)] %*%
  diag(decomp$sdev[seq_len(m)])
colnames(Lhat) <- paste0("PC", 1:2)
Psi_hat <- diag(cov(data) - tcrossprod(Lhat))

Lhat <- varimax(Lhat)$loadings
scores_pc <- scale(data, center = TRUE,
                   scale = TRUE) %*%
  solve(Rn) %*% Lhat
```

A different example xxxv

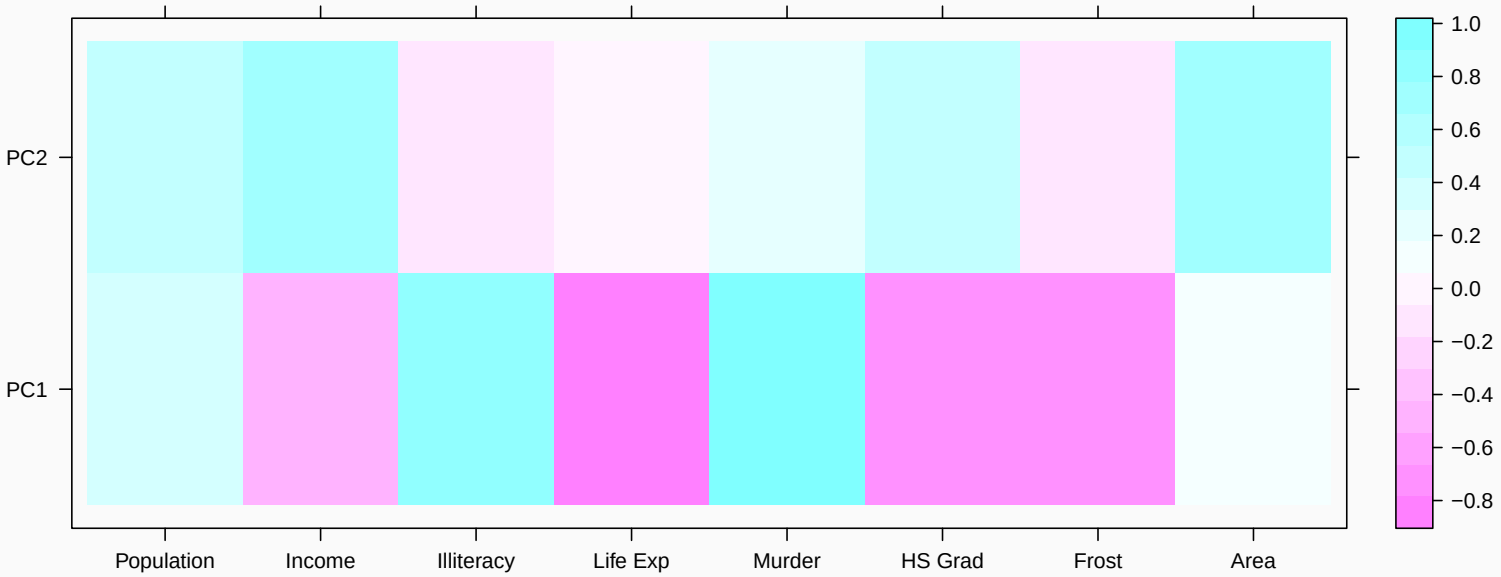
```
# MLE
fa_decomp <- factanal(data, factors = m)
Psi_mle <- fa_decomp$uniquenesses
Lmle <- fa_decomp$loadings

scores_mle <- scale(data, center = TRUE,
                    scale = TRUE) %*%
  solve(Rn) %*% Lmle

# Compare both loadings again
round(cor(scores_pc, scores_mle), 2)
```

A different example xxxvi

```
##     Factor1 Factor2
## PC1    0.91   -0.33
## PC2    0.22    0.84
```

```
levelplot(unclass(Lhat),
          xlab = "", ylab = "")
```

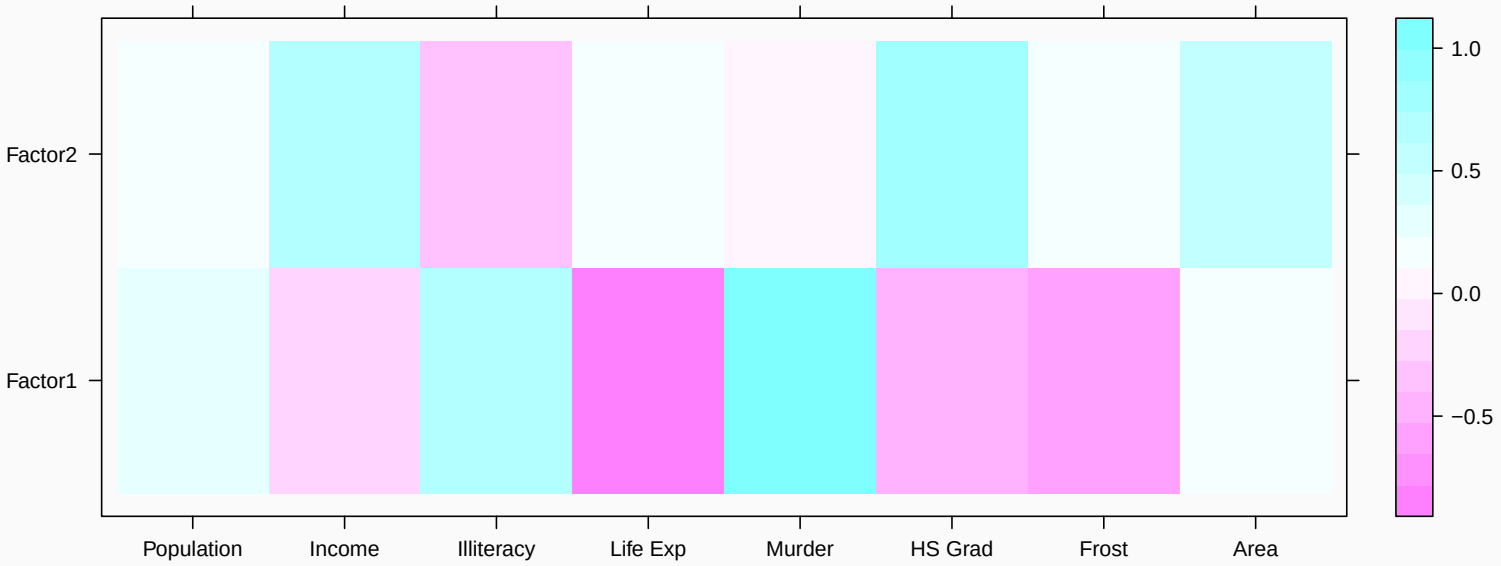
A different example xxxvii



A different example xxxviii

```
levelplot(unclass(Lmle),
          xlab = "", ylab = "")
```

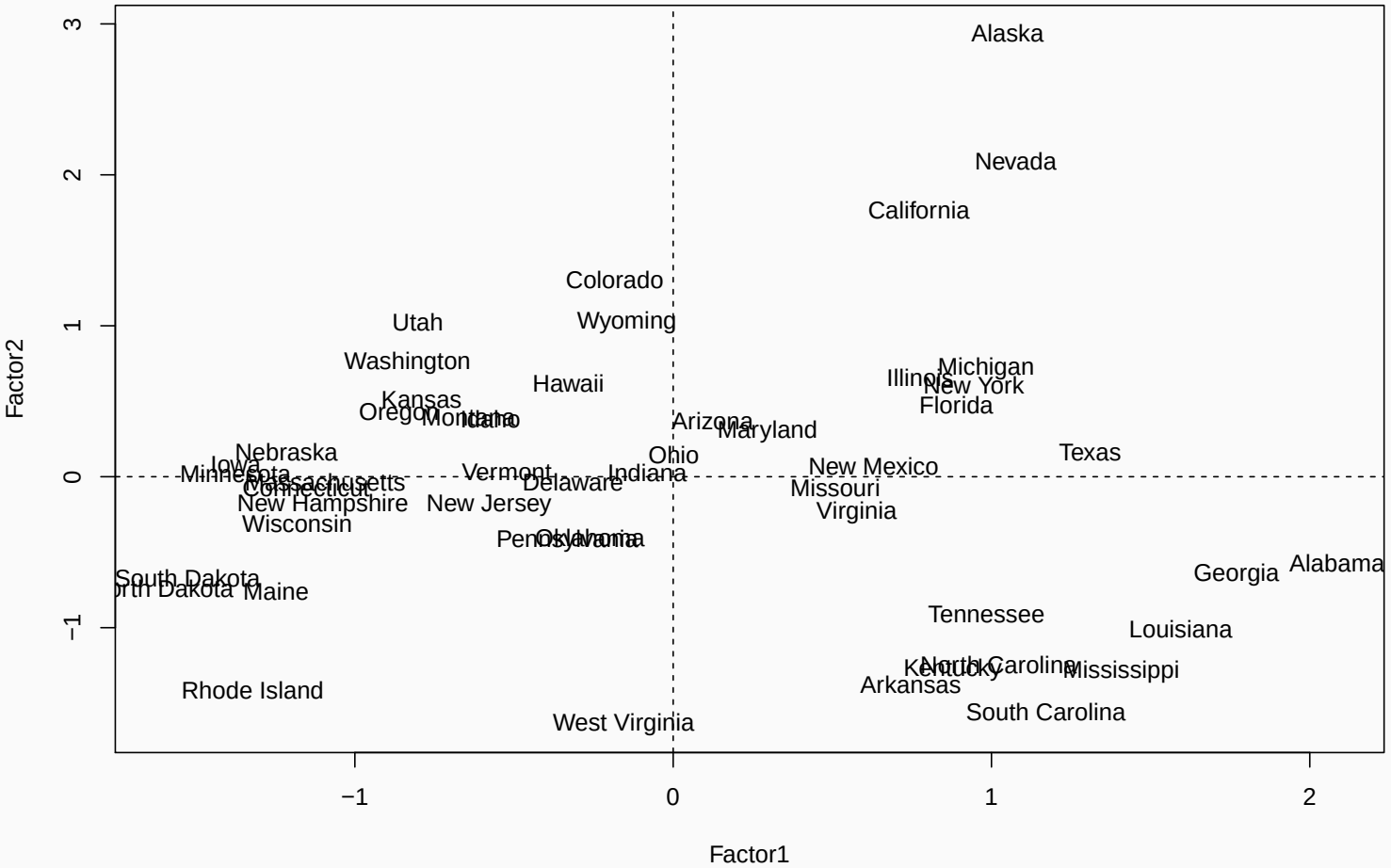
A different example xxxix



A different example xl

```
plot(scores_mle, type = 'n')
text(scores_mle, labels = rownames(scores_mle))
abline(h = 0, v = 0, lty = 2)
```

A different example xli



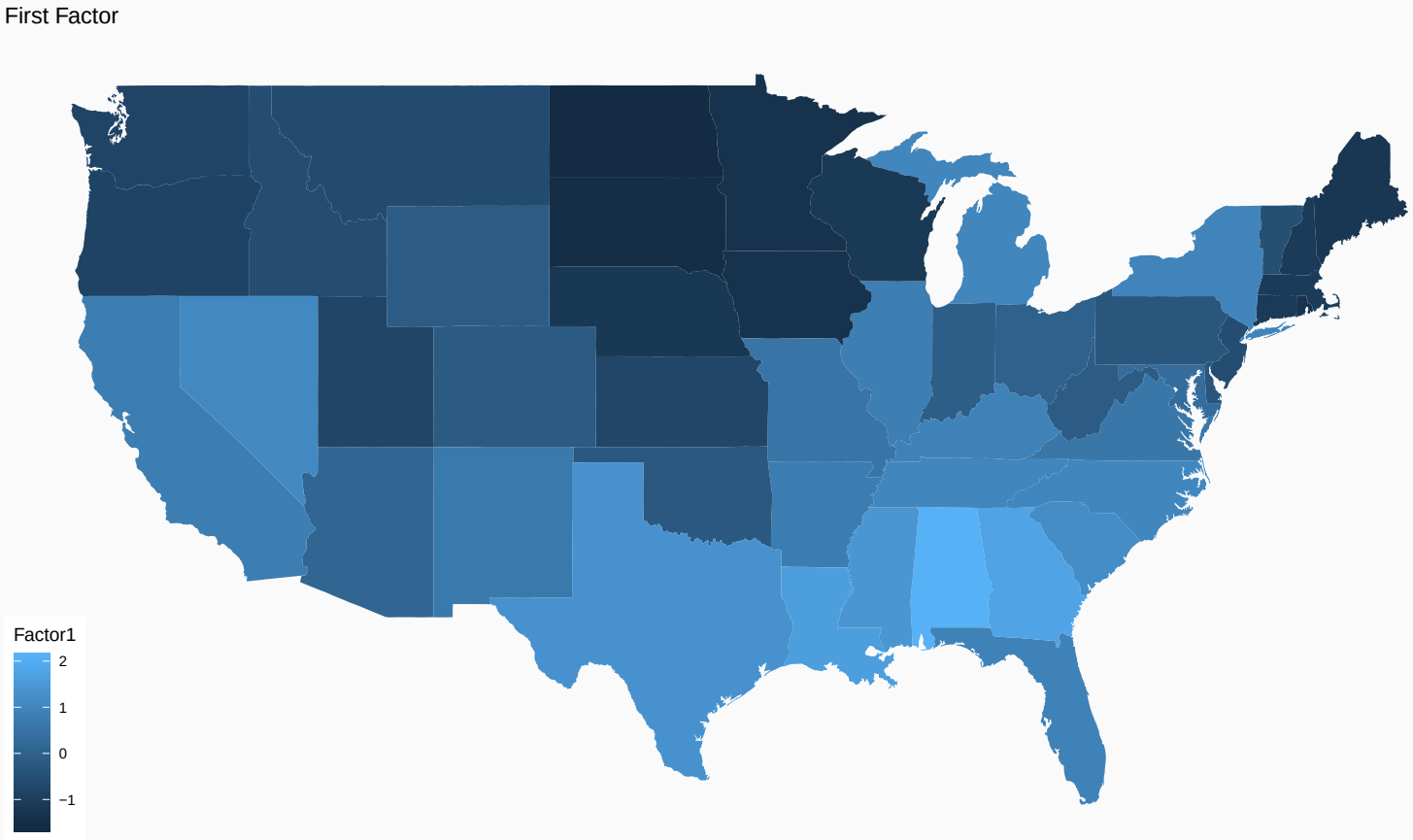
A different example xlii

```
# Let's plot the factors on the map
library(maps)
states <- map_data("state")

data_plot <- scores_mle %>%
  as.data.frame() %>%
  rownames_to_column("region") %>%
  mutate(region = tolower(region)) %>%
  inner_join(states, by = "region")
```

A different example xliii

```
ggplot(data = data_plot) +
  geom_polygon(aes(x = long, y = lat,
                   fill = Factor1,
                   group = group)) +
  coord_fixed(1.3) +
  ggthemes::theme_map() +
  ggtitle("First Factor")
```

A different example xliv



A different example xlv

```
ggplot(data = data_plot) +
  geom_polygon(aes(x = long, y = lat,
                   fill = Factor2,
                   group = group)) +
  coord_fixed(1.3) +
  ggthemes::theme_map() +
  ggtitle("Second Factor")
```

A different example xlvi