Chapter 2: Simulating data for latent variable analysis

William Revelle
Northwestern University
Prepared as part of course on latent variable analysis (Psychology 454)
and as a supplement to the Short Guide to R for psychologists

January 24, 2007

When comparing how well various models fit the data as well as the effect of model fit upon fit statistics, it is useful to know "Truth". With real data this is, of course, not possible. But if we simulate the data then we can examine how well various fitting procedures work in recovering the original data. This chapter is a brief discussion of how to simulate data in R and the use the **sem** package in R to fit it.

We use the **psych** package for descriptive statistics and graphics. As with the other chapters, it is assumed that these packages have been installed (although not necessarily loaded).

2.1 Simulate one latent variable and n observed variables

This is the classic reliability model of congeneric measurement (if $n \geq 4$). To do this we need to generate N subjects with $n+1$ latent variables (one true score for all the items, and n error scores, one for each item). We use the **rnorm** function to generate random normal deviates with mean of zero and standard deviation of 1.

2.1.1 Generate data that fit a congeneric model

First some preliminaries: specify the number of subjects, the number of variables, the true score loadings for the variables, and the form the pattern matrix of true score loadings and error loadings. We then calculate the population covariance matrix (which is equal to the population correlation matrix in this example because we are constraining the sum of the squared true and error weights to be 1).

Before starting we set the "random number generator" seed to a memorable constant value so that the simulations done by the reader will match the text.

```
> library(psych)
> set.seed(42)
> N <- 1000
> n <- 4
> loading <- matrix(c(0.8, 0.7, 0.6, 0.5), nrow = 4)
> error <- diag(1, nrow = 4)
> diag(error) <- sqrt(1 - loading^2)
> pattern <- cbind(loading, error)
> colnames(pattern) <- c("theta", paste("e", seq(1:n), sep = ""))
> rownames(pattern) <- c(paste("V", seq(1:n), sep = ""))
> round(pattern, 2)
```

```
   theta  e1   e2  e3   e4
V1   0.8 0.6 0.00 0.0 0.00
V2   0.7 0.0 0.71 0.0 0.00
V3   0.6 0.0 0.00 0.8 0.00
V4   0.5 0.0 0.00 0.0 0.87
```

```
> round(pattern %*% t(pattern), 2)
```

```
     V1   V2   V3   V4
V1 1.00 0.56 0.48 0.40
V2 0.56 1.00 0.42 0.35
V3 0.48 0.42 1.00 0.30
V4 0.40 0.35 0.30 1.00
```

To generate the latent scores for all the subjects and find the ${}_{N}observed_{n}$ scores we first generate a N x n+1 matrix of randomly distributed normal latent scores using the **rnorm** function. Then we take the product of the ${}_{N}latent_{n+1}$ matrix by the transposed ${}_{n+1}npattern_{n}$ matrix. We do this twice to demonstrate the effect of different samples.

```
> latent <- matrix(rnorm(N * (n + 1)), ncol = (n + 1))
> observed <- latent %*% t(pattern)
> pairs.panels(observed)
```

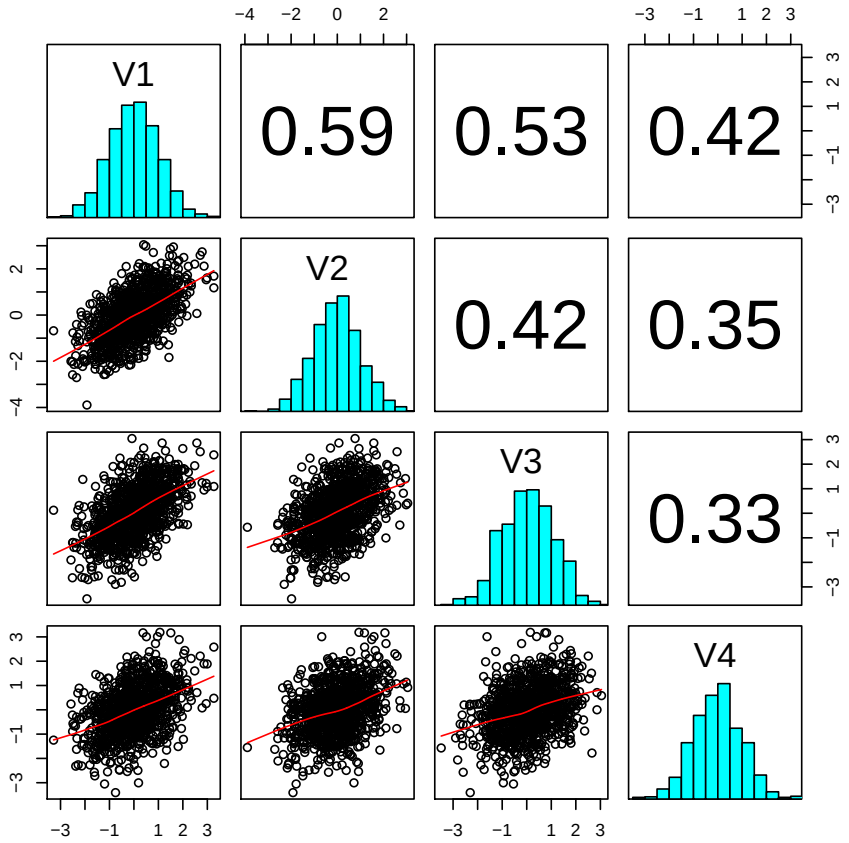

Figure 2.1: SPLOM plot and correlations of 4 congeneric measures, sample size = 1000

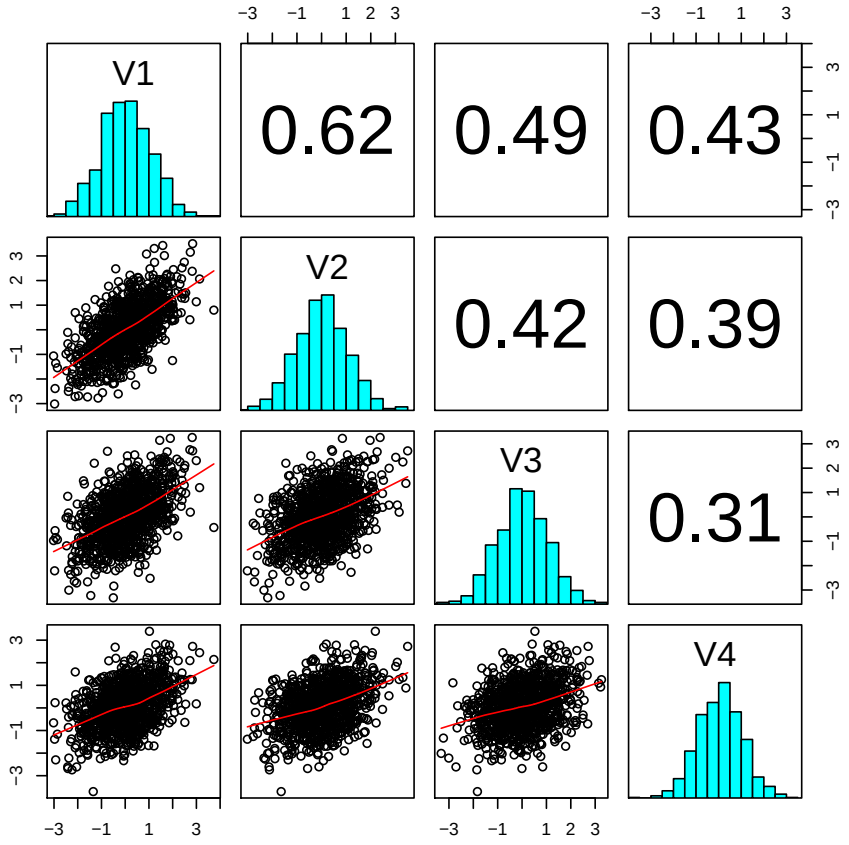

Figure 2.2: SPLOM plot of a new sample of 1000 subjects for 4 congeneric measures

2.1.2 Another simulation will vary from the first

Repeat the simulation to show the variation between two samples. This is the power of simulations for it allows us to see how well various procedures recover the population values given the simulated samples.

```
> latent <- matrix(rnorm(N * (n + 1)), ncol = (n + 1))
> observed <- latent %*% t(pattern)
> pairs.panels(observed)
```

2.1.3 Create a function to generate the data

In order to do simulations with different parameter values and different sample sizes we can create a new function. This will make the simulation operation more efficient and able to be done in a more complicated situation. Giving the function a name that reflects what it does, congeneric, is helpful. In addition, we can give it some default values so that if we do not specify sample size or the latent loadings, the function will use those default values.

```
> congeneric <- function(N = 1000, loads = c(0.8, 0.7, 0.6, 0.5)) {
+     n <- length(loads)
+     loading <- matrix(loads, nrow = n)
+     error <- diag(1, nrow = n)
+     diag(error) <- sqrt(1 - loading^2)
+     pattern <- cbind(loading, error)
+     colnames(pattern) <- c("theta", paste("e", seq(1:n), sep = ""))
+     rownames(pattern) <- c(paste("V", seq(1:n), sep = ""))
+     latent <- matrix(rnorm(N * (n + 1)), ncol = (n + 1))
+     observed <- latent %*% t(pattern)
+     return(observed)
+ }
```

We can use this function to produce a new (and larger) data set with different loadings than the previous two. Rather than show the scatter plot, we use the **cor** and **round** functions.

```
> observed <- congeneric(200, loads = c(0.8, 0.8, 0.8, 0.6, 0.6, 0.6))
> round(cor(observed), 2)
```

```
     V1   V2   V3   V4   V5   V6
V1 1.00 0.67 0.70 0.54 0.55 0.50
V2 0.67 1.00 0.73 0.50 0.56 0.48
V3 0.70 0.73 1.00 0.53 0.58 0.51
V4 0.54 0.50 0.53 1.00 0.49 0.33
V5 0.55 0.56 0.58 0.49 1.00 0.41
V6 0.50 0.48 0.51 0.33 0.41 1.00
```

2.2 SEM for one sample from the population

A congeneric reliability model is one of most simple of all latent variable models. For there is one latent variable and we just need to estimate the paths from the latent to the observed variable.

The **sem** package in R provides a basic structural equation analysis package for single groups. Help in using the package is available from the help menu (? sem). For a detailed discussion of the program, see J. Fox (2006), "Structural Equation Modeling With the sem Package in R." Structural Equation Modeling, 13:465-486. The package needs to be installed (downloaded from Cran) before you can use it. It also needs to be active.

It is helpful in understanding the goodness of fit test produced by **sem** to compare multiple samples from the same population. We use the congeneric function defined above to to generate a sample from the population, do a sem analysis, and then take another sample and do it again.

First find the covariance matrix, S, from the raw data. Then specify various model parameters, and then run the program. We reset the random seed to a particular value so that we wil get reproducible examples.

It is helpful to show the model being fit (model.congeneric) to check that the input specification is correct. In later examples using the same model, this is not necessary.

```
> set.seed(42)
> library(sem)
> observed <- congeneric()
> S.congeneric <- cov(observed)
> model.congeneric <- matrix(c("theta ->  V1", "a", NA, "theta -> V2", "b", NA,
 "theta -> V3", "c",  NA, "theta -> V4", "d", NA, "V1 <-> V1", "u", NA,
 "V2 <-> V2", "v", NA, "V3 <-> V3", "w", NA,
  "V4 <-> V4", "x", NA, "theta <-> theta", NA, 1), ncol = 3, byrow = TRUE)
> colnames(model.congeneric) <- c("path", "coefficient", "start value")
> model.congeneric
```

```
      path              coefficient start value
 [1,] "theta ->  V1"    "a"         NA
 [2,] "theta -> V2"     "b"         NA
 [3,] "theta -> V3"     "c"         NA
 [4,] "theta -> V4"     "d"         NA
 [5,] "V1 <-> V1"       "u"         NA
 [6,] "V2 <-> V2"       "v"         NA
 [7,] "V3 <-> V3"       "w"         NA
 [8,] "V4 <-> V4"       "x"         NA
 [9,] "theta <-> theta" NA          "1"
```

```
> sem.congeneric = sem(model.congeneric, S.congeneric, 1000)
> summary(sem.congeneric, digits = 3)
```

```
 Model Chisquare =  0.46   Df =  2 Pr(>Chisq) = 0.795
 Chisquare (null model) =  910   Df =  6
 Goodness-of-fit index =  1
 Adjusted goodness-of-fit index =  0.999
 RMSEA index =  0   90% CI: (NA, 0.0398)
 Bentler-Bonnett NFI =  1
 Tucker-Lewis NNFI =  1.01
 Bentler CFI =  1
 BIC =  -13.4

 Normalized Residuals
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.177000 -0.032200 -0.000271  0.010600  0.017000  0.319000

 Parameter Estimates
  Estimate Std Error z value Pr(>|z|)
a 0.829    0.0320    25.90   0        V1 <--- theta
b 0.657    0.0325    20.23   0        V2 <--- theta
c 0.632    0.0325    19.43   0        V3 <--- theta
d 0.503    0.0340    14.80   0        V4 <--- theta
u 0.316    0.0346     9.12   0        V1 <--> V1
```

```
v 0.580    0.0334    17.35   0        V2 <--> V2
w 0.604    0.0337    17.94   0        V3 <--> V3
x 0.776    0.0382    20.31   0        V4 <--> V4

 Iterations =  13
```

Take another sample of the same size, 1000, from the same population and do another sem. The model matrix is the same for this analysis. Compare the two sets of coefficients, as well as the fit statistics.

```
> N <- 1000
> observed <- congeneric()
> S.congeneric <- cov(observed)
> sem.congeneric = sem(model.congeneric, S.congeneric, N)
> summary(sem.congeneric, digits = 3)

 Model Chisquare =  1.07   Df =  2 Pr(>Chisq) = 0.586
 Chisquare (null model) =  1015   Df =  6
 Goodness-of-fit index =  1
 Adjusted goodness-of-fit index =  0.997
 RMSEA index =  0   90% CI: (NA, 0.0522)
 Bentler-Bonnett NFI =  0.999
 Tucker-Lewis NNFI =  1.00
 Bentler CFI =  1
 BIC =  -12.7

 Normalized Residuals
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.2150 -0.0945 -0.0106  0.0206  0.0211  0.5370

 Parameter Estimates
  Estimate Std Error z value Pr(>|z|)
a 0.858    0.0311    27.58   0        V1 <--- theta
b 0.713    0.0324    22.03   0        V2 <--- theta
c 0.655    0.0334    19.64   0        V3 <--- theta
d 0.509    0.0331    15.37   0        V4 <--- theta
u 0.284    0.0330     8.62   0        V1 <--> V1
v 0.558    0.0332    16.81   0        V2 <--> V2
w 0.668    0.0359    18.61   0        V3 <--> V3
x 0.754    0.0369    20.46   0        V4 <--> V4

 Iterations =  13
```

2.2.1 A smaller sample

Take another sample from the population, but this time, a much smaller one. Once again, the model is the same model. Notice how the parameter estimates and goodness of fit tests

are very different from the previous two analyses.

```
> N <- 100
> observed <- congeneric()
> S.congeneric <- cov(observed)
> model.congeneric
```

```
      path              coefficient start value
 [1,] "theta ->  V1"    "a"         NA
 [2,] "theta -> V2"     "b"         NA
 [3,] "theta -> V3"     "c"         NA
 [4,] "theta -> V4"     "d"         NA
 [5,] "V1 <-> V1"       "u"         NA
 [6,] "V2 <-> V2"       "v"         NA
 [7,] "V3 <-> V3"       "w"         NA
 [8,] "V4 <-> V4"       "x"         NA
 [9,] "theta <-> theta" NA          "1"
```

```
> sem.congeneric = sem(model.congeneric, S.congeneric, N)
> summary(sem.congeneric, digits = 3)
```

```
 Model Chisquare =  0.0186   Df =  2 Pr(>Chisq) = 0.99
 Chisquare (null model) =  89.8   Df =  6
 Goodness-of-fit index =  1
 Adjusted goodness-of-fit index =  1
 RMSEA index =  0   90% CI: (NA, NA)
 Bentler-Bonnett NFI =  1
 Tucker-Lewis NNFI =  1.07
 Bentler CFI =  1
 BIC =  -9.2

 Normalized Residuals
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.03300 -0.00785  0.00212  0.00307  0.00983  0.06500

 Parameter Estimates
  Estimate Std Error z value Pr(>|z|)
a 0.808    0.102     7.91    2.66e-15 V1 <--- theta
b 0.739    0.105     7.04    1.99e-12 V2 <--- theta
c 0.591    0.104     5.69    1.24e-08 V3 <--- theta
d 0.475    0.105     4.52    6.20e-06 V4 <--- theta
u 0.356    0.109     3.28    1.05e-03 V1 <--> V1
v 0.513    0.110     4.66    3.17e-06 V2 <--> V2
w 0.650    0.109     5.95    2.75e-09 V3 <--> V3
x 0.741    0.115     6.44    1.23e-10 V4 <--> V4

 Iterations =  13
```

2.2.2 Multiple simulations - varying sample size

To examine the sensitivity of the parameter estimates and the fit statistics to sample size, we run a small simulation for 4 sample sizes with 20 replications at each sample size. After each simulation and sem analysis, we take the summary statistics from the analysis and save them in a list (in this case, results.list). After the 80 simulations we then organize the results in a manner that allows us to graph the results. Figure 2.3 shows the range of parameter estimates across the 80 replications.

```
> S.N <- c(100, 200, 400, 800)
> S.iterations <- 20
> case <- 1
> results.list <- list()
> model.congeneric <- matrix(c("theta ->  V1", "a", NA, "theta -> V2", "b", NA,
"theta -> V3", "c",   NA, "theta -> V4", "d", NA, "V1 <-> V1", "u", NA,
"V2 <-> V2", "v", NA, "V3 <-> V3", "w", NA,
 "V4 <-> V4", "x", NA, "theta <-> theta", NA, 1), ncol = 3, byrow = TRUE)
> for (i in 1:length(S.N)) {
+     N <- S.N[i]
+     for (j in 1:S.iterations) {
+         observed <- congeneric()
+         S.congeneric <- cov(observed)
+         sem.congeneric = sem(model.congeneric, S.congeneric, N)
+         ss <- summary(sem.congeneric, digits = 3)
+         results.list[[case]] <- list(N = N, chisq = ss$chisq, GFI = ss$GFI,
             AGFI = ss$AGFI, RMSES = ss$RMSEA,
             NFI = ss$NFI, CFI = ss$CFI, BIC = ss$BIC, loadings = ss$coeff$Estimate)
+         case <- case + 1
+     }
+ }
> results.mat <- unlist(results.list)
> results.mat <- matrix(unlist(results.list), byrow = TRUE,
   nrow = S.iterations * length(S.N))
> colnames(results.mat) <- c("N", "chisq", "GFI", "AGFI", "RMSEA",
"RMSEAl", "RMSEAh", "RMSEA9", "NFI",
+     "CFI", "BIC", "a", "b", "c", "d", "e", "f", "g", "h")
> results.df <- data.frame(results.mat)
> boxplot(results.df[, 12:15], main = "parameter estimates")
```

Figure 2.3 showed how the parameter estimates vary from sample to sample. Although the central tendency of each estimate is roughly the correct parameter value, the range of the 20 samples differs as a function of sample size, particularly for the smaller parameter values. This is shown by creating boxplots for each of 4 sample sizes. We organize these results by sample size using a loop. (figure 2.4)

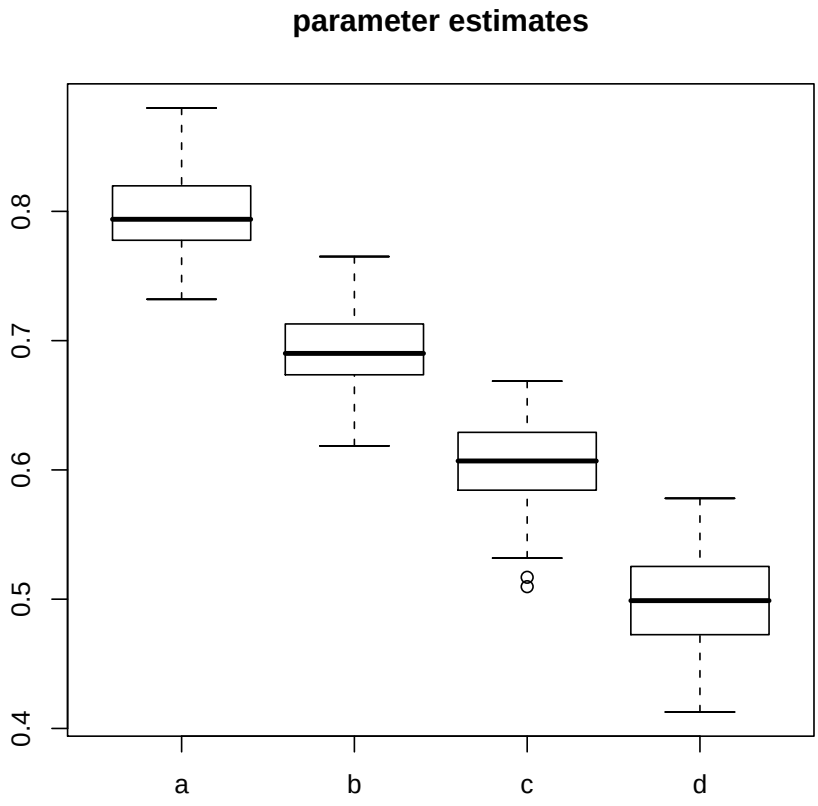

Figure 2.3: Variation in parameter estimates for 80 samples of varying sample size. Population values are .8, .6, .5, .4

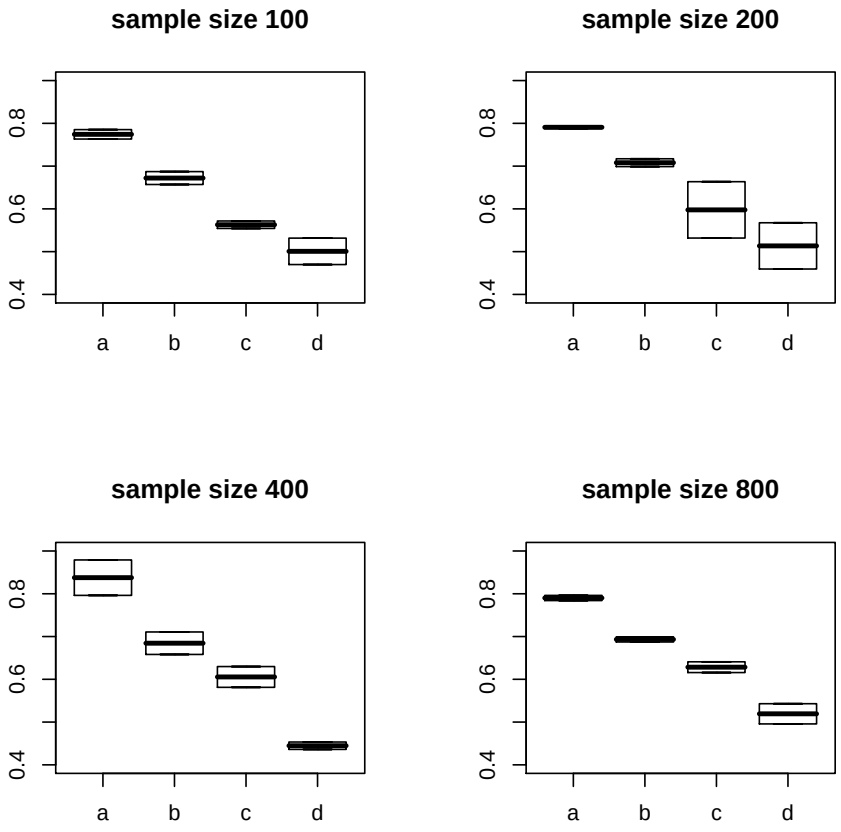

Figure 2.4: Parameter estimates as a function of sample size. N replications/sample size = 20. Population values are .8, .6, .5, .4

```
> par(mfrow = c(2, 2))
> for (i in 1:4) boxplot(results.df[c((i - 1) * S.iterations + 1, i * S.iterations),
       12:15], main = paste("sample size",  results.df[i * S.iterations, 1]),
       ylim = c(0.4, 0.9))
> par(mfrow = c(1, 1))
```

The previous figures (2.3, 2.4 showed how the parameter estimates vary from sample to sample and as as function of sample size. We can also examine how much the goodness of fit tests vary as a function of sample size by plotting sample size against the goodness of fit tests. We first remove various statistics from the data matrix to make the figure more manageable.

```
> small.results <- results.df[, -c(12:19, 6:8)]
> pairs.panels(small.results)
```

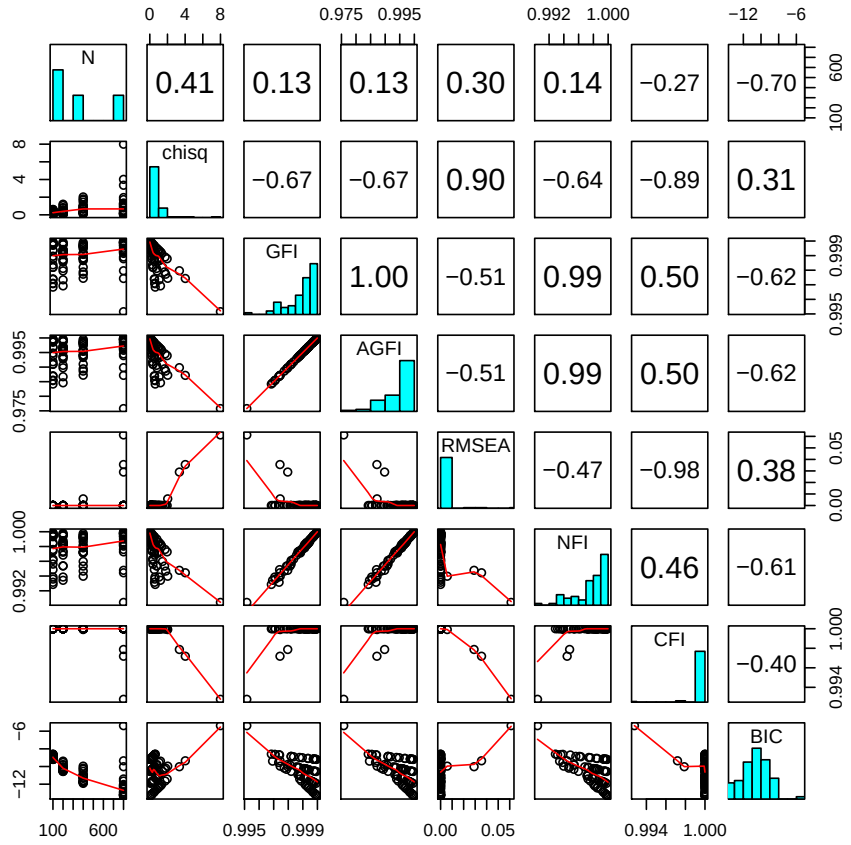

Figure 2.5: SEM goodness of fit tests for 20 replications at each of 4 sample sizes

2.3 Statistics for a more complicated model

The next data model is more complex and represents two correlated factors with six variables. The model can be correctly or incorrectly specified. An incorrect specification would try to fit one factor to the data, a better model would fit two correlated factors. The congeneric function can be modified to allow for two correlated latent variables. (See the next chapter for a more general simulation function for many latent variables.)

The factor structure can be varied by passing a parameter (loads) to the function. The default value of the loads matrix is a two factor loading matrix with loadings on the first factor of .8, .7 and .6 for the first three variables and of .7, .6 and .5 for the last three variables on the second factor. The default correlation of the factors is set to .4, but an be varied as a parameter. Note that the phi matrix is not symmetric but rather reflects a structure where the first factor leads to the second factor.

After creating the function we set the random number generator seed to a fixed value so that the output from the reader will match this example.

```
> twofactor <- function(N = 1000, n = 6, loads = c(0.8, 0.7, 0.6, 0, 0, 0,
                                                   0, 0, 0, 0.7, 0.6, 0.5),
+     phi12 = 0.4) {
+     loading <- matrix(loads, nrow = n)
+     error <- diag(1, nrow = n)
+     error <- diag(1, nrow = n)
+     diag(error) <- 1 - diag(loading %*% t(loading))
+     pattern <- cbind(loading, error)
+     phi <- diag(1, nrow = n + 2)
+     phi[2, 1] <- phi12
+     phi[2, 2] <- sqrt(1 - phi12^2)
+     colnames(pattern) <- c("theta1", "theta2", paste("e", seq(1:n), sep = ""))
+     rownames(pattern) <- c(paste("V", seq(1:n), sep = ""))
+     latent <- matrix(rnorm(N * (n + 2)), ncol = (n + 2))
+     observed <- latent %*% phi %*% t(pattern)
+     return(observed)
+ }
> set.seed(17)
> f2 <- twofactor()
> pairs.panels(f2)
```

2.3.1 Fitting a 1 factor model to two factor data

We can try to fit this model inappropriately, by thinking of it as a one factor model, or we can fit the appropriate correlated two factor solution. First try fitting a one factor model to the data.

We can see that the residuals of (the data - the model) suggest that the solution requires two factors.

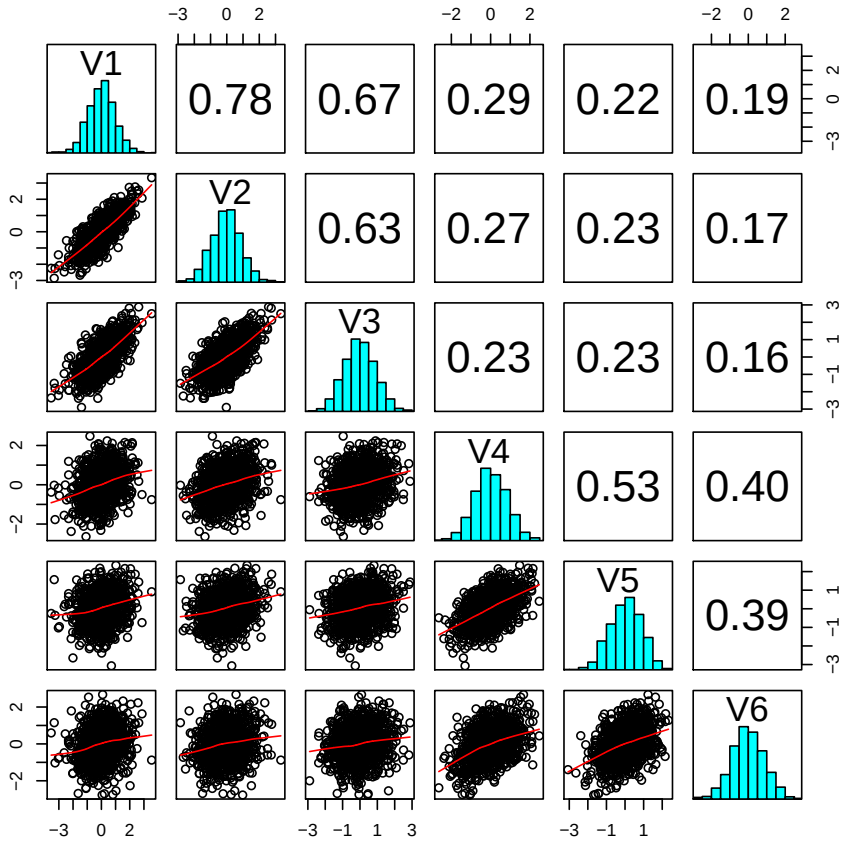

Figure 2.6: SPLOM and correlations for 6 observed scores loading on two correlated factors

```
> S.f2 <- cov(f2)
> model.f1 <- matrix(c("theta ->  V1", "a", NA, "theta -> V2", "b", NA,
      "theta -> V3", "c", NA, "theta -> V4",
+     "d", NA, "theta -> V5", "e", NA, "theta -> V6", "f", NA, "V1 <-> V1", "u", NA,
       "V2 <-> V2", "v",    NA, "V3 <-> V3", "w", NA, "V4 <-> V4", "x", NA,
        "V5 <-> V5", "y", NA, "V6 <-> V6", "z", NA, "theta <-> theta",
+     NA, 1), ncol = 3, byrow = TRUE)
> colnames(model.f1) <- c("path", "coefficient", "start value")
> model.f1
```

```
      path              coefficient start value
 [1,] "theta ->  V1"    "a"         NA
 [2,] "theta -> V2"     "b"         NA
 [3,] "theta -> V3"     "c"         NA
 [4,] "theta -> V4"     "d"         NA
 [5,] "theta -> V5"     "e"         NA
 [6,] "theta -> V6"     "f"         NA
 [7,] "V1 <-> V1"       "u"         NA
 [8,] "V2 <-> V2"       "v"         NA
 [9,] "V3 <-> V3"       "w"         NA
[10,] "V4 <-> V4"       "x"         NA
[11,] "V5 <-> V5"       "y"         NA
[12,] "V6 <-> V6"       "z"         NA
[13,] "theta <-> theta" NA          "1"
```

```
> sem.f1 = sem(model.f1, S.f2, N)
> summary(sem.f1, digits = 2)
```

```
 Model Chisquare =  369   Df =  9 Pr(>Chisq) = 0
 Chisquare (null model) =  1806   Df =  15
 Goodness-of-fit index =  0.86
 Adjusted goodness-of-fit index =  0.67
 RMSEA index =  0.22   90% CI: (0.20, 0.24)
 Bentler-Bonnett NFI =  0.8
 Tucker-Lewis NNFI =  0.66
 Bentler CFI =  0.8
 BIC =  309

 Normalized Residuals
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -1.120  -0.645  -0.018   1.390   0.143  12.200

 Parameter Estimates
  Estimate Std Error z value Pr(>|z|)
a 0.85     0.028     30.6    0.0e+00  V1 <--- theta
b 0.78     0.027     28.9    0.0e+00  V2 <--- theta
c 0.66     0.028     23.3    0.0e+00  V3 <--- theta
d 0.29     0.030      9.5    0.0e+00  V4 <--- theta
```

```
e 0.25     0.031      8.0    1.1e-15  V5 <--- theta
f 0.21     0.033      6.4    1.6e-10  V6 <--- theta
u 0.17     0.019      9.2    0.0e+00  V1 <--> V1
v 0.21     0.018     12.1    0.0e+00  V2 <--> V2
w 0.37     0.022     16.9    0.0e+00  V3 <--> V3
x 0.61     0.031     19.6    0.0e+00  V4 <--> V4
y 0.64     0.033     19.7    0.0e+00  V5 <--> V5
z 0.74     0.037     19.8    0.0e+00  V6 <--> V6

 Iterations =  15
```

```
> round(residuals(sem.f1), 2)
```

```
      V1    V2    V3    V4    V5    V6
V1  0.00  0.01  0.00 -0.02 -0.03 -0.02
V2  0.01  0.00  0.00 -0.02 -0.02 -0.03
V3  0.00  0.00  0.00 -0.02  0.01 -0.01
V4 -0.02 -0.02 -0.02  0.00  0.30  0.24
V5 -0.03 -0.02  0.01  0.30  0.00  0.24
V6 -0.02 -0.03 -0.01  0.24  0.24  0.00
```

2.3.2 Fitting a 2 factor model to two factor data

This leads us to try a different model, the two factor model. The residuals for this model are much smaller and the goodness of fit tests suggests a good solution.

```
> model.f2 <- matrix(c("theta1 ->  V1", "a", NA, "theta1 -> V2", "b", NA,
              "theta1 -> V3", "c", NA,
+     "theta2 -> V4", "d", NA, "theta2 -> V5", "e", NA, "theta2 -> V6", "f", NA,
   "V1 <-> V1", "u",
+     NA, "V2 <-> V2", "v", NA, "V3 <-> V3", "w", NA, "V4 <-> V4", "x", NA,
    "V5 <-> V5", "y", NA, "V6 <-> V6",
+     "z", NA, "theta1 <-> theta1", NA, 1, "theta2 <-> theta2", NA, 1,
     "theta1 <-> theta2", "g", NA),
+     ncol = 3, byrow = TRUE)
> colnames(model.f2) <- c("path", "coefficient", "start value")
> model.f2
```

```
      path                coefficient start value
 [1,] "theta1 ->  V1"     "a"         NA
 [2,] "theta1 -> V2"      "b"         NA
 [3,] "theta1 -> V3"      "c"         NA
 [4,] "theta2 -> V4"      "d"         NA
 [5,] "theta2 -> V5"      "e"         NA
 [6,] "theta2 -> V6"      "f"         NA
 [7,] "V1 <-> V1"         "u"         NA
 [8,] "V2 <-> V2"         "v"         NA
 [9,] "V3 <-> V3"         "w"         NA
```

```
[10,] "V4 <-> V4"         "x"         NA
[11,] "V5 <-> V5"         "y"         NA
[12,] "V6 <-> V6"         "z"         NA
[13,] "theta1 <-> theta1" NA          "1"
[14,] "theta2 <-> theta2" NA          "1"
[15,] "theta1 <-> theta2" "g"         NA
```

```
> sem.f2 = sem(model.f2, S.f2, N)
> summary(sem.f2, digits = 2)
```

```
 Model Chisquare =  6.6   Df =  8 Pr(>Chisq) = 0.58
 Chisquare (null model) =  1806   Df =  15
 Goodness-of-fit index =  1
 Adjusted goodness-of-fit index =  1
 RMSEA index =  0   90% CI: (NA, 0.036)
 Bentler-Bonnett NFI =  1
 Tucker-Lewis NNFI =  1
 Bentler CFI =  1
 BIC =  -47

 Normalized Residuals
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-7.8e-01 -7.0e-02 -1.9e-06  1.1e-02  2.1e-01  5.9e-01

 Parameter Estimates
  Estimate Std Error z value Pr(>|z|)
a 0.85     0.028     30.6    0.0e+00  V1 <--- theta1
b 0.78     0.027     28.6    0.0e+00  V2 <--- theta1
c 0.66     0.028     23.3    0.0e+00  V3 <--- theta1
d 0.63     0.033     19.2    0.0e+00  V4 <--- theta2
e 0.59     0.033     17.9    0.0e+00  V5 <--- theta2
f 0.48     0.034     13.9    0.0e+00  V6 <--- theta2
u 0.16     0.020      8.2    2.2e-16  V1 <--> V1
v 0.21     0.019     11.4    0.0e+00  V2 <--> V2
w 0.37     0.022     16.9    0.0e+00  V3 <--> V3
x 0.29     0.031      9.3    0.0e+00  V4 <--> V4
y 0.36     0.030     11.9    0.0e+00  V5 <--> V5
z 0.56     0.033     17.0    0.0e+00  V6 <--> V6
g 0.40     0.038     10.4    0.0e+00  theta2 <--> theta1

 Iterations =  25
```

```
> round(residuals(sem.f2), 2)
```

```
      V1    V2   V3   V4    V5    V6
V1  0.00  0.00 0.00 0.01 -0.02  0.00
V2  0.00  0.00 0.00 0.01 -0.01 -0.01
V3  0.00  0.00 0.00 0.01  0.02  0.00
V4  0.01  0.01 0.01 0.00  0.00  0.00
```

```
V5 -0.02 -0.01 0.02 0.00  0.00  0.01
V6  0.00 -0.01 0.00 0.00  0.01  0.00
```

2.3.3 A set of simulations for the two factor model

Once again, we can examine the effect of sample size on our solution by running multiple simulations using various sample sizes. Unfortunately, this does not produce stable results in all cases when N=100 and so we do the simulations for N varing from 200 ... 1600. Even so, this takes several runs to get a stable set of solutions.

```
> set.seed(17)
> model.f2 <- matrix(c("theta1 ->  V1", "a", NA, "theta1 -> V2", "b", NA,
     "theta1 -> V3", "c", NA,    "theta2 -> V4", "d", NA, "theta2 -> V5", "e", NA,
      "theta2 -> V6", "f", NA, "V1 <-> V1", "u",   NA, "V2 <-> V2", "v", NA,
      "V3 <-> V3", "w", NA, "V4 <-> V4", "x", NA, "V5 <-> V5", "y", NA, "V6 <-> V6",
+     "z", NA, "theta1 <-> theta1", NA, 1, "theta2 <-> theta2", NA, 1,
     "theta1 <-> theta2", "g", NA),
+     ncol = 3, byrow = TRUE)
> colnames(model.f2) <- c("path", "coefficient", "start value")
> model.f2
```

```
      path                coefficient start value
 [1,] "theta1 ->  V1"     "a"         NA
 [2,] "theta1 -> V2"      "b"         NA
 [3,] "theta1 -> V3"      "c"         NA
 [4,] "theta2 -> V4"      "d"         NA
 [5,] "theta2 -> V5"      "e"         NA
 [6,] "theta2 -> V6"      "f"         NA
 [7,] "V1 <-> V1"         "u"         NA
 [8,] "V2 <-> V2"         "v"         NA
 [9,] "V3 <-> V3"         "w"         NA
[10,] "V4 <-> V4"         "x"         NA
[11,] "V5 <-> V5"         "y"         NA
[12,] "V6 <-> V6"         "z"         NA
[13,] "theta1 <-> theta1" NA          "1"
[14,] "theta2 <-> theta2" NA          "1"
[15,] "theta1 <-> theta2" "g"         NA
```

```
> results.list <- list()
> S.N <- c(200, 400, 800, 1600)
> S.iterations <- 12
> case <- 1
>  for (i in 1:length(S.N)) {
 + for (j in 1:S.iterations) {

+         N <- S.N[i]
+         observed <- twofactor(N)
```

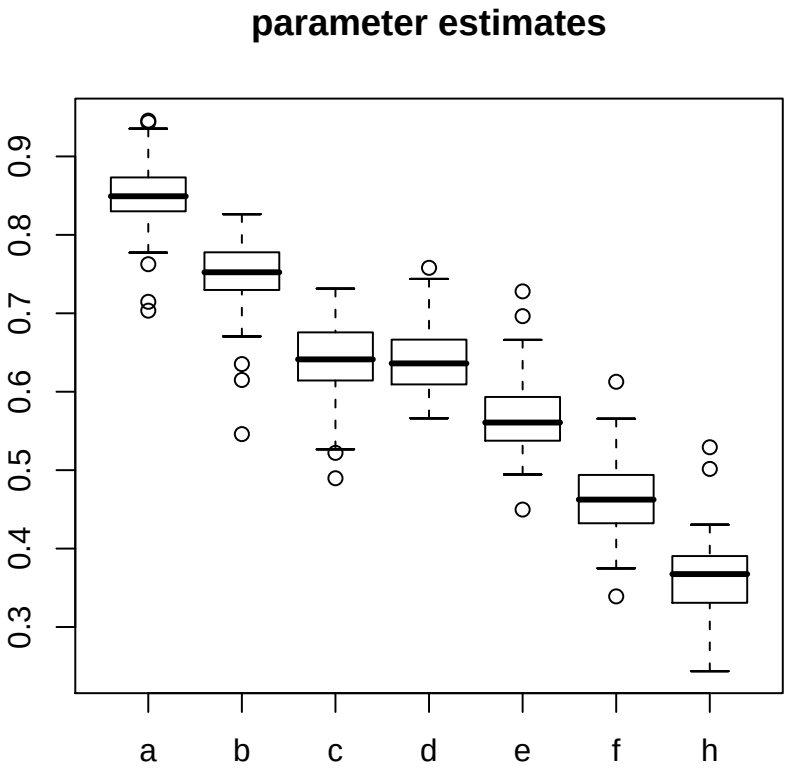

Figure 2.7: Estimated loadings on factor 1 (a-c), factor 2 (d-f) and path coefficient between factors (h) for 12 replications of 4 sample sizes.

```
+         S.f2 <- cov(observed)
+         sem.f2 = sem(model.f2, S.f2, N)
+         ss <- summary(sem.f2, digits = 3)
+         results.list[[case]] <- list(N = N, chisq = ss$chisq, GFI = ss$GFI,
           AGFI = ss$AGFI, RMSES = ss$RMSEA,
+            NFI = ss$NFI, CFI = ss$CFI, BIC = ss$BIC, loadings = ss$coeff$Estimate)
+         case <- case + 1
+     }
+ }
```

First we show the parameter estimates from all 48 simulations (Figure 2.7 and then examine the effects of sample size (Figure 2.8).

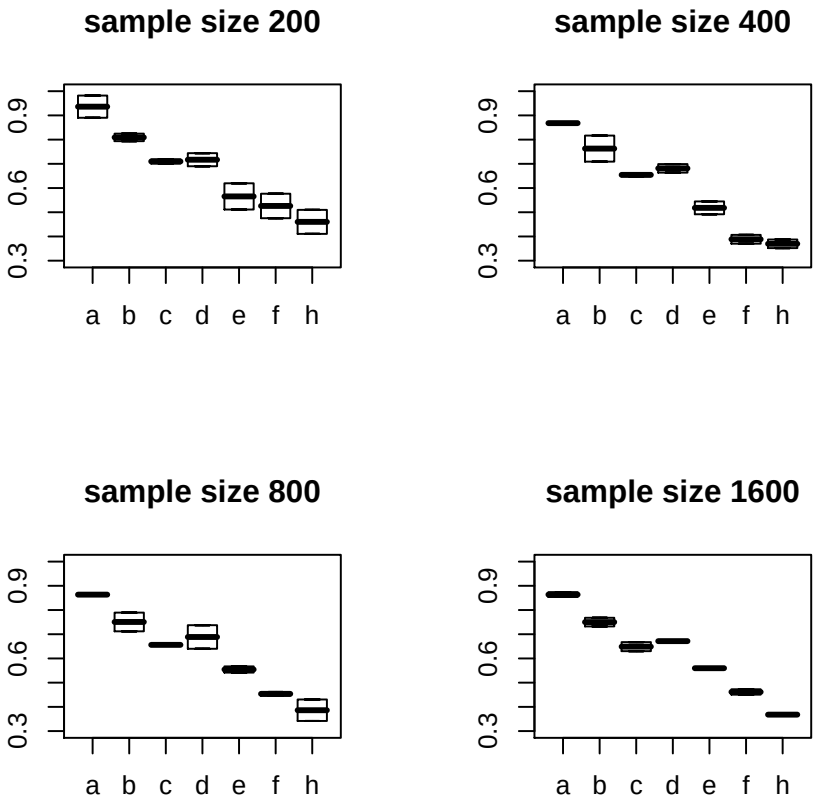

Figure 2.8: Variability in estimated parameters as a function of sample size