Model Searches

KMG & STL

May 22, 2019

Set up Environment

```
library(mvtnorm)
library(gimme)
```

```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
## Registered S3 methods overwritten by 'huge':
##   method    from
##   plot.sim  BDgraph
##   print.sim BDgraph
```

```
library(graphicalVAR)
library(perturbR)
```

Data generating function

```
genData <- function(A = NULL,
                    Phi = NULL,
                    obs = 200,
                    n   = 10 ){
  data2ls <- list()
  for (p in 1:n)
  {

    negA1 <- diag(5)-A

    time  <- matrix(0, nrow = 5, ncol = obs+400)

    time1 <- matrix(0, nrow = 5, ncol = obs+400)

    noise <- solve(negA1, matrix(rnorm(5*(obs+400),0,1), nrow = 5, ncol = obs+400))

    time[,1] <- noise[,1]

    time1[,1] <- solve(negA1, Phi) %*% time[,1] + noise[,1]

    time[,2]  <- time1[,1]
```

```r
    for (i in 2:(obs+400)){
      time1[,i]  <- solve(negA1, Phi) %*% time[,(i)] + noise[,i]
      if (i<(obs+400))
        time[,(i+1)] <- time1[,i]
    }

    data2ls[[p]] <- t(time[,400:600])
    names(data2ls)[p]<- paste0('ind', p)

  }
  return(gen_data = data2ls)
}
```

Data Generation

Now we'll generate data where half of the individuals have one pattern of relations and the other half have a couple of differences in their patterns.

Specifically, pattern A1 has variable 1 predict variable 4 contemporaneously and A2 variable 3 predict variable 5 contemporoeously. All other relations are the same between the two, with the exception that variable 5 predicted by 4 is positive for on subset of the data (A1) and negative for the other (see A2).

```r
# Generate first pattern #
A1 <- matrix(
    c(0, 0, 0, 0,  0,
      .7,  0, 0, 0,  0,
      0, .7, 0, 0,  0,
      .7,  0, 0, 0, 0,
      0,  0, 0, .7,  0), nrow = 5, ncol = 5, byrow = TRUE)

  Phi1 <- matrix(
    c(.5,   0, 0,  0,   0,
      0,  .5, 0,  0,   0,
      0,   0, .5,  0,   0,
      0,   0, 0, .5,   0,
      0,   0, 0,  0,  .5), nrow = 5, ncol = 5, byrow = TRUE)

Data1 <- genData(A = A1, Phi = Phi1, obs = 200)

# Generate second pattern #
A2 <- matrix(
  c(0, 0, 0, 0,  0,
    .7,  0, 0, 0,  0,
    0, .7, 0, 0,  0,
    0,  0, 0, 0, 0,
    0,  0, .7, -.7,  0), nrow = 5, ncol = 5, byrow = TRUE)

Phi2 <- matrix(
  c(.5,   0, 0,  0,   0,
    0,  .5, 0,  0,   0,
    0,   0, .5,  0,   0,
    0,   0, 0, .5,   0,
    0,   0, 0,  0,  .5), nrow = 5, ncol = 5, byrow = TRUE)
```

```r
Data2 <- genData(A = A2, Phi = Phi2, obs = 200)

# combine data into one list #
Data_all <- append(Data1, Data2)
# make sure each individual has a unique name
names(Data_all) <- paste0('ind', seq(1,length(Data_all)))
```

Investigate modification indices (MIs)

Let's select one individual and see what happens when we start with a null model wherein only the autoregressive effects are estimated.

Do the MIs that are significant make sense?

```r
# Take one participant and time-embed the data
data_test <- Data_all[[1]]
data_test2 <- embed(data_test,2)
# The first variables are time at zero lag, the second set are the
# lag-1 variables. Rearrange.
data_test3 <- matrix(,200,10)
data_test3[,1:5] <- data_test2[,6:10]
data_test3[,6:10] <- data_test2[,1:5]

# This model only has AR effects
modelAR <- '
V1 ~ 0*V6
V2 ~ 0*V7
V3 ~ 0*V8
V4 ~ 0*V9
V5 ~ 0*V10
V6 ~ V1
V7 ~ V2
V8 ~ V3
V9 ~ V4
V10 ~ V5

'
# lavaan will only give MIs for variables that are dependent variables
# in an equation. So we put nonsense paths with "0*" (Stephanie Lane convention).

fit <- lavaan::sem(modelAR, data_test3)

# Get MIs
mi <- lavaan::modindices(fit)
mi[mi$op == "~",][41:80,] # select those where time is predictede (remove time-1)
```

```
##     lhs op rhs     mi   epc sepc.lv sepc.all sepc.nox
## 106  V6  ~  V2  0.271 0.020   0.020    0.031    0.031
## 107  V6  ~  V3  0.373 0.017   0.017    0.037    0.037
## 108  V6  ~  V4  0.006 0.003   0.003    0.005    0.005
## 109  V6  ~  V5  0.910 0.032   0.032    0.057    0.057
## 110  V6  ~  V7 38.010 0.240   0.240    0.371    0.371
## 111  V6  ~  V8 12.127 0.096   0.096    0.210    0.210
## 112  V6  ~  V9 37.267 0.276   0.276    0.368    0.368
## 113  V6  ~ V10 13.538 0.122   0.122    0.222    0.222
## 114  V7  ~  V1  4.281 0.160   0.160    0.103    0.103
## 115  V7  ~  V3  0.014 0.004   0.004    0.006    0.006
## 116  V7  ~  V4  0.591 0.045   0.045    0.038    0.038
## 117  V7  ~  V5  1.684 0.055   0.055    0.065    0.065
## 118  V7  ~  V6 65.355 0.624   0.624    0.403    0.403
## 119  V7  ~  V8 25.863 0.180   0.180    0.254    0.254
## 120  V7  ~  V9 22.520 0.275   0.275    0.237    0.237
## 121  V7  ~ V10  9.904 0.133   0.133    0.157    0.157
## 122  V8  ~  V1 16.415 0.358   0.358    0.164    0.164
## 123  V8  ~  V2  9.981 0.180   0.180    0.128    0.128
## 124  V8  ~  V4  5.875 0.161   0.161    0.098    0.098
## 125  V8  ~  V5  2.923 0.083   0.083    0.069    0.069
## 126  V8  ~  V6 42.910 0.578   0.578    0.266    0.266
## 127  V8  ~  V7 70.123 0.478   0.478    0.340    0.340
## 128  V8  ~  V9 26.197 0.340   0.340    0.208    0.208
## 129  V8  ~ V10 14.370 0.184   0.184    0.154    0.154
## 130  V9  ~  V1  6.013 0.185   0.185    0.138    0.138
## 131  V9  ~  V2  1.799 0.065   0.065    0.076    0.076
## 132  V9  ~  V3  0.975 0.034   0.034    0.056    0.056
## 133  V9  ~  V5  0.822 0.037   0.037    0.051    0.051
## 134  V9  ~  V6 60.090 0.582   0.582    0.437    0.437
## 135  V9  ~  V7 22.430 0.230   0.230    0.267    0.267
## 136  V9  ~  V8 11.820 0.119   0.119    0.194    0.194
## 137  V9  ~ V10 44.493 0.275   0.275    0.376    0.376
## 138 V10  ~  V1 10.766 0.277   0.277    0.151    0.151
## 139 V10  ~  V2  4.462 0.115   0.115    0.097    0.097
## 140 V10  ~  V3  2.905 0.066   0.066    0.079    0.079
## 141 V10  ~  V4  4.357 0.132   0.132    0.096    0.096
## 142 V10  ~  V6 31.114 0.469   0.469    0.257    0.257
## 143 V10  ~  V7 14.716 0.209   0.209    0.177    0.177
## 144 V10  ~  V8 12.885 0.139   0.139    0.166    0.166
## 145 V10  ~  V9 73.529 0.542   0.542    0.396    0.396
```

The MIs point us to paths that, if added, would significantly improve the model fit. We see here that the MIs true (i.e., data-generating) paths tend to be high. For instance, V10~V9 is variable 5 at time regressed on variable 4 at time, which is in both groups' models.

Sequentially adding paths

One way to build a model is to start by adding the path with the highest MI for that individual, re-estimating this model to get new MIs, selecting the next path to add from these MIs, and so on until the model is considered a good fit.

This is what the function "indSEM" does in the *gimmme* R package. It does individual-level model searches in this manner and then provides summative results (as well as individual-level estimates).

Let's explore.

Results from offering no information

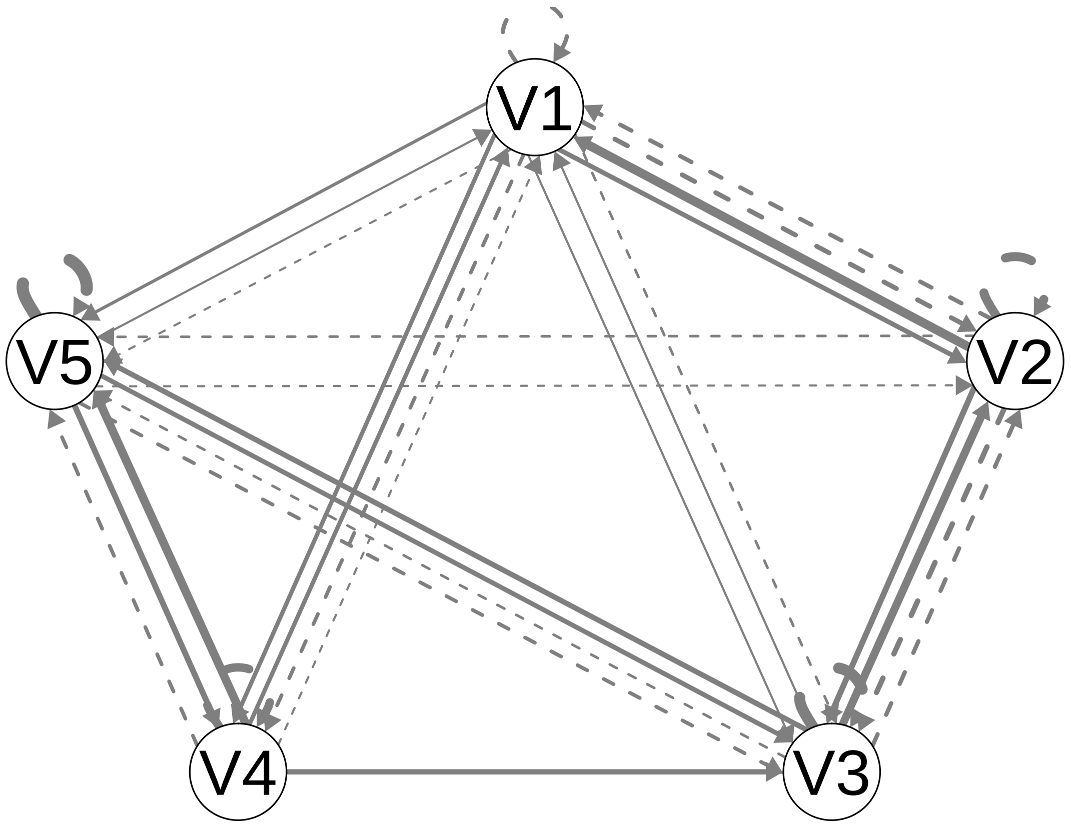
In what follows we provide no starting information and let the model search start from a null model where no paths are estimated - not even the AR effects.

```
ind_out <-indSEM(data = Data_all,
                 ar = FALSE)
```

The plot below depicts the results. The width of the lines indicates the proportion of individuals for whom the path was found to exist in their model.

```
plot(ind_out)
```

```
## Please specify a file id for individual plots. Otherwise, summary plot is presented.
```



We see a lot of variability in the paths, despite there being only 2 patterns for all individuals and a few paths that exist for all individuals. It seems we have a lot of spurious relations here.

Results from starting with AR effects estimated

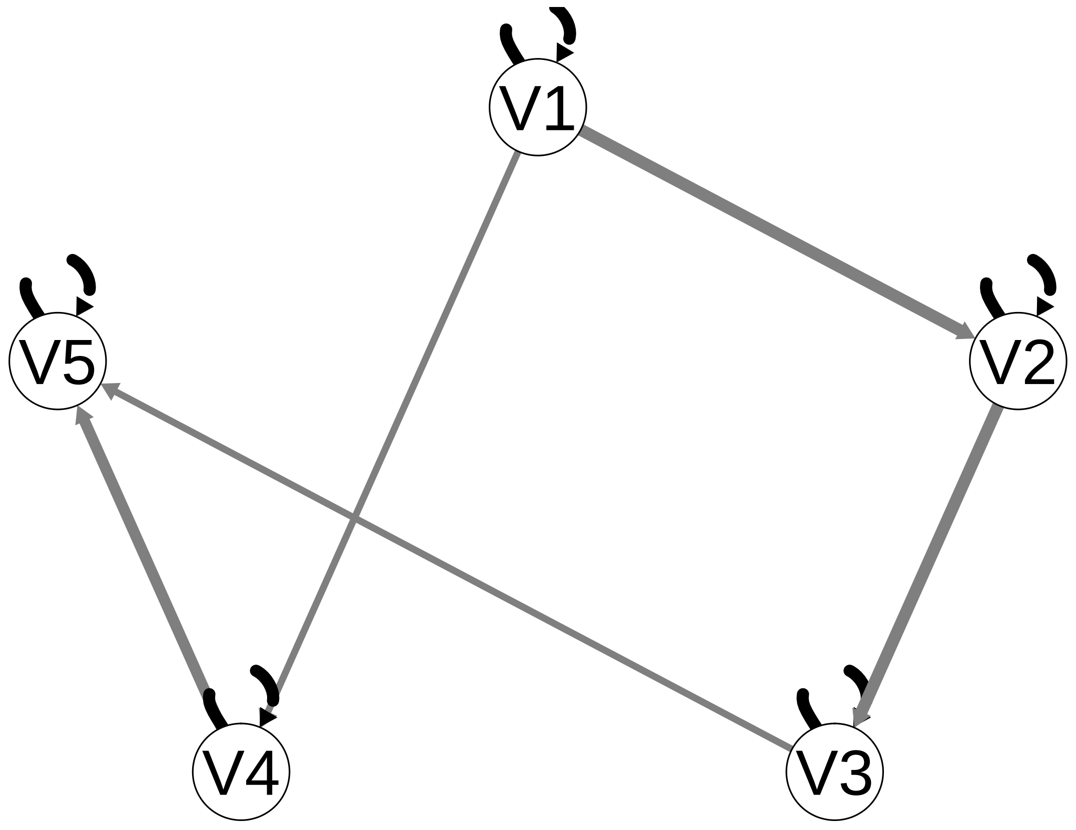
Now let's give the model some information. If we start with the AR effects estimated, the search procedure will probably work better. This is because by starting with the AR effects estimated then the directionality of the relation can be obtained.

```
ind_out_AR <-indSEM(data = Data_all,
                 ar = TRUE)
```

Now this looks much better. We see that only the paths that were in the data-generating structures are present here. Note that the AR effects (the dashed recursive lines on each variable) are black. This is because they are estimated for each individual.

```
plot(ind_out_AR)
```

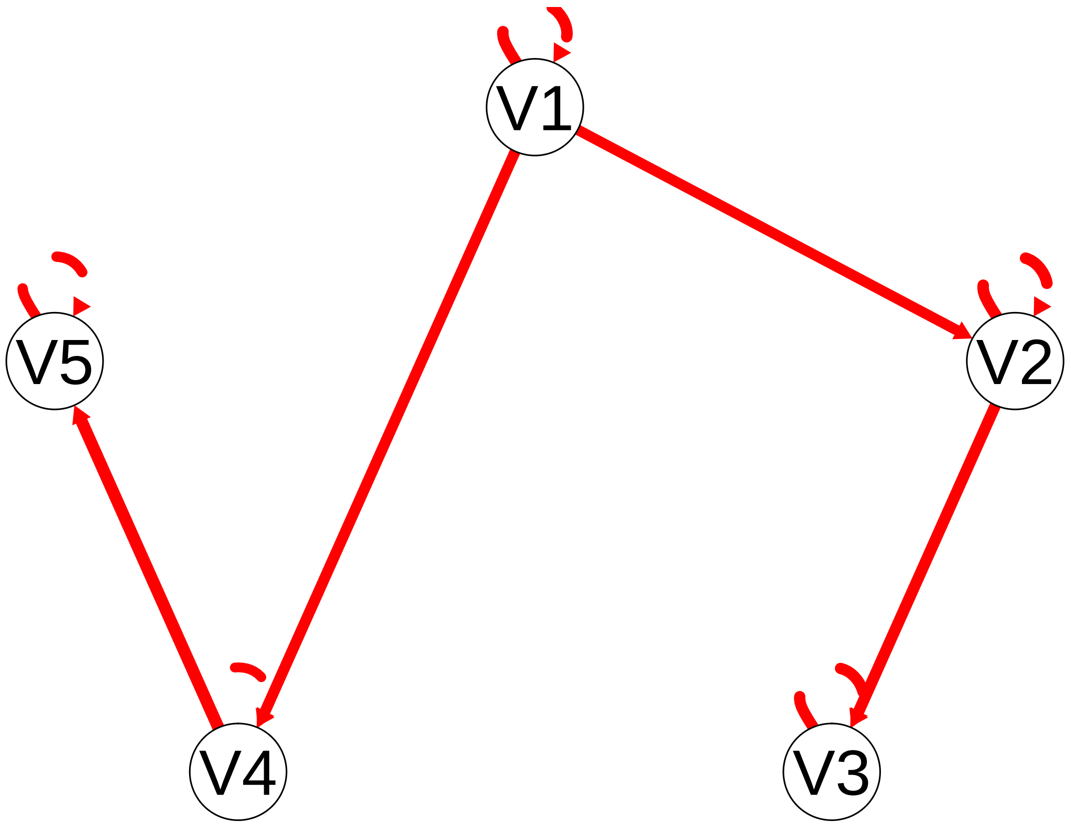
```
## Please specify a file id for individual plots. Otherwise, summary plot is presented.
```



Individual plot

We can look at just one individual. Let's grab person #1.

```
plot(ind_out_AR$plots[[1]])
```



Here, red lines indicate positive values (hot). We see that the data generating pattern for this individual (in subgroup 1) was perfectly recovered.

Graphical VAR

Graphical VAR conducts individual-level analyses using LASSO to induce sparsity in the VAR matrix of coefficients, the precision matrix of the residuals, and the lag-0 precision matrix of observed variables.

Let's see what results look like for person #1. Remember that the graphical VAR model differs from the data generating model - the contemporaneous relations are captured via partial correlations of the residuals rather than direct relations among observed variables. Nonetheless, one should be able to transform back and forth from an SVAR/uSEM (the data generating model in this case) to the VAR with correlated residuals, as shown in the discussion of SVAR in Chapter 4[1].

Specifically, we can premultiply the Φ matrix by $(I - A)^{-1}$, where I is an identity matrix of dimension p and the other matrices are as defined above. We can premultiply the Ψ matrix by $(I - A)^{-1}$ and post-multiply by $(I - A')^{-1}$ (where ' indicates transpose).

[1]More details can be found in: Gates, Molenaar, Hillary, Ram & Rovine, 2010, Automatic search for fMRI connectivity mapping: An alternative to Granger causality testing using formal equivalences among SEM path modeling, VAR, and unified SEM, *NeuroImage, 50*, pp. 1118-1125

```r
negAinv <- solve(diag(5)-A1)
phiNegAinv <- negAinv %*% Phi1
phiNegAinv
```

```
##       [,1] [,2] [,3] [,4] [,5]
## [1,] 0.500 0.00  0.0 0.00  0.0
## [2,] 0.350 0.50  0.0 0.00  0.0
## [3,] 0.245 0.35  0.5 0.00  0.0
## [4,] 0.350 0.00  0.0 0.50  0.0
## [5,] 0.245 0.00  0.0 0.35  0.5
```

```r
# the Psi matrix here is a diagonal of 1, the variance of the residuals
psiNegAinv <- negAinv %*% solve(diag(5)-t(A1))
psiNegAinv
```

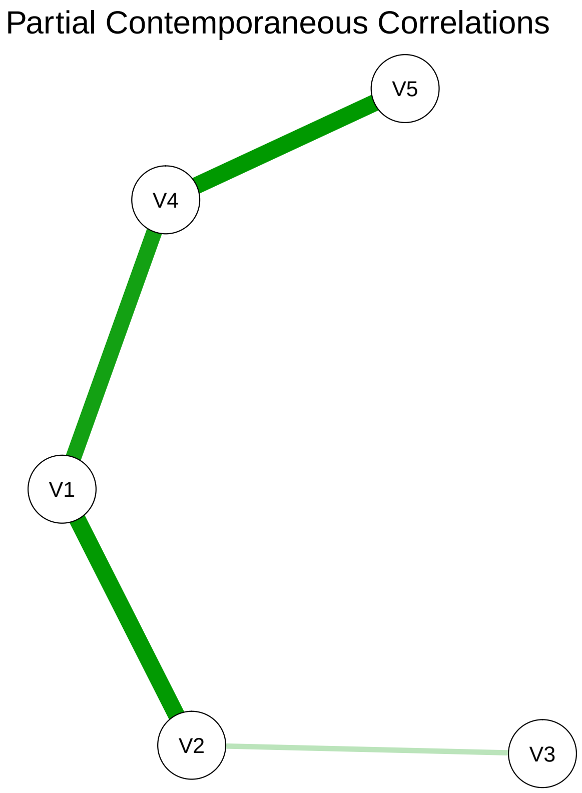
```
##      [,1]  [,2]   [,3]  [,4]   [,5]
## [1,] 1.00 0.700 0.4900 0.700 0.4900
## [2,] 0.70 1.490 1.0430 0.490 0.3430
## [3,] 0.49 1.043 1.7301 0.343 0.2401
## [4,] 0.70 0.490 0.3430 1.490 1.0430
## [5,] 0.49 0.343 0.2401 1.043 1.7301
```

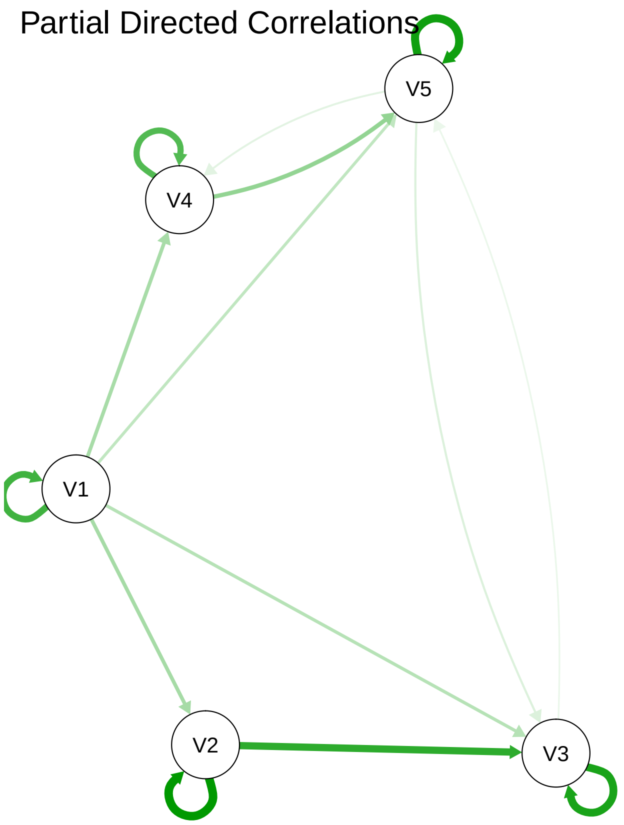
The transformed VAR matrix (phiNegAinv) suggests that we may expect directed lagged relations from Variable 1 to all other variables. We also will expect the relation from Variable 2 to 3 to be present, as well as the one from Variable 4 to 5.

The transformed Ψ matrix is a bit full, and not directly comparable to what we might expect from a partial correlation matrix (which is used in graphical VAR). So, we can compare those results to the A matrix to see if contemporaneous results were captured here.

```r
# run gVAR on person 1
gVAR_out <- graphicalVAR(Data_all[[1]])

plot(gVAR_out)
```





Here, the “Partial contemporaneous relations” are the relations among nodes after taking into account the lag-1 relations as well as partialling out the relations the nodes have with each other node. The “Partial Directed Correlations” are the standardized VAR estimates.

We see that the results correspond reasonably well to the data generating model despite the fact that the DGM did not match the model structure available here (i.e., the contemporaneous relations were directed in the DGM). Notably, the lag-1 standardized relations match almost exactly the transformed matrix above: V1 related to all other variables, V2 predicted V3, and V4 predicted V5. There was just one small spurious VAR relation from V3 to V5.

The contemporaneous correlations also mapped on well to the DGM. We see the relations between V1, V4, and V2 emerge here - these were in the contemporaneous matrix for the data generating model. We also found V2 to V3, but missed V4 to V5.

Taken together, this shows that both methods can recover the DGM reasonably well. This also demonstrates that what appears to be lagged effects may in fact be contemporaneous, and vice versa, since this is a transformation with no error.

# GIMME The above investigations have suggested a few things.

1. When the AR effects are not modeled spurious relations may emerge.
2. When directed contemporaneous relations exist, they might surface as lagged.

Oftentime individual-level searches lead to spurious relations as well. This is because the starting model that contains only AR effects might be far from the final model, and one wrong path selection early on will have deletrious downstream effects.

GIMME was developed to help individual-level model searches by first looking for relations that are consistent across people. It then starts the individual-model searches with these paths included.

Running basic gimme.

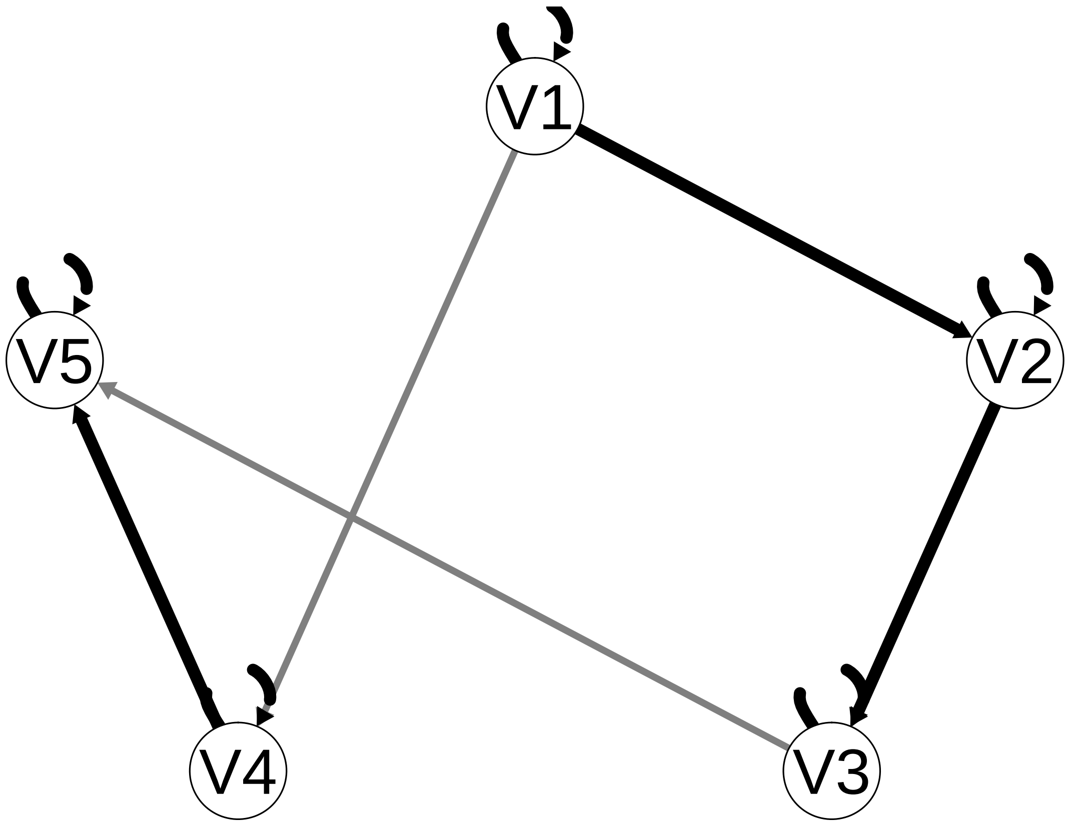
The default in gimme is to have AR effects estimatd as a starting point for the model. It then identifies which path is the best path to add when looking across the majority of individuals.

There are lots of options for gimme, but if your data are in list form then you only need to provide the name of the data. Here we also save results to an optional output folder as well as to an R object.

```
gimme_out <- gimme(data = Data_all,
                   out = "~/Desktop/output")
```

```
plot(gimme_out)
```

```
## Please specify a file id for individual plots. Otherwise, summary plot is presented.
```



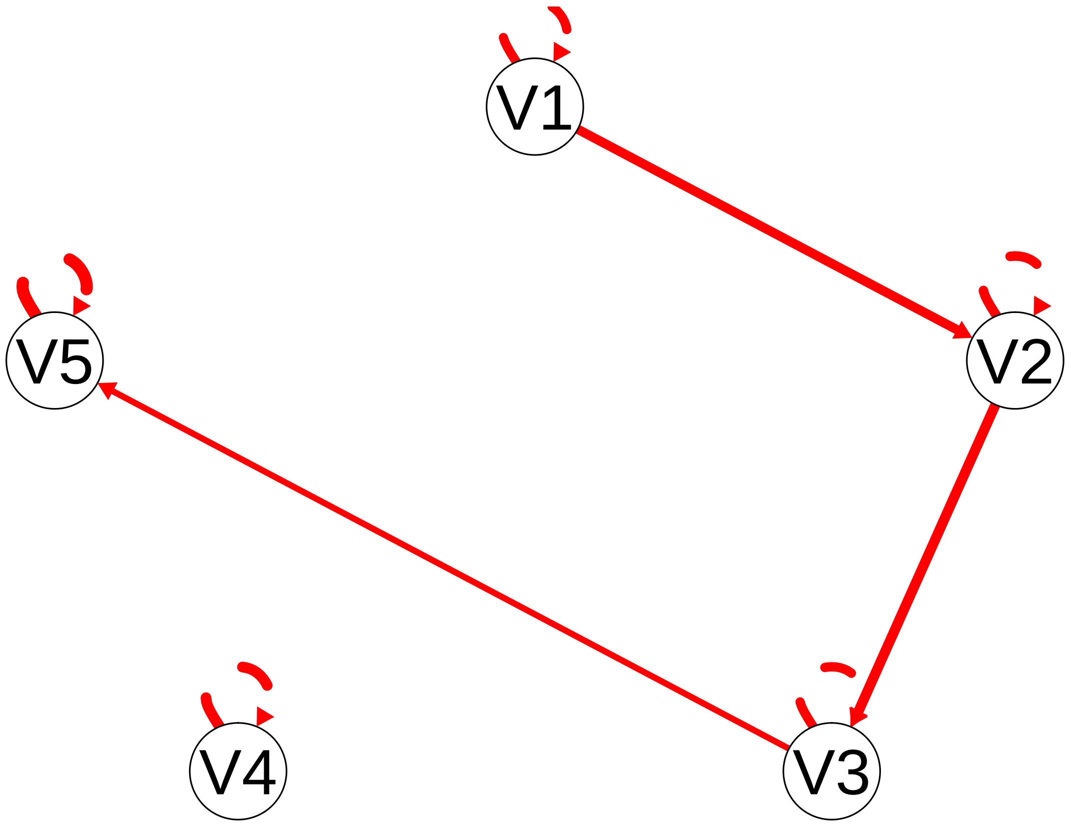
What would happen if we aggregated the data?

We can concatenate the data and arrive at results. This is similar to averaging lag-zero correlation matrices, a common practice in functional MRI research.

In the function aggSEM, paths are iteratively added according to the MIs.

Note that aggregating in this way is not endorsed by the gimme creators.

```r
agg_out <- aggSEM(data = Data_all)
plot(agg_out)
```



We see that the path from V4 to V5 is missing. This path was positive for half the individuals and negative for the other half.

The approach did appropriately find the paths fro V1 to V2 and V2 to V3. It included one of the subgroup-level paths.