BTVN-B working example

This notebook shows the basic steps for estimating a BTVN-B model on a simulated dataset. Just extract folder *btvn_example.zip* and set it as working directory.

Here are the libraries required for running this example. For instructions on installing package *cmdstanr*, see https://mc-stan.org/cmdstanr/articles/cmdstanr.html.

```
library(igraph) #version 1.5.0
```

```
## Warning: package 'igraph' was built under R version 4.2.3
```

```
library(cmdstanr) #version 0.5.3
library(posterior) #version 1.4.0
```

```
## Warning: package 'posterior' was built under R version 4.2.2
```

The functions used for edge selection and performance evaluation can be found in *btvn_functions.R*.

```
source("btvn_functions.R")
```

Loading the dataset: a Stan compatible list of simulation data, list of ground-truth adjacency matrices and list of covariance matrices used in the data simulation. The code for data generation can be found in *data_generation.R*.

```
example_data <- readRDS("btvn_sim_data.rds")
example_data_adj <- readRDS("btvn_sim_adj.rds")
example_data_true_sigma <- readRDS("btvn_sim_sigma.rds")
```

Load and compile Stan model file:

For small datasets, multi-threading is not necessary. With small datasets, just change *stan_threads* to *FALSE* Multiple model options are included in *btvn_master.stan*. To enable and disable features, remove or add //, respectively. Option 'force=*TRUE*' makes sure that the model is compiled again after making changes to the Stan file. For this demonstration, the extra shrinkage prior on the diagonal elements $\sqrt{\Omega_{tii}}$ was used.

```
btvn_b <- "btvn_master.stan"
mod_btvn_b <- cmdstan_model(btvn_b, cpp_options = list(stan_threads = TRUE),
                            stanc_options = list("O1"), force=TRUE)
```

The model parameters can then be estimated by running

```
fit_btvn_b <- mod_btvn_b$sample(
  data = example_data,
  seed = 12345,
  chains = 1,
  parallel_chains = 1,
  threads_per_chain = 8,
  iter_warmup = 500,
  iter_sampling = 1500,
  fixed_param = FALSE,
  adapt_delta = 0.99,
  max_treedepth = 10
)
```

The above estimation took 14256.1 seconds.

Next, we can save the MCMC samples and construct a list of adjacency matrices.

```
draws_btvn_b <- as_draws_rvars(fit_btvn_b$draws())
btvn_b_adj <- rho_res_mat_to_adj_list(rho_res_mat = fit_btvn_b, N = 100,
                              M = 100, n_time_points = 10, credible_level = 0.99)
```

Alternatively, the required parameter estimates (posterior median) are also available in *btvn_b_rho_l.RDS*, *btvn_b_sigma.RDS*, and the estimated adjacency matrices in *btvn_b_adj.RDS* This is since the samples take around 540 000kb of storage when compressed into *.RDS* file

```
rho_l_estimates <- readRDS("btvn_b_rho_l.RDS")
sigma_estimates <- readRDS("btvn_b_sigma.RDS")
adj_btvn_b <- readRDS("btvn_b_adj.RDS")
```

Calculating the performance scores

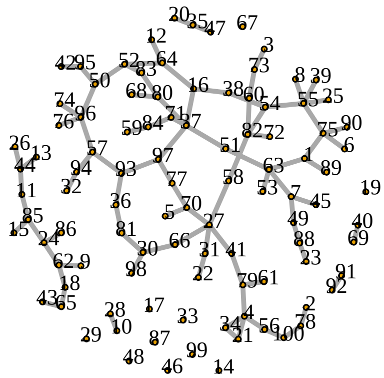
```
ts_scores(example_data_adj, adj_btvn_b, 10)
```

```
##             ACC   ACC_bal       MCC        F1       TPR       TNR       PPV
##  [1,] 0.9971717 0.9495876 0.9268553 0.9278351 0.9000000 0.9991753 0.9574468
##  [2,] 0.9971717 0.9495876 0.9268553 0.9278351 0.9000000 0.9991753 0.9574468
##  [3,] 0.9971717 0.9495876 0.9268553 0.9278351 0.9000000 0.9991753 0.9574468
##  [4,] 0.9971717 0.9495876 0.9268553 0.9278351 0.9000000 0.9991753 0.9574468
##  [5,] 0.9971717 0.9495876 0.9268553 0.9278351 0.9000000 0.9991753 0.9574468
##  [6,] 0.9973737 0.9427539 0.9348258 0.9346734 0.8857143 0.9997936 0.9893617
##  [7,] 0.9973737 0.9427539 0.9348258 0.9346734 0.8857143 0.9997936 0.9893617
##  [8,] 0.9973737 0.9427539 0.9348258 0.9346734 0.8857143 0.9997936 0.9893617
##  [9,] 0.9973737 0.9427539 0.9348258 0.9346734 0.8857143 0.9997936 0.9893617
## [10,] 0.9967677 0.9444845 0.9163350 0.9175258 0.8900000 0.9989691 0.9468085
##             NPV       FNR          FPR         FOR      LRp       LRn
##  [1,] 0.9979407 0.1000000 0.0008247423 0.002059308 1091.250 0.1000825
##  [2,] 0.9979407 0.1000000 0.0008247423 0.002059308 1091.250 0.1000825
##  [3,] 0.9979407 0.1000000 0.0008247423 0.002059308 1091.250 0.1000825
##  [4,] 0.9979407 0.1000000 0.0008247423 0.002059308 1091.250 0.1000825
##  [5,] 0.9979407 0.1000000 0.0008247423 0.002059308 1091.250 0.1000825
##  [6,] 0.9975288 0.1142857 0.0002063983 0.002471170 4291.286 0.1143093
##  [7,] 0.9975288 0.1142857 0.0002063983 0.002471170 4291.286 0.1143093
##  [8,] 0.9975288 0.1142857 0.0002063983 0.002471170 4291.286 0.1143093
##  [9,] 0.9975288 0.1142857 0.0002063983 0.002471170 4291.286 0.1143093
## [10,] 0.9977348 0.1100000 0.0010309278 0.002265239  863.300 0.1101135
```

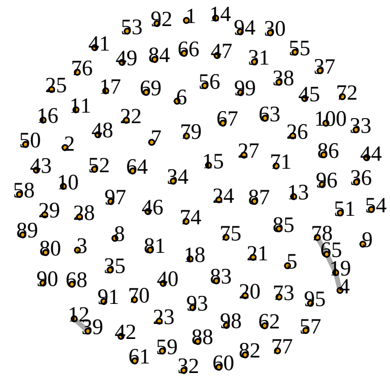
Visual inspection of the first network

```
similarity_plots(example_data_adj[[1]], adj_btvn_b[[1]])
```

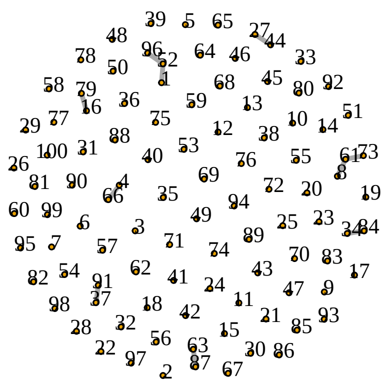
TRUE POSITIVES



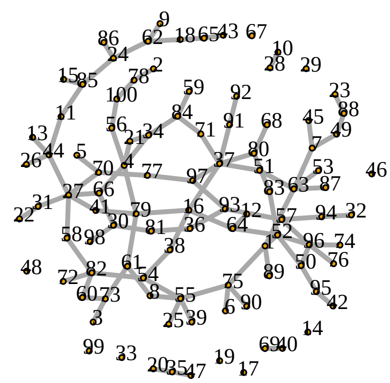
FALSE POSITIVES



FALSE NEGATIVES



TRUE PLOT



Stein's loss of the estimated precision matrices

```
stein_loss_res_result(example_data_true_sigma, rho_l_estimates, sigma_estimates,
                      t = 10, p = 100)
```

```
##  [1] 1.715430 1.820743 1.970933 2.033541 1.757574 1.642654 1.487993 2.050994
##  [9] 1.906196 2.387592
```