parallelization-demo

Data generation

```
library(covdepGE)
library(ggpubr)
```

```
## Loading required package: ggplot2
```

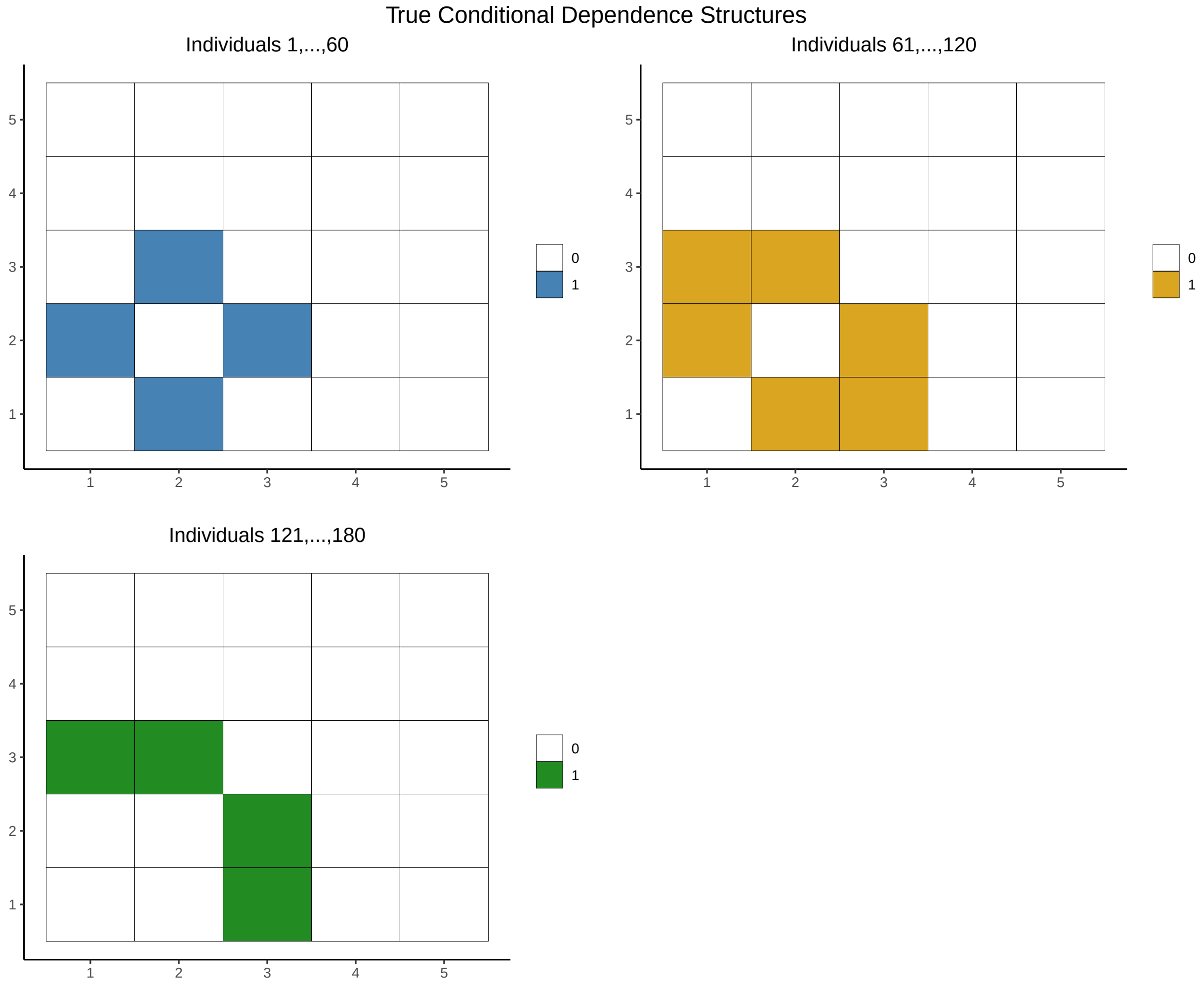
```
library(ggplot2)

setwd("~/TAMU/Research/An approximate Bayesian approach to covariate dependent/covdepGE/dev")
source("generate_data.R")
cont <- generate_continuous()
data_mat <- cont$data
dim(data_mat)
```

```
## [1] 180   5
```

```
Z <- cont$covts

# get all of the unique graphs from the data and visualize them
true_graphs <- lapply(cont$true_precision, function(prec_mat) (prec_mat != 0)
                      - diag(nrow(prec_mat)))
tr_gr_uq <- unique(true_graphs)
indv_gr <- lapply(tr_gr_uq, function(unique_graph) which(sapply(
  true_graphs, function(graph) identical(graph, unique_graph))))
indv_gr_sum <- sapply(indv_gr, function(idx_seq) paste0(min(idx_seq), ",...,",
                                                        max(idx_seq)))
colors <- c("steelblue", "goldenrod", "forestgreen", "tomato2",
            "dodgerblue", "darkorchid")
graph_viz <- lapply(1:length(tr_gr_uq), function(gr_idx) gg_adjMat(
  tr_gr_uq[[gr_idx]], color1 = colors[gr_idx]) +
    ggtitle(paste("Individuals", indv_gr_sum[gr_idx])))
annotate_figure(ggarrange(plotlist = graph_viz),
                top = text_grob("True Conditional Dependence Structures",
                                size = 15))
```



Parallel Variational Updates

Setting `parallel = T` in a call to `covdepGE` performs the variational updates for responses in parallel to one another. Parallel backend may be registered manually by the user, but will otherwise be done automatically. This allows flexibility for the user to configure the parallelization according to their needs.

Manual parallel backend registration:

```
# record time to register parallel backend
start <- Sys.time()
doParallel::registerDoParallel(5)
Sys.time() - start
```

```
## Time difference of 1.120217 secs
```

```r
# run covdepGE in parallel
out <- covdepGE(data_mat, Z, print_time = T, parallel = T, n_sigma = 5)
```

```
## Detected 5 workers
```

```
## Time difference of 1.21825 secs
```

Automatic parallel backend registration

```r
out <- covdepGE(data_mat, Z, print_time = T, parallel = T, num_workers = 7,
                stop_cluster = F, n_sigma = 5)
```

```
## Warning in covdepGE(data_mat, Z, print_time = T, parallel = T, num_workers =
## 7, : No registered workers detected; registering doParallel with 7 workers
```

```
## Time difference of 2.390075 secs
```

By setting `stop_cluster = F`, subsequent parallel calls to `covdepGE` are able to employ the same workers. This avoids the overhead of creating a new cluster.

Efficiency

Large number of candidates

The model in the previous section was relatively simple, with only 5 candidates. In this case, the time to create the cluster, distribute the tasks, and communication from the parent to the children workers outweighs the time savings of parallelizing the updates. Thus, sequential execution is faster for this small model.

```r
out <- covdepGE(data_mat, Z, print_time = T, n_sigma = 5)
```

```
## Time difference of 1.63673 secs
```

However, for a more complex model, the benefits of parallelization become apparent. To increase complexity, I will increase the number of candidate models to 200.

```r
# sequential
out_seq <- covdepGE(data_mat, Z, print_time = T, n_sigma = 200)
```

```
## Time difference of 59.10285 secs
```

```r
# parallel
out_par <- covdepGE(data_mat, Z, print_time = T, n_sigma = 200, parallel = T,
                    num_workers = 6)
```

```
## Detected 7 workers
## Time difference of 10.79052 secs
```

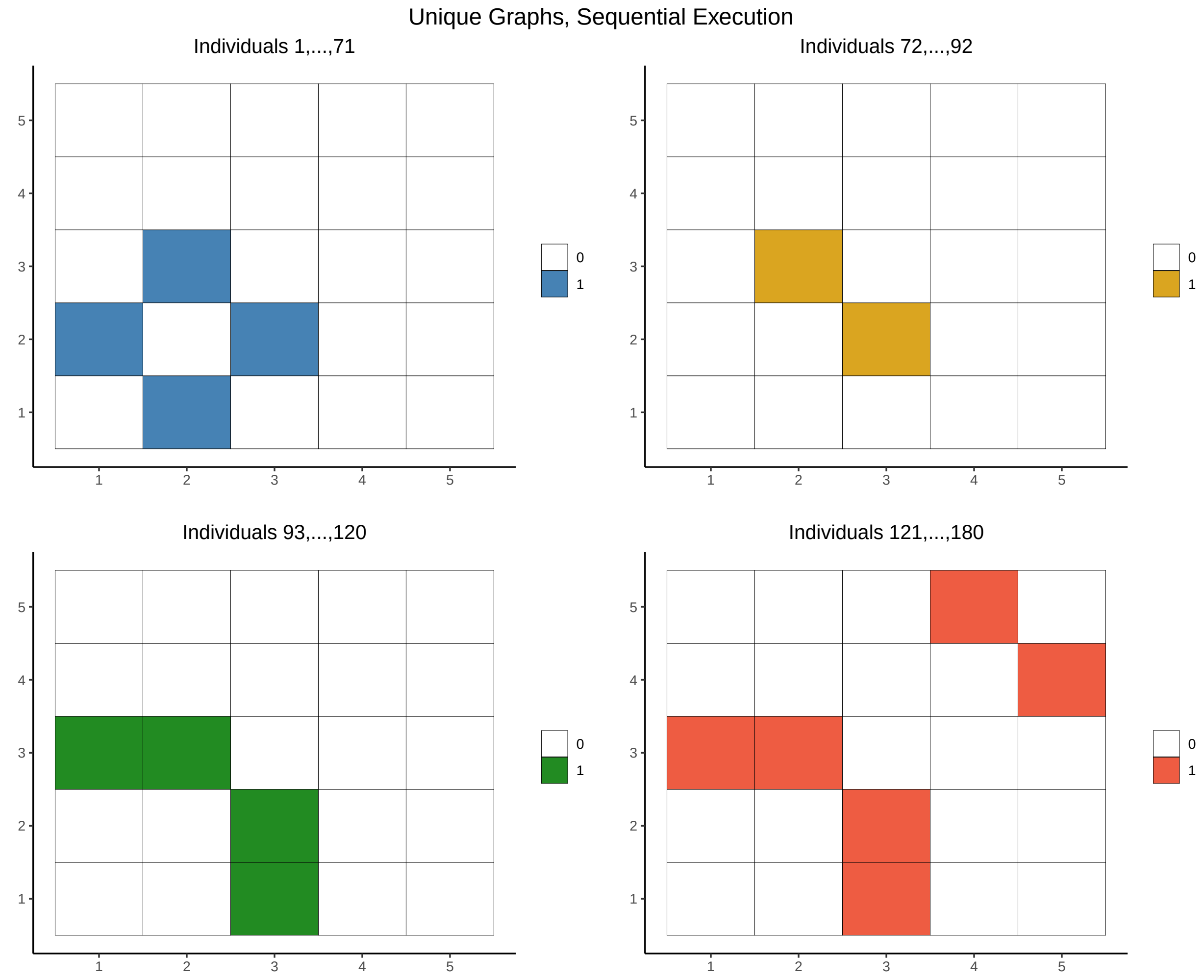
The parallel model outperforms the sequential - additionally, the models produce identical results.

Note the message displayed by the parallel model - it has detected that there are workers on an active cluster from the parallel model with `stop_cluster = F` above. It ignores the `num_workers` argument and re-uses the detected cluster.

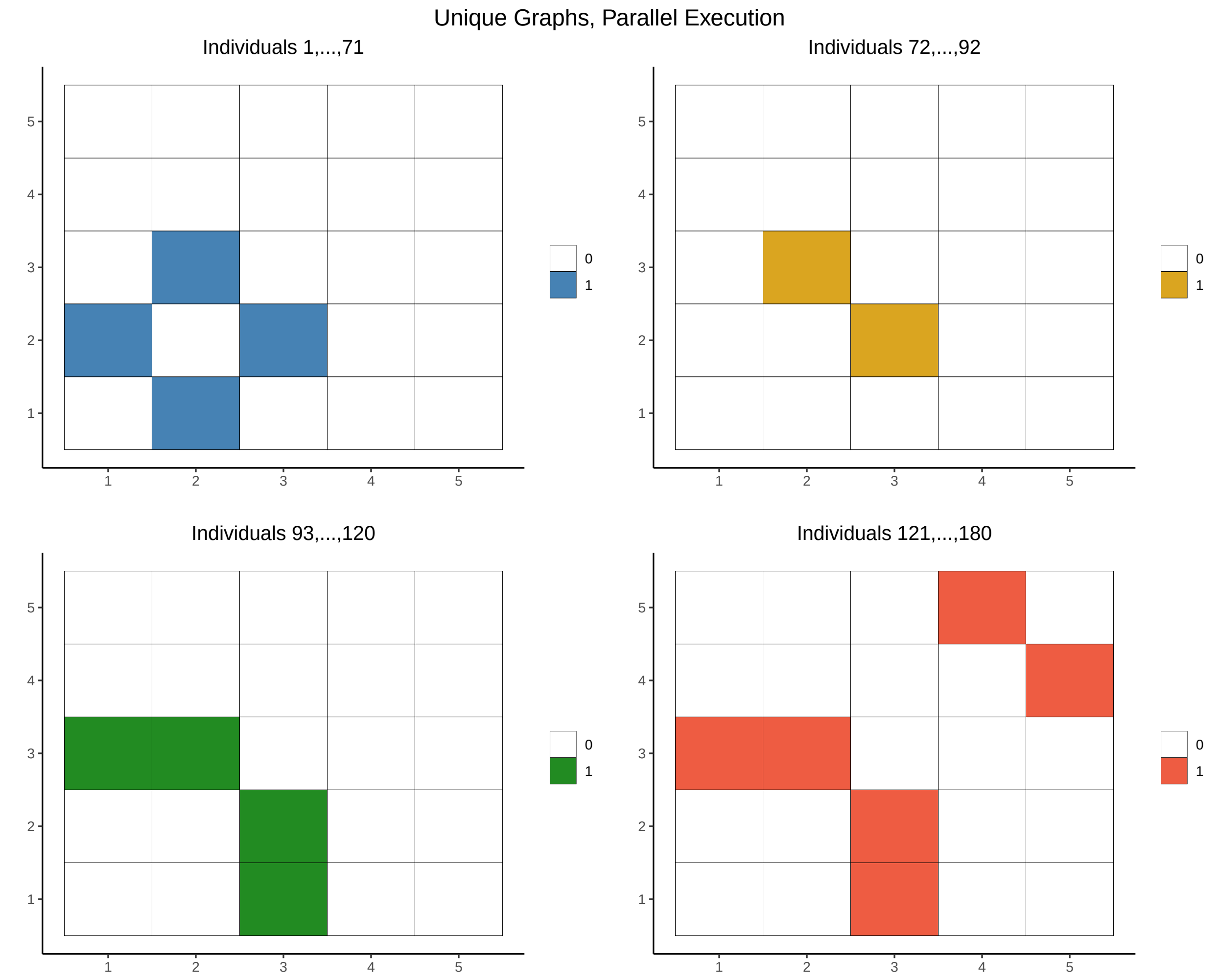
```
identical(out_par, out_seq)
```

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

```
annotate_figure(ggarrange(plotlist = gg_adjMats(out_seq, colors)),
                top = text_grob("Unique Graphs, Sequential Execution",
                                size = 15))
```



```
annotate_figure(ggarrange(plotlist = gg_adjMats(out_par, colors)),
                top = text_grob("Unique Graphs, Parallel Execution",
                                size = 15))
```



Large n

An increase in complexity can also be achieved by again choosing the number of candidate models to be 5 and increasing the sample size. Again, the parallellized updates beat the sequential updates while producing the same result.

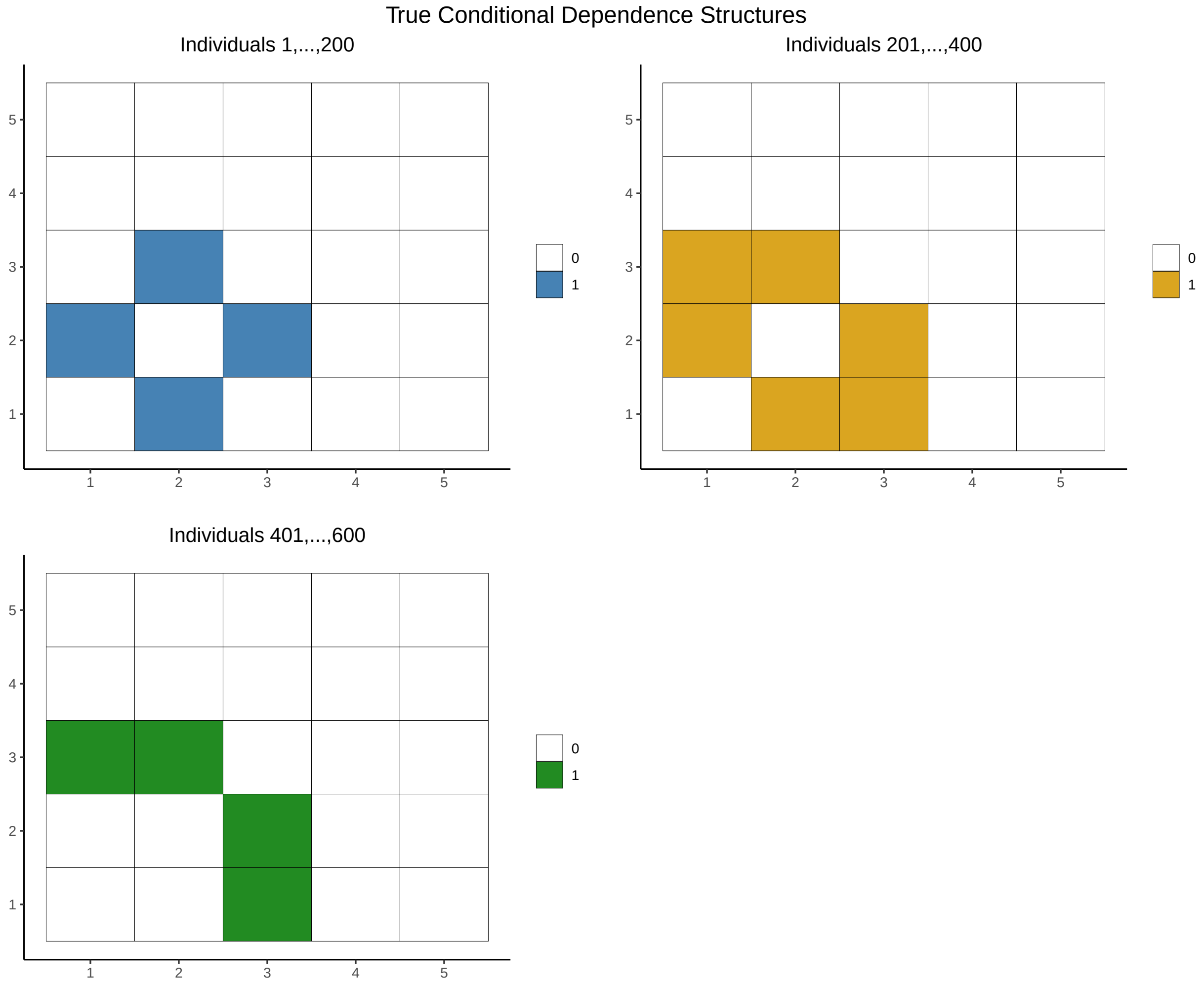
```
sz <- 200
cont <- generate_continuous(n1 = sz, n2 = sz, n3 = sz)
data_mat <- cont$data
dim(data_mat)
```

```
## [1] 600   5
```

```
Z <- cont$covts

# get all of the unique graphs from the data and visualize them
true_graphs <- lapply(cont$true_precision, function(prec_mat) (prec_mat != 0)
                      - diag(nrow(prec_mat)))
tr_gr_uq <- unique(true_graphs)
```

```r
indv_gr <- lapply(tr_gr_uq, function(unique_graph) which(sapply(
  true_graphs, function(graph) identical(graph, unique_graph))))
indv_gr_sum <- sapply(indv_gr, function(idx_seq) paste0(min(idx_seq), ",...,",
                                                        max(idx_seq)))
graph_viz <- lapply(1:length(tr_gr_uq), function(gr_idx) gg_adjMat(
  tr_gr_uq[[gr_idx]], color1 = colors[gr_idx]) +
    ggtitle(paste("Individuals", indv_gr_sum[gr_idx])))
annotate_figure(ggarrange(plotlist = graph_viz),
                top = text_grob("True Conditional Dependence Structures",
                                size = 15))
```


True Conditional Dependence Structures

Individuals 1,...,200

Individuals 201,...,400

Individuals 401,...,600

Note that since the last parallel call to `covdepGE` did not specify `stop_cluster = F`, the cluster must be re-created.

```r
# sequential
out_seq <- covdepGE(data_mat, Z, print_time = T, n_sigma = 5)
```

```
## Warning in covdepGE(data_mat, Z, print_time = T, n_sigma = 5): For 1/5
## responses, the selected value of sigmabeta_sq was on the grid boundary. See
## return value VB_details
## Time difference of 1.669731 mins
```

```
# parallel
out_par <- covdepGE(data_mat, Z, print_time = T, n_sigma = 5, parallel = T,
                    num_workers = 8)
```

```
## Warning in covdepGE(data_mat, Z, print_time = T, n_sigma = 5, parallel = T, : No
## registered workers detected; registering doParallel with 8 workers
```

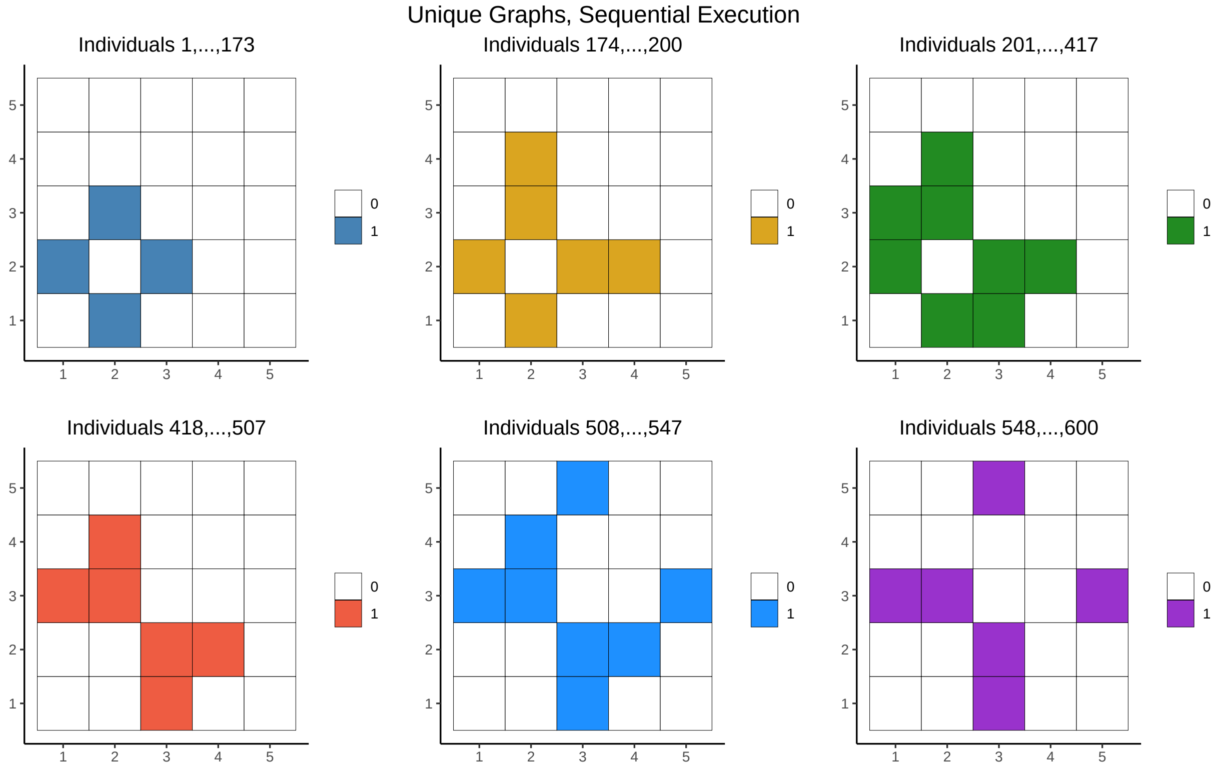
```
## Warning in covdepGE(data_mat, Z, print_time = T, n_sigma = 5, parallel = T, :
## For 1/5 responses, the selected value of sigmabeta_sq was on the grid boundary.
## See return value VB_details
```

```
## Time difference of 30.13334 secs
```

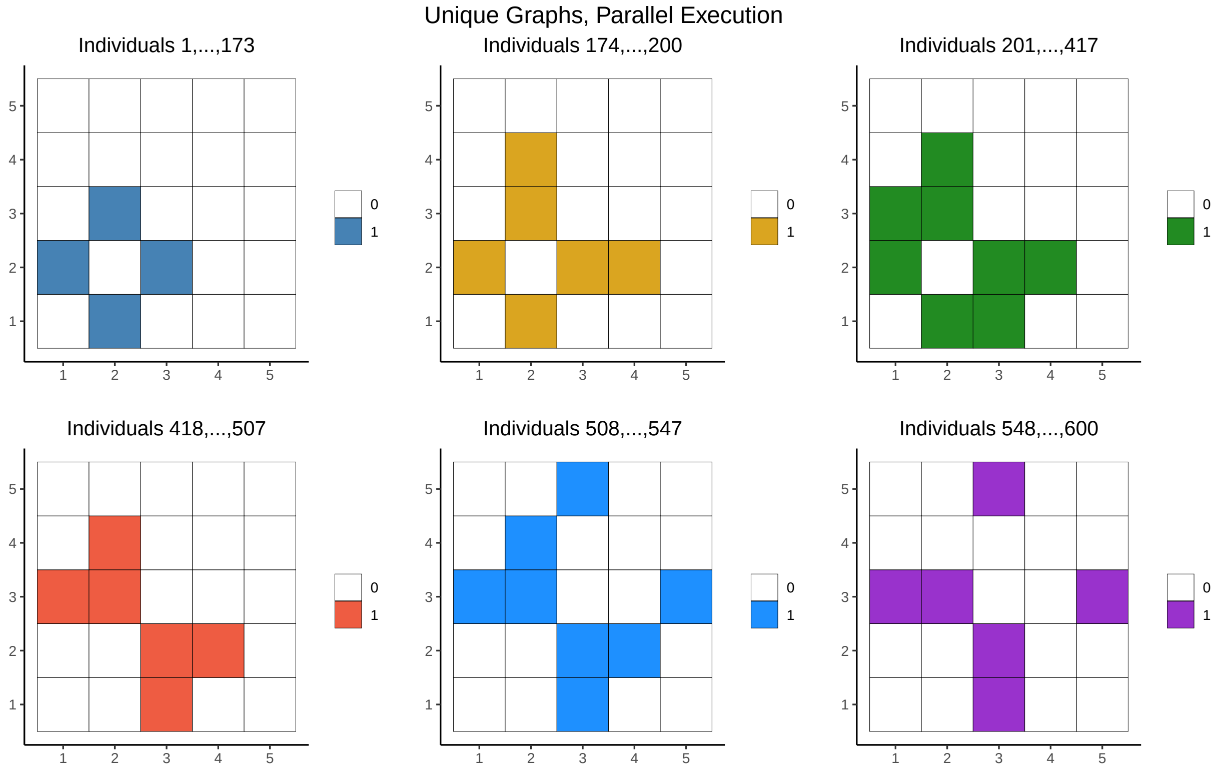
```
identical(out_par, out_seq)
```

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

```
annotate_figure(ggarrange(plotlist = gg_adjMats(out_seq, colors)),
                top = text_grob("Unique Graphs, Sequential Execution",
                                size = 15))
```



```
annotate_figure(ggarrange(plotlist = gg_adjMats(out_par, colors)),
                top = text_grob("Unique Graphs, Parallel Execution",
                                size = 15))
```



Unique Graphs, Parallel Execution

Individuals 1,...,173

Individuals 174,...,200

Individuals 201,...,417

Individuals 418,...,507

Individuals 508,...,547

Individuals 548,...,600