```r
rm(list = ls())
library(covdepGE)
library(loggle)
library(HeteroGGM)
library(mclust)
library(ggplot2)
library(ggpubr)
library(kableExtra)

now <- format(Sys.time(), "%Y%m%d_%H%M%S")

# initialize storage for results, time, and progress tracking
set.seed(1)
n_trials <- 2
results <- vector("list", n_trials + 1)
names(results) <- c(paste0("trial", 1:n_trials), "time")
results$time <- Sys.time()
pb <- txtProgressBar(0, n_trials, style = 3)
```

```
##   |                                                                               |
```

```r
# define data dimensions
p <- 25
(n <- round(2 * 3 * p))
```

```
## [1] 150
```

```r
(nj <- n %/% 3)
```

```
## [1] 50
```

```r
# function for surpressing cat
quiet <- function(x) {
  sink(tempfile())
  on.exit(sink())
  invisible(force(x))
}

# get number of available workers
(num_workers <- parallel::detectCores() - 1)
```

```
## [1] 55
```

```r
# function for evaluating estimated graphs compared to ground truth
eval_est <- function(est, true){

  # get true number of edges and non-edges
  num_edge <- sum(true, na.rm = T)
  num_non <- sum(true == 0, na.rm = T)
  # calculate sensitivity and specificity
  true_edge <- sum(est == true & true == 1, na.rm = T)
  true_non <- sum(est == true & true == 0, na.rm = T)
  sens <- true_edge / num_edge
  spec <- true_non / num_non

  list(sens = sens, spec = spec)
}
```

```
# perform trials
for (j in 1:n_trials){

  # generate the data and create storage for the models
  data <- generateData(p, nj, nj, nj)
  trial <- vector("list", 4)
  names(trial) <- c("data", "covdepGE", "loggle", "HeteroGGM")
  prec_arr_dim <- c(p, p, n)
  trial$data <- data

  # convert the true precision to an array and then to a graph; mask diagonal
  prec <- array(unlist(data$true_precision), prec_arr_dim)
  graph <- (prec != 0) * 1 + replicate(n, diag(rep(NA, p)) * 1)

  # fit each method, save details about results, time, etc.

  # covdepGE
  out_covdepGE <- tryCatch(suppressWarnings(covdepGE(
    data$X, data$Z, parallel = T, num_workers = num_workers)),
    error = function(e) list(error = e))
  if (names(out_covdepGE)[[1]] == "error"){
    out_covdepGE <- out_covdepGE$error
  }else{
    out_covdepGE$time <- as.numeric(out_covdepGE$model_details$elapsed, units = "secs")
    out_covdepGE$str <- array(unlist(out_covdepGE$graphs$graphs), dim = prec_arr_dim)
    out_covdepGE[c("sens", "spec")] <- eval_est(out_covdepGE$str, graph)[c("sens", "spec")]
  }
  trial$covdepGE <- out_covdepGE
  rm(list = "out_covdepGE")
  gc()

  # loggle
  start <- Sys.time()
  out_loggle <- tryCatch(quiet(loggle.cv(t(data$X), num.thread = num_workers)),
                         error = function(e) list(error = e))
  if (names(out_loggle)[[1]] == "error"){
    out_loggle <- out_loggle$error
  }else{
    out_loggle <- out_loggle$cv.select.result
    out_loggle$time <- as.numeric(Sys.time() - start, units = "secs")
    out_loggle$str <- array(unlist(lapply(lapply(
      out_loggle$adj.mat.opt, ‘-‘, diag(p)), as.matrix)), dim = prec_arr_dim)
    out_loggle[c("sens", "spec")] <- eval_est(out_loggle$str, graph)[c("sens", "spec")]
  }
```

```
trial$loggle <- out_loggle
rm(list = "out_loggle")
gc()

# HeteroGGM
clust <- Mclust(data$Z, verbose = F)
lambda <- genelambda.obo(lambda2_min = 0.15)
start <- Sys.time()
out_hetGGM <- tryCatch(GGMPF(lambda, data$X + clust$classification * 10, clust$G),
                       error = function(e) list(error = e))
if (names(out_hetGGM)[[1]] == "error"){
  out_hetGGM <- out_hetGGM$error
}else{
  out_hetGGM$time <- as.numeric(Sys.time() - start, units = "secs")
  out_hetGGM$str <- (out_hetGGM$Theta_hat.list[[out_hetGGM$Opt_num]] != 0) * 1
  out_hetGGM$str <- out_hetGGM$str - replicate(dim(out_hetGGM$str)[3], diag(p))
  out_hetGGM$str <- out_hetGGM$str[ , , out_hetGGM$member.list[[out_hetGGM$Opt_num]]]
  out_hetGGM[c("sens", "spec")] <- eval_est(out_hetGGM$str, graph)[c("sens", "spec")]
}
trial$HeteroGGM <- out_hetGGM
rm(list = "out_hetGGM")
gc()

# for hp tuning lambda grid
# het_gr <- vector("list", n)
# for (l in 1:n){
#   het_gr[[l]] <- out_hetGGM$str[ , , l]
# }
#
# # find the unique graphs
# unique_graphs <- unique(het_gr)
#
# # create a list where the j-th element is the j-th unique graph and the
# # indices of the observations corresponding to this graph
# unique_sum <- vector("list", length(unique_graphs))
# names(unique_sum) <- paste0("graph", 1:length(unique_graphs))
#
# # iterate over each of the unique graphs
# for (j in 1:length(unique_graphs)){
#
#   # fix the unique graph
#   graphj <- unique_graphs[[j]]
#
#   # find indices of the observations corresponding to this graph
#   graph_inds <- which(sapply(het_gr, identical, graphj))
#
#   # split up the contiguous subsequences of these indices
#   cont_inds <- split(sort(graph_inds), cumsum(c(1, diff(sort(graph_inds))
#                                                 != 1)))
#
#   # create a character summary for each of the contiguous sequences
#   inds_sum <- sapply(cont_inds, function(idx_seq) ifelse(length(
#     idx_seq) > 3, paste0(min(idx_seq), ",...,", max(idx_seq)),
  #     paste0(idx_seq, collapse = ","))
  #
  #   # combine the summary
  #   inds_sum <- paste0(inds_sum, collapse = ",")
  #
  #   # add the graph, indices, and summary to the unique graphs summary list
  #   unique_sum[[j]] <- list(graph = graphj, indices = graph_inds,
  #                           ind_sum = inds_sum)
  # }
  # for (graphj in unique_sum){
  #   print(matViz(graphj$graph) + ggtitle(graphj$ind_sum))
  # }
  # out_hetGGM$Opt_lambda
  # lambda

  # save the trial and update the progress bar
  results[[j]] <- trial
  setTxtProgressBar(pb, j)
}

# save the final time and the results
(results$time <- Sys.time() - results$time)
save(results, file = paste0("results_", now, ".Rda"))
```

```
## Analysis

#load("simulation_study/results.Rda")
load("/home/jacob.a.helwig/covdepGE/simulation_study/results_20220817_150045.Rda")

# extract each of the models
results <- results[setdiff(names(results), "time")]
covdepGE_models <- lapply(results, ‘[[‘, "covdepGE")
loggle_models <- lapply(results, ‘[[‘, "loggle")
hetGGM_models <- lapply(results, ‘[[‘, "HeteroGGM")

# extract the sensitivties, specificities, and time
covdepGE_sens <- sapply(covdepGE_models, ‘[[‘, "sens")
covdepGE_spec <- sapply(covdepGE_models, ‘[[‘, "spec")
covdepGE_times <- sapply(covdepGE_models, ‘[[‘, "time")
loggle_sens <- sapply(loggle_models, ‘[[‘, "sens")
loggle_spec <- sapply(loggle_models, ‘[[‘, "spec")
loggle_times <- sapply(loggle_models, ‘[[‘, "time")
hetGGM_sens <- sapply(hetGGM_models, ‘[[‘, "sens")
hetGGM_spec <- sapply(hetGGM_models, ‘[[‘, "spec")
hetGGM_times <- sapply(hetGGM_models, ‘[[‘, "time")

# sensitivity analysis
summary(covdepGE_sens)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.6714  0.7286  0.7857  0.7857  0.8429  0.9000
```

```
summary(loggle_sens)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.8914  0.8936  0.8957  0.8957  0.8979  0.9000
```

```
summary(hetGGM_sens)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.8571  0.8571  0.8571  0.8571  0.8571  0.8571
```

```
summary(covdepGE_sens - loggle_sens)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  -0.220  -0.165  -0.110  -0.110  -0.055   0.000
```

```
summary(covdepGE_sens - hetGGM_sens)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
## -0.18571 -0.12857 -0.07143 -0.07143 -0.01429  0.04286
```

```
# specificity analysis
summary(covdepGE_spec)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.9987  0.9988  0.9990  0.9990  0.9991  0.9992
```

```
summary(loggle_spec)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.9984  0.9985  0.9986  0.9986  0.9987  0.9988
```

```
summary(hetGGM_spec)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.9429  0.9542  0.9656  0.9656  0.9769  0.9883
```

```
summary(covdepGE_spec - loggle_spec)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max. 
## -0.0001568  0.0001008  0.0003583  0.0003583  0.0006159  0.0008735
```

```
summary(covdepGE_spec - hetGGM_spec)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
## 0.01095 0.02216 0.03337 0.03337 0.04458 0.05579
```

```
# time analysis
summary(covdepGE_times)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   6.682   6.753   6.823   6.823   6.894   6.965
```

```
summary(loggle_times)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   339.9   349.3   358.6   358.6   368.0   377.3
```

```
summary(hetGGM_times)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   36.67   39.08   41.48   41.48   43.88   46.28
```

```
# table of results
perf <- apply(cbind(covdepGE_sens, loggle_sens, hetGGM_sens,
                    covdepGE_spec, loggle_spec, hetGGM_spec,
                    covdepGE_times, loggle_times, hetGGM_times),
              2, function(x) paste0(round(mean(x), 3), " (", round(sd(x), 3), ")"))
perf_df <- data.frame(t(matrix(perf, 3, 3)))
row.names(perf_df) <- c("Sensitivity", "Specificity", "Time(s)")
colnames(perf_df) <- c("covdepGE", "loggle", "HeteroGGM")
kbl(perf_df, format = "latex", booktabs = T)
```

	covdepGE	loggle	HeteroGGM
Sensitivity	0.786 (0.162)	0.896 (0.006)	0.857 (0)
Specificity	0.999 (0)	0.999 (0)	0.966 (0.032)
Time(s)	6.823 (0.2)	358.611 (26.475)	41.477 (6.792)

```
# visualize graphs
# pred_graphs <- plot(covdepGE_models[[12]])
true_graphs <- unique(lapply(lapply(lapply(
  results$trial1$data$true_precision, ‘!=‘, 0), ‘*‘, 1), ‘-‘, diag(p)))
graph_inds <- c(paste0(c(1, nj), collapse = ",...,"),
                paste0(c(nj + 1, 2 * nj), collapse = ",...,"),
                paste0(c(2 * nj + 1, 3 * nj), collapse = ",...,"))
titles <- paste0("Graph ", 1:3, ", observations ", graph_inds)
true_graphs <- lapply(1:3, function(j)
  matViz(true_graphs[[j]]) +
    ggtitle(titles[[j]]) +
    geom_tile(color = "grey", size = 1e-10) +
    theme(axis.line = element_blank(),
          axis.text.x=element_blank(),
          axis.text.y = element_blank(),
          axis.ticks = element_blank(),
          axis.title.x = element_blank(),
          axis.title.y = element_blank(),
          legend.position = "none")
    )

# pred_graphs <- ggarrange(plotlist = pred_graphs, nrow = 1, legend = F)
true_graphs <- ggarrange(plotlist = true_graphs, nrow = 1, legend = F)
# ggsave("simulation_study/plots/preds.pdf", pred_graphs, height = 4, width = 11)
# ggsave("simulation_study/plots/true.pdf", true_graphs, height = 4, width = 11)
```