Implementation-details

```
# install the package if necessary
if  (!("covdepGE" %in% installed.packages())){
  devtools::install_github("JacobHelwig/covdepGE")
}
library(covdepGE)
?covdepGE
```

```
## starting httpd help server ... done
```

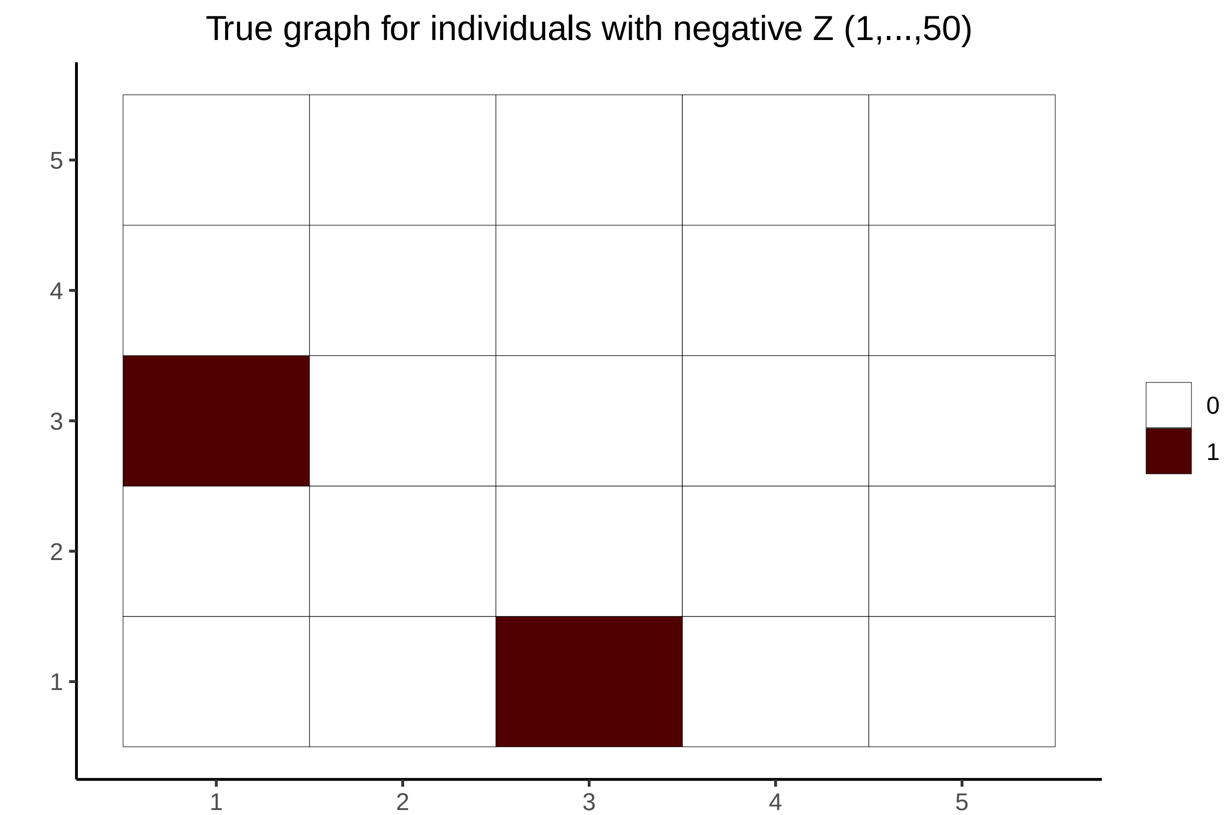
```
set.seed(1)
n <- 100
p <- 4

# generate the extraneous covariate
Z_neg <- sort(runif(n / 2) * -1)
Z_pos <- sort(runif(n / 2))
Z <- c(Z_neg, Z_pos)
summary(Z)
```

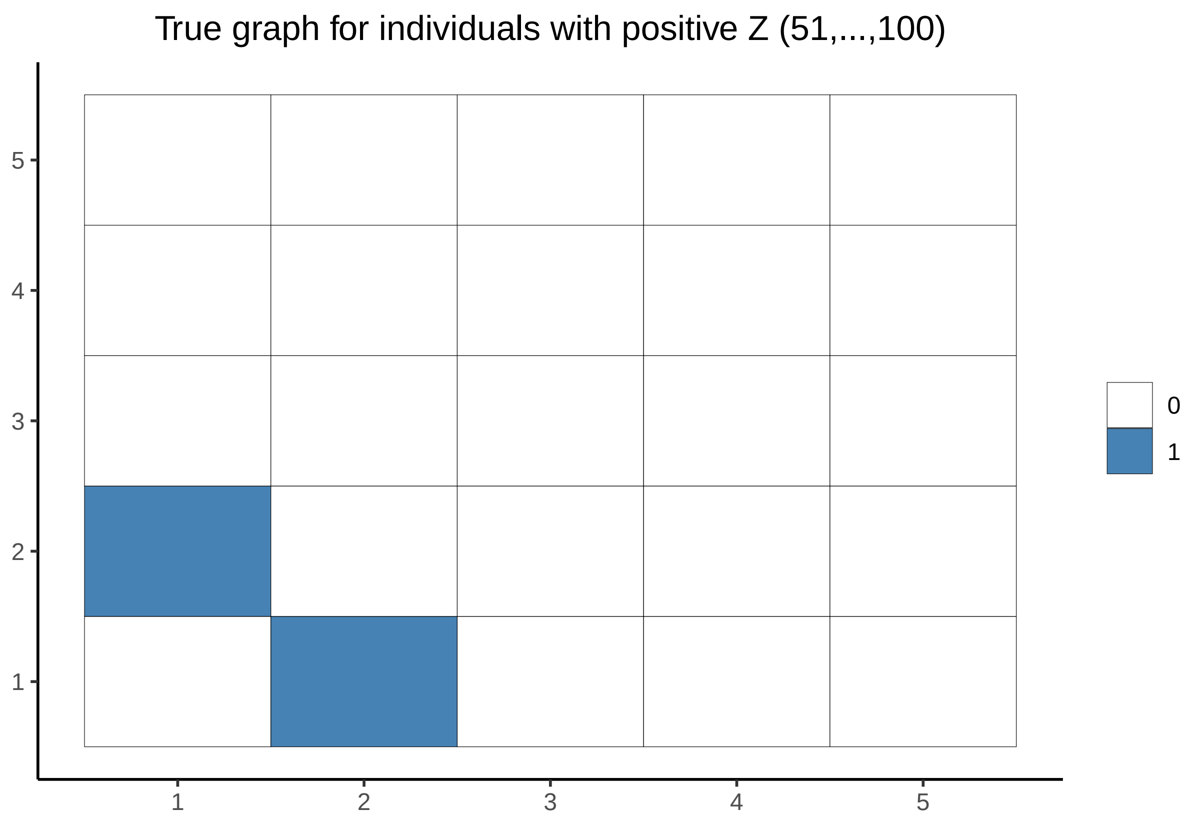
```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.99191 -0.55799  0.02277 -0.01475  0.45622  0.96062
```

```
# create true covariance structure for 2 groups: positive Z and negative Z
true_graph_pos <- true_graph_neg <- matrix(0, p + 1, p + 1)
true_graph_pos[1, 2] <- true_graph_pos[2, 1] <- 1
true_graph_neg[1, 3] <- true_graph_neg[3, 1] <- 1

# visualize the true covariance structures
(gg_adjMat(true_graph_neg) +
    ggplot2::ggtitle("True graph for individuals with negative Z (1,...,50)"))
```



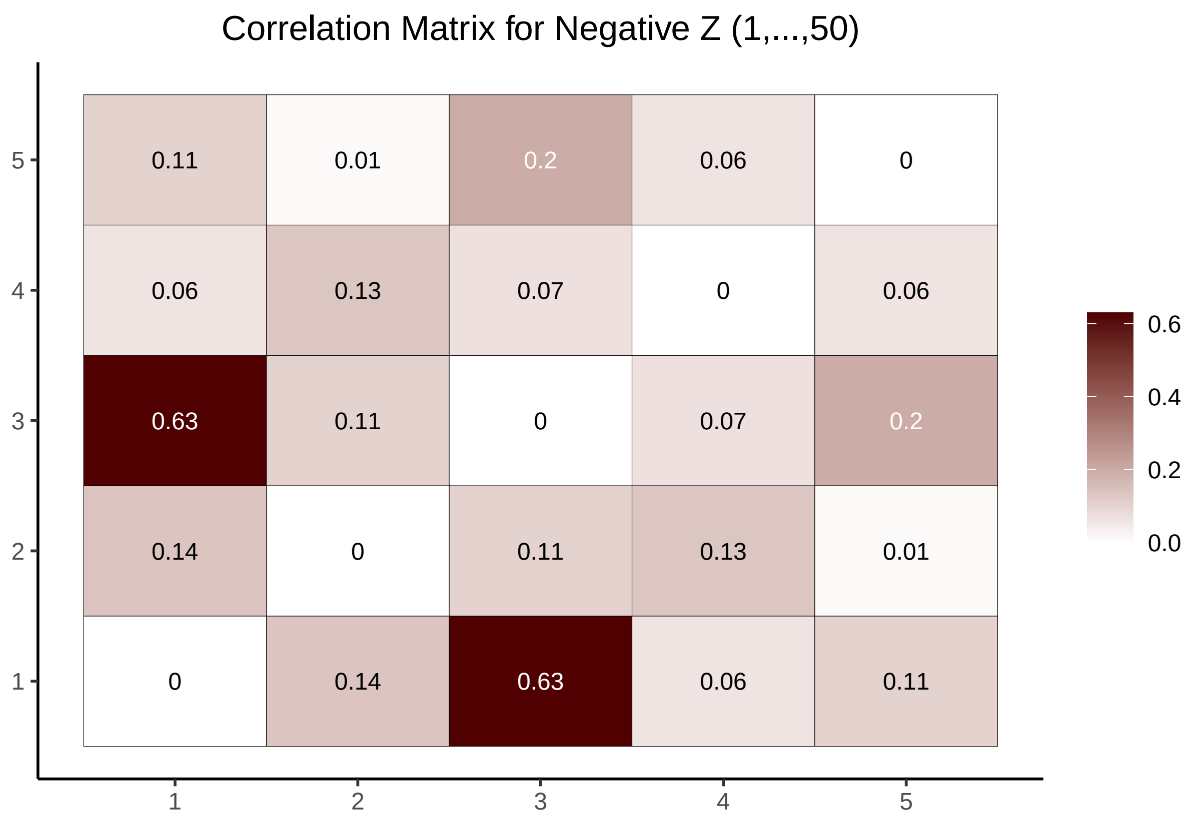

```
(gg_adjMat(true_graph_pos, color1 = "steelblue") +
    ggplot2::ggtitle("True graph for individuals with positive Z (51,...,100)"))
```



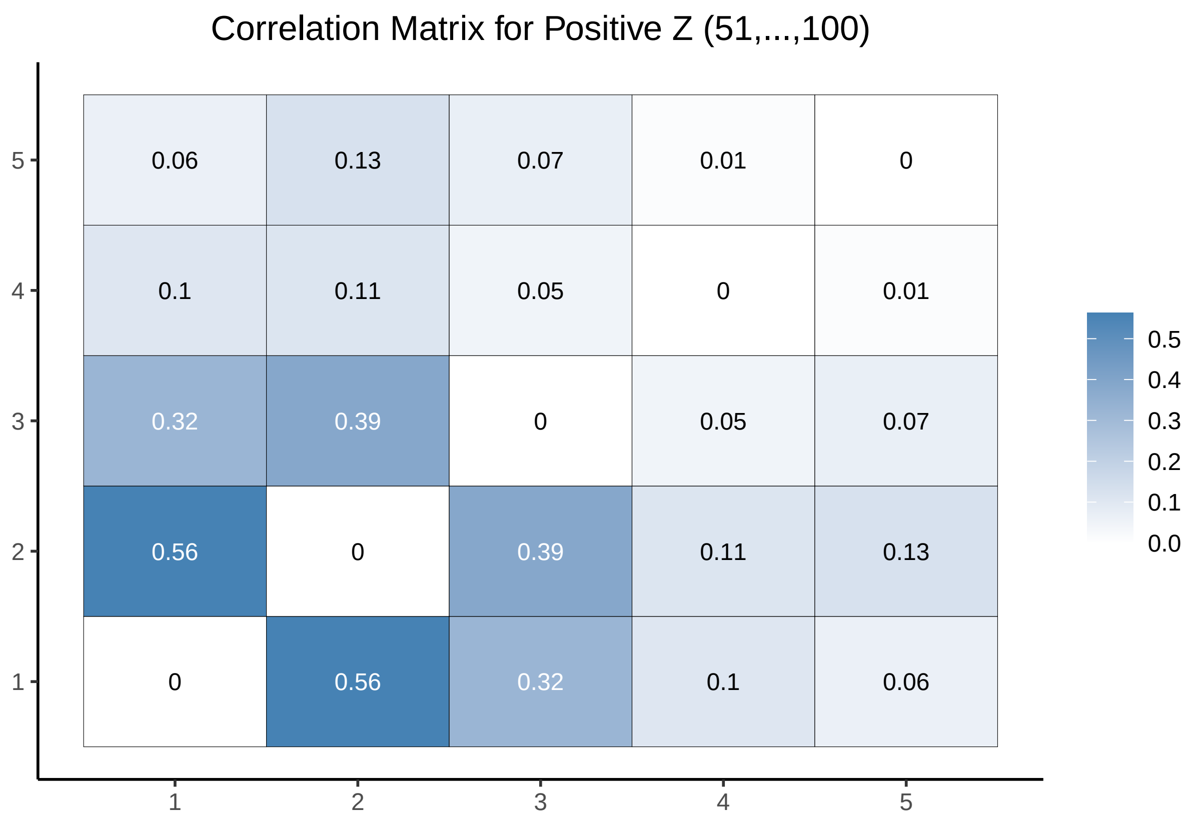

```
# generate the covariance matrices as a function of Z
sigma_mats_neg <- lapply(Z_neg, function(z) z * true_graph_neg + diag(p + 1))
sigma_mats_pos <- lapply(Z_pos, function(z) z * true_graph_pos + diag(p + 1))
sigma_mats <- c(sigma_mats_neg, sigma_mats_pos)

# generate the data using the covariance matrices
data_mat <- t(sapply(sigma_mats, MASS::mvrnorm, n = 1, mu = rep(0, p + 1)))

# visualize the sample correlation
(gg_adjMat(abs(cor(data_mat[1:(n / 2), ])) - diag(p + 1)) +
    ggplot2::ggtitle("Correlation Matrix for Negative Z (1,...,50)"))
```




```
(gg_adjMat(abs(cor(data_mat[(n / 2 + 1):n, ])) - diag(p + 1),
           color1 = "steelblue") +
    ggplot2::ggtitle("Correlation Matrix for Positive Z (51,...,100)"))
```




```r
# use varbvs to get the hyperparameter sigma
sigmasq <- rep(NA, p + 1)
for (j in 1:(p + 1)){
  sigmasq[j] <- mean(varbvs::varbvs(data_mat[ , -j], Z, data_mat[ , j], verbose = F)$sigma)
}
sigmasq
```

```
## [1] 0.7875369 1.2113484 0.8469619 0.7154704 1.0762645
```

```r
mean(sigmasq)
```

```
## [1] 0.9275164
```

```r
# estimate the conditional dependence structure
out <- covdepGE(
                data_mat,
                Z, # extraneous covariates
                kde = T, # whether KDE should be used to calculate bandwidths
                sigmasq = mean(sigmasq), # hyperparameter residual variance
                var_min = 1e-4, # smallest sigmabeta_sq grid value
                var_max = 1, # largest sigmabeta_sq grid value
                n_sigma = 10, # length of the sigmabeta_sq grid
                pi_vec = seq(0.1, 0.3, 0.05), # prior inclusion probability grid
                norm = Inf, # norm to calculate the weights with
                scale = T, # whether the extraneous covariates should be scaled
                tolerance = 1e-11, # variational parameter exit condition 1
                max_iter_final = 1e5, # variational parameter exit condition 2
                edge_threshold = 0.75, # minimum inclusion probability
                sym_method = "min", # how to symmetrize the alpha matrices
                warnings = T # whether warnings should be displayed
                )
```

```
## Warning in covdepGE(data_mat, Z, kde = T, sigmasq = mean(sigmasq), var_min =
## 1e-04, : For 1/5 variables, the selected value of sigmabeta_sq was on the grid
## boundary. See return value CAVI_details

## Warning in covdepGE(data_mat, Z, kde = T, sigmasq = mean(sigmasq), var_min =
## 1e-04, : For 5/5 variables, the selected value of pi was on the grid boundary.
## See return value cavi_details
```

```r
out
```

```
##                       Covariate Dependent Graphical Model
##
## Model ELBO: -25071.49              Unique conditional dependence structures: 3
## n: 100, variables: 5                       Hyperparameter grid size: 50 points
## CAVI converged for 5/5 variables
##
## Model fit completed in 4.907 secs
```

```r
# grid search results
out$CAVI_details
```

```
## [[1]]
## [[1]]$sigmabeta_sq
## [1] 0.129155
##
## [[1]]$pi
## [1] 0.3
##
## [[1]]$ELBO
## [1] -4206.716
##
## [[1]]$converged_iter
## [1] 44
```

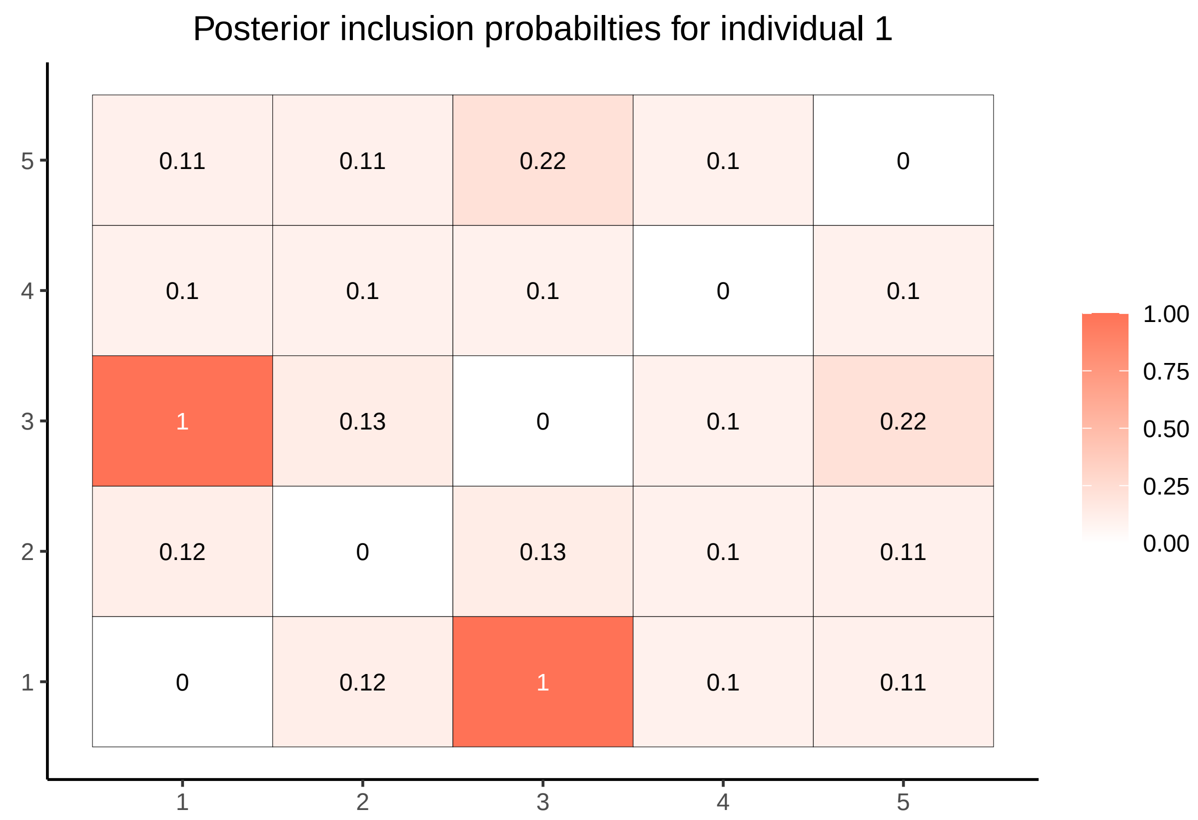
```
##
## [[1]]$ELBO_history
## NULL
##
## [[1]]$non_converged
## NULL
##
##
## [[2]]
## [[2]]$sigmabeta_sq
## [1] 0.129155
##
## [[2]]$pi
## [1] 0.3
##
## [[2]]$ELBO
## [1] -5978.314
##
## [[2]]$converged_iter
## [1] 56
##
## [[2]]$ELBO_history
## NULL
##
## [[2]]$non_converged
## NULL
##
##
## [[3]]
## [[3]]$sigmabeta_sq
## [1] 0.129155
##
## [[3]]$pi
## [1] 0.3
##
## [[3]]$ELBO
## [1] -4871.425
##
## [[3]]$converged_iter
## [1] 57
##
## [[3]]$ELBO_history
## NULL
##
## [[3]]$non_converged
## NULL
##
##
## [[4]]
## [[4]]$sigmabeta_sq
## [1] 1e-04
##
## [[4]]$pi
## [1] 0.1
```

```
##
## [[4]]$ELBO
## [1] -3968.535
##
## [[4]]$converged_iter
## [1] 3
##
## [[4]]$ELBO_history
## NULL
##
## [[4]]$non_converged
## NULL
##
##
## [[5]]
## [[5]]$sigmabeta_sq
## [1] 0.002154435
##
## [[5]]$pi
## [1] 0.3
##
## [[5]]$ELBO
## [1] -6046.502
##
## [[5]]$converged_iter
## [1] 10
##
## [[5]]$ELBO_history
## NULL
##
## [[5]]$non_converged
## NULL
```

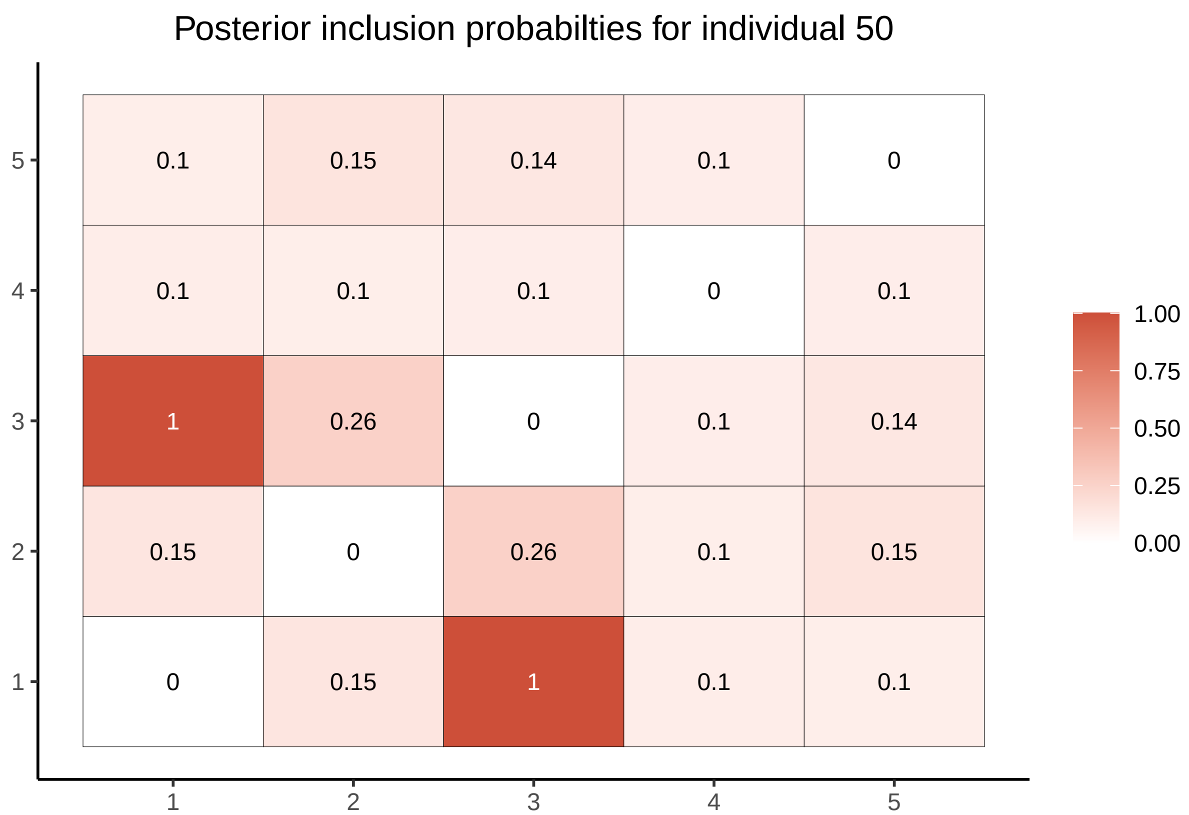
```
# individual-specific bandwidths calculated using KDE
out$bandwidths
```

```
##   [1] 0.8971926 0.8185741 0.8045396 0.7710095 0.7598766 0.7309506 0.6970164
##   [8] 0.6926737 0.6766866 0.6741037 0.6708585 0.6682201 0.6647626 0.6508583
##  [15] 0.6483599 0.6467354 0.6412826 0.6402868 0.6376642 0.6368427 0.6360554
##  [22] 0.6357334 0.6349457 0.6351369 0.6367247 0.6386644 0.6416763 0.6469182
##  [29] 0.6476813 0.6498723 0.6508363 0.6656945 0.6719934 0.6725067 0.6729458
##  [36] 0.6735493 0.6755900 0.6839099 0.7029164 0.7033434 0.7157298 0.7170107
##  [43] 0.7178789 0.7208353 0.7225332 0.7291790 0.7304082 0.7305625 0.7270552
##  [50] 0.7256076 0.7091386 0.7056539 0.7014279 0.6964823 0.6889947 0.6815783
##  [57] 0.6619228 0.6512742 0.6499261 0.6497488 0.6466575 0.6392574 0.6356924
##  [64] 0.6345264 0.6337265 0.6335805 0.6330526 0.6323850 0.6305934 0.6306508
##  [71] 0.6307874 0.6308795 0.6321204 0.6323676 0.6338561 0.6342392 0.6362190
##  [78] 0.6363787 0.6364964 0.6424749 0.6597773 0.6688075 0.6710351 0.6740264
##  [85] 0.6889789 0.6894551 0.7067720 0.7109540 0.7162684 0.7170686 0.7269144
##  [92] 0.7349568 0.7544093 0.7718605 0.7745784 0.7845610 0.7854562 0.8013423
##  [99] 0.8245164 0.8909799
```

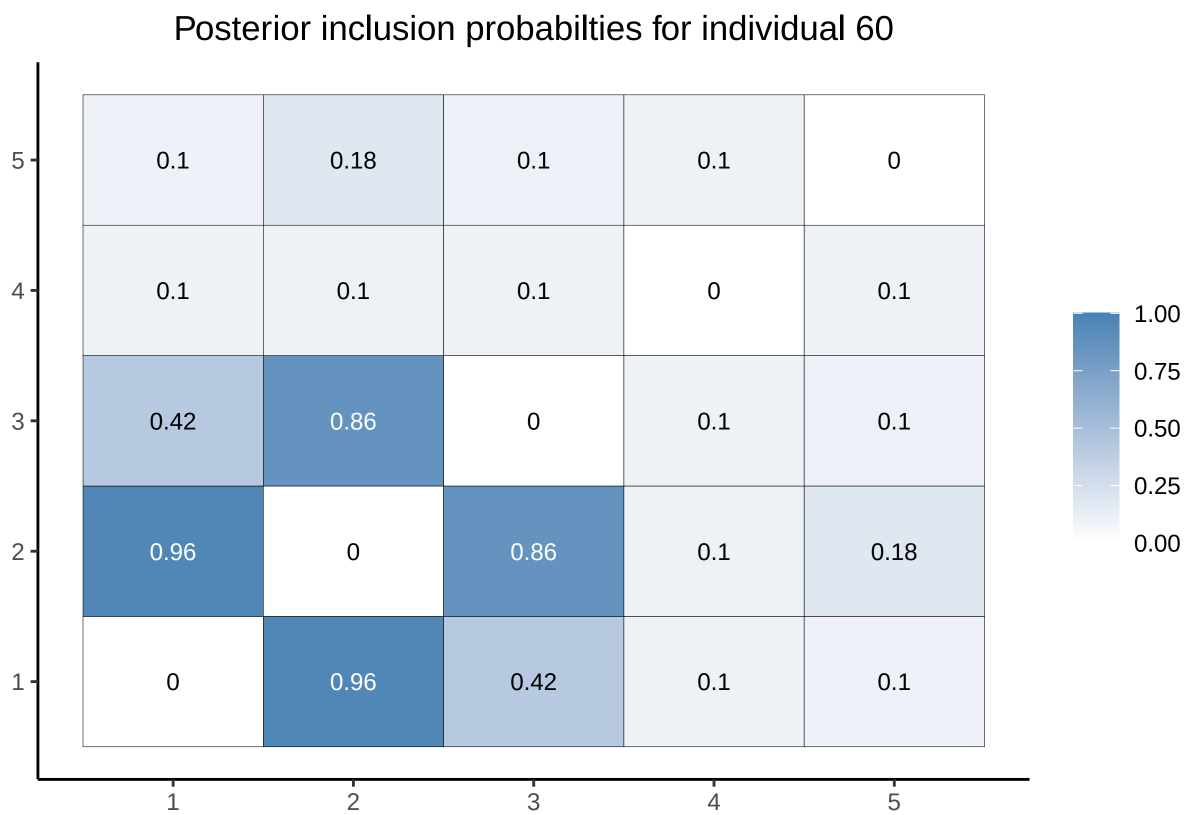
```r
# analyze results
gg_adjMat(out, 1, color1 = "coral1")
```



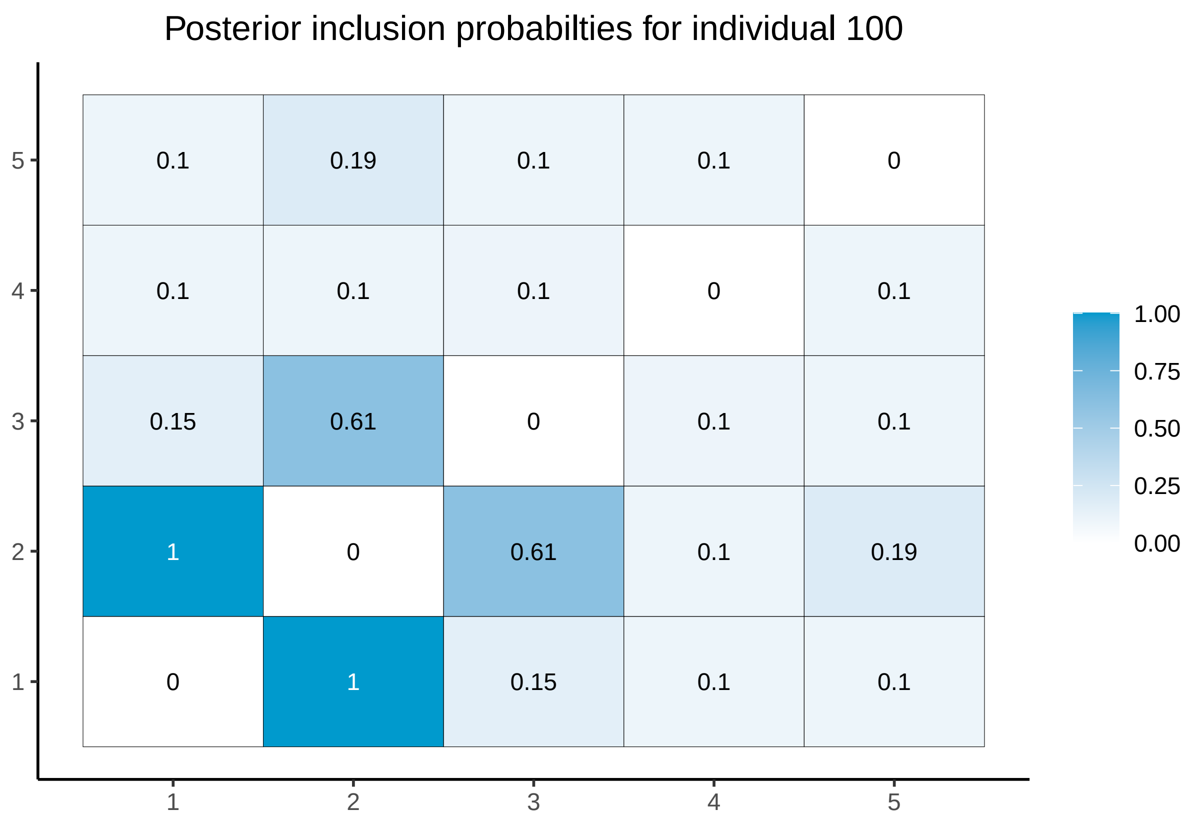

```r
gg_adjMat(out, 50, color1 = "tomato3")
```




```r
gg_adjMat(out, 60, color1 = "steelblue")
```

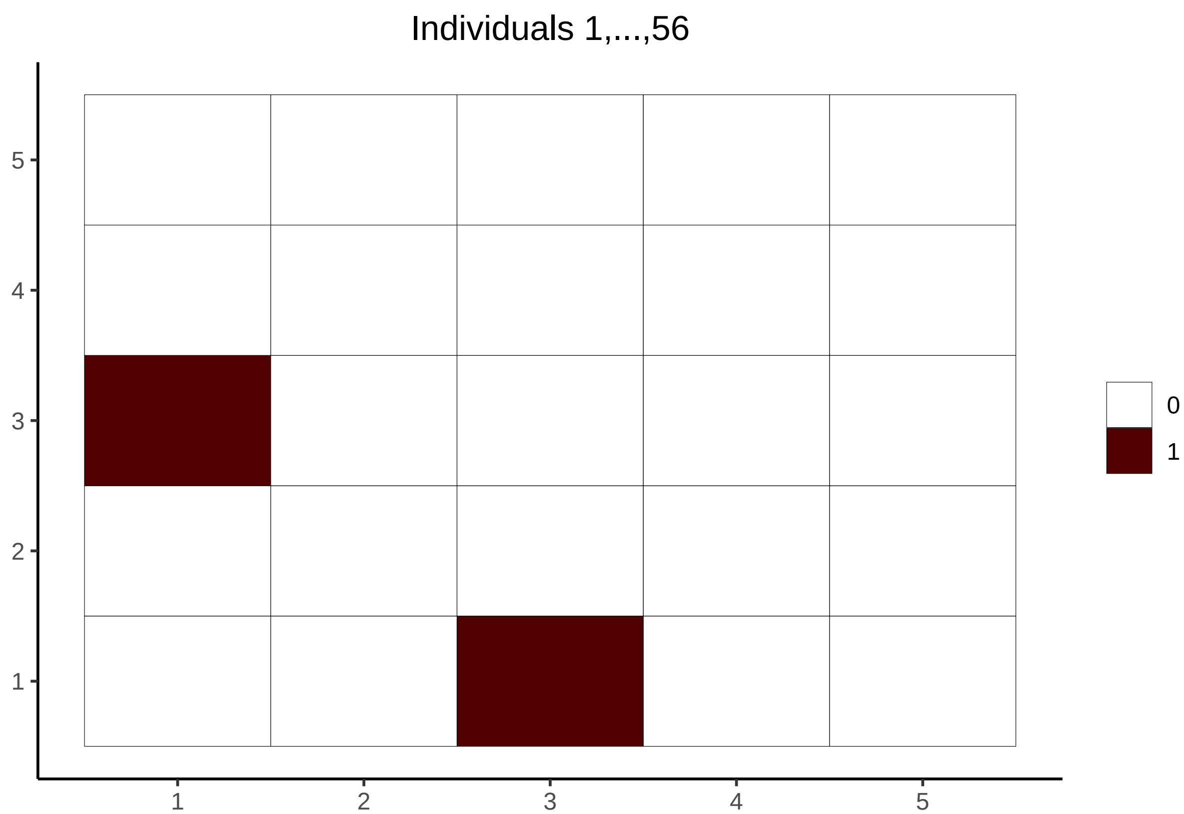



```
gg_adjMat(out, 100, color1 = "deepskyblue3")
```

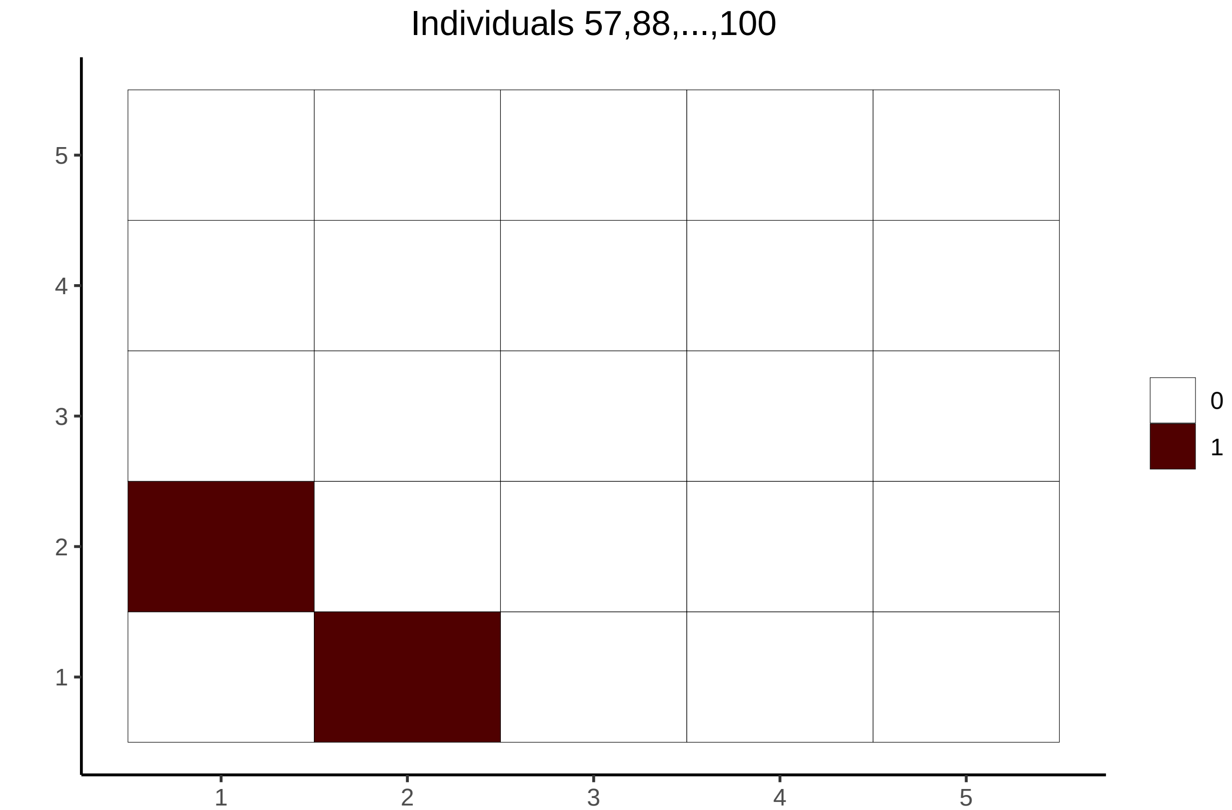

Posterior inclusion probabilties for individual 100
0.1
0.19
0.1
0.1
0
0.1
0.1
0.1
0
0.1
0.15
0.61
0
0.1
0.1
1
0
0.61
0.1
0.19
0
1
0.15
0.1
0.1
1.00
0.75
0.50
0.25
0.00
1
2
3
4
5

```
plot(out)
```

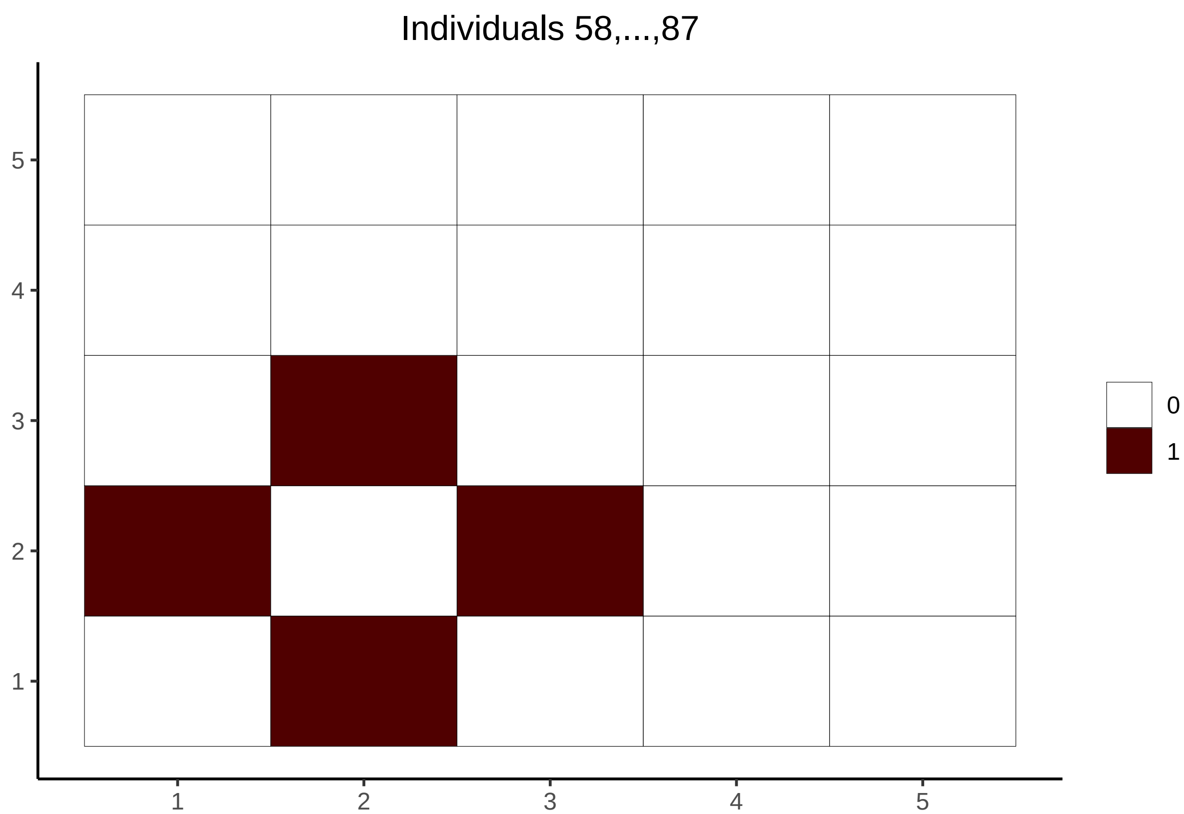
```
## [[1]]
```



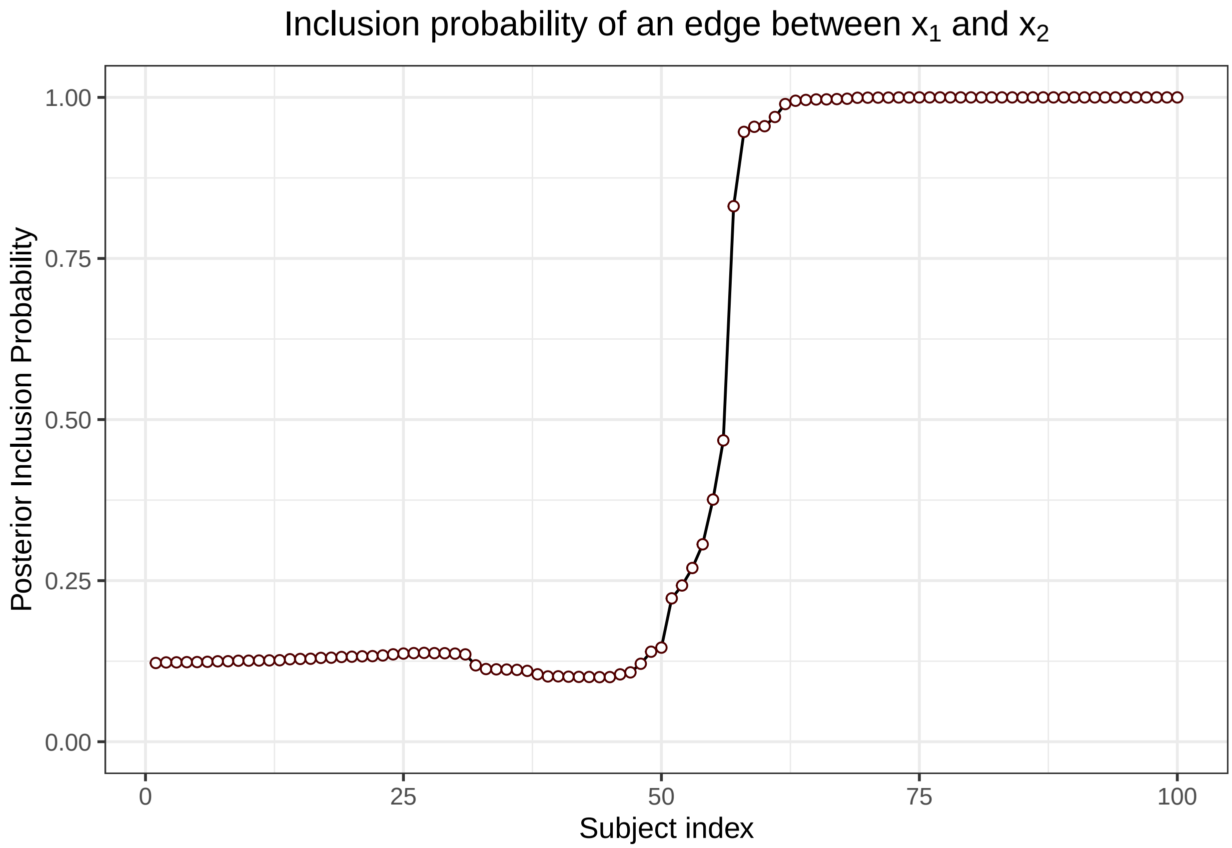

```
##
## [[2]]
```



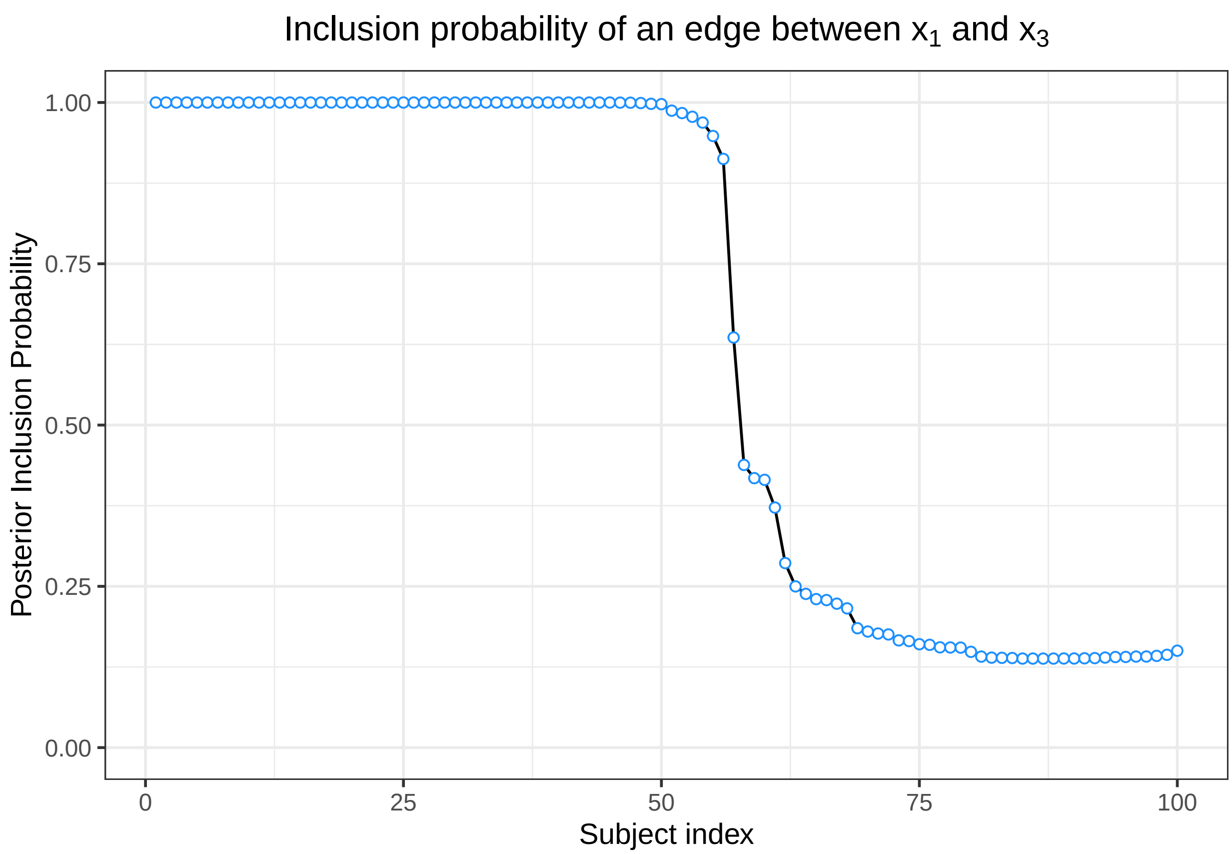

```
##
## [[3]]
```




```
?gg_inclusionCurve
gg_inclusionCurve(out, 1, 2)
```




```
gg_inclusionCurve(out, 1, 3, point_color = "dodgerblue")
```




```
# find sensitivity, specificity, and accuracy

# true positives
TP_neg <- sum(sapply(out$graphs[1:(n / 2)],
                     function(graph) sum(graph == 1 & true_graph_neg == 1)))
TP_pos <- sum(sapply(out$graphs[(n / 2 + 1):n],
                     function(graph) sum(graph == 1 & true_graph_pos == 1)))
TP <- TP_neg + TP_pos

# total positives
num_pos <- sum(true_graph_pos) * n / 2 + sum(true_graph_neg) * n / 2

# true negatives
TN_neg <- sum(sapply(out$graphs[1:(n / 2)],
                     function(graph) sum(graph == 0 & true_graph_neg == 0)))
TN_pos <- sum(sapply(out$graphs[(n / 2 + 1):n],
                     function(graph) sum(graph == 0 & true_graph_pos == 0)))
TN <- TN_neg + TN_pos

# total negatives
num_neg <- length(true_graph_pos) * n - num_pos

(sensitivity <- TP / num_pos)
```

```
## [1] 0.94
```

```r
(specificity <- TN / num_neg)
```

```
## [1] 0.9686957
```

```r
(accuracy <- (TN + TP) / (num_pos + num_neg))
```

```
## [1] 0.9664
```