Optimal Pi and Importance Sampling Analysis

Contents

Overview

In this document, I present the results of two experiments. The first describes the distribution of the optimal hyperparameters selected by grid search. In the second, I compare the results of a novel importance sampling - grid search hybrid scheme for hyperparameter specification.

The first two main sections contain the results of these experiments, while the third describes the data and extraneous covariate generation.

Optimal Pi

Hyperparameter specification

In this experiment, I applied the grid search algorithm in 100 trials. I generated the grid from the cartesian product of the following marginal grids:

```
# marginal grids
marg_grids
```

```
## $pip
##  [1] 0.00001 0.00010 0.00100 0.01000 0.05000 0.10000 0.15000 0.20000 0.25000
## [10] 0.35000 0.45000 0.55000 0.65000 0.75000 0.85000 0.95000 0.99000 0.99900
## [19] 0.99990 0.99999
##
## $ssq
## [1] 0.001 0.100 0.250 0.500 0.750 1.000 3.000
##
## $sbsq
##  [1] 1.0e-05 1.0e-03 1.0e-02 1.0e-01 2.5e-01 5.0e-01 1.0e+00 3.0e+00 5.0e+00
## [10] 1.0e+01
```

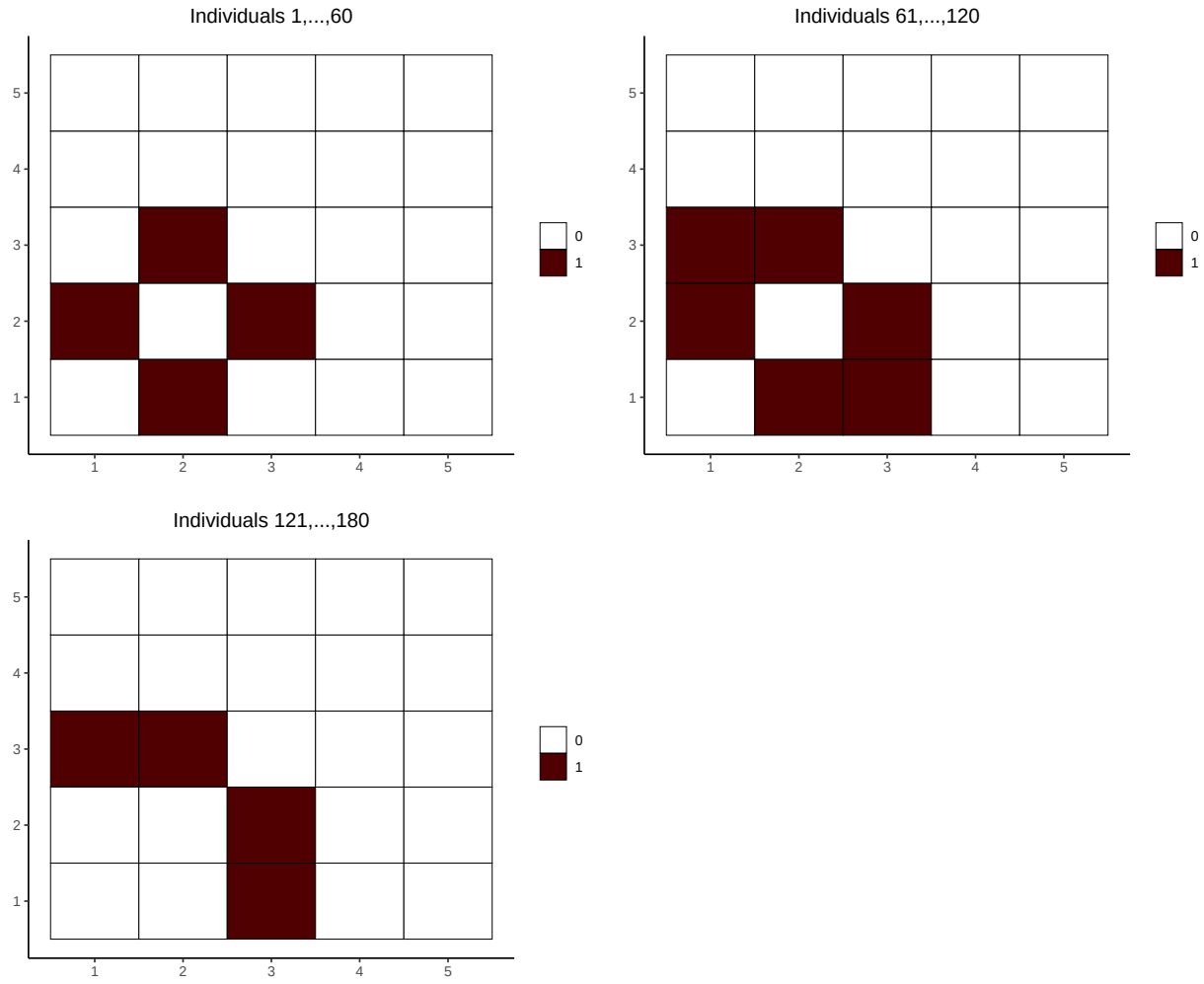
```
# number of grid points in the cartesian product
nrow(expand.grid(marg_grids))
```

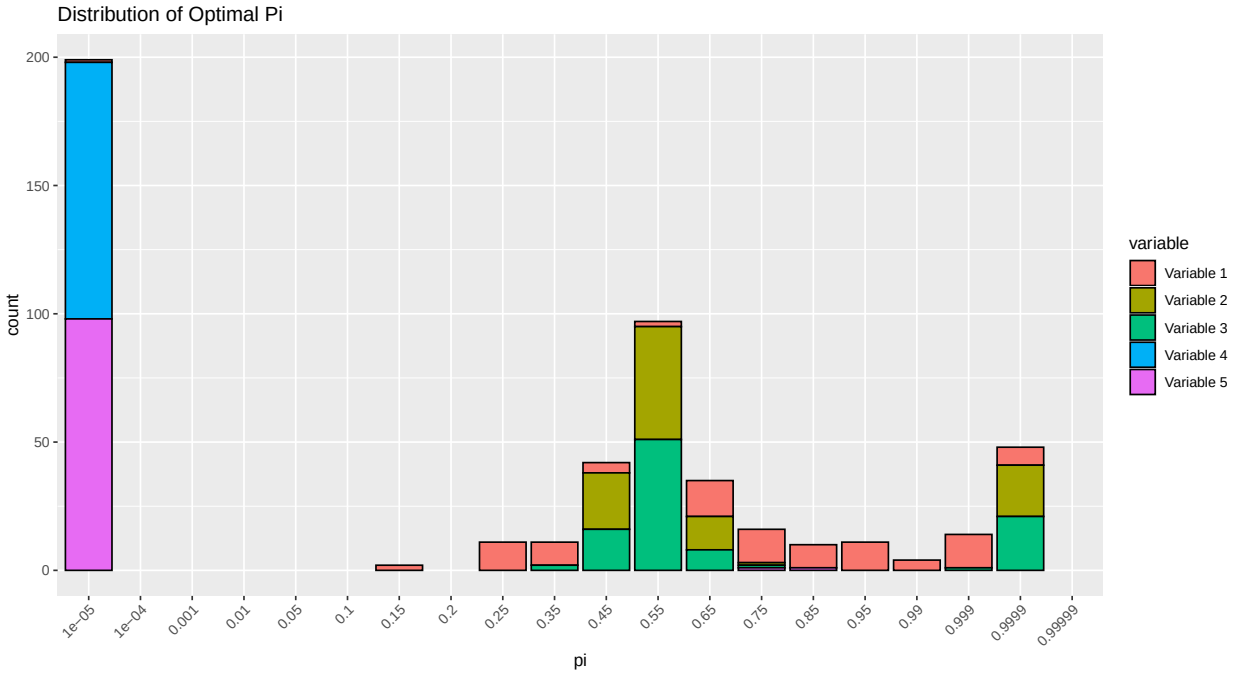
```
## [1] 1400
```

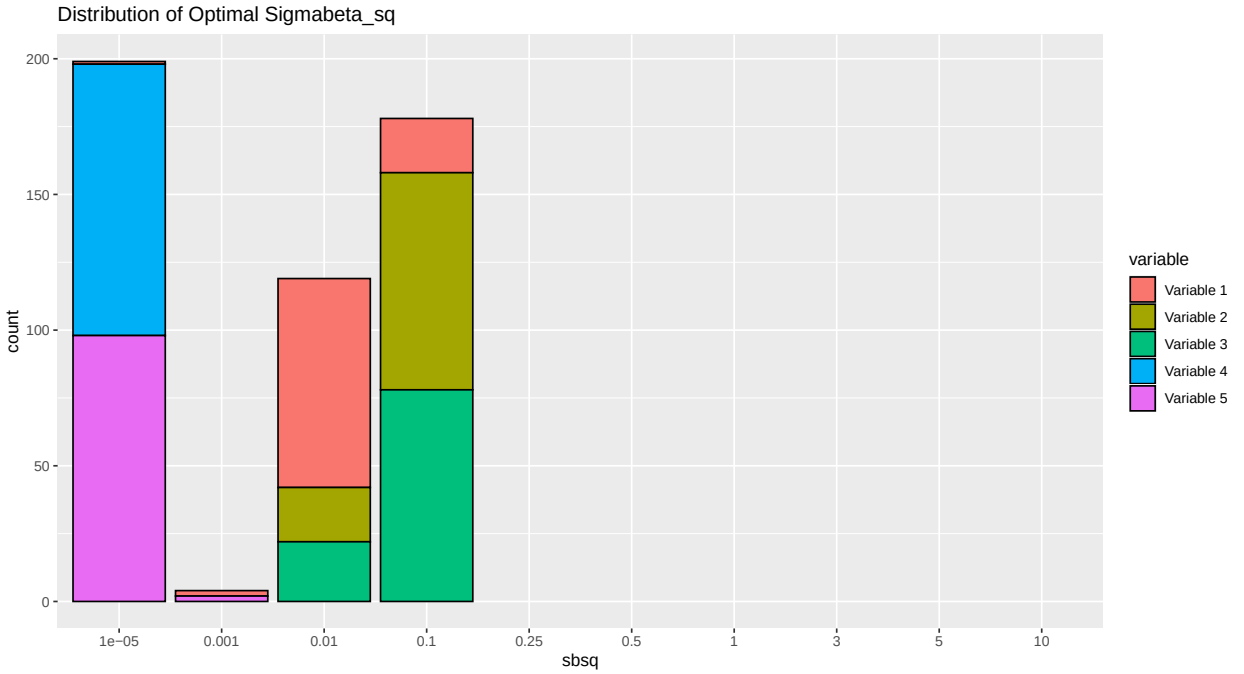
Optimal Hyperparameter Distribution

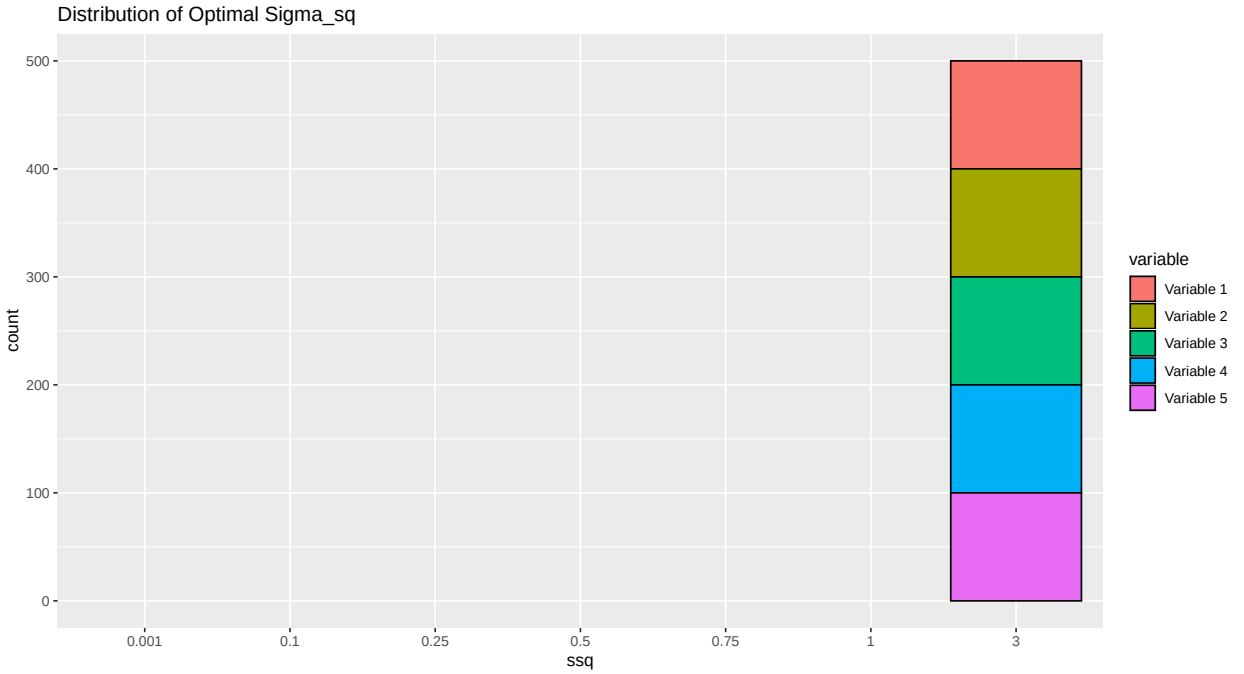
In this section, I display the distribution of optimal hyperparameters aggregated across all variables. Since a unique point in the hyperparameter grid was selected for each of the 5 variables, a total of 500 grid points are represented in each figure.

I begin by displaying the true underlying precision structures to provide context for the optimal pi.


Individuals 1,...,60
Individuals 61,...,120
Individuals 121,...,180
0
1
1
2
3
4
5


Distribution of Optimal Pi
count
pi
variable
Variable 1
Variable 2
Variable 3
Variable 4
Variable 5
0
50
100
150
200
1e−05
1e−04
0.001
0.01
0.05
0.1
0.15
0.2
0.25
0.35
0.45
0.55
0.65
0.75
0.85
0.95
0.99
0.999
0.9999
0.99999



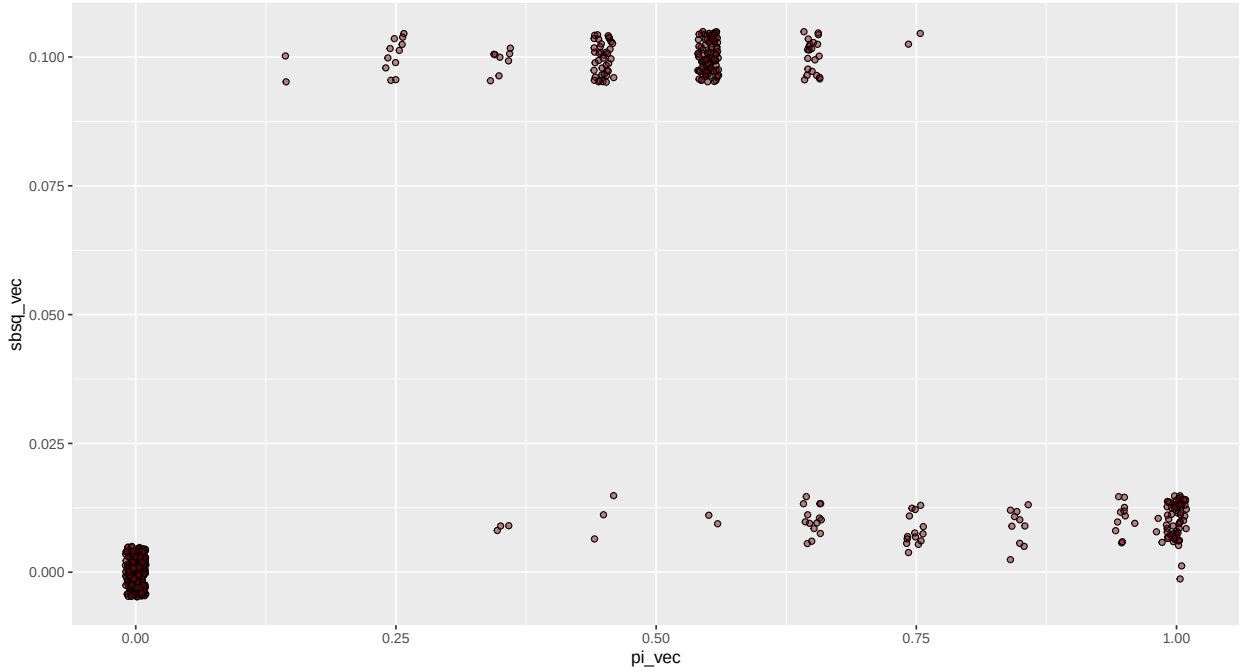


Optimal Sigmabeta_sq as a function of Pi

Here, I visualize the optimal sigmabeta_sq as a function of pi. Note that a small amount of jitter has been added to help distinguish the clusters.

```
##
##          1e-05 0.001 0.01 0.1 0.25 0.5   1   3   5  10
##   1e-05    199     0    0   0    0   0   0   0   0   0
##   1e-04      0     0    0   0    0   0   0   0   0   0
```

```
##    0.001       0     0     0    0     0    0    0    0    0    0
##    0.01        0     0     0    0     0    0    0    0    0    0
##    0.05        0     0     0    0     0    0    0    0    0    0
##    0.1         0     0     0    0     0    0    0    0    0    0
##    0.15        0     0     0    2     0    0    0    0    0    0
##    0.2         0     0     0    0     0    0    0    0    0    0
##    0.25        0     0     0   11     0    0    0    0    0    0
##    0.35        0     0     3    8     0    0    0    0    0    0
##    0.45        0     0     3   39     0    0    0    0    0    0
##    0.55        0     0     2   95     0    0    0    0    0    0
##    0.65        0     0    14   21     0    0    0    0    0    0
##    0.75        0     1    13    2     0    0    0    0    0    0
##    0.85        0     1     9    0     0    0    0    0    0    0
##    0.95        0     0    11    0     0    0    0    0    0    0
##    0.99        0     0     4    0     0    0    0    0    0    0
##    0.999       0     2    12    0     0    0    0    0    0    0
##    0.9999      0     0    48    0     0    0    0    0    0    0
##    0.99999     0     0     0    0     0    0    0    0    0    0
```

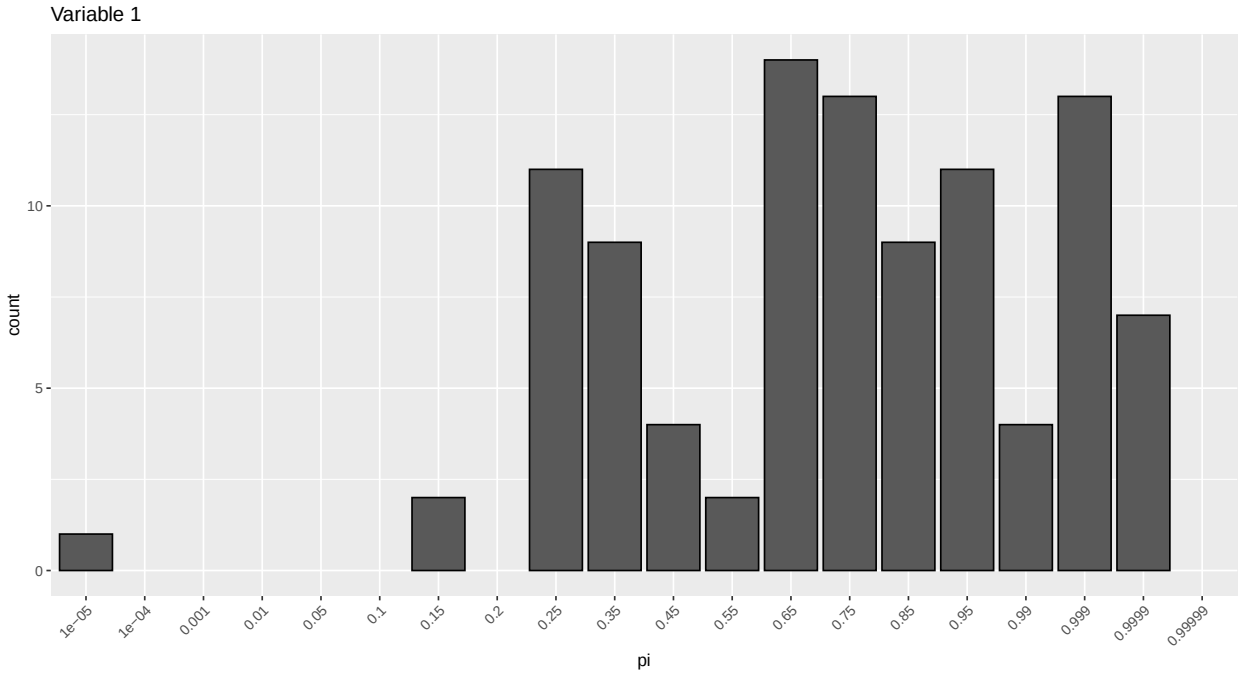


Optimal Pi by variable

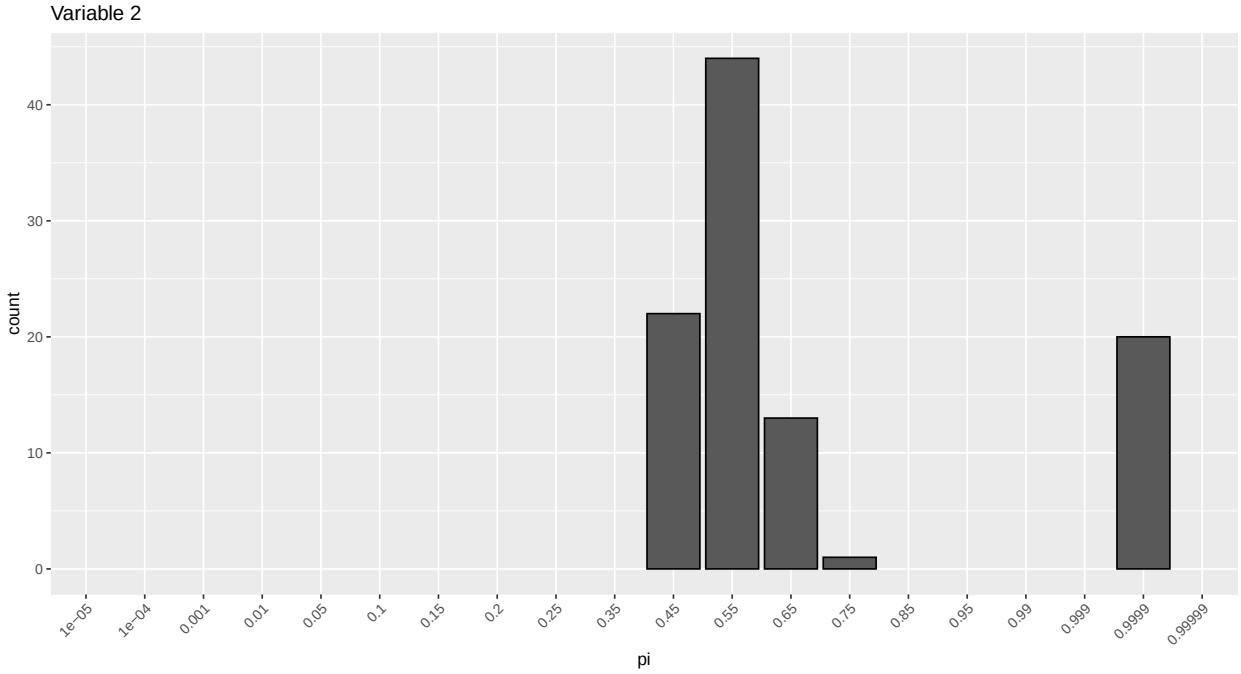
In this section, I visualize the optimal pi by variable.

```
# visualize the optimal pi by variable
var_names <- names(grid_hyp[[1]])
lapply(var_names, function(var_name) ggplot() +
         geom_bar(aes(factor(sapply(
           pi_opt, `[[`, var_name), levels = marg_grids$pip)), color = "black") +
         scale_x_discrete(drop = F) + xlab("pi") + ggtitle(var_name) +
         theme(axis.text.x = element_text(angle = 45, hjust=1)))
```

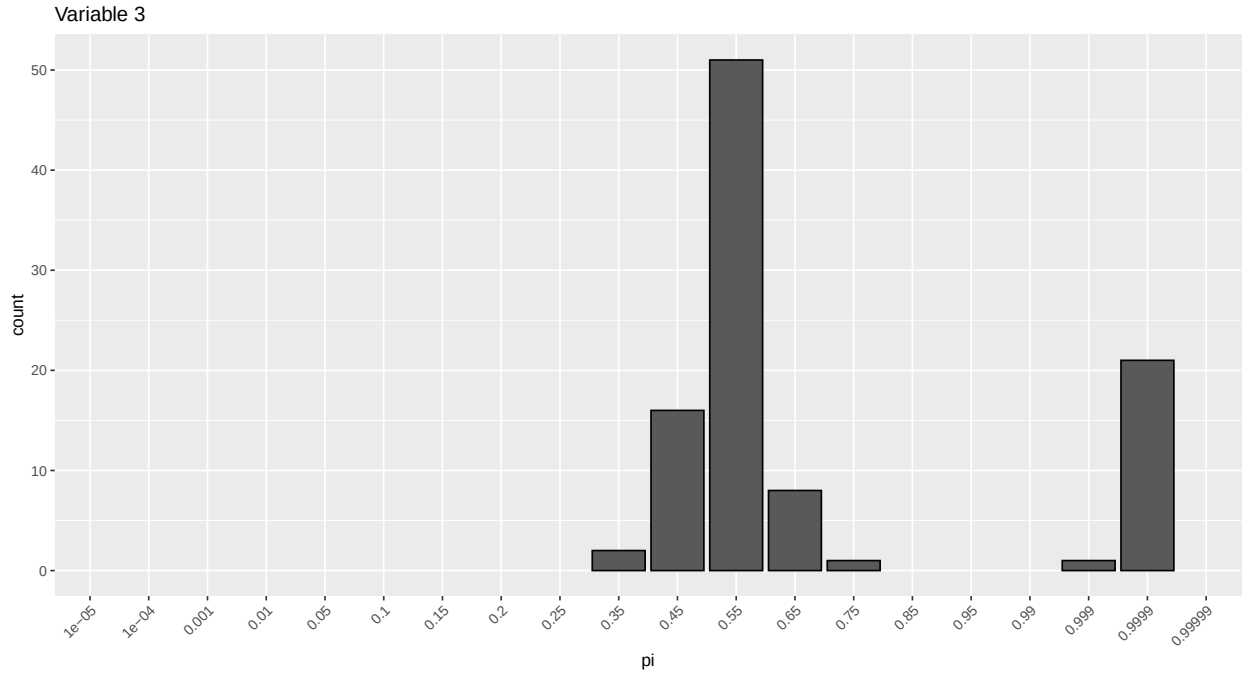
```
## [[1]]
```


Variable 1
count
0
5
10
1e−05
1e−04
0.001
0.01
0.05
0.1
0.15
0.2
0.25
0.35
0.45
0.55
0.65
0.75
0.85
0.95
0.99
0.999
0.9999
0.99999
pi

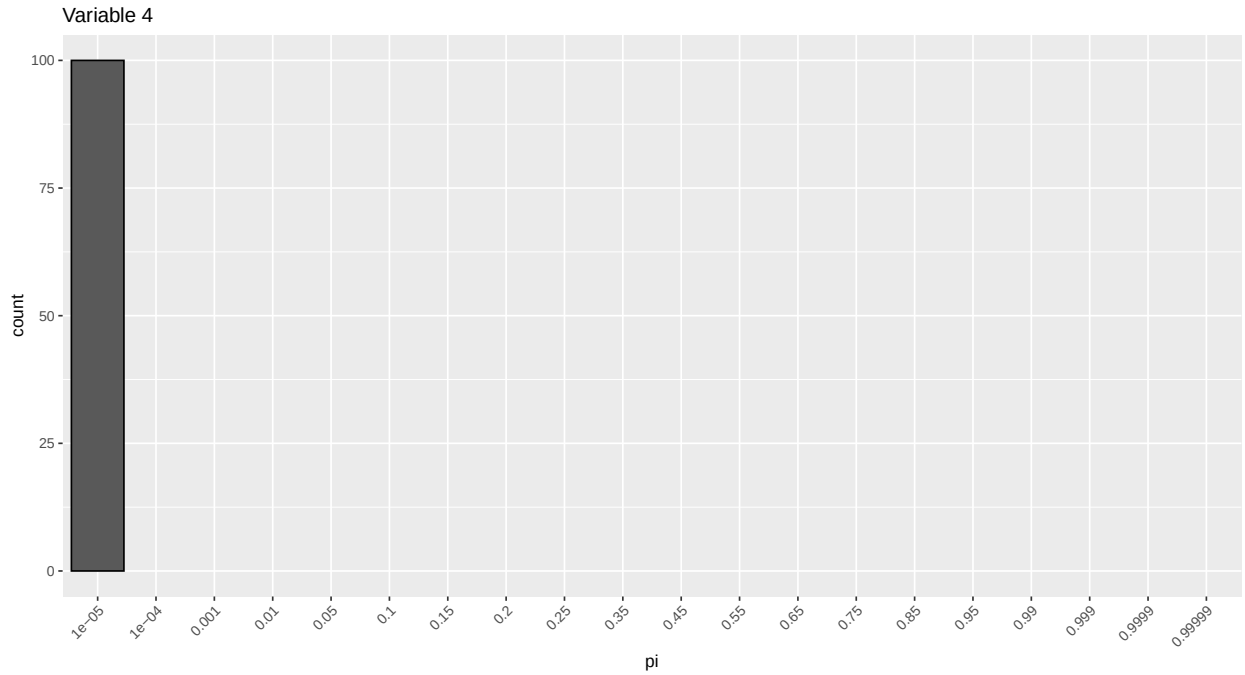
```
##
## [[2]]
```


Variable 2
count
0
10
20
30
40
1e−05
1e−04
0.001
0.01
0.05
0.1
0.15
0.2
0.25
0.35
0.45
0.55
0.65
0.75
0.85
0.95
0.99
0.999
0.9999
0.99999
pi

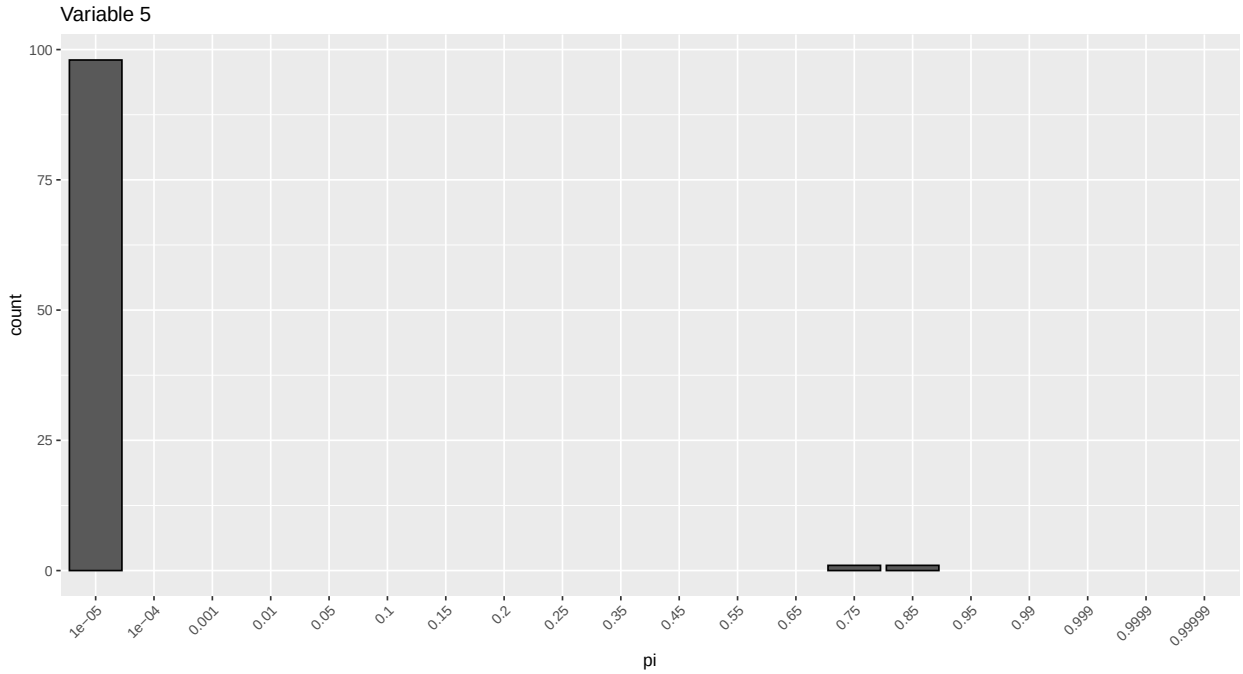
```
##
## [[3]]
```


Variable 3
count
pi
0
10
20
30
40
50
1e−05
1e−04
0.001
0.01
0.05
0.1
0.15
0.2
0.25
0.35
0.45
0.55
0.65
0.75
0.85
0.95
0.99
0.999
0.9999
0.99999

##
[[4]]


Variable 4
count
pi
0
25
50
75
100
1e−05
1e−04
0.001
0.01
0.05
0.1
0.15
0.2
0.25
0.35
0.45
0.55
0.65
0.75
0.85
0.95
0.99
0.999
0.9999
0.99999

##
[[5]]



Importance Sampling Hybrid

Algorithm Overview

This algorithm averages over a deterministic grid of pi. For each value of pi, the values of sigmasq and sigmasq_beta are chosen using grid search. The importance weights are calculated by assigning a uniform prior to each value of pi and approximating the marginal likelihood using the ELBO.

Hyperparameter Grid

Below, I display the grid of pi that are averaged over, as well as the marginal grids for sigmasq and sigma-beta_sq.

```
## $pip
##  [1] 0.00001 0.00010 0.00100 0.01000 0.02500 0.05000 0.10000 0.15000 0.20000
## [10] 0.25000 0.30000 0.35000 0.40000 0.45000 0.50000
##
## $ssq
##  [1] 0.001 0.010 0.050 0.100 0.250 0.500 0.750 1.000 2.000 3.000
##
## $sbsq
##  [1] 1.0e-05 1.0e-03 1.0e-02 1.0e-01 2.5e-01 5.0e-01 1.0e+00 3.0e+00 5.0e+00
## [10] 1.0e+01
```

Sensitivity and Specificity

In this section, I compare the sensitivity and the specificity for the two methods.

```r
# grid sensitivity
summary(grid_sens)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.4286  0.7667  0.8024  0.8020  0.8524  1.0000
```

```r
# importance sensitivity
summary(impt_sens)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.4048  0.6464  0.7048  0.6909  0.7530  0.8286
```

```r
# difference in the sensitivities
summary(grid_sens - impt_sens)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.06488 0.10595 0.11114 0.14167 0.29048
```

```r
# grid specificity
summary(grid_spec)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.9672  1.0000  1.0000  0.9992  1.0000  1.0000
```

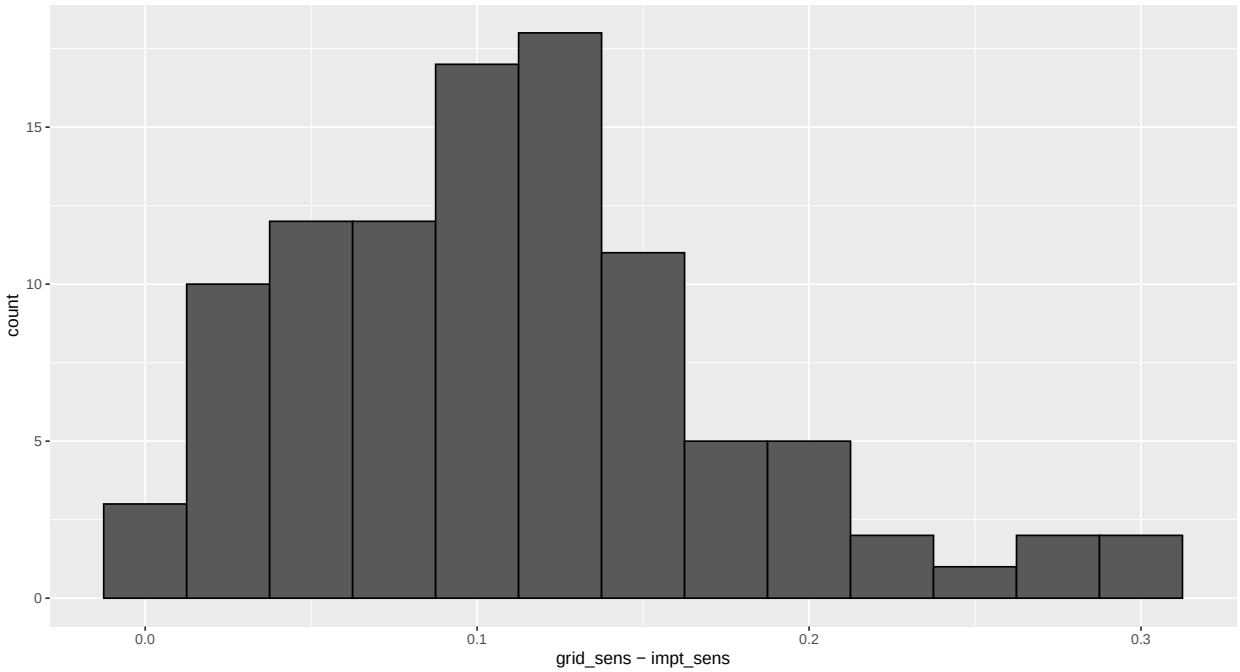
```r
# importance specificity
summary(impt_spec)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.9956  1.0000  1.0000  1.0000  1.0000  1.0000
```

```r
# difference in the specificities
summary(grid_spec - impt_spec)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -0.0284153  0.0000000  0.0000000 -0.0008033  0.0000000  0.0000000
```

```r
# visualize the difference in the sensitivity
ggplot() + geom_histogram(aes(grid_sens - impt_sens), binwidth = 0.025, color = "black")
```



Case Study

Here, I present the graphs estimated by the two methods in two cases; the first case represents the performance of the methods in the trial wherein the hybrid method had its maximum sensitivity. The second case represents the performance of the methods in the trial where the difference in the sensitivity of the two methods was maximal.

I also display the importance weights for variables 2 and 3 for individual 75 in both cases.

Case 1: Optimal Importance Performance

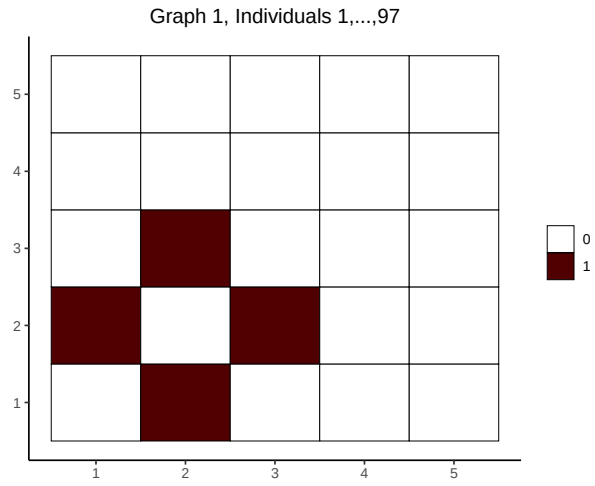
```
good_ind <- which.max(impt_sens)

# sensitivity for grid search
grid_sens[good_ind]
```

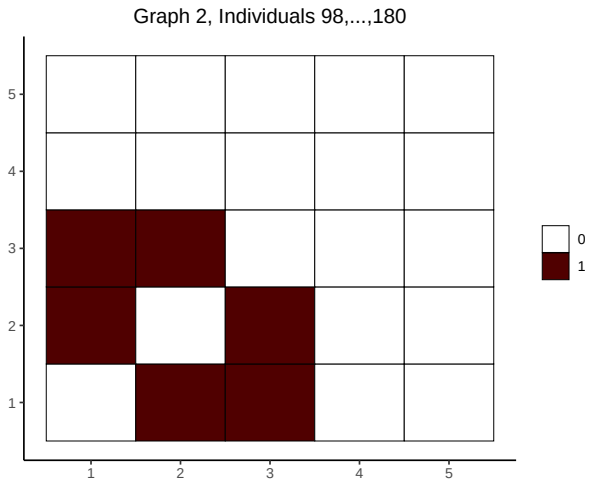
```
## [1] 0.9119048
```

```
# sensitivity for importance sampling
impt_sens[good_ind]
```

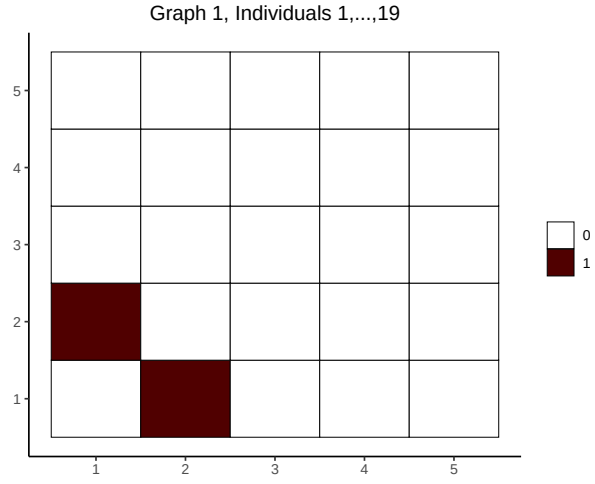
```
## [1] 0.8285714
```

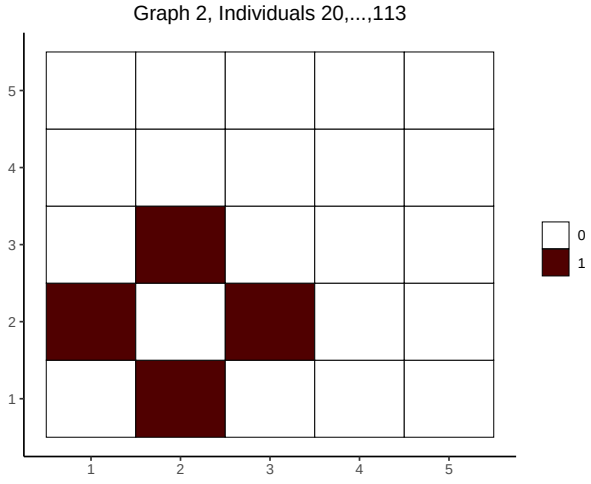
```
# graphs estimated by grid search
ggarrange(plotlist = plot(res[[good_ind]]$grid))
```

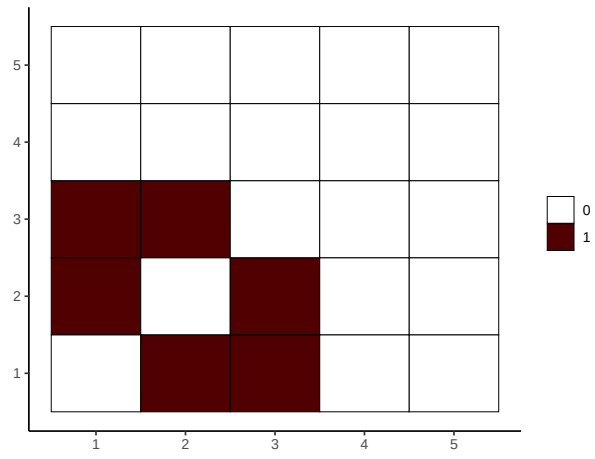


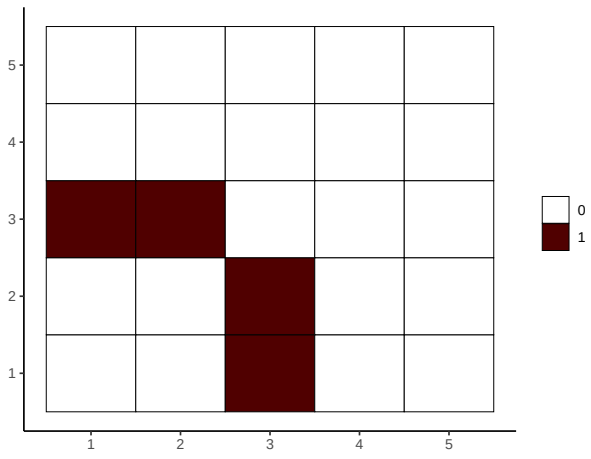



```
# graphs estimated by importance sampling
ggarrange(plotlist = plot(res[[good_ind]]$impt))
```









```
# visualize the weights for individual 75

# variable 2
res[[good_ind]]$impt$hyperparameters$`Variable 2`$weights[75, ]
```

Importance Weights

```
##           1e-05        1e-04        0.001         0.01        0.025         0.05
## 75 0.0006826342 0.0006928421 0.0007941974 0.001860554 0.004029789 0.008744038
##            0.1         0.15          0.2         0.25          0.3         0.35          0.4
## 75 0.02178456 0.03717867 0.05716057 0.07981907 0.1044763 0.1304914 0.1572595
##          0.45          0.5
## 75 0.1842117 0.2108142
```

```
# variable 3
res[[good_ind]]$impt$hyperparameters$`Variable 3`$weights[75, ]
```

```
##          1e-05       1e-04       0.001        0.01       0.025        0.05         0.1
## 75 0.04026568 0.04028912 0.04052297 0.04279559 0.046331 0.05157686 0.06568873
##           0.15         0.2        0.25         0.3        0.35         0.4        0.45
## 75 0.07429845 0.08060657 0.08487363 0.08733789 0.08822107 0.08773092 0.08606252
##            0.5
## 75 0.08339901
```

Case 2: Greatest Grid Search Margin

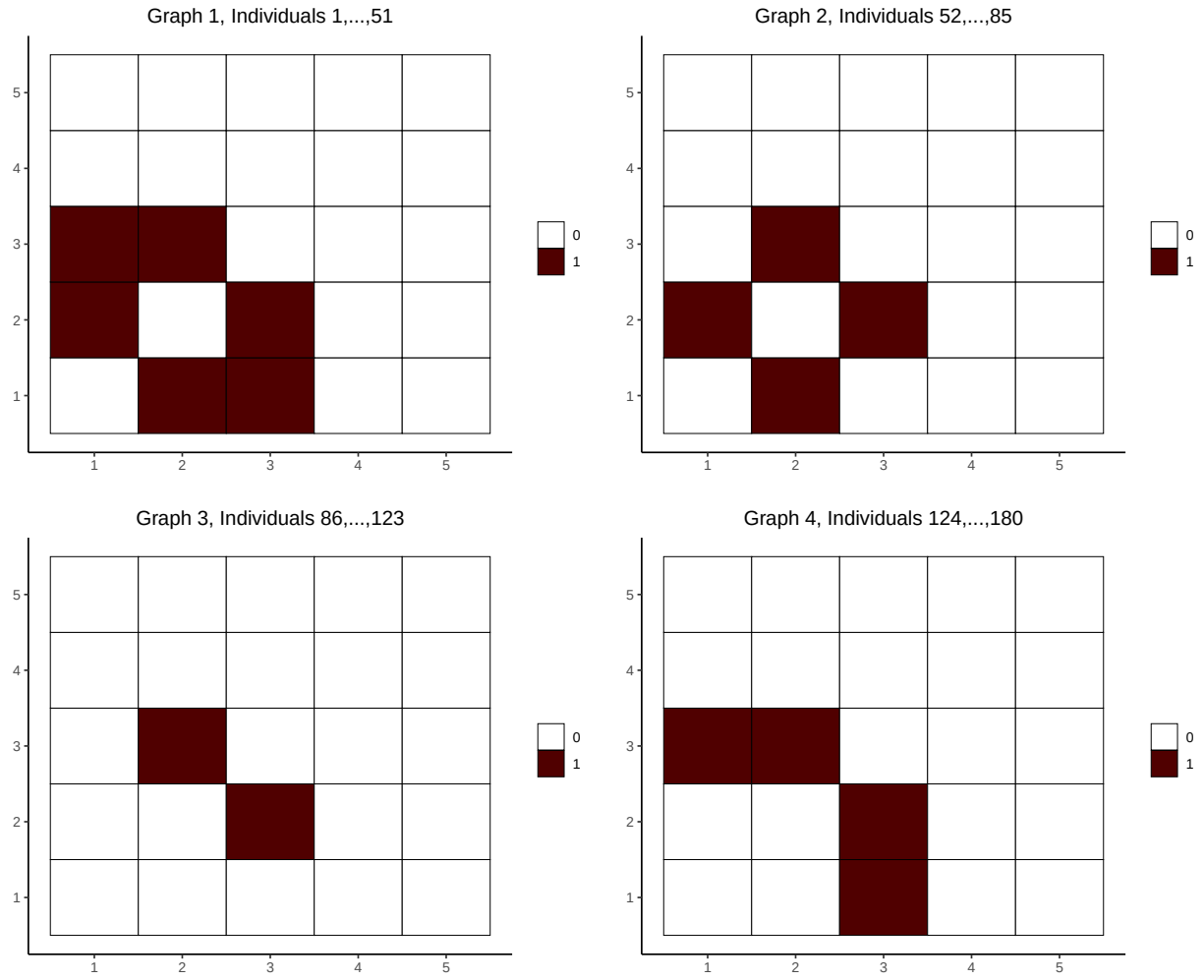
```
diff_ind <- which.max(grid_sens - impt_sens)

# sensitivity for grid search
grid_sens[diff_ind]
```

```
## [1] 0.7666667
```

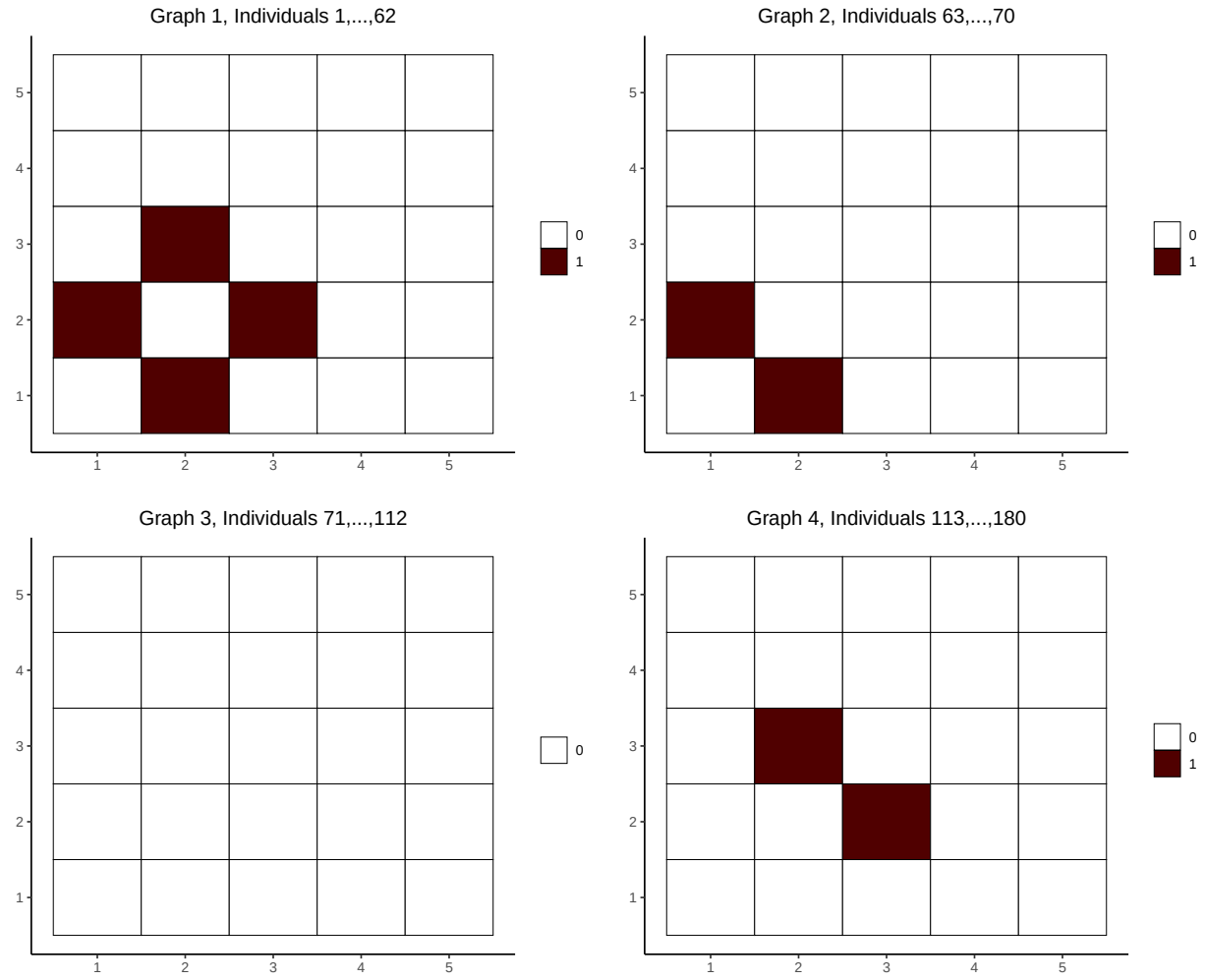
```
# sensitivity for importance sampling
impt_sens[diff_ind]
```

```
## [1] 0.4761905
```

```
# graphs estimated by grid search
ggarrange(plotlist = plot(res[[diff_ind]]$grid))
```




```
# graphs estimated by importance sampling
ggarrange(plotlist = plot(res[[diff_ind]]$impt))
```




```
# visualize the weights for individual 75

# variable 2
res[[diff_ind]]$impt$hyperparameters$`Variable 2`$weights[75, ]
```

Importance Weights

```
##         1e-05     1e-04      0.001       0.01      0.025       0.05        0.1
## 75 0.03333854 0.0333482 0.03344454 0.03518008 0.03783371 0.04200957 0.04959953
##          0.15        0.2       0.25        0.3       0.35        0.4       0.45
## 75 0.06350329 0.07259969 0.08123683 0.08939137 0.09703646 0.1041447 0.1106892
##          0.5
## 75 0.1166443
```

```
# variable 3
res[[diff_ind]]$impt$hyperparameters$`Variable 3`$weights[75, ]
```

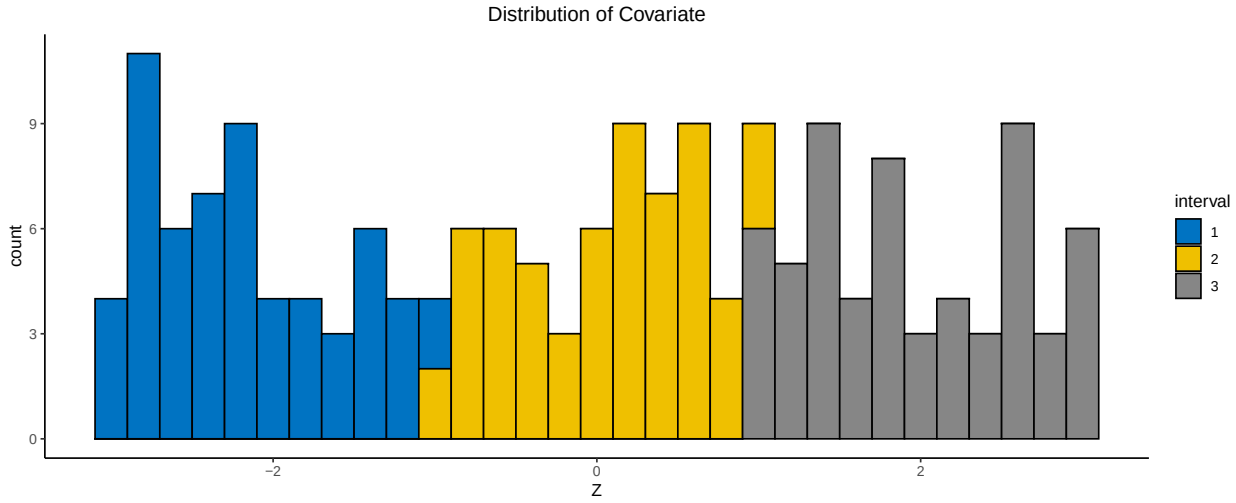
```
##         1e-05      1e-04      0.001       0.01      0.025       0.05        0.1
```

```
## 75 0.07183322 0.07183075 0.07180612 0.07154967 0.07108265 0.0702013 0.06809599
##          0.15        0.2      0.25        0.3       0.35        0.4       0.45
## 75 0.06560454 0.06280058 0.0597483 0.05650464 0.05312073 0.04964283 0.07833756
##           0.5
## 75 0.07784113
```

Data Generation

Extraneous Covariate

I generated the covariate, Z, as the union of three almost disjoint intervals of equal measure. That is, $Z = Z_1 \cup Z_2 \cup Z_3$ with $Z_1 = (-3, -1), Z_2 = (a, b) = (-1, 1), Z_3 = (1, 3)$. Within each interval, I generated 60 covariate values from a uniform distribution. For example:



Precision Matrix

All of the individuals in interval 1 had the same precision matrix, $\Omega^{(1)}$:

$$\Omega^{(1)}_{i,j} = \begin{cases} 2 & i = j \\ 1 & (i,j) \in \{(1,2),(2,1),(2,3),(3,2)\} \\ 0 & o.w. \end{cases}$$

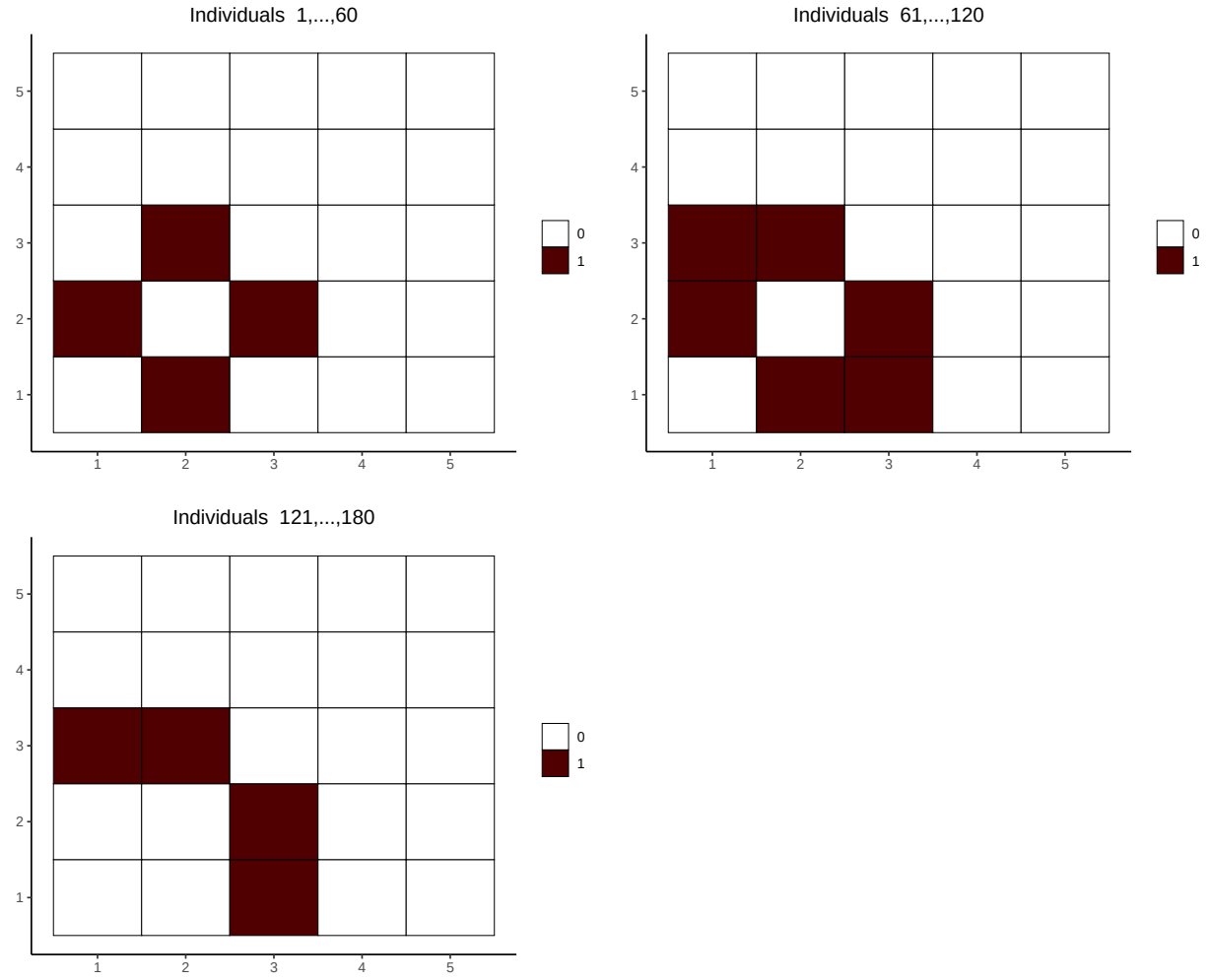
Also, all of the individuals in interval 3 had the same precision matrix, $\Omega^{(3)}$:

$$\Omega^{(3)}_{i,j} = \begin{cases} 2 & i = j \\ 1 & (i,j) \in \{(1,3),(3,1),(2,3),(3,2)\} \\ 0 & o.w. \end{cases}$$

However, the individuals in interval 2 had a precision matrix that was dependent upon Z and (a, b). Let $\beta_0 = -a/(b-a)$ and $\beta_1 = 1/(b-a)$. Then:

$$\Omega^{(2)}_{i,j}(z) = \begin{cases} 2 & i = j \\ 1 & (i,j) \in \{(2,3),(3,2)\} \\ 1 - \beta_0 - \beta_1 z & (i,j) \in \{(1,2),(2,1)\} \\ \beta_0 + \beta_1 z & (i,j) \in \{(1,3),(3,1)\} \\ 0 & o.w. \end{cases}$$

Thus, $\Omega^{(2)}(a) = \Omega^{(1)}$ and $\Omega^{(2)}(b) = \Omega^{(3)}$. That is, an individual on the left or right boundary of Z_2 would have precision matrix $\Omega^{(1)}$ or $\Omega^{(3)}$, respectively. The conditional dependence structures corresponding to each of these precision matrices are visualized below.



Data matrix

Let z_l be the extraneous covariate for the l-th individual. To generate the data matrix for the l-th individual, I took a random sample from $\mathcal{N}(0, \{\Omega_l(z_l)\}^{-1})$, where:

$$\Omega_l(z_l) = \begin{cases} \Omega^{(1)} & z_l \in Z_1 \\ \Omega^{(2)}(z_l) & z_l \in Z_2 \\ \Omega^{(3)} & z_l \in Z_3 \end{cases}$$