Simulation Example: Placing Prior on Rho

In this RMD file, we reproduce the simulation results discussed when placing a penalised complexity prior on ρ in Section 4.2.

1 Packages and Data Setup

```
library(data.table)
library(Rcpp)
library(RcppArmadillo)
library(ggplot2)
rm(list = ls())
```

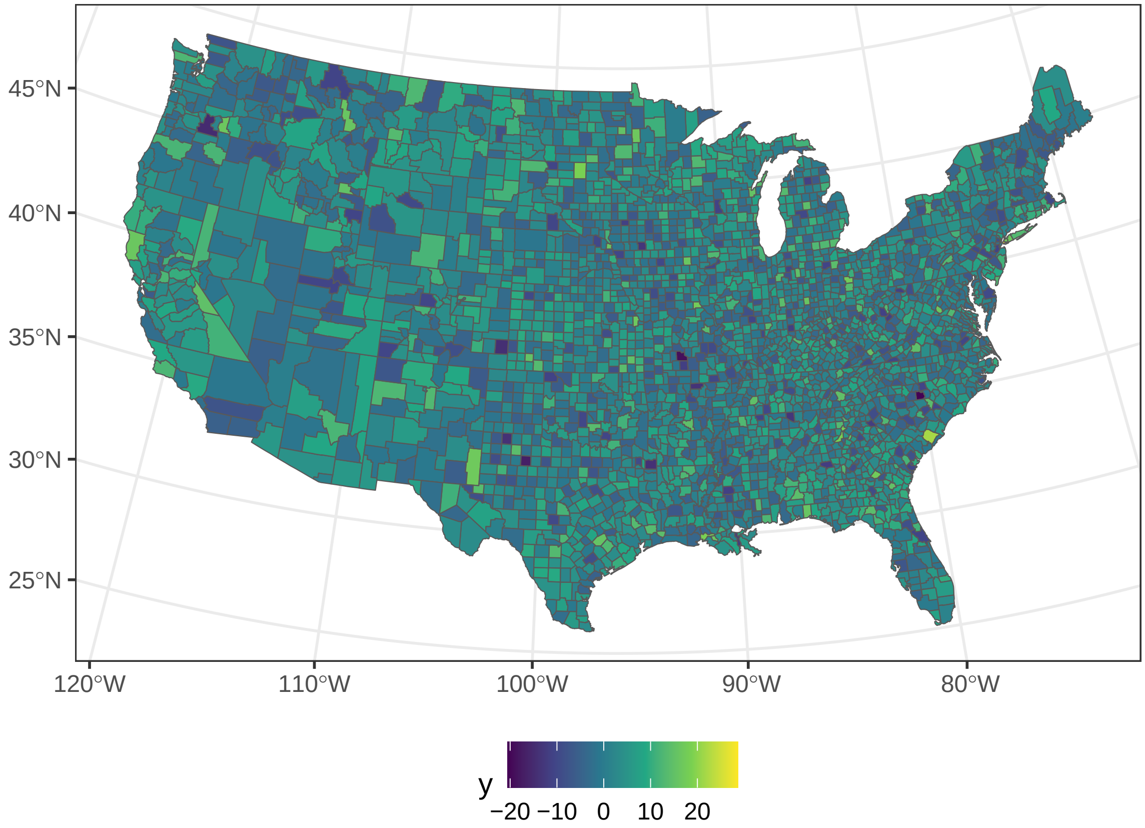
Generate simulation data and load in helper functions:

```
# Data generation
source(file.path(getwd(), "src", "R", "simulation", "US_data_generation.R"))
# Helper functions to compute posterior probabilities v_ij
source(file.path(getwd(), "src", "R", "simulation", "simulation_helper.R"))
source(file.path(getwd(), "src", "R", "eps_loss_FDR.R"))
source(file.path(getwd(), "src", "R", "vij_computation.R"))
# Gibbs sampler
Rcpp::sourceCpp(file.path(getwd(), "src", "rcpp", "BYM2_flatbeta_MCMC.cpp"))
# Fixed Rho Exact sampling
Rcpp::sourceCpp(file.path(getwd(), "src", "rcpp", "BYM2ExactSampling.cpp"))
# seed used to produce figures
set.seed(51624)
```

1.1 Data Map

Figure 1a: Simulated response data y where the spatial effects ϕ are simulated under a scaled CAR variance structure.

```
x1 <- cbind("y" = y, county_sf)
y_map <- ggplot() +
  geom_sf(data = x1, aes(fill = y)) +
  scale_fill_viridis_c() +
  coord_sf(crs = st_crs(5070)) +
  theme_bw() +
  theme(legend.position = "bottom")
y_map
```



2 Gibbs Sampling

Set priors and initialize sampler object:

```
# Priors
a_sigma <- 0.1
b_sigma <- 0.1
# since PC prior on rho, IG prior parameters will be ignored by sampler object
a_rho <- 0.0
b_rho <- 0.0
# limits on possible values of rho
lower_rho <- 0
upper_rho <- 1.0
# PC prior: pi(rho) = lambda * exp(-lambda * d(rho)), d(rho) = sqrt(2 * KLD(rho))
# PC prior hyperparameter selected in pc_prior_selection.Rmd
lambda_rho <- .0335
# INITIALIZE GIBBS SAMPLER OBJECT
gibbsSampler <- new(BYM2FlatBetaMCMC, X, y, Q_scaled)
gibbsSampler$setPriors(a_sigma, b_sigma, rho_prior_type = "pc",
                       a_rho, b_rho,
                       lower_rho, upper_rho, lambda_rho)
gibbsSampler$initOLS()
```

Take 10000 burn-in samples:

```
gibbsSampler$burnMCMCSample(10000)
```

Draw 30000 posterior samples:

```
# DRAW POSTERIOR SAMPLES
n_sim <- 30000
system.time({
  samps <- gibbsSampler$MCMCSample(n_sim)
})
```

```
##    user  system elapsed
## 328.101   5.672 345.465
```

```
beta_sim <- samps$beta
gamma_sim <- samps$gamma
sigma2_sim <- samps$sigma2
rho_sim <- samps$rho
YFit_sim <- samps$YFit
```

Obtain the posterior samples of ϕ and compute posterior samples of the differences $\frac{\phi_{k_1}-\phi_{k_2}}{\text{Var}(\phi_{k_1}-\phi_{k_2}|y,\rho,\sigma^2)}$:

```
phi_sim <- apply(gamma_sim, MARGIN = 2, function(x) {
  x / sqrt(sigma2_sim * rho_sim)
})
phi_diffs <- BYM2_StdDiff(phi_sim, rho_sim, Q_scaled, X, ij_list)
```

Since this is a simulation example, we have access to the true values of ϕ, hence we can compute the true differences.

```
phi_truediffs <- BYM2_StdDiff(matrix(phi, nrow = 1),
                              rho, Q_scaled, X, ij_list)
```

3 Difference Threshold Selection

We minimize the conditional entropy loss function as discussed in Section 3.2:

```
# Maximize entropy of posterior distribution with respect to epsilon
loss_function <- function(V, epsilon) -ConditionalEntropy(V)
eps_optim <- optim(median(abs(phi_diffs)), function(e) {
  e_vij <- ComputeSimVij(phi_diffs, epsilon = e)
  loss_function(e_vij, epsilon = e)
}, method = "Brent", lower = 0.0001, upper = 5.0)
optim_e <- eps_optim$par
```

Using the resulting difference threshold $\epsilon_{\text{CE}} = 1.6871305$, we compute Monte Carlo estimates of the difference probabilities:

```
optim_e_vij <- ComputeSimVij(phi_diffs, epsilon = optim_e)
```

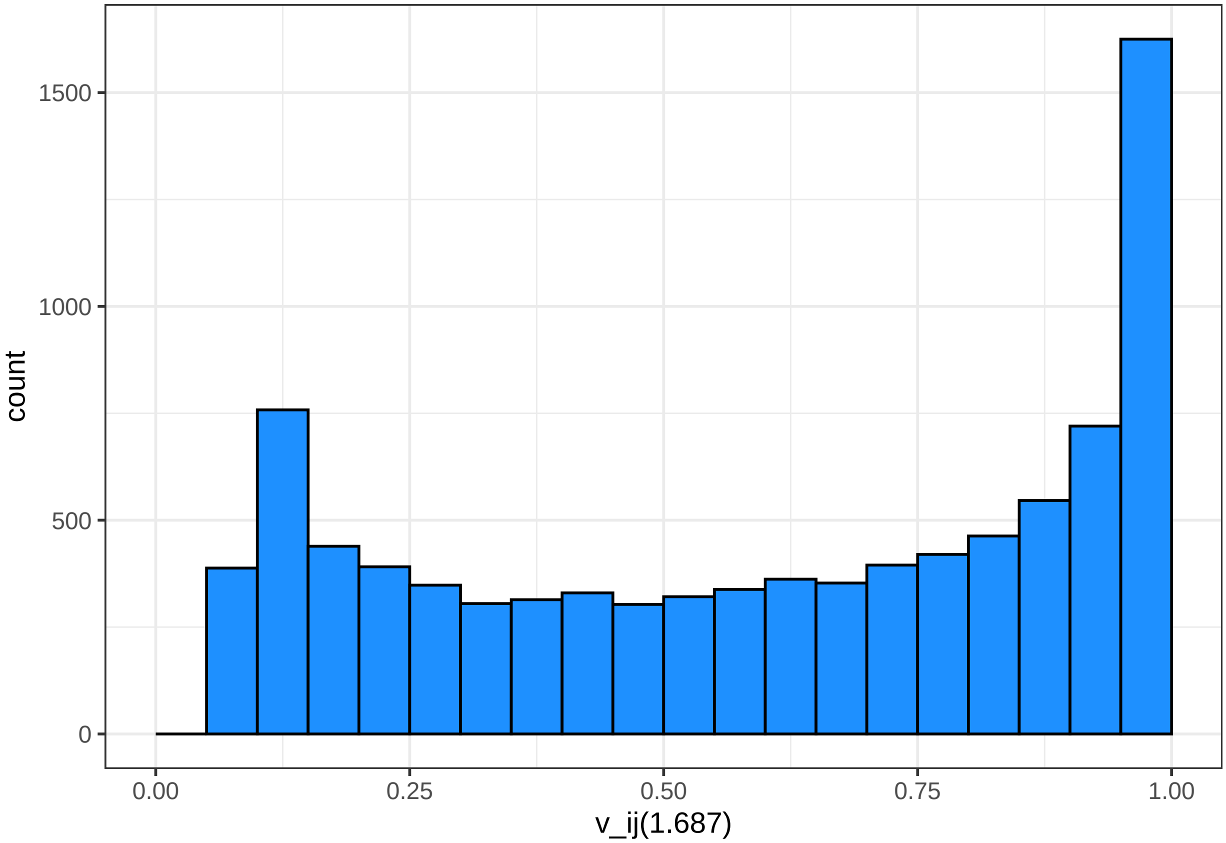
We also compute the true differences:

```
true_diff <- abs(phi_truediffs) > optim_e
mean(true_diff)
```

```
## [1] 0.5299923
```

3.1 Histogram of Difference Probabilities

```
optim_e_vij_hist <- ggplot() +
  geom_histogram(aes(x = optim_e_vij), fill = "dodgerblue", color = "black",
                 breaks = seq(0, 1, by = .05)) +
  lims(x = c(0, 1)) +
  labs(x = paste0("v_ij(", round(optim_e, digits = 3), ")")) +
  theme_bw()
optim_e_vij_hist
```



4 Bayesian FDR Control

We generate decisions using the Bayesian FDR control procedure discussed in Section 3.1 with a Bayesian FDR tolerance of $\eta = .30$ as an example to compute classification metrics.

```
# select cutoff t in d(i, j) = I(v_ij > t) to control FDR and minimize FNR
optim_e_vij_order <- order(optim_e_vij, decreasing = F)
t_seq_length <- 10000
t_seq <- seq(0, max(optim_e_vij) - .001, length.out = t_seq_length)
t_FDR <- vapply(t_seq, function(t) FDR_estimate(optim_e_vij, t, e = 0), numeric(1))
t_FNR <- vapply(t_seq, function(t) FNR_estimate(optim_e_vij, t, e = 0), numeric(1))
# bayesian FDR tolerance
eta <- .05
optim_t <- min(c(t_seq[t_FDR <= eta], 1))
optim_t
```

```
## [1] 0.8555293
```

```
print(paste0("Optimal epsilon: ", optim_e))
```

```
## [1] "Optimal epsilon: 1.68713053808316"
```

```
print(paste0("Optimal t: ", optim_t))
```

```
## [1] "Optimal t: 0.855529252925293"
```

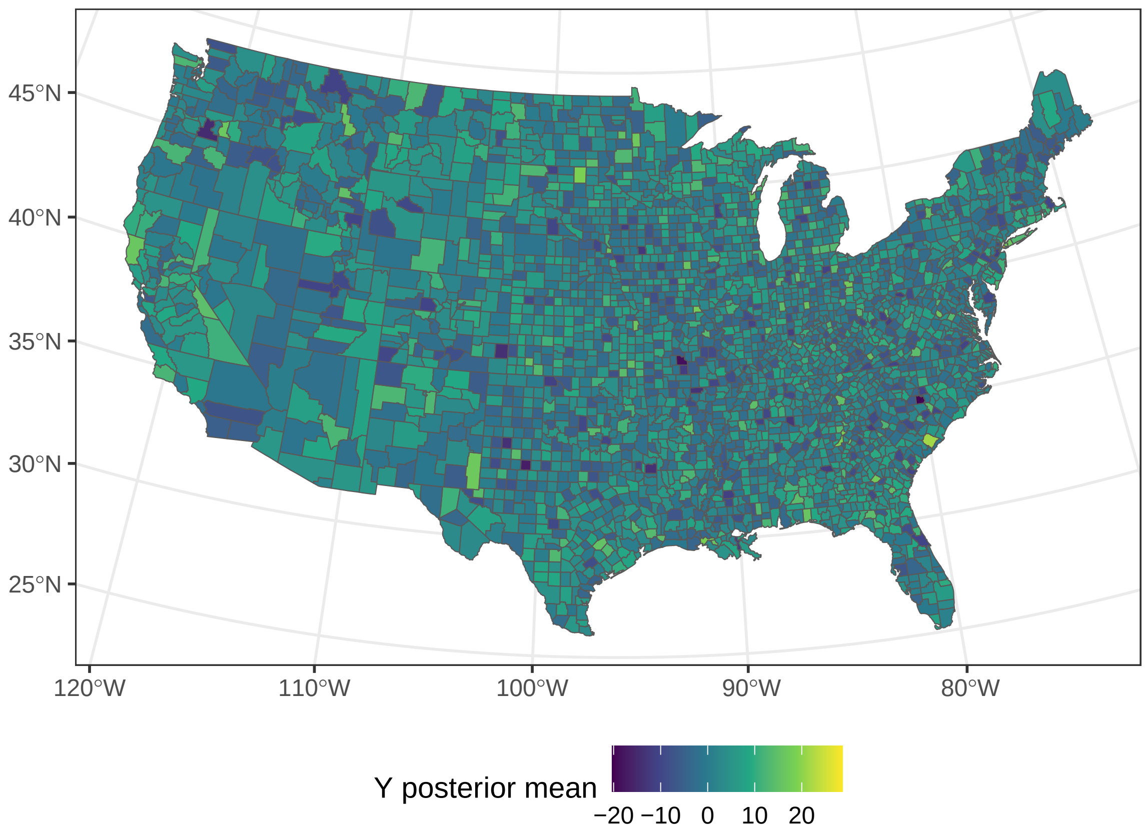
```
decisions <- logical(nrow(ij_list))
decisions[optim_e_vij >= optim_t] <- TRUE
print(ComputeClassificationMetrics(decisions, true_diff))
```

```
##         fdr         fnr sensitivity specificity          F1    accuracy
##  0.05612425  0.34346166  0.55327954  0.96290247  0.69762588  0.74580546
```

4.1 Posterior Maps

Posterior means of data computed using 10,000 posterior samples.

```
yfit_pmeans <- colMeans(YFit_sim)
yfit_pmeans_df <- cbind(y_pmeans = yfit_pmeans, county_sf)
y_pmeans_map <- ggplot() +
  geom_sf(data = yfit_pmeans_df, aes(fill = y_pmeans)) +
  scale_fill_viridis_c(name = "Y posterior mean") +
  coord_sf(crs = st_crs(5070)) +
  theme_bw() +
  theme(legend.position = "bottom")
y_pmeans_map
```



Histogram of posterior samples of ρ:

```
rho_hist <- ggplot() +
  geom_histogram(aes(x = rho_sim), fill = "dodgerblue", color = "black") +
  labs(x = paste0("rho")) +
  theme_bw()
```

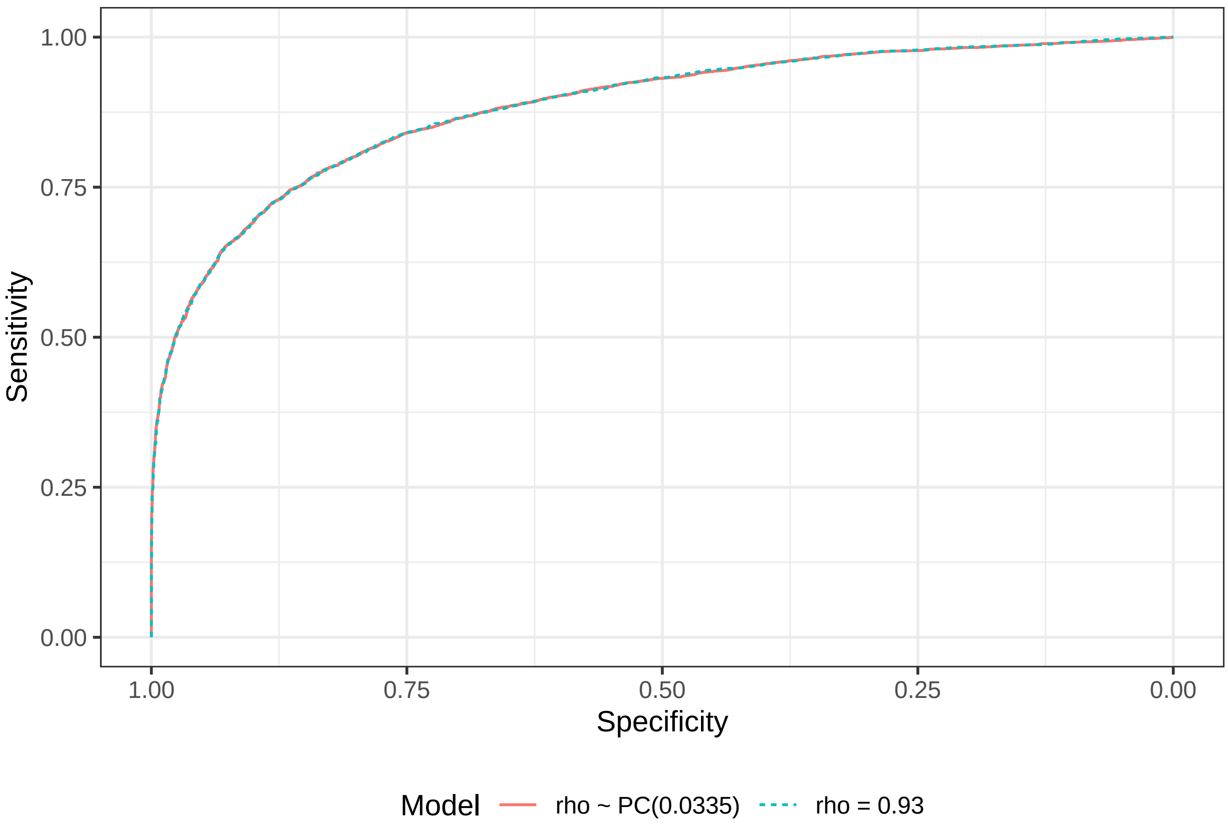
4.2 ROC Curves Comparison

Figure 5a: Classification performance of spatial disparity detection on simulated data when drawing exact samples conditional on $\rho = 0.93$ versus placing a PC prior on ρ and using Metropolis within Gibbs sampling. The classification performance of both models are compared using fixed difference threshold $\epsilon_{CE} = 1.611$. (a) ROC curves for each model. The area under each ROC curve is 0.881.

```
# compare to exact model
# output saved in BYM2_exact_simulation.Rmd
phi_diffs2 <- readRDS(file.path(getwd(), "output", "US_exact_sample_sim",
                                "phi_diffs.Rds"))
optim2_e_vij <- ComputeSimVij(phi_diffs2, epsilon = optim_e)
roc_list <- list(
  pROC::roc(as.vector(true_diff), as.vector(optim_e_vij)),
  pROC::roc(as.vector(true_diff), as.vector(optim2_e_vij))
)
names(roc_list) <- c(
  paste0("rho ~ PC(", lambda_rho, ")"),
  paste0("rho = ", rho)
)
auc <- lapply(roc_list, function(r) round(pROC::auc(r), 3))
auc
```

```
## $`rho ~ PC(0.0335)`
## [1] 0.883
##
## $`rho = 0.93`
## [1] 0.883
```

```
roc_plot <- pROC::ggroc(roc_list, aes = c("colour", "linetype")) +
  scale_color_discrete(name = "Model") +
  scale_linetype_discrete(name = "Model") +
  theme_bw() +
  theme(legend.position = "bottom") +
  labs(x = "Specificity", y = "Sensitivity")
roc_plot
```



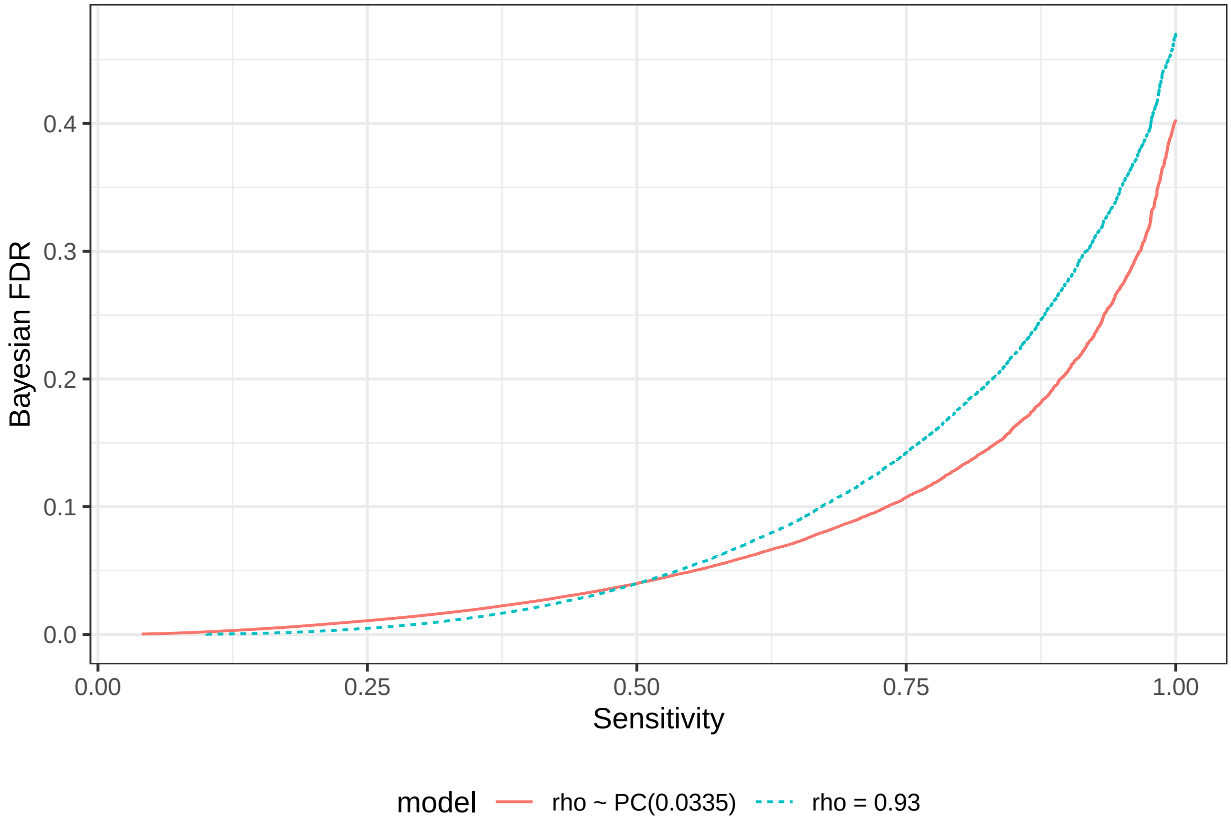
4.3 Bayesian FDR vs. Sensitivity

We compute the Bayesian FDR versus sensitivity curves for both the exact conjugate model conditioning on the true value of ρ and the BYM2 model with a penalized complexity prior on ρ.

Figure 5b: Classification performance of spatial disparity detection on simulated data when drawing exact samples conditional on $\rho = 0.93$ versus placing a PC prior on ρ and using Metropolis within Gibbs sampling. The classification performance of both models are compared using fixed difference threshold $\epsilon_{CE} = 1.611$. (b) Bayesian FDR versus true senstivity.

```r
sim2_FDR_sensitivity <- data.table(
  t = t_seq,
  FDR = vapply(t_seq, function(target_t) FDR_estimate(optim2_e_vij, target_t, e = 0), numeric(1)),
  true_sensitivity = vapply(t_seq, function(target_t) {
    decisions <- logical(nrow(ij_list))
    decisions[optim2_e_vij >= target_t] <- TRUE
    ComputeClassificationMetrics(decisions, true_diff)["sensitivity"]
  }, numeric(1))
)
sim_FDR_sensitivity <- data.table(
  t = t_seq,
  FDR = vapply(t_seq, function(target_t) FDR_estimate(optim_e_vij, target_t, e = 0), numeric(1)),
  true_sensitivity = vapply(t_seq, function(target_t) {
    decisions <- logical(nrow(ij_list))
    decisions[optim_e_vij >= target_t] <- TRUE
    ComputeClassificationMetrics(decisions, true_diff)["sensitivity"]
  }, numeric(1))
)

FDR_sensitivity_df <- rbind(
  cbind(sim_FDR_sensitivity, model = paste0("rho ~ PC(", lambda_rho, ")")),
  cbind(sim2_FDR_sensitivity, model = paste0("rho = ", rho))
)
FDR_sensitivity_df$model <- factor(FDR_sensitivity_df$model,
                                   levels = c(paste0("rho ~ PC(", lambda_rho, ")"),
                                              paste0("rho = ", rho)))
FDR_sensitivity_plot <- ggplot() +
  geom_line(data = FDR_sensitivity_df,
            aes(x = true_sensitivity, y = FDR, col = model, linetype = model)) +
  theme_bw() +
  theme(legend.position = "bottom") +
  labs(x = "Sensitivity", y = "Bayesian FDR")
FDR_sensitivity_plot
```



4.4 Rejection Order Graph

To check the stability of the rankings with respect to choice of the difference threshold, we plot the rankings of the difference probabilities under different values of the difference threshold ϵ and compare to those we previously computed using ϵ_{CE}.

Figure 6: Simulated rankings of ϵ-difference probabilities $\{v_{ij}(\epsilon)\}_{(i,j)\in L}$ for $\epsilon = 0.537, 1.074, 2.417, 4.833$ versus rankings of $\{v_{ij}(\epsilon)\}_{(i,j)\in L}$ using a PC prior on ρ. True posterior differences according to the true simulated values of ϕ and ρ are shown in blue.

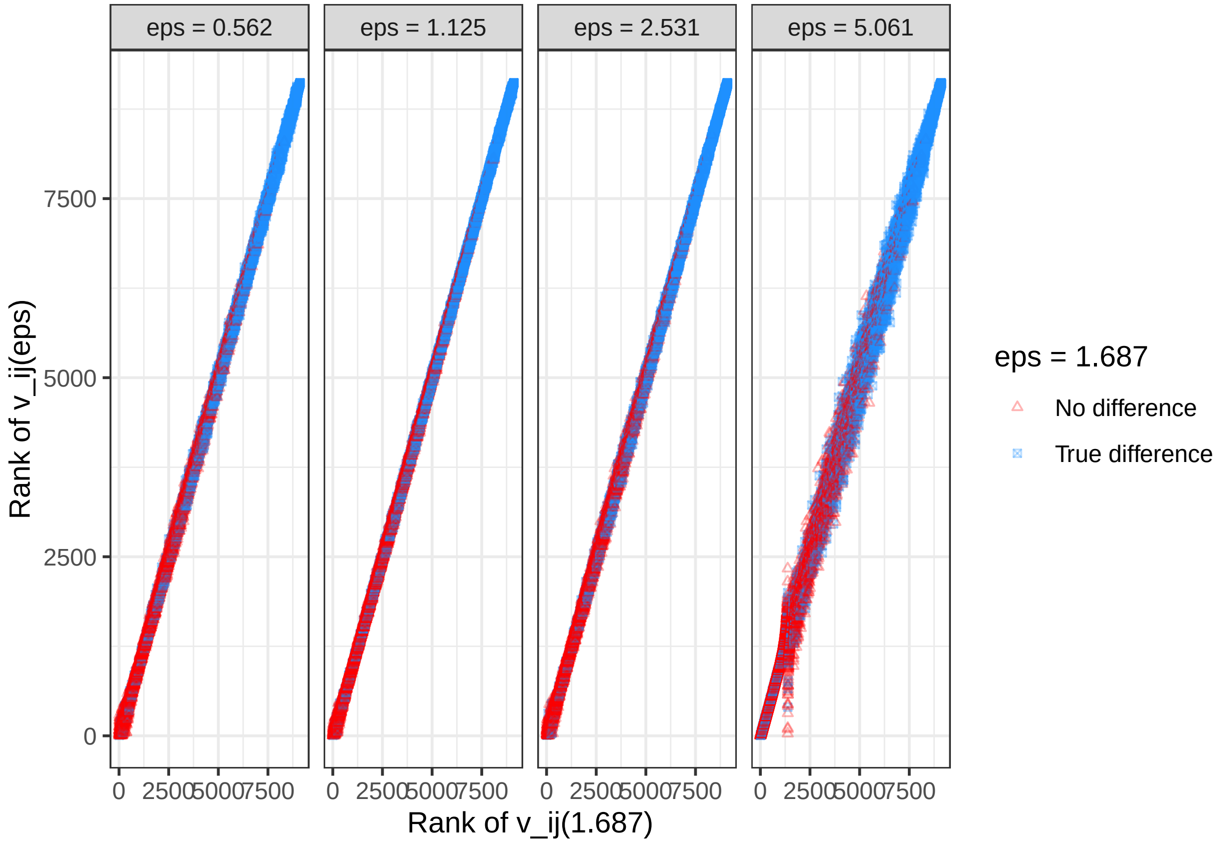
```
rej_indx <- optim_e_vij_order[optim_e_vij_order %in%
                                which(optim_e_vij >= optim_t)]
detected_borders <- ij_list[rej_indx,]
node1_all <- county_sf[detected_borders$i,]
node2_all <- county_sf[detected_borders$j,]

## use different difference threshold values
e2 <- round(optim_e / 3, digits = 3)
e3 <- round(optim_e / 1.5, digits = 3)
e4 <- round(optim_e * 1.5, digits = 3)
e5 <- round(optim_e * 3, digits = 3)

e2_vij <- ComputeSimVij(phi_diffs, epsilon = e2)
e3_vij <- ComputeSimVij(phi_diffs, epsilon = e3)
e4_vij <- ComputeSimVij(phi_diffs, epsilon = e4)
```

```
e5_vij <- ComputeSimVij(phi_diffs, epsilon = e5)

optim_e_vij_order <- order(optim_e_vij, decreasing = F)
e2_vij_order <- order(e2_vij[optim_e_vij_order], decreasing = F)
e3_vij_order <- order(e3_vij[optim_e_vij_order], decreasing = F)
e4_vij_order <- order(e4_vij[optim_e_vij_order], decreasing = F)
e5_vij_order <- order(e5_vij[optim_e_vij_order], decreasing = F)
rejection_path <- data.table(
  optim_e_vij = seq_along(optim_e_vij),
  e2_vij_order = e2_vij_order,
  e3_vij_order = e3_vij_order,
  e4_vij_order = e4_vij_order,
  e5_vij_order = e5_vij_order,
  true_diff = true_diff[optim_e_vij_order]
)
rejection_path <- melt(rejection_path,
                       id.vars = c("optim_e_vij", "true_diff"),
                       variable.name = "order_type",
                       value.name = "order")
rejection_path[, order_type := fcase(
  order_type == "e2_vij_order", paste0("eps = ", e2),
  order_type == "e3_vij_order", paste0("eps = ", e3),
  order_type == "e4_vij_order", paste0("eps = ", e4),
  order_type == "e5_vij_order", paste0("eps = ", e5)
)]
sim_vij_order_graph <- ggplot() +
  geom_point(data = rejection_path,
             aes(x = optim_e_vij, y = order, color = true_diff,
                 shape = true_diff),
             alpha = 0.3, size = 1) +
  #geom_vline(xintercept = nrow(ij_list) - sum(optim_e_vij == 1)) +
  facet_grid(~order_type) +
  labs(x = paste0("Rank of v_ij(", round(optim_e, digits = 3), ")"),
       y = "Rank of v_ij(eps)") +
  theme_bw() +
  scale_color_manual(name = paste0("eps = ", round(optim_e, digits = 3)),
                     labels = c("No difference", "True difference"),
                     values = c("FALSE" = "red", "TRUE" = "dodgerblue")) +
  scale_shape_manual(name = paste0("eps = ", round(optim_e, digits = 3)),
                     labels = c("No difference", "True difference"),
                     values = c("FALSE" = 2, "TRUE" = 7))
sim_vij_order_graph
```


eps = 0.562
eps = 1.125
eps = 2.531
eps = 5.061
Rank of v_ij(eps)
0
2500
5000
7500
Rank of v_ij(1.687)
0
2500
5000
7500
eps = 1.687
△ No difference
⊠ True difference