Simulation Example: Exact Conjugate Model

In this RMD file, we reproduce the simulation results for the exact conjugate model discussed in Section 4.1, conditioning on the true simulated value of ρ.

1 Packages and Data Setup

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

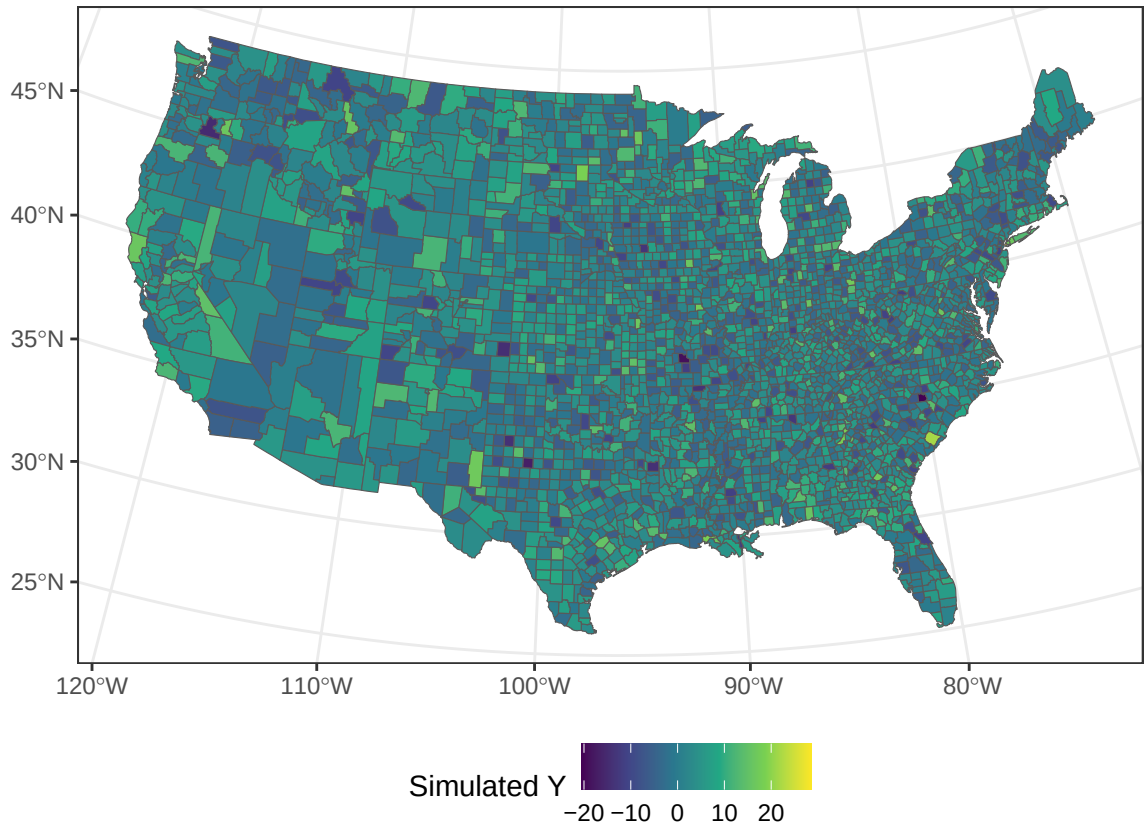
Generate simulation data and load in helper functions:

```
# 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"))
# Exact sampling
sourceCpp(file.path(getwd(), "src", "rcpp", "BYM2ExactSampling.cpp"))
set.seed(122)
```

1.1 Data Maps

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

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



Posterior map of true values of spatial residuals ϕ:

```
phi_map <- ggplot() +
  geom_sf(data = county_sf, aes(fill = phi)) +
  scale_fill_viridis_c(name = "Simulated phi") +
  theme_bw() +
  coord_sf(crs = st_crs(5070)) +
  theme(legend.position = "bottom")
```

2 Exact Sampling

Set priors and initialize sampler object, conditioning on true value of ρ:

```
model_rho <- rho
BYM2Sampler <- new(BYM2ExactSampler, X, y, Q_scaled, model_rho)
# Set priors
a_sigma <- 0.1
b_sigma <- 0.1
M_0inv <- diag(rep(1e-4, p))
m_0 <- rep(0, p)
BYM2Sampler$SetPriors(M_0inv, m_0, a_sigma, b_sigma)
```

Draw 10000 exact samples from the joint posterior:

```
n_sim <- 10000
system.time({
  exact_samps <- BYM2Sampler$ExactSample(n_sim)
})
```

```
##    user  system elapsed
## 102.060  30.771 133.507
```

```
beta_sim <- exact_samps$beta
gamma_sim <- exact_samps$gamma
sigma2_sim <- exact_samps$sigma2
YFit_sim <- exact_samps$YFit
```

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

```
denom <- sqrt(sigma2_sim * model_rho)
phi_sim <- apply(gamma_sim, MARGIN = 2, function(x) x / denom)
rm(denom)
phi_diffs <- BYM2_StdDiff(phi_sim, rep(model_rho, n_sim), Q_scaled, X, ij_list)
```

Save samples for later comparison to model with PC prior on ρ:

```
saveRDS(phi_diffs, file.path(getwd(), "output", "US_exact_sample_sim",
                             "phi_diffs.Rds"))
```

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 $\epsilon_{\text{CE}} = 1.2963012$, we compute Monte Carlo estimates of the difference probabilities:

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

We also compute the number of true differences:

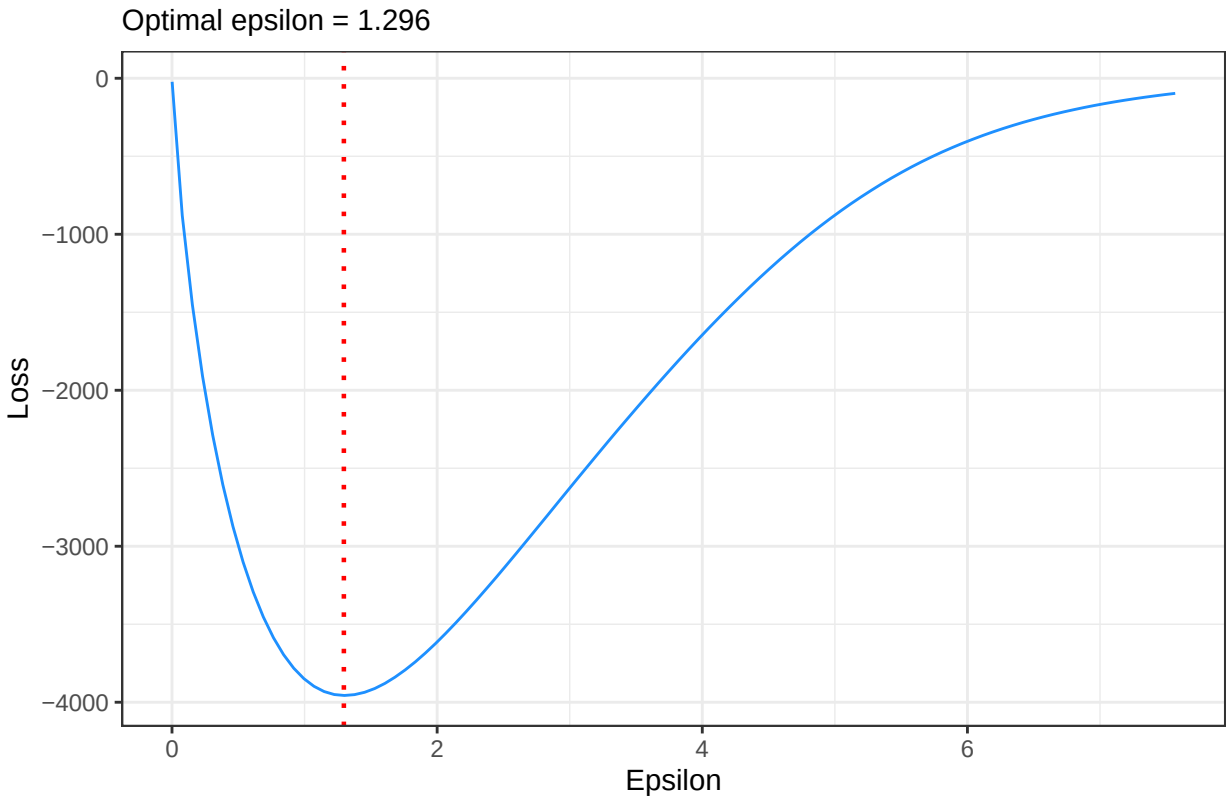
```r
# number of true disparities at e_CE level
true_diff <- (abs(phi_truediffs) > optim_e)
sum(true_diff)
```

```
## [1] 5750
```

3.1 Epsilon Loss Curve

Conditional Entropy Loss function evaluated across ϵ with a dotted red line indicating the numerically optimal $\epsilon_{\text{CE}} = 1.296$.

```r
eps_seq <- seq(0.001, max(abs(phi_diffs)) / 2, length.out = 100)
sim_vij <- ComputeSimVij(phi_diffs, epsilon = eps_seq)
sim_loss <- loss_function(sim_vij, epsilon = eps_seq)
eps_loss_graph <- ggplot() +
  geom_line(data = data.frame(sim_loss = sim_loss, epsilon = eps_seq),
            aes(x = epsilon, y = sim_loss), color = "dodgerblue") +
  labs(x = "Epsilon", y = "Loss", title = "",
       subtitle = paste0("Optimal epsilon = ", round(optim_e, digits = 3))) +
  geom_vline(xintercept = optim_e, lwd = 0.8, linetype = "dotted",
             color = "red") +
  theme_bw()
eps_loss_graph
```



3.2 Histogram of Difference Probabilities

Histogram of the posterior probabilities $\{h_{ij}(\epsilon)\}_{(i,j)\in L}$.

```
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("h_ij(", round(optim_e, digits = 3), ")")) +
  theme_bw()
```

4 Bayesian FDR Control

We generate decisions using the Bayesian FDR control procedure discussed in Section 3.1 with a Bayesian FDR tolerance of $\eta = .10$ as an example to compute classification metrics.

```
# select cutoff t in d(i, j) = I(v_ij > t) to control FDR and minimize FNR
eta <- .10
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))
optim_t <- min(c(t_seq[t_FDR <= eta], 1))
optim_t
```

```
## [1] 0.6479163
```

```
decisions <- logical(nrow(ij_list))
decisions[optim_e_vij >= optim_t] <- TRUE
print(paste0("Optimal epsilon: ", optim_e))
```

```
## [1] "Optimal epsilon: 1.29630124972874"
```

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

```
## [1] "Optimal t: 0.647916291629163"
```

```
print(ComputeClassificationMetrics(decisions, true_diff))
```

```
##         fdr         fnr sensitivity specificity          F1    accuracy
##   0.1021482   0.3629581   0.7123478   0.8616800   0.7944143   0.7675184
```

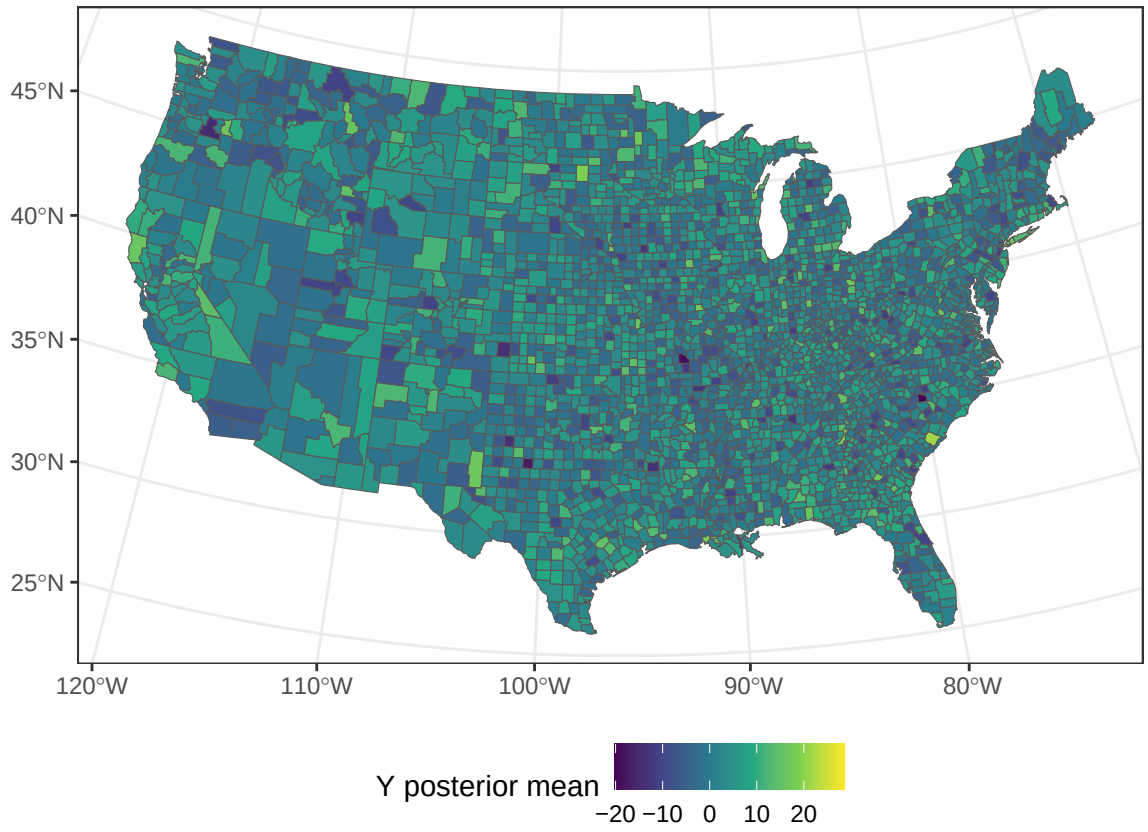
```
mean(decisions)
```

```
## [1] 0.5002742
```

4.1 Posterior Maps

Posterior means of data computed using 10,000 exact samples from the joint posterior.

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

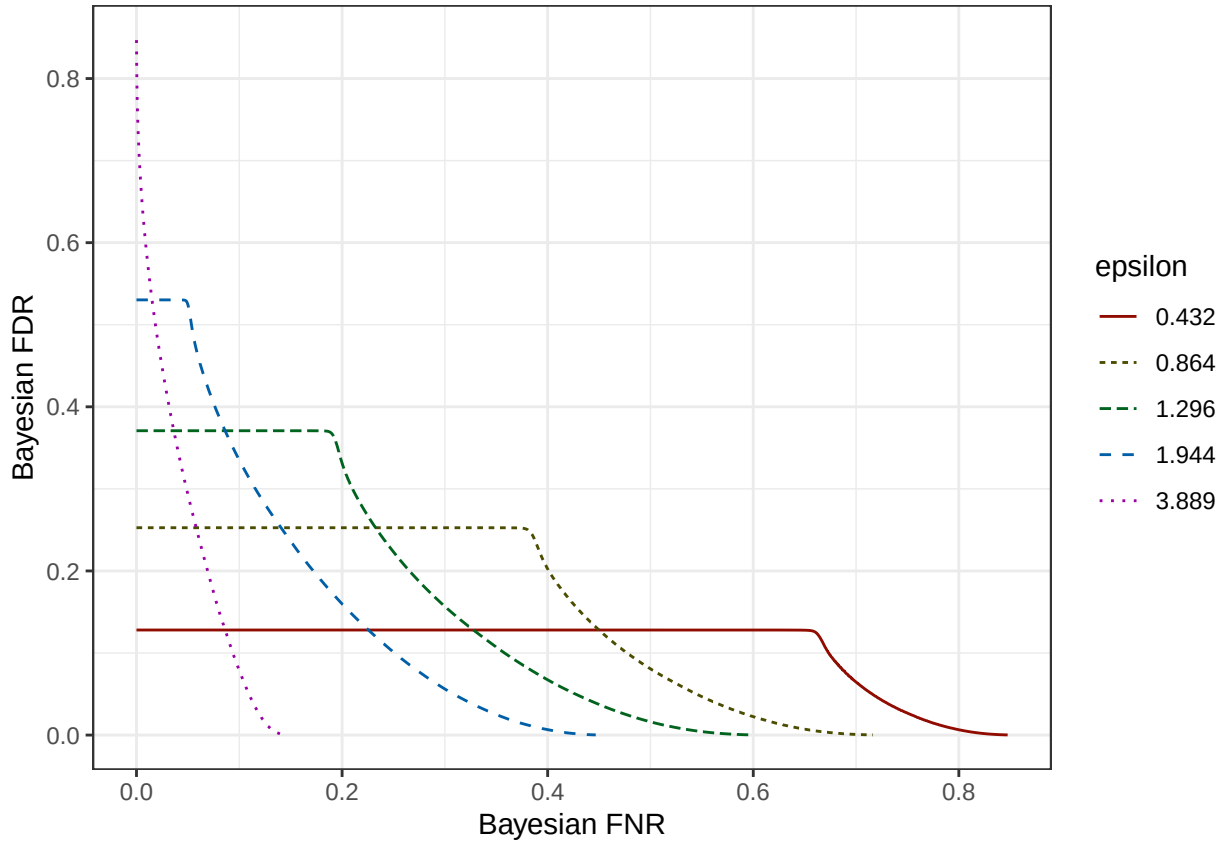


4.2 FDR vs. FNR curves

Figure 3: Bayesian FDR against FNR curves for different values of ϵ.

```
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)
eps_seq <- c(e2, e3, optim_e, e4, e5)
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)
e_vijs <- cbind(e2_vij, e3_vij, optim_e_vij, e4_vij, e5_vij)
FDR_FNR_curves <- lapply(seq_len(ncol(e_vijs)), function(i) {
```

```
  v <- e_vijs[, i]
  FDR <- vapply(t_seq, function(t) FDR_estimate(v, t, e = 0), numeric(1))
  FNR <- vapply(t_seq, function(t) FNR_estimate(v, t, e = 0.1), numeric(1))
  sensitivity <- data.table(epsilon = round(eps_seq[i], digits = 3),
                            t = t_seq,
                            FDR = FDR,
                            FNR = FNR)
}) %>% do.call(rbind, .)
FDR_FNR_curves[, epsilon := factor(epsilon, levels = sort(unique(epsilon)),
                                   ordered = T)]
FDRFNR_eps_graph <- ggplot() +
  geom_line(data = FDR_FNR_curves, aes(x = FNR, y = FDR, group = epsilon,
                                       color = epsilon, linetype = epsilon)) +
  labs(x = "Bayesian FNR", y = "Bayesian FDR") +
  scale_color_hue(l = 30) +
  theme_bw()
FDRFNR_eps_graph
```



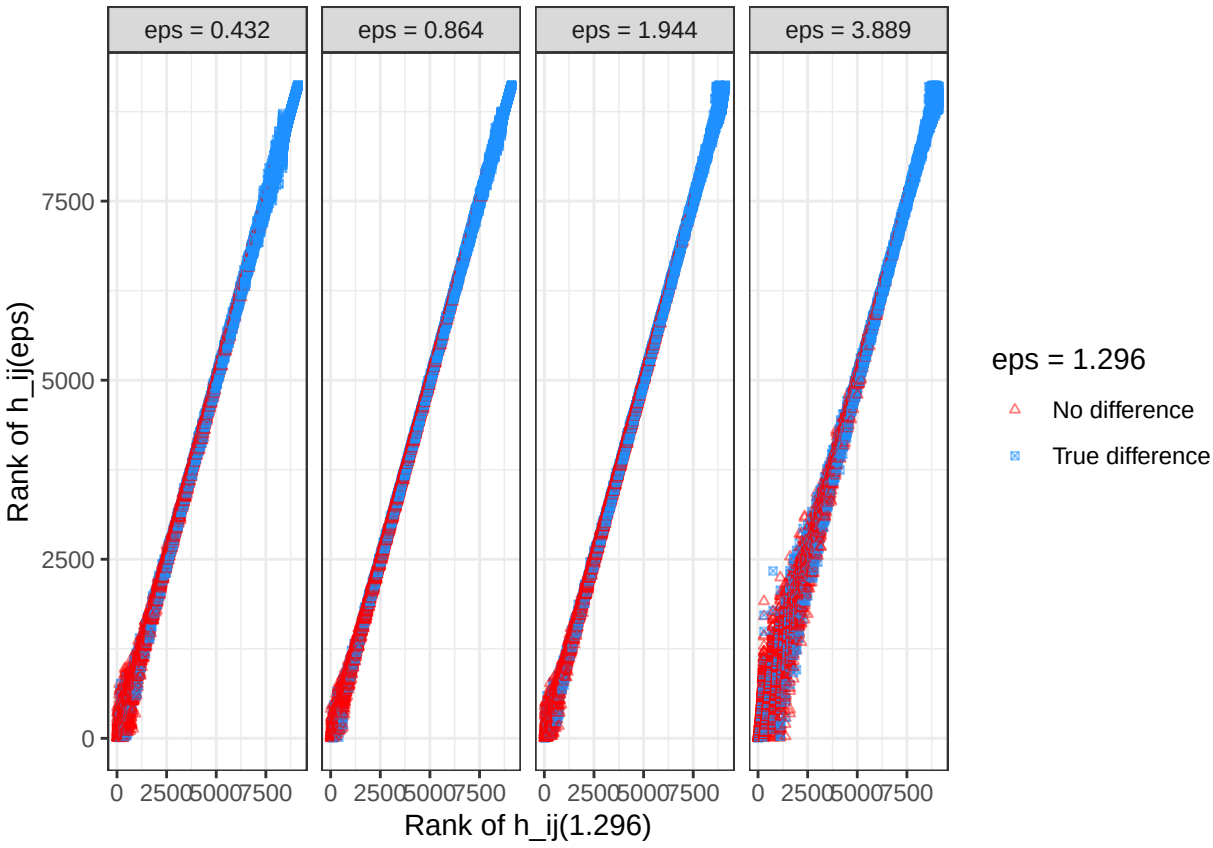
4.3 Rejection Order Graph

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}.

Rankings of the simulated difference probabilities $\{h_{ij}(\epsilon)\}_{(i,j)\in L}$ for $\epsilon = 0.432, 0.864, 1.944, 3.889$ versus

rankings of $\{h_{ij}(\epsilon_{CE})\}_{(i,j)\in L}$ when conditioning on the true value of ρ. True posterior differences according to the true values of ϕ and ρ are shown in blue.

```
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 = 0.5, size = 1) +
  #geom_vline(xintercept = nrow(ij_list) - sum(optim_e_vij == 1)) +
  facet_grid(~order_type) +
  labs(x = paste0("Rank of h_ij(", round(optim_e, digits = 3), ")"),
       y = "Rank of h_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 = 0.432
eps = 0.864
eps = 1.944
eps = 3.889
Rank of h_ij(eps)
0
2500
5000
7500
0 2500 5000 7500
Rank of h_ij(1.296)
eps = 1.296
No difference
True difference