Simulation Results

Owen G. Ward

11/8/2021

```
knitr::opts_chunk$set(echo = TRUE)
library(here)
```

```
## here() starts at C:/Users/owenw/Documents/Github_Local/OCD_Events
```

```
library(latex2exp)
source(here("Experiments/utils.R"))
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.3     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Included are some initial simulation results. For all experimental results shown below, a two class model was considered with a homogeneous Poisson process, with rate matrix given by

$$B = \begin{pmatrix} 2 & 0.05 \\ 0.05 & 1 \end{pmatrix}$$

and equal group proportions $(0.5, 0.5)$. Unless stated, all experiments are with $n = 100$ nodes.

Here we discuss

- Recovery of the rate matrix under random initialization
- Convergence of the windowed ELBO
- Simulation results investigating the agreement between theoretical results on the regret rate and observed regret.
- Performance of our algorithm in terms of online loss of predicted events.

Recovery for random initialization

Although our method, along with many variational optimization methods, relies heavily on the initialization, for the simulation scenario described above we see that good recovery of B can be achieved in many scenarios,

```
exp_1_files <- list.files(path = here("Experiments/exp_results/"),
                          pattern = "exp1_time")

elbo_dat <- exp_1_files %>%
  map(~readRDS(here("Experiments/exp_results/", .x))) %>%
  map(~.x %>%  tibble(
    clust = map_dbl(., "clust"),
    elbo = map(., "est_elbo"),
    time = map_dbl(., "time")
  ) %>%
    mutate(sim = row_number()) %>%
    select(sim, clust, time, elbo) %>%
    unnest(cols = c(elbo)) %>%
    group_by(sim) %>%
    mutate(wind = 1:unique(time))) %>%
  bind_rows()
```

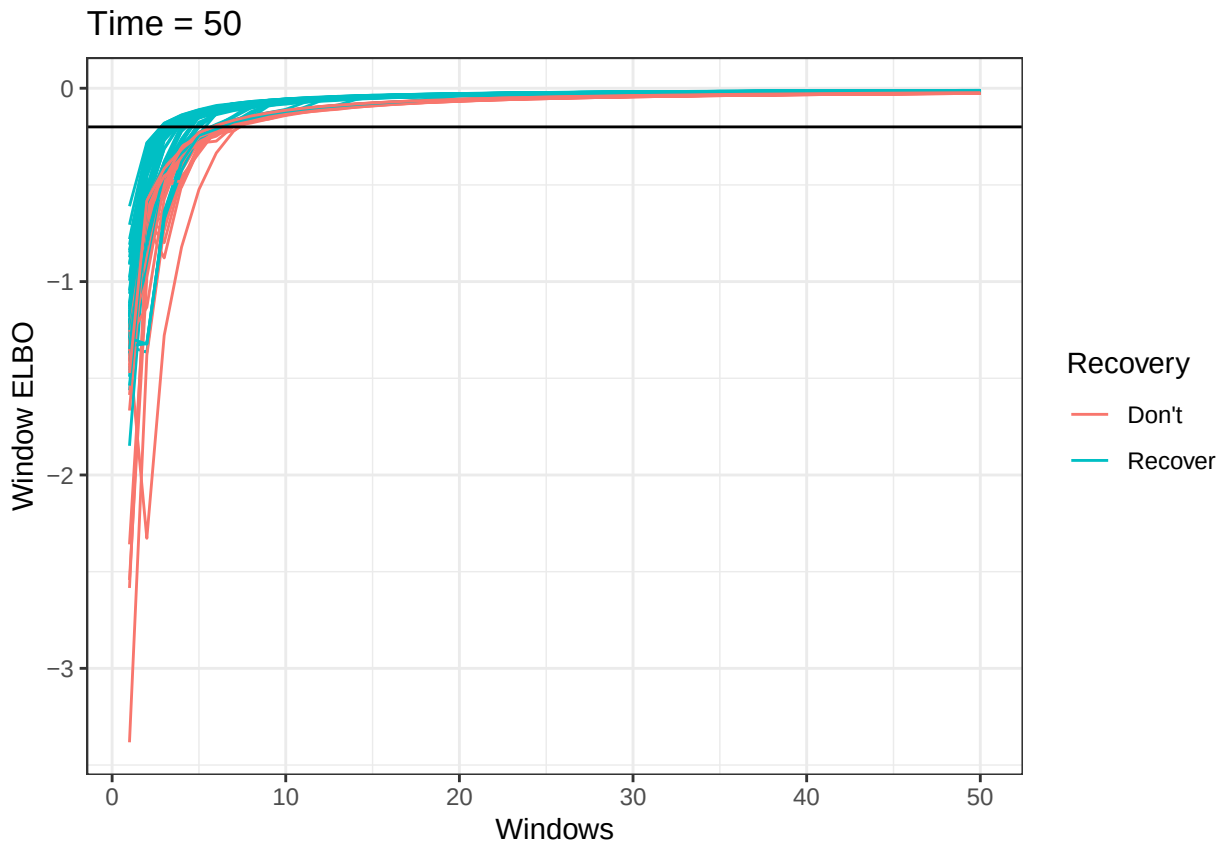
Convergence of the Window ELBO

The definition of the window ELBO here is given by

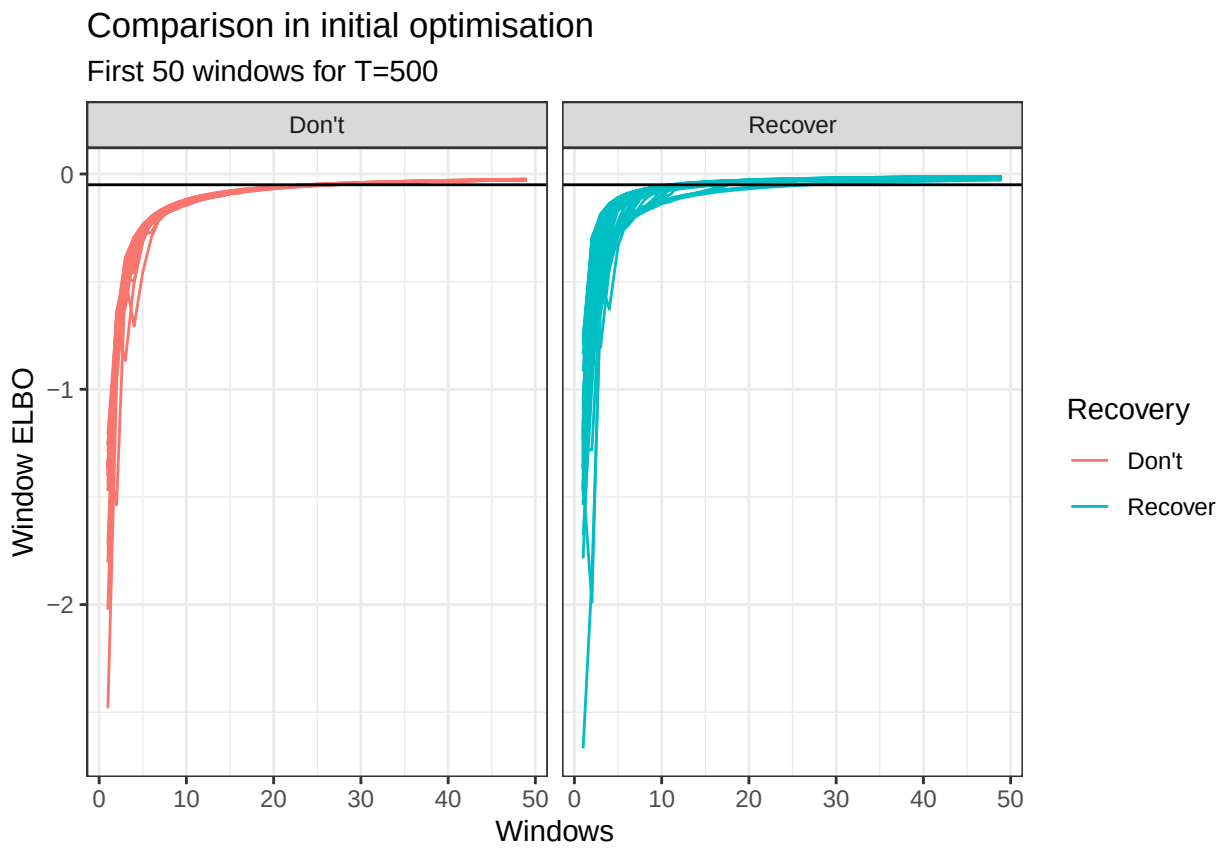
$$ELBO(wind_m) = \frac{1}{events_{cum}} \left[\sum_{(i,j) \in A} \sum_{k,l} \tau_{ik} \tau_{jl} \left\{ \int_{dTm}^{dT(m+1)} \log \lambda_{kl}(t) dN_{ij}(t) - \int_{dTm}^{dT(m+1)} \lambda_{kl}(t) dt \right\} + \sum_{i=1}^{n} \sum_{k=1}^{K} \tau_{ik} \log \pi_k / \tau_{ik} \right],$$

with $events_{cum}$ the number of events observed up to $dT(m+1)$. Although this is slightly different to what we define as the window ELBO in the paper, this is what we have always used in all experiments previously, as opposed to $|A|$.

```
elbo_dat %>%
  filter(time == 50) %>%
  mutate(correct = ifelse(clust == 1, "Recover", "Don't")) %>%
  ggplot(aes(wind, elbo, colour = correct)) + geom_line(aes(group = sim)) +
  labs(x = "Windows", y = "Window ELBO", colour = "Recovery",
       title = "Time = 50") +
  geom_hline(yintercept = -0.2)
```




```
elbo_dat %>%
  filter(time == 1000) %>%
  filter(wind < 50) %>%
  mutate(correct = ifelse(clust == 1, "Recover", "Don't")) %>%
  ggplot(aes(wind, elbo, colour = correct)) + geom_line(aes(group = sim)) +
  geom_hline(yintercept = -0.05) +
  labs(x = "Windows", y = "Window ELBO",  colour = "Recovery",
       title = "Comparison in initial optimisation",
       subtitle = "First 50 windows for T=500") +
  facet_wrap(~correct)
```



Parameter Recovery

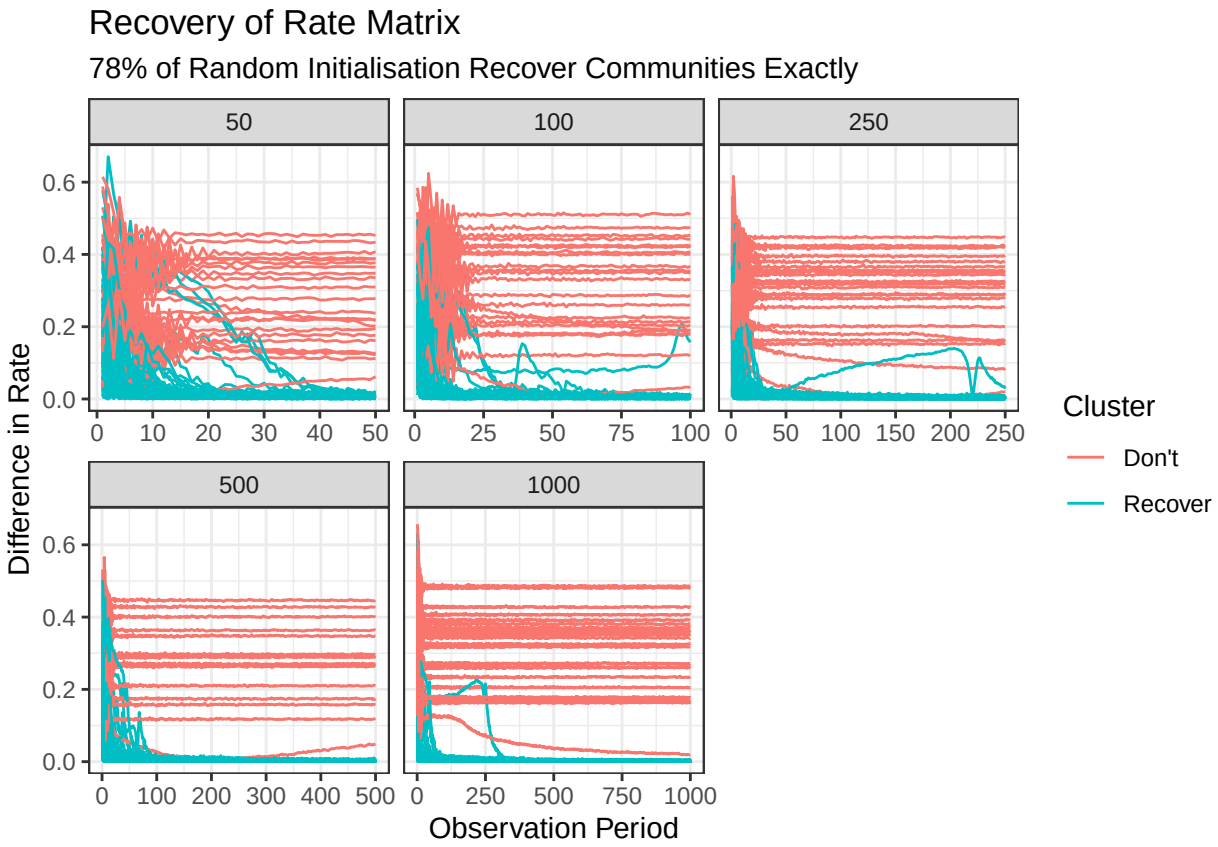
Here we are using

$$\frac{1}{K}|\sum B_{ij} - \sum \hat{B}_{ij}|$$

as the metric for convergence. We see that when the community is recovered (blue) the matrix is recovered.

```
all_b_files <- list.files(path = here("Experiments/exp_results/"),
                          pattern = "exp1_all")

b_ests <- all_b_files %>%
  map(~readRDS(here("Experiments/exp_results/", .x))) %>%
  flatten() %>%
  imap(~update_list(., sim = .y))

sum_b <- b_ests %>% map(~ apply(.x$est_B, 3, sum) )

all_b_df <- b_ests %>% {
  tibble(
    map_dfr(., `[`, c("sim", "clust", "time")),
    sum_B = sum_b
  )
```

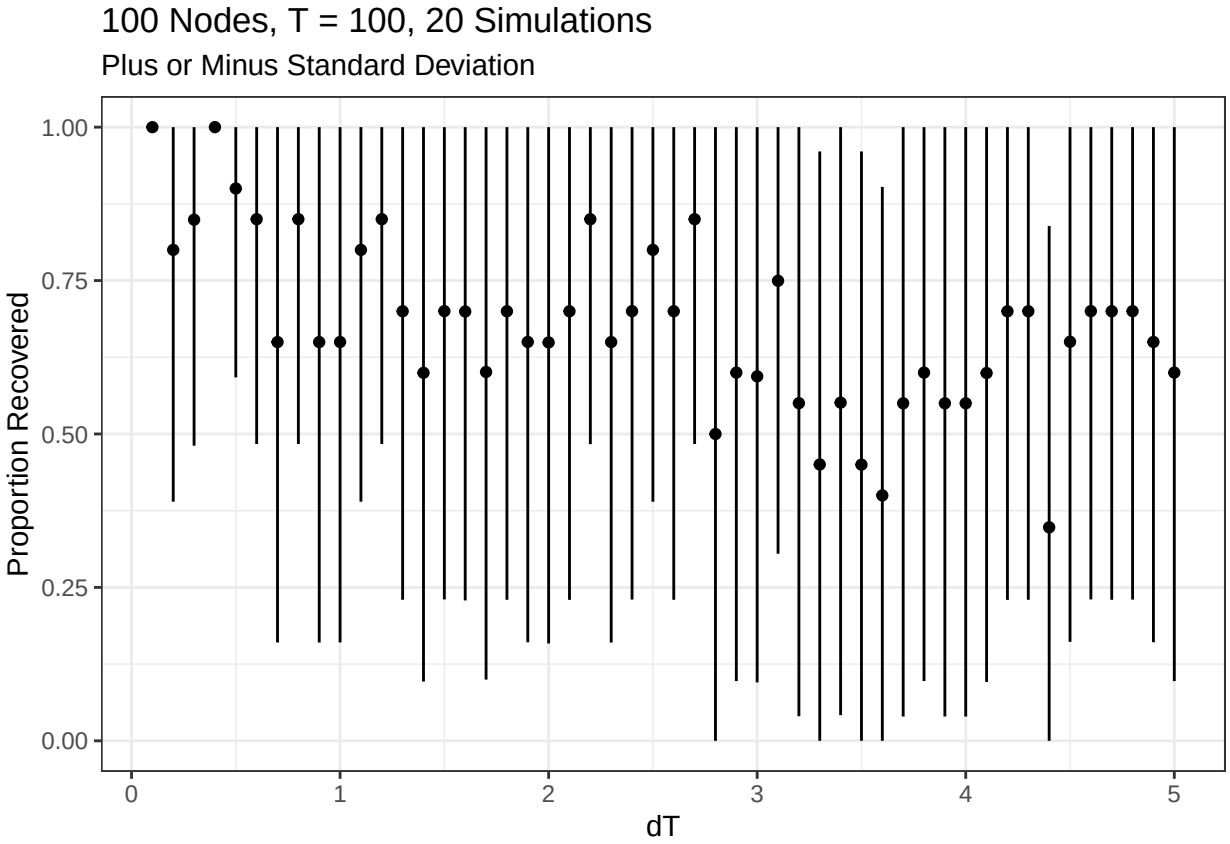
```
} %>%
  unnest(sum_B) %>%
  mutate(diff = abs(sum_B - 3.1)/4) %>%
  group_by(sim) %>%
  mutate(window = row_number()) %>%
  ungroup()

all_b_df %>%
  mutate(Cluster = ifelse(clust == 1, "Recover", "Don't"),
         time = as.factor(time)) %>%
  # filter(time == 1000) %>%
  ggplot(aes(window, diff)) +
  geom_line(aes(group = sim, colour = Cluster)) +
  facet_wrap(~time, scales = "free_x") +
  labs(x = "Observation Period", y = "Difference in Rate",
       subtitle = "78% of Random Initialisation Recover Communities Exactly",
       title = "Recovery of Rate Matrix")
```



Window Size

```
all_sims <- list.files(path = here("Experiments/exp_results/"),
           pattern = "exp_2")
```

```
all_output <- all_sims %>%
  map(~readRDS(here("Experiments/exp_results/", .x)))

sims_df <- all_output %>%
  imap(~ .x %>% map_dfr(`[`, c("clust", "dT")) %>%
                      mutate(sim = .y)) %>%
  bind_rows()


sims_df %>%
  group_by(dT) %>%
  summarise(mean = mean(clust), sd = sd(clust),
            lower = max(mean - sd,0), upper = min(mean + sd, 1)) %>%
  # mutate(lower = max(mean - sd,0), upper = min(mean + sd, 1)) %>%
  ggplot(aes(dT, mean)) +
  geom_point() +
  geom_linerange(aes(ymin = lower, ymax = upper)) +
  labs(y = "Proportion Recovered",
       subtitle = "Plus or Minus Standard Deviation",
       title = "100 Nodes, T = 100, 20 Simulations")
```



Number of Nodes

Here we quickly show the community recovery for a range of values for the number of nodes. Note that the current simulation procedure is not suited for simulating larger networks but could be modified to account for this.

```
exp_5_files <- list.files(path = here("Experiments/exp_results/"),
                          pattern = "exp5_")

exp_5_data <- exp_5_files %>%
  map(~readRDS(here("Experiments/exp_results/", .x))) %>%
  flatten() %>%
  imap(~update_list(., sim = .y)) %>%
  map_dfr(`[`, c("curr_n", "clust", "sim"))


exp_5_data %>%
  group_by(curr_n) %>%
  summarise(mean_ari = mean(clust), sd_ari = sd(clust)) %>%
  rename(NumberNodes = curr_n)
```

```
## # A tibble: 6 x 3
##   NumberNodes mean_ari sd_ari
##         <dbl>    <dbl>  <dbl>
## 1          20    0.647  0.493
## 2          50    0.751  0.443
## 3         100    0.800  0.410
## 4         150    0.65   0.489
## 5         200    0.8    0.410
## 6         250    0.8    0.410
```

Number of Groups

Here we show the average ARI for K groups where $K = 3, 4, 5, 6$ with rate matrix given by

$$B_k = \begin{pmatrix} 1 & 0.05 & \dots & 0.05 \\ 0.05 & 2 & 0.05 & \dots \\ \vdots & & \ddots & 0.05 \\ 0.05 & \dots & & K \end{pmatrix}$$

for a given K, with 50 simulated networks with 100 nodes for $T = 100$ for each simulation.

```
exp_6_files <- list.files(path = here("Experiments/exp_results/"),
                          pattern = "exp6_")

exp_6_data <- exp_6_files %>%
  map(~readRDS(here("Experiments/exp_results/", .x))) %>%
  flatten() %>%
  imap(~update_list(., sim = .y)) %>%
  map_dfr(`[`, c("K", "clust", "sim"))
```

```
exp_6_data %>%
  group_by(K) %>%
  summarise(mean_ari = mean(clust), sd_ari = sd(clust))
```

```
## # A tibble: 4 x 3
##       K mean_ari sd_ari
##   <int>    <dbl>  <dbl>
## 1     3    0.684  0.205
## 2     4    0.695  0.176
## 3     5    0.612  0.154
## 4     6    0.596  0.127
```

Regret Rate

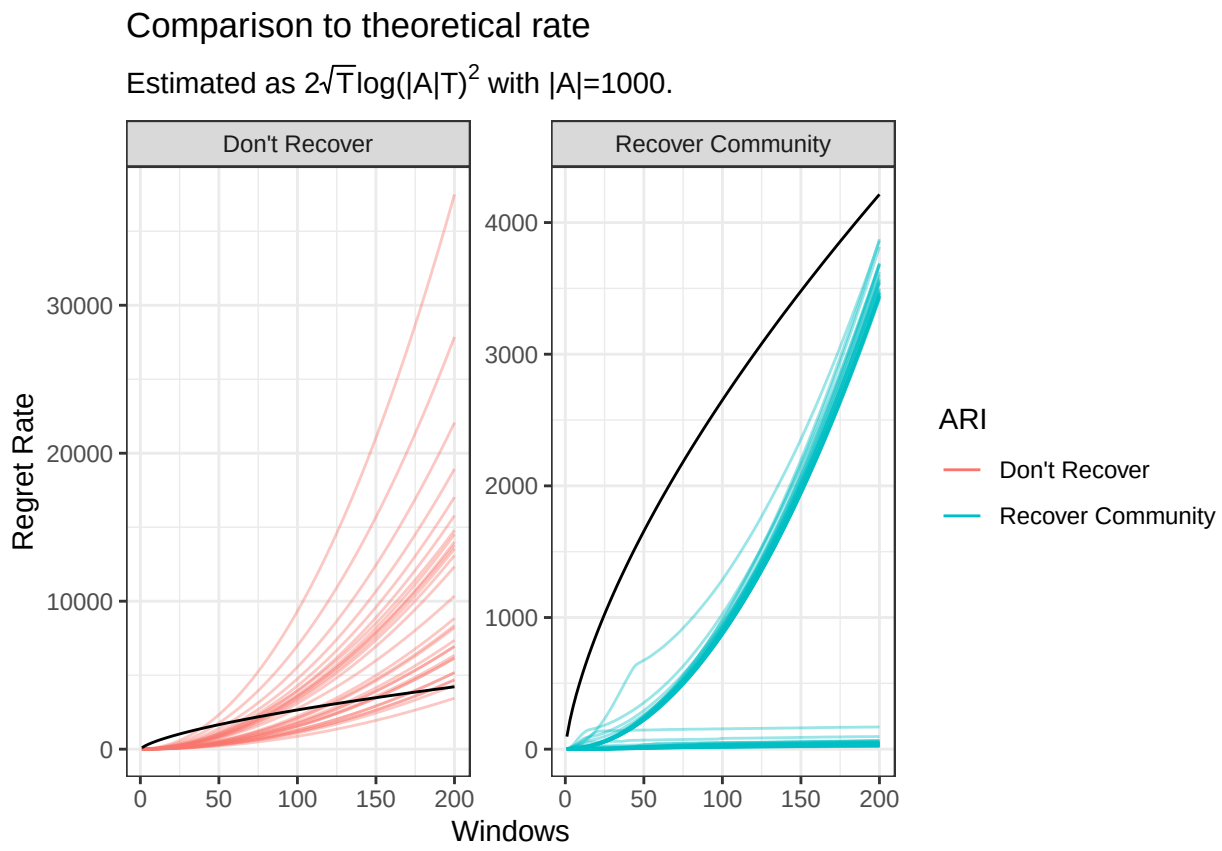
These experiments make a bit less sense. I'm not sure what to think of these, maybe a bug here?

Here the regret we are computing is defined as

$$Regret(T) = \frac{1}{|A|}\left(\sum_{m=1}^{M} \tilde{\ell}_m(\theta^{(m)}|z^*) - \sum_{m=1}^{M} \tilde{\ell}_m(\theta^*|z^*)\right),$$

where $\tilde{\ell}$ is the negative log likelihood in a given window. The $|A|$ is a constant not included in the paper currently.

```
exp_3_data <- readRDS(here("Experiments/exp_results/", "exp3_200.RDS"))

exp3_tidy <- exp_3_data %>% tibble(
  sim = 1:100,
  ARI = map_dbl(., "clust"),
  regret = map(., "regret"),
  dT = list(1:200)
) %>%
  unnest(cols = c(regret, dT)) %>%
  select(sim, ARI, regret, dT)

exp3_tidy %>%
  mutate(ARI = ifelse(ARI == 1, "Recover Community", "Don't Recover")) %>%
  # filter(ARI )
  filter(regret > 0) %>%
  ggplot(aes(dT, regret)) +
  geom_line(aes(group = sim, colour = ARI), alpha = 0.4) +
  stat_function(fun = function(x) 2*sqrt(x) * log(x*1000)^2,
                colour = "black") +
  facet_wrap(~ARI, scales = "free") +
  labs(y = "Regret Rate", x = "Windows",
       title = "Comparison to theoretical rate",
       subtitle =
         TeX("Estimated as $2\\sqrt_{T}\\log(|A|T)^2$ with $|A|=1000$.")) +
  guides(colour = guide_legend(override.aes = list(alpha = 1)))
```



Online Predictive Loss

We can also evaluate the loss for each subsequent window, given the current estimates of τ and B. We compare this to the batch loss, which is computed by fitting the batch estimator to the entire dataset and then averaging the predictions for each window from that model.

Here for the batch estimator we are using the package `ppsbm`, which is well optimised to fit this type of model.

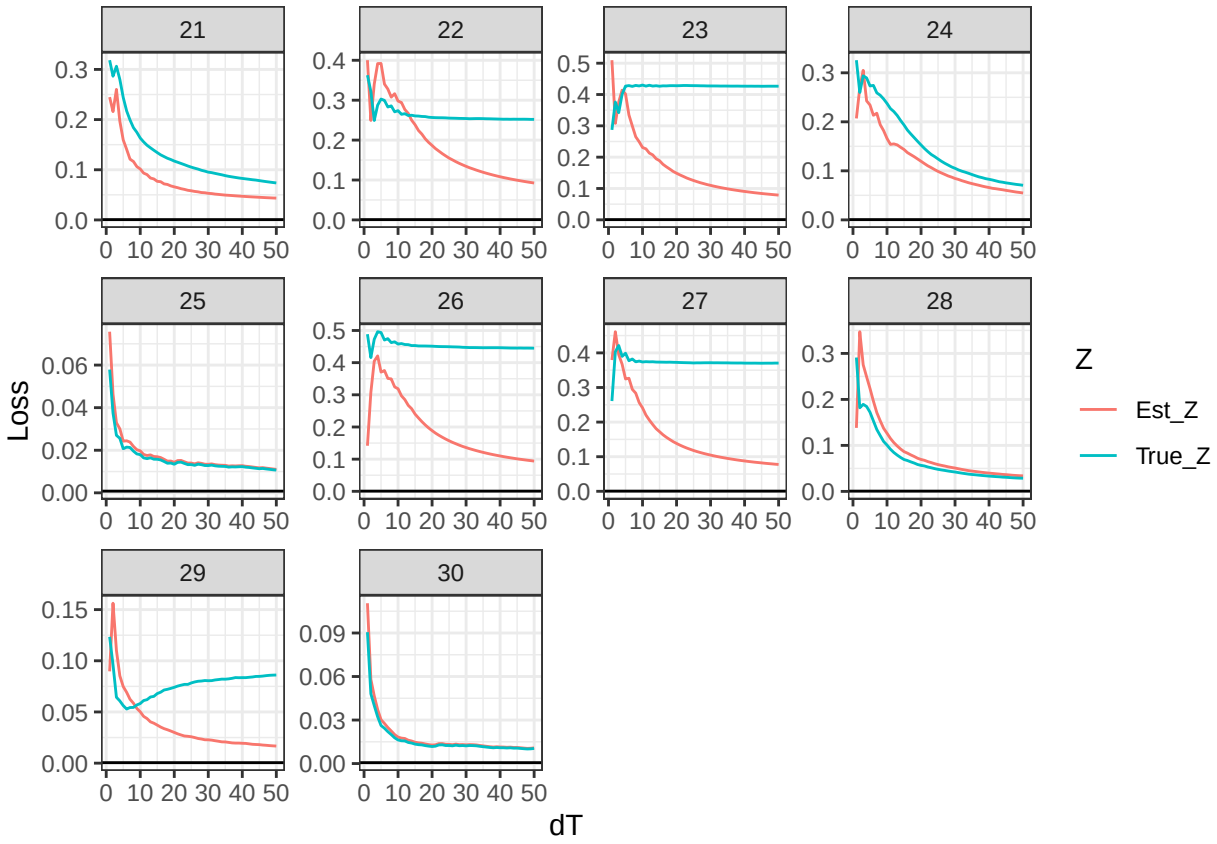
We can also show the estimates when we use the true Z vector instead of the estimate of τ.

There are scenarios where the predictive loss indicates that the estimated Z gives a better result than using the true Z, but this is due to label switching which is not incorporated into the estimates using the true model.

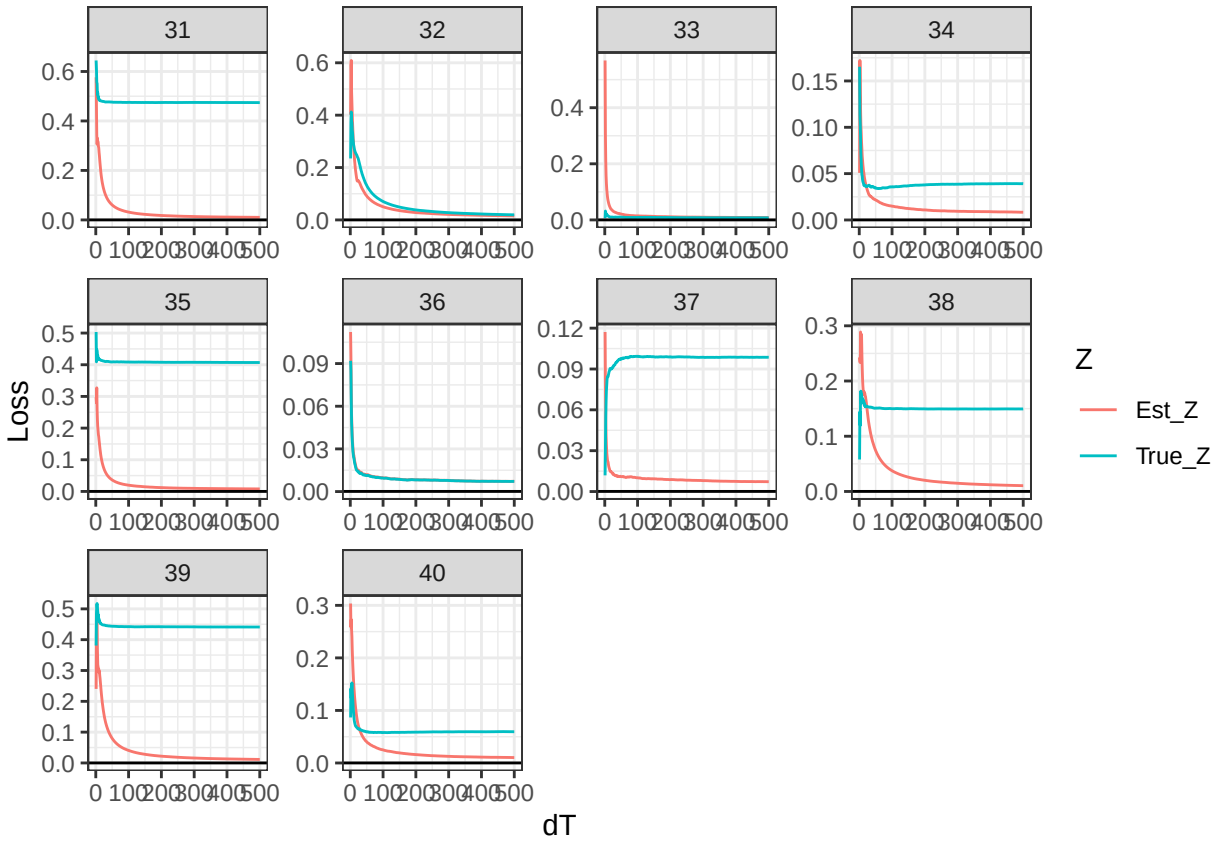
```
exp_4_files <- list.files(path = here("Experiments/exp_results/"),
                          pattern = "exp4_")

exp4_sims <- exp_4_files %>%
  map(~readRDS(here("Experiments/exp_results/", .x))) %>%
  flatten() %>%
  imap(~update_list(., sim = .y)) %>%
  map_dfr(`[`, c("online_loss", "batch_ave_loss", "sim"))


exp_4_tidy <- exp4_sims %>%
```

```
  reduce(data.frame) %>%
  as_tibble() %>%
  rename(Batch_Loss = elt, Sim = elt.1) %>%
  group_by(Sim) %>%
  mutate(Time = max(dT))
```

```
curr_time <- 50
exp_4_tidy %>%
  filter(Time == curr_time) %>%
  ggplot(aes(dT, Loss, colour = Z)) +
  geom_line() +
  geom_hline(aes(yintercept = Batch_Loss),
             exp_4_tidy %>% filter(Time == curr_time)) +
  facet_wrap(~Sim, scales = "free")
```




```
curr_time <- 500
exp_4_tidy %>%
  filter(Time == curr_time) %>%
  ggplot(aes(dT, Loss, colour = Z)) +
  geom_line() +
  geom_hline(aes(yintercept = Batch_Loss),
             exp_4_tidy %>% filter(Time == curr_time)) +
  facet_wrap(~Sim, scales = "free")
```


31
32
33
34
35
36
37
38
39
40
Loss
dT
Z
Est_Z
True_Z