

Exercise 5

Junsong Tang

junsong.tang@stat.ubc.ca

1 sequential updating

1. Denote $\theta|x^{(n)} \sim \pi$. Since $x_i|\theta$ is i.i.d, then $x^{(n)}|\theta \sim \nu_\theta^n$. Suppose μ is the common dominating measure of ν and ρ , i.e. $p(x_i|\theta) = \frac{d\nu}{d\mu}(x_i)$ and $p(\theta) = \frac{d\rho}{d\mu}(\theta)$, then by Bayes' Theorem, we have:

$$\frac{d\pi}{d\mu}(\theta) = p(\theta|x^{(n)}) = \frac{p(x^{(n)}|\theta) \cdot p(\theta)}{\int_{\Theta} p(x^{(n)}|\theta) \cdot p(\theta) d\mu}$$

Hence

$$\pi(d\theta) = \frac{p(x^{(n)}|\theta) \cdot \rho(d\theta)}{\int_{\Theta} p(x^{(n)}|\theta) \cdot \rho(d\theta)} = \frac{\nu_\theta^n(dx) \cdot \rho(d\theta)}{\int_{\Theta} \nu_\theta^n(dx) \cdot \rho(d\theta)}$$

2. Using π as the new prior, then by Bayes's Theorem, the posterior distribution of $\theta|x_{n+1}$ is given by:

$$\begin{aligned} & \frac{p(x_{n+1}|\theta) \cdot \pi(d\theta)}{\int_{\Theta} p(x_{n+1}|\theta) \cdot \pi(d\theta)} \\ &= \frac{\nu_\theta(dx_{n+1}) \cdot \nu_\theta^n(dx) \cdot \rho(d\theta)}{\int_{\Theta} \nu_\theta(dx_{n+1}) \cdot \nu_\theta^n(dx) \cdot \rho(d\theta)} \\ &= \frac{\nu_\theta^{n+1}(dx) \cdot \rho(d\theta)}{\int_{\Theta} \nu_\theta^{n+1}(dx) \cdot \rho(d\theta)} \end{aligned}$$

We can see that is equivalent of using ρ as prior with (x_1, \dots, x_{n+1}) observations.

2 Bayesian inference in the limit of increasing data

```

1 # global
2 library(ggplot2)
3 suppressPackageStartupMessages(library(extraDistr))
4 suppressPackageStartupMessages(library(distr))
5 source("./simple.R")
6 source("./simple_utils.R")
7 set.seed(2025)
8 K = 20
9
10 # 1
11 posterior_distribution = function(rho, n_successes, n_observations) {
12   K = length(rho) - 1
13   gamma = rho * dbinom(n_successes, n_observations, (0:K)/K)
14   normalizing_constant = sum(gamma)
15   gamma/normalizing_constant
16 }
17
18 # 2
19 posterior_mean = function(post_dist) {
20   return (sum((seq(0, K, 1)/K) * post_dist))
21 }

```

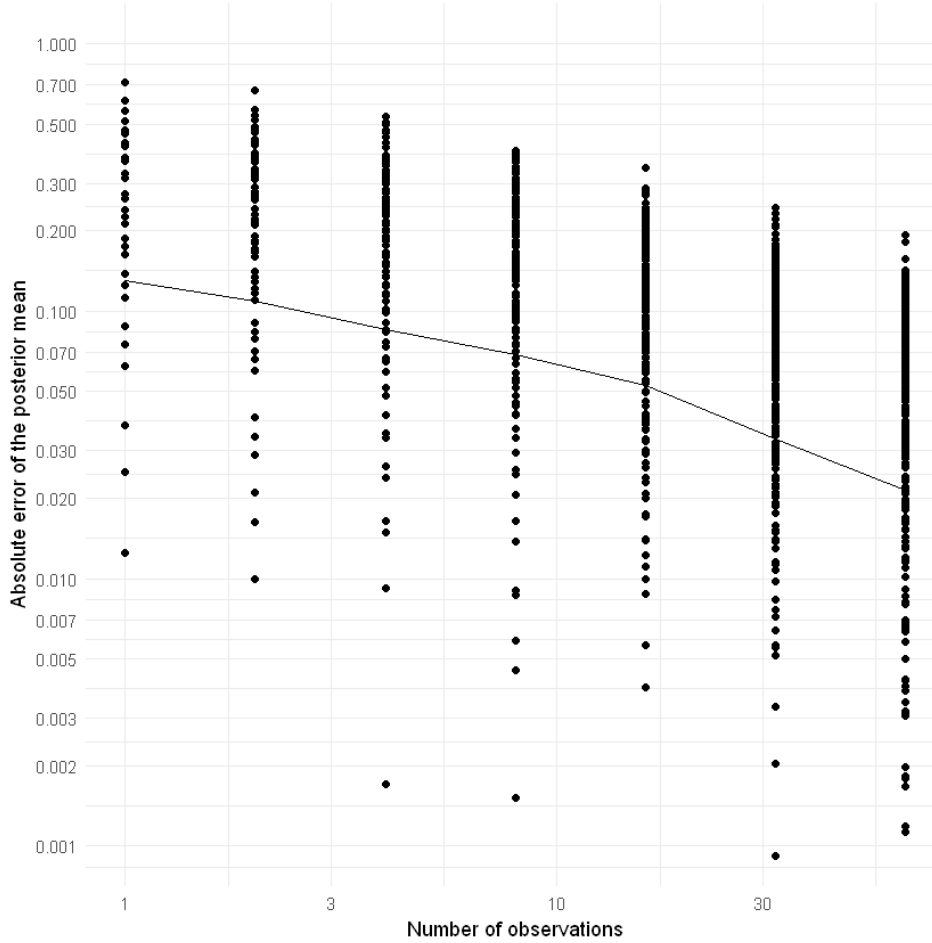
```

3. # 3
2 simulate_posterior_mean_error = function(rho_true, rho_prior, n_observations){
3   dist_p = DiscreteDistribution(supp = (1/K)*(0:K), prob = rho_true/sum(rho_true))
4   p_true = simulate(dist_p)
5   Y = replicate(n_observations, simulate(Bern(p_true)))
6   post_dist = posterior_distribution(rho_prior, sum(Y), n_observations)
7   post_mean = posterior_mean(post_dist)
8   return (abs(p_true - post_mean))
9 }

4. # 4
2 rho_true = rho_prior = 1:(K+1)
3 n_obs_vector <- 2^(0:6)
4 experiment_results = data.frame()
5 for (n_obs in n_obs_vector) {
6   errors = replicate(1000, simulate_posterior_mean_error(rho_true, rho_prior, n_obs))
7   df = data.frame(n_observations=rep(n_obs, 1000), replication=(1:1000), errors=errors)
8   experiment_results = rbind(experiment_results, df)
9 }
10 head(experiment_results)
11 tail(experiment_results)

5. # 5
2 ggplot(experiment_results, aes(x=n_observations, y=errors+1e-9)) + # avoid log(0)
3   stat_summary(fun = mean, geom="line") + # Line averages over 1000 replicates
4   scale_x_log10() + # Show result in log-log scale
5   scale_y_log10(n.breaks=16) +
6   coord_cartesian(ylim = c(1e-3, 1)) +
7   theme_minimal() +
8   geom_point() +
9   labs(x = "Number of observations",
10        y = "Absolute error of the posterior mean")

```



```

61 # 6
2 y7 = mean(experiment_results[experiment_results$n_observations == 2^6, ]$errors)
3 y5 = mean(experiment_results[experiment_results$n_observations==2^4, ]$errors)
4 (log10(y7) - log10(y5)) / (log10(2^6) - log10(2^4))
5 # -0.496968079903831

```

We fetch the corresponding value from the data frame we had in the previous part, and get:

$$\frac{y_7 - y_5}{x_7 - x_5} = \frac{\log_{10}(0.037) - \log_{10}(0.074)}{\log_{10}(2^6) - \log_{10}(2^4)} = -0.497$$

Let $k = \frac{y_j - y_i}{x_j - x_i}, j > i$, then $\frac{\log(\varepsilon_j/\varepsilon_i)}{\log(2^{j-i})} = k$, where ε_i is the i^{th} error. So that means $2^{k(j-i)} = \frac{\varepsilon_j}{\varepsilon_i}$. So the error will be scaled by a factor of $2^{k(j-i)}$ between two errors ε_j and ε_i .

```

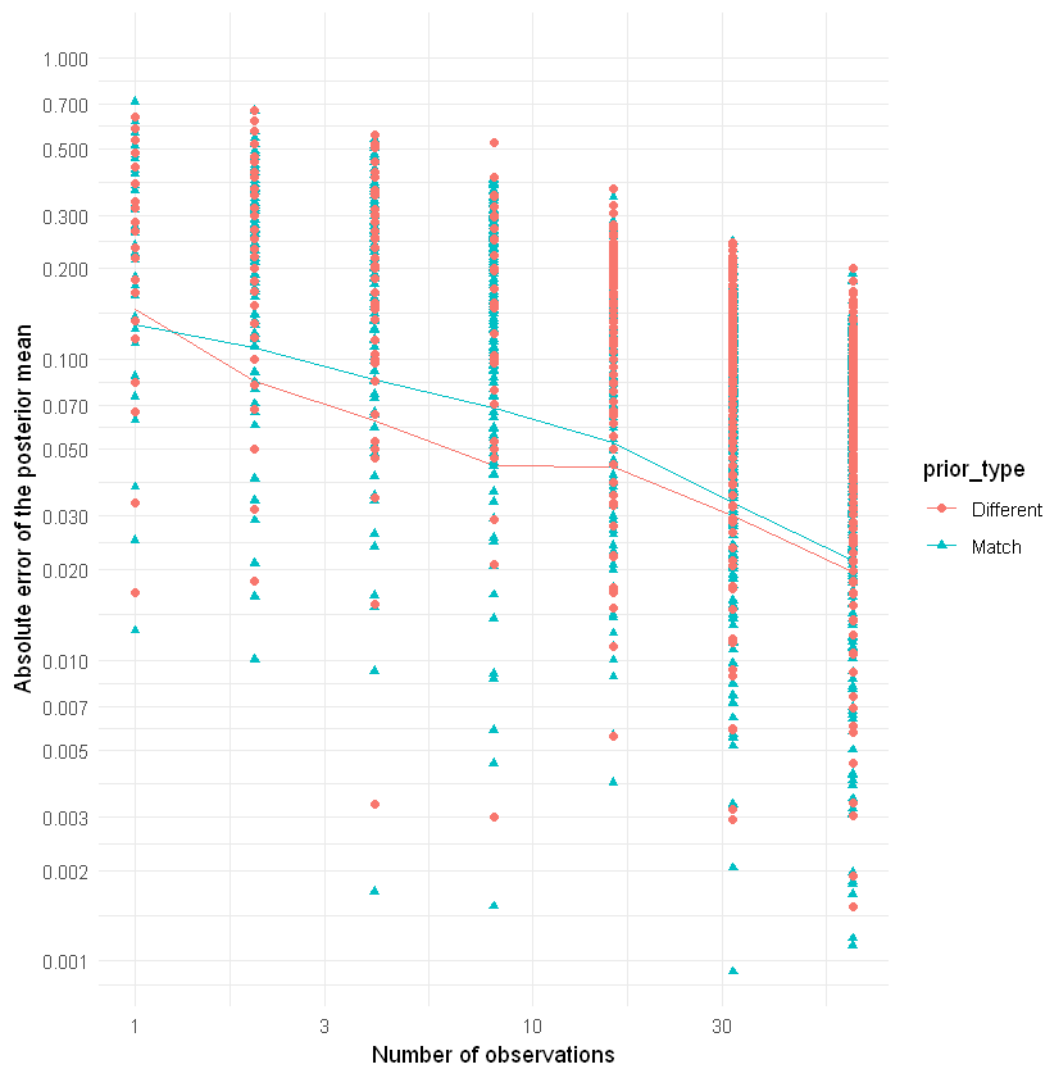
71 # 7
2 rho_true = 1:(K+1)
3 rho_prior = rep(1, K + 1)
4 new_results = data.frame()
5 for (n_obs in n_obs_vector) {
6   # errors = rep(simulate_posterior_mean_error(rho_true, rho_prior, n_obs), 1000)
7   errors = replicate(1000, simulate_posterior_mean_error(rho_true, rho_prior, n_obs))
8   df = data.frame(n_observations = rep(n_obs, 1000), replication = (1:1000), errors =
9     errors)
10  new_results = rbind(new_results, df)
11 }
12 new_results$prior_type = rep("Different", 1000*length(n_obs_vector))
13 experiment_results$prior_type = rep("Match", 1000 * length(n_obs_vector))
14 all_results = rbind(experiment_results, new_results)

```

```

15 ggplot(all_results, aes(x=n_observations, y=errors+1e-9, # avoid log(0)
16                        color=prior_type, shape=prior_type)) +
17   stat_summary(fun = mean, geom="line") + # Line averages over 1000 replicates
18   scale_x_log10() + # Show result in log-log scale
19   scale_y_log10(n.breaks=16) +
20   coord_cartesian(ylim = c(1e-3, 1)) +
21   theme_minimal() +
22   geom_point() +
23   labs(x = "Number of observations",
24        y = "Absolute error of the posterior mean")

```



From the plot above, we can see that when number of observation is small, then the error from the wrong prior is larger than from the true prior. But as the number of observation increases, the error tends to decrease regardless of the initial choice of prior. In addition, the decreasing trend from the wrong prior seems to coincide with the trend from the true prior in the long term.