STAT 447C: Bayesian Statistics **Fall 2024**

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 $x^{(n)}|\theta \sim v_\theta^n$. 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 $(x_1, \ldots, x_{n+1})$ observations.

2 Bayesian inference in the limit of increasing data

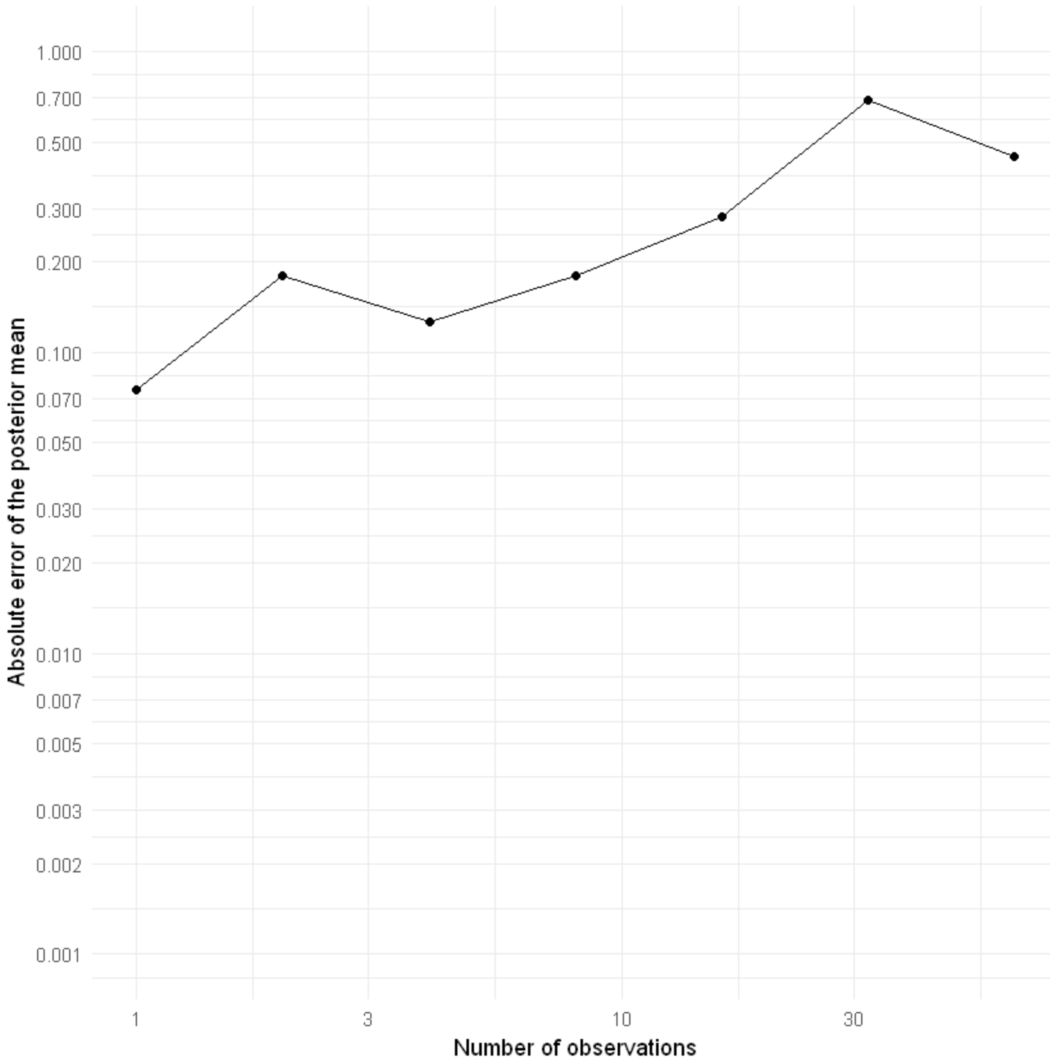
1.
```r
# global
library(ggplot2)
suppressPackageStartupMessages(library(extraDistr))
suppressPackageStartupMessages(library(distr))
source("./simple.R")
source("./simple_utils.R")
set.seed(2024)
K = 20

# 1
posterior_distribution = function(rho, n_successes, n_observations) {
  K = length(rho) - 1
  gamma = rho * dbinom(n_successes, n_observations, (0:K)/K)
  normalizing_constant = sum(gamma)
  gamma/normalizing_constant
}
```

2.
```r
# 2
posterior_mean = function(post_dist) {
    return (sum((seq(0, K, 1)/K) * post_dist))
}
```

3.
```r
# 3
simulate_posterior_mean_error = function(rho_true, rho_prior, n_observations){
    dist_p = DiscreteDistribution(supp = (1/K)*(0:K), prob = rho_true/sum(rho_true))
    p_true = simulate(dist_p)
    Y = rep(simulate(Bern(p_true)), n_observations)
    post_dist = posterior_distribution(rho_prior, sum(Y), n_observations)
    post_mean = posterior_mean(post_dist)
    return (abs(p_true - post_mean))
}
```

4.
```r
# 4
rho_true = rho_prior = 1:(K+1)
n_obs_vector <- 2^(0:6)
experiment_results = data.frame()
for (n_obs in n_obs_vector) {
    errors = rep(simulate_posterior_mean_error(rho_true, rho_prior, n_obs), 1000)
    df = data.frame(n_observations=rep(n_obs, 1000), replication=(1:1000), errors=errors
    )
    experiment_results = rbind(experiment_results, df)
}
head(experiment_results)
tail(experiment_results)
```

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



6. From the plot above, we can eyeball that

$$\frac{y_7 - y_5}{x_7 - x_5} = \frac{\log_{10}(0.46) - \log_{10}(0.28)}{\log_{10}(2^6) - \log_{10}(2^4)} = 0.33$$

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.

7.
```
# 7
rho_true = 1:(K+1)
rho_prior = rep(1, K + 1)
new_results = data.frame()
for (n_obs in n_obs_vector) {
    errors = rep(simulate_posterior_mean_error(rho_true, rho_prior, n_obs), 1000)
    df = data.frame(n_observations = rep(n_obs, 1000), replication = (1:1000), errors =
     errors)
    new_results = rbind(new_results, df)
}

new_results$prior_type = rep("Different", 1000*length(n_obs_vector))

experiment_results$prior_type = rep("Match", 1000 * length(n_obs_vector))

all_results = rbind(experiment_results, new_results)

```

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