STAT 447: Exercise 5

Kevin Liu (94200474)

9 February, 2025

Question 1

1.

$p(\theta|x_1, ..., x_n) \propto p(x_1, ..., x_n|\theta)p(\theta)$
With $x_i|\theta \sim v_\theta$, we have
$p(\theta|x_1, ..., x_n) = \prod_{i=1}^{n} v_\theta(x_i)$
We get $p(\theta|x_1, ..., x_n) \propto p(\theta) \prod_{i=1}^{n} v_\theta(x_i)$

2.

If we add an additional data point x_{n+1}, we get
$p(\theta|x_1, ..., x_n, x_{n+1}) \propto v_\theta(x_{n+1})p(x_1, ..., x_n|\theta)$
We can do the same substitution from part 1
$p(\theta|x_1, ..., x_n, x_{n+1}) \propto v_\theta(x_{n+1})p(\theta) \prod_{i=1}^{n} v_\theta(x_i)$
Which can be simplified to
$p(\theta) \prod_{i=1}^{n+1} v_\theta(x_i)$

Question 2

1.

```
# this is where your R code goes
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
# this is where your R code goes
posterior_mean = function(posterior_distribution, K) {
  x = (0:K) / K

  result = sum(x * posterior_distribution)

  return(result)
}
```

3.

```r
# this is where your R code goes
simulate_posterior_mean_error = function(rho_true, rho_prior, n_observations) {
  K = length(rho_prior) - 1
  x = (0:K) / K

  pt = sample(x, size = 1, prob = rho_true)

  ps = rbinom(n_observations, size = 1, prob = pt)

  pd = posterior_distribution(rho_prior, sum(ps), n_observations)

  pm = posterior_mean(pd, K)

  return(abs(pt - pm))
}
```

4.

```r
# this is where your R code goes
set.seed(1)

K = 20
rho_true = rho_prior = 1:(K+1)
n_obs_vector <- 2^(0:6)
n = 1000

experiment_results = data.frame(n_observations = integer(), replication = integer(), error = numeric())

for (i in n_obs_vector) {
  for (j in 1:n) {
    error = simulate_posterior_mean_error(rho_true, rho_prior, i)
    experiment_results = rbind(experiment_results, data.frame(n_observations = i, replication = j, erro
  }
}

head(experiment_results)
```

```
##   n_observations replication  error
## 1              1           1 0.0875
## 2              1           2 0.1750
## 3              1           3 0.1375
```

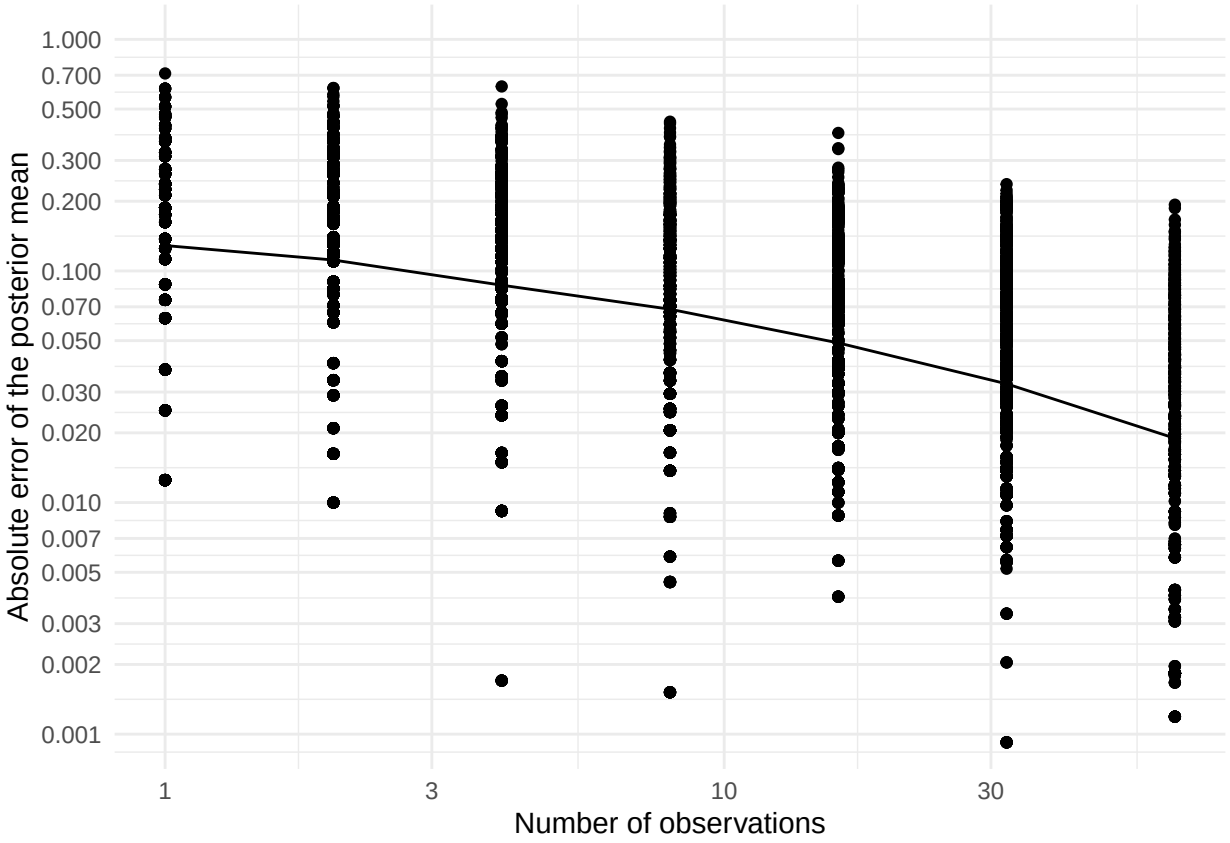
```
## 4              1           4 0.2750
## 5              1           5 0.1625
## 6              1           6 0.1375
```

```
tail(experiment_results)
```

```
##      n_observations replication      error
## 6995             64         995 0.07174783
## 6996             64         996 0.01809958
## 6997             64         997 0.03804049
## 6998             64         998 0.01327434
## 6999             64         999 0.01929102
## 7000             64        1000 0.02355547
```

5.

```
# this is where your R code goes
library(ggplot2)
ggplot(experiment_results, aes(x=n_observations, y=error+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.

```
# this is where your R code goes
slope_est = (log10(experiment_results$error[6]) - log10(experiment_results$error[4])) / (log10(64) - log
print(slope_est)
```

```
## [1] -0.5
```

From the Monte Carlo convergence rate chapter, we had $constant/\sqrt{(number\ of\ iterations)}$, which means the error decay is supposed to scale as $n^{-1/2}$

7.

```
# this is where your R code goes
set.seed(1)

K = 20
rho_true = 1:(K+1)
rho_prior = rep(1, K+1)
n_obs_vector <- 2^(0:6)
n = 1000

new_results = data.frame(n_observations = integer(), replication = integer(), error = numeric())
```

```r
for (i in n_obs_vector) {
  for (j in 1:n) {
    error = simulate_posterior_mean_error(rho_true, rho_prior, i)
    new_results = rbind(new_results, data.frame(n_observations = i, replication = j, error = error))
  }
}

head(new_results)
```

```
##   n_observations replication      error
## 1              1           1 0.16666667
## 2              1           2 0.33333333
## 3              1           3 0.21666667
## 4              1           4 0.11666667
## 5              1           5 0.08333333
## 6              1           6 0.21666667
```

```r
tail(new_results)
```

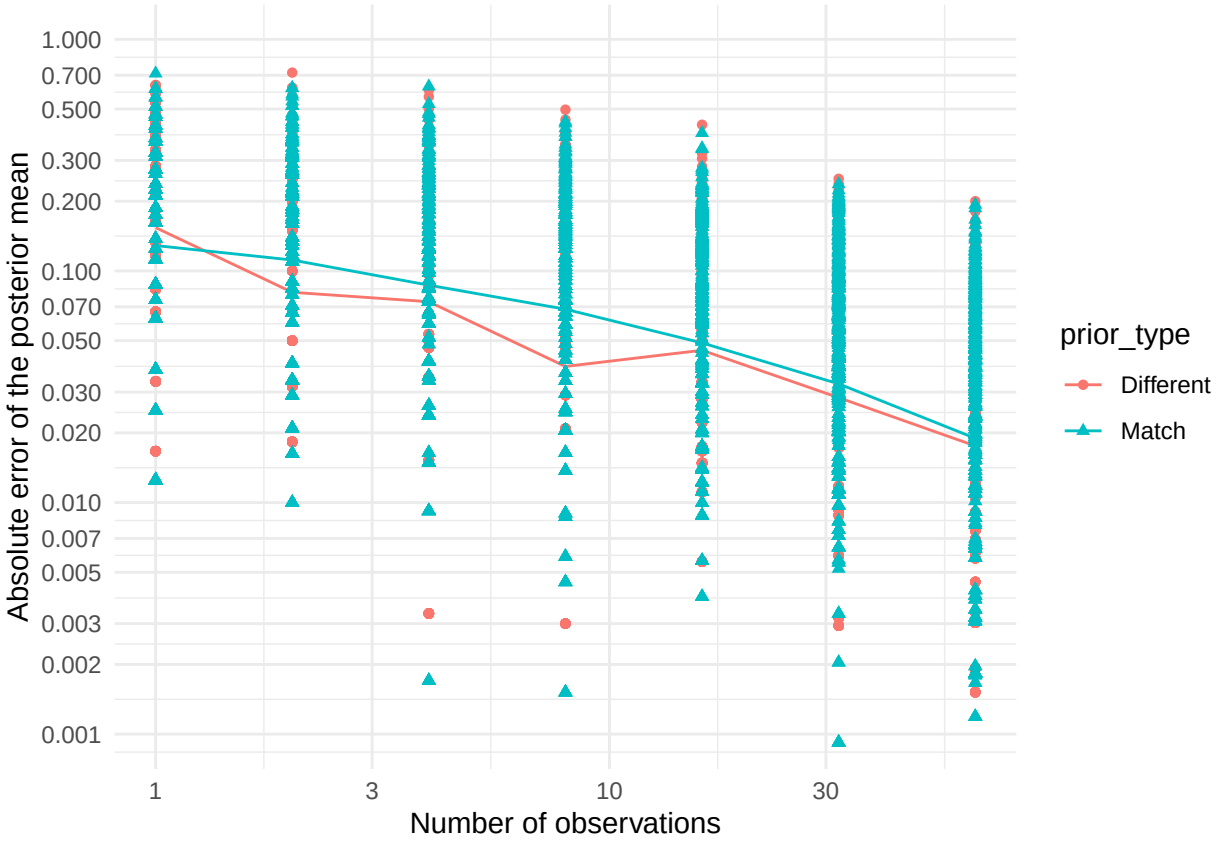
```
##      n_observations replication      error
## 6995             64         995 0.06515152
## 6996             64         996 0.02727266
## 6997             64         997 0.04696970
## 6998             64         998 0.02272703
## 6999             64         999 0.02854616
## 7000             64        1000 0.02878788
```

```r
experiment_results$prior_type = "Match"
new_results$prior_type = "Different"

all_results = rbind(new_results, experiment_results)



ggplot(all_results, aes(x=n_observations, y=error+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")
```



We can see that different priors starts at a higher error for n = 1, but drops below the match prior quickly. However, both priors end up at similar errors at the right side of the plot.