STAT 447 Assignment 5

Caden Hewlett

2024-02-10

Question 1: Sequential Updating

Consider a joint a joint probabilistic model given by

$$\theta \sim \rho, \text{ and } (x_i \mid \theta) \overset{iid}{\sim} \nu_\theta, \text{ where } i \in \{1, 2, \dots, n\}$$

where ρ is a prior distribution for the unknown parameter θ, and $\{x_i\}_{i=1}^n$ is a sequence of observations with conditional distribution ν_θ.

Part 1

Write down the posterior distribution of θ given $\{x_i\}_{i=1}^n$.

In a more verbose sense, let Θ be the random variable for the unknown parameter, and θ be the realization of this random variable under the proposed prior distribution. We can then write the following (*purely for nomenclature reasons*)

$$\rho = p_\Theta(\theta), \text{ where } p_\Theta(\theta) \text{ is the PMF/PDF given by } \rho$$
$$P(X = x \mid \theta) = p_{X|\Theta}(x, \theta), \text{ where } p_{X|\Theta}(x, \theta) \text{ is the PMF given by } \nu_\theta$$

With this in mind, we can write the posterior of θ given $\{x_i\}_{i=1}^n$, where we describe the event that $\{X_i\}_{i=1}^n = \{x_i\}_{i=1}^n$ using Bayes' Rule.

We will begin by using the most verbose notation possible, for complete clarity.

$$P(\Theta = \theta \mid \{X_i\}_{i=1}^n = \{x_i\}_{i=1}^n) = \frac{p_\Theta(\theta) P\big(\{X_i\}_{i=1}^n = \{x_i\}_{i=1}^n \mid \Theta = \theta\big)}{P\big(\{X_i\}_{i=1}^n = \{x_i\}_{i=1}^n\big)}$$

We start by considering the joint likelihood function.

$$P\big(\{X_i\}_{i=1}^n = \{x_i\}_{i=1}^n \mid \Theta = \theta\big) = P\big((X_n = x_n, X_{n-1} = x_{n-1}, \dots, X_1 = x_1) \mid \Theta = \theta\,\big)$$

Then, using intersections, we can write the likelihood as:

$$P\big(\{X_i\}_{i=1}^n = \{x_i\}_{i=1}^n \mid \Theta = \theta\big) = P\Big(\bigcap_{i=1}^n (X_i = x_i) \mid \Theta = \theta\Big)$$

Now, we use the following property of *iid* random variables

$$\text{I.I.D} \implies \forall (i \neq j) \in [1, n],\ (X_i \mid \Theta) \perp (X_j \mid \Theta) \implies \forall (i \neq j) \in [1, n], P(X_i \cap X_j \mid \Theta) = P(X_i \mid \Theta) P(X_j \mid \Theta)$$

Hence, we can write the likelihood as:

$$P\big(\{X_i\}_{i=1}^n = \{x_i\}_{i=1}^n \mid \Theta = \theta\big) = \prod_{i=1}^n \Big(P(X_i = x_i \mid \Theta = \theta)\Big) = \prod_{i=1}^n (\nu_\theta) = (\nu_\theta)^n$$

Which, as you can see, simplifies nicely to $(\nu_\theta)^n$.

Now, our expression simplifies a little bit to the following:

$$P(\Theta = \theta \mid \{X_i\}_{i=1}^n = \{x_i\}_{i=1}^n) = \frac{p_\Theta(\theta)\nu_\theta^n}{P\big(\{X_i\}_{i=1}^n = \{x_i\}_{i=1}^n\big)}$$

If we wished to, we could write the following proportionality directly to conclude:

$$p_{\Theta|X_{1:n}}(\theta, \{x_i\}_{i=1}^n) = \pi_n \propto \rho \cdot \nu_\theta^n$$

Or, letting normalizing constant $\mathcal{Z}_{1:n} = P\big(\{X_i\}_{i=1}^n = \{x_i\}_{i=1}^n\big)$, we can write:

$$\pi_n(\theta) = \frac{\rho \cdot \nu_\theta^n}{P\big(\{X_i\}_{i=1}^n = \{x_i\}_{i=1}^n\big)} = (\mathcal{Z}_{1:n}^{-1}) \cdot \rho \cdot \nu_\theta^n$$

Giving us a nice expression for the posterior, both in proportionality and equality according to a normalizing constant.

Part 2

Suppose now we get an additional data point x_{n+1} with the same conditional distribution ν_θ. Show that using the posterior from part 1 as the *prior* and data equal to just x_{n+1} gives the same posterior distribution as redoing part 1 with the $n+1$ data points.

We wish to show that:

$$\pi_{(n+1)}(\theta) = \frac{p_\Theta(\theta)P\big(\{X_i\}_{i=1}^{n+1} = \{x_i\}_{i=1}^{n+1} \mid \Theta = \theta\big)}{P\big(\{X_i\}_{i=1}^{n+1} = \{x_i\}_{i=1}^{n+1}\big)} = \frac{\pi_n(\theta)P(X_{n+1} = x_{n+1} \mid \Theta = \theta)}{P(X_{n+1} = x_{n+1})}$$

We'll evaluate each expression in turn.

With the first term, we can say directly that:

$$\text{LHS} = \frac{p_\Theta(\theta)P\big(\{X_i\}_{i=1}^{n+1} = \{x_i\}_{i=1}^{n+1} \mid \Theta = \theta\big)}{P\big(\{X_i\}_{i=1}^{n+1} = \{x_i\}_{i=1}^{n+1}\big)} = (\mathcal{Z}_{1:(n+1)}^{-1}) \cdot \rho \cdot \nu_\theta^{n+1} \propto \rho \cdot \nu_\theta^{n+1}$$

By the exact process used in **Part 1**.

More interestingly, we can expand the second term to as follows:

$$\pi_{(n+1)}(\theta) = \frac{\pi_n(\theta)P(X_{n+1} = x_{n+1} \mid \Theta = \theta)}{P(X_{n+1} = x_{n+1})} = (\mathcal{Z}_{n+1}^{-1})\pi_n(\theta)\nu_\theta$$

Then, substituting our expression for $\pi_n(\theta)$ from **Part 1**:

$$\text{RHS} = (\mathcal{Z}_{n+1}^{-1})\Big((\mathcal{Z}_{1:n}^{-1}) \cdot \rho \cdot \nu_\theta^n\Big)\nu_\theta = \big(\mathcal{Z}_{n+1}^{-1} \cdot \mathcal{Z}_{1:n}^{-1}\big)\rho\big(\nu^n \cdot \nu^1\big) = (\mathcal{Z}_{n+1} \cdot \mathcal{Z}_{1:n})^{-1}\rho \cdot \nu_\theta^{n+1} \propto \rho \cdot \nu_\theta^{n+1}$$

Now, consider the following Lemma which we will use without proof.

Lemma

Let $f(y)$ and $g(y)$ be probability distributions acting on the same space $y \in Y$. Let

$$\exists c \in \mathbb{R}^+ \text{ s.t. } \forall y \in Y, f(y) = c \cdot g(y) \implies f \text{ and } g \text{ describe equivalent distributions.}$$

In our specific case, we have:

$$(\mathcal{Z}_{1:(n+1)})^{-1}\text{LHS} = (\mathcal{Z}_{n+1} \cdot \mathcal{Z}_{1:n})^{-1}\text{RHS}$$

So, we can write:

$$\text{Let } c = \frac{(\mathcal{Z}_{n+1} \cdot \mathcal{Z}_{1:n})^{-1}}{(\mathcal{Z}_{1:(n+1)})^{-1}} = \frac{\mathcal{Z}_{1:(n+1)}}{\mathcal{Z}_{n+1} \cdot \mathcal{Z}_{1:n}}, \text{ then LHS} = c \cdot \text{RHS}$$

Which would imply that the two are equivalent under proportionality, and, assuming $c \in \mathbb{R}^+$, are equivalent under normalization.

Question 2: Bayesian Inference in the Limit of Increasing Data

We will use the tractability of the coin bag example to explore the behavior of the posterior distribution as the number of observations goes to infinity. Recall that its joint distribution is

$$p \sim \text{discrete}(\{0, 1/K, 2/K, \ldots, 1\}, \rho)$$
$$y_i | p \overset{\text{iid}}{\sim} \text{bern}(p), \quad i \in \{1, \ldots, n\}$$

Where $\rho = (\rho_0, \rho_1, \ldots, \rho_K)$ is the prior, where $\forall k \in [1, K]$ the proportion of coins of type k in the bag is ρ_k. For the sake of notation (and to avoid confusion with capital P), we will let p_{obs} be a realization of the random variable p.

Part 1

The following simulates the posterior for the above equation.

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

Note, we can verify that passing the following gives our familiar 1/17 result from Assignment 1.

```
assign_1 = posterior_distribution(c(0, 0.5, 1),  3,  3)
# check
all.equal( 1/17, assign_1[2], tolerance = 2e-10)
```

```
## [1] TRUE
```

Part 2

Write a function `posterior_mean` that computes the posterior mean given the output of `posterior_distribution`.

We recall that the posterior mean is computed as follows:

$$\mathbb{E}(p \mid Y = y) = \sum_{\{p\,:\,\pi(p)>0\}} p\,\pi(p)$$

From our "original" coin flip example, we'd have $K = 2$ and

$$\mathbb{E}(p \mid Y = y) = \left(\frac{1}{K}\right)\pi(1) + \left(\frac{2}{K}\right)\pi(2) = \left(\frac{1}{2}\right) \cdot \frac{1}{17} + \left(\frac{2}{2}\right)\frac{16}{17} = \frac{33}{34}$$

Which we will use to test our function.

```
posterior_mean <- function(pi){
  K = length(pi)-1
  return(sum(((0:K)/K)*pi))
}
all.equal(posterior_mean(assign_1), 33/34)
```

```
## [1] TRUE
```

Further, we'd expect a posterior mean of 1 in the case where all the weight was put on the 1 ("always heads") coin.

```
posterior_mean(posterior_distribution(c(0,0,1), 5, 5))
```

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

Part 3

Write another function called `simulate_posterior_mean_error`.

It will perform the following tasks:

a.) Simulate $p_{\text{true}} \sim \text{discrete}(\{0, 1/K, \ldots, K/K\}, \rho_{\text{true}})$, where ρ_{true} is supplied to the function.

This is done as follows (using the Assignment 1 coin flip example)

```
rho_true = c(0, 0.5, 1)/sum(c(0, 0.5, 1));
K = length(rho_true)-1
p_true <- DiscreteDistribution(supp = (0:K)/K, rho_true)
p_true_obs = simulate(p_true)
p_true_obs
```

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

b.) Simulates $\{y_i\}_{i=1}^{n_{\text{obs}}} = \{y_1, y_2, \ldots, y_{n_{\text{obs}}}\} = y_{1:n_{\text{obs}}}$ conditional on the simulated p_{true}. We recall that $\forall i \in [1, n_{\text{obs}}], (y_i \mid p_{\text{true}}) \sim \text{bern}(p_{\text{true}})$, where it should be noted that $n_{obs} = `n_observations` is supplied to the function.

```
n_observations = 3
n = n_observations
y_obs = rbern(n, p_true_obs)
y_obs
```

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

c.) Calls `posterior_distribution` using and the simulated data. Note that ρ_{prior} is the final parameter to the function.

```
# in the coin flip, we assumed all were equally likely
rho_prior = c(1, 1, 1)
pi_obs = posterior_distribution(
  rho_prior, sum(y_obs), 3
)
pi_obs
```

```
## [1] 0.0000000 0.1111111 0.8888889
```

d.) Computes the posterior mean $\mathbb{E}(p \mid y_{\text{obs}})$, given the observed posterior $\pi_{\text{obs}}(p)$.

```
post_mean = posterior_mean(pi_obs)
post_mean
```

```
## [1] 0.9444444
```

e.) Compute the absolute error ε_{obs}

$$\varepsilon_{\text{obs}} = \left| p_{\text{true}} - \mathbb{E}(p \mid y_{1:n_{\text{obs}}}) \right|$$

.

```
epsilon_obs = abs(p_true_obs - post_mean)
epsilon_obs
```

```
## [1] 0.05555556
```

Then we combine this all into one process

```
set.seed(200)
simulate_posterior_mean_error = function(rho_true, rho_prior, n_observations){
  # step 1
  p_true = DiscreteDistribution(supp = (0:K)/K, rho_true)
  p_true_obs = simulate(p_true)
  # print(p_true_obs)
  # step 2
  epsilons = c()
  for (value in n_observations){
    y_obs = rbern(value, p_true_obs)
    # step 3
    pi_obs = posterior_distribution(rho_prior, sum(y_obs), value)
    post_mean = posterior_mean(pi_obs)
    # step 4
    epsilons = c(epsilons, abs(p_true_obs - post_mean))
  }
  return(epsilons)
}
# plot(simulate_posterior_mean_error(rho_true, rho_prior, 1:1000), type = 'l')
```

Part 4

Now, we use the following setup, normalizing the given ρ values so it can work with the `DiscreteDistribution` function.

```
K = 100
rho_true = rho_prior = rep(1, K+1) / (K+1)
```

Applying it to each $n \in \{1, 2, 4, 8, 16, 32, 64\}$

```
n_obs_vector <- 2^(0:6)
simulate_posterior_mean_error(rho_true, rho_prior, n_obs_vector)
```

```
## [1] 0.21000000 0.29373134 0.20661111 0.15999994 0.12666667 0.09882353 0.03575758
```

Repeating each case $M = 1,000$ times.

```
set.seed(101)
M = 1000
rsf = c()
for (m in 1:M){
  rsf = c(rsf, simulate_posterior_mean_error(rho_true, rho_prior, n_obs_vector))
}
experiment_results = data.frame(errors = matrix(rsf))
experiment_results$replication = rep(1:M, each = 7)
experiment_results$n_observations = rep(n_obs_vector, times = M)
```

Then we can display the `head` and `tail` of this data frame.

```
kable(head(experiment_results))
```

errors	replication	n_observations
0.2900000	1	1
0.1337313	1	2
0.2866111	1	4
0.2844328	1	8
0.0466667	1	16
0.0858824	1	32

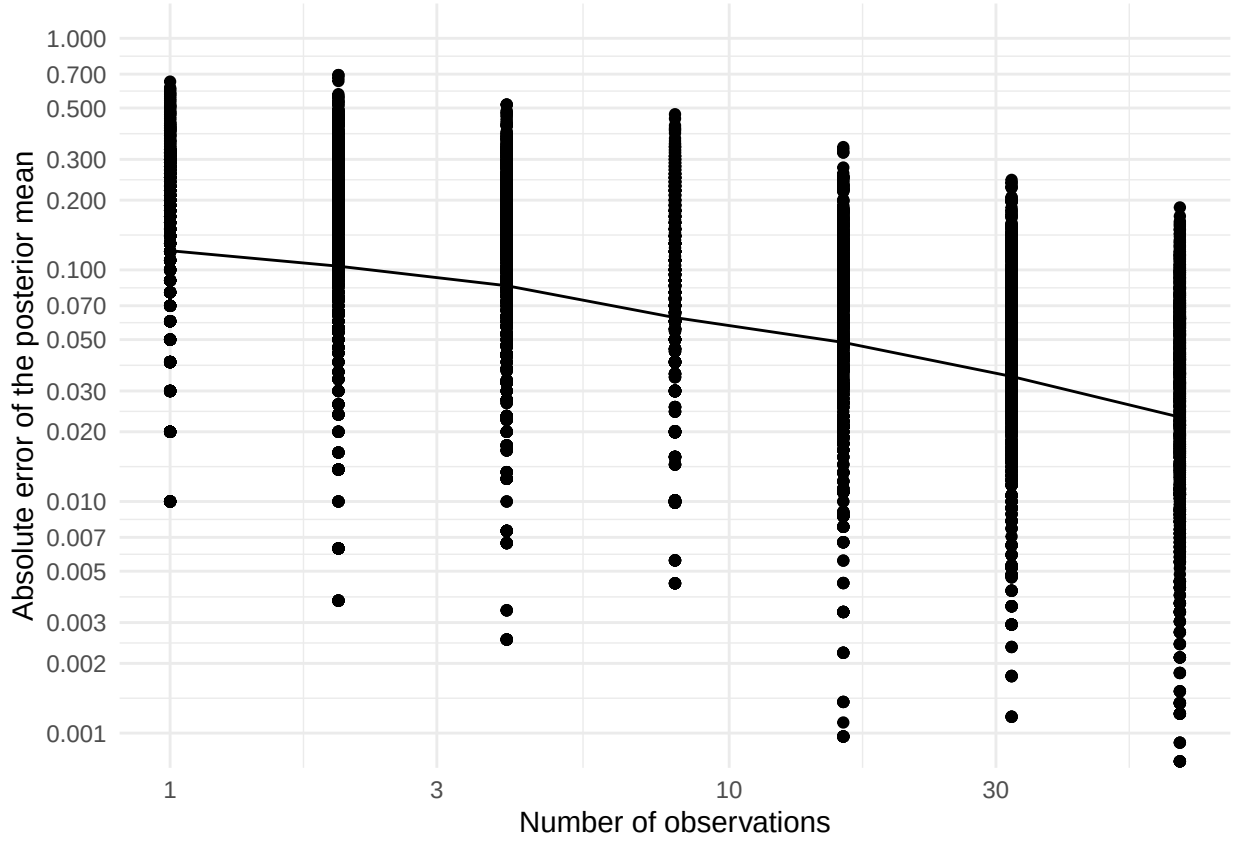
```
kable(tail(experiment_results))
```

	errors	replication	n_observations
6995	0.1062687	1000	2
6996	0.0225345	1000	4
6997	0.3600000	1000	8
6998	0.0266672	1000	16
6999	0.0658824	1000	32
7000	0.0187879	1000	64

Part 5

Visualize the above data using a log-log plot using the following code:

```
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")
```



Part 6

We'll estimate the slope with the following function, again letting ε be the error.

$$\hat{m} = \frac{y_7 - y_5}{x_7 - x_5} = \frac{\log_{10}(\bar{\varepsilon}_7) - \log_{10}(\bar{\varepsilon}_5)}{\log_{10}(n_7) - \log_{10}(n_5)}$$

Where, as shown above, x is $\log_{10}(n_i)$, is y is $\log_{10}(\bar{\varepsilon}_i)$, where $\bar{\varepsilon}$ is the mean absolute error of the posterior mean and the subscript indicates the column of points from left to right.

```
x7 = log10(n_obs_vector[7])
x5 = log10(n_obs_vector[5])
y7 = log10(mean(experiment_results$errors[
            experiment_results$n_observations == n_obs_vector[7]]))
y5 = log10(mean(experiment_results$errors[
            experiment_results$n_observations == n_obs_vector[5]]))
# y7 = log10( mean(all_simulations[, 7]) )
# y5 = log10( mean(all_simulations[, 5]) )
#
#head(experiment_results)
m = (y7 - y5)/(x7 - x5)
m
```

```
## [1] -0.4585616
```

What can you deduce about the scaling law of the error?

Well, if we recall the notation of $y = mx + b$, the slope m can be employed linearly as

$$\log_{10}(\bar{\varepsilon}_i) = \hat{m}\log_{10}(n_i) + c, \text{ for some c } \in \mathbb{R}$$

Then, by rearrangement, we would have:

$$\bar{\varepsilon}_i = Cn_i^{\hat{m}}, \text{ where } C = 10^c.$$

Notably, our slope is negative. Assuming C is negligible for the relationship between the variables, we can write the error values as:

$$\bar{\varepsilon}_i \propto n_i^{-\hat{m}} \approx n_i^{-0.46}$$

From this approximate relationship, to achieve an additional one decimal point of accuracy in $\bar{\varepsilon}_i$ for a given index i of the vector of $\vec{n}$ observation we need to solve the following for k:

$$0.10\bar{\varepsilon} = 0.10(n_i^{-0.46}) = (kn_i)^{-0.46}$$

Simplifying we see:

$$0.10(n_i^{-0.1}) = (k)^{-0.46}(n_i^{-0.1}), \text{ hence } 0.10 = (k)^{-0.46}$$

$$\log_{10}(0.10) = -0.46\log_{10}(k), \therefore k = 10^{1/0.46} = 151.6115 \approx 150$$

This means that - assuming my math here is correct and the approximation is decent - in order to increase the accuracy of $\bar{\varepsilon}$ by an additional decimal point the sample size must be increased by a factor of approximately 150.

We made quite a few simplifying assumptions for this metric, and the precise calculations here could be wrong. The larger point is to demonstrate that the scaling law of the error declines at a very slow rate.

Part 7

We repeat Part 4, but now we have the following:

```
rho_true = rep(1, K+1)
rho_true = rho_true/sum(rho_true)
rho_prior = 1:(K+1)
rho_prior = rho_prior/sum(rho_prior)
```

Our simulation is here (to be called `new_results`.)

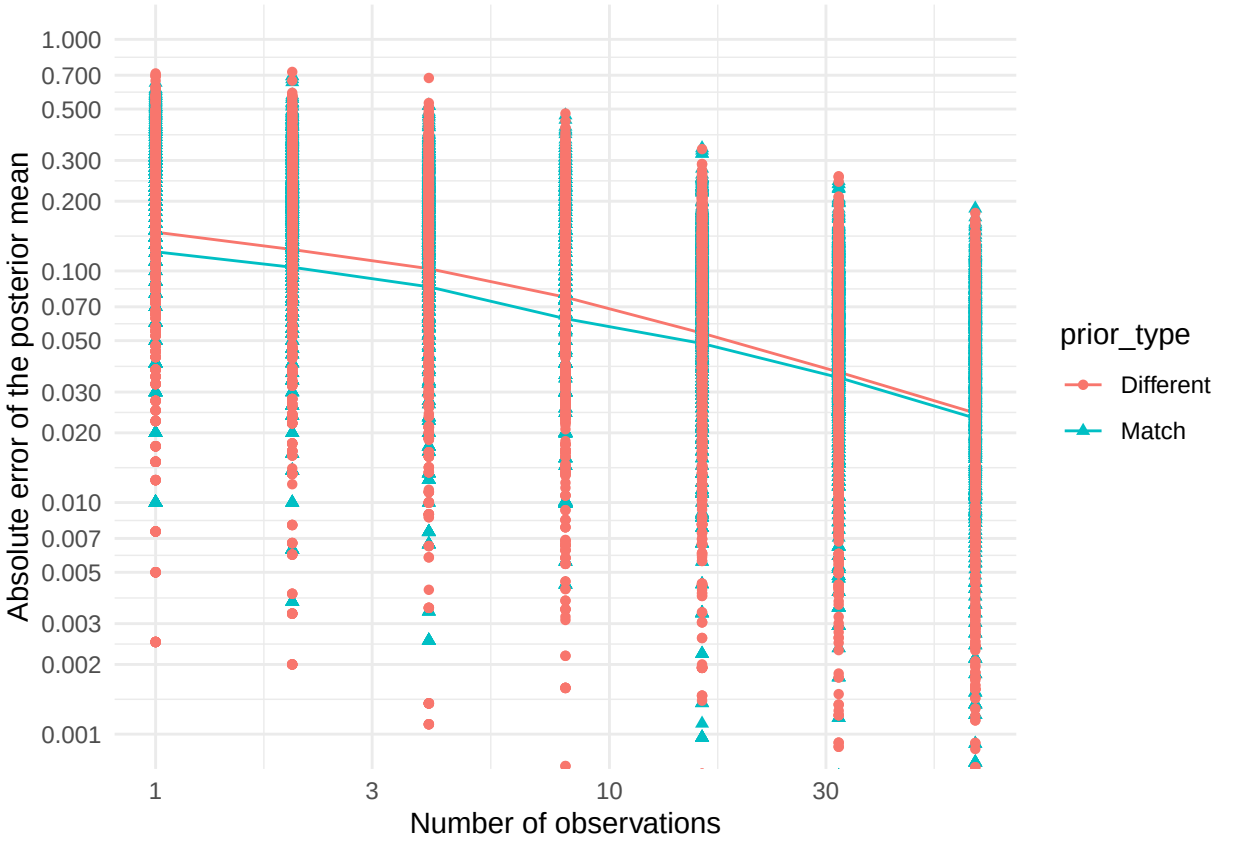
```
rsf = c()
for (m in 1:M){
  rsf = c(rsf, simulate_posterior_mean_error(rho_true, rho_prior, n_obs_vector))
}
new_results = data.frame(errors = matrix(rsf))
new_results$replication = rep(1:M, each = 7)
new_results$n_observations = rep(n_obs_vector, times = M)
```

Then, we can plot the results. First, however we must make a unified data frame called `all_results`.

```
new_results$prior_type = "Different"
experiment_results$prior_type = "Match"
all_results = rbind(experiment_results, new_results)
```

Now, we can make the unified plot.

```
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")
```



The main visual difference is the faster convergence rate of the "Different" plot in absolute error difference. In other words, the errors begin larger for small n, but converge to nearly the same value by the time we reach n_7. This implies that the even a prior that is very different from ρ_{true} will converge in ε to the "correct" $\rho_{\text{prior}} = \rho_{\text{true}}$ error rate.

This initial "ramping down" of the error rate for the "Different ρ" plot makes sense, as the posterior expectation of the incorrect model will quickly approach the true value p_{true} as $n \uparrow$, regardless of the prior (by the nice properties of Bayesian methods we have discussed often in this course.). Then, the remaining difference between both models is due to the "simulation error", which will slowly converge by the reasoning discussed in the previous question.