Project 2 - Bayesian Data Analysis

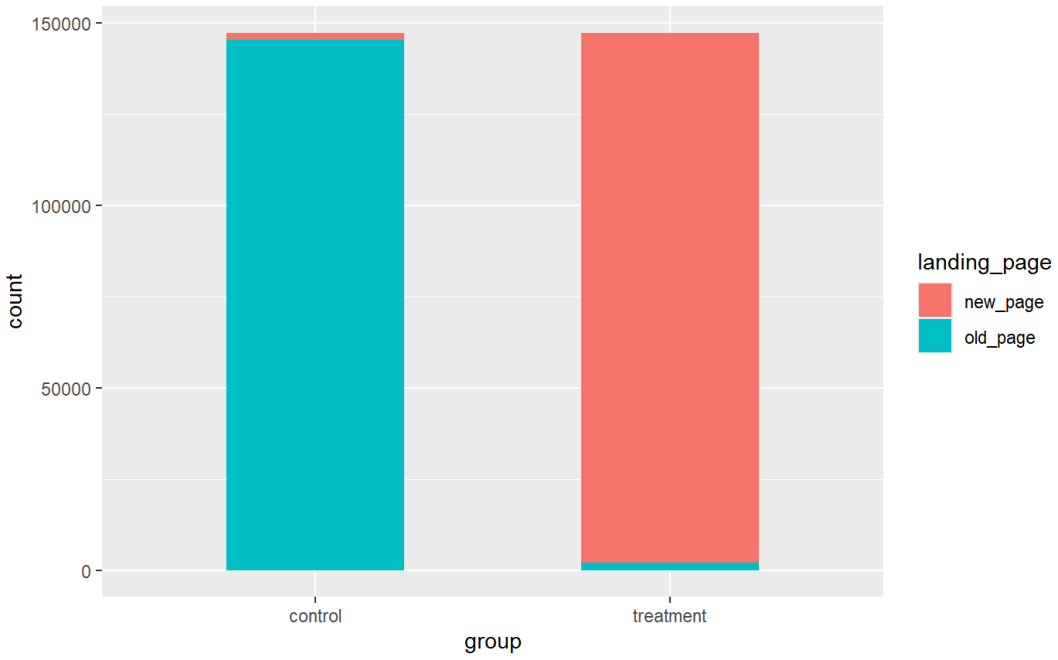
Jiayu Tan, Elizabeth Shevchenko, Léo Séror, Daniel Verdon

Bayesian AB Testing

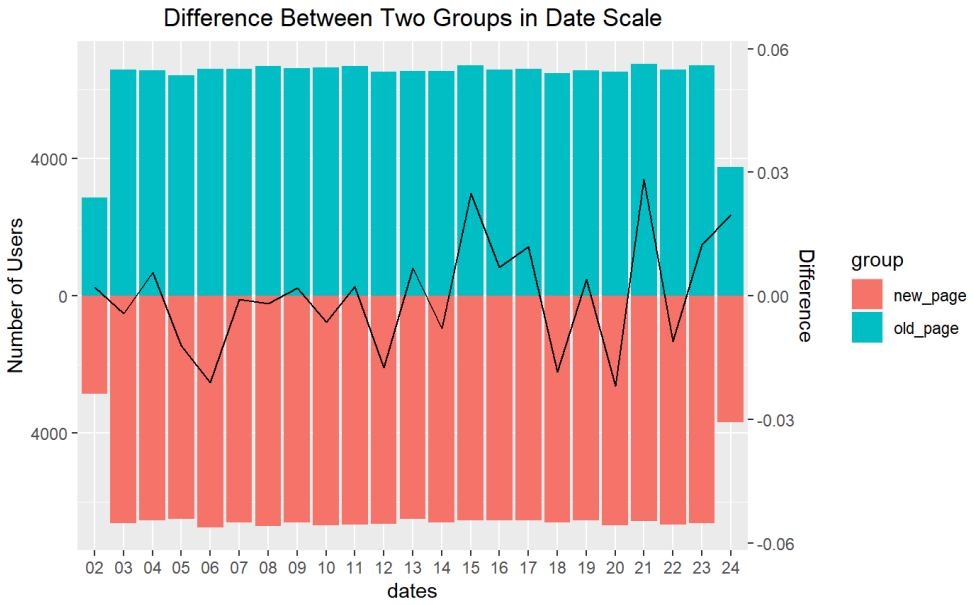
The dataset we are considering is 5 columns wide and 294,478 rows long, and sets out information relating to a AB test, in which clients are shown a webpage at random, a new version or old version of the webpage, and whether or not they were "converted" (performed a sought after action). The data dictionary is as follows:

1. **user_id:** A numerical identifier, represented as a six-digit number, uniquely associated with each user who visits the website.
2. **timestamp:** A character variable that captures the date and time at which the user accessed the website.
3. **group:** A character/categorical variable with two categories: "control" and "treatment". This indicates the experimental condition assigned to the user. The "control" group refers to users exposed to the old page, while the "treatment" group refers to users exposed to the new page.
4. **landing_page:** A character/categorical variable with two levels: "new" and "old". This shows which version of the page the user actually viewed during their visit.
5. **converted:** A numeric/binary variable (0 or 1) indicating whether the user converted during their visit. It can also be interpreted as a boolean value, with 1 representing a conversion and 0 representing no conversion.

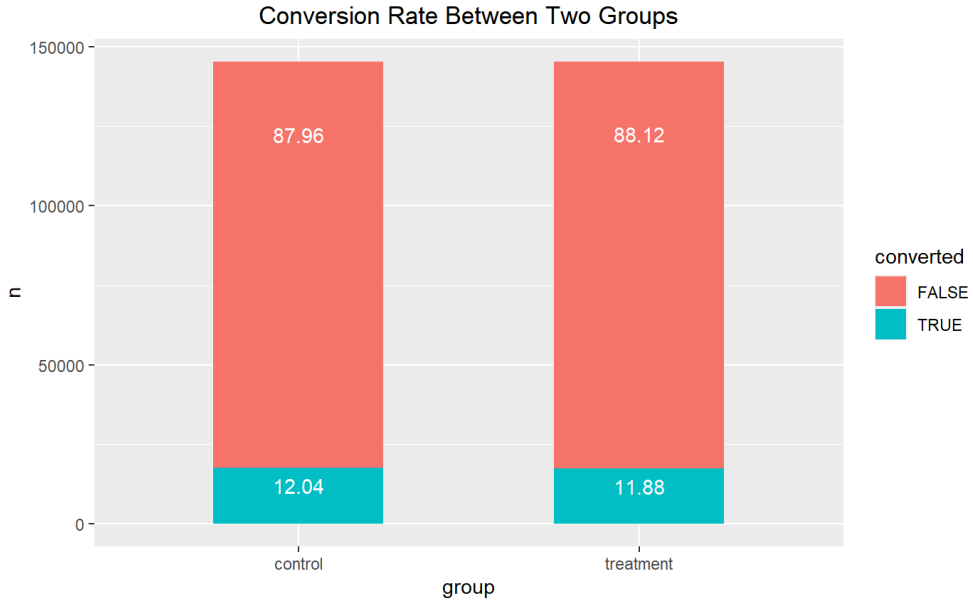
Let's first observe the distribution of users in the "control" and "treatment" groups.



From the figure above, we can see the number of users in the control group and treatment group are almost the same, but we have a small number of incongruous units in our data set. A very small percentage of users in both the control and treatment groups visited pages they should not have. All users in the control group should be directed to the old page, and users in the treatment group should be redirected to the new page. In our data exploration, we will treat these instances as errors and exclude them from our dataset to maintain data integrity.



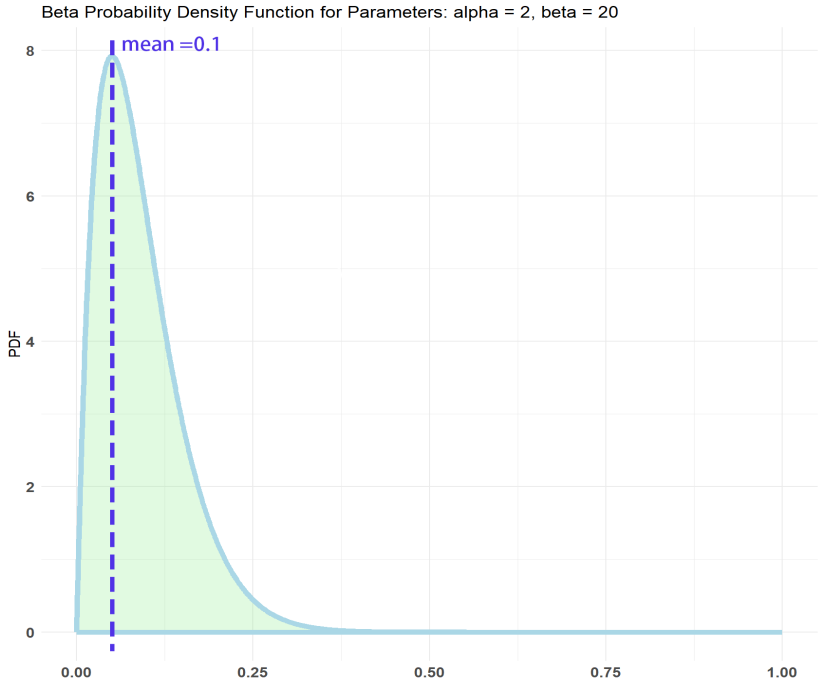
Next, analyzing the **timestamp** data reveals that all user visits occurred in the month of January. This uniformity in the time frame allows us to categorize and visualize the data based on dates, ensuring a consistent comparison. From the graph below (on the next page), we see how the proportion of users visiting the old and new pages remained roughly balanced (50-50) throughout the month, with no day showing a deviation larger than 3% from this balance.



In terms of conversion rates, the control and treatment groups exhibited very close values, with the old page (control group) showing only a 0.16% higher conversion rate than the new page (treatment group). This observation raises a critical question: Despite the minuscule difference, should we consider the old page more effective in terms of conversion rate than the new page? To answer this question, we employ a Bayesian A/B testing approach to find out which of the two webpage variations yields a higher conversion rate. We utilize a beta distribution as our prior distribution, a popular choice for modeling binary data such as conversions or clicks.

$$Prior = Beta(\alpha, \beta),\ \alpha = 2,\ \beta = 20$$

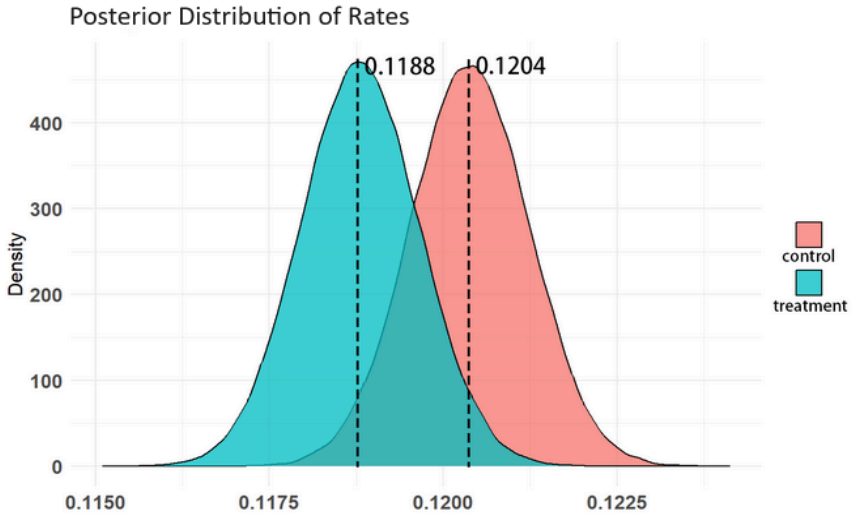
$$Mean = \frac{\alpha}{\alpha+\beta} = \frac{2}{2+20} \approx 0.1$$



Here we see that the average conversion rate is 10% (peak of the distribution) prior to data collection, suggesting that, on average, one in ten users is expected to convert. The beta distribution is particularly advantageous because it allows for real-time model updates as new data becomes available.

$$Posterior = Beta(\alpha + s, \beta + n - s)$$

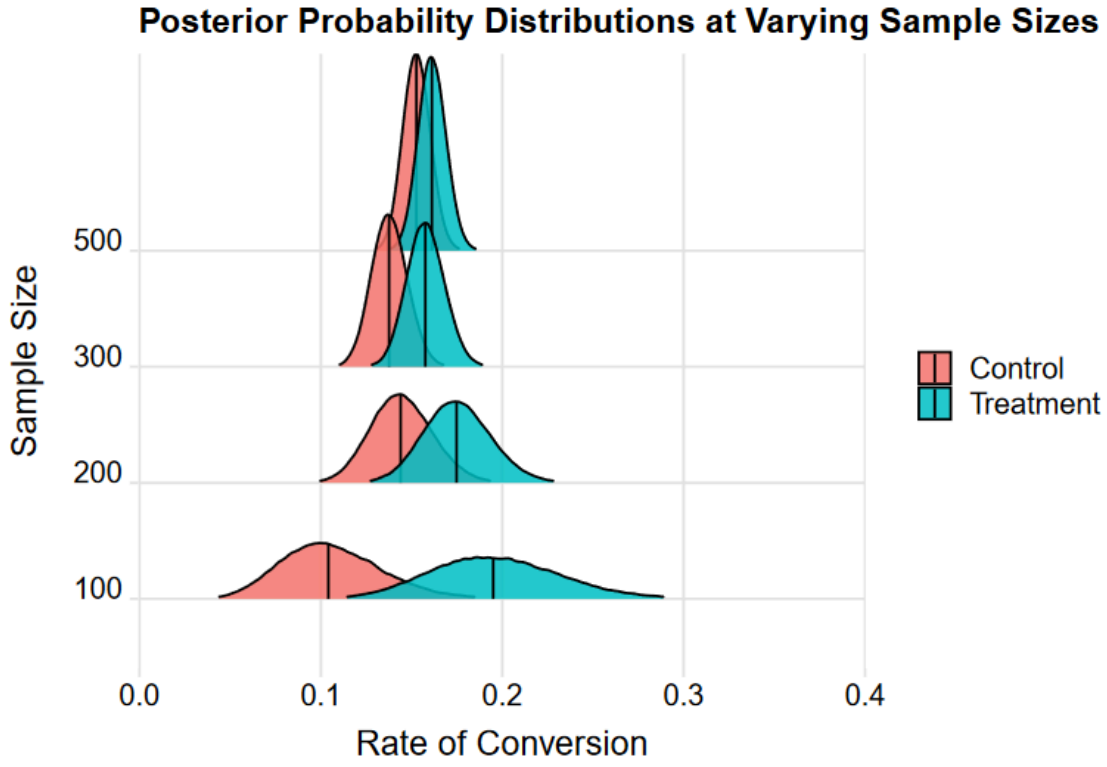
Here s denotes the number of converted users, n is the number of total users. In a nutshell, adding the number of successful conversions in the newly observed data to α and the number of failures to β allows us to calculate the beta posterior distribution for rate of conversion estimation. The following figure shows the upgraded posterior distribution with respect to treatment and control groups.



From the figures above, we observe that the control group outperforms the treatment group in terms of conversion rate, something that was already noted prior to this figure. However, due to the overlap between the two distributions, we cannot draw definitive conclusions solely from these plots.

To quantify the probability that one group outperforms the other, we employ Monte Carlo simulations. This technique involves randomly sampling from a distribution, with samples from high-probability intervals appearing more frequently. We randomly take 100,000 values from both beta distributions and record the number of values in the control group that are greater than those in the treatment group, then divide by the number of samples. This gives us a probability illustrating that the conversion rate of the control group is greater than that of the treatment group. The code in *Appendix 1* gives us that the probability of the control group having a better conversion rate than the treatment group is approximately 90%. It is important to note that the actual value may vary slightly due to the stochastic nature of Monte Carlo simulations.

To illustrate the effect on the shape of the posterior distributions for the control and treatment in the bayesian AB testing for the effectiveness of our new page, we use the code set out in *Appendix 2* to generate the probability distributions for the rate of conversion of each webpage, using randomly sampled datasets at sizes of 100, 200, 300, and 500.

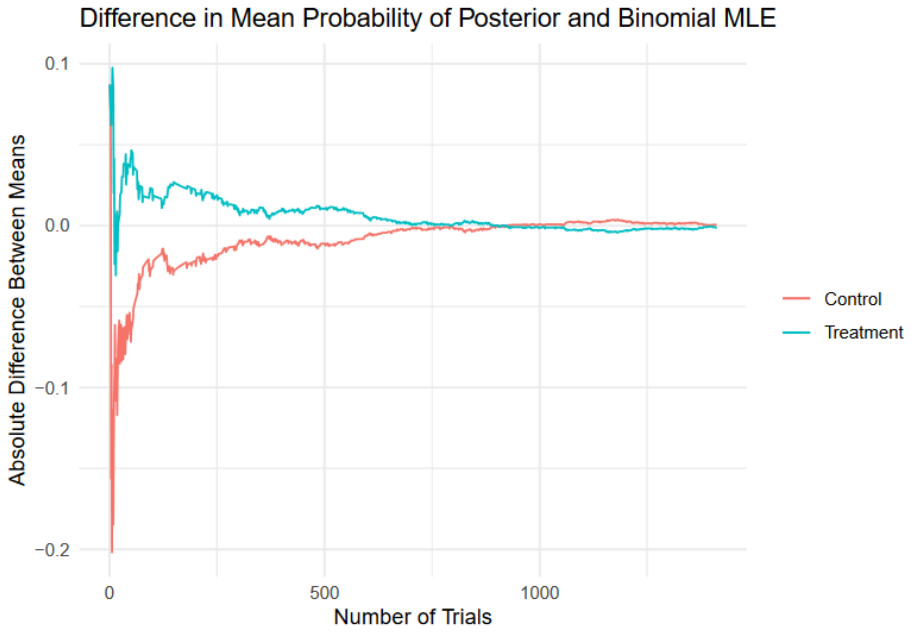


Sample Size	n=100	n=200	n=300	n=500
Control	0.106	0.145	0.138	0.156
Treatment Mean	0.197	0.175	0.158	0.166
P(B better than A)	0.978	0.8892	0.921	0.836

In the plot above, we see that the posterior distributions of the control and treatment rate of conversions are quite widespread at lower sample sizes and have much sharper peaks as the sample size increases. Here, we use the same method as above to calculate the mean of the distributions (averaging probabilities provided by a Monte Carlo simulation of the beta distribution and dividing by the sample size). Furthermore, the probability that the treatment has a higher conversion rate than the control page (B better than A) is obtained by generating a sample of 10000 probabilities from both distributions, and averaging the indicator function of the treatment sample being greater than the control sample. As one might assume, as we increase the sample size, the posterior distributions converge towards the posterior distributions that were generated previously using the full dataset, alongside the mean values for the control and treatment (seen by the vertical lines inside the distributions), as well as the probability that the new webpage has a higher conversion rate than the old webpage (here we see it decreasing as sample size increases).

This begs the question, can we quantify the effect that our sample size has on our resulting posterior distributions? By using the code set out in *Appendix 3*, we can estimate the rate of conversion in our posterior distribution and compare it to the maximum likelihood estimation of our binomial likelihood distribution:

$$\hat{p} = \frac{\sum_{i=1}^{n} x_i}{n}$$



By updating our AB test one Bernoulli trial at a time, we can see that the absolute difference between the mean of the posterior beta distribution and the binomial likelihood distribution converges towards zero. In the chart above, we set a stopping point where the absolute difference of the mean rate of conversion being less than 0.001, which had to hold true for the next 100 trials. Considering that the posterior distribution is proportional to the product of the prior distribution and likelihood, the result in the chart above indicates that the likelihood is playing the dominant role in shaping the posterior distribution, and the prior is effectively irrelevant. Our sample size that made the prior distribution effectively irrelevant was 1411, but this varies depending on the level of information the prior has, in addition to the random sample of data you take, so it must be considered on a case by case basis.

Bayesian Linear Regression

The dataset under examination comprises 58976 entries spanning across 29 columns, encompassing a variety of data types such as numerical, categorical and object. Each entry corresponds to a unique hospital admission, identified by a six-digit number under the column “hadm_id.” The dataset encapsulates diverse patient information, including demographic details like gender and age, hospital interactions or procedures (e.g., “NumCallouts” for the number of callouts), and diagnostic information (“AdmitDiagnosis”) amongst others.

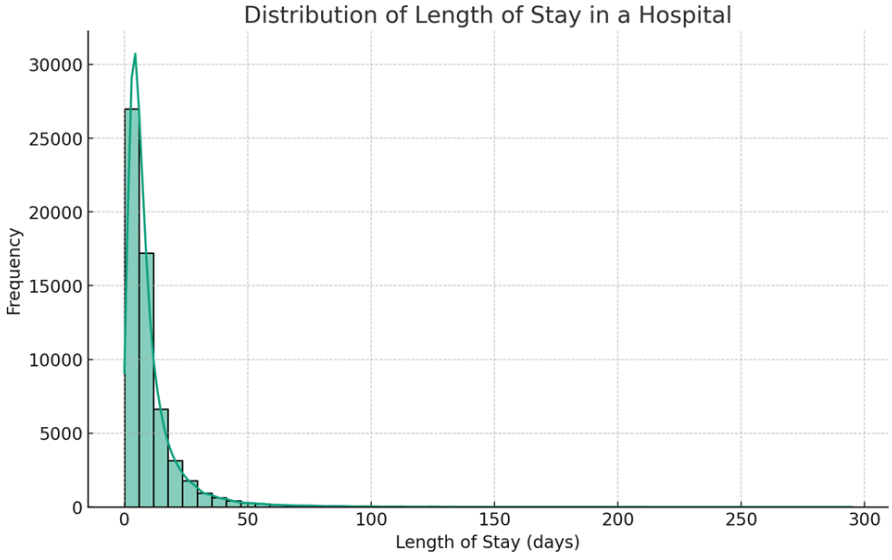
There are a lot of places for exploration in this dataset. For example, we can look at the “ExpiredHospital” column, a boolean type data indicating patient mortality during hospitalization, to discover that about 9.9% of the entries indicate a patient’s death in the hospital ($\frac{5854\ entries\ indicating\ a\ patient's\ death}{58976\ entries} \approx 0.099$). However, to keep things simple, our exploration will mainly focus on the main subject of our task, the length of stay in a hospital which is found under the column “LOSdays.” And so, in subsequent sections, we will only highlight categories that we will determine are relevant to the length of stay in a hospital.

First, let’s explore uniquely the data of the length of stay in a hospital for each entry. The type of data is numerical, and the length of stay is measured by days. We interpret this as, for example, 14.54 would mean that a patient stayed in a hotel for 14 days and a little over a half a day. The key statistic for “LOSdays” reveal the following:

- **Missing values:** There are no missing values.
- **Mean:** 10.11 days
- **Median:** 6.46 days
- **Min:** 0 days
- **Max:** 294.63 days
- **Standard deviation:** 12.46 days

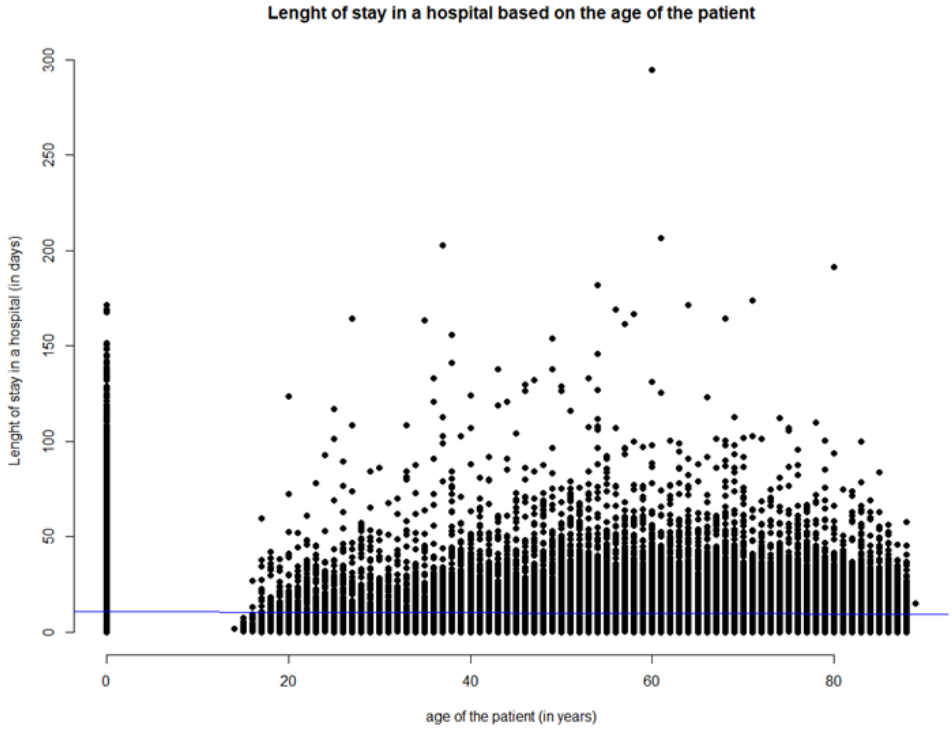
The wide range and substantial standard deviation of “LOSdays” underscores the considerable variability in hospital stay lengths. In other words, a patient’s experiences vary widely —some patients have very short stays, while others have much longer stays.

Furthermore, the mean is higher than the median, suggesting a right-skewed distribution, indicating that there are few patients with exceptionally long hospital stays. We can verify this by looking at the distribution histogram for the length of stay in a hospital shown below.



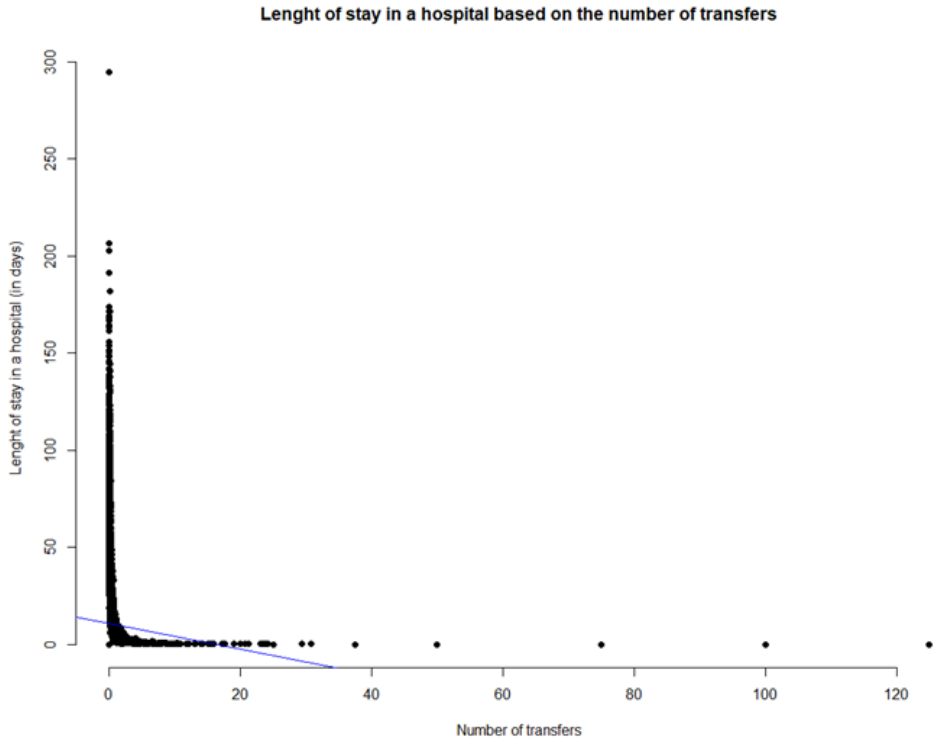
We notice that most frequently the length of stay in a hospital is below 25 days for a considerable number of patients. To be more specific, the most frequent entry for “LOSdays” was 1.96 days with a frequency of 365. We do see however, a long tail on the right, indicating that there are also many patients with very long hospital stays, aligning with the large standard deviation we’ve found earlier.

Until now, we have observed uniquely just the length of days on its own. It is natural to think that some outside influence explains the variability in the length of stay in a hospital. For example, one which would seem obvious would seem to be the influence from the diagnosis of the patient. The more severe it is, the longer the stay. And so, we headed out and analyzed each category of information found in the dataset to check which one would best fit as predictors for the “LOSdays.” Scatter plots were employed to visualize the relationship between each variable and “LOSdays.” For example, let’s verify the relationship between “LOSdays” and “age” with the scatter plot below.



We notice how the regression line (the blue line) is nearly horizontal, but it is slightly trending downwards. What this suggests is that surprisingly, the correlation between age and the length of stay in a hospital is very weak. It does still indicate that younger patients, though not strongly the case, stay in hospital longer.

Now, if we look at the scatter plot relating “LOSdays” with “NumTransfers” (found on the next page), we see how the regression line is significantly steeper compared to our previous regression line. And, since the line is going in a downward angle, this implies that the number of transfers has a notable negative correlation with the length of stay in a hospital.



Through the following process, we determined that "NumCallouts" representing the number of callouts, "NumRx" representing the number of prescriptions, "NumTransfers" representing the number of transfers and "NumDiagnosis" representing the number of diagnoses to be the regressors that are in value of explaining the variability in "LOSdays." Also, all these regressors have a negative correlation with the "LOSdays":

- **NumCallouts:** -0.20637 (The strongest correlation with "LOSdays" amongst the variables in the dataset).
- **NumRx:** -0.169386
- **NumTransfers:** -0.167765
- **NumDiagnosis:** -0.155828

It's crucial to address the "Admit_Diagnosis" variable, which represents the type of diagnosis a patient has received. Logically, one might expect this variable to play a significant role in explaining the variability in "LOSdays." However, our analysis did not reveal a strong correlation, which could be attributed to the heterogeneity of the diagnostic categories. A potential solution could involve reclassifying these categories into broader groups, such as "Neurological Issues," "Infections and Inflammations," etc., to possibly yield more insightful results. Nonetheless, due to our limited expertise in medical terminology and diagnostics, we will not attempt to consolidate these categories in this exploration. Instead, we acknowledge this limitation and proceed with our analysis based on the available data.

In summary, the analysis of the dataset reveals a right-skewed distribution of the length of stay in a hospital, with a mean of 10.11 days and a standard deviation of 12.46 days. The key variables identified as having significant negative correlations with "LOSdays" are "NumCallouts," "NumRx," "NumTransfers," and "NumDiagnosis." This enhanced understanding of the factors influencing hospitalization duration is crucial for optimizing patient care and hospital resource management.

To predict the length of stay in a hospital, let's create a regression model.We just concluded that "NumCallouts," "NumRx," "NumTransfers," and "NumDiagnosis." are the four most important regressors explaining the variability in the length of stay in a hospital. If we put X_1 as "NumCallouts", X_2 as "NumRx," X_3 as "NumTransfers" and X_4 as "NumDiagnosis," we can form the following multiple regression model which helps determine the value of Y, the length of a stay in a hospital (in days):

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \varepsilon$$

If we assume a linear relationship between the predicted value and the regressors (which seems to be the most likely), we can assume that the error value ε is normally distributed, a mean equal to 0 (meaning that on average, our predictions neither systematically overestimate or underestimate the true values) and variance σ^2.

Since we need to conduct a Bayesian linear regression analysis, we need to specify a prior distribution for each of the model parameters. For simplicity, let's assume that all the model parameters are normal priors. Thanks to software like SPSS Statistics Data Editor, we can obtain the following information from a Bayesian regression analysis:

ANOVA[a,b]

Source	Sum of Squares	df	Mean Square	F	Sig.
Regression	740877.162	4	185219.291	1298.829	<.001
Residual	8409547.547	58971	142.605		
Total	9150424.709	58975			

a. Dependent Variable: LOSdays

b. Model: (Intercept), NumCallouts, NumDiagnosis, NumRx, NumTransfers

Bayesian Estimates of Coefficients[a,b,c]

Parameter	Posterior Mode	Posterior Mean	Posterior Variance	95% Credible Interval Lower Bound	95% Credible Interval Upper Bound
(Intercept)	13.224	13.224	.005	13.090	13.357
NumCallouts	-14.294	-14.294	.093	-14.890	-13.697
NumDiagnosis	.055	.055	.000	.034	.076
NumRx	-.117	-.117	.000	-.126	-.108
NumTransfers	-.653	-.653	.001	-.703	-.603

a. Dependent Variable: LOSdays

b. Model: (Intercept), NumCallouts, NumDiagnosis, NumRx, NumTransfers

c. Assume conjugate priors.

Bayesian Estimates of Coefficients[a,b,c]

Parameter	Posterior Mode	Posterior Mean	Posterior Variance	95% Credible Interval Lower Bound	95% Credible Interval Upper Bound
(Intercept)	13.224	13.224	.005	13.090	13.357
NumCallouts	-14.294	-14.294	.093	-14.890	-13.697
NumDiagnosis	.055	.055	.000	.034	.076
NumRx	-.117	-.117	.000	-.126	-.108
NumTransfers	-.653	-.653	.001	-.703	-.603

a. Dependent Variable: LOSdays

b. Model: (Intercept), NumCallouts, NumDiagnosis, NumRx, NumTransfers

c. Assume conjugate priors.

With the information found in the second table, we can create our prediction formula:

$$Y = 13.224 - 14.294 + 0.055X_2 - 0.117X_3 - 0.653X_4$$

From here, we can determine the probability of a patient staying longer than 2 days:

$$P(Y > 2) = \frac{Number\ of\ Predictions\ that\ are > 2}{Number\ of\ Predictions}$$

To determine the number of predictions over 2, we can use a software like R. The script is attached in *Appendix 4*. This code shows us that:

$$P(Y > 2) \approx 0.98$$

To conclude, according to our dataset, there is a 98% chance that a patient- will stay longer than 2 days in a hospital and therefore definitely miss work. The inverse correlation with LOSDays of our key input variables align with our conclusion since, logically, the more attention a patient receives from staff, the faster they may be accurately diagnosed and treated.

We tried multiple methods to verify our answer such as a quick plot of the regular regression line and a simple pythagorean theorem calculation yielded that 84.27% of patients stayed over 2 days which gave us direction that the real answer using bayesian linear regression should be greater than this number as bayesian theory also considers more specific prior and likelihood knowledge of the variables.

We also created various other formulations of Bayesian regression not reliant on “point and click” software like SPSS, as visible in *Appendix 5* in which we formulated priors, to then be multiplied by their likelihoods and fed into a MCMC function. That model considered Age, NumTransfers and Num Callouts as the most relevant variables to LOSDays and yielded that:

$$P(Y > 2) \approx 0.92$$

However as we lacked the time and technical knowledge required to elaborate this model beyond a basic level, its inclusion in this report is more an exposition of our process and various attempts at model creation. Given more time, we would have loved to recreate this model to also consider non continuous variables like *ethnicity, diagnosi*s bins, *admit procedure* bins, *religion* and *marital* status (they say married men and single women are both healthier than unmarried men and married women). As well as to explore adding more variables to the model.

Appendix 1:

```
set.seed(991008)
# control
a = rbeta(100000,17489 + 2,127785 + 20)
# treatment
b = rbeta(100000,17264 + 2,128047 + 20)
total = 0
for (i in 1:100000){
  if (a[i] > b[i]){
    total = total + 1
  }
}
total/100000
```

Appendix 2:

```
#################
###### 100 #######
#################
#generating a random sample for ab_cont
set.seed(15432)
ab_cont1 <- ab_cont[sample(nrow(ab_cont), 100), ]
#remove these data points from ab_cont
ab_cont <- filter(ab_cont,!(ab_cont$user_id %in% ab_cont1$user_id))


#generating a random sample for ab_treat
ab_treat1 <- ab_treat[sample(nrow(ab_treat), 100), ]
#remove these data points from ab_treat
ab_treat <- filter(ab_treat,!(ab_treat$user_id %in% ab_treat1$user_id))

#using bayesTest to analyze the dataset
AB1 <- bayesTest(ab_cont1$converted, ab_treat1$converted,
        priors = c('alpha' = 2, 'beta' = 20),
        distribution = 'bernoulli')

p1 <- plot(AB1)

#Simulated dataset for posterior values (monte carlo)
AB_both <- p1$posteriors$Probability$data
A_only <- filter(AB_both,AB_both$recipe == 'A')
B_only <- filter(AB_both,AB_both$recipe == 'B')

#means for distribution
mean(A_only$value)
mean(B_only$value)

#generating a random sample of 10000 probabilities from A
set.seed(15432)
A_only <- A_only[sample(nrow(A_only), 10000), ]

#generating a random sample of 10000 probabilities from B
set.seed(15642)
B_only <- B_only[sample(nrow(B_only), 10000), ]

#calculating the probability that B is better than A using 10000 samples from both posteriors
prob_B_better <- mean(B_only$value > A_only$value)
prob_B_better
```

```
#################
###### 200 ######
#################

#generating a random sample for ab_cont
ab_cont2 <- ab_cont[sample(nrow(ab_treat), 100), ]

#remove these data points from ab_cont
ab_cont <- filter(ab_cont,!(ab_cont$user_id %in% ab_cont2$user_id))


#generating a random sample for ab_treat
ab_treat2 <- ab_treat[sample(nrow(ab_treat), 100), ]
#remove these data points from ab_treat
ab_treat <- filter(ab_treat,!(ab_treat$user_id %in% ab_treat2$user_id))


#generating counts for success and failure
success_count = 2 + sum(ab_cont1$converted) + sum(ab_treat1$converted)
fail_count = 20 + (100-sum(ab_cont1$converted)) + (100-sum(ab_treat1$converted))

#adding in the new data points
ab_cont_total <- rbind(ab_cont1,ab_cont2)
ab_treat_total <- rbind(ab_treat1,ab_treat2)

#using bayesTest to analyze the dataset
AB2 <- bayesTest(ab_cont_total$converted, ab_treat_total$converted,
        priors = c('alpha' = success_count, 'beta' = fail_count),
        distribution = 'bernoulli')

p2 <- plot(AB2)

#Simulated dataset for posterior values (monte carlo)
AB_both <- p2$posteriors$Probability$data

A_only <- filter(AB_both,AB_both$recipe == 'A')
B_only <- filter(AB_both,AB_both$recipe == 'B')

#means for distribution
mean(A_only$value)
mean(B_only$value)

#generating a random sample of 10000 probabilities from A
```

```
set.seed(15442)
A_only <- A_only[sample(nrow(A_only), 10000), ]

#generating a random sample of 10000 probabilities from B
set.seed(15742)
B_only <- B_only[sample(nrow(B_only), 10000), ]

#calculating the probability that B is better than A using 10000 samples from both posteriors
prob_B_better <- mean(B_only$value > A_only$value)
prob_B_better


##################
###### 300 ######
##################

#generating a random sample for ab_cont
ab_cont3 <- ab_cont[sample(nrow(ab_cont), 300), ]

#remove these data points from ab_cont
ab_cont <- filter(ab_cont,!(ab_cont$user_id %in% ab_cont3$user_id))


#generating a random sample for ab_treat
ab_treat3 <- ab_treat[sample(nrow(ab_treat), 300), ]
#remove these data points from ab_treat
ab_treat <- filter(ab_treat,!(ab_treat$user_id %in% ab_treat3$user_id))


#generating counts for success and failure
success_count = success_count + sum(ab_cont3$converted) + sum(ab_treat3$converted)
fail_count = success_count + (300-sum(ab_cont3$converted)) + (300-sum(ab_treat3$converted))

#adding in the new data points
ab_cont_total <- rbind(ab_cont_total,ab_cont3)
ab_treat_total <- rbind(ab_treat_total,ab_treat3)

#using bayesTest to draw a distribution for the posteriors
AB3 <- bayesTest(ab_cont_total$converted, ab_treat_total$converted,
        priors = c('alpha' = success_count, 'beta' = fail_count),
        distribution = 'bernoulli')

p3 <- plot(AB3)
```

```
#Simulated dataset for posterior values (monte carlo)
AB_both <- p3$posteriors$Probability$data
A_only <- filter(AB_both,AB_both$recipe == 'A')
B_only <- filter(AB_both,AB_both$recipe == 'B')

#means for distribution
mean(A_only$value)
mean(B_only$value)

#generating a random sample of 10000 probabilities from A
set.seed(15632)
A_only <- A_only[sample(nrow(A_only), 10000), ]

#generating a random sample of 10000 probabilities from B
set.seed(15612)
B_only <- B_only[sample(nrow(B_only), 10000), ]

#calculating the probability that B is better than A using 10000 samples from both posteriors
prob_B_better <- mean(B_only$value > A_only$value)
prob_B_better

#################
###### 500 ######
#################

#generating a random sample for ab_cont
ab_cont4 <- ab_cont[sample(nrow(ab_cont), 500), ]

#remove these data points from ab_cont
ab_cont <- filter(ab_cont,!(ab_cont$user_id %in% ab_cont4$user_id))


#generating a random sample for ab_treat
ab_treat4 <- ab_treat[sample(nrow(ab_treat), 500), ]
#remove these data points from ab_treat
ab_treat <- filter(ab_treat,!(ab_treat$user_id %in% ab_treat4$user_id))


#generating counts for success and failure
success_count = success_count + sum(ab_cont4$converted) + sum(ab_treat4$converted)
fail_count = success_count + (500-sum(ab_cont4$converted)) + (500-sum(ab_treat4$converted))

#adding in the new data points
ab_cont_total <- rbind(ab_cont_total,ab_cont4)
```

```
ab_treat_total <- rbind(ab_treat_total,ab_treat4)

#using bayesTest to draw a distribution for the posteriors
AB4 <- bayesTest(ab_cont_total$converted, ab_treat_total$converted,
          priors = c('alpha' = success_count, 'beta' = fail_count),
          distribution = 'bernoulli')

p4 <- plot(AB4)

#Simulated dataset for posterior values (monte carlo)
AB_both <- p4$posteriors$Probability$data
A_only <- filter(AB_both,AB_both$recipe == 'A')
B_only <- filter(AB_both,AB_both$recipe == 'B')

#means for distribution
mean(A_only$value)
mean(B_only$value)

#generating a random sample of 10000 probabilities from A
set.seed(15432)
A_only <- A_only[sample(nrow(A_only), 10000), ]

#generating a random sample of 10000 probabilities from B
set.seed(15642)
B_only <- B_only[sample(nrow(B_only), 10000), ]

#calculating the probability that B is better than A using 10000 samples from both posteriors
prob_B_better <- mean(B_only$value > A_only$value)
prob_B_better

####################################
#generating data frame for ggridges
####################################

#Dataset n=100
df1 <- p1$posteriors$Probability$data
#adding a column of 100s
df1$type <- '100'

#generating grouping column
df1 <- df1 %>%
  mutate(grouping = case_when(recipe == 'A' &type=='100' ~ 'A1',
                      recipe == 'B' &type=='100' ~ 'B1'))
```

```
#dataset n=200
df2 <- p2$posteriors$Probability$data
#adding a column of 200s
df2$type <-'200'

#making a grouping column
df2 <- df2 %>%
  mutate(grouping = case_when(recipe == 'A' &type=='200' ~ 'A2',
                      recipe == 'B' &type=='200' ~ 'B2'))

#dataset n=300
df3 <- p3$posteriors$Probability$data
#adding a column of 500s
df3$type <-'300'

#making a grouping column
df3 <- df3 %>%
  mutate(grouping = case_when(recipe == 'A' &type=='300' ~ 'A3',
                      recipe == 'B' &type=='300' ~ 'B3'))

#dataset n=500
df4 <- p4$posteriors$Probability$data
#adding a column of 1000s
df4$type <-'500'

#making a grouping column
df4 <- df4 %>%
  mutate(grouping = case_when(recipe == 'A' &type=='500' ~ 'A4',
                      recipe == 'B' &type=='500' ~ 'B4'))

#combining them
df_AB <- rbind(df1,df2,df3,df4)

#changing out A and B for control and treatment
df_AB<- df_AB %>%
  mutate(recipe = case_when(recipe == 'A' ~ 'Control',
                      recipe == 'B' ~ 'Treatment'))

#making ggridges graph

## only plot extreme season using dplyr from the tidyverse
ggplot(df_AB, aes(x=value,y=fct_reorder(type,as.numeric(type)),group=grouping,fill=recipe))+
  stat_density_ridges(quantile_lines = TRUE, quantiles = 0.5, scale = 1.7,rel_min_height =
0.01,alpha=0.85)+
```

```
  labs(x = "Rate of Conversion", y = "Sample Size")+
  ggtitle("Posterior Probability Distributions at Varying Sample Sizes")+
  theme_ridges()+
  theme(axis.title.y = element_text(hjust = 0.5))+
  theme(axis.title.y = element_text(vjust = 3))+
  theme(axis.title.x = element_text(hjust = 0.5))+
  theme(legend.title.align = (vjust=5))
```

Appendix 3:

```
#splitting the filtered data set into control and treatment
cont_trials<- filter(ab_data3,ab_data3$group=='control')
treat_trials<- filter(ab_data3,ab_data3$group=='treatment')

#reproducibility
set.seed(1234)

num_trials <- 1
#generating a random sample for cont_trials
cont_set <- cont_trials[sample(nrow(cont_trials),1), ]
#remove these data points from cont_trials
cont_trials <- filter(cont_trials,!(cont_trials$user_id %in% cont_set$user_id))

#generating a random sample for treat_trials
treat_set <- treat_trials[sample(nrow(treat_trials),1), ]
#remove these data points from treat_trials
treat_trials <- filter(treat_trials,!(treat_trials$user_id %in% treat_set$user_id))

bayes_tmp <- bayesTest(cont_set$converted, treat_set$converted,
        priors = c('alpha' = 2, 'beta' = 20),
        distribution = 'bernoulli')

df_trials <- data.frame('Prob_A'=mean(bayes_tmp$posteriors$Probability$A), 'Prob_B' =
mean(bayes_tmp$posteriors$Probability$B),'Prob_Bern_A' = mean(cont_set$converted), 'Prob_Bern_B'
= mean(treat_set$converted), 'Size' = 1)

#specifying variables for algorithm
success_trials = sum(treat_set$converted) + sum(cont_set$converted)
fail_trials = (1-sum(treat_set$converted))+(1-sum(cont_set$converted))
count_trials <- 0

#algorithm for producing means of distribution and MLE of the binomial likelihood
while (count_trials < 100){

num_trials = num_trials + 1

#generating a random sample for cont_trials
cont_tmp <- cont_trials[sample(nrow(cont_trials), 1), ]
cont_set <- rbind(cont_set,cont_tmp)

#remove these data points from cont_trials
cont_trials <- filter(cont_trials,!(cont_trials$user_id %in% cont_set$user_id))
```

```
#generating a random sample for cont_trials
treat_tmp <- treat_trials[sample(nrow(treat_trials), 1), ]
treat_set <- rbind(treat_set,treat_tmp)
#remove these data points from treat_trials
treat_trials <- filter(treat_trials,!(treat_trials$user_id %in% treat_set$user_id))

#using bayesTest to analyze the dataset
bayes_tmp <- bayesTest(cont_set$converted, treat_set$converted,
          priors = c('alpha' = 2+success_trials, 'beta' = 20+fail_trials),
          distribution = 'bernoulli')

df_tmp <- data.frame('Prob_A'=mean(bayes_tmp$posteriors$Probability$A), 'Prob_B' =
mean(bayes_tmp$posteriors$Probability$B),'Prob_Bern_A' = mean(cont_set$converted), 'Prob_Bern_B'
= mean(treat_set$converted), 'Size' = num_trials)

df_trials <- rbind(df_trials, df_tmp[1,])

#means to compare
meanA <- mean(bayes_tmp$posteriors$Probability$A)
meanB <- mean(bayes_tmp$posteriors$Probability$B)
meanC <- mean(cont_set$converted)
meanD <- mean(treat_set$converted)

#changing success and fail for next loop
success_trials = sum(treat_set$converted) + sum(cont_set$converted)
fail_trials = (length(treat_set$converted))-sum(treat_set$converted)+
          (length(cont_set$converted))-sum(cont_set$converted)



if ((abs(meanA-meanC)<0.001)&
  ((abs(meanB-meanD)<0.001))){

  count_trials = count_trials + 1
} else {
  count_trials = 0

}

}

df_trials <- mutate(df_trials,'Control' = Prob_A-Prob_Bern_A, 'Treatment' = Prob_B-Prob_Bern_B)

gg_trials <- head(df_trials,length(df_trials$Prob_A)-100)
```

```
gg1 <- ggplot(gg_trials,aes(x=Size,y=Prob_A))+
  geom_path(aes(y=Control,colour= 'Control'))+
  geom_path(aes(y=Treatment,colour = 'Treatment'))+
  theme_minimal()+
  scale_color_manual(name = NULL, values = c("Control" = "#F8766D", "Treatment" = "#00BFC4"))+
  labs(x='Number of Trials',y='Absolute Difference Between Means')+
  ggtitle('Difference in Mean Probability of Posterior and Binomial MLE')

gg1
```

Appendix 4:

```
> library(readxl)
> data <- read_xlsx("The pathway to our dataset.xlsx")
> install.packages("dplyr")
> library(dplyr)
> beta_0 <- 13.224
> beta_1 <- -14.294
> beta_2 <- 0.055
> beta_3 <- -0.117
> beta_4 <- -0.653
> patients_data <- data.frame(
        NumCallouts = c(data$NumCallouts),
        NumDiagnosis = c(data$NumDiagnosis),
        NumRx = c(data$NumRx),
        NumTransfers = c(data$NumTransfers))
> patients_data <- patients_data %>%
        mutate(predicted_length_of_stay = beta_0 + beta_1 * NumCallouts + beta_2
*NumDiagnosis + beta_3 * NumRx + beta_4 * NumTransfers)
> num_greater_than_2 <- sum(patients_data$predicted_length_of_stay > 2)
> proportion_greater_than_2 <- mean(patients_data$predicted_length_of_stay > 2)
> cat("Number of predictions > 2:", num_greater_than_2, "\n")
> cat("Proportion of predictions > 2:", proportion_greater_than_2, "\n")
```

Appendix 5:

```
library(readxl)
train_data <- read_excel("C:/Users/sheli/Desktop/Fall 2023/MAT 4376/Project 2/train_data.xlsx")
View(train_data

model <- lm(formula= LOSdays ~ age, data=train_data)
summary(model)

A <- train_data$age
B <- train_data$NumTransfers
C <- train_data$NumRx
y <- train_data$LOSdays

likehd <- function(param){
  intercept = param[1]
  slopeA = param[2]
  slopeB = param[3]
  slopeC = param[4]
  sd = param[5]
  pred = slopeA*A + slopeB*B + slopeC*C + intercept
  singlelikelihoods = dnorm(y, mean=pred, sd=sd, log=T)
  sumll = sum(singlelikelihoods)
  return(sumll)}

prior <- function(param){
  intercept = param[1]
  slopeA = param[2]
  slopeB = param[3]
  slopeC = param[4]
  sd = param[5]
  intercept_prior = dunif(intercept, min=0, max=100, log = T)
  slopeA_prior = dnorm(slopeA, mean=0, sd = 2, log = T)
  slopeB_prior = dnorm(slopeB, mean=0, sd = 2, log = T)
  slopeC_prior = dnorm(slopeC, mean=0, sd = 2, log = T)
  sdprior = dunif(sd, min=0, max=100, log = T)
  return(intercept_prior + slopeA_prior + slopeB_prior + slopeC_prior + sdprior)}

posterior <- function(param){return (likehd(param) + prior(param))}
prior(c(1,1,1,1,1,1,1,1,1))

proposalfunction <- function(param){return(rnorm(9,mean = param, sd= c(0.1,0.5,0.3)))}
proposalfunction(c(1,1,1,1,1,1,1,1,1))

run_metropolis_MCMC <- function(startvalue, iterations){
```

```
  chain = array(dim = c(iterations+1,5))
  chain[1,] = startvalue
  for (i in 1:iterations){
    proposal = proposalfunction(chain[i,])
    probab = exp(posterior(proposal) - posterior(chain[i,]))
    if (runif(1) < probab){
      chain[i+1,] = proposal}
    else{      chain[i+1,] = chain[i,]}}
  return(chain)}

#startvalue = c(4,1,10,1,1) # random choice
chain = run_metropolis_MCMC(c(1,1,1,1,1,1,1,1), 10000)

plot(chain)
burnIn = 5000
acceptance = 1-mean(duplicated(chain[-(1:burnIn),]))
par(mfrow = c(2,9)) #row 2 and 3 columns
hist(chain[-(1:burnIn),1],nclass=30, main="Posterior of intercept")
abline(v = mean(chain[-(1:burnIn),1]))
mean(chain[-(1:burnIn),1])
hist(chain[-(1:burnIn),2],nclass=30, main="Posterior of slopeA")
abline(v = mean(chain[-(1:burnIn),2]))
mean(chain[-(1:burnIn),2])
hist(chain[-(1:burnIn),9],nclass=30, main="Posterior of sd")
abline(v = mean(chain[-(1:burnIn),9]) )
```