Homework 02

Ramses Llobet

Problem 1: Working with profile likelihoods

Consider the following dataset: $y = (1, 0, 0, 1, 0, 0, 0, 0)$, which happens to be a series of independent realizations of a Bernoulli distributed random variable. Plot the likelihood function for the Bernoulli parameter π given y. If the experiment were repeated, roughly what fraction of the observations would you expect to be successes ($y = 1$)? Why?

Answer 1a:

In this question, we aim to simulate the likelihood of a Bernoulli distribution and visualize its profile.

For a sequence of n independent Bernoulli trials, each with success probability π, the probability mass function (p.m.f.) of each observation y_i (where $y_i = 1$ for success and $y_i = 0$ for failure) is:

$$f_{\text{Bern}}(y_i \mid \pi) = \pi^{y_i}(1-\pi)^{1-y_i} \quad \text{for } y_i = 0, 1.$$

The likelihood function $\mathcal{L}(\pi \mid \mathbf{y})$ for the entire sample $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ is the product of the individual p.m.f. values:

$$\mathcal{L}(\pi \mid \mathbf{y}) = k(y) \prod_{i=1}^{n} f_{\text{Bern}}(y_i \mid \pi) = k(y) \prod_{i=1}^{n} \left[\pi^{y_i}(1-\pi)^{1-y_i}\right]$$

where $k(y)$ is a **constant** that does not depend on π but affects the scaling of the likelihood function.

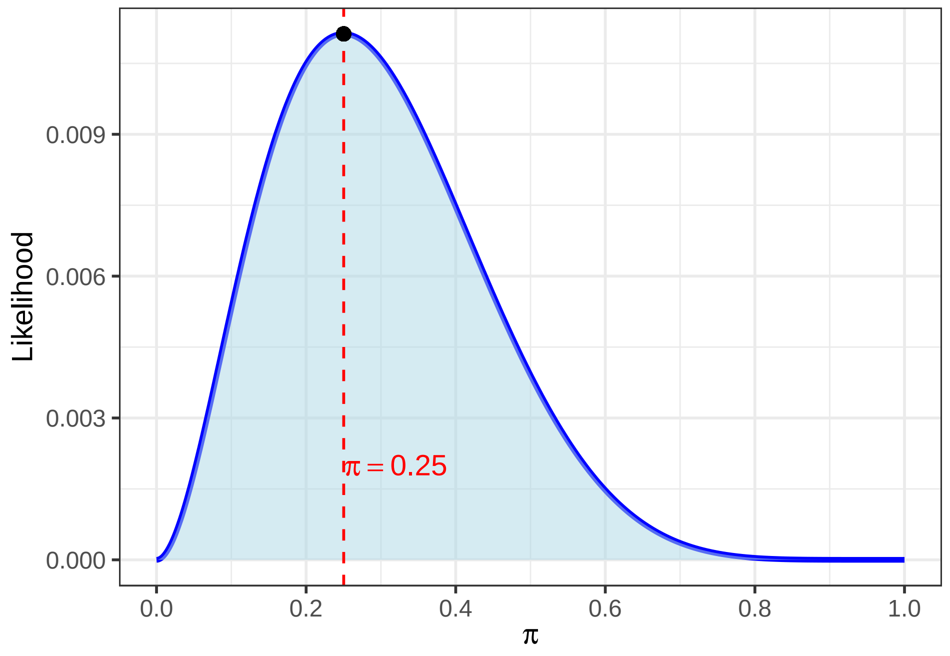
Since $k(y)$ is constant with respect to π, it does not influence the maximization of the likelihood with respect to π. Therefore, we can **drop** $k(y)$, making the likelihood only **proportional** to the remaining expression:

$$\mathcal{L}(\pi \mid \mathbf{y}) \propto \prod_{i=1}^{n} \pi^{y_i}(1-\pi)^{1-y_i}$$

This expression is now ready for maximizing with respect to π to obtain the maximum likelihood estimate (MLE). The code below performs this simulation and plotting.

```r
y = c(1, 0, 0, 1, 0, 0, 0, 0) # data


L_Bernoulli <- function(data, pi) {
  # data: vector of binary data (0 or 1)
  # pi: success probability parameter for each independent Bernoulli trial

  # The Bernoulli likelihood is the join probability (product) of individual Bernoulli probabilities
  res <- prod(pi^data * (1 - pi)^(1 - data))

  return(res)
}

# Now create vectors for loop

sims <- 1000

pi <- seq(0, 1, length.out=sims) # probability = [0,1]

lls <- numeric(sims) # to store the likelihoods


for(i in 1:sims) {
  lls[i] <- L_Bernoulli(data = y, pi[i]) # estimate and store L_Ben for every value of pi
}
```

```r
# Visualize

dt <- data.frame(pi,lls)

dt |>
  ggplot(aes(x = pi, y = lls)) +

  theme_bw() +

  scale_y_continuous(name = "Likelihood") +

  scale_x_continuous(limits = c(0, 1),
                     breaks = seq(0, 1, 0.2),
                     name = expression(pi)) +

  geom_line(aes(x = pi,
                y = lls),
            color = "blue", linewidth = 1) +

  geom_area(fill = "lightblue", alpha = 0.5) +

  geom_vline(linetype = "dashed",
             color = "red",
             xintercept = filter(dt, lls == max(lls))[,"pi"]) +

  geom_point(x = filter(dt, lls == max(lls))[,"pi"], # Selecting pi by maxium ll value
             y = max(dt$lls),
             color = "black", size = 2) +

  annotate(geom="text",
           x = .32,
           y = .002,
           color = "red",
           parse = TRUE,           # To allow mathematical expressions
           label = "pi == .25")
```



Answer 1b:

As an alternative approach, we can derive and visualize the log-likelihood function of the Bernoulli distribution below.

For each observation y_i (where y_i is either 0 or 1), the **Bernoulli probability mass function** is:

$$f_{\text{Bern}}(y_i \mid \pi) = \pi^{y_i}(1-\pi)^{1-y_i} \quad \text{for } y_i = 0, 1.$$

This function describes the probability of observing y_i given a success probability π.

To find the **likelihood** of observing the entire vector $\mathbf{y}' = (y_1, y_2, \ldots, y_n)$, we take the product (*joint probability*) of the Bernoulli PMFs for each individual observation. Assuming a constant probability π across all observations, we have:

$$\Pr(\mathbf{y} \mid \pi) = \prod_{i=1}^{n} \pi^{y_i}(1-\pi)^{1-y_i}.$$

For completeness, the **likelihood axiom** states that the likelihood is defined as the product of the probability mass function (p.m.f.) and a constant $k(y)$:

$$\mathcal{L}(\pi \mid \mathbf{y}) = k(y)\Pr(\mathbf{y} \mid \pi)$$
$$\mathcal{L}(\pi \mid \mathbf{y}) = k(y)\prod_{i=1}^{n} \pi^{y_i}(1-\pi)^{1-y_i}.$$

To simplify the computation, we take the **natural logarithm** of the likelihood function to obtain the **log-likelihood function**.

$$\log \mathcal{L}(\pi \mid \mathbf{y}) = \log k(y) + \sum_{i=1}^{n} \left(y_i \log \pi + (1-y_i)\log(1-\pi) \right).$$

Note that $\log(k(y))$ is constant with respect to π and can be omitted in the computation, as it only scales the likelihood's level without affecting its functional form. However, once we omit the constant, the likelihood becomes proportional to the right.hand side of the equation.

Dropping the constant $\log k(y)$ (since it does not affect the maximization), we get:

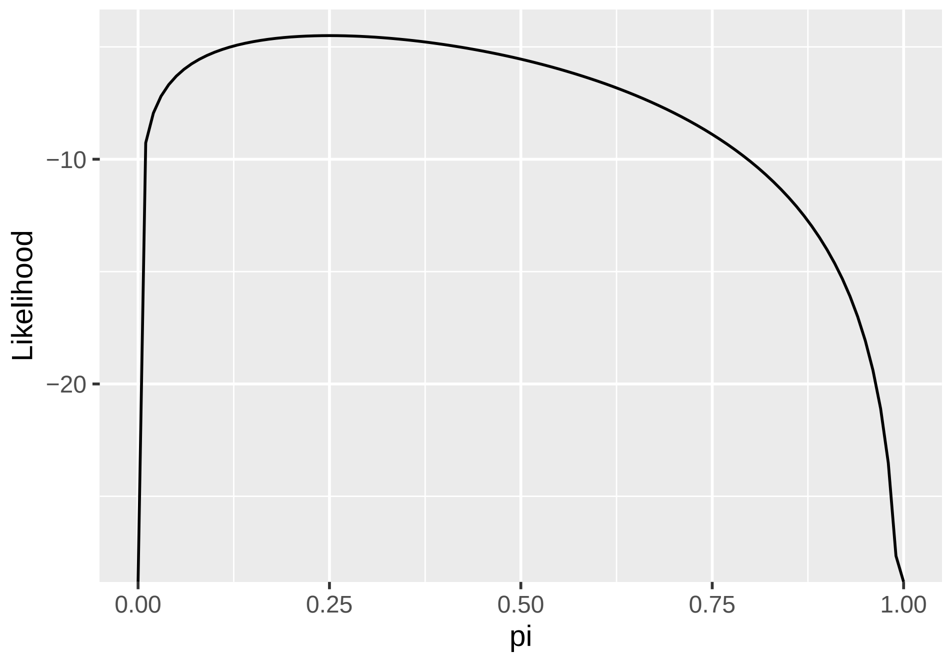
$$\log \mathcal{L}(\pi \mid \mathbf{y}) \propto \sum_{i=1}^{n} \left(y_i \log \pi + (1-y_i)\log(1-\pi)\right).$$

Now we can distribute the summation across terms involving y_i and $1 - y_i$. This allows us to rewrite the log-likelihood as:

$$\log \mathcal{L}(\pi \mid \mathbf{y}) \propto \left(\sum_{i=1}^{n} y_i\right) \log \pi + \left(\sum_{i=1}^{n}(1-y_i)\right)\log(1-\pi).$$

The resulting equation provide the computational routine to plot the **log-likelihood profile** of a Bernoulli distribution given some sample of $\mathbf{y}' = (y_1, y_2, \ldots, y_n)$.

```r
# vector of data on independent Bernoulli realizations
y <- c(1, 0, 0, 1, 0, 0, 0, 0)


# function of profie likelihood
fun1 <- function(pi){
  sum(y)*log(pi) + sum(1-y)*log(1-pi)
}

# visualize the
tibble(pi = 0) |>
  ggplot(mapping = aes(x=pi)) +
  # fun argument to plot profile
  stat_function(fun = fun1) +
  # make sure to provide the domain of pi
  xlim(0,1)+
  labs(x = "pi",
       y = "Likelihood")
```



To identify the maximum point, use `optimize` function.

```r
optimize(fun1, interval = c(0,1), maximum = TRUE)
```

```
## $maximum
## [1] 0.2500143
##
## $objective
## [1] -4.498681
```

If we were to repeat the experiment, assuming no small sample bias, the expected success rate in our observations ($y = 1$) would be of $1/4$ (the sample mean). As the likelihood maximizes the probability of success at 0.25.

Problem 2: Writing and testing a useful new maximum likelihood estimator

Answer 2a.:

$$y_i \sim f_{\text{Poisson}}(\lambda_i) = \frac{\exp(-\lambda_i)\lambda_i^{y_i}}{\mathbf{y}!}$$
$$\lambda_i = \exp(\mathbf{x}_i\beta)$$

$$\mathcal{L}(\lambda \mid y) = k(\mathrm{y})\Pr(y \mid \lambda) = k(\mathrm{y})\prod_{i=1}^{n}\frac{\exp(-\lambda_i)\lambda_i^{y_i}}{\mathrm{y}!}$$
$$\log\mathcal{L}(\lambda \mid y) = \log\prod_{i=1}^{n} k(y_i)\frac{\exp(-\lambda_i)\lambda_i^{y_i}}{\mathrm{y}!}$$
$$= \sum_{i=1}^{n}\log\left(k(y_i)\frac{\exp(-\lambda_i)\lambda_i^{y_i}}{\mathrm{y}!}\right)$$
$$= \sum_{i=1}^{n}\Big(\log k(y_i) + y_i\log\lambda_i - \log\exp(\lambda_i) - \log\mathrm{y}!\Big)$$
$$\propto \sum_{i=1}^{n}\Big(y_i\log\lambda_i - \lambda_i\Big)$$
$$\log\mathcal{L}(\beta \mid y) \propto \sum_{i=1}^{n}\Big(y_i(\mathrm{x}_i\beta) - \exp(\mathrm{x}_i\beta)\Big)$$

Answer 2b.:

$$f_{\mathrm{Poisson,\ t}}(y \mid \lambda, t) = \frac{\exp(-\lambda_i t_i)(\lambda_i t_i)^{y_i}}{\mathrm{y}!}$$
$$\Pr(y \mid \lambda, t) = \prod_{i=1}^{n}\frac{\exp(-\lambda_i t_i)(\lambda_i t_i)^{y_i}}{\mathrm{y}!}$$
$$\log\mathcal{L}(\lambda, t \mid y) = \log\prod_{i=1}^{n} k(y_i)\frac{\exp(-\lambda_i t_i)(\lambda_i t_i)^{y_i}}{\mathrm{y}!}$$
$$\propto \sum_{i=1}^{n}\Big(y_i\log(\lambda_i t_i) - \lambda_i t_i\Big)$$
$$\propto \sum_{i=1}^{n}\Big(y_i\big(\log\lambda_i + \log t_i\big) - \lambda_i t_i\Big)$$
$$\log\mathcal{L}(\beta \mid y) \propto \sum_{i=1}^{n}\Big(y_i(\mathrm{x}_i\beta) - t_i\exp(\mathrm{x}_i\beta)\Big)$$

Answer 2c.:

```r
LHp <- function(param, y, x, t) { # no constant version
                                  # so put constant in the xcovariate
                                  # when running optim

  x <- as.matrix(x)
  t <- as.matrix(t)
  beta <- param
  xb <- x %*% beta
  -sum(y * (xb) - t * exp(xb))
}

LHp2 <- function(param, y, x, t) { # yes constant version
                                   # so don't put constant in the xcovariate
                                   # when running optim
  x <- as.matrix(x)
  os <- rep(1, nrow(x))
  x <- cbind(os,x)
  t <- as.matrix(t)
  beta <- param
  xb <- x %*% beta
  -sum(y * (xb) - t * exp(xb))
}
```

Answer 2d.:

```
obs <- 1000
beta <- c(0, 1, 2)
x0 <- rep(1, obs)
x1 <- runif(obs, 0, 1)
x2 <- runif(obs, 0, 1)
x <- cbind(x0, x1, x2)
t <- sample(c(1,2,3,4,5), size = obs, replace = TRUE)
lambda <- exp(x %*% beta)

y <- rpois(obs, t * lambda)

y %>%
  as_tibble() %>%
  summarise_all(list(mean=mean, sd=sd)) %>%
  pander()
```

mean	sd
16.35	14.56

The mean of y_i is approximately 16.5, and the standard deviation of y_i is approximately 14.7.

Answer 2e.:

```
stval <- c(0, 0, 0)
result <-
  optim(stval,
        LHp2,
        method = "BFGS",
        hessian = TRUE,
        y = y,
        x = x[,-1], # see I just put x which includes constant
        t = t)

pe <- result$par   # point estimate
vc <- solve(result$hessian)  # var-cov matrix
se <- sqrt(diag(vc))    # standard errors
```

```r
ll <- -result$value  # likelihood at maximum


exp(pe)
```

```
## [1] 1.009228 2.657231 7.454056
```

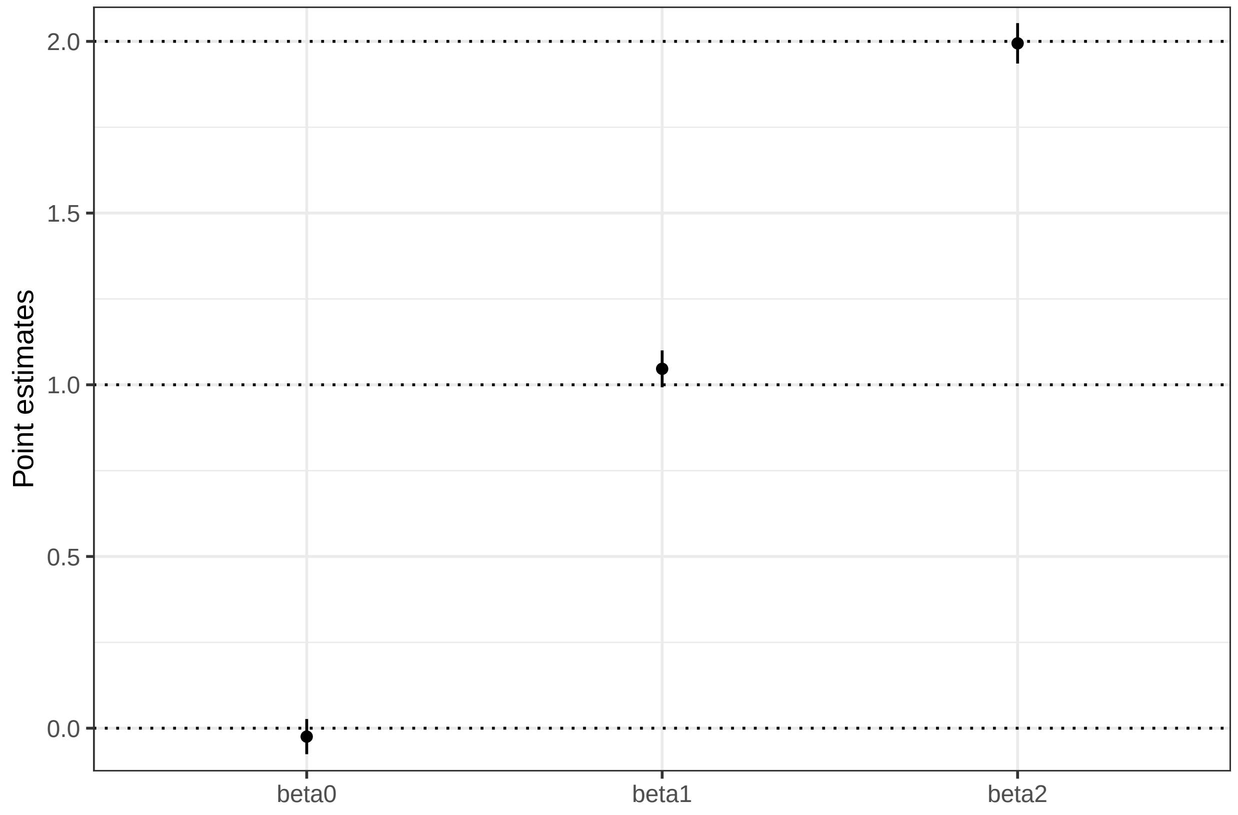
```r
r <- tibble(
  param = c("beta0", "beta1", "beta2"),
  Point_estimates = pe,
  Standard_errors = se)

r %>% pander()
```

param	Point_estimates	Standard_errors
beta0	0.009186	0.0268
beta1	0.9773	0.02742
beta2	2.009	0.03039

```r
pe  <- c(-0.056, 0.979, 2.096)
se <- c(0.026, 0.029, 0.030)


pe <- c(0.252, 0.957, 1.929)
se <- c(0.024, 0.025, 0.027)

pe <- c(-0.02446051, 1.04648278, 1.99426318)
se <- c(0.02611805, 0.02739817, 0.03003965)

dta <-
  tibble(pe = pe,
         se = se,
         trueBeta = c(0,1,2)) %>%
  mutate(lower = pe -1.96*se,
         upper = pe +1.96*se,
         Conf95 = case_when(lower<=trueBeta & upper >= trueBeta ~ 1,
                            TRUE ~0))

dta
```

```
## # A tibble: 3 x 6
##        pe     se trueBeta   lower  upper Conf95
##     <dbl>  <dbl>    <dbl>   <dbl>  <dbl>  <dbl>
## 1 -0.0245 0.0261        0 -0.0757 0.0267      1
## 2  1.05   0.0274        1  0.993  1.10        1
## 3  1.99   0.0300        2  1.94   2.05        1
```

```r
## x2
r %>%
  ggplot(aes(x = param, y = pe))+
  geom_point()+
  geom_linerange(ymin = pe - 1.96*se, ymax = pe + 1.96*se)+
  geom_hline(yintercept = c(0, 1, 2), linetype = "dotted")+
  labs(y = "Point estimates",
       x = "")+
  theme_bw()
```



As the above plot shows, the point estimates are close to true value with small standard error. To improve our confidence, we can increase the number of observations and sample size.

Problem 3: Solving the birthday problem using simulation

A famous probability problem asks "What is the probability that at least two people in a classroom of n people share the same birthday?" Write an R program to solve this problem using simulation. Produce as output a plot of the probabilities of at least one shared birthday for $n = \{2, ..., 50\}$.

Answer 3:

Method 1: Use `duplicated` in `dplyr`

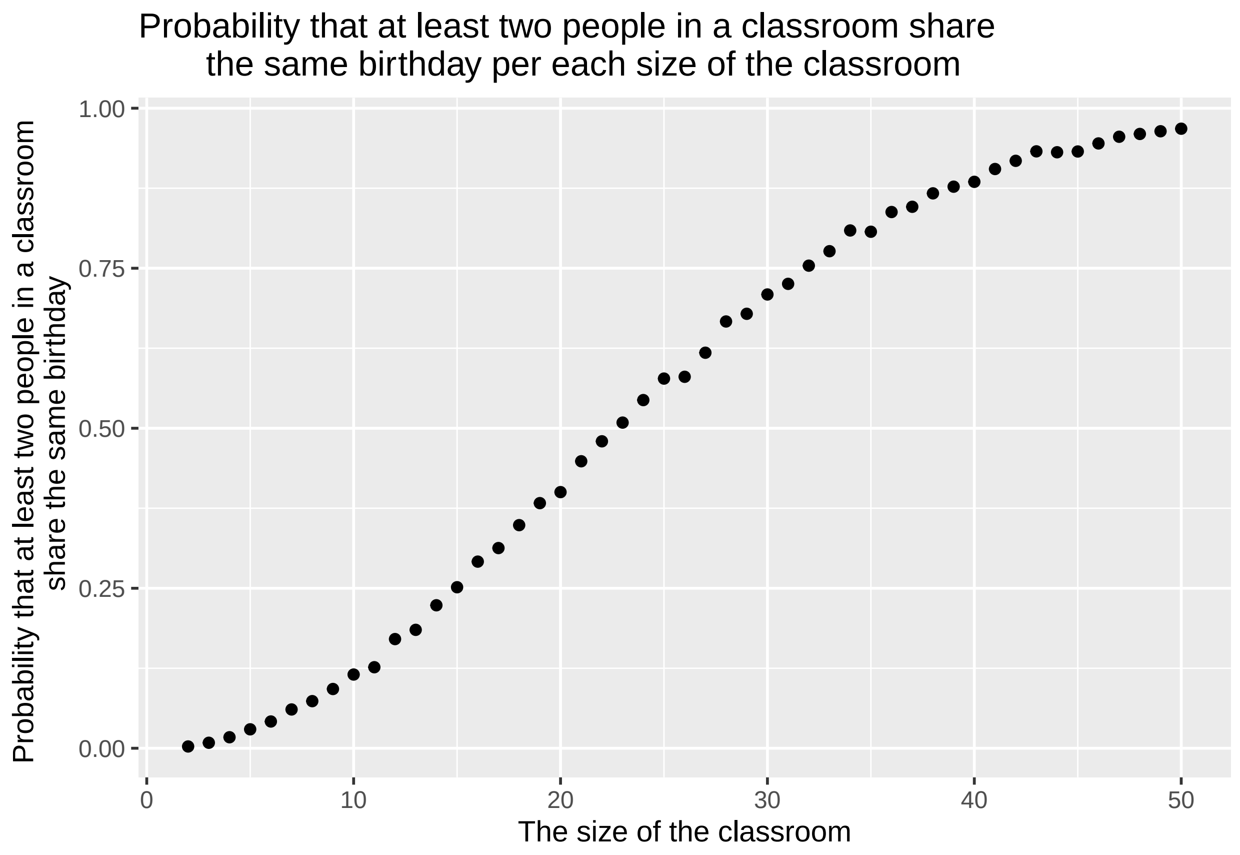
```
BdayFunc <- function(n){
 dd <-  tibble(sims = 1:1000) %>%
    rowwise() %>%
```

```r
    mutate(bdays = list(sample(c(1:365), size = n, replace = TRUE)),
           twopeople = TRUE %in% duplicated(bdays)) %>%
    ungroup() %>%
    summarize(prob = mean(twopeople)) %>% as.numeric()

 dd
}

BdayFunc(50)
```

```
## [1] 0.978
```

Method 2: Use loop

```r
n <- c(2:50) # n is from 2 to 50
prob <- rep(NA, length(n)) # the probability for each n

sim <- 5000 # the number of monte carlo simulations
trial <- rep(NA, sim) # the vector that stores simulated results (always the length of 1000)

for (i in 2:50) { # i stands for the size of the classroom, 2~50

  birth <- rep(NA, i) # birthdays of students per each size, which change in every simulation

  for (k in 1:sim){ # k -> 1 ~ 5000

    for (j in 1:i) { # the length of the birthday should match the size of the classroom
    birth[j] <- sample(1:365, 1, replace=T)
    } # or just try: birth <- sample(1:365, i, replace=T)
      # jth loop is to show you the process with consistency

    trial[k] <- ifelse(length(unique(birth))!=length(birth), 1, 0)
    # 1 if at least two of them have common birthdays
    # 0 if none shares birthdays
  }

prob[i-1] <- sum(trial)/sim # i-1 in order to store the result of the classroom size of 2
                            # as the first element of this vector
}

problem_3 <- data.frame(n=n, prob=prob)

ggplot(problem_3, aes(x=n, y=prob)) +
  geom_point() +
  labs(x="The size of the classroom",
       y="Probability that at least two people in a classroom \n share the same birthday",
       title="Probability that at least two people in a classroom share
       the same birthday per each size of the classroom")
```


Probability that at least two people in a classroom share the same birthday per each size of the classroom
Probability that at least two people in a classroom share the same birthday
1.00
0.75
0.50
0.25
0.00
0
10
20
30
40
50
The size of the classroom