Class 4: Estimation

Practice Exercises*

Diego Perez Ruiz <diego.perezruiz@manchester.ac.uk>

Teacher Assistants: Wei Zhuang <wei.zhuang-2@postgrad.manchester.ac.uk> and Shing Yan Kwong<shingyan.kwong@postgrad.manchester.ac.uk>

1 Simulating samples

Today we will simulate data from different distributions using R. Note that since we are drawing at random from the distributions, the simulated samples will differ for everyone in the class. Therefore, do not expect to get the exact same results as shown below in the examples or compared to someone next to you. Moreover in this class, we look at an example of how to perform maximum likelihood estimation in R.

1.1 Normal distribution

Assume that we know the population of a particular variable X, e.g. the amount of rain that falls measured in 10^7 cubic meters. In fact, we know that this variable follows a normal distribution with mean $\mu = 4.4$ and standard deviation $\sigma = 3.1$.

We can simulate measurements from this population. Let's start with just one value:

```
> rnorm(1,14.4,3.1)
[1] 16.05433
```

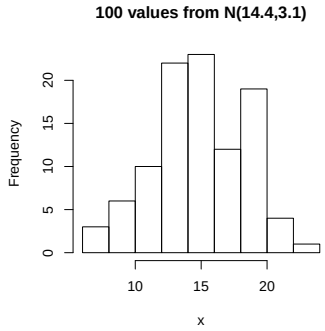
If we take many observations we should see that the values are gathered around the mean 14.4. We simulate 100 values and look at the 20 first ones:

```
x <- rnorm(100,14.4,3.1)
x[1:20]
```

Next we create a histogram of the values, calculate the mean and standard deviation:

```
hist(x, main = "100 values from N(14.4,3.1)")
mean(x)
sd(x)
```

*Please do these exercises before attending the online labs.



The average of our sample is quite close to the mean of the population. Try simulating another sample of 100 and check if the mean and standard deviations changes much.

Let's imagine that we have 1000 researchers who are all conducting the same experiment: take 100 observations from this population. We assume these are independent, so if two researchers get the same values, it is a coincidence. We can simulate this in R with the following algorithm:

1. initialize a vector to store our results
2. generate a random sample of 100 observations
3. calculate the mean of this sample
4. store it in our results vector
5. repeat steps 2-4 1000 times (once for each hypothetical researcher)

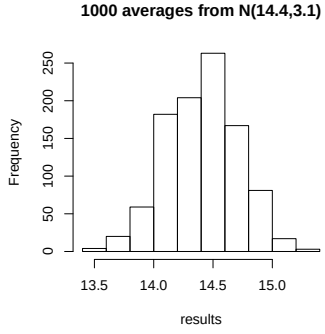
```
results <- c()
  for (i in 1:1000){
  x <- rnorm(100,14.4,3.1)
  mu <- mean(x)
  results <- c(results, mu)
}
```

Look at the first 5 averages. Think of these as the estimates of the population mean for the first 5 researchers:

```
> results[1:5]
[1] 14.21472 14.88761 14.59405 14.33898 14.42986
```

Look at a histogram of all 1000 averages:

```
hist(results, main = "1000 averages from N(14.4,3.1)")
```



We note that the values are not too spread out (we have consistency) and are all in a 'neighbourhood' of the true value: 14.4.

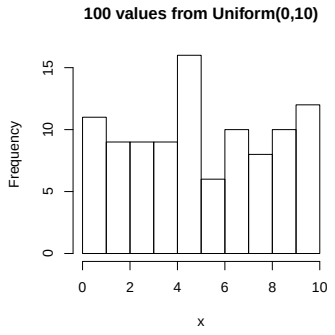
Following can be noted from this exercise:

- The 'central' value – the center of mass – is very close to the true mean of 14.4. We see that, on average, the sample average is equal to the population mean. The expected value of the sample average is equal to the population mean. In other words, the sample average is an unbiased estimator of the population mean.
- Moreover, the spread of sample averages as measured by their standard deviation – this is called the standard error – is smaller than the spread of the population (recall: $\mathrm{Var}(\overline{X}) = \sigma^2/n$). It shows us that in fact the spread decreases as the sample size increases.
- Linear combinations of normally distributed random variables are themselves normally distributed. The sample average consists of a sum of n random variables. Each random variable is taken from the same normal distribution. Therefore their sum is normally distributed and therefore their sum divided by n is normally distributed. This explains why the histogram of the sample averages looks so similar to a normal distribution.

1.2 Uniform distribution

To visualize the central limit theorem we consider the following exercise. This time we simulate 100 samples from a uniform distribution with parameters $a = 0$ and $b = 10$. The mean of this population is 5 (verify by hand). Here's the R command to do it, the first 10 observations, the average and standard deviation of the sample, and a histogram of the sample values:

```
x <- runif(100,0,10)
x[1:10]
mean(x)
sd(x)
hist(x, main = "100 from Uniform(0,10)")
```

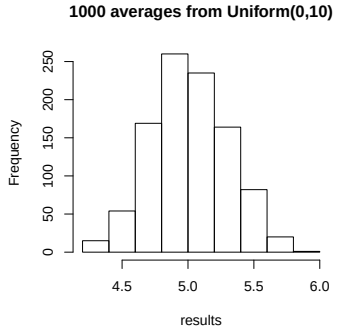


We note that the histogram of this sample will look like a uniform distribution (a rectangle).

Now we imagine our hypothetical 1000 researchers. Each of them is going to take a random sample of 100 observations from the uniform population and calculate the average of their sample. We perform the same commands as before, but this time with a condensed version of the code:

```
results <- c()
for (i in 1:1000) {
  results <- c(results, mean(runif(100,0,10)))
}
mean(results)
```

```
sd(results)
```



Here, the shape of the distribution of averages is a clear example of the central limit theorem at play. Here you can see that it is approximately normal.

2 Maximum likelihood

Assume we have 30 students, each tossing 10 coins each. The success is here defined as getting heads with rate 50%. This data can be simulated from a binomial distribution with $p = 0.5$ and $n = 10$:

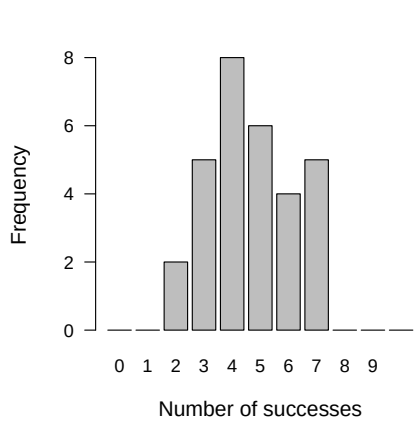
```
samplesize <- 30
p <- 0.5
n <- 10
data <- rbinom(samplesize, prob = 0.5, size = n)
table(data)
```

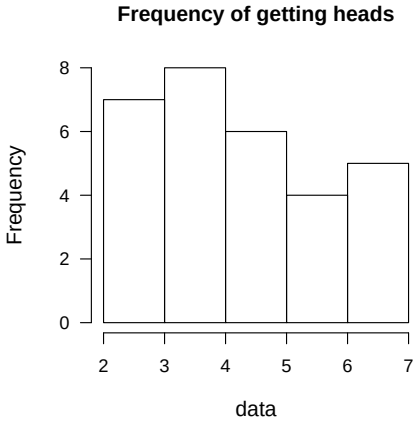
Visualised in a bar plot:

```
data_fac <- factor(data, levels = 0:n)
par(las = 1, cex.lab = 1.2)
barplot(table(data_fac), xlab = "Number of successes", ylab = "Frequency")
```

Visualize in a histogram:

```
hist(data, main = "Frequency of getting heads")
```

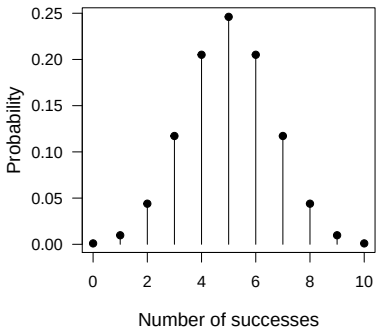




Now we are going to use the function `dbin()` instead. This function provides the probability for x

successes given the parameters p and n. Make sure the probabilities sum up to 1 and then visualize the distribution.

```
binom1 <- dbinom(x = 0:10, prob = p, size = n)
sum(binom1)
plot(binom1, type = "h", col = "black",
     xlab = "Number of successes", ylab = "Probability")
points(binom1, col = "black", pch = 19)
```



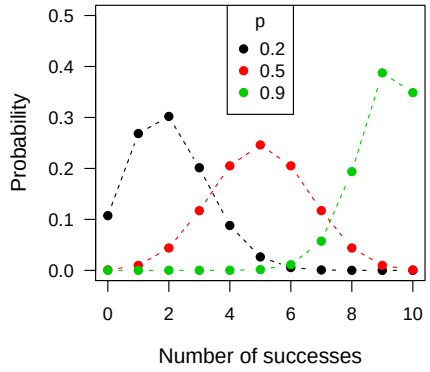
Note the shape of the distribution, why do you think it looks like this?

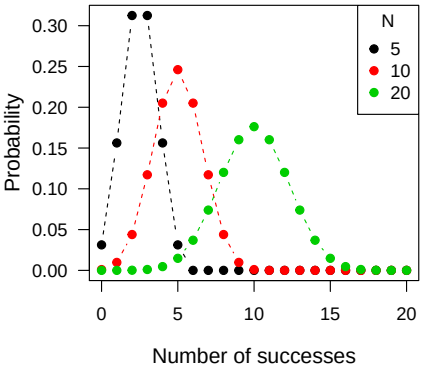
Next we evaluate how the function looks for varying p and create a vector for these varying parameters.

```
p_vec <- c(0.2, 0.5, 0.9)
x <- 0:n
par(las = 1, cex.lab = 1.2)
plot(x, y = dbinom(x, prob = p_vec[1], size = n), type = "n",
xlab = "Number of successes", ylab = "Probability", ylim = c(0, 0.5))
for (i in 1:length(p_vec)){
  points(x, y = dbinom(x, prob = p_vec[i], size = n),
  col = i, type = "b", pch = 19, lty = 2)
}
legend("top", legend = p_vec, col = 1:3, pch = 19, title = "p")
```

We also check what it looks like for varying n.

```
n_vec <- c(5, 10, 20)
p <- 0.5
x <- 0:20
par(las = 1, cex.lab = 1.2)
plot(x, y = dbinom(x, prob = p, size = n_vec[1]), type = "n",
xlab = "Number of successes", ylab = "Probability")
for (i in 1:length(n_vec)){
  points(x, y = dbinom(x, prob = p, size = n_vec[i]),
  col = i, type = "b", pch = 19, lty = 2)
}
legend("topright", legend = n_vec, col = 1:3, pch = 19, title = "N")
```





Let's now start calculating the likelihood and assume each data point is independent of another. Note that in this case we know the true value of p, because we simulated the data. However, when we have real data we usually do not know the true value. The parameter n is usually defined by the the sampling/experiment.

```
dbinom(x = data, prob = 0.5, size = 10)
```

This gives for each data point and we need for the whole data set. The overall likelihood and log likelihood of the data set is obtained by multiplying the single values. The use of the `log = T` argument performs the log transformation.

```
prod(dbinom(x = data, prob = 0.5, size = 10))
sum(dbinom(x = data, prob = 0.5, size = 10, log = T))
```

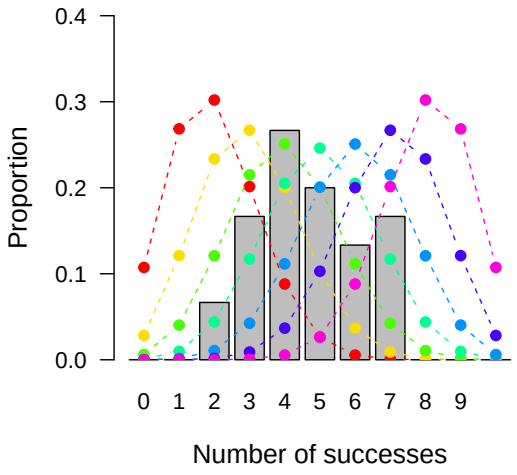
This gives the likelihood for a specific parameter value and we need to calculate the likelihood for any parameter values. Thus we define a function for the log-likelihood with the parameters and the data as arguments.

```
log_likelihood <- function(par_p, par_n, data){
  LL <- sum(dbinom(x = data, prob = par_p, size = par_n, log = T))
  return(LL)
}
log_likelihood(par_p = 0.5, par_n = 10, data = data)
```

As a last step we wish find maximum likelihood estimate of p. This means we would like to find the model, or parameter set respectively, that most likely generated the data. Let's start by visualizing the data with models of different parameter values. We set these varying parameter values to $0.2, 0.3, \ldots, 0.8$.For this we convert frequencies to proportions which standardizes the data and the theoretical probabilities to the same scale.

```
par(las = 1, cex.lab = 1.2)
prop_dat <- table(data_fac)/samplesize
xbars <- barplot(prop_dat, xlab = "Number of successes",
ylab = "Proportion", ylim = c(0, 0.4))
p_vec <- seq(0.2, 0.8, by = 0.1)
col_vec <- rainbow(length(p_vec))
```

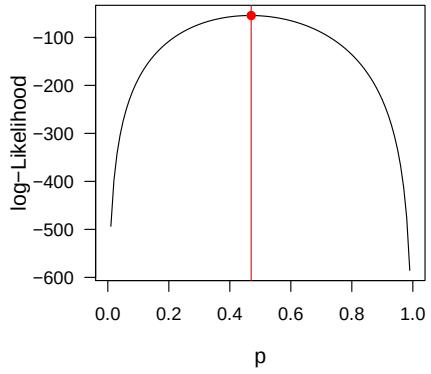
```
n <- 10
x <- 0:10
for (i in 1:length(p_vec)){
  points(xbars, dbinom(x, prob = p_vec[i], size = n),
  type = "b", col = col_vec[i], lty = 2, pch = 19)
  }
```



This figure gives a visual impression which parameter values might be useful, that is which parameter values most likely generated the data.

To derive the maximum likelihood estimate of p, we evaluate the likelihood function for a sequence of parameter values using a fine step size.

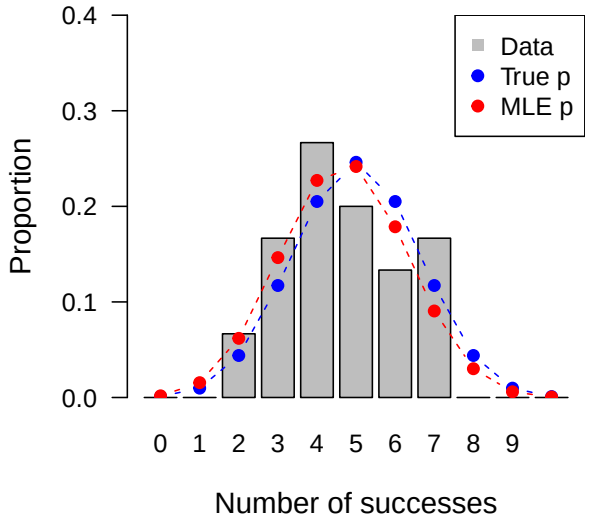
```
p_vec <- seq(0, 1, by = 0.01)
logLik <- sapply(p_vec, FUN = log_likelihood, par_n = 10, data = data)

par(las = 1, cex.lab = 1.2)
plot(p_vec, logLik, type = "l", xlab = "p", ylab = "log-Likelihood")

imax <- which.max(logLik)
p_MLE <- p_vec[imax]
points(p_MLE, max(logLik), pch = 19, col = "red")
abline(v = p_MLE, col = "red")
```



"p_MLE" gives you the numeric values of the mle. This estimate is close to, but differs from the true value used for the data generation. But the larger the sample size, the lower this difference will be on average.

As a summary, we plot the data with the theoretical underlying distribution as well as with the distribution based on the MLE estimate.

```
par(las = 1, cex.lab = 1.2)
xbars <- barplot(prop_dat, xlab = "Number of successes",
          ylab = "Proportion", ylim = c(0, 0.4))
          points(xbars, dbinom(x, prob = 0.5, size = 10),
          type = "b", col = "blue", pch = 19, lty = 2)
points(xbars, dbinom(x, prob = p_MLE, size = 10),
        type = "b", col = "red", pch = 19, lty = 2)
legend("topright", legend = c("Data","True p","MLE p"),
col = c("gray", "blue","red"), pch = c(15, 19, 19))
```



The analytical approach to maximum likelihood estimation is by

1. writing down the joint probability function which is the product of the marginal probabilities
2. take the logarithm and then the first derivative with respect to parameter
3. set equal to zero and solve

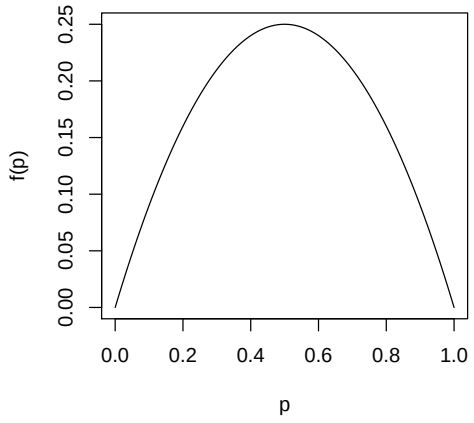
Here we are going to use R and the `optimize` function to do the same. How does the `optimize` function work?

Consider the function $f(p) = p(1-p)$ for $0 \leq p \leq 1$ (Bernoulli distribution). If you plot this distribution over the specified interval for p you should notice a maximum at $p = 1/2$:

```
p <- seq(0,1,0.001)
f <- p*(1-p)
plot(p,f,type="l",xlab="p",ylab="f(p)")
```

You can prove this maximum by solving

$$\frac{df(p)}{dp} = \frac{d(p(1-p))}{dp} = 1 - 2p = 0$$



With the `optimize` function we get:

```
fp <- function(p) {p*(1-p)}
optimize(fp,c(0,1),maximum=T)
$maximum
[1] 0.5
```

Now let's use the function `optimize` to find mle's numerically. The log likelihood function for a Poisson distribution is given by

$$\ell(\lambda) = \ln \lambda \sum_{i=1}^{n} x_i - n\lambda - \sum_{i=1}^{n} \ln(x_i)! \tag{1}$$

so that the mle for λ is given by

$$\hat{\lambda}_{mle} = \overline{X} = \frac{\sum_{i=1}^{n} x_i}{n}$$

We simulate 100 draws from a Poisson distribution with $\lambda = 10$ and calculate $\overline{X}$.

```
x <- rpois(100,10)
mean(x)
```

Next we write down the likelihood function given in Equation (1):

```
poiss <- function(lambda,data){sum(data)*log(lambda)-length(data)*lambda}
```

Note that the last term does not include the parameter λ and thus can be ignored. Now we use `optimize` to numerically to find $\hat{\lambda}$:

```
result <- optimize(poiss, c(0,10^6), maximum=T, data=x)
result$maximum
```

The value from calculated $\overline{X}$ and the optimization result should be very close to each other.