Chapter 11: Interactive Notebook for Students

Ram Gopal, Dan Philps, and Tillman Weyde

Summer 2022

Contents

Load packages

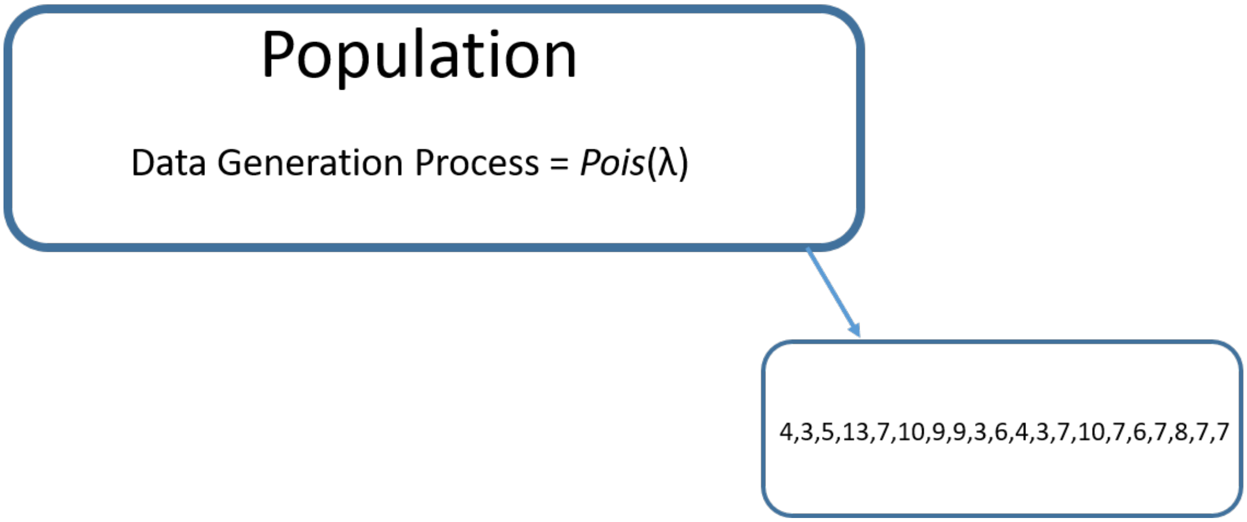
```
library(bbmle)
```

Introduction

Supose we have a sample of data as follows.

```
x = c(4,3,5,13,7,10,9,9,3,6,4,3,7,10,7,6,7,8,7,7)
```

We believe that this sample comes from a population with the data generation process that can be described by a Poisson distribution. This is depicted in the figure below.



In **statistical estimation**, we want to ask the question: what is the estimate of λ that best explains the data that we observe? The most popular approach for estimation is called **maximum likelihood estimation**.

Likelihood Principle

Our objective is to find the value of λ that best explains the data we have. Let us start this quest with an initial value of 6 for λ.

Notice that the first observation in our sample has a value of 4. If λ is 6, what is the probability of finding a value of 4?

```
x = c(4,3,5,13,7,10,9,9,3,6,4,3,7,10,7,6,7,8,7,7)
lam = 6
dpois(x = 4, lambda = lam)
```

```
## [1] 0.1339
```

The second observation has a value of 3. What is the probability of observing this?

```
dpois(x = 3, lambda = lam)
```

```
## [1] 0.08924
```

Similarly, we can compute the probabilities of each of the values in the sample.

```
lam = 6
p = dpois(x = x, lambda = lam)
print(round(p,3))
```

```
##  [1] 0.134 0.089 0.161 0.005 0.138 0.041 0.069 0.069 0.089 0.161 0.134 0.089
## [13] 0.138 0.041 0.138 0.161 0.138 0.103 0.138 0.138
```

Having computed the probabilities of individual observations, let us know compute the probability of observing the entire sample of data.

Let us introduce some notation first. Let $x_1, ..., x_n$ denote the sample observations and $P(x_1|\lambda), ..., P(x_n|\lambda)$ denote the corresponding probabilities for a given λ. If the observations are **independent**, then the probability of observing the entire sample is

$$\prod_{i=1}^{n} P(x_i|\lambda)$$

This is termed the likelihood function.

$$L = \prod_{i=1}^{n} P(x_i|\lambda)$$

The likelihood function as defined above invokes the **iid** assumption, which stands for independent and identically distributed observations.

Let us calculate the likelihood value for our sample with $\lambda = 6$.

```
lam = 6
p = dpois(x = x, lambda = lam)
L = prod(p)
L
```

```
## [1] 0.000000000000000000001559
```

You will notice that likelihood value is very small. Such low values are difficult to work with from a computational standpoint. It is more convenient to use the log of likelihood.

$$LL = log(\prod_{i=1}^{n} P(x_i|\lambda))$$

$$LL = log(P(x_i|\lambda), ...P(x_n|\lambda))$$

$$LL = \sum_{i=1}^{n} log(P(x_i|\lambda))$$

Let us compute the loglikelihood for our sample.

```
lam = 6
p = dpois(x = x, lambda = lam)
LL = sum(log(p))
LL
```

```
## [1] -47.91
```

Let us package this into a function.

```r
LLpois = function(lam){
  p = dpois(x = x, lambda = lam)
  LL = sum(log(p))
  return(LL)
}
LLpois(5)
```

```
## [1] -52.52
```

The **maximum likelihood principle** states that the best estimate of λ is one that maximizes the likelihood of the observed sample. This makes sense because L in essence is the joint probability of observing the entire sample of data. The best estimate of λ is the one that maximizes L. Note that maximizing L is equivalent to maximizing the loglikelihood LL.

In this simple example, it can be shown mathematically that the maximum likelihood estimate of λ is simply the sample average.

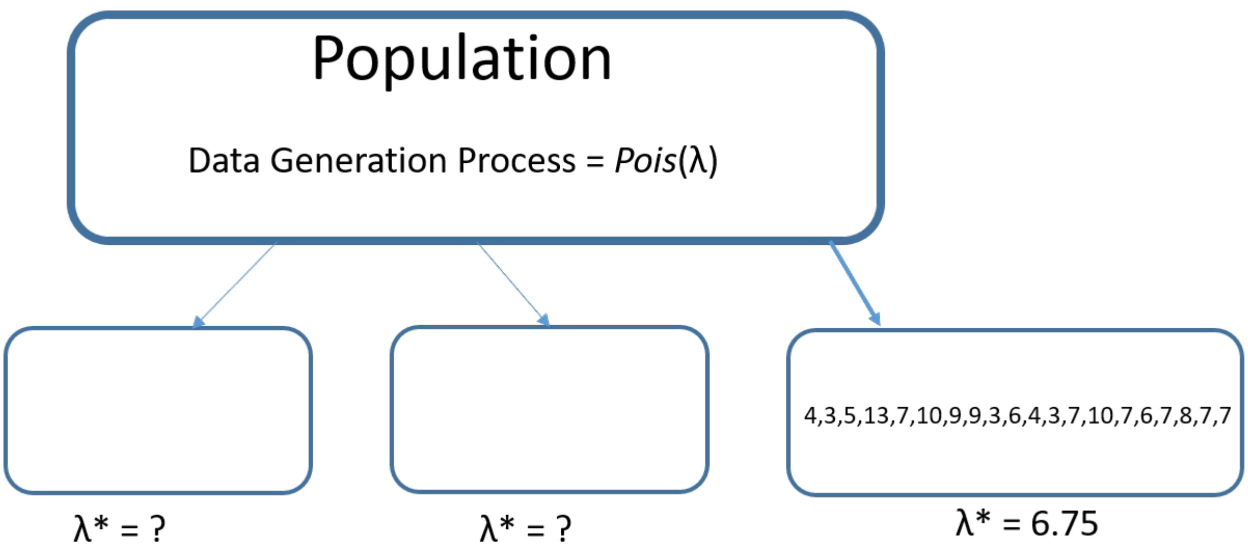
```r
mean(x)
```

```
## [1] 6.75
```

In more complicated settings, we may not be able to solve the optimization problem analytically. In such cases, we use numerical computational methods to get the maximum likelihood estimates.

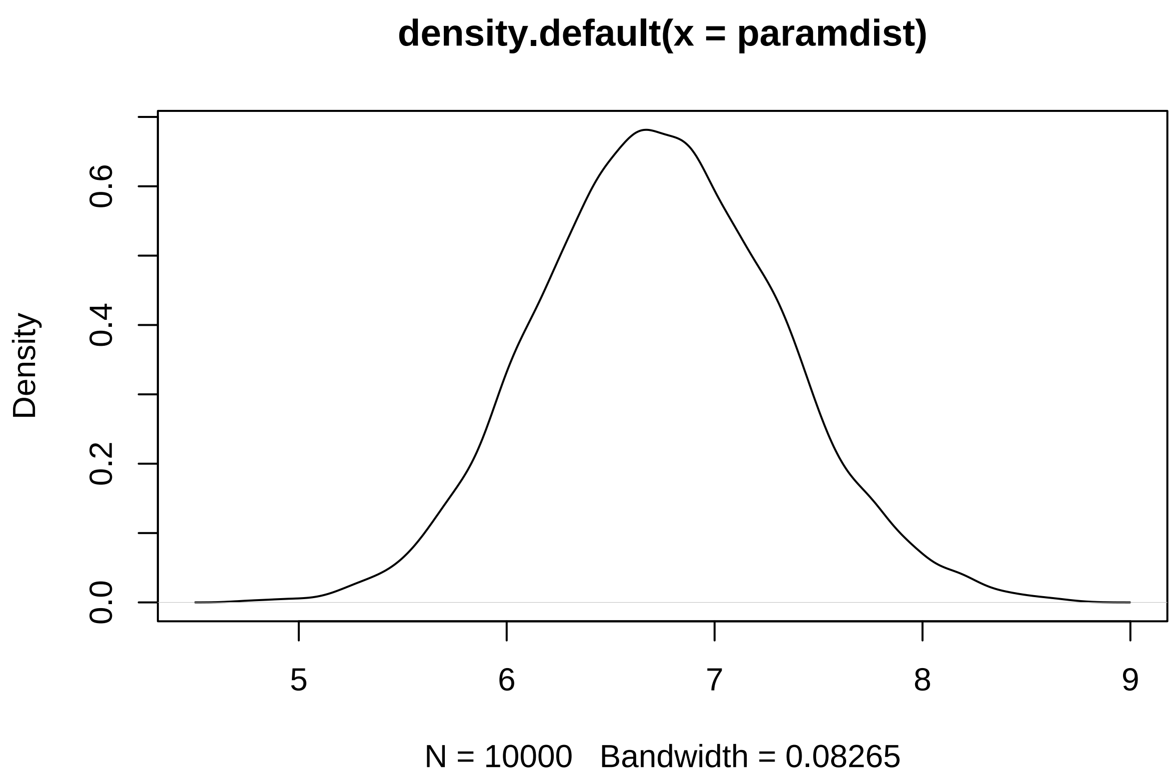
Yet Another Distribution!

Based on our sample of data, the maximum likelihood estimate for λ is 6.75.



What if you take another sample from the same population? A different sample drawn from the population could potentially yield a different estimate for λ. Let us create other samples using the bootstrapping approach and compute the estimate for each.

```r
set.seed(987654321)
paramdist = replicate(10000, mean(sample(x = x, replace = T)))
plot(density(paramdist))
```

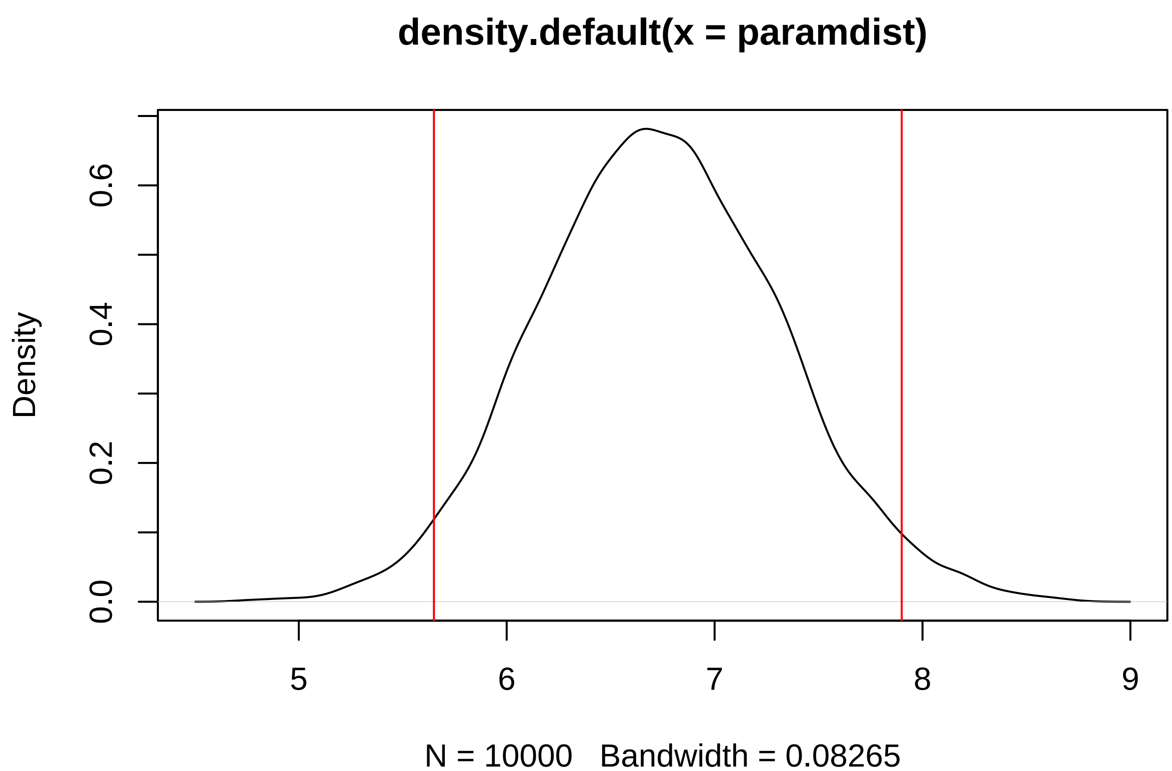


What we have is called a **parameter distribution**. Let us compute the 95% confidence interval and the standard deviation for this distribution.

```r
sd(paramdist)
```

```
## [1] 0.5794
```

```r
q2 = quantile(paramdist, c(.05/2,1-(.05/2)))
plot(density(paramdist))
abline(v = q2, col = "red")
```




```r
paste("95% Confidence interval = ", "[",q2[1], q2[2],"]")
```

```
## [1] "95% Confidence interval =  [ 5.65 7.9 ]"
```

When you take a different sample, you will obviously get a different estimate. Based on the parameter distribution, we can say the following.

1. The maximum likelihood estimate is 6.75. This is also called the **point estimate**.
2. We are 95% confident that the population value of λ is between 5.6 and 7.9.
3. The standard deviation for the parameter distribution is 0.582. This is also called the **standard error** of the parameter estimate.

One final point to be made here is that in most cases, the parameter distribution is reasonably normal.

Optimizing Computationally

We will use the `bbmle` package to computationally optimize the likelihood function.

The main function in the `bbmle` package that does the optimization is `mle2()`. The process is as follows.

1. Create a function that computes the log likelihood value for a given set of parameters. The parameters are the inputs to the function. The output of the function should be the negative of the loglikelihood as most optimization routines attempt to minimize a function.

2. Use the `mle2()` function to attempt to find the maximum likelihood parameter values. The basic syntax is

```
mle2(minuslogl=function to minimize, start=list("starting values of the parameters"))
```

Let us estimate the parameter for the example we just did.

```
LLpois = function(lam){
  p = dpois(x = x, lambda = lam)
  LL = sum(log(p))
  return(-1*LL)
}

res1 = mle2(minuslogl = LLpois, start = list(lam = 10))
summary(res1)
```

```
## Maximum likelihood estimation
##
## Call:
## mle2(minuslogl = LLpois, start = list(lam = 10))
##
## Coefficients:
##     Estimate Std. Error z value                Pr(z)
## lam    6.750      0.581    11.6 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -2 log L: 94.02
```

You will notice that the results are the same as before (when we mathematically optimized likelihood).

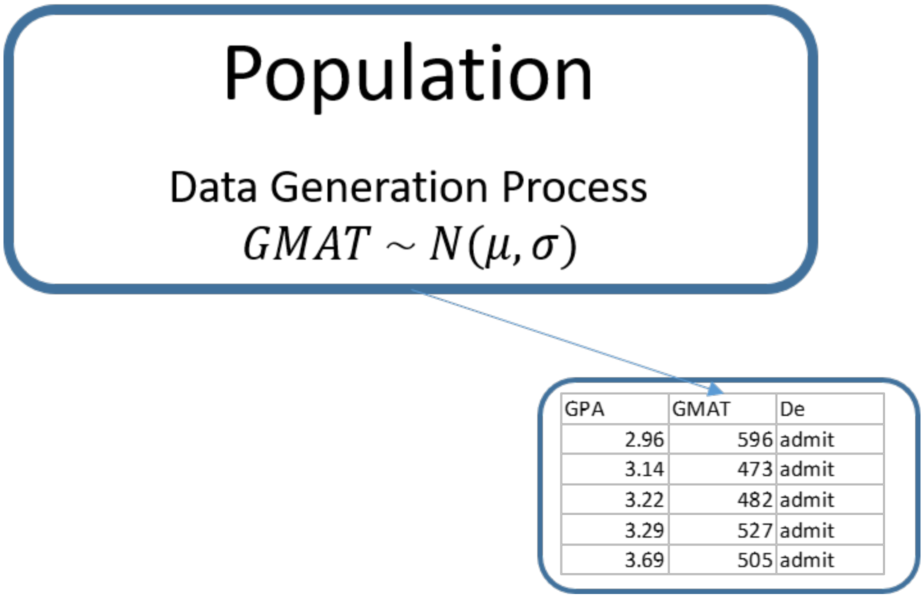
Numeric Outcome

Read the `admission.csv` file.

```
admission <- read.csv("../../data/admission.csv")
head(admission)
```

```
##    GPA GMAT    De
## 1 2.96  596 admit
## 2 3.14  473 admit
## 3 3.22  482 admit
## 4 3.29  527 admit
## 5 3.69  505 admit
## 6 3.46  693 admit
```

Let us assume that the data generation process for the variable `GMAT` is normal. This is depicted in the figure. We want to obtain the maximum likelihood estimates of the population mean and standard deviation.



GPA	GMAT	De
2.96	596	admit
3.14	473	admit
3.22	482	admit
3.29	527	admit
3.69	505	admit

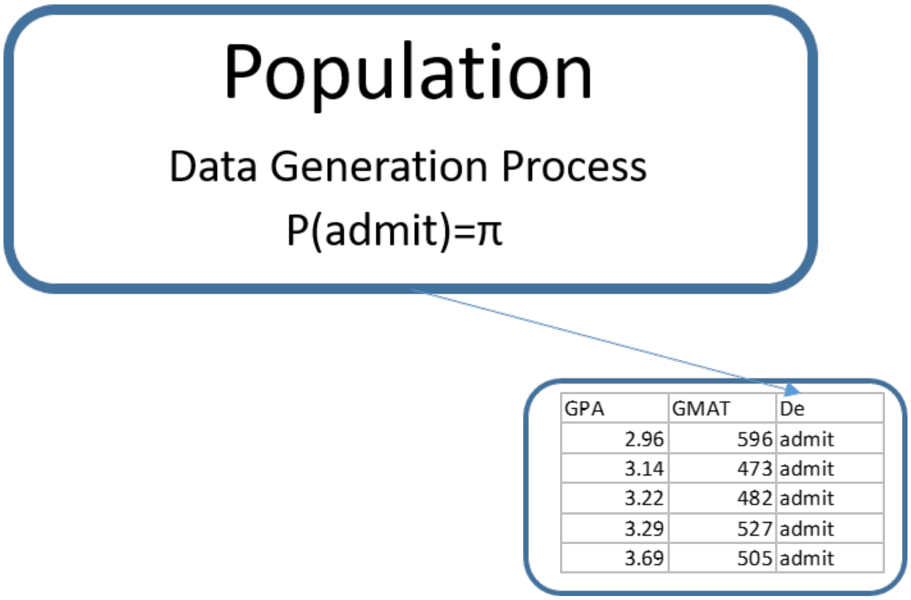
```r
LLnorm = function(mean1, standdev){
  p = dnorm(x = admission$GMAT, mean = mean1, sd = standdev)
  LL = sum(log(p))
  return(-1*LL)
}

res1 = mle2(minuslogl = LLnorm, start = list(mean1 = 500, standdev = 100))
summary(res1)
```

```
## Maximum likelihood estimation
##
## Call:
## mle2(minuslogl = LLnorm, start = list(mean1 = 500, standdev = 100))
##
## Coefficients:
##          Estimate Std. Error z value                Pr(z)
## mean1      488.45       8.79    55.6 <0.0000000000000002 ***
## standdev    81.04       6.22    13.0 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -2 log L: 988.4
```

Binary Outcome

We want to estimate the probability of `admit` based on the sample of data, as shown in the figure below.



GPA	GMAT	De
2.96	596	admit
3.14	473	admit
3.22	482	admit
3.29	527	admit
3.69	505	admit

```r
LLbinary = function(pi){
  p = ifelse(admission$De == "admit", pi, 1 - pi)
  LL = sum(log(p))
  return(-1*LL)
}

res1 = mle2(minuslogl = LLbinary, start = list(pi= .5))
summary(res1)
```

```
## Maximum likelihood estimation
##
## Call:
## mle2(minuslogl = LLbinary, start = list(pi = 0.5))
##
## Coefficients:
##    Estimate Std. Error z value            Pr(z)
## pi   0.3647     0.0522    6.99 0.0000000000028 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -2 log L: 111.5
```