Maxium Likelihood Estimation - How it works

Tonnar Castellano

February 04, 2021

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
```

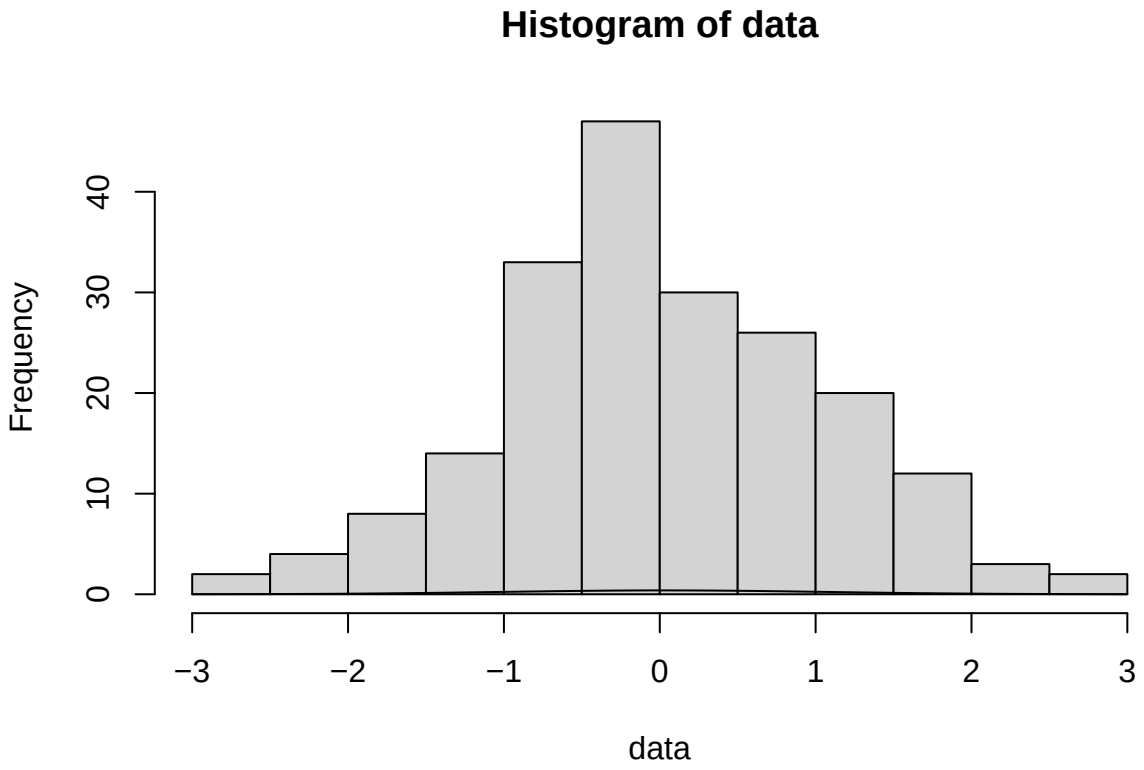
```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Median

```
N <- 201

data <- rnorm(N)

median(data)
```

```
## [1] -0.05922348
```

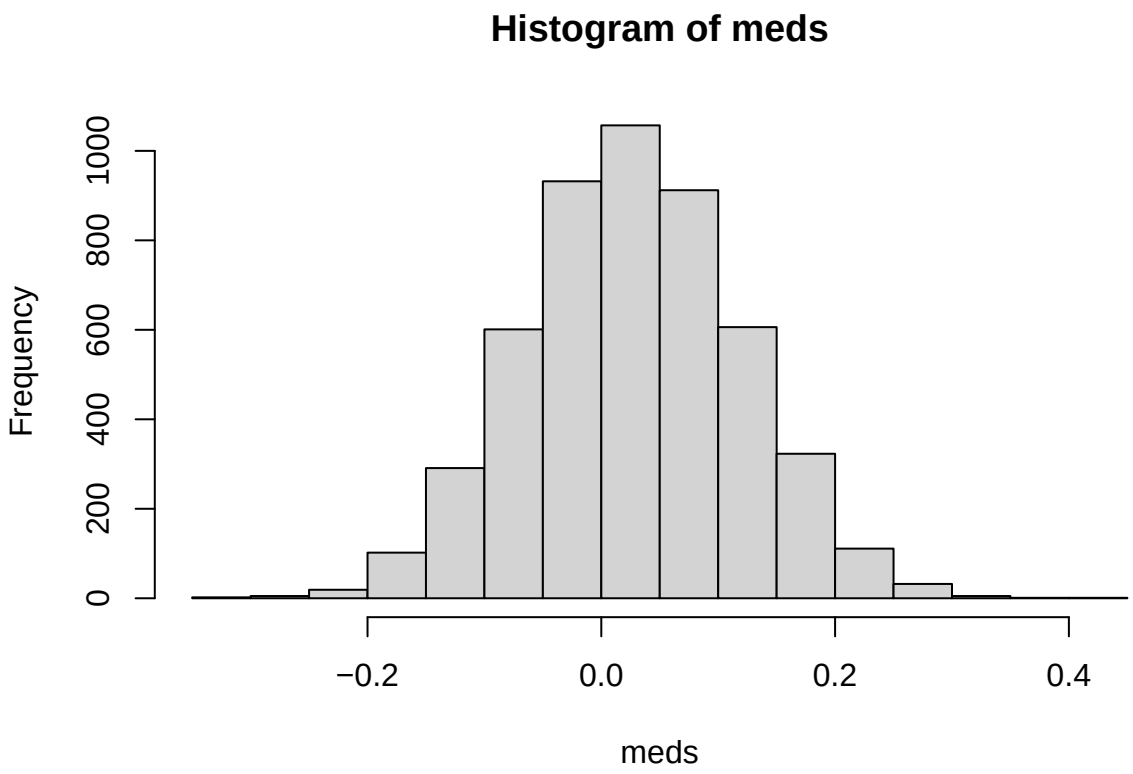
```
mle.mean <- mean(data)
mle.sd <- sqrt(((length(data)-1)/length(data))*var(data))

hist(data)
curve(dnorm(x,mle.mean,mle.sd),add = TRUE)
```



MLE is used to estimate the distribution by finding the maximum likihood function as we change the parameters. These parameters then allow us to understand features of the distribution which can be used for techinques like hypothesis testing or mean estimation. This can be done iteritavely or with closed form solutions. In the example above we use a closed form solution. However, sometimes that is not available. In certain cases we do not always have information about the sampling distribution. The MLE allows us to a potential solution to this problem. This allows us to understand more about the sample of our data.

Approximating the sampling distribution of the median, conditional on the estimate of the distribution in the previous steps.

```
n <- 1000
meds <- NA

for(i in 1:n){
    re.samp <- rnorm(N,mle.mean,mle.sd)
    meds[i] = median(re.samp)
}

meds = rbeta(5000,101,101) %>% qnorm(mean = mle.mean,sd = mle.sd)

hist(meds)
```



So there are a few different ways to find the median of the data which is defined as the midway point of all your samples points. We use a fast way called the B+Q method. Here we by using the beta distribution and piping that into the qnorm distribution we are able to find the median.

Calculating a 95% confidence interval from the approximated sampling distribution.

```
ci <- quantile(meds,c(0.025,0.975))

(ci[1] < 0 & ci[2] > 0)
```

```
## 2.5%
## TRUE
```

This is a two step process. The first is to figure out where 95% of the data falls. We do this by using the quantile function on our sample of all of our data. We then figure out if our estimate falls into this range using boolean expressions.

The concept of coverage probability. Explain your code for calculating the coverage probability.

```
gen.ci.med <- function(n = 100, N = 201,parm.int = 0){
  data <- rnorm(N)

  median(data)

  mle.mean <- mean(data)
  mle.sd <- sqrt(((length(data)-1)/length(data))*var(data))

  meds = rbeta(n,101,101) %>% qnorm(mean =   mle.mean,sd = mle.sd)
  ci <- quantile(meds,c(0.025,0.975))

  return((ci[1] < parm.int & ci[2] > parm.int))
}
```

```r
ci.contain <- NA
for(i in 1:1000){
  ci.contain[i] <- gen.ci.med()
}

mean(ci.contain)
```

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

Coverage probability is a confidence interval for your confidence interval. Said another way its how often our data falls into the confidence interval. If we are checking a 95% confidence interval we would want our data to fall at least into that confidence interval 95% of the time.

After performing the simulation we find our data falls into the CI roughly 98% of the time. This is an accurate confidence interval. There are a few different knobs we can turn. The first is the number of times we run the simulation to find out if we converge to the true CI or if our data is more/less representative. The next knob we could turn would be instead of getting samples of the median we could use the mean which would allow us to see another central tendency of our data.