EX 03 - Data Handling & MLE

Afek Adler

2020-03-22

Last excercise we did:

- Expectency and Variance of the sample mean and sample sum
- Central limit theoram
- Bias variance decomposition of a point estimator
- Derived an unbiased estimate for $\sigma^2 (S^2)$
- Covered the student's t-distribution and chi square distribution

Today we will:

- Cover methods for point estimattion
- Get to know `dplyr` package
- Try to develop a feeling for bayesian estimation.

Loss function

A quick recap of the MSE of an estimator:

$$\text{MSE}(\hat{\Theta}) = E((\hat{\Theta} - \theta)^2)$$

The squared loss did not come from heaven but from convienince. for example, another good criterion can be:

$$\text{MAE}(\hat{\Theta}) = E(|\hat{\Theta} - \theta)|)$$

Or many other types of error function. Also, at the lecture you have seen an *example* of Bayesian estimation where $\hat{\theta}_{\text{MMSE}} = \int \theta \text{p}(\theta|\mathbf{x}) \text{d}\theta = \text{E}(\theta|\mathbf{x})$,the derivation of this formula was taken under assumption of a square loss but there are also many other bayesian estimators like the maximum a posteriori estimation - $\hat{\theta}_{\text{MAP}} = \arg\max_{\theta} p(\theta|x)$. In the end of the excercise we will go deeper into this subject.

Point estimaion

Some nice to have charactraists

- Unbiased. If $E(\hat{\theta}) = \theta$
- Consistent. If the varaince of the estimator ~0 when N tends to ∞

Remember that the sample mean is unbiased estimator of the population mean

Point estimates with the method of moments (MOM)

The first moment

$$E(X) = \frac{\sum_{i=1}^{n} X_i}{n} \Rightarrow E(X) = \bar{X}$$

The Second moment

$$E\left(X^2\right) = \frac{\sum_{i=1}^{n} X_i^2}{n} \Rightarrow$$

$$V(X) = E\left(X^2\right) - E^2(X) \Rightarrow V(X) = \frac{\sum_{i=1}^{n} X_i^2}{n} - (\bar{X})^2$$

Q1: MOM

Let

$$X = \mathcal{U}\left(\theta, \theta + 6\right)$$

Estimate θ with the method of moments

Therfore

$$E(X) = (\theta + \theta + 6)/2 = \theta + 3$$

And

$$E(X) = \bar{X} = \theta + 3$$

By the equation of the first moment. Therfore

$$\hat{\theta} = \bar{X} - 3$$

Point estimates with the mazimum likelihood estimation (MLE)

In statistics, maximum likelihood estimation (MLE) is a method of estimating the parameters of a probability distribution by maximizing a likelihood function, so that under the assumed statistical model the observed data is most probable. The point in the parameter space that maximizes the likelihood function is called the maximum likelihood estimate. The logic of maximum likelihood is both intuitive and flexible, and as such the method has become a dominant means of statistical inference.

If the likelihood function is differentiable, the derivative test for determining maxima can be applied. In some cases, the first-order conditions of the likelihood function can be solved explicitly; for instance, the ordinary least squares estimator maximizes the likelihood of the linear regression model. Under most circumstances, however, numerical methods will be necessary to find the maximum of the likelihood function. ("Wikipedia")

Q2: MLE With the binomial distribution - suppose we had a trial with 49 success ou of 80.

$$L(p) = f_D(\mathrm{H} = 49|p) = \left(\begin{array}{c} 80 \\ 49 \end{array}\right) p^{49}(1-p)^{31} \tag{1}$$

$$0 = \frac{\partial}{\partial p}\left(\left(\begin{array}{c} 80 \\ 49 \end{array}\right) p^{49}(1-p)^{31}\right), \{discardbinomialcoefficient\} \tag{2}$$

$$0 = 49p^{48}(1-p)^{31} - 31p^{49}(1-p)^{30}, \{(uv)` = u`v + v`u\} \tag{3}$$

$$= p^{48}(1-p)^{30}[49(1-p) - 31p] \tag{4}$$

$$= p^{48}(1-p)^{30}[49 - 80p] \tag{5}$$

Can be solved also by applying log on the likelihood.

It's clear that the maximum is at p = 49/80. But let's see how we do it in R using the bulit in optimize function:

```
likelihood <- function(p) {
  p^49*((1-p)^31)
}
tolerance <- 10^(-4)
pmax <- optimize(likelihood, c(0, 1), tol = tolerance  , maximum = T)[[1]]
delta <- abs(pmax- (49/80))
delta
```

```
## [1] 6.814623e-07
```

HW1 q3

Best Pracitces for data Data handling with R

R main datatypes:

- vectors
- matrices
- data.frame - matrices with meatadata, added functionallity and allow multiple data types
- tibbles - modern take on dataframes

`dplyr` is a grammar of data manipulation, providing a consistent set of verbs that help you solve the most common data manipulation challenges:

- `mutate()` adds new variables that are functions of existing variables.
- `select()` picks variables based on their names.
- `filter()` picks cases based on their values.
- `summarize()` reduces multiple values down to a single summary.
- `arrange()` sorts the rows.

```
library(tidyverse)
library(nycflights13)
```

This dataset has 19 columns so the head function is not that usefull when knitting to html. It is always useful to know how many missing values we have in our dataset, sometimes missing values are not just given to us as NA.

```
head(flights,2)
```

```
## # A tibble: 2 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>
## 1  2013     1     1      517            515         2      830
## 2  2013     1     1      533            529         4      850
## # ... with 12 more variables: sched_arr_time <int>, arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dttm>
```

```
colSums(is.na(flights))/nrow(flights)
```

```
##           year          month            day       dep_time sched_dep_time
##    0.000000000    0.000000000    0.000000000    0.024511842    0.000000000
##      dep_delay       arr_time sched_arr_time      arr_delay        carrier
##    0.024511842    0.025871796    0.000000000    0.028000808    0.000000000
##         flight        tailnum         origin           dest       air_time
##    0.000000000    0.007458964    0.000000000    0.000000000    0.028000808
##       distance           hour         minute      time_hour
##    0.000000000    0.000000000    0.000000000    0.000000000
```

```
sapply(flights,class)
```

```
## $year
## [1] "integer"
##
## $month
## [1] "integer"
##
```

```
## $day
## [1] "integer"
##
## $dep_time
## [1] "integer"
##
## $sched_dep_time
## [1] "integer"
##
## $dep_delay
## [1] "numeric"
##
## $arr_time
## [1] "integer"
##
## $sched_arr_time
## [1] "integer"
##
## $arr_delay
## [1] "numeric"
##
## $carrier
## [1] "character"
##
## $flight
## [1] "integer"
##
## $tailnum
## [1] "character"
##
## $origin
## [1] "character"
##
## $dest
## [1] "character"
##
## $air_time
## [1] "numeric"
##
## $distance
## [1] "numeric"
##
## $hour
## [1] "numeric"
##
## $minute
## [1] "numeric"
##
## $time_hour
## [1] "POSIXct" "POSIXt"
```

At home - find a better way to print the classes and the % of missing values in R

`select()` picks variables based on their names.

```
flight_ditance_airtime <- flights %>% select( distance, air_time)
flight_ditance_airtime %>% head(2)
```

```
## # A tibble: 2 x 2
##   distance air_time
##      <dbl>    <dbl>
## 1     1400      227
## 2     1416      227
```

`mutate()` adds new variables that are functions of existing variables.

```
flight_ditance_airtime %>% mutate(mean_speed = distance/air_time) %>% head(2)
```

```
## # A tibble: 2 x 3
##   distance air_time mean_speed
##      <dbl>    <dbl>      <dbl>
## 1     1400      227       6.17
## 2     1416      227       6.24
```

If you only want to keep the new variables, use `transmute()`:

```
flight_ditance_airtime %>% transmute(mean_speed = distance/air_time) %>% head(2)
```

```
## # A tibble: 2 x 1
##   mean_speed
##        <dbl>
## 1       6.17
## 2       6.24
```

`filter()` picks cases based on their values.

```
flights %>% filter(is.na(dep_delay)) %>% head(2)
```

```
## # A tibble: 2 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>
## 1  2013     1     1       NA           1630        NA       NA
## 2  2013     1     1       NA           1935        NA       NA
## # ... with 12 more variables: sched_arr_time <int>, arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dttm>
```

`arrange()` picks cases based on their values.

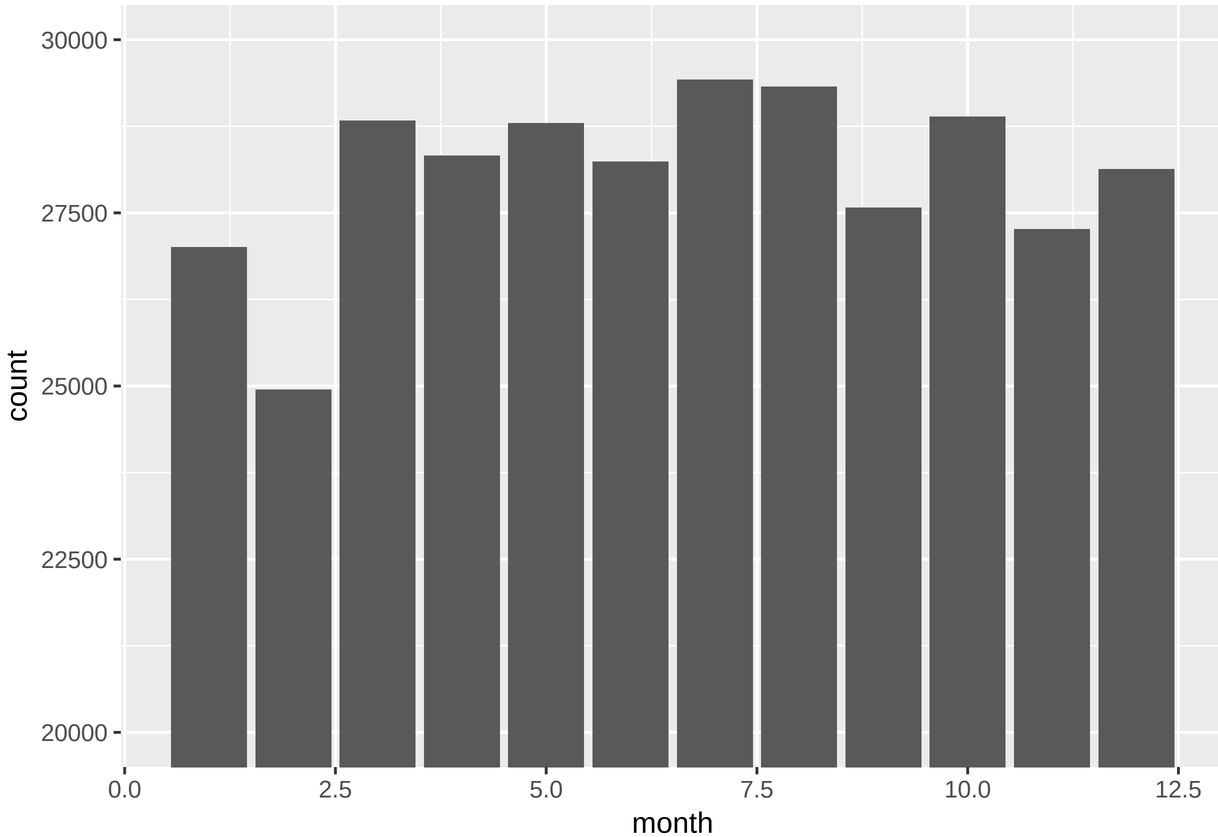
```
flights %>% arrange(desc(month)) %>% head(2)
```

```
## # A tibble: 2 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>
## 1  2013    12     1       13           2359        14      446
## 2  2013    12     1       17           2359        18      443
## # ... with 12 more variables: sched_arr_time <int>, arr_delay <dbl>,
```

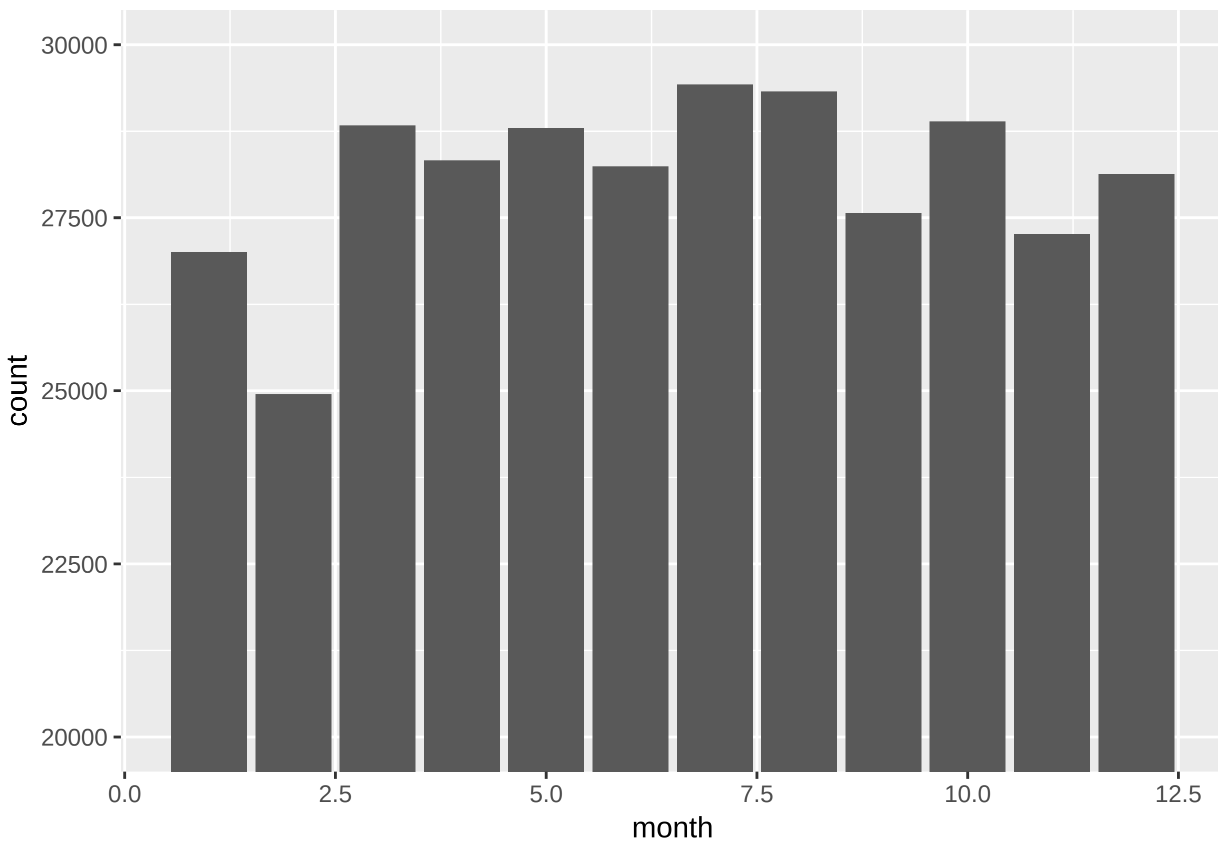
```
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dttm>
```

`summarize()` reduces multiple values down to a single summary.

```
by_month <- group_by(flights,month)
by_month %>% summarise(count = n()) %>%
  ggplot( mapping = aes(x = month, y = count)) + geom_bar(stat="identity") +  coord_cartesian(ylim = c(2
```




```
ggplot(data = flights) +
  geom_bar(mapping = aes(x = month)) +
 coord_cartesian(ylim = c(2*10^4, 3*10^4))
```



Additional resources:

- r4ds
- dplyr
- dplyr cheat sheet

Bayesian estimation

We want to minimize with respect to a given loss function -

$$\int L(\hat{\theta} - \theta) * p(\theta|x) d\theta$$

In the lecture, we have seen that when $L(\hat{\theta}, \theta) = (\hat{\theta} - \theta)^2$ than $\hat{\theta} = E(posterier)$, other types of loss functions will derive different estimators (like we have seen above). The logic of this method is as follows - we have somekind of a distribution over θ, but we need to choose only one of those. So we formulate an objective function and minimze it with respect to the parameter that we want to find. The most used ones are the mean, median, and common of that distribution, and as we said, thet are the bayesian estimators for different loss functions.