Data Science with R

Victor Coppin

2027-05-06

Table of contents

Preface

This is a Quarto book.

To learn more about Quarto books visit https://quarto.org/docs/books.

Part I

R Basics : Introduction to Data Science

1 The Tidyverse

The Tidyverse can be installed with a single line of code: install.packages("tidyverse")

This command installs the nine core packages of the Tidyverse: dplyr, forcats, ggplot2, lubridate, purrr, readr, stringr, tibble, and tidyr. These are considered the core of the Tidyverse because you'll use them in almost every analysis: - dplyr : manipulating data frames
- forcats : provides tools for dealing with categorical variables
- ggplot2 : producing statistical, or data, graphics
- lubridate : makes it easier to work with dates and times in R
- purr : working with functions and iteration in a functional programming style

#| label: load-tidyverse #| warning: false #| message: false

```
library(tidyverse)
```

```
Warning: package 'tidyverse' was built under R version 4.4.3
```

```
Warning: package 'ggplot2' was built under R version 4.4.3
```

```
Warning: package 'tibble' was built under R version 4.4.3
```

```
Warning: package 'tidyr' was built under R version 4.4.3
```

```
Warning: package 'readr' was built under R version 4.4.3
```

```
Warning: package 'purrr' was built under R version 4.4.3
```

```
Warning: package 'dplyr' was built under R version 4.4.3
```

```
Warning: package 'forcats' was built under R version 4.4.3
```

```
Warning: package 'lubridate' was built under R version 4.4.3
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.2     v tibble    3.3.0
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.0.4
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```r
library(dslabs)
```

```
Warning: package 'dslabs' was built under R version 4.4.3
```

```r
data(murders)
```

2 Manipulating Data frames with dplyr and purrr

2.1 Tidy Data

We say that a data table is in *tidy* format if each row represents one observation and columns represent the different variables available for each of these observations. The murders dataset is an example of a tidy data frame.

```r
head(murders)
```

```
       state abb region population total
1    Alabama  AL  South    4779736   135
2     Alaska  AK   West     710231    19
3    Arizona  AZ   West    6392017   232
4   Arkansas  AR  South    2915918    93
5 California  CA   West   37253956  1257
6   Colorado  CO   West    5029196    65
```

Each row represents a state with each of the five columns providing a different variable related to these states: name, abbreviation, region, population, and total murders.

2.2 Manipulating Data Frames

> "The dplyr package from the tidyverse introduces functions that perform some of the most common operations when working with data frames and uses names for these functions that are relatively easy to remember. For instance, to change the data table by adding a new column, we use mutate. To filter the data table to a subset of rows, we use filter. Finally, to subset the data by selecting specific columns, we use select."

2.2.1 The `mutate` function

The `mutate` function is used to add new columns to a data frame or modify existing ones.

```
# Add a new column 'rate' to the murders data frame
murders  <- mutate(murders, rate = total / population * 100000)
```

Note: to compute the rate, we used `total` and `population` columns, which are not defined in the global environment. The `mutate` function allows us to use the names of the columns directly.

> "This is one of dplyr's main features. Functions in this package, such as mutate, know to look for variables in the data frame provided in the first argument.
> In the call to mutate above, `total` will have the values in `murders$total`. This approach makes the code much more readable and concise."

```
head(murders)
```

```
       state abb region population total     rate
1    Alabama  AL  South    4779736   135 2.824424
2     Alaska  AK   West     710231    19 2.675186
3    Arizona  AZ   West    6392017   232 3.629527
4   Arkansas  AR  South    2915918    93 3.189390
5 California  CA   West   37253956  1257 3.374138
6   Colorado  CO   West    5029196    65 1.292453
```

Note: the `mutate` function does not change the original data frame.

> "Although we have overwritten the original **murders** object, this does not change the object that is loaded with `data(murders)`.
> If we load the murders data again, the original will overwrite our mutated version."

2.2.2 Subsetting with `filter`

The `filter` function is used to subset rows based on logical conditions.

Filter the murders data frame to include only the entries for which the murder rate is lower than 0.71.

```
# Syntax : data, conditional statement.
filter(murders, rate <= 0.71)
```

```
          state abb        region population total      rate
1        Hawaii  HI          West    1360301     7 0.5145920
2          Iowa  IA North Central    3046355    21 0.6893484
3 New Hampshire  NH     Northeast    1316470     5 0.3798036
4  North Dakota  ND North Central     672591     4 0.5947151
5       Vermont  VT     Northeast     625741     2 0.3196211
```

2.2.3 Selecting columns with `select`

The `select()` function is used to extract specific columns from a data frame.

In the example below: - We create a new data frame containing only the columns state, region, and rate. - We then apply `filter()` to keep only the rows where the murder rate is less than or equal to 0.71.

```
state_region_rate_table <- select(murders, state, region, rate)
filter(state_region_rate_table, rate <= 0.71)
```

```
          state        region      rate
1        Hawaii          West 0.5145920
2          Iowa North Central 0.6893484
3 New Hampshire     Northeast 0.3798036
4  North Dakota North Central 0.5947151
5       Vermont     Northeast 0.3196211
```

2.2.4 Exercises

1. Load the dplyr package and the murders dataset.

```
library(dplyr)
library(dslabs)
data(murders)
```

2. Use the function mutate to add a column rank containing the rank, from highest to lowest murder rate. Make sure you redefine murders so we can keep using this variable.

```
murders <- mutate(murders, rate = total / population * 10^5)
murders <- mutate(murders, rank = rank(-rate))
murders %>% head()
```

```
       state abb region population total     rate rank
1    Alabama  AL  South    4779736   135 2.824424   23
2     Alaska  AK   West     710231    19 2.675186   27
3    Arizona  AZ   West    6392017   232 3.629527   10
4   Arkansas  AR  South    2915918    93 3.189390   17
5 California  CA   West   37253956  1257 3.374138   14
6   Colorado  CO   West    5029196    65 1.292453   38
```

```
select(murders, state, population) %>% head()
```

```
       state population
1    Alabama    4779736
2     Alaska     710231
3    Arizona    6392017
4   Arkansas    2915918
5 California   37253956
6   Colorado    5029196
```

We can write population rather than murders$population. The function mutate knows we are grabbing columns from murders.

3. Use **select** to show the state names and abbreviations in murders. Do not redefine murders, just show the results.

```
select(murders, state, abb)
```

```
                  state abb
1               Alabama  AL
2                Alaska  AK
3               Arizona  AZ
4              Arkansas  AR
5            California  CA
6              Colorado  CO
7           Connecticut  CT
8              Delaware  DE
9  District of Columbia  DC
10              Florida  FL
11              Georgia  GA
12               Hawaii  HI
13                Idaho  ID
14             Illinois  IL
```

```
15              Indiana  IN
16                 Iowa  IA
17               Kansas  KS
18             Kentucky  KY
19            Louisiana  LA
20                Maine  ME
21             Maryland  MD
22        Massachusetts  MA
23             Michigan  MI
24            Minnesota  MN
25          Mississippi  MS
26             Missouri  MO
27              Montana  MT
28             Nebraska  NE
29               Nevada  NV
30        New Hampshire  NH
31           New Jersey  NJ
32           New Mexico  NM
33             New York  NY
34       North Carolina  NC
35         North Dakota  ND
36                 Ohio  OH
37             Oklahoma  OK
38               Oregon  OR
39         Pennsylvania  PA
40         Rhode Island  RI
41       South Carolina  SC
42         South Dakota  SD
43            Tennessee  TN
44                Texas  TX
45                 Utah  UT
46              Vermont  VT
47             Virginia  VA
48           Washington  WA
49        West Virginia  WV
50            Wisconsin  WI
51              Wyoming  WY
```

4. Use filter to show the top 5 states with the highest murder rates.

```
filter(murders,  rank <= 5)
```

```
                 state abb        region population total       rate rank
```

```
1 District of Columbia  DC         South     601723    99 16.452753    1
2            Louisiana  LA         South    4533372   351  7.742581    2
3             Maryland  MD         South    5773552   293  5.074866    4
4             Missouri  MO North Central    5988927   321  5.359892    3
5       South Carolina  SC         South    4625364   207  4.475323    5
```

5. Create a new data frame called no_south that removes states from the South region. How many states are in this category? You can use the function nrow for this.

Note: We can remove rows using the != operator. For example, to remove Florida, we would do this:

```
no_florida <- filter(murders, state != "Florida")
```

```
# Create the new data frame without south region
no_south <- filter(murders, region != "South")
# Compute how many states are not in the south
select(no_south, state) %>% nrow()
```

```
[1] 34
```

There are 34 states which are not in the south

Part II

ggplot2: Elegant Graphics for Data Analysis

3

Part III

Foundations of Statistical Analysis and Machine Learning

4

5 Mean Quadratic Error

The MQE is a measure of how close the estimator is to the true parameter value.

To compare estimator we can compute the mean quadratic Eroor, denoted by MQE :

$$\text{MQE}(\hat{\theta}_n) = \text{Var}\left(\hat{\theta}_n\right) + \left(b_\theta\left(\hat{\theta}_n\right)\right)^2$$

where $\beta_\theta(\hat{\theta}_n) = \mathbb{E}\left[\hat{\theta}_n\right] - \theta$ is the bias of the estimator $\hat{\theta}_n$.

We say that $\hat{\theta}_{n,1}$ is better than $\hat{\theta}_{n,2}$ if :

$$\forall n,\ \text{MQE}(\hat{\theta}_{n,1}) \leq \text{MQE}(\hat{\theta}_{n,2})$$

6 Example :

Let concider :
- $\widehat{\theta}_{n,1} = \max(X_k)$ and $\widehat{\theta}_{n,4} = \frac{n+1}{n} \cdot \widehat{\theta}_{n,1}$

We have :

- $\text{MQE}(\widehat{\theta}_{n,1}) = \dfrac{2\theta^2}{(n+1)(n+2)}$
- $\text{MQE}(\widehat{\theta}_{n,4}) = \dfrac{\theta^2}{n(n+1)}$

$\forall n \geq 2,\ MQE(\widehat{\theta}_{n,4}) < MQE(\widehat{\theta}_{n,1})$

Thus, we can conclude that $\widehat{\theta}_{n,4}$ is better than $\widehat{\theta}_{n,1}$

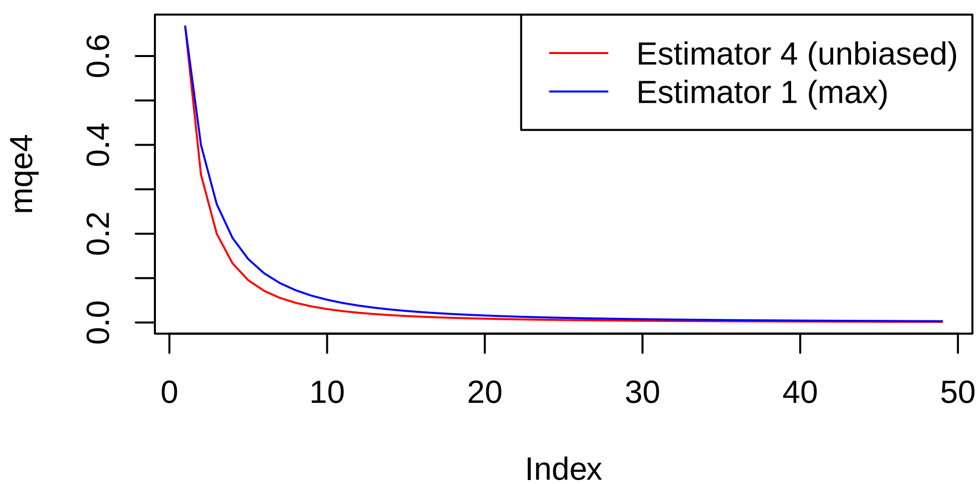
Remark: $\widehat{\theta}_{n,4}$ is the best among the two estimators we have considered. Since $\widehat{\theta}_{n,4}$ is unbiased, we know that for any unbiased estimator $\widehat{\theta}_n$, we have:

$$\text{Cramer Rao-Bound } \leq \text{Var}\left(\hat{\theta}_n\right)$$

If $\text{Var}\left(\widehat{\theta}_{n,4}\right)$ equals the Cramer-Rao bound, then the estimator cannot be improved; otherwise, improvement is possible.

7 Convergence Illustration in R

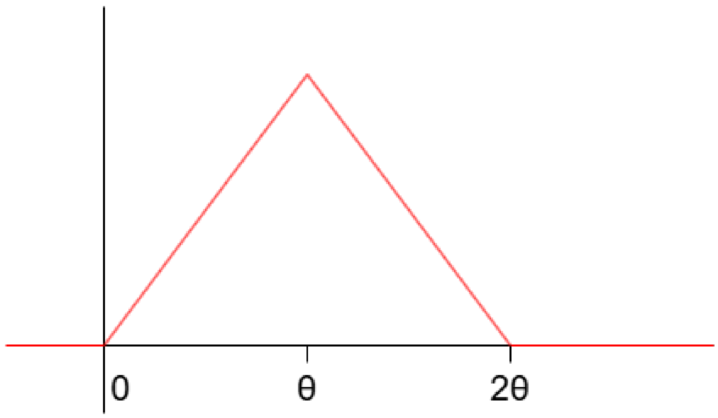
```
theta <- 2 # assumed true value of the parameter
# MQE or Variance of the estimators
mqe1 <- 2 * theta^2 / ((2:50 + 1) * (2:50 + 2))
mqe4 <- theta^2 / (2:50 * (2:50 + 1))
plot(mqe4, type = "l", col = "red")
lines(mqe1, col = "blue")
legend("topright",
       legend = c("Estimator 4 (unbiased)", "Estimator 1 (max)"),
       col = c("red", "blue"), lty = 1)
```



This plot shows that the unbiased estimator $\hat{\theta}n, 4$ consistently outperforms the maximum estimator $\hat{\theta}n, 1$ in terms of MQE, even for relatively small sample sizes (e.g., $n = 10$). However, as the sample size increases, the MQEs of both estimators get closer, meaning the performance gap narrows — although $\hat{\theta}_{n,4}$ remains superior for all n.

8 Inverse Transform Sampling

From FSML2 exercice we get the following CFD from the graph below



$$F_X(t) = \begin{cases} 0 & \text{if } t < 0 \\ \frac{t^2}{2\theta^2} & \text{if } t \in [0, \theta] \\ -\frac{t^2}{2\theta^2} + \frac{2t}{2\theta} - 1 & \text{if } t \in (\theta, 2\theta) \\ 1 & \text{if } t \geq 2\theta \end{cases}$$

The computation of the inverse function, $F_X(t)^{-1}$ give us :

$$F_X^{-1} : [0, 1] \to [0, 2\theta]$$

$$F_X^{-1}(t) = \begin{cases} \sqrt{2\theta^2 \cdot t} & \text{if } t \in [0, \frac{1}{2}] \\ 2\theta - \sqrt{2\theta^2 \cdot (1 - t)} & \text{if } t \in (\frac{1}{2}, 1] \end{cases}$$

Which could be written as a sum with indicator functions as:

$$F_X^{-1}(t) = \sqrt{2\theta^2 \cdot t}\, \mathbb{1}_{t\in[0,\frac{1}{2}]} + 2\theta - \sqrt{2\theta^2 \cdot (1-t)}\, \mathbb{1}_{t\in]\frac{1}{2},1]}$$

Note: becareful to count just one time the value $tfrac12$

8.1 Inverse density function with R

Thanks to the last equation form, we can write $F_X^{-1}(t)$ in R easily :

The logical expressions like (t <= 1/2) and (t > 1/2) act as "indicator functions".
In R, TRUE is treated as 1 and FALSE as 0 in arithmetic operations. This means only the correct formula is applied for each value of t.
For example, if t = 0.3, (t <= 1/2) is TRUE (1), so the first formula is used. If t = 0.7, (t > 1/2) is TRUE (1), so the second formula is used.

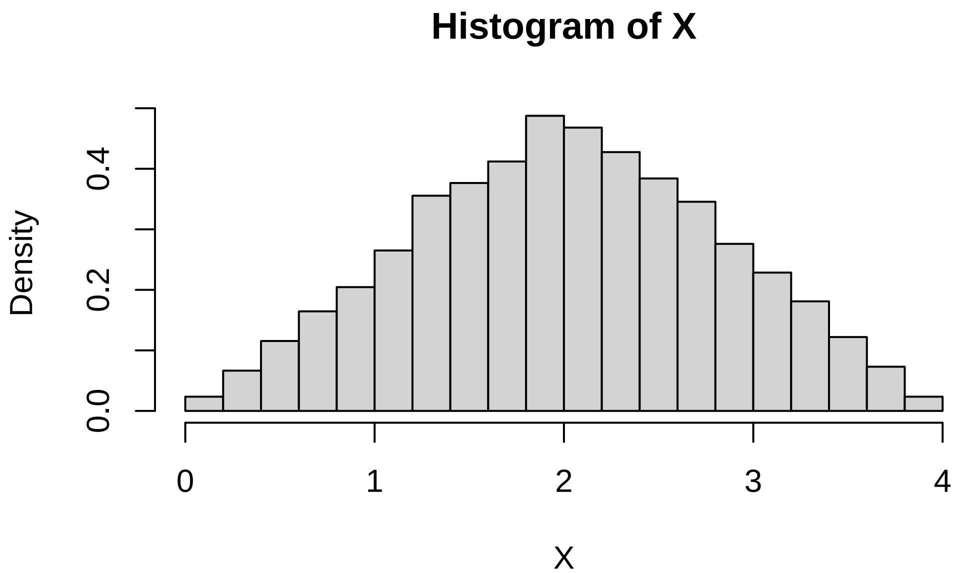
```
# Generate 10,000 random numbers uniformly distributed between 0 and 1
A <- runif(10000)

# Define the inverse transform function
invFX <- function(t, theta) {
# Logical expressions act as indicators (see explanation above)
  sqrt(2 * t * theta^2) * (0 <= t) * (t <= 1 / 2) +
    (2 * theta - sqrt(2 * theta^2 * (1 - t))) * (t > 1 / 2) * (t <= 1)
}

theta <- 2  # Set the parameter theta
X <- invFX(A, theta)   # Apply the inverse transform to the uniform random numbers
head(X)  # Display the first few values
```

```
[1] 1.688240 2.145106 1.741727 3.938706 1.986833 2.227638
```

Remark: In R, you do not need to use 'return()' if the value to return is the last line of the function. This is a common style in R, especially for simple functions.

8.2 Display the value of X

```r
hist(X,freq=FALSE)
```



8.3 Simulation of a density function thanks to uniform random variable

"We recognize the function f. To generate samples from a random variable X with an unknown density function, it is sufficient to know the inverse of its cumulative distribution function (i.e., $F_X^{-1}(t)$). By applying this inverse to samples from a uniform distribution, we can simulate values from X."

Part IV

Advanced Statistical Analysis and Machine Learning

9

Part V

Time Series Analysis

10

Part VI

Statistical Analysis of Massive and High Dimensional Data

11

12 Summary

In summary, this book has no content whatsoever.

References