Small Errors, Big Consequences: Lessons from Data Mismanagement in Business

Karla Hernández 400916125

Fresenius University of Applied Science

Data Science and Data Analytics (WS 2025/26)

Prof. Dr. Stephan Huber

2025-02-17

Author Note

Correspondence concerning this article should be addressed to Karla Hernández 400916125, Email: hernandez.karla@stud.hs-fresenius.de

Abstract

Data plays a major role in business decisions, from production and hiring to marketing and forecasting. However, even tiny mistakes in a dataset can lead to completely wrong conclusions and very expensive consequences; so if the data has mistakes, the decisions will have mistakes. The challenge is that many data errors are small and easy to overlook, especially when working in RStudio, because R does not warn us when a result is logically wrong; it only warns us when the code fails. This handout explains how small issues such as missing values, duplicated rows, incorrect data types or poorly executed joins can quietly break calculations and models. Using simple examples of bad practice vs better practice in R, you'll learn how common habits can hide errors and how better habits prevent them. The goal is not about perfection, but about developing good routines that help us catch mistakes early, before they affect real business decisions.

Small Errors, Big Consequences: Lessons from Data Mismanagement in Business

Word count: 936

1 "Small" Erros

When people think about data problems in business, they often imagine something dramatic; like a system failure, a cyberattack, or a huge coding bug. But in reality, business data disasters usually don't come from big dramatic mistakes. They come from very small, very ordinary mistakes that slip through because nobody notices them.

Here's why:

- Companies collect tons of data every day.
- Teams are under pressure to work fast and deliver results.
- Everyone assumes that software will catch mistakes automatically.
- RStudio runs perfectly even if the data is wrong.

So a tiny mistake, like a missing value or a duplicated customer, might not look dangerous at first. But once that mistake is used in reports, dashboards, forecasts, or decisions… it suddenly becomes a big problem with big consequences.

2 What May Count as a "Small Error"?

Here are some examples of small mistakes that can break results later:

Table 1

What may be considered as "small error" examples

Error	What it Means
"Mexico" vs "México"	R treats them as two different categories
Number stored as text	Math doesn't work the way we expect
Missing values	Averages and totals become wrong
Duplicate rows	Customers or employees counted twice
Wrong join	Rows multiply or disappear silently

3 Tiny Erros, Big Consequences

The previous mistakes look tiny, but later they affect totals, predictions, and business decisions. Let's take now a look of how these errors can be related to real life examples:

Table 2

Examples in real life

Area	Tiny Error	Big Result
Retail	Missing value in sales	Too little production that leads to lost on sales
HR	Employee duplicated during merge	Over-hired people that leads to unnecessary cost
Marketing	Duplicated customers	Overspent on ads

4 Bad Practice vs Better Practice in R

Now, we will see some examples on how to avoid small mistakes in R and replace risky habits with safer, clearer, and more reliable practices. Many of the "bad practices" shown below are considered wrong mainly because even though R runs them perfectly fine, the results are quietly broken. This means we will get broken results used in reports, dashboards, or business decisions, and here is when the damage is done.

This section is about learning to recognize moments where we *assume* something instead of **checking**, we let R *"guess"* instead of giving **clear instructions**, we work *faster* instead of **safer**, and we *trust* the output simply because **"there was no error message"**.

Now we'll see some tasks and how we can avoid bad practices with examples of the best common practices:

- **Importing:** R doesn't convert text into categories automatically.
  - *Bad Practice:* `read.csv("file.csv")`
  - *Good Practice:* `read.csv("file.csv", stringsAsFactors = FALSE)`
- **Missing values:** Get real averages instead of NA results that look "valid" but are unusable.
  - *Bad Practice:* `mean(x)`
  - *Good Practice:* `mean(x, na.rm = TRUE)`
- **Joining tables:** Avoid accidental duplicates and can see which rows didn't match.
  - *Bad Practice:* `merge(df1, df2)`

  - *Good Practice:* `left_join(df1, df2, by = "ID")` + `anti_join()`

- **Data types:** Verify and enforce data types instead of hiping R guessed correctly. ***Only applicable for versions prior to R 4.0***
  - *Bad Practice:* Assuming R knows the type
  - *Good Practice:* `str()`

- **Checking data:** Catch mistakes before modeling instead of discovering them too late.
  - *Bad Practice:* Not looking at the data
  - *Good Practice:* `summary()`, `head()`, `table()`, `view()`

- **Filtering:** Keeps the dataset clean and consistent instead of returning unpredictable formats.
  - *Bad Practice:* `data[data$age > 18]`
  - *Good Practice:* `filter(data, age > 18)`

- **Removing duplicates:** Removes duplicates but also lets us see exactly which rows were duplicated.
  - *Bad Practice:* `unique(data)`
  - *Good Practice:* `distinct(data, .keep_all = TRUE)`

- **Renaming columns:** Rename the correct column even if column order later changes.
  - *Bad Practice:* `colnames(data) [3] <- "sales"`
  - *Good Practice:* `rename(data, sales = old_name)`

- **Selecting columns:** Safer because column names don't change, column numbers do.
  - *Bad Practice:* `data[, c(1,2,5]`
  - *Good Practice:* `select(data, name, price, region)`

- **Dates:** Prevents R from guessing the wrong date format, which happens often.
  - *Bad Practice:* `as.Date(Date)`
  - *Good Practice:* `as.Date(date, format="%d/%m/%Y")` or `lubridate::dmy()`

- **Reading Excel:** Makes sure to always read the correct sheet even if the sheet order changes.

  - *Bad Practice:* `read:excel("file.xlsx")`
  - *Good Practice:* `read_excel("file.xlsx", sheet="Sales2024)`

- **Plotting:** Control exactly what gets plotted instead of letting R "guess".
  - *Bad Practice:* `plot(data)`
  - *Good Practice:* `ggplot(data, aes(x, y))` + `geom_line()`

- **Pipelines:** >filter()
  - *Bad Practice:* Many separate steps
  - *Good Practice:* 'data

- **Saving:** Always keep the original data safe in case of errors.
  - *Bad Practice:* Overwriting the raw file
  - *Good Practice:* Save a separate clean file

By comparing bad practices vs better practices, we can clearly see how very small choices in code, tiny details, can be the difference between a *result that is silently wrong* and a **result that can be trusted.**

5 Lessons from Data Mismanagement in Business

1. A tiny mistake in data can cause a huge mistake in a business decision.
2. RStudio won't warn you when the result is wrong, only when the code is.
3. Never assume the data is clean, always check it.
4. Good analysis starts before modeling with data validation.
5. Being explicit in your code prevents disasters.
6. Documentation and reproducible scripts protect you from mistakes.
7. Joins are one of the most dangerous operations, check them twice.
8. Data types matter more than most people realize.
9. Peer review is essential.
10. Great analysts aren't those who make no mistakes, they are the ones who catch mistakes early.

6 Affidavit

I hereby affirm that this submitted paper was authored unaided and solely by me. Additionally, no other sources than those in the reference list were used. Parts of this paper, including tables and figures, that have been taken either verbatim or analogously from other works have in each case been properly cited with regard to their origin and authorship. This paper either in parts or in its entirety, be it in the same or similar form, has not been submitted to any other examination board and has not been published.

I acknowledge that the university may use plagiarism detection software to check my thesis. I agree to cooperate with any investigation of suspected plagiarism and to provide any additional information or evidence requested by the university.

Checklist:

- ☒ The handout contains 3-5 pages of text.
- ☒ The submission contains the Quarto file of the handout.
- ☒ The submission contains the Quarto file of the presentation.
- ☒ The submission contains the HTML file of the handout.
- ☒ The submission contains the HTML file of the presentation.
- ☒ The submission contains the PDF file of the handout.
- ☒ The submission contains the PDF file of the presentation.
- ☒ The title page of the presentation and the handout contain personal details (name, email, matriculation number).
- ☒ The handout contains a bibliography, created using BibTeX with an APA citation style.
- ☒ Either the handout or the presentation contains R code that proofs the expertise in coding.
- ☒ The filled out Affidavit.
- ☒ The link to the presentation and the handout published on GitHub.
- ☐ In group work, each student's contribution is clearly defined, and individual performance can be assessed using specified sections, page numbers, or other objective criteria.

[Karla Hernández,] [2025-02-17,] [Köln, Germany]