ISSN 2607-9917 www.bdd.rdplf.org Volume 9, n° 2, June 2026 https://doi.org/10.25796/bdd.v9i2.87112

Bulletin de la Dialyse à Domicile

Home Dialysis Bulletin (BDD)

Journal international bilingue pour partager les connaissances et l'expérience en dialyse à domicile

(English edition) (French version available at same address)

Four Best Practices for Using ChatGPT/Claude as an Effective R Assistant

(Quatre bonnes pratiques pour faire de ChatGPT/Claude un vrai assistant R)

Claire Della Vedova

Biostatistics and clinical methodology consultant, R specialist

Manosque (France)

To cite: Della Vedova C. Four Best Practices for Using ChatGPT/Claude as an Effective R Assistant. Bull Dial Domic [Internet]. [cited 2026 May 29];9(2). doi: https://doi.org/10.25796/bdd.v9i2.87112

Summary

Artificial intelligence can be a considerable help to healthcare professionals who wish to publish their work; however, using it without a critical eye can easily lead to completely false results.
We thought it would be helpful to ask a biostatistician and trainer to raise our readers' awareness of this software's potential pitfalls when analyzing their data, specifically within the context of the R software widely used in medical research.
This article first presents four best practices for effectively using large language models (LLMs) such as ChatGPT or Claude in an R programming context. The author begins by describing the three most common errors produced by LLMs: inventing nonexistent functions, using obsolete syntax, and, even more seriously, suggesting inappropriate statistical methods. These errors can go unnoticed by beginners and lead to erroneous analyses. It is therefore essential to systematically verify the suggested functions, pay attention to R's warning messages, and understand the conditions under which the recommended statistical tests apply.
The article then details a three-step method for obtaining more reliable results: providing precise context for the data and the working environment, clearly describing the desired objective, and specifying the expected output format. The author places particular emphasis on data confidentiality.
The third section explains how to verify that generated code actually works. Four checks are suggested: ensuring there are no errors or warnings, verifying the structure of the results, assessing the plausibility of the obtained values, and testing the code on a small set of known data.
Finally, the article specifies when to prioritize an LLM, when to consult the official R documentation, and when to seek the advice of a competent human. The overall goal is to empower the user to use LLMs as assistive tools rather than substitutes for critical thinking.

Résumé

L'intelligence artificielle, peut apporter une aide considérable à celles et ceux qui souhaitent publier leurs travaux ; néanmoins son utilisation, sans esprit critique, peut aboutir à des résultats totalement faux. Il nous a paru utile de demander à une biostatisticienne et formatrice de sensibiliser nos lecteurs et lectrices aux pièges possibles de ces logiciels pour l'analyse de leurs données , en prenant l'exemple d'un logiciel très utilisés dans les travaux médicaux.
Cet article présente quatre bonnes pratiques pour utiliser efficacement les modèles de langage (LLM) comme ChatGPT ou Claude dans un contexte de programmation en R. L'auteur commence par décrire les trois erreurs les plus fréquentes produites par les LLM : l'invention de fonctions inexistantes, l'utilisation de syntaxes obsolètes et, plus grave encore, la proposition de méthodes statistiques inadaptées. Ces erreurs peuvent passer inaperçues chez les débutants et conduire à des analyses erronées. Il est donc essentiel de vérifier systématiquement les fonctions proposées, de prêter attention aux messages d'avertissement de R et de comprendre les conditions d'application des tests statistiques recommandés.
L'article détaille ensuite une méthode en trois étapes permettant d'obtenir des réponses plus fiables : fournir un contexte précis sur les données et l'environnement de travail, décrire clairement l'objectif recherché et préciser le format attendu de la réponse. L'auteure insiste particulièrement sur la confidentialité des données.
Une troisième partie explique comment vérifier qu'un code généré fonctionne réellement. Quatre vérifications sont proposées : contrôler l'absence d'erreurs et de "warnings", vérifier la structure des résultats, évaluer la plausibilité des valeurs obtenues et tester le code sur un petit jeu de données connues.
Enfin, l'article précise quand privilégier un LLM, quand consulter la documentation officielle de R et quand demander l'avis d'un humain compétent. L'objectif global est de rendre l'utilisateur autonome tout en utilisant les LLM comme outils d'assistance et non comme substituts à la réflexion critique.

Correspondence : Claire Della-Vedova -email : claire@delladata.fr - Site : https://delladata.fr

Keywords: R software, statistics, LLM, ethics, nephrology, dialysis

Mots-clés : logiciel R, statistiques, LLM, éthique, néphrologie, dialyse

journal officiel du Registre de Dialyse Péritonéale de Langue Française **RDPLF** www.rdplf.org

www.bdd.rdplf.org Volume 9, n° 2, June 2026
https://doi.org/10.25796/bdd.v9i2.87112
ISSN 2607-9917

Editor's note
Claire Della Vedova is a biostatistics engineer and data analyst. She uses the R software daily to analyze data. She has worked for over 15 years in the fields of the environment and health, and has trained numerous students and researchers in the use of R. Since November 2017, she has been running the blog "Statistics and R Software," which aims to help beginners better understand classical statistical methods and use R more effectively, particularly through tutorials: https://delladata.fr/blog. A few years ago, she wrote several introductory articles on this software in the BDD [1-6]. R is available for free download at https://www.r-project.org.

I'm being asked this question more and more often: "Now that ChatGPT writes R code, is it still worth learning R?"
My answer is yes! More than ever!
AI amplifies a skill; it doesn't create it. Without a foundation, it's impossible to verify what AI produces, correct an error, or adapt its code to your data. So, to use ChatGPT or Claude effectively with R, you must first master the fundamentals. To lay this foundation, I created my very first video course, accessible and designed for those who are self-taught: Getting Started With R and RStudio—The Essentials in 2 Hours and 15 Minutes (see Appendix I).

1. LLMs' Three Typical Errors in R

LLMs (ChatGPT, Claude, Copilot…) are trained on millions of lines of R code, but still regularly produce errors that a beginner might not notice. Here are the three main types to be aware of.

1.1 Error #1: The function that doesn't exist

The LLM invents a function that seems plausible but doesn't exist, or it assigns a function to the wrong package.

Question asked to ChatGPT:
"How do I calculate a standardized Z-score on my age column?"

Response provided:

```
library(tidyverse)
mydata <- mydata %>%
  mutate(z_age = standardize(age))
```

The problem:
The standardize() function does not exist in the tidyverse.
You must either use scale(age) (base R) or (age - mean(age)) / sd(age).

How can you spot this?
The code throws an error: "could not find function 'standardize'":

```
Error in `mutate()`:
i In argument: `z_age = standardize(age)`.
Caused by error in `standardize()`:
! could not find function «standardize»
```

If you type a function name in the RStudio console preceded by a "?" (for example: ?standar-

journal officiel du Registre de Dialyse Péritonéale de Langue Française RDPLF www.rdplf.org

www.bdd.rdplf.org Volume 9, n° 2, June 2026
https://doi.org/10.25796/bdd.v9i2.87112
ISSN 2607-9917

dize), and the documentation finds nothing, it's most likely a hallucination!

1.2 Error #2: Obsolete syntax

The LLM suggests syntax that worked 2 or 3 years ago but is now deprecated or has been replaced.

Typical examples:

• Suggesting `summarize_each(funs(mean))` instead of `summarize(across(everything(), mean))` (the first form has been obsolete since dplyr 1.0)

• Using `aggregate(data, by=list(group), FUN=mean)` is much more readable and efficient than `group_by() %>% summarize()`

• Mixing packages that are no longer maintained (e.g., plyr when dplyr has been the standard for a long time)

How can you spot this?

The code works, but RStudio displays a warning message such as "summarize_each() was deprecated in dplyr 1.0.0."

Pay attention to the warnings returned by R and displayed in the console—that's what they're there for!

Rule of thumb:

If a deprecation warning appears, ask the LLM, "Give me the current, non-deprecated version."

1.3 Error #3: Bad statistical advice

This is the most dangerous error, because the code works, but the analysis is wrong. The LLM suggests an inappropriate test, forgets to check the conditions of application, or interprets the results with too much confidence.

Typical examples:

• Suggesting a t-test on data that is clearly not normally distributed, without checking anything

• Using a chi-square test when the expected counts are less than 5 (the correct choice would be Fisher's exact test)

• Comparing four groups in pairs using t-tests without correcting for multiple testing

• Interpreting a p-value < 0.05 as "evidence of an effect" without considering power or effect size

How to spot it:

No errors or warnings are returned! That's why you need to be vigilant. Always ask the LLM: "What are the conditions for applying this test? How can they be verified?"

Rule of thumb:

Code alone isn't enough. Before applying a test, understand why this particular test is being used, not another. If you're unsure, it's time to seek expert human advice!

2. The Three-Step Prompting Method

journal officiel du Registre de Dialyse Péritonéale de Langue Française RDPLF www.rdplf.org

www.bdd.rdplf.org Volume 9, n° 2, June 2026
https://doi.org/10.25796/bdd.v9i2.87112
ISSN 2607-9917

The quality of the code an LLM returns depends 80% on the quality of your prompt. A sloppy prompt yields generic code that's poorly suited to your data and will require three rounds of rework. A structured prompt often yields code that's usable on the first try.
The method consists of three parts: context + objective + expected format.

2.1 Step 1: Context: Describe your data and your environment

The LLM cannot see your screen. It knows nothing about your table's structure, the packages you use, or the version of R you have installed. If you don't tell it anything, it will guess—and not always correctly.

What you need to provide:

- **The nature of the dataset**: how many rows, how many columns, what type of data (numeric, categorical, dates…)
- **The exact names of the columns** relevant to your question
- **The package(s)** you want to use (tidyverse? base R? ggplot2?)
- **The data structure**, not the data itself (we'll get to that next)

> Confidentiality: Do not paste your actual data into the LLM
>
> This is a habit you should adopt from the very beginning. If you're working with medical, HR, financial, educational, or customer data, or any other sensitive content, you are not permitted (either legally or ethically) to send it to ChatGPT, Claude, or Copilot. This data is sent to third-party servers and may be used to train future models. Even when it's not illegal, it's a bad practice.

What to do instead:

- **Describe the structure:** "I have a numeric age column, a 4-category diagnosis column, and a date_inclusion column in Date format."
- **Paste the output of `str(data)` or `glimpse(data)`:** It shows the data types and a few examples, without revealing the entire content.
- **Create a small fictional dataset** that resembles yours, with five made-up rows that share the same structure.
- **Anonymize the data** if you absolutely must show an excerpt (replace names, scramble identifiers, round off sensitive values).

Example of a well-defined context, without data leakage:

"I have a 320-row, 8-column 'patients' data frame, imported using read_csv(). The relevant columns are age (numeric, between 18 and 92), gender (2-level categorical: F/M), and systolic_blood_pressure (numeric, with about 5% NA). I'm working with the tidyverse."
The LLM has everything it needs to write correct code for you, without having seen a single row of your table.

2.2 Step 2: The goal: Say what you want to achieve, not how

When we're just starting out, we tend to write our prompts as technical requests:

"How do I do a group_by() followed by a summarize() with mean()?"

This isn't a good approach because you're imposing a solution on the LLM before it has even

journal officiel du Registre de Dialyse Péritonéale de Langue Française RDPLF www.rdplf.org

www.bdd.rdplf.org Volume 9, n° 2, June 2026
https://doi.org/10.25796/bdd.v9i2.87112
ISSN 2607-9917

understood your need. And if your solution isn't the right one, it will code it for you anyway! The objective should instead describe the expected result, allowing the LLM to choose the function.

Avoid:

"Give me a mutate() with case_when() to create an age_category variable."

Instead, try:

"I want to create a new column that classifies patients into three age groups: under 40, 40 to 65, and over 65."

The second phrasing allows the LLM to propose the cleanest solution. The first one locks it into a choice that may not be optimal.

2.3 Step 3: The expected format: Specify how you want the answer

This is a step that's often overlooked, but it will save you time. If you don't specify anything, the LLM will respond with a long block of explanations, alternatives, "there's also…," and you'll have to dig through it to find the useful code.

What you can ask for:

- **Code only**, without explanations ("respond only with the code block, without comments")
- **Code commented line by line** if you want to learn
- **Multiple versions** ("suggest two ways to do this: one using base R, one using tidyverse")
- **A specific output format** ("the result must be a data frame with exactly 3 columns: group, mean, standard deviation")

Example:

"Respond with a single block of tidyverse code, commented in English that produces a data frame sorted by descending mean."

The assembled prompt:

Here's what a complete prompt following the method looks like:

Background: I have a "patients" data frame with 320 rows and 8 columns, containing the variables age (numeric), sex (categorical: F/M), and systolic_blood_pressure (numeric, with a few NA values). I am working with the tidyverse.

Objective: I want to create a new column, age_category, that classifies patients into three age groups: under 40, 40 to 65, and over 65.

Expected format: A single block of tidyverse code, commented in English, that adds this new column to the existing data frame.

2.3.1 Tip

The benefit isn't just speed (you can dictate 3 to 4 times faster than you can type). It's also that

Dictate your prompts
Writing a structured prompt can be time-consuming, but you can use voice dictation. On a smartphone, the microphone icon on the keyboard works very well. On a computer, ChatGPT and Claude have a built-in microphone button.

www.bdd.rdplf.org Volume 9, n° 2, June 2026
https://doi.org/10.25796/bdd.v9i2.87112
ISSN 2607-9917

you phrase things more naturally. When speaking, you spontaneously explain the context, what you want, and how you want it. It's exactly the three-part structure, but without conscious effort.

You can simply say, "So here's the thing: I have a table of 320 patients with columns for age, gender, and blood pressure. I'd like to create a new column that classifies patients into three age groups—under 40, 40 to 65, and over 65—and just give me the tidyverse code with comments in English." The LLM understands this spoken language very well, and you'll get just as precise a response as with a carefully written prompt.

2.3.2 Why does it work?

This structure forces the LLM to focus on your specific problem, not on the generic problem it thinks it recognizes. It reduces hallucinations (as seen in Section 1) because by providing context, you're implicitly telling it, "Don't give me a function that doesn't apply to this data."
With a little practice, dictating a structured prompt takes 20 seconds and saves you 10 minutes of debugging.

Rule of thumb:

Before sending your prompt, reread it and check that it contains all three blocks. If one is missing, add it. This habit will become automatic in a few days.

3. How Do You Verify That the Generated Code Actually Works?

The LLM has returned a nice block of code. It looks clean and well-indented, with reassuring comments. The temptation is to copy and paste it into your script and move on.
That's exactly when the trouble can start.
Code can run without errors and still produce the completely wrong result. So verification isn't a luxury! Here are the four checks to perform in sequence.

3.1 Check 1: Does the code run without errors or warnings?

- **Run the entire block**, not line by line. Some errors only appear in context.
- Read the console: not just the last line, but also the intermediate messages.
- **Distinguish between the three types of output**:
  - An **error** (Error in...): The code stops, and you see it immediately.
  - A **warning** (Warning message:): The code has finished, but R reports a problem. Do **not ignore it**. This is often where deprecated functions, silently introduced NaNs, or questionable type conversions are hidden.
  - A **message** (plain text without a prefix): generally informative, but sometimes important ("Joining with by = ..." may reveal a join performed on the wrong column).

3.2 Check 2: Does the result have the correct form?

> **Tip**
>
> If you see a warning and don't understand it, paste it into the LLM and ask, "What does this warning mean, and do I need to fix it?"

journal officiel du Registre de Dialyse Péritonéale de Langue Française RDPLF www.rdplf.org

www.bdd.rdplf.org Volume 9, n° 2, June 2026
https://doi.org/10.25796/bdd.v9i2.87112
ISSN 2607-9917

Three quick checks:

• **dim()** or **nrow()**: How many rows and columns? Does this number make sense? If you were expecting a summary by group with three groups and you get 320 rows, something is wrong.

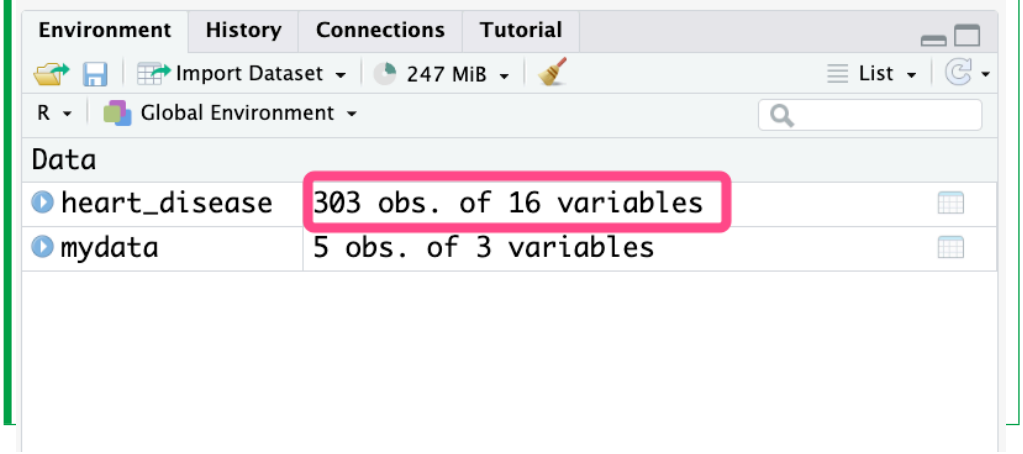
• **head()**: What do the first few rows look like? Are the columns what you expected?

• **str()** or **glimpse()**: Are the data types correct? A date stored as a character, a factor stored as numeric, a variable that's supposed to be an integer but is a double with strange decimals—these are all signs that something went wrong earlier in the process.

3.3 Check 3: Are the values plausible?

Tip

The dimensions are displayed directly in the Environment tab of RStudio, to the right of the object's name.



The code runs, and the result has the right format. But are the numbers themselves credible? This is where your domain knowledge becomes essential—and it's something an LLM lacks.

Some questions to ask yourself:

• **Are the orders of magnitude correct?** An average systolic blood pressure of 1,200 or 12 mmHg is suspicious. An average age of 450 years is too.

• **Do the summed numbers match the expected total?** If you group 320 patients by gender and get 180 + 90 = 270, 50 people are missing somewhere (likely unmanaged NA cases).

• **Are there any absurd values:** negatives where they're impossible, percentages over 100, dates in 1900 or 2087?

• **Are there any unexpected NA values?**

Two tools are very useful for answering these questions.

The first is the summary() function. Applied to an entire data frame, it gives you the minimum, maximum, median, mean, quartiles, and number of NAs for each numeric column.

For categorical columns, it gives you the counts per category. With a single command, you can see outliers, skewed distributions, and missing values at a glance.

If you just want to count the NAs column by column, colSums(is.na(data)) does the job more

journal officiel du Registre de Dialyse Péritonéale de Langue Française RDPLF www.rdplf.org

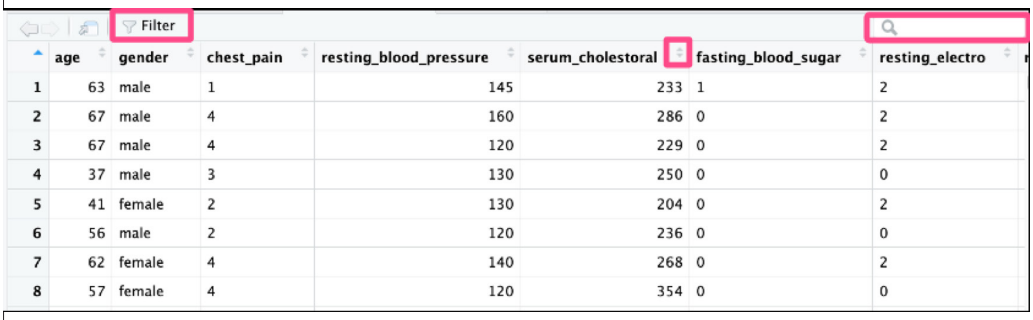
www.bdd.rdplf.org Volume 9, n° 2, June 2026
https://doi.org/10.25796/bdd.v9i2.87112
ISSN 2607-9917

directly.

```
> summary(heart_disease)
      age            gender    chest_pain resting_blood_pressure serum_cholestoral
 Min.   :29.00   female: 97   1: 23      Min.   : 94.0          Min.   :126.0
 1st Qu.:48.00   male  :206   2: 50      1st Qu.:120.0          1st Qu.:211.0
 Median :56.00                3: 86      Median :130.0          Median :241.0
 Mean   :54.44                4:144      Mean   :131.7          Mean   :246.7
 3rd Qu.:61.00                           3rd Qu.:140.0          3rd Qu.:275.0
 Max.   :77.00                           Max.   :200.0          Max.   :564.0

 fasting_blood_sugar resting_electro max_heart_rate   exer_angina        oldpeak
 0:258               0:151           Min.   : 71.0   Min.   :0.0000   Min.   :0.00
 1: 45               1:  4           1st Qu.:133.5   1st Qu.:0.0000   1st Qu.:0.00
                     2:148           Median :153.0   Median :0.0000   Median :0.80
                                     Mean   :149.6   Mean   :0.3267   Mean   :1.04
                                     3rd Qu.:166.0   3rd Qu.:1.0000   3rd Qu.:1.60
                                     Max.   :202.0   Max.   :1.0000   Max.   :6.20
```

The second is the RStudio viewer. Click on the data frame name in the Environment tab (top right), or type View(data) in the console. The table opens in a tab, where you can click on the header of any column to sort the values in ascending or descending order.
This is very handy for spotting, in just two clicks, an absurd maximum value, a negative number that shouldn't exist, or a birthdate in 1850. You can also use the small search bar at the top right of the viewer to quickly filter rows containing a specific value.
If something seems off, don't validate it. Go back to the code, or ask the LLM to explain it line



	age	gender	chest_pain	resting_blood_pressure	serum_cholestoral	fasting_blood_sugar	resting_electro
1	63	male	1	145	233	1	2
2	67	male	4	160	286	0	2
3	67	male	4	120	229	0	2
4	37	male	3	130	250	0	0
5	41	female	2	130	204	0	2
6	56	male	2	120	236	0	0
7	62	female	4	140	268	0	2
8	57	female	4	120	354	0	0

by line.

3.4 Verification 4: Testing on a known mini-example

Rather than blindly trusting the result across your 320 lines, apply the code to a mini dataset where you know the answer in advance.

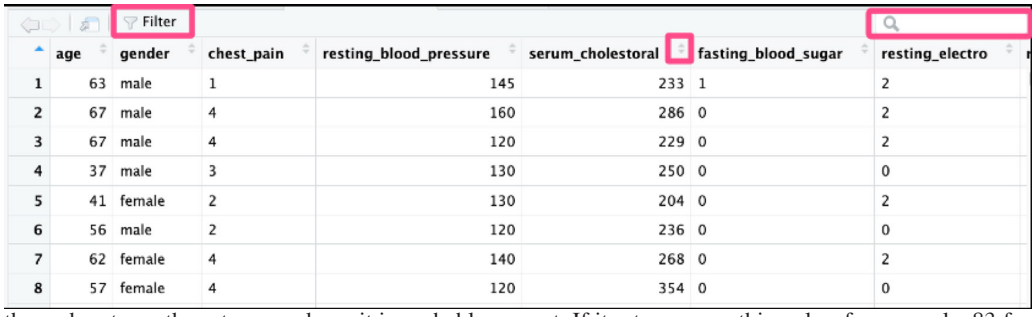
Example:
The LLM has written code for you to calculate the average voltage by gender with NA handling. Before applying it to your real data, create a dummy dataset:

```
test <- data.frame(
  sexe = c(«F», «F», «F», «M», «M», «M»),
  tension = c(120, 130, NA, 140, 150, 160)
)
```

You know in advance that the expected average is 125 for women (ignoring the NA) and 150 for men. If

journal officiel du Registre de Dialyse Péritonéale de Langue Française RDPLF www.rdplf.org

www.bdd.rdplf.org Volume 9, n° 2, June 2026
https://doi.org/10.25796/bdd.v9i2.87112
ISSN 2607-9917



Filter

	age	gender	chest_pain	resting_blood_pressure	serum_cholestoral	fasting_blood_sugar	resting_electro
1	63	male	1	145	233	1	2
2	67	male	4	160	286	0	2
3	67	male	4	120	229	0	2
4	37	male	3	130	250	0	0
5	41	female	2	130	204	0	2
6	56	male	2	120	236	0	0
7	62	female	4	140	268	0	2
8	57	female	4	120	354	0	0

the code returns these two numbers, it is probably correct. If it returns something else, for example, 83 for women (because the NA was counted as 0), you have found a bug before it affects your actual analysis.

3.4.1 The overall check

Let's recap the checks to perform:

1. Does the code run without errors or warnings?
2. Does the result have the correct dimensions and data types?
3. Are the values plausible?
4. Does the test on a known mini-example give the expected result?

If even one of these four checks fails, do not approve the code. Go back to it with the LLM, ask it for explanations, or check the official documentation (we'll discuss this in Section 4).

4. When to Use an LLM, When to Prefer the R Documentation, and When to Ask a Human?

The LLM is a fantastic tool, but it's not the only one, and it's not always the best.
Many beginners fall into the opposite trap from what we did a few years ago: Instead of searching for everything on Google, they ask ChatGPT about everything, including topics for which other resources are faster, more reliable, or more educational.
Here's a simple guide to help you choose the right channel based on your question.

4.1 Use an LLM when…

• You know what you want to do, but not how to write it in R: "I want to convert my table from wide format to long format, by date variable." The LLM produces a suitable pivot_longer() in seconds.

• You want to explore several approaches: "Give me three different ways to visualize the distribution of a numeric variable." The LLM is an excellent generator of alternatives.

• You need to decode an error message. Paste the error and the code, and ask for an explanation. It's often faster than searching on Stack Overflow.

• You want to translate code from one "dialect" to another. Switch from base R to the tidyverse, or vice versa.

• You want to quickly generate repetitive code. A loop that produces 20 similar plots, a script that renames 40 columns according to a pattern—the LLM saves you a ton of time. Note that GitHub Copilot is also a good option in this situation.

• You want a comment or an explanatory note on a block of code you've written to verify your understanding.

journal officiel du Registre de Dialyse Péritonéale de Langue Française RDPLF www.rdplf.org

www.bdd.rdplf.org Volume 9, n° 2, June 2026
https://doi.org/10.25796/bdd.v9i2.87112
ISSN 2607-9917

4.2 Choose R documentation when…

The official documentation remains irreplaceable, and too few beginners consult it. Here's when to prioritize it:

• You want to understand exactly what a function does. Type "?function_name" in the console: You'll get the exact list of arguments, their default values, the return type, and often reproducible examples at the bottom of the page. It's the definitive source.

• The LLM has given you working code, but you want to know if a different option might be better. The function's help lists all the arguments, whereas the LLM often only mentions the most common ones.

• You are working with a specialised package (epidemiology, time series, geostatistics, etc.). Many of these packages have vignettes (detailed instructional documents) accessible via `browseVignettes('package_name')`. These are written by the authors, are up to date, and are often worth ten prompts.

• You want to check for potentially obsolete syntax (see Chapter 1). The documentation displays deprecation warnings and indicates the replacement function.

• You are looking for a comprehensive list of a package's functions. Type help(package = 'package_name') or visit the package's website (often a very comprehensive pkgdown).

4.3 Ask a human when…

I recommend consulting a competent professional (colleague, supervisor, statistician, specialist forum) in the following cases:

• Choosing a statistical method for your specific problem. Which test? Which model? Which correction for multiple comparisons? The LLM will give you a plausible answer, but the right choice depends on your study design, the nature of your variables, the assumptions you can or cannot verify, and the actual scientific question. This is exactly the kind of decision where bad advice goes unnoticed — and where the consequences can be significant (see section 1, Error No. 3).

• Interpreting your results. An LLM can describe what a p-value generally means. It cannot tell you whether, in your context, your result is robust, biased, or trivial (in any case, you are taking a risk).

• Any question involving sensitive data that you cannot share with an LLM (see section 2). A human subject to the same confidentiality obligations (colleague, supervisor) can view the actual data — an LLM cannot.

• Key methodological decisions: experimental design, sample size, analysis plan, choice of protocol. These decisions shape your entire study. They warrant a proper discussion with someone who knows your field.

• When you have a hunch that something is amiss, but cannot put your finger on what it is. This intuition is important, and it is best discussed with a human who can ask the right questions in return.

5. Conclusion

There you have it—you now have the four essential best practices for turning ChatGPT or Claude into a true R assistant, rather than just a code generator that you copy and paste blindly:

1. Be aware of the three typical LLM errors: hallucinated functions, obsolete syntax, and, above all, bad statistical advice.

2. Structure your prompts into three blocks: context, objective, and expected format, without ever

journal officiel du Registre de Dialyse Péritonéale de Langue Française RDPLF www.rdplf.org

pasting sensitive data.

3. Systematically verify that the code produces the correct result, not just that it runs without errors.

4. Choose the right channel based on your question: LLM for “how-to” questions, R documentation for precision, and a human for methodological decisions.

These habits will become second nature after a few weeks of practice. At that point, you’ll be in the ideal situation: comfortable enough with R to distinguish a good answer from a bad one, and methodical enough with LLMs to use them without getting lost.`

LLMs don’t replace learning R; they accelerate it, provided you stay in control of what you’re doing. That’s exactly the spirit in which I designed my courses: to make you self-reliant, not dependent on a tool.

Disclosures

Author: Claire Della-Vedova designed and wrote this article entirely on her own.
Ethical statement and patient consent: Not applicable.
Funding: The author received no funding for this article.
Declaration of interest: The author works as a freelance consultant specialising in clinical methodology, biostatistics and R expertise. She also delivers professional training courses on statistics and the R software.

ORCID iDs
Claire Della-Vedova : https://orcid.org/0000-0002-1956-6579

Références

1. Della Vedova C. Introduction à l’analyse de données avec le logiciel R. Bull Dial Domic [Internet]. 16 juin 2019 [cité 27 mai 2026];2(2):75-88. doi: https://www.doi.org/10.25796/bdd.v2i2.20513
2. Della Vedova C. Introduction à la data visualisation sous R, avec l’add in Esquisse: formation à l’utilisation du logiciel statistique R. Bull Dial Domic [Internet]. 17 août 2019 [cité 27 mai 2026];2(3):165-74. doi: https://doi.org/10.25796/bdd.v2i3.21313
3. Della Vedova C. Réaliser des tableaux de bord avec le logiciel R. Bull Dial Domic [Internet]. 7 sept. 2020 [cité 27 mai 2026];3(3):177-90. doi: https://doi.org/10.25796/bdd.v3i3.58203
4. Della Vedova C. Initiation au Logiciel de statistiques R : réalisez vos premières visualisations avec le package ggplot2. Bull Dial Domic [Internet]. 15 déc. 2019 [cité 27 mai 2026];2(4):229-38. doi: https://doi.org/10.25796/bdd.v2i4.52303
5. Della Vedova C. Initiation au Logiciel de statistiques R : Initiation à Rmarkdown. Bull Dial Domic [Internet]. 13 avr. 2020 [cité 27 mai 2026];3(1):51-66. doi: https://doi.org/10.25796/bdd.v3i1.54523
6. Della Vedova C. Initiation à la manipulation de données avec le package dplyr. Bull Dial Domic [Internet]. 15 juin 2020 [cité 27 mai 2026];3(2):93-109. doi: https://doi.org/10.25796/bdd.v3i2.55463

APPENDIX I

2-HOUR 15-MINUTE VIDEO TRAINING accessible and designed for self-learners:

journal officiel du Registre de Dialyse Péritonéale de Langue Française **RDPLF** www.rdplf.org

www.bdd.rdplf.org Volume 9, n° 2, June 2026
https://doi.org/10.25796/bdd.v9i2.87112
ISSN 2607-9917

Language: French

Specifically, you will learn to:

- Work efficiently in RStudio, with well-organized projects
- Master the basics of the R language
- Import your own data without any issues (CSV/Excel files)
- Filter, transform, and summarize a dataset with dplyr
- Explore your data with clear summary tables
- Produce clear and polished graphs

And that's not all—the course also includes:

- ready-to-use cheat sheets (including a mind map of the modules)
- the complete training script
- datasets for practice
- transcripts of the videos
- a mini-guide to best practices for turning ChatGPT or Claude into a real R assistant

Check out the course and click here

Use Discount code (39€ instead of 49€) for readers of this article: LECTEURBDD

*journal officiel du **R**egistre de **D**ialyse **P**éritonéale de **L**angue **F**rançaise* **RDPLF** www.rdplf.org