ISSN 2607-9917 www.bdd.rdplf.org Volume 9, n° 2, Juin 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

(Edition française) (English version available at same address)

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

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

Claire Della Vedova
Consultante en biostatistiques et méthodologie clinique, expertise R
Manosque (France)

Pour citer : 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

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.

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.

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

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

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

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

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

Note de l'éditeur :
Claire Della Vedova est Ingénieure en biostatistique / data analyste, Elle utilise quotidiennement le logiciel R pour analyser des données. Elle a travaillé pendant plus de 15 ans dans les domaines de l'environnement et de la santé, et a formé de nombreux étudiants et chercheurs à l'utilisation de R. Elle anime depuis novembre 2017 le blog Statistique et Logiciel R dont le but est d'aider les débutants à mieux appréhender les méthodes statistiques classiques et à utiliser le logiciel R plus efficacement, notamment au travers de tutoriels : https://delladata.fr/blog. Il y a quelques années elle avait rédigé plusieurs articles d'initiation à ce logiciel dans le BDD [1-6].

Le logicel R est en libre téléchargement à l'adresse : https://www.r-project.org

---

On me pose de plus en plus souvent la question : « Maintenant que ChatGPT écrit du code R, est-il encore utile d'apprendre R ? »
Ma réponse est oui ! Et même plus que jamais !

L'IA amplifie une compétence, elle ne la crée pas. Sans bases, impossible de vérifier ce qu'elle produit, de corriger une erreur ou d'adapter son code à vos données. Pour bien utiliser ChatGPT ou Claude avec R, il faut d'abord en maîtriser les fondamentaux. C'est pour poser ce socle que j'ai créé ma toute première formation vidéo, accessible et pensée pour celles et ceux qui se forment par eux-mêmes : Bien démarrer avec R et RStudio – l'essentiel en 2h15.(*voir Annexe I*)

1. Les 3 erreurs typiques des LLM en R

Les LLM (ChatGPT, Claude, Copilot…) sont entraînés sur des millions de lignes de code R, mais produisent encore régulièrement des erreurs qu'un débutant ne voit pas. Voici les trois familles à connaître.

1.1 Erreur n°1 : La fonction qui n'existe pas

Le LLM invente une fonction qui aurait l'air plausible mais qui n'existe pas, ou il attribue une fonction au mauvais package.

Exemple de question posée à ChatGPT :

"Comment calculer un score Z standardisé sur ma colonne age ?"

Réponse fournie :

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

Le problème :
La fonction standardize() n'existe pas dans le tidyverse.
Il faut soit utiliser scale(age) (base R), soit (age - mean(age)) / sd(age).

Comment le repérer ?
Le code lance une erreur could not find function «standardize» :

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

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

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

Si vous tapez le nom d'une fonction dans la console RStudio (précédé d'un ?) , par exemple : ?standardize et que la doc ne trouve rien, c'est très probablement une hallucination !

1.2 Erreur n°2 : La syntaxe obsolète

Le LLM propose une syntaxe qui marchait il y a 2 ou 3 ans, mais qui est dépréciée ou remplacée.
Exemples typiques :

- Suggérer summarise_each(funs(mean)) au lieu de summarise(across(everything(), mean)) (la première forme est obsolète depuis dplyr 1.0)
- Utiliser aggregate(data, by=list(group), FUN=mean) quand group_by() %>% summarise() est beaucoup plus lisible et performant
- Mélanger des packages qui ne sont plus maintenus (ex : plyr alors que dplyr est le standard depuis longtemps)

Comment le repérer ?

Le code marche mais RStudio affiche un message d'avertissement type "summarise_each() was deprecated in dplyr 1.0.0".
Faites attentions aux warnings retournés par R et affichés dans la console, ils sont là pour ça !

Règle de réflexe :

Si un warning de dépréciation s'affiche, demandez au LLM "donne-moi la version actuelle, non dépréciée".

1.3 Erreur n°3 : Le mauvais conseil statistique

C'est l'erreur la plus dangereuse, parce que le code marche, mais l'analyse est fausse. Le LLM propose un test inadapté, oublie de vérifier les conditions d'application, ou interprète les résultats avec trop d'aplomb.

Exemples typiques :

- Proposer un t-test sur des données qui ne sont manifestement pas normales, sans rien vérifier
- Utiliser un Chi^2 alors que les effectifs attendus sont inférieurs à 5 (le bon choix serait un test exact de Fisher)
- Comparer 4 groupes deux à deux avec des t-tests sans corriger pour la multiplicité des tests
- Interpréter un p-value < 0,05 comme "preuve d'un effet" sans tenir compte de la puissance ni de la taille d'effet

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

Comment le repérer :

Aucune erreur, ni warning n'est retourné ! C'est pour ça qu'il faut être vigilant. Demandez systématiquement au LLM : "Quelles sont les conditions d'application de ce test ? Comment les vérifier ?"

Règle de réflexe :

Le code ne suffit pas. Avant d'appliquer un test, comprenez pourquoi c'est ce test-là, pas un autre. Si vous n'êtes pas sûr(e), c'est le moment de chercher un avis humain compétent !

2. La méthode de prompting en 3 étapes

La qualité du code que vous renvoie un LLM dépend à 80 % de la qualité de votre question. Un prompt bâclé donne un code générique, mal adapté à vos données, qu'il faudra retravailler trois fois. Un prompt structuré donne un code souvent utilisable du premier coup.

La méthode tient en trois blocs : contexte + objectif + format attendu.

2.1 Étape 1: Le contexte : décrivez vos données et votre environnement

Le LLM ne voit pas votre écran. Il ne connaît ni la structure de votre tableau, ni les packages que vous utilisez, ni la version de R que vous avez installée. Si vous ne lui dites rien, il va deviner, et pas toujours bien.

Ce qu'il faut donner :

- **La nature du jeu de données** : combien de lignes, combien de colonnes, quel type de données (numériques, catégorielles, dates…)
- **Les noms exacts des colonnes** concernées par votre question
- **Le ou les packages que vous voulez utiliser** (tidyverse ? base R ? ggplot2 ?)
- **La structure des données**, pas les données elles-mêmes (on y vient juste après)

> Confidentialité : ne collez pas vos vraies données dans le LLM
>
> C'est le réflexe à intégrer dès le départ. Si vous travaillez sur des données médicales, RH, financières, scolaires, des données clients ou tout autre contenu sensible, vous n'avez pas le droit (ni légalement, ni déontologiquement) de les envoyer à ChatGPT, Claude ou Copilot. Ces données partent sur des serveurs tiers et peuvent être utilisées pour entraîner de futurs modèles. Même quand ce n'est pas illégal, c'est une mauvaise pratique.

Ce qu'il faut faire à la place :

- Décrire la structure : "j'ai une colonne age numérique, une colonne diagnostic à 4 catégories, et une colonne date_inclusion au format Date"
- Coller la sortie de str(data) ou glimpse(data) : elle donne les types et quelques exemples, sans révéler tout le contenu
- Fabriquer un mini jeu de données fictif qui ressemble au vôtre, avec 5 lignes inventées qui ont la même structure
- Anonymiser si vous devez absolument montrer un extrait (remplacer les noms, brouiller les identifiants, arrondir les valeurs sensibles)

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

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

Exemple de contexte bien posé, sans fuite de données :

"J'ai un data frame patients de 320 lignes et 8 colonnes, importé avec read_csv(). Les colonnes pertinentes sont age (numérique, entre 18 et 92), sexe (facteur à 2 niveaux : F/M) et tension_systolique (numérique, avec environ 5 % de NA). Je travaille avec le tidyverse.
Le LLM a tout ce qu'il lui faut pour vous écrire un code correct, sans avoir vu une seule ligne réelle de votre tableau.

2.2 Étape 2: L'objectif : dites ce que vous voulez obtenir, pas comment

Quand on débute, on a tendance à écrire nos prompts comme une demande technique :
"Comment faire un group_by() puis un summarise() avec mean() ?"
Ce n'est pas une bonne approche parce que vous imposez une solution au LLM avant même qu'il ait compris votre besoin. Et si votre solution n'est pas la bonne, il va vous la coder quand même !
L'objectif doit davantage décrire le résultat attendu pour laisser le LLM choisir la fonction.

Evitez :
"Fais-moi un mutate() avec case_when() pour créer une variable âge_catégorie."

Mais préférez:
"Je veux créer une nouvelle colonne qui classe les patients en trois tranches d'âge : moins de 40 ans, 40 à 65 ans, plus de 65 ans."

La deuxième formulation laisse le LLM proposer la solution la plus propre. La première l'enferme dans un choix qui n'est peut-être pas optimal.

2.3 Étape 3 : Le format attendu : précisez comment vous voulez la réponse

C'est une étape souvent oubliée, mais qui vous fera gagner du temps. Si vous ne précisez rien, le LLM va vous répondre avec un long pavé d'explications, des alternatives, des "il existe aussi…", et il faudra fouiller pour trouver le code utile.

Ce que vous pouvez demander :

- Du code seul, sans explications ("réponds uniquement avec le bloc de code, sans commentaires")
- Du code commenté ligne par ligne si vous voulez apprendre
- Plusieurs versions ("propose-moi 2 façons de faire : une en base R, une en tidyverse")
- Un format de sortie précis ("le résultat doit être un data frame avec exactement 3 colonnes : groupe, moyenne, écart-type")

Exemple :
"Réponds avec un seul bloc de code tidyverse, commenté en français, qui produit un data frame trié par moyenne décroissante."
Le prompt assemblé:
Voici à quoi ressemble un prompt complet qui suit la méthode :
Contexte : J'ai un data frame patients de 320 lignes et 8 colonnes, avec les variables age (numérique), sexe (facteur F/M) et tension_systolique (numérique, quelques NA). Je travaille avec

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

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

le tidyverse.
Objectif : Je veux créer une nouvelle colonne age_categorie qui classe les patients en trois tranches : moins de 40 ans, 40 à 65 ans, et plus de 65 ans.
Format attendu : Un seul bloc de code tidyverse, commenté en français, qui ajoute cette nouvelle colonne au data frame existant.

2.3.1 Astuce

Dictez vos prompts à la voix

> **Dictez vos prompts à la voix**
>
> Écrire un prompt structuré peut être long, mais vous pouvez : **utiliser la dictée vocale**. Sur smartphone, l'icône micro du clavier fonctionne très bien. Sur ordinateur, ChatGPT et Claude ont un bouton micro intégré.

L'intérêt n'est pas seulement la vitesse (vous dictez 3 à 4 fois plus vite que vous ne tapez). C'est aussi que **vous formulez plus naturellement**. À l'oral, vous expliquez spontanément le contexte, ce que vous voulez, et comment vous le voulez. C'est exactement la structure en trois blocs, mais sans effort conscient.
Vous pouvez parfaitement dire : "*Alors voilà, j'ai un tableau de 320 patients avec des colonnes âge, sexe et tension, je voudrais créer une nouvelle colonne qui classe les patients en trois tranches d'âge — moins de 40 ans, 40 à 65 ans, plus de 65 ans — et donne-moi juste le code tidyverse commenté en français*." Le LLM comprend très bien ce langage parlé, et vous obtiendrez une réponse aussi précise qu'avec un prompt écrit soigné.

2.3.2 Pourquoi ça marche ?

Cette structure force le LLM à se concentrer sur votre problème, pas sur le problème générique qu'il croit reconnaître. Elle réduit les hallucinations (vues au chapitre 1), parce qu'en donnant le contexte vous lui dites implicitement "ne me sors pas une fonction qui ne s'applique pas à ces données".
Avec un peu de pratique, dicter un prompt structuré prend 20 secondes, et vous économise 10 minutes de débogage.
Règle de réflexe : avant d'envoyer votre prompt, relisez-le et vérifiez qu'il contient bien les trois blocs. Si l'un manque, complétez-le. Cette habitude va devenir automatique en quelques jours.

3. Comment vérifier que le code généré marche vraiment ?

Le LLM vous a renvoyé un beau bloc de code. Il a l'air propre, bien indenté, avec des commentaires rassurants. La tentation est de le copier-coller dans votre script et de passer à la suite.
C'est exactement à ce moment-là que les ennuis peuvent commencer…..
Un code peut s'exécuter sans erreur et produire un résultat complètement faux. La vérification n'est donc pas un luxe ! Voici les quatre vérifications à enchaîner.

3.1 Vérification 1 : Est ce quelLe code s'exécute sans erreur ni warning ?

- **Exécutez le bloc comple**t, pas ligne par ligne. Certaines erreurs n'apparaissent qu'en contexte.

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

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

• **Lisez la console** : pas seulement la dernière ligne, mais aussi les messages intermédiaires.

• Distinguez les trois types de retours :

- Une **erreur** (Error in...) : le code s'arrête, vous le voyez tout de suite.

- Un **warning** (Warning message:) : le code a fini, mais R signale un problème. Ne l'ignorez pas. C'est souvent là que se cachent les fonctions dépréciées, les NA introduits silencieusement, ou les conversions de types douteuses.

- Un **message** (texte simple sans préfixe) : généralement informatif, mais parfois important ("Joining with by = ..." peut révéler une jointure faite sur la mauvaise colonne).

Astuce

Si vous voyez un warning et que vous ne le comprenez pas, recollez-le au LLM en demandant : "que signifie ce warning, et faut-il que je le corrige ?"

3.2 Vérification 2 : Est ce que le résultat a la bonne forme ?

Trois réflexes :

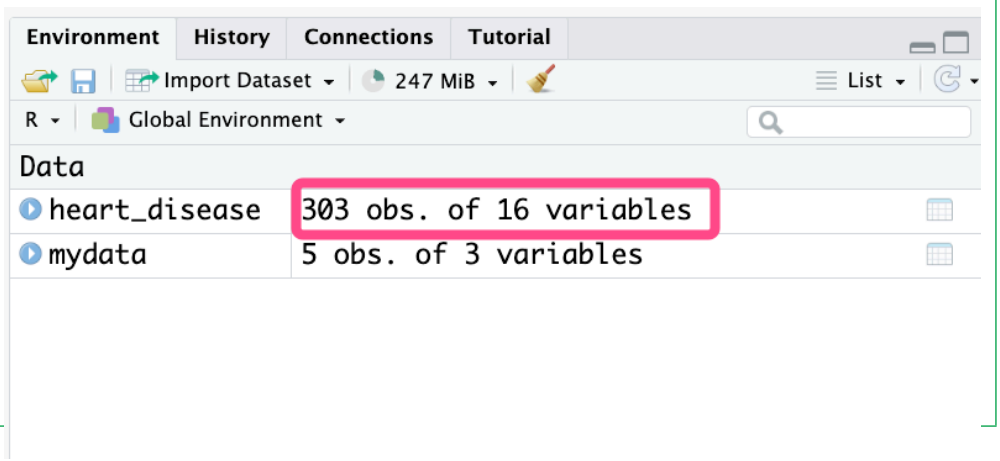
• **dim()** ou **nrow()** : combien de lignes et de colonnes ? Est-ce que ce nombre a du sens ? Si vous attendiez un résumé par groupe avec 3 groupes et que vous obtenez 320 lignes, quelque chose ne va pas.

• **head()** : à quoi ressemblent les premières lignes ? Les colonnes sont-elles bien celles que vous attendiez ?

• **str()** ou **glimpse()** : les types sont-ils corrects ? Une date stockée en character, un facteur stocké en numeric, une variable censée être un entier en double avec des décimales bizarres ; autant de signaux que quelque chose s'est mal passé en amont.

Astuce

Les dimensions s'affichent directement dans l'onglet Environment de RStudio, à droite du nom de l'objet



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

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

3.3 Vérification 3 : Les valeurs sont-elles plausibles ?

Le code tourne, le résultat a la bonne forme. Mais les chiffres eux-mêmes sont-ils crédibles ? C'est ici que votre **connaissance du domaine** devient indispensable, et c'est ce qu'un LLM n'a pas.
Quelques questions à vous poser :

- **Les ordres de grandeur sont-ils corrects ?** Une tension systolique moyenne à 1200 mmHg ou à 12, c'est suspect. Une moyenne d'âge à 450 ans aussi.
- **Les effectifs additionnés correspondent-ils au total attendu ?** Si vous regroupez 320 patients par sexe et que vous obtenez 180 + 90 = 270, il manque 50 personnes quelque part (probablement des NA non gérés).
- **Y a-t-il des valeurs absurdes** : des négatifs là où c'est impossible, des pourcentages au-dessus de 100, des dates en 1900 ou en 2087 ?
- **Y a-t-il des NA inattendus ?**

Deux outils sont de très bon alliés pour répondre à ces questions.

Le premier, c'est la fonction summary(). Appliquée à un data frame entier, elle vous donne pour chaque colonne numérique le minimum, le maximum, la médiane, la moyenne, les quartiles, et le nombre de NA.
Pour les colonnes facteur, elle vous donne les effectifs par modalité. En une seule commande,

```
> 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
```

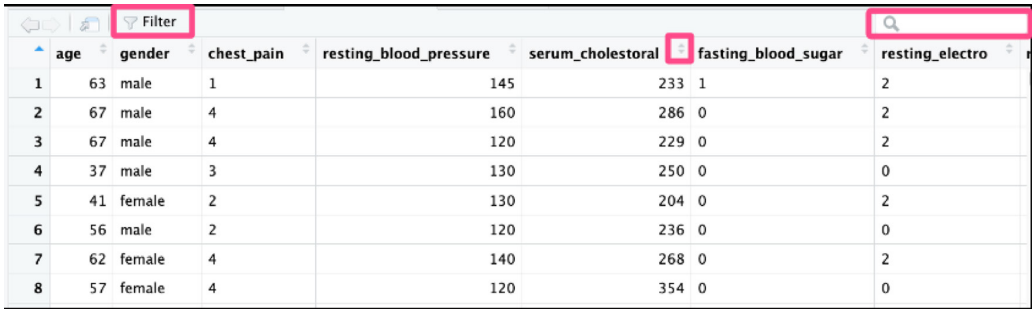
vous voyez d'un coup d'œil les bornes aberrantes, les distributions déséquilibrées et les valeurs manquantes.

Si vous voulez uniquement compter les NA colonne par colonne, colSums(is.na(data)) fait le travail plus directement.

Le second, c'est la visionneuse de RStudio. Cliquez sur le nom du data frame dans l'onglet Environment (en haut à droite), ou tapez View(data) dans la console. Le tableau s'ouvre dans un onglet, et là vous pouvez cliquer sur l'en-tête de n'importe quelle colonne pour trier les valeurs par ordre croissant ou décroissant.

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

www.bdd.rdplf.org Volume 9, n° 2, Juin 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

C'est très pratique pour repérer en deux clics la valeur maximale absurde, le négatif qui ne devrait pas exister, ou la date de naissance en 1850. Vous pouvez aussi utiliser la petite barre de recherche en haut à droite du viewer pour filtrer rapidement les lignes qui contiennent une valeur précise.

Si quelque chose vous semble bizarre, ne validez pas. Reprenez le code, ou demandez au LLM de l'expliquer ligne par ligne.

3.4 Vérification 4 : Le test sur un mini-exemple connu

Plutôt que de faire confiance aveuglément au résultat sur vos 320 lignes, appliquez le code à un mini jeu de données dont vous connaissez la réponse à l'avance.
Exemple : le LLM vous a écrit un code pour calculer la moyenne de tension par sexe avec gestion des NA. Avant de l'appliquer à vos vraies données, créez un faux jeu :

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

Vous savez d'avance que la moyenne attendue est 125 pour les femmes (en ignorant le NA) et 150 pour les hommes. Si le code renvoie ces deux chiffres, il est probablement correct. S'il renvoie autre chose, par exemple 83 pour les femmes (parce que le NA a été compté comme 0), vous avez trouvé un bug avant qu'il ne contamine votre vraie analyse.

3.4.1 Le réflexe global

Récapitulons les vérifications à faire:

1. **Est ce que le code tourne sans erreur ni warning ?**
2. **Est ce que le résultat a la bonne dimension et les bons types ?**
3. **Les valeurs sont-elles plausibles ?**
4. **Le test sur un mini-exemple connu donne-t-il la réponse attendue ?**

Si une seule de ces quatre vérifications échoue, ne validez pas le code. Reprenez-le avec le LLM, demandez-lui des explications, ou allez vérifier dans la documentation officielle (on en parle au paragraphe 4).

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

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

4. Quand utiliser un LLM, quand préférer la doc R, quand demander à un humain ?

Le LLM est un outil formidable, mais ce n'est pas le seul, et ce n'est pas toujours le meilleur.

Beaucoup de débutants tombent dans le piège inverse de ce que nous faisons il y a quelques années : au lieu de tout chercher sur Google, ils demandent tout à ChatGPT, y compris des choses pour lesquelles d'autres ressources sont plus rapides, plus fiables, ou plus formatrices.
Voici une grille simple pour vous aider à choisir le bon canal en fonction de votre question.

4.1 Utilisation d' un LLM

• Vous savez ce que vous voulez faire, mais pas comment l'écrire en R. "Je veux passer mon tableau du format large au format long, par variable de date." Le LLM produit un pivot_longer() adapté en quelques secondes.

• Vous voulez explorer plusieurs approches. "Donne-moi trois façons différentes de visualiser la distribution d'une variable numérique." Le LLM est un excellent générateur d'alternatives.

• Vous devez décoder un message d'erreur . Collez l'erreur et le code, demandez l'explication. C'est souvent plus rapide que de chercher sur Stack Overflow.

• Vous voulez traduire du code d'un "dialecte" à un autre. Passer de base R au tidyverse, ou inversement.

• Vous voulez générer du code répétitif rapidement. Une boucle qui produit 20 graphiques similaires, un script qui renomme 40 colonnes selon un motif : le LLM vous fait gagner un temps fou. A noter que Github copilot est aussi une bonne option dans cette situation.

• Vous voulez un commentaire ou une explication pédagogique d'un bloc de code que vous avez écrit, pour vérifier votre compréhension

4.2 Préférez la documentation R quand…

La doc officielle reste irremplaçable, et trop peu de débutants la consultent. Voici quand y aller en priorité :

• Vous voulez comprendre précisément ce que fait une fonction. Tapez ?nom_de_la_fonction dans la console : vous obtenez la liste exacte des arguments, leurs valeurs par défaut, le type de retour, et souvent des exemples reproductibles en bas de page. C'est la source de vérité.

• Le LLM vous a donné un code qui marche, mais vous voulez savoir si une option pourrait mieux convenir. L'aide de la fonction liste tous les arguments, alors que le LLM ne mentionne souvent que les plus courants.

• Vous travaillez avec un package spécialisé (épidémiologie, séries temporelles, géostatistique, etc.). Beaucoup de ces packages ont des vignettes (longs documents pédagogiques) accessibles par browseVignettes(«nom_du_package»). Elles sont écrites par les auteurs, sont à jour, et valent souvent dix prompts.

• Vous voulez vérifier une syntaxe potentiellement obsolète (voir chapitre 1). La doc affiche les avertissements de dépréciation et indique la fonction de remplacement.

• Vous cherchez la liste exhaustive des fonctions d'un package. Tapez help(package = «nom_du_package») ou consultez le site du package (souvent un pkgdown très complet).

4.3 Demandez à un humain quand…

Je vous conseille d'aller voir un humain compétent (collègue, encadrant, statisticien, forum spécialisé) dans les cas suivants :

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

• Le choix d'une méthode statistique pour votre problème spécifique. Quel test ? Quel modèle ? Quelle correction pour les comparaisons multiples ? Le LLM va vous donner une réponse plausible, mais le bon choix dépend de votre design d'étude, de la nature de vos variables, des hypothèses que vous pouvez ou non vérifier, et de la question scientifique réelle. C'est exactement le genre de décision où un mauvais conseil ne se voit pas — et où les conséquences peuvent être non négligeables (voir chapitre 1, erreur n°3).

• L'interprétation de vos résultats. Un LLM peut décrire ce que veut dire un p-value en général. Il ne peut pas vous dire si, dans votre contexte, votre résultat est solide, biaisé, ou trivial (en tout cas vous prenez un risque).

• Toute question impliquant des données sensibles que vous ne pouvez pas partager avec un LLM (voir chapitre 2). Un humain soumis à la même obligation de confidentialité (collègue, encadrant) peut voir les vraies données — un LLM, non.

• Les décisions méthodologiques structurantes : design expérimental, taille d'échantillon, plan d'analyse, choix d'un protocole. Ces décisions engagent toute votre étude. Elles méritent une vraie discussion avec quelqu'un qui connaît votre domaine.

• Quand vous avez l'intuition que quelque chose cloche, mais sans pouvoir dire quoi. Cette intuition est importante, et elle se discute mieux avec un humain qui peut poser les bonnes questions en retour.

Conclusion

Voilà, vous avez maintenant les 4 bonnes pratiques essentielles pour faire de ChatGPT ou Claude un vrai assistant R, et non plus un générateur de code qu'on copie-colle les yeux fermés

1. **Connaître les trois erreurs typiques des LLM** : fonctions hallucinées, syntaxe obsolète, et surtout mauvais conseils statistiques.
2. **Structurer vos prompts en trois blocs** : contexte, objectif, format attendu, sans jamais coller de données sensibles.
3. **Vérifier systématiquement que le code produit un résultat correct**, pas seulement qu'il s'exécute sans erreur.
4. **Choisir le bon canal selon votre question** : LLM pour le "comment", documentation R pour la précision, humain pour les décisions méthodologiques.

Ces réflexes vont devenir automatiques en quelques semaines de pratique. À ce moment-là, vous serez dans la situation idéale : assez à l'aise avec R pour reconnaître une bonne réponse d'une mauvaise, et assez méthodique avec les LLM pour les exploiter sans vous y perdre.

Le LLM ne remplace pas l'apprentissage de R, il l'accélère, à condition de garder la main sur ce que vous faites. C'est exactement l'esprit dans lequel j'ai conçu mes formations : vous rendre autonome, pas dépendant(e) d'un outil.

Déclarations

Auteur: Claire Della-Vedova a conçu et rédigé seule cet article
Déclaration éthique et consentement patient : sans objet
Financement : l'auteur n' a bénéficié d'aucun financement pour cet article.
Déclaration d'intérêt : L'auteure exerce une activité indépendante de consultante en méthodologie clinique, biostatistique et expertise R. Elle réalise également des formations professionnelles autour des statistiques et du logiciel R.

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

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

ANNEXE I

FORMATION VIDEO 2H15 accessible et pensée pour celles et ceux qui se forment par eux-mêmes :

Langue : Français
Concrètement, vous apprendrez à :

- -Travailler proprement dans RStudio, avec des projets bien organisés
- -Maîtriser les briques de base du langage R
- -Importer vos propres données sans bloquer (CSV / fichiers Excel)
- -Filtrer, transformer et résumer un jeu de données avec dplyr
- -Explorer vos données avec des tableaux de synthèse clairs
- -Produire des graphiques lisibles et soignés

Et ce n'est pas tout, la formation inclut aussi :

- - des aide-mémoire prêts à l'emploi (dont la mind map des modules)
- - le script complet de la formation
- - les jeux de données pour vous entraîner
- - la retranscription texte des vidéos
- - un mini-guide des bonnes pratiques pour faire de ChatGPT ou Claude un vrai assistant R

Découvrir la formation en cliquant ici

Code de réduction les lecteurs de cet article bénéficient d'un code de réducton (39€ au lieu de 49 €) : sasir **LECTEURBDD** au moment de l'inscription à la formation.