Becoming a Better **R** Programmer: Practical Tips for Beginners

Dr Tobias Niedermaier (MPH)
Institute of Medical Information Processing, Biometry and Epidemiology,
Ludwig-Maximilians University (LMU) Munich

August 4, 2025

Introduction

R is a powerful, but often quirky, programming language. This handout aims to help you build solid foundations, develop good habits, and avoid common pitfalls. You do not have to master everything at once. Learn steadily, experiment often, and embrace mistakes as learning opportunities.

1 Core Concepts

- **Multiple programming paradigms.** R supports functional, object-oriented (S3/S4/R6), and vectorized styles. At its core, everything is an object and every operation a function call. Begin with what feels intuitive.

- **Avoid repetitive code (DRY).** Repetition leads to errors and makes maintenance difficult.

2 Repetition & Abstraction

- **Start with `for`-loops.** Easy to grasp and universally applicable.

- **Learn functions early.** When repeating the same steps, encapsulate them via functions. Functions help reduce duplication and facilitate debugging.

3 Performance (Later)

Do not worry about optimization at first. Focus on correct, readable code. Speed only matters when handling large datasets or compute-intensive tasks like bootstrapping or multiple imputation.

4 Writing Readable Code

- **Comment clearly.** Explain what you did, and why. Your future self will appreciate the context.

- **Be consistent with style.** Formatting, indentation, and naming conventions improve readability. A helpful guide is «Good Habits in R Programming» (STAT133) by Gaston Sanchez.

5 Using AI Tools

ChatGPT and other LLMs can assist with debugging and suggestions. But do not copy-paste-use them as a hint system. Struggle first; compare approaches later. That struggle is essential for real learning.

6 Base R vs. Tidyverse

R usage often splits into two «dialects»: **base R** and the **tidyverse**. Tidyverse is popular for data manipulation and visualization. However, base R often offers more transparency and flexibility, especially at the start. Interested? Read the critique by Prof. Norm Matloff about the tidyverse philosophy.

7 Advanced Tips (when needed)

- **Benchmark your code** to find bottlenecks before optimizing.
- **Pre-allocate vectors** rather than growing them in loops.
- **Combine multiple operations in one loop** to avoid inefficiency.
- **Parallelization:** Explore `makeCluster()` + `%dopar%` via `parallel` and `foreach`.
- **Use `data.table`** for fast, memory-efficient data manipulation.

Summary

These practices will help you build strong, maintainable R programs. Focus on fundamentals before optimization. Learn gradually and do not fear mistakes - they are your stepping stones.

Good luck, and enjoy your journey!
Tobias Niedermaier