Project Structure

Professional Organization for R Projects

From Chaos to Clarity

Organized	**Easy**	**Production**
Structure	Navigation	Ready

Dr. Saad Laouadi

AI Expert & Data Science Educator

Independent Consultant

Version 1.0 | December 13, 2025

Professional R Development Series

Contents

1 Why Structure Matters

Good project structure makes your code **findable, shareable, and maintainable**.

1.1 The Problem

You start with one script. Six months later: 47 files, no clear organization, can't find anything.
Result: Wasted time, frustrated collaborators, deployment nightmares.

1.2 The Solution

✗ Bad Structure	✓ Good Structure
Files everywhere	Clear organization
No clear workflow	Logical workflow
Hard to collaborate	Team-friendly
Can't deploy easily	Deploy-ready

1.3 Benefits of Good Structure

- **Find** anything in seconds
- **Onboard** new team members quickly
- **Deploy** without restructuring
- **Maintain** projects months/years later

2 Standard R Project Structure

2.1 The Template

```
my_project/
 R/ # Function definitions
 data/ # Input data
  raw/
  processed/
 outputs/ # Results, plots, reports
 docs/ # Documentation
 tests/ # Unit tests
 renv/ # Environment (from renv)
 .gitignore # Git ignore rules
 README.md # Project overview
 DESCRIPTION # Project metadata
 main.R # Entry point script
```

Not every project needs all folders. Start with what you need, add as you grow.

2.2 Folder Purposes

Folder	What Goes Here
`R/`	Custom functions, utilities
`data/raw/`	Original, unmodified data
`data/processed/`	Cleaned, transformed data
`outputs/`	Plots, tables, final results
`docs/`	Documentation, notes, reports
`tests/`	Unit tests (testthat)
`renv/`	Package environment

3 Key Files Explained

3.1 README.md

Purpose: First thing people read. Explains what, why, and how.

```
# My Analysis Project

## Overview
Brief description of what this project does.

## Setup
"‘r
renv::restore()
"‘

## Usage
"‘r
source("main.R")
"‘

## Structure
- R/: Custom functions
- data/: Input datasets
- outputs/: Results and plots

## Author
Your Name
```

3.2 DESCRIPTION

Purpose: Project metadata. Used by R and package managers.

```
Package: MyProject
Type: Project
Title: Analysis of Customer Behavior
Version: 0.1.0
Authors@R: person("Your", "Name",
email = "you@example.com",
role = c("aut", "cre"))
Description: Analyzes customer purchase patterns.
License: MIT
Depends: R (>= 4.0.0)
Imports:
dplyr,
ggplot2
Encoding: UTF-8
```

Enables `devtools::load_all()`, dependency tracking, and professional project management.

3.3 .gitignore

Purpose: Tells Git what NOT to track.

```
# R specific
.Rhistory
.RData
.Rproj.user/

# renv
renv/library/
renv/staging/

# Data (if sensitive or large)
data/raw/*.csv
data/processed/*.rds

# Outputs
outputs/*.png
outputs/*.pdf

# System
.DS_Store
Thumbs.db
```

Large data files, credentials, API keys, or generated outputs.

4 The R/ Folder

4.1 Purpose

Store reusable functions. Keep scripts clean by extracting logic.

4.2 Organization

```
R/
 data_cleaning.R # Data prep functions
 analysis.R # Analysis functions
 plotting.R # Visualization functions
 utils.R # Utility functions
```

4.3 Example Function File

```
# R/data_cleaning.R


#' Clean Customer Data
#'
#' @param data Raw data frame
#' @return Cleaned data frame
#' @export
clean_customer_data <- function(data) {
data %>%
filter(!is.na(customer_id)) %>%
mutate(date = as.Date(date))
}
```

4.4 Loading Functions

```
# In main.R or analysis scripts:
source("R/data_cleaning.R")
source("R/plotting.R")


# Or load all at once:
devtools::load_all(".")
```

5 Data Organization

5.1 The Two-Tier Approach

data/raw/
Original, unmodified data. Never edit these files.

```
data/raw/
 customers.csv
 transactions.xlsx
 survey_results.rds
```

data/processed/
Cleaned, transformed data. Safe to regenerate.

```
data/processed/
 clean_customers.rds
 analysis_dataset.rds
 model_data.rds
```

5.2 Best Practices

1. **Never** edit `data/raw/` files
2. Document data sources in README
3. Use `.rds` for R-specific data (preserves types)
4. Use `.csv` for interoperability
5. Consider `.gitignore` for large files

6 Workflow Organization

6.1 Script Naming Convention

```
01_download_data.R
02_clean_data.R
03_exploratory_analysis.R
04_build_model.R
05_generate_report.R
```

Numeric prefixes show execution order. Makes workflow obvious.

6.2 Main Entry Point

```
# main.R - orchestrates entire analysis

# Setup
library(dplyr)
library(ggplot2)
source("R/data_cleaning.R")
source("R/analysis.R")

# Run pipeline
source("01_download_data.R")
source("02_clean_data.R")
source("03_exploratory_analysis.R")
source("04_build_model.R")
source("05_generate_report.R")

message("Analysis complete!")
```

6.3 Makefile Alternative

For complex projects, use a workflow manager:

```
# Using targets package
library(targets)

tar_plan(
tar_target(raw_data, read_csv("data/raw/data.csv")),
tar_target(clean_data, clean(raw_data)),
tar_target(model, build_model(clean_data)),
tar_target(report, render_report(model))
)
```

7 Common Patterns

7.1 Analysis Project

Data Analysis

```
analysis_project/
 R/
 data/raw/
 data/processed/
 outputs/figures/
 outputs/tables/
 reports/
 README.md
 analysis.Rmd
```

Focus: Exploratory analysis, reports, visualizations.

7.2 Package Development

R Package

```
mypackage/
 R/
 man/
 tests/testthat/
 vignettes/
 data/
 DESCRIPTION
 NAMESPACE
 README.md
```

Focus: Reusable functions, documentation, tests.

7.3 Shiny Application

Shiny App

```
shiny_app/
 R/ # Module functions
 data/
 www/ # CSS, images, JS
 app.R # Main app file
 ui.R # OR separate UI
 server.R # OR separate server
 global.R # Shared setup
```

Focus: Interactive web application.

8 Best Practices

Golden Rule

Structure your project so someone else (or future you) can understand it in 5 minutes.

1. Start with Structure

```
# Don't wait.  Create folders first.
dir.create(c("R", "data/raw", "data/processed",
"outputs", "docs"), recursive = TRUE)
```

Build structure before writing code.

2. Separate Data and Code

- Never hard-code paths
- Use `here::here()` for portable paths
- Keep code and data separate

3. Document Everything

- README for project overview
- Comments for complex code
- Roxygen for functions
- Changelog for updates

4. Use Consistent Naming

```
# Good
clean_customer_data.R
analyze_sales_trends.R


# Bad
stuff.R
final_FINAL_v2.R
```

Use snake_case. Be descriptive.

5. Version Control from Day One

```
git init
git add .
git commit -m "Initial project structure"
```

Track changes from the start.

9 Quick Start Template

9.1 Create Project Structure

```r
# Run this in R console

# Create main directories
dirs <- c("R", "data/raw", "data/processed",
"outputs/figures", "outputs/tables",
"docs", "tests")

lapply(dirs, dir.create, recursive = TRUE)

# Create key files
file.create(c("README.md", "DESCRIPTION",
".gitignore", "main.R"))

message("Project structure created!")
```

9.2 Basic README Template

README.md

```
# Project Name

Brief one-line description.

## Installation
renv::restore()

## Usage
source("main.R")

## Project Structure
- R/: Functions
- data/: Datasets
- outputs/: Results

## Author
Your Name
```

10 Troubleshooting

Issue 1: Path Problems

```
Error: cannot open file 'data/file.csv'
```

Fix: Use `here` package

```
library(here)
data <- read_csv(here("data", "raw", "file.csv"))
```

Issue 2: Functions Not Found

```
Error: could not find function "my_function"
```

Fix: Source function files

```
source("R/my_functions.R")
# Or
devtools::load_all()
```

Issue 3: Project Too Complex

Too many files, unclear organization.
Fix: Simplify

- Group related scripts
- Use clear folder hierarchy
- Number workflow scripts
- Document in README

11 Quick Reference

Project Structure Checklist

✓ Essential Folders

- `R/` - Functions
- `data/` - Datasets
- `outputs/` - Results

✓ Essential Files

- `README.md` - Overview
- `DESCRIPTION` - Metadata
- `.gitignore` - Git rules
- `main.R` - Entry point

Quick Setup

```
# 1.  Create structure
usethis::create_project("my_project")

# 2.  Initialize renv
renv::init()

# 3.  Initialize git
usethis::use_git()

# 4.  Add folders
usethis::use_directory("data/raw")
usethis::use_directory("outputs")
```

Core Principle

Organization = Clarity = Productivity

Structure today saves hours tomorrow.