zzcollab Analysis Profile Walkthrough: A 55-Item Step-by-Step Guide

Ronald G. Thomas

2026-06-10

Table of contents

2026-06-10 18:52 PDT

Fifty-five items, one terminal, one reproducible compendium: the checklist is the method.

Overview

This post combines the 55-item analysis initiation checklist with the concrete shell commands and R code from the zzcollab quickstart guide, applied specifically to the `analysis` profile. The `analysis` profile uses `rocker/tidyverse` as its base image, which pre-installs the tidyverse ecosystem, pandoc, and rmarkdown. You do not need to install those packages inside the container.

What this profile provides out of the box:

- Base image: `rocker/tidyverse` (~1.2 GB)
- Pre-installed R packages: `renv`, `devtools`, `tidyverse`, `here`
- Pandoc, rmarkdown, and knitr are available without additional installation

Prerequisites:

- Docker Desktop installed and running
- `zzcollab` installed (`./install.sh` from the repository)
- `gum` installed for styled prompts (optional): `brew install gum`
- GitHub account and a personal access token with `repo` permissions

Phase summary:

Phase	Items	Deliverable
1	1-8	Buildable workspace with documented raw data
2	9-22	Cleaned, validated derived dataset
3	23-31	EDA report with Table 1 and trajectory plots
4	32-38	Tested R functions for summary, modelling, plotting
5	39-45	Primary analysis report with sensitivity analyses
6	46-55	Reproducible compendium ready for archival

Work through items sequentially. Each item names a verifiable end state. Mark items that do not apply as 'N/A' with a one-line reason rather than skipping them – stable numbering allows unambiguous progress discussion with collaborators.

One-Time Identity Setup

Before creating any project, populate your user configuration. This runs once and is never prompted again.

```
# Check current settings
zzc config list

# Set your identity (if not already configured)
zzc config set author-name  "Your Name"
zzc config set author-email "you@example.com"
zzc config set github-account "youruser"
```

With `gum` installed, `zzc init` presents a styled TUI on first run. Without it, plain readline prompts are used. Either way, the values are saved to `~/.zzcollab/config.yaml` and pre-filled on every subsequent project.

Phase 1: Project Setup

Deliverable: a buildable workspace with documented raw data.

Before Item 1, confirm your identity settings are correct:

```
zzc config list
# key fields: author-name, author-email, github-account
```

Item 1 – Verify `Dockerfile` exists and builds successfully.

Create the project using the `analysis` quickstart command. This runs `zzc init`, generates `renv.lock`, and generates the Dockerfile in one step:

```
mkdir my-analysis && cd my-analysis
zzc analysis
```

The `analysis` command uses `rocker/tidyverse` as the base image. After the scaffold completes, build the image:

```
make docker-build
```

Verify the build:

```
docker images my-analysis
# should show a recent image, size ~1.2-1.5 GB
```

The generated `Dockerfile` will look like:

```
# Generated by zzcollab -- do not edit manually.
FROM rocker/tidyverse:4.6.0@sha256:<digest>

ENV LANG=en_US.UTF-8 LC_ALL=en_US.UTF-8 TZ=UTC \
    RENV_PATHS_LIBRARY=/opt/renv/library \
    RENV_PATHS_CACHE=/opt/renv/cache \
    ZZCOLLAB_CONTAINER=true \
    ZZCOLLAB_AUTO_RESTORE=false

RUN apt-get update && apt-get install -y --no-install-recommends \
    libcurl4-openssl-dev libssl-dev libxml2-dev && \
    rm -rf /var/lib/apt/lists/*

RUN R -e "install.packages('renv')"
RUN mkdir -p /opt/renv/library /opt/renv/cache
COPY renv.lock renv.lock
RUN R -e "renv::init(bare=TRUE, force=TRUE, restart=FALSE); renv::restore()"
```

Item 2 – Verify `renv.lock` contains required packages.

The `zzc analysis` command initialises `renv.lock` with the profile's base packages. Inspect it:

```
jq '.Packages | keys[]' renv.lock | head -20
```

At project start, `renv.lock` should list `renv`, `devtools`, `tidyverse`, and `here`. Transitive dependencies are resolved by `renv::restore()` inside the container at build time.

Item 3 – Verify `.Rprofile` activates renv.

```
grep 'renv/activate' .Rprofile
# Expected output: source("renv/activate.R")
```

The `.Rprofile` in the scaffold activates renv for interactive sessions outside the container. Inside the container, `ZZCOLLAB_AUTO_RESTORE=false` suppresses auto-restore because packages were baked into the image at build time.

Item 4 – Verify source code directories exist.

```
ls -d R/ analysis/scripts/ analysis/data/ tests/
```

If any directory is missing, create it:

```
mkdir -p R/ analysis/scripts/ analysis/data/raw_data/ \
         analysis/data/derived_data/ analysis/figures/ \
         analysis/report/ inst/tinytest/
```

Item 5 – Verify raw data is in `analysis/data/raw_data/` (read-only).

For this walkthrough, write the Palmer Penguins dummy dataset directly to the raw data directory. Create `analysis/data/raw_data/penguins_raw.csv`:

```
species,island,bill_length_mm,bill_depth_mm,flipper_length_mm,body_mass_g,sex,year
Adelie,Torgersen,39.1,18.7,181,3750,male,2007
Adelie,Torgersen,39.5,17.4,186,3800,female,2007
Adelie,Torgersen,40.3,18.0,195,3250,female,2007
Adelie,Torgersen,NA,NA,NA,NA,NA,2007
Gentoo,Biscoe,46.1,13.2,211,4500,female,2007
Gentoo,Biscoe,50.0,16.3,230,5700,male,2007
Chinstrap,Dream,46.5,17.9,192,3500,female,2007
Chinstrap,Dream,50.0,19.5,196,3900,male,2007
```

Row 4 simulates a field equipment failure and carries NAs in all measurement columns. After writing the file, set it read-only:

```
chmod 444 analysis/data/raw_data/penguins_raw.csv
ls -l analysis/data/raw_data/
# -r--r--r--  penguins_raw.csv
```

Never modify raw data files. If a correction is required, treat it as a new version with a new filename and document the change.

Item 6 – Update `analysis/data/README.md` with source, date, and use restrictions.

Create `analysis/data/README.md`:

```
# Data: Palmer Penguins (walkthrough example)

## Raw Data

**Source:** Simulated field measurements; based on Gorman et al. (2014),
  Palmer Station Antarctica LTER.
**Date received:** 2026-06-11 (walkthrough example, not real field data)
**File(s):**
- `raw_data/penguins_raw.csv` -- bill, flipper, and body mass
  measurements for 8 penguins across 3 species and 3 islands.

## IRB / Data Use Agreement

- IRB protocol: not applicable (simulated data)
- PHI present: no
- Data use agreement: not applicable

## Known Issues

- Row 4 (Adelie, Torgersen, 2007): all measurement columns are NA
  due to simulated equipment failure.
- Sample size (n=8) is illustrative only; real analyses require
  the full 344-row dataset from the palmerpenguins R package.
```

Item 7 – Record data dictionary.

Create `docs/data-dictionary.md` with one row per column:

```
# Data Dictionary: penguins_raw.csv

*Last updated: 2026-06-11*
```

```
| Column | Type | Units | Allowed values / range | Description |
|--------|------|-------|------------------------|-------------|
| species | character | -- | Adelie, Gentoo, Chinstrap | Penguin species (primary grouping) |
| island | character | -- | Torgersen, Biscoe, Dream | Island of observation |
| bill_length_mm | numeric | mm | 30-65, NA allowed | Culmen length |
| bill_depth_mm | numeric | mm | 10-25, NA allowed | Culmen depth |
| flipper_length_mm | integer | mm | 170-235, NA allowed | Flipper length |
| body_mass_g | integer | g | 2500-6500, NA allowed | Body mass |
| sex | character | -- | male, female, NA allowed | Recorded sex |
| year | integer | -- | 2007-2009 | Year of observation |
```

The data dictionary is a contract between the data provider and the analyst. Phase 2 tests validate this contract.

Item 8 – Install analysis packages inside the container.

Enter the container and install packages. The `analysis` profile pre-installs `tidyverse` and `here`; install additional packages as needed:

```
make r
```

Inside R:

```
install.packages(c('janitor', 'skimr', 'gtsummary', 'broom'))
# packages are captured in renv.lock on exit (auto-snapshot)
q()
```

After exiting, record the new packages explicitly to avoid snapshot side-effects:

```
# (run outside container, from project root)
renv::record(c('janitor', 'skimr', 'gtsummary', 'broom'))
```

Add each package to `DESCRIPTION Imports:` so the declared dependency set matches the environment:

```
Imports:
    dplyr,
    ggplot2,
    here,
    janitor,
    readr,
    skimr,
    tidyr
```

Commit the updated `renv.lock` and `DESCRIPTION`:

```
git add renv.lock DESCRIPTION
git commit -m "Phase 1 complete: add analysis packages"
```

Create the GitHub repository:

```
zzc github     # creates private repo and pushes
```

Phase 2: Data Ingestion and Validation

Deliverable: a cleaned, validated derived dataset.

Item 9 – Create `R/load_data.R` with a function to read raw data.

```
#' Load raw data
#'
#' @param path Path to raw CSV file.
#' @return A tibble of raw observations.
#' @export
load_raw_data <- function(
    path = here::here('analysis', 'data', 'raw_data', 'penguins_raw.csv')) {
  readr::read_csv(path, show_col_types = FALSE)
}
```

Use `here::here()` so the function resolves correctly from any working directory, including inside the container where the bind-mount root differs from the host.

Item 10 – Create `inst/tinytest/test_data_integrity.R`.

```
# inst/tinytest/test_data_integrity.R
# Run with: tinytest::run_test_file('inst/tinytest/test_data_integrity.R')

dat <- load_raw_data()

# Items 11-17 become expect_* calls in this file.
```

Run all project tests:

```
make docker-test
```

Item 11 – Test: expected number of columns.

```
N_EXPECTED_COLS <- 8L   # from data dictionary
expect_equal(ncol(dat), N_EXPECTED_COLS,
             info = 'column count matches data dictionary')
```

Item 12 – Test: expected column names present.

```
expected_names <- c('species', 'island', 'bill_length_mm',
                    'bill_depth_mm', 'flipper_length_mm',
                    'body_mass_g', 'sex', 'year')
expect_true(all(expected_names %in% names(dat)),
            info = 'all documented columns present')
```

Item 13 – Test: ID columns have no NAs and follow expected format.

The penguin dataset has no explicit identifier column; row position serves as the record key. Test that `species` and `island` – the two columns that define a record's provenance – are never NA:

```
expect_true(all(!is.na(dat$species)),
            info = 'species has no missing values')
expect_true(all(!is.na(dat$island)),
            info = 'island has no missing values')
```

Item 14 – Test: categorical variables match allowed levels.

```
allowed_species <- c('Adelie', 'Gentoo', 'Chinstrap')
expect_true(all(dat$species %in% allowed_species),
            info = 'species values are in the documented set')

allowed_islands <- c('Torgersen', 'Biscoe', 'Dream')
expect_true(all(dat$island %in% allowed_islands),
            info = 'island values are in the documented set')
```

Item 15 – Test: demographic categoricals match coding scheme.

```
# sex coded as 'male' / 'female' per data dictionary; NA allowed
allowed_sex <- c('male', 'female', NA)
expect_true(all(dat$sex %in% allowed_sex),
            info = 'sex coding matches documented scheme')
```

Item 16 – Test: time/visit variables fall in the pre-specified set.

The penguin data is cross-sectional within each year. Test that the `year` column is within the documented observation window:

```
allowed_years <- c(2007L, 2008L, 2009L)
expect_true(all(dat$year %in% allowed_years),
            info = 'year values are within documented range')
```

Item 17 – Test: numeric columns are within plausible ranges.

```
# bill_length_mm: 30-65 mm per data dictionary; NA allowed (row 4)
expect_true(
  all(dat$bill_length_mm >= 30 &
      dat$bill_length_mm <= 65, na.rm = TRUE),
  info = 'bill_length_mm within plausible range')

# body_mass_g: 2500-6500 g
expect_true(
  all(dat$body_mass_g >= 2500 &
      dat$body_mass_g <= 6500, na.rm = TRUE),
  info = 'body_mass_g within plausible range')
```

Item 18 – Create `analysis/scripts/01-clean-data.R`.

```
# analysis/scripts/01-clean-data.R
# Input:  analysis/data/raw_data/penguins_raw.csv
# Output: analysis/data/derived_data/clean_data.rds

library(dplyr)
library(here)
library(janitor)
library(readr)

raw <- load_raw_data()

clean <- raw |>
```

```
  janitor::clean_names() |>     # Item 19: snake_case column names
  mutate(                        # Item 20: factors with explicit levels
    species = factor(species,
                     levels = c('Adelie', 'Gentoo', 'Chinstrap')),
    island  = factor(island,
                     levels = c('Torgersen', 'Biscoe', 'Dream')),
    sex     = factor(sex, levels = c('female', 'male'))
  )
  # Item 21: row 4 has NA in all measurement columns (equipment failure).
  # Rows are retained; na.rm = TRUE used in summaries; models use
  # na.action = na.omit by default (documented in 03-primary-analysis.R).

saveRDS(clean,
        here('analysis', 'data', 'derived_data', 'clean_data.rds'))
message('Clean data written: ', nrow(clean), ' rows')
```

Item 19 – Clean column names with `janitor::clean_names()`.

This step is embedded in `01-clean-data.R` above. It converts all headers to `snake_case`, handles BOM-affected first columns, and makes all names consistent for downstream code. Update the data dictionary to record both the original and cleaned name for every column.

Item 20 – Convert categorical variables to factors with explicit levels.

Place the reference level first – control before treatment, baseline before follow-up – so model contrasts produce intuitive coefficients. Document the reference level choice in a code comment.

Item 21 – Handle missing values explicitly and document the approach.

Do not call `na.omit()` silently. For each column with NAs, add a comment:

```
# bill_length_mm, bill_depth_mm, flipper_length_mm, body_mass_g, sex:
# row 4 is NA in all measurement columns (equipment failure, documented
# in data README). Rows are retained; summaries use na.rm = TRUE;
# models drop the NA row via na.action = na.omit (documented in
# 03-primary-analysis.R). No imputation for this illustrative example.
```

Item 22 – Save cleaned data to `analysis/data/derived_data/`.

```
# Run the cleaning script inside the container
make r
```

```
source(here::here('analysis', 'scripts', '01-clean-data.R'))
```

Verify the output:

```
clean <- readRDS(here::here('analysis', 'data',
                            'derived_data', 'clean_data.rds'))
glimpse(clean)
```

Commit:

```
git add analysis/scripts/01-clean-data.R \
        inst/tinytest/test_data_integrity.R
git commit -m "Phase 2 complete: cleaning pipeline and integrity tests"
```

Phase 3: Exploratory Analysis

Deliverable: an EDA report with Table 1 and trajectory plots.

Item 23 – Create `analysis/scripts/02-eda.R`.

```
# analysis/scripts/02-eda.R
# Input:  analysis/data/derived_data/clean_data.rds
# Output: analysis/data/derived_data/summary_stats.csv
#         analysis/figures/outcome-distributions.png
#         analysis/figures/outcome-trajectories.png

library(dplyr)
library(ggplot2)
library(here)
library(skimr)

clean <- readRDS(here('analysis', 'data',
                      'derived_data', 'clean_data.rds'))
```

Item 24 – Generate summary statistics by group.

```
summary_stats <- clean |>
  group_by(group) |>
  summarise(
    n = n(),
    across(where(is.numeric),
           list(mean = ~mean(.x, na.rm = TRUE),
                sd   = ~sd(.x, na.rm = TRUE),
                med  = ~median(.x, na.rm = TRUE)),
           .names = '{.col}_{.fn}'),
    .groups = 'drop')

readr::write_csv(summary_stats,
                 here('analysis', 'data',
                      'derived_data', 'summary_stats.csv'))
```

Item 25 – Check baseline balance.

```
library(gtsummary)

table1 <- clean |>
  select(species, island, sex, bill_length_mm, body_mass_g) |>
  tbl_summary(
    by = species,
    statistic = list(
      all_continuous() ~ '{mean} ({sd})',
      all_categorical() ~ '{n} ({p}%)'),
    digits = all_continuous() ~ 1) |>
  modify_header(label ~ '**Variable**') |>
  bold_labels()

# Save as HTML for the EDA report
table1 |>
  as_gt() |>
  gt::gtsave(here('analysis', 'figures', 'table1.html'))
```

Item 26 – Visualise outcome distributions.

```
p_dist <- ggplot(clean,
                 aes(x = bill_length_mm, fill = species)) +
  geom_histogram(position = 'identity', alpha = 0.6,
                 bins = 10) +
  facet_wrap(~species) +
  labs(title = 'Bill length distribution by species',
```

```
       x = 'Bill length (mm)',
       y = 'Count') +
  theme_minimal() +
  theme(legend.position = 'none')

ggsave(here('analysis', 'figures',
            'outcome-distributions.png'),
       p_dist, width = 8, height = 4, dpi = 150)
```

Item 27 – Identify potential outliers.

```
# Boxplot for visual outlier detection
p_box <- ggplot(clean,
                aes(x = species, y = bill_length_mm,
                    colour = species)) +
  geom_boxplot(outlier.size = 2) +
  geom_jitter(width = 0.15, alpha = 0.4) +
  labs(title = 'Bill length by species',
       x = NULL, y = 'Bill length (mm)') +
  theme_minimal() +
  theme(legend.position = 'none')

ggsave(here('analysis', 'figures', 'boxplot-baseline.png'),
       p_box, width = 6, height = 4, dpi = 150)
```

Flag and document any observation more than 3 SD from the group mean in a tracking file: `docs/outlier-log.md`.

Item 28 – Create `analysis/report/01-eda.Rmd`.

```
---
title: "Exploratory Data Analysis"
author: "Author"
date: "2026-06-11"
output:
  html_document:
    toc: true
    code_folding: show
---



![  ](media/images/ambiance2.jpg){.img-fluid width=80% fig-align="center"}
```

````
## Sample overview

::: {.cell}

```{.r .cell-code}
clean <- readRDS(here('analysis', 'data',
                      'derived_data', 'clean_data.rds'))
```
:::

## Table 1: Baseline characteristics

::: {.cell}

```{.r .cell-code}
# (produced by 02-eda.R and loaded here)
knitr::include_graphics(here('analysis', 'figures', 'table1.html'))
```
:::
````

Render inside the container:

```
make r
```

```
rmarkdown::render(
  here::here('analysis', 'report', '01-eda.Rmd'),
  output_dir = here::here('analysis', 'report'))
```

Item 29 – Document sample size and missingness patterns.

Add a missingness section to `01-eda.Rmd`:

```
# Missingness summary
clean |>
  summarise(across(everything(),
                   ~sum(is.na(.x)) / n() * 100)) |>
  tidyr::pivot_longer(everything(),
                      names_to  = 'variable',
                      values_to = 'pct_missing') |>
  filter(pct_missing > 0) |>
  arrange(desc(pct_missing))
```

Item 30 – Include baseline characteristics table (Table 1).

The `gtsummary::tbl_summary()` call from Item 25 produces Table 1. For randomised trials, report without p-values for balance tests; they are uninformative when randomisation succeeded. Use `add_p()` only for pre-specified covariate-selection decisions.

Item 31 – Include outcome trajectory plots by visit and group.

```
# Requires long-format data with visit and outcome columns
p_traj <- clean |>
  tidyr::pivot_longer(
    cols = starts_with('outcome_'),
    names_to  = 'visit',
    values_to = 'outcome') |>
  mutate(visit = stringr::str_remove(visit, 'outcome_')) |>
  group_by(group, visit) |>
  summarise(
    mean  = mean(outcome, na.rm = TRUE),
    se    = sd(outcome, na.rm = TRUE) / sqrt(n()),
    .groups = 'drop') |>
  ggplot(aes(x = visit, y = mean,
             colour = group, group = group)) +
  geom_line() +
  geom_point() +
  geom_errorbar(aes(ymin = mean - se, ymax = mean + se),
                width = 0.15) +
  labs(title = 'Mean outcome trajectory by group',
       x = 'Visit', y = 'Mean outcome (SE)',
       colour = 'Group') +
  theme_minimal()

ggsave(here('analysis', 'figures', 'trajectories.png'),
       p_traj, width = 8, height = 5, dpi = 150)
```

Commit:

```
git add analysis/scripts/02-eda.R \
        analysis/report/01-eda.Rmd \
        analysis/figures/
git commit -m "Phase 3 complete: EDA script, Table 1, trajectory plots"
```

Phase 4: Analysis Functions

Deliverable: tested R functions for summary, modelling, and plotting.

Item 32 – Create `R/summarize_outcomes.R`.

```
#' Descriptive statistics for a numeric outcome
#'
#' @param data A data frame with columns `group` and `outcome`.
#' @param outcome_col Character name of the outcome column.
#' @return A tibble with one row per group.
#' @export
summarize_outcome <- function(data, outcome_col) {
  data |>
    dplyr::group_by(group) |>
    dplyr::summarise(
      n      = dplyr::n(),
      mean   = mean(.data[[outcome_col]], na.rm = TRUE),
      sd     = sd(.data[[outcome_col]], na.rm = TRUE),
      median = median(.data[[outcome_col]], na.rm = TRUE),
      q25    = quantile(.data[[outcome_col]], 0.25, na.rm = TRUE),
      q75    = quantile(.data[[outcome_col]], 0.75, na.rm = TRUE),
      n_miss = sum(is.na(.data[[outcome_col]])),
      .groups = 'drop')
}
```

Item 33 — Create `R/model_outcomes.R`.

```
#' Fit primary analysis model
#'
#' @param data Cleaned data frame.
#' @param outcome Character name of outcome column.
#' @param covariates Character vector of covariate names.
#' @return A list with `model` (lm/glm object) and `tidy`
#'   (broom::tidy output).
#' @export
fit_primary_model <- function(data, outcome, covariates = NULL) {
  rhs <- if (is.null(covariates)) {
    'group'
  } else {
    paste(c('group', covariates), collapse = ' + ')
  }
  formula <- stats::as.formula(paste(outcome, '~', rhs))
  model   <- stats::lm(formula, data = data)
  list(
    model = model,
    tidy  = broom::tidy(model, conf.int = TRUE),
    glance = broom::glance(model))
}
```

Item 34 – Create `R/plot_outcomes.R`.

```
#' Grouped boxplot for a numeric outcome
#'
#' Returns a ggplot object; the caller controls ggsave.
#'
#' @param data Data frame with columns `group` and outcome.
#' @param outcome_col Character name of the outcome column.
#' @param title Plot title string.
#' @return A ggplot object.
#' @export
plot_outcome_box <- function(data, outcome_col, title = NULL) {
  assertthat::assert_that(outcome_col %in% names(data),
    msg = paste('column not found:', outcome_col))
  ggplot2::ggplot(data,
    ggplot2::aes(x = group,
                 y = .data[[outcome_col]],
                 colour = group)) +
    ggplot2::geom_boxplot(outlier.size = 1.5) +
    ggplot2::geom_jitter(width = 0.12, alpha = 0.35) +
    ggplot2::labs(title = title, x = NULL,
                  y = outcome_col) +
    ggplot2::theme_minimal() +
    ggplot2::theme(legend.position = 'none')
}
```

Item 35 – Create `inst/tinytest/test_summarize.R`.

```
# inst/tinytest/test_summarize.R

# Build a minimal data frame with known answers
test_dat <- data.frame(
  group = factor(rep(c('control', 'treatment'), each = 4L)),
  score = c(10, 12, 11, 13, 20, 22, 21, 23))

result <- summarize_outcome(test_dat, 'score')

expect_equal(nrow(result), 2L)
expect_true('mean' %in% names(result))
expect_equal(result$mean[result$group == 'control'],   11.5)
expect_equal(result$mean[result$group == 'treatment'], 21.5)

# Missing data: n_miss column is correct
test_dat_na        <- test_dat
test_dat_na$score[1L] <- NA
```

```
result_na <- summarize_outcome(test_dat_na, 'score')
expect_equal(result_na$n_miss[result_na$group == 'control'], 1L)
```

Item 36 – Create `inst/tinytest/test_model.R`.

```
# inst/tinytest/test_model.R
set.seed(42L)
n   <- 100L
sim <- data.frame(
  group = factor(rep(c('control', 'treatment'), each = n / 2L)),
  y     = c(rnorm(n / 2L, mean = 10), rnorm(n / 2L, mean = 15)))

res <- fit_primary_model(sim, 'y')

expect_true(is.list(res))
expect_true(all(c('model', 'tidy', 'glance') %in% names(res)))
# treatment effect should be ~5
treat_est <- res$tidy$estimate[res$tidy$term == 'grouptreatment']
expect_true(abs(treat_est - 5) < 1.5,
            info = 'treatment estimate within 1.5 of true value')
```

Item 37 – Test edge cases: missing data handling.

```
# inst/tinytest/test_model.R (continued)
sim_na       <- sim
sim_na$y[1L] <- NA

# lm() with NA: default na.action = na.omit, should complete
res_na <- fit_primary_model(sim_na, 'y')
expect_true(!is.null(res_na$model),
            info = 'model fits with NA in outcome (na.omit default)')
```

Item 38 – Test edge cases: single-group scenario.

```
# Single-level factor: lm should error or return NA coefficient
single_group <- data.frame(
  group = factor(rep('control', 10L)),
  y     = rnorm(10L))
expect_error(fit_primary_model(single_group, 'y'),
             info = 'single-group data produces an error or degenerate model')
```

Run all tests inside the container:

```
make docker-test
```

Commit:

```
git add R/ inst/tinytest/
git commit -m "Phase 4 complete: analysis functions and unit tests"
```

Phase 5: Primary Analysis

Deliverable: a primary analysis report with sensitivity analyses.

Item 39 – Create `analysis/scripts/03-primary-analysis.R`.

```
# analysis/scripts/03-primary-analysis.R
# Input:  analysis/data/derived_data/clean_data.rds
# Output: analysis/data/derived_data/primary-results.rds
#         analysis/data/derived_data/primary-tidy.csv

library(here)

clean <- readRDS(here('analysis', 'data',
                      'derived_data', 'clean_data.rds'))

# Primary model: outcome_week12 ~ group + age + sex
primary <- fit_primary_model(clean, 'outcome_week12',
                             covariates = c('age', 'sex'))

# Sensitivity 1: complete cases only (no imputation)
clean_cc  <- clean[stats::complete.cases(clean), ]
sens_cc   <- fit_primary_model(clean_cc, 'outcome_week12',
                               covariates = c('age', 'sex'))

# Save results
saveRDS(list(primary = primary, sensitivity_cc = sens_cc),
        here('analysis', 'data',
             'derived_data', 'primary-results.rds'))

readr::write_csv(primary$tidy,
                 here('analysis', 'data',
                      'derived_data', 'primary-tidy.csv'))
message('Primary analysis complete.')
```

Item 40 – Implement the pre-specified analysis plan.

Every model call should correspond one-to-one with a row in the statistical analysis plan (SAP). Mark deviations clearly:

```
# SAP DEVIATION: age was pre-specified as continuous but is categorised
# here at the request of the clinical team (email 2026-05-15). The
# continuous model is retained as a sensitivity analysis.
```

Item 41 – Save model objects to `analysis/data/derived_data/`.

The `saveRDS()` calls in `03-primary-analysis.R` above fulfil this item. Naming convention: `model_<outcome>_<model_type>.rds`.

Item 42 – Create `analysis/report/02-results.Rmd`.

This report loads saved objects only – no model fitting inside the report:

````
---
title: "Primary Analysis Results"
date: "2026-06-11"
output:
  html_document:
    toc: true
    number_sections: true
---



::: {.cell}

```{.r .cell-code}
results <- readRDS(here('analysis', 'data',
                        'derived_data', 'primary-results.rds'))
primary <- results$primary
```
:::

## Primary outcome

::: {.cell}

```{.r .cell-code}
knitr::kable(primary$tidy,
             digits = 3,
             caption = 'Primary analysis: outcome_week12 ~ group + age + sex')
```
````

```
:::
```

Item 43 – Document primary outcome analysis.

The results section should present: model formula, point estimate, 95% confidence interval, and p-value for the `group` contrast:

```
# In the report chunk:
primary_row <- primary$tidy[primary$tidy$term == 'grouptreatment', ]
```

Then in prose (using inline R from the loaded `primary_row` object):

```
The treatment group showed a mean difference of
[primary_row$estimate] units (95% CI: [primary_row$conf.low] to
[primary_row$conf.high]) relative to control.
```

In a rendered report that executes against real data, replace the bracketed placeholders with knitr inline expressions of the form `round(primary_row$estimate, 2)`.

Item 44 – Document secondary outcomes.

Follow the same structure for each pre-specified secondary outcome. Declare the multiple-comparison context explicitly; label unadjusted secondary results as exploratory if no correction was pre-specified.

Item 45 – Document sensitivity analyses.

Report the complete-case result alongside the primary, then add any further pre-specified sensitivities (alternative covariates, imputation):

```
sens_row <- results$sensitivity_cc$tidy[
  results$sensitivity_cc$tidy$term == 'grouptreatment', ]
```

The reader should be able to see whether the primary conclusion changes under the sensitivity model.

Render both reports inside the container, then commit:

```
make r
```

```
rmarkdown::render(here::here('analysis', 'report', '02-results.Rmd'))
```

```
git add analysis/scripts/03-primary-analysis.R \
        analysis/report/02-results.Rmd
git commit -m "Phase 5 complete: primary analysis and results report"
```

Phase 6: Documentation and Reproducibility

Deliverable: a reproducible compendium ready for archival.

Item 46 – Update `DESCRIPTION`: Title and Description.

```
Package: myanalysis
Title: <Study name in fewer than 65 characters>
Version: 0.1.0
Description: A reproducible analysis of <study population>,
    examining <primary outcome> using <analytic approach>.
    Data from <source>. Analysis period: <dates>.
```

Item 47 – Update `DESCRIPTION`: Authors@R.

```
Authors@R: person(
  given   = 'Ronald G.',
  family  = 'Thomas',
  role    = c('aut', 'cre'),
  email   = 'rgthomas47@gmail.com',
  comment = c(ORCID = '0000-0000-0000-0000'))
```

Item 48 – Verify all package dependencies in `DESCRIPTION`.

```
# Run inside the container
deps <- renv::dependencies()
unique(deps$Package) |> sort()
```

Cross-check against the `Imports:` field. Any package called in `R/` or `analysis/scripts/` but absent from `Imports:` will fail in a clean container.

Item 49 – Create or update `README.md`.

```
# <Project name>

## Research Question

<One sentence stating the research question.>

## Data

<Source, n observations, primary variables.>

![  ](media/images/ambiance3.jpg){.img-fluid width=80% fig-align="center"}

## Reproduction

Clone, build, and run:

```bash
git clone https://github.com/<org>/<repo>.git
cd <repo>
make docker-build
make r
```

Inside the container:

```
source(here::here('analysis', 'scripts', '01-clean-data.R'))
source(here::here('analysis', 'scripts', '02-eda.R'))
source(here::here('analysis', 'scripts', '03-primary-analysis.R'))
rmarkdown::render(here::here('analysis', 'report', '01-eda.Rmd'))
rmarkdown::render(here::here('analysis', 'report', '02-results.Rmd'))
```

Directory Structure

```
<repo>/
├── R/                    # Reusable functions (Items 32-34)
├── analysis/
│   ├── data/
│   │   ├── raw_data/     # Immutable original data (Item 5)
│   │   └── derived_data/ # Cleaned and modelled outputs (Items 22, 41)
│   ├── figures/          # Saved plots (Items 26-27, 31)
│   ├── report/           # Rmd/qmd reports (Items 28, 42)
│   └── scripts/          # Analysis pipeline (Items 18, 23, 39)
├── inst/tinytest/        # Unit tests (Items 10, 35-38)
├── Dockerfile            # Item 1
├── renv.lock             # Item 2
└── .Rprofile             # Item 3
```
```

```
### Item 50 -- Document reproduction instructions.

Test the reproduction sequence on a fresh machine (or a fresh Docker
context with `docker system prune`) before claiming reproducibility.
Instructions that work only on the author's laptop are not reproduction
instructions.

### Item 51 -- Rebuild Docker image after all changes.

```bash
make docker-build
docker images <project-name>
```

Tag the image with the current git SHA:

```
git_sha=$(git rev-parse --short HEAD)
docker tag <project-name>:latest <project-name>:${git_sha}
```

Item 52 – Verify reproducibility in a fresh container.

```
# Remove any dangling state and rebuild from scratch
docker rmi <project-name>:latest
make docker-build
make r
```

Inside the container:

```
source(here::here('analysis', 'scripts', '01-clean-data.R'))
source(here::here('analysis', 'scripts', '02-eda.R'))
source(here::here('analysis', 'scripts', '03-primary-analysis.R'))
rmarkdown::render(here::here('analysis', 'report', '01-eda.Rmd'))
rmarkdown::render(here::here('analysis', 'report', '02-results.Rmd'))
```

Item 53 – Confirm all outputs match expected results.

Compare the freshly generated derived data and PDFs against the previously committed versions. Numerically identical floating-point results and structurally identical tables (same row and column counts, same column types) are the realistic target. Any divergence not explained by a documented code change requires investigation.

```
# Quick structural check
md5sum analysis/data/derived_data/*.rds
md5sum analysis/data/derived_data/*.csv
```
```

Item 54 – Run `make test` and verify all tests pass.

```
make docker-test
```

All tinytest and testthat suites must pass. A failing test at the publication stage is a stop-the-line event.

Item 55 – Final review: all Five Pillars complete and consistent.

Walk through each pillar:

1. **`Dockerfile`** – references `rocker/tidyverse:<r-version>`. The R version matches the one in `renv.lock` (check `jq '.R.Version' renv.lock`). The image builds cleanly from a fresh pull.
2. **`renv.lock`** – lists every direct dependency declared in `DESCRIPTION Imports:`. Run `make check-renv` to confirm.
3. **`.Rprofile`** – calls `source('renv/activate.R')`. `ZZCOLLAB_AUTO_RESTORE=false` is set in the Dockerfile so the lockfile is not re-restored on every session start inside the container.
4. **Source code** – all scripts in `analysis/scripts/` run end-to-end. All functions in `R/` are tested by `inst/tinytest/`. All tests pass.
5. **Research data** – raw data is read-only, documented in `analysis/data/README.md`, and has a data dictionary in `docs/data-dictionary.md`. Derived data was produced by the committed scripts only.

Final commit and push:

```
git add -A
git commit -m "Phase 6 complete: final reproducibility check passed"
git push
zzc dockerhub     # push image to Docker Hub (optional)
```

Next in this cluster

This post is the setup precursor to the *Palmer Penguins Analysis Arc*. Once the compendium is in place and the dummy data pipeline runs end-to-end, continue with the full 344-row dataset:

1. **This post** – zzcollab setup, dummy data pipeline, 55-item checklist
2. Palmer Penguins Part 1: Exploratory Data Analysis – full dataset EDA, simple linear regression, species-level patterns
3. Palmer Penguins Part 2: Multiple Regression
4. Palmer Penguins Part 3: Cross-Validation
5. Palmer Penguins Part 4: Model Diagnostics

6. Palmer Penguins Part 5: Random Forest versus Linear
7. Predictive Modeling of Penguin Body Mass
8. Functional Plot Generation with purrr

Total items: 55. Profile: analysis (rocker/tidyverse ~1.2 GB). Last updated: 2026-06-11.