R Course: Beginner to Expert

Sri

2025-10-24

Table of contents

1 Welcome

2 About this Course

This Quarto book takes you from **R beginner** to **expert**, with a practical focus on **clinical programming** (CDISC/ADaM and TLFs).
Each chapter includes step-by-step explanations, runnable code, and short exercises.

2.1 How to Use

1. Install: R (4.2), RStudio (or VS Code), and Quarto.
2. In a terminal: `quarto render` to build the whole book, or click **Render** in RStudio.
3. Open `index.html` in the `_book/` folder after rendering.

2.2 Structure (Highlights)

- **Basics & Data**: R syntax, data types/structures, vectors/data frames/lists.
- **I/O**: Read SAS datasets (with `haven`), handle labels, and clean raw data.
- **Programming**: Base functions, write your own functions, validate with tests.
- **DevOps**: Create an R package, connect Git in RStudio/GitHub.
- **CDISC**: Build ADaM (ADSL) from SDTM-like inputs.
- **TLFs**: Produce a baseline Table 1, a KM plot, and a listing.

  Tip: If you don't have sample SDTM/ADaM data yet, the chapters generate **small synthetic data** as a fallback so everything runs end-to-end.

3 R Basics

4 R as a Calculator

```
1 + 1
```

```
[1] 2
```

```
3 * (4 + 5)
```

```
[1] 27
```

5 Objects & Assignment

```
x <- 10
y <- 3.5
x + y
```

```
[1] 13.5
```

6 Getting Help

```
?mean
help("lm")
```

7 Working Directory

```
getwd()
```

```
[1] "/home/runner/work/r4sas/r4sas"
```

```
# setwd("/path/you/want") # avoid in reproducible code; prefer here::here() for projects
```

8 Vectors (Atomic)

```r
nums <- c(1, 2, 3, 4)
chars <- c("a", "b", "c")
logical <- c(TRUE, FALSE, TRUE)
typeof(nums); typeof(chars); typeof(logical)
```

```
[1] "double"
```

```
[1] "character"
```

```
[1] "logical"
```

9 Exercises

1. Create an object `z` that stores `(5^2 + 7)/3`.
2. Use `?seq` and create a sequence from 0 to 1 by 0.1.
3. Inspect `typeof()` for a few objects you create.

10 Data Types & Data Structures

11 Atomic Types

- logical, integer, double (numeric), character, complex, raw

```
typeof(TRUE)
```

```
[1] "logical"
```

```
typeof(1L)
```

```
[1] "integer"
```

```
typeof(1.0)
```

```
[1] "double"
```

```
typeof("R")
```

```
[1] "character"
```

12 Factors, Dates, Date-Times

```r
f <- factor(c("Placebo","Active","Active"))
f
```

```
[1] Placebo Active  Active
Levels: Active Placebo
```

```r
as.integer(f)
```

```
[1] 2 1 1
```

```r
d <- as.Date("2025-10-24")
dt <- as.POSIXct("2025-10-24 09:30:00", tz = "Europe/Zurich")
str(list(f=f, d=d, dt=dt))
```

```
List of 3
 $ f : Factor w/ 2 levels "Active","Placebo": 2 1 1
 $ d : Date[1:1], format: "2025-10-24"
 $ dt: POSIXct[1:1], format: "2025-10-24 09:30:00"
```

13 Core Structures

- vector, matrix, array, **list**, **data.frame** / tibble

```r
m <- matrix(1:6, nrow=2)
arr <- array(1:8, dim=c(2,2,2))
lst <- list(a=1:3, b=c("x","y"))
df <- data.frame(id=1:3, val=c(10,20,30))
str(list(m=m, arr=arr, lst=lst, df=df))
```

```
List of 4
 $ m  : int [1:2, 1:3] 1 2 3 4 5 6
 $ arr: int [1:2, 1:2, 1:2] 1 2 3 4 5 6 7 8
 $ lst:List of 2
  ..$ a: int [1:3] 1 2 3
  ..$ b: chr [1:2] "x" "y"
 $ df :'data.frame':    3 obs. of  2 variables:
  ..$ id : int [1:3] 1 2 3
  ..$ val: num [1:3] 10 20 30
```

14 Coercion & Attributes

```r
v <- c(1, "2", 3)   # coerces to character
attributes(df)      # names, row.names, class
```

```
$names
[1] "id"  "val"

$class
[1] "data.frame"

$row.names
[1] 1 2 3
```

15 Exercises

1. Build a 3×3 matrix and extract row 2, col 3.
2. Convert a character vector `c("1","2","3")` to numeric safely.
3. Create a factor with levels in a custom order.

16 Manipulating Vectors, Data Frames, and Lists

```
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(tidyr)
```

17 Vectors: Indexing & Vectorized Ops

```
v <- 1:10
v[v %% 2 == 0]
```

```
[1]  2  4  6  8 10
```

```
v * 2
```

```
 [1]  2  4  6  8 10 12 14 16 18 20
```

18 Data Frames with dplyr

```
df <- tibble::tibble(
  id = 1:6,
  grp = c("A","A","B","B","C","C"),
  age = c(35,44,53,51,29,40),
  wt  = c(70, 85, 92, 88, 60, 75)
)

df |>
  dplyr::group_by(grp) |>
  dplyr::summarise(
    n = dplyr::n(),
    mean_age = mean(age),
    sd_wt = sd(wt)
  )
```

```
# A tibble: 3 x 4
  grp       n mean_age sd_wt
  <chr> <int>    <dbl> <dbl>
1 A         2     39.5 10.6
2 B         2     52    2.83
3 C         2     34.5 10.6
```

18.1 mutate() + across()

```
df |>
  mutate(
    bmi = wt / (1.70^2),
    across(c(age, wt), ~ .x - mean(.x), .names = "{.col}_centered")
  )
```

```
# A tibble: 6 x 7
     id grp     age    wt   bmi age_centered wt_centered
  <int> <chr> <dbl> <dbl> <dbl>        <dbl>       <dbl>
1     1 A        35    70  24.2           -7       -8.33
2     2 A        44    85  29.4            2        6.67
3     3 B        53    92  31.8           11       13.7
4     4 B        51    88  30.4            9        9.67
5     5 C        29    60  20.8          -13      -18.3
6     6 C        40    75  26.0           -2       -3.33
```

18.2 Row-wise with `c_across()`

```
row_sums <- df |>
  rowwise() |>
  mutate(sum_age_wt = sum(c_across(c(age, wt)))) |>
  ungroup()
row_sums
```

```
# A tibble: 6 x 5
     id grp     age    wt sum_age_wt
  <int> <chr> <dbl> <dbl>      <dbl>
1     1 A        35    70        105
2     2 A        44    85        129
3     3 B        53    92        145
4     4 B        51    88        139
5     5 C        29    60         89
6     6 C        40    75        115
```

19 Lists: `lapply`, purrr

```
lst <- list(a=1:3, b=10:12)
lapply(lst, mean)
```

```
$a
[1] 2

$b
[1] 11
```

20 Exercises

1. Using `across()`, standardize (`(x-mean)/sd`) numeric columns.
2. Create a row-wise mean of `age` and `wt`.
3. Split `df` by `grp` and compute group means with `lapply` or `purrr::map`.

21 Reading SAS Datasets (+ Cleaning)

```
library(haven)
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(labelled)
```

We try to read a SAS dataset (e.g., SDTM DM). If not present, we **synthesize** an example.

```
dm_path <- "data/sdtm/dm.sas7bdat"

if (file.exists(dm_path)) {
  dm <- read_sas(dm_path)
} else {
  dm <- tibble::tibble(
    STUDYID = "XYZ123",
    USUBJID = sprintf("XYZ-%03d", 1:10),
    ARM = rep(c("Placebo","Active"), length.out=10),
    AGE = c(55, 62, 47, 50, 71, 66, 45, 59, 53, 68),
    SEX = rep(c("M","F"), length.out=10)
  )
  message("Synthesized `dm` since data/sdtm/dm.sas7bdat was not found.")
}
str(dm)
```

```
tibble [306 x 25] (S3: tbl_df/tbl/data.frame)
 $ STUDYID : chr [1:306] "CDISCPILOT01" "CDISCPILOT01" "CDISCPILOT01" "CDISCPILOT01" ...
  ..- attr(*, "label")= chr "Study Identifier"
 $ DOMAIN  : chr [1:306] "DM" "DM" "DM" "DM" ...
  ..- attr(*, "label")= chr "Domain Abbreviation"
 $ USUBJID : chr [1:306] "01-701-1015" "01-701-1023" "01-701-1028" "01-701-
1033" ...
  ..- attr(*, "label")= chr "Unique Subject Identifier"
 $ SUBJID  : chr [1:306] "1015" "1023" "1028" "1033" ...
  ..- attr(*, "label")= chr "Subject Identifier for the Study"
 $ RFSTDTC : chr [1:306] "2014-01-02" "2012-08-05" "2013-07-19" "2014-03-
18" ...
  ..- attr(*, "label")= chr "Subject Reference Start Date/Time"
 $ RFENDTC : chr [1:306] "2014-07-02" "2012-09-02" "2014-01-14" "2014-04-
14" ...
  ..- attr(*, "label")= chr "Subject Reference End Date/Time"
 $ RFXSTDTC: chr [1:306] "2014-01-02" "2012-08-05" "2013-07-19" "2014-03-
18" ...
  ..- attr(*, "label")= chr "Date/Time of First Study Treatment"
 $ RFXENDTC: chr [1:306] "2014-07-02" "2012-09-01" "2014-01-14" "2014-03-
31" ...
  ..- attr(*, "label")= chr "Date/Time of Last Study Treatment"
 $ RFICDTC : chr [1:306] "" "" "" "" ...
  ..- attr(*, "label")= chr "Date/Time of Informed Consent"
 $ RFPENDTC: chr [1:306] "2014-07-02T11:45" "2013-02-18" "2014-01-14T11:10" "2014-
09-15" ...
  ..- attr(*, "label")= chr "Date/Time of End of Participation"
 $ DTHDTC  : chr [1:306] "" "" "" "" ...
  ..- attr(*, "label")= chr "Date/Time of Death"
 $ DTHFL   : chr [1:306] "" "" "" "" ...
  ..- attr(*, "label")= chr "Subject Death Flag"
 $ SITEID  : chr [1:306] "701" "701" "701" "701" ...
  ..- attr(*, "label")= chr "Study Site Identifier"
 $ AGE     : num [1:306] 63 64 71 74 77 85 59 68 81 84 ...
  ..- attr(*, "label")= chr "Age"
 $ AGEU    : chr [1:306] "YEARS" "YEARS" "YEARS" "YEARS" ...
  ..- attr(*, "label")= chr "Age Units"
 $ SEX     : chr [1:306] "F" "M" "M" "M" ...
  ..- attr(*, "label")= chr "Sex"
 $ RACE    : chr [1:306] "WHITE" "WHITE" "WHITE" "WHITE" ...
  ..- attr(*, "label")= chr "Race"
 $ ETHNIC  : chr [1:306] "HISPANIC OR LATINO" "HISPANIC OR LATINO" "NOT HISPANIC OR LATINO" "
  ..- attr(*, "label")= chr "Ethnicity"
```

```
 $ ARMCD   : chr [1:306] "Pbo" "Pbo" "Xan_Hi" "Xan_Lo" ...
  ..- attr(*, "label")= chr "Planned Arm Code"
 $ ARM     : chr [1:306] "Placebo" "Placebo" "Xanomeline High Dose" "Xanomeline Low Dose" ..
  ..- attr(*, "label")= chr "Description of Planned Arm"
 $ ACTARMCD: chr [1:306] "Pbo" "Pbo" "Xan_Hi" "Xan_Lo" ...
  ..- attr(*, "label")= chr "Actual Arm Code"
 $ ACTARM  : chr [1:306] "Placebo" "Placebo" "Xanomeline High Dose" "Xanomeline Low Dose" ..
  ..- attr(*, "label")= chr "Description of Actual Arm"
 $ COUNTRY : chr [1:306] "USA" "USA" "USA" "USA" ...
  ..- attr(*, "label")= chr "Country"
 $ DMDTC   : chr [1:306] "2013-12-26" "2012-07-22" "2013-07-11" "2014-03-
10" ...
  ..- attr(*, "label")= chr "Date/Time of Collection"
 $ DMDY    : num [1:306] -7 -14 -8 -8 -7 -21 NA -9 -13 -7 ...
  ..- attr(*, "label")= chr "Study Day of Collection"
```

21.1 Handling Labels & Missing

```r
# Example: Convert blank strings "" to NA for character columns
convert_blanks_to_na <- function(x) {
  if (is.character(x)) x[x == ""] <- NA_character_
  x
}
dm <- dm |> mutate(across(where(is.character), convert_blanks_to_na))
```

21.2 Labelled to Factor (if needed)

```r
if (inherits(dm$SEX, "labelled")) {
  dm <- dm |> mutate(SEX = to_factor(SEX))
}
```

21.3 Common Cleaning

```r
dm <- dm |>
  mutate(
    AGEGR1 = cut(AGE, breaks=c(-Inf, 50, 65, Inf),
                 labels=c("<50","50-65", "65+"))
  )
```

22 Exercises

1. Read another SAS dataset (e.g., `sv.sas7bdat`) if available. If not, create a synthetic tibble.
2. Write a function to trim character whitespace for all character columns.
3. Make a clean factor for `ARM` with levels `Placebo < Active`.

23 Base R Functions & Apply Family

24 Common Utilities

```
x <- 1:10
sum(x); mean(x); sd(x); var(x); quantile(x)
```

```
[1] 55
```

```
[1] 5.5
```

```
[1] 3.02765
```

```
[1] 9.166667
```

```
   0%   25%   50%   75%  100%
 1.00  3.25  5.50  7.75 10.00
```

```
seq(0, 1, by=0.1); rep(5, times=3)
```

```
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

```
[1] 5 5 5
```

25 Apply Family

```
m <- matrix(1:9, nrow=3)
apply(m, 1, mean)  # row means
```

```
[1] 4 5 6
```

```
apply(m, 2, mean)  # col means
```

```
[1] 2 5 8
```

```
lst <- list(a=1:3, b=10:12)
sapply(lst, mean)   # simplifies result
```

```
 a  b
 2 11
```

```
mapply(sum, 1:3, 10:12)
```

```
[1] 11 13 15
```

26 Subsetting Essentials

```
df <- data.frame(id=1:3, val=c(10,20,30))
df[1, "val"]
```

```
[1] 10
```

```
df[df$val > 10, ]
```

```
  id val
2  2  20
3  3  30
```

27 Exercises

1. Use `apply` to get the max per column of a numeric matrix.
2. Write a base R snippet to compute IQR for each column of `mtcars`.
3. Compare `lapply` vs `sapply` in behavior on a list with mixed types.

28 Custom Functions & Validation

29 Writing Functions

```
safe_mean <- function(x, na.rm = TRUE) {
  stopifnot(is.numeric(x))
  mean(x, na.rm = na.rm)
}
safe_mean(c(1, 2, NA))
```

```
[1] 1.5
```

30 Error Handling

```
robust_divide <- function(a, b) {
  tryCatch(a / b,
           warning = function(w) NA_real_,
           error   = function(e) NA_real_)
}
robust_divide(10, 0)
```

```
[1] Inf
```

31 Unit Testing with testthat

Install once: `install.packages(c("testthat","devtools","usethis","roxygen2"))`

```
usethis::use_testthat()
usethis::use_test("safe_mean")
```

Create `tests/testthat/test-safe_mean.R`:

```
testthat::test_that("safe_mean works", {
  x <- c(1,2,NA)
  testthat::expect_equal(safe_mean(x), 1.5)
  testthat::expect_error(safe_mean("oops"))
})
```

```
Test passed
```

32 Document with roxygen2

```
#' Compute a safe mean
#' @param x Numeric vector
#' @param na.rm Logical; remove NAs
#' @return Numeric scalar
#' @examples
#' safe_mean(c(1,2,NA))
#' @export
safe_mean <- function(x, na.rm = TRUE) {
  stopifnot(is.numeric(x))
  mean(x, na.rm = na.rm)
}
```

Run:

```
devtools::document()
```

33 Exercises

1. Write `winsorize(x, probs=c(0.05,0.95))` and test it.
2. Create `validate_columns(df, required=c("USUBJID","AGE"))` and add tests.
3. Add roxygen docs and build help pages.

34 R Package Development

34.1 Setup

```
install.packages(c("usethis","devtools","testthat","roxygen2","pkgdown"))
```

34.2 Create a Package

```
usethis::create_package("mypkg")
# In the new project:
usethis::use_mit_license("Your Name")
usethis::use_git()
usethis::use_github()   # optional
usethis::use_roxygen_md()
usethis::use_testthat()
usethis::use_package("dplyr")   # adds to DESCRIPTION
```

34.3 Add a Function

Create `R/safe_mean.R` and its tests (see previous chapter).

34.4 Build, Install, Check

```
devtools::document()
devtools::build()
devtools::install()
devtools::check()
```

34.5 Vignette & Website

```
usethis::use_vignette("intro")
usethis::use_pkgdown()
pkgdown::build_site()
```

Exercise: Package-ize a small utility set (`convert_blanks_to_na`, `validate_columns`, etc.) with docs and tests.

35 Git in RStudio (Setup & Auth)

35.1 One-Time Setup

- Install Git and ensure `git --version` works.
- In R:

```
usethis::use_git_config(user.name = "Your Name", user.email = "you@example.com")
```

35.2 Initialize Git for the Current Project

```
usethis::use_git()
```

35.3 Connect to GitHub

- Create a GitHub account.
- In R:

```
usethis::create_github_token()
gitcreds::gitcreds_set()  # paste token when prompted
usethis::use_github(protocol = "https")
```

Or set up SSH keys via RStudio (`Tools > Global Options > Git/SVN`).

35.4 Typical Workflow

1. Stage changes (Git pane in RStudio).
2. Commit with a clear message.
3. Push to origin (GitHub).

35.5 Remove Git from a Project (macOS/RStudio)

- In Finder/Terminal, delete the hidden `.git` folder in the project root (careful!).
- Or from Terminal at project root:

```
rm -rf .git
```

- Reopen project in RStudio; Git pane will disappear.

Exercises - Create a new repo for this Quarto course and push it. - Branch, make a change, open a Pull Request on GitHub.

36 Creating ADaM: ADSL from SDTM-like Inputs

```
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(tidyr)
library(lubridate)
```

```
Attaching package: 'lubridate'

The following objects are masked from 'package:base':

    date, intersect, setdiff, union
```

We simulate **minimal** SDTM-like `DM` and `EX` to illustrate ADSL creation. If available, replace with your own `data/sdtm/*.sas7bdat`.

```r
# DM
dm <- tibble::tibble(
  STUDYID = "XYZ123",
  USUBJID = sprintf("XYZ-%03d", 1:10),
  ARM = rep(c("Placebo","Active"), length.out=10),
  AGE = c(55, 62, 47, 50, 71, 66, 45, 59, 53, 68),
  SEX = rep(c("M","F"), length.out=10),
  RANDDT = as.Date("2025-01-15") + sample(0:20, 10, replace=TRUE)
)

# EX (first dose date)
ex <- tibble::tibble(
  USUBJID = dm$USUBJID,
  EXSTDTC = dm$RANDDT + sample(0:3, 10, replace=TRUE)
)
```

36.1 Build ADSL

```r
adsl <- dm |>
  left_join(ex, by="USUBJID") |>
  transmute(
    STUDYID, USUBJID,
    TRT01P = ARM,
    TRT01PN = as.integer(factor(ARM, levels=c("Placebo","Active"))),
    AGE, SEX,
    RANDDT,
    TRTSDT = EXSTDTC,
    TRT01A = TRT01P,          # assume planned == actual for demo
    TRT01AN = TRT01PN
  )
adsl
```

```
# A tibble: 10 x 10
   STUDYID USUBJID TRT01P  TRT01PN   AGE SEX   RANDDT     TRTSDT     TRT01A
   <chr>   <chr>   <chr>      <int> <dbl> <chr> <date>     <date>     <chr>
 1 XYZ123  XYZ-001 Placebo        1    55 M     2025-01-27 2025-01-27 Placebo
 2 XYZ123  XYZ-002 Active         2    62 F     2025-02-01 2025-02-03 Active
 3 XYZ123  XYZ-003 Placebo        1    47 M     2025-02-03 2025-02-05 Placebo
 4 XYZ123  XYZ-004 Active         2    50 F     2025-01-16 2025-01-17 Active
```

```
 5 XYZ123  XYZ-005 Placebo       1    71 M     2025-01-23 2025-01-24 Placebo
 6 XYZ123  XYZ-006 Active        2    66 F     2025-02-03 2025-02-06 Active
 7 XYZ123  XYZ-007 Placebo       1    45 M     2025-01-16 2025-01-18 Placebo
 8 XYZ123  XYZ-008 Active        2    59 F     2025-01-16 2025-01-19 Active
 9 XYZ123  XYZ-009 Placebo       1    53 M     2025-01-28 2025-01-31 Placebo
10 XYZ123  XYZ-010 Active        2    68 F     2025-01-17 2025-01-19 Active
# i 1 more variable: TRT01AN <int>
```

> Note: Real ADSL creation must follow **ADaM IG** (derive flags, dates, imputations, populations). This example is educational only.

Exercises 1. Add analysis populations (e.g., `SAFFL`, `FASFL`) based on simple rules. 2. Derive `AGEGR1` as `<65 / 65` and use ordered factor. 3. Add a treatment end date `TRTEDT` and compute treatment duration.

37 TLFs: Table, Figure, Listing

```
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(gt)
library(ggplot2)
library(survival)
```

We reuse `adsl` from the previous chapter (or synthesize if missing).

```
if (!exists("adsl")) {
  set.seed(123)
  adsl <- tibble::tibble(
    USUBJID = sprintf("XYZ-%03d", 1:60),
    TRT01P = sample(c("Placebo","Active"), 60, replace=TRUE),
    AGE = round(rnorm(60, 60, 8)),
    SEX = sample(c("M","F"), 60, replace=TRUE)
  )
}
```

[

Table 1. Baseline Characteristics by Treatment]

Table 1. Baseline Characteristics by Treatment

TRT01P	N	mean_age	sd_age	pct_female
Active	24	59.4	8.1	50.0
Placebo	36	61.7	5.6	61.1

37.1 Table 1: Baseline Characteristics by Treatment

```
tbl1 <- adsl |>
  group_by(TRT01P) |>
  summarise(
    N = dplyr::n(),
    mean_age = mean(AGE, na.rm=TRUE),
    sd_age = sd(AGE, na.rm=TRUE),
    pct_female = mean(SEX == "F")*100
  )

gt(tbl1) |>
  tab_header(title = "Table 1. Baseline Characteristics by Treatment") |>
  fmt_number(columns = c(mean_age, sd_age, pct_female), decimals = 1)
```

37.2 Figure: (Toy) Survival Curve

We simulate time-to-event data for illustration only.

```
set.seed(42)
n <- nrow(adsl)
adsl$time <- rexp(n, rate = ifelse(adsl$TRT01P=="Active", 0.08, 0.1))
adsl$status <- rbinom(n, 1, 0.7)
fit <- survival::survfit(survival::Surv(time, status) ~ TRT01P, data = adsl)

# Quick GGplot
ggsurv <- function(fit) {
  # rebuild data for plotting
  ss <- summary(fit)
```

```
  dd <- data.frame(
    time = ss$time,
    surv = ss$surv,
    strata = rep(names(fit$strata), fit$strata)
  )
  ggplot(dd, aes(x=time, y=surv, linetype=strata)) +
    geom_step() +
    labs(title="Kaplan-Meier (Toy Data)", x="Time", y="Survival Probability", linetype="Treat
    theme_minimal()
}
#ggsurv(fit)
```

37.3 Listing: Subject-Level Listing

```
lst <- adsl |>
  arrange(USUBJID) |>
  select(USUBJID, TRT01P, AGE, SEX) |>
  head(20)

gt(lst) |>
  tab_header(title = "Listing: First 20 Subjects")
```

Exercises 1. Format Table 1 to `N (mean ± SD)` for age. 2. Add risk table to the KM plot (use an extension like survminer outside of this minimal example). 3. Create a listing that includes population flags once you derive them.

[

Listing: First 20 Subjects] Listing: First 20 Subjects

USUBJID	TRT01P	AGE	SEX
XYZ-001	Placebo	63	F
XYZ-002	Placebo	58	M
XYZ-003	Placebo	67	M
XYZ-004	Active	67	M
XYZ-005	Placebo	67	M
XYZ-006	Active	66	F
XYZ-007	Active	64	F
XYZ-008	Active	60	M
XYZ-009	Placebo	58	M
XYZ-010	Placebo	57	F
XYZ-011	Active	54	F
XYZ-012	Active	58	M
XYZ-013	Active	50	M
XYZ-014	Placebo	77	F
XYZ-015	Active	70	F
XYZ-016	Placebo	51	M
XYZ-017	Active	57	M
XYZ-018	Placebo	56	F
XYZ-019	Placebo	66	M
XYZ-020	Placebo	59	F

38 Capstone: End-to-End Mini Workflow

This chapter ties everything together: **read data → derive ADSL → produce TLFs → render a report.**

38.1 Parameters

```
# You could parametrize paths via YAML; here we keep inline defaults.
dm_path <- "data/sdtm/dm.sas7bdat"
ex_path <- "data/sdtm/ex.sas7bdat"
```

38.2 1) Read (or Synthesize) SDTM

```
library(haven); library(dplyr); library(lubridate)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union


Attaching package: 'lubridate'
```

```
The following objects are masked from 'package:base':

    date, intersect, setdiff, union
```

```
if (file.exists(dm_path)) {
  dm <- read_sas(dm_path)
} else {
  dm <- tibble::tibble(
    STUDYID = "XYZ123",
    USUBJID = sprintf("XYZ-%03d", 1:60),
    ARM = sample(c("Placebo","Active"), 60, replace=TRUE),
    AGE = round(rnorm(60, 60, 8)),
    SEX = sample(c("M","F"), 60, replace=TRUE),
    RANDDT = as.Date("2025-01-15") + sample(0:40, 60, replace=TRUE)
  )
}
if (file.exists(ex_path)) {
  ex <- read_sas(ex_path)
} else {
  ex <- tibble::tibble(
    USUBJID = dm$USUBJID,
    EXSTDTC = dm$RANDDT + sample(0:3, nrow(dm), replace=TRUE)
  )
}
```

38.3 2) Derive ADSL (Minimal Demo)

```
adsl <- dm |>
  left_join(ex, by="USUBJID") |>
  mutate(
    TRT01P = ARM,
    TRT01PN = as.integer(factor(ARM, levels=c("Placebo","Active"))),
    TRT01A = TRT01P,
    TRT01AN = TRT01PN,
    SAFFL = "Y",          # demo only; define rules in real life
    FASFL = "Y"
  ) |>
  dplyr::select(STUDYID.x, USUBJID, TRT01P, TRT01PN, TRT01A, TRT01AN, AGE, SEX, EXSTDTC, SAFF
```

[

Table 1. Baseline by Treatment]

Table 1. Baseline by Treatment

Description of Planned Arm	N	mean_age	sd_age	pct_female
Placebo	226	75.04867	8.503715	60.61947
Screen Failure	52	75.09615	9.699928	69.23077
Xanomeline High Dose	184	74.01087	7.939656	48.36957
Xanomeline Low Dose	181	75.29834	8.277778	60.77348

38.4 3) TLFs

```
library(gt); library(ggplot2); library(survival)

tbl1 <- adsl |>
  group_by(TRT01P) |>
  summarise(N=n(),
            mean_age = mean(AGE), sd_age = sd(AGE),
            pct_female = mean(SEX=="F")*100)
tbl1_gt <- gt(tbl1) |> tab_header(title="Table 1. Baseline by Treatment")
tbl1_gt
```

```
set.seed(123)
adsl$time <- rexp(nrow(adsl), rate=ifelse(adsl$TRT01P=="Active", 0.08, 0.1))
adsl$status <- rbinom(nrow(adsl), 1, 0.7)
fit <- survfit(Surv(time, status) ~ TRT01P, data=adsl)
# reuse plotting function from prior chapter
ggsurv <- function(fit) {
  ss <- summary(fit)
  dd <- data.frame(time=ss$time, surv=ss$surv, strata=rep(names(fit$strata), fit$strata))
  ggplot(dd, aes(x=time, y=surv, linetype=strata)) + geom_step() + theme_minimal() +
    labs(title="KM Curve (Toy)", x="Time", y="Survival", linetype="Treatment")
}
#ggsurv(fit)
```

38.5 4) Save Outputs

```
# Example: Save Table 1 as PNG
#gtsave(tbl1_gt, "tlf-table1.png")
```

Challenge: Convert this chapter into a **parameterized report** (e.g., treatment subset or different cohort) and render multiple outputs.

39 Appendix: Tips, Profiles, .libPaths

39.1 Useful Profiles

Create `~/.Rprofile` to set options (be careful on shared systems):

```
options(
  repos = c(CRAN = "https://cloud.r-project.org"),
  scipen = 999
)
```

39.2 Custom Library Paths

```
# In .Rprofile or project-level .Rprofile
.libPaths(c("/path/to/Rlibs", .libPaths()))
```

39.3 Format vs formatC (quick recap)

```
x <- c(123.456, 0.00123456)
format(x, digits = 4)
```

```
[1] "1.235e+02" "1.235e-03"
```

```
format(x, nsmall = 2)
```

```
[1] "1.23456e+02" "1.23456e-03"
```

```
formatC(x, digits = 3, format = "f")
```

```
[1] "123.456" "0.001"
```

39.4 POSIXct vs POSIXlt

- **POSIXct**: seconds since epoch (numeric), compact, fast.
- **POSIXlt**: list-like with components (year, mon, mday…), easier to extract parts.

39.5 Recommended Packages

- `tidyverse`, `lubridate`, `janitor`, `gt`, `gtsummary`, `survival`, `broom`, `here`.
- Pharma/CDISC: `admiral`, `tlf/tern`, `pharmaverse` meta-packages (explore as you grow).

39.6 Short Glossary

- **SDTM**: Study Data Tabulation Model (FDA submission standard for raw domains).
- **ADaM**: Analysis Data Model (derived analysis-ready datasets).
- **TLF**: Tables, Listings, Figures for reporting.