R for Clinical Study Reports and Submission

Yilong Zhang Nan Xiao Keaven Anderson

Contents

Welcome!

Welcome to R for clinical study report and submission. Clinical study reports (CSR) are the essential part of clinical trials development. a CSR is an "integrated" full scientific report of an individual clinical trial.

The ICH E3: strcuture and content of clinical study reports provide guidance to assist sponsors in the development of a CSR. In this book, you will learn how to use R to prepare CSR and submit to regulatory agency.

This is the work-in-progress draft of the book.

0.1 List of Authors and Contributors

The document is maintained by a community. While reading the document, you can be a contributor as well. The quality of this document relies on you.

- Author: contribute majority content of at least one chapter.
- Contributor: contribute at least one commit to the source code

```
data.frame(contributors = as.character(contributors)) %>%
  kable(format = "html") %>%
  kable_styling()
```

Chapter 1

Introduction

1.1 Folder structure

In clinical trial development, developers needs to organize source code to deliver Study Data Tabulation Model (SDTM) or Analysis Dataset Model (ADaM) datasets and table, listing and figures (TLFs). For example, a Phase 3 clinical trial typically require hundreds of TLFs. A consistent and well-defined folder structure is crucial to manage a clinical trial analysis and reporting (A&R) project.

We recommend to use R package folder structure to organize all the A&R related source code and documentation for a clinical trial A&R project. By using the R package folder structure, you can follow the convention to organize your code as other contributors on CRAN. This consistent approach simplifies the communication for all developers within and across organizations.

- For a new R developer, it is an essential step to learn R package when you want to share your work with others. You will learn one folder structure that is widely used in R community with outstanding tutorial and tools for free.
- For an experienced R developer, there is minimal learning curve.
- For an organization, it simplify process, tool, template, and training development, because a unified folder structure is used to develop and maintain standard tool and analysis project.

The workflow around an R package can also improve the tractability and reproducibility for an analysis project (Marwick, Boettiger, and Mullen 2018).

In addition, R package folder structure are also recommended to develop Shiny app as discussed in Chapter 20 of Master Shiny book and Engineering Production-Grade Shiny Apps book.

1.2 In this book

This book is an intermediate level book by assuming reader has equipped some R programming and clinical development knowledge. The assumption of each part is as below:

- Part 1, "Delivering TLFs in CSR" provides general information with examples to create table, listing and figures. In this part, we assume readers are an individual contributor of a clinical project with some experience in R programming. We expect readers are familiar with data manipulation in R. Some good reference includes Hands-On Programming with R, R for Data Science and Data Manipulation with R.
- Part 2, "Clinical Trial Project" provides general information with examples to manage a clinical trial A&R project. In this part, we assume a reader is a project lead with experience in R package development. Some good reference includes R packages and The tidyverse style guide.
- Part 3, "eCTD Submission package" provides general information in preparing submission package related to clinical study report (CSR) in electronic Common Technical Document (eCTD). In this part, we assume a reader is a project lead of a clinical project with experience in R package development as well.

1.3 Philosophy

We share the same philosophy described in Section 1.1 of R Packages book and quote here.

- "Anything that can be automated, should be automated."
- "Do as little as possible by hand. Do as much as possible with functions."
- "The goal is to spend your time thinking about what you want to do rather than thinking about the minutiae of package structure."

Part I

Delivering TLFs in CSR

Chapter 2

Overview

2.1 Background

In clinical trial development, it is a critical step to submit information on medicinal products to regulatory agencies. Electronic Common Technical Document (eCTD) has become a worldwide regulatory submission standard format. For example, the United States Food and Drug Administration (US FDA) required new drug applications, biologics license applications must be submitted using eCTD format. Clinical Data Interchange Standards Consortium (CDISC) provided a pilot project following ICH E3 guidance.

Within eCTD, clinical study reports (CSR) are located at module 5. ICH E3 guidance provide compilation of structure and content of clinical study reports.

A typical CSR contain full details of an individual clinical study including clinical and statistical description, presentations, and analyses. A large number of tables and figures are incorporated into the main text and appendices of a CSR. In CDISC pilot project, an example CSR is also provided. If you are interested in more examples of clinical study reports, the European Medicines Agency (EMA) publishes clinical data including clinical study reports submitted by pharmaceutical companies on EMA clincal data website.

Building CSR is a team work with clinicians, medical writers, statisticians and statistical programmers. Here, we focus on the work and deliverables completed by statisticians and statistical programmers. In an organization, A group of statisticians and statistical programmers commonly work together to define, develop, validate and deliver tables, listings and figures (TLFs) required for a CSR. Microsoft Word is widely used to prepare CSR in pharmaceutical industry. Therefore, `rtf`, `doc`, `docx` are commonly used format to deliver TLFs.

In this chapter, our focus is to illustrate how to create tables, listings and figures (TLFs) in RTF format that is commonly used in a CSR. The examples are in

compliance of FDA's Portable Document Format (PDF) Specifications

> FDA's PDF specification is a general reference, each organization may define more specific TLF format requirement that can be different from the examples in this book.

These TLFs are typically used to summarize efficacy and safety statistical analysis results of a drug (or treatment).

2.2 Structure and Content

In the rest of this chapter, we are following the ICH E3 guidance on structure and content of clinical study reports.

In a CSR, most of TLFs are located in

- Section 10: Study patients
- Section 11: Efficacy evaluation
- Section 12: Safety evaluation
- Section 14: Tables, Figures and Graphs referred to but not included in the text
- Section 16: Appendices

2.3 Datasets

We used public available CDISC pilot study data located at CDISC Bitbucket repository.

For simplicity, we have downloaded all these datasets into `adam_data` folder of this project and transfer them from `.xpt` format to `sas7bdat` format.

The dataset structure is following CDISC Analysis Data Model (ADaM).

2.4 Tools

In this part, we mainly use R packages below to illustrate how to deliver TLFs in CSR.

- tidyverse: prepare reporting ready datasets.
- r2rtf: create RTF outputs

2.4.1 tidyverse

`tidyverse` is a collection of R packages to simplify the workflow to manipulate, visualize and analyze data in R. Those R packages share the tidy tools manifesto and easy to use for interactive data analysis.

RStudio provided outstanding cheatsheets, and tutorials for `tidyverse`.

There are also books to introduce tidyverse. We assume reader to have experience in using `tidyverse` in this book.

- The tidyverse cookbook
- R for Data Science

2.4.2 r2rtf

`r2rtf` is an R package to create production ready tables and figures in RTF format. The R package is designed to

- provide simple "verb" functions that correspond to each component of a table, to help you translate data frame to table in RTF file.
- enable pipes (`%>%`).
- only focus on **table format**. Data manipulation and analysis shall be handled by other R packages. (e.g. `tidyverse`)

Before creating an RTF table we need to:

- Figure out table layout.
- Split the layout into small tasks in the form of a computer program.
- Execute the program.

We provided a brief introduction of `r2rtf` and show how to transfer data frames into Table, Listing, and Figure (TLFs).

Other extended examples and features are covered in `r2rtf` package website.

To explore the basic RTF generation verbs in `r2rtf`, we'll use the dataset `r2rtf_adae` saved in the `r2rtf` package. This dataset contain adverse events (AE) information from a clinical trial.

Below is the meaning of relevant variables. More information can be found in help page of the dataset (`?r2rtf_adae`)

In this example we consider three variables:

- USUBJID: Unique Subject Identifier
- TRTA: Actual Treatment
- AEDECOD: Dictionary-Derived Term

```
r2rtf_adae %>%
  select(USUBJID, TRTA, AEDECOD) %>%
  head(4)
```

```
##       USUBJID    TRTA                    AEDECOD
## 1 01-701-1015 Placebo APPLICATION SITE ERYTHEMA
## 2 01-701-1015 Placebo APPLICATION SITE PRURITUS
## 3 01-701-1015 Placebo                 DIARRHOEA
## 4 01-701-1023 Placebo                  ERYTHEMA
```

`dplyr` and `tidyr` packages within `tidyverse` are used for data manipulation to create a data frame that contains all the information we want to add in an RTF table.

```
tbl <- r2rtf_adae %>%
  count(TRTA, AEDECOD) %>%
  pivot_wider(names_from = TRTA, values_from = n, values_fill = 0)

tbl %>% head(4)
```

```
## # A tibble: 4 x 4
##   AEDECOD        Placebo `Xanomeline High Dose` `Xanomeline Low Dose`
##   <chr>            <int>                  <int>                 <int>
## 1 ABDOMINAL PAIN       1                      2                     3
## 2 AGITATION            2                      1                     2
## 3 ALOPECIA             1                      0                     0
## 4 ANXIETY              2                      0                     4
```

Now we have a dataset `tbl` in preparing the final RTF table.

`r2rtf` aims to provide one function for each type of table layout. Commonly used verbs includes:

- `rtf_page`: RTF page information
- `rtf_title`: RTF title information
- `rtf_colheader`: RTF column header information
- `rtf_body`: RTF table body information
- `rtf_footnote`: RTF footnote information
- `rtf_source`: RTF data source information

All these verbs is designed to enables usage of pipes (`%>%`). A full list of all functions can be found in package reference

A minimal example below illustrates how to combine verbs using pipes to create an RTF table.

- `rtf_body` is used to define table body layout.
- `rtf_encode` transfers table layout information into RTF syntax.
- `write_rtf` save RTF encoding into a file with file extension `.rtf`

```
head(tbl) %>%
  rtf_body() %>%                 # Step 1 Add table  attributes
  rtf_encode() %>%               # Step 2 Convert attributes to RTF encode
  write_rtf("tlf/intro-ae1.rtf") # Step 3 Write to a .rtf file
```

AEDECOD	Placebo	Xanomeline High Dose	Xanomeline Low Dose
ABDOMINAL PAIN	1	2	3
AGITATION	2	1	2
ALOPECIA	1	0	0
ANXIETY	2	0	4
APPLICATION SITE DERMATITIS	9	12	15
APPLICATION SITE ERYTHEMA	3	23	20

If we want to adjust the width of each column to provide more space to the first column, this can be achieved by updating `col_rel_width` argument in `rtf_body`.

In this example, the input of `col_rel_width` is a vector with same length for number of columns. This argument defines the relative width of each column within a pre-defined total column width.

In this example, the defined relative width is `3:2:2:2`. Only the ratio of `col_rel_width` is used. Therefore it is equivalent to use `col_rel_width = c(6,4,4,4)` or `col_rel_width = c(1.5,1,1,1)`.

```
head(tbl) %>%
  rtf_body(col_rel_width = c(3, 2, 2, 2)) %>%

  # define relative width
  rtf_encode() %>%
  write_rtf("tlf/intro-ae2.rtf")
```

AEDECOD	Placebo	Xanomeline High Dose	Xanomeline Low Dose
ABDOMINAL PAIN	1	2	3
AGITATION	2	1	2
ALOPECIA	1	0	0
ANXIETY	2	0	4
APPLICATION SITE DERMATITIS	9	12	15
APPLICATION SITE ERYTHEMA	3	23	20

In previous example, we found issue of misaligned column header. We can fix the issue by using `rtf_colheader` function.

In `rtf_colheader`, `colheader` argument is used to provide content of column header. We use `"|"` to separate the columns.

In the example below, `"Adverse Events | Placebo | Xanomeline High Dose | Xanomeline Low Dose"` define a column header with 4 columns.

```
head(tbl) %>%

  rtf_colheader(
    colheader = "Adverse Events | Placebo | Xanomeline High Dose | Xanomeline Low Dose",
    col_rel_width = c(3, 2, 2, 2)
  ) %>%

  rtf_body(col_rel_width = c(3, 2, 2, 2)) %>%

  rtf_encode() %>%

  write_rtf("tlf/intro-ae3.rtf")
```

Adverse Events	Placebo	Xanomeline High Dose	Xanomeline Low Dose
ABDOMINAL PAIN	1	2	3
AGITATION	2	1	2
ALOPECIA	1	0	0
ANXIETY	2	0	4
APPLICATION SITE DERMATITIS	9	12	15
APPLICATION SITE ERYTHEMA	3	23	20

In `rtf_*` functions such as `rtf_body`, `rtf_footnote`, `text_justification` argument is used to align text. Default is `"c"` for center justification. To vary text justification by column, use character vector with length of vector equal to number of columns displayed (e.g., `c("c","l","r")`).

All possible inputs can be found in the table below.

```
r2rtf:::justification()
```

```
##   type      name rtf_code_text rtf_code_row
## 1    l      left          \\ql       \\trql
## 2    c    center          \\qc       \\trqc
## 3    r     right          \\qr       \\trqr
```

```
## 4    d   decimal          \\qj
## 5    j justified          \\qj
```

Below is an example to make first column left aligned and center aligned for the rest.

```
head(tbl) %>%

  rtf_body(text_justification = c("l", "c", "c", "c")) %>%

  rtf_encode() %>%

  write_rtf("tlf/intro-ae5.rtf")
```

AEDECOD	Placebo	Xanomeline High Dose	Xanomeline Low Dose
ABDOMINAL PAIN	1	2	3
AGITATION	2	1	2
ALOPECIA	1	0	0
ANXIETY	2	0	4
APPLICATION SITE DERMATITIS	9	12	15
APPLICATION SITE ERYTHEMA	3	23	20

In `rtf_*` functions such as `rtf_body`, `rtf_footnote`, etc., `border_left`, `border_right`, `border_top`, and `border_bottom` control cell borders.

If we want to remove the top border of `"Adverse Events"` in header, we can change default value `"single"` to `""` in `border_top` argument as showed below.

`r2rtf` support 26 different border types. The details can be found in the r2rtf package website.

In this example, we also demonstrate the possibility to add multiple column headers.

```
head(tbl) %>%

  rtf_colheader(
    colheader = " | Treatment",
    col_rel_width = c(3, 6)
  ) %>%

  rtf_colheader(
    colheader = "Adverse Events | Placebo | Xanomeline High Dose | Xanomeline Low Dose",
    border_top = c("", "single", "single", "single"),
    col_rel_width = c(3, 2, 2, 2)
  ) %>%

  rtf_body(col_rel_width = c(3, 2, 2, 2)) %>%

  rtf_encode() %>%

  write_rtf("tlf/intro-ae7.rtf")
```

Adverse Events	Treatment		
	Placebo	Xanomeline High Dose	Xanomeline Low Dose
ABDOMINAL PAIN	1	2	3
AGITATION	2	1	2
ALOPECIA	1	0	0
ANXIETY	2	0	4
APPLICATION SITE DERMATITIS	9	12	15
APPLICATION SITE ERYTHEMA	3	23	20

In the `r2rtf` R package get started page there are more examples to illustrate how to customize

- title, subtitle
- footnote, data source
- special character
- etc.

Those features will be introduced when we first use them in the rest of chapters.

There are other R packages that can create TLFs in `.rtf` or `.docx` format. Interested reader can refer "Microsoft/LibreOffice Formats" section in CRAN Task View: Reproducible Research for an overview.

Chapter 3

Disposition

Following ICH E3 guidance, we need to summarize accounting of all patients who entered the study, in the section 10.1, disposition of patients of a CSR.

```
library(haven) # Read SAS data
library(dplyr) # Manipulate data
library(tidyr) # Manipulate data
library(r2rtf) # Reporting in RTF format
```

In this chapter, we illustrate how to create a disposition table as below.

Disposition of Patients

	Placebo		Xanomeline Low Dose		Xanomeline High Dose	
	n	(%)	n	(%)	n	(%)
Patients in population	86		84		84	
Completed	58	67.4	25	29.8	27	32.1
Discontinued	28	32.6	59	70.2	57	67.9
Adverse Event	8	9.3	44	52.4	40	47.6
Death	2	2.3	1	1.2	0	0.0
I/E Not Met	1	1.2	0	0.0	2	2.4
Lack of Efficacy	3	3.5	0	0.0	1	1.2
Lost to Follow-up	1	1.2	1	1.2	0	0.0
Physician Decision	1	1.2	0	0.0	2	2.4
Protocol Violation	1	1.2	1	1.2	1	1.2
Sponsor Decision	2	2.3	2	2.4	3	3.6
Withdrew Consent	9	10.5	10	11.9	8	9.5

The first step is to read relevant datasets into R. For disposition table, all the required information is saved in the ADSL dataset. We can use `haven` package to read the dataset.

```
adsl <- read_sas("adam_data/adsl.sas7bdat")
```

We illustrate how to prepare a report data for a simplified disposition of patients table using variables below:

- USUBJID: Unique subject identifier
- TRT01P: Planed treatment
- TRT01PN: Planned treatment encoding
- DISCONFL: Discontinued from study flag

- DCREASCD: Discontinued from study reason

```
adsl %>%
  select(USUBJID, TRT01P, TRT01PN, DISCONFL, DCREASCD) %>%
  head(4)
```

```
## # A tibble: 4 x 5
##   USUBJID     TRT01P                TRT01PN DISCONFL DCREASCD
##   <chr>       <chr>                   <dbl> <chr>    <chr>
## 1 01-701-1015 Placebo                     0 ""       Completed
## 2 01-701-1023 Placebo                     0 "Y"      Adverse Event
## 3 01-701-1028 Xanomeline High Dose       81 ""       Completed
## 4 01-701-1033 Xanomeline Low Dose        54 "Y"      Sponsor Decision
```

In the code below, we calculated patients in the analysis population.

```
n_rand <- adsl %>%
  group_by(TRT01PN) %>%
  summarize(n = n()) %>%
  pivot_wider(
    names_from = TRT01PN,
    names_prefix = "n_",
    values_from = n
  ) %>%
  mutate(row = "Patients in population")
n_rand
```

```
## # A tibble: 1 x 4
##     n_0  n_54  n_81 row
##   <int> <int> <int> <chr>
## 1    86    84    84 Patients in population
```

In the code below, we calculate number and percentage of patients who discontinued the study.

Here we use `formatC()` to customize the numeric value to be 1 decimal with fixed width of 5.

```
n_disc <- adsl %>%
  group_by(TRT01PN) %>%
  summarize(
    n = sum(DISCONFL == "Y"),
    pct = formatC(n / n() * 100,
      digits = 1, format = "f", width = 5
    )
  ) %>%
  pivot_wider(
    names_from = TRT01PN,
    values_from = c(n, pct)
```

```
  ) %>%
  mutate(row = "Discontinued")

n_disc
```

```
## # A tibble: 1 x 7
##     n_0  n_54  n_81 pct_0   pct_54  pct_81  row
##   <int> <int> <int> <chr>   <chr>   <chr>   <chr>
## 1    28    59    57 " 32.6" " 70.2" " 67.9" Discontinued
```

In the code below, we calculate number and percentage of patients who completed/discontinued the study in different reasons.

```
n_reason <- adsl %>%
  group_by(TRT01PN) %>%
  mutate(n_total = n()) %>%
  group_by(TRT01PN, DCREASCD) %>%
  summarize(
    n = n(),
    pct = formatC(n / unique(n_total) * 100,
      digits = 1, format = "f", width = 5
    )
  ) %>%
  pivot_wider(
    id_cols = DCREASCD,
    names_from = TRT01PN,
    values_from = c(n, pct),
    values_fill = list(n = 0, pct = "  0.0")
  ) %>%
  rename(row = DCREASCD)

n_reason
```

```
## # A tibble: 10 x 7
##    row                 n_0  n_54  n_81 pct_0   pct_54  pct_81
##    <chr>             <int> <int> <int> <chr>   <chr>   <chr>
##  1 Adverse Event         8    44    40 "  9.3" " 52.4" " 47.6"
##  2 Completed            58    25    27 " 67.4" " 29.8" " 32.1"
##  3 Death                 2     1     0 "  2.3" "  1.2" "  0.0"
##  4 I/E Not Met           1     0     2 "  1.2" "  0.0" "  2.4"
##  5 Lack of Efficacy      3     0     1 "  3.5" "  0.0" "  1.2"
##  6 Lost to Follow-up     1     1     0 "  1.2" "  1.2" "  0.0"
##  7 Physician Decision    1     0     2 "  1.2" "  0.0" "  2.4"
##  8 Protocol Violation    1     1     1 "  1.2" "  1.2" "  1.2"
##  9 Sponsor Decision      2     2     3 "  2.3" "  2.4" "  3.6"
## 10 Withdrew Consent      9    10     8 " 10.5" " 11.9" "  9.5"
```

In the code below, we calculate number and percentage of patients who complete the study. We split `n_reason` because we need to customize the row order of the table.

```
n_complete <- n_reason %>% filter(row == "Completed")

n_complete
```

```
## # A tibble: 1 x 7
##   row         n_0  n_54  n_81 pct_0   pct_54  pct_81
##   <chr>     <int> <int> <int> <chr>   <chr>   <chr>
## 1 Completed    58    25    27 " 67.4" " 29.8" " 32.1"
```

In the code below, we calculate number and percentage of patients who discontinued the study in different reasons. Here we use `paste0("    ", row)` to add some leading space for individual reasons for study discontinuation.

```
n_reason <- n_reason %>%
  filter(row != "Completed") %>%
  mutate(row = paste0("    ", row))

n_reason
```

```
## # A tibble: 9 x 7
##   row                       n_0  n_54  n_81 pct_0   pct_54  pct_81
##   <chr>                   <int> <int> <int> <chr>   <chr>   <chr>
## 1 "    Adverse Event"         8    44    40 "  9.3" " 52.4" " 47.6"
## 2 "    Death"                 2     1     0 "  2.3" "  1.2" "  0.0"
## 3 "    I/E Not Met"           1     0     2 "  1.2" "  0.0" "  2.4"
## 4 "    Lack of Efficacy"      3     0     1 "  3.5" "  0.0" "  1.2"
## 5 "    Lost to Follow-up"     1     1     0 "  1.2" "  1.2" "  0.0"
## 6 "    Physician Decision"    1     0     2 "  1.2" "  0.0" "  2.4"
## 7 "    Protocol Violation"    1     1     1 "  1.2" "  1.2" "  1.2"
## 8 "    Sponsor Decision"      2     2     3 "  2.3" "  2.4" "  3.6"
## 9 "    Withdrew Consent"      9    10     8 " 10.5" " 11.9" "  9.5"
```

Now we combined individual rows into the whole table for reporting purpose. `tbl_disp` is used as input for `r2rtf` to create final report.

```
tbl_disp <- bind_rows(n_rand, n_complete, n_disc, n_reason) %>%
  select(row, ends_with(c("_0", "_54", "_81")))

tbl_disp
```

```
## # A tibble: 12 x 7
##    row                       n_0 pct_0    n_54 pct_54   n_81 pct_81
##    <chr>                   <int> <chr>   <int> <chr>    <int> <chr>
##  1 "Patients in population"    86  <NA>      84  <NA>      84  <NA>
##  2 "Completed"                 58 " 67.4"    25 " 29.8"    27 " 32.1"
```

```
##  3 "Discontinued"             28 " 32.6"    59 " 70.2"    57 " 67.9"
##  4 "    Adverse Event"         8 "  9.3"    44 " 52.4"    40 " 47.6"
##  5 "    Death"                 2 "  2.3"     1 "  1.2"     0 "  0.0"
##  6 "    I/E Not Met"           1 "  1.2"     0 "  0.0"     2 "  2.4"
##  7 "    Lack of Efficacy"      3 "  3.5"     0 "  0.0"     1 "  1.2"
##  8 "    Lost to Follow-up"     1 "  1.2"     1 "  1.2"     0 "  0.0"
##  9 "    Physician Decision"    1 "  1.2"     0 "  0.0"     2 "  2.4"
## 10 "    Protocol Violation"    1 "  1.2"     1 "  1.2"     1 "  1.2"
## 11 "    Sponsor Decision"      2 "  2.3"     2 "  2.4"     3 "  3.6"
## 12 "    Withdrew Consent"      9 " 10.5"    10 " 11.9"     8 "  9.5"
```

We start to define the format of the output. We highlighted items that is not discussed in previous discussion.

`rtf_title` define table title. We can provide a vector for the title argument. Each value is a separate line. The format can also be controlled by providing a vector input in text format.

```
tbl_disp %>%

  # Table title
  rtf_title("Disposition of Patients") %>%

  # First row of column header
  rtf_colheader(" | Placebo | Xanomeline Low Dose| Xanomeline High Dose",
    col_rel_width = c(3, rep(2, 3))
  ) %>%

  # Second row of column header
  rtf_colheader(" | n | (%) | n | (%) | n | (%)",
    col_rel_width = c(3, rep(c(0.7, 1.3), 3)),
    border_top = c("", rep("single", 6)),
    border_left = c("single", rep(c("single", ""), 3))
  ) %>%

  # Table body
  rtf_body(
    col_rel_width = c(3, rep(c(0.7, 1.3), 3)),
    text_justification = c("l", rep("c", 6)),
    border_left = c("single", rep(c("single", ""), 3))
  ) %>%

  # Encoding RTF syntax
  rtf_encode() %>%

  # Save to a file
  write_rtf("tlf/tbl_disp.rtf")
```

Disposition of Patients

	Placebo		Xanomeline Low Dose		Xanomeline High Dose	
	n	(%)	n	(%)	n	(%)
Patients in population	86		84		84	
Completed	58	67.4	25	29.8	27	32.1
Discontinued	28	32.6	59	70.2	57	67.9
Adverse Event	8	9.3	44	52.4	40	47.6
Death	2	2.3	1	1.2	0	0.0
I/E Not Met	1	1.2	0	0.0	2	2.4
Lack of Efficacy	3	3.5	0	0.0	1	1.2
Lost to Follow-up	1	1.2	1	1.2	0	0.0
Physician Decision	1	1.2	0	0.0	2	2.4
Protocol Violation	1	1.2	1	1.2	1	1.2
Sponsor Decision	2	2.3	2	2.4	3	3.6
Withdrew Consent	9	10.5	10	11.9	8	9.5

Chapter 4

Analysis Population

Following ICH E3 guidance, we need to summarize which patients were included in each efficacy analysis in the section 11.1, data sets analysed of a CSR.

```
library(haven) # Read SAS data
library(dplyr) # Manipulate data
library(tidyr) # Manipulate data
library(r2rtf) # Reporting in RTF format
```

In this chapter, we illustrate how to create a summary table for analysis population in a study.

Participants Accounting in Analysis Population
(All Participants Randomized)

	Placebo n (%)	Xanomeline line Low Dose n (%)	Xanomeline line High Dose n (%)
Participants in Population	86	84	84
Participants included in ITT population	86 (100.0)	84 (100.0)	84 (100.0)
Participants included in efficacy population	79 (91.9)	81 (96.4)	74 (88.1)
Participants included in safety population	86 (100.0)	84 (100.0)	84 (100.0)

The first step is to read relevant datasets into R. For analysis population table, all the required information is saved in the ADSL dataset. We can use `haven` package to read the dataset.

```
adsl <- read_sas("adam_data/adsl.sas7bdat")
```

We illustrate how to prepare a report data for a simplified analysis population table using variables below:

- USUBJID: Unique subject identifier
- ITTFL: Intent-to-treat population flag
- EFFFL: Efficacy population flag
- SAFFL: Safty population flag

```
adsl %>%
  select(USUBJID, ITTFL, EFFFL, SAFFL) %>%
  head(4)
```

```
## # A tibble: 4 x 4
##   USUBJID     ITTFL EFFFL SAFFL
##   <chr>       <chr> <chr> <chr>
## 1 01-701-1015 Y     Y     Y
## 2 01-701-1023 Y     Y     Y
## 3 01-701-1028 Y     Y     Y
## 4 01-701-1033 Y     Y     Y
```

4.1 Helper Functions

Before we write analysis code, let's discuss the possibility to reuse R code by writing some toy helper function.

As discussed in R for data science, "You should consider writing a function whenever you've copied and pasted a block of code more than twice"

In Chapter 3, there are few repeating steps to:

- format percentage using `formatC`
- calculate number and percentage by groups

We create two ad-hoc functions and apply them to all rest of tables.

To format number and percentage, we created a function called `fmt_num`. It is a very simple function that is a wrapper of `formatC`.

```
fmt_num <- function(x, digits, width = digits + 4) {
  formatC(x,
    digits = digits,
    format = "f",
    width = width
  )
}
```

The main reason to create `fmt_num` is to enhance readability of the analysis code.

For example, we can compare the two versions of code to format percentage used in 3 and `fmt_num`.

```
formatC(n / n() * 100,
  digits = 1, format = "f", width = 5
)
```

```
fmt_num(n / n() * 100, digits = 1)
```

To calculate number and percentage of patients by groups, we provide a simple (but not robust) wrapper function, `count_by` using `dplyr` and `tidyr`.

The function can be enhanced in multiple ways, but here we only focus on simplicity and readability. More details about writing R functions can be found in Stat454 course.

```
count_by <- function(data, # Input data set
                     grp, # Group variable
                     var, # Analysis variable
                     var_label = var, # Analysis variable label
                     id = "USUBJID") { # Subject ID variable
  data <- data %>% rename(grp = !!grp, var = !!var, id = !!id)

  left_join(
    count(data, grp, var),
    count(data, grp, name = "tot"),
    by = "grp",
  ) %>%
    mutate(
      pct = fmt_num(100 * n / tot, digits = 1),
      n = fmt_num(n, digits = 0),
      npct = paste0(n, " (", pct, ")")
    ) %>%
    pivot_wider(
      id_cols = var,
      names_from = grp,
      values_from = c(n, pct, npct),
      values_fill = list(n = "0", pct = fmt_num(0, digits = 0))
    ) %>%
    mutate(var_label = var_label)
}
```

By using `count_by`, we can simplify the analysis code as below.

```
count_by(adsl, "TRT01PN", "EFFFL") %>%
  select(- ends_with(c("_54", "_81")))
```

```
## # A tibble: 2 x 5
##   var   n_0    pct_0   npct_0         var_label
##   <chr> <chr>  <chr>   <chr>          <chr>
## 1 N     "   7" "  8.1" "   7 (  8.1)" EFFFL
## 2 Y     "  79" " 91.9" "  79 ( 91.9)" EFFFL
```

4.2 Analysis Code

With the helper functions `count_by`, we can easily prepare report dataset as

```
# Derive a randomization flag
adsl <- adsl %>% mutate(RANDFL = "Y")

pop <- count_by(adsl, "TRT01PN", "RANDFL",
                var_label = "Participants in Population") %>%
  select(var_label, starts_with("n_"))
```

```
pop1 <- bind_rows(
  count_by(adsl, "TRT01PN", "ITTFL",
    var_label = "Participants included in ITT population"
  ),
  count_by(adsl, "TRT01PN", "EFFFL",
    var_label = "Participants included in efficacy population"
  ),
  count_by(adsl, "TRT01PN", "SAFFL",
    var_label = "Participants included in safety population"
  )
) %>%
  filter(var == "Y") %>%
  select(var_label, starts_with("npct_"))
```

Now we combined individual rows into the whole table for reporting purpose. `tbl_pop` is used as input for `r2rtf` to create final report.

```
names(pop) <- gsub("n_", "npct_", names(pop))
tbl_pop <- bind_rows(pop, pop1)

tbl_pop %>% select(var_label, npct_0)
```

```
## # A tibble: 4 x 2
##   var_label                                  npct_0
##   <chr>                                      <chr>
## 1 Participants in Population                 "  86"
## 2 Participants included in ITT population    "  86 (100.0)"
## 3 Participants included in efficacy population "  79 ( 91.9)"
## 4 Participants included in safety population   "  86 (100.0)"
```

We start to define the format of the output.

```
rel_width <- c(2, rep(1, 3))
colheader <- " | Placebo | Xanomeline line Low Dose| Xanomeline line High Dose"
tbl_pop %>%
  # Table title
  rtf_title(
```

```
    "Participants Accounting in Analysis Population",
    "(All Participants Randomized)"
  ) %>%
  # First row of column header
  rtf_colheader(colheader,
    col_rel_width = rel_width
  ) %>%
  # Second row of column header
  rtf_colheader(" | n (%) | n (%) | n (%)",
    border_top = "",
    col_rel_width = rel_width
  ) %>%
  # Table body
  rtf_body(
    col_rel_width = rel_width,
    text_justification = c("l", rep("c", 3))
  ) %>%
  # Encoding RTF syntax
  rtf_encode() %>%
  # Save to a file
  write_rtf("tlf/tbl_pop.rtf")
```

Participants Accounting in Analysis Population
(All Participants Randomized)

	Placebo n (%)	Xanomeline line Low Dose n (%)	Xanomeline line High Dose n (%)
Participants in Population	86	84	84
Participants included in ITT population	86 (100.0)	84 (100.0)	84 (100.0)
Participants included in efficacy population	79 (91.9)	81 (96.4)	74 (88.1)
Participants included in safety population	86 (100.0)	84 (100.0)	84 (100.0)

Chapter 5

Baseline Characteristics

Following ICH E3 guidance, we need to summarize critical demographic and baseline characteristics of the patients in the section 11.2, demographic and other baseline characteristics of a CSR.

In this chapter, we illustrate how to create a simplified baseline characteristics table in a study.

Participant Baseline Characteristics
(All Participants Randomized)

	Placebo (N=86)	Xanomeline High Dose (N=84)	Xanomeline Low Dose (N=84)	Overall (N=254)
SEX				
Female	53 (61.6%)	40 (47.6%)	50 (59.5%)	143 (56.3%)
Male	33 (38.4%)	44 (52.4%)	34 (40.5%)	111 (43.7%)
Age				
Mean (SD)	75.2 (8.59)	74.4 (7.89)	75.7 (8.29)	75.1 (8.25)
Median [Min, Max]	76.0 [52.0, 89.0]	76.0 [56.0, 88.0]	77.5 [51.0, 88.0]	77.0 [51.0, 89.0]
RACE				
Black or African American	8 (9.3%)	9 (10.7%)	6 (7.1%)	23 (9.1%)
White	78 (90.7%)	74 (88.1%)	78 (92.9%)	230 (90.6%)
American Indian or Alaska Native	0 (0%)	1 (1.2%)	0 (0%)	1 (0.4%)

There is many R packages that can efficiently summarize baseline information. The `table1` R package is one of them.

```
library(table1)
library(r2rtf)
library(haven)
library(dplyr)
library(tidyr)
library(stringr)
library(tools)
```

As in previous chapters, we first read `adsl` dataset that contain all required

information for baseline characteristics table.

```
adsl <- read_sas("adam_data/adsl.sas7bdat")
```

For simplicity, we only analyze `SEX`, `AGE` and `RACE` in this example using the `table1` R package. More details of the `table` R package can be found in the package vignettes

The `table1` R package directly create an HTML report.

```
ana <- adsl %>%
  mutate(SEX = factor(SEX, c("F", "M"), c("Female", "Male")),
         RACE = toTitleCase(tolower(RACE)))

tbl <- table1(~ SEX + AGE + RACE | TRT01P, data = ana)
tbl
```

	Placebo	Xanomeline High Dose	Xanomeline Low Dose	Overal
	(N=86)	(N=84)	(N=84)	(N=25
SEX				
Female	53 (61.6%)	40 (47.6%)	50 (59.5%)	143 (5
Male	33 (38.4%)	44 (52.4%)	34 (40.5%)	111 (4
Age				
Mean (SD)	75.2 (8.59)	74.4 (7.89)	75.7 (8.29)	75.1 (8
Median [Min, Max]	76.0 [52.0, 89.0]	76.0 [56.0, 88.0]	77.5 [51.0, 88.0]	77.0 [5
RACE				
Black or African American	8 (9.3%)	9 (10.7%)	6 (7.1%)	23 (9.1
White	78 (90.7%)	74 (88.1%)	78 (92.9%)	230 (9
American Indian or Alaska Native	0 (0%)	1 (1.2%)	0 (0%)	1 (0.4%

The code below transfer the output into a dataframe that only contain ASCII character recommended by regulatory agencies. `tbl_base` is used as input for `r2rtf` to create final report.

```
tbl_base <- tbl %>%
              as.data.frame() %>%
              as_tibble() %>%
              mutate( across(everything(),
                      ~ str_replace_all(.x, intToUtf8(160), " ") )
                    )


names(tbl_base) <- str_replace_all(names(tbl_base), intToUtf8(160), " ")
tbl_base
```

```
## # A tibble: 11 x 5
##    ` `                    Placebo        `Xanomeline High D~ `Xanomeline Low D~ Overall
```

```
##    <chr>                <chr>          <chr>              <chr>              <chr>
##  1 ""                   "(N=86)"       "(N=84)"           "(N=84)"           "(N=1
##  2 "SEX"                ""             ""                 ""                 ""
##  3 "  Female"           "53 (61.6%)"   "40 (47.6%)"       "50 (59.5%)"       "143
##  4 "  Male"             "33 (38.4%)"   "44 (52.4%)"       "34 (40.5%)"       "111
##  5 "Age"                ""             ""                 ""                 ""
##  6 "  Mean (SD)"        "75.2 (8.59)"  "74.4 (7.89)"      "75.7 (8.29)"      "75.1
##  7 "  Median [Min, Max]" "76.0 [52.0, ~ "76.0 [56.0, 88.0]" "77.5 [51.0, 88.0~ "77.0
##  8 "RACE"               ""             ""                 ""                 ""
##  9 "  Black or African ~ "8 (9.3%)"     "9 (10.7%)"        "6 (7.1%)"         "23
## 10 "  White"            "78 (90.7%)"   "74 (88.1%)"       "78 (92.9%)"       "230
## 11 "  American Indian o~ "0 (0%)"       "1 (1.2%)"         "0 (0%)"           "1 (0
```

We start to define the format of the output. We highlighted items that is not discussed in previous discussion.

`text_indent_first` and `test_indent_left` are used to control the indent space of text. They are helpful when you need to control the white space of a long sentence. For example, "AMERICAN INDIAN OR ALASKA NATIVE" in the table.

```
colheader1 <- paste(names(tbl_base), collapse = "|")
colheader2 <- paste(tbl_base[1, ], collapse = "|")
rel_width <- c(2.5, rep(1, 4))

tbl_base[-1, ] %>%
      rtf_title("Participant Baseline Characteristics",
                "(All Participants Randomized)") %>%
      rtf_colheader(colheader1,
                    col_rel_width = rel_width) %>%
      rtf_colheader(colheader2,
                    border_top = "",
                    col_rel_width = rel_width) %>%
      rtf_body(col_rel_width = rel_width,
               text_justification = c("l", rep("c", 4)),
               text_indent_first = -240,
               text_indent_left = 180) %>%
      rtf_encode() %>%
      write_rtf("tlf/tlf_base.rtf")
```

Participant Baseline Characteristics
(All Participants Randomized)

	Placebo (N=86)	Xanomeline High Dose (N=84)	Xanomeline Low Dose (N=84)	Overall (N=254)
SEX				
Female	53 (61.6%)	40 (47.6%)	50 (59.5%)	143 (56.3%)
Male	33 (38.4%)	44 (52.4%)	34 (40.5%)	111 (43.7%)
Age				
Mean (SD)	75.2 (8.59)	74.4 (7.89)	75.7 (8.29)	75.1 (8.25)
Median [Min, Max]	76.0 [52.0, 89.0]	76.0 [56.0, 88.0]	77.5 [51.0, 88.0]	77.0 [51.0, 89.0]
RACE				
Black or African American	8 (9.3%)	9 (10.7%)	6 (7.1%)	23 (9.1%)
White	78 (90.7%)	74 (88.1%)	78 (92.9%)	230 (90.6%)
American Indian or Alaska Native	0 (0%)	1 (1.2%)	0 (0%)	1 (0.4%)

Chapter 6

Efficacy

Following ICH E3 guidance, we need to summarize primary and secondary efficacy endpoints. in the section 11.4, efficacy results and tabulations of individual patient data of a CSR.

```
library(haven)   # Read SAS data
library(dplyr)   # Manipulate data
library(tidyr)   # Manipulate data
library(r2rtf)   # Reporting in RTF format
library(emmeans) # LS mean estimation
```

For efficacy analysis, we only analyze change from baseline glucose data at Week 24.

ANCOVA of Change from Baseline Glucose (mmol/L) at Week 24
LOCF
Efficacly Analysis Population

	Baseline		Week 24		Change from Baseline		
Treatment	N	Mean (SD)	N	Mean (SD)	N	Mean (SD)	LS Mean (95% CI)[a]
Placebo	79	5.7 (2.23)	57	5.7 (1.83)	57	-0.1 (2.68)	0.07 (-0.27, 0.41)
Xanomeline Low Dose	79	5.4 (0.95)	26	5.7 (1.26)	26	0.2 (0.82)	-0.11 (-0.45, 0.23)
Xanomeline High Dose	74	5.4 (1.37)	30	6.0 (1.92)	30	0.5 (1.94)	0.40 (0.05, 0.75)

Pairwise Comparison	Difference in LS Mean (95% CI)[a]	p-Value
Xanomeline Low Dose - Placebo	-0.17 (-0.65, 0.30)	0.757
Xanomeline High Dose - Placebo	0.33 (-0.16, 0.82)	0.381

[a]Based on an ANCOVA model after adjusting baseline value. LOCF approach is used to impute missing values.
ANCOVA = Analysis of Covariance, LOCF = Last Observation Carried Forward
CI = Confidence Interval, LS = Least Squares, SD = Standard Deviation

6.1 Analysis Dataset

To prepare analysis dataset, we need both `adsl` and `adlbc` datasets for this analysis.

```
adsl <- read_sas("adam_data/adsl.sas7bdat")
adlb <- read_sas("adam_data/adlbc.sas7bdat")
```

We first define the analysis dataset using efficacy population flag `EFFFL` and all records post baseline (`AVISITN > 1`) and on or before Week 24 (`AVISITN <= 12`).

```
gluc <- adlb %>%
  left_join(adsl %>% select(USUBJID, EFFFL), by = "USUBJID") %>%
  filter(EFFFL == "Y" & PARAMCD == "GLUC") %>%
  arrange(TRTPN) %>%
  mutate(TRTP = factor(TRTP, levels = unique(TRTP)))

ana <- gluc %>%
  filter(AVISITN > 0 & AVISITN <= 24) %>%
  arrange(AVISITN) %>%
  mutate(AVISIT = factor(AVISIT, levels = unique(AVISIT)))
```

Below is the first few records of the analysis dataset.

```
ana %>%
  select(USUBJID, TRTPN, AVISIT, AVAL, BASE, CHG) %>%
  head(4)
```

```
## # A tibble: 4 x 6
##   USUBJID     TRTPN AVISIT              AVAL  BASE     CHG
##   <chr>       <dbl> <fct>              <dbl> <dbl>   <dbl>
## 1 01-701-1015     0 "          Week 2"  4.66  4.72 -0.0555
## 2 01-701-1023     0 "          Week 2"  5.77  5.33  0.444
## 3 01-701-1047     0 "          Week 2"  5.55  5.55  0
## 4 01-701-1118     0 "          Week 2"  4.88  4.05  0.833
```

6.2 Helper Functions

To prepare report dataframe, we created a few helper functions by using the `fmt_num` function defined in Chapter 4.

- Format Estimators

```
fmt_est <- function(.mean,
                    .sd,
                    digits = c(1, 2)) {
  .mean <- fmt_num(.mean, digits[1], width = digits[1] + 4)
  .sd <- fmt_num(.sd, digits[2], width = digits[2] + 3)
  paste0(.mean, " (", .sd, ")")
}
```

- Format Confidence Interval

```
fmt_ci <- function(.est,
                   .lower,
                   .upper,
                   digits = 2,
                   width = digits + 3) {
```

```
  .est <- fmt_num(.est, digits, width)
  .lower <- fmt_num(.lower, digits, width)
  .upper <- fmt_num(.upper, digits, width)
  paste0(.est, " (", .lower, ",", .upper, ")")
}
```

- Format P-Value

```
fmt_pval <- function(.p, digits = 3) {
  scale <- 10^(-1 * digits)
  p_scale <- paste0("<", digits)
  if_else(.p < scale, p_scale, fmt_num(.p, digits = digits))
}
```

6.3 Summary of Observed Data

We first summarize observed data at Baseline and Week 24

```
t11 <- gluc %>%
  filter(AVISITN %in% c(0, 24)) %>%
  group_by(TRTPN, TRTP, AVISITN) %>%
  summarise(
    n = n(),
    mean_sd = fmt_est(mean(AVAL), sd(AVAL))
  ) %>%
  pivot_wider(
    id_cols = c(TRTP, TRTPN),
    names_from = AVISITN,
    values_from = c(n, mean_sd)
  )

t11
```

```
## # A tibble: 3 x 6
## # Groups:   TRTPN, TRTP [3]
##   TRTP                TRTPN   n_0  n_24 mean_sd_0       mean_sd_24
##   <fct>               <dbl> <int> <int> <chr>           <chr>
## 1 Placebo                 0    79    57 "  5.7 ( 2.23)" "  5.7 ( 1.83)"
## 2 Xanomeline Low Dose    54    79    26 "  5.4 ( 0.95)" "  5.7 ( 1.26)"
## 3 Xanomeline High Dose   81    74    30 "  5.4 ( 1.37)" "  6.0 ( 1.92)"
```

We also summarize observed change from baseline glucose at Week 24.

```
t12 <- gluc %>%
  filter(AVISITN %in% 24) %>%
  group_by(TRTPN, AVISITN) %>%
  summarise(
```

```
    n_chg = n(),
    mean_chg = fmt_est(
      mean(CHG, na.rm = TRUE),
      sd(CHG, na.rm = TRUE)
    )
  )

t12
```

```
## # A tibble: 3 x 4
## # Groups:   TRTPN [3]
##   TRTPN AVISITN n_chg mean_chg
##   <dbl>   <dbl> <int> <chr>
## 1     0      24    57 " -0.1 ( 2.68)"
## 2    54      24    26 "  0.2 ( 0.82)"
## 3    81      24    30 "  0.5 ( 1.94)"
```

6.4 Missing Data Imputation

In clinical trials, missing data is inevitable. In this study, there are also missing values in glucose data.

```
count(ana, AVISIT)
```

```
## # A tibble: 8 x 2
##   AVISIT                n
##   <fct>             <int>
## 1 "          Week 2"   229
## 2 "          Week 4"   211
## 3 "          Week 6"   197
## 4 "          Week 8"   187
## 5 "         Week 12"   167
## 6 "         Week 16"   147
## 7 "         Week 20"   126
## 8 "         Week 24"   113
```

For simplicity and illustration purpose, we use the last observation carried forward (LOCF) approach to handle missing data. LOCF approach is a single imputation approach that is **not recommended** in real studies. Interested readers can find more discussion on missing data approaches in the book: The Prevention and Treatment of Missing Data in Clinical Trials.

```
ana_locf <- ana %>%
  group_by(USUBJID) %>%
  mutate(locf = AVISITN == max(AVISITN)) %>%
  filter(locf)
```

6.5 ANCOVA model

We start to analyze the imputed data using ANCOVA model with treatment and baseline glucose as covariates.

```
fit <- lm(CHG ~ BASE + TRTP, data = ana_locf)
summary(fit)
```

```
##
## Call:
## lm(formula = CHG ~ BASE + TRTP, data = ana_locf)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.9907 -0.7195 -0.2367  0.2422  7.0754
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)                3.00836    0.39392   7.637 6.23e-13 ***
## BASE                      -0.53483    0.06267  -8.535 2.06e-15 ***
## TRTPXanomeline Low Dose   -0.17367    0.24421  -0.711    0.478
## TRTPXanomeline High Dose   0.32983    0.24846   1.327    0.186
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.527 on 226 degrees of freedom
##   (2 observations deleted due to missingness)
## Multiple R-squared:  0.2567, Adjusted R-squared:  0.2468
## F-statistic: 26.01 on 3 and 226 DF,  p-value: 1.714e-14
```

Then we use the `emmeans` R package to obtain within and between group least square mean (LS mean)

```
fit_within <- emmeans(fit, "TRTP")
fit_within
```

```
##  TRTP                  emmean    SE  df lower.CL upper.CL
##  Placebo               0.0676 0.172 226   -0.272    0.407
##  Xanomeline Low Dose  -0.1060 0.173 226   -0.447    0.235
##  Xanomeline High Dose  0.3975 0.179 226    0.045    0.750
##
## Confidence level used: 0.95
```

```
t13 <- fit_within %>%
  as_tibble() %>%
  mutate(ls = fmt_ci(emmean, lower.CL, upper.CL)) %>%
  select(TRTP, ls)
t13
```

```
## # A tibble: 3 x 2
##   TRTP                 ls
##   <fct>                <chr>
## 1 Placebo              " 0.07 (-0.27, 0.41)"
## 2 Xanomeline Low Dose  "-0.11 (-0.45, 0.23)"
## 3 Xanomeline High Dose " 0.40 ( 0.05, 0.75)"
```

```
fit_between <- pairs(fit_within, reverse = TRUE)
fit_between
```

```
##  contrast                                  estimate    SE  df t.ratio p.value
##  Xanomeline Low Dose - Placebo               -0.174 0.244 226 -0.711  0.7571
##  Xanomeline High Dose - Placebo               0.330 0.248 226  1.327  0.3814
##  Xanomeline High Dose - Xanomeline Low Dose   0.504 0.249 226  2.024  0.1087
##
## P value adjustment: tukey method for comparing a family of 3 estimates
```

```
t2 <- fit_between %>%
  as_tibble() %>%
  mutate(
    ls = fmt_ci(
      estimate,
      estimate - 1.96 * SE,
      estimate + 1.96 * SE
    ),
    p = fmt_pval(p.value)
  ) %>%
  filter(str_detect(contrast, "- Placebo")) %>%
  select(contrast, ls, p)

t2
```

```
## # A tibble: 2 x 3
##   contrast                      ls                  p
##   <chr>                         <chr>               <chr>
## 1 Xanomeline Low Dose - Placebo "-0.17 (-0.65, 0.30)" "  0.757"
## 2 Xanomeline High Dose - Placebo " 0.33 (-0.16, 0.82)" "  0.381"
```

6.6 Reporting

We combine `t11`, `t12` and `t13` to get the first part of the report data

```
t1 <- cbind(
  t11 %>% ungroup() %>% select(TRTP, ends_with("0"), ends_with("24")),
  t12 %>% ungroup() %>% select(ends_with("chg")),
  t13 %>% ungroup() %>% select(ls)
)
```

```
t1
```

```
##                   TRTP n_0     mean_sd_0 n_24    mean_sd_24 n_chg      mean_chg
## 1              Placebo  79   5.7 ( 2.23)   57   5.7 ( 1.83)    57  -0.1 ( 2.68)
## 2  Xanomeline Low Dose  79   5.4 ( 0.95)   26   5.7 ( 1.26)    26   0.2 ( 0.82)
## 3 Xanomeline High Dose  74   5.4 ( 1.37)   30   6.0 ( 1.92)    30   0.5 ( 1.94)
##                     ls
## 1  0.07 (-0.27, 0.41)
## 2 -0.11 (-0.45, 0.23)
## 3  0.40 ( 0.05, 0.75)
```

Then we use `r2rtf` to prepare the table format for `t1`. We also highlight how to handle special character in this example.

Special characters `^` and `_` are used to define superscript and subscript of text. And `{}` is to define the part that will be impacted. For example, `{^a}` provide a superscript `a` for footnote notation.
`r2rtf` also support most LaTeX characters. Examples can be found in `r2rtf` Get Started Page. The `text_convert` argument in `r2rtf_xxx` functions control whether convert special characters.

```
t1_rtf <- t1 %>%
  data.frame() %>%
  rtf_title(c(
    "ANCOVA of Change from Baseline Glucose (mmol/L) at Week 24",
    "LOCF",
    "Efficacly Analysis Population"
  )) %>%
  rtf_colheader("| Baseline | Week 24 | Change from Baseline",
    col_rel_width = c(2.5, 2, 2, 4)
  ) %>%
  rtf_colheader(paste(
    "Treatment |",
    paste0(rep("N | Mean (SD) | ", 3), collapse = ""),
    "LS Mean (95% CI){^a}"
  ),
  col_rel_width = c(2.5, rep(c(0.5, 1.5), 3), 2)
  ) %>%
  rtf_body(text_justification = c("l", rep("c", 7)),
           col_rel_width = c(2.5, rep(c(0.5, 1.5), 3), 2)) %>%
  rtf_footnote(c(
    "{^a}Based on an ANCOVA model after adjusting baseline value. LOCF approach is used
    "ANCOVA = Analysis of Covariance, LOCF = Last Observation Carried Forward",
    "CI = Confidence Interval, LS = Least Squares, SD = Standard Deviation"
  ))

t1_rtf %>%
```

```
  rtf_encode() %>%
  write_rtf("tlf/tlf_eff1.rtf")
```

ANCOVA of Change from Baseline Glucose (mmol/L) at Week 24
LOCF
Efficacly Analysis Population

	Baseline		Week 24		Change from Baseline		
Treatment	N	Mean (SD)	N	Mean (SD)	N	Mean (SD)	LS Mean (95% CI)[a]
Placebo	79	5.7 (2.23)	57	5.7 (1.83)	57	-0.1 (2.68)	0.07 (-0.27, 0.41)
Xanomeline Low Dose	79	5.4 (0.95)	26	5.7 (1.26)	26	0.2 (0.82)	-0.11 (-0.45, 0.23)
Xanomeline High Dose	74	5.4 (1.37)	30	6.0 (1.92)	30	0.5 (1.94)	0.40 (0.05, 0.75)

[a]Based on an ANCOVA model after adjusting baseline value. LOCF approach is used to impute missing values.
ANCOVA = Analysis of Covariance, LOCF = Last Observation Carried Forward
CI = Confidence Interval, LS = Least Squares, SD = Standard Deviation

we also use `r2rtf` to prepare the table format for `t2`

```
t2_rtf <- t2 %>%
  data.frame() %>%
  rtf_colheader("Pairwise Comparison | Difference in LS Mean (95% CI){^a} | p-Value",
    col_rel_width = c(4.5, 4, 2)
  ) %>%
  rtf_body(text_justification = c("l", "c", "c"),
           col_rel_width = c(4.5, 4, 2))
```

```
t2_rtf %>%
  rtf_encode() %>%
  write_rtf("tlf/tlf_eff2.rtf")
```

Pairwise Comparison	Difference in LS Mean (95% CI)[a]	p-Value
Xanomeline Low Dose - Placebo	-0.17 (-0.65, 0.30)	0.757
Xanomeline High Dose - Placebo	0.33 (-0.16, 0.82)	0.381

Finally we combine the two parts to get the final table using `r2rtf`. This is achieved by providing a list of `t1_rtf` and `t2_rtf` as input for `rtf_encode`.

```
list(t1_rtf, t2_rtf) %>%
  rtf_encode() %>%
  write_rtf("tlf/tlf_eff.rtf")
```

ANCOVA of Change from Baseline Glucose (mmol/L) at Week 24
LOCF
Efficacly Analysis Population

	Baseline		Week 24		Change from Baseline		
Treatment	N	Mean (SD)	N	Mean (SD)	N	Mean (SD)	LS Mean (95% CI)[a]
Placebo	79	5.7 (2.23)	57	5.7 (1.83)	57	-0.1 (2.68)	0.07 (-0.27, 0.41)
Xanomeline Low Dose	79	5.4 (0.95)	26	5.7 (1.26)	26	0.2 (0.82)	-0.11 (-0.45, 0.23)
Xanomeline High Dose	74	5.4 (1.37)	30	6.0 (1.92)	30	0.5 (1.94)	0.40 (0.05, 0.75)

Pairwise Comparison	Difference in LS Mean (95% CI)[a]	p-Value
Xanomeline Low Dose - Placebo	-0.17 (-0.65, 0.30)	0.757
Xanomeline High Dose - Placebo	0.33 (-0.16, 0.82)	0.381

[a]Based on an ANCOVA model after adjusting baseline value. LOCF approach is used to impute missing values.
ANCOVA = Analysis of Covariance, LOCF = Last Observation Carried Forward
CI = Confidence Interval, LS = Least Squares, SD = Standard Deviation

Chapter 7

AE Summary

Following ICH E3 guidance, we need to summarize which patients were included in each efficacy analysis in the section 12.2, adverse events of a CSR.

```
library(haven) # Read SAS data
library(dplyr) # Manipulate data
library(tidyr) # Manipulate data
library(r2rtf) # Reporting in RTF format
```

In this chapter, we illustrate how to summarize AE information in a study.

Analysis of Adverse Event Summary
(Safety Analysis Population)

	Placebo		Xanomeline Low Dose		Xanomeline High Dose	
	n	(%)	n	(%)	n	(%)
Participants in population	86		84		84	
With one or more adverse events	69	(80.2)	77	(91.7)	79	(94.0)
With drug-related adverse events	44	(51.2)	73	(86.9)	70	(83.3)
With serious adverse events	0	(0.0)	1	(1.2)	2	(2.4)
With serious drug-related adverse events	0	(0.0)	1	(1.2)	1	(1.2)
Who died	2	(2.3)	1	(1.2)	0	(0.0)
Every subject is counted a single time for each applicable row and column.						

The data used to summarize AE information is in `adsl` and `adae` datasets.

```
adsl <- read_sas("adam_data/adsl.sas7bdat")
adae <- read_sas("adam_data/adae.sas7bdat")
```

We first summarize participants in population by treatment group.

```
pop <- adsl %>%
  filter(SAFFL == "Y") %>%
  rename(TRTAN = TRT01AN) %>%
  count(TRTAN, name = "tot")

pop
```

```
## # A tibble: 3 x 2
##   TRTAN   tot
##   <dbl> <int>
## 1     0    86
## 2    54    84
## 3    81    84
```

We transform the data to simplify the analysis of each required AE criteria of interest.

- With one or more adverse events
- With drug-related adverse events
- With serious adverse events
- With serious drug-related adverse events
- Who died"

```
tidy_ae <- adae %>%
  mutate(
    all = SAFFL == "Y",
    drug = AEREL %in% c("POSSIBLE", "PROBABLE"),
    ser = AESER == "Y",
    drug_ser = drug & ser,
    die = AEOUT == "FATAL"
  ) %>%
  select(USUBJID, TRTAN, all, drug, ser, drug_ser, die) %>%
  pivot_longer(cols = c(all, drug, ser, drug_ser, die))

tidy_ae %>% head(4)
```

```
## # A tibble: 4 x 4
##   USUBJID     TRTAN name     value
##   <chr>       <dbl> <chr>    <lgl>
## 1 01-701-1015     0 all      TRUE
## 2 01-701-1015     0 drug     TRUE
## 3 01-701-1015     0 ser      FALSE
## 4 01-701-1015     0 drug_ser FALSE
```

We summarize the number and percentage of patients meet each AE criteria.

```
ana <- tidy_ae %>%
  filter(value == TRUE) %>%
  group_by(TRTAN, name) %>%
  summarise(n = n_distinct(USUBJID)) %>%
  left_join(pop, by = "TRTAN") %>%
  mutate(
    pct = fmt_num(n / tot * 100, digits = 1),
    n = fmt_num(n, digits = 0),
```

```
    pct = paste0("(", pct, ")")
  )

ana %>% head(4)
```

```
## # A tibble: 4 x 5
## # Groups:   TRTAN [2]
##   TRTAN name  n         tot pct
##   <dbl> <chr> <chr>  <int> <chr>
## 1     0 all   "  69"    86 ( 80.2)
## 2     0 die   "   2"    86 (  2.3)
## 3     0 drug  "  44"    86 ( 51.2)
## 4    54 all   "  77"    84 ( 91.7)
```

We prepare reporting ready dataset for each AE criteria.

```
t_ae <- ana %>%
  pivot_wider(
    id_cols = "name",
    names_from = TRTAN,
    values_from = c(n, pct),
    values_fill = list(
      n = "   0",
      pct = "(  0.0)"
    )
  )

t_ae <- t_ae %>%
  mutate(name = factor(
    name,
    c("all", "drug", "ser", "drug_ser", "die"),
    c(
      "With one or more adverse events",
      "With drug-related adverse events",
      "With serious adverse events",
      "With serious drug-related adverse events",
      "Who died"
    )
  )) %>%
  arrange(name)
```

We prepare reporting ready dataset for analysis population.

```
t_pop <- adsl %>%
  filter(SAFFL == "Y") %>%
  count_by("TRT01AN", "SAFFL",
    var_label = "Participants in population"
```

```
  ) %>%
  rename(name = var_label) %>%
  select(name, starts_with("n_"))

t_pop
```

```
## # A tibble: 1 x 4
##   name                        n_0    n_54   n_81
##   <chr>                       <chr>  <chr>  <chr>
## 1 Participants in population "  86" "  84" "  84"
```

The final report data is saved in `tbl_ae_summary`

```
tbl_ae_summary <- bind_rows(t_pop, t_ae) %>%
  select(name, ends_with("_0"), ends_with("_54"), ends_with("_81"))

tbl_ae_summary
```

```
## # A tibble: 6 x 7
##   name                                  n_0    pct_0   n_54   pct_54  n_81   pct_81
##   <chr>                                 <chr>  <chr>   <chr>  <chr>   <chr>  <chr>
## 1 Participants in population            "  86" <NA>    "  84" <NA>    "  84" <NA>
## 2 With one or more adverse events       "  69" ( 80.2) "  77" ( 91.7) "  79" ( 94.0)
## 3 With drug-related adverse events      "  44" ( 51.2) "  73" ( 86.9) "  70" ( 83.3)
## 4 With serious adverse events           "   0" (  0.0) "   1" (  1.2) "   2" (  2.4)
## 5 With serious drug-related adverse events "   0" (  0.0) "   1" (  1.2) "   1" (  1.2)
## 6 Who died                              "   2" (  2.3) "   1" (  1.2) "   0" (  0.0)
```

We start to define the format of the output.

```
tbl_ae_summary %>%
  rtf_title(
    "Analysis of Adverse Event Summary",
    "(Safety Analysis Population)"
  ) %>%
  rtf_colheader(" | Placebo | Xanomeline Low Dose| Xanomeline High Dose",
    col_rel_width = c(3.5, rep(2, 3))
  ) %>%
  rtf_colheader(" | n | (%) | n | (%) | n | (%)",
    col_rel_width = c(3.5, rep(c(0.7, 1.3), 3)),
    border_top = c("", rep("single", 6)),
    border_left = c("single", rep(c("single", ""), 3))
  ) %>%
  rtf_body(
    col_rel_width = c(3.5, rep(c(0.7, 1.3), 3)),
    text_justification = c("l", rep("c", 6)),
    border_left = c("single", rep(c("single", ""), 3))
```

```
) %>%
rtf_footnote("Every subject is counted a single time for each applicable row and colu
rtf_encode() %>%
write_rtf("tlf/tlf_ae_summary.rtf")
```

Analysis of Adverse Event Summary
(Safety Analysis Population)

	Placebo		Xanomeline Low Dose		Xanomeline High Dose	
	n	(%)	n	(%)	n	(%)
Participants in population	86		84		84	
With one or more adverse events	69	(80.2)	77	(91.7)	79	(94.0)
With drug-related adverse events	44	(51.2)	73	(86.9)	70	(83.3)
With serious adverse events	0	(0.0)	1	(1.2)	2	(2.4)
With serious drug-related adverse events	0	(0.0)	1	(1.2)	1	(1.2)
Who died	2	(2.3)	1	(1.2)	0	(0.0)

Every subject is counted a single time for each applicable row and column.

Chapter 8

Specific AE

Following ICH E3 guidance, we need to summarize which patients were included in each efficacy analysis in the section 12.2, adverse events of a CSR.

```
library(haven) # Read SAS data
library(dplyr) # Manipulate data
library(tidyr) # Manipulate data
library(r2rtf) # Reporting in RTF format
```

In this chapter, we illustrate how to summarize simplified specific AE information in a study.

Analysis of Participants With Specific Adverse Events
(Safty Analysis Population)

	Placebo n	Xanomeline Low Dose n	Xanomeline High Dose n
Participants in population	86	84	84
Cardiac Disorders	**13**	**13**	**18**
Atrial Fibrillation	1	1	3
Atrial Flutter	0	1	1
Atrial Hypertrophy	1	0	0
Atrioventricular Block First Degree	1	1	0
Atrioventricular Block Second Degree	2	0	3
Bradycardia	1	0	0
Bundle Branch Block Left	1	0	0
Bundle Branch Block Right	1	1	0
Cardiac Disorder	0	0	1
Cardiac Failure Congestive	1	0	0
Myocardial Infarction	4	2	4
Palpitations	0	2	0
Sinus Arrhythmia	1	0	0
Sinus Bradycardia	2	7	8
Supraventricular Extrasystoles	1	1	1
Supraventricular Tachycardia	0	1	0
Tachycardia	1	0	0
Ventricular Extrasystoles	0	2	1
Ventricular Hypertrophy	1	0	0
Wolff-Parkinson-White Syndrome	0	1	0
Congenital, Familial and Genetic Disorders	**0**	**1**	**2**
Ventricular Septal Defect	0	1	2
Ear and Labyrinth Disorders	**1**	**2**	**1**
Cerumen Impaction	0	1	0
Ear Pain	1	0	0
Tinnitus	0	1	0
Vertigo	0	1	1
Eye Disorders	**4**	**2**	**1**
Conjunctival Haemorrhage	0	1	0
Conjunctivitis	2	0	0

The data used to summarize AE information is in `adsl` and `adae` datasets.

```
adsl <- read_sas("adam_data/adsl.sas7bdat")
adae <- read_sas("adam_data/adae.sas7bdat")
```

For illustration purpose, we only provide count in the simplified table. The percentage of participants for each AE criteria can be calculated as in Chapter 7.

In this way, let's focus on the analysis script for advanced feature for table layout.

- group content: AE can be summarized in multiple nested layer. (e.g. by

system organ class (SOC, `AESOC`) and specific AE term (`AEDECOD`))
- pagenization: there are many AE terms that can not be covered in one page. Column headers and SOC information needs to be repeated in every page.

In the code below, we count number of subjects in each AE term by SOC and treatment group.

```
ana <- adae %>%
  mutate(AESOC = toTitleCase(tolower(AESOC)),
         AEDECOD =  toTitleCase(tolower(AEDECOD)))

t1 <-ana %>%
  group_by(TRTAN, AESOC) %>%
  summarise(n = fmt_num(n_distinct(USUBJID), digits = 0)) %>%
  mutate(AEDECOD = AESOC, order = 0)

t1 %>% head(4)
```

```
## # A tibble: 4 x 5
## # Groups:   TRTAN [1]
##   TRTAN AESOC                       n      AEDECOD                     order
##   <dbl> <chr>                       <chr>  <chr>                       <dbl>
## 1     0 Cardiac Disorders           "  13" Cardiac Disorders              0
## 2     0 Ear and Labyrinth Disorders "   1" Ear and Labyrinth Disorders    0
## 3     0 Eye Disorders               "   4" Eye Disorders                  0
## 4     0 Gastrointestinal Disorders  "  17" Gastrointestinal Disorders     0
```

In the code below, we count number of subjects in each AE term by SOC, AE term, and treatment group.

```
t2 <- ana %>%
  group_by(TRTAN, AESOC, AEDECOD) %>%
  summarise(n = fmt_num(n_distinct(USUBJID), digits = 0)) %>%
  mutate(order = 1)

t2 %>% head(4)
```

```
## # A tibble: 4 x 5
## # Groups:   TRTAN, AESOC [1]
##   TRTAN AESOC             AEDECOD                              n      order
##   <dbl> <chr>             <chr>                                <chr>  <dbl>
## 1     0 Cardiac Disorders Atrial Fibrillation                  "   1"     1
## 2     0 Cardiac Disorders Atrial Hypertrophy                   "   1"     1
## 3     0 Cardiac Disorders Atrioventricular Block First Degree  "   1"     1
## 4     0 Cardiac Disorders Atrioventricular Block Second Degree "   2"     1
```

We prepare reporting data for AE information.

```
t_ae <- bind_rows(t1, t2) %>%
        pivot_wider(id_cols = c(AESOC, order, AEDECOD),
                    names_from = TRTAN,
                    names_prefix = "n_",
                    values_from = n,
                    values_fill = fmt_num(0, digits = 0)) %>%
        arrange(AESOC, order, AEDECOD) %>%
        select(AESOC, AEDECOD, starts_with("n"))

t_ae %>% head(4)
```

```
## # A tibble: 4 x 5
##   AESOC             AEDECOD             n_0    n_54   n_81
##   <chr>             <chr>               <chr>  <chr>  <chr>
## 1 Cardiac Disorders Cardiac Disorders   "  13" "  13" "  18"
## 2 Cardiac Disorders Atrial Fibrillation "   1" "   1" "   3"
## 3 Cardiac Disorders Atrial Flutter      "   0" "   1" "   1"
## 4 Cardiac Disorders Atrial Hypertrophy  "   1" "   0" "   0"
```

We prepare reporting data for analysis population.

```
t_pop <- adsl %>%
  filter(SAFFL == "Y") %>%
  count_by("TRT01AN", "SAFFL",
    var_label = "Participants in population"
  ) %>%
  mutate(
    AESOC = "pop",
    AEDECOD = var_label) %>%
  select(AESOC,AEDECOD, starts_with("n_"))

t_pop
```

```
## # A tibble: 1 x 5
##   AESOC AEDECOD                    n_0    n_54   n_81
##   <chr> <chr>                      <chr>  <chr>  <chr>
## 1 pop   Participants in population "  86" "  84" "  84"
```

The final report data is saved in `tbl_ae_spec`. We also add a blank row between population and AE information in the reporting table.

```
tbl_ae_spec <- bind_rows(t_pop,
                         data.frame(AESOC = "pop"),
                         t_ae) %>%
  mutate(AEDECOD = ifelse(AEDECOD == AESOC ,
                          AEDECOD, paste0("  ", AEDECOD)))
```

```
tbl_ae_spec %>% head(4)
```

```
## # A tibble: 4 x 5
##   AESOC             AEDECOD                        n_0    n_54   n_81
##   <chr>             <chr>                          <chr>  <chr>  <chr>
## 1 pop               "  Participants in population" "  86" "  84" "  84"
## 2 pop                <NA>                           <NA>   <NA>   <NA>
## 3 Cardiac Disorders "Cardiac Disorders"           "  13" "  13" "  18"
## 4 Cardiac Disorders "  Atrial Fibrillation"       "   1" "   1" "   3"
```

We start to define the format of the output.

To obtain the nested layout, we use `page_by` argument in `rtf_body` function. By defining `page_by="AESOC"`, `r2rtf` recognize the variable as a group indicator.

After setting `pageby_row = "first_row"`, the first row is displayed as group header. If a group of information is breaked into multiple page, the group header row is repeated in each page by default.

We can also customize the text format by providing a matrix that have same dimension of the input dataset (i.e. `tbl_ae_spec`). In the code below, we illustrate how to display **bold** text for group headers to highlight the nested structure of the table layout.

```
n_row <- nrow(tbl_ae_spec)
n_col <- ncol(tbl_ae_spec)
id <- tbl_ae_spec$AESOC == tbl_ae_spec$AEDECOD
id <- ifelse(is.na(id), FALSE, id)

text_format   <- ifelse(id, "b", "")
```

More discussion on `page_by`, `group_by` and `subline_by` features can be found in the [`r2rtf` package website](https://merck.github.io/r2rtf/articles/example-sublineby-pageby-groupby.html.

```
tbl_ae_spec %>%
  rtf_title("Analysis of Participants With Specific Adverse Events",
            "(Safty Analysis Population)") %>%

    rtf_colheader(" | Placebo | Xanomeline Low Dose| Xanomeline High Dose",
    col_rel_width = c(3, rep(1, 3))
  ) %>%
  rtf_colheader(" | n |  n | n ",
    border_top = "",
    border_bottom = "single",
    col_rel_width = c(3, rep(1, 3))
  ) %>%
```

```
  rtf_body(
    col_rel_width = c(1, 3, rep(1, 3)),
    text_justification = c("l", "l", rep("c", 3)),
    text_format = matrix(text_format, nrow = n_row, ncol = n_col),
    page_by = "AESOC",
    pageby_row = "first_row") %>%

  rtf_footnote("Every subject is counted a single time for each applicable row and colu

  rtf_encode() %>%

  write_rtf("tlf/tlf_spec_ae.rtf")
```

Analysis of Participants With Specific Adverse Events
(Safty Analysis Population)

	Placebo n	Xanomeline Low Dose n	Xanomeline High Dose n
Participants in population	86	84	84
Cardiac Disorders	**13**	**13**	**18**
Atrial Fibrillation	1	1	3
Atrial Flutter	0	1	1
Atrial Hypertrophy	1	0	0
Atrioventricular Block First Degree	1	1	0
Atrioventricular Block Second Degree	2	0	3
Bradycardia	1	0	0
Bundle Branch Block Left	1	0	0
Bundle Branch Block Right	1	1	0
Cardiac Disorder	0	0	1
Cardiac Failure Congestive	1	0	0
Myocardial Infarction	4	2	4
Palpitations	0	2	0
Sinus Arrhythmia	1	0	0
Sinus Bradycardia	2	7	8
Supraventricular Extrasystoles	1	1	1
Supraventricular Tachycardia	0	1	0
Tachycardia	1	0	0
Ventricular Extrasystoles	0	2	1
Ventricular Hypertrophy	1	0	0
Wolff-Parkinson-White Syndrome	0	1	0
Congenital, Familial and Genetic Disorders	**0**	**1**	**2**
Ventricular Septal Defect	0	1	2
Ear and Labyrinth Disorders	**1**	**2**	**1**
Cerumen Impaction	0	1	0
Ear Pain	1	0	0
Tinnitus	0	1	0
Vertigo	0	1	1
Eye Disorders	**4**	**2**	**1**
Conjunctival Haemorrhage	0	1	0
Conjunctivitis	2	0	0

Marwick, Ben, Carl Boettiger, and Lincoln Mullen. 2018. "Packaging Data Analytical Work Reproducibly Using r (and Friends)." *The American Statistician* 72 (1): 80–88.