# teaching_material_book

Stefan Thoma

`r paste(Sys.time())`

# Table of contents

# Preface

## Who is this for

This tutorial is aimed at data scientists / analysts who want to write their first `teal` application. We expect you to already have some R experience and that you know what you want to achieve with the application. You do not need to know how to create a shiny application. The `teal` package aims to make the creation of (teal-specific) shiny applications relatively intuitive without the need to understand how shiny works exactly. Shiny knowledge is useful once you want to either modify an application, or write your own teal module from scratch.

## What will you learn

## How is it structured

All chapters in this book follow a common structure: First, the learning goals of a chapter is defined. We then go through the step-by-step process of using specific `teal` modules. In each chapter we combine what we learned into one code chunk with which we can create a functional teal application. As every `teal` app requires data, the app created in the introduction will be the fundamental building block of the apps in the subsequent chapters. In the *toy apps* created in each chapter you should be able to understand every line of code. The code for a more developed app (but built for the same purpose) and a deployed instance of that app is linked at the end of each chapter. What you learned in this chapter should give you a good understanding of how the more developed application works.

## Where can you find more

# 1 Introduction

**At the end of this chapter you should be able to**

- understand the structure and utlility of `teal` apps
- understand the utility of `teal` apps
- determine whether `teal` apps are useful for *you*

## 1.1 Structure of teal apps

teal is a shiny-based modular framework. It provides an interactive user interface allowing users to customize tables and visualizations facilitating the exploration and analysis of clinical trial data. Dynamic analysis and display control, data filtering and code reproducibility are included in each module.

teal modules are laid out in three panels.

- The `Left Panel`, referred to as the encoding panel, includes elements used to adjust the output interactively.
- The `Center Panel`, referred to as the output panel, where the tables and visualizations are displayed.
- The `Right Panel`, referred to as the data filter panel, includes elements used to subset the data interactively.

The layout is pre-set within each module. However, user can decide overall tab order and nesting. See **?@fig-layout** for the typical teal app structure.

### 1.1.1 Modules

`teal` modules are modular building blocks for your `teal` app. They implement specific app functionalities, such as outlier exploration, data visualizations, and survival analysis. Although modules can be created from scratch, many `teal` modules have already been released and can be found in the following packages:

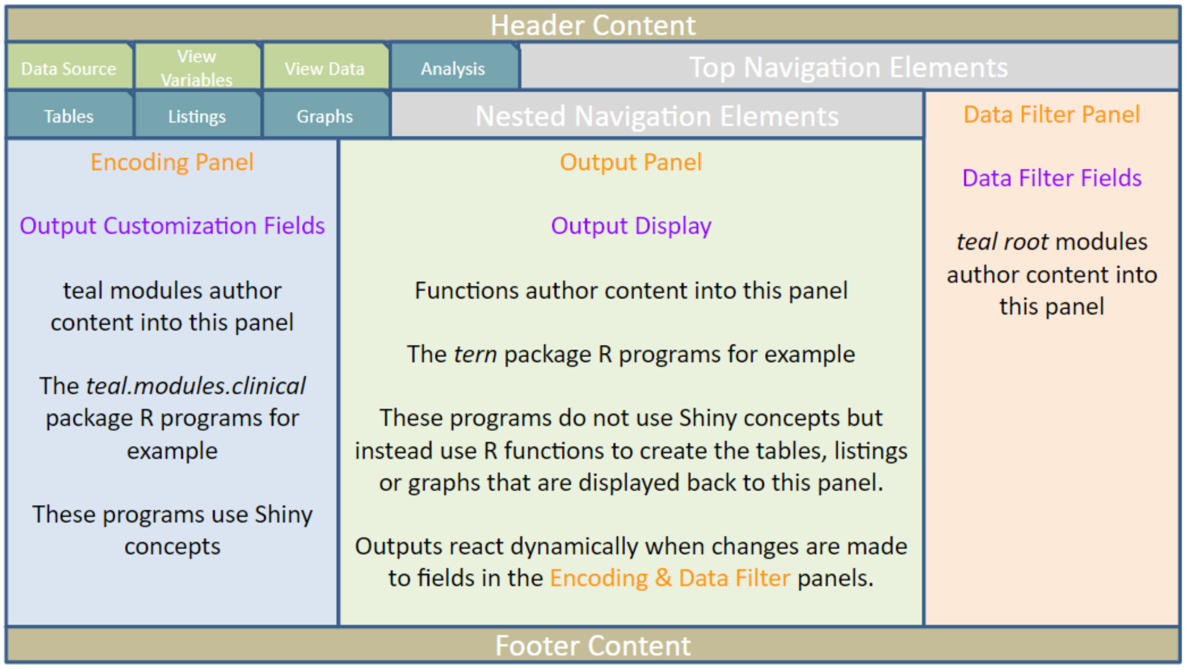

Header Content
Data Source
View Variables
View Data
Analysis
Top Navigation Elements
Tables
Listings
Graphs
Nested Navigation Elements
Data Filter Panel
Data Filter Fields
*teal root* modules author content into this panel
Encoding Panel
Output Customization Fields
teal modules author content into this panel
The *teal.modules.clinical* package R programs for example
These programs use Shiny concepts
Output Panel
Output Display
Functions author content into this panel
The *tern* package R programs for example
These programs do not use Shiny concepts but instead use R functions to create the tables, listings or graphs that are displayed back to this panel.
Outputs react dynamically when changes are made to fields in the Encoding & Data Filter panels.
Footer Content

Figure 1.1: Typical teal app layout

- `teal.modules.general`: general modules for exploring relational/independent/CDISC data
- `teal.modules.clinical`: modules specific to CDISC data and clinical trial reporting
- `teal.modules.hermes`: modules for analyzing MultiAssayExperiment objects

are shiny modules built within the teal framework that specify analysis to be performed.

# 2 Data app

The App created at the end of this chapter will be the basic building block of all subsequent chapters in this book. The structure of the app code will remain consistent throughout the book — make sure you understand each line of code of the *toy* application at the end.

**At the end of this chapter you should be able to**

- understand the code structure of `teal` apps
- understand the workflow to create a `teal` app
- install and load the relevant packages
- write your own app that displays data

## 2.1 Setup

In this tutorial we will be working with realistic looking but simulated data.

```r
if (!require("remotes")) install.packages("remotes")

# install teal
remotes::install_github("insightsengineering/teal@*release")

# install teal.modules.general
remotes::install_github("insightsengineering/teal.modules.general@*release")

# install tern mmrm (required for teal.modules.clinical)
remotes::install_github("insightsengineering/tern.mmrm@*release")

# install teal.modules.clinical
remotes::install_github("insightsengineering/teal.modules.clinical@*release")

# install teal.modules.hermes
remotes::install_github("insightsengineering/teal.modules.hermes@*release")
```

```r
library(random.cdisc.data) # this package provides the data used
library(teal.data)
```

```
Loading required package: shiny
```

```r
library(scda)
```

```
There are no scda.XXXX libraries installed, like scda.2021.
Please install an scda database to take full advantage of the scda package.
```

```r
library(teal.modules.general)
```

```
Loading required package: ggmosaic

Loading required package: ggplot2

Loading required package: shinyTree

Loading required package: teal

Loading required package: teal.transform

Loading required package: magrittr


You are using teal version 0.12.0
```

```r
library(sparkline)
library(teal)
library(tidyverse)
```

```
-- Attaching packages --------------------------------------- tidyverse 1.3.2 --
```

```
v tibble  3.1.8      v dplyr   1.0.10
v tidyr   1.2.1      v stringr 1.4.1
v readr   2.1.3      v forcats 0.5.2
v purrr   0.3.5
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x tidyr::extract()    masks magrittr::extract()
x dplyr::filter()     masks stats::filter()
x dplyr::lag()        masks stats::lag()
x tidyr::replace_na() masks random.cdisc.data::replace_na()
x purrr::set_names()  masks magrittr::set_names()
```

We first load two synthetic data sets. `random.cdisc.data::cadsl` is a Subject Level Analysis Dataset (ADSL) with one record (row) per subject. The unique identifier per subject is stored in the variable `USUBJID`. According to CDISC the main purpose of ADSL is to provide a "(…) source for denominators for populations of interest, stratification variables, and other important subject subgroups".

We can now load the three data-sets:

Data Structure for Adverse Event Analysis

| Data-set name | Explanation | Purpose |
|---|---|---|
| ADSL | Analysis data-set subject level | Filtering |
| ADAE | Analysis data-set for adverse events | Adverse events |
| ADTTE | Analysis data-set for time to event | Efficacy |

```
ADSL <- random.cdisc.data::cadsl
ADAE <- random.cdisc.data::cadae
ADTTE <- random.cdisc.data::cadtte
```

Let's look at the first three rows of the data-frames.

```
map(list(ADSL, ADAE, ADTTE), head, 3)
```

```
[[1]]
# A tibble: 3 x 56
  STUDYID USUBJID     SUBJID SITEID   AGE AGEU  SEX   RACE  ETHNIC COUNTRY DTHFL
  <chr>   <chr>       <chr>  <chr>  <int> <fct> <fct> <fct> <fct>  <fct>   <fct>
1 AB12345 AB12345-CH~ id-128 CHN-3     32 YEARS M     ASIAN HISPA~ CHN     Y
2 AB12345 AB12345-CH~ id-262 CHN-15    35 YEARS M     BLAC~ NOT H~ CHN     N
3 AB12345 AB12345-RU~ id-378 RUS-3     30 YEARS F     ASIAN NOT H~ RUS     N
```

```
# ... with 45 more variables: INVID <chr>, INVNAM <chr>, ARM <fct>,
#   ARMCD <fct>, ACTARM <fct>, ACTARMCD <fct>, TRT01P <fct>, TRT01A <fct>,
#   TRT02P <fct>, TRT02A <fct>, REGION1 <fct>, STRATA1 <fct>, STRATA2 <fct>,
#   BMRKR1 <dbl>, BMRKR2 <fct>, ITTFL <fct>, SAFFL <fct>, BMEASIFL <fct>,
#   BEP01FL <fct>, AEWITHFL <fct>, RANDDT <date>, TRTSDTM <dttm>,
#   TRTEDTM <dttm>, TRT01SDTM <dttm>, TRT01EDTM <dttm>, TRT02SDTM <dttm>,
#   TRT02EDTM <dttm>, AP01SDTM <dttm>, AP01EDTM <dttm>, AP02SDTM <dttm>, ...

[[2]]
# A tibble: 3 x 92
  STUDYID USUBJID     SUBJID SITEID   AGE AGEU  SEX   RACE  ETHNIC COUNTRY DTHFL
  <chr>   <chr>       <chr>  <chr>  <int> <fct> <fct> <fct> <fct>  <fct>   <fct>
1 AB12345 AB12345-BR~ id-134 BRA-1     47 YEARS M     WHITE NOT H~ BRA     Y
2 AB12345 AB12345-BR~ id-134 BRA-1     47 YEARS M     WHITE NOT H~ BRA     Y
3 AB12345 AB12345-BR~ id-134 BRA-1     47 YEARS M     WHITE NOT H~ BRA     Y
# ... with 81 more variables: INVID <chr>, INVNAM <chr>, ARM <fct>,
#   ARMCD <fct>, ACTARM <fct>, ACTARMCD <fct>, TRT01P <fct>, TRT01A <fct>,
#   TRT02P <fct>, TRT02A <fct>, REGION1 <fct>, STRATA1 <fct>, STRATA2 <fct>,
#   BMRKR1 <dbl>, BMRKR2 <fct>, ITTFL <fct>, SAFFL <fct>, BMEASIFL <fct>,
#   BEP01FL <fct>, AEWITHFL <fct>, RANDDT <date>, TRTSDTM <dttm>,
#   TRTEDTM <dttm>, TRT01SDTM <dttm>, TRT01EDTM <dttm>, TRT02SDTM <dttm>,
#   TRT02EDTM <dttm>, AP01SDTM <dttm>, AP01EDTM <dttm>, AP02SDTM <dttm>, ...

[[3]]
# A tibble: 3 x 68
  STUDYID USUBJID     SUBJID SITEID   AGE AGEU  SEX   RACE  ETHNIC COUNTRY DTHFL
  <chr>   <chr>       <chr>  <chr>  <int> <fct> <fct> <fct> <fct>  <fct>   <fct>
1 AB12345 AB12345-BR~ id-105 BRA-1     38 YEARS M     BLAC~ HISPA~ BRA     N
2 AB12345 AB12345-BR~ id-105 BRA-1     38 YEARS M     BLAC~ HISPA~ BRA     N
3 AB12345 AB12345-BR~ id-105 BRA-1     38 YEARS M     BLAC~ HISPA~ BRA     N
# ... with 57 more variables: INVID <chr>, INVNAM <chr>, ARM <fct>,
#   ARMCD <fct>, ACTARM <fct>, ACTARMCD <fct>, TRT01P <fct>, TRT01A <fct>,
#   TRT02P <fct>, TRT02A <fct>, REGION1 <fct>, STRATA1 <fct>, STRATA2 <fct>,
#   BMRKR1 <dbl>, BMRKR2 <fct>, ITTFL <fct>, SAFFL <fct>, BMEASIFL <fct>,
#   BEP01FL <fct>, AEWITHFL <fct>, RANDDT <date>, TRTSDTM <dttm>,
#   TRTEDTM <dttm>, TRT01SDTM <dttm>, TRT01EDTM <dttm>, TRT02SDTM <dttm>,
#   TRT02EDTM <dttm>, AP01SDTM <dttm>, AP01EDTM <dttm>, AP02SDTM <dttm>, ...
```

## 2.2 Your first app

As is generally the case when you start a project, you should have an approximate idea of what you would like to achieve. In this case, the aim is to create an app that visualizes variables of three connected study data-sets.

## 2.3 Code structure

First, you want to define the project metadata. This can include the author name, the study name, the molecule name, etc. then, we have a section where we can define parameters for the app. This can be simple parameter values but they can also be short code snippets that would otherwise often be repeated throughout the application, e.g. pre-selected variables of interest.

After that, we define the app using the `init()` function. This function takes data, modules, and html input and returns a list (usually named `app`) containing the user interface (the `ui` object) and back end (the `server` object) of the application. In the final step we instantiate the app by calling `shinyApp(app$ui, app$server)`.

## 2.4 `init()`

The most crucial function of the `teal` package(s) is the `teal::init()` function, which is structured as follows:

```r
init(
  data,
  modules,
  title = NULL,
  filter = list(),
  header = tags$p("Add Title Here"),
  footer = tags$p("Add Footer Here"),
  id = character(0)
)
```

The `init()` function sets up a shiny app that consists of `teal` modules. Let's go through the arguments:

**`data`**

In the `data` argument we define one or more data-frames for the application. If more than one data-frames are specified, they should be combined as a list, e.g. `data = list(ADSL,` ADTR). For ADaM data-sets, the package `teal.data` provides helper functions with which the `data` argument of `init()` can be specified, — the `cdisc_data()` and the `cdisc_dataset()` functions. They allow the teal app to know the merge key variables of the data-sets. The `cdisc_data`function returns an S6 object.

`keys` are pre-specified variables in ADaM data to merge data-sets. Each type of data-set requires specific keys to be specified. The package `teal.data` automatically chooses the correct key variables based on the `dataname` for the following data-sets: ADSL, ADAE, ADEG, ADTTE, ADAETTE, ADCM, ADEX, ADLB, ADMH, ADQS, ADRS, ADSAFTTE, ADVS, ADDV, ADSUB, ADHY.
If you want to work with other ADaM data-sets you must specify the keys manually. ADTR is a Tumor Results Analysis data-set where there is one record (row) per subject. As of now, this is not part of the default cdisc keys in the `teal.data` package, so we would have to specify them manually:

```r
ADTR <- random.cdisc.data::cadtr


cdisc_data(
  cdisc_dataset("ADSL", ADSL),
  cdisc_dataset("ADTR", ADTR, keys = c("STUDYID", "USUBJID", "PARAMCD", "AVISIT"))
)
```

Note that all keys have to be specified.

pre-processing

Generally, it is recommended to collect all data pre-processing steps in a separate R-script, which can then directly be applied to the ‘`cdisc_data()`‘ output. One such pre-processing steps could be the creation of a new variable in the ADSL data-set which splits patients into two age groups where patients younger than 40 are `AGEGR = 0` and patients older than 40 are `AGEGR = 1`.

```r
# you would write this in a separate file directly, without the writeLines command.
writeLines(
  text = "
ADSL <- ADSL %>%
  dplyr::mutate(AGEGR = if_else(AGE < 40, 0, 1, NA_real_))",
  con = "ADSL_preproc.R"
)
```

The resulting R file looks something like this:

```r
ADSL <- ADSL %>%
  dplyr::mutate(AGEGR = if_else(AGE < 40, 0, 1, NA_real_))
```

```r
ADSL_processed <- ADSL %>%
  cdisc_dataset("ADSL", .) %>%
  mutate_dataset(script = "ADSL_preproc.R")
# check if the new variable exists:
ADSL_processed$get_raw_data()$AGEGR
```

```
  [1] 0 0 0 0 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
 [38] 0 0 1 1 0 0 0 0 0 1 0 0 0 0 1 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0
 [75] 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 1
[112] 0 0 0 1 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 1 0 1 0 0 0 1 1 0 0 0 0
[149] 0 1 1 0 1 0 0 1 0 0 0 1 0 1 0 0 1 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
[186] 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0
[223] 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 1 0 0
[260] 0 1 1 0 0 1 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0
[297] 0 0 0 0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0
[334] 1 0 0 1 1 1 1 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 1
[371] 0 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1
```

In a real world case we would not necessarily want to separately save the pre-processed `ADSL` in our environment. Instead, we would apply the pre-processing within the `data` argument of the `init()` function, e.g., like this:

```r
app <- teal::init(
  data = cdisc_data(
    cdisc_dataset("ADSL", ADSL) %>%
      mutate_dataset(script = "ADSL_preproc.R"),
    cdisc_dataset("ADAE", ADAE),
    cdisc_dataset("ADTTE", ADTTE)
    # <<additional dataset code>>
  )
  # , <<additional init arguments code>>
)
```

### 2.4.1 `modules`

**header**

**footer**

**id**

Combining everything above, our `init()` function looks like this:

```
app <- teal::init(
  data = cdisc_data(
    cdisc_dataset("ADSL", ADSL) %>%
      mutate_dataset(script = "ADSL_preproc.R"),
    cdisc_dataset("ADAE", ADAE),
    cdisc_dataset("ADTTE", ADTTE)
  ),
  header = "My first application",
  footer = "footer"
)
```

## 2.5 Toy app

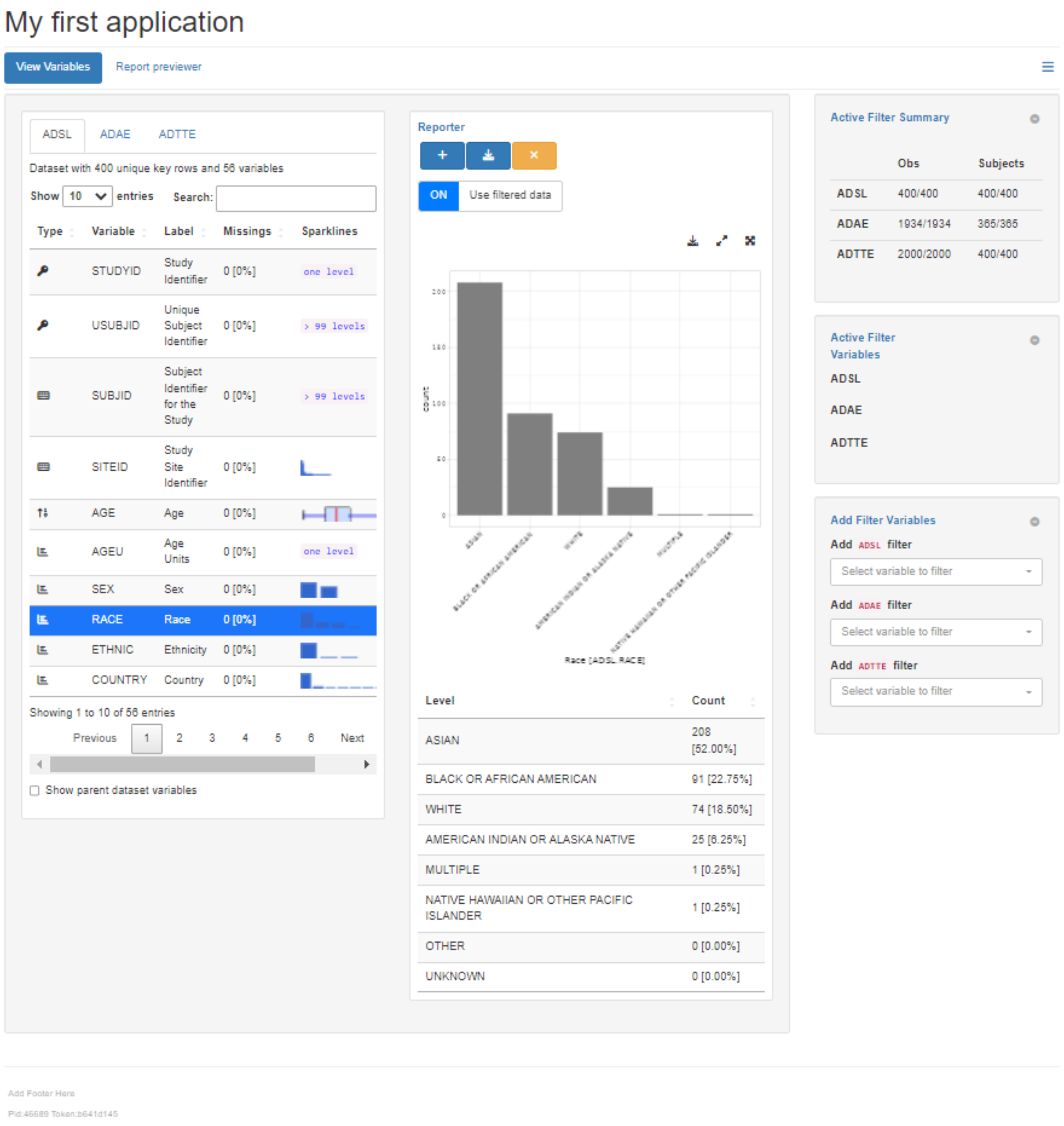
We have now gone through all steps to create your first teal app.

```
# data app
## ---- load packages data app ----
library(random.cdisc.data) # this package provides the data used
library(teal.data)
library(scda)
library(teal.modules.general)
library(sparkline)
library(teal)
library(tidyverse)

## ---- parameter data app ----
author <- "Stefan Thoma"
molecule <- "hexadromedar"


## ---- load data data app ----
ADSL <- random.cdisc.data::cadsl
ADAE <- random.cdisc.data::cadae
ADTTE <- random.cdisc.data::cadtte
```

```
## ---- init data app ----
app <- teal::init(
  data = cdisc_data(
    cdisc_dataset("ADSL", ADSL) %>%
      mutate_dataset(script = "ADSL_preproc.R"),
    cdisc_dataset("ADAE", ADAE),
    cdisc_dataset("ADTTE", ADTTE)
  ),
  header = "My first application",
  footer = "footer"
)


## ---- run data app ----
shinyApp(app$ui, app$server)
?teal.modules.clinical::tm_g_forest_tte
```

The output should look something like this:

## 2.6 Production app



# My first application

Dataset with 400 unique key rows and 56 variables

Show 10 entries Search:

| Type | Variable | Label | Missings | Sparklines |
|---|---|---|---|---|
|  | STUDYID | Study Identifier | 0 [0%] | one level |
|  | USUBJID | Unique Subject Identifier | 0 [0%] | > 99 levels |
|  | SUBJID | Subject Identifier for the Study | 0 [0%] | > 99 levels |
|  | SITEID | Study Site Identifier | 0 [0%] |  |
|  | AGE | Age | 0 [0%] |  |
|  | AGEU | Age Units | 0 [0%] | one level |
|  | SEX | Sex | 0 [0%] |  |
|  | RACE | Race | 0 [0%] |  |
|  | ETHNIC | Ethnicity | 0 [0%] |  |
|  | COUNTRY | Country | 0 [0%] |  |

Showing 1 to 10 of 56 entries

Previous 1 2 3 4 5 6 Next
Show parent dataset variables
Reporter
ON
Use filtered data
count
0
50
100
150
200
ASIAN
BLACK OR AFRICAN AMERICAN
WHITE
AMERICAN INDIAN OR ALASKA NATIVE
MULTIPLE
NATIVE HAWAIIAN OR OTHER PACIFIC ISLANDER
Race [ADSL.RACE]

| Level | Count |
|---|---|
| ASIAN | 208 [52.00%] |
| BLACK OR AFRICAN AMERICAN | 91 [22.75%] |
| WHITE | 74 [18.50%] |
| AMERICAN INDIAN OR ALASKA NATIVE | 25 [6.25%] |
| MULTIPLE | 1 [0.25%] |
| NATIVE HAWAIIAN OR OTHER PACIFIC ISLANDER | 1 [0.25%] |
| OTHER | 0 [0.00%] |
| UNKNOWN | 0 [0.00%] |

Active Filter Summary

|  | Obs | Subjects |
|---|---|---|
| ADSL | 400/400 | 400/400 |
| ADAE | 1934/1934 | 365/365 |
| ADTTE | 2000/2000 | 400/400 |

Active Filter Variables

ADSL

ADAE

ADTTE

Add Filter Variables

Add ADSL filter

Add ADAE filter

Add ADTTE filter

Add Footer Here

Pid:46689 Token:b641d145

Figure 2.1: Data App preview

# 3 efficacy

Stefan Thoma

http://docs.roche.com/#/agile-R/2022_01_28/teal/sample_apps/sample-app-efficacy/

```r
sessionInfo()
```

```
R version 4.2.1 (2022-06-23)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 20.04.5 LTS

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.9.0
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

loaded via a namespace (and not attached):
 [1] compiler_4.2.1  magrittr_2.0.3  fastmap_1.1.0   cli_3.4.1
 [5] tools_4.2.1     htmltools_0.5.3 rstudioapi_0.14 yaml_2.3.6
 [9] stringi_1.7.8   rmarkdown_2.17  knitr_1.40      stringr_1.4.1
[13] xfun_0.34       digest_0.6.30   jsonlite_1.8.2  rlang_1.0.6
[17] evaluate_0.17
```

# 4 Summary

In summary, this book has no content whatsoever.

```
1 + 1
```

```
[1] 2
```

## References