Package 'tidyCDISC'

February 3, 2023

Title Quick Table Generation & Exploratory Analyses on ADaM-Ish Datasets

Version 0.2.0

Description Provides users a quick exploratory dive into common visualizations without writing a single line of code given the users data follows the Analysis Data Model (ADaM) standards put forth by the Clinical Data Interchange Standards Consortium (CDISC) <https://www.cdisc.org>. Prominent modules/ features of the application are the Table Generator, Population Explorer, and the Individual Explorer. The Table Generator allows users to drag and drop variables and desired statistics (frequencies, means, ANOVA, t-test, and other summary statistics) into bins that automagically create stunning tables with validated information. The Population Explorer offers various plots to visualize general trends in the population from various vantage points. Plot modules currently include scatter plot, spaghetti plot, box plot, histogram, means plot, and bar plot. Each plot type allows the user to plot uploaded variables against one another, and dissect the population by filtering out certain subjects. Last, the Individual Explorer establishes a cohesive patient narrative, allowing the user to interact with patient metrics (params) by visit or plotting important patient events on a timeline. All modules allow for concise filtering & downloading bulk outputs into html or pdf formats to save for later.

License AGPL (>= 3)

URL https://Biogen-Inc.github.io/tidyCDISC/

BugReports https://github.com/Biogen-Inc/tidyCDISC/issues

Depends R (>= 2.10)

Imports cicerone, config, dplyr, DT, GGally, ggcorrplot, ggplot2, glue, golem, gt, haven, IDEAFilter, plotly, purrr, rlang, rmarkdown, shiny, shinyjs, shinyWidgets, sjlabelled, stringr, survival, tidyr, timevis, tippy (== 0.1.0)

Suggests knitr, spelling, testthat

VignetteBuilder knitr

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.2.0

NeedsCompilation no

Author Aaron Clark [aut, cre] (<https://orcid.org/0000-0002-0123-0970>),
Jeff Thompson [aut],
Teresa Wilson [aut],
Nate Mockler [ccp, led],
Maya Gans [aut],
Robert Krajcik [ctb],
Marly Gotti [ctb],
Biogen Inc [cph]

Maintainer Aaron Clark <clark.aaronchris@gmail.com>

Repository CRAN

Date/Publication 2023-02-03 17:10:02 UTC

R topics documented:

adae *ADAE*

Description

Adverse Events Analysis Data from PHUSE Test Data Factory Project's GitHub.

Usage

```
adae
```

Format

Data frame with 32,139 features and 34 fields

Source

<https://github.com/phuse-org/TestDataFactory/blob/master/Updated/TDF_ADaM/adae.xpt>, downloaded 2020-06-17

adlbc *ADLBC*

Description

Laboratory Results Chemistry Analysis Data from PHUSE Test Data Factory Project's GitHub.

Usage

```
adlbc
```

Format

Data frame with 32,740 features and 58 fields

Source

<https://github.com/phuse-org/TestDataFactory/blob/master/Updated/TDF_ADaM/adlbc.xpt>, downloaded 2020-06-17

adsl *ADSL*

Description

Subject Level Analysis Data from PHUSE Test Data Factory Project's GitHub.

Usage

```
adsl
```

Format

Data frame with 254 features and 51 fields

Source

<https://github.com/phuse-org/TestDataFactory/blob/master/Updated/TDF_ADaM/adsl.xpt>, downloaded 2020-06-17

adtte *ADTTE*

Description

Time to Event Analysis Data from PHUSE Test Data Factory Project's GitHub.

Usage

```
adtte
```

Format

Data frame with 32,740 features and 58 fields

Source

<https://github.com/phuse-org/TestDataFactory/blob/master/Updated/TDF_ADaM/adtte.xpt>, downloaded 2021-01-26

advs *ADVS*

Description

Vital Signs Analysis Data from PHUSE Test Data Factory Project's GitHub.

Usage

```
advs
```

Format

Data frame with 32,139 features and 34 fields

Source

<https://github.com/phuse-org/TestDataFactory/blob/master/Updated/TDF_ADaM/advs.xpt>, downloaded 2020-06-17

app_methods *Find the proper function to apply to each statistical and column block pairing and use the metadata associated with each column block for the function's arguments*

Description

Find the proper function to apply to each statistical and column block pairing and use the metadata associated with each column block for the function's arguments

Usage

```
app_methods(agg, column, week, group, data, totals, filter = NA)
```

Arguments

agg	the statistic to apply given the block name
column	the column to apply that statistic too, and class of the column dictated by the data frame it came from
week	the week if needed for calculation
group	whether to perform a group_by and if so by which column
data	the dataset to perform all functions on
totals	the totals data frame that contains denominator N's use when calculating column percentages
filter	a string denoting the additional filter to apply to the dataset

Value

the table corresponding to the proper function to perform given the supplied column. This is used within a map to apply to all blocks inside the table generator module.

Examples

```
if(interactive()){
  data(example_dat1, package = "tidyCDISC")

  # Create non-missing table section
  app_methods("NON_MISSING",
              structure("USUBJID", class = c("character", "ADSL")), NA,
              "TRT01P", example_dat1$AE, example_dat1$totals)

  # Create ANOVA table section
  app_methods("ANOVA",
            structure("TEMP", class = c("character", "BDS")), "Week 2",
            "TRT01P", example_dat1$BDS, example_dat1$totals)

  # Create change table section
  app_methods("CHG",
            structure("WEIGHT", class = c("character", "BDS")), "Week 12",
            "TRT01P", example_dat1$BDS, example_dat1$totals)

  # Create mean table section
  app_methods("MEAN",
            structure("PULSE", class = c("character", "BDS")), "Baseline",
            "TRT01P", example_dat1$BDS, example_dat1$totals)
}
```

---

col_for_list_expr *GT Column Names*

Description

The function creates the labels for each column using the total function so the columns are now NAME N= X

Usage

```
col_for_list_expr(col_names, col_total)
```

Arguments

col_names	A vector of column names
col_total	A vector of column totals

Value

A character object of class `from_markdown`.

Examples

```
data(example_dat2, package = "tidyCDISC")

labels <- col_for_list_expr(example_dat2$col_names, example_dat2$col_totals)
labels

if (interactive()) {
# TG table without nice column labels or totals
example_dat2$TG_table

# TG table with nice column labels and totals
gt::cols_label(example_dat2$TG_table, .list = labels)
}
```

`common_rownames` *Identify Names of Columns*

Description

A function to transform the `gt` row names from generics to the column name and the total N of each column

Usage

```
common_rownames(data, group)
```

Arguments

`data`	the data to create columns with
`group`	whether to group the data to calculate Ns

Value

A character vector

Examples

```
data(adsl, package = "tidyCDISC")

# Values of TRT01P
unique(adsl$TRT01P)

# Common row names based on TRT01P
common_rownames(adsl, "TRT01P")
```

data_to_filter *The smallest possible data set we could filter to semi-join later*

Description

The smallest possible data set we could filter to semi-join later

Usage

```
data_to_filter(datafile, input_filter_df)
```

Arguments

datafile	list of ADaM-ish dataframes
input_filter_df	The name of a dataset stored in ‘datafile‘

Value

A ‘data.frame‘ object based on the reduction of ‘datafile‘ from ‘input_filter_df‘.

Examples

```
if(interactive()) {
  datalist <- list(ADSL = tidyCDISC::adsl, ADAE = tidyCDISC::adae,
                   ADVS = tidyCDISC::advs, ADLBC = tidyCDISC::adlbc,
                   ADTTE = tidyCDISC::adtte)

  # Returns combined dataset
  data_to_filter(datalist, c("ADSL", "ADAE"))
}
```

data_to_use_str *Function to clean and combine ADAE dataset with ADSL*

Description

Function to clean and combine ADAE dataset with ADSL

Usage

```
data_to_use_str(x, ae_data, bds_data)
```

Arguments

x	string, naming a data.frame.
ae_data	data.frame, of the AE variety
bds_data	data.frame, of the BDS variety

Value

A ‘data.frame‘ object containing data of the AE variety if ‘x == "ADAE"‘ or one of the BDS variety if not.

Examples

```
if(interactive()) {
  datalist <- list(ADSL = tidyCDISC::adsl, ADVS = tidyCDISC::advs,
                   ADAE = tidyCDISC::adae, ADLBC = tidyCDISC::adlbc)

  pre_adsl <- prep_adsl(datalist$ADSL, input_recipe = 'NONE')
  pre_adae <- prep_adae(datalist, pre_adsl$data, 'NONE')
  ae_data <- pre_adae$data
  bds_data <- prep_bds(datalist, ADSL = pre_adsl$data)

  all.equal(data_to_use_str("ADAE", ae_data, bds_data), ae_data)
  all.equal(data_to_use_str("ADSL", ae_data, bds_data), bds_data)
}
```

---

example_dat1 *Example Data Set 1*

Description

Pre-processed data for purposes of demonstrating app_methods.

Usage

```
example_dat1
```

Format

A list with 3 elements:

AE data frame, pre-processed AE dataset

BDS data frame, pre-processed BDS dataset

totals data frame, contains totals by grouping variable for pre-processed data

example_dat2 *Example Data Set 2*

Description

Pre-processed data for the purposes of demonstrating col_for_list_expr.

Usage

```
example_dat2
```

Format

A list with 3 elements:

TG_table data frame, pre-processed `gt` table object with basic column names

col_names vector, the column names

col_totals vector, totals corresponding to each column

get_levels *Get Factor Levels*

Description

Extracts the factor levels of a vector or returns the unique values if the vector is not a factor.

Usage

```
get_levels(x)
```

Arguments

x	a vector

Value

x vector

References

A character vector containing the levels of the factor/vector

Examples

```
data(adae, package = "tidyCDISC")

# Create levels based on VARN
varN_fctr_adae <- varN_fctr_reorder(adae)

# `adae` does not have factor but `varN_fctr_adae` does
levels(adae$RACE)
levels(varN_fctr_adae$RACE)

# `get_levels()` either creates the factor or retrieves it
get_levels(adae$RACE)
get_levels(varN_fctr_adae$RACE)
```

---

prep_adae *Function to pre-filter the ADAE depending on the stan table selected*

Description

Function to pre-filter the ADAE depending on the stan table selected

Usage

```
prep_adae(datafile, ADSL, input_recipe)
```

Arguments

datafile	list of ADaM-ish dataframes
ADSL	an ADSL data.frame
input_recipe	The shiny input that keeps track of the recipe selected

Value

A 'list' containing a 'data.frame' object and character vector specifying the pre-filter applied.

Examples

```
if(interactive()) {
  datalist <- list(ADSL = tidyCDISC::adsl, ADVS = tidyCDISC::advs,
                   ADAE = tidyCDISC::adae, ADLBC = tidyCDISC::adlbc)

  pre_adsl <- prep_adsl(datalist$ADSL, input_recipe = 'NONE')

  # Create AE data set
  prep_adae(datalist, pre_adsl$data, input_recipe = 'NONE')
}
```

prep_adsl *Function to pre-filter the ADSL depending on the stan table selected*

Description

Function to pre-filter the ADSL depending on the stan table selected

Usage

```
prep_adsl(ADSL, input_recipe)
```

Arguments

ADSL	an ADSL data.frame
input_recipe	The shiny input that keeps track of the recipe selected

Value

A ‘list‘ containing a ‘data.frame‘ object and character vector specifying the pre-filter applied.

Examples

```
data(adsl, package = "tidyCDISC")

# Process ADSL data for STAN table

prep_adsl(adsl, input_recipe = 'Table 3: Accounting of Subjects')

# Return ADSL data if no STAN table selected

prep_adsl(adsl, input_recipe = "NONE")
```

prep_bds *Combine BDS Data Frames*

Description

A function to combine all BDS data frames into one large data set.

Usage

```
prep_bds(datafile, ADSL)
```

Arguments

datafile	list of ADaM-ish data frames
ADSL	A data frame which contains the ADSL data

Value

A data frame containing the BDS data bound by rows.

Examples

```
if(interactive()) {
  datalist <- list(ADSL = tidyCDISC::adsl, ADVS = tidyCDISC::advs,
                   ADAE = tidyCDISC::adae, ADLBC = tidyCDISC::adlbc)

  pre_adsl <- prep_adsl(datalist$ADSL, input_recipe = 'NONE')

  prep_bds(datalist, ADSL = pre_adsl$data)
}
```

---

pretty_IDs *Create Pretty IDs for TG Table*

Description

Replaces ugly ID patterns of a stat block with pretty replacements for display purposes (e.g. NON_MISSING becomes Subject Count for those with Non Missing values)

Usage

```
pretty_IDs(ID)
```

Arguments

ID	The ID vector of a TG table

Value

A character vector of pretty IDs.

Examples

```
# List of patterns that can be replaced
patterns <- c("MEAN", "FREQ", "CHG", "Y_FREQ", "MAX_FREQ", "NON_MISSING",
              "NESTED_FREQ_DSC", "NESTED_FREQ_ABC")
IDs <- paste(patterns, "of VAR")

IDs
pretty_IDs(IDs)
```

run_app *Run the Shiny Application*

Description

Run the Shiny Application

Usage

```
run_app(...)
```

Arguments

...	A series of options to be used inside the app.

Value

No return value, called to run the application.

std_footnote *Create Standard Footnotes for TG Table*

Description

Creates a footnote with a source on the left and date run on the right.

Usage

```
std_footnote(data, source)
```

Arguments

data	The ‘gt‘ table object to append the footnote
source	The source of the data in the table

Value

a ‘gt‘ object

tg_gt *Create the gt table object for TG*

Description

A wrapper for other functions to create the ‘gt‘ object from the data

Usage

```
tg_gt(tg_datalist, blockData, total_df, group)
```

Arguments

tg_datalist	A list containing the data frames used to create the table
blockData	The data for the construction of the blocks in the table
total_df	A data frame containing the totals by grouping variable
group	A character denoting the grouping variable

Value

a data.frame containing output polished for presentation in ‘gt‘

varN_fctr_reorder *Re-order Factor Levels by VARN*

Description

Function to that looks for VARN counterparts to any character or factor VAR variables in any dataframe and re-orders there factor levels, taking the lead from VARN’s numeric guide.

Usage

```
varN_fctr_reorder(data)
```

Arguments

data	a dataframe, including one enriched with SAS labels attributes

Value

The data frame after having factor levels re-ordered by VARN

Examples

```
data(adae, package = "tidyCDISC")

varN_fctr_adae <- varN_fctr_reorder(adae)

unique(adae[,c("AGEGR1", "AGEGR1N")])
levels(adae$AGEGR1)
levels(varN_fctr_adae$AGEGR1)

unique(adae[,c("RACE", "RACEN")])
levels(adae$RACE)
levels(varN_fctr_adae$RACE)
```

Index