Package ‘repfun’

October 30, 2025

Title Create TLFs using R functions styled after SAS macros

Version 0.0.0.9000

Description
A clinical reporting toolkit of R functions that are written to mimic the style of SAS reporting macros. The purpose is to generate TLFs (tables, listings and figures) to support clinical trials.

License Apache License (== 2.0)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports arrow,
data.table,
glue,
jsonlite,
rlang,
xportr,
Hmisc,
r2rtf,
haven,
dplyr,
tidyr,
magrittr,
stringr

Depends R (>= 3.5)

Suggests knitr,
rmarkdown,
testthat (>= 3.0.0),
rprojroot,
DT,
kableExtra,
admiral,
tibble,
lubridate,
ggplot2 (>= 4.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

LazyData true

URL https://glowing-adventure-mrjp2gp.pages.github.io

Contents

`adae` *ADaM ADAE*

Description

adae

Usage

```
adae
```

Format

A data frame with 105 columns:

STUDYID	Study Identifier
DOMAIN	Domain Abbreviation
USUBJID	Unique Subject Identifier
AESEQ	Sequence Number
AESPID	Sponsor-Defined Identifier
AETERM	Reported Term for the Adverse Event
AELLT	Lowest Level Term
AELLTCD	Lowest Level Term Code
AEDECOD	Dictionary-Derived Term
AEPTCD	Preferred Term Code
AEHLT	High Level Term
AEHLTCD	High Level Term Code
AEHLGT	High Level Group Term
AEHLGTCD	High Level Group Term Code
AEBODSYS	Body System or Organ Class
AEBDSYCD	Body System or Organ Class Code
AESOC	Primary System Organ Class
AESOCCD	Primary System Organ Class Code
AESEV	Severity/Intensity
AESER	Serious Event
AEACN	Action Taken with Study Treatment
AEREL	Causality
AEOUT	Outcome of Adverse Event
AESCAN	Involves Cancer
AESCONG	Congenital Anomaly or Birth Defect

AESDISAB Persist or Signif Disability/Incapacity

AESDTH Results in Death

AESHOSP Requires or Prolongs Hospitalization

AESLIFE Is Life Threatening

AESOD Occurred with Overdose

AEDTC Date/Time of Collection

AESTDTC Start Date/Time of Adverse Event

AEENDTC End Date/Time of Adverse Event

AESTDY Study Day of Start of Adverse Event

AEENDY Study Day of End of Adverse Event

TRTSDT Date of First Exposure to Treatment

TRTEDT Date of Last Exposure to Treatment

DTHDT Date of Death

EOSDT End of Study Date

ASTDTM Analysis Start Date/Time

ASTDTF Analysis Start Date Imputation Flag

ASTTMF Analysis Start Time Imputation Flag

AENDTM Analysis End Date/Time

AENDTF Analysis End Date Imputation Flag

AENTMF Analysis End Time Imputation Flag

ASTDT Analysis Start Date

AENDT Analysis End Date

ASTDY Analysis Start Relative Day

AENDY Analysis End Relative Day

ADURN Analysis Duration (N)

ADURU Analysis Duration Units

LDOSEDTM End Date/Time of Last Dose

ASEV Analysis Severity/Intensity

AREL Analysis Causality

TRTEMFL Treatment Emergent Analysis Flag

ASEVN Analysis Severity/Intensity (N)

AOCCIFL 1st Max Sev./Int. Occurrence Flag

SUBJID Subject Identifier for the Study

RFSTDTC Subject Reference Start Date/Time

RFENDTC Subject Reference End Date/Time

RFXSTDTC Date/Time of First Study Treatment

RFXENDTC Date/Time of Last Study Treatment

RFICDTC	Date/Time of Informed Consent
RFPENDTC	Date/Time of End of Participation
DTHDTC	Date/Time of Death
DTHFL	Subject Death Flag
SITEID	Study Site Identifier
AGE	Age
AGEU	Age Units
SEX	Sex
RACE	Race
ETHNIC	Ethnicity
ARMCD	Planned Arm Code
ARM	Description of Planned Arm
ACTARMCD	Actual Arm Code
ACTARM	Description of Actual Arm
COUNTRY	Country
DMDTC	Date/Time of Collection
DMDY	Study Day of Collection
TRT01P	Planned Treatment for Period 01
TRT01A	Actual Treatment for Period 01
TRTSDTM	Datetime of First Exposure to Treatment
TRTSTMF	Time of First Exposure Imput. Flag
TRTEDTM	Datetime of Last Exposure to Treatment
TRTETMF	Time of Last Exposure Imput. Flag
TRTDURD	Total Treatment Duration (Days)
SCRFDT	Screen Failure Date
EOSSTT	End of Study Status
FRVDT	Final Retrievel Visit Date
RANDDT	Date of Randomization
DTHDTF	undocumented field
DTHADY	Relative Day of Death
LDDTHELD	Elapsed Days from Last Dose to Death
DTHCAUS	undocumented field
DTHDOM	undocumented field
DTHCGR1	undocumented field
LSTALVDT	Date Last Known Alive
SAFFL	Safety Population Flag
RACEGR1	Pooled Race Group 1

AGEGR1	Pooled Age Group 1
REGION1	Geographic Region 1
LDDTHGR1	Last Dose to Death - Days Elapsed Grp 1
DTH30FL	Death Within 30 Days of Last Trt Flag
DTHA30FL	Death After 30 Days from Last Trt Flag
DTHB30FL	Death Within 30 Days of First Trt Flag

Details

Adverse Events Analysis

Source

Generated from admiral package (template ad_adae.R).

References

None

Examples

```
data("adae")
```

`adsl` *ADaM ADSL*

Description

adsl

Usage

```
adsl
```

Format

A data frame with 54 columns:

STUDYID	Study Identifier
USUBJID	Unique Subject Identifier
SUBJID	Subject Identifier for the Study
RFSTDTC	Subject Reference Start Date/Time
RFENDTC	Subject Reference End Date/Time
RFXSTDTC	Date/Time of First Study Treatment
RFXENDTC	Date/Time of Last Study Treatment

RFICDTC Date/Time of Informed Consent

RFPENDTC Date/Time of End of Participation

DTHDTC Date/Time of Death

DTHFL Subject Death Flag

SITEID Study Site Identifier

AGE Age

AGEU Age Units

SEX Sex

RACE Race

ETHNIC Ethnicity

ARMCD Planned Arm Code

ARM Description of Planned Arm

ACTARMCD Actual Arm Code

ACTARM Description of Actual Arm

COUNTRY Country

DMDTC Date/Time of Collection

DMDY Study Day of Collection

TRT01P Planned Treatment for Period 01

TRT01A Actual Treatment for Period 01

TRTSDTM Datetime of First Exposure to Treatment

TRTSTMF Time of First Exposure Imput. Flag

TRTEDTM Datetime of Last Exposure to Treatment

TRTETMF Time of Last Exposure Imput. Flag

TRTSDT Date of First Exposure to Treatment

TRTEDT Date of Last Exposure to Treatment

TRTDURD Total Treatment Duration (Days)

SCRFDT Screen Failure Date

EOSDT End of Study Date

EOSSTT End of Study Status

FRVDT Final Retrievel Visit Date

RANDDT Date of Randomization

DTHDT Date of Death

DTHDTF Date of Death Imputation Flag

DTHADY Relative Day of Death

LDDTHELD Elapsed Days from Last Dose to Death

DTHCAUS undocumented field

DTHDOM undocumented field

DTHCGR1	undocumented field
LSTALVDT	Date Last Known Alive
SAFFL	Safety Population Flag
RACEGR1	Pooled Race Group 1
AGEGR1	Pooled Age Group 1
REGION1	Geographic Region 1
LDDTHGR1	Last Dose to Death - Days Elapsed Grp 1
DTH30FL	Death Within 30 Days of Last Trt Flag
DTHA30FL	Death After 30 Days from Last Trt Flag
DTHB30FL	Death Within 30 Days of First Trt Flag

Details

Subject Level Analysis

Source

Generated from admiral package (template ad_adsl.R).

References

None

Examples

```
data("adsl")
```

---

advs *ADaM ADVS*

Description

advs

Usage

```
advs
```

Format

A data frame with 105 columns:

STUDYID	Study Identifier
DOMAIN	Domain Abbreviation
USUBJID	Unique Subject Identifier
VSSEQ	Sequence Number
VSTESTCD	Vital Signs Test Short Name
VSTEST	Vital Signs Test Name
VSPOS	Vital Signs Position of Subject
VSORRES	Result or Finding in Original Units
VSORRESU	Original Units
VSSTRESC	Character Result/Finding in Std Format
VSSTRESN	Numeric Result/Finding in Standard Units
VSSTRESU	Standard Units
VSSTAT	Completion Status
VSLOC	Location of Vital Signs Measurement
VSBLFL	Baseline Flag
VISITNUM	Visit Number
VISIT	Visit Name
VISITDY	Planned Study Day of Visit
VSDTC	Date/Time of Measurements
VSDY	Study Day of Vital Signs
VSTPT	Planned Time Point Name
VSTPTNUM	Planned Time Point Number
VSELTM	Planned Elapsed Time from Time Point Ref
VSTPTREF	Time Point Reference
TRTSDT	Date of First Exposure to Treatment
TRTEDT	Date of Last Exposure to Treatment
TRT01A	Actual Treatment for Period 01
TRT01P	Planned Treatment for Period 01
ADT	Analysis Date
ADY	Analysis Relative Day
PARAMCD	Parameter Code
AVAL	Analysis Value
ATPTN	Analysis Timepoint (N)
ATPT	Analysis Timepoint
AVISIT	Analysis Visit

AVISITN	Analysis Visit (N)
DTYPE	Derivation Type
ONTRTFL	On Treatment Record Flag
ANRLO	Analysis Normal Range Lower Limit
ANRHI	Analysis Normal Range Upper Limit
A1LO	Analysis Range 1 Lower Limit
A1HI	Analysis Range 1 Upper Limit
ANRIND	Analysis Reference Range Indicator
BASETYPE	Baseline Type
ABLFL	Baseline Record Flag
BASE	Baseline Value
BNRIND	Baseline Reference Range Indicator
CHG	Change from Baseline
PCHG	Percent Change from Baseline
ANL01FL	Analysis Flag 01
TRTP	Planned Treatment
TRTA	Actual Treatment
ASEQ	Analysis Sequence Number
AVALCAT1	Analysis Value Category 1
AVALCA1N	Analysis Value Category 1 (N)
PARAM	Parameter
PARAMN	Parameter (N)
SUBJID	Subject Identifier for the Study
RFSTDTC	Subject Reference Start Date/Time
RFENDTC	Subject Reference End Date/Time
RFXSTDTC	Date/Time of First Study Treatment
RFXENDTC	Date/Time of Last Study Treatment
RFICDTC	Date/Time of Informed Consent
RFPENDTC	Date/Time of End of Participation
DTHDTC	Date/Time of Death
DTHFL	Subject Death Flag
SITEID	Study Site Identifier
AGE	Age
AGEU	Age Units
SEX	Sex
RACE	Race
ETHNIC	Ethnicity

ARMCD	Planned Arm Code
ARM	Description of Planned Arm
ACTARMCD	Actual Arm Code
ACTARM	Description of Actual Arm
COUNTRY	Country
DMDTC	Date/Time of Collection
DMDY	Study Day of Collection
TRTSDTM	Datetime of First Exposure to Treatment
TRTSTMF	Time of First Exposure Imput. Flag
TRTEDTM	Datetime of Last Exposure to Treatment
TRTETMF	Time of Last Exposure Imput. Flag
TRTDURD	Total Treatment Duration (Days)
SCRFDT	Screen Failure Date
EOSDT	End of Study Date
EOSSTT	End of Study Status
FRVDT	Final Retrievel Visit Date
RANDDT	Date of Randomization
DTHDT	Date of Death
DTHDTF	undocumented field
DTHADY	Relative Day of Death
LDDTHELD	Elapsed Days from Last Dose to Death
DTHCAUS	undocumented field
DTHDOM	undocumented field
DTHCGR1	undocumented field
LSTALVDT	Date Last Known Alive
SAFFL	Safety Population Flag
RACEGR1	Pooled Race Group 1
AGEGR1	Pooled Age Group 1
REGION1	Geographic Region 1
LDDTHGR1	Last Dose to Death - Days Elapsed Grp 1
DTH30FL	Death Within 30 Days of Last Trt Flag
DTHA30FL	Death After 30 Days from Last Trt Flag
DTHB30FL	Death Within 30 Days of First Trt Flag

Details

Vital Signs Analysis

Source

Generated from admiral package (template ad_advs.R).

References

None

Examples

```
data("advs")
```

`ae` *Adverse Events*

Description

An updated SDTM AE dataset that uses the CDISC pilot project

Usage

```
ae
```

Format

A data frame with 35 columns:

STUDYID Study Identifier

DOMAIN Domain Abbreviation

USUBJID Unique Subject Identifier

AESEQ Sequence Number

AESPID Sponsor-Defined Identifier

AETERM Reported Term for the Adverse Event

AELLT Lowest Level Term

AELLTCD Lowest Level Term Code

AEDECOD Dictionary-Derived Term

AEPTCD Preferred Term Code

AEHLT High Level Term

AEHLTCD High Level Term Code

AEHLGT High Level Group Term

AEHLGTCD High Level Group Term Code

AEBODSYS Body System or Organ Class

AEBDSYCD Body System or Organ Class Code

AESOC Primary System Organ Class

AESOCCD	Primary System Organ Class Code
AESEV	Severity/Intensity
AESER	Serious Event
AEACN	Action Taken with Study Treatment
AEREL	Causality
AEOUT	Outcome of Adverse Event
AESCAN	Involves Cancer
AESCONG	Congenital Anomaly or Birth Defect
AESDISAB	Persist or Signif Disability/Incapacity
AESDTH	Results in Death
AESHOSP	Requires or Prolongs Hospitalization
AESLIFE	Is Life Threatening
AESOD	Occurred with Overdose
AEDTC	Date/Time of Collection
AESTDTC	Start Date/Time of Adverse Event
AEENDTC	End Date/Time of Adverse Event
AESTDY	Study Day of Start of Adverse Event
AEENDY	Study Day of End of Adverse Event

Details

Adverse Events

An updated SDTM AE dataset that uses the CDISC pilot project

Author(s)

Gopi Vegesna

Source

Access the source of the Adverse Events dataset.

airquality_4test *Updated airquality for testing.*

Description

airquality_4test

Usage

```
airquality_4test
```

Format

A data frame with 100 observations and 8 columns:

Ozone	Ozone Score
Solar.R	Solar Score
Wind	Wind Score
Temp	Temperature
Month	Month of Year
Day	Day of Month
CharVar1	Dummy Character Variable 1
CharVar2	Dummy Character Variable 2

Details

Updated version of the airquality data frame that includes filled NA values for testing.

Source

Generated from repfun package.

References

None

Examples

```
data("airquality_4test")
```

`airquality_updated` *Updated airquality*

Description

airquality_updated

Usage

```
airquality_updated
```

Format

A data frame with 100 observations and 8 columns:

Ozone	Ozone Score
Solar.R	Solar Score
Wind	Wind Score
Temp	Temperature
Month	Month of Year
Day	Day of Month
CharVar1	Dummy Character Variable 1
CharVar2	Dummy Character Variable 2

Details

Updated version of the airquality data frame that includes NA values.

Source

Generated from repfun package.

References

None

Examples

```
data("airquality_updated")
```

`copydata` *Copy package data to the directory specified.*

Description

Copy all package data into a temporary directory that can be used when running examples.

Usage

```
copydata(p)
```

Arguments

`p`	A path as a string.

Value

No return value, the current working directory is set.

Author(s)

Yongwei Wang, <`yongwei.x.wang@viivhealthcare.com`>
Chris Rook, <`christopher.x.rook@gsk.com`>

Examples

```
library(repfun)
copydata(tempdir())
```

`dm` *Demography*

Description

A SDTM DM dataset from the CDISC pilot project

Usage

```
dm
```

Format

A data frame with 25 columns:

STUDYID	Study Identifier
DOMAIN	Domain Abbreviation
USUBJID	Unique Subject Identifier
SUBJID	Subject Identifier for the Study
RFSTDTC	Subject Reference Start Date/Time
RFENDTC	Subject Reference End Date/Time
RFXSTDTC	Date/Time of First Study Treatment
RFXENDTC	Date/Time of Last Study Treatment
RFICDTC	Date/Time of Informed Consent
RFPENDTC	Date/Time of End of Participation
DTHDTC	Date/Time of Death
DTHFL	Subject Death Flag
SITEID	Study Site Identifier
AGE	Age
AGEU	Age Units
SEX	Sex
RACE	Race
ETHNIC	Ethnicity
ARMCD	Planned Arm Code
ARM	Description of Planned Arm
ACTARMCD	Actual Arm Code
ACTARM	Description of Actual Arm
COUNTRY	Country
DMDTC	Date/Time of Collection
DMDY	Study Day of Collection

Details

Demography

A SDTM DM dataset from the CDISC pilot project

Source

Access the source of the Demography dataset.

`formats` *Data frame of SAS formats*

Description

formats

Usage

```
formats
```

Format

A data frame with 5 columns and 8 observations:

FMTNAME	Name of Format
START	Start Value of Format
END	End Value of Format
LABEL	Label of Format
TYPE	Type of Format

Details

Formats Data Frame (SAS Style)

Source

Generated for repfun package.

References

None

Examples

```
data("formats")
```

`mtcars_w2lbls` *This is the mtcars data frame with labels added for columns mpg and cyl.*

Description

mtcars_w2lbls

Usage

```
mtcars_w2lbls
```

Format

A data frame with 32 observations and 11 columns:

mpg Miles/(US) gallon

cyl Number of cylinders

disp Displacement (cu.in.)

hp Gross horsepower

drat Rear axle ratio

wt Weight (1000 lbs)

qsec 1/4 mile time

vs Engine (0 = V-shaped, 1 = straight)

am Transmission (0 = automatic, 1 = manual)

gear Number of forward gears

carb Number of carburetors

Details

Updated version of the mtcars data frame that includes labels for 2 variables.

Source

Generated from repfun package.

References

None

Examples

```
data("mtcars_w2lbls")
```

`rem_pg_nums` *Remove floating page numbers from RTF files.*

Description

Pass in an RTF file to have floating page numbers removed and result saved to the same file.

Usage

```
rem_pg_nums(infile)
```

Arguments

`infile`	A path to the RTF file as a string.

Value

No return value, the file is modified.

Author(s)

Yongwei Wang, <yongwei.x.wang@viivhealthcare.com>
Chris Rook, <christopher.x.rook@gsk.com>

Examples

```
## Not run:
library(repfun)
repfun::rem_pg_nums('/path/to/an/rtf/file.rtf')

## End(Not run)
```

`rs_setup` *Pass values to setup function and the global environment will be arranged for use reporting tools.*

Description

Pass values to setup function and the global environment will be arranged for use reporting tools.

Usage

```
rs_setup(
  R_DICTION = "./inst/formats",
  R_MACDIRS = NULL,
  R_DDDATA = NULL,
  R_OTHERDATA = paste0("./data"),
  R_INPUTDATA = paste0("./data"),
  R_RAWDATA = paste0("./data"),
  R_SDTMDATA = paste0("./data"),
  R_ADAMDATA = paste0("./data"),
  R_RFMTDIR = paste0("./inst/formats"),
  D_CENTID = "SITEID",
  D_DATADATE = NULL,
  D_DSPLYNUM = 1,
  D_DSPLYTYP = "T",
  D_FONTSIZE = 10,
  D_KEEPPOPVARS = NULL,
  D_OUTFILE = "./inst/t_ru_list_1.rtf",
  D_PGMPTH = "./R/rs_setup.R",
  D_STUDYID = NULL,
  D_STUDY_DESC = NULL,
  D_POP = "ITTEFL",
  D_POPDATA = NULL,
  D_POPLBL = "Intent-to-Treat",
  D_SUBJID = "USUBJID",
  D_SUBPOP = NULL,
  D_SUBSET = NULL,
  D_TITLE1 = NULL,
  D_TITLE2 = NULL,
  D_TITLE3 = NULL,
  D_TITLE4 = NULL,
  D_TITLE5 = NULL,
  D_TITLE6 = NULL,
  D_TITLE7 = NULL,
  D_FOOT1 = NULL,
  D_FOOT2 = NULL,
  D_FOOT3 = NULL,
  D_FOOT4 = NULL,
  D_FOOT5 = NULL,
  D_FOOT6 = NULL,
  D_FOOT7 = NULL,
  D_FOOT8 = NULL,
  D_FOOT9 = NULL,
  D_USERID = Sys.getenv("DOMINO_USER_NAME"),
  D_RTFYN = "N",
  D_DEBUG = 0
)
```

Arguments

`R_DICTION`	Location of reporting dictionaries.
`R_MACDIRS`	List of folders to search for functions when they are invoked.
`R_DDDATA`	Location to write DDDATA reporting data sets.
`R_OTHERDATA`	Location of additional production data sets.
`R_INPUTDATA`	Location of permanent formats data sets and miscellaneous data sets.
`R_RAWDATA`	Location of raw data sets.
`R_SDTMDATA`	Location of SDTM data sets.
`R_ADAMDATA`	Location of ADAM data sets.
`R_RFMTDIR`	Location of format catalogs and corresponding lists.
`D_CENTID`	Variable name for investigational center.
`D_DATADATE`	Date of data sets for use in titles/footnotes.
`D_DSPLYNUM`	Display number for title.
`D_DSPLYTYP`	Type of Display ('T','L','F').
`D_FONTSIZE`	Size of font on output file which is RTF by default (with automated PDF conversion via script).
`D_KEEPPOPVARS`	Variables to keep on the population data set when merging to apply populations and sub-populations.
`D_OUTFILE`	Production location for output TLFs.
`D_PGMPTH`	Path of the driver file that generates current outputs or data sets.
`D_STUDYID`	Protocol ID for the study.
`D_STUDY_DESC`	Description of Study.
`D_POP`	Population variable from ADSL that must equal Y for subjects to be included in the analysis.
`D_POPDATA`	Data set that contains the population to be analyzed.
`D_POPLBL`	Label for population being analyzed which can be used in the TLF header.
`D_SUBJID`	The variable used to uniquely identify a subject in this analysis.
`D_SUBPOP`	Condition to identify the sub-population when applied to ADSL.
`D_SUBSET`	Condition to filter data from incoming source data sets used for this TLF.
`D_TITLE1`	First title text.
`D_TITLE2`	Second title text.
`D_TITLE3`	Third title text.
`D_TITLE4`	Fourth title text.
`D_TITLE5`	Fifth title text.
`D_TITLE6`	Sixth title text.
`D_TITLE7`	Seventh title text.
`D_FOOT1`	First footnote text.

D_FOOT2	Second footnote text.
D_FOOT3	Third footnote text.
D_FOOT4	Fourth footnote text.
D_FOOT5	Fifth footnote text.
D_FOOT6	Sixth footnote text.
D_FOOT7	Seventh footnote text.
D_FOOT8	Eighth footnote text.
D_FOOT9	Ninth footnote text.
D_USERID	Domino user name.
D_RTFYN	Y or N to generate RTf output.
D_DEBUG	Level of debugging to show in log files.

Value

Global variables defined for use with the reporting tools.

Author(s)

Yongwei Wang, <yongwei.x.wang@viivhealthcare.com>
Chris Rook, <christopher.x.rook@gsk.com>

Examples

```
library(repfun)
library(dplyr)
rfenv <- if (exists('rfenv') && is.environment(get('rfenv'))){
             rfenv
         } else {
             rfenv <- new.env(parent = emptyenv())
             rfenv$G_DEBUG <- 0
             rfenv
         }
datdir <- file.path(tempdir(),"datdir"); dir.create(datdir,showWarnings=FALSE)
outdir <- file.path(tempdir(),"outdir"); dir.create(outdir,showWarnings=FALSE)
fmtdir <- file.path(tempdir(),"fmtdir"); dir.create(fmtdir,showWarnings=FALSE)
repfun::copydata(datdir)
repfun::rs_setup(
D_CENTID="SITEID",
D_DATADATE=Sys.Date(),
D_DSPLYNUM=1,
D_DSPLYTYP='T',
D_FONTSIZE=10,
D_FOOT1='1.) Only treatment emergent events related to lipids are displayed.',
D_FOOT2='2.) Subjects are counted once in each body system & preferred term.',
D_KEEPPOPVARS=c('STUDYID','USUBJID','SAFFL'),
D_USERID=Sys.getenv("DOMINO_USER_NAME"),
D_STUDYID='ABCXYZPDQ',
D_POP="SAFFL",
D_POPDATA=repfun::adsl %>% dplyr::filter(SAFFL=='Y'),
D_POPLBL="Safety",
D_SUBJID=c("STUDYID","USUBJID"),
D_TITLE1=paste0('Table 1: Summary of Treatment Emergent Adverse Events'),
D_RTFYN="Y",
D_DEBUG=0,
R_DICTION=NULL,
R_OTHERDATA=NULL,
R_INPUTDATA=NULL,
R_RAWDATA=NULL,
R_SDTMDATA=NULL,
R_ADAMDATA=datdir,
R_RFMTDIR=fmtdir,
R_DDDATA=paste0(outdir,'/t_ru_list_1.rds'),
D_OUTFILE=paste0(outdir,"/t_ru_list_1.rtf"),
D_PGMPTH="/path/to/code/rs_setup.R")
```

ru_addbignvar *Assign Big N to Data Frame.*

Description

Pass in a data frame along with identification options and have Big N added to it.

Usage

```
ru_addbignvar(
  dsetintoaddbign,
  dsetintocount,
  countdistinctvars = c("STUDYID", "USUBJID"),
  groupbyvars = NULL,
  totalforvar = NULL,
  totalid = NULL,
  totaldecode = c("Total"),
  codedecodevarpairs = NULL,
  varcodelistpairs = NULL,
  codelistnames = list(),
  addbigntovarvalue = TRUE,
  splitchar = " "
)
```

Arguments

`dsetintoaddbign`	The data set that will hold the derived big N value.
`dsetintocount`	The data set that will be counted to generate big N.

`countdistinctvars`	Variable(s) that contain values to be counted uniquely within any output grouping.
`groupbyvars`	Variables in DSETINTOCOUNT to group the data by when counting to deriving the big N.
`totalforvar`	Variable for which overall totals are required within all other grouped class variables.
`totalid`	Value(s) used to populate the variable(s) specified in totalforvar.
`totaldecode`	Value(s) used to populate the variable(s) of the decode variable(s) of the totalforvar.
`codedecodevarpairs`	Specifies code and decode variable pairs. Those variables should be in parameter GROUPBYVARSNUMER. One variable in the pair will contain the code, which is used in counting and ordering, and the other will contain decode, which is used for presentation.
`varcodelistpairs`	List of code/decode pairs of variables.
`codelistnames`	List of decodes for use with decoding code/decode pairs.
`addbigntovarvalue`	Place big N in a new variable or append to an existing variable (last groupbyvars value)?
`splitchar`	Text to insert between existing string and big N.

Value

A data frame based on the incoming data frame but collapsed by groups with descriptive statistics added.

Author(s)

Yongwei Wang, `<yongwei.x.wang@viivhealthcare.com>`
Chris Rook, `<christopher.x.rook@gsk.com>`

Examples

```
library(repfun)
library(dplyr)
library(tibble)
rfenv <- if (exists('rfenv') && is.environment(get('rfenv'))){
             rfenv
         } else {
             rfenv <- new.env(parent = emptyenv())
             rfenv$G_DEBUG <- 0
             rfenv
         }
datdir <- file.path(tempdir(),"datdir"); dir.create(datdir,showWarnings=FALSE)
repfun::copydata(datdir)
```

```
repfun::rs_setup(D_POPDATA=repfun::adsl %>% dplyr::filter(SAFFL =='Y'),
                 D_SUBJID=c("STUDYID","USUBJID"),
                 R_DICTION=NULL,
                 R_OTHERDATA=NULL,
                 R_INPUTDATA=NULL,
                 R_RAWDATA=NULL,
                 R_SDTMDATA=NULL,
                 R_ADAMDATA=datdir)
G_POPDATA <- rfenv$G_POPDATA %>% dplyr::mutate(TRT01AN=
                                        ifelse(TRT01A=='Placebo',1,
                                        ifelse(TRT01A=='Xanomeline Low Dose',2,3)))
attr(G_POPDATA$TRT01AN,"label") <- 'Actual Treatment for Period 01 (n)'
adae <- tibble::as_tibble(rfenv$adamdata$adae.rda()) %>%
        dplyr::inner_join(G_POPDATA,
                          by=c('STUDYID','USUBJID','SAFFL','TRT01A')) %>%
        dplyr::filter(TRTEMFL=='Y')
addbign <- repfun::ru_addbignvar(adae,
                         G_POPDATA,
                         groupbyvars=c("TRT01AN", "TRT01A"),
                         countdistinctvars=c("STUDYID", "USUBJID"),
                         totalforvar=c("TRT01AN"),
                         totalid = 99,
                         totaldecode = 'Total',
                         codedecodevarpairs=c("TRT01AN", "TRT01A"),
                         varcodelistpairs=c(""),
                         codelistnames=list(),
                         addbigntovarvalue=TRUE,
                         splitchar="~") %>%
           dplyr::select(STUDYID, USUBJID, TRT01AN, TRT01A, AEBODSYS, AEDECOD)
```

ru_addpage *Add a Page Number Column to an Existing Dataframe*

Description

Take incoming dataframe and add page number variable to it accounting for grouping variables, stacked variables, and no-split variables.

Usage

```
ru_addpage(
  dsetin,
  grpvars = NULL,
  stackvars = NULL,
  varlabls = NULL,
  rowsprbdy = NULL,
  startpaging = 0,
  lastbygrp = FALSE,
  fpage = "all",
  nftnotes = 0,
  nosplitvars = FALSE,
  npgvars = 0
)
```

Arguments

`dsetin`	The dataframe for which a paging variable will be added.
`grpvars`	Grouping variables used in the output (used for nosplitvars).
`stackvars`	Specify stacked grouping variables (reduces # of page lines available for data).
`varlabls`	Apply labels to outgoing dataframe.
`rowsprbdy`	Number of rows in the body of the report.
`startpaging`	Set to zero on first call, and > 0 on recalls to fix widows.
`lastbygrp`	Set to true if this page is processing the last value of the grouping variables (used when footnote is applied only to last page).
`fpage`	Setting to 'last' indicates that footnotes are only displayed on the last page.
`nftnotes`	Enter the number of footnotes (determines # of page lines available for data).
`nosplitvars`	Setting to true requires all values of the last grouping/stackvar must be on the same page (if possible).
`npgvars`	Number of page-by variables for this report (reduces # of page lines available for data).

Value

A dataframe based on the incoming dataframe but with a paging variable added.

Author(s)

Chris Rook, `<christopher.x.rook@gsk.com>`
Yongwei Wang, `<yongwei.x.wang@viivhealthcare.com>`

Examples

```
library(repfun)
library(dplyr)
rfenv <- if (exists('rfenv') && is.environment(get('rfenv'))){
             rfenv
         } else {
             rfenv <- new.env(parent = emptyenv())
             rfenv$G_DEBUG <- 0
             rfenv
         }
datdir <- file.path(tempdir(),"datdir"); dir.create(datdir,showWarnings=FALSE)
repfun::copydata(datdir)
repfun::rs_setup(D_POPDATA=repfun::adsl %>% dplyr::filter(SAFFL =='Y'),
                 D_SUBJID=c("STUDYID","USUBJID"),
                  R_DICTION=NULL,
                  R_OTHERDATA=NULL,
                  R_INPUTDATA=NULL,
                  R_RAWDATA=NULL,
                  R_SDTMDATA=NULL,
                  R_ADAMDATA=datdir)
G_POPDATA <- rfenv$G_POPDATA %>%
 dplyr::mutate(TRT01AN=ifelse(TRT01A=='Placebo',1,
               ifelse(TRT01A=='Xanomeline Low Dose',2,3))) %>%
 repfun::ru_labels(varlabels=list('TRT01AN'='Actual Treatment for Period 001 (n)'))
adae <- rfenv$adamdata$adae.rda() %>%
        dplyr::inner_join(G_POPDATA, by=c('STUDYID','USUBJID','SAFFL','TRT01A'))
aesum_t <- repfun::ru_freq(
             adae,
             dsetindenom=G_POPDATA,
             countdistinctvars=c('STUDYID','USUBJID'),
             groupbyvarsnumer=c('TRT01AN','TRT01A','AEBODSYS','AEDECOD'),
             anyeventvars = c('AEBODSYS','AEDECOD'),
             anyeventvalues = c('ANY EVENT','ANY EVENT'),
             groupbyvarsdenom=c('TRT01AN'),
             resultstyle="NUMERPCT",
             totalforvar=c('TRT01AN'),
             totalid=99,
             totaldecode='Total',
             codedecodevarpairs=c("TRT01AN", "TRT01A"),
             varcodelistpairs=c(""),
             codelistnames=list(),
             resultpctdps=0) %>%
 repfun::ru_denorm(varstodenorm=c("tt_result", "PERCENT"),
           groupbyvars=c("tt_summarylevel", "AEBODSYS", "AEDECOD"),
           acrossvar="TRT01AN",
           acrossvarlabel="TRT01A",
           acrossvarprefix=c("tt_ac", "tt_p")) %>%
 dplyr::mutate(ord1=ifelse(tt_summarylevel==0,0,1)) %>%
 dplyr::rename(ord2=tt_summarylevel) %>%
 dplyr::arrange(ord1,AEBODSYS,ord2,AEDECOD) %>%
 dplyr::select(-c(starts_with('tt_p'),starts_with('ord')))

# Example 1: Simple paging.
aesum_p1 <- repfun::ru_addpage(aesum_t,grpvars=c('AEBODSYS'),rowsprbdy=30)
print(head(aesum_p1,40))

# Example 2: No splitvars, but 30 rows won't work for this data.
aesum_p2 <- repfun::ru_addpage(aesum_t,grpvars=c('AEBODSYS'),rowsprbdy=30,
                       nosplitvars=TRUE)
print(head(aesum_p2,40))

# Example 3: No splitvars, but 35 rows is enough.
aesum_p3 <- repfun::ru_addpage(aesum_t,grpvars=c('AEBODSYS'),rowsprbdy=35,
                       nosplitvars=TRUE)
print(head(aesum_p3,40))
```

ru_addsupp *Cobmine an SDTM domain with its corresponding Supplemental data set*

Description

Pass in an SDTM data frame along with its Supplemental version and they will be combined.

Usage

```
ru_addsupp(dsetin, dsetinsupp)
```

Arguments

`dsetin`	Incoming data frame to have supplemental data added.
`dsetinsupp`	The supplemental data set to add.

Value

The original SDTM data set with its supplemental data appended as new variables.

Author(s)

Yongwei Wang, `<yongwei.x.wang@viivhealthcare.com>`
Chris Rook, `<christopher.x.rook@gsk.com>`

Examples

```
library(repfun)
library(knitr)
ae <- repfun::ae
suppae <- repfun::suppae
aesupp <- repfun::ru_addsupp(dsetin=ae,dsetinsupp=suppae)
knitr::kable(head(aesupp,5), caption = "SDTM.AE combined with SDTM.SUPPAE")
```

ru_align *Align Columns for Reporting*

Description

Pass in a data frame alignment criteria to have columns aligned for reporting.

Usage

```
ru_align(
  dsetin,
  varsin,
  byvars = NULL,
  alignment = "Right",
  compresschryn = "Y",
  ncspaces = 1
)
```

Arguments

`dsetin`	The data set holding the columns to align.
`varsin`	Columns to align.
`byvars`	Set of group-by variables.
`alignment`	Type of alignment.
`compresschryn`	Compress by removing leading and trailing spaces for the resulting aligned column?
`ncspaces`	Number of spaces between N and Percent.

Value

A data frame based having the requested columns aligned for reporting.

Author(s)

Yongwei Wang, `<yongwei.x.wang@viivhealthcare.com>`
Chris Rook, `<christopher.x.rook@gsk.com>`

Examples

```
#=================
# Adverse Events
#=================
library(repfun)
library(dplyr)
rfenv <- if (exists('rfenv') && is.environment(get('rfenv'))){
             rfenv
         } else {
             rfenv <- new.env(parent = emptyenv())
             rfenv$G_DEBUG <- 0
             rfenv
         }
datdir <- file.path(tempdir(),"datdir"); dir.create(datdir,showWarnings=FALSE)
repfun::copydata(datdir)
repfun::rs_setup(D_POP="SAFFL",
                 D_POPLBL="Safety",
                 D_POPDATA=repfun::adsl %>% dplyr::filter(SAFFL =='Y'),
                 D_SUBJID=c("STUDYID","USUBJID"),
                 R_DICTION=NULL,
                 R_OTHERDATA=NULL,
                 R_INPUTDATA=NULL,
                 R_RAWDATA=NULL,
                 R_SDTMDATA=NULL,
                 R_ADAMDATA=datdir)
G_POPDATA <- rfenv$G_POPDATA %>% dplyr::mutate(TRT01AN=
                     ifelse(TRT01A=='Placebo',1,
                     ifelse(TRT01A=='Xanomeline Low Dose',2,3)))
attr(G_POPDATA$TRT01AN,"label") <- 'Actual Treatment for Period 01 (n)'
adae <- rfenv$adamdata$adae.rda() %>% dplyr::select(-SAFFL) %>%
        repfun::ru_getdata(G_POPDATA, c("STUDYID", "USUBJID"),
        keeppopvars=c("TRT01AN", "TRT01A"))
aesum_t <- repfun::ru_freq(adae,
              dsetindenom=G_POPDATA,
              countdistinctvars=c('STUDYID','USUBJID'),
              groupbyvarsnumer=c('TRT01AN','TRT01A','AEBODSYS','AEDECOD'),
              anyeventvars = c('AEBODSYS','AEDECOD'),
              anyeventvalues = c('ANY EVENT','ANY EVENT'),
              groupbyvarsdenom=c('TRT01AN'),
              resultstyle="NUMERPCT",
              totalforvar=c('TRT01AN'),
              totalid=99,
              totaldecode='Total',
              codedecodevarpairs=c("TRT01AN", "TRT01A"),
              varcodelistpairs=c(""),
              codelistnames=list(),
              resultpctdps=0) %>%
           repfun::ru_denorm(varstodenorm=c("tt_result", "PERCENT"),
                    groupbyvars=c("tt_summarylevel", "AEBODSYS", "AEDECOD"),
                    acrossvar="TRT01AN",
                    acrossvarlabel="TRT01A",
                    acrossvarprefix=c("tt_ac", "tt_p"))
print('Before Aligning')
print(head(aesum_t[,grep('(AEBODSYS|AEDECOD|tt_ac)',names(aesum_t))],20))
aesum_t_a <- repfun::ru_align(aesum_t, "tt_ac:")
print('After Aligning')
print(head(aesum_t_a[,grep('(AEBODSYS|AEDECOD|tt_ac)',names(aesum_t_a))],20))

#===========================
# Baseline Characteristics
#===========================
repfun::rs_setup(D_POP="SAFFL",
                 D_POPLBL="Safety",
                 D_POPDATA=repfun::adsl %>% dplyr::filter(SAFFL =='Y'),
                 D_SUBJID=c("STUDYID","USUBJID"),
                 R_DICTION=NULL,
                 R_OTHERDATA=NULL,
                 R_INPUTDATA=NULL,
                 R_RAWDATA=NULL,
                 R_SDTMDATA=NULL,
                 R_ADAMDATA=datdir)
G_POPDATA <- rfenv$G_POPDATA %>%
             dplyr::mutate(TRT01AN=ifelse(TRT01A=='Placebo',1,
                                ifelse(TRT01A=='Xanomeline Low Dose',2,3))) %>%
  repfun::ru_labels(varlabels=list('TRT01AN'='Actual Treatment for Period 01 (n)'))
demstats_t <- repfun::ru_sumstats(G_POPDATA,
                 analysisvars=c("AGE","TRTDURD"),
                 groupbyvars=c("STUDYID","TRT01AN"),
                 codedecodevarpairs=c("TRT01AN", "TRT01A"),
                 totalforvar="TRT01AN", totalid=99,
                 totaldecode="Total",
                 statsinrowsyn = "Y",
                 analysisvardps=list("AGE"=1,"TRTDURD"=2),
                 statslist=c("n", "mean", "median", "sd", "min", "max")) %>%
  repfun::ru_denorm(varstodenorm=c("tt_result"),
            groupbyvars=c("tt_avid", "tt_avnm", "tt_svid", "tt_svnm"),
            acrossvar="TRT01AN", acrossvarlabel="TRT01A",
            acrossvarprefix=c("tt_ac"))

print('Before Aligning')
print(head(demstats_t,20))
demstats_t_a <- repfun::ru_align(demstats_t, "tt_ac:", ncspaces=10)
print('After Aligning')
print(head(demstats_t_a,20))
```

ru_contents — *Generate proc contents of data frame similar to that generated by SAS*

Description

Pass in a data frame and the contents will be displayed.

Usage

```
ru_contents(dsetin)
```

Arguments

`dsetin`	Incoming data frame to have proc contents generated.

Value

The proc contents output will be displayed.

Author(s)

Yongwei Wang, <yongwei.x.wang@viivhealthcare.com>
Chris Rook, <christopher.x.rook@gsk.com>

Examples

```
library(repfun)
repfun::ru_contents(mtcars)
```

---

ru_data2codelist *Return a list with codelist, code and label based on input codelist dataset.*

Description

Pass in a data set from a SAS format catalog (or similar) and have a list returned in the structure of a SAS format for decoding variables.

Usage

```
ru_data2codelist(
  dsetin,
  codelistvarname = "FMTNAME",
  codevarname = "START",
  decodevarname = "LABEL",
  typevarname = "TYPE"
)
```

Arguments

`dsetin`	Name of incoming data set structured as a SAS format catalog saved as a data set.
`codelistvarname`	Name of the variable containing the SAS format or similar.
`codevarname`	Name of the variable that holds the code value.
`decodevarname`	Name of the variable that holds the decode value.
`typevarname`	Type of format (character or numeric).

Value

A data frame based on the incoming data frame but with decode values added along with records when completetypes is true.

Author(s)

Yongwei Wang, `<yongwei.x.wang@viivhealthcare.com>`
Chris Rook, `<christopher.x.rook@gsk.com>`

Examples

```
library(repfun)
rfenv <- if (exists('rfenv') && is.environment(get('rfenv'))){
             rfenv
         } else {
             rfenv <- new.env(parent = emptyenv())
             rfenv$G_DEBUG <- 0
             rfenv
         }
datdir <- file.path(tempdir(),"datdir"); dir.create(datdir,showWarnings=FALSE)
repfun::copydata(datdir)
repfun::rs_setup(R_RFMTDIR=datdir)
list <- repfun::ru_data2codelist(rfenv$rfmtdata$formats.rda())
list$SEXS$START[[1]] # Code value 1
list$SEXS$LABEL[[1]] # Decode value 1
list$SEXS$START[[2]] # Code value 2
list$SEXS$LABEL[[2]] # Decode value 2
```

---

ru_datacompare *Compare 2 data frames and report differences.*

Description

Pass in a base and compare data frame to find out if they are equal similar to proc compare in SAS.

Usage

```
ru_datacompare(dsetinbase, dsetincomp, idvars, maxprint = 50)
```

Arguments

dsetinbase	First data set.
dsetincomp	Second data set.
idvars	Match on these values prior to comparing records.
maxprint	Maximum number of differences per variable to display.

Value

An output similar to proc compare will be displayed.

Author(s)

Yongwei Wang, <yongwei.x.wang@viivhealthcare.com>
Chris Rook, <christopher.x.rook@gsk.com>

Examples

```
library(repfun)
repfun::ru_datacompare(iris,iris,idvars='Species')
```

`ru_datetime` *Add numeric datetimes to data frame that only has character versions.*

Description

Pass in a data frame and variables ending in "DTC" will have corresponding numeric versions created and saved.

Usage

```
ru_datetime(dsetin, includevars = NULL)
```

Arguments

`dsetin`	Incoming data frame to have numeric datetimes added.
`includevars`	Specify which variables ending in DTC will be processed.

Value

The proc contents output will be displayed.

Author(s)

Yongwei Wang, <yongwei.x.wang@viivhealthcare.com>
Chris Rook, <christopher.x.rook@gsk.com>

Examples

```
library(repfun)
library(knitr)
ae <- repfun::ru_datetime(repfun::ae)
knitr::kable(head(ae[,grepl('(DT$|TM$|DTC$)',names(ae))],5),
             caption = "After Invoking ru_labels()")
```

ru_denorm *Transpose a Data Frame*

Description

Pass in a data frame along with identification options and have it transposed (denormalized, long to wide) to display treatment columns.

Usage

```
ru_denorm(
  dsetin,
  varstodenorm = NULL,
  groupbyvars = NULL,
  acrossvar = NULL,
  acrossvarlabel = NULL,
  acrossvarprefix = "tt_",
  acrossvarsuffix = NULL
)
```

Arguments

`dsetin`	The data set to transpose.
`varstodenorm`	The variables to transpose.
`groupbyvars`	Definition of one row in the output data frame.
`acrossvar`	Variable to define the columns in the transposed data frame.
`acrossvarlabel`	Variable to define the labels in the transposed data frame.
`acrossvarprefix`	Add to the beginning of each value in the across variable in the output data frame.
`acrossvarsuffix`	Add to the end of each value in the across variable in the output data frame.

Details

dsetin, varstodenorm=NULL, groupbyvars=NULL, acrossvar=NULL, acrossvarlabel=NULL, acrossvarprefix="tt_", acrossvarsuffix=NULL

Value

A data frame based on the incoming data frame transposed from long to wide.

Author(s)

Yongwei Wang, `<yongwei.x.wang@viivhealthcare.com>`
Chris Rook, `<christopher.x.rook@gsk.com>`

Examples

```
#====================
# AEs: N and Percent
#====================
library(repfun)
library(dplyr)
rfenv <- if (exists('rfenv') && is.environment(get('rfenv'))){
             rfenv
         } else {
             rfenv <- new.env(parent = emptyenv())
             rfenv$G_DEBUG <- 0
             rfenv
         }
datdir <- file.path(tempdir(),"datdir"); dir.create(datdir,showWarnings=FALSE)
repfun::copydata(datdir)
repfun::rs_setup(D_POP="SAFFL",
                 D_POPLBL="Safety",
                 D_POPDATA=repfun::adsl %>% dplyr::filter(SAFFL =='Y'),
                 D_SUBJID=c("STUDYID","USUBJID"),
                 R_DICTION=NULL,
                 R_OTHERDATA=NULL,
                 R_INPUTDATA=NULL,
                 R_RAWDATA=NULL,
                 R_SDTMDATA=NULL,
                 R_ADAMDATA=datdir)
G_POPDATA <- rfenv$G_POPDATA %>%
  dplyr::mutate(TRT01AN=ifelse(TRT01A=='Placebo',1,
                 ifelse(TRT01A=='Xanomeline Low Dose',2,3))) %>%
  repfun::ru_labels(varlabels=list('TRT01AN'='Actual Treatment for Period 01 (n)'))
adae <- rfenv$adamdata$adae.rda() %>% select(-SAFFL) %>%
        repfun::ru_getdata(G_POPDATA, c("STUDYID", "USUBJID"),
                   keeppopvars=c("TRT01AN", "TRT01A"))
aesum_t <- repfun::ru_freq(adae,
              dsetindenom=G_POPDATA,
              countdistinctvars=c('STUDYID','USUBJID'),
              groupbyvarsnumer=c('TRT01AN','TRT01A','AEBODSYS','AEDECOD'),
              anyeventvars = c('AEBODSYS','AEDECOD'),
              anyeventvalues = c('ANY EVENT','ANY EVENT'),
              groupbyvarsdenom=c('TRT01AN'),
              resultstyle="NUMERPCT",
              totalforvar=c('TRT01AN'),
              totalid=99,
              totaldecode='Total',
              codedecodevarpairs=c("TRT01AN", "TRT01A"),
              varcodelistpairs=c(""),
              codelistnames=list(),
              resultpctdps=0) %>%
  repfun::ru_denorm(varstodenorm=c("tt_result", "PERCENT"),
            groupbyvars=c("tt_summarylevel", "AEBODSYS", "AEDECOD"),
            acrossvar="TRT01AN",
            acrossvarlabel="TRT01A",
            acrossvarprefix=c("tt_ac", "tt_p"))
```

```
#=======================================
# Demography Statistics: N and Percent
#=======================================
repfun::rs_setup(D_POP="SAFFL",
                 D_POPLBL="Safety",
                 D_POPDATA=repfun::adsl %>% dplyr::filter(SAFFL =='Y'),
                 D_SUBJID=c("STUDYID","USUBJID"),
                 R_DICTION=NULL,
                 R_OTHERDATA=NULL,
                 R_INPUTDATA=NULL,
                 R_RAWDATA=NULL,
                 R_SDTMDATA=NULL,
                 R_ADAMDATA=datdir)
G_POPDATA <- rfenv$G_POPDATA %>%
              dplyr::mutate(TRT01AN=
                     ifelse(TRT01A=='Placebo',1,
                     ifelse(TRT01A=='Xanomeline Low Dose',2,3))) %>%
  repfun::ru_labels(varlabels=list('TRT01AN'='Actual Treatment for Period 01 (n)'))
demstats_t <- repfun::ru_sumstats(G_POPDATA,
                 analysisvars=c("AGE","TRTDURD"),
                 groupbyvars=c("STUDYID","TRT01AN"),
                 codedecodevarpairs=c("TRT01AN", "TRT01A"),
                 totalforvar="TRT01AN", totalid=99,
                 totaldecode="Total",
                 statsinrowsyn = "Y",
                 analysisvardps=list("AGE"=1,"TRTDURD"=2),
                 statslist=c("n", "mean", "median", "sd", "min", "max")) %>%
  repfun::ru_denorm(varstodenorm=c("tt_result"),
            groupbyvars=c("tt_avid", "tt_avnm", "tt_svid", "tt_svnm"),
            acrossvar="TRT01AN", acrossvarlabel="TRT01A",
            acrossvarprefix=c("tt_ac"))
```

ru_expvarlist *Expand SAS Style Variable/Column List*

Description

Pass in a data frame along with column/variable identifiers formatted with SAS Style (i.e., using colon) and it will be expanded to the actual variable list.

Usage

```
ru_expvarlist(dsetin, varsin = NULL, keepnotexist = FALSE)
```

Arguments

`dsetin`	A dataframe holding columns whose names will be expanded.
`varsin`	A SAS style list of variable names.
`keepnotexist`	If the variable does not exist on the dataframe it will be excluded from the list.

Value

A list of column/variable names.

Author(s)

Yongwei Wang, `<yongwei.x.wang@viivhealthcare.com>`
Chris Rook, `<christopher.x.rook@gsk.com>`

Examples

```
library(repfun)
df <- data.frame(tt_ac01=c('1','2','3'),
                 tt_ac02=c('a','b','b'),
                 tt_ac03=c('10','11','12'))
chk <- repfun::ru_expvarlist(df, varsin="tt_ac:")
print(chk)
```

`ru_fillcodedcode` *Fill missing code/decode records*

Description

Pass in a data frame with along code/decode variables and values to have missings populated.

Usage

```
ru_fillcodedcode(
  dsetin,
  codedecodevarpairs = NULL,
  varcodelistpairs = NULL,
  codelistnames = list(),
  groupbyvars = NULL,
  completetypes = TRUE
)
```

Arguments

`dsetin`	The data set that will be counted to generate numerators for counts and percents.
`codedecodevarpairs`	Specifies code and decode variable pairs. Those variables should be in parameter GROUPBYVARSNUMER. One variable in the pair will contain the code, which is used in counting and ordering, and the other will contain decode, which is used for presentation.
`varcodelistpairs`	List of code/decode pairs of variables.

`codelistnames`	List of decodes for use with decoding code/decode pairs.
`groupbyvars`	Set of by-variables used to merge the incoming data set with the decode data set.
`completetypes`	Keep all code/decode pairs even it not present on the incoming data set?

Value

A data frame based on the incoming data frame but with decode values added along with records when completetypes is true.

Author(s)

Yongwei Wang, `<yongwei.x.wang@viivhealthcare.com>`
Chris Rook, `<christopher.x.rook@gsk.com>`

Examples

```
library(repfun)
rfenv <- if (exists('rfenv') && is.environment(get('rfenv'))){
             rfenv
         } else {
             rfenv <- new.env(parent = emptyenv())
             rfenv$G_DEBUG <- 0
             rfenv
         }
fmtlist <- list('SEXS'=list('START'=list('M','F'),
                'LABEL'=c('Male','Female')))
adsl <- repfun::adsl
adsl2 <- repfun::ru_fillcodedcode(adsl, codedecodevarpairs=c("SEX", "SEXDCD"),
                          varcodelistpairs=c("SEX", "SEXS"),
                          codelistnames=fmtlist)
unique(adsl2[c("SEX","SEXDCD")])
```

`ru_fillna` *Fill NA values with specified values or zeros/blanks by default.*

Description

Pass in a data frame along with a vector of variables and a vector of fill values. (Default fill is 0 for numeric and blank " " for character.)

Usage

```
ru_fillna(dsetin, vars = NULL, fills = NULL)
```

Arguments

`dsetin`	Incoming data frame to have labels added to columns.
`vars`	Vector of variables to replace NA values.
`fills`	Vector of fill values.

Value

The incoming data frame with the requested NA values replaced.

Author(s)

Yongwei Wang, <yongwei.x.wang@viivhealthcare.com>
Chris Rook, <christopher.x.rook@gsk.com>

Examples

```
library(repfun)
repfun::ru_fillna(airquality, vars=c('Ozone','Solar.R'), fills=c(1111,2222))
```

`ru_freq` *Create Percentage based on Numerator and Denominator Data*

Description

Pass in a data frame along with identification options and have descriptive statistics derived.

Usage

```
ru_freq(
  dsetin,
  dsetindenom = NULL,
  countdistinctvars = NULL,
  groupbyvarsnumer = NULL,
  groupbyvarsdenom = NULL,
  resultstyle = "NUMERPCT",
  totalforvar = NULL,
  totalid = NULL,
  totaldecode = c("Total"),
  anyeventvars = NULL,
  anyeventvalues = NULL,
  codedecodevarpairs = NULL,
  varcodelistpairs = NULL,
  codelistnames = list(),
  groupminmaxvar = NULL,
  resultpctdps = 0
)
```

Arguments

`dsetin`	The data set that will be counted to generate descriptive statistics.
`dsetindenom`	Input dataset containing data to be counted to obtain the denominator.
`countdistinctvars`	Variable(s) that contain values to be counted uniquely within any output grouping.
`groupbyvarsnumer`	Variables in DSETINNUMER to group the data by when counting to obtain the numerator.
`groupbyvarsdenom`	Variables in DSETINDENOM to group the data by when counting to obtain the denominator.
`resultstyle`	The appearance style of the result columns that will be displayed in the report.
`totalforvar`	Variable for which overall totals are required within all other grouped class variables.
`totalid`	Value(s) used to populate the variable(s) specified in totalforvar.
`totaldecode`	Value(s) used to populate the variable(s) of the decode variable(s) of the totalforvar.
`anyeventvars`	Set of variables for which total rows will be added.
`anyeventvalues`	Set of text values for total rows generated above.
`codedecodevarpairs`	Specifies code and decode variable pairs. Those variables should be in parameter GROUPBYVARSNUMER. One variable in the pair will contain the code, which is used in counting and ordering, and the other will contain decode, which is used for presentation.
`varcodelistpairs`	List of code/decode pairs of variables.
`codelistnames`	List of decodes for use with decoding code/decode pairs.
`groupminmaxvar`	Specify if frequency of each group should be from first or last value of a variable in format MIN(variables).
`resultpctdps`	The reporting precision for percentages.

Value

A data frame based on the incoming data frame but collapsed by groups with descriptive statistics added.

Author(s)

Yongwei Wang, `<yongwei.x.wang@viivhealthcare.com>`
Chris Rook, `<christopher.x.rook@gsk.com>`

Examples

```
library(repfun)
library(dplyr)
library(tibble)
rfenv <- if (exists('rfenv') && is.environment(get('rfenv'))){
             rfenv
         } else {
             rfenv <- new.env(parent = emptyenv())
             rfenv$G_DEBUG <- 0
             rfenv
         }
datdir <- file.path(tempdir(),"datdir"); dir.create(datdir,showWarnings=FALSE)
outdir <- file.path(tempdir(),"outdir"); dir.create(outdir,showWarnings=FALSE)
fmtdir <- file.path(tempdir(),"fmtdir"); dir.create(fmtdir,showWarnings=FALSE)
repfun::copydata(datdir)
repfun::rs_setup(D_POPDATA=repfun::adsl %>% dplyr::filter(SAFFL =='Y'),
                 D_SUBJID=c("STUDYID","USUBJID"),
                 R_DICTION=NULL,
                 R_OTHERDATA=NULL,
                 R_INPUTDATA=NULL,
                 R_RAWDATA=NULL,
                 R_SDTMDATA=NULL,
                 R_ADAMDATA=datdir)
G_POPDATA <- rfenv$G_POPDATA %>%
  dplyr::mutate(TRT01AN=ifelse(TRT01A=='Placebo',1,
                 ifelse(TRT01A=='Xanomeline Low Dose',2,3)))
attr(G_POPDATA$TRT01AN,"label") <- 'Actual Treatment for Period 01 (n)'
adae <- rfenv$adamdata$adae.rda() %>%
  dplyr::inner_join(G_POPDATA, by=c('STUDYID','USUBJID','SAFFL','TRT01A'))
aesum <- repfun::ru_freq(adae,
                 dsetindenom=G_POPDATA,
                 countdistinctvars=c('STUDYID','USUBJID'),
                 groupbyvarsnumer=c('TRT01AN','TRT01A','AEBODSYS','AEDECOD'),
                 anyeventvars = c('AEBODSYS','AEDECOD'),
                 anyeventvalues = c('ANY EVENT','ANY EVENT'),
                 groupbyvarsdenom=c('TRT01AN'),
                 resultstyle="NUMERPCT",
                 totalforvar=c('TRT01AN'),
                 totalid=99,
                 totaldecode='Total',
                 codedecodevarpairs=c("TRT01AN", "TRT01A"),
                 varcodelistpairs=c(""),
                 codelistnames=list(),
                 resultpctdps=0)
```

`ru_getdata` *Assign Big N to Data Frame.*

Description

Merge input data with population data to keep only subjects which are in population sub data,

Usage

```
ru_getdata(
  dsetin,
  dsetinpop = rfenv$G_POPDATA,
  subjidvars = c("STUDYID", "USUBJID"),
  subpop = rfenv$G_SUBPOP,
  pop = rfenv$G_POP,
  keeppopvars = rfenv$G_KEEPPOPVARS
)
```

Arguments

`dsetin`	The data set that will be merged with the population data set.
`dsetinpop`	The population data set.
`subjidvars`	Variable(s) that define a unique subject.
`subpop`	A sub-population expression where variables are on DSETINPOP.
`pop`	The population expression (SAFFL=='Y').
`keeppopvars`	Variables to keep on the population data set.

Value

A data frame based on the incoming data frame but restricted to the population of interest with relevant population variables added.

Author(s)

Yongwei Wang, `<yongwei.x.wang@viivhealthcare.com>`
Chris Rook, `<christopher.x.rook@gsk.com>`

Examples

```
library(repfun)
library(dplyr)
rfenv <- if (exists('rfenv') && is.environment(get('rfenv'))){
             rfenv
         } else {
             rfenv <- new.env(parent = emptyenv())
             rfenv$G_DEBUG <- 0
             rfenv
         }
datdir <- file.path(tempdir(),"datdir"); dir.create(datdir,showWarnings=FALSE)
outdir <- file.path(tempdir(),"outdir"); dir.create(outdir,showWarnings=FALSE)
fmtdir <- file.path(tempdir(),"fmtdir"); dir.create(fmtdir,showWarnings=FALSE)
repfun::copydata(datdir)
repfun::rs_setup(D_POP="SAFFL",
                 D_POPLBL="Safety",
                 D_POPDATA=repfun::adsl,
                 D_SUBJID=c("STUDYID","USUBJID"),
                 R_DICTION=NULL,
                 R_OTHERDATA=NULL,
                 R_INPUTDATA=NULL,
                 R_RAWDATA=NULL,
                 R_SDTMDATA=NULL,
                 R_ADAMDATA=datdir)
G_POPDATA <- rfenv$G_POPDATA %>%
  dplyr::mutate(TRT01AN=ifelse(TRT01A=='Placebo',1,
                 ifelse(TRT01A=='Xanomeline Low Dose',2,3)),
         SAFFL=ifelse((row_number() %% 10) == 0,'N',SAFFL))
attr(G_POPDATA$TRT01AN,"label") <- 'Actual Treatment for Period 01 (n)'
attr(G_POPDATA$SAFFL,"label") <- 'Safety Population Flag'
adae <- rfenv$adamdata$adae.rda() %>% dplyr::select(-SAFFL)
adae2 <- repfun::ru_getdata(adae, G_POPDATA, c("STUDYID", "USUBJID"),
                    keeppopvars=c("TRT01AN", "TRT01A"))
```

`ru_groupbyvars` *Modify groupbyvars by adding or removing decode vars.*

Description

Pass in a vector of group-by variables along with a vector of code/decode pairs to have decode variables added or removed.

Usage

```
ru_groupbyvars(groupbyvars, codedecodevarpairs, adddecode = TRUE)
```

Arguments

`groupbyvars`	Vector of group-by variables.
`codedecodevarpairs`	Specifies code and decode variable pairs. Those variables should be in parameter GROUPBYVARSNUMER. One variable in the pair will contain the code, which is used in counting and ordering, and the other will contain decode, which is used for presentation.
`adddecode`	Add decode variables (true) or remove (false).

Value

A data frame based on the incoming data frame but collapsed by groups with counts and percents added.

Author(s)

Yongwei Wang, <yongwei.x.wang@viivhealthcare.com>
Chris Rook, <christopher.x.rook@gsk.com>

Examples

```
library(repfun)
add_decode <- repfun::ru_groupbyvars(
               c("TRTCD", "TRTGRP", "ATOXGRN", "AEDECOD", "AEBODSYS"),
               c("TRTCD", "TRTGRP", "ATOXGRN", "ATOXGR"), TRUE)
rem_decode <- repfun::ru_groupbyvars(
               c("TRTCD", "TRTGRP", "ATOXGRN", "AEDECOD", "AEBODSYS"),
               c("TRTCD", "TRTGRP", "ATOXGRN", "ATOXGR"), FALSE)
```

---

ru_labels *Assign labels to variables in a data frame*

Description

Pass in a data frame along with a named list of columns with their corresponding labels.

Usage

```
ru_labels(dsetin, varlabels = list(), style = c("base", "Hmisc"))
```

Arguments

dsetin	Incoming data frame to have labels added to columns.
varlabels	List of variables and their labels.
style	Type of method used to add labels.

Value

The incoming data frame with labels added.

Author(s)

Yongwei Wang, <yongwei.x.wang@viivhealthcare.com>
Chris Rook, <christopher.x.rook@gsk.com>

Examples

```
library(repfun)
ru_labels(mtcars,varlabels=list(mpg='Miles per gallon',
                                 cyl='Number of cylinders'))
```

`ru_libname` *Automate retrieval of data sets/frames in specified folder.*

Description

Provide a folder containing data sets/frames and have a list of function calls returned that can be used to quickly access individual data sets/frames.

Usage

```
ru_libname(datapath)
```

Arguments

`datapath`	Location of reporting data sets.

Value

List of function calls for use in quickly accessing individual data sets/frames.

Author(s)

Yongwei Wang, `<yongwei.x.wang@viivhealthcare.com>`
Chris Rook, `<christopher.x.rook@gsk.com>`

Examples

```
library(repfun)
datdir <- file.path(tempdir(),"datdir"); dir.create(datdir,showWarnings=FALSE)
repfun::copydata(datdir)
adamdata <- repfun::ru_libname(datdir)
adamdata$adae.rda() %>% dplyr::filter(TRTEMFL=='Y') -> teae
print(head(teae[,c(1:10)]),10)
```

`ru_list` *R function to mimic the SAS macro %tu_list.*

Description

Pass in a dataframe and reporting settings to have RTF output generated.

Usage

```
ru_list(
  dsetin,
  stackvar1 = NULL,
  stackvar2 = NULL,
  stackvar3 = NULL,
  stackvar4 = NULL,
  stackvar5 = NULL,
  stackvar6 = NULL,
  stackvar7 = NULL,
  stackvar8 = NULL,
  stackvar9 = NULL,
  stackvar10 = NULL,
  stackvar11 = NULL,
  stackvar12 = NULL,
  stackvar13 = NULL,
  stackvar14 = NULL,
  stackvar15 = NULL,
  display = "Y",
  varlabelstyle = "NOT IMPLEMENTED",
  dddatasetlabel = NULL,
  splitchar = "\n",
  getdatayn = "N",
  labelvarsyn = NULL,
  computebeforepagelines = NULL,
  computebeforepagevars = NULL,
  columns = NULL,
  ordervars = NULL,
  descending = NULL,
  orderformatted = "NOT IMPLEMENTED",
  orderfreq = "NOT IMPLEMENTED",
  orderdata = NULL,
  noprintvars = NULL,
  byvars = NULL,
  flowvars = "NOT IMPLEMENTED",
  widths = NULL,
  defaultwidths = "NOT IMPLEMENTED",
  skipvars = NULL,
  pagevars = NULL,
  idvars = NULL,
  linevars = NULL,
  centrevars = NULL,
  leftvars = NULL,
  rightvars = NULL,
  colspacing = 2,
  varspacing = "NOT IMPLEMENTED",
  formats = "NOT IMPLEMENTED",
  labels = NULL,
  break1 = "NOT IMPLEMENTED",
  break2 = "NOT IMPLEMENTED",
  break3 = "NOT IMPLEMENTED",
  break4 = "NOT IMPLEMENTED",
  break5 = "NOT IMPLEMENTED",
  nowidowvar = NULL,
  sharecolvars = NULL,
  sharecolvarsindent = 2,
  overallsummary = "n",
  proptions = "HEADLINE",
  denormyn = "N",
  varsToDenorm = NULL,
  groupByVars = NULL,
  acrossVar = NULL,
  acrossVarLabel = NULL,
  acrossColVarPrefix = NULL,
  acrossVarListName = NULL,
  lpp = 24,
  rpp = 50,
  toprow = "single",
  spanlbls = NULL,
  spanwidths = NULL,
  spanjust = NULL,
  spanbbord = NULL,
  spantbord = NULL,
  span2lbls = NULL,
  span2widths = NULL,
  span2just = NULL,
  span2bbord = NULL,
  xptyn = "N"
)
```

Arguments

`dsetin`	Incoming data frame or list of data frames.
`stackvar1`	Create Stacked variables (e.g. stackvar1=list(varsin=c('invid','subjid'), varout='st_inv_subj', sepc='/', splitc='\n'))
`stackvar2`	Create Stacked variables (e.g. stackvar2=list(varsin=c('invid','subjid'), varout='st_inv_subj', sepc='/', splitc='\n'))
`stackvar3`	Create Stacked variables (e.g. stackvar3=list(varsin=c('invid','subjid'), varout='st_inv_subj', sepc='/', splitc='\n'))
`stackvar4`	Create Stacked variables (e.g. stackvar4=list(varsin=c('invid','subjid'), varout='st_inv_subj', sepc='/', splitc='\n'))
`stackvar5`	Create Stacked variables (e.g. stackvar5=list(varsin=c('invid','subjid'), varout='st_inv_subj', sepc='/', splitc='\n'))
`stackvar6`	Create Stacked variables (e.g. stackvar6=list(varsin=c('invid','subjid'), varout='st_inv_subj', sepc='/', splitc='\n'))

`stackvar7`	Create Stacked variables (e.g. stackvar7=list(varsin=c('invid','subjid'), varout='st_inv_subj', sepc='/', splitc='\n'))
`stackvar8`	Create Stacked variables (e.g. stackvar8=list(varsin=c('invid','subjid'), varout='st_inv_subj', sepc='/', splitc='\n'))
`stackvar9`	Create Stacked variables (e.g. stackvar9=list(varsin=c('invid','subjid'), varout='st_inv_subj', sepc='/', splitc='\n'))
`stackvar10`	Create Stacked variables (e.g. stackvar10=list(varsin=c('invid','subjid'), varout='st_inv_subj', sepc='/', splitc='\n'))
`stackvar11`	Create Stacked variables (e.g. stackvar11=list(varsin=c('invid','subjid'), varout='st_inv_subj', sepc='/', splitc='\n'))
`stackvar12`	Create Stacked variables (e.g. stackvar12=list(varsin=c('invid','subjid'), varout='st_inv_subj', sepc='/', splitc='\n'))
`stackvar13`	Create Stacked variables (e.g. stackvar13=list(varsin=c('invid','subjid'), varout='st_inv_subj', sepc='/', splitc='\n'))
`stackvar14`	Create Stacked variables (e.g. stackvar14=list(varsin=c('invid','subjid'), varout='st_inv_subj', sepc='/', splitc='\n'))
`stackvar15`	Create Stacked variables (e.g. stackvar15=list(varsin=c('invid','subjid'), varout='st_inv_subj', sepc='/', splitc='\n'))
`display`	Specifies whether the report should be created
`varlabelstyle`	Specifies the label style for variables (SHORT or STD)
`dddatasetlabel`	Label to be applied to the DD dataset
`splitchar`	Split character
`getdatayn`	Control execution of tu_getdata
`labelvarsyn`	Control execution of tu_labelvars
`computebeforepagelines`	Specifies the text to be produced for the Compute Before Page lines (labelkey labelfmt colon labelvar)
`computebeforepagevars`	Names of variables that shall define the sort order for Compute Before Page lines
`columns`	Column parameter
`ordervars`	Order variables
`descending`	Descending ORDERVARS
`orderformatted`	ORDER=FORMATTED variables
`orderfreq`	ORDER=FREQ variables
`orderdata`	ORDER=DATA variables
`noprintvars`	No print vars (usually used to order the display)
`byvars`	By variables

`flowvars`	Variables with flow option
`widths`	Column widths
`defaultwidths`	List of default column widths
`skipvars`	similar to SAS statement skipvars Break after / skip
`pagevars`	similar to SAS statement pagevars Break after / page
`idvars`	ID variables
`linevars`	Order variable printed with line statements.
`centrevars`	Centre justify variables
`leftvars`	Left justify variables
`rightvars`	Right justify variables
`colspacing`	Overall spacing value.
`varspacing`	Spacing for individual variables.
`formats`	Format specification
`labels`	Label definitions.
`break1`	Break statements.
`break2`	Break statements.
`break3`	Break statements.
`break4`	Break statements.
`break5`	Break statements.
`nowidowvar`	Not in version 1
`sharecolvars`	Order variables that share print space.
`sharecolvarsindent`	Indentation factor
`overallsummary`	Overall summary line at top of tables
`proptions`	PROC REPORT statement options
`denormyn`	Controls whether denormalisation will occur
`varsToDenorm`	List of variables to be denormalised/transposed. Passed one at a time to the PROC TRANSPOSE VAR statement.
`groupByVars`	List of BY variables passed to PROC TRANSPOSE BY statement.
`acrossVar`	Variable used in the PROC TRANSPOSE ID statement.
`acrossVarLabel`	Variable used in the PROC TRANSPOSE IDLABEL statement.
`acrossColVarPrefix`	Text passed to the PROC TRANSPOSE PREFIX statement.
`acrossVarListName`	Macro variable name to contain the list of columns created by the transpose of the first variable in VARSTODENORM.
`lpp`	Lines within body of report (only used with manual paging).

`rpp`	Total lines per page, when there is no wrapping and excluding titles and footnotes - passed directly to r2rtf().
`toprow`	Control lines above first column header.
`spanlbls`	List of level 1 column spanning header labels.
`spanwidths`	List of level 1 column spanning header widths.
`spanjust`	List of level 1 column spanning header justifications.
`spanbbord`	List of level 1 column spanning bottom border values.
`spantbord`	List of level 1 column spanning top border values.
`span2lbls`	List of level 2 column spanning header labels (above Level 1).
`span2widths`	List of level 2 column spanning header widths (above Level 1).
`span2just`	List of level 2 column spanning header justifications (above Level 1).
`span2bbord`	List of level 2 column spanning bottom border values (above Level 1).
`xptyn`	Write DDDATA data frame as XPT file in addition to RDS?

Value

A formattted RTF report is generated to the specified file.

Author(s)

Chris Rook, `<christopher.x.rook@gsk.com>`
Yongwei Wang, `<yongwei.x.wang@viivhealthcare.com>`

Examples

```
library(repfun)
library(dplyr)
library(tibble)
rfenv <- if (exists('rfenv') && is.environment(get('rfenv'))){
             rfenv
         } else {
             rfenv <- new.env(parent = emptyenv())
             rfenv$G_DEBUG <- 0
             rfenv
         }
datdir <- file.path(tempdir(),"datdir"); dir.create(datdir,showWarnings=FALSE)
outdir <- file.path(tempdir(),"outdir"); dir.create(outdir,showWarnings=FALSE)
#===================================
# Set up the reporting environment.
#===================================
setup <- function(tlfid){
  repfun::rs_setup(
    D_DATADATE=Sys.Date(),
    D_DSPLYNUM=tlfid,
    D_FOOT1='1.) Only treatment emergent events related to lipids are displayed.',
    D_FOOT2='2.) Subjects counted once in each body system & preferred term.',
    D_KEEPPOPVARS=c('STUDYID','USUBJID','SAFFL'),
    D_STUDYID='ABCXYZPDQ',
    D_POP="SAFFL",
    D_POPDATA=repfun::adsl %>%
      dplyr::filter(SAFFL =='Y') %>%
      dplyr::mutate(TRT01AN=ifelse(TRT01A=='Placebo',1,
                           ifelse(TRT01A=='Xanomeline Low Dose',2,3))) %>%
      repfun::ru_labels(varlabels=
                 list('TRT01AN'='Actual Treatment for Period 01 (n)')),
    D_POPLBL="Safety",
    D_SUBJID=c("STUDYID","USUBJID"),
    D_TITLE1=paste0('Table ',tlfid,': Summary of Treatment Emergent Adverse Events'),
    D_OUTFILE=paste0(outdir,"/t_ru_list_",tlfid,".rtf"),
    D_PGMPTH="/path/to/code/ru_list.R",
    R_DDDATA=paste0(outdir,'/t_ru_list_',tlfid,'.rds'),
    R_ADAMDATA=datdir)
}

#=============================================
# Process ADAE - derive counts and percents.
#=============================================
setup(1)
aesum <- repfun::ru_freq(rfenv$adamdata$adae.rda() %>% dplyr::select(-SAFFL) %>%
                 repfun::ru_getdata(rfenv$G_POPDATA, c("STUDYID", "USUBJID"),
                 keeppopvars=c("TRT01AN", "TRT01A")),
                 dsetindenom=rfenv$G_POPDATA,
                 countdistinctvars=c('STUDYID','USUBJID'),
                 groupbyvarsnumer=c('TRT01AN','TRT01A','AEBODSYS','AEDECOD'),
                 anyeventvars = c('AEBODSYS','AEDECOD'),
                 anyeventvalues = c('ANY EVENT','ANY EVENT'),
                 groupbyvarsdenom=c('TRT01AN'),
                 resultstyle="NUMERPCT",
                 totalforvar=c('TRT01AN'),
                 totalid=99,
                 totaldecode='Total',
                 codedecodevarpairs=c("TRT01AN", "TRT01A"),
                 varcodelistpairs=c(""),
                 codelistnames=list(),
                 resultpctdps=0) %>%
dplyr::arrange(TRT01AN,TRT01A,AEBODSYS,tt_summarylevel,AEDECOD,NUMERCNT,DENOMCNT) %>%
repfun::ru_align("tt_result")

#=======================================
# Table 1:  Summary of Adverse Events
#=======================================
repfun::ru_list(aesum,
        columns=c('AEBODSYS','AEDECOD','tt_01','tt_02','tt_03','tt_99'),
        nowidowvar='AEBODSYS',
        widths=c(5.5,4.5,1.75,1.9,1.9,1.75),
        skipvars=c('AEBODSYS'),
        centrevars=c('tt_01','tt_02','tt_03','tt_99'),
        ordervars=c('AEBODSYS','tt_summarylevel','AEDECOD'),
        noprintvars=c('tt_summarylevel'),
        denormyn='Y',
        varsToDenorm=c('tt_result'),
            groupByVars=c('AEBODSYS','tt_summarylevel','AEDECOD'),
            acrossVar="TRT01AN",
            acrossVarLabel="TRT01A",
            acrossColVarPrefix='tt_',
            dddatasetlabel=paste0('DD Dataframe for AE Table ',rfenv$G_DSPLYNUM),
            lpp=23)

#===========================================================================
# Table 2:  Summary of Adverse Events using NOWIDOWVAR (remove SOCs that
# will not fit on 1 page with 10pt font)
#===========================================================================
setup(2)
SOCterms <- aesum %>% dplyr::distinct(AEBODSYS,AEDECOD)
SOCcnts <- table(SOCterms$AEBODSYS)
repfun::ru_list(aesum %>% dplyr::filter(!(AEBODSYS %in% names(SOCcnts[SOCcnts>=20]))),
        columns=c('AEBODSYS','AEDECOD','tt_01','tt_02','tt_03','tt_99'),
        nowidowvar='AEBODSYS',
        widths=c(5.5,4.5,1.75,1.9,1.9,1.75),
        skipvars=c('AEBODSYS'),
        centrevars=c('tt_01','tt_02','tt_03','tt_99'),
        ordervars=c('AEBODSYS','tt_summarylevel','AEDECOD'),
        noprintvars=c('tt_summarylevel'),
        denormyn='Y',
        varsToDenorm=c('tt_result'),
        groupByVars=c('AEBODSYS','tt_summarylevel','AEDECOD'),
        acrossVar="TRT01AN",
        acrossVarLabel="TRT01A",
        acrossColVarPrefix='tt_',
        dddatasetlabel=paste0('DD Dataframe for AE Table ',rfenv$G_DSPLYNUM),
        lpp=24)
```

`ru_load_library` *Load a list of libraries*

Description

Given a list of packages, check if installed and generate message, otherwise load package.

Usage

```
ru_load_library(pkgs)
```

Arguments

`pkgs`	A list of packages to check if installed and then load.

Value

No return value, packages are loaded.

Author(s)

Yongwei Wang, <yongwei.x.wang@viivhealthcare.com>
Chris Rook, <christopher.x.rook@gsk.com>

Examples

```
library(repfun)
repfun::ru_load_library(c("dplyr", "haven", "magrittr", "r2rtf"))
```

---

ru_setdata *Append data sets even when variables do not match*

Description

Pass in a collection of data frames separated by commas and they will be appended.

Usage

```
ru_setdata(..., keeprownames = TRUE)
```

Arguments

...	A collection of data frames.
keeprownames	Convert row names on data frame to columns and keep the column.

Value

The incoming data frames combined.

Author(s)

Yongwei Wang, <yongwei.x.wang@viivhealthcare.com>
Chris Rook, <christopher.x.rook@gsk.com>

Examples

```
library(repfun)
repfun::ru_setdata(mtcars,airquality)
```

ru_stackvar *Stack Columns of a Dataframe into New Column*

Description

Pass in a dataframe and columns to stack and have new dataframe returned that contains the stacked columns.

Usage

```
ru_stackvar(
  dsetin,
  sepc = "/",
  splitc = "\\line",
  varsin = NULL,
  varout = NULL,
  varlabel = NULL
)
```

Arguments

`dsetin`	Name of incoming dataframe with columns to have stacked.
`sepc`	Separator character between the stacked columns.
`splitc`	Split character between stacked columns.
`varsin`	List of variables to be stacked.
`varout`	Name of stacked column in dataframe.
`varlabel`	Label for new stacked column.

Value

The incoming dataframe with columns stacked as requested.

Author(s)

Chris Rook, `<christopher.x.rook@gsk.com>`
Yongwei Wang, `<yongwei.x.wang@viivhealthcare.com>`

Examples

```
library(repfun)
library(dplyr)
rfenv <- if (exists('rfenv') && is.environment(get('rfenv'))){
             rfenv
         } else {
             rfenv <- new.env(parent = emptyenv())
             rfenv$G_DEBUG <- 0
             rfenv
         }
datdir <- file.path(tempdir(),"datdir"); dir.create(datdir,showWarnings=FALSE)
repfun::copydata(datdir)
repfun::rs_setup(D_POPDATA=repfun::adsl %>% dplyr::filter(SAFFL =='Y'),
                 D_SUBJID=c("STUDYID","USUBJID"),
                 R_DICTION=NULL,
                 R_OTHERDATA=NULL,
                 R_INPUTDATA=NULL,
                 R_RAWDATA=NULL,
                 R_SDTMDATA=NULL,
                 R_ADAMDATA=datdir)
G_POPDATA <- rfenv$G_POPDATA %>%
  dplyr::mutate(TRT01AN=ifelse(TRT01A=='Placebo',1,
                 ifelse(TRT01A=='Xanomeline Low Dose',2,3))) %>%
  repfun::ru_labels(varlabels=list('TRT01AN'='Actual Treatment for Period 01 (n)'))
adae <- rfenv$adamdata$adae.rda() %>%
        dplyr::inner_join(G_POPDATA, by=c('STUDYID','USUBJID','SAFFL','TRT01A'))
aesum_p <- repfun::ru_freq(adae,
                   dsetindenom=G_POPDATA,
                   countdistinctvars=c('STUDYID','USUBJID'),
                   groupbyvarsnumer=c('TRT01AN','TRT01A','AEBODSYS','AEDECOD'),
                   anyeventvars = c('AEBODSYS','AEDECOD'),
                   anyeventvalues = c('ANY EVENT','ANY EVENT'),
                   groupbyvarsdenom=c('TRT01AN'),
                   resultstyle="NUMERPCT",
                   totalforvar=c('TRT01AN'),
                   totalid=99,
                   totaldecode='Total',
                   codedecodevarpairs=c("TRT01AN", "TRT01A"),
                   varcodelistpairs=c(""),
                   codelistnames=list(),
                   resultpctdps=0) %>%
repfun::ru_denorm(varstodenorm=c("tt_result", "PERCENT"),
          groupbyvars=c("tt_summarylevel", "AEBODSYS", "AEDECOD"),
          acrossvar="TRT01AN", acrossvarlabel="TRT01A",
          acrossvarprefix=c("tt_ac", "tt_p")) %>%
dplyr::mutate(ord1=ifelse(tt_summarylevel==0,0,1)) %>%
dplyr::rename(ord2=tt_summarylevel) %>%
dplyr::arrange(ord1,AEBODSYS,ord2,AEDECOD) %>%
dplyr::select(-c(starts_with('tt_p'),starts_with('ord'))) %>%
repfun::ru_stackvar(varsin=c('AEBODSYS','AEDECOD'),varout='SYSPREF',
                    varlabel='Body System/Preferred Term')
```

`ru_sumstats` *Calculate Descriptive Statistics*

Description

Pass in a data frame along with identification options and have descriptive statistics derived.

Usage

```
ru_sumstats(
  dsetin,
  analysisvars = NULL,
  analysisvarlabels = "",
  groupbyvars = NULL,
  statslist = c("n", "mean", "median", "min", "max", "sd", "q1", "q3"),
  statsinrowsyn = "N",
  analysisvardps = 0,
  statsdps = list(mean = 1, median = 1, sd = 2, se = 2),
  codedecodevarpairs = c(""),
  varcodelistpairs = c(""),
  codelistnames = list(),
  totalforvar = NULL,
  totalid = NULL,
  totaldecode = c("Total")
)
```

Arguments

`dsetin`	The data set that will be ounted to generate descriptive statistics.
`analysisvars`	The variables to be analysed.
`analysisvarlabels`	Specify a label statement which will be used to defined labels for statistics analysis variables defined in parameter ANALYSISVARS.
`groupbyvars`	Specifies the variables whose values define the subgroup combinations for the analysis. The variables can be divided by statements inside of (and) to represent different levels of subgroup.
`statslist`	Specifies a list of summary statistics to be produced.
`statsinrowsyn`	Place resulting descriptive statistics in rows or columns.
`analysisvardps`	Base precision of descriptive statistics prior to incorporating STATSDPS details.
`statsdps`	List of additional statistic-specific precision values to add to ANALYSISVARDPS.
`codedecodevarpairs`	Specifies code and decode variable pairs. Those variables should be in parameter GROUPBYVARSNUMER. One variable in the pair will contain the code, which is used in counting and ordering, and the other will contain decode, which is used for presentation.
`varcodelistpairs`	List of code/decode pairs of variables.
`codelistnames`	List of decodes for use with decoding code/decode pairs.
`totalforvar`	Variable for which overall totals are required within all other grouped class variables.

`totalid`	Value(s) used to populate the variable(s) specified in totalforvar.
`totaldecode`	Value(s) used to populate the variable(s) of the decode variable(s) of the totalforvar.

Value

A data frame based on the incoming data frame but collapsed by groups with descriptive statistics added.

Author(s)

Yongwei Wang, `<yongwei.x.wang@viivhealthcare.com>`
Chris Rook, `<christopher.x.rook@gsk.com>`

Examples

```
library(repfun)
library(dplyr)
rfenv <- if (exists('rfenv') && is.environment(get('rfenv'))){
             rfenv
         } else {
             rfenv <- new.env(parent = emptyenv())
             rfenv$G_DEBUG <- 0
             rfenv
         }
datdir <- file.path(tempdir(),"datdir"); dir.create(datdir,showWarnings=FALSE)
repfun::copydata(datdir)
rs_setup(D_POPDATA=repfun::adsl %>% dplyr::filter(SAFFL =='Y'),
         D_SUBJID=c("STUDYID","USUBJID"),
         R_DICTION=NULL,
         R_OTHERDATA=NULL,
         R_INPUTDATA=NULL,
         R_RAWDATA=NULL,
         R_SDTMDATA=NULL,
         R_ADAMDATA=datdir)
G_POPDATA <- rfenv$G_POPDATA %>% dplyr::mutate(
   TRT01AN=ifelse(TRT01A=='Placebo',1,
           ifelse(TRT01A=='Xanomeline Low Dose',2,3))) %>%
           ru_labels(varlabels=list('TRT01AN'='Actual Treatment for Period 01 (n)'))
ru_sumstats(G_POPDATA,
            analysisvars=c("AGE","TRTDURD"),
            groupbyvars=c("STUDYID","TRT01AN"),
            codedecodevarpairs=c("TRT01AN", "TRT01A"),
            totalforvar="TRT01AN", totalid=99,
            totaldecode="Total",
            statsinrowsyn = "Y",
            analysisvardps=list("AGE"=1,"TRTDURD"=2),
            statslist=c("n", "mean", "median", "sd", "min", "max"))
```

`ru_width_rtf` *Create a List of Relative Widths of Columns for RTF Outputs*

Description

Pass in a data set and identify the columns for reporting to have estimated relative column width returned.

Usage

```
ru_width_rtf(dsetin, varsin = list(), widths = list(), type = "PCT")
```

Arguments

`dsetin`	A dataframe containing columns for reporting.
`varsin`	A vector of variables on the reporting dataframe that will be displayed in the output.
`widths`	Provide a set of default widths for some or all variables as desired. (These will be used.)
`type`	Specify the type of width to be computed and returned. (Currently supports PCT, which refers to relative widths not percentages.)

Value

A list of widths of the same size as the list specified by varsin.

Author(s)

Yongwei Wang, <yongwei.x.wang@viivhealthcare.com>
Chris Rook, <christopher.x.rook@gsk.com>

Examples

```
library(repfun)
library(dplyr)
rfenv <- if (exists('rfenv') && is.environment(get('rfenv'))){
             rfenv
         } else {
             rfenv <- new.env(parent = emptyenv())
             rfenv$G_DEBUG <- 0
             rfenv
         }
datdir <- file.path(tempdir(),"datdir"); dir.create(datdir,showWarnings=FALSE)
repfun::copydata(datdir)
repfun::rs_setup(D_POPDATA=repfun::adsl %>% dplyr::filter(SAFFL =='Y'),
         D_SUBJID=c("STUDYID","USUBJID"),
         R_DICTION=NULL,
         R_OTHERDATA=NULL,
          R_INPUTDATA=NULL,
          R_RAWDATA=NULL,
          R_SDTMDATA=NULL,
          R_ADAMDATA=datdir)
G_POPDATA <- rfenv$G_POPDATA %>%
    dplyr::mutate(
         TRT01AN=ifelse(TRT01A=='Placebo',1,
                 ifelse(TRT01A=='Xanomeline Low Dose',2,3))) %>%
    repfun::ru_labels(varlabels=list('TRT01AN'='Actual Treatment for Period 01 (n)'))
adae <- rfenv$adamdata$adae.rda() %>%
        dplyr::inner_join(G_POPDATA, by=c('STUDYID','USUBJID','SAFFL','TRT01A'))
aesum_p <- repfun::ru_freq(adae,
                   dsetindenom=G_POPDATA,
                   countdistinctvars=c('STUDYID','USUBJID'),
                   groupbyvarsnumer=c('TRT01AN','TRT01A','AEBODSYS','AEDECOD'),
                   anyeventvars = c('AEBODSYS','AEDECOD'),
                   anyeventvalues = c('ANY EVENT','ANY EVENT'),
                   groupbyvarsdenom=c('TRT01AN'),
                   resultstyle="NUMERPCT",
                   totalforvar=c('TRT01AN'),
                   totalid=99,
                   totaldecode='Total',
                   codedecodevarpairs=c("TRT01AN", "TRT01A"),
                   varcodelistpairs=c(""),
                   codelistnames=list(),
                   resultpctdps=0) %>%
                   repfun::ru_denorm(varstodenorm=c("tt_result", "PERCENT"),
                             groupbyvars=c("tt_summarylevel", "AEBODSYS", "AEDECOD"),
                             acrossvar="TRT01AN", acrossvarlabel="TRT01A",
                             acrossvarprefix=c("tt_ac", "tt_p")) %>%
                   dplyr::mutate(ord1=ifelse(tt_summarylevel==0,0,1)) %>%
                   dplyr::rename(ord2=tt_summarylevel) %>%
                   dplyr::arrange(ord1,AEBODSYS,ord2,AEDECOD) %>%
                   dplyr::select(-c(starts_with('tt_p'),starts_with('ord'))) %>%
                   repfun::ru_addpage(grpvars=c('AEBODSYS'),rowsprbdy=35,nosplitvars=TRUE)

widths1 <- repfun::ru_width_rtf(aesum_p,
           c('AEBODSYS','AEDECOD','tt_ac01','tt_ac02','tt_ac03','tt_ac99'))
print(widths1)
widths2 <- repfun::ru_width_rtf(aesum_p,
           c('AEBODSYS','AEDECOD','tt_ac01','tt_ac02','tt_ac03','tt_ac99'),
           list('AEBODSYS'=35, 'AEDECOD'=30))
print(widths2)
```

`setpath` *Set the current working directory.*

Description

Set the current working directory so that relative paths work as expected.

Usage

```
setpath(p)
```

Arguments

`p`	A path as a string.

Value

No return value, the current working directory is set.

Author(s)

Yongwei Wang, `<yongwei.x.wang@viivhealthcare.com>`
Chris Rook, `<christopher.x.rook@gsk.com>`

Examples

```
## Not run:
library(repfun)
repfun::setpath()

## End(Not run)
```

`suppae` *Supplemental Adverse Events*

Description

A SDTM SUPPAE dataset from the CDISC pilot project

Usage

```
suppae
```

Format

A data frame with 10 columns:

STUDYID Study Identifier
RDOMAIN Related Domain Abbreviation
USUBJID Unique Subject Identifier
IDVAR Identifying Variable
IDVARVAL Identifying Variable Value
QNAM Qualifier Variable Name
QLABEL Qualifier Variable Label
QVAL Data Value
QORIG Origin
QEVAL Evaluator

Details

Supplemental Adverse Events

A SDTM SUPPAE dataset from the CDISC pilot project

Source

Access the source of the Supplemental Adverse Events dataset.

`suppdm` *Supplemental Demography*

Description

A SDTM SUPPDM dataset from the CDISC pilot project

Usage

```
suppdm
```

Format

A data frame with 10 columns:

STUDYID Study Identifier

RDOMAIN Related Domain Abbreviation

USUBJID Unique Subject Identifier

IDVAR Identifying Variable

IDVARVAL Identifying Variable Value

QNAM Qualifier Variable Name

QLABEL Qualifier Variable Label

QVAL Data Value

QORIG Origin

QEVAL Evaluator

Details

Supplemental Demography

A SDTM SUPPDM dataset from the CDISC pilot project

Source

Generated dataset.

Index