Package ‘cdashQC’

November 21, 2016

Title Quality control for TFLs

Version 0.1.2

Depends R(>= 3.1.1)

Imports lubridate(>= 1.6.0),
reshape2(>= 1.4.1),
dplyr(>= 0.5.0),
haven(>= 0.2.1),
knitr(>= 1.14)

Author Bin Zhuo

Maintainer Bin Zhuo <bin.zhuo@celerion.com>

Description This package creates table and listings for QC of cdash format data.

License What license is it under?

LazyData TRUE

RoxygenNote 5.0.1

NeedsCompilation no

R topics documented:

`clean_lab` *Clean the lab test data set.*

Description

clean the lab data, handling rechecks/unscheduled included.

Usage

```
clean_lab(lab, ex, included)
```

Arguments

`lab`	the dataset returned by `create_lab`
`ex`	the ex data set.
`included`	the included data set.

Value

lab data with necessary variables kept for further summarization.

See Also

`create_lab` and `create_included`

`codes` *the data set codes.*

Description

A dataset containing the lab test code and its corresponding full name

Usage

```
codes
```

Format

a data frame containing all possible `LB_TESTC` and the corresponding full names.

TESTNUM Maybe the number code?

LB_TESTC the code for laboratory test

TTL the full name

COLWIDE The column width? ...

Source

https://github.com/statswork/cdashQC/tree/master/data-raw/

`count_percent` *add percentage to a frequency table*

Description

count to percent

Usage

```
count_percent(data, var1, var2, digit_keep = 3)
```

Arguments

`data`	the data should contain at least 2 columns
`var1`	which column should be the group
`var2`	which columns are used to calculate percentage (could be a vector)
`digit_keep`	how many digits should be kept.

Value

a data frame

Examples

```
trait = rep(c("A", "B", "C"), 3); subtype = paste(trait, rep(1:3, each=3), sep = "")
data <- data.frame(trait, subtype, count1 = rpois(9, 5), count2 = rpois(9, 10))
count_percent(data, var1= 1, var2 = 3:4)

r1 <- get_summary_stats(eg, group = "PERIOD", var = "EG_TEST")
count_percent(r1, var1 = 1, var2 = 3:5)
```

create_aet   *do what createaet does.*

Description

createaet.

Usage

```
create_aet(ae, ex, included, improv = 99)
```

Arguments

ae	the dataset ae read from sas
ex	the dataset ex read from sas
included	the dataset included from sas, can be created using `create_included()`
improv	the same argument as the sas macro `createaet`

Value

aet the data

create_baseline   *create baseline indicator.*

Description

create an extra column inidcating whether the current row should be considered as baseline.

Usage

```
create_baseline(data, var_identifier = "_TEST")
```

Arguments

data	currently support `vs`, eg and `lb_cq`.
var_identifier	a string that can be used to identify the variable name e.g.(`VS_TEST, EG_TEST, LB_TEST`)

Value

a data frame with an extra column `status` whose value could be one of `BASELINE`, `POSTDOSE` and `PREDOSE (NOT BASELINE)`

See Also

`guess_base_phour`

`create_dem` *create the demographics data.*

Usage

```
create_dem(dm, ex, vs, included)
```

Arguments

`dm`	the dm data set
`ex`	the ex data set
`vs`	the vs data set
`inlcuded`	the included data set created by `create_included`

Value

a data frame

See Also

`create_included`

`create_eg` *create variable columns for the ECGs parameters.*

Description

dcast the ecg variables

Usage

```
create_eg(eg)
```

Arguments

`eg`	the dataset eg read from sas

Value

eg1 data with necessary variables kept for further summarization.

`create_included` *do what new_create_included does.*

Description

create_included.

Usage

```
create_included(ex, dm, cr, ds)
```

Arguments

`ex`	the dataset ex read from sas
`dm`	the dataset dm read from sas
`cr`	the dataset cr read from sas
`ds`	the dataset ds read from sas

Value

the included data set

`create_lab` *create the lab test data set.*

Description

create the lab data

Usage

```
create_lab(lb_cq)
```

Arguments

`lb_cq`	the dataset lb_cq read from sas

Value

lab data with necessary variables kept for further summarization.

create_phour *Create protocol hour*

Description

Create protocol hour variable if it's not already in the data set.

Usage

```
create_phour(data)
```

Arguments

data	either eg, vs or lb_cq

Value

the same data set with one more column if Phour is created, otherwise returns the input data.

create_vs *create the vital sign data set.*

Description

create the vs data

Usage

```
create_vs(vs)
```

Arguments

vs	the dataset dm read from sas

Value

vs data with necessary variables kept for further summarization.

get_summary_stats *get summary statistics*

Usage

```
get_summary_stats(data, group = "EX_TRT_C", var = "race", na.rm = TRUE)
```

Arguments

data	the data
group	which column should be the group
var	which columns are used to summarize
na.rm	should missing value be removed? TRUE by default.

Value

a data frame

Examples

```
SEQ = rep(c("A", "B", "C"), 3); subtype = sample(c("ONE", "TWO", "THREE"), 9, replace = TRUE)
data <- data.frame(SEQ, subtype, BMI = rnorm(9, 25, 4), HEIGHT = rnorm(9, 175,3))
get_summary_stats(data, group = "SEQ", var = "subtype")
get_summary_stats(data, group = "SEQ", var = "BMI")
```

guess_base_phour *find the baseline hour for each test*

Description

find the baseline hour for each test category

Usage

```
guess_base_phour(data, var_identifier = "_TEST")
```

Arguments

data	currently support vs, eg and lb_cq.
var_identifier	a string that can be used to identify the variable name e.g.(VS_TEST, EG_TEST, LB_TEST)

Value

a data frame where by-subject baseline hour is determined.

Examples

```
# the following two will give you exactly the same result
r1 <- guess_base_phour(vs)
r2 <- guess_base_phour(vs, var_identifier = "VS_TEST")

# If you want to find baseline hours for eg, the following two methods are equivalent
r3 <- guess_base_phour(eg)
r4 <- guess_base_phour(eg, var_identifier = "EG_TEST")
```

`guess_race_age` *race and ethnicity indicator, age.*

Description

create necessary variable for the demographic summary tables and listing.

Usage

```
guess_race_age(dm, ex)
```

Arguments

`dm`	the dataset dm read from sas
`ex`	the dataset ex read from sas

Value

a data frame with additional columns listed as follows

`race`	the race of the subject
`ethnic`	has two levels, "NOT HISPANIC OR LATINO" and "HISPANIC OR LATINO".
`EX_TRT_C`	the treatment groups
`ptno`	convert CLIENTID to numerical values of subject number
`age`	Age calculated from start date of treatment

`guess_reps` *guess the number of replicates*

Description

guess the number of replicates for each PERIOD/PHOUR/TESTCODE

Usage

```
guess_reps(data, var_identifier = "_TEST")
```

Arguments

data currently support eg and vs

var_identifier,
which columns is the variable for test code names.

Value

a data frame containing number of reps per PERIOD/PHOUR/TESTCODE

Examples

```
d1 <- guess_reps(eg)
```

guess_test *guess the name of the test*

Description

guess the name of the test code: it should end with var_identifier.

Usage

```
guess_test(data, var_identifier = "_TEST")
```

Arguments

data currently support eg and vs

var_identifier,
which columns is the variable for test code names.

Value

the test code

Examples

```
guess_test(eg)
```

`lab_oor` *Find OOR*

Description

Find the out-of-range values

Usage

```
lab_oor(lab)
```

Arguments

`lab`	the data returned by `create_lab`.

Value

a list containg OOR.

`listing_ae` *list Adverse Envent*

Description

list the ae

Usage

```
listing_ae(aet, type = 1)
```

Arguments

`aet`	created by `create_aet`
`type`	an indicator. Should I list `ae1 (type = 1)`, `ae2 (type = 2)` or `ae3 (type = 3)`?

Value

a data frame

See Also

`create_aet`

`listing_concom` *list concomitant.*

Description

list concomitant medication data

Usage

```
listing_concom(cm)
```

Arguments

`cm`	the dataset cm read from sas

Value

the data with related columns

`listing_dem` *demographics listing.*

Description

Summarize the demographic data

Usage

```
listing_dem(dmt)
```

Arguments

`dmt`	the data set created by `create_dem`.

Value

a data frame

See Also

`create_included` and `create_dem`

`normal_range` *Find normal range for lab test*

Description

Find the normal range of each test code

Usage

```
normal_range(lab)
```

Arguments

`lab`	the dataset generated by `create_lab`

Value

a data frame containing test codes and their corresponding normal range

`replicate_average` *get the averanges of the replicates for `vs` or `eg` data.*

Description

Get the averages of the replicates

Usage

```
replicate_average(data_clean, included, digits = NULL, na_rm = TRUE)
```

Arguments

`data_clean`	an object returned from `replicate_clean`
`included`	the included data set created by `create_included`
`digits`	should the averages be rounded? Default NO.
`na_rm`	should missing values be excluede? Default `TRUE`.

Value

the averages

See Also

`replicate_clean`

Examples

```
eg2 <- replicate_data(eg) # find the triplicates
```

replicate_check *check the replicates and issue a message if there's issue*

Description

check data qualities, reporting missing and RECHECKS

Usage

```
replicate_check(data, reps = NULL, printed = TRUE)
```

Arguments

`data` currently support `eg` and `vs`

`reps` the number of replicates planned. (should be tabled by TESTCODE, PERIOD and PHOUR). If `NULL`, `guess_reps` will be invoked to guess the number of replicates.

`printed` whether to print the observations having issues at the console.

Value

a data frame containing subject and issues at given period and protocol hours

See Also

`guess_reps`.

Examples

```
d1 <- replicate_check(eg)
```

replicate_clean *get the cleaned replicates by combining the "clean" data and "dirty" data*

Description

get cleaned replicates

Usage

```
replicate_clean(data, rm_row = NULL)
```

Arguments

`data` an object returned from `replicate_data` (works for vs and eg)

`rm_row` a vector of integers specifying which rows should be removed from the dirty data.

Value

the cleaned replicates

See Also

replicate_data

Examples

```
eg2 <- replicate_data(eg) # step 1: find the triplicates
eg_prob <- eg2$data_dirty         # need manual check
# the following rows should be removed
rows_removed <- c(2, 4, 7, 13, 14, 15, 19, 25, 28, 32, 37, 40, 44, 52, 57, 59, 64, 65)
eg3 <- replicate_clean(eg2, rows_removed)
```

---

replicate_data *get the replicates for eg or vs data.*

Description

get the replicates for each protocol hour (PHOUR)

Usage

```
replicate_data(data, reps = NULL)
```

Arguments

data	the data set created by create_eg (for ecg data) or by create_vs (for vital signs data).
reps	specifies the structures of reps. If not specified, guess_resp will be invoked. Default set to be NULL.

Value

a list

data_clean	the subjects containing correct number of resplicates
data_dirty	the subjects that have different number of replicates than desired

See Also

create_eg, guess_resp and replicate_check

`round_df` *round data frame*

Description

round a data frame if the column is numerical

Usage

```
round_df(df, digits = 3)
```

Arguments

`df`	the data frame
`digits`	how many digits you want to keep

Value

a data frame with numerical columns rounded.

`summary_dem` *demographic summary.*

Description

Summarize the demographic data

Usage

```
summary_dem(dmt, group = "EX_TRT_C", na.rm = TRUE)
```

Arguments

`dmt`	the data set created by `create_dem`.
`group`	which variable name would be used to calculate summary statistics?
`na.rm`	should missing values be included? `TRUE` by default.

Value

a data frame

See Also

`create_included` and `create_dem`

Examples

```
included <- create_included(ex, dm, cr, ds)
dmt <- create_dem(dm, ex, vs, included)
summary_dem(dmt, group = "SEQ")   # the summary by group
summary_dem(dmt, group = "SPONSOR")  # to get the overall summary
```

summary_lab *summary statistics for lab shift.*

Description

Summarize the lab statistics

Usage

```
summary_lab(lb_clean, digits = 3)
```

Arguments

lb_clean	the dataset returned by clean_lab.
digits	how many digits should be kept for the final data.

Value

the summary statistics by test code, time point and by treatment.

See Also

clean_lab

Examples

```
included <- create_included(ex, dm, cr, ds)
lab <- create_lab(lb_cq)
lb_clean <- clean_lab(lab, included= included, ex= ex)
s1 <- summary_lab(clean_lab)  #not urinalysis
```

summary_labshift *create the lab shift table.*

Description

Create the lab shift table.

Usage

```
summary_labshift(lb_clean)
```

Arguments

lb_clean	the dataset returned by clean_lab.

Value

the shift table

See Also

clean_lab

Examples

```
included <- create_included(ex, dm, cr, ds)
lab <- create_lab(lb_cq)
lb_clean <- clean_lab(lab, included, ex)
lbshift <- summary_labshift(lb_clean)
```

summary_vs_eg *summary statistics*

Description

Summary statistics for vital signs or eg

Usage

```
summary_vs_eg(data_clean, included, inter_digit = NULL, final_digits = 3,
  na_rm = TRUE, ischangefrombase = FALSE)
```

Arguments

`data_clean` the clean data set returned by `replicate_clean`.

`included` the included data set created by `create_included`

`inter_digit` if rounding happens for the intermediate averages, what digits should be kept.

`na_rm` should missing values be excluede? Default set to be `TRUE`.

`ischangefrombase` Is this summary for change from baseline? Default set to be `FALSE`

`final_digit` what is the digit for final summary?

Value

a data frame with summary statistics by test, time point and treatment.

`weight_height_bmi` *read BMI Weight and Height info from Screening stage.*

Description

extract BMI, weight and height from vs.sas7bat.

Usage

```
weight_height_bmi(vs)
```

Arguments

`vs`	the vs sas data

Value

a data frame containing BMI, WEIGHT and HEIGHT from admission stage

Index