

Package ‘dbGaPCheckup’

February 22, 2023

Type Package

Title dbGaP Checkup

Version 1.0.2

URL <https://lwheinsberg.github.io/dbGaPCheckup/>,
<https://github.com/lwheinsberg/dbGaPCheckup>

Description Contains functions that check for formatting of the Subject Phenotype data set and data dictionary as specified by the National Center for Biotechnology Information (NCBI) Database of Genotypes and Phenotypes (dbGaP) <<https://www.ncbi.nlm.nih.gov/gap/docs/submissionguide/>>.

Imports readxl, tidyr, dplyr, formatR, ggplot2, pander, magrittr, rmarkdown, purrr, questionr, tibble, rlang, labelled, stats, utils, graphics

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Suggests testthat, knitr

BugReports <https://github.com/lwheinsberg/dbGaPCheckup/issues>

Depends R (>= 3.5.0)

VignetteBuilder knitr, rmarkdown, formatR

NeedsCompilation no

Author Lacey W. Heinsberg [aut, cre],
Daniel E. Weeks [aut],
University of Pittsburgh [cph]

Maintainer Lacey W. Heinsberg <law145@pitt.edu>

Repository CRAN

Date/Publication 2023-02-22 18:10:03 UTC

R topics documented:

| | |
|-----------------------------------|----|
| add_missing_fields | 4 |
| check_report | 5 |
| complete_check | 6 |
| create_awareness_report | 7 |
| create_report | 8 |
| dat_function | 9 |
| dat_function_selected | 10 |
| DD.dict.A | 11 |
| DD.dict.B | 11 |
| DD.dict.C | 11 |
| DD.dict.D | 12 |
| DD.dict.E | 12 |
| DD.dict.F | 12 |
| DD.dict.G | 13 |
| DD.dict.H | 13 |
| DD.dict.I | 13 |
| DD.dict.J | 14 |
| DD.dict.K | 14 |
| DD.dict.L | 14 |
| DD.dict.M | 15 |
| DD.dict.N | 15 |
| DD.dict.Q | 15 |
| decimal_check | 16 |
| description_check | 17 |
| dictionary_search | 17 |
| dimension_check | 18 |
| DS.data.A | 19 |
| DS.data.B | 19 |
| DS.data.C | 20 |
| DS.data.D | 20 |
| DS.data.E | 20 |
| DS.data.F | 21 |
| DS.data.G | 21 |
| DS.data.H | 21 |
| DS.data.I | 22 |
| DS.data.J | 22 |
| DS.data.K | 22 |
| DS.data.L | 23 |
| DS.data.M | 23 |
| DS.data.N | 23 |
| DS.data.O | 24 |
| DS.data.P | 24 |
| DS.data.Q | 24 |
| dup_values | 25 |
| eval_function | 25 |
| ExampleA | 26 |

| | |
|--------------------------------|----|
| ExampleB | 26 |
| ExampleC | 27 |
| ExampleD | 27 |
| ExampleE | 28 |
| ExampleF | 29 |
| ExampleG | 29 |
| ExampleH | 30 |
| ExampleI | 30 |
| ExampleJ | 31 |
| ExampleK | 32 |
| ExampleL | 32 |
| ExampleM | 33 |
| ExampleN | 33 |
| ExampleO | 34 |
| ExampleP | 35 |
| ExampleQ | 35 |
| field_check | 36 |
| id_check | 36 |
| id_first_data | 37 |
| id_first_dict | 38 |
| integer_check | 39 |
| int_check | 39 |
| label_data | 40 |
| minmax_check | 41 |
| misc_format_check | 42 |
| missingness_summary | 43 |
| missing_value_check | 43 |
| mm_precheck | 44 |
| mv_precheck | 45 |
| name_check | 46 |
| name_correct | 46 |
| name_precheck | 47 |
| NA_check | 48 |
| NA_precheck | 49 |
| pkg_field_check | 49 |
| reorder_data | 50 |
| reorder_dictionary | 51 |
| row_check | 52 |
| short_field_check | 52 |
| short_precheck | 53 |
| super_short_precheck | 54 |
| type_check | 54 |
| values_check | 55 |
| values_precheck | 56 |
| value_meaning_table | 57 |
| value_missing_table | 57 |

add_missing_fields *Add Missing Fields*

Description

This function adds additional fields required by this package including variable type (TYPE), minimum value (MIN), and maximum value (MAX).

Usage

```
add_missing_fields(DD.dict, DS.data)
```

Arguments

| | |
|---------|------------------|
| DD.dict | Data dictionary. |
| DS.data | Data set. |

Details

Even though MIN, MAX, and TYPE are not required by dbGaP, our package was created to use these variables in a series of other checks and awareness functions (e.g., render_report, values_check, etc.). MIN/MAX columns will be added as empty columns as dbGaP instructions state that the MIN and MAX should be the "logical" MIN/MAX for the data, not necessarily the observed MIN/MAX, which would be study and variable specific. TYPE will be inferred from the data set and data dictionary VALUES columns. Note however, that if the VALUES columns are not set up correctly, then this function can't properly infer the data TYPE from the data set and data dictionary.

Value

A data frame containing the updated data dictionary with missing fields added in, or NULL if any required pre-checks fail.

Examples

```
# Example
data(ExampleD)
DD.dict.updated <- add_missing_fields(DD.dict.D, DS.data.D)
```

| | |
|--------------|---------------------|
| check_report | <i>Check Report</i> |
|--------------|---------------------|

Description

This function generates a user-readable report of the checks run by the `complete_check` function.

Usage

```
check_report(DD.dict, DS.data, non.NA.missing.codes = NA, compact = TRUE)
```

Arguments

| | |
|----------------------|---|
| DD.dict | Data dictionary. |
| DS.data | Data set. |
| non.NA.missing.codes | A user-defined vector of numerical missing value codes (e.g., -9999). |
| compact | When TRUE, the function prints a compact report, listing information from only the non-passed checks. |

Value

Tibble, returned invisibly, containing the following information for each check: (1) Time (Time stamp); (2) Name (Name of the function); (3) Status (Passed/Failed); (4) Message (A copy of the message the function printed out); (5) Information (More detailed information about the potential errors identified).

See Also

[complete_check](#)

Examples

```
# Example 1: Incorrectly showing as pass check on first attempt
data(ExampleB)
report <- check_report(DD.dict.B, DS.data.B)
# Addition of missing value codes calls attention to error
# at missing_value_check
report <- check_report(DD.dict.B, DS.data.B, non.NA.missing.codes=c(-4444, -9999))

# Example 2: Several fail checks or not attempted
data(ExampleC)
report <- check_report(DD.dict.C, DS.data.C, non.NA.missing.codes=c(-4444, -9999))
# Note you can also run report using compact=FALSE
report <- check_report(DD.dict.C, DS.data.C, non.NA.missing.codes=c(-4444, -9999), compact = FALSE)
```

| | |
|----------------|-----------------------|
| complete_check | <i>Complete Check</i> |
|----------------|-----------------------|

Description

This function runs a full workflow check including `field_check`, `pkg_field_check`, `dimension_check`, `name_check`, `id_check`, `row_check`, `NA_check`, `type_check`, `values_check`, `integer_check`, `decimal_check`, `misc_format_check`, `description_check`, `minmax_check`, and `missing_value_check`.

Usage

```
complete_check(
  DD_dict,
  DS_data,
  non.NA.missing.codes = NA,
  reorder.dict = FALSE,
  name.correct = FALSE
)
```

Arguments

| | |
|-----------------------------------|--|
| <code>DD_dict</code> | Data dictionary. |
| <code>DS_data</code> | Data set. |
| <code>non.NA.missing.codes</code> | A user-defined vector of encoded, numerical (i.e., non-NA) missing value codes (e.g., -9999). |
| <code>reorder.dict</code> | When TRUE, and only if the names between the data and data dictionary match perfectly but are in the wrong order, the function will reorder the rows of the dictionary to match the columns of the data; note please use with caution: we recommend first running the function with the default set to FALSE to understand potential errors. |
| <code>name.correct</code> | When TRUE, if name mismatches are identified, the function will rename the variable names in the data set to match the data dictionary; note please use with caution: we recommend first running the function with the default set to FALSE to identify order/dimension mismatches (vs. name mismatches). |

Value

Tibble containing the following information for each check: (1) Time (time stamp); (2) Name (name of the function); (3) Status (Passed/Failed/Warning); (4) Message (A copy of the message the function printed out); (5) Information (More detailed information about the potential errors identified).

See Also

[check_report](#)

Examples

```
# Example 1
# Note in this example, the missing value codes are not defined,
# so the last check ('missing_value_check') doesn't know to
# to check for encoded values
data(ExampleB)
complete_check(DD.dict.B, DS.data.B)
# Rerun check after defining missing value codes
complete_check(DD.dict.B, DS.data.B, non.NA.missing.codes=c(-9999, -4444))

# Example 2
data(ExampleA)
complete_check(DD.dict.A, DS.data.A, non.NA.missing.codes=c(-9999, -4444))

# Example 3
data(ExampleD)
results <- complete_check(DD.dict.D, DS.data.D, non.NA.missing.codes=c(-9999, -4444))
# View output in greater detail
results$Message[2] # Recommend using add_missing_fields
results$Information$pkg_field_check.Info # We see that MIN, MAX, and TYPE are all missing
# Use the add_missing_fields function to add in data
DD.dict.updated <- add_missing_fields(DD.dict.D, DS.data.D)
# Be sure to call in the new version of the dictionary (DD.dict.updated)
complete_check(DD.dict.updated, DS.data.D)
```

create_awareness_report

Create Awareness Report

Description

This function generates an awareness report in HTML format, and optionally opens it in the web browser.

Usage

```
create_awareness_report(  
  DD.dict,  
  DS.data,  
  non.NA.missing.codes = NA,  
  threshold = 95,  
  output.path = tempdir(),  
  open.html = TRUE,  
  fn.stem = "AwarenessReport"  
)
```

Arguments

| | |
|----------------------|---|
| DD.dict | Data dictionary. |
| DS.data | Data set. |
| non.NA.missing.codes | A user-defined vector of numerical missing value codes (e.g., -9999). |
| threshold | Threshold for missingness of concern (as a percent). |
| output.path | Path to the folder in which to create the HTML report document. |
| open.html | If TRUE, open the HTML report document in the web browser. |
| fn.stem | File name stem. |

Value

Full path to the HTML report document.

See Also

[value_missing_table](#)

[missingness_summary](#)

Examples

```
data(ExampleB)
create_awareness_report(DD.dict.B, DS.data.B, non.NA.missing.codes=c(-9999),
  output.path= tempdir(), open.html = FALSE)
```

create_report

Create Report

Description

This function calls `eval_function` to generate a textual and graphical report of the selected variables in HTML format, and optionally opens it in the web browser.

Usage

```
create_report(
  DD.dict,
  DS.data,
  sex.split = FALSE,
  sex.name = NULL,
  start = 1,
  end = 1,
  non.NA.missing.codes = NA,
  output.path = tempdir(),
```



```

    open.html = TRUE,
    fn.stem = "Report"
  )

```

Arguments

| | |
|----------------------|--|
| DD.dict | Data dictionary. |
| DS.data | Data set. |
| sex.split | When TRUE, split reports by the field named as defined by the sex.name variable. |
| sex.name | Character string specifying the name of the sex field. |
| start | Starting index of the first select trait. |
| end | Ending index of the last selected trait. |
| non.NA.missing.codes | A user-defined vector of numerical missing value codes (e.g., -9999). |
| output.path | Path to the folder in which to create the HTML report document. |
| open.html | If TRUE, open the HTML report document in the web browser. |
| fn.stem | File name stem. |

Value

Full path to the HTML report document.

Examples

```

data(ExampleB)
create_report(DD.dict.B, DS.data.B, sex.split=TRUE, sex.name= "SEX",
  start = 3, end = 7, non.NA.missing.codes=c(-9999,-4444),
  output.path= tempdir(), open.html = FALSE)

```

dat_function

Data Utility Function

Description

This function calls eval_function to generate a textual and graphical report of the selected variables.

Usage

```

dat_function(
  DS.dataset,
  DD.dictionary,
  sex.split = FALSE,
  sex.name = NULL,
  DS.dataset.na
)

```

Arguments

| | |
|---------------|---|
| DS.dataset | Data set. |
| DD.dictionary | Data dictionary. |
| sex.split | When TRUE, split reports by the field named by the sex.name string. |
| sex.name | Character string giving the name of the sex field. |
| DS.dataset.na | Data set with missing values set to NA. |

Value

Invisible NULL, called for its side effects.

dat_function_selected *Data Selected Utility Function*

Description

This function calls eval_function to generate a textual and graphical report of the selected variables.

Usage

```
dat_function_selected(
  dataset,
  dictionary,
  sex.split = FALSE,
  sex.name = NULL,
  start = 1,
  end = 1,
  dataset.na,
  h.level = 2
)
```

Arguments

| | |
|------------|--|
| dataset | Data set. |
| dictionary | Data dictionary. |
| sex.split | When TRUE, split reports by the field named 'Sex'. |
| sex.name | Character string giving the name of the sex field. |
| start | Starting index of the first selected trait. |
| end | Ending index of the last selected trait. |
| dataset.na | Data set with missing values set to NA. |
| h.level | Header level for pandoc function. |

Value

Invisible NULL, called for its side effects

DD.dict.A

DD.dict.A

Description

Data dictionary embedded in ExampleA.

Usage

`data(ExampleA)`

See Also

[ExampleA](#)

DD.dict.B

DD.dict.B

Description

Data dictionary embedded in ExampleB.

Usage

`data(ExampleB)`

See Also

[ExampleB](#)

DD.dict.C

DD.dict.C

Description

Data dictionary embedded in ExampleC.

Usage

`data(ExampleC)`

See Also

[ExampleC](#)

DD.dict.D

DD.dict.D

Description

Data dictionary embedded in ExampleD.

Usage

data(ExampleD)

See Also

[ExampleD](#)

DD.dict.E

DD.dict.E

Description

Data dictionary embedded in ExampleE.

Usage

data(ExampleE)

See Also

[ExampleE](#)

DD.dict.F

DD.dict.F

Description

Data dictionary embedded in ExampleF.

Usage

data(ExampleF)

See Also

[ExampleF](#)

`DD.dict.G`*DD.dict.G*

Description

Data dictionary embedded in ExampleG.

Usage

```
data(ExampleG)
```

See Also

[ExampleG](#)

`DD.dict.H`*DD.dict.H*

Description

Data dictionary embedded in ExampleH.

Usage

```
data(ExampleH)
```

See Also

[ExampleH](#)

`DD.dict.I`*DD.dict.I*

Description

Data dictionary embedded in ExampleI.

Usage

```
data(ExampleI)
```

See Also

[ExampleI](#)

DD.dict.J

DD.dict.J

Description

Data dictionary embedded in ExampleJ.

Usage

```
data(ExampleJ)
```

See Also

[ExampleJ](#)

DD.dict.K

DD.dict.K

Description

Data dictionary embedded in ExampleK.

Usage

```
data(ExampleK)
```

See Also

[ExampleK](#)

DD.dict.L

DD.dict.L

Description

Data dictionary embedded in ExampleL.

Usage

```
data(ExampleL)
```

See Also

[ExampleL](#)

DD.dict.M

DD.dict.M

Description

Data dictionary embedded in ExampleM.

Usage

data(ExampleM)

See Also

[ExampleM](#)

DD.dict.N

DD.dict.N

Description

Data dictionary embedded in ExampleN.

Usage

data(ExampleN)

See Also

[ExampleN](#)

DD.dict.Q

DD.dict.Q

Description

Data dictionary embedded in ExampleQ.

Usage

data(ExampleQ)

See Also

[ExampleQ](#)

| | |
|---------------|----------------------|
| decimal_check | <i>Decimal Check</i> |
|---------------|----------------------|

Description

This function searches for variables that appear to be incorrectly listed as TYPE decimal.

Usage

```
decimal_check(DD.dict, DS.data, verbose = TRUE)
```

Arguments

| | |
|---------|---|
| DD.dict | Data dictionary. |
| DS.data | Data set. |
| verbose | When TRUE, the function prints the Message out, as well as a list of variables that may be incorrectly labeled as TYPE decimal. |

Value

Tibble, returned invisibly, containing: (1) Time (Time stamp); (2) Name (Name of the function); (3) Status (Passed/Failed); (4) Message (A copy of the message the function printed out); (5) Information (Names of variables that are listed as TYPE decimal, but do not appear to be decimals).

Examples

```
# Example 1: Fail check
data(ExampleF)
decimal_check(DD.dict.F, DS.data.F)
print(integer_check(DD.dict.F, DS.data.F, verbose=FALSE))

# Example 2: Required pre-check fails
data(ExampleE)
decimal_check(DD.dict.E, DS.data.E)
print(decimal_check(DD.dict.E, DS.data.E, verbose=FALSE))

# Example 3: Pass check
data(ExampleA)
decimal_check(DD.dict.A, DS.data.A)
print(decimal_check(DD.dict.A, DS.data.A, verbose=FALSE))
```

| | |
|-------------------|--------------------------|
| description_check | <i>Description Check</i> |
|-------------------|--------------------------|

Description

This function checks that there is a unique description for every variable in the data dictionary (VARDESC column).

Usage

```
description_check(DD.dict, verbose = TRUE)
```

Arguments

| | |
|---------|---|
| DD.dict | Data dictionary. |
| verbose | When TRUE, the function prints the Message out, as well as a list of the variables that are missing a VARDESC or have a duplicated VARDESC. |

Value

Tibble, returned invisibly, containing: (1) Time (Time stamp); (2) Name (Name of the function); (3) Status (Passed/Failed); (4) Message (A copy of the message the function printed out); (5) Information (Names of the variables with missing or duplicated descriptions).

Examples

```
# Example 1: Fail check
data(ExampleG)
description_check(DD.dict.G)
print(description_check(DD.dict.G, verbose=FALSE))

# Example 2: Pass check
data(ExampleA)
description_check(DD.dict.A)
print(description_check(DD.dict.A, verbose=FALSE))
```

| | |
|-------------------|-------------------------------|
| dictionary_search | <i>Data Dictionary Search</i> |
|-------------------|-------------------------------|

Description

This awareness function helps you search the data dictionary for a specific term; intended for use as an investigative aid to supplement other checks in this package.

Usage

```
dictionary_search(
  DD.dict,
  search.term = c("blood pressure"),
  search.column = c("VARDESC")
)
```

Arguments

DD.dict Data dictionary.
 search.term Search term.
 search.column Column of the data dictionary to search.

Value

Tibble containing dictionary rows in which the search term was detected in specified column or an error message if the search column could not be detected.

Examples

```
# Successful search
data(ExampleB)
dictionary_search(DD.dict.B, search.term=c("skinfold"), search.column=c("VARDESC"))
# Attempted search in wrong column
dictionary_search(DD.dict.B, search.term=c("skinfold"), search.column=c("VARIABLE_DESCRIPTION"))
```

| | |
|-----------------|------------------------|
| dimension_check | <i>Dimension Check</i> |
|-----------------|------------------------|

Description

This function checks that the number of variables match between the data set and the data dictionary.

Usage

```
dimension_check(DD.dict, DS.data, verbose = TRUE)
```

Arguments

DD.dict Data dictionary.
 DS.data Data set.
 verbose When TRUE, the function prints the Message out, as well as the number of variables in the data set and data dictionary.

Value

Tibble, returned invisibly, containing: (1) Time (Time stamp); (2) Name (Name of the function); (3) Status (Passed/Failed); (4) Message (A copy of the message the function printed out); (5) Information (number of variables in the data and dictionary and names of mismatched variables if applicable).

Examples

```
# Example 1: Fail check
data(ExampleG)
dimension_check(DD.dict.G, DS.data.G)
print(dimension_check(DD.dict=DD.dict.G, DS.data=DS.data.G, verbose=FALSE))

# Example 2: Pass check
data(ExampleA)
dimension_check(DD.dict.A, DS.data.A)
print(dimension_check(DD.dict.A, DS.data.A, verbose=FALSE))
```

DS.data.A

DS.data.A

Description

Data set embedded in ExampleA.

Usage

```
data(ExampleA)
```

See Also

[ExampleA](#)

DS.data.B

DS.data.B

Description

Data set embedded in ExampleB.

Usage

```
data(ExampleB)
```

See Also

[ExampleB](#)

DS.data.C

DS.data.C

Description

Data set embedded in ExampleC.

Usage

data(ExampleC)

See Also

[ExampleC](#)

DS.data.D

DS.data.D

Description

Data set embedded in ExampleD.

Usage

data(ExampleD)

See Also

[ExampleD](#)

DS.data.E

DS.data.E

Description

Data set embedded in ExampleE.

Usage

data(ExampleE)

See Also

[ExampleE](#)

DS.data.F

DS.data.F

Description

Data set embedded in ExampleF.

Usage

```
data(ExampleF)
```

See Also

[ExampleF](#)

DS.data.G

DS.data.G

Description

Data set embedded in ExampleG.

Usage

```
data(ExampleG)
```

See Also

[ExampleG](#)

DS.data.H

DS.data.H

Description

Data set embedded in ExampleH.

Usage

```
data(ExampleH)
```

See Also

[ExampleH](#)

DS.data.I

DS.data.I

Description

Data set embedded in ExampleI.

Usage

```
data(ExampleI)
```

See Also

[ExampleI](#)

DS.data.J

DS.data.J

Description

Data set embedded in ExampleJ.

Usage

```
data(ExampleJ)
```

See Also

[ExampleJ](#)

DS.data.K

DS.data.K

Description

Data set embedded in ExampleK.

Usage

```
data(ExampleK)
```

See Also

[ExampleK](#)

DS.data.L

DS.data.L

Description

Data set embedded in ExampleL.

Usage

data(ExampleL)

See Also

[ExampleL](#)

DS.data.M

DS.data.M

Description

Data set embedded in ExampleM.

Usage

data(ExampleM)

See Also

[ExampleM](#)

DS.data.N

DS.data.N

Description

Data set embedded in ExampleN.

Usage

data(ExampleN)

See Also

[ExampleN](#)

DS.data.O

DS.data.O

Description

Data set embedded in ExampleO.

Usage

```
data(ExampleO)
```

See Also

[ExampleO](#)

DS.data.P

DS.data.P

Description

Data set embedded in ExampleP.

Usage

```
data(ExampleP)
```

See Also

[ExampleP](#)

DS.data.Q

DS.data.Q

Description

Data set embedded in ExampleQ.

Usage

```
data(ExampleQ)
```

See Also

[ExampleQ](#)

| | |
|------------|----------------------------------|
| dup_values | <i>Duplicate Values Function</i> |
|------------|----------------------------------|

Description

This function checks for duplicate VALUES column names in the data dictionary.

Usage

```
dup_values(DD.dict)
```

Arguments

| | |
|---------|------------------|
| DD.dict | Data dictionary. |
|---------|------------------|

Value

Logical, TRUE if only one VALUES column is detected.

| | |
|---------------|------------------------------------|
| eval_function | <i>Evaluation Utility Function</i> |
|---------------|------------------------------------|

Description

This function generates a textual and graphical report of the selected variables.

Usage

```
eval_function(
  dataset,
  dictionary,
  sex.split = FALSE,
  sex.name = NULL,
  dataset.na,
  h.level = 2
)
```

Arguments

| | |
|------------|--|
| dataset | Data set. |
| dictionary | Data dictionary. |
| sex.split | When TRUE, split reports by the field named 'Sex'. |
| sex.name | Name of the Sex field. |
| dataset.na | Data set with missing values set to NA. |
| h.level | Header level for pandoc function. |

Value

Invisible NULL, called for its side effects.

ExampleA

ExampleA

Description

Example data set and data dictionary with no errors.

Usage

```
data(ExampleA)
```

Format

R data file that contains two objects:

DD.dict.A Data dictionary

DS.data.A Data set

Source

```
DD.path <- system.file("extdata", "3b_SSM_DD_Example1.xlsx", package = "dbGaPCheckup", mustWork=TRUE)
DD.dict.A <- readxl::read_xlsx(DD.path)
path <- system.file("extdata", "DS_Example.txt", package = "dbGaPCheckup", mustWork=TRUE)
DS.data.A <- read.table(DS.path, header=TRUE, sep="\t", quote="", as.is = TRUE)
save(DD.dict.A, DS.data.A, file = "ExampleA.rda")
```

ExampleB

ExampleB

Description

Example data set and data dictionary with intentional errors.

Usage

```
data(ExampleB)
```

Format

R data file that contains two objects:

DD.dict.B Data dictionary

DS.data.B Data set

Source

```
DD.path <- system.file("extdata", "3b_SSM_DD_Example1b.xlsx", package = "dbGaPCheckup", mustWork=TRUE)
DD.dict.B <- readxl::read_xlsx(DD.path)
DS.path <- system.file("extdata", "DS_Example1b.txt", package = "dbGaPCheckup", mustWork=TRUE)
DS.data.B <- read.table(DS.path, header=TRUE, sep="\t", quote="", as.is = TRUE)
save(DD.dict.B, DS.data.B, file = "ExampleB.rda")
```

ExampleC

ExampleC

Description

Example data set and data dictionary with intentional errors.

Usage

```
data(ExampleC)
```

Format

R data file that contains two objects:

DD.dict.C Data dictionary

DS.data.C Data set

Source

```
DD.path <- system.file("extdata", "3b_SSM_DD_Example2d.xlsx", package = "dbGaPCheckup", mustWork=TRUE)
DD.dict.C <- readxl::read_xlsx(DD.path)
DS.path <- system.file("extdata", "DS_Example1b.txt", package = "dbGaPCheckup", mustWork=TRUE)
DS.data.C <- read.table(DS.path, header=TRUE, sep="\t", quote="", as.is = TRUE)
save(DD.dict.C, DS.data.C, file = "ExampleC.rda")
```

ExampleD

ExampleD

Description

Example data set and data dictionary with intentional errors.

Usage

```
data(ExampleD)
```

Format

R data file that contains two objects:

DD.dict.D Data dictionary

DS.data.D Data set

Source

```
path <- system.file("extdata", "3b_SSM_DD_Example2f.xlsx", package = "dbGaPCheckup", mustWork=TRUE)
DD.dict.D <- readxl::read_xlsx(path)
DS.path <- system.file("extdata", "DS_Example.txt", package = "dbGaPCheckup", mustWork=TRUE)
DS.data.D <- read.table(DS.path, header=TRUE, sep="\t", quote="", as.is = TRUE)
save(DD.dict.D, DS.data.D, file = "ExampleD.rda")
```

ExampleE

ExampleE

Description

Example data set and data dictionary with intentional errors.

Usage

```
data(ExampleE)
```

Format

R data file that contains two objects:

DD.dict.E Data dictionary

DS.data.E Data set

Source

```
DD.path <- system.file("extdata", "3b_SSM_DD_Example2b.xlsx", package = "dbGaPCheckup", mustWork=TRUE)
DD.dict.E <- readxl::read_xlsx(DD.path)
DS.path <- system.file("extdata", "DS_Example2.txt", package = "dbGaPCheckup", mustWork=TRUE)
DS.data.E <- read.table(DS.path, header=TRUE, sep="\t", quote="", as.is = TRUE)
save(DD.dict.E, DS.data.E, file = "ExampleE.rda")
```

ExampleF

ExampleF

Description

Example data set and data dictionary with intentional errors.

Usage

```
data(ExampleF)
```

Format

R data file that contains two objects:

DD.dict.F Data dictionary

DS.data.F Data set

Source

```
DD.path <- system.file("extdata", "3b_SSM_DD_Example4.xlsx", package = "dbGaPCheckup", mustWork=TRUE)
DD.dict.F <- readxl::read_xlsx(DD.path)
DS.path <- system.file("extdata", "DS_Example3d.txt", package = "dbGaPCheckup", mustWork=TRUE)
DS.data.F <- read.table(DS.path, header=TRUE, sep="\t", quote="", as.is = TRUE)
save(DD.dict.F, DS.data.F, file = "ExampleF.rda")
```

ExampleG

ExampleG

Description

Example data set and data dictionary with intentional errors.

Usage

```
data(ExampleG)
```

Format

R data file that contains two objects:

DD.dict.G Data dictionary

DS.data.G Data set

Source

```
DD.path <- system.file("extdata", "3b_SSM_DD_Example2.xlsx", package = "dbGaPCheckup", mustWork=TRUE)
DD.dict.G <- readxl::read_xlsx(DD.path)
DS.path <- system.file("extdata", "DS_Example.txt", package = "dbGaPCheckup", mustWork=TRUE)
DS.data.G <- read.table(DS.path, header=TRUE, sep="\t", quote="", as.is = TRUE)
save(DD.dict.G, DS.data.G, file = "ExampleG.rda")
```

 ExampleH

ExampleH

Description

Example data set and data dictionary with intentional errors.

Usage

```
data(ExampleH)
```

Format

R data file that contains two objects:

DD.dict.H Data dictionary

DS.data.H Data set

Source

```
DD.path <- system.file("extdata", "3b_SSM_DD_Example1.xlsx", package = "dbGaPCheckup", mustWork=TRUE)
DD.dict.H <- readxl::read_xlsx(DD.path)
DS.path <- system.file("extdata", "DS_Example3c.txt", package = "dbGaPCheckup", mustWork=TRUE)
DS.data.H <- read.table(DS.path, header=TRUE, sep="\t", quote="", as.is = TRUE)
save(DD.dict.H, DS.data.H, file = "ExampleH.rda")
```

 ExampleI

ExampleI

Description

Example data set and data dictionary with intentional errors.

Usage

```
data(ExampleI)
```

Format

R data file that contains two objects:

DD.dict.I Data dictionary

DS.data.I Data set

Source

```
DD.path <- system.file("extdata", "3b_SSM_DD_Example2c.xlsx", package = "dbGaPCheckup", mustWork=TRUE)
DD.dict.I <- readxl::read_xlsx(DD.path)
DS.path <- system.file("extdata", "DS_Example2c.txt", package = "dbGaPCheckup", mustWork=TRUE)
DS.data.I <- read.table(DS.path, header=TRUE, sep="\t", quote="", as.is = TRUE)
save(DD.dict.I, DS.data.I, file = "ExampleI.rda")
```

ExampleJ

ExampleJ

Description

Example data set and data dictionary with intentional errors.

Usage

```
data(ExampleJ)
```

Format

R data file that contains two objects:

DD.dict.J Data dictionary

DS.data.J Data set

Source

```
DD.path <- system.file("extdata", "3b_SSM_DD_Example2d.xlsx", package = "dbGaPCheckup", mustWork=TRUE)
DD.dict.J <- readxl::read_xlsx(DD.path)
DS.path <- system.file("extdata", "DS_Example2.txt", package = "dbGaPCheckup", mustWork=TRUE)
DS.data.J <- read.table(DS.path, header=TRUE, sep="\t", quote="", as.is = TRUE)
save(DD.dict.J, DS.data.J, file = "ExampleJ.rda")
```

ExampleK

ExampleK

Description

Example data set and data dictionary with intentional errors.

Usage

```
data(ExampleK)
```

Format

R data file that contains two objects:

DD.dict.K Data dictionary

DS.data.K Data set

Source

```
DD.path <- system.file("extdata", "3b_SSM_DD_Example2d.xlsx", package = "dbGaPCheckup", mustWork=TRUE)
DD.dict.K <- readxl::read_xlsx(DD.path)
DS.path <- system.file("extdata", "DS_Example2b.txt", package = "dbGaPCheckup", mustWork=TRUE)
DS.data.K <- read.table(DS.path, header=TRUE, sep="\t", quote="", as.is = TRUE)
save(DD.dict.K, DS.data.K, file = "ExampleK.rda")
```

ExampleL

ExampleL

Description

Example data set and data dictionary with intentional errors.

Usage

```
data(ExampleL)
```

Format

R data file that contains two objects:

DD.dict.L Data dictionary

DS.data.L Data set

Source

```
DD.path <- system.file("extdata", "3b_SSM_DD_Example2b.xlsx", package = "dbGaPCheckup", mustWork=TRUE)
DD.dict.L <- readxl::read_xlsx(DD.path)
DS.path <- system.file("extdata", "DS_Example2c.txt", package = "dbGaPCheckup", mustWork=TRUE)
DS.data.L <- read.table(DS.path, header=TRUE, sep="\t", quote="", as.is = TRUE)
save(DD.dict.L, DS.data.L, file = "ExampleL.rda")
```

 ExampleM

ExampleM

Description

Example data set and data dictionary with intentional errors.

Usage

```
data(ExampleM)
```

Format

R data file that contains two objects:

DD.dict.M Data dictionary

DS.data.M Data set

Source

```
DD.path <- system.file("extdata", "3b_SSM_DD_Example2b.xlsx", package = "dbGaPCheckup", mustWork=TRUE)
DD.dict.M <- readxl::read_xlsx(DD.path)
DS.path <- system.file("extdata", "DS_Example.txt", package = "dbGaPCheckup", mustWork=TRUE)
DS.data.M <- read.table(DS.path, header=TRUE, sep="\t", quote="", as.is = TRUE)
save(DD.dict.M, DS.data.M, file = "ExampleM.rda")
```

 ExampleN

ExampleN

Description

Example data set and data dictionary with intentional errors.

Usage

```
data(ExampleN)
```

Format

R data file that contains two objects:

DD.dict.N Data dictionary

DS.data.N Data set

Source

```
DD.path <- system.file("extdata", "3b_SSM_DD_Example2e.xlsx", package = "dbGaPCheckup", mustWork=TRUE)
DD.dict.N <- readxl::read_xlsx(DD.path)
DS.path <- system.file("extdata", "DS_Example.txt", package = "dbGaPCheckup", mustWork=TRUE)
DS.data.N <- read.table(DS.path, header=TRUE, sep="\t", quote="", as.is = TRUE)
save(DD.dict.N, DS.data.N, file = "ExampleN.rda")
```

Example0

ExampleO

Description

Example data set with intentional errors.

Usage

```
data(Example0)
```

Format

R data file that contains a single object:

DS.data.O Data set

Source

```
DS.path <- system.file("extdata", "DS_Example3.txt", package = "dbGaPCheckup", mustWork=TRUE)
DS.data.O <- read.table(DS.path, header=TRUE, sep="\t", quote="", as.is = TRUE)
save(DS.data.O, file = "ExampleO.rda")
```

ExampleP

ExampleP

Description

Example data set with intentional errors.

Usage

```
data(ExampleP)
```

Format

R data file that contains a single object:

DS.data.P Data set

Source

```
DS.path <- system.file("extdata", "DS_Example3b.txt", package = "dbGaPCheckup", mustWork=TRUE)
DS.data.P <- read.table(DS.path, header=TRUE, sep="\t", quote="", as.is = TRUE)
save(DS.data.P, file = "ExampleP.rda")
```

ExampleQ

ExampleQ

Description

Example data set and data dictionary with no errors.

Usage

```
data(ExampleQ)
```

Format

R data file that contains two objects:

DD.dict.Q Data dictionary

DS.data.Q Data set

Source

```
DD.path <- system.file("extdata", "3b_SSM_DD_Example5.xlsx", package = "dbGaPCheckup", mustWork=TRUE)
DD.dict.Q <- readxl::read_xlsx(DD.path)
DS.path <- system.file("extdata", "DS_Example5.txt", package = "dbGaPCheckup", mustWork=TRUE) ### FIX T
DS.data.Q <- read.table(DS.path, header=TRUE, sep="\t", quote="", as.is = TRUE)
save(DD.dict.Q, DS.data.Q, file = "ExampleQ.rda")
```

| | |
|-------------|--------------------|
| field_check | <i>Field Check</i> |
|-------------|--------------------|

Description

This function checks for dbGaP required fields variable name (VARNAME), variable description (VARDESC), units (UNITS), and variable value and meaning (VALUES).

Usage

```
field_check(DD.dict, verbose = TRUE)
```

Arguments

| | |
|---------|---|
| DD.dict | Data dictionary. |
| verbose | When TRUE, the function prints the Message out, as well as a list of the fields not found in the data dictionary. |

Value

Tibble, returned invisibly, containing: (1) Time (Time stamp); (2) Name (Name of the function); (3) Status (Passed/Failed); (4) Message (A copy of the message the function printed out); (5) Information (Named vector of TRUE/FALSE values alerting user if checks passed (TRUE) or failed (FALSE) for VARNAME, VARDESC, UNITS, and VALUE).

Examples

```
data(ExampleA)
field_check(DD.dict.A)
print(field_check(DD.dict.A, verbose=FALSE))
```

| | |
|----------|-----------------|
| id_check | <i>ID Check</i> |
|----------|-----------------|

Description

This function checks that the first column of the data set is the primary ID for each participant labeled as SUBJECT_ID, that values contain no illegal characters or padded zeros, and that each participant has an ID.

Usage

```
id_check(DS.data, verbose = TRUE)
```

Arguments

| | |
|---------|--|
| DS.data | Data set. |
| verbose | When TRUE, the function prints the Message out, as well as more detailed diagnostic information. |

Details

Subject IDs should be an integer or string value. Integers should not have zero padding. IDs should not have spaces. Specifically, only the following characters can be included in the ID: English letters, Arabic numerals, period (.), hyphen (-), underscore (_), at symbol (@), and the pound sign (#). All IDs should be filled in (i.e., no missing IDs are allowed).

Value

Tibble, returned invisibly, containing: (1) Time (Time stamp); (2) Name (Name of the function); (3) Status (Passed/Failed); (4) Message (A copy of the message the function printed out); (5) Information (Detailed information about the four ID checks that were performed).

Examples

```
# Example 1: Fail check, 'SUBJECT_ID' not present
data(Example0)
id_check(DS.data.0)
print(id_check(DS.data.0, verbose=FALSE))

# Example 2: Fail check, 'SUBJECT_ID' includes illegal spaces
data(ExampleP)
id_check(DS.data.P)
results <- id_check(DS.data.P)
results$Information[[1]]$details
print(id_check(DS.data.P, verbose=FALSE))

# Example 3: Pass check
data(ExampleA)
id_check(DS.data.A)
print(id_check(DS.data.A, verbose=FALSE))
```

id_first_data

Relocate SUBJECT_ID to First Column of Data Set

Description

This utility function reorders the data set so that SUBJECT_ID comes first.

Usage

```
id_first_data(DS.data)
```

Arguments

DS.data Data set.

Details

SUBJECT_ID is required to be the first column of the data set and first variable listed in the data dictionary.

Value

Updated data set with SUBJECT_ID as first column.

Examples

```
data(ExampleQ)
head(DS.data.Q)
DS.data.updated <- id_first_data(DS.data.Q)
head(DS.data.updated)
```

id_first_dict

Relocate SUBJECT_ID to First Column of Data Dictionary

Description

This utility function reorders the data dictionary so that SUBJECT_ID comes first.

Usage

```
id_first_dict(DD.dict)
```

Arguments

DD.dict Data dictionary.

Details

SUBJECT_ID is required to be the first column of the data set and first variable listed in the data dictionary.

Value

Updated data dictionary with SUBJECT_ID as first variable.

Examples

```
data(ExampleQ)
head(DD.dict.Q)
DD.dict.updated <- id_first_dict(DD.dict.Q)
head(DD.dict.updated)
```

| | |
|---------------|----------------------|
| integer_check | <i>Integer Check</i> |
|---------------|----------------------|

Description

This function searches for variables that appear to be incorrectly listed as TYPE integer.

Usage

```
integer_check(DD.dict, DS.data, verbose = TRUE)
```

Arguments

| | |
|---------|---|
| DD.dict | Data dictionary. |
| DS.data | Data set. |
| verbose | When TRUE, the function prints the Message out, as well as a list of variables that may be incorrectly labeled as TYPE integer. |

Value

Tibble, returned invisibly, containing: (1) Time (Time stamp); (2) Name (Name of the function); (3) Status (Passed/Failed); (4) Message (A copy of the message the function printed out); (5) Information (Names of variables that are listed as TYPE integer, but do not appear to be integers).

Examples

```
# Example 1: Fail check
data(ExampleH)
integer_check(DD.dict.H, DS.data.H)
print(integer_check(DD.dict.H, DS.data.H, verbose=FALSE))

# Example 2: Pass check
data(ExampleA)
integer_check(DD.dict.A, DS.data.A)
print(integer_check(DD.dict.A, DS.data.A, verbose=FALSE))
```

| | |
|-----------|------------------------------------|
| int_check | <i>Integer Check Base Function</i> |
|-----------|------------------------------------|

Description

This function checks for integer values.

Usage

```
int_check(data)
```

Arguments

data Number or vector of numbers.

Value

Logical, TRUE if all non-missing entries in the input vector are integers.

| | |
|------------|-----------------------|
| label_data | <i>Label the data</i> |
|------------|-----------------------|

Description

Using the information in the data dictionary, this function adds non-missing information from the data dictionary as attributes to the data.

Usage

```
label_data(DD.dict, DS.data, non.NA.missing.codes = NA)
```

Arguments

DD.dict Data dictionary.
 DS.data Data set.
 non.NA.missing.codes
 A user-defined vector of numerical missing value codes (e.g., -9999).

Value

A tibble containing the labelled data set, with the data dictionary information embedded as attributes and variables labelled using Haven SPSS conventions.

Examples

```
data(ExampleB)
DS_labelled_data <- label_data(DD.dict.B, DS.data.B, non.NA.missing.codes=c(-9999))
labelled::var_label(DS_labelled_data$SEX)
labelled::val_labels(DS_labelled_data$SEX)
attributes(DS_labelled_data$SEX)
labelled::na_values(DS_labelled_data$HX_DEPRESSION)
```

| | |
|--------------|---|
| minmax_check | <i>Mimumum and Maximum Values Check</i> |
|--------------|---|

Description

This function flags variables that have values exceeding the MIN or MAX listed in the data dictionary.

Usage

```
minmax_check(DD.dict, DS.data, verbose = TRUE, non.NA.missing.codes = NA)
```

Arguments

| | |
|----------------------|--|
| DD.dict | Data dictionary. |
| DS.data | Data set. |
| verbose | When TRUE, the function prints the Message out, as well as a list of variables that violate the listed MIN or MAX. |
| non.NA.missing.codes | A user-defined vector of numerical missing value codes (e.g., -9999). |

Value

Tibble, returned invisibly, containing: (1) Time (Time stamp); (2) Name (Name of the function); (3) Status (Passed/Failed); (4) Message (A copy of the message the function printed out); (5) Information (A list of variables that exceed the listed MIN or MAX values).

Examples

```
# Example 1
# Fail check (incorrectly flagging NA value codes -9999
# and -4444 as outside of the min max range)
data(ExampleA)
minmax_check(DD.dict.A, DS.data.A)
# View out of range values:
details <- minmax_check(DD.dict.A, DS.data.A)$Information
details[[1]]$OutOfRangeValues
# Attempt 2, specifying -9999 and -4444 as missing value
# codes so check works correctly
minmax_check(DD.dict.A, DS.data.A, non.NA.missing.codes=c(-9999, -4444))

# Example 2
data(ExampleI)
minmax_check(DD.dict.I, DS.data.I, non.NA.missing.codes=c(-9999, -4444))
# View out of range values:
details <- minmax_check(DD.dict.I, DS.data.I, non.NA.missing.codes=c(-9999, -4444))$Information
details[[1]]$OutOfRangeValues
```

| | |
|-------------------|-----------------------------------|
| misc_format_check | <i>Miscellaneous Format Check</i> |
|-------------------|-----------------------------------|

Description

This function checks miscellaneous dbGaP formatting requirements to ensure (1) no empty variable names; (2) no duplicate variable names; (3) variable names do not contain "dbgap"; (4) there are no duplicate column names in the dictionary; and (5) column names falling after VALUES column are unnamed.

Usage

```
misc_format_check(DD.dict, DS.data, verbose = TRUE)
```

Arguments

| | |
|---------|--|
| DD.dict | Data dictionary. |
| DS.data | Data set. |
| verbose | When TRUE, the function prints the Message out, as well as more detailed information about which formatting checks failed. |

Value

Tibble, returned invisibly, containing: (1) Time (time stamp); (2) Name (name of the function); (3) Status (Passed/Failed); (4) Message (A copy of the message the function printed out); (5) Information (Names of variables that fail one of these checks).

Examples

```
# Example 1: Fail check
data(ExampleJ)
misc_format_check(DD.dict.J, DS.data.J)
print(misc_format_check(DD.dict.J, DS.data.J, verbose=FALSE))

# Example 2: Pass check
data(ExampleA)
misc_format_check(DD.dict.A, DS.data.A)
print(misc_format_check(DD.dict.A, DS.data.A, verbose=FALSE))
```

missingness_summary *Missingness Summary*

Description

This awareness function summarizes the amount of missingness in the data set.

Usage

```
missingness_summary(DS.data, non.NA.missing.codes = NA, threshold = 95)
```

Arguments

| | |
|----------------------|---|
| DS.data | Data set. |
| non.NA.missing.codes | A user-defined vector of numerical missing value codes (e.g., -9999). |
| threshold | Threshold for missingness of concern (as a percent). |

Value

Tibble containing: (1) Message containing information on the number of variables with a % missingness greater than the threshold; (2) Missingness by variable summary; and (3) Summary of missingness for variables with a missingness level greater than the threshold.

See Also

[create_awareness_report](#)

Examples

```
# Correct useage
data(ExampleA)
missingness_summary(DS.data.A, non.NA.missing.codes=c(-4444, -9999))
```

missing_value_check *Missing Value Check*

Description

This function flags variables that have non-encoded missing value codes.

Usage

```
missing_value_check(
  DD.dict,
  DS.data,
  verbose = TRUE,
  non.NA.missing.codes = NA
)
```

Arguments

| | |
|----------------------|--|
| DD.dict | Data dictionary. |
| DS.data | Data set. |
| verbose | When TRUE, the function prints the Message out, as well as a list of variables that have non-encoded missing values. |
| non.NA.missing.codes | A user-defined vector of numerical missing value codes (e.g., -9999). |

Value

Tibble, returned invisibly, containing: (1) Time (Time stamp); (2) Name (Name of the function); (3) Status (Passed/Failed); (4) Message (A copy of the message the function printed out); (5) Information (A list of variables where a missing value code is not properly encoded).

Examples

```
data(ExampleB)
missing_value_check(DD.dict.B, DS.data.B, non.NA.missing.codes = c(-9999,-4444))
```

| | |
|-------------|------------------------------------|
| mm_precheck | <i>Min Max Required Pre-checks</i> |
|-------------|------------------------------------|

Description

This function runs a workflow of the minimum number of checks required for a user to run min-max_check; the checks include pkg_field_check, dimension_check, and name_check.

Usage

```
mm_precheck(dict, data)
```

Arguments

| | |
|------|------------------|
| dict | Data dictionary. |
| data | Data set. |

Value

Tibble containing the following information for each check: (1) Time (time stamp); (2) Name (name of the function); (3) Status (Passed/Failed); (4) Message (A copy of the message the function printed out); (5) Information (More detailed information about the potential errors identified).

Examples

```
data(ExampleB)
mm_precheck(DD.dict.B, DS.data.B)
```

| | |
|-------------|---|
| mv_precheck | <i>Missing Values Required Pre-checks</i> |
|-------------|---|

Description

This function runs a workflow of the minimum number of checks required for a user to run `missing_value_check`; the checks include `field_check` and `pkg_field_check`.

Usage

```
mv_precheck(dict, data)
```

Arguments

| | |
|------|------------------|
| dict | Data dictionary. |
| data | Data set. |

Value

Tibble containing the following information for each check: (1) Time (time stamp); (2) Name (name of the function); (3) Status (Passed/Failed); (4) Message (A copy of the message the function printed out); (5) Information (More detailed information about the potential errors identified).

Examples

```
data(ExampleB)
mv_precheck(DD.dict.B, DS.data.B)
```

| | |
|------------|-------------------|
| name_check | <i>Name Check</i> |
|------------|-------------------|

Description

This function checks if the variable names match between the data dictionary and the data.

Usage

```
name_check(DD.dict, DS.data, verbose = TRUE)
```

Arguments

| | |
|---------|---|
| DD.dict | Data dictionary. |
| DS.data | Data set. |
| verbose | When TRUE, the function prints the Message out, as well as a list of the non-matching variable names. |

Value

Tibble, returned invisibly, containing: (1) Time (time stamp); (2) Name (name of the function); (3) Status (Passed/Failed); (4) Message (A copy of the message the function printed out); (5) Information (Names of variables that mismatch between the data and data dictionary).

Examples

```
# Example 1: Fail check (name mismatch)
data(ExampleM)
name_check(DD.dict.M, DS.data.M)
DS.data_updated <- name_correct(DD.dict.M, DS.data.M)
name_check(DD.dict.M, DS.data_updated)

# Example 2: Pass check
data(ExampleA)
name_check(DD.dict.A, DS.data.A)
print(name_check(DD.dict.A, DS.data.A, verbose=FALSE))
```

| | |
|--------------|---|
| name_correct | <i>Name Correction Utility Function</i> |
|--------------|---|

Description

This utility function updates the data set so variable names match those listed in the data dictionary.

Usage

```
name_correct(DD.dict, DS.data)
```

Arguments

DD.dict Data dictionary.
DS.data Data set.

Details

Recommend use with caution; perform name_check first.

Value

Updated data set with variables renamed to match the data dictionary.

Examples

```
data(ExampleM)
name_check(DD.dict.M, DS.data.M)
DS.data_updated <- name_correct(DD.dict.M, DS.data.M)
name_check(DD.dict.M, DS.data_updated)
```

| | |
|---------------|------------------------|
| name_precheck | <i>Name Pre-checks</i> |
|---------------|------------------------|

Description

This function runs a workflow of the minimum number of checks required for a user to run min-max_check; the checks include pkg_field_check, dimension_check, and name_check.

Usage

```
name_precheck(dict, data)
```

Arguments

dict Data dictionary.
data Data set.

Value

Tibble containing the following information for each check: (1) Time (time stamp); (2) Name (name of the function); (3) Status (Passed/Failed); (4) Message (A copy of the message the function printed out); (5) Information (More detailed information about the potential errors identified).

Examples

```
data(ExampleB)
name_precheck(DD.dict.B, DS.data.B)
```

| | |
|----------|---------------------------------|
| NA_check | <i>Missing Value (NA) Check</i> |
|----------|---------------------------------|

Description

Checks for NA values in the data set; if NA values are present, also performs check for NA value=meaning.

Usage

```
NA_check(DD.dict, DS.data, verbose = TRUE)
```

Arguments

| | |
|---------|--|
| DD.dict | Data dictionary. |
| DS.data | Data set. |
| verbose | When TRUE, the function prints the Message out, as well as the number of NA values observed in the data set. |

Value

Tibble, returned invisibly, containing: (1) Time (Time stamp); (2) Name (Name of the function); (3) Status (Passed/Failed); (4) Message (A copy of the message the function printed out); (5) Information (the number of NA values in the data set and information on if NA is a properly encoded value).

Examples

```
# Example 1: Fail check
data(ExampleK)
NA_check(DD.dict.K, DS.data.K)
print(NA_check(DD.dict.K, DS.data.K, verbose=FALSE))

# Example 2: Pass check
data(ExampleA)
NA_check(DD.dict.A, DS.data.A)
print(NA_check(DD.dict.A, DS.data.A, verbose=FALSE))
```

| | |
|-------------|------------------------------------|
| NA_precheck | <i>Min Max Required Pre-checks</i> |
|-------------|------------------------------------|

Description

This function runs a workflow of the minimum number of checks required for a user to run `min-max_check`; the checks include `pkg_field_check`, `dimension_check`, and `name_check`.

Usage

```
NA_precheck(dict, data)
```

Arguments

| | |
|-------------------|------------------|
| <code>dict</code> | Data dictionary. |
| <code>data</code> | Data set. |

Value

Tibble containing the following information for each check: (1) Time (time stamp); (2) Name (name of the function); (3) Status (Passed/Failed); (4) Message (A copy of the message the function printed out); (5) Information (More detailed information about the potential errors identified).

Examples

```
data(ExampleB)
NA_precheck(DD.dict.B, DS.data.B)
```

| | |
|------------------------------|-------------------------------------|
| <code>pkg_field_check</code> | <i>Package Required Field Check</i> |
|------------------------------|-------------------------------------|

Description

This function checks for additional fields required by this package including variable type (TYPE), minimum value (MIN), and maximum value (MAX).

Usage

```
pkg_field_check(DD.dict, DS.data, verbose = TRUE)
```

Arguments

| | |
|----------------------|---|
| <code>DD.dict</code> | Data dictionary. |
| <code>DS.data</code> | Data set. |
| <code>verbose</code> | When TRUE, the function prints the Message out, as well as a list of the fields not found in the data dictionary. |

Details

Even though MIN, MAX, and TYPE are not required by dbGaP, our package was created to use these variables in a series of other checks and awareness functions (e.g., render_report, values_check, etc.). If this function fails, the add_missing_fields function can be used.

Value

Tibble, returned invisibly, containing: (1) Time (Time stamp); (2) Name (Name of the function); (3) Status (Passed/Failed); (4) Message (A copy of the message the function printed out); (5) Information (Named vector of TRUE/FALSE values alerting user if checks passed (TRUE) or failed (FALSE) for TYPE, MIN, and MAX).

See Also

[add_missing_fields](#)

Examples

```
# Example 1: Fail check
data(ExampleD)
pkg_field_check(DD.dict.D, DS.data.D)
# Use the add_missing_fields function to add in data
DD.dict.updated <- add_missing_fields(DD.dict.D, DS.data.D)
# Be sure to call in the new version of the dictionary (DD.dict.updated)
pkg_field_check(DD.dict.updated, DS.data.D)

# Example 2: Pass check
data(ExampleA)
pkg_field_check(DD.dict.A, DS.data.A)
print(pkg_field_check(DD.dict.A, DS.data.A, verbose=FALSE))
```

reorder_data

Reorder Data Set Utility Function

Description

This utility function reorders the data set to match the data dictionary.

Usage

```
reorder_data(DD.dict, DS.data)
```

Arguments

| | |
|---------|------------------|
| DD.dict | Data dictionary. |
| DS.data | Data set. |

Value

Updated data set with variables reordered to match the data dictionary.

Examples

```
data(ExampleN)
name_check(DD.dict.N, DS.data.N)
DS.data_updated <- reorder_data(DD.dict.N, DS.data.N)
name_check(DD.dict.N, DS.data_updated)
```

reorder_dictionary *Reorder Data Dictionary Utility Function*

Description

This utility function reorders the data dictionary to match the data set.

Usage

```
reorder_dictionary(DD.dict, DS.data)
```

Arguments

| | |
|---------|------------------|
| DD.dict | Data dictionary. |
| DS.data | Data set. |

Value

Updated data dictionary with variables reordered to match the data set.

Examples

```
data(ExampleN)
name_check(DD.dict.N, DS.data.N)
DD.dict_updated <- reorder_dictionary(DD.dict.N, DS.data.N)
name_check(DD.dict_updated, DS.data.N)
```

| | |
|-----------|------------------|
| row_check | <i>Row Check</i> |
|-----------|------------------|

Description

This function checks for empty or duplicate rows in the data set and data dictionary.

Usage

```
row_check(DD.dict, DS.data, verbose = TRUE)
```

Arguments

| | |
|---------|---|
| DD.dict | Data dictionary. |
| DS.data | Data set. |
| verbose | When TRUE, the function prints the Message out, as well as the row numbers of any problematic rows. |

Value

Tibble, returned invisibly, containing: (1) Time (Time stamp); (2) Name (Name of the function); (3) Status (Passed/Failed); (4) Message (A copy of the message the function printed out); (5) Information (A list of problematic row and participant ID numbers).

Examples

```
# Example 1: Fail check
data(ExampleK)
row_check(DD.dict.K, DS.data.K)
print(row_check(DD.dict.K, DS.data.K, verbose=FALSE))

# Example 2: Pass check
data(ExampleC)
row_check(DD.dict.C, DS.data.C)
print(row_check(DD.dict.C, DS.data.C, verbose=FALSE))
```

| | |
|-------------------|------------------------------|
| short_field_check | <i>Truncated Field Check</i> |
|-------------------|------------------------------|

Description

This function checks for dbGaP required fields variable name (VARNAME), and variable description (VARDESC) as a pre-check embedded in name_check.

Usage

```
short_field_check(DD.dict, verbose = TRUE)
```

Arguments

| | |
|---------|---|
| DD.dict | Data dictionary. |
| verbose | When TRUE, the function prints the Message out, as well as a list of the fields not found in the data dictionary. |

Value

Tibble, returned invisibly, containing: (1) Time (Time stamp); (2) Name (Name of the function); (3) Status (Passed/Failed); (4) Message (A copy of the message the function printed out); (5) Information (Named vector of TRUE/FALSE values alerting user if checks passed (TRUE) or failed (FALSE) for VARNAME, VARDESC, UNITS, and VALUE).

Examples

```
data(ExampleA)
short_field_check(DD.dict.A)
```

| | |
|----------------|----------------------------|
| short_precheck | <i>Truncated Pre-check</i> |
|----------------|----------------------------|

Description

This function runs a workflow of the minimum number of checks required for a user to run dbGaPCheckup_required_field_check; the checks include dbGaP_required_field_check, dimension_check, and name_check.

Usage

```
short_precheck(dict, data)
```

Arguments

| | |
|------|------------------|
| dict | Data dictionary. |
| data | Data set. |

Value

Tibble containing the following information for each check: (1) Time (time stamp); (2) Name (name of the function); (3) Status (Passed/Failed); (4) Message (A copy of the message the function printed out); (5) Information (More detailed information about the potential errors identified).

Examples

```
data(ExampleB)
short_precheck(DD.dict.B, DS.data.B)
```

super_short_precheck *Very Truncated Pre-check*

Description

This function runs a workflow of the minimum number of checks required for a user to run db-GaPCheckup_required_field_check; the checks include dbGaP_required_field_check, dimension_check, and name_check.

Usage

```
super_short_precheck(dict, data)
```

Arguments

| | |
|------|------------------|
| dict | Data dictionary. |
| data | Data set. |

Value

Tibble containing the following information for each check: (1) Time (time stamp); (2) Name (name of the function); (3) Status (Passed/Failed); (4) Message (A copy of the message the function printed out); (5) Information (More detailed information about the potential errors identified).

Examples

```
# Example 1: Pass check
data(ExampleB)
super_short_precheck(DD.dict.B, DS.data.B)
```

type_check *Type Check*

Description

If a TYPE field exists, this function checks for any TYPE entries that aren't allowable per dbGaP instructions.

Usage

```
type_check(DD.dict, verbose = TRUE)
```

Arguments

| | |
|---------|--|
| DD.dict | Data dictionary. |
| verbose | When TRUE, the function prints the Message out, as well as more detailed diagnostic information. |

Details

Allowable entries in TYPE column include: integer; decimal; encoded value; or string. For mixed values, list all types present using commas to separate (e.g., integer, encoded value).

Value

Tibble, returned invisibly, containing: (1) Time (Time stamp); (2) Name (Name of the function); (3) Status (Passed/Failed); (4) Message (A copy of the message the function printed out); (5) Information (List of illegal TYPE entries).

Examples

```
data(ExampleB)
type_check(DD.dict.B)
print(type_check(DD.dict.B, verbose=FALSE))
```

values_check

Values Check

Description

This function checks for potential errors in the VALUES columns by ensuring (1) required format of VALUE=MEANING (e.g., 0=Yes or 1=No); (2) no leading/trailing spaces near the equals sign; (3) all variables of TYPE encoded have VALUES entries; and (4) all variables with VALUES entries are listed as TYPE encoded.

Usage

```
values_check(DD.dict, verbose = TRUE)
```

Arguments

| | |
|---------|--|
| DD.dict | Data dictionary. |
| verbose | When TRUE, the function prints the Message out, as well as a list of variables that fail one of the values checks. |

Value

Tibble, returned invisibly, containing: (1) Time (Time stamp); (2) Name (Name of the function); (3) Status (Passed/Failed); (4) Message (A copy of the message the function printed out); (5) Information (Details of which checks passed/failed for which value=meaning instances).

Examples

```
# Example 1: Fail check
data(ExampleE)
values_check(DD.dict.E)
print(values_check(DD.dict.E, verbose=FALSE))

# Example 2: Pass check
data(ExampleA)
values_check(DD.dict.A)
print(values_check(DD.dict.A, verbose=FALSE))
```

values_precheck

Values Pre-Check

Description

This function runs a workflow of the minimum number of checks required for a user to run `values_check`; the checks include `field_check`, and `type_check`.

Usage

```
values_precheck(dict)
```

Arguments

`dict` Data dictionary.

Value

Tibble containing the following information for each check: (1) Time (time stamp); (2) Name (name of the function); (3) Status (Passed/Failed); (4) Message (A copy of the message the function printed out); (5) Information (More detailed information about the potential errors identified).

Examples

```
data(ExampleB)
values_precheck(DD.dict.B)
```

value_meaning_table *Value-Meaning Table*

Description

This function generates a value-meaning table by parsing the VALUES fields.

Usage

```
value_meaning_table(DD.dict)
```

Arguments

DD.dict Data dictionary.

Value

A data frame with the columns VARNAME, TYPE, VALUE, MEANING.

Examples

```
data(ExampleB)
head(value_meaning_table(DD.dict.B))
```

value_missing_table *Values Missing Table Awareness Function*

Description

This function checks for consistent usage of encoded values and missing value codes between the data dictionary and the data itself.

Usage

```
value_missing_table(DD.dict, DS.data, non.NA.missing.codes = NA)
```

Arguments

DD.dict Data dictionary.
DS.data Data set.
non.NA.missing.codes
 A user-defined vector of numerical missing value codes (e.g., -9999).

Details

For each variable, we have three sets of possible values: the set D of all the unique values observed in the data, the set V of all the values explicitly encoded in the VALUES columns of the data dictionary, and the set M of the missing value codes defined by the user via the `non.NA.missing.codes` argument. This function examines various intersections of these three sets, providing awareness checks to the user about possible issues of concern. While ideally all defined values in set V should be observed in the data (e.g., in set D), it is not necessarily an error if one does not. This function checks for:

- (A) In Set M and Not in Set D: If the user defines a missing value code that is not present in the data.
- (B) In Set V and Not in Set D: If a VALUES entry defines an encoded code value, but that code value is not present in the data.
- (C) In Set M and Not in Set V: If the user defines a missing value code that is not defined in a VALUES entry.
- (D) M in Set D and Not in Set V: If a defined global missing value code is present in the data for a given variable, but that variable does not have a corresponding VALUES entry.
- (E) (Set V values that are not in Set M) that are NOT in Set D = (Set V not in M) not in D: If a VALUES entry is not defined as a missing value code AND is not detected in the data.

Value

A list, returned invisibly, with two components:

- "report" Tibble containing: (1) Name (Name of the function) and (2) Information (Details of all potential flagged variables).
- "tb" Tibble with detailed information used to construct the Information.

See Also

[create_awareness_report](#)

Examples

```
data(ExampleB)
value_missing_table(DD.dict.B, DS.data.B, non.NA.missing.codes = c(-9999))
print(value_missing_table(DD.dict.B, DS.data.B, non.NA.missing.codes = c(-9999)))
results <- value_missing_table(DD.dict.B, DS.data.B, non.NA.missing.codes = c(-9999))
results$report$Information$details
```

Index

[add_missing_fields](#), [4](#), [50](#)

[check_report](#), [5](#), [6](#)
[complete_check](#), [5](#), [6](#)
[create_awareness_report](#), [7](#), [43](#), [58](#)
[create_report](#), [8](#)

[dat_function](#), [9](#)
[dat_function_selected](#), [10](#)
[DD.dict.A](#), [11](#)
[DD.dict.B](#), [11](#)
[DD.dict.C](#), [11](#)
[DD.dict.D](#), [12](#)
[DD.dict.E](#), [12](#)
[DD.dict.F](#), [12](#)
[DD.dict.G](#), [13](#)
[DD.dict.H](#), [13](#)
[DD.dict.I](#), [13](#)
[DD.dict.J](#), [14](#)
[DD.dict.K](#), [14](#)
[DD.dict.L](#), [14](#)
[DD.dict.M](#), [15](#)
[DD.dict.N](#), [15](#)
[DD.dict.Q](#), [15](#)
[decimal_check](#), [16](#)
[description_check](#), [17](#)
[dictionary_search](#), [17](#)
[dimension_check](#), [18](#)
[DS.data.A](#), [19](#)
[DS.data.B](#), [19](#)
[DS.data.C](#), [20](#)
[DS.data.D](#), [20](#)
[DS.data.E](#), [20](#)
[DS.data.F](#), [21](#)
[DS.data.G](#), [21](#)
[DS.data.H](#), [21](#)
[DS.data.I](#), [22](#)
[DS.data.J](#), [22](#)
[DS.data.K](#), [22](#)
[DS.data.L](#), [23](#)
[DS.data.M](#), [23](#)
[DS.data.N](#), [23](#)
[DS.data.O](#), [24](#)
[DS.data.P](#), [24](#)
[DS.data.Q](#), [24](#)
[dup_values](#), [25](#)

[eval_function](#), [25](#)
[ExampleA](#), [11](#), [19](#), [26](#)
[ExampleB](#), [11](#), [19](#), [26](#)
[ExampleC](#), [11](#), [20](#), [27](#)
[ExampleD](#), [12](#), [20](#), [27](#)
[ExampleE](#), [12](#), [20](#), [28](#)
[ExampleF](#), [12](#), [21](#), [29](#)
[ExampleG](#), [13](#), [21](#), [29](#)
[ExampleH](#), [13](#), [21](#), [30](#)
[ExampleI](#), [13](#), [22](#), [30](#)
[ExampleJ](#), [14](#), [22](#), [31](#)
[ExampleK](#), [14](#), [22](#), [32](#)
[ExampleL](#), [14](#), [23](#), [32](#)
[ExampleM](#), [15](#), [23](#), [33](#)
[ExampleN](#), [15](#), [23](#), [33](#)
[ExampleO](#), [24](#), [34](#)
[ExampleP](#), [24](#), [35](#)
[ExampleQ](#), [15](#), [24](#), [35](#)

[field_check](#), [36](#)

[id_check](#), [36](#)
[id_first_data](#), [37](#)
[id_first_dict](#), [38](#)
[int_check](#), [39](#)
[integer_check](#), [39](#)

[label_data](#), [40](#)

[minmax_check](#), [41](#)
[misc_format_check](#), [42](#)
[missing_value_check](#), [43](#)
[missingness_summary](#), [8](#), [43](#)
[mm_precheck](#), [44](#)

mv_precheck, [45](#)

NA_check, [48](#)

NA_precheck, [49](#)

name_check, [46](#)

name_correct, [46](#)

name_precheck, [47](#)

pkg_field_check, [49](#)

reorder_data, [50](#)

reorder_dictionary, [51](#)

row_check, [52](#)

short_field_check, [52](#)

short_precheck, [53](#)

super_short_precheck, [54](#)

type_check, [54](#)

value_meaning_table, [57](#)

value_missing_table, [8](#), [57](#)

values_check, [55](#)

values_precheck, [56](#)