

# Package: HighFrequencyChecks (via r-universe)

October 10, 2026

**Type** Package

**Title** High Frequency Checks

**Version** 0.5.0

**Maintainer** Edouard Legoupil <legoupil@unhcr.org>

**Description** During the data collection, a series of automatic check, aka: High Frequency checks, are required. The functions shared here are useful during the data collection process to check periodically for possible errors, and will provide meaningful inputs to the enumerators. All these functions do not have to be ran at the same period of time. They are provided there to help data supervisor to build reports. This work is an adaptation of a Stata Package from [Innovations for Poverty Action](<https://github.com/PovertyAction/high-frequency-checks>).

**License** MIT + file LICENSE

**URL** <https://edouard-legoupil.github.io/HighFrequencyChecks>

**BugReports** <https://github.com/Edouard-Legoupil/HighFrequencyChecks/issues>

**Depends** R (>= 3.5.0)

**Imports** config, DescTools, dplyr, ggplot2, golem, kobocruncher, magrittr, outliers, readr, reshape2, riddle, shiny, shinydashboard, sjlabelled, sp, stats, stringi, tidyverse, unhcrshiny, unhcrthemes, utils

**Suggests** data.table, DT, gsubfn, knitr, rmarkdown, spelling, testthat

**Remotes** unhcr-dataviz/unhcrdown, unhcr-dataviz/unhcrthemes, edouard-legoupil/unhcrshiny, edouard-legoupil/kobocruncher, edouard-legoupil/riddle

**VignetteBuilder** knitr

**Config/fusen/version** 0.5.2

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.2.3.9000

**Language** en-US

**Config/pak/sysreqs** libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev make libharfbuzz-dev libmagick++-dev gsfonts libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libwebp-dev libxml2-dev libssl-dev libx11-dev zlib1g-dev

**Repository** <https://unhcrverse.r-universe.dev>

**Date/Publication** 2023-11-02 19:37:01 UTC

**RemoteUrl** <https://github.com/Edouard-Legoupil/HighFrequencyChecks>

**RemoteRef** HEAD

**RemoteSha** 92c85ccda651dd111b0334c3304687e313ad1d58

## Contents

|                                                  |    |
|--------------------------------------------------|----|
| assessmentDailyValidSurveys . . . . .            | 3  |
| assessmentDuration . . . . .                     | 4  |
| assessmentDurationOutliers . . . . .             | 5  |
| assessmentInterviewTime . . . . .                | 6  |
| assessmentProductivity . . . . .                 | 7  |
| assessmentTrackingSheet . . . . .                | 8  |
| enumeratorErrorsSummary . . . . .                | 10 |
| enumeratorIsLazy . . . . .                       | 11 |
| enumeratorProductivity . . . . .                 | 12 |
| enumeratorProductivityOutliers . . . . .         | 13 |
| enumeratorSurveysConsent . . . . .               | 14 |
| enumeratorSurveysDuration . . . . .              | 15 |
| isInterviewAtTheSamplePoint . . . . .            | 16 |
| isInterviewCompleted . . . . .                   | 18 |
| isInterviewInTheCorrectSite . . . . .            | 19 |
| isInterviewTooShort . . . . .                    | 21 |
| isInterviewTooShortForTheHouseholdSize . . . . . | 23 |
| isInterviewWithConsent . . . . .                 | 24 |
| isSurveyEndBeforeItStarts . . . . .              | 25 |
| isSurveyMadeInTheFuture . . . . .                | 27 |
| isSurveyOnMoreThanADay . . . . .                 | 28 |
| isSurveyStartedBeforeTheAssessment . . . . .     | 30 |
| isuniquerespondantIDDuplicated . . . . .         | 31 |
| isuniquerespondantIDMissing . . . . .            | 33 |
| run_app . . . . .                                | 34 |
| surveyBigValues . . . . .                        | 35 |
| surveyDistinctValues . . . . .                   | 36 |
| surveyMissingValues . . . . .                    | 37 |
| surveyOtherValues . . . . .                      | 38 |
| surveyOutliers . . . . .                         | 39 |
| surveySmallValues . . . . .                      | 41 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>43</b> |
|--------------|-----------|

---

assessmentDailyValidSurveys

*Daily number of filled forms per consent status*


---

## Description

This function display the number of filled forms conducted per day per consent status.

## Usage

```
assessmentDailyValidSurveys(
  ds = NULL,
  surveyDate = NULL,
  dateFormat = NULL,
  surveyConsent = NULL,
  attempt = NULL
)
```

## Arguments

|                       |                                                                                            |
|-----------------------|--------------------------------------------------------------------------------------------|
| ds                    | dataset containing the survey (from kobo): labelled data.frame                             |
| surveyDate            | name of the field in the dataset where the date of the survey is stored: string            |
| dateFormat            | format used for the date: string ('%m/%d/%Y')                                              |
| surveyConsent         | name of the field in the dataset where the survey consent is stored: string                |
| attempt               | name of the field in the dataset where the interview attempt output is stored: string      |
| checkperiod           | if not null number of day before today when the check should be made                       |
| consentForValidSurvey | value defined in the kobo form to acknowledge the surveyed person gave his consent: string |

## Value

result a list that includes: \* dst same dataset as the inputed one but with survey marked for deletion if errors are found and delete = TRUE (or NULL) \* ret\_log list of the errors found (or NULL) \* var a list of value (or NULL) \* graph graphical representation of the results (or NULL)

## Examples

```
load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
surveyDate <- "survey_date"
dateFormat <- "%m/%d/%Y"
surveyConsent <- "survey_consent"
result <- assessmentDailyValidSurveys(
  ds = ds,
```

```

surveyDate = surveyDate,
dateFormat = dateFormat,
surveyConsent = surveyConsent)

knitr::kable(head(result[["ret_log"]], 10))
print(result[["graph"]])

```

---

|                    |                                                           |
|--------------------|-----------------------------------------------------------|
| assessmentDuration | <i>Compute the average and total time for the surveys</i> |
|--------------------|-----------------------------------------------------------|

---

## Description

This function compute the average and total time for the surveys Warning: If there are uncorrected mistakes in the survey dates, it can lead to have the length of the survey in seconds and this check will not performed well

## Usage

```
assessmentDuration(ds = NULL, dates = NULL, attempt = NULL)
```

## Arguments

|                       |                                                                                                                                            |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| ds                    | dataset containing the survey (from kobo): labelled data.frame                                                                             |
| dates                 | name of the fields where the information about the start and end date of the survey is stored: list of string (c('start_date','end_date')) |
| attempt               | name of the field in the dataset where the interview attempt output is stored: string                                                      |
| checkperiod           | if not null number of day before today when the check should be made                                                                       |
| surveyConsent         | name of the field in the dataset where the survey consent is stored: string                                                                |
| consentForValidSurvey | value defined in the kobo form to acknowledge the surveyed person gave his consent: string                                                 |

## Value

result a list that includes: \* dst same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE (or NULL) \* ret\_log list of the errors found (or NULL) \* var a list of value (or NULL) \* graph graphical representation of the results (or NULL)

## Examples

```

load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
dates <- c("survey_start", "end_survey")

result <- assessmentDuration(ds = ds, dates=dates)
knitr::kable(head(result[["ret_log"]],10))
print(result[["graph"]])

```

---

assessmentDurationOutliers

*Report the outlier durations for the surveys*


---

## Description

This function report the outlier durations for the surveys

## Usage

```
assessmentDurationOutliers(
  ds = NULL,
  dates = NULL,
  sdval = 2,
  attempt = NULL,
  startDataCollection = NULL,
  reportingColumns = c(enumeratorID, uniquerespondantID)
)
```

## Arguments

|                       |                                                                                                                                                                                        |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ds                    | dataset containing the survey (from kobo): labelled data.frame                                                                                                                         |
| dates                 | name of the fields where the information about the start and end date of the survey is stored: list of string (c('start_date','end_date'))                                             |
| sdval                 | (Optional, by default set to 2) number of standard deviation for which the data within is considered as acceptable: integer                                                            |
| attempt               | name of the field in the dataset where the interview attempt output is stored: string                                                                                                  |
| startDataCollection   | Date when the data collections started                                                                                                                                                 |
| reportingColumns      | (Optional, by default it is built from the enumeratorID and the uniquerespondantID) name of the columns from the dataset you want in the result: list of string (c('coll','col2',...)) |
| checkperiod           | if not null number of day before today when the check should be made                                                                                                                   |
| uniquerespondantID    | name of the field where the survey unique ID is stored: string                                                                                                                         |
| enumeratorID          | name of the field where the enumerator ID is stored: string                                                                                                                            |
| surveyConsent         | name of the field in the dataset where the survey consent is stored: string                                                                                                            |
| consentForValidSurvey | value defined in the kobo form to acknowledge the surveyed person gave his consent: string                                                                                             |

**Value**

result a list that includes: \* dst same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE (or NULL) \* ret\_log list of the errors found (or NULL) \* var a list of value (or NULL) \* graph graphical representation of the results (or NULL)

**Examples**

```
load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
dates <- c("survey_start", "end_survey")
uniquerespondantID <- "X_uuid"
enumeratorID <- "enumerator_id"
reportingColumns <- c(enumeratorID, uniquerespondantID)
sdval<-2
result <- assessmentDurationOutliers(ds = ds,
                                     dates=dates,
                                     sdval=sdval,
                                     reportingColumns=reportingColumns)

knitr::kable(head(result[["ret_log"]], 10))
print(result[["graph"]])
```

---

assessmentInterviewTime

*Daily number of filled forms per consent status*

---

**Description**

This function display the number of filled forms conducted per day per consent status.

**Usage**

```
assessmentInterviewTime(
  ds = NULL,
  surveyDate = NULL,
  dateFormat = NULL,
  surveyConsent = NULL
)
```

**Arguments**

|               |                                                                                 |
|---------------|---------------------------------------------------------------------------------|
| ds            | dataset containing the survey (from kobo): labelled data.frame                  |
| surveyDate    | name of the field in the dataset where the date of the survey is stored: string |
| dateFormat    | format used for the date: string ('%m/%d/%Y')                                   |
| surveyConsent | name of the field in the dataset where the survey consent is stored: string     |

|                       |                                                                                            |
|-----------------------|--------------------------------------------------------------------------------------------|
| consentForValidSurvey | value defined in the kobo form to acknowledge the surveyed person gave his consent: string |
| attempt               | name of the field in the dataset where the interview attempt output is stored: string      |

### Value

result a list that includes: \* dst same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE (or NULL) \* ret\_log list of the errors found (or NULL) \* var a list of value (or NULL) \* graph graphical representation of the results (or NULL)

### Examples

```
load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
surveyDate <- "survey_date"
dateFormat <- "%m/%d/%Y"
surveyConsent <- "survey_consent"
result <- assessmentInterviewTime(ds = ds,
                                surveyDate=surveyDate,
                                dateFormat=dateFormat,
                                surveyConsent=surveyConsent)

knitr::kable(head(result[["ret_log"]], 10))
print(result[["graph"]])
```

---

assessmentProductivity

*Summary of daily average productivity*

---

### Description

This function display the number of interview conducted per day.

### Usage

```
assessmentProductivity(
  ds = NULL,
  surveyDate = NULL,
  dateFormat = NULL,
  surveyConsent = NULL
)
```

**Arguments**

**ds** dataset containing the survey (from kobo): labelled data.frame  
**surveyDate** name of the field in the dataset where the date of the survey is stored: string  
**dateFormat** format used for the date: string ('%m/%d/%Y')  
**surveyConsent** name of the field in the dataset where the survey consent is stored: string  
**checkperiod** if not null number of day before today when the check should be made  
**consentForValidSurvey** value defined in the kobo form to acknowledge the surveyed person gave his consent: string

**Value**

result a list that includes: \* dst same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE (or NULL) \* ret\_log list of the errors found (or NULL) \* var a list of value (or NULL) \* graph graphical representation of the results (or NULL)

**Examples**

```

load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
surveyDate <- "survey_date"
dateFormat <- "%m/%d/%Y"
surveyConsent <- "survey_consent"
result <- assessmentProductivity(ds = ds,
                                surveyDate = surveyDate,
                                dateFormat = dateFormat,
                                surveyConsent = surveyConsent)

knitr::kable(head(result[["ret_log"]], 10))
print(result[["graph"]])
  
```

---

 assessmentTrackingSheet

*Overall tracking sheet*


---

**Description**

This function display the overall tracking sheet.

**Usage**

```

assessmentTrackingSheet(
  ds = NULL,
  dsSite = NULL,
  sampleSizeTable = NULL,
  sampleSizeTableSite = NULL,

```

```

    sampleSizeTableTarget = NULL,
    sampleSizeTableAvailable = NULL,
    surveyConsent = NULL,
    consentForValidSurvey = NULL
  )

```

## Arguments

|                                       |                                                                                                |
|---------------------------------------|------------------------------------------------------------------------------------------------|
| <code>ds</code>                       | dataset containing the survey (from kobo): labelled data.frame                                 |
| <code>dsSite</code>                   | name of the field in the dataset where the site is stored: string                              |
| <code>sampleSizeTable</code>          | dataset containing the sampling frame: data.frame                                              |
| <code>sampleSizeTableSite</code>      | name of the field in the sampling frame where the site is stored: string                       |
| <code>sampleSizeTableTarget</code>    | name of the field where the target number of survey is stored in the sampling frame: string    |
| <code>sampleSizeTableAvailable</code> | name of the field where the number of points generated is stored in the sampling frame: string |
| <code>surveyConsent</code>            | name of the field in the dataset where the survey consent is stored: string                    |
| <code>consentForValidSurvey</code>    | value defined in the kobo form to acknowledge the surveyed person gave his consent: string     |
| <code>checkperiod</code>              | if not null number of day before today when the check should be made                           |

## Value

result a list that includes: \* `dst` same dataset as the inputed one but with survey marked for deletion if errors are found and `delete=TRUE` (or `NULL`) \* `ret_log` list of the errors found (or `NULL`) \* `var` a list of value (or `NULL`) \* `graph` graphical representation of the results (or `NULL`)

## Examples

```

load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
dsSite <- "union_name"
load(system.file("SampleSize.RData", package = "HighFrequencyChecks"))
sampleSizeTable <- SampleSize

sampleSizeTableSite <- "Union"
sampleSizeTableTarget <- "SS"
sampleSizeTableAvailable <- "TotPts" # Usually the Target + a buffer
surveyConsent <- "survey_consent"
consentForValidSurvey <- "yes" # consent value for yes

result <- assessmentTrackingSheet(ds = ds,
                                dsSite = dsSite,

```

```

sampleSizeTable = sampleSizeTable,
sampleSizeTableSite = sampleSizeTableSite,
sampleSizeTableTarget = sampleSizeTableTarget,
sampleSizeTableAvailable = sampleSizeTableAvailable,
surveyConsent = surveyConsent,
consentForValidSurvey = consentForValidSurvey)

knitr::kable(head(result[["ret_log"]], 10))
print(result[["graph"]])

```

---

enumeratorErrorsSummary

*Create a dashboard displaying the number of errors by enumerators*

---

## Description

This function display the number of errors made by the enumerator, one graph is generated by enumerator showing for each

## Usage

```
enumeratorErrorsSummary(enumeratorID = NULL, reports = NULL)
```

## Arguments

|                       |                                                                                                                                                                                                                                                                                           |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| enumeratorID          | name of the field where the enumerator ID is stored: string                                                                                                                                                                                                                               |
| reports               | reports names generated from the other checks included in this package, be sure when you choose the columns to be included in each report generated that the enumeratorID is selected before including the report as a parameter to this function: list of string(c(report1,report2,...)) |
| surveyConsent         | name of the field in the dataset where the survey consent is stored: string                                                                                                                                                                                                               |
| consentForValidSurvey | value defined in the kobo form to acknowledge the surveyed person gave his consent: string                                                                                                                                                                                                |

## Value

result a list that includes: \* dst same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE (or NULL) \* ret\_log list of the errors found (or NULL) \* var a list of value (or NULL) \* graph graphical representation of the results (or NULL)

## Examples

```
# enumeratorID <- "enumerator_id"
# reports <- c( "isInterviewCompleted",
#             "isInterviewInTheCorrectSite",
#             "isInterviewTooShort",
#             "isInterviewTooShortForTheHouseholdSize",
#             "isInterviewWithConsent",
#             "isSurveyEndBeforeItStarts",
#             "isSurveyMadeInTheFuture",
#             "isSurveyOnMoreThanADay",
#             "isSurveyStartedBeforeTheAssessment",
#             "isuniquerespondantIDDuplicated",
#             "isuniquerespondantIDMissing")
#
# result <- enumeratorErrorsSummary(enumeratorID=enumeratorID,ds = ds,
#                                   # surveyDate=surveyDate,
#                                   # dateFormat=dateFormat,
#                                   # surveyConsent=surveyConsent
#                                   reports=reports)
#
# print(result[["graph"]])
```

---

|                  |                                                                                    |
|------------------|------------------------------------------------------------------------------------|
| enumeratorIsLazy | <i>Check the enumerators who pick up less than X answers per specific question</i> |
|------------------|------------------------------------------------------------------------------------|

---

## Description

This function display the enumerators who picked up less than a specified amount of answers per specific question. This can be useful for select\_multiple questions where respondent shall give at least 3 options for instance.

## Usage

```
enumeratorIsLazy(
  ds = NULL,
  enumeratorID = NULL,
  questionsEnumeratorIsLazy = NULL
)
```

## Arguments

|                           |                                                                                                                                                                                                                                                      |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ds                        | dataset containing the survey (from kobo): labelled data.frame                                                                                                                                                                                       |
| enumeratorID              | name of the field where the enumerator ID is stored: string                                                                                                                                                                                          |
| questionsEnumeratorIsLazy | columns name from the dataset and value you want to check against (c(col1=value1,col2=value2,...)): named list of integer the column name is the main part of the name generated by kobo (eg: for the question 'main_income', kobo will generate one |

TRUE/FALSE column per possible answer as 'main\_income.work', 'main\_income.remittance' ..., only the main part 'main\_income' has to be specified here)

checkperiod      if not null number of day before today when the check should be made

surveyConsent    name of the field in the dataset where the survey consent is stored: string

consentForValidSurvey  
                    value defined in the kobo form to acknowledge the surveyed person gave his consent: string

### Value

result a list that includes: \* dst same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE (or NULL) \* ret\_log list of the errors found (or NULL) \* var a list of value (or NULL) \* graph graphical representation of the results (or NULL)

### Examples

```
load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
enumeratorID <- "enumerator_id"
questionsEnumeratorIsLazy <- c(consent_received.shelter_nfi.non_food_items=3,
                               consent_received.food_security.main_income=3,
                               consent_received.child_protection.boy_risk=3,
                               consent_received.child_protection.girl_risk=3)

result <- enumeratorIsLazy(ds = ds,
                           enumeratorID=enumeratorID,
                           questionsEnumeratorIsLazy=questionsEnumeratorIsLazy)

knitr::kable(head(result[["ret_log"]], 10))
```

---

enumeratorProductivity

*Check the number of Interview by enumerator*

---

### Description

This function display the total number of survey made and the average per day per enumerator.

### Usage

```
enumeratorProductivity(ds = NULL, surveyDate = NULL, enumeratorID = NULL)
```

**Arguments**

|                       |                                                                                            |
|-----------------------|--------------------------------------------------------------------------------------------|
| ds                    | dataset containing the survey (from kobo): labelled data.frame                             |
| surveyDate            | name of the field in the dataset where the date of the survey is stored: string            |
| enumeratorID          | name of the field where the enumerator ID is stored: string                                |
| checkperiod           | if not null number of day before today when the check should be made                       |
| surveyConsent         | name of the field in the dataset where the survey consent is stored: string                |
| consentForValidSurvey | value defined in the kobo form to acknowledge the surveyed person gave his consent: string |

**Value**

result a list that includes: \* dst same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE (or NULL) \* ret\_log list of the errors found (or NULL) \* var a list of value (or NULL) \* graph graphical representation of the results (or NULL)

**Examples**

```
load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
surveyDate <- "survey_date"
enumeratorID <- "enumerator_id"

result <- enumeratorProductivity(ds = ds,
                                surveyDate=surveyDate,
                                enumeratorID=enumeratorID)

knitr::kable(head(result[["ret_log"]], 10))
print(result[["graph"]])
```

---

```
enumeratorProductivityOutliers
```

*Check the enumerators with very low or high productivity*

---

**Description**

This function display the enumerators with very low or high productivity.

**Usage**

```
enumeratorProductivityOutliers(
  ds = NULL,
  enumeratorID = NULL,
  surveyDate = NULL,
  sdval = 2
)
```

**Arguments**

|                           |                                                                                                                             |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| <code>ds</code>           | dataset containing the survey (from kobo): labelled data.frame                                                              |
| <code>enumeratorID</code> | name of the field where the enumerator ID is stored: string                                                                 |
| <code>surveyDate</code>   | name of the field in the dataset where the date of the survey is stored: string                                             |
| <code>sdval</code>        | (Optional, by default set to 2) number of standard deviation for which the data within is considered as acceptable: integer |
| <code>checkperiod</code>  | if not null number of day before today when the check should be made                                                        |

**Value**

result a list that includes: \* `dst` same dataset as the inputed one but with survey marked for deletion if errors are found and `delete=TRUE` (or `NULL`) \* `ret_log` list of the errors found (or `NULL`) \* `var` a list of value (or `NULL`) \* `graph` graphical representation of the results (or `NULL`)

**Examples**

```
load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
enumeratorID <- "enumerator_id"
surveyDate <- "survey_date"
sdval<-2

result <- enumeratorProductivityOutliers(ds = ds,
                                         enumeratorID=enumeratorID,
                                         surveyDate=surveyDate,
                                         sdval=sdval)

knitr::kable(head(result[["ret_log"]], 10))
print(result[["graph"]])
```

---

enumeratorSurveysConsent

*Percentage of non-completed interviews by enumerator*

---

**Description**

This function display the percentage of non-completed interviews per enumerator.

**Usage**

```
enumeratorSurveysConsent(ds = NULL, surveyConsent = NULL, enumeratorID = NULL)
```

**Arguments**

|                       |                                                                                            |
|-----------------------|--------------------------------------------------------------------------------------------|
| ds                    | dataset containing the survey (from kobo): labelled data.frame                             |
| surveyConsent         | name of the field in the dataset where the survey consent is stored: string                |
| enumeratorID          | name of the field where the enumerator ID is stored: string                                |
| checkperiod           | if not null number of day before today when the check should be made                       |
| consentForValidSurvey | value defined in the kobo form to acknowledge the surveyed person gave his consent: string |

**Value**

result a list that includes: \* dst same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE (or NULL) \* ret\_log list of the errors found (or NULL) \* var a list of value (or NULL) \* graph graphical representation of the results (or NULL)

**Examples**

```
load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
surveyConsent <- "survey_consent"
enumeratorID <- "enumerator_id"

result <- enumeratorSurveysConsent(ds = ds,
                                   surveyConsent=surveyConsent,
                                   enumeratorID=enumeratorID)

knitr::kable(head(result[["ret_log"]], 10))
print(result[["graph"]])
```

---

enumeratorSurveysDuration

*Check the average interview duration by enumerator*

---

**Description**

This function display the average interview duration per enumerator.

**Usage**

```
enumeratorSurveysDuration(ds = NULL, dates = NULL, enumeratorID = NULL)
```

**Arguments**

|                       |                                                                                                                                            |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| ds                    | dataset containing the survey (from kobo): labelled data.frame                                                                             |
| dates                 | name of the fields where the information about the start and end date of the survey is stored: list of string (c('start_date','end_date')) |
| enumeratorID          | name of the field where the enumerator ID is stored: string                                                                                |
| checkperiod           | if not null number of day before today when the check should be made                                                                       |
| surveyConsent         | name of the field in the dataset where the survey consent is stored: string                                                                |
| consentForValidSurvey | value defined in the kobo form to acknowledge the surveyed person gave his consent: string                                                 |

**Value**

result a list that includes: \* dst same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE (or NULL) \* ret\_log list of the errors found (or NULL) \* var a list of value (or NULL) \* graph graphical representation of the results (or NULL)

**Examples**

```
load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
dates <- c("survey_start", "end_survey")
enumeratorID <- "enumerator_id"
result <- enumeratorSurveysDuration(ds = ds,
                                   dates=dates,
                                   enumeratorID=enumeratorID)

knitr::kable(head(result[["ret_log"]], 10))
print(result[["graph"]])
```

---

isInterviewAtTheSamplePoint

*GIS check surveys if fall without Xm radius from a sampled point*

---

**Description**

This function check that all interviews in the dataset were made within a distance from a sampled point. It is based on a GIS shapefile providing the sample points for the assessment. The function is based on the GPS data filled in the survey to determine their location. There is an option to automatically mark for deletion the surveys which are to far away from a sampled point.

One internal function "make\_GeodesicBuffer" used to create the buffers is created by Valentin <https://stackoverflow.com/users/5193830/valentin>

**Usage**

```
isInterviewAtTheSamplePoint(
  ds = NULL,
  dsCoordinates = NULL,
  sampledPoints = NULL,
  buffer = 10,
  surveyConsent = NULL,
  reportingColumns = c(enumeratorID, uniquerespondantID),
  deleteIsInterviewAtTheSamplePoint = FALSE
)
```

**Arguments**

|                                   |                                                                                                                                                                                        |
|-----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ds                                | dataset containing the survey (from kobo): labelled data.frame                                                                                                                         |
| dsCoordinates                     | name of the fields from the dataset where the information about the GPS coordinates are stored: list of string (c('Long','Lat'))                                                       |
| sampledPoints                     | dataset containing the shapefile of the households sampled - Regardless the projection used for the shapefile, it is transformed to WGS84                                              |
| buffer                            | value in meter to determine the buffer from a sampled point which is acceptable: integer                                                                                               |
| surveyConsent                     | name of the field in the dataset where the survey consent is stored: string                                                                                                            |
| reportingColumns                  | (Optional, by default it is built from the enumeratorID and the uniquerespondantID) name of the columns from the dataset you want in the result: list of string (c('col1','col2',...)) |
| deleteIsInterviewAtTheSamplePoint | (Optional, by default set as FALSE) if TRUE, the survey in error will be marked as 'deletedIsInterviewAtTheSamplePoint': boolean (TRUE/FALSE)                                          |
| checkperiod                       | if not null number of day before today when the check should be made                                                                                                                   |
| consentForValidSurvey             | value defined in the kobo form to acknowledge the surveyed person gave his consent: string                                                                                             |
| uniquerespondantID                | name of the field where the survey unique ID is stored: string                                                                                                                         |
| enumeratorID                      | name of the field where the enumerator ID is stored: string                                                                                                                            |

**Value**

result a list that includes: \* dst same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE (or NULL) \* ret\_log list of the errors found (or NULL) \* var a list of value (or NULL) \* graph graphical representation of the results (or NULL)

**Examples**

```
load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
```

```

load(system.file("SamplePts.RData", package = "HighFrequencyChecks"))
sampledPoints <- SamplePts

dsCoordinates <- c("X_gps_reading_longitude", "X_gps_reading_latitude")
buffer <- 10
surveyConsent <- "survey_consent"
uniquerespondantID <- "X_uuid"
enumeratorID <- "enumerator_id"
reportingColumns <- c(enumeratorID, uniquerespondantID)

# result <- isInterviewAtTheSamplePoint(ds = ds,
#                                     dsCoordinates = dsCoordinates,
#                                     sampledPoints=sampledPoints,
#                                     buffer=buffer,
#                                     surveyConsent=surveyConsent,
#                                     reportingColumns=reportingColumns,
#                                     deleteIsInterviewAtTheSamplePoint=FALSE)
# knitr::kable(head(result[["ret_log"]], 10))
# print(result[["graph"]])

```

---

isInterviewCompleted    *Check that all interviews were completed*

---

## Description

This function check that all interviews in the dataset are completed, meaning all the interviews have an end date and time. There is an option to automatically mark for deletion the surveys which have not an end date.

## Usage

```

isInterviewCompleted(
  ds = NULL,
  surveyConsent = NULL,
  dates = NULL,
  reportingColumns = c(enumeratorID, uniquerespondantID),
  deleteIsInterviewCompleted = FALSE
)

```

## Arguments

|                  |                                                                                                                                                                                        |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ds               | dataset containing the survey (from kobo): labelled data.frame                                                                                                                         |
| surveyConsent    | name of the field in the dataset where the survey consent is stored: string                                                                                                            |
| dates            | name of the fields where the information about the start and end date of the survey is stored: list of string (c('start_date','end_date'))                                             |
| reportingColumns | (Optional, by default it is built from the enumeratorID and the uniquerespondantID) name of the columns from the dataset you want in the result: list of string (c('col1','col2',...)) |

```

deleteIsInterviewCompleted
    (Optional, by default set as FALSE) if TRUE, the survey in error will be marked
    as 'deletedIsInterviewCompleted': boolean (TRUE/FALSE)

checkperiod    if not null number of day before today when the check should be made

consentForValidSurvey
    value defined in the kobo form to acknowledge the surveyed person gave his
    consent: string

uniquerespondantID
    name of the field where the survey unique ID is stored: string

enumeratorID   name of the field where the enumerator ID is stored: string

```

### Value

result a list that includes: \* dst same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE (or NULL) \* ret\_log list of the errors found (or NULL) \* var a list of value (or NULL) \* graph graphical representation of the results (or NULL)

### Examples

```

load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
surveyConsent <- "survey_consent"
dates <- c("survey_start", "end_survey")
uniquerespondantID <- "X_uuid"
enumeratorID <- "enumerator_id"
reportingColumns <- c(enumeratorID, uniquerespondantID)

result <- isInterviewCompleted(ds = ds,
                              surveyConsent=surveyConsent,
                              dates=dates,
                              reportingColumns=reportingColumns,
                              deleteIsInterviewCompleted=FALSE)

knitr::kable(head(result[["ret_log"]], 10))
print(result[["graph"]])

```

---

isInterviewInTheCorrectSite

*GIS check surveys for site*

---

### Description

This function check that all interviews in the dataset were made in the correct site. It is based on a GIS shapefile providing the boundaries of each site with their names. The function is based on the GPS data filled in the survey to determine their location. There is an option to automatically correct the site in the surveys whith the correct location.

**Usage**

```
isInterviewInTheCorrectSite(
  ds = NULL,
  dsSite = NULL,
  dsCoordinates = NULL,
  adminBoundaries = NULL,
  adminBoundariesSite = NULL,
  surveyConsent = NULL,
  reportingColumns = c(enumeratorID, uniquerespondantID),
  correctIsInterviewInTheCorrectSite = FALSE
)
```

**Arguments**

|                                                 |                                                                                                                                                                                        |
|-------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ds</code>                                 | dataset containing the survey (from kobo): labelled data.frame                                                                                                                         |
| <code>dsSite</code>                             | name of the field in the dataset where the site is stored: string                                                                                                                      |
| <code>dsCoordinates</code>                      | name of the fields from the dataset where the information about the GPS coordinates are stored: list of string (c('Long','Lat'))                                                       |
| <code>adminBoundaries</code>                    | dataset containing the shapefile of the site boundaries - Regardless the projection used for the shapefile, it is transformed to WGS84                                                 |
| <code>adminBoundariesSite</code>                | name of the field in the shapefile where the site is stored: string                                                                                                                    |
| <code>surveyConsent</code>                      | name of the field in the dataset where the survey consent is stored: string                                                                                                            |
| <code>reportingColumns</code>                   | (Optional, by default it is built from the enumeratorID and the uniquerespondantID) name of the columns from the dataset you want in the result: list of string (c('col1','col2',...)) |
| <code>correctIsInterviewInTheCorrectSite</code> | (Optional, by default set as FALSE) if TRUE, the site in the survey which is wrong will be replaced by the real one: boolean (TRUE/FALSE)                                              |
| <code>checkperiod</code>                        | if not null number of day before today when the check should be made                                                                                                                   |
| <code>consentForValidSurvey</code>              | value defined in the kobo form to acknowledge the surveyed person gave his consent: string                                                                                             |
| <code>uniquerespondantID</code>                 | name of the field where the survey unique ID is stored: string                                                                                                                         |
| <code>enumeratorID</code>                       | name of the field where the enumerator ID is stored: string                                                                                                                            |

**Value**

result a list that includes: \* `dst` same dataset as the inputed one but with survey marked for deletion if errors are found and `delete=TRUE` (or `NULL`) \* `ret_log` list of the errors found (or `NULL`) \* `var` a list of value (or `NULL`) \* `graph` graphical representation of the results (or `NULL`)

**Examples**

```

load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
dsSite <- "union_name"
dsCoordinates <- c("X_gps_reading_longitude", "X_gps_reading_latitude")
load(system.file("admin.RData", package = "HighFrequencyChecks"))
adminBoundaries <- admin
adminBoundariesSite <- "Union"
surveyConsent <- "survey_consent"
uniquerespondantID <- "X_uuid"
enumeratorID <- "enumerator_id"
reportingColumns <- c(enumeratorID, uniquerespondantID)

result <- isInterviewInTheCorrectSite(ds = ds,
                                     dsSite=dsSite,
                                     dsCoordinates = dsCoordinates,
                                     adminBoundaries=adminBoundaries,
                                     adminBoundariesSite=adminBoundariesSite,
                                     surveyConsent=surveyConsent,
                                     reportingColumns=reportingColumns,
                                     correctIsInterviewInTheCorrectSite=FALSE)

knitr::kable(head(result[["ret_log"]], 10))
print(result[["graph"]])

```

---

|                     |                                                                           |
|---------------------|---------------------------------------------------------------------------|
| isInterviewTooShort | <i>Check that the duration of each interview is more than a threshold</i> |
|---------------------|---------------------------------------------------------------------------|

---

**Description**

This function check that the duration of each interview is more than a specified threshold. There is an option to automatically mark for deletion the surveys which are under the threshold. Warning: If there are uncorrected mistakes in the survey dates, it can lead to have the length of the survey in seconds and this check will not performed well

**Usage**

```

isInterviewTooShort(
  ds = NULL,
  surveyConsent = NULL,
  dates = NULL,
  minimumSurveyDuration = 30,
  reportingColumns = c(enumeratorID, uniquerespondantID),
  deleteIsInterviewTooShort = FALSE
)

```

**Arguments**

|                                        |                                                                                                                                                                                                                   |
|----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ds</code>                        | dataset containing the survey (from kobo): labelled data.frame                                                                                                                                                    |
| <code>surveyConsent</code>             | name of the field in the dataset where the survey consent is stored: string                                                                                                                                       |
| <code>dates</code>                     | name of the fields where the information about the start and end date of the survey is stored: list of string (c('start_date','end_date'))                                                                        |
| <code>minimumSurveyDuration</code>     | minimum acceptable survey duration in minutes: integer                                                                                                                                                            |
| <code>reportingColumns</code>          | (Optional, by default it is built from the <code>enumeratorID</code> and the <code>uniquerespondantID</code> ) name of the columns from the dataset you want in the result: list of string (c('col1','col2',...)) |
| <code>deleteIsInterviewTooShort</code> | (Optional, by default set as FALSE) if TRUE, the survey in error will be marked as 'deletedIsInterviewTooShort': boolean (TRUE/FALSE)                                                                             |
| <code>checkperiod</code>               | if not null number of day before today when the check should be made                                                                                                                                              |
| <code>consentForValidSurvey</code>     | value defined in the kobo form to acknowledge the surveyed person gave his consent: string                                                                                                                        |
| <code>uniquerespondantID</code>        | name of the field where the survey unique ID is stored: string                                                                                                                                                    |
| <code>enumeratorID</code>              | name of the field where the enumerator ID is stored: string                                                                                                                                                       |

**Value**

result a list that includes: \* `dst` same dataset as the inputed one but with survey marked for deletion if errors are found and `delete=TRUE` (or `NULL`) \* `ret_log` list of the errors found (or `NULL`) \* `var` a list of value (or `NULL`) \* `graph` graphical representation of the results (or `NULL`)

**Examples**

```
load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
surveyConsent <- "survey_consent"
dates <- c("survey_start", "end_survey")
uniquerespondantID <- "X_uuid"
enumeratorID <- "enumerator_id"
minimumSurveyDuration <- 30
reportingColumns <- c(enumeratorID, uniquerespondantID)

result <- isInterviewTooShort(ds = ds,
                             surveyConsent=surveyConsent,
                             dates=dates,
                             minimumSurveyDuration=minimumSurveyDuration,
                             reportingColumns=reportingColumns,
                             deleteIsInterviewTooShort=FALSE)

knitr::kable(head(result[["ret_log"]], 10))
print(result[["graph"]])
```

---

isInterviewTooShortForTheHouseholdSize

*Check that the duration relative to the household size of each interview is more than a threshold*

---

## Description

This function check that the duration relative to the household size of each interview is more than a specified threshold. There is an option to automatically mark for deletion the surveys which are under the threshold. Warning: If there are uncorrected mistakes in the survey dates, it can lead to have the length of the survey in seconds and this check will not performed well

## Usage

```
isInterviewTooShortForTheHouseholdSize(
  ds = NULL,
  surveyConsent = NULL,
  dates = NULL,
  householdSize = NULL,
  minimumSurveyDurationByIndividual = 10,
  reportingColumns = c(enumeratorID, uniquerespondantID),
  deleteIsInterviewTooShortForTheHouseholdSize = FALSE
)
```

## Arguments

|                                              |                                                                                                                                                                                        |
|----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ds                                           | dataset containing the survey (from kobo): labelled data.frame                                                                                                                         |
| surveyConsent                                | name of the field in the dataset where the survey consent is stored: string                                                                                                            |
| dates                                        | name of the fields where the information about the start and end date of the survey is stored: list of string (c('start_date','end_date'))                                             |
| householdSize                                | name of the field in the dataset where the household size is stored: string                                                                                                            |
| minimumSurveyDurationByIndividual            | minimum acceptable survey duration for one individual in minutes: integer                                                                                                              |
| reportingColumns                             | (Optional, by default it is built from the enumeratorID and the uniquerespondantID) name of the columns from the dataset you want in the result: list of string (c('col1','col2',...)) |
| deleteIsInterviewTooShortForTheHouseholdSize | (Optional, by default set as FALSE) if TRUE, the survey in error will be marked as 'deletedIsInterviewTooShortForTheHouseholdSize': boolean (TRUE/FALSE)                               |
| checkperiod                                  | if not null number of day before today when the check should be made                                                                                                                   |
| consentForValidSurvey                        | value defined in the kobo form to acknowledge the surveyed person gave his consent: string                                                                                             |

uniquerespondantID      name of the field where the survey unique ID is stored: string  
 enumeratorID      name of the field where the enumerator ID is stored: string

### Value

result a list that includes: \* dst same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE (or NULL) \* ret\_log list of the errors found (or NULL) \* var a list of value (or NULL) \* graph graphical representation of the results (or NULL)

### Examples

```

load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
surveyConsent <- "survey_consent"
dates <- c("survey_start", "end_survey")
householdSize <- "consent_received.respondent_info.hh_size"
uniquerespondantID <- "X_uuid"
enumeratorID <- "enumerator_id"
minimumSurveyDurationByIndividual <- 10
reportingColumns <- c(enumeratorID, uniquerespondantID)

result <- isInterviewTooShortForTheHouseholdSize(ds = ds,
  surveyConsent=surveyConsent,
  dates=dates,
  householdSize=householdSize,
  minimumSurveyDurationByIndividual=minimumSurveyDurationByIndividual,
  reportingColumns=reportingColumns,
  deleteIsInterviewTooShortForTheHouseholdSize=FALSE)
knitr::kable(head(result[["ret_log"]], 10))
print(result[["graph"]])
  
```

---

isInterviewWithConsent

*Check that all surveys have consent*

---

### Description

This function check that all interviews in the dataset have information about the consent of the people surveyed, meaning all the field where this information is stored is not empty. There is an option to automatically mark for deletion the surveys which have not consent information.

### Usage

```

isInterviewWithConsent(
  ds = NULL,
  surveyConsent = NULL,
  reportingColumns = c(enumeratorID, uniquerespondantID),
  deleteIsInterviewWithConsent = FALSE
)
  
```

**Arguments**

**ds** dataset containing the survey (from kobo): labelled data.frame

**surveyConsent** name of the field in the dataset where the survey consent is stored: string

**reportingColumns** (Optional, by default it is built from the enumeratorID and the uniquerespondantID) name of the columns from the dataset you want in the result: list of string (c('col1','col2',...))

**deleteIsInterviewWithConsent** (Optional, by default set as FALSE) if TRUE, the survey in error will be marked as 'deletedIsInterviewWithConsent': boolean (TRUE/FALSE)

**checkperiod** if not null number of day before today when the check should be made

**consentForValidSurvey** value defined in the kobo form to acknowledge the surveyed person gave his consent: string

**uniquerespondantID** name of the field where the survey unique ID is stored: string

**enumeratorID** name of the field where the enumerator ID is stored: string

**Value**

result a list that includes: \* dst same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE (or NULL) \* ret\_log list of the errors found (or NULL) \* var a list of value (or NULL) \* graph graphical representation of the results (or NULL)

**Examples**

```
load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
surveyConsent <- "survey_consent"
uniquerespondantID <- "X_uuid"
enumeratorID <- "enumerator_id"
reportingColumns <- c(enumeratorID, uniquerespondantID)

result <- isInterviewWithConsent(ds = ds,
                                surveyConsent=surveyConsent,
                                reportingColumns=reportingColumns,
                                deleteIsInterviewWithConsent=FALSE)
knitr::kable(head(result[["ret_log"]], 10))
print(result[["graph"]])
```

---

isSurveyEndBeforeItStarts

*Surveys where end date/time is before the start date/time*

---

**Description**

This function check that all interviews in the dataset start before they end. There is an option to automatically mark for deletion the surveys which have an ending date/time before the starting ones.

**Usage**

```
isSurveyEndBeforeItStarts(
  ds = NULL,
  surveyConsent = NULL,
  dates = NULL,
  reportingColumns = c(enumeratorID, uniquerespondantID),
  deleteIsSurveyEndBeforeItStarts = FALSE
)
```

**Arguments**

|                                              |                                                                                                                                                                                        |
|----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ds</code>                              | dataset containing the survey (from kobo): labelled data.frame                                                                                                                         |
| <code>surveyConsent</code>                   | name of the field in the dataset where the survey consent is stored: string                                                                                                            |
| <code>dates</code>                           | name of the fields where the information about the start and end date of the survey is stored: list of string (c('start_date','end_date'))                                             |
| <code>reportingColumns</code>                | (Optional, by default it is built from the enumeratorID and the uniquerespondantID) name of the columns from the dataset you want in the result: list of string (c('col1','col2',...)) |
| <code>deleteIsSurveyEndBeforeItStarts</code> | (Optional, by default set as FALSE) if TRUE, the survey in error will be marked as 'deletedIsSurveyEndBeforeItStarts': boolean (TRUE/FALSE)                                            |
| <code>consentForValidSurvey</code>           | value defined in the kobo form to acknowledge the surveyed person gave his consent: string                                                                                             |
| <code>uniquerespondantID</code>              | name of the field where the survey unique ID is stored: string                                                                                                                         |
| <code>enumeratorID</code>                    | name of the field where the enumerator ID is stored: string                                                                                                                            |

**Value**

result a list that includes: \* `dst` same dataset as the inputed one but with survey marked for deletion if errors are found and `delete=TRUE` (or `NULL`) \* `ret_log` list of the errors found (or `NULL`) \* `var` a list of value (or `NULL`) \* `graph` graphical representation of the results (or `NULL`)

**Examples**

```
load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
surveyConsent <- "survey_consent"
dates <- c("survey_start", "end_survey")
uniquerespondantID <- "X_uuid"
enumeratorID <- "enumerator_id"
```

```
reportingColumns <- c(enumeratorID, uniquerespondantID)

result <- isSurveyEndBeforeItStarts(ds = ds,
                                   surveyConsent=surveyConsent,
                                   dates=dates,
                                   reportingColumns=reportingColumns,
                                   deleteIsSurveyEndBeforeItStarts=FALSE)
knitr::kable(head(result[["ret_log"]], 10))
print(result[["graph"]])
```

---

isSurveyMadeInTheFuture

*Surveys that have start date/time after system date*

---

## Description

This function check that all interviews in the dataset do not start after the current date. There is an option to automatically mark for deletion the surveys which have a start date in the future.

## Usage

```
isSurveyMadeInTheFuture(
  ds = NULL,
  surveyConsent = NULL,
  dates = NULL,
  reportingColumns = c(enumeratorID, uniquerespondantID),
  deleteIsSurveyMadeInTheFuture = FALSE
)
```

## Arguments

|                               |                                                                                                                                                                                        |
|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ds                            | dataset containing the survey (from kobo): labelled data.frame                                                                                                                         |
| surveyConsent                 | name of the field in the dataset where the survey consent is stored: string                                                                                                            |
| dates                         | name of the fields where the information about the start and end date of the survey is stored: list of string (c('start_date','end_date'))                                             |
| reportingColumns              | (Optional, by default it is built from the enumeratorID and the uniquerespondantID) name of the columns from the dataset you want in the result: list of string (c('col1','col2',...)) |
| deleteIsSurveyMadeInTheFuture | (Optional, by default set as FALSE) if TRUE, the survey in error will be marked as 'deletedIsSurveyMadeInTheFuture': boolean (TRUE/FALSE)                                              |
| checkperiod                   | if not null number of day before today when the check should be made                                                                                                                   |
| consentForValidSurvey         | value defined in the kobo form to acknowledge the surveyed person gave his consent: string                                                                                             |

```

uniquerespondantID
    name of the field where the survey unique ID is stored: string
enumeratorID      name of the field where the enumerator ID is stored: string

```

### Value

result a list that includes: \* dst same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE (or NULL) \* ret\_log list of the errors found (or NULL) \* var a list of value (or NULL) \* graph graphical representation of the results (or NULL)

### Examples

```

load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
dates <- c("survey_start", "end_survey")
surveyConsent <- "survey_consent"
uniquerespondantID <- "X_uuid"
enumeratorID <- "enumerator_id"
reportingColumns <- c(enumeratorID, uniquerespondantID)

result <- isSurveyMadeInTheFuture(ds = ds,
                                   surveyConsent=surveyConsent,
                                   dates=dates,
                                   reportingColumns=reportingColumns,
                                   deleteIsSurveyMadeInTheFuture = FALSE)

knitr::kable(head(result[["ret_log"]], 10))
print(result[["graph"]])

```

---

isSurveyOnMoreThanADay

*Surveys that do not end on the same day as they started*

---

### Description

This function check that all interviews in the dataset start and end the same day. There is an option to automatically mark for deletion the surveys which have different starting and ending dates.

### Usage

```

isSurveyOnMoreThanADay(
  ds = NULL,
  surveyConsent = NULL,
  dates = NULL,
  reportingColumns = c(enumeratorID, uniquerespondantID),
  deleteIsSurveyOnMoreThanADay = FALSE
)

```

**Arguments**

|                                           |                                                                                                                                                                                                                    |
|-------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ds</code>                           | dataset containing the survey (from kobo): labelled data.frame                                                                                                                                                     |
| <code>surveyConsent</code>                | name of the field in the dataset where the survey consent is stored: string                                                                                                                                        |
| <code>dates</code>                        | name of the fields where the information about the start and end date of the survey is stored: list of string (c('start_date','end_date'))                                                                         |
| <code>reportingColumns</code>             | (Optional, by default it is built from the <code>enumeratorID</code> and the <code>uniquerrespondantID</code> ) name of the columns from the dataset you want in the result: list of string (c('col1','col2',...)) |
| <code>deleteIsSurveyOnMoreThanADay</code> | (Optional, by default set as FALSE) if TRUE, the survey in error will be marked as 'deletedIsSurveyOnMoreThanADay': boolean (TRUE/FALSE)                                                                           |
| <code>consentForValidSurvey</code>        | value defined in the kobo form to acknowledge the surveyed person gave his consent: string                                                                                                                         |
| <code>uniquerrespondantID</code>          | name of the field where the survey unique ID is stored: string                                                                                                                                                     |
| <code>enumeratorID</code>                 | name of the field where the enumerator ID is stored: string                                                                                                                                                        |

**Value**

result a list that includes: \* `dst` same dataset as the inputed one but with survey marked for deletion if errors are found and `delete=TRUE` (or NULL) \* `ret_log` list of the errors found (or NULL) \* `var` a list of value (or NULL) \* `graph` graphical representation of the results (or NULL)

**Examples**

```
load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
surveyConsent <- "survey_consent"
dates <- c("survey_start","end_survey")
uniquerrespondantID <- "X_uuid"
enumeratorID <- "enumerator_id"
reportingColumns <- c(enumeratorID, uniquerrespondantID)

result <- isSurveyOnMoreThanADay(ds = ds,
                                surveyConsent=surveyConsent,
                                dates=dates,
                                reportingColumns=reportingColumns,
                                deleteIsSurveyOnMoreThanADay=FALSE)

knitr::kable(head(result[["ret_log"]], 10))
print(result[["graph"]])
```

---

isSurveyStartedBeforeTheAssessment

*Surveys that show start date earlier than first day of data collection*


---

## Description

This function check that all interviews in the dataset start after the actual first day of data collection. There is an option to automatically mark for deletion the surveys which have started before the first day of data collection.

## Usage

```
isSurveyStartedBeforeTheAssessment(
  ds = NULL,
  dates = NULL,
  surveyConsent = NULL,
  startDataCollection = NULL,
  reportingColumns = c(enumeratorID, uniquerespondantID),
  deleteIsSurveyStartedBeforeTheAssessment = FALSE
)
```

## Arguments

|                                          |                                                                                                                                                                                        |
|------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ds                                       | dataset containing the survey (from kobo): labelled data.frame                                                                                                                         |
| dates                                    | name of the fields where the information about the start and end date of the survey is stored: list of string (c('start_date','end_date'))                                             |
| surveyConsent                            | name of the field in the dataset where the survey consent is stored: string                                                                                                            |
| startDataCollection                      | date of the first day of the data collection: string ('yyyy-mm-dd')                                                                                                                    |
| reportingColumns                         | (Optional, by default it is built from the enumeratorID and the uniquerespondantID) name of the columns from the dataset you want in the result: list of string (c('col1','col2',...)) |
| deleteIsSurveyStartedBeforeTheAssessment | (Optional, by default set as FALSE) if TRUE, the survey in error will be marked as 'deletedIsSurveyStartedBeforeTheAssessment': boolean (TRUE/FALSE)                                   |
| checkperiod                              | if not null number of day before today when the check should be made                                                                                                                   |
| consentForValidSurvey                    | value defined in the kobo form to acknowledge the surveyed person gave his consent: string                                                                                             |
| uniquerespondantID                       | name of the field where the survey unique ID is stored: string                                                                                                                         |
| enumeratorID                             | name of the field where the enumerator ID is stored: string                                                                                                                            |

**Value**

result a list that includes: \* dst same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE (or NULL) \* ret\_log list of the errors found (or NULL) \* var a list of value (or NULL) \* graph graphical representation of the results (or NULL)

**Examples**

```
load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
dates <- c("survey_start", "end_survey")
surveyConsent <- "survey_consent"
startDataCollection <- "2018-11-11"
uniquerespondantID <- "X_uuid"
enumeratorID <- "enumerator_id"
reportingColumns <- c(enumeratorID, uniquerespondantID)

result <- isSurveyStartedBeforeTheAssessment(
  ds = ds,
  dates=dates,
  surveyConsent=surveyConsent,
  startDataCollection=startDataCollection,
  reportingColumns=reportingColumns,
  deleteIsSurveyStartedBeforeTheAssessment = FALSE)
knitr::kable(head(result[["ret_log"]], 10))
print(result[["graph"]])
```

---

isuniquerespondantIDDuplicated

*Duplicates in unique ID*

---

**Description**

This function check that all interviews in the dataset have an ID which is unique. There is an option to automatically mark for deletion the surveys which have a duplicated unique ID.

**Usage**

```
isuniquerespondantIDDuplicated(
  ds = NULL,
  uniquerespondantID = NULL,
  surveyConsent = NULL,
  attempt = NULL,
  reportingColumns = c(enumeratorID, uniquerespondantID, attempt),
  deleteIsuniquerespondantIDDuplicated = FALSE
)
```

**Arguments**

**ds** dataset containing the survey (from kobo): labelled data.frame

**uniquerespondantID** name of the field where the survey unique ID is stored: string

**surveyConsent** name of the field in the dataset where the survey consent is stored: string

**attempt** name of the field in the dataset where the interview attempt output is stored: string

**reportingColumns** (Optional, by default it is built from the enumeratorID and the uniquerespondantID) name of the columns from the dataset you want in the result: list of string (c('col1','col2',...))

**deleteIsuniquerespondantIDDuplicated** (Optional, by default set as FALSE) if TRUE, the survey in error will be marked as 'deletedIsuniquerespondantIDDuplicated': boolean (TRUE/FALSE)

**checkperiod** if not null number of day before today when the check should be made

**consentForValidSurvey** value defined in the kobo form to acknowledge the surveyed person gave his consent: string

**enumeratorID** name of the field where the enumerator ID is stored: string

**Value**

result a list that includes: \* dst same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE (or NULL) \* ret\_log list of the errors found (or NULL) \* var a list of value (or NULL) \* graph graphical representation of the results (or NULL)

**Examples**

```
load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
uniquerespondantID <- "X_uuid"
surveyConsent <- "survey_consent"
enumeratorID <- "enumerator_id"
reportingColumns <- c(enumeratorID, uniquerespondantID)

result <- isuniquerespondantIDDuplicated(ds = ds,
                                         uniquerespondantID=uniquerespondantID,
                                         surveyConsent=surveyConsent,
                                         reportingColumns=reportingColumns,
                                         deleteIsuniquerespondantIDDuplicated=FALSE)
knitr::kable(head(result[["ret_log"]], 10))
print(result[["graph"]])
```

---

isuniquerespondantIDMissing  
*Missing unique ID*

---

## Description

This function check that all interviews in the dataset have an ID. There is an option to automatically mark for deletion the surveys which have not an ID.

## Usage

```
isuniquerespondantIDMissing(
  ds = NULL,
  uniquerespondantID = NULL,
  surveyConsent = NULL,
  reportingColumns = c(enumeratorID, uniquerespondantID),
  deleteIsuniquerespondantIDMissing = FALSE
)
```

## Arguments

|                                   |                                                                                                                                                                                        |
|-----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ds                                | dataset containing the survey (from kobo): labelled data.frame                                                                                                                         |
| uniquerespondantID                | name of the field where the survey unique ID is stored: string                                                                                                                         |
| surveyConsent                     | name of the field in the dataset where the survey consent is stored: string                                                                                                            |
| reportingColumns                  | (Optional, by default it is built from the enumeratorID and the uniquerespondantID) name of the columns from the dataset you want in the result: list of string (c('coll','col2',...)) |
| deleteIsuniquerespondantIDMissing | (Optional, by default set as FALSE) if TRUE, the survey in error will be marked as 'deletedIsuniquerespondantIDMissing': boolean (TRUE/FALSE)                                          |
| consentForValidSurvey             | value defined in the kobo form to acknowledge the surveyed person gave his consent: string                                                                                             |
| enumeratorID                      | name of the field where the enumerator ID is stored: string                                                                                                                            |

## Value

result a list that includes: \* dst same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE (or NULL) \* ret\_log list of the errors found (or NULL) \* var a list of value (or NULL) \* graph graphical representation of the results (or NULL)

## Examples

```
load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
uniquerespondantID <- "X_uuid"
surveyConsent <- "survey_consent"
enumeratorID <- "enumerator_id"
reportingColumns <- c(enumeratorID, uniquerespondantID)

result <- isuniquerespondantIDMissing(ds = ds,
                                     uniquerespondantID=uniquerespondantID,
                                     surveyConsent=surveyConsent,
                                     reportingColumns=reportingColumns,
                                     deleteIsuniquerespondantIDMissing=FALSE)
knitr::kable(head(result[["ret_log"]], 10))
print(result[["graph"]])
```

---

run\_app

*Run the Shiny Application*


---

## Description

Run the Shiny Application

## Usage

```
run_app(
  onStart = NULL,
  options = list(),
  enableBookmarking = NULL,
  uiPattern = "/",
  ...
)
```

## Arguments

- |                   |                                                                                                                                                                                                                                                                                                                     |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| onStart           | A function that will be called before the app is actually run. This is only needed for shinyAppObj, since in the shinyAppDir case, a global.R file can be used for this purpose.                                                                                                                                    |
| options           | Named options that should be passed to the runApp call (these can be any of the following: "port", "launch.browser", "host", "quiet", "display.mode" and "test.mode"). You can also specify width and height parameters which provide a hint to the embedding environment about the ideal height/width for the app. |
| enableBookmarking | Can be one of "url", "server", or "disable". The default value, NULL, will respect the setting from any previous calls to <a href="#">enableBookmarking()</a> . See <a href="#">enableBookmarking()</a> for more information on bookmarking your app.                                                               |

`uiPattern` A regular expression that will be applied to each GET request to determine whether the `ui` should be used to handle the request. Note that the entire request path must match the regular expression in order for the match to be considered successful.

`...` arguments to pass to `golem_opts`. See `'?golem::get_golem_options'` for more details.

## Value

a shiny app

## Examples

```
# run_app()
```

---

|                 |                                                                              |
|-----------------|------------------------------------------------------------------------------|
| surveyBigValues | <i>Report the values greater than a specified value per specified fields</i> |
|-----------------|------------------------------------------------------------------------------|

---

## Description

This function provide a report showing all values which are greater than a certain threshold for a specified list of fields.

## Usage

```
surveyBigValues(
  ds = NULL,
  questionsSurveyBigValues = NULL,
  enumeratorID = NULL,
  reportingColumns = c(enumeratorID, uniquerespondantID),
  enumeratorCheck = FALSE
)
```

## Arguments

`ds` dataset containing the survey (from kobo): labelled data.frame

`questionsSurveyBigValues` columns name from the dataset and value you want to check against (`c(col1=value1,col2=value2,...)`): named list of integer

`enumeratorID` name of the field where the enumerator ID is stored: string

`reportingColumns` (Optional, by default it is built from the `enumeratorID` and the `uniquerespondantID`) name of the columns from the dataset you want in the result: list of string (`c('col1','col2',...)`)

`enumeratorCheck` (Optional, by default set to FALSE) specify if the report has to be displayed for each enumerator or not: boolean (TRUE/FALSE)

checkperiod      if not null number of day before today when the check should be made  
 surveyConsent    name of the field in the dataset where the survey consent is stored: string  
 consentForValidSurvey  
                  value defined in the kobo form to acknowledge the surveyed person gave his  
                  consent: string  
 uniquerespondantID  
                  name of the field where the survey unique ID is stored: string

### Value

result a list that includes: \* dst same dataset as the inputed one but with survey marked for deletion  
 if errors are found and delete=TRUE (or NULL) \* ret\_log list of the errors found (or NULL) \* var  
 a list of value (or NULL) \* graph graphical representation of the results (or NULL)

### Examples

```

load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
questionsSurveyBigValues <-c(consent_received.food_security.spend_food=25000,
                             consent_received.food_security.spend_medication=25000,
                             consent_received.food_security.spend_education=25000,
                             consent_received.food_security.spend_fix_shelter=25000,
                             consent_received.food_security.spend_clothing=25000,
                             consent_received.food_security.spend_hygiene=25000,
                             consent_received.food_security.spend_fuel=25000,
                             consent_received.food_security.spend_hh_items=25000,
                             consent_received.food_security.spend_transport=25000,
                             consent_received.food_security.spend_communication=25000,
                             consent_received.food_security.spend_tobacco=25000,
                             consent_received.food_security.spend_rent=25000,
                             consent_received.food_security.spend_debts=25000,
                             consent_received.food_security.spend_other=25000)

enumeratorID <- "enumerator_id"
uniquerespondantID <- "X_uuid"
reportingColumns <- c(enumeratorID, uniquerespondantID)

result <- surveyBigValues(ds = ds,
                          questionsSurveyBigValues=questionsSurveyBigValues,
                          enumeratorID=enumeratorID,
                          reportingColumns=reportingColumns,
                          enumeratorCheck=FALSE)

knitr::kable(head(result[["ret_log"]], 10))
  
```

---

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| surveyDistinctValues | <i>Number of distinct values (not missing) per fields</i> |
|----------------------|-----------------------------------------------------------|

---

### Description

This function provide a report showing the number of distinct values for each fields. This report  
 can be global (all the surveys) or displayed for each enumerator ID

**Usage**

```
surveyDistinctValues(ds = NULL, enumeratorID = NULL, enumeratorCheck = FALSE)
```

**Arguments**

**ds** dataset containing the survey (from kobo): labelled data.frame

**enumeratorID** name of the field where the enumerator ID is stored: string

**enumeratorCheck** (Optional, by default set to FALSE) specify if the report has to be displayed for each enumerator or not: boolean (TRUE/FALSE)

**checkperiod** if not null number of day before today when the check should be made

**surveyConsent** name of the field in the dataset where the survey consent is stored: string

**consentForValidSurvey** value defined in the kobo form to acknowledge the surveyed person gave his consent: string

**Value**

result a list that includes: \* dst same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE (or NULL) \* ret\_log list of the errors found (or NULL) \* var a list of value (or NULL) \* graph graphical representation of the results (or NULL)

**Examples**

```
load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
enumeratorID <- "enumerator_id"

result <- surveyDistinctValues(ds = ds,
                               enumeratorID=enumeratorID,
                               enumeratorCheck=FALSE)

knitr::kable(head(result[["ret_log"]], 10))
```

---

surveyMissingValues      *Report the percentage of missing values (NA) per fields*

---

**Description**

This function provide a report showing the percentage of missing values (NA) for each fields. This report can be global (all the surveys) or displayed for each enumerator ID

**Usage**

```
surveyMissingValues(ds = NULL, enumeratorID = NULL, enumeratorCheck = FALSE)
```

**Arguments**

**ds** dataset containing the survey (from kobo): labelled data.frame  
**enumeratorID** name of the field where the enumerator ID is stored: string  
**enumeratorCheck** (Optional, by default set to FALSE) specify if the report has to be displayed for each enumerator or not: boolean (TRUE/FALSE)  
**checkperiod** if not null number of day before today when the check should be made  
**surveyConsent** name of the field in the dataset where the survey consent is stored: string  
**consentForValidSurvey** value defined in the kobo form to acknowledge the surveyed person gave his consent: string

**Value**

result a list that includes: \* dst same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE (or NULL) \* ret\_log list of the errors found (or NULL) \* var a list of value (or NULL) \* graph graphical representation of the results (or NULL)

**Examples**

```

load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
enumeratorID <- "enumerator_id"

result <- surveyMissingValues(ds = ds,
                              enumeratorID=enumeratorID,
                              enumeratorCheck=FALSE)

knitr::kable(head(result[["ret_log"]], 10))
  
```

---

|                   |                                                                                |
|-------------------|--------------------------------------------------------------------------------|
| surveyOtherValues | <i>List of other distinct values (not missing) per fields other with count</i> |
|-------------------|--------------------------------------------------------------------------------|

---

**Description**

This function provide a report showing all distinct other values and the number of occurrences for each fields "other". This report can be global (all the surveys) or displayed for each enumerator ID

**Usage**

```

surveyOtherValues(
  ds = NULL,
  otherPattern = NULL,
  enumeratorID = NULL,
  enumeratorCheck = FALSE
)
  
```

**Arguments**

|                                    |                                                                                                                                |
|------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| <code>ds</code>                    | dataset containing the survey (from kobo): labelled data.frame                                                                 |
| <code>otherPattern</code>          | pattern to identify the fields containing others values (eg: <code>'_other\$'</code> ): string                                 |
| <code>enumeratorID</code>          | name of the field where the enumerator ID is stored: string                                                                    |
| <code>enumeratorCheck</code>       | (Optional, by default set to FALSE) specify if the report has to be displayed for each enumerator or not: boolean (TRUE/FALSE) |
| <code>checkperiod</code>           | if not null number of day before today when the check should be made                                                           |
| <code>surveyConsent</code>         | name of the field in the dataset where the survey consent is stored: string                                                    |
| <code>consentForValidSurvey</code> | value defined in the kobo form to acknowledge the surveyed person gave his consent: string                                     |

**Value**

result a list that includes: \* `dst` same dataset as the inputed one but with survey marked for deletion if errors are found and `delete=TRUE` (or `NULL`) \* `ret_log` list of the errors found (or `NULL`) \* `var` a list of value (or `NULL`) \* `graph` graphical representation of the results (or `NULL`)

**Examples**

```
load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
otherPattern <- "_other"
enumeratorID <- "enumerator_id"

result <- surveyOtherValues(ds = ds,
                           otherPattern=otherPattern,
                           enumeratorID=enumeratorID,
                           enumeratorCheck=FALSE)
knitr::kable(head(result[["ret_log"]], 10))
```

---

surveyOutliers

---

*Report the outlier values for all numerical field*


---

**Description**

This function provide a report showing all outlier values for each numerical fields. The function will try to automatically determine the type of distribution (between Normal and Log-Normal) based on the difference between mean and median between untransformed normalized and log transformed normalized distribution.

**Usage**

```
surveyOutliers(
  ds = NULL,
  enumeratorID = NULL,
  sdval = 2,
  reportingColumns = c(enumeratorID, uniquerespondantID),
  enumeratorCheck = FALSE
)
```

**Arguments**

|                                    |                                                                                                                                                                                                                   |
|------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ds</code>                    | dataset containing the survey (from kobo): labelled data.frame                                                                                                                                                    |
| <code>enumeratorID</code>          | name of the field where the enumerator ID is stored: string                                                                                                                                                       |
| <code>sdval</code>                 | (Optional, by default set to 2) number of standard deviation for which the data within is considered as acceptable: integer                                                                                       |
| <code>reportingColumns</code>      | (Optional, by default it is built from the <code>enumeratorID</code> and the <code>uniquerespondantID</code> ) name of the columns from the dataset you want in the result: list of string (c('coll','col2',...)) |
| <code>enumeratorCheck</code>       | (Optional, by default set to FALSE) specify if the report has to be displayed for each enumerator or not: boolean (TRUE/FALSE)                                                                                    |
| <code>checkperiod</code>           | if not null number of day before today when the check should be made                                                                                                                                              |
| <code>surveyConsent</code>         | name of the field in the dataset where the survey consent is stored: string                                                                                                                                       |
| <code>consentForValidSurvey</code> | value defined in the kobo form to acknowledge the surveyed person gave his consent: string                                                                                                                        |
| <code>uniquerespondantID</code>    | name of the field where the survey unique ID is stored: string                                                                                                                                                    |

**Value**

`dst` same dataset as the inputed one but with survey marked for deletion if errors are found and `delete=TRUE` (or `NULL`)

`ret_log` list of the errors found (or `NULL`)

`var` a list of value (or `NULL`)

`graph` graphical representation of the results (or `NULL`)

---

|                   |                                                                            |
|-------------------|----------------------------------------------------------------------------|
| surveySmallValues | <i>Report the values lower than a specified value per specified fields</i> |
|-------------------|----------------------------------------------------------------------------|

---

### Description

This function provide a report showing all values which are lower than a certain threshold for a specified list of fields.

### Usage

```
surveySmallValues(
  ds = NULL,
  questionsSurveySmallValues = NULL,
  enumeratorID = NULL,
  reportingColumns = c(enumeratorID, uniquerespondantID),
  enumeratorCheck = FALSE
)
```

### Arguments

|                            |                                                                                                                                                                                        |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ds                         | dataset containing the survey (from kobo): labelled data.frame                                                                                                                         |
| questionsSurveySmallValues | columns name from the dataset and value you want to check against (c(col1=value1,col2=value2,...)): named list of integer                                                              |
| enumeratorID               | name of the field where the enumerator ID is stored: string                                                                                                                            |
| reportingColumns           | (Optional, by default it is built from the enumeratorID and the uniquerespondantID) name of the columns from the dataset you want in the result: list of string (c('col1','col2',...)) |
| enumeratorCheck            | (Optional, by default set to FALSE) specify if the report has to be displayed for each enumerator or not: boolean (TRUE/FALSE)                                                         |
| checkperiod                | if not null number of day before today when the check should be made                                                                                                                   |
| surveyConsent              | name of the field in the dataset where the survey consent is stored: string                                                                                                            |
| consentForValidSurvey      | value defined in the kobo form to acknowledge the surveyed person gave his consent: string                                                                                             |
| uniquerespondantID         | name of the field where the survey unique ID is stored: string                                                                                                                         |

### Value

result a list that includes: \* dst same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE (or NULL) \* ret\_log list of the errors found (or NULL) \* var a list of value (or NULL) \* graph graphical representation of the results (or NULL)

**Examples**

```
load(system.file("sample_dataset.RData", package = "HighFrequencyChecks"))
ds <- sample_dataset
questionsSurveySmallValues <-c(consent_received.food_security.spend_food=25000,
                               consent_received.food_security.spend_medication=25000,
                               consent_received.food_security.spend_education=25000,
                               consent_received.food_security.spend_fix_shelter=25000,
                               consent_received.food_security.spend_clothing=25000,
                               consent_received.food_security.spend_hygiene=25000,
                               consent_received.food_security.spend_fuel=25000,
                               consent_received.food_security.spend_hh_items=25000,
                               consent_received.food_security.spend_transport=25000,
                               consent_received.food_security.spend_communication=25000,
                               consent_received.food_security.spend_tobacco=25000,
                               consent_received.food_security.spend_rent=25000,
                               consent_received.food_security.spend_debts=25000,
                               consent_received.food_security.spend_other=25000)

enumeratorID <- "enumerator_id"
uniquerespondantID <- "X_uuid"
reportingColumns <- c(enumeratorID, uniquerespondantID)

result <- surveySmallValues(ds = ds,
                            questionsSurveySmallValues=questionsSurveySmallValues,
                            enumeratorID=enumeratorID,
                            reportingColumns=reportingColumns,
                            enumeratorCheck=FALSE)

knitr::kable(head(result[["ret_log"]], 10))
```

# Index

assessmentDailyValidSurveys, [3](#)  
assessmentDuration, [4](#)  
assessmentDurationOutliers, [5](#)  
assessmentInterviewTime, [6](#)  
assessmentProductivity, [7](#)  
assessmentTrackingSheet, [8](#)

enableBookmarking(), [34](#)  
enumeratorErrorsSummary, [10](#)  
enumeratorIsLazy, [11](#)  
enumeratorProductivity, [12](#)  
enumeratorProductivityOutliers, [13](#)  
enumeratorSurveysConsent, [14](#)  
enumeratorSurveysDuration, [15](#)

isInterviewAtTheSamplePoint, [16](#)  
isInterviewCompleted, [18](#)  
isInterviewInTheCorrectSite, [19](#)  
isInterviewTooShort, [21](#)  
isInterviewTooShortForTheHouseholdSize,  
    [23](#)  
isInterviewWithConsent, [24](#)  
isSurveyEndBeforeItStarts, [25](#)  
isSurveyMadeInTheFuture, [27](#)  
isSurveyOnMoreThanADay, [28](#)  
isSurveyStartedBeforeTheAssessment, [30](#)  
isuniquerespondantIDDuplicated, [31](#)  
isuniquerespondantIDMissing, [33](#)

run\_app, [34](#)

surveyBigValues, [35](#)  
surveyDistinctValues, [36](#)  
surveyMissingValues, [37](#)  
surveyOtherValues, [38](#)  
surveyOutliers, [39](#)  
surveySmallValues, [41](#)