Package 'HighFrequencyChecks'

December 10, 2019

Type Package

Title High Frequency Checks

Version 0.2.0

Author Yannick Pascaud

Maintainer Yannick Pascaud <yannick.pascaud@free.fr>

Depends R (>= 2.10)

Description This package groups the functions used to perform the High Frequency checks during the data collection.
These are usual functions to be ran periodically during the data collection process to check for possible errors and 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 you to build a report.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports dplyr,
sp,
rgdal,
rgeos,
geosphere,
stringi,
data.table,
lazyeval,
plotly,
outliers,
gsubfn

Suggests knitr, rmarkdown

VignetteBuilder knitr

R topics documented:

`admin` *boundary dataset to be used as an example*

Description

This dataset contains the boudaries for the Unions in both Ukhia and Teknaf.

Usage

```
sample_dataset
```

Format

spatial dataset

chk1a_interview_completed *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

```
chk1a_interview_completed(ds,
                          survey_consent,
                          dates,
                          reportingcol,
                          delete)
```

Arguments

ds	dataset as a data.frame object
survey_consent	name as a string of the field in the dataset where the survey consent is stored
dates	fields as a list of string where the survey start and end date is stored (c('start_date','end_date'))
reportingcol	columns as a list of string name from the dataset you want in the result (c('col1','col2',...))
delete	delete action to be done as a boolean (TRUE/FALSE)

Value

ds	same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE
errors	list of the errors found

Author(s)

Yannick Pascaud

Examples

```
df<-sample_dataset
sc<-"survey_consent"
dt<-c("survey_start","end_survey")
rc<-c("enumerator_id","X_uuid")
dl<-FALSE

chk1a_interview_completed(df, sc, dt, rc, dl)
```

chk1b_survey_consent *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

```
chk1b_survey_consent(ds,
                     survey_consent,
                     reportingcol,
                     delete)
```

Arguments

ds	dataset as a data.frame object
survey_consent	name as a string of the field in the dataset where the survey consent is stored
reportingcol	columns as a list of string name from the dataset you want in the result (c('col1','col2',...))
delete	delete action to be done as a boolean (TRUE/FALSE)

Value

ds	same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE
errors	list of the errors found

Author(s)

Yannick Pascaud

Examples

```
df<-sample_dataset
sc<-"survey_consent"
rc<-c("enumerator_id","X_uuid")
dl<-FALSE

chk1b_survey_consent(df, sc, rc, dl)
```

chk1dii_GIS_Xm *check if the surveys fall within Xm radius of 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.

Usage

```
chk1dii_GIS_Xm(pts,
               ds,
               ds_coord,
               buff=10,
               survey_consent,
               reportingcol,
               delete)
```

Arguments

pts	dataset containing the shapefile
ds	dataset as a data.frame object
ds_coord	columns as a list of string name from the dataset where the GPS coordinates are stored (c('Long','Lat'))
buff	value as an integer in meter to determine the buffer from a sampled point which is acceptable
survey_consent	name as a string of the field in the dataset where the survey consent is stored
reportingcol	columns as a list of string name from the dataset you want in the result (c('col1','col2',...))
delete	delete action to be done as a boolean (TRUE/FALSE)

Value

ds	same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE
errors	list of the errors found

Warning

Regardless the projection used for the shapefile, it is transformed to WGS84

Note

One internal function "make_GeodesicBuffer" used to create the buffers is created by Valentin https://stackoverflow.com/users/5193830/valentin

Author(s)

Yannick Pascaud

Examples

```
pts<-SamplePts
df<-sample_dataset
df_coord<-c("X_gps_reading_longitude","X_gps_reading_latitude")
bu<-10
sc<-"survey_consent"
rc<-c("enumerator_id","X_uuid")
dl<-FALSE

chk1dii_GIS_Xm(pts, df, df_coord, bu, sc, rc, dl)
```

`chk1di_GIS_site` *Check if the surveys are made in the correct 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

```
chk1di_GIS_site(adm,
                ds,
                ds_site,
                ds_coord,
                adm_site,
                survey_consent,
                reportingcol,
                correct)
```

Arguments

`adm`	dataset containing the shapefile
`ds`	dataset as a data.frame object
`ds_site`	name as a string of the field in the dataset where the site is stored
`ds_coord`	columns as a list of string name from the dataset where the GPS coordinates are stored (c('Long','Lat'))
`adm_site`	name as a string of the field in the shapefile where the site is stored
`survey_consent`	name as a string of the field in the dataset where the survey consent is stored
`reportingcol`	columns as a list of string name from the dataset you want in the result (c('col1','col2',...))
`correct`	correction action to be done as a boolean (TRUE/FALSE)

Value

`ds`	same dataset as the inputed one but with the site in the surveys corrected if errors are found and correct=TRUE
`errors`	list of the errors found

Warning

Regardless the projection used for the shapefile, it is transformed to WGS84

Author(s)

Yannick Pascaud

Examples

```
admin<-admin
df<-sample_dataset
df_site<-"union_name"
df_coord<-c("X_gps_reading_longitude","X_gps_reading_latitude")
admin_site<-"Union"
sc<-"survey_consent"
rc<-c("enumerator_id","X_uuid")
co<-FALSE

chk1di_GIS_site(admin, df, df_site, df_coord, admin_site, sc, rc, co)
```

---

chk2a_missing_id *Check that all interviews have an 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

```
chk2a_missing_id(ds,
                 UniqueID,
                 survey_consent,
                 reportingcol,
                 delete)
```

Arguments

ds	dataset as a data.frame object
UniqueID	name as a string of the field in the dataset where the unique ID is stored
survey_consent	name as a string of the field in the dataset where the survey consent is stored
reportingcol	columns as a list of string name from the dataset you want in the result (c('col1','col2',...))
delete	delete action to be done as a boolean (TRUE/FALSE)

Value

ds	same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE
errors	list of the errors found

Author(s)

Yannick Pascaud

Examples

```
df<-sample_dataset
uuid<-"X_uuid"
sc<-"survey_consent"
rc<-c("enumerator_id","X_uuid")
dl<-FALSE

chk2a_missing_id(df, uuid, sc, rc, dl)
```

---

chk2b_unique_id *check for 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

```
chk2b_unique_id(ds,
                UniqueID,
                survey_consent,
                reportingcol,
                delete)
```

Arguments

ds	dataset as a data.frame object
UniqueID	name as a string of the field in the dataset where the unique ID is stored
survey_consent	name as a string of the field in the dataset where the survey consent is stored
reportingcol	columns as a list of string name from the dataset you want in the result (c('col1','col2',...))
delete	delete action to be done as a boolean (TRUE/FALSE)

Value

ds	same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE
errors	list of the errors found

Author(s)

Yannick Pascaud

Examples

```
df<-sample_dataset
uuid<-"X_uuid"
sc<-"survey_consent"
rc<-c("enumerator_id","X_uuid")
dl<-FALSE

chk2b_unique_id(df, uuid, sc, rc, dl)
```

`chk3a_date_mistake` *Check for 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

```
chk3a_date_mistake(ds,
                   survey_consent,
                   dates,
                   reportingcol,
                   delete)
```

Arguments

`ds`	dataset as a data.frame object
`survey_consent`	name as a string of the field in the dataset where the survey consent is stored
`dates`	fields as a list of string where the survey start and end date is stored (c('start_date','end_date'))
`reportingcol`	columns as a list of string name from the dataset you want in the result (c('col1','col2',...))
`delete`	delete action to be done as a boolean (TRUE/FALSE)

Value

`ds`	same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE
`errors`	list of the errors found

Author(s)

Yannick Pascaud

Examples

```
df<-sample_dataset
sc<-"survey_consent"
dt<-c("survey_start","end_survey")
rc<-c("enumerator_id","X_uuid")
dl<-FALSE

chk3a_date_mistake(df, sc, dt, rc, dl)
```

chk3b_date_mistake *Check for 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

```
chk3b_date_mistake(ds,
                   survey_consent,
                   dates,
                   reportingcol,
                   delete)
```

Arguments

ds	dataset as a data.frame object
survey_consent	name as a string of the field in the dataset where the survey consent is stored
dates	fields as a list of string where the survey start and end date is stored (c('start_date','end_date'))
reportingcol	columns as a list of string name from the dataset you want in the result (c('col1','col2',...))
delete	delete action to be done as a boolean (TRUE/FALSE)

Value

ds	same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE
errors	list of the errors found

Author(s)

Yannick Pascaud

Examples

```
df<-sample_dataset
sc<-"survey_consent"
dt<-c("survey_start","end_survey")
rc<-c("enumerator_id","X_uuid")
dl<-FALSE

chk3b_date_mistake(df, sc, dt, rc, dl)
```

chk3c_date_mistake *Check for 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

```
chk3c_date_mistake(ds,
                   survey_consent,
                   dates,
                   start_collection,
                   reportingcol,
                   delete)
```

Arguments

ds	dataset as a data.frame object
survey_consent	name as a string of the field in the dataset where the survey consent is stored
dates	fields as a list of string where the survey start and end date is stored (c('start_date','end_date'))
start_collection	date as a string of the first day of data collection ('yyyy-mm-dd')
reportingcol	columns as a list of string name from the dataset you want in the result (c('col1','col2',...))
delete	delete action to be done as a boolean (TRUE/FALSE)

Value

ds	same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE
errors	list of the errors found

Author(s)

Yannick Pascaud

Examples

```
df<-sample_dataset
sc<-"survey_consent"
dt<-c("survey_start","end_survey")
st<-"2018-11-11"
rc<-c("enumerator_id","X_uuid")
dl<-FALSE

chk3c_date_mistake(df, sc, dt, st, rc, dl)
```

chk3d_date_mistake *Check for 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

```
chk3d_date_mistake(ds,
                   survey_consent,
                   dates,
                   reportingcol,
                   delete)
```

Arguments

ds	dataset as a data.frame object
survey_consent	name as a string of the field in the dataset where the survey consent is stored
dates	fields as a list of string where the survey start and end date is stored (c('start_date','end_date'))
reportingcol	columns as a list of string name from the dataset you want in the result (c('col1','col2',...))
delete	delete action to be done as a boolean (TRUE/FALSE)

Value

ds	same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE
errors	list of the errors found

Author(s)

Yannick Pascaud

Examples

```
df<-sample_dataset
sc<-"survey_consent"
dt<-c("survey_start","end_survey")
rc<-c("enumerator_id","X_uuid")
dl<-FALSE

chk3d_date_mistake(df, sc, dt, rc, dl)
```

chk4aiii_missing_pct *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

```
chk4aiii_missing_pct(ds,
                     enumeratorID,
                     enumeratorcheck)
```

Arguments

ds	dataset as a data.frame object
enumeratorID	name as a string of the field in the dataset where the enumerator ID is stored
enumeratorcheck	specify if the report has to be displayed for each enumerator or not as a boolean (TRUE/FALSE)

Value

logf	the report

Author(s)

Yannick Pascaud

Examples

```
df<-sample_dataset
eid<-"enumerator_id"
ec<-FALSE

chk4aiii_missing_pct(df, eid, ec)
```

chk4bii_distinct_values *Report the number of distinct values per fields*

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

```
chk4bii_distinct_values(ds,
                        enumeratorID,
                        enumeratorcheck)
```

Arguments

ds	dataset as a data.frame object
enumeratorID	name as a string of the field in the dataset where the enumerator ID is stored
enumeratorcheck	specify if the report has to be displayed for each enumerator or not as a boolean (TRUE/FALSE)

Value

logf	the report

Author(s)

Yannick Pascaud

Examples

```
df<-sample_dataset
eid<-"enumerator_id"
ec<-FALSE

chk4bii_distinct_values(df, eid, ec)
```

---

chk4biv_others_values *Report the values for other responses per fields*

Description

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

Usage

```
chk4biv_others_values(ds,
                      otherpattern,
                      enumeratorID,
                      enumeratorcheck)
```

Arguments

ds	dataset as a data.frame object
otherpattern	pattern as string to identify the fields containing others values ('_other$')
enumeratorID	name as a string of the field in the dataset where the enumerator ID is stored
enumeratorcheck	specify if the report has to be displayed for each enumerator or not as a boolean (TRUE/FALSE)

Value

logf	the report

Author(s)

Yannick Pascaud

Examples

```
df<-sample_dataset
op<-"_other$"
eid<-"enumerator_id"
ec<-FALSE

chk4biv_others_values(df, op, eid, ec)
```

---

chk4d_outliers *Report the outlier values for all numerical fields*

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

```
chk4d_outliers(ds,
               sdval,
               reportingcol,
               enumeratorID,
               enumeratorcheck)
```

Arguments

ds	dataset as a data.frame object
sdval	number of standard deviation for which the data within is considered as acceptable
reportingcol	columns as a list of string name from the dataset you want in the result (c('col1','col2',...))
enumeratorID	name as a string of the field in the dataset where the enumerator ID is stored (NOT USED: use the reportingcol instead)
enumeratorcheck	specify if the report has to be displayed for each enumerator or not as a boolean (TRUE/FALSE) (NOT USED: use the reportingcol instead)

Value

logf	the report

Author(s)

Yannick Pascaud

Examples

```
df<-sample_dataset
sdv<-2
rc<-c("enumerator_id","X_uuid")
eid<-"enumerator_id"
ec<-FALSE

chk4d_outliers(df, sdv, rc , eid, ec)
```

---

chk4e_values_greater_X

Report the values greater than a specified value per specified fields

---

Description

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

Usage

```
chk4e_values_greater_X(ds,
                       questions,
                       value,
                       reportingcol,
                       enumeratorID,
                       enumeratorcheck)
```

Arguments

ds	dataset as a data.frame object
questions	columns as a list of string name from the dataset you want to check against (c('col1','col2',...))
value	maximum acceptable value as integer for the checked fields
reportingcol	columns as a list of string name from the dataset you want in the result (c('col1','col2',...))
enumeratorID	name as a string of the field in the dataset where the enumerator ID is stored (NOT USED: use the reportingcol instead)
enumeratorcheck	specify if the report has to be displayed for each enumerator or not as a boolean (TRUE/FALSE) (NOT USED: use the reportingcol instead)

Value

logf	the report

Author(s)

Yannick Pascaud

Examples

```
df<-sample_dataset
qu<-c("consent_received.food_security.spend_food",
      "consent_received.food_security.spend_medication",
      "consent_received.food_security.spend_education",
      "consent_received.food_security.spend_fix_shelter",
      "consent_received.food_security.spend_clothing",
      "consent_received.food_security.spend_hygiene",
      "consent_received.food_security.spend_fuel",
      "consent_received.food_security.spend_hh_items",
      "consent_received.food_security.spend_transport",
      "consent_received.food_security.spend_communication",
      "consent_received.food_security.spend_tobacco",
      "consent_received.food_security.spend_rent",
      "consent_received.food_security.spend_debts",
      "consent_received.food_security.spend_other")
v<-25000
rc<-c("enumerator_id","X_uuid")
eid<-"enumerator_id"
ec<-FALSE

chk4e_values_greater_X(df, qu, v, rc, eid, ec)
```

`chk5a_duration` *Compute the average and total time for the surveys*

Description

This function compute the average and total time for the surveys

Usage

```
chk5a_duration(ds,
               dates)
```

Arguments

`ds`	dataset as a data.frame object
`dates`	fields as a list of string where the survey start and end date is stored (c('start_date','end_date'))

Value

`avg`	average time per survey
`tot`	total time

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

Author(s)

Yannick Pascaud

Examples

```
df<-sample_dataset
dt<-c("survey_start","end_survey")

chk5a_duration(df, dt)
```

`chk5b_duration_Xmin` *Check that the duration of each interview is more than a threshold*

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.

Usage

```
chk5b_duration_Xmin(ds,
                    survey_consent,
                    dates,
                    reportingcol,
                    minduration,
                    delete)
```

Arguments

`ds`	dataset as a data.frame object
`survey_consent`	name as a string of the field in the dataset where the survey consent is stored
`dates`	fields as a list of string where the survey start and end date is stored (c('start_date','end_date'))
`reportingcol`	columns as a list of string name from the dataset you want in the result (c('col1','col2',...))
`minduration`	minimum acceptable survey duration as integer
`delete`	delete action to be done as a boolean (TRUE/FALSE)

Value

`ds`	same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE
`errors`	list of the errors found

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

Author(s)

Yannick Pascaud

Examples

```
df<-sample_dataset
sc<-"survey_consent"
dt<-c("survey_start","end_survey")
rc<-c("enumerator_id","X_uuid")
md<-30
dl<-FALSE

chk5b_duration_Xmin(df, sc, dt, rc, md, dl)
```

`chk5c_duration_Xmin_HHSize` *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.

Usage

```
chk5c_duration_Xmin_HHSize(ds,
                           survey_consent,
                           dates,
                           HHSize,
                           reportingcol,
                           minduration,
                           delete)
```

Arguments

`ds`	dataset as a data.frame object
`survey_consent`	name as a string of the field in the dataset where the survey consent is stored
`dates`	fields as a list of string where the survey start and end date is stored (c('start_date','end_date'))
`HHSize`	name as a string of the field in the dataset where the household size is stored
`reportingcol`	columns as a list of string name from the dataset you want in the result (c('col1','col2',...))
`minduration`	minimum acceptable survey duration as integer
`delete`	delete action to be done as a boolean (TRUE/FALSE)

Value

`ds`	same dataset as the inputed one but with survey marked for deletion if errors are found and delete=TRUE
`errors`	list of the errors found

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

Author(s)

Yannick Pascaud

Examples

```
df<-sample_dataset
sc<-"survey_consent"
dt<-c("survey_start","end_survey")
hs<-"consent_received.respondent_info.hh_size"
rc<-c("enumerator_id","X_uuid")
md<-10
dl<-FALSE

chk5c_duration_Xmin_HHSize(df, sc, dt, hs, rc, md, dl)
```

---

chk5d_duration_outliers

Report the outlier durations for the surveys

Description

This function report the outlier durations for the surveys

Usage

```
chk5d_duration_outliers(ds,
                        dates,
                        sdval,
                        reportingcol)
```

Arguments

ds	dataset as a data.frame object
dates	fields as a list of string where the survey start and end date is stored (c('start_date','end_date'))
sdval	number of standard deviation for which the duration within is considered as acceptable
reportingcol	columns as a list of string name from the dataset you want in the result (c('col1','col2',...))

Value

logf	the report

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

Author(s)

Yannick Pascaud

Examples

```
df<-sample_dataset
dt<-c("survey_start","end_survey")
sdv<-2
rc<-c("enumerator_id","X_uuid")

chk5d_duration_outliers(df, dt, sdv, rc)
```

`chk6a_refusal` *Check the percentage of survey refusals by enumerator*

Description

This function display the percentage of survey refusal per enumerator.

Usage

```
chk6a_refusal(ds,
              survey_consent,
              enumeratorID)
```

Arguments

`ds`	dataset as a data.frame object
`survey_consent`	name as a string of the field in the dataset where the survey consent is stored
`enumeratorID`	name as a string of the field in the dataset where the enumerator ID is stored

Value

`logf`	the report

Author(s)

Yannick Pascaud

Examples

```
df<-sample_dataset
sc<-"survey_consent"
eid<-"enumerator_id"

chk6a_refusal(df, sc, eid)
```

chk6b_duration *Check the average interview duration by enumerator*

Description

This function display the average interview duration per enumerator.

Usage

```
chk6b_duration(ds,
               dates,
               enumeratorID)
```

Arguments

ds	dataset as a data.frame object
dates	fields as a list of string where the survey start and end date is stored (c('start_date','end_date'))
enumeratorID	name as a string of the field in the dataset where the enumerator ID is stored

Value

logf	the report

Author(s)

Yannick Pascaud

Examples

```
df<-sample_dataset
dt<-c("survey_start","end_survey")
eid<-"enumerator_id"

chk6b_duration(df, dt, eid)
```

chk6c_nb_survey *Check the number of surveys by enumerator*

Description

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

Usage

```
chk6c_nb_survey(ds,
                surveydate,
                enumeratorID)
```

Arguments

ds	dataset as a data.frame object
surveydate	name as a string of the field in the dataset where the date of the survey is stored
enumeratorID	name as a string of the field in the dataset where the enumerator ID is stored

Value

logf	the report

Author(s)

Yannick Pascaud

Examples

```
df<-sample_dataset
sdte<-"survey_date"
eid<-"enumerator_id"

chk6c_nb_survey(df, sdte, eid)
```

---

chk6f_productivity *Check the surveyors with very low or high productivity*

Description

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

Usage

```
chk6f_productivity(ds,
                   enumeratorID,
                   surveydate,
                   sdval)
```

Arguments

ds	dataset as a data.frame object
enumeratorID	name as a string of the field in the dataset where the enumerator ID is stored
surveydate	name as a string of the field in the dataset where the date of the survey is stored
sdval	number of standard deviation for which the duration within is considered as normal

Value

logf	the report

Author(s)

Yannick Pascaud

Examples

```
df<-sample_dataset
eid<-"enumerator_id"
sdte<-"survey_date"
sdv<-2

chk6f_productivity(df, eid, sdte, sdv)
```

`chk6g_question_less_X_answers` *Check the enumerators who pick up less than X answers per specific question*

Description

This function display the surveyors who picked up less than a specified amount of answers per specific question.

Usage

```
chk6g_question_less_X_answers(ds,
                              enumeratorID,
                              questions,
                              minnbanswers)
```

Arguments

`ds`	dataset as a data.frame object
`enumeratorID`	name as a string of the field in the dataset where the enumerator ID is stored
`questions`	columns as a list of string name from the dataset you want to check against (c('col1','col2',...))
`minnbanswers`	minimum anumber of answers expected per question

Value

`logf`	the report

Author(s)

Yannick Pascaud

Examples

```
df<-sample_dataset
eid<-"enumerator_id"
qu<-c("consent_received.shelter_nfi.non_food_items[.]",
      "consent_received.food_security.main_income[.]",
      "consent_received.child_protection.boy_risk[.]",
      "consent_received.child_protection.girl_risk[.]")
mna<-3

chk6g_question_less_X_answers(df, eid, qu, mna)
```

chk7aii_productivity_hist *Overall productivity histogram*

Description

This function create an histogram showing the overall productivity per consent status per day.

Usage

```
chk7aii_productivity_hist(ds,
                          surveydate,
                          dateformat,
                          survey_consent)
```

Arguments

ds	dataset as a data.frame object
surveydate	name as a string of the field in the dataset where the date of the survey is stored
dateformat	format as a string used for the date ('%m/%d/%Y')
survey_consent	name as a string of the field in the dataset where the survey consent is stored

Value

graph	the graphic as a plot.ly object

Author(s)

Yannick Pascaud

Examples

```
df<-sample_dataset
sdte<-"survey_date"
dtf<-"%m/%d/%Y"
sc<-"survey_consent"

chk7aii_productivity_hist(df, sdte, dtf, sc)
```

chk7ai_productivity *Summary of daily average productivity*

Description

This function display the number of surveys conducted per day.

Usage

```
chk7ai_productivity(ds,
                    surveydate,
                    dateformat,
                    survey_consent)
```

Arguments

ds	dataset as a data.frame object
surveydate	name as a string of the field in the dataset where the date of the survey is stored
dateformat	format as a string used for the date ('%m/%d/%Y')
survey_consent	name as a string of the field in the dataset where the survey consent is stored

Value

logf	the report

Author(s)

Yannick Pascaud

Examples

```
df<-sample_dataset
sdte<-"survey_date"
dtf<-"%m/%d/%Y"
sc<-"survey_consent"

chk7ai_productivity(df, sdte, dtf, sc)
```

---

chk7bii_tracking *Overall tracking sheet*

Description

This function display the overall tracking sheet.

Usage

```
chk7bii_tracking(ds,
                 sf,
                 dssite,
                 sfsite,
                 survcons,
                 sftarget,
                 sfnbpts,
                 formul,
                 colorder)
```

Arguments

ds	dataset as a data.frame object
sf	sampling frame as a data.frame object
dssite	name as a string of the field in the dataset where the site is stored
sfsite	name as a string of the field in the sampling frame where the site is stored
survcons	name as a string of the field in the dataset where the survey consent is stored
sftarget	name as a string of the field where the target number of survey is stored in the sampling frame
sfnbpts	name as a string of the field where the number of points generated is stored in the sampling frame
formul	formulas as a list of string used to compute the final number of eligible surveys and the variance from the target (C('formula1','formula2')). the values/fields available are: done and the ones generated according the survey consent values (one per value)
colorder	column names as a list of string to order the colums in the result (C('col1','col2',...)). the columns available are: site, done, final, variance and the ones generated according the survey consent values (one per value)

Value

logf	the report

Author(s)

Yannick Pascaud

Examples

```
df=sample_dataset
sf=SampleSize
dssite="union_name"
sfsite="Union"
sc="survey_consent"
sftarget="SS"
sfnbpts="TotPts"
#formul=c("done-no-not_eligible-deleted","done-no-not_eligible-deleted-SS")
#colorder=c("site","SS","Provision","done","not_eligible","no","deleted","yes","final","variance")
formul=c("done-no-not_eligible","done-no-not_eligible-SS")
colorder=c("site","SS","TotPts","done","not_eligible","no","yes","final","variance")

chk7bii_tracking(df, sf, dssite, sfsite, sc, sftarget, sfnbpts, formul, colorder)
```

---

chk7bi_nb_status *Daily number of survey per consent status*

Description

This function display the number of surveys conducted per day per constent status.

Usage

```
chk7bi_nb_status(ds,
                 surveydate,
                 dateformat,
                 survey_consent)
```

Arguments

ds	dataset as a data.frame object
surveydate	name as a string of the field in the dataset where the date of the survey is stored
dateformat	format as a string used for the date ('%m/%d/%Y')
survey_consent	name as a string of the field in the dataset where the survey consent is stored

Value

logf	the report

Author(s)

Yannick Pascaud

Examples

```
df<-sample_dataset
sdte<-"survey_date"
dtf<-"%m/%d/%Y"
sc<-"survey_consent"

chk7bi_nb_status(df, sdte, dtf, sc)
```

SamplePts *points sampled to be surveyed dataset to be used as an example*

Description

This dataset is a sample of the points to be surveyed in both Ukhia and Teknaf.

For privacy purpose, the GPS coordinates are faked but are still located in the designated areas.

Usage

```
sample_dataset
```

Format

spatial dataset

SampleSize *sample size used*

Description

This dataset comes from the Host Community Multi Sector Needs Assessment conducted in 2018 by IMPACT/REACH on behalf of the ISCG and funded by ECHO. This is the sample size per union used in both Ukhia and Teknaf.

Usage

```
sample_dataset
```

Format

A data frame with 11 observations on 7 variables.

sample_dataset *household dataset to be used as an example*

Description

This dataset comes from the Host Community Multi Sector Needs Assessment conducted in 2018 by IMPACT/REACH on behalf of the ISCG and funded by ECHO. This is a sample of the data collected in both Ukhia and Teknaf.

For privacy purpose, the GPS coordinates are faked but are still located in the designated areas.

Usage

```
sample_dataset
```

Format

A data frame with 498 observations on 587 variables.

Index