esmprep - An R package to prepare data from Experience-Sampling methods for statistical analysis

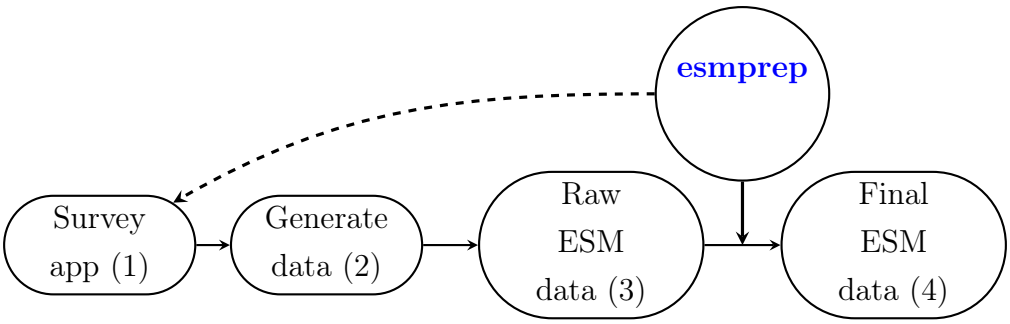
Marcel Miché
Mail: esmprep@gmail.com

July 5, 2019

Contents

Summary

An ESM (**E**xperience **S**ampling **M**ethodology) project roughly contains 4 elements:



esmprep prepares raw ESM data for statistical analysis, i.e. it generates one final ESM dataset. Also it can help increasing the survey app quality, if required (dashed arrow).

Preface

The explicit goal in developing the **esmprep** package was to make it easy to use. Everybody who is able to install R (https://www.r-project.org/) and then within R first install a package (`install.packages('esmprep', dependencies=TRUE)`) and load it (`library(esmprep)`), should be able to use the **esmprep** package. For any further guidance use this document.
BEWARE: If the ESM app contains weaknesses of any kind, e.g. participants can fill out more than one questionnaire within a certain time interval even though they are not supposed to do so, detecting such 'errors' and handling them afterwards is indicated because analysing ESM data simply requires a clean(ed) dataset.

1 Introduction

The **Experience Sampling Methodology** (**ESM**) comprises research, in which people fill out 1+ questionnaires on an electronic mobile device for a certain period of time (1+ consecutive days) in their natural environment.
Research projects including ESM have increased in number, a trend which can be expected to continue. One of the purported reasons for this increase is the easy implementation of an ESM questionnaire app on a mobile device. However, it isn't 'easy' per se!
The justification as well as the motivation for developing the **esmprep** package is 3-fold:

1. To save researchers (students, doctoral candidates, postdocs, ...) precious time, which they otherwise might have to spend on preparing the raw ESM data[1].

2. To help preparing ESM datasets prior to statistical analysis.

3. To be of help in improving the ESM questionnaire app, which optimally prevents unintended data to appear in the raw ESM dataset.

On the subsequent pages I will introduce you to a hierarchy of data cleaning procedures, applied to raw ESM data. The data itself is simulated, the errors, on the other hand, stem from real ESM research projects! And so does the data structure. It is therefore not 'unrealistically' messy.

[1] Depending both on their own software skills and on the quality of the ESM questionnaire app.

2 Applied ESM

The combinations of ESM research options cannot exceed these three scenarios:

1. Number of ESM questionnaire versions = 1; i.e. during the entire ESM period all participants always get to answer the exact same questions in the same order.

2. Number of ESM questionnaire versions > 1; e.g. the ESM questions' content and/or number differs across an ESM day.

3. The times of when the participant shall answer the daily ESM questionnaire(s) either are scheduled (alarm signal/ prompt) and/or the participant fills out ESM questionnaires due to personal reasons (event contingent).

The third point can be split into 3 separate points, i.e. **all** daily ESM questionnaires are prompted, **none** is prompted, or **some** are promted and some are not. The **esmprep** package covers all of these cases. See table 1.

Table 1: Combinations (C) that **esmprep** can be used for. See also figure 1.

	Daily ESM questionnaires prompted		
ESM versions	All	None	Some
1	C1	C2	C3
> 1	C4	C5	C6

Throughout this document I shall refer to table 1 when explaining which function makes sense and how exactly it needs to be used, depending on the specific combination C1–C6 in your research project. C1 and C4 are probably the most frequently applied cases.

3 Install and load 'esmprep'

After having installed R, open R with the IDE[2] of your choice, e.g. RStudio. With an R IDE open, first install the 2 packages lubridate and **esmprep** by entering into the console:

```
install.packages(c("lubridate", "esmprep")) # Install only once, not every time!
```

From now on use the **esmprep** package just by entering at the beginning of each session:

```
library(esmprep) # Load the package every time.
```

The package lubridate, from which a few functions are used, doesn't need to be loaded separately.

[2]IDE: integrated development environment.

For additional help and the most up to date version[3] of **esmprep** go to its GitHub page.

```
# In case you DON'T already have the package 'devtools' installed:
install.packages("devtools", dependencies=TRUE)
library(devtools)                    # Load the package devtools
install_github("mmiche/esmprep")     # Install esmprep via GitHub
```

4 Overview of the function hierarchy within the 'esmprep' package

Table 2: All functions within the **esmprep** package in hierarchical order.

No.	C1–C3	Function name	No.	C1–C3	Function name
01a		relevantREFVN	12a		findChars
01b		**setREF**	**12b**		**convertChars**
02a		relevantESVN	**13**		**esAssign**
02b		**setES**	**14**		**missingEndDateTime**
03		esList	**15**		**esIdentical**
04		**genKey**	16a		suggestShift
05		**genDateTime**	**16b**		**makeShift**
05		**splitDateTime**	16c		printSuggestedShift
06		refPlausible	**17**	–	**expectedPromptIndex**
07a		**rmInvalid**	**18**	–	**intolerable**
07b		printRmInvalid	**19**		**randomMultSelection**
08	–	esItems	**20**		**computeTimeLag**
09	–	**esPlausible**	**21**		**computeDuration**
10		**esComplete**	**22**		**computeTimeBetween**
11	–	esMerge	**23**		**esFinal**

C1–C3 = –: Function for C1–C3 in table 1 is redundant or impossible. In **bold font**: Main functions. In normal font: Supporting functions. In red font: Error handling functions. Letters a–c behind numbers: Functions related to one another.

5 Setting things up

With the **esmprep** package 7 exemplary datasets are loaded automatically: referenceDf, morningControl, dayControl, eveningControl, morningTest, dayTest, and eveningTest. To find out about these datasets in the R console enter ?referenceDf, ?morningControl, etc.

[3] R policy recommend to update a package on CRAN no more than every 1–2 months.

5.1 Loading the relevant datasets in R

Display the exemplary reference dataset in R like this:

```
referenceDf
```

The first line of the exemplary reference dataset (referenceDf) can be seen in table 3.

Table 3: Example of the first participant in the exemplary reference dataset.

id	imei	start_date	start_time	st1	st2	st3	st4	end_date	end_time
P001	526745224596318	2007-07-29	21:10:00	09:10:00	13:10:00	17:10:00	21:10:00	2007-08-05	17:10:00

The exemplary raw ESM datasets can be displayed in the same way, e.g.

```
morningControl
```

In our exemplary ESM project there are 6 raw ESM datasets (called ESM versions), i.e. 3 versions (morning, day, and evening) for each of 2 different study groups (test group vs control group).

5.2 Most relevant variable names

Before being able to make use of the main purpose of the **esmprep** package, the most relevant dataset information must be passed to the package, according to your ESM study variant. See C1–C6 in table 1.

5.2.1 relevantREFVN (Function 1 of 29)

There **must** be a reference (REF) dataset which must include variables that are relevant for subsequent functions. VN stands for variable names.
The variable names' meaning in table 3 are:

1. **id**: The unique(!) identification number of the participant.
2. **imei**: The IMEI number (also called MEID), which is the unique identification number (15 digits) of the mobile device[4].
3. **start_date**: The date at which the ESM period of the participant started.
4. **start_time**: The prompt at the start date when the participant was supposed to fill out the first ESM questionnaire on his/her own.
5. **st**: The scheduled time(s) (prompt(s)) at which the participant is supposed to fill

[4]Entering *#06# on the dial keyboard of your smartphone should return its IMEI number.

out the daily ESM questionnaire(s). In the example there are 4 daily prompts.

6. **end_date**: The date at which the ESM period of the participant ended.

7. **end_time**: The prompt at the end date when the participant was supposed to fill out the last ESM questionnaire on his/her own.

The function `relevantREFVN` expects the above 7 arguments in the above order(!), i.e. the 7 most relevant variable names of your reference dataset. Your reference dataset of course might have more variable names (e.g. age, sex) but they are irrelevant for the **esmprep** package's purpose.
Arguments that have to be passed to the function `relevantREFVN`, for the exemplary reference dataset in table 3. `relevantREFVN` example 1:

```
relRef <- relevantREFVN("id", "imei", "st", "start_date", "start_time",
                        "end_date", "end_time")
```

If your reference dataset contains date-time objects, i.e. date and time combined in one variable, for the ESM start and end, respectively, please pass the arguments like in `relevantREFVN` example 2:

```
relRef <- relevantREFVN("id", "imei", "st",
                        START_DATETIME = "start_dateTime",
                        END_DATETIME = "end_dateTime")
```

In the `relevantREFVN` example 2 I assumed your variable names to be `start_dateTime` and `end_dateTime`. Of course they could be any character string.
The result of the function `relevantREFVN` has been assigned to the variable `relRef`, which can be displayed:

```
> relRef
$REF_ID
[1] "id"
$REF_IMEI
[1] "imei"
$REF_ST
[1] "st"
$REF_START_DATE
[1] "start_date"
$REF_START_TIME
[1] "start_time"
$REF_END_DATE
[1] "end_date"
$REF_END_TIME
```

```
[1] "end_time"
$REF_DATETIMES_SEP
[1] TRUE
```

The last element `REF_DATETIMES_SEP` is relevant only for subsequent functions, which ask if it is `TRUE`, i.e. whether date and time are separated variables in the reference dataset, or if it is `FALSE`, i.e. whether date and time is combined in one variable.

5.2.2 setREF (Function 2 of 29)

The function `setREF` sets up all that **esmprep** requires concerning the reference dataset. `setREF` expects 2 arguments: The first argument is the number of daily prompts (in the example in table 3 there are 4 daily prompts). The second argument is the result from the function `relevantREFVN`.

```
RELEVANTVN_REF <- setREF(4, relRef)
```

`setREF` returns a list to the user. Subsequent functions will use this list, therefore if this list is not created or if the name is deleted from the R workspace, subsequent functions can't (co-)operate. Recommended name for that list: `RELEVANTVN_REF`.

5.2.3 relevantESVN (Function 3 of 29)

`relevantESVN` has the same purpose as the function `relevantREFVN` (5.2.1), i.e. to pass the most relevant variable names, only this time of the raw ESM dataset(s). The meaning as well as the order of the arguments expected by `relevantESVN` is:

1. `svyName`: The column of the ESM dataset that holds the specific ESM questionnaire version as a character string. This argument is mandatory for combinations C4–C6 in table 1, it is optional for combinations C1–C3, in which case pass `NULL`.

2. `IMEI`: The column of the ESM dataset that holds the IMEI (MEID) numbers of the mobile devices that have been used by the research participants.

3. `STARTDATE`: The column of the ESM dataset that holds the date at which each ESM questionnaire was started.

4. `STARTTIME`: The column of the ESM dataset that holds the time at which each ESM questionnaire was started.

5. `ENDDATE`: The column of the ESM dataset that holds the date at which each ESM questionnaire was ended.

6. `ENDTIME`: The column of the ESM dataset that holds the time at which each ESM questionnaire was ended.

The result of the function `relevantESVN` is assigned to the variable `relEs`:

```
relEs <- relevantESVN("survey_name", "IMEI", "start_date", "start_time",
                      "end_date", "end_time")
```

The result of `relevantESVN` can be displayed:

```
$ES_SVY_NAME
[1] "survey_name"
$ES_IMEI
[1] "IMEI"
$ES_START_DATE
[1] "start_date"
$ES_START_TIME
[1] "start_time"
$ES_END_DATE
[1] "end_date"
$ES_END_TIME
[1] "end_time"
$ES_DATETIMES_SEP
[1] TRUE
```

The last element `ES_DATETIMES_SEP` fulfills the same purpose as does `REF_DATETIMES_SEP` in function `relevantREFVN` (5.2.1).

5.2.4 setES (Function 4 of 29)

The function `setES` sets up all that **esmprep** requires concerning the ESM dataset(s). `setES` expects 4 arguments, of which all 4 are mandatory for all combinations C1–C6 in table 1. The first argument is the number of daily prompts (in the example in table 3 there are 4 daily prompts). The second argument is a vector of all IMEI numbers used in the ESM project. The IMEI numbers, however, must not be numeric, instead they must be character strings, like this:

```
imeiNumbers <- c("526745224593822", "526745224599058", ..., "526745224592943")
```

Tip: Since there must be a column in the reference dataset, which holds the IMEI number of each participant, you don't have to retype all IMEI numbers. Instead, use the already existing column in the reference dataset:

```
imeiNumbers <- unique(as.character(referenceDf[,"imei"]))
```

The third argument must be a vector of character strings, specifying all the ESM questionnaire versions used in the study:

```
surveyNames <- c(
"morningTestGroup", "dayTestGroup", "eveningTestGroup",
"morningControlGroup", "dayControlGroup", "eveningControlGroup")
```

The last argument is the result from the function `relevantESVN`, which in our examplary ESM data looks like this:

```
RELEVANT_ES <- setES(4, imeiNumbers, surveyNames, relEs)
RELEVANTINFO_ES <- RELEVANT_ES[["RELEVANTINFO_ES"]]
RELEVANTVN_ES <- RELEVANT_ES[["RELEVANTVN_ES"]]
```

`setES` returns 2 lists to the user. Subsequent functions will use both lists, therefore if they are not created or if the names are deleted from the R workspace, subsequent functions can't (co-)operate. Recommended names for these lists: `RELEVANTINFO_ES` and `RELEVANTVN_ES`.

5.3 esList (Function 5 of 29)

`esList`: Use for C1–C6 in table 1.
`esList` includes all raw ESM datasets in one single data structure, in order to keep things together which will end up together. Our 6 exemplary raw ESM datasets, of which each one must be of class data.frame[5]:

```
esLs <- esList(list(morningControl, dayControl, eveningControl, morningTest, dayTest,
       eveningTest), RELEVANTVN_ES)
```

The second argument RELEVANTVN_ES is one of the results of function `setES` 5.2.4.

5.4 genKey (Function 6 of 29)

`genKey`: Use for C1–C6 in table 1.
In order for the user to be in control of all the raw ESM data at any time, a unique number is assigned to each line of raw ESM data.

```
keyLs <- genKey(esLs)
```

The result of `genKey` is assigned to the variable `keyLs`. An excerpt of one of the raw ESM datasets with the new first column 'KEY' looks like this:

[5] When using the function `read.table` as shown in section 5.1 the loaded dataset is of class data.frame.

Table 4: Excerpt of a raw ESM dataset with the new first column 'KEY'.

KEY	survey_name	IMEI	start_date	start_time	end_date	end_time
1016	dayControlGroup	526745224596286	2007-07-30	21:00:28	2007-07-30	21:05:09
1017	dayControlGroup	526745224596286	2007-08-01	21:20:34	2007-08-01	21:29:57

5.5 genDateTime or splitDateTime (Function 7 of 29)

`genDateTime` and/or `splitDateTime`: Use for C1–C6 in table 1.
Some subsequent functions require the date and time information (e.g the time stamps of each ESM questionnaire) to be separate from one another. Other subsequent functions require date-time objects, i.e. the date and time information combined in one variable. The date and time information in both the reference dataset and the raw ESM datasets must contain either the separated or the combined form. Therefore, if date and time information is separated the user must apply the function `genDateTime` (short for 'generate date-time'):

```
referenceDfList <- genDateTime(referenceDf, "REF", RELEVANTINFO_ES,
RELEVANTVN_ES, RELEVANTVN_REF)
```

The first argument is either a single dataset (e.g. the reference dataset) or a list list of datasets (e.g. the raw ESM datasets). The second argument is a character string specifying whether it is the reference dataset or the (list of) raw ESM dataset(s). The other arguments RELEVANTINFO_ES, RELEVANTVN_ES, and RELEVANTVN_REF are the results either of function `setES` or of function `setREF`. These latter 3 data objects are modified once, right here, by the function `genDateTime`. Therefore, apart from the output dataset, they also need to be extracted from the result:

```
# Extract reference dataset from output
referenceDfNew <- referenceDfList[["refOrEsDf"]]
# Extract extended list of relevant variables names of reference dataset
RELEVANTVN_REF <- referenceDfList[["extendedVNList"]]
```

The same procedure for the raw ESM dataset(s):

```
keyList <- genDateTime(keyLs, "ES", RELEVANTINFO_ES,
RELEVANTVN_ES, RELEVANTVN_REF)
# Extract list of raw ESM datasets from output
keyLsNew <- keyList[["refOrEsDf"]]
# Extract extended list of relevant variables names of raw ESM datasets
RELEVANTVN_ES <- keyList[["extendedVNList"]]
```

If, on the other hand, the date and time information already exists in combined form the

user must apply the function `splitDateTime`:

```
referenceDfList <- splitDateTime(referenceDf, "REF", RELEVANTINFO_ES,
RELEVANTVN_ES, RELEVANTVN_REF)
```

The same procedure for the raw ESM dataset(s):

```
keyList <- splitDateTime(keyLs, "ES", RELEVANTINFO_ES,
RELEVANTVN_ES, RELEVANTVN_REF)
```

The resulting dataset will contain new variables. For instance, after `genDateTime` has been applied to the reference dataset the new variables are REF_START_DATETIME and REF_END_DATETIME.

For both functions `genDateTime` and `splitDateTime` there are 2 further functions that might be helpful for the user: `dateTimeFormats` and `dateTimeFormats2`. The purpose of both functions is to give some orientation if needed. Just enter the function like this:

```
dateTimeFormats()
```

It will print to the console:

```
  dateTimeFormat     examplaryInput                    output
1        ymd_HMS 2017-02-06 07:11:23  2017-06-02 07:11:23 UTC
2        mdy_HMS   02-06-17 07:11:23  2017-06-02 07:11:23 UTC
3        dmy_HMS   06-02-17 07:11:23  2017-06-02 07:11:23 UTC
4         ymd_HM      17-02-06 07:11  2017-06-02 07:11:00 UTC
5         mdy_HM    02-06-2017 07:11  2017-06-02 07:11:00 UTC
6         dmy_HM    06-02-2017 07:11  2017-06-02 07:11:00 UTC
```

Whereas `dateTimeFormats2()` will print to the console:

```
[1] "ymd_HMS" "mdy_HMS" "dmy_HMS" "ymd_HM" "mdy_HM" "dmy_HM"
```

The meaning of it is: ymd = year month day. hms = hours minutes seconds.

5.6 refPlausible (Function 8 of 29)

`refPlausible`: Use for C1–C6 in table 1.

Error: The reference dataset contains all relevant information, according to which subsequent functions (especially function esAssign) can clearly identify which ESM questionnaires were answered by any single participant. One of the things that can go wrong is to enter the start of the ESM period for a participant after its end, which is impossible for non-time-travelers. Another possibility is that the ESM period (as set by the user)

is shorter or longer than it actually was. Usually, in an ESM study the ESM period is aimed to be equal across all participants. With the function `refPlausible` the user can check whether the ESM period is plausible across all participants, e.g. that there are no negative periods and that the periods are relatively equal across participants. Several other potential errors are checked also, e.g. if all participant IDs are unique. In case of any duplicated IDs the user is required to correct this. Another potential error is that the overall start and/or end time of a participant's ESM study period is not part of his/her prompts. If this error occurs, the user is required to correct this. `refPlausible` assists in the correction procedure by printing the erroneous parts of the reference dataset to the console.

```
refDfCheck <- refPlausible(referenceDfNew, units="days", RELEVANTVN_REF=RELEVANTVN_REF)
```

The first argument is the reference dataset as returned (as part of a list) by the function genDateTime (or by the opposite function splitDateTime), after being applied to the initial reference dataset. The second argument has `days` as default (further options are: auto, secs, mins, hours, and weeks, as explained in the documentation of the R-base function difftime; enter `?difftime` in the R console). The third argument RELEVANTVN_REF also is part of the list that is returned by genDateTime or splitDateTime.
The result of `refPlausible` is the reference dataset that was passed to the function, with at least one additional column. If the ESM period was selected to be computed as days, the new column's name will be `ESM_PERIODDAYS`, if mins (minutes) was selected, the name will be `ESM_PERIODMINS` and so on. If the increasing order of the prompts per participant is interrupted (only in case there are ≥ 2 prompts) two additional columns are added, which refer to a possible time anomaly. For example, if prompt 1 and prompt 2 were 9 a.m. and 8 a.m., respectively, it is an error and must be corrected. However, if prompts 3 and 4 were 10 p.m. and 1 a.m., respectively, this might be correct, in which case a single ESM day crossed midnight, which is why the user is supposed to pass `TRUE` to the argument `midnightPrompt` when using the function esAssign. `refPlausible` suggests this to the user whenever a possible time anomaly occurs.

6 ESM preparatory functions

Each function will be introduced by a very short explanation of its purpose, e.g. what kind of error it handles.

6.1 rmInvalid (Function 9 of 29)

`rmInvalid`: Use for C1–C6 in table 1.
Error: A line of raw ESM data does not contain any data whatsoever. Instead there is a warning message of some sort[6]. Instead of searching for the word 'Warning' `rmInvalid` registers a line of data to be invalid if there exists neither a start date nor a start time for a questionnaire:

```
rmInvLs <- rmInvalid(keyLsNew, RELEVANTVN_ES)
```

The second argument RELEVANTVN_ES is one of the results either of function `genDateTime` or of function `splitDateTime`.
`rmInvalid` displays to the console some rudimentary information about whether and how many lines of data were removed due to being invalid. The result of `rmInvalid`, here assigned to the variable **rmInvLs**, is a named list with 4 elements. The elements' names and content are:

1. **dfValid**: Same content as in the result of esList, except that lines of data were removed when registered to be invalid.

2. **listInvalid**: A list with all removed lines of data, which consists only of missing values and/or a warning message.

3. **rmInvalidDfFinished**: A logical value. `TRUE` if all datasets within the list of raw ESM datasets where searched for invalid data, else `FALSE`.

4. **noLinesRemovedAtAll**: A vector of logical values. `TRUE` whenever at least one line of data was registered to be invalid, else `FALSE`.

Relevant to the user is the first element only, **dfValid**, which is used in the subsequent function `esPlausible` (6.4). However, if the invalid data shall be displayed in the console, the user can do this by applying the function `printRmInvalid` (6.2).

[6] An app developer might invoke a particular warning message whenever the app crashes.

6.2 printRmInvalid (Function 10 of 29)

`printRmInvalid`: Use for C1–C6 in table 1. Yet only if `rmInvalid` has been applied before.

Error: None. `printRmInvalid` prints to the console what the user wants to be displayed, concerning the questionnaires that have been removed by `rmInvalid`. The first argument is the result of the function `rmInvalid`. The last argument `smr` (short for summarize) can be `"tabulate"` (which is the default), it can be `"detail"`, and it can be `"both"`. In response to `"tabulate"` `printRmInvalid` displays a table, showing the number of removed questionnaires for each ESM version. The argument `"detail"` displays the removed questionnaires in full detail. The argument `"both"` displays both the table and the removed questionnaires in detail.

```
key_rmLs <- printRmInvalid(rmInvLs, RELEVANTVN_ES, smr="tabulate")
```

Its result for our exemplary datasets looks like this:

```
Summarizing table:
            versions notRemoved removed  Sum
1 morningControlGroup        14       0   14
2     dayControlGroup        89       1   90
3 eveningControlGroup        27       1   28
4    morningTestGroup        20       0   20
5        dayTestGroup        65       1   66
6    eveningTestGroup        13       1   14
7                Sum       228       4  232
```

Also, `printRmInvalid` returns a list to the user with as many elements as there are ESM versions. For each of the ESM versions, the value of the variable **KEY** (see function `genKey` (5.4)) is contained for all of the removed questionnaires. In case that there was an incorrect removal, the user will easily be able to reinsert the data.

6.3 esItems (Function 11 of 29)

`esItems`: Use for C4–C6 in table 1; optional, though redundant, for C1–C3.

Error: None. `esItems` is a preparatory function for the following function `esPlausible` (6.4), by extracting only the necessary information for it: The ESM questionnaire items:

```
plausibItems <- esItems(dfList=rmInvLs[["dfValid"]], RELEVANTVN_ES)
```

In R you access the content of a list with double square brackets and the name of the accessed element in quotes; optionally instead of the quoted name you can use the index number of the element, in this case it would be the first element (`rmInvLs[[1]]`).

6.4 esPlausible (Function 12 of 29)

`esPlausible`: Use for C4–C6 in table 1; impossible for C1–C3.

Errors: For instance, variable names in at least 2 different ESM questionnaire versions are identical, although their content differs; the range of item values for items with identical names differ from one another; an item contains no values at all; etc.

Optimally `esPlausible` is applied in the pilot phase of the ESM project, when it makes most sense to have all combinations of variable names and their corresponding item content to be correct. This should be written down (e.g. paper, Excel, etc.) and be up to date! However, at the latest, `esPlausible` should be applied before searching any other errors in the raw ESM datasets. If this is not checked more errors will be generated!

```
plausibLs <- esPlausible(dfList=rmInvLs[["dfValid"]], itemVecList=plausibItems)
```

`esPlausible` returns a list with 4 elements:

1. **plausibNames**: Overview of the variable names as specified by the user.

   Table 5: Excerpt of list element **plausibNames**.

namesAll	names1	names2	names3	names4	names5	names6	namesCheck
V1	V1	V1	V1	V1	V1	V1	TRUE
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
V2_1	V2_1	V2_1	V2_1	V2_1	V2_1	V2_1	TRUE
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
V8			V8		V8	V8	TRUE

   Note. Variable V8 only in versions 3, 5, and 6.

2. **plausibClass**: Overview of the variable classes as registered by R.

   Table 6: Excerpt of list element **plausibClass**.

namesAll	class1	class2	class3	class4	class5	class6	classCheck
V1	num	num	num	num	num	num	TRUE
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
V2_1	log	int	int	int	int	int	FALSE
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
V8			cha		cha	cha	TRUE

   Note. Inconsistent classes in variable V2_1.

3. **plausibRowNa**: Overview of number of datalines and percentage of missing values.

Table 7: Excerpt of list element **plausibRowNa**.

namesAll	nRows1	naPercent1	...	nRows5	naPercent5	nRows6	naPercent6
V1	14	0	...	65	0.0	13	0.0
⋮	⋮	⋮	...	⋮	⋮	⋮	⋮
V2_1	14	100	...	65	3.1	13	7.7
⋮	⋮	⋮	...	⋮	⋮	⋮	⋮
V8			...	65	0.0	13	0.0

Note. In variable V2_1 (version 1) all values (100%) are missing. The values TRUE, FALSE, and NA have the class logical. nRow specifies the number of questionnaires.

4. **plausibMinMax**: Overview of minimum and maximum numeric values.

Table 8: Excerpt of list element **plausibMinMax**.

namesAll	min1	max1	min2	max2	min3	max3	min4	max4	min5	max5	min6	max6
V1	0	3	−1	3	0	3	0	3	0	3	0	3
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
V2_1			0	98	0	99	1	97	0	100	12	98
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
V8												

Note. V1 contains a negative value (version 2), V8 contains contains text (no min, max possible).

Note that `esPlausible` can merely give hints about errors. Only someone with knowledge about the items' content and about the expected values – i.e. their range – in the raw ESM dataset can find out whether implausibilities point to actual errors.
In our exemplary ESM dataset some of the implausibilities have turned out to be errors as described at the beginning of subsection 6.4. One hint can be seen in table 6.
It turned out that V2_1 in version 1 contains no values, i.e. all values are NA. Since `class(NA)` returns the class 'logical', the variable V2_1 correctly is of that class.
If, however, the plausibility check turned out that this variable contained values, TRUE and FALSE in the case of the class logical, then the variable name V2_1 was erroneous, because the other variables of the same name contain integers, i.e. numbers. In such a hypothetical case, when there are identical variable names for different items, prior to any proceedings a new variable name must be assigned, e.g.

```
# Change column name in version 1 at index 6 (which so far has been V2_1)
colnames(rmInvLs[["dfValid"]][[1]])[6] <- "V2_2"
```

During the pilot phase of the ESM project the app developer should change the relevant variable name(s). However, if the ESM project was over, somebody else would have to change the variable name(s) accordingly[7].

[7] Choose some variable name that **isn't** in **any** of the ESM dataset(s).

Another error turned up only by inspecting **plausibMinMax** (see table 8). There was a negative value in version 2 of item V1.
After all errors were found and corrected, again apply `esItems` (6.3) and `esPlausible` (6.4), in order to see whether all corrected errors don't show up any more.
Note in particular that inspecting all 4 list elements returned by `esPlausib` is important, not just inspecting 1 or 2 elements.

6.5 esComplete (Function 13 of 29)

`esComplete`: Use for C1–C6 in table 1.
Error: An ESM questionnaire has neither an end date nor an end time, nonetheless the questionnaire might be complete, i.e. the last item of the questionnaire that is expected to contain a value in order to be considered complete, acually contains a value. `esComplete` expects 2 arguments, the first of which is the current state of the raw ESM dataset(s). The second argument `lastItemList` might seem difficult at first glance. It isn't.

```
lastItemList <- list( list("morningTestGroup", "V6", 0, "V6_1"),
                      list("dayTestGroup", NA, NA, "V7"),
                      list("eveningTestGroup", "V9", 1, "V9_1"),
                      list("morningControlGroup", "V6", 0, "V6_1"),
                      list("dayControlGroup", NA, NA, "V7"),
                      list("eveningControlGroup", "V8_1", 1:5, "V8_3"),
                      list("eveningControlGroup", "V8_1", 0, "V8_2") )
```

The second argument in `esComplete` will be the variable **lastItemList**. It is a list of lists, i.e. (inner) lists nested within one (outer) list. The upper example is undoubtably a complex one. However, once understood, all simpler cases can then be applied very easily. Let's translate the principle:
Example 1:

```
                      list("morningTestGroup", "V6", 0, "V6_1"),
```

In the ESM version `morningTestGroup`, if the variable `V6` is 0, then variable `V6_1` is the last item that is expected to contain a value, otherwise `V6` is the last item that is expected to contain a value.

What if there is no such condition? That makes things much easier!
Example 2:

```
                      list("dayTestGroup", NA, "V7", NA),
```

In the ESM version `dayTestGroup` the variable `V7` is the last item that is expected to

contain a value. Because there is no condition, the user must set the second and third list element to NA.
But what if there are 2 or even more conditions? That makes things a bit more difficult.
Example 3:

```
                    list("eveningControlGroup", "V8_1", 1:5, "V8_3"),
                    list("eveningControlGroup", "V8_1", 0, "V8_2")
```

Since the ESM version `eveningControlGroup` has two conditions, just repeat the same procedure for both conditions, as learned in **example 1**:
If variable `V8_1` is between 1 and 5, then variable `V8_3` is expected to be the last variable to contain a value.
If `V8_1` is 0, then variable `V8_2` is expected to be the last variable to contain a value.

After the second argument `lastItemList` has been generated, apply `esComplete`[8]:

```
isCompleteLs <- esComplete(rmInvLs[["dfValid"]], lastItemList)
```

The result is assigned to the variable **isCompleteLs**. It is a named list with 7 elements, because the procedure for the last ESM version was applied twice, whereas for the remaining 5 ESM versions it was applied just once. This is also represented in the names of the resulting list:

```
names(isCompleteLs)
[1]"morningTestGroup_1"  "dayTestGroup_1"      "eveningTestGroup_1"
[4]"morningControlGroup_1" "dayControlGroup_1" "eveningControlGroup_1"
[7]"eveningControlGroup_2"
```

Each single raw ESM dataset contains the new column **INCOMPLETE_i**, where **i** represents the incrementing number of how often the procedure was applied within `esComplete`. For our exemplary ESM datasets the procedure was applied once for the versions 1–5, which is why the new column is **INCOMPLETE_1**. Only version 6 has the column **INCOMPLETE_1** and **INCOMPLETE_2**, because the procedure was applied twice. The column **INCOMPLETE_i** is dichotomous, representing a questionnaire to be complete by the value 0, incompleteness therefore by the value 1. See version 2 (`"dayTestGroup_1"`). Of 65 lines of data, 63 are complete, 2 are incomplete.

```
table(isCompleteLs[["dayTestGroup_1"]]["INCOMPLETE_1"])
 0  1
63  2
```

[8]The first argument is the result from function `rmInvalid` (6.1). More specifically, it is the list element named `"dfValid"`, since `rmInvalid` returns a list with 4 elements. Also, `esPlausible` (6.4) should have been applied as advised in this vignette, before applying `esComplete`!

If for every ESM version the procedure must be applied once (unlike **example 3**), then the new column's name will be **INCOMPLETE**, i.e. without an appended number.

Table 9: Incomplete ESM questionnaires in version **dayTestGroup_1**.

KEY	V7	survey_name	end_date	end_time	INCOMPLETE_1
1165		dayTestGroup			1
1166		dayTestGroup			1

Note. Neither end date nor end time and no value in variable V7.

6.6 esMerge (Function 14 of 29)

`esMerge`: Use for C4–C6 in table 1, impossible for C1–C3.
Error: None. This function merges all raw ESM datasets into one single dataset:

```
esMerged <- esMerge(isCompleteLs, RELEVANTVN_ES)
# If preferred convert the 15 digit IMEI number from scientfic notation to text.
esMerged[,RELEVANTVN_ES[["ES_IMEI"]]] <-
as.character(esMerged[,RELEVANTVN_ES[["ES_IMEI"]]])
```

After merging the raw ESM datasets, lets take a look at an excerpt of the structure of our exemplary ESM dataset:

```
'data.frame': 228 obs. of 32 variables:
 $ KEY             : int 1015 1153 1154 1016 1155 1156 1157 1158 1017 1159 ...
 $ V1              : num 1 0 1 3 1 0 2 3 2 1 ...
 $ V1_1            : int 1 1 0 1 1 0 0 0 1 0 ...
 $ survey_name     : chr "dayControlGroup" "dayTestGroup" "dayTestGroup" ...
 $ IMEI            : chr "526745224596286" "526745224596286" "526745224596286" ...
 $ ES_START_DATETIME: POSIXct, format: "2007-07-29␣21:00:28" "2007-07-30␣13:01:46" ...
 $ ES_END_DATETIME : POSIXct, format: "2007-07-29␣21:07:52" "2007-07-30␣13:06:40" ...
 $ INCOMPLETE_1    : num 0 0 0 0 0 0 0 0 0 0 ...
 $ INCOMPLETE_2    : num NA NA NA NA NA NA NA NA NA NA ...
```

6.7 findChars (Function 15 of 29)

`findChars`: Use for C1–C6 in table 1.
Error: None. `findChars` is a preparatory function for the following function `convertChars` (6.8), in that it picks the variables within the (merged) raw ESM dataset which contain text:

```
findTextIdx <- findChars(esMerged)
```

The result (as printed by `findChars` to the console) might look like this[9]:

```
findTextIdx <- findChars(esMerged)
'data.frame': 228 obs. of 10 variables:
 $ V1_2      : chr "erklärt" "Börse" "erklärt" "Schloß" ...
 $ V3_1      : chr "Jahr" "Börse" "entrée" "Übung" ...
 $ survey_name: chr "dayControlGroup" "dayTestGroup" "dayTestGroup" ...
 $ IMEI      : chr "526745224596286" "526745224596286" "526745224596286" ...
 $ start_date : chr "2007-07-29" "2007-07-30" "2007-07-30" "2007-07-30" ...
 $ start_time : chr "21:00:28" "13:01:46" "17:00:11" "21:00:28" ...
 $ end_date  : chr "2007-07-29" "2007-07-30" "2007-07-30" "2007-07-30" ...
 $ end_time  : chr "21:07:52" "13:06:40" "17:08:19" "21:05:09" ...
 $ V8        : chr NA "" "" NA ...
 $ V9_1      : chr NA NA NA NA ...
```

In the exemplary ESM datasets delivered with the **esmprep** package there are some meaningless words, representing participants' text answers. Some of these words might contain letters like ä, ü, or ß – e.g. in German text. In other countries these might be other words like 'entrée'.

`findChars` searches for all variables in the dataset which are of the class character. However, the user must specify which of these variables contain letters that need to be converted. In our exemplary ESM dataset the variables `V1_2`, `V3_1`, `V8`, and `V9_1` might need conversion. `findChars` returns the indices of the variables containing text:

```
findTextIdx    # Overall 10 variables are identified as belonging to class character
      V1_2       V3_1 survey_name      IMEI  start_date start_time   end_date
         3         11          20        21          22         23         24
  end_time         V8       V9_1
        25         30         41
```

The user must pick from them (`V1_2` = 1 in `c(1,2,9,10)`, `V3_1` = 2, etc.):

```
findTextIdx1 <- findTextIdx[c(1,2,9,10)]
```

Elements 1, 2, 9, and 10 within the variable **findTextIdx** are the target variables within the ESM dataset to pick. Notice the new assignment to variable **findTextIdx 1**:

```
names(findTextIdx1)   # findTextIdx contained 10 values, findTextIdx1 contains 4 values
[1] "V1_2" "V3_1" "V8" "V9_1"
```

Next, before applying the function `convertChars` we must generate one of its arguments. This argument must be a data.frame. The first column of it includes the old (i.e. to be converted) letters, the second column of it includes the new letters:

[9]Since R doesn't permit umlauts in a package, there are no umlauts in the exemplary ESM datasets.

Table 10: On = before conversion, Off = after conversion.

On	Off
ä	ae
ß	ss
é	e
è	e
Ä	Ae

6.8 convertChars (Function 16 of 29)

`convertChars`: Use for C1–C6 in table 1.

Error: Depending on the encoding system somebody uses when reading data files, some letters can cause trouble. Sometimes the data file is not even imported, instead the user might receive an error message about so-called escape sequences. In order to avoid any trouble that might be caused by encoding systems, troublesome letters as specified by the user might be converted to letters that don't cause trouble:

```
esMerged1 <- convertChars(esMerged, textColumns, convertCharsDf)
```

The first argument in `convertChars` is the current state of the ESM dataset. The second argument are the column names of the variables that contain text within which there might be letters that need to be conversed. The second argument can easily be obtained by using the result of `findChars` (compare subsection 6.7):

```
textColumns <- names(esMerged)[findTextIdx1]
```

The third argument is the conversion data.frame, an example of which is table 10. There is a fourth argument which is optional: `ignoreCase` (default = `FALSE`). If upper/lower case can be ignored when converting, the user has to set it to `TRUE`:

```
esMerged1 <- convertChars(esMerged, textColumns, convertCharsDf, TRUE)
```

After the conversion the result in our exemplary ESM dataset looks like this:

```
str(esMerged1[,findTextIdx1])
'data.frame':  228 obs. of 4 variables:
 $ V1_2      : chr "erklaert" "Boerse" "erklaert" "Schloss" ...
 $ V3_1      : chr "Jahr" "Boerse" "entree" "Uebung" ...
 $ V8        : chr NA "" "" NA ...
 $ V9_1      : chr NA NA NA NA ...
```

7 ESM questionnaire assignment and error handling functions

7.1 esAssign (Function 17 of 29)

`esAssign`: Use for C1–C6 in table 1.
Error: None. `esAssign` can be considered to be the heart of the **esmprep** package. `esAssign` assigns the lines of data within the raw ESM dataset to the participants which are listed in the reference dataset (see subsection 5.1, as well as table 3):

```
esAssigned <- esAssign(esDf = esMerged1, refDf = referenceDfNew, RELEVANTINFO_ES,
RELEVANTVN_ES, RELEVANTVN_REF)
```

The first argument in `esAssign` is the ESM dataset in its current state. The second argument is the reference dataset.
`esAssign` comes with 5 optional arguments:

- **singleParticipant**: If the user wishes to (temporarily) have the ESM data to be assigned to one specific participant only, then the participant's ID – as specified in the reference dataset – must be passed to the argument **singleParticipant**. Per default the ESM data is assigned to all participants listed in the reference dataset.
- **prompted**: If none of the ESM questionnaires were prompted, i.e. all ESM questionnaires were event contingent (see ESM combination C2 and C5 in table 1), then the user must pass `FALSE` to the argument **prompted**.
- **promptTimeframe**: If in order to be valid, the start of a single questionnaire must fall within a specified time frame around each prompt, e.g. 30 minutes, then the user must pass that time frame in minutes to the argument **promptTimeframe**.
- **midnightPrompt**: If the survey app enabled or prompted the participant to start a questionnaire after midnight, then the user must pass `TRUE` to the argument **midnightPrompt**. Per default it is set to `FALSE`.
- **dstDates**: If the user wishes to check whether the time frame around each single prompt was influenced by the daylight saving time (1 hour shift at the last weekend in both March and October), then the user must pass the daylight saving date to the argument **dstDates**[10].

Additional details on why the arguments **promptedTimeframe** and **dstDates** contribute important information:

[10] **dstDates** can also be a vector of dates, in case the ESM project spans over an entire year or longer.

The argument **dstDates** makes sense in case that the time on the mobile device, used by the participants, is assumed not to be always correct. The argument **dstDates** checks if the ESM period of each single participant includes the **dstDates**. An example of why this can be important information is when the participant gets reimbursed for the ESM questionnaires that have been filled out within a certain time frame, say 30 minutes. If the mobile device of the participant does not have the correct time all the time and the **dstDates** is included in the individual's ESM period, then some questionnaires might not get reimbursed for being outside of the 30 minutes time frame. However, they might be inside the time frame, only the 1 hour shift might make them seem to be outside of it. Applying `esAssign` not only assigns the ESM data to the selected participants, it also prints information to the console:

```
P004
No. 4 of 8
Daylight saving time is contained in the ESM period.
Event sampling period - completion rates per prompt
    1   2   3   4
% 100 100 100 87.5
```

The printed information concerns the participant's ID, whether the daylight saving date (**dstDates**) is included in the participant's ESM period, and how many of each of the prompts have been answered by the participant (in percent) during the ESM period.
By applying the function to the ESM dataset the user receives a list with 3 elements:

1. **ES**: a data.frame. This dataset contains all the ESM questionnaires that were assigned to the participant(s), which are listed in the reference dataset. Also this dataset now contains additional columns:

    - **ID**: Unique identification code of each participant.
    - **CV_ES**: CV is short for count variable. It counts all the questionnaires that have been filled out by the participant during the ESM period. In incrementing order it starts at 1 and skips a number, whenever a questionnaire is missing.
    - **CV_ESDAY**: Count of single ESM days. In incrementing order it starts at 1 and skips a number, whenever all questionnaires of that day are missing.
    - **CV_ESWEEKDAY**: Count of weekday (Monday = 1, ..., Sunday = 7).
    - **PROMPT**: Correspondance of the actual start time of the questionnaire to its prompt (in our examplary dataset this ranges between 1 and 4).
    - **PROMPTEND**: Correspondance of the actual end time of the questionnaire to its prompt (in our examplary dataset this ranges between 1 and 4).

- **ES_MULT**: Dichotomous variable. The value 1 represents a questionnaire that has been filled out repeatedly at one specific prompt.
- **ES_MULT2**: Alternative representation of ES_MULT. The very first questionnaire at a prompt is represented by the value 1, the second questionnaire (i.e. the first repeatedly filled out q.) is represented by the value 2, etc.
- **ST**: Assigns the prompt (Scheduled Time, ST) to the actual start time of a questionnaire, by choosing the miminal time difference between all possible prompts (per participant) and the actual start time of the questionnaire.
- **STDATE**: Variable is returned only if argument midnightPrompt is set to TRUE. Possible values and meaning: -1 = scheduled start date is prior to actual start date; 0 = scheduled start date and actual start date are equal to one another; 1 = scheduled date is subsequent to actual start date.
- **TFRAME**: Dichotomous variable. 1 = a questionnaire within the time frame, as specified by the user (see argument `promptTimeframe`).
- **DST**: Dichotomous variable. 1 = a questionnaire's date is equal or greater than the daylight saving date.
- **QWST**: Dichotomous variable. 1 = a questionnaire is fully within the scheduled time, i.e. both the actual start and end time have been assigned to the same prompt.

2. **ESopt** a data.frame. This dataset contains the most relevant information for each participant in terms of comparing the actual ESM dataset (after `esAssign` has been applied) with the ESM dataset considered to be optimal (short: opt). The optimal dataset is defined as every participant having filled out all of the prompted questionnaires during the ESM period. The user usually wants to know how many questionnaires (in percent) were filled out, both individually and on average.

3. **ESout**: a data.frame. This dataset contains all the ESM questionnaires that were not assigned to the participant(s), which are listed in the reference dataset. One of the possible reasons ESM questionnaires were not assigned is that a participant's information has not yet been added to the reference dataset.

4. **ESrate**: a data.frame. This dataset contains the average completion rates per participant, both per prompt and across prompts (per participant).
   Note that **ESrate** must not necessarily contain the final completion rates. The rates might change according to whether and how many questionnaires get shifted to a neighboring prompt (see function `suggestShift` 7.4). The final function (see section 8.4) of **esmprep** returns the final completion rates.

7.2 missingEndDateTime (Function 18 of 29)

`missingEndDateTime`: Use for C1–C6 in table 1.
Error: Some of the ESM questionnaires' end date and end time are missing. `missingEndDateTime` finds out about this:

```
noEndDf <- missingEndDateTime(esAssigned[["ES"]], RELEVANTVN_ES)
```

`missingEndDateTime` expects the ESM dataset in its current state as its first argument. Applying the function to the ESM dataset adds 2 additional columns to it, named **NOENDDATE** and **NOENDTIME**. They are dichotomous, representing a missing end date and end time by the value 1, else 0.

7.3 esIdentical (Function 19 of 29)

`esIdentical`: Use for C1–C6 in table 1.
Error: Some ESM questionnaires, after having downloaded them, exist in duplicated form, i.e. one specific line of ESM data identically exists more than once in the raw ESM dataset. `esIdentical` finds out about this:

```
identDf <- esIdentical(noEndDf, RELEVANTVN_ES)
```

`esIdentical` expects the ESM dataset in its current state as its first argument. Applying the function to the ESM dataset adds 1 additional column to it, named **IDENT**. It is dichotomous, representing identical questionnaires by the value 1, unique questionnaires by the value 0.

7.4 suggestShift (Function 20 of 29)

`suggestShift`: Use for C1, C3, C4, and C6 in table 1.
Error: Some ESM questionnaires that were registered by `esAssign` to exist repeatedly at a certain prompt, might be shifted backward or forward by the user – due to some condition. In the following example this condition is: If between the scheduled time and the actual start time of a questionnaire 100 minutes or more have passed, that questionnaire is suggested to be shifted to a neighboring prompt:

```
sugShift <- suggestShift(identDf, 100, RELEVANTINFO_ES, RELEVANTVN_ES)
```

Something like this is printed to the console by `suggestShift`:

```
     ID  KEY    survey_name CV_ES CV_ESDAY ES_START_DATETIME    ST
159 P001 1161   dayTestGroup   19       6    2007-08-03 13:07:46 13:00:00
19  P001 1019 dayControlGroup 21       6    2007-08-03 19:17:31 21:00:00
20  P001 1020 dayControlGroup 21       6    2007-08-03 21:05:21 21:00:00

    PROMPT PROMPTEND ES_MULT SHIFT SHIFTKEY LAG_MINUTES
159      2         2       0     0       NA          NA
19       4         4       0     1     1019        -102
20       4         4       1     0       NA          NA
```

Let's take a look at the reference dataset for P001.

Table 11: Extraction of the reference dataset.

id	st1	st2	st3	st4
P001	09:00:00	13:00:00	17:00:00	21:00:00

Note. id = identification code, st = scheduled time

When comparing what was printed to the console with the relevant data in the reference dataset (see table 11) we notice that for P001 the assigned **PROMPT** that is suggested to be shifted (see column **SHIFT** = 1) belongs to a questionnaire that was started 2 hours and 17 minutes after the scheduled time of 17:00:00. The interval of 4 hours between 2 subsequent prompts mean that the questionnaire 'almost' made it to prompt 3 (it missed by 17 minutes and 32 seconds), therefore it might be justified to shift this particular questionnaire to prompt 3, since there is no questionnaire already assigned to it.
If at least one questionnaire is suggested to be shifted to a neighboring prompt, the output of `suggestShift` is a list with three elements:

1. **esDf**: The raw ESM dataset in its current state.
2. **suggestShiftDf**: Relevant excerpt of each questionnaire that is suggested to be shifted to a neighboring prompt.
3. **printShiftDf**: Relevant information (concerning indices within the current raw ESM dataset), that is going to be used by the function `makeShift` (7.6).

BEWARE: If no questionnaire shift is suggested, the output of `suggestShift` is a list with exactly two elements, first, the same data.frame that was passed to it as input, and second, a character string confirming that no shift was suggested.
Notice that `suggestShift` adds 2 columns to the current raw ESM dataset, named **SHIFT** and **SHIFTKEY**. If no questionnaire shift is suggested, then **SHIFT** will only contain 0, whereas **SHIFTKEY** will only contain NA.

7.5 printSuggestedShift (Function 21 of 29)

`printSuggestedShift`: Use only if `suggestShift` or `makeShift` has been applied before.
Error: None. `printSuggestedShift` prints to the console what is printed by the function `suggestShift`. Printing is the sole purpose of `printSuggestedShift`! This is useful to check if the shifting has been executed correctly:

```
printSuggestedShift(madeShift, RELEVANTVN_ES)
```

Printed to the console by `printSuggestedShift` after shift has been executed:

```
     ID  KEY    survey_name CV_ES CV_ESDAY ES_START_DATETIME    ST
159 P001 1161   dayTestGroup   19       6    2007-08-03 13:07:46 13:00:00
19  P001 1019 dayControlGroup 20       6    2007-08-03 19:17:31 17:00:00
20  P001 1020 dayControlGroup 21       6    2007-08-03 21:05:21 21:00:00

    PROMPT PROMPTEND ES_MULT SHIFT SHIFTKEY LAG_MINUTES
159      2         2       0     0       NA          NA
19       3         3       0     1     1019        -102
20       4         4       0     0       NA          NA
```

The prompt for P001, which before was 4 now is 3 (see columns **PROMPT** and **PROMPTEND**.

7.6 makeShift (Function 22 of 29)

`makeShift`: Use for C1, C3, C4, and C6 in table 1. Yet only if at least one questionnaire shall be shifted from one prompt to another.
Error: None. `makeShift` executes the shifting:

```
madeShift <- makeShift(sugShift, referenceDfNew, keyPromptDf,
RELEVANTINFO_ES, RELEVANTVN_REF)
```

The function `makeShift` expects as first argument the output of function `suggestShift` and as second argument the reference dataset. The third argument must be generated. To this end, part of the ouput of function `suggestShift` is of help. Select from the dataset **suggestShiftDf** the columns NEW_PROMPT and SHIFTKEY. If you don't want to shift all suggested questionnaires, select the rows of **suggestShiftDf**, that contain the questionnaires you want to shift.

```
keyPromptDf <- sugShift$suggestShiftDf[,c("NEW_PROMPT", "SHIFTKEY")]
```

7.7 expectedPromptIndex (Function 23 of 29)

`expectedPromptIndex`: Use for C4 and C6 in table 1, don't use for C1-C3 and C5.
Error: A questionnaire that belongs to a certain time of day (say morning) has been filled out at another time of day, when it **might** make no sense (say day or evening).
Before applying `expectedPromptIndex`, the second argument of the function must be generated, as in this example for our exemplary ESM dataset:

```
expIdxList <- list(   list("morningTestGroup", 1, 1),
                      list("dayTestGroup", c(2,3), 2),
                      list("eveningTestGroup", 4, 3),
                      list("morningControlGroup", 1, 1),
                      list("dayControlGroup", c(2,3), 2),
                      list("eveningControlGroup", 4, 3))
```

General explanation: The second argument is a list of lists (compare the second argument in subsection 6.5). In the inner lists, the first element is a character string, specifying the ESM version, which in our exemplary ESM study refers to the categories morning, day, and evening, each for a test and a control group. The second element in the inner lists either is a single numeric value or a vector of values, specifying the prompt(s) that belong to the ESM version which was specified as first element. The third element always is a single numeric value, specifying what value we expect the second element of the inner lists to be combined with.
Detailed explanation: For the morning version we expect prompt 1 (which belongs to the ESM version of 'morning') to be combined with the value 1, which I – as user – specify as the first category (morning category).

```
                      list("morningTestGroup", 1, 1),
```

For the day version we expect prompts 2 and 3 (which belong to the ESM version of 'day') to be combined with the value 2, which I – as user – specify as the second category (day category).

```
                      list("dayTestGroup", c(2,3), 2),
```

For the evening version we expect prompt 4 (which belongs to the ESM version of 'evening') to be combined with the value 3, which I – as user – specify as the third category (evening category).

```
                      list("eveningTestGroup", 4, 3),
```

`expectedPromptIndex` will then search for all prompts that diverge from the user specified expectation.

Note that the chosen categories 1, 2, and 3 for the third element of the inner lists is arbitrary, as `expectedPromptIndex` only checks whether there are any divergences from the specified combinations that the user expects. The user could also have chosen 45, 29, and 2 instead of 1, 2, and 3, or any other values, as long as they differ from one another. With the second argument now generated, we apply `expectedPromptIndex`:

```
expectedDf <- expectedPromptIndex(madeShift$esDf, expIdxList,
RELEVANTINFO_ES, RELEVANTVN_ES)
```

The first argument is the ESM dataset in its current state. The second argument is the list of lists we have just generated. An excerpt of the result of `expectedPromptIndex` for our exemplary ESM dataset looks like this:

Table 12: Excerpt of the result of `expectedPromptIndex`.

ID	survey_name	ES_START_DATETIME	PROMPT	PROMPTFALSE	EXPCATEGORY
P005	dayControlGroup	2007-11-06 09:06:56	1	1	2
P005	dayControlGroup	2007-11-06 13:03:06	2	0	2
P005	dayControlGroup	2007-11-06 17:14:19	3	0	2

Note. ID = identification code, EXPCATEGORY = expected category as specified by the user. The first questionnaire is unexpected: A day questionnaire (category 2) filled out in the morning (prompt 1).

Note that `expectedPromptIndex` adds 2 new columns to the ESM dataset in its current state, named **PROMPTFALSE** and **EXPCATEGORY**.
PROMPTFALSE is dichotomous. 1 = divergence from the user specified expectation, else 0. **EXPCATEGORY** displays the category that the user specified to be combined with that prompt, which belongs to the ESM version (compare the explanation for how the second argument was generated for `expectedPromptIndex`). The ESM version in our exemplary ESM dataset is represented by the variable **survey_name** (see table 12).

7.8 intolerable (Function 24 of 29)

`intolerable`: Use for C4 and C6 in table 1, don't use for C1-C3 and C5.
Error: A questionnaire that belongs to a certain time (say morning) has been filled out at another time (say evening), when it **certainly** makes no sense at all.
Before applying `intolerable`, the second argument of the function must be generated:

```
intoleranceDf <- data.frame(prompts=c(2:4, 1, 1), unexpected=c(rep(1,times=3), 2, 3))
```

The second argument for `intolerable` is a data.frame.

Table 13: Second argument for function `intolerable`.

prompt	unexpected
2	1
3	1
4	1
1	2
1	3

Note. Exactly 2 columns.

Regarding our exemplary ESM data, table 13 translates to:
Day questionnaires (prompts 2 and 3) are not tolerated with category 1 (morning). An evening questionnaire (prompt 4) is also not tolerated with category 1. A morning questionnaire (prompt 1) is not tolerated with category 2 (day), or with category 3 (evening). All other combinations therefore are tolerable and will thus not be removed from the ESM dataset. Apply function `intolerable`:

```
intolLs <- intolerable(expectedDf, intoleranceDf, RELEVANTINFO_ES)
```

The first argument in function `intolerable` is the ESM dataset in its current state. The second argument is the data.frame of intolerable combinations. `intolerable` returns a list to the user, containing 2 data.frames.

1. **cleanedDf**: This data.frame contains all tolerable ESM data.
2. **intoleranceDf**: This data.frame contains all intolerable ESM data.

An excerpt of the **intoleranceDf** for our exemplary ESM dataset looks like this:

Table 14: Excerpt of the result of `intolerable`, list element no.2: **intoleranceDf**.

ID	survey_name	ES_START_DATETIME	PROMPT	PROMPTFALSE	EXPCATEGORY
P003	dayControlGroup	2007-09-15 08:50:27	1	1	2

Note. ID = identification code, EXPCATEGORY = expected category as specified by the user.

Note for P003 that a day questionnaire (EXPCATEGORY = 2) was filled out at morning time (PROMPT = 1), which can't be tolerated in any further data analyses.

7.9 randomMultSelection (Function 25 of 29)

`randomMultSelection`: Use for C1, C3, C4, and C6 in table 1.
Error: There is at least one repeatedly filled out questionnaire of at least one participant. Multilevel statistical analyses deal with repeated measurements, yet each measurement is

expected to be unique at each specified prompt (per participant). Multiple measurements per prompt usually are not wanted. Unless they are valid (see combination C5 in table 1), repeatedly filled out questionnaires have to be removed from the raw ESM dataset. `randomMultSelection` uses the variable **ES_MULT**, which was generated by the function `esAssign` (7.1). For each participant, it randomly selects one questionnaire among the multiple questionnaires (per prompt and participant). The deselected questionnaires get discarded:

```
randSelLs <- randomMultSelection(intolLs[["cleanedDf"]])
```

`randomMultSelection` expects as only argument the ESM dataset in its current state. As result the user receives a list with 2 elements:

1. **esRandSelIn**: a data.frame. This dataset contains the ESM dataset in its current state, after intolerable questionnaires have been discarded.

2. **esRandSelOut**: a data.frame. This dataset contains all the discarded questionnaires.

Note that `randomMultSelection` updates the column **ES_MULT2** in the ESM dataset, since the shifting of questionnaires (in case it took place) makes this update necessary.

8 ESM dataset – Finalizing functions

8.1 computeTimeLag (Function 26 of 29)

`computeTimeLag`: Use for C1, C3, C4, and C6 in table 1.
Error: None. `computeTimeLag` computes the time difference between the scheduled start time (the prompt) of a questionnaire and its actual start time.

```
lagDf <- computeTimeLag(randSelLs[["esRandSelIn"]], RELEVANTVN_ES)
```

`computeTimeLag` expects as its first argument the ESM dataset in its current state. It will be returned to the user with 3 new columns, named **TIME_LAG**, which contains the time difference as described above, **LAG_PA**, which is dichotomous (0 = a questionnaire's actual start time is **P**rior to the scheduled time, else 1 = **A**fter the scheduled time). The third column's name is **ST_DATETIME**, which is the date-time object of the scheduled time/prompt.

8.2 computeDuration (Function 27 of 29)

`computeDuration`: Use for C1–C6 in table 1.
Error: None. `computeDuration` computes the time difference between the actual start time of a questionnaire and the actual end time of it. If there is no end time the function will return `NA` for this questionnaire.

```
durDf <- computeDuration(lagDf, RELEVANTVN_ES)
```

`computeDuration` expects as its first argument the ESM dataset in its current state. It will be returned to the user with 1 new column, named **DUR**, which contains the time difference as described above.

8.3 computeTimeBetween (Function 28 of 29)

`computeTimeBetween`: Use for C1–C6 in table 1.
Error: None. For each participant `computeTimeBetween` computes the time difference between the actual end time of a questionnaire and the actual start time of the subsequent questionnaire. If there is no end time the function will return `NA` for the subsequent questionnaire.

```
tbsqDf <- computeTimeBetween(esDf = durDf, refDf = referenceDfNew, RELEVANTVN_ES,
RELEVANTVN_REF)
```

`computeTimeBetween` expects as its first argument the ESM dataset in its current state and as second argument the reference dataset. The returned ESM dataset contains 1 new column: **TBSQ** (T_ime B_etween S_ubsequent Q_uestionnaires).

8.4 esFinal (Function 29 of 29)

`esFinal`: Use for C1–C6 in table 1.
Error: None. `esFinal` generates the final ESM dataset, i.e. for each missed questionnaire an empty line will be inserted. Also each participant in the final ESM dataset will have an equal amount of lines. To this end, `esFinal` searches across all participants in the ESM dataset for the maximum number of valid questionnaires per participant. This maximum number will then be assigned to each participant.

```
esDfFin <- esFinal(tbsqDf, esOpt=esAssigned[["ESopt"]], complianceRate=50,
RELEVANTINFO_ES, RELEVANTVN_ES, maxRows=28)
```

As its first argument `esFinal` expects the ESM dataset at its current state. The second argument is a data.frame that was returned as part of a list by the function esAssign.

The third argument makes the function remove all participants from the final dataset who answered less questionnaires (in percent) than specified by this third argument. A short explanation is necessary for what the last argument means: In order to statistically analyse the data, the number of questionnaires must be equal across all participants, which usually is not the case in any real world setting. Therefore, if participant i actually filled out say 19 questionnaires, whereas the maximum number (filled out by participant j) is 22, participant i will be assigned 3 empty lines of data at the end. These 3 lines are not missing questionnaires, they are fillers (see explanation below). If the user wants to constrain the maximum number of questionnaires to be less than the actually observed maximum number, the argument `maxRows` makes this possible. Otherwise, the user may simply ignore the argument `maxRows`:

```
esDfFin <- esFinal(tbsqDf, esOpt=esAssigned[["ESopt"]], complianceRate=50,
RELEVANTINFO_ES, RELEVANTVN_ES)
```

A list will be returned, with three elements: First, the final ESM dataset containing 2 new columns, both dichotomous, named **MISSED**, i.e. all questionnaires that should have been answered by the participant but weren't (1 = a missing questionnaire). The other column's name is **FILLER**, i.e. the empty questionnaires at the end of each participant, that are needed to get the number of questionnaires to be equal across all participants. Second, a data.frame that contains the completion rates for each participant. Third, a data.frame with all participants that were deselected due to having answered less questionnaires than the minimum percentage, as specified in the argument `complianceRate`, or, if no participant was deselected, a character string is returned that confirms that no deselection took place.

Don't forget to save the final ESM dataset by writing it to the hard disc of your computer!

9 R glossary in relation to this vignette (http-Links)

data.frame

character

list

vector

numeric

10 Quick guide: Adapt 'esmprep' to YOUR project

ESM versions = 1 means that every day each participant answered the exact same questionnaire each time.

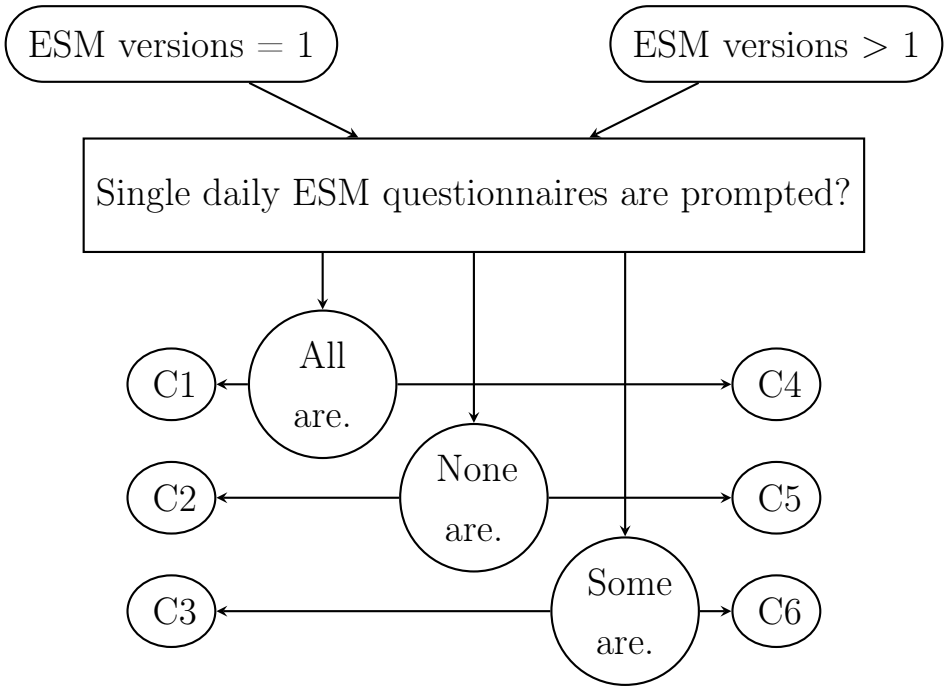


Figure 1: Combinations (C) of ESM variants that **esmprep** can be adapted to. See also table 1.

11 'esmprep' versions, up-dates, and changes

Version no.	Date (CRAN)	Date (GitHub)
0.1.0	2017-08-30	2017-08-30 15:40:27 UTC
0.1.1	–	2017-09-08 18:04:00 UTC
0.1.2	–	2017-09-15 10:20:44 UTC
0.1.3	2018-01-11	2018-01-11 13:55:54 UTC
0.1.4	2018-04-18	2018-04-18 11:17:02 UTC
0.1.5	–	2018-06-26 10:39:02 UTC
0.1.6	–	2018-06-27 09:05:58 UTC
0.1.7	–	2019-02-22 12:06:56 UTC
0.1.8	–	2019-04-26 17:11:46 UTC
0.1.9	–	2019-05-17 09:47:03 UTC
0.2.0	–	2019-05-28 18:03:31 UTC

Changes from 0.1.0 to 0.1.1

Overall reason for the change:
The package was unable to deal with when a participant was assigned exactly one ESM questionnaire, which should be an extremely rare case. The remaining 2 cases (assignment of no questionnaire and ≥ 2 questionnaires) was not problematic.

- **Function** `genDateTime`
  Error handling removed, capturing whether the raw ESM dataset at this point in the '**esmprep**'-hierarchy of functions is single or not (`if(length(RELEVANTINFO_ES[["SVYNAMES"]]) == 1) {...}`). Consequence: Function `esList` has to be applied, irrespective of whether the number of ESM versions is 1 or ≥ 1.

- **Function** `overallCounter` (not exported/invisible to the user)
  An error was thrown when exactly 1 ESM questionnaire was assigned to a participant (see function `esAssign`). Reason for the error was `diffAssigned <- c(0, diff(idxAssigned))`, i.e. diff() requires at least 2 numeric values. Problem has been fixed.

- **Function** `esAssign`
  The variable `CV_ESDAY` in output dataset `ES` is generated in 2 different ways, because of the boolean (TRUE/FALSE) argument `midnightPrompt`. If set to TRUE, `CV_ESDAY` is computed in a more complex way, compared to the default FALSE. Using the result of

this complex computation might cause `CV_ESDAY` to be wrong, if `midnightPrompt` is set to TRUE, although FALSE would be the correct setting[11]. Therefore, if `midnightPrompt` is correctly set to FALSE, the less complex computation of `CV_ESDAY` is used from now on.

Changes from 0.1.1 to 0.1.2

Overall reason for the change:
The function `esList` strongly depends on the raw ESM dataset(s) to contain a column that specifies the name of the ESM version. However, such a column doesn't have to exist necessarily in all ESM combinations, which is why the functions `relevantESVN` and `setES` give the user the option to keep the respective arguments' default value (NULL). The user is recommended to solve the issue differently, yet if the option is chosen against this recommendation, the function `esList` now generates columns that specify a bogus ESM version name.

- **Function** `relevantESVN`
  Instead of 'NO_ESM_COLUMN_PRESENT_SPECIFYING_SURVEY_NAME' the list element `svyName` now contains 'ES_SVY_NAME'.
  Also, if no column for the survey name is passed, the function doesn't return a warning message any more, but simply a message, that states that the user has not passed a column name that specifies the name of the ESM survey version.

- **Function** `setES`
  Instead of 'NO_SURVEY_NAMES_PRESENT' the list element `SVYNAMES` now contains 'NO_SURVEY_NAMES_PASSED_BY_USER'.

- **Function** `esList`
  See description of the overall reason for the change from 'esmprep' version 0.1.1 to 0.1.2.

- **Function** `esComplete`
  If there is not more than one ESM dataset, the function `esComplete` won't append '_1' to the name of the survey version, because it makes no sense. It makes sense in other cases, though! This is why it happened also in the case of one ESM survey. This has nothing to do with the overall reason for the change.

Changes from 0.1.2 to 0.1.3

- **Function** `computeTimeBetween`
  Error handling added, capturing whether there are participants in the reference dataset, that haven't been assigned by the function `esAssign`. Without this error handling the function can't fully execute. A warning message is prompted to the user for each unassigned participant.

[11] TRUE is the correct setting if at least one of the participants was prompted around midnight and/or if it was possible for any participant to fill out a questionnaire around midnight (even though all daily prompts were in large distance to midnight).

Changes from 0.1.3 to 0.1.4

- **Functions** `overallCounter` and `suggestShift`
  Someone from R wrote an e-mail to esmprep@gmail.com, prompting me to remove any `break` and/or `next` commands from the source code, when 'no loop was visible'. After checking the source code I removed two `break` commands and changed the code in such a way that functionality of both functions remained intact.

Changes from 0.1.4 to 0.1.5

- **Functions** `findMin1` and `findMin2`
  Both functions are not exported, i.e. they are not visible to the user. They are used within the function `esAssign`. Both functions are finding the minimum time difference between the actual start time of an ESM questionnaire and each of the scheduled start times (see columns `PROMPT` and `ST` of resulting dataset `ES`, under the condition that at least one of the ESM questionnaires was prompted). Both functions are doing the same for the actual end time of an ESM questionnaire, which is where the bug was discovered. Although very unlikely, it can happen that an actual end time of an ESM questionnaire has the exact same time difference to two of the scheduled start times. Of course, this is also true for any of the actual start times. In such a case, the first of the two scheduled times is chosen, yet so far, this was only done for the actual start time, not for the actual end time. This has been fixed.
  Before the line of code looked like this: `minEnd1 <- which(minEnd == absEnd_iSec)`.
  The corrected code now looks like this: `minEnd1 <- which(minEnd == absEnd_iSec)[1]`.

- **Function** `esAssign`
  Inside the function `esAssign` the ESM questionnaires are assigned to the respective participant by extracting all ESM questionnaires that have been started between the two timepoints, which must have been set in the reference dataset. Timepoint 1 is the first possible ESM questionnaire that the participant is supposed to answer after the ESM period has started. Timepoint 2 is the last possible ESM questionnaire that the participant is supposed to answer before the ESM period has ended. The problem concerns timepoint 2, i.e. if the last possible ESM questionnaire is scheduled at say 12:00 h, then so far this last questionnaire was excluded by the code, whenever the actual start time was after the scheduled time, e.g. at 12:01 h. The problem has now been solved by **extending** the time for timpoint 2 (the very last scheduled time within the ESM period) by the amount of minutes that are passed to the argument `promptTimeframe` in the function `esAssign`.
  The argument `promptTimeframe` is now set to a default of 30 minutes. Passing NA, NULL, or the value 0 to the argument will result in an error. Passing less than 30 minutes will result in a warning message, stating that this might lead to the unintended exclusion of the last ESM questionnaire of a participant, namely if the actual start time of the last ESM questionnaire is after the **extended** scheduled end time of the ESM period for that

participant.

BEWARE: Even if the ESM study doesn't contain any prompted questionnaires at all (only event contingent ones), still there must be one scheduled start time and one scheduled end time, for which a time frame can be set.

Changes from 0.1.5 to 0.1.6

- **Functions** `makeShift`

  Bug fix.

  Details: The reference dataset must contain the single prompts of when the ESM questionnaires were expected to be answered. The column names of these variables can be chosen freely. However, the root across all of these variable names must be identical, e.g. st. If there were 4 prompts, the variable names would read st1, st2, st3, and st4. Inside the function `makeShift` the root will be extracted. If the root of at least one of these variable names is not identical to the other variable names, the function `makeShift` will return an error.

Changes from 0.1.6 to 0.1.7

- **Function** `esFinal`

  Extending the function's output: In the end of an ESM study, the researcher usually needs to know how much percent of possible questionnaires each participant actually answered, because for multilevel analyses a certain minimum number of observations per participant is necessary for interpreting the results. The function now returns a data.frame which contains each participant's completion rates, both for each prompt and across all prompts. Furthermore, the function removes all participants from the final dataset, who answered less than a certain percentage of questionnaires, as specified by the user with the argument `complianceRate`. The removed participants will be returned in a separate data.frame, called `ESfinalOut`. Also, a bug was fixed concerning the two new columns `MISSED` and `FILLER`. For this bug fix the new argument `esOpt` came in very handy.

- **Function** `refPlausible`

  New function: `refPlausible` is strongly recommended to be used right after function `genDateTime` (or the opposite function `splitDateTime`) was applied to the reference dataset. `refPlausible` returns the ESM time period for each participant, as specified by the user with the argument `units`, e.g. days. The user can then check whether the ESM time period is roughly equal across all participants, which usually is the goal in an ESM study. For instance, the R base function `summary` can be used to get a quick overview. Especially, there shouldn't be any negative values, meaning that the start of an ESM period is after the ESM period's end, which is impossible, in case time travelling can be ruled out.

- **Function** `suggestShift`

  Making the function's output consistent across both possible cases: Case 1, of all ques-

tionnaires in the ESM dataset at least one questionnaire is suggested to shift from one prompt to a neighbouring prompt. In this case the output always was a list, nothing changed compared to earlier versions. Case 2, no questionnaire is suggested to shift. In this case the output wasn't a list containing a data.frame as its first element. Instead the output was the exact same data.frame that was passed to the function. However, in case 2 the function that needs to be applied after the function `suggestShift` is the function `expectedPromptIndex`, which takes a part of the result of the function `makeShift` as its first argument (That is: It takes the first element of the list (returned by `makeShift`), which is the ESM dataset in its current state). However, in case 2 the function `makeShift` won't be applied, which is why the function `expectedPromptIndex` must use the result of the function `suggestShift`. Since in case 1 and case 2 the function `suggestShift` returned different data structures (case 1: a list of length 3; case 2: a data.frame), the use of the function `expectedPromptIndex` was unnecessarily complicated because the first argument in case 1 was the first element of a list, whereas in case 2 it was a data.frame. From version 0.1.7 onwards the result of the function `suggestShift` will always be a list, of which the first element will be the ESM dataset in its current state. Therefore, the function `expectedPromptIndex` doesn't need to be adapted any longer because case 2 exists no longer.

Changes from 0.1.7 to 0.1.8

- **Function** `refPlausible`
  Extended functionality: The function will check whether the time series of the prompts are strictly increasing per participant, because 2 or more prompts set at the exact same time is as wrong as a prompt being set at an earlier (instead of a later) time than the previous prompt. One possible 'exception' occurs when two subsequent prompts cross midnight (change of one date to the next). Therefore, the function will also display a message that maybe the user should set the argument `midnightPrompt` to `TRUE` in the function `esAssign`. Additionally, the function will check whether there are any duplicates among the participant IDs, in which case an error message is returned. The problematic part of the reference dataset will be displayed in the R console.

Changes from 0.1.8 to 0.1.9

- **Function** `refPlausible`:
  Bug fix.

Changes from 0.1.9 to 0.2.0

- **Function** `refPlausible`
  Extended functionality: For each participant, the function will additionally check whether the overall start time and end time of the ESM period is among the prompts of the respective participant. If not, an error will be returned.