Package: appkitr (via r-universe)

November 14, 2026

Title Utilities to Work with 'GESIS AppKit' Data

Version 2.0.0.5

Description Provides helper functions for importing and processing survey data exported from the 'GESIS AppKit' platform <https://www.gesis.org/en/gesis-appkit>. Includes support for extracting variable and value labels from 'SPSS' syntax files, and automatically attaching these labels to raw CSV survey data for clean analysis.

License Apache License (>= 2)

URL https://gitlab.com/gesis-appkit/appkitr, https://gesis-appkit.gitlab.io/appkitr

BugReports https://gitlab.com/gesis-appkit/appkitr/-/work_items

Depends R (>= 4.1.0)

Imports cli, stats, utils, withr

Suggests testthat (>= 3.0.0)

Config/roxygen2/version 8.1.0

Encoding UTF-8

LazyData true

NeedsCompilation no

Author Maximilian Haag [aut, cre] (ORCID: <https://orcid.org/0000-0002-0334-9270>), Charlotte de Alwis [aut] (ORCID: <https://orcid.org/0009-0004-8903-7500>), Julian Kohne [aut] (ORCID: <https://orcid.org/0000-0002-6710-7545>), GESIS - Leibniz Institute for the Social Sciences [cph]

Maintainer Maximilian Haag <mail@mxhg.org>

Repository https://cran.r-universe.dev

Date/Publication 2026-11-13 16:40:27 UTC

RemoteUrl https://github.com/cran/appkitr

RemoteRef HEAD

RemoteSha 941b93f7787cea9fc4d1b733babbb735cfdc3558

Contents

appkit_survey_entry . . . 2
appkit_survey_esm . . . 4
appkit_survey_exit . . . 5
calculate_payout . . . 6
create_payout_scheme . . . 8
drop_participants . . . 9
inspect_variable . . . 10
inspect_variables . . . 10
participation_table . . . 11
read_appkit_survey . . . 12
read_appkit_surveys . . . 14

Index **16**

`appkit_survey_entry` *Example Entry Survey Data*

Description

This is synthetic example data intended to allow users to test the package's functions. The variables are from a study conducted by the GESIS AppKit Team on measuring emotionality in social interactions through momentary assessments. This is the entry survey, which includes items from the affect intensity measure questionnaire.

Usage

```
appkit_survey_entry
```

Format

A data frame with 20 rows and 40 columns:

`personalParticipantCode` PersonalParticipantCode

`generalLoginCode` GeneralLoginCode

`status` Participant Status

`lastStatusChange` Last Status Change

`note` Note

`firstOpened` First Opened

`committed` Committed

`scheduled` Scheduled

`expired` Expired

`published` Published

`exported` Exported

`osVersion` OS Version

`smartphoneType` SmartphoneType

`pushNotificationAllowed` Push Notification Allowed

`notificationType` Notification Type

`notificationTitle` Notification Title

`notificationText` Notification Text

`age` How old are you (in years)?

`sex` Which sex was assigned to you at birth?

`gender` What is your gender?

`aim01` When I feel happiness, it is a quite type of contentment.

`aim02` When a person in a wheelchair can't get through a door, I have strong feelings of pity.

`aim03` I get upset easily.

`aim04` When I succeed at something, my reaction is calm contentment.

`aim05` I get really happy or really unhappy.

`aim06` I am a fairly quiet person.

`aim07` When I'm happy, I feel fairly energetic.

`aim08` Seeing a picture of some violent car accident in a newspaper makes me feel sick to my stomach.

`aim09` When I'm happy, I feel like I'm bursting with joy.

`aim10` I would be very upset, if I got a traffic ticket.

`aim11` Looking at beautiful scenery really doesn't affect me much.

`aim12` The weather doesn't affect my mood.

`aim13` Others tend to get more excited than I do.

`aim14` I am not an extremely enthusiastic individual.

`aim15` Calm and cool could easily describe me.

`aim16` When I'm feeling well it is easy for me to go from being in a good mood to being really joyful.

`aim17` When I worry, it is so mild I hardly notice it.

`aim18` I get overly enthusiastic.

`aim19` My happy moods are so strong that I feel like I'm 'in heaven'.

`aim20` When something bad happens, others tend to be more unhappy than I.

`appkit_survey_esm` *Example ESM Data*

Description

This is synthetic example data intended to allow users to test the package's functions. The variables are from a study conducted by the GESIS AppKit Team on measuring emotionality in social interactions through momentary assessments. This is the experience sampling survey, which captures the momentary assessments of emotions in social interactions.

Usage

```
appkit_survey_esm
```

Format

A data frame with 180 rows and 36 columns:

`personalParticipantCode` PersonalParticipantCode

`generalLoginCode` GeneralLoginCode

`status` Participant Status

`lastStatusChange` Last Status Change

`note` Note

`firstOpened` First Opened

`committed` Committed

`scheduled` Scheduled

`expired` Expired

`published` Published

`exported` Exported

`osVersion` OS Version

`smartphoneType` SmartphoneType

`pushNotificationAllowed` Push Notification Allowed

`notificationType` Notification Type

`notificationTitle` Notification Title

`notificationText` Notification Text

`screening` Have you been part of any social interaction in the last two hours?

`description` Please briefly describe the type of social interaction.

`delay` How many minutes ago did this interaction take place?

`duration` How long did the interaction last?

`groupsize` How many people took part in the interaction (you included)?

`partner_gender` What gender do you ascribe to your (main) interaction partner?

`partner_close` How close do you consider the relationship to your (main) interaction partner?

`interaction_quality` Please indicate your perceived quality of the interaction.

`interaction_intimacy` Please indicate the intimacy of the interaction.

`interaction_control` Please indicate your perceived degree of control over the interaction in comparison to your main interaction partner.

`interaction_intensity` Please indicate the overall intensity of your emotional experience during the interaction.

`emo_happy` Please indicate how happy you felt during the interaction.

`emo_sad` Please indicate how sad you felt during the interaction.

`emo_nervous` Please indicate how nervous you felt during the interaction.

`emo_surprise` Please indicate how surprised you felt during the interaction.

`emo_angry` Please indicate how angry you felt during the interaction.

`emo_embarrassed` Please indicate how embarrassed you felt during the interaction.

`emo_ashamed` Please indicate how ashamed you felt during the interaction.

`dataquality` Thank you very much for supporting our research. For ensuring data quality, please indicate whether you answered the questions within this survey conscientiously:

`appkit_survey_exit` *Example Exit Survey Data*

Description

This is synthetic example data intended to allow users to test the package's functions. The variables are from a study conducted by the GESIS AppKit Team on measuring emotionality in social interactions through momentary assessments. This is the exit survey, which includes items from the personal attributes questionnaire.

Usage

```
appkit_survey_exit
```

Format

A data frame with 20 rows and 33 columns:

`personalParticipantCode` PersonalParticipantCode

`generalLoginCode` GeneralLoginCode

`status` Participant Status

`lastStatusChange` Last Status Change

`note` Note

`firstOpened` First Opened

committed Committed

scheduled Scheduled

expired Expired

published Published

exported Exported

osVersion OS Version

smartphoneType SmartphoneType

pushNotificationAllowed Push Notification Allowed

notificationType Notification Type

notificationTitle Notification Title

notificationText Notification Text

paqf1 I am emotional.

paqf2 I am able to devote myself completely to others.

paqf3 I am understanding.

paqf4 I am gentle.

paqf5 I am helpful to others.

paqf6 I am kind.

paqf7 I am unaware of the feelings of others.

paqf8 I am warm in relation to others.

paqm1 I am independent.

paqm2 I am a proactive person.

paqm3 I am competitive.

paqm4 I have difficulties making decisions.

paqm5 I give up easily.

paqm6 I feel superior.

paqm7 I am self-confident.

paqm8 I function well under pressure.

`calculate_payout` *Calculate payout per participant based on survey participation*

Description

Calculates the payout per participant according to a payout scheme. Supports two payment types per survey: a fixed amount per committed questionnaire, and/or a one-time bonus once a minimum number of committed questionnaires per survey is reached.

Usage

```
calculate_payout(
  data,
  payout_scheme,
  participant_col = .appkit_col("participant"),
  scheduled_col = .appkit_col("scheduled"),
  committed_col = .appkit_col("committed"),
  exported_col = .appkit_col("exported")
)
```

Arguments

`data`	Either a single survey data frame (as returned by `read_appkit_survey()`), or a list of survey data frames (as returned by `read_appkit_surveys()`, or assembled manually, e.g. `list(entry = entry_survey, exit = exit_survey)`).
`payout_scheme`	A data frame with one row per payout rule, and columns: **survey** Character. Survey name, must match a name in `data`. **amount** Numeric. Amount per questionnaire (default rule type, `"per_questionnaire"`), or one-time bonus amount (`"threshold"` rule type). **type** Optional. Character, either `"per_questionnaire"` (pay `amount` for every committed questionnaire in this survey) or `"threshold"` (pay `amount` once if the participant's committed count for this survey is `>= threshold_n`). Defaults to `"per_questionnaire"` for every row if the column is omitted entirely. **threshold_n** Optional. Integer, minimum number of committed questionnaires required for a `"threshold"` rule. Required (non-`NA`) only for rows with `type = "threshold"`. Omit the column if you only use `"per_questionnaire"` rules.
`participant_col`	Character string naming the participant ID column (default: `"personalParticipantCode"`).
`scheduled_col`	Character string naming the scheduled-timestamp column (default: `"scheduled"`).
`committed_col`	Character string naming the committed-timestamp column (default: `"committed"`).
`exported_col`	Character string naming the export-timestamp column, used internally by `participation_table()` (default: `"exported"`).

Value

A data frame with one row per participant and one column per survey (payout amount for that survey), plus a `total` column (sum across all surveys).

Disclaimer

This function is provided as a calculation aid only. It comes with no guarantee of correctness. Always manually verify payout amounts before making any actual payments.

Examples

```
payout_scheme <- data.frame(
  survey = c("appkit_survey_esm", "appkit_survey_esm"),
  type = c("per_questionnaire", "threshold"),
  threshold_n = c(NA, 10),
  amount = c(2.00, 15.00)
)

payout <- calculate_payout(appkit_survey_esm, payout_scheme)
head(payout)
```

`create_payout_scheme` *Interactively create a payout scheme*

Description

This command walks the user through each survey in the input `data`, asking whether to include a payout rule for each one and, if so, which type(s). This avoids manually constructing a `payout_scheme` data frame, which might be more error-prone.

Usage

```
create_payout_scheme(data)
```

Arguments

`data`	Either a single survey data frame (as returned by `read_appkit_survey()`), or a list of survey data frames (as returned by `read_appkit_surveys()`, or assembled manually, e.g. `list(entry = entry_survey, exit = exit_survey)`).

Value

A `payout_scheme` data frame, suitable for `calculate_payout()`.

See Also

`calculate_payout()`

Examples

```
if (interactive()) {
  payout_scheme <- create_payout_scheme(appkit_survey_esm)
  payout_scheme
}
```

`drop_participants` *Drop Participants from Survey Data*

Description

Removes rows matching specified participant IDs from a data frame or a list of data frames (as returned by `read_appkit_surveys()`).

Usage

```
drop_participants(data, participant_ids, id_col = .appkit_col("participant"))
```

Arguments

`data`	A data frame or a named list of data frames.
`participant_ids`	Character vector of participant IDs to remove.
`id_col`	Character string naming the column that contains participant IDs (default: the AppKit participant code column `"personalParticipantCode"`).

Details

When `data` is a list of data frames, the function applies the drop operation to each element and reports results per survey.

If any values in `participant_ids` are not found in the data, a warning is issued listing the unmatched IDs.

Value

Object of the same type as `data`, with matching rows removed.

Examples

```
# Drop two participants from the bundled example survey
ids <- appkit_survey_entry$personalParticipantCode[1:2]
entry_small <- drop_participants(appkit_survey_entry, ids)

# Drop participants from a named list of surveys
surveys <- list(entry = appkit_survey_entry)
surveys <- drop_participants(surveys, ids)
```

inspect_variable *Inspect a Single Variable*

Description

Extracts and formats information about a single variable including its label, unique values, and value labels.

Usage

```
inspect_variable(var, var_name = NULL, max_unique = 10, max_width = 50)
```

Arguments

var	The variable to inspect
var_name	The name of the variable (optional, auto-detected if not provided)
max_unique	Maximum number of unique values to display (default: 10)
max_width	Maximum width for displaying unique values (default: 50)

Value

A one-row data frame with variable information

Examples

```
# Inspect a single variable of the bundled example survey
var_info <- inspect_variable(appkit_survey_entry$aim01)
```

inspect_variables *Inspect Variables in a Data Frame or Single Variable*

Description

Provides a formatted overview of variables in a data frame, showing variable names, labels, unique values, and value labels. Can also inspect a single variable.

Usage

```
inspect_variables(data, max_unique = 10, max_width = 50, verbose = TRUE)
```

Arguments

data	A data frame or a single variable to inspect
max_unique	Integer indicating the maximum number of unique values to display (default: 10)
max_width	Integer indicating the maximum width for displaying unique values (default: 50)
verbose	Logical, if set to FALSE, suppresses all console output (default: TRUE)

Value

A tibble-like display of variable information (returned invisibly)

Examples

```
# Inspect all variables in the bundled example survey
inspect_variables(appkit_survey_entry)
```

---

participation_table *Report committed questionnaires by participant*

Description

Creates a table summarizing, for each participant, how many questionnaires they committed per survey, either as absolute counts or as a percentage.

Usage

```
participation_table(
  data,
  value = c("count", "percentage"),
  include_future_scheduled = FALSE,
  participant_col = .appkit_col("participant"),
  scheduled_col = .appkit_col("scheduled"),
  committed_col = .appkit_col("committed"),
  exported_col = .appkit_col("exported")
)
```

Arguments

data	Either a single survey data frame (as returned by `read_appkit_survey()`), or a list of survey data frames (as returned by `read_appkit_surveys()`, or assembled manually, e.g. `list(entry = entry_survey, exit = exit_survey)`).
value	Character. Either `"count"` (absolute number of committed questionnaires per participant per survey, default) or `"percentage"` (committed questionnaires as a percentage of that participant's total questionnaires recorded for that survey, i.e. a per-participant completion rate).

include_future_scheduled
: Logical. The export contains rows for questionnaires scheduled in the future. Calculating percentages mid-study can therefore be misleading. If FALSE (default), rows with a scheduled timestamp after the export's exported timestamp are excluded; this requires AppKit >= 1.17.0. Set to TRUE to include them in the calculation.

participant_col
: Character string naming the participant ID column (default: "personalParticipantCode").

scheduled_col
: Character string naming the scheduled-timestamp column (default: "scheduled").

committed_col
: Character string naming the committed-timestamp column (default: "committed").

exported_col
: Character string naming the export-timestamp column, used as the reference time for include_future_scheduled. (default: "exported").

Value

A data frame with one row per participant and one column per survey (plus personalParticipantCode), containing counts or completion percentages. Participants are included if they appear in at least one of the supplied surveys.

Examples

```
# count of committed questionnaires per participant
survey_count <- participation_table(appkit_survey_esm, include_future_scheduled = TRUE)
head(survey_count)

# percentage of committed questionnaires per participant
survey_pct <- participation_table(
  appkit_survey_esm,
  value = "percentage",
  include_future_scheduled = TRUE
)
head(survey_pct)

# participation for multiple surveys, combined in a list
surveys_count <- participation_table(
  list(survey_esm = appkit_survey_esm, survey_exit = appkit_survey_exit),
  include_future_scheduled = TRUE
)
head(surveys_count)
```

---

read_appkit_survey *Read AppKit Survey Data from Directory*

Description

Reads an AppKit survey dataset from a directory containing both a CSV file and an SPS syntax file. Automatically applies variable and value labels, converts eligible character columns to logical, and handles NA values.

Usage

```
read_appkit_survey(
  survey_directory,
  na_values = list(-1, "", "Not applicable", "N/A"),
  timestamp_cols = .appkit_timestamp_cols()
)
```

Arguments

survey_directory	A character string specifying the path to a directory containing both a .csv data file and a .sps syntax file.
na_values	Optional list or vector of values to treat as NA (default: list(-1, "", "Not applicable", "N/A")). All specified values will be converted to NA across all columns with appropriate type handling.
timestamp_cols	Character vector of column names to parse as timestamps (default: the AppKit timestamp columns lastStatusChange, firstOpened, committed, scheduled, expired, published, exported). Columns not present in the data are ignored.

Details

The function performs the following operations

- Reads the CSV file using semicolon separator
- Converts specified values to NA with type-appropriate handling
- Converts character columns containing only "true"/"false" to logical
- Parses the `timestamp_cols` columns with multiple format attempts
- Extracts variable and value labels from the SPS file
- Applies labels to the data frame (variable labels as attributes, value labels as factors)

Value

A data frame containing the processed survey data with variable and value labels applied as attributes and appropriate data types set.

Examples

```
survey_dir <- system.file("extdata", "example_survey",
  package = "appkitr", mustWork = TRUE
)

# Read survey from directory with default settings
survey <- read_appkit_survey(survey_dir)

# Read survey and treat custom values as missing
survey <- read_appkit_survey(survey_dir, na_values = c(-1, -99))
```

`read_appkit_surveys` *Read Multiple AppKit Surveys from Directory*

Description

Reads multiple AppKit surveys from a parent directory where each subdirectory contains one survey (CSV + SPS files). Returns a named list of data frames.

Usage

```
read_appkit_surveys(
  surveys_directory,
  na_values = list(-1, "", "Not applicable", "N/A"),
  duplicate_names = "error",
  timestamp_cols = .appkit_timestamp_cols()
)
```

Arguments

`surveys_directory`
: A character string specifying the path to a directory containing subdirectories, each with survey data (CSV + SPS files).

`na_values`
: Optional list or vector of values to treat as NA across all surveys (default: list(-1, "", "Not applicable", "N/A")). All specified values will be converted to NA across all columns with appropriate type handling.

`duplicate_names`
: Character string specifying how to handle surveys with identical names. Options are:

  "error" (default) Throw an error when duplicate survey names are found

  "merge" Combine surveys with the same name using rbind()

  "distinct" Keep surveys distinct by adding numeric suffixes (e.g., "survey", "survey_2", "survey_3")

`timestamp_cols`
: Character vector of column names to parse as timestamps across all surveys (default: the AppKit timestamp columns). Columns not present in a survey are ignored.

Value

A named list where each element is a processed survey data frame. Names are derived from the CSV filenames (with .csv extension removed and spaces replaced with underscores).

Examples

```
surveys_dir <- system.file("extdata", "example_surveys",
  package = "appkitr", mustWork = TRUE
)
```

```
# Read all surveys from a parent directory (default: error on duplicates)
surveys <- read_appkit_surveys(surveys_dir)

# Read surveys and merge those with identical names
surveys <- read_appkit_surveys(surveys_dir, duplicate_names = "merge")

# Read surveys and keep duplicates distinct with suffixes
surveys <- read_appkit_surveys(surveys_dir, duplicate_names = "distinct")

# Read surveys with custom NA values
surveys <- read_appkit_surveys(surveys_dir, na_values = c(-1, -99, ""))

# Access individual surveys
survey1 <- surveys[["survey_name_1"]]
```

Index

∗ **datasets**
appkit_survey_entry, 2
appkit_survey_esm, 4
appkit_survey_exit, 5

appkit_survey_entry, 2
appkit_survey_esm, 4
appkit_survey_exit, 5

calculate_payout, 6
calculate_payout(), *8*
create_payout_scheme, 8

drop_participants, 9

inspect_variable, 10
inspect_variables, 10

participation_table, 11

read_appkit_survey, 12
read_appkit_surveys, 14
read_appkit_surveys(), *9*