Package 'aidR'

November 18, 2026

Type Package

Title Reading and Standardizing Automated Insulin Delivery Data

Version 0.2.5

Description Parses the native exports of several automated insulin delivery (AID) platforms into a common, platform-independent format for continuous glucose monitoring (CGM), basal, bolus, carbohydrate, and self-monitored blood glucose (SMBG) data. Additional functions check completeness, filter, produce visualizations, and convert the data to other formats.

License GPL-3

Encoding UTF-8

Depends R (>= 4.1.0)

Imports readxl, tools, dplyr, lubridate, rjson, tidyr, stringr, utils, rlang, stats, graphics, iglu, hms

Config/roxygen2/version 8.1.0

Suggests knitr, rmarkdown, cgmquantify, ggplot2, testthat (>= 3.0.0), withr

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/madleina/aidR

BugReports https://github.com/madleina/aidR/issues

NeedsCompilation no

Author Madleina Caduff [aut, cre] (ORCID: <https://orcid.org/0000-0001-8941-4274>), Aritz Lizoain [aut] (ORCID: <https://orcid.org/0009-0004-5516-2573>), Diabetes Center Berne (DCB) [cph, fnd]

Maintainer Madleina Caduff <madleina.caduff@dcberne.com>

Repository CRAN

Date/Publication 2026-11-18 07:50:02 UTC

Contents

check_completeness . . . 2
check_completeness_range . . . 3
check_insulin_totals . . . 4
filter_carbs . . . 5
filter_CGM . . . 6
get_cgmquantify_format . . . 7
get_iglu_format . . . 8
merge_all . . . 8
parse_data . . . 9
plot_day . . . 10
removed_entries . . . 12
write_DIAX_format . . . 12

Index **14**

`check_completeness` *Check if data for different types are available for at least one day*

Description

Accepts either the output of `parse_data` for a single individual or the output of `merge_all` for multiple individuals. Every individual found in the data is checked separately.

Usage

```
check_completeness(data, necessary_types = c("cgm", "basal", "bolus", "carbs"))
```

Arguments

`data`	A list with all data found for one individual (the output of `parse_data`) or for multiple individuals (the output of `merge_all`). Requires cleaned and standardized data.
`necessary_types`	A character vector with type names for which data should be available.

Value

No return value, called for side effects. Issues a message for every individual and data type with missing data.

Examples

```
# A mylife export contains no CGM and no basal data
path <- system.file("extdata", "mylife_example.csv", package = "aidR")
data <- parse_data(id = "1", paths = path)
check_completeness(data)

# Only require bolus and carbohydrate data
check_completeness(data, necessary_types = c("bolus", "carbs"))
```

`check_completeness_range` *Check if data for different types are available for a range of dates*

Description

Accepts either the output of `parse_data` for a single individual or the output of `merge_all` for multiple individuals. Every individual found in the data is checked separately.

Usage

```
check_completeness_range(
  data,
  first_date,
  last_date,
  necessary_types = c("cgm", "basal", "bolus", "carbs")
)
```

Arguments

`data`	A list with all data found for one individual (the output of `parse_data`) or for multiple individuals (the output of `merge_all`). Requires cleaned and standardized data.
`first_date`	A character string representing the first date for which data should be available.
`last_date`	A character string representing the last date (included) for which data should be available.
`necessary_types`	A character vector with type names for which data should be available.

Value

No return value, called for side effects. Issues a message for every individual and data type with missing days.

Examples

```
# The example export covers 2026-06-22 only
path <- system.file("extdata", "glooko_example.zip", package = "aidR")
data <- parse_data(id = "1", paths = path)
check_completeness_range(data, first_date = "2026-06-21", last_date = "2026-06-23")
```

`check_insulin_totals` *Check that the summed insulin matches the daily totals reported by an export*

Description

Compares the insulin summed over the individual data points with the daily total reported by the export itself. Only daily totals with `source == "reported"` are checked: where aidR summed the total itself (`source == "summed"`), the two agree by construction. In practice this means Glooko (basal and bolus) and Tandem Source (basal only).

Usage

```
check_insulin_totals(data, types = "basal", rel_tol = 0.2, abs_tol = 1)
```

Arguments

`data`	A list with all data found for one individual (the output of `parse_data`) or for multiple individuals (the output of `merge_all`). Requires cleaned and standardized data.
`types`	Character vector with the types of insulin that are checked, any of `"basal"` and `"bolus"`.
`rel_tol`	Numeric, the relative tolerance of the comparison.
`abs_tol`	Numeric, the absolute tolerance of the comparison, in units of insulin. A day is flagged when the absolute difference exceeds both `abs_tol` and `rel_tol` times the reported total.

Details

Accepts either the output of `parse_data` for a single individual or the output of `merge_all` for multiple individuals. Every individual found in the data is checked separately.

By default, only the basal insulin is checked, which is where the mismatches described below arise. Pass `types = c("basal", "bolus")` to check the bolus insulin as well.

A large mismatch means that the individual data points do not cover the whole day. The most common cause are AID systems that record basal rates for manual mode only (e.g. Omnipod on Glooko): the basal insulin delivered in automated mode is missing from the basal rates, but is included in the reported total.

Note that the first and the last day of an export are usually only partially covered, so a mismatch on those days is expected.

Value

Invisibly, a data frame with one row per checked individual, day and type, holding the summed and the reported total and whether the two agree. Called for its side effects: a message is issued for every day where they do not.

Examples

```
path <- system.file("extdata", "glooko_example.zip", package = "aidR")
data <- parse_data(id = "1", paths = path)
check_insulin_totals(data)

# Check bolus insulin as well, and keep the comparison
res <- check_insulin_totals(data, types = c("basal", "bolus"))
res
```

`filter_carbs` *Filter carbohydrate entries*

Description

Removes carbohydrate entries that were overwritten by another entry.

Usage

```
filter_carbs(data_id, max_minutes_overwrite = 2)
```

Arguments

`data_id`	A list with all data found for one individual. Filtering is only performed if the carbohydrate and bolus data were cleaned and standardized.
`max_minutes_overwrite`	Maximum time difference (in minutes) between an overwritten entry and another entry. Default 2 minutes. If NULL, nothing is removed.

Details

In the bolus calculator, carbs may be entered, but the bolus may not be delivered (or cancelled), and the carbs may then be entered again, with a bolus shortly after. Some platforms (e.g. Glooko) export all of these entries.

A carbohydrate entry is removed if

- its bolus is zero (found via `link_id`)
- another carbohydrate entry of the same `format` (or sharing one, for combined formats such as `"glooko/tidepool"`), with a different `link_id`, lies at most `max_minutes_overwrite` minutes before or after it.
- it is not flagged as a hypo treatment (`is_hypo_treatment`).

Slowly absorbed carbohydrates merged during cleaning (linked entries with an `estimated_absorption_duration`, see `clean.glooko`) are never removed and do not count as another entry. The zero bolus linked to a removed entry is removed from the bolus data as well.

Value

A list with all data found for one individual, filtered, if necessary. Removed rows are stored in the attribute `removed`, see `removed_entries()`.

Examples

```
path <- system.file("extdata", "glooko_example.zip", package = "aidR")
data <- parse_data(id = "1", paths = path)
nrow(data$carbs)

data_filtered <- filter_carbs(data)
nrow(data_filtered$carbs)
removed_entries(data_filtered)
```

`filter_CGM` *Filter CGM values*

Description

Filter CGM values

Usage

```
filter_CGM(
  data_id,
  truncate_min_cgm = 40,
  truncate_max_cgm = 400,
  min_minutes_difference_cgm = 5
)
```

Arguments

`data_id`	A list with all data found for one individual. Filtering is only performed if the CGM data was cleaned and standardized.
`truncate_min_cgm`	CGM values smaller than this threshold will be truncated. Default 40 mg/dL, if NULL, no filtering is performed.
`truncate_max_cgm`	CGM values larger than this threshold will be truncated. Default 400 mg/dL, if NULL, no filtering is performed.
`min_minutes_difference_cgm`	CGM values are downsampled to this time interval (in minutes). Default 5 minutes, if NULL, no downsampling is performed.

Value

A list with all data found for one individual, filtered, if necessary.

Examples

```
path <- system.file("extdata", "tandem_source_example.zip", package = "aidR")
# Read data
data <- parse_data(id = "1", paths = path)
range(data$cgm$value)
length(data$cgm$value)

# Truncate to 40-400 mg/dL and downsample to 5-minute intervals
data_filtered <- filter_CGM(data)
range(data_filtered$cgm$value)
length(data_filtered$cgm$value)

# Truncate only, without downsampling
data_truncated <- filter_CGM(data, min_minutes_difference_cgm = NULL)
range(data_truncated$cgm$value)
length(data_truncated$cgm$value)
```

`get_cgmquantify_format` *Convert CGM data to a format compatible with the R-package cgmquantify*

Description

Builds the data frame that the metric functions of the R-package cgmquantify expect, with columns `Time`, `glucose`, `Date`, `time_of_day` and `type_of_event`. This is the same layout that `cgmquantify::readfile()` produces from a Dexcom export.

Usage

```
get_cgmquantify_format(data, id = NULL)
```

Arguments

`data`	A list with all data found for one individual (the output of `parse_data`) or for multiple individuals (the output of `merge_all`). Requires cleaned and standardized CGM data.
`id`	Character or numeric participant identifier, naming the individual to convert. Can be omitted if `data` holds the CGM data of a single individual.

Details

Note that cgmquantify always related to a single subject.

Value

A data frame with columns `Time` (`POSIXct`), `glucose` (mg/dL), `Date`, `time_of_day` and `type_of_event` (`1` above 180 mg/dL, `-1` below 70 mg/dL and `0` in between), sorted by time and compatible with the R-package cgmquantify.

Examples

```
path <- system.file("extdata", "dexcom_clarity_example.csv", package = "aidR")
data <- parse_data(id = "1", paths = path)
cgmquantify_df <- get_cgmquantify_format(data)
cgmquantify::summary_glucose(cgmquantify_df)
```

---

get_iglu_format *Convert CGM data to a format compatible with the R-package iglu*

Description

Convert CGM data to a format compatible with the R-package iglu

Usage

```
get_iglu_format(data)
```

Arguments

data	A list with all data found for one individual (the output of parse_data) or for multiple individuals (the output of merge_all). Requires cleaned and standardized CGM data.

Value

A data frame with columns id, time and gl, compatible with the R-package iglu.

Examples

```
path <- system.file("extdata", "dexcom_clarity_example.csv", package = "aidR")
data <- parse_data(id = "1", paths = path)
iglu_df <- get_iglu_format(data)
iglu::plot_glu(iglu_df)
```

---

merge_all *Merge data available for all individuals, separately per data type*

Description

Merge data available for all individuals, separately per data type

Usage

```
merge_all(
  data,
  types = c("cgm", "basal", "bolus", "carbs", "SMBG", "total_basal", "total_bolus")
)
```

Arguments

data	A named list with the data of each individual (the output of parse_data), named by participant identifier.
types	A character vector with the data type names to be merged. Defaults to the five standardized types and the two daily insulin totals.

Value

A named list with one data frame per data type, holding the data of all individuals. Individuals without data of a given type contribute no rows.

Examples

```
all_data <- list(
  "1" = parse_data("1", system.file("extdata", "glooko_example.zip", package = "aidR")),
 "2" = parse_data("2", system.file("extdata", "dexcom_clarity_example.csv", package = "aidR"))
)
combined <- merge_all(all_data)
names(combined)
head(combined$cgm)
table(combined$cgm$id)

# Merge CGM data only
merge_all(all_data, types = "cgm")
```

---

parse_data *Parse data files for a given participant ID*

Description

Iterates over one or more paths, discovers all contained files (including inside ZIP archives), and dispatches each file to the appropriate format-specific reader. Supported formats are Glooko, CareLink, mylife, Tidepool, YourLoops, Tandem Source and Dexcom Clarity.

Usage

```
parse_data(id, paths, clean = TRUE, ...)
```

Arguments

id	Character or numeric participant identifier. Used for logging and stored in the id column of the cleaned data.
paths	Character vector. One or more file or directory paths to search for data files. Directories are traversed recursively; ZIP archives are extracted automatically.
clean	Logical, if TRUE (the default), the main data types (CGM, basal, bolus, carbohydrates and SMBG) are cleaned and standardized. If FALSE, all data is returned as found in the export.

... Additional arguments passed on to the format-specific cleaning methods, e.g. `merge_slow_carbs` for Glooko (see `clean.glooko`). Arguments that do not apply to a format are skipped. Ignored if `clean = FALSE`.

Value

A list with all data found for one individual, merged per data type and sorted by `timestamp`. Duplicate rows are removed. Rows removed during cleaning are stored in the attribute `removed`, see `removed_entries`. `NULL` if no valid data files were found.

Examples

```
path <- system.file("extdata", "glooko_example.zip", package = "aidR")

# Raw data, as found in the export
data_raw <- parse_data(id = "1", paths = path, clean = FALSE)
names(data_raw)
head(data_raw$cgm)

# Cleaned and standardized data
data <- parse_data(id = "1", paths = path)
names(data)
head(data$cgm)
```

---

plot_day *Plot one or more days of diabetes data for a single subject*

Description

Draws a two-panel plot per day. The upper panel shows the CGM trace over a shaded target range, with the SMBG measurements as diamonds; the lower panel shows the basal rate as a filled step area, the boluses as labelled circles and the carbohydrates as labelled boxes, headed by the insulin delivered over the whole day. Both panels share a 0-24 h axis of the local (wall-clock) time, and a legend is drawn underneath them.

Usage

```
plot_day(
  data,
  id = NULL,
  days = NULL,
  target = NULL,
  ylim_cgm = NULL,
  min_bolus = 0,
  units = "mg/dl",
  main = NULL
)
```

Arguments

data	A list with all data found for one individual (the output of parse_data) or for multiple individuals (the output of merge_all). Requires cleaned and standardized data. The data types cgm, basal, bolus, carbs and SMBG are drawn where they are present; a type that is absent or that holds no data for a day is skipped.
id	Character or numeric participant identifier, naming the individual to plot. Can be omitted if data holds the data of a single individual.
days	The day(s) to plot, as a Date or as a "YYYY-MM-DD" string. Defaults to every day with CGM data, i.e. one plot per day.
target	Numeric vector of length two with the glucose target range, drawn as a shaded band.
ylim_cgm	Numeric vector of length two with the limits of the glucose axis.
min_bolus	Numeric, the smallest bolus that is drawn in full, in units. Small correction boluses crowd the panel without adding much, and min_bolus = 1 draws everything below 1 U as a bare tick at its time, without a circle and without its value. Compared against the total of a bolus, so that a small immediate part followed by a large extended one is drawn in full. Defaults to 0, i.e. every bolus is drawn in full.
units	Character string, the units of the glucose data, either "mg/dl" or "mmol/l".
main	Character string with the plot title. Defaults to the id and the day. Note that the same title is used for every day if several are plotted.

Details

The daily totals are taken from total_basal and total_bolus where the export provides them, and are summed over the day otherwise. They therefore cover insulin that the drawn data may not, such as the basal delivered in automated mode by an Omnipod, and they are unaffected by min_bolus. A total that is not recorded at all is shown as "-". Where min_bolus leaves boluses out of the labelled circles, a note next to the totals says so.

Value

No return value, called for side effects: one plot is drawn per day.

Examples

```
path <- system.file("extdata", "glooko_example.zip", package = "aidR")
data <- parse_data(id = "1", paths = path)
plot_day(data, days = "2026-06-22")

# Custom target range, with boluses below 1 U drawn as ticks only
plot_day(data, days = "2026-06-22", target = c(70, 140), min_bolus = 1)
```

removed_entries *Get all entries removed during cleaning and filtering*

Description

Rows can be removed while cleaning (e.g. merged slowly absorbed carbohydrates in `clean.glooko`) and by filters (e.g. overwritten carbohydrate entries in `filter_carbs`).

Usage

```
removed_entries(data_id)
```

Arguments

`data_id`	A list with all data found for one individual.

Value

A named list with one data frame per data type, holding the removed rows and the column `removed_reason`. `NULL` if nothing was removed.

Examples

```
path <- system.file("extdata", "glooko_example.zip", package = "aidR")
data <- filter_carbs(parse_data(id = "1", paths = path))
removed_entries(data)
```

write_DIAX_format *Write AID data (CGM, basal, bolus, carbs and SMBG, where available) to a JSON file in DIAX format*

Description

Accepts either the output of `parse_data` for a single individual or the output of `merge_all` for multiple individuals. One JSON file following the DIAX specification is written per individual.

Usage

```
write_DIAX_format(data, prefix, ids = NULL)
```

Arguments

`data`	A list with data of one individual (output of `parse_data()`) or of multiple individuals (output of `merge_all()`).
`prefix`	Character string prepended to the id to build each output filename, e.g. `prefix = "output/id_"` writes `"output/id_1.json"` for id `"1"`.
`ids`	Character or numeric vector with the individual(s) to write. Defaults to all ids found in `data`.

Value

No return value, called for side effects: one JSON file is written per id.

Examples

```
path <- system.file("extdata", "glooko_example.zip", package = "aidR")
data <- parse_data(id = "1", paths = path)

out_dir <- tempfile("diax_")
dir.create(out_dir)
write_DIAX_format(data, prefix = file.path(out_dir, "id_"))
list.files(out_dir)

unlink(out_dir, recursive = TRUE)
```

Index

check_completeness, 2
check_completeness_range, 3
check_insulin_totals, 4
clean.glooko, *5*, *10*, *12*

filter_carbs, 5, *12*
filter_CGM, 6

get_cgmquantify_format, 7
get_iglu_format, 8

merge_all, *2–4*, *7*, *8*, 8, *11*, *12*

parse_data, *2–4*, *7–9*, 9, *11*, *12*
plot_day, 10

removed_entries, *10*, 12

write_DIAX_format, 12