# Package 'chms'

October 17, 2026

**Title** Accelerometer Processing Methods for Cycle 7 of the CHMS

**Version** 7.2

**Description** 'ActiGraph wGT3X-BT' accelerometer processing methods using the standardized workflow developed by Statistics Canada for cycle 7 of the Canadian Health Measures Survey (CHMS). The package promotes transparent and reproducible data processing while supporting the harmonization of analytical approaches among researchers wishing to align with Statistics Canada's methods. For general details about the processing methods, please consult Clarke J, Gribbon A, St-Laurent M, Ferrao T, Barnes J, Kuzik N, Colley R (2026) <doi:10.25318/82-003-x202600200001-eng>.

**License** MIT + file LICENSE

**URL** https://github.com/statcan/chms

**BugReports** https://github.com/statcan/chms/issues

**Depends** R (>= 4.1.0)

**Config/roxygen2/version** 8.1.0

**Encoding** UTF-8

**Imports** cli, DBI, dbplyr, dplyr, haven, hms, jsonlite, knitr, lubridate, mirai, parallelly, purrr, R6, readr, rlang, RSQLite, stats, stringr, tidyr, utils, zoo

**Suggests** config, ggplot2, janitor, kableExtra, PhysicalActivity, quarto, scales, testthat, tibble

**NeedsCompilation** no

**Author** Joel Barnes [aut, cre], Janine Clarke [ctb], Rachel Colley [ctb], His Majesty the King in Right of Canada, as represented by Statistics Canada [cph]

**Maintainer** Joel Barnes <joel.barnes@statcan.gc.ca>

**Repository** CRAN

**Date/Publication** 2026-10-17 14:30:02 UTC

# Contents

agd . . . 2
agd_worker . . . 6
apply_barreira_algo . . . 9
apply_general_bout_algo . . . 10
apply_non_wear_algo . . . 12
apply_sadeh_algo . . . 14
apply_tudor_locke_algo . . . 15
classify_agd_data . . . 16
clean_agd_data . . . 17
get_chms_meta . . . 18
load_agd_data . . . 19
load_agd_settings . . . 21
plot.agd . . . 21
plot.agd_worker . . . 22
run_agd_job . . . 23
summarize_agd_data . . . 24
summary.agd . . . 25
summary.agd_worker . . . 27

**Index** **28**

---

`agd` *R6 class:* `agd`

---

## Description

`agd` is an R6 class that runs a data processing pipeline on one or more jobs that include two `.agd` (ActiGraph; github.com/actigraph) accelerometer files with `LowFrequencyExtension` and `Normal` filters per participant. This class makes calls to the agd_worker R6 class.

## Public fields

`args` A list of arguments passed into `agd$new()`. See documentation for `agd$new()` for details.

`log` A tibble with `method`, `timestamp`, `status`, and `message` vectors providing a record of events during the pipeline run.

`jobs` A tibble with vectors based on the `args` passed into `agd$new()` representing jobs to be run that are distributed across the number of CPUs set by `cpu_max` in `agd$new()`.

`results` A list of five tibbles (`summary_full`, `summary_full_stc`, `summary_run`, `summary_sleeping_hours`, `summary_waking_hours`) summarizing pipeline run(s).

## Methods

### Public methods:

- agd$new()
- agd$run()
- agd$export()
- agd$print()
- agd$sanity_check()
- agd$view()
- agd$clone()

`agd$new()`: This method creates an instance of an `agd_worker` class.

*Usage:*

```
agd$new(
  id,
  age,
  agd_lfe,
  agd_nml,
  epoch_length = 60,
  day_max = 7,
  sleep_algo = "barreira",
  non_wear_algo = "barreira",
  start_date = NA,
  cpu_max = 1,
  dir = NA
)
```

*Arguments:*

`id` Required: a vector representing unique participant ID(s).

`age` Required: an integer vector representing participant age(s) in years.

`agd_lfe` Required: a character vector representing the full path to `.agd` file(s) with the `LowFrequencyExtension` filter.

`agd_nml` Required: a character vector representing the full path to `.agd` file(s) with the `Normal` filter.

`epoch_length` Required (default: `60`): an integer vector (length-one or the same length as `id`) representing the epoch length(s) at which to process the data. Statistics Canada currently uses `15` for participants under 18 years and `60` for participants at all other ages. Note: if `sleep_algo` is set to `"barreira"` and `epoch_length` is set to `15`, the sleep algorithm will be applied to 60-second epoch data and the results will be applied to the 15-second epoch data.

`day_max` Required (default: `7`): an integer vector (length-one or the same length as `id`) representing the maximum number of days of data to load from `agd_lfe` and `agd_nml`.

`sleep_algo` Required (default: `"barreira"`): a character vector (length-one or the same length as `id`) representing the sleep algorithm to apply. Options currently include `"barreira"`. See apply_barreira_algo() for more details.

`non_wear_algo` Required (default: `"barreira"`): a character vector (length-one or the same length as `id`) representing the non-wear algorithm to apply. Options currently include

"barreira", "20-min-algo", "60-min-algo", "90-min-algo" and "choi". See apply_barreira_algo() and apply_non_wear_algo() for more details.

start_date Optional (default: NA): a character or date vector (format: yyyy-mm-dd) that is length-one or the same length as id representing the first day of data to load from agd_lfe and agd_nml. If not set, data will be loaded from the first available day until day_max is reached.

cpu_max Required (default: 1): a length-one integer vector representing the number of CPUs to distribute the data processing across.

dir Optional (default: NA): a length-one character vector representing the full path to the location where the results list will be exported tibble by tibble in .csv format.

*Returns:* Returns an object of class agd.

agd$run(): This method iterates jobs and calls run_agd_job().

*Usage:*

```
agd$run()
```

*Returns:* Returns the results list from an agd_worker object and binds it to self (an instance of agd).

agd$export(): This method exports self$results tibble by tibble in .csv format.

*Usage:*

```
agd$export(dir = self$args$dir, stc = FALSE)
```

*Arguments:*

dir Optional (default: self$args$dir): a length-one character vector representing the full path to the location where the results list will be exported tibble by tibble in .csv format.

stc Optional (default: FALSE): a length-one logical vector indicating whether to export only statcan-formatted results.

*Returns:* Returns the agd object (self) invisibly.

agd$print(): This method renders details (arguments, issues, log) about an instance agd to the console.

*Usage:*

```
agd$print()
```

*Returns:* Returns the agd object (self) invisibly.

agd$sanity_check(): This method renders a sanity check report in .html format.

*Usage:*

```
agd$sanity_check(
  name = "sanity-check-report",
  id = self$jobs$id,
  dir = self$args$dir,
  include_plot = FALSE
)
```

*Arguments:*

`name` Required (default: `"sanity-check-report"`): a length-one character vector representing the file name of the report.

`id` Required (default: `self$jobs$id`): a vector representing unique participant ID(s).

`dir` Required (default: `self$args$dir`): a length-one character vector representing the full path to the location where the report will be exported in `.html` format.

`include_plot` Optional (default: `FALSE`): a length-one logical vector representing whether to render scatterplots.

*Returns:* Returns an .html-formatted report.

`agd$view()`: This method renders data frames from `$results` in tab.

*Usage:*

```
agd$view(results = c(names(self$results), "issues", "log"))
```

*Arguments:*

`results` Optional (default: `c(names(self$results), "issues", "log")`): a character vector representing the results to render to tab. Options include: `"summary_full"`, `"summary_full_stc"`, `"summary_run"`, `"summary_sleeping_hours"`, `"summary_waking_hours"`, `"issues"`, `"log"`.

*Returns:* Returns the agd object (`self`) invisibly.

`agd$clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
agd$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## Examples

```
# Create meta data frame (external/non-statcan users)
meta <- data.frame(
  id = c("jane-canuck", "john-canuck"),
  age = c(10, 40),
  agd_lfe = c(
    system.file("extdata", "jane-canuck-lfe.agd", package = "chms"),
    system.file("extdata", "john-canuck-lfe.agd", package = "chms")
  ),
  agd_nml = c(
    system.file("extdata", "jane-canuck-nml.agd", package = "chms"),
    system.file("extdata", "john-canuck-nml.agd", package = "chms")
  ),
  start_date = c("2021-05-30", "2021-05-27"),
  epoch_length = c(15, 60)
)

# Initialize agd R6 class
agd_data <- agd$new(
  id = meta$id,
  age = meta$age,
  agd_lfe = meta$agd_lfe,
```

```
  agd_nml = meta$agd_nml,
  epoch_length = meta$epoch_length,
  day_max = 2,
  sleep_algo = "barreira",
  non_wear_algo = "barreira",
  start_date = meta$start_date,
  cpu_max = 1
)

# Run data processing pipeline (load, clean, classify and summarize data)
agd_data$run()
```

---

agd_worker *R6 class:* agd_worker

---

## Description

agd_worker is an R6 class that runs a data processing pipeline on two .agd (ActiGraph; github.com/actigraph) accelerometer files with LowFrequencyExtension and Normal filters for a single participant. This class does all the heavy lifting for the agd R6 class.

## Public fields

`args` A list of arguments passed into `agd_worker$new()`. See documentation for `agd_worker$new()` for details.

`data` A list of four tibbles created during `agd_worker$load()` (`settings`, `raw`), `agd_worker$clean()` (`clean`) and `agd_worker$classify()` (`classify`) representing `.agd` data at various stages of the pipeline.

`issues` A list of seven length-one, dichotomous (`"yes"/""`) character vectors (`file_missing`, `files_identical`, `files_mismatched`, `non_midnight_start`, `no_complete_days`, `sleep_missing`, `age_out_of_range`) representing flags for issues affecting pipeline results.

`log` A tibble with `method`, `timestamp`, `status` and `message` vectors providing a record of events during the pipeline run.

`results` A list of four tibbles (`summary_full`, `summary_full_stc`, `summary_sleeping_hours`, `summary_waking_hours`) summarizing the pipeline run.

## Methods

### Public methods:

- `agd_worker$new()`
- `agd_worker$load()`
- `agd_worker$clean()`
- `agd_worker$classify()`
- `agd_worker$summarize()`
- `agd_worker$run()`
- `agd_worker$print()`

- agd_worker$clone()

`agd_worker$new()`: This method creates an instance of an `agd_worker` class.

*Usage:*

```
agd_worker$new(
  id,
  age,
  agd_lfe,
  agd_nml,
  epoch_length = 60,
  day_max = 7,
  sleep_algo = "barreira",
  non_wear_algo = "barreira",
  start_date = NA
)
```

*Arguments:*

`id` Required: a length-one vector representing a unique participant ID.

`age` Required: a length-one integer vector representing a participant's age in years.

`agd_lfe` Required: a length-one character vector representing the full path to an `.agd` file with the `LowFrequencyExtension` filter.

`agd_nml` Required: a length-one character vector representing the full path to an `.agd` file with the `Normal` filter.

`epoch_length` Required (default: `60`): a length-one integer vector representing the epoch length at which to process the data. Statistics Canada currently uses `15` for participants under 18 years and `60` for participants at all other ages. Note: if `sleep_algo` is set to `"barreira"` and `epoch_length` is set to `15`, the sleep algorithm will be applied to 60-second epoch data and the results will be applied to the 15-second epoch data.

`day_max` Required (default: `7`): a length-one integer vector representing the maximum number of days of data to load from `agd_lfe` and `agd_nml`.

`sleep_algo` Required (default: `"barreira"`): a length-one character vector representing the sleep algorithm to apply. Options currently include `"barreira"`. See `apply_barreira_algo()` for more details.

`non_wear_algo` Required (default: `"barreira"`): a length-one character vector representing the non-wear algorithm to apply. Options currently include `"barreira"`, `"20-min-algo"`, `"60-min-algo"`, `"90-min-algo"` and `"choi"`. See `apply_barreira_algo()` and `apply_non_wear_algo()` for more details.

`start_date` Optional (default: `NA`): a length-one date vector (format: yyyy-mm-dd) representing the first day of data to load from `agd_lfe` and `agd_nml`. If not set, data will be loaded from the first available day until `day_max` is reached.

*Returns:* Returns an object of class `agd_worker`.

`agd_worker$load()`: This method calls `load_agd_settings()` and `load_agd_data()`.

*Usage:*

```
agd_worker$load()
```

*Returns:* Returns the `agd_worker` object (`self`) invisibly.

`agd_worker$clean()`: This method calls `clean_agd_data()`.

*Usage:*

```
agd_worker$clean()
```

*Returns:* Returns the `agd_worker` object (`self`) invisibly.

`agd_worker$classify()`: This method calls `classify_agd_data()`.

*Usage:*

```
agd_worker$classify()
```

*Returns:* Returns the `agd_worker` object (`self`) invisibly.

`agd_worker$summarize()`: This method calls `summarize_agd_data()`.

*Usage:*

```
agd_worker$summarize()
```

*Returns:* Returns the `agd_worker` object (`self`) invisibly.

`agd_worker$run()`: This method calls `agd_worker$load()`, `agd_worker$clean()`, `agd_worker$classify()` and `agd_worker$summarize()`.

*Usage:*

```
agd_worker$run()
```

*Returns:* Returns the `agd_worker` object (`self`) invisibly.

`agd_worker$print()`: This method renders details (arguments, issues, log) about an instance of `agd_worker` to the console.

*Usage:*

```
agd_worker$print()
```

*Returns:* Returns the `agd_worker` object (`self`) invisibly.

`agd_worker$clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
agd_worker$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## Examples

```
# Initialize agd_worker R6 class
agd_data <- agd_worker$new(
  id = "jane-canuck",
  age = 10,
  agd_lfe = system.file("extdata", "jane-canuck-lfe.agd", package = "chms"),
  agd_nml = system.file("extdata", "jane-canuck-nml.agd", package = "chms"),
  epoch_length = 15,
  day_max = 3,
  sleep_algo = "barreira",
  non_wear_algo = "barreira"
)

# Run data processing pipeline (load, clean, classify and summarize data)
agd_data$run()
```

`apply_barreira_algo` *Classify an accelerometer axis vector of 60-second epochs as sleep time or awake time.*

## Description

This function uses the Barreira algorithm (pubmed.ncbi.nlm.nih.gov/25202840) to classify an accelerometer axis vector of 60-second epochs as sleep time or awake time. This function was validated against output from the official SAS version of the algorithm (www.pbrc.edu/pdf/PBRCSleepEpisodeTimeMacroCode.p

## Usage

```
apply_barreira_algo(
  x,
  time_stamp = "dataTimestamp",
  axis1 = "axis1",
  incline_off = "inclineOff",
  incline_standing = "inclineStanding",
  incline_sitting = "inclineSitting",
  incline_lying = "inclineLying",
  age,
  return = "everything"
)
```

## Arguments

| | |
|---|---|
| `x` | Required: a data frame of accelerometer data in 60-second epochs. |
| `time_stamp` | Required (default "`dateTimestamp`"): a length-one character vector representing the name of the timestamp vector. |
| `axis1` | Required (default: "`axis1`"): a length-one character vector representing the name of the vertical axis. |
| `incline_off` | Required (default: "`inclineOff`"): a length-one character vector representing the name of the corresponding vector in an ActiGraph .agd file. |
| `incline_standing` | Required (default: "`inclineStanding`"): a length-one character vector representing the name of the corresponding vector in an ActiGraph .agd file. |
| `incline_sitting` | Required (default: "`inclineSitting`"): a length-one character vector representing the name of the corresponding vector in an ActiGraph .agd file. |
| `incline_lying` | Required (default: "`inclineLying`"): a length-one character vector representing the name of the corresponding vector in an ActiGraph .agd file. |
| `age` | Required: a length-one integer vector representing the participant's age in years. This parameter is used to determine when the first sleep bout can begin. Statistics Canada sets the time to 18:00 for participants under five years and younger, and 19:00 for all other ages. |

`return` Required (default: "everything"): a character vector representing which vectors in x to return. If set to "everything", the data frame in x will be returned along with all vectors that were derived while applying the Barreira algorithm.

## Value

Returns the data frame x along with all vectors that were derived while applying the Barreira algorithm.

## Examples

```
# Initialize agd_worker R6 class
agd_data <- agd_worker$new(
  id = "jane-canuck",
  age = 10,
  agd_lfe = system.file("extdata", "jane-canuck-lfe.agd", package = "chms"),
  agd_nml = system.file("extdata", "jane-canuck-nml.agd", package = "chms"),
  epoch_length = 15,
  day_max = 3,
  sleep_algo = "barreira",
  non_wear_algo = "barreira"
)

# Load and clean data
agd_data$load()$clean()


# Apply Barreira sleep algorithm
dt <- apply_barreira_algo(
  x = agd_data$data$clean |>
    dplyr::filter(
      epoch_length == 60,
      filter == "LowFrequencyExtension"
    ),
  age = agd_data$args$age
)
```

---

`apply_general_bout_algo` *Classify an accelerometer axis vector of epochs based on a set of bout rules.*

---

## Description

This function uses a general algorithm to classify an accelerometer axis vector of of epochs based on a set of bout rules.

## Usage

```
apply_general_bout_algo(
  x,
  axis1 = "axis1",
  is_wearing = "is_wearing",
  min_bout_length = 10,
  target_values,
  max_exceptions = 2,
  return = "everything"
)
```

## Arguments

| | |
|---|---|
| x | Required: a data frame of accelerometer data. |
| axis1 | Required: a length-one character vector representing the name of the vertical axis (default: "axis1"). |
| is_wearing | Required: a character vector of "Yes" and "No" values representing wear time and non-wear time respectively (default: "is_wearing"). See apply_non_wear_algo() for more details. |
| min_bout_length | Required: a length-one numeric vector representing the minimum number of epochs in a bout (default: 10). |
| target_values | Required: a numeric vector representing the epoch value(s) that belong to a bout. |
| max_exceptions | Required: a length-one numeric vector representing the maximum number of epochs permitted within a bout that is outside the range of the values in the target_values argument (default: 2). |
| return | Required: a character vector representing which vectors to return (default: "everything"). If set to "everything", the data frame in the x argument will be returned along with all vectors that were derived while applying the bout algorithm. |

## Value

Returns the data frame in the x argument along with all vectors that were derived while applying the bout algorithm.

## Examples

```
# Initialize agd_worker R6 class
agd_data <- agd_worker$new(
  id = "jane-canuck",
  age = 10,
  agd_lfe = system.file("extdata", "jane-canuck-lfe.agd", package = "chms"),
  agd_nml = system.file("extdata", "jane-canuck-nml.agd", package = "chms"),
  epoch_length = 15,
  day_max = 3,
  sleep_algo = "barreira",
```

```
  non_wear_algo = "barreira"
)

# Load, clean and classify data
agd_data$load()$clean()$classify()

# Look for 10+ minute bouts of moderate-intensity physical activity
dt <- apply_general_bout_algo(
  x = agd_data$data$classify,
  min_bout_length = 10 * (60 / agd_data$args$movement_epoch_length),
  target_values = 574:1002,
  max_exceptions = 2 * (60 / agd_data$args$movement_epoch_length),
  return = c("dataTimestamp", "axis1", "in_bout")
)

# Get bout lengths
rle(dt$in_bout)
```

---

`apply_non_wear_algo` *Classify an accelerometer axis vector of epochs as wear time or non-wear time based on a set of bout rules.*

---

## Description

This function uses a general algorithm to classify an accelerometer axis vector of epochs as wear time or non-wear time based on a set of bout rules

## Usage

```
apply_non_wear_algo(
  x,
  axis1 = "axis1",
  min_bout_length = 90,
  target_values,
  max_exceptions = 2,
  return = "everything"
)
```

## Arguments

| | |
|---|---|
| `x` | Required: a data frame of accelerometer data. |
| `axis1` | Required: a length-one character vector representing the name of the vertical axis (default: "axis1"). |
| `min_bout_length` | Required: a length-one numeric vector representing the minimum number of epochs in a bout (default: 10). |
| `target_values` | Required: a numeric vector representing the epoch value(s) that belong to a bout. |

`max_exceptions` Required: a length-one numeric vector representing the maximum number of epochs permitted within a bout that outside the range of the values in the `target_values` argument (default: 2).

`return` Required: a character vector representing which vectors to return (default: "everything"). If set to "everything", the data frame in the `x` argument will be returned along with all vectors that were derived while applying the bout algorithm.

## Value

Returns the data frame in the `x` argument along with all vectors that were derived while applying the bout algorithm.

## Examples

```
# Initialize agd_worker R6 class
agd_data <- agd_worker$new(
  id = "jane-canuck",
  age = 10,
  agd_lfe = system.file("extdata", "jane-canuck-lfe.agd", package = "chms"),
  agd_nml = system.file("extdata", "jane-canuck-nml.agd", package = "chms"),
  epoch_length = 15,
  day_max = 3,
  sleep_algo = "barreira",
  non_wear_algo = "barreira"
)

# Load and clean data
agd_data$load()$clean()

# Apply Barreira sleep algorithm
dt <- apply_barreira_algo(
  x = agd_data$data$clean |>
    dplyr::filter(
      epoch_length == 60,
      filter == "LowFrequencyExtension"
    ),
  age = agd_data$args$age
)

# Apply general non-wear algorithm
dt <- apply_non_wear_algo(
  x = dt,
  min_bout_length = 60,
  target_values = 0,
  max_exceptions = 2,
  return = c("is_wearing", "is_sleeping", "ymd_hm")
)
```

`apply_sadeh_algo` *Classify an accelerometer axis vector of 60-second epochs as sleep time or awake time.*

## Description

This function uses the Sadeh algorithm (https://pubmed.ncbi.nlm.nih.gov/7939118) to classify an accelerometer axis vector of 60-second epochs as sleep time or awake time.

## Usage

```
apply_sadeh_algo(
  x,
  axis1 = "axis1",
  censor_counts = FALSE,
  return = "everything"
)
```

## Arguments

| | |
|---|---|
| `x` | Required: a data frame of accelerometer data. |
| `axis1` | Required default (`"axis1"`): a length-one character vector representing the name of the vertical axis. |
| `censor_counts` | Optional (default: `FALSE`): a length-one logical vector representing whether to censor accelerometer counts to a maximum of 300 before applying the Sadeh algorithm. |
| `return` | Required (default: `"everything"`): a character vector representing which vectors to return. If set to "everything", the data frame in `x` will be returned along with all vectors that were derived while applying the Sadeh algorithm. |

## Value

Returns `x` along with all vectors that were derived while applying the Sadeh algorithm. Note: values in the `sadeh_sleep_score` vector that are greater than -4 are interpreted as sleep time (see https://actigraphcorp.my.site.com/support/s/article/Where-can-I-find-documentation-for-the-Sadeh-and-Cole-Kripke-algorithms). # nolint: line_length_linter.

## Examples

```
# Initialize agd_worker R6 class
agd_data <- agd_worker$new(
  id = "jane-canuck",
  age = 10,
  agd_lfe = system.file("extdata", "jane-canuck-lfe.agd", package = "chms"),
  agd_nml = system.file("extdata", "jane-canuck-nml.agd", package = "chms"),
  epoch_length = 15,
  day_max = 3,
```

```
  sleep_algo = "barreira",
  non_wear_algo = "barreira"
)

# Load and clean data
agd_data$load()$clean()

# Apply Sadeh sleep algorithm
dt <- apply_sadeh_algo(
  x = agd_data$data$clean |>
    dplyr::filter(
      epoch_length == 60,
      filter == "LowFrequencyExtension"
    )
)
```

---

`apply_tudor_locke_algo` *Classify an accelerometer axis vector of 60-second epochs as sleep time or awake time.*

---

## Description

This function uses the Tudor-Locke algorithm (https://pubmed.ncbi.nlm.nih.gov/24383507) to classify an accelerometer axis vector of 60-second epochs as sleep time or awake time. This function was validated indirectly against output from the official SAS version of the Barreira algorithm (www.pbrc.edu/pdf/PBRCSleepEpisodeTimeMacroCode.pdf).

## Usage

```
apply_tudor_locke_algo(
  x,
  time_stamp = "dataTimestamp",
  axis1 = "axis1",
  incline_off = "inclineOff",
  incline_standing = "inclineStanding",
  incline_sitting = "inclineSitting",
  incline_lying = "inclineLying",
  return = "everything"
)
```

## Arguments

| | |
|---|---|
| `x` | Required: a data frame of accelerometer data in 60-second epochs. |
| `time_stamp` | Required (default: `"dataTimestamp"`): a length-one character vector representing the name of the time stamp vector. |
| `axis1` | Required (default: `"axis1"`): a length-one character vector representing the name of the vertical axis. |

incline_off Required (default: "inclineOff"): a length-one character vector representing the name of the corresponding vector in an ActiGraph .agd file.

incline_standing Required (default: "inclineStanding"): a length-one character vector representing the name of the corresponding vector in an ActiGraph .agd file.

incline_sitting Required (default: "inclineSitting"): a length-one character vector representing the name of the corresponding vector in an ActiGraph .agd file.

incline_lying Required (default: "inclineLying"): a length-one character vector representing the name of the corresponding vector in an ActiGraph .agd file.

return Required (default: "everything"): a character vector representing which vectors to return. If set to "everything", the data frame in x will be returned along with all vectors that were derived while applying the Tudor-Locke algorithm.

## Value

Returns x along with all vectors that were derived while applying the Tudor-Locke algorithm.

## Examples

```
# Initialize agd_worker R6 class
agd_data <- agd_worker$new(
  id = "jane-canuck",
  age = 10,
  agd_lfe = system.file("extdata", "jane-canuck-lfe.agd", package = "chms"),
  agd_nml = system.file("extdata", "jane-canuck-nml.agd", package = "chms"),
  epoch_length = 15,
  day_max = 3,
  sleep_algo = "barreira",
  non_wear_algo = "barreira"
)

# Load and clean data
agd_data$load()$clean()

# Apply Tudor-Locke sleep algorithm
dt <- apply_tudor_locke_algo(
  x = agd_data$data$clean |>
    dplyr::filter(
      epoch_length == 60,
      filter == "LowFrequencyExtension"
    )
)
```

---

# classify_agd_data *Classify epochs in an ActiGraph .agd file.*

---

## Description

This function classifies epochs in an ActiGraph .agd file.

## Usage

```
classify_agd_data(x)
```

## Arguments

| | |
|---|---|
| x | Required: an agd_worker object. |

## Value

Returns NULL invisibly.

## Examples

```
# Initialize agd_worker R6 class
agd_data <- agd_worker$new(
  id = "jane-canuck",
  age = 10,
  agd_lfe = system.file("extdata", "jane-canuck-lfe.agd", package = "chms"),
  agd_nml = system.file("extdata", "jane-canuck-nml.agd", package = "chms"),
  epoch_length = 15,
  day_max = 3,
  sleep_algo = "barreira",
  non_wear_algo = "barreira"
)

# Load and clean data
agd_data$load()$clean()

# Classify data
classify_agd_data(agd_data)

# Store updated data
dt <- agd_data$data$classify
```

---

# clean_agd_data *Prepare an ActiGraph .agd file for downstream classification.*

---

## Description

This function prepares an ActiGraph .agd file for downstream classification (e.g., remove incomplete days, aggregate data to 60-second epochs).

## Usage

```
clean_agd_data(x)
```

## Arguments

| | |
|---|---|
| x | Required: an agd_worker object. |

## Value

Returns a NULL invisibly.

## Examples

```
# Initialize agd_worker R6 class
agd_data <- agd_worker$new(
  id = "jane-canuck",
  age = 10,
  agd_lfe = system.file("extdata", "jane-canuck-lfe.agd", package = "chms"),
  agd_nml = system.file("extdata", "jane-canuck-nml.agd", package = "chms"),
  epoch_length = 15,
  day_max = 3,
  sleep_algo = "barreira",
  non_wear_algo = "barreira"
)

# Load data
agd_data$load()

# Clean data
clean_agd_data(agd_data)

# Store updated data
dt <- agd_data$data$clean
```

---

# get_chms_meta *Gets CHMS participant metadata.*

---

## Description

This function gets CHMS participant metadata.

## Usage

```
get_chms_meta(
  clinic_file,
  agd_dir,
  clinic_id = "CLINICID",
  site = "SITE",
  age = "CLC_AGE",
  day = "V2_DAY",
  month = "V2_MTH",
  year = "V2_YEAR"
)
```

### Arguments

| | |
|---|---|
| clinic_file | Required: a length-one character vector representing the full path to the clinic file. Note: a .sas7bdat file is expected. |
| agd_dir | Required: a character vector representing the full path(s) to the site directory ("full/path/to/data/site"). |
| clinic_id | Required (default: "CLINICID"): a length-one character vector representing the clinic ID vector name in clinic_file. |
| site | Required (default: "SITE"): a length-one character vector representing the site vector name in clinic_file. |
| age | Required (default: "CLC_AGE"): a length-one character vector representing the age vector name in clinic_file. |
| day | Required (default: "V2_DAY"): a length-one character vector representing the day vector name of the MEC visit date in clinic_file. |
| month | Required (default: "V2_MTH"): a length-one character vector representing the month vector name of the MEC visit date in clinic_file. |
| year | Required (default: "V2_YEAR"): a length-one character vector representing the year vector name of the MEC visit date in clinic_file. |

### Value

Returns a tibble with eight vectors (id, age, site, agd_lfe, agd_nml, mec_visit_date, start_date, epoch_length).

### Examples

```
# Create participant meta (statcan users)
meta <- get_chms_meta(
  clinic_file = "path/to/clinic/file.sas7bdat",
  agd_dir = "path/to/agd/files/site",
  clinic_id = "CLINICID",
  site = "SITE",
  age = "CLC_AGE",
  day = "V2_DAY",
  month = "V2_MTH",
  year = "V2_YEAR"
)
```

---

## load_agd_data *Load the* data *table from an ActiGraph* .agd *file.*

---

### Description

This function loads the data table from an ActiGraph .agd file.

## Usage

```
load_agd_data(
  file,
  col_select = "everything",
  day_max = 7,
  start_date,
  settings
)
```

## Arguments

| | |
|---|---|
| file | Required: a length-one character vector representing the full path to an .agd file. |
| col_select | Required (default: "everything"): a character vector representing which vectors to load. |
| day_max | Required (default: 7): a length-one integer vector representing the maximum number of days of data to load from file. |
| start_date | Optional: a length-one date vector (format: yyyy-mm-dd) representing the first day of data to load from file. If not set, data will be loaded from the first available day until day_max is reached. |
| settings | Optional: a tibble from file previously returned from load_agd_settings(). |

## Value

Returns a tibble of the data table.

## Examples

```
# Initialize agd_worker R6 class
agd_data <- agd_worker$new(
  id = "jane-canuck",
  age = 10,
  agd_lfe = system.file("extdata", "jane-canuck-lfe.agd", package = "chms"),
  agd_nml = system.file("extdata", "jane-canuck-nml.agd", package = "chms"),
  epoch_length = 15,
  day_max = 3,
  sleep_algo = "barreira",
  non_wear_algo = "barreira"
)

# Load data with low frequency extension
dt <- load_agd_data(
  file = agd_data$args$agd_lfe,
  start_date = agd_data$args$start_date
)
```

## load_agd_settings *Load the* settings *table from an ActiGraph* .agd *file.*

### Description

This function loads the `settings` table from an ActiGraph `.agd` file.

### Usage

```
load_agd_settings(file)
```

### Arguments

| | |
|---|---|
| `file` | Required: a length-one character vector representing the full path to an `.agd` file. |

### Value

Returns a tibble of the `settings` table in a wide shape.

### Examples

```
# Initialize agd_worker R6 class
agd_data <- agd_worker$new(
  id = "jane-canuck",
  age = 10,
  agd_lfe = system.file("extdata", "jane-canuck-lfe.agd", package = "chms"),
  agd_nml = system.file("extdata", "jane-canuck-nml.agd", package = "chms"),
  epoch_length = 15,
  day_max = 3,
  sleep_algo = "barreira",
  non_wear_algo = "barreira"
)

# Load ActiGraph settings
dt <- load_agd_settings(agd_data$args$agd_lfe)
```

## plot.agd *Plot method for the agd R6 class.*

### Description

This method renders scatter plots iteratively and interactively using results from an agd object.

### Usage

```
## S3 method for class 'agd'
plot(x, ..., id)
```

### Arguments

| | |
|---|---|
| x | Required: an agd object. |
| ... | Optional: arguments to be passed to methods. **Note:** currently not used. |
| id | Optional: a vector representing participant IDs in x for which to render scatter plots. If id is unset, scatter plots for all participants in x will be rendered iteratively. |

### Value

Returns ggplot2 objects invisibly.

### Examples

```
# Initialize agd R6 class
agd_data <- agd$new(
  id = "jane-canuck",
  age = 10,
  agd_lfe = system.file("extdata", "jane-canuck-lfe.agd", package = "chms"),
  agd_nml = system.file("extdata", "jane-canuck-nml.agd", package = "chms"),
  epoch_length = 15,
  day_max = 2,
  sleep_algo = "barreira",
  non_wear_algo = "barreira"
)

# Run data processing pipeline (load, clean, classify and summarize data)
agd_data$run()

# Plot data
plot(agd_data)
```

---

## plot.agd_worker *Plot method for the agd_worker R6 class.*

---

### Description

This method renders a scatter plot using results from an agd_worker object.

### Usage

```
## S3 method for class 'agd_worker'
plot(x, ..., title_size = 11, axis_size = 9, label_size = 3)
```

### Arguments

| | |
|---|---|
| x | Required: an agd_worker object. |
| ... | Optional: arguments to be passed to methods. **Note:** currently not used. |
| title_size | Required (default: 11): a length-one numeric vector representing the title font size. |
| axis_size | Required (default: 9): a length-one numeric vector representing the axis font size . |
| label_size | Required (default: 3): a length-one numeric vector representing the label font size. |

### Value

Returns a ggplot2 object invisibly.

### Examples

```
# Initialize agd_worker R6 class
agd_data <- agd_worker$new(
  id = "jane-canuck",
  age = 10,
  agd_lfe = system.file("extdata", "jane-canuck-lfe.agd", package = "chms"),
  agd_nml = system.file("extdata", "jane-canuck-nml.agd", package = "chms"),
  epoch_length = 15,
  day_max = 2,
  sleep_algo = "barreira",
  non_wear_algo = "barreira"
)

# Run data processing pipeline (load, clean, classify and summarize data)
agd_data$run()

# Plot data
plot(agd_data)
```

---

## run_agd_job *Run a job that initiates an ActiGraph data processing pipeline for a single participant.*

---

### Description

This function runs a job that processes ActiGraph data for a single participant by calling agd_worker$new()$run(). This function is used heavily by the agd R6 class.

### Usage

```
run_agd_job(x)
```

## Arguments

| | |
|---|---|
| x | Required: a one-row tibble from an agd object jobs data structure. |

## Value

Returns the results list from an agd_worker object.

## Examples

```
# Create meta data frame (external/non-statcan users)
meta <- data.frame(
  id = c("jane-canuck", "john-canuck"),
  age = c(10, 40),
  agd_lfe = c(
    system.file("extdata", "jane-canuck-lfe.agd", package = "chms"),
    system.file("extdata", "john-canuck-lfe.agd", package = "chms")
  ),
  agd_nml = c(
    system.file("extdata", "jane-canuck-nml.agd", package = "chms"),
    system.file("extdata", "john-canuck-nml.agd", package = "chms")
  ),
  start_date = c("2021-05-30", "2021-05-27"),
  epoch_length = c(15, 60)
)

# Initialize agd R6 class
agd_data <- agd$new(
  id = meta$id,
  age = meta$age,
  agd_lfe = meta$agd_lfe,
  agd_nml = meta$agd_nml,
  epoch_length = meta$epoch_length,
  day_max = 3,
  sleep_algo = "barreira",
  non_wear_algo = "barreira",
  start_date = meta$start_date,
  cpu_max = 2
)

# Run data processing pipeline (load, clean, classify and summarize data)
# on first participant in agd_data
lst <- run_agd_job(agd_data$jobs[1,])
```

---

# summarize_agd_data *Summarizes an ActiGraph* .agd *file.*

---

## Description

This function summarizes an ActiGraph .agd file.

## Usage

```
summarize_agd_data(x)
```

## Arguments

| | |
|---|---|
| x | Required: an agd_worker object. |

## Value

Returns a NULL invisibly.

## Examples

```
# Initialize agd_worker R6 class
agd_data <- agd_worker$new(
  id = "jane-canuck",
  age = 10,
  agd_lfe = system.file("extdata", "jane-canuck-lfe.agd", package = "chms"),
  agd_nml = system.file("extdata", "jane-canuck-nml.agd", package = "chms"),
  epoch_length = 15,
  day_max = 3,
  sleep_algo = "barreira",
  non_wear_algo = "barreira"
)

# Load, clean and classify data
agd_data$load()$clean()$classify()

# Summarize data
summarize_agd_data(agd_data)

# Store updated data
dt <- agd_data$results$summary_full
```

---

# summary.agd *Summary method for the agd R6 class.*

---

## Description

This method renders summary tibbles for waking and sleeping hours from an agd object to the console.

## Usage

```
## S3 method for class 'agd'
summary(object, ..., row_max = 10)
```

## Arguments

| | |
|---|---|
| `object` | Required: an agd_worker object. |
| `...` | Optional: additional arguments affecting the summary produced. **Note:** currently not used. |
| `row_max` | Required (default: `10`): a length-one integer vector representing the number of rows of summary data to render to the console. |

## Value

Returns `NULL` invisibly.

## Examples

```
# Create meta data frame (external/non-statcan users)
meta <- data.frame(
  id = c("jane-canuck", "john-canuck"),
  age = c(10, 40),
  agd_lfe = c(
    system.file("extdata", "jane-canuck-lfe.agd", package = "chms"),
    system.file("extdata", "john-canuck-lfe.agd", package = "chms")
  ),
  agd_nml = c(
    system.file("extdata", "jane-canuck-nml.agd", package = "chms"),
    system.file("extdata", "john-canuck-nml.agd", package = "chms")
  ),
  start_date = c("2021-05-30", "2021-05-27"),
  epoch_length = c(15, 60)
)

# Initialize agd R6 class
agd_data <- agd$new(
  id = meta$id,
  age = meta$age,
  agd_lfe = meta$agd_lfe,
  agd_nml = meta$agd_nml,
  epoch_length = meta$epoch_length,
  day_max = 2,
  sleep_algo = "barreira",
  non_wear_algo = "barreira",
  start_date = meta$start_date,
  cpu_max = 1
)

# Run data processing pipeline (load, clean, classify and summarize data)
agd_data$run()

# Summarize data
summary(agd_data)
```

summary.agd_worker *Summary method for the agd_worker R6 class.*

## Description

This method renders summary tibbles for waking and sleeping hours from an agd_worker object to the console.

## Usage

```
## S3 method for class 'agd_worker'
summary(object, ...)
```

## Arguments

| | |
|---|---|
| object | Required: an agd_worker object. |
| ... | Optional: additional arguments affecting the summary produced. **Note:** currently not used. |

## Value

Returns NULL invisibly.

## Examples

```
# Initialize agd_worker R6 class
agd_data <- agd_worker$new(
  id = "jane-canuck",
  age = 10,
  agd_lfe = system.file("extdata", "jane-canuck-lfe.agd", package = "chms"),
  agd_nml = system.file("extdata", "jane-canuck-nml.agd", package = "chms"),
  epoch_length = 15,
  day_max = 3,
  sleep_algo = "barreira",
  non_wear_algo = "barreira"
)

# Run data processing pipeline (load, clean, classify and summarize data)
agd_data$run()

# Summarize data
summary(agd_data)
```

# Index


agd, 2, *6*, *21–25*
agd_worker, *2*, *4*, 6, *17, 18*, *22–27*
apply_barreira_algo, 9
apply_barreira_algo(), *3, 4*, *7*
apply_general_bout_algo, 10
apply_non_wear_algo, 12
apply_non_wear_algo(), *4*, *7*, *11*
apply_sadeh_algo, 14
apply_tudor_locke_algo, 15

classify_agd_data, 16
classify_agd_data(), *8*
clean_agd_data, 17
clean_agd_data(), *8*

get_chms_meta, 18

load_agd_data, 19
load_agd_data(), *7*
load_agd_settings, 21
load_agd_settings(), *7*, *20*

plot.agd, 21
plot.agd_worker, 22

run_agd_job, 23
run_agd_job(), *4*

summarize_agd_data, 24
summarize_agd_data(), *8*
summary.agd, 25
summary.agd_worker, 27