Package 'rester'

November 12, 2026

Title Estimate Sleep Stages and Respiratory Events from Accelerometry Data

Version 0.1.0

Description Wraps the 'AcceleRest' model from Lorenzen et al. (2026) <doi:10.64898/2026.01.28.26345056> <https://github.com/NielsRLorenzen/AcceleRest> to estimate sleep stages and respiratory events from wrist-worn 'accelerometry' data. Provides functions to download the model code and pretrained weights and to run the model using 'reticulate'.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 4.1.0)

SystemRequirements Python (>= 3.9, < 3.13), 'torch' Python module (>= 2.2), 'actipy' Python module (>= 3.8.0)

Imports actibase (>= 0.8.0), actiread (>= 0.8.0), assertthat, curl, dplyr, methods, readr, reticulate (>= 1.41.0), rlang, stats, tools, utils

Suggests callr, testthat (>= 3.0.0)

URL https://github.com/jhuwit/rester

BugReports https://github.com/jhuwit/rester/issues

Config/roxygen2/version 8.1.1

Config/testthat/edition 3

NeedsCompilation no

Author John Muschelli [aut, cre] (ORCID: <https://orcid.org/0000-0001-6469-1750>)

Maintainer John Muschelli <muschellij2@gmail.com>

Repository CRAN

Date/Publication 2026-11-12 19:10:02 UTC

Contents

ar_cache_dir . . . 2
ar_download_repo . . . 3
ar_example_data . . . 4
ar_load_model . . . 4
ar_load_weights . . . 5
ar_python_modules_installed . . . 6
ar_read . . . 6
ar_setup . . . 8
ar_weights . . . 9
estimate_rest . . . 10
py_estimate_rest . . . 11
py_require_rester . . . 12

Index **13**

`ar_cache_dir` `rester` *Cache Directory*

Description

The default location for the downloaded AcceleRest code and weights. It can be set using `options(rester.cache_dir = )` or the `RESTER_CACHE_DIR` environment variable, and otherwise is `tools::R_user_dir("rester", "cache")`.

Usage

```
ar_cache_dir()
```

Value

A character path. The directory is not created.

Examples

```
ar_cache_dir()
```

ar_download_repo *Download and Extract the AcceleRest Code*

Description

Downloads the source archive of https://github.com/NielsRLorenzen/AcceleRest at a fixed commit and extracts the files needed to build the models (`hubconf.py` and the `accelerest` Python package). The code is downloaded rather than bundled as the AcceleRest repository does not declare a license.

Usage

```
ar_download_repo(
  repo_dir = ar_cache_dir(),
  ref = .ar_default_ref,
  overwrite = FALSE,
  quiet = TRUE,
  ...
)

ar_extract_repo(zip_file, repo_dir = ar_cache_dir())

ar_have_repo(repo_dir = ar_cache_dir())
```

Arguments

`repo_dir`	directory to extract the code into. The code is placed in `file.path(repo_dir, "AcceleRest")`.
`ref`	git commit, tag, or branch of AcceleRest to download
`overwrite`	If `TRUE`, re-download and replace an existing copy
`quiet`	passed to `curl::curl_download()`
`...`	additional arguments to pass to `curl::curl_download()`
`zip_file`	a source archive of AcceleRest, e.g. downloaded from GitHub

Value

The path to the extracted repository. `ar_have_repo` returns a logical.

Examples

```
## Not run:
  # downloads the AcceleRest code to ar_cache_dir()
  path = ar_download_repo()

## End(Not run)
ar_have_repo()
```

ar_example_data *Example Accelerometry Data*

Description

Reads the example 12 hours of 30Hz wrist accelerometry data included in the package (the same recording as in the `sleeper` package), reconstructing the time stamps (stored rounded to the second) from the sample rate.

Usage

```
ar_example_data(n_hours = NULL)
```

Arguments

`n_hours`	number of hours of data to read (up to 12). `NULL` reads all of it.

Value

A `tibble` with columns `time`, `X`, `Y`, and `Z`, with the `sample_rate` attribute set to 30.

Examples

```
data = ar_example_data(n_hours = 0.01)
head(data)
```

ar_load_model *Load the AcceleRest Model*

Description

Builds the multi-head AcceleRest model from downloaded code and weights (see `ar_setup()`). All heads share a single encoder backbone.

Usage

```
ar_load_model(
  model = NULL,
  repo_dir = ar_cache_dir(),
  weights_dir = ar_cache_dir(),
  device = NULL
)
```

Arguments

model	the prediction heads to include, one or more of "linear_sleepstages", "lstm_sleepstages", and "linear_respevents". Defaults to all three.
repo_dir	directory with the downloaded code, see ar_download_repo()
weights_dir	directory with the downloaded weights, see ar_download_weights()
device	torch device, e.g. "cpu", "cuda", or "mps". If NULL, uses "cuda" if available, otherwise "cpu".

Value

A Python torch module.

Examples

```
# Runs only after ar_setup(), so Python is not set up otherwise
if (ar_have_repo() && ar_have_weights("linear_sleepstages") &&
    ar_python_modules_installed()) {
  model = ar_load_model("linear_sleepstages")
}
```

---

ar_load_weights *Load AcceleRest Model Weights*

Description

Extracts the torch state dictionary (the named parameter tensors) from a downloaded weights file.

Usage

```
ar_load_weights(
  model = "linear_sleepstages",
  weights_dir = ar_cache_dir(),
  as_r = TRUE
)
```

Arguments

model	a single model name, see ar_weights()
weights_dir	directory where the weights are (to be) stored. If NULL, the file column is just the file name.
as_r	If TRUE, convert each tensor to an R array; otherwise return the Python OrderedDict of tensors.

Value

A named list of arrays or a Python object.

Examples

```
# Runs only after ar_setup(), so Python is not set up otherwise
if (ar_have_weights("linear_sleepstages") &&
    ar_python_modules_installed()) {
  sd = ar_load_weights("linear_sleepstages")
  names(sd)[1:5]
}
```

`ar_python_modules_installed` *Check for the Python Modules Used by* `rester`

Description

Check for the Python Modules Used by `rester`

Usage

```
ar_python_modules_installed()
```

Value

A logical value indicating whether all Python modules needed by `rester` can be found.

Examples

```
# Checking starts Python, which installs the Python packages, so this
# runs only after ar_setup()
if (ar_have_repo() && ar_have_weights()) {
  ar_python_modules_installed()
}
```

`ar_read` *Read and Preprocess Data for AcceleRest*

Description

AcceleRest expects 30Hz, gravity-calibrated data without gaps. Device files (`.cwa`, `.gt3x`, `.bin`, optionally compressed) are read with `actiread::acti_read()`, and CSV files with `readr::read_csv()`. Gaps (all-zero rows from ActiGraph idle sleep mode, and missing values) are filled with the last observation carried forward using `actibase::acti_fill_zeros()`, as ActiLife does. The data are then low-pass filtered, calibrated, and resampled to 30Hz with the `actipy` Python module, using the same settings as the AcceleRest command-line tool. `.h5` files must already be preprocessed, with an `axes x samples` array in `data/accelerometry`.

Usage

```
ar_read(
  file,
  sample_rate = NULL,
  fill_zeros = TRUE,
  lowpass_hz = 15,
  calibrate_gravity = TRUE,
  detect_nonwear = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

file	a data.frame with a time column and X, Y, Z axes (see actibase::acti_standardize_data()), or a path to a device, CSV, or .h5 file, optionally compressed (.gz, .bz2, .xz). Time stamps must be unique; data with time stamps rounded to the second should have times reconstructed using the sample rate.
sample_rate	sample rate (Hz) of the data. If NULL, it is found with actibase::get_sample_rate() (from the sample_rate attribute, the header, or the time stamps).
fill_zeros	If TRUE, fill gaps with actibase::acti_fill_zeros(). Any gap left in the data makes the predictions for every context window covering it (about 2.1 hours) missing.
lowpass_hz, calibrate_gravity, detect_nonwear	preprocessing options passed to actipy. Non-wear detected by actipy is set to NA.
verbose	print diagnostic messages
...	additional arguments passed to actiread::acti_read() for device files

Value

A list with the Python (numpy) array of `3 x samples` data (accelerometry), the time of the first sample (start_time, NULL for .h5 files), and actipy information about the data (info).

Examples

```
# Runs only after ar_setup(), so Python is not set up otherwise
if (ar_have_repo() && ar_have_weights() &&
    ar_python_modules_installed()) {
  data = ar_example_data(n_hours = 0.1)
  out = ar_read(data)
  out$start_time
}
```

ar_setup *Download AcceleRest Code and Weights*

Description

Runs ar_download_repo() and ar_download_weights().

Usage

```
ar_setup(
  model = NULL,
  cache_dir = ar_cache_dir(),
  overwrite = FALSE,
  quiet = TRUE,
  ...
)
```

Arguments

model	the models to download, see ar_weights()
cache_dir	directory to store the code and weights in
overwrite	If TRUE, re-download and replace an existing copy
quiet	passed to curl::curl_download()
...	additional arguments to pass to curl::curl_download()

Value

A list with the repository path and the weights data.frame.

Examples

```
## Not run:
  # downloads the AcceleRest code and ~100MB of weights to ar_cache_dir()
  ar_setup()

## End(Not run)
```

ar_weights *AcceleRest Model Weights*

Description

The pretrained weights released at https://github.com/NielsRLorenzen/AcceleRest/releases. The heads "linear_sleepstages", "lstm_sleepstages", and "linear_respevents" are used for prediction; the autoencoder weights are the pretrained (not fine-tuned) backbones.

Usage

```
ar_weights(weights_dir = NULL, release = .ar_default_release)

ar_have_weights(
  model = NULL,
  weights_dir = ar_cache_dir(),
  quiet = TRUE,
  release = .ar_default_release
)

ar_download_weights(
  model = NULL,
  weights_dir = ar_cache_dir(),
  overwrite = FALSE,
  quiet = TRUE,
  release = .ar_default_release,
  ...
)
```

Arguments

weights_dir	directory where the weights are (to be) stored. If NULL, the file column is just the file name.
release	the GitHub release tag of the weights
model	the models to check or download. Defaults to the three prediction heads.
quiet	If FALSE, prints a message if weights are missing (ar_have_weights), or shows download progress (ar_download_weights).
overwrite	If TRUE, re-download files that already exist
...	additional arguments to pass to curl::curl_download()

Value

A data.frame with the model name, file, URL, expected size in bytes, and the class labels (for prediction heads).

Examples

```
ar_weights()
ar_have_weights()
## Not run:
  # downloads ~35MB per model to ar_cache_dir()
  files = ar_download_weights("linear_sleepstages")

## End(Not run)
```

`estimate_rest` *Estimate Sleep Stages and Respiratory Events with AcceleRest*

Description

Runs the AcceleRest model on 30-second epochs of wrist-worn accelerometry. Predictions are made over context windows of 256 epochs (about 2.1 hours), so at least that much data is needed; predictions for an epoch are averaged across the overlapping windows covering it.

Usage

```
estimate_rest(
  file,
  model = NULL,
  cache_dir = ar_cache_dir(),
  repo_dir = cache_dir,
  weights_dir = cache_dir,
  context_window_shift = 1L,
  max_batch_size = 16L,
  device = NULL,
  ...
)
```

Arguments

`file`	a `data.frame` or file path, see `ar_read()`, or the output of `ar_read()`
`model`	the prediction heads to include, one or more of `"linear_sleepstages"`, `"lstm_sleepstages"`, and `"linear_respevents"`. Defaults to all three.
`cache_dir`	directory with the downloaded code and weights, see `ar_setup()`
`repo_dir`	directory with the downloaded code, see `ar_download_repo()`
`weights_dir`	directory with the downloaded weights, see `ar_download_weights()`
`context_window_shift`	number of 30-second epochs to shift consecutive context windows. Smaller is slower but averages over more windows.
`max_batch_size`	number of context windows per forward pass. Higher is faster but uses more memory.

device	torch device, e.g. "cpu", "cuda", or "mps". If NULL, uses "cuda" if available, otherwise "cpu".
...	additional arguments passed to ar_read()

Value

A named list with a tibble for each head, with the epoch number, start time of the epoch (NA for .h5 files), the probability of each class, and the most probable class (prediction). Epochs with no valid data have NA values. Sleep stage classes are deep, light, rem, and wake; respiratory event classes are no_event and event. The actipy information about the data is in attr(x, "info").

References

Lorenzen NR, Brink-Kjaer A, Jennum PJ, Mignot E (2026). AcceleRest: A Physiology-Aware Masked Autoencoder for Wrist Accelerometer-based Sleep Staging and Apnea Evaluation. *medRxiv*. doi:10.64898/2026.01.28.26345056

Examples

```
# Runs only after ar_setup(), so Python is not set up otherwise
if (ar_have_repo() && ar_have_weights("linear_sleepstages") &&
    ar_python_modules_installed()) {
  data = ar_example_data(n_hours = 2.5)
  out = estimate_rest(data, model = "linear_sleepstages",
                      context_window_shift = 32L)
  head(out$linear_sleepstages)
}
```

---

py_estimate_rest *Run* estimate_rest *with Python in a Separate Process*

Description

Run estimate_rest with Python in a Separate Process

Usage

```
py_estimate_rest(
  ...,
  pyenv_function = function() {
     rester::py_require_rester()
 },
  show = FALSE
)
```

Arguments

...	arguments to pass to estimate_rest()
pyenv_function	function that sets up Python in the child process. By default, it uses py_require_rester(). If this function has an args argument, the output of pyenv_function will be re-assigned to args.
show	Logical, whether to show the standard output on the screen while the child process is running, passed to callr::r()

Value

Output from estimate_rest().

Examples

```
# Runs only after ar_setup(), so Python is not set up otherwise
if (requireNamespace("callr", quietly = TRUE) &&
    ar_have_repo() && ar_have_weights("linear_sleepstages")) {
  data = ar_example_data(n_hours = 2.5)
  out = py_estimate_rest(data, model = "linear_sleepstages",
                         context_window_shift = 32L)
}
```

---

py_require_rester *Command for* py_require *for* rester

Description

Declares the Python packages needed to run AcceleRest using reticulate::py_require(). This is called when rester is loaded, so calling it directly is only needed if you pass additional arguments.

Usage

```
py_require_rester(...)
```

Arguments

...	arguments to pass to reticulate::py_require()

Value

The output of reticulate::py_require(), invisibly.

Examples

```
# Only declares the requirements; nothing is installed until Python starts
py_require_rester()
```

Index

actibase::acti_fill_zeros(), *6, 7*
actibase::acti_standardize_data(), *7*
actibase::get_sample_rate(), *7*
actiread::acti_read(), *6, 7*
ar_cache_dir, 2
ar_download_repo, 3
ar_download_repo(), *5*, *8*, *10*
ar_download_weights *(*ar_weights*)*, 9
ar_download_weights(), *5*, *8*, *10*
ar_example_data, 4
ar_extract_repo *(*ar_download_repo*)*, 3
ar_have_repo *(*ar_download_repo*)*, 3
ar_have_weights *(*ar_weights*)*, 9
ar_load_model, 4
ar_load_weights, 5
ar_python_modules_installed, 6
ar_read, 6
ar_read(), *10, 11*
ar_setup, 8
ar_setup(), *4*, *10*
ar_weights, 9
ar_weights(), *5*, *8*

callr::r(), *12*
curl::curl_download(), *3*, *8, 9*

estimate_rest, 10
estimate_rest(), *12*

py_estimate_rest, 11
py_require_rester, 12
py_require_rester(), *12*

readr::read_csv(), *6*
reticulate::py_require(), *12*