Package: telemetar (via r-universe)

September 3, 2024

Title Archetypes for Targets and Fish Telemetry

Version 0.0.0.9000

Description What the package does (one paragraph).

License MIT + file LICENSE

Imports data.table, qs, rvdat, tarchetypes, targets (>= 1.4.0), utils

Suggests curl, testthat (>= 3.0.0)

Remotes mhpob/rvdat

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Repository https://ocean-tracking-network.r-universe.dev

RemoteUrl https://github.com/mhpob/telemetar

RemoteRef HEAD

RemoteSha 899eb2bd415da56b26076fb95b02346d0e40b11a

Contents

`tar_vdat_read` *Dynamic branching over VDAT files*

Description

Dynamic branching over VDAT files

Usage

```
tar_vdat_read(
  name,
  vdat_dirs,
  csv_outdir,
  batch_size = 10,
  batches = NULL,
  format = c("file", "file_fast", "url", "aws_file"),
  repository = targets::tar_option_get("repository"),
  iteration = targets::tar_option_get("iteration"),
  error = targets::tar_option_get("error"),
  memory = targets::tar_option_get("memory"),
  garbage_collection = targets::tar_option_get("garbage_collection"),
  priority = targets::tar_option_get("priority"),
  resources = targets::tar_option_get("resources"),
  cue = targets::tar_option_get("cue")
)
```

Arguments

`name`	Symbol, name of the target. A target name must be a valid name for a symbol in R, and it must not start with a dot. Subsequent targets can refer to this name symbolically to induce a dependency relationship: e.g. `tar_target(downstream_target, f(upstream_target))` is a target named `downstream_target` which depends on a target `upstream_target` and a function `f()`. In addition, a target's name determines its random number generator seed. In this way, each target runs with a reproducible seed so someone else running the same pipeline should get the same results, and no two targets in the same pipeline share the same seed. (Even dynamic branches have different names and thus different seeds.) You can recover the seed of a completed target with `tar_meta(your_target, seed)` and run `tar_seed_set()` on the result to locally recreate the target's initial RNG state.
`vdat_dirs`	Nonempty character vector of known existing directories of VDAT files to track for changes.
`csv_outdir`	file path to the output directory
`batch_size`	Positive integer of length 1, number of files to partition into a batch. The default is ten files per batch.

`batches` Positive integer of length 1, number of batches to partition the files. The default is one file per batch (maximum number of batches) which is simplest to handle but could cause a lot of overhead and consume a lot of computing resources. Consider reducing the number of batches below the number of files for heavy workloads.

`format` Character, either `"file"`, `"file_fast"`, or `"url"`. See the `format` argument of `targets::tar_target()` for details.

`repository` Character of length 1, remote repository for target storage. Choices:

- `"local"`: file system of the local machine.
- `"aws"`: Amazon Web Services (AWS) S3 bucket. Can be configured with a non-AWS S3 bucket using the `endpoint` argument of `tar_resources_aws()`, but versioning capabilities may be lost in doing so. See the cloud storage section of https://books.ropensci.org/targets/data.html for details for instructions.
- `"gcp"`: Google Cloud Platform storage bucket. See the cloud storage section of https://books.ropensci.org/targets/data.html for details for instructions.

Note: if `repository` is not `"local"` and `format` is `"file"` then the target should create a single output file. That output file is uploaded to the cloud and tracked for changes where it exists in the cloud. The local file is deleted after the target runs.

`iteration` Character, iteration method. Must be a method supported by the `iteration` argument of `targets::tar_target()`. The iteration method for the upstream target is always `"list"` in order to support batching.

`error` Character of length 1, what to do if the target stops and throws an error. Options:

- `"stop"`: the whole pipeline stops and throws an error.
- `"continue"`: the whole pipeline keeps going.
- `"abridge"`: any currently running targets keep running, but no new targets launch after that. (Visit https://books.ropensci.org/targets/debugging.html to learn how to debug targets using saved workspaces.)
- `"null"`: The errored target continues and returns `NULL`. The data hash is deliberately wrong so the target is not up to date for the next run of the pipeline.

`memory` Character of length 1, memory strategy. If `"persistent"`, the target stays in memory until the end of the pipeline (unless `storage` is `"worker"`, in which case `targets` unloads the value from memory right after storing it in order to avoid sending copious data over a network). If `"transient"`, the target gets unloaded after every new target completes. Either way, the target gets automatically loaded into memory whenever another target needs the value. For cloud-based dynamic files (e.g. `format = "file"` with `repository = "aws"`), this memory strategy applies to the temporary local copy of the file: `"persistent"` means it remains until the end of the pipeline and is then deleted, and `"transient"` means it gets deleted as soon as possible. The former conserves bandwidth, and the latter conserves local storage.

`garbage_collection`
Logical, whether to run `base::gc()` just before the target runs.

priority	Numeric of length 1 between 0 and 1. Controls which targets get deployed first when multiple competing targets are ready simultaneously. Targets with priorities closer to 1 get dispatched earlier (and polled earlier in tar_make_future()).
resources	Object returned by tar_resources() with optional settings for high-performance computing functionality, alternative data storage formats, and other optional capabilities of targets. See tar_resources() for details.
cue	An optional object from tar_cue() to customize the rules that decide whether the target is up to date. Only applies to the downstream target. The upstream target always runs.

Examples

```
# example code
```

`tar_vue_csvs` *Dynamic branching over VUE/VDAT-exported CSV detection files.*

Description

Dynamic branching over VUE/VDAT-exported CSV detection files.

Usage

```
tar_vue_csvs(
  name,
  csv_dirs,
  pattern = "^[VH]R.*\\.csv$",
  batch_size = 10,
  format = c("file", "file_fast", "url", "aws_file"),
  repository = targets::tar_option_get("repository"),
  iteration = targets::tar_option_get("iteration"),
  error = targets::tar_option_get("error"),
  memory = targets::tar_option_get("memory"),
  garbage_collection = targets::tar_option_get("garbage_collection"),
  priority = targets::tar_option_get("priority"),
  resources = targets::tar_option_get("resources"),
  cue = targets::tar_option_get("cue")
)
```

Arguments

name	Symbol, name of the target. A target name must be a valid name for a symbol in R, and it must not start with a dot. Subsequent targets can refer to this name symbolically to induce a dependency relationship: e.g. tar_target(downstream_target, f(upstream_target)) is a target named downstream_target which depends on a target upstream_target and a function f(). In addition, a target's name

determines its random number generator seed. In this way, each target runs with a reproducible seed so someone else running the same pipeline should get the same results, and no two targets in the same pipeline share the same seed. (Even dynamic branches have different names and thus different seeds.) You can recover the seed of a completed target with `tar_meta(your_target, seed)` and run `tar_seed_set()` on the result to locally recreate the target's initial RNG state.

`csv_dirs` Nonempty character vector of known existing directories of CSV files to track for changes.

`pattern` a regular expression to search for the applicable CSV files. Defaults to `"^[VH]R.*\\.csv$"`.

`batch_size` Positive integer of length 1, number of files to partition into a batch. The default is ten files per batch.

`format` Character, either `"file"`, `"file_fast"`, or `"url"`. See the `format` argument of `targets::tar_target()` for details.

`repository` Character of length 1, remote repository for target storage. Choices:

- `"local"`: file system of the local machine.
- `"aws"`: Amazon Web Services (AWS) S3 bucket. Can be configured with a non-AWS S3 bucket using the `endpoint` argument of `tar_resources_aws()`, but versioning capabilities may be lost in doing so. See the cloud storage section of https://books.ropensci.org/targets/data.html for details for instructions.
- `"gcp"`: Google Cloud Platform storage bucket. See the cloud storage section of https://books.ropensci.org/targets/data.html for details for instructions.

Note: if `repository` is not `"local"` and `format` is `"file"` then the target should create a single output file. That output file is uploaded to the cloud and tracked for changes where it exists in the cloud. The local file is deleted after the target runs.

`iteration` Character, iteration method. Must be a method supported by the `iteration` argument of `targets::tar_target()`. The iteration method for the upstream target is always `"list"` in order to support batching.

`error` Character of length 1, what to do if the target stops and throws an error. Options:

- `"stop"`: the whole pipeline stops and throws an error.
- `"continue"`: the whole pipeline keeps going.
- `"abridge"`: any currently running targets keep running, but no new targets launch after that. (Visit https://books.ropensci.org/targets/debugging.html to learn how to debug targets using saved workspaces.)
- `"null"`: The errored target continues and returns `NULL`. The data hash is deliberately wrong so the target is not up to date for the next run of the pipeline.

`memory` Character of length 1, memory strategy. If `"persistent"`, the target stays in memory until the end of the pipeline (unless `storage` is `"worker"`, in which case `targets` unloads the value from memory right after storing it in order to avoid sending copious data over a network). If `"transient"`, the target gets unloaded

after every new target completes. Either way, the target gets automatically loaded into memory whenever another target needs the value. For cloud-based dynamic files (e.g. `format = "file"` with `repository = "aws"`), this memory strategy applies to the temporary local copy of the file: `"persistent"` means it remains until the end of the pipeline and is then deleted, and `"transient"` means it gets deleted as soon as possible. The former conserves bandwidth, and the latter conserves local storage.

`garbage_collection`
: Logical, whether to run `base::gc()` just before the target runs.

`priority`
: Numeric of length 1 between 0 and 1. Controls which targets get deployed first when multiple competing targets are ready simultaneously. Targets with priorities closer to 1 get dispatched earlier (and polled earlier in `tar_make_future()`).

`resources`
: Object returned by `tar_resources()` with optional settings for high-performance computing functionality, alternative data storage formats, and other optional capabilities of `targets`. See `tar_resources()` for details.

`cue`
: An optional object from `tar_cue()` to customize the rules that decide whether the target is up to date. Only applies to the downstream target. The upstream target always runs.

Examples

```
targets::tar_dir({
  ## Download example data
  download.file(
    file.path('https://raw.githubusercontent.com/ocean-tracking-network/glatos',
              'main/inst/extdata/VR2W_109924_20110718_1.csv'),
    'VR2W_109924_20110718_1.csv'
  )

  for(i in 2:12){
    file.copy(
      'VR2W_109924_20110718_1.csv',
      paste0('VR2W_109924_20110718_', i, '.csv')
    )
  }

  ## Run workflow
  targets::tar_script({
    list(
      telemetar::tar_vue_csvs(
        my_detections,
        getwd()
      )
    )
  })

  targets::tar_make(callr_function = NULL)

})
```

Index