`RAPToR` - Data-packages

RAPToR 1.2.0

Romain Bulteau

March 2023

Contents

1 Introduction

This vignette is aimed at those who've already familiarized themselves with `RAPToR` reference building. It's also for us to keep track of guidelines to continue improving `RAPToR` with new references.

If you've already built some references and want to make them available to the world (or use them more easily yourself), you're in the right place. You will need some basic knowledge of R package development. Data-packages are, after all, *packages*. This document only details how to set up your data-package to interact properly with `RAPToR`.

2 What's a data-package ?

By definition, a "data-package" is an R package in which one stores large datasets (over a few Mo). This is a good practice for several reasons.

1. If your data rarely or never changes, updates to the data-package (and thus, download of the data) will be minimal. If included in a standard package, large data can be a burden during install.
2. The data may never be used. Why have users download data they won't need ?
3. CRAN standards limit package size to 5MB (documentation included). A large dataset is better off separated from methods and functions that may need it.

Hadley Wickam gives thorough advice on organizing data in packages in his *R packages* book.

3 Reference data

RAPToR uses references which can sometimes be tedious to build, so we want to give users access to pre-built references as much as possible. References are stored as .RData objects, which must include everything needed to make the interpolated reference:

- gene expression data,
- optimal interpolation parameters,
- some metadata (*e.g.* time units).

There can be as many references as needed in a data-package; we group packages by organism.

Being consistent, clear and concise with naming can help users (or yourself !) find their way around references. For example, wormRef references are named with an organism code Cel (*C. elegans*) followed by the developmental period covered by the reference *e.g.* larval.

Document the references. It is standard practice to document data. What is the data? Where is the data from? Publication? etc.

The structure of the Cel_larval reference object is detailed below as an example.

```
str(wormRef::Cel_larval, vec.len = 2)
#> List of 6
#>  $ g          : num [1:18718, 1:62] 2.72 3.06 ...
#>   ..- attr(*, "dimnames")=List of 2
#>   .. ..$ : chr [1:18718] "WBGene00007063" "WBGene00007064" ...
#>   .. ..$ : chr [1:62] "DH2_N2_0" "DH2_N2_2" ...
#>  $ p          :'data.frame': 62 obs. of  5 variables:
#>   ..$ sname    : chr [1:62] "DH2_N2_0" "DH2_N2_2" ...
#>   ..$ age      : num [1:62] 0 2 4 6 8 ...
#>   ..$ cov      : Factor w/ 3 levels "0.20","0.25",..: 1 1 1 1 1 ...
#>   ..$ age_ini  : num [1:62] 0 2 4 6 8 ...
#>   ..$ accession: chr [1:62] "GSM1192801" "GSM1192802" ...
#>  $ geim_params:List of 4
#>   ..$ formula: chr "X ~ s(age, bs = 'ds') + cov"
#>   ..$ method : chr "gam"
#>   ..$ dim_red: chr "pca"
#>   ..$ nc     : num 40
#>  $ t.unit     : chr "h past egg-laying (20C)"
#>  $ cov.levels :List of 1
#>   ..$ cov: chr "0.20"
#>  $ metadata   :List of 3
#>   ..$ organism  : chr "C. elegans"
#>   ..$ profiling : chr "whole-organism, bulk"
#>   ..$ technology: chr "RNAseq"
```

With

- g The gene expression matrix (genes as rows, samples as columns).
- p A dataframe of phenotypic data on the samples :
  - sname sample names,
  - age *developmental* age of the samples (scaled),
  - cov covariate, factor indicating which of 3 time series,
  - age_ini *chronological* age of the samples,
  - accession sample accession ID for GEO.
- geim_params A list with necessary parameters for interpolation
- t.unit the time unit.
- cov.levels A named list with covariate levels to interpolate as.
- metadata A named list with any extra metadata

The documentation for Cel_larval can be accessed with ?Cel_larval.

4 Data-package interface with RAPToR

For a user to access data-package information and references directly from RAPToR, we've set up a standard system using reference names.

A few objects are *necessary* for this interface to work.

4.1 .prepref_ functions

.prepref_ functions (note the "dot") are the key of the interface : they **prep**are the **ref**erence for the user. They must respect the naming convention .prepref_ref_name() (*e.g.* .prepref_Cel_larval()) and take n.inter and/or by.inter as arguments.

These functions are the backbone called by prepare_refdata() when fetching a reference, and thus should output the reference ref object with the specified parameters. This means building the GEIM model, and calling make_ref() with the appropriate parameters and metadata.

We have made a function factory to generate these functions. It inputs the reference data object described above, and returns the corresponding prepref function.

```
.prepref_skel <- function(data, from=NULL, to=NULL){
  # .prepref function factory
  f <- function(n.inter=NULL, by.inter=NULL){
    m <- RAPToR::ge_im(
      X = data$g,
      p = data$p,
      formula = data$geim_params$formula,
      method = data$geim_params$method,
      dim_red = data$geim_params$dim_red,
      nc = data$geim_params$nc
    )
    return(RAPToR::make_ref(m,
                            n.inter = n.inter,
                            by.inter = by.inter,
                            from = from,
                            to = to,
                            t.unit = data$t.unit,
```

```
                            cov.levels = data$cov.levels,
                            metadata = data$metadata)
    )
  }
  return(f)
}
```

To make `.prepref_Cel_larval()`, we simply include the following code in the data-package, along with the function factory.

```
.prepref_Cel_larval <- .prepref_skel(wormRef::Cel_larval)
```

4.2 `ref_list` object

`RAPToR` expects a `ref_list` object in the data-package. This is what's displayed when calling the `list_refs(datapkg)` function.

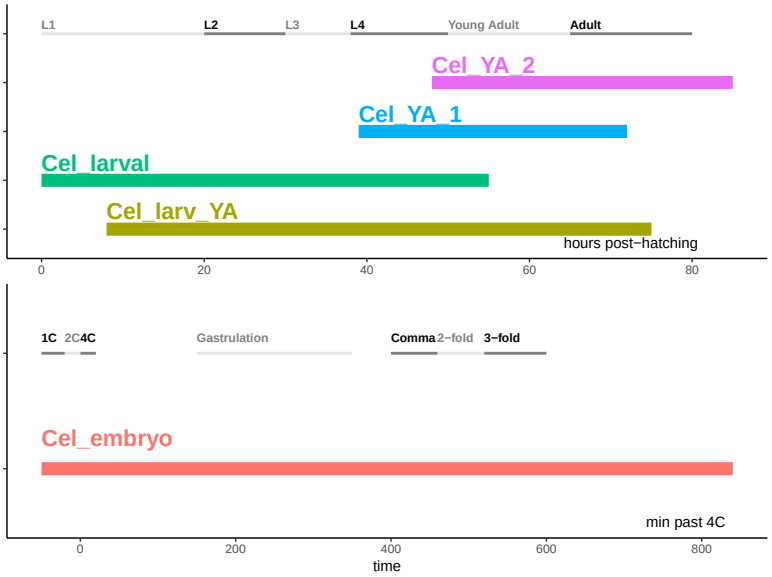
```
library(RAPToR)
library(wormRef)

list_refs(datapkg = "wormRef")
#>          name               organism                 description
#> 1  Cel_embryo Caenorhabditis elegans       Embryonic development
#> 2  Cel_larval Caenorhabditis elegans          Larval development
#> 3 Cel_larv_YA Caenorhabditis elegans Larval to adult development
#> 4    Cel_YA_1 Caenorhabditis elegans              YA development
#> 5    Cel_YA_2 Caenorhabditis elegans              YA development
#>                                range
#> 1 1-cell(-50) to 840 min past 4-cell
#> 2      0 to 55 h post-hatching (20C)
#> 3      5 to 76 h post-hatching (20C)
#> 4   41.5 to 72 h post-hatching (20C)
#> 5     48 to 87 h post-hatching (20C)
```

The form/layout of this object is free, but reference names should be included somewhere, as the user needs them to access the reference through `prepare_refdata()`.

4.3 `.plot_refs()` function

This function is optional, but very useful to guide users to the correct reference for their samples. Including a `.plot_refs()` function (note the "dot") in a data-package will allow it to be called by the `plot_refs(datapkg)` function in `RAPToR`.

```
plot_refs(datapkg = "wormRef")
```



4.4 Other objects

You're free to include any extra objects in your data-package that may be useful. For example, the `wormRef` package has a `Cel_devstages` object with information on key developmental stages of *C. elegans* (which is used for building the plot in `.plot_refs()` above).

SessionInfo

```
sessionInfo()
```

```
#> R version 4.1.2 (2021-11-01)
#> Platform: x86_64-pc-linux-gnu (64-bit)
#> Running under: Ubuntu 22.04.1 LTS
#>
#> Matrix products: default
#> BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
#> LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.20.so
#>
#> locale:
#>  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
#>  [3] LC_TIME=C                  LC_COLLATE=en_US.UTF-8
#>  [5] LC_MONETARY=fr_FR.UTF-8    LC_MESSAGES=en_US.UTF-8
#>  [7] LC_PAPER=fr_FR.UTF-8       LC_NAME=C
#>  [9] LC_ADDRESS=C               LC_TELEPHONE=C
#> [11] LC_MEASUREMENT=fr_FR.UTF-8 LC_IDENTIFICATION=C
#>
#> attached base packages:
#> [1] splines   parallel  stats     graphics  grDevices utils
#> [7] datasets  methods   base
#>
#> other attached packages:
```

```
#>  [1] vioplot_0.4.0      zoo_1.8-11         sm_2.2-5.7.1
#>  [4] ica_1.0-3          drosoRef_0.2.0     limma_3.50.3
#>  [7] wormRef_0.5.0      RAPToR_1.2.0       ggpubr_0.6.0
#> [10] ggplot2_3.4.0.9000 BiocStyle_2.22.0
#>
#> loaded via a namespace (and not attached):
#>  [1] Rcpp_1.0.10         lattice_0.20-45     tidyr_1.3.0
#>  [4] digest_0.6.31       utf8_1.2.3          R6_2.5.1
#>  [7] backports_1.4.1     evaluate_0.20       highr_0.10
#> [10] pillar_1.8.1        Rdpack_2.4          rlang_1.0.6
#> [13] rstudioapi_0.14     data.table_1.14.6   car_3.1-1
#> [16] jquerylib_0.1.4     magick_2.7.3        Matrix_1.5-3
#> [19] rmarkdown_2.20.1    labeling_0.4.2      stringr_1.5.0
#> [22] tinytex_0.44        munsell_0.5.0       broom_1.0.3
#> [25] compiler_4.1.2      xfun_0.37           pkgconfig_2.0.3
#> [28] mgcv_1.8-41         htmltools_0.5.4     tidyselect_1.2.0
#> [31] tibble_3.1.8        bookdown_0.32       codetools_0.2-18
#> [34] fansi_1.0.4         dplyr_1.1.0         withr_2.5.0
#> [37] rbibutils_2.2.13    grid_4.1.2          nlme_3.1-161
#> [40] jsonlite_1.8.4      gtable_0.3.1        lifecycle_1.0.3
#> [43] magrittr_2.0.3      scales_1.2.1        cli_3.6.0
#> [46] stringi_1.7.12      cachem_1.0.6        carData_3.0-5
#> [49] farver_2.1.1        ggsignif_0.6.4      pryr_0.1.6
#> [52] bslib_0.4.2         generics_0.1.3      vctrs_0.5.2
#> [55] tools_4.1.2         glue_1.6.2          beeswarm_0.4.0
#> [58] purrr_1.0.1         abind_1.4-5         fastmap_1.1.0
#> [61] yaml_2.3.7          colorspace_2.1-0    BiocManager_1.30.19
#> [64] rstatix_0.7.2       knitr_1.42          sass_0.4.5
```