Package: ReUseData (via r-universe)

November 5, 2026

Title Reusable and reproducible Data Management

Version 1.12.0

Description ReUseData is an _R/Bioconductor_ software tool to provide a systematic and versatile approach for standardized and reproducible data management. ReUseData facilitates transformation of shell or other ad hoc scripts for data preprocessing into workflow-based data recipes. Evaluation of data recipes generate curated data files in their generic formats (e.g., VCF, bed). Both recipes and data are cached using database infrastructure for easy data management and reuse. Prebuilt data recipes are available through ReUseData portal (``https://rcwl.org/dataRecipes/'') with full annotation and user instructions. Pregenerated data are available through ReUseData cloud bucket that is directly downloadable through ``getCloudData()''.

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Imports Rcwl, RcwlPipelines, BiocFileCache, S4Vectors, stats, tools, utils, methods, jsonlite, yaml, basilisk

Suggests knitr, rmarkdown, testthat (>= 3.0.0), BiocStyle

Config/testthat/edition 3

VignetteBuilder knitr

biocViews Software, Infrastructure, DataImport, Preprocessing, ImmunoOncology

License GPL-3

URL https://github.com/rworkflow/ReUseData

BugReports https://github.com/rworkflow/ReUseData/issues

Config/pak/sysreqs cmake libglpk-dev make libgit2-dev libicu-dev libpng-dev libssh2-1-dev libuv1-dev libxml2-dev libssl-dev python3 libx11-dev zlib1g-dev

Repository Bioconductor 3.23

Date/Publication 2026-04-28 12:59:39 UTC

RemoteUrl https://github.com/bioc/ReUseData

RemoteRef RELEASE_3_23

RemoteSha bb3489b74bbb9d73551e0aee01cc5e29d2c836a6

Contents

annData 2
dataHub 3
dataSearch 5
dataUpdate 6
getCloudData 8
getData 9
meta_data 10
recipeHub 11
recipeLoad 12
recipeMake 13
recipeSearch 15
recipeUpdate 16
ReUseData 17

Index **18**

`annData` *annData*

Description

Add annotation or meta information to existing data

Usage

```
annData(
  path,
  notes,
  date = Sys.Date(),
  recursive = TRUE,
  md5 = FALSE,
  skip = "*.md|meta.yml",
  force = FALSE,
  ...
)
```

Arguments

path	The data path to annotate.
notes	User assigned notes/keywords to annotate the data and be used for keywords matching in dataSearch(keywords =).
date	The date of the data.
recursive	Whether to annotate all data recursively.
md5	Whether to generate md5 values for all files.
skip	Patter to skip files in the path.
force	Whether to force regenerate meta.yml.
...	The other options from list.files

---

dataHub *dataHub Class*

Description

dataHub class, constructor, and methods.

Usage

```
dataHub(BFC)

dataHub(BFC)

## S4 method for signature 'dataHub'
show(object)

dataNames(object)

dataParams(object)

dataNotes(object)

dataPaths(object)

dataYml(object)

dataTags(object)

## S4 method for signature 'dataHub'
dataTags(object)

dataTags(object, append = TRUE) <- value
```

```
## S4 replacement method for signature 'dataHub'
dataTags(object, append = FALSE) <- value

## S4 method for signature 'dataHub,ANY,ANY,ANY'
x[i, j, drop]

## S4 replacement method for signature 'dataHub,ANY,ANY,ANY'
x[i, j] <- value

## S4 method for signature 'dataHub'
c(x, ...)

toList(
  x,
  listNames = NULL,
  format = c("list", "json", "yaml"),
  type = NULL,
  file = character()
)
```

Arguments

BFC	A BiocFileCache object created for data and recipes.
object	A dataHub object.
append	Whether to append new tag or replace all tags.
value	A dataHub object
x	A dataHub object.
i	The integer index of the dataHub object, or a logical vector same length as the dataHub object.
j	inherited from [generic.
drop	Inherited from [generic.
...	More dataHub objects to combine.
listNames	A vector of names for the output list.
format	can be "list", "json" or "yaml". Supports partial match. Default is list.
type	The type of workflow input list, such as cwl.
file	The file name to save the data list in required format. The data extension needs to be included, e.g., ".json" or ".yml".

Value

dataHub: a dataHub object.

dataNames: the names of datasets in dataHub object.

dataParams: the data recipe parameter values for datasets in dataHub object.

dataNotes: the notes of datasets in dataHub object.

dataPaths: the file paths of datasets in dataHub object.

dataYml: the yaml file paths of datasets in dataHub object.

dataTags: the tags of datasets in dataHub object.

toList: A list of datasets in specific format, and a file if file argument is specified.

Examples

```
outdir <- file.path(tempdir(), "SharedData")
dataUpdate(outdir, cloud = TRUE)
dd <- dataSearch(c("liftover", "GRCh38"))
dataNames(dd)
dataParams(dd)
dataNotes(dd)
dataTags(dd)
dataYml(dd)
toList(dd)
toList(dd, format = "yaml")
toList(dd, format = "json", file = tempfile())
```

---

dataSearch *dataSearch search data in local data caching system*

Description

dataSearch search data in local data caching system

Usage

```
dataSearch(keywords = character(), cachePath = "ReUseData")
```

Arguments

keywords	character vector of keywords to be matched to the local datasets. It matches the "notes" when generating the data using getData(notes =). Keywords can be a tag with the data in #tag format. If not specified, function returns the full data list.
cachePath	A character string for the data cache. Must match the one specified in dataUpdate(). Default is "ReUseData".

Value

a dataHub object containing the information about local data cache, e.g., data name, data path, etc.

Examples

```
dataSearch()
dataSearch(c("gencode"))
dataSearch("#gatk")
```

dataUpdate *dataUpdate*

Description

Function to update the local data records by reading the yaml files in the specified directory recursively.

Usage

```
dataUpdate(
  dir,
  cachePath = "ReUseData",
  outMeta = FALSE,
  keepTags = TRUE,
  cleanup = FALSE,
  cloud = FALSE,
  remote = FALSE,
  checkData = TRUE,
  duplicate = FALSE
)
```

Arguments

dir	a character string for the directory where all data are saved. Data information will be collected recursively within this directory.
cachePath	A character string specifying the name for the BiocFileCache object to store all the curated data resources. Once specified, must match the cachePath argument in dataSearch. Default is "ReUseData".
outMeta	Logical. If TRUE, a "meta_data.csv" file will be generated in the dir, containing information about all available datasets in the directory: The file path to the yaml files, and yaml entries including parameter values for data recipe, file path to datasets, notes, version (from getData()), if available and data generating date.
keepTags	If keep the prior assigned data tags. Default is TRUE.
cleanup	If remove any invalid intermediate files. Default is FALSE. In cases one data recipe (with same parameter values) was evaluated multiple times, the same data file(s) will match to multiple intermediate files (e.g., .yml). cleanup will remove older intermediate files, and only keep the most recent ones that matches the data file. When there are any intermediate files that don't match to any data file, cleanup will also remove those.
cloud	Whether to return the pregenerated data from Google Cloud bucket of ReUseData. Default is FALSE.
remote	Whether to use the csv file (containing information about pregenerated data on Google Cloud) from GitHub, which is most up-to-date. Only works when cloud = TRUE. Default is FALSE.

checkData — check if the data (listed as "# output: " in the yml file) exists. If not, do not include in the output csv file. This argument is added for internal testing purpose.

duplicate — Whether to remove duplicates. If TRUE, older version of duplicates will be removed.

Details

Users can directly retrieve information for all available datasets by using `meta_data(dir=)`, which generates a data frame in R with same information as described above and can be saved out. `dataUpdate` does extra check for all datasets (check the file path in "output" column), remove invalid ones, e.g., empty or non-existing file path, and create a data cache for all valid datasets.

Value

a `dataHub` object containing the information about local data cache, e.g., data name, data path, etc.

Examples

```
## Generate data
## Not run:
library(Rcwl)
outdir <- file.path(tempdir(), "SharedData")

echo_out <- recipeLoad("echo_out")
Rcwl::inputs(echo_out)
echo_out$input <- "Hello World!"
echo_out$outfile <- "outfile"
res <- getData(echo_out,
               outdir = outdir,
               notes = c("echo", "hello", "world", "txt"),
               showLog = TRUE)

ensembl_liftover <- recipeLoad("ensembl_liftover")
Rcwl::inputs(ensembl_liftover)
ensembl_liftover$species <- "human"
ensembl_liftover$from <- "GRCh37"
ensembl_liftover$to <- "GRCh38"
res <- getData(ensembl_liftover,
        outdir = outdir,
        notes = c("ensembl", "liftover", "human", "GRCh37", "GRCh38"),
        showLog = TRUE)

## Update data cache (with or without prebuilt data sets from ReUseData cloud bucket)
dataUpdate(dir = outdir)
dataUpdate(dir = outdir, cloud = TRUE)

## newly generated data are now cached and searchable
dataSearch(c("hello", "world"))
dataSearch(c("ensembl", "liftover")) ## both locally generated data and google cloud data!

## End(Not run)
```

getCloudData *getCloudData Download the pregenerated curated data sets from ReUseData cloud bucket*

Description

getCloudData Download the pregenerated curated data sets from ReUseData cloud bucket

Usage

```
getCloudData(datahub, outdir = character())
```

Arguments

datahub	The dataHub object returned from dataSearch() with 1 data record available on ReUseData cloud bucket.
outdir	The output directory for the data (and concomitant annotation files) to be downloaded. It is recommended to use a new folder under a shared folder for a new to-be-downloaded data.

Value

Data and concomitant annotation files will be downloaded to the user-specified folder that is locally searchable with dataSearch().

Examples

```
outdir <- file.path(tempdir(), "gcpData")
dh <- dataSearch(c("ensembl", "GRCh38"))
dh <- dh[grep("http", dataPaths(dh))]

## download data from google bucket
getCloudData(dh[1], outdir = outdir)

## Update local data caching
dataUpdate(outdir)  ## no "cloud=TRUE" here, only showing local data cache

## Now the data is available to use locally
dataSearch(c("ensembl", "GRCh38"))
```

getData *getData*

Description

Evaluation of data recipes to generate curated dataset of interest.

Usage

```
getData(
  rcp,
  outdir,
  prefix = NULL,
  notes = c(),
  conda = FALSE,
  BPPARAM = NULL,
  ...
)
```

Arguments

rcp	the data recipe in cwlProcess S4 class.
outdir	Character string specifying the directory to store the output files. Will automatically create if not exist or provided.
prefix	Character string specifying the file name of the annotation files (.yml, .cwl, .sh, .md5).
notes	User assigned notes/keywords to annotate the data and be used for keywords matching in dataSearch(keywords =).
conda	Whether to use conda to install required software when evaluating the data recipe as a CWL workflow. Default is FALSE.
BPPARAM	The options for BiocParallel::bpparam.
...	Arguments to be passed into Rcwl:runCWL().

Value

The data files and 4 meta files: .cwl: The cwl script that was internally run to get the data; .yml: the input parameter values for the data recipe and user specified data annotation notes, versions etc; .sh: The script for data processing; .md: checksum file to verify the integrity of generated data files.

Examples

```
## Not run:
library(Rcwl)
outdir <- file.path(tempdir(), "SharedData")
```

```
## Example 1
echo_out <- recipeLoad("echo_out")
Rcwl::inputs(echo_out)
echo_out$input <- "Hello World!"
echo_out$outfile <- "outfile"
res <- getData(echo_out,
               outdir = outdir,
               notes = c("echo", "hello", "world", "txt"),
               showLog = TRUE)

# Example 2
ensembl_liftover <- recipeLoad("ensembl_liftover")
Rcwl::inputs(ensembl_liftover)
ensembl_liftover$species <- "human"
ensembl_liftover$from <- "GRCh37"
ensembl_liftover$to <- "GRCh38"

res <- getData(ensembl_liftover,
        outdir = outdir,
        notes = c("ensembl", "liftover", "human", "GRCh37", "GRCh38"),
        showLog = TRUE)
dir(outdir)

## End(Not run)
```

`meta_data` *meta_data*

Description

Functions to generate the meta csv file for local cached dataset.

Usage

```
meta_data(dir = "", cleanup = FALSE, checkData = TRUE)
```

Arguments

`dir`	The path to the shared data folder.
`cleanup`	If remove any invalid intermediate files. Default is FALSE. In cases one data recipe (with same parameter values) was evaluated multiple times, the same data file(s) will match to multiple intermediate files (e.g., .yml). `cleanup` will remove older intermediate files, and only keep the most recent ones that matches the data file. When there are any intermediate files that don't match to any data file, `cleanup` will also remove those.
`checkData`	check if the data (listed as "# output: " in the yml file) exists. If not, do not include in the output csv file. This argument is added for internal testing purpose.

Value

a data.frame with yml file name, parameter values, data file paths, date, and user-specified notes when generating the data with getData().

Examples

```
outdir <- file.path(tempdir(), "SharedData")
meta_data(outdir)
```

---

recipeHub *recipeHub*

Description

recipeHub class, constructor, and methods.

Usage

```
recipeHub(BFC)

recipeHub(BFC)

## S4 method for signature 'recipeHub'
show(object)

## S4 method for signature 'recipeHub,ANY,ANY,ANY'
x[i]

recipeNames(object)
```

Arguments

BFC	A BiocFileCache object created for recipe and recipes.
object	The recipeHub object
x	The recipeHub object
i	The integer index of the recipeHub object

Value

recipeHub: a recipeHub object.

[: A recipeHub object that was subsetted.

recipeNames: the recipe names for the recipeHub object.

Examples

```
rcps <- recipeSearch(c("gencode"))
## rcp1 <- rcps[1]
## recipeNames(rcp1)
```

recipeLoad *recipeLoad*

Description

To load data recipe(s) into R environment.

Usage

```
recipeLoad(
  rcp = c(),
  cachePath = "ReUseDataRecipe",
  env = .GlobalEnv,
  return = TRUE
)
```

Arguments

rcp	The (vector of) character string of recipe name or file path (`recipeNames()` or `mcols()$fpath` column of the `recipeHub` object returned from `recipeSearch`).
cachePath	A character string for the recipe cache. Must match the one specified in `recipeUpdate()`. Default is "ReUseDataRecipe".
env	The R environment to export to. Default is `.GlobalEnv`.
return	Whether to return the recipe to a user-assigned R object. Default is TRUE, where user need to assign a variable name to the recipe. e.g., `rcp1 <- recipeLoad()`. If FALSE, it loads the recipe and uses its original name, and user doesn't need to assign a new name. e.g., `recipeLoad(return=TRUE)`. If multiple recipes are to be loaded, `return=FALSE` must be used.

Value

A data recipe of `cwlProcess` S4 class, which is ready to be evaluated in *R*.

Examples

```
########################
## Load single recipe
########################

library(Rcwl)
recipeUpdate()
recipeSearch("liftover")
rcp <- recipeLoad("ensembl_liftover")
Rcwl::inputs(rcp)
rm(rcp)

gencode_annotation <- recipeLoad("gencode_annotation")
```

```
inputs(gencode_annotation)
rm(gencode_annotation)

########################
## Load multiple recipes
########################

rcphub <- recipeSearch("gencode")
recipeNames(rcphub)
recipeLoad(recipeNames(rcphub), return=FALSE)
inputs(gencode_transcripts)
```

`recipeMake` *recipeMake*

Description

Constructor function of data recipe

Usage

```
recipeMake(
  shscript = character(),
  paramID = c(),
  paramType = c(),
  outputID = c(),
  outputType = c("File[]"),
  outputGlob = character(0),
  requireTools = character(0)
)
```

Arguments

`shscript`	character string. Can take either the file path to the user provided shell script, or directly the script content, that are to be converted into a data recipe.
`paramID`	Character vector. The user specified parameter ID for the recipe.
`paramType`	Character vector specifying the type for each `paramID`. One parameter can be of multiple types in list. Valid values are "int" for integer, "boolean" for boolean, "float" for numeric, "File" for file path, "File[]" for an array of files, etc. Can also take "double", "long", "null", "Directory". See details.
`outputID`	the ID for each output.
`outputType`	the output type for each output.
`outputGlob`	the glob pattern of output files. E.g., "hg19.*".
`requireTools`	the command-line tools to be used for data processing/curation in the user-provided shell script. The value here must exactly match the tool name. E.g., "bwa", "samtools", etc. A particular version of that tool can be specified in the format of "tool=version", e.g., "samtools=1.3".

Details

For parameter types, more details can be found here: "https://www.commonwl.org/v1.2/CommandLineTool.html#CWLType"

`recipeMake` is a convenient function for wrapping a shell script into a data recipe (in `cwlProcess` S4 class). Please use `Rcwl::cwlProcess` for more options and functionalities, especially when the recipe gets complicated, e.g., needs a docker image for a command-line tool, or one parameter takes multiple types, etc. Refer to this recipe as an example: https://github.com/rworkflow/ReUseDataRecipe/blob/master/reference

Value

a data recipe in `cwlProcess` S4 class with all details about the shell script for data processing/curation, inputs, outputs, required tools and corresponding docker files. It is readily taken by `getData()` to evaluate the shell scripts included and generate the data locally. Find more details with `?Rcwl::cwlProcess`.

Examples

```
## Not run:
library(Rcwl)
##############
### example 1
##############

script <- "
input=$1
outfile=$2
echo \"Print the input: $input\" > $outfile.txt
"
rcp <- recipeMake(shscript = script,
                  paramID = c("input", "outfile"),
                  paramType = c("string", "string"),
                  outputID = "echoout",
                  outputGlob = "*.txt")
inputs(rcp)
outputs(rcp)
rcp$input <- "Hello World!"
rcp$outfile <- "outfile"
res <- getData(rcp, outdir = tempdir(),
               notes = c("echo", "hello", "world", "txt"),
               showLog = TRUE)
readLines(res$out)

##############
### example 2
##############

shfile <- system.file("extdata", "gencode_transcripts.sh", package = "ReUseData")
readLines(shfile)
rcp <- recipeMake(shscript = shfile,
                  paramID = c("species", "version"),
                  paramType = c("string", "string"),
                  outputID = "transcripts",
                  outputGlob = "*.transcripts.fa*",
```

```
                    requireTools = c("wget", "gzip", "samtools")
                    )
Rcwl::inputs(rcp)
rcp$species <- "human"
rcp$version <- "42"
res <- getData(rcp,
        outdir = tempdir(),
        notes = c("gencode", "transcripts", "human", "42"),
        showLog = TRUE)
res$output
dir(tempdir())

## End(Not run)
```

---

recipeSearch *recipeSearch*

Description

Search existing data recipes.

Usage

```
recipeSearch(keywords = character(), cachePath = "ReUseDataRecipe")
```

Arguments

keywords	character vector of keywords to be matched to the recipe names. If not specified, function returns the full recipe list.
cachePath	A character string for the recipe cache. Must match the one specified in `recipeUpdate()`. Default is "ReUseDataRecipe".

Value

A `recipeHub` object.

Examples

```
recipeSearch()
recipeSearch("gencode")
recipeSearch(c("STAR", "index"))
```

recipeUpdate *recipeUpdate*

Description

Function to sync and get the most updated and newly added data recipes through the pubic "rworkflow/ReUseDataRecipe" GitHub repository or user-specified private GitHub repository.

Usage

```
recipeUpdate(
  cachePath = "ReUseDataRecipe",
  force = FALSE,
  remote = FALSE,
  repos = "rworkflow/ReUseDataRecipe"
)
```

Arguments

cachePath	A character string specifying the name for the BiocFileCache object to store the ReUseData recipes. Once specified here, must use the same for cachePath argument in recipeSearch, and recipeLoad. Default is "ReUseDataRecipe".
force	Whether to remove existing and regenerate recipes cache. Default is FALSE. Only use if any old recipes that have been previously cached locally are updated remotely (on GitHub repos).
remote	Whether to download the data recipes directly from a GitHub repository. Default is FALSE.
repos	The GitHub repository containing data recipes that are to be synced to local cache. Only works when remote=TRUE. Default is "rworkflow/ReUseDataRecipe" GitHub repository where public data recipes are saved, which might be more up-to-date than the recipes contained inReUseData package. It can also be a private GitHub repository where users save their own data recipes.

Value

a recipeHub object.

Examples

```
## recipeUpdate()
## recipeUpdate(force=TRUE)
## recipeUpdate(force = TRUE, remote = TRUE)
```

ReUseData *ReUseData*

Description

ReUseData is an *R/Bioconductor* software tool to provide a systematic and versatile approach for standardized and reproducible data management. ReUseData facilitates transformation of shell or other ad hoc scripts for data preprocessing into workflow-based data recipes. Evaluation of data recipes generate curated data files in their generic formats (e.g., VCF, bed). Both recipes and data are cached using database infrastructure for easy data management and reuse. Prebuilt data recipes are available through ReUseData portal ("https://rcwl.org/dataRecipes/") with full annotation and user instructions. Pregenerated data are available through ReUseData cloud bucket that is directly downloadable through "getCloudData()".

Index

[,dataHub,ANY,ANY,ANY-method *(dataHub)*, 3
[,recipeHub,ANY,ANY,ANY-method *(recipeHub)*, 11
[<-,dataHub,ANY,ANY,ANY-method *(dataHub)*, 3

annData, 2

c,dataHub-method *(dataHub)*, 3

dataHub, 3
dataNames *(dataHub)*, 3
dataNotes *(dataHub)*, 3
dataParams *(dataHub)*, 3
dataPaths *(dataHub)*, 3
dataSearch, 5
dataTags *(dataHub)*, 3
dataTags,dataHub-method *(dataHub)*, 3
dataTags<- *(dataHub)*, 3
dataTags<-,dataHub-method *(dataHub)*, 3
dataUpdate, 6
dataYml *(dataHub)*, 3

getCloudData, 8
getData, 9

meta_data, 10

recipeHub, 11
recipeLoad, 12
recipeMake, 13
recipeNames *(recipeHub)*, 11
recipeSearch, 15
recipeUpdate, 16
ReUseData, 17

show,dataHub-method *(dataHub)*, 3
show,recipeHub-method *(recipeHub)*, 11

toList *(dataHub)*, 3