Package: qtl2cl (via r-universe)

May 7, 2026

Version 0.23-1

Date 2023-08-24

Title Command-Line Interface for R/qtl2

Description Support for a command-line interface for R/qtl2.

Author Karl W Broman [aut, cre]
(<https://orcid.org/0000-0002-4914-6671>)

Maintainer Karl W Broman <broman@wisc.edu>

Depends R (>= 3.1.0)

Imports qtl2 (>= 0.33-1), qtl2convert (>= 0.8), optparse, jsonlite, yaml

Suggests testthat, devtools, roxygen2

License GPL-3

URL https://github.com/rqtl/qtl2cl

LazyData true

Encoding UTF-8

ByteCompile true

RoxygenNote 7.2.3

Roxygen list(markdown=TRUE)

Repository https://kbroman.r-universe.dev

Date/Publication 2024-10-07 12:53:28 UTC

RemoteUrl https://github.com/rqtl/qtl2cl

RemoteRef HEAD

RemoteSha 30936c4ea95ef44c522b2b080f6405107aff91d0

Contents

`cross2rds` *Read cross and save as rds*

Description

Read cross files and save as rds

Usage

```
cross2rds(input_file, output_file, compress = FALSE)
```

Arguments

`input_file`	Character string with path to the YAML or JSON file containing all of the control information. This could instead be a zip file containing all of the data files, in which case the contents are unzipped to a temporary directory and then read.
`output_file`	Character string with path to RDS file for output
`compress`	If TRUE, save a compressed RDS file (smaller but slower).

Examples

```
input_file <- paste0("https://github.com/rqtl/qtl2data/",
                     "blob/master/B6BTBR/b6btbr.zip")
## Not run: cross2rds(input_file, "b6btbr.rds")
```

`read_file` *Read a file*

Description

Read a file that is either RDS, CSV, JSON, or YAML, with the method determined by the file extension.

Usage

```
read_file(file, ...)
```

Arguments

`file`	Character string of input file; must have file extension `.rds`, `.csv`, `.json`, or `.yaml`.
`...`	Passed to whatever function it calls

Details

For CSV files, we use `qtl2::fread_csv()`, which calls `data.table::fread()` with a particular set of options; note that columns are forced to be numeric in this case. For RDS files, we use `base::readRDS()`. For JSON files, we use `base::readLines()` and `jsonlite::fromJSON()`. For YAML files, we use `yaml::yaml.load_file()`.

Value

Could be most anything; whatever gets read in by the corresponding R function for the file type. See Details.

Examples

```
## Not run: w <- read_file("myfile.rds")
x <- read_file("myfile.csv")
y <- read_file("myfile.json")
z <- read_file("myfile.yaml")
## End(Not run)
```

`run_calcgenoprob` *Run calc_genoprob and save result to rds*

Description

Run calc_genoprob and save result to rds

Usage

```
run_calcgenoprob(
  cross_file,
  output_file,
  map_file = NULL,
  step = 0,
  off_end = 0,
  stepwidth = c("fixed", "max"),
  error_prob = 0.0001,
  map_function = c("haldane", "kosambi", "c-f", "morgan"),
  cores = 1,
  compress = FALSE
)
```

Arguments

cross_file	Character string with path to RDS file containing cross
output_file	Character string with path to RDS file for output
map_file	Character string with path to RDS file for writing genetic map (with inserted pseudomarkers)
step	Distance between pseudomarkers and markers; if step=0 no pseudomarkers are inserted.
off_end	Distance beyond terminal markers in which to insert pseudomarkers.
stepwidth	Indicates whether to use a fixed grid (stepwidth="fixed") or to use the maximal distance between pseudomarkers to ensure that no two adjacent markers/pseudomarkers are more than step apart.
error_prob	Assumed genotyping error probability
map_function	Character string indicating the map function to use to convert genetic distances to recombination fractions.
cores	Number of CPU cores to use, for parallel calculations. (If 0, use parallel::detectCores().) Alternatively, this can be links to a set of cluster sockets, as produced by parallel::makeCluster().
compress	If TRUE, save compressed RDS files (smaller but slower).

Examples

```
input_file <- paste0("https://github.com/rqtl/qtl2data/",
                     "blob/master/B6BTBR/b6btbr.zip")
## Not run: cross2rds(input_file, "b6btbr.rds")
## Not run: run_calcgenoprob("b6btbr.rds", "b6btbr_probs.rds", "b6btbr_gmap.rds")
```

---

run_calckinship *Calculate kinship matrix*

Description

Calculate genetic similarity among individuals (kinship matrix) from conditional genotype probabilities.

Usage

```
run_calckinship(
  input_file,
  output_file,
  type = c("overall", "loco", "chr"),
  omit_x = FALSE,
  use_allele_probs = TRUE,
  cores = 1,
  compress = FALSE
)
```

Arguments

input_file	Input RDS file containing genotype or allele probabilities
output_file	Output RDS file for calculated kinship matrix
type	Indicates whether to calculate the overall kinship ("overall", using all chromosomes), the kinship matrix leaving out one chromosome at a time ("loco"), or the kinship matrix for each chromosome ("chr").
omit_x	If TRUE, only use the autosomes; ignored when type="chr".
use_allele_probs	If TRUE, assess similarity with allele probabilities (that is, first run qtl2::genoprob_to_alleleprob()); otherwise use the genotype probabilities.
cores	Number of CPU cores to use, for parallel calculations. (If 0, use parallel::detectCores().) Alternatively, this can be links to a set of cluster sockets, as produced by parallel::makeCluster().
compress	If TRUE, save a compressed RDS file (smaller but slower).

Examples

```
input_file <- paste0("https://github.com/rqtl/qtl2data/",
                     "blob/master/B6BTBR/b6btbr.zip")
## Not run: cross2rds(input_file, "b6btbr.rds")
## Not run: run_calcgenoprob("b6btbr.rds", "b6btbr_probs.rds")
## Not run: run_calckinship("b6btbr_probs.rds", "b6btbr_kinship.rds")
```

---

run_getXcovar *Get X covariates and write to file*

Description

Read cross from RDS file, determine X chromosome covariates, and write to another RDS file.

Usage

```
run_getXcovar(input_file, output_file, compress = FALSE)
```

Arguments

input_file	Input RDS file for cross
output_file	Output RDS file for X chromosome covariates
compress	If TRUE, save a compressed RDS file (smaller but slower).

Examples

```
input_file <- paste0("https://github.com/rqtl/qtl2data/",
                     "blob/master/B6BTBR/b6btbr.zip")
## Not run: cross2rds(input_file, "b6btbr.rds")
## Not run: run_getXcovar("b6btbr.rds", "b6btbr_xcovar.rds")
```

run_gp2ap *Run genoprob_to_alleleprob*

Description

Read in genotype probabilities, convert them to allele probabilities, and write them back out.

Usage

```
run_gp2ap(input_file, output_file, cores = 1, compress = FALSE)
```

Arguments

input_file	Name of input file (should be RDS)
output_file	Name of output file (will be RDS)
cores	Number of CPU cores to use, for parallel calculations. (If 0, use parallel::detectCores().) Alternatively, this can be links to a set of cluster sockets, as produced by parallel::makeCluster().
compress	If TRUE, save a compressed RDS file (smaller but slower).

Examples

```
input_file <- paste0("https://github.com/rqtl/qtl2data/",
                     "blob/master/B6BTBR/b6btbr.zip")
## Not run: cross2rds(input_file, "b6btbr.rds")
## Not run: run_calcgenoprob("b6btbr.rds", "b6btbr_probs.rds")
## Not run: run_gp2ap("b6btbr_probs.rds", "b6btbr_aprobs.rds")
```

run_scan1 *Run scan1*

Description

Read in a bunch of data and then run qtl2::scan1().

Usage

```
run_scan1(
  genoprobs_file,
  pheno_file,
  output_file = NULL,
  map_file = NULL,
  kinship_file = NULL,
  addcovar_file = NULL,
  Xcovar_file = NULL,
  intcovar_file = NULL,
```

```
  weights_file = NULL,
  reml = TRUE,
  cores = 1,
  compress = FALSE
)
```

Arguments

genoprobs_file	Name of file with genotype probabilities
pheno_file	Name of file with phenotypes
output_file	Optional output RDS file. If NULL, print output as a table.
map_file	Optional (RDS) file containing map. Needed if output_file is NULL.
kinship_file	Optional file containing kinship matrix
addcovar_file	Optional file containing additive covariates
Xcovar_file	Optional file containing X chromosome covariates
intcovar_file	Optional file containing interactive covariates
weights_file	Optional file containing covariates
reml	If TRUE, use REML; otherwise, use maximum likelihood
cores	Number of CPU cores to use
compress	If TRUE, save a compressed RDS file (smaller but slower).

Index