Package ‘gdxtools’

October 22, 2026

Title Manipulate 'GDX' Files

Type Package

Version 1.2.0

Date 2026-10-20

SystemRequirements GAMS (>= 24.2.1) is only required if you call gams()

Description Read and write 'GDX' files ('GAMS' data exchange) and convert parameters, sets and variables to data frames. Backed by the 'GAMS'-maintained 'gamstransfer' package; no compiled code is shipped. See <https://www.gams.com/latest/docs/UG_GDX.html> for the 'GDX' format.

License EPL-1.0

URL https://github.com/lolow/gdxtools

BugReports https://github.com/lolow/gdxtools/issues

Imports gamstransfer (>= 3.0)

Suggests testthat, microbenchmark

Encoding UTF-8

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Laurent Drouet [aut, cre]

Maintainer Laurent Drouet <ldrouet@pm.me>

Repository CRAN

Date/Publication 2026-10-21 23:40:02 UTC

Contents

all_items 2
batch_extract 2
extract 3
extract.gdx 4

gams 5
gdx 5
igdx 6
load_records 7
write.gdx 7
write2.gdx 9

Index **10**

`all_items` *Return the list of all items*

Description

Return the list of all items

Usage

```
all_items(x, ...)
```

Arguments

`x`	the gdx object
`...`	ignored

Value

A named `list` with four character vectors of symbol names: `variables`, `parameters`, `sets` and `equations`.

Author(s)

Laurent Drouet

`batch_extract` *Extract a list of items from many GDX*

Description

Extract a list of items from many GDX

Usage

```
batch_extract(items, files = NULL, gdxs = NULL, ...)
```

Arguments

items	vector or list of items to extract
files	list of files; if NULL, gdxs must be supplied
gdxs	list of gdx objects; if NULL, files must be supplied
...	passed to extract.gdx

Value

A named list with one data.frame per requested item, each the row-bind of that item extracted from every file with a gdx column identifying the source file.

Author(s)

Laurent Drouet

Examples

```
f1 <- tempfile(fileext = ".gdx")
f2 <- tempfile(fileext = ".gdx")
write.gdx(f1, list(myparam = data.frame(i = c("a", "b"), value = 1:2)))
write.gdx(f2, list(myparam = data.frame(i = c("a", "b"), value = 3:4)))
allparam <- batch_extract("myparam", files = c(f1, f2))
```

---

extract *Extract data from a gdx*

Description

Extract data from a gdx

Usage

```
extract(x, ...)
```

Arguments

x	the gdx object
...	arguments passed to extract.gdx

Value

A data.frame; see extract.gdx for the column layout and attributes.

`extract.gdx` *Extract parameter or variable data from the gdx file*

Description

Extract parameter or variable data from the gdx file

Usage

```
## S3 method for class 'gdx'
extract(x, item, field = "l", addgdx = FALSE, ...)
```

Arguments

`x`	the gdx object
`item`	the name of the item to extract
`field`	the field of the variable to be extracted. Can be 'l', 'm', 'lo', 'up' (level, marginal, lower, upper). Defaults to level.
`addgdx`	if `TRUE`, append a `gdx` column with the filename.
`...`	ignored; for backward compatibility.

Value

A `data.frame` with one character column per domain index plus a numeric `value` column (parameters, variables and equations); sets have no `value` column. A `gams` attribute carries the symbol's description text, and `addgdx = TRUE` adds a `gdx` column with the source filename.

Author(s)

Laurent Drouet

Examples

```
f <- tempfile(fileext = ".gdx")
write.gdx(f, list(travel_cost = data.frame(city = c("paris", "lyon"),
                                           value = c(12, 7))))
mygdx <- gdx(f)
travel_cost <- mygdx["travel_cost"]
travel_cost <- extract(mygdx, "travel_cost")
```

gams *Run the GAMS executable*

Description

Thin wrapper around `system2("gams", ...)`. Returns the integer exit status from GAMS.

Usage

```
gams(gmsAndArgs)
```

Arguments

`gmsAndArgs`	command-line string passed to `gams`

Value

exit status (integer)

gdx *GDX file*

Description

Constructs a `gdx` object backed by a `gamstransfer::Container`. By default the file is opened in **lazy mode**: only symbol metadata (names, dimensions, domains, descriptions) is read up front. Each symbol's records are loaded on first access via `[.gdx` / `extract()` and then cached on the container for subsequent calls. Pass `lazy = FALSE` to read everything eagerly (legacy 1.0.0 behavior).

Usage

```
gdx(filename, lazy = TRUE, ...)
```

Arguments

`filename`	filename of the gdx file
`lazy`	if `TRUE` (default), defer reading symbol records until the symbol is actually accessed. If `FALSE`, load all records at open time.
`...`	extra fields stored on the resulting object

Value

An object of class `gdx`: an S3 list holding the live `gamstransfer::Container`(s) and per-type symbol metadata (`variables`, `parameters`, `sets`, `equations` data.frames with `name`, `text` and `dim` columns).

Author(s)

Laurent Drouet

Examples

```
f <- tempfile(fileext = ".gdx")
write.gdx(f, list(demand = data.frame(city = c("paris", "lyon"),
                                      value = c(50, 20))))
mygdx <- gdx(f)                 # lazy by default
mygdx["demand"]                 # triggers a targeted read
eager <- gdx(f, lazy = FALSE)
```

`igdx` *GAMS sysdir locator (kept for backward compatibility)*

Description

Pre-1.0 gdxtools relied on the gdxrrw API and required the user to point it at a GAMS install. The gamstransfer backend is self-contained and does not need this. `igdx` now simply discovers `gams` on the PATH (or uses the directory the user supplies) and returns it; calling it has no side effect on gdx I/O.

Usage

```
igdx(gamsSysDir = NULL, silent = FALSE, returnStr = FALSE)
```

Arguments

`gamsSysDir`	path to the GAMS system directory (optional)
`silent`	if `TRUE`, do not print the discovered path
`returnStr`	if `TRUE`, return the discovered path as a string

Details

Deprecated since 1.2.0: it is not needed by `gdx()` or `write.gdx()` and will be removed in a future release. Use `Sys.which("gams")` to locate GAMS.

Value

the discovered GAMS system directory, or `""` if none found

load_records *Eagerly load records for one or more symbols*

Description

Lazy-opened gdx objects defer reading records until first access. When you know up front which symbols you will use (or you want to amortize the I/O cost), call `load_records()` to read them in a single batched `gamstransfer::Container$read()` call.

Usage

```
load_records(x, symbols = NULL)
```

Arguments

`x`	a gdx object
`symbols`	character vector of symbol names; `NULL` (default) means every symbol in the file.

Value

`x` invisibly. Records are stored on the underlying container.

Author(s)

Laurent Drouet

Examples

```
f <- tempfile(fileext = ".gdx")
write.gdx(f, list(a = data.frame(i = c("x", "y"), value = 1:2),
                  b = data.frame(i = c("x", "y"), value = 3:4)))
g <- gdx(f)
load_records(g, c("a", "b"))
g["a"]  # already cached, no I/O
```

write.gdx *Write a list of parameters / sets / variables to a GDX*

Description

Builds a `gamstransfer::Container` from the supplied data and writes it to `file`. Variables can be given as separate level / lower / upper data.frames (keyed by the same variable name across the three lists); missing entries fall back to the gamstransfer defaults (level 0, lower -Inf, upper +Inf for free variables).

Usage

```
write.gdx(
  file,
  params = list(),
  vars_l = list(),
  vars_lo = list(),
  vars_up = list(),
  sets = list(),
  removeLST = TRUE,
  usetempdir = TRUE,
  digits = 16,
  compress = FALSE,
  na = c("drop", "keep", "error"),
  dup = c("first", "last", "error")
)
```

Arguments

file	the output gdx filename
params	named list of parameter data.frames
vars_l	named list of variable level data.frames
vars_lo	named list of variable lower-bound data.frames
vars_up	named list of variable upper-bound data.frames
sets	named list of set data.frames
removeLST	kept for backward compatibility; ignored.
usetempdir	kept for backward compatibility; ignored.
digits	kept for backward compatibility; ignored (gamstransfer preserves full numeric precision).
compress	when TRUE, write a compressed gdx.
na	how to handle NA / NaN values in parameter value columns: "drop" (default, legacy v0.7 behavior: NA rows are silently discarded and GAMS reads 0 for those keys), "keep" (preserve as GAMS NA / undef), or "error" (stop with an informative message).
dup	how to collapse duplicate index keys: "first" (default, matches legacy write2.gdx/wgdx behavior), "last" (matches the legacy write.gdx GAMS-process path, where each assignment overwrote the previous one), or "error" (stop so the caller can dedupe upstream). When rows are dropped a warning reports the count.

Value

Invisibly returns 0; called for the side effect of writing the gdx file at file.

Author(s)

Laurent Drouet

Examples

```
param1 <- data.frame(x = c('1', '2'), value = 1:2)
param2 <- data.frame(a = c('london', 'paris'), value = c(50, 0.2))
write.gdx(tempfile(fileext = ".gdx"),
          list(param1 = param1, param2 = param2))
```

`write2.gdx` *Write parameters and sets to a gdx (alias of write.gdx for backward compatibility)*

Description

Historically this was a faster path that bypassed the GAMS process used by the legacy `write.gdx`. With the gamstransfer backend both functions use the same fast path; this entry point is kept for code that calls it explicitly.

Usage

```
write2.gdx(
  file,
  params = list(),
  sets = list(),
  na = c("drop", "keep", "error"),
  dup = c("first", "last", "error")
)
```

Arguments

`file`	the output gdx filename
`params`	named list of parameter data.frames
`sets`	named list of set data.frames
`na`	how to handle NA / NaN values; see `write.gdx`.
`dup`	how to collapse duplicate index keys; see `write.gdx`.

Value

Invisibly returns `0`; called for the side effect of writing the gdx file at `file`.

Author(s)

Laurent Drouet

Index

all_items, 2

batch_extract, 2

extract, 3
extract.gdx, *3*, 4

gams, 5
gdx, 5

igdx, 6

load_records, 7

write.gdx, 7, *9*
write2.gdx, 9