Package: glpsol (via r-universe)

October 17, 2026

Type Package

Title Bundled 'GLPK' Solver for 'MathProg', LP and MPS Models

Version 0.1.0.9000

Description Ships the 'glpsol' stand-alone solver of the GNU Linear Programming Kit ('GLPK') with R and runs it as a separate process with a time limit and an optional live log. 'glpsol' reads 'MathProg' ('GMPL') models with separate data files, 'CPLEX' LP and MPS files, and solves them with the simplex, interior-point or branch-and-cut methods, writing the solution to a file. Windows and macOS binary builds carry a static copy of 'GLPK'; other platforms link against the system library.

License GPL-3

Encoding UTF-8

SystemRequirements GLPK (>= 5.0): libglpk-dev (deb), glpk-devel (rpm), glpk (brew)

URL https://optimal2050.github.io/glpsol/, https://github.com/optimal2050/glpsol

BugReports https://github.com/optimal2050/glpsol/issues

Suggests testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.1.1

Config/pak/sysreqs libglpk-dev

Repository https://optimal2050.r-universe.dev

Date/Publication 2026-10-16 22:53:49 UTC

RemoteUrl https://github.com/optimal2050/glpsol

RemoteRef HEAD

RemoteSha 6a0cb3df78f3ee603e2ac822853aa6347fc2e8f5

Contents

glpk_version 2
glpsol_path 2
glpsol_read_solution 3
glpsol_run 4

Index **5**

`glpk_version` *Version of the linked GLPK library*

Description

Version of the linked GLPK library

Usage

```
glpk_version()
```

Value

A character string, e.g. ‘"5.0"‘.

Examples

```
glpk_version()
```

`glpsol_path` *Path to the bundled glpsol executable*

Description

The package installs GLPK’s stand-alone solver ‘glpsol‘ next to its shared library. Use this path wherever a ‘glpsol‘ command is expected.

Usage

```
glpsol_path()
```

Value

Normalised path to the executable. An error is raised if it is missing from the installation.

See Also

[glpsol_run()]

Examples

```
glpsol_path()
```

---

glpsol_read_solution *Read a glpsol solution file*

Description

Reads the plain-text solution that ‘glpsol‘ writes with ‘-w‘ (basic, interior-point or integer, whichever the run produced) into data frames. That file identifies rows and columns by index only; pass the problem file written by ‘–wglp‘ in the same run to attach their names.

Usage

```
glpsol_read_solution(file, names = NULL)
```

Arguments

file	Path to the solution file written by ‘glpsol -w‘.
names	Optional path to the problem file written by ‘glpsol –wglp‘ in the same run. Its ‘n i‘/‘n j‘ lines give the row and column names.

Value

A list of class ‘"glpsol_solution"‘ with * ‘kind‘: ‘"basic"‘, ‘"interior"‘ or ‘"integer"‘, * ‘status‘: ‘"optimal"‘, ‘"feasible"‘, ‘"infeasible"‘, ‘"no feasible"‘ or ‘"undefined"‘, * ‘primal_status‘, ‘dual_status‘: the two parts of a basic solution’s status (‘NA‘ for the other kinds), * ‘objective‘: the objective value, * ‘problem‘, ‘objective_name‘: names from the problem file, if given, * ‘rows‘, ‘cols‘: data frames with ‘index‘, ‘name‘, ‘status‘, ‘primal‘ and ‘dual‘. ‘status‘ is the basis status (‘"basic"‘, ‘"lower"‘, ‘"upper"‘, ‘"free"‘, ‘"fixed"‘) and is ‘NA‘ for non-basic solution kinds; ‘dual‘ is ‘NA‘ for integer solutions.

See Also

[glpsol_run()]

Examples

```
mod <- system.file("models", "transp.mod", package = "glpsol")
out <- tempfile("glpsol-"); dir.create(out)
sol <- file.path(out, "transp.txt")
glp <- file.path(out, "transp.glp")

glpsol_run(c("-m", mod, "-w", sol, "--wglp", glp), echo = FALSE)
s <- glpsol_read_solution(sol, names = glp)
s$status
s$objective
s$cols[, c("name", "primal", "dual")]
```

```
unlink(out, recursive = TRUE)
```

`glpsol_run` *Run the bundled glpsol executable*

Description

Runs ‘glpsol‘ as a separate process with the given command-line arguments. A crash or a runaway solve cannot take down the R session, and ‘timeout‘ stops it on every platform.

Usage

```
glpsol_run(args = "--help", wd = NULL, timeout = 0, echo = TRUE)
```

Arguments

`args`	Character vector of command-line arguments, as for the ‘glpsol‘ CLI, e.g. ‘c("-m", "model.mod", "-d", "model.dat", "-o", "model.sol")‘. Each element is quoted for the shell, so file names may contain spaces.
`wd`	Working directory for the run; relative paths in ‘args‘ are resolved against it. ‘NULL‘ uses the current directory.
`timeout`	Time limit in seconds for the process; ‘0‘ means no limit. Unlike ‘–tmlim‘, this also covers model translation and output writing.
`echo`	If ‘TRUE‘, the solver log is printed to the console as it runs. If ‘FALSE‘, it is captured and returned in ‘log‘.

Value

Invisibly, a list with * ‘status‘: exit status, ‘0‘ on success, * ‘timed_out‘: ‘TRUE‘ if ‘timeout‘ was reached, * ‘log‘: captured output lines (‘character(0)‘ when ‘echo = TRUE‘).

See Also

[glpsol_path()]

Examples

```
mod <- system.file("models", "transp.mod", package = "glpsol")
res <- glpsol_run(c("-m", mod), echo = FALSE)
res$status
tail(res$log, 3)
```

Index

glpk_version, 2
glpsol_path, 2
glpsol_read_solution, 3
glpsol_run, 4