Package: clusters (via r-universe)

June 5, 2026

Type Package

Title Utilities to manage, estimate speed, optimize use of parallel clusters

Description Miscellaneous utilities developed by the Predictive Ecology Group (<http://predictiveecology.org>).

URL http://clusters.predictiveecology.org, https://github.com/PredictiveEcology/clusters

Date 2025-12-23

Version 0.0.22

Depends R (>= 4.2)

Imports data.table, parallel, parallelly, reproducible (>= 2.1.2.9045), Require

Suggests knitr, roxygen2, qs, rmarkdown, testthat

Remotes PredictiveEcology/reproducible@development, PredictiveEcology/Require@development

Encoding UTF-8

Language en-CA

License GPL-3

VignetteBuilder knitr, rmarkdown

BugReports https://github.com/PredictiveEcology/clusters/issues

ByteCompile yes

RoxygenNote 7.3.3

Roxygen list(markdown = TRUE)

Config/pak/sysreqs cmake make libuv1-dev

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

Date/Publication 2026-02-17 20:11:17 UTC

RemoteUrl https://github.com/PredictiveEcology/clusters

RemoteRef HEAD

RemoteSha b24155b0d96db40e80aae96c4fedca9baaff5052

Contents

`clusterSetup` *Setup a cluster for DEoptim*

Description

This includes copying files over to unique(cores) machines, then loading all objects from disk in each of the parallel cores.

Usage

```
clusterSetup(
  messagePrefix = "DEoptim_",
  itermax = 500,
  trace = TRUE,
  strategy = 3,
  initialpop = NULL,
  NP = NULL,
  cores,
  logPath,
  libPath,
  objsNeeded,
  pkgsNeeded,
  nCoresNeeded = 100,
  envir = parent.frame()
)
```

Value

A list of items that can be passed to `DEoptim.control()`

dirNew *Remove duplicate figures, keeping most recent duplicate only*

Description

This is for cleaning up cases where an interrupted optimization is leading to multiple files for the same .runName. This will remove duplicates, keeping only the most recent.

Usage

```
dirNew(
  path,
  secsAgo = Inf,
  after = Sys.time() - secsAgo,
  pattern = "^(.+)\\_[[:digit:]]{6,8}.*\\.png"
)
```

Arguments

path	A folder in which to search for duplicates
pattern	The regular expression to search for, to identify the files. This must have 1 set of parentheses (), as only the content between the () will be used for duplicate assessment, i.e., remove anything in the file that shouldn't be used.
delete	Logical. Default FALSE, which will only list the files that will be deleted. If TRUE, then the identified files will be deleted

Value

For side effects: removed files

makeClusterPSOCK *Lightweight wrapper for parallelly::makeClusterPSOCK*

Description

Lightweight wrapper for parallelly::makeClusterPSOCK

Usage

```
makeClusterPSOCK(
  workers,
  outfile = NULL,
  rscript_libs = NULL,
  ...,
  port = NULL,
```

```
  rshopts = c("-o", "ExitOnForwardFailure=yes"),
  tries = 5L,
  delay = 5,
  renice = 20,
  revtunnel = TRUE
)
```

Arguments

workers	The hostnames of workers (as a character vector) or the number of localhost workers (as a positive integer).
outfile	Optional log file path.
rscript_libs	Optional library paths for workers.
...	Additional arguments passed to parallelly::makeClusterPSOCK.
port	Optional port or port block start.
tries, delay	Maximum number of attempts done to launch each node with makeNode() and the delay (in seconds) in-between attempts. If argument port specifies more than one port, e.g. port = "random" then a random port will be drawn and validated at most tries times. Arguments tries and delay are used only when setup_strategy == "sequential".
renice	A numerical 'niceness' (priority) to set for the worker processes.
revtunnel	If TRUE, a reverse SSH tunnel is set up for each worker such that the worker R process sets up a socket connection to its local port (port + rank - 1) which then reaches the master on port port. If FALSE, then the worker will try to connect directly to port port on master. If NA, then TRUE or FALSE is inferred from inspection of rshcmd[1]. For more details, see below.

---

monitorCluster *Watch active threads across PSOCK workers and track per-core peaks*

Description

Continuously queries each worker for the number of active threads via clusters::numActiveThreads() and displays a single, aligned status line that updates in place. It also tracks the **peak** thread count per core (named by the cores vector). On interrupt (Ctrl-C), it prints a final aligned summary of peak values and returns them invisibly.

Usage

```
monitorCluster(cl, cores, pad = 2, interval = 1)
```

Arguments

`cl`	A PSOCK cluster object (e.g., created with `parallelly::makeClusterPSOCK()`) connected to the remote machines corresponding to `cores`.
`cores`	A character vector of core (host) names. The order and names define column layout and are used to name the returned peak vector.
`pad`	Integer number of spaces to insert *between* columns (default: `2`). Padding is not part of the alignment width.
`interval`	Numeric number of seconds to wait between refreshes (default: `1`).

Details

The display renders a one-line header of core names (left-aligned), followed by a live one-line status of current counts (right-aligned within the width of each core name). Lines are redrawn in-place using ANSI erase sequences (`\033[2K\r`) which are supported in RStudio Server and most ANSI-aware terminals.

- The function calls `clusters::numActiveThreads()` on each worker. If a worker errors, that tick treats the value as `NA` for display and as `0` when updating the peak (so peaks are not reduced by `NA`s).
- The header is printed once. Each subsequent update clears and redraws only the live values line using ANSI sequences. If ANSI is not supported by the current console, the escape codes may appear literally; in that case, run the function in RStudio Server, a modern terminal, or adapt it to use fallbacks.
- The function runs until interrupted (e.g., `Ctrl-C`). On interrupt, it clears the live line, prints the final peaks aligned under the header, and returns the named peak vector invisibly.

Value

Invisibly returns a named integer vector of peak active thread counts, with names matching `cores`. Also prints a final aligned summary on interrupt.

Requirements on workers

The **clusters** package must be installed on each worker. This function issues `parallel::clusterEvalQ(cl, requireNamespace("clusters"))` once to assert availability, but it does not install packages remotely.

Note

This function uses ANSI escape sequences to clear and redraw a single line (`"\033[2K\r"`). RStudio Server consoles interpret ANSI by default. If your output is redirected (non-TTY) or ANSI is disabled, consider adapting the clearing step to use a fallback (e.g., overwrite with spaces).

See Also

`makeClusterPSOCK`, `clusterEvalQ`, `numActiveThreads`

Examples

```
## Not run:
if (FALSE) {
library(parallelly)
library(parallel)

cores <- c("birds", "biomass", "camas", "carbon", "caribou", "coco",
           "core", "dougfir", "fire", "mpb", "sbw", "mega",
           "acer", "abies", "pinus")

# Create a PSOCK cluster over SSH (example; configure to your environment)
cl <- parallelly::makeClusterPSOCK(
 workers = cores,
  rshcmd = "ssh",
   homogeneous = FALSE
 )

# Watch and track peaks; press Ctrl-C to stop.
peak <- monitorCluster(cl, cores, pad = 2, interval = 1)

# After interrupt, 'peak' is a named integer vector with per-core maxima.
print(peak)

stopCluster(cl)
}

## End(Not run)
```

---

numActiveThreads *Estimate the number of active threads currently being used*

Description

This only works on non-linux operating systems, as it uses `ps`

Usage

```
numActiveThreads(pattern = "", minCPU = 50)
```

Arguments

`pattern`	An optional search pattern to look for when identifying threads that are active. If left at default, then all threads that are active will count.
`minCPU`	The minimum CPU (in percent , i.e.,0 to 100) that a thread must be using for it to count as actively being used.

Value

An integer representing the current number of threads that are being used as a CPU% greater than `minCPU`. This can be used, e.g., with `parallelly::availableCores()` to estimate the number of cores that are available to be used.

Note

This does not address memory or disk use issues.

Examples

```
# This will show the active number, updated every 0.5 seconds
cat("Number Active CPUs right now:\n");
while(TRUE) {
  numAC <- clusters::numActiveThreads();
  cat("\r"); cat(numAC); cat("  ");
  Sys.sleep(0.5)
}
```

`plotMachine` *Plot the outputs from* `testMachine`

Description

Plot the outputs from `testMachine`

Usage

```
plotMachine(
  Ncores,
  coreTimes,
  estTotTime,
  optimisticTotTime,
  systemTimes,
  N = 100,
  nam,
  detectedCores
)
```

Examples

```
par(mfrow = c(2,length(hosts)))
Map(out = outs, nam = names(outs), function(out, nam)
    do.call(plotMachine, append(out, list(nam = nam))))
```

`resourcesUsed` *Assess the machine resources used*

Description

Uses `vmstat` (must be installed; it is by default on linux).

Usage

```
resourcesUsed(machines = "localhost", resource = "us")
```

Arguments

`machines`	Character vector of the name(s) of the PSOCK resource to query, e.g., `"n168"`
`resources`	Column extracted in vmstat. Defaults to `"us"` or "user CPU"

Value

Returns the outputs from `vmstat`

`rmIncompleteDups` *This is for cleaning up cases where an interrupted optimization is leading to multiple files for the same .runName. This will remove duplicates, keeping only the most recent.*

Description

This is for cleaning up cases where an interrupted optimization is leading to multiple files for the same .runName. This will remove duplicates, keeping only the most recent.

Usage

```
rmIncompleteDups(
  path,
  pattern = "^(.+)\\_[[:digit:]]{5,8}.*\\.png",
  delete = FALSE
)
```

Arguments

`path`	A folder in which to search for duplicates
`pattern`	The regular expression to search for, to identify the files. This must have 1 set of parentheses (), as only the content between the () will be used for duplicate assessment, i.e., remove anything in the file that shouldn't be used.
`delete`	Logical. Default `FALSE`, which will only list the files that will be deleted. If `TRUE`, then the identified files will be deleted

Value

For side effects: removed files

`runTests` *Runs test on each machine in* `hosts`

Description

Runs test on each machine in `hosts`

Usage

```
runTests(
  hosts,
  repos = c("predictiveecology.r-universe.dev", getOption("repos")),
  clustersBranch = "main",
  RscriptPath = "/usr/local/bin/Rscript",
  Npops = 100
)
```

Value

a list; same as `getHostCombination` return.

`summaryOutputFolder` *Summary – wrapper around* `dirNew` *and* `tableFiles`

Description

Convenient wrapper.

Usage

```
summaryOutputFolder(
  path,
  pattern = "^.+hists/(.+)\\_iter.+\\_[[:digit:]]{6,8}.*\\.png"
)
```

Arguments

`path`	A folder in which to search for duplicates
`pattern`	The regular expression to base the `table` on. It should have one and only one parenthesis i.e., `(.+)`, which will be the basis of the `table`. Everything outside of the `(.+)` will be removed

Value

Tabulation of the files, based on the pattern.

tableFiles *Tabulate the filenames into groups based on* pattern

Description

Using a regular expression, with a single (.+) identifying the parts of the filenames to keep, and therefore to base the table on. Everything before

Usage

```
tableFiles(
  files,
  pattern = "^.+hists/(.+)\\_iter.+\\_[[:digit:]]{6,8}.*\\.png"
)
```

Arguments

files	A vector of full filenames
pattern	The regular expression to base the table on. It should have one and only one parenthesis i.e., (.+), which will be the basis of the table. Everything outside of the (.+) will be removed

Value

Tabulation of the files, based on the pattern.

termsInDEoptim termsInDEoptim

Description

termsInDEoptim

Usage

```
termsInDEoptim(fireSense_spreadFormula, thresh, numParams)
```

Arguments

fireSense_spreadFormula	The formula to be submitted to DEoptim::DEoptim(), from e.g., sim$fireSense_spreadFormula.
thresh	The threshold for accepting fits; e.g., from mod$thresh.
numParams	The number of parameters (TODO: improve description)

testMachine *Test machines*

Description

Test machines

Usage

```
testMachine(NcoresMax = parallel::detectCores(), N = 100, thinning = 5)
```

Examples

```
# example code
hosts <- c("97", "106", "184", "189", "213", "217", "220")#, "102")
hosts <- makeHosts(ipbase = "spades", hosts)
hosts <- c(hosts, "132.156.148.105", "localhost")
cl <- parallelly::makeClusterPSOCK(hosts,
                                   rscript = c("nice", "/usr/local/bin/Rscript"))
parallel::clusterExport(cl, c("testMachine", "pkgs"))
parallel::clusterEvalQ(cl, {
  if (!require("Require")) install.packages("Require",
                                   repos = c("https://predictiveecology.r-universe.dev",
                                             getOption("repos")))
  Require::Require(pkgs)
})

st <- system.time(outs <- parallel::clusterApply(cl, seq_along(cl), function(x)
  testMachine(N = 1, NcoresMax = 10, thinning = 3)))

names(outs) <- hosts
print(st)
getHostCombination(outs, Npops = 100)
```

visualizeDE *Make histograms of* DEoptim *object* pars

Description

Make histograms of DEoptim object pars

Usage

```
visualizeDE(DE, cachePath, titles, lower, upper)
```

Arguments

DE	An object from a DEoptim call
cachePath	A cacheRepo to pass to showCache and loadFromCache if DE is missing.

Index