

Package: PEcAn.workflow (via r-universe)

October 16, 2026

Type Package

Title PEcAn Functions Used for Ecological Forecasts and Reanalysis

Version 1.10.0.9000

Description The Predictive Ecosystem Carbon Analyzer (PEcAn) is a scientific workflow management tool that is designed to simplify the management of model parameterization, execution, and analysis. The goal of PEcAn is to streamline the interaction between data and models, and to improve the efficacy of scientific investigation. This package provides workhorse functions that can be used to run the major steps of a PEcAn analysis.

URL <https://pecanproject.github.io>

BugReports <https://github.com/PecanProject/pecan/issues>

License BSD_3_clause + file LICENSE

Imports dplyr, PEcAn.data.atmosphere, PEcAn.data.land, PEcAn.data.remote, PEcAn.DB, PEcAn.logger, PEcAn.remote, PEcAn.settings, PEcAn.uncertainty, PEcAn.utils, purrr (>= 0.2.3), rlang, XML

Suggests mockery, testthat, withr

Copyright Authors

Encoding UTF-8

RoxygenNote 7.3.3

X-schema.org-keywords PEcAn, data-assimilation, ecosystem-modeling, reproducibility

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev make libmagick++-dev gsfonts jags libicu-dev libpng-dev libuv1-dev libxml2-dev libnetcdf-dev libssl-dev libproj-dev python3 libsqlite3-dev libudunits2-dev libx11-dev xz-utils zlib1g-dev

Repository <https://pecanproject.r-universe.dev>

Date/Publication 2026-10-16 08:17:39 UTC

RemoteUrl <https://github.com/PecanProject/pecan>

RemoteRef HEAD
RemoteSha fdb9b89d6133f78473728f63b7c1b3deb4ee29ec
RemoteSubdir base/workflow

Contents

do_conversions	2
run.write.configs	3
runModule.get.trait.data	4
runModule.run.write.configs	5
start_model_runs	6
Index	7

do_conversions	<i>do_conversions</i>
----------------	-----------------------

Description

Input conversion workflow

Usage

```
do_conversions(  
  settings,  
  overwrite.met = FALSE,  
  overwrite.fia = FALSE,  
  overwrite.ic = FALSE  
)
```

Arguments

settings PEcAn settings list
overwrite.met, overwrite.fia, overwrite.ic
 logical

Author(s)

Ryan Kelly, Rob Kooper, Betsy Cowdery, Istem Fer

run.write.configs	<i>Write model-specific run scripts and configuration files</i>
-------------------	---

Description

Generates run scripts and configuration files for all analyses (ensemble and/or sensitivity analysis) specified in the provided settings. Delegates the model-specific config writing to the appropriate `write.config.*` function (e.g. `write.config.ED2`, `write.config.SIPNET`).

Usage

```
run.write.configs(
  settings,
  ensemble.size,
  input_design,
  write = TRUE,
  posterior.files = rep(NA, length(settings$pfts)),
  overwrite = TRUE,
  samples = NULL
)
```

Arguments

<code>settings</code>	a PEcAn settings list
<code>ensemble.size</code>	number of ensemble runs
<code>input_design</code>	Input design data.frame coordinating input files across runs. Contains columns for each sampled input (met, param, etc.) with row indices, as documented in runModule.run.write.configs .
<code>write</code>	should the runs be written to the database?
<code>posterior.files</code>	Filenames for posteriors for drawing samples for ensemble and sensitivity analysis (e.g. <code>post.distns.Rdata</code> , or <code>prior.distns.Rdata</code>).
<code>overwrite</code>	logical: Replace output files that already exist?
<code>samples</code>	Optional pre-computed parameter samples (a list with <code>trait.samples</code> , <code>sa.samples</code> , <code>ensemble.samples</code>). When supplied, these are used directly and <code>samples.Rdata</code> is not read from disk. When NULL (default), the function falls back to loading <code>samples.Rdata</code> from <code>settings\$outdir</code> .

Details

Upstream contract (reads from `settings$outdir`):

`samples.Rdata` Produced by [get.parameter.samples](#). Contains 5 bundled objects: `trait.samples`, `sa.samples`, `ensemble.samples`, `runs.samples`, `env.samples`. This function loads `trait.samples` and `sa.samples` to build model configuration files. If `input_design` contains a `param` column, `ensemble.samples` is rebuilt by subsetting `trait.samples` according to the design indices.

File-based side effects (saved to settings\$outdir):

sensitivity.samples.<ensemble_id>.Rdata Contains sa.run.ids (named list of run IDs per PFT/trait/quantile), sa.ensemble.id, sa.samples, pft.names, and trait.names. Saved when sensitivity analysis is configured.

ensemble.samples.<ensemble_id>.Rdata Contains ens.run.ids (vector of run IDs), ens.ensemble.id, ens.samples, pft.names, and trait.names. Saved when ensemble is configured.

runs_manifest.csv A CSV table tracking all runs created, appended across ensemble and SA analyses.

Downstream contract: The sensitivity.samples.*.Rdata and ensemble.samples.*.Rdata files are loaded by [get.results](#) to match model outputs to their corresponding parameter sets. This implicit file-based coupling is a refactoring target.

The default value for posterior.files is NA, in which case the most recent posterior or prior (in that order) for the workflow is used. When specified, posterior.files should be a vector of filenames with one entry for each PFT. Specify filenames with no path; PFT outdirs will be appended. This forces use of only files within this workflow, to avoid confusion.

Value

The settings list (invisibly), updated with ensemble IDs for SA and ensemble analysis (e.g. settings\$sensitivity.analysis\$ensemble.id, settings\$ensemble\$ensemble.id).

Author(s)

David LeBauer, Shawn Serbin, Ryan Kelly, Mike Dietze, Akash B V

```
runModule.get.trait.data
```

Get trait data for all PFTs in a settings list

Description

Get trait data for all PFTs in a settings list

Usage

```
runModule.get.trait.data(settings)
```

Arguments

settings PEcAn configuration list. Must have class ‘Settings’ or ‘MultiSettings’

runModule.run.write.configs

Generate model-specific run configuration files for one or more PEcAn runs

Description

This function serves as the orchestration layer between PEcAn workflows and the config-writing machinery. It generates appropriate input designs (ensemble and/or SA) if not provided. For MultiSettings, it generates designs once from the first site then shares across all sites for consistent sampling. Finally, it delegates to [run.write.configs](#) for actual config generation. The input design determines how parameter samples and input files (met, soil, etc.) are coordinated across runs. Ensemble designs typically use random or quasi-random sampling, while SA designs hold non-parameter inputs constant (OAT methodology).

Usage

```
runModule.run.write.configs(settings, overwrite = TRUE, input_design = NULL)
```

Arguments

- | | |
|--------------|---|
| settings | a PEcAn Settings or MultiSettings object |
| overwrite | logical: Replace config files if they already exist? |
| input_design | <p>Optional. The parameter/input design for the runs, normally the full result of <code>generate_joint_ensemble_design()</code>: a list with <code>design_matrix</code> (a data.frame whose param column selects rows of <code>trait.samples/ensemble.samples</code>, plus optional columns named for <code>settings\$run\$inputs</code> tags such as met or soil) and samples (the parameter bundle those indices point into). Can be:</p> <ul style="list-style-type: none"> • The <code>list(design_matrix, samples)</code> returned by <code>generate_joint_ensemble_design()</code> • NULL to generate the design and samples internally from settings |

A bare design data.frame is not accepted: its param indices are only meaningful together with the samples they index into, so the design and its samples must travel together.

Value

A modified settings object, invisibly

start_model_runs	<i>Start selected ecosystem model runs within PEcAn workflow</i>
------------------	--

Description

Note: When using a queue system, PEcAn queries each job individually every ‘check_interval’ seconds. The default interval of ten seconds was chosen for small sets of jobs in a fast-moving queue, and if used for large runs it can create excessive load. If you expect the whole run to take hours, consider checking much less often (every few minutes?) instead.

Usage

```
start_model_runs(
  settings,
  write = TRUE,
  stop.on.error = TRUE,
  check_interval = 10
)

runModule_start_model_runs(settings, stop.on.error = TRUE, ...)
```

Arguments

settings	PEcAn settings object
write	(logical) Whether or not to write to the database. Default TRUE.
stop.on.error	logical: Kill entire run if any job throws an error?
check_interval	Time in seconds to wait between queries of qsub job status.
...	passed on to start_model_runs

Functions

- runModule_start_model_runs(): A lightweight wrapper around ‘start_model_runs’ that takes ‘write’ from ‘settings’ instead of as a separate argument.

Author(s)

Shawn Serbin, Rob Kooper, David LeBauer, Alexey Shiklomanov

Examples

```
## Not run:
settings <- PEcAn.settings::read.settings("pecan.xml")
start_model_runs(settings)

## End(Not run)
```

Index

`do.conversions (do_conversions)`, [2](#)
`do_conversions`, [2](#)

`get.parameter.samples`, [3](#)
`get.results`, [4](#)

`run.write.configs`, [3](#), [5](#)
`runModule.get.trait.data`, [4](#)
`runModule.run.write.configs`, [3](#), [5](#)
`runModule_start_model_runs`
 `(start_model_runs)`, [6](#)

`start_model_runs`, [6](#)