

Package: apsimR (via r-universe)

October 15, 2026

Title APSIM Next Generation as a Contract-Emitting Analysis Member

Version 0.3.2.9000

Description A typed, server-capable R interface to APSIM Next Generation, the process-based agricultural systems simulator. Drives the simulator for prediction over scenario designs, mechanism-parameter estimation, parameter calibration against observations (a bounded optimiser, or an iterative ensemble smoother when 'PESTO' is present), global sensitivity analysis (Morris and Sobol), Gaussian-process emulation with leave-one-out diagnostics, validation against observations (goodness-of-fit metrics and a typed verdict), and the generation of known-causal-structure datasets for testing causal-inference methods, and returns results as a versioned, provenance-complete 'apsim_manifest' object that is emitted natively into the shared orchestra manifest contract, so simulator output composes directly with downstream calibration, uncertainty-quantification and causal-inference tools; an ensemble inversion additionally bridges to a 'PESTO' ensemble manifest. Execution degrades gracefully: every entry point abstains with a typed reason when the simulator or its runtime is absent.

License MIT + file LICENSE

URL <https://github.com/max578/apsimR>

BugReports <https://github.com/max578/apsimR/issues>

Depends R (>= 4.1.0)

Imports DBI, digest, jsonlite, lhs, processx, RSQLite, S7, sensitivity, stats, tools, utils

Suggests ggplot2, knitr, optimix (>= 0.1.0), PESTO (>= 0.6.0), rmarkdown, testthat (>= 3.0.0)

Remotes max578/PESTO

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8
Language en-AU
Roxygen list(markdown = TRUE)
SystemRequirements APSIM Next Generation (>= 2024.1) and the .NET 8.0 Runtime, for the bundled 'Models' and 'apsim-server' executables. The package builds, installs and checks without them; entry points that need the simulator abstain when it is absent.
Config/roxygen2/version 8.1.0
Config/pak/sysreqs cmake make libicu-dev libuv1-dev zlib1g-dev
Repository <https://max578.r-universe.dev>
Date/Publication 2026-10-15 23:00:52 UTC
RemoteUrl <https://github.com/max578/apsimR>
RemoteRef main
RemoteSha e8e953a4992f8dabe204437faca530647f06fc64

Contents

apsim_abstention	3
apsim_available	4
apsim_calibrate	4
apsim_configure	6
apsim_design	7
apsim_edit	8
apsim_emulate	8
apsim_estimate	10
apsim_example	11
apsim_external_facts	11
apsim_fact_status	12
apsim_forward_model	12
apsim_ground_truth	13
apsim_manifest	15
apsim_output	16
apsim_predict	17
apsim_read	18
apsim_reports	18
apsim_run	19
apsim_sensitivity	20
apsim_server	21
apsim_sim	22
apsim_simulations	23
apsim_validate	23
apsim_validation_plot	24
apsim_version	25
apsim_write	26

<i>apsim_abstention</i>	3
as_orchestra_manifest	26
as_pesto_manifest	27
is_apsim_abstention	28
manifest_summary	29
orchestra_manifest	30
verify_manifest	30
Index	32

apsim_abstention	<i>Construct a typed abstention</i>
------------------	-------------------------------------

Description

Returns a small classed object signalling that a simulator-dependent operation could not be performed, with a machine-readable reason. Used in place of an error so that callers (and the orchestration layer) can branch on a refusal rather than catch an exception.

Usage

```
apsim_abstention(reason, detail = NA_character_, scope = "call")
```

Arguments

reason	Character scalar reason code. One of "runtime_unavailable" (no Models executable or .NET runtime found), "run_failed" (the simulator was invoked but did not complete), or "no_output" (the run produced no readable DataStore).
detail	Character scalar human-readable detail, or NA.
scope	Character scalar naming what was refused (default "call").

Value

An object of class "apsim_abstention": a list with reason, detail, scope and abstained = TRUE.

Examples

```
a <- apsim_abstention("runtime_unavailable", "no Models executable found")
is_apsim_abstention(a)
```

apsim_available	<i>Is the APSIM simulator available?</i>
-----------------	--

Description

Reports whether a runnable Models executable has been found (by `apsim_configure()` or auto-discovery). When this is FALSE, the simulator-dependent verbs return an `apsim_abstention()` rather than running.

Usage

```
apsim_available()
```

Value

A single logical.

Examples

```
if (apsim_available()) {
  apsim_version()
}
```

apsim_calibrate	<i>Calibrate APSIM parameters against observations</i>
-----------------	--

Description

Estimates the `parm_paths` parameters whose APSIM-simulated observations best match observed, over the box `[lower, upper]`. The `ies` backend (default when PEST0 is installed) returns a posterior parameter ensemble through an iterative ensemble smoother; the `optim` backend returns a single bounded least-squares point estimate with no extra dependency. Several targets are calibrated jointly – pass a named observed vector and, optionally, per-target `obs_sd` or `weights`.

Usage

```
apsim_calibrate(
  sim,
  parm_paths,
  lower,
  upper,
  observed,
  output,
  obs_sd = NULL,
  weights = NULL,
```

```

    backend = c("auto", "ies", "optim"),
    n_real = 50L,
    noptmax = 4L,
    inflation = 0.8,
    report = NULL,
    seed = NULL
  )

```

Arguments

<code>sim</code>	An <code>apsim_sim</code> object or a path to an <code>.apsimx</code> file.
<code>parm_paths</code>	Character vector of APSIM node paths to estimate.
<code>lower, upper</code>	Numeric box bounds aligned to <code>parm_paths</code> (recycled from length one).
<code>observed</code>	Named numeric vector of observed target values; the names must match what output produces.
<code>output</code>	Either a function(<code>report</code>) $\rightarrow$ named numeric reducing a run's report to the comparable observations, or a character vector of report column names (see apsim_forward_model()).
<code>obs_sd</code>	Numeric observation standard deviation(s): a scalar or a vector aligned to <code>observed</code> . Defaults to ten percent of each <code> observed </code> (floored away from zero) when <code>NULL</code> .
<code>weights</code>	Optional misfit weights aligned to <code>observed</code> for the <code>optim</code> backend; defaults to $1 / \text{obs_sd}^2$. The <code>ies</code> backend always weights by $1 / \text{obs_sd}$.
<code>backend</code>	Character: "auto" (default – <code>ies</code> if PESTO is installed, else <code>optim</code>), "ies", or "optim".
<code>n_real</code>	Integer ensemble size for the <code>ies</code> backend.
<code>noptmax</code>	Integer number of ensemble-smoother iterations (<code>ies</code>).
<code>inflation</code>	Numeric RTPS inflation strength in $[0, 1]$ for the <code>ies</code> backend (0 disables it); guards against ensemble collapse.
<code>report</code>	Character report-table name, or <code>NULL</code> for the first report.
<code>seed</code>	Optional integer RNG seed (prior ensemble / reproducibility).

Value

An `apsim_manifest` with `inferential_target = "parameters"` whose `params` holds the point estimate (`optim`) or the posterior ensemble (`ies`), or an [apsim_abstention\(\)](#) when the simulator (or, for `ies`, PESTO) is unavailable.

Examples

```

# A calibration runs the simulator many times, so the example is wrapped in
# \dontrun{}; see the package vignette for a runnable walk-through.
## Not run:
f <- apsim_example("Wheat")
if (!is.na(f)) {
  apsim_calibrate(

```

```
f, parm_paths = "[Fertilise at sowing].Script.Amount",
lower = 0, upper = 250, observed = c(Yield = 2200), output = "Yield",
backend = "optim", seed = 1L)
}

## End(Not run)
```

apsim_configure

*Configure the APSIM runtime explicitly***Description**

Pins the paths apsimR uses, overriding auto-discovery. Call this once per session when APSIM is installed in a non-standard location or when more than one version is present and a specific one is wanted. With all arguments NULL it clears any pinned paths and reverts to discovery.

Usage

```
apsim_configure(models = NULL, server = NULL, dotnet_root = NULL)
```

Arguments

models	Character path to the Models executable, or NULL to discover.
server	Character path to the apsim-server executable, or NULL.
dotnet_root	Character path to the .NET runtime root (sets DOTNET_ROOT for child processes), or NULL to discover ~/.dotnet-x64.

Value

Invisibly, the resolved runtime list (see [apsim_available\(\)](#)).

Examples

```
# Revert to auto-discovery.
apsim_configure()
```

apsim_design

*Build an experimental design over an APSIM parameter space***Description**

Generates parameter points filling the box [lower, upper] for the named parameters. A Latin-hypercube design (the default) spreads n points evenly through the space – the standard choice for training an emulator; a grid gives an exhaustive factorial (n becomes points-per-dimension); random gives an independent uniform sample. The design is reproducible through seed.

Usage

```
apsim_design(
  parm_paths,
  lower,
  upper,
  n = 20L,
  method = c("lhs", "grid", "random"),
  seed = NULL
)
```

Arguments

parm_paths	Character vector of APSIM node paths (the design columns).
lower, upper	Numeric vectors of box bounds, each aligned to parm_paths (recycled from length one).
n	Integer. For method = "lhs" / "random", the number of points; for method = "grid", the number of points per dimension (so the design has $n^{\text{length}(\text{parm_paths})}$ rows).
method	Character: "lhs" (default), "grid" or "random".
seed	Optional integer RNG seed for the stochastic methods.

Value

A numeric matrix with $\text{length}(\text{parm_paths})$ named columns, one row per design point.

Examples

```
d <- apsim_design(c("[Sow].Script.Population", "[Fertilise].Amount"),
  lower = c(50, 0), upper = c(200, 200), n = 8, seed = 1L)
dim(d)
```

apsim_edit	<i>Apply [Node].Property = value edits through APSIM's validator</i>
------------	--

Description

Writes an APSIM --apply config file from edits and runs the simulator's own editor, so the change is validated against the model schema rather than poked into raw JSON. Requires the simulator; returns an `apsim_abstention()` when it is absent.

Usage

```
apsim_edit(x, edits, path = tempfile(fileext = ".apsimx"))
```

Arguments

x	An <code>apsim_sim</code> object.
edits	A named character vector of edits, names being APSIM node paths and values the new values, for example <code>c("[Clock].End" = "1991-12-31", "[Sow].Script.Population" = "120")</code> .
path	Character output path for the edited file. Defaults to a temporary <code>.apsimx</code> .

Value

A new `apsim_sim` for the edited file, or an `apsim_abstention`.

Examples

```
f <- system.file("extdata", "Wheat.apsimx", package = "apsimR")
if (nzchar(f)) {
  apsim_edit(apsim_sim(f), c("[Clock].End" = "1991-12-31"))
}
```

apsim_emulate	<i>Emulate an expensive APSIM output with a Gaussian-process surrogate</i>
---------------	--

Description

Runs APSIM at a space-filling design over `[lower, upper]`, fits a GP surrogate mapping the `parm_paths` parameters to a single scalar output, and returns it with leave-one-out diagnostics. The default exact backend needs no extra package; `pesto` uses PESTO's surrogate when installed. Predict at new parameters with `predict()`.

Usage

```
apsim_emulate(
  sim,
  parm_paths,
  lower,
  upper,
  output,
  target = NULL,
  n = 30L,
  design = c("lhs", "grid", "random"),
  backend = c("exact", "pesto"),
  report = NULL,
  seed = NULL
)
```

Arguments

<code>sim</code>	An <code>apsim_sim</code> object or a path to an <code>.apsimx</code> file.
<code>parm_paths</code>	Character vector of APSIM node paths (the emulator inputs).
<code>lower, upper</code>	Numeric box bounds aligned to <code>parm_paths</code> (recycled from length one).
<code>output</code>	A function(<code>report</code>) -> named numeric or report column name(s) identifying the output (see apsim_forward_model()).
<code>target</code>	Character naming the output column to emulate when output yields several; defaults to the first.
<code>n</code>	Integer number of design (training) points.
<code>design</code>	Character design method passed to apsim_design() : "lhs" (default), "grid" or "random".
<code>backend</code>	Character: "exact" (default, dependency-free; a marginal-likelihood-fitted ARD exact GP) or "pesto".
<code>report</code>	Character report-table name, or NULL for the first report.
<code>seed</code>	Optional integer RNG seed (design reproducibility).

Value

An `apsim_emulator`, or an [apsim_abstention\(\)](#) when the simulator (or, for `backend = "pesto"`, PESTO) is unavailable.

Examples

```
# Fitting an emulator runs the simulator at every design point, so the example
# is wrapped in \dontrun{}; see the package vignette for a runnable walk-through.
## Not run:
f <- apsim_example("Wheat")
if (!is.na(f)) {
  em <- apsim_emulate(
    f, parm_paths = "[Fertilise at sowing].Script.Amount",
```

```

    lower = 0, upper = 250, output = "Yield", n = 12L, seed = 1L)
  if (!is_apsim_abstention(em)) predict(em, matrix(c(50, 150), ncol = 1L))
}

## End(Not run)

```

apsim_estimate

*Estimate a mechanism by running APSIM across an input***Description**

Runs the simulator on a copy of `sim`, reads the report, and fits a reduced-form response of the outcome `y` on the input `x` (the report must already vary `x`, for example through a factorial over an N-rate). Returns the fitted parameters as a parameters `apsim_manifest()`, or an `apsim_abstention()` when the simulator is unavailable.

Usage

```

apsim_estimate(
  sim,
  x,
  y,
  response = c("mitscherlich", "linear"),
  report = NULL
)

```

Arguments

<code>sim</code>	An <code>apsim_sim</code> object or a path to an <code>.apsimx</code> file.
<code>x</code>	Character name of the input column in the report.
<code>y</code>	Character name of the outcome column in the report.
<code>response</code>	Character mechanism form: "mitscherlich" (default, a saturating dose-response) or "linear".
<code>report</code>	Character report-table name, or NULL for the first report.

Value

An `apsim_manifest` with `inferential_target = "parameters"`, or an `apsim_abstention`.

Examples

```

f <- apsim_example("Wheat")
if (!is.na(f)) {
  # apsim_estimate(f, x = "Nitrogen", y = "Yield")
  apsim_available()
}

```

apsim_example	<i>Path to an installed APSIM example simulation</i>
---------------	--

Description

Returns the path to one of the .apsimx examples shipped with the installed APSIM, so examples and tests can run against a real simulation without the package bundling APSIM's own (separately-licensed) files. Returns NA_character_ when the simulator or the named example is not present.

Usage

```
apsim_example(name = "Wheat")
```

Arguments

name Character example name without extension (default "Wheat").

Value

A single character path, or NA_character_.

Examples

```
apsim_example("Wheat")
```

apsim_external_facts	<i>apsimR's external-authority (APSIM) fact registry</i>
----------------------	--

Description

Enumerates the class-8 facts apsimR asserts about the APSIM simulator – the DataStore SQLite schema, the "Current" checkpoint, report column names, and manager node paths – with the oracle that grounds each and the date it was last diffed against a live APSIM install. A fact is *grounded* only when verified_on is a non-NA date; the DataStore-schema and report-column facts are grounded by the replayed-real-run oracle test, not by a skipped test.

Usage

```
apsim_external_facts()
```

Value

A data.frame with fact_family, asserted_value, oracle_kind, verified_on (Date/NA), evidence_path, and a derived grounding token ("grounded" / "ungrounded").

Examples

```
apsim_external_facts()
```

```
apsim_fact_status
```

Grounding status summary for apsimR's external facts

Description

Collapses [apsim_external_facts\(\)](#) to one row per fact family, so a caller can see at a glance which classes of APSIM fact are grounded by a replayed real run and which are still "ungrounded".

Usage

```
apsim_fact_status()
```

Value

A data.frame with one row per fact_family: n_facts, n_grounded, n_unverified.

Examples

```
status <- apsim_fact_status()
status
# families with at least one unverified fact
status[status$n_unverified > 0, ]
```

```
apsim_forward_model
```

Build an APSIM forward model

Description

Returns a closure of the orchestra-standard forward-model shape: `function(theta) -> obs`, where `theta` is an `nreal x npar` numeric matrix (one row per parameter realisation, columns aligned to `parm_paths`) and `obs` is an `nreal x nobs` numeric matrix. Each row is one APSIM run with the corresponding parameters substituted; failed runs become NA rows. The closure is what [apsim_calibrate\(\)](#), [apsim_sensitivity\(\)](#) and [apsim_emulate\(\)](#) all evaluate, and it is directly accepted by `PEST0::pesto_ies_callback()` and by the `sensitivity` package's `model` argument.

Usage

```
apsim_forward_model(
  sim,
  parm_paths,
  output,
  report = NULL,
  checkpoint = "Current",
  reduce = function(x) x[[length(x)]]
)
```

Arguments

<code>sim</code>	An <code>apsim_sim</code> object or a path to an <code>.apsimx</code> file.
<code>parm_paths</code>	Character vector of APSIM node paths to vary, for example <code>c("[Sow].Script.Population", "[Wheat].Phenology.MinimumLeafNumber")</code> .
<code>output</code>	Either a <code>function(report) -> named numeric</code> reducing a run's report to its observation vector, or a character vector of report column names (reduced by <code>reduce</code>).
<code>report</code>	Character report-table name, or <code>NULL</code> for the first report.
<code>checkpoint</code>	Character checkpoint name (default <code>"Current"</code>).
<code>reduce</code>	When <code>output</code> names columns, the <code>function(numeric) -> numeric(1)</code> reducing each column (default takes the final value).

Value

A function of one argument `theta` (an `nreal x npar` matrix or a length-`npair` vector) returning an `nreal x nobs` numeric matrix, or an `apsim_abstention()` when the simulator is unavailable.

Examples

```
f <- apsim_example("Wheat")

if (!is.na(f)) {
  fm <- apsim_forward_model(f, "[Clock].End", output = "Wheat.AboveGround.Wt")
  if (!is_apsim_abstention(fm)) fm(matrix("1991-04-30", 1L, 1L))
}
```

<code>apsim_ground_truth</code>	<i>Generate a known-causal-structure dataset from APSIM (an OSSE)</i>
---------------------------------	---

Description

Runs each unit under a control and a treatment value of `treatment` – so both potential outcomes are known – then assigns each unit to treatment or control and reveals the corresponding outcome, producing a dataset whose true effects are recorded. Each unit is one row of units, whose columns are further APSIM node paths defining that unit's context (the heterogeneity, and the source of confounding). The result is the test-bench for a causal member: hand it the observed (covariates, `W`, `Y`) and compare its estimate to the recorded average treatment effect.

Usage

```
apsim_ground_truth(
  sim,
  treatment,
  control,
```

```

    treated,
    units,
    output,
    target = NULL,
    assignment = c("random", "confounded"),
    prob = 0.5,
    confounder = NULL,
    confound_strength = 1,
    report = NULL,
    seed = NULL
  )

```

Arguments

<code>sim</code>	An <code>apsim_sim</code> object or a path to an <code>.apsimx</code> file.
<code>treatment</code>	Character APSIM node path of the intervened parameter.
<code>control, treated</code>	Numeric values of treatment under control ($W = 0$) and treatment ($W = 1$).
<code>units</code>	A data.frame whose columns are APSIM node paths and whose rows are the units' contexts (covariates). At least one column.
<code>output</code>	A function(<code>report</code>) -> named numeric or report column name(s) identifying the outcome (see apsim_forward_model()).
<code>target</code>	Character naming the outcome column when output yields several; defaults to the first.
<code>assignment</code>	Character: "random" (treatment independent of covariates – no confounding) or "confounded" (treatment probability driven by a covariate that also affects the outcome).
<code>prob</code>	Numeric treatment probability for assignment = "random".
<code>confounder</code>	Character naming the units column that drives assignment when assignment = "confounded".
<code>confound_strength</code>	Numeric logistic coefficient on the standardised confounder (larger means stronger confounding).
<code>report</code>	Character report-table name, or NULL for the first report.
<code>seed</code>	Optional integer RNG seed (assignment reproducibility).

Value

An `apsim_ground_truth_result`, or an [apsim_abstention\(\)](#) when the simulator is unavailable.

Examples

```

# An OSSE varying sowing population (the unit covariate) under two N rates,
# with treatment confounded by population. Runs the simulator twice per unit.
## Not run:
f <- apsim_example("Wheat")

```

```

units <- data.frame(`[Sow using a variable rule].Script.Population` = c(80, 120,
  160, 200), check.names = FALSE)
apsim_ground_truth(
  f, treatment = "[Fertilise at sowing].Script.Amount",
  control = 0, treated = 150, units = units, output = "Yield",
  assignment = "confounded",
  confounder = "[Sow using a variable rule].Script.Population", seed = 1L)

## End(Not run)

```

apsim_manifest

*An APSIM result as a contract-emitting manifest***Description**

The typed result every verb returns. `inferential_target` follows the orchestra contract enum ("predictions", "parameters"); `params` carries a fitted-parameter table (empty for a pure prediction); `outputs` carries the simulated report; `provenance` pins the APSIM version, an integrity hash, and the seed.

Usage

```

apsim_manifest(
  inferential_target,
  method,
  params = data.frame(),
  outputs = NULL,
  metadata = list(),
  seed = NA_integer_
)

```

Arguments

<code>inferential_target</code>	Character: "predictions", "parameters", "sensitivity", "emulator" or "validation".
<code>method</code>	Character method tag, for example "apsim:predict".
<code>params</code>	A data.frame of fitted parameters (default empty).
<code>outputs</code>	The simulated report (data.frame) or NULL.
<code>metadata</code>	A named list of auxiliary provenance.
<code>seed</code>	Integer RNG seed (NA when none).

Value

An `apsim_manifest` S7 object.

Examples

```
m <- apsim_manifest("parameters", "apsim:estimate:mitscherlich",
                    params = data.frame(ymax = 4.2, rate = 0.018, y0 = 1.1))
m@inferential_target
```

apsim_output	<i>Read a report from the server after a run</i>
--------------	--

Description

Fetches columns of a report table produced by the most recent [apsim_run\(\)](#), decoding the server's typed column stream into a `data.frame`. The named report must contain rows (the server signals an empty report as an error, not an empty frame).

Usage

```
apsim_output(server, table, columns)
```

Arguments

server	An open <code>apsim_server</code> handle.
table	Character report-table name.
columns	Character vector of column names to fetch.

Value

A `data.frame` with one column per requested name.

Examples

```
f <- apsim_example("Wheat")
if (!is.na(f) && apsim_available()) {
  srv <- apsim_server(f)
  if (!is_apsim_abstention(srv) && !is_apsim_abstention(apsim_run(srv))) {
    d <- apsim_output(srv, "Report", "Yield")
  }
  close(srv)
}
```

apsim_predict	<i>Run APSIM forward over a simulation file</i>
---------------	---

Description

Runs the simulator on a copy of `sim` (the source file and its `DataStore` are never touched), reads the requested report from the resulting `DataStore`, and returns it as a predictions `apsim_manifest()`. A single `Models` invocation builds the file once and runs every simulation it contains (including any factorial fan-out) across cores. Returns an `apsim_abstention()` when the simulator is unavailable or the run fails.

Usage

```
apsim_predict(sim, report = NULL, simulations = NULL, cpu = NULL)
```

Arguments

<code>sim</code>	An <code>apsim_sim</code> object or a path to an <code>.apsimx</code> file.
<code>report</code>	Character report-table name, or <code>NULL</code> for the first report.
<code>simulations</code>	Optional regular expression selecting simulations to run (<code>--simulation-names</code>).
<code>cpu</code>	Optional integer thread cap; <code>NULL</code> uses all cores.

Value

An `apsim_manifest` with `inferential_target = "predictions"` carrying the simulated report in outputs, or an `apsim_abstention`.

Examples

```
# Running APSIM over a real simulation file takes several seconds, so the
# call is wrapped in \donttest{}; it runs under R CMD check --run-donttest.
f <- apsim_example("Wheat")

if (!is.na(f)) {
  m <- apsim_predict(f)
  m@inferential_target
}
```

apsim_read	<i>Read a report from an APSIM DataStore</i>
------------	--

Description

Returns one report table joined to its simulation names and filtered to a checkpoint. Reading is a single pushed-down SQL JOIN, so only the requested columns and rows leave SQLite.

Usage

```
apsim_read(db, report = NULL, checkpoint = "Current")
```

Arguments

db	Character path to a .db DataStore.
report	Character report-table name, or NULL (default) for the first report in the file.
checkpoint	Character checkpoint name (default "Current").

Value

A data.frame: the report columns, prefixed with a SimulationName column when the report is simulation-keyed.

Examples

```
# d <- apsim_read(run$db, report = "Report")
```

apsim_reports	<i>List the report tables in an APSIM DataStore</i>
---------------	---

Description

List the report tables in an APSIM DataStore

Usage

```
apsim_reports(db)
```

Arguments

db	Character path to a .db DataStore.
----	------------------------------------

Value

A character vector of report-table names (metadata tables, prefixed with _, are excluded).

Examples

```
# apsim_reports(run$db)
```

apsim_run	<i>Run the server's in-memory simulation, optionally with modified parameters</i>
-----------	---

Description

Runs the .apsimx the server holds, optionally overriding model parameters for this run. The file is not reloaded, so an adaptive sweep pays the load cost once. Read the results with [apsim_output\(\)](#).

Usage

```
apsim_run(server, changes = NULL)
```

Arguments

server	An open apsim_server handle.
changes	A named list of parameter overrides: names are APSIM model paths (for example "[Wheat].Phenology.MinimumLeafNumber"), values are R scalars (numeric, integer, logical, Date, character) or a numeric vector (passed as a double array). NULL (default) runs the file unchanged.

Value

Invisibly TRUE on success; errors carry the simulator's message.

Examples

```
f <- apsim_example("Wheat")
if (!is.na(f) && apsim_available()) {
  srv <- apsim_server(f)
  if (!is_apsim_abstention(srv)) {
    apsim_run(srv)
    close(srv)
  }
}
```

Description

Quantifies how the `parm_paths` parameters drive a single scalar APSIM output over the box `[lower, upper]`. Morris screening (default) ranks the factors by mean absolute elementary effect (`mu_star`) and flags interaction / non-linearity (`sigma`) cheaply; the Sobol-Jansen method decomposes the output variance into first-order (S) and total (T) indices at a higher run cost. The two stages use the sensitivity package over the same forward model as the other verbs.

Usage

```
apsim_sensitivity(
  sim,
  parm_paths,
  lower,
  upper,
  output,
  target = NULL,
  method = c("morris", "sobol", "both"),
  morris_r = 10L,
  morris_levels = 6L,
  sobol_n = 100L,
  nboot = 100L,
  report = NULL,
  seed = NULL
)
```

Arguments

<code>sim</code>	An <code>apsim_sim</code> object or a path to an <code>.apsimx</code> file.
<code>parm_paths</code>	Character vector of APSIM node paths (the factors).
<code>lower, upper</code>	Numeric box bounds aligned to <code>parm_paths</code> (recycled from length one).
<code>output</code>	A function(<code>report</code>) -> named numeric or report column name(s) identifying the output (see apsim_forward_model()).
<code>target</code>	Character naming the single output column to analyse when output yields several; defaults to the first.
<code>method</code>	Character: "morris" (default), "sobol", or "both".
<code>morris_r</code>	Integer number of Morris elementary-effect repetitions.
<code>morris_levels</code>	Integer Morris grid levels.
<code>sobol_n</code>	Integer base sample size for Sobol (total runs $n * (p + 2)$).
<code>nboot</code>	Integer bootstrap replicates for Sobol confidence intervals.
<code>report</code>	Character report-table name, or NULL for the first report.
<code>seed</code>	Optional integer RNG seed.

Value

An `apsim_manifest` with `inferential_target = "sensitivity"` whose `params` holds one row per factor with the requested indices, or an `apsim_abstention()` when the simulator is unavailable.

Examples

```
# Sensitivity analysis runs the simulator r(p + 1) times, so the example is
# wrapped in \dontrun{}; see the package vignette for a runnable walk-through.
## Not run:
f <- apsim_example("Wheat")
if (!is.na(f)) {
  apsim_sensitivity(
    f, parm_paths = "[Fertilise at sowing].Script.Amount",
    lower = 0, upper = 250, output = "Yield", method = "morris",
    morris_r = 4L, seed = 1L)
}

## End(Not run)
```

apsim_server

Start a persistent APSIM server holding a simulation in memory

Description

Launches `apsim-server` over loopback TCP, holding `sim` in memory, and opens a connection to it. The returned handle runs the file repeatedly – optionally with modified parameters – without re-paying APSIM's per-run load cost, which is the throughput path for calibration and uncertainty loops. Returns an `apsim_abstention()` when the server executable is unavailable.

Usage

```
apsim_server(sim, port = NULL, timeout = 30)
```

Arguments

<code>sim</code>	An <code>apsim_sim</code> object or a path to an <code>.apsimx</code> file.
<code>port</code>	Integer loopback port, or <code>NULL</code> (default) to pick a free one.
<code>timeout</code>	Numeric seconds to wait for the server to accept a connection (default 30).

Details

Always close the handle with `close()` to stop the server process.

Value

An `apsim_server` handle (a classed environment), or an `apsim_abstention`.

Examples

```
f <- apsim_example("Wheat")
if (!is.na(f) && apsim_available()) {
  srv <- apsim_server(f)
  if (!is_apsim_abstention(srv)) {
    apsim_run(srv)
    close(srv)
  }
}
```

apsim_sim

A simulation file (.apsimx) as a typed object

Description

Reads an .apsimx (JSON) file into an S7 object holding the absolute file path, the parsed model tree, and the APSIM version that produced it (or NA). The object round-trips: [apsim_write\(\)](#) writes the tree back unchanged.

Usage

```
apsim_sim(file)
```

Arguments

file Character path to an .apsimx file.

Value

An apsim_sim S7 object.

Examples

```
f <- system.file("extdata", "Wheat.apsimx", package = "apsimR")
if (nzchar(f)) {
  sim <- apsim_sim(f)
  apsim_simulations(sim)
}
```

apsim_simulations	<i>List the simulation names in an apsim_sim</i>
-------------------	--

Description

Walks the model tree and returns the Name of every Models.Core.Simulation node. Needs no simulator (it reads the parsed JSON).

Usage

```
apsim_simulations(x)
```

Arguments

x An apsim_sim object.

Value

A character vector of simulation names (possibly empty).

Examples

```
f <- system.file("extdata", "Wheat.apsimx", package = "apsimR")
if (nzchar(f)) apsim_simulations(apsim_sim(f))
```

apsim_validate	<i>Validate simulated output against observations</i>
----------------	---

Description

Pairs predicted with observed values and reports goodness-of-fit metrics plus a typed performance verdict. predicted may be an [apsim_predict\(\)](#) manifest, a data.frame, or a numeric vector; observed a data.frame or numeric vector. When both are data frames and by is given, they are joined on that key (matching simulated and observed rows by, say, date) before scoring; otherwise the two columns are paired by position and must be equal length.

Usage

```
apsim_validate(
  predicted,
  observed,
  predicted_col = NULL,
  observed_col = NULL,
  by = NULL
)
```

Arguments

`predicted` An `apsim_manifest`, `data.frame`, or numeric vector of simulated values.

`observed` A `data.frame` or numeric vector of observed values.

`predicted_col, observed_col` Column names when the corresponding input is a `data.frame`.

`by` Optional key column present in both data frames to join on before pairing.

Value

An `apsim_manifest` with `inferential_target = "validation"`: `params` holds the one-row metrics table, outputs the paired observed/predicted values, and `metadata$verdict` the performance grade.

Verdict caveat

The typed grade in `metadata$verdict` reads on Nash-Sutcliffe efficiency (NSE) alone. The bands are those of Moriasi et al. (2007) [unverified against the source paper], derived for monthly watershed and streamflow simulations, not paddock-scale crop yield; that paper's own recommendation is a joint reading of NSE with `rsr` and `pbias` (both already reported in `params`), not NSE on its own. Treat the grade as a diagnostic starting point, and read all three metrics before trusting it.

References

Moriasi, D. N. et al. (2007) Model evaluation guidelines for systematic quantification of accuracy in watershed simulations. *Transactions of the ASABE* 50(3), 885-900. doi:[10.13031/2013.23153](https://doi.org/10.13031/2013.23153)

Examples

```
obs <- c(2.1, 3.4, 4.0, 5.2)
sim <- c(2.0, 3.6, 3.8, 5.0)
v <- apsim_validate(sim, obs)
v@metadata$verdict
```

`apsim_validation_plot` *Plot observed against predicted for a validation manifest*

Description

Builds an observed-versus-predicted scatter with the 1:1 line, titled with the verdict and key metrics. Requires the optional `ggplot2` package; returns an `apsim_abstention()` when it is absent.

Usage

```
apsim_validation_plot(validation)
```

Arguments

validation An apsim_manifest of inferential_target = "validation" (the result of `apsim_validate()`).

Value

A ggplot object, or an apsim_abstention.

Examples

```
v <- apsim_validate(c(2, 3.6, 3.8, 5), c(2.1, 3.4, 4, 5.2))
apsim_validation_plot(v)
```

apsim_version	<i>Installed APSIM version string</i>
---------------	---------------------------------------

Description

Runs `Models --version` and returns the reported version (for example "2026.5.8046.0"), cached for the session. Returns `NA_character_` when the simulator is not available.

Usage

```
apsim_version()
```

Value

A single character string, or `NA_character_`.

Examples

```
apsim_version()
```

apsim_write	<i>Write an apsim_sim back to an .apsimx file</i>
-------------	---

Description

Serialises the model tree to JSON. With no edits this is a faithful round-trip of the file `apsim_sim()` read. Needs no simulator.

Usage

```
apsim_write(x, path = tempfile(fileext = ".apsimx"))
```

Arguments

<code>x</code>	An <code>apsim_sim</code> object.
<code>path</code>	Character output path. Defaults to a new temporary <code>.apsimx</code> .

Value

Invisibly, the output path.

Examples

```
f <- system.file("extdata", "Wheat.apsimx", package = "apsimR")
if (nzchar(f)) apsim_write(apsim_sim(f), tempfile(fileext = ".apsimx"))
```

as_orchestra_manifest	<i>Emit an apsimR result as an orchestra manifest</i>
-----------------------	---

Description

The federation's emit and adapt generic. The `apsim_manifest()` method lifts any `apsimR` result into the shared contract: the `apsimR` target is mapped onto the contract's target enum (a sensitivity table is a statement about parameters, an emulator and a validation are statements about predictions), the assimilation context of an ensemble result is carried into the contract's weights and `obs_target` slots, and a validation verdict is lifted into the typed summary rather than left buried in metadata.

Usage

```
as_orchestra_manifest(x, ...)
```

Arguments

`x` An `apsim_manifest()` object.

`...` Method-specific arguments. The `apsim_manifest()` method accepts `run_id`, an optional character run identifier; when omitted a content hash is derived, so two identical results get the same identifier.

Details

Nothing is discarded: the original `apsimR` target, the APSIM build that produced the result and `apsimR`'s own payload hash all ride in metadata, so a consumer can trace a contract manifest back to the `apsimR` result it came from.

Value

An `orchestra_manifest` S7 object.

See Also

`verify_manifest()`, `as_pesto_manifest()`

Examples

```
m <- apsim_manifest("validation", "apsim:validate",
  params = data.frame(nse = 0.82, rmse = 0.31),
  outputs = data.frame(observed = c(2, 3.6),
    predicted = c(2.1, 3.4)),
  metadata = list(verdict = "good"))
as_orchestra_manifest(m)
```

<code>as_pesto_manifest</code>	<i>Bridge an <code>apsim_manifest</code> to a PESTO ensemble manifest</i>
--------------------------------	---

Description

Converts to PESTO's `pesto_ensemble_manifest` S7 object (the orchestra's C2 contract) so an APSIM inversion result can be consumed by the PESTO / `kernR` / `flexyBayes` stack.

Usage

```
as_pesto_manifest(x)
```

Arguments

`x` An `apsim_manifest` object.

Details

PESTO's contract is an *ensemble* contract, and the bridge respects that rather than forcing every payload through it. It requires two things, both read from the installed PESTO (0.10.1 at the time of writing) rather than assumed: a method tag drawn from PESTO's own enum, and a parameter ensemble whose rows align one-for-one with a simulated-observation ensemble. Exactly one apsimR result satisfies that – the posterior ensemble from `apsim_calibrate()`'s `ies` backend, which PESTO's own ensemble smoother produced. Every other apsimR result (a prediction, a screening table, an emulator, a validation verdict) is not an ensemble inversion, so the bridge returns a `feature_unsupported` `apsim_abstention()` naming the reason instead of a manifest whose slots would have to be invented. Those results compose through `as_orchestra_manifest()`, the federation's general contract.

The bridged manifest is re-hashed with PESTO's own payload recipe, so `PESTO::verify_manifest()` verifies it; apsimR's own hash is kept in the bridge's provenance chain through the source manifest.

Value

A `PESTO::pesto_ensemble_manifest`, or an `apsim_abstention` with reason `"runtime_unavailable"` (PESTO absent) or `"feature_unsupported"` (the payload is not an ensemble inversion).

See Also

`as_orchestra_manifest()` for the general orchestra contract.

Examples

```
# A prediction is not an ensemble inversion, so the bridge declines.
as_pesto_manifest(apsim_manifest("predictions", "apsim:predict",
                                outputs = data.frame(Yield = 3.8)))
```

<code>is_apsim_abstention</code>	<i>Is an object an apsimR abstention?</i>
----------------------------------	---

Description

Is an object an apsimR abstention?

Usage

```
is_apsim_abstention(x)
```

Arguments

`x` Any object.

Value

A single logical: TRUE when `x` is an `apsim_abstention`.

Examples

```
is_apsim_abstention(apsim_abstention("run_failed"))
is_apsim_abstention(42)
```

manifest_summary

A typed home for a headline verdict

Description

A non-ensemble inference – a validation verdict, a decision – has no meaningful values for the ensemble payload slots and should not ride in params as a one-row pseudo-parameter table. The contract’s optional summary slot carries the headline label, whether the emitter abstained, and the key metrics. It is integrity-hashed with the rest of the payload, so a tampered verdict is detected. The class and field names match the reference contract’s `manifest_summary()` exactly, so a consumer reads an `apsimR` verdict with no special casing.

Usage

```
manifest_summary(
  headline,
  abstained = FALSE,
  metrics = list(),
  abstain_reason = NA_character_
)
```

Arguments

headline	Character scalar: the verdict itself, for example "good".
abstained	Logical scalar: did the emitter decline to stand behind the result?
metrics	Named list of supporting numbers.
abstain_reason	Character scalar reason when abstained is TRUE.

Value

A `manifest_summary` list.

Examples

```
manifest_summary("good", metrics = list(nse = 0.82))
```

orchestra_manifest	<i>The orchestra manifest contract (apsimR-side implementation)</i>
--------------------	---

Description

A versioned, hashed, provenance-complete S7 result object, property-compatible with the federation's reference orchestra_manifest and carrying the same bare class identity, so any orchestra consumer reads an apsimR result without special casing. Build one from an [apsim_manifest\(\)](#) with [as_orchestra_manifest\(\)](#) rather than calling this constructor directly.

Value

An S7 object of class orchestra_manifest.

See Also

[as_orchestra_manifest\(\)](#), [verify_manifest\(\)](#)

verify_manifest	<i>Verify an orchestra manifest's payload integrity</i>
-----------------	---

Description

Recomputes the payload hash from the object's own load-bearing slots and compares it to the stored data_hash. A mismatch means the payload was modified after emission. Matches the reference contract's [verify_manifest\(\)](#), so the check is symmetric across members.

Usage

```
verify_manifest(m)
```

Arguments

m An orchestra_manifest object.

Value

A list with a logical ok and a human-readable message.

See Also

[as_orchestra_manifest\(\)](#)

Examples

```
m <- apsim_manifest("predictions", "apsim:predict",  
                    outputs = data.frame(Yield = c(3.8, 4.1)))  
verify_manifest(as_orchestra_manifest(m))
```

Index

apsim_abstention, 3
apsim_abstention(), 4, 5, 8–10, 13, 14, 17, 21, 24, 28
apsim_available, 4
apsim_available(), 6
apsim_calibrate, 4
apsim_calibrate(), 12, 28
apsim_configure, 6
apsim_configure(), 4
apsim_design, 7
apsim_design(), 9
apsim_edit, 8
apsim_emulate, 8
apsim_emulate(), 12
apsim_estimate, 10
apsim_example, 11
apsim_external_facts, 11
apsim_external_facts(), 12
apsim_fact_status, 12
apsim_forward_model, 12
apsim_forward_model(), 5, 9, 14, 20
apsim_ground_truth, 13
apsim_manifest, 15
apsim_manifest(), 10, 17, 26, 27, 30
apsim_output, 16
apsim_output(), 19
apsim_predict, 17
apsim_predict(), 23
apsim_read, 18
apsim_reports, 18
apsim_run, 19
apsim_run(), 16
apsim_sensitivity, 20
apsim_sensitivity(), 12
apsim_server, 21
apsim_sim, 22
apsim_simulations, 23
apsim_validate, 23
apsim_validate(), 25
apsim_validation_plot, 24
apsim_version, 25
apsim_write, 26
apsim_write(), 22
as_orchestra_manifest, 26
as_orchestra_manifest(), 28, 30
as_pesto_manifest, 27
as_pesto_manifest(), 27
is_apsim_abstention, 28
manifest_summary, 29
orchestra_manifest, 30
predict(), 8
verify_manifest, 30
verify_manifest(), 27, 30