Package 'apsimeval'

September 9, 2026

Type Package

Title Evaluation, Visualisation and Sensitivity Analysis of 'APSIM' Classic Output

Version 0.1.0

Description Reads Agricultural Production Systems sIMulator ('APSIM') Classic 7.x '.out' files, pairs simulated series with sparse observed measurements, and computes the goodness-of-fit statistics used in crop-model calibration and validation, including the decomposition of root mean squared error into systematic and unsystematic components. Produces publication grade figures with 'ggplot2': one-to-one scatter plots, residual and Taylor diagrams, probability of exceedance curves, multi-variable timelines with a secondary axis, and distribution plots, assembled into multi-panel figures at journal column widths in colour, greyscale or line art. Treatment structure encoded in simulation names is parsed into factor columns, and the sensitivity of an output to those factors is quantified by variance decomposition, one-at-a-time analysis or range screening. A built-in 'shiny' interface watches the output directory and refreshes incrementally as 'APSIM' regenerates its files.

License GPL-3

Encoding UTF-8

Depends R (>= 4.1.0)

Imports ggplot2 (>= 3.4.0), stats, utils, tools, grDevices

Suggests patchwork, shiny, DT, bslib, yaml, readxl, multcompView, testthat (>= 3.0.0), knitr, rmarkdown, spelling

Config/testthat/edition 3

RoxygenNote 7.3.1

VignetteBuilder knitr

Language en-GB

NeedsCompilation no

Author Sukamal Sarkar [aut, cre] (ORCID: <https://orcid.org/0000-0002-1438-1778>)

Maintainer Sukamal Sarkar <sukamal.sarkar@gm.rkmvu.ac.in>

Repository CRAN

Date/Publication 2026-09-09 14:10:24 UTC

Contents

`accent_col` *Accent colour for single-series plots, honouring the colour mode*

Description

Accent colour for single-series plots, honouring the colour mode

Usage

```
accent_col()
```

Value

A single colour as a hex string or colour name.

Examples

```
accent_col()
```

`aggregate_sim` *Collapse daily output to seasonal or annual values*

Description

Scenario plots almost always run on aggregated output, not raw daily rows. Each variable needs the right operation: yield takes its value at harvest, rainfall and drainage sum, stress indices average.

Usage

```
aggregate_sim(
  data,
  by = c("sim", "year"),
  vars,
  default = "max",
  crop_only = FALSE,
  crop_col = "crop_name",
  fallow = c("fallow", "?", NA)
)
```

Arguments

`data`	Wide APSIM output (from ‘read_out()‘).
`by`	Grouping columns, typically c("sim", "year") plus factor columns.
`vars`	Named character vector of variable = operation pairs. Operations: "max", "last", "sum", "mean", "min", "first", "n". Unnamed entries default to ‘default‘.
`default`	Operation for variables listed without one.
`crop_only`	Restrict to rows where ‘crop_col‘ is not the fallow label, so a season mean is not diluted by fallow days.
`crop_col`	Column naming the crop on each row.
`fallow`	Values of ‘crop_col‘ treated as fallow.

Value

A data frame with one row per group in ‘by‘ and one column per aggregated variable, carrying ‘units‘ and ‘aggregation‘ attributes.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
ag <- aggregate_sim(sim, by = c("sim", "year", "N"), vars = c(wrr = "max"))
head(ag)
```

apply_factors *Apply a project's stored factor mapping to freshly read output*

Description

Apply a project's stored factor mapping to freshly read output

Usage

```
apply_factors(data, project, col = NULL)
```

Arguments

data	Freshly read simulated output.
project	An 'apsim_project'.
col	Simulation column; defaults to the one stored in the project.
	The point of the whole configuration: new .out files from a re-run get the same treatment columns without the user re-entering anything. Names that do not match the stored pattern are reported rather than silently dropped.

Value

'data' with the project's factor columns added.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
p <- set_project_factors(new_project("demo", d), sim)
head(apply_factors(read_out_dir(d), p))
```

apply_project *Push a project's labels and palette into the session*

Description

Push a project's labels and palette into the session

Usage

```
apply_project(project)
```

Arguments

project	An ‘apsim_project‘.
	Called automatically by ‘load_project()‘. Separated out so the GUI can re-apply after an edit without touching the file.

Value

The project, invisibly.

Examples

```
apply_project(new_project("demo", tempdir()))
```

---

apsim_palette *Colour-blind safe treatment palette*

Description

Okabe-Ito. Register a fixed treatment -> colour map once with ‘set_palette()‘ so every figure in a manuscript agrees.

Usage

```
apsim_palette(n = NULL)
```

Arguments

n	Number of colours to return; NULL gives the full palette.

Value

A character vector of hex colours.

Examples

```
apsim_palette(3)
```

apsim_vars *List variables and units without loading the data block*

Description

Reads only the header, so it stays fast on large daily output. Use it to populate variable pickers in the GUI.

Usage

```
apsim_vars(file)
```

Arguments

file	Path to a .out file.

Value

A data frame with columns ‘variable‘ and ‘unit‘.

Examples

```
f <- system.file("extdata", "Gosaba_Rice_AWD_N120.out", package = "apsimeval")
apsim_vars(f)
```

as_long *Pivot wide APSIM output to long form*

Description

Pivot wide APSIM output to long form

Usage

```
as_long(x, id = c("sim", "Date", "year"), vars = NULL, keep = NULL)
```

Arguments

x	Wide APSIM output.
id	Identifier columns kept alongside the pivoted values.
vars	Variables to pivot; NULL uses every numeric column.
keep	Extra columns (treatment factors) carried through the pivot, so the long form can still be grouped and faceted by them.

Value

A long-format data frame with columns for the identifiers, ‘variable‘, ‘value‘ and ‘unit‘.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
head(as_long(sim, vars = "wrr"))
```

`build_figure` *Build a multi-panel figure from a list of panel specs*

Description

Build a multi-panel figure from a list of panel specs

Usage

```
build_figure(
  specs,
  sim = NULL,
  paired = NULL,
  sens = NULL,
  ncol = NULL,
  nrow = NULL,
  tags = "A",
  keep_legend = "auto",
  legend = "top",
  base_size = 9,
  widths = NULL,
  heights = NULL
)
```

Arguments

`specs`	List of ‘panel_spec‘ objects.
`sim`, `paired`, `sens`	Data sources passed to ‘render_panel()‘.
`ncol`, `nrow`	Panel grid; NULL lets patchwork choose.
`tags`	Tag style: "A", "a", "1", or NULL for none.
`keep_legend`	"auto" keeps one legend per distinct legend variable, an integer vector keeps those panels’ legends, NULL leaves all in place.
`legend`	Legend position.
`base_size`	Type size applied to all panels; tiled panels need smaller type than a full-width single figure.
`widths`, `heights`	Relative panel sizes, passed to patchwork.

Value

A ggplot or patchwork object.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
build_figure(list(panel_spec("box", var = "wrr", group = "N"),
                  panel_spec("density", var = "wrr", group = "N")),
             sim = sim, ncol = 2)
```

---

cache_data *Combined data from a cache, with duplicate Titles resolved*

Description

Combined data from a cache, with duplicate Titles resolved

Usage

```
cache_data(cache)
```

Arguments

cache	An ‘out_cache‘.

Value

A single data frame combining every parsed file, or NULL when the cache is empty.

Examples

```
d <- system.file("extdata", package = "apsimeval")
cc <- refresh(out_cache(d))
head(cache_data(cc))
```

check_levels *Check observed treatment labels against the simulated ones*

Description

Mismatched labels are the commonest reason a join silently returns nothing. Reports what matched, what did not, and suggests near-matches.

Usage

```
check_levels(obs, sim, col = "sim")
```

Arguments

obs	Observed column name.
sim	Simulated output.
col	Column name.

Value

A list with elements ‘matched‘, ‘unmatched‘, ‘suggestions‘ and ‘sim_levels‘.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
obs <- data.frame(sim = "Gosaba_Rice_AWD_N120",
                  Date = as.Date(c("1999-10-31", "2003-10-31",
                                   "2008-10-31", "2013-10-31")),
                  variable = "wrr", value = c(8600, 7400, 9000, 7800),
                  obs_sd = c(300, 280, 350, 310), unit = "kg/ha")
long <- as_long(sim, vars = "wrr")
pr <- pair_obs(long, obs, tol_days = 3)
check_levels(obs, sim)$matched
```

check_variables *Check observed variable names against the simulated ones*

Description

Spreadsheet headings ("Grain_yield", "Biomass") rarely match APSIM variable names ("wrr", "wagt"). An unmatched name pairs with nothing and plots nothing, so this reports the mismatch and suggests the closest simulated variable for each.

Usage

```
check_variables(obs, sim, var_col = "variable")
```

Arguments

obs	Observed data in long form with a ‘variable‘ column.
sim	Simulated output, wide or long.
var_col	Name of the variable column in ‘obs‘.

Value

A list of ‘matched‘, ‘unmatched‘, ‘suggestions‘ and ‘sim_vars‘.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
obs <- data.frame(sim = "Gosaba_Rice_AWD_N120",
                  Date = as.Date(c("1999-10-31", "2003-10-31",
                                   "2008-10-31", "2013-10-31")),
                  variable = "wrr", value = c(8600, 7400, 9000, 7800),
                  obs_sd = c(300, 280, 350, 310), unit = "kg/ha")
long <- as_long(sim, vars = "wrr")
pr <- pair_obs(long, obs, tol_days = 3)
check_variables(obs, sim)$matched
```

---

cld_letters *Compact letter display from a Tukey test*

Description

Attaches significance letters for annotating box and bar plots — the standard requirement for treatment comparisons in agronomy journals.

Usage

```
cld_letters(data, response, group, alpha = 0.05, year_col = "year")
```

Arguments

data	A data frame.
response	Response variable.
group	Column mapped to colour or grouping.
alpha	Fill transparency.
year_col	Name of the year column.

Value

A data frame of levels, significance letters and label heights, or NULL when there are fewer than two levels.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
ag <- aggregate_sim(sim, by = c("sim", "year", "N"), vars = c(wrr = "max"))
cld_letters(ag, "wrr", "N")
```

---

combine_plots *Combine panels with shared legend and A/B/C tags*

Description

Combine panels with shared legend and A/B/C tags

Usage

```
combine_plots(
  ...,
  ncol = NULL,
  nrow = NULL,
  tags = "A",
  keep_legend = 1,
  legend = "top",
  base_size = 9
)
```

Arguments

...	Further arguments.
ncol	Number of columns.
nrow	Number of rows.
tags	Panel tag style.
keep_legend	Which panels keep their legend.
legend	Legend position.
base_size	Base font size in points.

Value

A patchwork object.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
ag <- aggregate_sim(sim, by = c("sim", "year", "N"), vars = c(wrr = "max"))
combine_plots(plot_box(ag, "wrr", "N"), plot_density(ag, "wrr", group = "N"))
```

---

detect_factors *Guess the delimiter and which name positions vary*

Description

Pre-fills the mapping fields in the GUI. Harmless when wrong, since the user overwrites it; right often enough to save typing.

Usage

```
detect_factors(sims, col = "sim")
```

Arguments

sims	Character vector of simulation names (or a data frame + ‘col‘).
col	Name of the simulation column when ‘sims‘ is a data frame.

Value

A list with ‘sep‘, ‘n_fields‘, ‘varying‘, ‘constant‘, and a suggested ‘parts‘ vector.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
detect_factors(sim$sim)
```

---

diff_manifest *Compare two manifests*

Description

Compare two manifests

Usage

```
diff_manifest(old, new)
```

Arguments

old	See Details.
new	See Details.

Value

A list of ‘added‘, ‘changed‘, ‘removed‘ and ‘unchanged‘ paths, plus ‘any‘ (TRUE if anything moved) and a one-line ‘summary‘.

Examples

```
d <- system.file("extdata", package = "apsimeval")
m <- scan_out(d)
diff_manifest(m, m)$summary
```

---

exceedance *Empirical probability of exceedance*

Description

Weibull plotting position, P(X >= x). Feed straight to a step plot for long-term scenario comparison.

Usage

```
exceedance(x, by = NULL, data = NULL)
```

Arguments

x	Object to operate on.
by	Grouping columns.
data	A data frame.

Value

A data frame of ‘value‘ and exceedance probability ‘prob‘, with a ‘group‘ column when ‘by‘ is given.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
ag <- aggregate_sim(sim, by = c("sim", "year", "N"), vars = c(wrr = "max"))
head(exceedance(ag$wrr))
```

factor_template *Editable mapping template for names with no usable structure*

Description

Returns one row per unique simulation with blank factor columns. Fill it in (in the interface grid or a spreadsheet) and pass it back as ‘map‘.

Usage

```
factor_template(sims, factors = c("treatment"), col = "sim")
```

Arguments

sims	Simulation names, or a data frame plus ‘col‘.
factors	Factor columns.
col	Column name.

Value

A data frame with one row per unique simulation name and an empty column per factor.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
factor_template(sim$sim, factors = c("water", "N"))
```

get_pal *Colours for a set of levels, honouring the current colour mode*

Description

Colours for a set of levels, honouring the current colour mode

Usage

```
get_pal(levels, aes = c("fill", "colour"))
```

Arguments

levels	Character vector of levels.
aes	Which aesthetic the colours are for: "fill" or "colour". In "mono" the two differ - outlines and lines go black while fills keep a grey ramp.

Value

A named character vector of colours, one per level.

Examples

```
get_pal(c("N0", "N120"))
```

`gof` *Goodness-of-fit statistics for observed vs simulated pairs*

Description

Returns the metrics normally requested by reviewers of APSIM calibration and validation work, including the Willmott decomposition of RMSE into systematic and unsystematic components.

Usage

```
gof(obs, pred, data = NULL, by = NULL, min_n = 5, digits = 2)
```

Arguments

`obs`, `pred`	Numeric vectors of equal length, or column names in 'data'.
`data`	Optional data frame holding obs/pred (and grouping columns).
`by`	Character vector of grouping columns in 'data'.
`min_n`	Statistics that are unstable on tiny samples (R2, NSE, NNSE, KGE, CCC) are returned as NA below this pair count. n is always reported.
`digits`	Decimal places for the returned table (default 2). Use NULL to keep full precision, e.g. when feeding results into further arithmetic.

Value

A data.frame, one row per group.

Examples

```
o <- c(10, 12, 14, 16, 18)
p <- c(11, 11, 15, 15, 19)
gof(o, p)
```

heat_cols — *Sequential and diverging fills for heatmaps, honouring the colour mode*

Description

Sequential and diverging fills for heatmaps, honouring the colour mode

Usage

```
heat_cols(diverging = TRUE)
```

Arguments

diverging	Return a low/mid/high triple for a diverging scale.

Value

A list with elements ‘low‘, ‘mid‘ and ‘high‘.

Examples

```
heat_cols()
```

load_project — *Read a project back*

Description

Read a project back

Usage

```
load_project(file, apply = TRUE)
```

Arguments

file	Path to a ‘.apsimproj‘ file.
apply	Push labels and palette into the session on load.

Value

An object of class ‘apsim_project‘.

Examples

```
p <- new_project("demo", tempdir())
f <- file.path(tempdir(), "demo.apsimproj")
save_project(p, f)
load_project(f)
unlink(f)
```

`map_obs` *Apply a confirmed column mapping and pivot to long form*

Description

Excel dates that arrive as serial numbers are converted; the origin differs between the Windows and Mac 1904 workbook settings, so it is explicit.

Usage

```
map_obs(
  raw,
  date_col,
  value_cols,
  treatment_col = NULL,
  rep_col = NULL,
  sd_cols = NULL,
  units = NULL,
  var_map = NULL,
  excel_origin = c("1900", "1904"),
  sim_col = "sim",
  join_sep = "_",
  keep_id = TRUE
)
```

Arguments

`raw`	Data frame from ‘read_obs_raw()‘.
`date_col, treatment_col`	Column names.
`value_cols`	Columns holding observations.
`rep_col`	Column identifying the replicate (plot, block, rep). When given, replicates are averaged per date and treatment, and their spread is returned as ‘obs_sd‘ with the count as ‘obs_n‘.
`sd_cols`	Optional named vector mapping a value column to its SD column.
`units`	Optional named vector of units per value column.

var_map	Named vector renaming observed columns to the APSIM variable they correspond to, e.g. ‘c(Grain_yield = "wrr", Biomass = "wagt")‘. Spreadsheet headings almost never match APSIM variable names, and an unmapped name silently matches nothing downstream.
excel_origin	"1900" (Windows default) or "1904" (legacy Mac).
sim_col	Name of the simulation column.
join_sep	Separator used to join several treatment columns into the label matched against the simulation names.
keep_id	Keep the original treatment/site columns alongside the joined key, so they remain available for grouping and faceting.

Value

A long-format data frame of observed values with columns for the simulation key, ‘Date‘, ‘variable‘, ‘value‘, ‘obs_sd‘, ‘obs_n‘ and ‘unit‘.

Examples

```
raw <- data.frame(Date = c("2016-10-31", "2016-10-31"),
                  Treatment = "Gosaba_Rice_AWD_N120", Replication = 1:2,
                  Grain_yield = c(8600, 8900))
map_obs(raw, "Date", "Grain_yield", treatment_col = "Treatment",
        rep_col = "Replication", var_map = c(Grain_yield = "wrr"))
```

---

new_project *Project configuration*

Description

Everything that varies between experiments — the factor mapping, display labels, the treatment colour map, aggregation rules and the observed-file column mapping — lives in one small YAML file. Re-running APSIM then costs nothing: load the project, re-read the output directory, and every figure comes back with the same labels, colours and treatment order.

Usage

```
new_project(name = "untitled", out_dir = ".")
```

Arguments

name	Short project name.
out_dir	Directory holding the .out files.

Value

An object of class "apsim_project".

Examples

```
new_project("my_trial", tempdir())
```

`obs_sheets` *Sheet names in a workbook, for the GUI's sheet picker*

Description

Sheet names in a workbook, for the GUI's sheet picker

Usage

```
obs_sheets(file)
```

Arguments

`file`	File path.

Value

A character vector of worksheet names, or NA for non-Excel files.

Examples

```
f <- file.path(tempdir(), "obs.csv")
utils::write.csv(data.frame(a = 1), f, row.names = FALSE)
obs_sheets(f)
unlink(f)
```

`out_cache` *Incremental cache over a directory of .out files*

Description

Re-parses only what changed. APSIM output is regenerated on every run, so a full re-read of a large factorial directory on each refresh is the main thing standing between the user and a responsive GUI.

Usage

```
out_cache(path, pattern = "\\.out$", recursive = FALSE, hash = FALSE, ...)
```

Arguments

path	Directory of .out files.
pattern	File name filter.
recursive	See Details.
hash	Compute content hashes.
...	Passed to ‘read_out()‘ (e.g. ‘vars‘).

Value

An object of class "out_cache"; call ‘refresh()‘ on it.

Examples

```
d <- system.file("extdata", package = "apsimeval")
cc <- refresh(out_cache(d))
cc
```

---

pair_obs *Pair observed values to simulated output with a date tolerance*

Description

Pair observed values to simulated output with a date tolerance

Usage

```
pair_obs(
  sim,
  obs,
  tol_days = 3,
  by = "sim",
  date_col = "Date",
  var_col = "variable",
  value_col = "value"
)
```

Arguments

sim	Long-form simulated data (sim, Date, variable, value).
obs	Long-form observed data with the same key columns.
tol_days	Maximum absolute date gap accepted for a match.
by	Extra key columns present in both (e.g. treatment factors).
date_col	Name of the date column.
var_col	Name of the variable column.
value_col	Name of the value column.

Value

A data frame of matched observed/simulated pairs with columns ‘obs‘, ‘pred‘, ‘date_sim‘ and ‘gap_days‘, or NULL when nothing matched.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
obs <- data.frame(sim = "Gosaba_Rice_AWD_N120",
                  Date = as.Date(c("1999-10-31", "2003-10-31",
                                   "2008-10-31", "2013-10-31")),
                  variable = "wrr", value = c(8600, 7400, 9000, 7800),
                  obs_sd = c(300, 280, 350, 310), unit = "kg/ha")
long <- as_long(sim, vars = "wrr")
pr <- pair_obs(long, obs, tol_days = 3)
pr[, c("Date", "obs", "pred", "gap_days")]
```

---

panel_source *Which data source a panel type needs*

Description

"paired" types need observed data; "sim" types run on simulated output alone; "sens" types consume a sensitivity table. The GUI uses this to grey out panels that cannot be built yet.

Usage

```
panel_source(type)
```

Arguments

type	Plot type.

Value

A single string: "paired", "sim" or "sens".

Examples

```
panel_source("one2one")
```

panel_spec *Describe one panel of a multi-panel figure*

Description

A panel is a plot type plus its own settings. Collecting several specs and passing them to ‘build_figure()‘ is how a manuscript figure with panels A, B and C gets assembled — each panel keeps its own variable, grouping, faceting and options, while palette and theme stay shared.

Usage

```
panel_spec(
  type,
  var = NULL,
  group = NULL,
  facet = NULL,
  title = NULL,
  ylab = NULL,
  xlab = NULL,
  ...
)
```

Arguments

type	One of "one2one", "resid", "taylor", "ts", "exceedance", "box", "violin", "bar", "density", "heatmap", "anomaly", "tornado", "variance".
var	Variable plotted (ignored by the evaluation types, which use the paired obs/pred columns).
group, facet	Column names for colour and faceting.
title	Optional panel title.
ylab, xlab	Optional axis label overrides.
...	Type-specific options: ‘stats‘, ‘sd_band‘, ‘obs_sd‘ (one2one); ‘against‘ (resid); ‘ref‘, ‘orientation‘, ‘step‘ (exceedance); ‘error‘, ‘letters‘ (bar); ‘y‘, ‘baseline‘, ‘percent‘ (heatmap); ‘baseline‘ (anomaly); ‘measure‘ (tornado); ‘drop_residual‘, ‘stacked‘ (variance); ‘secondary‘, ‘layout‘, ‘colour_by‘, ‘free_y‘ (series/ts); ‘agg‘, ‘crop_only‘ (how daily output is collapsed before plotting).

Value

A ‘panel_spec‘ object.

Examples

```
panel_spec("exceedance", var = "wrr", group = "N", ref = 7000)
```

plot_anomaly *Anomaly of each year against the treatment mean*

Description

Shows which seasons drive a scenario difference, rather than only the aggregate shift.

Usage

```
plot_anomaly(
  data,
  value,
  x = "year",
  group = NULL,
  baseline = NULL,
  facet = NULL,
  ylab = NULL
)
```

Arguments

data	A data frame.
value	Column holding the value to plot.
x	Object to operate on.
group	Column mapped to colour or grouping.
baseline	See Details.
facet	Column to facet by.
ylab	Y axis label.

Value

A ggplot object.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
ag <- aggregate_sim(sim, by = c("sim", "year", "N"), vars = c(wrr = "max"))
plot_anomaly(ag, "wrr", group = "N")
```

plot_bar *Treatment means with error bars and significance letters*

Description

Treatment means with error bars and significance letters

Usage

```
plot_bar(
  data,
  value,
  group,
  facet = NULL,
  error = c("sd", "se", "ci95"),
  letters = FALSE,
  alpha = 0.05,
  ylab = NULL
)
```

Arguments

data	A data frame.
value	Column holding the value to plot.
group	Column mapped to colour or grouping.
facet	Column to facet by.
error	"sd", "se", or "ci95".
letters	Add compact letter display from a Tukey test.
alpha	Fill transparency.
ylab	Y axis label.

Value

A ggplot object.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
ag <- aggregate_sim(sim, by = c("sim", "year", "N"), vars = c(wrr = "max"))
plot_bar(ag, "wrr", "N", error = "se")
```

plot_box *Distribution of a simulated variable by treatment*

Description

Box plus jittered points, so reviewers can see the years behind the box.

Usage

```
plot_box(
  data,
  value,
  group,
  facet = NULL,
  violin = FALSE,
  points = TRUE,
  ylab = NULL,
  notch = FALSE
)
```

Arguments

data	A data frame.
value	Column holding the value.
group	Column mapped to colour or grouping.
facet	Column to facet by.
violin	Draw a violin.
points	Overlay individual points.
ylab	Y axis label.
notch	Notched boxes.

Value

A ggplot object.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
ag <- aggregate_sim(sim, by = c("sim", "year", "N"), vars = c(wrr = "max"))
plot_box(ag, "wrr", "N")
```

plot_density *Distribution density by treatment*

Description

Distribution density by treatment

Usage

```
plot_density(
  data,
  value,
  group = NULL,
  facet = NULL,
  alpha = 0.35,
  rug = TRUE,
  xlab = NULL
)
```

Arguments

data	A data frame.
value	Column holding the value.
group	Column mapped to colour or grouping.
facet	Column to facet by.
alpha	Significance level or transparency.
rug	Draw a rug of observations.
xlab	X axis label.

Value

A ggplot object.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
ag <- aggregate_sim(sim, by = c("sim", "year", "N"), vars = c(wrr = "max"))
plot_density(ag, "wrr", group = "N")
```

`plot_exceedance` *Probability of exceedance curve*

Description

The workhorse for long-term scenario comparison: P(X >= x) against the variable, one line per treatment or scenario.

Usage

```
plot_exceedance(
  data,
  value,
  group = NULL,
  facet = NULL,
  orientation = c("prob_x", "prob_y"),
  step = FALSE,
  ref = NULL,
  xlab = NULL,
  ylab = NULL,
  percent = TRUE,
  linewidth = 0.7
)
```

Arguments

`data`	Data frame of simulated values (one row per year/season).
`value`	Column to analyse.
`group`	Treatment/scenario column.
`facet`	Optional facet column.
`orientation`	"prob_x" puts probability on the x axis (common in agronomy); "prob_y" puts it on y (common in hydrology).
`step`	Draw as a step function rather than a smooth line.
`ref`	Optional horizontal/vertical reference value, e.g. a break-even yield.
`xlab`	X axis label.
`ylab`	Y axis label.
`percent`	Express as a percentage.
`linewidth`	Line width.

Value

A ggplot object.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
ag <- aggregate_sim(sim, by = c("sim", "year", "N"), vars = c(wrr = "max"))
plot_exceedance(ag, "wrr", group = "N")
```

`plot_heatmap` *Treatment x scenario heatmap*

Description

Mean response, or change against a baseline level, as a tile grid. The natural summary when a factorial has two large factors.

Usage

```
plot_heatmap(
  data,
  value,
  x,
  y,
  baseline = NULL,
  percent = TRUE,
  stat = mean,
  digits = 1,
  low = NULL,
  mid = NULL,
  high = NULL
)
```

Arguments

`data`	A data frame.
`value`	Column holding the value to plot.
`x`	Object to operate on.
`y`	See Details.
`baseline`	Level of ‘x‘ to express other levels relative to; NULL for absolute means.
`percent`	Show relative change as a percentage.
`stat`	Function collapsing replicates.
`digits`	Decimal places.
`low`	Low end colour.
`mid`	Midpoint colour.
`high`	High end colour.

Value

A ggplot object.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
ag <- aggregate_sim(sim, by = c("sim", "year", "N", "water"),
                    vars = c(wrr = "max"))
plot_heatmap(ag, "wrr", x = "N", y = "water")
```

---

plot_one2one *Observed vs simulated 1:1 plot*

Description

Draws the 1:1 line, an optional fitted line, and a statistics box. Axes are forced square and share one range so the 1:1 line is a true diagonal — the single most common mistake in published APSIM validation figures.

Usage

```
plot_one2one(
  data,
  obs = "obs",
  pred = "pred",
  group = NULL,
  facet = NULL,
  stats = c("n", "r2", "rmse", "sd_obs", "d"),
  fit = TRUE,
  xlab = NULL,
  ylab = NULL,
  pos = c("topleft", "bottomright"),
  min_n = 5,
  point_size = 2.2,
  obs_sd = NULL,
  pred_sd = NULL,
  sd_band = FALSE,
  digits = 2,
  free_scales = FALSE,
  stats_by = NULL
)
```

Arguments

data	Paired data (output of ‘pair_obs()‘ or any obs/pred frame).
obs, pred	Column names.
group	Column mapped to colour and shape (treatment).
facet	Optional faceting column, e.g. "variable".
stats	Metrics to print in the corner; NULL to omit.
fit	Add the least-squares line.
xlab, ylab	Axis labels. NULL derives them from the variable/unit.
pos	Corner for the stats box: "topleft" or "bottomright".
min_n	Minimum pairs for unstable metrics.
point_size	Point size.
obs_sd, pred_sd	Optional column names holding replicate standard deviations, drawn as error bars on the observed (horizontal) and simulated (vertical) axes.
sd_band	Shade a band of +/- SD of the observed values around the 1:1 line. RMSE smaller than SD_obs means the model beats the observed mean as a predictor; the band makes that comparison visual.
digits	Decimal places.
free_scales	Let faceted panels use their own axis ranges. Needed when faceting by variable (different units); panels are then not square.
stats_by	Columns the corner statistics are grouped by. Defaults to ‘facet‘, so each panel reports its own; pass ‘character(0)‘ for one pooled set even when the plot is faceted.

Value

A ggplot object.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
obs <- data.frame(sim = "Gosaba_Rice_AWD_N120",
                  Date = as.Date(c("1999-10-31", "2003-10-31",
                                   "2008-10-31", "2013-10-31")),
                  variable = "wrr", value = c(8600, 7400, 9000, 7800),
                  obs_sd = c(300, 280, 350, 310), unit = "kg/ha")
long <- as_long(sim, vars = "wrr")
pr <- pair_obs(long, obs, tol_days = 3)
plot_one2one(pr)
```

plot_resid *Residual plot*

Description

Residual against simulated (or against date, to expose seasonal drift). A 1:1 plot hides sign structure; this does not.

Usage

```
plot_resid(
  data,
  obs = "obs",
  pred = "pred",
  group = NULL,
  facet = NULL,
  against = c("pred", "obs", "date"),
  date_col = "Date",
  smooth = TRUE,
  ylab = NULL
)
```

Arguments

data	A data frame.
obs	Observed column name.
pred	Simulated column name.
group	Column mapped to colour or grouping.
facet	Column to facet by.
against	"pred", "obs", or "date".
date_col	Name of the date column.
smooth	Add a loess trend through the residuals.
ylab	Y axis label.

Value

A ggplot object.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
obs <- data.frame(sim = "Gosaba_Rice_AWD_N120",
```

```
                  Date = as.Date(c("1999-10-31", "2003-10-31",
                                   "2008-10-31", "2013-10-31")),
                  variable = "wrr", value = c(8600, 7400, 9000, 7800),
                  obs_sd = c(300, 280, 350, 310), unit = "kg/ha")
long <- as_long(sim, vars = "wrr")
pr <- pair_obs(long, obs, tol_days = 3)
plot_resid(pr)
```

`plot_series` *Multi-variable, multi-treatment time series*

Description

The workhorse view for judging season-by-season behaviour. Extends ‘plot_ts()‘ in two directions that matter for model evaluation:

Usage

```
plot_series(
  sim,
  vars,
  secondary = NULL,
  obs = NULL,
  facet_by = NULL,
  layout = c("overlay", "stacked"),
  colour_by = c("variable", "treatment"),
  ncol = NULL,
  free_y = FALSE,
  cutoff = NULL,
  phase_labels = c("Calibration", "Validation"),
  shade = TRUE,
  obs_sd = "obs_sd",
  date_col = "Date",
  value_col = "value",
  var_col = "variable",
  point_size = 2.1,
  linewidth = 0.6,
  errorbar_width = 6,
  sec_label = NULL
)
```

Arguments

`sim`	Simulated output, wide (from ‘read_out_dir()‘) or already long.
`vars`	Variables to draw. Order sets the legend order.
`secondary`	Variables drawn against the right-hand axis. Ignored when ‘layout = "stacked"‘.

obs	Observed data in long form, with a ‘variable‘ column. Only rows whose variable is in ‘vars‘ are drawn.
facet_by	Column panelling the plot - typically "sim" for every treatment, or a factor column such as "water".
layout	"overlay" puts all variables in one panel (using ‘secondary‘ where given); "stacked" gives each variable its own row.
colour_by	"variable" (default) or "treatment". Colouring by treatment only makes sense with a single variable.
ncol	Columns in the treatment grid; NULL lets ggplot2 choose.
free_y	Let each treatment panel scale independently. Off by default: treatments are being compared, so a shared scale is usually wanted.
cutoff	Calibration/validation cutoff date, or NULL.
phase_labels	Labels for the two phases.
shade	Shade the calibration and validation spans.
obs_sd	Column of ‘obs‘ holding replicate standard deviations.
date_col, value_col, var_col	Column names in the long data.
point_size	Observed point size.
linewidth	Line width.
errorbar_width	Error bar cap width in days.
sec_label	Optional label for the right-hand axis; defaults to the secondary variables and their units.

Details

Several variables at once Yield, biomass and a stress index have incompatible ranges, so variables listed in ‘secondary‘ are drawn against a right-hand axis. The rescaling is linear and the right axis is labelled in the original units, so nothing has to be read off a transformed scale.

All treatments side by side ‘facet_by‘ panels the same variables across every treatment, which is how a treatment effect is normally judged - the shapes are compared, not just the endpoints.

A dual axis is a real hazard: two series with different units can be made to look correlated by the choice of scaling alone. ‘layout = "stacked"‘ avoids it by giving each variable its own panel and free y scale, and is the safer default when the variables are not meant to be compared point by point. Both layouts share the x axis, so seasons still line up.

Value

A ggplot object.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
long <- as_long(sim, vars = c("wagt", "wrr"), keep = "N")
plot_series(long, vars = c("wagt", "wrr"), facet_by = "N")
```

`plot_taylor` *Taylor diagram*

Description

Puts correlation, normalised standard deviation and centred RMS difference on one polar plot, so several treatments or model versions are compared at a glance. Distance from the reference point is the centred RMS error.

Usage

```
plot_taylor(
  data,
  obs = "obs",
  pred = "pred",
  group = "sim",
  ref_label = "Observed"
)
```

Arguments

`data`	A data frame.
`obs`	See Details.
`pred`	See Details.
`group`	Column mapped to colour or grouping.
`ref_label`	See Details.

Value

A ggplot object.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
obs <- data.frame(sim = "Gosaba_Rice_AWD_N120",
                  Date = as.Date(c("1999-10-31", "2003-10-31",
                                   "2008-10-31", "2013-10-31")),
                  variable = "wrr", value = c(8600, 7400, 9000, 7800),
                  obs_sd = c(300, 280, 350, 310), unit = "kg/ha")
long <- as_long(sim, vars = "wrr")
pr <- pair_obs(long, obs, tol_days = 3)
plot_taylor(pr, group = "sim")
```

plot_tornado *Tornado plot of one-at-a-time or range sensitivity*

Description

Tornado plot of one-at-a-time or range sensitivity

Usage

```
plot_tornado(x, measure = NULL, xlab = NULL)
```

Arguments

x	Output of ‘sens_oat()‘ or ‘sens_range()‘.
measure	Column to plot: "pct", "delta", "elasticity" (OAT) or "range", "pct_range" (range screening).
xlab	X axis label.

Value

A ggplot object.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
ag <- aggregate_sim(sim, by = c("sim", "year", "N"), vars = c(wrr = "max"))
plot_tornado(sens_oat(ag, "wrr", "N"))
```

plot_ts *Simulated time series with observed points overlaid*

Description

When a calibration/validation cutoff is supplied the simulated line stays continuous across it. A single APSIM run is one continuous series, so faceting it by phase would imply two separate simulations; the phases are distinguished by a background band and a marker at the cutoff instead, and observed points are shaped by the phase they fall in.

Usage

```
plot_ts(
  sim,
  obs = NULL,
  value = "value",
  date_col = "Date",
  group = NULL,
  facet = NULL,
  ylab = NULL,
  linewidth = 0.6,
  point_size = 2.2,
  cutoff = NULL,
  phase_col = NULL,
  shade = TRUE,
  obs_sd = "obs_sd",
  errorbar_width = 6,
  phase_labels = c("Calibration", "Validation")
)
```

Arguments

sim	Simulated output in long form.
obs	Observed data to overlay, or NULL.
value	Column holding the value.
date_col	Name of the date column.
group	Column mapped to colour or grouping.
facet	Column to facet by.
ylab	Y axis label.
linewidth	Line width.
point_size	Observed point size.
cutoff	Date separating calibration from validation; NULL for none.
phase_col	Column in ‘obs‘ already holding the phase, for the case where calibration and validation come from separate simulations rather than a date split.
shade	Shade the calibration and validation spans.
obs_sd	Column of ‘obs‘ holding the replicate standard deviation, drawn as vertical error bars. NULL to omit. Points with a single replicate have no SD and get no bar.
errorbar_width	Cap width in days.
phase_labels	Labels for the two phases.

Value

A ggplot object.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
obs <- data.frame(sim = "Gosaba_Rice_AWD_N120",
                  Date = as.Date(c("1999-10-31", "2003-10-31",
                                   "2008-10-31", "2013-10-31")),
                  variable = "wrr", value = c(8600, 7400, 9000, 7800),
                  obs_sd = c(300, 280, 350, 310), unit = "kg/ha")
long <- as_long(sim, vars = "wrr")
pr <- pair_obs(long, obs, tol_days = 3)
plot_ts(long, obs = obs)
```

---

plot_variance *Variance decomposition from sens_anova()*

Description

Stacked share of total variance, with interaction and season terms kept visually distinct from main effects.

Usage

```
plot_variance(x, drop_residual = FALSE, min_si = 1, stacked = FALSE)
```

Arguments

x	Object to operate on.
drop_residual	Hide the residual term.
min_si	Collapse terms below this percentage into "other".
stacked	Draw stacked.

Value

A ggplot object.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
ag <- aggregate_sim(sim, by = c("sim", "year", "N"), vars = c(wrr = "max"))
plot_variance(sens_anova(ag, "wrr", "N", order = 1))
```

preview_factors *Preview a factor mapping before applying it*

Description

Shows the first few names beside the columns they would produce, so a bad regex is obvious immediately rather than after a failed plot.

Usage

```
preview_factors(sims, n = 10, col = "sim", ...)
```

Arguments

sims	Simulation names, or a data frame plus ‘col‘.
n	Number of items.
col	Column name.
...	Further arguments.

Value

A data frame of the first ‘n‘ simulation names beside the factor columns the mapping would produce.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
preview_factors(sim$sim, parts = c("site", "crop", "water", "N"))
```

read_obs_raw *Read observed data from Excel or CSV*

Description

Returns the raw sheet plus a guess at which columns are the date, the treatment and the observations, for the interface's mapping menus to pre-populate. Nothing is pivoted until the user confirms the mapping.

Usage

```
read_obs_raw(file, sheet = 1)
```

Arguments

file	File path.
sheet	Worksheet name or index.

Value

A data frame of the sheet as read, carrying a ‘guess‘ attribute naming the likely date, treatment, replicate and value columns.

Examples

```
f <- file.path(tempdir(), "obs.csv")
utils::write.csv(data.frame(Date = "2016-10-31", Treatment = "T1",
                            Replication = 1, Grain_yield = 8000), f,
                 row.names = FALSE)
attr(read_obs_raw(f), "guess")
unlink(f)
```

`read_out` *Read an APSIM Classic .out file*

Description

Parses the preamble (key = value lines), the variable-name row and the units row, then the data block. Missing values ("?", "*", "NA", "") become NA. Units are kept as a per-column attribute and as an "units" attribute on the returned data frame.

Usage

```
read_out(file, vars = NULL, sim = NULL, date_col = "Date")
```

Arguments

`file`	Path to a .out file.
`vars`	Optional character vector of columns to keep (Date/year are always kept if present). Subsetting happens after parsing but before type conversion, so it is cheap on wide daily files.
`sim`	Simulation label. Defaults to the Title = line; if that is missing or duplicated across files, use the file stem.
`date_col`	Name of the date column. Default "Date".

Value

A data.frame with attributes: units, apsim_version, title, file, freq.

Examples

```
f <- system.file("extdata", "Gosaba_Rice_AWD_N120.out", package = "apsimeval")
d <- read_out(f)
head(d)
attr(d, "units")
```

`read_out_dir` *Read every .out file in a directory*

Description

Read every .out file in a directory

Usage

```
read_out_dir(
  path,
  pattern = "\\.out$",
  recursive = FALSE,
  sim_from = c("auto", "title", "file"),
  ...
)
```

Arguments

`path`	Directory to scan.
`pattern`	Regular expression selecting files.
`recursive`	Recurse into subdirectories.
`sim_from`	"auto" uses the Title line, falling back to the file stem when Titles are duplicated across files (common with factorial runs).
`...`	Passed to read_out().

Value

A data frame combining every parsed file, with ‘units‘ and ‘freq‘ attributes.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
table(sim$sim)
```

refresh *Re-scan and re-parse only changed files*

Description

Re-scan and re-parse only changed files

Usage

```
refresh(cache, force = FALSE)
```

Arguments

cache	An ‘out_cache‘.
force	Re-parse everything regardless of the manifest.

Value

The cache, invisibly. Use ‘cache_data()‘ for the combined frame.

Examples

```
d <- system.file("extdata", package = "apsimeval")
cc <- out_cache(d)
refresh(cc)
nrow(cache_data(cc))
```

render_panel *Render one panel from its spec*

Description

Render one panel from its spec

Usage

```
render_panel(spec, sim = NULL, paired = NULL, sens = NULL)
```

Arguments

spec	A ‘panel_spec‘.
sim	Simulated output (wide, from ‘read_out_dir()‘).
paired	Paired obs/pred data, for evaluation panels.
sens	A sensitivity table, for sensitivity panels.

Value

A ggplot object.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
render_panel(panel_spec("box", var = "wrr", group = "N"), sim = sim)
```

---

run_app *Launch the apsimeval GUI*

Description

A Shiny front end over the package functions. Nothing here computes anything the engine cannot do headlessly — the app only handles file watching, mapping and figure export.

Usage

```
run_app(
  out_dir = getwd(),
  project = NULL,
  poll_ms = 3000,
  launch.browser = TRUE
)
```

Arguments

out_dir	Directory of .out files to open on start.
project	Optional .apsimproj file to load on start.
poll_ms	How often to check the output directory, in milliseconds.
launch.browser	Open in the system browser.

Value

Called for its side effect of starting the application; returns a 'shiny' app object.

Examples

```
if (interactive()) {
  run_app(out_dir = tempdir())
}
```

save_project *Write a project to disk*

Description

Write a project to disk

Usage

```
save_project(project, file)
```

Arguments

project	An ‘apsim_project‘.
file	Destination path. Required - no default is used, so the function never writes to a location the caller did not name. Written as ’YAML’ so the file stays readable, easy to compare between versions, and editable by hand; a lookup-table factor map is written alongside as CSV rather than inlined.

Value

The path written, invisibly.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
p <- new_project("demo", d)
p <- set_project_factors(p, sim)
f <- file.path(tempdir(), "demo.apsimproj")
save_project(p, f)
load_project(f)
unlink(f)
```

save_pub *Save at journal column widths and print resolution*

Description

Save at journal column widths and print resolution

Usage

```
save_pub(
  plot,
  file,
  width = "single",
  height = 80,
  dpi = 600,
  format = "tiff",
  units = "mm",
  compression = "lzw"
)
```

Arguments

plot	A ggplot object.
file	File path.
width	"single" (89 mm), "onehalf" (140 mm), "double" (190 mm), or a numeric width in the given units.
height	Figure height.
dpi	600 for line art and combined art, 300 acceptable for photos.
format	One or more of "tiff", "png", "pdf", "eps", "jpeg".
units	Units per variable.
compression	TIFF compression.

Value

The plot, invisibly.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
ag <- aggregate_sim(sim, by = c("sim", "year", "N"), vars = c(wrr = "max"))
p <- plot_box(ag, "wrr", "N")
f <- file.path(tempdir(), "figure")
save_pub(p, f, format = "png", dpi = 72)
unlink(paste0(f, ".png"))
```

scan_out *Manifest of .out files in a directory*

Description

Records path, size and modification time. Comparing two manifests is how the GUI decides what to re-parse after an APSIM re-run — hashing every file on every poll would be wasteful, and mtime plus size catches essentially every real edit. Content hashing is available via ‘hash = TRUE‘ for the pathological case of a re-run that produces byte-identical timestamps.

Usage

```
scan_out(path, pattern = "\\.out$", recursive = FALSE, hash = FALSE)
```

Arguments

path	Directory to scan.
pattern	File filter.
recursive	Recurse into subdirectories.
hash	Also compute an MD5 of each file (slower, more certain).

Value

A data frame with columns ‘path‘, ‘size‘, ‘mtime‘ and ‘md5‘.

Examples

```
d <- system.file("extdata", package = "apsimeval")
scan_out(d)[, c("size", "mtime")]
```

sens_anova *Variance-based sensitivity from a factorial simulation set*

Description

APSIM factorial runs are a complete design, so the variance of the output can be decomposed directly by ANOVA. The share of the total sum of squares attributable to each factor is a first-order sensitivity index, and the interaction terms show where factors are not additive — the thing a one-at-a-time analysis cannot see.

Usage

```
sens_anova(
  data,
  response,
  factors,
  order = 2,
  year_as_rep = TRUE,
  year_col = "year"
)
```

Arguments

data	Aggregated simulation output (one row per run/season).
response	Output variable name.
factors	Factor columns to decompose.
order	Highest interaction order to include (1 = main effects only).
year_as_rep	Include ‘year‘ as a blocking term so between-season variation is separated from treatment effects rather than lumped into the residual.
year_col	Name of the year column.

Value

A data frame of terms with sums of squares, sensitivity index (share of explained variance), df and p value.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
ag <- aggregate_sim(sim, by = c("sim", "year", "N"), vars = c(wrr = "max"))
sens_anova(ag, "wrr", "N", order = 1)
```

---

sens_oat *One-at-a-time sensitivity around a baseline run*

Description

Reports the change in output when each factor moves away from its baseline level, in absolute terms, as a percentage, and as an elasticity (percent output change per percent input change) where the factor is numeric. Elasticity is the comparable quantity across factors with different units.

Usage

```
sens_oat(data, response, factors, baseline = NULL, stat = mean)
```

Arguments

data	Aggregated output.
response	Output variable.
factors	Factor columns.
baseline	Named list giving the baseline level of each factor. If NULL, the most frequent level of each is used.
stat	Function collapsing replicate years to one value per run.

Value

A data frame with one row per off-baseline level, giving the change in the response, its percentage and, for numeric factors, the elasticity.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
ag <- aggregate_sim(sim, by = c("sim", "year", "N"), vars = c(wrr = "max"))
sens_oat(ag, "wrr", "N")
```

---

sens_range *Range of output spanned by each factor*

Description

The simplest screening measure and the one most readable in a tornado plot: how far the response moves between a factor's lowest and highest level, averaged over everything else.

Usage

```
sens_range(data, response, factors, stat = mean)
```

Arguments

data	A data frame.
response	Response variable.
factors	Factor columns.
stat	Function collapsing replicates.

Value

A data frame with one row per factor giving the low and high levels and the range they span.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
ag <- aggregate_sim(sim, by = c("sim", "year", "N"), vars = c(wrr = "max"))
sens_range(ag, "wrr", "N")
```

---

set_colour_mode *Colour mode for every figure*

Description

Some journals charge for colour or accept only greyscale or line art. Setting the mode once changes every subsequent figure without touching the plotting calls.

Usage

```
set_colour_mode(mode = c("colour", "grey", "mono"))

colour_mode()
```

Arguments

mode	One of "colour", "grey", "mono".

Details

colour Okabe-Ito palette, or whatever ‘set_palette()‘ registered.

grey A light-to-dark grey ramp. Shapes and line types still vary, so groups stay separable where the greys are close.

mono Line art: every line and outline black. Areas that must be told apart by fill alone (bars, boxes, tiles) keep a grey ramp, since pure black-and-white fills cannot encode more than two levels.

Value

The mode, invisibly.

Examples

```
old <- colour_mode()
set_colour_mode("grey")
get_pal(c("N0", "N120"))
set_colour_mode(old)
```

`set_factors` *Split simulation names into treatment factor columns*

Description

APSIM encodes treatments in the simulation name ("Gosaba_Rice_AWD_N120"). This turns that string into real columns so every plot and statistic can be grouped, faceted and coloured by treatment. Three strategies, in increasing order of effort:

Usage

```
set_factors(
  data,
  parts = NULL,
  sep = "_",
  regex = NULL,
  map = NULL,
  col = "sim",
  strip = NULL,
  numeric_from = NULL,
  warn_unmatched = TRUE
)
```

Arguments

`data`	A data frame with a simulation-name column.
`parts`	Character vector naming the fields in order. Use NA to skip a position.
`sep`	Regular expression separating fields. Default "_".
`regex`	Alternative to parts/sep: a regex with (?<name>...) groups.
`map`	Alternative lookup table; must contain ‘col‘ plus factor columns.
`col`	Name of the simulation column. Default "sim".
`strip`	Optional regex removed from each name before splitting, e.g. a common prefix or a trailing run number.
`numeric_from`	Fields to coerce to numbers after stripping non-digits, e.g. "N" turning "N120" into 120. Keeps axis ordering sensible.
`warn_unmatched`	Warn when a simulation name yields no factors.

Details

delimiter ‘parts‘ names the fields in order, ‘sep‘ splits them.

regex ‘regex‘ with named capture groups, for irregular names.

lookup ‘map‘, a data frame keyed on the simulation column, for names with no usable structure at all.

Parsing never modifies existing data — it only adds columns — so a wrong mapping is fixed by re-running, not by re-parsing the .out files.

Value

‘data‘ with factor columns added, and a "factor_spec" attribute recording how they were derived.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
```

---

set_palette *Fix treatment colours, display labels and units for the whole session*

Description

Fix treatment colours, display labels and units for the whole session

Usage

```
set_palette(map)

set_labels(map)
```

Arguments

map	Named vector or list mapping levels to colours, or variables to display labels.

Value

The map, invisibly.

Examples

```
set_palette(c(N0 = "#D55E00", N120 = "#0072B2"))
set_labels(list(wrr = "Grain yield"))
```

set_project_factors *Record the current mapping into a project*

Description

Record the current mapping into a project

Usage

```
set_project_factors(project, data)
```

Arguments

project	An ‘apsim_project‘.
data	Data carrying a ‘factor_spec‘ attribute.

Value

The project with its ‘factors‘ slot updated.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
set_project_factors(new_project("demo", d), sim)
```

specs_to_list *Turn panel specs into plain lists for the project file, and back*

Description

Turn panel specs into plain lists for the project file, and back

Usage

```
specs_to_list(specs)

specs_from_list(x)
```

Arguments

specs	List of ‘panel_spec‘ objects.
x	Object to operate on.

Value

A list of plain lists (‘specs_to_list‘) or of ‘panel_spec‘ objects (‘specs_from_list‘).

Examples

```
s <- list(panel_spec("box", var = "wrr", group = "N"))
identical(specs_from_list(specs_to_list(s))[[1]]$type, "box")
```

---

split_phase *Split paired data into calibration and validation subsets*

Description

Crop-model papers report the two phases separately. The split is usually by season or year (early years calibrate, later years validate), sometimes by treatment (one N rate held back), sometimes recorded as a column in the observed file.

Usage

```
split_phase(
  data,
  method = c("date", "treatment", "column", "fraction"),
  at = NULL,
  col = "sim",
  validation = NULL,
  date_col = "Date"
)
```

Arguments

data	Paired obs/pred data.
method	"date" splits at ‘at‘; "treatment" assigns the levels in ‘validation‘ to validation; "column" reads an existing column; "fraction" takes the first ‘at‘ proportion of rows in date order.
at	Split point: a date, or a proportion for method "fraction".
col	Column holding the treatment or the pre-existing phase label.
validation	Levels assigned to validation for method "treatment".
date_col	Name of the date column.

Value

‘data‘ with a ‘phase‘ factor added.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
obs <- data.frame(sim = "Gosaba_Rice_AWD_N120",
                  Date = as.Date(c("1999-10-31", "2003-10-31",
                                   "2008-10-31", "2013-10-31")),
                  variable = "wrr", value = c(8600, 7400, 9000, 7800),
                  obs_sd = c(300, 280, 350, 310), unit = "kg/ha")
long <- as_long(sim, vars = "wrr")
pr <- pair_obs(long, obs, tol_days = 3)
table(split_phase(pr, "date", at = "2005-01-01")$phase)
```

---

suggest_cutoff *Suggest a calibration/validation cutoff date*

Description

The usual APSIM convention is to calibrate on the first season and validate on the rest. Returns a date midway between the last observation of the calibration seasons and the first of the validation seasons, so the boundary does not sit exactly on a measurement.

Usage

```
suggest_cutoff(x, date_col = "Date", seasons = 1)
```

Arguments

x	Data with a date column.
date_col	Name of the date column.
seasons	Number of leading seasons assigned to calibration.

Value

A single Date, or NULL when the input has no dates.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
obs <- data.frame(sim = "Gosaba_Rice_AWD_N120",
                  Date = as.Date(c("1999-10-31", "2003-10-31",
                                   "2008-10-31", "2013-10-31")),
                  variable = "wrr", value = c(8600, 7400, 9000, 7800),
                  obs_sd = c(300, 280, 350, 310), unit = "kg/ha")
```

```
long <- as_long(sim, vars = "wrr")
pr <- pair_obs(long, obs, tol_days = 3)
suggest_cutoff(pr)
```

---

theme_apsim *Publication theme for APSIM figures*

Description

Serif-free, black-boxed panels, ticks pointing in — the convention most agronomy and field-crops journals expect. Sizes are set for figures that will be printed at one or two column widths, not for on-screen viewing.

Usage

```
theme_apsim(base_size = 11, base_family = "", grid = FALSE, legend = "top")
```

Arguments

base_size	Base font size in points (10-12 suits most journals).
base_family	Font family. "" uses the device default.
grid	Draw faint major grid lines.
legend	Legend position: "top", "right", "bottom", "none", or c(x, y).

Value

A ggplot2 theme object.

Examples

```
d <- system.file("extdata", package = "apsimeval")
sim <- read_out_dir(d)
sim <- set_factors(sim, parts = c("site", "crop", "water", "N"),
                   numeric_from = "N")
ag <- aggregate_sim(sim, by = c("sim", "year", "N"), vars = c(wrr = "max"))
plot_box(ag, "wrr", "N") + theme_apsim(base_size = 9)
```

Index