Package: earthUI (via r-universe)

June 2, 2026

Title Interactive 'shiny' GUI for the 'earth' Package

Version 0.8.0

Description Provides a 'shiny'-based graphical user interface for the 'earth' package, enabling interactive building and exploration of Multivariate Adaptive Regression Splines (MARS) models. Features include data import from CSV and 'Excel' files, automatic detection of categorical variables, interactive control of interaction terms via an allowed matrix, comprehensive model diagnostics with variable importance and partial dependence plots, and publication-quality report generation via 'Quarto'.

License AGPL (>= 3)

URL https://github.com/wcraytor/earthUI

BugReports https://github.com/wcraytor/earthUI/issues

Depends R (>= 4.1.0)

Imports DBI, earth (>= 5.3.0), ggplot2, jsonlite, openxlsx, plotly, readxl, RSQLite, shiny, stats, tools, utils

Suggests bslib, callr, DT, knitr, quarto, rmarkdown, shinyFiles, showtext, sysfonts, testthat (>= 3.0.0), tinytex, withr, writexl

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

RoxygenNote 7.3.3

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libssl-dev zlib1g-dev

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

Date/Publication 2026-06-02 07:15:03 UTC

RemoteUrl https://github.com/wcraytor/earthui

RemoteRef HEAD

RemoteSha 73048f440a2658cc9084ebab0754afe5a8fd2727

RemoteSubdir pkg

Contents

`auto_export_for_mgcv` *Export an earthUI result as RDS for consumption by mgcvUI*

Description

Saves the result via `saveRDS()` to `<file_name>_earthUI_result_<timestamp>.rds` and verifies the file is readable and can produce a prediction. If verification fails, the file is deleted. Skipped silently for models with `degree > 2` (mgcvUI only supports pairwise interactions).

Usage

```
auto_export_for_mgcv(result, output_folder, file_name)
```

Arguments

`result`	A fit result list as returned by `fit_earth()` (class `earthUI_result`).
`output_folder`	Character scalar. May be `NULL` or empty (defaults to `~/Downloads`).
`file_name`	Character scalar. Used to derive the output filename.

Value

Invisibly, `NULL`.

build_allowed_function *Build an allowed function for earth()*

Description

Converts an allowed interaction matrix into a function compatible with the `allowed` parameter of `earth::earth()`. The function checks that ALL pairwise combinations among the predictors in a proposed interaction term are TRUE in the matrix.

Usage

```
build_allowed_function(allowed_matrix, block_degree1 = NULL)
```

Arguments

`allowed_matrix`	A symmetric logical matrix as returned by `build_allowed_matrix()`.
`block_degree1`	Optional character vector of predictor names to block from entering the model as degree-1 (main effect) terms. These variables can still participate in interactions (degree >= 2). This is useful when a variable like `sale_age` should only enter through an interaction (e.g. with `living_area`) to capture size-varying time adjustments.

Details

The returned function implements the standard `earth()` allowed function contract. When earth proposes a new hinge function involving predictor `pred` with existing parent predictors indicated by the `parents` logical vector, the function checks that every pair of involved predictors is allowed in the matrix.

For a 3-way interaction between X, Y, Z, the function verifies that (X,Y), (Y,Z), and (X,Z) are all TRUE in the matrix.

When `block_degree1` is specified, any predictor in that list is blocked from entering as a degree-1 term but is allowed in higher-degree interactions (subject to the allowed matrix).

Value

A function with signature `function(degree, pred, parents, namesx, first)` suitable for the `allowed` parameter of `earth::earth()`.

Examples

```
mat <- build_allowed_matrix(c("sqft", "bedrooms", "pool"))
mat["sqft", "pool"] <- FALSE
mat["pool", "sqft"] <- FALSE
func <- build_allowed_function(mat)

# Block sale_age from degree 1 (interaction only)
```

```
mat2 <- build_allowed_matrix(c("sale_age", "living_area", "lot_size"))
func2 <- build_allowed_function(mat2, block_degree1 = "sale_age")
```

`build_allowed_matrix` *Build an allowed interaction matrix*

Description

Creates a symmetric logical matrix indicating which pairs of predictors are allowed to interact. By default, all interactions are allowed.

Usage

```
build_allowed_matrix(variable_names, default = TRUE)
```

Arguments

`variable_names`	Character vector of predictor variable names.
`default`	Logical. Default value for all entries. Default is `TRUE` (all interactions allowed).

Value

A symmetric logical matrix with `variable_names` as both row and column names.

Examples

```
mat <- build_allowed_matrix(c("sqft", "bedrooms", "pool"))
mat["sqft", "pool"] <- FALSE
mat["pool", "sqft"] <- FALSE
mat
```

`build_sales_grid` *Build a Sales Comparison Grid Excel workbook*

Description

Generates a multi-sheet xlsx workbook formatted as a Sales Comparison Grid from an RCA-adjusted data frame. Each sheet shows the subject and up to three comparable sales side by side, with factual values, value contributions, and adjustments per regression variable, plus rows for grouped variables (location, site, age), residual feature inputs, and an Adjusted Sale Price formula.

Usage

```
build_sales_grid(
  rca_df,
  comp_rows,
  output_file,
  specials = list(),
  title_prefix = "Intermediate Sales Comparable Grid",
  progress_fn = NULL
)
```

Arguments

rca_df	A data frame produced by the RCA workflow. Row 1 is the subject; rows 2+ are comps. Must contain columns produced by compute_rca_adjustments() (basis, residual, cqa, <var>_contribution, <var>_adjustment, net_adjustments, gross_adjustments, subject_cqa).
comp_rows	Integer vector of row numbers (>= 2) to include in the grid. Maximum 30 (10 sheets, 3 comps per sheet).
output_file	Character scalar. Destination xlsx path.
specials	Named list mapping a special type (e.g. "contract_date", "dom", "latitude", "longitude", "area", "concessions", "lot_size", "site_dimensions", "actual_age", "effective_age", "living_area", "sale_age") to the corresponding column name in rca_df. Default list().
title_prefix	Character scalar. Sheet-title prefix. Defaults to "Intermediate Sales Comparable Grid".
progress_fn	Optional function called after each sheet is written with arguments sheet, total_sheets, comps_done, total_comps. Used by the Shiny app to drive withProgress(). Default NULL.

Details

This is the non-Shiny computation kernel used by the earthUI Shiny app's Sales Grid download button, and is also suitable for use from batch scripts that already have rca_df in memory.

Value

Invisibly, the output_file path.

---

city_abbreviation *Generate a city path code from a full name*

Description

Applies the rule: lowercase + strip all non-alphanumerics + first 6 characters. If the resulting code already exists in existing_codes, append _1, _2, ... until unique.

Usage

```
city_abbreviation(full_name, existing_codes = character(0))
```

Arguments

full_name	Character scalar (e.g. "Carmel-by-the-Sea").
existing_codes	Character vector of codes already in the parent folder. Defaults to character(0).

Value

Character scalar.

---

compute_intermediate_output

Compute intermediate output data frame

Description

Returns an enhanced copy of data with per-target model columns appended: est_<target>, residual, cqa, optional residual_sf / cqa_sf (when a living-area column is supplied), per-g-function <var>_contribution columns, basis, and calc_residual. For appraisal or market purposes, rows are sorted by residual_sf (or residual) descending, with the subject row (row 1) pinned on top when applicable. Ranking columns (residual_sf, cqa_sf, residual, cqa) are moved to the leftmost positions.

Usage

```
compute_intermediate_output(
  data,
  result = NULL,
  purpose = c("general", "appraisal", "market"),
  skip_subject_row = FALSE,
  living_area_col = NULL
)
```

Arguments

data	A data frame (the raw imported data). Must contain the target column(s) named in result$target.
result	A fit result list as returned by fit_earth(), or NULL. When NULL, data is returned unchanged (the Shiny app's "non-adjusted" export path).
purpose	Character scalar: "general", "appraisal", or "market". Controls subject-row handling and sort behavior. Default "general".
skip_subject_row	Logical. In "market" mode, indicates whether row 1 should be treated as a subject (pinned on top during sort). Ignored for other purposes. Default FALSE.

`living_area_col`
: Character scalar or `NULL`. If supplied and present in `data`, per-SF residual and CQA_SF columns are added. Default `NULL`.

Details

This is the non-Shiny computation kernel used by the earthUI Shiny app's Download Intermediate Output button, and is also suitable for use from batch scripts.

Value

A data frame.

`compute_rca_adjustments`
: *Compute RCA (Residual Comparable Adjustment) output for an appraisal*

Description

Starting from the raw data (subject in row 1, comps in rows 2+) and a fitted earth model, produces the adjusted comparables data frame used by the Sales Comparison Grid. The user-supplied subject CQA score is converted into a subject residual via linear interpolation of the comp CQA/residual curve; per-g-function contributions and adjustments are then added for each comp, along with net/gross adjustments, percentages, and the final `adjusted_sale_price`.

Usage

```
compute_rca_adjustments(
  data,
  result,
  user_cqa,
  cqa_type = c("cqa", "cqa_sf"),
  living_area_col = NULL,
  weight_col = NULL
)
```

Arguments

`data`	A data frame (subject + comps) matching `result$data` in schema. Row 1 is the subject.
`result`	A fit result list as returned by `fit_earth()`.
`user_cqa`	Numeric scalar in `[0, 9.99]` — the subject CQA score.
`cqa_type`	Character: `"cqa"` (default) uses total-residual CQA; `"cqa_sf"` uses per-SF CQA (requires `living_area_col`).

`living_area_col`
: Character scalar or `NULL`. If supplied and present in `data`, `residual_sf` and `cqa_sf` columns are added and `cqa_type = "cqa_sf"` is supported.

`weight_col`
: Character scalar or `NULL`. If supplied and present in `data`, rows with `weight == 0` are treated as additional "subject-like" rows that receive imputed values instead of real comp values.

Details

For multi-target models, the primary target drives the subject residual calculation. Additional targets receive their own residual interpolation and adjustment columns, imputed only for zero-weight rows.

This is the non-Shiny computation kernel used by the earthUI Shiny app's RCA Raw Output button, and is also suitable for use from batch scripts.

Value

An enhanced data frame with model columns, contributions, adjustments, `subject_value`, `subject_cqa`, and `adjusted_sale_price`.

`compute_sale_age` *Compute sale age in integer days*

Description

Given a vector of contract (sale) dates and a single effective date, returns the difference in integer days (`effective_date - contract_date`). Contract dates may be supplied as `POSIXct`, `Date`, character strings parseable by `as.POSIXct()`, or numeric Excel serial date numbers (origin 1899-12-30).

Usage

```
compute_sale_age(contract_vals, effective_date)
```

Arguments

`contract_vals`
: A vector of contract/sale dates. Accepted types: `POSIXct`, `Date`, `character`, or `numeric` (Excel serial dates).

`effective_date`
: The effective (appraisal) date. Accepted types: `Date`, `POSIXct`, or `character` parseable by `as.POSIXct()`.

Details

This function is the non-Shiny computation kernel used by the earthUI Shiny app when computing a `sale_age` column from a designated `contract_date` column. It is also suitable for use from batch scripts.

Value

An integer vector the same length as `contract_vals`, giving the number of whole days between `effective_date` and each contract date. `NA` contract values propagate to `NA` results.

Examples

```
compute_sale_age(
  contract_vals = as.Date(c("2024-01-15", "2024-06-01")),
  effective_date = as.Date("2025-01-01")
)
```

---

convert_quarto_file *Convert a Quarto source file to one or more output formats*

Description

Renders any `.qmd` file (not just earthUI-generated ones) to the requested formats via `quarto::quarto_render()`. Useful for converting hand-edited or manually-combined Quarto reports — e.g., a master document that uses `{{< include >}}` to pull in multiple project reports.

Usage

```
convert_quarto_file(
  qmd_path,
  formats = c("html"),
  output_dir = NULL,
  paper_size = "letter"
)
```

Arguments

`qmd_path`	Path to a Quarto source (`.qmd`) file.
`formats`	Character vector of output formats. Any subset of `c("html", "docx", "pdf")`.
`output_dir`	Directory to write the rendered output(s). Defaults to the same directory as `qmd_path`.
`paper_size`	Character: `"letter"` or `"a4"`. Used only for PDF output. Default `"letter"`.

Value

Invisibly, a character vector of output file paths.

country_choices *List all countries with shipped admin schemas*

Description

Returns a named character vector of country display names indexed by ISO 3166-1 alpha-2 code. Suitable for a `selectInput` choices list.

Usage

```
country_choices()
```

Value

Named character vector. Names are display names, values are lowercase 2-letter codes.

country_schema *Get the admin-level schema for a country*

Description

Returns the ordered character vector of admin level labels for the given ISO 3166-1 alpha-2 country code. Used by the UI cascade and by `regproj_path()` to validate path depth.

Usage

```
country_schema(cc)
```

Arguments

`cc`	Character scalar. Lowercase ISO 3166-1 alpha-2 country code (e.g. `"us"`, `"jp"`, `"it"`).

Details

Unknown country codes return a generic 2-level fallback (`c("region", "city")`) so users in countries not yet covered by the shipped table still get a sensible cascade.

Value

Character vector of admin level labels, top to bottom.

default_regproj_root *Default regProj root for the current OS*

Description

Resolution order:

1. `REGPROJ_ROOT` environment variable.
2. `regproj_root` field in user prefs (`earthui_prefs_path()`).
3. Per-OS default: `C:/regProj` on Windows; `~/regProj` elsewhere.

Usage

```
default_regproj_root()
```

Value

Character scalar. Absolute path.

detect_categoricals *Detect likely categorical variables in a data frame*

Description

Returns a logical named vector indicating which columns are likely categorical. Character and factor columns are always flagged. Numeric columns with fewer than `max_unique` unique values are also flagged.

Usage

```
detect_categoricals(df, max_unique = 10L)
```

Arguments

`df`	A data frame.
`max_unique`	Integer. Numeric columns with this many or fewer unique values are flagged as likely categorical. Default is 10.

Value

A named logical vector with one element per column. `TRUE` indicates the column is likely categorical.

Examples

```
df <- data.frame(
  price = c(100, 200, 300, 400),
  pool = c("Y", "N", "Y", "N"),
  bedrooms = c(2, 3, 2, 4),
  sqft = c(1200, 1500, 1300, 1800)
)
detect_categoricals(df)
```

`detect_types` *Detect column types in a data frame*

Description

Inspects each column and returns a best-guess R type string. Character columns are tested for common date patterns. Numeric columns containing only 0/1 values (with both present) are flagged as logical.

Usage

```
detect_types(df)
```

Arguments

`df`	A data frame.

Value

A named character vector with one element per column. Possible values: `"numeric"`, `"integer"`, `"character"`, `"logical"`, `"factor"`, `"Date"`, `"POSIXct"`, `"unknown"`.

Examples

```
df <- data.frame(
  price = c(100.5, 200.3, 300.1),
  rooms = c(2L, 3L, 4L),
  pool  = c("Y", "N", "Y"),
  sold  = c(TRUE, FALSE, TRUE)
)
detect_types(df)
```

earthui_prefs_path *Path to the per-user earthUI preferences file*

Description

Returns the path to <R_user_dir("earthUI","config")>/prefs.json. The file holds user-level configuration that lives outside the regProj tree itself — most importantly, the location of the regProj root.

Usage

```
earthui_prefs_path()
```

Value

Character scalar.

earthui_prefs_read *Read user preferences (returns empty list if file missing)*

Description

Read user preferences (returns empty list if file missing)

Usage

```
earthui_prefs_read()
```

Value

Named list.

earthui_prefs_write *Write user preferences (atomic; creates the config dir if needed)*

Description

Write user preferences (atomic; creates the config dir if needed)

Usage

```
earthui_prefs_write(prefs)
```

Arguments

prefs	Named list to save.

Value

Invisibly, the prefs path.

`export_settings` *Export saved earthUI settings for one file+purpose to a JSON file*

Description

The Shiny app persists per-file, per-purpose settings in a SQLite DB keyed by "<filename>||<purpose>". `export_settings()` reads that row and writes a single JSON file containing the full settings bundle (target, earth parameters, variable selections, type/special overrides, and interactions), plus an `rca` block for batch RCA inputs — the subject CQA score and CQA score type.

Usage

```
export_settings(filename, purpose, output_json)
```

Arguments

filename	Character scalar. The filename as stored in the DB (e.g. "Appraisal_1.csv", without any path prefix).
purpose	Character scalar: "general", "appraisal", or "market".
output_json	Character scalar. Destination file path (.json).

Details

If `output_json` already exists, the `rca` block of the existing file is preserved — re-exporting from the UI does not clobber hand-edited CQA inputs.

The emitted `rca` block has two fields:

cqa_score `null` or a number in `[0.00, 10.00]`.

cqa_score_type "CQA/sf" (based on residual / living-area, default) or "CQA" (based on residual).

The emitted `reports` field is an array of formats to render in batch mode: any subset of "html", "pdf", "docx". An empty array `[]` (default) means no reports are generated.

Value

Invisibly, the `output_json` path.

Examples

```
## Not run:
export_settings("Appraisal_1.csv", "appraisal",
                "~/configs/Appraisal_1.json")

## End(Not run)
```

`fit_earth` *Fit an earth model*

Description

Wrapper around `earth::earth()` with parameter validation and automatic cross-validation when interaction terms are enabled.

Usage

```
fit_earth(
  df,
  target,
  predictors,
  categoricals = NULL,
  linpreds = NULL,
  type_map = NULL,
  degree = 1L,
  allowed_func = NULL,
  allowed_matrix = NULL,
  nfold = NULL,
  nprune = NULL,
  thresh = NULL,
  penalty = NULL,
  minspan = NULL,
  endspan = NULL,
  fast.k = NULL,
  pmethod = NULL,
  glm = NULL,
  trace = NULL,
  nk = NULL,
  newvar.penalty = NULL,
  fast.beta = NULL,
  ncross = NULL,
  stratify = NULL,
  varmod.method = NULL,
  varmod.exponent = NULL,
  varmod.conv = NULL,
  varmod.clamp = NULL,
  varmod.minspan = NULL,
```

```
  keepxy = NULL,
  Scale.y = NULL,
  Adjust.endspan = NULL,
  Auto.linpreds = NULL,
  Force.weights = NULL,
  Use.beta.cache = NULL,
  Force.xtx.prune = NULL,
  Get.leverages = NULL,
  Exhaustive.tol = NULL,
  wp = NULL,
  weights = NULL,
  ...,
  .capture_trace = TRUE
)
```

Arguments

df	A data frame containing the modeling data.
target	Character string. Name of the response variable.
predictors	Character vector. Names of predictor variables.
categoricals	Character vector. Names of predictors to treat as categorical (converted to factors before fitting). Default is NULL.
linpreds	Character vector. Names of predictors constrained to enter the model linearly (no hinge functions). Default is NULL.
type_map	Named list or character vector. Maps column names to declared types (e.g., "numeric", "Date", "factor"). When provided, columns are coerced before fitting: Date/POSIXct columns are converted to numeric (days/seconds since epoch), and type-derived categoricals are merged into categoricals. Default is NULL (no coercion).
degree	Integer. Maximum degree of interaction. Default is 1 (no interactions). When >= 2, cross-validation is automatically enabled.
allowed_func	Function or NULL. An allowed function as returned by build_allowed_function(). Only used when degree >= 2.
allowed_matrix	Logical matrix or NULL. The allowed interaction matrix. Stored in the result for report export. Not used for fitting (use allowed_func instead).
nfold	Integer. Number of cross-validation folds. Automatically set to 10 when degree >= 2 unless explicitly provided. Set to 0 to disable.
nprune	Integer or NULL. Maximum number of terms in the pruned model.
thresh	Numeric. Forward stepping threshold. Default is earth's default (0.001).
penalty	Numeric. Generalized cross-validation penalty per knot. Default is earth's default (if degree > 1, 3; otherwise 2).
minspan	Integer or NULL. Minimum number of observations between knots.
endspan	Integer or NULL. Minimum number of observations before the first and after the last knot.

fast.k	Integer. Maximum number of parent terms considered at each step of the forward pass. Default is earth's default (20).
pmethod	Character. Pruning method. One of "backward", "none", "exhaustive", "forward", "seqrep", "cv". Default is "backward".
glm	List or NULL. If provided, passed to earth's glm argument to fit a GLM on the earth basis functions.
trace	Numeric. Trace earth's execution. 0 (default) = none, 0.3 = variance model, 0.5 = cross validation, 1-5 = increasing detail.
nk	Integer or NULL. Maximum number of model terms before pruning.
newvar.penalty	Numeric or NULL. Penalty for adding a new variable in the forward pass (Friedman's gamma). Default 0.
fast.beta	Numeric or NULL. Fast MARS ageing coefficient. Default 1.
ncross	Integer or NULL. Number of cross-validations. Default 1.
stratify	Logical or NULL. Stratify cross-validation samples. Default TRUE.
varmod.method	Character or NULL. Variance model method. One of "none", "const", "lm", "rlm", "earth", "gam", "power", "power0", "x.lm", "x.rlm", "x.earth", "x.gam".
varmod.exponent	Numeric or NULL. Power transform for variance model.
varmod.conv	Numeric or NULL. Convergence criterion for IRLS.
varmod.clamp	Numeric or NULL. Minimum estimated standard deviation.
varmod.minspan	Integer or NULL. minspan for internal variance model.
keepxy	Logical or NULL. Retain x, y in model object. Default FALSE.
Scale.y	Logical or NULL. Scale response internally. Default TRUE.
Adjust.endspan	Numeric or NULL. Interaction endspan multiplier. Default 2.
Auto.linpreds	Logical or NULL. Auto-detect linear predictors. Default TRUE.
Force.weights	Logical or NULL. Force weighted code path. Default FALSE.
Use.beta.cache	Logical or NULL. Cache coefficients in forward pass. Default TRUE.
Force.xtx.prune	Logical or NULL. Force X'X-based pruning. Default FALSE.
Get.leverages	Logical or NULL. Calculate hat values. Default TRUE.
Exhaustive.tol	Numeric or NULL. Condition number threshold for exhaustive pruning. Default 1e-10.
wp	Numeric vector or NULL. Response weights.
weights	Numeric vector or NULL. Case weights passed to earth.
...	Additional arguments passed to earth::earth().
.capture_trace	Logical. If TRUE (default), capture earth's trace output. Set to FALSE when running in a background process.

Value

A list with class "earthUI_result" containing:

model The fitted earth model object.

target Name of the response variable.

predictors Names of predictor variables used.

categoricals Names of categorical predictors.

degree Degree of interaction used.

cv_enabled Logical; whether cross-validation was used.

data The data frame used for fitting.

Examples

```
# Using the included demo appraisal dataset
demo_file <- system.file("extdata", "Appraisal_1.csv", package = "earthUI")
df <- import_data(demo_file)
result <- fit_earth(df, target = "sale_price",
                    predictors = c("living_sqft", "lot_size", "age"))
format_summary(result)
```

---

format_anova *Format ANOVA decomposition*

Description

Extracts the ANOVA table from a fitted earth model.

Usage

```
format_anova(earth_result)
```

Arguments

earth_result	An object of class "earthUI_result" as returned by fit_earth().

Value

A data frame with the ANOVA decomposition showing which predictors contribute to each basis function and their importance.

Examples

```
result <- fit_earth(mtcars, "mpg", c("cyl", "disp", "hp", "wt"))
format_anova(result)
```

`format_model_equation` *Format earth model as LaTeX equation*

Description

Converts a fitted earth model into a LaTeX-formatted mathematical representation using g-function notation. Basis functions are grouped by degree (constant, first-degree, second-degree, third-degree) and labeled with indices that encode the group, position, and factor variable count.

Usage

```
format_model_equation(earth_result, digits = 7L, response_idx = NULL)
```

Arguments

`earth_result`	An object of class `"earthUI_result"` as returned by `fit_earth()`.
`digits`	Integer. Number of significant digits for coefficients and cut points. Default is 7.
`response_idx`	Integer or `NULL`. For multivariate models, which response column to generate the equation for (1-based). Default `NULL` returns all response equations in an `earthUI_equation_multi` object.

Value

A list containing:

latex Character string. LaTeX array environment for HTML/MathJax rendering.

latex_inline Character string. Wrapped in display math delimiters for MathJax/HTML rendering.

latex_pdf Character string. LaTeX for native PDF output with escaped special characters in text blocks.

latex_word Character string. LaTeX for Word/docx output.

groups List of group structures for programmatic access.

Examples

```
result <- fit_earth(mtcars, "mpg", c("cyl", "disp", "hp", "wt"))
eq <- format_model_equation(result)
cat(eq$latex)
```

`format_summary` *Format earth model summary*

Description

Extracts key statistics from a fitted earth model including coefficients, basis functions, R-squared, GCV, GRSq, and RSS.

Usage

```
format_summary(earth_result)
```

Arguments

`earth_result`	An object of class `"earthUI_result"` as returned by `fit_earth()`.

Value

A list containing:

coefficients Data frame of model coefficients and basis functions.

r_squared Training R-squared.

gcv Generalized cross-validation value.

grsq Generalized R-squared (1 - GCV/variance).

rss Residual sum of squares.

n_terms Number of terms in the pruned model.

n_predictors Number of predictors used in the final model.

n_obs Number of observations.

cv_rsq Cross-validated R-squared (if CV was used, else NA).

Examples

```
result <- fit_earth(mtcars, "mpg", c("cyl", "disp", "hp", "wt"))
summary_info <- format_summary(result)
summary_info$r_squared
```

format_variable_importance

Format variable importance

Description

Extracts variable importance scores from a fitted earth model using `earth::evimp()`.

Usage

```
format_variable_importance(earth_result)
```

Arguments

`earth_result`	An object of class `"earthUI_result"` as returned by `fit_earth()`.

Value

A data frame with columns `variable`, `nsubsets`, `gcv`, and `rss`, sorted by overall importance (nsubsets).

Examples

```
result <- fit_earth(mtcars, "mpg", c("cyl", "disp", "hp", "wt"))
format_variable_importance(result)
```

generate_quarto_report

Generate a Quarto report source bundle (without rendering)

Description

Writes a self-contained Quarto project under `<dest_dir>/<base>_qmd/` containing the populated `<base>.qmd` source, all pre-generated plots (PNG + PDF), the report data RDS, and `reference.docx` for Word rendering. The resulting bundle can be edited or combined with other Quarto sources, then rendered to HTML / Word / PDF via `convert_quarto_file()`.

Usage

```
generate_quarto_report(earth_result, dest_dir, base = "earth_report")
```

Arguments

`earth_result`	An object of class `"earthUI_result"` as returned by `fit_earth()`.
`dest_dir`	Directory to write the bundle into. The bundle itself lives at `<dest_dir>/<base>_qmd/`.
`base`	Bundle base name (no extension). The .qmd file inside is named `<base>.qmd`.

Details

Use this when you want the Quarto source as a first-class artifact — e.g., to combine reports from multiple projects into a master document before publishing.

Value

Invisibly, the absolute path to the generated `.qmd` file.

`get_project_settings` *Get model settings for a project*

Description

Reads model settings for a project from `<REGPROJ_ROOT>/projects.sqlite`. Used by the Shiny UI to restore settings on file open, and by external tools (ValEngr, batch scripts) to inspect project state. Settings are scoped by project + purpose and are shared across all data files in the project (a small test extract and the full dataset share one config).

Usage

```
get_project_settings(
  project_path,
  method = "earth",
  purpose = "general",
  root = default_regproj_root()
)
```

Arguments

`project_path`	Absolute path to the project root.
`method`	One of `"earth"`, `"glmnet"`, `"mgcv"`. Default `"earth"`.
`purpose`	Settings scope: one of `"general"`, `"appraisal"`, `"market"`. Settings are stored independently per purpose. Default `"general"`.
`root`	regProj root. Defaults to `default_regproj_root()`.

Value

Named list with `settings`, `variables`, and (for `earth`) `interactions` (each a JSON string), or `NULL` if no row exists.

import_data *Import data from CSV or Excel files*

Description

Reads a CSV (.csv) or 'Excel' (.xlsx, .xls) file and returns a data frame. Column names are converted to snake_case and duplicates are made unique.

Usage

```
import_data(filepath, sheet = 1, sep = ",", dec = ".", ...)
```

Arguments

filepath	Character string. Path to the data file. Supported formats: .csv, .xlsx, .xls.
sheet	Character or integer. For Excel files, the sheet to read. Defaults to the first sheet. Ignored for CSV files.
sep	Character. Field separator for CSV files. Default ",". Use ";" for European-style CSVs.
dec	Character. Decimal separator for CSV files. Default ".". Use "," for European-style CSVs.
...	Additional arguments passed to utils::read.csv() or readxl::read_excel().

Value

A data frame with column names converted to snake_case. Duplicate column names are made unique by appending numeric suffixes.

Examples

```
# Load the included demo appraisal dataset
demo_file <- system.file("extdata", "Appraisal_1.csv", package = "earthUI")
df <- import_data(demo_file)
head(df)
```

is_project_dir *Test whether a path is a regProj project leaf folder*

Description

Returns TRUE if path exists, sits two directories deep under root (<purpose>/<flat_segment>/), and the flat segment parses cleanly.

Usage

```
is_project_dir(path, root = default_regproj_root())
```

Arguments

path	Absolute path to check.
root	regProj root. Defaults to default_regproj_root().

Value

Logical scalar.

---

launch *Launch the earthUI Shiny application*

Description

Opens an interactive 'shiny' GUI for building and exploring 'earth' (MARS-style) models. The application provides data import, variable configuration, model fitting, result visualization, and report export.

Usage

```
launch(port = 7878L, ...)
```

Arguments

port	Integer. Port number for the Shiny app. Defaults to 7878. A fixed port keeps browser-side UI preferences (theme, last-used purpose) consistent across sessions. (Model configuration is saved server-side in the project database, not in the browser.)
...	Additional arguments passed to shiny::runApp().

Value

This function does not return a value; it launches the Shiny app.

Examples

```
if (interactive()) {
  launch()
}
```

`list_g_functions` *List g-function groups from a fitted earth model*

Description

Returns a data frame describing each non-intercept g-function group from the model equation, including degree, factor count, graph dimensionality, and the number of terms. The g-function notation is ${}^{f}g_{k}^{j}$ where f = number of factor variables (top-left), j = degree of interaction (top-right), k = position within the degree group (bottom-right).

Usage

```
list_g_functions(earth_result)
```

Arguments

`earth_result`	An object of class `"earthUI_result"` as returned by `fit_earth()`.

Value

A data frame with columns:

index Integer. Sequential index (1-based).

label Character. Variable names in the group.

g_j Integer. Degree of the g-function (top-right superscript).

g_k Integer. Position within the degree (bottom-right subscript).

g_f Integer. Number of factor variables (top-left superscript).

d Integer. Graph dimensionality (degree minus factor count).

n_terms Integer. Number of terms in the group.

Examples

```
result <- fit_earth(mtcars, "mpg", c("cyl", "disp", "hp", "wt"))
list_g_functions(result)
```

os_detect *Detect the current operating system as a regProj segment*

Description

Returns "mac" on Darwin, "ubuntu" on Linux, "win11" on Windows. This is the value used as the <os> segment in regProj paths so that multi-OS output can be merged cleanly on a developer's machine.

Usage

```
os_detect()
```

Value

Character scalar: "mac", "ubuntu", or "win11".

plot_actual_vs_predicted *Plot actual vs predicted values*

Description

Creates a scatter plot of actual vs predicted values with a 1:1 reference line.

Usage

```
plot_actual_vs_predicted(earth_result, response_idx = NULL)
```

Arguments

earth_result	An object of class "earthUI_result" as returned by fit_earth().
response_idx	Integer or NULL. For multivariate models, which response column to plot (1-based). Default NULL uses the first response.

Value

A ggplot2::ggplot object.

Examples

```
result <- fit_earth(mtcars, "mpg", c("cyl", "disp", "hp", "wt"))
plot_actual_vs_predicted(result)
```

plot_contribution *Plot variable contribution*

Description

Creates a scatter plot showing each variable's actual contribution to the prediction. For each observation, the contribution is the sum of coefficient * basis function value across all terms involving that variable.

Usage

```
plot_contribution(earth_result, variable, response_idx = NULL)
```

Arguments

earth_result	An object of class "earthUI_result" as returned by fit_earth().
variable	Character string. Name of the predictor variable to plot.
response_idx	Integer or NULL. For multivariate models, which response column to plot (1-based). Default NULL uses the first response.

Value

A ggplot2::ggplot object.

Examples

```
result <- fit_earth(mtcars, "mpg", c("cyl", "disp", "hp", "wt"))
plot_contribution(result, "wt")
```

plot_correlation_matrix *Plot correlation matrix*

Description

Creates a heatmap of pairwise correlations among the target variable and numeric predictors, with cells colored by degree of correlation and values printed in each cell.

Usage

```
plot_correlation_matrix(earth_result)
```

Arguments

earth_result	An object of class "earthUI_result" as returned by fit_earth().

Value

A ggplot2::ggplot object.

Examples

```
result <- fit_earth(mtcars, "mpg", c("cyl", "disp", "hp", "wt"))
plot_correlation_matrix(result)
```

`plot_g_contour` *Plot g-function as a static contour (for reports)*

Description

Creates a ggplot2 visualization for any g-function group. For d <= 1, produces a 2D scatter plot (same as `plot_g_function()`). For d >= 2, produces a filled contour plot suitable for static formats like PDF and Word.

Usage

```
plot_g_contour(earth_result, group_index, response_idx = NULL)
```

Arguments

`earth_result`	An object of class `"earthUI_result"` as returned by `fit_earth()`.
`group_index`	Integer. Index of the g-function group (1-based, from `list_g_functions()`).
`response_idx`	Integer or `NULL`. For multivariate models, which response column to plot (1-based). Default `NULL` uses the first response.

Value

A ggplot2::ggplot object.

Examples

```
result <- fit_earth(mtcars, "mpg", c("cyl", "disp", "hp", "wt"))
plot_g_contour(result, 1)
```

plot_g_function *Plot g-function contribution*

Description

Creates a contribution plot for a specific g-function group. For degree-1 groups (single variable), produces a 2D scatter + piecewise-linear plot with slope labels and knot markers. For degree-2 groups (two variables), produces a 3D surface plot using plotly if available, or a filled contour plot.

Usage

```
plot_g_function(earth_result, group_index, response_idx = NULL)
```

Arguments

earth_result	An object of class "earthUI_result" as returned by fit_earth().
group_index	Integer. Index of the g-function group (1-based, from list_g_functions()).
response_idx	Integer or NULL. For multivariate models, which response column to plot (1-based). Default NULL uses the first response.

Value

A ggplot2::ggplot object for d <= 1, or a plotly widget for d >= 2 (when plotly is installed). Falls back to ggplot2 contour if plotly is not available.

Examples

```
result <- fit_earth(mtcars, "mpg", c("cyl", "disp", "hp", "wt"))
plot_g_function(result, 1)
```

plot_g_persp *Plot g-function as a static 3D perspective (for reports)*

Description

Creates a base R persp() 3D surface plot for g-function groups with d >= 2. For d <= 1, produces a 2D scatter plot (same as plot_g_function()). The surface is colored by contribution value using a blue-white-red scale. Suitable for PDF and Word output where interactive plotly is not available.

Usage

```
plot_g_persp(
  earth_result,
  group_index,
  theta = 30,
  phi = 25,
  response_idx = NULL
)
```

Arguments

earth_result	An object of class "earthUI_result" as returned by fit_earth().
group_index	Integer. Index of the g-function group (1-based, from list_g_functions()).
theta	Numeric. Azimuthal rotation angle in degrees. Default 30.
phi	Numeric. Elevation angle in degrees. Default 25.
response_idx	Integer or NULL. For multivariate models, which response column to plot (1-based). Default NULL uses the first response.

Value

Invisible NULL (base graphics). For d <= 1, returns a ggplot object.

Examples

```
result <- fit_earth(mtcars, "mpg", c("cyl", "disp", "hp", "wt"), degree = 2L)
plot_g_persp(result, 1)
```

---

plot_partial_dependence

Plot partial dependence

Description

Creates a partial dependence plot for a selected variable from a fitted earth model.

Usage

```
plot_partial_dependence(
  earth_result,
  variable,
  n_grid = 50L,
  response_idx = NULL
)
```

Arguments

earth_result	An object of class "earthUI_result" as returned by fit_earth().
variable	Character string. Name of the predictor variable to plot.
n_grid	Integer. Number of grid points for the partial dependence calculation. Default is 50.
response_idx	Integer or NULL. For multivariate models, which response column to plot (1-based). Default NULL uses the first response.

Value

A ggplot2::ggplot object.

Examples

```
result <- fit_earth(mtcars, "mpg", c("cyl", "disp", "hp", "wt"))
plot_partial_dependence(result, "wt")
```

---

plot_qq *Plot Q-Q plot of residuals*

Description

Creates a normal Q-Q plot of the model residuals.

Usage

```
plot_qq(earth_result, response_idx = NULL)
```

Arguments

earth_result	An object of class "earthUI_result" as returned by fit_earth().
response_idx	Integer or NULL. For multivariate models, which response column to plot (1-based). Default NULL uses the first response.

Value

A ggplot2::ggplot object.

Examples

```
result <- fit_earth(mtcars, "mpg", c("cyl", "disp", "hp", "wt"))
plot_qq(result)
```

plot_residuals *Plot residual diagnostics*

Description

Creates a two-panel diagnostic plot: residuals vs fitted values and a Q-Q plot of residuals.

Usage

```
plot_residuals(earth_result, response_idx = NULL)
```

Arguments

earth_result	An object of class "earthUI_result" as returned by fit_earth().
response_idx	Integer or NULL. For multivariate models, which response column to plot (1-based). Default NULL uses the first response.

Value

A ggplot2::ggplot object showing residuals vs fitted values. Use plot_qq() for the Q-Q plot separately.

Examples

```
result <- fit_earth(mtcars, "mpg", c("cyl", "disp", "hp", "wt"))
plot_residuals(result)
```

plot_variable_importance *Plot variable importance*

Description

Creates a horizontal bar chart of variable importance from a fitted earth model.

Usage

```
plot_variable_importance(earth_result, type = "nsubsets")
```

Arguments

earth_result	An object of class "earthUI_result" as returned by fit_earth().
type	Character. Importance metric to plot: "nsubsets" (default), "gcv", or "rss".

Value

A ggplot2::ggplot object.

Examples

```
result <- fit_earth(mtcars, "mpg", c("cyl", "disp", "hp", "wt"))
plot_variable_importance(result)
```

---

prepare_report_assets *Prepare report assets*

Description

Pre-generates all plots and data for the earth model report. Returns the path to a directory containing all assets. This directory can be passed to `render_report()` to avoid re-computing anything during rendering.

Usage

```
prepare_report_assets(earth_result, assets_dir = NULL)
```

Arguments

`earth_result`	An object of class `"earthUI_result"` as returned by `fit_earth()`.
`assets_dir`	Character. Path to write assets. If `NULL`, a temporary directory is created.

Value

The path to the assets directory (invisibly).

Examples

```
result <- fit_earth(mtcars, "mpg", c("cyl", "disp", "hp", "wt"))
assets <- prepare_report_assets(result)
```

regproj_flat_segment *Encode the flat regProj segment from components*

Description

Joins country, admin levels, and project name with _ to produce the single hierarchy-encoding folder name used at the project level under <root>/<purpose>/. Validates each component: admin codes must match ^[a-z0-9-]+$ (no internal underscores), project name allows ^[A-Za-z0-9_-]+$.

Usage

```
regproj_flat_segment(country, levels, project_name)
```

Arguments

country	Lowercase ISO 3166-1 alpha-2 country code (e.g. "us").
levels	Character vector of admin codes, ordered top to bottom. Length must match length(country_schema(country)). Each element must satisfy ^[a-z0-9-]+$ (no underscores — they are the segment separator).
project_name	Project leaf name. Must satisfy ^[A-Za-z0-9_-]+$, max 24 characters. Underscores are allowed because the parser uses the country schema to know how many tokens are admin vs project.

Value

Character scalar (e.g. "us_ca_081_burlin_20251231_j").

regproj_geo_db_connect *Open (and initialize if needed) the regProj geo database*

Description

Returns a DBI connection. Creates the schema on first call. The caller is responsible for DBI::dbDisconnect().

Usage

```
regproj_geo_db_connect(root = default_regproj_root())
```

Arguments

root	regProj root. Defaults to default_regproj_root().

Details

On first creation, the tables are seeded from:

- the shipped reference data (`regproj_reference()`) — 24 countries, 51 US states, 3,076 US counties.
- the shipped places data (`pkg/inst/extdata/regproj_geo.rds`) — US incorporated places + GeoNames-derived city/admin data for GB, DE, IT, FR, SE, SG.
- any pre-existing `<REGPROJ_ROOT>/.regproj-index.json` (legacy migration).

Value

A `DBIConnection` to the SQLite database.

---

regproj_geo_db_path *Path to the regProj geo SQLite database*

Description

`<REGPROJ_ROOT>/geo.sqlite` — holds country codes and a flexible, variable-depth `admin_entries` table for state / county / city / deeper-level admin codes per country. Travels with the regProj tree.

Usage

```
regproj_geo_db_path(root = default_regproj_root())
```

Arguments

`root`	regProj root. Defaults to `default_regproj_root()`.

Value

Character scalar.

---

regproj_in_files *List input files in an active project's in/ folder*

Description

Returns the basenames of regular files in `<project_path>/<os>_in/`. Hidden dotfiles and subdirectories are excluded. Sorted alphabetically.

Usage

```
regproj_in_files(project_path, os = os_detect())
```

Arguments

project_path	Absolute path to the project root (the flat segment folder).
os	OS segment to use. Defaults to os_detect().

Value

Character vector of file basenames.

---

regproj_index_get *Get a code for a full name within a scope*

Description

Looks up full_name under scope (e.g. "us/ca") in the geo SQLite DB (which is seeded with shipped reference data on first creation). Returns NULL if not found.

Usage

```
regproj_index_get(scope, full_name, root = default_regproj_root())
```

Arguments

scope	Character scalar. Slash-separated path of parent codes (e.g. "us/ca" for California's counties; "us/ca/001" for Alameda's cities). Empty string for top-level (countries).
full_name	Character scalar. The display name to look up.
root	regProj root. Defaults to default_regproj_root().

Value

Character scalar code, or NULL.

---

regproj_index_path *Path to the central regProj name-to-code index (legacy)*

Description

The geo data is now stored in <REGPROJ_ROOT>/geo.sqlite (see regproj_geo_db_path()). This path returns the location of the legacy .regproj-index.json file, which is migrated into the SQLite DB on first connect and is no longer written to.

Usage

```
regproj_index_path(root = default_regproj_root())
```

Arguments

root	regProj root. Defaults to default_regproj_root().

Value

Character scalar.

---

regproj_index_put *Set a name-to-code mapping in the geo SQLite DB*

Description

Inserts (or replaces) the mapping under the given scope.

Usage

```
regproj_index_put(scope, full_name, code, root = default_regproj_root())
```

Arguments

scope	Character scalar. Slash-separated path of parent codes (e.g. "us/ca" for California's counties; "us/ca/001" for Alameda's cities). Empty string for top-level (countries).
full_name	Character scalar. The display name to look up.
code	Character scalar. The path code to assign.
root	regProj root. Defaults to default_regproj_root().

Value

Invisibly, the code.

---

regproj_index_read *Read the central regProj index from the geo SQLite DB*

Description

Returns the entire geo DB as a nested list keyed by scope (slash- separated parent-code path), for backward compatibility with callers that iterate. New code should prefer regproj_index_get() or direct DB queries via regproj_geo_db_connect().

Usage

```
regproj_index_read(root = default_regproj_root())
```

Arguments

root	regProj root. Defaults to default_regproj_root().

Value

Named list. Outer key is scope ("" for countries, then slash-separated codes per admin level). Inner key is full name; value is the code.

---

regproj_index_write *Deprecated. The geo data is now in SQLite; this is a no-op kept for backward compatibility.*

Description

Deprecated. The geo data is now in SQLite; this is a no-op kept for backward compatibility.

Usage

```
regproj_index_write(idx, root = default_regproj_root())
```

Arguments

idx	Ignored.
root	regProj root. Defaults to default_regproj_root().

Value

Invisibly, the geo DB path.

---

regproj_last_file *Last-used input file marker (per project)*

Description

Stores the basename of the most-recently-selected input file in <project>/<os>/.regproj-last, so it can be auto-selected the next time the project is opened.

Usage

```
regproj_last_file_path(project_path, os = os_detect())

regproj_last_file_get(project_path, os = os_detect())

regproj_last_file_set(project_path, basename, os = os_detect())
```

Arguments

project_path	Absolute project path.
os	OS segment. Defaults to os_detect().
basename	File basename to remember.

Value

Path / basename / invisible path.

---

regproj_list_projects *List all projects under a regProj root*

Description

Walks <root>/<purpose>/<country>/<state>/<county>/<city>/<project_name>/ and returns a data frame describing each project found, with its mtime (most recent of the in/ and out/ trees, falling back to the project folder itself).

Usage

```
regproj_list_projects(
  root = default_regproj_root(),
  sort_by = c("recent", "alpha")
)
```

Arguments

root	regProj root. Defaults to default_regproj_root().
sort_by	"recent" (mtime descending, default) or "alpha" (project name ascending).

Details

Recognized purposes: gen, appr, mktarea. Other top-level subdirectories under <root> are ignored, including any leading-dot files like .regproj-index.json.

Value

Data frame with columns: project_path (absolute), purpose, country, state, county, city, project_name, mtime (POSIXct). Empty data frame (correct columns, zero rows) if no projects exist or root is missing.

regproj_parse_flat *Decode a flat regProj segment back into components*

Description

Inverse of `regproj_flat_segment()`. Splits on _, takes the first token as country, then the next `length(country_schema(country))` tokens as admin levels; the remaining tokens (joined by _) are the project name. Returns `NULL` on parse failure.

Usage

```
regproj_parse_flat(segment)
```

Arguments

`segment`	Character scalar.

Value

Named list with `country`, `levels` (character vector), `project_name` — or `NULL` if parsing failed.

regproj_path *Build a regProj-canonical path*

Description

Composes the path `<root>/<purpose>/<flat_segment>/<os>_<in|out>[_<method>]` from its components. The hierarchy (country / admin levels / project name) is concatenated into a single folder under `<purpose>/` to keep the tree shallow. Pure path computation — does not create any directories unless `create = TRUE`.

Usage

```
regproj_path(
  purpose,
  country,
  levels,
  project_name,
  os = os_detect(),
  in_or_out = c("out", "in"),
  method = "earth",
  root = default_regproj_root(),
  create = FALSE
)
```

Arguments

purpose	"gen", "appr", or "mktarea".
country	Lowercase ISO 3166-1 alpha-2 country code (e.g. "us").
levels	Character vector of admin codes, ordered top to bottom. Length must match length(country_schema(country)). Each element must satisfy ^[a-z0-9-]+$ (no underscores — they are the segment separator).
project_name	Project leaf name. Must satisfy ^[A-Za-z0-9_-]+$, max 24 characters. Underscores are allowed because the parser uses the country schema to know how many tokens are admin vs project.
os	One of "mac", "ubuntu", "win11". Defaults to os_detect().
in_or_out	One of "in" or "out". Defaults to "out".
method	Optional method subdir (e.g. "earth", "glmnet", "mgcv", "combined"). When in_or_out = "out" and method is non-empty, the leaf becomes <os>_out_<method>. Ignored for "in".
root	Optional explicit regProj root. Defaults to default_regproj_root().
create	Logical. If TRUE, the directory tree is created with dir.create(recursive = TRUE). Defaults to FALSE.

Value

Character scalar. Absolute normalized path.

---

regproj_projects_db_connect

Open (and initialize if needed) the regProj projects database

---

Description

Returns a DBI connection. Creates the schema on first call. The caller is responsible for DBI::dbDisconnect().

Usage

```
regproj_projects_db_connect(root = default_regproj_root())
```

Arguments

root	regProj root. Defaults to default_regproj_root().

Value

A DBIConnection to the SQLite database.

regproj_projects_db_path *Path to the regProj projects SQLite database*

Description

<REGPROJ_ROOT>/projects.sqlite — one row per project, keyed by the flat segment (e.g. "us_ca_081_burlin_20251231
Holds project metadata and per-method settings as JSON blobs.

Usage

```
regproj_projects_db_path(root = default_regproj_root())
```

Arguments

root	regProj root. Defaults to default_regproj_root().

Value

Character scalar.

regproj_reference *Load the shipped regProj reference data*

Description

Reads pkg/inst/extdata/regproj_reference.json once per session and caches the result. Contains country names, US states, and US counties (with FIPS codes). Used by the UI cascades to populate dropdowns.

Usage

```
regproj_reference()
```

Value

A nested list with components version, countries, states, counties.

render_report *Render an earth model report*

Description

Renders a parameterized 'Quarto' report from the fitted 'earth' model results. Requires the 'quarto' R package and a 'Quarto' installation.

Usage

```
render_report(
  earth_result,
  output_format = "html",
  output_file = NULL,
  paper_size = "letter",
  assets_dir = NULL
)
```

Arguments

earth_result	An object of class "earthUI_result" as returned by fit_earth().
output_format	Character. Output format: "html", "pdf", or "docx". Default is "html".
output_file	Character. Path for the output file. If NULL, a temporary file is created.
paper_size	Character. Paper size for PDF output: "letter" (US) or "a4" (European). Default is "letter". Ignored for HTML/Word.
assets_dir	Character or NULL. Path to pre-generated assets from prepare_report_assets(). If NULL, assets are generated on-the-fly.

Value

The path to the rendered output file (invisibly).

Examples

```
result <- fit_earth(mtcars, "mpg", c("cyl", "disp", "hp", "wt"))
render_report(result, output_format = "html",
              output_file = tempfile(fileext = ".html"))
```

`select_sales_grid_comps` *Select comparable sales for the Sales Comparison Grid*

Description

Given an RCA-adjusted data frame (subject in row 1, comps in rows 2+), builds comp-summary tables, filters by weight, sorts by gross adjustment percentage, splits into "recommended" (gross adjustment < threshold) and "others", and caps the recommended list.

Usage

```
select_sales_grid_comps(
  rca_df,
  sale_age_col = "sale_age",
  min_weight = 0,
  max_gross_adj_pct = 0.25,
  max_recommended = 30L
)
```

Arguments

`rca_df`	A data frame produced by the RCA workflow. Row 1 is the subject; rows 2+ are comps. Expected columns (any may be missing; NA is substituted): `id`, `street_address`, `sale_price`, `weight`, `gross_adjustments`, plus a sale-age column named by `sale_age_col`.
`sale_age_col`	Character scalar giving the column name that holds sale age in days. Defaults to `"sale_age"`.
`min_weight`	Numeric. Comps with `weight` strictly greater than this value are eligible. Default `0`.
`max_gross_adj_pct`	Numeric. Comps whose `gross_adj_pct` is strictly below this threshold are classed as "recommended". Default `0.25` (25%).
`max_recommended`	Integer. Upper bound on the number of recommended comps returned. Default `30`.

Details

This is the non-Shiny computation kernel used by the earthUI Shiny app's Sales Grid modal, and is also suitable for use from batch scripts.

Value

A named list with:

recommended Data frame of recommended comps, sorted by `sale_age` ascending, capped at `max_recommended` rows.

others Data frame of eligible comps not in the recommended set, sorted by `gross_adj_pct` ascending.

Each data frame has columns: `row`, `id`, `address`, `sale_price`, `sale_age`, `weight`, `gross_adj`, `gross_adj_pct`.

`set_project_settings` *Set model settings for a project*

Description

Writes model settings for a project to `<REGPROJ_ROOT>/projects.sqlite`. Used by the Shiny UI to persist settings, and by external tools to seed projects programmatically. Settings are scoped by project + purpose and shared across all data files in the project.

Usage

```
set_project_settings(
  project_path,
  settings = NULL,
  variables = NULL,
  interactions = NULL,
  method = "earth",
  purpose = "general",
  root = default_regproj_root()
)
```

Arguments

`project_path`	Absolute path to the project root.
`settings`, `variables`, `interactions`	JSON strings (or `NULL` to leave unchanged).
`method`	One of `"earth"`, `"glmnet"`, `"mgcv"`. Default `"earth"`.
`purpose`	Settings scope: one of `"general"`, `"appraisal"`, `"market"`. Settings are stored independently per purpose. Default `"general"`.
`root`	regProj root. Defaults to `default_regproj_root()`.

Value

Invisibly, `NULL`.

`validate_types` *Validate declared column types against actual data*

Description

Checks each selected predictor's actual data against the user-declared type. Returns a list of errors (blocking), warnings (non-blocking), and any Date/POSIXct columns that will be auto-converted to numeric.

Usage

```
validate_types(df, type_map, predictors)
```

Arguments

`df`	A data frame.
`type_map`	Named list or character vector. Names are column names, values are declared types (e.g., `"numeric"`, `"Date"`).
`predictors`	Character vector of selected predictor column names.

Value

A list with components:

ok Logical. `TRUE` if no blocking errors found.

warnings Character vector of non-blocking warnings.

errors Character vector of blocking errors.

date_columns Character vector of Date/POSIXct predictor columns that will be auto-converted to numeric.

Examples

```
df <- data.frame(price = c(100, 200, 300), city = c("A", "B", "C"))
types <- list(price = "numeric", city = "character")
validate_types(df, types, predictors = c("price", "city"))
```

write_earth_output *Write an earth model summary to a text file*

Description

Writes the model print-out, `summary.earth()`, and optional variance model / trace log to `<file_name>_earth_output_<tim` in `output_folder`.

Usage

```
write_earth_output(result, output_folder, file_name)
```

Arguments

`result`	A fit result list as returned by `fit_earth()` (must have `$model` at minimum; `$elapsed`, `$seed`, `$trace_output` are optional).
`output_folder`	Character scalar. May be `NULL` or empty (defaults to `~/Downloads`).
`file_name`	Character scalar. Used to derive the output filename.

Value

Invisibly, `NULL`.

write_fit_log *Write an earthUI fitting log to a text file*

Description

Writes the timestamped contents of an earth fitting trace to `<file_name>_earth_log_<timestamp>.txt` in `output_folder`. If `output_folder` is `NULL` or empty, `~/Downloads` is used. The folder is created if it doesn't exist. Errors are caught and reported via `message()` so batch pipelines don't fail on logging issues.

Usage

```
write_fit_log(output_folder, lines, file_name)
```

Arguments

`output_folder`	Character scalar. Directory in which to write the log. May be `NULL` or empty.
`lines`	Character vector of log lines.
`file_name`	Character scalar. Used to derive the log filename (the extension is stripped).

Value

Invisibly, `NULL`.

Index