Package: WMFM (via r-universe)

July 5, 2026

Title Explore Fitted Linear and Generalised Linear Models with 'shiny'

Version 1.0.4

Description Provides a 'shiny' application that helps learners connect regression tables to fitted generalised linear models. Users construct models via drag-and-drop controls, obtain fitted equations and plain-language explanations generated by a large language model, and can view plots of the fitted model in settings with a single continuous covariate.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

URL https://github.com/jmcurran/WMFM

BugReports https://github.com/jmcurran/WMFM/issues

Depends R (>= 4.1.0)

Imports bslib, ellmer, ggplot2, ggrepel, glue, grDevices, htmltools, jsonlite, patchwork, rlang, s20x, sandwich, scales, shiny, sodium, sortable, tibble, utils, yaml

Suggests clipr, devtools, rstudioapi, testthat (>= 3.0.0), withr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

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

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

Date/Publication 2026-06-21 21:49:39 UTC

RemoteUrl https://github.com/jmcurran/wmfm

RemoteRef HEAD

RemoteSha 6092618c32f8fbccd7be80da7e129e171a3e599e

Contents

`addDerivedVariableToData` *Add a derived variable to a data frame from a single assignment line*

Description

Validates and evaluates a single assignment statement such as `t = 1:nrow(data)` or `month = factor(rep(1:12, 12))`. The RHS is evaluated in a safe environment created by `makeSafeEvalEnv()`, so only the data columns plus a small allowlist of base functions are available.

Usage

```
addDerivedVariableToData(data, txt, allowOverwrite = FALSE)
```

Arguments

`data`	A data frame to modify.
`txt`	A length-1 character string containing one assignment statement.
`allowOverwrite`	Logical; if `FALSE` (default), refuse to overwrite an existing column.

Details

The derived variable is appended to the data frame as a new column. Scalars are recycled to the number of rows; otherwise the result must have length `nrow(data)`.

Value

A list with elements:

- ok: logical, whether the operation succeeded.
- msg: character status message.
- data: the updated data frame (if ok), otherwise the original data.
- name: the new variable name (if ok).
- transformation: derived-variable metadata (if ok).

Examples

```
df = data.frame(passengers = 1:144)
res = addDerivedVariableToData(df, "t = 1:nrow(data)")
res$ok
names(res$data)

res2 = addDerivedVariableToData(res$data, "month = factor(rep(1:12, 12))")
table(res2$data$month)
```

---

as.data.frame.wmfmScores

Coerce a WMFM scores object to a data frame

Description

Flattens a wmfmScores object into a long or wide data frame for analysis, plotting, and comparison.

Usage

```
## S3 method for class 'wmfmScores'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  format = c("long", "wide"),
  ...
)
```

Arguments

x	A wmfmScores object.
row.names	Ignored. Included for S3 compatibility.
optional	Ignored. Included for S3 compatibility.
format	Character. One of "long" or "wide".
...	Unused. Included for S3 compatibility.

Value

A data frame.

`auditBadExplanationGrading` *Audit whether bad explanations are being penalised by the current rubric*

Description

Builds a compact audit summary around a set of already graded bad explanations, with optional comparison against a graded good explanation. This is intended as a lightweight helper for diagnosing adversarial or deliberately flawed explanations that may be slipping through the current deterministic rubric.

Usage

```
auditBadExplanationGrading(
  x,
  goodGrade = NULL,
  method = c("deterministic", "llm"),
  minExpectedDrop = 1,
  flaggedThreshold = 9,
  expectedMetrics = NULL,
  ...
)
```

Arguments

`x`	A `wmfmGradeListObj` containing graded bad explanations.
`goodGrade`	Optional `wmfmGrade` object for a reference good explanation.
`method`	Character scalar. Grading method to audit. One of `"deterministic"` or `"llm"`.
`minExpectedDrop`	Numeric scalar. Minimum drop in mark, relative to the good explanation, required before an explanation is treated as clearly penalised.
`flaggedThreshold`	Numeric scalar. Any explanation with a mark greater than or equal to this threshold is flagged.
`expectedMetrics`	Optional named list. Each element name should match an explanation name in `x`, and each element value should be a character vector of metric names that you expected to lose marks.
`...`	Unused.

Value

An object of class `wmfmBadExplanationAudit`.

Examples

```
if (interactive()) {
badGrades = grade(modelObj, explanation = badVec, method = "deterministic")
goodGrade = grade(modelObj, explanation = modelObj$explanation, method = "deterministic")

audit = auditBadExplanationGrading(
  badGrades,
  goodGrade = goodGrade,
  expectedMetrics = list(
    effectDirectionError = c("factualScore"),
    wrongScaleError = c("factualScore", "clarityScore"),
    rSquaredOverclaim = c("factualScore", "calibrationScore"),
    logicalContradiction = c("factualScore", "clarityScore")
  )
)

print(audit)
}
```

`buildScalePhrasingRules` *Generate language rules for interpreting a linear contrast*

Description

This helper constructs a short, deterministic set of language constraints to guide natural-language interpretation of a contrast. The rules are designed to enforce wording that is consistent with the model scale (identity, log, logit, etc.) and any transformation applied to the response in an `lm()` model.

Usage

```
buildScalePhrasingRules(
  isGlm,
  effectiveScale,
  respTransform,
  nounPhrase = NULL
)
```

Arguments

`isGlm`	Logical. `TRUE` if the fitted model inherits from `"glm"`, `FALSE` for `lm()` models.
`effectiveScale`	Character string giving the interpretation scale. For GLMs this should typically be the model link (e.g. `"identity"`, `"log"`, `"logit"`). For `lm()` models this is usually `"identity"` or `"other"`.

respTransform	Character string describing the detected response transformation for lm() models. Expected values include "none", "log", "log10", "log1p", "sqrt", "inverse", or "unknown".
nounPhrase	Optional character string naming the outcome in student-facing language (e.g. "expected count"). If NULL, a generic phrase ("the outcome") is used.

Details

The returned text is intended to be embedded verbatim in an LLM prompt to prevent inconsistent phrasing (e.g., additive vs multiplicative language).

Value

A single character string containing bullet-point language rules. The text is suitable for direct inclusion in an LLM prompt.

Examples

```
buildScalePhrasingRules(
  isGlm = FALSE,
  effectiveScale = "identity",
  respTransform = "none"
)

buildScalePhrasingRules(
  isGlm = TRUE,
  effectiveScale = "logit",
  respTransform = "none",
  nounPhrase = "disease status"
)
```

---

buildWmfmRunRecord *Build a single WMFM repeated-run record*

Description

Creates a structured record for one repeated WMFM run using a schema that separates:

1. metadata and model context,
2. raw explanation text and text summaries, and
3. extracted semantic claims.

Usage

```
buildWmfmRunRecord(
  runId,
  exampleName,
  package,
  modelType,
  formula,
  equationsText,
  explanationText,
  researchQuestion = "",
  errorMessage = NA_character_,
  interactionTerms = character(0),
  interactionMinPValue = NA_real_,
  interactionAlpha = 0.05,
  hasFactorPredictors = FALSE,
  adjustmentVariables = character(0),
  primaryVariables = character(0),
  followupScoringContext = NULL
)
```

Arguments

`runId`	Integer run identifier.
`exampleName`	Character example identifier.
`package`	Character package name.
`modelType`	Character model class.
`formula`	Character model formula.
`equationsText`	Character fitted-equation text.
`explanationText`	Character explanation text.
`researchQuestion`	Character research question associated with the model.
`errorMessage`	Character error message, or `NA`.
`interactionTerms`	Character vector of interaction-term names from the fitted model.
`interactionMinPValue`	Numeric minimum p-value across fitted interaction terms, or `NA` if no interaction terms are present or no p-value could be extracted.
`interactionAlpha`	Numeric interaction threshold metadata stored with the run record.
`hasFactorPredictors`	Logical. Whether the fitted model includes at least one factor or character predictor. Used by downstream scoring gates to distinguish genuinely applicable factor criteria from numeric-only models.

`adjustmentVariables`
: Character vector of adjustment-variable names to carry into explanation scoring. These variables are treated as controls rather than primary narrative targets.

`primaryVariables`
: Character vector of variables of scientific interest to carry into explanation scoring.

`followupScoringContext`
: Optional named list of deterministic follow-up prediction metadata to carry into scoring prompts and extracted evidence.

Details

This function is intentionally limited to describing what happened during a run and what the explanation appeared to say. It does *not* create any judged scoring fields or aggregate score placeholders. Scoring is handled later by downstream functions such as `score()` and `scoreWmfmRepeatedRuns()`, which should operate on the raw run record rather than being partially embedded in it.

Output schema:

The returned named list contains the following groups of fields.

Metadata and model context:

runId Integer run identifier.

timestamp Character timestamp created at record build time.

exampleName Example identifier.

package Package name.

modelType High-level model class.

modelFamily Optional model family. Initialised to `NA`.

linkFunction Optional link function. Initialised to `NA`.

formula Model formula as character.

equationsText Fitted-equation text.

interactionTerms Pipe-separated interaction-term names.

hasInteractionTerms Logical. Whether the fitted model includes interactions.

nInteractionTerms Integer count of interaction terms.

interactionMinPValue Minimum interaction-term p-value, or `NA`.

interactionAlpha Stored interaction threshold metadata from the run.

hasError Logical. Whether an error was recorded for the run.

errorMessage Run error message, or `NA`.

Raw explanation text and summaries:

explanationText Raw explanation text.

normalizedExplanation Normalised explanation used for duplicate detection.

explanationPresent Logical. Whether a non-empty explanation is present.

wordCount Approximate word count.

sentenceCount Approximate sentence count.

Extracted claim variables:

ciMention Logical. Whether a confidence or credible interval is mentioned.

percentLanguageMention Logical. Whether percent language is used.

referenceGroupMention Logical. Whether reference/baseline framing is mentioned.

interactionMention Logical. Whether interaction or differential-pattern language is mentioned.

uncertaintyMention Logical. Whether uncertainty or evidential caution is mentioned.

usesInferentialLanguage Logical. Whether inferential wording is present.

usesDescriptiveOnlyLanguage Logical. Whether wording is descriptive only.

overclaimDetected Logical. Whether overstrong language is detected.

underclaimDetected Logical. Whether unusually weak language is detected.

conditionalLanguageMention Logical. Whether conditional wording such as "depends on" or subgroup-conditioned interpretation is present.

comparisonLanguageMention Logical. Whether explicit or implicit comparison wording is present.

outcomeMention Logical. Placeholder for whether the response is clearly named. Initialised conservatively.

predictorMention Logical. Placeholder for whether the focal predictor is clearly named. Initialised conservatively.

effectDirectionClaim Character. One of `increase`, `decrease`, `mixed_or_both`, `not_stated`.

effectScaleClaim Character. One of `additive`, `multiplicative`, `probability_or_odds`, `mixed_or_unclear`, `not_stated`.

interactionSubstantiveClaim Character. One of `difference_claimed_strongly`, `difference_claimed_cautiously`, `no_clear_difference`, `not_mentioned`, `unclear`, `not_applicable`.

inferentialRegister Character. One of `inferential`, `descriptive_only`, `overclaiming`, `unclear`.

uncertaintyTypeClaim Character. One of `none`, `generic_uncertainty`, `confidence_interval`, `mixed`, `unclear`.

Value

Named list containing one structured raw run record.

`chooseFactorLayout` *Choose fitted-means table layout for 1–3 factor predictors*

Description

Implements the layout rules for displaying fitted means when all predictors are factors:

- 1 factor: one-way table (single column of fitted means).
- 2 factors: two-way table with the factor having the most levels on rows.
- 3 factors: a set of two-way tables split by the factor with the fewest levels. Within each two-way table, the factor with the most levels is on rows and the remaining factor is on columns.

Usage

```
chooseFactorLayout(mf, predictors)
```

Arguments

mf	A model frame (as returned by model.frame(m)).
predictors	Character vector of predictor names (columns in mf).

Value

A list describing the layout:

type One of "oneWay", "twoWay", "threeWay", or "other".

rowVar Row factor name (for "oneWay" and "twoWay" and "threeWay").

colVar Column factor name (for "twoWay" and "threeWay").

splitVar Split factor name (for "threeWay" only).

Examples

```
df = data.frame(
  y = rnorm(24),
  A = factor(rep(c("a1","a2"), each = 12)),
  B = factor(rep(paste0("b", 1:4), times = 6)),
  C = factor(rep(paste0("c", 1:6), times = 4))
)
mod = lm(y ~ A * B * C, data = df)
mf = model.frame(mod)
chooseFactorLayout(mf, c("A","B","C"))
```

---

classifyEffectScaleClaim

Classify the effect scale described in a WMFM explanation

Description

Uses simple text heuristics to classify the scale on which the explanation describes the main effect. The intent is to identify the dominant scale used for coefficient interpretation, rather than unrelated numeric summaries such as model-fit percentages.

Usage

```
classifyEffectScaleClaim(text)
```

Arguments

text	Character scalar explanation text.

Details

The classifier is deliberately conservative about assigning `"mixed_or_unclear"`. In particular, generic percentage language such as an R^2 statement should not by itself force a multiplicative label when the coefficient interpretation is otherwise clearly additive.

Value

One of `"additive"`, `"multiplicative"`, `"probability_or_odds"`, `"mixed_or_unclear"`, or `"not_stated"`.

---

cleanExplanationText *Clean generated explanation text before deterministic processing*

Description

Removes simple LLM formatting artifacts that can leak into the visible explanation, such as leading `Answer`, `Answer:`, or `Answer -` tokens. This is a deterministic surface cleanup step. It does not rewrite the statistical content of the explanation.

Usage

```
cleanExplanationText(text)
```

Arguments

`text`	Character vector of explanation text.

Value

A character vector with formatting artifacts removed.

---

compare *Compare WMFM objects*

Description

Generic for comparing WMFM objects.

Usage

```
compare(x, y = NULL, ...)
```

Arguments

`x`	An object to compare.
`y`	Optional second object to compare.
`...`	Additional arguments passed to methods.

Value

Method specific comparison object.

`compare.wmfmGrade` *Compare grading results for wmfmGrade objects*

Description

Compares deterministic and llm grading either within a single `wmfmGrade` object that contains both methods, or across two separate `wmfmGrade` objects.

Usage

```
## S3 method for class 'wmfmGrade'
compare(x, y = NULL, methods = c("deterministic", "llm"), ...)
```

Arguments

`x`	A scored `wmfmGrade` object.
`y`	Optional second scored `wmfmGrade` object.
`methods`	Character vector of length 2 naming the methods to compare when comparing within a single object. Defaults to `c("deterministic", "llm")`.
`...`	Unused. Included for S3 compatibility.

Value

An object of class `wmfmGradeComparison`.

`compare.wmfmScores` *Compare WMFM score results*

Description

Compares score results either:

- within a single `wmfmScores` object containing two methods, or
- between two `wmfmScores` objects.

Usage

```
## S3 method for class 'wmfmScores'
compare(
  x,
  y = NULL,
  xMethod = NULL,
  yMethod = NULL,
  registry = getWmfmMetricRegistry(),
  ...
)
```

Arguments

x	A wmfmScores object.
y	Optional second wmfmScores object.
xMethod	Optional character string naming the method to use from x.
yMethod	Optional character string naming the method to use from y.
registry	Optional WMFM metric registry. Defaults to getWmfmMetricRegistry().
...	Unused. Included for S3 compatibility.

Details

The comparison summarizes agreement separately for binary, ordinal, and continuous score fields using the WMFM metric registry.

Value

An object of class wmfmScoreComparison.

---

computeFactorOnlyContrast

Compute a single contrast for factor-only models

Description

Computes an estimate and 95% confidence interval for a user-specified contrast in a factor-only model, optionally using robust (sandwich) variance estimators.

Usage

```
computeFactorOnlyContrast(
  model,
  newData,
  weights,
  ciType = c("standard", "sandwich"),
  hcType = c("HC0", "HC3"),
  level = 0.95
)
```

Arguments

model	A fitted model object, typically an lm or glm.
newData	A data frame whose rows define the conditions being contrasted.
weights	Numeric vector of contrast weights, one per row of newData.
ciType	One of "standard" or "sandwich".
hcType	Sandwich type passed to sandwich::vcovHC() (e.g. "HC0", "HC3").
level	Confidence level (default 0.95).

Details

The contrast is defined through a set of weights applied to the rows of newData. For example, a pairwise contrast between level A and level B can be represented with weights c(1, -1, 0, ..., 0).

For GLMs, inference is carried out on the linear predictor (link) scale and then back-transformed for interpretation when possible:

- Poisson (log link): reports a mean ratio $exp(contrast)$.
- Binomial (logit link): reports an odds ratio $exp(contrast)$.

Value

A list with estimates and confidence intervals on the link scale and (when applicable) an interpreted scale.

---

computeMeanCi *Compute confidence intervals for fitted mean responses*

Description

Computes pointwise confidence intervals for the fitted mean response evaluated at newData. These are confidence intervals for the conditional mean (i.e., the fitted line), not prediction intervals.

Usage

```
computeMeanCi(
  model,
  newData,
  ciType = "standard",
  hcType = "HC0",
  level = 0.95
)
```

Arguments

model	A fitted lm or glm object.
newData	A data frame of covariate values at which to evaluate the fitted mean.
ciType	Character string: "standard" or "sandwich".
hcType	Character string specifying the HC estimator (e.g. "HC0", "HC3"). Used only when ciType = "sandwich".
level	Confidence level for the intervals.

Details

For generalized linear models, intervals are computed on the link scale and transformed back to the response scale.

Robust (sandwich) confidence intervals are supported via vcovHC().

Value

A data frame containing newData plus columns fit, lower, and upper.

---

countWmfmSentences *Count sentences in text*

Description

Count sentences in text

Usage

```
countWmfmSentences(x)
```

Arguments

x	Character text.

Value

Integer.

countWmfmWords *Count words in text*

Description

Count words in text

Usage

```
countWmfmWords(x)
```

Arguments

x	Character text.

Value

Integer.

describeField *Describe a field for a WMFM object*

Description

S3 generic for describing fields stored in WMFM objects.

Usage

```
describeField(x, field, ...)
```

Arguments

x	A WMFM object.
field	Character scalar naming the field to describe. This may be a canonical field name, a recognised alias, or a pretty plot label if the underlying field registry supports it.
...	Additional arguments passed to methods.

Value

Method-specific description output.

`describeField.wmfmRuns` *Describe a field for a WMFM runs object*

Description

Describes a field that is present in a `wmfmRuns` object. The method accepts canonical field names, recognised aliases, and supported pretty plot labels. It validates that the resolved canonical field is actually stored in the raw run records before delegating to `describeWmfmField()`.

Usage

```
## S3 method for class 'wmfmRuns'
describeField(
  x,
  field,
  format = c("text", "list", "data.frame"),
  includeExamples = TRUE,
  includeAliases = TRUE,
  ...
)
```

Arguments

`x`	A `wmfmRuns` object.
`field`	Character scalar naming the field to describe.
`format`	Character. One of `"text"`, `"list"`, or `"data.frame"`.
`includeExamples`	Logical. Should examples be included?
`includeAliases`	Logical. Should recognised aliases be included?
`...`	Unused. Included for S3 compatibility.

Value

The result of `describeWmfmField()` in the requested format. This is a character scalar for `format = "text"`, a named list for `format = "list"`, or a one-row `data.frame` for `format = "data.frame"`.

`describeField.wmfmScores` *Describe a field for a WMFM scores object*

Description

Describes a field that is present in a `wmfmScores` object. The method accepts canonical field names, recognised aliases, and supported pretty plot labels. It validates that the resolved canonical field is actually stored in the scored records before delegating to `describeWmfmField()`.

Usage

```
## S3 method for class 'wmfmScores'
describeField(
  x,
  field,
  format = c("text", "list", "data.frame"),
  includeExamples = TRUE,
  includeAliases = TRUE,
  ...
)
```

Arguments

`x`	A `wmfmScores` object.
`field`	Character scalar naming the field to describe.
`format`	Character. One of `"text"`, `"list"`, or `"data.frame"`.
`includeExamples`	Logical. Should examples be included?
`includeAliases`	Logical. Should recognised aliases be included?
`...`	Unused. Included for S3 compatibility.

Value

The result of `describeWmfmField()` in the requested format. This is a character scalar for `format = "text"`, a named list for `format = "list"`, or a one-row `data.frame` for `format = "data.frame"`.

`describeWmfmField` *Describe a WMFM field*

Description

Prints a human-readable description of a field used in the WMFM workflow. This is intended to help interpret the raw run fields, judged fields, and aggregate score fields that appear in WMFM objects and related plots.

Usage

```
describeWmfmField(
  field,
  format = c("text", "list", "data.frame"),
  includeExamples = TRUE,
  includeAliases = TRUE
)
```

Arguments

`field`	Character scalar naming the field to describe. This may be a canonical field name, a recognised alias, or one of the pretty labels shown on the heatmap x-axis.
`format`	Character. One of `"text"`, `"list"`, or `"data.frame"`. The default `"text"` prints a formatted character vector with `cat()` and returns that vector invisibly. `"list"` returns the underlying structured description. `"data.frame"` returns a one-row data frame.
`includeExamples`	Logical. Should examples be included in the returned output?
`includeAliases`	Logical. Should recognised aliases be included in the returned output?

Details

The function accepts:

- current schema names such as `interactionEvidenceAppropriate`,
- older aliases such as `interactionInference`, and
- the pretty x-axis labels used by `plotWmfmExplanationClaimHeatmap()`, such as `"Direction claim"`, `"Scale claim"`, or `"CI mention"`.

By default the function prints a formatted explanation to the console using `cat()` and returns the underlying character vector invisibly. This makes it convenient for interactive use while still allowing the text to be captured programmatically when needed.

For judged fields scored on a `0`/`1`/`2` scale, the usual interpretation is:

0 Poor, missing, inappropriate, or clearly problematic.

1 Partly adequate, weak, borderline, or unclear.

2 Appropriate, clear, or well handled.

Value

If `format = "text"`, a formatted character vector returned invisibly after being printed to the console. Otherwise a named list or a one-row data frame describing the requested field.

Examples

```
describeWmfmField("interactionEvidenceAppropriate")
describeWmfmField("Direction claim")
describeWmfmField("CI mention")
describeWmfmField("overallScore", format = "list")
```

---

detectImplicitComparison

Detect implicit comparison language in WMFM explanations

Description

Detects comparison structure that may not use explicit phrases such as "compared with" or "relative to", but still clearly expresses a contrast between groups or conditions.

Usage

```
detectImplicitComparison(text)
```

Arguments

`text`	Character scalar explanation text.

Details

This helper is intended to support extraction in `buildWmfmRunRecord()`, especially for explanations that use constructions such as:

- "higher ... than those who did not"
- "more ... than students who did not attend"
- "lower ... than those without ..."

The function looks for comparative wording together with at least one structural cue indicating contrast or grouping.

Value

Logical scalar. `TRUE` if implicit comparison language is detected, otherwise `FALSE`.

detectRangeExpression *Detect numeric range expressions (implicit uncertainty)*

Description

Detects expressions like "from 3.0 to 4.0" or "between 3.8 and 12.2", which indicate uncertainty without explicitly mentioning confidence intervals.

Usage

```
detectRangeExpression(text)
```

Arguments

text	Character string

Value

Logical

detectWmfmPattern *Detect pattern in text*

Description

Detect pattern in text

Usage

```
detectWmfmPattern(x, pattern)
```

Arguments

x	Character text.
pattern	Regex pattern.

Value

Logical.

diagnose *Diagnose scoring disagreement*

Description

S3 generic for diagnosing disagreement between deterministic and LLM scoring outputs.

Usage

```
diagnose(x, ...)
```

Arguments

x	An object to diagnose.
...	Additional arguments passed to methods.

Value

Method-specific diagnosis output.

diagnose.wmfmScores *Diagnose disagreement for a WMFM scores object*

Description

Diagnoses disagreement between deterministic and LLM scoring for either a single metric or all available metrics.

Usage

```
## S3 method for class 'wmfmScores'
diagnose(x, metric = NULL, runs = NULL, cmp = NULL, maxExamples = 5, ...)
```

Arguments

x	A wmfmScores object.
metric	Optional single metric name. If NULL, diagnose all comparable metrics.
runs	Optional wmfmRuns object or compatible run container used to provide explanation text and claim-level context.
cmp	Optional wmfmScoreComparison object.
maxExamples	Maximum number of flagged disagreement examples to retain for each metric diagnosis.
...	Reserved for future method-specific arguments.

Details

When `metric` is supplied, the function returns a `wmfmMetricDiagnosis` object. When `metric = NULL`, it returns a `wmfmScoresDiagnosis` object summarising all metrics that can be compared.

Value

A `wmfmMetricDiagnosis` or `wmfmScoresDiagnosis` object.

`diagnoseExplanationSurfaceProcessing`
Diagnose deterministic explanation surface processing

Description

Runs the deterministic explanation surface processor in debug mode and records surface-language issues before and after processing. This helper is intended for developer review and tests. It does not change the explanation pipeline and it does not make statistical claims.

Usage

```
diagnoseExplanationSurfaceProcessing(text, audit = NULL)
```

Arguments

`text`	Character vector of explanation text.
`audit`	Optional explanation audit object passed to `postProcessExplanationText()`.

Value

A list with original text, processed text, applied rule names, detected issues before processing, detected issues after processing, and the number of unresolved issues.

Examples

```
diagnoseExplanationSurfaceProcessing(
  "A one-unit rise multiplies the expected count by 0.21."
)
```

editWmfmConfig *Edit the WMFM local configuration file*

Description

Opens the WMFM user configuration file in the default R editor. If the configuration directory or file does not yet exist, they are created first.

Usage

```
editWmfmConfig(editor = utils::file.edit)
```

Arguments

editor	Function used to open the configuration file. Defaults to utils::file.edit(). This argument is mainly provided so tests can avoid opening an editor.

Value

Invisibly returns the path to the configuration file.

explainWmfmFieldScore *Explain why a WMFM field received its score for a specific run*

Description

Provides a human-readable explanation of how a particular field (claim, judged field, or score) was derived for a given run.

Usage

```
explainWmfmFieldScore(field, x, runIndex = 1L)
```

Arguments

field	Character. Field name or plotted label (e.g. "Overclaim", "Interaction evidence", "overallScore").
x	A wmfmRuns object, typically returned by runExample(), or a data.frame containing run records.
runIndex	Integer. Which run to explain.

Details

This function works directly with a repeated-runs object and will internally call scoreWmfmRepeatedRuns() if needed.

Value

Invisibly returns a character vector explanation. Also prints to console using cat().

---

extractWmfmText *Extract text from WMFM output objects*

Description

Converts model explanation or equation objects into plain text.

Usage

```
extractWmfmText(x)
```

Arguments

x	Object returned by WMFM functions.

Value

Character string.

---

fillMissingPredictors *Fill missing predictor columns in new data using the model's training frame*

Description

Fill missing predictor columns in new data using the model's training frame

Usage

```
fillMissingPredictors(model, newData)
```

Arguments

model	A fitted model object with a model frame (e.g., lm/glm).
newData	A data frame intended for prediction/design-matrix construction.

Value

newData with any missing predictor columns added, using values from the first row of the model frame and preserving factor levels.

findExplanationSurfaceIssues *Find remaining surface-language issues in explanation text*

Description

Scans explanation text for recurring surface-language issues that the deterministic post-processing layer is intended to control. This helper is diagnostic only: it does not rewrite the text and it does not make any statistical claims.

Usage

```
findExplanationSurfaceIssues(
  text,
 issueTypes = c("modelMechanismLanguage", "unitChangePhrasing", "verbalFractions",
    "confidenceIntervalTerminology", "longSentencePatterns")
)
```

Arguments

text	Character vector of explanation text.
issueTypes	Character vector naming issue groups to scan for. Defaults to all supported issue groups.

Value

A data frame with one row per detected issue and columns element, issueType, pattern, and matchedText.

formatSummaryValue *Format a numeric summary value for display*

Description

Format a single numeric value for display in a summary table using a significant-digits rule that aims to keep only the digits that carry useful information. Integer values are shown without decimal places. Non-integer values are rounded to a chosen number of significant digits.

Usage

```
formatSummaryValue(x, sigDigits = 3)
```

Arguments

x	A numeric scalar to format.
sigDigits	Integer. The number of significant digits to keep for non-integer values. Defaults to 3.

Details

This is useful for compact summary tables where fixed decimal places can produce distracting trailing zeros, for example showing 11 rather than 11.000.

Value

A character scalar. Returns NA_character_ when x has length 0 or is NA.

Examples

```
formatSummaryValue(11)
formatSummaryValue(52.8767)
formatSummaryValue(68.5)
formatSummaryValue(0.012345, sigDigits = 2)
```

---

formatWmfmElapsedTime *Format elapsed time for WMFM progress messages*

Description

Converts elapsed time in seconds to a compact human-readable string for console progress reporting.

Usage

```
formatWmfmElapsedTime(seconds)
```

Arguments

seconds	Numeric. Elapsed time in seconds.

Value

Character string such as "12s", "3m 08s", or "1h 02m 15s".

`generateBadExplanation` *Generate one or more deliberately bad model explanations*

Description

Generic for generating plausible but intentionally flawed explanations from a fitted model.

Usage

```
generateBadExplanation(x, ...)

## S3 method for class 'wmfmModel'
generateBadExplanation(
  x,
  explanation = NULL,
  type = "auto",
  severity = c("subtle", "moderate", "severe"),
  n = 1,
  mixTypes = FALSE,
  labelErrors = FALSE,
  provider = NULL,
  showProgress = interactive(),
  showTiming = interactive(),
  ...
)
```

Arguments

`x`	An object.
`...`	Additional arguments reserved for future use.
`explanation`	Optional character scalar giving the base good explanation. If `NULL`, `x$explanation` is used.
`type`	Character vector of bad explanation types, or `"auto"`.
`severity`	Character scalar. One of `"subtle"`, `"moderate"`, or `"severe"`.
`n`	Integer. Number of bad explanations to generate.
`mixTypes`	Logical. Should individual explanations combine multiple bad types?
`labelErrors`	Logical. Should the return value include error labels and severity metadata?
`provider`	Optional chat provider. If `NULL`, uses `getChatProvider()`.
`showProgress`	Logical. Should command-line progress messages be shown? Defaults to `interactive()`.
`showTiming`	Logical. Should command-line timing summaries be shown? Defaults to `interactive()`.

Value

Method-dependent output.

If `n = 1` and `labelErrors = FALSE`, a character scalar.

If `n > 1` and `labelErrors = FALSE`, a named character vector.

If `labelErrors = TRUE`, a named list containing explanations, error type labels, and severity labels.

Methods (by class)

- `generateBadExplanation(wmfmModel)`: Generate plausible but intentionally flawed explanations starting from a good explanation, either supplied directly or stored in a `wmfmModel` object. The generated explanations are returned in a form that can be passed directly to `grade()`.

`getFactorPredictors` *Extract factor predictors from a fitted model*

Description

Returns the names of predictor variables in a fitted model that are factors in the supplied data set.

Usage

```
getFactorPredictors(model, data)
```

Arguments

`model`	A fitted model object (e.g. `lm`, `glm`).
`data`	A data frame containing the variables used to fit the model.

Value

A character vector of factor predictor names. May be empty.

Examples

```
df = data.frame(
  y = rpois(10, 3),
  site = factor(rep(c("A", "B"), each = 5)),
  depth = rnorm(10)
)
mod = glm(y ~ site + depth, family = poisson, data = df)
getFactorPredictors(mod, df)
```

getMetricComparisonData *Extract run-level comparison data for a metric*

Description

Returns run-level paired scores for a given metric from a `wmfmScoreComparison` object.

Usage

```
getMetricComparisonData(x, metric)
```

Arguments

`x`	A `wmfmScoreComparison` object.
`metric`	Character name of the metric. Must be one of the metric names stored in `x$registry$metricName`.

Details

This accessor uses the metric registry stored on the comparison object as the source of truth for valid metric names. Because the choices are dynamic, `match.arg()` is used for validation and partial matching, but IDE tab completion will not show the possible values automatically.

Value

An object of class `metricComparisonData` with one row per run. When one of the compared methods is deterministic, the returned data frame includes a `detValue` column and a method-specific value column such as `llmValue`. When the comparison is between deterministic and LLM scoring, the output also includes `llmMinusDet` and `detMinusLlm`.

getModelEquations *Get fitted-model equations using the selected equation engine*

Description

Provides a single entry point for equation generation. Deterministic equations are now the default. The older LLM equation path remains available as an opt-in compatibility route during the transition.

Usage

```
getModelEquations(
  model,
  method = c("deterministic", "llm"),
  chat = NULL,
  digits = 2
)
```

Arguments

model	A fitted model object, typically of class lm or glm.
method	Character string giving the equation engine. Must be one of "deterministic" or "llm".
chat	Optional chat provider object. Required when method = "llm".
digits	Integer number of decimal places for displayed coefficients in the deterministic path. Defaults to 2.

Value

Either a deterministic equation table or the existing language-model equation object.

---

getWmfmClaimColorMap *Get the default WMFM claim colour map*

Description

Returns a named character vector of colours for semantic values used in WMFM raw repeated-run claim-profile heatmaps.

Usage

```
getWmfmClaimColorMap()
```

Details

The palette is designed for raw extracted claim fields and aims to keep semantically different values visually distinct.

Value

A named character vector of hexadecimal colours.

Examples

```
getWmfmClaimColorMap()
```

getWmfmConfigDir *Get the WMFM local configuration directory*

Description

Returns the directory that WMFM uses for user-specific local configuration. The path respects `options(wmfm.config_dir = ...)` when that option has been set.

Usage

```
getWmfmConfigDir()
```

Value

Character scalar path to the WMFM configuration directory.

getWmfmConfigPath *Get the WMFM local configuration file path*

Description

Returns the full path to the WMFM user configuration file. This is useful when checking or backing up a local configuration during setup and testing.

Usage

```
getWmfmConfigPath()
```

Value

Character scalar path to `config.json`.

getWmfmMetricRegistry *Return the WMFM metric registry*

Description

Defines metric metadata used by WMFM score comparison, stability summaries, and plotting. The registry is intended to be the single source of truth for which metrics are available and how they should be treated.

Usage

```
getWmfmMetricRegistry()
```

Value

A data frame containing metric metadata.

getWmfmRunsClaimProfileData *Build claim-profile heatmap data for a WMFM runs object*

Description

Extracts selected raw claim fields from a `wmfmRuns` object and reshapes them into long format for heatmap plotting.

Usage

```
getWmfmRunsClaimProfileData(
  x,
  fieldColumns = NULL,
  naLabel = "(missing)",
  prettyFieldLabels = TRUE,
  fieldOrder = c("semantic", "purity")
)
```

Arguments

`x`	A `wmfmRuns` object.
`fieldColumns`	Optional character vector of raw claim fields to include. If `NULL`, a default raw-only claim profile is used.
`naLabel`	Character label used for missing values.
`prettyFieldLabels`	Logical. Should field names be converted to more readable display labels?
`fieldOrder`	Character. One of `"semantic"` or `"purity"`.

Details

Runs are intended to be shown on rows and claim fields on columns. Runs are ordered by run purity. Fields can be ordered semantically or by field purity.

Value

A data frame with columns `runId`, `field`, `fieldLabel`, `value`, `modalValue`, `fieldPurity`, and `runPurity`.

---

getWmfmRunsClaimsData *Build extracted-claim frequency data for a WMFM runs object*

Description

Build extracted-claim frequency data for a WMFM runs object

Usage

```
getWmfmRunsClaimsData(x)
```

Arguments

x	A `wmfmRuns` object.

Value

A data frame summarising claim frequencies.

---

getWmfmRunsTextMetricsData *Build per-run text and timing metric data for a WMFM runs object*

Description

Extracts selected per-run descriptive metrics from a `wmfmRuns` object for plotting and inspection.

Usage

```
getWmfmRunsTextMetricsData(x)
```

Arguments

x	A `wmfmRuns` object.

Value

A data frame with one row per run.

grade *Grade WMFM objects*

Description

Generic for grading user-written explanations against a prepared WMFM model context.

Usage

```
grade(x, ...)
```

Arguments

x	An object to grade against.
...	Additional arguments passed to methods.

Value

A method-specific graded output.

grade.wmfmModel *Grade one or more explanations against a WMFM model*

Description

Creates either a wmfmGrade object for a single explanation or a wmfmGradeListObj when multiple explanations are supplied. By default the returned object is immediately scored using either the deterministic or LLM WMFM grading rubric.

Usage

```
## S3 method for class 'wmfmModel'
grade(
  x,
  explanation,
  modelAnswer = NULL,
  method = c("deterministic", "llm", "both"),
  autoScore = TRUE,
  score = NULL,
  scoreScale = 10,
  nLlm = 1L,
  confirmLargeLlmJob = FALSE,
  maxLlmJobsWithoutConfirmation = 20L,
  ...
)
```

Arguments

x	A wmfmModel object, typically created by runModel().
explanation	Character vector, character scalar, or list of character scalars giving the explanation(s) to grade.
modelAnswer	Optional character scalar giving a reference answer.
method	Character. One of "deterministic", "llm", or "both".
autoScore	Logical. Should the returned object be scored immediately? Defaults to TRUE.
score	Deprecated logical alias for autoScore, retained for backward compatibility. When supplied, its logical value is used for autoScore.
scoreScale	Numeric scalar giving the displayed mark scale. Defaults to 10.
nLlm	Integer. Number of repeated LLM gradings per explanation when method includes "llm".
confirmLargeLlmJob	Logical. Set to TRUE to allow large LLM grading requests.
maxLlmJobsWithoutConfirmation	Integer. Maximum number of LLM grading calls allowed without explicit confirmation.
...	Additional arguments passed to score() when autoScore = TRUE.

Value

An object of class wmfmGrade or wmfmGradeListObj.

---

isFactorOnlyModel *Check whether a fitted model has only factor predictors*

Description

Determines whether all predictors in a fitted linear or generalised linear model are factors in the supplied data set.

Usage

```
isFactorOnlyModel(model, data)
```

Arguments

model	A fitted model object (e.g. lm, glm).
data	A data frame containing the variables used to fit the model.

Details

Intercept-only models are not considered factor-only.

Value

Logical scalar. Returns TRUE if all predictors are factors, otherwise FALSE.

Examples

```
df = data.frame(
  y = rpois(20, 5),
  site = factor(rep(c("A", "B"), each = 10))
)
mod = glm(y ~ site, family = poisson, data = df)
isFactorOnlyModel(mod, df)
```

---

isFactorOnlyPredictorModel

Detect factor-only predictor models

Description

Determines whether a fitted model has *only factor predictors* in its model frame (i.e., every predictor column is a factor). This is used to switch the "Fitted equations" view into an ANOVA-style "Fitted means" view.

Usage

```
isFactorOnlyPredictorModel(m)
```

Arguments

m	A fitted model object, typically an lm or glm.

Details

The response is not checked for being a factor; in typical "fitted means" usage the response is numeric and predictors are factors. Intercept-only models (no predictors) return FALSE.

Value

A logical scalar. TRUE if all predictors in model.frame(m) are factors; otherwise FALSE.

Examples

```
df = data.frame(y = rnorm(12), A = factor(rep(letters[1:3], each = 4)))
mod = lm(y ~ A, data = df)
isFactorOnlyPredictorModel(mod)
```

listBadExplanationTypes

List supported bad explanation types

Description

Returns the currently supported error types that can be requested when generating deliberately flawed model explanations.

Usage

```
listBadExplanationTypes()
```

Value

A character vector of supported bad explanation types.

listWMFMExampleDetails

List packaged WMFM example details

Description

Returns metadata-backed details for packaged examples stored under `inst/extdata/examples`. This is intended for release-mode and developer-mode example-library displays that need more than the example names alone.

Usage

```
listWMFMExampleDetails(package = "WMFM", includeTestExamples = FALSE)
```

Arguments

`package`	A character string giving the package name containing the packaged examples.
`includeTestExamples`	Logical. Should developer-only examples marked as `developer` or `test` in the example metadata be included? Defaults to `FALSE`.

Value

A data frame with one row per available example and columns for example name, path, audience, model family, difficulty, teaching topic, and developer purpose.

Examples

```
if (interactive()) {
  listWMFMExampleDetails(package = "WMFM")
}
```

`listWMFMExamples` *List packaged WMFM examples*

Description

Lists packaged examples stored in the package's `inst/extdata/examples` directory by looking for specification files ending in `.spec.yml`.

Usage

```
listWMFMExamples(package = "WMFM", includeTestExamples = FALSE)
```

Arguments

`package`	A character string giving the package name containing the packaged examples.
`includeTestExamples`	Logical. Should developer-only examples marked as `developer` or `test` in the example metadata be included? Defaults to `FALSE`.

Details

Each example is expected to live in its own subdirectory under `extdata/examples/<name>/` and to contain a specification file named `<name>.spec.yml`. Example visibility is controlled by the optional `extdata/examples/example-metadata.yml` manifest.

Value

A character vector of available example names. If no examples are found, an empty character vector is returned.

Examples

```
if (interactive()) {
  listWMFMExamples(package = "WMFM")
}
```

makeDeveloperModePasswordHash *Create a developer-mode password hash*

Description

Creates a salted password hash suitable for storing in the WMFM_DEVELOPER_MODE_PASSWORD_HASH environment variable.

Usage

```
makeDeveloperModePasswordHash(password)
```

Arguments

password	Character scalar giving the plain-text password.

Value

A salted password hash string.

makeFactorOnlyPlot *Plot response by factor predictors for factor-only models*

Description

Produces a grouped plot for models with only factor predictors.

Usage

```
makeFactorOnlyPlot(
  model,
  data,
  ciType = c("standard", "sandwich"),
  hcType = c("HC0", "HC3")
)
```

Arguments

model	A fitted model object (e.g. lm, glm).
data	A data frame containing the variables used to fit the model.
ciType	Confidence-interval type. One of "standard" (model-based) or "sandwich" (robust).
hcType	Heteroskedasticity-consistent estimator type for robust CIs. One of "HC0" or "HC3". Only used when ciType="sandwich".

Details

- Uses boxplots when all groups have at least 10 observations
- Uses jittered point plots when any group has fewer than 10 observations
- Adds fitted means and 95% confidence intervals alongside each group
- Supports standard (model-based) and robust (sandwich) confidence intervals
- Applies a log(1 + y) scale when the model is Poisson

For multiple factor predictors, groups are defined by their interaction.

Robust confidence intervals are computed on the linear predictor scale using X V X^T, where V is either vcov(model) or sandwich::vcovHC(model, type = ...). For GLMs, intervals are then transformed back to the response scale using the inverse link function.

Value

A `ggplot` object.

`makeFittedMeansData` *Build fitted means for factor-only predictor models*

Description

Creates a full grid of factor-level combinations for all predictors in the fitted model frame and computes fitted values for each combination.

Usage

```
makeFittedMeansData(m)
```

Arguments

`m`	A fitted model object, typically an `lm` or `glm`.

Details

For `glm` models, fitted values are returned on the response scale via `predict(type = "response")`. For `lm` models, fitted values are returned from `predict()`.

Value

A list with components:

response The response variable name.

predictors Character vector of predictor names (in model frame order).

grid A data frame of predictor level combinations with an added numeric column `.fit`.

mf The model frame used for extracting factor levels.

Examples

```
df = data.frame(
  y = rnorm(12),
  A = factor(rep(c("a","b"), each = 6)),
  B = factor(rep(c("u","v","w"), times = 4))
)
mod = lm(y ~ A * B, data = df)
out = makeFittedMeansData(mod)
head(out$grid)
```

`makeMeanEquation` *Construct a fitted-mean equation from model coefficients*

Description

Builds an explicit equation for a single fitted mean using the model's coefficient vector and the corresponding design row from `model.matrix()`. This is designed to show how each fitted mean is constructed from the regression table coefficients.

Usage

```
makeMeanEquation(m, oneRowDf, label)
```

Arguments

`m`	A fitted model object, typically an `lm` or `glm`.
`oneRowDf`	A one-row data frame containing predictor values for which to construct the equation. Column names must match the model terms.
`label`	A character label to place on the left-hand side of the equation.

Details

For GLMs, the regression coefficients combine to form the linear predictor (often written η). This function shows that linear predictor and then shows the back-transformation to the mean on the response scale.

Value

A length-1 character string containing the equation (may include newline characters for multi-step GLM working).

Examples

```
df = data.frame(y = rnorm(8), A = factor(rep(c("a","b"), each = 4)))
mod = lm(y ~ A, data = df)
oneRow = data.frame(A = factor("b", levels = levels(df$A)))
makeMeanEquation(mod, oneRow, "Mean(A=b)")
```

`makeSafeEvalEnv` *Create a safe evaluation environment for derived-variable expressions*

Description

Builds an environment that contains the columns of `data` and a small allowlist of safe base functions. This is intended for evaluating the right-hand side (RHS) of user-entered derived-variable assignments in a Shiny app without exposing powerful functions like file I/O or system calls.

Usage

```
makeSafeEvalEnv(data)
```

Arguments

`data`	A data frame.

Details

The returned environment contains:

- All columns of `data` as symbols.
- A symbol `data` bound to the full data frame.
- Only the allowlisted functions (from `base`) in the parent environment.

Value

An environment suitable for `eval(rhs, envir = env)`.

Examples

```
df = data.frame(x = 1:5)
env = makeSafeEvalEnv(df)
eval(parse(text = "log(x)"), envir = env)
eval(parse(text = "factor(rep(1:2, length.out = nrow(data)))"), envir = env)
```

makeWmfmDeterministicCategoryColors *Make deterministic category colours for WMFM heatmaps*

Description

Generates a stable, perceptually stronger mapping from category values to colours using only base R / grDevices. The same category value always maps to the same colour, regardless of the order in which values appear.

Usage

```
makeWmfmDeterministicCategoryColors(values, naLabel = "(missing)")
```

Arguments

values	A vector of category values.
naLabel	Character label used for missing values after coercion.

Details

Colours are generated in HCL space using a simple deterministic string hash. This gives better separation than short fixed palettes, especially when many distinct values are present.

Value

A named character vector of colours, where names are category values.

Examples

```
makeWmfmDeterministicCategoryColors(c("yes", "no", "mixed", "(missing)"))
```

makeWmfmLegendLabels *Make readable WMFM legend labels*

Description

Converts raw semantic claim values into nicer display labels for plot legends.

Usage

```
makeWmfmLegendLabels(values)
```

Arguments

values	Character vector of raw legend values.

Value

A character vector of display labels.

Examples

```
makeWmfmLegendLabels(c("TRUE", "mixed_or_unclear", "(missing)", ""))
```

`newWmfmGrade` *Create a WMFM grade object*

Description

Creates a classed `wmfmGrade` object that stores a candidate explanation, an optional reference answer, and the model context needed for scoring.

Usage

```
newWmfmGrade(
  x,
  explanation,
  modelAnswer = NULL,
  scoreScale = 10,
  records = list(),
  scores = list(),
  feedback = list(),
  meta = list()
)
```

Arguments

`x`	A `wmfmModel` object.
`explanation`	Character scalar. The explanation being graded.
`modelAnswer`	Optional character scalar giving a reference answer.
`scoreScale`	Numeric scalar giving the displayed mark scale. Defaults to `10`.
`records`	Optional named list of run records.
`scores`	Optional named list of scored data frames.
`feedback`	Optional named list of feedback components.
`meta`	Optional named list of metadata.

Value

An object of class `wmfmGrade`.

newWmfmGradeListObj *Create a WMFM grade list object*

Description

Creates a classed `wmfmGradeListObj` object that stores multiple `wmfmGrade` objects together with batch-level metadata.

Usage

```
newWmfmGradeListObj(grades, model, inputs, meta = list())
```

Arguments

`grades`	Named list of `wmfmGrade` objects.
`model`	A `wmfmModel` object.
`inputs`	Named list describing the supplied explanations.
`meta`	Optional named list of metadata.

Value

An object of class `wmfmGradeListObj`.

newWmfmModel *Create a WMFM model object*

Description

Creates a classed `wmfmModel` object that stores a fitted model together with the additional context and generated outputs used by the WMFM command-line workflow.

Usage

```
newWmfmModel(
  model,
  formula,
  modelType,
  data,
  dataContext = NULL,
  researchQuestion = NULL,
  equations = NULL,
  explanation = NULL,
  explanationAudit = NULL,
  explanationClaimEvidenceMap = NULL,
  modelProfile = NULL,
```

```
  variableTransformations = list(),
  responseTransformationMode = "both",
  interactionTerms = character(0),
  interactionMinPValue = NA_real_,
  meta = list()
)
```

Arguments

model	A fitted model object.
formula	A model formula.
modelType	Character string giving the model family.
data	A data.frame used to fit the model.
dataContext	Optional character string giving dataset context.
researchQuestion	Optional character string giving the research question associated with the fitted model.
equations	Generated equations object, or NULL.
explanation	Generated explanation text, or NULL.
explanationAudit	Deterministic explanation-audit object, or NULL. When present, this should inherit from wmfmExplanationAudit and follow the same top-level contract produced by buildModelExplanationAudit().
explanationClaimEvidenceMap	Deterministic claim-to-evidence map, or NULL.
modelProfile	Deterministic explanation model-profile metadata, or NULL.
variableTransformations	Named list of derived-variable transformation records used by this fitted model.
responseTransformationMode	Character scalar describing how later response-scale interpretation should handle recognised response transformations. One of "both", "model", or "original".
interactionTerms	Character vector of fitted interaction-term names.
interactionMinPValue	Minimum p-value across fitted interaction terms, or NA_real_.
meta	Optional named list of metadata.

Value

An object of class wmfmModel.

newWmfmScores *Create an empty WMFM scores object*

Description

Creates a wmfmScores object aligned to a wmfmRuns object but containing no scoring results yet. This is a constructor/helper for the scoring workflow. Actual score values are added later by score().

Usage

```
newWmfmScores(x, methods = character(0))
```

Arguments

x	A wmfmRuns object.
methods	Character vector of scoring methods to reserve. Allowed values are "deterministic" and "llm".

Value

An object of class wmfmScores.

normaliseWmfmText *Normalise text for comparison*

Description

Lowercases and removes extra whitespace.

Usage

```
normaliseWmfmText(x)
```

Arguments

x	Character text.

Value

Character string.

orderWmfmLegendValues *Order WMFM legend values*

Description

Returns legend values in a semantic order rather than alphabetical order. This makes the legend easier to read by grouping related claim values together.

Usage

```
orderWmfmLegendValues(values, includeBreaks = TRUE)
```

Arguments

values	Character vector of legend values to order.
includeBreaks	Logical. Should blank spacer rows be inserted between semantic groups?

Details

If includeBreaks = TRUE, blank entries are inserted between groups so the legend displays with visual spacing.

Value

A character vector of ordered legend values. If includeBreaks = TRUE, the returned vector may contain empty strings used as spacer rows.

Examples

```
orderWmfmLegendValues(c("unclear", "TRUE", "appropriate", "(missing)"))
orderWmfmLegendValues(
  c("unclear", "TRUE", "appropriate", "(missing)"),
  includeBreaks = TRUE
)
```

parseSingleAssignment *Parse a single assignment statement from text*

Description

Parses user input and verifies it is exactly one assignment statement of the form name = expr or name <- expr. This is designed for validating a derived-variable input box in a Shiny app.

Usage

```
parseSingleAssignment(txt)
```

Arguments

txt	A length-1 character string containing R code.

Value

A list with elements:

- ok: logical, whether parsing/validation succeeded.
- msg: character message if ok = FALSE.
- name: (if ok) the variable name on the LHS.
- rhs: (if ok) the RHS expression (language object).

Examples

```
parseSingleAssignment("t = 1:10")$ok
parseSingleAssignment("1:10")$ok
parseSingleAssignment("x = log(y)")$name
```

---

plot.metricComparisonData

Plot a metric comparison data object

Description

Provides quick diagnostic plots for the run-level output returned by getMetricComparisonData().

Usage

```
## S3 method for class 'metricComparisonData'
plot(x, type = c("confusion", "runs"), ...)
```

Arguments

x	A metricComparisonData object.
type	Plot type. One of "confusion" or "runs".
...	Unused. Included for S3 compatibility.

Value

A ggplot2 object.

plot.metricComparisonSummary *Plot a metric comparison summary*

Description

Plots the relationship between disagreement and deterministic ease for a `metricComparisonSummary` object.

Usage

```
## S3 method for class 'metricComparisonSummary'
plot(
  x,
  type = c("scatter", "lollipop", "diagnostic"),
  metricType = NULL,
  labelPoints = TRUE,
  orderBy = c("disagreement", "ease"),
  disagreementThreshold = 0.5,
  easeThreshold = 0.8,
  ...
)
```

Arguments

`x`	A `metricComparisonSummary` object created by `summariseMetricComparison()`.
`type`	Plot type. One of `"scatter"`, `"lollipop"`, or `"diagnostic"`.
`metricType`	Optional metric type filter. One of `NULL`, `"ordinal"`, `"binary"`, or `"continuous"`.
`labelPoints`	Logical. Should points be labelled in the scatter-style plots?
`orderBy`	Ordering for the lollipop plot. One of `"disagreement"` or `"ease"`.
`disagreementThreshold`	Numeric threshold used by `type = "diagnostic"` to decide which metrics to label and highlight.
`easeThreshold`	Numeric threshold used by `type = "diagnostic"` to decide which metrics to label and highlight.
`...`	Unused. Included for S3 compatibility.

Value

A `ggplot2` object.

plot.wmfmRuns *Plot a WMFM runs object*

Description

Produces plots for a raw wmfmRuns object. These plots are descriptive and focus on run-to-run variation in generated outputs, extracted claims, and run metrics. Judged fields and score summaries are intentionally excluded and belong to wmfmScores objects instead.

Usage

```
## S3 method for class 'wmfmRuns'
plot(x, type = c("claims", "textMetrics", "claimProfile"), ...)
```

Arguments

x	A wmfmRuns object.
type	Character. Plot type. One of "claims", "textMetrics", or "claimProfile".
...	Passed through to lower-level plotting helpers.

Details

Supported plot types are:

"claims" Bar plot of extracted binary claim frequencies across runs.

"textMetrics" Bar plot of per-run text and timing metrics.

"claimProfile" Heatmap of raw extracted claim fields across runs.

Value

A ggplot2 object.

plot.wmfmScoreComparison *Plot a WMFM score comparison object*

Description

Provides plotting modes for a wmfmScoreComparison object.

Usage

```
## S3 method for class 'wmfmScoreComparison'
plot(x, type = c("agreement", "overall", "heatmap"), ...)
```

Arguments

x	A wmfmScoreComparison object.
type	Character. One of "agreement", "overall", or "heatmap".
...	Additional arguments passed to the underlying helper.

Details

type = "overall" A Bland-Altman plot for paired overall scores.

type = "agreement" An ordinal-agreement summary plot showing exact agreement, adjacent agreement, weighted kappa, and mean absolute difference.

type = "heatmap" A run-by-metric disagreement heatmap based on run-level comparison pairs.

Value

A ggplot object.

---

plot.wmfmScores *Plot a WMFM scores object*

Description

Provides plotting methods for a wmfmScores object returned by score(). Score heatmaps use a continuous colour scale and, by default, include only the 0 to 2 dimension scores to avoid distortion from the 0 to 100 overall score.

Usage

```
## S3 method for class 'wmfmScores'
plot(
  x,
  method = NULL,
  type = c("scores", "overall", "summary"),
  fieldColumns = NULL,
  ...
)
```

Arguments

x	A wmfmScores object.
method	Optional character. Scoring method to plot. One of "deterministic" or "llm". If omitted, an available method is chosen automatically.
type	Character. One of "scores", "overall", or "summary".
fieldColumns	Optional character vector of score columns to plot when type = "scores" or type = "summary".
...	Additional arguments for future extensibility.

Value

A ggplot2 object.

`plot.wmfmScoreStability` *Plot a WMFM score stability object*

Description

Creates a visual summary of within-method score stability.

Usage

```
## S3 method for class 'wmfmScoreStability'
plot(
  x,
  type = c("continuous", "ordinal", "binary"),
  metric = NULL,
  value = NULL,
  ...
)
```

Arguments

`x`	A `wmfmScoreStability` object.
`type`	Character. One of `"continuous"`, `"ordinal"`, or `"binary"`.
`metric`	Character or `NULL`. Optional metric filter.
`value`	Character. Quantity to plot. Allowed values depend on `type`.
`...`	Unused. Included for S3 compatibility.

Value

A ggplot2 object.

plotCiControlsUi *Plot CI controls for the Plot tab*

Description

UI controls for confidence intervals. The UI adapts to the plot type:

- `"factorOnly"`: show CI type (standard vs robust) and HC choice.
- `"continuous"`: show an optional "Show confidence intervals" checkbox; when enabled, show level, CI type, and HC choice.

Usage

```
plotCiControlsUi(
  mode = c("factorOnly", "continuous"),
  showCiInputId = "plotShowCi",
  ciLevelInputId = "plotCiLevel",
  ciTypeInputId = "plotCiType",
  hcTypeInputId = "plotHcType"
)
```

Arguments

`mode`	Either `"factorOnly"` or `"continuous"`.
`showCiInputId`	Input id for the "show confidence intervals" checkbox.
`ciLevelInputId`	Input id for the confidence level slider.
`ciTypeInputId`	Input id for CI type radio buttons.
`hcTypeInputId`	Input id for HC type dropdown.

Details

Designed to be placed in the Plot tab sidebar / controls area.

Value

A Shiny tag list.

`plotModelPlot` *Draw a student-facing model plot*

Description

Draw a student-facing model plot

Usage

```
plotModelPlot(
  model,
  plotType = c("observedFitted", "residualFitted"),
  showSmoothTrend = TRUE
)
```

Arguments

`model`	A fitted model object or `wmfmModel` object.
`plotType`	Plot type.
`showSmoothTrend`	Logical; if `TRUE`, add an automatic blue smoother for supported linear-model plots.

Value

A `ggplot` object, or `NULL` for unsupported models.

`plotWmfmExplanationClaimHeatmap`
Plot a raw-claim heatmap for WMFM runs

Description

Draws a run-by-field heatmap for a `wmfmRuns` object using raw extracted claim fields only. Rows represent runs and columns represent claim fields.

Usage

```
plotWmfmExplanationClaimHeatmap(
  x,
  fieldColumns = NULL,
  naLabel = "(missing)",
  main = "Claim profile across repeated runs",
  xlab = NULL,
  ylab = "Run ID",
```

```
  prettyFieldLabels = TRUE,
  fieldOrder = c("semantic", "purity"),
  xLabelAngle = 45,
  includeLegendBreaks = FALSE
)
```

Arguments

x	A wmfmRuns object.
fieldColumns	Optional character vector of raw claim fields to plot.
naLabel	Character label used for missing values.
main	Character plot title.
xlab	Character x-axis label.
ylab	Character y-axis label.
prettyFieldLabels	Logical. Should field names be prettified for display?
fieldOrder	Character. One of "semantic" or "purity".
xLabelAngle	Numeric rotation angle for x-axis tick labels.
includeLegendBreaks	Logical. Passed to orderWmfmLegendValues().

Value

A ggplot2 object.

---

plotWmfmScoreAgreementSummary

Plot ordinal agreement summary for WMFM score comparison

Description

Plot ordinal agreement summary for WMFM score comparison

Usage

```
plotWmfmScoreAgreementSummary(x, orderBy = c("worst", "registry"))
```

Arguments

x	A wmfmScoreComparison object
orderBy	"worst" or "registry"

Value

ggplot object

`plotWmfmScoreHeatmap` *Plot a run-level disagreement heatmap for WMFM score comparisons*

Description

Visualises run-level disagreement between two scoring methods stored in a `wmfmScoreComparison` object. The heatmap is built from `pairData` and is intended primarily for ordinal and binary metrics.

Usage

```
plotWmfmScoreHeatmap(
  x,
  includeMetricTypes = c("ordinal", "binary"),
  orderBy = c("worst", "registry", "alphabetical"),
  facetBy = c("group", "metricType"),
  prettyLabels = TRUE,
  showRunEvery = 1L,
  ...
)
```

Arguments

`x`	A `wmfmScoreComparison` object.
`includeMetricTypes`	Character vector of metric types to include. Defaults to `c("ordinal", "binary")`.
`orderBy`	Character. One of `"worst"`, `"registry"`, or `"alphabetical"`.
`facetBy`	Character. One of `"group"` or `"metricType"`.
`prettyLabels`	Logical. Should metric labels be prettified using the stored labels in `pairData`?
`showRunEvery`	Integer. Show every kth run label on the x-axis.
`...`	Unused. Included for future compatibility.

Details

Ordinal metrics are classified into disagreement classes using the absolute difference between the paired scores:

- `exact` = no difference,
- `adjacent` = difference of 1,
- `moderate` = difference of 2 or more.

Binary metrics are classified as either `exact` or `different`.

Metrics can be ordered by mean disagreement severity so that the worst-agreeing metrics appear first.

Value

A ggplot object.

postProcessExplanationText

Post-process generated explanation text before display

Description

Applies deterministic surface-level cleanup to generated explanation text. This step is deliberately conservative: it removes known formatting artifacts, standardises a small set of recurring pedagogical phrasing problems, and avoids changing numeric values or statistical meaning.

Usage

```
postProcessExplanationText(text, audit = NULL, debug = FALSE)
```

Arguments

text	Character vector of explanation text.
audit	Optional explanation audit object. Reserved for future audit-aware cleanup rules.
debug	Logical. If TRUE, return the original text, processed text, and the names of deterministic rules that changed the text.

Value

A character vector with deterministic surface cleanup applied, or a list with original, processed, and rulesApplied when debug = TRUE.

print.metricComparisonSummary

Print a metric comparison summary

Description

Print a metric comparison summary

Usage

```
## S3 method for class 'metricComparisonSummary'
print(x, ...)
```

Arguments

x	A metricComparisonSummary object.
...	Unused. Included for S3 compatibility.

Value

The input object, invisibly.

`print.summary.wmfmGrade` *Print a WMFM grade summary*

Description

Print a WMFM grade summary

Usage

```
## S3 method for class 'summary.wmfmGrade'
print(x, digits = 2, ...)
```

Arguments

`x`	A `summary.wmfmGrade` object.
`digits`	Number of digits for numeric output.
`...`	Unused.

Value

Invisibly returns `x`.

`print.summary.wmfmGradeListObj` *Print a WMFM grade list summary*

Description

Print a WMFM grade list summary

Usage

```
## S3 method for class 'summary.wmfmGradeListObj'
print(x, digits = 2, ...)
```

Arguments

`x`	A `summary.wmfmGradeListObj` object.
`digits`	Number of digits for numeric output.
`...`	Unused.

Value

Invisibly returns `x`.

`print.summary.wmfmRuns`

Print a summary.wmfmRuns object

Description

Print a summary.wmfmRuns object

Usage

```
## S3 method for class 'summary.wmfmRuns'
print(x, ...)
```

Arguments

`x`	A `summary.wmfmRuns` object.
`...`	Reserved for future extensions.

Value

`x`, invisibly.

`print.wmfmBadExplanationAudit`

Print a bad explanation grading audit

Description

Print a bad explanation grading audit

Usage

```
## S3 method for class 'wmfmBadExplanationAudit'
print(x, digits = 2, maxExamples = 10, ...)
```

Arguments

`x`	A `wmfmBadExplanationAudit` object.
`digits`	Number of digits for numeric output.
`maxExamples`	Maximum number of flagged examples to print in detail.
`...`	Unused.

Value

Invisibly returns `x`.

`print.wmfmEquationTable`

Print a deterministic equation table

Description

Prints one teaching case at a time, preserving separate scale-specific lines where available.

Usage

```
## S3 method for class 'wmfmEquationTable'
print(x, ...)
```

Arguments

`x`	A `wmfmEquationTable` object.
`...`	Unused.

Value

Invisibly returns `x`.

`print.wmfmExplanationAudit`

Print a WMFM explanation audit

Description

Print a WMFM explanation audit

Usage

```
## S3 method for class 'wmfmExplanationAudit'
print(x, ...)
```

Arguments

`x`	A `wmfmExplanationAudit` object.
`...`	Unused.

Value

Invisibly returns `x`.

print.wmfmExplanationSurfaceDiagnosis *Print an explanation surface diagnosis*

Description

Print an explanation surface diagnosis

Usage

```
## S3 method for class 'wmfmExplanationSurfaceDiagnosis'
print(x, ...)
```

Arguments

x	A wmfmExplanationSurfaceDiagnosis object.
...	Additional arguments, currently unused.

Value

The input object, invisibly.

print.wmfmGrade *Print a WMFM grade object*

Description

Print a WMFM grade object

Usage

```
## S3 method for class 'wmfmGrade'
print(
  x,
  method = NULL,
  format = c("plaintext", "html"),
  digits = 2,
  maxRows = 6,
  ...
)
```

Arguments

x	A wmfmGrade object.
method	Optional character. One of "deterministic" or "llm".
format	Character. One of "plaintext" or "html".
digits	Number of digits for numeric output.
maxRows	Maximum rows per section.
...	Unused.

Value

Invisibly returns x for plaintext and an HTML document for html.

---

print.wmfmGradeComparison *Print a wmfmGradeComparison object*

Description

Print a wmfmGradeComparison object

Usage

```
## S3 method for class 'wmfmGradeComparison'
print(x, digits = 2, maxRows = 6, ...)
```

Arguments

x	A wmfmGradeComparison object.
digits	Number of digits for numeric output.
maxRows	Maximum number of rows to print in each section.
...	Unused. Included for S3 compatibility.

Value

Invisibly returns x.

print.wmfmGradeListObj

Print a WMFM grade list object

Description

Print a WMFM grade list object

Usage

```
## S3 method for class 'wmfmGradeListObj'
print(x, digits = 2, maxExamples = 6, ...)
```

Arguments

x	A wmfmGradeListObj object.
digits	Number of digits for numeric output.
maxExamples	Maximum number of explanation marks to print.
...	Unused.

Value

Invisibly returns x.

print.wmfmMetricDiagnosis

Print a metric diagnosis object

Description

Print a metric diagnosis object

Usage

```
## S3 method for class 'wmfmMetricDiagnosis'
print(x, ...)
```

Arguments

x	A wmfmMetricDiagnosis object.
...	Unused.

Value

The input object, invisibly.

`print.wmfmScoreComparison` *Print a WMFM score comparison object*

Description

Provides a concise console summary of agreement between two WMFM scoring result sets.

Usage

```
## S3 method for class 'wmfmScoreComparison'
print(x, digits = 2, ...)
```

Arguments

`x`	A `wmfmScoreComparison` object.
`digits`	Integer. Number of digits to display.
`...`	Unused. Included for S3 compatibility.

Value

Invisibly returns `x`.

`print.wmfmScoresDiagnosis` *Print a scores diagnosis object*

Description

Print a scores diagnosis object

Usage

```
## S3 method for class 'wmfmScoresDiagnosis'
print(x, ...)
```

Arguments

`x`	A `wmfmScoresDiagnosis` object.
`...`	Unused.

Value

The input object, invisibly.

print.wmfmScoreStability *Print a WMFM score stability object*

Description

Provides a concise console summary of within-method score stability.

Usage

```
## S3 method for class 'wmfmScoreStability'
print(x, digits = 2, ...)
```

Arguments

x	A wmfmScoreStability object.
digits	Integer. Number of digits to display.
...	Unused. Included for S3 compatibility.

Value

Invisibly returns x.

readWmfmConfigPath *Read the WMFM local configuration file path*

Description

Alias for getWmfmConfigPath().

Usage

```
readWmfmConfigPath()
```

Value

Character scalar path to config.json.

rebuildWmfmRunRecords *Rebuild raw WMFM run records without rerunning the LLM*

Description

Recomputes raw extracted fields for an existing `wmfmRuns` object by re-running `buildWmfmRunRecord()` on the stored run metadata and generated text. This is useful when extraction rules change and you want to refresh the raw run records without generating new LLM outputs.

Usage

```
rebuildWmfmRunRecords(x, preserveClass = TRUE)
```

Arguments

`x`	A `wmfmRuns` object.
`preserveClass`	Logical. Should the class of `x` be preserved on the returned object? Defaults to `TRUE`.

Details

This function is intentionally limited to rebuilding raw run records. It does not rescore runs and does not compute summaries. If scoring is needed after rebuilding, call `score()` on the returned object.

Value

A rebuilt `wmfmRuns` object with refreshed `runs` records.

renderOneWayTable *Render a one-way fitted-means table as HTML tags*

Description

Creates a simple HTML table (as Shiny tag objects) for a one-factor fitted means display.

Usage

```
renderOneWayTable(df, rowVar, valueName)
```

Arguments

`df`	A data frame containing the factor column and a fitted mean column.
`rowVar`	The name of the factor column used for rows.
`valueName`	The name of the numeric fitted-mean column (e.g. `".fit"`).

Value

A Shiny tag object representing an HTML table.

Examples

```
if (requireNamespace("shiny", quietly = TRUE)) {
  df = data.frame(A = c("a","b"), .fit = c(1.2, 2.3))
  renderOneWayTable(df, "A", ".fit")
}
```

---

renderTwoWayTable *Render a two-way fitted-means table as HTML tags*

Description

Creates a simple HTML two-way table (as Shiny tag objects) where one factor defines the rows and another defines the columns. Cells are filled from `df[[valueName]]` for each `rowVar`-`colVar` combination.

Usage

```
renderTwoWayTable(df, rowVar, colVar, valueName)
```

Arguments

`df`	A data frame containing `rowVar`, `colVar`, and `valueName`.
`rowVar`	Name of the factor used for table rows.
`colVar`	Name of the factor used for table columns.
`valueName`	Name of the numeric fitted-mean column (e.g. `".fit"`).

Value

A Shiny tag object representing an HTML table.

Examples

```
if (requireNamespace("shiny", quietly = TRUE)) {
  df = data.frame(
    A = rep(c("a","b"), each = 2),
    B = rep(c("u","v"), times = 2),
    .fit = c(1, 2, 3, 4)
  )
  renderTwoWayTable(df, "A", "B", ".fit")
}
```

runExample *Run a packaged WMFM example one or more times*

Description

Loads a packaged example, fits the specified model, generates equations and a plain-language explanation, and stores each run as a separate raw run record.

Usage

```
runExample(
  name,
  package = "WMFM",
  nRuns = 1,
  printOutput = FALSE,
  pauseSeconds = 0,
  showProgress = TRUE,
  useExplanationCache = FALSE,
  interactionAlpha = 0.05,
  ...
)
```

Arguments

name	Character. Name of the packaged example.
package	Character. Package containing the example. Defaults to "WMFM".
nRuns	Integer. Number of runs to perform. Defaults to 1.
printOutput	Logical. Passed to runModel().
pauseSeconds	Numeric. Optional delay between runs.
showProgress	Logical. Should a console progress bar and timing summary be shown when work is repeated?
useExplanationCache	Logical. Passed to runModel(). Defaults to FALSE so repeated runs will usually query the LLM afresh.
interactionAlpha	Numeric. Threshold used when judging whether interaction evidence wording is appropriate in buildWmfmRunRecord().
...	Additional arguments passed to runModel().

Details

This function is a unified replacement for separate single-run and repeated-run entry points. Use nRuns = 1 for a single run.

No scoring is performed here. Scoring should be done later using score() on the returned object.

Value

An object of class `wmfmRuns` with elements:

exampleName Example name.

package Package name.

spec Parsed example specification.

dataContext Optional example context text.

researchQuestion Optional example research question text.

runs List of raw run records.

meta Metadata about the run set, including elapsed time, average per-run time, and per-run timing details.

Examples

```
if (interactive()) {
  x = runExample("Course")
  y = runExample("Course", nRuns = 20)
}
```

`runModel` *Fit a WMFM model and generate command-line outputs*

Description

Fits a model using a supplied dataset and formula, optionally attaches dataset context, and then attempts to generate fitted equations and a model explanation using the same helper functions used by the app.

Usage

```
runModel(
  data,
  formula,
  modelType = c("lm", "logistic", "poisson"),
  dataContext = NULL,
  researchQuestion = NULL,
  variableTransformations = NULL,
  responseTransformationMode = "both",
  ollamaBaseUrl = NULL,
  generateExplanation = TRUE,
  printOutput = TRUE,
  useExplanationCache = TRUE,
  equationMethod = c("deterministic", "llm")
)
```

Arguments

data	A data.frame containing the variables used in the model.
formula	A model formula, either as a formula object or a character string that can be converted to a formula.
modelType	A character string giving the model family. Must be one of "lm", "logistic", or "poisson".
dataContext	Optional character string giving additional context about the dataset, study, variables, coding, or research aim.
researchQuestion	Optional character string giving the research question the user wants the fitted model to help answer.
variableTransformations	Optional named list of derived-variable transformation records to preserve when the fitted formula uses derived variables.
responseTransformationMode	Character scalar describing how later response-scale interpretation should handle recognised response transformations. One of "both", "model", or "original".
ollamaBaseUrl	Optional character string giving the base URL for the language model service.
generateExplanation	Logical. If TRUE, attempt to contact the configured chat provider and generate model-explanation text. Set to FALSE in deterministic tests or offline workflows that only need the fitted model, equations, audit, and metadata.
printOutput	Logical. If TRUE, prints the model summary, fitted equations, and explanation to the console.
useExplanationCache	Logical. Should cached explanation text be reused when the same fitted model is encountered? Defaults to TRUE.
equationMethod	Character string giving the equation engine. Must be one of "deterministic" or "llm". Defaults to "deterministic".

Details

Supported model types are linear regression, logistic regression, and Poisson regression.

Explanation caching can be controlled via useExplanationCache. For normal usage this can remain TRUE, but when repeatedly querying the language model for the same fitted model it is often useful to set it to FALSE so that each run makes a fresh explanation request.

The returned object also includes interaction-term names and the minimum interaction-term p-value, which can be used later when evaluating whether the explanation interpreted interaction evidence appropriately.

Value

Invisibly returns an object of class wmfmModel.

runWMFMApp *Run the Model Builder app*

Description

Launches the WMFM Shiny application. The Ollama base URL used for language-model calls can be configured here.

Usage

```
runWMFMApp(ollamaBaseUrl = NULL)
```

Arguments

ollamaBaseUrl	Optional character string giving the base URL of the Ollama server, for example "http://localhost:11434". If NULL, the function uses the current value of the option "wmfm.ollama_base_url" if set, otherwise a built-in default.

Value

A shiny.appobj, invisibly.

score *Score WMFM objects*

Description

Generic for scoring WMFM objects.

Usage

```
score(x, ...)
```

Arguments

x	An object to score.
...	Additional arguments passed to methods.

Value

Method-specific scored output.

score.wmfmGrade *Score a WMFM grade object*

Description

Scores a wmfmGrade object using either the deterministic WMFM rubric or an LLM-based scoring rubric. The candidate explanation is always scored. If a reference answer is present, it is scored separately and used to enrich feedback.

Usage

```
## S3 method for class 'wmfmGrade'
score(
  x,
  method = c("deterministic", "llm"),
  preferredMinWords = 80L,
  preferredMaxWords = 220L,
  fatalFlawCap = 40,
  passThreshold = 65,
  chat = NULL,
  useCache = FALSE,
  showProgress = TRUE,
  verbose = FALSE,
  nLlm = 1L,
  ...
)
```

Arguments

x	A wmfmGrade object.
method	Character. One of "deterministic" or "llm".
preferredMinWords	Integer. Passed to deterministic scoring.
preferredMaxWords	Integer. Passed to deterministic scoring.
fatalFlawCap	Numeric. Passed to deterministic scoring.
passThreshold	Numeric. Passed to deterministic scoring.
chat	Optional chat provider object. Used for LLM scoring.
useCache	Logical. Passed to LLM scoring.
showProgress	Logical. Should progress messages be shown for LLM scoring?
verbose	Logical. Passed to LLM scoring.
nLlm	Integer. Number of repeated LLM gradings for the student explanation.
...	Additional arguments passed to the relevant scoring helper.

Value

A scored `wmfmGrade` object.

`score.wmfmGradeListObj`

Score a WMFM grade list object

Description

Scores each contained `wmfmGrade` object and records batch-level timing.

Usage

```
## S3 method for class 'wmfmGradeListObj'
score(
  x,
  method = c("deterministic", "llm", "both"),
  nLlm = 1L,
  confirmLargeLlmJob = FALSE,
  maxLlmJobsWithoutConfirmation = 20L,
  showProgress = TRUE,
  ...
)
```

Arguments

`x`	A `wmfmGradeListObj` object.
`method`	Character. One of `"deterministic"`, `"llm"`, or `"both"`.
`nLlm`	Integer. Number of repeated LLM gradings per explanation.
`confirmLargeLlmJob`	Logical. Whether to allow large requests.
`maxLlmJobsWithoutConfirmation`	Integer. Maximum number of LLM calls allowed without explicit confirmation.
`showProgress`	Logical. Should progress messages be shown?
`...`	Additional arguments passed to `score.wmfmGrade()`.

Value

A scored `wmfmGradeListObj` object.

score.wmfmRuns *Score a WMFM runs object*

Description

Scores a wmfmRuns object using deterministic scoring, LLM scoring, or both, and returns a separate wmfmScores object.

Usage

```
## S3 method for class 'wmfmRuns'
score(
  x,
  method = c("deterministic", "llm", "both"),
  chat = NULL,
  useCache = FALSE,
  showProgress = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

x	A wmfmRuns object created by runExample().
method	Character. One of "deterministic", "llm", or "both".
chat	Optional chat provider object. If omitted and LLM scoring is requested, a provider is obtained via getChatProvider().
useCache	Logical. Passed to LLM scoring helpers.
showProgress	Logical. Should progress and timing be shown for LLM scoring?
verbose	Logical. Passed to LLM scoring helpers.
...	Reserved for future method-specific arguments.

Details

This method assumes that x is a raw runs object produced by runExample(). Judged fields and aggregate scores are created during the scoring step and are stored only in the returned wmfmScores object.

Value

An object of class wmfmScores.

scoreWmfmRepeatedRuns *Score repeated WMFM explanation runs using a multidimensional rubric*

Description

Applies a rubric-based scoring framework to repeated WMFM explanation runs. The scoring structure is designed to align with the revised `buildWmfmRunRecord()` schema by separating:

Usage

```
scoreWmfmRepeatedRuns(
  runsDf,
  preferredMinWords = 80L,
  preferredMaxWords = 220L,
  penaliseDuplicates = TRUE,
  duplicatePenalty = 5,
  fatalFlawCap = 40,
  passThreshold = 65,
  factualWeight = 0.3,
  inferenceWeight = 0.25,
  completenessWeight = 0.2,
  clarityWeight = 0.15,
  calibrationWeight = 0.1
)
```

Arguments

`runsDf`	A data.frame of repeated-run outputs, or a list containing a `runsDf` element.
`preferredMinWords`	Integer. Lower bound for a preferred explanation length band.
`preferredMaxWords`	Integer. Upper bound for a preferred explanation length band.
`penaliseDuplicates`	Logical. Should exact duplicate explanations be penalised?
`duplicatePenalty`	Numeric penalty subtracted from the final `overallScore` for exact duplicates.
`fatalFlawCap`	Numeric in `0` to `100`. Maximum allowed overall score when `fatalFlawDetected` is `TRUE`.
`passThreshold`	Numeric in `0` to `100`. Threshold used to create `overallPass`.
`factualWeight`	Numeric weight for the factual dimension.
`inferenceWeight`	Numeric weight for the inference dimension.
`completenessWeight`	Numeric weight for the completeness dimension.

`clarityWeight` Numeric weight for the clarity dimension.

`calibrationWeight`
Numeric weight for the calibration dimension.

Details

1. descriptive metadata about the run,
2. extracted claim variables describing what the explanation said,
3. judged quality variables describing whether those claims were appropriate, and
4. aggregate dimension scores.

The function accepts either:

- a data.frame of run records, or
- a list containing a `runsDf` element.

Rubric dimensions:

The function scores explanations across five dimensions:

Factual score How well the explanation states the main effect, effect scale, reference-group structure, and interaction substance.

Inference score How appropriate the explanation's inferential language is, especially for uncertainty and interaction evidence.

Completeness score Whether the explanation covers the important ingredients that the model structure suggests should be present.

Clarity score Whether the explanation is reasonably clear, sensibly expressed, and not excessively short or long.

Calibration score Whether the explanation avoids overclaiming and severe underclaiming.

Judged fields created here:

This function fills or overwrites the following judged fields when they are missing or `NA`:

effectDirectionCorrect Integer in `0`, `1`, `2`.
effectScaleAppropriate Integer in `0`, `1`, `2`.
referenceGroupHandledCorrectly Integer in `0`, `1`, `2`.
interactionCoverageAdequate Integer in `0`, `1`, `2`.
interactionSubstantiveCorrect Integer in `0`, `1`, `2`.
uncertaintyHandlingAppropriate Integer in `0`, `1`, `2`.
inferentialRegisterAppropriate Integer in `0`, `1`, `2`.
mainEffectCoverageAdequate Integer in `0`, `1`, `2`.
referenceGroupCoverageAdequate Integer in `0`, `1`, `2`.
clarityAdequate Integer in `0`, `1`, `2`.
numericExpressionAdequate Integer in `0`, `1`, `2`.
comparisonStructureClear Integer in `0`, `1`, `2`.
fatalFlawDetected Logical.

Aggregate fields created here:

factualScore Numeric score on a `0` to `2` scale.

inferenceScore Numeric score on a `0` to `2` scale.

completenessScore Numeric score on a `0` to `2` scale.

clarityScore Numeric score on a `0` to `2` scale.

calibrationScore Numeric score on a `0` to `2` scale.

overallScore Numeric weighted score on a `0` to `100` scale.

overallPass Logical.

Duplicate detection is based on `normalizedExplanation` when available, otherwise on trimmed explanation text.

Fatal flaws do not force the score to zero, but they cap the final overall score using `fatalFlawCap`.

Value

A data.frame with judged quality columns and dimension scores added.

`scoreWmfmRunsWithLlm` *Score multiple WMFM runs using an LLM*

Description

Scores each raw run record in a `wmfmRuns` object using an LLM-based scorer. The returned value is a list of score records containing only judged fields and score summaries. The input run records are not modified.

Usage

```
scoreWmfmRunsWithLlm(
  runRecords,
  chat,
  useCache = FALSE,
  showProgress = TRUE,
  verbose = FALSE
)
```

Arguments

`runRecords`	List of raw WMFM run records.
`chat`	Chat provider object used for LLM scoring.
`useCache`	Logical. Whether to allow cached LLM responses.
`showProgress`	Logical. Whether to display progress and timing information during scoring.
`verbose`	Logical. Whether to print additional diagnostic information.

Details

Timing metadata is attached to the returned list as an attribute named `"timing"`.

Value

A list of score records. Each element corresponds to one input run record and contains only judged fields and score summaries. A `"timing"` attribute is attached to the returned list.

`scoreWmfmRunWithLlm` *Score a single WMFM run record using a language model*

Description

Sends a scoring request to the supplied chat provider and returns a pure score record containing judged fields and score summaries. The input run record is not modified.

Usage

```
scoreWmfmRunWithLlm(runRecord, chat, useCache = FALSE, verbose = FALSE)
```

Arguments

`runRecord`	Named list produced by `buildWmfmRunRecord()`.
`chat`	A chat provider object as returned by `getChatProvider()`.
`useCache`	Logical. Should scoring results be cached and reused for identical run records? Defaults to `FALSE`.
`verbose`	Logical. Should the raw scoring response be printed? Defaults to `FALSE`.

Details

This function expects a chat provider object returned by `getChatProvider()` and uses `chat$chat(prompt)`.

Value

Named list containing only scored fields and LLM scoring metadata.

`stability` *Assess stability of WMFM objects*

Description

Generic for assessing stability of WMFM objects.

Usage

```
stability(x, ...)
```

Arguments

x	An object to assess.
...	Additional arguments passed to methods.

Value

Method-specific stability output.

---

stability.wmfmScores *Assess stability of WMFM score results*

Description

Summarises within-method stability of scores stored in a wmfmScores object. Stability is summarised separately for binary, ordinal, and continuous score fields.

Usage

```
## S3 method for class 'wmfmScores'
stability(x, ...)
```

Arguments

x	A wmfmScores object.
...	Unused. Included for S3 compatibility.

Value

An object of class wmfmScoreStability.

---

summariseDeveloperScoringAudit *Summarise repeated developer scoring fixture stability*

Description

Builds audit-only summary tables from developer scoring fixtures exported by the developer scoring UI. This helper does not rescore explanations or change any scoring decisions; it only summarises the scores and metric values already present in the supplied fixtures.

Usage

```
summariseDeveloperScoringAudit(
  fixtures,
  metricLabels = NULL,
  unstableSpreadThreshold = 2,
  lowMarkThreshold = 4,
  metricSpreadThreshold = 1
)
```

Arguments

fixtures	Named list of developer scoring fixture payloads.
metricLabels	Character vector of metric labels to audit. When NULL, all metric labels found in the fixtures are included.
unstableSpreadThreshold	Numeric mark spread at or above which an example is flagged as unstable.
lowMarkThreshold	Numeric mark at or below which a run is flagged as a low-mark run.
metricSpreadThreshold	Numeric metric-value spread at or above which a metric is flagged as unstable within an example.

Value

A list with runSummary, exampleSummary, metricSummary, and unstableMetrics data frames.

---

summariseMetricComparison

Summarise metric-level comparison and deterministic ease

Description

Builds a metric-level summary combining cross-method disagreement with deterministic stability/ease measures.

Usage

```
summariseMetricComparison(
  scores,
  comparison,
  deterministicMethod = "deterministic",
  orderBy = NULL
)
```

Arguments

scores	A wmfmScores object.
comparison	A wmfmScoreComparison object.
deterministicMethod	Name of deterministic method.
orderBy	Optional ordering: NULL, "disagreement", or "ease".

Value

An object of class metricComparisonSummary.

---

summary.wmfmGrade *Summarise a WMFM grade object*

Description

Produces a compact summary for a wmfmGrade object. When repeated LLM grading has been used, the summary includes run-to-run mark variability and per-dimension ranges.

Usage

```
## S3 method for class 'wmfmGrade'
summary(object, method = NULL, ...)
```

Arguments

object	A wmfmGrade object.
method	Optional character. One of "deterministic" or "llm".
...	Unused.

Value

An object of class summary.wmfmGrade.

summary.wmfmGradeListObj *Summarise a WMFM grade list object*

Description

Summarise a WMFM grade list object

Usage

```
## S3 method for class 'wmfmGradeListObj'
summary(object, ...)
```

Arguments

object	A wmfmGradeListObj object.
...	Unused.

Value

An object of class summary.wmfmGradeListObj.

summary.wmfmRuns *Summarise a WMFM runs object*

Description

Produces a concise summary of a raw wmfmRuns object. The summary focuses on run-level variability, text metrics, timing, duplication, and extracted claim frequencies. Judged fields and score summaries are intentionally excluded and belong to wmfmScores objects instead.

Usage

```
## S3 method for class 'wmfmRuns'
summary(object, ...)
```

Arguments

object	A wmfmRuns object.
...	Reserved for future extensions.

Value

An object of class summary.wmfmRuns.

Index