

Package: trama.models (via r-universe)

October 10, 2026

Title Statistical Models Collection for trama

Version 0.6.9

Description Nodes for statistical models in a trama graph: linear and generalized linear models from columns or a typed formula; analysis of variance for the classical experimental designs (completely randomized, randomized blocks, latin square, factorial and split plot) with type I, II and III sums of squares; linear mixed models with lme4 and lmerTest; estimated marginal means, compact letter displays and pairwise or Dunnett comparisons with emmeans; Duncan, Waller-Duncan and Scott-Knott grouping; effect sizes and coefficient forest plots; residual diagnostics and assumption tests; and the classical hypothesis tests with the Dunn post hoc. Every p-value is shown on a logarithmic gauge with significance stars. Builds on trama.data for tables and trama.view for plots.

License MIT + file LICENSE

Encoding UTF-8

Depends R (≥ 4.1)

Imports trama ($\geq 0.5.16$), trama.data ($\geq 0.4.4$), trama.view, boot, car, emmeans, lme4, reformulas, lmerTest, nlme, MASS, agricolae, ggplot2 ($\geq 4.0.0$), generics, rlang, tibble, datasets, stats, utils

Suggests broom, testthat ($\geq 3.0.0$), png, multcomp, ScottKnott, yardstick, PRROC, Skillings.Mack, coin, statmod

Config/testthat/edition 3

Config/trama/requires trama.data, trama.view

Remotes uaipedro/trama,
trama.data=github::uaipedro/trama/collections/trama.data,
trama.view=github::uaipedro/trama/collections/trama.view

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev make libharfbuzz-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libwebp-dev libssl-dev libx11-dev zlib1g-dev

Repository <https://uaipedro.r-universe.dev>
Date/Publication 2026-10-09 16:23:25 UTC
RemoteUrl <https://github.com/uaipedro/trama>
RemoteRef HEAD
RemoteSha 1bdeb4f693df836f77e762f2642cc18a3956f8b1
RemoteSubdir collections/trama.models

Contents

tr_models_anova_dbc	4
tr_models_anova_dic	4
tr_models_anova_dql	5
tr_models_anova_factorial	5
tr_models_anova_split_plot	6
tr_models_anova_table	7
tr_models_as_fit	7
tr_models_bartlett	8
tr_models_bootstrap	9
tr_models_breusch_pagan	10
tr_models_chisq	10
tr_models_clean_name	11
tr_models_coefficients	11
tr_models_coefs.tr_models_dose	12
tr_models_coefs.tr_models_glmer	13
tr_models_coefs.tr_models_gls	13
tr_models_coefs.tr_models_nls	14
tr_models_cohen_d	14
tr_models_compare	15
tr_models_confusion	15
tr_models_contract	16
tr_models_cor_test	17
tr_models_duncan	18
tr_models_dunn	19
tr_models_effect_size	19
tr_models_effects	20
tr_models_emmeans	21
tr_models_errors	22
tr_models_evaluate	22
tr_models_example	23
tr_models_fisher_exact	24
tr_models_fit_stats	24
tr_models_friedman	25
tr_models_glm	26
tr_models_glmer	26
tr_models_gls	27
tr_models_importance_table	28

tr_models_influence	29
tr_models_kruskal	29
tr_models_levene	30
tr_models_likelihood	30
tr_models_linear_hypothesis	31
tr_models_lm	32
tr_models_lmer	32
tr_models_nls	33
tr_models_one_sample_t	34
tr_models_paired_t	34
tr_models_pairwise	35
tr_models_permutation	35
tr_models_plot_caterpillar	36
tr_models_plot_coefficients	37
tr_models_plot_diagnostics	38
tr_models_plot_means	38
tr_models_plot_regression	39
tr_models_polynomial	40
tr_models_pr_curve	40
tr_models_predict	41
tr_models_predict_cv.tr_models_nls	42
tr_models_random_effects	43
tr_models_random_test	43
tr_models_report_effects	44
tr_models_report_emm	44
tr_models_report_fit	45
tr_models_resid.tr_models_dose	45
tr_models_resid.tr_models_gls	46
tr_models_residuals	46
tr_models_rls_fn	47
tr_models_roc	47
tr_models_scott_knott	48
tr_models_select	49
tr_models_shapiro	49
tr_models_shapiro_residuals	50
tr_models_stats.tr_models_glmer	50
tr_models_stats.tr_models_gls	51
tr_models_stats.tr_models_nls	51
tr_models_t_test	51
tr_models_tidy	52
tr_models_tukey_additivity	53
tr_models_waller_duncan	54
tr_models_wilcoxon	54
trama_collection	55

tr_models_anova_dbc	<i>ANOVA de um delineamento em blocos casualizados (DBC).</i>
---------------------	---

Description

ANOVA de um delineamento em blocos casualizados (DBC).

Usage

```
tr_models_anova_dbc(dados, resposta = "", tratamento = "", bloco = "")
```

Arguments

dados	tabela.
resposta	coluna da resposta.
tratamento	coluna do tratamento (vira fator).
bloco	coluna do bloco (vira fator).

Value

objeto 'tr_models_fit'.

tr_models_anova_dic	<i>ANOVA de um delineamento inteiramente casualizado (DIC).</i>
---------------------	---

Description

ANOVA de um delineamento inteiramente casualizado (DIC).

Usage

```
tr_models_anova_dic(dados, resposta = "", tratamento = "")
```

Arguments

dados	tabela.
resposta	coluna da resposta.
tratamento	coluna do tratamento (vira fator).

Value

objeto 'tr_models_fit'.

`tr_models_anova_dql` *ANOVA de um delineamento em quadrado latino (DQL).*

Description

ANOVA de um delineamento em quadrado latino (DQL).

Usage

```
tr_models_anova_dql(  
  dados,  
  resposta = "",  
  tratamento = "",  
  linha = "",  
  coluna = ""  
)
```

Arguments

<code>dados</code>	tabela.
<code>resposta</code>	coluna da resposta.
<code>tratamento</code>	coluna do tratamento (vira fator).
<code>linha, coluna</code>	colunas das duas restrições (viram fator).

Value

objeto ‘tr_models_fit’.

`tr_models_anova_factorial` *ANOVA de um fatorial (2 ou 3 fatores), em DIC ou em blocos.*

Description

ANOVA de um fatorial (2 ou 3 fatores), em DIC ou em blocos.

Usage

```
tr_models_anova_factorial(dados, resposta = "", fatores = "", bloco = "")
```

Arguments

<code>dados</code>	tabela.
<code>resposta</code>	coluna da resposta.
<code>fatores</code>	de 2 a 3 colunas, separadas por vírgula (viram fator).
<code>bloco</code>	coluna do bloco; em branco, o fatorial é em DIC.

Value

objeto ‘tr_models_fit’.

`tr_models_anova_split_plot`

ANOVA de parcelas subdivididas em blocos.

Description

O quadro tem DOIS erros: o da parcela (erro a, $\text{bloco} \times \text{parcela}$), que testa o fator da parcela, e o de dentro (erro b), que testa a subparcela e a interação. Um ‘lm’ com um erro só testaria o fator da parcela contra o erro errado — com gl a mais e p-valor pequeno demais, que é o engano clássico.

Usage

```
tr_models_anova_split_plot(  
  dados,  
  resposta = "",  
  parcela = "",  
  subparcela = "",  
  bloco = ""  
)
```

Arguments

<code>dados</code>	tabela.
<code>resposta</code>	coluna da resposta.
<code>parcela</code>	coluna do fator da parcela.
<code>subparcela</code>	coluna do fator da subparcela.
<code>bloco</code>	coluna do bloco.

Details

Guarda três ajustes: o ‘aov’ com ‘Error()’ (o quadro), um ‘lm’ com ‘bloco:parcela’ fixo (os resíduos do erro b, para os pressupostos) e o ‘lmer’ com ‘(1 | bloco:parcela)’, que dá os mesmos F no balanceado e é o que o ‘emmeans’ sabe usar — sobre o ‘aovlist’ ele devolve erro padrão NaN.

Value

objeto ‘tr_models_fit’.

`tr_models_anova_table`*Quadro da ANOVA.*

Description

Quadro da ANOVA.

Usage

```
tr_models_anova_table(modelo, tipo_sq = "I")
```

Arguments

`modelo` objeto ‘tr_models_fit’.
`tipo_sq` “I” (sequencial), “II” ou “III”.

Value

objeto ‘tr_models_effects’.

`tr_models_as_fit`*Embrulha um ajuste feito por outra coleção como ‘models/fit’.*

Description

Para a coleção que monta o próprio ajuste (a regressão de uma série, com o tempo e o período construídos por ela) e quer que todos os leitores da models o aceitem sem reimplementar o contrato: ‘classe’ escolhe os métodos (“lm” ou “gls”), e o resto é o mesmo objeto que os blocos daqui produzem.

Usage

```
tr_models_as_fit(  
  ajuste,  
  classe,  
  rotulo,  
  formula,  
  dados,  
  resposta,  
  nota = "",  
  extra = list()  
)
```

Arguments

ajuste	o objeto 'lm' ou 'gls' já ajustado.
classe	"lm" ou "gls".
rotulo	texto do topo do card.
formula	a fórmula do ajuste.
dados	a tabela usada no ajuste.
resposta	nome da coluna resposta.
nota	aviso do ajuste, em texto (" " se não houver).
extra	lista de campos a mais que a coleção guarda no objeto (e que os leitores daqui ignoram).

Value

objeto 'tr_models_fit'.

tr_models_bartlett	<i>Bartlett: as variâncias dos grupos são iguais?</i>
--------------------	---

Description

Só sem bloco: nos resíduos de um DBC/DQL o teste não tem correção publicada para a correlação dos resíduos, e o bloco recusa apontando o Levene de O'Neill & Mathews.

Usage

```
tr_models_bartlett(modelo)
```

Arguments

modelo	objeto 'tr_models_fit'.
--------	-------------------------

Value

objeto 'tr_models_test'.

tr_models_bootstrap	<i>Bootstrap de casos de um ajuste linear.</i>
---------------------	--

Description

Reamostra as linhas com reposição (dentro de **grupo**, quando dado), reajusta e resume cada quantidade: estimativa, viés, erro-padrão, intervalos percentil e BCa (`'boot::boot.ci'`) e a fração de reamostras com o mesmo sinal.

Usage

```
tr_models_bootstrap(  
  modelo,  
  quantidade = "coeficientes",  
  especs = "",  
  grupo = "",  
  reamostras = 1999L,  
  confianca = 0.95,  
  aspecto = "16:9",  
  tema = "padrão",  
  titulo = "",  
  rotulo_x = "",  
  rotulo_y = "",  
  legenda = "direita",  
  .seed = 1L  
)
```

Arguments

modelo	um <code>'models/fit'</code> da classe <code>lm</code> .
quantidade	<code>'coeficientes'</code> , <code>'médias'</code> (marginais de <code>'especs'</code>) ou <code>'diferenças'</code> (pares de médias, a de nível posterior menos a anterior).
especs	o fator das médias; em branco, o primeiro fator do modelo.
grupo	coluna dos estratos; em branco, <code>'especs'</code> nas médias e diferenças (cada nível mantém seu <code>n</code>) e nenhum nos coeficientes.
reamostras	B.
confianca	nível dos intervalos.
.seed	semente (vem do <code>card</code>).

Value

`'list(out, tabela, distribuicao)'`.

| `tr_models_breusch_pagan` |

Breusch-Pagan (versão studentizada de Koenker).

Description

Regride o quadrado dos resíduos nos preditores do modelo: se eles explicam o tamanho do resíduo, a variância não é constante. A versão de Koenker ($n \cdot R^2$) não pressupõe normalidade, e é a padrão do ‘`lmtest::bptest`’.

Usage

```
tr_models_breusch_pagan(modelo)
```

Arguments

`modelo` objeto ‘`tr_models_fit`’.

Value

objeto ‘`tr_models_test`’.

| `tr_models_chisq` |

Qui-quadrado de independência.

Description

Qui-quadrado de independência.

Usage

```
tr_models_chisq(dados, linha = "", coluna = "", correcao = FALSE)
```

Arguments

`dados` tabela.

`linha, coluna` as duas colunas categóricas.

`correcao` correção de continuidade de Yates (só em 2×2). Desligada por padrão: torna o teste conservador (Agresti 2002).

Value

objeto ‘`tr_models_test`’.

tr_models_clean_name	<i>Nome de coluna saneado: o que vem depois de ‘prob_’ numa probabilidade.</i>
----------------------	--

Description

Nível de fator é texto livre ("Iris setosa", "não-germinou", "2"), e coluna com espaço ou hífen obriga crase em todo 'data/mutate' a jusante. Troca cada sequência de caracteres que não são letra ou dígito (Unicode: acento fica) por '_', apara as pontas, e desempata com 'make.unique' — dois níveis que saneiam igual ("a b" e "a-b") não podem virar a mesma coluna.

Usage

```
tr_models_clean_name(x)
```

Arguments

x vetor de textos (os níveis).

Value

vetor de mesmo tamanho, sem repetição.

tr_models_coefficients	<i>Coefficientes.</i>
------------------------	-----------------------

Description

Coefficientes.

Usage

```
tr_models_coefficients(  
  modelo,  
  exponenciar = FALSE,  
  escala = "unidade",  
  confianca = 0.95,  
  intervalo = "padrão"  
)
```

Arguments

modelo	objeto 'tr_models_fit'.
exponenciar	no GLM, devolver 'exp()' da estimativa e do intervalo (razão de chances na binomial, razão de taxas na poisson).
escala	"unidade" (por unidade da preditora) ou "desvio padrão" (por desvio padrão da coluna da matriz de design; t e p não mudam).
confianca	nível do intervalo; as colunas saem 'li_<nível>', 'ls_<nível>' ('li_95' no padrão).
intervalo	"padrão" (o de cada modelo: t exato no 'lm', Wald no GLM e no misto, perfilado na logística da 'multi'), "perfilado" (GLM e logística da 'multi': verossimilhança perfilada) ou "Wald".

Value

objeto 'tr_models_effects'; modelo de outra coleção pode trazer a coluna 'grupo' (a classe de cada linha numa logística multinomial).

tr_models_coefs.tr_models_dose

*Coefficientes com o erro da ANOVA: $V = QM_{res} \cdot (X'WX)^{-1}$,
t com os gl do resíduo.*

Description

É o que os programas de experimentação fazem (a variância de uma média é QM/r), e não o 'summary()' do 'lm' nas médias, cujo resíduo é o desvio da regressão.

Usage

```
## S3 method for class 'tr_models_dose'
tr_models_coefs(
  x,
  exponenciar = FALSE,
  escala = "unidade",
  confianca = 0.95,
  ...
)
```

```
tr_models_coefs.tr_models_glmer
```

Coefficientes do GLMM: z de Wald e intervalo de Wald, como no 'summary()' do 'lme4'; o perfil de verossimilhança reajusta o modelo dezenas de vezes.

Description

Coefficientes do GLMM: z de Wald e intervalo de Wald, como no 'summary()' do 'lme4'; o perfil de verossimilhança reajusta o modelo dezenas de vezes.

Usage

```
## S3 method for class 'tr_models_glmer'
tr_models_coefs(
  x,
  exponenciar = FALSE,
  escala = "unidade",
  confianca = 0.95,
  ...
)
```

```
tr_models_coefs.tr_models_gls
```

Coefficientes do GLS: t com $n - p$ gl e intervalo do 'nlme::intervals'.

Description

Coefficientes do GLS: t com $n - p$ gl e intervalo do 'nlme::intervals'.

Usage

```
## S3 method for class 'tr_models_gls'
tr_models_coefs(
  x,
  exponenciar = FALSE,
  escala = "unidade",
  confianca = 0.95,
  ...
)
```

```
tr_models_coefs.tr_models_nls
```

Coeficientes com IC de Wald (t com os gl do resíduo): o perfil de verossimilhança seria mais honesto na assimetria, mas falha justo nos ajustes difíceis, e um card que ora tem IC ora não confunde mais.

Description

Coeficientes com IC de Wald (t com os gl do resíduo): o perfil de verossimilhança seria mais honesto na assimetria, mas falha justo nos ajustes difíceis, e um card que ora tem IC ora não confunde mais.

Usage

```
## S3 method for class 'tr_models_nls'
tr_models_coefs(
  x,
  exponenciar = FALSE,
  escala = "unidade",
  confianca = 0.95,
  ...
)
```

```
tr_models_cohen_d      d de Cohen e g de Hedges entre dois grupos, com intervalo.
```

Description

$d = (\text{média1} - \text{média2}) / \text{DP combinado}$, com o DP combinado das duas amostras (o do t de Student). $g = J \cdot d$, com $J = 1 - 3 / (4(n_1 + n_2) - 9)$, que tira o viés para cima do d em amostra pequena. O intervalo é o normal de Hedges & Olkin (1985): $EP(d)^2 = (n_1 + n_2) / (n_1 n_2 + d^2 / (2(n_1 + n_2)))$; o de g é o de d vezes J. Aproximado, e bom a partir de uns 10 por grupo — o exato, pela t não central, difere pouco.

Usage

```
tr_models_cohen_d(dados, resposta = "", grupo = "", confianca = 0.95)
```

Arguments

<code>dados</code>	tabela.
<code>resposta</code>	coluna numérica.
<code>grupo</code>	coluna com dois grupos.
<code>confianca</code>	nível do intervalo.

Value

tibble com 'medida', 'estimativa', 'li', 'ls', 'n1', 'n2'.

tr_models_compare	<i>Dois modelos aninhados: o termo a mais melhora o ajuste?</i>
-------------------	---

Description

Dois modelos aninhados: o termo a mais melhora o ajuste?

Usage

```
tr_models_compare(modelo, outro)
```

Arguments

modelo, outro objetos 'tr_models_fit' da mesma família, nas mesmas linhas.

Value

objeto 'tr_models_test'.

tr_models_confusion	<i>Matriz de confusão: classe real $\times$ classe prevista.</i>
---------------------	---

Description

Matriz de confusão: classe real $\times$ classe prevista.

Usage

```
tr_models_confusion(  
  modelo = NULL,  
  dados = NULL,  
  validacao = "cruzada",  
  resposta = "",  
  predito = "previsto",  
  tabela = "matriz",  
  permitir_treino = FALSE  
)
```

Arguments

modelo	um ‘models/fit’ de classificação, ou ‘NULL’ (modo tabela).
dados	tabela: sem ‘modelo’, já com a resposta e o previsto; com ‘modelo’, a tabela a prever (precisa da resposta do modelo).
validacao	só com ‘modelo’ sem ‘dados’: “cruzada” (cada linha prevista sem ela) ou “resubstituição”.
resposta, predito	só no modo tabela: as colunas da classe real e da prevista.
tabela	“matriz” (padrão) ou “métricas” (acurácia, acurácia balanceada, kappa e precisão/revocação/F1 por classe: ‘medida’, ‘grupo’, ‘valor’).
permitir_treino	com a tabela marcada pelo ‘ml/split’ (em ‘dados’): FALSE (padrão) recusa avaliar linhas do treino (‘tr_ml_error_train_eval’); TRUE avalia, avisa e acrescenta a ‘nota’ de otimismo. Tabela sem a marca é avaliada como chega.

Value

tibble: ‘real’, uma coluna de contagem por classe prevista, ‘total’, ‘acertos’, ‘taxa_acerto’; a última linha, ‘real = “total”’, é o geral.

tr_models_contract	<i>Contrato de modelo da ‘trama.models’</i>
---------------------------	---

Description

Todo objeto que viaja no tipo ‘models/fit’ tem classe ‘c(“<específica>”, “tr_models_fit”)’ e implementa estes genéricos. O store aceita qualquer classe assim desde que [tr_models_info()] devolva uma descrição válida; os demais métodos são pedidos só por quem os usa, e o que faltar vira erro ‘tr_models_error_no_method’ nomeando a classe.

Usage

```
tr_models_info(x)

tr_models_predict_raw(x, novos, ...)

tr_models_predict_cv(x, validacao = "resubstituição")

tr_models_coefs(x, ...)

tr_models_stats(x)

tr_models_resid(x)

tr_models_importance(x)
```

```
tr_models_card(x, ctx)

tr_models_serialize(x)

tr_models_unserialize(x)

tr_models_as_table(x)
```

Arguments

x um modelo (`'inherits(x, "tr_models_fit")'`).

novos data.frame com as colunas de `'info$preditores'`.

... argumentos da classe (na models: `'intervalo'`, `'confianca'`).

validacao `"ressubstituição"` ou `"cruzada"`.

ctx o contexto do preview (`'ctx$file(ext)'`).

Details

- `'tr_models_info(x)'`: ****obrigatório****. Lista com `'tarefa'` (`"regressao"` ou `"classificacao"`), `'resposta'` (texto), `'preditores'` (texto; as colunas que `'novos'` precisa ter), `'niveis'` (texto, os níveis da resposta na ordem do modelo — obrigatório na classificação, `'NULL'` na regressão), `'n'` (linhas do ajuste), `'rotulo'` (o topo do card) e `'familia'` (texto ou `'NULL'`).

- `'tr_models_predict_raw(x, novos, ...)'`: prevê em `'novos'` (data.frame já com as colunas de `'info$preditores'`). Devolve `'list(previsto, prob, extra)'`: `'previsto'` numérico na regressão, FATOR com os níveis de `'info$niveis'` na classificação; `'prob'` matriz $n \times k$ com `'colnames = info$niveis'` (ou `'NULL'` na regressão / sem probabilidade); `'extra'` data.frame de `'n'` linhas para colunas a mais (`'li'`, `'ls'`) ou `'NULL'`. As probabilidades saem para a tabela como `'prob_<nivel>'`, com o nível saneado por `[tr_models_clean_name()]`.

- `'tr_models_predict_cv(x, validacao)'`: o mesmo formato, no TREINO: `"ressubstituição"` (prevê as linhas que ajustaram) ou `"cruzada"` (cada linha prevista por um modelo que não a viu).

- `'tr_models_coefs(x, ...)'`: um `'tr_models_effects'` (colunas `'termo'`, `'p_valor'`; `'grupo'` opcional).

- `'tr_models_stats(x)'`: tibble de UMA linha com as medidas de ajuste.

- `'tr_models_resid(x)'`: a tabela do ajuste com os resíduos (só regressão).

- `'tr_models_importance(x)'`: tibble `'termo'`, `'importancia'`, `'medida'`.

- `'tr_models_card(x, ctx)'`: o `'tr_preview()'` do card.

- `'tr_models_serialize(x)'` / `'tr_models_unserialize(x)'`: o que o store grava e o que devolve do que leu. Default: identidade (o objeto vai inteiro no RDS); existe para quem guarda ponteiro externo (xgboost).

- `'tr_models_as_table(x)'`: o adaptador `'models/fit' → 'data/table'`.

<code>tr_models_cor_test</code>	<i>Teste de correlação.</i>
---------------------------------	-----------------------------

Description

Teste de correlação.

Usage

```
tr_models_cor_test(dados, x = "", y = "", metodo = "pearson")
```

Arguments

dados tabela.

x, y as duas colunas numéricas.

metodo “”pearson””, “”spearman”” ou “”kendall””.

Value

objeto ‘tr_models_test’.

tr_models_duncan	<i>Teste de Duncan.</i>
------------------	-------------------------

Description

Teste de Duncan.

Usage

```
tr_models_duncan(modelo, tratamento = "", confianca = 0.95)
```

Arguments

modelo objeto ‘tr_models_fit’ de uma ANOVA.

tratamento fator (ou até 3, para as combinações).

confianca nível de confiança; o Duncan roda a $\alpha = 1 - \text{confianca}$.

Value

objeto ‘tr_models_emm’.

tr_models_dunn	<i>Comparações múltiplas de Dunn (1964), o post hoc do Kruskal-Wallis.</i>
----------------	--

Description

À mão, e não pelo ‘FSA’/‘dunn.test’: são dez linhas, e o pacote traria uma dependência para uma fórmula. Os postos são os da amostra INTEIRA (é o que distingue o Dunn de rodar Wilcoxon par a par): $z = (R_i - R_j) / EP$, com $EP^2 = [N(N + 1)/12 - \Sigma(t^3 - t) / (12(N - 1))] \cdot (1/n_i + 1/n_j)$, a correção de empates dentro dos colchetes. O p é bilateral, $2 \cdot P(Z > |z|)$.

Usage

```
tr_models_dunn(dados, resposta = "", grupo = "", ajuste = "holm")
```

Arguments

dados	tabela.
resposta	coluna numérica.
grupo	coluna dos grupos.
ajuste	correção dos p-valores: “holm” (padrão), “bonferroni”, “sidak” ou “nenhum”.

Value

objeto ‘tr_models_effects’, um par por linha.

tr_models_effect_size	<i>Tamanhos de efeito de cada termo da ANOVA: eta², eta² parcial e ômega².</i>
-----------------------	---

Description

Tudo sai do quadro do ‘tr_models_anova_table()’, com o erro de CADA termo: o resíduo logo abaixo dele no quadro. Na parcela subdividida isso dá o erro (a) para o fator da parcela e o (b) para a subparcela e a interação, que é o que o F de cada um usa.

Usage

```
tr_models_effect_size(modelo, tipo_sq = "I")
```

Arguments

modelo objeto ‘tr_models_fit’ de uma ANOVA (lm, delineamento, subdividida).
tipo_sq “I”, “II” ou “III”, como no quadro.

Details

- $\eta^2 = SQ / SQ \text{ total}$: a fração da variação total do termo. Soma no máximo 1 entre os termos, e diminui quando se põem mais termos no modelo. Com SQ tipo II/III as SQ não somam o total, e o denominador é a soma das SQ do quadro — a coluna ‘base_eta2’ diz qual foi. - $\eta^2 \text{ parcial} = SQ / (SQ + SQ \text{ do erro})$: o termo contra o próprio erro. - $\omega^2 = (SQ - gl \cdot QM \text{ do erro}) / (SQ \text{ total} + QM \text{ do erro})$: o η^2 corrigido do viés. Só com UM erro: na subdividida o total mistura os dois estratos, e a coluna sai NA. - $\omega^2 \text{ parcial (Olejnik \& Algina 2003, fatores manipulados)} = (SQ - gl \cdot QM \text{ do erro}) / (SQ + (N - gl) \cdot QM \text{ do erro})$, com o erro do termo: é o que vale para a subdividida.

Bloco, linha e coluna do DQL e o bloco da subdividida SAEM da tabela: são controle da casualização, e um “tamanho de efeito do bloco” ao lado do do tratamento convida a comparar o que não se compara.

Value

tibble com ‘termo’, ‘gl’, ‘eta2’, ‘eta2_parcial’, ‘omega2’, ‘omega2_parcial’, ‘erro’ e ‘base_eta2’.

tr_models_effects	<i>Um quadro de efeitos (‘models/effects’), para o ‘tr_models_coefs()’ de modelo de outra coleção.</i>
--------------------------	--

Description

Exportado porque o contrato promete que ‘tr_models_coefs()’ devolve um ‘tr_models_effects’, e a coleção irmã que implementa o método não tem como montar um sem repetir aqui os campos que o store confere.

Usage

```
tr_models_effects(
  tabela,
  titulo,
  coluna_estat = NULL,
  rodape = list(),
  nota = "",
  fonte = ""
)
```

Arguments

<code>tabela</code>	data.frame com ‘termo’ e ‘p_valor’ (e o que mais o quadro tiver).
<code>titulo</code>	o topo do card.
<code>coluna_estat</code>	o nome da coluna da estatística (‘’z’’, ‘’t’’), ou ‘NULL’.
<code>rodape</code>	lista nomeada de textos curtos.
<code>nota, fonte</code>	textos do pé do card.

<code>tr_models_emmeans</code>	<i>Médias ajustadas, com letras.</i>
--------------------------------	--------------------------------------

Description

Médias ajustadas, com letras.

Usage

```
tr_models_emmeans(
  modelo,
  especs = "",
  por = "",
  ajuste = "tukey",
  confianca = 0.95,
  escala = "resposta"
)
```

Arguments

<code>modelo</code>	objeto ‘tr_models_fit’.
<code>especs</code>	fatores das médias, separados por vírgula (‘’tratamento’’, ou ‘’A, B’’ para as combinações).
<code>por</code>	fatores de condição: as médias e as letras são feitas DENTRO de cada nível deles (o desdobramento da interação).
<code>ajuste</code>	correção das comparações que fazem as letras.
<code>confianca</code>	nível de confiança do intervalo; as letras são feitas a $\alpha = 1 - \text{confianca}$.
<code>escala</code>	‘’resposta’’ ou ‘’ligação’’ (só muda algo no GLM).

Value

objeto ‘tr_models_emm’.

tr_models_errors	<i>Classes de erro da coleção ‘models’.</i>
------------------	---

Description

Mesmo princípio de ‘trama::tr_errors()’ e das coleções irmãs, e mesmo teste de duas direções: uma tabela que um teste confere contra o código, pra não envelhecer em silêncio.

Usage

```
tr_models_errors()
```

Details

Modelo tem o seu modo de falha que sai verde: o ‘lm’ ajusta dose 50/100/150 como reta quando se queria tratamento, o Shapiro roda nos resíduos de um GLM de contagem onde normalidade nunca foi pressuposto, o ‘anova’ compara dois modelos ajustados em linhas diferentes. Nada disso erra no R. Aqui vira card vermelho, dizendo o que fazer.

Value

data.frame com ‘class’ e ‘when’.

tr_models_evaluate	<i>Métricas de previsão: o erro (regressão) ou o acerto (classificação).</i>
--------------------	--

Description

Métricas de previsão: o erro (regressão) ou o acerto (classificação).

Usage

```
tr_models_evaluate(
  modelo = NULL,
  dados = NULL,
  validacao = "cruzada",
  resposta = "",
  predito = "previsto",
  positiva = "",
  permitir_treino = FALSE
)
```

Arguments

modelo	um 'models/fit' de classificação, ou 'NULL' (modo tabela).
dados	tabela: sem 'modelo', já com a resposta e o previsto; com 'modelo', a tabela a prever (precisa da resposta do modelo).
validacao	só com 'modelo' sem 'dados': "cruzada" (cada linha prevista sem ela) ou "resubstituição".
resposta, predito	só no modo tabela: as colunas da classe real e da prevista.
positiva	classificação binária: a classe das 'sensitivity' e 'specificity'; vazio = o segundo nível.
permitir_treino	com a tabela marcada pelo 'ml/split' (em 'dados'): FALSE (padrão) recusa avaliar linhas do treino ('tr_ml_error_train_eval'); TRUE avalia, avisa e acrescenta a 'nota' de otimismo. Tabela sem a marca é avaliada como chega.

Value

tibble 'metrica', 'valor', 'n' (as linhas que contaram); na classificação também 'classe' (NA nas globais; a classe nas 'precision', 'recall' e 'f1' por classe, com o suporte em 'n').

tr_models_example	<i>Carrega um conjunto de exemplo.</i>
--------------------------	--

Description

Carrega um conjunto de exemplo.

Usage

```
tr_models_example(dataset = "PlantGrowth")
```

Arguments

dataset	nome do conjunto (ver a ajuda do nó).
----------------	---------------------------------------

Value

tibble.

`tr_models_fisher_exact`*Exato de Fisher.*

Description

Exato de Fisher.

Usage

```
tr_models_fisher_exact(dados, linha = "", coluna = "")
```

Arguments

`dados` tabela.

`linha, coluna` as duas colunas categóricas.

Value

objeto ‘tr_models_test’.

`tr_models_fit_stats` *Medidas de ajuste, em colunas FIXAS.*

Description

As mesmas colunas para todo modelo, com NA onde a medida não existe: é o que deixa um ‘data/bind_rows’ de três modelos virar a tabela de comparação.

Usage

```
tr_models_fit_stats(modelo)
```

Arguments

`modelo` objeto ‘tr_models_fit’.

Value

tibble de uma linha.

tr_models_friedman	<i>Friedman, Durbin e Skillings-Mack: tratamentos em blocos, por postos.</i>
--------------------	--

Description

A alternativa não paramétrica ao DBC com UMA observação por bloco e tratamento, e aos blocos incompletos. Os três testes respondem à mesma pergunta (algum tratamento difere, com os postos tomados dentro do bloco) e diferem só no desenho que aceitam: - "friedman": blocos completos (Friedman 1937); bloco incompleto sai inteiro, e a nota conta quantos e aponta os outros dois. - "durbin": blocos incompletos balanceados (BIB: todo bloco com $k < t$ parcelas, todo tratamento r vezes, todo par junto $\frac{r(r-1)}{2}$ vezes), estatística T_1 de Durbin (1951) na forma de Conover (1999), que corrige empates. Desenho que não é BIB recusa e aponta o Skillings-Mack. - "skillings_mack": faltantes quaisquer (Skillings & Mack 1981); bloco com uma observação só sai. Empates recebem posto médio, sem correção. - "auto" (padrão): Friedman com os blocos completos, Durbin com BIB, Skillings-Mack no resto; a nota diz qual foi.

Usage

```
tr_models_friedman(  
  dados,  
  resposta = "",  
  tratamento = "",  
  bloco = "",  
  metodo = "auto"  
)
```

Arguments

dados	tabela.
resposta	coluna numérica.
tratamento	coluna do tratamento.
bloco	coluna do bloco.
metodo	"auto", "friedman", "durbin" ou "skillings_mack".

Details

Casela repetida recusa (nenhum dos três a define).

Value

objeto 'tr_models_test'.

tr_models_glm	Modelo linear generalizado.
---------------	-----------------------------

Description

Modelo linear generalizado.

Usage

```
tr_models_glm(
  dados,
  resposta = "",
  preditores = "",
  formula = "",
  familia = "poisson"
)
```

Arguments

dados	tabela.
resposta	coluna da resposta (ignorada quando há fórmula).
preditores	colunas dos preditores, separadas por vírgula.
formula	fórmula digitada, “y ~ x1 + x2”; quando preenchida, vence.
familia	“gaussiana”, “binomial”, “poisson”, “gama”, “quasipoisson” ou “quasi-binomial”, cada uma com a ligação canônica (log na gama).

Value

objeto ‘tr_models_fit’.

tr_models_glmer	GLM misto (<i>lme4::glmer</i>): binomial ou Poisson com efeitos aleatórios.
-----------------	---

Description

Máxima verossimilhança pela aproximação de Laplace (o padrão do ‘lme4’). ‘nivel_obs’ soma um intercepto aleatório por linha, ‘(1 | .obs)’; a variância extra-binomial ou extra-Poisson vira um componente de variância (Harrison 2014), a saída do misto para a superdispersão.

Usage

```
tr_models_glmer(
  dados,
  formula = "",
  resposta = "",
  fixos = "",
  grupo = "",
  familia = "binomial",
  nivel_obs = FALSE
)
```

Arguments

dados	tabela.
formula	fórmula digitada, “ $y \sim x1 + x2$ ”; quando preenchida, vence.
resposta	coluna da resposta (ignorada quando há fórmula).
fixos	colunas dos efeitos fixos (atalho sem fórmula).
grupo	coluna do intercepto aleatório (atalho sem fórmula).
familia	“binomial” ou “poisson” (ligações canônicas).
nivel_obs	soma um efeito aleatório por observação.

Value

objeto ‘tr_models_fit’.

tr_models_gls	<i>Mínimos quadrados generalizados (‘nlme::gls’): erro correlacionado dentro do grupo e variância por nível.</i>
----------------------	--

Description

Para medidas repetidas no tempo sem supor esfericidade: a correlação entre as medidas de um mesmo grupo (animal, parcela) é AR(1), simetria composta ou não estruturada, e ‘variancia_por’ dá uma variância a cada nível (‘varIdent’). As posições no grupo vêm de ‘tempo’ (ordenado) ou, sem ele, da ordem das linhas.

Usage

```
tr_models_gls(
  dados,
  formula = "",
  correlacao = "ar1",
  grupo = "",
  tempo = "",
  variancia_por = "",
  reml = TRUE
)
```

Arguments

dados	tabela.
formula	fórmula dos efeitos fixos.
correlacao	“nenhuma”, “ar1”, “car1” (AR(1) em tempo contínuo, ‘corCAR1’, para ocasiões desigualmente espaçadas: correlação $\hat{\phi}_{t-s}$ na distância real), “simetria_composta” ou “nao_estruturada”.
grupo	coluna das unidades com medidas repetidas.
tempo	coluna da ocasião (opcional).
variancia_por	coluna cujos níveis têm variâncias próprias (opcional).
reml	REML (padrão) ou máxima verossimilhança.

Value

objeto ‘tr_models_fit’.

tr_models_importance_table
<i>Importância das preditoras, pelo contrato.</i>

Description

Cada classe diz a sua medida (coluna ‘medida’): |t| nos modelos daqui, redução de impureza ou ganho nas árvores da ‘ml’. Não se compara entre medidas.

Usage

tr_models_importance_table(modelo)

Arguments

modelo um ‘models/fit’.

Value

tibble ‘termo’, ‘importancia’, ‘medida’, do maior para o menor.

tr_models_influence	<i>Medidas de influência por observação para um ajuste linear.</i>
---------------------	--

Description

Medidas de influência por observação para um ajuste linear.

Usage

```
tr_models_influence(  
  modelo,  
  aspecto = "16:9",  
  tema = "padrão",  
  titulo = "",  
  rotulo_x = "",  
  rotulo_y = "",  
  legenda = "direita"  
)
```

Arguments

modelo objeto ‘models/fit’ de classe ‘lm’.

Value

‘tr_models_effects’ com uma linha por observação.

tr_models_kruskal	<i>Kruskal-Wallis: dois ou mais grupos.</i>
-------------------	---

Description

Kruskal-Wallis: dois ou mais grupos.

Usage

```
tr_models_kruskal(dados, resposta = "", grupo = "")
```

Arguments

dados tabela.
resposta coluna numérica.
grupo coluna dos grupos.

Value

objeto ‘tr_models_test’.

tr_models_levene	<i>Levene: as variâncias dos grupos são iguais?</i>
------------------	---

Description

No DIC e nos modelos de fórmula, ANOVA dos desvios absolutos dos resíduos em relação ao centro de cada grupo ('car::leveneTest'). Nos delineamentos com bloco (DBC, fatorial em DBC, DQL), o teste de O'Neill & Mathews (2002): ANOVA dos |resíduos| de mínimos quadrados em tratamento + bloco (+ linha e coluna) com o F corrigido; ali o centro é sempre o ajuste do modelo (média), e 'centro' não muda o resultado.

Usage

```
tr_models_levene(modelo, centro = "mediana")
```

Arguments

modelo	objeto 'tr_models_fit'.
centro	"mediana" (Brown-Forsythe, robusto; padrão) ou "média" (o Levene original). Ignorado nos delineamentos com bloco.

Value

objeto 'tr_models_test'.

tr_models_likelihood	<i>Verossimilhança e aninhamento para 'models/compare' e 'models/select'</i>
----------------------	--

Description

Opcionais. As classes daqui (lm, glm, lmer, glmer, gls, nls) são lidas por dentro; um modelo de outra coleção entra nos dois blocos implementando:

Usage

```
tr_models_loglik(x)

## Default S3 method:
tr_models_loglik(x)

tr_models_nesting(x, outro)

## Default S3 method:
tr_models_nesting(x, outro)
```

Arguments

x, outro modelos ('tr_models_fit').

Details

- 'tr_models_loglik(x)': 'list(ll, k, n)' — log-verossimilhança MÁXIMA, n^o de parâmetros estimados (variância incluída) e n^o de observações que entram na verossimilhança. É o que 'models/select' usa no AIC, AICc e BIC. - 'tr_models_nesting(x, outro)': 'x' está aninhado em 'outro'? Devolve o texto dos termos a mais (vai na hipótese nula do teste) ou aborta dizendo por que não; 'models/compare' faz então a razão de verossimilhança com 'tr_models_loglik()'.
 O default dos dois é 'NULL': "não implementa".

tr_models_linear_hypothesis

Teste F da hipótese linear geral.

Description

Teste F da hipótese linear geral.

Usage

```
tr_models_linear_hypothesis(modelo, hipoteses = "", fator = "")
```

Arguments

modelo objeto 'tr_models_fit'.
hipoteses as linhas de L, separadas por ';' ou quebra de linha, cada uma com rótulo opcional ("nome: ...").
fator coluna-fator cujas médias os contrastes combinam; em branco, as hipóteses são sobre os coeficientes.

Value

objeto 'tr_models_test'.

tr_models_lm	<i>Regressão linear.</i>
--------------	--------------------------

Description

Texto e lógico viram fator; número fica número. A decisão de tratar uma dose como fator é da pessoa, na fórmula (`'factor(dose)'`), e não do bloco — nos blocos de ANOVA é o contrário, e a ajuda dos dois diz isso.

Usage

```
tr_models_lm(dados, resposta = "", preditores = "", formula = "")
```

Arguments

<code>dados</code>	tabela.
<code>resposta</code>	coluna da resposta (ignorada quando há fórmula).
<code>preditores</code>	colunas dos preditores, separadas por vírgula.
<code>formula</code>	fórmula digitada, <code>"y ~ x1 + x2"</code> ; quando preenchida, vence.

Value

objeto `'tr_models_fit'`.

tr_models_lmer	<i>Modelo linear misto ('lme4', com os p-valores do 'lmerTest').</i>
----------------	--

Description

Pela fórmula, com os termos aleatórios na sintaxe do `'lme4'` — `'(1 | bloco)'`, `'(dias | sujeito)'` —, ou pelo atalho de colunas, que monta um intercepto aleatório por `'grupo'`. O `'lmerTest::lmer'` e não o `'lme4::lmer'`: é o mesmo ajuste, com a classe que sabe dar gl de Satterthwaite ao quadro e aos coeficientes.

Usage

```
tr_models_lmer(
  dados,
  formula = "",
  resposta = "",
  fixos = "",
  grupo = "",
  reml = TRUE
)
```

Arguments

dados	tabela.
formula	fórmula digitada, “ $y \sim x1 + x2$ ”; quando preenchida, vence.
resposta	coluna da resposta (ignorada quando há fórmula).
fixos	colunas dos efeitos fixos (atalho sem fórmula).
grupo	coluna do intercepto aleatório (atalho sem fórmula).
reml	REML (padrão) ou máxima verossimilhança.

Value

objeto ‘tr_models_fit’.

tr_models_nls	<i>Regressão não linear com modelos prontos.</i>
----------------------	--

Description

Regressão não linear com modelos prontos.

Usage

```
tr_models_nls(dados, resposta = "", preditor = "", modelo = "logístico")
```

Arguments

dados	tabela.
resposta	coluna da resposta.
preditor	a coluna numérica da preditora (uma só).
modelo	“logístico”, “Michaelis-Menten”, “exponencial assintótico”, “Gompertz” ou “linear-platô”.

Value

objeto ‘tr_models_fit’ de classe ‘tr_models_nls’.

```
tr_models_one_sample_t
```

t para uma amostra.

Description

t para uma amostra.

Usage

```
tr_models_one_sample_t(dados, variavel = "", mu = 0, alternativa = "bilateral")
```

Arguments

dados	tabela.
variavel	coluna numérica testada.
mu	valor de referência.
alternativa	“bilateral”, “menor” ou “maior” (o primeiro grupo em relação ao segundo, na ordem dos níveis).

Value

objeto ‘tr_models_test’.

```
tr_models_paired_t
```

t pareado.

Description

t pareado.

Usage

```
tr_models_paired_t(dados, antes = "", depois = "", alternativa = "bilateral")
```

Arguments

dados	tabela.
antes, depois	as duas colunas numéricas medidas na mesma unidade.
alternativa	“bilateral”, “menor” ou “maior” (o primeiro grupo em relação ao segundo, na ordem dos níveis).

Value

objeto ‘tr_models_test’.

tr_models_pairwise	<i>Comparações de médias: todos os pares, ou contra um controle (Dunnett).</i>
--------------------	--

Description

Comparações de médias: todos os pares, ou contra um controle (Dunnett).

Usage

```
tr_models_pairwise(
  medias,
  metodo = "todos os pares",
  controle = "",
  ajuste = "tukey",
  .seed = NULL
)
```

Arguments

medias	objeto ‘tr_models_emm’.
metodo	“”todos os pares”” ou “”contra controle””.
controle	nível do controle (com “”contra controle””).
ajuste	correção; “dunnett” é o padrão natural contra controle. É o Dunnett exato: ‘adjust = ”mvt”’ do emmeans, integração da t multivariada (Genz-Bretz), com erro numérico de $\sim 1e-3$ a partir de 3 contrastes.
.seed	semente do nó: torna o Dunnett (Monte Carlo quase-aleatório) reproduzível, sem mexer no estado aleatório da sessão.

Value

objeto ‘tr_models_effects’.

tr_models_permutation	<i>Testa um termo por permutação da resposta, mantendo fixa a matriz do modelo.</i>
-----------------------	---

Description

Testa um termo por permutação da resposta, mantendo fixa a matriz do modelo.

Usage

```
tr_models_permutation(
  modelo,
  termo = "",
  grupo = "",
  reamostras = 9999L,
  aspecto = "16:9",
  tema = "padrão",
  titulo = "",
  rotulo_x = "",
  rotulo_y = "",
  legenda = "direita",
  .seed = 1L
)
```

```
tr_models_plot_caterpillar
```

Gráfico de lagarta dos efeitos aleatórios.

Description

Gráfico de lagarta dos efeitos aleatórios.

Usage

```
tr_models_plot_caterpillar(
  modelo,
  grupo = "",
  intervalo = "IC",
  confianca = 0.95,
  faixa_desvio = TRUE,
  ordenar = TRUE,
  aspecto = "3:4",
  tema = "padrão",
  titulo = "",
  rotulo_x = "",
  rotulo_y = "",
  legenda = "abaixo"
)
```

Arguments

modelo	objeto ‘tr_models_fit’ de um misto (ou parcela subdividida).
grupo	fator de agrupamento; em branco, o primeiro do modelo.
intervalo	largura das barras: “IC” (com ‘confianca’), “ ± 1 EP” ou “ ± 2 EP”.
confianca	nível do IC quando ‘intervalo = “IC”’; ignorado nos EP.

faixa_desvio sombrear ± 1 desvio padrão do componente de variância.
ordenar ordenar os níveis pelo efeito.

tr_models_plot_coefficients

Gráfico de floresta dos coeficientes.

Description

Lê o ‘tr_models_coefs()’ do contrato, e não o ajuste: por isso vale para todo modelo com coeficiente — ‘lm’, ‘glm’, misto, a logística da ‘multi’ e a referência linear da ‘ml’. Substitui o ‘multi/plot_odds’, que era este mesmo gráfico só para a logística; um fluxo antigo abre aqui com ‘exponenciar’ e a escala por desvio padrão que eram o padrão de lá.

Usage

```
tr_models_plot_coefficients(  
  modelo,  
  exponenciar = FALSE,  
  escala = "unidade",  
  confianca = 0.95,  
  ordenar = "modelo",  
  aspecto = "16:9",  
  tema = "padrão",  
  titulo = "",  
  rotulo_x = "",  
  rotulo_y = "",  
  legenda = "direita"  
)
```

Arguments

modelo objeto ‘tr_models_fit’.

exponenciar no GLM, devolver ‘exp()’ da estimativa e do intervalo (razão de chances na binomial, razão de taxas na poisson).

escala “unidade” (por unidade da preditora) ou “desvio padrão” (por desvio padrão da coluna da matriz de design; t e p não mudam).

confianca nível do intervalo; as colunas saem ‘li_<nível>’, ‘ls_<nível>’ (‘li_95’ no padrão).

ordenar “modelo” (a ordem dos termos no modelo, de cima para baixo) ou “estimativa” (do maior para o menor).

Details

A cor diz se o intervalo cruza a referência (0, ou 1 exponenciado): é a leitura que o gráfico existe para facilitar. Exponenciado, o eixo é log, em que dobrar e reduzir à metade ficam à mesma distância do 1.

Value

ggplot.

`tr_models_plot_diagnostics`

Diagnóstico gráfico dos resíduos: quatro painéis.

Description

Resíduos $\times$ ajustados (linearidade, variância), Q-Q normal, escala-locção (variância) e histograma. Numa figura só, e não quatro nós, porque se leem juntos: o funil do primeiro painel e a cauda do Q-Q costumam ser a mesma coisa vista de dois lados.

Usage

```
tr_models_plot_diagnostics(
  modelo,
  aspecto = "1:1",
  tema = "padrão",
  titulo = "",
  rotulo_x = "",
  rotulo_y = "",
  legenda = "direita"
)
```

Arguments

`modelo` objeto ‘tr_models_fit’.

`tr_models_plot_means` *Médias com intervalo de confiança e letras.*

Description

Médias com intervalo de confiança e letras.

Usage

```
tr_models_plot_means(
  medias,
  letras = TRUE,
  aspecto = "16:9",
  tema = "padrão",
  titulo = "",
  rotulo_x = "",
  rotulo_y = "",
  legenda = "direita"
)
```

Arguments

<code>medias</code>	objeto ‘tr_models_emm’.
<code>letras</code>	escrever as letras acima dos intervalos.

`tr_models_plot_regression`*Gráfico de regressão: pontos, curva, equação e R^2 .*

Description

A figura das teses: na dose-resposta os pontos são as MÉDIAS das doses (a curva foi ajustada a elas) e, na parábola, a linha tracejada marca a dose de máxima (ou mínima) eficiência técnica; no não linear os pontos são as observações, e no linear-platô a linha marca o início do platô.

Usage

```
tr_models_plot_regression(  
  modelo,  
  observacoes = FALSE,  
  equacao = TRUE,  
  aspecto = "4:3",  
  tema = "padrão",  
  titulo = "",  
  rotulo_x = "",  
  rotulo_y = "",  
  legenda = "direita"  
)
```

Arguments

<code>modelo</code>	objeto ‘tr_models_fit’ de ‘models/polinomial’ ou ‘models/nls’.
<code>observacoes</code>	na dose-resposta, mostrar também as parcelas (em cinza) atrás das médias.
<code>equacao</code>	escrever a equação e o R^2 no gráfico.

Value

ggplot.

tr_models_polinomial *Regressão polinomial nos tratamentos quantitativos da ANOVA.*

Description

Regressão polinomial nos tratamentos quantitativos da ANOVA.

Usage

```
tr_models_polinomial(
  modelo,
  tratamento = "",
  grau = "automático",
  grau_max = 3L,
  confianca = 0.95
)
```

Arguments

modelo	objeto ‘tr_models_fit’ de uma ANOVA em DIC, DBC ou DQL.
tratamento	o fator de doses (os níveis têm de ser números).
grau	“automático” (o maior componente significativo) ou “1” a “5”, o grau fixado da curva.
grau_max	maior grau testado no desdobramento (1 a 5, e menor que o número de níveis); sobe sozinho até o ‘grau’ fixado.
confianca	o nível dos testes dos componentes (e do intervalo dos coeficientes): o grau automático é o maior com $p < 1 - \text{confianca}$.

Value

lista com ‘modelo’ (objeto ‘tr_models_fit’ de classe ‘tr_models_dose’) e ‘quadro’ (‘tr_models_effects’, o desdobramento).

tr_models_pr_curve *Curva precisão-revocação, com a precisão média (AP) e a área de Davis & Goadrich.*

Description

Veio da ‘ml/pr_curve’ (main) na integração 9.2, com os três modos da ‘models/roc’: modelo (validação no treino), modelo + dados novos, ou só a tabela com a probabilidade. As contas não mudaram (‘tr_models_pr_pontos’, a mesma da ‘ml’): um ponto por limiar distinto, empates num degrau; $AP = \sum \Delta R \cdot P$, sem interpolação; ‘area’ com a interpolação não linear de Davis & Goadrich (2006), integrada em forma fechada (Keilwagen, Grosse & Grau 2014). A referência do acaso é a prevalência da positiva (Saito & Rehmsmeier 2015).

Usage

```
tr_models_pr_curve(
  modelo = NULL,
  dados = NULL,
  validacao = "cruzada",
  resposta = "",
  probabilidade = "",
  positiva = "",
  permitir_treino = FALSE,
  aspecto = "16:9",
  tema = "padrão",
  titulo = "",
  rotulo_x = "",
  rotulo_y = "",
  legenda = "direita"
)
```

Arguments

modelo	um ‘models/fit’ de classificação, ou ‘NULL’ (modo tabela).
dados	tabela: sem ‘modelo’, já com a resposta e o previsto; com ‘modelo’, a tabela a prever (precisa da resposta do modelo).
validacao	só com ‘modelo’ sem ‘dados’: “cruzada” (cada linha prevista sem ela) ou “resubstituição”.
probabilidade	só no modo tabela: a coluna da probabilidade da positiva; vazio = ‘prob_<positiva>’ (ou todas as ‘prob_<nível>’ na multiclasse).
positiva	classe de interesse. Vazia: no modo tabela, a do nome da coluna ‘prob_<classe>’ (ou o segundo nível sem coluna); com modelo, o segundo nível. Com três ou mais classes é obrigatória (a curva é ela contra as outras).
permitir_treino	com a tabela marcada pelo ‘ml/split’ (em ‘dados’): FALSE (padrão) recusa avaliar linhas do treino (‘tr_ml_error_train_eval’); TRUE avalia, avisa e acrescenta a ‘nota’ de otimismo. Tabela sem a marca é avaliada como chega.

Value

ggplot; os dados trazem ‘limiar’, ‘recall’, ‘precision’, ‘ap’, ‘area’ e ‘prevalencia’.

tr_models_predict	<i>Prevê a resposta de um modelo: em dados novos, ou no próprio treino.</i>
--------------------------	---

Description

Prevê a resposta de um modelo: em dados novos, ou no próprio treino.

Usage

```
tr_models_predict(
  modelo,
  dados = NULL,
  validacao = "ressubstituição",
  intervalo = "nenhum",
  confianca = 0.95
)
```

Arguments

modelo	Um ‘models/fit’. Não a parcela subdividida — o ajuste dela é uma LISTA de modelos (‘aovlist’, um por estrato de erro), sem um único ‘predict()’ que valha; o card diz para usar o misto equivalente.
dados	A tabela nova, ou ‘NULL’: sem ela, prevê as linhas do ajuste, pela ‘validacao’.
validacao	Só sem ‘dados’: “ressubstituição” (o modelo completo prevê as linhas que o ajustaram) ou “cruzada” (cada linha prevista por um modelo ajustado sem ela). Com ‘dados’ é ignorada — uma tabela nova nunca esteve no ajuste, e já é uma previsão honesta.
intervalo	“nenhum”, “confianca” (em torno da média prevista) ou “predicao” (em torno de uma observação nova, mais largo porque soma a variância do erro) — só em ‘models/lm’, e sem ‘dados’ só por ressubstituição. O GLM não tem ‘interval’ em ‘predict.glm()’ e o misto não tem erro padrão de predição fechado — os dois recusam.
confianca	nível do intervalo (o ‘level’ de ‘predict()’); só vale com intervalo diferente de “nenhum”.

Value

‘dados’ (ou o treino) com ‘previsto’, ‘prob_<nível>’ na classificação, e ‘li’, ‘ls’ com intervalo. Colunas com esses nomes que já existiam são SUBSTITUÍDAS: prever a saída de outra previsão não cria ‘previsto...2’ ao lado da velha.

tr_models_predict_cv.tr_models_nls

A cruzada reajusta sem cada linha, partindo dos parâmetros do ajuste completo (que já estão perto); linha cujo reajuste não converge sai NA.

Description

A cruzada reajusta sem cada linha, partindo dos parâmetros do ajuste completo (que já estão perto); linha cujo reajuste não converge sai NA.

Usage

```
## S3 method for class 'tr_models_nls'
tr_models_predict_cv(x, validacao = "resubstituição")
```

```
tr_models_random_effects
```

Componentes de variância.

Description

Componentes de variância.

Usage

```
tr_models_random_effects(modelo)
```

Arguments

modelo objeto 'tr_models_fit' de um misto (ou de parcela subdividida).

Value

tibble 'grupo', 'componente', 'variancia', 'desvio_padrao', 'correlacao', 'proporcao'.

```
tr_models_random_test
```

Teste dos efeitos aleatórios: cada um sai, e o ajuste piora?

Description

Teste dos efeitos aleatórios: cada um sai, e o ajuste piora?

Usage

```
tr_models_random_test(modelo)
```

Arguments

modelo objeto 'tr_models_fit' de um misto.

Value

objeto 'tr_models_effects'.

`tr_models_report_effects`*Um quadro de efeitos no relatório exportado.*

Description

O quadro da ANOVA, dos coeficientes ou das comparações como tabela, com título, rodapé (CV, média, n) e fonte. É o ‘report’ do tipo ‘models/effects’.

Usage`tr_models_report_effects(x)`**Arguments**

`x` um quadro de efeitos (‘models/effects’).

Value

objeto ‘trama.data::tr_data_report’.

`tr_models_report_emm` *Médias ajustadas no relatório exportado.*

Description

A tabela das médias com erro-padrão, intervalo e as letras, e o ajuste com que as letras foram feitas. É o ‘report’ do tipo ‘models/emm’.

Usage`tr_models_report_emm(x)`**Arguments**

`x` médias ajustadas (‘models/emm’).

Value

objeto ‘trama.data::tr_data_report’.

`tr_models_report_fit` *Um modelo ajustado no relatório exportado.*

Description

O resumo que o card mostra: a fórmula, as medidas de ajuste que existem para o modelo (R^2 , CV, AIC...), o teste global quando há um, e o quadro de efeitos — a ANOVA nos delineamentos, os coeficientes no resto. É o ‘report’ do tipo ‘models/fit’.

Usage

```
tr_models_report_fit(x)
```

Arguments

`x` um modelo (‘models/fit’).

Value

objeto ‘trama.data::tr_data_report’.

`tr_models_resid.tr_models_dose`

A tabela da ANOVA com a curva ao lado: o resíduo é o desvio de cada parcela à curva (bloco e falta de ajuste inclusos), padronizado pelo erro da ANOVA.

Description

A tabela da ANOVA com a curva ao lado: o resíduo é o desvio de cada parcela à curva (bloco e falta de ajuste inclusos), padronizado pelo erro da ANOVA.

Usage

```
## S3 method for class 'tr_models_dose'
tr_models_resid(x)
```

```
tr_models_resid.tr_models_gls
```

GLS: resíduos normalizados (descontadas a correlação e a variância do modelo).

Description

GLS: resíduos normalizados (descontadas a correlação e a variância do modelo).

Usage

```
## S3 method for class 'tr_models_gls'
tr_models_resid(x)
```

```
tr_models_residuals
```

Resíduos ao lado da tabela.

Description

No GLM sai também ‘residuo_quantilico’, o resíduo quantílico normalizado randomizado (Dunn & Smyth 1996): $N(0, 1)$ sob o modelo certo, inclusive com contagens baixas ou resposta 0/1, onde o de desvio fica em faixas. O sorteio usa a semente do nó.

Usage

```
tr_models_residuals(modelo, .seed = 1L)
```

Arguments

<code>modelo</code>	objeto ‘tr_models_fit’.
<code>.seed</code>	semente do nó (só o resíduo quantílico do GLM sorteia).

Value

a tabela usada no ajuste com ‘ajustado’, ‘residuo’ e ‘residuo_padronizado’ à direita; no GLM, também ‘residuo_quantilico’.

tr_models_rls_fn	<i>O 'fn' de 'models/rls' — nunca roda de verdade.</i>
------------------	--

Description

Um nó 'online' só existe DENTRO de uma região ('node.R' recusa 'step' sem entrada de fluxo), e é o 'step' quem o driver chama a cada passo — 'fn' fica de fora do caminho de execução. Existe porque 'tr_node()' o exige sempre (é o nível 1 da API), e os formais espelham os PARAMS um a um, com os MESMOS defaults do spec: é o que "todo param do spec existe no fn com o mesmo default" cobra de cada nó da coleção ('test-catalogo.R'), a mesma garantia que evita spec e fn discordando em silêncio em qualquer outro nó.

Usage

```
tr_models_rls_fn(dados, resposta = "y", preditores = "", lambda = 1e+06)
```

tr_models_roc	<i>Curva ROC: sensibilidade × especificidade em todos os cortes, com a AUC.</i>
---------------	---

Description

Curva ROC: sensibilidade × especificidade em todos os cortes, com a AUC.

Usage

```
tr_models_roc(
  modelo = NULL,
  dados = NULL,
  validacao = "cruzada",
  resposta = "",
  probabilidade = "",
  positiva = "",
  confianca = 0.95,
  permitir_treino = FALSE,
  aspecto = "1:1",
  tema = "padrão",
  titulo = "",
  rotulo_x = "",
  rotulo_y = "",
  legenda = "direita"
)
```

Arguments

modelo	um ‘models/fit’ de classificação, ou ‘NULL’ (modo tabela).
dados	tabela: sem ‘modelo’, já com a resposta e o previsto; com ‘modelo’, a tabela a prever (precisa da resposta do modelo).
validacao	só com ‘modelo’ sem ‘dados’: “cruzada” (cada linha prevista sem ela) ou “resubstituição”.
probabilidade	só no modo tabela: a coluna da probabilidade da positiva; vazio = ‘prob_<positiva>’ (ou todas as ‘prob_<nível>’ na multiclasse).
positiva	a classe positiva; vazio = o segundo nível (binária) ou uma curva por classe, cada uma contra as outras (três ou mais).
permitir_treino	com a tabela marcada pelo ‘ml/split’ (em ‘dados’): FALSE (padrão) recusa avaliar linhas do treino (‘tr_ml_error_train_eval’); TRUE avalia, avisa e acrescenta a ‘nota’ de otimismo. Tabela sem a marca é avaliada como chega.

Value

ggplot.

tr_models_scott_knott

Agrupamento de Scott-Knott.

Description

Agrupamento de Scott-Knott.

Usage

```
tr_models_scott_knott(modelo, tratamento = "", confianca = 0.95)
```

Arguments

modelo	objeto ‘tr_models_fit’ de uma ANOVA.
tratamento	fator (ou até 3, para as combinações).
confianca	nível de confiança; o Duncan roda a $\alpha = 1 - \text{confianca}$.

Value

objeto ‘tr_models_emm’.

tr_models_select	<i>Seleção de modelos por critério de informação.</i>
------------------	---

Description

‘AICc = $AIC + 2k(k + 1) / (n - k - 1)$ ’ (Hurvich e Tsai, 1989); os pesos de Akaike são ‘ $\exp(-\text{delta} / 2)$ ’ normalizados a soma 1 (Burnham e Anderson, 2002, cap. 2). Com ‘ $n - k - 1 \leq 0$ ’ o AICc não existe e sai ‘NA’ (o modelo fica fora do ranking, no fim).

Usage

```
tr_models_select(modelos, criterio = "AICc")
```

Arguments

modelos	lista de objetos ‘tr_models_fit’ (a porta variádica entrega uma lista), com a mesma resposta e o mesmo número de linhas.
criterio	“AICc”, “AIC” ou “BIC”.

Value

tibble ordenado do melhor ao pior: ‘entrada’ (posição na porta), ‘modelo’, ‘formula’, ‘n’, ‘k’, ‘log_verossimilhanca’, o critério, ‘delta’, ‘peso’, ‘peso_acumulado’ e ‘razao_evidencia’ (peso do melhor / peso do modelo).

tr_models_shapiro	<i>Shapiro-Wilk numa coluna.</i>
-------------------	----------------------------------

Description

Shapiro-Wilk numa coluna.

Usage

```
tr_models_shapiro(dados, variavel = "")
```

Arguments

dados	tabela.
variavel	coluna numérica testada.

Value

objeto ‘tr_models_test’.

```
tr_models_shapiro_residuals
```

Shapiro-Wilk nos resíduos do modelo.

Description

No GLM o teste vai para os resíduos quantílicos randomizados, com o p mediano de ‘sorteios’ realizações calibrado por Monte Carlo (‘reamostras’).

Usage

```
tr_models_shapiro_residuals(
  modelo,
  sorteios = 11L,
  reamostras = 199L,
  .seed = 1L
)
```

Arguments

<code>modelo</code>	objeto ‘tr_models_fit’.
<code>sorteios</code>	GLM: quantos sorteios do resíduo quantílico; o p mediano deles é a estatística.
<code>reamostras</code>	GLM: quantas respostas simuladas do ajuste calibram o p mediano.
<code>.seed</code>	semente do nó (só o GLM sorteia).

Value

objeto ‘tr_models_test’.

```
tr_models_stats.tr_models_glmer
```

No GLMM não há R^2 de uso geral (o de Nakagawa pede a variância da distribuição na escala da ligação, que muda com a família e a ligação): as medidas são as de verossimilhança.

Description

No GLMM não há R^2 de uso geral (o de Nakagawa pede a variância da distribuição na escala da ligação, que muda com a família e a ligação): as medidas são as de verossimilhança.

Usage

```
## S3 method for class 'tr_models_glmer'
tr_models_stats(x)
```

```
tr_models_stats.tr_models_gls
```

GLS: sigma do ‘nlme’ e gl do resíduo $n - p$ (os dos coeficientes e do quadro).

Description

GLS: sigma do ‘nlme’ e gl do resíduo $n - p$ (os dos coeficientes e do quadro).

Usage

```
## S3 method for class 'tr_models_gls'
tr_models_stats(x)
```

```
tr_models_stats.tr_models_nls
```

R^2 como $1 - SQ_{res}/SQ_{total}$ (pseudo: fora do linear ele não é a fração explicada nem fica entre 0 e 1 garantido) e a raiz do erro quadrático médio.

Description

R^2 como $1 - SQ_{res}/SQ_{total}$ (pseudo: fora do linear ele não é a fração explicada nem fica entre 0 e 1 garantido) e a raiz do erro quadrático médio.

Usage

```
## S3 method for class 'tr_models_nls'
tr_models_stats(x)
```

```
tr_models_t_test
```

t de Student / Welch para duas amostras independentes.

Description

t de Student / Welch para duas amostras independentes.

Usage

```
tr_models_t_test(
  dados,
  resposta = "",
  grupo = "",
  variancias_iguais = FALSE,
  alternativa = "bilateral"
)
```

Arguments

dados	tabela.
resposta	coluna numérica.
grupo	coluna com os dois grupos.
variâncias_iguais	TRUE para o t de Student clássico; FALSE (padrão) para o de Welch, que não pressupõe variâncias iguais.
alternativa	“bilateral”, “menor” ou “maior” (o primeiro grupo em relação ao segundo, na ordem dos níveis).

Value

objeto ‘tr_models_test’.

tr_models_tidy	<i>Tabelas tidy de um modelo da trama</i>
----------------	---

Description

Métodos de [generics::tidy()], [generics::glance()] e [generics::augment()] para todo objeto do tipo ‘models/fit’ (inclusive os de outras coleções que cumprem [tr_models_contract]). Os números são os de [tr_models_coefs()], [tr_models_stats()] e [tr_models_resid()]; muda só o nome das colunas.

Usage

```
## S3 method for class 'tr_models_fit'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)

## S3 method for class 'tr_models_fit'
glance(x, ...)

## S3 method for class 'tr_models_fit'
augment(x, newdata = NULL, ...)
```

Arguments

x	um modelo (‘inherits(x, "tr_models_fit")’).
conf.int	inclui ‘conf.low’ e ‘conf.high’?
conf.level	nível do intervalo (0,5 a 0,999).
exponentiate	exponencia estimativa e intervalo (razão de chances ou de taxas; só GLM com ligação logit ou log).
...	em ‘tidy()’, repassados a [tr_models_coefs()] (‘escala’, ‘intervalo’); nos outros, ignorados. Com ‘conf.int = FALSE’ o intervalo ainda é calculado pelo leitor, só não é devolvido.
newdata	data.frame com as colunas de ‘tr_models_info(x)\$preditores’.

Details

- `'tidy()'`: `'term'`, `'estimate'`, `'std.error'`, `'statistic'`, `'p.value'` e, com `'conf.int = TRUE'`, `'conf.low'` e `'conf.high'`. Colunas extras do modelo (`'gl'`, `'grupo'`) seguem no fim, com o nome original. - `'glance()'`: uma linha; `'nobs'`, `'df.residual'`, `'r.squared'`, `'adj.r.squared'`, `'sigma'`, `'logLik'`, `'AIC'`, `'BIC'` no nome do `'broom'`, e as demais medidas (`'cv_pct'`, `'r2_marginal'`, ...) com o nome da trama. - `'augment()'`: sem `'newdata'`, a tabela do ajuste com `'fitted'`, `'resid'` e `'std.resid'` (só regressão, como `[tr_models_resid()]`). Com `'newdata'`, `'fitted'` na regressão, ou `'pred_class'` e `'prob_<nível>'` na classificação.

Value

um tibble.

```
tr_models_tukey_additivity
```

Não aditividade de Tukey: bloco e tratamento interagem?

Description

O DBC pressupõe que o efeito do tratamento é o mesmo em todo bloco. O teste de um grau de liberdade acrescenta o quadrado do ajustado ao modelo aditivo: se ele é significativo, os efeitos se multiplicam em vez de somar, e uma transformação (log, quase sempre) costuma resolver.

Usage

```
tr_models_tukey_additivity(modelo)
```

Arguments

<code>modelo</code>	objeto <code>'tr_models_fit'</code> de um delineamento com bloco (DBC, fatorial em DBC ou DQL).
---------------------	---

Value

objeto `'tr_models_test'`.

```
tr_models_waller_duncan
```

Teste de Waller-Duncan (razão K).

Usage

```
tr_models_waller_duncan(modelo, tratamento = "", k = 100L)
```

Arguments

modelo objeto ‘tr_models_fit’ de uma ANOVA.
tratamento fator (ou até 3, para as combinações).
k razão de custos dos erros tipo I e II: 100 (~5
 objeto ‘tr_models_emm’.
 Teste de Waller-Duncan (razão K).

```
tr_models_wilcoxon      Wilcoxon-Mann-Whitney para duas amostras independentes.
```

Description

Wilcoxon-Mann-Whitney para duas amostras independentes.

Usage

```
tr_models_wilcoxon(dados, resposta = "", grupo = "", alternativa = "bilateral")
```

Arguments

dados tabela.
resposta coluna numérica.
grupo coluna com os dois grupos.
alternativa “bilateral”, “menor” ou “maior” (o primeiro grupo em relação ao segundo, na ordem dos níveis).

Value

objeto ‘tr_models_test’.

<code>trama_collection</code>	<i>A coleção 'models'.</i>
-------------------------------	----------------------------

Description

Carrega DEPOIS de 'trama.data' e 'trama.view': as portas usam 'data/table' e 'view/plot', e o registro recusa porta com tipo desconhecido.

Usage

```
trama_collection()
```

Details

Os ids de categoria têm prefixo ('modelo_*') porque o registro de categorias é GLOBAL: uma categoria 'testes' aqui sobrescreveria em silêncio a de outra coleção que usasse o mesmo id.

Index

`augment.tr_models_fit`
 (*tr_models_tidy*), 52

`glance.tr_models_fit`
 (*tr_models_tidy*), 52

`tidy.tr_models_fit` (*tr_models_tidy*),
 52

`tr_models_anova_dbc`, 4
`tr_models_anova_dic`, 4
`tr_models_anova_dql`, 5
`tr_models_anova_factorial`, 5
`tr_models_anova_split_plot`, 6
`tr_models_anova_table`, 7
`tr_models_as_fit`, 7
`tr_models_as_table`
 (*tr_models_contract*), 16
`tr_models_bartlett`, 8
`tr_models_bootstrap`, 9
`tr_models_breusch_pagan`, 10
`tr_models_card` (*tr_models_contract*),
 16
`tr_models_chisq`, 10
`tr_models_clean_name`, 11
`tr_models_coefficients`, 11
`tr_models_coefs` (*tr_models_contract*),
 16
`tr_models_coefs.tr_models_dose`, 12
`tr_models_coefs.tr_models_glmer`, 13
`tr_models_coefs.tr_models_gls`, 13
`tr_models_coefs.tr_models_nls`, 14
`tr_models_cohen_d`, 14
`tr_models_compare`, 15
`tr_models_confusion`, 15
`tr_models_contract`, 16
`tr_models_cor_test`, 17
`tr_models_duncan`, 18
`tr_models_dunn`, 19
`tr_models_effect_size`, 19
`tr_models_effects`, 20
`tr_models_emmeans`, 21
`tr_models_errors`, 22
`tr_models_evaluate`, 22
`tr_models_example`, 23
`tr_models_fisher_exact`, 24
`tr_models_fit_stats`, 24
`tr_models_friedman`, 25
`tr_models_glm`, 26
`tr_models_glmer`, 26
`tr_models_gls`, 27
`tr_models_importance`
 (*tr_models_contract*), 16
`tr_models_importance_table`, 28
`tr_models_influence`, 29
`tr_models_info` (*tr_models_contract*),
 16
`tr_models_kskruskal`, 29
`tr_models_levene`, 30
`tr_models_likelihoood`, 30
`tr_models_linear_hypothesis`, 31
`tr_models_lm`, 32
`tr_models_lmer`, 32
`tr_models_loglik`
 (*tr_models_likelihoood*), 30
`tr_models_nesting`
 (*tr_models_likelihoood*), 30
`tr_models_nls`, 33
`tr_models_one_sample_t`, 34
`tr_models_paired_t`, 34
`tr_models_pairwise`, 35
`tr_models_permutation`, 35
`tr_models_plot_caterpillar`, 36
`tr_models_plot_coefficients`, 37
`tr_models_plot_diagnostics`, 38
`tr_models_plot_means`, 38
`tr_models_plot_regression`, 39
`tr_models_polinomial`, 40
`tr_models_pr_curve`, 40
`tr_models_predict`, 41

- `tr_models_predict_cv`
 - `(tr_models_contract)`, [16](#)
- `tr_models_predict_cv.tr_models_nls`, [42](#)
- `tr_models_predict_raw`
 - `(tr_models_contract)`, [16](#)
- `tr_models_random_effects`, [43](#)
- `tr_models_random_test`, [43](#)
- `tr_models_report_effects`, [44](#)
- `tr_models_report_emm`, [44](#)
- `tr_models_report_fit`, [45](#)
- `tr_models_resid (tr_models_contract)`, [16](#)
- `tr_models_resid.tr_models_dose`, [45](#)
- `tr_models_resid.tr_models_gls`, [46](#)
- `tr_models_residuals`, [46](#)
- `tr_models_ribs_fn`, [47](#)
- `tr_models_roc`, [47](#)
- `tr_models_scott_knott`, [48](#)
- `tr_models_select`, [49](#)
- `tr_models_serialize`
 - `(tr_models_contract)`, [16](#)
- `tr_models_shapiro`, [49](#)
- `tr_models_shapiro_residuals`, [50](#)
- `tr_models_stats (tr_models_contract)`, [16](#)
- `tr_models_stats.tr_models_glmer`, [50](#)
- `tr_models_stats.tr_models_gls`, [51](#)
- `tr_models_stats.tr_models_nls`, [51](#)
- `tr_models_t_test`, [51](#)
- `tr_models_tidy`, [52](#)
- `tr_models_tukey_additivity`, [53](#)
- `tr_models_unserialize`
 - `(tr_models_contract)`, [16](#)
- `tr_models_waller_duncan`, [54](#)
- `tr_models_wilcoxon`, [54](#)
- `trama_collection`, [55](#)