Package 'acR'

October 14, 2026

Type Package

Title Content Analysis in R: Integrated Qualitative (LLMs) and Quantitative Pipeline

Version 0.4.0

Description Provides an integrated pipeline for content analysis combining qualitative coding assisted by large language models (LLMs) with classical quantitative text analysis. Includes modules for pre-processing (tokenization, stopwords for Brazilian Portuguese), descriptive statistics, keyness, co-occurrence networks, word clouds (including comparative and X-ray variants), sentiment analysis via OpLexicon, Latent Dirichlet Allocation (LDA), inter-coder reliability metrics (Krippendorff, Gwet), and modern visualizations based on ggplot2. Special focus on Brazilian corpora and political-institutional codebooks. Inspired by Maerz and Benoit (2025) <https://quallmer.github.io/quallmer/>.

License MIT + file LICENSE

URL https://github.com/andersonheri/acR, https://ahenriquecp.com/acR/

BugReports https://github.com/andersonheri/acR/issues

Depends R (>= 4.1)

Imports cli, dplyr, httr2, irr, jsonlite, purrr, rlang, stats, stringr, tibble, tidyr, tidyselect, utils, withr, yaml

Suggests callr, cluster, covr, ellmer, ggplot2, ggraph, ggwordcloud, igraph, ipeaplot, knitr, shiny, openxlsx, pkgdown, quanteda, readtext, rmarkdown, scales, senatebR, spelling, stopwords, stringi, tesseract, testthat, tidytext, topicmodels, wordcloud, writexl

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Anderson Henrique [aut, cre] (ORCID:
<https://orcid.org/0000-0002-1842-2725>)

Maintainer Anderson Henrique <anderson.henrique@usp.br>

Repository CRAN

Date/Publication 2026-10-14 19:22:07 UTC

Contents

ac_calibrate_tokens . . . 3
ac_chat_local . . . 4
ac_clean . . . 5
ac_clean_stopwords . . . 8
ac_cluster_documents . . . 9
ac_cooccurrence . . . 11
ac_corpus . . . 12
ac_count . . . 14
ac_estimate_tokens . . . 15
ac_export . . . 16
ac_fetch_camara . . . 18
ac_fetch_senado . . . 21
ac_import . . . 23
ac_keyness . . . 25
ac_lda . . . 27
ac_lda_tune . . . 29
ac_palette . . . 30
ac_plot_cluster . . . 31
ac_plot_cooccurrence . . . 32
ac_plot_keyness . . . 33
ac_plot_lda_topics . . . 35
ac_plot_lda_tune . . . 36
ac_plot_sentiment . . . 37
ac_plot_tf_idf . . . 38
ac_plot_top_terms . . . 39
ac_plot_wordcloud_comparative . . . 41
ac_plot_xray . . . 42
ac_qual_code . . . 43
ac_qual_codebook . . . 47
ac_qual_codebook_add . . . 50
ac_qual_codebook_history . . . 51
ac_qual_codebook_hybrid . . . 52
ac_qual_codebook_merge . . . 53
ac_qual_codebook_remove . . . 54
ac_qual_codebook_translate . . . 55
ac_qual_export_for_review . . . 56
ac_qual_import_human . . . 57

ac_qual_irr 58
ac_qual_list_models 61
ac_qual_load_codebook 62
ac_qual_recommend_model 63
ac_qual_reliability 65
ac_qual_report 67
ac_qual_report_full 68
ac_qual_sample 70
ac_qual_save_codebook 72
ac_qual_search_literature 73
ac_sentiment 75
ac_tf_idf 76
ac_tokenize 78
ac_top_terms 80
ac_wordcloud 81
as_prompt 83
is_ac_corpus 84
theme_ac 85

Index **86**

`ac_calibrate_tokens` *Calibrar a razao caracteres/token contra um modelo real*

Description

`ac_calibrate_tokens()` mede, contra um `Chat` real, quantos tokens de entrada um conjunto de textos de amostra realmente consome, e devolve a razao caracteres/token observada – para usar como `chars_per_token` em `ac_estimate_tokens()` (ou como `chars_per_token` em `ac_qual_code()` / futuras funcoes que dependam da estimativa).

Faz uma chamada real ao modelo por texto de amostra (com uma mensagem minima, para medir so o consumo de tokens de entrada), entao usa poucos textos – o suficiente para calibrar, nao uma amostra representativa de todo o corpus.

Usage

```
ac_calibrate_tokens(chat_factory, textos)
```

Arguments

`chat_factory`	Funcao `function(temperature, system_prompt) Chat` que constroi o chat a usar (o mesmo padrao aceito por `model` em `ac_qual_code()`). Permite calibrar contra o provedor/modelo exato que sera usado na rodada real.
`textos`	Vetor de caracteres com textos de amostra (poucos bastam; ideal variar o tamanho).

Value

Numero: a mediana de caracteres por token observada na amostra.

Examples

```
## Not run:
chat_factory <- function(temperature, system_prompt) {
  ellmer::chat_openai(model = "gpt-4.1-mini", temperature = temperature,
                      system_prompt = system_prompt, echo = "none")
}
razao <- ac_calibrate_tokens(chat_factory, c("texto curto de teste.",
                                             "um texto um pouco mais longo de teste."))
coded <- ac_qual_code(corpus, cb, model = chat_factory)

## End(Not run)
```

`ac_chat_local` *Conectar a um servidor LLM local compativel com OpenAI*

Description

`ac_chat_local()` cria um `Chat` do `ellmer` apontando para um servidor local compativel com a API da OpenAI – LM Studio, llama.cpp (`llama-server`), vLLM, Ollama em modo OpenAI, etc. – sem exigir uma chave de API real.

Classificar com um modelo local (via `ac_qual_code()`) elimina o custo por token e mantem os documentos fora de qualquer rede externa, ao preco de gerencia propria do servidor: tamanho de contexto, tempo de resposta e throughput passam a depender do hardware local. Use junto com `checkpoint_dir` e `max_retries` em `ac_qual_code()` para rodadas longas, e `ac_estimate_tokens()` para evitar estouro de contexto.

Usage

```
ac_chat_local(
  model,
  base_url = "http://localhost:1234/v1",
  api = c("chat_completions", "responses"),
  temperature = NULL,
  system_prompt = NULL,
  ...
)
```

Arguments

`model`	Nome/identificador do modelo como o servidor local o reconhece (ex.: o nome exibido pelo LM Studio, ou o `--model` passado ao `llama-server`).
`base_url`	URL base do servidor OpenAI-compatible. Padrao: `"http://localhost:1234/v1"` (porta padrao do LM Studio).
`api`	Qual API o servidor expoe: `"chat_completions"` (padrao – a maioria dos servidores locais: LM Studio, llama.cpp, vLLM) ou `"responses"` (servidores que implementam a API Responses da OpenAI).

temperature	Temperatura padrao do chat. NULL (padrao): usa o padrao do servidor.
system_prompt	Prompt de sistema inicial. NULL (padrao).
...	Argumentos adicionais passados a ellmer::chat_openai() ou ellmer::chat_openai_compatible().

Value

Objeto Chat do ellmer, pronto para uso em ac_qual_code() (via chat = ac_chat_local(...)).

Examples

```
## Not run:
# LM Studio rodando localmente, servindo um modelo na porta padrao
chat_local <- ac_chat_local(model = "qwen2.5-14b-instruct")

cb <- ac_qual_codebook(
  name = "tom", instructions = "Classifique o tom.",
  categories = list(
    positivo = list(definition = "Tom propositivo."),
    negativo = list(definition = "Tom critico.")
  )
)
df <- data.frame(id = "d1", texto = "Proponho cooperacao.")
corpus <- ac_corpus(df, text = texto, docid = id)
coded <- ac_qual_code(corpus, cb, chat = chat_local,
                      checkpoint_dir = "checkpoints/")

## End(Not run)
```

---

ac_clean *Limpar e normalizar texto de um corpus*

Description

ac_clean() aplica um conjunto configurável de transformações ao texto de um objeto ac_corpus, retornando um novo corpus com o texto modificado. As transformações são aplicadas em ordem lógica e registradas como atributo para auditoria.

Usage

```
ac_clean(
  corpus,
  lower = TRUE,
  remove_punct = TRUE,
  remove_numbers = FALSE,
  remove_url = TRUE,
  remove_email = TRUE,
  remove_symbols = FALSE,
```

```
  remove_hashtags = FALSE,
  remove_mentions = FALSE,
  remove_stopwords = NULL,
  remove_accents = FALSE,
  normalize_pt = FALSE,
  protect = NULL,
  extra_stopwords = NULL,
  min_char = NULL,
  custom_replacements = NULL,
  handle_na = c("preserve", "empty", "remove"),
  strip_whitespace = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

corpus	Objeto de classe ac_corpus, criado por ac_corpus().
lower	Se TRUE (padrão), converte texto para minúsculas.
remove_punct	Se TRUE (padrão), remove pontuação.
remove_numbers	Se TRUE, remove dígitos. Padrão: FALSE.
remove_url	Se TRUE (padrão), remove URLs (http, https, www).
remove_email	Se TRUE (padrão), remove endereços de e-mail.
remove_symbols	Se TRUE, remove símbolos e emojis (@, #, *, etc.). Padrão: FALSE. Cuidado: hashtags e menções podem ser informativas.
remove_hashtags	Se TRUE, remove hashtags (#termo). Padrão: FALSE.
remove_mentions	Se TRUE, remove menções (\@usuario). Padrão: FALSE.
remove_stopwords	Pode ser: • NULL (padrão): não remove stopwords; • uma string com nome de preset ("pt", "pt-br-extended", "pt-legislativo", "en"); • um vetor character com stopwords customizadas.
remove_accents	Se TRUE, remove acentos (ex: "ação" vira "acao"). Padrão: FALSE.
normalize_pt	Se TRUE, aplica normalizações ortográficas do português brasileiro coloquial: "pra" -> "para", "tá" -> "está", etc. Padrão: FALSE.
protect	Vetor character com termos a preservar exatamente como estão. Útil para siglas (c("PT", "PSDB", "CCJ")). Padrão: NULL.
extra_stopwords	Vetor character com stopwords adicionais a remover antes de qualquer análise, combinado ao preset de remove_stopwords. Use ac_clean_stopwords() para inspecionar e editar o objeto padrão. Padrão: NULL.

`min_char` Inteiro. Descarta tokens com menos de `min_char` caracteres após a limpeza. Padrão: `NULL` (sem filtro).

`custom_replacements`
Lista nomeada de substituições livres aplicadas antes das demais transformações. Ex: `list("pres\\." = "presidente", "dep\\." = "deputado")`. Padrão: `NULL`.

`handle_na` Como tratar valores `NA` no texto:

- `"preserve"` (padrão): mantém `NA` como `NA`;
- `"empty"`: converte `NA` para `""`;
- `"remove"`: remove documentos com `NA` do corpus.

`strip_whitespace`
Se `TRUE` (padrão), colapsa espaços consecutivos.

`verbose` Se `TRUE`, exibe resumo das operações e estatísticas de remoção por etapa. Padrão: `FALSE`.

`...` Ignorado com aviso se houver argumentos não reconhecidos.

Details

Ordem de aplicação das transformações:

1. `handle_na`: tratamento de NAs
2. `custom_replacements`: substituições livres
3. `protect`: proteção de termos com placeholders
4. `remove_url` e `remove_email`
5. `remove_hashtags` e `remove_mentions`
6. `remove_symbols`
7. `lower`
8. `remove_accents`
9. `normalize_pt`
10. `remove_numbers`
11. `remove_punct`
12. `remove_stopwords` + `extra_stopwords`
13. `min_char`: remoção de tokens curtos
14. Restauração dos termos protegidos
15. `strip_whitespace` (sempre)

Value

Um novo objeto `ac_corpus` com a coluna `text` transformada e o atributo `cleaning_steps` registrando as operações aplicadas em ordem. Quando `verbose = TRUE`, também imprime um resumo com tokens removidos por etapa e documentos que ficaram vazios após limpeza.

See Also

`ac_corpus()`, `ac_clean_stopwords()`

Examples

```
df <- data.frame(
  id = c("a", "b", "c"),
  texto = c(
    "O deputado do PT disse: 'Defendo a CCJ!' Veja em https://exemplo.org",
    "Sra. presidente, o Sr. senador apresentou o requerimento n\u00ba 123.",
    "T\u00e1 na hora de votar, pra acabar com isso."
  )
)
corpus <- ac_corpus(df, text = texto, docid = id)

# Limpeza básica
ac_clean(corpus)

# Limpeza completa com todas as opções
ac_clean(
  corpus,
  remove_stopwords  = "pt-legislativo",
  extra_stopwords   = c("isso", "aquilo", "coisa"),
  protect           = c("PT", "CCJ"),
  normalize_pt      = TRUE,
  custom_replacements = list("pres\\." = "presidente"),
  min_char          = 3L,
  verbose           = TRUE
)
```

`ac_clean_stopwords` *Inspecionar e editar stopwords extras do acR*

Description

`ac_clean_stopwords()` retorna (e opcionalmente modifica) o vetor de stopwords adicionais que o pesquisador pode passar a `ac_clean()` via `extra_stopwords`. Funciona como um ponto de partida editável: o pesquisador inspeciona o vetor, adiciona ou remove termos conforme o corpus, e passa o resultado para `extra_stopwords`.

Usage

```
ac_clean_stopwords(
  add = NULL,
  remove = NULL,
  preset = c("empty", "pt", "pt-br-extended", "pt-legislativo")
)
```

Arguments

add	Vetor `character` com termos a adicionar ao vetor padrão.

remove	Vetor character com termos a remover do vetor padrão.
preset	String indicando o ponto de partida: "empty" (padrão), "pt", "pt-br-extended" ou "pt-legislativo".

Value

Vetor character de stopwords pronto para passar a ac_clean(..., extra_stopwords = ...).

See Also

ac_clean()

Examples

```
# Ver o vetor padrão vazio e adicionar termos
sw <- ac_clean_stopwords(add = c("nobre", "ilustre", "respeitavel"))
print(sw)

# Partir do preset legislativo e remover termos que interessam ao corpus
sw <- ac_clean_stopwords(
  preset = "pt-legislativo",
  remove = c("lei", "projeto")   # manter: são relevantes para a análise
)

# Usar na limpeza
# ac_clean(corpus, remove_stopwords = "pt", extra_stopwords = sw)
```

---

ac_cluster_documents *Agrupamento nao supervisionado de documentos*

Description

ac_cluster_documents() particiona os documentos de um corpus em grupos "duros" (hard clustering) com base na similaridade de vocabulario. Serve para descobrir tipologias latentes, montar amostras estratificadas de revisao humana ou produzir dendrogramas para relatorios metodologicos.

Nao substitui ac_lda(): LDA e clustering *soft* (cada documento vira uma mistura de topicos), enquanto esta funcao devolve **uma etiqueta por documento**.

Usage

```
ac_cluster_documents(
  corpus,
  method = c("hclust", "kmeans", "pam"),
  features = c("tfidf", "count"),
  k = NULL,
  distance = c("cosine", "euclidean"),
  min_docs = 15L
)
```

Arguments

corpus	Objeto ac_corpus.
method	Algoritmo. "hclust" (padrao) usa agrupamento hierarquico com ligacao Ward.D2; "kmeans" usa k-means com nstart = 25L; "pam" usa Partitioning Around Medoids (requer cluster em Suggests).
features	Como representar cada documento. "tfidf" (padrao) pesa por TF-IDF; "count" usa contagens brutas.
k	Numero de grupos. Se NULL (padrao) e method %in% c("hclust","kmeans"), tenta escolher automaticamente por silhouette entre 2 e 8; requer cluster. Se cluster nao estiver instalado, k = 3 e o fallback.
distance	Metrica de dissimilaridade. "cosine" (padrao) e o classico em textos; "euclidean" funciona melhor com vetores normalizados.
min_docs	Minimo de documentos para nao emitir warning de corpus pequeno. Padrao: 15L.

Value

Objeto de classe ac_cluster com:

assignments tibble com doc_id e cluster (integer).

fit Objeto bruto do algoritmo (hclust, kmeans ou pam).

k Numero final de grupos.

method, features, distance Parametros usados.

silhouette Silhueta media (NA se cluster nao instalado).

dtm Matriz documento-termo usada (para plotagem).

See Also

ac_lda(), ac_plot_cluster()

Examples

```
# Corpus com dois blocos tematicos
df <- data.frame(
  id    = paste0("d", 1:8),
  texto = c("democracia participacao voto liberdade",
            "cidadania direitos participacao democracia",
            "voto direitos liberdade cidadania",
            "democracia voto participacao popular",
            "mercado economia eficiencia privatizacao",
            "privatizacao mercado livre eficiencia",
            "economia crescimento investimento mercado",
            "eficiencia mercado economia livre"),
  stringsAsFactors = FALSE
)
corpus <- ac_corpus(df, text = texto, docid = id)
clust  <- ac_cluster_documents(corpus, k = 2)
clust$assignments
```

`ac_cooccurrence` *Calcular co-ocorrências de termos*

Description

`ac_cooccurrence()` calcula pares de termos que co-ocorrem dentro de janelas deslizantes ou dentro do mesmo documento, retornando frequências e medidas de associação (PMI, Dice).

Usage

```
ac_cooccurrence(
  corpus,
  window = 5L,
  unit = c("window", "document"),
  measure = c("count", "pmi", "dice"),
  min_count = 2L,
  ...
)
```

Arguments

`corpus`	Objeto `ac_corpus` ou tibble com colunas `doc_id` e `token` (saída de `ac_tokenize()`).
`window`	Tamanho da janela deslizante (número de tokens de cada lado). Padrão: `5`. Ignorado se `unit = "document"`.
`unit`	Unidade de co-ocorrência: `"window"` (padrão) ou `"document"`.
`measure`	Medidas de associação a calcular. Um ou mais de: `"count"` (frequência conjunta), `"pmi"` (pointwise mutual information), `"dice"` (coeficiente Dice). Padrão: `c("count", "pmi")`.
`min_count`	Frequência mínima de co-ocorrência para incluir o par. Padrão: `2`.
`...`	Ignorado.

Value

Tibble com colunas:

- `word1`, `word2`: par de termos (ordenado alfabeticamente);
- `cooc`: frequência de co-ocorrência;
- `pmi` (se solicitado): pointwise mutual information;
- `dice` (se solicitado): coeficiente Dice.

See Also

`ac_tokenize()`, `ac_plot_cooccurrence()`

Examples

```
# Corpus tematico: coocorrencia revela associacoes conceituais
# (que palavras "andam juntas" no discurso).
df <- data.frame(
  id = paste0("d", 1:6),
  texto = c(
    "democracia participacao voto liberdade cidadania popular",
    "cidadania direitos participacao democracia representacao politica",
    "voto direitos liberdade cidadania soberania popular",
    "mercado economia eficiencia privatizacao competicao livre",
    "privatizacao mercado livre eficiencia produtividade lucro",
    "economia crescimento investimento mercado capital juros"
  ),
  stringsAsFactors = FALSE
)

corpus <- ac_corpus(df, text = texto, docid = id) |>
  ac_clean()
tokens <- ac_tokenize(corpus)

# Janela de 3 tokens, sem filtro de frequencia minima
# (em corpora reais use min_count >= 5 para reduzir ruido)
cooc <- ac_cooccurrence(tokens, window = 3L, min_count = 1L)
head(cooc)

# Coocorrencia com PMI (associacao normalizada por acaso)
cooc_pmi <- ac_cooccurrence(tokens, window = 3L, measure = "pmi",
                            min_count = 1L)
head(cooc_pmi)
```

`ac_corpus` *Construir um corpus para análise de conteúdo*

Description

`ac_corpus()` é a porta de entrada do pipeline do pacote `acR`. Constrói um objeto estruturado de corpus a partir de um `data.frame`, um vetor de caracteres, ou um corpus do `quanteda`, validando a entrada e preservando metadados associados aos documentos.

Usage

```
ac_corpus(data, text = NULL, docid = NULL, meta = NULL, lang = "pt", ...)
```

Arguments

`data`	Entrada do corpus. Pode ser: • um `data.frame` ou `tibble` com pelo menos uma coluna de texto; • um vetor `character` com textos (um documento por elemento);

- um objeto corpus do pacote quanteda.

text	Nome da coluna de texto, sem aspas (usa *tidyselect* / *non-standard evaluation*). Apenas relevante quando data é um data.frame. Se NULL (padrão), a função tenta detectar automaticamente uma coluna chamada text, texto, doc, content ou conteudo.
docid	Nome da coluna de identificador único, sem aspas. Se NULL (padrão), gera IDs sequenciais no formato doc_1, doc_2, ...
meta	Colunas de metadados a preservar, via *tidyselect* (ex: c(partido, data, regiao) ou tidyselect::starts_with("var_")). Se NULL, preserva todas as demais colunas do data.frame.
lang	Código ISO do idioma do corpus (padrão: "pt"). Utilizado em etapas posteriores (lematização, stopwords, léxicos).
...	Argumentos reservados para extensão futura. No momento, são ignorados com aviso se nomeados de forma desconhecida.

Details

A função realiza cinco validações obrigatórias e falha cedo em caso de problema:

1. data deve ser de um dos tipos suportados;
2. a coluna de texto deve existir (ou ser detectável automaticamente);
3. a coluna de texto não pode ser inteiramente NA ou vazia;
4. doc_id não pode ter valores duplicados;
5. doc_id não pode conter NA.

Documentos com texto vazio ou NA geram aviso (warning) mas são mantidos no corpus com texto "" — o usuário decide se filtra depois.

Value

Um objeto de classe ac_corpus (que herda de tbl_df, tbl, data.frame) com as colunas:

- doc_id (character): identificador único do documento.
- text (character): texto do documento.
- Demais colunas de metadados, preservadas de data. O objeto traz também o atributo lang indicando o idioma.

Examples

```
# A partir de um data.frame
df <- data.frame(
  id = c("a", "b", "c"),
  texto = c("Primeiro texto.", "Segundo texto.", "Terceiro."),
  partido = c("PT", "PL", "MDB")
)
corpus <- ac_corpus(df, text = texto, docid = id, meta = partido)
corpus
```

```
# A partir de um vetor character (doc_id gerado automaticamente)
ac_corpus(c("Texto um.", "Texto dois."))

# Detecção automática da coluna de texto
df2 <- data.frame(text = c("A", "B"), autor = c("X", "Y"))
ac_corpus(df2)
```

`ac_count` *Contar frequências de tokens ou n-gramas em um corpus*

Description

`ac_count()` calcula frequências de tokens ou n-gramas a partir de um objeto `ac_corpus()`, usando internamente `ac_tokenize()` seguido de uma agregação com `dplyr::count()`.

É o insumo direto de quase toda análise quantitativa do pacote: `ac_top_terms()` pega o topo, `ac_tf_idf()` recalibra por distintividade, `ac_keyness()` compara grupos, `ac_wordcloud()` visualiza. O argumento by decide se a unidade de análise é o documento ou o grupo agregado.

Pode operar em dois níveis:

- por documento (`by = NULL`): contagens por `doc_id` e `token`;
- por metadados (`by = c("partido", ...)`): contagens por variáveis de agrupamento e `token` (agregando vários documentos).

Usage

```
ac_count(corpus, n = 1L, drop_punct = FALSE, by = NULL, sort = TRUE, ...)
```

Arguments

`corpus`	Objeto de classe `ac_corpus()`.
`n`	Tamanho do n-grama a ser tokenizado. Encaminhado para `ac_tokenize()`. Veja `?ac_tokenize` para detalhes.
`drop_punct`	Logico. Se `TRUE`, remove tokens compostos apenas por pontuacao antes de calcular as frequencias (via `drop_punct = TRUE` em `ac_tokenize()`).
`by`	Vetor de nomes de colunas de metadados em `corpus` a serem usados como grupos de agregacao. Se `NULL` (padrao), as contagens sao feitas por documento (`doc_id`). Se nao for `NULL`, as colunas indicadas sao usadas como grupos e `doc_id` nao entra no resultado.
`sort`	Logico. Se `TRUE` (padrao), ordena a saida em ordem decrescente de frequencia (`n`).
`...`	Argumentos adicionais encaminhados para `ac_tokenize()` (por exemplo, `keep_empty`).

Value

Um `tibble::tibble()`:

- Se by = NULL: colunas doc_id, token, n;
- Se by nao nulo: colunas by, token, n.

See Also

`ac_corpus()`, `ac_clean()`, `ac_tokenize()`

Examples

```
df <- data.frame(
  id      = c("d1", "d2", "d3"),
  texto   = c(
    "O deputado do PT falou na CCJ.",
    "O deputado do PL falou novamente.",
    "O senador do PT falou na CCJ."
  ),
  partido = c("PT", "PL", "PT"),
  stringsAsFactors = FALSE
)

corp <- ac_corpus(df, text = texto, docid = id, meta = partido)

# Frequencia de palavras (unigramas) por documento
ac_count(corp)

# Frequencia de palavras por partido
ac_count(corp, by = "partido")

# Frequencia de bigramas por partido, removendo apenas pontuacao
ac_count(corp, n = 2, drop_punct = TRUE, by = "partido")
```

---

ac_estimate_tokens *Estimar numero de tokens de um texto*

Description

`ac_estimate_tokens()` estima o numero de tokens de entrada que um texto vai consumir, usando uma heuristica de caracteres por token. Util para prever o custo de uma rodada de `ac_qual_code()` ou para detectar, antes de enviar ao LLM, documentos que provavelmente excederao a janela de contexto do modelo.

E uma estimativa deliberadamente barata: nao faz nenhuma chamada de rede nem depende do tokenizador exato de um modelo especifico (que varia por provedor e as vezes nem e publico). Para calibrar a heuristica contra um modelo real, use `ac_calibrate_tokens()` com uma amostra pequena de textos.

Usage

```
ac_estimate_tokens(text, chat = NULL, chars_per_token = 3.2)
```

Arguments

text	Vetor de caracteres com os textos.
chat	Ignorado nesta versao (reservado para uma futura estimativa especifica de provedor via chat$token_count()). Receber o argumento ja na assinatura evita quebrar codigo que o passar.
chars_per_token	Caracteres por token, usado para a estimativa. Padrao 3.2 (razoavel para portugues via BPE; varia por tokenizador e idioma). Calibre com ac_calibrate_tokens() se precisar de precisao maior para um modelo especifico.

Value

Vetor inteiro com a estimativa de tokens por texto.

Examples

```
ac_estimate_tokens("Este e um texto de exemplo para estimar tokens.")
ac_estimate_tokens(c("texto curto", "um texto um pouco mais longo que o anterior"))
```

---

ac_export *Exporta resultados de analise de conteudo em multiplos formatos*

Description

Exporta data.frames de resultados (corpus codificado, tabelas de frequencia, metricas de confiabilidade) para CSV, LaTeX, Excel (.xlsx) ou RDS. Pensada para facilitar a transicao entre o pipeline acR e a escrita academica (tabelas em artigos) ou o compartilhamento de dados replicaveis.

Usage

```
ac_export(
  x,
  path = NULL,
  format = NULL,
  overwrite = TRUE,
  latex_caption = NULL,
  latex_label = NULL,
  latex_digits = 3L,
  excel_sheet = "acR",
  verbose = TRUE
)
```

Arguments

x	data.frame ou objeto ac_irr. Dados a exportar.
path	character. Caminho do arquivo de saida, incluindo extensao (ex.: "resultados.csv", "tabela1.tex", "dados.xlsx"). Se NULL, o formato e inferido pelo argumento format e o arquivo e salvo no diretorio de trabalho com nome "acR_export".
format	character. Formato de saida: "csv", "latex", "xlsx", "rds". Se NULL (padrao), inferido pela extensao de path.
overwrite	logical. Se TRUE (padrao), sobrescreve arquivo existente. Se FALSE, lanca erro se o arquivo ja existir.
latex_caption	character ou NULL. Legenda da tabela LaTeX.
latex_label	character ou NULL. Label para \\ref{} no LaTeX. Ex.: "tab:resultados".
latex_digits	integer. Casas decimais para colunas numericas na saida LaTeX. Padrao: 3.
excel_sheet	character. Nome da aba no arquivo Excel. Padrao: "acR".
verbose	logical. Se TRUE (padrao), confirma o caminho salvo.

Details

CSV:

Usa utils::write.csv() com row.names = FALSE e encoding UTF-8. Separador: virgula. Adequado para importacao em Stata, SPSS, Python e R.

LaTeX:

Gera codigo LaTeX via knitr::kable() com format = "latex", empacotado em ambiente table com \\centering, \\caption e \\label. Adicione \\usepackage{booktabs} no preambulo do documento para melhor tipografia.

Excel:

Usa writexl::write_xlsx() — sem dependencia de Java ou LibreOffice. Cria arquivo .xlsx compativel com Excel 2007+, Google Sheets e LibreOffice Calc.

RDS:

Serializa o objeto R completo via base::saveRDS(). Preserva tipos, atributos e classes (incluindo objetos ac_irr, corpus etc.). Ideal para replicabilidade interna ao projeto.

Value

Invisivel: o caminho do arquivo salvo (character).

See Also

ac_qual_irr(), ac_qual_code()

Examples

```
resultados <- data.frame(
  id_discurso   = c("d1", "d2", "d3"),
  nome_deputado = c("Dep. A", "Dep. B", "Dep. C"),
  categoria     = c("progressista", "conservador", "tecnocratico"),
  confianca     = c(0.94, 0.91, 0.87)
)

# Escrevemos em tempdir() para nao poluir o filesystem do usuario
out_csv <- file.path(tempdir(), "resultados.csv")
out_tex <- file.path(tempdir(), "resultados.tex")
out_rds <- file.path(tempdir(), "resultados.rds")

# CSV
ac_export(resultados, out_csv)

# LaTeX (para incluir em artigo)
ac_export(
  resultados,
  out_tex,
  latex_caption = "Classificacao do tom dos discursos por LLM",
  latex_label   = "tab:tom_discursos"
)

# RDS (replicabilidade)
ac_export(resultados, out_rds)

# Excel (requer openxlsx ou writexl instalado)
if (requireNamespace("writexl", quietly = TRUE) ||
    requireNamespace("openxlsx", quietly = TRUE)) {
  out_xlsx <- file.path(tempdir(), "resultados.xlsx")
  ac_export(resultados, out_xlsx, excel_sheet = "Classificacao")
}

# Objeto ac_irr (comparar codificacao humana vs. LLM)
gold <- data.frame(
  id_discurso = c("d1", "d2", "d3"),
  categoria   = c("progressista", "conservador", "tecnocratico")
)
predicted <- data.frame(
  id_discurso = c("d1", "d2", "d3"),
  categoria   = c("progressista", "conservador", "progressista")
)
irr_result <- ac_qual_irr(gold, predicted, verbose = FALSE)
ac_export(irr_result, file.path(tempdir(), "confiabilidade.csv"))
```

`ac_fetch_camara` *Busca discursos de deputados federais via API da Camara dos Deputados*

Description

Coleta discursos parlamentares diretamente da API publica da Camara dos Deputados (v2), retornando um `data.frame` padronizado e pronto para uso nas funcoes do `acR`. A coleta e feita em duas etapas: (1) lista deputados conforme os filtros informados; (2) para cada deputado, busca os discursos no periodo solicitado, com paginacao automatica.

Usage

```
ac_fetch_camara(
  data_inicio,
  data_fim,
  legislatura = NULL,
  partido = NULL,
  uf = NULL,
  n_max = 100,
  tipo_discurso = "plenario",
  verbose = TRUE,
  sleep = 0.5
)
```

Arguments

`data_inicio`	`character`. Data de inicio no formato `"YYYY-MM-DD"`.
`data_fim`	`character`. Data de fim no formato `"YYYY-MM-DD"`.
`legislatura`	`integer` ou `NULL`. Numero da legislatura (ex.: `57` para 2023-2027). Se `NULL`, usa o periodo definido por `data_inicio` e `data_fim` sem filtrar por legislatura.
`partido`	`character` ou `NULL`. Sigla do partido para filtrar deputados (ex.: `"PT"`, `"PL"`, `"MDB"`). Aceita vetor de siglas. Se `NULL`, inclui todos os partidos.
`uf`	`character` ou `NULL`. Sigla da UF para filtrar deputados (ex.: `"SP"`, `"MG"`). Aceita vetor de UFs. Se `NULL`, inclui todas.
`n_max`	`integer`. Numero maximo de discursos a retornar. Padrao: `100`. Use `Inf` para coletar todos (atencao: pode ser lento).
`tipo_discurso`	`character`. Tipo de evento parlamentar. Opcoes principais: `"plenario"` (padrao), `"comissao"`, `"todos"`.
`verbose`	`logical`. Se `TRUE` (padrao), exibe mensagens de progresso.
`sleep`	`numeric`. Tempo de espera (em segundos) entre chamadas a API para respeitar o rate limit. Padrao: `0.5`.

Details

A API Dados Abertos da Camara (v2) nao dispoe de endpoint unico para discursos por periodo. O fluxo de coleta e:

1. `GET /api/v2/deputados` - lista deputados com filtros de partido/UF.
2. `GET /api/v2/deputados/{id}/discursos` - discursos de cada deputado, com paginacao (max. 100 itens por pagina).

O filtro `tipo_discurso` atua sobre o campo `tipoDiscurso` retornado pela API. Valores observados: `"DISCURSO"`, `"DISCURSO ENCAMINHADO"`, `"BREVE COMUNICACAO"`, `"PELA ORDEM"`, `"COMUNICACAO PARLAMENTAR"`.

Value

Um `data.frame` com as colunas:

`id_discurso character`. Identificador unico do discurso.

`id_deputado integer`. ID do deputado na API da Camara.

`nome_deputado character`. Nome civil do parlamentar.

`partido character`. Sigla do partido na data do discurso.

`uf character`. UF da bancada do parlamentar.

`data Date`. Data do discurso.

`hora_inicio character`. Hora de inicio (HH:MM).

`tipo_discurso character`. Tipo de fase do evento.

`sumario character`. Sumario do discurso (quando disponivel).

`texto character`. Texto integral do discurso (quando disponivel).

`uri_discurso character`. URI do recurso na API.

References

CAMARA DOS DEPUTADOS. Dados Abertos da Camara dos Deputados - API v2. Brasilia, 2024. Disponivel em: https://dadosabertos.camara.leg.br/swagger/api.html. Acesso em: abr. 2026.

See Also

`ac_corpus()` para transformar o resultado em corpus.

Examples

```
## Not run:
# Discursos do plenario, marco de 2024
disc <- ac_fetch_camara(
  data_inicio = "2024-03-11",
  data_fim    = "2024-03-15",
  n_max       = 50
)

# Apenas PT e PL
disc_partidos <- ac_fetch_camara(
  data_inicio = "2024-01-01",
  data_fim    = "2024-03-31",
  partido     = c("PT", "PL"),
  n_max       = 100
)
```

```
# Todos os tipos de discurso
disc_todos <- ac_fetch_camara(
  data_inicio   = "2024-03-01",
  data_fim      = "2024-03-31",
  uf            = "SP",
  tipo_discurso = "todos",
  n_max         = 100
)

## End(Not run)
```

`ac_fetch_senado` *Busca discursos de senadores federais via senatebR*

Description

Coleta discursos parlamentares do Senado Federal usando o pacote **senatebR** como backend, retornando um `data.frame` no mesmo formato padronizado de `ac_fetch_camara()`. Isso permite combinar corpora das duas casas legislativas sem atrito.

O periodo pode ser especificado por legislatura (atalho conveniente) ou por datas exatas (controle fino). Se ambos forem fornecidos, as datas prevalecem.

Usage

```
ac_fetch_senado(
  data_inicio = NULL,
  data_fim = NULL,
  legislatura_inicio = NULL,
  legislatura_fim = NULL,
  partido = NULL,
  uf = NULL,
  nome_senador = NULL,
  n_max = 100,
  verbose = TRUE,
  sleep = 0.3
)
```

Arguments

`data_inicio`	`character` ou `NULL`. Data de inicio no formato `"YYYY-MM-DD"`. Se `NULL`, usa o inicio da `legislatura_inicio`.
`data_fim`	`character` ou `NULL`. Data de fim no formato `"YYYY-MM-DD"`. Se `NULL`, usa o fim da `legislatura_fim`.
`legislatura_inicio`	`integer` ou `NULL`. Numero da legislatura de inicio (ex.: `56` para 2019-2023, `57` para 2023-2027). Ignorado se `data_inicio` for fornecido.

`legislatura_fim`
: `integer` ou `NULL`. Numero da legislatura de fim. Se `NULL`, usa o mesmo valor de `legislatura_inicio`.

`partido`
: `character` ou `NULL`. Sigla(s) do partido para filtrar senadores (ex.: `"PT"`, `c("PT", "PL")`). Filtragem pos-coleta. Se `NULL`, inclui todos os partidos.

`uf`
: `character` ou `NULL`. Sigla(s) da UF para filtrar senadores (ex.: `"SP"`, `c("SP", "MG")`). Filtragem pos-coleta. Se `NULL`, inclui todas as UFs.

`nome_senador`
: `character` ou `NULL`. Padrao de texto para filtrar pelo nome do senador (busca parcial, case-insensitive). Ex.: `"Lula"`, `"Pacheco"`. Se `NULL`, inclui todos.

`n_max`
: `integer`. Numero maximo de discursos a retornar. Padrao: `100`. Use `Inf` para coletar todos.

`verbose`
: `logical`. Se `TRUE` (padrao), exibe mensagens de progresso.

`sleep`
: `numeric`. Pausa em segundos entre requisicoes. Padrao: `0.3`.

Details

Legislaturas do Senado Federal:

Legislatura	Periodo
55	2015-2019
56	2019-2023
57	2023-2027

Compatibilidade com ac_fetch_camara():
O `data.frame` retornado tem as mesmas colunas de `ac_fetch_camara()`, com a adicao da coluna `casa` (`"senado"`). Isso permite combinar os dois corpora com `rbind()` ou `dplyr::bind_rows()`.

Backend:
Esta funcao usa **senatebR** (Santos, 2026) como backend para acesso a API do Senado Federal. O **senatebR** deve estar instalado: `install.packages("senatebR")`.

Value

Um `data.frame` com as mesmas colunas de `ac_fetch_camara()`:

`id_discurso character`. Identificador unico.

`id_deputado character`. Codigo do senador na API.

`nome_deputado character`. Nome do senador.

`partido character`. Sigla do partido.

`uf character`. UF da representacao.

`data Date`. Data do discurso.

`hora_inicio character`. Hora de inicio (HH:MM).

`tipo_discurso character`. Tipo/fase do discurso.

`sumario character`. Resumo do discurso.

`texto character`. Texto integral (quando disponivel).

`uri_discurso character`. URL do recurso.

`casa character`. Sempre `"senado"`.

References

SANTOS, V. **senatebR**: Collect Data from the Brazilian Federal Senate Open Data API. CRAN, 2026. Disponivel em: https://github.com/vsntos/senatebR.

See Also

ac_fetch_camara(), ac_corpus()

Examples

```
## Not run:
# Por legislatura
disc_sen <- ac_fetch_senado(
  legislatura_inicio = 57,
  n_max = 50
)

# Por datas + filtro de partido
disc_pt <- ac_fetch_senado(
  data_inicio = "2024-01-01",
  data_fim    = "2024-06-30",
  partido     = c("PT", "PL"),
  n_max       = 100
)

# Combinar Camara + Senado
disc_camara <- ac_fetch_camara(
  data_inicio = "2024-03-01",
  data_fim    = "2024-03-31",
  n_max       = 50
)
disc_senado <- ac_fetch_senado(
  data_inicio = "2024-03-01",
  data_fim    = "2024-03-31",
  n_max       = 50
)
corpus_bicameral <- dplyr::bind_rows(disc_camara, disc_senado)

## End(Not run)
```

---

ac_import *Importar arquivos para um corpus acR*

Description

Importa arquivos de texto em diferentes formatos (PDF, Word, Excel, CSV, TXT, JSON, imagens com OCR) e retorna diretamente um objeto ac_corpus. Suporta caminhos individuais, vetores de arquivos e globs (dados/*.pdf).

É a **via de entrada principal** do pipeline quando você trabalha com arquivos em disco (relatórios, transcrições, PDFs escaneados). Faz o que `ac_corpus()` faria a partir de um `data.frame`, mas cobrindo o passo anterior: extração de texto de formatos heterogêneos com detecção automática pelo sufixo do arquivo. Para PDFs escaneados e imagens, aciona OCR via `tesseract` com português como padrão.

Usage

```
ac_import(path, text_field = NULL, lang_ocr = "por", id_from = "filename", ...)
```

Arguments

`path`	Caminho para um arquivo, vetor de arquivos ou glob (ex: `'dados/*.docx'`). Formatos suportados: `.pdf`, `.doc`, `.docx`, `.xlsx`, `.xls`, `.csv`, `.txt`, `.json`, `.png`, `.jpg`, `.jpeg`, `.tiff`.
`text_field`	Nome da coluna que contem o texto, para arquivos tabulares (`.xlsx`, `.xls`, `.csv`). Obrigatorio nesses formatos.
`lang_ocr`	Lingua para OCR em imagens e PDFs escaneados. Padrao: `'por'` (portugues). Use `tesseract::tesseract_info()` para ver linguas instaladas.
`id_from`	Como gerar os IDs dos documentos. `'filename'` (padrao) usa o nome do arquivo sem extensao. `'rownum'` usa numeros sequenciais. Ou um vetor de strings com os IDs desejados.
`...`	Argumentos adicionais passados para `readtext::readtext()`.

Details

O `ac_import()` detecta automaticamente o formato pelo extension e aciona o parser correto:

- **PDF com texto selecionavel**: `readtext` + `pdftools`
- **PDF escaneado / imagem**: `tesseract` (OCR)
- **Word** (`.doc`, `.docx`): `readtext`
- **Excel** (`.xlsx`, `.xls`): `readtext` (requer `text_field`)
- **CSV**: `readtext` (requer `text_field`)
- **TXT / JSON**: `readtext`
- **Pasta inteira / glob**: todos os arquivos compativeis de uma vez

A ordem dos documentos no corpus resultante segue a ordem de entrada em `path` (ou a ordem alfabetica retornada por `Sys.glob()`), independente de o parser ser OCR ou readtext. IDs duplicados (dois arquivos com o mesmo `basename`) sao desambiguados automaticamente com um sufixo `_2`, `_3`, ... e um aviso e emitido.

Dependencias opcionais: `readtext` e `tesseract` nao sao importados automaticamente – o `ac_import()` verifica se estao instalados e orienta a instalacao caso necessario.

Value

Um objeto `ac_corpus` pronto para uso com todas as funcoes do acR.

References

Benoit, K., et al. (2018). readtext: Import and Handling for Plain and Formatted Text Files. R package. https://CRAN.R-project.org/package=readtext

Ooms, J. (2024). tesseract: Open Source OCR Engine. R package. https://CRAN.R-project.org/package=tesseract

See Also

ac_corpus(), ac_clean(), ac_tokenize()

Examples

```
## Not run:
# PDF com texto selecionavel
corpus <- ac_import('relatorio.pdf')

# Pasta inteira de Word
corpus <- ac_import('proposicoes/*.docx')

# Excel -- indicar a coluna de texto
corpus <- ac_import('respostas.xlsx', text_field = 'resposta')

# PDF escaneado (OCR em portugues)
corpus <- ac_import('ata_manuscrita.pdf', lang_ocr = 'por')

# Imagem (OCR)
corpus <- ac_import('captura.png')

# Pasta mista (PDF + DOCX + TXT)
corpus <- ac_import('dados/*')

## End(Not run)
```

---

ac_keyness *Calcular estatisticas de keyness entre dois grupos*

Description

ac_keyness() calcula estatisticas de "keyness" para comparar a distribuicao de termos entre dois grupos. O criterio que define os grupos e do pesquisador: partido, periodo, tema, regiao, autor, condicao experimental ou qualquer variavel categorica no corpus. A funcao e inspirada em quanteda.textstats::textstat_keyness() e utiliza tabelas 2x2 por termo.

A entrada tipica e uma tabela de frequencias gerada por ac_count(), agregada por uma coluna de grupo:

- ac_count(corp, by = "grupo") seguido de
- ac_keyness(freq, group = "grupo", target = "A").

Usage

```
ac_keyness(x, group, target, measure = c("chi2", "ll"), sort = TRUE)
```

Arguments

x	Um data.frame ou tibble::tibble() contendo, no minimo, as colunas token, n e uma coluna de grupo.
group	Nome da coluna em x que identifica os grupos (string). Essa coluna deve possuir exatamente dois valores distintos.
target	Valor de group que sera considerado o grupo alvo (por exemplo, "Governo"). O outro valor sera tratado como grupo de referencia.
measure	Estatistica de keyness a ser usada. Pode ser "chi2" (padrao) para qui-quadrado com 1 grau de liberdade, ou "ll" para log-likelihood (G^2).
sort	Logico. Se TRUE (padrao), ordena a saida por keyness em ordem decrescente (termos mais caracteristicos do grupo alvo no topo).

Details

Para cada termo, a funcao constroi uma tabela 2x2:

- a = n_target (frequencia do termo no grupo alvo)
- b = n_reference (frequencia do termo no grupo de referencia)
- c = total_target - a
- d = total_reference - b

Em seguida, calcula:

- qui-quadrado com 1 g.l. na opcao measure = "chi2";
- log-likelihood ratio (G^2) na opcao measure = "ll".

Em ambos os casos, a estatistica e multiplicada pelo sinal da diferenca de frequencias relativas (a / total_target - b / total_reference), de forma que valores positivos indiquem termos mais caracteristicos do grupo alvo e valores negativos termos mais caracteristicos do grupo de referencia.

Value

Um tibble::tibble() com uma linha por termo, contendo:

- token
- group (nome da coluna de grupo, repetido para referencia)
- target, reference
- n_target, n_reference
- total_target, total_reference
- keyness (estatistica assinada, positiva quando o termo e mais frequente no grupo alvo, negativa quando e mais frequente no grupo de referencia)
- direction (nome do grupo em que o termo e relativamente mais frequente)

See Also

ac_count(), ac_top_terms(), ac_tf_idf()

Examples

```
# Comparar vocabulario entre dois grupos (aqui, dois temas legislativos).
# O criterio de grupo e do pesquisador: partido, periodo, regiao, tema...
df <- data.frame(
  id    = paste0("d", 1:8),
  texto = c(
    "reforma tributaria simplifica sistema impostos empresas",
    "IVA dual substitui PIS COFINS ICMS federal",
    "aliquotas excecoes tributarias setores economicos",
    "reforma tributaria unifica impostos indiretos",
    "programa habitacional amplia recursos moradia popular",
    "deficit habitacional afeta familias baixa renda",
    "urbanizacao favelas regularizacao fundiaria municipios",
    "moradia digna direito social constituicao"
  ),
  tema  = rep(c("tributario", "habitacao"), each = 4),
  stringsAsFactors = FALSE
)

corp <- ac_corpus(df, text = texto, docid = id)
freq <- ac_count(corp, by = "tema")

# Termos-chave do grupo "tributario" versus o resto
key <- ac_keyness(freq, group = "tema", target = "tributario")
head(key)
```

---

ac_lda *Ajustar modelo LDA (Latent Dirichlet Allocation)*

Description

ac_lda() ajusta um modelo de tópicos LDA sobre um ac_corpus, retornando um objeto com os resultados do modelo e tibbles tidy de termos por tópico e prevalência por documento.

Usage

```
ac_lda(corpus, k = 10L, seed = 42L, method = c("VEM", "Gibbs"), ...)
```

Arguments

corpus	Objeto ac_corpus.
k	Número de tópicos. Padrão: 10.
seed	Semente para reprodutibilidade. Padrão: 42.
method	Método de estimação: "VEM" (padrão) ou "Gibbs".
...	Argumentos adicionais passados a topicmodels::LDA().

Value

Lista de classe `ac_lda` com:

- `model`: objeto `LDA` do pacote `topicmodels`;
- `terms`: tibble com beta (probabilidade de cada termo por tópico);
- `documents`: tibble com gamma (prevalência de cada tópico por documento);
- `k`: número de tópicos;
- `params`: parâmetros usados.

References

Blei, D. M.; Ng, A. Y.; Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3, 993-1022.

Vignette: `vignette("lda", package = "acR")` — tutorial completo com corpus real e visualizações.

See Also

`ac_lda_tune()`, `ac_plot_lda_topics()`

Examples

```
# 1. Corpus sintetico com 3 temas nitidos (democracia, economia, saude, educacao)
df <- data.frame(
  id = paste0("d", 1:10),
  texto = c(
    "democracia participacao cidadania direitos politica",
    "mercado economia privatizacao crescimento fiscal",
    "saude hospital medico doenca tratamento",
    "educacao escola professor ensino aprendizagem",
    "democracia eleicao voto politica partido",
    "economia inflacao juros fiscal orcamento",
    "saude sus medico hospital remedio",
    "educacao universidade pesquisa ciencia",
    "participacao social cidadania direitos igualdade",
    "mercado trabalho emprego salario industria"
  )
)
corpus <- ac_corpus(df, text = texto, docid = id)

# 2. Ajustar LDA com k=3 topicos (o corpus tem ~4 temas; k=3 for a experimentacao)
lda <- ac_lda(corpus, k = 3)
lda  # imprime resumo do modelo
```

ac_lda_tune *Ajustar múltiplos modelos LDA para selecionar k*

Description

ac_lda_tune() ajusta modelos LDA para diferentes valores de k e calcula métricas de qualidade para auxiliar na seleção do número ideal de tópicos.

Usage

```
ac_lda_tune(
  corpus,
  k_range = 5:20,
  seed = 42L,
  method = c("VEM", "Gibbs"),
  ...
)
```

Arguments

corpus	Objeto ac_corpus.
k_range	Vetor de valores de k a testar. Padrão: 5:20.
seed	Semente. Padrão: 42.
method	Método de estimação. Padrão: "VEM".
...	Ignorado.

Value

Tibble com colunas k, perplexity e, se disponível, métricas do pacote ldatuning.

See Also

ac_lda(), ac_plot_lda_tune()

Examples

```
# 1. Corpus sintetico (mesmos temas de ac_lda(), palavras diferentes)
df <- data.frame(
  id = paste0("d", 1:10),
  texto = c(
    "democracia participacao politica voto eleicao",
    "mercado economia fiscal privatizacao",
    "saude hospital medico doenca sus",
    "educacao escola professor universidade",
    "democracia cidadania direitos igualdade",
    "economia inflacao juros orcamento",
    "saude medico remedio tratamento",
```

```
    "educacao pesquisa ciencia tecnologia",
    "participacao social democracia cidadania",
    "mercado trabalho emprego salario"
  )
)
corpus <- ac_corpus(df, text = texto, docid = id)

# 2. Testar k de 2 a 5 e comparar perplexidade (menor = melhor ajuste)
tune <- ac_lda_tune(corpus, k_range = 2:5)
tune
```

`ac_palette` *Paleta categórica do acR*

Description

Retorna a paleta categórica padrão do pacote — cores compatíveis com contraste WCAG AA e teste de daltonismo (deuteranopia). Deriva da paleta Okabe-Ito, referência para acessibilidade em visualização científica.

Usage

```
ac_palette(n = 8L)
```

Arguments

`n`	Número de cores a retornar (max 8). Padrão: `8`.

Value

Vetor `character` com códigos hex.

See Also

`theme_ac()`

Examples

```
ac_palette()      # todas as 8 cores
ac_palette(3)     # primeiras 3

if (requireNamespace("ggplot2", quietly = TRUE)) {
  ggplot2::ggplot(iris, ggplot2::aes(Sepal.Length, Petal.Length,
                                     color = Species)) +
    ggplot2::geom_point(size = 2, alpha = 0.85) +
    ggplot2::scale_color_manual(values = ac_palette(3)) +
    theme_ac()
}
```

ac_plot_cluster *Visualiza um objeto ac_cluster*

Description

Tres formas complementares de olhar para o mesmo cluster:

- "dendrogram" (padrao para method = "hclust"): arvore de fusoes com cortes coloridos por grupo.
- "scatter": projecao 2D via PCA sobre a matriz documento-termo, com pontos coloridos por cluster.
- "heatmap": mapa de calor da matriz de dissimilaridade, ordenado pelo dendrograma.

Usage

```
ac_plot_cluster(
  x,
  kind = c("auto", "dendrogram", "scatter", "heatmap"),
  title = NULL,
  palette = NULL
)
```

Arguments

x	Objeto ac_cluster (saida de ac_cluster_documents()).
kind	Tipo de grafico. "auto" (padrao) escolhe conforme o metodo (dendrograma para hclust, scatter para kmeans/pam).
title	Titulo do grafico.
palette	Vetor de cores para os clusters. Padrao: ac_palette().

Value

Objeto ggplot.

See Also

ac_cluster_documents()

Examples

```
df <- data.frame(
  id = paste0("d", 1:8),
  texto = c("democracia participacao voto",
            "cidadania direitos participacao",
            "voto direitos liberdade",
            "democracia voto participacao",
            "mercado economia eficiencia",
```

```
            "privatizacao mercado livre",
            "economia crescimento mercado",
            "eficiencia mercado livre")
)
corpus <- ac_corpus(df, text = texto, docid = id)
clust  <- ac_cluster_documents(corpus, k = 2)
ac_plot_cluster(clust)
```

---

ac_plot_cooccurrence *Visualizar rede de co-ocorrência de termos*

Description

ac_plot_cooccurrence() gera um gráfico de rede a partir de um tibble de co-ocorrências (saída de ac_cooccurrence()), usando ggplot2 e ggraph.

Usage

```
ac_plot_cooccurrence(
  cooc,
  top_n = 50L,
  weight = c("cooc", "pmi", "dice"),
  layout = "fr",
  node_color = "#0072B2",
  edge_color = "grey70",
  title = NULL,
  ...
)
```

Arguments

cooc	Tibble com co-ocorrências, saída de ac_cooccurrence().
top_n	Número de pares mais frequentes a exibir. Padrão: 50.
weight	Coluna a usar como peso das arestas: "cooc" (padrão), "pmi" ou "dice".
layout	Layout do grafo. Qualquer layout suportado por ggraph::ggraph(): "fr" (Fruchterman-Reingold, padrão), "kk", "circle", etc.
node_color	Cor dos nós. Padrão: "#0072B2".
edge_color	Cor das arestas. Padrão: "grey70".
title	Título do gráfico. Padrão: NULL.
...	Ignorado.

Value

Objeto ggplot.

See Also

ac_cooccurrence()

Examples

```
# 1. Corpus pequeno de tres documentos
df <- data.frame(
  id = c("d1", "d2", "d3"),
  texto = c(
    "democracia participacao cidadania",
    "participacao politica democracia",
    "cidadania direitos participacao"
  )
)

# 2. Calcular co-ocorrencias de termos
corpus <- ac_corpus(df, text = texto, docid = id) |> ac_clean()
cooc <- ac_cooccurrence(ac_tokenize(corpus), min_count = 1)

# 3. Rede visualizada com ggraph (dependencia opcional)
if (requireNamespace("ggraph", quietly = TRUE)) {
  ac_plot_cooccurrence(cooc)
}
```

---

ac_plot_keyness *Plotar estatisticas de keyness*

Description

ac_plot_keyness() cria um grafico de barras com as estatisticas de keyness calculadas por ac_keyness(), destacando os termos mais caracteristicos do grupo alvo e do grupo de referencia.

A funcao usa ggplot2 como base e pode, opcionalmente, aplicar o estilo editorial do pacote ipeaplot.

Usage

```
ac_plot_keyness(
  x,
  n = NULL,
  style = c("default", "ipea"),
  flip = TRUE,
  show_reference = TRUE
)
```

Arguments

x	Um data.frame ou tibble::tibble() contendo, no minimo, as colunas token, keyness e direction.
n	Numero de termos a exibir por direcao. Se NULL (padrao), usa todas as linhas de x. Se informado, seleciona os top n termos com keyness positivo e os top n com keyness negativo.
style	Estilo grafico. Pode ser "default" (padrao) ou "ipea". Quando "ipea", a funcao tenta aplicar ipeaplot::theme_ipea().
flip	Logico. Se TRUE (padrao), usa barras horizontais com ggplot2::coord_flip().
show_reference	Logico. Se TRUE (padrao), mostra termos caracteristicos do grupo alvo e do grupo de referencia. Se FALSE, mostra apenas os termos com keyness positivo.

Value

Um objeto ggplot.

See Also

ac_count(), ac_keyness()

Examples

```
# 1. Corpus fabricado com duas subamostras (Governo vs Oposicao)
df <- data.frame(
  id    = c("d1", "d2", "d3", "d4"),
  texto = c(
    "A A A B",
    "A B",
    "A B B B",
    "B B C"
  ),
  lado  = c("Governo", "Governo", "Oposicao", "Oposicao"),
  stringsAsFactors = FALSE
)

# 2. Calcular termos-chave (sobre-representados no grupo alvo)
corp <- ac_corpus(df, text = texto, docid = id, meta = lado)
freq <- ac_count(corp, by = "lado")
key <- ac_keyness(freq, group = "lado", target = "Governo")

# 3. Grafico dos 5 termos com maior chi-quadrado
ac_plot_keyness(key, n = 5)
```

`ac_plot_lda_topics` *Visualizar top termos por tópico*

Description

`ac_plot_lda_topics()` gera um gráfico de barras com os termos de maior probabilidade (beta) para cada tópico do modelo LDA.

Usage

```
ac_plot_lda_topics(lda_result, top_n = 10L, ncol = NULL, title = NULL, ...)
```

Arguments

`lda_result`	Objeto `ac_lda`, saída de `ac_lda()`.
`top_n`	Número de termos por tópico. Padrão: `10`.
`ncol`	Número de colunas nos facets. Padrão: `NULL` (automático).
`title`	Título. Padrão: `NULL`.
`...`	Ignorado.

Value

Objeto `ggplot`.

See Also

`ac_lda()`

Examples

```
# Corpus sintetico com dois blocos tematicos
df <- data.frame(
  id = paste0("d", 1:8),
  texto = c(
    "democracia participacao voto cidadania",
    "cidadania direitos participacao democracia",
    "voto direitos liberdade cidadania",
    "democracia voto participacao popular",
    "mercado economia eficiencia privatizacao",
    "privatizacao mercado livre eficiencia",
    "economia crescimento investimento mercado",
    "eficiencia mercado economia livre"
  ),
  stringsAsFactors = FALSE
)
corpus <- ac_corpus(df, text = texto, docid = id)
lda    <- ac_lda(corpus, k = 2)
```

```
# Grafico com os 5 termos mais probabilisticos por topico
ac_plot_lda_topics(lda, top_n = 5)
```

`ac_plot_lda_tune` *Visualizar curva de seleção de k (perplexidade)*

Description

`ac_plot_lda_tune()` gera um gráfico de linha da perplexidade (ou outras métricas) em função do número de tópicos k, auxiliando na escolha do k ideal para o modelo LDA.

Usage

```
ac_plot_lda_tune(tune_result, title = NULL, ...)
```

Arguments

`tune_result`	Tibble retornado por `ac_lda_tune()`.
`title`	Título. Padrão: `NULL`.
`...`	Ignorado.

Value

Objeto `ggplot`.

See Also

`ac_lda_tune()`

Examples

```
# Corpus sintetico com dois blocos tematicos
df <- data.frame(
  id = paste0("d", 1:8),
  texto = c(
    "democracia participacao voto cidadania",
    "cidadania direitos participacao democracia",
    "voto direitos liberdade cidadania",
    "democracia voto participacao popular",
    "mercado economia eficiencia privatizacao",
    "privatizacao mercado livre eficiencia",
    "economia crescimento investimento mercado",
    "eficiencia mercado economia livre"
  ),
  stringsAsFactors = FALSE
)
corpus <- ac_corpus(df, text = texto, docid = id)
```

```
# Testar k de 2 a 4 e visualizar a curva de perplexidade
# (ponto de inflexao/"cotovelo" sugere um bom k)
tune <- ac_lda_tune(corpus, k_range = 2:4)
ac_plot_lda_tune(tune)
```

`ac_plot_sentiment` *Visualizar sentimento ao longo dos documentos*

Description

`ac_plot_sentiment()` gera visualizações de sentimento: barras por documento, linha temporal, ou distribuição de scores.

Usage

```
ac_plot_sentiment(
  sentiment_tbl,
  type = c("bar", "line", "density"),
  x_col = "doc_id",
  fill_col = "sentiment",
  title = NULL,
  ...
)
```

Arguments

`sentiment_tbl`	Tibble retornado por `ac_sentiment()`.
`type`	Tipo de visualização: `"bar"` (padrão), `"line"`, `"density"`.
`x_col`	Coluna do eixo X. Padrão: `"doc_id"`. Pode ser uma coluna de data para `type = "line"`.
`fill_col`	Coluna para preenchimento/cor. Padrão: `"sentiment"`.
`title`	Título do gráfico. Padrão: `NULL`.
`...`	Ignorado.

Value

Objeto `ggplot`.

See Also

`ac_sentiment()`

Examples

```
# Corpus com quatro documentos de valencias distintas
df <- data.frame(
  id = c("a", "b", "c", "d"),
  texto = c(
    "excelente otimo positivo bom",
    "pessimo terrivel negativo ruim",
    "aprovada proposta reuniao",
    "bom resultado positivo otimo"
  )
)
corpus <- ac_corpus(df, text = texto, docid = id)

# Calcular scores e visualizar em barras (positivo/neutro/negativo por doc)
sent <- ac_sentiment(corpus)
ac_plot_sentiment(sent)
```

`ac_plot_tf_idf` *Plotar termos mais caracteristicos por tf-idf*

Description

`ac_plot_tf_idf()` cria um grafico de barras com os termos mais caracteristicos a partir de uma tabela com a coluna `tf_idf`, tipicamente gerada por `ac_tf_idf()`.

A funcao usa ggplot2 como base e pode, opcionalmente, aplicar o estilo editorial do pacote `ipeaplot`.

Usage

```
ac_plot_tf_idf(
  x,
  by = NULL,
  n = NULL,
  style = c("default", "ipea"),
  flip = TRUE
)
```

Arguments

`x`	Um `data.frame` ou `tibble::tibble()` contendo, no minimo, as colunas `token` e `tf_idf`.
`by`	Vetor de nomes de colunas em `x` a serem usados como grupos de facetas. Se `NULL` (padrao), produz um unico grafico. Se nao for `NULL`, cria facetas por combinacao das colunas informadas.
`n`	Numero de termos a exibir. Se `NULL` (padrao), usa todas as linhas de `x`. Se informado, seleciona os top `n` termos por `tf_idf` no geral ou em cada grupo definido por `by`.

style	Estilo grafico. Pode ser "default" (padrao) ou "ipea". Quando "ipea", a funcao tenta aplicar ipeaplot::theme_ipea().
flip	Logico. Se TRUE (padrao), usa barras horizontais com ggplot2::coord_flip().

Value

Um objeto ggplot.

See Also

ac_count(), ac_tf_idf(), ac_top_terms()

Examples

```
# 1. Corpus de 3 discursos com metadado de partido
df <- data.frame(
  id      = c("d1", "d2", "d3"),
  texto   = c(
    "O deputado do PT falou na CCJ.",
    "O deputado do PL falou novamente.",
    "O senador do PT falou na CCJ."
  ),
  partido = c("PT", "PL", "PT"),
  stringsAsFactors = FALSE
)

# 2. Pipeline padrao: corpus -> frequencia -> tf-idf
corp  <- ac_corpus(df, text = texto, docid = id, meta = partido)
freq  <- ac_count(corp)
tfidf <- ac_tf_idf(freq)

# 3. Grafico dos 10 termos com maior tf-idf (mais distintivos)
ac_plot_tf_idf(tfidf, n = 10)

freq_by <- ac_count(corp, by = "partido")
tfidf_by <- ac_tf_idf(freq_by, by = "partido")
ac_plot_tf_idf(tfidf_by, by = "partido", n = 5)
```

---

ac_plot_top_terms *Plotar termos mais frequentes*

Description

ac_plot_top_terms() cria um grafico de barras com os termos mais frequentes a partir de uma tabela de frequencias, tipicamente gerada por ac_count() ou filtrada por ac_top_terms().

A funcao usa ggplot2 como base e pode, opcionalmente, aplicar o estilo editorial do pacote ipeaplot.

Usage

```
ac_plot_top_terms(
  x,
  by = NULL,
  n = NULL,
  style = c("default", "ipea"),
  flip = TRUE
)
```

Arguments

x	Um data.frame ou tibble::tibble() contendo, no minimo, as colunas token e n.
by	Vetor de nomes de colunas em x a serem usados como grupos de facetas. Se NULL (padrao), produz um unico grafico. Se nao for NULL, cria facetas por combinacao das colunas informadas.
n	Numero de termos a exibir. Se NULL (padrao), usa todas as linhas de x. Se informado, seleciona os top n termos no geral ou em cada grupo definido por by.
style	Estilo grafico. Pode ser "default" (padrao) ou "ipea". Quando "ipea", a funcao tenta aplicar ipeaplot::theme_ipea().
flip	Logico. Se TRUE (padrao), usa barras horizontais com ggplot2::coord_flip().

Value

Um objeto ggplot.

See Also

ac_count(), ac_top_terms()

Examples

```
# 1. Corpus curto com metadado de partido
df <- data.frame(
  id      = c("d1", "d2", "d3"),
  texto   = c(
    "O deputado do PT falou na CCJ.",
    "O deputado do PL falou novamente.",
    "O senador do PT falou na CCJ."
  ),
  partido = c("PT", "PL", "PT"),
  stringsAsFactors = FALSE
)
corp <- ac_corpus(df, text = texto, docid = id, meta = partido)

# 2. Top 10 termos do corpus inteiro
freq <- ac_count(corp)
ac_plot_top_terms(freq, n = 10)
```

```
# 3. Top 5 termos POR partido (facets)
freq_by <- ac_count(corp, by = "partido")
ac_plot_top_terms(freq_by, by = "partido", n = 5)
```

`ac_plot_wordcloud_comparative` *Nuvem de palavras comparativa entre grupos*

Description

`ac_plot_wordcloud_comparative()` gera nuvens de palavras comparativas entre N grupos de documentos, dispostas em facets lado a lado. Usa TF-IDF (calculado tratando cada grupo como um "documento") para identificar os termos mais distintivos de cada grupo.

Aceita 2, 3, 4+ grupos: cada grupo vira uma faceta. Para dois grupos a leitura fica naturalmente lado a lado; para mais, o layout se organiza em uma linha (ou grade, se muitos grupos).

Usage

```
ac_plot_wordcloud_comparative(
  corpus,
  group,
  max_words = 50L,
  colors = NULL,
  title = NULL,
  seed = 42L,
  backend = c("auto", "ggwordcloud", "ggplot"),
  ...
)
```

Arguments

`corpus`	Objeto `ac_corpus` com coluna de metadado de grupo.
`group`	Coluna de agrupamento (nome sem aspas ou string). Deve ter pelo menos 2 valores únicos.
`max_words`	Número máximo de palavras por grupo. Padrão: `50`.
`colors`	Vetor de cores (uma por grupo, na ordem alfabética dos grupos). Padrão: as primeiras N cores de `ac_palette()` (Okabe-Ito).
`title`	Título do gráfico. Padrão: `NULL`.
`seed`	Semente para o posicionamento aleatorio dos termos. Padrao `42L` (garante layout reproduzivel entre chamadas).
`backend`	Motor de renderizacao: `"auto"` (padrao, prefere `ggwordcloud` com facets), `"ggwordcloud"` ou `"ggplot"` (facets com `geom_text` + jitter reproduzivel).
`...`	Ignorado.

Value

Objeto ggplot.

See Also

ac_tf_idf()

Examples

```
# Corpus dividido em dois grupos com vocabulario contrastante
df <- data.frame(
  id     = paste0("d", 1:6),
  texto  = c(
    "democracia participacao popular voto",
    "direitos cidadania liberdade democracia",
    "participacao popular igualdade direitos",
    "mercado economia privatizacao eficiencia",
    "privatizacao mercado livre eficiencia",
    "economia crescimento mercado investimento"
  ),
  grupo = c("A","A","A","B","B","B")
)
corpus <- ac_corpus(df, text = texto, docid = id)

# Nuvem comparativa: termos distintivos de cada grupo
ac_plot_wordcloud_comparative(corpus, group = grupo)
```

---

ac_plot_xray *Gráfico X-ray — dispersão lexical de termos no corpus*

Description

ac_plot_xray() exibe a posição de ocorrência de um ou mais termos ao longo do texto de cada documento, como marcações verticais numa linha horizontal. Útil para visualizar padrões de uso ao longo de discursos, capítulos ou documentos longos.

Usage

```
ac_plot_xray(
  corpus,
  terms,
  ignore_case = TRUE,
  colors = NULL,
  title = NULL,
  ...
)
```

Arguments

corpus	Objeto ac_corpus.
terms	Vetor de termos a rastrear (após limpeza e tokenização).
ignore_case	Se TRUE (padrão), ignora diferenças de capitalização.
colors	Vetor de cores para os termos. Se NULL, usa paleta padrão.
title	Título do gráfico. Padrão: NULL.
...	Ignorado.

Value

Objeto ggplot.

See Also

ac_corpus()

Examples

```
# Dois documentos com repeticao de termos-alvo em posicoes diferentes
df <- data.frame(
  id = c("d1", "d2"),
  texto = c(
    "democracia liberdade igualdade democracia direitos democracia",
    "mercado liberdade privatizacao mercado eficiencia mercado"
  )
)
corpus <- ac_corpus(df, text = texto, docid = id)

# X-ray: pontos marcam a posicao relativa de cada termo dentro do texto
ac_plot_xray(corpus, terms = c("democracia", "mercado", "liberdade"))
```

---

ac_qual_code *Classificar textos com LLM usando um codebook*

Description

ac_qual_code() classifica os textos de um ac_corpus de acordo com um ac_codebook, usando um modelo de linguagem via ellmer. Retorna um tibble com a classificacao, grau de certeza (via self-consistency) e raciocinio da LLM para cada documento.

E o **motor de classificacao** do pipeline qualitativo do acR. Assume que o codebook ja foi construido com ac_qual_codebook() e (idealmente) testado numa amostra piloto. A saida e sempre validada com ac_qual_reliability() contra uma amostra codificada por humano – nenhuma analise categorial publicavel dispensa essa etapa.

Tres parametros determinam qualidade e custo:

- `k_consistency`: numero de rodadas de *self-consistency* (Wang et al., 2023). O mesmo texto e classificado k vezes com pequena variacao de temperatura, e a categoria final e a moda. O `confidence_score` sai dessa concordancia – 1,0 significa que todas as k rodadas concordaram. Padrao `k = 3` custa 3x mais tokens que uma rodada unica, e e a configuracao minima para reportar confianca de forma defensavel.
- `reasoning`: pede raciocinio estruturado. Melhora a qualidade em casos ambiguos mas praticamente dobra o custo por documento. Use `"short"` como padrao e `"detailed"` so quando planeja auditar decisoes individuais.
- `live`: tira a classificacao do modo "caixa-preta" mostrando cada documento em tempo real. Numa rodada de 500 discursos, voce percebe nos primeiros 10 se o codebook esta funcionando – antes de gastar todo o orcamento de tokens.

Usage

```
ac_qual_code(
  corpus,
  codebook,
  model = "anthropic/claude-sonnet-4-5",
  chat = NULL,
  confidence = c("total", "by_variable", "both", "none"),
  k_consistency = 3L,
  temperature = 0.3,
  reasoning = TRUE,
  reasoning_length = c("short", "medium", "detailed"),
  main_temperature = 0,
  reasoning_extras = FALSE,
  live = c("off", "terminal", "progress", "shiny"),
  max_retries = 2L,
  checkpoint_dir = NULL,
  ...
)
```

Arguments

`corpus`	Objeto `ac_corpus`.
`codebook`	Objeto `ac_codebook`, saida de `ac_qual_codebook()`.
`model`	Modelo LLM a usar. Aceita string no formato `"provedor/modelo"` (ex: `"anthropic/claude-sonnet-4` `"openai/gpt-4.1"`), objeto `Chat` do pacote `ellmer` pre-configurado, ou funcao fabrica `function(temperature, system_prompt)` `Chat` para controle total sobre a construcao do chat a cada rodada. Quando `chat` e fornecido, `model` e ignorado.
`chat`	Objeto `Chat` do pacote `ellmer` (ex: `chat_google_gemini()`, `chat_openai()`, `chat_ollama()`). Quando fornecido, tem prioridade sobre `model`. Permite usar qualquer provedor suportado pelo `ellmer`.
`confidence`	Como calcular certeza: • `"total"` (padrao): uma coluna `confidence_score` com a proporcao de rodadas de self-consistency que concordaram com a categoria vencedora (moda);

- "by_variable": uma coluna <categoria>_confidence por categoria do codebook, com a proporcao de rodadas que escolheram *aquela* categoria especificamente (nao so a vencedora) – util para montar uma matriz de confianca por tema/categoria;
- "both": as colunas por categoria + confidence_score (a da vencedora);
- "none": nao calcula certeza (mais rapido, menor custo).

k_consistency Numero de rodadas para self-consistency. Padrao: 3. Ignorado se confidence = "none".

temperature Temperatura das rodadas de consistency. Padrao: 0.3.

reasoning Logico. Se TRUE (padrao), inclui coluna raciocinio com justificativa da classificacao.

reasoning_length
Tamanho do raciocinio: "short" (1 frase, padrao), "medium" (3 frases), "detailed" (paragrafo).

main_temperature
Temperatura da rodada principal (a que gera categoria, categoria_label e raciocinio no resultado final). Padrao 0 (deterministica). Separado de temperature (usado so nas rodadas extras de self-consistency) porque os dois cumprem papeis diferentes: a rodada principal deve ser reprodutivel, as extras devem variar para medir concordancia.

reasoning_extras
Logico. Se TRUE, pede raciocinio tambem nas rodadas extras de self-consistency (mais caro, nao usado no resultado final). Padrao FALSE.

live Visualizacao em tempo real da classificacao:

- "off" (padrao): sem live view;
- "terminal": barra de progresso com doc atual, categoria, confianca e inicio do raciocinio a cada iteracao;
- "progress": barra de progresso enxuta (doc atual e status), sem ecoar categoria/raciocinio – util para rodadas grandes onde o terminal ficaria poluido;
- "shiny": abre janela Shiny em background com tabela atualizando conforme documentos sao classificados (requer shiny e callr).

max_retries Numero maximo de novas tentativas por rodada quando a chamada ao LLM falha (erro de rede, timeout, resposta invalida). Padrao 2. Espera exponencial entre tentativas (ate 30s). Quando o erro e classificado como "estouro_contexto", nao ha nova tentativa (repetir nao resolve). Se mais da metade dos ultimos 5+ documentos falhar, a execucao e abortada com sugestao de usar checkpoint_dir.

checkpoint_dir Diretorio para salvar o resultado de cada documento assim que classificado. Se ja existir um checkpoint valido para um doc_id (mesmos codebook, model, k_consistency e temperature), o documento e pulado e o resultado salvo e reutilizado – permite retomar uma rodada interrompida sem reclassificar tudo. NULL (padrao): sem checkpoint.

... Argumentos adicionais passados a ellmer::chat(). Permite uso de APIs OpenAI-compatible self-hosted via base_url.

Value

Tibble com colunas:

- doc_id: identificador do documento;
- Metadados originais do corpus;
- categoria: slug interno da categoria (chave da lista categories no codebook). Em multilabel = TRUE, string pipe-separada sem espaco ("tecnica|politica").
- categoria_label: rotulo humano correspondente, vindo do campo label de cada categoria (se ausente, repete o slug). Em multilabel, unidos por " | " (espaco dos dois lados).
- confidence_score: grau de certeza da categoria vencedora (0-1). Presente quando confidence e "total" ou "both".
- confidence_level: "alta", "media", "baixa" – interpretacao de confidence_score (Landis & Koch, 1977). Acompanha confidence_score.
- <categoria>_confidence: uma coluna por categoria do codebook (ex.: positivo_confidence, negativo_confidence), com a proporcao de rodadas de self-consistency que escolheram aquela categoria. Presentes quando confidence e "by_variable" ou "both".
- raciocinio: justificativa da classificacao (se reasoning = TRUE).
- status: resultado da rodada principal: "ok", "resposta_invalida" (JSON nao parseavel), "estouro_contexto", "timeout" ou "erro_servidor".
- n_tentativas: numero de tentativas ate obter status = "ok" (ou ate esgotar max_retries).
- tokens_entrada, tokens_saida: tokens de entrada/saida da rodada principal, via chat$get_tokens() (NA se o provedor nao expoe).
- segundos: tempo total da rodada principal, incluindo novas tentativas.
- tokens_por_segundo: tokens_saida / segundos.

O resultado carrega tambem um atributo run_info (ver attributes()) com metadados da execucao: versoes de acR e ellmer, modelo, base_url, parametros efetivos aplicados ao provedor, temperaturas pedidas, k_consistency, hash do codebook e timestamps de inicio/fim.

References

Wang, X. et al. (2023). Self-Consistency Improves Chain of Thought Reasoning in Language Models. *EMNLP*.

Landis, J. R.; Koch, G. G. (1977). The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33(1), 159-174.

Gilardi, F.; Alizadeh, M.; Kubli, M. (2023). ChatGPT Outperforms Crowd Workers for Text-Annotation Tasks. *PNAS*, 120(30).

Examples

```
## Not run:
cb <- ac_qual_codebook(
  name         = "tom",
  instructions = "Classifique o tom do discurso.",
  categories   = list(
```

```
    positivo = list(definition = "Tom propositivo e colaborativo."),
    negativo = list(definition = "Tom critico e confrontacional.")
  )
)

df <- data.frame(
  id    = c("d1", "d2"),
  texto = c("Proponho cooperacao.", "Este governo e um fracasso.")
)
corpus <- ac_corpus(df, text = texto, docid = id)

# Usando string de modelo (comportamento padrao)
coded <- ac_qual_code(corpus, cb, model = "anthropic/claude-sonnet-4-5")

# Usando objeto Chat do ellmer (recomendado para controle fino)
chat_obj <- ellmer::chat_google_gemini(model = "gemini-2.5-flash", echo = "none")
coded <- ac_qual_code(corpus, cb, chat = chat_obj)

# Groq (inferencia rapida, plano gratuito)
chat_groq <- ellmer::chat_groq(model = "openai/gpt-oss-120b", echo = "none")
coded <- ac_qual_code(corpus, cb, chat = chat_groq)

# Ollama (modelos locais, sem envio de dados externos)
chat_local <- ellmer::chat_ollama(model = "llama3.2", echo = "none")
coded <- ac_qual_code(corpus, cb, chat = chat_local)

# confidence = "both": alem de confidence_score (categoria vencedora),
# ganha uma coluna <categoria>_confidence por categoria do codebook --
# util para montar uma matriz de confianca por tema entre documentos
# ou comparar modelos na mesma tarefa.
coded_both <- ac_qual_code(corpus, cb, model = "anthropic/claude-sonnet-4-5",
                           confidence = "both")
coded_both[, c("doc_id", "categoria", "confidence_score",
               "positivo_confidence", "negativo_confidence")]

## End(Not run)
```

---

ac_qual_codebook *Criar um codebook para análise de conteúdo qualitativa*

Description

ac_qual_codebook() cria um livro de códigos estruturado para classificação de textos via LLM. É o **instrumento central** da análise de conteúdo assistida por IA: nenhum resultado publicável de codificação automática dispensa um codebook explícito, versionável e passível de revisão por pares (Krippendorff, 2018).

Um codebook do acR operacionaliza cinco elementos por categoria: definição (o que é), exemplos positivos (o que é, concretamente), exemplos negativos (o que não é, para desambiguar categorias

vizinhas), referências (ancorar a categoria em literatura publicada) e peso (indicação relativa de prioridade no prompt). Um codebook bem construído é a diferença entre a LLM adivinhar (com prior próprio, não replicável) e a LLM **aplicar** uma operacionalização reproduzível (Gilardi et al., 2023).

Suporta três modos de construção:

- `"manual"` (padrão): o pesquisador fornece definições, exemplos e referências diretamente.
- `"induced"`: a LLM induz categorias automaticamente a partir de uma amostra do corpus, sugerindo nomes, definições e exemplos. Útil como ponto de partida quando você não tem um esquema teórico a priori.
- `"literature"`: a LLM busca definições na literatura acadêmica, gerando um banco estruturado com trecho original, tradução, autor, ano, revista e link. O pesquisador revisa e aprova interativamente.

Todo codebook mantém um `history` das modificações — quem alterou, quando e o quê. Essencial para auditoria metodológica quando o instrumento evolui entre a versão piloto e a versão final publicada.

Usage

```
ac_qual_codebook(
  name,
  instructions,
  categories = list(),
  corpus = NULL,
  n_categories = 5L,
  mode = c("manual", "induced", "literature"),
  multilabel = FALSE,
  lang = "pt",
  chat = NULL,
  model = "anthropic/claude-sonnet-4-5",
  journals = "default",
  n_refs = 5L,
  check_overlap = FALSE,
  ...
)
```

Arguments

`name`	Nome identificador do codebook (string).
`instructions`	Instrução geral para a LLM.
`categories`	Lista nomeada de categorias. Cada elemento pode conter: • `definition`: definição operacional da categoria (obrigatório). • `label`: rótulo humano para exibição em relatórios e tabelas (opcional). Se ausente, o `acR` usa o próprio nome da categoria (chave da lista). Exemplo: `label = "Gestão fiscal-administrativa geral"` para uma categoria de nome interno `"gestao_fiscal_administrativa_geral"`. A saída de `ac_qual_code()` ganha uma coluna `categoria_label` traduzida.

- `examples_pos`: vetor de exemplos positivos (recomendado).
- `examples_neg`: vetor de exemplos negativos (recomendado).
- `references`: vetor de referências bibliográficas (opcional).
- `weight`: número entre 0 e 1 indicando a importância relativa da categoria para a LLM (padrão: 1). Categorias raras ou difíceis podem receber peso maior para instrução extra.

`corpus`	Objeto `ac_corpus`. Obrigatório no modo `"induced"`.
`n_categories`	Inteiro. Número de categorias a induzir. Padrão: `5L`.
`mode`	`"manual"` (padrão), `"induced"` ou `"literature"`.
`multilabel`	Lógico. Se `TRUE`, um documento pode pertencer a mais de uma categoria. Padrão: `FALSE`.
`lang`	Idioma do corpus: `"pt"` (padrão) ou `"en"`.
`chat`	Objeto `Chat` do pacote `ellmer`. Tem prioridade sobre `model`.
`model`	Modelo LLM. Padrão: `"anthropic/claude-sonnet-4-5"`.
`journals`	Periódicos para busca de literatura.
`n_refs`	Número de referências por categoria. Padrão: `5`.
`check_overlap`	Se `TRUE`, verifica sobreposição semântica entre definições e avisa o pesquisador. Requer `chat` ou `model`. Padrão: `FALSE`.
`...`	Ignorado.

Value

Objeto de classe `ac_codebook`.

References

Gilardi, F., Alizadeh, M., & Kubli, M. (2023). ChatGPT outperforms crowd workers for text-annotation tasks. *PNAS*, 120(30).

Krippendorff, K. (2018). *Content Analysis: An Introduction to Its Methodology* (4th ed.). SAGE.

Sampaio, R. C., & Lycarião, D. (2021). *Análise de conteúdo categorial: manual de aplicação*. Brasília: ENAP.

Examples

```
# Codebook manual com duas categorias, cada uma com exemplos positivos
# (o que E) e negativos (o que NAO e, para desambiguar categorias vizinhas)
cb <- ac_qual_codebook(
  name         = "tom_discurso",
  instructions = "Classifique o tom geral do discurso.",
  categories   = list(
    positivo = list(
      definition   = "Discurso com tom propositivo e colaborativo.",
      examples_pos = c("Proponho que trabalhemos juntos nesta agenda."),
      examples_neg = c("Este governo e um desastre completo."),
      weight       = 1  # peso relativo no prompt; use >1 para priorizar
    ),
```

```
    negativo = list(
      definition   = "Discurso com tom critico ou confrontacional.",
      examples_pos = c("Esta proposta vai arruinar o pais."),
      examples_neg = c("Apresento esta emenda para melhorar o texto."),
      weight       = 1
    )
  )
)
cb  # imprime resumo do codebook
```

---

ac_qual_codebook_add *Adicionar categoria a um codebook existente*

Description

ac_qual_codebook_add() adiciona uma ou mais categorias a um ac_codebook já criado, sem precisar recriar o objeto do zero. Útil para refinamento iterativo do codebook durante a análise.

Usage

```
ac_qual_codebook_add(codebook, ...)
```

Arguments

codebook	Objeto ac_codebook.
...	Categorias a adicionar, nomeadas. Cada elemento deve ser uma lista com definition e, opcionalmente, examples_pos, examples_neg, weight e references.

Value

Objeto ac_codebook atualizado.

See Also

ac_qual_codebook(), ac_qual_codebook_remove()

Examples

```
# Codebook inicial com 2 categorias
cb <- ac_qual_codebook(
  name         = "tom",
  instructions = "Classifique o tom.",
  categories   = list(
    positivo = list(definition = "Tom propositivo."),
    negativo = list(definition = "Tom critico.")
  )
)
```

```
# Adicionar uma terceira categoria (nome do argumento vira nome da categoria)
cb <- ac_qual_codebook_add(cb,
  neutro = list(
    definition   = "Tom neutro, sem posicionamento claro.",
    examples_pos = c("O projeto foi apresentado na sessao de hoje.")
  )
)
names(cb$categories)  # "positivo" "negativo" "neutro"
```

`ac_qual_codebook_history` *Exibir histórico de modificações de um codebook*

Description

`ac_qual_codebook_history()` retorna e imprime o histórico de ações registradas em um `ac_codebook` (adições, remoções, merges, traduções etc.).

Usage

```
ac_qual_codebook_history(codebook, n = Inf)
```

Arguments

`codebook`	Objeto `ac_codebook`.
`n`	Número máximo de entradas a exibir. Padrão: `Inf` (todas).

Value

Tibble com colunas `timestamp`, `action` e `detail` (invisível).

Examples

```
# Criar codebook e aplicar duas modificacoes
cb <- ac_qual_codebook(
  name         = "sentimento",
  instructions = "Classifique o sentimento.",
  categories   = list(
    positivo = list(definition = "Sentimento positivo."),
    negativo = list(definition = "Sentimento negativo.")
  )
)
cb <- ac_qual_codebook_add(cb,
  neutro = list(definition = "Sem valencia clara.")
)
cb <- ac_qual_codebook_remove(cb, "neutro")

# Ver historico completo de acoes registradas
ac_qual_codebook_history(cb)
```

ac_qual_codebook_hybrid *Enriquecer codebook com literatura via LLM (modo híbrido)*

Description

ac_qual_codebook_hybrid() re-ancora as definições de um ac_codebook existente em referências bibliográficas buscadas via LLM, combinando definições manuais com fundamento teórico induzido da literatura.

Usage

```
ac_qual_codebook_hybrid(
  codebook,
  chat = NULL,
  model = "anthropic/claude-sonnet-4-5",
  concepts = NULL,
  journals = "default",
  n_refs = 3L,
  lang = "pt"
)
```

Arguments

codebook	Objeto ac_codebook.
chat	Objeto Chat do pacote ellmer. Tem prioridade sobre model.
model	Modelo LLM. Padrão: "anthropic/claude-sonnet-4-5".
concepts	Lista nomeada com conceitos por categoria (opcional).
journals	Periódicos para busca. Padrão: "default".
n_refs	Número de referências por categoria. Padrão: 3L.
lang	Idioma: "pt" (padrão) ou "en".

Value

Objeto ac_codebook com definições atualizadas e literatura anexada.

Examples

```
## Not run:
# Requer conexao com a internet + credenciais da LLM (ANTHROPIC_API_KEY).

# 1. Codebook manual como ponto de partida
cb <- ac_qual_codebook(
  name         = "populismo",
  instructions = "Identifique tom populista no discurso.",
  categories   = list(
```

```
    populista     = list(definition = "Apela ao povo contra a elite."),
    nao_populista = list(definition = "Discurso tecnico ou institucional.")
  )
)

# 2. Enriquecer com literatura: busca 3 refs por categoria e reescreve
#    as definicoes com base na literatura recuperada
cb_ancorado <- ac_qual_codebook_hybrid(
  codebook = cb,
  n_refs   = 3L,
  lang     = "pt"
)

# 3. Inspecionar as referencias anexadas a cada categoria
cb_ancorado$categories$populista$references

## End(Not run)
```

`ac_qual_codebook_merge` *Fundir dois codebooks em um*

Description

`ac_qual_codebook_merge()` combina as categorias de dois objetos `ac_codebook` em um único codebook, com controle de conflitos de nomes.

Usage

```
ac_qual_codebook_merge(
  cb1,
  cb2,
  name = NULL,
  on_conflict = c("error", "keep_first", "keep_second", "rename_second"),
  instructions = NULL
)
```

Arguments

`cb1`	Objeto `ac_codebook` (base).
`cb2`	Objeto `ac_codebook` (a fundir).
`name`	Nome do codebook resultante. Padrão: `"cb1_cb2"`.
`on_conflict`	Estratégia em caso de categorias com o mesmo nome: `"error"` (padrão), `"keep_first"`, `"keep_second"` ou `"rename_second"`.
`instructions`	Instrução geral do novo codebook. Se `NULL`, usa a de `cb1`.

Value

Objeto ac_codebook fundido.

Examples

```
# Dois codebooks pequenos que cobrem dimensoes distintas
cb_tom <- ac_qual_codebook(
  name         = "tom",
  instructions = "Classifique o tom.",
  categories   = list(
    positivo = list(definition = "Tom positivo."),
    negativo = list(definition = "Tom negativo.")
  )
)
cb_estilo <- ac_qual_codebook(
  name         = "estilo",
  instructions = "Classifique o estilo retorico.",
  categories   = list(
    pathos = list(definition = "Apelo emocional."),
    logos  = list(definition = "Apelo racional.")
  )
)

# Fundir em um unico codebook com 4 categorias
cb <- ac_qual_codebook_merge(cb_tom, cb_estilo, name = "tom_estilo")
names(cb$categories)  # "positivo" "negativo" "pathos" "logos"
```

---

ac_qual_codebook_remove

Remover categoria de um codebook existente

Description

ac_qual_codebook_remove() remove uma ou mais categorias de um ac_codebook existente.

Usage

```
ac_qual_codebook_remove(codebook, categories)
```

Arguments

codebook	Objeto ac_codebook.
categories	Vetor character com os nomes das categorias a remover.

Value

Objeto ac_codebook atualizado.

See Also

ac_qual_codebook(), ac_qual_codebook_add()

Examples

```
# Codebook com 3 categorias
cb <- ac_qual_codebook(
  name         = "tom",
  instructions = "Classifique o tom.",
  categories   = list(
    positivo = list(definition = "Tom propositivo."),
    negativo = list(definition = "Tom critico."),
    neutro   = list(definition = "Tom neutro.")
  )
)

# Remover a categoria "neutro" (aceita tambem um vetor de nomes)
cb <- ac_qual_codebook_remove(cb, "neutro")
names(cb$categories)  # "positivo" "negativo"
```

---

ac_qual_codebook_translate

Traduzir codebook para outro idioma via LLM

Description

ac_qual_codebook_translate() traduz as instruções, definições e exemplos de um ac_codebook para o idioma alvo usando uma LLM, preservando a estrutura e os metadados do objeto.

Usage

```
ac_qual_codebook_translate(
  codebook,
  to = c("en", "pt"),
  chat = NULL,
  model = "anthropic/claude-sonnet-4-5",
  translate_examples = TRUE
)
```

Arguments

codebook	Objeto ac_codebook.
to	Idioma alvo: "en" (padrão) ou "pt".
chat	Objeto Chat do pacote ellmer. Tem prioridade sobre model.
model	Modelo LLM. Padrão: "anthropic/claude-sonnet-4-5".
translate_examples	Se TRUE (padrão), traduz também os exemplos.

Value

Objeto ac_codebook traduzido.

Examples

```
## Not run:
# Requer credenciais da LLM (ANTHROPIC_API_KEY ou GROQ_API_KEY).

cb_pt <- ac_qual_codebook(
  name         = "polaridade",
  instructions = "Classifique a polaridade do texto.",
  categories   = list(
    favor  = list(definition = "Apoia a proposta.",
                  examples_pos = "Sou totalmente a favor desta reforma."),
    contra = list(definition = "Opoe-se a proposta.",
                  examples_pos = "Esta proposta e um retrocesso.")
  )
)

# Traduzir para ingles preservando estrutura e exemplos
cb_en <- ac_qual_codebook_translate(cb_pt, to = "en")
cb_en$lang  # "en"
cb_en$categories$favor$definition

## End(Not run)
```

---

ac_qual_export_for_review

Exportar amostra para revisão humana em Excel

Description

ac_qual_export_for_review() exporta uma amostra de documentos classificados para um arquivo Excel, com colunas para o codificador humano preencher.

Usage

```
ac_qual_export_for_review(
  sample,
  path = "validacao_humana.xlsx",
  corpus = NULL,
  ...
)
```

Arguments

sample	Tibble, saída de ac_qual_sample().
path	Caminho do arquivo .xlsx. Padrão: "validacao_humana.xlsx".
corpus	Objeto ac_corpus original (opcional). Se fornecido, inclui o texto completo de cada documento na planilha.
...	Ignorado.

Value

Invisível: caminho do arquivo gerado.

Examples

```
if (requireNamespace("openxlsx", quietly = TRUE)) {
  # Amostra de documentos ja classificados pela LLM
  coded <- tibble::tibble(
    doc_id           = paste0("doc_", 1:5),
    categoria        = c("favor", "contra", "favor", "contra", "favor"),
    confidence_score = c(0.6, 0.9, 0.8, 0.7, 0.55)
  )
  amostra <- ac_qual_sample(coded, n = 3, strategy = "uncertainty")

  # Exportar para revisao humana em arquivo temporario
  arquivo <- tempfile(fileext = ".xlsx")
  ac_qual_export_for_review(amostra, path = arquivo)
  file.exists(arquivo)
}
```

---

ac_qual_import_human *Importar classificação humana de Excel*

Description

ac_qual_import_human() importa um arquivo Excel preenchido por um codificador humano, retornando um tibble compatível com ac_qual_reliability().

Usage

```
ac_qual_import_human(
  path,
  cat_col = "categoria_humano",
  id_col = "doc_id",
  ...
)
```

Arguments

path	Caminho do arquivo .xlsx.
cat_col	Nome da coluna com a classificação humana. Padrão: "categoria_humano".
id_col	Nome da coluna de identificador. Padrão: "doc_id".
...	Ignorado.

Value

Tibble com colunas doc_id e categoria.

Examples

```
if (requireNamespace("openxlsx", quietly = TRUE)) {
  # Simular uma planilha ja preenchida pelo codificador humano
  arquivo <- tempfile(fileext = ".xlsx")
  openxlsx::write.xlsx(
    data.frame(
      doc_id           = paste0("doc_", 1:3),
      categoria_humano = c("favor", "contra", "favor")
    ),
    arquivo
  )

  # Importar de volta para o R para uso com ac_qual_reliability()
  humano <- ac_qual_import_human(arquivo)
  humano
}
```

---

ac_qual_irr *Calcular métricas de confiabilidade inter-anotador*

Description

Compara as classificações de dois ou mais anotadores (humanos ou LLMs) e retorna métricas padronizadas de concordância. Suporta Cohen's Kappa (dois anotadores), Fleiss' Kappa (multi-anotador), Krippendorff's Alpha e percentual de concordância simples.

Complementa ac_qual_reliability(): aquela compara **um par** LLM × humano com bootstrap; esta é o motor genérico para **qualquer par ou painel** de anotadores. Use-a diretamente quando você tem duas rodadas humanas para calibrar entre codificadores antes de acender a LLM, ou quando quer comparar LLMs entre si sobre a mesma amostra.

A função aceita dois formatos de entrada: (a) dois data.frames com colunas id e categoria, representando anotador 1 e anotador 2; ou (b) um único data.frame em formato largo com uma coluna por anotador.

Usage

```
ac_qual_irr(
  gold,
  predicted,
  method = "all",
  id_col = "id_discurso",
  cat_col = "categoria",
  weight = "unweighted",
  conf_level = 0.95,
  verbose = TRUE
)
```

Arguments

gold	data.frame. Anotacoes de referencia (anotador humano ou gold standard). Deve conter colunas id_discurso (ou id) e categoria.
predicted	data.frame. Anotacoes a comparar (ex.: saida do LLM via ac_qual_code()). Mesmas colunas exigidas.
method	character. Metrica(s) a calcular. Opcoes: "all" (padrao), "cohen_kappa", "fleiss_kappa", "krippendorff", "percent_agreement". Aceita vetor de multiplas opcoes.
id_col	character. Nome da coluna de identificador nos data.frames. Padrao: "id_discurso".
cat_col	character. Nome da coluna de categoria nos data.frames. Padrao: "categoria".
weight	character. Tipo de ponderacao para Cohen's Kappa: "unweighted" (padrao), "linear", "squared". Ignorado para categorias nominais sem ordem natural.
conf_level	numeric. Nivel de confianca para intervalos (0-1). Padrao: 0.95.
verbose	logical. Se TRUE (padrao), imprime resumo formatado.

Details

Interpretacao do Kappa (Landis & Koch, 1977):

Kappa	Concordancia
< 0.00	Pobre
0.00 - 0.20	Leve
0.21 - 0.40	Razoavel
0.41 - 0.60	Moderada
0.61 - 0.80	Substancial
0.81 - 1.00	Quase perfeita

Fleiss' Kappa:

Extensao do Kappa de Cohen para mais de dois anotadores. Requer que predicted contenha uma coluna por anotador adicional, ou que sejam passados como lista via ....

Krippendorff's Alpha:

Metrica mais geral: funciona com qualquer numero de anotadores, lida com dados faltantes e suporta escalas nominais, ordinais e de intervalo (KRIPPENDORFF, 2018).

Value

Um objeto de classe `ac_irr` (lista) com os elementos:

`metrics` `data.frame` com colunas `metric`, `estimate`, `ci_lower`, `ci_upper`, `interpretation`.

`confusion` `table`. Matriz de confusao entre os anotadores.

`n_docs` `integer`. Numero de documentos comparados.

`n_annotators` `integer`. Numero de anotadores.

`categories` `character`. Categorias encontradas.

`method` `character`. Metrica(s) calculadas.

References

KRIPPENDORFF, K. **Content Analysis: An Introduction to Its Methodology**. 4. ed. Thousand Oaks: SAGE, 2018.

LANDIS, J. R.; KOCH, G. G. The measurement of observer agreement for categorical data. **Biometrics**, v. 33, n. 1, p. 159-174, 1977.

See Also

`ac_qual_code()`, `ac_qual_sample()`

Examples

```
# Comparar LLM vs. anotador humano
humano <- data.frame(
  id_discurso = c("d1", "d2", "d3", "d4", "d5"),
  categoria   = c("progressista", "conservador", "tecnocratico",
                  "progressista", "conservador")
)

llm <- data.frame(
  id_discurso = c("d1", "d2", "d3", "d4", "d5"),
  categoria   = c("progressista", "conservador", "progressista",
                  "progressista", "conservador")
)

resultado <- ac_qual_irr(gold = humano, predicted = llm)
print(resultado)

# So Cohen's Kappa
ac_qual_irr(humano, llm, method = "cohen_kappa")
```

`ac_qual_list_models` *Listar modelos LLM disponíveis para análise de conteúdo*

Description

`ac_qual_list_models()` retorna um tibble com os modelos LLM disponíveis para uso com `ac_qual_code()`, incluindo informações de custo, janela de contexto e compatibilidade com análise de conteúdo qualitativa em Ciências Sociais.

Dois modos de operação:

- `live = FALSE` (padrão): usa banco interno curado, funciona offline.
- `live = TRUE`: consulta a API do provedor via `ellmer::models_*()` para obter a lista mais atualizada. Requer chave de API configurada.

Usage

```
ac_qual_list_models(
  provider = "all",
  filter = NULL,
  sort_by = c("cost", "name", "context"),
  live = FALSE,
  ...
)
```

Arguments

`provider`	Provedor(es) a listar. Pode ser `"all"` (padrão) ou um ou mais de: `"anthropic"`, `"openai"`, `"google"`, `"groq"`, `"deepseek"`, `"mistral"`, `"ollama"`.
`filter`	String para filtrar modelos por nome ou ID (ex: `"claude"`, `"gpt-4"`). Padrão: `NULL` (sem filtro).
`sort_by`	Como ordenar os resultados: `"cost"` (padrão, menor custo primeiro), `"name"`, `"context"` (maior janela de contexto primeiro).
`live`	Lógico. Se `TRUE`, consulta a API do provedor ao vivo via `ellmer::models_*()`. Requer chave de API. Padrão: `FALSE`.
`...`	Ignorado.

Value

Tibble com colunas:

- `provider`: nome do provedor;
- `model_id`: identificador do modelo para uso em `ac_qual_code()`;
- `name`: nome legível;
- `context_k`: janela de contexto em milhares de tokens;
- `cost_input`: custo por 1M tokens de entrada (USD), `NA` se gratuito/local;

- `cost_output`: custo por 1M tokens de saída (USD);
- `tier`: categoria (`"frontier"`, `"balanced"`, `"fast"`, `"free"`, `"local"`);
- `pt_support`: suporte estimado ao português (`"alto"`, `"medio"`, `"baixo"`);
- `acr_string`: string pronta para uso em `model = ...` no `acR`.

See Also

`ac_qual_recommend_model()`, `ac_qual_code()`

Examples

```
# Listar todos os modelos do banco interno
ac_qual_list_models()

# Só modelos Anthropic
ac_qual_list_models(provider = "anthropic")

# Modelos baratos com suporte a PT
ac_qual_list_models(sort_by = "cost") |>
  dplyr::filter(pt_support == "alto", cost_input < 1)
```

`ac_qual_load_codebook` *Carregar codebook de arquivo YAML*

Description

Le um arquivo YAML gerado por `ac_qual_save_codebook()` e reconstroi o objeto `ac_codebook` na sessao atual. Uso tipico: retomar uma analise iniciada em outra sessao ou por outro pesquisador.

Usage

```
ac_qual_load_codebook(path, ...)
```

Arguments

`path`	Caminho do arquivo `.yaml`.
`...`	Ignorado.

Value

Objeto `ac_codebook`.

Examples

```
# Preparar um codebook e salvar em arquivo temporario
cb <- ac_qual_codebook(
  name         = "polaridade",
  instructions = "Classifique a polaridade do texto.",
  categories   = list(
    favor   = list(definition = "Apoia a proposta."),
    contra  = list(definition = "Opoe-se a proposta.")
  )
)
arquivo <- tempfile(fileext = ".yaml")
ac_qual_save_codebook(cb, path = arquivo)

# Recarregar em outra sessao
cb_novo <- ac_qual_load_codebook(arquivo)
names(cb_novo$categories)  # "favor" "contra"
```

`ac_qual_recommend_model`

Recomendar modelo LLM para análise de conteúdo qualitativa

Description

`ac_qual_recommend_model()` sugere o(s) modelo(s) mais adequado(s) para uma tarefa específica de análise de conteúdo qualitativa, considerando custo, desempenho em português e tipo de tarefa.

As recomendações são baseadas em benchmarks de classificação de texto em Ciências Sociais (Gilardi et al., 2023; Törnberg, 2023; Alizadeh et al., 2023) e na experiência prática com corpora em português brasileiro.

Usage

```
ac_qual_recommend_model(
  task = c("coding", "literature", "both"),
  budget = c("medium", "low", "high", "free"),
  lang = "pt",
  local = FALSE,
  n = 3L,
  ...
)
```

Arguments

`task`	Tipo de tarefa: • `"coding"` (padrão): classificação de textos com codebook existente; • `"literature"`: geração de definições e busca de referências; • `"both"`: ambas as tarefas.

budget	Orçamento disponível: • "free": apenas modelos gratuitos ou locais; • "low": até USD 1/1M tokens de entrada; • "medium": até USD 5/1M tokens (padrão); • "high": sem restrição de custo.
lang	Idioma predominante do corpus: "pt" (padrão) ou "en".
local	Lógico. Se TRUE, prioriza modelos locais (Ollama). Padrão: FALSE.
n	Número de recomendações a retornar. Padrão: 3.
...	Ignorado.

Value

Tibble com as colunas de ac_qual_list_models() mais:

- rank: posição na recomendação;
- score: pontuação composta (0-100);
- justificativa: texto explicando por que o modelo foi recomendado.

References

Gilardi, F.; Alizadeh, M.; Kubli, M. (2023). ChatGPT Outperforms Crowd Workers for Text-Annotation Tasks. *PNAS*, 120(30).

Tornberg, P. (2023). ChatGPT-4 Outperforms Experts and Crowd Workers in Annotating Political Twitter Messages with Zero-Shot Learning. *PLOS ONE*, 18(4).

Alizadeh, M. et al. (2023). Open-Source LLMs for Text Annotation: A Practical Guide for Model Setting and Fine-Tuning. *arXiv*, 2307.02179.

See Also

ac_qual_list_models(), ac_qual_code()

Examples

```
# Recomendação padrão para classificacao em PT com orcamento medio
ac_qual_recommend_model()

# Opcao gratuita para explorar
ac_qual_recommend_model(budget = "free")

# Local (Ollama) para dados sigilosos
ac_qual_recommend_model(local = TRUE)
```

`ac_qual_reliability` *Calcular confiabilidade entre codificação LLM e humana*

Description

`ac_qual_reliability()` calcula métricas de concordância entre a classificação feita pela LLM e uma classificação humana de referência, com intervalos de confiança via bootstrap.

É a **última porta antes de publicar** um estudo de análise de conteúdo assistida por LLM. Nenhuma referência metodológica atual (Krippendorff, 2018; Gwet, 2014; Gilardi et al., 2023) aceita rotulagem automática sem um subconjunto codificado por humano e concordância documentada.

Quatro métricas são computadas por padrão, cada uma respondendo a uma pergunta diferente:

- **Percent agreement** (`percent_agreement`): fração de documentos em que LLM e humano concordam. Fácil de reportar, mas ignora concordância por acaso — infla para categorias binárias com distribuição desigual.
- **Alpha de Krippendorff** (`krippendorff`): padrão-ouro em análise de conteúdo. Corrige por acaso, aceita missings, funciona com qualquer número de categorias. Valores > 0,80 = quase perfeita; > 0,67 = aceitável para publicação; < 0,60 = codebook precisa ser refinado.
- **AC1 de Gwet** (`gwet_ac1`): alternativa a Krippendorff mais estável quando as categorias são muito desbalanceadas (o "paradoxo do kappa" — kappa despenca com alta concordância se uma categoria domina).
- **F1 macro** (`f1_macro`): média não ponderada dos F1 por categoria. Útil quando você quer garantir que a LLM está indo bem também nas categorias raras, não só nas frequentes.

Intervalos de confiança de 95% vêm de *bootstrap* não paramétrico (padrão: 1000 réplicas). Reporte pelo menos Krippendorff e F1 macro no artigo, com os ICs; discuta divergências humano × LLM em amostra de casos representativos.

Usage

```
ac_qual_reliability(
  llm,
  human,
  cat_col = "categoria",
  metrics = c("krippendorff", "gwet_ac1", "f1_macro", "percent_agreement"),
  bootstrap = 1000L,
  ci_level = 0.95,
  by_variable = FALSE,
  exclude_status = c("estouro_contexto", "timeout", "erro_servidor", "parcial"),
  ...
)
```

Arguments

`llm`	Tibble com classificação LLM, saída de `ac_qual_code()`.
`human`	Tibble com classificação humana, saída de `ac_qual_import_human()`.

cat_col	Nome da coluna de categoria. Padrão: "categoria".
metrics	Vetor de métricas a calcular. Padrão: c("krippendorff", "gwet_ac1", "f1_macro", "percent_agreement").
bootstrap	Número de amostras bootstrap para IC. Padrão: 1000.
ci_level	Nível de confiança do IC. Padrão: 0.95.
by_variable	Lógico. Se TRUE, calcula também métricas binárias (concordância, precisão, recall, F1, Gwet AC1, alfa de Krippendorff) para cada categoria do codebook individualmente, tratando a presença de cada categoria como um problema sim/não — essencial quando multilabel = TRUE, onde um único categoria ("a|b") mistura várias decisões binárias independentes que as métricas agregadas (acima) não conseguem diagnosticar separadamente. Denominadores zero (ex.: uma categoria nunca escolhida por nenhum dos dois lados) geram NA, nunca 0 ou 1 artificiais. Padrão FALSE.
exclude_status	Vetor de valores da coluna status (ver ac_qual_code()) a excluir da comparação antes de calcular as métricas por variável — documentos que falharam na classificação não têm uma categoria confiável para comparar. Ignorado se llm não tiver coluna status. Padrão: c("estouro_contexto", "timeout", "erro_servidor", "parcial").
...	Ignorado.

Value

Tibble com colunas: metric, estimate, ci_lower, ci_upper, interpretation. Quando by_variable = TRUE, o resultado carrega também um atributo by_variable: um tibble com uma linha por categoria do codebook e colunas categoria, n, vp, fp, fn, vn, concordancia, precisao, recall, f1, gwet_ac1, krippendorff_alfa.

References

Krippendorff, K. (2018). *Content Analysis: An Introduction to Its Methodology* (4th ed.). SAGE.

Gwet, K. L. (2014). *Handbook of Inter-Rater Reliability* (4th ed.). Advanced Analytics.

Landis, J. R.; Koch, G. G. (1977). The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33(1), 159-174.

Examples

```
# Simular saidas: LLM (coded) e revisao humana (humano_df)
# ambos com as colunas doc_id + categoria
coded <- tibble::tibble(
  doc_id    = paste0("d", 1:5),
  categoria = c("favor", "contra", "favor", "contra", "favor")
)
humano_df <- tibble::tibble(
  doc_id    = paste0("d", 1:5),
  categoria = c("favor", "contra", "contra", "contra", "favor")
)

# Calcular metricas de confiabilidade (bootstrap curto so para demonstrar;
```

```
# em uso real, deixe o padrao de 1000 replicas)
rel <- ac_qual_reliability(
  llm       = coded,
  human     = humano_df,
  bootstrap = 50
)
print(rel)
```

`ac_qual_report` *Gerar relatório de replicabilidade da análise qualitativa*

Description

`ac_qual_report()` gera um documento estruturado, pronto para artigo ou relatório, com todas as decisões metodológicas da rodada de codificação qualitativa: codebook completo, histórico de modificações, configuração da LLM, distribuição de resultados, métricas de confiabilidade e referências bibliográficas.

É a resposta do `acR` a um risco real de comunicação em análises assistidas por LLM: **o revisor ou leitor não consegue reproduzir a rodada** sem saber exatamente qual modelo, qual codebook, quais parâmetros e qual amostra foram usados. Sem esse relatório, o leitor tem que confiar no autor — algo que a tradição de análise de conteúdo (Krippendorff, 2018) sempre rejeitou. O documento gerado é autocontido e pode ser anexado como material suplementar do artigo, publicado como apêndice ou depositado num repositório de dados junto com o codebook em YAML.

Suporta saída em Markdown (`.md`) ou HTML autocontido (`.html`), em português ou inglês (para submissões internacionais).

Usage

```
ac_qual_report(
  coded,
  codebook,
  reliability = NULL,
  chat = NULL,
  title = NULL,
  author = NULL,
  method = NULL,
  format = c("md", "html"),
  path = NULL,
  lang = c("pt", "en")
)
```

Arguments

`coded`	Tibble com resultado de `ac_qual_code()`.
`codebook`	Objeto `ac_codebook` usado na classificacao.

reliability	Opcional. Saida de ac_qual_reliability(); se fornecido, adiciona secao de confiabilidade inter-codificador.
chat	Opcional. Objeto Chat do ellmer (ou string de modelo); se fornecido, extrai provedor/modelo/parametros para o relatorio.
title	Titulo do relatorio. Padrao: gerado a partir do nome do codebook.
author	Autor(es) do estudo (opcional).
method	Descricao livre do metodo de coleta do corpus (opcional).
format	Formato de saida: "md" (padrao) ou "html".
path	Caminho do arquivo destino. Se NULL, usa tempfile().
lang	Idioma: "pt" (padrao) ou "en".

Value

Invisivel: caminho do arquivo gerado.

Examples

```
# Simular resultado de ac_qual_code para o exemplo
cb <- ac_qual_codebook(
  name         = "polaridade",
  instructions = "Classifique a polaridade do texto.",
  categories   = list(
    favor  = list(definition = "Apoio a proposta."),
    contra = list(definition = "Oposicao a proposta.")
  )
)

coded <- tibble::tibble(
  doc_id           = paste0("d", 1:5),
  categoria        = c("favor", "contra", "favor", "contra", "favor"),
  confidence_score = c(1.00, 0.67, 1.00, 1.00, 0.67),
  reasoning        = rep("...", 5)
)

# Gerar relatorio em markdown temporario
arquivo <- tempfile(fileext = ".md")
ac_qual_report(coded, cb, path = arquivo, author = "Fulano de Tal")
# readLines(arquivo, n = 20)
```

---

ac_qual_report_full *Gerar relatório consolidado de múltiplas variáveis de conteúdo*

Description

ac_qual_report_full() gera **um único** documento (Markdown ou HTML) consolidando resultados de N variáveis de codificação qualitativa — cada uma com seu próprio codebook, rodada de ac_qual_code() e, opcionalmente, métricas de ac_qual_reliability() ou ac_qual_irr().

É a resposta natural ao caso de uso mais comum de análise de conteúdo categorial (Krippendorff, 2018): um estudo tem várias variáveis de conteúdo (tom, posicionamento, tema, framing...). Antes desta função, o pesquisador precisava chamar ac_qual_report() N vezes e emendar os relatórios manualmente, ou escrever um consolidador ad hoc por projeto.

Aceita variáveis multilabel = TRUE e multilabel = FALSE no mesmo relatório — cada seção herda a apresentação apropriada para o tipo.

Usage

```
ac_qual_report_full(
  variables,
  chat = NULL,
  title = NULL,
  author = NULL,
  method = NULL,
  format = c("md", "html"),
  path = NULL,
  lang = c("pt", "en")
)
```

Arguments

variables	Lista **nomeada** onde cada elemento é uma lista com: • coded: tibble com resultado de ac_qual_code() (obrigatório). • codebook: objeto ac_codebook (obrigatório). • reliability: opcional, saída de ac_qual_reliability() ou ac_qual_irr(). Os nomes da lista viram os títulos das seções do relatório.
chat	Opcional. Objeto Chat do ellmer ou string de modelo "provider/model". Aplica-se a todas as variáveis (assumindo que todas foram classificadas com o mesmo modelo — se não, deixe NULL).
title	Título do relatório consolidado. Padrão: "Relatório consolidado de análise de conteúdo".
author	Autor(es) do estudo.
method	Descrição do método de coleta do corpus.
format	"md" (padrão) ou "html".
path	Caminho do arquivo destino. Se NULL, usa tempfile().
lang	"pt" (padrão) ou "en".

Value

Invisível: caminho do arquivo gerado.

See Also

ac_qual_report() para relatório de uma única variável.

Examples

```
# Duas variaveis de conteudo, cada uma com seu codebook
cb_tom <- ac_qual_codebook(
  name = "tom",
  instructions = "Classifique o tom.",
  categories = list(
    pos = list(definition = "Positivo.", label = "Positivo"),
    neg = list(definition = "Negativo.", label = "Negativo")
  )
)
cb_pos <- ac_qual_codebook(
  name = "posicao",
  instructions = "Classifique a posicao.",
  categories = list(
    favor  = list(definition = "A favor.",  label = "A favor"),
    contra = list(definition = "Contra.",   label = "Contra")
  )
)

coded_tom <- tibble::tibble(
  doc_id = paste0("d", 1:3),
  categoria = c("pos", "neg", "pos"),
  confidence_score = c(1, 0.67, 1)
)
coded_pos <- tibble::tibble(
  doc_id = paste0("d", 1:3),
  categoria = c("favor", "contra", "favor"),
  confidence_score = c(1, 1, 0.67)
)

arquivo <- tempfile(fileext = ".md")
ac_qual_report_full(
  variables = list(
    "Tom do discurso" = list(coded = coded_tom, codebook = cb_tom),
    "Posicao politica" = list(coded = coded_pos, codebook = cb_pos)
  ),
  path = arquivo,
  author = "Fulano de Tal"
)
```

---

ac_qual_sample *Amostrar documentos para validação humana*

Description

ac_qual_sample() seleciona uma amostra de documentos classificados pela LLM para validação por um codificador humano, usando diferentes estratégias para maximizar a eficiência da validação.

Usage

```
ac_qual_sample(
  coded,
  n = 50L,
 strategy = c("uncertainty", "stratified", "random", "disagreement", "proportional"),
  strata = NULL,
  min_per_stratum = 1L,
  min_stratum_size = 10L,
  exclude_ids = NULL,
  seed = 42L,
  ...
)
```

Arguments

coded	Tibble com classificação LLM, saída de ac_qual_code().
n	Número de documentos a amostrar. Padrão: 50.
strategy	Estratégia de amostragem: • "uncertainty": prioriza documentos com menor confidence_score (maior incerteza da LLM); • "stratified": garante representação proporcional de todas as categorias; • "random": amostra aleatória simples; • "disagreement": prioriza documentos onde rodadas de self-consistency divergiram (requer confidence_score < 1); • "proportional": amostragem estratificada com alocação proporcional ao tamanho de cada estrato (método do maior resto), respeitando um mínimo por estrato. Diferente de "stratified" (que divide n igualmente entre categorias), preserva a distribuição original de strata na amostra — a escolha correta quando a amostra alimenta uma estimativa populacional (ex.: ac_qual_reliability() ponderada), não só uma checagem qualitativa de cada categoria.
strata	Nome da coluna usada para estratificar quando strategy = "proportional". Obrigatório nesse caso.
min_per_stratum	Mínimo de documentos por estrato na amostra proporcional, mesmo que a alocação exata dê menos. Padrão 1.
min_stratum_size	Estratos da população com menos documentos que isso são agrupados em "Outras" antes de alocar — evita alocar uma fração ínfima da amostra a uma categoria com 2 documentos na população inteira. Padrão 10.

exclude_ids	Vetor de doc_id a excluir do universo amostrado (ex.: documentos já validados em uma rodada anterior). Padrão NULL.
seed	Semente para reprodutibilidade. Padrão: 42.
...	Ignorado.

Value

Tibble com os documentos selecionados, incluindo uma coluna sample_reason indicando por que cada documento foi selecionado. Com strategy = "proportional", ganha também peso_amostral (n_populacao_estrato / n_amostra_estrato) para uso em estimativas ponderadas.

Examples

```
# Simular saida de ac_qual_code() com 20 documentos ja classificados
set.seed(1)
coded <- tibble::tibble(
  doc_id           = paste0("doc_", 1:20),
  categoria        = sample(c("favor", "contra"), 20, replace = TRUE),
  confidence_score = runif(20, 0.5, 1.0)
)

# Priorizar casos mais incertos para revisao humana
ac_qual_sample(coded, n = 5, strategy = "uncertainty")

# Garantir cobertura proporcional das duas categorias
ac_qual_sample(coded, n = 6, strategy = "stratified")
```

---

ac_qual_save_codebook *Salvar codebook em arquivo YAML*

Description

Serializa um objeto ac_codebook em disco no formato YAML, preservando instrucoes, categorias, exemplos, referencias e historico de modificacoes. YAML foi escolhido por ser legivel por humanos e versionavel em Git.

Usage

```
ac_qual_save_codebook(codebook, path = NULL, ...)
```

Arguments

codebook	Objeto ac_codebook.
path	Caminho do arquivo .yaml. Se NULL, deriva do nome do codebook.
...	Ignorado.

Value

Invisivel: caminho do arquivo gerado.

Examples

```
# Criar um codebook simples
cb <- ac_qual_codebook(
  name         = "tom_discurso",
  instructions = "Classifique o tom do discurso.",
  categories   = list(
    positivo = list(definition = "Tom propositivo e colaborativo."),
    negativo = list(definition = "Tom critico e confrontacional.")
  )
)

# Salvar em arquivo temporario (usar caminho real fora de exemplos)
arquivo <- tempfile(fileext = ".yaml")
ac_qual_save_codebook(cb, path = arquivo)

# Reabrir para conferir
cb_recarregado <- ac_qual_load_codebook(arquivo)
identical(cb$categories, cb_recarregado$categories)
```

---

ac_qual_search_literature

Buscar referencias bibliograficas sobre um conceito via OpenAlex e LLM

Description

`ac_qual_search_literature()` busca referencias academicas reais na API do OpenAlex e usa um modelo de linguagem via `ellmer` para sintetizar os abstracts em portugues. Retorna um tibble com metadados bibliograficos verificados e definicoes sintetizadas pela LLM.

A arquitetura e: OpenAlex recupera registros reais (autor, ano, DOI, abstract, revista, numero de citacoes); a LLM sintetiza o abstract em portugues e extrai o trecho mais relevante. Isso evita alucinacoes bibliograficas comuns quando a LLM opera sem fonte externa.

Usage

```
ac_qual_search_literature(
  concept,
  chat = NULL,
  model = "anthropic/claude-sonnet-4-5",
  n_refs = 5L,
  journals = "default",
  lang = "pt",
  min_citations = 0L,
```

```
  ...
)
```

Arguments

concept	String. Conceito ou termo teorico a buscar (ex: "democratic backsliding", "state capacity").
chat	Objeto Chat do pacote ellmer (ex: chat_google_gemini(), chat_openai(), chat_ollama()). Quando fornecido, tem prioridade sobre model.
model	String no formato "provedor/modelo". Ignorado quando chat e fornecido.
n_refs	Inteiro. Numero de referencias a retornar. Padrao: 5.
journals	Periodicos a considerar. Opcoes: • "default": lista curada de periodicos de referencia em CP/CS/AP; • "all": sem restricao de periodico; • Vetor de strings: lista customizada (ex: c("default", "RBCS")).
lang	Idioma das definicoes sintetizadas. Padrao: "pt".
min_citations	Inteiro. Numero minimo de citacoes. Padrao: 0.
...	Argumentos adicionais passados a ellmer::chat().

Value

Tibble com colunas: conceito, autor, ano, revista, n_citacoes, trecho_original, definicao_pt, abstract_original, link.

References

Priem, J. et al. (2022). OpenAlex: A fully-open index of the global research system. *arXiv*, 2205.01833.

Gilardi, F.; Alizadeh, M.; Kubli, M. (2023). ChatGPT Outperforms Crowd Workers for Text-Annotation Tasks. *PNAS*, 120(30).

Examples

```
## Not run:
# Requer internet (busca no OpenAlex) e credenciais da LLM (para sintese)

# Chat via Groq (plano gratuito com gpt-oss-120b)
chat_obj <- ellmer::chat_groq(model = "openai/gpt-oss-120b", echo = "none")

# Buscar 5 referencias mais citadas sobre o conceito e sintetizar cada uma
lit <- ac_qual_search_literature(
  concept       = "democratic backsliding",
  n_refs        = 5,
  min_citations = 50,  # filtra papers com pouca reverberacao
  chat          = chat_obj
)
```

```
# Colunas principais do resultado
print(lit[, c("autor", "ano", "revista", "n_citacoes", "definicao_pt")])

## End(Not run)
```

`ac_sentiment` *Análise de sentimento com OpLexicon*

Description

`ac_sentiment()` calcula a polaridade de sentimento dos documentos de um `ac_corpus` usando o OpLexicon (Souza & Vieira, 2012), retornando pontuações por documento e, opcionalmente, por grupo ou janela temporal.

Usage

```
ac_sentiment(
  corpus,
  by = NULL,
  lexicon = c("oplexicon"),
  method = c("sum", "mean", "ratio"),
  ...
)
```

Arguments

`corpus`	Objeto `ac_corpus`.
`by`	Coluna(s) de agrupamento para agregar o sentimento além do documento (ex: `"partido"`, `"data"`). Padrão: `NULL` (por documento).
`lexicon`	Léxico a usar. Atualmente apenas `"oplexicon"` (padrão).
`method`	Método de agregação por documento: • `"sum"` (padrão): soma das polaridades. • `"mean"`: média das polaridades. • `"ratio"`: razão entre positivos e negativos.
`...`	Ignorado.

Value

Tibble com colunas:

- `doc_id`: identificador do documento;
- Colunas de metadado (se `by` especificado);
- `n_pos`: número de tokens positivos;
- `n_neg`: número de tokens negativos;
- `n_neu`: número de tokens neutros;
- `score`: pontuação de sentimento (método escolhido);
- `sentiment`: classificação (`"positivo"`, `"negativo"`, `"neutro"`).

References

Souza, M.; Vieira, R. (2012). Sentiment Analysis on Twitter Data for Portuguese Language. *PROPOR*.

Souza, M.; Vieira, R.; Busetti, D.; Chishman, R.; Alves, I. M. (2011). Construction of a Portuguese Opinion Lexicon from multiple resources. *STIL/SBC*.

See Also

`ac_plot_sentiment()`

Examples

```
# Discursos com valencia afetiva variada
df <- data.frame(
  id = paste0("d", 1:6),
  texto = c(
    "A reforma tributaria e um passo excelente para o pais",
    "Esta proposta e um desastre, um retrocesso terrivel",
    "O relator apresentou o parecer na sessao ordinaria",
    "Comemoramos essa vitoria historica com muita alegria",
    "Rejeitamos com veemencia essa medida injusta e ilegal",
    "A comissao encerra os trabalhos as 18h"
  ),
  partido = rep(c("A", "B"), 3),
  stringsAsFactors = FALSE
)
corpus <- ac_corpus(df, text = texto, docid = id)

# Polaridade agregada por documento (soma de scores OpLexicon)
ac_sentiment(corpus)

# Agregado por grupo (aqui, por partido)
ac_sentiment(corpus, by = "partido")

# Metodo alternativo: razao positivos/negativos
ac_sentiment(corpus, method = "ratio")
```

`ac_tf_idf` *Calcular tf-idf para termos em documentos ou grupos*

Description

`ac_tf_idf()` calcula a frequencia de termos (`tf`), a frequencia inversa de documentos (`idf`) e o produto `tf_idf` a partir de uma tabela de frequencias de termos (tipicamente o resultado de `ac_count()`).

A funcao segue a mesma logica de `tidytext::bind_tf_idf()`, mas adaptada para o fluxo de trabalho do pacote:

- sem grupos (by = NULL): cada doc_id e tratado como um documento;
- com grupos (by = c("partido", ...)): cada combinacao de metadados e tratada como um "documento" no calculo de idf.

Usage

```
ac_tf_idf(x, by = NULL)
```

Arguments

x	Um data.frame ou tibble::tibble() contendo, no minimo, as colunas token e n. Em geral, o resultado de ac_count().
by	Vetor de nomes de colunas em x que identificam documentos ou grupos. Se NULL (padrao), usa doc_id (que deve existir em x). Se nao for NULL, cada combinacao de by e tratada como um documento no calculo de idf.

Details

A definicao de tf, idf e tf_idf segue a literatura padrao de tf-idf em mineracao de texto:

- tf e a frequencia relativa do termo no documento;
- idf e log(N / df), em que N e o numero de documentos e df e o numero de documentos que contem o termo;
- tf_idf e o produto tf * idf.

A tabela de entrada deve ter exatamente uma linha por combinacao de documento/grupo e termo (isto e, uma linha por termo-em-documento).

Value

Um tibble::tibble() com as colunas originais de x mais tres colunas numeric as:

- tf: frequencia do termo no documento/grupo;
- idf: frequencia inversa de documentos;
- tf_idf: produto tf * idf.

See Also

ac_count(), ac_top_terms(), tidytext::bind_tf_idf()

Examples

```
# Corpus com discursos sobre tres agendas distintas.
# TF-IDF vai destacar o vocabulario proprio de cada texto,
# nao os termos comuns a todos (que a frequencia bruta mostraria).
df <- data.frame(
  id    = paste0("d", 1:6),
  texto = c(
    "reforma tributaria simplifica sistema impostos empresas",
    "IVA dual substitui PIS COFINS ICMS federal",
```

```
    "programa habitacional amplia recursos moradia popular urbana",
    "deficit habitacional afeta familias baixa renda cidades",
    "educacao basica recebe recursos Fundeb Congresso",
    "alfabetizacao idade certa meta Plano Nacional Educacao"
  ),
  tema  = rep(c("tributario", "habitacao", "educacao"), each = 2),
  stringsAsFactors = FALSE
)

corp <- ac_corpus(df, text = texto, docid = id)

# TF-IDF por documento
freq  <- ac_count(corp)
tfidf <- ac_tf_idf(freq)
head(tfidf)

# TF-IDF por tema (cada tema tratado como "documento agregado")
freq_by  <- ac_count(corp, by = "tema")
tfidf_by <- ac_tf_idf(freq_by, by = "tema")
head(tfidf_by)
```

`ac_tokenize` *Tokenizar textos de um corpus acR*

Description

`ac_tokenize()` recebe um objeto `ac_corpus()` e retorna um `tibble` em formato *tidy*, com um token por linha, no estilo usado em análises de texto no ecossistema tidy.

A função implementa tokenização em palavras (n = 1) ou n-gramas de tamanho arbitrário (n > 1), usando janelas contíguas de tokens dentro de cada documento.

É o segundo passo canônico do pipeline quantitativo, geralmente entre `ac_clean()` e `ac_count()`. Para análises que dependem de expressões compostas (\"reforma tributária\", \"desenvolvimento sustentável\"), use `n = 2L` para bigramas — cada n-grama vira uma unidade de contagem independente.

Usage

```
ac_tokenize(
  corpus,
  token = c("word"),
  n = 1L,
  keep_empty = FALSE,
  drop_punct = FALSE,
  ...
)
```

Arguments

corpus	Objeto de classe ac_corpus().
token	Tipo de tokenizacao desejada. Atualmente apenas "word" e suportado (padrao), reservado para futura expansao.
n	Tamanho do n-grama. Deve ser um inteiro maior ou igual a 1. Para n = 1, o resultado sao tokens individuais; para n = 2, bigramas ("A B"), para n = 3, trigramas etc.
keep_empty	Logico. Se FALSE (padrao), documentos que resultarem em texto vazio apos a limpeza nao geram linhas na saida. Se TRUE, cada documento vazio gera uma linha com token = NA quando n = 1.
drop_punct	Logico. Se TRUE, remove da sequencia tokens que consistem apenas de pontuacao (por exemplo "!", "...") antes de construir n-gramas. Tokens que misturam letras e pontuacao (por exemplo "ola,") sao mantidos.
...	Ignorado, reservado para argumentos futuros.

Value

Um tibble::tibble() com colunas:

- doc_id: identificador do documento (herdado de ac_corpus);
- token_id: posicao (1, 2, 3, ...) do token ou n-grama no documento;
- token: texto do token ou n-grama.

See Also

ac_corpus(), ac_clean()

Examples

```
df <- data.frame(
  id    = c("d1", "d2"),
  texto = c(
    "O deputado do PT falou na CCJ.",
    "Votar pra valer, agora!"
  )
)

corp <- ac_corpus(df, text = texto, docid = id)

# Tokenizacao simples em palavras
tokens <- ac_tokenize(corp)
tokens

# Removendo tokens que sao apenas pontuacao
df2 <- data.frame(
  id    = "d1",
  texto = "Ola, mundo ! ..."
)
```

```
corp2   <- ac_corpus(df2, text = texto, docid = id)
tokens2 <- ac_tokenize(corp2, drop_punct = TRUE)
tokens2

# Bigramas
ac_tokenize(corp, n = 2)
```

`ac_top_terms` *Selecionar os termos mais frequentes*

Description

`ac_top_terms()` seleciona os `n` termos mais frequentes a partir de uma tabela de frequencias (tipicamente o resultado de `ac_count()`).

Pode operar em dois modos:

- sem grupos (`by = NULL`): retorna os `n` termos mais frequentes no geral;
- com grupos (`by = c("partido", ...)`): retorna os `n` termos mais frequentes em cada combinacao de metadados.

Usage

```
ac_top_terms(x, n = 20L, by = NULL, sort = TRUE)
```

Arguments

`x`	Um `data.frame` ou `tibble::tibble()` contendo, no minimo, as colunas `token` e `n`. Em geral, o resultado de `ac_count()`.
`n`	Numero de termos a selecionar. Valor inteiro >= 1.
`by`	Vetor de nomes de colunas em `x` a serem usados como grupos de agregacao. Se `NULL` (padrao), a selecao e feita no conjunto total. Se nao for `NULL`, os `n` termos mais frequentes sao selecionados dentro de cada combinacao de `by`.
`sort`	Logico. Se `TRUE` (padrao), ordena a saida em ordem decrescente de frequencia (`n`). Se `FALSE`, preserva a ordem retornada pela operacao interna de selecao, apenas garantindo que os grupos (quando houver) venham juntos.

Value

Um `tibble::tibble()` com as mesmas colunas de `x`, mas restrito aos `n` termos mais frequentes (no geral ou por grupo).

See Also

`ac_count()`, `ac_tokenize()`

Examples

```
df <- data.frame(
  id      = c("d1", "d2", "d3"),
  texto   = c(
    "O deputado do PT falou na CCJ.",
    "O deputado do PL falou novamente.",
    "O senador do PT falou na CCJ."
  ),
  partido = c("PT", "PL", "PT"),
  stringsAsFactors = FALSE
)

corp <- ac_corpus(df, text = texto, docid = id, meta = partido)

# Top 10 termos no corpus inteiro
freq <- ac_count(corp)
ac_top_terms(freq, n = 10)

# Top 5 termos por partido
freq_by <- ac_count(corp, by = "partido")
ac_top_terms(freq_by, n = 5, by = "partido")
```

`ac_wordcloud` *Criar nuvem de palavras*

Description

`ac_wordcloud()` cria uma nuvem de palavras a partir de uma tabela de frequências, tipicamente gerada por `ac_count()`.

Por padrão prefere `ggwordcloud` (retorna `ggplot`, layout mais agradável, tipografia melhor); cai para `wordcloud` clássico se o primeiro não estiver instalado.

Usage

```
ac_wordcloud(
  x,
  max_words = 100,
  min_n = 1,
  colors = NULL,
  backend = c("auto", "ggwordcloud", "wordcloud"),
  title = NULL,
  seed = 42L,
  ...
)
```

Arguments

x	Um data.frame ou tibble contendo, no mínimo, as colunas token e n.
max_words	Número máximo de palavras a desenhar. Padrão: 100.
min_n	Frequência mínima para incluir um termo. Padrão: 1.
colors	Vetor de cores usado no gráfico. Padrão: paleta ac_palette().
backend	Motor a usar: "auto" (padrão, prefere ggwordcloud), "ggwordcloud" ou "wordcloud".
title	Título opcional (apenas em modo ggwordcloud).
seed	Semente para reprodutibilidade do layout. Padrão: 42L. Use NULL para usar o RNG corrente da sessão. A semente é escopada via withr::with_seed() e não altera o .Random.seed global.
...	Argumentos adicionais encaminhados para o motor escolhido (ggwordcloud::geom_text_wordcloud ou wordcloud::wordcloud).

Value

Um objeto ggplot (backend ggwordcloud) ou, invisivelmente, o data.frame filtrado (backend wordcloud).

See Also

ac_count(), ac_top_terms(), ac_palette()

Examples

```
# Corpus pequeno para demonstrar
df <- data.frame(
  id    = paste0("d", 1:8),
  texto = c(
    "reforma tributaria simplifica sistema empresas",
    "reforma reduz distorcoes fiscais brasileiras",
    "sistema tributario complexo prejudica empresas",
    "reforma modernizacao arrecadacao federal",
    "IVA substitui impostos indiretos federais",
    "reforma tributaria arrecadacao IVA aliquotas",
    "simplificacao impostos aliquotas empresas",
    "reforma federal moderniza sistema tributario"
  ),
  stringsAsFactors = FALSE
)
corp <- ac_corpus(df, text = texto, docid = id) |>
  ac_clean(remove_stopwords = "pt")
freq <- ac_count(corp)

# Motor ggplot moderno (recomendado)
if (requireNamespace("ggwordcloud", quietly = TRUE)) {
  ac_wordcloud(freq, max_words = 30, title = "Termos frequentes")
}

# Motor classico (fallback)
```

```
if (requireNamespace("wordcloud", quietly = TRUE)) {
  ac_wordcloud(freq, max_words = 30, backend = "wordcloud")
}
```

`as_prompt` *Converter codebook em system prompt para LLM*

Description

`as_prompt()` é um genérico S3 que converte um objeto em system prompt formatado para uso com LLMs. O método `as_prompt.ac_codebook()` gera o prompt a partir de um `ac_codebook`, incluindo instruções, categorias, exemplos, pesos e, opcionalmente, raciocínio estruturado.

Usage

```
as_prompt(x, ...)

## Default S3 method:
as_prompt(x, ...)

## S3 method for class 'ac_codebook'
as_prompt(
  x,
  reasoning = TRUE,
  reasoning_length = c("short", "medium", "detailed"),
  ...
)
```

Arguments

`x`	Objeto a converter (para `as_prompt.ac_codebook`: um `ac_codebook`).
`...`	Argumentos adicionais passados ao método.
`reasoning`	Lógico. Se `TRUE`, inclui campo de raciocínio no JSON de saída. Padrão: `TRUE`.
`reasoning_length`	Extensão do raciocínio: `"short"`, `"medium"` ou `"detailed"`.

Value

String com o system prompt (invisível).

Examples

```
# Codebook base
cb <- ac_qual_codebook(
  name         = "tom",
  instructions = "Classifique o tom do texto.",
```

```
  categories   = list(
    formal   = list(definition = "Linguagem tecnica e impessoal."),
    informal = list(definition = "Linguagem coloquial ou emotiva.")
  )
)

# Gerar o system prompt para uso direto com objetos Chat do ellmer
prompt <- as_prompt(cb, reasoning = TRUE, reasoning_length = "short")
substr(prompt, 1, 200)  # inspecionar o comeco do prompt
```

`is_ac_corpus` *Verificar se um objeto é um corpus do* `acR`

Description

Função auxiliar para testar se um objeto pertence à classe `ac_corpus`. Útil para validação de argumentos em funções do pipeline.

Usage

```
is_ac_corpus(x)
```

Arguments

`x`	Objeto qualquer a ser testado.

Value

`TRUE` se `x` é um objeto de classe `ac_corpus`, `FALSE` caso contrário.

Examples

```
# Objeto criado via ac_corpus() sempre tem a classe
corpus <- ac_corpus(c("Texto um.", "Texto dois."))
is_ac_corpus(corpus)         # TRUE

# Estruturas comuns nao contam
is_ac_corpus(data.frame())   # FALSE
is_ac_corpus("texto solto")  # FALSE
```

theme_ac *Tema visual consistente do acR*

Description

theme_ac() retorna um tema ggplot2 minimalista e consistente, usado por todos os ac_plot_*() do pacote. Deriva de ggplot2::theme_minimal() com ajustes editoriais: tipografia mais compacta, gridlines suaves, títulos em negrito com espaçamento negativo (visual editorial).

Também expõe ac_palette() para uma paleta categórica coerente com o tema (compatível com acessibilidade AA).

Usage

```
theme_ac(base_size = 12, base_family = "")
```

Arguments

base_size	Tamanho base da fonte. Padrão: 12.
base_family	Família tipográfica. Padrão: "" (usa sistema).

Value

Um objeto ggplot2::theme.

See Also

ac_palette()

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  ggplot2::ggplot(mtcars, ggplot2::aes(mpg, wt)) +
    ggplot2::geom_point(color = ac_palette()[1]) +
    ggplot2::labs(title = "MPG vs. peso", subtitle = "Tema editorial acR") +
    theme_ac()
}
```

Index

∗ **corpus**
ac_clean, 5
ac_corpus, 12
ac_count, 14
ac_keyness, 25
ac_plot_keyness, 33
ac_plot_tf_idf, 38
ac_plot_top_terms, 39
ac_tf_idf, 76
ac_tokenize, 78
ac_top_terms, 80
is_ac_corpus, 84
∗ **qualitative**
ac_calibrate_tokens, 3
ac_chat_local, 4
ac_estimate_tokens, 15
ac_qual_code, 43
ac_qual_codebook, 47
ac_qual_codebook_add, 50
ac_qual_codebook_history, 51
ac_qual_codebook_hybrid, 52
ac_qual_codebook_merge, 53
ac_qual_codebook_remove, 54
ac_qual_codebook_translate, 55
ac_qual_export_for_review, 56
ac_qual_import_human, 57
ac_qual_list_models, 61
ac_qual_load_codebook, 62
ac_qual_recommend_model, 63
ac_qual_reliability, 65
ac_qual_report, 67
ac_qual_report_full, 68
ac_qual_sample, 70
ac_qual_save_codebook, 72
ac_qual_search_literature, 73
as_prompt, 83
∗ **quantitative**
ac_cluster_documents, 9
ac_cooccurrence, 11
ac_lda, 27
ac_lda_tune, 29
ac_sentiment, 75
∗ **visualization**
ac_palette, 30
ac_plot_cluster, 31
ac_plot_cooccurrence, 32
ac_plot_lda_topics, 35
ac_plot_lda_tune, 36
ac_plot_sentiment, 37
ac_plot_wordcloud_comparative, 41
ac_plot_xray, 42
ac_wordcloud, 81
theme_ac, 85

ac_calibrate_tokens, 3
ac_calibrate_tokens(), *15, 16*
ac_chat_local, 4
ac_clean, 5
ac_clean(), *8, 9*, *15*, *25*, *78, 79*
ac_clean_stopwords, 8
ac_clean_stopwords(), *6, 7*
ac_cluster_documents, 9
ac_cluster_documents(), *31*
ac_cooccurrence, 11
ac_cooccurrence(), *32, 33*
ac_corpus, 12, *23*
ac_corpus(), *6, 7*, *14, 15*, *20*, *25*, *43*, *78, 79*
ac_count, 14
ac_count(), *25*, *27*, *34*, *39, 40*, *76–78*, *80–82*
ac_estimate_tokens, 15
ac_estimate_tokens(), *3, 4*
ac_export, 16
ac_fetch_camara, 18, *21–23*
ac_fetch_senado, 21
ac_import, 23
ac_keyness, 25
ac_keyness(), *14*, *33, 34*
ac_lda, 27
ac_lda(), *9, 10*, *29*, *35*

ac_lda_tune, 29
ac_lda_tune(), *28*, *36*
ac_palette, 30
ac_palette(), *31*, *41*, *82*, *85*
ac_plot_cluster, 31
ac_plot_cluster(), *10*
ac_plot_cooccurrence, 32
ac_plot_cooccurrence(), *11*
ac_plot_keyness, 33
ac_plot_lda_topics, 35
ac_plot_lda_topics(), *28*
ac_plot_lda_tune, 36
ac_plot_lda_tune(), *29*
ac_plot_sentiment, 37
ac_plot_sentiment(), *76*
ac_plot_tf_idf, 38
ac_plot_top_terms, 39
ac_plot_wordcloud_comparative, 41
ac_plot_xray, 42
ac_qual_code, 43
ac_qual_code(), *3–5*, *15*, *17*, *48*, *59–62*, *64–67*, *69*, *71*
ac_qual_codebook, 47
ac_qual_codebook(), *43, 44*, *50*, *55*
ac_qual_codebook_add, 50
ac_qual_codebook_add(), *55*
ac_qual_codebook_history, 51
ac_qual_codebook_hybrid, 52
ac_qual_codebook_merge, 53
ac_qual_codebook_remove, 54
ac_qual_codebook_remove(), *50*
ac_qual_codebook_translate, 55
ac_qual_export_for_review, 56
ac_qual_import_human, 57
ac_qual_import_human(), *65*
ac_qual_irr, 58
ac_qual_irr(), *17*, *69*
ac_qual_list_models, 61
ac_qual_list_models(), *64*
ac_qual_load_codebook, 62
ac_qual_recommend_model, 63
ac_qual_recommend_model(), *62*
ac_qual_reliability, 65
ac_qual_reliability(), *43*, *57, 58*, *68, 69*
ac_qual_report, 67
ac_qual_report(), *69, 70*
ac_qual_report_full, 68
ac_qual_sample, 70
ac_qual_sample(), *57*, *60*
ac_qual_save_codebook, 72
ac_qual_save_codebook(), *62*
ac_qual_search_literature, 73
ac_sentiment, 75
ac_sentiment(), *37*
ac_tf_idf, 76
ac_tf_idf(), *14*, *27*, *38, 39*, *42*
ac_tokenize, 78
ac_tokenize(), *11*, *14, 15*, *25*, *80*
ac_top_terms, 80
ac_top_terms(), *14*, *27*, *39, 40*, *77*, *82*
ac_wordcloud, 81
ac_wordcloud(), *14*
as_prompt, 83
attributes(), *46*

base::saveRDS(), *17*

dplyr::count(), *14*

ggplot2, *33*, *38, 39*
ggplot2::coord_flip(), *34*, *39, 40*

is_ac_corpus, 84

knitr::kable(), *17*

quanteda.textstats::textstat_keyness(), *25*

theme_ac, 85
theme_ac(), *30*
tibble::tibble(), *15*, *26*, *34*, *38*, *40*, *77*, *79, 80*
tidytext::bind_tf_idf(), *76*
topicmodels::LDA(), *27*

utils::write.csv(), *17*

writexl::write_xlsx(), *17*