Package 'datacaged'

September 9, 2026

Title CAGED Microdata Download and Storage in 'DuckDB'

Version 0.2.1

Description Tools to download, parse, and store Brazilian CAGED (Cadastro Geral de Empregados e Desempregados) microdata using 'DuckDB'. Supports both legacy CAGED (pre-2020), Novo CAGED (2020+), and adjustment files, enabling efficient local storage and analysis workflows. Data is sourced from the HuggingFace repository <https://huggingface.co/datasets/alexsandroprado/caged>.

SystemRequirements 7-Zip (optional, required for PPMd-compressed files from CAGED antigo and CAGED Ajustes)

Depends R (>= 4.2.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

LazyDataCompression bzip2

Date 2026-08-26

Imports archive, cli, DBI, dplyr, duckdb, future, furrr, glue, httr2 (>= 1.0.0), progressr, purrr, readr (>= 2.1.0), stringr, tibble, tools, utils

Suggests testthat (>= 3.0.0), knitr, rmarkdown, dbplyr

VignetteBuilder knitr

Config/testthat/edition 3

URL https://github.com/gecomt/datacaged

BugReports https://github.com/gecomt/datacaged/issues

Config/Needs/website rmarkdown

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Alexsandro Prado [aut, cre] (ORCID: <https://orcid.org/0000-0002-7072-3621>)

Maintainer Alexsandro Prado <alexsandro.prado@ufersa.edu.br>

Repository CRAN

Date/Publication 2026-09-09 16:40:07 UTC

Contents

`datacaged-package` *datacaged: Microdados do CAGED em DuckDB*

Description

O pacote `datacaged` facilita o acesso aos microdados do CAGED (Cadastro Geral de Empregados e Desempregados), cobrindo:

- **Novo CAGED** (janeiro/2020 em diante): layout reformulado pelo MTE
- **CAGED antigo** (até dezembro/2019): séries históricas (`CAGEDEST_*.7z`)
- **CAGED Ajustes** (até dezembro/2019): correções retroativas do antigo

Os dados são baixados do repositório HuggingFace (`https://huggingface.co/datasets/alexsandroprado/caged`), extraídos de arquivos `.7z` e carregados em um banco DuckDB local para consultas rápidas com SQL ou dplyr.

Details

Fluxo típico de uso

```
library(datacaged)

# Baixa, parseia e grava tudo em um comando
caged_load(
  years   = 2022:2023,
  months  = seq_len(12L),
  db_path = "meu_caged.duckdb"
)

# Conecta e consulta
con <- caged_connect("meu_caged.duckdb")
dplyr::tbl(con, "caged_mov") |>
  dplyr::filter(uf == 35) |>
  dplyr::count(competenciamov)
```

Funções principais

Novo CAGED (2020+) e CAGED antigo (até 2019):

- `caged_load()`: pipeline completo (download -> parse -> DuckDB)
- `caged_download()`: só baixa os arquivos .7z
- `caged_parse()`: lê e normaliza um arquivo `.7z`
- `caged_parse_batch()`: lê e normaliza múltiplos arquivos `.7z`
- `caged_to_duckdb()`: grava um data.frame no banco
- `caged_connect()`: abre conexão com o banco DuckDB
- `caged_info()`: lista tabelas e registros disponíveis no banco

CAGED Ajustes (correções retroativas até 2019):

- `caged_adjustments_load()`: pipeline completo para o CAGED Ajustes

Utilitários:

- `caged_status()`: verifica se o repositório HuggingFace está acessível
- `caged_hf_files()`: lista competências disponíveis no repositório HuggingFace

Tabelas no DuckDB

Tabela	Fonte	Período
`caged_mov`	`CAGEDMOV*.7z`	Jan/2020+
`caged_for`	`CAGEDFOR*.7z`	Jan/2020+
`caged_exc`	`CAGEDEXC*.7z`	Jan/2020+
`caged_antigo`	`CAGEDEST_*.7z`	1992–Dez/2019
`caged_ajustes`	`CAGEDEST_AJUSTES_*.7z`	1992–Dez/2019

Author(s)

Maintainer: Alexsandro Prado <alexsandro.prado@ufersa.edu.br> (ORCID)

Authors:

- Alexsandro Prado <alexsandro.prado@ufersa.edu.br> (ORCID)

See Also

Useful links:

- https://github.com/gecomt/datacaged
- Report bugs at https://github.com/gecomt/datacaged/issues

`caged_adjustments_load`

Pipeline completo para o CAGED Ajustes

Description

Baixa, parseia e grava os arquivos do CAGED Ajustes no banco DuckDB local.

Usage

```
caged_adjustments_load(
  years,
  months = seq_len(12L),
  db_path,
  destdir = NULL,
  force_download = FALSE,
  overwrite_competencies = FALSE,
  timeout = 300
)
```

Arguments

`years`	integer vector. Desired years. Maximum: 2019.
`months`	integer vector. Desired months (1–12). Default: `seq_len(12L)`.
`db_path`	character. Path to the file `.duckdb`.
`destdir`	character or NULL. Cache directory for `.7z` files. Default: `tools::R_user_dir("datacaged", "cache")`.
`force_download`	logical. If TRUE, re-downloads even if already cached. Default: FALSE.
`overwrite_competencies`	logical. If TRUE, overwrites competencies already in the database. Default: FALSE.
`timeout`	integer. Timeout in seconds. Default: 300.

Details

O CAGED Ajustes contém correções retroativas de vínculos do CAGED antigo (até 2019). Cada arquivo `CAGEDEST_AJUSTES_{MM}{AAAA}.7z` registra movimentações ajustadas após a declaração original — essencial para reconstrução de séries históricas mais precisas.

Os dados são gravados na tabela `caged_ajustes` do banco DuckDB, com o mesmo schema do `caged_antigo`.

Value

invisível: tibble com estatísticas do banco após a carga.

See Also

`caged_load()` para Novo CAGED (2020+) e CAGED antigo (até 2019).

`caged_hf_files()` com `type = "ajustes"` para listar disponibilidade.

Examples

```
## Not run:
# List available adjustment competencies first
avail <- caged_hf_files(type = "ajustes", n = 3, verbose = FALSE)

if (nrow(avail) > 0) {
  db <- file.path(tempdir(), "caged_ajustes.duckdb")

  # Download and load adjustments
  caged_adjustments_load(
    years   = avail$ano[1],
    months  = avail$mes[1],
    db_path = db
  )

  con <- caged_connect(db)
  # List tables and records
  dplyr::tbl(con, "caged_ajustes") |>
    dplyr::group_by(competencia) |>
    dplyr::count() |>
    dplyr::collect()
  DBI::dbDisconnect(con, shutdown = TRUE)
}

## End(Not run)
```

caged_connect *Abre uma conexão com o banco DuckDB do CAGED*

Description

Cria o arquivo `.duckdb` se ainda não existir. A conexão retornada é compatível com `dplyr::tbl()`, `DBI::dbGetQuery()` e `dbplyr`.

Usage

```
caged_connect(db_path, read_only = FALSE, quiet = FALSE)
```

Arguments

`db_path`	character. Path to the file `.duckdb`. Use `":memory:"` para um banco temporário em memória.
`read_only`	logical. If TRUE, abre somente leitura. Default: FALSE.
`quiet`	logical. If TRUE, suppresses the connection message. Default: FALSE.

Details

Lembre-se de fechar a conexão com `DBI::dbDisconnect()` ao terminar.

Value

objeto de conexão DBI (`duckdb_connection`).

See Also

`caged_info()` para listar tabelas no banco.

Examples

```
con <- caged_connect(file.path(tempdir(), "caged.duckdb"))

# Listar tabelas disponíveis
DBI::dbListTables(con)

# Sempre fechar ao terminar
DBI::dbDisconnect(con, shutdown = TRUE)
```

caged_download *Baixa microdados do CAGED do repositório HuggingFace*

Description

Baixa os arquivos `.7z` para um diretório local, criando uma estrutura de pastas por ano. Arquivos já baixados não são re-baixados (cache local).

Usage

```
caged_download(
  years,
  months = seq_len(12L),
  states = NULL,
  destdir = NULL,
  force = FALSE,
  timeout = 300,
  workers = getOption("datacaged.workers", min(3L, future::availableCores()))
)
```

Arguments

`years`	integer vector. Desired years (ex: `2020:2023`).
`months`	integer vector. Desired months (1–12). Default: `seq_len(12L)` (todos os meses).
`states`	character vector or NULL. **Ignorado** — mantido por compatibilidade.
`destdir`	character or NULL. Local directory to save os arquivos. If NULL, uses the default directory do sistema via `tools::R_user_dir("datacaged", "cache")`.
`force`	logical. If TRUE, re-downloads files already in cache. Default: FALSE.
`timeout`	integer. Timeout por arquivo em segundos. Default: 300.
`workers`	integer. Number of parallel downloads. Default: 3. Use `1` for sequential mode. Controlled via `options(datacaged.workers = N)`.

Details

Para o **Novo CAGED** (2020+), são baixados três tipos por competência: movimentações (`MOV`), fora do prazo (`FOR`) e exclusões (`EXC`).

Para o **CAGED antigo** (até 2019), o download é nacional (arquivo único por competência).

Compatibilidade entre plataformas:

Recurso	Windows	macOS	Linux
Download (HTTPS)	OK	OK	OK
Downloads paralelos	OK	OK	OK
Novo CAGED (2020+, LZMA)	OK	OK	OK
CAGED antigo (PPMd)	(!)	(!)	(!)

O CAGED antigo (pré-2020) usa PPMd e requer 7-Zip instalado. O Novo CAGED usa LZMA e funciona em todas as plataformas sem dependências extras.

Value

data.frame invisível com colunas `arquivo`, `competencia`, `type`, `uf` e `status` (baixado / cache / nao_encontrado / erro).

See Also

`caged_load()` para o pipeline completo (download + parse + DuckDB).

`caged_hf_files()` para listar competências disponíveis no HuggingFace.

Examples

```
## Not run:
# Baixa Novo CAGED de jan-mar/2023
manifest <- caged_download(years = 2023, months = c(1L, 2L, 3L))

# Ver status de cada arquivo (baixado / cache / nao_encontrado / erro)
dplyr::count(manifest, status)

# Baixa CAGED antigo de 2018
caged_download(years = 2018)

# Forçar re-download mesmo com cache
caged_download(years = 2023, months = 1, force = TRUE)

# Salvar em diretório personalizado
caged_download(years = 2023, months = 1, destdir = file.path(tempdir(), "caged_cache"))

## End(Not run)
```

`caged_download_layouts` *Baixa layouts (dicionários) do CAGED e Novo CAGED*

Description

Lista dinamicamente os dicionários de variáveis do CAGED disponíveis no FTP do MTE e baixa todos os arquivos `.xls` ou `.xlsx` cujo nome contenha a palavra "layout" (case-insensitive).

Usage

```
caged_download_layouts(
  type = "ambos",
  destdir = NULL,
  force = FALSE,
  timeout = 120,
  verbose = TRUE
)
```

Arguments

type	character. Qual base consultar: "antigo", "novo", "ajustes" ou "ambos". Default "ambos".
destdir	character or NULL. Local directory to save os arquivos. Se NULL, usa tools::R_user_dir("datacaged" "cache")/layouts.
force	logical. If TRUE, re-downloads files that already exist. Default FALSE.
timeout	integer. Timeout por arquivo em segundos. Default 120.
verbose	logical. Se TRUE, exibe mensagens no console. Default TRUE.

Value

tibble invisível com colunas base, arquivo, url, destino e status ("baixado", "cache" ou "naoencontrado").

See Also

caged_status() para verificar conectividade.

caged_hf_files() para consultar os microdados disponíveis.

Examples

```
## Not run:
caged_download_layouts()
caged_download_layouts(type = "antigo")
caged_download_layouts(type = "novo")
caged_download_layouts(type = "ajustes")

## End(Not run)
```

`caged_hf_files` *Lista as competências disponíveis no repositório HuggingFace*

Description

Consulta a API do HuggingFace e retorna os meses disponíveis para download, ordenados do mais recente para o mais antigo.

Usage

```
caged_hf_files(type = "novo", n = 12, timeout = 15, verbose = TRUE)
```

Arguments

`type`	character. Which series to query: `"novo"` (Novo CAGED, 2020+), `"antigo"` (CAGED antigo, até 2019) ou `"ajustes"` (CAGED Ajustes, série histórica de correções). Default: `"novo"`.
`n`	integer. Number of most recent competencies to return. Use `Inf` to return all. Default: `12`.
`timeout`	integer. Timeout in seconds. Default: 15.
`verbose`	logical. If TRUE, exibe tabela no console. Default: TRUE.

Value

tibble invisível com colunas `competencia` (AAAAMM), `ano`, `mes` e `url`, ordenado do mais recente para o mais antigo.

See Also

`caged_status()` para verificar se o repositório HF está acessível.

`caged_load()` para baixar após identificar as competências.

Examples

```
## Not run:
# Últimos 12 meses disponíveis (padrão)
caged_hf_files()

# Todos os meses do Novo CAGED
caged_hf_files(n = Inf)

# CAGED antigo
caged_hf_files(type = "antigo")

# CAGED Ajustes
caged_hf_files(type = "ajustes")
```

```
# Usar o resultado para baixar automaticamente o mês mais recente
disp <- caged_hf_files(n = 1, verbose = FALSE)
caged_load(
  years    = disp$ano,
  months   = disp$mes,
  db_path = file.path(tempdir(), "caged.duckdb")
)

## End(Not run)
```

`caged_info` *Lista tabelas e estatísticas do banco DuckDB do CAGED*

Description

Exibe no console uma tabela com nome, número de registros, competências disponíveis e tamanho estimado de cada tabela CAGED.

Usage

```
caged_info(db_path)
```

Arguments

`db_path`	character. Path to the file `.duckdb`.

Value

tibble invisível com as estatísticas.

See Also

`caged_connect()` para obter uma conexão DBI com o banco.

Examples

```
caged_info(file.path(tempdir(), "caged.duckdb"))
```

caged_load *Pipeline completo: download -> parse -> DuckDB*

Description

Combina `caged_download()`, `caged_parse()` e `caged_to_duckdb()` em um único comando. É o jeito mais simples de popular o banco local.

Usage

```
caged_load(
  years,
  months = seq_len(12L),
  states = NULL,
  db_path = file.path(tempdir(), "caged.duckdb"),
  destdir = NULL,
  force_download = FALSE,
  overwrite_competencies = FALSE,
  timeout = 300,
  workers = getOption("datacaged.workers", min(3L, future::availableCores()))
)
```

Arguments

`years`	integer vector. Desired years.
`months`	integer vector. Desired months (1–12). Default: `seq_len(12L)` (todos os meses).
`states`	character vector or NULL. **Ignored** — the parameter is validated but does not filter downloads for any series (Novo CAGED, antigo or Ajustes), as all files are national in scope. Kept for backwards compatibility.
`db_path`	character. Path to the file `.duckdb`. Default: `file.path(tempdir(), "caged.duckdb")`.
`destdir`	character or NULL. Cache directory for `.7z` files.
`force_download`	logical. Re-downloads files already in cache. Default: FALSE.
`overwrite_competencies`	logical. Overwrites competencies already in the database. Default: FALSE.
`timeout`	integer. Timeout per file in seconds. Default: 300.
`workers`	integer. Number of parallel downloads. Default: 3. Use 1 for sequential mode. Controlled via `options(datacaged.workers = N)`.

Value

invisible: tibble with final database statistics (via `caged_info()`).

See Also

caged_adjustments_load() para o CAGED Ajustes (correções retroativas até 2019).

caged_download() para baixar sem gravar no banco.

caged_info() para inspecionar o banco após a carga.

Examples

```
## Not run:
# Download real --- exemplos nao executados automaticamente (requerem rede e tempo)

# Novo CAGED: 1 mes recente
caged_load(
  years   = 2024,
  months  = 1,
  db_path = file.path(tempdir(), "caged.duckdb")
)

# CAGED antigo: 1 ano
caged_load(
  years   = 2019,
  months  = seq_len(12L),
  db_path = file.path(tempdir(), "caged_historico.duckdb")
)

# Conecta e consulta
con <- caged_connect(file.path(tempdir(), "caged.duckdb"))
if ("caged_mov" %in% DBI::dbListTables(con)) {
  dplyr::tbl(con, "caged_mov") |>
    dplyr::group_by(competencia, uf) |>
    dplyr::summarise(saldo = sum(saldomovimentacao, na.rm = TRUE)) |>
    dplyr::collect()
}
DBI::dbDisconnect(con, shutdown = TRUE)

## End(Not run)
```

---

caged_parse *Lê microdados de um arquivo .7z do CAGED*

Description

Lê os `.txt` internos diretamente via stream (sem extrair para disco), usando `archive::archive_read()`. Se o stream falhar, faz fallback para extração em diretório temporário. Devolve um único tibble normalizado.

Usage

```
caged_parse(path, type = NULL)
```

Arguments

path	Path to the file .7z.
type	character or NULL. File type ("MOV", "FOR", "EXC", "ANTIGO"). Se NULL, detecta pelo nome.

Value

tibble com os microdados, ou NULL se o arquivo estiver vazio/inválido.

See Also

caged_parse_batch() para processar múltiplos arquivos de uma vez.

caged_to_duckdb() para gravar o resultado no banco.

Examples

```
# Toy example com arquivo incluído no pacote (executa sem internet)
path_mov <- system.file("extdata", "CAGEDMOV202301_exemplo.7z", package = "datacaged")
df <- caged_parse(path_mov, type = "MOV")
head(df)

## Not run:
# Com arquivo baixado manualmente via caged_download() ou pelo navegador:
df_mov <- caged_parse(file.path("~", "Downloads", "CAGEDMOV202301.7z"))
dplyr::glimpse(df_mov)

# Tipo detectado automaticamente
df_for <- caged_parse(file.path("~", "Downloads", "CAGEDFOR202301.7z"))

## End(Not run)
```

---

caged_parse_batch *Lê e normaliza múltiplos arquivos .7z do CAGED*

Description

Wrapper sobre caged_parse() que processa um vetor de caminhos, empilha os resultados e reporta progresso.

Usage

```
caged_parse_batch(paths, .progress = TRUE)
```

Arguments

paths	character vector. Caminhos dos arquivos .7z.
.progress	logical. Exibe barra de progresso. Default: TRUE.

Details

Tipicamente você não chama esta função diretamente — ela é usada internamente por `caged_load()`. Mas é útil quando você quer controle manual sobre o que parsear.

Value

tibble empilhado com todos os registros, ou NULL se nenhum arquivo for legível.

See Also

`caged_parse()` para processar um arquivo individual.

`caged_to_duckdb()` para gravar o resultado no banco.

Examples

```
# Ler todos os arquivos MOV de um diretório de cache
cache <- tools::R_user_dir("datacaged", "cache")
arquivos_mov <- list.files(
  file.path(cache, "NOVO_CAGED", "2023"),
  pattern    = "CAGEDMOV",
  full.names = TRUE
)
df <- caged_parse_batch(arquivos_mov)

# Gravar no banco após parsear
caged_to_duckdb(df, db_path = file.path(tempdir(), "caged.duckdb"))
```

`caged_status` *Verifica se o repositório HuggingFace do CAGED está acessível*

Description

Faz uma requisição leve à API do HuggingFace e retorna o status. Útil para diagnosticar problemas de conectividade antes de iniciar um download com `caged_load()` ou `caged_download()`.

Usage

```
caged_status(timeout = 10, verbose = TRUE)
```

Arguments

`timeout`	integer. Timeout in seconds. Default: 10.
`verbose`	logical. If TRUE, exibe mensagem detalhada no console. Default: TRUE.

Value

invisível: lista com campos `online` (logical), `latencia_ms` (numeric), `url` (character) e `mensagem` (character).

See Also

`caged_hf_files()` para listar competências disponíveis.

Examples

```
## Not run:
caged_status()

# Só verificar sem imprimir
st <- caged_status(verbose = FALSE)
st$online

## End(Not run)
```

`caged_to_duckdb` *Grava um data.frame de microdados do CAGED em uma tabela DuckDB*

Description

Usa `INSERT OR IGNORE` por competência para evitar duplicatas: se uma competência já existe na tabela, ela é pulada (útil para re-rodar o pipeline sem apagar dados anteriores).

Usage

```
caged_to_duckdb(
  df,
  db_path = NULL,
  table = NULL,
  overwrite_competencies = FALSE,
  .con = NULL
)
```

Arguments

`df`	data.frame or tibble. Data returned by `caged_parse()`.
`db_path`	character or NULL. Path to the file `.duckdb`. Pode ser NULL se `.con` for fornecido.
`table`	character or NULL. Name of the destination table. If NULL, detecta automaticamente pela coluna `fonte_tipo` do data.frame.

`overwrite_competencies`
: logical. If TRUE, deletes records for the competency before inserting (avoids duplicates on re-runs). Default: FALSE.

`.con`
: a DBI connection object (optional). If provided, `db_path` is ignored and the existing connection is reused. The caller is responsible for closing the connection.

Details

As tabelas criadas são:

- `caged_mov`, `caged_for`, `caged_exc` — para dados de 2020 em diante
- `caged_antigo` — para dados até 2019 (CAGED antigo)
- `caged_ajustes` — ajustes retroativos do CAGED antigo

Value

invisible: number of rows inserted.

See Also

`caged_parse()` para gerar o data.frame de entrada.

`caged_load()` para o pipeline completo.

Examples

```
## Not run:
# Parse e grava em um único fluxo (requer arquivo baixado)
df <- caged_parse("CAGEDMOV202301.7z")
caged_to_duckdb(df, db_path = file.path(tempdir(), "caged.duckdb"))

# Regravar uma competência já existente
caged_to_duckdb(df, db_path = file.path(tempdir(), "caged.duckdb"),
                overwrite_competencies = TRUE)

## End(Not run)
```

`caged_to_parquet` *Exporta tabelas do banco DuckDB para arquivos Parquet*

Description

Usa o mecanismo nativo `COPY TO ... (FORMAT PARQUET)` do DuckDB para exportação eficiente. Significativamente mais rápido que `collect()` + `arrow::write_parquet()` para grandes volumes.

Usage

```
caged_to_parquet(
  db_path,
  output_dir,
  tables = NULL,
  partition_by = NULL,
  overwrite = FALSE
)
```

Arguments

db_path	character. Path to the file .duckdb.
output_dir	character. Output directory for .parquet files. Criado automaticamente se não existir.
tables	character vector or NULL. Tables to export. If NULL, exporta todas as tabelas CAGED presentes no banco.
partition_by	character or NULL. Column for partitioning (ex: "uf" ou "competencia"). If NULL, generates one file per table.
overwrite	logical. Overwrite existing files. Default: FALSE.

Value

invisível: tibble com tabela, arquivo e registros exportados.

See Also

caged_info() para listar tabelas disponíveis.

Examples

```
## Not run:
db <- file.path(tempdir(), "caged.duckdb")
caged_load(years = 2023, months = 1, db_path = db)

# Exportar todas as tabelas
caged_to_parquet(db, output_dir = tempdir())

# Exportar só caged_mov, particionado por UF
caged_to_parquet(
  db,
  output_dir   = tempdir(),
  tables       = "caged_mov",
  partition_by = "uf"
)

## End(Not run)
```

caged_update *Atualiza o banco com as competências mais recentes disponíveis*

Description

Consulta o HuggingFace para descobrir as competências disponíveis, compara com o que já existe no banco e baixa apenas o que está faltando. É o modo mais prático de manter o banco atualizado sem re-baixar tudo.

Usage

```
caged_update(
  db_path,
  series = "novo",
  destdir = NULL,
  workers = getOption("datacaged.workers", min(3L, future::availableCores())),
  timeout = 300,
  verbose = TRUE
)
```

Arguments

db_path	character. Path to the file .duckdb.
series	character vector. Series to update: "novo", "antigo", "ajustes" ou qualquer combinação. Default: "novo".
destdir	character or NULL. Cache directory for .7z files.
workers	integer. Number of parallel downloads. Default: 3.
timeout	integer. Timeout por arquivo em segundos. Default: 300.
verbose	logical. Displays details in the console. Default: TRUE.

Value

invisível: tibble com estatísticas do banco após a atualização, ou NULL se o banco já estiver atualizado.

See Also

caged_load() para carga inicial completa.

caged_info() para inspecionar o banco.

caged_hf_files() para listar competências disponíveis.

Examples

```
## Not run:
# Atualizar Novo CAGED com as competências mais recentes
caged_update(db_path = file.path(tempdir(), "caged.duckdb"))

# Atualizar todas as séries
caged_update(
  db_path = file.path(tempdir(), "caged.duckdb"),
  series  = c("novo", "antigo", "ajustes")
)

## End(Not run)
```

`uf_codigos` *Tabela de UFs brasileiras com códigos IBGE e regiões*

Description

Data frame com as 27 unidades federativas do Brasil, seus códigos numéricos do IBGE e suas regiões geográficas. Útil para joins com os microdados do CAGED, que armazenam apenas o código numérico da UF.

Usage

```
uf_codigos
```

Format

Data frame com 27 linhas e 3 colunas:

sigla Sigla da UF (character), ex: "SP", "RJ", "RN".

codigo Código numérico IBGE da UF (integer), ex: 35, 33, 24.

regiao Região geográfica (character): "Norte", "Nordeste", "Sudeste", "Sul" ou "Centro-Oeste".

Source

IBGE (Instituto Brasileiro de Geografia e Estatistica).

Examples

```
data(uf_codigos)
head(uf_codigos)

# Join com microdados do CAGED (requer banco populado com caged_load())
## Not run:
con <- caged_connect(file.path(tempdir(), "caged.duckdb"))
if ("caged_mov" %in% DBI::dbListTables(con)) {
```

```
  df <- dplyr::tbl(con, "caged_mov") |> dplyr::collect()
  dplyr::left_join(df, uf_codigos, by = c("uf" = "codigo"))
}
DBI::dbDisconnect(con, shutdown = TRUE)

## End(Not run)
```

Index