Package: wrds (via r-universe)

May 29, 2026

Title Access 'Wharton Research Data Services' ('WRDS')

Version 0.1.1.9000

Description Provides simple functions for accessing data from 'Wharton Research Data Services' ('WRDS'), a widely used financial database in academic research. Includes credential management via the system keyring, database tools, and functions for downloading generic tables, 'Compustat' fundamentals, and linking tables.

Depends R (>= 4.1)

License MIT + file LICENSE

LazyData true

Encoding UTF-8

Config/roxygen2/markdown TRUE

URL https://statzhero.github.io/wrds/, https://github.com/statzhero/wrds

BugReports https://github.com/statzhero/wrds/issues

Suggests testthat (>= 3.0.0), withr

Config/testthat/edition 3

Imports cli, DBI, dbplyr, dplyr, keyring, rlang, RPostgres, tidylog

Config/roxygen2/version 8.0.0

Config/Needs/website pkgdown

Config/pak/sysreqs libicu-dev libsecret-1-dev libpq-dev

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

Date/Publication 2026-05-29 16:12:59 UTC

RemoteUrl https://github.com/statzhero/wrds

RemoteRef HEAD

RemoteSha f4f71b26ee985498f638bcfd8071b9d0c3a5445f

Contents

`describe_table` *Describe a table*

Description

Displays a glimpse-like summary of a WRDS table showing column names, types, and human-readable labels, similar to `dplyr::glimpse()`.

Usage

```
describe_table(wrds, library, table, n = 20, max_cols = 25)
```

Arguments

`wrds`	A `DBIConnection` object returned by `wrds_connect()`.
`library`	Character. The name of the library (schema).
`table`	Character. The name of the table.
`n`	Integer. Number of sample rows to fetch for value preview. Default is 20.
`max_cols`	Integer. Maximum number of columns to display. Default is 25.

Value

Invisibly returns a list with components:

columns A data frame with `column_name`, `data_type`, and `label`

description Table description, or `NA` if unavailable

nrow Row count

sample A data frame with sample rows (if `n > 0`)

Examples

```
## Not run:
wrds <- wrds_connect()
describe_table(wrds, "comp", "funda")
wrds_disconnect(wrds)

## End(Not run)
```

`get_company` *Download Compustat company header data*

Description

Downloads company-level static data from Compustat including header SIC codes, NAICS codes, state of incorporation, and other identifying information.

Usage

```
get_company(
  wrds,
  region = c("na", "global"),
  columns = NULL,
  n = Inf,
  lazy = FALSE
)
```

Arguments

`wrds`	A `DBIConnection` object returned by `wrds_connect()`.
`region`	One of `"na"` (North America, default) or `"global"`.
`columns`	Character vector of columns to return. Defaults to key identifiers and classification codes.
`n`	Maximum number of rows to return. Defaults to `Inf` (all rows). Use a smaller value (e.g., `n = 100`) to preview data before downloading the full table.
`lazy`	If `TRUE`, returns a lazy `tbl` instead of collecting. Defaults to `FALSE`.

Details

The `sic` column contains the "header" SIC code, which is the company's most recent SIC classification stored as a character. For historical SIC codes that change over time, use `get_compustat()` with `fill_sic = TRUE`, which coalesces the historical `sich` (integer) with the header `sic`.

Value

A tibble with company header data. Default columns vary by region:

North America (from `comp.company`):

- `gvkey`, `conm`: Identifiers
- `sic`, `naics`: Industry classifications (character)
- `state`, `fic`, `loc`: Geographic information

Global (from `comp.g_company`):

- `gvkey`, `conm`: Identifiers
- `sic`, `naics`: Industry classifications
- `fic`, `loc`: Geographic information

See Also

`get_compustat()` for fundamentals data with optional SIC filling

Examples

```
## Not run:
wrds <- wrds_connect()

# Get company header data
company <- get_company(wrds)

# Get global companies
g_company <- get_company(wrds, region = "global")

# Lazy query
get_company(wrds, lazy = TRUE) |>
  dplyr::filter(sic == "7370") |>
  dplyr::collect()

wrds_disconnect(wrds)

## End(Not run)
```

`get_compustat` *Download Compustat fundamentals*

Description

Downloads financial statement data from Compustat with standard filters for clean, analysis-ready data.

Usage

```
get_compustat(
  wrds,
  frequency = c("annual", "quarterly"),
  region = c("na", "global"),
  start_date = NULL,
  end_date = NULL,
  columns = NULL,
  add_columns = NULL,
  indfmt = "INDL",
  consol = "C",
  fill_sic = FALSE,
  n = Inf,
  lazy = FALSE
)
```

Arguments

wrds	A DBIConnection object returned by wrds_connect().
frequency	One of "annual" (default) or "quarterly".
region	One of "na" (North America, default) or "global".
start_date	Start date for filtering. Character string in "YYYY-MM-DD" format or a Date object. Defaults to NULL (no filter).
end_date	End date for filtering. Character string in "YYYY-MM-DD" format or a Date object. Defaults to NULL (no filter).
columns	Character vector of columns to return, replacing the defaults. Use describe_table() to see available columns.
add_columns	Character vector of additional columns to include beyond the defaults. Ignored if columns is specified.
indfmt	Industry format filter. Defaults to "INDL" (industrial). Use "FS" for financial services format.
consol	Consolidation level. Defaults to "C" (consolidated). Use "B" for both consolidated and non-consolidated.
fill_sic	If TRUE, fills missing historical SIC codes (sich) with header SIC codes from comp.company. Only supported for North America. When used with lazy = TRUE, returns the table with sich but without the join (requires manual joining with get_company()). Defaults to FALSE.
n	Maximum number of rows to return. Defaults to Inf (all rows). Use a smaller value (e.g., n = 100) to preview data before downloading the full table.
lazy	If TRUE, returns a lazy tbl instead of collecting. Defaults to FALSE.

Details

Default filters follow standard practice for most research applications. Region-specific filters are applied automatically based on region:

- `datafmt`: `"STD"` for North America, `"HIST_STD"` for Global
- `popsrc`: `"D"` (domestic) for North America, `"I"` (international) for Global

North America and Global data have different structures and should not be combined without careful column harmonization.

Value

A tibble with Compustat fundamentals. Default columns vary by region:

North America (from `comp.funda` / `comp.fundq`):

- Identifiers: `gvkey`, `cusip`, `tic`, `conm`, `datadate`
- Time: `fyear`/`fyearq`, `fyr`/`fqtr`
- Income: `ni`/`niq`, `ib`/`ibq`, `oiadp`/`oiadpq`, `revt`/`revtq`
- Balance sheet: `at`/`atq`, `lt`/`ltq`, `seq`/`seqq`, `ceq`/`ceqq`
- Market: `csho`/`cshoq`, `prcc_f`/`prccq`
- Other: `sale`/`saleq`, `capx`/`capxy`, `che`/`cheq`, `dlc`/`dlcq`, `dltt`/`dlttq`
- Industry: `sich` (historical SIC); `sic` (when `fill_sic = TRUE`, coalesced from `sich` and header SIC)

Global (from `comp.g_funda` / `comp.g_fundq`):

- Identifiers: `gvkey`, `isin`, `conm`, `datadate`
- Geography: `loc`, `fic`, `exchg`
- Similar financial variables (with some differences, e.g., `nit`/`nitq` instead of `ni`/`niq`)

See Also

`link_ccm()` for CRSP-Compustat linking, `get_company()` for company header data

Examples

```
## Not run:
wrds <- wrds_connect()

# Annual North America fundamentals
funda <- get_compustat(wrds)

# Quarterly with date filter
fundq <- get_compustat(wrds,
  frequency = "quarterly",
  start_date = "2020-01-01",
  end_date = "2023-12-31"
)

# Global annual
g_funda <- get_compustat(wrds, region = "global")

# Lazy query for further filtering
```

```
get_compustat(wrds, lazy = TRUE) |>
  dplyr::filter(fyear >= 2020) |>
  dplyr::select(gvkey, datadate, at, lt) |>
  dplyr::collect()

# Fill missing SIC codes with header SIC from comp.company
funda_sic <- get_compustat(wrds, fill_sic = TRUE)

# Preview first 100 rows before full download
preview <- get_compustat(wrds, n = 100)

wrds_disconnect(wrds)

## End(Not run)
```

`get_table` *Download data from any WRDS table*

Description

Generic function to download data from any table in the WRDS database. Returns a lazy table by default, allowing you to build queries with dplyr before collecting.

Usage

```
get_table(wrds, library, table, columns = NULL, n = Inf, lazy = TRUE)
```

Arguments

`wrds`	A `DBIConnection` object returned by `wrds_connect()`.
`library`	Character. The name of the library (schema), e.g., `"crsp"`, `"comp"`, `"ibes"`.
`table`	Character. The name of the table within the library.
`columns`	Character vector of columns to return. If `NULL` (default), returns all columns. Use `describe_table()` to see available columns.
`n`	Maximum number of rows to return. Defaults to `Inf` (all rows). Use a smaller value (e.g., `n = 100`) to preview data.
`lazy`	If `TRUE` (default), returns a lazy `tbl` for further filtering with dplyr. Set to `FALSE` to collect immediately.

Details

This function provides generic access to any WRDS table. For commonly-used tables with standard research filters, prefer the specialized functions:

- `get_compustat()` for Compustat fundamentals with standard filters
- `get_company()` for company header data
- `link_ccm()` for CRSP-Compustat linking

The lazy table can be filtered, selected, and mutated using dplyr verbs, which are translated to SQL and executed on the server:

```
get_table(wrds, "crsp", "msf") |>
  filter(date >= "2025-01-01") |>
  select(permno, date, ret, prc) |>
  collect()
```

Value

A `tbl_lazy` object (if `lazy = TRUE`) or a tibble (if `lazy = FALSE`).

See Also

`describe_table()` to explore table structure, `list_tables()` to list available tables in a library

Examples

```
## Not run:
wrds <- wrds_connect()

# Preview table structure first
describe_table(wrds, "crsp", "msf")

# Get a lazy table and build your query
get_table(wrds, "crsp", "msf") |>
  dplyr::filter(date >= "2025-01-01") |>
  dplyr::select(permno, date, ret, prc, vol) |>
  dplyr::collect()

# Collect immediately with specific columns
get_table(wrds, "crsp", "dsf",
  columns = c("permno", "date", "ret", "prc"),
  lazy = FALSE,
  n = 1000
)

# Access any table in any library
get_table(wrds, "ibes", "statsum_epsus") |>
  dplyr::filter(fpedats >= "2025-01-01") |>
  dplyr::collect()

wrds_disconnect(wrds)

## End(Not run)
```

`link_ccm` *Get CRSP-Compustat linking table*

Description

Downloads the CCM (CRSP-Compustat Merged) linking table that maps CRSP PERMNOs to Compustat GVKEYs with valid date ranges.

Usage

```
link_ccm(
  wrds,
  linktype = c("LC", "LU", "LS"),
  linkprim = c("P", "C"),
  n = Inf,
  lazy = FALSE
)
```

Arguments

`wrds`	A `DBIConnection` object returned by `wrds_connect()`.
`linktype`	Character vector. Types of links to include. Defaults to `c("LC", "LU", "LS")`: • `"LC"`: Link confirmed by Compustat • `"LU"`: Link unconfirmed (valid but less certain) • `"LS"`: Link valid for secondary securities
`linkprim`	Character vector. Link primacy filters. Defaults to `c("P", "C")`: • `"P"`: Primary link identified by Compustat • `"C"`: Primary link identified by CRSP
`n`	Maximum number of rows to return. Defaults to `Inf` (all rows). Use a smaller value (e.g., `n = 100`) to preview data before downloading the full table.
`lazy`	If `TRUE`, returns a lazy `tbl` instead of collecting. Defaults to `FALSE`.

Details

The linking table comes from `crsp.ccmxpf_lnkhist`. Missing `linkenddt` values indicate ongoing links and are replaced with the maximum date in the table for easier date-range joins.

To use the link, join on gvkey and ensure your observation date falls within the `linkdt` to `linkenddt` range.

Value

A tibble with columns:

gvkey Compustat company identifier

permno CRSP permanent security identifier

linkdt Start date of the link

linkenddt End date of the link (missing values replaced with max date)

linktype Type of link

linkprim Link primacy

References

Ian Gow, *Financial Accounting Research*, Chapter on Identifiers: https://iangow.github.io/far_book/identifiers.html

See Also

get_compustat()

Examples

```
## Not run:
wrds <- wrds_connect()
ccm <- link_ccm(wrds)

# Join with Compustat data
compustat <- get_compustat(wrds)
compustat |>
  dplyr::inner_join(ccm, by = dplyr::join_by(gvkey)) |>
  dplyr::filter(datadate >= linkdt, datadate <= linkenddt)

wrds_disconnect(wrds)

## End(Not run)
```

---

link_ibes_crsp *Get IBES-CRSP linking table*

Description

Downloads the WRDS-provided linking table that maps IBES tickers to CRSP PERMNOs with valid date ranges and match quality scores.

Usage

```
link_ibes_crsp(wrds, max_score = 5L, n = Inf, lazy = FALSE)
```

Arguments

wrds	A DBIConnection object returned by wrds_connect().
max_score	Maximum match quality score to include. Defaults to 5, which excludes score 6 (the worst matches). Lower scores indicate better matches: • 1: Best match (CUSIP, ticker, and company name all match) • 2-5: Progressively weaker matches • 6: Worst match (excluded by default)
n	Maximum number of rows to return. Defaults to Inf (all rows). Use a smaller value (e.g., n = 100) to preview data before downloading the full table.
lazy	If TRUE, returns a lazy tbl instead of collecting. Defaults to FALSE.

Details

The linking table comes from wrdsapps_link_crsp_ibes.ibcrsphist.

To use the link, join on ticker and ensure your observation date falls within the sdate to edate range.

Value

A tibble with columns:

ticker IBES ticker

permno CRSP permanent security identifier

sdate Start date of the link

edate End date of the link

score Match quality score (1 = best, 6 = worst)

References

WRDS IBES-CRSP Linking Table Documentation: https://wrds-www.wharton.upenn.edu/documents/796/IBES_CRSP_Linking_Table_by_WRDS.pdf

See Also

link_ccm()

Examples

```
## Not run:
wrds <- wrds_connect()
ibes_link <- link_ibes_crsp(wrds)

# Join with IBES data on ticker and date range
ibes_data |>
  dplyr::inner_join(ibes_link, by = dplyr::join_by(ticker)) |>
  dplyr::filter(date >= sdate, date <= edate)
```

```
wrds_disconnect(wrds)

## End(Not run)
```

`list_subscriptions` *List subscribed data products*

Description

Returns a tibble of WRDS schemas the user has access to, with human-readable product names where available.

Usage

```
list_subscriptions(wrds)
```

Arguments

`wrds`	A `DBIConnection` object returned by `wrds_connect()`.

Value

A tibble with columns `schema` and `product`.

Examples

```
## Not run:
wrds <- wrds_connect()
list_subscriptions(wrds)
wrds_disconnect(wrds)

## End(Not run)
```

`list_tables` *List tables in a library*

Description

Returns a tibble of table names within a WRDS library (schema), with human-readable descriptions where available.

Usage

```
list_tables(wrds, library)
```

Arguments

wrds	A DBIConnection object returned by wrds_connect().
library	Character. The name of the library (schema) to query.

Value

A tibble with columns table and description.

Examples

```
## Not run:
wrds <- wrds_connect()
list_tables(wrds, "comp")
wrds_disconnect(wrds)

## End(Not run)
```

---

sic_2digit *Convert SIC codes to 2-digit industry codes*

Description

Extracts the first two characters from SIC codes to create broader industry classifications.

Usage

```
sic_2digit(sic)
```

Arguments

sic	A numeric or character vector of SIC codes.

Details

SIC codes are hierarchical: the first two digits represent major industry groups (e.g., "54" = Retail-Food Stores), while the full 4-digit code provides more specific classifications (e.g., "5412" = Retail-Convenience Stores).

Value

A character vector of 2-digit SIC codes.

Examples

```
# Convenience Stores (SIC 5412) -> Retail-Food Stores (54)
sic_2digit(5412)
# [1] "54"

sic_2digit(c(5412, 5400))
# [1] "54" "54"
```

wrds_connect *Connect to WRDS*

Description

Establishes a connection to the WRDS PostgreSQL server using credentials stored securely in the system keyring.

Usage

```
wrds_connect(user_key = "wrds_user", password_key = "wrds_pw", keyring = NULL)
```

Arguments

user_key	Name of the keyring entry storing the WRDS username. Defaults to "wrds_user".
password_key	Name of the keyring entry storing the WRDS password. Defaults to "wrds_pw".
keyring	Optional keyring name. If NULL (default), uses the default keyring.

Details

Credentials must be set up before first use with wrds_set_credentials(). The connection uses bigint = "numeric" so that 64-bit integers from PostgreSQL are returned as doubles, which avoids overflow and works well with tidyverse functions.

Value

A DBIConnection object for the WRDS PostgreSQL database.

See Also

wrds_disconnect(), wrds_set_credentials()

Examples

```
## Not run:
wrds <- wrds_connect()
list_subscriptions(wrds)
wrds_disconnect(wrds)

## End(Not run)
```

wrds_disconnect *Disconnect from WRDS*

Description

Closes a WRDS database connection.

Usage

```
wrds_disconnect(wrds)
```

Arguments

wrds	A DBIConnection object returned by wrds_connect().

Value

Invisibly returns TRUE if disconnection was successful.

Examples

```
## Not run:
wrds <- wrds_connect()
wrds_disconnect(wrds)

## End(Not run)
```

wrds_products *WRDS product catalog*

Description

A mapping of WRDS schema names to human-readable product names. Used by list_subscriptions() to enrich results, and available for users who want to look up product names directly.

Usage

```
wrds_products
```

Format

A data frame with 613 rows and 2 columns:

schema WRDS schema or product code

product Human-readable product name

Source

https://wrds-www.wharton.upenn.edu/users/products/

`wrds_set_credentials` *Set WRDS credentials*

Description

Interactively stores WRDS username and password in the system keyring for secure, persistent storage.

Usage

```
wrds_set_credentials(
  user_key = "wrds_user",
  password_key = "wrds_pw",
  keyring = NULL
)
```

Arguments

`user_key`	Name for the username keyring entry. Defaults to `"wrds_user"`.
`password_key`	Name for the password keyring entry. Defaults to `"wrds_pw"`.
`keyring`	Optional keyring name. If `NULL` (default), uses the default keyring.

Details

This function prompts for username and password interactively. Credentials are stored securely using the operating system's keyring (Keychain on macOS, Credential Manager on Windows, Secret Service on Linux).

Value

Invisibly returns `TRUE` on success.

Examples

```
## Not run:
wrds_set_credentials()

## End(Not run)
```

`wrds_update_password` *Update WRDS password*

Description

Interactively updates the WRDS password stored in the system keyring without changing the username.

Usage

```
wrds_update_password(password_key = "wrds_pw", keyring = NULL)
```

Arguments

`password_key`	Name for the password keyring entry. Defaults to `"wrds_pw"`.
`keyring`	Optional keyring name. If `NULL` (default), uses the default keyring.

Value

Invisibly returns `TRUE` on success.

See Also

`wrds_set_credentials()`, `wrds_connect()`

Examples

```
## Not run:
wrds_update_password()

## End(Not run)
```

Index