Package 'DownBallotR'

May 7, 2026

Title Access Federal, State, and Local Election Data

Version 0.1.0

Description Provides an 'R' interface for downloading and standardizing election data to support research workflows. Election results are published by states through heterogeneous and often dynamic web interfaces that are not consistently accessible through existing 'R' packages or APIs. To address this, the package wraps state-specific 'Python' web scrapers through the 'reticulate' package, enabling access to dynamic content while exposing consistent 'R' functions for querying election availability and results across jurisdictions. The package is intended for responsible use and relies on publicly accessible election result pages.

License Apache License (>= 2.0)

URL https://gchickering21.github.io/DownBallotR/, https://github.com/gchickering21/DownBallotR

BugReports https://github.com/gchickering21/DownBallotR/issues

Encoding UTF-8

Language en-US

SystemRequirements Python (>= 3.10), pip

Depends R (>= 4.1.0)

Imports purrr, reticulate, rlang

Suggests knitr, dplyr, pak, remotes, rmarkdown, testthat (>= 3.0.0), withr

Config/testthat/edition 3

RoxygenNote 7.3.3

NeedsCompilation no

Author Graham Chickering [aut, cre], Chris Warshaw [ctb]

Maintainer Graham Chickering <grahamchickering@gmail.com>

Repository CRAN

Date/Publication 2026-04-23 20:30:07 UTC

Contents

`db_available_years` *Show data availability for election scrapers*

Description

Returns a data frame listing the earliest available year for each state and scraper source tracked by DownBallotR. All sources include data through the current calendar year.

Usage

```
db_available_years(state = NULL)
```

Arguments

`state`	Optional state name to filter results (e.g. `"Virginia"`). Pass `NULL` (default) to return all states.

Value

A `data.frame` with columns `source`, `state`, `start_year`, and `end_year`.

Examples

```
# All sources
db_available_years()

# Filter to one state
db_available_years(state = "Virginia")
```

db_list_sources *List all registered Python scraper sources*

Description

List all registered Python scraper sources

Usage

```
db_list_sources()
```

Value

Character vector of source names.

db_list_states *List states supported by DownBallotR scrapers*

Description

List states supported by DownBallotR scrapers

Usage

```
db_list_states(source = NULL)
```

Arguments

source	One of the sources returned by `db_list_sources()`, or `NULL` (default) to return all states with dedicated scrapers across all sources.

Value

Named character vector of canonical state names. When `source = NULL` each element is named by its source; when a single source is given the names are omitted.

`downballot_install_python` *Install Python dependencies for downballotR*

Description

Creates/uses a named virtual environment and installs Python requirements (pandas, requests, lxml, bs4, playwright), then installs Playwright Chromium.

Usage

```
downballot_install_python(
  envname = "downballotR",
  python = NULL,
  reinstall = FALSE,
  install_chromium = TRUE,
  quiet = FALSE
)
```

Arguments

`envname`	Name of the virtualenv to create/use.
`python`	Path to a python executable to use when creating the env (optional).
`reinstall`	If TRUE, reinstall packages even if already installed.
`install_chromium`	If TRUE, install Playwright Chromium browser. In interactive sessions, the user will be prompted for explicit consent before the download (~100-200 MB) begins. In non-interactive sessions, the function will error if Chromium is missing; set `install_chromium = FALSE` to suppress this.
`quiet`	If TRUE, suppress progress messages.

Details

Python must already be installed on your system before calling this function. `downballot_install_python()` creates a virtual environment using an existing Python interpreter — it does not install Python itself. If Python is not found, `reticulate` will error with a message about being unable to create a virtualenv.

- **Windows**: Install Python from https://www.python.org/downloads/. Make sure to check "Add Python to PATH" during installation.
- **macOS**: Python 3 is available via Xcode Command Line Tools (`xcode-select --install`) or https://www.python.org/downloads/.
- **Linux**: Install via your package manager, e.g. `sudo apt install python3 python3-venv` (Debian/Ubuntu) or `sudo dnf install python3` (Fedora/RHEL).

If the environment already exists and all required packages are present, the function prints a message and returns without doing work (unless Chromium is missing and `install_chromium = TRUE`). In all cases, it attempts to initialize reticulate to the selected interpreter for this session.

Value

Called for side effects. Returns `invisible(TRUE)` on success, or `invisible(FALSE)` if the user declines the Chromium download.

`downballot_python_status`

Check Python environment status for downballotR

Description

Reports whether the Python virtual environment exists, whether reticulate is initialized (and which Python is active), which required packages are missing, and whether Playwright Chromium is available.

Usage

```
downballot_python_status(
  envname = "downballotR",
  required_pkgs = db_required_python_packages(),
  quiet = FALSE
)
```

Arguments

`envname`	Name of the virtualenv to check.
`required_pkgs`	Character vector of required Python packages. Defaults to `db_required_python_packages()`.
`quiet`	If `TRUE`, do not print. (You can still `print()` the returned object explicitly.)

Details

This function does not modify the environment.

Value

An object of class `downballot_python_status`. Invisibly when `quiet = FALSE`.

downballot_use_python *Use the DownBallotR Python virtualenv in this R session*

Description

Pins reticulate to the package's virtualenv for the current R session. If reticulate is already initialized to a different interpreter, this errors with a clear message (reticulate cannot switch interpreters mid-session).

Usage

```
downballot_use_python(envname = "downballotR")
```

Arguments

envname	Name of the virtualenv to use.

Value

Invisibly TRUE on success.

print.downballot_python_status *Print a downballot_python_status object*

Description

Print a downballot_python_status object

Usage

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

Arguments

x	A downballot_python_status object.
...	Further arguments passed to or from other methods (unused).

Value

Invisibly returns x, the downballot_python_status object passed in, following the S3 print method convention.

scrape_elections *Scrape election data*

Description

A single entry point that automatically routes to the appropriate scraper based on `state`. Use `db_list_states("election_stats")` to see states supported by the general-election scraper.

Usage

```
scrape_elections(
  state = NULL,
  year_from = NULL,
  year_to = NULL,
  level = c("all", "state", "county", "precinct", "town", "parish"),
  parallel = TRUE,
  max_workers = 4L,
  include_vote_methods = FALSE
)
```

Arguments

`state`	State name or 2-letter abbreviation, accepted in any case or spacing style (e.g. `"VA"`, `"virginia"`, `"Virginia"`, `"south_carolina"`, `"SC"`). The value is normalised automatically before being passed to the underlying scraper, so callers do not need to worry about the exact format.
`year_from`	Start year, inclusive (default `NULL`). When `NULL`, ElectionStats starts at `1789`; all state-portal scrapers apply no lower bound (data is clamped to each scraper's earliest confirmed year).
`year_to`	End year, inclusive (default `NULL`). When `NULL`, the current calendar year is used as the upper bound.
`level`	What to return. `"all"` (default) returns a named list with `$state`, `$county`, and (when available) `$precinct` data frames (ElectionStats); `$state` and `$county` data frames (Georgia / Utah / Indiana); `$state` and `$town` data frames (Connecticut); `$state` and `$parish` data frames (Louisiana); or `$precinct`, `$county`, and `$state` data frames (NC). `"state"` returns statewide candidate-level results only; `"county"` returns county vote breakdowns (ElectionStats / Georgia / Utah / Indiana); `"precinct"` returns precinct-level vote breakdowns — columns: `state`, `election_id`, `candidate_id`, `county`, `precinct`, `candidate`, `votes` (ElectionStats classic states: CO, MA, ID; v2 states: SC, NM, VA; NC via `state="NC"`; Georgia and Utah — navigates county pages to find and scrape each precinct, columns: `state`, `election_name`, `election_type`, `election_year`, `election_date`, `office_level`, `office`, `district`, `county`, `precinct`, `candidate`, `party`, `votes`, `vote_pct` (write-ins are excluded, so column may not sum to 100\ `precinct_winner`, `url`); `"town"` returns town-level results only (Connecticut); `"parish"` returns parish-level results only (Louisiana).

`parallel` (ElectionStats) Use parallel county scraping for classic (requests-based) states (default TRUE). Ignored automatically for Playwright-based states (SC, NM, NY, VA).

`max_workers` (Georgia / Utah / Connecticut / Louisiana) Maximum number of parallel Chromium browsers (default 4L). For Georgia and Utah, controls county-level parallelism; for Connecticut, controls town-level parallelism; for Louisiana, controls parish-level parallelism (default is capped at 2 for LA). Ignored for all other states.

`include_vote_methods` (Georgia only) If TRUE, also return a vote-method breakdown table (Advance in Person, Election Day, Absentee by Mail, Provisional) for Georgia results (default FALSE). Ignored for all other states.

Details

Routing rules (applied in order):

1. `state` matches North Carolina (e.g. `"NC"`, `"north_carolina"`) → NC State Board of Elections scraper (2000–present).
2. `state` matches Connecticut (e.g. `"CT"`, `"connecticut"`) → Connecticut CTEMS scraper (2016–present).
3. `state` matches Georgia (e.g. `"GA"`, `"georgia"`) → Georgia Secretary of State scraper (2000–present).
4. `state` matches Utah (e.g. `"UT"`, `"utah"`) → Utah election results scraper (2023–present).
5. `state` matches Indiana (e.g. `"IN"`, `"indiana"`) → Indiana General Election results scraper (2019–present).
6. `state` matches Louisiana (e.g. `"LA"`, `"louisiana"`) → Louisiana Secretary of State scraper (1982–present).
7. All other states → ElectionStats multi-state scraper.

Value

A `data.frame`, or a named list when `level = "all"`: `$state` + `$county` (+ `$precinct` when available) for ElectionStats; `$state` + `$county` + `$precinct` for Georgia / Utah (or just `$state` / `$county` / `$precinct` alone when the corresponding `level` is specified); `$state` + `$county` for Indiana; `$state` + `$town` for Connecticut; `$state` + `$parish` for Louisiana; `$precinct` + `$county` + `$state` for North Carolina. Each component is also assigned directly into the calling environment (e.g. `ga_state`, `ga_county`) when `level = "all"`.

Examples

```
# General election results — Virginia
df <- scrape_elections(state = "virginia", year_from = 2023, year_to = 2023,
                       level = "state")

# General election results — Virginia, both state and county levels
res <- scrape_elections(state = "virginia", year_from = 2023, year_to = 2023)
res$state   # candidate-level data frame
res$county  # county vote breakdown data frame
```

```
# North Carolina — single year
df <- scrape_elections(state = "NC", year_from = 2024, year_to = 2024)

# Connecticut — statewide + town results for 2024
res <- scrape_elections(state = "CT", year_from = 2024, year_to = 2024)
res$state  # statewide totals
res$town   # town-level results

# Connecticut — statewide only (faster; no town scraping)
df <- scrape_elections(state = "CT", year_from = 2024, year_to = 2024,
                       level = "state")

# Connecticut — with more parallel workers
res <- scrape_elections(state = "CT", year_from = 2022, year_to = 2022,
                        max_workers = 4L)

# Georgia — statewide + county results
res <- scrape_elections(state = "GA", year_from = 2024, year_to = 2024)

# Georgia — statewide only (faster)
df <- scrape_elections(state = "GA", year_from = 2024, year_to = 2024,
                       level = "state")

# Georgia — with vote-method breakdown
res <- scrape_elections(state = "GA", year_from = 2024, year_to = 2024,
                        include_vote_methods = TRUE)

# Utah — statewide + county results
res <- scrape_elections(state = "UT", year_from = 2024, year_to = 2024)

# Indiana — General Election results (statewide + county)
res <- scrape_elections(state = "IN", year_from = 2024, year_to = 2024)
res$state   # statewide candidate totals
res$county  # county-level breakdown

# Indiana — statewide only (faster)
df <- scrape_elections(state = "IN", year_from = 2022, year_to = 2022,
                       level = "state")

# Louisiana — statewide + parish results
res <- scrape_elections(state = "LA", year_from = 2024, year_to = 2024)
res$state   # statewide candidate totals
res$parish  # parish-level breakdown

# Louisiana — statewide only (faster; skips parish scraping)
df <- scrape_elections(state = "LA", year_from = 2023, year_to = 2023,
                       level = "state")
```

---

summarize_results *Summarize an election results data frame*

Description

Computes aggregate statistics for a data frame of election results. The state is detected automatically from the `state` column when present, or from the variable name (e.g. `ga_results` -> "Georgia").

Usage

```
summarize_results(df, state = NULL)
```

Arguments

`df`	A data frame returned by `scrape_elections`.
`state`	Optional two-letter state abbreviation or full state name. Overrides auto-detection when supplied.

Value

A named list (printed on call) with:

`state` Detected or supplied state name.

`years` Integer vector of election years present.

`n_years` Number of distinct election years.

`n_elections` Number of distinct elections.

`n_candidates` Number of distinct candidate names.

`office_level_breakdown` Named integer vector: distinct elections by level (Federal / State / Local).

`offices_by_level` Named list: distinct office names per level.

Examples

```
ga_results <- scrape_elections("GA", 2020, 2024)
summarize_results(ga_results)
```

Index