Package 'cvpa'

November 13, 2026

Type Package

Title Canadian Voting and Policy Attitudes

Version 0.3.0

Description The main focus of the package is to give weighted party support and issue attitudes by demographic groups over time. As of 0.2.0, policy-issue and vote data are read on demand from the cp3 project's hosted Parquet files (see update_cvpa_data()) rather than bundled with the package, covering several hundred policy questions across seven issue domains in addition to vote choice/intention. The vote data are described in Lucas, Romualdi, Armstrong, and Goodyear-Grant (2025) <doi:10.1017/S0008423925000241>.

License MIT + file LICENSE

URL https://policyattitudes.ca/app/, https://github.com/lucasjacklucas/cvpa_dataset

BugReports https://github.com/lucasjacklucas/cvpa_dataset/issues

Depends R (>= 3.5.0)

Imports broom, dplyr, nanoparquet, rio, rlang, stats, tidyr

Suggests ggplot2, knitr, rmarkdown, testthat (>= 3.1.5), withr

VignetteBuilder knitr

Encoding UTF-8

Config/roxygen2/version 8.0.0

Config/testthat/edition 3

NeedsCompilation no

Author Dave Armstrong [aut, cre] (ORCID: <https://orcid.org/0000-0002-9358-2489>), Jack Lucas [aut] (ORCID: <https://orcid.org/0000-0002-3498-7267>), Tyler Romualdi [aut]

Maintainer Dave Armstrong <davearmstrong.ps@gmail.com>

Repository CRAN

Date/Publication 2026-11-13 16:40:22 UTC

Contents

cvpa_cache_dir 2
cvpa_example_url 3
cvpa_grouping_vars 3
demographic_vars 4
gap_analysis 5
make_flag 6
question_catalog 7
question_data 8
question_response_labels 9
response_gap_analysis 10
update_cvpa_data 11
vote_data 12
wtd_response 13
wtd_vote 15

Index **17**

`cvpa_cache_dir` *Local cache directory used by cvpa*

Description

Where [update_cvpa_data()] stores downloaded Parquet files and every other cvpa function reads them from. Defaults to a per-user cache directory managed by R (see [tools::R_user_dir()]); override with ‘options(cvpa.cache_dir = ...)‘.

Usage

```
cvpa_cache_dir()
```

Value

The cache directory path (created if it doesn’t exist yet).

Examples

```
# Point the cache at a temporary directory (the default is a per-user
# directory under tools::R_user_dir("cvpa", "cache")).
old <- options(cvpa.cache_dir = tempfile("cvpa-cache-"))
cvpa_cache_dir()
unlink(cvpa_cache_dir(), recursive = TRUE)
options(old)
```

cvpa_example_url *URL of the small example dataset bundled with cvpa*

Description

cvpa ships a tiny copy of the hosted cp3 Parquet layout – a handful of questions (‘bri_econ_013‘, ‘env_econ_027‘, ‘vote_choice‘, ‘vote_intention‘) and a random sample of respondents from each survey year – so that the examples in the help pages can run without a network connection. Point cvpa at it by setting the ‘cvpa.data_url‘ option to this URL (and, so you don’t touch your real cache, ‘cvpa.cache_dir‘ to a temporary directory), then call [update_cvpa_data()] as usual.

Usage

```
cvpa_example_url()
```

Details

The example data is for illustration only: it is a subset, and estimates from it are not estimates from the full data. Because it is not the official data, [update_cvpa_data()] and the functions that read the cache warn while it is in use. Undo with ‘options(cvpa.data_url = NULL)‘, or restore the options you saved.

Value

A ‘file://‘ URL (with a trailing slash) for the bundled example data, suitable for ‘options(cvpa.data_url = ...)‘.

Examples

```
old <- options(
  cvpa.data_url = cvpa_example_url(),
  cvpa.cache_dir = tempfile("cvpa-cache-")
)
update_cvpa_data(questions = "bri_econ_013", quiet = TRUE)  # warns: example data
head(question_catalog(domain = "econ"))
unlink(getOption("cvpa.cache_dir"), recursive = TRUE)
options(old)
```

cvpa_grouping_vars *Grouping variables available for any question*

Description

Every table returned by [question_data()] carries these columns in addition to ‘response‘, ‘weight‘, and ‘year‘ – eleven demographic variables (see [demographic_vars] for what each one measures) plus ‘vote_intention‘ and ‘vote_choice‘, which travel with every question’s data and can be used as grouping variables just like a demographic (this is what lets you, say, break a policy question down by how people voted). Used to validate the ‘grouping_vars‘/‘grouping_var‘ argument in [wtd_response()] and [response_gap_analysis()].

Usage

```
cvpa_grouping_vars
```

Value

A character vector (13 elements) of the column names that can be used as grouping variables: ‘vote_intention‘, ‘vote_choice‘ and the eleven demographic variables in [demographic_vars].

See Also

[demographic_vars], [question_data()]

Examples

```
cvpa_grouping_vars
```

---

demographic_vars *Demographic variables available on every question’s response data*

Description

Every table [question_data()] returns carries these eleven demographic columns (denormalized onto each response row by cp3’s data pipeline), in addition to ‘vote_intention‘ and ‘vote_choice‘, which are also always present and usable as grouping variables in their own right (see [cvpa_grouping_vars]). Category lists are given in the order the underlying ‘code_labels‘ table sorts them, which is also the factor level order [question_data()] applies.

Usage

```
demographic_vars
```

Format

A data frame with 11 rows and 3 variables:

variable Column name as it appears in [question_data()]’s output

description What the variable measures

categories Ordered category labels, separated by ‘"; "‘

Value

A data frame with one row for each of the eleven demographic variables and the columns ‘variable‘, ‘description‘ and ‘categories‘ (described under Format).

See Also

[cvpa_grouping_vars], [question_data()]

Examples

```
demographic_vars
```

`gap_analysis` *Analyze Significance of Voting Gaps*

Description

The function does a difference of proportions test for the two identified groups for each of the parties. Note, for gender, there are not many surveys with non-binary/other response options so those are removed from consideration here.

Usage

```
gap_analysis(
  data,
  vote_type = c("intention", "choice"),
  incl_undecided = FALSE,
  years = NULL,
 grouping_var = c("age_cats", "religion", "degree", "gender", "province", "region",
    "language", "union_household", "community_size", "occupation"),
  levels = NULL,
  ...
)
```

Arguments

`data`	Data from inst/extdata/integrated_with_weights.RDS.
`vote_type`	Use vote intention or choice
`incl_undecided`	If using vote intention, should undecided voters be included?
`years`	Select desired years. ‘NULL‘ (default) uses every year in ‘data‘ – so newly added years are never silently dropped.
`grouping_var`	For which group do you want separate estimates (gender, degree, union_household, community_size)
`levels`	Character vector of length 2 giving values of ‘grouping_var‘ to use for comparison. ‘NULL‘ is valid if ‘grouping_var‘ only has two valid values. Otherwise, values must be specified.
`...`	Other arguments to be passed down, not implemented

Details

See [response_gap_analysis()] for the same test generalized to any policy question, not just vote choice/intention.

Value

A data frame (a tibble, grouped by ‘year‘) with one row per year and party for which both groups have enough respondents to test. The columns are ‘year‘; ‘vote‘, the party; ‘cnt_<level>‘ and ‘n_<level>‘ for each of the two groups (for example ‘cnt_Man‘ and ‘n_Woman‘), the weighted count of the group’s respondents who chose that party and the weighted count of the group’s respondents in that year; ‘flag‘ (‘"p-val reliable"‘ or ‘"p-val unreliable"‘, see [make_flag()]); ‘estimate1‘ and ‘estimate2‘, the party’s weighted share in the first and second group; ‘p.value‘, from a two-sample test of equal proportions ([stats::prop.test()]); ‘diff‘, ‘estimate2 - estimate1‘; and ‘conf.low‘ and ‘conf.high‘, the limits of the 95

Examples

```
data(vote_data)
gap_analysis(vote_data, "intention", years=2000:2022, grouping_var="gender")
```

`make_flag` *Make Reliability Flag for Weighted Proportions Test*

Description

Makes a flag to identify if the p-value for the difference of proportions test is reliable (all expected counts greater than 5).

Usage

```
make_flag(x, n)
```

Arguments

`x`	A vector of two values of category counts.
`n`	A vector of two values of sample sizes

Value

‘"p-val reliable"‘ if every expected count is at least 5, otherwise ‘"p-val unreliable"‘.

Examples

```
make_flag(c(30, 40), c(100, 120))   # large counts: reliable
make_flag(c(2, 3), c(10, 20))       # small counts: unreliable
```

question_catalog *List available policy and vote questions*

Description

One row per question, deduplicated across the several hundred survey-wave rows each question has in the underlying catalog (see [update_cvpa_data()]). Covers every domain in the cp3 project, including ‘"vote"‘ (‘vote_choice‘/‘vote_intention‘) – from cvpa’s point of view vote is just one more question, decoded and analyzed the same way as any policy question (see [question_data()], [wtd_response()]).

Usage

```
question_catalog(domain = NULL, type = NULL)
```

Arguments

domain	Optional character vector of domain codes to restrict to (see the ‘domain‘ column of this function’s own output for valid values, e.g. ‘"econ"‘, ‘"soim"‘, ‘"vote"‘).
type	Optional character vector of question types to restrict to: ‘"b"‘ (binary), ‘"lsm"‘ (3-point less/same/more-style), ‘"nsa"‘ (3-point never/sometimes/always-style), or ‘"party"‘ (vote_choice, vote_intention).

Value

A data frame with one row per question: ‘question‘, ‘domain‘, ‘domain_label‘, ‘issue_label‘, ‘question_wording‘, ‘value_labels‘ (comma-separated, ascending – see [question_response_labels()] for a decoded, one-row-per-category version), ‘type‘, and ‘n_waves‘ (how many survey waves asked this question).

Examples

```
# Uses the small example data bundled with cvpa (see ?cvpa_example_url),
# in a temporary cache, so this runs offline and leaves your own cache
# alone. For real analysis, skip the options() call and use the default,
# the official live data.
old <- options(
  cvpa.data_url = cvpa_example_url(),
  cvpa.cache_dir = tempfile("cvpa-cache-")
)
update_cvpa_data(quiet = TRUE)
question_catalog(domain = "econ")
question_catalog(type = "party")

unlink(getOption("cvpa.cache_dir"), recursive = TRUE)
options(old)
```

question_data *Fetch and decode one question's respondent-level data*

Description

Reads the locally cached 'resp/question=<question>/data_0.parquet' file (see [update_cvpa_data()]) and, by default, decodes 'response' and every demographic/vote-grouping column (see [cvpa_grouping_vars]) into labeled factors. Works identically for a policy question and for '"vote_choice"'/'"vote_intention"' – vote is just one more question.

Usage

```
question_data(question, years = NULL, decode = TRUE)
```

Arguments

question	A question code from [question_catalog()], e.g. '"bri_econ_013"' or '"vote_choice"'.
years	Optional integer vector to filter 'year' to.
decode	If 'TRUE' (default), decode 'response' and the demographic/vote columns into labeled factors; if 'FALSE', return the raw numeric codes as stored in the Parquet file.

Value

A data frame with one row per respondent-question-wave: 'response', 'weight', 'year', 'response_id', 'survey_id', and the columns in [cvpa_grouping_vars].

Examples

```
# Uses the small example data bundled with cvpa (see ?cvpa_example_url),
# in a temporary cache, so this runs offline and leaves your own cache
# alone. For real analysis, skip the options() call and use the default,
# the official live data.
old <- options(
  cvpa.data_url = cvpa_example_url(),
  cvpa.cache_dir = tempfile("cvpa-cache-")
)
update_cvpa_data(questions = "bri_econ_013", quiet = TRUE)
d <- question_data("bri_econ_013")
head(d)
table(d$year)

unlink(getOption("cvpa.cache_dir"), recursive = TRUE)
options(old)
```

question_response_labels *Response category labels for one question*

Description

A decoded, one-row-per-category view of what [question_catalog()] reports as a single comma-separated ‘value_labels‘ string (or, for ‘vote_choice‘/‘vote_intention‘, of the relevant rows of ‘code_labels.parquet‘) – the same table [question_data()] uses internally to turn a question's raw ‘response‘ codes into a labeled factor.

Usage

```
question_response_labels(question)
```

Arguments

question	A question code, e.g. ‘"bri_econ_013"‘ or ‘"vote_choice"‘.

Value

A data frame with ‘code‘, ‘label‘, and ‘sort_order‘, ordered by ‘sort_order‘.

Examples

```
# Uses the small example data bundled with cvpa (see ?cvpa_example_url),
# in a temporary cache, so this runs offline and leaves your own cache
# alone. For real analysis, skip the options() call and use the default,
# the official live data.
old <- options(
  cvpa.data_url = cvpa_example_url(),
  cvpa.cache_dir = tempfile("cvpa-cache-")
)
update_cvpa_data(quiet = TRUE)
question_response_labels("bri_econ_013")
question_response_labels("vote_choice")

unlink(getOption("cvpa.cache_dir"), recursive = TRUE)
options(old)
```

`response_gap_analysis` *Analyze Significance of Response Gaps for any question*

Description

Generalizes [gap_analysis()] to any question's data (see [question_data()]), not just vote choice/intention: a difference of proportions test between two levels of ‘grouping_var‘, for each response category, for each year.

Usage

```
response_gap_analysis(
  data,
  grouping_var,
  years = NULL,
  levels = NULL,
  avail_grps = cvpa_grouping_vars,
  response_var = "response"
)
```

Arguments

`data`	A data frame as returned by [question_data()].
`grouping_var`	Which grouping variable to compare two levels of; must be one of [cvpa_grouping_vars] (or ‘avail_grps‘, if you pass a narrower/different list).
`years`	Select desired years. ‘NULL‘ (default) uses every year in ‘data‘.
`levels`	Character vector of length 2 giving values of ‘grouping_var‘ to use for comparison. ‘NULL‘ is valid if ‘grouping_var‘ only has two valid values in ‘data‘; otherwise values must be specified.
`avail_grps`	Grouping variables considered valid; defaults to [cvpa_grouping_vars].
`response_var`	Name of the column in ‘data‘ holding the response category; defaults to ‘"response"‘.

Value

A data frame with one row per (year, response category): the two groups' counts and sample sizes (suffixed with each group's label), a reliability ‘flag‘ (see [make_flag()]), the two estimated proportions, a p-value, a confidence interval for the difference, and ‘diff‘.

Examples

```
# Uses the small example data bundled with cvpa (see ?cvpa_example_url),
# in a temporary cache, so this runs offline and leaves your own cache
# alone. For real analysis, skip the options() call and use the default,
# the official live data.
old <- options(
```

```
  cvpa.data_url = cvpa_example_url(),
  cvpa.cache_dir = tempfile("cvpa-cache-")
)
update_cvpa_data(questions = "bri_econ_013", quiet = TRUE)
d <- question_data("bri_econ_013")
response_gap_analysis(d, grouping_var = "gender")

unlink(getOption("cvpa.cache_dir"), recursive = TRUE)
options(old)
```

`update_cvpa_data` *Download or refresh cvpa's local cache of the cp3 policy-attitude data*

Description

cvpa reads policy-attitude and vote data from Parquet files hosted alongside the cp3 web app, not from data bundled in the package. Call this first (and again whenever you want the latest data) before using [question_catalog()], [question_data()], [wtd_response()], or [response_gap_analysis()].

Usage

```
update_cvpa_data(questions = NULL, force = FALSE, quiet = FALSE)
```

Arguments

`questions`	Which questions' respondent-level data to fetch, in addition to the catalog: 'NULL' (default) fetches none, or a character vector of specific question codes (see [question_catalog()]). There is deliberately no "fetch everything" option – if you want the full dataset, download 'dataset.parquet' from <https://github.com/lucasjacklucas/cvpa_dataset> instead.
`force`	If 'FALSE' (default), question files already in the cache are left alone. If 'TRUE', every requested file is re-downloaded.
`quiet`	Suppress progress messages.

Details

The question catalog ('vbl_data.parquet') and the demographic/party value-label table ('code_labels.parquet') are small and are always refreshed. Per-question respondent-level data ('resp/question=<code>/data_0.parquet') is only fetched for the questions you name in 'questions' – there are several hundred of these across all domains, so nothing downloads them until you ask.

Downloads are written to a temporary file and checked to be valid Parquet before they replace anything in the cache, so a failed or interrupted download never leaves a corrupt file behind or overwrites a good one. If the catalog or the code labels can't be downloaded, this stops with an error (and your existing cached copies are untouched). If an individual *question* file can't be downloaded – e.g. a question that is listed in the catalog but has no respondent-level data available – it is skipped with a warning naming the question(s), and the rest are still fetched.

Data comes from the official live data, ‘https://policyattitudes.ca/cp3_parquet/‘, unless you have changed ‘options(cvpa.data_url = ...)‘ (for example to a local mirror, or to the small example data from [cvpa_example_url()]). If you have, this warns on every call and the functions that read the cache warn too, because the data may be a subset or out of date; ‘options(cvpa.data_url = NULL)‘ restores the official URL.

Value

Invisibly, ‘TRUE‘, with an attribute ‘"skipped"‘ holding the question codes (if any) that could not be downloaded.

Examples

```
# These examples use the tiny example data bundled with cvpa, in a
# temporary cache, so they run offline and leave your own cache alone.
# For real analysis, skip the options() lines: by default cvpa downloads
# the official live data.
old <- options(
  cvpa.data_url = cvpa_example_url(),
  cvpa.cache_dir = tempfile("cvpa-cache-")
)

update_cvpa_data()                                  # just the catalog
update_cvpa_data(questions = "bri_econ_013")        # + one question
update_cvpa_data(questions = c("vote_choice", "vote_intention"))
dir(cvpa_cache_dir(), recursive = TRUE)

unlink(getOption("cvpa.cache_dir"), recursive = TRUE)
options(old)
```

---

vote_data *Vote Choice/Intention Data*

Description

Data on vote choice and intention from several surveys. The data are in long form (in intentnion/choice) and aggregate to profile across the variables ‘region‘-‘language‘ described below.

Format

A data frame with 78,614 rows and 11 variables

vote Factor indicating vote choice or intention depending on `type`

type String indicating vote whever `vote` is intention or actual choice

year Year of survey

region Factor indicating region of residence: Atlantic, Quebec, Ontario, Prairies, BC, The North

province Factor indicating province of residence: Newfoundland and Labrador, PEI, Nova Scotia, New Brunswick, Quebec, Ontario, Manitoba, Saskatchewan, Alberta, BC, Yukon, Northwest Territories, Nunavut

community_size Factor identifying community size of respondent's residence: < 100,000, >= 100,000

gender Factor identifying respondent gender Man, Woman. Non-binary category begins in 2019 and is available in the full data on the Canadian Opinion Research Archive.

age_cats Factor identifying respondent's age category: 18-29, 30-49, 50+

religion Factor giving respondent's religious affiliation: Catholic, Protestant, Jewish, Other, None

degree Factor indicating respondent's level of education: No Degree, University Degree

union_household Factor identifying union membership: Non-union Household, Union Household

language Language spoken by respondent - English, French, English + French, Other

occupation Respondent's Occupation: Managers/Executives/Business Owners, Professional, Routine Non-Manual, Working Class, Armed Forces, Farmer/Fisherman, Retired, Unemployed, Student, Other (incl. housewife, homemaker)

weight Sum of weights for demographic profile

Details

Fill in some details

References

Lucas, Jack, Tyler Romualdi, David A. Armstrong II, and Elizabeth Goodyear-Grant. 2025. "Demographic Divides in Canadian Voting, 1945–2023: The Canadian Vote Intention Dataset." *Canadian Journal of Political Science* 58 (3): 666–80. doi:10.1017/S0008423925000241.

Examples

```
data(vote_data)
head(vote_data)
table(vote_data$type)
```

`wtd_response` *Weighted proportion of each response category, by year and group*

Description

The general engine behind [wtd_vote()]: given a data frame with a 'response' column (any set of categories – vote choice, a policy question's answer categories, whatever 'response_var' points at), 'weight', 'year', and zero or more grouping columns, compute each response category's weighted share of the total for every year/grouping-variable combination. [question_data()] produces data in exactly this shape (its 'response' column decoded to a factor by question type), so this works identically for a policy question or for vote choice/intention – see [response_gap_analysis()] for the significance-test analog.

Usage

```
wtd_response(
  data,
  years = NULL,
  grouping_vars = NULL,
  avail_grps = cvpa_grouping_vars,
  response_var = "response"
)
```

Arguments

data	A data frame as returned by [question_data()] (or any data frame with the same shape: ‘response_var‘, ‘weight‘, ‘year‘, plus whichever of ‘avail_grps‘ you want to group by).
years	Select desired years. ‘NULL‘ (default) uses every year in ‘data‘.
grouping_vars	For which groups do you want separate estimates; must be a subset of ‘avail_grps‘.
avail_grps	Grouping variables considered valid; defaults to [cvpa_grouping_vars].
response_var	Name of the column in ‘data‘ holding the response category. Defaults to ‘"response"‘, the column [question_data()] produces.

Value

A data frame with ‘year‘, ‘grouping_vars‘, the response category (named ‘response_var‘), ‘count‘ (summed weight), and ‘response_share‘ (that category’s share of the year/group total).

Examples

```
# Uses the small example data bundled with cvpa (see ?cvpa_example_url),
# in a temporary cache, so this runs offline and leaves your own cache
# alone. For real analysis, skip the options() call and use the default,
# the official live data.
old <- options(
  cvpa.data_url = cvpa_example_url(),
  cvpa.cache_dir = tempfile("cvpa-cache-")
)
update_cvpa_data(questions = "bri_econ_013", quiet = TRUE)
d <- question_data("bri_econ_013")
wtd_response(d, grouping_vars = "gender")

unlink(getOption("cvpa.cache_dir"), recursive = TRUE)
options(old)
```

`wtd_vote` *Make weighted mean of vote intention/choice*

Description

Make weighted mean of vote intention/choice

Usage

```
wtd_vote(
  data,
  vote_type = c("intention", "choice"),
  incl_undecided = FALSE,
  years = NULL,
  grouping_vars = NULL,
  ...
)
```

Arguments

`data`	Data from inst/extdata/integrated_with_weights.RDS.
`vote_type`	Use vote intention or choice
`incl_undecided`	If using vote intention, should undecided voters be included?
`years`	Select desired years. 'NULL' (default) uses every year in 'data' – so newly added years are never silently dropped.
`grouping_vars`	For which groups do you want separate estimates (region or province, gender, age_cats, degree, religion, language, union_household, community_size)
`...`	Other arguments to be passed down, not implemented

Details

Since 0.2.0, this is a thin wrapper around the general [wtd_response()] engine that also powers policy-question analysis (see [question_data()]) – its inputs, outputs, and validation behavior are unchanged from earlier versions.

Value

A data frame (a tibble) with one row per combination of year, party and, if 'grouping_vars' is given, each combination of the grouping variables. Its columns are 'year'; 'vote', the party (a factor); one column for each of 'grouping_vars'; 'count', the sum of the survey weights of the respondents in that cell; and 'party_support', that party's weighted share of the votes counted in the same year and group, so the shares sum to 1 within each year and group. Respondents with a missing vote are never counted, and undecided respondents are left out unless 'incl_undecided = TRUE'.

Examples

```
data(vote_data)
wtd_vote(vote_data, "intention", years=2000:2022)
```

Index

∗ **datasets**
vote_data, 12

cvpa_cache_dir, 2
cvpa_example_url, 3
cvpa_grouping_vars, 3

demographic_vars, 4

gap_analysis, 5

make_flag, 6

question_catalog, 7
question_data, 8
question_response_labels, 9

response_gap_analysis, 10

update_cvpa_data, 11

vote_data, 12

wtd_response, 13
wtd_vote, 15