`cesR`: A package to make using the Canadian Election Study a little easier*

Paul A. Hodgetts, University of Toronto Rohan Alexander, University of Toronto

27 August 2020

Abstract

TBD

1 Introduction

An election season brings with it a stream of predictions made using a variety of resources and methods. While election predictions can provide insight as to the direction of an election and voter intention during an election, a retrospective on an election can be equally enlightening. Survey data of this sort can provide a better understanding of an election after-the-fact and can provide clarity as to the changes in the political climate over the years. The Canadian Election Study (CES) is a series of large-scale surveys conducted every election year that seeks to provide such a retrospective understanding to Canadian elections. By providing data on Canadian elections since 1965 up to, most recently, 2019 the CES provides insight into such topics as the intentions of voters, what issues voters deem important, and the perception of parties and candidates (Canadian Election Study, 2019). For example, Bittner (2018) used CES data to assess the level of presidentialization experienced in Canadian elections over time; while other studies have used CES data to assess the level of Islamophobia in Canada (Wilkins-Laflamme, 2018), how anti-party rhetoric affects voter behaviour (Gidengil et al., 2001), and how political trust influences voter behaviour (Belanger and Nadeau, 2005). The collection of and access to such data is integral to this research. The R package `cesR` ('caesar'), as presented in this paper, seeks to complement the work of the CES, and enhance social science research and the teaching of statistical processes using survey data by providing a means for R users to easily access CES survey datasets.

The CES survey datasets are publicly available from various repositories across the internet in various data types (Canadian Election Study, 2019; UBC, 2015; ODESI, 2020). As the ability to find and access data is integral to quality research in any domain, the `cesR` package brings together these various sources, providing R users with a means of easily accessing each CES survey dataset while also circumventing the need to download and store survey data files. The structure of `cesR` also removes the need to call a data file from a directory, thereby making it easier to work with CES datasets between different workstations. Additionally, `cesR` provides a function, `get_decon()`, that creates a subset of the 2019 CES online survey that has been prepared in an opinionated way. This subset of the 2019 CES online survey provides a tool for educators that can be used to aid in the teaching of statistical analysis of large survey datasets.

The `cesR` package is built using the R programming language and is designed to work within the R integrated development environment (IDE) RStudio (R Core Team, 2020; RStudio Team, 2020). Providing access to data is common in R packages and the `cesR` package follows and was inspired by the work being done in the R community through such packages as the `opendatatoronto` package (Gelfand, 2020), the `Lahman` package

*We thank **THANKS**. Our code and datasets are available at: **LINK**. Comments on the 27 August 2020 version of this paper are welcome at: rohan.alexander@utoronto.ca. For the latest version please click here.

(Friendly et al., 2020), `fueleconomy` (Wickham, 2020), and `nasaweather` (Wickham', 2014). Furthermore, the `cesR` package is complementary to packages such as `dataverse` (Leeper, 2017) and `cancensus` (von Bergmann et al., 2020) that provide R users access to Canadian survey and census data. Packages such as these are important to R users as they improve the functionality of working within R by minimizing the number of steps required to load data and increasing the availability of data to R users. The `cesR` package contributes to this area of R package development and the utility of R by providing access to CES survey data while also providing a tool for educators.

This paper introduces the `cesR` package and its functions `get_ces()`, `get_cescodes()`, `get_question()`, `get_preview()`, and `get_decon()`. In addition to discussing the construction of each function, examples and vignettes as to common use-cases are provided.

2 Package Functions

The `cesR` package has five functions that allow R users to download and explore the CES survey datasets. CES survey datasets can be downloaded using either the `get_ces()` function or the `get_preview()`. Where `get_ces()` provides users with complete CES survey datasets, `get_preview()` cr eates truncated datasets for initial exploratory analysis. Datasets can be further explored using the `get_question()` function, which returns the survey question associated with a given column in a given dataset. CES survey datasets are called using designated character string arguments. The helper function `get_cescodes()` provides users a convenient means of looking up the survey calls within the R workspace. Lastly, the `get_decon()` provides users with a pre-created subset of the 2019 CES online survey that can be used to assist with the education of the exploratory analysis of survey data.

2.1 `get_ces()`

When called, the `get_ces()` function returns a requested CES survey as a data object and prints to the console the associated citation and URL for the survey dataset repository. The function takes one argument in the form of a character string. This argument is a vector member that has been associated with a CES survey through the body of code in the `get_ces()` function that when used calls the download URL for that survey on an associated GitHub repository named `ces_data`. If the provided character string argument matches a member of the built-in vector `ces_codes`, the associated file is downloaded using the `download.file()` function from the `utils` R package (R Core Team, 2020) as a compressed .zip folder and is stored temporarily in `inst/extdata` directory in the greater package directory. Upon downloading the file, the compressed folder is unzipped using the `unzip()` function from the `utils` R package (R Core Team, 2020) and read into R using either the `read_dta()` or `read_sav()` functions from the `haven` R package (Wickham and Miller, 2020) depending on the file extension of the downloaded file. A data frame is then assigned using the `assign()` function from the `base` R package (R Core Team, 2020) as a data object in the global environment. The downloaded file and file directory are then removed from the computer using the `unlink()` function from the `base` R package (R Core Team, 2020). Finally, the recommended citation for the requested survey dataset and URL of the survey data storage location are printed in the console (see *2.1.1* ***`get_ces()`*** *basics* for an example of a basic `get_ces()` function call).

If the provided character string argument does not have a match in the built-in vector, then the function process is stopped and a warning message stating `Error in get_ces(): Warning: Code not in table` is printed in the RStudio console (see *2.1.2* ***`get_ces()`*** *error*).

2.1.1 `get_ces()` basics

```
# install the cesR package from GitHub
devtools::install_github("hodgettsp/cesR")
```

```
# load the cesR package into RStudio
library(cesR)

# call the 2019 CES online survey
get_ces("ces2019_web")

TO CITE THIS SURVEY FILE: Stephenson, Laura B; Harell, Allison; Rubenson, Daniel;
Loewen, Peter John, 2020, '2019 Canadian Election Study -
Online Survey', https://doi.org/10.7910/DVN/DUS88V, Harvard Dataverse, V1

LINK: https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/DUS88V
```

2.1.2 `get_ces()` error

```
# install the cesR package from GitHub
devtools::install_github("hodgettsp/cesR")

# load the cesR package into RStudio
library(cesR)

# call the 2019 CES online survey
get_ces("2019ces_web")

Error in get_ces("2019ces_web") : Warning: Code not in table.
```

The character string argument calls for each CES survey are provided in the `get_cescodes()` function discussed in the *2.2 Supporting functions* section.

The structure of `get_ces()` makes it possible to call a CES survey more than once, but doing so will recreate the data object. When `get_ces()` is called, before downloading the requested survey file, `get_ces()` checks if the file already exists in the `inst/extdata` directory. While `get_ces()` is designed to remove the downloaded file, checking if the file already exists alerts the function if something is wrong. By checking if the file exists and not if the data object exists, the `get_ces()` function is able to load the requested dataset more than once, thereby allowing an unmanipulated version of a dataset to be loaded if so required. For example, if a loaded data object becomes corrupted or an unmanipulated version is needed, then using `get_ces()` to call the same CES survey will provide a clean copy.

CES survey dataset files uses either a `.dta()` or `.sav()` file extension, meaning the datasets are loaded into R and RStudio as the type labelled. The `get_ces()` function does not convert the values of the loaded tables to a factor type so that personal work-flow practices are not interfered with. It is suggested that to convert the dataset values to a factor type that the `to_factor()` function from the `labelled` package (Larmarange, 2020) be used (see *2.1.3 `to_factor()` function* for an example as to using the `to_factor()` function and `labelled` package).

2.1.3 `to_factor()` function

```
# install cesR package from GitHub
devtools::intall_github("hodgettsp/cesR")

# install labelled package from CRAN
install.packages("labelled")

# load cesR package into RStudio
```

```
library(cesR)

# load the labelled package
library(labelled)

# request 2019 CES online survey
get_ces("ces2019_web")

# convert dataframe values to factor type
# check column heads for dataframe
ces2019_web <- to_factor(ces2019_web)
head(ces2019_web)
```

```
>head(ces2019_web)
# A tibble: 6 x 620
  cps19_StartDate     cps19_EndDate       cps19_ResponseId cps19_consent cps19_citizensh~
  <dttm>              <dttm>              <chr>            <fct>         <fct>
1 2019-09-13 08:09:44 2019-09-13 08:36:19 R_1OpYXEFGzHRUp~ I consent to~ Canadian citizen
2 2019-09-13 08:39:09 2019-09-13 08:57:06 R_2qdrL3J618rxY~ I consent to~ Canadian citizen
3 2019-09-13 10:01:19 2019-09-13 10:27:29 R_USWDAPcQEQiMm~ I consent to~ Canadian citizen
4 2019-09-13 10:05:37 2019-09-13 10:50:53 R_3IQaeDXy0tBzE~ I consent to~ Canadian citizen
5 2019-09-13 10:05:52 2019-09-13 10:32:53 R_27WeMQ1asip2c~ I consent to~ Canadian citizen
6 2019-09-13 10:10:20 2019-09-13 10:29:45 R_3LiGZcCWJEcWV~ I consent to~ Canadian citizen
```

2.2 get_cescodes()

The `get_cescodes()` function provides a user a means of looking up the argument calls that are used to access each CES survey dataset. The `get_cescodes()` function does not take any arguments. Instead when the function is called it prints to the console a dataframe that contains the survey codes and their associated argument calls. See *2.2.1 `get_cescodes()` function* for an example of the print output and the *3. Vignette* section for an example of the `get_cescodes()` used as part of the full `cesR` package.

2.2.1 get_cescodes() function

```
# install cesR package from GitHub
devtools::install_github("hodgettsp/cesR")

# load cesR package into RStudio
library(cesR)

# get CES survey argument calls
get_cescodes()
```

```
>get_cescodes()
   index ces_survey_code get_ces_call_char
1      1     ces2019_web     "ces2019_web"
2      2   ces2019_phone   "ces2019_phone"
3      3     ces2015_web     "ces2015_web"
4      4   ces2015_phone   "ces2015_phone"
5      5   ces2015_combo   "ces2015_combo"
6      6         ces2011         "ces2011"
7      7         ces2008         "ces2008"
```

```
8      8         ces2004         "ces2004"
9      9         ces0411         "ces0411"
10    10         ces0406         "ces0406"
11    11         ces2000         "ces2000"
12    12         ces1997         "ces1997"
13    13         ces1993         "ces1993"
14    14         ces1988         "ces1988"
15    15         ces1984         "ces1984"
16    16         ces1974         "ces1974"
17    17         ces7480         "ces7480"
18    18      ces72_jnjl      "ces72_jnjl"
19    19       ces72_sep       "ces72_sep"
20    20       ces72_nov       "ces72_nov"
21    21         ces1968         "ces1968"
22    22         ces1965         "ces1965"
```

The `get_cescodes()` function works by constructing two vectors, one vector contains the CES survey codes and the other containing the associated survey argument calls. The function then creates dataframes of two vectors using the `data.frame()` function from the `base` package and adds an index number column using the `seq()` function from the `base` package (R Core Team, 2020) by which to merge the dataframes. Using the `merge()` function from the `base` package (R Core Team, 2020), the dataframes are then merged by the index number into a new dataframe. Using the index number ensures the dataframes remain ordered and merge correctly across rows. Column names in the new dataframe are then renamed using the `rename()` function from the `dplyr` package (Wickham et al., 2020a) and the vector objects are removed from the environment. The `get_cescodes()` function does not create any variable that is available in the global environment.

2.3 get_question()

The `get_question()` provides users with the ability to look up a survey question associated with a given column name. The function takes two arguments in the form of character strings, those being the name of a data object and the name of a column in the given data object. The function works such that it checks whether the given data object exists using the `exists()` function from the `base` package (R Core Team, 2020). If the object does not exist, the function will print out a warning in the console stating `Warning: Data object does not exist` (see *2.3.2 `get_question()` data object error* for an example of this error). If the object does exist, `get_question()` will check if the given column name exists in the given data object. This is done using a combination of the `hasName()` function from the `utils` package and the `get()` function from the `base` package (R Core Team, 2020). The `hasName()` function checks if the given column name is in the given data object. Because the arguments are given as character strings the `get()` function is used to return the actual data object instead of the provided character string. Otherwise, the `hasName()` function would only check if the given column name argument occurred in the given character string argument and not the actual data object. If the column does not exist in the data object a warning is printed in the console stating `Warning: Variable is not in dataset`. See *2.3.3 `get_question()` variable error* for an example of this error message. If the given column exists in the given data object, `get_question()` will print the variable label of the given column to the console using a combination of the `var_label()` function from the `labelled` package (Larmarange, 2020) and the `get()` function from the `base` package (R Core Team, 2020). See *2.3.1 `get_question()` basics* for an example of the general use of the `get_question()` function and the *3. Vignette* section for its use as part of the whole `cesR` package.

2.3.1 get_question() basics

```
# install cesR package from GitHub
devtools::install_github("hodgettsp/cesR")
```

```
# load package into library
library(cesR)

# load 2019 phone dataset
get_ces("ces2019_phone")

# get question for column q11
get_question("ces2019_phone", "q11")

Which party will you likely to vote for
```

2.3.2 `get_question()` data object error

```
# install cesR package from GitHub
devtools::install_github("hodgettsp/cesR")

# load package into library
library(cesR)

# load 2019 phone dataset
get_ces("ces2019_phone")

# get question for column q11
get_question("2019_phone", "q11")

Warning: Data object does not exist
```

2.3.3 `get_question()` variable error

```
# install cesR package from GitHub
devtools::install_github("hodgettsp/cesR")

# load package into library
library(cesR)

# load 2019 phone dataset
get_ces("ces2019_phone")

# get question for column q11
get_question("ces2019_phone", "11")

Warning: Variable is not in dataset
```

In addition to being usable with the data objects created from the `get_ces()` function, the `get_question()` function is structured in such a way that it can also be used to return the column label for the `decon` dataset or any dataset of the labelled type. *2.3.4 `get_question()` `decon`* presents an example of the `get_question()` function being used with the `decon` dataset.

2.3.4 get_question() decon

```
# install cesR package from GitHub
devtools::install_github("hodgettsp/cesR")

# load package into library
library(cesR)

# load decon dataset
get_decon()
head(decon)

# get question for education column
get_question("decon", "education")
```

```
What is the highest level of education that you have completed?
```

2.4 get_preview()

To make the data exploratory phase easier it may be prudent to have access to a truncated version of a dataset. Additionally, a truncated dataset provides a resource that can be used by educators in the teaching of exploratory data analysis. The `get_preview()` function provides such truncated versions of the CES datasets. The function takes two arguments, a character string to call a survey of the same style as used for the `get_ces()` function and a numerical value that sets the number of rows returned (see 2.4.1 `get_preview()` basics). If no value is provided for the number of rows, a default of six is returned (see 2.4.2 `get_preview()` no value provided).

2.4.1 get_preview() basics

```
#install cesR package
devtools::install_github("hodgettsp/cesR")

# load package into library
library(cesR)

# get preview for 2019 CES online survey for 10 rows
get_preview("ces2019_web", 10)
```

2.4.2 get_preview() no value provided

```
# install cesR package
devtools::install_github("hodgettsp/cesR")

# load package into library
library(cesR)

# get preview for 2019 CES online survey
get_preview("ces2019_web")
```

cps19_StartDate	cps19_EndDate	cps19_ResponseId	cps19_consent	cps19_citizensh~
2019-09-13 08:09:44	2019-09-13 08:36:19	R_1OpYXEFGzHRUp~	I consent to~	Canadian citizen
2019-09-13 08:39:09	2019-09-13 08:57:06	R_2qdrL3J618rxY~	I consent to~	Canadian citizen
2019-09-13 10:01:19	2019-09-13 10:27:29	R_USWDAPcQEQiMm~	I consent to~	Canadian citizen
2019-09-13 10:05:37	2019-09-13 10:50:53	R_3IQaeDXy0tBzE~	I consent to~	Canadian citizen
2019-09-13 10:05:52	2019-09-13 10:32:53	R_27WeMQ1asip2c~	I consent to~	Canadian citizen
2019-09-13 10:10:20	2019-09-13 10:29:45	R_3LiGZcCWJEcWV~	I consent to~	Canadian citizen

In the initial argument classifications for the `get_preview()` function the variable `x` is set to equal six. This sets a default value for the number of rows to be returned if this value is left empty while allowing the value to be overwritten if provided a value. Like the `get_ces()` function, the `get_preview()` function checks if a file exists in the cesR directory before downloading the requested survey data file associated with the given character string argument. If the file does not already exist, the survey data file is downloaded using the `download.file()` function from the `base` package to a temporary directory set using the `tempfile()` function also from the `base` package (R Core Team, 2020). The downloaded file is then unzipped into the `cesR` package directory using the `unzip()` function from the `utils` package (R Core Team, 2020). The extracted survey data file is read into R within the `get_preview()` function using either the `read_dta()` or `read_sav()` functions from the `haven` package depending upon the file extension (Wickham and Miller, 2020). The `get_preview()` function then creates a truncated version of the requested dataset by converting the dataset values to a factor type using the `to_factor()` function from the `labelled` package (Larmarange, 2020) and truncating the dataset using the `head()` function from the `utils` package (R Core Team, 2020) by the numeric value provided by the argument `x`. This dataset is assigned as a variable with the suffix `_preview` to the global environment using the `assign()` function from the `base` package (R Core Team, 2020). The function then removes the downloaded files using the `unlink()` function from the `base` package and removes the local variable holding the full dataset using the `rm()` function also from the `base` package (R Core Team, 2020). See 3. Vignette for a use of the `get_preview()` function in conjunction with the other `cesR` functions.

2.5 get_decon()

When called, creates a subset of the 2019 CES online survey under the name `decon` (demographics and economics) that provides a tool for educators in the teaching of the analysis of large survey datasets. The `get_decon()` function takes no arguments. The function first checks the global environment if an object named `decon` exists using the `exists()` function from the `base` package (R Core Team, 2020). This prevents the `decon` dataset from being recreated if the object exists. If the `get_decon()` function is run when an object with the name `decon` already exists a warning will print in the console stating `Error in get_decon() : Warning: File already exists.` See *2.5.1 `get_decon()` error* for an example of the error message. If a situation arises in which the `decon` dataset needs to be recreated, then the best course of action is to use the `rm()` function from the `base` package (R Core Team, 2020) to remove the `decon` object and then run the `get_decon()` function again (see *2.5.2 `get_decon()` reload*).

2.5.1 `get_decon()` error

```
# install cesR package
devtools::install_github("hodgettsp/cesR")
```

```
# load cesR package
library(cesR)

# load decon dataset
get_decon()
head(decon)

# reload decon
get_decon()

Error in get_decon() : Warning: File already exists.
```

2.5.2 `get_decon()` reload

```
# install cesR package
devtools::install_github("hodgettsp/cesR")

# load cesR package
library(cesR)

# load decon dataset
get_decon()
head(decon)

# remove decon object
rm(decon)

# relaoad decon
get_decon()
```

After checking if an object with the name `decon` exists, the `get_decon()` function performs similarly to the `get_ces()` function. It assigns a temporary file extension with a .zip type using the `tempfile()` function from the `base` package, and calls on the URL for the 2019 CES online survey and temporarily downloads the associated file using the `download.file()` function from the `base` package (R Core Team, 2020). After unpacking the compressed folder with the `unzip()` function from the `utils` package (R Core Team, 2020), the file is read into R using the `read_dta()` function from the `haven` package (Wickham and Miller, 2020). Using the `select()` function from the `dplyr` package, 20 variables are selected from the main CES 2019 online survey and renamed using the `rename` function from the `dplyr` package (Wickham et al., 2020a). The dataframe values are then converted to a factor type using the `to_factor()` function from the `labelled` package (Larmarange, 2020). A new variable, consisting of participants' responses to their self-perceived position on the political spectrum, is then created from two variables using the `unite()` function from the `tidyr` package (Wickham and Henry, 2020) and all empty cells are replaced with `NA` values using the `mutate()` function from the `dplyr` package (Wickham et al., 2020a). The dataset is made available in the global environment by using the `assign()` function from the `base` package to create a new data object under the name `decon`. Lastly, the downloaded files are removed from the computer using the `unlink()` function from the `base` package (R Core Team, 2020) and a citation for the 2019 CES online survey and a URL to the storage location is printed to the console. See *2.5.3 `get_decon()` basics* for an example of a general use of the `get_decon()` function.

2.5.3 `get_decon()` basics

```
# install cesR package
```

```
devtools::install_github("hodgettsp/cesR")

# load cesR package into RStudio
library(cesR)

# call decon dataset
get_decon()
head(decon)

TO CITE THIS SURVEY FILE: Stephenson, Laura B; Harell, Allison; Rubenson, Daniel;
Loewen, Peter John, 2020, '2019 Canadian Election Study - Online Survey',
https://doi.org/10.7910/DVN/DUS88V, Harvard Dataverse, V1

LINK: https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/DUS88V

# A tibble: 6 x 22
  ces_code citizenship yob   gender province_territ~ education lr    lr_bef lr_aft religion
  <chr>    <fct>       <fct> <fct>  <fct>            <fct>     <chr> <chr>  <chr>  <fct>
1 ces2019~ Canadian c~ 1989  A wom~ Quebec           Master's~ 2     NA     2      Don't k~
2 ces2019~ Canadian c~ 1998  A wom~ Quebec           Master's~ 2     NA     2      Catholi~
3 ces2019~ Canadian c~ 2000  A wom~ Ontario          Some uni~ 4     4      NA     Catholi~
4 ces2019~ Canadian c~ 1998  A man  Ontario          Some uni~ 7     7      NA     Jewish/~
5 ces2019~ Canadian c~ 2000  A wom~ Ontario          Complete~ 4     4      NA     Catholi~
6 ces2019~ Canadian c~ 1999  A wom~ Ontario          Some uni~ 4     4      NA     Catholi~
```

3 Vignette

3.1 Creating a subset of the CES 2019 phone survey dataset

While the `get_decon()` provides a subset of the CES 2019 online survey dataset, the general use of the `cesR` package is to access CES data and the subsetting of any of the CES survey datasets.

The following presents a vignette of installing `cesR` from its GitHub repository as well as calling and producing a subset of the 2019 CES phone survey dataset. This vignette uses functions from the `cesR`, `devtools` (Wickham et al., 2020b), `labelled` (Larmarange, 2020), `dplyr` (Wickham et al., 2020a), and `utils` (R Core Team, 2020) packages.

To begin, install and load the `cesR` package (and all other necessary packages) into RStudio. Currently, this is currently only available through the use of the `install_github` function from the `devtools` package (Wickham et al., 2020b). During installation, RStudio may request to update other packages. It is best to press enter with an empty line (see *3.1.1 Install `cesR`*).

3.1.1 Install `cesR`

```
# uncomment any package that needs to be installed
# install.packages("devtools")
# install.packages("labelled")
# install.packages("dplyr")
# install.packages("tidyr")

# install cesR package from GitHub
devtools::install_github("hodgettsp/cesR")
```

```
library(cesR)
library(labelled)
library(dplyr)
```

```
> devtools::install_github("hodgettsp/cesR")
Downloading GitHub repo hodgettsp/cesR@HEAD
```

Upon installation and loading of the `cesR` package, we can use the `get_cescodes()` function to look up the code for the desired CES survey. In this case that is the 2019 CES phone survey (see *3.1.2 Lookup CES codes*).

3.1.2 Lookup CES codes

```
# look up survey codes
get_cescodes()
```

```
> get_cescodes()
   index ces_survey_code get_ces_call_char
1      1     ces2019_web     "ces2019_web"
2      2   ces2019_phone   "ces2019_phone"
3      3     ces2015_web     "ces2015_web"
4      4   ces2015_phone   "ces2015_phone"
5      5   ces2015_combo   "ces2015_combo"
6      6         ces2011         "ces2011"
7      7         ces2008         "ces2008"
8      8         ces2004         "ces2004"
9      9         ces0411         "ces0411"
10    10         ces0406         "ces0406"
11    11         ces2000         "ces2000"
12    12         ces1997         "ces1997"
13    13         ces1993         "ces1993"
14    14         ces1988         "ces1988"
15    15         ces1984         "ces1984"
16    16         ces1974         "ces1974"
17    17         ces7480         "ces7480"
18    18      ces72_jnjl      "ces72_jnjl"
19    19       ces72_sep       "ces72_sep"
20    20       ces72_nov       "ces72_nov"
21    21         ces1968         "ces1968"
22    22         ces1965         "ces1965"
```

In the printed data frame we can see that the argument code for the 2019 CES phone survey is "ces2019_phone" (see *3.1.2 Lookup CES codes*). This code can be used with the `get_preview()` function to look up a truncated preview of the 2019 CES phone survey dataset (see *3.1.3 Preview dataset*).

3.1.3 Preview dataset

```
# get preview of first 8 rows of 2019 CES phone survey dataset
get_preview("ces2019_phone", 8)
```

sample_id	survey_end_CES	survey_end_mont~	survey_end_day_~	num_attempts_CES
18	2019-09-23 15~	9	23	5
32	2019-09-12 18~	9	12	1
39	2019-09-10 18~	9	10	1
59	2019-10-10 15~	10	10	6
61	2019-09-12 16~	9	12	1
69	2019-09-17 17~	9	17	1
157	2019-09-12 16~	9	12	1
158	2019-09-14 10~	9	14	4

Using the same survey code argument with the `get_ces()` function we can retrieve the complete 2019 CES phone survey (see *3.1.4 `get_ces()`*). Additionally, we can see when `get_ces()` is called, the survey citation and link to repository prints to the console.

3.1.4 `get_ces()`

```
# call 2019 CES phone survey
get_ces("ces2019_phone")
```

```
TO CITE THIS SURVEY FILE: Stephenson, Laura B; Harell, Allison; Rubenson, Daniel;
Loewen, Peter John, 2020, '2019 Canadian Election Study - Phone Survey',
https://doi.org/10.7910/DVN/8RHLG1, Harvard Dataverse, V1,
UNF:6:eyR28qaoYlHj9qwPWZmmVQ== [fileUNF]

LINK: https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/8RHLG1
```

To check that the dataset has loaded into RStudio correctly, it is best to check the dataset with the `head()` function from the `utils` package (R Core Team, 2020). Unfortunately, the `head()` function does not work with labelled data, so it is best to convert the values to factors using the `to_factor()` function from the `labelled` package (Larmarange, 2020) (see *3.1.5 Check 2019 CES phone data*).

3.1.5 Check 2019 CES phone data

```
# convert values to factor type
ces2019_phone <- to_factor(ces2019_phone)
head(ces2019_phone)

# alternatively the factored table can be assigned to its own data object
# to leave the original untouched.
# ces2019_phone_factor <- labelled::to_factor(ces2019_phone)
# head(ces2019_phone_factor)
```

```
>head(ces2019_phone)
# A tibble: 6 x 278
  sample_id survey_end_CES survey_end_mont~ survey_end_day_~ num_attempts_CES
      <dbl> <chr>                     <dbl>            <dbl>            <dbl>
1        18 2019-09-23 15~                9               23                5
2        32 2019-09-12 18~                9               12                1
3        39 2019-09-10 18~                9               10                1
4        59 2019-10-10 15~               10               10                6
```

```
5        61 2019-09-12 16~               9              12               1
6        69 2019-09-17 17~               9              17               1
```

With the dataset loaded, the `select()` function from the `dplyr` package (Wickham et al., 2020a) is an efficient way of selecting the required variable columns. In this case, the language, phone type, weight, citizenship, year of birth, interviewer gender, identified gender, province, satisfaction with Canadian democracy, and most important election issue are selected using the column index (see *3.1.4 Select columns*). Additionally, the columns are not all named in an understandable way and so will be renamed using the `rename()` function from the `dplyr` package (Wickham et al., 2020a).

3.1.6 Select columns

```
ces2019_phone_subset <- ces2019_phone %>%
  select(7, 8, 9, 20, 25:30)

head(ces2019_phone_subset)
```

```
>head(ces2019_phone_subset)
# A tibble: 6 x 10
  interviewer_gender~ language_CES phonetype_CES weight_CES q1     q2    q3      q4
  <chr>               <fct>        <fct>              <dbl> <fct>  <fct> <fct>   <fct>
1 Female              (2) French   (2) Wireless       0.902 (1) Y~ 1963  (1) Ma~ (5) Que~
2 Male                (1) English  (2) Wireless       0.902 (1) Y~ 1973  (1) Ma~ (5) Que~
3 Female              (2) French   (2) Wireless       0.902 (1) Y~ 1994  (1) Ma~ (5) Que~
4 Female              (2) French   (2) Wireless       1.23  (1) Y~ 2000  (1) Ma~ (5) Que~
5 Male                (2) French   (2) Wireless       0.902 (1) Y~ 1984  (1) Ma~ (5) Que~
6 Female              (2) French   (2) Wireless       0.902 (1) Y~ 1939  (1) Ma~ (5) Que~
```

We can see that some of the column names of the subset are not clearly named. The remedy this we can use the `rename` function from the `dplyr` package (Wickham et al., 2020a). When using `rename()`, the order goes from new name to old name (see *3.1.6 Rename columns*). To gain an idea of what to rename a column we can use the `get_question()` function to find the associated survey question (see *3.1.5 `get_question()`*).

3.1.7 `get_question()`

```
get_question("ces2019_phone_subset", "q1")
get_question("ces2019_phone_subset", "q2")
get_question("ces2019_phone_subset", "q3")
get_question("ces2019_phone_subset", "q4")
get_question("ces2019_phone_subset", "q6")
get_question("ces2019_phone_subset", "q7")
```

```
Are you a Canadian Citizen?
In what year were you born?
Gender
In which province or territory are you currently living?
On the whole, are you very satisfied, fairly satisfied, not very satisfied
Most important issue in this FEDERAL election
```

Using the labels returned by `get_question()` we can better name the subset columns (see *3.1.6 Rename colums*).

3.1.8 Rename columns

```
ces2019_phone_subset <- ces_phone_subset %>%
  rename(interviewer_gender = interviewer_gender_CES,
         language = language_CES,
         phonetype = phonetype_CES,
         weight = weight_CES,
         citizenship = q1,
         yob = q2,
         respondent_gender = q3,
         province_territory = q4,
         dmcrcy_satisfaction = q6,
         imprtnt_issue = q7)
```

This will have now created a subset of the CES 2019 phone survey with renamed variables and factor values.

As a final note, it is recommended to download the codebook for a requested survey dataset. These are available from the link printed on a survey call as well as on the package GitHub repository README.

4 Next steps and cautions

The `cesR` package is dependent upon the functions from the `haven` (Wickham and Miller, 2020) and `labelled` (Larmarange, 2020) packages to be able to read the CES survey datasets. Thus, changes to the `haven` or `labelled` package may affect the functionality of the `cesR` package. As such, a future consideration would be to move away from this dependency to make the cesR package more independent and self-contained.

Regarding the CES survey datasets, currently the datasets are downloaded from an associated GitHub repository. This means that a dataset will be downloaded as it currently exists on that repository. While these datasets are relatively stable (meaning the data is unchanging), updates are performed on the datasets to fix incorrect values or mislabelled variables. While the package will be maintained to ensure the most up-to-date survey datasets are included, it may be the case that an update is not immediately performed on a dataset file. A step to eliminate this possible issue is to link the `get_ces()` call directly to the download URL of the survey as opposed to the current call to the GitHub repository.

The `cesr` package was written to not include datasets so as to minimize its size and speed up install times. However, as the package does need to download files, the speed at which the package functions is dependent upon the speed of a user's internet. This may slow down performance in some cases of poor internet connection.

Lastly, the CES surveys are generally divided into themed sections. A future step will be to build more subsets of the surveys, including the division of survey sections into their own subsets, so that topics may be more easily analysed.

References

Belanger, E. and Nadeau, R. (2005). Political trust and the vote in multiparty elections: The Canadian case. *European Journal of Political Research*, 44(1):121–146.

Bittner, A. (2018). Leaders always mattered: The persistence of personality in canadian elections. *Electoral Studies*, 54:297–302.

Canadian Election Study (2019). *Welcome to the 2019 Canadian Election Study*. http://www.ces-eec.ca/.

Friendly, M., Dalzell, C., Monkman, M., and Murphy, D. (2020). *Lahman: Sean 'Lahman' Baseball Database*. R package version 8.0-0.

Gelfand, S. (2020). *opendatatoronto: Access the City of Toronto Open Data Portal*. R package version 0.1.3.

Gidengil, E., Blais, A., Nevitte, N., and Nadeau, R. (2001). The correlates and consequences of anti-partyism in the 1997 canadian election. *Party Politics*, 7(4):491–513.

Larmarange, J. (2020). *labelled: Manipulating Labelled Data*. R package version 2.5.0.

Leeper, T. J. (2017). *dataverse: R Client for Dataverse 4*. R package version 0.2.0.

ODESI (2020). *ODESI*. https://search1.odesi.ca/#/.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

RStudio Team (2020). *RStudio: Integrated Development Environment for R*. RStudio, PBC, Boston, MA.

UBC (2015). *Canadian Election Study / Étude électorale canadienne*. https://ces-eec.arts.ubc.ca/english-section/surveys/.

von Bergmann, J., Shkolnik, D., and Jacobs, A. (2020). *cancensus: R package to access, retrieve, and work with Canadian Census data and geography*. R package version 0.3.2.

Wickham', H. (2014). *nasaweather: Collection of datasets from the ASA 2006 data expo*. R package version 0.1.

Wickham, H. (2020). *fueleconomy: EPA Fuel Economy Data*. R package version 1.0.0.

Wickham, H., François, R., Henry, L., and Müller, K. (2020a). *dplyr: A Grammar of Data Manipulation*. R package version 1.0.0.

Wickham, H. and Henry, L. (2020). *tidyr: Tidy Messy Data*. R package version 1.1.0.

Wickham, H., Hester, J., and Chang, W. (2020b). *devtools: Tools to Make Developing R Packages Easier*. R package version 2.3.1.

Wickham, H. and Miller, E. (2020). *haven: Import and Export 'SPSS', 'Stata' and 'SAS' Files*. R package version 2.3.1.

Wilkins-Laflamme, S. (2018). Islamophobia in canada: Measuring the realities of negative attitudes toward muslims and religious discrimination. *Canadian Review of Sociology/Revue canadienne de sociologie*, 55(1):86–110.