Data Frames

An Aside About Data Sets

- Data sets are typically messy.
  - NA's might be `-9` (numeric), `“-9. Refused”`, `“-8. Don't know”`, “`-2. Missing, other not codeable to 1-5`”, or some combination of these.
  - `“Strong Democrat”` is a `1`.
  - `“Other”` is `"5. Other party {SPECIFY}"`
  - Data sets are not “tidy” (rows are observations; columns are variables).
  - Factors are strangely ordered or are character vectors.
  - Data are stored in different data files.
  - Lots of unnecessary columns or rows.
  - Uniformatively named columns, such as `var1003b`.
- The data I give you are clean and tidy.
- The skill of taking messy data files and cleaning and tidying is called “data wrangling.” We talk very little about data wrangling.

Terminology

- **data set**: a collection of information stored somehow, somewhere.
- **data file**: a specific file containing a data set.
- **file type**: the specific format in which the data are stored (e.g., .xlsx, .dta, .rds, .csv)
- **data frame**: a type of object in R; think of as a "box of vectors."
    - other objects include scalars, vectors, and functions
    - all vectors in a box have the same length (number of elements)
    - most functions for modeling and graphing are designed to work with data frames via a `data =` argument, not vectors

Data reading functions
create data frames from data files.

`read_csv()`, `read_dta()`, `read_excel()`, `read_rds()`

thinking about data frames

```
x <- c(6, 4, 5, 6, 2, 3)  # create a numeric vector
```

x

```
my_logic <- c(TRUE, FALSE, FALSE)  # create logical vector
```

x

my_logic

x

my_logic

```
ch.vec <- c(“word1”, “word2”)  # create character vector
```

ch.vec

x

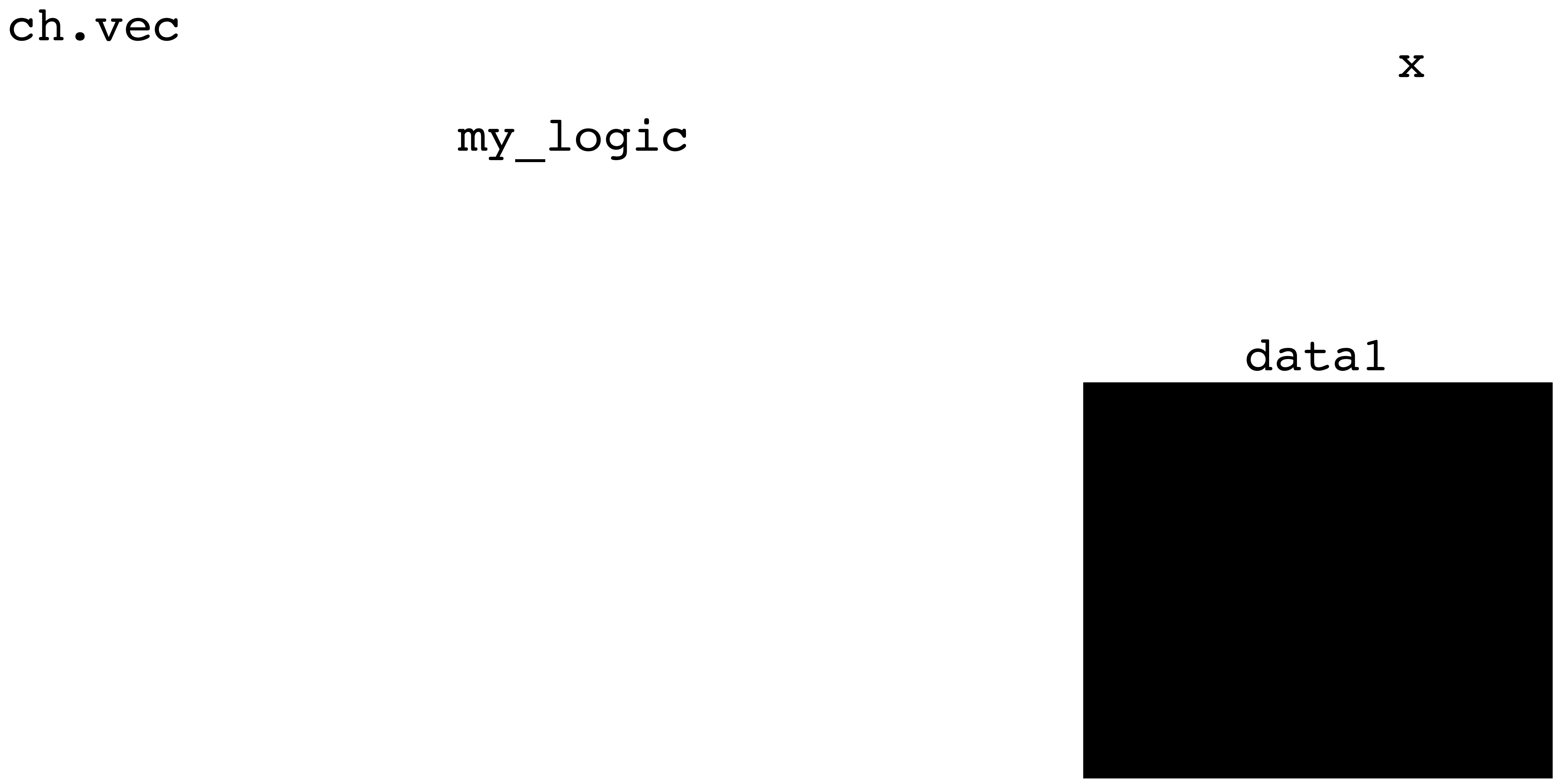
my_logic

ch.vec

x

my_logic

```
data1 <- read_csv(“nominate.csv”)  # read data set
```

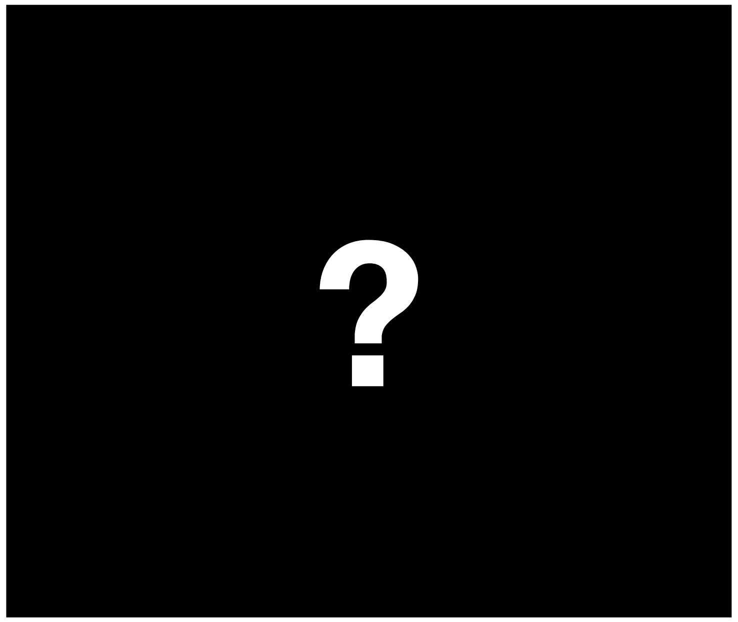

ch.vec
x
my_logic
data1

ch.vec

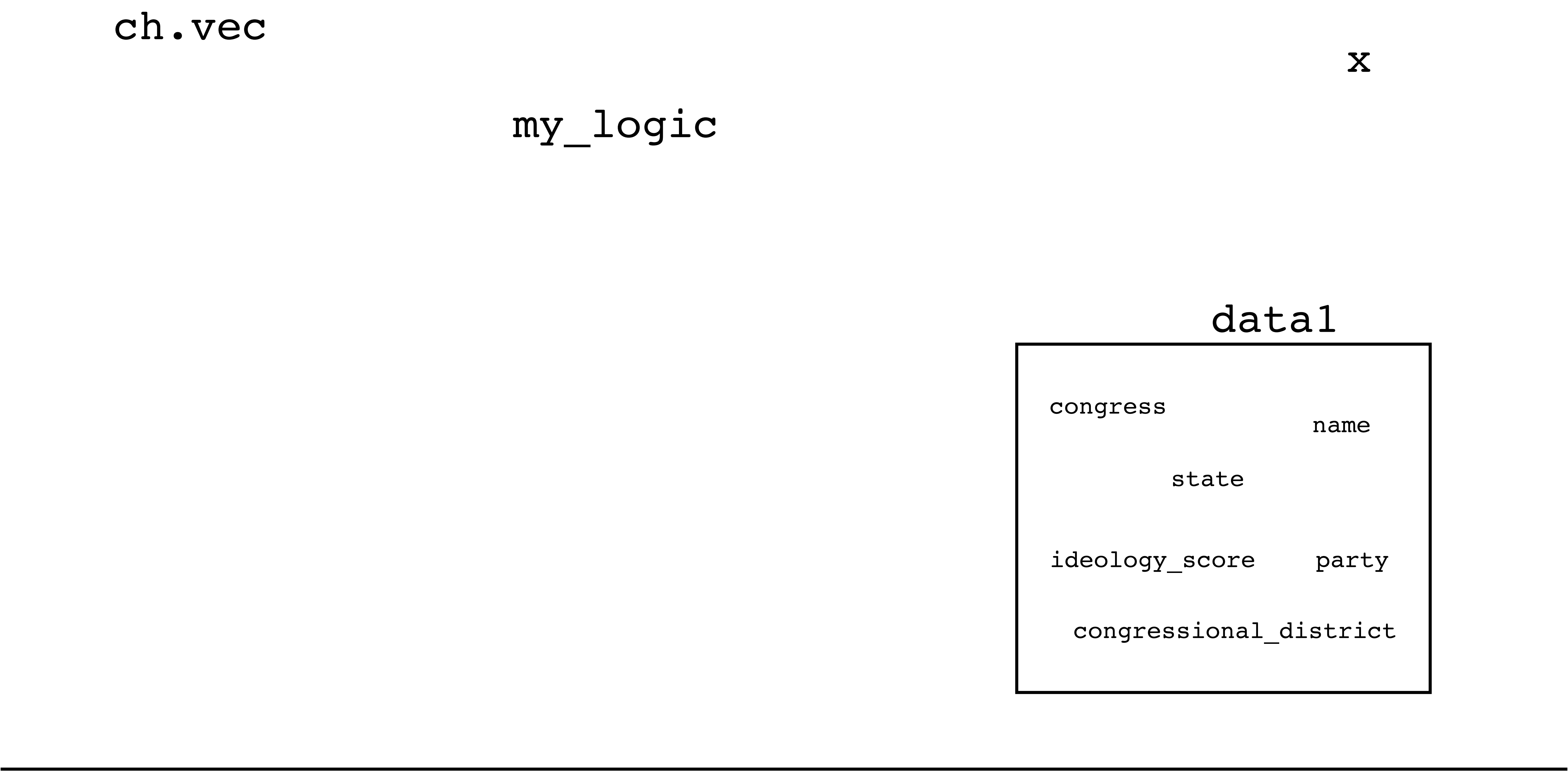
x

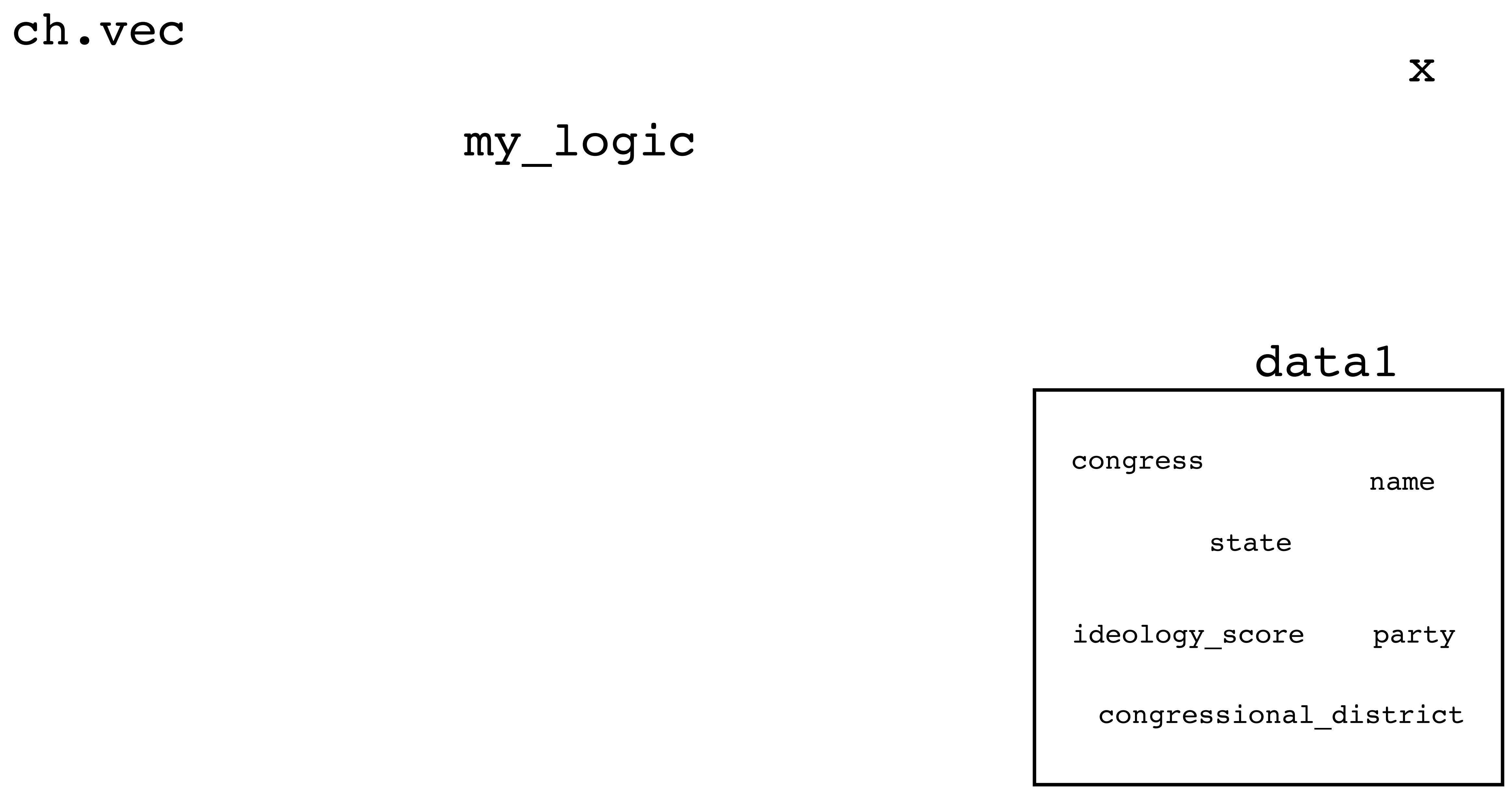
my_logic

data1

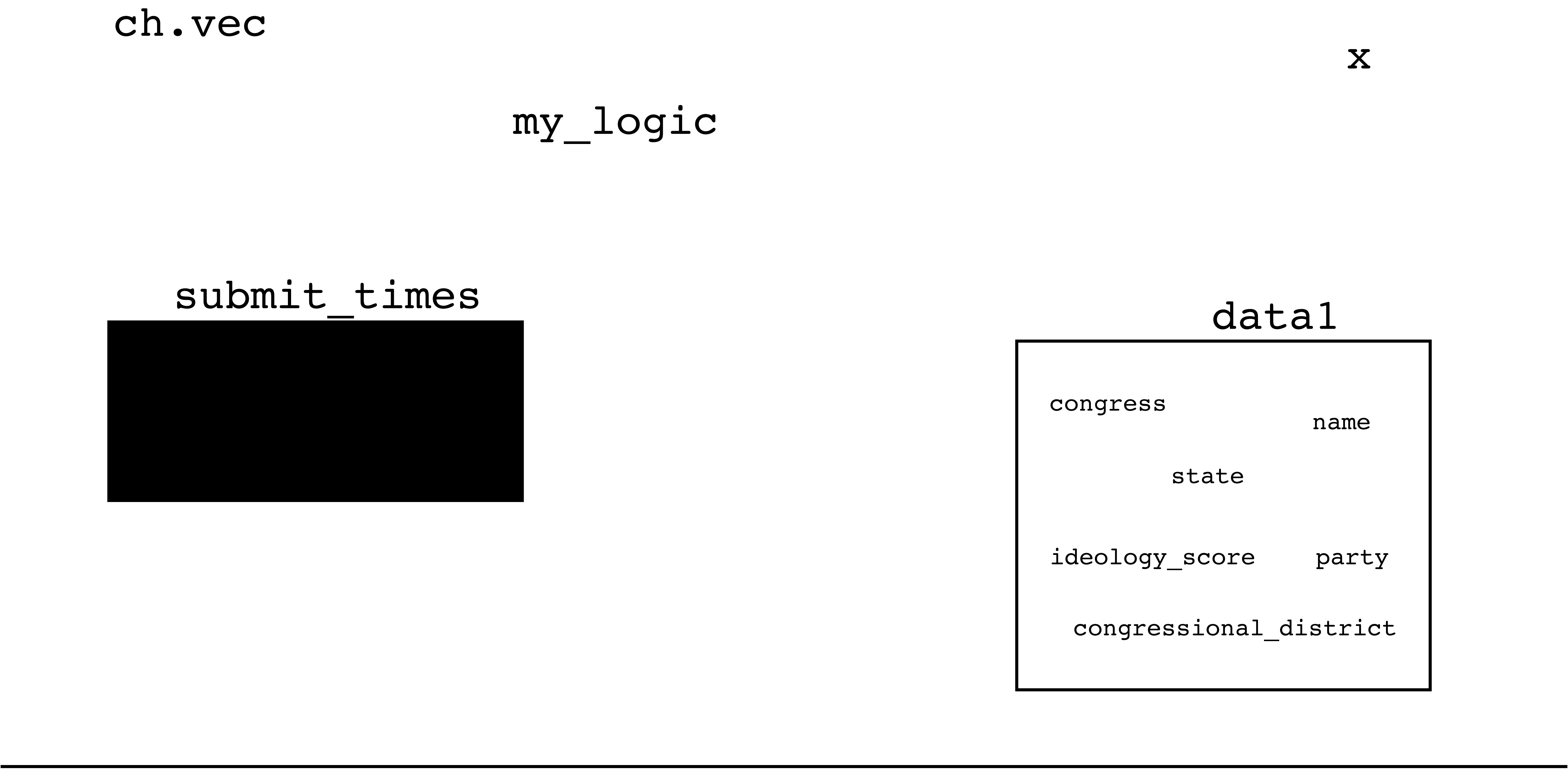


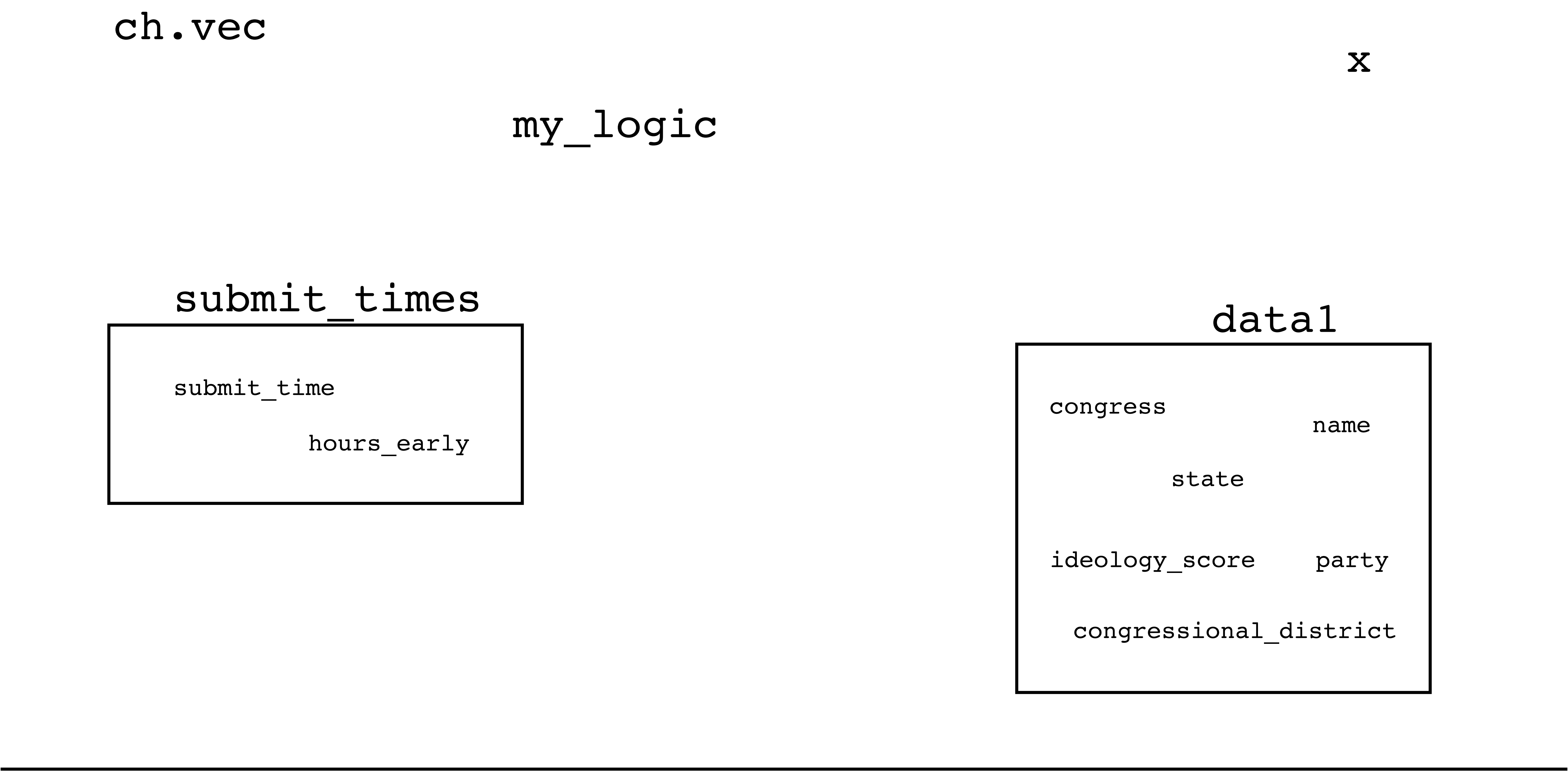
```
> glimpse(data1)
Observations: 6,159
Variables: 6
$ congress               <int> 100, 100, 100, 100, 100, 100, 1...
$ state                  <fctr> ALABAMA, ALABAMA, ALABAMA, ALA...
$ congressional_district <int> 1, 2, 3, 4, 5, 6, 7, 1, 1, 2, 3...
$ party                  <fctr> Republican, Republican, Democr...
$ name                   <fctr> CALLAHAN, DICKINSON, NICHOLS  ...
$ ideology_score         <dbl> 0.358, 0.349, -0.039, -0.203, -...
```

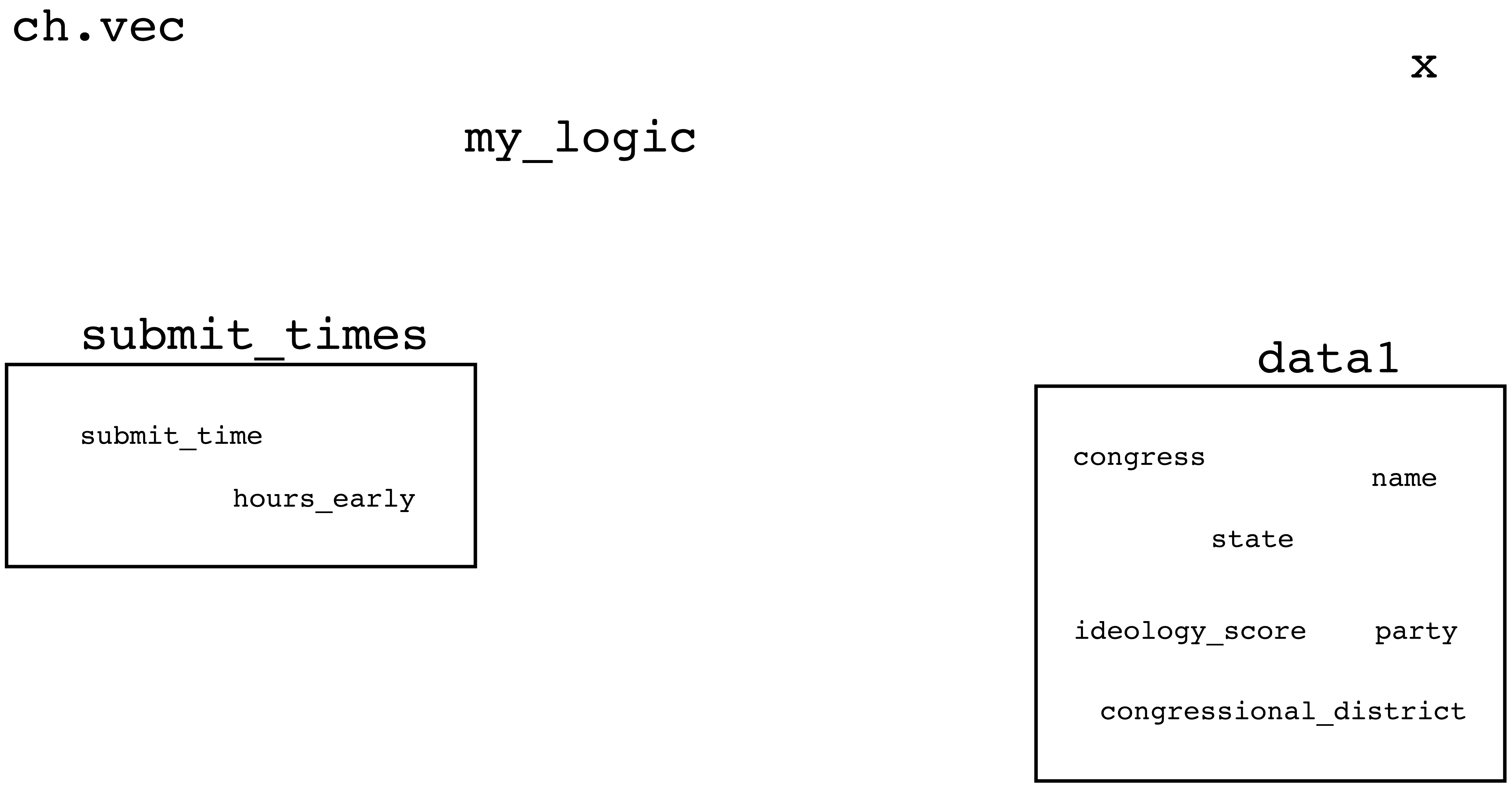

ch.vec
x
my_logic
data1
congress
name
state
ideology_score
party
congressional_district



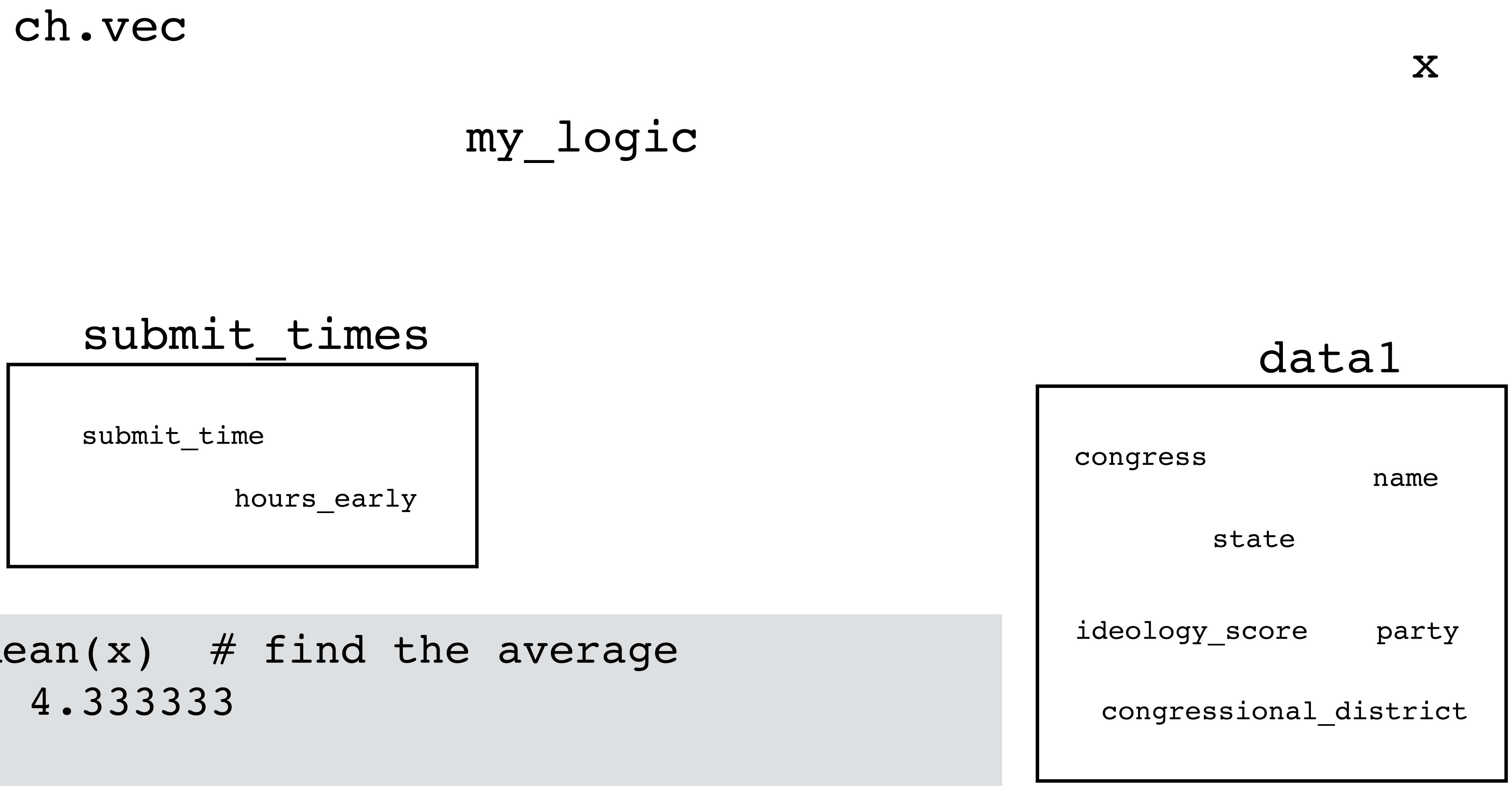
```
submit_times <- read_rds("submit_times.rds")  # read data
```


ch.vec
x
my_logic
submit_times
data1
congress
name
state
ideology_score
party
congressional_district

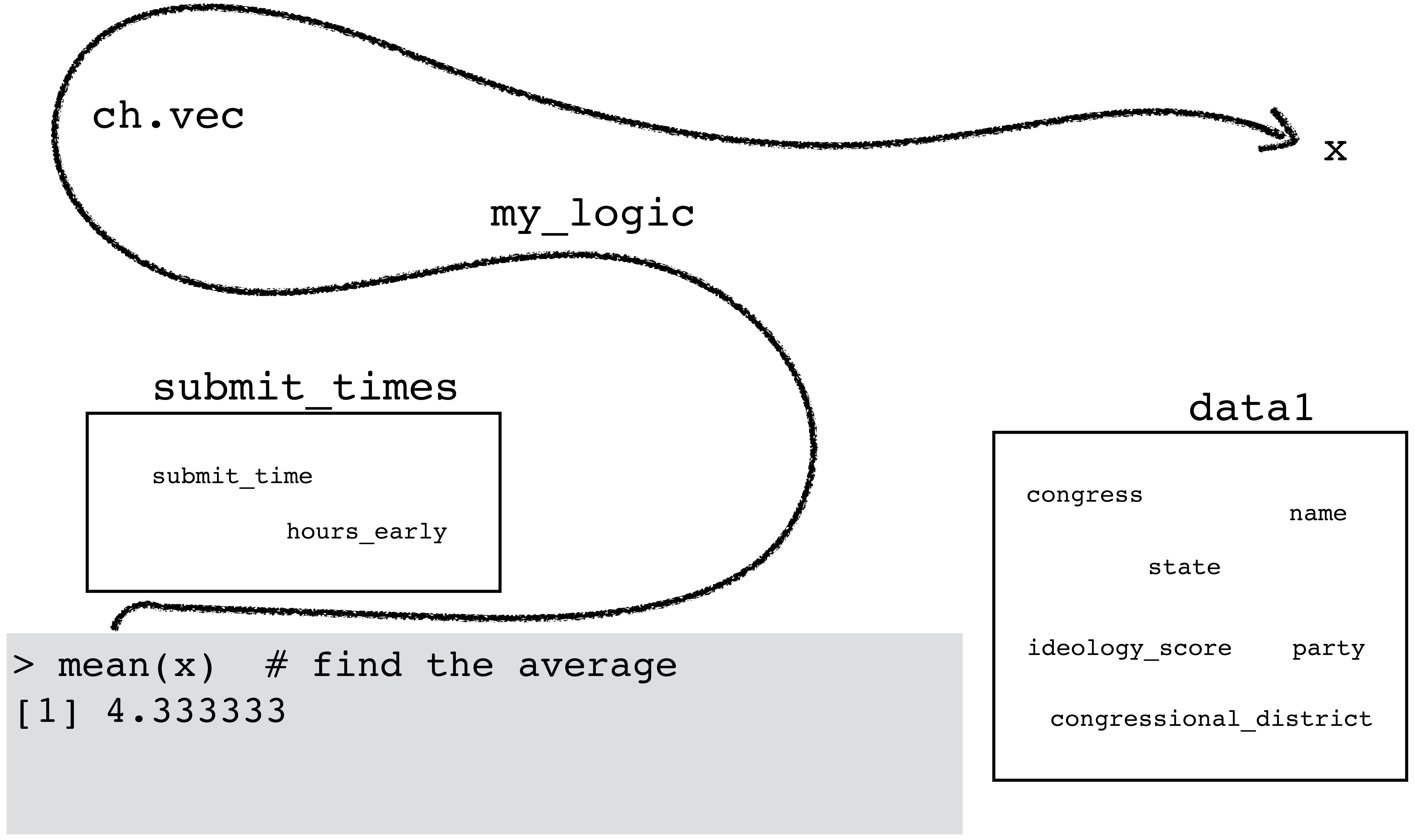

ch.vec
x
my_logic
submit_times
submit_time
hours_early
data1
congress
name
state
ideology_score
party
congressional_district



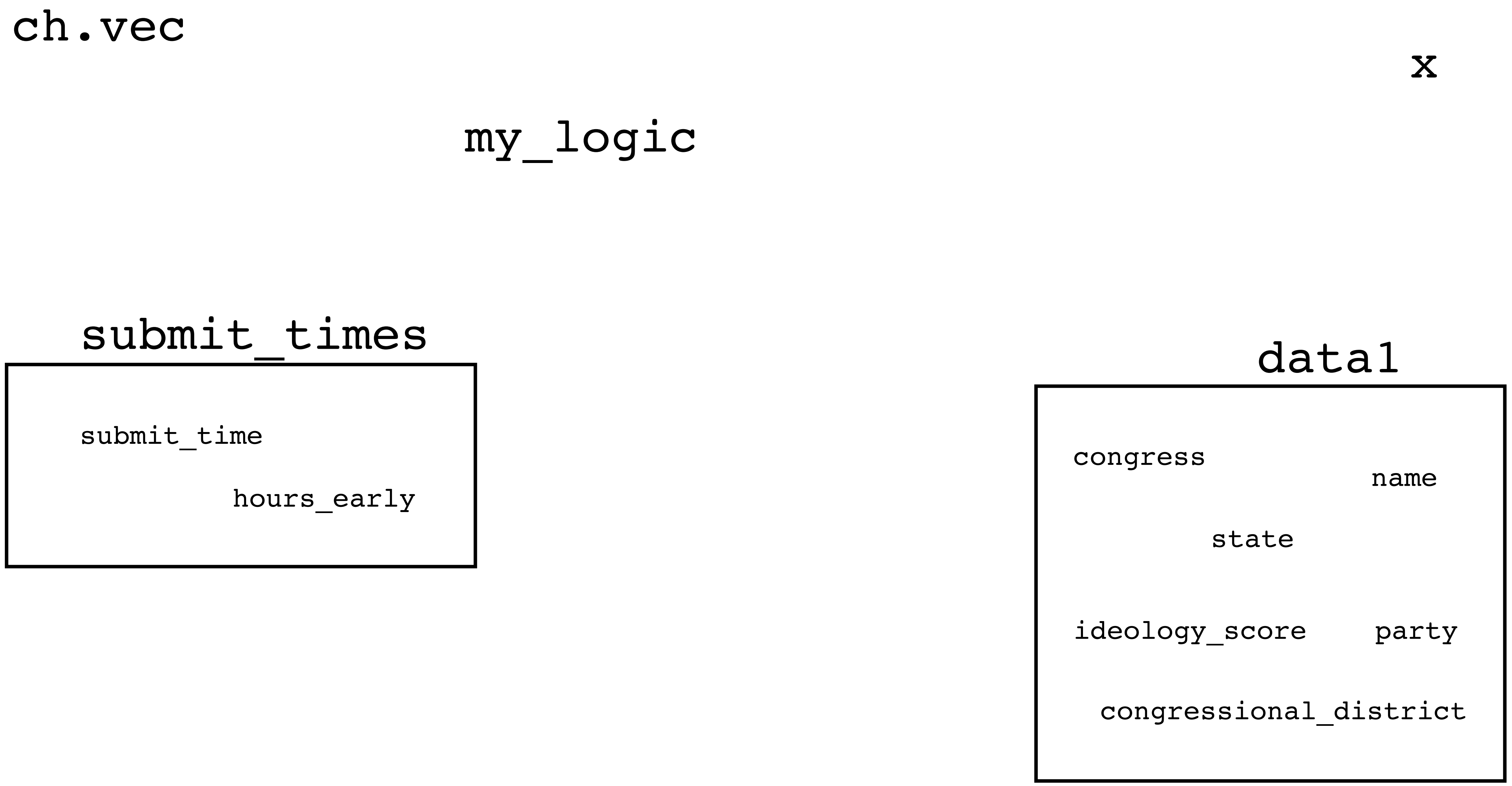
```
mean(x)  # find the average
```



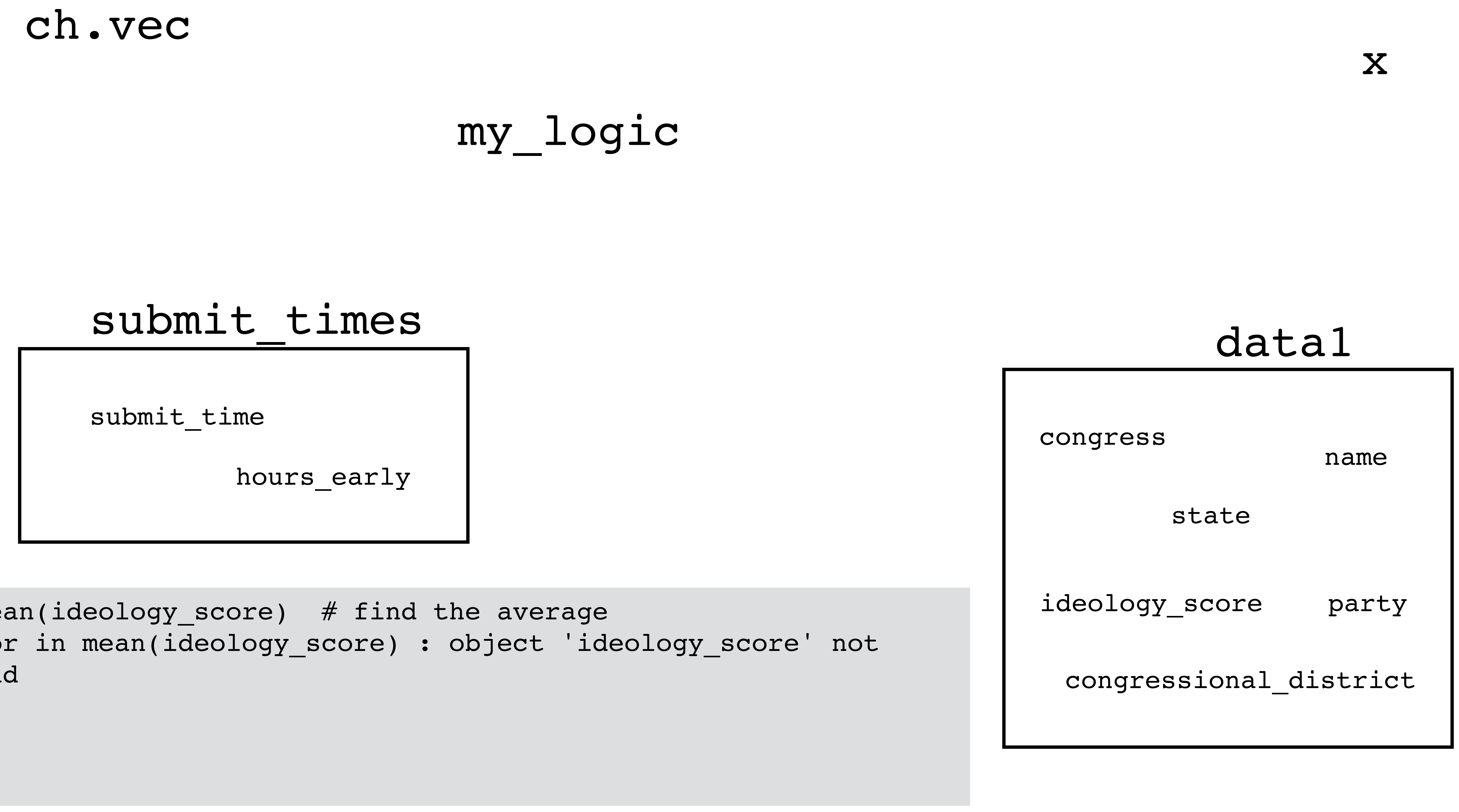
```
> mean(x)  # find the average
[1] 4.333333
```



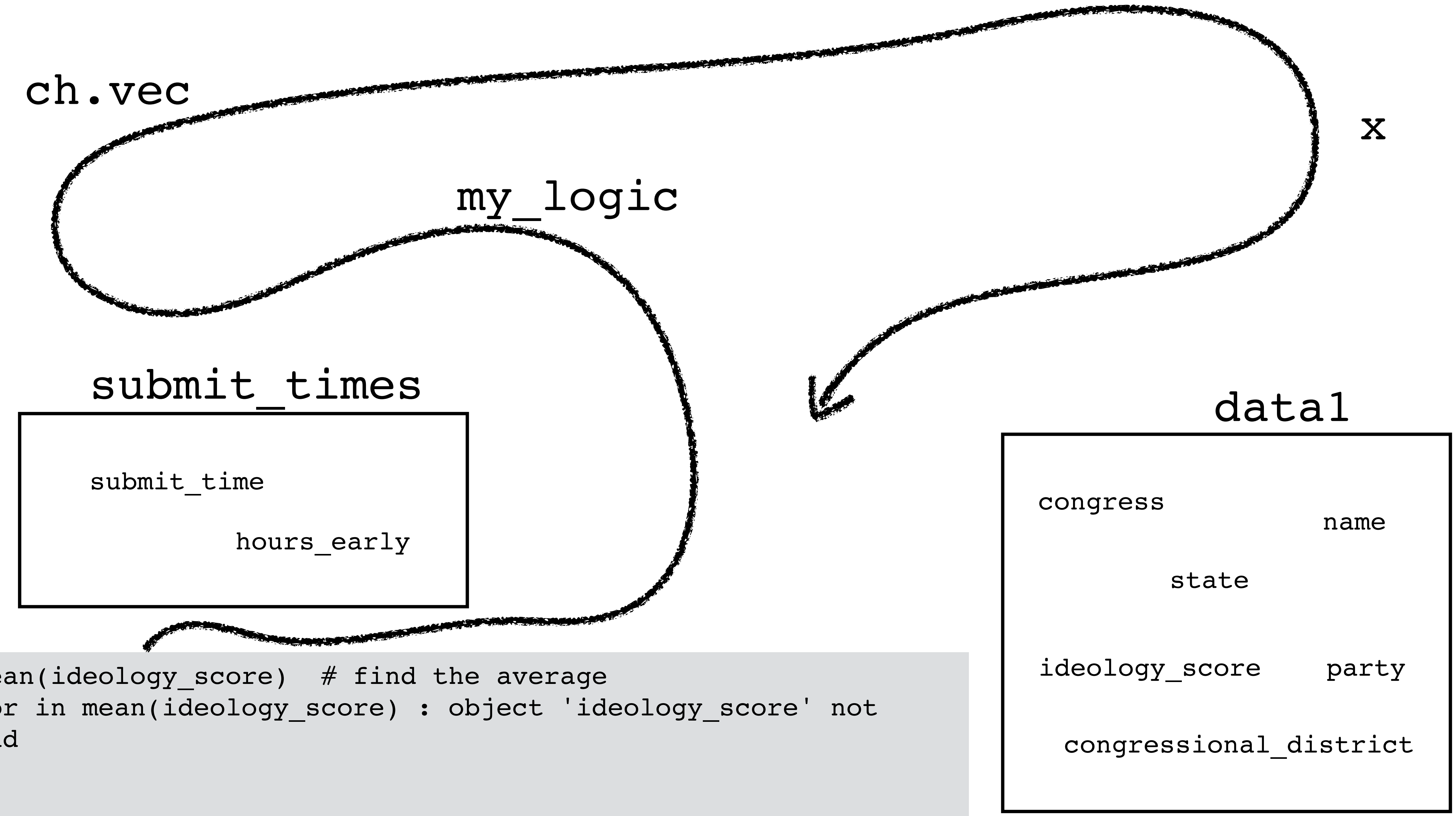
```
> mean(x)  # find the average
[1] 4.333333
```



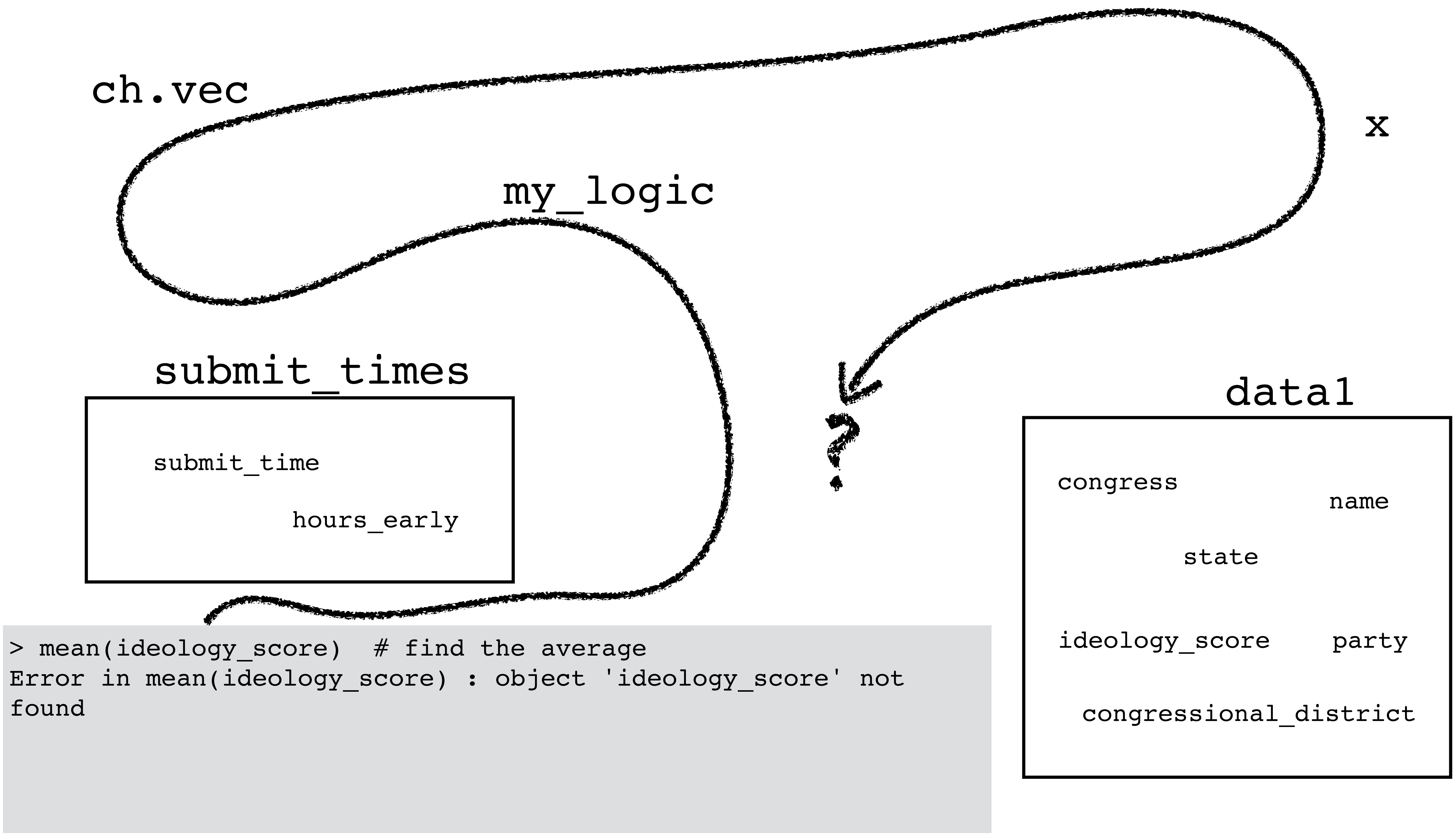
```
mean(ideology_score)  # find the average
```



```
> mean(ideology_score)  # find the average
Error in mean(ideology_score) : object 'ideology_score' not
found
```

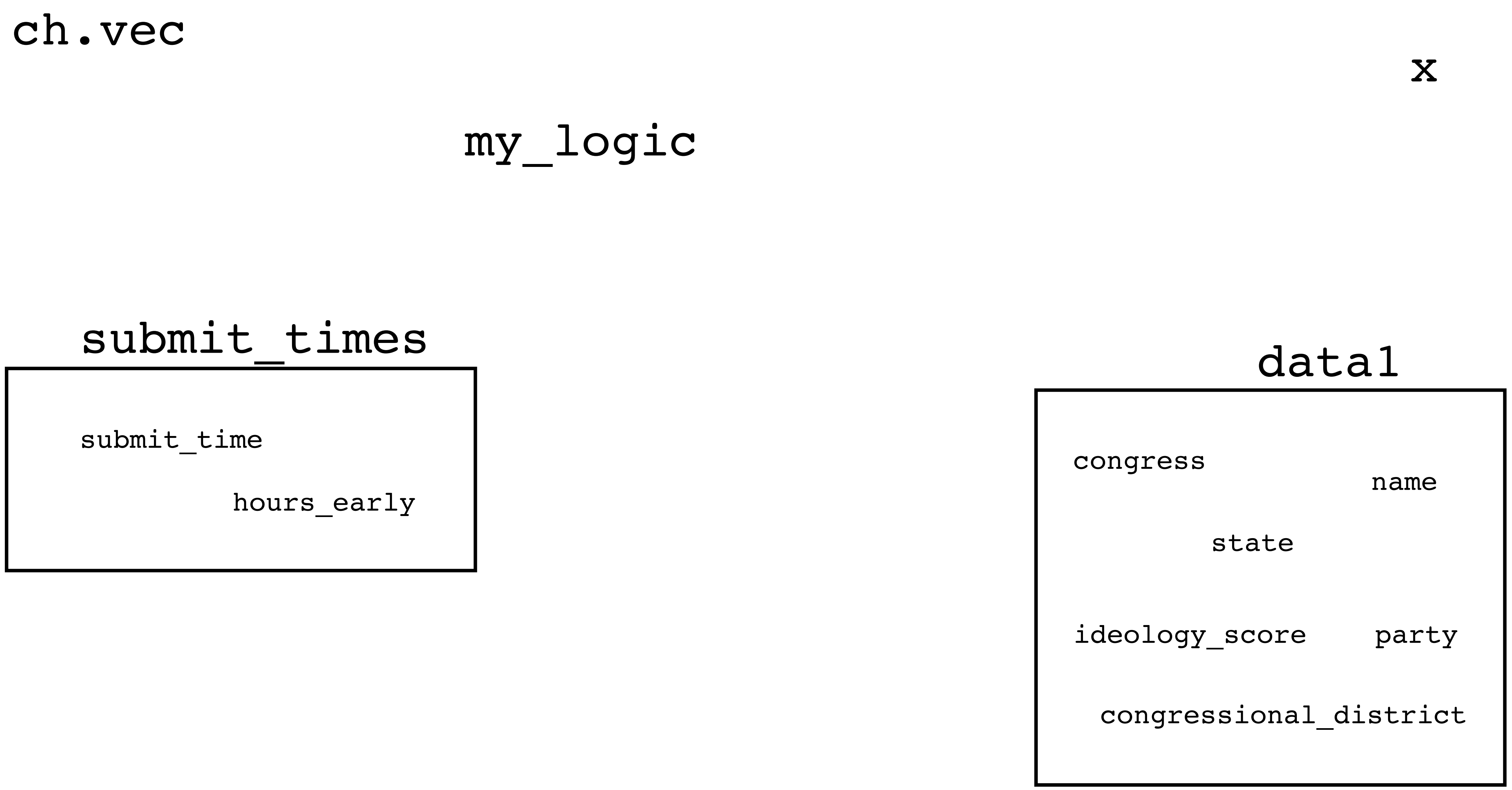


```
> mean(ideology_score)  # find the average
Error in mean(ideology_score) : object 'ideology_score' not
found
```

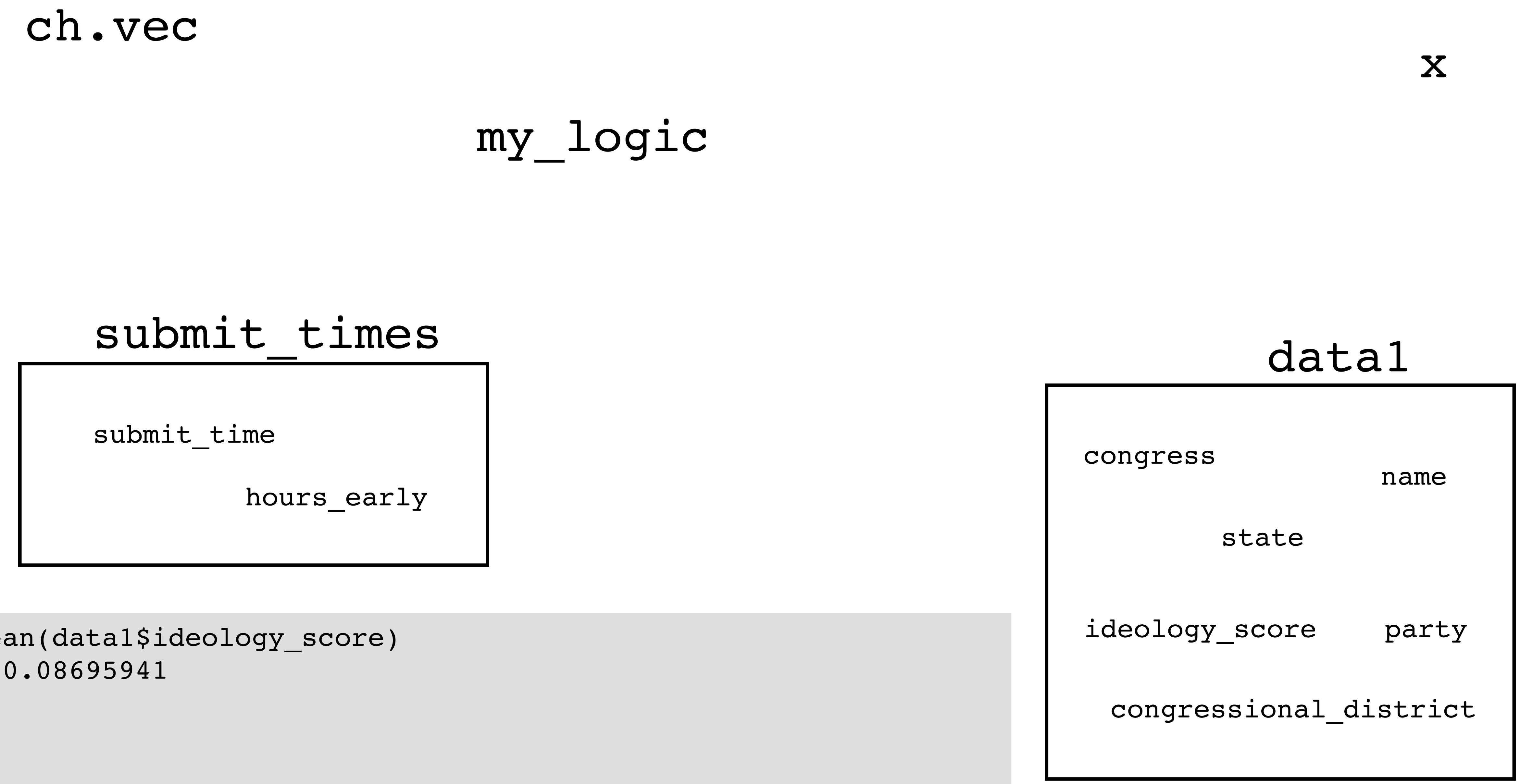


```
> mean(ideology_score)  # find the average
Error in mean(ideology_score) : object 'ideology_score' not
found
```

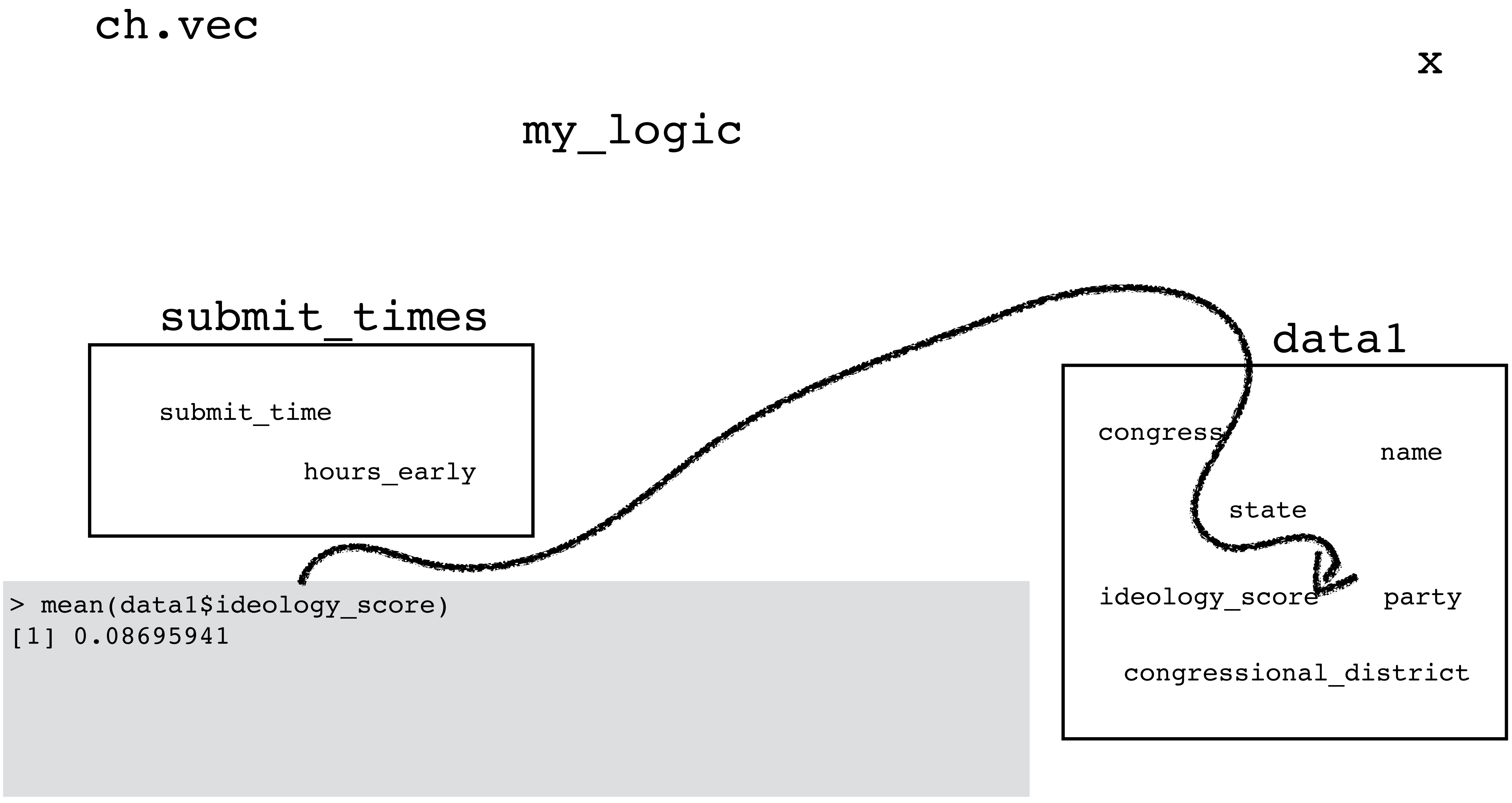
When looking for a vector,
R does not look inside data
frames unless you ask it.



```
mean(data1$ideology_score)  # find the average
```



```
> mean(data1$ideology_score)
[1] 0.08695941
```



```
> mean(data1$ideology_score)
[1] 0.08695941
```

the key syntax

```
data$variable
```

However, **most** functions for modeling and graphing are designed to work with data frames via a `data =` argument, not vectors

- no: mean(), sd(), log(), sqrt()
- yes: ggplot(), lm()

If the function takes (and you supply) a data argument, then you do **not** need to use `data$variable`.