`attritevis` Package: An R Vignette

This version: September 2022

Introduction

Attrition, the loss of study units from a sample, can often occur throughout an experimental study and at times pose a threat to inference. There are several studies, and accompanying R packages, that provide ex-post solutions to missingness such as double-sampling or extreme bounds. We provide a visually based guidance to assessing the types of missingness a study may have with a particular eye towards experimental and design adjustments a researcher can make after piloting a study.

Usage

- Visualizing survey attrition across treatment condition and over-time.
- Utilizing and comparing balance tests at precise moments in the survey.
- Incorporating estimation and visualization of Manski bounds for studies suffering from problematic attrition.

Assumptions

- Data must be *ordered by survey questions*, i.e. if respondents answered Q1 before Q2, the variable Q1 must appear before Q2 (i.e. in an earlier column) in the dataframe.
- When attrition is defined as completely leaving the survey. Hence, when attrition is reported in the package it does not count skippers, i.e. respondents who skipped a question(s) but continued later in the survey, as attrited. In the function `plot_attrition` users can count skippers by setting y = "responded".
- For balance tests, treatment and control conditions must be defined.

Functions

`attrition()`

Description

- Converts survey data into a frame that includes:
  - `attrited` – how many respondents attrited (left the survey) at each question.
  - `proportion` – number of attrited respondents / number of respondents who entered survey.
  - `prop_q` – number of attrited respondents / number of respondents entering into the question. - `questions` – question names.
  - `responded` – how many respondents responded in each question.
  - `prop_r` – number of respondents who responded / number of respondents who entered survey.

Arguments

- `data` - a `data.frame` where variables are ordered by survey questions, such that earlier survey questions appear in smaller valued columns.

attrition_table()

Description

- Yields same data.frame as function `attrition`, but converts it into a table. Allows to subset table by treatment and control groups, which yields several tables by condition.

Arguments

- `data` - a `data.frame` where variables are ordered by survey questions, such that earlier survey questions appear in smaller valued columns.
- `treatment_q` - a `string` character that corresponds to treatment variable. When specified, the function yields several tables by condition.

plot_attrition()

Description

- Plots attrition or response in survey data over time.

Arguments

*`data` must be data.frame. Note that this function works only if the order of variables = order of questions in the survey. Users must remove irrelevant observations, for instance individuals who did not meet quotas, so as not to confuse them with attrited respondents. Note that using the qualtrics embedded data feature users can note which respondents failed an attention check or were removed since they did not meet a quota.

*`y` is a character that corresponds to the Y axis. When y = `attrited`, attrition is plotted. When y= `responded` responses are plotted. Default is y = `attrited`.

*`freq` is a logical argument that notes whether Y axis is a raw number or a proportion. Default is freq=TRUE, which is the frequency of attrited OR responded respondents. When freq=FALSE Y axis is the proportion of total N (attrited OR responded), calculated as number of attrited OR responded divided by the number of respondents entering into the question.

*`treatment_q` is a character of name(s) of question(s) in which treatments were administered. Marked in the plot with a red vertical line.

*`outcome_q` is a character of name(s) of outcome question(s). Marked in the plot with a blue vertical line.

*`mycolors` is a character of color names to be used as values in `scale_colour_manual` argument in ggplot. Default is mycolors=NULL, which defaults to greyscale. `mycolors` must be == length of the unique values of the `treatment_q` variable. To use this argument, users should specify which color corresponds to which factor (for example, `treatment = "red"`).

*`title` is a character to be used for plot title.

balance_cov()

Description

- Tests whether specified covariates are balanced across specified treatment and control groups. Output is a t-test if covariate is a numeric or integer, and a 2-sample proportion test if covariate is a factor.

Arguments

- `data` - a data.frame, from which `treatment` and `question` are taken.
- `treatment` - a `string` character that corresponds to the name of the treatment variable. Note that values of said variable must be specified as `treatment` and `control`.
- `question` - a `string` character that corresponds to the name of the point in survey (question), for which balance test is required.
- `factor` - logical argument that specifies whether `question` is a factor. Default is `factor = FALSE` (i.e. question is a numeric or integer).
- `factor_name` - character that corresponds to specific factor (i.e. female), if question is a factor (i.e. sex).
- `p_adjust` - Vector of numbers that correspond to p-values obtained in all tests. Use this to adjust for p-values if running multiple tests.

`balance_attrite()`

Description

- Tests whether specified treatment causes attrition in a specified question. Output is a logistic regression, regressing attrition (remain in survey=0, attrited=1) over specified treatment.

Arguments

- `data` - a data.frame, from which `treatment` and `question` are taken.
- `treatment` - a `string` character that corresponds to the name of the treatment variable. Note that values of said variable must be specified as `treatment` and `control`.
- `question` - a `string` character that corresponds to the name of the point in survey (question), for which balance test is required.

`bounds()`

Description

- Yields extreme (Manski) bounds or trimming (Lee) bounds, using the `attrition` package by Alex Coppock.

Arguments

- `data` - a data.frame, from which `treatment` and `DV` are taken.
- `treatment` - a `string` character that corresponds to the name of the treatment variable. Note that values of said variable must be specified as `treatment` and `control`.
- `DV` - a `string` character that corresponds to the name of the outcome variable.
- `type` - character that corresponds to the type of bounds required (`"Manski"` or `"Lee"`). Default is `type = "Manski"`.

`vis_miss_treat()`

Description

- Calls the `vis_miss` function from `visdat` package. We allow users to facet missingness by conditions, creating several missingness maps per condition.

Arguments

- `data` - a data.frame.
- `treatment_q` - a `string` character that corresponds to the name of the treatment variable. If treatment_q = NULL, missingness map appears for all data, when treatment_q is not NULL, missingness is faceted by condition.

Example

Let's begin demonstrating the uses of `attritevis`, with a working example. We load test data from Lo, Renshon, and Bassan-Nygate 2021 (study 5B) which is an experimental survey study on whether peer-praise can encourage respondents to choose an empathy task.

The experiment manipulates peer-praise and measures empathy in a behavioral task. There are two arms in the peer-praise randomization: peer-praise and no praise (control). In the first arm, a word cloud of praise, drawn from real praise collected in a pilot study, is given for people who behave empathetically, with a line of text about peer group average thermometer ratings towards people who are empathetic – "Peers of yours on this platform have said they hold favorable feelings towards people who engage in empathetic behavior, with an average of 7.9, on a scale of 0 (least favorable) to 10 (most favorable), That same peer group provided real feedback for empathetic behavior which is pictured in the word cloud below". The word cloud is presented in figure 1. Respondents in the control condition do not receive any additional information.

Figure 1: Figure 1: Word cloud of real praise presented to treated respondents.

Our outcome of interest is choosing to empathize with an image in a behavioral task. In the task, subjects choose between two "cards" a FEEL and a DESCRIBE task, that correspond to an empathy or objective task, in which they empathize/describe an image of a man. The cards are presented in figure 2. Below is a description of the survey, with information on the various variables collected.

After answering pre-treatment covariates, respondents in the study were asked to complete two practice rounds of the main empathy task. After completing the practice rounds, respondents complete three trials of the above mentioned tasks. Before each task, respondents are randomized into treatment and control groups. Treated respondents received the light-touch peer-praise treatment. During each trial, before respondents select between the FEEL and DESCRIBE tasks, happiness, the hypothesized mechanism, is measured. Treatment variables are labeled `treat1`, `treat2`, etc. Outcome variables, which are the choice-task card

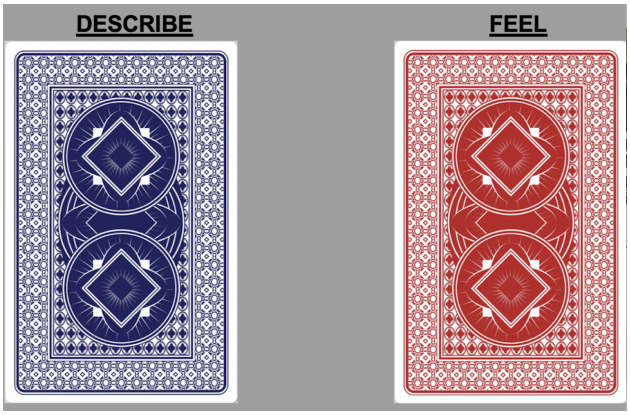


Figure 2: Figure 2: Choice task FEEL and DESCRIBE cards.

questions, are labeled `card1`, `card2`, etc. Mediators, which are measures of the emotion happiness, are labeled `Happy_1_1`, `Happy_1_2`... `Happy_2_1`, `Happy_2_2`... `Happy_3_1`, `Happy_3_1`, etc. After respondents complete all three trials post-task and post-treatment covariates are collected. Importantly, the dataframe `test_data` is organized based on the survey questions order. That is, if Q1 came before Q2 in the survey, the variable Q1 comes before the variable Q2 in the dataframe.

After loading the test data and ensuring that variables are ordered by survey questions, we may want to transform our dataframe to an attrition dataframe, using the function `attrition`.

Attrition dataframe

```
attrition_data <- attrition(test_data)
```

This function creates a frame that indicates, per question:

```
- 'attrited' -- how many respondents attrited (left the survey) at each question.
- 'proportion' -- number of attrited respondents / number of respondents who entered survey.
- 'prop_q' -- number of attrited respondents / number of respondents entering into the question.     -
- 'responded' -- how many respondents responded in each question.
- 'prop_r' -- number of respondents who responded / number of respondents who entered survey.
```

Using base `R` we can explore how many people attrited overall, and what proportion of the general population this is.

```
sum(attrition_data$attrited) #How many respondents attrited overall?
```

```
## [1] 129
```

```
sum(attrition_data$attrited)/nrow(test_data) #What proportion of the overall sample is this? (0.21)
```

```
## [1] 0.2067308
```

Next, we can look at specific variables, and learn whether respondents attrited. Let's choose the variable `cards_a` to demonstrate. This is a variable that notes whether respondents clicked the "FEEL" or "DESCRIBE" button during their first practice round. Using base `R` we can extract the number of attrited respondents, as well as the proportion of total N attrited, for this question.

```
attrition_data[attrition_data$questions == 'cards_a', 'attrited']
```

```
## [1] 37
```

```
attrition_data[attrition_data$questions == 'cards_a', 'proportion']
```

```
## [1] 0.06
```

We learn that at the question `cards_a` 37 respondents attrited from the survey. This is equivalent to 6% of the number of respondents who entered the survey at this question. Is this a lot though? Where else do we see attrition in the study? To assess, we visualize attrition across the survey timeline.

Attrition table

We can further create that of this dataframe using the function `attrition_table`.

```
attrition_table(test_data)
```

attrited	prop_q	proportion	questions	responded	prop_r
0	0.00	0.00	consent	624	1.00
3	0.00	0.00	age	621	1.00
0	0.00	0.00	sex	618	0.99
0	0.00	0.00	education	621	1.00
0	0.00	0.00	state	621	1.00
0	0.00	0.00	income	621	1.00
0	0.00	0.00	part_id	621	1.00
0	0.00	0.00	race	621	1.00
0	0.00	0.00	religion	621	1.00
1	0.00	0.00	attrition_1	620	0.99
6	0.01	0.01	attrition_2	614	0.98
37	0.06	0.06	cards_a	577	0.92
0	0.00	0.00	pa	553	0.89
0	0.00	0.00	pb_1	536	0.86
0	0.00	0.00	pb_2	536	0.86
0	0.00	0.00	pb_3	536	0.86
0	0.00	0.00	pc	534	0.86
0	0.00	0.00	cards_b	534	0.86
0	0.00	0.00	p2a	522	0.84
0	0.00	0.00	p2b_1	515	0.83
0	0.00	0.00	p2b_2	515	0.83
0	0.00	0.00	p2b_3	515	0.83
0	0.00	0.00	p2c	515	0.83
0	0.00	0.00	treat1	577	0.92
0	0.00	0.00	Happy_1_1	514	0.82
0	0.00	0.00	Happy_1_2	514	0.82
0	0.00	0.00	Happy_1_3	514	0.82
0	0.00	0.00	cards1	505	0.81
0	0.00	0.00	X1a	502	0.80
0	0.00	0.00	X1b_1	502	0.80
0	0.00	0.00	X1b_2	502	0.80
0	0.00	0.00	X1b_3	502	0.80
0	0.00	0.00	X1c	502	0.80
0	0.00	0.00	treat2	577	0.92
0	0.00	0.00	Happy_2_1	502	0.80
0	0.00	0.00	Happy_2_2	502	0.80
0	0.00	0.00	Happy_2_3	502	0.80
0	0.00	0.00	cards2	500	0.80
0	0.00	0.00	X2a	498	0.80
0	0.00	0.00	X2b_1	497	0.80
0	0.00	0.00	X2b_2	497	0.80
0	0.00	0.00	X2b_3	497	0.80
0	0.00	0.00	X2c	497	0.80
0	0.00	0.00	treat3	577	0.92
80	0.14	0.13	Happy_3_1	497	0.80
0	0.00	0.00	Happy_3_2	497	0.80
0	0.00	0.00	Happy_3_3	497	0.80
0	0.00	0.00	cards3	497	0.80
0	0.00	0.00	X3a	497	0.80
0	0.00	0.00	X3b_1	497	0.80
0	0.00	0.00	X3b_2	497	0.80
0	0.00	0.00	X3b_3	497	0.80
0	0.00	0.00	post1	491	0.79
0	0.00	0.00	post2_7	497	0.80
0	0.00	0.00	post3	497	0.80
0	0.00	0.00	post4	497	0.80
0	0.00	0.00	post5	496	0.79

We can also use the argument `treatment_q` to facet attrition table by condition. This is a character that corresponds to a specific variable, which is where the treatment conditions were administered.

```
attrition_table(data= test_data,
                treatment_q = "treat1"
                )
```

[[1]]

attrited	prop_q	proportion	questions	responded	prop_r
0	0.00	0.00	control.consent	289	1.00
0	0.00	0.00	control.age	289	1.00
0	0.00	0.00	control.sex	287	0.99
0	0.00	0.00	control.education	289	1.00
0	0.00	0.00	control.state	289	1.00
0	0.00	0.00	control.income	289	1.00
0	0.00	0.00	control.part_id	289	1.00
0	0.00	0.00	control.race	289	1.00
0	0.00	0.00	control.religion	289	1.00
0	0.00	0.00	control.attrition_1	289	1.00
0	0.00	0.00	control.attrition_2	289	1.00
0	0.00	0.00	control.cards_a	289	1.00
0	0.00	0.00	control.pa	278	0.96
0	0.00	0.00	control.pb_1	270	0.93
0	0.00	0.00	control.pb_2	270	0.93
0	0.00	0.00	control.pb_3	270	0.93
0	0.00	0.00	control.pc	268	0.93
0	0.00	0.00	control.cards_b	268	0.93
0	0.00	0.00	control.p2a	262	0.91
0	0.00	0.00	control.p2b_1	260	0.90
0	0.00	0.00	control.p2b_2	260	0.90
0	0.00	0.00	control.p2b_3	260	0.90
0	0.00	0.00	control.p2c	260	0.90
0	0.00	0.00	control.cond_new	289	1.00
0	0.00	0.00	control.Happy_1_1	260	0.90
0	0.00	0.00	control.Happy_1_2	260	0.90
0	0.00	0.00	control.Happy_1_3	260	0.90
0	0.00	0.00	control.cards1	251	0.87
0	0.00	0.00	control.X1a	249	0.86
0	0.00	0.00	control.X1b_1	249	0.86
0	0.00	0.00	control.X1b_2	249	0.86
0	0.00	0.00	control.X1b_3	249	0.86
0	0.00	0.00	control.X1c	249	0.86
0	0.00	0.00	control.treat2	289	1.00
0	0.00	0.00	control.Happy_2_1	249	0.86
0	0.00	0.00	control.Happy_2_2	249	0.86
0	0.00	0.00	control.Happy_2_3	249	0.86
0	0.00	0.00	control.cards2	248	0.86
0	0.00	0.00	control.X2a	246	0.85
0	0.00	0.00	control.X2b_1	246	0.85
0	0.00	0.00	control.X2b_2	246	0.85
0	0.00	0.00	control.X2b_3	246	0.85
0	0.00	0.00	control.X2c	246	0.85
0	0.00	0.00	control.treat3	289	1.00
43	0.15	0.15	control.Happy_3_1	246	0.85
0	0.00	0.00	control.Happy_3_2	246	0.85
0	0.00	0.00	control.Happy_3_3	246	0.85
0	0.00	0.00	control.cards3	246	0.85
0	0.00	0.00	control.X3a	246	0.85
0	0.00	0.00	control.X3b_1	246	0.85
0	0.00	0.00	control.X3b_2	246	0.85
0	0.00	0.00	control.X3b_3	246	0.85
0	0.00	0.00	control.post1	244	0.84
0	0.00	0.00	control.post2_7_9	246	0.85
0	0.00	0.00	control.post3	246	0.85
0	0.00	0.00	control.post4	246	0.85
0	0.00	0.00	control.post5	246	0.85

[[2]]

attrited	prop_q	proportion	questions	responded	prop_r
0	0.00	0.00	treatment.consent	288	1.00
0	0.00	0.00	treatment.age	288	1.00
0	0.00	0.00	treatment.sex	288	1.00
0	0.00	0.00	treatment.education	288	1.00
0	0.00	0.00	treatment.state	288	1.00
0	0.00	0.00	treatment.income	288	1.00
0	0.00	0.00	treatment.part_id	288	1.00
0	0.00	0.00	treatment.race	288	1.00
0	0.00	0.00	treatment.religion	288	1.00
0	0.00	0.00	treatment.attrition_1	288	1.00
0	0.00	0.00	treatment.attrition_2	288	1.00
0	0.00	0.00	treatment.cards_a	288	1.00
0	0.00	0.00	treatment.pa	275	0.95
0	0.00	0.00	treatment.pb_1	266	0.92
0	0.00	0.00	treatment.pb_2	266	0.92
0	0.00	0.00	treatment.pb_3	266	0.92
0	0.00	0.00	treatment.pc	266	0.92
0	0.00	0.00	treatment.cards_b	266	0.92
0	0.00	0.00	treatment.p2a	260	0.90
0	0.00	0.00	treatment.p2b_1	255	0.89
0	0.00	0.00	treatment.p2b_2	255	0.89
0	0.00	0.00	treatment.p2b_3	255	0.89
0	0.00	0.00	treatment.p2c	255	0.89
0	0.00	0.00	treatment.cond_new	288	1.00
0	0.00	0.00	treatment.Happy_1_1	254	0.88
0	0.00	0.00	treatment.Happy_1_2	254	0.88
0	0.00	0.00	treatment.Happy_1_3	254	0.88
0	0.00	0.00	treatment.cards1	254	0.88
0	0.00	0.00	treatment.X1a	253	0.88
0	0.00	0.00	treatment.X1b_1	253	0.88
0	0.00	0.00	treatment.X1b_2	253	0.88
0	0.00	0.00	treatment.X1b_3	253	0.88
0	0.00	0.00	treatment.X1c	253	0.88
0	0.00	0.00	treatment.treat2	288	1.00
0	0.00	0.00	treatment.Happy_2_1	253	0.88
0	0.00	0.00	treatment.Happy_2_2	253	0.88
0	0.00	0.00	treatment.Happy_2_3	253	0.88
0	0.00	0.00	treatment.cards2	252	0.88
0	0.00	0.00	treatment.X2a	252	0.88
0	0.00	0.00	treatment.X2b_1	251	0.87
0	0.00	0.00	treatment.X2b_2	251	0.87
0	0.00	0.00	treatment.X2b_3	251	0.87
0	0.00	0.00	treatment.X2c	251	0.87
0	0.00	0.00	treatment.treat3	288	1.00
37	0.13	0.13	treatment.Happy_3_1	251	0.87
0	0.00	0.00	treatment.Happy_3_2	251	0.87
0	0.00	0.00	treatment.Happy_3_3	251	0.87
0	0.00	0.00	treatment.cards3	251	0.87
0	0.00	0.00	treatment.X3a	251	0.87
0	0.00	0.00	treatment.X3b_1	251	0.87
0	0.00	0.00	treatment.X3b_2	251	0.87
0	0.00	0.00	treatment.X3b_3	251	0.87
0	0.00	0.00	treatment.post1	247	0.86
0	0.00	0.00	treatment.post2_1	251	0.87
0	0.00	0.00	treatment.post3	251	0.87
0	0.00	0.00	treatment.post4	251	0.87
0	0.00	0.00	treatment.post5	250	0.87

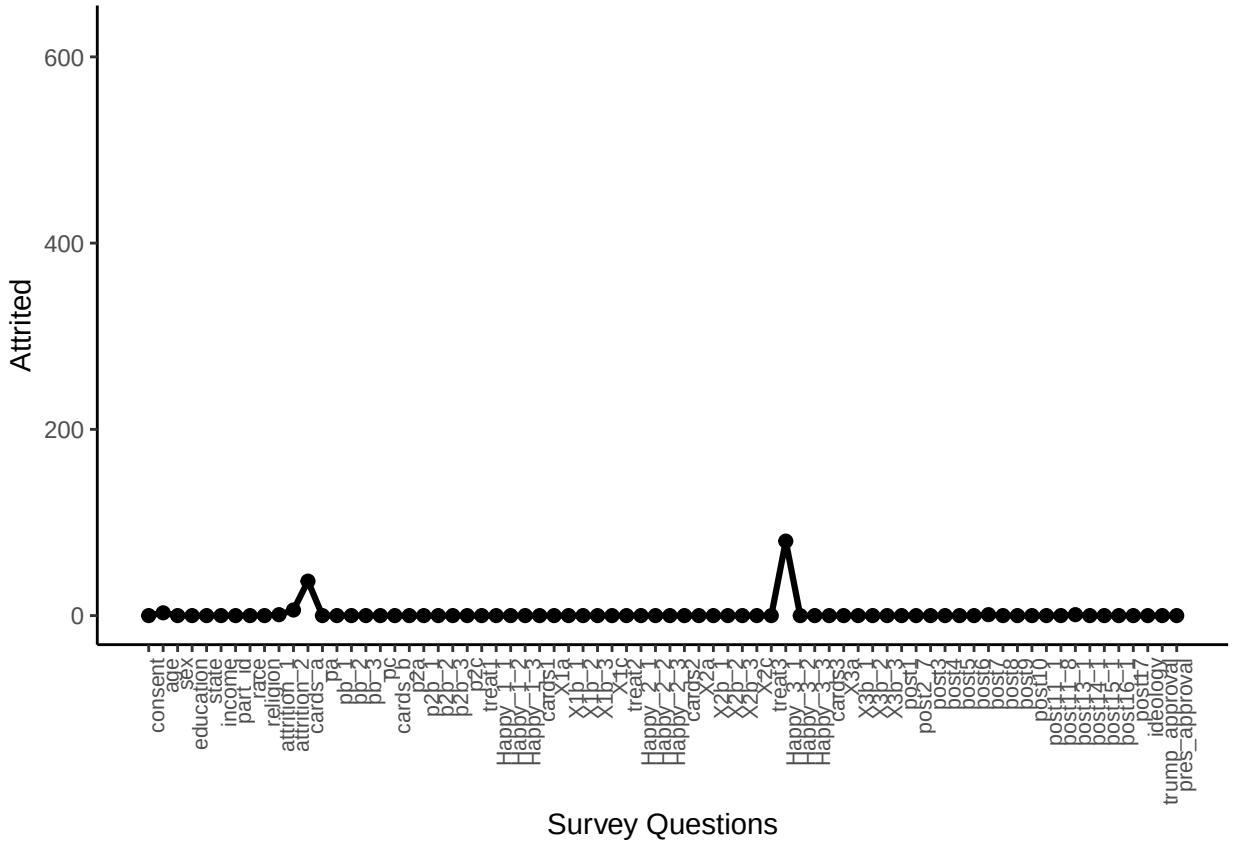
Visualizing attrition

Attrition timeline

We may want to visualize attrition across the survey, to look at all the survey questions at once. The function `plot_attrition` allows us to plot attrition across survey questions, indicating where treatment and outcome questions were collected.

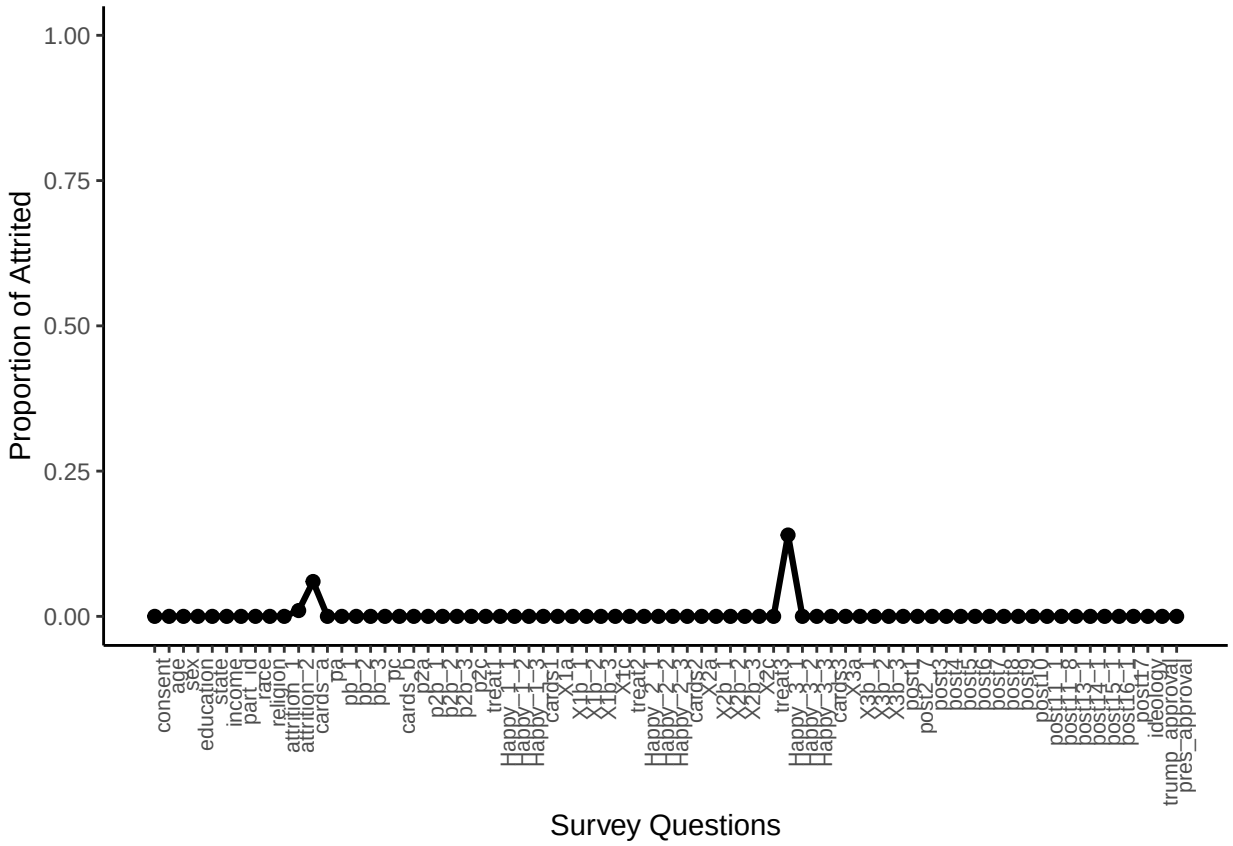
There are several ways in which users can use this function. Simply plugging in the dataset into the function yields a figure that plots the number of respondents that attrited (left the survey completely) over each question in the study.

```
plot_attrition(test_data)
```



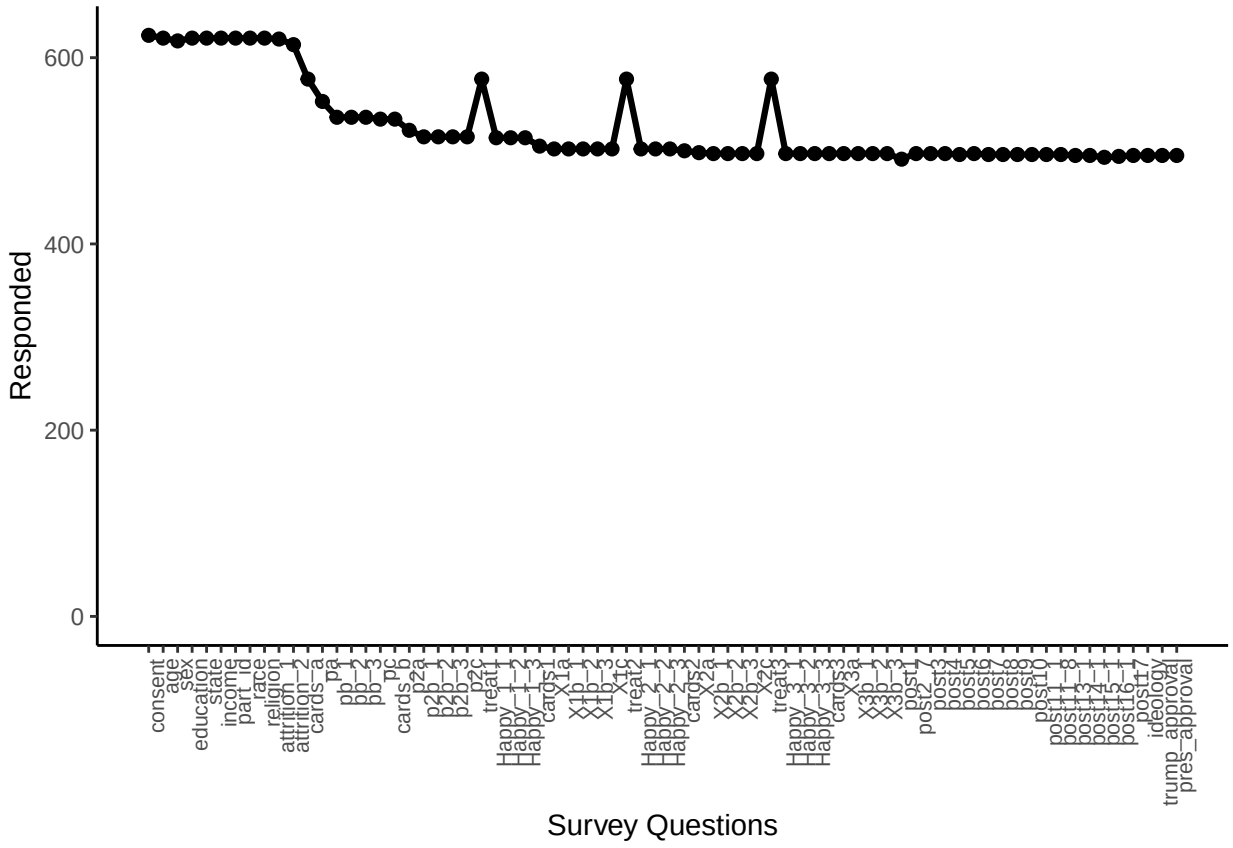
When users specify `freq=FALSE`, the y axis plots the proportion of attrited.

```
plot_attrition(test_data,
               freq=FALSE)
```



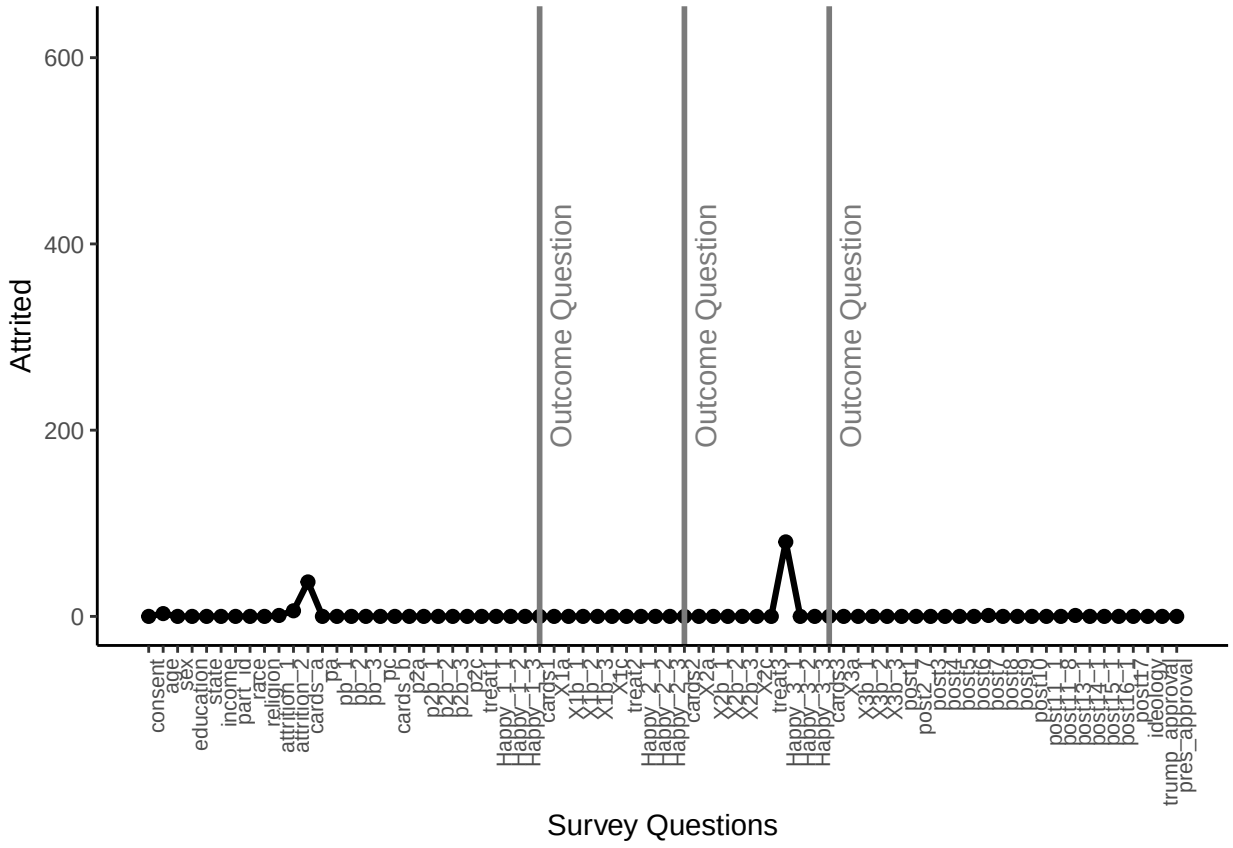
Users can further specify `y="resonded"` to account for response, rather than attrition. This argument can be used with either freq=TRUE (default), or freq=FALSE, plotting response or proportion of responded, accordingly.

```
plot_attrition(test_data,
               y="responded")
```



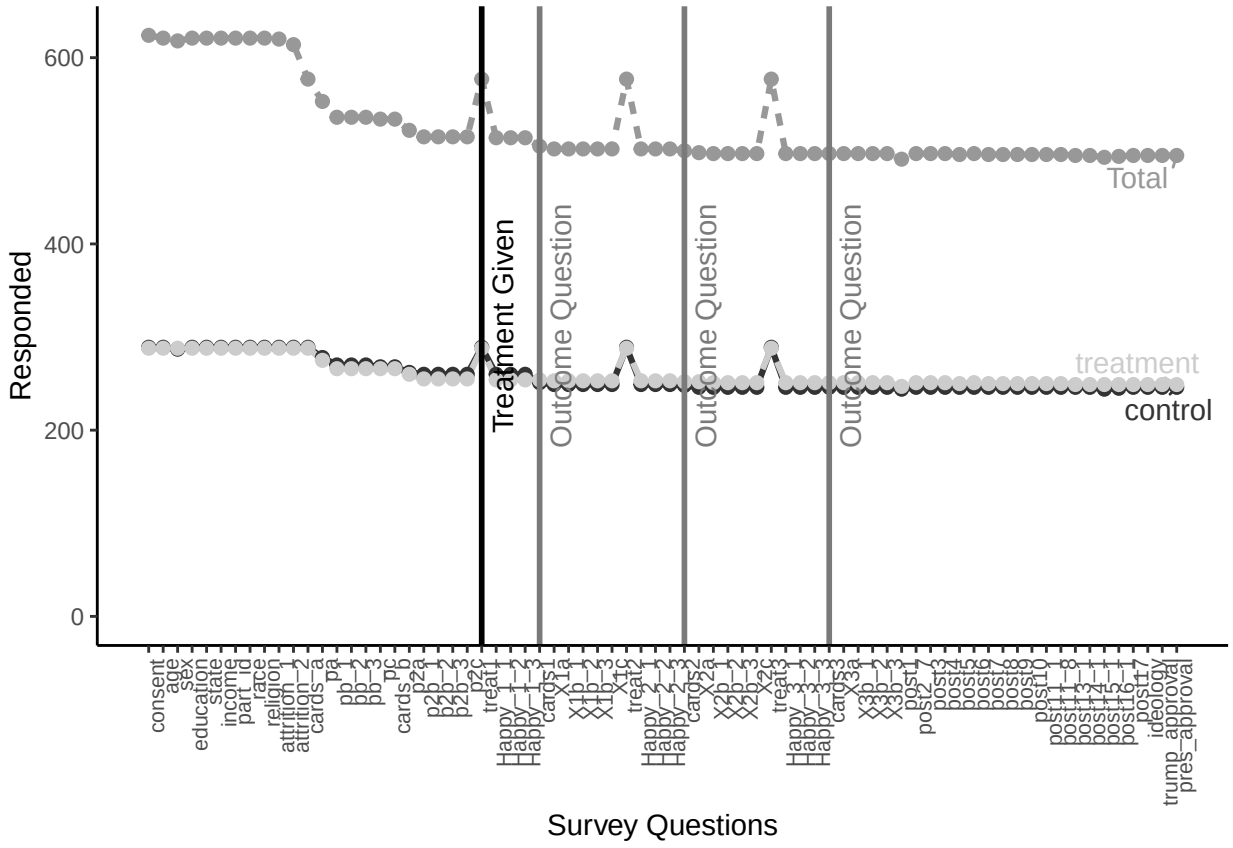
Using the `outcome_q` argument, users can specify where outcome questions were measure. These are noted with gray vertical lines.

```
plot_attrition(test_data,
               outcome_q = c("cards1", "cards2",  "cards3"))
```



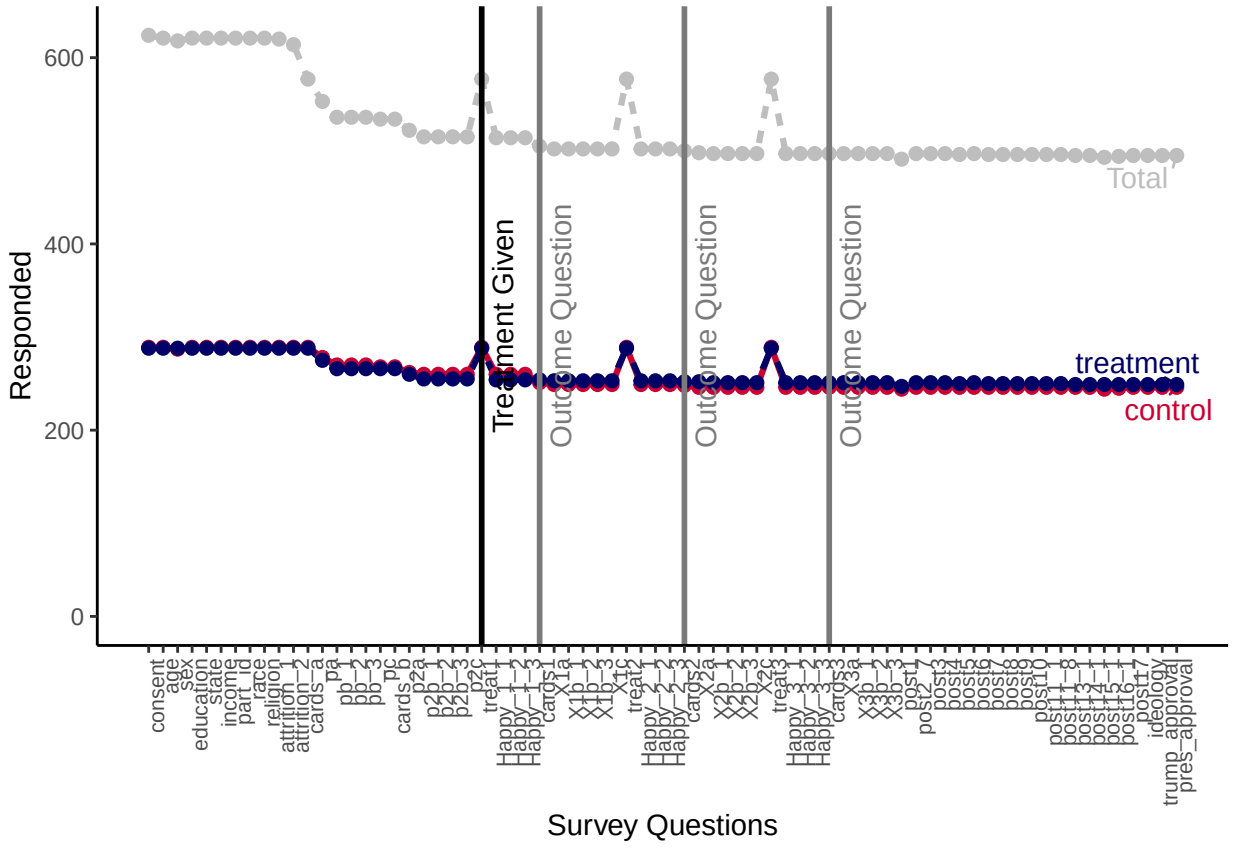
When `treatment_q`, which corresponds to treatment variable, is not NULL, the plot both notes where treatment was collected with a vertical line, and breaks down attrition by treatment conditions.

```
plot_attrition(test_data,
               y = "responded",
               outcome_q = c("cards1", "cards2",  "cards3"),
               treatment_q = "treat1")
```



Color default is greyscale, but users can use the `mycolors` argument to specify which colors they want to use to mark each conditions' geom_line. The length of `mycolors` must be equal to the length of unique(`treatment_q`). To use this argument, users should specify which color corresponds to which factor. See below the running example:

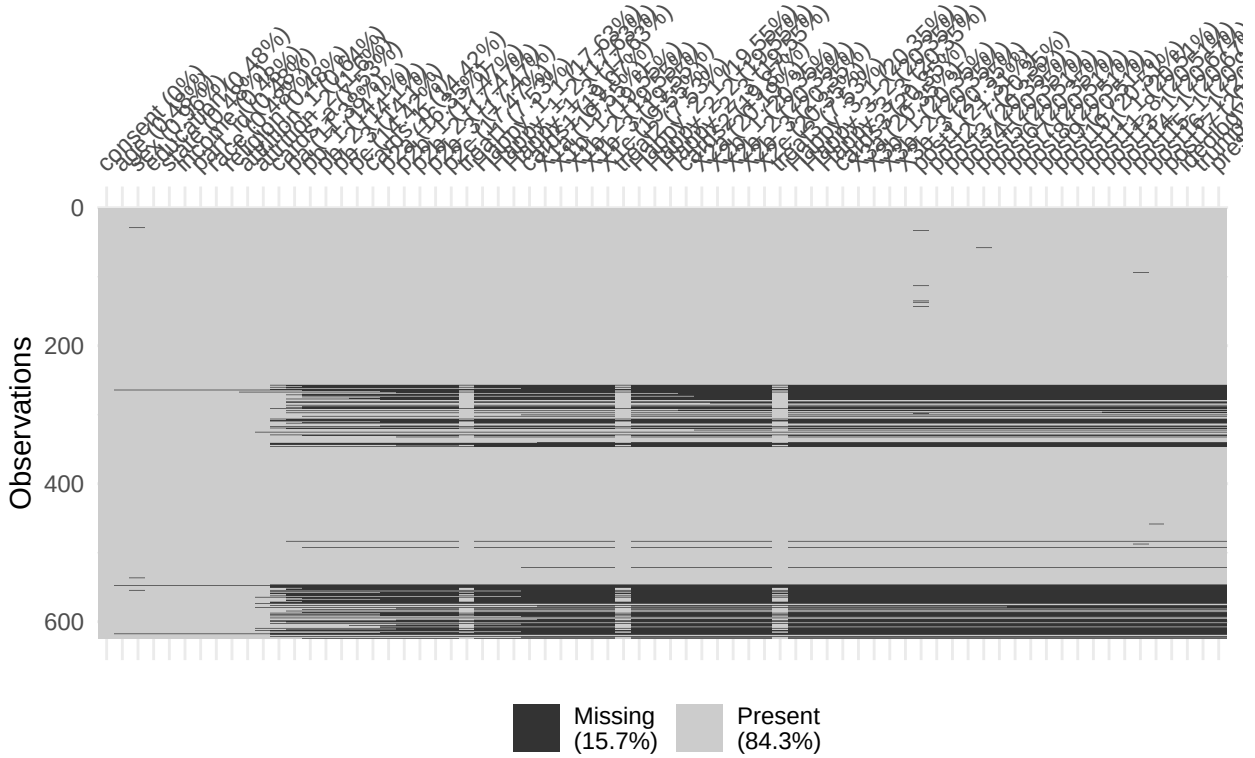
```
plot_attrition(test_data,
               y = "responded",
               outcome_q = c("cards1", "cards2",  "cards3"),
               treatment_q = "treat1",
               mycolors = c(treatment = "#000066",
                            control = "#CC0033"))
```



Vis miss

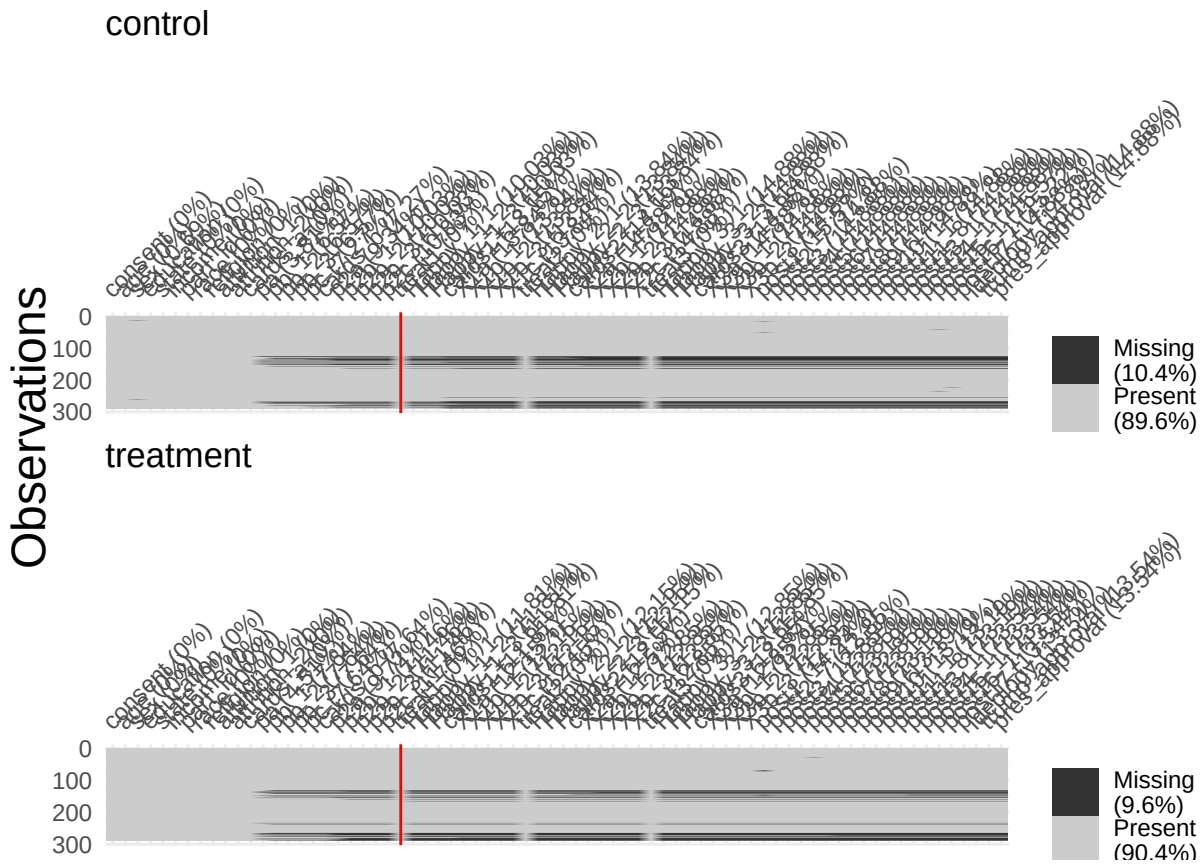
Users can also visualize missingness with the `vis_miss_treat()` function that calls the `vis_miss` function from the `visdat` package.

```
vis_miss_treat(test_data)
```



`attritevis` allows users to facet missingness by conditions, creating several missingness maps per condition, and marks treatment variable with a red vertical line.

```
vis_miss_treat(test_data,
               treatment_q = "treat1")
```



Balance tests

Once we have identified the specific survey points where attrition takes place, we want to conduct balance tests at these specific points to ensure balance across treatment and control, and learn if (and when) balance became an issue. We can do this using the functions `balance_cov()` and `balance_attrite()`.

Balance across covariates

Once we've identified whether (and when) attrition occurs in our survey, we want to know that our treatment and control groups are balanced across covariates throughout the survey, to detect differential attrition. We can do this using the function `balance_cov()`, which we will demonstrate with three covariates: `age`, `sex`, and `ideology`.

We begin with the covariate `age`, which was collected pretreatment and is a numeric variable. In order to use the function `balance_cov()` we must define `treatment` and `control` arms under the `treatment` variables. We define `treat1` as the treatment variable, and `age` as the question.

```
unique(test_data$treat1)
```

```
## [1] "treatment" "control"   NA
```

```
balance_cov(data = test_data,
        treatment = "treat1",
        question = "age")
```

```
##
##  Welch Two Sample t-test
##
## data:  treat_data$question1 and control_data$question1
## t = -0.32688, df = 568.57, p-value = 0.7439
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.002600  1.431137
## sample estimates:
## mean of x mean of y
##  37.42361  37.70934
```

The output is a t-test that determines whether there is a difference between the average age of the control group and the treatment group. We learn that age is balanced across treatment and control groups, with a mean of approximately 37.4 years old in treated respondents and 37.7 in controled respondents (p=0.7).

We can also use the function `balance_cov()` when the covariate (`question`) is a factor, but we must specify which factor we are interested in. For example, let's say we want to test whether at the question `sex` in the survey missingness created observable differences across treatment and control groups. Sex is a factor variable with two factors: female and male. We can look at whether the proportion of female still remains similar across groups. To do so, we must determine that `factor = TRUE` and specify the `factor_name` (in this case, female).

```
balance_cov(data = test_data,
        treatment = "treat1",
        question = "sex",
        factor = TRUE,
        factor_name = "female")
```

```
##
##  2-sample test for equality of proportions with continuity correction
##
## data:  x out of n
## X-squared = 1.1305, df = 1, p-value = 0.2877
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  -0.12931498  0.03623038
## sample estimates:
##    prop 1    prop 2
## 0.3576389 0.4041812
```

The output is a 2-sample proportion test. We learn that sex is also balanced between treatment and control, with similar proportions of females across the groups (p=0.3).

There are certain post-treatment variables for which we may want to ensure balance across treatment and control as well. Note, however, that these should be variables that we hypothesize would stay stable after treatment. For example, we occasionally include demographic questions at the end of the survey to avoid survey fatigue before treatments. In our running example, the `ideology` question was collected post-treatment, but we expect it to stay stable across treatment and control.

```
balance_cov(data = test_data,
        treatment = "treat1",
        question = "ideology")
```

```
##
##  Welch Two Sample t-test
##
## data:  treat_data$question1 and control_data$question1
## t = 1.023, df = 492.91, p-value = 0.3068
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.1660012  0.5266633
## sample estimates:
## mean of x mean of y
##  3.879518  3.699187
```

If users run several balance tests, we recommend adjusting p-values. `balance_cov` allows users to do so by specifying the p-values of the tests they ran (p_adjust).

In our running example, since we ran three balance tests (with ideology, sex, and age), we want to adjust our p-values:

```
balance_cov(data = test_data,
        treatment = "treat1",
        question = "ideology",
        p_adjust = c(0.7,0.3,0.3))
```

```
##
##  Welch Two Sample t-test
##
## data:  treat_data$question1 and control_data$question1
## t = 1.023, df = 492.91, p-value = 0.3068
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.1660012  0.5266633
## sample estimates:
## mean of x mean of y
##  3.879518  3.699187
```

Original_p....p_adjust	Adjusted_p....a
0.7	0.70
0.3	0.45
0.3	0.45

Balance across attrition

Next, we can check whether our treatment is correlated with attrition at any moment in the survey. The `balance_attrite()` function converts the specified `question` into a binary variable such that attrition = 1, and remaining in survey = 0, and runs a lositic regression (regressing the specified `question` over the specified `treatment`) to examine whether treatment affects attrition.

Using our visualization, we identified that attrition occurs at the post-treatment question `Happy_3_1`. We can use the function `balance_attrite()`, to examine whether our treatment caused attrition at this point in the survey:

```
balance_attrite(data = test_data,
        treatment = "treat1",
        question = "Happy_3_1")
```

```
##
## Call:
## glm(formula = question1 ~ treatment1, family = binomial(link = "logit"),
##     data = data2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.5676  -0.5676  -0.5244  -0.5244   2.0259
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -1.7441     0.1653 -10.552   <2e-16 ***
## treatment1treatment  -0.1704     0.2415  -0.706     0.48
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 464.49  on 576  degrees of freedom
## Residual deviance: 463.99  on 575  degrees of freedom
##   (47 observations deleted due to missingness)
## AIC: 467.99
##
## Number of Fisher Scoring iterations: 4
```

We learn that our `treat1` does not affect attrition in variable `Happy_3_1`.

Simulated data

As we demonstrated above, attrition doesn't seem to pose a threat to inference in our dataset. But what does it look like when attrition is an issue? We simulate attrition on `test_data` to demonstrate what this would look like.

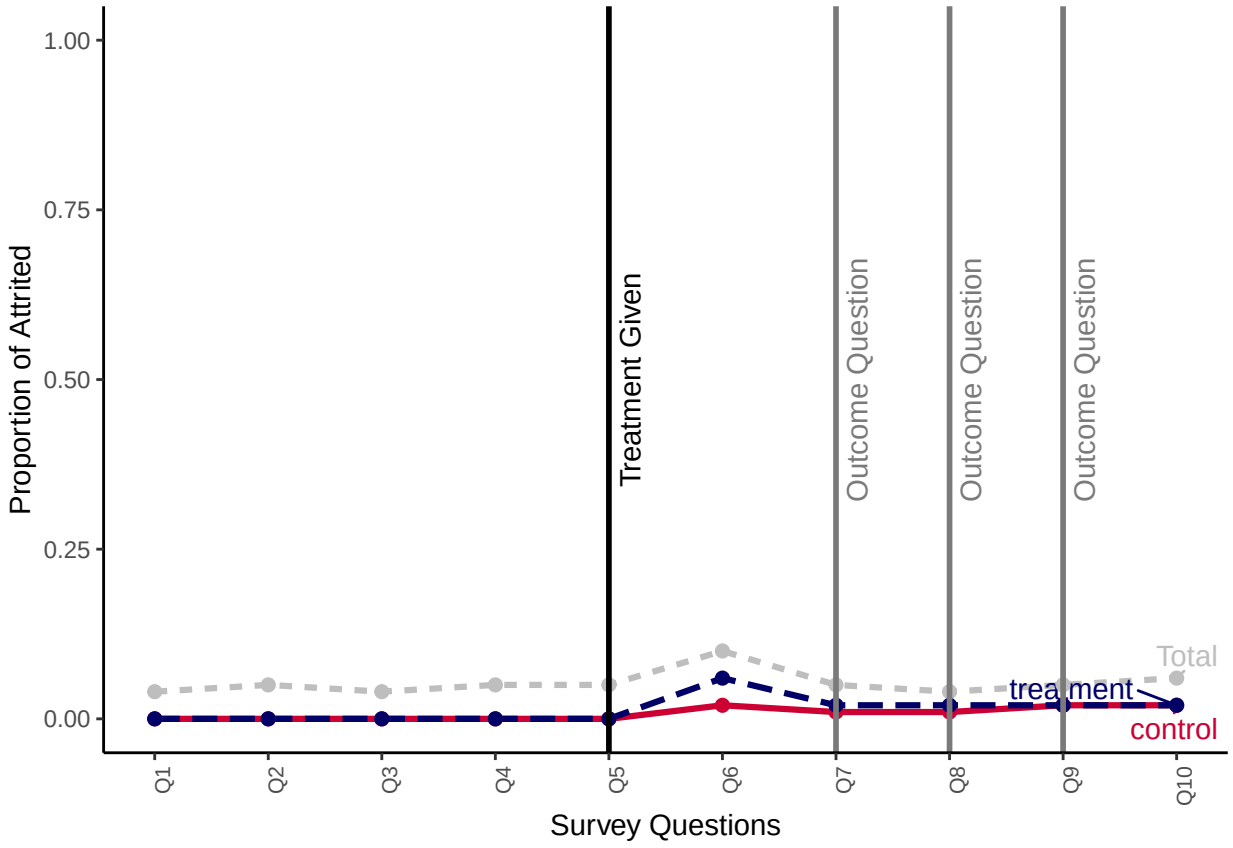
Treatment causes attrition

In a toy example (`test_sim`), suppose respondents enter a survey (Q1-Q10), where treatment is assigned at "Q5". We generate attrition such that treatment is causing respondents to drop out of the survey right after treatment. We might see something like this if respondents are particularly taxed by a treatment in the survey and therefore more likely to drop out after receiving treatment.

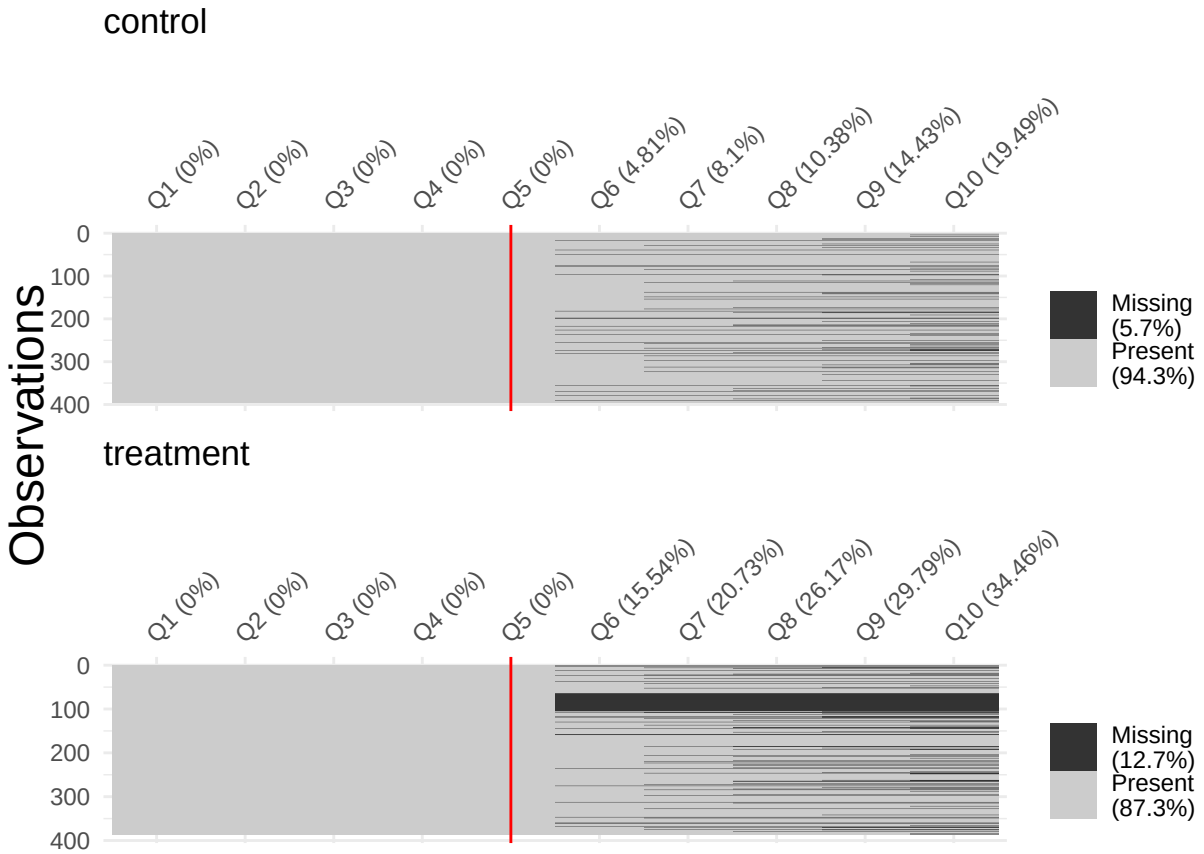
Plot

We visualize attrition using the `plot_attrition()` and `vis_miss_treat()` functions.

```
plot_attrition(test_sim,
               treatment_q = "Q5",
               outcome_q = c("Q7", "Q8",  "Q9"),
               freq = FALSE,
               mycolors = c(treatment = "#000066",
                            control = "#CC0033")
               )
```




```
vis_miss_treat(test_sim, treatment_q = "Q5")
```



We learn that attrition mostly occurs after Q6, and that treated respondents seem to be attriting more.

Balance

We learn that most respondents attrite at the post-treatment question `Q6`, and conduct a balance test. Note that `Q6` is an outcome, and we expect our treatment to affect it. It thus does not make sense to use the `balance_cov()` function. Instead, we want to examine whether our treatment caused attrition, and thus use the function `balance_attrite()`:

```
balance_attrite(data = test_sim,
        treatment = "Q5",
        question = "Q6")
```

```
##
## Call:
## glm(formula = question1 ~ treatment1, family = binomial(link = "logit"),
##     data = data2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.5813  -0.5813  -0.3140  -0.3140   2.4635
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -2.9852     0.2351 -12.696  < 2e-16 ***
## treatment1treatment   1.2926     0.2739   4.719 2.37e-06 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 511.72  on 780  degrees of freedom
## Residual deviance: 485.91  on 779  degrees of freedom
##   (219 observations deleted due to missingness)
## AIC: 489.91
##
## Number of Fisher Scoring iterations: 5
```

We learn that treated respondents are more likely to attrite, treatment is positively associated with attrition and is statistically significant.

Bounds

Next, we use the function `bounds()` to to get extreme value (Manski) bounds. This function calls the function `estimator_ev` from the `attrition` package by Alex Coppock. `treatment` is the assignment indicator (Z). `DV` is the outcome of interest (Y). Our `bounds()` function removes respondents who attrited pre-treatment and calculates R (the respose indicator variable) based on missingness on the DV (missing=0, response=1), based on the assumptions drawn by Manski.

The default for the bounds type is `type = "Manski"`, but we can also specify the type of bounds such that `type = "Lee"` to get Trimming (Lee) bounds. Since we cannot defy the monotonicity assumption, Lee bounds cannot be yielded here, however we demonstrate the use of `type = "Lee"` in the next section.

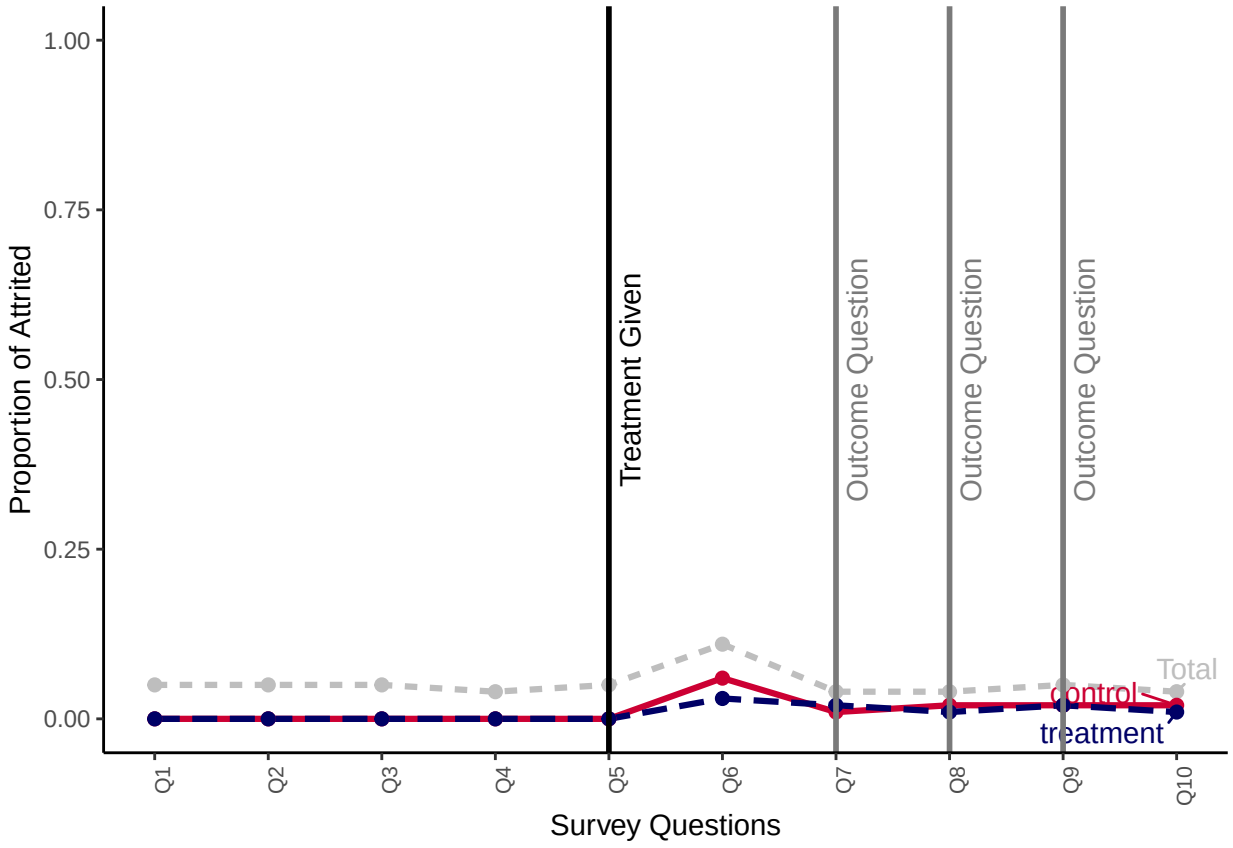
Control causes attrition

We repeat this process, but instead we look at a case where control causes attrition in `test_sim2`. We might see something like this if positive emotions (like happiness) are ramped up with treatment, making attrition less likely.

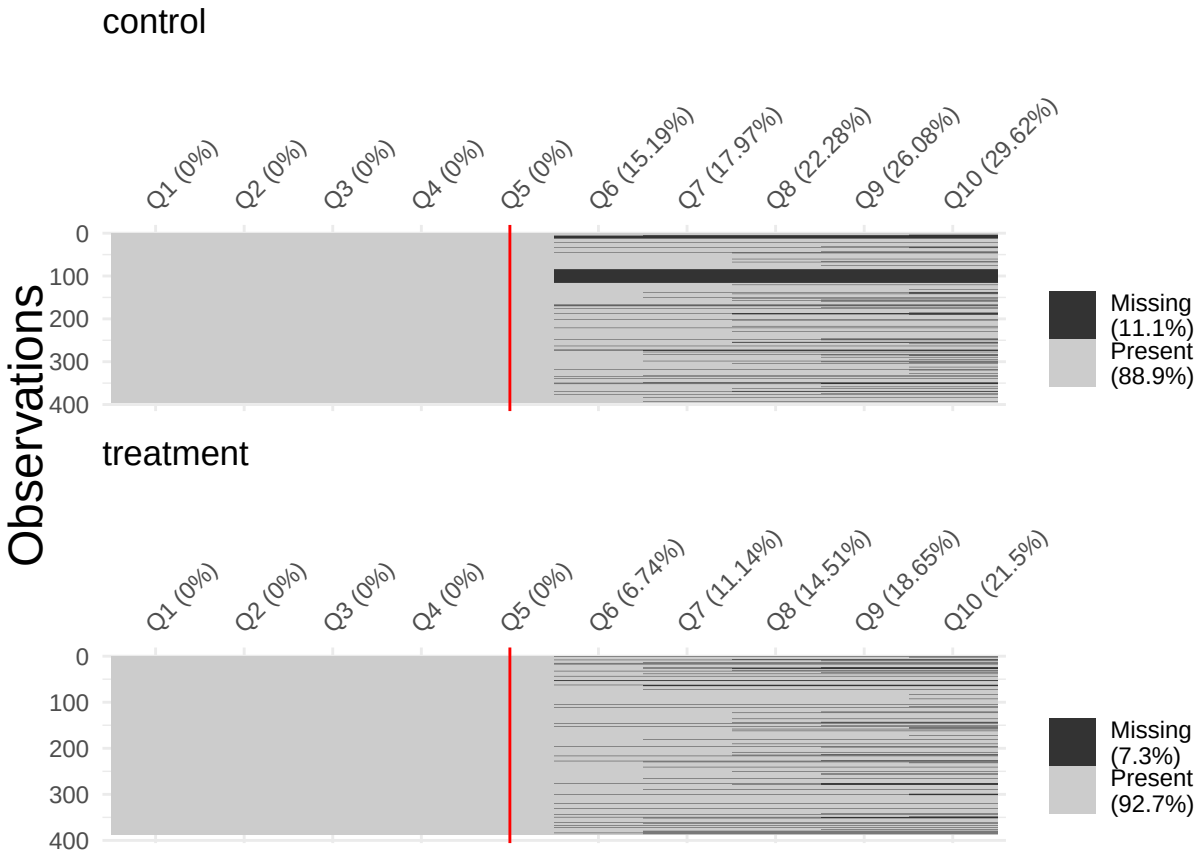
Plot

We visualize attrition using the `plot_attrition()` and `vis_miss_treat()` functions.

```
plot_attrition(test_sim2,
               treatment_q = "Q5",
               outcome_q = c("Q7", "Q8",  "Q9"),
               freq = FALSE,
               mycolors = c(treatment = "#000066",
                            control = "#CC0033")
               )
```




```
vis_miss_treat(test_sim2, treatment_q = "Q5")
```



We learn that attrition mostly occurs after Q6, and that treated respondents seem to be attriting more.

Balance

We learn that most respondents attrite at the post-treatment question `Q6`, and conduct a balance test. Note that `Q6` is an outcome, and we expect our treatment to affect it. It thus does not make sense to use the `balance_cov()` function. Instead, we want to examine whether our treatment caused attrition, and thus use the function `balance_attrite()`:

```
balance_attrite(data = test_sim2,
        treatment = "Q5",
        question = "Q6")
```

```
##
## Call:
## glm(formula = question1 ~ treatment1, family = binomial(link = "logit"),
##     data = data2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.5740  -0.5740  -0.3735  -0.3735   2.3228
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -1.7198     0.1402 -12.268  < 2e-16 ***
## treatment1treatment   -0.9082     0.2468  -3.681 0.000233 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 541.63  on 780  degrees of freedom
## Residual deviance: 527.02  on 779  degrees of freedom
##   (219 observations deleted due to missingness)
## AIC: 531.02
##
## Number of Fisher Scoring iterations: 5
```

We learn that treated respondents are more likely to attrite, treatment is positively associated with attrition and is statistically significant.

Bounds

Next, we use the function `bounds()` to to get extreme value (Manski) bounds and lee sharp bounds.

```
bounds(data = test_sim2,
       treatment = "Q5",
       DV = "Q8")
```

```
##    ci_lower    ci_upper     low_est     upp_est     low_var     upp_var
## -0.01143973  0.59474136  0.29165082  0.29165082  0.02391380  0.02391380
```

```
#lee sharp
bounds(data = test_sim2,
       treatment = "Q5",
       DV = "Q7",
       type = "Lee")
```

```
##     upper_bound     lower_bound       Out0_mono      Out1L_mono      Out1U_mono
##      0.23824824     -0.26437725      3.93209877      3.66772152      4.17034700
## control_group_N   treat_group_N               Q              f1              f0
##    324.00000000    343.00000000      0.07691626      0.11139896      0.17974684
##          pi_r_1          pi_r_0
##      0.88860104      0.82025316
```