Statistical Perceptions Serverless Survey

Contents

These documents outline how to use for researchers and for potential contributors.

Details that are for contributors denoted with the </> icon, when mixed into a page with other information it will be in a dropdown.

Overview

What it does

Serverless survey transforms a configuration file of questions and plot settings and generates a set of html files that comprise a survey that will, upon submission forward to another service, typically qualtrics.

Recommended Use

We recommend using our provided template repo and creating one repository per study or project. A study may have multiple surveys in it, but will all be hosted in the same place. Treat each repository as a separate folder, with no nesting.

In this case your overall workflow will include the following steps:

1. Use template (more details in template README)
2. Start your qualtrics study(ies) to get the URLs for forward from the generated questions
3. Set up your study in the configuration file following yaml guide, page settings, and question specific settings
4. Build the survey using actions in your study repo (more details in template README)
5. Use the generated instructions to set up your qualtrics survey to forward and recieve data using embedded data following our qualtrics instructions page
6. check at the end of the instructions page that your study does not need to send more than the maximum amount of data to qualtrics at once
7. Take your survey, entering plausible values to check that your data is showing up as expected in qualtrics (check all of the surveys).
8. Run your study!

Notes:

Skip to main content

- there is a limited number of characters that can be passed, but it is not well documented and varies browser to browser. We recommend no more than 2000 characters passing to qualtrics at each time. You can split into multiple surveys for longer studies.

You'll then have a set of data files to merge together to combine multiple sections and be able to analyze your data.

Use details

There is a CLI tool and it generates html files in a single folder. These files can then be hosted anywhere.

The CLI includes two main commands:

- `ssgeneratehtml` generates the html files and instructions md file.
- `sslengthcheck` checks the length of the forwarded message after a study has been generated. It uses the instructions file as input

Working with Configuration Files

A configuration file is yaml

YAML Format

There are two supported formats for the configuration file

No shared parameters

This file is setup like:

```
- question_id: unique_id_for_q1
  name_of_var1_for_q1: value_for_var1_q1
  name_of_var2_for_q1: value_for_var2_q1
  figure_values:
    name_of_fig_var1_for_q1: value_for_fig_var1_for_q1
- question_id: unique_id_for_q2
  name_of_var1_for_q2: value_for_var1_q2
  name_of_var3_for_q2: value_for_var3_q2
  name_of_var3_for_q2: |
        value_for_var3_q2_line_1
        value_for_var3_q2_line_3
        value_for_var3_q2_line_3
  figure_values:
    name_of_fig_var1_for_q2: value_for_fig_var1_for_q2
```

Notes:

- Each `name_of_varX_for_qY` has to be a variable that the `make_question_page` function accepts
- any variables not specified will get the default value as stated in the documentation
- `figure_values` is a special variable that takes more variables. the names of the fig variables are defined for each question
- the variables can be in any order
- `question_id` must be stated, there is no default value for it
- only the first variable for each question gets a `-`
- `name_of_var3_for_q2` is an example of how to format a long value if you do not leave it on a single line.

Some come from the question and others are for the pate

Skip to main content

Shared parameters

To share values across question it can be set up so that the top level is a single entry with two keys (`shared` and `unique`) where the `shared` key includes the parameter values that are to be applied to all questiona and `unique` includes a list defining individiual questions as above. Any values defined in both, the `unique` will overwrite the `shared` value.

For example:

```
shared:
  name_of_var1_shared: value_for_var1_shared
  name_of_var2_shared: value_for_var2_shared
  figure_values:
    name_of_fig_var2_shared: value_for_fig_var2_shared
unique:
- question_id: unique_id_for_q1
  name_of_var4_for_q1: value_for_var4_q1
  figure_values:
    name_of_fig_var1_for_q1: value_for_fig_var1_for_q1
- question_id: unique_id_for_q2
  name_of_var4_for_q2: value_for_var4_q2
  figure_values:
    name_of_fig_var1_for_q2: value_for_fig_var1_for_q2
```

This is equivalent to (but, for large number of questions, more compact than):

```
- question_id: unique_id_for_q1
  name_of_var4_for_q1: value_for_var4_q1
  name_of_var1_shared: value_for_var1_shared
  name_of_var2_shared: value_for_var2_shared
  figure_values:
    name_of_fig_var2_shared: value_for_fig_var2_shared
    name_of_fig_var1_for_q1: value_for_fig_var1_for_q1
- question_id: unique_id_for_q2
  name_of_var4_for_q2: value_for_var4_q2
  name_of_var1_shared: value_for_var1_shared
  name_of_var2_shared: value_for_var2_shared
  figure_values:
    name_of_fig_var2_shared: value_for_fig_var2_shared
    name_of_fig_var1_for_q2: value_for_fig_var1_for_q2
```

Configuring your study

To configure the study you will need the urls to each follow-up survey. They do not have to be fully configured first though.

Each question is a single page with a figure on it.

Page level Settings

These settings control the rest of the question page, other than the figure.

- `question_id` : string {required} name for the question internally
- `figure_type` : string string name of valid plot type in ssbuilder
- `figure_values` : dictionary parameters to pass to plotting function
- `page_title` : string what to show in the tab title default = 'Normal Curve Question',
- `question_text` : string the text of the questions
- `confirm_message` : text prompt for confirmation
- `skip_message` : text prompt for skipping
- `button_text` : string text on button
- `out_html_file` : string name fo the html file, that will be in the url for the participant if not passed will add ".html" to the questionid
- `out_rel_path` : string or file buffer where to write the files.
- `logging_vars` : dictionary dictionary of names for the variable types the specific question requires
- `confirm_var_name` : string {'confirm'} name for the variable, if not passed will be question_id + 'confirm' +question_id
- `var_name_suffix` : boolean {True} if true, add question_id to the passed values for all _var_names. Default is True, can be changed to False if you specify the variable names directly
- `pass_through_vars` : list of strings ['id'] list of variables to pass through from previous to next
- `next_question_url` : string question id or url for the qualtrics question
- `pretty_url` : boolean {False} if True make pages like `/IndentiCurve/name/` instead of `/IdentiCurve/name.html`
- `full_html` : boolean {True} generate a full html page or if False, generate only a segment of the page (eg for combining or embedding)

> **Warning**
>
> You cannot use `end` as a question ID, or `end.html` as an output file name

Question Text

Questions' text goes in the page level parameter `question_text`. It may include markdown formatting to be rendered.

Some key examples:

- `**bold**`
- `*italic*`
- `- bulleted text`
- `[link display text](url/for/link)`
- a blank space at the end of aline will make a new paragraph

Figure specific Settings

These settings vary by question type and the options are detailed on Question Types

Build Level Options

Serverless Survey has some settings that are for a whole study or about how to process the configuration file.

These are set as CLI arguments if you build offline. If you use our template repo, you will have a set of options for these controls in your actions tab.

This is where the path to save the output html files are set as well as the url of the hosted site for generating the instructions.

supplemental materials).

```
%%bash
ssgeneratehtml --help
```

```
Usage: ssgeneratehtml [OPTIONS]

  Generate html files from a configuration file

  Parameters ---------- config_file : string or None     file name, if none,
  configureation.yml assumed repo_name : string {None}     name of the repo
  out_url : string {None}     specify if not github with org/repo gh_org :
  string {None}     name of the gh org or user that owns the repo to build the
  URL debug : bool     print debuggin information or not out_rel_path : string
  relative path where to save the html files  fragment : bool     generate a
  fragment or not all_in_one : bool     merge files to a single htmlfile, this
  version will not work as as a survey

Options:
  -f, --config-file TEXT
  -p, --out_rel_path TEXT
  -r, --repo_name TEXT
  -o, --gh_org TEXT
  -d, --debug
  --fragment
  -a, --all_in_one
```

Skip to main content

```
-v, --study-pass-through-vars TEXT
```

```
--help                          Show this message and exit.
```

Question Types

Question Implementation

Each question type is implemented as a class the class also specifies the HTML/js templates to use for that question type. The constructor documents the logging variables that can be passed. See more on

All questions must have certain parameters:

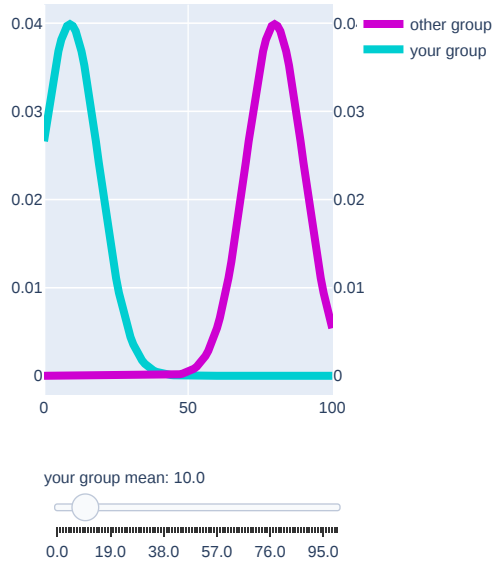
- `question_text`
- `question_id`

the `question_text` can include markdown formatting which will be rendered with the markdown package

Normal Curve Questions

This question has two normal curves, one moves and one does not.

Example



Note

All example figures on this page done with defaults which are vis on the developer page

Settings

Use `figure_type: NormalCurveSlider` with the following parameters in `figure_values`

Skip to main content

- `static_name` : string legend text for static curve
- `static_color` : hex including # hex code for the color to use for static curve, including a # sign as the first character
- `static_mean` : number location of the static curve
- `static_curve_width` : number width of curve, as the scipy.norm scale
- `dynamic_name` : string legend text for dynamic curve
- `dynamic_color` : hex including # hex code for the color to use for dynamic curve, including a # sign as the first character
- `dynamic_starting_mean` : number the location where the slider starts
- `curve_width` : number width of curves
- `num_slider_locs` : integer number of slilder locations
- `min_slider_value` : number the minimum value for the slider
- `max_slider_value` : number the maximum value for the slider
- `overlap_decimals` : integer number of place values to round the % overlap value to for both display and reporting, positive to the right of the decimal, negative for left of decimal (eg -2 rounds to nearest 100)
- `mean_decimals` : integer number of place values to round the mean (position) value to for both display and reporting positive to the right of the decimal, negative for left of decimal (eg -2 rounds to nearest 100)
- `xaxis_title` : string text label for the x axis

Tradeoff Questions

this question type trades off between two two extremes over a number of models in the middle

Both formats accept the same dataset files, where each row represents one bar heigh for one slider location. The values should be compatible with deisplay in the figures, for example converting .74 to 75 to display percentages. There may be extra columns that are not used.

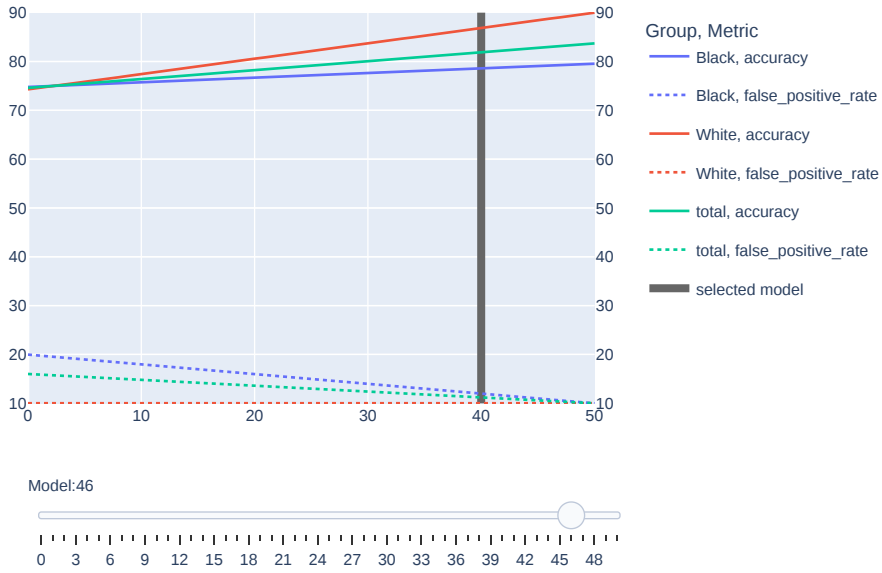
An example (that is used in the example plots):

	alpha	variable	value	metric	group	model_number	percent
0	0.00	acc_0	0.747873	accuracy	Black	0	74.787268
1	0.02	acc_0	0.748818	accuracy	Black	1	74.881815
2	0.04	acc_0	0.749764	accuracy	Black	2	74.976363
3	0.06	acc_0	0.750709	accuracy	Black	3	75.070911
4	0.08	acc_0	0.751655	accuracy	Black	4	75.165459

Line Graph Tradeoff

This uses lines to show all metrics on one set of axes.

Example



Settings

Use `figure_type: TradeoffLine` with the following parameters for use in `figure_values`:

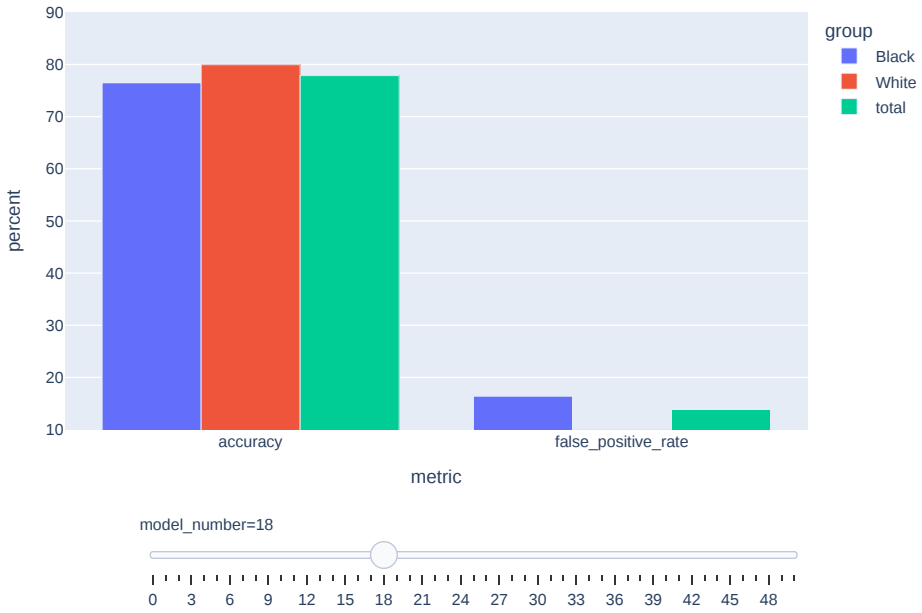
- `pretty_data_file` : string file name of a tidy (tall) dataset with pretty content. that is any data transformations should occur on the data (eg scaling .7523943 to 75.23943 and expanding column names) column names can still rely on python conventions, before display the `_` will be converted to space
- `slider_column` : string name of column to use for the slider
- `slider_label` : string name to display when labeling the slider postion values (and in hovertext)
- `x_col,y_col` : string name of column to use for the xor y axis
- `trace_value1,trace_value2` : same as the values of `x_col` in the data file first,second value to filter (left, right metric)
- `trace1_hover,` trace2_hover : string noun versions to use in the hovertext
- `y_min,y_max` : numerical minimum and maximum values to fix the plot axies, if none, allow plotly to decide
- `num_digits` : num digits to display color_col: string name of colum to use for the colring of the lines
- `color_hover` : string noun to use for groups
- `disable_zoom` : bool disable the zoom on the generated plot
- `anchor_name` : string name for vertical bar default_selection :int model that is selected when laoding

Bar Graph Tradeoff

This uses a set of bar graphs.

Example

Skip to main content



Settings

Use `figure_type: TradeoffBar` with the following parameters for use in `figure_values`

- `pretty_data_file` : string file name of a tidy (tall) dataset with pretty content. that is any data transformations should occur on the data (eg scaling .7523943 to 75.23943 and expanding column names) column names can still rely on python conventions, before display the `_` will be converted to space
- `slider_column` : string name of column to use for the slider
- `slider_label` : string name to display when labeling the slider postion values (and in hovertext)
- `x_col,y_col` : string name of column to use for the xor y axis
- `x_value1,x_value2` : same as the values of `x_col` in the data file first,second value to filter (left, right metric)
- `x_value1_hover,` x_value2_hover : string noun versions to use in the hovertext
- `y_min,y_max` : numerical minimum and maximum values to fix the plot axies, if none, allow plotly to decide
- `num_digits` : num digits to display color_col: string name of colum to use for the colring of the bars color_hover: hover text to use for groups created by color
- `disable_zoom` : bool disable the zoom on the generated plot default_selection :int model that is selected when laoding

Coordinating with Qualtrics

> **Important**
>
> This is the base documentation that may need updates

Basic use

1. the first qualtrics survey sends embedded data via forwarding only
2. middle ones receive data and send data
3. the last one receives data only

Skip to main content

Sending the ID from Qualtrics

1. Add an embedded data block with the identifier to forward (eg panel ID or Response ID)
2. redirect end of survey to a url
3. embed the response id in the forwarding url:

Template

```
https://statistical-perceptions.github.io/IdentiCurve/<question_out_html_file>.html?id=<qualtrics piped t
```

Example

```
https://statistical-perceptions.github.io/sample-nobackend/?id=${e://Field/ResponseID}
```

in this case, I only had one question page so there is no question_id set and I used the ResponseID feild.

note:

- if needed, we can pass more than a single unique identifier on, but that requires code changes

Recieving Data into Qualtrics

1. set up embedded data as the first block on the workflows tab. set the variables as per the instructions output
2. (if applicable) use piped text to refer to those values in the question text
3. (optional) add a branch after the embedded data to have people skip the survey “if id is Equal to demo”

- getting data from url

Branching to different ss questions based on qualtrics answers

1. on the survey flow tab add a branch
2. set the condition to be based on a question
3. add the url to forward to
4. (if applicable) add the variable value to the forward url

Template

```
https://statistical-perceptions.github.io/IdentiCurve/<question_out_html_file>.html?id=<qualtrics piped t
```

Example

```
https://statistical-perceptions.github.io/sample-nobackend/?id=${e://Field/ResponseID}&group=w
```

in this case, I only had one question page so there is no question_id set and I used the ResponseID feild.

Branching to different ss questions based on embedded data

1. make sure the embedded data received includes the pass through variable
2. on the survey flow tab add a branch
3. set the condition to be based on embedded data and choose the values

Skip to main content

5. (if applicable) add the variable value to the forward url

Template

```
https://org-or-user.github.io/repo/question_out_html_file.html?id=<qualtrics piped text>&var=value
```

Example

```
https://statistical-perceptions.github.io/IdentiCurve/nc2t1w.html?id=${e://Field/ResponseID}&group=${e://
```

Semi-automatic forwarding to different qustions

In order to have a survey where there are different versions of the questions for different participants depending on a response of a question, set up a question in qualtrics where the response will be the “logic variable.” Then make it so that the logic variable is passed along with the ID and its values appear in the url so that the variable in piped text can make it work.

1. set up configurations so that the logic variable values are in the question ids (or html file names)
2. make sure embedded data received includes the variable used for logic
3. set up the url like (for `group` as the logic variable)

```
https://ghorg.github.io/repo/page_url${e://Field/logicvariable}.html?id=${e://Field/ResponseID}&group=${e
```

For example if the following is in qualtrics as the forward url

```
https://statistical-perceptions.github.io/IdentiCurve/nc2t1${e://Field/group}.html?id=${e://Field/Respons
```

when `group=a` the url will become

```
https://statistical-perceptions.github.io/IdentiCurve/nc2t1a.html?id=RH2904&group=a
```

Qualtrics Help

- Passing with Query Strings: about the url
- Piped text: about getting and formatting variables for the url or for using responses to modify qualtrics questions
- embedded data: for storing data to send and receive it

Question Types

Question Implementation

Each question type is implemented as a class the class also specifies the HTML/js templates to use for that question type. The constructor documents the logging variables that can be passed.

Normal Curve Questions

This question has two normal curves, one moves and one does not.

```
class ssbuilder.NormalCurveSlider(logging_vars={'location_var_name': 'loc',
'overlap_var_name': 'ov'})
```

Skip to main content

generate_figure(*static_name='other group', static_color='#CE00D1', static_mean=80, static_curve_width=10, dynamic_name='your group', dynamic_color='#00CED1', dynamic_starting_mean=10, dynamic_curve_width=10, num_slider_locs=101, min_slider_value=None, max_slider_value=None, overlap_decimals=2, mean_decimals=None, xaxis_title=''*)

Generate a normal curve question object on a default scale of

Parameters:

- **static_name** (*string*) – legend text for static curve
- **static_color** (*hex including #*) – hex code for the color to use for static curve, including a # sign as the first character
- **static_mean** (*number*) – location of the static curve
- **static_curve_width** (*number*) – width of curve, as the scipy.norm scale
- **dynamic_name** (*string*) – legend text for dynamic curve
- **dynamic_color** (*hex including #*) – hex code for the color to use for dynamic curve, including a # sign as the first character
- **dynamic_starting_mean** (*number*) – the location where the slider starts
- **curve_width** (*number*) – width of curves
- **num_slider_locs** (*integer*) – number of slilder locations
- **min_slider_value** (*number*) – the minimum value for the slider
- **max_slider_value** (*number*) – the maximum value for the slider
- **overlap_decimals** (*integer*) – number of place values to round the % overlap value to for both display and reporting, positive to the right of the decimal, negative for left of decimal (eg -2 rounds to nearest 100)
- **mean_decimals** (*integer*) – number of place values to round the mean (position) value to for both display and reporting positive to the right of the decimal, negative for left of decimal (eg -2 rounds to nearest 100)
- **xaxis_title** (*string*) – text label for the x axis

Returns:

fig – figure object based on parameters

Return type:

plotly figure object

curve is drawn with scipy.norm

Trade Off Questions

this quesiton typ trades off between two two extremes over a number of models in the middle

class ssbuilder.**TradeoffLine**(*logging_vars={'location_var_name': 'model_number'}*)

generate_figure(*pretty_data_file, slider_label='Model', trace_col='metric', x_col='model_number', trace_value1='accuracy', trace1_hover='accurate', trace_value2='false_positive_rate', trace2_hover='false positives', y_col='percent', y_min=None, y_max=None, num_digits=2, color_col='group', color_hover='people', anchor_name='selected model', disable_zoom=True, default_selection=10*)

make the lineplot

Parameters:

- **pretty_data_file** (*string*) – file name of a tidy (tall) dataset with pretty content. that is any data transformations should occur on the data (eg scaling .7523943 to 75.23943 and expanding column names) column names can still rely on python conventions, before display the _ will be converted to space
- **slider_column** (*string*) – name of column to use for the slider

'ertext)

- **x_col** (*string*) – name of column to use for the xor y axis
- **y_col** (*string*) – name of column to use for the xor y axis
- **trace_value1** (same as the values of *x_col* in the data file) – first,second value to filter (left, right metric)
- **trace_value2** (same as the values of *x_col* in the data file) – first,second value to filter (left, right metric)
- **trace1_hover** (*string*) – noun versions to use in the hovertext
- **trace2_hover** (*string*) – noun versions to use in the hovertext
- **y_min** (*numerical*) – minimum and maximum values to fix the plot axies, if none, allow plotly to decide
- **y_max** (*numerical*) – minimum and maximum values to fix the plot axies, if none, allow plotly to decide
- **num_digits** (*num digits to display*) –
- **color_col** (*string*) – name of colum to use for the colring of the lines
- **color_hover** (*string*) – noun to use for groups
- **disable_zoom** (*bool*) – disable the zoom on the generated plot
- **anchor_name** (*string*) – name for vertical bar
- **default_selection** (*int*) – model that is selected when laoding

Returns:

- **fig** (*plotly figure object*)
- *figure object based on parameters*

class ssbuilder.**TradeoffBar**(*logging_vars={'location_var_name': 'model_number'}*)

generate_figure(*pretty_data_file, slider_column='model_number', slider_label='Model', x_col='metric', x_value1='accuracy', x_value1_hover='accurate', x_value2='false_positive_rate', x_value2_hover='false positives', y_col='percent', y_min=None, y_max=None, num_digits=1, color_col='group', color_hover='people', disable_zoom=True, default_selection=10*)

make the barplot

Parameters:

- **pretty_data_file** (*string*) – file name of a tidy (tall) dataset with pretty content. that is any data transformations should occur on the data (eg scaling .7523943 to 75.23943 and expanding column names) column names can still rely on python conventions, before display the _ will be converted to space
- **slider_column** (*string*) – name of column to use for the slider
- **slider_label** (*string*) – name to display when labeling the slider postion values (and in hovertext)
- **x_col** (*string*) – name of column to use for the xor y axis
- **y_col** (*string*) – name of column to use for the xor y axis
- **x_value1** (same as the values of *x_col* in the data file) – first,second value to filter (left, right metric)
- **x_value2** (same as the values of *x_col* in the data file) – first,second value to filter (left, right metric)
- **x_value1_hover** (*string*) – noun versions to use in the hovertext
- **x_value2_hover** (*string*) – noun versions to use in the hovertext
- **y_min** (*numerical*) – minimum and maximum values to fix the plot axies, if none, allow plotly to decide
- **y_max** (*numerical*) – minimum and maximum values to fix the plot axies, if none, allow plotly to decide
- **num_digits** (*num digits to display*) –
- **color_col** (*string*) – name of colum to use for the colring of the bars
- **color_hover** – hover text to use for groups created by color
- **disable_zoom** (*bool*) – disable the zoom on the generated plot
- **default_selection** (*int*) – model that is selected when laoding

Returns:

- **fig** (*plotly figure object*)
- *figure object based on parameters*

Skip to main content

Configuration Details

Page level

class ssbuilder.builder.**make_question_page**(*question_id, figure_type='NormalCurveSlider', figure_values=None, page_title='Normal Curve Question', question_text='Move the slider', confirm_message='Confirm my answer', skip_message='Prefer not to answer', button_text='Submit', out_html_file=None, out_rel_path=None, logging_vars=None, confirm_var_name=None, var_name_suffix=True, pretty_url=False, pass_through_vars=['id'], out_url=None, next_question_url=None, debug=False, full_html=True*)

generate html file

Parameters:

- **question_id** (*string {required}*) – name for the question internally
- **figure_type** (*string*) – string name of valid plot type in ssbuilder
- **figure_values** (*dictionary*) – parameters to pass to plotting function
- **page_title** (*string*) – what to show in the tab title default = 'Normal Curve Question',
- **question_text** (*string*) – the text of the questions
- **confirm_message** (*text*) – prompt for confirmation
- **skip_message** (*text*) – prompt for skipping
- **button_text** (*string*) – text on button
- **out_html_file** (*string*) – name fo the html file, that will be in the url for the participant if not passed will add ".html" to the questionid
- **out_rel_path** (*string or file buffer*) – where to write the files.
- **logging_vars** (*dictionary*) – dictionary of names for the variable types the specific question requires
- **confirm_var_name** (*string {'confirm'}*) – name for the variable, if not passed will be question_id + 'confirm' +question_id
- **var_name_suffix** (*boolean {True}*) – if true, add question_id to the passed values for all _var_names. Default is True, can be changed to False if you specify the variable names directly
- **pass_through_vars** (*list of strings ['id']*) – list of variables to pass through from previous to next
- **next_question_url** (*string*) – question id or url for the qualtrics question
- **pretty_url** (*boolean {False}*) – if True make pages like */IndentiCurve/name/* instead of */IdentiCurve/name.html*
- **full_html** (*boolean {True}*) – generate a full html page or if False, generate only a segment of the page (eg for combining or embedding)

Notes

variables with _var_name + "id" will be passed to qualtrics

Building from config

ssbuilder.**generate_from_configuration**

alias of <Command generate-from-configuration>