Bracket Continuous Values

Scripting Solutions

Additional scripting solutions will be added in the future. Please reach out to Alchemer with comments and suggestions on solutions you'd like to see via the link here.

This script is not currently functioning as expected, we do not recommend using this script in your survey.

Goal

Automatically convert a continuous value, like age 25, into a bracketed range, like 18-34. Bracketed ranges will then be accessible in a Standard Report to segment or view as a pie chart.

Effort: ✔ ✔ ✔

Solution

When the page is submitted the solution sets a hidden Radio Button with options for the bracket ranges based on a value entered by the user or passed as a URL variable.

Step 1: Get the continuous value

Option 1: From a Textbox Question

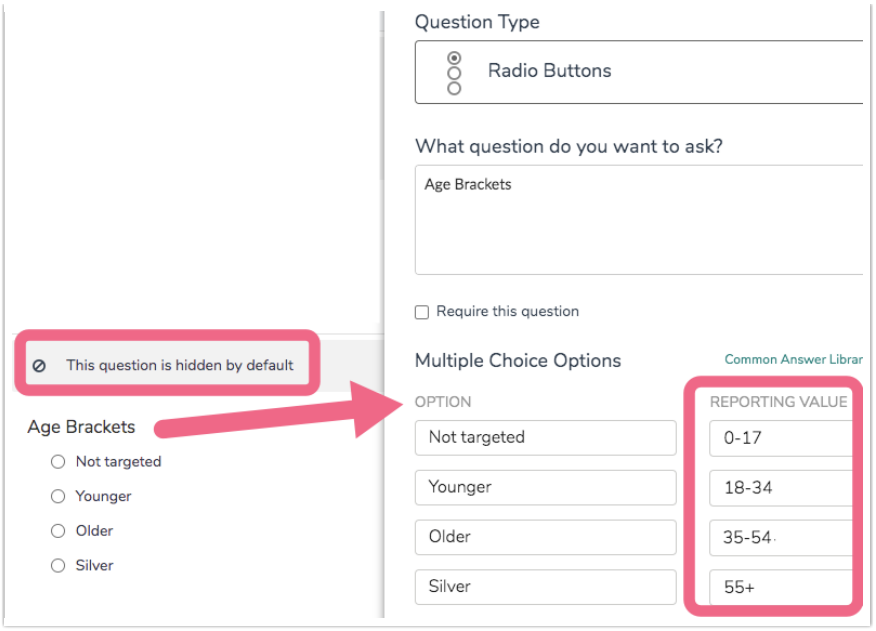
Add a Textbox Question Type. Set the Validation Answer Format to Number, Force Whole Number, Force Positive Numbers. Respondent enters a number.

Option 2: From a URL Variable

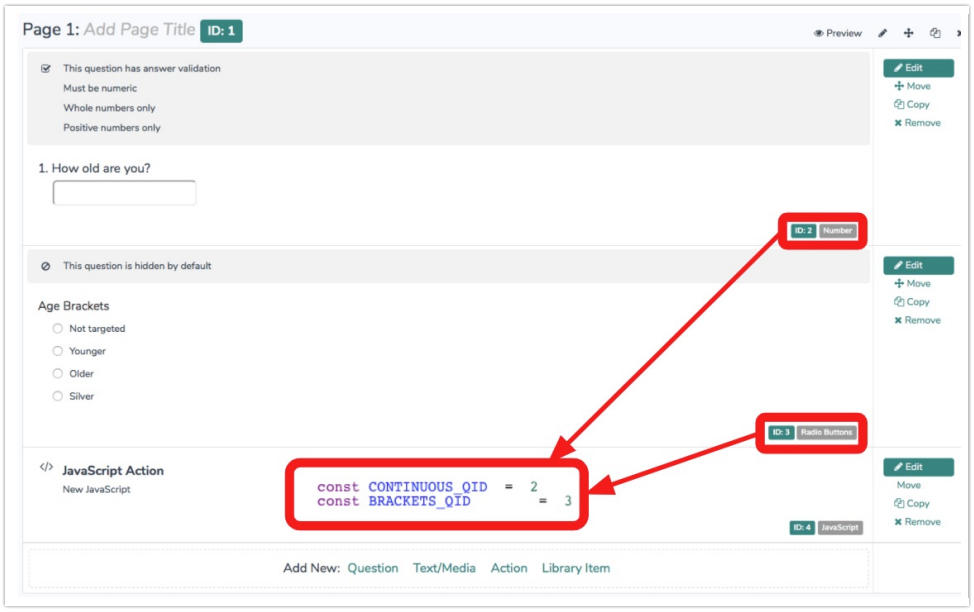
Add a Hidden Value Action and set the "*Populate with the following*" field to a merge code for a URL variable, such as: [url("age")].

Step 2: Add a hidden radio button question with bracket values

1. Add a Radio Button question just below the Textbox Question.
2. Set the Answer Option Reporting Values to define the brackets. Reporting values must define all ranges from 0 to infinity, starting with '0' and ending with '+'
3. Hide the question from the respondent by selecting: **Logic > Hide this question...** .



A view of the entire build:



Step 3: Add the Javascript Action

1. Paste the code below into a Javascript Action on the same page.
2. Replace the yellow highlighted IDs with the Question IDs of the elements added above.

```
/* Alchemer v01

  Bracket a continuous numeric value from a textbox or URL variable into a radio button question.

  Documentation and updates: https://help.alchemer.com/help/age-brackets-buckets-javascriptlua

*/

document.addEventListener("DOMContentLoaded", function() {
```

```
document.addEventListener("DOMContentLoaded", function() {

  const CONTINUOUS_QID  = 21  // the textbox or hidden value qid for the continuous value
  const BRACKETS_QID    = 22  // the radio button question of bracket ranges

  // ******************************
  // *** no changes needed below ***
  // ******************************

  const LOG = true

  /***
   * Test boolean value, alert() and throw Error if it's false
   *
   * bool {t/f} value to test
   * msg {string} message to alert and throw in new Error
   */
  const assert = (bool, msg) => {
    msg = "Javascript Assert Error: " + msg
    if (!bool) {
      alert(msg)
      console.error(msg)
      const err = new Error(msg)
      console.error(err)
      throw err
      }
  }

  /***
   * Get an element based on its Question ID
   *
   * qid {int/string} question ID
   * section = "element" {string} the final section of the element id
   * return {element} looks for id's in the form: "sgE-1234567-12-123-element"
   */
  const getElemByQid = (qid, section = "element") => {
    const id = "sgE-" + SGAPI.survey.surveyObject.id + "-" + SGAPI.survey.pageId + "-" + qid + "-" + section
    const elem = document.getElementById(id)
    assert(elem, "Javascript: can't find element with id = " + id + ", section = " + section)
    return elem
  }

  /***
   * Set the selections of a radio button or checkbox question
   *
   * qid {int/string} the question ID of a radio button or checkbox question to clear
   * checkOptionsIds {int or array of int} a single OptionID or array of OptionIDs to be checked
   */
  const setCheckedByQid = (qid, _checkOptionIds) => {

    // convert param to an array if it was a single value
    let checkOptionIds = Array.isArray(_checkOptionIds) ? _checkOptionIds : [_checkOptionIds]

    // ensure array is all integers
    checkOptionIds = checkOptionIds.map(id => parseInt(id))

    // go through all options and check or uncheck them
    getElemByQid(qid, "box").querySelectorAll('.sg-question-options input')
      .forEach(inputElem =>
        inputElem.checked = checkOptionIds.includes(parseInt(inputElem.value)))
  }

  /***
   * Get the option ids and reporting values for a radio button or
   * checkbox qid from the SGAPI object
   *
```

```
 * qid {int / string} question ID
 * return {array of obj} array of option objects:
 *             [
 *               { optionId: "10014", reportingValue: "4" },
 *               { optionId: "10015", reportingValue: "3" }
 *             ]
 */
const getOptionReportingValues = (qid) => {
  assert(SGAPI.survey.surveyObject.questions[qid], "Can't find qid on this page: " + qid)

  const optionsObj = SGAPI.survey.surveyObject.questions[qid].options
  assert(optionsObj, "QID isn't a radio button or checkbox question: " + qid)

  return Object.keys(optionsObj).map(optionId =>
    ({ optionId: optionId, reportingValue: optionsObj[optionId].value }) )
}

/***
 * Load brackets and ensure they represent contiguous ranges 0-MAX_SAFE_INTEGER
 *
 * bracketQid {int/string} question ID of the bracket Radio Button question
 * return {array of obj} array of enhanced options for bracketQid sorted by min value:
 *         [
 *           { optionId: "10014", reportingValue: "0-17",  min:  0, max: 17 },
 *           { optionId: "10014", reportingValue: "18-54", min: 18, max: 54 },
 *           { optionId: "10015", reportingValue: "55+",   min: 55, max: MAX_SAFE_INTEGER }
 *         ]
 */
const getBrackets = (bracketQid) => {

  /***
   * Checks if ranges are contiguous from 0-MAX_SAFE_INTEGER
   *
   * return {t/f}
   */
  const rangesAreContiguous = (optionObjs_withRanges) => {
    let currMax = -1;
    optionObjs_withRanges.forEach(obj => {
      if (obj.min !== (currMax + 1))
        return false
      currMax = obj.max
    })
    return currMax === Number.MAX_SAFE_INTEGER
  }

  /***
   * Clear the selections of a radio button or checkbox question
   *
   * qid {int/string} the question ID of a radio button or checkbox question to clear
   */
  const clearCheckedByQid = (qid) =>
    getElemByQid(qid, "box").querySelectorAll('.sg-question-options input')
      .forEach(inputElem => inputElem.checked = false)

  /***
   * main()
   */
  const optionObjs_sorted = getOptionReportingValues(bracketQid)
                      .sort((a, b) => a.reportingValue.localeCompare(b.reportingValue))

  const optionObjs_withRanges = optionObjs_sorted.map(optionObj => {
    // 55+
    if (optionObj.reportingValue.slice(-1) === '+') {
      const aParsed = optionObj.reportingValue.match(/^(\d+)\+$/)
      assert(aParsed && aParsed.length === 2, "Unable to parse bracket reporting value: " + optionObj.reportingValue)
```

```
      assert(aParsed && aParsed.length === 2, "Unable to parse bracket reporting value: ", optionObj.reportingValue)
      optionObj.min = parseInt(optionObj.reportingValue)
      optionObj.max = Number.MAX_SAFE_INTEGER
    // 25-34
    } else {
      const aParsed = optionObj.reportingValue.match(/^(\d+)-(\d+)$/)
      assert(aParsed && aParsed.length === 3, "Unable to parse bracket reporting value: ", optionObj.reportingValue)
      optionObj.min = parseInt(aParsed[1])
      optionObj.max = parseInt(aParsed[2])
    }
    return optionObj
  })

  assert(rangesAreContiguous(optionObjs_withRanges),"Bracket ranges are not contiguous, don't start with 0, or do
n't end with value followed by '+' for qid: " + bracketQid)

  return optionObjs_withRanges
}

/***
 * ()
 */
const getBracketOptionId = (brackets, continuousValue) => {
  for (let i = 0; i < brackets.length; i++) {
    if (brackets[i].min <= continuousValue && continuousValue <= brackets[i].max)
      return brackets[i].optionId
  }
  assert(false, "Logic error, the bracket should have been found for: " + continuousValue + "in: " + JSON.stringify(brac
kets))
}

/***
 * main()
 */

document.forms[0].addEventListener("submit", function() {

  const continuousValue = parseInt(getElemByQid(CONTINUOUS_QID).value)

  // set the bracket radio button question
  if (!isNaN(continuousValue)) {

    const brackets = getBrackets(BRACKETS_QID)
    if (LOG) console.log("brackets = ", JSON.stringify(getBrackets(BRACKETS_QID)))

    const optionId = getBracketOptionId(brackets, continuousValue)
    if (LOG) console.log("optionId = ", optionId)

    setCheckedByQid(BRACKETS_QID, optionId)

  // clear the bracket radio button question
  } else {
    setCheckedByQid(BRACKETS_QID, [])
  }

  return true
})
})
```

Related Articles