Ego-Network Surveys with LimeSurvey

Purpose

The following documents some experiences I made using LimeSurvey for an ego-network survey. Some features required considerable changes in LimeSurvey's source code which you also need to perform with your own installation of LimeSurvey.
This survey template is also available from https://www.limesurvey.org/downloads/category/13-surveys. Furthermore, I published the survey structure as .lss file and describe it here.

Structure of Network Survey

The personal network study as implemented in the attached .lss file consists of four parts:

1. Name generator questions
2. Name interpreter questions
3. Eliciting alteri-alteri relations
4. Ego

Features

- Name generator input fields show up one after another. This may mitigate heaping effects that occur when a certain number of input fields is visible.
- For name generators it is checked whether the same name was entered in another NG field (also across name generators)
- For name generators it is also checked whether respondent typed a "," or ";" which often indicates the misunderstanding of entering several persons in one text field.
- Alteri elicited are sampled in order to apply name interpreters to reduce respondent's burden. The number of sampled alteri can be specified.
- Once the respondent passes the name generator section she is no longer able to edit, delete, or add responses to the NG. An according hint is given.
- Name interpreter and alteri-alteri relation questions are only displayed for sampled alteri. I.e. if a respondent enters only five names NI and alteri-alteri relations are only displayed for these five persons.
- Slider enables an analogous scale for closeness (bar initially hidden)
- Translations in English and German (except ego related questions)

Adding languages

- When you add a new language do not forget to copy the equations to the new language! List of equations (ordered by occurance):
    - ng_complete
    - num_sampled

- ng_1
- ng_2
- ng_3
- ng_4
- ng_5
- ng_6
- ng_7
- ng_8
- ng_array
- ng_complete_set2
- ng_complete_set3

Incorporating additional code

As mentioned above the survey's logic requires some additional functions to be added to LS code:

- Validation function
- The proper display of name generator entry fields requires the adaptation of template.css. It ensures that the gap between entry fields is minimizes and prevents the display of unnecessary borders.

Line numbers refer to LimeSurvey Version 2.00+ Build 121116!

Line	Content	Description
limesurvey\application\helpers\expressions\em_core_helper.php		
232	'checkNG' => array('exprmgr_checkNG', 'LEMcheckNG', $this->gT('Checks NG input fields'), 'boolean checkNG(pos, arg1, ..., argN)', '', -3),	Define LS function that checks for display of NG fields
232	'randNG' => array('exprmgr_randNG', 'LEMrandNG', $this->gT('Returns a random ng value'), 'int randNG(arg1, ..., argN)', '', -1),	Define LS function for sampling alteri
232	'uniqueToFirst' => array('exprmgr_uniqueToFirst', 'LEMuniqueToFirst', $this->gT('Checkes whether there is another value in the list like the [position]th'), 'boolean uniqueToFirst(group, position, arg1, ..., argN)', '', -1),	Define LS function that checks
232	'sliderOnClick' => array('exprmgr_sliderOnClick', 'LEMsliderOnClick', $this->gT('Displays slider handle'), 'boolean sliderOnClick(q-code)', '', -1),	Define LS function for showing slider bar

585-590	// SH: only check for group above NG groups:
	if ($this->groupSeq >5) {
	if (($groupSeq != -1 && $this->groupSeq != -1) && ($groupSeq > $this->groupSeq))
	{
		Patch that prevents LS from causing an error because some vars are only defined

	$this->RDP_AddError($this->gT("Variable not declared until a later page"),$token); return false; }	after the current NG group.

3531	/**
	* SH:
	* Check whether either ...
	* ...the entry before i is not empty or
	* ...any value after is not empty
	* @param type $args
	*/
	function exprmgr_checkNG($args)
	{
	// error handling:
	if (sizeof($args) < 2) {
	echo $this->gT('At least two arguments need to be provided!');
	return false;
	}
	
	$pos = array_shift($args);
	
	if ($pos < 2) {
	echo $this->gT('Position may not be smaller than 2!');
	return false;
	}
	
	if (sizeof($args) < $pos) {
	echo $this->gT('Position may not be greater than length of items!');
	return false;
	}
	
	if (($args[$pos - 2] != '') && ($args[$pos - 2] != NULL)) {
	//var_dump($args[$pos - 2]);
	return true;
	}
	
	foreach (range($pos, sizeof($args)) as $i)
	{
		Implementing checkNG($args)

	//var_dump($args[$i - 1]);
	if (($args[$i - 1] != '') && ($args[$i - 1] != NULL)) {
	return true;
	}
	}
	return false;
	}	

3531	/**
	* SH:
	* 1st param: number n of names to sample
	* 2nd-nth. : sampled names vars
	* (n+1)th-x: name vars
	* @param type $args
	*/
	function exprmgr_randNG($args)
	{
	$current = array_shift($args);
	$n = array_shift($args);
	
	// array of already sampled names:
	$sampled = array();
	
	// remaining non-empty names:
	$names = array();
	
	// extract first arguments as already sampled names
	foreach (range(0, $current - 2) as $i) {
	$sampled[$i] = array_shift($args);
	}
	
	foreach (range($current, $n) as $j) {
	array_shift($args);
	}
	
	// copy non-empty vars and not already sampled names:
	$counter = -1;
	foreach ($args as $arg) {
	// TODO
	if (($arg != NULL) and ($arg != '') and (!in_array($arg, $sampled))) {
		Implementing randNG($args)

	$counter = $counter + 1;
	$names[$counter] = $arg;
	}
	}
	// return rand(0, $counter);//var_dump($names);
	if ($counter == -1) {
	return '';
	}
	
	return $names[rand(0, $counter)];	

3531	/**
	* SH:
	* Returns true if all non-empty values are different from the ith entry
	* @param type $args
	*/
	function exprmgr_uniqueToFirst($args)
	{
	$group = array_shift($args);
	$i = array_shift($args) + ($group - 1) * 25 ;
	
	$current = $args[intval($i) - 1];
	
	if (trim($current)=='')
	{
	return true;
	}
	
	if (strpos($current,',') or strpos($current,';'))
	{
	return false;
	}
	
	unset($args[intval($i) - 1]);
	
	foreach ($args as $arg)
	{
	if (trim($arg)=='')
	{
	continue; // ignore blank answers
	}
		Implementing uniqueToFirst($args)

	if ($arg == $current)
	{
	return false;
	}
	}
	return true;
	}	

3531	/**
	* SH:
	* Since the slider function is only relevant in JavaScript
	* it retruns true always.
	* @param type $args
	*/
	function exprmgr_sliderOnClick($args)
	{
	return true;
	}
		Implementing sliderOnClick($args)

limesurvey\scripts\expressions\em_javascript.js		

2630	function LEMcheckNG() {
	
	window.scrollTo(0, document.body.scrollHeight);
	
	//disable if NGs have been completed:
	if (checkDefined('ng_complete') &&
	LEMval('ng_complete')===1) {
	inputTextFields =
	document.getElementsByClassName("text ");
	for (var i = 0; i < inputTextFields.length; i++)
	{
	inputTextFields[i].style.display='none';
	//inputTextFields[i].disabled = true;
	}
	}
	
	var pos = arguments[0];
	// error handling:
	if (arguments.length < 2) {
	// 'At least two arguments need to be provided!';
	return false;
	}
	
	if (pos < 2) {
		Implementing checkNG()

	// 'Position may not be smaller than 2!';
	return false;
	}
	
	if (arguments.length < pos) {
	// 'Position may not be greater than length of items!'
	return false;
	}
	
	if ((arguments[pos - 1] != '') && (typeof
	arguments[pos - 1] !== "undefined")) {
	return true;
	}
	
	for (var i = pos; i < arguments.length; i++) {
	if ((arguments[i] != '') && (typeof arguments[i] !==
	"undefined")) {
	return true;
	}
	}
	return false;
	}	

2630	function checkDefined() {
	
	var arg = arguments[0];
	if (typeof LEMalias2varName[arg] === 'undefined') {
	return false;
	}
	jsName = LEMalias2varName[arg];
	if (typeof LEMvarNameAttr[jsName] === 'undefined') {
	return false;
	}
	return true;
	}
		Helper function for LEMcheckNG()

2630	function LEMrandNG() {
	
	var i;
	
	// array of already sampled names:
	var sampled = new Array();
	
	// remaining non-empty names:	

	var names = new Array();
	
	var current = arguments[0];
	var numSampled = arguments[1];
	
	// extract first arguments till current as already sampled names
	for (i = 0; i < current - 1; i++) {
	sampled[i] = arguments[i + 2];
	}
	
	// copy non-empty vars and not already sampled names:
	var counter = -1;
	
	for (i = numSampled + 2; i < arguments.length; i++) {
	if ((typeof arguments[i] !== "undefined") && (arguments[i] != '') && (!contains(sampled, arguments[i]))) {
	counter++;
	names[counter] = arguments[i];
	}
	}
	if (counter == -1) {
	return '';
	}
	return names[Math.floor(Math.random() * (counter + 1))];
	}	

2630	function contains(a, obj) {
	var i;
	for (i = 0; i < a.length; i++) {
	if (a[i] === obj) {
	return true;
	}
	}
	return false;
	}
		Helper function for LEMrandNG()

2630	function LEMuniqueToFirst()
	{
	var group = arguments[0];
	var pos = 1 + +arguments[1] + (group - 1) * 25;
		Implementing uniqueToFirst()

	if ((trim(arguments[pos])=='') || (typeof arguments[pos] == "undefined")) {
	return true;
	}
	
	if ((contains(arguments[pos],',')) || (contains(arguments[pos],';'))) {
	return false;
	}
	
	for (i = 2; i < arguments.length; ++i) {
	var arg = arguments[i];
	if (trim(arg)=='')
	continue;
	if (i != pos && arg == arguments[pos]) {
	return false;
	}
	}
	return true;
	}	

2630	function LEMsliderOnClick()
	{
	if ((typeof document.sliderHandle1 === "undefined") || (typeof document.sliderHandle2 === "undefined"))
	{
	var sliders = new Array();
	var sliderHandles = new Array();
	sliders = document.getElementsByClassName("ui-slider-1");
	sliderHandles = document.getElementsByClassName("ui-slider-handle");
	document.sliderHandle1 = sliderHandles[0];
	sliders[0].onclick = function(){document.sliderHandle1.style.width = "15px";};
	document.sliderHandle2 = sliderHandles[1];
	sliders[1].onclick = function(){document.sliderHandle2.style.width = "15px";};
	}
		Implementing sliderOnClick()

```
        /* The problem with a for loop: The object that is
        passed to the onclick function may not be
        overwritten! */

        return true;
}
```

Generating Questions

For name generators and for name interpreters similar questions need to be constructed very often. In Earlier versions of LimeSurvey it is possible to export the questionnaire structure as CSV/Excel file, adapt it e.g. by addition rows for additional questions and import it back into the LimeSurvey questionnaire. However, this is no longer possible. Another solution is to generate questions by a script into a .lsq (single questions or sets of questions) or .lsg (entire groups of questions) files and import these into LimeSurvey. For this purpose some python scripts were created. Follow these steps in order to generate (groups of) questions with less effort:

1. Of course, you need a running python installation. See http://www.python.org/getit/
2. Then, if you want to change the questions in the questionnaire template, do it via the web interface. Export the prototype question (for name generators) and/or prototype group (for name interpreters) as .lsq or .lsg group respectively.
3. Look at the python scripts in the folder "python" of the release and adapt it to your needs. Note that you also need to adapt the pre- and post-text files in most cases.
4. Import the .lsq and/or .lsg files into your LimeSurvey questionnaire.
5. In most cases you need to reorder the questions in LimeSurvey.
6. Test it.

Some Tricks that needed to be applied

- LimeSurvey allows the filtering of rows in matrix questions. This is applied in groups "(3) Information about persons". However, to do so it requires a question whose answers determine displayed rows (i.e. the row names/subquestion titles of the question that is to be filtered need to potentially match the answers of the filtering question. For this survey, a question of type array is used, and the answers are pre-defined by {ng_X} as the sampled names. If a sample variable is empty it gets filtered out.

Footer

Please, send any suggestions/comments/typos/mistakes relating to the topic to info@sascha-holzhauer.de.