z-Tree Cheat Sheet

Prepared by Ali Seyhun Saral
Comments and suggestions are welcomed
saral@posteo.de

Operators

`=`	assignment	`>=`	greater or equal
`+`	addition	`==`	equals
`-`	substraction	`!=`	unequal
`*`	multiplication	`&`	and
`/`	division	`|`	or
`<`	smaller	`:`	next higher scope
`>`	greater	`\`	highest scope (globals)
`<=`	smaller or equal		

Functions

Variable Functions

`abs(x)`	Absolute value
`gettime()`	Seconds since computer was started
`if(cond,x,y)`	If cond is true, x, otherwise y
`max(x,y)`	Maximum of x and y
`min(x,y)`	Minimum of x and y
`mod(x,y)`	Remainder of x/y
`power(x,y)`	x^y
`random()`	Uniform random between 0 and 1
`randomgauss()`	Normal random with avg 0 and sd1
`round(x,y)`	Rounds x to a multiple of y
`rounddown(x,y)`	Rounds x down to a multiple of y
`roundup(x,y)`	Rounds x up to a multiple of y
`same(x)`	Equality check of x with higher scope (`x==:x`)
`sqrt(x)`	Square root of x

Table Functions

(*cond* is optional for all functions below)

`average(`*cond*`,`*x*`)`	Average of the numeric values
`count(`*cond*`)`	Number of (found) records
`find(`*cond*`,`*x*`)`	First (found)value
`maximum(`*cond*`,`*x*`)`	Maximum of the (found) values
`minimum(`*cond*`,`*x*`)`	Minimum of the (found) values
`sum(`*cond*`,`*x*`)`	Sum of the (found) values

Arrays

`array` *myarray* `[`*n*`]`	Creates an array with n elements
myarray `[i]`	Calls i^{th} element of the array *myarray*

Conditional Statements

`if (`*cond*`) { `*exprs*` }`	If *cond* is TRUE, *exprs* are executed
`if (`*cond*`) { `*exprs1*` } elseif (`*cond2*`) { exprs2 }`	If *cond1* is TRUE, *exprs1* are executed; otherwise if *cond2* is TRUE, *exprs2* are executed.
`if (`*cond*`) { `*exprs*` } else { otherexprs }`	If cond is TRUE, *exprs* are executed; otherwise *otherexprs* are executed

Loops and Iterators

`while(`*cond*`){`*exprs*`}`	While *cond* is true, *expr* are executed
`repeat {`*exprs*`} while (`*cond*`);`	*expr* are executed, then while *cond* is true expr are executed
`later(`*x*`) do {`*exprs*`}`	After x seconds, *expr* are executed
`later(`*x*`) repeat {`*exprs*`}`	Each x seconds, *expr* are executed

(!) Iterators are replaced with *for* loops from version 4+

`iterator(i,n)`	i runs from 1 to n
`iterator(i,m,n)`	i runs from m to n in steps on 1
`iterator(i,m,n,s)`	i runs from m to n in steps of s

Built-in Variables

Globals

`Period`	Current period
`NumPeriods`	Total number of periods
`RepeatTreatment`	Repeat treatment if > 0

Subjects

`Period`	Current period
`Subject`	Subject number
`Group`	Group number
`TotalProfit`	Total profit in treatment
`Participate`	Enter stage if 1, do not if 0
`Leave stage`	Leave active stage if 1

Session

`FinalProfit`	Income without show-up fee
`ShowUpFee`	Show-up fee
`MoneyEarned`	FinalProfit + ShowUpFee
`MoneyAdded`	Credit given to subject
`MoneyToPay`	FinalProfit + ShowUpFee + MoneyAdded

Layouts

`!text:` *value1* `= "`*Label 1*`"; `*value2* `= "`*Label 2*`";`
`!button:` *value1* `= "`*Label 1*`"; `*value2* `= "`*Label 2*`";`
`!radio:` *value1* `= "`*Label 1*`"; `*value2* `= "`*Label 2*`";`
`!radiosequence:` *value1* `= "`*Label 1*`"; `*value2* `= "Label 2";`
`!radioline:` `leftvalue = "`*LabelLeft*`"; rightvalue = "LabelRight"; `*numberofbuttons*
`!slider:` `leftvalue = "`*LabelLeft*`"; rightvalue = "`*LabelRight*`"; `*numberofincrements*
`!scrollbar:` `leftvalue = "`*LabelLeft*`"; rightvalue = "`*LabelRight*`"; `*numberofincrements*
`!checkbox:` `1 = "`*Label*`";`
`!string`

Text Formatting

`<>`	Process variables inside labels
`<x | layout>`	Print the value of the variable x inside label
`{\rtf ... }`	RTF formatted text

RTF Codes

`\fs18`	font size 18pt	`\b`	start bold
`\tab`	tabulator	`\b0`	end bold
`\line`	new line	`\i`	start italic
`\ql`	aligned to left	`\i0`	end italic
`\qr`	aligned to right	`\colortbl`	define colors
`\qc`	aligned to center	`\cf1`	start color 1

RTF Example

```
{\rtf \fs21 This is \i italic \i0 and this is \b bold \b0 text }
```

Result: This is *italic* and this is **bold** text.

```
{\rtf {\colortbl;\red0 \green0 \blue0;\red255\green0\blue0;} \This is \cf2 red \cf1 and the rest is black.  }
```

Result: This is red and the rest is black

Common Operations

Getting opponents' variable in two player games

```
Opponent_x = find(same(Group) & not(same(Subject)),x);
```

Rank according to a variable number within the group

```
Rank = count(same(Subject) & :x >= x);
```

Conditional participation to a stage

```
Participate = if(x == 1,1,0);
```

Getting variable values from previous period

```
x = OLDsubjects.find(same(Subject), x);
```

Keyboard Shortcuts

Start treatment	`F5`	Restart clock	`Shift + F12`
Stop clock	`F12`	Break loop	`Ctrl + Shift + F5`

References

Fischbacher, Bendrick, Schmidt (2005) *z-Tree 3.5 Tutorial and Reference Manual.* `www.ztree.uzh.ch/static/doc/manual.pdf`.