



# INTRODUCTION TO Z-TREE

Antonio Alonso Arechar  
Yale University



# SCHEDULE

- Interactive games
- Sequential games
- **Advanced Topics**
- Frontiers



# ADVANCED TOPICS

1. Contracts table
2. Chat boxes +
3. External programs

# CONTRACTS TABLE

- In our PGG we calculated and displayed the sum of contributions
- What if we wish to show **individual contributions**?
- In the **Subjects table** we could sketch some rules to find group members

```
waitingscreen
Results =|= (30)
subjects.do { ... }
  SumC = sum( same( Group ), Contribution);
  Profit = M - Contribution + EfficiencyFactor * SumC / 3;
  Player_B = maximum( same( Group ) & not( same(Subject)), Subject);
  Player_C = minimum( same( Group ) & not( same(Subject)), Subject);
  Player_B_contribution = find( Subject == Player_B, Contribution);
  Player_C_contribution = find( Subject == Player_C, Contribution);
Active screen
Standard
  Your contribution: OUT( Contribution )
  Sum of contributions: OUT( SumC )
  Player B's individual contribution: OUT( Player_B )
  Player C's individual contribution: OUT( Player_C )
  Your profit: OUT( Profit )
  OK
Waitingscreen
```

Subject	Group	Profit	TotalProfit	Participate	Contribution	TimeOK	DeSumC	Player_B	Player_C	Player_B_	Player_C_
1	1	13	13	1	1	9	6	3	2	0	0
2	1	12	12	1	2	13	6	3	1	0	0
3	1	11	11	1	3	16	6	2	1	0	0
4	2	16	16	1	4	20	15	6	5	0	0
5	2	15	15	1	5	23	15	6	4	0	0
6	2	14	14	1	6	26	15	5	4	0	0

# CONTRACTS TABLE

- How to find subjects' contributions using *current* records?
- The scope operator (:)**!

A	B	C = sum( A * B );	D = sum( :A * B );	E = sum( :A * :B );
2	5	10 + 48 + 56 = 114	10 + 24 + 14 = 48	10 + 10 + 10 = 30
4	12	10 + 48 + 56 = 114	20 + 48 + 28 = 96	48 + 48 + 48 = 144
8	7	10 + 48 + 56 = 114	40 + 96 + 56 = 192	56 + 56 + 56 = 168

```

WaitingScreen
Results =|= (30)
subjects.do { ... }
  SumC = sum( same( Group ), Contribution);
  Profit = M - Contribution + EfficiencyFactor * SumC / 3;
  Player_B = maximum( same( Group ) & not( same( Subject)), Subject);
  Player_C = minimum( same( Group ) & not( same( Subject)), Subject);
  Player_B_contribution = find( Subject == :Player_B, Contribution);
  Player_C_contribution = find( Subject == :Player_C, Contribution);
Active screen
Standard
  Your contribution: OUT( Contribution )
  Sum of contributions: OUT( SumC )
  Player B's individual contribution: OUT( Player_B_contribution )
  Player C's individual contribution: OUT( Player_C_contribution )
  Your profit: OUT( Profit )
  OK
WaitingScreen
  
```

Subject	Group	Profit	TotalProfit	Participate	Contribution	TimeOK	DeSumC	Player_B	Player_C	Player_B_	Player_C_
1	1	13	13	1	1	22	6	3	2	3	2
2	1	12	12	1	2	18	6	3	1	3	1
3	1	11	11	1	3	12	6	2	1	2	1
4	2	16	16	1	4	7	15	6	5	6	5
5	2	15	15	1	5	2	15	6	4	6	4
6	2	14	14	1	6	-3	15	5	4	5	4

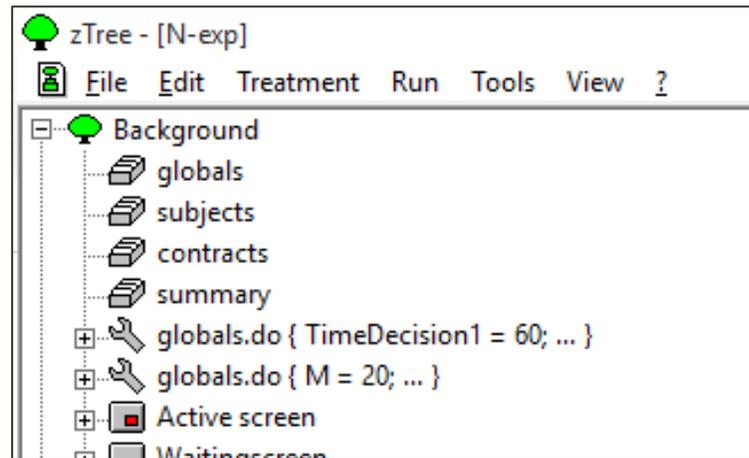
# CONTRACTS TABLE

- The scope operator is so useful z-Tree set a default we've already seen!
- `SumC = sum( same( Group), Contribution);`
- `SumC = sum(Group == : Group), Contribution);`

*“(My) SumC is the sum over the contributions of those whose Group is equal to mine”*

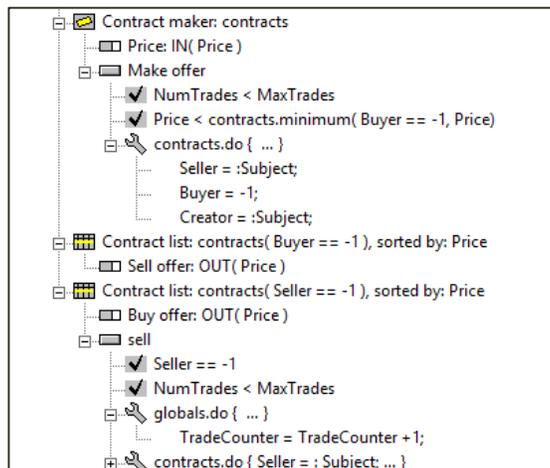
# CONTRACTS TABLE

- It is hard to handle others' records in the **Subjects table** as group size increases
- **Contracts table** can contain an indefinite number of rows
- Holds default variable `Period` only



# CONTRACTS TABLE

- Records in the **Contracts table** can be created, selected, viewed and edited in boxes:
  - Contract creation box
  - Contract list box
  - Contract grid box
- By using several boxes, it is possible to do different things in one screen – in one stage.



# CONTRACTS TABLE

- Here we will display individual records in a new stage called *Individual Contributions*
- We need to plan the active screen's layout
- **Treatment → New Box → Contact Grid Box (sic)**

Help Box

Name:   With frame

Width [p/%]:

Height [p/%]:

Distance to the margin [p/%]:

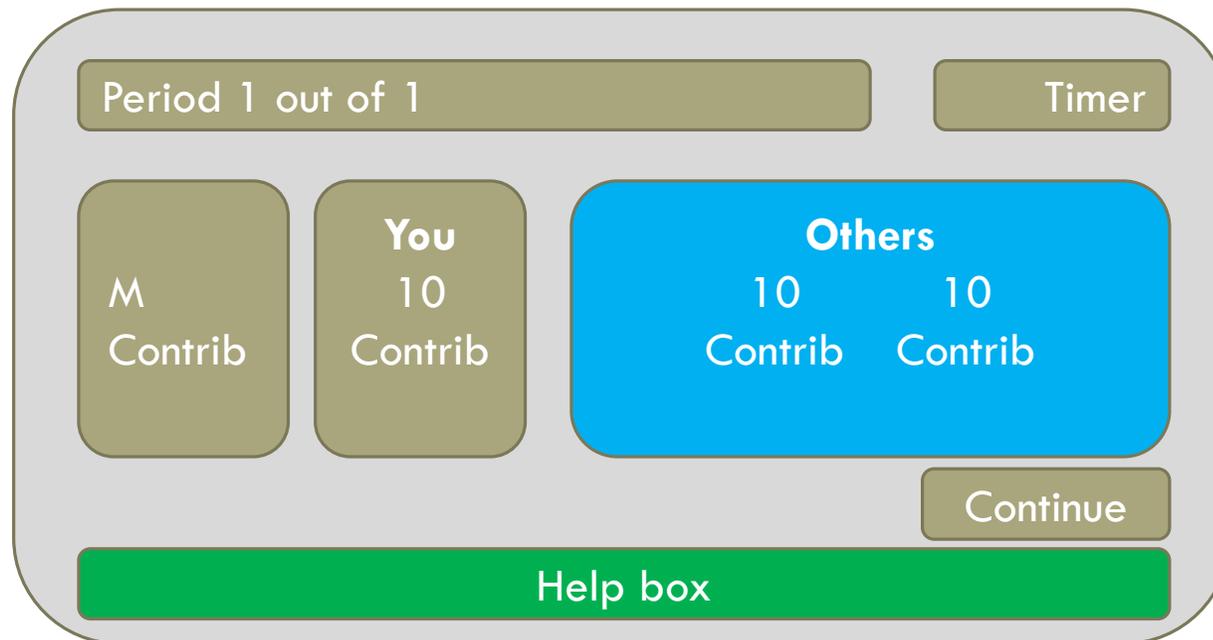
Standard Box

Name:   with Frame

Width [p/%]:

Height [p/%]:

Distance to the margin [p/%]:



Contract Box

Name:   With frame

Width [p/%]:

Height [p/%]:

Distance to the margin [p/%]:

Standard Box

Name:   with Frame

Width [p/%]:

Height [p/%]:

Distance to the margin [p/%]:

Standard Box

Name:   with Frame

Width [p/%]:

Height [p/%]:

Distance to the margin [p/%]:

# CONTRACTS TABLE

- Now that we have the skeleton of our active screen we need to populate it
- Input of the Standard boxes is easy and already defined for the subject (M, C)
- For the contracts box we need a program:

```
subjects.do{
  if( same(Group) & not( same( Subject) )) {
    contracts.new{
      Sender = ::Subject; // Note the double scope here takes us back to Subjects table!
      Receiver = :Subject;
      Group = :Group;
      Endowment = :M;
      OthersContribution = :Contribution;
      Random = random();
    }
  }
}
```

contracts	Period	Sender	Receiver	Group	Endowme	OthersCor	Random
contracts	1	1	2	1	10	3	0.415471
contracts	1	1	3	1	10	2	0.482208
contracts	1	2	1	1	10	1	0.609844
contracts	1	2	3	1	10	2	0.977137
contracts	1	3	1	1	10	1	0.3239
contracts	1	3	2	1	10	3	0.987702

# CONTRACTS TABLE

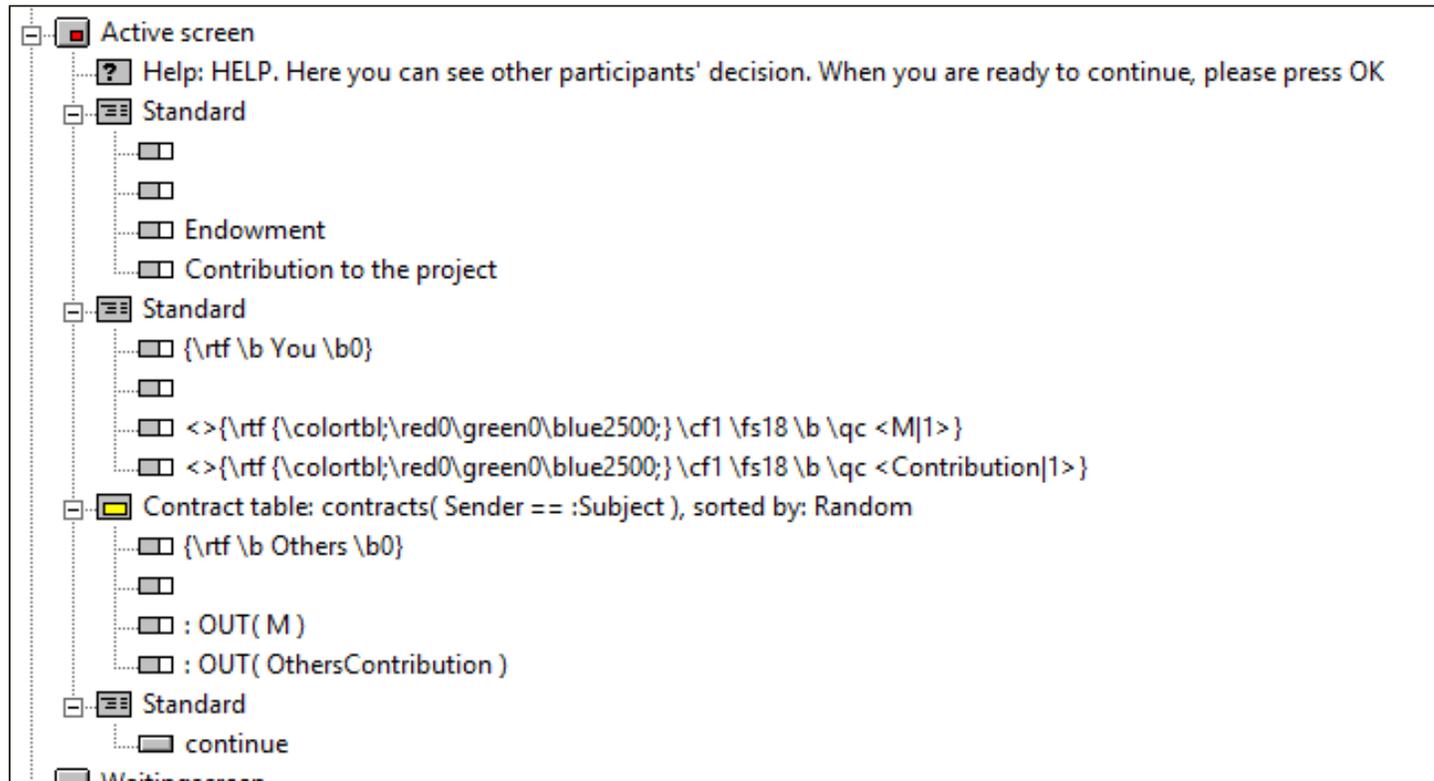
- Note that the Contract Box has specific features:
  - Condition: Defines what records to show (if left blank will produce all records)
  - Sorting: Sorts the variables introduced in the requested order

The image shows a configuration window for a 'Contract Box'. It contains several sections for setting up data display:

- Table:** A dropdown menu set to 'contracts'.
- Owner var.:** An empty text input field.
- Condition:** A text input field containing the expression 'Sender == :Subject'.
- Sorting:** A dropdown menu set to 'Random'.
- Scrolling:** A section with two sub-sections:
  - Records' display:** Two radio buttons: 'Labels on the top' (unselected) and 'Labels on the left' (selected).
  - Empty records allowed:** An unchecked checkbox.
  - All records empty allowed:** An unchecked checkbox.
- Buttons:** A section with two sub-sections:
  - Position:** A 3x3 grid of radio buttons. The bottom-right button is selected.
  - Arrangement:** Two radio buttons: 'In rows' (selected) and 'In columns' (unselected).

# CONTRACTS TABLE

- That's it! We can now start filling in the boxes:



# CONTRACTS TABLE

- Note that the contracts table also allow us to scale up our group sizes:

Period		1 of 1					Remaining time [sec]: 7	
	You	Others						
Endowment	10	10	10	10	10	10		
Contribution to the project	1	4	3	2	3	4		

continue

HELP  
Here you can see other participants' decision. When you are ready to continue, please press OK

# CONTRACTS TABLE

- Or include a stage with individual punishment / rewards:

Period: 1 out of 1 Remaining time [sec]: 158

Endowment	20	20	20
Contribution	20	0	10
Contribution in percent of the endowment	100	0	50
Your decision in stage 2	-	<input type="text" value="-5"/>	<input type="text" value="-2"/>

Assign no points: 0  
Assign negative-points: negative number

Total cost of the deduction points you assigned 7

HELP  
Please insert your decision. Note the sign of your assigned points. Then press the button "calculation".  
When you are done press "OK".

# CHAT BOXES

- The **chat box** allows controlled communication between subjects
- Subjects see a text entry field, and when they press ↵, the text is stored in contracts
- Suppose we wish to see if *post-decision* communication affects cooperation in the PD
- Open **PD.ztt** and add a new Chat Box on the active screen of Results

Standard Box

Name   with Frame

Width [p/%)

Height [p/%)

Distance to the margin [p/%)

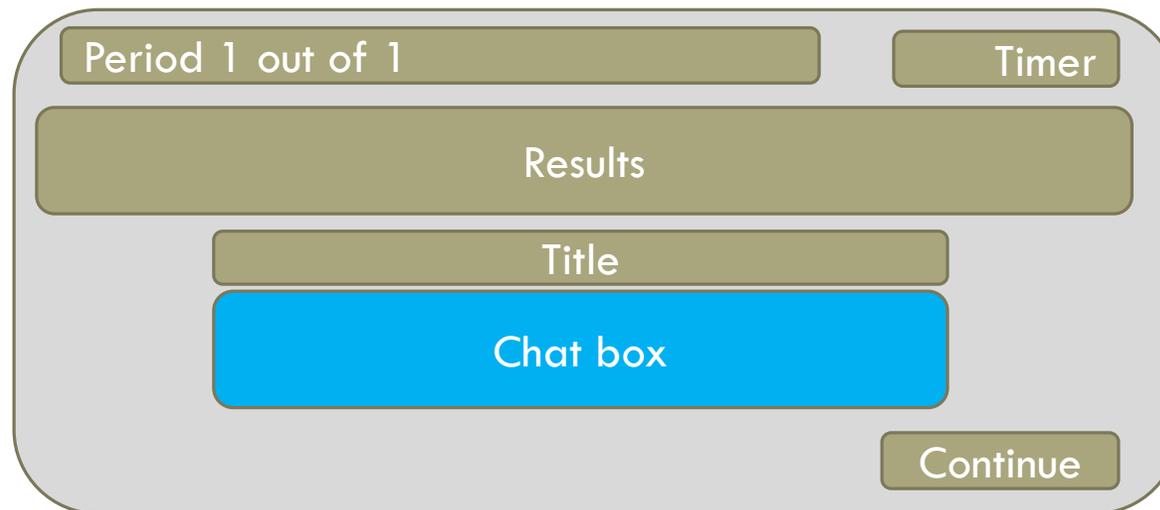
Chat Box

Name   With frame

Width [p/%)

Height [p/%)

Distance to the margin [p/%)



Standard Box

Name   with Frame

Width [p/%)

Height [p/%)

Distance to the margin [p/%)

Standard Box

Name   with Frame

Width [p/%)

Height [p/%)

Distance to the margin [p/%)

# CHAT BOXES

- Chat boxes have not only display conditions but also **Condition**
- In Condition we specify what output from Contracts we want to be displayed
- Chat boxes work as contracts boxes so we can show *current* input as output
- Remember to fill in the other boxes!

The screenshot shows a dialog box titled "Chat Box" with a close button (X) in the top right corner. The dialog is divided into several sections:

- Name:** "Chat box" with a checked "With frame" checkbox.
- Width [p%]:** An empty text box.
- Height [p%]:** "20%".
- Distance to the margin [p%]:** A sub-dialog with three input boxes, each containing "20%".
- Adjustment of the remaining box:** Four checkboxes: "left" (unchecked), "top" (unchecked), "right" (unchecked), and "bottom" (unchecked).
- Display condition:** A large empty text area.
- Table:** A dropdown menu showing "contracts".
- Input var.:** "Words".
- Number of characters:** An empty text box.
- Number of lines:** An empty text box.
- Condition:** "Group == :Group".
- Output text:** A text area containing the code: `<><if(Subject==:Subject,1,0)||text:1="" You said: ";0=" Other said: "> <Words:1>`.
- Buttons:** "OK" and "Cancel" buttons in the top right.
- Options:** "Wrap text" and "Output text centered" checkboxes at the bottom.

# CHAT BOXES

- Note that our chat box is storing records of *Words* but not of the people sending them
- Once participants are ready open the xls file we are editing
- Create a program for the Chat Box that stores those variables!

Program

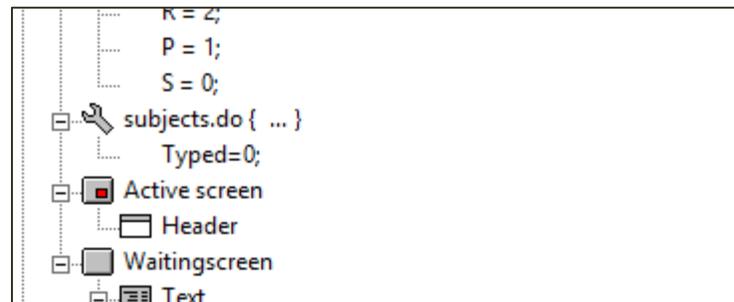
Table: contracts Owner Var

Condition

Program: Subject = :Subject;  
Group = :Group;

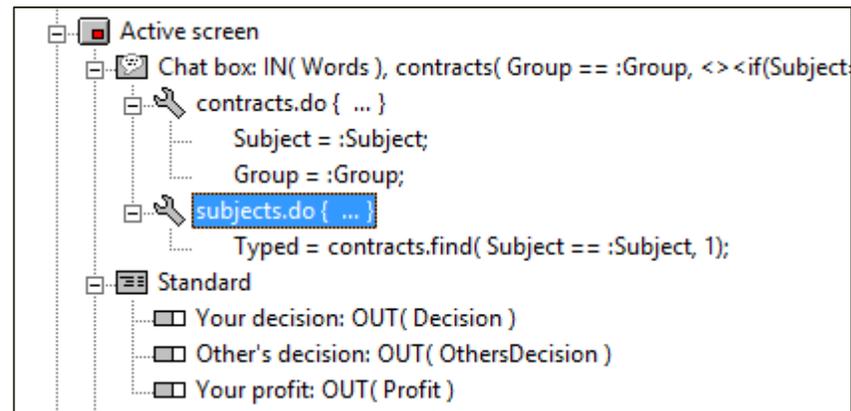
# CHAT BOXES

- What if we want to **force** participants to enter some text before leaving the stage?
- We could do so by having some way to check that a participant typed something
- **Checkers** allow you to do so!
- First let's *initialize* a subjects variable called Typed in the Background
- Run a period of this treatment and open **Run** → **Subjects Table**



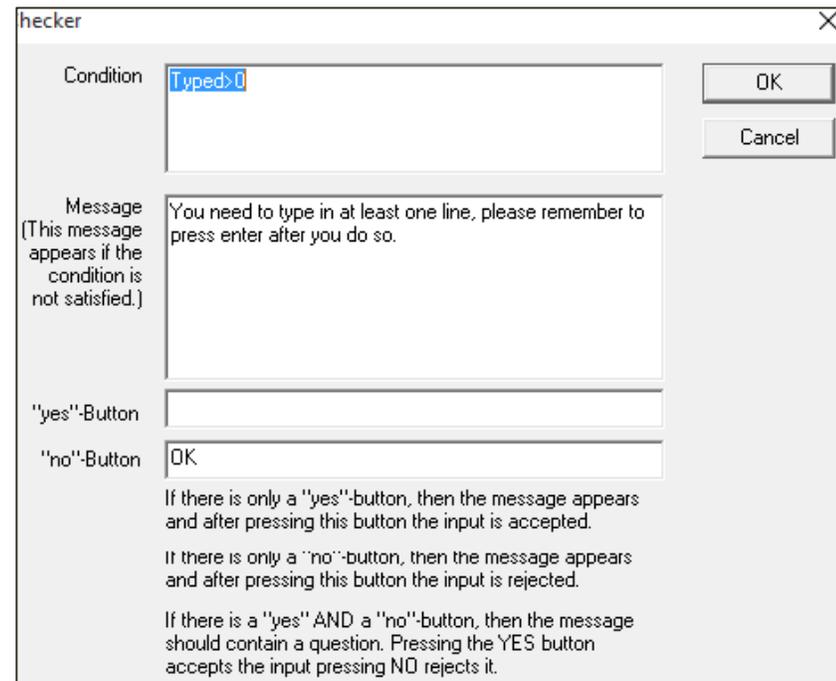
# CHAT BOXES

- Now we need to *activate* Typed, how?
- Recall that records for the chat are only created when something is typed
- Let's create a trigger program once the event happens
- Run a period of this treatment and open **Run** → **Subjects Table & Contracts Table**



# CHAT BOXES

- A natural place to put a checker is a button
- Select the OK button and press **Treatment** → **New Checker**



The screenshot shows a dialog box titled "hecker" with a close button (X) in the top right corner. The dialog is divided into several sections:

- Condition:** A text input field containing "Typed> 0".
- Message:** A text area containing the text "You need to type in at least one line, please remember to press enter after you do so." Below this text area are two buttons: "OK" and "Cancel".
- "yes"-Button:** An empty text input field.
- "no"-Button:** A text input field containing "OK".

Below the "no"-Button field, there is explanatory text:

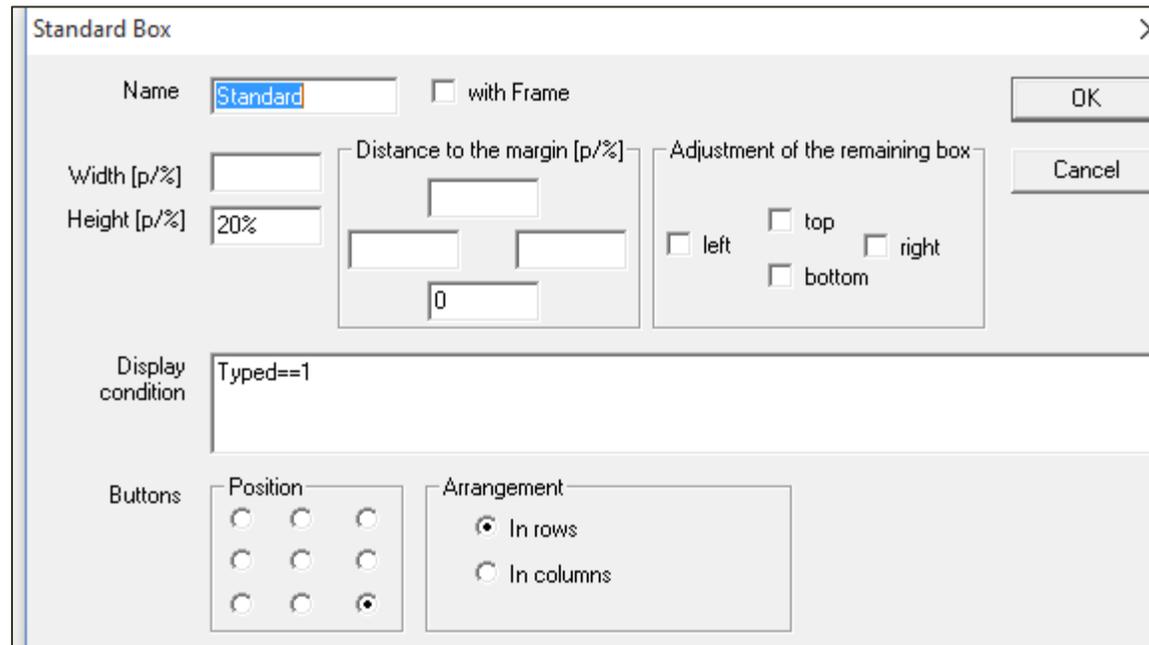
If there is only a "yes"-button, then the message appears and after pressing this button the input is accepted.

If there is only a "no"-button, then the message appears and after pressing this button the input is rejected.

If there is a "yes" AND a "no"-button, then the message should contain a question. Pressing the YES button accepts the input pressing NO rejects it.

# CHAT BOXES

- A *nicer* alternative would be to show the OK button only if  $Typed > 0$
- Note that a similar approach allows you to conditionally display anything!

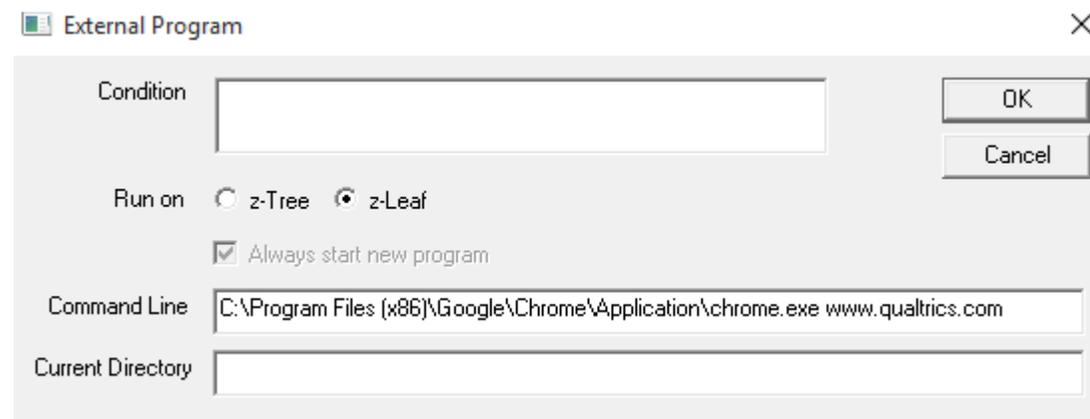


The image shows a dialog box titled "Standard Box" with a close button (X) in the top right corner. The dialog contains the following fields and controls:

- Name:** A text field containing "Standard" and a checkbox labeled "with Frame".
- Width [p/%]:** An empty text field.
- Height [p/%]:** A text field containing "20%".
- Distance to the margin [p/%]:** A text field containing "0".
- Adjustment of the remaining box:** A group box containing four checkboxes: "left", "top", "right", and "bottom".
- Display condition:** A text field containing the expression "Typed==1".
- Buttons:** A group box containing two sub-sections:
  - Position:** A 3x3 grid of radio buttons. The bottom-right radio button is selected.
  - Arrangement:** Two radio buttons: "In rows" (selected) and "In columns".
- OK and Cancel buttons:** Located on the right side of the dialog.

# EXTERNAL PROGRAMS

- Sometimes you might want participants to go to Qualtrics instead of a questionnaire
- z-Tree allows you to open external programs
- Create a final screen (Participate?) and select **Treatment** → **New External Program**
- As participants work on the survey you can run an empty Address form!



The screenshot shows a dialog box titled "External Program" with a close button (X) in the top right corner. The dialog contains the following fields and options:

- Condition:** An empty text input field.
- Run on:** Two radio buttons, "z-Tree" (unselected) and "z-Leaf" (selected).
- Always start new program:** A checked checkbox.
- Command Line:** A text input field containing the command: `C:\Program Files (x86)\Google\Chrome\Application\chrome.exe www.qualtrics.com`
- Current Directory:** An empty text input field.
- Buttons:** "OK" and "Cancel" buttons are located on the right side of the dialog.

# EXTERNAL PROGRAMS

- How to link participants' choices and their demographics though?
- One solution is to ask participants to enter their computer id, but errors happen!
- Remember that Qualtrics records IP addresses
- You can get each of the computers' ip addresses in the lab by googling "my ip"

