Z-tree: the Zurich Toolbox for Readymade Economic Experiments

A fast-forward course

Alexey Charkov

ICEF HSE

August 24, 2010

Z-tree: What and Why

- The 'industry standard' framework for conducting economic experiments
  Developed at the University of Zurich and widely used worldwide. Available from the authors free of charge after signing a written license agreement.
  Suitable for a wide range of quantitative and qualitative studies, from questionnaires to simple games to complex multi-market real-time interactions.
- Provides portability and replicability to experimental treatments
- Requires little prior infrastructural arrangements to set up a lab
- Abstracts away most of the routine programming, allowing to concentrate development effort on the treatment itself

Z-tree: Architecture

- Server program: ztree.exe
  The server is run on the experimenter's PC. Loading treatments, setting parameters, synchronizing interactions, tracking and logging the course of experiment.
  Also the place where treatments are created and edited.
- Client program: zleaf.exe
  The client is run on subjects' PCs. Display of relevant information about the treatment, processing of input.
  Relies on network communication (TCP/IP) with the server for any of its functions.

Z-tree: Starting up

As Z-tree is a network-oriented application, it may require unblocking in a firewall (usually asked at the first run). So, administrative privileges on all used computers may be required for initial setup.

- The server program: just start ztree.exe as a normal Windows application
- The client program: needs to know the network location of ztree.exe
  May require manual setting command line options for:
  - Server IP address (/server 192.168.0.55)
  - Unique client identification name (/name Client1)

  (either run from the command prompt or create a shortcut to z-leaf and put the parameters in its properties)

Z-tree: Main concepts

- Every treatment consists of periods, every period consists of stages
- Each stage shows predefined screens to clients
  - Provide static and dynamic information about the treatment
  - Accept user input
  - Wait for other players when needed
- All information is stored in a built-in database with predefined tables
- Multiple programs may be inserted at various stages, as well as at the beginning of treatment and in push buttons
- Every program executes in its 'own' database table, **sequentially for every record**

Z-tree: Running a ready-made treatment

1. Load the treatment file with *File* $\Rightarrow$ *Open...*
2. Set an appropriate language for built-in interface elements with *Treatment* $\Rightarrow$ *Language*
3. Set up a questionnaire including at least an address form to be able to properly conclude a session and assign payments
   - *File* $\Rightarrow$ *New Questionnaire*
   - *Questionnaire* $\Rightarrow$ *New Address Form*
   - Clear field names you do not want to be requested from subjects, edit the rest as you see fit
   - Or you can load a predefined questionnaire with *File* $\Rightarrow$ *Open...*
4. Ensure that all clients are connected by inspecting *Run* $\Rightarrow$ *Clients Table*
5. **Select** the treatment sub-window and start it via *Run* $\Rightarrow$ *Start Treatment* (or press F5)

Programming brush-up

Any program consists of expressions, statements and control structures

- Expressions: manipulate information
  - Variables: *VariableA*
  - Arithmetics: *VariableA + VariableB*
  - Booleans (true/false): *VariableA >= VariableB*
  - Other operators: *if(VariableA == VariableB, 1, 0)*
  - Function calls: *round(exp(2 * pi() / 3), 1)*
- Statements: request something to be done
  - Assignment: *MyVariable = SomeExpression;*
- Control structures: change how and whether statements are executed
  - Conditions: *if (condition) { statements } else { other statements }*
  - Loops: *while (condition) { statements }*

Z-tree: Programming specifics

- Multiple programs can be added to a treatment (before a stage, in a push button...)
- Programs operate on database tables
- Variables are columns in database tables
- A program executes fully on each database record, then moves to the next record until the end of a table (beware of errors)
- Other tables and records can be accessed from a program:
  - Table functions: *find(condition, variable)*
  - Referencing other tables: *contracts.sum(BuyerID > 0, Price)*
  - The **scope operator** (a colon prefix). It allows to refer to the value of a variable at the outer level of nesting: *sum(Role == :Role, Contribution)*