z-Tree 入門講座

飯田善郎

2005 年 4 月

目次

1　実験を行う

京都産業大学の実験室で z-Tree を用いて実験する手順について説明します。

1.1　起動：教卓マシンをサーバとして使う場合

・サーバの起動

1. 準備室のサーバの電源を入れる。（これは教員が行う。サーバへのログインは不要）
2. 教卓マシンの電源を入れる。
3. ユーザー名：ztree2
   パスワード：ztree2
   ログイン先：ORC
   でログインする。
4. デスクトップ上の「z-Tree」フォルダのショートカットをクリックしてフォルダをあける。
5. 結果を入れるフォルダ ” results ” フォルダを作る。フォルダの新規作成→フォルダ名を ” results ” に変える。
   注意:既に ” results ” フォルダがある場合は、中を確認します。空ならそのまま使ってください。中身がある場合、適当な名前（中のファイルを見てそのファイルの日付にするのがいいでしょう）に名前を付け替え、新たに “ results ” フォルダを作ります。
6. ”z-Tree のショートカット ” アイコンをダブルクリック。
   注意:ショートカットでない方はクリックしないこと！

図 1　ショートカットの方をクリックして起動する

・クライアントの起動

1. サーバで z-Tree が起動していることを確認する。
2. 必要数のクライアントの電源を入れる。
3. ユーザー名：zleaf2
   パスワード：zleaf2
   ログイン先：ORC
   でログインする。

ログインしたら画面上の z-Leaf へのショートカットをダブルクリックして z-Leaf を起動する。

1.2 起動：準備室のサーバを直接使う場合

注意：基本的に授業での実験は教卓マシンをサーバとして使います。準備室のサーバを使う場合も以下に説明しますが基本的には何かトラブルでもない限りは教卓マシンを使ってください。

・サーバの起動

1. 準備室の Windows および Linux サーバの電源を入れる。
2. Windows サーバに
   ユーザー名：ztree
   パスワード：ztree
   でログインする。
3. デスクトップ上の「z-Tree」フォルダのショートカットをクリックしてフォルダをあける。
4. 結果を入れるフォルダ ”results ” フォルダを作る。フォルダの新規作成→フォルダ名を ” result ” に変える。
5. ”z-Tree のショートカット ” アイコンをダブルクリック。

・クライアントの起動

1. サーバで z-Tree が起動していることを確認する。
2. 必要数のクライアントの電源を入れる
3. ユーザー名：zleaf
   パスワード：zleaf
   ログイン先：ORC
   でログインする。

1.3 ゲームの呼び出し

1. File → Open でファイルを呼び出す。

1.4 実験の開始

1. 画面上の木の形のアイコン「Background」をダブルクリックする。「General Parameters」のダイアログボックスが開くので、Number of subjects にその日の参加人数（zleaf を起動しているクライアントの数）を入れます。Number of groups は一人当たりのグループの数を入れます。Practice period に練習セッションの回数、 Paying period に本番のセッションの回数を入れます。
2. 参加者がグループ数で割り切れない場合、端数の人たちは端数同士で組まされます。その場合 zleaf を立ち上げた順番が遅い順に端数扱いされます。
3. メニューから「Run」→「Start Treatment」を選択すると実験が開始されます。

1.5 実験前、実験中の被験者のモニター

- 実験前
  Run → Client ’ s Table で z-Tree に接続している z-Leaf を確認できます。開始前に必要な人数のクライアントが接続しているか確認します。
- 実験中
  Run → Subjects Table で各被験者がどのステージにいるかや被験者の意思決定などが確認できます。

1.6 実験の継続

全セッションが終了した後でまた実験する場合は、クライアントの画面がすべて「お待ちください」の表示になっているのを確認した上で、もう一度 Run → Start Treatment を選択すると再開されます。

1.7 実験の終了

実験が終了したら File → Open で z-Tree フォルダ内の「BasicGames」内の Question ファイルを呼び出して Run → Start Questionnaire を実行します。氏名と学生証番号を入力する画面になるのでそれを入力させれば終了です。ここでは日本語入力ができないので、氏名などはローマ字で入力させてください。この Questionnaire を実行しないと z-Tree の処理が終わらないので注意してください。Questionnaire を実行すると

```
XXYYZZQQ.pay
```

というファイルが作られます。ここにはクライアントのコンピューターの番号、z-Tree が割り振った被験者 ID、被験者が入力した自分の名前、被験者の得点が一覧になってテキストファイル形式で記録されます。後で被験者が誰かを確認したり、得点を成績等に反映させるときに便利でしょう。

2 実験結果の整理と分析

2.1 XLS ファイル

実験結果は result フォルダに入っています。実験が終わったら result フォルダは名前を変えてください。自分たちのグループの名前にするのが望ましいでしょう。（もし名前を変えようとするとエラーが出るようなら、Ctrl+Alt+Del で「タスクマネージャ」を出し、「プロセス」のタブをクリックし、「イメージ名」から z-Tree を選択して「タスクの終了」をクリックしてください。）

実験結果は次のようなファイル名で、エクセルファイルとして ” result ” フォルダに入っています。それ以外のファイルも生成されていますが、結果を見るだけならこれだけで十分です。

```
XXYYZZQQ.xls
```

“ XXYYZZ ” はそれぞれ西暦、月、日で、QQ は実験の開始時刻で決まるアルファベットか数字です。ここには z-Tree で使われたほぼすべての変数がセッションごとにどうなったかが記録されています。あまり重要でないデータも書き込まれていますので、次のようにして整理することを薦めます。

1. エクセルで読み込み、C の列に注目する。
2. C 列のセルが「subjects」となっている行が被験者の行動にかかわる部分です。被験者の行動の分析にはこれだけあれば十分ですので、これ以外の行はすべて削除するか、別のところに移してしまいます。ただし、ダブルオークション実験の場合は「contract」となっている行も残してください。

2.2 SeparateTable でファイルを分割する

必要な部分だけ切り分ける方法として、SeparateTable という機能があります。次のようにしてください。

1. z-Tree 上の全てのトリートメントを閉じ、z-Tree も一旦終了する
2. 結果の xls ファイルを適当なフォルダに移動させる（もし移動をしようとするとエラーが出るようなら、Ctrl+Alt+Del で「タスクマネージャ」を出し、z-Tree を選択して「タスクの終了」をクリックしてください。）
3. z-Tree を起動しメニューから　Tools → Separate Tables を選択する。
4. 「ファイルを開く」のダイアログボックスが開くので、結果の xls ファイルをクリックして「OK」を押す
5. 結果のファイル名の後に grobal, session, subjects, summary がそれぞれついた新しいファイルが 4 つ生成される。

オークションの実験でなければ一般に subjects の結果だけを見ればいいでしょう。

実験で注目すべき列は次のようになります。

- D 列　Period　実験が何回目か
- E 列　Subject　被験者番号　zleaf を立ち上げた順に割り振られます
- F 列　Group　グループ番号です。誰が誰と組んだかが分かります。

- G 列　Profit　各セッションで各被験者がどれだけ利益を上げたか。

無論、各主体の入力を記録した列もあるはずです。それはプログラムごとに違うのでそれも残してください。次の節で具体的にデータの見方を説明します。

2.3　データの見方の例１　公共財実験の場合

トリートメントの例「公共財実験」の場合、実験結果が結果ファイルにどのように記録されていて、どこに注意してみたらよいかを確認しましょう。

まずは結果の xls ファイルを SeparateTable で分割してできる”XYYZZQQ __ subjects.xls”を開いてみてください。次のようになっています。

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	***** Thank you for using z-Tree. Please do not forget to mention in your paper that you used z-Tree. For instance as follows: The experiment												
2	SessionID	Treatment	subjects	Period	Subject	Group	Profit	TotalProfi	Participat	Efficiency	Endowmer	SumC	Contribution
3	050406KG	1	subjects	1	1	1	8	8	1	2	5	9	3
4	050406KG	1	subjects	1	2	1	6	6	1	2	5	9	5
5	050406KG	1	subjects	1	3	1	10	10	1	2	5	9	1
6	050406KG	1	subjects	1	4	2	6.333333	6.333333	1	2	5	5	2
7	050406KG	1	subjects	1	5	2	5.333333	5.333333	1	2	5	5	3
8	050406KG	1	subjects	1	6	2	8.333333	8.333333	1	2	5	5	0
9	050406KG	1	subjects	1	7	3	9.666667	9.666667	1	2	5	7	0
10	050406KG	1	subjects	1	8	3	7.666667	7.666667	1	2	5	7	2
11	050406KG	1	subjects	1	9	3	4.666667	4.666667	1	2	5	7	5
12	050406KG	1	subjects	1	10	4	7	7	1	2	5	3	0
13	050406KG	1	subjects	1	11	4	4	4	1	2	5	3	3
14	050406KG	1	subjects	1	12	4	7	7	1	2	5	3	0
15	050406KG	1	subjects	1	13	5	8.666667	8.666667	1	2	5	10	3
16	050406KG	1	subjects	1	14	5	9.666667	9.666667	1	2	5	10	2
17	050406KG	1	subjects	1	15	5	6.666667	6.666667	1	2	5	10	5
18	050406KG	1	subjects	1	16	6	7	7	1	2	5	5	3
19	050406KG	1	subjects	1	17	6	8	8	1	2	5	5	2
20	050406KG	1	subjects	2	1	1	7	15	1	2	5	9	4
21	050406KG	1	subjects	2	2	1	8	14	1	2	5	9	3
22	050406KG	1	subjects	2	3	1	9	19	1	2	5	9	2
23	050406KG	1	subjects	2	4	2	4.666667	11	1	2	5	1	1
24	050406KG	1	subjects	2	5	2	5.666667	11	1	2	5	1	0
25	050406KG	1	subjects	2	6	2	5.666667	14	1	2	5	1	0
26	050406KG	1	subjects	2	7	3	9	18.66667	1	2	5	6	0
27	050406KG	1	subjects	2	8	3	7	14.66667	1	2	5	6	2
28	050406KG	1	subjects	2	9	3	5	9.666667	1	2	5	6	4
29	050406KG	1	subjects	2	10	4	5	12	1	2	5	0	0
30	050406KG	1	subjects	2	11	4	5	9	1	2	5	0	0
31	050406KG	1	subjects	2	12	4	5	12	1	2	5	0	0
32	050406KG	1	subjects	2	13	5	7.666667	16.33333	1	2	5	4	0
33	050406KG	1	subjects	2	14	5	7.666667	17.33333	1	2	5	4	0
34	050406KG	1	subjects	2	15	5	3.666667	10.33333	1	2	5	4	4
35	050406KG	1	subjects	2	16	6	6	13	1	2	5	4	3
36	050406KG	1	subjects	2	17	6	8	16	1	2	5	4	1

図 2　実験結果のデータ

重要なのは D 列の回数 Period、E 列の被験者番号 Subject,　F 列のグループ番号 Group,　そして M 列の被験者のグループへの投資量 Contribution です。図では分かりやすくするためこれらの列を太字にしています。

青の枠線の中を見てください。青の枠内の Period は全て 1 です。ここには各被験者の 1 回目のピリオドの状態や行動が記録されています。（この枠線も説明のためにつけたもので、実際には表示されません。）Subjects には 1 から 17 の数値があり、この実験に 17 人の被験者が参加していたことがわかります。Group の欄を見ればどの被験者同士が同じグループなのかわかります。各被験者の状態や行動は列を見てゆくことでわかります。例えばワークシートの 3 行目を見てください。図 2 では青の枠の中の、水色のセルの箇所です。これは 1 回目のピリオド（Period）では被験者番号（Subject）1 番の被験者はグループ番号 1 番のグループ（Group）に属し、グループへの投資額（Contribution）は 3 だった、ということを表しています。

図 3　データの読み方

他の被験者の行動も同様に読んでゆくことができます。赤い枠内の Period は全て 2 です。ここには 2 回目のピリオドの内容が記録されています。先ほどの被験者番号 1 番が 2 回目のピリオドでなにをしたかはピンク色のセルを見てください。

このようにピリオドごとの行動が上から下へと記録されています。このままだと被験者が全ピリオドを通じてどのように行動したか、一覧性があまりよくありません。データをコピー・ペーストして被験者の投資額だけを図 4 のように整理するとのちのち扱いやすくなるでしょう。

被験者のグループへの投資額(Contribution)

		回数(Period)									
被験者番号 (Subject)	グループ番号 (Group)	1	2	3	4	5	6	7	8	9	10
1	1	3	4	5	2	3	3	3	3	3	3
2	1	5	3	2	2	1	3	2	2	2	2
3	1	1	2	1	4	1	2	3	1	4	2
4	2	2	1	0	5	4	3	2	5	4	5
5	2	3	0	0	2	2	2	1	2	3	4
6	2	0	0	3	0	0	0	2	3	5	5
7	3	0	0	0	0	0	0	0	0	0	0
8	3	2	2	2	2	0	2	2	1	0	0
9	3	5	4	4	4	4	3	3	0	1	2
10	4	0	0	0	1	2	4	0	0	0	0
11	4	3	0	0	2	3	3	4	2	0	0
12	4	0	0	3	2	4	3	4	0	0	1
13	5	3	0	1	5	2	0	1	5	4	5
14	5	2	0	2	4	0	3	1	5	3	0
15	5	5	4	1	5	5	0	0	5	5	0
16	6	3	3	4	2	1	3	2	3	3	4
17	6	2	1	2	2	0	0	1	3	2	1

図 4　整理した実験データ

2.4　マクロ機能を用いたデータの整理

エクセルの機能を使ってデータを手っ取り早く整理する方法を以下に示します。慣れれば 5 分ほどで作業が終わるので、実験の結果をその場で被験者に見てもらうような場合にも使えるでしょう。

1. まず、結果の xls ファイルを SeparateTable で分割します。結果のファイルをエクセルで開くと図 5 のようになっています。重要なのは E 列の被験者番号 Subject,　F 列のグループ番号 Group,　そして

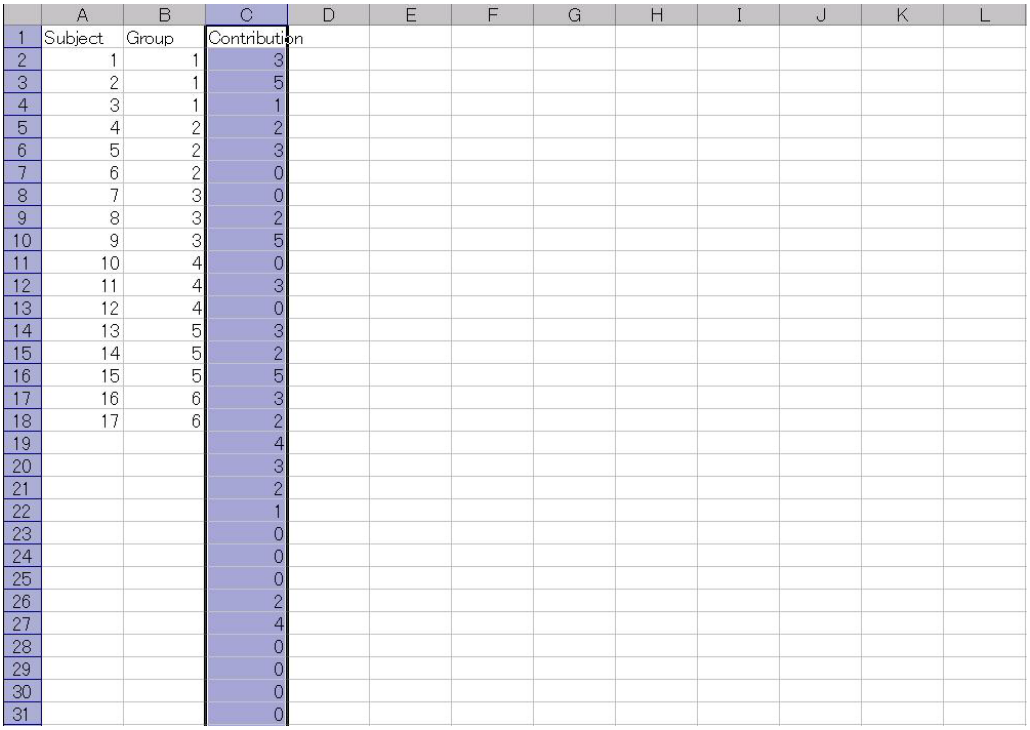
M 列の被験者の公共財への投資量 Contribution です。これらを抽出し、整理してゆくわけです。

2. 被験者番号「Subject」とグループ番号「Group」部分を選択して他のワークシートにコピーします。（図 5 参照）

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	***** Thank you for using z-Tree. Please do not forget to mention in your paper that you used z-Tree. For instance as follows: The experi												
2	SessionID	Treatment	subjects	Period	Subject	Group	Profit	TotalProfi	Participate	EfficiencyF	Endowmer	SumC	Contributi
3	050406KG	1	subjects	1	1	1	8	8	1	2	5	9	3
4	050406KG	1	subjects	1	2	1	6	6	1	2	5	9	5
5	050406KG	1	subjects	1	3	1	10	10	1	2	5	9	1
6	050406KG	1	subjects	1	4	2	6.333333	6.333333	1	2	5	5	2
7	050406KG	1	subjects	1	5	2	5.333333	5.333333	1	2	5	5	3
8	050406KG	1	subjects	1	6	2	8.333333	8.333333	1	2	5	5	0
9	050406KG	1	subjects	1	7	3	9.666667	9.666667	1	2	5	7	0
10	050406KG	1	subjects	1	8	3	7.666667	7.666667	1	2	5	7	2
11	050406KG	1	subjects	1	9	3	4.666667	4.666667	1	2	5	7	5
12	050406KG	1	subjects	1	10	4	7	7	1	2	5	3	0
13	050406KG	1	subjects	1	11	4	4	4	1	2	5	3	3
14	050406KG	1	subjects	1	12	4	7	7	1	2	5	3	0
15	050406KG	1	subjects	1	13	5	8.666667	8.666667	1	2	5	10	3
16	050406KG	1	subjects	1	14	5	9.666667	9.666667	1	2	5	10	2
17	050406KG	1	subjects	1	15	5	6.666667	6.666667	1	2	5	10	5
18	050406KG	1	subjects	1	16	6	7	7	1	2	5	5	3
19	050406KG	1	subjects	1	17	6	8	8	1	2	5	5	2
20	050406KG	1	subjects	2	1	1	7	15	1	2	5	9	4
21	050406KG	1	subjects	2	2	1	8	14	1	2	5	9	3
22	050406KG	1	subjects	2	3	1	9	19	1	2	5	9	2
23	050406KG	1	subjects	2	4	2	4.666667	11	1	2	5	1	1
24	050406KG	1	subjects	2	5	2	5.666667	11	1	2	5	1	0

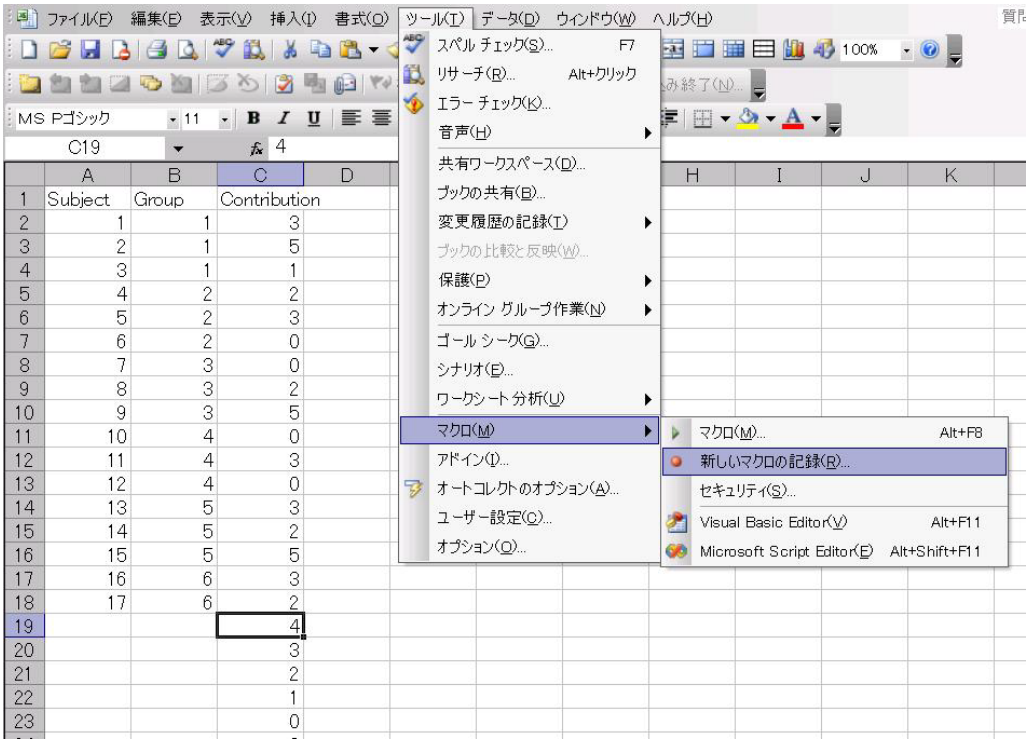
図 5　実験結果の整理 (1)：必要箇所の選択コピー

3. Contribution の列もコピーしてワークシートに貼り付けます。（図 6 参照）
4. 被験者の意思決定は縦一列に並んでいるので横に並べなおすためにカットとペーストを繰り返す必要があります。ピリオドが 10 回ある場合は 10 回、同じ作業を繰り返します。同じ作業の繰り返しは、次のようにマクロを使用することで手間を省くことが出来ます。
   （a）2 回目の最初の被験者のデータのセル（この例では C19）をクリックしてアクティブセルにし，ツール→マクロ→新しいマクロの記録をクリックする。(図 7 参照)
   （b）ダイアログボックスが出てきたら、(図 8 参照) ＣＴＲＬ＋」の右側のテキストボックスの中に「ａ」（半角小文字）をタイプし，ＯＫボタンを押す。
   （c）記録終了ボックスが出てきたら，相対参照ボタンをクリックしておく。(図 9 参照)
   注意：記録終了ボックスが出てこない場合、ツールバーの余白部分にマウスポインタを持っていって右クリックしてください。出てくるメニューの中の「記録終了」にチェックを入れると記録終了ボックスが出ます。
   （d）C19 のセルから下の最後のセルまで範囲指定する。
   （e）編集→切り取り→貼り付けで D2 のセルに貼り付ける。
   （f）3 回目の最初の被験者のデータのセル（この例では D19) をクリックしてアクティブにし，記録終了ボックスの終了ボタンを押す。(図 10 参照)
   （g）ＣＴＲＬキーを押しながら「ａ」キーを押すと 3 回目以降のデータが自動的にカット・ペーストされる。ピリオドの回数だけこれを繰り返す。
   注意：マクロの処理が終わったら、あるいはマクロに失敗してやり直す前にはツール→マクロ→マ



	A	B	C
1	Subject	Group	Contribution
2	1	1	3
3	2	1	5
4	3	1	1
5	4	2	2
6	5	2	3
7	6	2	0
8	7	3	0
9	8	3	2
10	9	3	5
11	10	4	0
12	11	4	3
13	12	4	0
14	13	5	3
15	14	5	2
16	15	5	5
17	16	6	3
18	17	6	2
19			4
20			3
21			2
22			1
23			0
24			0
25			0
26			2
27			4
28			0
29			0
30			0
31			0

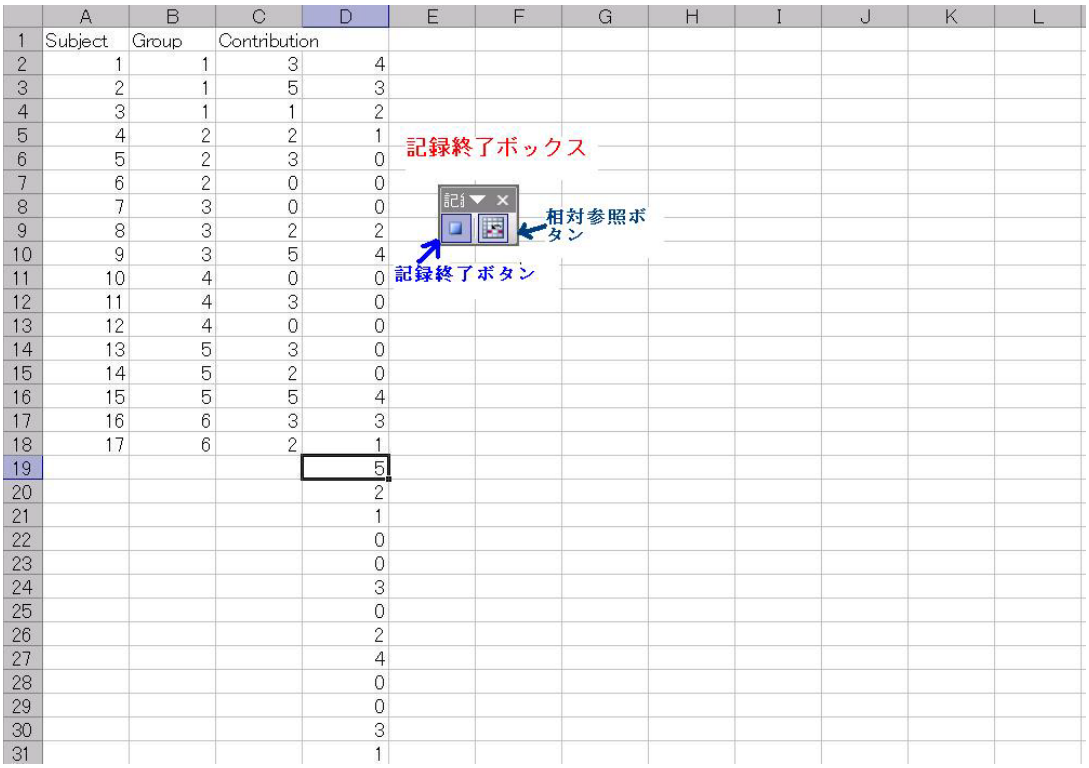
図 6　実験結果の整理 (2)：Contribution のコピー・ペースト



	A	B	C
1	Subject	Group	Contribution
2	1	1	3
3	2	1	5
4	3	1	1
5	4	2	2
6	5	2	3
7	6	2	0
8	7	3	0
9	8	3	2
10	9	3	5
11	10	4	0
12	11	4	3
13	12	4	0
14	13	5	3
15	14	5	2
16	15	5	5
17	16	6	3
18	17	6	2
19			4
20			3
21			2
22			1
23			0

図 7　実験結果の整理 (3)：マクロの記録

図 8　実験結果の整理 (4)：マクロの記録ダイアログボックス



	A	B	C	D
1	Subject	Group	Contribution	
2	1	1	3	4
3	2	1	5	3
4	3	1	1	2
5	4	2	2	1
6	5	2	3	0
7	6	2	0	0
8	7	3	0	0
9	8	3	2	2
10	9	3	5	4
11	10	4	0	0
12	11	4	3	0
13	12	4	0	0
14	13	5	3	0
15	14	5	2	0
16	15	5	5	4
17	16	6	3	3
18	17	6	2	1
19				5
20				2
21				1
22				0
23				0
24				3
25				0
26				2
27				4
28				0
29				0
30				3
31				1

図 9　実験結果の整理 (5)：マクロの記録終了ボックス

クロでマクロボックスを出し，作成したマクロを削除する。

5. データのコピーとペーストが終わったらラベルを入れたり行の幅を調節したりして見やすく整理してください。これでデータを扱いやすくなったはずです。
   注意:SPSS などの統計処理ソフトでデータを処理する場合はオリジナルのデータの方が扱いやすい場合があります。オリジナルのデータを消さないようにしましょう。

2.5　データの見方の例２　ダブルオークション

ダブルオークションのデータは次の図のようになっています。ここではデータの分割を使わないでおいた方が便利なので分割しないでおきます。図では見やすくするために一部のセルを表示していません。また説明のために先ほどと同様にここでは一部のセルを枠で囲ったり、字に色をつけてあります。

	A	B	C	D	E	F	G	H	W
1	050408UE	1	globals	Period	NumPerio	RepeatTre	SELLERTYPE	BUYERTYPE	AuctionNoStop
2	050408UE	1	globals	1	1	0	1	2	
3	050408UE	1	subjects	Period	Subject	Group	Profit	TotalProfit	Type
4	050408UE	1	subjects	1	1	1	350	350	2
5	050408UE	1	subjects	1	2	1	180	0	1
6	050408UE	1	subjects	1	3	1	220	0	2
7	050408UE	1	subjects	1	4	1	220	220	1
8	050408UE	1	summary	Period					
9	050408UE	1	summary	1					
10	050408UE	1	contracts	Period	Buyer	Seller	Price	Creator	TimeWueBuyerA
11	050408UE	1	contracts	1	1	2	200	2	
12	050408UE	1	contracts	1	3	4	180	3	
13	050408UE	1	contracts	1	−2	4	210	4	
14	050408UE	1	contracts	1	−1	2	190	2	
15	050408UE	1	contracts	1	1	4	180	4	
16	050408UE	1	session	Subject	FinalProfit	ShowUpFe	ShowUpFee	MoneyAdde	MoneyEarned
17	050408UE	1	session	1	350	0	0	0	
18	050408UE	1	session	2	180	0	0	0	
19	050408UE	1	session	3	220	0	0	0	
20	050408UE	1	session	4	220	0	0	0	

図 10　ダブルオークションの実験結果

このデータをどのように見てゆけばいいのか確認しましょう。

まずは赤い枠の部分ですがこれは Globals テーブルと言う実験全体に共通する設定が記録されている表です。赤字になっている部分を見てください。SELLERTYPE が 1、BUYERTYPE が 2 となっています。この実験では売り手は 1、買い手は 2 という識別のための番号が与えられています。これはどの被験者でも同じ扱いなので Grobal テーブルに記述されているのです。

次に青枠を見てください。この部分は Subjects テーブルといい、各被験者それぞれの設定や行動が記録されます。公共財や囚人のジレンマ実験では Subjects テーブルに被験者の行動も記録されますが、オークション実験では被験者の行動は次に述べる Contracts テーブルに記録されます。Subjects テーブルの青字になっている部分を見てください。Subjects の欄に 1,2,3,4 とあります。これは 4 人の被験者がこの実験に参加していてそれぞれ 1 から 4 の被験者番号を与えられていることを意味します。Type の欄に 2,1,2,1 とあります。これは先ほど Globals テーブルで定義された売り手と買い手の識別番号です。つまり被験者 1 と 3 は Type が 2 なので買い手、被験者 2 と 4 は Type が 1 なので売り手であることがわかります。

ダブルオークションのデータを検証するときに最も重要なのが緑の枠で囲まれた Contracts テーブルです。ここではどのように取引が推移したか見ることができます。一般には Creator, Price, Buyer, Seller この 4 つの項目を見ればいいでしょう。これらの項目は次のように解釈できます。

「Buyer と Seller の間で Price で取引が成立した。その Price は Creator が提示した」

たとえば緑枠内のセルの 2 行目を見てください。Creator が 2、Price が 200、Seller が 2、Buyer が 1 になっています。これは「被験者番号 2 番の売り手が価格 200 を提示し、被験者番号 1 番がそれに応じたため、2 番の売り手と 1 番の買い手の間で価格 200 で取引が成立した」ことを意味します。

図 11　成立した取引の記録の例

3 行目を見てみましょう。Creator が 3、Price が 180、Seller が 4、Buyer が 3 になっています。これは「被験者番号 3 番の買い手が価格 180 を提示し、被験者番号 4 番の売り手がこれに応じて販売したため、3 番の買い手と 4 番の売り手の間で取引が成立した」ことを意味します。

4 行目、5 行目では買い手の欄がマイナスになっています。マイナスは成立しなかった価格提示を意味します。-1 の「１」や-2 の「2」はここでは被験者番号ではないので気をつけてください。-1 は取引が成立しなかったことを意味します。緑の枠内の 5 行目を見てください。2 番の売り手は 190 の価格を提示したのですが時間切れになるまで誰も購入しなかったため、提示が受け入れられなかった印として-1 が買い手の欄についています。

図 12　成立しなかった取引の記録の例

-2 は、本人が値段を下げ、下げた値段が受け入れられたため無意味になった下げる前の価格提示につけられる印です。

従って 4 行目から 6 行目にかけて次のようなことが起きたことが読み取れます。まず、4 番の売り手が価格 210 を提示しました。それに対して 2 番がより低い価格 190 を提示しました。4 番は対抗してさらに低い 180 という価格をつけました。この提案に買い手 1 番が応じました。この取引は 6 行目で 4 番の売り手と 1 番の買い手の間で価格 180 で取引が成立したという形で記録されています。4 行目で 4 番が提示した価格 210 は無効となり、その印として-2 が買い手欄につけられました。5 行目で 2 番が提示した価格は最後まで受け入れられなかったため-1 が買い手欄につけられています。

今は売り手の提案が受け入れられなかった例だけを説明しましたが、当然買い手の側の提案が受け入れられない場合もあります。その場合は売り手の欄の方に-1 や-2 がつきます。

成立した取引の価格の推移だけを見たい場合は、Seller や Buyer の欄に-1 や-2 がある行を除外してみてゆ

けばよいでしょう。

3　z-Tree プログラム講習のための準備

3.1　教室での準備

z-Tree を学習のため実験室のクライアント PC で使う場合は次のようにしてから始めてください。実験室のクライアント PC は定期的にメンテナンスが入って初期状態に戻されるため、個人で作成したファイルなどは削除されます。必要なファイルやデータは実験室の PC の中には残さないでフロッピーなどに移しておいてください。

1. 実験室の PC の電源を入れる。
2. ユーザー名：student
   パスワード：student
   ログイン先：このコンピュータ
   でログインする
3. デスクトップ上のフォルダ b「z-Tree」のショートカットをクリック。
4. フォルダ内の z-Tree のアイコンをダブルクリック
   ・これで z-Tree が起動するはずです
5. 使用言語を日本語にしておきます。
   (a) Untitled Treatment 1 というウインドウが開いているはずです。右上の [×] を押して消してください。
   (b) メニューから　Treatment → Language → Nihongo　と選択してください。
   (c) メニューから　File → New Treatment　を選択してください。
   (d) 新しく開いたウインドウの一番下の行が「しばらくお待ちください」になっていたら OK です。
6. z-leaf のショートカットのアイコンをダブルクリック

CD-ROM のフォルダには 2 個の z-Leaf 用のショートカットアイコンが用意されています。それぞれ z-leaf1、z-leaf2 と名づけられています。授業で使うときはこのショートカットアイコンの方を使ってください。一つのゲームに 2 人以上の参加する場合 2 つ以上の z-Leaf が必要です。その場合は次のようにしてください。

例：3 つ目のショートカットアイコン「zleaf3」を作る場合

- z-leaf 1 か z-leaf2 のショートカットアイコンをコピー。
- 名前を右クリックしてアイコンの名前を zleaf3 に変更。
- アイコンを右クリック→プロパティを選択
- リンク先を次のように変更。変更部分を太字で示します。
  変更前　C:￥z-TreeCDrom￥zLeaf.exe /name zleaf1 /size 900x700 /Language Nihongo
  変更後　C:￥z-TreeCDrom￥zLeaf.exe /**name zleaf3** /**size 925x675** /Language Nihongo
  /name のオプションで名前を変更し、/size　オプションで横×高さの大きさを変更します。
  大きさは他の zleaf の画面とぴったり重なってしまわないように少しずつ変えましょう。上の例では zleaf3 は zleaf1 よりも少し横長で縦に短くしています。
- 「OK」を押して終了

3.2 各自のコンピュータでの準備

各自が自習できるよう z-Tree とプログラムの雛形が入った CD − ROM を渡しています。CD − ROM は「z-TreeCDrom」フォルダと「treatments」のフォルダを C ドライブにコピーすれば教室とほぼ同じ環境になります。

注意: この CD-ROM に入っている z-Tree は京都産業大学内で使用する許諾だけを得ています。決して人に譲渡したり Web 上などで公開してはいけません。

4　z-Tree プログラムのテストラン

プログラムを呼び出してテストランをしてみましょう。

1. File → Open で「treatments」のフォルダをあけ、「pd」のファイルを呼び出してください。
   これは囚人のジレンマと呼ばれるゲームのプログラムです。このプログラムは z-Tree の開発者である Urs Fichbacher の Web ページ (http://www.iew.unizh.ch/ztree/index.php) で公開されているものです。囚人のジレンマがどのようなゲームかは以下に説明します。

   囚人のジレンマ

   2 人の容疑者が別々に取調べを受けています。警察はこの 2 人がある重大事件の容疑者であるとにらんでいますが決定的証拠がありません。そこで別件の犯罪で逮捕し、重大犯罪についても自白させようとしています。今捕まっている別件の犯罪は免れようがなくこのまま黙秘を続けても 2 年の服役は免れないことは容疑者もわかっています。警察は重大事件のほうを自白させようと 2 人にこのような取引を個別に持ちかけています。

   「重大事件について、お前の相棒が黙秘しているうちにお前が自白したら捜査協力をしたということで司法取引で 1 年の服役で済むようにしてやろう。そのときには相棒は 4 年の服役になるだろう。しかしお前が黙秘を続けている間に相棒が自白したらお前が 4 年で相棒が 1 年の服役になる。二人ともが重大事件について自白したら 3 年の服役だ。」

   この二人の置かれている状況は次のように整理できます。服役は二人にとって楽しくないことなので服役年数をマイナスで示します。

		相　棒	
		黙秘	自白
自分	黙秘	-2 ＼-2	-4 ＼-1
	自白	-1 ＼-4	-3 ＼-3

表 1　囚人のジレンマ：利得表（1）

セルの中の数字は「自分の服役年数＼相棒の服役年数」です。あなたがこの容疑者だとしたらどのように行動するのが正しいでしょうか。相棒が黙秘するだろうと思うなら自白すれば刑はたった 1 年で済みます。しかし相棒も自白したら 3 年の刑です。それなら二人とも黙秘をして受ける 2 年の刑の方がましです。しかし自分が黙秘としているうちに相手が自白してしまったら自分はなんと 4 年も服役しなくてはなりません！　この例ではマイナスの利得を減らすためにどうするかを考えるゲームになっていますが、上の表の数字に全部 5 を足すことでプラスの利得を増やすためのゲームに置き換えることが出来ます。二人の選択肢も相手への協力と裏切りと言い換えると囚人のジレンマは次のような表で表されるゲームになります。

		相　棒	
		黙秘	自白
自分	黙秘	3 ＼ 3	1 ＼ 4
	自白	4 ＼ 1	2 ＼ 2

表 2　囚人のジレンマ：利得表（2）

2. zleaf1 と zleaf2　を立ち上げてください。ここではテストランなので実験者の役割と同時に自分で被験者役もやらなくてはなりません。
3. z-Tree が立ち上げた 2 つの z-Leaf を認識しているか確認してみましょう。メニューから Run → Clients ' Table を選択してください。今接続している z-leaf が表示されます。

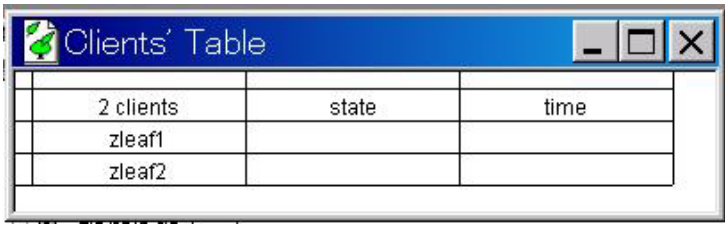


図 13　Clients ' Table で接続している z-Leaf を確認する

4. ゲームに何人参加して、同じゲームを何回繰り返すかを決めなくてはなりません。トリートメントの一番最初にある、木の形のアイコン「Background」をダブルクリックしてください。

図 14　Background アイコンをダブルクリックする

図 15 のような General Parameters のダイアログボックスが現れます。Number of subjects には被験者数 2 を入れます。Number of Groups にはグループ数を入れますが囚人のジレンマは 2 人一組なのでここでは 1 のままで結構です。# paying periods には実験を何回繰り返すかを入れます。ここも今は 1 のままで結構です。入力が終わったら「OK」を押してダイヤログボックスを閉じます。

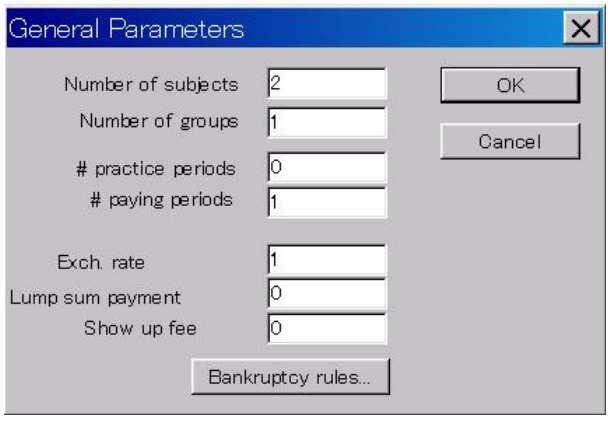


図 15　General Parameters を設定する

5. ではプログラムを実行してみましょう。メニューから Run → Start Treatment を選択します。スクリーンには図 16 のような表示が現れます。

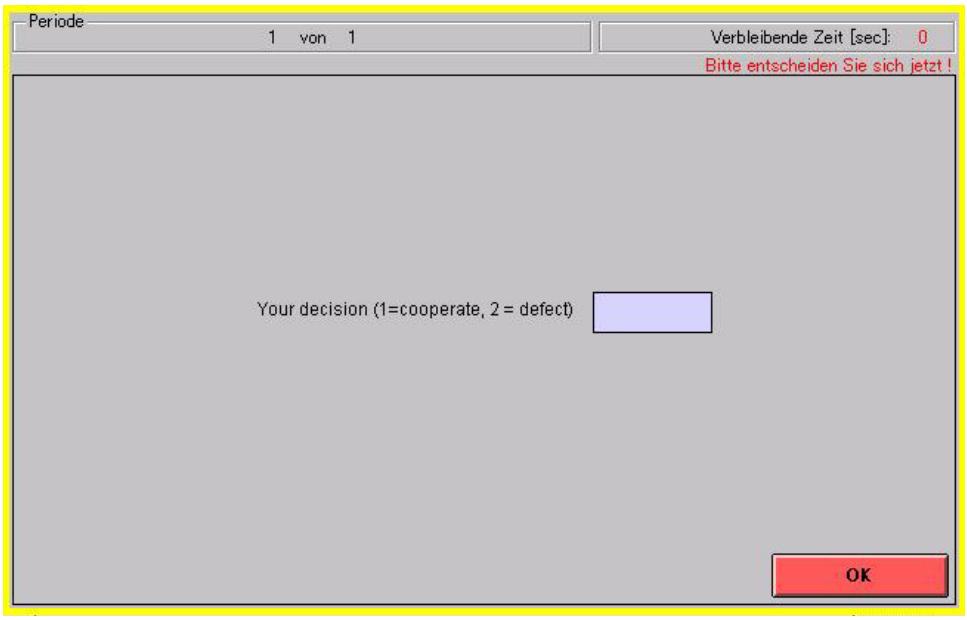


図 16　実験のスクリーン (1)　意思決定の入力画面

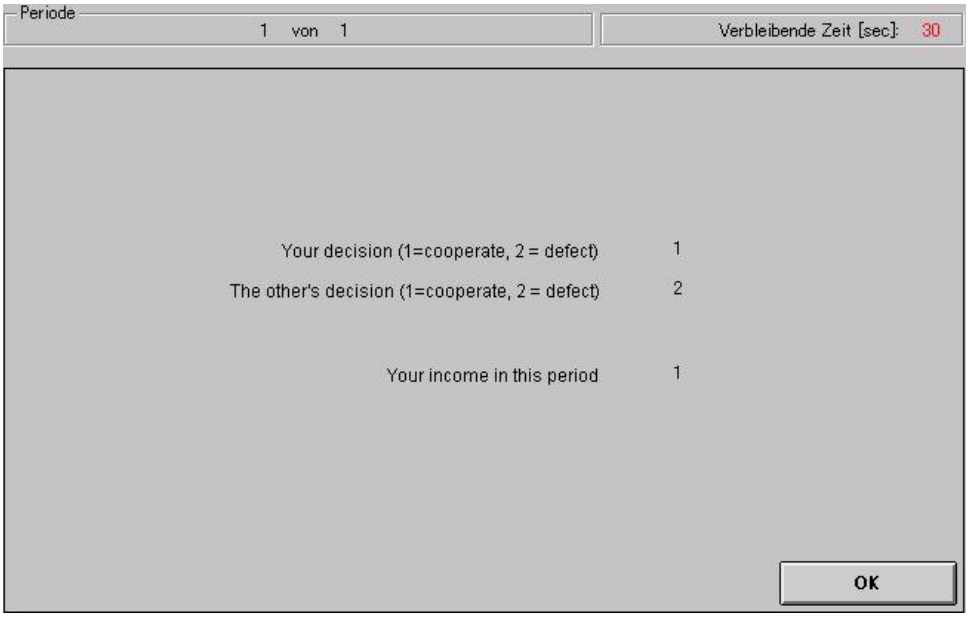


図 17　実験のスクリーン (2)　結果表示の画面

協力するなら 1、裏切るなら 2 を入力して OK を押します。z-Leaf は 2 つ起動していますので両方の z-Leaf 上で入力します。画面に表示される z-Leaf は [Alt]+[Tab] で切り替えることが出来ますが、

2 つの z-Leaf は少しずつ縦と横のサイズが違うので、どちらかが手前にあってももう一方が完全に隠れてしまうことはないはずです。見えている一部をクリックすることでも一番手前に表示させることができます。

画面の上部の表記がドイツ語になっているのに気づかれたでしょうか。これはこのトリートメントのプログラムがドイツ語環境に設定された z-Tree 上で作られたからです。後でこれを含めて表記を全て日本語に直してみましょう。

2 つの z-Leaf に入力が済むと図 17 のような結果の出力画面に変わります。これもあとで日本語表記に直してみましょう。

6. 結果を確認してみましょう。ウインドウの右上の [x] を押して z-Tree を終了します。警告メッセージが出ますが、実験本番でない限り無視しても問題ありません。z-Leaf は Alt を押しながら F4 を押すと終了します。

   実験結果のファイルは、z-Tree が置かれたフォルダに作成されています。ファイルは z-Tree と z-Leaf の接続のためや実験が途中で不具合に陥った場合に復帰させるために作成されるものなど実験結果とは無関係のものも作成されます。一般には 1 回の実行につき一つだけ作成される xls ファイルに結果が記述されていますので、それ以外は削除してしまってかまわないでしょう。実験結果のファイルの見方は 2 章を参照してください。被験者の行動は「Decision」の行に記述されています。

5　プログラムの改造

5.1　日本語環境への移行

4 章でテストランした囚人のジレンマの実験の画面表示を日本語に直してみましょう。まず 3.1 章の 3 で説明したとおりメニューで Treatment → Language → Nihongo　を選択し、メニューから　File → New Treatment　を選択してください。新しいウインドウが開きます。（すでに開いているならこの操作は不要です）4 章でテストランした pd のファイルも開いてください。その二つのファイルのウインドウを並べて、pd の内容をコピーして新しいファイルに移していきます。

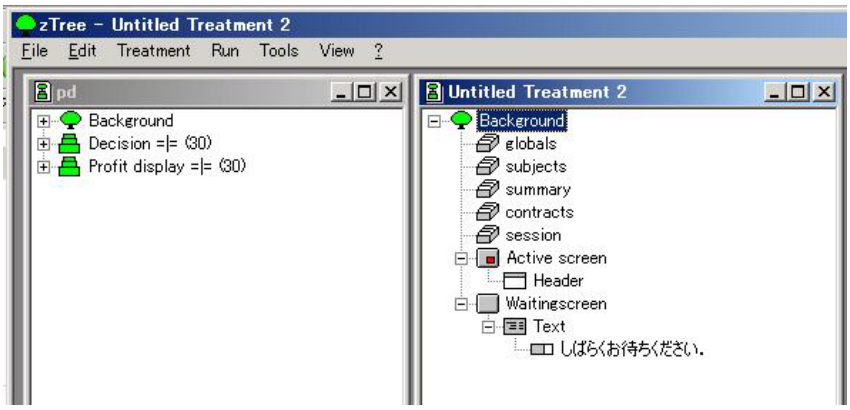


図 18　日本語環境への移行（1）

木の形のアイコンがバックグラウンドアイコンと呼ばれることは先に説明したとおりです。その下の階段状の緑のアイコンはステージアイコンと呼ばれます。pd には Decision と Profit display という二つのステージがあります。このステージを新しいファイルにコピーしましょう。

1. Decision のアイコンをクリック
2. メニューから Edit → Copy を選択
3. 新しいファイルのバックグラウンドアイコンをクリック
4. メニューから Edit → Paste を選択

これで Decision のアイコンがコピーできたはずです。同様に Profit display のアイコンもコピーします。その際 3 . の手順のところでは Background ではなく Decision のアイコンをクリックしてから貼り付けてください。新しいステージは現在選択されているアイコンのすぐ後ろに貼り付けられます。

これで各ステージが移植されましたが、まだ終わりではありません。バックグラウンドの中に各種の設定やプログラムが組み込まれているのでそれも移植します。各アイコンの左の [+] はステージの中身が折りたたまれて表示されていない状況にあることを示します。pd のファイルのバックグラウンドの [+] をクリックしてください。バックグラウンドの内容が表示されます。pd のファイルの方には新しいファイルの方にはない Globals.do…というスパナの形のアイコンがあります。これをコピーして新しいファイルのほうに貼り付けます。

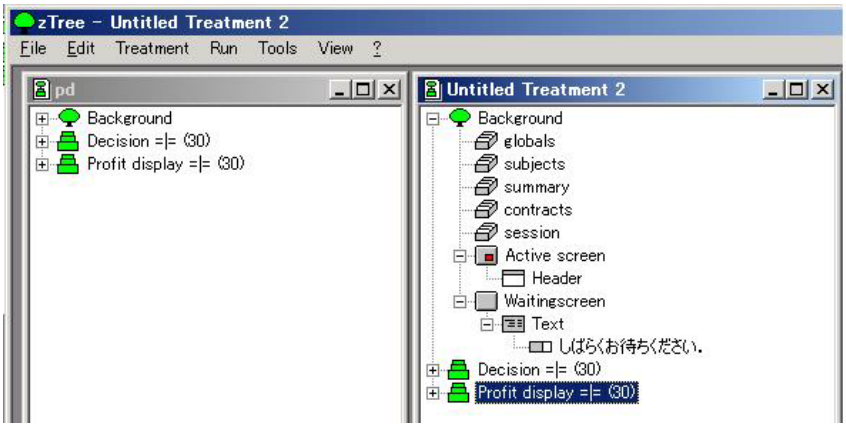


図 19　日本語環境への移行（2）ステージアイコンのコピー

1. Globals.do…のアイコンをクリック
2. メニューから Edit → Copy を選択
3. 新しいファイルの session のアイコンをクリック
4. メニューから Edit → Paste を選択

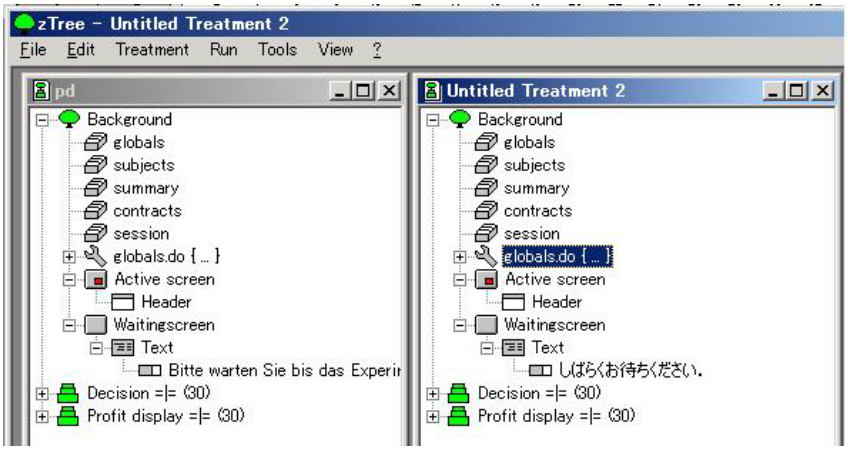


図 20　日本語環境への移行（3）プログラムアイコンのコピー

これで日本語環境のファイルに内容を移すことができました。ここからは実際に表記を日本語に直しますが、その前にファイル名をつけて保存しておきましょう。File → Save As…で適当な名前をつけて保存してください。ここで試しに再びテストランしてみてください。画面の上部の「Period」が「回数」に、「Verbleibende Zeit」が「残り時間」に変わっていることが確認できるでしょう。

意思決定の入力画面のメッセージを日本語に直してみましょう。「Decision」ステージの [+] をクリックして展開してみてください。ひとつのステージは大きく分けて Active Screen と Waiting Screen からなります。Active Screen は情報を表示したり被験者の入力を受け付ける画面です。Waiting Screen は表示や入力が終わった後次の画面に進むまでの待機状態のときに表示する画面です。Active Screen はボックスという小画面

を並べて構成されます。ボックスは何種類かありますが、一番基本となるのが Standard Box です。この例でもメインの画面は Standard Box で構成されています。

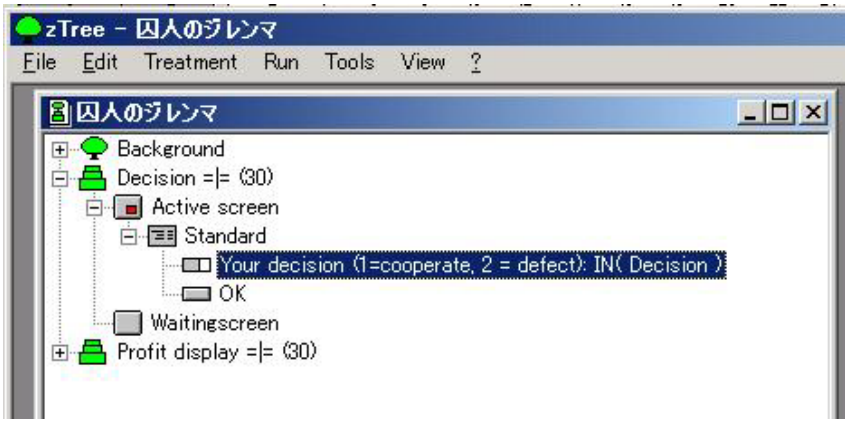


図 21　日本語環境への移行（4）スタンダードボックスとアイテム

Standard ボックスのアイコンの下層にある四角いアイコンはアイテムと呼ばれます。Your decision のアイテムをダブルクリックしてください。アイテムの内容を表示するダイアログボックスが開きます。Label の欄にある「Your decision(1=cooperate 2=defect)」を「あなたの意思決定（1=協力, 2=裏切り）に書き換えます。

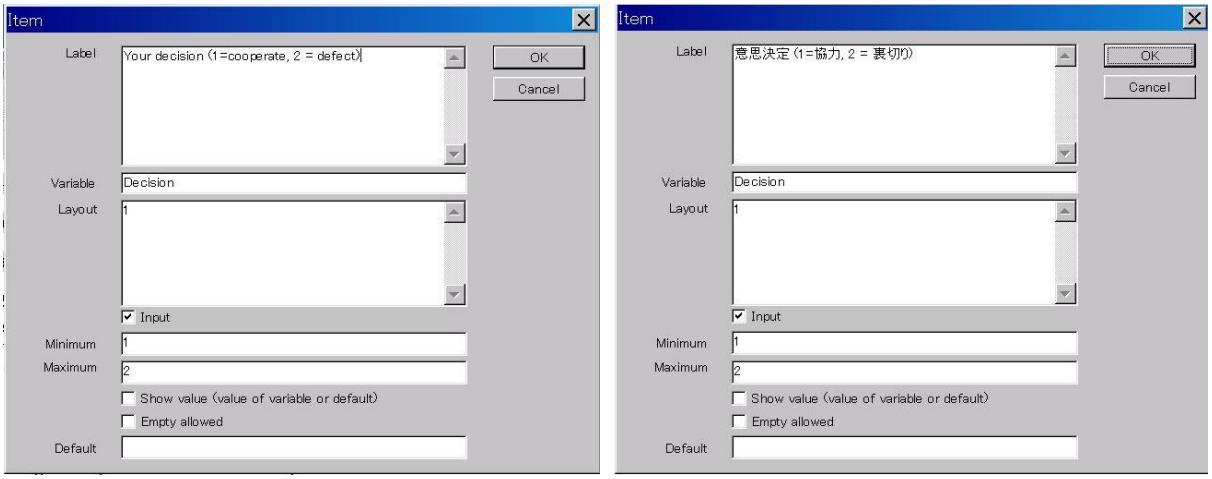


図 22　日本語環境への移行（5）アイテム内の書き換え

同様にして、一つ下のステージ Profit display の [+] をクリックして展開し、中のアイテムの Label を図 23 のように直します。

これで画面表示が日本語になりました。上書き保存の上でテストランしてみましょう。z-leaf を二つ起動し、トリートメント上の Background をダブルクリックしてパラメータを適切にセットして走らせてみてください。画面上のメッセージが日本語で表示されていれば成功です。

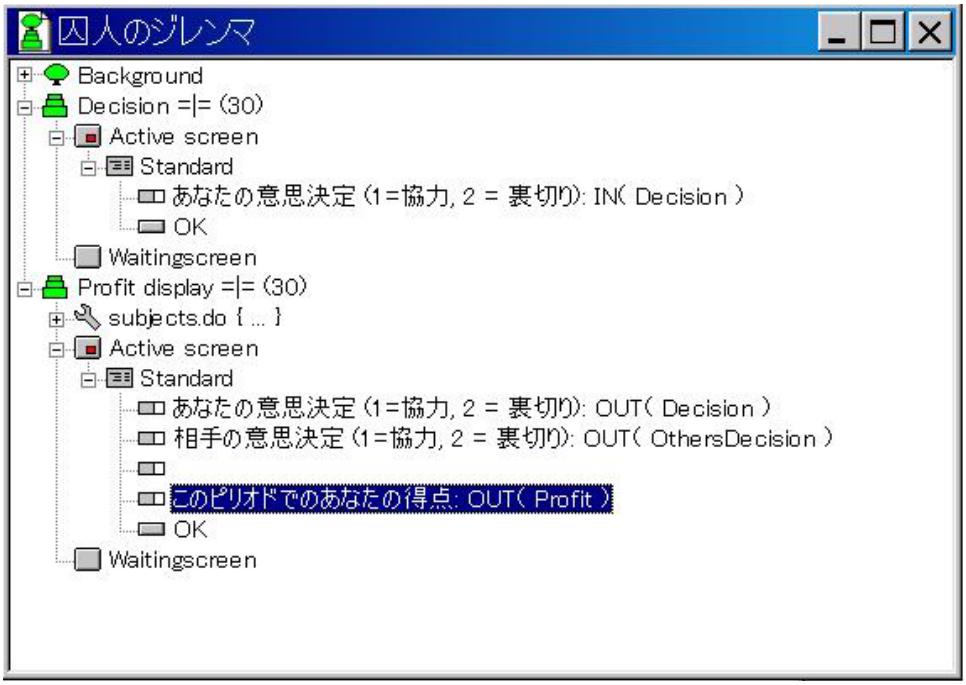


図 23　日本語環境への移行（6）完成

5.2　利得関数の改造

この囚人のジレンマのトリートメントは簡単に別のゲームに作り変えることが出来ます。ここでは「鹿狩りゲーム」に改造してみましょう。「鹿狩りゲーム」ついては以下に説明します。

鹿狩りゲーム (Stag Hunt Game)

2 人の狩人が鹿とウサギのいる山に出かけます。鹿は 2 人でかかれば捕らえることができますが。1 人では捕らえられず骨折り損になります。ウサギは 1 人でも捕らえられます。鹿は 10 の、ウサギは 3 の価値があり、一日働くと 1 の費用がかかります。このとき二人の利得は次のようにまとめることができます。

		相　棒	
		鹿	ウサギ
自分	鹿	4 ＼ 4	-1 ＼ 2
	ウサギ	2 ＼ -1	2 ＼ 2

表 3　鹿狩りゲーム：利得表

このゲームは基本的には利得を書き換えるだけで可能です。ステージツリーのバックグラウンド内のスパナ型のアイコンを見てください。

これはプログラムアイコンと呼ばれ、数値の設定や計算でこのアイコンを使います。これをダブルクリックしてください。中に書かれたプログラムを編集できるダイアログボックスが開きます。

図 24　プログラムアイコン

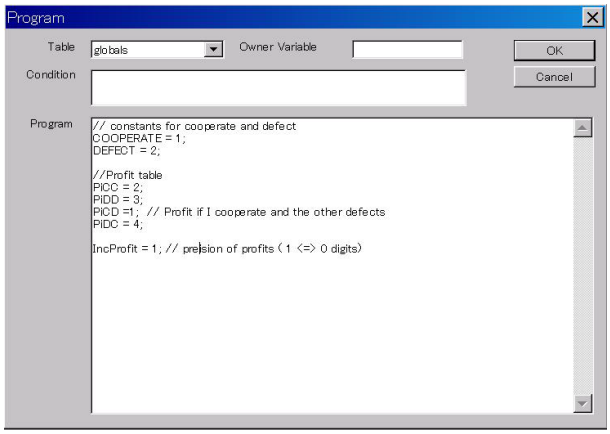


図 25　プログラムダイアログボックス

この中で注目すべきは Pi ●○＝数値；となっている 4 つの行です。これは自分が●、相手が○を選択した場合の自分の利得を表します。例えば PiCD＝１；は自分が C（協力）、相手が D（裏切り）を選択した場合の利得が 1 であることを表します。鹿狩りゲームでは C を鹿、D をウサギに読み替えて数値を変えることにしましょう。すると利得は次のように書き換えればよいことになります。

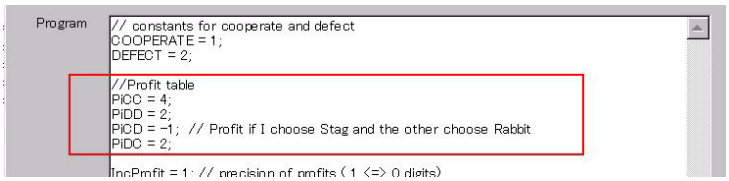


図 26　利得表の設定プログラム

「//」より右の部分はコメントと言ってプログラムにつけることが出来る注釈です。実行内容には影響しないのでここではあまり気にする必要はありません。

英語を日本語に直したときの要領で入力や結果表示のラベルをゲームの内容に合わせて変えれば「鹿狩りゲーム」のトリートメントプログラムの出来上がりです。出来上がりを図 27 に示します。

出来上がったら保存してテストランをしてみてください。利得表どおりの結果が出るなら成功です。

このゲームの更なる応用として「両性の戦い」があります。「両性の戦い」は次のようなゲームです。

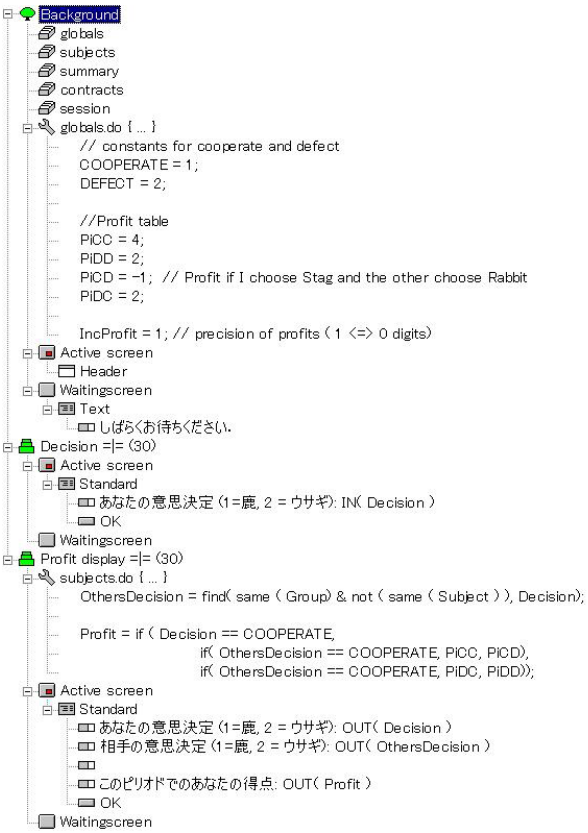


図 27　鹿狩りゲーム

両性の戦い（Battle of Sex）

マイケルとメアリーはデートにボクシングを見に行くか、バレエを見に行くか相談しています。マイケルはボクシングが好きなのでバレエはあまり楽しくありません。メアリーはバレエが好きなのでボクシングはあまり楽しめません。では二人が別々にボクシングとバレエを見に行くことにしたら？これは二人にとってはデートと言う最も重要な目的を果たしていないので、二人はまったく楽しくないことになります。この二人の選択肢と満足の度合いは次のようにまとめることが出来ます。

		女	
		ボクシング	バレエ
男	ボクシング	2 ＼ 1	0 ＼ 0
	バレエ	0 ＼ 0	1 ＼ 2

表 4　両性の戦い：利得表

アメリカでは性差別に敏感な人たちに配慮しているのか、男性がボクシング、女性がバレエを好むといういわゆる典型的な男女の好みの差を明確に出さないストーリーで説明されることが多くなっているようです。

このゲームをプログラムしようとすると男と女で利得が異なるのでプレイヤーを 2 つのタイプに分け、それぞれ違った利得を定義してやる必要が出てきます。その技法はここでは扱わないことにしますが、どう実現するかはチュートリアルマニュアルの 3.3.2 から 3.3.6 にありますので興味のある方はトライしてみてください。

6　１からのプログラム

非常に基本的なプログラムを一つ、１から作ってみましょう。ここではまず期待効用仮説を確認する実験のプログラムを作ります。つぎにアレのパラドックスと呼ばれる、期待効用仮説に対する反例として有名な事象を検証するプログラムを作って見ましょう。

期待効用仮説

ここに 2 つのくじ A と B があるとしましょう。A はひくと確実に 1000 円が当たります。B はひくと 50 %の確率で 2000 円が当たり、50 %の確率で 0 円になります。どちらかのくじを選ぶとしたらみなさんはどちらを選びますか？

期待効用仮説によると人は B より A を選択すると考えられます。くじを引く前の状況を考えると、A は確実に 1000 円を得られるのでくじから期待できる所得は 1000 円です。B は 0.5 の確率で 2000 円、また 0.5 の確率で 0 円なのでくじから期待できる所得は 0.5 × 2000 + 0.5 × 0=1000 円でくじ A と変わらないはずです。

それでもくじ A の方が好まれるのは、不確実な所得から期待できる効用 (満足) よりも確実な所得から得られる効用の方を高く評価する、危険回避の傾向が人々にあるからだと説明されます。

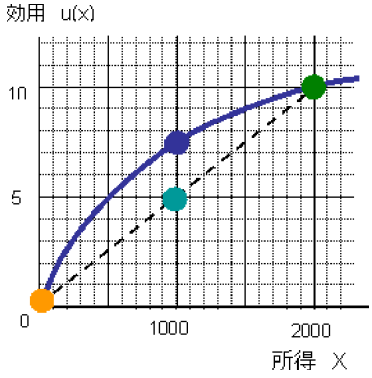


図 28　期待効用仮説

この図はこのような所得と効用の関係を横軸を所得、縦軸を効用にしたグラフから説明するものです。所得が 0 のときの効用を u(0)、2000 のときの効用を u(2000) とします。u(0) は図上の黄色の点、u(2000) は図上の緑の点で表されます。それらがそれぞれ 0.5 の確率で得られるので、くじ B の効用は 0.5u(0)+0.5u(2000) となります。これは図上の水色の点に相当します。危険回避傾向は、図上での青色の点にあたる、確実に 1000 円が得られるときの効用 u(1000) がこの水色の点より上にあるという形で表されます。

実際に上記の例のような選択肢が与えられた場合、人々は期待効用仮説に従って行動するでしょうか。実験のためのプログラムを作ってみましょう。

6.1　準備

- z-Tree を起動します。

- Untitled Treatment 1 というウインドウが開いているはずです。右上の×を押して消してください。
- メニューから　Treatment → Language → Nihongo　と選択してください。
- メニューから　File → New Treatment　を選択してください。
- 新しく開いたウインドウの一番下の行が「しばらくお待ちください」になっていたら OK です。

z-Tree で実験者が作るプログラムはステージツリーと呼ばれます。実験者はエレメントと呼ばれる、画面表示や入力の受付、計算などを実現するステージツリーの部品を実験の流れに沿って配置することでステージツリーを作ってゆきます。ステージツリーの頭にはかならずバックグラウンドと呼ばれる木の形のアイコンがあります。ここで参加人数や被験者の初期条件など、実験全体にかかわる設定を行います。New Treatment を開くとこのバックグラウンドが自動的に生成されます。

6.2 数値の初期設定をする

くじ A と B の賞金の金額をバックグラウンドで設定しておきます。数値の設定や計算はプログラムというエレメントを適切な位置においてそこに書き込みます。

1. Background の session をクリックします。メニューから Treatment → New Program を選択するとプログラムを入力するダイアログボックスが開きます。
2. くじ A の賞金を LotA, くじ B の当たった場合の賞金を LotBWin, 外れた場合の賞金を LotBLose とします。Program のボックスの中に次のように書き込みます。

```
LotA=1000;
LotBWin=2000;
LotBLose=0;
```

3. 入力が済んだら「OK」を押してダイアログボックスを閉じます。

あとで賞金額を変えたい場合はこの部分だけ変えればよいことになります。LotA や LotBWin,LotBlose のような中に数値を代入する文字列を変数といい、変数に数値を代入する式を割り当て構文といいます。割り当て構文を書くときは最後にセミコロン「;」を付けなくてはなりません。

バックグラウンドに初期設定を集めて記述しておくのは、その方があとで実験の設定を変えたいときに便利だからです。

これで基本の設定が出来上がりました。プログラムはステージツリー上ではスパナの形のアイコンで表示されます。プログラムダイヤログボックスは今後もよく使いますが、同時にエラーで悩まされる事も多くなるでしょう。エラーが出たら 6.5 章を参照してください。

6.3 入力ステージを作る

被験者にはくじ A かくじ B かを選んでもらう必要があります。そのためのステージを作りましょう。

1. Background の木の形のステージをクリックします。メニューから Treatment → NewStage を選択すると新しいステージが追加され、その設定を決めるダイアログボックスが開きます。設定はデフォルトでいいので name のところだけ InputDecision と変えておきましょう。

注意:メニューに NewStage が現れない場合は、Background がクリックされているかどうかを確認してください。Background の「中」のエレメントがクリックされているときには、Background の「中」に追加できるものしかメニューには出ません。Background と Stage はステージツリーの中で一番大きいプログラム上の単位です。Background の「中」に Stage は作れません。Background そのものをクリックしてあれば、Stage は Background の下に挿入されます。Stage の下に新たに Stage を作る場合も、かならず新しい Stage のひとつ上のステージアイコンをクリックしてから挿入してください。

2. Active screen をクリックします。Treatment → New Box → Standard Box を選択します。Standard box のアイコンがステージツリー上に挿入されます。 z-Tree の画面はボックスと言う単位で構成されます。Standard Box は最も基本的なボックスです。
3. Standard box をクリックします。Treatment → New Item を選択します。ダイアログボックスには図 29 のように入力します。画面に何か文章を表示したいときはこのアイテムの Label 欄に書き込みます。Label の欄に「くじ 1 は確実に 1000 円が当たります。くじ 2 は 50 %の確率で 2000 円があたり、50 %の確率で 0 円が当たります」と書き込んでおきましょう。

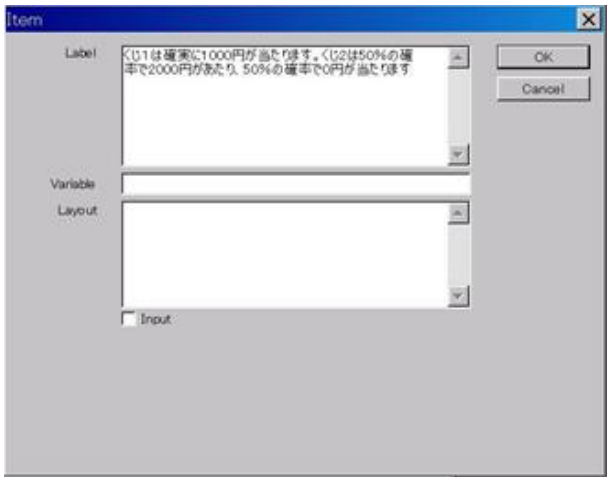


図 29　アイテムダイアログボックスをメッセージ表示に使用する

4. 被験者が自分の決定を入力するためのアイテムを作ります。まず今作ったアイテムをクリックします。Treatment → New Item を選択すると新しいアイテムが作られます。ダイアログボックスの「Input」の項目にチェックを入れ、あとは図 30 のように入力します。ダイアログボックスの項目は次のようになっています。

   Variable　変数。ここでは Decision としているので被験者が入力した数値がこの Decision に代入されます。

   Layout　桁数。1 となっている場合、入力が許されるのは 1 の倍数だけです。0. や 1.33 といった小数点以下の数値を入力しようとするとエラーメッセージが出ます。

   Input　このチェックボックスにチェックを入れるかいれないかで、このアイテムをメッセージや数値を表示する出力アイテムにするか、入力を受け付ける入力アイテムにするかを決めることが出来ます。

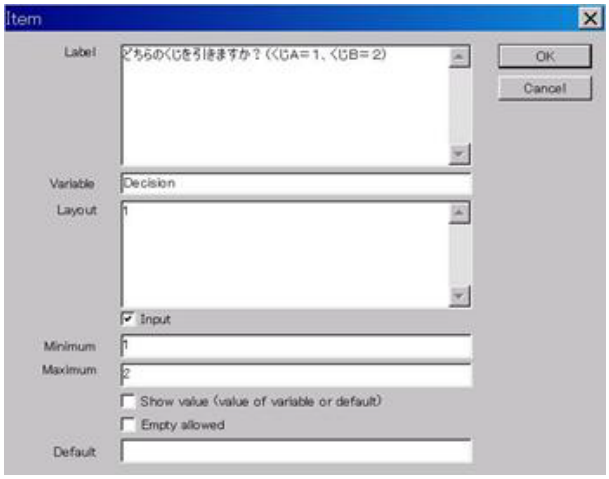


図 30　アイテムダイアログボックスを入力受け付けに使用する

Minimum　入力が許容される最小値をここで設定します。

Maximum　入力が許容される最大値をここで設定します。

5. ステージの最後には被験者がそのステージでの入力や結果の確認が終わって、次の画面に行くために押すボタンをつける必要があります。（このボタンがないと基本的に次の画面に行けません。）メニューから Treatment → New Button を選択します。ボタンのダイアログボックスがでてきます。そのまま OK を押してください。

これで入力ステージは完成です。次に結果を計算して表示するステージを作りましょう。

6.4　結果の計算と表示

被験者がくじ 2 を引いた場合は確率 50 %で 2000 円、50 %で 0 円を提供しなくてはなりません。このためのプログラムを書きましょう。

- 新しいステージを挿入します。Input Decision のステージのアイコンをクリックし、メニューから Treatment → New Stage を選択します。ダイアログボックスではステージの名前を ShowResults にしておきます。
- この新しいステージをクリックし、Treatment → New Program を選択します。

くじは次のような形で実現させます。

1）0 から 1 までの乱数（でたらめな数）を発生させる。

2）乱数が 0.5 より大きいならくじ 2 の結果は LotBWin にし、0.5 以下なら LotBLose にする。

まず 1）から作ってゆきましょう。z-Tree には random() という関数があります。これは 0 から 1 までの乱数を発生させます。出来たでたらめな数は Random という変数に代入することにします。従ってプログラムダイアログボックスに次のように書きます。

```
Random = random();
```

この数値が 0.5 より大きかったらくじ B の結果を LotBWin にし, 低かったら LotBLose にするわけですが、このような条件判断には if 構文を使います。if 構文は次のような構造を持ちます。

```
if( 条件 ) { 割り当て構文１} else { 割り当て構文２} ;
```

条件が満たされていれば割り当て構文１が実行され、そうでなければ割り当て構文２が実行されます。else 以降を省略することも出来ます。その場合は条件が満たされていれは構文 1 が実行され、条件が満たされていなければ何もおきません。

if 構文を使ってくじ B の結果を決めましょう。くじ B の結果を LotB とします。

```
if (Random>0.5) {LotB=LotBWin; } else { LotB=LotBLose; }
```

これで Random の値が 0.5 より大きいなら LotB には LotBWin の値が代入され、そうでないなら LotBLose の値が代入されます。

次に引いたくじによって得点を決めてやります。もし被験者がくじ A を引いているなら被験者の得点 (Profit) に LotA を、くじ B を引いているなら被験者の得点に LotB を代入してやればいいわけです。これも条件文ですから同じようにしてやれば OK です。

```
if (Decision==1) {Profit=LotA; } else { Profit=LotB; }
```

Decision==１のところでイコールが 2 つ重なっていることに注意してください。割り当て構文の中ではイコール（＝）は「右辺の値を左辺に代入する」という意味で使われます。一般的な数学での意味である「左辺と右辺が等しい」という意味では用いられません。if 構文の条件の中で左辺と右辺が等しいときという条件を作るためにイコールを使いたいときは、イコールを二つ重ねる（＝＝）必要があります。

この行ではもし被験者の入力（Decision）が１だったら被験者の利益（Profit）には LotA の値を代入し、そうでなければ LotB の値を代入します。

ちなみに割り当て構文の中に条件を入れることも出来ます。そのときの構文は次のようになります。

```
変数＝ if（条件，値１，値 2）;
```

このように書くと、条件が満たされていれば変数に値１が代入され、満たされていなければ値 2 が代入されます。従ってさっきの行は次のように書いても結果は同じです。

```
Profit=if(Decision==1,LotA,LotB);
```

適宜使いやすいほうを使ってやればいいでしょう。

こうして決まった被験者の得点を表示する画面を作りましょう。Active screen をクリックしてメニューから Standard box を挿入し、その下に Item を挿入します。手順は 6.3 章の入力ステージと同じです。ただし今度は結果の表示だけなので Input チェックボックスにチェックを入れる必要はありません。Label の欄に説明を、Variable の欄には表示させたい変数である Profit を、Layout の欄に１を入れます。同様にして被験者の意思決定を表示するアイテムも作って見ましょう。最後にメニューからボタンを呼び出して挿入してやれば完成です。完成すると図 31 のようなステージツリーが出来上がるはずです。テストランをしてうまく動くか確認してみてください。今回は被験者同士の相互取引がないので z-leaf は一つ立ち上げれば十分です。

図 31　期待効用仮説検証プログラム

6.5　エラーが出たら

プログラムボックス内でプログラムを書いて「OK」を押すとエラーが出る場合がありますが、原因はほとんどの場合文法が間違っているか、関数が未定義であるかのどちらかです。

6.5.1　文法エラーの場合

文法が間違っている場合、次のようなエラーメッセージが出ます。

```
syntax error;
```

この場合次の事をチェックしてみてください。

- 各行の最後はセミコロン「;」になっているか
- 括弧やカンマ、イコールが全角になっていないか（日本語入力のまま数式を入力していないか）
- 代入の「=」と左辺右辺が等しいという条件の「==」を間違えていないか
- ≦や≧は「<=」や「>=」と書くべきところを「=<」や「=>」と書いていないか
- 関数のつづりを間違っていないか

手本どおりに入力しているはずなのにエラーが出る場合は大抵上記のような単純な入力ミスです。注意して見直してみてください。

文法の様式は全てのリストがリファレンスマニュアルの 7 章、44 ページに載っています。自分でプログラムを書くとき文法に自信がない場合はそちらを参考にしましょう。

6.5.2 変数未定義のエラー

変数が未定義だと言うエラーもよく出ます。この場合、次のようなエラーメッセージが出ます。

```
Unknown variabler; XXXXXX
```

これは、あなたが計算に使えといっている XXXXX はまだ中身が定義されていないので計算に使えない、というメッセージです。もし単に初期値を設定し忘れているのであれば、そのプログラムより前のところに XXXXXX=10; などのように数値を代入してやるプログラムを書く事で解決します。しかし手本どおりに入力しているはずで、事前に定義もしているはずのにエラーが出る場合があります。これはたいていの場合

- つづりの間違い。「Decision」を「Desicion」と書き違えるなど。
- 大文字・小文字の不統一。変数名は大文字小文字の差を認識します。例えば「Profit」と「profit」、「LotBWin」と「LotbWin」はそれぞれ違う関数とみなされます。

どちらも既に定義してある変数とちょっと違うつづりになっているのに気がつかなかったり、大文字小文字が違っているのに気がつかないと生じるエラーです。起きないはずのエラーが起きるとき、ほとんどの場合それはごく小さなミスに起因します。注意して見直してみてください。

6.5.3 どうしても原因が分からないとき：コメントアウト

エラーが出ているけれども、どこが間違っているのか分からないということが時々あります。このような時はコメントアウトを使って間違っている箇所を絞り込みます。プログラムダイアログボックス内では「//」と書くとそこからその行の終わりまではプログラムではなく、コメントであるとみなされ、z-Tree は実行のときこの部分を無視します。コメントは通常自分が書いたプログラムの内容を忘れないようにするメモ書きとして使うものですが、z-Tree がコメントを無視するという機能を使う事でエラーのある場所を絞り込む事ができます。例えば次のプログラムはどこかが間違っているのでエラーがでてしまいます。

```
Random=round(random()*100,0)+1; //random from 1 to 100
if (Random>Boundary) { Lot2=Lot2Win; } elese { Lot2=Lot2Lose; }
Profit=if(Input==1,Lot1,Lot2);
```

どこが間違っているか絞り込むためあやしいと思う行の最初に//を入れてコメント化します。これをコメントアウトといいます。どこがあやしいかも見当がつかない場合は最後の行からコメントアウトしていって見ましょう。（最初の行からやると別の問題を引き起こしやすくなります、例えばこの例では 1 行目で定義している変数 Random を 2 行目で使うため、1 行目をコメントアウトすると文法エラーのほかに変数未定義エラーが出てしまうので面倒です。）

```
Random=round(random()*100,0)+1; //random from 1 to 100
if (Random>Boundary) { Lot2=Lot2Win; } elese { Lot2=Lot2Lose; }
//Profit=if(Input==1,Lot1,Lot2);
```

これで「OK」ボタンを押してみます。まだエラーがでるなら下から 2 行目もコメントアウトしましょう。

```
Random=round(random()*100,0)+1; //random from 1 to 100
//if (Random>Boundary) { Lot2=Lot2Win; } elese { Lot2=Lot2Lose; }
//Profit=if(Input==1,Lot1,Lot2);
```

ここでエラーが出なかったら、この行があやしいと言うことになります。よく見直してみると、「else」を「elese」と書き違えている事に気づくでしょう。このようにして間違っているところを絞り込んで直します。

文法の誤りだけでなく、プログラムを書きかけのままいったんダイアログボックスを閉じて別の箇所を確認したいというようなときも、このコメントアウトが使えます。

6.6 アレのパラドックス

さらに応用としてアレのパラドックスを確認する実験を作って見ましょう。

アレのパラドックス

つぎのような 2 つのくじがあるとします。これをセット X と呼びます。

くじ 1：確実に 1000 円を得る

くじ 2：1 %の確率で 0 円、10 %の確率で 5000 円、89 %の確率で 1000 円を得る。

さらに次のようなくじを考えます。こちらはセット Y と呼びましょう。

くじ 3：11 %の確率で 1000 円、89 %の確率で 0 円

くじ 4：10 %の確率で 5000 円、90 %の確率で 0 円

あなたはセット X ではどちらのくじを選び、セット Y ではどちらのくじを選びますか？

アレのパラドックスについての詳しい説明は実際に実験をしてみてからの方がよいかもしれません。とりあえずこの二つの実験のプログラムを考えましょう。

セット X で悩ましいのは確率によって結果が三つあるということですね。それぞれの結果を LotB1=0、LotB2=5000、LotB3=1000 として、Random が 0.01 以下なら LotB1、0.01 より大きくて 0.11 以下なら LotB2、0.11 より大きいなら LotB3 を入れてやればよいわけです。この場合結果が 3 つあるので先ほどのように 1 行で結果を書き表すことができなさそうです。そこで if 文の構造を次のように 3 行にしてやります。

```
if (Random<=0.01){LotB=LotB1;}
if (Random>0.01 & Random<=0.11){LotB=LotB2;}
if (Random>0.11){LotB=LotB3;}
```

プログラムでは「≦」を「<=」と書き、「≧」を「>=」と書くことに注意してください。また 2 行目の if 文のように、条件を&で繋いで二つの条件が同時に成立するときのみ実現する内容を作ることができる点にも注意してください。

セット Y に関してはどちらのくじを引いても結果が二つあるわけですから、LotA の結果も LotB と同じようにプログラムしてやる必要があります。

このセット X とセット Y に関しては課題と言うことにしましょう。各自プログラムしてみてください。入力ステージを二つ作り、それぞれセット X とセット Y の意思決定を入力させます。どちらも入力のステージですが同じステージ名にならないように InputDecisionX と InputDecisionY のように、それぞれ違う名前を

つけてください。意思決定の入力が代入される変数もそれぞれのセットごとに違う変数を用意してやる必要があります。セット X の意思決定は DecisionX、セット Y の意思決定は DecisionY などとしてやりましょう。ProfitDisplay のステージでは両方の結果を計算しましょう。その際にも得点を表す変数が二つ必要ですのでX の得点は ProfitX、Y の得点は ProfitY として、それぞれ計算して表示させてください。最後に

```
Profit = ProfitX+ProfitY;
```

と言う行を付けて総得点を出してやればいいでしょう。これも最後に表示させましょう。プログラムができたらテストランをしながら、自分ならどのような選択をするか、よく考えてみてください。

7　被験者の管理：被験者番号とグループ分け

沢山の被験者を相手に実験をする場合、実験者は被験者に被験者番号をわりふり、適当な大きさのグループに分ける必要があります。また被験者間でのゲームを行うためには同じグループ内の適切な相手と情報や戦略をやり取りさせることも必要です。z-Tree でこれらをどうやって実現するかを、公共財実験を下敷きにしながら確認してゆきましょう。

公共財実験

ここに５人の村人で構成される村があるとしましょう。その村では人々は手持ちの資源を使って所得を得ます。ここでは 1 日 8 時間働くことができるという、8 時間分の労働力を手持ちの資源としましょう。村人はその資源を自分のためだけに使うこともできますし、公共財と呼ばれる、その村に住む人全員にとって役に立つなにか（例としては村の観光収入が増えるように村の美化のために働いたり、村の財政に補助金を出してくれるように国に陳情するなどの活動が考えられます。）を実現するためにも使うことができます。

その村では自分のために 1 時間働くと 1,000 円の収入が得られます。公共財のために 1 時間使うと村全体では 1,500 円の収入増になります。しかし公共財からの収入は個人の懐に直接入るわけではなくて、村全体での収入なのでその 1,500 円を村の構成員全員で割った値が個人の収入となります。村人が置かれている状況は次の式のように整理できます。

収入＝ 1000 × (8 −村のために使った時間) + 1500 × (村人が村のために使った時間の合計) ÷ 5

１項目は手持ちの 8 時間から村のために使う時間をひいた残りに 1,000 円をかけたもので、これは自分のために時間を使って得られる収入です。2 項目は村人全員の働きから得られる自分の収入です。この村人にとって手持ちの 8 時間をどう使うのが賢い選択でしょうか。

- もし 5 人全員が 8 時間を自分のために使ったら、皆の収入はそれぞれ、$8 \times 1{,}000 = 8{,}000$ です。
- もし 5 人全員が 8 時間を村全体のために使ったら、延べ 40 時間が村全体のための活動に使われます。つまり村全体では $40 \times 1{,}500 = 60{,}000$　の収入があります。これを 5 人で分けると皆の収入はそれぞれ 12,000 です。

こうしてみると村全体のために全ての時間を使うのがよい選択と思われるでしょう。しかしそうでしょうか。今、手持ちの 1 時間を収入に変えるとき、自分のために使うなら 1,000 円が確実に手に入ります。しかし村全体のために 1 時間を使うとき、確実に手に入るのは自分の働きで増える村の収入 1,500 円を人数分で割った 300 円だけです。他の人が村のために働いてくれず、自分だけが村のために 1 時間使った場合、その 1 時間分の労働で 1,000 円稼げたはずがたった 300 円になってしまうのです。あなたはこの状況で、手持ちの 8 時間をどのように自分のためと村のために分配しますか？

公共財実験は被験者実験でもよく行われており、その研究成果は人間が必ずしも理論的な予言に従って合理的に行動しない例の宝庫になっています。まずは z-Tree を使ってこの実験のプログラムを作りましょう。プログラム自体は 42 ページに載っています。確認しながら進めてください。

7.1 被験者管理とグループ分け

被験者番号

この実験を可能にするためには被験者を 5 人ずつのグループに分ける必要があります。グループ分けをするためには z-Tree から見て被験者の一人一人の区別がつくように被験者に識別番号をつけてやる必要があります。

被験者に番号を割り振る作業は z-Tree が自動的に行います。z-Tree を起動した後で z-Leaf を起動すると両者の間の接続が行われますが、1 番最初に接続した z-Leaf に z-Tree は 1 という被験者番号を与え、それを Subject という変数に代入して保持します。つまりその席の被験者の被験者番号は 1 で、それは Subject という変数をみることで確認できます。2 番目に接続した z-Leaf の Subject 変数の中身は 2 になります。以降 3 番目 4 番目も同じです。ですから A 君の席の被験者番号を 1 に、B 君の席の被験者番号を 2 にしたい場合は z-Leaf の立ち上げの順番に気をつける必要があります。

ただし一旦立ち上げたからといってもう変えられない訳ではありません。被験者番号はメニューの Run → Client ' s Table で見られるクライアントテーブルに並んでいる順でもあります。一番上が被験者番号 1、次が被験者番号 2 となります。テーブル上のクライアントをドラッグ&ドロップしたり、メニューからソートを選んで並べ替えると被験者番号も連動して変わります。

グループ分け：ダイアログボックスを使う

被験者を適当なグループに分けるには BackGround のアイコンをダブルクリックして出てくるダイアログボックスを使います。Number of groups の欄に数値を入れることでその数に相当する人数のグループに分けられます。もし人数に端数が出る場合はその端数になった人たちだけで一つのグループを作ります。グループ番号は Group という変数に格納されます。これがマニュアルどおりの手続きですが、（なぜか）うまく機能しないこともしばしばあります。従って**グループ分けでは次に説明するようなプログラムによるグループ分けを行うことを強くお勧めします**。

グループ分け：プログラム上でグループ分けを行う。

バックグラウンドでのグループ分けが機能しないことがあるので、プログラム上でグループ分けを行うほうがより安全です。グループ分けのためには各被験者のもつ Group という変数に適当なグループ番号を代入してやればよいのです。プログラム上ではそれは図 32 のようなプログラムエレメントを追加する事で実現させます。

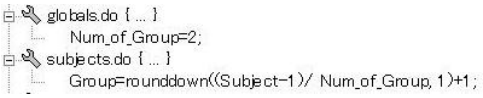


図 32　プログラム上でグループ分けをする

このプログラムの意味を説明しておきましょう。まずひとつ目のプログラムで 1 グループ内の人数を `Num_of_Group=`に代入してやります。これは全ての被験者にとって同じ条件のものですから Grobal テーブル上で定義します。次のプログラムはちょっと難解に思えるかもしれません。

```
Group=rounddown((Subject-1)/ Num_of_Group, 1)+1;
```

これは被験者に被験者番号の若い順にグループ番号を割り振っています。実際の例を見ながら確認していただくのがいいでしょう。

今 6 人の被験者を 2 人ずつのグループに分けたいとします。被験者に割り当てられた番号 Subject は 1 から 6 です。

Subject	1	2	3	4	5	6

そこから 1 を引きます。

Subject-1	0	1	2	3	4	5

あらかじめ 2 を代入してある Num_of_Group で割るとそれぞれこうなります。

(Subject-1)/ Num_of_Group	0	0.5	1	1.5	2	2.5

関数 rounddown(数値, 1) を使って数値が 1 の倍数になるように小数点以下を切って丸めます。

rounddown((Subject-1)/ Num_of_Group, 1)	0	0	1	1	2	2

グループ番号は 1 からなので 1 を足しておきます。

rounddown((Subject-1)/ Num_of_Group, 1)+1	1	1	2	2	3	3

こうしてできた数値を Group に代入することで被験者番号の若い順に 2 人ずつのグループ分けが実現するわけです。

今ここの例ではテストランをしやすいように 2 人ずつにグループ分けをしていますが、5 人ずつに分けたい場合は `Num_of_Group=5;` と直します。

7.2 グループ内の相手の選択を見つけ出す：テーブル関数の活用

この実験では各被験者の公共財への貢献量は変数 Contribution へ代入されます。それぞれの被験者の報酬を計算するためには村全体でどれだけの資源が公共財に使われたかを知る必要があります。当人以外の被験者の変数（ここでは Contribution）を見つけるためにはテーブル関数（Table Function）と呼ばれる特殊な関数を利用します。テーブル関数の全リストは 9 章にもありますし、マニュアルのリファレンスの 47-48 ページにもありますが、主なものを下表 5 に紹介しておきます。

sum(条件、変数)	条件を満たす変数の合計
find(条件、変数)	条件を満たす変数の値
count(条件)	条件を満たす被験者の数
maximum(条件、変数)	条件を満たす変数の中で最大の値
minimum(条件、変数)	条件を満たす変数の中で最小の値

表 5　主なテーブル関数の例

テーブル関数の使用例を説明しましょう。例えばある被験者から見て自分と同じグループの被験者の Contribution の合計を SumC に代入したいときには、プログラムアイコンに

```
SumC=sum(Group==:Group, Contribution);
```

と入れます。ここで肝心なのが Group ==**：Group** という条件です。太字の **Group** の前のコロン：に注意してください。これはスコープオペレーターと呼ばれます。スコープオペレーターの厳密な意味合いはマニュアルに譲りますが直感的には「自分の」という意味だと考えてもらえばいいでしょう。つまり**：Group** は「**自分の Group 番号**」という意味です。sum(Group==:Group, Contribution) という条件を与えられると z-Tree は全ての被験者の変数を見てまわり、「当の被験者のグループ番号と同じグループ番号」という条件を満たす被験者の Contribution を（本人の分を含めて）全部拾ってきて合計を計算し、SumC に代入します。

もし同じグループで最も公共財への貢献量が少ない人の貢献量を見つけてきて DOKECHI という変数に代入したい場合は次のようにすればよいわけです。

```
DOKECHI=minimum (Group==:Group, Contribution);
```

この条件で拾ってくる数値には自分の数値も含まれるので、本人の貢献量がグループで一番小さかった場合にはその本人の貢献量が DOKECHI に代入されます。　「自分の変数と同じ変数をもつ」という条件ははあまりによく使われるため、いちいちスコープオペレーターを使って条件を指示するのは面倒です。このため same(変数) という関数が作られました。treatments のフォルダに入っているお手本のプログラムもほとんどの場合 same(変数) を使っています。　このプリントのお手本でもグループの貢献量の合計は

```
SumC=sum(same(Group), Contribution);
```

と言う形で記述されています。この関数を使ってもスコープオペレーターを使っても同じ結果になりますのでこの違いは単に書きやすさや分かりやすさの問題です。お手本には他に

```
N=count(same(Group));
```

というテーブル関数も使われています。これは同じグループ内に自分も含めて何人いるかを勘定して N に代入するテーブル関数です。これも same(変数) を使わずに

```
N=count(Group==:Group);
```

と書いても同じ結果になります。

ここまで説明したところで以前学んだ囚人のジレンマのプログラムを見返してみてください。相手の戦略を見つけてきて OthersDecision に代入するのにどのようなプログラムが書かれているでしょうか。当の部分を抜書きするとこのようになっています。

```
OthersDecision = find( same ( Group) & not ( same ( Subject ) ), Decision);
```

この式の意味するところは自分と同じグループで、自分のではない Decison を見つけて OthersDecision に代入しろ、ということだというのが見て取れると思います。参考までにこれを same 関数を用いずに書くとこのようになります。

```
OthersDecision = find( Group==:Group & Subject<>:Subject , Decision);
```

囚人のジレンマの場合は 1 グループに 2 人しかいないので、同じグループでかつ自分ではないとする条件をつけることで相手を特定できました。しかし 3 人以上いる場合に特定の誰かの変数を見つけたいと思ったら、

グループ内でのメンバー番号を作って割り振ってやるなどする必要があるでしょう。例えば村にリーダーがいて、その人の公共財への貢献は他の人の 2 倍の成果をだすなどという設定を実現したいと思ったら、一工夫が必要になります。余力があれば考えてみてください。

図 33　公共財実験のプログラム

8　参考資料 1：変数の扱い

ここまでで基本的な事は一通り学んだ事になります。より進んだ内容についてはチュートリアルマニュアルを参照しながら学習を進めてください。3 章 4 節を見ながら画面を見やすく変更したり、入力をスライドバーやラジオボタンでできるようにしてみるのもよいでしょう。ただ 3 章 2 節や 3 章 3 節のところはとっつきにくく、特にスコープオペレータの箇所は分かりにくいのではないかと思います。スコープオペレータの件は無理に理解しなくとも 7 章の説明とこの章の内容を理解しておけばまったく問題ないでしょう。その他チュートリアルマニュアルでは分かりにくい点や説明が不十分な点をここでまとめて解説しながら補います。

8.1　テーブルの概念と変数の属性

8.1.1　テーブル

z-Tree でのプログラミングは、ステージツリーと呼ばれる流れ図を画面上で組み立てる形で行います。流れ図の構成要素はエレメントと呼ばれます。変数を扱う場合、プログラムエレメントと呼ばれるエレメント（スパナの形のアイコンです）を適切な箇所に挿入し、そのダイアログボックス内にプログラムを記述する形で行います。このダイヤログボックスにテーブルを指示するプルダウンメニューがあり、実験者は必ずどのテーブル上で変数を扱うのかを決めなくてはならなりません。

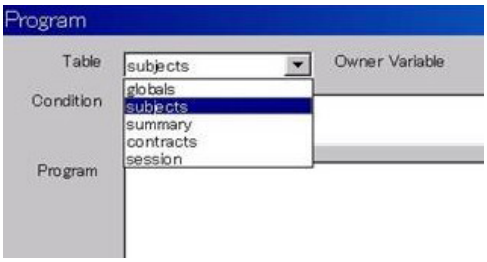


図 34　テーブルの選択

どのテーブルにするかでプログラムの働きが違ってくるので注意が必要です。（とはいえ一般的には Subjects テーブルと Globals テーブルをしっかり理解しておけば大丈夫でしょう。）

8.1.2　Subjects テーブル

Subjects テーブルは被験者ごとに割り当てられるレコードと呼ばれる小単位で構成されます。それぞれの被験者の変数はそのレコードの中に置かれます。z-Tree が変数を計算するとき、z-Tree はその瞬間計算しているレコード内から見える変数しか見ません。他の被験者のレコード内の変数は見えないので、同じ変数 Score でも被験者番号 1 の被験者のレコード上の Score と被験者番号 2 の被験者のレコード上の Score は別の変数とみなされ、個別に計算されます。このおかげで被験者が何人いても原則的に一人分のプログラムを書くだけでよいわけです。

この仕組みは他のプレイヤの変数を見に行けないと言う問題を生じさせます。ゲームの結果を出すためには自分（当該プレイヤ）の意思決定の他に相手プレイヤの意思決定の情報が必要ですが、Subjects テーブルのあるレコード上で見つけられる変数は当プレイヤのレコードの変数だけなのです。相手プレイヤの意思決定を知

るにはテーブル関数と呼ばれる関数を介する必要があります。これについては後述します。

8.1.3 Globals テーブル

Globals テーブルは、生産関数のパラメータや初期保有など全被験者にとって共通の変数を定義するために用います。このためグローバルテーブルに含まれるレコードはひとつだけです。

8.1.4 Session テーブル

Subjects テーブルや Globals テーブルで定義された変数は、1 回の試行が終了すると（一般的にはステージツリーの先頭から最後までを経過すると）変数は全て記録された上でリセットされます。複数回の試行の間リセットされない変数が必要な場合、Session テーブル上で定義します。ここで注意すべきなのは Session テーブルに書かれたプログラムは毎回実行されることです。複数の回数にわたって保持したい値がある場合次の回にまたプログラムが実行されてに上書きされてしまわないようにしなければなりません。

8.1.5 各テーブルの実験実施における機能

Subjects, Globals, Session の 3 つのテーブルは上述のように 1）いつ変数がリセットされるか 2）各被験者が個別のレコードを持つか否か、の 2 つのベクトルから区別されます。

3 人の被験者が 3 ピリオドの実験を行う場合のテーブルの模式図を図 35 に示します。

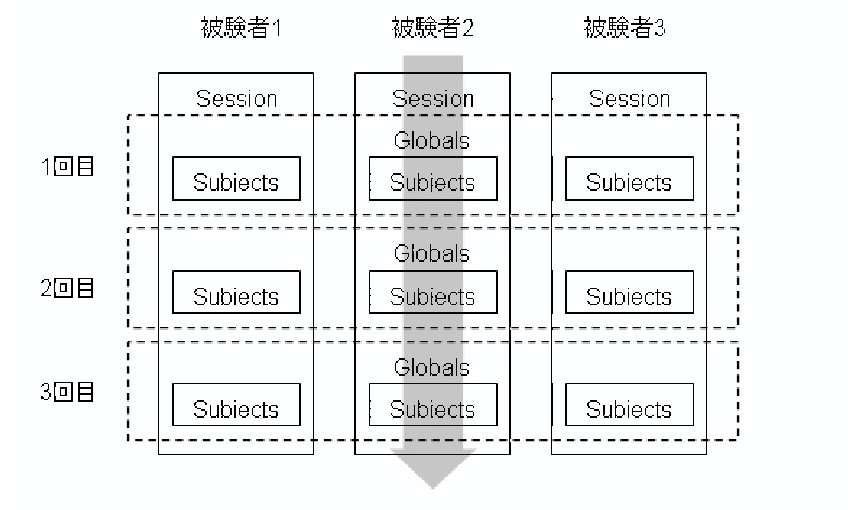


図 35　テーブル概念図

この図において箱は点線、実線にかかわらずそれぞれ個別のレコードを表します。Subjects テーブルでは各被験者にレコードが用意されるのでピリオドごとにレコードは 3 つ用意され、被験者共通である Globals テーブルではレコードはひとつ用意されます。ピリオドが変わるとそれらはリセットされ、同じものがまた用意されます。Session テーブルは被験者ごとにレコードが用意され、その実験が終了してアンケートが終了するまでリセットされません。レコードは 3 ピリオドの実験を行うこの例では全部で 15 個あるということになります。

それぞれのレコードは独立しているため、同じ変数名をそれぞれのレコードで使っても別の変数として扱わ

れます。

図では Subjects テーブルのレコードの箱が Globals テーブルや Session テーブルの箱の中に入っていますが、これは Subjects テーブルが Globals テーブルや Session テーブルの一部であるという意味ではなく、Subjcet テーブルの各レコードを基準としてみたときの変数の共有を表しています。例えば同一のピリオドにおいて各被験者は Globals テーブル上の同じ変数を共有しています。同様にある被験者にとってはセッションが終了するまで Session テーブル上の変数はどのピリオドでアクセスしても（代入された数値は変わっているかもしれませんが）同じ変数です。

また、この箱はテーブルのデータがどの時点でリセットされるかも表します。灰色の矢に沿って実験が経過するとしましょう。実験が開始され、第 1 回目の試行に入ると（矢印が Subject, Globals, Session の箱に入ると）Subject, Globals, Session テーブル上のステートメントが実行され、変数が定義されます。 1 回目の試行が終わると（矢印が Subject, Globals の箱の外に出ると）Subject, Globals テーブル上の変数はリセットされます。Session テーブルの変数はリセットされず、セッションのあとのアンケートが終了するまで変数の値が保持されます。第 2 回目の試行が始まり、矢印が 2 行目の箱に入ると再び Subejct, Globals, Session テーブル上のプログラムが実行されます。

このように、テーブル上の変数がどの被験者に属し、ステートメントがいつ実行され、いつリセットされるかはどのテーブル上で変数が定義されているかで決まります。これをまとめたのが表 6 です。(Summary テーブルについては後述します。)

	Subjects	Globals	Session	Summary
変数・レコードが被験者ごとに用意される	○	×	○	×
ピリオドが終わるたびに変数をリセット	○	○	×	○
ピリオドごとにプログラムを実行	○	○	○	○
プログラムの実行中に過去の履歴をモニターできる	×	×	×	○

表 6　各テーブルの特徴

8.2 データのアクセス

8.2.1 テーブル・レコードをまたぐデータのアクセス

z-Tree のプログラムにおいては必要な機能に合わせてテーブルを選びますが、テーブル間あるいはレコード間でデータを受け渡しする必要が出てきます。z-Tree では直接他のレコードやテーブルの値を参照できる場合とできない場合があり、注意が必要です。

データのアクセスは、2 つの方向から整理しておく必要があります。ひとつはテーブルの種類が同じで、異なる被験者同士のレコード間でのデータの受け渡しでもうひとつは、同じ被験者が異なる種類のテーブル間でデータを行き来させる場合です。

8.2.2 同じテーブル内の異なるレコード間でのデータのアクセス

同じ種類で異なる被験者に属するデータの相互アクセスを可能にするのがテーブル関数です。テーブル関数は z-Tree の変数概念が生んだ特有の関数です。

テーブル関数の使用例は 7 章で説明したとおりです。またテーブル関数の一覧は次の 9 章にありますので参照してください。

8.2.3 異なる種類のテーブル間でのデータのアクセス

次に異なる種類のテーブル間でのデータの受け渡しですが、これは 2 つの方法があります。

ひとつは直接参照です。たとえば Globals テーブルで定義された変数はそのまま Subjects テーブルや Session テーブルで使用することができます。しかし Globals テーブルから Subjects テーブルを見る場合、どの被験者の Subjects テーブルを見るか、すなわちどのレコードの変数を見るのかという条件を与えなくてはいけません。ここで再びテーブル関数が登場します。

テーブル間でのデータのアクセスのもうひとつの方法は、テーブル関数の前にテーブル名をつけることです。たとえば公共財実験では全被験者の Subjects テーブルの中にはそれぞれ Contribution の値がありますが、その中の最大のものを Globals テーブルの中の変数 MaxContribution に代入したいとしましょう。その場合 Globals テーブルのプログラムダイヤログボックス内で次のように記述します。

```
MaxContribution=subjects.max(Conrtibution);
```

このステートメントに行き当たった z-Tree は、Subjects テーブルにゆき、全てのレコードの Contirubution をチェックして最大の値を拾って MaxContribution に代入します。もうひとつ例を挙げましょう。同じく Globals テーブル上の変数 DecOfOne に被験者番号 (Subject) 1 番の被験者の意思決定 Decision を代入したい場合、次のように記述します。

```
DecOfOne=subject.find(Subject==1,Decision);
```

このステートメントでは z-Tree は Subjects テーブルのレコード内の変数 Subject が 1 である被験者を探し出し、そのレコードにある変数 Decision の値をを拾ってきて RepDec に代入します。

テーブル間のデータのアクセスで、直接参照できる場合と”テーブル名. テーブル関数”が必要になる場合の関係は次の図 36 のように整理できます。

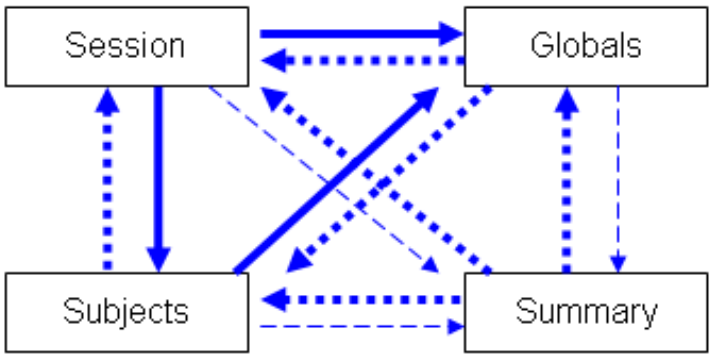


図 36　テーブル間のアクセス

太い実線は直接アクセスできる関係で、点線は”テーブル名. テーブル関数”でアクセスできる関係です。細い点線は”テーブル名. テーブル関数”でアクセスできるが、1 回目以降更新されません。

直感的にはどの被験者のどの瞬間のデータかを厳密に指定する必要がある場合には”テーブル名. テーブル関数”が必要になると考えてください。たとえば Subjects テーブルから見て Globals テーブルのデータは全員

にとって共通で、過去の値はリセットされているので値はどの時点でもひとつしかありません。従ってどの変数かを指定する必要はありません。しかし Globals テーブルから見て Subjects テーブルに書かれた変数は被験者の数だけありますから、どの変数なのかを指定しなくては値を見つけられません。

8.3 Summary テーブル

Summary テーブルは実験中の被験者行動を時系列に沿って監視するために使います。（この点マニュアルの記述がややミスリーディングなところがあり、把握しにくいかもしれません。）z-Tree では実験者は実験中、Globals や Session、Subjects テーブル内のデータを専用のウィンドウで監視できますが、監視できるのは当該ピリオドのみで、実験中に過去のデータを遡って見る事はできません。一方 Summary テーブルのウィンドウ内では過去のピリオドも表示されるため、実験中に時系列に沿って変動を把握したい数値がある場合、その数値を Summary テーブル上の変数に渡すことで過去のデータを画面上に表示させ続ける事ができます。

Summary テーブルは被験者ごとに用意されているものではないので、参加者の利得の平均値など、名前どおりサマリーとして全体の概要を記述する使い方が一般的です。

8.4 Parameters テーブル

基本的に z-Tree の Subjects テーブルと Globals テーブルは、被験者が対称的で、同じ実験を繰り返す場合に便利な設計になっています。しかし被験者が複数のタイプに分かれているような実験、あるいは実験の設定がピリオドによって異なるような実験を行いたい場合もありえます。このような場合に Parameter テーブルが用いられます。

Parametesr テーブルはメニューから別ウィンドウで展開される図 37 のような表で、図では S1, S2, S3 のラベルがついている列ラベル、1,2,3 のラベルがついている行ラベル、左肩にグループ番号を示す小さな箱かついたセルで構成され、それぞれクリックするとプログラムを書き込めるダイヤログボックスが開きます。

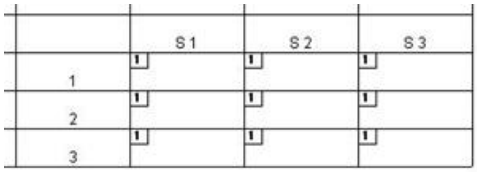


図 37　Parameter テーブルウィンドウ

列ラベルをクリックして現れるダイヤログボックスは Role Parameters といい、そこに記述したパラメータ設定などのステートメントはその列番号の被験者に対して全期間を通じて実行されます。

行ラベルをクリックして現れるダイヤログボックスは Period Parameters といい、そこに記述したステートメントはそのピリオドの全被験者に対して実行されます。

セルをクリックして現れるダイヤログボックスは Specific Parameters で、記述されるステートメントはその被験者のそのピリオドでのみ実行されます。Parameters テーブルはその名前から Subjects テーブルや Globals テーブルのような独立した変数テーブルであるかのように誤解されやすいのですが、実際はこの 3 つのパラメータは既存のテーブルのどれかに属します。そしてどのテーブルに属しているかで 3 つのパラメータの機能と相互関係が決まってきます。それぞれの機能などについては以下の表 7 を参照してください。

パラメータ名	Role	Period	Specific
定義する場所	列ラベル	行ラベル	セル
プログラムが実行される範囲	その列の被験者に対して全ピリオド	そのピリオドの全被験者	その列のそのピリオドの被験者
テーブル	Subjects	Globals	Subjects
備考	あらかじめステージツリーで定義した変数のみ扱える	Role、Specific とはテーブルが違うので同じ変数名を使っても違う変数とみなされる。従って Role や Specific で使っている変数の値をここで変える事はできない。	Role が優先されるので Role で定義されている変数の値ををSpecific で定義しなおすことはできない

表 7　各テーブルの特徴

9　参考資料 2：関数・ステートメントなど

オペレータ

マスマティカルオペレータ

+	加法
-	減法
*	乗法
/	除法

リレーショナルオペレータ

<	未満
<=	以下
==	等号
!=	不等号
>=	以上
>	大なり

ロジカルオペレータ

&	and
|	or

スコープオペレータ

:	次の高次スコープ
/	最も高次のスコープ　これは常にグローバルテーブルのレコードに帰着する

ステートメント

x=y;	ｙの値をｘに代入
if (a) {ss}	a が正しい場合ステートメント ss が実行される
if (a) {ss1} else {ss2}	a が正しい場合ステートメント ss1 が実行され、そうでなければステートメント ss2 が実行される
if (a) {ss0} elsif (b1) {ss1} elsif (b2) {ss2}....	a が正しい場合ステートメント ss0 が実行され、そうでなく b1 が正しい場合ステートメント ss1 が実行される。以下同様。
t.do{ss}	テーブル t の中のステートメント ss がすべてレコードに対して実行される。

if (a) {ss0} elsif (b1) {ss1} elsif (b2) {ss2}.... else {sse}	a が正しい場合ステートメント ss0 が実行され、そうでなく b1 が正しい場合ステートメント ss1 が実行される。以下同様。どの条件文も正しくない場合、sse が実行される。
t.new{ss}	テーブル t の中で新しいレコードが作成され、ステートメント ss が実行される。
array v[x];	行列変数 (インデックスが付けられた変数) の定義。行列変数は v[x] でアクセスすることができる。インデックスは 1 のステップで 1 から x まで用意される。
array v[x, y];	行列変数 (インデックスが付けられた変数) の定義。行列変数は v[x] でアクセスすることができる。インデックスは 1 のステップで x から y まで用意される。
array v[x, y, z];	行列変数 (インデックスが付けられた変数) の定義。行列変数は v[x] でアクセスすることができる。インデックスは z のステップで x から y まで用意される。
while(a) {ss}	a が正しい間ステートメント ss が実行される。ループは<Ctrl><Alt> [F5] で止められる。
repeat {ss} while (a);	まずステートメント ss が実行され、次に a がチェックされる。a が正しい間ステートメント ss が実行される。ループは<Ctrl><Alt> [F5] で止められる。
later (a) do {ss}	a が計算され、その値の秒数後に ss が実行される。ss はこのコマンドが使われるスコープエンバイロメントの中で使われる。
later (a) repeat {ss}	a が計算され、その値の秒数後に ss が実行される。ss はこのコマンドが使われるスコープエンバイロメントの中で使われる。その後 a が計算され、その値の秒数後に再び ss が実行される。a がマイナスになるとこのコマンドはキャンセルされる。

関数

abs(x)	x の絶対値
and(a, b)	a と b が正しいときに正しいとする。(True の値を返す)
atan(x)	x のアークタンジェント
cos(x)	x のコサイン
exp(x)	x の指数関数
gettime()	コンピュータがスタートするまでの秒数
if(a, x, y)	もし a が正しいなら関数の値は x、そうでなければ y
ln(x)	x の自然対数
log(x)	x の 10 を底とする対数
max(x, y)	x,y の最大値
min(x, y)	x,y の最小値
mod(x, y)	x を y で割った余り
not(a)	条件 a が真でない場合に真
or(a)	条件 a か条件 b が真である場合に真
pi()	円周率

power(x, y)	x が正の場合は x の y 乗、負の場合は y が偶数なら x の y 乗、奇数なら x の絶対値の y 乗
random()	0 から 1 までの一様分布の乱数
randomgauss()	平均 0 の標準正規分布の乱数
randompoisson(x)	平均 x のポアソン分布の乱数
round(x, y)	x を y の倍数で丸める（y の倍数で最も近い値にする）
rounddown(x, y)	x を y の倍数で切り下げ
same(x)	x == :x の短縮形
sin(x)	x のサイン
sqrt(x)	x の平方根

テーブル関数

average(x), average(a, x)	x の平均値、a の条件を満たす x の平均値
count(), count(a)	テーブルのレコード数、a の条件を満たす x のレコード数
find(x), find(a, x)	レコードの先頭の x の値、a の条件を満たす最初のレコード x
maximum(x), maximum(a,x)	x の最大値、条件 a を満たす x の最大値
median(x), median(a, x)	x の中位値、条件 a を満たす x の中位値
minimum(x), minimum(a, x)	x の最小値、条件 a を満たす x の最小値
product(x), product(a, x)	x の積、条件 a を満たす x の積
regressionslope(x, y)	x、y の線形回帰の傾斜
regressionslope(a, x, y)	条件 a を満たす x、y の線形回帰の傾斜
stddev(x), stddev(a, x)	x の標準偏差、条件 a を満たす x の標準偏差

10　参考資料 3：実験のインストラクション

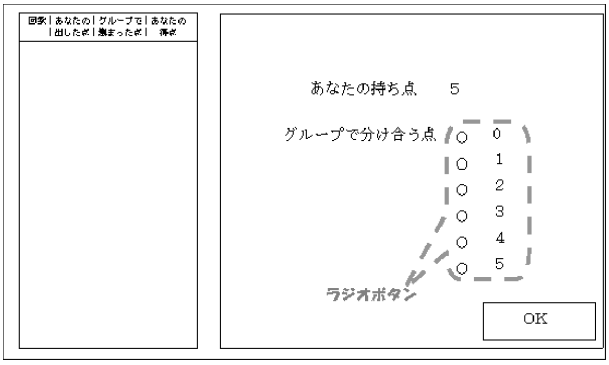
被験者実験のインストラクションの例を以下のページに掲載します。

実験の説明

ある経済環境を再現したゲームを行います。ゲームであなたが稼ぐ得点はあなたとあなた以外の人がどのような決定をするかで決まります。しかしあなたは事前に他のプレイヤーと相談することはできません。自分が最もよいと思う戦略をよく考えて選択し、なるべく沢山の得点をとることを目指してください。

実験のルール

これから実験のルールを説明していきます。実験はコンピューターのスクリーン上で行います。実験が始まると画面は次のようになります。



これから行う実験は　　回繰り返されます。毎回実験をはじめる前に、この画面になります。画面左側は２回目以降それまでの結果が表示されます。右側は毎回のあなたの意思決定を入力する場所です。縦に並んでいるラジオボタンをマウスポインタで押す事で選択を行います。

みなさんは（　）人で１グループになります。グループが複数になる場合、誰がどの人と同じグループかは実験中明らかにされません。全員誰が誰と同じグループか知りません。実験中、あなたは同じ人と同じグループになります。

実験でみなさんが手に入れることのできる得点は、自分がどんな決定をするかと、自分のグループの他の人たちがどんな決定をするかによって決まります。

あなたは、毎回の実験の最初に５点を与えられます。あなたは５点のうち何点を自分のためにとっておき、何点をグループで分け合うかを決めます。あなたはグループ内で分け合うことに決めた点を画面上から入力します。

グループ内のメンバーがグループで分け合うことにした点数は、全員分合計されてつぎのように計算されてグループの各メンバーに分けられます。

1.6 ×（グループ全体で集まった合計点数）1 ÷グループの人数

この計算の意味は、

・グループ全体で集まった点数の合計にまず 1.6 がかけられます。

・それをグループの人数で割ります。

たとえば、あなたのグループ全員で１０点集まったとします。あなたのグループが 5 人だとするとグループの全員がそれぞれ受け取る点は、

10 × 1.6 = 16

16 ÷ 5 = 3.2

から、3.2 になります。全員の意思決定が済むと、コンピューターが皆さんの点を計算して画面に表示します。

あなたが最初に 2 点を分け合うことにして画面に入力していたとします。このときあなたは手持ちの 5 点から 2 点を引いた 3 点を自分のために取っておいたことになります。あなたのこの場合の総得点は

3 + 3.2 = 6.2

で 6.2 点になります。

このようにあなたの得点は

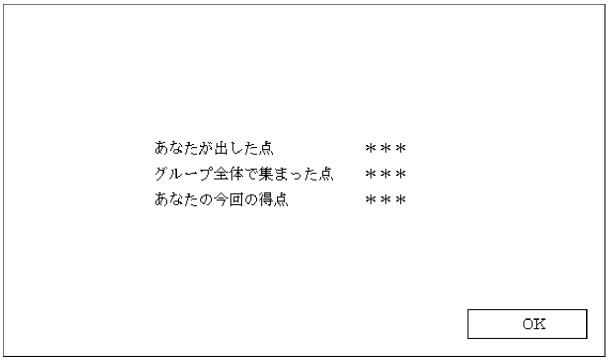
$$(5-\text{グループで分け合う点}) + \frac{1.6*(\text{グループ全体で集まった合計点数})}{\text{グループの人数}}$$

で計算されます。以下に 5 人グループの場合の例を幾つか挙げておきます。

- あなたが 0 点をグループと分け合い、グループ全体で集まった点が 0 点だったとき
  （5 × 0）+ 1.6 × 0 ÷ 5 = 5　**5 点**
- あなたが 3 点をグループと分け合い、グループ全体で集まった点が 15 点だったとき
  （(5 × 3）+ 1.6 × 15 ÷ 5 = 6.8　**6.8 点**
- あなたが 4 点をグループと分け合い、グループ全体で集まった点が 8 点だったとき
  （5 × 4）+ 1.6 × 8 ÷ 5 = 3.56　**3.56 点**
- あなたが 1 点をグループと分け合い、グループ全体で集まった点が 20 点だったとき
  （(5 × 1）+ 1.6 × 20 ÷ 5 = 10.4　**10.4 点**

計算の仕方わかりましたか？わからなければ手を挙げてください。

結果の表示



このような結果が 1 回ごとに画面に表示されます。結果の表示画面は次のようなものです。

画面の得点は、上で説明した方法で計算されたものです。グループで何点集まったかや、自分のもらった得点を声を出して言ったりしないでください。もし誰かがそのようなことをしたら実験をすぐに中止します。こ

の結果は次の回から意思決定の画面の左側に常に表示されます。画面を確認したら「OK」ボタンを押してください。次の回の実験に進みます。

入力の仕方

グループで分け合う点は 0 点から 5 点までの範囲で選ぶことができます。決めたら画面上の任意の点のラジオボタンを押して「○」を「◎」の状態にしてから「OK」ボタンを押して決定してください。ほかの人の意思決定を待つ画面になります。何点を出すかは「OK」ボタンを押すまでは何度でも変えることができます。もしラジオボタンを押さずに「OK」ボタンを押すと、「ラジオボタンを押してください」という警告が出ます。

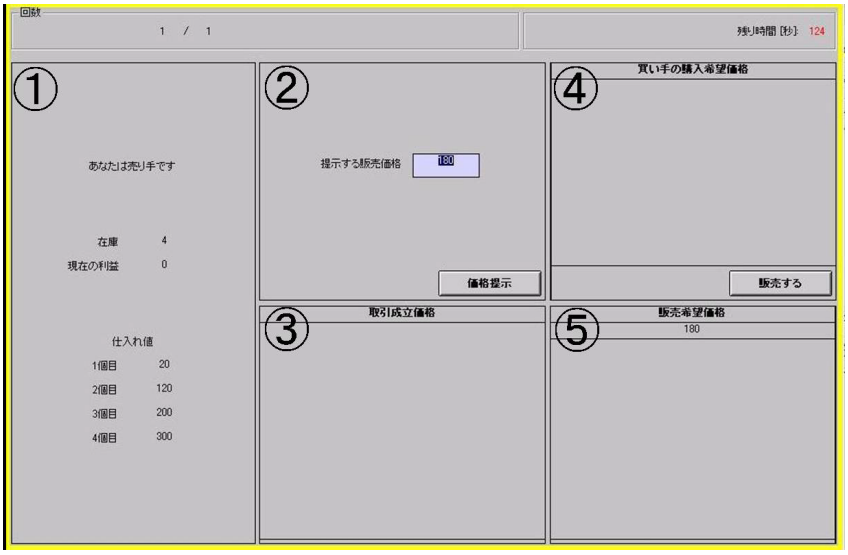
実験の説明

実験の内容

仮想的な市場で売り手と買い手に別れて財を売買します。売り手になった人はなるべく高く財を販売し、買い手になった人はなるべく安く買うことで得点を稼ぎます。実験が開始されるとコンピューターがあなたを売り手か買い手かを決めます。

売り手の場合

実験が始まると次のような画面になります。((1)〜(5) の数字は説明のためのもので実際には表示されません)



一番左の (1) のエリアにはあなたが売り手であること、あなたの手持ちの在庫数（ゲームの開始時に 4 つ)、現在の利益、在庫の仕入れ値がかかれています。

あなたは 4 つの商品を仕入れ値より高い価格で販売することで利益を上げます。販売には 2 つの方法があります。

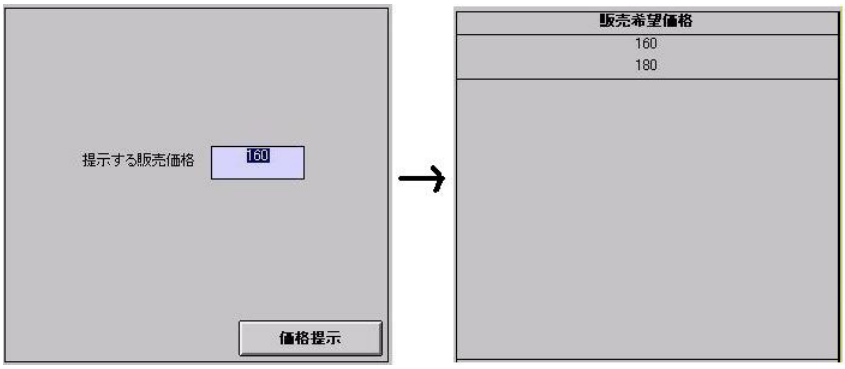
販売方法

1. 一つ目の販売方法は「自分で価格をつけて、その価格で買ってくれる人が現れるのを待つ」という方法です。

   自分が販売したい価格を　のエリアの「提示する販売価格」に入力して「価格提示」のボタンを押します。あなたが提示した価格は　の「販売希望価格」に表示されます。

   この販売希望価格の欄にはあなたを含む売り手全員が提示した価格が表示されます。買い手も同じ情報を見ています。買い手があなたの示した価格に満足して購入すると、あなたはその財を販売できたことになります。

   あなたより先に別の人が販売希望価格を提示している場合、あなたはその価格より安い価格をつけなくてはなりません。図の例では販売希望価格に既に 180 という価格が提示されているので、それよりも安い 160 という価格を提示しています。



2. 二つ目の方法は「買い手がつけた買値を見て、満足の行く価格ならその価格で売る」という方法です。(4) のエリアに買い手が示した「この価格なら払ってもいい」という購入希望価格が示されています。あなたがこの価格に納得して「販売する」ボタンを押すと、あなたの財が一つ売れます。

購入希望価格が複数あるときはその中から好きな価格をクリックして選ぶことができますが、一番高い価格がはじめから選択されていますので、その価格で販売するのが利益を上げるためには正しい選択です。図の例では 170 と 190 という価格が提示されており、190 が選択されています。ここで「販売する」ボタンを押すとあなたは手持ちの財を 190 という価格で販売できます。あなたが売ろうとしても他の売り手が先に売ってしまうことがあります。その場合その買い手はいなくなります。

販売のされ方

1),2) どちらの方法でも財は仕入れ値の安い順に出荷されます。もしあなたの財の仕入れ値が 50,100,150,200 円だった場合、一つ目を販売すると自動的に 50 円の財が出荷されます。次に出荷されるのは 100 円の財です。

結果の表示

取引が成立すると１）の場合は (5) の「販売希望価格」の欄から売り手が提示した価格が消え、２）の場合は (4) の「購入希望価格」の欄から買い手が提示した価格が消えます。そしてそのときの価格が (3) の「取引成立価格」のところに表示されます。ここにはあなたを含む全員の取引の結果が表示されます。

得点

得点は

得点＝取引成立価格－仕入れ値

で計算されます。もし 50 円の仕入れ値のものを 90 円で販売した場合、あなたの得点は 90-50=40　点です。仕入れ値を下回る価格で販売すると損しますので気をつけてください。一つ目の財で得点が上がった価格と同じ価格で二つ目を売っても、二つ目の財は仕入れ値が上がっているので損になるかもしれません。常に今売ろうとしている財の仕入れ値より高く売れるように努力してください。

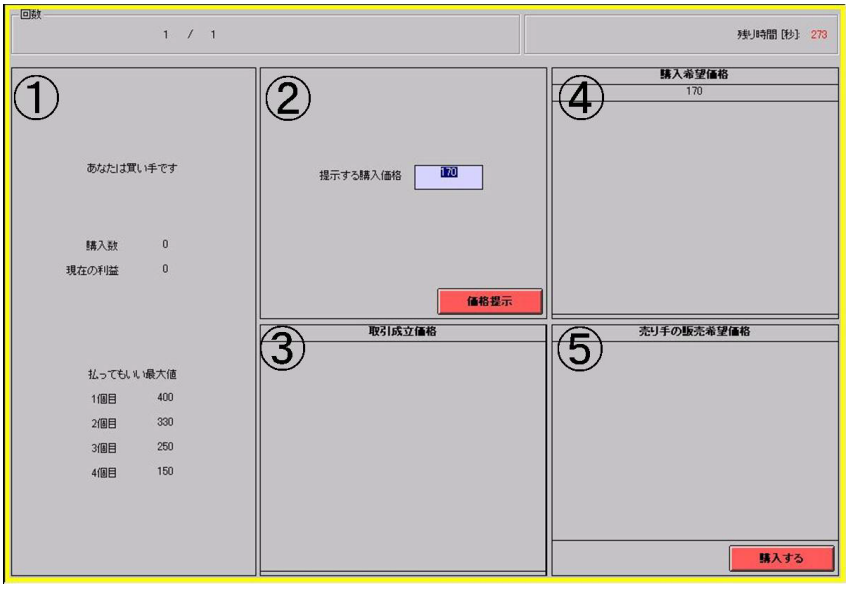
ゲームの終了

ひとつの実験は（　）分間で終わります。残り時間は右上に表示されています。時間が残っていても 4 つの財をすべて販売してしまうとあなたのゲームは終わりです。終了まで静かに待ってください。

注意「販売する」のボタンを押すときは「購入希望価格」欄を見て、誰かが価格を提示していることを確認してください。誰も提示していなくて空欄になっているときは押しても何もおきません。

買い手の場合

実験が始まると次のような画面になります。((1)〜(5) の数字は説明のためのもので実際には表示されません)

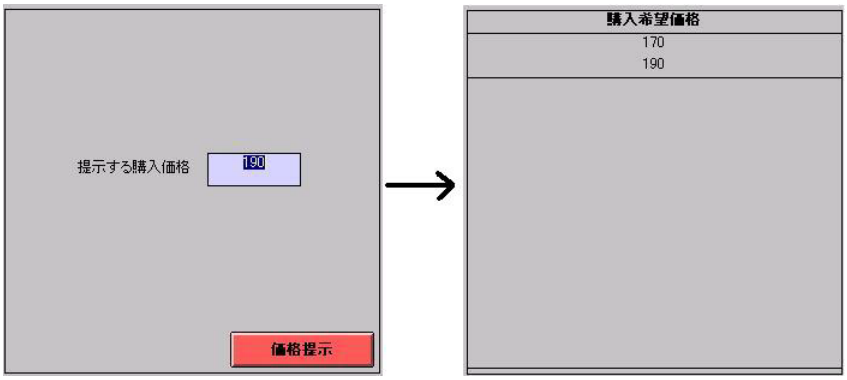


一番左の (1) のエリアにはあなたが買い手であること、あなたが購入した財の数（4 つまで)、現在の利益、払ってもいい最大値がかかれています。あなたは払ってもいい最大値より安い価格で財を購入することで利益を上げます。購入には 2 つの方法があります。

購入方法

1. 一つ目の方法は「自分で価格をつけて、その価格で売ってくれる人が現れるのを待つ」という方法です。

   自分が購入したい価格を (2) のエリアの「提示する購入価格」に入力して「価格提示」を押します。あなたが提示した価格は (4) の「購入希望価格」に表示されます。



この購入希望価格欄にはあなたを含む買い手全員の提示した価格が表示されます。売り手も同じ情報を見ています。もし売り手があなたの示した価格に満足して販売の意思決定をすると、あなたはその財

を購入できます。

あなたより先に別の人が購入希望価格を提示した場合、あなたはその価格より高い価格をつけなくてはなりません。図の例ではすでに誰かが 170 という価格を提示しているのでそれよりも高い 190 という価格を提示しています。

2. 二つ目の方法は「売り手がつけた価格を見て、満足の行く価格ならその価格で買う」という方法です。

(5) のエリアに売り手が示した「この価格なら売ってもいい」という販売希望価格が示されています。あなたがこの価格に納得して「購入する」ボタンを押すとあなたは財を買うことができます。

販売希望価格が複数あるときはその中から好きな価格をクリックして選ぶことができますが、一番安い価格がはじめから選択されているので、その価格で購入するのが利益を上げるためには正しい選択です。図の例では 160 と 180 という価格が提示されており、160 が選択されています。ここで「購入する」ボタンを押すとあなたは財を 160 という価格で購入できます。あなたが買おうとしても他の買い手が先に買ってしまうことがあります。その場合その売り手はいなくなります。

購入の仕方

1),2) どちらの方法でも財の購入は「払ってもいい最大値」の大きい順に行われます。もしあなたの「払ってもいい最大値」が 200,150,100,50 だった場合、一つ目を買うときの払ってもいい最大値は 200 円です。次の二つ目の財に対して払ってもいい最大値は 150 円です。

結果の表示

取引が成立すると１）の場合は (4) の「購入希望価格」の欄から買い手が提示した価格が消え、２）の場合は (5) の「販売希望価格」の欄から売り手が提示した価格が消えます。そしてそのときの価格が (3) の「取引成立価格」のところに表示されます。ここにはあなたを含む全員の取引の結果が表示されます。

得点

得点は

得点＝払ってもいい最大値－取引成立価格

で計算されます。もし 100 円払ってもいいものを 90 円で購入した場合、あなたの得点は 100-90=10　点です。払ってもいい最大値を上回る価格で購入すると損しますので気をつけてください。一つ目の財で得点が上がった価格と同じ価格で二つ目を買っても、二つ目の財は払ってもいい最大値が下がっているので損になるかもしれません。常に今の払ってもいい最大値より安く買うように努力してください。

ゲームの終了

ひとつの実験は（　）分間で終わります。残り時間は右上に表示されています。時間が残っていても財を４つ購入してしまうとあなたのゲームは終わりです。終了まで静かに待ってください。

注意

「購入する」のボタンを押すときは「販売希望価格」欄を見て、誰かが価格を提示していることを確認してください。誰も提示していなくて空欄になっているときは押さないでください。