

HARRIS SCHOOL WORKING PAPER
SERIES 06.05L

APPENDIX: NOTES ON THE TEMPLATE *

Howard Margolis

**This is a draft chapter from Cognition and Extended Rational Choice
(forthcoming Routledge 2007).*

Appendix. Notes on the *template*

To handle the data solicited from experimenters, I gradually built up a 'template' for storing the characteristics and results of experiments and for running across-experiments analysis. A routine called "takedata" extracts, codes and compactly stores the experiment characteristics, parameters and results. The software is intended to then search across datasets stored to find suitable experiments for a test, then searches within the chosen datasets for particular situations which may be of interest. But since I was motivated only to do what I needed for my own project, things are in an incomplete state. But enough is in hand to show an interested reader how all that can work.

The tests currently coded are prompted by the NSNX model of choice which is the focus of my own interest. But the template is not intrinsically tied to NSNX. It always starts by retrieving a stored vector which codes for the type of experiment and for the particular parameters in play. It then can compile the relevant data into several multi-dimensional abstract arrays. A user with the modest familiarity with coding required to write simple nested loops will then be able to define routines which will run through the arrays to test whatever theories they wish to explore.

For a single experiment the basic array is 4-dimensional (GRP x RD x ID x TESTDATA). In its current state, the final dimension of the 4D array contains: (a) the choice for that GRP x RD x ID, (b) the ratio between the choice and the average of others' choices for that GRP x RD, and (c) the ratio of player's payoff to her starting endowment for that round. A supplementary 3D array contains (in its third dimension) the pool and the standard deviation for each GRP x RD. For particular tests these basic arrays can be extended to hold additional numbers in their final dimension, such as penalties imposed by some players on others. Tests that run across multiple experiments require a 5th dimension (XP x GRP x RD x ID x TESTDATA). For a 4D array it is easy to envision the array as a 3D grid with a sequence of three (or more) numbers within each 3D cell. For 5th or higher order arrays this intuition is lost, but of course the mechanics of handling the data remain trivial for the computer.

A minor but significant use for the template is simply to be able to efficiently locate within what could gradually become a very large repertoire of stored datasets an experiment or two that have some characteristic(s) a person finds reason to want to

examine closely, for example in preparation for running an experiment on similar or related lines, or that a person recalls but not with enough specificity to easily locate it.

But the *template* has a more ambitious potential for creating what I'll call "pseudo-natural experiments": so-called because the raw material for the experiments I have in mind arises not in actual natural settings but fortuitously in the course of experiments. The template will be able to scan quickly over a large repertoire of raw data from many experiments, picking out situations which only occasionally and unpredictably arise in an actual experiment, but which are bound to be found many times in a large repertoire of datasets. So we have a statistically meaningful basis for distinguishing (to mention a salient example) situations which strongly prompt reciprocal behavior from others -- within the same or interestingly related sorts of experiments -- where reciprocity falls far short of what common experience and systematic study by social psychologists would lead us to expect. And if we are exploring a model with some analytical bite -- of course that is what I want to claim for NSNX but also what people interested in other models must want to claim as well -- we will be prompted by the model as to what sorts of comparison situations are likely to prove especially fruitful.

As with the 'reciprocity' issue, many other questions are currently in sight (the stability of 'types', the conditions for pareto-damaging indifference aversion, the conditions for marked inequality indifference or the lack of it, conditions which make 'cheap talk' highly effective or not so, and so on. And no doubt as these issues are clarified, deeper or less transparent questions will surface.

An important, not controversial but often neglected point here turns on the striking propensity *and* also the striking ability of human beings to concoct some sort of story to justify not believing unwelcome evidence and uncritically accepting agreeable evidence. Overcoming the tendency in either direction usually requires finding an effect repeatedly in different places, in different ways. Single pieces of evidence blunt enough and reliable enough to persuade are rare. What is usually needed is multiple pieces of evidence supporting the same inference... as indeed it ought to be, since any single piece of evidence might prove to be misleading. A great merit, I think, of the *template* is that it will greatly facilitate constructing tests that reinforce sound claims and embarrass faulty claims.

Since the NSNX account I am exploring is out of the mainstream, the need for multiple evidence is especially acute for me. That is probably why I was especially motivated to work up this kind of software. But from the absence (so far as I know) of any papers that survey *data* across more than a handful of experiments (*not* just results, as in most literature reviews), apparently software of the sort I have been constructing has not been worked on by others. But it ought to be.

Developing the template on a lone wolf basis would hardly be sensible. Readers here are unlikely to need instruction on the interactions of individual incentives and public goods. The template as it stands can quickly take up and compactly store data, parameters and subsidiary features of experimental data (rewards, exclusion, heterogeneous endowments, etc.), and then search across its repertoire to compile test arrays meeting prescribed conditions. It provides a running start for developing things further, either from this start in hand or perhaps starting from scratch but with a trial run at hand.

My coding is in VBA/Excel. But the template should be usable by people who don't know VBA. So if development were to proceed from what is here, there should be a transfer point where someone who prefers to work in another format, such as STATA or SASS, converts the test arrays constructed by the template to their preferred platform.

But there are a couple of reasons that leaving the template as VBA coding might be best:

*. VBA is the most transparent programming language that will fluently do what is needed. Anyone who can write code in any language can pretty well read VBA code at sight. So to the extent that someone writing code in their preferred language needs (or wants) to see exactly what the template is doing, leaving it in VBA is almost as good as leaving it in her own preferred language. Similarly for someone new to writing code who wants to design a pseudo-random experiment, the simplest language to learn to write her custom instructions would be VBA. Personally, I had never used a spreadsheet nor ever written a line of code before needing to do so for this project.

The template is organized to set up a very small number (perhaps just one) multi-dimensional array on which some test is run. Running a search requires only responses to ordinary language pop-up prompts. The user has no need to know anything about the

code, hence could not seriously care whether it is in VBA or STATA or anything else. And the resulting search array(s) in VBA can easily be translated by a short, canned subroutine into STATA or any other language of the user's choice.

The crudest but already probably quite good enough way to arrange the transfer is to "flatten" the multi-dimensional VBA arrays into two dimensions. This can then be pasted into a (2-dimensional, of course) spreadsheet, and another program such as STATA can take up the information to handle it in its usual way. OR the (usually 5-dimensional) VBA test array can be saved as a one-dimensional CSV (comma-separated-variables) file, then directly operated on by whatever machinery the user customarily uses, given that the row of numbers in the CSV is organized in a predetermined way, so that serial position in the row implies whatever else the user needs to know.

On the FOSS model, perhaps there need be no centralized entity doing this..

The most obvious things that need to be done seem to be: (1) editing of my surely idiosyncratic and amateurish coding by someone with real competence at this sort of thing; (2) extension of my notes and comments within the coding to an adequate User's Guide; (3) extension of the "takedata" coding to intermesh easily with the Z-tree software commonly used run choice experiments. This last is what is needed to make the template really useful and really easily-used. It could not be difficult to do. But it requires participation by one or more people interested in the project and intimately familiar with the Z-tree coding.

A reader interested in exploring what use she might make of the template can download a copy (which would include a "starter" set of data from nearly 100 experiments) from the Harris School website, or download the Working Paper version of this Appendix, which provides a printout of the coding. So a copy of the template as an Excel workbook is available, or a printout of the coding.

TEMPLATE CODING (as is, as of January 2007).

Option Base 1

Sub zcolumns()

'sets up take of VT zsquare data

'takes defaults

ReDim zset(6)

zset = Evaluate("zzset")

JUMP1: inputs = InputBox("VT columns currently set at:" & vbCr & vbCr & _

"1: Column for grps = " & zset(1) & vbCr & vbCr & _

"2: Column for rnds = " & zset(2) & vbCr & vbCr & _

"3: Column for IDin = " & zset(3) & vbCr & vbCr & _

"4: Column for IDout = " & zset(4) & vbCr & vbCr & _

"5: Column for value1 = " & zset(5) & vbCr & vbCr & _

"6: Column for value2 = " & zset(6) & vbCr & vbCr & _

"To set a parameter, enter #x#: e.g. to set grpcolumn to D, enter 1xD" & vbCr & vbCr
& _

"If ok as is just hit Enter." & vbCr & vbCr & _

"NOTE... only VT parameters are used. Others shld be ignored")

If inputs <> "" Then

'code says, 1st character is parameter #, ignore 2nd character, take rest as input

'value1 will be "give" for CHOICEDATA, value2 might be punishment points, votes,
etc.

'if more than 2 value columns are needed, report 'value2' as eg., f/z/gg: code will spot
this

lenx = Len(inputs)

i = nnn + Mid(inputs, 1, 1)

zset(i) = Mid(inputs, 3, lenx - 2)

Dim resetzsave%: resavezset = 1

GoTo JUMP1

End If

'save parameters (if revised)

If resavezset = 1 Then Names.Add Name:="zzset", RefersTo:=zset

End Sub

Sub zsort()

'sorts to grp/rd/id [whose columns are zset(1,2,3)]

Set alpha = ace.End(xlToLeft): Set beta = alpha.Offset(Xrows - 1, 1)

Range(alpha, beta).Select

Range(Selection, Selection.End(xlToRight)).Select

Selection.Sort Key1:=Range(zset(1) & "1"), order1:= _
xlAscending, Key2:=Range(zset(2) & "1"), order2:=xlAscending _
, Key3:=Range(zset(3) & "1"), order3:=xlAscending, Header:= _
xlNo

End Sub

```

Sub setdataranges() '###
'next line handles generic 'take' (xptype = 0) or 2nd (idout) 'take' for Points segments
If segment = "POINTS" Then Stop
If xptype = 0 Or getidout = 1 Then GoTo JUMP0
shtset = Evaluate("zshtset")
zset = Evaluate("zzset")
nnn = nnn * 10 'reserves 10 slots/segment, moves to next set of sheet
parameters for new segment
If nnn + 10 > 199 Then MsgBox ("MUST redim up shtset") 'in case eventually >19
segments coded for
'using 'shtset' inputs from previous 'take'
'If segment = "IDGRID" Then 'MAKES DEFAULT.. pick up blocks, rows, cols
from CHOICEDATA
' shtset(12, 1) = shtset(2, 1): shtset(13, 1) = shtset(3, 1): shtset(14, 1) = shtset(4, 1)
JUMP1: Cells.Interior.ColorIndex = xlNone
inputs = InputBox("Columns zset is: " & zset(1) & ", " & zset(2) & ", " & zset(3) & ", " &
zset(4) & _
" " & zset(5) & ", " & zset(6) & ", " & vbCr & vbCr & _
"Spreadsheet parameters for " & segment & " currently set are:" & vbCr & _
vbCr & "1: Xstart = " & shtset(nnn + 1, 1) & " = upper left cell for " & segment &
vbCr & _
vbCr & "2: Xblocks = " & shtset(nnn + 2, 1) & vbCr & _
vbCr & "3: Xrows = " & shtset(nnn + 3, 1) & vbCr & _
vbCr & "4: Xcolumns = " & shtset(nnn + 4, 1) & vbCr & _
vbCr & "5: Xoffsetr = " & shtset(nnn + 5, 1) & " = number of rows to slip between
blocks " & vbCr & _
vbCr & "6: Xoffsetc = " & shtset(nnn + 6, 1) & " = number of columns to slip between
blocks " & vbCr & _
vbCr & "7: Xsort = " & shtset(nnn + 7, 1) & ". Set to 1 to resort blocks putting rows
in sequence." _
& "Set to 2 if columns. Enter 0 if no sort needed." & vbCr & vbCr & _
"To reset a parameter, enter #x#: e.g. to set Xblocks to 7 enter 2x7. Blank return
accepts" _
& " parameters as shown.")
If inputs <> "" Then
' this assumes never more than 9 (i.e., single digit) parameter inputs
'code says, 1st character is input #, ignore 2nd character, take rest as input
lenx = Len(inputs)
i = nnn + Mid(inputs, 1, 1)
shtset(i, 1) = Mid(inputs, 3, lenx - 2)
If segment = "CHOICEDATA" And thirdsrt = 2 Then 'copies 'choicedata'
parameters to IDGRID with shift of column for xstart
shtset(10 + i, 1) = Mid(inputs, 3, lenx - 2)
shtset(11, 1) = zset(3) & Mid(inputs, 4, lenx - 3)
End If
Dim resetshtsave%: resaveshtset = 1

```

```

GoTo JUMP1
End If
'save parameters (if revised)
If resaveshtset = 1 Then Names.Add Name:="zshtset", RefersTo:=shtset
'set for this take
  If segtype = 1 And nnn > 0 Then
    lenx = Len(shtset(1, 1))
    Xstart = zc & Mid(shtset(1, 1), 2, lenx - 1)
    Else: Xstart = shtset(nnn + 1, 1)
  End If
  Xblocks = shtset(nnn + 2, 1)
  Xrows = shtset(nnn + 3, 1)
  Xcolumns = shtset(nnn + 4, 1)
  Xoffsetr = shtset(nnn + 5, 1)
  Xoffsetc = shtset(nnn + 6, 1)
  Xsort = shtset(nnn + 7, 1)
JUMP0:
ReDim setrng(Xblocks), Xace(Xblocks)
Set ace = Range(Xstart)
ReDim Xsetrng(Xblocks)
For h = 1 To Xblocks
  If h > 1 Then
    If Xoffsetc = 0 Then
      Set ace = ace(Xrows + 1 + Xoffsetr, 1)
    Else: Set ace = ace(1, Xcolumns + Xoffsetc)
    End If
  End If
'ace.Select: Stop 'for tracking error in setdata
'[10.31] sets up array here of starting points for each block of data in case 'sort' sub is
needed
  If h = 1 Then Set Xace(h) = Range(Xstart) Else Set Xace(h) = ace '[11.1]
  Set setrng(h) = ace.Range(Cells(1, 1), Cells(Xrows, Xcolumns))
  If h = Xblocks Then setrng(h).Select 'used for visually checking final block of
'take'
  'resort VT block (if needed) to grp/rd/id
  If sortorder <> 1 And thirdsort = 2 And segtype = 1 Then zsort
  'copy range (setrng) into array (Xsetrng)
  '[10.31]... this makes each (of 'h') 'dataget' arrays an element of Xsetrng
  Dim dataget: ReDim dataget(Xrows * Xcolumns)
  For i = 1 To Xrows * Xcolumns
    dataget(i) = setrng(h)(i)
  Next i
  Xsetrng(h) = dataget
  'transpose blocks if needed to get correct take order (grp, rd, id)
  If Xtranspose = 1 Then WorksheetFunction.Transpose (Xsetrng(h))
Next h

```

```

setrng(h - 1)(Xrows * Xcolumns).Activate
'accepts take from spreadsheet as correct, or jumps back to adjust parameters
JUMP2: inputs = InputBox("IF final block of " & segment & " is correct, press ENTER.
" & vbCr & _
    vbCr & "If a sheet parameter needs correction (Xstart, Xoffsetr, Xoffsetc, Xsort),
enter 1." & vbCr & vbCr & _
    "If an xpt parameter (Xblocks, Xrows, Xcolumns -- depend on grps, rnds, n),
enter 2." & _
    vbCr & vbCr & "Enter 999 to PAUSE, X to abort 'take' at this point.")

```

```

If inputs = "x" Or inputs = "X" Then
    abort = 1
    Exit Sub
End If

```

```

'returns to sheet parameters (if 1) or xpt parameters (if 2) to allow
'corrections if final block not right
If inputs = 1 Then
    GoTo JUMP1
    ElseIf inputs = 2 Then
        fixparameters = 2
End If

```

```

'note that (Excel quirk) though 'holddata' is just a serial list, it (& other data that might
exceed 256 points _
    is dimensioned as 2D: (serial position X actual data pt).
'Ditto for such data when arrays are reconstructed for use. This quirk comes via
Excel's orientation to 2D
'spreadsheets with 256 columns to a row (more than 256 needs more than 1 row, so is
2D in a spreadsheet).
'So any list that might exceed 256 points must be coded as 2D... e.g. holddata(300,1)to
allow the list
'to be 'wrapped' into successive rows.
If inputs = 999 Then    'final block highlighted when 999 is returned
    Stop    'if needed click Alt F11 to toggle to spreadsheet
    'use right-side 'handles' to move spreadsheet
    'using cursor to search turns off highlight of last block
    GoTo JUMP2

```

```

End If
'if needed, change e.g. rank-ordered data into serial-ordered
count = 0
If Xsort > 0 And segment = "VOTING" Then
    For h = 1 To Xblocks
        sortgrid    'sorts rows(if Xsort = 1), cols (if Xsort = 2) if ranges are e.g. rank-
rather than serial-ordered
    Next h
End If

```

```
'at this point all should be in order
If segtype = 2 Then getidout = 1      'alert to get inputs for IDout array: GoTo JUMP3
End Sub      'returns to "ccc" for current segment
```

```
Sub sortgrid()
'changes, eg., rank ordering of block into required serial ordering
'next if/then allows sort by columns (not rows)
'[10.31] CHECK if CODE IS CORRECT FOR COLUMNS RESORT
Dim ordera, orderb
If Xsort = 2 Then
    ordera = Xcolumns: orderb = Xrows
    Else: orderb = Xcolumns: ordera = Xrows
End If
```

```
ReDim index(ordera, 1) 'extra dimension here for compatibility with 'fixdigitsX' &
'fix0data'
    For i = 1 To ordera
        If Xsort = 1 Then index(i, 1) = Xace(h)(i, 0) Else index(i, 1) = Xace(h)(0, i)
    Next i
```

```
'rest CODED ONLY for sort rows
'prelim if-then for sub to strip e.g., "id3" to "3", if needed
    If Not IsNumeric(index(1, 1)) Then
        ReDim fixdigitsX(ordera, 1)
        fixdigitsX = index
        fixdigits      'in this sub fixdigitsX(xxx,1) to allow for (unlikely but possible)
exceptionally large IDGRID
        index = fixdigitsX
    End If
```

```
'fix0 if needed
If WorksheetFunction.Min(index) = 0 Then
    ReDim fix0data(ordera, 1): fix0data = index
    fix0base
    index = fix0data
End If
```

```
'set temporary holder for serial-sorted array for this setrng
Dim xxsetrng(), ii%: ReDim xxsetrng(ordera * orderb)
    For i = 1 To ordera
        For j = 1 To orderb
            ii = (index(i, 1) - 1) * orderb + j
            xxsetrng(ii) = Xsetrng(h)((i - 1) * orderb + j)
        Next j
    Next i
'reset serial-ordered Xsetrng
'[10.31] NOT CHECKED for sort by columns
```

```

    For i = 1 To ordera
      For j = 1 To orderb
        Xsetrng(h)((i - 1) * orderb + j) = xxsetrng((i - 1) * orderb + j)
      Next j
    Next i
  End Sub
MODULE 9^^^
-----
MODULE 1.....

Option Explicit
Option Base 1

Sub getstrgrid() 'gets strgrid for xpt z (of zzz in this test)... written Brandts/Schram
'contribution function' data... uses prV array (10 values), stored as 'bsvs' to
'construct choicegrid from stored prV serials... SHLD be easily
adapted to other 'strategy method' data
Dim str, yyy '[1.20... need more general, not just br/schr
xxx = Evaluate("bsvs")
makearrays
bsv = yyyy '10 prVs
'reconstructs choice grid strategy(grp,mix,id,str)
'FINISH [1.20]
Dim count: ReDim holddata(grps * N * rnds * strategies, 1)
'CHECKED for code 15, indicating strategy matrix to reach this sub, 151=same for all
in grp, default = varying across players)
count = 0
holddata = Evaluate("str" & xpname(z))
For grp = 1 To grps
  For rd = 1 To rnds
    For id = 1 To N
      For str = 1 To strategies
        count = count + 1
        strgrid(z, grp, rd, id, str) = holddata(count, 1)
      Next str
    Next id
  Next rd
Next grp

' converts stored lottery category XP() into actual choice-that-counts xp()
For grp = 1 To grps
  For rd = 1 To rnds
    For id = 1 To N
      yyy = xp(z, grp, rd, id, 1)
      xp(z, grp, rd, id, 1) = strgrid(z, grp, rd, id, yyy)
    Next id
  
```

```
Next rd
Next grp
End Sub
```

```
Sub sch() 'tests for brandts/schram data
```

```
End Sub
```

```
Sub pp() 'DIFF test for P&P '97
```

```
'checks mean choice for P&P DIFF < 6 (ceiling for puV(z) = 15) across treatments
```

```
nn = ppp(z, 2, 1) * ppp(z, 5, 1) * ppp(z, 6, 1) 'total # of choices
```

```
inputs = 4 'allows 'ppfill' to recognize 1st time through
```

```
ReDim set1(16, 7), set2(16, 4), puV(zzz)
```

```
For z = 1 To zzz
```

```
getagrid 'gets prV assignments for this xpt
```

```
y = ppp(z, 4, 1): puV(z) = ppp(z, 3, 1) 'endowment & puV for this xpt
```

```
For grp = 1 To grps
```

```
For rd = 1 To rnds
```

```
For id = 1 To N
```

```
'1> total choice...2>G* choice...3>#G*... 4> -G* choice...5> # -G*... 6> +G*... 7>#
+G*
```

```
Select Case agrid(z, grp, rd, id) '[1.7... ALWAYS PICKS 1ST CASE????!]
```

```
Case Is < puV(z) + 0: set1(z, 1) = set1(z, 1) + xp(z, grp, rd, id, 1)
```

```
set1(z, 2) = set1(z, 2) + 1
```

```
Case Is > puV(z) + 5: set1(z, 5) = set1(z, 5) + xp(z, grp, rd, id, 1)
```

```
set1(z, 6) = set1(z, 6) + 1
```

```
Case Else: set1(z, 3) = set1(z, 3) + xp(z, grp, rd, id, 1) 'total choice in [0,5]
```

```
set1(z, 4) = set1(z, 4) + 1 'cases in [0,5]
```

```
End Select
```

```
Next id
```

```
Next rd
```

```
Next grp
```

```
'convert accumulated set1's to 2 decimals set2's: 1 > mean G; 2> # -G; 3 > mean -G*; 4
> #G*; 4> mean G*; 5> # +G*; 6> mean +G*
```

```
set2(z, 1) = Round((set1(z, 1) + set1(z, 3) + set1(z, 5)) / (nn * y), 2) 'grand mean
```

```
set2(z, 2) = Round(set1(z, 1) / (set1(z, 2) * y), 2) 'mean below [0,5]
```

```
set2(z, 3) = Round(set1(z, 3) / (set1(z, 4) * y), 2) 'mean in [0,5]
```

```
'ceiling in play for puV(z) = 15
```

```
If puV(z) < 15 Then _
```

```
set2(z, 4) = Round(set1(z, 5) / (set1(z, 6) * y), 2) 'mean above [0,5]
```

```
Next z
```

```
'dislay results
```

```
Set ace = [d4]
```

```
For z = 1 To zzz
```

```
ace(z, 1) = xpname(z)
```

```

    ace(z, 2) = ppp(z, 3, 1)
    ace(z, 3) = ppp(z, 4, 1)
Next z
ace(0, 1) = "Match": ace(0, 2) = "puV(z)": ace(0, 3) = " Y": ace(0, 4) = "XPavg": ace(0,
5) = "#"
    ace(0, 6) = "<[0,5]": ace(0, 7) = "#": ace(0, 8) = "in[0,5]": ace(0, 9) = "#": ace(0, 10)
= ">[0,5]"
'XSXSace(-1, 4) = 1: ace(-1, 5) = 2: ace(-1, 6) = 3: ace(-1, 7) = 4: ace(-1, 8) = 5: ace(-1,
9) = 6: ace(-1, 10) = 7
ppfill      'fills matrix with results
inputs = InputBox("do [1,5] or [0]?  1 = [1,5], 2 = [0], 3 = both, NULL = neither")
If inputs = 1 Then doA
If inputs = 2 Then doB
If inputs = 3 Then doC
[a1].Columns("A:C").EntireColumn.Delete Shift:=xlToLeft
Range("a4:c19").Interior.ColorIndex = 40
[a1].Select
End Sub

Sub ppfill()
ace(0, 1).Range("A1:R1").Select
    With Selection
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlBottom
    End With
ace(-1, -1).Select
With Selection
    .Range("f6:l6").Borders(xlEdgeBottom).LineStyle = xlContinuous
    .Range("f10:l10").Borders(xlEdgeBottom).LineStyle = xlContinuous
    .Range("f14:l14").Borders(xlEdgeBottom).LineStyle = xlContinuous
    .Range("H3:H18").Interior.ColorIndex = 40
    .Range("J3:J18").Interior.ColorIndex = 40
    .Range("L3:L18").Interior.ColorIndex = 40
    .Range("c1:m1").Font.FontStyle = "Bold"
    .Range("g3:l18").Select: boxcell
If inputs = 4 Then
    .Columns("c:i").ColumnWidth = 6: Columns("e:f").ColumnWidth = 3
    .Range("f4:f19").Borders(xlRight).LineStyle = xlContinuous
End If
End With
For z = 1 To zzz
    If inputs = 4 Then ace(z, 4) = set2(z, 1)  'XPmean
    ace(-1, 5) = 1: ace(-1, 6) = 2: ace(-1, 7) = 3: ace(-1, 8) = 4: ace(-1, 9) = 5: ace(-1, 10) =
6
    ace(z, 5) = set1(z, 2)  '#<[...]
    ace(z, 6) = set2(z, 2)  'mean for < [...]

```

```

ace(z, 7) = set1(z, 4)   '#in[...]'
ace(z, 8) = set2(z, 3)   ' mean in [...]'
ace(z, 9) = set1(z, 6)   '#>[...]'
ace(z, 10) = set2(z, 4) 'mean for > [...]'
' If IsNumeric(ace(z, 9)) Then ace(z, 11) = ace(z, 5) + ace(z, 7) + ace(z, 9) _
    Else ace(z, 11) = ace(z, 5) + ace(z, 7)
Next z
End Sub

Sub doA()      'need puV(z)
ReDim set1(16, 7), set2(16, 4)
For z = 1 To zzz
y = ppp(z, 4, 1): puV(z) = ppp(z, 3, 1)
For grp = 1 To grps
    For rd = 1 To rnds
        For id = 1 To N
            '1> total choice...2>G* choice...3>#G*... 4> -G* choice...5> # -G*... 6> +G*... 7>#
            +G*
            Select Case agrid(z, grp, rd, id)
                Case 1 To puV(z) - 0: set1(z, 1) = set1(z, 1) + xp(z, grp, rd, id, 1)
                    set1(z, 2) = set1(z, 2) + 1
                Case Is > puV(z) + 5: set1(z, 5) = set1(z, 5) + xp(z, grp, rd, id, 1)
                    set1(z, 6) = set1(z, 6) + 1
                Case Else: set1(z, 3) = set1(z, 3) + xp(z, grp, rd, id, 1)   'total choice in [0,5]
                    set1(z, 4) = set1(z, 4) + 1                               'cases in [0,5]
            End Select
        Next id
    Next rd
Next grp
'convert accumulated set1's to 2 decimals set2's: 1 > mean G; 2> # -G; 3 > mean -G*; 4
> #G*; 4> mean G*; 5> # +G*; 6> mean +G*
    set2(z, 1) = Round((set1(z, 1) + set1(z, 3) + set1(z, 5)) / (nn * y), 2) 'grand mean
    set2(z, 2) = Round(set1(z, 1) / (set1(z, 2) * y), 2) 'mean below [1,5]
    set2(z, 3) = Round(set1(z, 3) / (set1(z, 4) * y), 2) 'mean in [1,5]
'ceiling in play for puV(z) = 15
    If puV(z) < 15 Then _
        set2(z, 4) = Round(set1(z, 5) / (set1(z, 6) * y), 2) 'mean above [1,5]
Next z

'dislay results
Set ace = ace(1, 8)
ace(0, 5) = "#": ace(0, 6) = "<[1,5]": ace(0, 7) = "#": ace(0, 8) = "[1,5]": ace(0, 9) = "#"
ace(0, 10) = "> [1,5]"
'XSXS: ace(-1, 4) = 1: ace(-1, 5) = 2: ace(-1, 6) = 3: ace(-1, 7) = 4: ace(-1, 8) = 5: ace(-1,
9) = 6: ace(-1, 10) = 7
ppfill

```

End Sub

Sub doB()

ReDim set1(16, 7), set2(16, 4)

For z = 1 To zzz

y = ppp(z, 4, 1): puV(z) = ppp(z, 3, 1)

For grp = 1 To grps

For rd = 1 To rnds

For id = 1 To N

'1> total choice...2>G* choice...3>#G*... 4> -G* choice...5> # -G*... 6> +G*... 7>#
+G*

Select Case agrid(z, grp, rd, id)

Case Is < puV(z) + 0: set1(z, 1) = set1(z, 1) + xp(z, grp, rd, id, 1)

set1(z, 2) = set1(z, 2) + 1

Case Is > puV(z) + 0: set1(z, 5) = set1(z, 5) + xp(z, grp, rd, id, 1)

set1(z, 6) = set1(z, 6) + 1

Case Else: set1(z, 3) = set1(z, 3) + xp(z, grp, rd, id, 1) 'total choice in puV(z)

set1(z, 4) = set1(z, 4) + 1 'cases in puV(z)

End Select

Next id

Next rd

Next grp

'convert accumulated set1's to 2 decimals set2's: 1 > mean G; 2> # -G; 3 > mean -G*; 4 >
#G*; 4> mean G*; 5> # +G*; 6> mean +G*

set2(z, 1) = Round((set1(z, 1) + set1(z, 3) + set1(z, 5)) / (nn * y), 2) 'grand mean

set2(z, 2) = Round(set1(z, 1) / (set1(z, 2) * y), 2) 'mean below puV(z)

set2(z, 3) = Round(set1(z, 3) / (set1(z, 4) * y), 2) 'mean in puV(z)

set2(z, 4) = Round(set1(z, 5) / (set1(z, 6) * y), 2) 'mean above puV(z)

Next z

'dislay results

Set ace = ace(1, 8)

ace(0, 5) = "#": ace(0, 6) = "<puV": ace(0, 7) = "#": ace(0, 8) = "HIT": ace(0, 9) = "#"

ace(0, 10) = "> puV"

ppfill

'checkagrid

End Sub

Sub doC()

doA

doB

End Sub

Sub checkagrid()

'check agrid distribution

Dim check(0 To 16, 0 To 20)

```

For i = 1 To 20
    check(0, i) = i
Next i
For z = 1 To zzz
    check(z, 0) = z
For grp = 1 To grps
    For rd = 1 To rnds
        For id = 1 To N
            check(z, agrid(z, grp, rd, id)) = check(z, agrid(z, grp, rd, id)) + 1
        Next id: Next rd: Next grp: Next z
Set beta = [e23]
For z = 0 To zzz
    For j = 0 To 20
        beta(z, j) = check(z, j)
    Next j
Next z
'Selection.Interior.ColorIndex = 40
'Selection = check
End Sub

```

```

Sub xsp() 'SCRAP
Set beta = [a23]
For z = 0 To zzz
    For i = 1 To 10
        For k = 1 To 4
            For j = 1 To 20
                If z > 0 Then check(z, j) = check(z, j) + 1 Else beta(0, i) = i
            Next j
        Next i
    Next z
'ace.Range(Cells(1, 1), Cells(grps, 1)).NumberFormat = "0"

End Sub

```

```

Sub sort221()
Set ace = [c1]
For i = 1 To 95
    ace(i, 7) = 0.25 * (WorksheetFunction.sum(ace(i, 1), ace(i, 2), ace(i, 3), ace(i, 4), ace(i, 5)) - WorksheetFunction.Min(ace(i, 1), ace(i, 2), ace(i, 3), ace(i, 4), ace(i, 5)))
Next i
End Sub

```

```

Sub pupol221()
Dim DEUCE
Set ace = [c2]: Set DEUCE = [j2]
For i = 1 To 95

```

```

For j = 1 To 6
  Select Case ace(i, j)
    Case Is = "A": DEUCE(i, j) = 10
    Case Is = "A-": DEUCE(i, j) = 9
    Case Is = "B+": DEUCE(i, j) = 8
    Case Is = "B": DEUCE(i, j) = 7
    Case Is = "B-": DEUCE(i, j) = 6
    Case Is = "C+": DEUCE(i, j) = 5
    Case Else: DEUCE(i, j) = 4
  End Select
Next j: Next i
End Sub

Sub red()
' Keyboard Shortcut: Ctrl+r
  Selection.Font.ColorIndex = 3
End Sub

Sub boxcell()
  With Selection
    .Borders(xlEdgeLeft).Weight = xlMedium
    .Borders(xlEdgeTop).Weight = xlMedium
    .Borders(xlEdgeBottom).Weight = xlMedium
    .Borders(xlEdgeRight).Weight = xlMedium
  End With
End Sub

Sub zzUTILITY()
'retrieves "zz" from xpname(z) OR xpname(z) from "zz" OR lists all
newsheet
getzlist
inputs = InputBox("Enter experiment # or name" & vbCr & vbCr & "or NULL for list")
'FULL SET PRINTOUT
Set ace = Range("a1")
If inputs = "" Then
  ace = "ZZ LIST"
  For z = 1 To uuu
    j = z Mod 20: k = Int(z / 20)
    ace(j + 1, k * 5 + 1) = z
    ace(j + 1, k * 5 + 2) = zlist(z, 1)
  Next z
  Exit Sub
End If
'for ZZ input --> CODE output

```

```

If IsNumeric(inputs) Then
    MsgBox ("zlist(" & inputs & ")" & " --> " & zlist(inputs, 1))
    Exit Sub
Else:
'for CODE input --> ZZ output
z = 1
    Do
        If zlist(z, 1) = inputs Then
            MsgBox (inputs & "--> zlist(" & z & ")")
            Exit Do
        End If
        z = z + 1
    Loop
End If
End Sub

```

```

Sub startup()
With CommandBars("standard")
    .Controls("New").Delete
    .Controls("print (fax)").Delete
    .Controls("save").Delete
'    .Controls("search").Delete
'    .Controls("spelling").Delete
    .Controls("cut").Delete
    .Controls("copy").Delete
    .Controls("undo").Delete
    .Controls("redo").Delete
'    .Controls("insert hyperlink").Delete
    .Controls("autosum").Delete
    .Controls("sort ascending").Delete
    .Controls("sort descending").Delete
    .Controls("chart wizard").Delete
    .Controls("print preview").Delete
    .Controls("open").Delete
    .Controls("zoom:").Delete
'    .Controls("insert hyperlink").Delete
    .Controls("paste").Delete
End With
With CommandBars("formatting")
'    .Controls("font").Delete
    .Controls("currency style").Delete
    .Controls("percent style").Delete
    .Controls("italic").Delete
    .Controls("underline").Delete
    .Controls("bold").Delete
    .Controls("align left").Delete

```

```

        .Controls("align right").Delete
        .Controls("center").Delete
        .Controls("decrease indent").Delete
    ' .Controls("decrease indent").Delete
    ' .Controls("New").Delete
End With
With CommandBars("visual basic")
    .Controls("microsoft script editor").Delete
    .Controls("design mode").Delete
    .Controls("control toolbox").Delete
    .Controls("visual basic editor").Delete
    .Controls("security...").Delete
End With
End Sub

```

```

Sub newsheet()
' sets sht at .75 zoom, cell wide 5, light borders
    Set sht = Sheets.Add
    Range("a1").Select
    ActiveWindow.Zoom = 75
    With Cells
        .ColumnWidth = 5
        .Borders(xlDiagonalDown).LineStyle = xlNone
        .Borders(xlDiagonalUp).LineStyle = xlNone
        .Borders(xlEdgeLeft).LineStyle = xlNone
        .Borders(xlEdgeTop).LineStyle = xlNone
        .Borders(xlEdgeBottom).LineStyle = xlNone
        .Borders(xlEdgeRight).LineStyle = xlNone
        .Borders(xlInsideVertical).LineStyle = xlNone
        .Borders(xlInsideHorizontal).LineStyle = xlNone
    End With
End Sub

```

```

Sub border()
'sets thin border around selection
    With Selection
        .Borders(xlEdgeLeft).Weight = xlThin
        .Borders(xlEdgeTop).Weight = xlThin
        .Borders(xlEdgeBottom).Weight = xlThin
        .Borders(xlEdgeRight).Weight = xlThin
    End With
End Sub

```

```

Sub center()
'merge & center
    With Selection
        .HorizontalAlignment = xlCenter
    End With
End Sub

```

```

        .VerticalAlignment = xlBottom
        .ShrinkToFit = True
        .ReadingOrder = xlContext
        .MergeCells = False
    End With
    Selection.Merge
End Sub

Sub printtest()
    Application.ScreenUpdating = False
    Select Case testname
        Case "ud": Range("n2:X28").Select
            border
            ActiveSheet.PageSetup.PrintArea = "$B$2:$X$28"
            ActiveWindow.Zoom = 0.75
        Case Else: InputBox ("Test to print")
    End Select
    Selection.PrintOut Copies:=2
End Sub

Sub holdit()
With ActiveSheet.PageSetup
    .LeftMargin = Application.InchesToPoints(0.75)
    .RightMargin = Application.InchesToPoints(0.75)
    .TopMargin = Application.InchesToPoints(1)
    .BottomMargin = Application.InchesToPoints(1)
    .HeaderMargin = Application.InchesToPoints(0.5)
    .FooterMargin = Application.InchesToPoints(0.5)
    .PrintQuality = 600
    .Orientation = xlLandscape
    .PaperSize = xlPaperLetter
    .FirstPageNumber = xlAutomatic
    .Order = xlDownThenOver
    .FitToPagesWide = 1
    .FitToPagesTall = 1
    .PrintErrors = xlPrintErrorsDisplayed
End With
End Sub

Sub frame()
'puts frame around selection
With Selection
    .Borders(xlEdgeLeft).Weight = xlMedium
    .Borders(xlEdgeTop).Weight = xlMedium
    .Borders(xlEdgeBottom).Weight = xlMedium
    .Borders(xlEdgeRight).Weight = xlMedium

```

End With
End Sub

Sub getgrps()

'assembles summary data by group for 'show' & chart of group mean vs. grp avg n.s.d.
choices = grps

Dim gmean As Variant, fgx() As Variant

ReDim fgx(zzz, grps, 4), xarray(grps), yarray(grps)

' fgx 1.is grp, 2. is mean s.d., 3. is grandmean 'choice' ratio for group, 4. is overall
'getratio

For grp = 1 To grps

fgx(z, grp, 1) = grp

For rd = 1 To rnds

fgx(z, grp, 3) = fgx(z, grp, 3) + mx(z, grp, rd, 1) / (N * y * rnds)

' ^accumulates mean choice for this grp

If mx(z, grp, rd, 1) = 0 Then

mx(z, grp, rd, 1) = 1

mx(z, grp, rd, 2) = 1

End If

' ^avoids div by 0 later

fgx(z, grp, 2) = fgx(z, grp, 2) + N * mx(z, grp, rd, 2) / mx(z, grp, rd, 1)

' ^accumulates avg normalized st. dev for this group... for each rd, adds (1/rnds) *

(s.d./pool for that rd)

Next rd

fgx(z, grp, 2) = fgx(z, grp, 2) / rnds 'converts sum of nsd's to avg nsd

' overall 'getratio' for group = n*a*meanchoice + 1-meanchoice

fgx(z, grp, 4) = N * a * fgx(z, grp, 3) + (1 - fgx(z, grp, 3))

'???If fgx(z, grp, 3) = 0 Then fgx(z, grp, 3) = "****" 'Else ace(grp,2) = fgx(z,grp,2) /

fgx(z,grp,1)

Next grp

'set arrays for nsd chart

Set ace = Range("g3"): ace.Activate

For grp = 1 To grps

xarray(grp) = fgx(z, grp, 2)

yarray(grp) = fgx(z, grp, 3)

Next grp

Chart

With ActiveSheet.Shapes("Chart 1")

.ScaleWidth 0.55, msoFalse, msoScaleFromTopLeft

.ScaleHeight 0.81, msoFalse, msoScaleFromTopLeft

.IncrementLeft 43.5

.IncrementTop 75#

End With

'movesumchart

'paste data

ace.Activate

```

If zzz = 1 Then z = 1
If z = 1 Then ReDim mxd(zzz, maxgrps, maxrnds, 3)
'ace(-1, 1) = xplist2(z)
ace(0, 1) = "GRP": ace(0, 2) = "n.s.d.": ace(0, 3) = "gmean": ace(0, 4) = "getratio"
' n.s.d. here is group's normalized s.d. (= s.d./pool) averaged across all rounds
ace.Range(Cells(1, 2), Cells(grps, 4)).NumberFormat = "0.00"
ace.Range(Cells(1, 1), Cells(grps, 1)).NumberFormat = "0"
'make fgx into 2D for pasting
Dim fgy(): ReDim fgy(grps, 4)
For grp = 1 To grps
    For i = 1 To 4
        fgy(grp, i) = fgx(1, grp, i)
    Next i
Next grp
'paste sum data
ace.Range(Cells(1, 1), Cells(grps, 4)) = fgy
'SORT by gmean
ace.Range(Cells(1, 1), Cells(grps, 4)).Select
Selection.Sort Key1:=Range("j1"), order1:=xlDescending
'frame around 'groups' summary
Range("h2:k19").Select
frame
'move display to right if NOT showing full data
End Sub

```

```

Sub holdsubforSUM()

```

```

If status < 2 Then GoTo SKIP3
'PASTE
'sets ACE for by-groups summary for 'showbygroups'
If status = 1 Then Set ace = ace(1, 7) 'correct ace when this a sub in 'showdata'
If status < 2 Then Set ace = ActiveCell(2, 1)
ace = "GRP": ace(1, 2) = "NSD": ace(1, 3) = "EFF": ace(0, 1) = xpname
ace = ace(2, 1)
ReDim chartdata(grps, 2): Dim grpj
For z = 1 To zzz
    setup
    For grp = 1 To grps
        For j = 1 To 3
            If j > 1 Then chartdata(grp, j - 1) = groupsx(grp, j)
            ace(grpj) = groupsx(grp, j)
        Next j
    Next grp: Next z
ace.Range(Cells(1, 2), Cells(grps, 3)).Select
    Selection.NumberFormat = "0.00"
End Sub

```

```

Sub sortN2groups()      ' N=2 utility
'needed for n = 2 groups, since spreadsheet does not need
'explicit groups but we do, since original needs player serial
' IDs to handle accounting (payments), but we can discard that
' and use explicit groups since Tmacros then more flexible across xpts effecient
Dim us As Byte, them As Byte, xrd As Integer
Dim s1 As Integer, s2 As Integer, s3 As Integer, s4 As Integer
Dim reps As Byte, grp As Byte, rng As Range
N = 2 'this sub usual needed only for n=2
nn = 14 'InputBox("number of players per session")
us = 5 'InputBox("columns left of 'groups' to player ID") - 1 "'-1" since 1st col to left is
0
them = 4 'InputBox("columns left of 'groups' to other ID") - 1 ' same
grps = nn / N
'Stop 'set ACE
Set ace = Range("h6")
xrds = 200 'InputBox("total rounds in spreadsheet")
rnds = 10
reps = 1 'FIX.....
ReDim group(nn * xrds, 1) As Variant
For xrd = 1 To xrds
s1 = (xrd - 1) * nn 'sets round counter for next round1 for this rd
  For z = 1 To zzz
  setup
  For grp = 1 To grps
    'set 1st member of this pair
    s2 = 1 'counter for next open slot
    Do While group(s1 + s2, 1) > 0 'ends loop at 1st empty cell in "grp"
      s2 = s2 + 1
    Loop
    group(s1 + s2, 1) = grp 'sets 1st member of this pair
    s3 = s2 + 1 'start to look for 2nd of pair
    'find & assign 2nd of this pair
    Do While ace(s1 + s2, -us) <> ace(s1 + s3, -them)
      s3 = s3 + 1
    Loop
    group(s1 + s3, 1) = grp
  Next grp: Next z
'check for error in this round
  s4 = 0 'reset counter to use for summing groups, which shld be 2Xsum(1... nn/2)
  For i = 1 To nn
    s4 = s4 + group(s1 + i, 1)
  Next i
  If s4 <> (nn / 2) * (nn / 2 + 1) Then MsgBox ("ERROR in Rd" & xrd)
Next xrd

```

```

'paste groups
'checkpartners    'checks that partners in last round are same as in Rd 1
For i = 1 To s1 + nn
    ace(i) = group(i, 1)
Next i
End Sub

```

```

Sub mspayoffs()
'for morgan/selten 'lottery' xprt
Dim i As Byte, j As Byte, zzzz() As Variant, rng As Range, lotteryvalue As Byte
lotteryvalue = InputBox("value of lottery")
ReDim zzzz(0 To 10, 0 To 10) As Variant
If lotteryvalue = 0 Then Set ace = [b2] Else Set ace = [b15]
For i = 0 To 10    'vt own 'choice'
    ace(i + 1, 0) = i
    For j = 0 To 10    'hz other's 'choice'
        If i = 10 Then ace(0, j + 1) = j
        If i = 0 And j = 0 Then zzzz(i, j) = 10
        If i = 0 And j = 0 Then GoTo skip
        zzzz(i, j) = (10 - i) + (i / (i + j)) * lotteryvalue + 0.75 * (i + j)
skip: If lotteryvalue = 0 Then zzzz(i, j) = zzzz(i, j) + 6
    Next j
Next i
Set rng = ace.Range(Cells(1, 1), Cells(11, 11))
rng = zzzz
rng.NumberFormat = "0.00"

```

```

Dim yyyy(0 To 10, 0 To 10) As Variant
For i = 0 To 10
    For j = 0 To 10
        yyyy(i, j) = zzzz(i, j) + zzzz(j, i)
    Next j
Next i
Set rng = ace.Range(Cells(1, 14), Cells(11, 24))
rng = yyyy
End Sub

```

```

Sub sleep()
'speeds return of results by turning off screen updating
Application.ScreenUpdating = False
End Sub

```

```

Sub wakeup()
'speeds return of results by turning off screen updating
Application.ScreenUpdating = True
End Sub

```

```

Sub clearsheet()
' Keyboard Shortcut: Ctrl+Shift+X
'clears cells but leaves embedded charts
Application.ScreenUpdating = False
    Cells.Clear
    ActiveCell.Select
End Sub

```

```

Sub fgx() 'FIRSTGRAND test [12.14... 'fg'does not want to work as name of sub!,
hence fgx
Application.ScreenUpdating = False
'gmean is group across rnds>2, grandmean is mean across grps, rd1mean is rd1 only all
grps
Dim gmean, grandmean!, fgx(), i%, j%, xpmean!, rd1data(), tester%
Dim ggmean() 'ggmean is mean choice across grps for rnds>1
Dim cat%, result(), rd1sd(), rd1mean(), rd1sdmean()
ReDim result(zzz, maxgrps), rd1sd(zzz), rd1mean(zzz), rd1sdmean(zzz), ggmean(zzz),
rd1data(zzz, maxgrps * maxn)
'if 3 trials: mean VS. Rd 1: A) n.s.d.; B) mean; C) s.d., if 1 trial nsd
trials = 3: tester = 0 'set test (charts = 0, categories=1)
'reset cat counters & redim allgrps
For i = 1 To 9
    fgcat(i) = 0
Next i
ReDim allgrps(trials, countgrps), fgx(zzz, countgrps, 4)

```

```

'start test
For z = 1 To zzz
setup 'gets parameters for xpt z
'extend xpparray data for the zth xpt
    mxdx
ggmean(z) = 0 'set accumulator for mean across all rnds > 1 for all groups
grandmean = 0 'set accumulator for mean across all rnds (incl 1) & all grps
'set fgx array: 1.is mean for rd 1... 2. is s.d./y for rd 1... 3. s.d./mean... 4.is gmean rds
2-10
For grp = 1 To grps
    gmean = 0 'set accumulator for mean across all rnds > 1 for this group
    For rd = 1 To rnds
        grandmean = grandmean + mxd(z, grp, rd, 1) / (N * y * grps)
        If rd > 1 Then GoTo xround
        fgx(z, grp, 1) = mxd(z, grp, 1, 1) / (N * y)
        'next sets rd 1 s.d./y & s.d./m allowing for (rare) case where rd 1 pool is 0
        If (mxd(z, grp, rd, 1) > 0) Then fgx(z, grp, 2) = mxd(z, grp, 1, 2) _
            Else fgx(z, grp, 2) = 1 's.d./y except if pool (hence also s.d.)is 0, set to 1
    
```

```

    If (mxd(z, grp, rd, 1) > 0) Then fgx(z, grp, 3) = mxd(z, grp, 1, 2) * N / mxd(z,
grp, 1, 1) _
    Else fgx(z, grp, 2) = 1 's.d./m except if pool is 0, set to 1
    'if s.d.=0 with m > 0, changes s.d./y & s.d./m from 0 to .01 to allow usual log
regression
    If (mxd(z, grp, rd, 2) = 0 And mxd(z, grp, rd, 1)) > 0 Then
        fgx(z, grp, 3) = 0.01: fgx(z, grp, 2) = 0.01
    End If
'sum fraction of full choice summed across rds 2ff to set fgtest dependent variable
around:  If rd > 1 Then gmean = gmean + mxd(z, grp, rd, 1) / ((rnds - 1) * y)
    If rd = rnds Then fgx(z, grp, 4) = gmean / N
    Next rd
'accumulate mean>1 across grps xpt z
ggmean(z) = ggmean(z) + fgx(z, grp, 4)
Next grp

'get rd1sdmean (all players) = rd1(s.d.)/rd1(mean) for this xpt
s1 = 0
For grp = 1 To grps
    For id = 1 To N
        s1 = s1 + 1
        rd1data(z, s1) = xp(z, grp, 1, id, 1)
    Next id
Next grp
rd1sd(z) = WorksheetFunction.StDevP(rd1data)
rd1mean(z) = WorksheetFunction.Average(rd1data)
If rd1mean(z) = 0 Then MsgBox (" rd 1 ZERO... " & xpname(z))
If rd1mean(z) > 0 Then rd1sdmean(z) = rd1sd(z) / rd1mean(z) Else rd1sdmean(z) = 1

'testing
Set ace = Range("c3")
'adjust per grp ggmean(z)
ggmean(z) = ggmean(z) / grps
'debug print
'For grp = 1 To grps
'    ace(grp, z) = fgx(z, grp, 4)
'    ace(grp, z + 2) = fgx(z, grp, 4) / ggmean(z)
'Next grp

'assign grps to categories
If tester = 1 Then
    For grp = 1 To grps
        result(z, grp) = fgx(z, grp, 4) / ggmean(z)
        Select Case fgx(z, grp, 3) / rd1sdmean(z)
            Case Is < 0.5: If result(z, grp) > 1.5 Then fgc(1) = fgc(1) + 1 _

```

```

        Else If result(z, grp) < 0.5 Then fgcat(3) = fgcat(3) + 1 Else fgcat(2) =
fgcat(2) + 1
        Case Is > 1.5: If result(z, grp) > 1.5 Then fgcat(7) = fgcat(7) + 1 _
        Else If result(z, grp) < 0.5 Then fgcat(9) = fgcat(9) + 1 Else fgcat(8) =
fgcat(8) + 1
        Case Else: If result(z, grp) > 1.5 Then fgcat(4) = fgcat(4) + 1 _
        Else If result(z, grp) < 0.5 Then fgcat(6) = fgcat(6) + 1 Else fgcat(5) =
fgcat(5) + 1
        End Select
    Next grp
End If
Next z

```

```

'for category test
If tester = 1 Then
    Set ace = Range("c20")
    k = 0
    ace(3, -1) = "NSD": ace(-1, 2) = "MEAN>1": ace(0, 1) = "hi": ace(0, 2) = "mid":
ace(0, 3) = "low"
    ace(1, 0) = "low": ace(2, 0) = "mid": ace(3, 0) = "high"
    For i = 1 To 3
        For j = 1 To 3
            k = k + 1
            ace(i, j) = fgcat(k)
        Next j
    Next i
    Exit Sub
End If

```

```

'debuchoicey...
Set ace = Range("c3")
ace(0, 0) = "zzz=" & zzz
    For z = 1 To zzz
        ace(i, 1) = rd1sd(z)
        ace(i, 2) = rd1mean(z)
        ace(i, 3) = rd1sdmean(z)
        ace(i, 4) = xplist2(z)
    Next z

```

```

'for fg charts, assign normalized grp result to output xpXcountgrpsXdata... 1,2,3=xvalues
by trial, 4= yvalue
Dim previousgrps%
'previousgrps counts groups already in chartseries
ReDim allgrps(countgrps, 4)
previousgrps = 0 'reset
    "previousgrps" keeps count of grps (from which to insert more here)

```

```

For z = 1 To zzz
setup
For grp = 1 To grps
' note that indexes shift to order used in display (from order in which constructed)
  allgrps(previousgrps + grp, 1) = fgx(z, grp, 3) / rd1sdmean(z) 'n.s.d. (= s.d./mean) rd
  1
  allgrps(previousgrps + grp, 2) = fgx(z, grp, 1) 'mean choice rd 1
  allgrps(previousgrps + grp, 3) = fgx(z, grp, 2) / y 's.d./y rd 1
  '6.29 +normed rel to mean>1 choice
  allgrps(previousgrps + grp, 4) = fgx(z, grp, 4) / ggmean(z) 'normalized mean
choice rds 2 to end
Next grp

```

```

Dim allgrpavg: allgrpavg = 0
For grp = 1 To grps
  allgrpavg = allgrpavg + allgrps(previousgrps + grp, 4) / grps
Next grp
previousgrps = previousgrps + grps
Next z

```

'make set individual choice xarrays for each grp for plots of appropriate case...

```

SKIP1: For trial = 1 To trials
ReDim xarray(countgrps), yarray(countgrps)
' set yvalues series
For grp = 1 To countgrps
  yarray(grp) = allgrps(grp, 4) '= normed mean>rd1
Next grp
'set xvalues series & other chart inputs
ytitle = "mean choice rds 2 to last"
choices = countgrps
'If zzz = 1 Then xpname(z) = xplist2(1) Else xpname(z) = zzz & "xpts" "???" 9.04
Select Case trial
  Case 1: chartyyy = "fg(nsd)" 'mean>rd 1 VS. n.s.d. (= s.d./mean) rd 1
    If zzz > 1 Then
      xtitle = "Rd 1 s.d./mean" & vbCr & "M=" & Int(100 * grandmean / 100)
    Else: xtitle = "Rd 1 s.d./mean" & vbCr & "M=" & Int(100 * grandmean / 100)
    End If
    For grp = 1 To countgrps
      xarray(grp) = allgrps(grp, 1)
    Next grp
    high = 1: wide = 0.8: down = 15.1: across = 9
  '
  ' If zzz > 0 Then
'..... ADD LEGEND "for " & z & "XPs & " & countgrps & " grps"
'..... non-default conditions for this test
  Case 2: chartyyy = "fg(m)" 'mean>1 VS mean choice rd 1
    xtitle = "Rd 1 mean"

```

```

    For grp = 1 To countgrps
        xarray(grp) = allgrps(grp, 2)
    Next grp
    high = 1: wide = 0.6: down = 15: across = 9
    Case 3: chartyyy = "fg(sd/y)" 'mean>1 VS s.d. rd 1 normalized by endowment
        xtitle = "Rd 1 s.d./y"
        For grp = 1 To countgrps
            xarray(grp) = allgrps(grp, 3)
        Next grp
        high = 1: wide = 0.8: down = 15: across = 8
    End Select
'make charts (nsd, sd, m)
    chartindex = trial
    Chart
    'photo
    chartplace
Next trial
adjustfg
Set beta = Range("b22")
Range("a1").Activate
End Sub

Sub adjustfg()
'moves fgm chart a bit right
If trials = 1 Then Exit Sub
    With ActiveSheet
        .Shapes("Chart 2").IncrementLeft 32.25
        .Shapes("Chart 2").IncrementTop 2.25
    End With
End Sub

Sub chartranges()
'paste chart arrays into ranges
    Set xrange = rng.Offset(0, trial - 1)
    xrange = WorksheetFunction.Transpose(allgrps)
    Set yrange = rng.Offset(grps, trial - 1)
    yrange = WorksheetFunction.Transpose(yyarray)
End Sub

Sub fgpaste()
'paste data
ace(1, 7) = xpname(z) & "/" & runname
'ace.Range(Cells(1, 1), Cells(grps, 6)).NumberFormat = "0.00"
    For z = 1 To zzz
        setup
        For grp = 1 To grps

```

```

ace(grp, 1) = fgx(z, grp, 1)
ace(grp, 2) = fgx(z, grp, 3)
If fgx(z, grp, 2) > 0 Then ace(grp, 3) = fgx(z, grp, 2) Else ace(grp, 3) = 0.1
' ^ "if" allows usual log regression even if all grps are same, hence s.d. = 0
ace(grp, 4) = fgx(z, grp, 3)
If fgx(z, grp, 2) > 0 Then ace(grp5) = fgx(z, grp, 2) / fgx(z, grp, 1) Else ace(grp5) =
0.1
' ^ "if" allows usual log regression even if all choices are same, hence s.d. = 0
' ace(grp5) = fgx(z,grp,2) / fgx(z,grp,1)
ace(grp6) = fgx(z, grp, 3)
Next grp: Next z
ace.Range(Cells(1, 1), Cells(grps, 2)).Select
End Sub

```

```

Sub findbottom()
'find bottom of previous result & paste data there
Set ace = Range("a2")
Do Until ace(1, 1).Value = ""
Set ace = ace(2, 1)
Loop
End Sub

```

```

Sub shtfix()
ActiveSheet.Delete
Sheets.Add
Set sht = ActiveSheet
sht.Name = sheetname
End Sub

```

```

Sub fixforarrayfailures()
Set ace = Range("h2")
Application.ScreenUpdating = False
If trial = 0 Then j = 1 Else j = trial - 1
With ace
Set xrange = .Range(.Offset(1, 1), .Offset(choices, 1))
Set yrange = .Range(.Offset(1, 2), .Offset(choices, 2))
'mark bottom of xrange
'.Offset(choices, 20 + j).Interior.ColorIndex = 6
End With
xrange = WorksheetFunction.Transpose(xarray)
yrange = WorksheetFunction.Transpose(yarray)
'set zseries via pasted ranges
zseries.Values = yrange
zseries.XValues = xrange
End Sub

```

```

Sub Chart()

```

```

Application.ScreenUpdating = False
Set sht = ActiveSheet
If sheetname = "" Then sheetname = sht.Name
Set cht = Charts.Add
Set cht = cht.Location(Where:=xlLocationAsObject, Name:=sheetname)
Set zseries = cht.SeriesCollection.NewSeries
'pastes x,y arrays into spreadsheet ranges when EXCEL somehow! won't accept array
inputs
i = 0 'DEBUGG ONERR 8.6
On Error GoTo ERR
zseries.Values = yarray '7.14 test error routine
zseries.XValues = xarray
SKIP1: 'CANCEL ERR HERE
With cht 'CIRCULARHERE!
.ApplyCustomType ChartType:=xlUserDefined, TypeName:=testname
.HasTitle = True
With .ChartTitle
If zzz = 1 Then .Text = xplist2(zzz) & " " & testname Else _
.Text = testname & "..."
.AutoScaleFont = False
With .Font
.Name = "Arial"
.FontStyle = "Bold"
.Size = 8
End With
End With
.Axes(xlCategory, xlPrimary).HasTitle = True
.Axes(xlCategory, xlPrimary).AxisTitle.Characters.Text = xtitle
.Axes(xlValue, xlPrimary).HasTitle = True
.Axes(xlValue, xlPrimary).AxisTitle.Characters.Text = ytitle
End With
If testname = "show" Then GoTo SKIP2
With cht.Parent
.ShapeRange.ScaleWidth wide, msoFalse, msoScaleFromTopLeft
.ShapeRange.ScaleHeight high, msoFalse, msoScaleFromBottomRight
End With
'chartplace
SKIP2: Exit Sub
ERR: i = i + 1
If i > 4 Then Stop 'why this lockup? [12.14]
fixforarrayfailures
Resume SKIP1
End Sub

Sub trendline()
Set cht = ActiveChart

```

```

inputs = InputBox("Choose trendline as 1 (linear), 2 ( log) 3 (power) " & _
                vbCr & "4 (exponential) 5 (moving average)" & _
                vbCr & "IF no prior trendline to replace return double.. 11, 22, etc.")
If Int(inputs / 10) = 0 Then
    cht.SeriesCollection(1).Trendlines(1).Delete
    Else: inputs = Int(inputs / 10)
End If
Select Case inputs
    Case 1: trendlinear
    Case 2: trendlog
    Case 3: trendpower
    Case 4: trendexp
    Case 5: trendmvgavg
    Case Else: MsgBox ("input ERROR"): Exit Sub
End Select
With cht.SeriesCollection(1).Trendlines(1).DataLabel
    .Left = 29
    .Top = 226
    .AutoScaleFont = False
    With .Font
        .Name = "Arial"
        .FontStyle = "Bold"
        .Size = 8
    End With
    .NumberFormat = "0.00"
End With
cht.ChartArea.Select
End Sub

Sub trendlog()
    cht.SeriesCollection(1).Trendlines.Add(Type:=xlLogarithmic, Forward _
        :=0, Backward:=0, DisplayEquation:=True, DisplayRSquared:=True).Select
End Sub

Sub trendlinear()
    cht.SeriesCollection(1).Trendlines.Add(Type:=xlLinear, Forward _
        :=0, Backward:=0, DisplayEquation:=True, DisplayRSquared:=True).Select
End Sub

Sub trendexp()
    cht.SeriesCollection(1).Trendlines.Add(Type:=xlExponential, Forward _
        :=0, Backward:=0, DisplayEquation:=True, DisplayRSquared:=True).Select
End Sub

Sub trendpower()
    cht.SeriesCollection(1).Trendlines.Add(Type:=xlPower, Forward _

```

```
:=0, Backward:=0, DisplayEquation:=True, DisplayRSquared:=True).Select  
End Sub
```

```
Sub trendmvgavg()  
    periods = InputBox("how many periods for moving average?")  
    cht.SeriesCollection(1).Trendlines.Add(Type:=xlmvgaverage, Forward_  
        :=0, Backward:=0, DisplayEquation:=True, DisplayRSquared:=True).Select  
End Sub
```

```
Sub bigarray()  
    For i = 1 To big  
        Set zseries = cht.SeriesCollection.NewSeries  
        zseries.Values = yiarray  
        zseries.XValues = xiarray  
    End Sub
```