




Creating Customized Data Files in E-Prime: Practical Tutorial

Osman İyilikci ^a, 

^aCyprus International University, Department of Psychology

Abstract ■ There are packages that simplify experiment generation by taking advantage of graphical user interface. Typically, such packages create output files that are not in an appropriate format to be directly analyzed with a statistical package. For this reason, researchers must complete several time consuming steps, and use additional software to prepare data for a statistical analysis. The present paper suggests a particular E-Basic technique which is time saving in data analysis process and applicable to a wide range of experiments that measure reaction time and response accuracy. The technique demonstrated here makes it possible to create a customized and ready-to-analyze data file automatically while running an experiment designed in E-Prime environment.

Keywords ■ Experiment programming, E-Basic, E-Prime. **Tools** ■ E-Prime.

Acting Editor ■ Denis Cousineau (Université d'Ottawa)

Reviewers
■ No reviewer.

 oiyilikci@ciu.edu.tr

 **ORCID** [OI: 0000-0001-5760-6319](https://orcid.org/0000-0001-5760-6319)

 **DOI** [10.20982/tqmp.14.1.p073](https://doi.org/10.20982/tqmp.14.1.p073)

Introduction

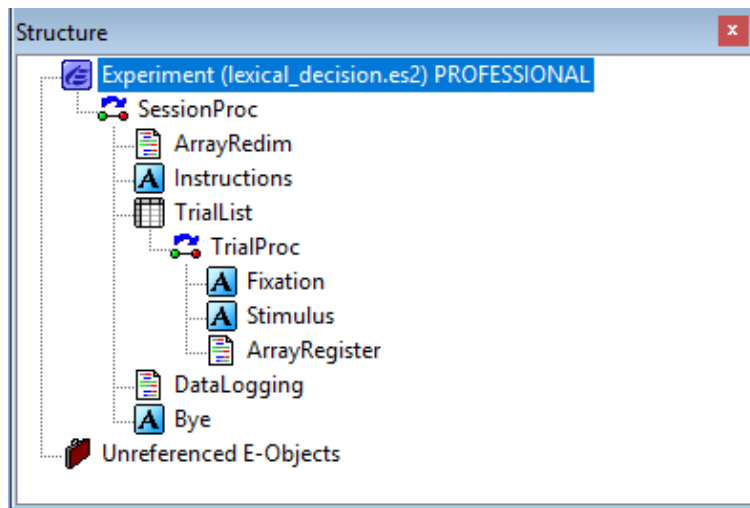
There are packages such as E-Prime (Psychology Software Tools, 2017), OpenSesame (Mathôt, Schreij, & Theeuwes, 2012) or SuperLab (Haxby, Parasuraman, Lalonde, & Abboud, 1993) that make designing psychology experiments easier by taking advantage of graphical user interfaces (GUI). However, the GUI environment may be limited and in some occasion, may fail to meet the researchers' need. In order to overcome this limitation of GUI, some of the packages have an underlying scripting language which makes the experiment design software more flexible. By virtue of the scripting language, researchers become capable of adding lines of code to the GUI environment so that more complex procedures can be programmed or additional features of the package can be used. For example, Python code and E-Basic code can be added to experiments designed using OpenSesame and E-Prime, respectively. Such scripting languages may be highly beneficial in cases where creating customized data files is an essential step in statistical analyses. By using E-Prime's scripting language, E-Basic, the present paper demonstrates a particular operation to create ready-to-analyze data files.

E-prime and almost all other experiment design packages create separate data files for each subject. Further-

more, each data file contains separate lines for each stimulus data. These lines indicate values such as reaction time, accuracy, onset time, etc. for every single stimulus. This configuration of data logging is required for detailed examination of computer's timing errors, device failures or any problems stemming from programming faults (e.g. inexact balancing of conditions). On the other hand, such a data arrangement is not convenient for statistical analysis as in most cases, hypothesis testing in psychology requires constructing one data row for each subject and one column for each stimulus data. For this reason, researchers must complete many steps in order to prepare data files for the analysis: 1) merging subject files into one master file; 2) filtering out data that is not relevant for statistical analyses (system time, trial and block numbers, file names etc.); 3) organizing a data table which is adequately formatted for analyzing data; 4) exporting data table into a text or excel file 5) importing text or excel file into a statistical package. These steps are time consuming; furthermore, researchers must also get acquainted with additional software to complete them successfully. For example, E-Merge and E-DataAid must be used to prepare data files that were created by E-Prime. To avoid these additional steps while preparing logged data for a statistical package, researchers can take advantage of E-Prime's scripting language E-Basic.



Figure 1 ■ Structure of the lexical decision experiment.



The present study aims to demonstrate how to create customized, ready-to-analyze data files by using E-Basic operations. In order to simplify the example shown here, E-Basic code was used in a simple one-block lexical decision task. Even if the demonstrated lexical decisions task is quite simple, the sample code is applicable to any experiment which measures accuracy and/or reaction time as a dependent variable. In the following sections, lexical decision task is described and line by line explanation of E-Basic code is demonstrated.

Description of the Lexical Decision Task Example

In a typical lexical decision experiment, participants categorize targets as words or nonwords (see Meyer & Schvaneveldt, 1971). On each trial of the task, participants are asked to make a decision about a stimulus which is composed of a string of letters. Participants need to decide whether or not the presented string is a meaningful word. After displaying instruction screen, 24 trials are presented. On each trial, a fixation dot (+ sign) is displayed for 1 sec. Following the fixation, the string (word, pseudoword or nonword) is displayed until participant responses. After all the strings are presented, a goodbye screen is displayed for 3 seconds. Fixation, strings and goodbye screen are presented using *TextDisplay* objects. The structure of the experiment is shown in Figure 1.

Characteristic of each trial (presented string, correct response, etc.) is defined using a *List* object (see Figure 2).

Descriptions of list columns are presented in Table 1.

As it can be seen in Table 1, there are two independent variables: stimulus type and string length. The experiment has a 3x2 design: stimulus type has three levels, whereas string length has two. In the *StimulusType* attribute of *TrialList*, 1 indicates word, 2 indicates pseudoword, 3 indicates nonword. For *StringLength* attribute, 1 indicates short, 2 indicates long. The E-Prime experiment is available on the journal’s web site.

Explanation of Code Lines

Four set of instructions were written. One set of instruction was written in the script window¹ and the other three were written in *InLine* objects.

Script Window Code All public variables were declared in the user tab of the script window. They were not declared locally in *InLine* objects because these variables must cumulatively hold particular values² or must be referred in several steps of the experiment. The lines of instructions written in the script window are seen in Listing 1.

There are two data types used in variable declarations: single and integer. Single data type can hold real numbers with up to seven digits. For this reason, the variables that necessitate calculations with real numbers were defined as single. Integer data type on the other hand, was used for counting the number of correct responses (which does not necessitate decimals). Table 2 shows descriptions of all

¹Script window can be viewed by clicking on Script under View menu.

² Local variables (variables which are defined in *InLine* object) erase themselves each time the *InLine* object operates.

³Empty parentheses which are next to variable name indicate the variable is a redimensionable array.



Figure 2 ■ List columns

Summary
 24 Samples (1 cycle x 24 samples/cycle)
 1 Cycle equals 24 samples
 Random Selection (Repeat After Reset Possible)

ID	Weight	Procedure	Nested	Stimulus	CorrectResponse	StimulusType	StringLenght	StimulusNumber
1	1	TrialProc		car	1	1	1	1
2	1	TrialProc		table	1	1	1	2
3	1	TrialProc		sky	1	1	1	3
4	1	TrialProc		fit	1	1	1	4
5	1	TrialProc		tip	1	1	1	5
6	1	TrialProc		show	1	1	1	6
7	1	TrialProc		computer	1	1	2	7
8	1	TrialProc		hospital	1	1	2	8
9	1	TrialProc		translate	1	1	2	9
10	1	TrialProc		refrigerator	1	1	2	10
11	1	TrialProc		allocate	1	1	2	11
12	1	TrialProc		temporary	1	1	2	12
13	1	TrialProc		taer	2	2	1	13
14	1	TrialProc		pluy	2	2	1	14
15	1	TrialProc		baor	2	2	1	15
16	1	TrialProc		yallerman	2	2	2	16
17	1	TrialProc		pumelad	2	2	2	17
18	1	TrialProc		simdayer	2	2	2	18
19	1	TrialProc		aeer	2	3	1	19
20	1	TrialProc		xzwa	2	3	1	20
21	1	TrialProc		rtsz	2	3	1	21
22	1	TrialProc		oaerrrt	2	3	2	22
23	1	TrialProc		iertswq	2	3	2	23
24	1	TrialProc		zvbanm	2	3	2	24

public variables.

As can be seen in the script window, the variables declared in Lines 1-4 are redimensionable arrays.³ The purpose of declaring these variables as arrays is that each of them must hold several values. For example `rt_array()` and `accur_array()` (Line 1) keep 24 different values for 24 stimulus/trials. Variables that hold dependent variable summaries (Lines 2-3) were also declared as arrays because each of them keeps track of six values as there are six conditions in the experiment. `ntrial_condition()` is another array which holds the number of trials for each experimental condition (Line 4).

InLine Object Codes Three *InLine* objects were written to create a customized data file: *ArrayRedim*, *ArrayRegister* and *DataLogging*.

In the object *ArrayRedim*, arrays were redimensioned to the size of *TrialList*. As seen in Line 6 and Line 7, the *Size* property of *TrialList* was used to determine the size of the arrays. Between Line 10 and Line 13, all variables were redimensioned to a 3x2 matrix as the experiment has

six (3x2) conditions. All code lines that were written in *ArrayRedim* are shown in Listing 2.

`rt_array()` and `accur_array()` could have been directly declared as 24 element arrays as there are 24 trials in the experiment. On the other hand, to be able to make any modification about number of trials without manipulating code lines, all references regarding number of trials in the code were made by calling *size* property of *TrialList*. Similarly, in Lines 10-13, arrays were redimensioned by referring to `level_IV1` and `level_IV2` in order to make further modifications about the number of independent variable levels effortlessly.

In Lines 14-19, the number of trials for each of all six experimental conditions was entered into `ntrial_condition()`.

In Line 20 and Line 21 of the object *ArrayRegister*, reaction time and accuracy data of each stimulus are recorded to `rt_array()` and `accur_array()` respectively. For reaction time, we use the *RT* property; for accuracy, we use the *ACC* property of the *Stimulus* object. Because the or-



Listing 1 ■ The instructions contained in the script window.

```

1 Dim rt_array() As Single, accur_array() As Integer
2 Dim meanrt_array () As Single, totalacc_array() As Single
3 Dim meanrtc_array() As Single
4 Dim ntrial_condition() As Integer
5 Dim level_IV1 As Integer, level_IV2 As Integer

```

Table 1 ■ Description of list columns.

Column Name	Description
Weight	Number of times the trial will be presented
Procedure	Name of trial procedure
Stimulus	Presented string on each trial
CorrectResponse	Correct response key on each trial
StimulusType	Level of first independent variable
StringLenght	Level of second independent variable
StimulusNumber	Index number of each stimulus

der of stimulus presentation is random, the value of *StimulusNumber* column of *TrialList* is used to determine what current stimulus is. To get the value of stimulus number, the *GetAttrib* method of *Context* object (*c*) is used. Afterwards, total reaction time, total accuracy and total reaction time of correct response trials are calculated and recorded to *meanrt_array()*, *totalacc_array()* and *meanrtc_array()* respectively. The lines of instructions written in *ArrayRegister* are seen in Listing 3.

While recording total reaction time, total accuracy and total reaction time of correct responses (Lines 22-26), *StimulusType* and *StringLenght* attributes of *TrialList* are used to determine which current experimental condition is. In other words, these attributes are used as indices for matrices *meanrt_array()*, *totalacc_array()* and *meanrtc_array()*. Differently from *meanrt_array()*, and *totalacc_array()*, summation of *meanrtc_array()* is controlled by an *If* statement (Lines 24-26) as it is calculated only for correct responses.

In the object *DataLogging*, data file is created then accuracy and reaction time data are written into the data file (line 24-39). The lines of instructions written in *DataLogging* are shown in Listing 4 and line by line explanation of the code is discussed subsequently.

The local variable *i* and *j* are (Line 27) used as loop counters in *For . . . Next* statements. In Line 28, by means of *FileExists* function, *If* statement checks whether or not the data file has been created before. If the file has not been created (i.e., the current subject is the first subject), the data file is opened as a text file in append mode (Line 29) in the experiment's folder and all data column labels (variable names to be used by the statistical package) are written in the first line of the file (Line 30-34). Column labels are delimited by tab characters using the *ebTab* constant. The labels and their descriptions are given in Table 3.

If the data file has been created before, that is, *FileExists* is true, Line 36 runs and file is opened to add the current subject to previously saved data.

Table 2 ■ Description of public variables.

Variable	Description	Data Type
rt_array()	Reaction time for each stimulus (in miliseconds)	Single
accur_array()	Accuracy for each stimulus	Integer
meanrt_array()	Mean reaction time for each of six conditions	Single
totalacc_array()	Total accuracy for each of six conditions	Integer
meanrtc_array()	Mean reaction time of correct responses for each of six conditions	Single
ntrial_condition()	Number of trials for each of six conditions	Integer
level_IV1	Number of levels for the first independent variable	Integer
level_IV2	Number of levels for the second independent variable	Integer

**Listing 2** ■ The instructions contained in the object *ArrayRedim*.

```
6 ReDim rt_array(1 To TrialList.Size)
7 ReDim accur_array(1 To TrialList.Size)
8 level_IV1 = 3
9 level_IV2 = 2
10 ReDim meanrt_array (1 To level_IV1, 1 To level_IV2)
11 ReDim totalacc_array (1 To level_IV1, 1 To level_IV2)
12 ReDim meanrtc_array (1 To level_IV1, 1 To level_IV2)
13 ReDim ntrial_condition (1 To level_IV1, 1 To level_IV2)
14 ntrial_condition(1,1)=6
15 ntrial_condition(1,2)=6
16 ntrial_condition(2,1)=3
17 ntrial_condition(2,2)=3
18 ntrial_condition(3,1)=3
19 ntrial_condition(3,2)=3
```

Listing 3 ■ The instructions contained in the object *ArrayRegister*.

```
20 rt_array (val(c.GetAttrib("StimulusNumber"))) = Stimulus.RT
21 accur_array (val(c.GetAttrib("StimulusNumber"))) = Stimulus.ACC
22 meanrt_array(val(c.GetAttrib("StimulusType")), val(c.GetAttrib("StringLenght")))=
    meanrt_array(val(c.GetAttrib("StimulusType")), val(c.GetAttrib("StringLenght")))
    + Stimulus.RT
23 totalacc_array(val(c.GetAttrib("StimulusType")), val(c.GetAttrib("StringLenght")))
    = totalacc_array(val(c.GetAttrib("StimulusType")), val(c.GetAttrib("StringLenght
    "))) + Stimulus.ACC
24 If Stimulus.ACC = 1 Then
25     meanrtc_array(val(c.GetAttrib("StimulusType")), val(c.GetAttrib("StringLenght")))
        = meanrtc_array(val(c.GetAttrib("StimulusType")), val(c.GetAttrib("
        StringLenght"))) + Stimulus.RT
26 End If
```

Printing subject data into the file starts from Line 40. Same as column labels, each variable is also delimited by a tab character using *ebTab* constant. First, subject number is printed using *GetAttrib* method of *Context* object. Current date and time is added via *Date* and *Time* functions. The *For ... Next* statement (Line 39-41) prints reaction time and accuracy data of each stimulus using *rt_array()* and *accur_array()* values, then summaries of accuracy and reaction time data are printed in Lines 42-52. Nested *For ... Next* statements print summary variables because all of them are 3 (word, pseudoword, nonword) x 2 (short, long) matrices. Mean reaction time for each experimental condition is calculated by using *meanrt_array(i, j)/ntrial_condition(i, j)* statement. Similarly, *totalacc_array(i, j)* prints total accuracy, whereas *meanrtc_array(i, j)/*

totalacc_array(i, j) prints mean reaction time of correct response trials for each condition. If there is not a single correct response for a particular experimental condition, -1 is printed as mean reaction time of correct responses (Line 49). Finally, file is closed in Line 39.

After running the lexical decision task for several subjects, the data file looks like the one shown in Figure 3.

Because column labels and data are tab delimited, the text file can be easily imported into any statistical package without doing any additional modification to the file. For example, in SPSS, the *Read Text Data* procedure (under *File* menu) can be used to import the text file.⁴

Conclusion

There are numerous publications that provide methodological explanation regarding techniques such as data for-

⁴E-Basic uses the dot mark as a decimal separator. If the computer's decimal separator is not a dot mark, SPSS imports reaction time data as string values. To avoid this problem, change the computer's decimal symbol to dot mark using regional settings of Windows, before importing the text file.

**Listing 4** ■ The instructions contained in the object *DataLogging*.

```
27 Dim i As Integer, j As Integer
28 If Not FileExists ("data.txt") Then
29     Open "data.txt" For Append As #1
30     Print #1, "subject" & ebTab & "date" & ebTab & "time" & ebTab;
31     For i = 1 To TrialList.Size
32         print #1, "stim" & i & "rt" & ebTab & "stim" & i & "acc" & ebTab;
33     Next i
34     Print #1, "mean_rtws" & ebTab & "accur_ws" & ebTab & "mean_rtwsc" & ebTab & "
mean_rtwl" & ebTab & "accur_wl" & ebTab & "mean_rtwlc" & ebTab & "mean_rtpws" &
ebTab & "accur_pws" & ebTab & "mean_rtpwsc" & ebTab & "mean_rtpwl" & ebTab & "
accur_pwl" & ebTab & "mean_rtpwlc" & ebTab & "mean_rtnws" & ebTab & "accur_nws"
& ebTab & "mean_rtnwsc" & ebTab & "mean_rtnwl" & ebTab & "accur_nwl" & ebTab & "
mean_rtnwlc"
35 Else
36     Open "data.txt" For Append As #1
37 End If
38 Print #1, c.GetAttrib ("Subject") & ebTab & Date & ebTab & Time & ebTab;
39 For i = 1 To TrialList.Size
40     Print #1, rt_array(i) & ebTab & accur_array(i) & ebTab;
41 Next i
42 For i = 1 To level_IV1
43     For j = 1 To level_IV2
44         Print #1, meanrt_array (i, j)/ntrial_condition(i,j) & ebTab &
totalacc_array (i, j) & ebTab;
45         If Not totalacc_array (i, j) = 0 Then
46             Print #1, meanrtc_array (i,j)/totalacc_array(i,j) & ebTab;
47         Else
48             print #1, -1 & ebTab;
49         End If
50     Next j
51 Next i
52 Next i
53 Print #1,
54 Close #1
```

matting (Lacroix & Giguère, 2006), E-Studio usage (Richard & Charbonneau, 2009), E-Basic programming (Cousineau, 2009), image manipulation (Ball, Elzemann, & Busch, 2014) and measurement of timing accuracy (De Clercq, Crombez, Buysse, & Roeyers, 2003; Smyth, Cardy, & Purcell, 2017). The contribution of the present paper to this literature is that it provides E-Basic instructions to save time when ready to analyze data. The method shown here is simple and applicable to any experiment that measures accuracy and/or reaction time as a dependent variable. By making minor modifications to the sample code, it is possible to create customized data files in a wide range of stimulus presentation scenarios.

Authors' note

The author thanks Editor Denis Cousineau for his helpful suggestions and Elvan Arıkan İyilikci for her comments on earlier versions of the manuscript. Correspondence concerning this article should be addressed to Osman İyilikci, Department of Psychology, Cyprus International University, Lefkoşa, via Mersin 10, TURKEY.

References

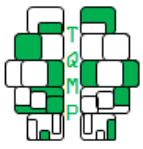
- Ball, F., Elzemann, A., & Busch, N. A. (2014). The scene and the unseen: manipulating photographs for experiments on change blindness and scene memory. *Behavior Research Methods*, 46, 689–701. doi:10.3758/s13428-013-0414-2



- Cousineau, D. (2009). Using mathematica within e-prime. *Tutorials in Quantitative Methods for Psychology*, 5, 59–67. doi:[10.20982/tqmp.05.2.p059](https://doi.org/10.20982/tqmp.05.2.p059)
- De Clercq, A., Crombez, G., Buysse, A., & Roeyers, H. (2003). A simple and sensitive method to measure timing accuracy. *Behavior Research Methods, Instruments, & Computers*, 35, 109–115. doi:[10.3758/BF03195502](https://doi.org/10.3758/BF03195502)
- Haxby, J. V., Parasuraman, R., Lalonde, F., & Abboud, H. (1993). Superlab: general-purpose macintosh software for human experimental psychology and psychological testing. *Behavior Research Methods, Instruments, & Computers*, 25, 400–405. doi:[10.3758/BF03204531](https://doi.org/10.3758/BF03204531)
- Lacroix, G. L. & Giguère, G. (2006). Formatting data files for repeated-measures analyses in spss: using the aggregate and restructure procedures. *Tutorials in Quantitative Methods for Psychology*, 2, 20–25. doi:[10.20982/tqmp.02.1.p020](https://doi.org/10.20982/tqmp.02.1.p020)
- Mathôt, S., Schreij, D., & Theeuwes, J. (2012). Opensesame: an open-source, graphical experiment builder for the social sciences. *Behavior Research Methods*, 44, 314–324. doi:[10.3758/s13428-011-0168-7](https://doi.org/10.3758/s13428-011-0168-7)
- Meyer, D. E. & Schvaneveldt, R. W. (1971). Facilitation in recognizing pairs of words: evidence of a dependence between retrieval operations. *Journal of Experimental Psychology*, 90, 227–234. doi:[10.1037/h0031564](https://doi.org/10.1037/h0031564)
- Psychology Software Tools, I. (2017). *E-prime: documentation article*. E-Prime: PSTNet. Retrieved from <https://support.pstnet.com>
- Richard, L. & Charbonneau, D. (2009). An introduction to e-prime. *Tutorials in Quantitative Methods for Psychology*, 5, 68–76. doi:[10.20982/tqmp.05.2.p068](https://doi.org/10.20982/tqmp.05.2.p068)
- Smyth, R. E., Cardy, J. O., & Purcell, D. (2017). Testing the accuracy of timing reports in visual timing tasks with a consumer-grade digital camera. *Behavior Research Methods*, 49, 967–971. doi:[10.3758/s13428-016-0757-6](https://doi.org/10.3758/s13428-016-0757-6)

Figure 3 ■ Data file.

subject	date	time	stim1rt	stim1acc	stim2rt	stim2acc	stim3rt	stim3acc	stim4rt	
1	28.01.2018	03:24:06	624	1	446	1	445	1	1029	1
2	28.01.2018	03:25:13	523	1	490	1	657	1	2042	1
3	28.01.2018	03:26:27	634	1	544	1	515	1	603	1

**Table 3** ■ Description of column labels.

Variable	Description
subject	Subject number
date	Current date
time	Current time
stim1-24rt	Reaction time of each stimulus
stim1-24acc	Accuracy of each stimulus
mean_rtws	Mean reaction time for short words
accur_ws	Total accuracy for short words
mean_rtwsc	Mean reaction time for correct responses of short words
mean_rtwl	Mean reaction time for long words
accur_wl	Total accuracy for long words
mean_rtwlc	Mean reaction time for correct responses of long words
mean_rtpws	Mean reaction time for short pseudowords
accur_pws	Total accuracy for short pseudowords
mean_rtpwsc	Mean reaction time for correct responses of short pseudowords
mean_rtpwl	Mean reaction time for long pseudowords
accur_pwl	Total accuracy for long pseudowords
mean_rtpwlc	Mean reaction time for correct responses of long pseudowords
mean_rtnws	Mean reaction time for short nonwords
accur_nws	Total accuracy for short nonwords
mean_rtnwsc	Mean reaction time for correct responses of short nonwords
mean_rtnwl	Mean reaction time for long nonwords
accur_nwl"	Total accuracy for long nonwords
mean_rtnwlc	Mean reaction time for correct responses of long nonwords

Open practices

📄 The *Open Material* badge was earned because supplementary material(s) are available on the [journal's web site](#).

Citation

İyilikci, O. (2018). Creating customized data files in E-Prime: A practical tutorial. *The Quantitative Methods for Psychology*, 14(1), 73–80. doi:10.20982/tqmp.14.1.p073

Copyright © 2018, İyilikci. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) or licensor are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

Received: 01/02/2018 ~ Accepted: 16/01/2018