

Instructions for Running E-Z Reader 10.4 Simulations

© Erik D. Reichle (23 January 2025)

1. Introduction

These are the instructions for running simulations using the *E-Z Reader* model of eye-movement control in reading. If you haven't already done so, please read Reichle (2011), Reichle et al. (2012), or Veldre et al. (2023) (see References, below) so that you have a good understanding of how the model works prior to running simulations. I also encourage you to have some background using the Java programming language because it will also be necessary to run simulations.

The .zipped file that you have downloaded, *EZReader 10.4.zip*, contains these instructions and another folder named *EZReader 10.4*. The latter contains the project and source (.java) code for the E-Z Reader model. The project has been compiled and tested using the Apache Netbeans IDE (ver. 17) running Java 17.0.9. In my experience, Java is robust, and you shouldn't have difficulty opening and running the model using whatever version of Java you have installed on your computer. However, if you don't already have Java and/or an IDE, you can easily download them for free using the following links:

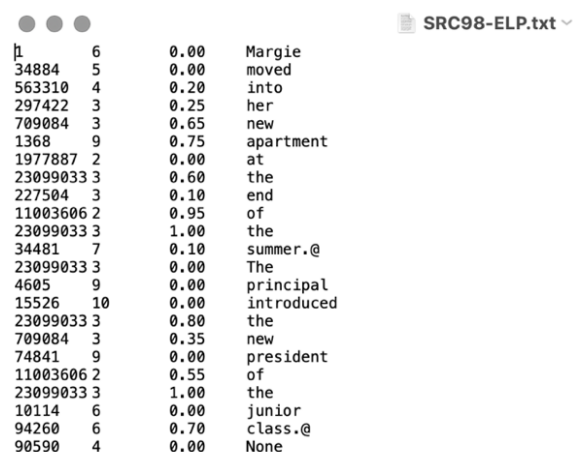
Java: <https://www.oracle.com/index.html>

Apache Netbeans: <https://netbeans.apache.org>

2. Preliminary Steps

To run the program, you will need two .txt files: (1) a corpus file containing information about the sentences that will be used in the simulation; and (2) a target file specifying the ordinal position of a specific target word in each of the sentences.

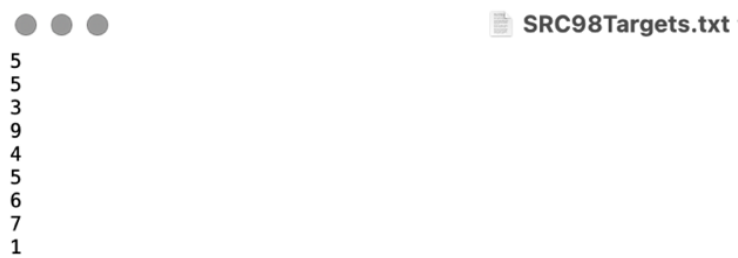
In the *EZReader 10.4* folder, the corpus file is named *SRC98-ELP.txt*. This file contains information about the 48 sentences used by Schilling et al. (1998). If you open the file, the top portion will look like Figure 1. Information about the sentences is arranged into four columns indicating each word's: (1) frequency of occurrence (e.g., as tabulated in the English Lexicon Project; Balota et al., 2007); (2) length (i.e., number of letters); (3) cloze predictability (e.g., see Taylor, 1953); and (4) the actual word. Note that the last word in each sentence is indicated by an ampersand symbol (i.e., @). To run simulations using another sentence corpus, the file must be formatted exactly like the example in Figure 1. (Please be careful that the files don't include control characters because this can be problematic for the program.)



hl	6	0.00	Margie
34884	5	0.00	moved
563310	4	0.20	into
297422	3	0.25	her
709084	3	0.65	new
1368	9	0.75	apartment
1977887	2	0.00	at
23099033	3	0.60	the
227504	3	0.10	end
11003606	2	0.95	of
23099033	3	1.00	the
34481	7	0.10	summer.@
23099033	3	0.00	The
4605	9	0.00	principal
15526	10	0.00	introduced
23099033	3	0.80	the
709084	3	0.35	new
74841	9	0.00	president
11003606	2	0.55	of
23099033	3	1.00	the
10114	6	0.00	junior
94260	6	0.70	class.@
90590	4	0.00	None

Figure 1. Example corpus file.

There is also a target file in *EZReader 10.4* folder named *SRC98Targets.txt*. This file contains a single column of numbers indicating the within-sentence ordinal positions of specific target words in the sentences. In the Schilling et al. (1998) sentences, these were the high- and low-frequency target words that were of interest in the experiment. If you open the target file, the top part will look like Figure 2. There, the first number, 5, refers to the word “apartment” in the first sentence of Figure 1. Note that “apartment” is actually the sixth word in the sentence, not the fifth. As per standard Java conventions, an array of N items is indexed using the numbers 0 to $N-1$. For example, a target word that is the seventh word of a 12-word sentence would be word 6 with the indices for the first and last words being 0 and 11, respectively. Finally, if using another sentence corpus, it is not necessary to have designated target words, but it is necessary to have a target file. Such a file can be dummy coded using a single column of $N-1$ zeros, with N equaling the number of sentences in the corpus.



```
5
5
3
9
4
5
6
7
1
```

Figure 2. Example target file.

3. Running Simulations

To run E-Z Reader simulations, first open your IDE and then open the project, *EZReader 10.4*. Then open the class containing the main method, *EZReader10.java*. The top part of this class should look like Figure 3.

```

// I/O file names:
static String corpusFileName = "SRC98-ELP"; // file containing the sentence corpus
static String outputFileName = "SimulationResults"; // simulation results will be written to this file
static String targetFileName = "SRC98Targets"; // file containing a list of within-sentence locations of target words

// Model free parameters:
static double A = 25.0; // attention shift latency
static double Alpha1 = 124.0; // lexical processing rate intercept
static double Alpha2 = 11.1; // lexical processing rate word-frequency slope
static double Alpha3 = 76.0; // lexical processing rate word-predictability slope
static double Delta = 1.68; // L2 vs. L1 proportional difference
static double Epsilon1 = 0.1; // degradation due to acuity
static double Epsilon2 = 0.5; // masking from medial letters
static double Epsilon3 = 1.0; // masking from lateral letters
static double Eta1 = 0.5; // saccade random error intercept
static double Eta2 = 0.1; // saccade random error slope
static double I = 50.0; // post-lexical integration latency
static double ITarget = 50.0; // post-lexical integration latency (of target words)
static double Lambda = 0.25; // automatic refixation probability slope
static double M1 = 150.0; //labile saccadic programming latency
static double M2 = 25.0; // non-labile saccadic programming latency
static double Omega1 = 6.0; // systematic range error intercept
static double Omega2 = 3.0; // systematic range error slope
static double pF = 0.01; // probability of integration failure
static double pFTarget = 0.01; // probability of integration failure (for target words)
static double Psi = 7.0; // preferred saccade length
static double S = 25.0; // saccade latency
static double SigmaGamma = 20.0; // controls standard deviation gamma distribution
static double V = 60.0; // pre-attentive visual processig (i.e., eye-mind latency)
static double Xi = 0.5; // proportion of labile programming related to target selection

// Fixed simulation parameters:
static int maxLength = 16; // i.e., number letters + black space to the left of the word
static int maxSentenceLength = 50; // maximum number of words per sentence
static double minValue = 0.00001; // value to prevent using zero values
static int NFrequencyClasses = 8; // number of frequency classes
static int NSentences = 48; // number of sentences in corpus
static int NSubjects = 1000; // number of simulated readers

// Simulation toggles:
static boolean includeRegressionTrials = false; // include trials with inter-word regressions?
static boolean includeTargetWords = true; // include specific targets words?

// Display toggles:
static boolean displayCorpus = false; // IVs for each word in corpus
static boolean displayFrequencyClassMeans = false; // frequency-class means
static boolean displayMeanDistributions = true; // display mean distributions for each word length
static boolean displayWordStatistics = true; // mean DVs for each word
static boolean displayParameters = true; // model parameter values
static boolean displayPrRegression = false; // proportion of trials having inter-word regressions
static boolean displayRMSD = true; // goodness-of-fit metric
static boolean displayStates = false; // E-Z Reader's internal states
static boolean displayTraces = false; // traces showing individual fixations

// Initialize Schilling et al. (1998) observed means and standard deviations [SD = p (1 - p); see the Appendix of Reich
// using frequency norms from the English Lexicon Project (Balota et al., 2007):
static double[] obsMeanFFD = new double[] { 274, 237, 243, 246, 238, 242, 224, 205 }; // first-fixation durations
static double[] obsSDFFD = new double[] { 42, 20, 20, 37, 36, 37, 74, 84 };
static double[] obsMeanGD = new double[] { 336, 284, 304, 301, 266, 260, 235, 209 }; // gaze durations
static double[] obsSDGD = new double[] { 60, 26, 53, 65, 45, 52, 82, 86 };
static double[] obsMeanSFD = new double[] { 281, 257, 253, 260, 246, 246, 228, 204 }; // single-fixation durations
static double[] obsSDSFD = new double[] { 52, 21, 26, 45, 37, 39, 82, 84 };
static double[] obsMeanPr1 = new double[] { 0.725, 0.655, 0.695, 0.669, 0.714, 0.575, 0.373, 0.311 }; // probability of
static double[] obsSDPr1 = new double[] { 0.446, 0.475, 0.460, 0.471, 0.452, 0.494, 0.484, 0.463 };
static double[] obsMeanPr2 = new double[] { 0.203, 0.177, 0.254, 0.223, 0.128, 0.053, 0.025, 0.008 }; // probability of
static double[] obsSDPr2 = new double[] { 0.402, 0.382, 0.435, 0.416, 0.334, 0.224, 0.157, 0.088 };
static double[] obsMeanPrS = new double[] { 0.071, 0.167, 0.051, 0.108, 0.158, 0.372, 0.601, 0.681 }; // probability of
static double[] obsSDPrS = new double[] { 0.258, 0.373, 0.220, 0.311, 0.364, 0.483, 0.490, 0.466 };

```

Figure 3. The EZReader10.java class.

Starting at the top of Figure 3, *EZReader10.java* specifies the names and values of variables corresponding to the input/output (I/O) files that are used in running the simulations, E-Z Reader's free parameters, several (fixed) simulation parameters, and several Boolean toggles that control what's included in the simulation output. More specifically:

(i) *corpusFileName* – This is the name of the sentence corpus file used in the simulation (e.g., *SRC98-ELP.txt*). Change the file name if you want to use a different sentence corpus.

(ii) *outputFileName* – This is the name of the file where the simulation results are written. The file does not have to be provided but will instead be created by the program.

(iii) *targetFileName* – This is the name of the file containing the target words (e.g., *SRC98Targets.txt*). Change the file name if you want to use a different sentence corpus.

(iv) *A, Alpha1, Alpha2, ... Xi* – These are E-Z Reader's free parameters and their default values. These values allow the model to generate eye movements that closely resemble those reported by Schilling et al. (1998).

(v) *maxLength* – This is the maximum word length (i.e., maximum number of letters) in the simulation; it should be set equal to a value that accommodates the longest word in the corpus file.

(vi) *maxSentenceLength* – This is the maximum number of words in any of the sentences in the corpus file.

(vii) *mixValue* – This is a safeguard used to prevent process times from taking on values less than 0 ms.

(viii) *NFrequencyClasses* – This is the number of frequency classes using the ELP word-frequency norms (Balota et al., 2007) and bins defined by $\log_{10}(\text{frequency})$.

(ix) *NSentences* – This is the actual number of sentences in the corpus file.

(x) *NSubjects* – This is the number of statistical subjects used in the simulation.

(xi) *includeRegressionTrials* – By setting this Boolean toggle to "true," simulated trials containing inter-word regressions will be included in the final simulation output. Please note that the default value of this toggle is "false," with only those trials comprised of first-pass fixations being used to generate the simulation results.

(xii) *includeTargetWords* – Setting this Boolean toggle to "true" will tag the target words with asterisks so that they are easier to identify in the output.

(xiii) *displayCorpus, ... displayTraces* – Setting these Boolean toggles to "true" will cause the program to write different simulation results to the output file. More information about these options is provided in the next section of this document.

(xiv) *obsMeanFFD, ... obsSDPrS* – These are the observed means and standard deviations for the various fixation-duration and probability measures on the 8 frequency classes of words used by Schilling et al. (1998). These observed values are used to calculate the model's goodness-of-fit (i.e., RMSD) to the Schilling et al. (1998) corpus. The values should thus be modified accordingly to calculate the goodness-of-fit to any other corpus.

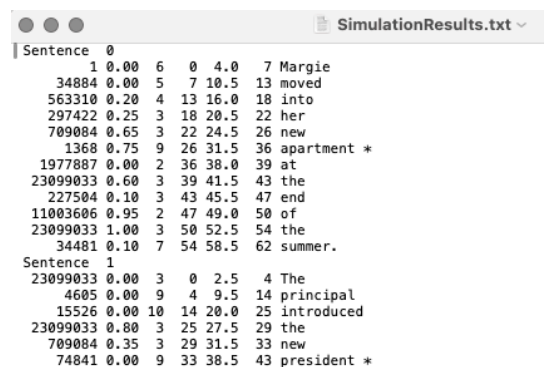
After you have provided the names of the corpus and target files, set the parameters values, and indicated which types of output to store in the output file (by setting the values of the appropriate toggles to "true"), you can start the simulation by clicking on the "run" button in your IDE. (In Apache Netbeans, clicking on the green arrow on the top menu bar will start the program.) The output file should then appear in the program folder, *EZReader 10.4*. Examples and explanations of simulation output are provided below. On a reasonably fast computer, a simulation using 1,000

statistical subjects and the Schilling et al. (1998) sentences should require only a few seconds to complete.

4. Simulation Output

As mentioned in the previous section of this document, the display toggles determine the type of simulation results that will be written to the output file. The possible types of output are as follows:

(i) *displayCorpus* – Setting this toggle to “true” will write out the independent variables that are generated for each of word in the sentence corpus. An example of this output is shown in Figure 4. The output is organized by sentences (from the top of the file, starting with Sentence 0) and by words (each row corresponds to a word). The columns code the following information about each word’s: (1) frequency; (2) cloze predictability; (3) length; (4) cumulative character position of the left edge of the blank space preceding the word; (5) center of word, or *optimal viewing position* (OVP; see Vitu et al., 2001); (6) right edge of the word’s final letter; and (7) the actual word. For example, the word “Margie” is low frequency (= 1), completely unpredictable (= 0.00), and 6 letters in length. Because the word is the first in the sentence, the beginning of the blank space to the left of the word is located at position 0 and the rightmost edge of the final letter is located at position 7. The OVP for “Margie” is located between the letters “r” and “g,” at position 4. Finally, words that are designated targets are marked by asterisk symbols (i.e., *) to make them easier to identify (e.g., for the purpose of analyses). It is useful to check the corpus information prior to running simulations to ensure that the program is reading the corpus file correctly. (Note that punctuation is included in the calculation of word length; e.g., the length of “summer” is 7 because the period ending the sentence is included with the word.)



```

SimulationResults.txt
Sentence 0
1 0.00 6 0 4.0 7 Margie
34884 0.00 5 7 10.5 13 moved
563310 0.20 4 13 16.0 18 into
297422 0.25 3 18 20.5 22 her
709084 0.65 3 22 24.5 26 new
1368 0.75 9 26 31.5 36 apartment *
1977887 0.00 2 36 38.0 39 at
23099033 0.60 3 39 41.5 43 the
227504 0.10 3 43 45.5 47 end
11003606 0.95 2 47 49.0 50 of
23099033 1.00 3 50 52.5 54 the
34481 0.10 7 54 58.5 62 summer.
Sentence 1
23099033 0.00 3 0 2.5 4 The
4605 0.00 9 4 9.5 14 principal
15526 0.00 10 14 20.0 25 introduced
23099033 0.80 3 25 27.5 29 the
709084 0.35 3 29 31.5 33 new
74841 0.00 9 33 38.5 43 president *

```

Figure 4. Example output using *displayCorpus*.

(ii) *displayFrequencyClassMeans* – Setting this toggle to “true” will write out the mean observed and predicted values (across 8 word-frequency classes, f) of 6 eye-movement measures: (1) first-fixation duration (FFD); (2) single-fixation durations (SFD); (3) gaze duration (GD); (4) probably of making a single fixation ($Pr1$); (5) probability of making two or more fixations ($Pr2$); and (6) probably of skipping (PrS). An example of such output is shown in Figure 5. Within each pair of values, the observed value is displayed first and the simulated value is displayed second. For example, across the words in the first frequency class (i.e., $f = 1$), the mean observed and simulated gaze durations are 336 ms and 328 ms, respectively.

f	SFD	FFD	GD	Pr1	Pr2	PrS
1	281 277	274 259	336 328	0.73 0.68	0.20 0.27	0.07 0.05
2	257 253	237 240	284 307	0.66 0.68	0.18 0.26	0.17 0.06
3	253 248	243 236	304 286	0.70 0.74	0.25 0.18	0.05 0.08
4	260 238	246 231	301 265	0.67 0.77	0.22 0.13	0.11 0.10
5	246 238	238 232	266 263	0.71 0.74	0.13 0.11	0.16 0.15
6	246 221	242 219	260 240	0.58 0.67	0.05 0.07	0.37 0.26
7	228 228	224 227	235 234	0.37 0.60	0.03 0.02	0.60 0.38
8	204 212	205 212	209 219	0.31 0.51	0.01 0.02	0.68 0.47

Figure 5. Example output using *displayFrequencyClassMeans*.

(iii) *displayMeanDistributions* – Setting this toggle to “true” will write out three kinds of distributions for words of each length: (1) first-fixation landing-site distributions; (2) refixation-probability distributions; and (3) IOVP distributions. The first distributions correspond to the proportions of initial fixations on each letter within a word. For example, for 1-letter words, 0.44 of the initial fixations are on the blank space preceding the word, while the remaining 0.56 land on the word itself. The second distributions show the proportion of initial fixations that were followed by a refixation as a function of the initial fixation’s location. For example, for the 5-letter words, initial fixations at the first 3 locations were followed by refixations on 0.42, 0.20, and 0.07 of the simulated trials, respectively. Finally, the last distributions show the mean single-fixation durations as a function of their locations. For example, for 2-letter words, the mean single-fixation duration was 220 ms when the fixation was on the first letter of the word. (Note that there are missing values because the sentences only contained one word longer than 12 letters and it was excluded from the analyses because it was a sentence-final word.)

```

Pr1 distributions (i.e., first-fixation landing sites):
1-letter: 0.44 0.56
2-letter: 0.24 0.34 0.43
3-letter: 0.20 0.20 0.30 0.30
4-letter: 0.10 0.13 0.25 0.31 0.20
5-letter: 0.10 0.10 0.17 0.28 0.24 0.12
6-letter: 0.09 0.07 0.12 0.22 0.26 0.16 0.06
7-letter: 0.11 0.07 0.10 0.18 0.23 0.18 0.09 0.05
8-letter: 0.08 0.05 0.07 0.15 0.22 0.22 0.13 0.05 0.03
9-letter: 0.07 0.04 0.05 0.10 0.20 0.24 0.18 0.08 0.03 0.02
10-letter: 0.11 0.06 0.05 0.08 0.15 0.20 0.17 0.10 0.04 0.03 0.02
11-letter: 0.07 0.04 0.03 0.07 0.12 0.20 0.19 0.13 0.07 0.04 0.02 0.02
12-letter: 0.08 0.02 0.01 0.04 0.10 0.17 0.23 0.17 0.09 0.04 0.02 0.01 0.01
13-letter: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
14-letter: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
15-letter: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

Pr2 distributions (i.e., refixation probabilities):
1-letter: 0.01 0.01
2-letter: 0.05 0.01 0.02
3-letter: 0.14 0.04 0.02 0.05
4-letter: 0.26 0.09 0.02 0.03 0.07
5-letter: 0.42 0.20 0.07 0.03 0.07 0.12
6-letter: 0.54 0.31 0.15 0.05 0.04 0.09 0.13
7-letter: 0.63 0.47 0.25 0.12 0.05 0.08 0.15 0.20
8-letter: 0.65 0.46 0.32 0.15 0.06 0.04 0.08 0.12 0.14
9-letter: 0.59 0.43 0.29 0.19 0.10 0.04 0.07 0.10 0.13 0.13
10-letter: 0.61 0.59 0.49 0.25 0.16 0.06 0.04 0.09 0.10 0.20 0.20
11-letter: 0.78 0.71 0.52 0.42 0.26 0.13 0.03 0.10 0.09 0.15 0.31 0.17
12-letter: 0.88 0.68 0.76 0.43 0.19 0.08 0.03 0.09 0.13 0.10 0.10 0.00
13-letter: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
14-letter: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
15-letter: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

SFD distributions (i.e., inverted optimal viewing position effects):
1-letter: 213 205
2-letter: 220 221 222
3-letter: 217 216 215 212
4-letter: 222 231 239 240 232
5-letter: 236 230 232 238 231 223
6-letter: 227 233 236 243 241 232 218
7-letter: 234 236 240 243 246 237 228 217
8-letter: 225 229 232 235 243 244 232 217 203
9-letter: 253 221 228 238 246 254 243 229 214 218
10-letter: 234 217 227 241 257 269 265 248 229 220 187
11-letter: 235 227 221 247 260 272 277 260 250 229 224 206
12-letter: 230 199 210 217 249 262 272 267 244 233 215 201 208
13-letter: 0 0 0 0 0 0 0 0 0 0 0 0 0
14-letter: 0 0 0 0 0 0 0 0 0 0 0 0 0
15-letter: 0 0 0 0 0 0 0 0 0 0 0 0 0

```

Figure 6. Example output using *displayDists*.

(iv) *displayWordStatistics* – This is arguably the most useful output. Setting this toggle to “true” will write out several means for each word in the sentence corpus, as shown in Figures 7-10. The beginning of the file (Figure 7) shows the mean values of nine word-based measures: (1) single-

fixation duration (*SFD*); (2) first-fixation duration (*FFD*); (3) gaze duration (*GD*); (4) total times (*TT*); (5) go-past times (*GP*); (6) probability of fixating (*PrF*); (7) probability of fixating once (*Pr1*); (8) probability of fixating two or more times (*Pr2*); (9) probability of skipping (*PrS*). The final column shows the word itself, with target words in this example being marked by asterisks (because *includeTargetWords* was set equal to “true”). Note that the first and last words in each sentence are not included in this output (as per convention) because processing starts and ends abruptly on these words. For example, in the first sentence, the words “Margie” and “summer.” are not included in the output.

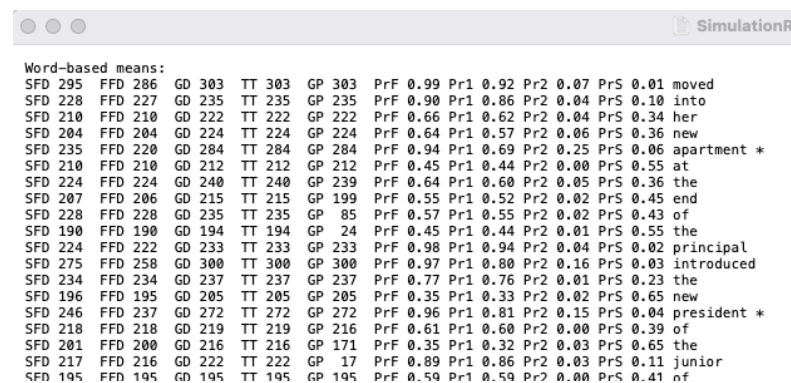


Figure 7. Example output using *displayWordStatistics* showing the word-based means for each word in the sentence corpus.

The next part of the output file (Figure 8) shows the proportions of first-fixations on each possible letter position within a word, with the first “letter” position corresponding to the blank space preceding the word. For example, for the word “moved,” 0.01, 0.06, and 0.26 of the simulated first fixations were located on the first, second, and third letter positions, respectively.

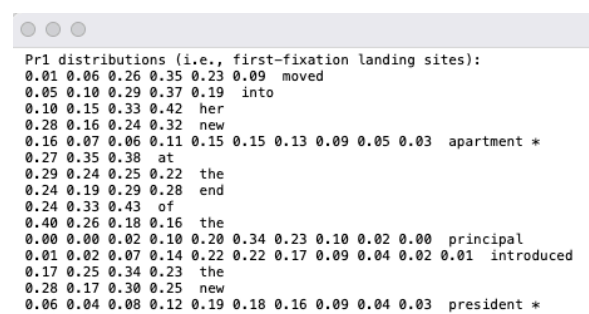


Figure 8. Example output using *displayWordStatistics* showing the first-fixation landing-site distributions for each word in the sentence corpus.

The next part of the output file (Figure 9) shows the proportions of initial fixations at each possible position that were followed by refixations, with the first “letter” position again corresponding to the blank space preceding the word. For example, for the word “new,” initial fixations on the first three positions were respectively followed by refixations on 0.20, 0.06, and 0.04 of the simulated trials.


```

SimulationResults.txt
Parameter values:
A = 25.00
Alpha1 = 124.00
Alpha2 = 11.10
Alpha3 = 76.00
Delta = 1.68
Epsilon1 = 0.10
Epsilon2 = 0.50
Epsilon3 = 1.00
Eta1 = 0.50
Eta2 = 0.10
I = 50.00
ITarget = 50.00
Lambda = 0.25
M1 = 150.00
M2 = 25.00
Omega1 = 6.00
Omega2 = 3.00
pF = 0.01
pFTarget = 0.01
Psi = 7.00
S = 25.00
SigmaGamma = 20.00
V = 60.00
Xi = 0.50

NSubjects = 1000
Include regression trials? false
Include target words? false
-----

Word-based means:
SFD 295 FFD 288 GD 306 TT 306 GP 306 PrF 0.99 Pr1 0.91 Pr2 0.07 PrS 0.01 moved
SFD 233 FFD 231 GD 244 TT 244 GP 244 PrF 0.91 Pr1 0.86 Pr2 0.05 PrS 0.09 into
SFD 217 FFD 217 GD 221 TT 221 GP 221 PrF 0.64 Pr1 0.62 Pr2 0.01 PrS 0.36 her

```

Figure 11. Example output using *displayParameters*. (In this example, the *displayWordStatistics* toggle was also set equal to “true.”)

(vi) *displayPrRegression* – Setting this toggle to true will write out a single value corresponding to the proportion of trials containing at least one inter-word regression. These trials will be excluded from the statistical analyses if *includeRegressionTrials* is set equal to “false.”

(vii) *displayRMSD* – Setting this toggle to “true” will write out a single value corresponding to the model’s performance (i.e., it’s goodness-of-fit or RMSD to the corpus being simulated).

(viii) *displayStates* – Setting this toggle to “true” writes out a chronological record (starting at the top of the output) of the states that the model “moves” through in generating eye movements for each sentence. The columns showing this information from left to right correspond to the: (1) sentence number (*S*); (2) word number being attended (*N*); (3) fixation number (*fix#*); (4) word being fixated (*word*); (5) cumulative position of the fixation (*pos*); (6) cumulative duration of the fixation (*dur*); (7) current rate of lexical processing (*rate*); (8) process that has just completed (in square brackets); (9) any on-going processes and their durations; and (10) the number(s) of any word(s) that resulted in integration failure being indicated after the “IF:”. For example, in the second line of Figure 12, the first sentence is being read (i.e., *S* = 0) with both attention (i.e., *N* = 0) and the initial fixation (i.e., *fix#* = 0) being on the first word (i.e., *word* = 0). This initial fixation is on the middle of the word (i.e., *pos* = 4.0) and its duration now is 170 ms (i.e., *dur* = 170) because only one process has completed so far (i.e., [*L1*]). Other ongoing processes at this time (i.e., upon completion of *L1*) include *L2* (with has a duration of 214 ms) and *M1* (which has a duration of 143 ms). And, in the third line, *M1* completes (because its duration was shorter than that of *L2*) and *M2* is initiated. As shown in the third line, the duration of *L2* is decremented by 143 ms so that its duration is now 71 ms (i.e., 214 – 143 = 71).

In addition to showing the duration of each ongoing process, the following information is provided:

- (1) For *V*, *A*, *L1*, *L2*, and *I*, the word number on which the process is completing is indicated inside the square brackets. For example, *L1* in line 1 is being completed on the first word (i.e., *L1* 170 [0]) while *L2* in line 10 is being completed for the second word (i.e., *L2* 136 [1]).

- Warning:* Because simulations typically generate large number of model states, its best to generate this output using only a small number of simulated subjects (i.e., $NSubjects < 10$).

Figure 12. Example output using *displayStates*.

- Warning:* Because simulations typically generate large number of fixations, its best to generate this output using only a small number of simulated subjects (i.e., *NSubjects* < 10).

Figure 13. Example output using *displayTraces*.

5. Final Notes

I've done my best to make the code as easy to understand and use as possible. However, it's still a complicated program and it's difficult for me to anticipate how the program will be used. So if you run into any problems, please let me know and I'll do my best to help you. Good luck!

6. References

- Balota, D., Yap, M. J., Hutchinson, K. A., Cortese, M. J., Kessler, B., Loftis, B., ... Treisman, R. (2007). The English Lexicon Project. *Behavior Research Methods*, 39, 445-459.
- Reichle, E. D. (2011). Serial attention models of reading. In S. P. Liversedge, I. D. Gilchrist, & S. Everling (Eds.), *Oxford handbook on eye movements* (pp. 767-786). Oxford, UK: Oxford University Press.
- Reichle, E. D., Pollatsek, A., Fisher, D. L., & Rayner, K. (1998). Toward a model of eye-movement control in reading. *Psychological Review*, 105, 125-157.
- Reichle, E. D., Pollatsek, A., & Rayner, K. (2012). Using E-Z Reader to simulate eye movements in non-reading tasks: A unified framework for understanding the eye-mind link. *Psychological Review*, 119, 155-185.
- Schilling, H. E. H., Rayner, K., & Chumbley, J. I. (1998). Comparing naming, lexical decision, and eye fixation times: Word frequency effects and individual differences. *Memory & Cognition*, 26, 1270-1281.
- Taylor, W. L. (1953). Cloze procedure: A new tool for measuring readability. *Journalism Quarterly*, 30, 415-433.
- Veldre, A., Reichle, E. D., Yu, L., & Andrews, S. (2023). Understanding the visual constraints on lexical processing: New empirical and simulation results. *Journal of Experimental Psychology: General*, 152, 693-722.
- Vitu, F., McConkie, G. W., Kerr, P., & O'Regan, J. K. (2001). Fixation location effects on fixation durations during reading: An inverted optimal viewing position effect. *Vision Research*, 41, 3513-3533.

Other useful references for learning about eye movements in reading include:

- Rayner, K. (1998). Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin*, 124, 372-422.
- Rayner, K. (2009). Eye movements and attention in reading, scene perception, and visual search. *The Quarterly Journal of Experimental Psychology*, 62, 1457-1506.
- Rayner, K., Pollatsek, A., Ashby, J., & Clifton, C., Jr. (2012). *The psychology of reading*, 2nd Ed. New York, NY, USA: Psychology Press.
- Reichle, E. D. (2021). *Computational models of reading: A handbook*. Oxford, UK: Oxford University Press.