Supporting Information for: Bestgen Y. Measuring Lexical Diversity in Texts: The Twofold Length Problem. Article accepted in Language Learning.

Appendix S1: Profile Graphs for the Four Methods and the Two Data Collections, Except for Those Shown in Figure 3

Figure S1.1

Profile Graphs for ICNALE in the Parallel Sampling Method

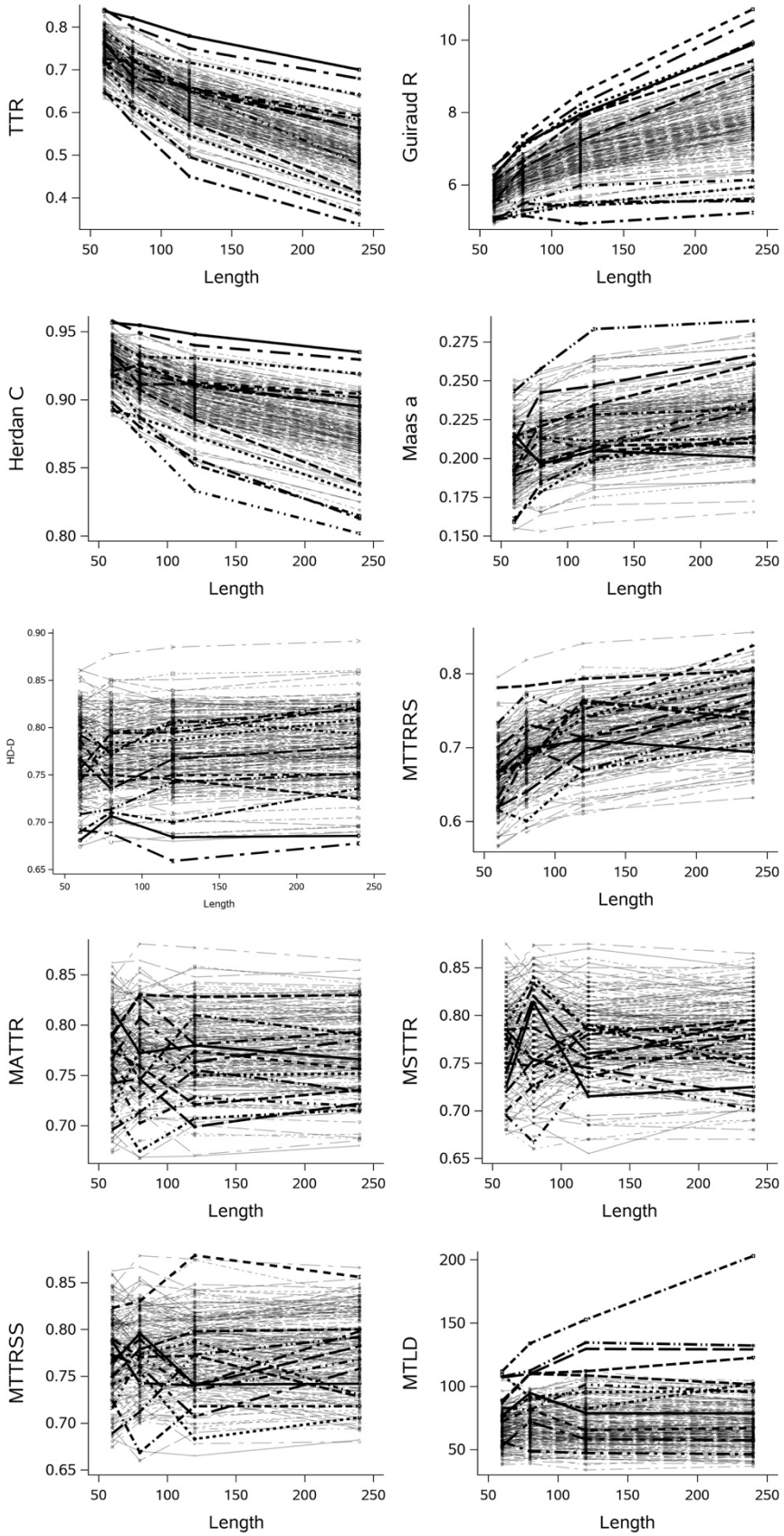


Figure S1.2

Profile Graphs for ICNALE in the Random Sampling Method

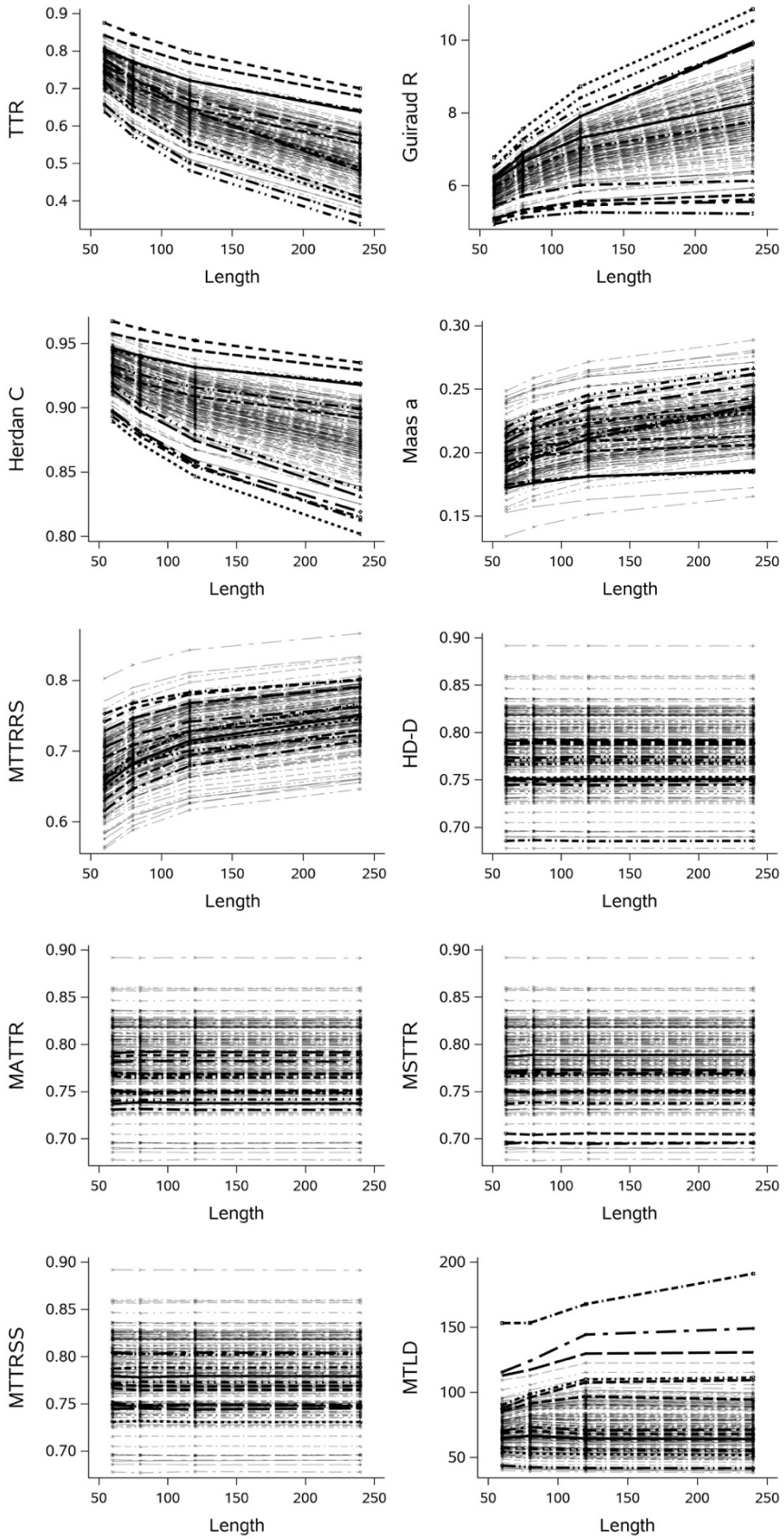


Figure S1.3

Profile Graphs for ICNALE in the Ordered Random Sampling Method

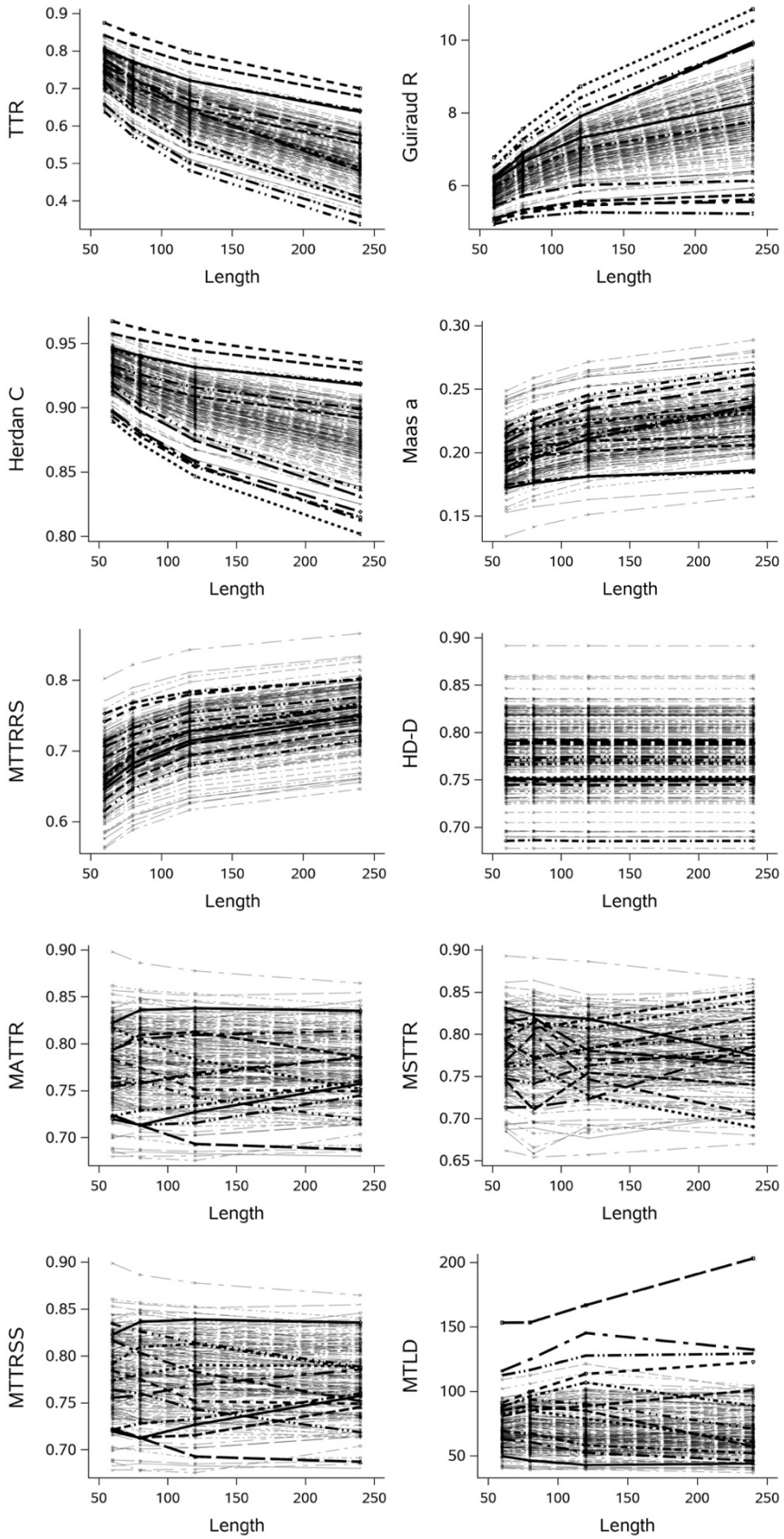


Figure S1.4

Profile Graphs for ICLE in the Parallel Sampling Method

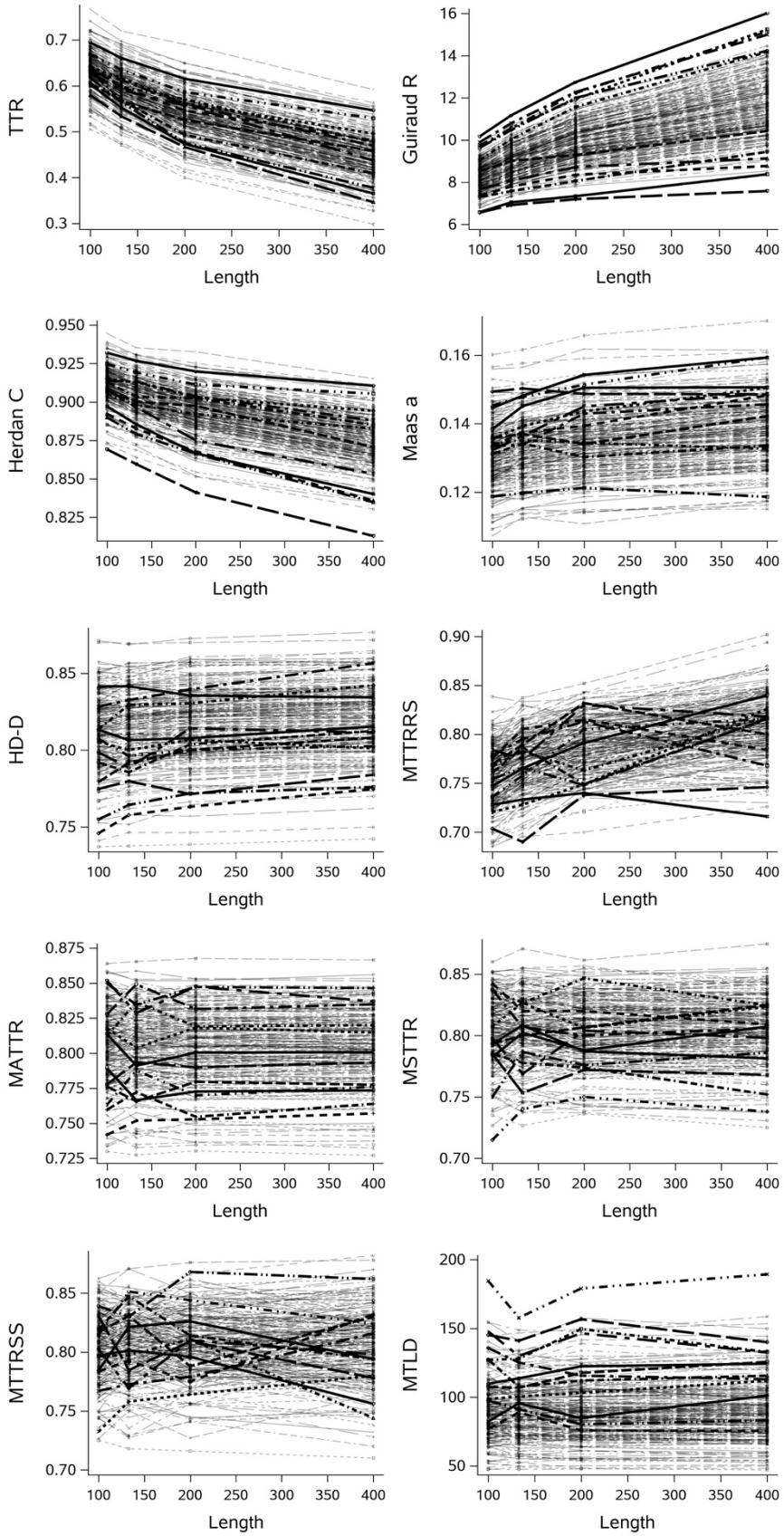


Figure S1.5

Profile Graphs for ICLE in the Random Sampling Method

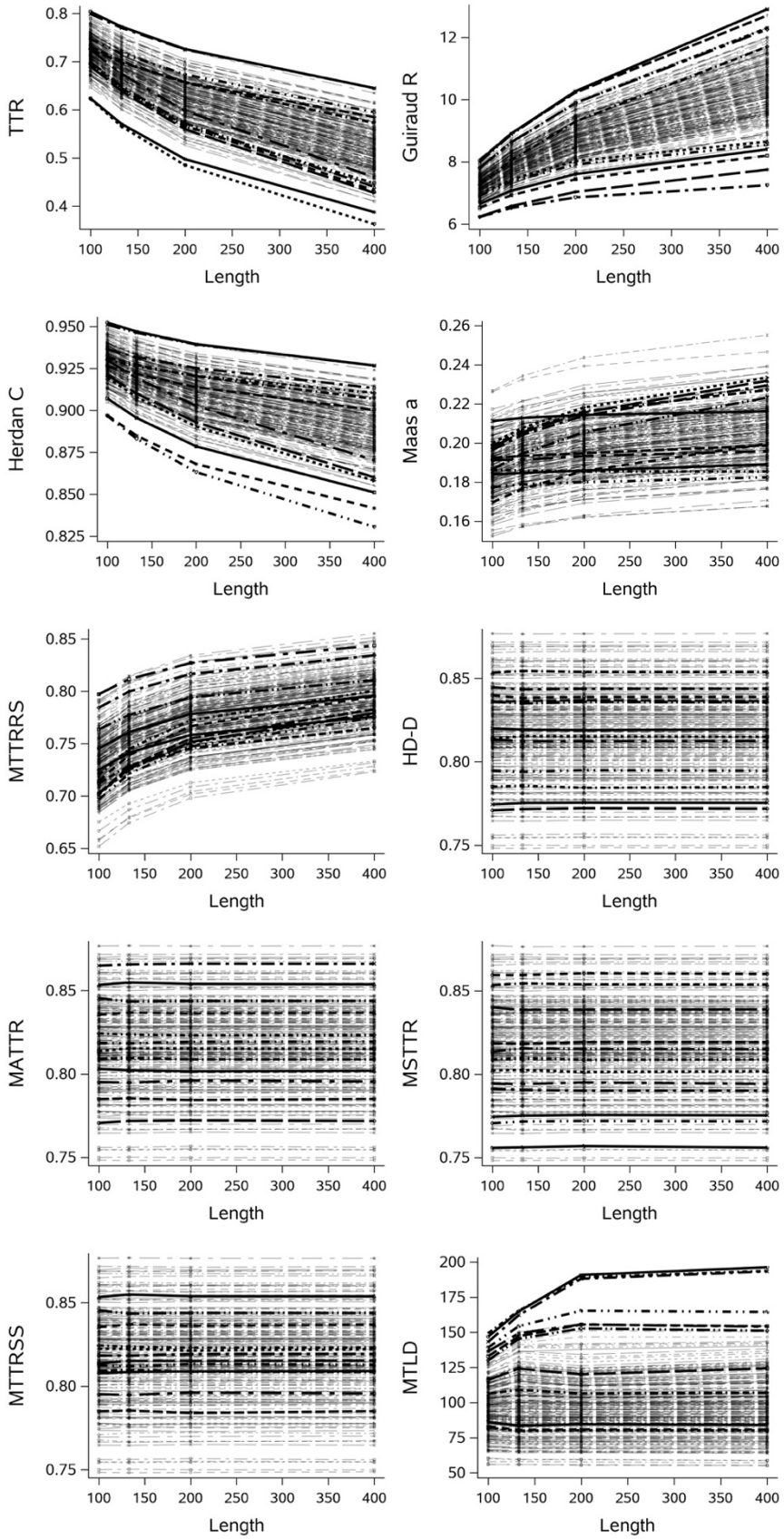


Figure S1.6

Profile Graphs for ICLE in the Ordered Random Sampling Method

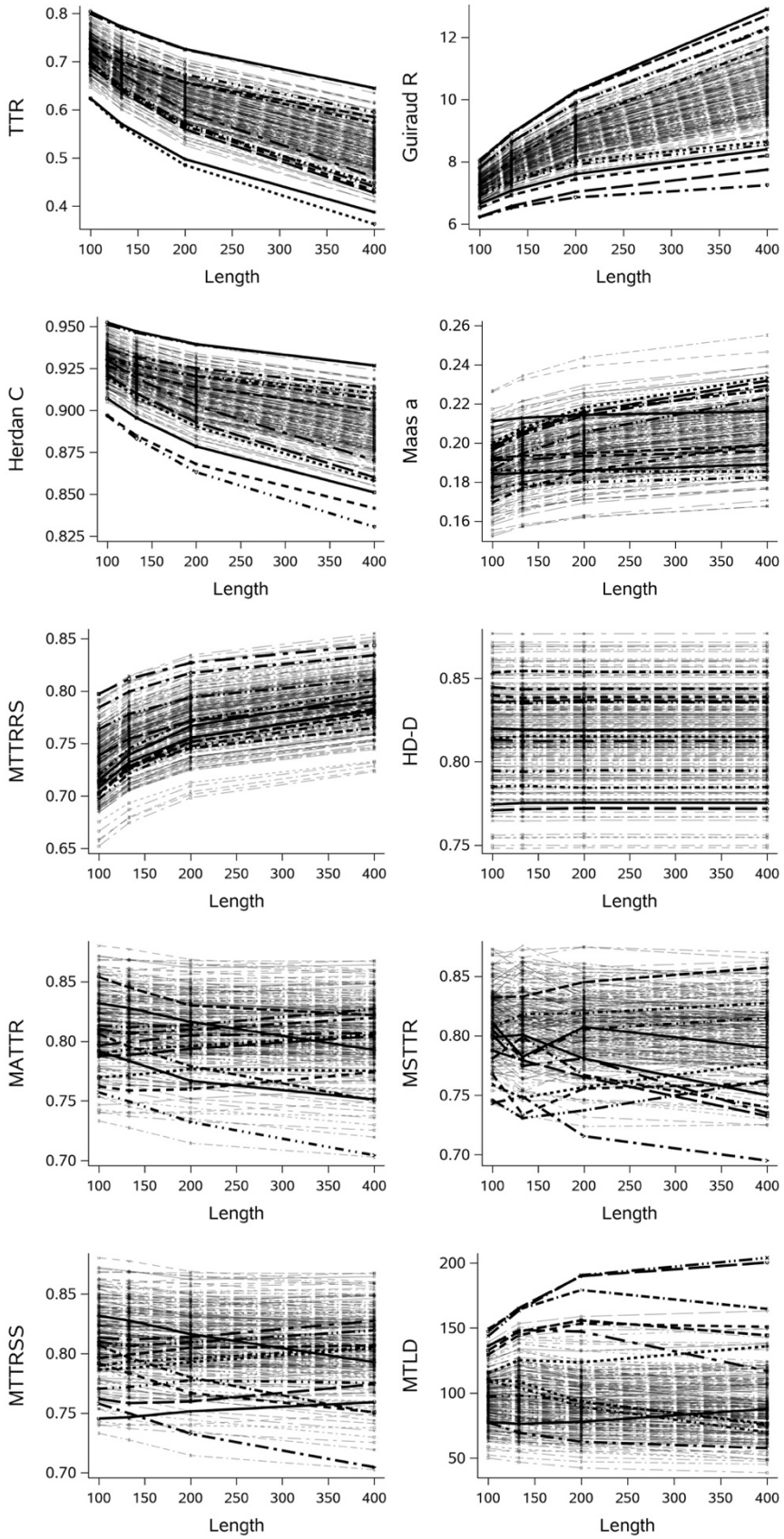
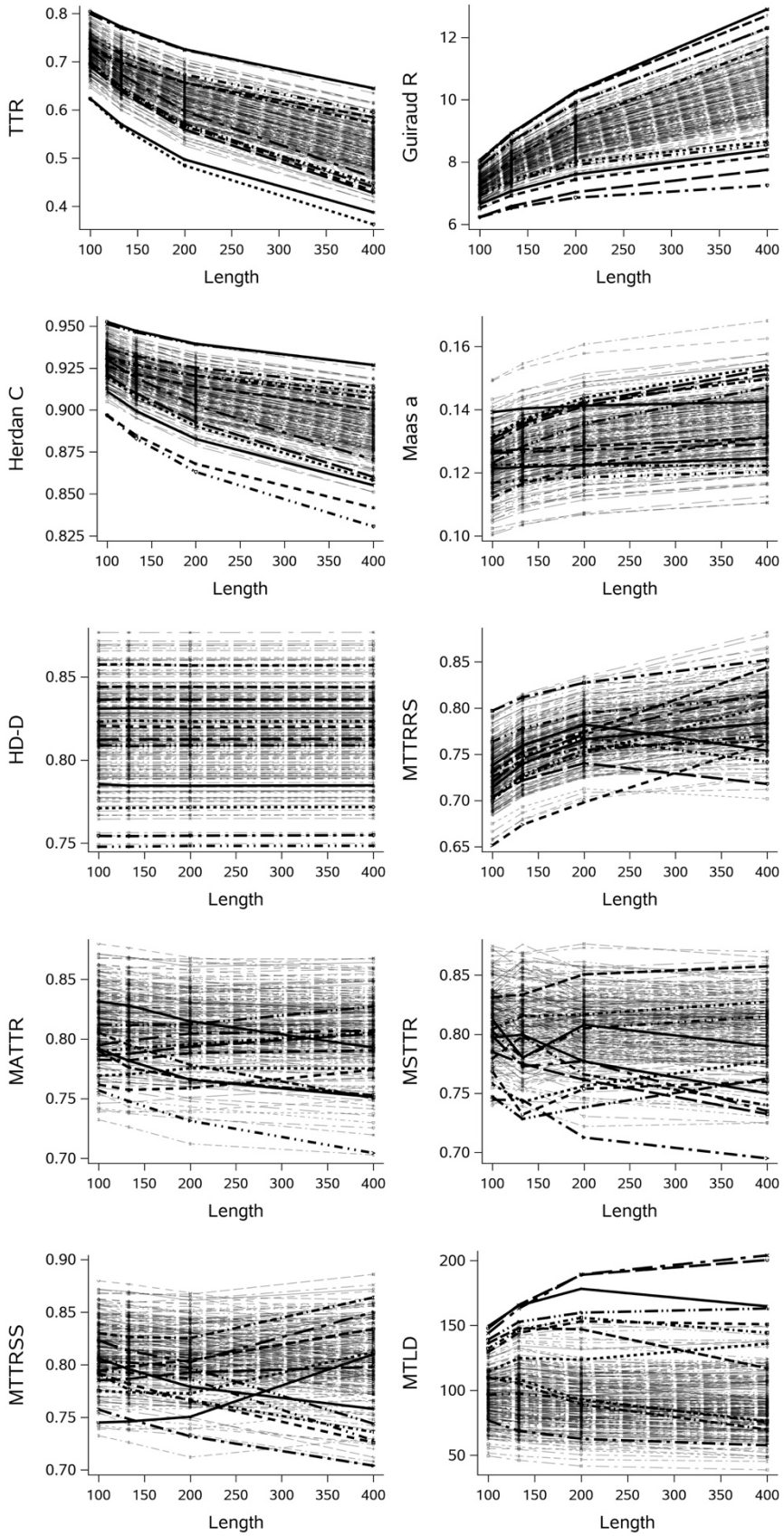


Figure S1.7

Profile Graphs for ICLE in the Alternating Token Sampling Method



Appendix S2: Weight of Each Token in an LD Index

The weight represents the impact that a given token has on an LD score. For example, in the TTR, each token is counted once and only once in the calculation. All tokens have thus the same weight, which is equal to $1/N$, in which N is the number of tokens.

For MSTTR, we have the same result only if N is a multiple of n, the length of a segment. If it is not the case, a certain number of tokens are not taken into account. If we represent by $mod(x,y)$ the modulo operation, which returns the remainder of a division of x by y, the number of tokens not taken into account, is equal to $Mod(N,n)*n$. These remaining tokens received a weight of 0 while the others received a weight of $1/(N-Mod(N,n)*n)$. When N and n are unknown, we can only say that the remainder can go from 0 to $n-1$ and that all these cases are equiprobable. The expectation of this number is $(n-1)/2$. Thus, for a segment length of n tokens, we can estimate that on average $(n-1)/2$ tokens will have a weight of zero. Whether these tokens are frequent, rare or the only occurrence of a type in the whole text, they do not affect any calculated TTR. There is therefore an impact of the weight of a token on the lexical diversity score.

For MTTRSS, all tokens between positions 1 and $N-n+1$ have the same probability of starting a segment while tokens occupying positions $N-n+2$ to N cannot start a segment because it would be incomplete. A token at position i only intervenes in a given segment if the initial token of this segment is at one of the positions from $i-n+1$ to i inclusive.
It follows that:

- The first token will only be used when this first token is selected as the starting point. This will occur with a probability of $1/(N-n+1)$. The last token will be used only when the token in position $N-n+1$ is selected as starting point.
- The second token will be used only when the first token or the second one is selected as starting point. This will happen with a probability of $2/(N-n+1)$. This probability also applies to the penultimate token.
- The same reasoning applies to the tokens that occupy positions 3 to $n-1$ and $N-n+2$ to $N-2$. For example, the probability that the token in position $n-1$ occurs in a segment is $(n-1)/(N-n+1)$. It is also the probability that the token in position $N-n+2$ occurs in a segment.
- All the tokens occupying the positions going from n to $N-n+1$ have a probability to occur in a segment of $n/(N-n+1)$.

The weight of the tokens in the MTTRSS calculation is proportional to this probability.

In the case of MATTR, it is not correct to talk about the probability of intervening in a segment because the procedure is deterministic. On the other hand, it possible to quantify the

frequency with which a token will be used according to its position in the sequence, the weight being proportional to this frequency. The first token occurs only in one sequence, the one that begins with it. The second token occurs only in two sequences, those which begin with it and with the first. This reasoning applies to the tokens occupying the positions *3* to *n-1*. From the position *n* until the position *N-n+1*, the tokens occur in *n* sequences. From position *N-n+2* to *N*, the number of sequences in which a token occurs decreases linearly exactly as it increases linearly for the *n* first tokens of the text. For example, the last token occurs in only one sequence, the one starting at position *N-n+1*. This reasoning is illustrated in Table S2.1 which gives for *N=10* and *n=4* the number of segments in which a token in position *i* can occur in the calculation of MATTR.

Table S2.1

Number of segments in which a given token occurs in the calculation of MATTR for N=10 and n=4

Position	1	2	3	4	5	6	7	8	9	10
Presence in a segment	x	x	x	x						
		x	x	x	x					
			x	x	x	x				
				x	x	x	x			
					x	x	x	x		
						x	x	x	x	
							x	x	x	x
Number of segments in which the token occurs	1	2	3	4	4	4	4	3	2	1

The impact of the weight of a token on the MATTR score, which is proportional to the number of windows in which this token is present, can be illustrated by the following example that uses the approach proposed by Covington and McFall (2010) to discuss the properties of MATTR. Let us consider a sequence of ten tokens composed of a hapax and a single other word that is present nine times. The length of the window is set to four. Depending on the position of the hapax, the MATTR score varies from .286 to .393, as shown in Table S2.2. The only possible explanation for the differences in the scores is that the hapax does not always receive the same weight in the MATTR score.

Table S2.2: Computation of the MATTR scores for the ten sequences

Line	Sequence	MATTR score
1	ABBBBBBBBB	((2/4)*1+(1/4)*6)/7 = .286
2	BABBBBBBBB	((2/4)*2+(1/4)*5)/7 = .321
3	BBABBBBBBB	((2/4)*3+(1/4)*4)/7 = .357
4	BBBABBBBBB	((2/4)*4+(1/4)*3)/7 = .393
5	BBBBABBBBB	((2/4)*4+(1/4)*3)/7 = .393
6	BBBBBABBBB	((2/4)*4+(1/4)*3)/7 = .393
7	BBBBBBABBB	((2/4)*4+(1/4)*3)/7 = .393
8	BBBBBBBABB	((2/4)*3+(1/4)*4)/7 = .357
9	BBBBBBBBAB	((2/4)*2+(1/4)*5)/7 = .321
10	BBBBBBBBBA	((2/4)*1+(1/4)*6)/7 = .286

It is important to note that this example does not simply show that MATTR is sequence sensitive, which it is, because moving the hapax in the middle part of the sequence (lines 4 to 7) has no impact. It is only when the hapax is at the beginning or at the end of the sequence that an impact is observed; that is, when it receives a different weight. It is technically possible to build a MATTR index that allocates the same weight to all the tokens. To accomplish this, it is sufficient to use the wrap-around procedure of MTLD-W (Vidal & Jarvis, 2020) and thus to complete the incomplete windows at the end of the text using the tokens that are at the beginning of it. However, whether joining the end of a text to the beginning is more respectful of the sequential nature of a text than the global approach is questionable.

Since, to my knowledge, the problem encountered by MATTR is not highlighted in the existing literature, it is useful to formulate it also as follows. As the name suggests, MATTR is based on a moving average procedure. It is well known that this procedure does not allow to obtain a score for all points of a sequence. There are several formulations of this limitation depending on which point in the moving window the average value is assigned to: the last one as proposed by Covington and McFall (2010) or the middle one as it is frequently the case in time series. If the score is assigned to the last token, Covington and McFall (2010, p. 96) observes that "MATTR yields a value for every point in the text except for those less than one window length from the beginning." When the average of the obtained scores is computed, the first tokens of the text are not taken into account in the same way as the

following ones. These tokens have therefore a different weight in the final score. If the score is assigned to the middle token, MATTR lacks information at the beginning and at the end of the sequence: "Moving averages do not allow estimates of *f(t)* near the ends of the time series (in the first *k* and last *k* periods)" (Hyndman, 2011, p. 867). The closer to the beginning (or to the end) of the sequence, the more information is missing. It is this lack of information that is quantified by the token weight.

References

Covington, M. A., & McFall, J. D. (2010). Cutting the Gordian knot: The moving-average type-token ratio (MATTR). Journal of Quantitative Linguistics, 17(2), 94-100.

Hyndman, R. J. (2011). Moving Averages. In: M. Lovric (Ed.), *International Encyclopedia of Statistical Science* (pp. 866-869). Springer.

Vidal, K., & Jarvis, S. (2020). Effects of English-medium instruction on Spanish students' proficiency and lexical diversity in English. *Language Teaching Research, 24*(5), 568-587.

Appendix S3: Additional Information for Table 2

Table S3.1

Means, Standard Deviations and Confidence Limits for the Four Lengths for the Parallel and Random Sampling Methods

	Parallel Sampling			Random Sampling		
Sample length	*M*	*SD*	CI	*M*	*SD*	CI
60	.7740	.037	[.7687, .7794]	.7803	.035	[.7752, .7854]
80	.7744	.037	[.7691, .7798]	.7803	.035	[.7752, .7854]
120	.7750	.036	[.7698, .7800]	.7802	.035	[.7751, .7853]
240	.7753	.036	[.7701, .7804]	.7802	.035	[.7751, .7853]

The 95% confidence intervals for the ANOVA effect size, partial eta^2, are as follows:

Parallel Sampling: partial eta^2 = .007, [.000, .031]

Random Sampling: partial eta^2 = .058, [.003, .121]

Appendix S4: Procedure for Selecting 12 Profiles in the Graphs

1. Calculate the differences in the scores between the conditions taken two by two for each text.
2. For each text, count the number of times a difference is among the four largest differences or among the four smallest differences.
3. Select

- the four texts that have most of the largest differences,
- of those that remain, select the four texts that have most of the small differences, and
- of those that remain, the select four texts that have the most extreme differences (large and small).

Appendix S5: Results of Steiger's (1980) Test

Table S5.1 shows the full results of Steiger's (1980) test for two non-independent correlations that was used to determine whether there was a statistically significant difference between the largest and smallest correlation of each index with the text quality. The limits of the 95% confidence interval for the difference were obtained using Zou's (2007) procedure.

Table S5.1

Comparison of the Largest and Smallest Correlations between an Index and the Text Quality for ICNALE and ICLE

ICNALE						
Index	Largest *r*	Smallest *r*	*t*(185)	*p*	Lower CI	Upper CI
HDD	.342	.285	1.45	.1474	-.019	.132
MATTR	.391	.341	0.87	.3863	-.061	.159
MSTTR	.385	.321	1.01	.3119	-.059	.188
MTTRSS	.373	.299	1.18	.2383	-.048	.196
MTLD	.320	.242	3.18	.0017	.029	.127
ICLE N = 223 Df = 120						
Index	Largest *r*	Smallest *r*	*t*(220)	*p*	Lower CI	Upper CI
HDD	.515	.480	1.43	.1537	-.013	.082
MATTR	.474	.445	1.55	.1234	-.008	.066
MSTTR	.501	.417	1.97	.0506	-.000	.168
MTTRSS	.471	.294	3.35	.0009	.072	.283
MTLD	.461	.436	1.25	.2113	-.014	.064

References

Steiger, J. H. (1980). Tests for comparing elements of a correlation matrix. Psychological Bulletin, 87(2), 245-251.

Zou, G. Y. (2007). Toward using confidence intervals to compare correlations. Psychological Methods, 12(4), 399-413.

Appendix S6: Analysis of English Learners' Monologues

The main objective of the study was to determine the extent to which LD indices are sensitive to the length of the texts to which they are applied. Empirical evaluations were performed on two datasets that were relatively different in terms of text length and the participants' L1s. Both datasets consistently led to the same conclusions. This result was expected because the study focused on the logical-mathematical properties of indices and evaluation procedures. Nevertheless, the selected datasets both contained written essays. To further support the generality of the findings, this appendix presents the full analysis of a third dataset that consisted of oral productions. The conclusions reached via the analyses were identical to those reported in the main text.

Dataset

The spoken data used for these analyses were taken from the Corpus of English as a Foreign Language (COREFL: corefl.learnercorpora.com, version 1.0 - September 2021), which is freely available under a Creative Commons license (CC BY-NC-ND 3.0 ES). Compiled since 2012 at the University of Granada, it includes written and spoken texts from Spanish and German learners of English and from native English speakers (Lozano, Díaz-Negrillo, & Callies, 2021). The analyzed sub-corpus consists of monologues that were recorded while the English learners were asked to retell a short clip from Charles Chaplin's film *The Kid* (available at https://www.youtube.com/watch?v=eO1HvF2G2Sw). The audio files were transcribed orthographically and were tokenized by the team that collected them. The pre-processing consisted of removing the filled pauses, such as "uh" or "um", and false starts, such as " in= inside". The 130 texts containing at least 240 tokens were analyzed. The average length was 417 tokens (SD = 104, min = 261, max = 729). All the learners included in this dataset answered the Oxford Quick Placement Test to evaluate their levels of English proficiency.

Analyses and results

The analyses were identical to those described in the main text.

First Problem: Evaluation of the Text Length Sensitivity

ICC Analysis and Graphical Representation of the Profiles

Figure S6.1 shows the ICCs for the four evaluation methods and the ten indices. The mean ICC is represented by a circle, while the bars represent the 95% confidence interval for the ICC population values. As explained in the main text, the parallel sampling method is problematic and its conclusions are not reliable. Figures S6.2 to S6.5 show the impact of length on the indices using the LD text profiles.

TTR, Guiraud, Herdan, Maas, and MTTRSS clearly showed a length bias. The profiles for MATTR, MSTTR, MTTRSS, and MTLD did not show any bias, but there were important crossings, which explained why the ICC was sometimes quite far from 1. Only the profiles for HD-D were almost perfectly stable.

Figure S6.1

ICCs for Text Length Sensitivity for COREFL

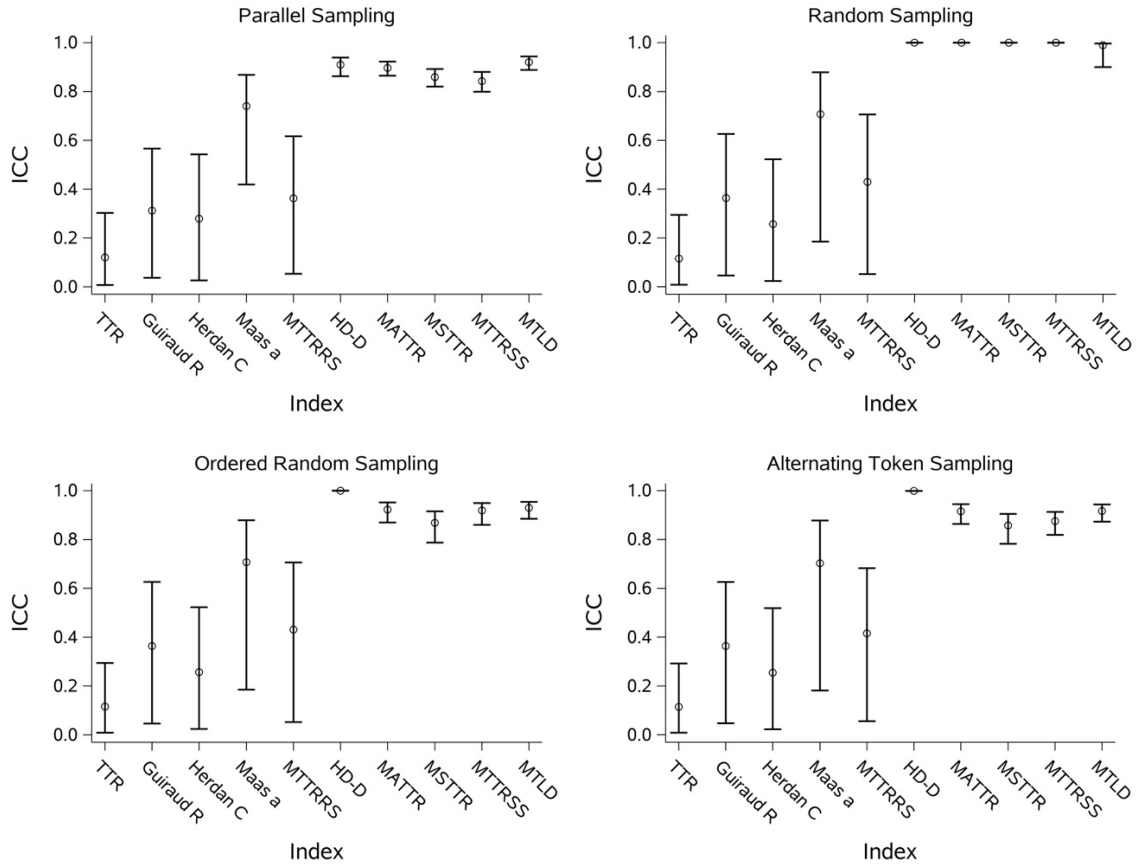


Figure S6.2

Profile Graphs for COREFL in the Parallel Sampling Method

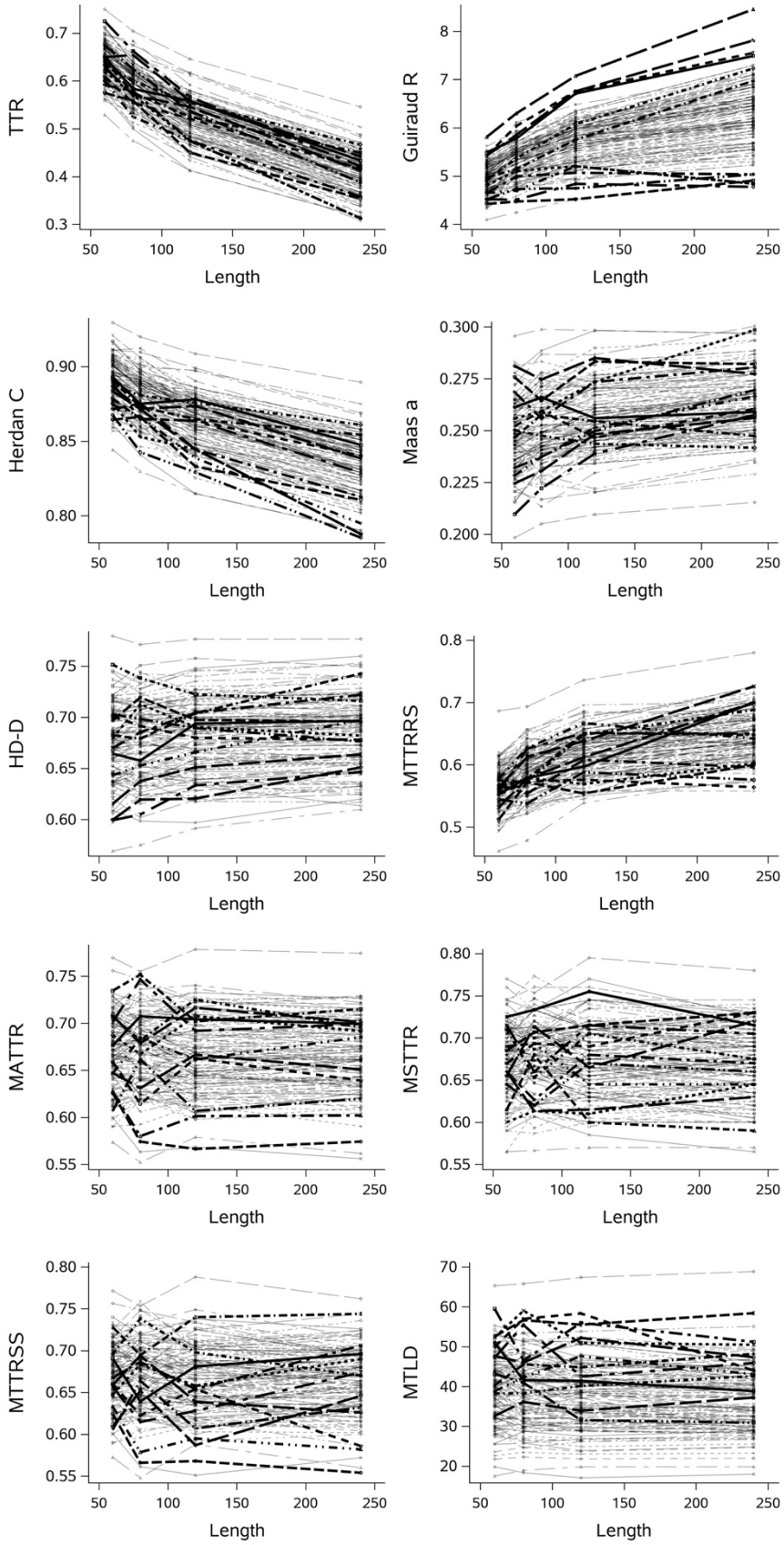


Figure S6.3

Profile Graphs for COREFL in the Random Sampling Method

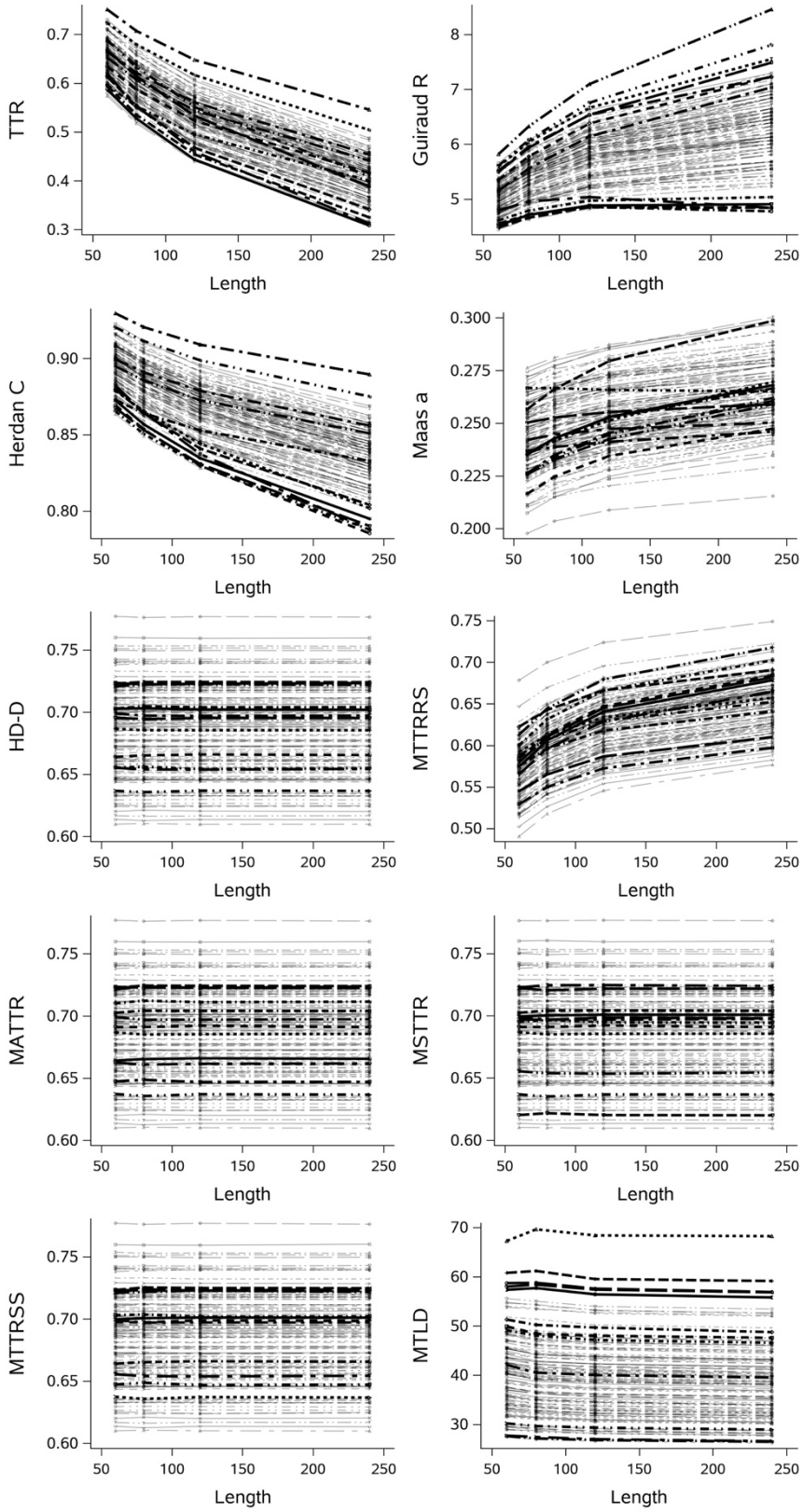


Figure S6.4

Profile Graphs for COREFL in the Ordered Random Sampling Method

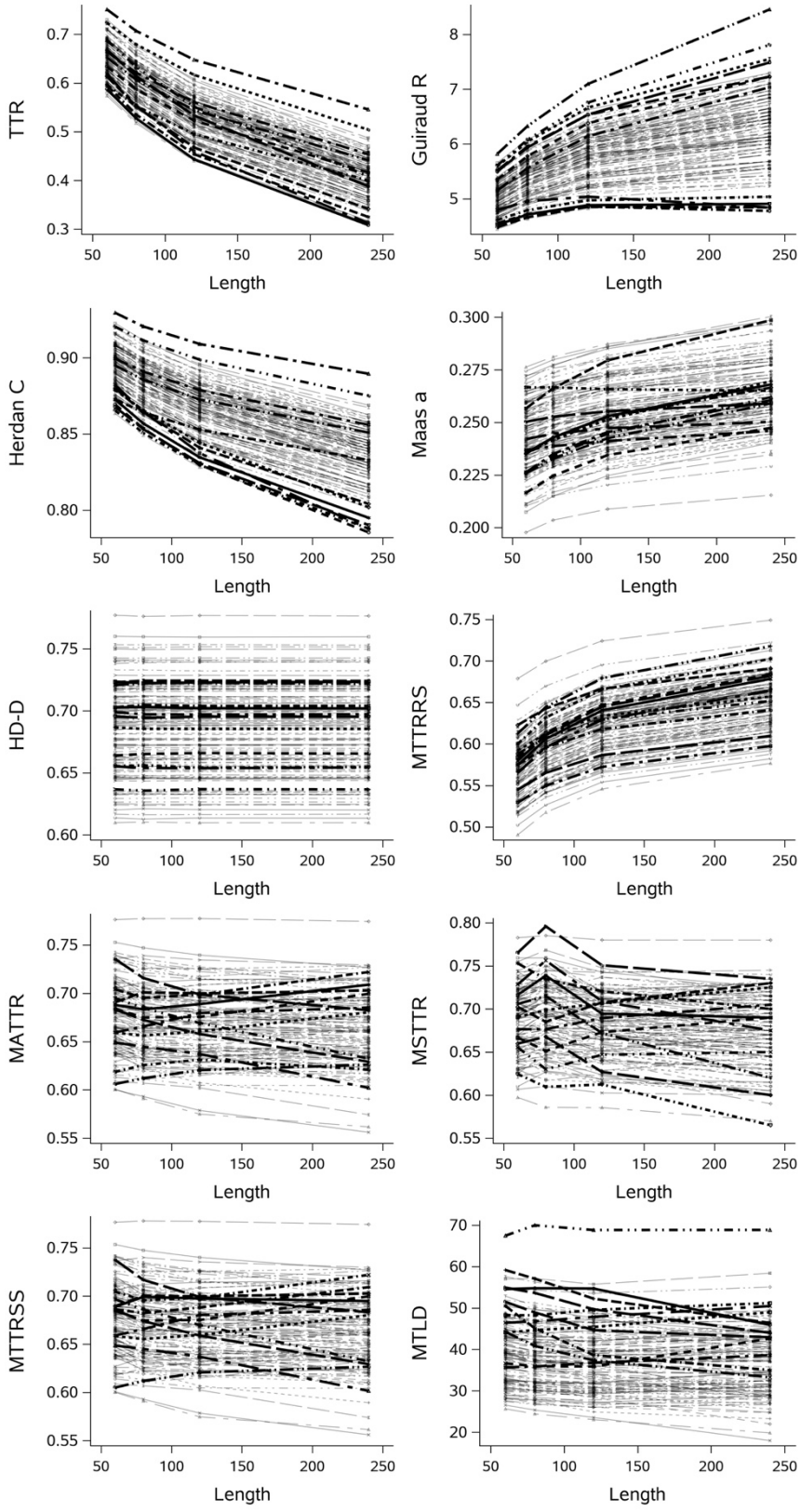
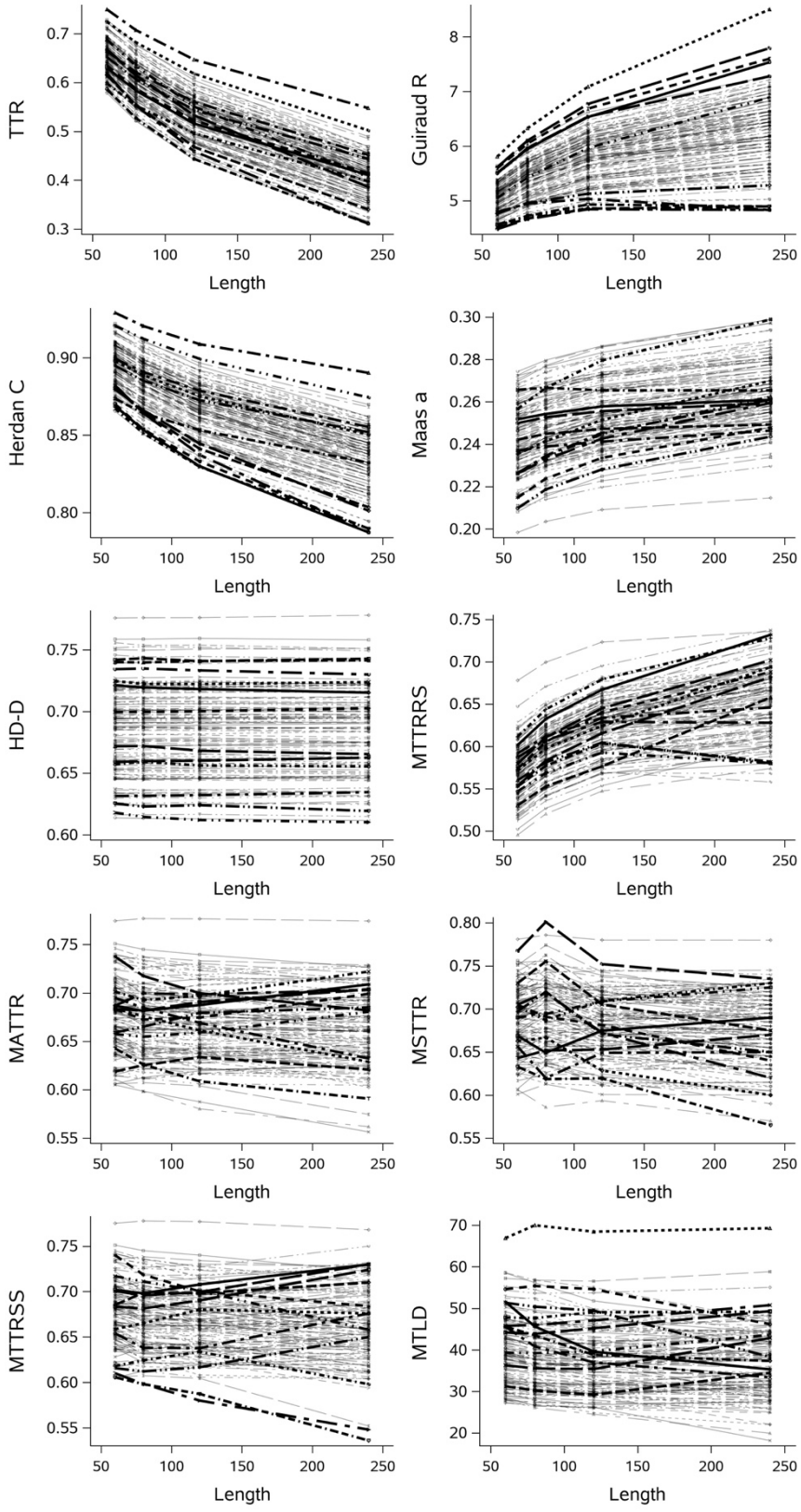


Figure S6.5

Profile Graphs for COREFL in the Alternating Token Sampling Method



Second Problem: Impact of the Index Parameter

Figure S6.6 shows the consistency of the indices as a function of the parameter value measured by the ICC. MTLD and HDD were less affected, followed by MATTR. Figure S6.7 illustrates the impact of parameter manipulation on the LD scores. In this figure, the scores for each index for each value of the parameter were centered on 0 in order to eliminate bias and to match the data used for the ICC precisely when it measures consistency. For all the indices, the LD score of some texts increased with the increase in the parameter value, while this was the opposite for other texts and produced profile crossings.

Figure S6.6

ICCs for the Impact of the Index Parameter for COREFL

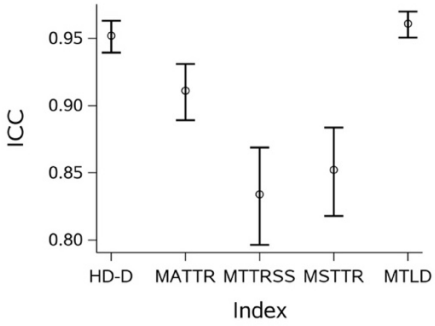
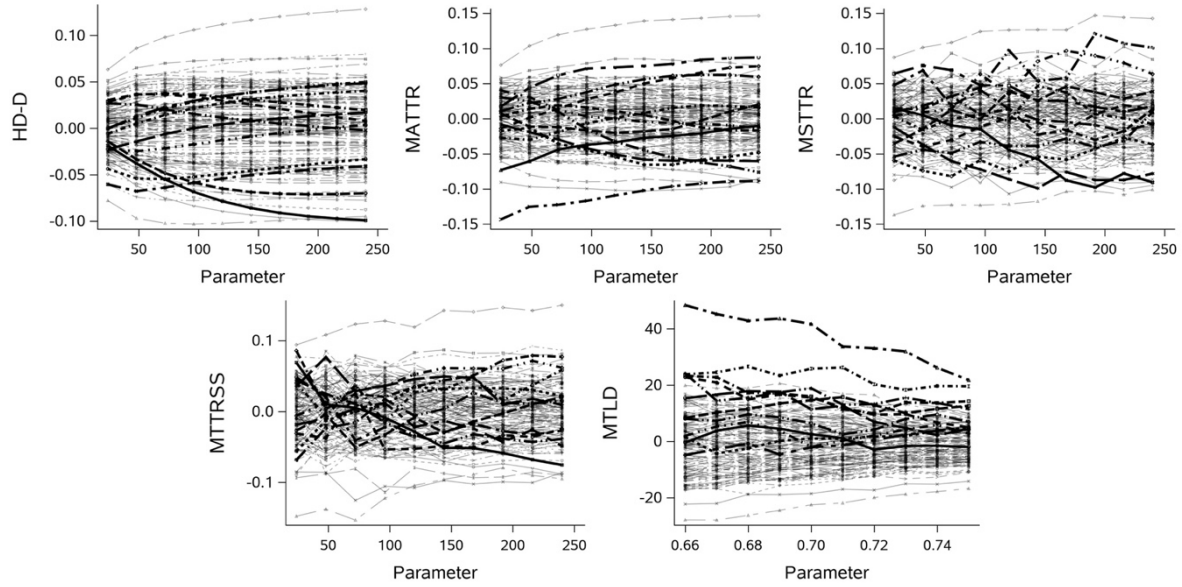


Figure S6.7

Profile Graphs for the Parameter Impact for COREFL



Impact on the Correlation between LD and English Proficiency level

Table S6.1 shows the correlation between the LD score and the learner's level of English proficiency for each parameter value. The largest value for each index is in bold, and the

smallest is underlined. As shown in Table S6.2, none of the differences between the largest and smallest correlations were statistically significant according to Steiger's (1980) test.

Table S6.1

Correlations between the LD and the Text Quality for COREFL for Different Parameter Values

Length	HD-D	MATTR	MSTTR	MTTRSS	Factor	MTLD
24	.358	.374	.313	.306	.66	.300
48	.394	.375	.279	.316	.67	.306
72	.408	.370	.343	.370	.68	.330
96	.412	.382	.311	.374	.69	.299
120	**.413**	**.385**	.312	.380	.70	.311
144	.411	.383	.304	**.402**	.71	.313
168	.410	.377	**.346**	.365	.72	.331
192	.408	.371	.314	.352	.73	.321
216	.406	.358	.330	.354	.74	.314
240	.404	.335	.336	.309	.75	**.332**

Table S6.2

Comparison of the Largest and Smallest Correlations between an Index and the Text Quality for COREFL

	CORELF					
Index	Largest *r*	Smallest *r*	*t*(127)	*p*	Lower CI	Upper CI
HDD	.413	.358	1.58	.117	-.013	.123
MATTR	.385	.335	1.85	.066	-.003	.103
MSTTR	.346	.279	1.25	.213	-.038	.172
MTTRSS	.402	.306	1.43	.154	-.035	.228
MTLD	.331	.299	1.16	.249	-.022	.086

Conclusion

The analysis of this third dataset confirmed the conclusions for the two written datasets presented in the main text.

References

Lozano, C., Díaz-Negrillo, A., & Callies, M. (2020). Designing and compiling a learner corpus of written and spoken narratives: COREFL. In C. Bongartz & J. Torregrossa (Eds.), *What's in a Narrative? Variation in Story-Telling at the Interface between Language and Literacy* (pp. 21-46). Peter Lang.

Steiger, J. H. (1980). Tests for comparing elements of a correlation matrix. Psychological Bulletin, 87(2), 245-251.

Appendix S7: Obtaining the Main LD Indices by Means of Two Freely Available Tools

This appendix explains how to obtain the main LD indices that were analyzed in this paper by means of two freely available tools, TAALED and the koRpus package. It also shows that the indices calculated by these tools were identical to those calculated by the SAS program used in this paper. This SAS program is included at the end.

TAALED (Kyle et al., 2021) is standalone software that is designed to calculate LD indices. It is controlled through an easy-to-use window interface.

The koRpus package (Michalke, 2020) works with R software. A text can be analyzed using the following commands.

```
-----
install.packages("koRpus")
install.koRpus.lang(c("en"))
library("koRpus")
library(koRpus.lang.en)
setwd("/Users/c/Desktop/Example")
tagged.text.obj <-
tokenize("./AliceNoun2.txt",lang="en",detect=c(parag=TRUE, hline=TRUE))
TTR<-TTR(tagged.text.obj)
TTR@TTR
R<-R.ld(tagged.text.obj)
R@R.ld
C<-C.ld(tagged.text.obj)
C@C.ld
maas<-maas(tagged.text.obj)
maas@Maas
HDD<-HDD(tagged.text.obj)
HDD@HDD
MATTR<-MATTR(tagged.text.obj,window=50)
MATTR@MATTR
MSTTR<-MSTTR(tagged.text.obj,segment=50)
MSTTR@MSTTR
MTLD<-MTLD(tagged.text.obj,MA=FALSE)
MTLD@MTLD
-----
```

As an example, I used the beginning of Lewis Caroll's *Alice in Wonderland*, which is the text that was used as the main example by Baayen (2001), and stopped the excerpt after the first 100 types, or 165 tokens. The tokenized extract is presented below.

alice was beginning to get very tired of sitting by her sister on the bank and of having nothing to do once or twice she had peeped into the book her sister was reading but it had no pictures or conversations in it and what is the use of a book thought alice without pictures or conversations so she was considering in her own mind as well as she could for the hot day made her feel very sleepy and stupid whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies when suddenly a white rabbit with pink eyes ran close by her there was nothing so very remarkable in that nor did alice think it so very much out of the way to hear the rabbit say to itself oh dear oh dear i shall be late when she thought it over afterwards it occurred to her that she ought to have wondered at this

The main difficulty when comparing several types of text analysis software is that they frequently perform a re-tokenization of the text, which includes the modification of some tokens and even the deletion of others. As was the case for this extract, each type was replaced by a noun that was treated in the same way by the evaluated tools. For example, *Alice* was recoded as *art* and *the* as *two*. The recoded extract is presented below.

art team case state friend teacher side lot president city hand power member two business back lot group life state fact minute moment student point girl name home two change hand power team party child house girl law number moment country history house back war hour two system lot air change service art word number moment country program point team company history hand mother kid body time body point day force two head end issue hand father teacher problem back question way two others lot job air education year car world two story lot game study back night two door water reason air week parent woman office family part community city hand right team life program teacher people history research level face art room house program teacher kind month lot two thing state guy two parent person state idea man eye man eye health place car information water point service house morning area house line state hand research point money state government work book school

Table S7.1 provides the main indices that were computed using this extract by the three approaches. These indices are identical except for that of Maas. The reason is that TAALED calculates a^2 while koRpus and the SAS program calculate a. The base 10 is used for the logarithm, as in Maas (1972).

The SAS program for this example is available through the following OSF repository: https://osf.io/5xpcw/ (Cr01LexDivAnaExample.sas).

Table S7.1

LD Indices Produced by Three Types of Software

	Token	Type	TTR	Guiraud	Herdan	Maas	HD-D	MATTR	MSTTR	MTLD
TAALED	165	100	.6061	7.7850	.9019	.0442	.8278	.7993	.8133	77.8337
koRpus	165	100	.6061	7.7850	.9019	.1386	.8278	.7993	.8133	77.8337
SAS	165	100	.6061	7.7850	.9019	.1386	.8278	.7993	.8133	77.8337

Note: HD-D is called ATTR in koRpus.

References

Baayen, R. H. (2001). Word Frequency Distributions. Springer.

Bestgen, Y. (2023). Code repository for "Measuring Lexical Diversity in Texts: The Twofold Length Problem." https://osf.io/5xpcw/

Kyle, K., Crossley, S. A., & Jarvis, S. (2021) Assessing the validity of lexical diversity indices using direct judgements. *Language Assessment Quarterly, 18*, 154-170.

Maas, Heinz-Dieter (1972). Über den Zusammenhang zwischen Wortschatzumfang und Länge eines Textes. *Zeitschrift für Literaturwissenschaft und Linguistik, 8*, 73-96.

Michalke, M. (2020). koRpus: Text Analysis with Emphasis on POS Tagging, Readability and Lexical Diversity (Version 0.13-4). Available from https://reaktanz.de/?c=hacking&s=koRpus

Appendix S8: SAS Code and R Function Used

The SAS code and the R function used in this study are available through the following OSF repository: https://osf.io/5xpcw/ (SASCodeAndR.zip). The following programs are included.

For the example:

Cr01LexDivAnaExample.sas

For the experiments:

The programs take an SAS dataset containing one observation per token with a variable that identifies the text and another that indicates the order of the tokens in the text as input.

As an example, Cr11ICNALERead.sas reads all the tokenized ICNALE files in a folder and puts them in this format

*Calculate the indices for the four methods

Cr21LexDivGenFileAllMethods.sas

Output for Parallel Sampling: zIceMeth_pasa

Output for Random Sampling and Ordered Random Sampling: zIceMeth_rnd2

Output for Alternating Token Sampling: zIceMeth_splitr

*Calculate the indices for the parameter analysis

Cr22LexDivGenFileParameters.sas

Output for HDD: zIceParam_hdd

Output for MATTR: zIceParam_mattr

Output for MSTTR: zIceParam_msttr

Output for MTTRSS: zIceParam_mttrss

Output for MTLD: zIceParam_mtld

*Output files to calculate ICCs in R for the methods

Cr31LexDivMethodsOutForICC.sas

Output: a txt file for each method and each index named zicemeth_pasa_ttr.txt (for example)

*Output files to calculate ICCs in R for the parameters

Cr32LexDivParametersOutForICC.sas

Output: a txt file for each index named zIceParam_hdd.txt (for example)

*Calculate ICCs in R for the methods
MyICCMeth.R
Output: ICCResMeth.txt

*Calculate ICCs in R for the parameters
MyICCParam.R
Output: ICCResParam.txt

*ICC graphs for the methods
Cr41LexDivMethodsGraphICC.sas

*Profile graphs for the methods
Cr42LexDivMethodsGraphIdx.sas

*ICC graphs for the parameters
Cr43LexDivParametersGraphICC.sas

*Profile graphs and correlations for the parameters
Cr44LexDivParametersGraphIdx.sas

*Program for Figure 6
Cr51SimulHdd.sas

References

Bestgen, Y. (2023). Code repository for "Measuring Lexical Diversity in Texts: The Twofold Length Problem." https://osf.io/5xpcw/

Appendix S9: Implementing the Sampling Methods in Python

In order to facilitate the replication of the analyses, but also the application of the proposed sampling methods to other datasets (L1, longer or shorter texts...), this appendix provides two Python scripts, available through the following OSF repository: https://osf.io/5xpcw/ (PythonScripts.zip), that implement the random sampling method, the ordered random sampling method and the alternating token sampling method.

Rand_OrdRand.py implements the random sampling method and the ordered random sampling method so that the same samples of tokens are analyzed by both methods. The only difference is that the tokens are replaced in their original order before the indices are computed for the ordered random sampling method but not for the random sampling method. To do so, it is not the tokens in the text that are permuted, but an index vector containing integers ranging from 0 to the number of tokens in the text minus 1 (in Python, indexing starts at 0). To obtain a sample of *m* tokens using the random sampling method, the tokens in the text at the first *m* positions of the permuted index vector are extracted. For the ordered random sampling method, the same tokens are extracted, but they are reordered according to the indexes.

Alternating.py implements the alternating token sampling method which generalize the split-half procedure proposed by McKee et al. (2000). This method constructs samples containing half, a third or a quarter of the tokens in the text by randomly selecting one token out of two, three or four successive tokens. To do so, the tokens are permuted independently within each successive snippet of two, three or four tokens (the *Pnsplit* parameter) and the samples are obtained by taking one token out of *Pnsplit* tokens, starting with the first, then the second, up to the *Pnsplit*-th.

These scripts read as input a .txt file containing a tokenized text (see TaaledSample.txt in the archive). They provide two types of output.

The samples produced by each method.
Two parameters (SHOW_INSTANCE and N_SHOW_INSTANCE) allow to obtain the samples produced by each method. To highlight the characteristics of these samples, the scripts can be applied to a pseudo-text composed of numbers from 1 to 303 (Numbers.txt in the zip file). By way of example, here are the first samples produced by the random sampling method and by the ordered random sampling method on the basis of this pseudo-text:

```
Random | Sample length = 151 | Niter = 1
['253', '72', '169', '301', '98', '95', '263', '81', '182', '275', '87', '86',
'297', '32', '281', '54', '163', '97', '108', '104', '298', '139', '93', '101',
'91', '45', '217', '242', '183', '194', '248', '254', '129', '114', '172', '110',
'197', '218', '200', '160', '211', '162', '208', '187', '165', '255', '273', '43',
'238', '245', '264', '125', '35', '29', '173', '221', '219', '156', '33', '198',
'249', '294', '116', '234', '90', '170', '246', '30', '7', '83', '178', '224',
'303', '220', '189', '196', '209', '205', '120', '150', '34', '71', '250', '47',
'4', '192', '282', '181', '144', '158', '277', '203', '113', '299', '107', '126',
'268', '223', '239', '215', '10', '293', '259', '8', '236', '240', '302', '79',
'283', '118', '143', '272', '204', '167', '166', '53', '100', '52', '140', '199',
'74', '12', '122', '149', '300', '175', '258', '288', '103', '124', '176', '60',
'216', '66', '155', '260', '41', '186', '89', '94', '230', '243', '127', '214',
'292', '287', '290', '57', '152', '190', '70']
Ordered Random | Sample length = 151 | Niter = 1
['4', '7', '8', '10', '12', '29', '30', '32', '33', '34', '35', '41', '43', '45',
'47', '52', '53', '54', '57', '60', '66', '70', '71', '72', '74', '79', '81', '83',
'86', '87', '89', '90', '91', '93', '94', '95', '97', '98', '100', '101', '103',
'104', '107', '108', '110', '113', '114', '116', '118', '120', '122', '124', '125',
```

```
'126', '127', '129', '139', '140', '143', '144', '149', '150', '152', '155', '156',
'158', '160', '162', '163', '165', '166', '167', '169', '170', '172', '173', '175',
'176', '178', '181', '182', '183', '186', '187', '189', '190', '192', '194', '196',
'197', '198', '199', '200', '203', '204', '205', '208', '209', '211', '214', '215',
'216', '217', '218', '219', '220', '221', '223', '224', '230', '234', '236', '238',
'239', '240', '242', '243', '245', '246', '248', '249', '250', '253', '254', '255',
'258', '259', '260', '263', '264', '268', '272', '273', '275', '277', '281', '282',
'283', '287', '288', '290', '292', '293', '294', '297', '298', '299', '300', '301',
'302', '303']

Random | Sample length = 101 | Niter = 1
['164', '258', '264', '261', '59', '118', '36', '222', '28', '185', '275', '260',
'287', '210', '8', '148', '217', '121', '52', '292', '282', '192', '29', '207',
'214', '45', '170', '31', '20', '278', '297', '158', '268', '169', '87', '291',
'21', '74', '76', '198', '95', '128', '152', '177', '237', '104', '155', '188',
'167', '295', '72', '129', '181', '213', '99', '252', '141', '30', '271', '3',
'56', '108', '130', '256', '126', '232', '142', '303', '143', '151', '246', '221',
'115', '228', '145', '204', '14', '176', '93', '39', '111', '84', '242', '86',
'231', '116', '83', '249', '112', '201', '103', '157', '247', '60', '131', '233',
'19', '27', '286', '190', '183']
Ordered Random | Sample length = 101 | Niter = 1
['3', '8', '14', '19', '20', '21', '27', '28', '29', '30', '31', '36', '39', '45',
'52', '56', '59', '60', '72', '74', '76', '83', '84', '86', '87', '93', '95', '99',
'103', '104', '108', '111', '112', '115', '116', '118', '121', '126', '128', '129',
'130', '131', '141', '142', '143', '145', '148', '151', '152', '155', '157', '158',
'164', '167', '169', '170', '176', '177', '181', '183', '185', '188', '190', '192',
'198', '201', '204', '207', '210', '213', '214', '217', '221', '222', '228', '231',
'232', '233', '237', '242', '246', '247', '249', '252', '256', '258', '260', '261',
'264', '268', '271', '275', '278', '282', '286', '287', '291', '292', '295', '297',
'303']

Random | Sample length = 75 | Niter = 1
['243', '32', '111', '156', '204', '301', '188', '14', '138', '13', '298', '151',
'129', '237', '164', '63', '97', '96', '287', '288', '108', '110', '15', '302',
'190', '8', '192', '274', '28', '168', '162', '73', '86', '67', '43', '139', '100',
'10', '145', '277', '35', '161', '84', '238', '265', '144', '50', '77', '226',
'90', '52', '80', '112', '158', '27', '94', '276', '117', '39', '270', '53', '184',
'150', '133', '72', '91', '38', '22', '25', '42', '292', '29', '223', '101', '259']
Ordered Random | Sample length = 75 | Niter = 1
['8', '10', '13', '14', '15', '22', '25', '27', '28', '29', '32', '35', '38', '39',
'42', '43', '50', '52', '53', '63', '67', '72', '73', '77', '80', '84', '86', '90',
'91', '94', '96', '97', '100', '101', '108', '110', '111', '112', '117', '129',
'133', '138', '139', '144', '145', '150', '151', '156', '158', '161', '162', '164',
'168', '184', '188', '190', '192', '204', '223', '226', '237', '238', '243', '259',
'265', '270', '274', '276', '277', '287', '288', '292', '298', '301', '302']
```

The HD-D and MATTR indices

To show how HD-D and MATTR can be calculated on the basis of the samples produced by each method, the scripts includes several Python functions (with slight modifications) from the underlying code for TAALED (Kyle et al., 2021), more precisely from ld_32.py, authored by Kristopher Kyle and available at https://github.com/LCR-ADS-Lab/TAALED/blob/main/dev/ld_32.py. By adjusting a parameter, the methods can be applied to an example text of TAALED (TaaledSample.txt). Here are the results obtained for each method by performing 5,000 randomizations:

```
Random   | Sample length =  140 | HDD =  0.871083 | MATTR = 0.850942
Ordered  | Sample length =  140 | HDD =  0.871083 | MATTR = 0.840838
Random   | Sample length =   93 | HDD =  0.871010 | MATTR = 0.851485
Ordered  | Sample length =   93 | HDD =  0.871010 | MATTR = 0.850711
Random   | Sample length =   70 | HDD =  0.871239 | MATTR = 0.851522
Ordered  | Sample length =   70 | HDD =  0.871239 | MATTR = 0.850743

Alternating | Segment length =  140 | HDD =  0.870906 | MATTR = 0.839625
Alternating | Segment length =   93 | HDD =  0.870912 | MATTR = 0.850383
Alternating | Segment length =   70 | HDD =  0.871491 | MATTR = 0.850603
```

References

Bestgen, Y. (2023). Code repository for "Measuring Lexical Diversity in Texts: The Twofold Length Problem." https://osf.io/5xpcw/

Kyle, K., Crossley, S. A., & Jarvis, S. (2021) Assessing the validity of lexical diversity indices using direct judgements. Language Assessment Quarterly, 18, 154-170.

McKee, G., Malvern, D., & Richards, B. (2000). Measuring vocabulary diversity using dedicated software. Literacy and Linguistic Computing, 15, 323-338.

Appendix S10: In-Depth Discussion of the Most Important Studies Claiming That HD-D is Sensitive to Text Length

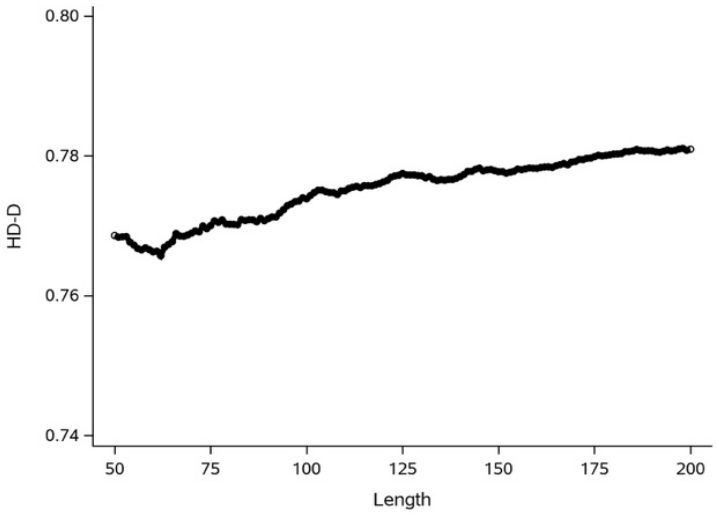
The present study highlights the extremely high efficiency of HD-D for controlling the length effect. This conclusion contradicts that of several previous studies. What follows is an in-depth discussion of these most important studies.

McCarthy, P. M., & Jarvis, S. (2007). vocd: A theoretical and empirical evaluation. Language Testing, 24, 459-488.

McCarthy and Jarvis (2007) argued that HD-D is sensitive to text length on the basis of three arguments. First, an empirical analysis (pp. 475-476) described as follows (ATTR-42 is the name used by McCarthy and Jarvis to refer to HD-D):

> To confirm that this method of measurement truly does exhibit text-length effects in our own database, we calculated ATTR-42 for the first 50 through the first 200 tokens of each of the 141 texts in our database that are longer than 200 words in length. (We used ATTR instead of SOP because the ATTR scale, which ranges from 0 to 1.00, is easier to interpret, and we used ATTR instead of D because ATTR is more precise and consistent.) Given that it is not feasible to illustrate the ATTR-42 growth curves for all 141 texts, we have chosen instead to show a single curve representing the central tendency of these texts. This is shown in Figure 5. This figure shows that after a slight dip in ATTR-42 from about the 50th to the 100th token, the central tendency is for the ATTR-42 of texts to move gradually and steadily upward.

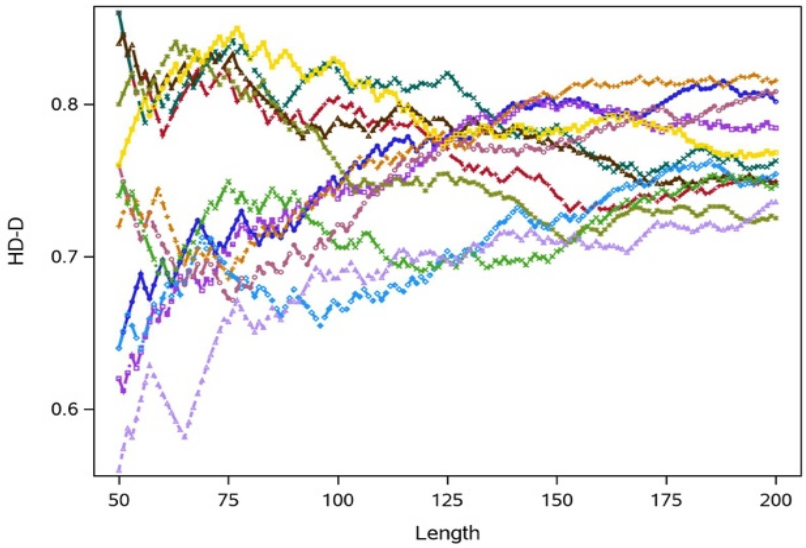
Figure S10.1: Evolution of the Mean HD-D Computed on the 188 Texts of ICNALE.



This demonstration is problematic because it relies on samples that obviously do not have the same lexical content, since the next token is added to the sample each time. If there is an evolution of lexical diversity in these texts, a length-insensitive index will detect it. To support this explanation, I performed McCarthy and Jarvis' analysis on the ICNALE dataset used in the manuscript. Figure S10.1 shows the evolution of the mean HD-D computed on the 188 texts. As in McCarthy and Jarvis' work, a small decrease is observed at the beginning of the curve, followed by an increase. Figure S10.2 shows the 12 profiles selected by the systematic procedure used in the manuscript; it shows that some profiles increased, but also that others decreased. This observation is incompatible with the assertion that HD-D

increases with text length. However, it is perfectly compatible with an evolution of lexical diversity in texts, which is one phenomenon that motivated Covington and McFall (2010) to develop MATTR.

Figure S10.2: Twelve Individual HD-D Profiles from ICNALE.



In the next part of McCarthy and Jarvis' (2007) work, which is one of the most important articles for the present study, the authors used the parallel sampling method (p. 479). As explained in the main text, these analyses cannot be used to claim that HD-D is sensitive to text length.

Finally, McCarthy and Jarvis (2007) proposed a mathematical argument by analyzing the impact of adding an occurrence of a type already present in a 100-word text on HD-D by varying the frequency of that type. The authors showed that the reduction of HD-D was increasingly lower the less frequent the type was, up to the point at which the addition of a second occurrence of a type that was only present once in the text produced an increase of HD-D and thus of lexical diversity. This observation, which the authors rightly called "surprising and profound" (p. 475) has noting to do with any sensibility to text length as explained at the end of the discussion of Figure 6 of the main text.

Fergadiotis, G., Wright, H. H., & Green, S. B. (2015). Psychometric evaluation of lexical diversity indices: Assessing length effects. Journal of Speech, Language, and Hearing Research, 58, 840-852.

Fergadiotis, G., Wright, H. H., & West, T. M. (2013). Measuring lexical diversity in narrative discourse of people with aphasia. American Journal of Speech-Language Pathology, 22, S397-S408.

Fergadiotis and colleagues (2013, 2015) also concluded that HD-D is somewhat sensitive to text length. Their work is based on a different approach from the one that is almost always used in the field. Instead of trying to determine directly whether LD indices are sensitive to text length, the authors used an indirect approach based on multidimensional analysis techniques (factor analysis and structural equation modeling). These techniques are mainly aimed at determining whether different indices are manifestations of the same latent variable;

that is, LD. It is well established that the results of such analyses are affected by the variables that are selected for the analysis:

> One of the characteristics of factor analysis that bothers many users is the fact that what you find depends on what you decide to analyze. If one dimension is overrepresented by tests, while another is underrepresented by tests, the former dimension will be the most dominant dimension in the analysis, which may or may not have anything to do with its potency in real life settings. Many users are familiar with this property, which might be called the 'what you get out of it depends on what you put into it' phenomenon (Dorans & Lawrence, 1999, p. 6).

For example, in one of their studies, Fergadiotis et al. (2015) analyzed two local indices, MATTR and MTLD, and one global index, vocd, as well as Maas' index, which is another global index that is well known to be length sensitive (McCarthy & Jarvis, 2013; Tweedie & Baayen, 1998). Further studies are needed to assess the impact of this choice on the results but, in general, Fergadiotis and colleagues' (2013, 2015) arguments are indirect, unlike those presented in the manuscript.

References

Covington, M. A., & McFall, J. D. (2010). Cutting the Gordian knot: The moving-average type-token ratio (MATTR). Journal of Quantitative Linguistics, 17(2), 94-100.

Dorans, N. J., & Lawrence, I. M. (1999). The Role of the Unit of Analysis in Dimensionality Assessment. Educational Testing Service, Research Report, RR-99-14.

McCarthy, P. & Jarvis, S. (2013). From intrinsic to extrinsic issues of lexical diversity assessment. An ecological validation study. In: S. Jarvis & M. Daller (Eds.), Vocabulary knowledge human ratings and automated measures (pp. 45-78). Benjamins.

Tweedie, F. J., & Baayen, R. H. (1998). How variable may a constant be? Measures of lexical richness in perspective. Computers & the Humanities, 32, 323-352.