Reproducing Shakespeare

In honor of Bradley Efron's 80th Birthday

Ronald A. Thisted
The University of Chicago

May 23, 2018

Abstract

In 1976, Efron and I published an article studying the problem of estimating the number of unseen species, using as a running example the (unseen) words in Shakespeare's vocabulary that he hadn't employed in the extant corpus of his work. The approaches we considered required computations that were not then—and are not today—in the repertoire of standard statistical software. Roughly half way between the appearance of our paper and today, Buckheit and Donoho (1995) introduced ideas of reproducible research. Here we consider the reproducibility of the figures and tables in the Shakespeare paper, what we might have done in 1976 that would have aided reproducibility in 2018, and what lessons that exercise suggests for maximizing the likelihood that today's computationally intensive research can be reproduced in decades to come.

Contents

1 Introduction

So on the tip of his subduing tongue
All kind of arguments and question deep,
All replication prompt and strong,
For his advantage still did wake and sleep.
— A Lover's Complaint

In 1976, Efron and I published a paper in *Biometrika* that explored aspects of the unseen species problem, *i.e.,* how to estimate the number of unseen types based on the distribution of counts of types that had been observed (Efron and Thisted, 1976), henceforth ET. The motivating—and captivating—example we used was to estimate the number of words that Shakespeare knew but which he had never been seen to use in his published corpus of plays and poetry. The paper generated considerable interest at the time, and a follow-on paper some ten years later concerning potential Shakespearean authorship of a (not very good) poem that had recently been discovered led to a minor industry in developing quantitative methods to judge matters of Shakespearean authorship. Originally viewed as frivolous by serious scholars of Shakespeare, "stylometrics" has come to have a recognized place in literary debate. [Mention *Federalist*, Elliott/Valenza, Foster].

In the ensuing years, interest in our original paper has been unabated, and it has continued to be cited—some 478 times as of May 2018 according to Google Scholar. Over the years a number of readers have contacted one or the other of us to suggest what we should have done, or to inform us of the logical flaws in our analysis, or most commonly, to ask questions about what we did or how we did it. We tried to respond to simple queries as best we could, and where we could not each of us would suggest to our correspondents that they contact the other.

Last year Brad received a query from a Chinese student who sought to understand how we got some of the results we reported in the 1976 paper, and Brad suggested that the student write to me. The main query was how we obtained the parameter estimates for the negative binomial model that we reported in Table 3. Surely, I thought, I could whip together a few lines of code that I could send to our correspondent to help him on his way. I soon discovered that it wasn't any longer quite obvious just what that negative binomial model consisted of—that is, the likelihood to maximize wasn't immediately obvious, even from reading our pellucid article. It also wasn't a straightforward task to get maximum likelihood estimates from any of the plausible candidate models. And there was a cryptic comment in the manuscript whose import also wasn't immediately apparent: "Table 3 shows the maximum likelihood fits, obtained by iterative search for various choices of x_0."

The challenge of reproducing a key ingredient of our 1976 paper evokes the current focus on "reproducible research," a concept introduced to the statistics community by Buckheit and Donoho in their influential 1995 paper, which is related to Donald Knuth's notion of "literate programming" (Knuth, 1984), and one for which I and many others have

advocated as a fundamental building block for quality science. Had Brad and I created a reproducible manuscript when we were writing our paper, in 2018 we would have the benefit of our 1976 selves both telling us and showing us exactly what they had done.

It seemed like a fitting project to explore the answers to two questions: (1) what would a reproducible manuscript for the Shakespearean vocabulary paper look like today, and (2) what challenges might thwart achieving a result that would still be useful some 40+ years after the original writing.

2 Elements of reproducibility

In this project, we take "reproducibility" to mean the ability to reproduce, from the original data, the main computations, tables, and figures from a report of research work. Ideally, a compendium of the artifacts that enable a work to be reproduced, along with the work itself, will be available to the scientific community to check, to modify, and to build upon.

There are (at least) four components of a research project compendium that affect the ability for the final article to be reproduced:

1. the computer programs or scripts used for calculation,
2. the underlying data,
3. the hardware and operating system under which those programs were run, and
4. the choices of inputs to the calculations, and the process by which those choices were made.

In our case we shall focus on a subset of the main calculations and figures from Efron and Thisted (1976). In the Shakespeare case, we are up against some challenges in each of these domains listed above, however in this talk we shall focus primarily on the first of these factors, with a few words reserved for the other three items. Specifically, we shall take as given the raw data reported in ET, and then undertake to reproduce calculations in the paper conditional on the reported data.

The data, programs, and results of this investigation are available from a GitHub repository (`https://github.com/rthisted/EfronAt80`), from which they can be freely accessed.

2.1 Computer programs

One of the key ideas of reproducible research is that it is a means to communicate not only with colleagues who may wish to understand or to build on our work, but also to communicate with our future selves when we seek to remind ourselves of what we did and how we did it. Since there is no GitHub repository of the work we did—GitHub not

Table 1. *Shakespeare's word type frequencies*

x	1	2	3	4	5	6	7	8	9	10	Row total
0+	14376	4343	2292	1463	1043	837	638	519	430	364	26305
10+	305	259	242	223	187	181	179	130	127	128	1961
20+	104	105	99	112	93	74	83	76	72	63	881
30+	73	47	56	59	53	45	34	49	45	52	513
40+	49	41	30	35	37	21	41	30	28	19	331
50+	25	19	28	27	31	19	19	22	23	14	227
60+	30	19	21	18	15	10	15	14	11	16	169
70+	13	12	10	16	18	11	8	15	12	7	122
80+	13	12	11	8	10	11	7	12	9	8	101
90+	4	7	6	7	10	10	15	7	7	5	78

Entry x is n_x, the number of word types used exactly x times. There are 846 word types which appear more than 100 times, for a total of 31 534 word types.

Figure 1: Table 1 from Efron and Thisted 1976

having been invented until 33 years after we started our work—I had to rely on the next best thing, namely, paper files from that era. Those files are part of my papers that have undergone many moves and many downsizings; it wasn't obvious to me that I would have kept anything of value. But as it turns out, at least some useful material survived, a testament to the enduring value of paper records, a storage medium and communications format that has survived the test of time.

The ET paper relies heavily on computations based on the raw data in Table 1 of ET, reproduced in Figure 1, for which custom programs were written, together with some calculations that were apparently done "by hand" using the calculator technology of the day.

The main computations appear in Tables 2 through 7. Table 3 shows maximum likelihood parameter estimates for Fisher's (truncated) negative binomial model for various choices of the truncation parameter, and Table 2 shows the estimated means for word types seen exactly x times calculated from one set of the negative binomial parameters. Table 4 shows calculated Euler coefficients for the generalized linear vocabulary estimator. Table 5 contains calculated lower bounds for the expected number of new words that one would see in a body of work t times as large as Shakespeare's extant corpus, and Table 6 shows lower and upper bounds for the same quantities based on a linear programming formulation of the problem.

Table 3. *Maximum likelihood fits of the negative binomial model to the first x_0 values of* n_x; $\hat{\beta} = \hat{\gamma}/(1-\hat{\gamma})$

x_0	$\sum_{x=1}^{x_0} n_x$	$\hat{\alpha}$	$\hat{\gamma}$	$\hat{\beta}$	$\chi^2_{x_0-3}$
5	23517	−0·3834	0·9795	47·82	0·027
10	26305	−0·3906	0·9884	85·44	2·024
15	27521	−0·3889	0·9861	70·78	3·815
20	28266	−0·3901	0·9875	78·77	8·832
30	29147	−0·3944	0·9899	97·92	16·874
40	29660	−0·3954	0·9905	104·26	30·437

Figure 2: Table 3 from Efron and Thisted 1976. These are the main results whose reproducibility is at issue.

Computing circa 1975

The first task is to determine whether we can replicate the main results, here taken to be the contents of Table 3 from Efron and Thisted 1976[1]. These target results are displayed in Figure 2

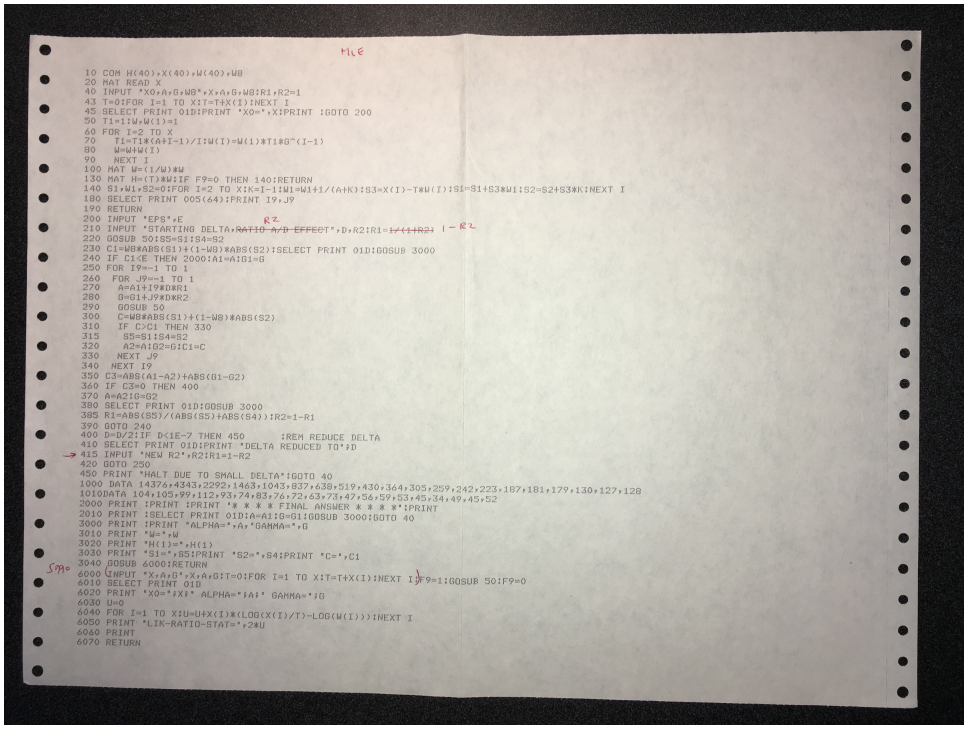
A starting point is what we did in 1975 when working on our paper.

The computer programs for those calculations used the technology of the day, which in this case involved writing programs in a dialect of BASIC, PL/I, and a script for a IBM proprietary mathematical programming system called MPS/360. As near as I can reconstruct, the PL/I programs were used to generate the input to the MPS software, which was used to solve a large linear optimization program. IBM discontinued sales and support of MPS in 2005. At the time `S`, (the statistical programming language that John Chambers developed at Bell Laboratories and the predecessor to `R`, did not yet exist. Indeed, there were no general-purposed statistical programming environments on which to draw.

Fortunately, one of us (Thisted) not only seems to have saved large chunks of the working papers from the Shakespeare project, but also to have refrained from tossing them out in the course of multiple relocations and office downsizings. And I was able to find what may be a program that was used to perform the calculations in Table 3. This program is shown in Figure 3.

This program was written for the Wang 2200 computer, the departmental computing device of the time, which could be only programmed in this dialect of BASIC. The program was transcribed, and is included both in Appendix A and in the GitHub repository `compendium/Programs/mle.bas.txt`. Unfortunately, there are no Wang 2200s available

[1]In this paper we shall focus on the calculations in Tables 2 and 3 to illustrate the issues involved; the calculations of Tables 4 through 6 raise additional difficulties which are beyond the scope of this project; we touch on those issues briefly.



```
MLE
 10 COM W(40),X(40),W(40),WB
 20 MAT READ X
 40 INPUT "XO,A,G,WB",X,A,G,WB:R1,R2=1
 43 T=0:FOR I=1 TO X:T=T+X(I):NEXT I
 45 SELECT PRINT 01D:PRINT "XO=";X:PRINT :GOTO 200
 50 T1=1:W,W(1)=1
 60 FOR I=2 TO X
 70   T1=T1*(A+I-1)/I:W(I)=W(1)*T1*G^(I-1)
 80   W=W+W(I)
 90   NEXT I
100 MAT W=(1/W)*W
130 MAT H=(T)*W:IF F9=0 THEN 140:RETURN
140 S1,W1,S2=0:FOR I=2 TO X:K=I-1:W1=W1+1/(A+K):S3=X(I)-T*W(I):S1=S1+S3*W1:S2=S2+S3*K:NEXT I
180 SELECT PRINT 005(64):PRINT I9,J9
190 RETURN
200 INPUT "EPS",E                                    R2
210 INPUT "STARTING DELTA,RATIO A/B EFFECT",D,R2:R1=1/(1+R2)   1 - R2
220 GOSUB 501S5=S1:S4=S2
230 C1=WB*ABS(S1)+(1-WB)*ABS(S2):SELECT PRINT 01D:GOSUB 3000
240 IF C1<E THEN 2000:A1=A:D1=G
250 FOR I9=-1 TO 1
260  FOR J9=-1 TO 1
270   A=A1+I9*D*R1
280   G=G1+J9*D*R2
290   GOSUB 50
300   C=WB*ABS(S1)+(1-WB)*ABS(S2)
310   IF C>C1 THEN 330
315    S5=S1:S4=S2
320    A2=A:G2=G:C1=C
330   NEXT J9
340  NEXT I9
350 C3=ABS(A1-A2)+ABS(G1-G2)
360 IF C3=0 THEN 400
370 A=A2:G=G2
380 SELECT PRINT 01D:GOSUB 3000
385 R1=ABS(S5)/(ABS(S5)+ABS(S4)):R2=1-R1
390 GOTO 240
400 D=D/2:IF D<1E-7 THEN 450      :REM REDUCE DELTA
410 SELECT PRINT 01D:PRINT "DELTA REDUCED TO";D
415 INPUT "NEW R2",R2:R1=1-R2
420 GOTO 250
450 PRINT "HALT DUE TO SMALL DELTA":GOTO 40
1000 DATA 14376,4343,2292,1463,1043,837,638,519,430,364,305,259,242,223,187,181,179,130,127,128
1010DATA 104,105,99,112,93,74,83,76,72,63,73,47,56,59,53,45,34,49,45,52
2000 PRINT :PRINT :PRINT "* * * * FINAL ANSWER * * * *":PRINT
2010 PRINT :SELECT PRINT 01D:A=A1:G=G1:GOSUB 3000:GOTO 40
3000 PRINT :PRINT "ALPHA=";A;"GAMMA=";G
3010 PRINT "W=";W
3020 PRINT "H(1)=",H(1)
3030 PRINT "S1=";S5:PRINT "S2=";S4:PRINT "C=",C1
3040 GOSUB 6000:RETURN
6000 INPUT "X,A,G",X,A,G:T=0:FOR I=1 TO X:T=T+X(I):NEXT I:F9=1:GOSUB 50:F9=0
6010 SELECT PRINT 01D
6020 PRINT "XO=";X;" ALPHA=";A;" GAMMA=";G
6030 U=0
6040 FOR I=1 TO X:U=U+X(I)*(LOG(X(I)/T)-LOG(W(I))):NEXT I
6050 PRINT "LIK-RATIO-STAT=",2*U
6060 PRINT
6070 RETURN
```

Figure 3: Program used to calculate maximum likelihood estimates in Table 3 of Efron and Thisted 1976

to me on which I can run this program which, I note, requires manual input of starting values and convergence parameters.

The inputs and outputs to this program were not, unfortunately, among the pieces of papers that I preserved, although one page, shown in Figure 4, gives a hint, both as to where one set of values (corresponding to $x_0 = 40$) may have come from, and to how much the process of what actually transpired has been lost. It is also clear that the output in this figure was not produced by the computer program in Figure 3. It isn't clear whether the preserved output is from an early prototype of the final program, or the preserved program is an early prototype of the one that produced the output of the latter.

Recomputing Table 3 from Table 1

So how would we do this computation in 2018, and do so in a way that might be more readily reproduced 40 years hence than our work from 40 years ago seems to have been? The first step is to determine just what model is being fit. Although the paper itself isn't crystal clear on the point, the model corresponding to x_0 is simply fitting a multinomial with x_0 categories (corresponding to $x = 1$ through x_0), the probability p_x for category x

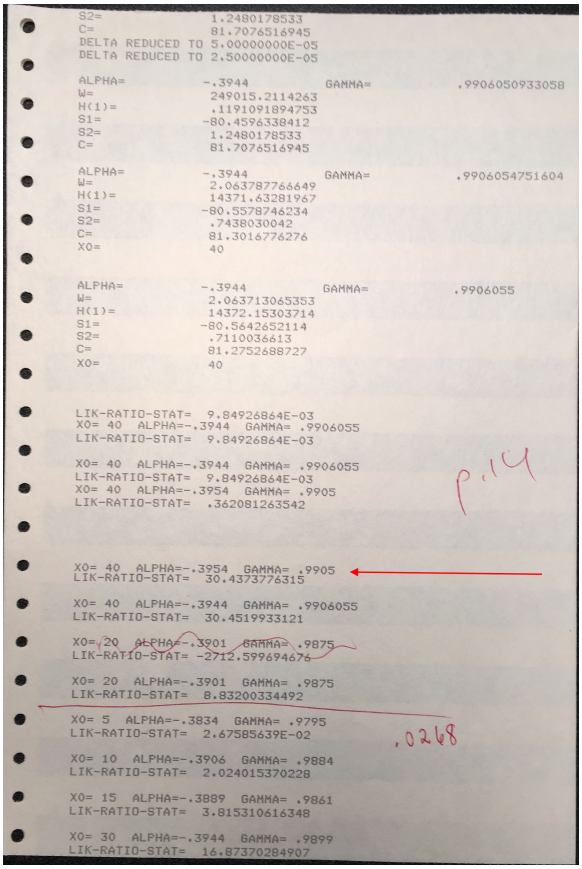


Figure 4: The apparent source for $\hat{\alpha}$ and $\hat{\gamma}$ corresponding to $x_0 = 40$ in Table 3 of Efron and Thisted 1976. The red arrow points to the parameter values that appear in the published table.

being proportional to

$$p_x \propto \frac{\Gamma(x+\alpha)}{x!\Gamma(1+\alpha)}\gamma^{x-1},$$

for $\gamma < 1$ and $\alpha > -1$, and where n_x is the number of word types observed in category x. The trickiness in the calculation results from the "proportional to" description of the probabilities, as the proportionality factor that ensures the probabilities sum to unity itself is a function of the parameters.

In response to one of the recent queries about how we got the values in Table 3, I resorted to the quickest (and dirtiest) method possible: I used the black-box optimizer built into Excel! This spreadsheet (`ET-Table3-origdata.xlsx`, locked 12:36pm CDT 12 May 2018) is available in the GitHub repository for those interested. This required building

x_0	$\sum n_x$	$\hat{\alpha}$	$\hat{\gamma}$	$\hat{\beta}$	$\chi^2_{x_0-3}$
5	23517	-0.3834	0.9795	47.71	0.027
10	26305	-0.3902	0.9883	84.48	2.024
15	27521	-0.3867	0.9854	67.42	3.752
20	28266	-0.3894	0.9872	77.20	8.822
30	29147	-0.3945	0.9899	98.18	16.873
40	29660	-0.3973	0.9912	112.22	30.437

Table 1: Maximum likelihood estimates calculated using Microsoft Excel's built-in optimizing routine.

a separate pane for each choice of x_0, in which the calculations were performed. The results analogous to ET's Table 3 are in the Summary pane, and are reproduced in Table 1.

A comparison of Table 3 from ET shown in Figure 2 and the results from Excel in Table 1 shows reasonably good agreement, although there are a few things to note. First, the results for $\hat{\alpha}$ agree in the first two decimal places, but diverge in the third, with the greatest divergence for $x_0 = 40$. Second, the χ^2 values for the Excel calculation are always less than or equal to those of the original calculation. This suggests that Excel is actually finding a slightly larger value for the maximized log likelihood. Third, although the χ^2 values are identical at $x_0 = 40$, the values for both $\hat{\alpha}$ and $\hat{\gamma}$ differ in the third decimal place, which suggests that the likelihood function is relatively flat in the neighborhood of the solution.

Of course if the goal is to maximize reproducibility in the future, an Excel spreadsheet is very nearly the worst possible choice. Such spreadsheets are easily corrupted by accident, provide no method for documenting changes, and are opaque as to computing formulæ and processes.

Today there are many well-supported programming environments with high-quality numerics and descriptive scripting languages that make even non-standard problems like that of fitting the Fisher negative binomial model relatively straightforward to solve. For purposes of recreating Table 3, I used Stata 2017, a computer package widely used by epidemiologists, biostatisticians, and econometricians, since it has a general-purpose maximum likelihood capability. Stata also has a strong focus on both the quality of the numerical algorithms it employs and on documentation of those algorithms, so using Stata affords a comparison with the black-box, unscripted Excel spreadsheet.

Solving the problem in Stata involves two scripts. The first contains a program that calculates log-likelihood values that Stata's built in maximum likelihood program (`ml`) calls when it is run; the second is the wrapper for the calculation that invokes the `ml` command as applied to the model at hand. These scripts are reproduced in Appendices B and C (as well as the GitHub repository at `compendium/Analysis/et.do` and `compendium/Analysis/table3mlFromTable1.do`, respectively.) Both scripts include the

x_0	$\sum n_x$	$\hat{\alpha}$	$\hat{\gamma}$	$\hat{\beta}$	$\chi^2_{x_0-3}$
5	23517	-0.3835	0.9795	47.80	0.027
10	26305	-0.3902	0.9883	84.50	2.022
15	27521	-0.3868	0.9854	67.44	3.752
20	28266	-0.3895	0.9872	77.19	8.822
30	29147	-0.3945	0.9899	98.19	16.873
40	29660	-0.3973	0.9912	112.22	30.217
100	30688	-0.3991	0.9918	121.64	86.536

Table 2: Maximum likelihood estimates calculated using Stata's `ml` program for maximum likelihood calculation, coupled with the user-written likelihood specification script `et.do`.

line "`version 15.1`", which ensures that should the script be run under later versions of Stata, it will be executed under the rules prevailing under version 15.1.

The results from running the Stata script are given in Table 2, which also includes a line for $x_0 = 100$, a set of values that we did not calculate in 1975.

The Stata-based results agree very well with those from the Excel spreadsheet, with values for $\hat{\gamma}$ being identical between the two, and values for $\hat{\alpha}$ differing by no more than one unit in the fourth decimal place. Stata also provides asymptotic standard errors for the parameters.

2.2 Data sources

One posssible reason for non-reproducibility is using a data set for the reproduction enterprise that is actually different from the data set used in the original investigation. The inability to reproduce the estimate of $\hat{\Delta}(1) = 11430$ from equation (2.5) by applying expression (2.4) to the data in Table 1 suggests that this might be the case.

The data on which the paper is based consist of the counts of the number of "word types[2]" n_x that appear in the Riverside edition of Shakespeare's complete works exactly x times. For instance, there are 14375 word types that appear on only one occasion in Shakespeare, and there are 4343 word types that appear on exactly two occasions. These word type frequencies are displayed in Table 1 of ET for the 30688 word types that appear 100 times or fewer. An additional 846 word types are said to appear more than 100 times each, but their exact frequencies are not supplied in the ET article.

The counts in Table 1 were taken from secondary sources, and reproducibility of the results would be enhanced if we could confirm at this remove that, for instance, there were

[2]A "word type" is a lexicographically distinct string of characters. Plural versions of singular nouns constitute distinct word types, as do different tenses of the same verb. Different meanings of the same word, e.g., "bear" as a verb and as a noun are not distinguished; they are the same word type.

no transcription errors in extracting data from the original sources.

The sources for these frequencies are "based on Spevack's (1968) concordance and on the summary appearing in an unpublished report by J. Gani and I. Saunders." An article apparently based on the report appeared in the 1976 volume of *Sankhyā* (Gani and Saunders, 1976). That article contains a table that exactly reproduces the first two lines of ET's Table 1, which suggests that both ET and Gani and Saunders were both based on a common original source. Because Gani and Saunders (1976) does not contain the counts for $x = 21$ through 100, we surmise that either those values were contained in the unpublished technical report referred to in ET but omitted for space reasons in the published version, or that there is a third source based on Spevack that contains those values.

Fortunately, Spevack's six-volume *Concordance* resides in the University of Chicago's Regenstein Library, which I consulted in the hopes that these questions could be resolved.

High frequency counts

Appendix A in volume 6 of Spevack lists all of the words in the Shakespearean corpus by frequency. Starting with the most frequently used words, the word "the" appears 27457. Because there is only one such word with that frequency, $x_{27457} = 1$[3]. Each observed frequency is listed, followed by a list of all of the words having that frequency. The first page of Spevack's Appendix A, together with a more typical page, are shown in Figure 5.

Figure 6 shows detail from the right-hand display in Figure 5, specifically the entries corresponding to the frequencies $x = 99$ and $x = 100$. The number of words shown, $x_{99} = 7$ and $x_{100} = 5$ agree with the corresponding entries in Table 1 of ET.

Efron and Thisted report

Reproducing Table 1 of Efron and Thisted

TO WRITE UP

Double counting

Although Spevack reports that the most common word in Shakespeare ("the") appears 27457 times, he also reports that it appears an additional 278 times in square brackets in the Evans edition of Shakespeare on which the Spevack *Concordance* is based. These words resulting from editorial emendation are counted as separate word types in the ET and Gani/Saunders compilation, and this is probably an error. The bracketed words should be counted as if they were used by Shakespeare, as they constitute the best editorial asssessment of what Shakespeare wrote. In that case, both x_{27457} and x_{278} are each too large by one, and x_{27735} should be incremented by one). In this case, the total number of

[3] As explained in section 2.2, x_{27457} should probably be taken to be zero, as the frequency count for "the" should be greater than 27457.

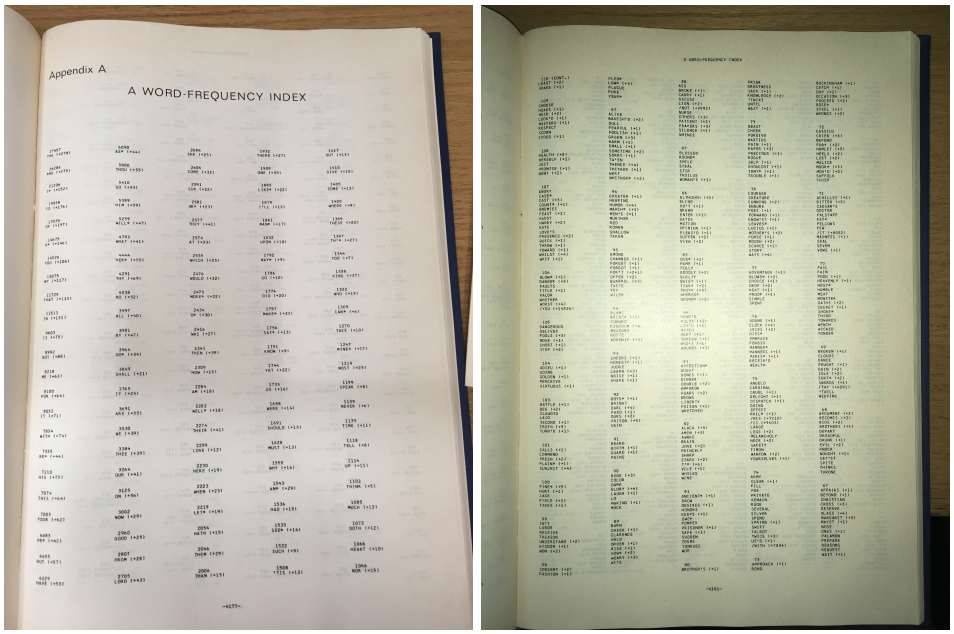


Figure 5: First page of Appendix A in Volume 6 of Spevack (1968, p. 4177), showing very high frequency words (left), and typical page (p. 4181) showing words with frequencies 67 through 109 (right).

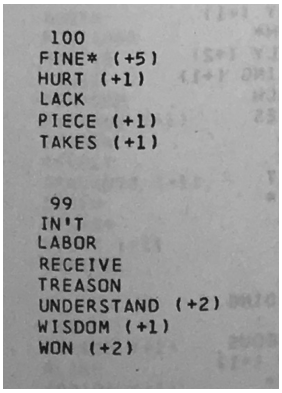


Figure 6: Detail from Appendix A in Volume 6 of Spevack (1968, p. 4181), showing words that appear 99 and 100 times in Shakespeare. The asterisk appearing next to the word `FINE` indicates that the word is a homograph, that is a word type that Shakespeare used with at least two different meanings in different settings. "Fine," for instance, can be a noun indicating a penalty, a verb meaning to purify, or an adjective meaning exquisite, among many other meanings. In any event, homographs have no affect on any of the analyses (or models) in ET.

unique words reported in ET is too large, as words that ever appear in editorial brackets are counted twice.

Figure 6 illustrates how the effect of double counting is potentially substantial. For instance, four of the five entries for $x = 100$ have parenthetical counts that correspond to the number of times the same word appears in editorial brackets. Thus, only one of the five words listed under $x = 100$ appeared exactly 100 times!

An alternative way to account for bracketed words would be to say that, because they are a result of editorial rather than Shakespearean action, that they should be disregarded entirely. In that case, x_{27457} would remain at one, but x_{278} should be decremented by one to account for disregarding "[the]".

In either case, the total number of word types reported by ET is be overstated. Indeed, Spevack reports that the total number of unique words in the Shakespearean corpus is 29066 (Spevack, vol 4, p. 1), and this figure is echoed by Gani and Saunders. Both are at odds with the total of 31534 reported in ET. This suggests that roughly 2500 words were counted twice.

Recalculating Table 3 and everything that follows from it to take this double counting into account would require a substantial amount of effort to identify, extract, and correct the myriad instances of this type of double counting, an effort which I may take on at a later point in time.

2.3 Hardware and operating system environments

The main calculations (for Table 3, for instance) were carried out on a Wang 2200B, a high-end business-oriented microcomputer. A dialect of BASIC, very close to the original language described by Kemeny and Kurtz, and Keller in 1964[4] was embedded in its microcode. It could only be programmed in BASIC. Programs had to be entered at the keyboard, but could be stored and retrieved using a cassette tape recorder. Programs could be stored for posterity (and future reproducibility studies), provided posterity was not expected to last longer than two or three years. s The programs written in PL/I (a general-purpose programming language of the day) and the ones written as input to MPS/360 were run on an IBM/360 mainframe computer. The days of using punched cards for such programs were waning, and it had just become possible to create, save, and edit text files interactively, and then to submit them as if they were stacks of punched cards for execution. I do not recall whether I carted boxes of punched computer cards with me when I left Stanford for Chicago in 1976, but I can say definitively that none remain.

One of the challenges for reproducible research is that programming languages and software systems (such as Stata, R, or MPS/360) can either become obsolete or evolve to

[4]In fact, the only available variable names were the single letters A–Z and a letter followed by a single digit. Array names had the same limitation, but the array and scalar namespaces were distinct. Thus, one could have both a scalar variable `W` and an array named `W`, leading to constructions like `MAT W=(1/W)*W`, a line that actually appears in the code used to generate Table 3.

the point of unrecognizability (and incompatibility). The hardware on which programs run also becomes obsolete or unavailable. And even were the same hardware still available, the underlying programs may no longer run under current versions of the hardware's operating system.

There is one additional aspect of computer hardware that can affect reproducibility, especially in the context of the Shakespeare example. In 1975, numerics were poorly understood by computer architects of the day. The first IEEE Standards for floating-point computation did not exist until 1985, ten years after our work. As a result, the numerical quality of computed results could be quite poor, and without there being any indicator that that was the case. So even if the calculations could be reproduced on the hardware of the day, there would still be good reason to check to see whether the results are different today!

2.4 Processes and choices

Even if the data sources underlying the computations are verified and all of the software, hardware, and operating systems are available and can be used, reproducibility is not guaranteed. The output of those programs, in our case, depended on initial values for parameters that we had to select and enter. Without a record of which parameter values were used to obtain the results reported in the paper, and without mathematical guarantees that the results are insensitive to starting values, we cannot be certain as to how the results were actually obtained. (This was a major motivation for Buckheit and Donoho (1995).)

3 A simple unbiased estimator

The maximum likelihood estimates for the Fisher negative binomial model, which serves as a benchmark for many of the results in ET, are presented in Table 3. They, and the calculations leading up to them, will be the focus for the rest of this paper.

The very first nontrivial calculation in ET is expression (2.5), which provides a simple unbiased estimate for $\hat{\Delta}(1)$, the expected number of new words to be seen in a newly-discovered corpus of work equal in size to the corpus of Shakespeare that we already have in hand. The expression reduces to

$$\hat{\Delta}(1) = n_1 - n_2 + n_3 - n_4 + \dots \,, \tag{1}$$

and expression (2.5) asserts that $\hat{\Delta}(1) = 11430$.

We report the first 100 values of n_x in our Table 1. As it turns out, however, the alternating sum of those values turns out to be 11486, a figure that is about half a percent larger that the one reported in the paper!

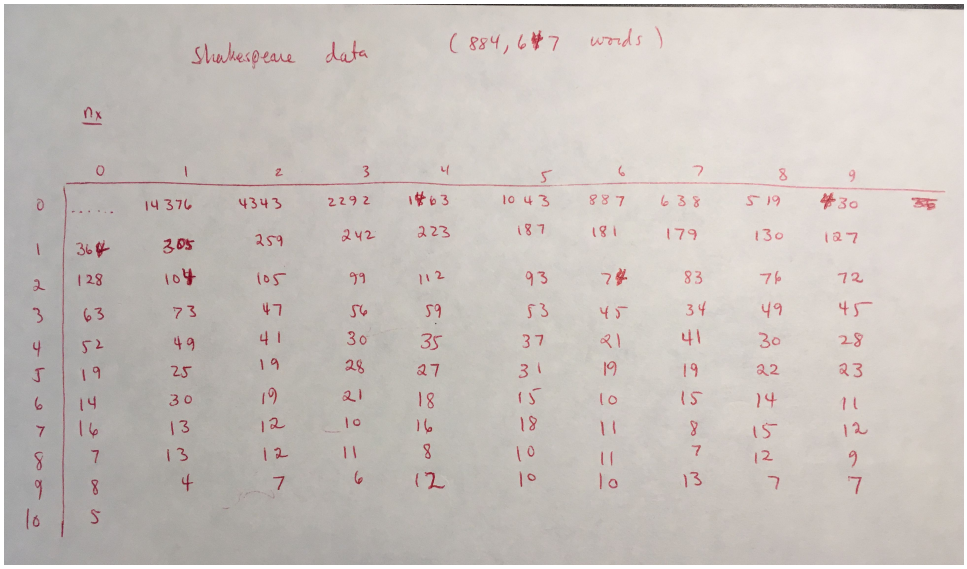


Figure 7: Probably an early version of Table 1 from ET, which differs in three locations from the from the published text (at $x = 6, 94, 97$).

What is going on? One possibility is that we used fewer than the 100 values in Table 1 in calculating (1) However, none of the partial sums

$$s_m = \sum_{x=1}^{m} (-1)^{x+1} n_x$$

equals 11430.

A second possibility is that we actually had *all* of the n_x values (up to $x = 27457$)in hand and that we used them all to obtain our estimate. However, if we append the frequency counts for $x > 100$ to those in Table 1, expression (1) turns out to be 11543.

A third possibility is that we calculated $\hat{\Delta}(1)$ using frequency counts from an early version of the dataset that contained errors, that we subsequently corrected the errors in Table 1, but we didn't recalculate $\hat{\Delta}(1)$. In fact, I discovered a hand-written version of Table 1 in my files, and three of the counts in that version differ from the counts reported in Table 1 of ET. The hand-written table is shown in Figure 7. The hand-written entries that differ are $n_6^* = 887$ (instead of 837), $n_{94}^* = 12$ (instead of 7), and $n_{97}^* = 13$ (instead of 15). When expression (1) is applied to the hand-written table, the resulting estimate is $\hat{\Delta}^*(1) = 11429$, almost—but not exactly—the value of 11430 reported in ET!

> *Lesson.* Archive data sets and version control them, and run all scripts only on the final version.

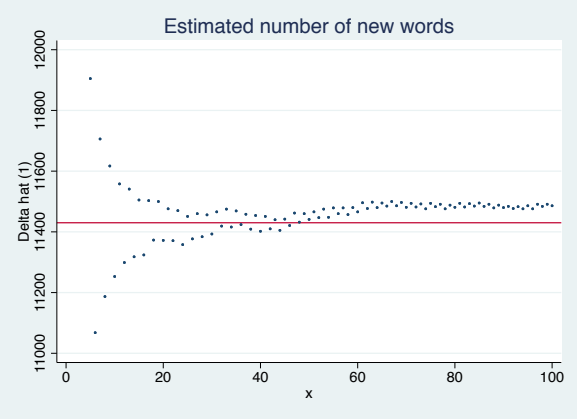


Figure 8: As x increases, $\hat{\Delta}_x(1)$ oscillates. The solid line is at $\hat{\Delta}(1) = 11430$, the figure cited in the text of ET. Values for $x < 5$ are omitted to show detail.

A fourth possibility is that the calculation was done by hand using a calculator rather than computer program. In that case, one or more frequencies could have been entered in error. For instance, if $x_{33} = 56$ were accidentally omitted from the calculation—an error easily done when doing the calculation by hand—then the end result for $\hat{\Delta}_{100}(1)$ would be exactly 11430.

A fifth possiblity also posits hand calculation. The person doing the calculation could have lost track of the alternating sign parity at some point. Specifically, if instead of subtracting n_{38} from the running sum, n_{38} were added instead, and then n_{39} subtracted, n_{40} added, and so forth, then the resulting calculation would again produce 11430 exactly.

Note, too, that as x_0 increases, the values of $\hat{\Delta}_x(1)$ oscillate (see Figure 8). This suggests that a smoothed version, say the average of $\hat{\Delta}_{99}(1)$ and $\hat{\Delta}_{100}(1)$ would be superior. So a fifth possibility is that we did some smoothing and forgot to mention it in the paper.

In fact, the values for $\hat{\Delta}_x(1)$ for x near $x_0 = 40$—a choice used repeatedly later in the paper—are reasonably close to 11430; the average of $\hat{\Delta}_{37}(1)$ and $\hat{\Delta}_{38}(1)$ is 11433.5, and the average of $\hat{\Delta}_{39}(1)$ and $\hat{\Delta}_{40}(1)$ is 11428.

In my view, the most likely explanation for the error in expression (2.5) of ET is that it arose in the process of hand calculation, with a parity error being the simplest explanation. The Stata script with the calculations carried out to support the explorations in this section is reproduced in Appendix E.

So in the end, if 11430 is actually an error and the correct number is 11486, what difference would it make to the rest of the ET paper? The answer is not much, except that the Fisher negative binomial estimate that we report (using $x_0 = 40$) is 11483, virtually

identical to the simple unbiased estimator based on $x_0 = 100$ alternating sums!

4 The Fisher negative binomial model

So let's move on to the heart of ET, our study of a negative binomial model for species counts advanced by Fisher. These are some notes that I wrote to the Chinese student whose query initiated this inquiry, pointing out some tips in dealing with the somewhat tricky maximum likelihood calculation.

The calculation is not, actually, a straightforward negative binomial problem. Rather, the problem is to maximize a multinomial likelihood where the probabilities are proportional to the parameterized negative binomial. (Equivalently, we are maximizing a conditional negative binomial likelihood, conditional on observing only $1 \le x \le x_0$). Norming these probabilities to sum to one introduces the trickiness.

Notes on Efron and Thisted (1976)
Ronald Thisted

The maximum likelihood calculation is tricky. Consider just the first row of Table 3, which only looks at the first five word types (those seen once, twice, up to five times). The likelihood is

$$L(p_1, p_2, \ldots, p_5) = p_1^{n_1} p_2^{n_2} p_3^{n_3} p_4^{n_4} p_5^{n_5}, \tag{1}$$

where

$$p_x = c \frac{\Gamma(x+\alpha)}{x!\Gamma(1+\alpha)} \gamma^{x-1}, \tag{2}$$

and where c is the constant that makes $\sum_1^5 p_x = 1$. Maximizing the likelihood with respect to α and γ is equivalent to maximizing the logarithm of the likelihood, which is considerably easy to work with. Notice that c^N can be factored out of the likelihood, so that the value of c doesn't matter, so long as in the end the estimated probabilities add up to one.

The log likelihood then looks like this:

$$\ell(p_1, p_2, \ldots, p_5) = \ell(\alpha, \gamma) = \sum_{x=1}^{5} n_x \ln(p_x), \tag{3}$$

where each p_x is written in terms of α and γ. Since

$$p_x = \left(\frac{x+\alpha}{x+1} \right) \gamma \cdot p_{x-1}, \tag{4}$$

for any given values for α and γ, one can take $p_1 = 1$, calculate p_2 through p_5 using the recursion stated above, calculate $c = \sum p_x$, and then divide each of the p_x values by c to make them sum to one (as they are probabilities). With the p_x values corresponding to α and γ, the next step is to calculate the log likelihood.

One approach to finding the maximum likelihood estimates for α and γ is by iterative search. Select starting values such as $\alpha = 0$ and $\gamma = 1$ for the two parameters. Without changing γ, calculate the log likelihood for different choices of α. When the log likelihood stops increasing, take smaller steps within the interval containing the maximum until you get adequate precision. Then fix α at this new value and then proceed to do the same thing with γ. When this step converges, hold γ at its new value and repeat the process with α. Alternate between the two parameters until you have adequate precision.

Unfortunately, this process converges very slowly to the maximum value, in part because the likelihood function is very flat near the maximum.

Other approaches, such as Fisher scoring or Newton-Raphson, can optimize for both α and γ at the same time, but they require derivatives of the log likelihood function. There are also methods that numerically approximate the derivatives. I believe that the "Solver" functions in Microsoft Excel use that approach.

Here is an approach to doing the calculations of the log-likelihood in Excel. First, create two cells that hold α and γ, (a) a column with x values (1 through 5, or 40), (b) a column of n_x values, (c) a column with values proportional to equation (2), calculated using the recursion of equation (4), a cell that sums the entries in that column, (d) a column that divides column (c) by the sum–these are the estimated probabilities $\hat{p}_x$, (e) a column with that row's contribution to the log likelihood ($n_x \ln(\hat{p}_x)$), and then the sum of column (e), which gives the log-likelihood. By changing the values for α and γ, the log likelihood is automatically updated.

There is an R package, `https://r-how.com/packages/preseqR`, that may do some (or all) of the negative binomial fitting to data such as these, although I haven't tried it.

5 Notes on additional computations

Discussion of issues related to Tables 4 thorugh 6, especially the linear programming software issue.

A BASIC MLE program

```
10 COM H(40),X(40),W(40),W8
20 MAT READ X
40 INPUT "X0,A,G,W8",X,A,G,W8:R1,R2=1
43 T=O:FOR I=1 TO X:T=T+X(I):NEXT I
45 SELECT PRINT 01D:PRINT "X0=",X:PRINT :GOTO 200
50 T1=1:W,W(1)=1
60 FOR I=2 TO X
70 T1=T1*(A+I-1)/I:W(I)=W(1)*T1*G^(I - 1)
80 W=W+W(I)
90 NEXT I
100 MAT W=(1/W)*W
130 MAT H=(T)*W:IF F9=0 THEN 140:RETURN
140 S1,W1,S2=0:FOR I=2 TO X:K=I-1:W1=W1+1/(A+K):S3=X(I)-T*W(I):S1=S1+S3*W1:S2=S2+S3*K:NEXT I
180 SELECT PRINT 005(64):PRINT I9,J9
190 RETURN
200 INPUT "EPS",E
210 INPUT "STARTING DELTA,RATIO A/D EFFECT",D,R2:R1=1/(1+R2)
220 GOSUB 50:S5=S1:S4=S2
230 C1=W8*ABS(S1)+(1-W8)*ABS(S2):SELECT PRINT 01D:GOSUB 3000
240 IF C1<E THEN 2000:A1=A:G1=G
250 FOR I9=-1 TO 1
260 FOR J9=-1 TO 1
270 A=A1+I9*D*R1
280 G=G1+J9*D*R2
290 GOSUB 50
300 C=W8*ABS(S1)+(1-W8)*ABS(S2)
310 IF C>C1 THEN 330
315 S5=S1:S4=S2
320 A2=A:G2=G:C1=C
330 NEXT J9
340 NEXT I9
350 C3=ABS(A1-A2)+ABS(G1-G2)
360 IF C3=0 THEN 400
370 A=A2:  G=G2
380 SELECT PRINT 01D:GOSUB 3000
385 R1=ABS(S5)/(ABS(S5)+ABS(S4)):R2=1-R1
390 GOTO 240
400 D=D/2:IF D<1E-7 THEN 450 :REM REDUCE DELTA
410 SELECT PRINT 01D:PRINT "DELTA REDUCED TO";D
415 INPUT "NEW R2",R2:R1=1-R2
420 GOTO 250
450 PRINT "HALT DUE TO SMALL DELTA":GOTO 40
1000 DATA 14376,4343,2292,1463,1043,837,638,519,430,364,305,259,242,223,187,181,179,130,127,128
1010 DATA 104,105,99,112,93,74,83,76,72,63,73,47,56,59,53,45,34,49,45,52
2000 PRINT :PRINT :PRINT "* * * * FINAL ANSWER * * * *":PRINT
2010 PRINT :SELECT PRINT 01D:A=A1:G=G1:GOSUB 3000:GOTO 40
3000 PRINT :PRINT "ALPHA=",A,"GAMMA=",G
3010 PRINT :PRINT "W=",W
```

```
3020 PRINT "H(1)=",H(1)
3030 PRINT "S1=",S5:PRINT "S2=",S4:PRINT "C=",C1
3040 GOSUB 6000:RETURN
6000 INPUT "X0,A,G",X,A,G:T=0:FOR I=1 TO X:T=T+X(I):NEXT I:F9=1:GOSUB 50:F9=0
6010 SELECT PRINT 01D
6020 PRINT "X0=",X;" ALPHA=",A;" GAMMA=",G
6030 U=0
6040 FOR I=1 TO X:U=U+X(I)*(LOG(X(I)/T)-LOG(W(I))):NEXT I
6050 PRINT "LIK-RATIO-STAT=",2*U
6060 PRINT
6070 RETURN
```

B Stata negative binomial log likelihood

```
program define et
    // This is the beta parameterized likelihood
// Assumption: the index x for each frequency count is stored in a variable
//             in the dataset named -x-
//              (this is because I couldn't figure out how to pass it to
//              the maximizer as a parameter)
    version 15.1                                  // Stata version
args todo b lnf g negH                       // standard Stata ML parameters

// set up local variable names for use here
tempname alpha beta gamma psnormalizer nsum
tempvar lnpxstar pxstar pstarsum nxsum llcomponent llsum nxtemp

// obtain current estimates for alpha and beta from the passed parameters
mleval `alpha' = `b', scalar eq(1)
mleval `beta'  = `b', scalar eq(2)

// infer the value of gamma
scalar `gamma' = `beta'/(1+`beta')

// calculate unnormalized log likelihood contribution from counts of
// words with frequency x
// All calculations are done 'quietly' ("qui") to suppress  results of
//     intermediate calculations during iterations
qui gen double `lnpxstar' = lngamma(x+`alpha') ///
                          +(x-1)*ln(`gamma') ///
                          - lnfactorial(x)   ///
                          -lngamma(1+`alpha') if $ML_samp==1
// calculate the norming constant to make probabilities sum to one
qui gen double `pxstar'   = exp(`lnpxstar') if $ML_samp==1
qui gen double `pstarsum' = sum(`pxstar')
scalar `psnormalizer' = `pstarsum'[_N]

// zero out any counts for x>x0, and calculate sum of counts
qui gen double `nxtemp' = $ML_y1*$ML_samp
qui gen double `nxsum'  = sum(`nxtemp')
scalar `nsum' = `nxsum'[_N]

// finally calculate the actual log likelihood contribution of each row,
//     add them up, and return them, normalized appropriately
qui gen double `llcomponent' = $ML_y1*`lnpxstar'*$ML_samp
qui gen double `llsum' = sum(`llcomponent')
scalar `lnf' = `llsum'[_N]-ln(`psnormalizer')*`nsum'
end
```

C Stata Table 3 reproduction

```
version 15.1
set more off
set seed 99574774

clear

// load the maximum likelihood defining program
// (first deleting any version that already exists)
capture program drop et
run et

// read in the data, generate index numbers -x-,
//      and calculate running sums of frequency counts
infile nx using ../Raw/Table1-list.txt, clear
gen x=_n
gen nxsum=sum(nx)

// This uses the beta-parameterized likelihood for the computations
// "qui" is an abbreviation for 'quietly', which suppresses output
// "noi" is an abbreviation for 'noisily', which forces output

quietly {
noisily di _newline " x0 sum nx    alpha    gamma    beta   chisq"

foreach x0 in 5 10 15 20 30 40 100 {
   // first obtain mle's for NB model based on first x0 word counts
   qui ml model d0 et (nx=) () in 1/`x0'
   qui ml init eq1:_cons=-0.4 eq2:_cons=.9
   qui ml maximize, nolog

   // calculate saturated log likelihood for chi-squared computation
   // here, e(ll) is the log likelihood returned from the MLE calculation
   qui gen double satll= nx*ln(nx)
   qui gen double sumsatll = sum(satll)
   scalar SLL = sumsatll[`x0']-nxsum[`x0']*ln(nxsum[`x0'])
   drop satll sumsatll
   scalar chisq = -2*(e(ll)-SLL)

   // finally calculate the value for gamma-hat from the estimate for beta-hat
   scalar gamma = _b[#2:_cons]/(1+_b[#2:_cons])

   // ... and display the results for comparison to Table 3 of ET
   noi display %3.0f `x0' %7.0f nxsum[`x0'] %9.4f _b[_cons] ///
               %9.4f gamma %8.2f _b[#2:_cons] %8.3f chisq
}
}
```

D Example of Stata `ml` output

Stata's `ml` command is a general-purpose optimizer that is designed to provide standard inferential tools in a standardized format as a by-product. The program scripts `et.do` and `et1.do` provide the code needed to calculate the log likelihood for the conditional negative binomial model considered in Section 3 of Efron and Thisted (1976), the former parameterized using β and the latter parameterized using γ.

The tables in section 2.1 were produced by a script that suppressed the standard output from each individual estimation command, instead compiling and printing only the tabular results corresponding to Table 3 of Efron and Thisted. To illustrate the utility of the approach using Stata, the output below shows estimates of α, β, and γ, together with their standard errors and confidence intervals.

Here are the results for $x_0 = 40$, the setting used for several purposes in ET. Lines that start with a period show inputs to Stata. In this example we explicitly provide initial values to the `ml` maximizer, but using Stata's defaults produce virtually the same result, differing only by one unit in the sixth decimal place of $\hat{\alpha}$. The estimate for α is $\hat{\alpha} = -0.3973$, and its standard error is 0.006. The two estimates are associated in the output below with `eq1:_cons` (α) and `eq2:_cons` (β).

```
. ml model d0 et (nx=) () in 1/40
. ml init eq1:_cons=-0.4 eq2:_cons=50
. ml maximize, nolog

                                                Number of obs     =         40
                                                Wald chi2(0)      =          .
Log likelihood = -61615.249                     Prob > chi2       =          .

------------------------------------------------------------------------------
          nx |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
eq1          |
       _cons |  -.3973074   .0059839   -66.40   0.000    -.4090355   -.3855792
-------------+----------------------------------------------------------------
eq2          |
       _cons |   112.2214   18.51752     6.06   0.000     75.92778    148.5151
------------------------------------------------------------------------------

. scalar gamma = _b[#2:_cons]/(1+_b[#2:_cons])
. display "Delta-hat_ml(1)= " %8.0f -nx[1]*((1+gamma)^(-_b[_cons]) -1)/(_b[_cons]*gamma) "  Gamma= " %7.5f gamm
Delta-hat_ml(1)=    11490  Gamma= 0.99117

. // get standard error for gamma hat and for Delta-hat(1)
. nlcom _b[#2:_cons]/(1+_b[#2:_cons])

       _nl_1:  _b[#2:_cons]/(1+_b[#2:_cons])

------------------------------------------------------------------------------
          nx |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       _nl_1 |   .9911678   .0014445   686.15   0.000     .9883365     .993999
------------------------------------------------------------------------------
```

Usefully, Stata can not only calculate $\hat{\Delta}(1)$, but can also obtain its standard error via the delta method.

```
. nlcom -nx[1]*((1+gamma)^(-_b[_cons]) -1)/(_b[_cons]*gamma)

      _nl_1:  -nx[1]*((1+gamma)^(-_b[_cons]) -1)/(_b[_cons]*gamma)

------------------------------------------------------------------------------
          nx |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       _nl_1 |   11489.66   24.75408   464.15   0.000     11441.15    11538.18
------------------------------------------------------------------------------
```

By repeating the calculation parameterizing using γ ($= \beta/(1+\beta)$) instead of β, we obtain standard errors for $\hat{\gamma}$ as well. Note that the log likelihood is the same for both parametrizatoins. Because γ is constrained to be strictly less than one (by its relationship to β), and because our estimates for γ are near that upper boundary, we would expect the computation based on β (which is unconstrained) to be somewhat more stable numerically, and for that reason our computations in the text are based on the β parameterization.

```
. // repeat using gamma parameterization
. ml model d0 et1 (nx=) () in 1/40

. ml maximize, nolog
initial:       log likelihood =     -<inf>  (could not be evaluated)
feasible:      log likelihood = -96277.713
rescale:       log likelihood = -96277.713
rescale eq:    log likelihood = -85253.513

                                                Number of obs     =         40
                                                Wald chi2(0)      =          .
Log likelihood = -61615.249                     Prob > chi2       =          .

------------------------------------------------------------------------------
          nx |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
eq1          |
       _cons |   -.397306   .0059838   -66.40   0.000     -.409034   -.3855779
-------------+----------------------------------------------------------------
eq2          |
       _cons |   .9911673   .0014445   686.15   0.000     .9883361    .9939986
------------------------------------------------------------------------------
```

E Simple unbiased estimate for $\Delta(1)$

```
// explore possibilities for the miscalculation of Delta-hat(1)
// in expression (2.5) of ET
version 15.1
set more off
set seed 995874121

// load the ET data table
use ../Data/table1, clear
gen nsigned = n*(-1)^(x+1)
// total contains the running alternating sum of frequencies to give (2.5)
// The last entry in the column gives the answer, which ET reports as 11430.
gen total = sum(nsigned)
di total[_N]

// calculate the total number of words in Table 1 of ET
gen foo=sum(n)
di foo[100]

// and add in the 846 words with frequencies x>100
di foo[100]+846

twoway scatter total x if x>4, yline(11430) msize(tiny) ///
   yti("Delta hat (1)") xti("x") ti("Estimated number of new words") ///
   name(delta1hat, replace)
graph export ../Figures/delta1hat.pdf, replace

// generate smoothed version by doing running means of 2
gen ma = (total[_n-1]+total[_n])/2
two scatter ma  x if x>4, yline(11430) msize(tiny) ///
   yti("Delta hat (1)") xti("x") ti("Smoothed estimated number of new words") ///
   name(delta1hatma, replace)
 graph export ../Figures/delta1hatma.pdf, replace

 // what if x_{33} was accidentally skipped?
replace nsigned=0 in 33
gen bar=sum(nsigned)
replace nsigned=56 in 33

di bar[100]

// here we reload the ET data table
use ../Data/table1, clear
rename n nx
count

list in l
gen nxsum=sum(nx)
```

```
gen sign = (-1)^(x+1)
gen signnx = sign*nx
gen sumalt = sum(signnx)
gsort - x
gen revsumalt = sum(signnx)
sort x

list in 1/5
list in -5/l

// this explores the possibility of an inadvertent parity error between
// x=37 and x=38, that gets propagated

gen errsign = (-1)^(x+1+(x>=38))
gen errsignnx = errsign*nx
gen errsumalt = sum(errsignnx)
list in l

// repeat for the hand-archived table

// here we load the hand-written frequency counts from Thisted's archives
use ../Data/table1-hand-archived, clear
rename n nx
count

list in l
gen nxsum=sum(nx)

gen sign = (-1)^(x+1)
gen signnx = sign*nx
gen sumalt = sum(signnx)
gsort - x
gen revsumalt = sum(signnx)
sort x

list in 1/5
list in -5/l

// and finally, we do the calculation on the first 100 actual Spevack counts
infile x nx using ../Raw/lowFreqSpevack.txt, clear
count
list in l

gen nxsum=sum(nx)

gen sign = (-1)^(x+1)
gen signnx = sign*nx
gen sumalt = sum(signnx)
gsort - x
gen revsumalt = sum(signnx)
```

```
sort x

list in 1/5
list in -5/l
```

Bibliography

Jonathan B. Buckheit and David L. Donoho.
WaveLab and reproducible research.
In Anestis Antoniadis and George Oppenheim, editors, *Wavelets and Statistics*, Lecture Notes in Statistics 103, pages 55–81.
Springer-Verlag, 1995.

Bradley Efron and Ronald A. Thisted.
Estimating the number of unseen species: How many ords did Shakespeare know?
Biometrika, 63(3):435–447, 1976.

J. Gani and I. Saunders.
Some vocabulary studies of literary texts.
Sankhyā: The Indian Journal of Statistics, Series AB (1960–2002), 38(2):101–111, 1976.

Donald Ervin Knuth.
Literate programming.
The Computer Journal, 27(2):97–111, 1984.

Marvin Spevack.
A Complete and Systematic Concordance to the Works of Shakespeare, volume 1–6.
George Olms, Hildesheim, 1968.

StataCorp.
Stata Statistical Software: Release 15.
StataCorp LLC, College Station, TX, 2017.