Published in final edited form as:

Am Stat. 2018 ; 72(2): 199–205. doi:10.1080/00031305.2017.1392361.

Facilitating the Calculation of the Efficient Score Using Symbolic Computing

Alexander Sibley^{*,†},

Duke Cancer Institute, Duke University Medical Center

Zhiguo Li^{*,†,‡},

Biostatistics and Bioinformatics, Duke University School of Medicine

Yu Jiang^{*,†},

Biostatistics and Bioinformatics, Duke University School of Medicine

Yi-Ju Li^{*},

Biostatistics and Bioinformatics, Duke University School of Medicine

Cliburn Chan^{*,†},

Biostatistics and Bioinformatics, Duke University School of Medicine

Andrew Allen^{*,†,‡}, and

Biostatistics and Bioinformatics, Duke University School of Medicine

Kouros Owzar^{*,†,‡}

Biostatistics and Bioinformatics, Duke University School of Medicine Duke Cancer Institute, Duke University Medical Center

Abstract

The score statistic continues to be a fundamental tool for statistical inference. In the analysis of data from high-throughput genomic assays, inference on the basis of the score usually enjoys greater stability, considerably higher computational efficiency, and lends itself more readily to the use of resampling methods than the asymptotically equivalent Wald or likelihood ratio tests. The score function often depends on a set of unknown nuisance parameters which have to be replaced by estimators, but can be improved by calculating the efficient score, which accounts for the variability induced by estimating these parameters. Manual derivation of the efficient score is tedious and error-prone, so we illustrate using computer algebra to facilitate this derivation. We

*The authors gratefully acknowledge the Associate Editor and three reviewers, whose comments and suggestions helped to substantially improve and clarify this manuscript.

[†]A preliminary version of this paper was submitted to the proceedings of the 2014 Joint Statistical Meetings (Sibley et al. 2014).

[‡]This research was supported in part by Award number P01CA142538 from the National Cancer Institute (ZL, AA, and KO). The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institute of Health.

SUPPLEMENTARY MATERIAL

SymbCalcCode: Jupyter notebook of the SymPy code used in this report. (.ipynb file)

SymbCalcCode: PDF version of the Jupyter notebook. (.pdf file)

SymbCalcCode: Sage worksheet version of the SymPy code used in this report. (.sws file)

GaussianExample: Jupyter notebook outlining another example application of the use of symbolic processing in the derivation of the efficient score, including more detailed descriptions of SymPy and its functions. (.ipynb file)

GaussianExample: PDF version of the Jupyter notebook. (.pdf file)

demonstrate this process within the context of a standard example from genetic association analyses, though the techniques shown here could be applied to any derivation, and have a place in the toolbox of any modern statistician. We further show how the resulting symbolic expressions can be readily ported to compiled languages, to develop fast numerical algorithms for high-throughput genomic analysis. We conclude by considering extensions of this approach. The code featured in this report is available online as part of the supplementary material.

Keywords

computer algebra; nuisance parameters; mathematical statistics; python; trio data; genome-wide association study

1 Introduction

The Wald, likelihood ratio, and score tests compose the three primary non-Bayesian methods of inference for data from high-throughput genomic assays (Sorensen & Gianola 2013). The score statistic (Rao 1948) is more stable, and can be employed where the variability is difficult to estimate, whereas the likelihood ratio test requires an estimation of variability. When conducting inference on the basis of the score, nuisance parameters are optimized only once, and the parameters of interest never have to be optimized, giving the score higher computational efficiency. To properly account for multiple testing in the context of high-dimensional genomics problems, it is necessary to approximate the joint sampling distribution of a vector of statistics. This can be readily estimated for a vector of these scores using resampling or numerical integration methods (Lin 2005, Seaman & Müller-Myhsok 2005, Conneely & Boehnke 2007). All of this is accomplished while remaining asymptotically equivalent to the Wald and likelihood ratio tests (Hall & Mathiason 1990). And while all three methods suffer from variability induced by the estimation of the nuisance parameters, the score test can account for this by employing the efficient score. The cost of this correction is additional algebraic derivation, required prior to implementation. In this report, we show how computer algebra systems can relieve us of this computational burden, while offering additional benefits.

While the example in this report pertains to genetic association analyses, practitioners in any field may find it useful to have at their disposal an implementation of the efficient score test unencumbered by laborious manual derivations. More generally, some of the techniques shown here could be applied to any derivation, and may therefore be elucidating for any statistician unfamiliar with symbolic computing software. For example, the iterative structure of a Taylor series expansion makes it an ideal candidate for symbolic processing. Additionally, the application of symbolic computing software, especially packages which, as we will later discuss, allow symbolic equations to be directly translated into usable code, is in line with the 2015 “ASA Guidelines for Undergraduate Programs in Statistical Science” (Horton & Hardin 2015), which emphasized heavily the need to expand the student curriculum to incorporate higher-level programming languages and statistical analysis software. In particular, courses in statistical computing, data science, and modern inference could benefit greatly from integrating computation and theory.

1.1 The Efficient Score

The efficient score test is applicable to a wide variety of data types and models. In statistical genetics it is often used to test for associations of genotype, as measured by high-throughput genomic assays, with different types of clinical outcomes, including binary traits (Ekström Smedby et al. 2006, Freedman et al. 2009), quantitative measures (Zhang et al. 2014, Pang et al. 2015), and censored time-to-event outcomes (Gaudet et al. 2010, Siamakpour-Reihani et al. 2015). The models can also be expanded to include covariates that could potentially affect the outcome.

We begin by establishing a generalized notation for the efficient score. Let $\boldsymbol{\theta}$ be the vector of parameters, which we parse into $\boldsymbol{\beta}$, our parameters of interest, and $\boldsymbol{\eta}$, the nuisance parameters. We denote $\ell(\boldsymbol{\theta}|\mathbf{Z})$ to be our log-likelihood function of $\boldsymbol{\theta}$, given our random variables, $\mathbf{Z}$. We define $S_t(\boldsymbol{\theta} | \mathbf{Z}) = \frac{\partial}{\partial t} \ell(\boldsymbol{\theta} | \mathbf{Z})$, where t can be $\boldsymbol{\beta}$, $\boldsymbol{\eta}$ or $\boldsymbol{\theta}$ (Magnus & Neudecker 2007). We calculate the efficient score (Tsiatis 2006) as:

$$S_{\text{eff}}(\boldsymbol{\theta} | \mathbf{Z}) = S_{\boldsymbol{\beta}}(\boldsymbol{\theta} | \mathbf{Z}) - I_{\boldsymbol{\beta}\boldsymbol{\eta}}(\boldsymbol{\theta} | \mathbf{Z}) \{I_{\boldsymbol{\eta}\boldsymbol{\eta}}(\boldsymbol{\theta} | \mathbf{Z})\}^{-1} S_{\boldsymbol{\eta}}(\boldsymbol{\theta} | \mathbf{Z}) \quad (1)$$

where

$$I_{\boldsymbol{\beta}\boldsymbol{\eta}}(\boldsymbol{\theta} | \mathbf{Z}) = \mathbb{E} \left[S_{\boldsymbol{\beta}}(\boldsymbol{\theta} | \mathbf{Z}) S_{\boldsymbol{\eta}}^T(\boldsymbol{\theta} | \mathbf{Z}) \right], \text{ i.e., the covariance matrix, and } \quad (2)$$

$$I_{\boldsymbol{\eta}\boldsymbol{\eta}}(\boldsymbol{\theta} | \mathbf{Z}) = \mathbb{E} \left[S_{\boldsymbol{\eta}}(\boldsymbol{\theta} | \mathbf{Z}) S_{\boldsymbol{\eta}}^T(\boldsymbol{\theta} | \mathbf{Z}) \right], \text{ i.e., the variance matrix. } \quad (3)$$

The final efficient score statistic is comprised of the sum of (1) over all observations in the data.

As defined, (2) and (3) are sub-matrices of the Fisher information matrix, $I(\boldsymbol{\theta}) = \mathbb{E} \left[S_{\boldsymbol{\theta}}(\boldsymbol{\theta} | \mathbf{Z}) S_{\boldsymbol{\theta}}^T(\boldsymbol{\theta} | \mathbf{Z}) \right]$, and require calculation of the expected values of the vector products. The use of expected values guarantees convergence to the true variance and covariance matrices of the statistic, but determining these values is often intractable. An alternative approach is to use instead the empirical estimators of the expectations. These values are readily derived by averaging the observed values from the data.

Recall that for a log-likelihood function ℓ with parameter $\boldsymbol{\theta}$, under certain regularity conditions, $\mathbb{E} \left[\frac{\partial \ell}{\partial \boldsymbol{\theta}} \frac{\partial \ell}{\partial \boldsymbol{\theta}^T} \right] = - \mathbb{E} \left[\frac{\partial^2 \ell}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T} \right]$. Define $\mathcal{J}(\boldsymbol{\theta} | z_i) = - \frac{\partial^2}{\partial \boldsymbol{\theta}^2} \ell(\boldsymbol{\theta} | z_i)$, i.e., the observed information matrix. Notice $\mathbb{E} [\mathcal{J}(\boldsymbol{\theta} | z_i)] = I(\boldsymbol{\theta})$. Further, define the average observed information as $\mathcal{J}^{(n)}(\boldsymbol{\theta} | \mathbf{z}) = \frac{1}{n} \sum_{i=1}^n \mathcal{J}(\boldsymbol{\theta} | z_i)$, and note that, by the WLLN, $\mathcal{J}^{(n)}(\boldsymbol{\theta} | \mathbf{z}) \xrightarrow{P} I(\boldsymbol{\theta})$ (Freedman 2007).

The covariance and variance matrices in (2) and (3) would instead be based on the average observed information, so that

$$\mathcal{S}_{\text{eff}}(\boldsymbol{\theta} | \mathbf{z}) = S_{\beta}(\boldsymbol{\theta} | \mathbf{z}) - \mathcal{F}_{\beta\eta}^{(n)}(\boldsymbol{\theta} | \mathbf{z}) \{ \mathcal{F}_{\eta\eta}^{(n)}(\boldsymbol{\theta} | \mathbf{z}) \}^{-1} S_{\eta}(\boldsymbol{\theta} | \mathbf{z}) \quad (4)$$

While usually simpler to obtain, this method may generate negative variance estimates, causing inconsistency in the score test.

Both methods described here are evaluated under the null hypothesis. Note that Freedman (2007) also describes a third approach, specifically, using the MLE for the observed information unconstrained by the null hypothesis. This approach avoids both the potential inconsistency of using the constrained average observed information, and the need for computing the expected values of the Fisher information, and therefore may be attractive in some applications, especially when testing only a small number of associations. However, the example provided here is set in the context of high-throughput genomic analysis, which can involve testing millions of associations in a single study. Using the unconstrained MLE would necessitate recomputing the observed information for each genotypic marker, adding significant computational overhead. For this reason, we limit further discussion to only the first two approaches.

Deriving either version of the efficient score therefore requires taking multiple derivatives, computing summations, carrying out matrix multiplication and inversion, and if the Fisher information is used, taking expected values, compounding the initial complexity of the likelihood function with each step. Such manipulations afford many opportunities for errors if the derivations are carried out manually. Symbolic computing offers a more reliable alternative.

1.2 Symbolic Computation

Neither symbolic computing nor symbolic computing software are new tools (Caviness 1986). Macsyma, one of the oldest computer algebra systems, was developed at MIT in the 1960's (Moses 2012). Its descendant, Maxima (Maxima 2016), continues to be actively maintained today. Andrews & Stafford (2000) extensively described symbolic algorithms for a wide variety of statistical methodologies, from densities of random variables, to asymptotic expansions, to simplification of likelihood quantities. Software implementations of such algorithms allow computer algebra systems to do the algebraic derivation and manipulation of equations formerly confined to white boards or pencil and paper. These systems are able to carry out, to varying degrees, assorted operations used in algebra, linear algebra, and statistical derivations. The transition from analog to digital offers the same benefits to algebra as it did for arithmetic. Namely, the steps can be performed faster, repeatably, and more reliably by a computer than by a person.

Symbolic computing therefore offers a convenient means of addressing the main drawback of using the efficient score, i.e., the sometimes arduous derivation of the score function. Once the likelihood function has been transcribed to the symbolic software language, the

rest of the derivation is implemented programmatically. This makes symbolic computation less vulnerable to error than manual derivations. Additionally, since the steps of the derivation procedure do not change from implementation to implementation, much of the code for symbolic manipulations generated for one application can be reused for another. The transition from single instance to reusable pipeline is trivial. Furthermore, because the equations exist digitally, various options exist for transferring the formulae to either a markup language for presentation purposes, or to functional analytical code. This makes symbolic processing software useful for rapid prototyping, as it is easy to convert algebraic equations and formulae into code for simulation and testing. Further, whether in prototyping or development, the effects of any changes to the initial specifications can easily be propagated through the entire workflow.

Whatever the application, users have a variety of symbolic processors from which to choose. Some, such as Maple (Maplesoft, a division of Waterloo Maple Inc. 2014) and Mathematica (Wolfram Research, Inc. 2012), are proprietary, (though may offer free or reduced price versions for students). Others, such as SageMath (The Sage Developers 2017), are freely available via general public license. The aim of SageMath is to combine a variety of open-source software packages into a single common interface. One of the packages it is built on is SymPy (SymPy Development Team 2014), an open-source module which allows symbolic processing in Python. For the purposes of this report, we chose to use SymPy because it is both freely available, and the code and results can be directly integrated with L^AT_EX via PythonTEX (Poore 2013). The code used here is also included as part of the Supplementary Material of this report, in both a Jupyter notebook and a Sage worksheet.

The aim of this report is to highlight the benefits of symbolic computing software by application to a practical example. The software packages needed to execute the code presented here have been packaged by the Anaconda (Continuum Analytics 2016) distribution available for the Linux, OSX, and Windows operating systems. In the example below, executed Python code is enclosed in solid frames, immediately followed by the PythonTEX-formatted results.

2 An Example

We use a standard example from the field of statistical genetics to demonstrate the calculation of the efficient score using symbolic computing software: testing the association of genotype with a continuous phenotype in high-throughput genomic assays, in the context of family-based trio data (Jiang et al. 2017). In practice, a variety of phenotypes can be tested, such as body mass index, years of exposure to cigarette smoke, or gene or protein expression level. The term “trio data” refers to the fact that genotype data is collected not only for the proband, but for their mother and father as well. The genetic loci being tested are single nucleotide polymorphisms (SNPs). For each SNP, an individual’s genotype is either *RR*, *AR*, or *AA*, i.e., they either have two copies of the reference (normal) allele, one alternate (mutant) allele and one reference allele, or two alternate alleles. For their paper, Jiang et al. used simulated data generated using COSI (Schaffner et al. 2005).

Our model is based on Y_i , the phenotype of interest of the offspring of the i th family, and G_i , the genotype of the i th offspring at the locus being interrogated, where $G_{M,i}$ and $G_{F,i}$ are the corresponding parental genotypes, and $G \in \{0, 1, 2\}$ is a count of the number of alternate alleles. We then separate the genotype into between- and within-family components (Abecasis et al. 2000) by defining $G_{bi} = (G_{M,i} + G_{F,i})/2$, and $G_{wi} = G_i - G_{bi}$. This separation offers the advantage that the within-family component, G_{wi} , is robust against confounding due to population stratification (Fulker et al. 1999). Finally, we define $\mathbf{X}_i$, a vector of p cofactors (possibly including an intercept) also included in the model. The cofactors are used to further control for confounding, and can be of varying types including binary, e.g., gender or treatment group, or continuous, such as age or drug dosage.

We suppose that, for all offspring, the conditional distribution of Y given $G_b = g_b, G_w = g_w$ and $\mathbf{X} = \mathbf{x}$ is normal, with mean $\mathbf{a}^T \mathbf{x}_i + b_b g_{b,i} + b_w g_{w,i}$ and variance σ^2 , i.e.,

$$(Y_i | G_{bi} = g_b, G_{wi} = g_w, \mathbf{X}_i = \mathbf{x}) \sim N(a_0 + a_1 x_1 + a_2 x_2 + b_b g_b + b_w g_w, \sigma^2).$$

So, under the efficient score notation, $\mathbf{Z}_i = (Y_i, \mathbf{X}_i, G_{bi}, G_{wi})^T$ and $\boldsymbol{\theta} = (a_0, a_1, a_2, b_b, b_w, \sigma)^T$. Note that, for the purposes of this example, we are restricting the model to two cofactors and an intercept, though a more complex model could easily be implemented.

Under this model, the null hypothesis, H_0 , is that there is no genetic effect on the phenotype, i.e., $b_b = b_w = 0$, so our parameters of interest are $\boldsymbol{\beta} = (b_b, b_w)^T$. However, finding the likelihood of $\boldsymbol{\beta}$ requires estimating the nuisance parameters, $\boldsymbol{\eta} = (a_0, a_1, a_2, \sigma)^T$, which may induce variability, thus necessitating the use of the efficient score.

This model, and the corresponding null hypothesis and statistical test, apply to a single genetic locus. Since our example is set within the context of a high-throughput genomic assay, the derivation of the efficient score that follows would be used to test not just one, but thousands or even millions of loci in the genotyping data. For an example application of the efficient score to a simpler statistical model, see the `GaussianExample.ipynb` Jupyter notebook included in the Supplementary Material of this report.

2.1 Initial Specifications

We begin by loading the SymPy module and the tools necessary for our derivation.

```
from sympy import *
from sympy.printing import *
from sympy.stats import *
```

The first step in using SymPy is to declare the expressions which will represent algebraic variables.

```
a0,a1,a2,b_b,b_w = symbols("a0 a1 a2 b_b b_w", real=True)
y, x1,x2,g_b,g_w = symbols("y x1 x2 g_b g_w", real=True)
sigma = symbols("sigma", positive=True)
```

We then define our two sets of parameters.

```
beta = [b_b, b_w]
eta = [a0, a1, a2] + [sigma]
```

Next, we use SymPy's built-in probability distributions to initialize the conditional distribution of $Y/G_b, G_w, X$.

```
distY = Normal("distY",
               a0 + a1*x1 + a2*x2 + b_b*g_b + b_w*g_w, sigma)
print(latex(density(distY)(y)))
```

$$\frac{\sqrt{2}}{2\sqrt{\pi}\sigma} e^{-\frac{1}{2\sigma^2}(-a_0 - a_1x_1 - a_2x_2 - b_bg_b - b_wg_w + y)^2}$$

Printing the density, we see it has been properly defined. Note that the equation above was constructed by SymPy and automatically rendered for this report by PythonTEX. No L_ATEX coding was required to display it.

The basis of the efficient score is the log-likelihood function for Y_i , which we get from the above formula.

```
likelihood = simplify(density(distY)(y))
logLikelihood = simplify(log(likelihood))
print(latex(logLikelihood))
```

$$-\log(\sigma) - \frac{1}{2}\log(\pi) - \frac{1}{2}\log(2) - \frac{1}{2\sigma^2}(a_0 + a_1x_1 + a_2x_2 + b_bg_b + b_wg_w - y)^2$$

Notice that the `simplify()` function has factored a negative one out of the final squared term. This equation represents the contribution to the log-likelihood for one observation, $z_i = (y_i, g_{bi}, g_{wi}, x_i)$, while the total log-likelihood is given by summing over all families.

2.2 Derivatives

To get the efficient score, we take the derivative of the above log-likelihood, with respect to each of the parameters. Depending on the form of the initial likelihood function and the number of parameters, deriving the gradient could entail extensive amounts of algebra. Instead, we exploit the symbolic processor. We define a function to take the derivative of our equation with respect to a list of parameters,

```
def S(params):
    return [simplify(diff(logLikelihood, var))
            for var in params]
```

and apply it to get the partial derivatives, $S_{\beta}(\theta|Z_i)$,

```
S_beta = S(beta)
print(latex(Matrix(S_beta)))
```

$$\begin{bmatrix} -\frac{g_b}{\sigma^2}(a_0 + a_1x_1 + a_2x_2 + b_bg_b + b_wg_w - y) \\ -\frac{g_w}{\sigma^2}(a_0 + a_1x_1 + a_2x_2 + b_bg_b + b_wg_w - y) \end{bmatrix}$$

and $S_{\eta}(\theta|Z_i)$,

```
S_eta = S(eta)
print(latex(Matrix(S_eta)))
```


$$\begin{bmatrix} \frac{1}{\sigma^2}(-a_0 - a_1x_1 - a_2x_2 - b_bg_b - b_wg_w + y) \\ -\frac{x_1}{\sigma^2}(a_0 + a_1x_1 + a_2x_2 + b_bg_b + b_wg_w - y) \\ -\frac{x_2}{\sigma^2}(a_0 + a_1x_1 + a_2x_2 + b_bg_b + b_wg_w - y) \\ \frac{1}{\sigma^3}(-\sigma^2 + (a_0 + a_1x_1 + a_2x_2 + b_bg_b + b_wg_w - y)^2) \end{bmatrix}$$

Next we define a function to take partial derivatives of a vector with respect to a list of parameters,

```
def secPar(score, pars):
    return transpose(Matrix([[simplify(diff(x, p))
                             for x in list(score)] for p in pars]))
```

and apply it to our first partial derivatives.

```
cmat = secPar(S_beta, eta)
print(latex(cmat))
```

$$\begin{bmatrix} -\frac{g_b}{\sigma^2} - \frac{g_b x_1}{\sigma^2} - \frac{g_b x_2}{\sigma^2} - \frac{2g_b}{\sigma^3}(a_0 + a_1x_1 + a_2x_2 + b_bg_b + b_wg_w - y) \\ -\frac{g_w}{\sigma^2} - \frac{g_w x_1}{\sigma^2} - \frac{g_w x_2}{\sigma^2} - \frac{2g_w}{\sigma^3}(a_0 + a_1x_1 + a_2x_2 + b_bg_b + b_wg_w - y) \end{bmatrix}$$

```
vmat = secPar(S_eta, eta)
K = symbols('K')
print(latex(vmat.subs(a0 + a1*x1 + a2*x2 + b_b*g_b + b_w*g_w - y, K)))
```

$$\begin{bmatrix} -\frac{1}{\sigma^2} & -\frac{x_1}{\sigma^2} & -\frac{x_2}{\sigma^2} & \frac{2K}{\sigma^3} \\ -\frac{x_1}{\sigma^2} & -\frac{x_1^2}{\sigma^2} & -\frac{x_1x_2}{\sigma^2} & \frac{2K}{\sigma^3}x_1 \\ -\frac{x_2}{\sigma^2} & -\frac{x_1x_2}{\sigma^2} & -\frac{x_2^2}{\sigma^2} & \frac{2K}{\sigma^3}x_2 \\ \frac{2K}{\sigma^3} & \frac{2K}{\sigma^3}x_1 & \frac{2K}{\sigma^3}x_2 & \frac{1}{\sigma^4}(-3K^2 + \sigma^2) \end{bmatrix}$$

The expected values of these matrices will give us the covariance and variance matrices of the efficient score equation. For the vmat matrix, we use the substitution $K = a_0 + a_1x_1 + a_2x_2 + b_bg_b + b_wg_w - y$ in order to make the result more concise. This has the added benefit of making it easier to identify patterns in the structure of the resulting matrix. Note that the value of vmat is unchanged; the substitution only occurs in the printed results.

2.3 Expected Values

We require the expectation of the above terms over the joint distribution of Y_i, X_i, G_{bi} , and G_{wi} . One of the limitations of SymPy (and many other symbolic processors) is that SymPy does not currently have the functionality to take expectations over a joint distribution. However, by virtue of our model definition, we can attempt to avoid this impasse by taking a double-expectation instead, i.e., $\mathbb{E}[\mathbb{E}[Y/G_b, G_w, X]]$.

First, we use the (limited) expectation functionality to take the conditional expectation of $Y/G_b, G_w, X$ using the distribution we specified above.

```
Cmat = -E(cmat.subs(y, distY))
print(latex(Cmat))
```

$$\begin{bmatrix} \frac{g_b}{\sigma^2} & \frac{g_b x_1}{\sigma^2} & \frac{g_b x_2}{\sigma^2} & 0 \\ \frac{g_w}{\sigma^2} & \frac{g_w x_1}{\sigma^2} & \frac{g_w x_2}{\sigma^2} & 0 \end{bmatrix}$$

Unfortunately several compound terms remain, i.e., $G_{bi}X_i$ and $G_{wi}X_i$. In order to proceed, we must either determine the expectation over their joint distribution outside of SymPy, (giving us $\mathcal{S}_{\text{eff}}(\theta|Z)$ from (1)), or replace the terms with their empirical estimates, (giving us $\mathcal{S}_{\text{eff}}(\theta|z)$ from (4)). In either case, the necessary values can be represented by new symbolic terms which are substituted for the remaining random variables and products.

For some terms we can easily derive the expected values. For example, by listing all possible values of $G_{F,i}$ and $G_{M,i}$ it can be shown that $\mathbb{E}[G_i] = (G_{M,i} + G_{F,i})/2$. Therefore $\mathbb{E}[G_w] = \mathbb{E}[G_i] - (G_{M,i} + G_{F,i})/2 = 0$. Also note $\mathbb{E}[G_w \mathbf{X}] = \int \mathbf{x} \mathbb{E}[G_w \mathbf{X} / \mathbf{X} = \mathbf{x}] P(\mathbf{X} = \mathbf{x}) d\mathbf{x}$. Again, by considering all values of $G_{F,i}$ and $G_{M,i}$ it can be shown that values of $G_i - (G_{M,i} + G_{F,i})/2$ are symmetrically distributed around zero. Therefore $\mathbb{E}[G_w \mathbf{x}]$ is always 0, and so $\mathbb{E}[G_w \mathbf{X}] = 0$. Given these two assertions, the bottom row of the matrix will be zero. For the remaining terms, we declare the new symbolic variables

```
EGb, EGbX1, EGbX2 = symbols(
    "EGb EGbX1 EGbX2", real=True)
```

then make the substitution. Note that we must take care to substitute non-separable terms first, as the `.subs()` function does not take into account the fact that we are working within the context of expected values, and simply replaces all instances of a symbol.

```
Cmat = Cmat.subs([(g_b*x1, EGbX1), (g_b*x2, EGbX2)])
Cmat = Cmat.subs([(g_b, EGb), (g_w, 0)])
print(latex(Cmat))
```

$$\begin{bmatrix} \frac{EGb}{\sigma^2} & \frac{EGbX_1}{\sigma^2} & \frac{EGbX_2}{\sigma^2} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

This gives our final covariance matrix. As asserted above, the cells corresponding to b_w (the second row) are all zero. This indicates the score for b_w is trivially efficient. Notice also that the cells corresponding to the nuisance parameter σ (the last column) are zero as well. From this we conclude that σ is in fact orthogonal to our parameters of interest, i.e., it does not induce variability on the score function.

We repeat the expected value process for the variance matrix, first taking the conditional expectation using our function and SymPy's built-in capabilities.

```
Vmat = -E(vmat.subs(y, distY))
print(latex(Vmat))
```

$$\begin{bmatrix} \frac{1}{\sigma^2} & \frac{x_1}{\sigma^2} & \frac{x_2}{\sigma^2} & 0 \\ \frac{x_1}{\sigma^2} & \frac{x_1^2}{\sigma^2} & \frac{x_1 x_2}{\sigma^2} & 0 \\ \frac{x_2}{\sigma^2} & \frac{x_1 x_2}{\sigma^2} & \frac{x_2^2}{\sigma^2} & 0 \\ 0 & 0 & 0 & \frac{2}{\sigma^2} \end{bmatrix}$$

Next we make a simplifying assumption that the cofactors are normalized, allowing us to specify $\mathbb{E}[\mathbf{X}_i] = \mathbf{0}$ and $\mathbb{E}[\mathbf{X}_i^2] = \mathbf{1}$. We declare one more new symbolic variable for the remaining product and make our substitutions, again being careful to maintain non-separable terms.

```
EX1X2 = symbols("EX1X2", real=True)
Vmat = Vmat.subs([(x1**2,1),(x2**2,1),(x1*x2,EX1X2)])
Vmat = Vmat.subs([(x1, 0),(x2, 0)])
print(latex(Vmat))
```

$$\begin{bmatrix} \frac{1}{\sigma^2} & 0 & 0 & 0 \\ 0 & \frac{1}{\sigma^2} & \frac{EX1X2}{\sigma^2} & 0 \\ 0 & \frac{EX1X2}{\sigma^2} & \frac{1}{\sigma^2} & 0 \\ 0 & 0 & 0 & \frac{2}{\sigma^2} \end{bmatrix}$$

Notice the block-diagonal structure.

We now have all of the components we need to compose (either version of) the efficient score. The new symbolic variables can be replaced with known expected values or average observed values, and the full equation can be constructed. Once more, symbolic processing makes this process simple.

```
Vinv = simplify(Vmat.inv())
Seff = simplify(Matrix(S_beta)-(Cmat*Vinv*Matrix(S_eta)))
```

```
K1, K2 = symbols("K1 K2")
print(latex(Seff.subs(a0 + a1*x1 + a2*x2 - y, K1).subs(EX1X2**2-1, K2)))
```

$$\left[\begin{array}{c} \frac{1}{K_2 \sigma^2} (K_1 + b_b g_b + b_w g_w) (K_2 (E G b - g_b) - x_1 (E G b X_1 - E G b X_2 E X_1 X_2) + x_2 (E G b X_1 E X_1 X_2 - E G b X_2)) \\ - \frac{g_w}{\sigma^2} (K_1 + b_b g_b + b_w g_w) \end{array} \right]$$

Again, substitution is used to make the result more concise. The repeated terms $(a_0 + a_1 x_1 + a_2 x_2 - y)$ and $\mathbb{E}[X_1 X_2]^2 - 1$ are replaced with K_1 and K_2 , respectively.

Ultimately the statistic is then calculated under H_0 , i.e., the parameters of interest, β , are set to zero, and the nuisance parameters are replaced with MLEs (also computed under the null). We make the same substitutions for readability, and to allow direct comparison between these equations and the previous result.

```
Sstar = Seff.subs([(b_b, 0), (b_w, 0)])
print(latex(Sstar.subs(a0 + a1*x1 + a2*x2 - y, K1).subs(EX1X2**2-1, K2)))
```

$$\left[\begin{array}{c} \frac{K_1}{K_2 \sigma^2} (K_2 (E G b - g_b) - x_1 (E G b X_1 - E G b X_2 E X_1 X_2) + x_2 (E G b X_1 E X_1 X_2 - E G b X_2)) \\ - \frac{K_1 g_w}{\sigma^2} \end{array} \right]$$

The sum of $\mathcal{S}^*$ over all families will yield the efficient score statistic for our hypothesis. To reiterate, no manual derivations were required to execute this example, and no L_ATEX coding had to be done to generate the equations in this report.

3 Discussion

With the aid of symbolic computing, we arrive at the efficient score for our model. This is done with significantly less effort than a pencil and paper calculation, and we are spared the trouble of finding and correcting the arithmetic or algebraic errors which might plague a manual derivation, reducing our task to the application of theory. These benefits scale dramatically with the complexity of the likelihood function. Further, notice that all of section 2.2 is agnostic to the model being used. This code is reusable across any future applications (excepting the specification of the likelihood function and expected values).

While symbolic computing software makes the efficient score easier to derive, it should not be applied indiscriminately. Taking into account the structure of the model, there may be

theoretical properties which allow us to simplify the problem. Symbolic computing can then be applied to the simplified model, or at least implicitly take into account these properties. For example, if we knew that the nuisance parameters were orthogonal to β , then the covariance between the nuisance parameters and the parameters of interest, (2), would be zero. Alternatively, if the maximum likelihood estimate of the nuisance parameter is available, then the $S_{\eta}(\theta|Z)$ term in (1) will be $S_{\eta}(\beta_0, \hat{\eta}_0|Z) = 0$, where $\beta_0 = 0$ and $\hat{\eta}_0$ is the MLE of η under H_0 . In either case, the term correcting for the variance induced by the nuisance parameters goes to zero, and the naive score, $S_{\beta}(\theta|Z)$, is itself efficient. While using symbolic computing to account for the (nonexistent) variance induced by the nuisance parameters would still yield correct results, doing so would be an unnecessary complication of the problem.

As seen in our example, the built-in functionality of SymPy does not offer everything we need “right out of the box”. Some utilities can easily be added by the user. Whether using predefined or user-defined functions, it is necessary to confirm that each manipulation of equations gives a reasonable result, in order to ensure that errors are not allowed to propagate blindly. However, in practice, ex post facto corrections can be carried out instantly by simply re-executing the subsequent portions of the script. Other utilities, like the expectation over a joint distribution, are difficult to implement in SymPy, but may be available in other symbolic processors.

The readability of SymPy code, its ease of integration with L_ATEX via PythonTEX, and the fact that it is free and open source make it an attractive choice, despite its limitations. Conversely, Mathematica, while not freely available like SymPy, is far more highly developed. Unlike SymPy, it has the capability to compute expectations over joint distributions. An independently developed package, NCAIgebra (Helton et al. 1996), adds the capability for non-commutative algebra (i.e., matrix multiplication). However, while Mathematica has the functionality to import and export L_ATEX code, it does not offer the same direct integration into TEX documents as PythonTEX, and so affords less flexibility in the formal reporting of results. Alternatively, Maple offers the depth and robust capabilities of Mathematica, including the option of exporting code to L_ATEX. Maple also has the distinct ability to generate computationally efficient code based on user-specified equations. In the event the resulting efficient score is very complex, Maple can translate the symbolic function into optimized code for several compiled languages, including Python and C. SymPy also has utilities for generating code in other languages, including C, using the `codegen()` function, however the resulting routines are not programmatically optimized.

The portability of equations from derivation to implementation allows one to create a pipeline which flows directly into simulation or analysis, making symbolic processors attractive tools for rapid prototyping. The computer algebra system would allow one to quickly implement changes to equations, which could then be used in simulations to check the implications of different assumptions, or to explore the operating characteristics of a statistic.

4 Conclusion

Symbolic computing simplifies and streamlines the derivation of the efficient score, removing the burden of calculation and reducing the potential for arithmetic error. Much of the code for the derivation is reusable and it is entirely reproducible. Depending on the choice of symbolic computing software, the results can be automatically converted into computationally optimized code, to easily apply the inferential method, or directly integrated into L_ATEX, for reporting and presentation. Symbolic computing removes the complications of an attractive but potentially complex analysis, conducting inference on the basis of the score, and allows code to flow directly from derivation to analysis to reporting.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

References

- Abecasis GR, Cardon LR, Cookson WO. A general test of association for quantitative traits in nuclear families. *The American Journal of Human Genetics*. 2000; 66:279–292. [PubMed: 10631157]
- Andrews DF, Stafford JE. *Symbolic Computation for Statistical Inference*. Oxford University Press; New York, NY: 2000.
- Caviness B. Computer algebra: Past and future. *Journal of Symbolic Computation*. 1986; 2(3):217–236.
- Conneely KN, Boehnke M. So many correlated tests, so little time! rapid adjustment of p values for multiple correlated tests. *The American Journal of Human Genetics*. 2007; 81(6):1158–1168. [PubMed: 17966093]
- Continuum Analytics. Anaconda software distribution, v2-2.4.0. 2016. <https://continuum.io>
- Ekström Smedby K, Lindgren CM, Hjalgrim H, Humphreys K, Schöllkopf C, Chang ET, Roos G, Ryder LP, Falk KI, Palmgren J, Kere J, Melbye M, Glimelius B, Adami H-O. Variation in dna repair genes *ercc2*, *xrcc1*, and *xrcc3* and risk of follicular lymphoma. *Cancer Epidemiology and Prevention Biomarkers*. 2006; 15(2):258–265.
- Freedman DA. How can the score test be inconsistent? *The American Statistician*. 2007; 61(4):291–295.
- Freedman ND, Ahn J, Hou L, Lissowska J, Zatonski W, Yeager M, Chanock SJ, Chow WH, Abnet CC. Polymorphisms in estrogen- and androgen-metabolizing genes and the risk of gastric cancer. *Carcinogenesis*. 2009; 30(1):71–77. [PubMed: 19015200]
- Fulker D, Cherny S, Sham P, Hewitt J. Combined linkage and association sib-pair analysis for quantitative traits. *The American Journal of Human Genetics*. 1999; 64(1):259–267. [PubMed: 9915965]
- Gaudet MM, Kirchhoff T, Green T, Vijai J, Korn JM, Guiducci C, Segr AV, McGee K, McGuffog L, Kartsonaki C, Morrison J, Healey S, Sinilnikova OM, Stoppa-Lyonnet D, Mazoyer S, Gauthier-Villars M, Sobol H, Longy M, Frenay M, Collaborators GS, Hogervorst FBL, Rookus MA, Colle JM, Hoogerbrugge N, van Roozendaal KEP, Collaborators HS, Piedmonte M, Rubinstein W, Nerenstone S, Van Le L, Blank SV, Calds T, de la Hoya M, Nevanlinna H, Aittomäki K, Lazaro C, Blanco I, Arason A, Johannsson OT, Barkardottir RB, Devilee P, Olopade OI, Neuhausen SL, Wang X, Fredericksen ZS, Peterlongo P, Manoukian S, Barile M, Viel A, Radice P, Phelan CM, Narod S, Rennert G, Lejbkowitz F, Flugelman A, Andrulis IL, Glendon G, Ozelik H, OCGN, Toland AE, Montagna M, D'Andrea E, Friedman E, Laitman Y, Borg A, Beattie M, Ramus SJ, Domchek SM, Nathanson KL, Rebbeck T, Spurdle AB, Chen X, Holland H, kConFab John EM, Hopper JL, Buys SS, Daly MB, Southey MC, Terry MB, Tung N, Overeem Hansen TV, Nielsen FC, Greene MI, Mai PL, Osorio A, Durn M, Andres R, Bentez J, Weitzel JN, Garber J, Hamann U, Peock S, Cook M, Oliver C, Frost D, Platte R, Evans DG, Lalloo F, Eeles R, Izatt L, Walker L,

- Eason J, Barwell J, Godwin AK, Schmutzler RK, Wappenschmidt B, Engert S, Arnold N, Gadzicki D, Dean M, Gold B, Klein RJ, Couch FJ, Chenevix-Trench G, Easton DF, Daly MJ, Antoniou AC, Altshuler DM, Offit K. Common genetic variants and modification of penetrance of *brca2*-associated breast cancer. *PLOS Genetics*. 2010; 6(10):1–12.
- Hall WJ, Mathiason DJ. On large-sample estimation and testing in parametric models. *International Statistical Review/Revue Internationale de Statistique*. 1990; 58(1):77–97.
- Helton JW, Miller RL, Stankus M. NCAIgebra: A Mathematica package for doing noncommuting algebra. 1996. <http://math.ucsd.edu/ncalg>
- Horton NJ, Hardin JS. Teaching the next generation of statistics students to think with data: Special issue on statistics and the undergraduate curriculum. *The American Statistician*. 2015; 69(4):259–265.
- Jiang Y, Ji Y, Sibley AB, Li Y-J, Allen AS. Leveraging population information in family-based rare variant association analyses of quantitative traits. *Genetic Epidemiology*. 2017; 41(2):98–107. [PubMed: 27917519]
- Lin DY. An efficient monte carlo approach to assessing statistical significance in genomic studies. *Bioinformatics*. 2005; 21(6):781. [PubMed: 15454414]
- Magnus JR, Neudecker H. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. 3. John Wiley & Sons Ltd; Chichester, England: 2007.
- Maplesoft, a division of Waterloo Maple Inc. Maple v18.0. Springer Science+ Business Media, LLC; Waterloo, Ontario: 2014.
- Maxima. Maxima, a computer algebra system, v5.39.0. 2016. <http://maxima.sourceforge.net/>
- Moses J. Maccsma: A personal history. *Journal of Symbolic Computation*. 2012; 47(2):78–84.
- Pang H, Kim I, Zhao H. Random effects model for multiple pathway analysis with applications to type ii diabetes microarray data. *Statistics in Biosciences*. 2015; 7(2):167–186. [PubMed: 26640601]
- Poore GM. Reproducible documents with pythontex. *Proceedings of the 12th Python in Science Conference*; 2013. 123–130.
- Rao CR. Tests of significance in multivariate analysis. *Biometrika*. 1948; 35:58–79. [PubMed: 18867413]
- Schaffner S, Foo C, Gabriel S, Reich D, Daly M, Altshuler D. Calibrating a coalescent simulation of human genome sequence variation. *Genome Research*. 2005; 15(11):1576–1583. [PubMed: 16251467]
- Seaman S, Müller-Myhsok B. Rapid simulation of p values for product methods and multiple-testing adjustment in association studies. *The American Journal of Human Genetics*. 2005; 76(3):399–408. [PubMed: 15645388]
- Siamakpour-Reihani S, Owzar K, Jiang C, Scarbrough PM, Craciunescu OI, Horton JK, Dressman HK, Blackwell KL, Dewhirst MW. Genomic profiling in locally advanced and inflammatory breast cancer and its link to dce-mri and overall survival. *International Journal of Hyperthermia*. 2015; 31(4):386–395. [PubMed: 25811737]
- Sibley AB, Li Z, Jiang Y, Chan C, Allen A, Owzar K. Facilitating the calculation of the efficient score using symbolic computing. *JSM Proceedings, Section on Statistical Education*; Alexandria, VA: American Statistical Association; 2014. 3136–3143.
- Sorensen D, Gianola D. *Likelihood, Bayesian, and MCMC Methods in Quantitative Genetics*. Springer-Verlag New York Inc; New York, NY: 2013.
- SymPy Development Team. SymPy: Python library for symbolic mathematics. 2014. <http://www.sympy.org>
- The Sage Developers. SageMath, the Sage Mathematics Software System, v7.6. 2017. <http://www.sagemath.org>
- Tsiatis AA. *Semiparametric Theory and Missing Data*. Springer Science+Business Media, LLC; New York, NY: 2006.
- Wolfram Research, Inc. Mathematica v9.0. Wolfram Research, Inc; Champaign, IL: 2012.
- Zhang X, Johnson AD, Hendricks AE, Hwang S-J, Tanriverdi K, Ganesh SK, Smith NL, Peyser PA, Freedman JE, O'Donnell CJ. Genetic associations with expression for genes implicated in gwas

studies for atherosclerotic cardiovascular disease and blood phenotypes. *Human Molecular Genetics*. 2014; 23(3):782. [PubMed: 24057673]

Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript