

Relating Disparate Measures of Coagulopathy Using Unorthodox Data: A Hybrid Mechanistic-Statistical Approach

*Arya A. Pourzanjani, Tie Bo Wu, Benjamin B. Bales, Linda R. Petzold**

Traumatic injury is the leading cause of death for people under the age of 44 (Hoyert and Xu 2012). Many of these deaths are the result of uncontrolled bleeding due to a trauma-induced disorder called Acute Traumatic Coagulopathy, or known more simply as Coagulopathy (Brohi et al. 2003). How major trauma causes Coagulopathy and how to treat the disease is still a subject of ongoing research. There are various competing hypotheses for why so many trauma patients are coagulopathic. The Coagulation Cascade and the Fibrinolytic system are complex networks of dynamically interacting proteins in blood that are responsible for forming and breaking up clots (Gonzalez et al. 2014).

It is generally understood that Coagulopathy comes as a result of a malfunction in one of or both these systems. To better study these complex networks, and how they are affected during trauma, doctors and scientists have two major assays at their disposal: direct protein concentration measurements and Thromboelastography, known as TEG (Figure 1). Direct protein concentration measurements can tell us the concentration levels of key players in the body’s coagulation system, thus they can help us to understand why a patient’s blood is not clotting and how they can be treated. Unfortunately, these tests are only available at very select number of hospitals specializing in trauma, they are expensive to run, and most importantly they are slow to run in a setting where applying the correct treatment as quickly as possible is of the utmost importance.

In contrast to direct protein measurements, TEG measurements are ubiquitous, inexpensive to run, and can provide results in as little as 20 minutes. However, they do not measure protein concentrations directly. Instead, TEG works by placing a small sample of blood in a cup, chemically initiating the clotting process, then using a metal probe to measure the physical size of the resulting clot over time in millimeters (mm) (Figure 2). The resulting output is a measure of clot thickness over time for the patient that is indicative of several important features of their clotting state including:

- How long it takes for a patient’s blood to start forming a clot
- How fast the clot grows once clotting is initiated
- How strong the patient’s clots become
- How long the clots are able to maintain their integrity before being broken up

The Goal: Inferring Protein Concentrations Using TEG and Our Mechanistic Understanding of the Coagulation System.

While TEG measurements clearly contain useful information regarding a patient’s clotting state, they are simply a proxy for the latent system of clotting proteins in the blood that is much more difficult to measure. Ideally, we can use our mechanistic understanding of the coagulation system in the form of Ordinary Differential Equations (ODEs) along with a statistical model, to better understand what exactly TEG is telling us about the state of the underlying coagulation system, and furthermore to infer a patient’s protein concentrations using solely their TEG measurements.

*University of California, Santa Barbara

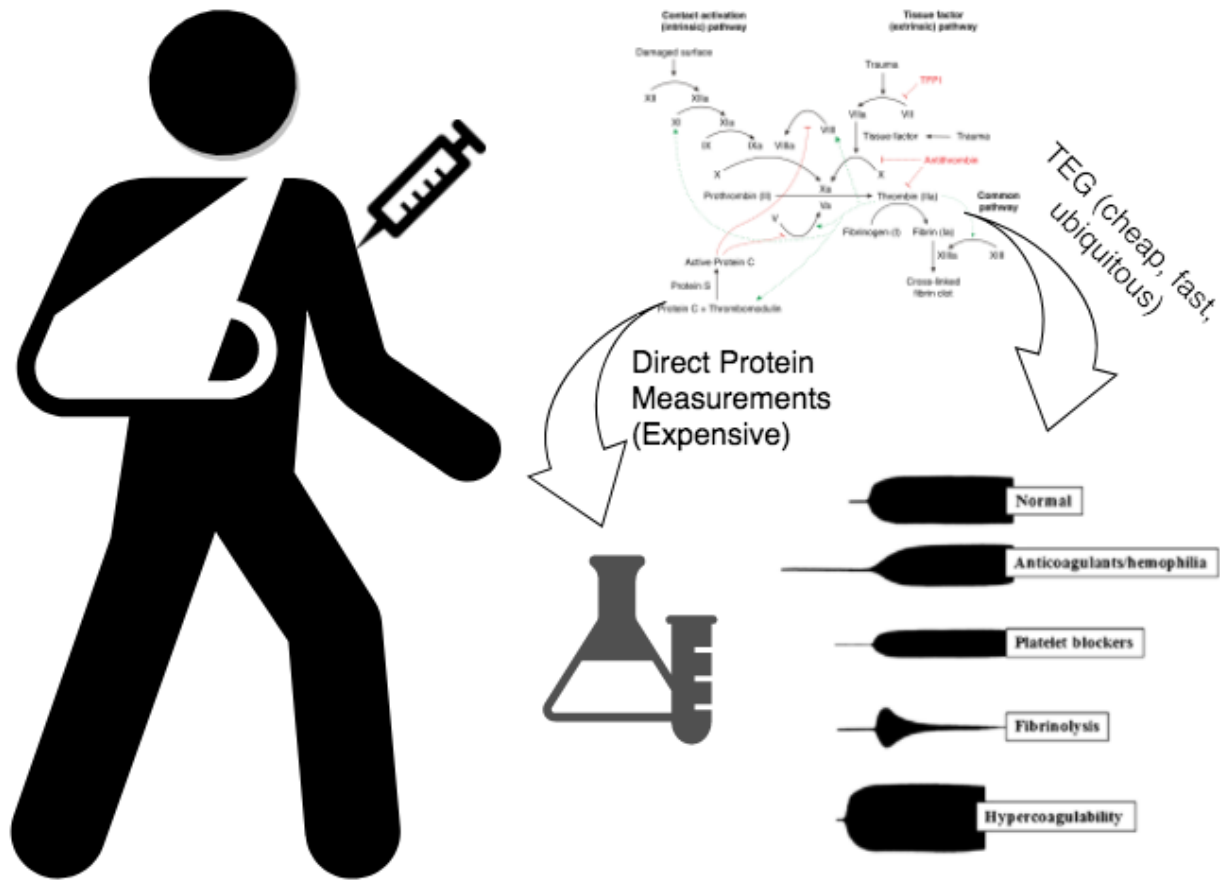


Figure 1: Figure 1: Direct protein concentration measurements and TEG are used to examine the state of the Coagulation Cascade of trauma patients using a small blood sample. While direct protein measurements are obviously more informative of the exact state of patient's blood at any given time, they are slow and expensive to obtain compared to TEG.

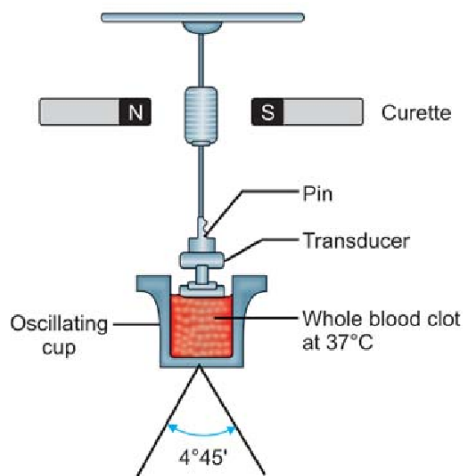


Figure 2: Figure 2: In a TEG assay a small sample of blood is placed in a cup which is spun around quickly to initiate the clotting process. A thin metal pin then measures the size over time in millimeters (mm) of the resulting clot.

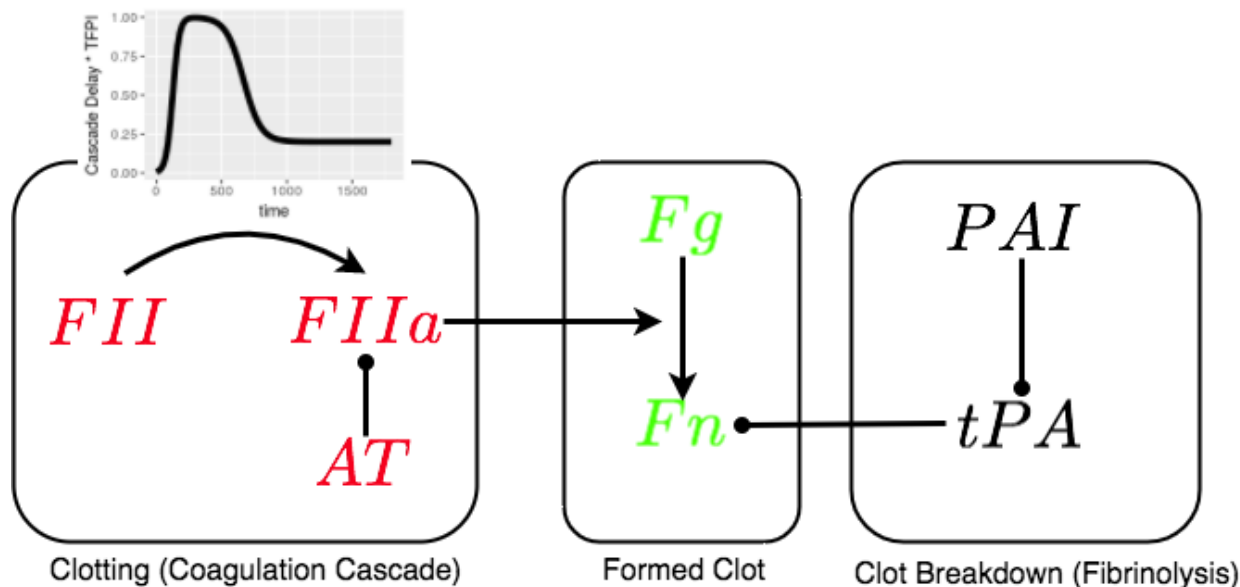


Figure 3: **Figure 3:** A seven-state model of coagulation that models the clotting process, actual clots, and the clot breakdown process. The rate of activation of FII in the coagulation cascade is summarized by a delay term that is governed by parameters that are specific to the patient.

A Mechanistic Model of the Coagulation System

The coagulation system is a well-studied, with several mechanistic ODE models in the literature that describe how the system evolves dynamically over time. Models for the coagulation system vary widely in the number of states, reactions, and parameters they contain, including complex models with up to 80 states (Mitrophanov, Wolberg, and Reifman 2014). For our purposes we developed a simple reduced-order model based off of elements from both the work by (Mitrophanov, Wolberg, and Reifman 2014) as well as (Sagar and Varner 2015) that captures the most important players in the coagulation system. The model includes the most basic components of the clotting process: the coagulation cascade responsible for forming clots, actual clot material, and clot breakdown or Fibrinolysis (Figure 3).

We model the coagulation cascade as primarily consisting of the activation of the blood protein FactorII (FII) to its activated form $FIIa$ via the action of a sigmoid delay function that is parameterized by parameters b and c . We model these parameter values as specific to the patient. Importantly, $FIIa$ can be blocked by antithrombin (AT). $FIIa$ once activated can then facilitate the conversion of raw clot material, Fibrinogen, or Fg in to an actual clot Fibrin, or Fn . Once a clot is formed, the clot can be broken up by the protein tPA , which itself can be blocked by the protein PAI . The differential equations for this model summarize this process and are shown below. Values for reaction constants represent how fast these respective reactions occur with respect to on another. The values we used for these constant were either gathered from the literature or fit using Maximum A-Posteriori (MAP) estimation.

$$\begin{aligned}
\frac{dFII}{dt} &= -\text{CascadeDelay}(t, b, c) \cdot \text{TFPI}(t) \cdot \frac{FII}{K_{FIIa} + FII} \\
\frac{dFIIa}{dt} &= \text{CascadeDelay}(t, b, c) \cdot \text{TFPI}(t) \cdot \frac{FII}{K_{FIIa} + FII} - k_{AT} \cdot FIIa \cdot AT \\
\frac{dAT}{dt} &= -k_{AT} \cdot FIIa \cdot AT \\
\frac{dFg}{dt} &= -k_{clot} \cdot FIIa \cdot \frac{Fg}{K_{clot} + Fg} \\
\frac{dFn}{dt} &= k_{clot} \cdot FIIa \cdot \frac{Fg}{K_{clot} + Fg} - k_{lys} \cdot tPA \cdot \frac{Fn}{K_{lys} + Fn} \\
\frac{dtPA}{dt} &= -k_{PAI} \cdot tPA \cdot PAI \\
\frac{dPAI}{dt} &= -k_{PAI} \cdot tPA \cdot PAI
\end{aligned}$$

Using the R package deSolve, we can forward simulate from this ODE to get an idea of what the solutions for this system look like:

```

library(tidyverse)
library(deSolve)

seven.state <- function(t, state, parameters) {
  with(as.list(c(state, parameters)), {

    cascade_delay <- 1/(1+exp(c-b*t))
    tfpi <- (1-tfpi_min)/(1+exp(-ct+bt*t)) + tfpi_min

    r_FIIa <- k_FIIa*cascade_delay*tfpi*((FII/g_FII)^c_FIIa)/(K_FIIa + (FII/g_FII)^c_FIIa)
    r_AT <- k_AT*FIIa*(AT/g_AT)
    r_clot <- k_clot*FIIa*((Fg/9e-6)^c_clot)/(K_clot + (Fg/9e-6)^c_clot)
    r_lys <- k_lys*tPA*(Fn^c_lys)/(K_lys + Fn^c_lys)
    r_PAi <- k_PAi*tPA*PAI

    dFII <- g_FII*(-r_FIIa)           # pct activity
    dFIIa <- r_FIIa - r_AT           # 10s of nmol/L
    dAT <- -g_AT*r_AT                # pct activity
    dFg <- -r_clot                   # 1 mg/dl = 29.41 nmol/L
    dFn <- 1e7*(r_clot - r_lys)      # mm of clot
    dtPA <- -r_PAi                   # 1 ng/mL = 14.29 pmol/L
    dPAI <- -r_PAi                   # 1 ng/mL = 23.26 pmol/L

    list(c(dFII, dFIIa, dAT, dFg, dFn, dtPA, dPAI))
  })
}

parameters <- c(g_FII = 100/1.4e-6, g_AT = 100/3.4e-6,
               c = 5.0, b = 0.4e-1,
               ct = 10, bt = 0.15e-1, tfpi_min = 0.2,
               k_FIIa = 3.5e-9, c_FIIa = 1, K_FIIa = 1.4e-6,
               k_AT = 1.6e4,
               k_clot = 3.0, c_clot = 1, K_clot = 0.75,
               k_lys = 1, c_lys = 1, K_lys = 0.5,
               k_PAi = 4.5e5)

```

```

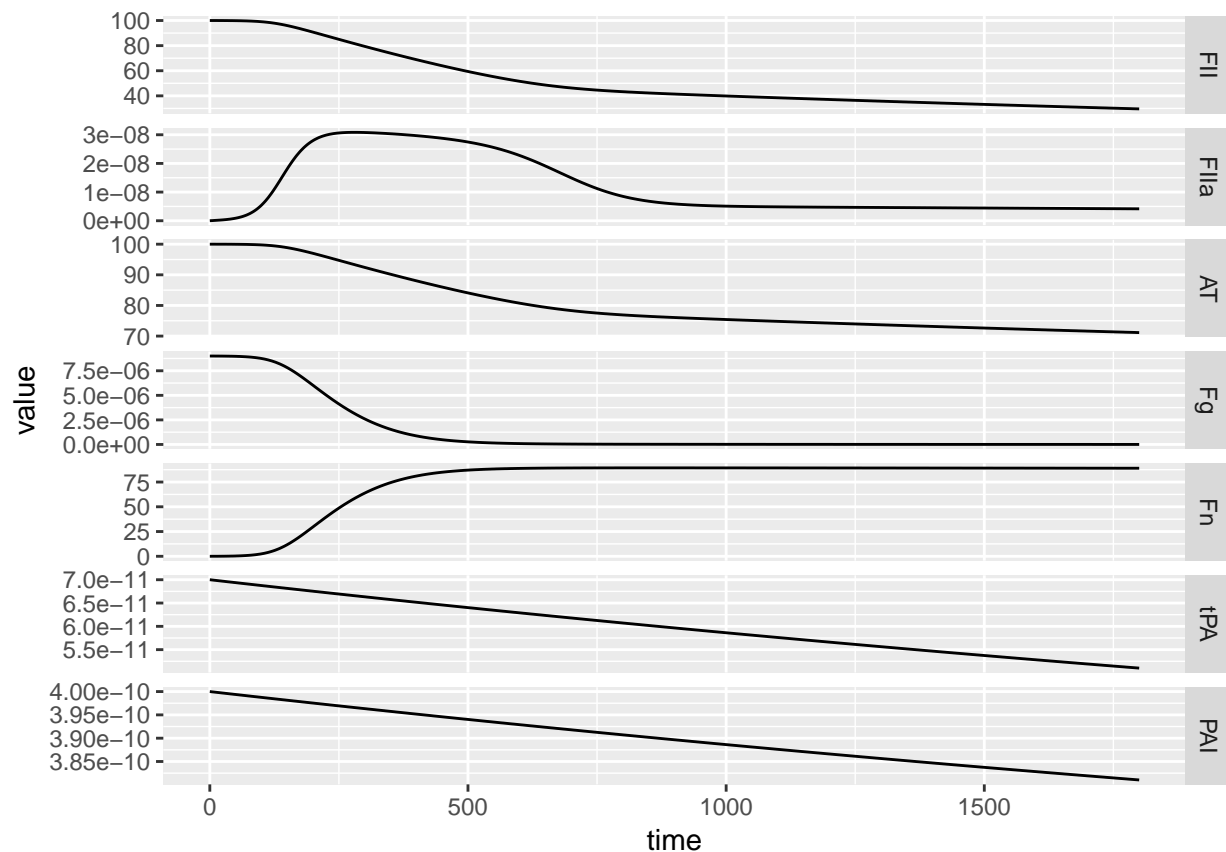
state <- c(FII = 1e2, FIIa = 0, AT = 100, Fg = 9e-6, Fn = 0, tPA = 7e-11, PAI = 4e-10)

times <- seq(0, 1800, by = 2)

out <- ode(y = state, times = times, func = seven.state,
          parms = parameters, method = "bdf",
          atol = 1e-6, rtol = 1e-5) %>% as.data.frame %>% as_tibble

out %>%
  gather(state, value, -time) %>%
  mutate(state = factor(state, levels = c("FII", "FIIa", "AT", "Fg", "Fn", "tPA", "PAI"))) %>%
  ggplot(aes(time,value)) + geom_line() + facet_grid(state ~ ., scales = "free")

```



A Mechanistic Model for TEG

While the model includes the concentration of F_n (Fibrin) which is a critical component of clots, the actual clot thickness which TEG measures is not measuring F_n per se, but some function of it. Following Sagar and Varner (2015) , we used the Hill function

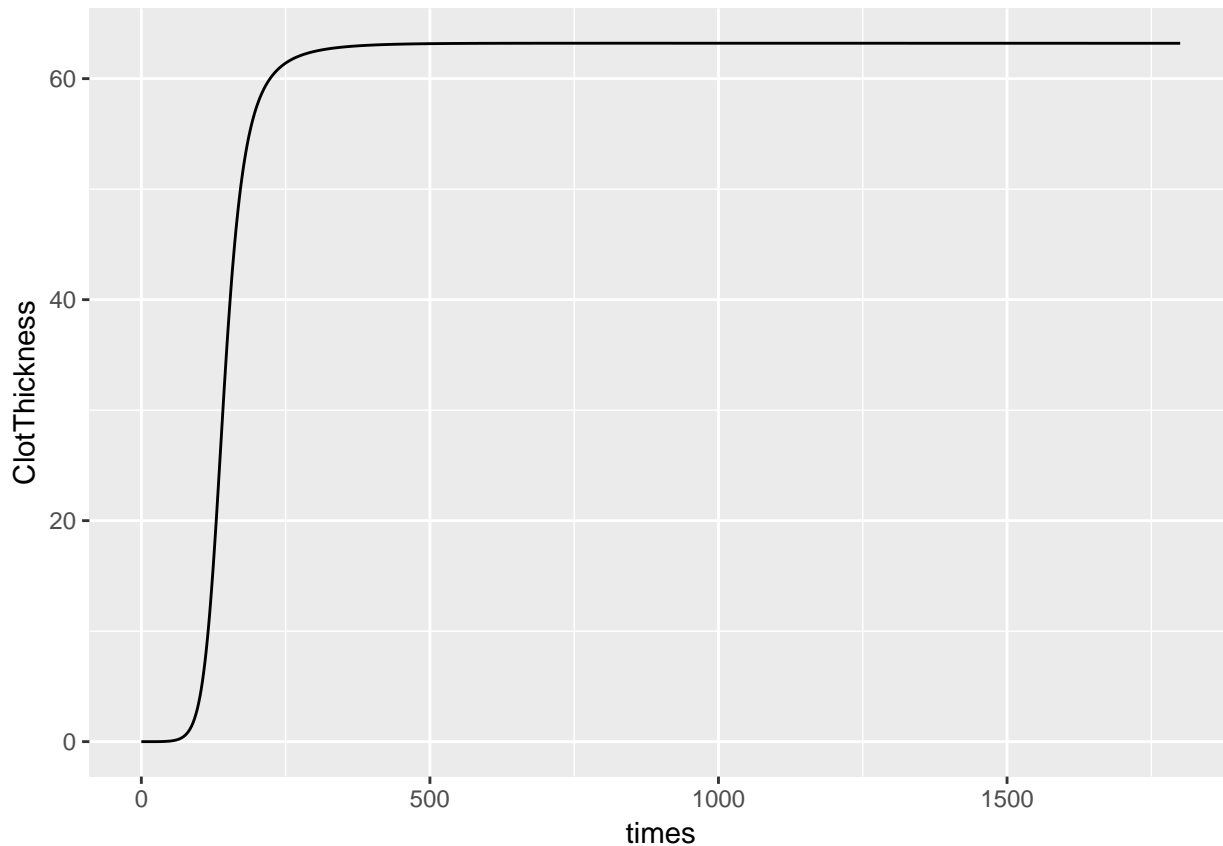
$$\text{ClotThickness}(t) = k \frac{F_n^2}{K + F_n^2}$$

with $k = 64.0$ and $K = 100.0$ to translate F_n to clot thickness. Clot thickness is plotted below for the ODE we simulated above:

```

Fn <- out$Fn
ClotThickness <- 64.0 * Fn^2/(100.0 + Fn^2)
qplot(times, ClotThickness, geom = "line")

```



Inferring Hybrid Mechanistic-Statistical Models and the Unorthodox Nature of TEG Data

Typically, mechanistic ODE models are fit in Stan using data that consists of the states of the ODE over time, see e.g. (Carpenter 2018) or (Margossian and Gillespie 2017) . In these settings one typically posits an error distribution for the data that is centered around the forward simulated values of the ODE. In contrast, our TEG data does not come in the form of clot thickness over time, but rather in the form of four quantities derived from the clot thickness curve that are typically used in the medical community to summarize the most important properties of a TEG curve (Figure 4). These four quantities are described below:

1. **R:** The time (in minutes) for the clot to reach 2 mm. This quantity represents the time it takes for the clotting process to initiate.
2. **K:** The time (in minutes) for the clot to reach 20 mm from the time it reached 2 mm. This quantity represents speed of clot formation.
3. **MA:** The maximum amplitude of the clot i.e. the size of the clot when it is at its largest. This quantity measures the strength of a patient’s clot.
4. **Ly30:** The percentage of the clot which has broken down after 30 minutes as compared to the maximum amplitude of the clot. This quantity measures how fast clots are being broken up.

To use these quantities to infer unknown parameters and initial conditions, we must first forward simulate our ODE, compute the clot thickness as a function of F_n , then use the trajectory of clot of thickness over

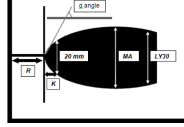


Figure 4: **Figure 4: TEG curves represent the thickness of a clot over time and are summarized by the four key quantities R, K, MA, and Ly30.**

time to derive our simulated TEG data, which we finally can compare to our data by positing some statistical model. We describe the finer points of this process in the following section.

Inferring ODEs Using Hitting Time and Max Data in Stan

In the typical ODE estimation setting, where our data consists of the value of the ODE states over time, y_n , $n = 1, \dots, N$ we typically have a likelihood of the form

$$\prod_{n=1}^N p(y_n | y_n^{(sim)}(y_0, \theta))$$

where $y_n^{(sim)}(y_0, \theta)$ is a forward simulation of our ODE given the initial conditions y_0 and the parameters θ . The Hamiltonian Monte Carlo (HMC) algorithm in Stan works by taking derivatives of the likelihood with respect to the quantities we are trying to estimate, in this case y_0 and θ .

In our case, the data consists of hitting times like R , which is the time the clot takes to reach a size of 2 mm, and we instead set up a probabilistic model, or likelihood of the form

$$p(R | R^{(sim)}(y_0, \theta)).$$

For HMC to get the correct gradients, we have to be careful about how we compute the hitting time $R^{sim}(y_0, \theta)$ in our Stan program. First, note that the ODE solver in Stan will return the value of the solution at discrete time points, which we will then use to compute clot thickness, C_n at the discrete time points t_1, \dots, t_N . R is a continuous value formally defined as

$$R := \inf\{t : C(t|y_0, \theta) > 2.0\}$$

where $C(t)$ is the clot thickness over time. The discrete analogue we would be able to compute with our ODE solution C_n is

$$R := \min\{t_n : C_n(y_0, \theta) > 2.0\}$$

which unfortunately does not have a smooth derivative with respect to the initial conditions and unknown parameters. In practice, this would cause problems for HMC and Stan, because the log-likelihood should be a smooth function of the unknowns, and ideally have smooth derivatives as well.

To ameliorate this, we can use our discrete solution points $\{C_n(y_0, \theta)\}$ to obtain a continuous function $C(t|y_0, \theta)$ by interpolating the solution points using cubic splines. The interpolated function will have two smooth derivatives, allowing Stan's HMC to run smoothly.

Using Cubic Splines to Obtain Continuous ODE Solutions in Stan

Fortunately, the Stan language is expressive enough to allow us to easily implement code for computing a smooth function $C(t)$ that interpolates through the discrete points C_n , $n = 0, \dots, N$ defined at the points t_0, \dots, t_N . In particular, we can write a function in Stan to fit a cubic interpolating spline through a given set of points. We provide a quick review of cubic smoothing splines and how to compute them loosely following the exposition in (Quarteroni, Sacco, and Saleri 2010). Actual Stan code for computing cubic splines is freely available in our Stan files.

Our aim is to derive the functional form of a group of cubic splines $s_{3,i-1}(t)$ over the intervals $[t_{i-1}, t_i]$. The functions will be cubic polynomials and will be twice differentiable, even at the nodes t_0, \dots, t_N . We first define $f_i = s_3(t_i)$, $m_i = s_3'(t_i)$, and $M_i = s_3''(t_i)$ for $i = 0, \dots, N$. Since $s_{3,i-1}$ is a cubic polynomial, its second derivative is linear. Since the cubic spline must have continuous second derivatives we have

$$s_{3,i-1}''(t) = M_{i-1} \frac{t_i - t}{h_i} + M_i \frac{t - t_{i-1}}{h_i}$$

for $t \in [t_{i-1}, t_i]$ where $h_i = t_i - t_{i-1}$. Integrating twice we obtain

$$s_{3,i-1}(t) = M_{i-1} \frac{(t_i - t)^3}{6h_i} + M_i \frac{(t - t_{i-1})^3}{6h_i} + C_{i-1}(t - t_{i-1}) + \tilde{C}_{i-1}$$

The constants C_{i-1} and \tilde{C}_{i-1} are uniquely determined by imposing the end point values $s_3(t_{i-1}) = f_{i-1}$ and $s_3(t_i) = f_i$, yielding for $i = 1, \dots, N - 1$

$$\begin{aligned} \tilde{C}_{i-1} &= f_{i-1} - M_{i-1} \frac{h_i^2}{6} \\ C_{i-1} &= \frac{f_i - f_{i-1}}{h_i} - \frac{h_i}{6} (M_i - M_{i-1}) \end{aligned}$$

By imposing continuity of the first derivatives at the nodes we arrive at the linear system

$$\mu_i M_{i-1} + 2M_i + \lambda_i M_{i+1} = d_i, \quad i = 1, \dots, N_1$$

where

$$\begin{aligned} \mu_i &= \frac{h_i}{h_i + h_{i+1}} \\ \lambda_i &= \frac{h_{i+1}}{h_i + h_{i+1}} \\ d_i &= \frac{6}{h_i + h_{i+1}} \left(\frac{f_{i+1} - f_i}{h_{i+1}} - \frac{f_i - f_{i-1}}{h_i} \right) \end{aligned}$$

for $i = 1, \dots, N - 1$. Setting $\lambda_0 = \mu_N = 1$ and $d_0 = d_1$ leads to the following linear equation which defines our spline coefficients:

$$\begin{pmatrix} 2 & \lambda_0 & 0 & \cdots & 0 \\ \mu_1 & 2 & \lambda_1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \mu_{N-1} & 2 & \lambda_{N-1} \\ 0 & \cdots & 0 & \mu_N & 2 \end{pmatrix} \begin{pmatrix} M_0 \\ M_1 \\ \vdots \\ M_{N-1} \\ M_N \end{pmatrix} = \begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_{N-1} \\ d_N \end{pmatrix}$$

Because this system is tridiagonal, we can solve it in $\mathcal{O}(N)$ time using the Thomas algorithm, which is implemented in our provided Stan files.

Obtaining Hitting Times from the Spline Interpolation

With our continuous clot thickness function in hand, we are now able to compute stopping times that are a smooth function of our unknowns by using an algebraic solver on our continuous function and appealing to the implicit function theorem. Letting $C(t|y_0, \theta)$ represent our spline function that interpolates the discrete points $C_n(y_0, \theta)$, R is defined implicitly as

$$C(R(y_0, \theta)|y_0, \theta) = 2.0.$$

Note that R , the time that the clot hits 2.0 mm is dependent on the initial value of the system, as well as the parameter values of the system. Taking the partial derivative of both sides of the equation with respect to θ yields

$$\frac{\partial}{\partial R}C(R|y_0, \theta)\frac{\partial R}{\partial \theta} + \frac{\partial}{\partial \theta}C(R|y_0, \theta) = 0.0.$$

which then yields the correct partial derivative we need for HMC:

$$\frac{\partial R}{\partial \theta} = -\frac{\partial}{\partial \theta}C(R|y_0, \theta) \left(\frac{\partial}{\partial R}C(R|y_0, \theta) \right)^{-1}.$$

This solve can be accomplished and the correct partial derivative will be used in Stan's HMC implementation by simply passing the function $C(t)$ to the algebraic solver available in Stan which uses a modified version of Newton's method to find the solution of nonlinear systems of equations (Margossian 2018) .

Because Newton Iterations may diverge with a bad starting point, and at the time of this writing Stan's algebraic solver does not support variable initial guesses, we opted to implement a custom C++ solver based off of the bisection method. For a tutorial on using custom C++ functions and their gradients in Stan models see (Bales and Petzold 2018) .

Obtaining the Max of Our Spline Function

Note that the MA, or maximum amplitude TEG values also requires a nonlinear solve to obtain. To compute this value we also use custom C++ code based off of the bisection method on the derivative of the function. In this case the appropriate gradient can be computed similarly.

Testing our Code on the ODE Test Equation

We first try out our code for inferring unknowns using hitting times on the simple ODE test equation:

$$\frac{dy}{dt} = -\lambda y.$$

with initial value $y(0) = 1$ and $\lambda = 1$. The solution of this equation is simply $y(t) = e^{-t}$, which hits the value $y(t) = 0.6$ at $t = -\ln 0.6 \approx 0.5108$. We set up a Stan model to infer the value of the initial condition given the value of the hitting time.

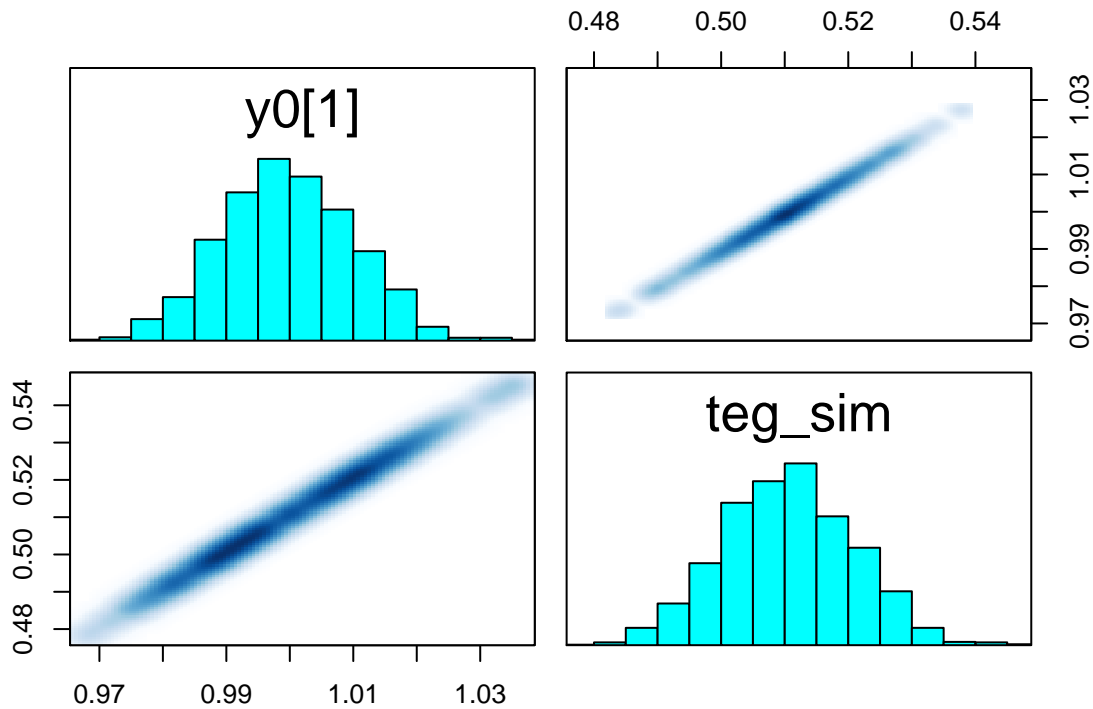
```
library(rstan)
```

```
## Loading required package: StanHeaders
```



```
##
## Samples were drawn using NUTS(diag_e) at Sun Jun 3 13:47:44 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
pairs(test.eq.stanfit, pars = c("y0", "teg_sim"))
```



```
# compare prior to posterior
```

```
test.eq.stan.samples <- rstan::extract(test.eq.stanfit, pars = c("y0", "teg_sim"))
```

```
tibble(y0.prior = rnorm(2000, 1, 0.1), y0.posterior = test.eq.stan.samples$y0[,1]) %>%
```

```
  gather(Distribution, Value) %>%
```

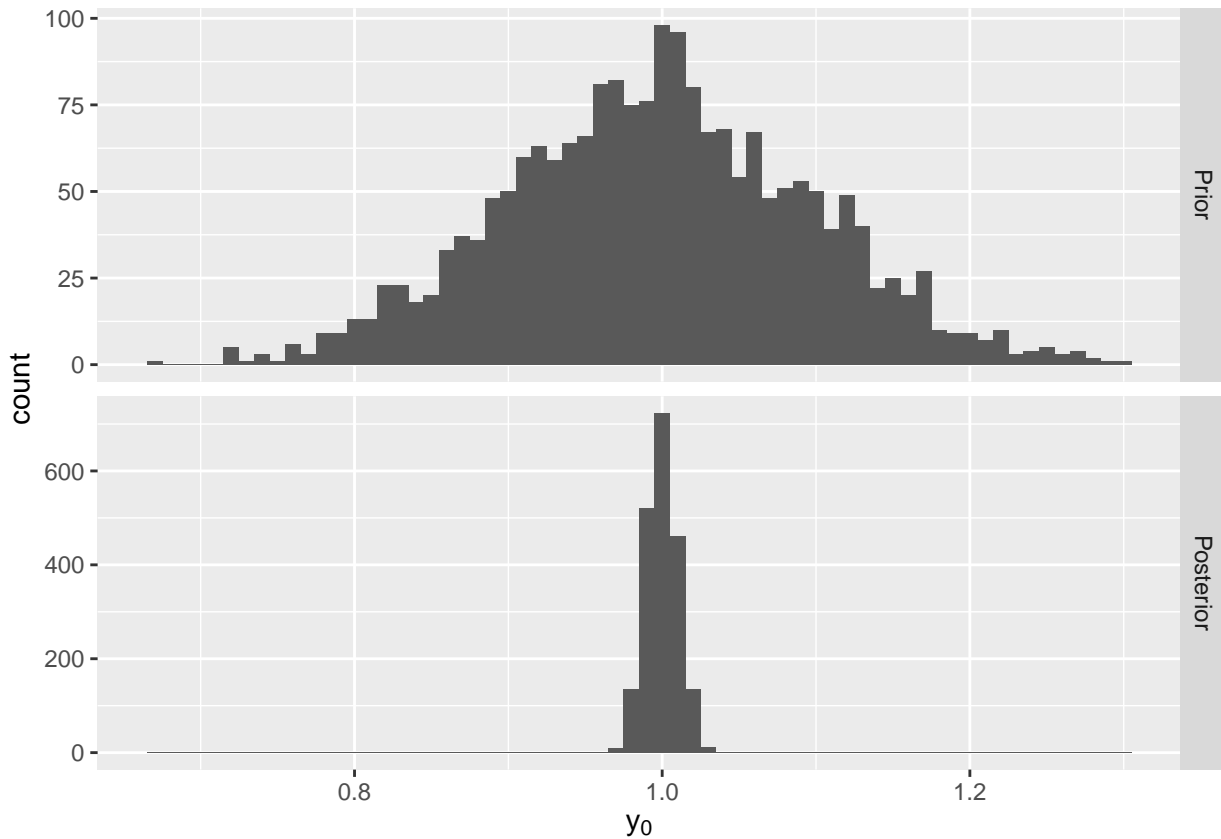
```
  mutate(Distribution = factor(Distribution, levels = c("y0.prior", "y0.posterior"), labels = c("Prior"
```

```
ggplot(aes(Value)) +
```

```
  geom_histogram(binwidth = 0.01) +
```

```
  facet_grid(Distribution ~ ., scales = "free") +
```

```
  xlab(expression(y[0]))
```



Putting it All Together to Infer Protein Concentrations Using TEG Data

Now that we know our Stan code works on a simple problem, it's time to try it out on some real trauma data. We will try out our model on the following trauma patient for whom we have TEG values for and initial protein values for all the proteins in our model except for *tPA* and *PAI*. Our goal here will be to infer this patient's initial concentration of *tPA* given their TEG measurements. In theory this should be possible because *tPA* is a protein primarily responsible for clot breakup and the Ly30 measurement of TEG measures the amount of clot breakup after 30 minutes. In this case, the patient has an unusually high Ly30 value of 1.7, indicating that 1.7% of their clot already broke up after only 30 minutes. In light of this, we should expect this patient to have a higher than average *tPA* concentration.

```
patient.data <- tibble(sex = "Male", age = 22, inj.mech = "StabWound",
  FII = 90, AT = 115, Fg = 162*29.41*1e-9, tPA = NA, PAI = NA,
  R = 0.8, K = 1.5, MA = 60.2, Ly30 = 1.7)
```

```
patient.data
```

```
## # A tibble: 1 x 12
##   sex   age inj.mech   FII   AT      Fg tPA   PAI     R     K     MA
##   <chr> <dbl> <chr>   <dbl> <dbl> <dbl> <lg1> <lg1> <dbl> <dbl> <dbl>
## 1 Male   22. StabWound  90.  115.  4.76e-6 NA    NA    0.800  1.50  60.2
## # ... with 1 more variable: Ly30 <dbl>
```

Choosing Priors for Unknown Protein Concentrations and Model Parameters

Recall that in our mechanistic model every patient has “cascade delay” parameters b and c that describe the specific state of the coagulation cascade. Using our four pieces of TEG data, we must infer these two parameters and also the value of the unknown protein concentrations tPA and PAI for our patient. To do this reasonably, it helps to incorporate any prior knowledge we have about these parameters. For the protein concentrations, we set the prior distributions to exponential distributions with respective means $4e-10$ and $9.3e-10$. These are the distributions of these protein concentrations for general trauma patients. With the information given it’s reasonable to assume our trauma patient’s protein values are drawn from the distribution of protein values for trauma patients. For b and c , we use weakly informative priors that are representative of a wide variety of possible coagulation profiles that we would expect to see in trauma patients.

With our priors set, we are ready to fit our model, integrating our prior knowledge and our data to produce a posterior distribution of this patient’s tPA values:

```
# format data for Stan
times <- c(seq(0, 600, by = 6), seq(660, 1800, by = 60))

proteins <- patient.data %>% select(FII, AT, Fg, tPA, PAI) %>% as.matrix %>% as.vector
proteins <- ifelse(is.na(proteins), -1, proteins)

teg <- patient.data %>% select(R, K, MA, Ly30) %>% as.matrix %>% as.vector

num_missing <- c(0,0,0,1,1)

dat <- list(Nt = length(times), ts = times, proteins = proteins, teg = teg, num_missing = num_missing)

# compile Stan model with custom C++ code
teg.cpp.src <- stanc("stan/fit_teg.stan", allow_undefined = TRUE)$cppcode
teg.stan.model <- stan_model("stan/fit_teg.stan", allow_undefined = TRUE,
                             includes = paste0('\n#include "', file.path(getwd(), 'src/cubic_spline_solvers

## In file included from /home/arya/R/x86_64-pc-linux-gnu-library/3.4/BH/include/boost/config.hpp:39:0,
## from /home/arya/R/x86_64-pc-linux-gnu-library/3.4/BH/include/boost/math/tools/config.hpp:39:0,
## from /home/arya/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/math/rev/core.hpp:39:0,
## from /home/arya/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/math/rev/core.hpp:39:0,
## from /home/arya/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/math/rev/core.hpp:39:0,
## from /home/arya/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/math.hpp:39:0,
## from /home/arya/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/src/stan/model/model.hpp:39:0,
## from filebcae762fb880.cpp:8:
## /home/arya/R/x86_64-pc-linux-gnu-library/3.4/BH/include/boost/config/compiler/gcc.hpp:186:0: warning
## # define BOOST_NO_CXX11_RVALUE_REFERENCES
## ^
## <command-line>:0:0: note: this is the location of the previous definition

# fit model
teg.fit <- sampling(teg.stan.model, chains = 1, iter = 1000, data = dat,
                   control = list(adapt_delta = 0.99, max_treedepth = 8), refresh = 100,
                   init = list(list(FII_missing = rep(72.0, num_missing[1]),
                                   AT_missing = array(82.0, num_missing[2]),
                                   Fg_missing = rep(5.5e-06, num_missing[3]),
                                   tPA_missing = array(1.6e-10, num_missing[4]),
                                   PAI_missing = array(3.4e-10, num_missing[5]),
                                   theta = c(3.0, 0.03))))
```

```

##
## SAMPLING FOR MODEL 'fit_teg' NOW (CHAIN 1).
##
## Gradient evaluation took 0.00296 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 29.6 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:  1 / 1000 [ 0%] (Warmup)
## Iteration: 100 / 1000 [ 10%] (Warmup)
## Iteration: 200 / 1000 [ 20%] (Warmup)
## Iteration: 300 / 1000 [ 30%] (Warmup)
## Iteration: 400 / 1000 [ 40%] (Warmup)
## Iteration: 500 / 1000 [ 50%] (Warmup)
## Iteration: 501 / 1000 [ 50%] (Sampling)
## Iteration: 600 / 1000 [ 60%] (Sampling)
## Iteration: 700 / 1000 [ 70%] (Sampling)
## Iteration: 800 / 1000 [ 80%] (Sampling)
## Iteration: 900 / 1000 [ 90%] (Sampling)
## Iteration: 1000 / 1000 [100%] (Sampling)
##
## Elapsed Time: 91.924 seconds (Warm-up)
##                79.0762 seconds (Sampling)
##                171 seconds (Total)

```

Our MCMC diagnostics all pass, and we are ready to examine the posterior distribution of this patient's *tPA* concentration. Sure enough, this patient's *tPA* concentration is higher than the population average, which we used as our prior distribution. This is reflective of their relatively high Ly30 value, just as we expected.

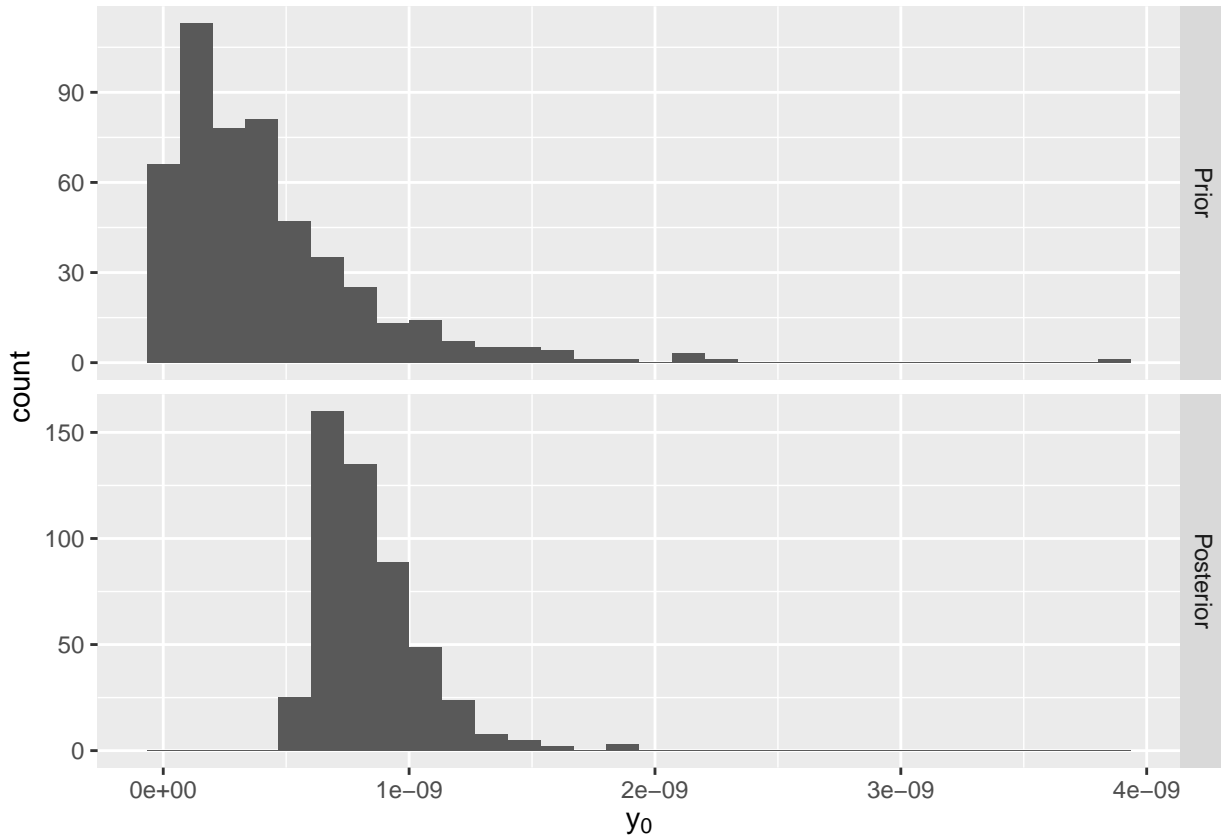
```

teg.stan.samples <- rstan::extract(teg.fit, pars = c("tPA_missing"))

tibble(tPA.prior = rexp(500,1/(4e-10)), tPA.posterior = teg.stan.samples$tPA_missing[,1]) %>%
  gather(Distribution, Value) %>%
  mutate(Distribution = factor(Distribution, levels = c("tPA.prior", "tPA.posterior"), labels = c("Prior", "Posterior")))
ggplot(aes(Value)) +
  geom_histogram() +
  facet_grid(Distribution ~ ., scales = "free") +
  xlab(expression(y[0]))

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

```



Acknowledgements

Research reported in this publication was performed by the Systems Biology Coagulopathy of Trauma Program of the US Army Medical Research and Materiel Command under award number W911QY-15-C-0026.

References

- Bales, Tresa Pollock, Brent Goodlet, and Linda Petzold. 2018. “Bayesian Estimation of Mechanical Elastic Constants.” https://github.com/stan-dev/stancon_talks/blob/master/2018/Contributed-Talks/09_bales/rus.Rmd.
- Brohi, Karim, Jasmin Singh, Mischa Heron, and Timothy Coats. 2003. “Acute traumatic coagulopathy.” *The Journal of Trauma* 54: 1127–30. doi:10.1097/ACO.0b013e3283509675.
- Carpenter, Bob. 2018. “Predator-Prey Population Dynamics: The Lotka-Volterra Model in Stan.” <http://mc-stan.org/users/documentation/case-studies/lotka-volterra-predator-prey.html>.
- Gonzalez, E, EE Moore, HB Moore, MP Chapman, CC Silliman, and A Banerjee. 2014. “Trauma-Induced Coagulopathy: An Institution’s 35 Year Perspective on Practice and Research.” *Scandinavian Journal of Surgery* 103 (2). SAGE Publications Sage UK: London, England: 89–103.
- Hoyert, D, and Jiaquan Xu. 2012. “National Vital Statistics Reports: Deaths: Preliminary Data for 2011.” *National Vital Statistics Reports* 61 (6). <http://scholar.google.com/scholar?hl=en/&btnG=Search/&q=intitle:National+Vital+Statistics+Reports+Deaths+:++Preliminary+Data+for+2011/#4>.
- Margossian, Charles. 2018. “Computing Steady States with Stan’s Nonlinear Algebraic Solver.” https://github.com/stan-dev/stancon_talks/blob/master/2018/Contributed-Talks/08_margossian/script/

AlgebraSolver.Rmd.

Margossian, Charles, and Bill Gillespie. 2017. “Differential Equations Based Models in Stan.” https://github.com/stan-dev/stancon_talks/blob/master/2017/Contributed-Talks/05_margossian/ODEStan.Rmd.

Mitrophanov, Alexander Y, Alisa S Wolberg, and Jaques Reifman. 2014. “Kinetic Model Facilitates Analysis of Fibrin Generation and Its Modulation by Clotting Factors: Implications for Hemostasis-Enhancing Therapies.” *Molecular BioSystems* 10 (9). Royal Society of Chemistry: 2347–57.

Quarteroni, Alfio, Riccardo Sacco, and Fausto Saleri. 2010. *Numerical Mathematics*. Vol. 37. Springer Science & Business Media.

Sagar, Adithya, and Jeffrey D Varner. 2015. “Dynamic Modeling of the Human Coagulation Cascade Using Reduced Order Effective Kinetic Models.” *Processes* 3 (1). Multidisciplinary Digital Publishing Institute: 178–203.