Flexible Modeling of Alzheimer's Disease Progression with I-Splines

Arya A. Pourzanjani, Benjamin B. Bales, Linda R. Petzold[*], *Michael Harrington*[†]

Alzheimer's Disease Progression

The progression of Alzheimer's Disease (AD) is characterized by the gradual deterioration of biomarkers and eventual loss of basic memory and decision-making functions (Figure 1). Using these biomarker values and other tests to estimate how far an individual has progressed in the disease is valuable in diagnosis as well as in assessing the efficacy of interventions. Additionally, prediction of how the individual will continue to progress is critical in decision making. While it is known that AD only gets worse over time, it is believed that patients with the disease progress at different rates and at different stages of their lives. There is no standard path of progression for people with the disease, which makes estimation of disease severity and future progression difficult. In addition to estimating these paths for an individual given their measurements, it is of clinical and biological significance to be able to understand the order in which certain biomarkers begin to deteriorate and what their distribution might look like for various stages of the disease.

To accomplish these tasks, we introduce a probabilistic model that includes a latent, monotonically increasing variable that measures the continually worsening progression of AD in an individual given their biomarker values. To flexibly model this monotonically increasing progression, we used a basis of monotonic functions, known as I-splines. The model also describes how, and in what order the biomarker values deteriorate over the progression of AD. We illustrate how we fit the model in Stan to real patient data to realistically capture the progression of AD.

Data and Exploratory Analysis

To simplify our notebook, we used a pre-cleaned tibble of biomarker data from the publicly available Alzheimer's Disease Neuroimaging Initiative (ADNI) dataset. The tibble (tidyverse dataframe) contains various biomarker measurements for 1,737 patients, as well as their age at the time of measurement.

```
library(tidyverse)
library(dplyr)

#load pre-cleaned tibble and preview
load("data/adni_precleaned.Rdata")
adni %>% sample_n(10)
```

```
## # A tibble: 10 x 6
##      RID PTGENDER   AGE DX              BIOMARKER   VALUE
##    <int> <chr>    <dbl> <fct>           <chr>       <dbl>
##  1   204 Female    72.1 MCI to Dementia TAU        71.6
##  2  4928 Male      77.8 MCI             TAU       103.
##  3  2274 Male      66.0 MCI             MMSE       25.0
##  4  2109 Male      77.2 MCI             HIPPO       0.601
##  5  4750 Female    84.7 MCI to NL       HIPPO       0.648
##  6  4022 Male      86.4 MCI             TAU       140.
##  7   478 Male      56.6 MCI             HIPPO       0.695
```

[*]University of California, Santa Barbara
[†]Huntington Medical Research Institute

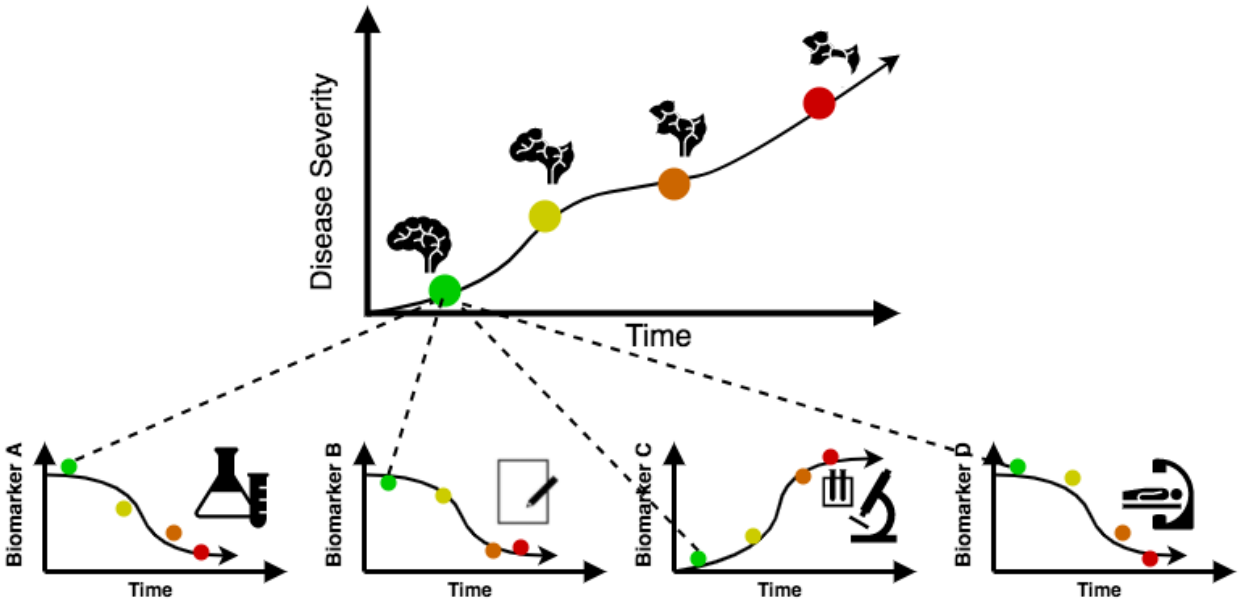


Figure 1: **Figure 1: Any patient test, such as a blood test, written exam, or MRI, can constitute a biomarker. As AD continuosly progresses, various biomarkers begin to degrade in unison, usually following an S-like pattern where degradation is initially slow, then rapidly progresses, then eventually levels out.**

```
##  8   378 Female    70.0 MCI             ABETA     149.
##  9   352 Female    82.0 NL              MMSE       30.0
## 10  4894 Female    61.7 Dementia        MMSE       18.0
```

The biomarkers measured include the following:

- **ABETA**: Concentration of the amyloid-beta protein in cerebrospinal fluid (CSF), measured in pg/ml. In healthy individuals this harmful protein is actively cleared from the brain in to the CSF. In individuals with AD, the amyloid-beta protein concentrates in the brain to form harmful plaques. Low levels in the CSF indicate the protein is not being cleared from the brain, and is thus an indication of AD.
- **HIPPO**: Volume of the individual's hippocampal brain region as measured by MRI and normalized to their baseline brain volume. In AD the hippocampus is known to shrink in size as the disease progresses. Volume is typically estimated using software applied to images like Figure 2.
- **MMSE**: Mini-Mental State Examination (MMSE) test. A 30-point questionnaire administered by psychologists to assess working memory and brain function in an individual. Patients with AD show lower scores in MMSE, but unfortunately lowered MMSE scores typically only show once the disease has progressed significantly.
- **TAU**: Concentration of tau protein in CSF, measured in pg/ml. Tau is a protein that shows up as a byproduct of dead neurons. High tau in the CSF indicates an abundance of dying neurons in the brain.

Currently, staging of AD is accomplished by the thorough review of a patient by a panel of expert doctors. While this diagnosis serves as a gold-standard, the diagnosis process is cumbersome, prone to subjectivity, and typically only includes three discrete levels, which does not represent the continuously progressing nature of AD. These three stages are in order of severity: Normal (NL), Mild Cognitive Impairment (MCI), and Dementia.

Population histograms of these biomarkers during the three stages of the disease reveal a monotonic pattern in the four biomarkers we are considering. For example ABETA appears to be a biomodal distribution where the lower mode becomes more common as the disease worsens. The distribution of HIPPO appears to shift to the left with increased worsening of AD. These observations lead us to believe that these biomarkers

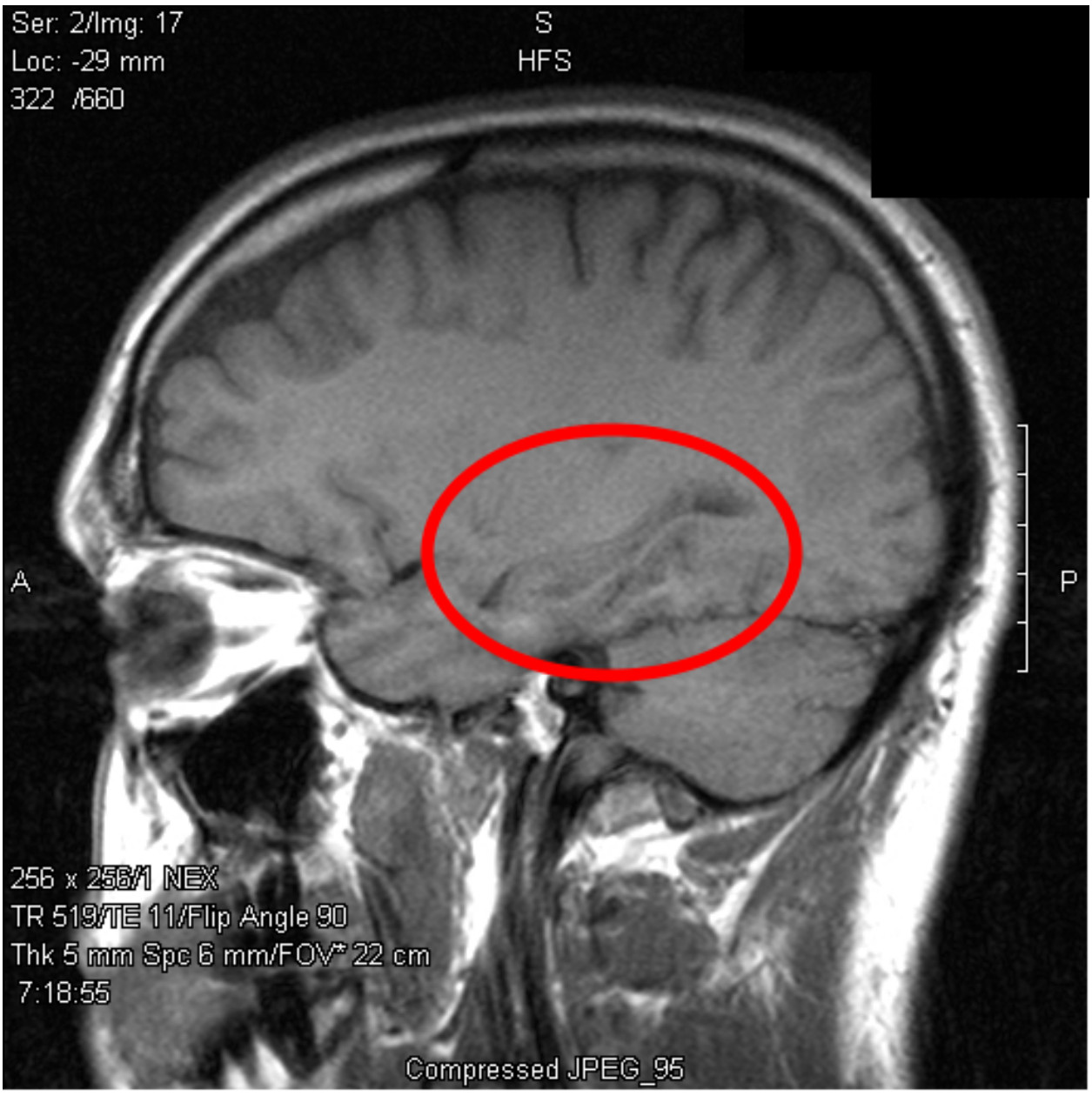
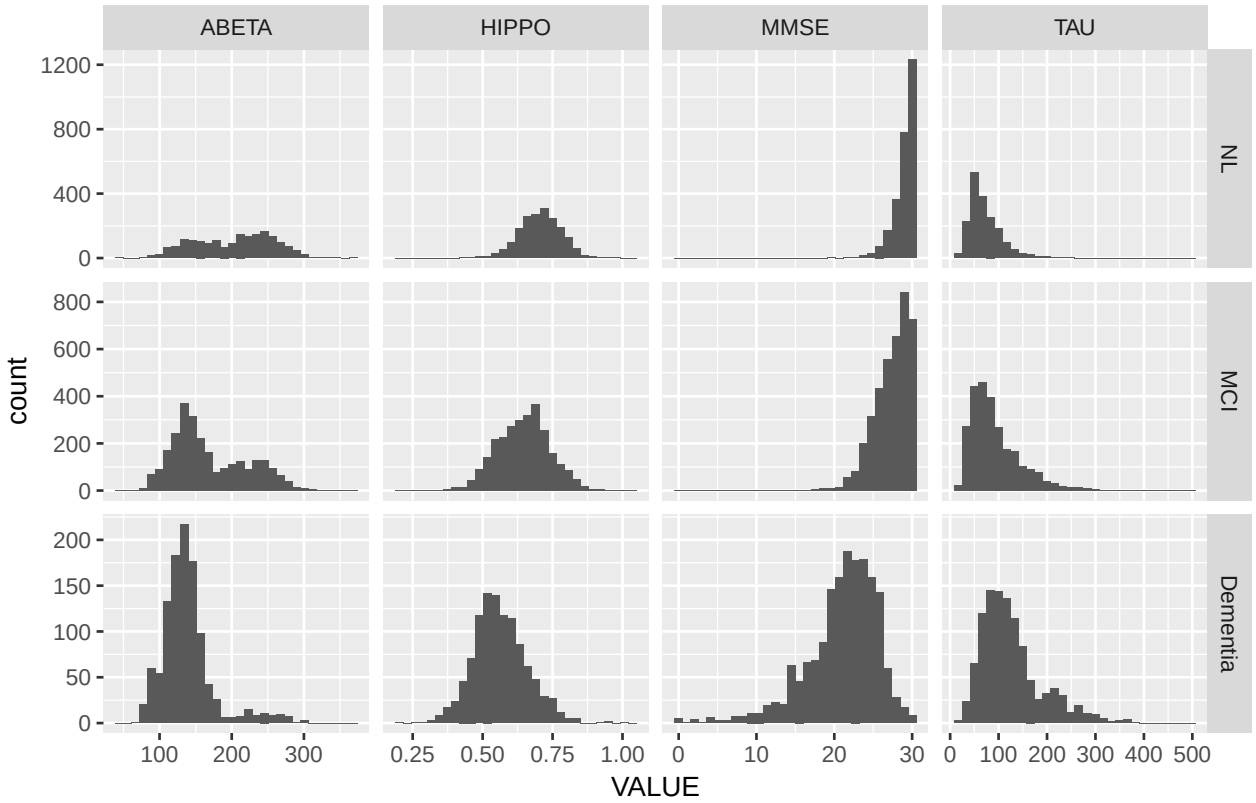


Figure 2: **Figure 2: Hippocampus of a graduate student, obtained by MRI.**

can be modeled as coming from a single continuous latent variable that we can interpret as being a disease progression score.

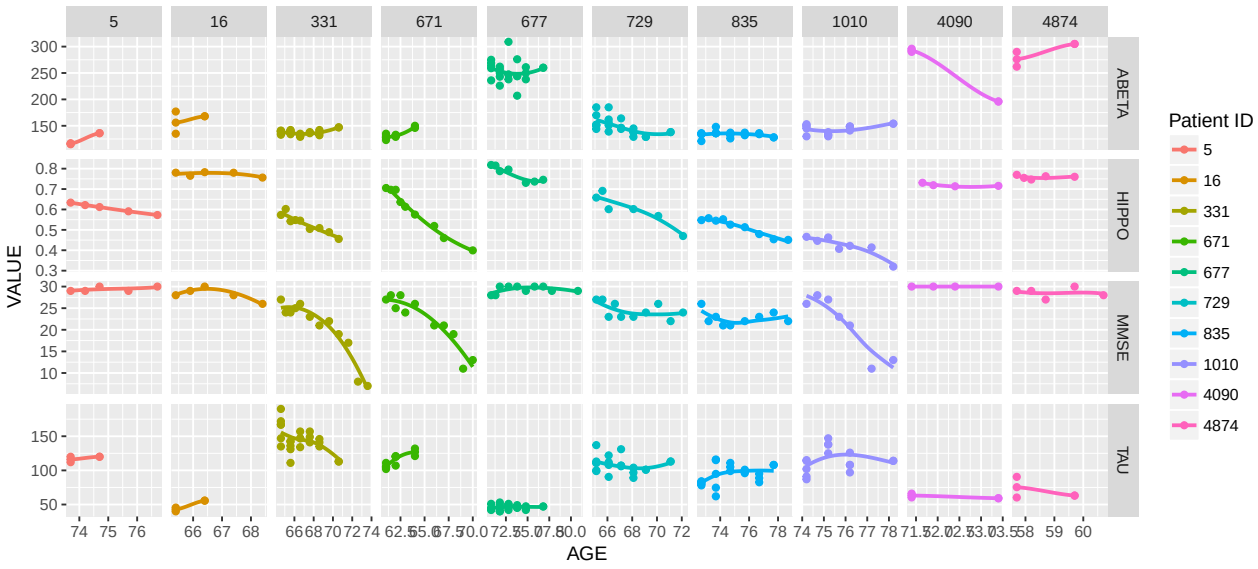
```
adni %>%
  filter(DX %in% c("NL", "MCI", "Dementia")) %>%
  ggplot(aes(VALUE)) +
  geom_histogram() +
  facet_grid(DX ~ BIOMARKER, scales = "free")
```

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



For clarity and graphical ease, we focus on 10 specific patients for the remainder of the notebook. We selected these patients to be of different genders, ages, and stages of AD. Plots of biomarkers generally reveal monotonic trends in time for each patient:

```
adni10 <- adni %>% filter(RID %in% c(331,729,835,671,1010,5,16,677,4090,4874))

adni10 %>%
  ggplot(aes(AGE, VALUE, color = as.factor(RID))) +
  geom_point() +
  geom_smooth(se = FALSE, span = 2) +
  facet_grid(BIOMARKER ~ RID, scales = "free") +
  scale_color_discrete(name = "Patient ID")
```



Existing Models of Alzheimer's Disease Progression

Existing models of AD progression utilize monotonic functions to capture the continual worsening of biomarkers. Lorenzi et al. (2017) use a monotonic Gaussian Process to model how each biomarker deteriorates over time in the population, and they use ordinary Gaussian Processes for each individual to describe random deviations from this general progression at the individual level. To achieve monotonicity in their Gaussian Processes they use the method proposed by Riihimäki and Vehtari (2010), and fit their model using Expectation Propagation (EP). B. M. Jedynak et al. (2012) take a different approach; they posit that the deterioration of individual biomarkers is tied to a single latent state for each patient that increases monotonically over time. This latent variable can thus be used as a disease progression score, representing how far in the disease an individual has progressed. For our analyses we start with this latter model, and add several improvements, including I-Splines.

Linear Latent Progression Model

While the model of B. M. Jedynak et al. (2012) was originally presented in a frequentist least-squares framework, we translated the model to a probabilistic generative model and implemented it in Stan. We then made several expansions to the model to more realistically fit the data, most notably the use of I-splines which we describe in the next section. If we let $y_{ij}(t)$ denote the jth biomarker for the ith patient at age t, the original model of B. M. Jedynak et al. (2012) can be translated to the following probabilistic generative model:

$$\begin{aligned} s_i(t) &= \alpha_i t + \beta_i \\ y_{ij}(t) &\sim \mathcal{N}(f(s_i(t) \,|\, a_j, b_j, c_j, d_j), \sigma_j) \end{aligned}$$

where σ_j represents the measurement variability for the jth biomarker and $f(\cdot|a_j, b_j, c_j, d_j)$ represents a biomarker-specific 4-parameter sigmoid-curve for the jth biomarker

$$f(s \,|\, a_j, b_j, c_j, d_j) = \frac{a_j}{1 + e^{-b_j s - c_j}} + d_j.$$

The parameters c_j and b_j are of particular importance because they indicate when a biomarker starts deteriorating and how fast it deteriorates once it does. The form of the degradation of the biomarkers takes

the form of a logistic (or S-curve) because in practice biomarkers like ABETA and HIPPO seem to get worse slowly at first, then rapidly, and eventually hit a saturation point. We note that one of the biomarkers has to have fixed values for b and c, to ensure identifiability of the parameters. We chose ABETA to have $b = 1$ and $c = 0$.

In this model, $s_i(t)$ is a monotonically increasing transformation of age that represents a continuous disease progression score for the ith individual. The individual specific parameters α_i and β_i determine the rate of progression and relative onset of AD respectively for the ith individual. To ensure the progression variable $s_i(t)$ is monotonic in time, α_i is constrained to be positive.

We note that in this model the parameters of individual biomarker evolution are not allowed to vary from individual to individual, only the latent disease progression score. This implies that the degradation of the biomarkers all happen in the same relative order and relative rate in all patients, an assumption that would have to be properly verified by examining the relative rates of deterioration in individuals.

We implemented this model in Stan to obtain a baseline model. We note that in the original work of B. M. Jedynak et al. (2012), each patient in the model required at least two observations at different time points so that the parameter α_i was identifiable. In our Stan model, we instead rely on a hierarchical prior over α_i so that we can more flexibly fit those patients with few and even only one observation. We think the hierarchical normal prior is a valid modeling choice because we expect individual progressions to not vary too widely from one another, i.e. the distribution should not be heavy-tailed. Furthermore, human population characteristics are often normally distributed, and we have no reason to believe the progressions parameters are multi-modal. This involves the following extension to the model:

$$\gamma \sim \text{HalfNormal}(0, 10)$$
$$\alpha_i \sim \text{HalfNormal}(0, \gamma).$$

We use a weakly informative scale parameter of 10 in the hyperparameter, as we do not have much prior information on what the scale of the α_i parameter should be.

Fitting the Model in Stan

```
library(rstan)
```

```
Loading required package: StanHeaders
```

```
rstan (Version 2.17.3, GitRev: 2e1f913d3ca3)
```

```
For execution on a local, multicore CPU with excess RAM we recommend calling
options(mc.cores = parallel::detectCores()).
To avoid recompilation of unchanged Stan programs, we recommend calling
rstan_options(auto_write = TRUE)
```

```

Attaching package: 'rstan'
```

```
The following object is masked from 'package:tidyr':

    extract
```

```
dat <- list(N = adni10 %>% pull(RID) %>% unique %>% length,
            K = 4,
            tot_obs = nrow(adni10),
            patient_idx = adni10 %>% pull(RID) %>% as.factor %>% as.integer,
            biomarker_idx = adni10 %>% pull(BIOMARKER) %>% as.factor %>% as.integer,
            age = adni10 %>% pull(AGE) %>% (function(x) x - mean(x)),
```

```
            y = adni10 %>% pull(VALUE))

linear_progression_fit <- stan("stan/linear_progression.stan", data = dat, chains = 1, iter = 2000,refre
```

```
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.hp
                 from /home/arya/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/math/rev/co
                 from /home/arya/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/math/rev/co
                 from /home/arya/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/math/rev/co
                 from /home/arya/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/math/rev/ma
                 from /home/arya/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/math.hpp:4,
                 from /home/arya/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/src/stan/model/m
                 from fileb4a46c25c1af.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

SAMPLING FOR MODEL 'linear_progression' NOW (CHAIN 1).

Gradient evaluation took 0.000153 seconds
1000 transitions using 10 leapfrog steps per transition would take 1.53 seconds.
Adjust your expectations accordingly!


Iteration:    1 / 2000 [  0%]  (Warmup)
Iteration: 1000 / 2000 [ 50%]  (Warmup)
Iteration: 1001 / 2000 [ 50%]  (Sampling)
Iteration: 2000 / 2000 [100%]  (Sampling)

 Elapsed Time: 17.8026 seconds (Warm-up)
               15.8313 seconds (Sampling)
               33.6338 seconds (Total)
```

```
linear_progression_fit %>% print(pars = c("a_abeta", "a_hippo", "a_tau", "d_abeta", "d_hippo", "d_tau",
```

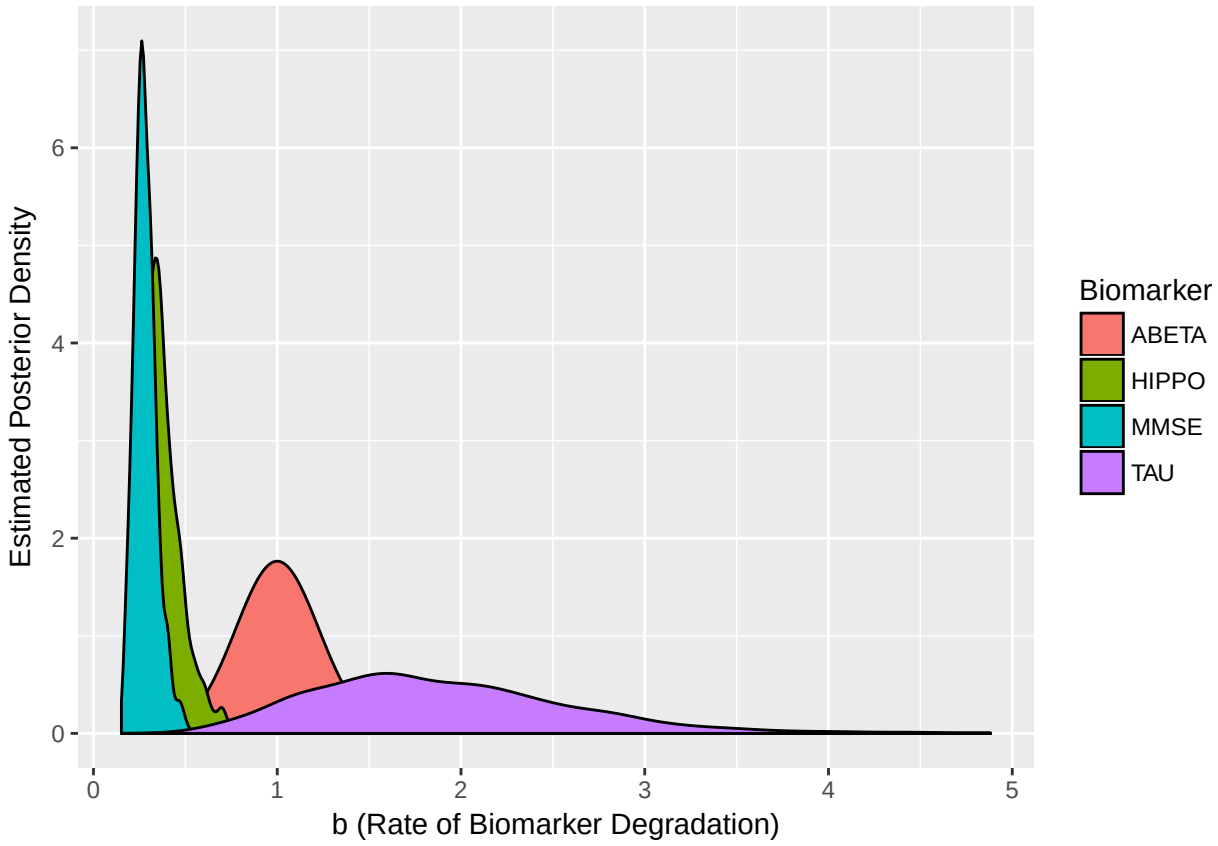
```
Inference for Stan model: linear_progression.
1 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=1000.

              mean se_mean   sd    2.5%     25%     50%     75%   97.5%
a_abeta     -109.46    0.03 0.88 -111.18 -110.06 -109.45 -108.85 -107.78
a_hippo       -0.39    0.00 0.03   -0.47   -0.41   -0.39   -0.37   -0.34
a_tau         63.15    0.13 3.52   56.67   60.82   63.05   65.39   69.98
d_abeta      247.00    0.03 0.90  245.20  246.39  246.97  247.65  248.70
d_hippo        0.76    0.00 0.01    0.75    0.76    0.76    0.77    0.78
d_tau         52.50    0.11 2.93   46.76   50.49   52.60   54.53   58.31
b_hippo        0.38    0.01 0.10    0.23    0.31    0.36    0.43    0.63
b_mmse         0.28    0.00 0.06    0.18    0.24    0.28    0.32    0.43
b_tau          1.89    0.02 0.70    0.80    1.41    1.79    2.30    3.47
c_hippo        4.21    0.05 0.97    2.77    3.50    4.05    4.74    6.70
c_mmse         4.95    0.04 0.78    3.70    4.45    4.84    5.36    6.81
c_tau          8.49    0.14 4.00    2.97    5.47    7.71   10.85   18.44
sigma_abeta   17.98    0.04 1.16   15.82   17.17   17.95   18.80   20.25
```

```
sigma_hippo    0.03    0.00 0.00    0.02    0.03    0.03    0.03    0.04
sigma_mmse     2.35    0.01 0.23    1.96    2.18    2.33    2.50    2.80
sigma_tau     20.43    0.04 1.25   18.12   19.62   20.34   21.15   23.14
gamma          0.58    0.01 0.25    0.27    0.42    0.53    0.69    1.21
            n_eff Rhat
a_abeta      1000 1.00
a_hippo       639 1.00
a_tau         750 1.00
d_abeta      1000 1.00
d_hippo      1000 1.00
d_tau         756 1.00
b_hippo       243 1.00
b_mmse        228 1.00
b_tau         914 1.00
c_hippo       310 1.00
c_mmse        320 1.01
c_tau         807 1.00
sigma_abeta  1000 1.00
sigma_hippo  1000 1.01
sigma_mmse    882 1.01
sigma_tau    1000 1.00
gamma         356 1.00

Samples were drawn using NUTS(diag_e) at Sun Jun  3 13:05:55 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).
```

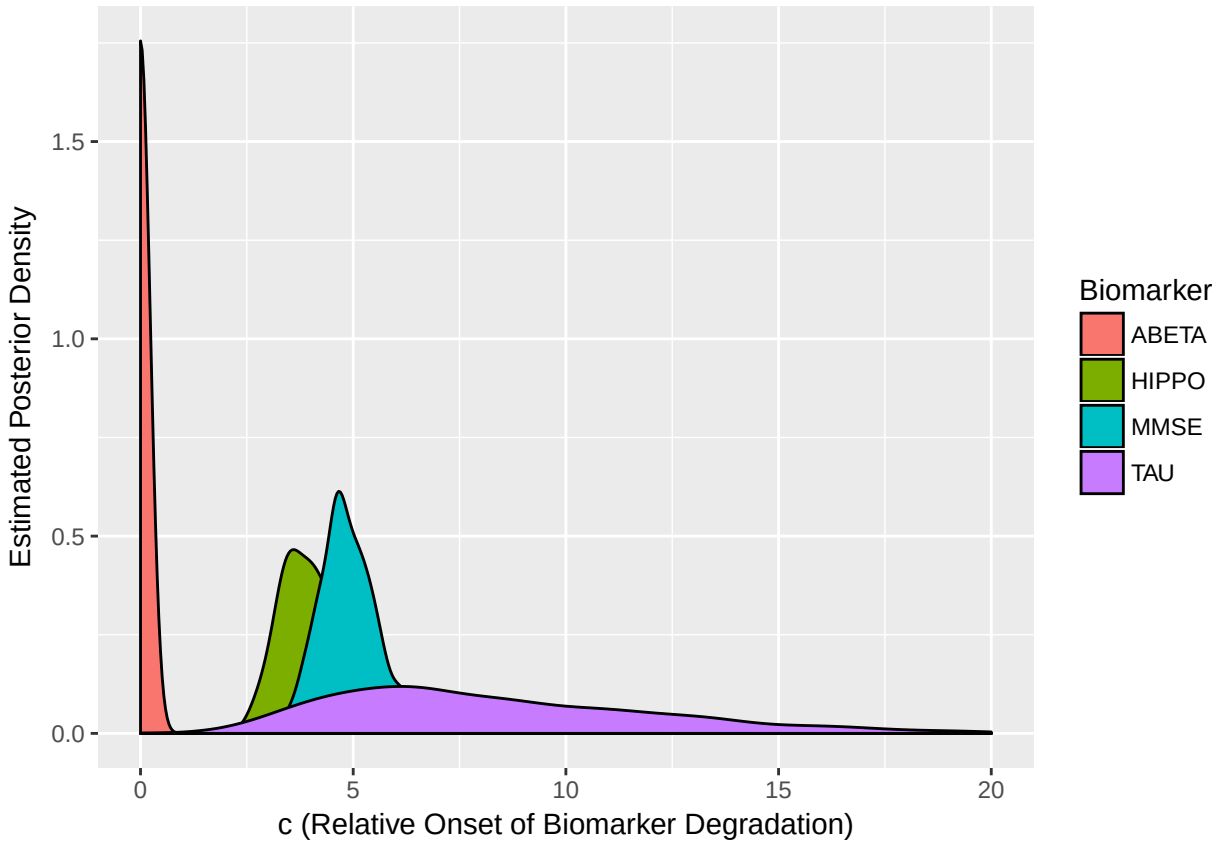
Posterior Analysis of Biomarker Comparisons

After ensuring reasonable Rhat values, and no divergences or other warnings, we see that the model fit reveals the relative order of biomarker degradation onset (c) and the speed of deterioration (b). From posterior histograms of c, we can see that relative to the onset of worsening of ABETA, HIPPO seems to start deteriorating about 3 years later, while MMSE starts to deteriorate about 5 years later. For TAU there is far too much posterior uncertainty to tell when the drop off starts. This could be due to model miss-specification in how TAU is modeled. While ABETA begins to deteriorate the earliest of the biomarkers, when HIPPO and MMSE begin to deteriorate, they seem to do so much faster, according to posterior histograms of b.

```
#extract posterior samples in to tibbles with the correct column names
posterior_samples_bc <- linear_progression_fit %>%
  rstan::extract(pars = c("b","c")) %>%
  map(as_tibble) %>%
  map(. %>% set_names(c("ABETA","HIPPO","MMSE","TAU"))) %>%
  map(. %>% gather(Biomarker, Value))

#create plots for B and C to compare
posterior_plots_bc <- posterior_samples_bc %>%
  map(function(s) s %>% ggplot(aes(Value, group = Biomarker, fill = Biomarker)) + geom_density())

#reveal plots
posterior_plots_bc$b + xlab("b (Rate of Biomarker Degradation)") + ylab("Estimated Posterior Density")
```



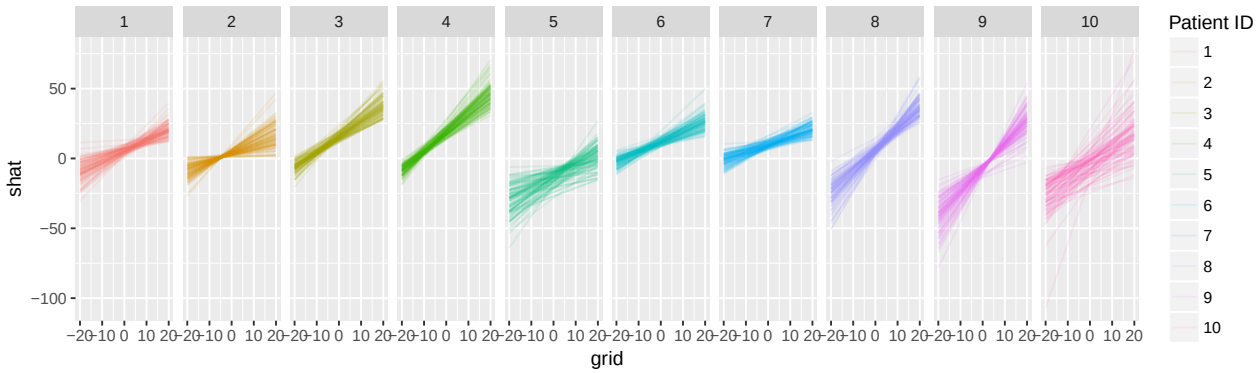

```
posterior_plots_bc$c + xlim(0,20) + xlab("c (Relative Onset of Biomarker Degradation)") + ylab("Estimat
```



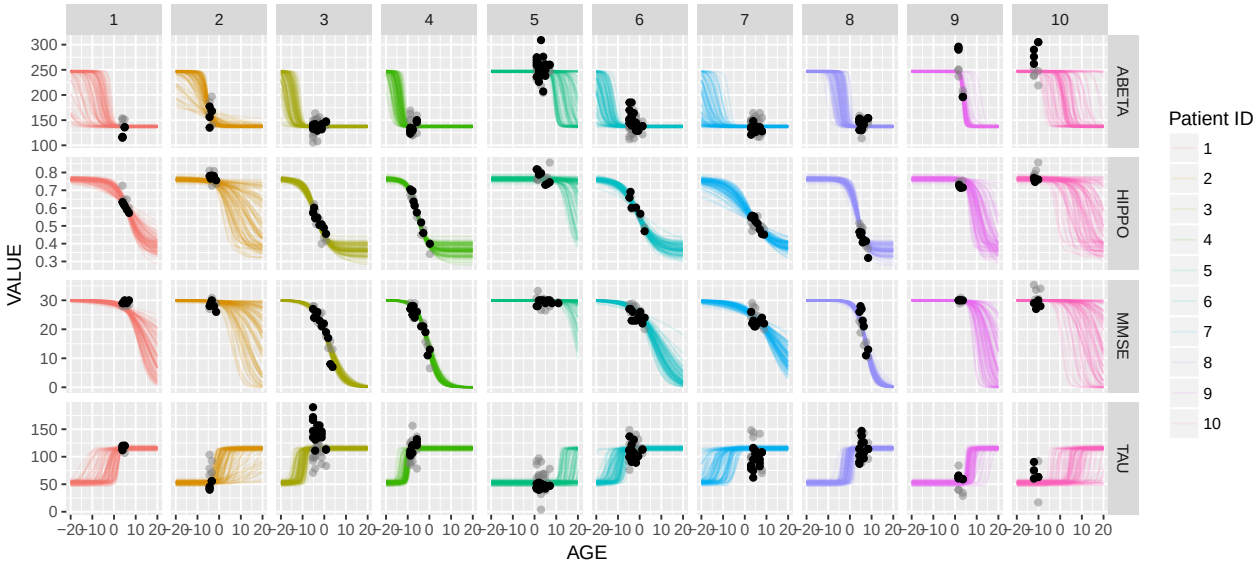
Posterior Analysis of Model Fit

With our Stan model fit, we can examine the posterior distribution of disease progression curves for individuals, as well as the degradation of their biomarkers over time. Note that we center the age by subtracting by the average age of 69.8. This is for numerical reasons.

```
library(stringr)

#extract generated posterior samples of disease progression curves for each patient
#(sample only 100 samples from the posterior for better plot visibility)
shat_posterior_samples <- (linear_progression_fit %>% rstan::extract(pars = "shat"))$shat[sample(1:1000

#create function and use map to put samples in to easily plottable tibble
sample_to_grid_s <- function(shat_sample) shat_sample %>% t %>% as_tibble %>%
  mutate(grid = seq(-20,20,0.5)) %>%
  gather(patient_idx,shat,-grid) %>%
  mutate(patient_idx = as.integer(str_extract(patient_idx,"[0-9]+")))

master_grid_s <- map(1:dim(shat_posterior_samples)[1], function(i) sample_to_grid_s(shat_posterior_samp

#plot once in tibble form
master_grid_s %>%
  ggplot(aes(grid, shat, group = sample_num, color = as.factor(patient_idx))) +
  geom_line(alpha = 0.1) +
  facet_grid(. ~ patient_idx, scale = "free") +
  scale_color_discrete(name = "Patient ID")
```




```
#extract posterior generated samples of biomarkers into tibble so we can plot it
fhat_posterior_samples <- (linear_progression_fit %>% rstan::extract(pars = "fhat"))$fhat[sample(1:1000
biomarker_names <- c("ABETA","HIPPO","MMSE","TAU")

sample_to_grid <- function(fhat_sample_i) map(1:dim(fhat_posterior_samples)[2], function(j) fhat_sample
  bind_rows

master_grid <- map(1:dim(fhat_posterior_samples)[1], function(i) sample_to_grid(fhat_posterior_samples[

#extract posterior predicted samples
yhat_posterior_samples <- (linear_progression_fit %>% rstan::extract(pars = "yhat"))$yhat[500,]

adni10 %>%
  mutate(yhat = yhat_posterior_samples) %>%
  mutate(patient_idx = as.integer(as.factor(RID))) %>%
  mutate(AGE = AGE - mean(AGE)) %>%
  ggplot(aes(AGE, VALUE, color = as.factor(patient_idx))) +
  geom_line(aes(grid, yhat, group = sample_num), alpha = 0.1, data = master_grid) +
  geom_point(aes(AGE,yhat),color = "grey50", alpha = 0.5) +
  geom_point(color = "black") +
  facet_grid(BIOMARKER ~ patient_idx, scales = "free") +
  scale_color_discrete(name = "Patient ID")
```



The linear fit does a reasonable job of capturing each individual's progression of biomarkers over time (posterior predictive points in grey). However, the linear disease progression model leaves much to be desired.

The model assumes that the rate of progression of the disease is constant, i.e. that patients' disease status continuously progress at the same rate. In reality, it is probably the case that there are plateaus of AD progression, where progression can slow down then later pick back up either due to endogenous or exogenous circumstances. To more flexibly represent disease progression curves we need to be able to flexibly model monotonic functions of age. For this we turn to I-Splines.

I-Splines

I-Splines are a flexible and adjustable set of monotone basis functions used to model monotonic functions (Ramsay 1988). By taking a linear combination of these functions, and constraining the coefficients to be positive, one can flexibly model a wide class of monotone functions. Similarly, one can model increasing functions with a range from 0 to 1, by taking a linear combination of the I-Splines where the coefficients are constrained to sum to one.

To our knowledge, the only other known method for flexibly modeling monotone functions are monotone Gaussian Processes (Riihimäki and Vehtari 2010), which were used by Lorenzi et al. (2017). This method relies on constraining the Gaussian Process to have a positive derivative at a set number of points, which then usually forces the function to monotone. While attractive, the method may not return true monotone functions and it is not clear how to select the points where the derivative should be positive.

M-Splines

Given a domain, and a set of nodes $t_1, \cdots, t_D$ on that domain, the I-Spline basis functions are obtained by integrating the piece-wise-defined M-Splines, themselves a set of spline functions defined on the same domain using the same nodes. M-Splines are defined recursively. The order one M-Spline functions are piece-wise constant functions. The ith M-Spline of order one is defined as

$$M_{i1}(x) := \begin{cases} \frac{1}{(t_{i+1}-t_i)}, & t_i \leq x < t_{i+1} \\ 0, & \text{otherwise} \end{cases}$$

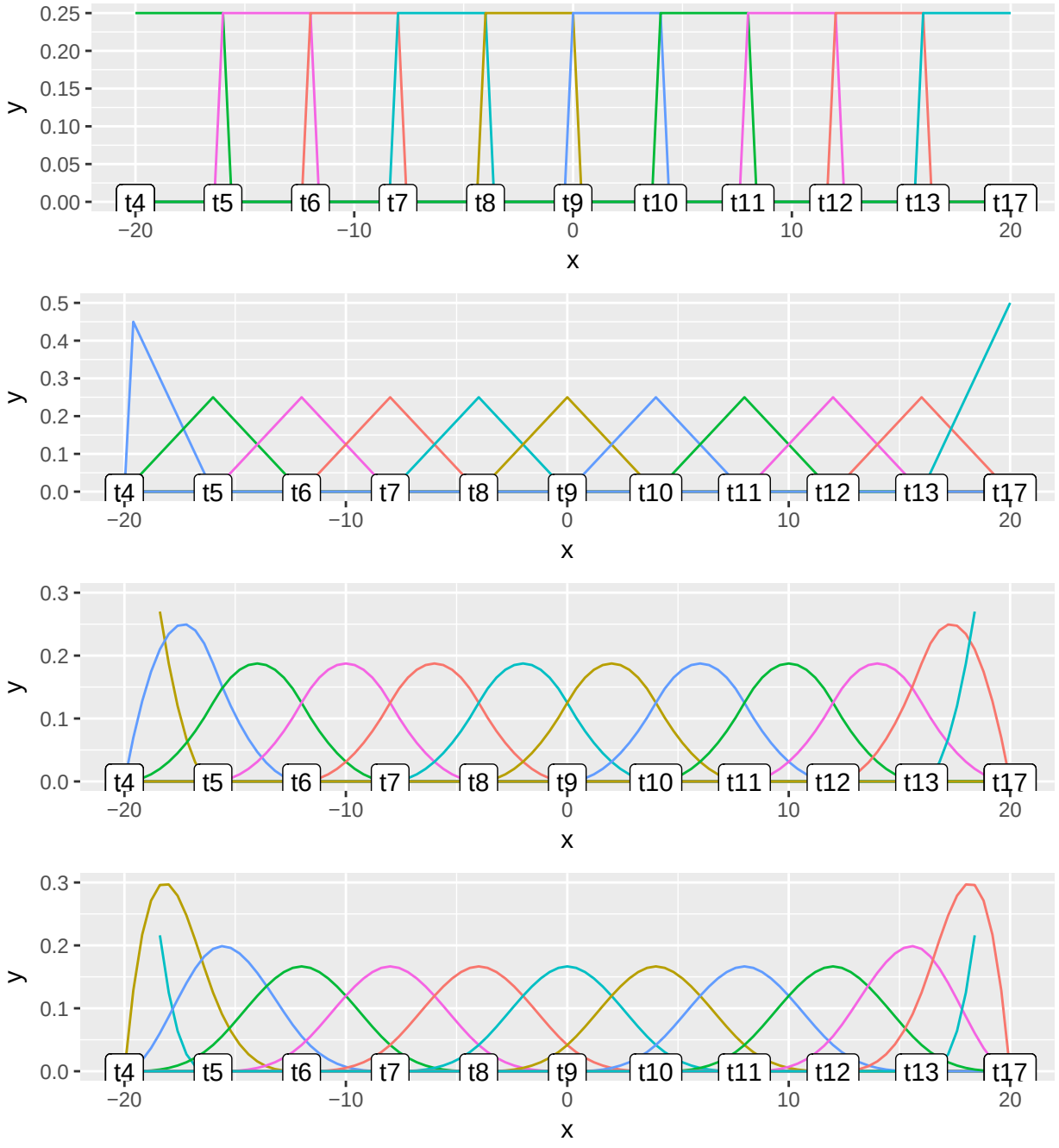
while the ith M-Spline of order k is defined recursively as

$$M_{ik}(x) := \left(\frac{k}{k-1}\right) \frac{(x-t_i)M_{i,k-1}(x) + (t_{i+k}-x)M_{i+1,k-1}(x)}{t_{i+k}-t_i}.$$

One can show that the M-Spline basis functions each integrate to one over the specified domain and are positive. We plot the M-Spline basis functions for orders $k = 1, 2, 3, 4$ using custom code written to generate M-Splines in R.

```
library(gridExtra)
source("R/isplines.R")

nodes <- list(t1=-20,t2=-20,t3=-20,t4=-20,t5=-16,t6=-12,t7=-8,t8=-4,t9=0,t10=4,t11=8,t12=12,t13=16,t14=

M_spline_expr <- get_order_4_M_spline_expr(K = 9, nodes = nodes)
M_spline_functions <- expr_to_R_functions(M_spline_expr,nodes = nodes)
M_spline_plots <- M_spline_functions %>% map(plot_function_list, nodes = nodes)

#adjust plots for better visibility
M_spline_plots[[3]] <- M_spline_plots[[3]] + ylim(0,0.3)
M_spline_plots[[4]] <- M_spline_plots[[4]] + ylim(0,0.3)
```

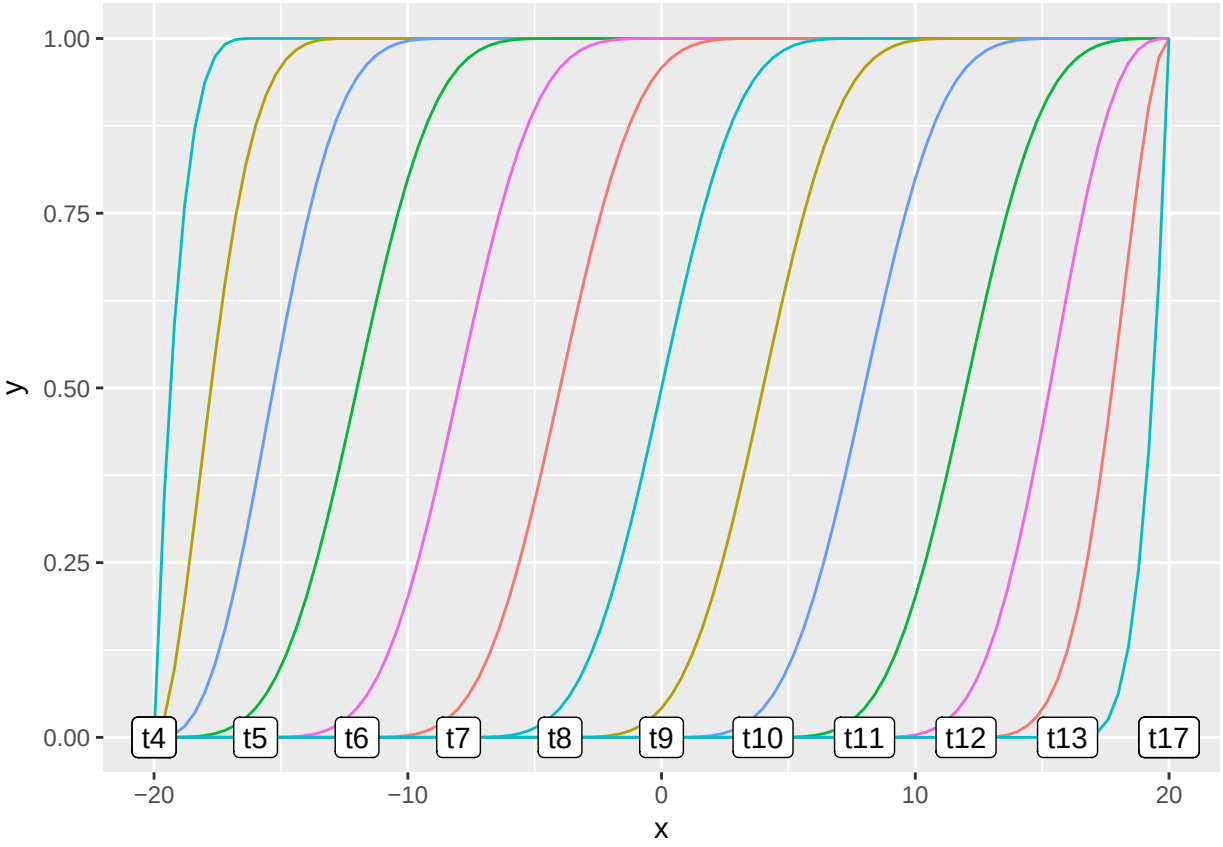
```
do.call(grid.arrange, c(M_spline_plots, nrow = 4))
```



Obtaining I-Splines from M-Splines

Because M-Splines integrate to one over the specified domain and are positive, their integral will be a set of functions that monotonically increase from 0 to 1 over the domain. Since the M-Splines are piece-wise polynomials, this integral is easy to compute. We illustrate the I-Splines that we computed using the Ryacas system for symbolic integration along with other custom code we wrote, available in the isplines.R file.

```
M4_expr <- M_spline_expr[[4]]
I_splines_functions <- M4_expr_to_R_I_spline(M4_expr, nodes)
plot_function_list(I_splines_functions, nodes)
```



Using I-Splines to Represent Monotonic Functions

Positive linear combinations of I-Splines will be monotone, while symplectic linear combinations will be monotone and range from 0 to 1. We illustrate the latter case, by taking different linear combinations of our $D = 13$ I-Splines over the domain, where the coefficients are drawn from a uniform Dirichlet distribution, and hence sum to one.

```
library(MCMCpack)
```

```
## Loading required package: coda
```

```
##
## Attaching package: 'coda'
```

```
## The following object is masked from 'package:rstan':
##
##     traceplot
```

```
## Loading required package: MASS
```

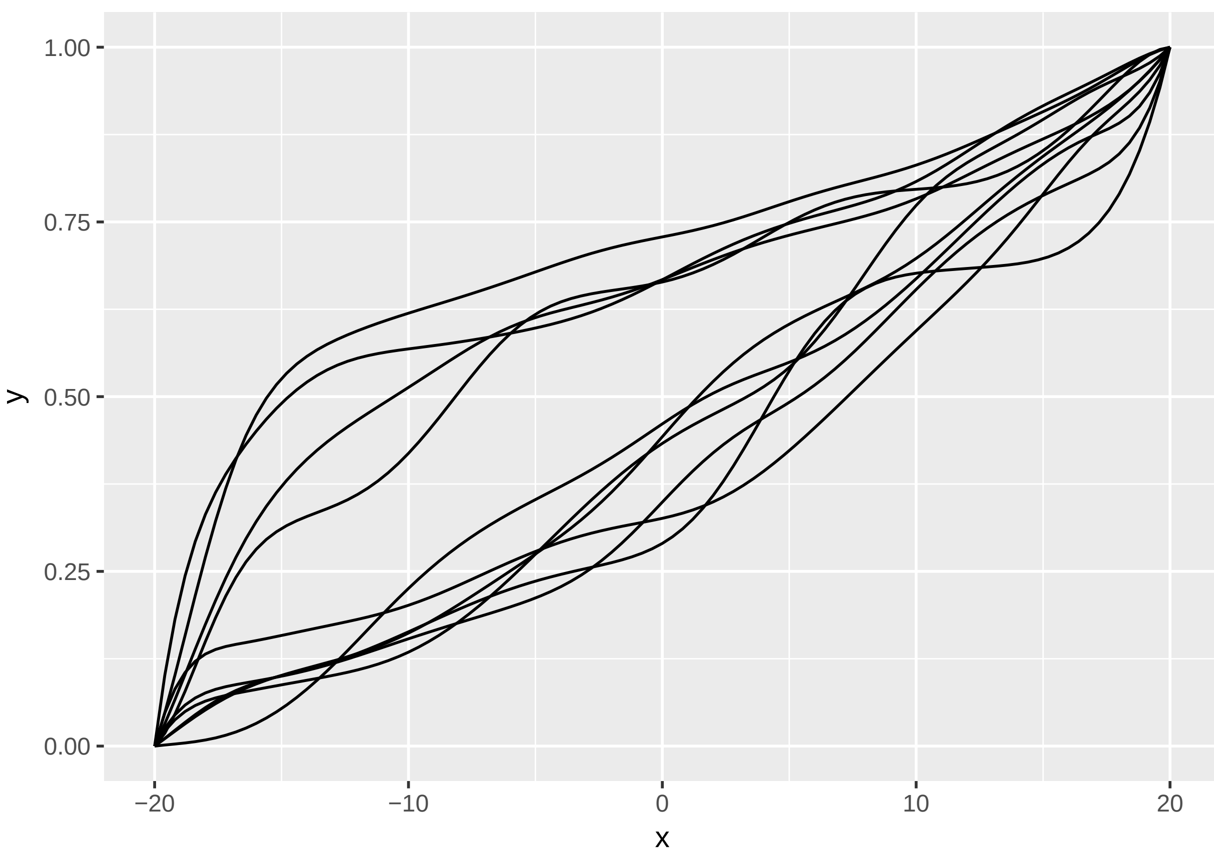
```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
```

```
##     select
```

```
## ##
## ## Markov Chain Monte Carlo Package (MCMCpack)
## ## Copyright (C) 2003-2018 Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park
## ##
## ## Support provided by the U.S. National Science Foundation
## ## (Grants SES-0350646 and SES-0350613)
## ##
```

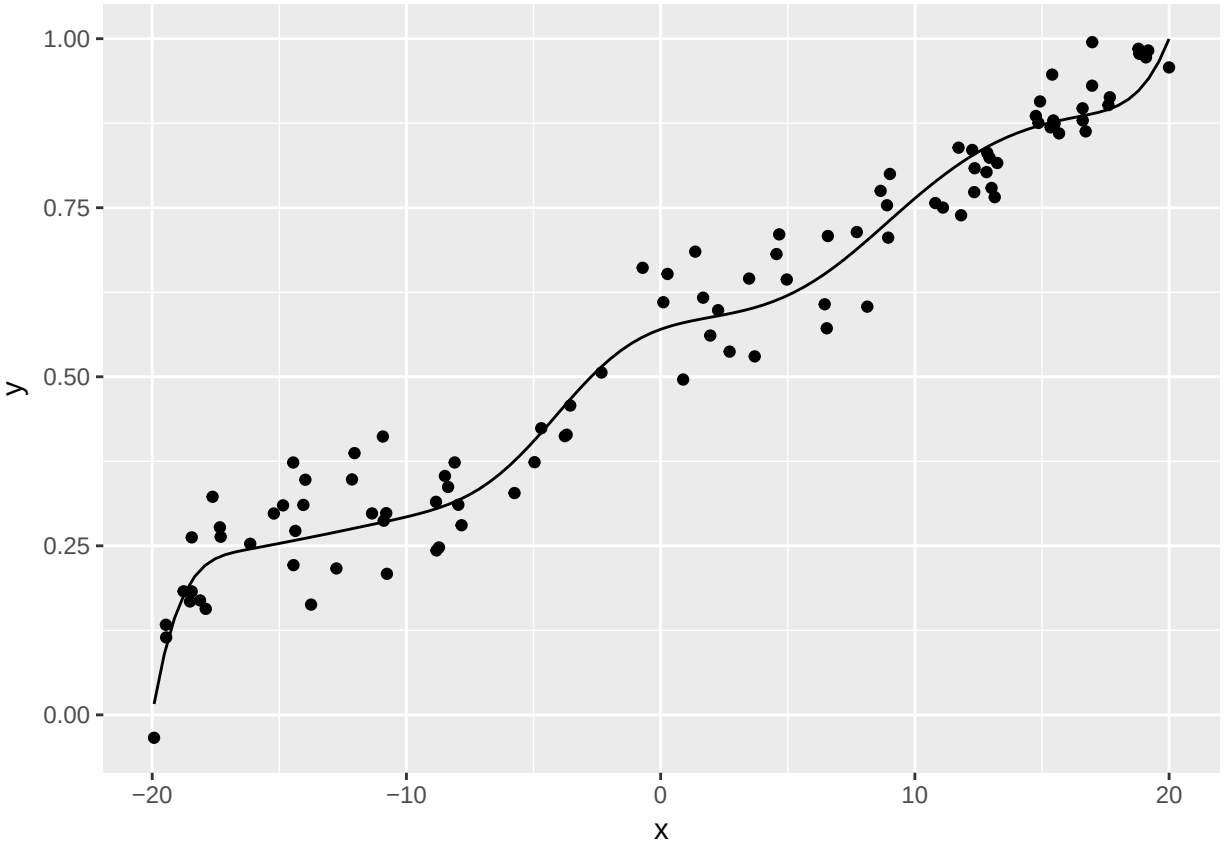
```
D <- length(I_splines_functions) #number of I-Spline Functions in our Basis
ggplot(aes(x), data = tibble(x = c(-20,20))) +
  stat_function(fun = convex_combination(rdirichlet(1, rep(1,D)), I_splines_functions)) +
  stat_function(fun = convex_combination(rdirichlet(1, rep(1,D)), I_splines_functions)) +
  stat_function(fun = convex_combination(rdirichlet(1, rep(1,D)), I_splines_functions)) +
  stat_function(fun = convex_combination(rdirichlet(1, rep(1,D)), I_splines_functions)) +
  stat_function(fun = convex_combination(rdirichlet(1, rep(1,D)), I_splines_functions)) +
  stat_function(fun = convex_combination(rdirichlet(1, rep(1,D)), I_splines_functions)) +
  stat_function(fun = convex_combination(rdirichlet(1, rep(1,D)), I_splines_functions)) +
  stat_function(fun = convex_combination(rdirichlet(1, rep(1,D)), I_splines_functions)) +
  stat_function(fun = convex_combination(rdirichlet(1, rep(1,D)), I_splines_functions)) +
  stat_function(fun = convex_combination(rdirichlet(1, rep(1,D)), I_splines_functions))
```



I-Splines in Stan

We use custom code from the file isplines.R to convert the symbolic expressions, M4_expr into Stan code so that we can copy and paste the code in to Stan (see the functions expr_to_stan_str and linear_combination_stan). Using this we can infer monotonic functions in Stan. We draw from a uniform Dirichlet distribution a set of coefficients, use these as coefficients of the I-Splines to create a monotonic function, then create noisy observations of the function using Gaussian noise. We then infer the monotonic function from the resulting data using I-Splines in Stan.

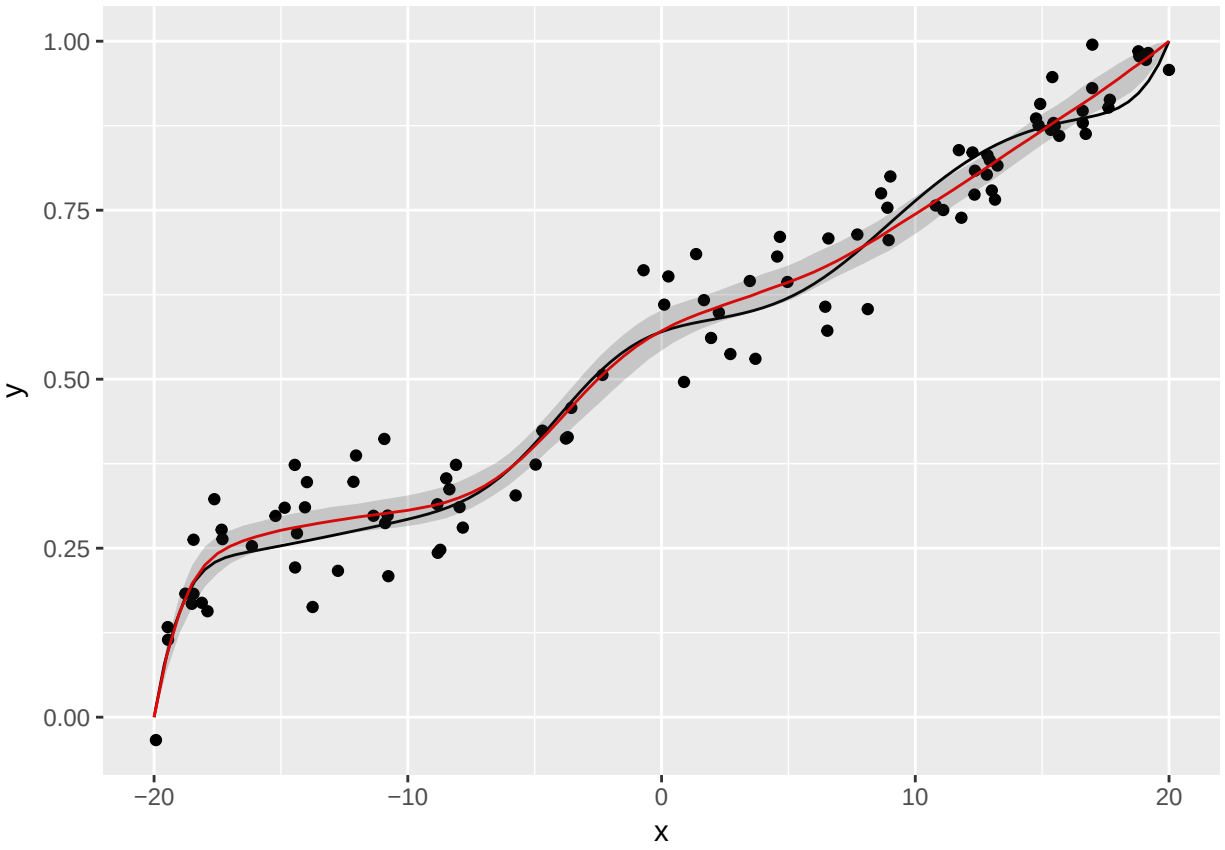
```
#create random monotonic function
#we use a fixed draw and a set seed for consistency
#s <- rdirichlet(1, rep(1,D))
s <- c(0.2244201, 0.01539845, 0.01960153, 0.0337366, 0.0289699, 0.2522589, 0.01027158, 0.02960878, 0.160
f <- convex_combination(s, I_splines_functions)

N <- 100
set.seed(1991)

#simulate random observations of the function
sim <- tibble(x = runif(N, -20, 20)) %>% mutate(y = f(x) + rnorm(N, sd = 0.05))

#plot generated data
ggplot(sim, aes(x,y)) + geom_point() + stat_function(fun = f)
```




```
#fit in Stan
dat_ispline <- list(N = N, x = sim$x, y = sim$y)
fit <- stan("stan/isplines.stan", data = dat_ispline, chains = 1, iter = 1000,refresh=1000)
```

```
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.hp
                 from /home/arya/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/math/rev/co
                 from /home/arya/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/math/rev/co
                 from /home/arya/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/math/rev/co
                 from /home/arya/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/math/rev/ma
                 from /home/arya/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/math.hpp:4,
                 from /home/arya/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/src/stan/model/m
                 from fileb4a4196b20e1.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

SAMPLING FOR MODEL 'isplines' NOW (CHAIN 1).

Gradient evaluation took 0.000166 seconds
1000 transitions using 10 leapfrog steps per transition would take 1.66 seconds.
Adjust your expectations accordingly!


Iteration:   1 / 1000 [  0%]  (Warmup)
Iteration: 501 / 1000 [ 50%]  (Sampling)
Iteration: 1000 / 1000 [100%]  (Sampling)

 Elapsed Time: 4.13849 seconds (Warm-up)
               2.35747 seconds (Sampling)
               6.49595 seconds (Total)
```

```r
posterior_summary <- (summary(fit, pars = "fhat")$summary)[,c("2.5%", "50%", "97.5%")] %>%
  as_tibble %>%
  mutate(x = seq(-20, 20, by = 0.5))

#plot posterior mode in red
ggplot() +
  geom_point(aes(x,y), data = sim) +
  stat_function(fun = f, geom = "line", data = data.frame(x = c(-20, 20))) +
  geom_line(aes(x,`50%`),data = posterior_summary, color = "red") +
  geom_ribbon(aes(x, ymin=`2.5%`,ymax=`97.5%`), data = posterior_summary, alpha = 0.2)
```



Modeling AD Progression with I-Splines

With I-Splines in our toolbelt, we can finally expand the linear latent progression model to a more flexible latent progression using I-Splines. We make the following improvements to the linear progression model:

1. We replace the latent disease progression function, $s_i(t)$, to be a positive linear combination of I-Splines rather than a linear function.
2. We let the measurement variability of the ABETA measurements be different for each individual, and place a hierarchical prior over these parameters. The biomarker plots suggest that the random variation in these measurements may differ in magnitude from person to person.

```
library(rstan)

# tree depth lowered for faster compilation
ispline_progression_fit <- stan("stan/ispline_progression.stan", data = dat, chains = 1, iter = 1000,re
                                control = list(adapt_delta = 0.99, max_treedepth = 10))
```

```
Warning in readLines(file, warn = TRUE): incomplete final line found on '/
home/arya/Projects/Stancon2018_Alzheimers/stan/ispline_progression.stan'
```

```
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.hp
                 from /home/arya/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/math/rev/co
                 from /home/arya/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/math/rev/co
                 from /home/arya/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/math/rev/co
                 from /home/arya/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/math/rev/ma
                 from /home/arya/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/stan/math.hpp:4,
```

```
                 from /home/arya/R/x86_64-pc-linux-gnu-library/3.4/StanHeaders/include/src/stan/model/m
                 from fileb4a41014c3f8.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

SAMPLING FOR MODEL 'ispline_progression' NOW (CHAIN 1).

Gradient evaluation took 0.000704 seconds
1000 transitions using 10 leapfrog steps per transition would take 7.04 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: 237.334 seconds (Warm-up)
               269.619 seconds (Sampling)
               506.953 seconds (Total)

Warning: There were 500 transitions after warmup that exceeded the maximum treedepth. Increase max_treed
http://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded

Warning: Examine the pairs() plot to diagnose sampling problems
```

```
ispline_progression_fit %>% print(pars = c("a_abeta", "a_hippo", "a_tau", "d_abeta", "d_hippo", "d_tau"
```

```
Inference for Stan model: ispline_progression.
1 chains, each with iter=1000; warmup=500; thin=1;
post-warmup draws per chain=500, total post-warmup draws=500.

                 mean se_mean    sd    2.5%     25%     50%     75%
a_abeta       -141.09    1.51  9.67 -157.96 -147.99 -141.09 -134.86
a_hippo         -0.40    0.00  0.03   -0.46   -0.42   -0.40   -0.38
a_tau           63.11    0.74  3.88   56.22   60.21   63.06   65.71
d_abeta        277.29    1.47  9.66  258.06  270.93  277.52  284.23
d_hippo          0.76    0.00  0.01    0.75    0.76    0.76    0.76
d_tau           52.72    0.58  3.19   46.88   50.55   52.72   54.67
b_abeta          0.36    0.03  0.16    0.18    0.25    0.32    0.41
b_hippo          0.46    0.10  0.25    0.17    0.27    0.39    0.58
b_mmse           0.31    0.07  0.15    0.12    0.18    0.27    0.41
b_tau            1.32    0.11  0.57    0.49    0.84    1.26    1.65
c_abeta          4.01    0.07  0.59    2.84    3.60    3.99    4.42
c_hippo         13.28    2.15  4.98    5.96    8.94   12.90   16.98
c_mmse          10.66    1.50  3.46    5.59    7.67   10.21   13.31
```

c_tau	27.62	1.15	9.27	13.45	20.09	26.83	33.60
sigma_abeta[1]	20.56	0.71	6.75	11.50	15.59	19.50	23.60
sigma_abeta[2]	21.50	0.64	8.07	11.05	16.22	19.64	24.91
sigma_abeta[3]	5.08	0.09	0.85	3.66	4.45	4.98	5.65
sigma_abeta[4]	10.54	0.24	2.54	6.73	8.69	10.18	11.93
sigma_abeta[5]	22.01	0.35	3.35	16.35	19.64	21.66	24.39
sigma_abeta[6]	20.29	0.28	2.74	16.17	18.35	19.87	21.79
sigma_abeta[7]	6.36	0.13	1.22	4.51	5.51	6.20	7.04
sigma_abeta[8]	12.17	0.23	2.47	8.41	10.49	11.76	13.42
sigma_abeta[9]	44.46	1.74	12.60	24.68	35.09	43.46	51.63
sigma_abeta[10]	32.15	1.42	12.13	14.20	23.67	29.99	38.58
sigma_hippo	0.03	0.00	0.00	0.02	0.02	0.03	0.03
sigma_mmse	2.16	0.02	0.20	1.82	2.02	2.13	2.29
sigma_tau	20.42	0.12	1.30	17.93	19.49	20.35	21.31
gamma[1]	2.43	0.59	3.14	0.02	0.41	1.14	3.05
gamma[2]	7.02	1.26	5.91	0.06	2.00	5.67	11.09
gamma[3]	3.19	0.73	3.83	0.04	0.54	1.63	4.33
gamma[4]	8.94	2.20	6.64	0.13	3.42	8.25	13.49
gamma[5]	3.98	1.06	3.80	0.16	1.20	2.77	5.58
gamma[6]	4.53	1.02	3.32	0.25	2.10	3.94	6.39
gamma[7]	3.29	0.83	3.42	0.13	0.65	1.94	4.66
gamma[8]	9.42	1.76	5.08	1.99	5.55	8.92	12.51
gamma[9]	4.65	0.56	3.74	0.28	1.81	3.93	6.49
gamma[10]	8.63	0.54	5.72	0.98	3.99	7.91	11.88
gamma[11]	7.35	0.47	6.04	0.27	2.48	5.58	10.71
gamma[12]	8.41	0.37	6.11	0.49	3.56	7.11	11.81
gamma[13]	8.31	0.48	5.76	0.81	4.21	7.27	10.98

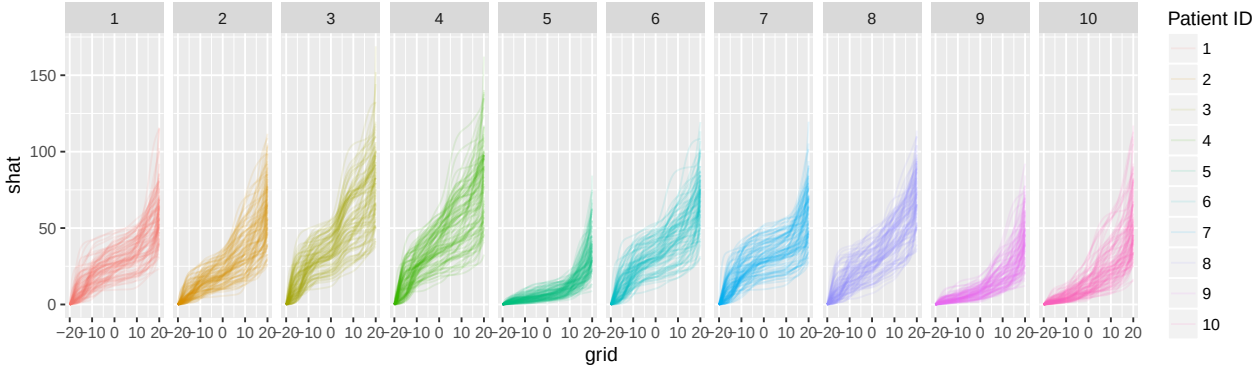
	97.5%	n_eff	Rhat
a_abeta	-121.60	41	1.01
a_hippo	-0.35	70	1.00
a_tau	70.90	28	1.02
d_abeta	294.78	43	1.01
d_hippo	0.77	67	1.02
d_tau	58.80	31	1.01
b_abeta	0.82	27	1.00
b_hippo	1.13	7	1.45
b_mmse	0.66	6	1.61
b_tau	2.63	27	1.04
c_abeta	5.18	69	1.00
c_hippo	22.87	5	1.48
c_mmse	17.41	5	1.52
c_tau	47.61	64	1.01
sigma_abeta[1]	37.67	89	1.00
sigma_abeta[2]	43.17	157	1.00
sigma_abeta[3]	6.81	97	1.02
sigma_abeta[4]	16.95	108	1.01
sigma_abeta[5]	29.48	91	1.01
sigma_abeta[6]	27.08	96	1.01
sigma_abeta[7]	9.35	91	1.00
sigma_abeta[8]	17.48	112	1.00
sigma_abeta[9]	75.88	52	1.00
sigma_abeta[10]	62.81	73	1.01
sigma_hippo	0.03	66	1.05
sigma_mmse	2.60	67	1.00

```
sigma_tau          22.95   121 1.01
gamma[1]           10.99    28 1.02
gamma[2]           19.65    22 1.21
gamma[3]           14.83    27 1.06
gamma[4]           23.01     9 1.24
gamma[5]           13.80    13 1.16
gamma[6]           11.85    11 1.15
gamma[7]           12.06    17 1.04
gamma[8]           20.46     8 1.39
gamma[9]           14.63    45 1.01
gamma[10]          22.62   111 1.02
gamma[11]          22.87   162 1.00
gamma[12]          22.62   275 1.02
gamma[13]          22.35   141 1.00

Samples were drawn using NUTS(diag_e) at Sun Jun  3 13:17:13 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).
```

An Improved Fit with I-Splines

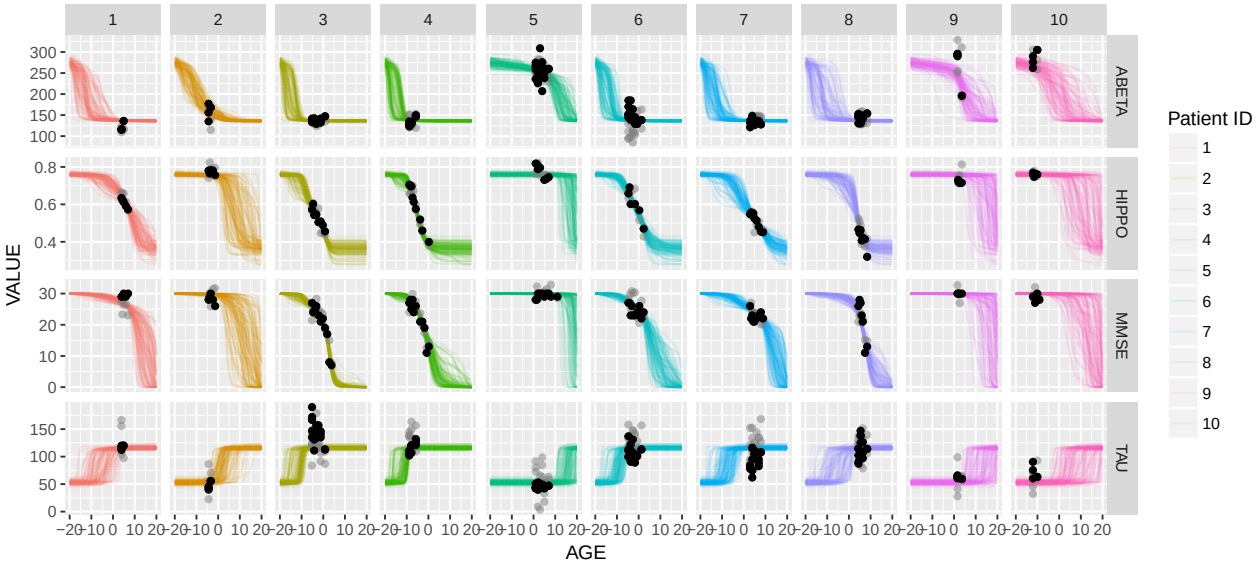
Posterior draws of individual disease progression curves immediately reveal possible non-linearities in disease progression. While some patients still maintain approximately linear disease progression such as patient 4, others such as 5 or 9 display slow initial progressions then a sharp worsening in the disease. Others, such as patient 7 seem to worsen then have a flattened prior where progression is slow, followed by another quickly progressing epoch. These more flexible disease progression curves also manifest in seemingly more accurate biomarker progression curves, leaving us with an overall more trustworthy and robust model.

```
#repeat earlier progression plots
shat_posterior_samples <- (ispline_progression_fit %>% rstan::extract(pars = "shat"))$shat[sample(1:500
master_grid_s <- map(1:dim(shat_posterior_samples)[1], function(i) sample_to_grid_s(shat_posterior_sampl

#plot once in tibble form
master_grid_s %>%
  ggplot(aes(grid, shat, group = sample_num, color = as.factor(patient_idx))) +
  geom_line(alpha = 0.1) +
  facet_grid(. ~ patient_idx, scale = "free") +
  scale_color_discrete(name = "Patient ID")
```




```
#repeat biomarker progression plots for new I-Spline model
fhat_posterior_samples <- (ispline_progression_fit %>% rstan::extract(pars = "fhat"))$fhat[sample(1:500
```

```
master_grid <- map(1:dim(fhat_posterior_samples)[1], function(i) sample_to_grid(fhat_posterior_samples[

#extract posterior predicted samples
yhat_posterior_samples <- (ispline_progression_fit %>% rstan::extract(pars = "yhat"))$yhat[500,]

adni10 %>%
  mutate(yhat = yhat_posterior_samples) %>%
  mutate(patient_idx = as.integer(as.factor(RID))) %>%
  mutate(AGE = AGE - mean(AGE)) %>%
  ggplot(aes(AGE, VALUE, color = as.factor(patient_idx))) +
  geom_line(aes(grid, yhat, group = sample_num), alpha = 0.1, data = master_grid) +
  geom_point(aes(AGE,yhat),color = "grey50", alpha = 0.5) +
  geom_point(color = "black") +
  facet_grid(BIOMARKER ~ patient_idx, scales = "free") +
  scale_color_discrete(name = "Patient ID")
```



Future Directions

- **Model individual responses in MMSE separately**: Rather than using the MMSE score at face value, it may be more prudent to model the response to individual questions on the MMSE using the tools of Item Response Theory (IRT), since the MMSE score could be calculated in an arbitrary way. This was done in in the Alzheimers setting by Vandemeulebroecke et al. (2017).
- **Model the distribution of ABETA as a mixture of Gaussians**: The bimodal nature of the population histograms of ABETA suggest that the value advances from a health Gaussian mode, to an unhealthy one as AD progresses, rather than evolving continuously as a sigmoid might suggest. Depending on how individual time series of ABETA appear, and domain knowledge, we may decide to model ABETA as a mixture whose mixture probability is governed by the latent disease progression via a logistic regression.
- **Allow variability in the biomarker progression parameters at the individual level**: We may want to examine the literature and our data to see whether progression of the various biomarkers could occur in a different order or at different relative rates depending on the patient. One possible cause of a different ordering of biomarker degradation could be differing types of dementia such as vascular dementia, as oppose to AD.

References

Jedynak, Bruno M, Andrew Lang, Bo Liu, Elyse Katz, Yanwei Zhang, Bradley T Wyman, David Raunig, et al. 2012. "A Computational Neurodegenerative Disease Progression Score: Method and Results with the Alzheimer's Disease Neuroimaging Initiative Cohort." *Neuroimage* 63 (3). Elsevier: 1478–86.

Lorenzi, Marco, Maurizio Filippone, Daniel C Alexander, and Sebastien Ourselin. 2017. "Disease Progression Modeling and Prediction Through Random Effect Gaussian Processes and Time Transformation." *arXiv Preprint arXiv:1701.01668*.

Ramsay, James O. 1988. "Monotone Regression Splines in Action." *Statistical Science*. JSTOR, 425–41.

Riihimäki, Jaakko, and Aki Vehtari. 2010. "Gaussian Processes with Monotonicity Information." In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 645–52.

Vandemeulebroecke, Marc, Björn Bornkamp, Tillmann Krahnke, Johanna Mielke, Andreas Monsch, and Peter Quarg. 2017. "A Longitudinal Item Response Theory Model to Characterize Cognition over Time in Elderly Subjects." *CPT: Pharmacometrics & Systems Pharmacology*. Wiley Online Library.