Vignette for `bayesian_isotonic()`

Philip S. Boonstra

June 24, 2021

Introduction

This vignette presents a step-by-step approach for using the `R` function `bayesian_isotonic()`, located in the project script `functions.R`.

We create a simple dataset with a binary outcome Y and a continuous predictor X where $\Pr(Y = 1|X)$ is definitely not logistic shaped. We categorize the predictor into its groups, since the methodology assumes that the predictor is categorical (this is, in my opinion, not a particularly constraining assumption, since even methods for which the theory is based upon a continuous predictor ultimately categorize the predictor when it comes times to fit the model).

We group the data into the format expected by the function and then show how you can apply the HSIPV and GAIPV priors to estimate the probability curve.

Draw data

First we draw the data:

```
n = 200;
set.seed(7201969) # Moon landing!
x <- sample(x = seq(0.05, 0.95, length = 10),
            size = n,
            replace = TRUE);
true_prob = function(x) {
  0.35 * plogis(x, location = 0.25, scale = 0.015) +
    0.65 * plogis(x, location = 0.75, scale = 0.015);
}

y <- rbinom(n, 1, true_prob(x))
```

We make 200 draws of a discrete uniform random variable to use as our predictor. To calculate the true probabilities, we use a function called `true_prob()` that is a mixture of two logistic CDFs and then sample a Bernoulli outcome according to these true probabilities. Note that this is the same data generating mechanism as in scenario 2 of the **Varying-data evaluation** of the manuscript.

The function `bayesian_isotonic()` requires that the data will be provided as a data.frame with two named columns: `y` and `n`, where the `y` value is essentially a binomial random variable for the number of successes out of the corresponding `n` trials. The ordering of the rows represents the ordered categories of `x`. That is, ξ_1 corresponds to the first row, ξ_2 to the second row, and so on. Here is how you would turn an `x` and a `y` into an appropriate data frame:

```
data_grouped <-
  tibble(x = x,
         y = y) %>%
  group_by(x) %>%
  summarize(n = length(x),
            y = sum(y)) %>%
```

```
  arrange(x)

n_breaks <- nrow(data_grouped) # This is K
```

If categorizing a continuous predictor

The theory we've developed assumes that `x` is categorical. If you have a continuous predictor `x`, we've provided a helper function called `make_grouped_data()` that will take a continuous `x` and binary `y` (and, optionally, a value of the desired breakpoints) and return a properly formatted data.frame. If breaks is not provided, the function divides the data into quintiles, i.e. five equally sized categories.

I would point out that most methods that non-parametrically estimate the x-y curve will at some point categorize the predictor, so the fact that our theory starts with an assumption of a categorical x (rather than a continuous x but then categorizing it to actually fit the model) isn't really as different as it seems.

```
# Not run
n_breaks = round(n / 10)
# Not run
breaks <-
  quantile(x, probs = seq(0, 1, length = n_breaks + 1)) %>%
  as.numeric() %>%
  replace(which.min(.), -Inf) %>%
  replace(which.max(.), Inf)
# Not run
data_grouped <-
  make_grouped_data(x, y, breaks)
```

HSIPV

Now we can fit the HSIPV-based model to these data. The `bayesian_isotonic()` function requires that you point it to the stan file that implements the desired prior via the `stan_path` argument and that you provide a list of named arguments required by the prior via `stan_args`. Instead of `stan_path`, you can provide the name of compiled stan object via `stan_fit`. If you want to have access to the individual draws from the posterior distribution, set `verbose = T`. There are other arguments, but at a minimum you need to provide values for `data_grouped`, `stan_path` (or `stan_fit`), and `stan_args`.

There is a function called `solve_for_hs_scale` that identifies the value of c that solves equation (9) in the manuscript. For a selected value of $\tilde{m}_{\text{eff}}$, set `target_mean` equal to $\tilde{m}_{\text{eff}}/(K+1)$. Here we are targeting $\tilde{m}_{\text{eff}} = 0.5$, so `target_mean` $= 0.045$.

```
hs_stan_filenames = "iso_horseshoe.stan";

hs_stan_args =
  list(
    local_dof_stan = 1,
    global_dof_stan = 1,
    alpha_scale_stan = solve_for_hs_scale(
      target_mean = 0.5 / (n_breaks + 1), #target_mean * (K+1) = m_eff
      local_dof = 1,
      global_dof = 1,
      slab_precision = 1,
      n = (n - 2),
      sigma = 2
    )$scale,
    slab_precision_stan = 1);

hs_fit =
  bayesian_isotonic(data_grouped = data_grouped,
```

```
                    stan_path = hs_stan_filenames,
                    stan_args = hs_stan_args,
                    verbose = T);
```

```
##
## SAMPLING FOR MODEL 'iso_horseshoe' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 4.9e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.49 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 7500 [  0%]  (Warmup)
## Chain 1: Iteration:  750 / 7500 [ 10%]  (Warmup)
## Chain 1: Iteration: 1500 / 7500 [ 20%]  (Warmup)
## Chain 1: Iteration: 2250 / 7500 [ 30%]  (Warmup)
## Chain 1: Iteration: 2501 / 7500 [ 33%]  (Sampling)
## Chain 1: Iteration: 3250 / 7500 [ 43%]  (Sampling)
## Chain 1: Iteration: 4000 / 7500 [ 53%]  (Sampling)
## Chain 1: Iteration: 4750 / 7500 [ 63%]  (Sampling)
## Chain 1: Iteration: 5500 / 7500 [ 73%]  (Sampling)
## Chain 1: Iteration: 6250 / 7500 [ 83%]  (Sampling)
## Chain 1: Iteration: 7000 / 7500 [ 93%]  (Sampling)
## Chain 1: Iteration: 7500 / 7500 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 1.6537 seconds (Warm-up)
## Chain 1:                3.59296 seconds (Sampling)
## Chain 1:                5.24666 seconds (Total)
## Chain 1:
```

GAIPV

Use the same function to fit the GAIPV-based model. The two arguments that `iso_gamma_stan` expects are the shape parameter and the lower truncation of the distribution to prevent underflow. We use $0.5/(K+1) = 0.045$ as the shape parameter, corresponding to 0.5 effective prior observations, and 2.22×10^{-16} as the lower truncation value. These choices respectively correspond to the $\mathbf{GA}_1$ and $\mathbf{GA}_4$ methods that we report in the manuscript.

```
ga_stan_filenames = "iso_gamma.stan"

ga_stan_args =
  list(
    list(alpha_shape_stan = 0.5 / (n_breaks + 1),
         tiny_positive_stan = .Machine$double.eps),
    list(alpha_shape_stan = 0.5 / (n_breaks + 1),
         tiny_positive_stan = 0))

ga1_fit =
  bayesian_isotonic(data_grouped = data_grouped,
                    stan_path = ga_stan_filenames,
                    stan_args = ga_stan_args[[1]],
                    verbose = T);
```

```
##
## SAMPLING FOR MODEL 'iso_gamma' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.9e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.29 seconds.
## Chain 1: Adjust your expectations accordingly!
```

```
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 7500 [  0%]  (Warmup)
## Chain 1: Iteration:  750 / 7500 [ 10%]  (Warmup)
## Chain 1: Iteration: 1500 / 7500 [ 20%]  (Warmup)
## Chain 1: Iteration: 2250 / 7500 [ 30%]  (Warmup)
## Chain 1: Iteration: 2501 / 7500 [ 33%]  (Sampling)
## Chain 1: Iteration: 3250 / 7500 [ 43%]  (Sampling)
## Chain 1: Iteration: 4000 / 7500 [ 53%]  (Sampling)
## Chain 1: Iteration: 4750 / 7500 [ 63%]  (Sampling)
## Chain 1: Iteration: 5500 / 7500 [ 73%]  (Sampling)
## Chain 1: Iteration: 6250 / 7500 [ 83%]  (Sampling)
## Chain 1: Iteration: 7000 / 7500 [ 93%]  (Sampling)
## Chain 1: Iteration: 7500 / 7500 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 2.25053 seconds (Warm-up)
## Chain 1:                5.48107 seconds (Sampling)
## Chain 1:                7.7316 seconds (Total)
## Chain 1:
```

```
ga4_fit =
  bayesian_isotonic(data_grouped = data_grouped,
                    stan_path = ga_stan_filenames,
                    stan_args = ga_stan_args[[2]],
                    verbose = T);
```

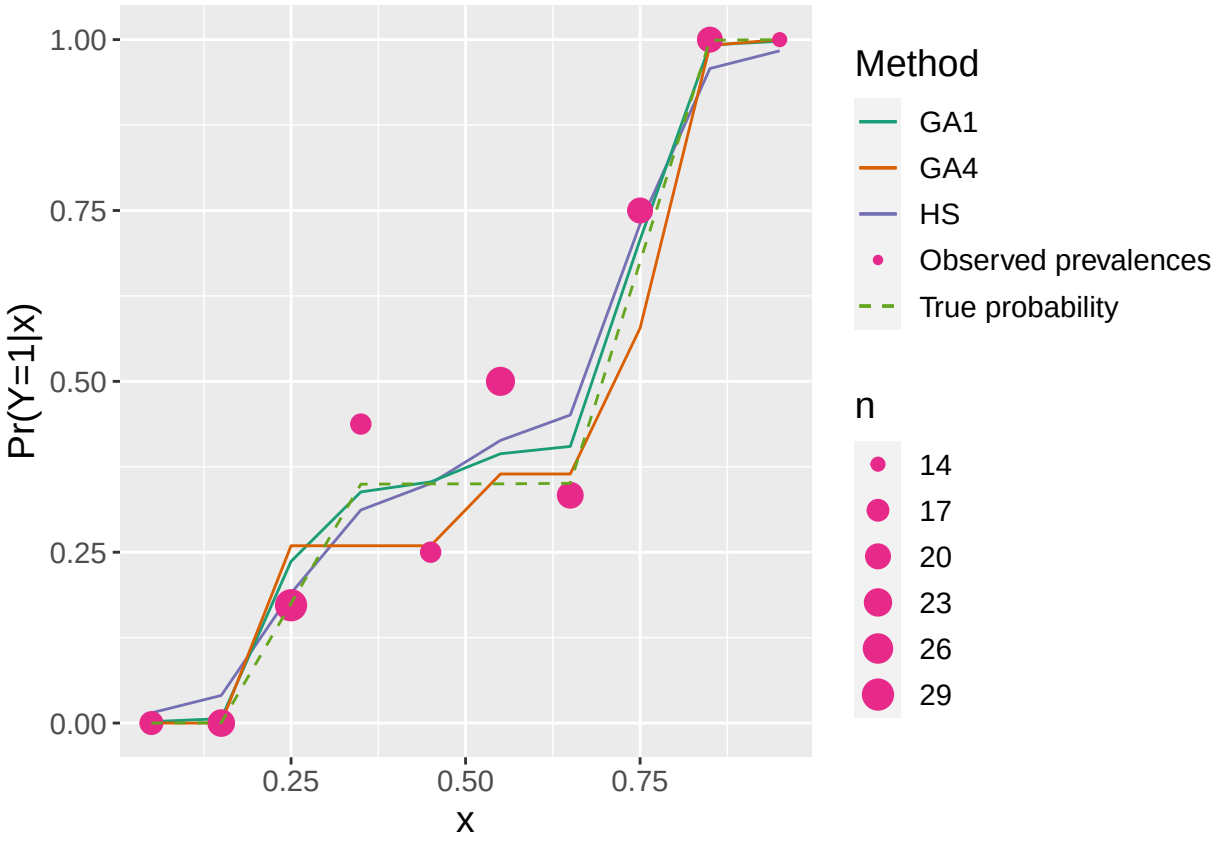
```
##
## SAMPLING FOR MODEL 'iso_gamma' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.8e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.28 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 7500 [  0%]  (Warmup)
## Chain 1: Iteration:  750 / 7500 [ 10%]  (Warmup)
## Chain 1: Iteration: 1500 / 7500 [ 20%]  (Warmup)
## Chain 1: Iteration: 2250 / 7500 [ 30%]  (Warmup)
## Chain 1: Iteration: 2501 / 7500 [ 33%]  (Sampling)
## Chain 1: Iteration: 3250 / 7500 [ 43%]  (Sampling)
## Chain 1: Iteration: 4000 / 7500 [ 53%]  (Sampling)
## Chain 1: Iteration: 4750 / 7500 [ 63%]  (Sampling)
## Chain 1: Iteration: 5500 / 7500 [ 73%]  (Sampling)
## Chain 1: Iteration: 6250 / 7500 [ 83%]  (Sampling)
## Chain 1: Iteration: 7000 / 7500 [ 93%]  (Sampling)
## Chain 1: Iteration: 7500 / 7500 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 242.384 seconds (Warm-up)
## Chain 1:                696.304 seconds (Sampling)
## Chain 1:                938.687 seconds (Total)
## Chain 1:

## Warning: There were 40 divergent transitions after warmup. See
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.

## Warning: There were 4960 transitions after warmup that exceeded the maximum treedepth. Increase max_tre
## http://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded
```

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

## Warning: The largest R-hat is NA, indicating chains have not mixed.
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#r-hat

## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be un
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#bulk-ess

## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quantile
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#tail-ess

## Warning in bayesian_isotonic(data_grouped = data_grouped, stan_path =
## ga_stan_filenames, : there were 40 divergent transitions
```

Plot results

We can plot the fitted models with the following code. In addition to plotting the interpolated fitted models, we also plot the true probability curve and the observed prevalence of the outcome in each category of the predictor. The size of the bubbles correspond to the number of observations in that category.

```
ggplot(data = data_grouped) +
  geom_line(aes(x = x,
                y = colMeans(hs_fit$all_draws$xi),
                color = "HS")) +
  geom_line(aes(x = x,
                y = colMeans(ga1_fit$all_draws$xi),
                color = "GA1")) +
  geom_line(aes(x = x,
                y = colMeans(ga4_fit$all_draws$xi),
                color = "GA4")) +
  geom_point(aes(x = x,
                 y = y / n,
                 size = n,
                 color = "Observed prevalences")) +
  geom_line(aes(x = x,
                y = true_prob(x),
                color = "True probability"),
            linetype = 2) +
  scale_color_brewer(palette = "Dark2") +
  scale_size_continuous(range = c(2,5),
                        breaks = seq(min(data_grouped$n),
                                     max(data_grouped$n),
                                     by = 3)) +
  guides(color = guide_legend(override.aes =
                                list(shape = c(NA, NA, NA, 16, NA),
                                     linetype = c(1, 1, 1, 0, 2))),
         size = guide_legend(override.aes =
                               list(color = RColorBrewer::brewer.pal(5, "Dark2")[4]))) +
  coord_cartesian(ylim = c(0, 1)) +
  labs(x = "x", y = "Pr(Y=1|x)", color = "Method") +
  theme(text = element_text(size = 14))
```



The `bayesian_isotonic()` function returns other results. Below we can see that the horseshoe prior runs faster and with fewer divergences than the gamma-based prior.

```
names(hs_fit)
```

```
##  [1] "data_grouped"           "conf_level"             "local_dof_stan"
##  [4] "global_dof_stan"        "alpha_scale_stan"       "slab_precision_stan"
##  [7] "sample_from_prior_only" "number_divergences"     "max_rhat"
## [10] "xi_number_nan"          "alpha_number_nan"       "any_nan"
## [13] "all_draws"              "chain_run_times_secs"   "total_run_time_secs"
```

```
hs_fit$number_divergences
```

```
## [1] 0
```

```
hs_fit$total_run_time_secs
```

```
## [1] 5.25
```

```
names(ga4_fit)
```

```
##  [1] "data_grouped"           "conf_level"             "alpha_shape_stan"
##  [4] "tiny_positive_stan"     "sample_from_prior_only" "number_divergences"
##  [7] "max_rhat"               "xi_number_nan"          "alpha_number_nan"
## [10] "any_nan"                "all_draws"              "chain_run_times_secs"
## [13] "total_run_time_secs"
```

```
ga4_fit$number_divergences
```

```
## [1] 40
```

```
ga4_fit$total_run_time_secs
```

```
## [1] 939
```