Generative Modeling of Customer Conversion

Michael Betancourt

July 2024

Table of contents

In this short case study I demonstrate narratively modeling by implementing the conceptual analysis discussed in Section 3.3 of my much more comprehensive discussion of the subject. If you are just getting started with the topic then I recommend first reviewing my chapters on probabilistic modeling and Bayesian inference, Stan, and model critique and iterative model development.

1 The Conceptual Data Generating Process

The basis of narratively generative modeling is domain expertise. We cannot model a data generating process unless we know something about its possible behaviors!

Moreover we all have our own experiences and none of us share exactly the same domain expertise. An immediate consequence of this diversity is that there is no universal application that resonates with everyone's domain expertise. To make this case study as broadly accessible as possible I will consider an abstract marketing example and simply provide the relevant domain expertise whenever it is necessary.

With all of that said let's say that we are interested in how often people who engage with a business end up *converting* to some deeper engagement. For example we might be curious about how likely someone who is exposed to an advertisement for a business is to visit a physical location of that business. Alternatively we might be more concerned with how likely someone who visits a visits a physical location is to buy any item or even a particular item, or how likely someone who has made a purchase previously is to make a future purchase. Similarly we might be interested in how likely someone who browses a website is to signup for a mailing list, download a piece of software, and the like.

These interactions between a potential customer and a business motivate a relatively straightforward data generating process. Because a conversion is either made or not individual interactions result in a binary outcome

$$y \in \{0, 1\}$$

where 1 denotes a conversion and 0 denotes no conversion. Any compatible data generating process is then responsible for defining the probability of these outcomes.

Now the probability of conversion is a complicated object. Conceptually it will depend on the behavior of each visitor, the behavior of the business with which they engage, and the details of that engagement. The possibilities can quickly become overwhelming. We can always make the modeling more manageable, however, by starting as simple as possible.

Before that, however, let's set up our computing environment and take a look at the available data.

2 Computational Environment Setup

Here we'll be using `R` with the base graphics cleaned up a bit.

```r
par(family="serif", las=1, bty="l",
    cex.axis=1, cex.lab=1, cex.main=1,
    xaxs="i", yaxs="i", mar = c(5, 5, 3, 5))
```

Next we'll load `Stan` and configure a few settings.

```r
library(rstan)
rstan_options(auto_write = TRUE)            # Cache compiled Stan programs
options(mc.cores = parallel::detectCores()) # Parallelize chains
parallel:::setDefaultClusterOptions(setup_strategy = "sequential")
```

Finally we'll load some utility functions into a local environment to facilitate the implementation of Bayesian inference.

```r
util <- new.env()
```

First is a suite Markov chain Monte Carlo diagnostics and estimation tools; this code and supporting documentation are both available on GitHub.

```r
source('mcmc_visualization_tools.R', local=util)
```

Second is a suite of posterior and posterior predictive visualization functions based on Markov chain Monte Carlo estimation. Again the code and supporting documentation are available on GitHub.

```r
source('mcmc_analysis_tools_rstan.R', local=util)
```

3 Data Exploration

With our environment sorted let's load in the available data which consists of two arrays, each of size `N = 1500`.

```r
data <- read_rdump('data/logs.data.R')

names(data)
```

```
[1] "x" "y" "N"
```

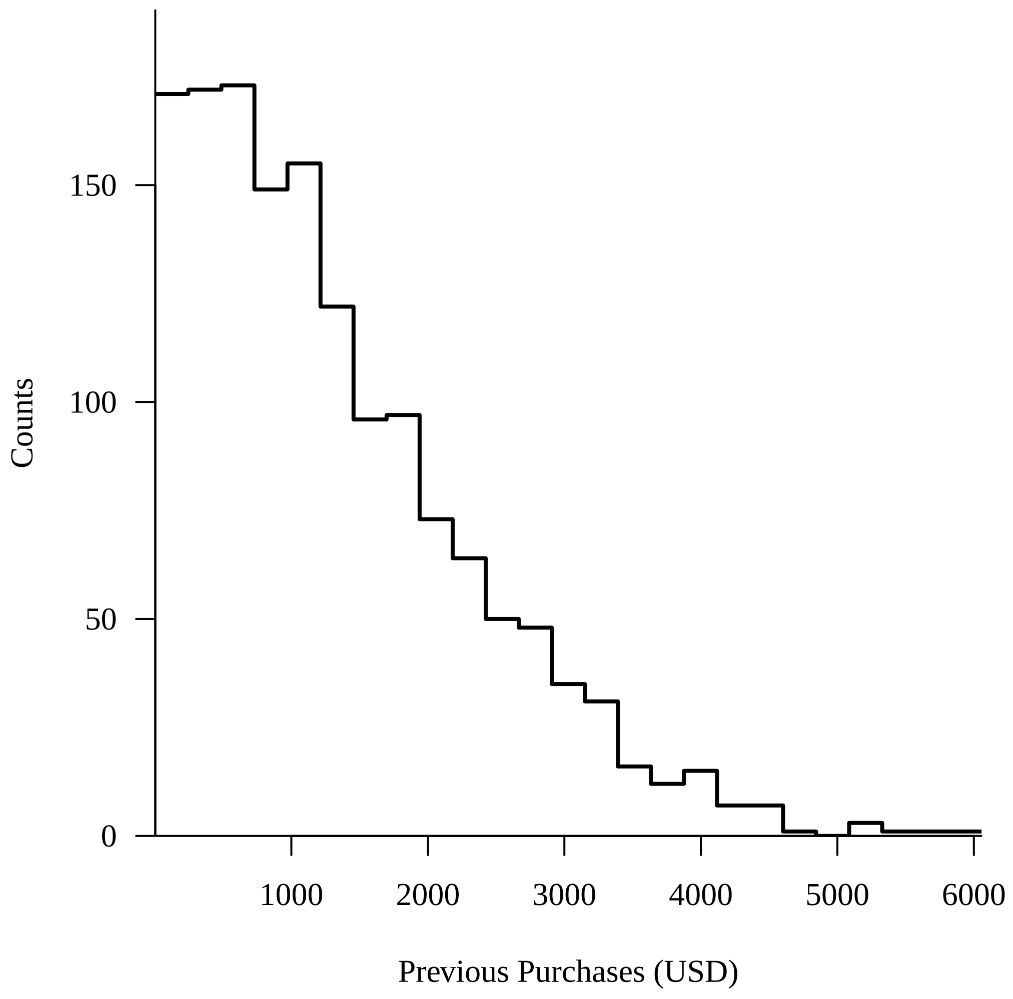
The array variable `y` contains the conversion outcomes of previous engagements, with successful conversions coded as a 1 and unsuccessful conversions coded as a 0. Overall there are a bit more failures than successes.

```
table(data$y)
```

```
  0   1
859 641
```

We also have access to the array variable `x` which contains the total purchases that each customer has made, in United States Dollar or USD, prior to the observed engagement. Most customers have spent a moderate amount with a decaying tail of more prolific shoppers.

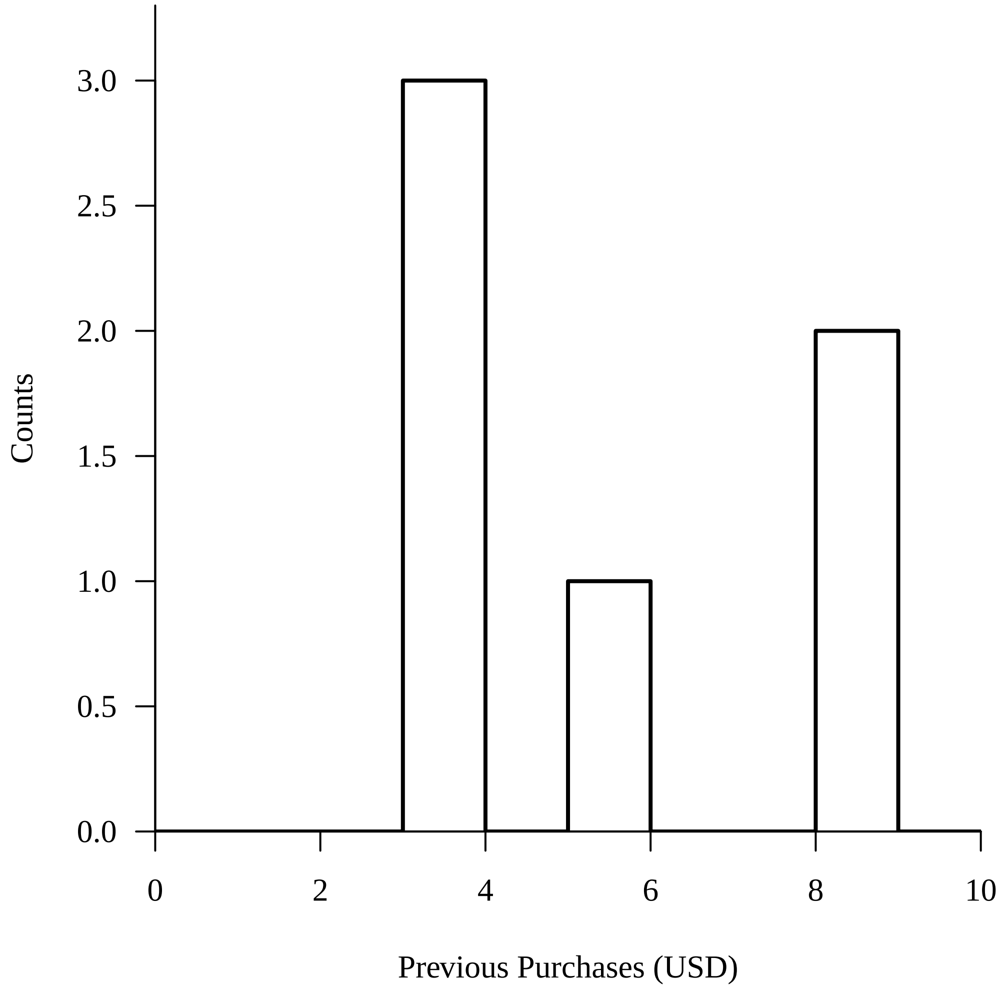
```
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

util$plot_line_hist(data$x, xlab="Previous Purchases (USD)")
```



Interestingly no customers have zero prior purchases, indicating that our data set consistent entirely of repeat customers. This limits what we might be able to learn about entirely new customers.

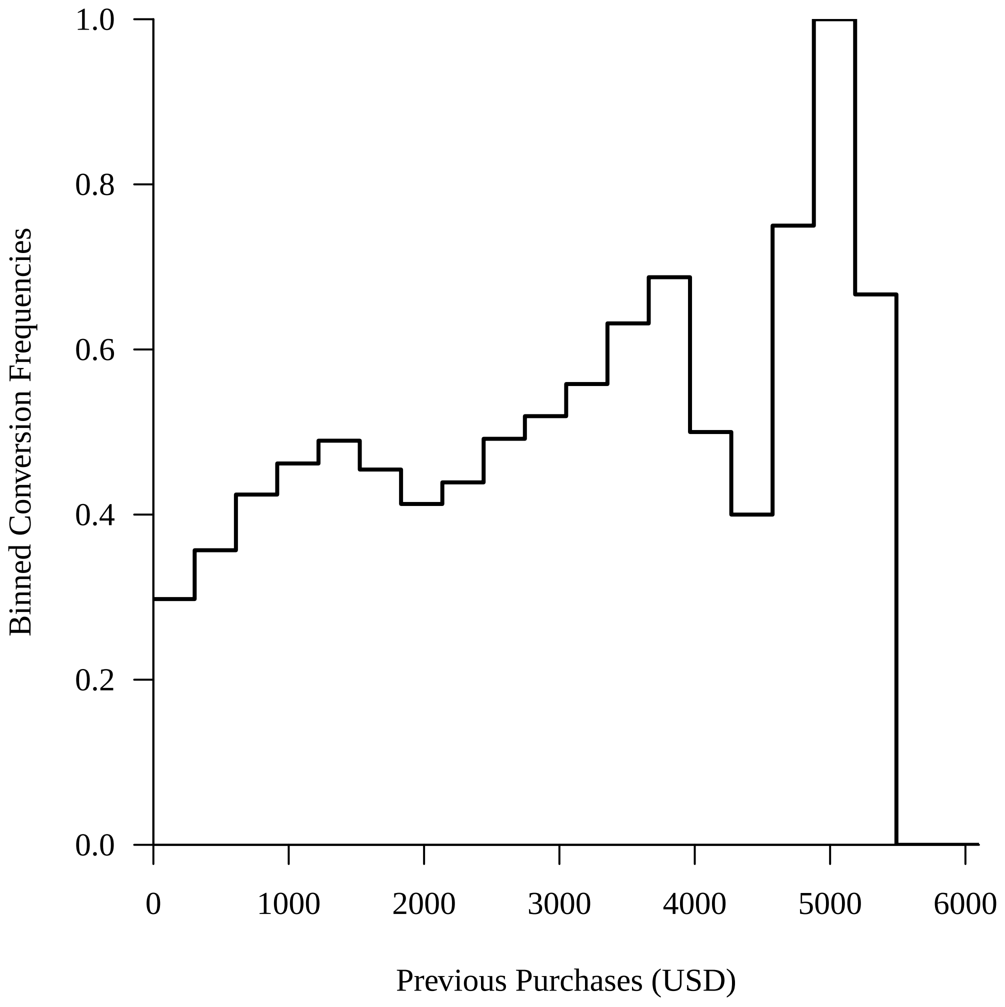
```r
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

util$plot_line_hist(data$x, 0, 10, 1, xlab="Previous Purchases (USD)")
```

```
Warning in check_bin_containment(bin_min, bin_max, values): 1494 values (99.6%)
fell above the binning.
```



The availability of data beyond just the conversion outcomes allows us to consider more sophisticated data generating processes. In particular we can consider not just the aggregate conversion probability but also the conversion probability conditional on the previous purchases. Just by looking at the data we can see that this relationship is not trivial.

```r
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

bins <- seq(0, 6100, 305)
B <- length(bins) - 1

idxs <- rep(1:B, each=2)
xs <- sapply(1:length(idxs),
             function(b) if(b %% 2 == 1) bins[idxs[b]]
                         else            bins[idxs[b] + 1])

binned_freqs <- sapply(1:B, function(b)
                       mean(data$y[bins[b] < data$x & data$x <= bins[b + 1]]))
binned_freqs <- rep(binned_freqs, each=2)

plot(xs, binned_freqs, type="l", lwd=2,
     xlab="Previous Purchases (USD)",
     ylab="Binned Conversion Frequencies")
```



4 Modeling

Having reasoned through our domain expertise and explored the available data we are finally ready to consider modeling the relevant data generating process. In order to avoid being overwhelmed by the potential complexities we will start simple and then iteratively incorporate more complexity based on the posterior retrodictive performance of each model.

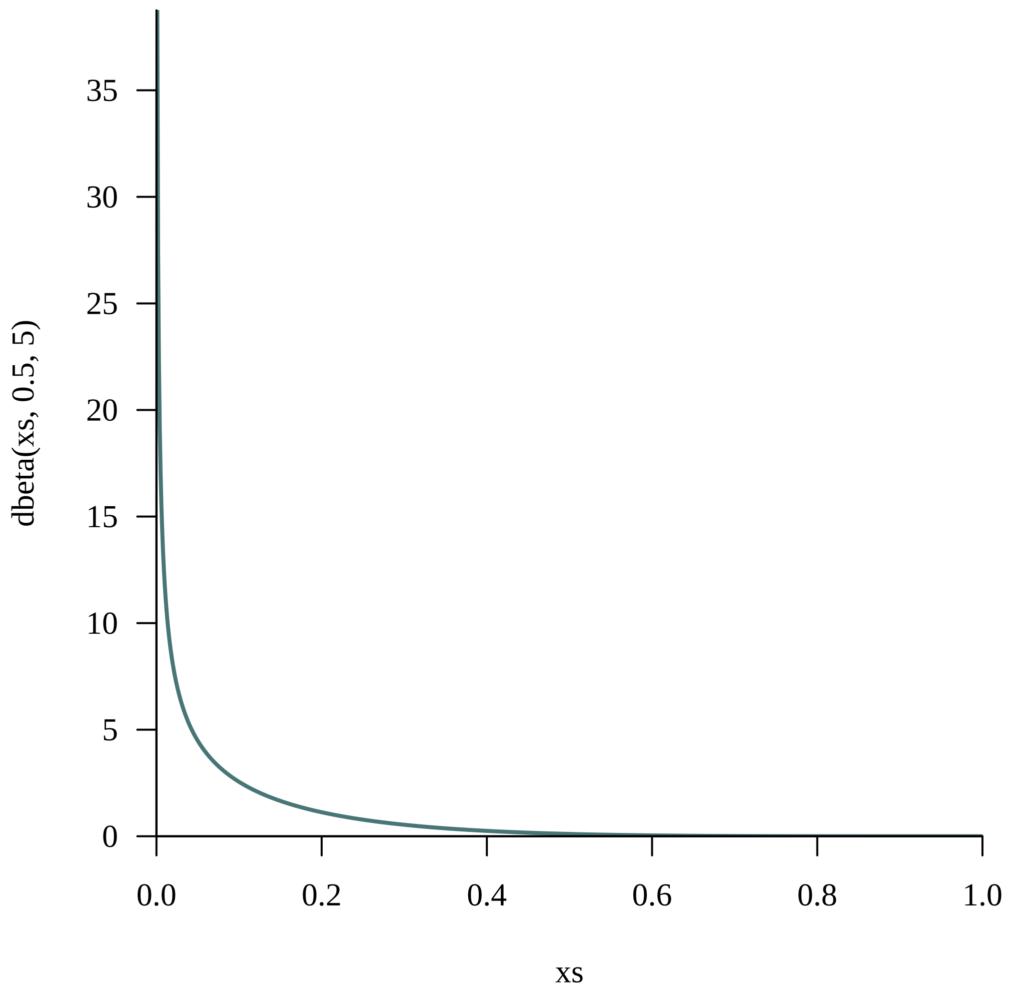
4.1 Model 1

If we assume that all customers behave identically and ignore the previous purchases altogether then we can model the aggregate conversion outcomes with a Bernoulli observational model parameterized by a single probability parameter θ,

$$\pi(y_1, \ldots, y_N \mid \theta) = \prod_{n=1}^{N} \text{Bernoulli}\,(y_n \mid \theta).$$

Any available domain expertise about the reasonable behaviors in this data generating process informs a prior model for θ. For example if we are confident that the true conversion probability is less than 0.5 then we could define a prior model with a beta probability density function that contains 99% of the prior probability to the interval $0 \le \theta \le 0.5$.

```
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

xs <- seq(0, 1, 0.001)
plot(xs, dbeta(xs, 0.5, 5.0), type="l", lwd=2, col=util$c_mid_teal)
```



For more on containment prior modeling strategies see Section 3 of my prior modeling chapter.

The prior model then lifts the observational model into a full Bayesian model

$$
\begin{aligned}
\pi(y_1, \ldots, y_N, \theta) &= \pi(y_1, \ldots, y_N \mid \theta) \, \pi(\theta) \\
&= \left[\prod_{n=1}^{N} \pi(y_n \mid \theta) \right] \pi(\theta) \\
&= \left[\prod_{n=1}^{N} \text{Bernoulli} \left(y_n \mid \theta \right) \right] \pi(\theta).
\end{aligned}
$$

We can also visualize the conditional structure of this full Bayesian model with a directed graph (Figure 1). For an introduction to directed graphs as representations of probability distributions see Section 4 of my chapter on probability theory on product spaces.

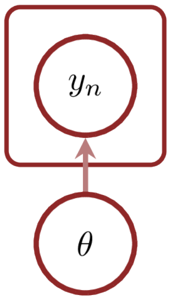


Figure 1: Our first model of customer conversion assumes that all customers behave the same, allowing their behavior to be captured in a single probability parameter θ.

To evaluate this full Bayesian model on the observed data and extract posterior insights we will implement it as a probabilistic program in the `Stan` modeling language. In addition to the full Bayesian model we also implement posterior predictions in the `generated quantities` block. Note that I am focusing on explicitness and clarity in this Stan program and those that follow; consequently these programs will not always demonstrate coding techniques for optimal performance.

We can then use the dynamic Hamiltonian Monte Carlo sampler in `Stan` to quantify the behavior of the posterior distribution.

```
fit <- stan(file="stan_programs/model1.stan",
            data=data, seed=8438338,
            warmup=1000, iter=2024, refresh=0)
```

Before examining the posterior output we need to check for any indications that our computation is inaccurate. Fortunately there are no signs of unfaithful posterior quantification here.

```
diagnostics1 <- util$extract_hmc_diagnostics(fit)
util$check_all_hmc_diagnostics(diagnostics1)
```

```
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
```

```
samples1 <- util$extract_expectands(fit)
base_samples <- util$filter_expectands(samples1, c('theta'))
util$check_all_expectand_diagnostics(base_samples)
```

```
All expectands checked appear to be behaving well enough for reliable
Markov chain Monte Carlo estimation.
```

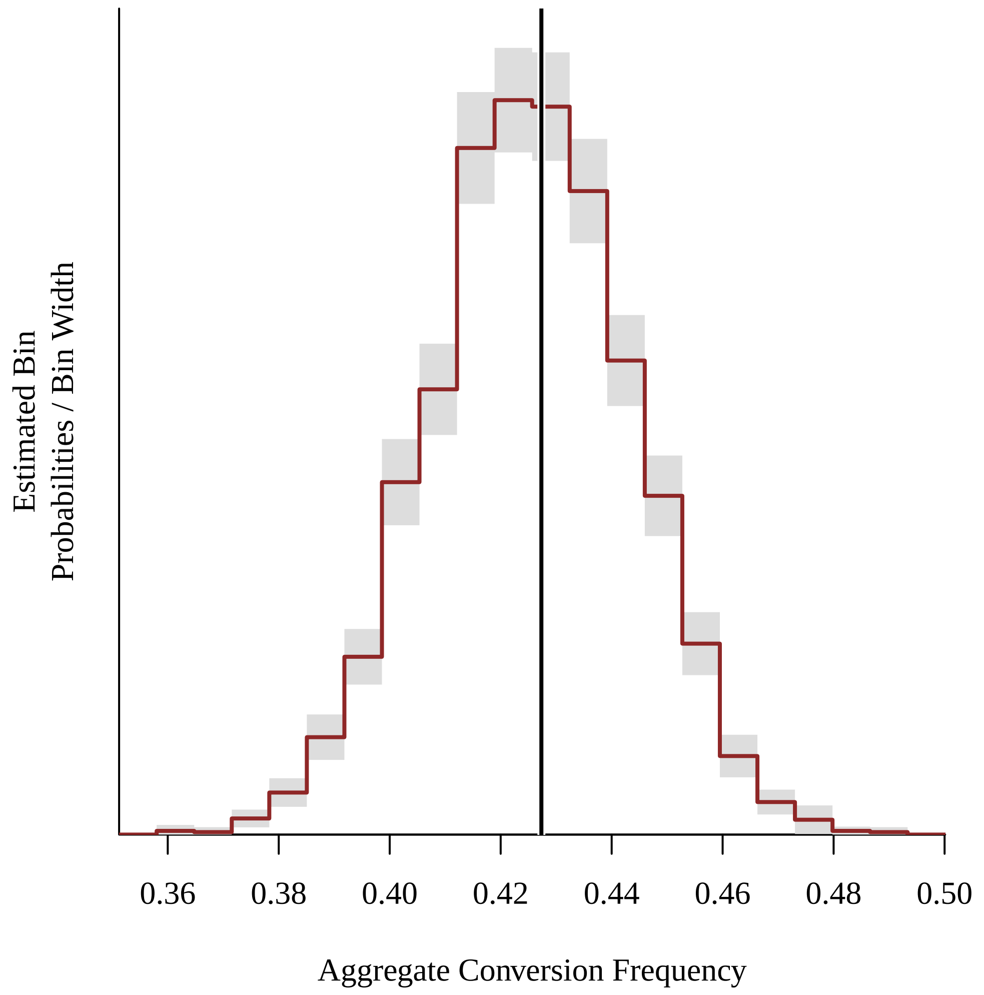
Next we need to evaluate the adequacy of our modeling assumptions with posterior retrodictive checks. Posterior retrodictive checks compare the behavior of the observed data to the behavior of the posterior predictive distribution for any inconsistencies. For much more on visual posterior retrodictive checks see Section 1.4.3 of my iterative model development chapter.

Let's first consider retrodictive performance within the scope of the empirical conversion frequency summary statistic that maps all $N = 1500$ binary outcomes to a single real-valued outcome,

$$\hat{p}(y_1, \ldots, y_N) = \frac{\sum_{n=1}^{N} y_n}{N}.$$

Fortunately everything looks consistent.

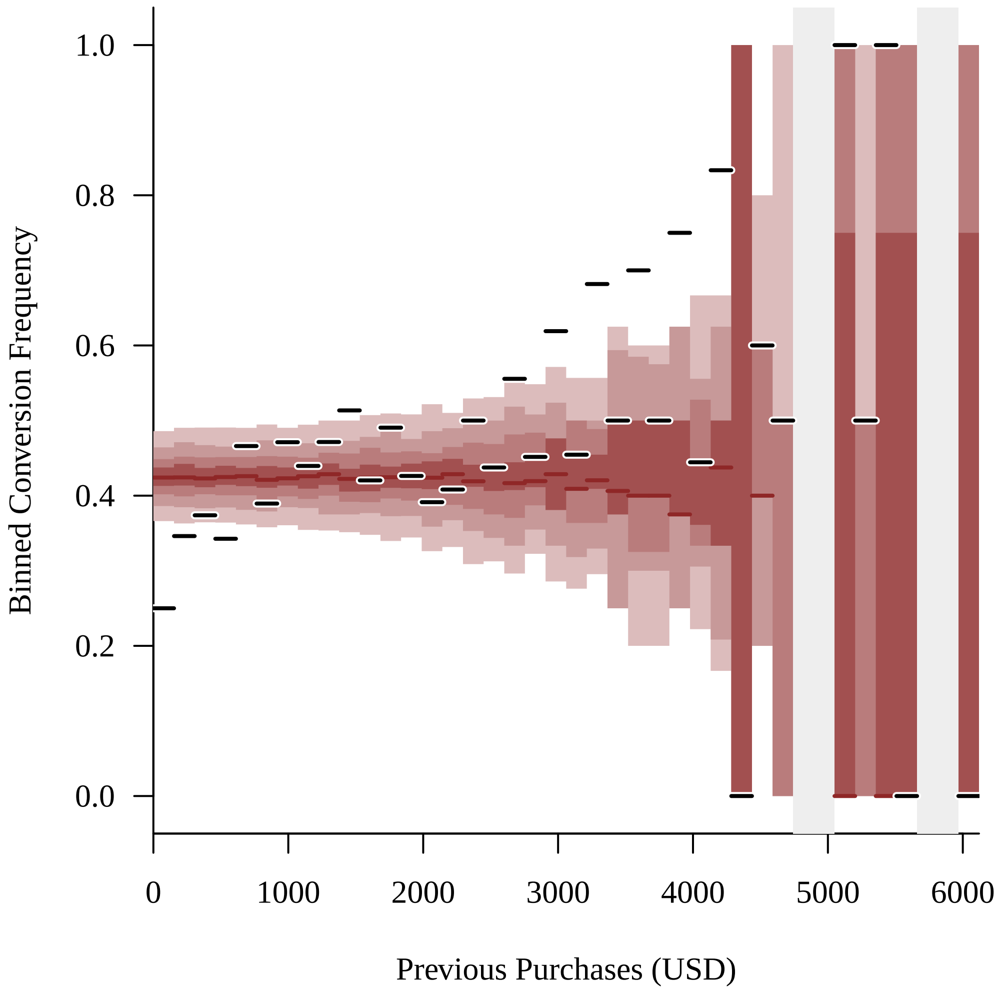
```
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

util$plot_expectand_pushforward(samples1[['p_hat_pred']], 20,
                                display_name="Aggregate Conversion Frequency",
                                baseline=mean(data$y))
```



Next we can explore how consistent the observed outcomes and posterior predictive outcomes are in bins of previous purchases. The statistic needed for this check is a bit more complicated; see Section 2.5 of my Taylor modeling chapter for a detailed construction.

The posterior predictive behavior concentrates around the same point in all of the previous purchases bins. In hindsight this is unsurprising given our simple model assumptions. On the other hand the observed data exhibits a nontrivial relationship between conversion frequency and previous purchases. In particular the disagreement extends beyond the posterior predictive uncertainties, suggesting that our model is not flexible enough to capture the true data generating behavior.

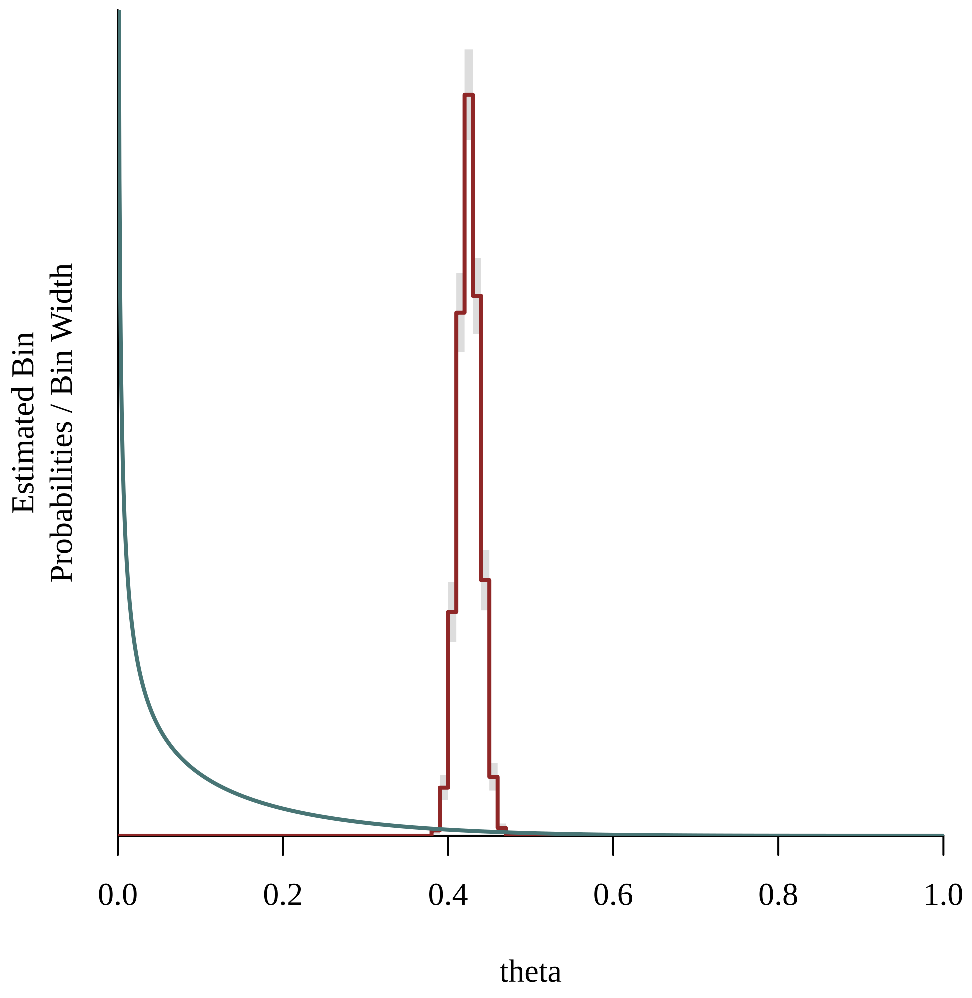
```
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

pred_names <- sapply(1:data$N, function(n) paste0('y_pred[', n, ']'))
util$plot_conditional_mean_quantiles(samples1, pred_names, data$x,
                                     0, 6120, 153,
                                     baseline_values=data$y,
                                     xlab="Previous Purchases (USD)",
                                     ylab="Binned Conversion Frequency")
```



At this point we *could* examine posterior inferences, but we have to be careful with their interpretation. Models that are too rigid often contort themselves in order to accommodate as much as possible the particular idiosyncrasies of the observed data, pulling the individual parameters away from any generalizeable interpretation.

Interestingly we see here that the posterior distribution for θ is pushing up against our prior model which could be a sign that the prior model is offering some regularization against this kind of contortion.

```
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

util$plot_expectand_pushforward(samples1[['theta']], 100,
                                display_name="theta",
                                flim=c(0, 1))
xs <- seq(0, 1, 0.001)
lines(xs, dbeta(xs, 0.5, 5.0), lwd=2, col=util$c_mid_teal)
```



That said until we address the missing features we are just speculating.

4.2 Model 2

An immediate limitation of our initial model is the assumption of a monolithic conversion probability. We might be able to address the inadequacy of the initial model by allowing the conversion probability to vary with the known previous purchases.

Now there are many possible data generating processes that result in coupled behavior between conversions and previous purchases. Here we will take a slightly less generative approach, assuming that the behavior of previous purchases is independent to the behavior of conversions given previous purchases and modeling the relationship heuristically by replacing the parameter θ with the output of some parametric function of the previous purchases, $\theta(x, \psi)$.

Our full Bayesian model then takes the form (Figure 2)

$$\pi(y_1, \ldots, y_N, \psi; x_1, \ldots, x_N) = \left[\prod_{n=1}^{N} \text{Bernoulli} \left(y_n \mid \theta \left(x_n, \psi \right) \right) \right] \pi(\psi).$$

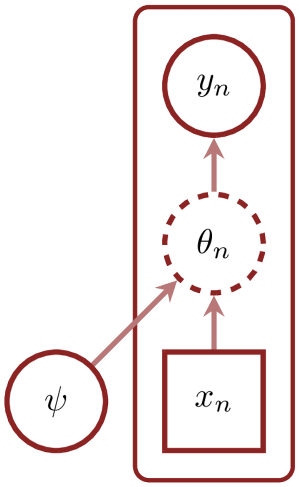


Figure 2: One way to allow conversion outcomes to vary with previous purchases is to replace the conversion probability parameter with the output of a deterministic function of the previous purchases.

The model is not complete, however, until we have an explicit form for this probability function. Specifically we need to engineer functional behavior that is consistent with our domain expertise.

For example let's say that we believe that the conversion probability should initially rise with previous purchases but also that any rise cannot continue forever. In other words any rise should slow and eventually saturate for sufficiently large previous purchases. One way to accommodate this behavior is with the functional model

$$\theta(x, \psi) = \theta(x, \psi_1, \psi_2) = \psi_1 \cdot \left(1 - \exp(-\psi_2 \cdot x)\right).$$

```r
theta <- function(x, psi1, psi2) {
  psi1 * (-expm1(-psi2 * x))
}
```

Notice that as ψ_2 goes to infinity the function $\theta(x, \psi_1, \psi_2)$ converges to a constant function and our model reduces to the previous model with ψ_1 replacing θ. In other words with this functional form our model *expands* upon the previous model by adding new functionality *but not excluding any existing functionality.* This helps make our iterative model development robust to potential problems like over-fitting.

At this point we need to develop a prior model for ψ_1 and ψ_2 that ensures reasonable functional behaviors. Here we will assume an independent component prior model,

$$\pi(\psi_1, \psi_1) = \pi(\psi_1) \, \pi(\psi_2).$$

Because ψ_1 determines the range of possible conversion probabilities we can carry over the domain expertise that we elicited about the aggregate conversion probability as a conservative prior model for ψ_1,

$$\pi(\psi_1) = \text{beta}(\psi_1 \mid 0.5, 5.0).$$

An appropriate prior model for ψ_2 depends on what we know about how quickly the conversion probability can saturate. In particular the previous purchases where the conversion probability reaches half of its maximum is given by

$$\begin{aligned}
\psi_1 \frac{1}{2} &= \psi_1 \left(1 - \exp\left(-\psi_2 \cdot x_{\frac{1}{2}}\right)\right) \\
\frac{1}{2} &= 1 - \exp\left(-\psi_2 \cdot x_{\frac{1}{2}}\right) \\
\exp\left(-\psi_2 \cdot x_{\frac{1}{2}}\right) &= \frac{1}{2} \\
-\psi_2 \cdot x_{\frac{1}{2}} &= \log \frac{1}{2} \\
\psi_2 &= \frac{\log 2}{x_{\frac{1}{2}}}.
\end{aligned}$$

Now let's say that we are fairly confident that it will take at least 2000 USD of previous purchases to achieve reach five times $x_{\frac{1}{2}}$,

$$\begin{aligned}
5 \, x_{\frac{1}{2}} &> 2000 \, \text{USD} \\
x_{\frac{1}{2}} &> 400 \, \text{USD}.
\end{aligned}$$

This implies that

$$\begin{aligned}\psi_2 &= \frac{\log 2}{x_{\frac{1}{2}}} \\ &< \frac{\log 2}{400}\,\mathrm{USD}^{-1} \\ &< 1.7 \cdot 10^{-3}\,\mathrm{USD}^{-1} \\ &\lessapprox 2\,\mathrm{kUSD}^{-1}.\end{aligned}$$

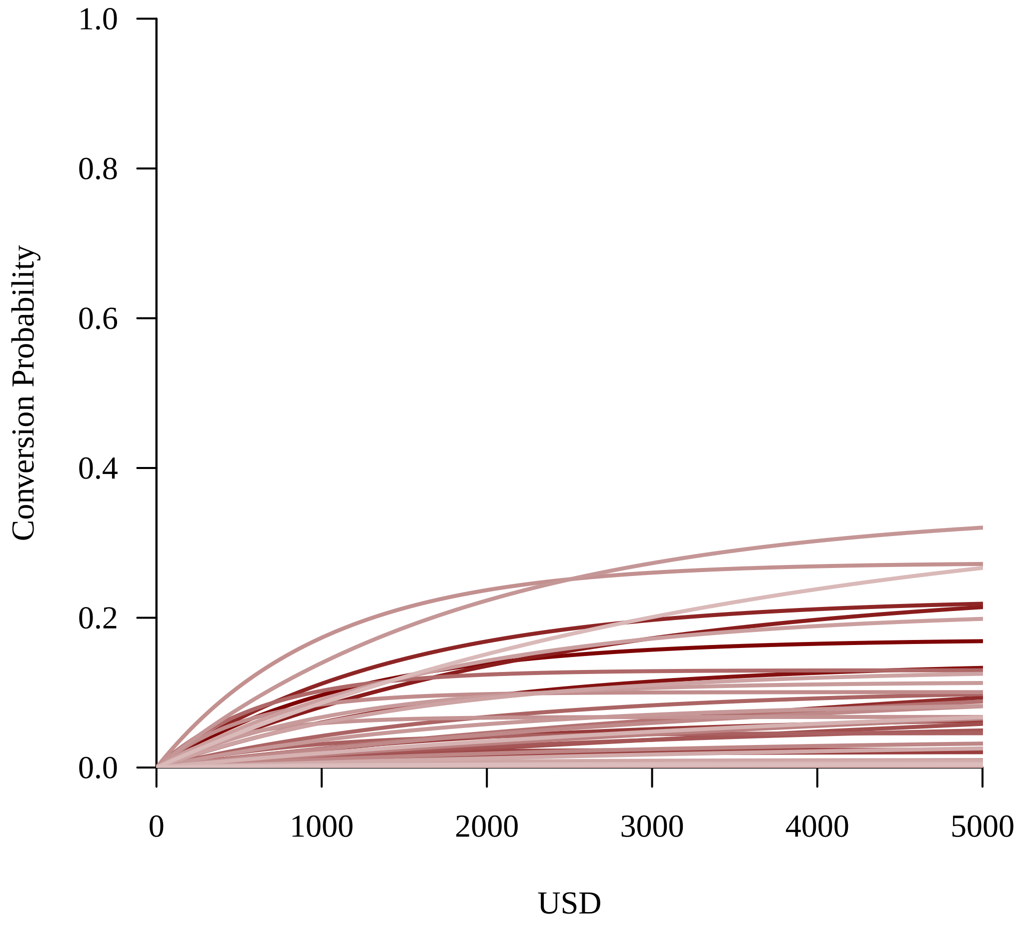
We can ensure this containment with a half-normal prior model for ψ_2,

$$\pi(\psi_2) = \text{half-normal}\left(\psi_2 \mid 0, \frac{2}{2.57}\right).$$

For an explanation of the factor of 2.57 see Section 3.3 of my prior modeling chapter.

In developing this prior model we have considered the two parameters ψ_1 and ψ_2 separately from each other even though they interact when evaluating the conversion probability at a given value for previous purchases. To verify that there are not any undesired behaviors hiding in those interactions we can perform a prior pushforward check where we visualize an ensemble of possible conversion function behaviors. Fortunately none of these derived behaviors exhibit any pathological behavior.

```
J <- 50
nom_colors <- c("#DCBCBC", "#C79999", "#B97C7C",
                "#A25050", "#8F2727", "#7C0000")
line_colors <- colormap(colormap=nom_colors, nshades=J)

par(mfrow=c(1, 1), mar=c(5, 5, 3, 1))

plot(0, type='n',
     xlim=c(0, 5000), xlab="USD",
     ylim=c(0, 1), ylab="Conversion Probability")

for (j in 1:J) {
  psi1 <- rbeta(1, 0.5, 5)
  psi2 <- abs(rnorm(1, 0, 2 / 2.57))
  xs <- seq(0, 5000, 10)
  ys <- sapply(xs, function(x) theta(x, psi1, 1e-3 * psi2))
  lines(xs, ys, lwd=2, col=line_colors[j])
}
```



Armed with a carefully considered prior model we can let loose our expanded model onto the observed data.

```
fit <- stan(file="stan_programs/model2.stan",
            data=data, seed=8438338,
            warmup=1000, iter=2024, refresh=0)
```

Fortunately the computational diagnostics are clean.

```
diagnostics2 <- util$extract_hmc_diagnostics(fit)
util$check_all_hmc_diagnostics(diagnostics2)
```

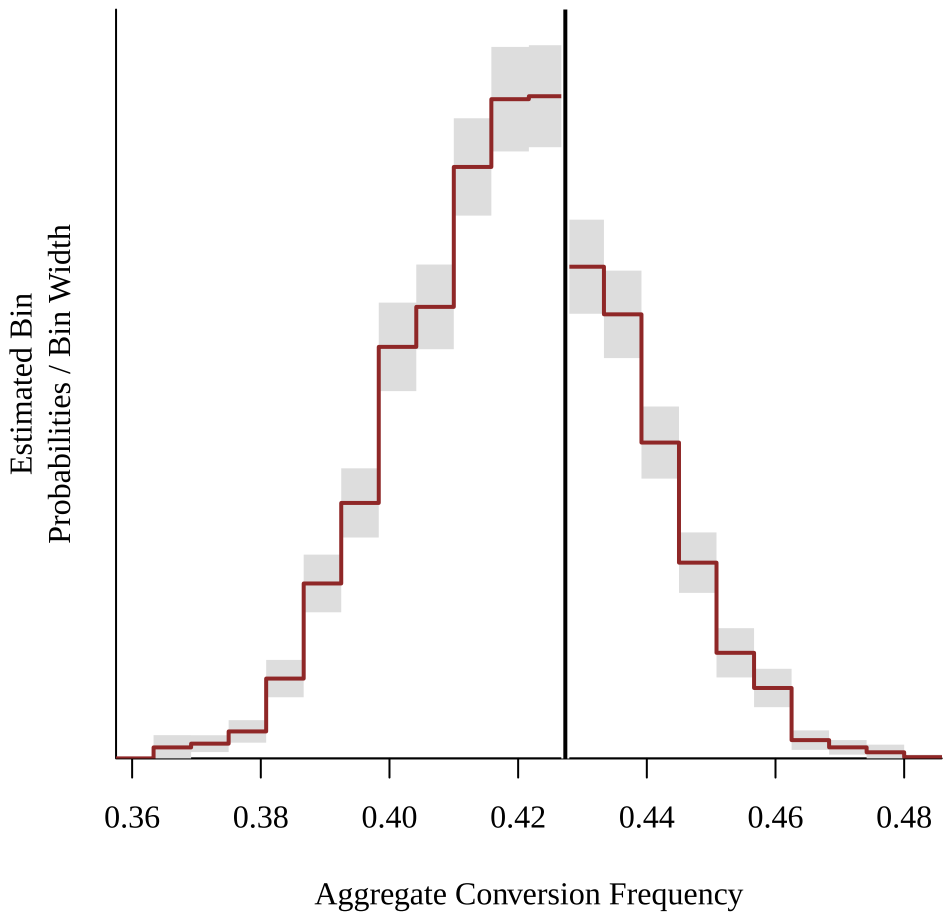
```
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
```

```r
samples2 <- util$extract_expectands(fit)
base_samples <- util$filter_expectands(samples2,
                                       c('psi1', 'psi2'))
util$check_all_expectand_diagnostics(base_samples)
```

```
All expectands checked appear to be behaving well enough for reliable
Markov chain Monte Carlo estimation.
```

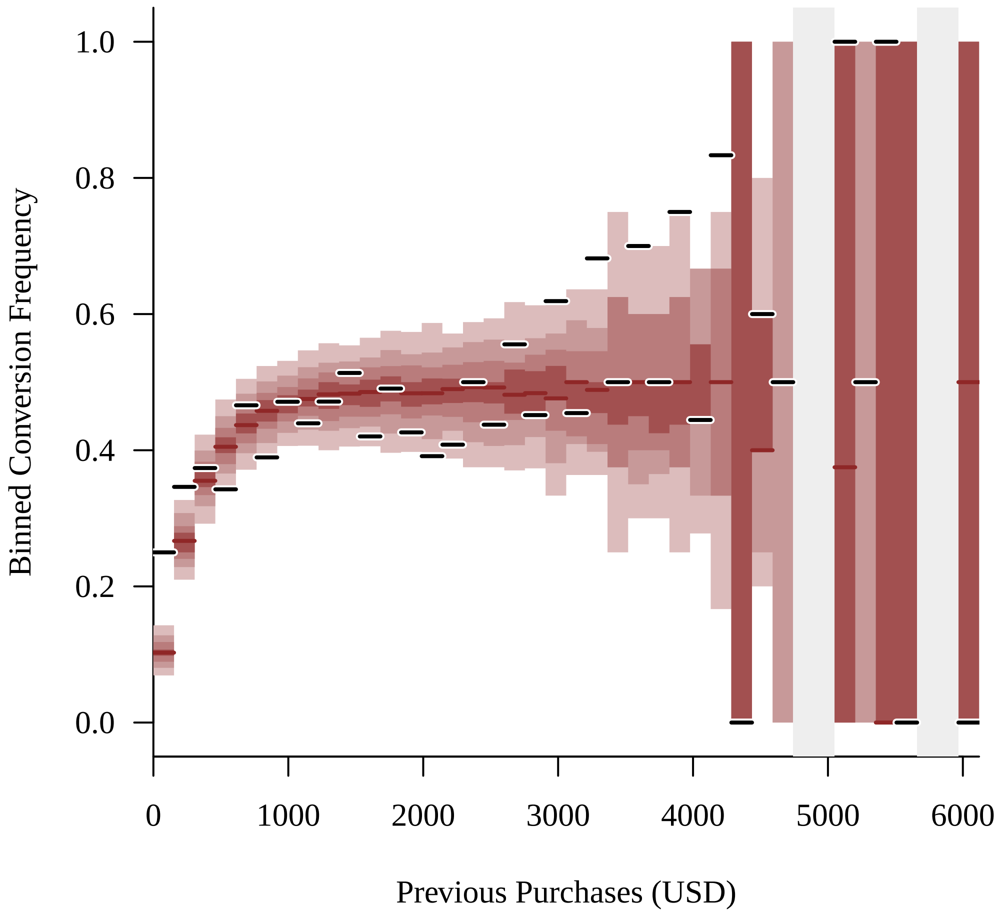
The retrodictive behavior in the empirical frequency summary statistic also looks good.

```r
par(mfrow=c(1, 1), mar=c(5, 5, 3, 1))

util$plot_expectand_pushforward(samples2[['p_hat_pred']], 20,
                                display_name="Aggregate Conversion Frequency",
                                baseline=mean(data$y))
```



While the retrodictive behavior in the binned empirical frequencies is a substantial improvement to that of the previous model it still leaves much to be desired. In particular the posterior predictive behavior starts at zero conversion frequency whereas the observed conversion frequencies appear to start at a non-zero value.

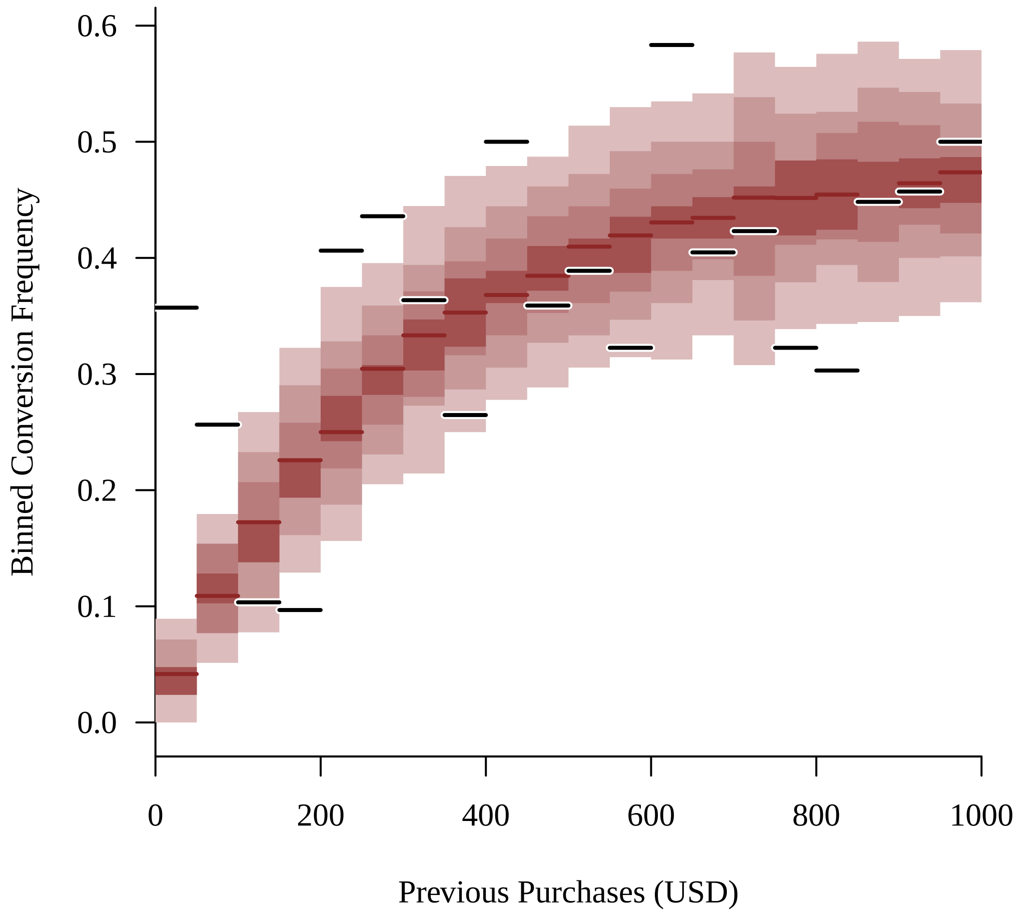
```
par(mfrow=c(1, 1), mar=c(5, 5, 3, 1))

pred_names <- sapply(1:data$N, function(n) paste0('y_pred[', n, ']'))
util$plot_conditional_mean_quantiles(samples2, pred_names, data$x,
                                     0, 6120, 153,
                                     baseline_values=data$y,
                                     xlab="Previous Purchases (USD)",
                                     ylab="Binned Conversion Frequency")
```



We can focus on these discrepancies by zooming into smaller previous purchases.

```
par(mfrow=c(1, 1), mar=c(5, 5, 3, 1))

pred_names <- sapply(1:data$N, function(n) paste0('y_pred[', n, ']'))
util$plot_conditional_mean_quantiles(samples2, pred_names, data$x,
                                     0, 1000, 50,
                                     baseline_values=data$y,
                                     xlab="Previous Purchases (USD)",
                                     ylab="Binned Conversion Frequency")
```

```
Warning in check_bin_containment(bin_min, bin_max, obs_xs, "conditioning
value"): 809 conditioning values (53.9%) fell above the binning.
```



The challenge now is to leverage our domain expertise into a hypothesize about what could be inadequate about our current modeling assumptions. Why might new customers with no previous purchases exhibit a non-zero conversion probability?

4.3 Model 3

One of the subtle issues with domain expertise is that it takes energy and time to transform from implicit, qualitative information to explicit, quantitative information. After a finite amount of effort we will only ever be working with a finite amount of our, or our team's, domain expertise. The challenge to making elicitation productive is identifying what kind of domain expertise to elicit. Conveniently modeling problems are often highly effective at triggering unused knowledge.

For example in trying to reason about non-zero initial conversion probabilities we might remember, be informed by our collaborators, or even learn from public literature that our target market is not homogeneous but instead consists of typical customers and another, more engaged group of special customers. This might happen, for instance, when the engagement we are offering is branded with a popular cultural icon, such as a sports team or celebrity, and fans of that icon will have the same propensity to engage regardless of their previous business.

This narrative suggests not one but rather two observational models:

$$\text{Bernoulli}\left(y_n \mid \theta\left(x_n, \psi\right)\right)$$

for the regular customers and

$$\text{Bernoulli}\left(y_n \mid \theta^{\text{VIP}}\right)$$

for the very important customers, or *VIPs*.

The problem, however, is that we don't know to which of these groups a given customer in our observed data belongs, and hence which data generating process to apply. In order to account for the possibility that each customer could belong to either group we can appeal to a mixture model with the mixture parameter λ determining the overall proportion of VIP customers (Figure 3),

$$\begin{aligned}
\pi(y_1, \ldots, y_N, &\psi, \theta^{\text{VIP}}, \lambda; x_1, \ldots, x_N) \\
&= \left[\prod_{n=1}^{N} (1 - \lambda) \cdot \text{Bernoulli}\left(y_n \mid \theta\left(x_n, \psi\right)\right) + \lambda \cdot \text{Bernoulli}\left(y_n \mid \theta^{\text{VIP}}\right) \right] \\
&\quad \cdot \pi(\psi)\, \pi(\theta^{\text{VIP}})\, \pi(\lambda).
\end{aligned}$$

All that remains is a prior model for the proportion of VIP customers λ and the conversion probability of the VIP customers θ^{VIP}. As always these prior models depend on our domain expertise, which in this case is entirely hypothetical. For example if are aware of an ongoing promotion that is likely to resonate with the VIP customers then we might engineer a prior model that concentrates on large values of θ^{VIP}. Here we'll take a diffuse prior model for λ,

$$\pi(\lambda) = \text{beta}(\lambda \mid 1, 1),$$

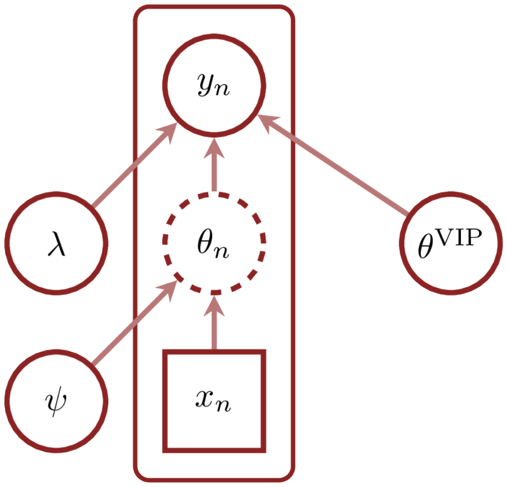


Figure 3: A mixture model allows us to integrate two distinct data generating processes together even when we don't know from which data generating process a given observation was drawn.

and a prior model that contains θ^{VIP} above 0.5 to account for the higher engagement,

$$\pi(\theta^{\text{VIP}}) = \text{beta}(\theta^{\text{VIP}} \mid 5.0, 0.5).$$

Let's implement this further expanded model in `Stan` and then take it for a spin.

```r
fit <- stan(file="stan_programs/model3.stan",
            data=data, seed=8438338,
            warmup=1000, iter=2024, refresh=0)
```

Unfortunately the computational diagnostics are no longer as quiet as they have been. The divergent transitions indicate that there are model configurations consistent with the data that `Stan` has not been able to explore. For more on dealing with divergences see my chapter on inferential degeneracy.

```r
diagnostics3 <- util$extract_hmc_diagnostics(fit)
util$check_all_hmc_diagnostics(diagnostics3)
```

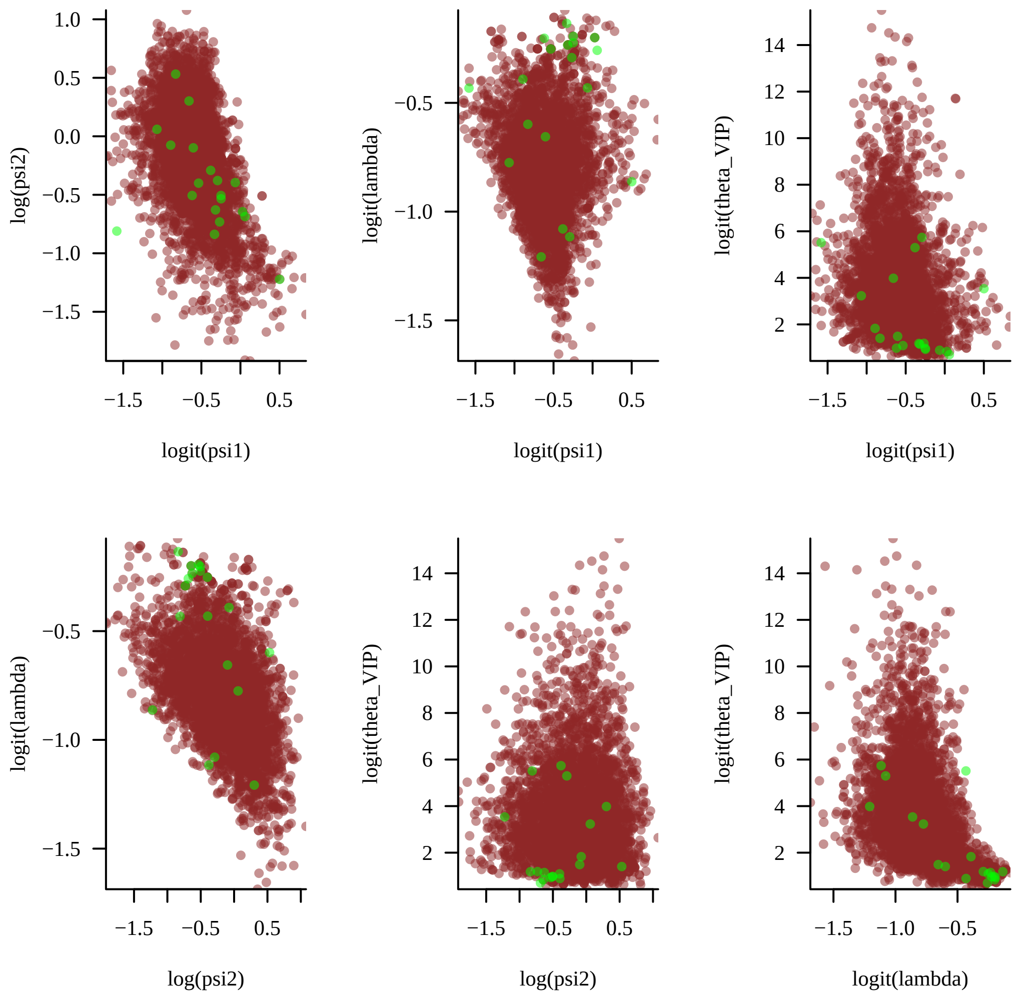
```
  Chain 1: 2 of 1024 transitions (0.2%) diverged.

  Chain 2: 2 of 1024 transitions (0.2%) diverged.
  Chain 3: 8 of 1024 transitions (0.8%) diverged.

  Chain 4: 7 of 1024 transitions (0.7%) diverged.

  Divergent Hamiltonian transitions result from unstable numerical
trajectories.  These instabilities are often due to degenerate target
geometry, especially "pinches".  If there are only a small number of
divergences then running with adept_delta larger than 0.801 may reduce
the instabilities at the cost of more expensive Hamiltonian
transitions.
```

```r
samples3 <- util$extract_expectands(fit)
base_samples <- util$filter_expectands(samples3,
                                       c('psi1', 'psi2',
                                         'lambda', 'theta_VIP'))
util$check_all_expectand_diagnostics(base_samples)
```

```
All expectands checked appear to be behaving well enough for reliable
Markov chain Monte Carlo estimation.
```

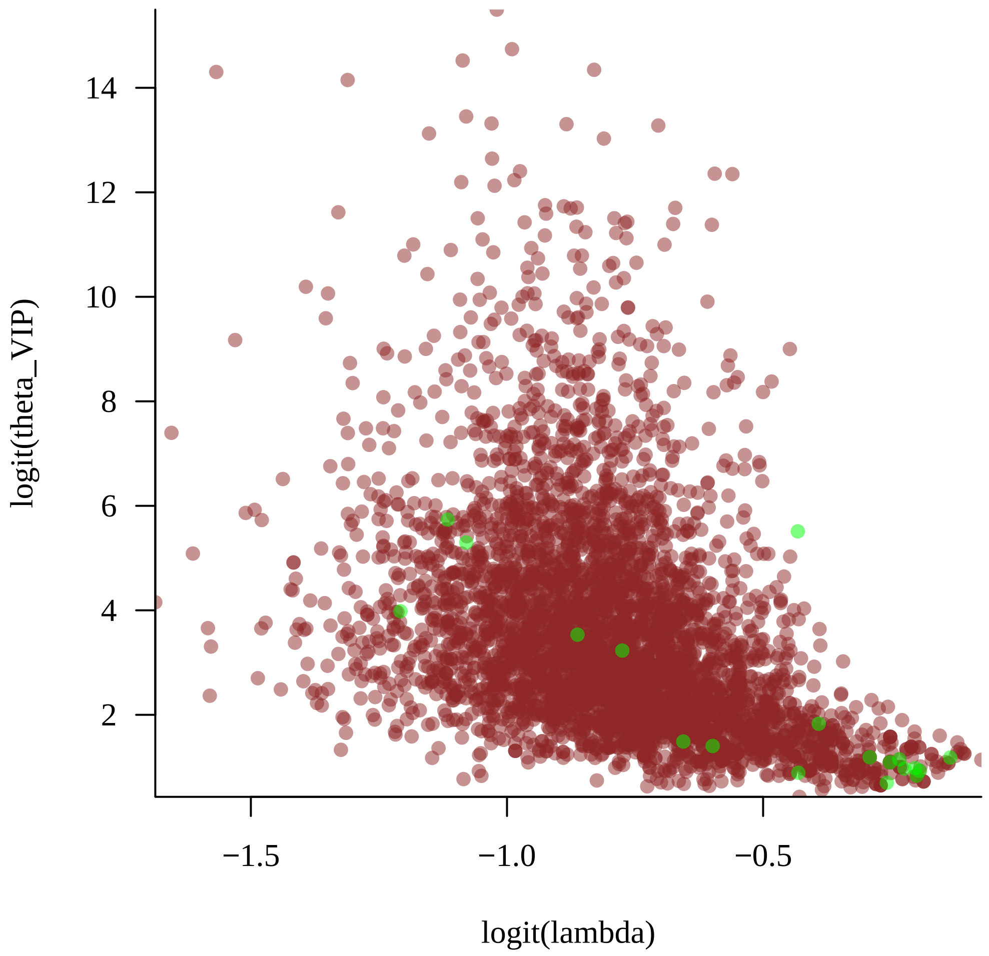
One way that we can identify where problems might be arising in the model configuration space is to examine pairs plots with the divergent and non-divergent Markov chain iterations drawn in different colors. To better assess the computational issues we'll first unconstrain each parameter, mirroring how `Stan` automatically accommodates parameter constraints.

```r
names <- c('psi1', 'psi2', 'lambda', 'theta_VIP')
util$plot_div_pairs(names, names, samples3, diagnostics3,
                    transforms=list("psi1" = 2, "psi2" = 1,
                                    "lambda" = 2, "theta_VIP" = 2))
```



We see that the divergent iterations seem to concentrate in a region of the model configuration space where the posterior samples of logit(λ) and logit(θ^{VIP}) pinch together.

```r
util$plot_div_pairs(c('lambda'), c('theta_VIP'), samples3, diagnostics3,
                    transforms=list("lambda" = 2, "theta_VIP" = 2))
```



The problem here is that our model is now sufficiently flexible that many distinct behaviors are similarly consistent with the observed data. In particular the observations appear to be insensitive to model configurations with a larger proportion of VIP customers and smaller VIP conversion probability and model configurations with a smaller proportion of VIP customers and larger VIP conversion probabilities. The posterior distribution extends out to all of these consistent model configurations, but those uncertainties frustrate accurate computation.

One way to force a more refined, but more expensive, exploration with the dynamic Hamiltonian Monte Carlo sampler in `Stan` is to force a less aggressive step size adaptation. This is done by increasing the `adapt_delta` configuration from it's default value of 0.8. Here we'll try 0.99.

```
fit <- stan(file="stan_programs/model3.stan",
            data=data, seed=8438338,
            warmup=1000, iter=2024, refresh=0,
            control=list('adapt_delta'=0.99))
```

It looks like this may have done the trick as we no longer see any divergence warnings.

```r
diagnostics3_99 <- util$extract_hmc_diagnostics(fit)
util$check_all_hmc_diagnostics(diagnostics3_99)
```

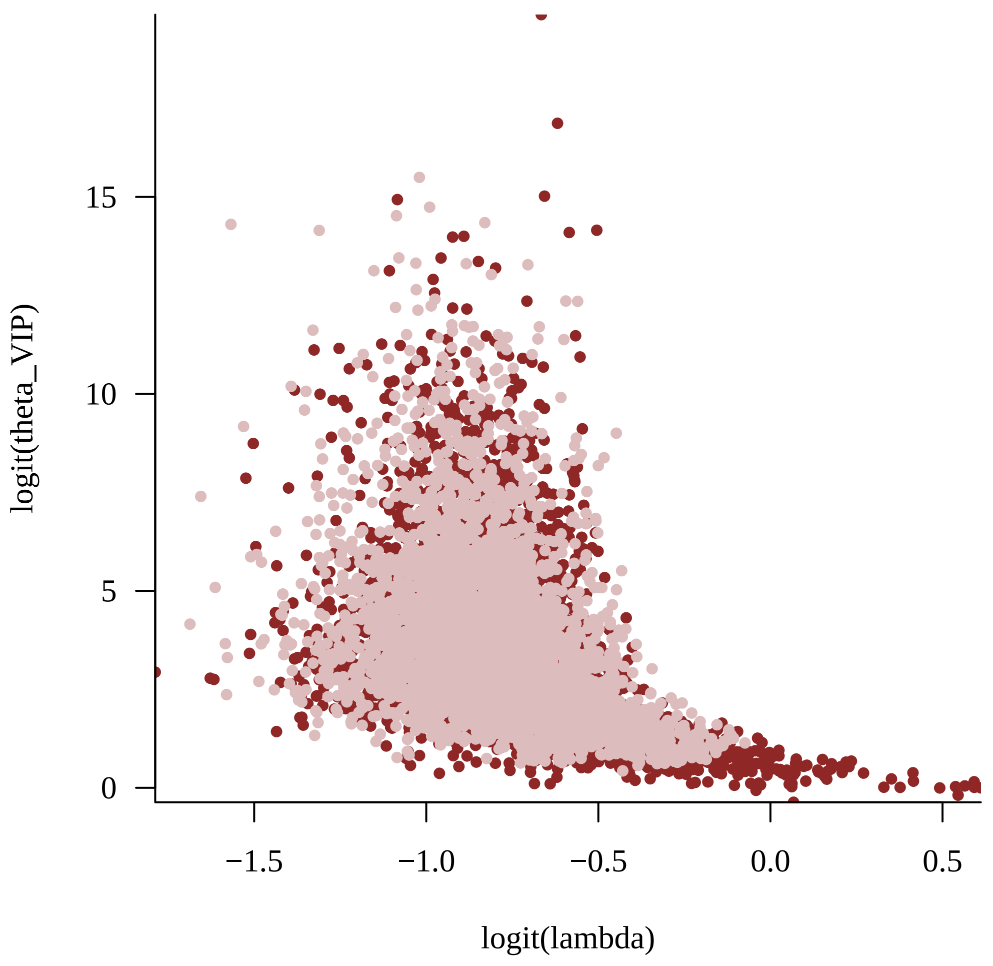
```
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
```

```r
samples3_99 <- util$extract_expectands(fit)
base_samples <- util$filter_expectands(samples3_99,
                                       c('psi1', 'psi2',
                                         'lambda', 'theta_VIP'))
util$check_all_expectand_diagnostics(base_samples)
```

```
All expectands checked appear to be behaving well enough for reliable
Markov chain Monte Carlo estimation.
```

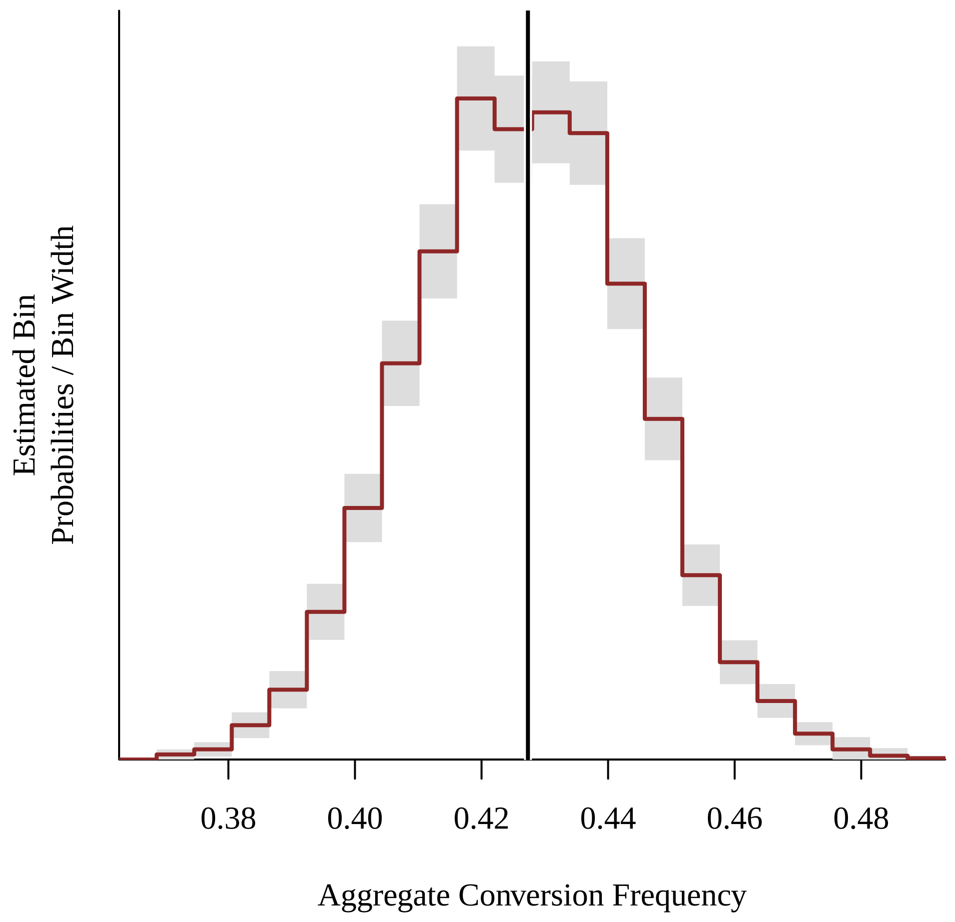
Indeed comparing the two posterior quantifications we can see that using a less aggressive adaptation allows us to capture more model behaviors than before. These new model configurations have always been consistent with the observed data, they were just ignored by our first, inaccurate fit.

```r
par(mfrow=c(1, 1))

logit <- function(ps) {
  log(ps / (1 - ps))
}

plot(logit(c(samples3_99['lambda'], recursive=TRUE)),
     logit(c(samples3_99['theta_VIP'], recursive=TRUE)),
     col=util$c_dark, pch=16, cex=0.8,
     xlab="logit(lambda)", ylab="logit(theta_VIP)")

points(logit(c(samples3['lambda'], recursive=TRUE)),
       logit(c(samples3['theta_VIP'], recursive=TRUE)),
       col=util$c_light, pch=16, cex=0.8)
```



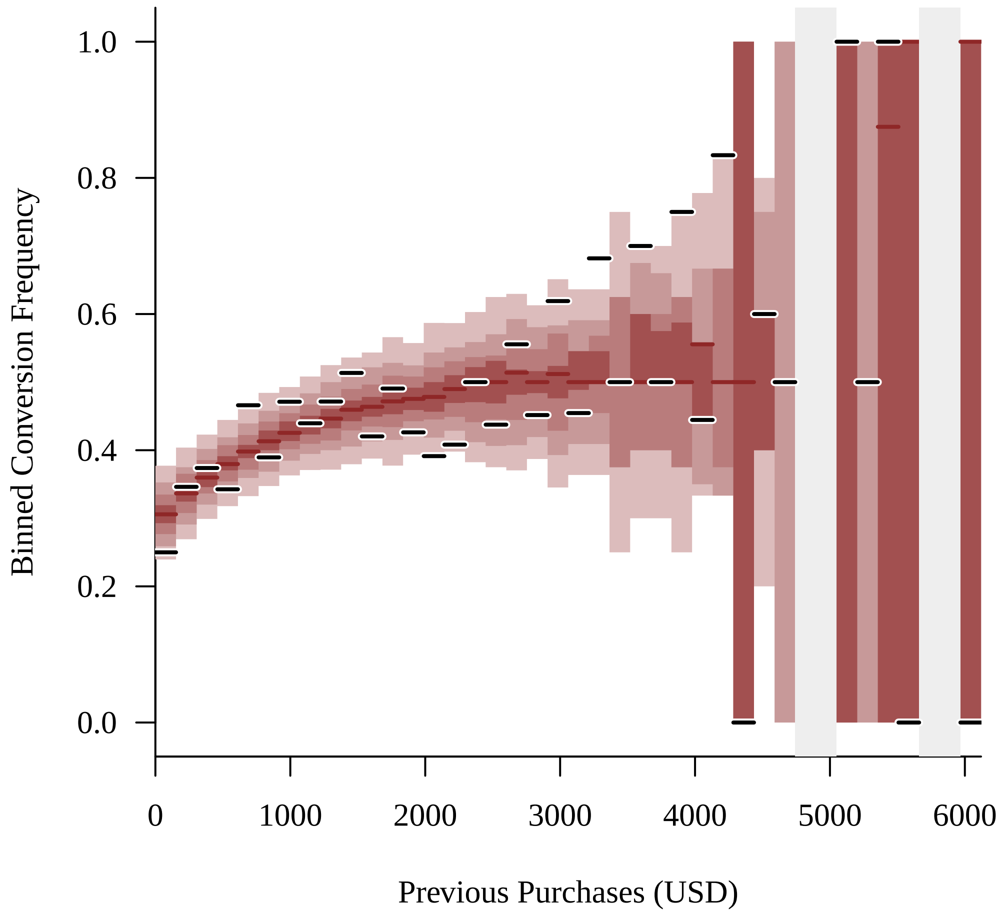
With an accurate posterior quantification we can consider the retrodictive performance of this model. The aggregate conversion frequency continues to show no signs of retrodictive tension.

```
par(mfrow=c(1, 1), mar=c(5, 5, 3, 1))

util$plot_expectand_pushforward(samples3_99[['p_hat_pred']], 20,
                                display_name="Aggregate Conversion Frequency",
                                baseline=mean(data$y))
```



Now, however, the retrodictive behavior for the binned conversion frequencies is much better. Allowing for two groups of customers has finally allowed our model to adequately recover the behavior of the observed data.

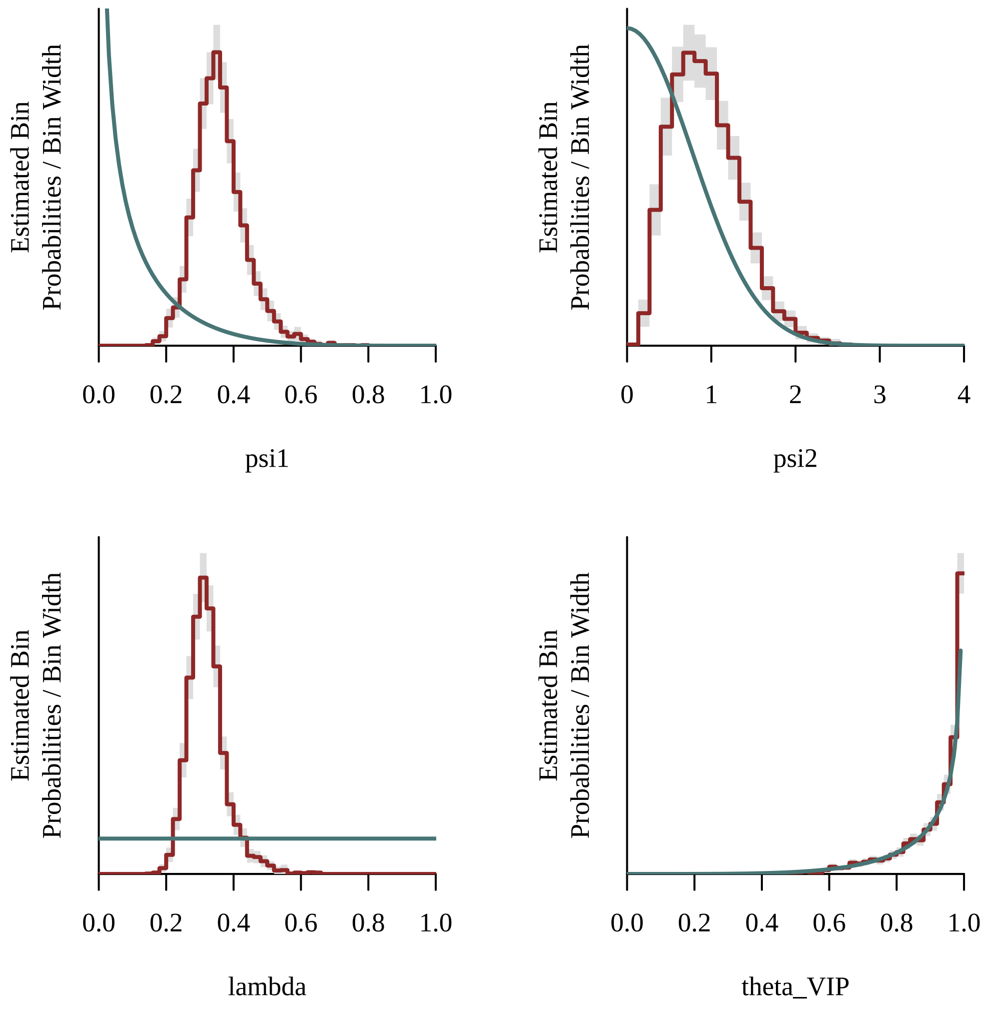
```
par(mfrow=c(1, 1), mar=c(5, 5, 3, 1))

pred_names <- sapply(1:data$N, function(n) paste0('y_pred[', n, ']'))
util$plot_conditional_mean_quantiles(samples3_99, pred_names, data$x,
                                     0, 6120, 153,
                                     baseline_values=data$y,
                                     xlab="Previous Purchases (USD)",
                                     ylab="Binned Conversion Frequency")
```



That said the actual posterior uncertainties each all of the parameters except for λ are pretty strong, limiting how well we can inform decisions, predictions, and the like.

```
par(mfrow=c(2, 2), mar=c(5, 5, 1, 1))

util$plot_expectand_pushforward(samples3_99[['psi1']], 50,
                                display_name="psi1",
                                flim=c(0, 1))
xs <- seq(0, 1, 0.01)
lines(xs, dbeta(xs, 0.5, 5.0), lwd=2, col=util$c_mid_teal)

util$plot_expectand_pushforward(samples3_99[['psi2']], 30,
                                display_name="psi2",
                                flim=c(0, 4))
xs <- seq(0, 4, 0.05)
lines(xs, 2 * dnorm(xs, 0, 2.0 / 2.57), lwd=2, col=util$c_mid_teal)
```

```
util$plot_expectand_pushforward(samples3_99[['lambda']], 50,
                                display_name="lambda",
                                flim=c(0, 1))
xs <- seq(0, 1, 0.01)
lines(xs, dbeta(xs, 1, 1), lwd=2, col=util$c_mid_teal)

util$plot_expectand_pushforward(samples3_99[['theta_VIP']], 50,
                                display_name="theta_VIP",
                                flim=c(0, 1))
xs <- seq(0, 1, 0.01)
lines(xs, dbeta(xs, 5.0, 0.5), lwd=2, col=util$c_mid_teal)
```



We don't seem to have the enough information to resolve the individual behaviors for the two data generating processes needed to adequate fit the observed data.

4.4 Model 4

The benefit of Bayesian inference, at least when implemented with accurate computation, is that we use *all* of the model configurations consistent with the observed data and the domain expertise encoded in the prior model to inform summaries, decisions, predictions, and the like. This ensures that we don't for example make fragile statements or risky decisions when we are burdened with strong posterior uncertainties. That said robustness to large uncertainties doesn't mean that smaller uncertainties are not still preferable!

Here our ignorance about the customer categorization dilutes how well the data separately inform the two component behaviors. An immediate way to improve our inferences is to incorporate additional information about either behavior. In a Bayesian analysis this information can be introduced through either the prior model or the observational model.

For example let's say that upon hearing about our inferential frustrations a colleague mentions the existence of an auxiliary set of observed conversions

$$y_1^{\text{aux}}, \ldots, y_{N^{\text{aux}}}^{\text{aux}}$$

that is sensitive to *only* VIP customers.

```r
aux_data <- read_rdump('data/aux_logs.data.R')

names(aux_data)
```

```
[1] "N_aux" "y_aux"
```

Indeed these observations exhibit very different aggregate behavior than our initial observations.

```r
table(data$y)
```

```

  0   1 
859 641 
```

```r
table(aux_data$y_aux)
```

```

  0   1 
 14 186 
```

If we can integrate a data generating process for this new data and our current model into a single narratively generative model then these auxiliary observations will directly inform θ^{VIP}, allowing our initial observations to better inform λ and $\psi = (\psi_1, \psi_2)$ all while taking into account the subtle coupling and even any redundancies between the component data generating processes. Because this new data depends on only θ^{VIP} building a consistent joint model is particularly straightforward (Figure 4),

$$
\begin{aligned}
\pi(y_1, \ldots, y_N, &y_1^{\text{aux}}, \ldots, y_{N^{\text{aux}}}^{\text{aux}}, \psi, \theta^{\text{VIP}}, \lambda; x_1, \ldots, x_N) \\
&\quad\quad \pi(y_1, \ldots, y_N \mid \psi, \theta^{\text{VIP}}, \lambda, x_1, \ldots, x_N) \\
&\quad\quad \cdot \pi(y_1^{\text{aux}}, \ldots, y_{N^{\text{aux}}}^{\text{aux}} \mid \theta^{\text{VIP}}) \\
&\quad\quad \cdot \pi(\psi, \theta^{\text{VIP}}, \lambda) \\
&= \left[\prod_{n=1}^{N} \lambda \cdot \text{Bernoulli}\left(y_n \mid \theta\left(x_n, \psi\right)\right) + (1 - \lambda) \cdot \text{Bernoulli}\left(y_n \mid \theta^{\text{VIP}}\right) \right] \\
&\quad\quad \cdot \left[\prod_{n=1}^{N^{\text{aux}}} \text{Bernoulli}\left(y_n^{\text{aux}} \mid \theta^{\text{VIP}}\right) \right] \\
&\quad\quad \cdot \pi(\psi)\, \pi(\theta^{\text{VIP}})\, \pi(\lambda).
\end{aligned}
$$

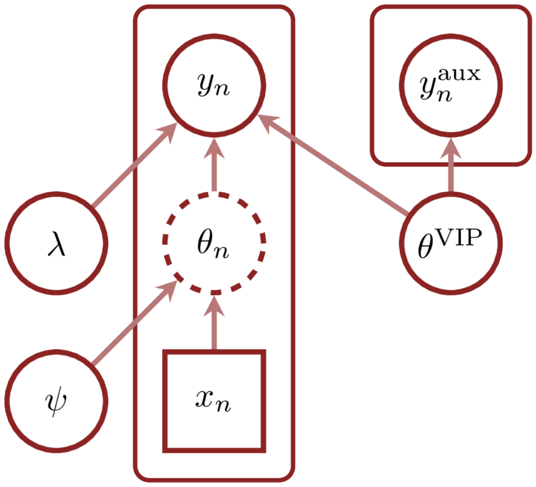


Figure 4: A joint model that incorporates multiple data generating processes allows us to combine disparate sources of data together into more precise, but consistent, inferences. Here we combine our initial observations that could have been generated from one of two data generating processes with auxiliary observations that are unambiguously generated from a third data generating process that depends on only θ^{VIP}.

We can now implement this expanded model in the `Stan` modeling language, being careful to also include posterior predictions for the new observations.

```r
data <- c(data, aux_data)

fit <- stan(file="stan_programs/model4.stan",
            data=data, seed=8438338,
            warmup=1000, iter=2024, refresh=0)
```

Pleasantly we no longer see signs of computational issues when working with the default adaptation settings. This suggests that the additional data may indeed have improved the posterior uncertainties.

```r
diagnostics4 <- util$extract_hmc_diagnostics(fit)
util$check_all_hmc_diagnostics(diagnostics4)
```

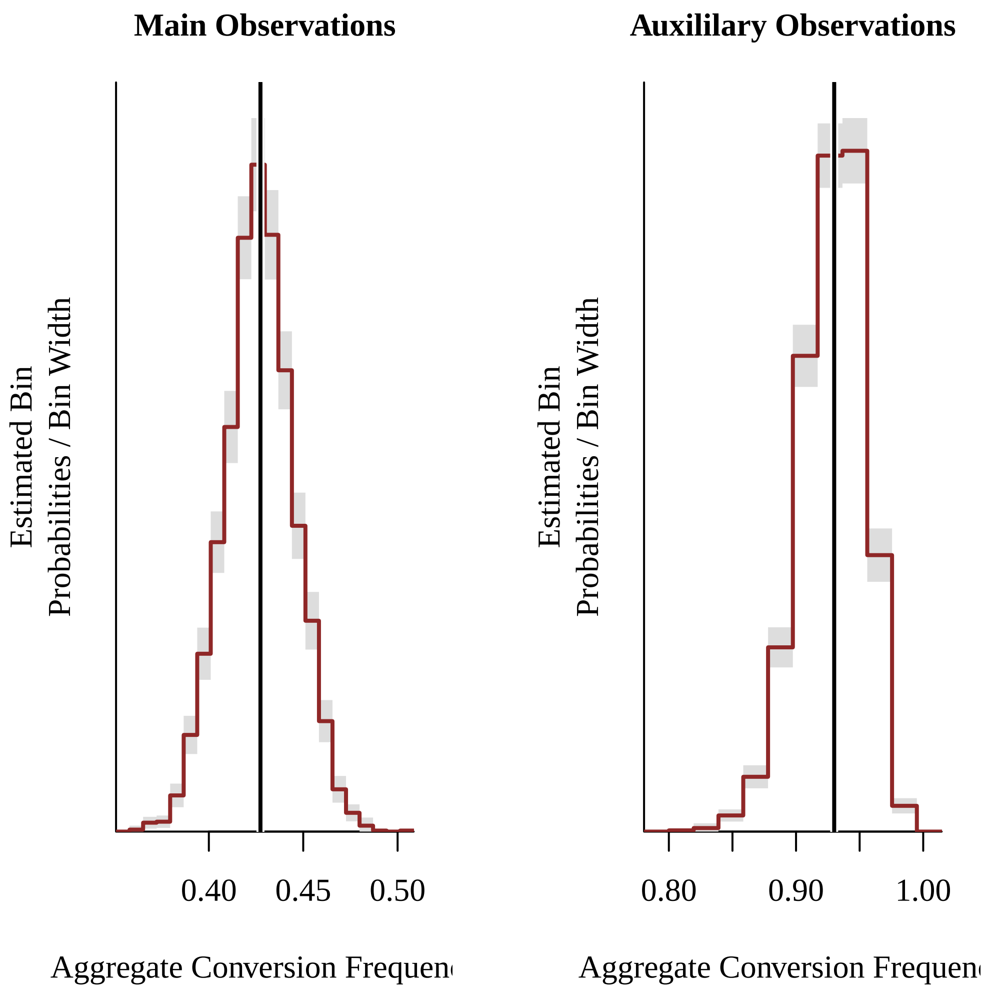
```
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
```

```r
samples4 <- util$extract_expectands(fit)
base_samples <- util$filter_expectands(samples4,
                                       c('psi1', 'psi2',
                                         'lambda', 'theta_VIP'))
util$check_all_expectand_diagnostics(base_samples)
```

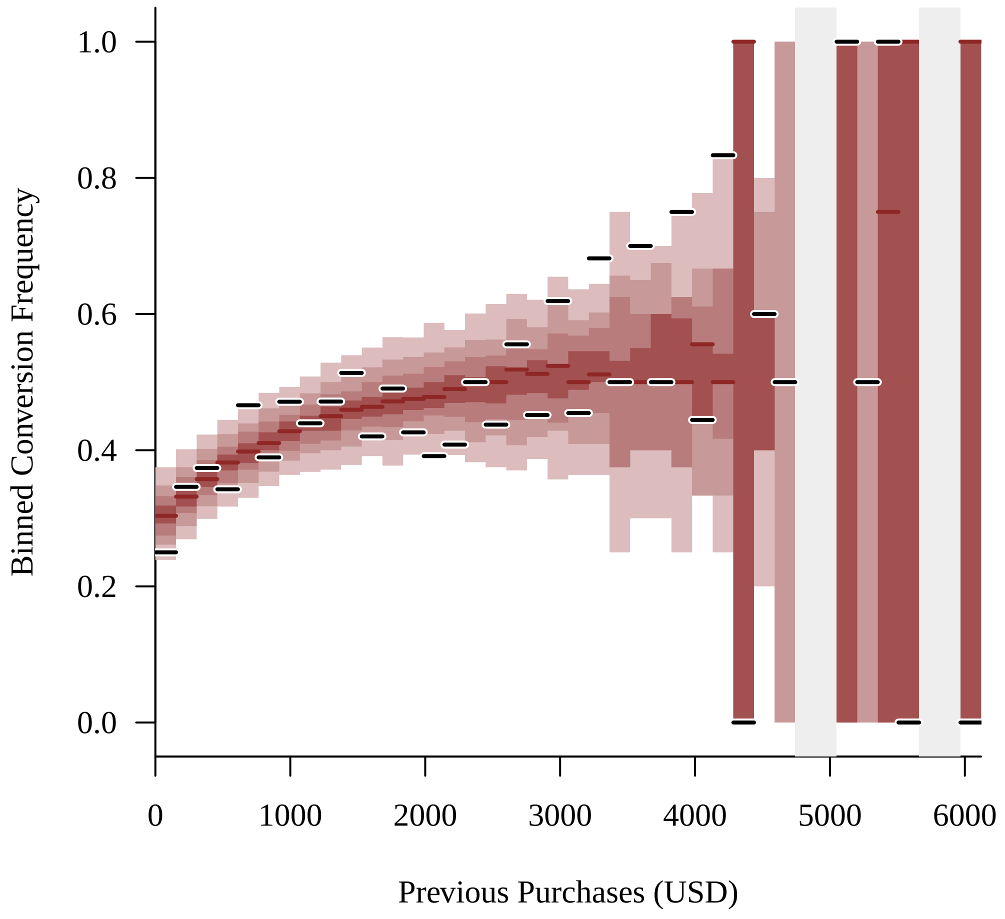
```
All expectands checked appear to be behaving well enough for reliable
Markov chain Monte Carlo estimation.
```

None of our visual retrodictive checks indicate any model inadequacies, including the aggregate conversion frequency of the newly incorporated data.

```r
par(mfrow=c(1, 2), mar=c(5, 5, 3, 1))

util$plot_expectand_pushforward(samples4[['p_hat_pred']], 20,
                                display_name="Aggregate Conversion Frequency",
                                baseline=mean(data$y),
                                main="Main Observations")

util$plot_expectand_pushforward(samples4[['p_hat_aux_pred']], 10,
                                display_name="Aggregate Conversion Frequency",
                                baseline=mean(data$y_aux),
                                main="Auxililary Observations")
```



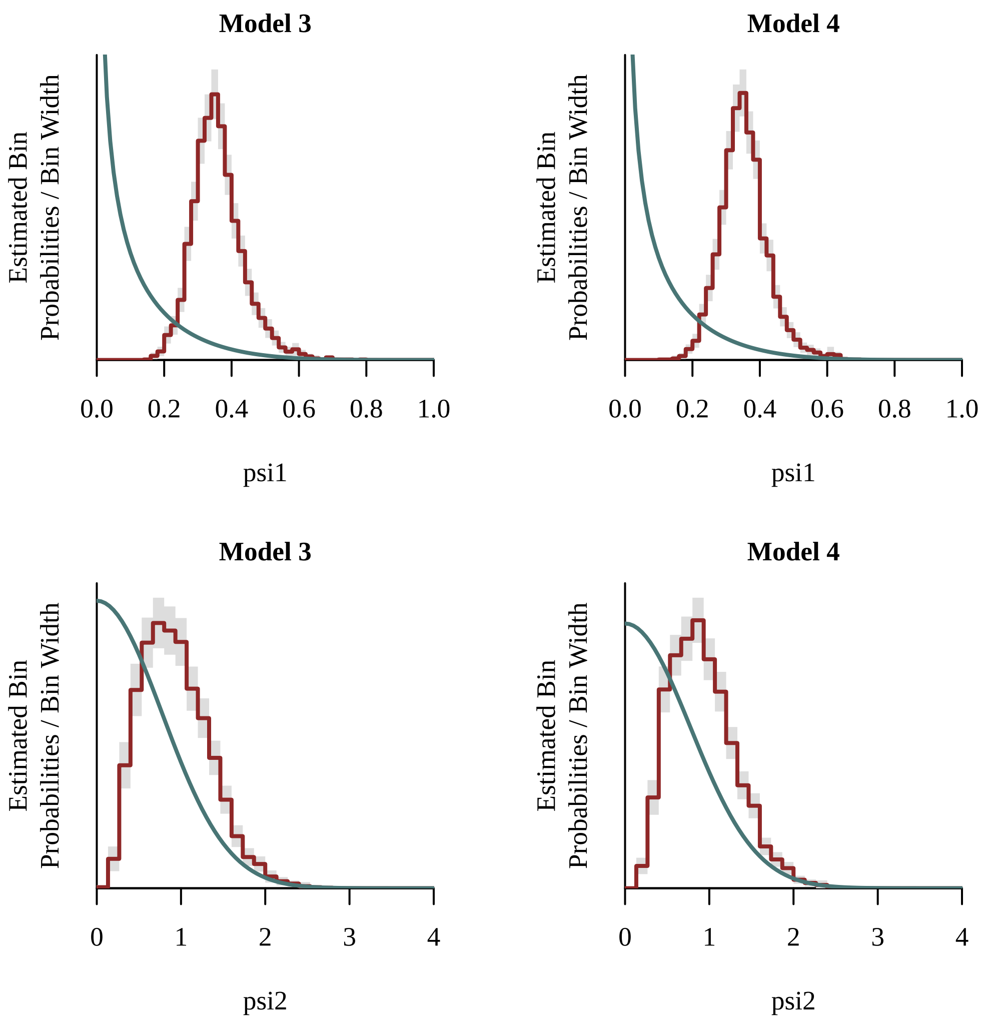
```r
par(mfrow=c(1, 1), mar=c(5, 5, 3, 1))

pred_names <- sapply(1:data$N, function(n) paste0('y_pred[', n, ']'))
util$plot_conditional_mean_quantiles(samples4, pred_names, data$x,
                                     0, 6120, 153,
                                     baseline_values=data$y,
                                     xlab="Previous Purchases (USD)",
                                     ylab="Binned Conversion Frequency")
```



Have our posterior uncertainties actually been reduced by the addition of these auxiliary observations? The marginal posterior inferences for ψ_1 and ψ_2 are similar to what we encountered previously.

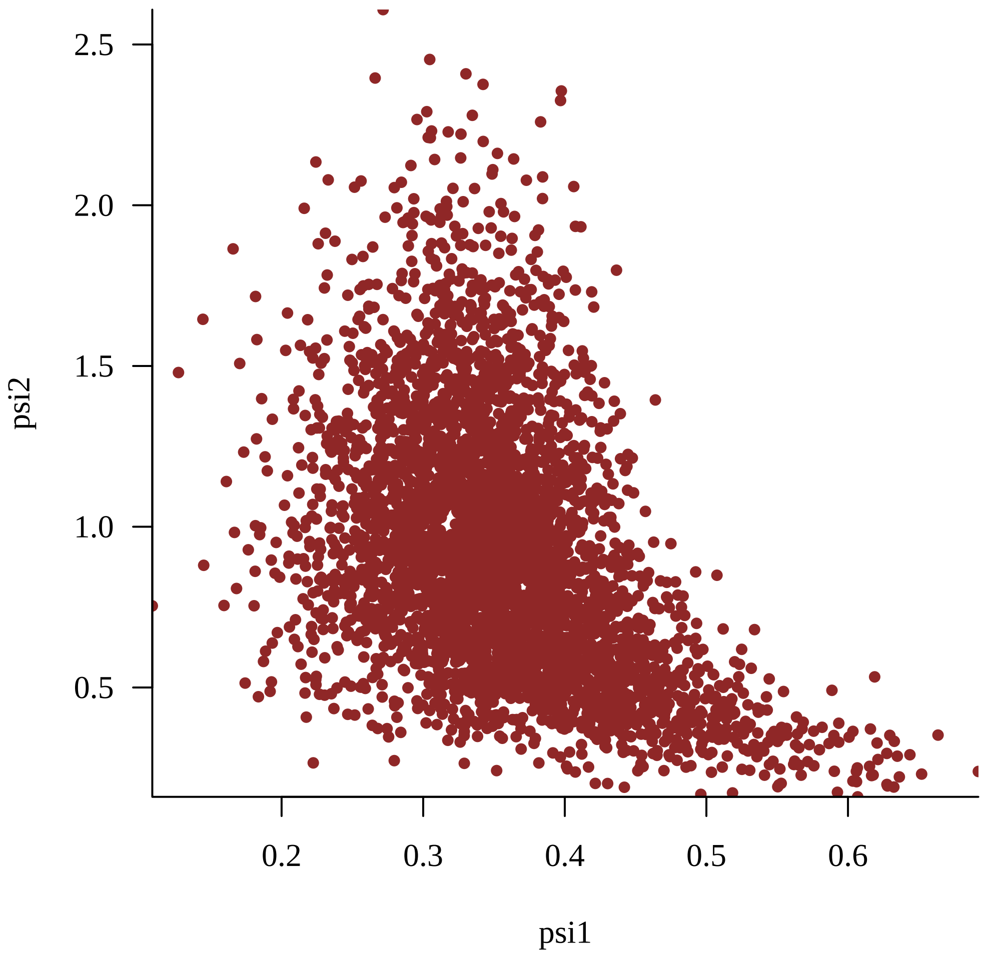
```
par(mfrow=c(2, 2), mar=c(5, 5, 2, 1))

util$plot_expectand_pushforward(samples3_99[['psi1']], 50,
                                display_name="psi1",
                                flim=c(0, 1),
                                main="Model 3")
xs <- seq(0, 1, 0.01)
lines(xs, dbeta(xs, 0.5, 5.0), lwd=2, col=util$c_mid_teal)

util$plot_expectand_pushforward(samples4[['psi1']], 50,
                                display_name="psi1",
                                flim=c(0, 1),
                                main="Model 4")
```

```r
xs <- seq(0, 1, 0.01)
lines(xs, dbeta(xs, 0.5, 5.0), lwd=2, col=util$c_mid_teal)

util$plot_expectand_pushforward(samples3_99[['psi2']], 30,
                                display_name="psi2",
                                flim=c(0, 4),
                                main="Model 3")
xs <- seq(0, 4, 0.05)
lines(xs, 2 * dnorm(xs, 0, 2.0 / 2.57), lwd=2, col=util$c_mid_teal)

util$plot_expectand_pushforward(samples4[['psi2']], 30,
                                display_name="psi2",
                                flim=c(0, 4),
                                main="Model 4")
xs <- seq(0, 4, 0.05)
lines(xs, 2 * dnorm(xs, 0, 2.0 / 2.57), lwd=2, col=util$c_mid_teal)
```



Looking at the corresponding pairs plot we can see that there are many different configurations of the conversion probability function consistent with the observed data and our domain expertise: larger maximum conversion probabilities can be accommodated by reducing the speed of saturation and vice versa. In order to better inform these two parameters we would probably need to collect more conversion data at larger previous purchases.

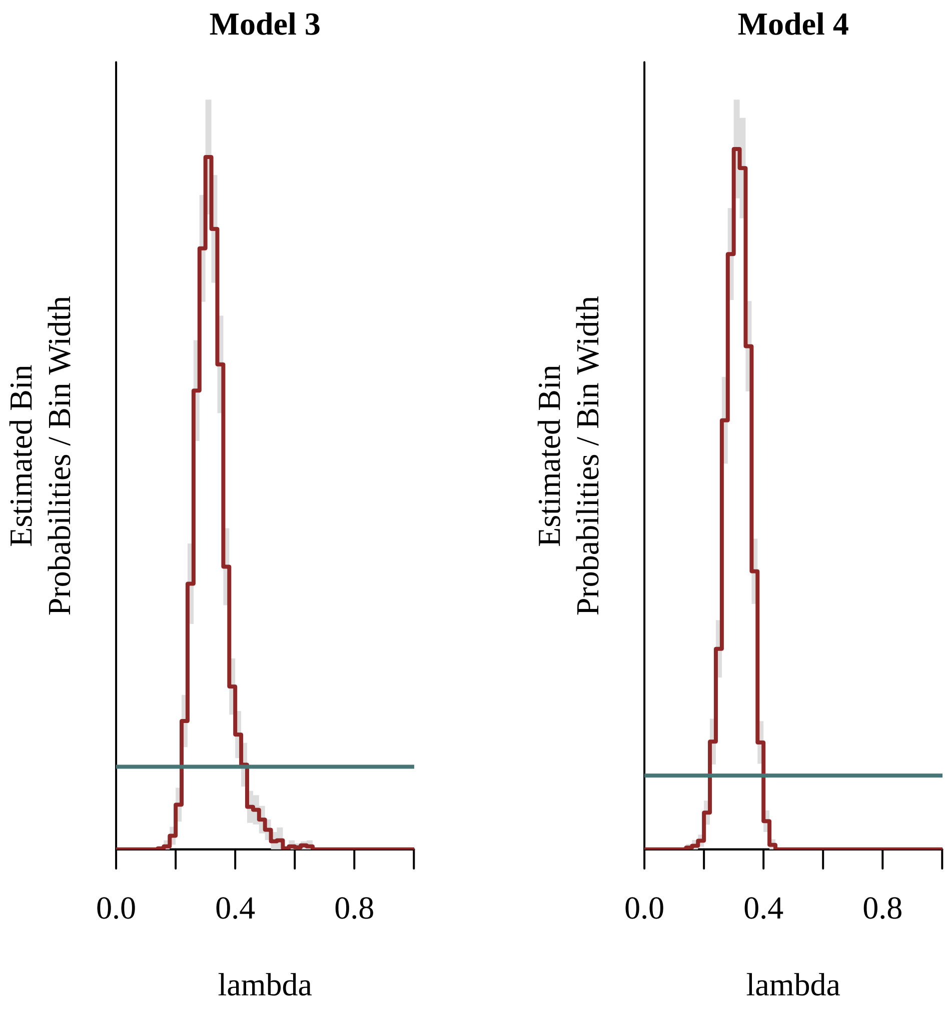
```r
par(mfrow=c(1, 1))

plot(c(samples4['psi1'], recursive=TRUE),
     c(samples4['psi2'], recursive=TRUE),
     col=util$c_dark, pch=16, cex=0.8,
     xlab="psi1", ylab="psi2")
```



On the other hand the marginal posterior distribution for λ suppresses large values than before.

```
par(mfrow=c(1, 2), mar=c(5, 5, 2, 1))

util$plot_expectand_pushforward(samples3_99[['lambda']], 50,
                                display_name="lambda",
                                flim=c(0, 1),
                                main="Model 3")
xs <- seq(0, 1, 0.01)
lines(xs, dbeta(xs, 1, 1), lwd=2, col=util$c_mid_teal)

util$plot_expectand_pushforward(samples4[['lambda']], 50,
                                display_name="lambda",
                                flim=c(0, 1),
                                main="Model 4")
```

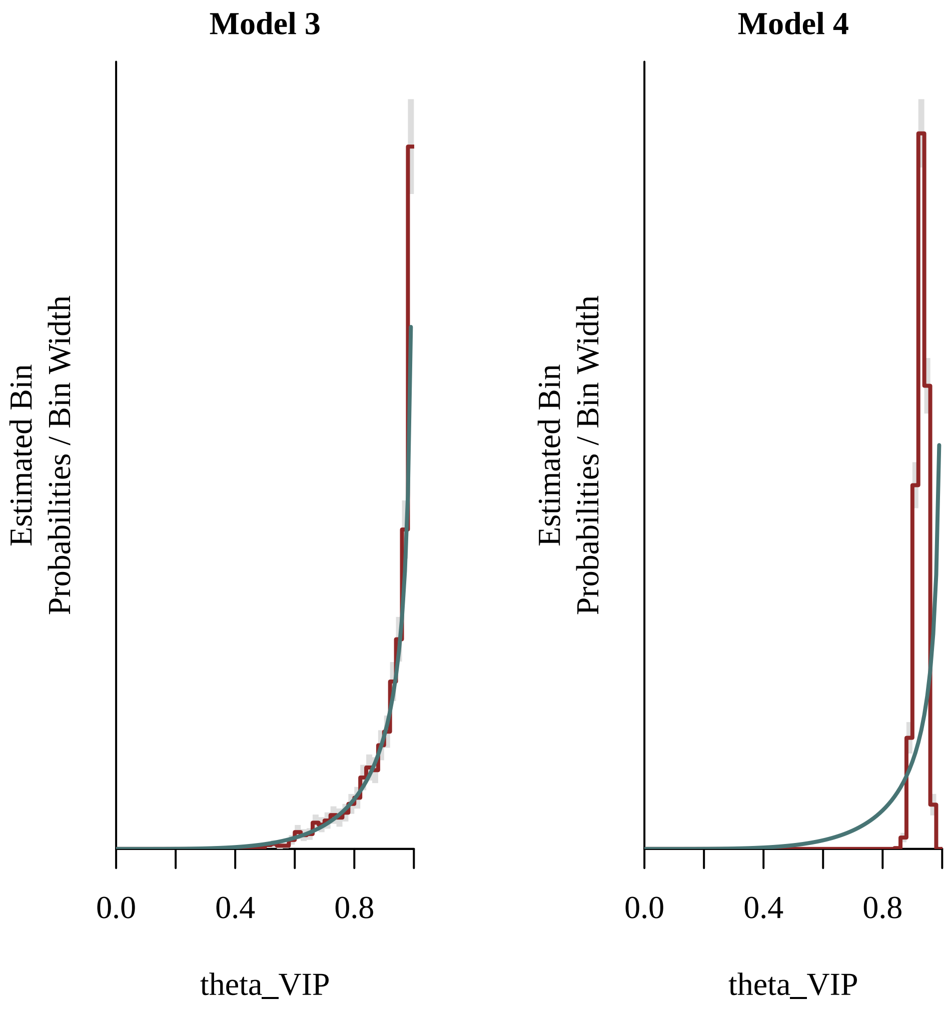
```
xs <- seq(0, 1, 0.01)
lines(xs, dbeta(xs, 1, 1), lwd=2, col=util$c_mid_teal)
```



More importantly the marginal posterior distribution for θ^{VIP} now strongly contracts away from the prior model.

```
par(mfrow=c(1, 2), mar=c(5, 5, 2, 1))
util$plot_expectand_pushforward(samples3_99[['theta_VIP']], 50,
                                display_name="theta_VIP",
                                flim=c(0, 1),
                                main="Model 3")
xs <- seq(0, 1, 0.01)
lines(xs, dbeta(xs, 5.0, 0.5), lwd=2, col=util$c_mid_teal)
```

```r
util$plot_expectand_pushforward(samples4[['theta_VIP']], 50,
                                display_name="theta_VIP",
                                flim=c(0, 1),
                                main="Model 4")
xs <- seq(0, 1, 0.01)
lines(xs, dbeta(xs, 5.0, 0.5), lwd=2, col=util$c_mid_teal)
```



4.5 Hypothetical Predictions

Now that we can reasonably disentangle all of the data generating behaviors we can use them to inform inferences, decisions, and predictions about new circumstances. These tasks are ubiquitous in both science and industry but they are denoted by a wildly confusing array of adjectives, such as hypothetical, counterfactual, out-of-sample, generalized, and the like.

For example let's say that we want to understand what would happen if we could redesign the customer experience to exclude VIP customers entirely. In this case the behavior of the regular customers would not be properly modeled by

$$\text{Bernoulli}\left(y_n \mid \theta\left(x_n, \psi\right)\right) + (1 - \lambda) \cdot \text{Bernoulli}\left(y_n \mid \theta^{\text{VIP}}\right)$$

nor

$$\text{Bernoulli}\left(y_n^{\text{aux}} \mid \theta^{\text{VIP}}\right)$$

but rather

$$\text{Bernoulli}\left(y_n \mid \theta\left(x_n, \psi\right)\right).$$

Fortunately we can readily expand our model to accommodate this new behavior (Figure 5).

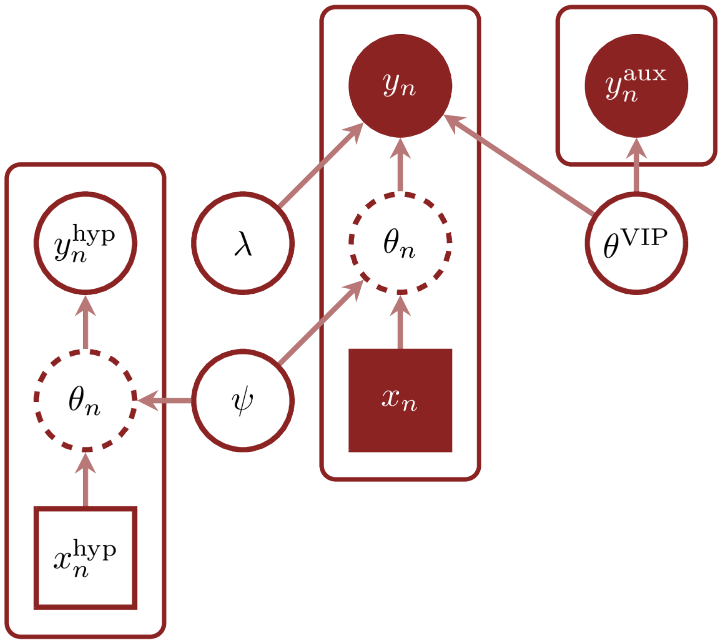


Figure 5: Narratively generative modeling allows us to consider not just the data generating processes responsible for observed data but also those that might be responsible for data to which we do not yet have access, such as the observations that would be generated by hypothetical or future measurements.

In order to actually implement these new, hypothetical predictions we need to determine appropriate values for the previous purchases. This is complicated by the fact that we have been explicitly modeling these values! Although this might initially come across as a burden it can actually be a really useful opportunity. For example if we want to analyze the behavior across all possible previous purchases we can select previous purchases from a uniform grid of values.

```
x_grid <- seq(0, 5000, 125)
data$N_hyp <- 2050
data$x_hyp <- rep(x_grid, each=50)
```

Incorporating these new predictions into a `Stan` program just requires expanding the `data` and `generated quantities` blocks. The `parameters` and `model` blocks remain exactly the same. We are not changing our inferences, just what we do with that information.

```
fit <- stan(file="stan_programs/model5.stan",
            data=data, seed=8438338,
            warmup=1000, iter=2024, refresh=0)

samples5 <- util$extract_expectands(fit)
```

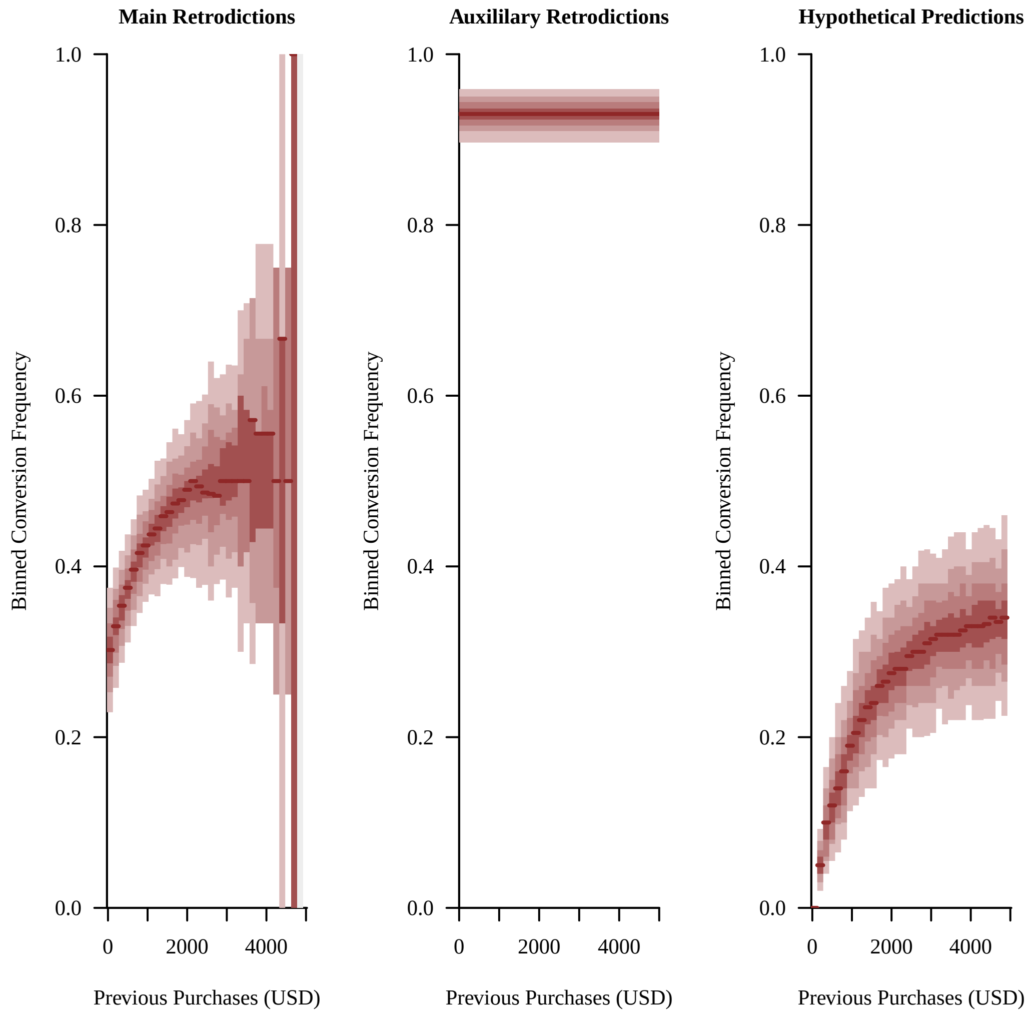
Because we are effectively rerunning the same computation we don't need to check the computational diagnostics nor the posterior retrodictive checks.

As expected our predictions for the new circumstances don't look like any of our retrodictions. The hypothetical data generating process yields conversion frequencies that increase but quickly saturate with increasing previous purchases but are not contaminated by VIP customers. This results in conversion frequencies that vary with previous purchases but are uniformly smaller than what we see in the observed data.

```
par(mfrow=c(1, 3), mar=c(5, 5, 3, 1))

pred_names <- sapply(1:data$N, function(n) paste0('y_pred[', n, ']'))
util$plot_conditional_mean_quantiles(samples5, pred_names, data$x,
                                     0, 5000, 150,
                                     xlab="Previous Purchases (USD)",
                                     ylab="Binned Conversion Frequency",
                                     display_ylim=c(0, 1),
                                     main="Main Retrodictions")
```

```
Warning in check_bin_containment(bin_min, bin_max, obs_xs, "conditioning
value"): 6 conditioning values (0.4%) fell above the binning.
```

```
pred_names <- sapply(1:data$N_aux, function(n) paste0('y_aux_pred[', n, ']'))
probs <- c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9)

mean_quantiles <- rep(0, 9)
for (c in 1:4) {
  values <- t(sapply(pred_names, function(name) samples5[[name]][c,]))
  means <- sapply(1:1024, function(s) mean(values[,s]))
  mean_quantiles <- mean_quantiles + quantile(means, probs=probs) / 4
}

plot(1, type="n", main="Auxililary Retrodictions",
     xlim=c(0, 5000), xlab="Previous Purchases (USD)",
     ylim=c(0, 1), ylab="Binned Conversion Frequency")

polygon(c(0, 5000, 5000, 0),
        c(mean_quantiles[1], mean_quantiles[1],
          mean_quantiles[9], mean_quantiles[9]),
        col=util$c_light, border = NA)
polygon(c(0, 5000, 5000, 0),
        c(mean_quantiles[2], mean_quantiles[2],
          mean_quantiles[8], mean_quantiles[8]),
        col=util$c_light_highlight, border = NA)
polygon(c(0, 5000, 5000, 0),
        c(mean_quantiles[3], mean_quantiles[3],
          mean_quantiles[7], mean_quantiles[7]),
        col=util$c_mid, border = NA)
polygon(c(0, 5000, 5000, 0),
        c(mean_quantiles[4], mean_quantiles[4],
          mean_quantiles[6], mean_quantiles[6]),
        col=util$c_mid_highlight, border = NA)
lines(c(0, 5000), c(mean_quantiles[5], mean_quantiles[5]),
          col=util$c_dark, lwd=2)


pred_names <- sapply(1:data$N_hyp, function(n) paste0('y_hyp_pred[', n, ']'))
util$plot_conditional_mean_quantiles(samples5, pred_names, data$x_hyp,
                                     0, 5000, 150,
                                     xlab="Previous Purchases (USD)",
                                     ylab="Binned Conversion Frequency",
                                     display_ylim=c(0, 1),
                                     main="Hypothetical Predictions")
```



5 Conclusion

While I have intentionally made this example as simple as possible the modeling principles that is demonstrates generalize to much more sophisticated problems.

For example we might be interested in the interaction between individual customers and individual products in which case we would need to model the particular proclivity of a potential customer to certain kinds of products or even certain product properties and how that proclivity manifests in observed sales, survey outcomes, focus group feedback, or even combinations thereof. Once we have developed an adequate model we could then use it to predict how sales change as customer behavior and/or product behavior inevitably evolves.

While we have been using a marketing example these ideas are just are relevant to science. In order to inform robust scientific insights we need to model not just the phenomena of interest but also the experiments that we use to probe them. To consistently accumulate knowledge across multiple experiments we need to account for their idiosyncrasies. Similarly to make accurate predictions about the performance of new experiments we need to account for our uncertainty in the relevant phenomenological and the potential experimental behaviors.

As with any tool narratively generative modeling requires experience to use as effectively as possible. All we have to do to start building that experience, however, is to actively consider what data generating processes we want to model in a given application and what data generating processes do our analyses implicitly assume.

Acknowledgements

A very special thanks to everyone supporting me on Patreon: Adam Fleischhacker, Adriano Yoshino, Alessandro Varacca, Alexander Noll, Alexander Rosteck, Andrea Serafino, Andrew Mascioli, Andrew Rouillard, Andrew Vigotsky, Ara Winter, Austin Rochford, Avraham Adler, Ben Matthews, Ben Swallow, Benoit Essiambre, Bertrand Wilden, Bradley Kolb, Brandon Liu, Brendan Galdo, Brynjolfur Gauti Jónsson, Cameron Smith, Canaan Breiss, Cat Shark, CG, Charles Naylor, Charles Shaw, Chase Dwelle, Chris Jones, Christopher Mehrvarzi, Colin Carroll, Colin McAuliffe, Damien Mannion, dan mackinlay, Dan W Joyce, Dan Waxman, Dan Weitzenfeld, Daniel Edward Marthaler, Daniel Saunders, Darshan Pandit, Darthmaluus , David Galley, David Wurtz, Doug Rivers, Dr. Jobo, Dr. Omri Har Shemesh, Dylan Maher, Ed Cashin, Edgar Merkle, Eric LaMotte, Ero Carrera, Eugene O'Friel, Felipe González, Fergus Chadwick, Finn Lindgren, Florian Wellmann, Geoff Rollins, Håkan Johansson, Hamed Bastan-Hagh, Hauke Burde, Hector Munoz, Henri Wallen, hs, Hugo Botha, Ian, Ian Costley, idontgetoutmuch, Ignacio Vera, Ilaria Prosdocimi, Isaac Vock, J, J Michael Burgess, jacob pine, Jair Andrade, James Hodgson, James Wade, Janek Berger, Jason Martin, Jason Pekos, Jason Wong, jd, Jeff Burnett, Jeff Dotson, Jeff Helzner, Jeffrey Erlich, Jessica Graves, Joe Sloan, Joe Wagner, John Flournoy, Jonathan H. Morgan, Jonathon Vallejo, Joran Jongerling, JU, Justin Bois, Kádár András, Karim Naguib, Karim Osman, Kejia Shi, Kristian Gårdhus Wichmann, Lars Barquist, lizzie , LOU ODETTE, Luís F, Marcel Lüthi, Marek Kwiatkowski, Mark Donoghoe, Markus P., Márton Vaitkus, Matt Moores, Matthew, Matthew Kay, Matthieu LEROY, Mattia Arsendi, Maurits van der Meer, Michael DeWitt, Michael Dillon, Michael Lerner, Mick Cooney, Mike Lawrence, N Sanders, N.S. , Name, Nathaniel Burbank, Neel, Nic Fishman, Nicholas Clark, Nicholas Cowie, Nick S, Octavio Medina, Ole Rogeberg, Oliver Crook, Patrick Kelley, Patrick Boehnke, Pau Pereira Batlle, Peter Johnson, Pieter van den Berg, ptr, Ramiro Barrantes Reynolds, Raúl Peralta Lozada, Ravin Kumar, Rémi , Rex Ha, Riccardo Fusaroli, Richard Nerland, Robert Frost, Robert Goldman, Robert kohn, Robin Taylor, Ryan Grossman, Ryan Kelly, S Hong, Sean Wilson, Sergiy Protsiv, Seth Axen, shira, Simon Duane, Simon Lilburn, sssz, Stan_user, Stephen Lienhard, Stew Watts, Stone Chen, Susan Holmes, Svilup, Tao Ye, Tate Tunstall, Tatsuo Okubo, Teresa Ortiz, Theodore Dasher, Thomas Siegert, Thomas Vladeck, Tobychev , Tomáš Frýda, Tony Wuersch, Virginia Fisher,

Vladimir Markov, Wil Yegelwel, Will Farr, woejozney, yolhaj , yureq , Zach A, Zad Rafi, and Zhengchen Cai.

License

Original Computing Environment

```r
writeLines(readLines(file.path(Sys.getenv("HOME"), ".R/Makevars")))
```

```
CC=clang

CXXFLAGS=-O3 -mtune=native -march=native -Wno-unused-variable -Wno-unused-function -Wno-macr
CXX=clang++ -arch x86_64 -ftemplate-depth-256

CXX14FLAGS=-O3 -mtune=native -march=native -Wno-unused-variable -Wno-unused-function -Wno-ma
CXX14=clang++ -arch x86_64 -ftemplate-depth-256
```

```r
sessionInfo()
```

```
R version 4.3.2 (2023-10-31)
Platform: x86_64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.4.1

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/lib/libRlapack.dylib;
```

```
locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: America/New_York
tzcode source: internal

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] colormap_0.1.4     rstan_2.32.6       StanHeaders_2.32.7

loaded via a namespace (and not attached):
 [1] gtable_0.3.4       jsonlite_1.8.8     compiler_4.3.2     Rcpp_1.0.11
 [5] stringr_1.5.1      parallel_4.3.2     gridExtra_2.3      scales_1.3.0
 [9] yaml_2.3.8         fastmap_1.1.1      ggplot2_3.4.4      R6_2.5.1
[13] curl_5.2.0         knitr_1.45         tibble_3.2.1       munsell_0.5.0
[17] pillar_1.9.0       rlang_1.1.2        utf8_1.2.4         V8_4.4.1
[21] stringi_1.8.3      inline_0.3.19      xfun_0.41          RcppParallel_5.1.7
[25] cli_3.6.2          magrittr_2.0.3     digest_0.6.33      grid_4.3.2
[29] lifecycle_1.0.4    vctrs_0.6.5        evaluate_0.23      glue_1.6.2
[33] QuickJSR_1.0.8     codetools_0.2-19   stats4_4.3.2       pkgbuild_1.4.3
[37] fansi_1.0.6        colorspace_2.1-0   rmarkdown_2.25     matrixStats_1.2.0
[41] tools_4.3.2        loo_2.6.0          pkgconfig_2.0.3    htmltools_0.5.7
```

Stan
Program 1 `model1.stan`

```
data {
  // Observed data
  int<lower=1> N;                       // Number of unique visits
  array[N] int<lower=0, upper=1> y; // Conversion indicator
}

parameters {
  real<lower=0, upper=1> theta; // Conversion probability
}

model {
  // Prior model
  theta ~ beta(0.5, 5.0); // 0 <~ theta <~ 0.5

  // Observational model
  for (n in 1:N) {
    target += bernoulli_lpmf(y[n] | theta);
  }
}

generated quantities {
  // Posterior predictions
  array[N] real y_pred;
  real p_hat_pred;

  for (n in 1:N) {
    y_pred[n] = bernoulli_rng(theta);
  }
  p_hat_pred = mean(y_pred);
}
```

Stan
Program 2 `model2.stan`

```stan
functions {
  // Saturating conversion probability function
  real theta(real x, real psi1, real psi2) {
    return psi1 * (-expm1(-psi2 * x));
  }
}

data {
  // Observed data
  int<lower=1> N;                      // Number of unique visits
  array[N] int<lower=0, upper=1> y;    // Conversion indicator
  array[N] real x;                     // Previous purchases (USD)
}

parameters {
  real<lower=0, upper=1> psi1; // Maximum conversion probability
  real<lower=0> psi2;          // Rate of saturation (1 / kUSD)
}

model {
  // Prior model
  psi1 ~ beta(0.5, 5.0);        // 0 <~ psi1 <~ 0.5
  psi2 ~ normal(0, 2.0 / 2.57); // 0 <~ psi2 <~ 2

  // Observational model
  for (n in 1:N) {
    target += bernoulli_lpmf(y[n] | theta(x[n], psi1, 1e-3 * psi2));
  }
}

generated quantities {
  // Posterior predictions
  array[N] real y_pred;
  real p_hat_pred;

  for (n in 1:N) {
    y_pred[n] = bernoulli_rng(theta(x[n], psi1, 1e-3 * psi2));
  }
  p_hat_pred = mean(y_pred);
}
```

Stan
Program 3 `model3.stan`

```
functions {
  // Saturating conversion probability function
  real theta(real x, real psi1, real psi2) {
    return psi1 * (-expm1(-psi2 * x));
  }
}

data {
  // Observed data
  int<lower=1> N;                       // Number of unique visits
  array[N] int<lower=0, upper=1> y;  // Conversion indicator
  array[N] real x;                      // Previous purchases (USD)
}

parameters {
  real<lower=0, upper=1> psi1;      // Maximum conversion probability
  real<lower=0> psi2;               // Rate of saturation (1 / kUSD)
  real<lower=0, upper=1> lambda;    // Proportion of VIP visitors
  real<lower=0, upper=1> theta_VIP; // VIP conversion probability
}

model {
  // Prior model
  psi1 ~ beta(0.5, 5.0);         // 0   <~    psi1   <~ 0.5
  psi2 ~ normal(0, 2.0 / 2.57); // 0   <~    psi2   <~ 2
  lambda ~ beta(1, 1);           // Uniform probability density function
  theta_VIP ~ beta(5.0, 0.5);    // 0.5 <~ theta_VIP <~ 1

  // Observational model
  for (n in 1:N) {
    target += log_mix(lambda,
                      bernoulli_lpmf(y[n] | theta_VIP),
                      bernoulli_lpmf(y[n] | theta(x[n], psi1, 1e-3 * psi2)));
  }
}

generated quantities {
  // Posterior predictions
  array[N] real y_pred;
  real p_hat_pred;

  for (n in 1:N) {
    if (bernoulli_rng(lambda)) {
      y_pred[n] = bernoulli_rng(theta_VIP);
    } else {
      y_pred[n] = bernoulli_rng(theta(x[n], psi1, 1e-3 * psi2));
    }
  }
  p_hat_pred = mean(y_pred);
}
```

Stan
Program 4 `model4.stan`

```
functions {
  // Saturating conversion probability function
  real theta(real x, real psi1, real psi2) {
    return psi1 * (-expm1(-psi2 * x));
  }
}

data {
  // Observed data
  int<lower=1> N;                       // Number of unique visits
  array[N] int<lower=0, upper=1> y;     // Conversion indicator
  array[N] real x;                      // Previous purchases (USD)

  int<lower=1> N_aux;                           // Number of VIP-only visits
  array[N_aux] int<lower=0, upper=1> y_aux; // Conversion indicator
}

parameters {
  real<lower=0, upper=1> psi1;      // Maximum conversion probability
  real<lower=0> psi2;               // Rate of saturation (1 / kUSD)
  real<lower=0, upper=1> lambda;    // Proportion of VIP visitors
  real<lower=0, upper=1> theta_VIP; // VIP conversion probability
}

model {
  // Prior model
  psi1 ~ beta(0.5, 5.0);         // 0   <~    psi1    <~ 0.5
  psi2 ~ normal(0, 2.0 / 2.57); // 0   <~    psi2    <~ 2
  lambda ~ beta(1, 1);           // Uniform probability density function
  theta_VIP ~ beta(5.0, 0.5);    // 0.5 <~ theta_VIP <~ 1

  // Observational model
  for (n in 1:N) {
    target += log_mix(lambda,
                      bernoulli_lpmf(y[n] | theta_VIP),
                      bernoulli_lpmf(y[n] | theta(x[n], psi1, 1e-3 * psi2)));
  }

  for (n in 1:N_aux) {
    target += bernoulli_lpmf(y_aux[n] | theta_VIP);
  }
}
```

```
generated quantities {
  // Posterior predictions
  array[N] real y_pred;
  real p_hat_pred;

  array[N_aux] real y_aux_pred;
  real p_hat_aux_pred;
```

Stan
Program 5 `model5.stan`

```
functions {
  // Saturating conversion probability function
  real theta(real x, real psi1, real psi2) {
    return psi1 * (-expm1(-psi2 * x));
  }
}

data {
  // Observed data
  int<lower=1> N;                      // Number of unique visits
  array[N] int<lower=0, upper=1> y;  // Conversion indicator
  array[N] real x;                     // Previous purchases (USD)

  int<lower=1> N_aux;                         // Number of VIP-only visits
  array[N_aux] int<lower=0, upper=1> y_aux; // Conversion indicator

  // Hypothetical data
  int<lower=1> N_hyp;
  array[N_hyp] real x_hyp;
}

parameters {
  real<lower=0, upper=1> psi1;      // Maximum conversion probability
  real<lower=0> psi2;                 // Rate of saturation (1 / kUSD)
  real<lower=0, upper=1> lambda;    // Proportion of VIP visitors
  real<lower=0, upper=1> theta_VIP; // VIP conversion probability
}

model {
  // Prior model
  psi1 ~ beta(0.5, 5.0);         // 0   <~    psi1   <~ 0.5
  psi2 ~ normal(0, 2.0 / 2.57); // 0   <~    psi2   <~ 2
  lambda ~ beta(1, 1);           // Uniform probability density function
  theta_VIP ~ beta(5.0, 0.5);    // 0.5 <~ theta_VIP <~ 1

  // Observational model
  for (n in 1:N) {
    target += log_mix(lambda,
                      bernoulli_lpmf(y[n] | theta_VIP),
                      bernoulli_lpmf(y[n] | theta(x[n], psi1, 1e-3 * psi2)));
  }

  for (n in 1:N_aux) {
    target += bernoulli_lpmf(y_aux[n] | theta_VIP);
  }
}

generated quantities {
  // Posterior predictions
  array[N] real y_pred;
```