A Hands-on Introduction to BART, BCF, and Other Bayesian Tree Ensembles for Applied Causal Inference

Jared Murray and Drew Herren, January 2025

https://bit.ly/ichps_trees

Our Agenda

Introduce new software for fitting Bayesian tree models (stochtree)

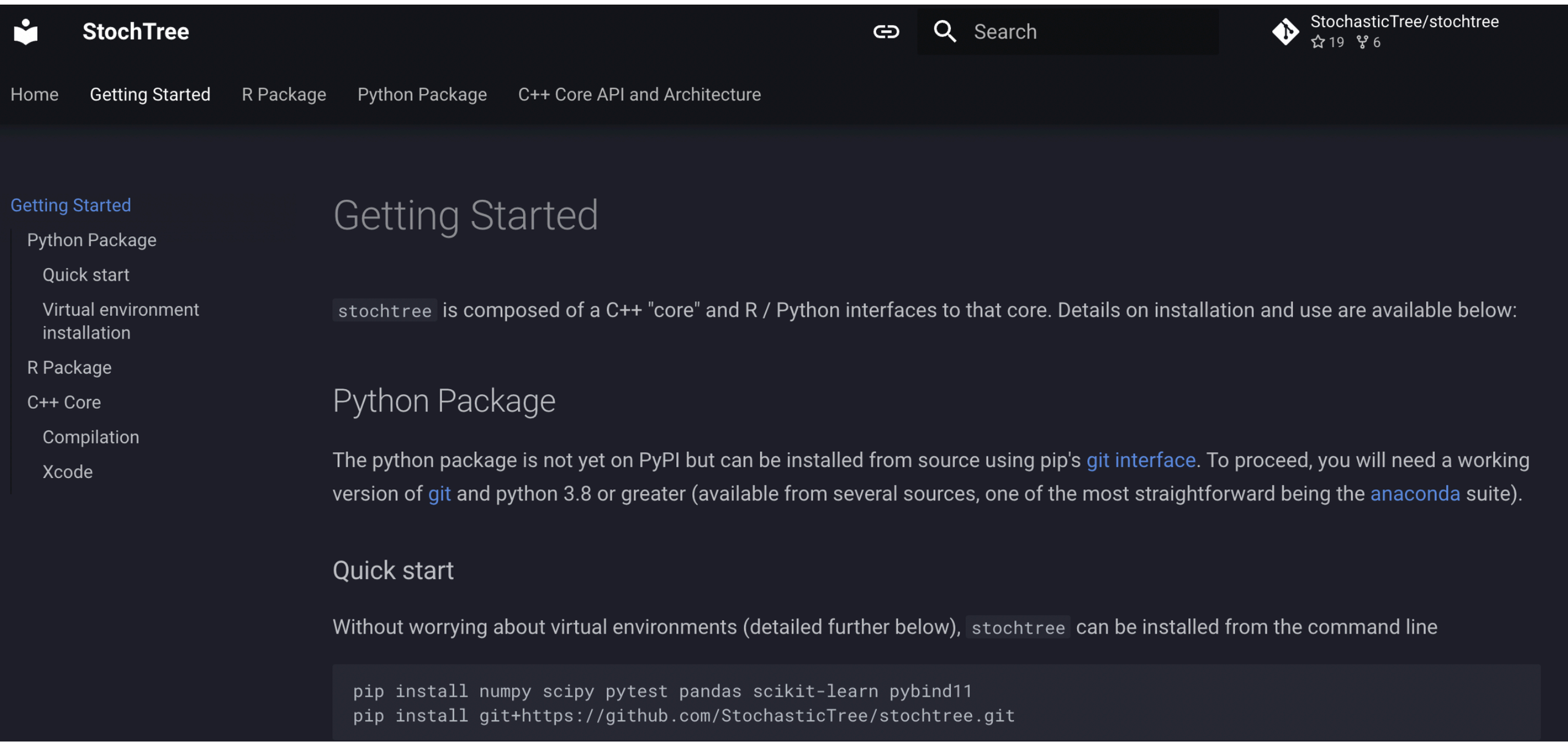
Review (Bayesian) trees

Specialize these to causal inference

Demo!

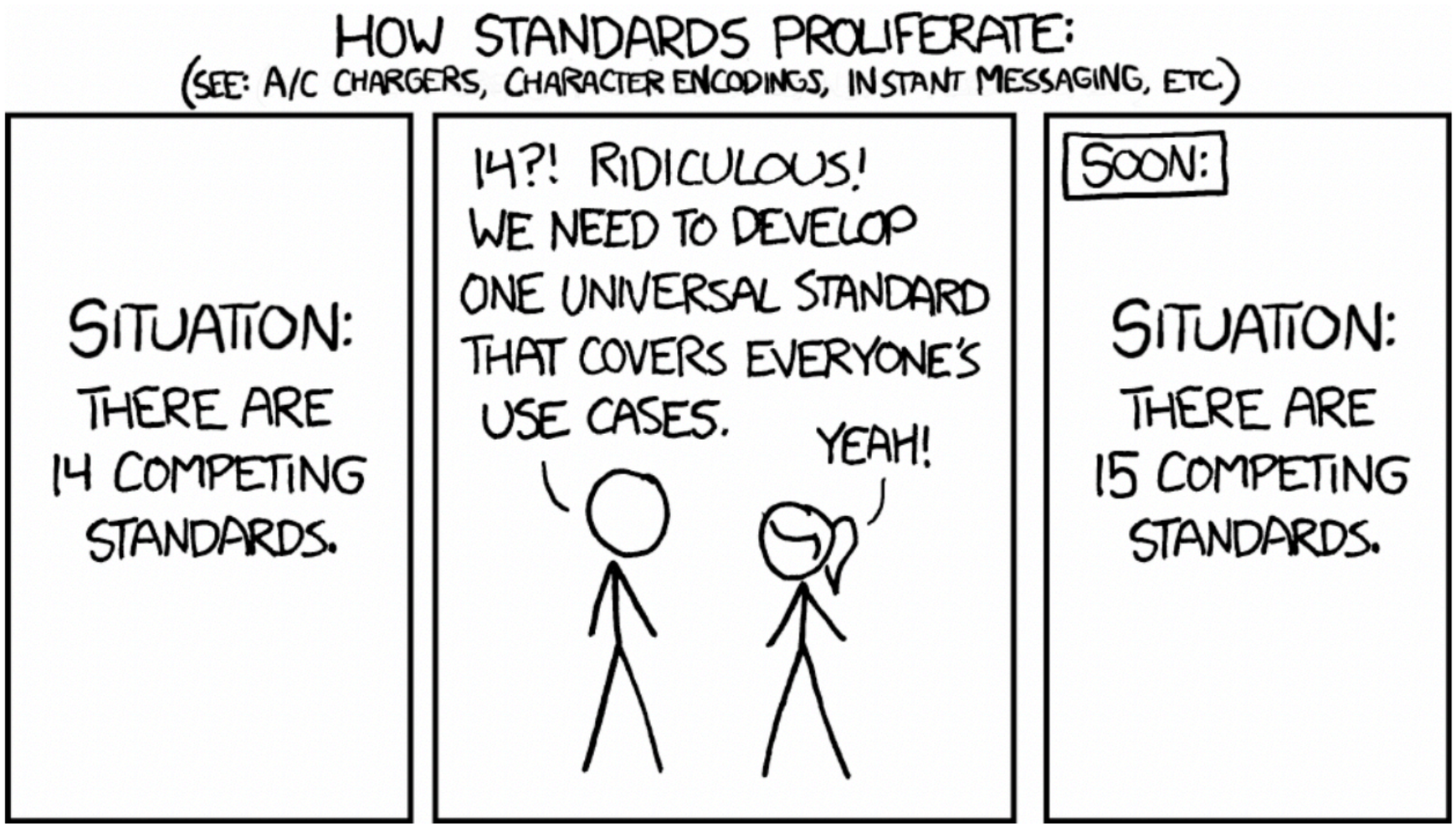
Stochtree

https://stochtree.ai



Current state of Bayesian tree software

1	Just want to fit a simple model?	Use one of (too) many R packages (BART, dbarts, flexBART, bcf, …)
2	Want to use new method from recent paper (i.e. warm-start BART)?	*[If lucky]* follow instructions given on Github and modify example scripts
3	Want to prototype and evaluate a new research idea?	Fork an existing repo, learn C++, write anywhere between 10 and 1,000 new lines of code, cry at least once

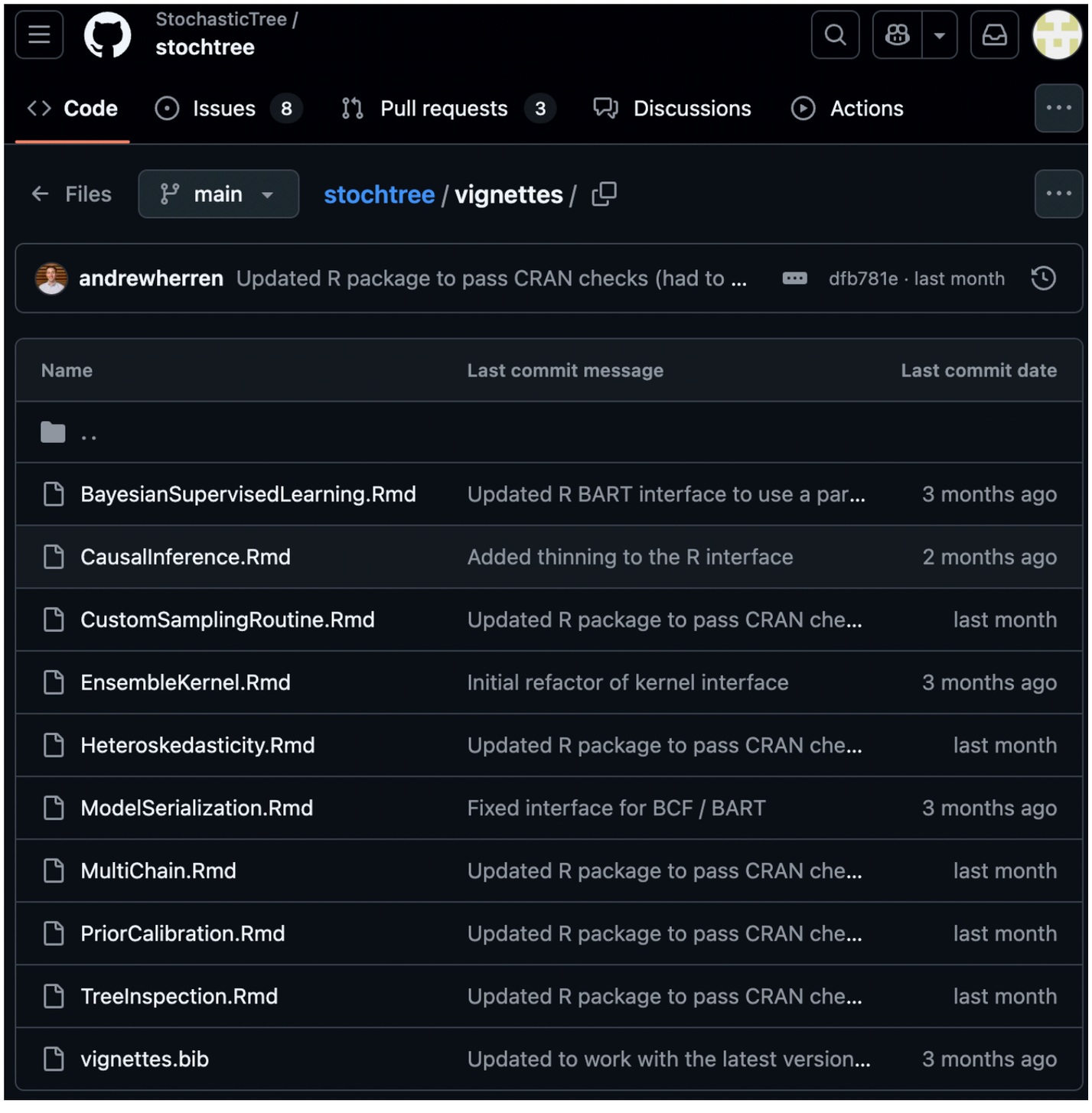


For use case #1, most packages are fine!

So … why build stochtree?

1. **Get various methods out of the "debug Github README" phase**: this includes things like smart initialization of MCMC, heteroskedastic models, semiparametric models, and more
2. **Python**!!
3. **Get methods out of the "debug Github README" phase**: this includes things like smart initialization of MCMC, heteroskedastic models, semiparametric models, and more
4. **Expose lower-level functionality at the R / Python level**: make it easier to "prototype" new models and updates to trees

Vignettes for advanced features



Core functionality

Embedding BART in custom models

Heteroskedastic errors

Combining multiple MCMC chains

Docs for Complete APIs

https://stochtree.ai/

StochTree R API Reference

Overview of the `stochtree` R library's key classes and functions, built as a self-contained doc site using the pkgdown format. The `stochtree` interface is divided into two "levels":

1. "High level": end-to-end implementations of stochastic tree ensembles for supervised learning (BART / XBART) and causal inference (BCF / XBCF).
    a. The BART (supervised learning) interface is documented here.
    b. The BCF (causal inference) interface is documented here.
2. "Low level": we provide access to most of the C++ sampling objects and functionality via R, which allow for custom sampling algorithms and integration of other model terms. This interface consists broadly of the following components:
    a. Data API: loading and storing in-memory data needed to train `stochtree` models.
    b. Forest API: creating, storing, modifying, and sampling ensembles of decision trees that underlie all `stochtree` models.
    c. Serialization API: serializing models to JSON (files or in-memory strings).
    d. Random Effects API: sampling from additive random effects models.

StochTree Python API Reference

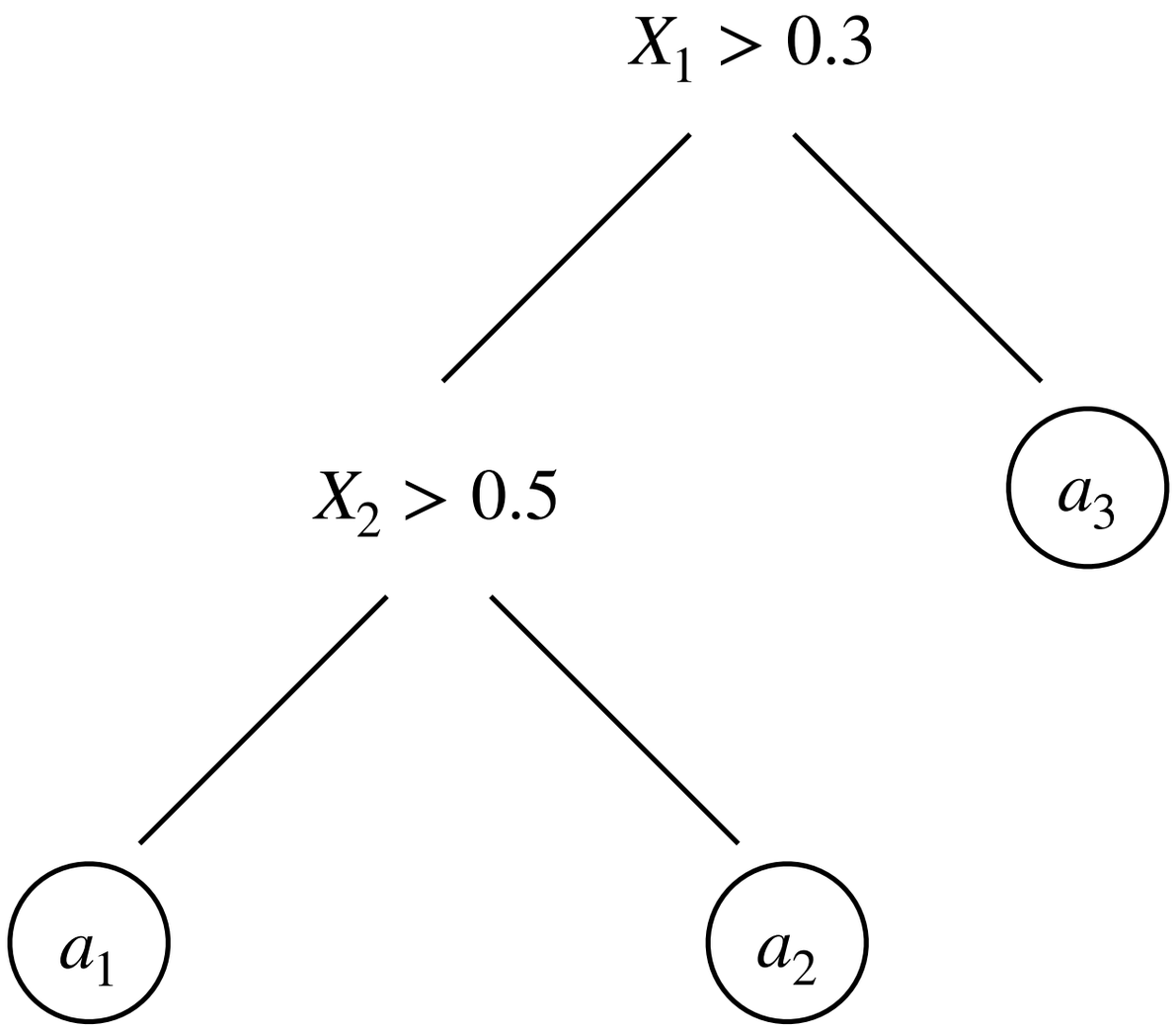
Overview of the `stochtree` python library's key classes and functions.

The `stochtree` interface is divided into two "levels":

1. "High level": end-to-end implementations of stochastic tree ensembles for supervised learning (BART / XBART) and causal inference (BCF / XBCF). Both interfaces are designed to mirror the scikit-learn estimator style, with the `.fit()` method replaced by a `.sample()` method.
    a. The BART (supervised learning) interface is documented here.
    b. The BCF (causal inference) interface is documented here.
2. "Low level": we provide access to most of the C++ sampling objects and functionality via Python, which allow for custom sampling algorithms and integration of other model terms. This interface is documented here and consists broadly of the following components:
    a. Data API: loading and storing in-memory data needed to train `stochtree` models.
    b. Forest API: creating, storing, and modifying ensembles of decision trees that underlie all `stochtree` models.
    c. Sampler API: sampling from stochastic tree ensemble models as well as several supported parametric models.
    d. Utilities API: seeding a C++ random number generator, preprocessing data, and serializing models to JSON (files or in-memory strings).

A (Brief!) Review of Tree Methods

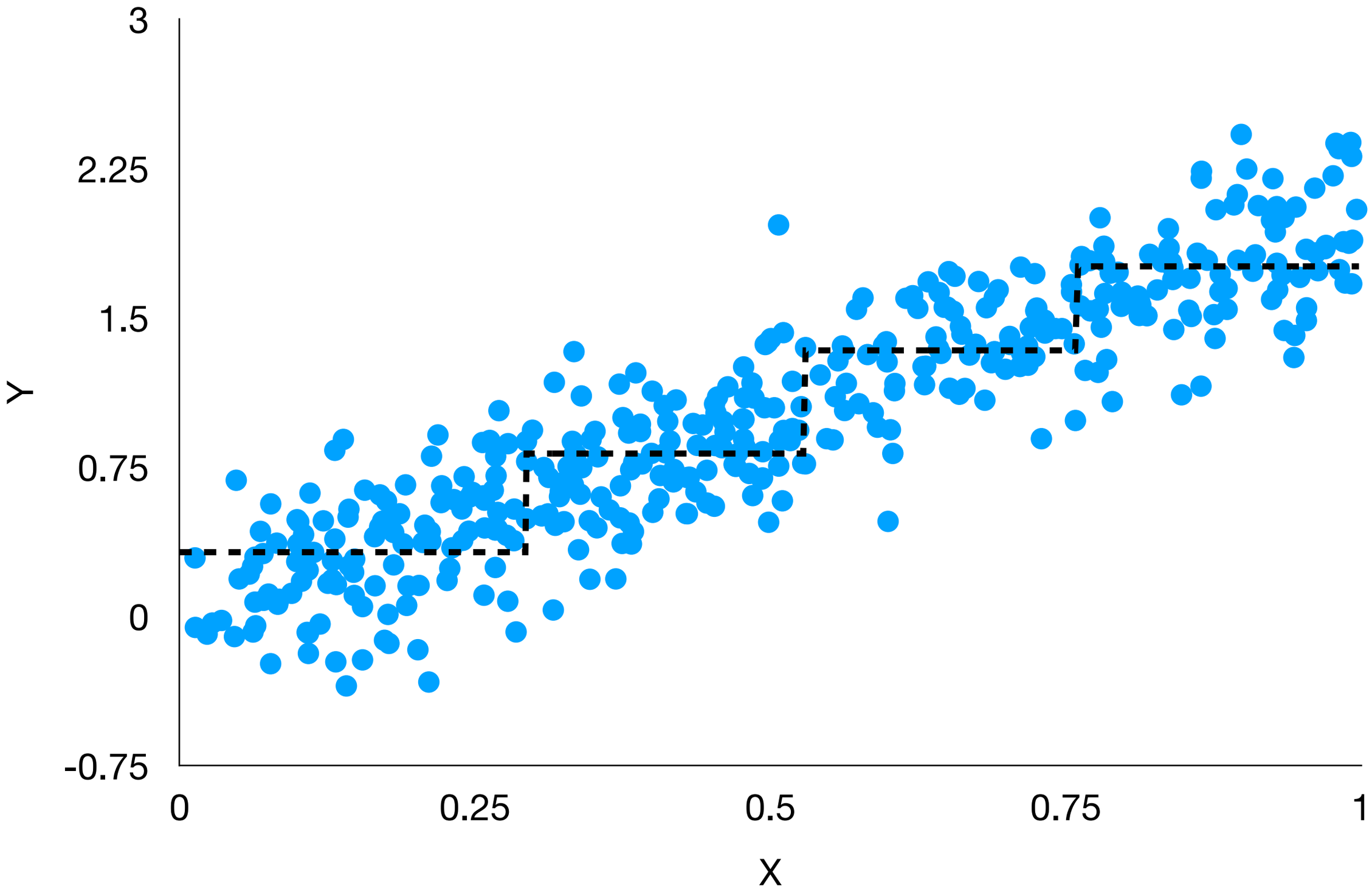
Trees are a simple but powerful machine learning tool …



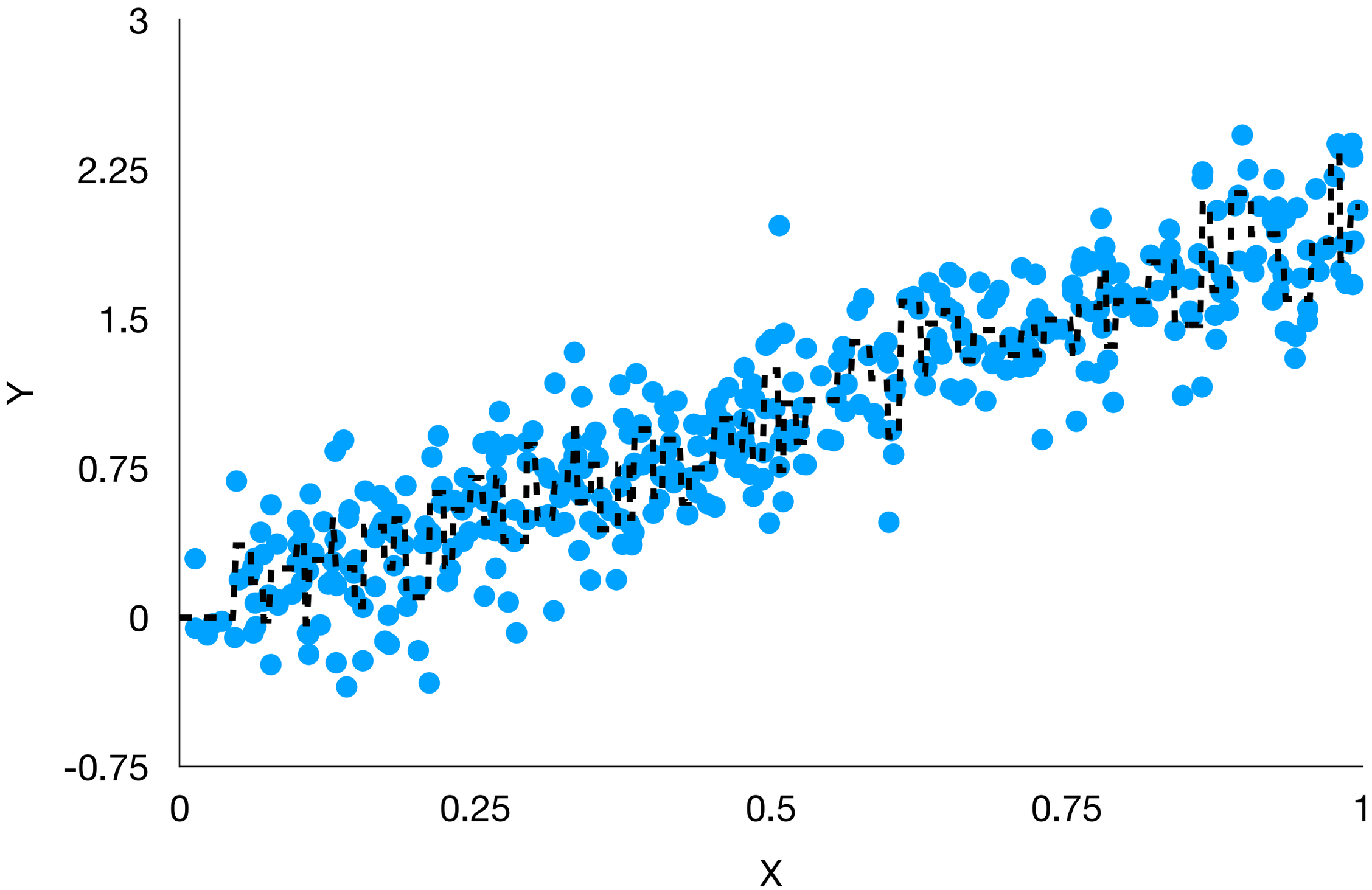
- Algorithm partitions the data - no need to specify model form like regression or neural networks
- Easy to understand model as a series of “if feature 1 < 0.5 then predict a_j” statements

… but they present a difficult overfitting tradeoff

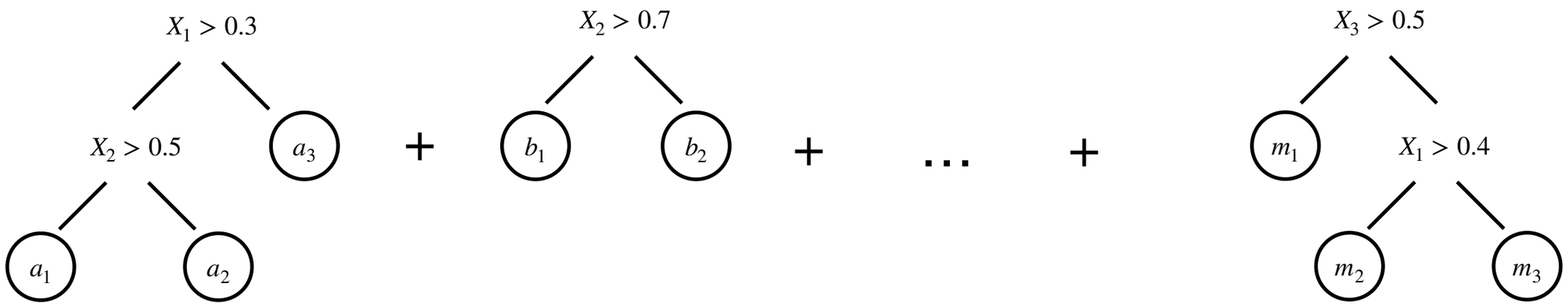
Simple decision trees don’t capture much complexity



Complex decision trees unlikely to generalize well!

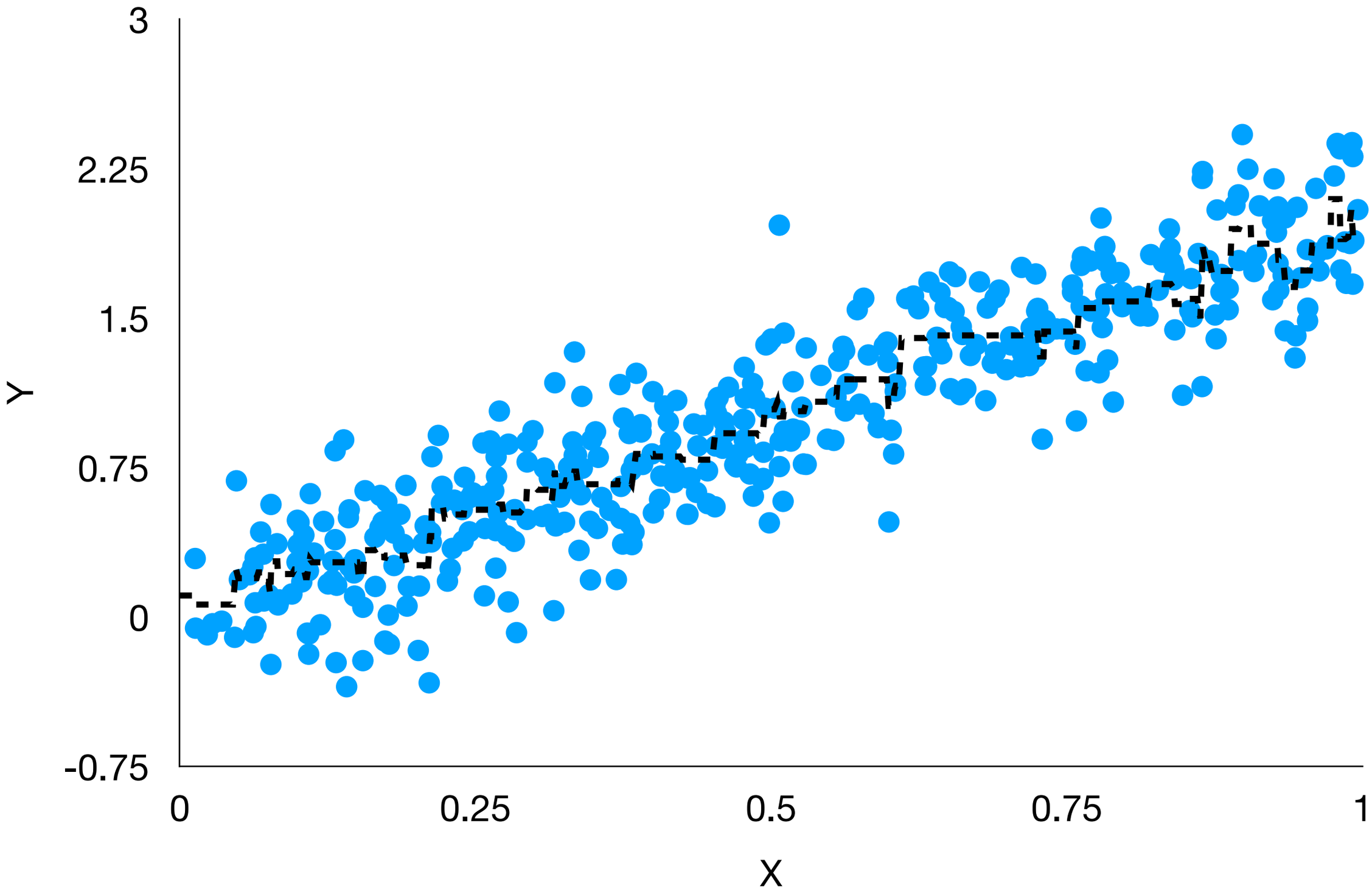


Solution: ensemble of simple decision trees



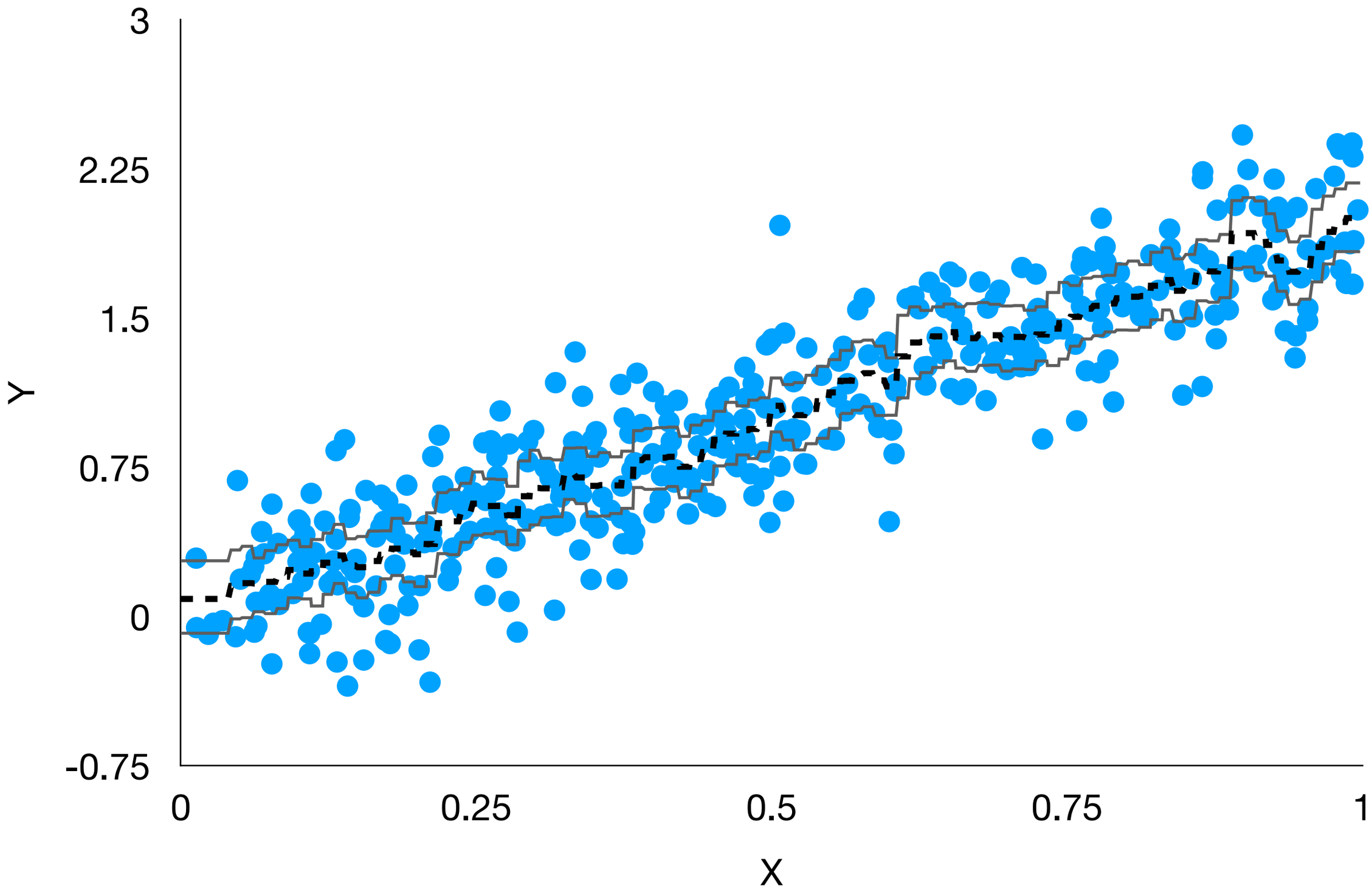
This includes common methods like random forest and gradient boosting, implemented in scikit-learn, xgboost, lightgbm, ranger

More flexibility, less “overfitting” concerns!



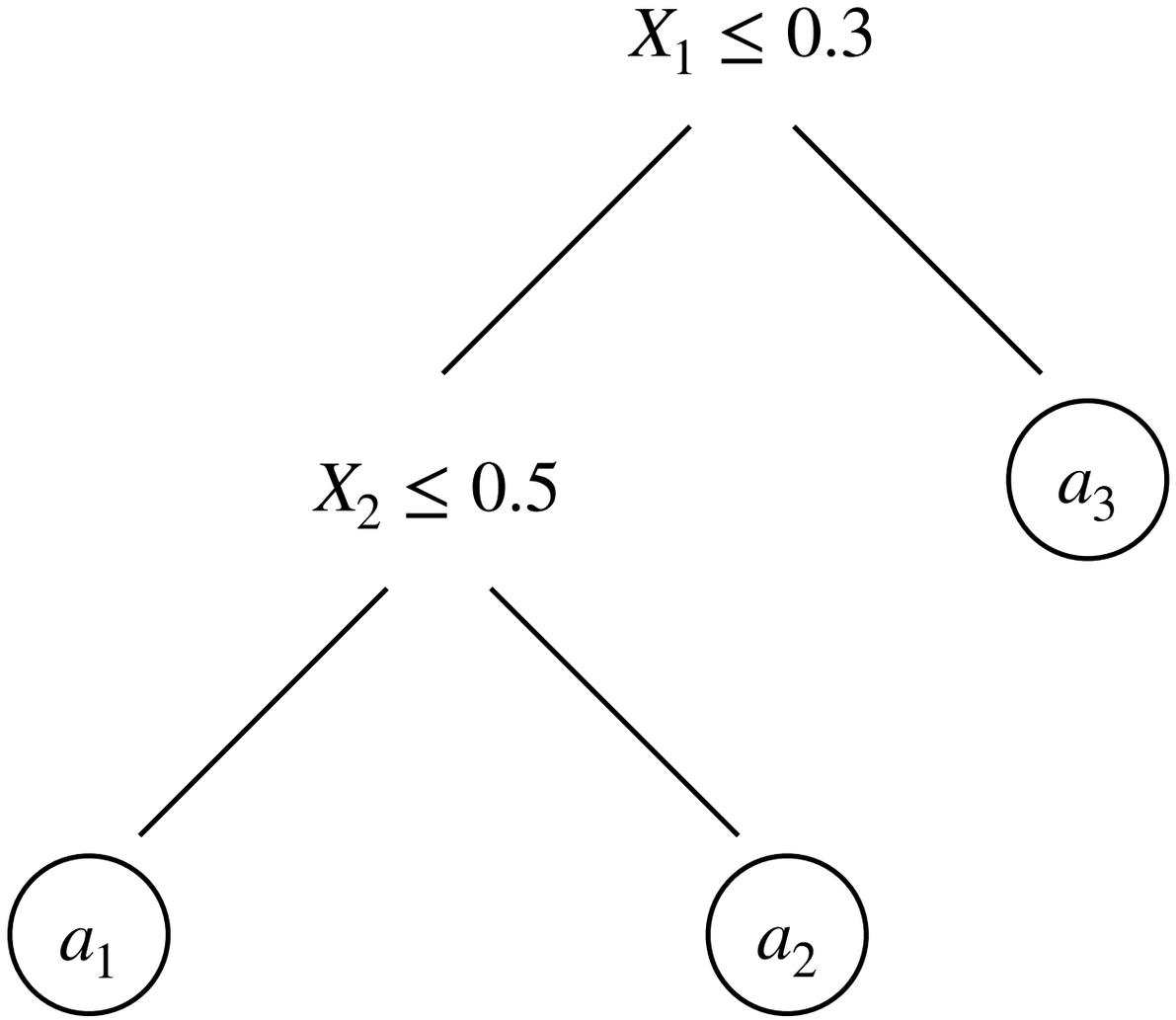
What if we want uncertainty intervals instead of just good predictions?

BART gives us that!



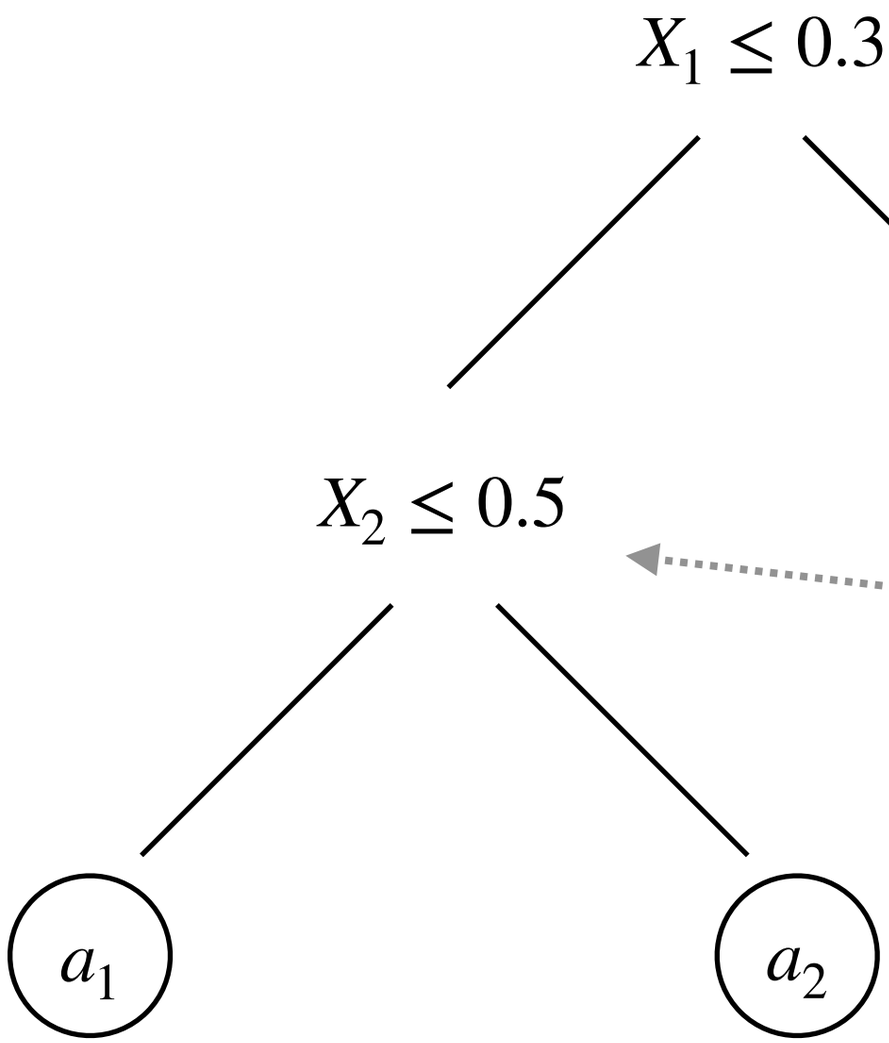
Let’s first get comfortable with trees

Tree is a conceptually simple data structure



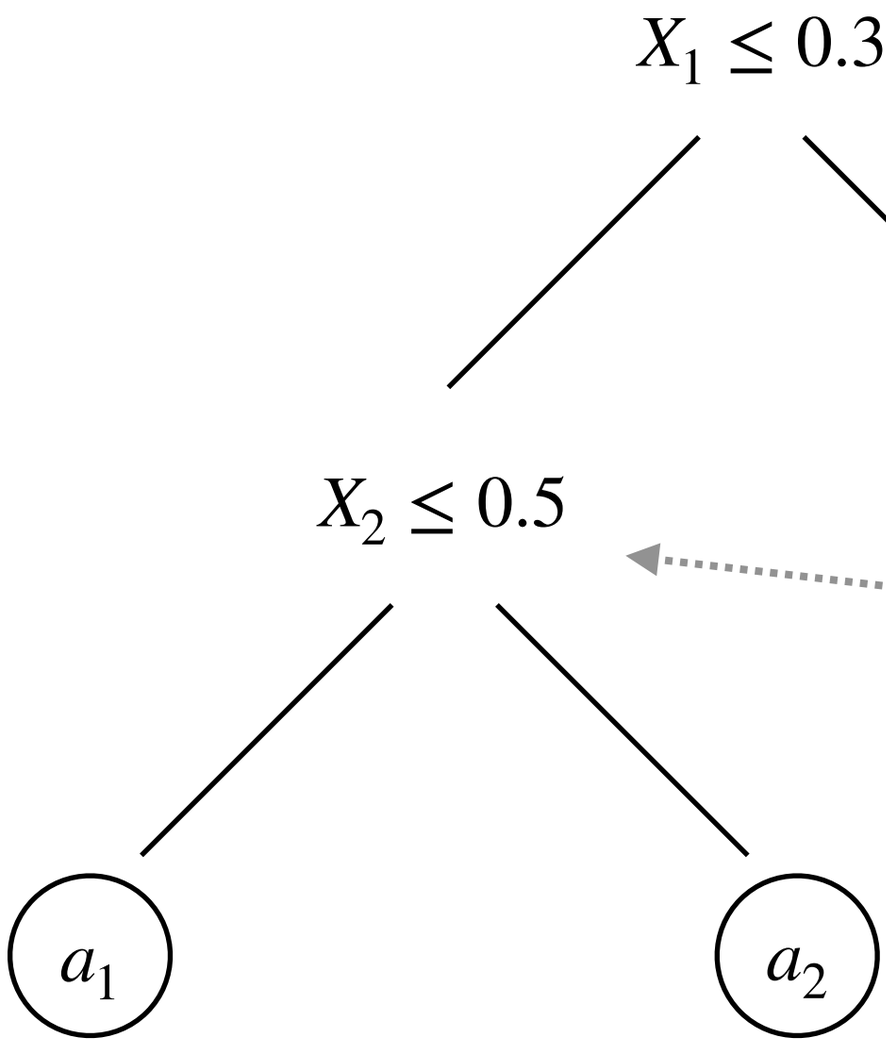
The entire structure pictured here is a **tree**

Tree is a conceptually simple data structure



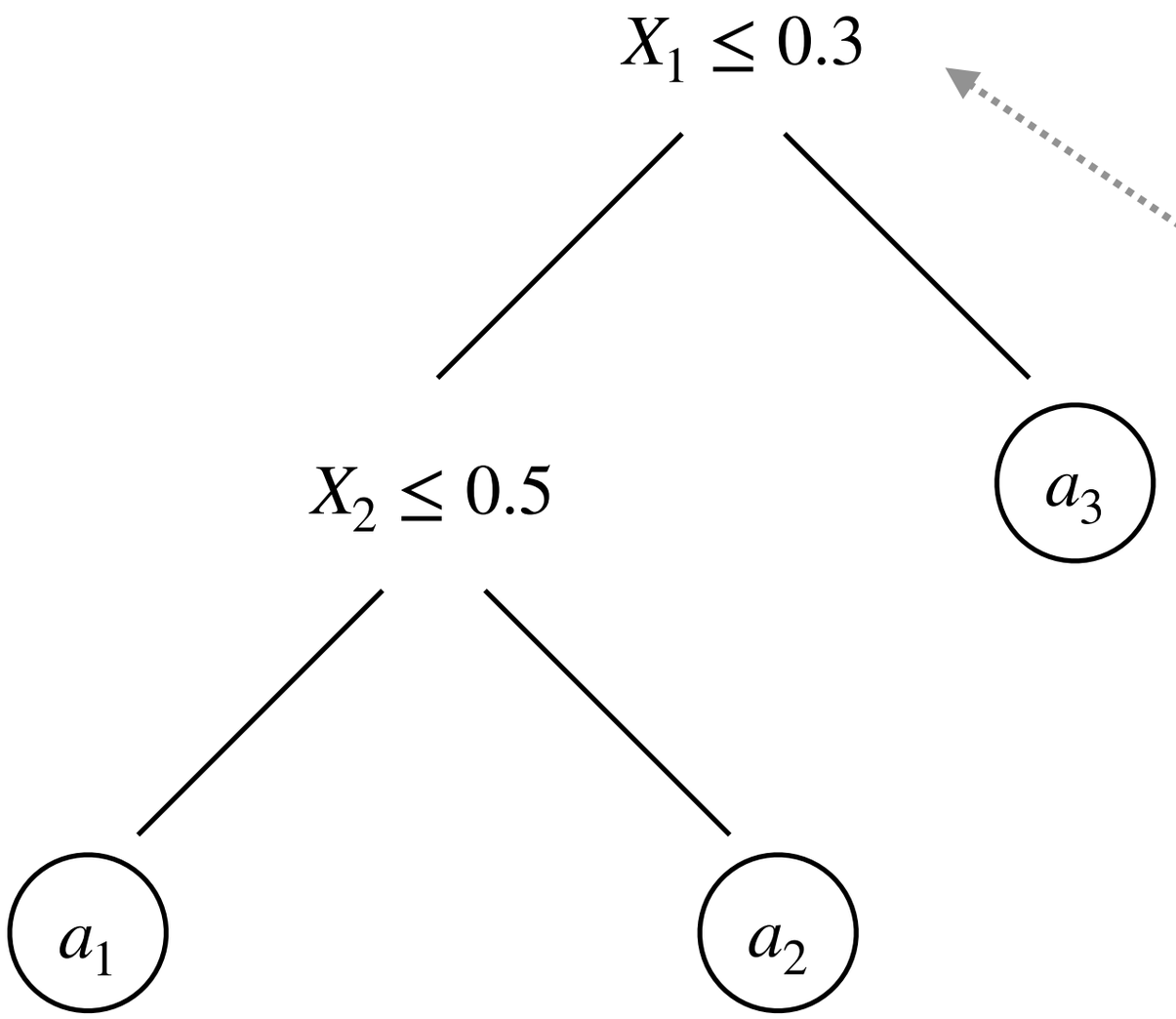
Individual elements of the tree are "**nodes**"

Tree is a conceptually simple data structure



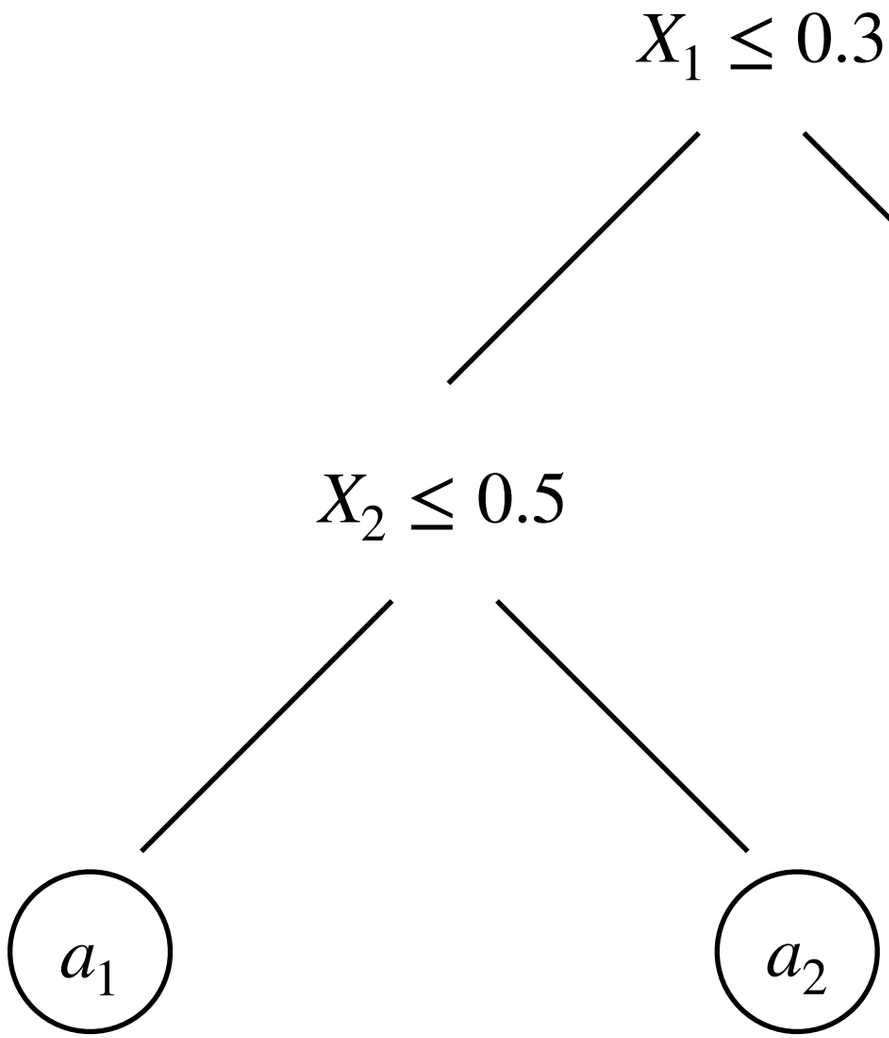
Elements defining split rules are “**split nodes**”

Tree is a conceptually simple data structure



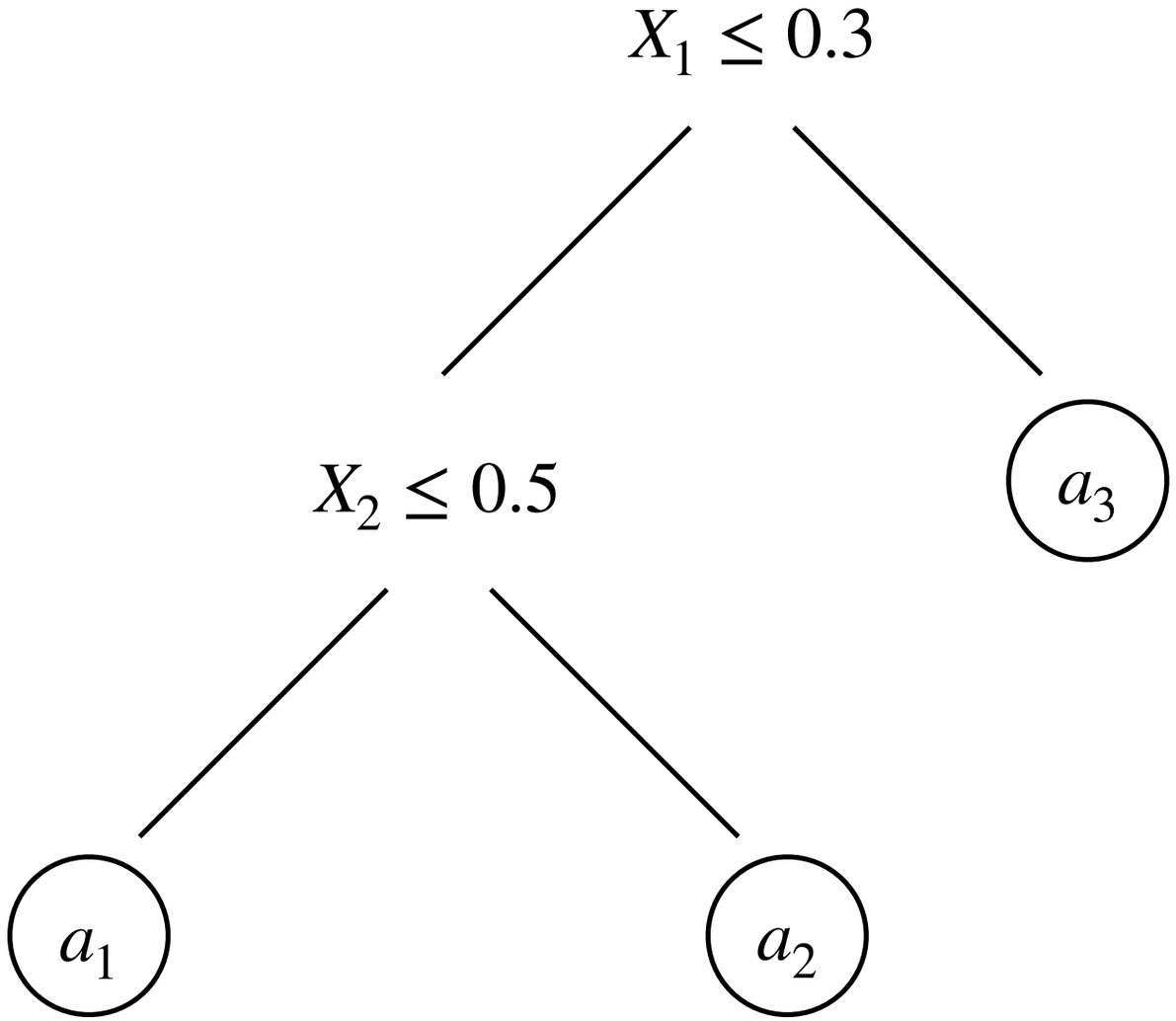
The first node is known as the "**root**" node

Tree is a conceptually simple data structure



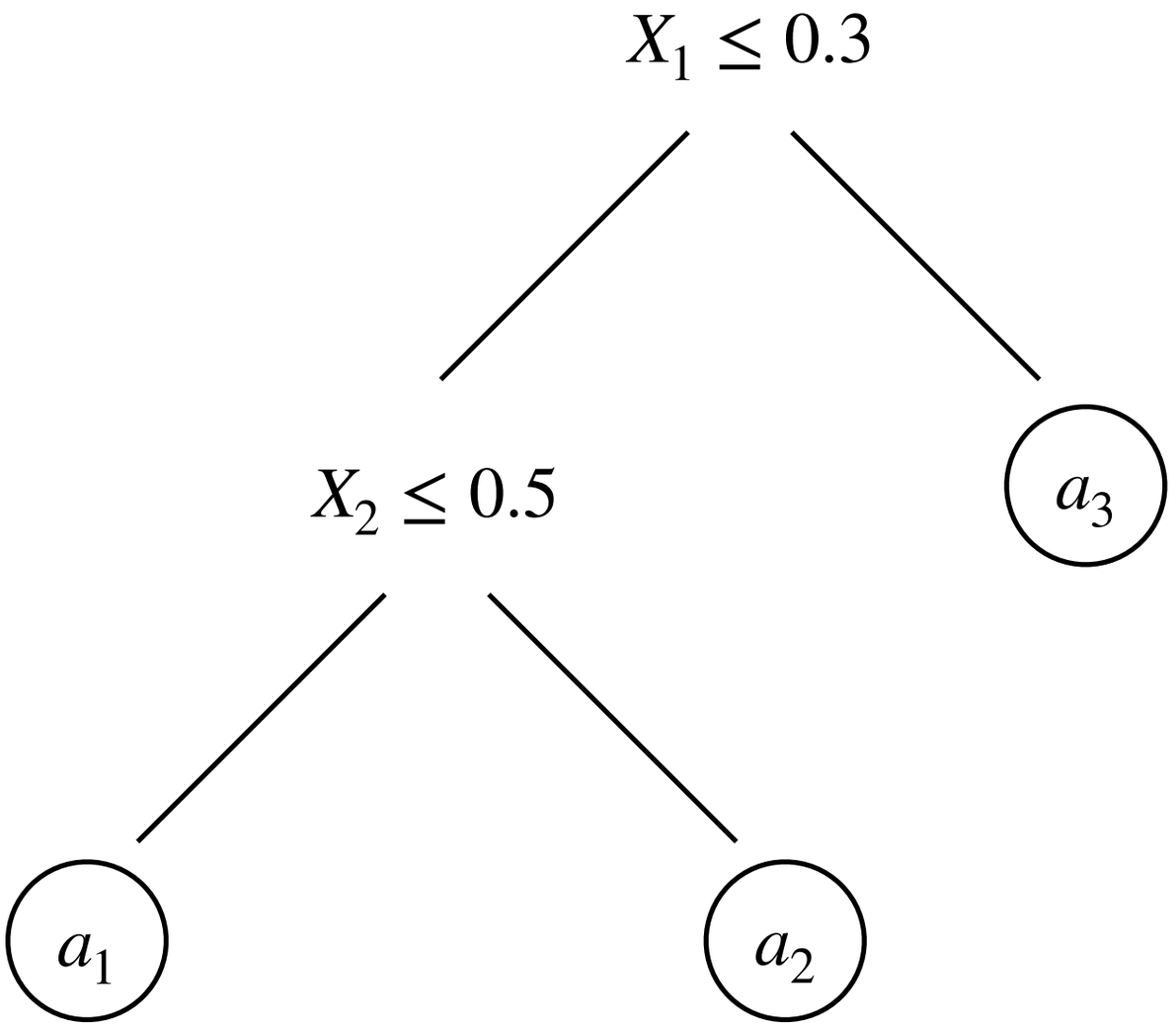
Elements with no further splits are "**leaves**"

Tree is a conceptually simple data structure

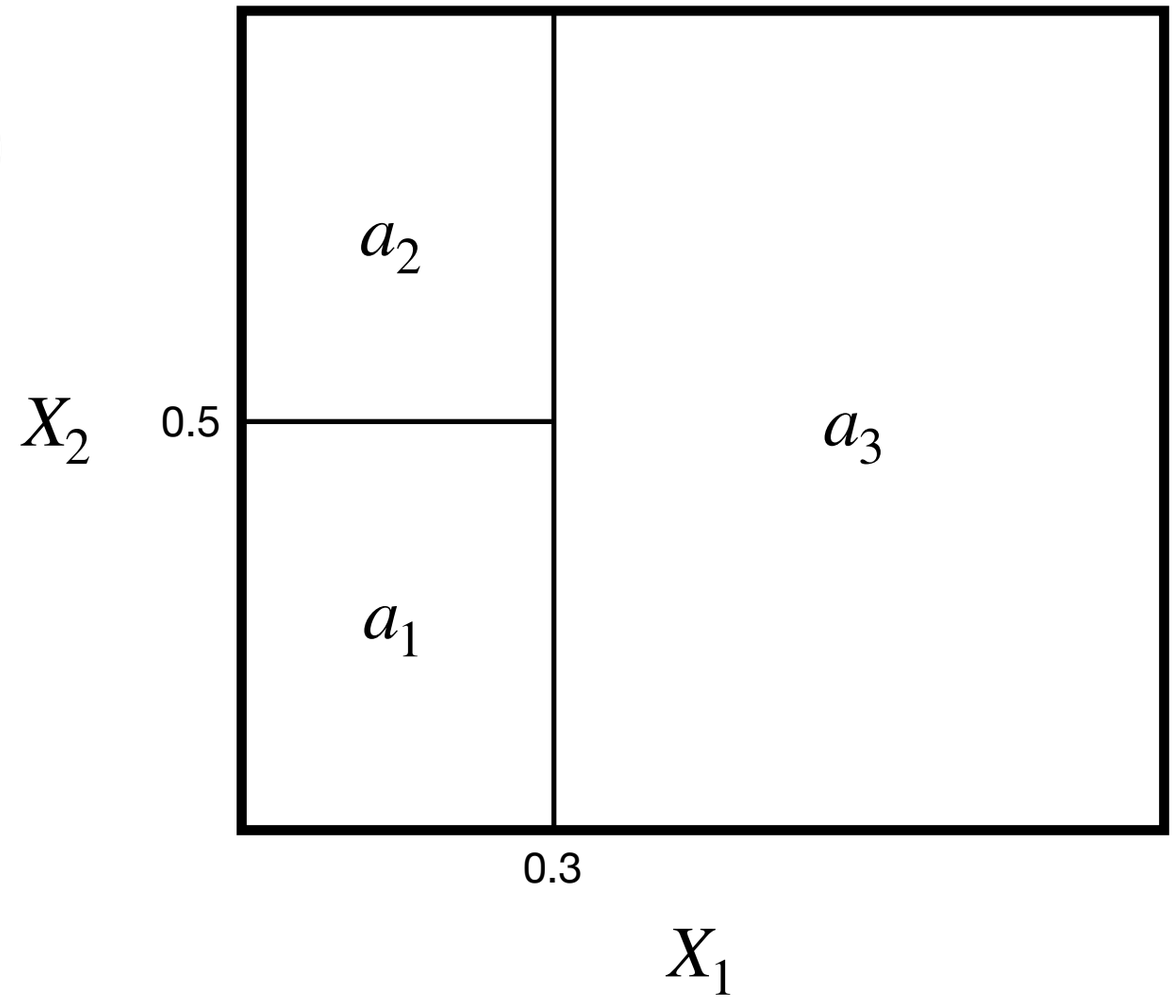


The structure is called a tree because each split creates new "**branches**"

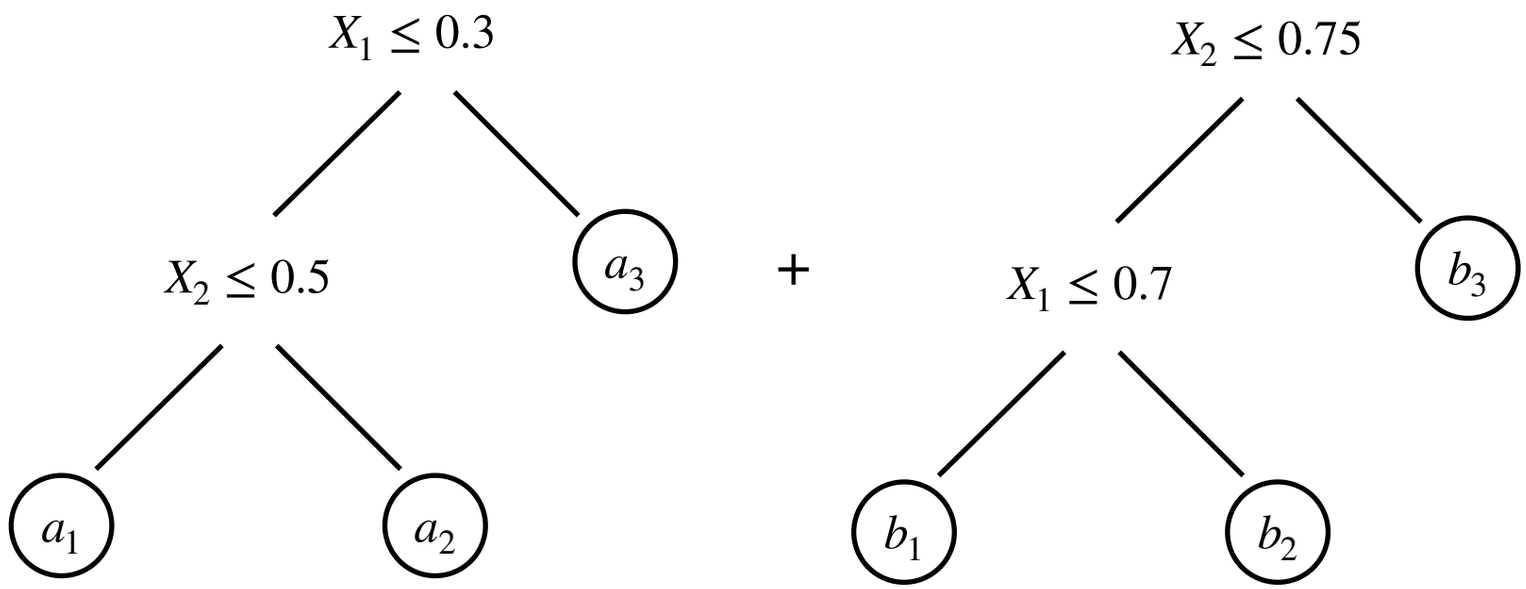
Can also think about it as a “partition” of a data set

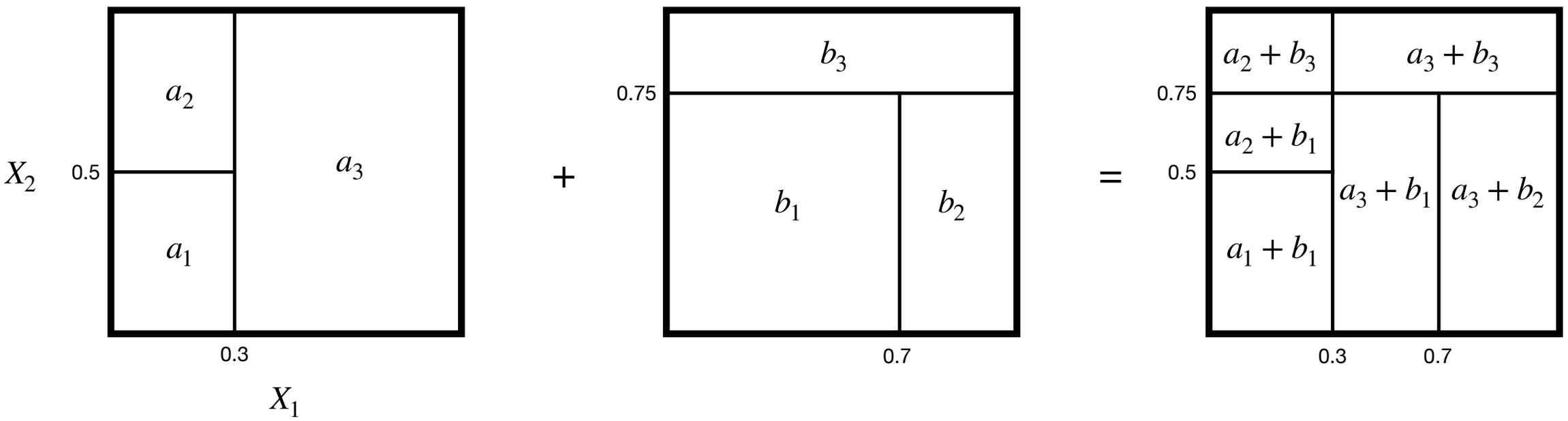


This tree divides or “partitions” the data into three separate regions with different predictions



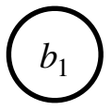
A sum of trees is a more refined partition





Classical decision tree training: recursive partitioning

Recursive partitioning offers fast blueprint for growing trees

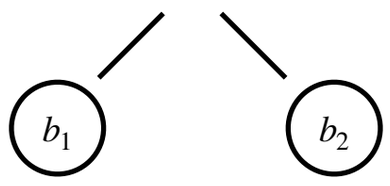


Start with a root node and check if stopping condition is met
(minimum leaf sample size, max leaf depth, etc…)

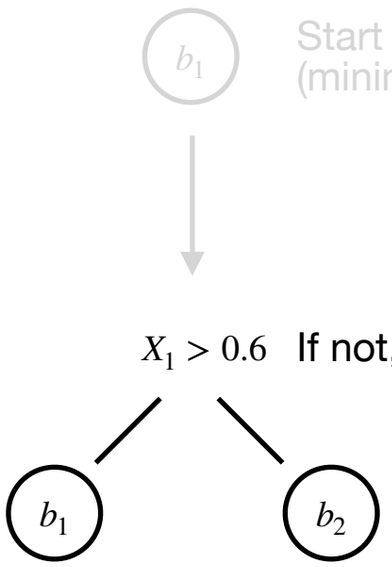
Recursive partitioning offers fast blueprint for growing trees

Start with a root node and check if stopping condition is met
(minimum leaf sample size, max leaf depth, etc…)

$X_1 > 0.6$ If not, split root at the variable & cutpoint that maximizes split criterion



Recursive partitioning offers fast blueprint for growing trees

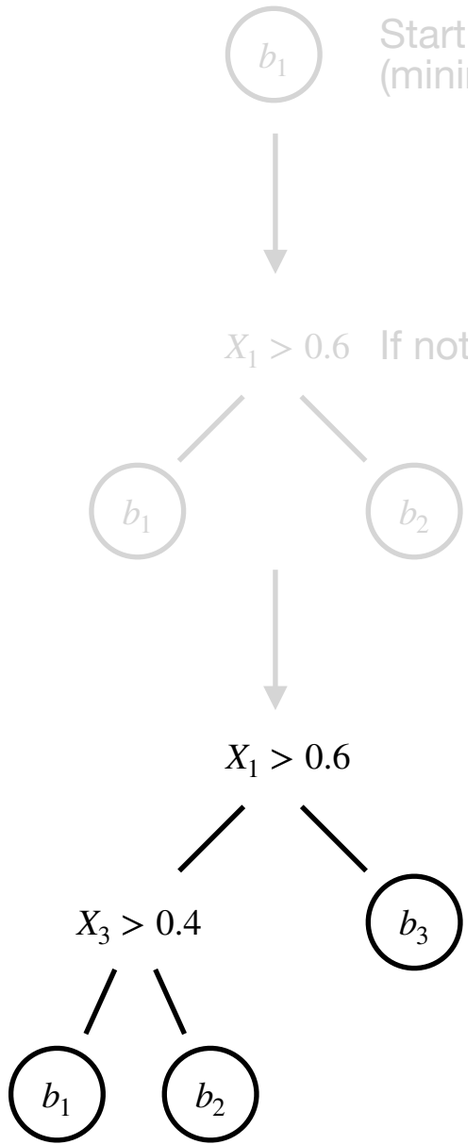


Start with a root node and check if stopping condition is met (minimum leaf sample size, max leaf depth, etc…)

If not, split root at the variable & cutpoint that maximizes split criterion

Check if stopping condition is met for new left and right nodes

Recursive partitioning offers fast blueprint for growing trees



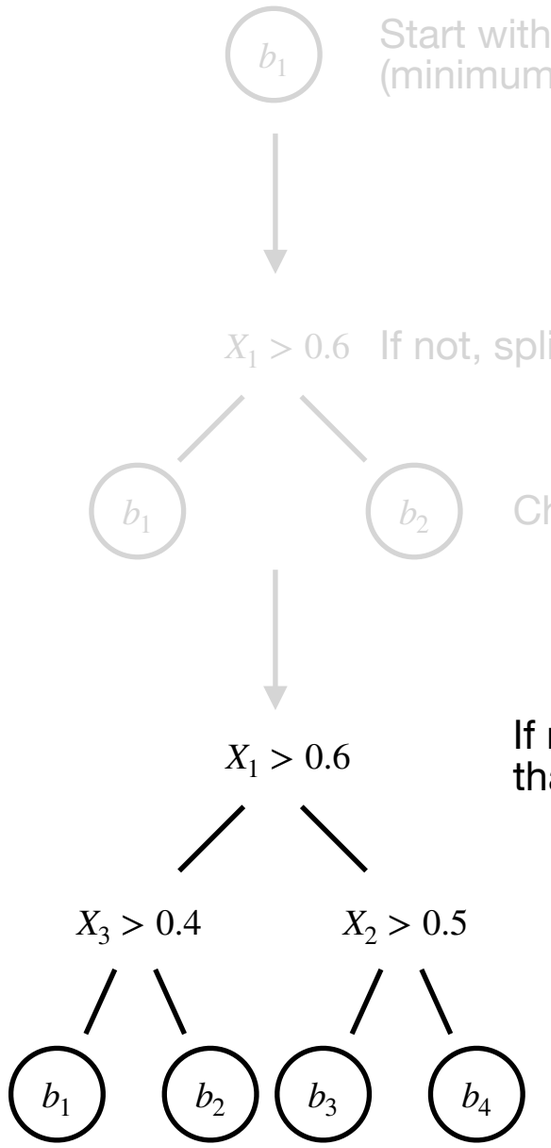
Start with a root node and check if stopping condition is met (minimum leaf sample size, max leaf depth, etc…)

If not, split root at the variable & cutpoint that maximizes split criterion

Check if stopping condition is met for new left and right nodes

If not met for left node, recursively split at the variable & cutpoint that maximizes split criterion

Recursive partitioning offers fast blueprint for growing trees



Start with a root node and check if stopping condition is met (minimum leaf sample size, max leaf depth, etc…)

If not, split root at the variable & cutpoint that maximizes split criterion

Check if stopping condition is met for new left and right nodes

If not met for right node, recursively split at the variable & cutpoint that maximizes split criterion

Recursive partitioning offers fast blueprint for growing trees

Start with a root node and check if stopping condition is met (minimum leaf sample size, max leaf depth, etc…)

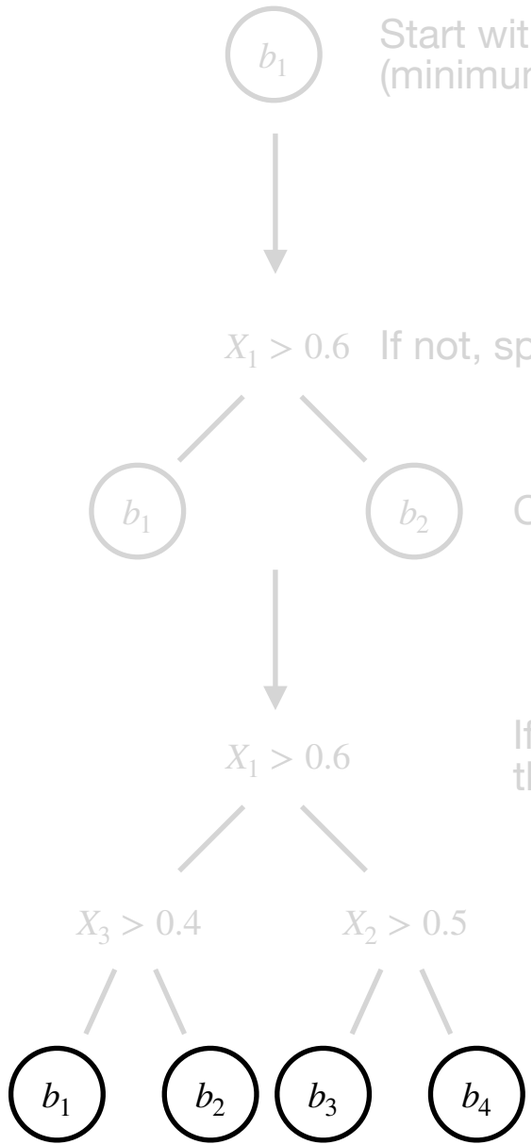
If not, split root at the variable & cutpoint that maximizes split criterion

Check if stopping condition is met for new left and right nodes

If not met for right node, recursively split at the variable & cutpoint that maximizes split criterion

Continue until stopping condition satisfied for all leaves

Recursive partitioning offers fast blueprint for growing trees



Start with a root node and check if stopping condition is met (minimum leaf sample size, max leaf depth, etc…)

If not, split root at the variable & cutpoint that maximizes split criterion

Check if stopping condition is met for new left and right nodes

If not met for right node, recursively split at the variable & cutpoint that maximizes split criterion

Continue until stopping condition satisfied for all leaves

Once the tree is done growing, set prediction values ($b_1, b_2, \ldots$) for each of the leaves

Tree boosting: recursive partitioning for ensemble of trees

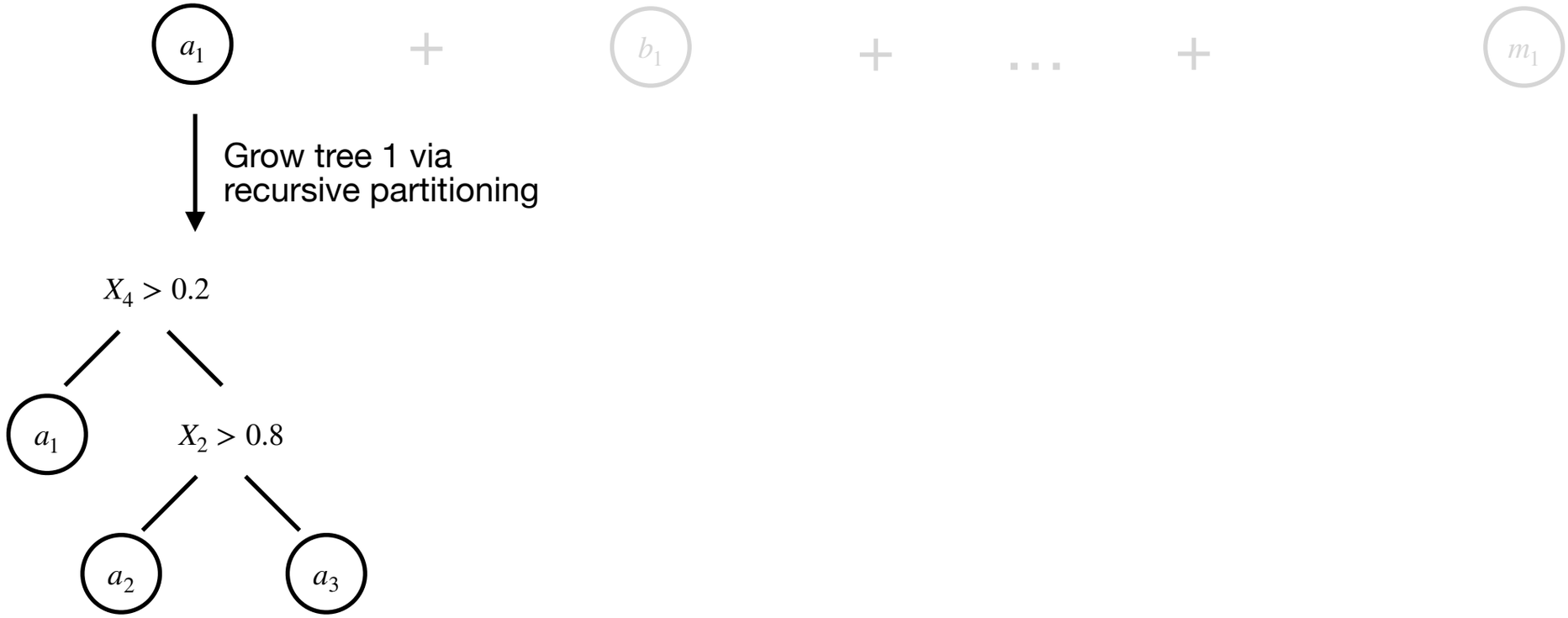
Boosting fits ensemble of trees using recursive partitioning

r_1
2.5
-4
3.1
-3.7
…
-1.4
2.7

Start with original labels as training data for first tree

Boosting fits ensemble of trees using recursive partitioning

r_1
2.5
-4
3.1
-3.7
...
-1.4
2.7



Boosting fits ensemble of trees using recursive partitioning

r_1
2.5
-4
3.1
-3.7
…
-1.4
2.7

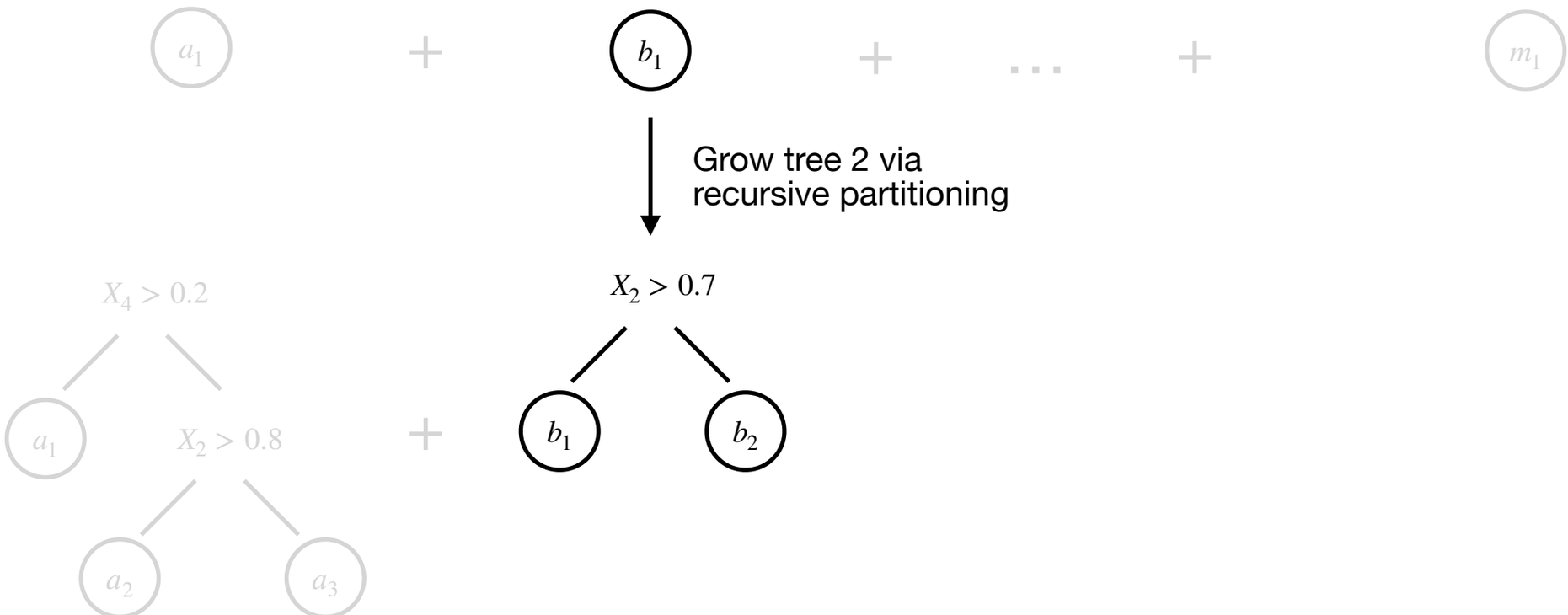
r_2
$2.5 - a_1$
$-4 - a_3$
$3.1 - a_2$
$-3.7 - a_1$
…
$-1.4 - a_2$
$2.7 - a_1$

Subtract predictions from first tree, use residuals as training data for second tree

Boosting fits ensemble of trees using recursive partitioning

r_1
2.5
-4
3.1
-3.7
...
-1.4
2.7

r_2
$2.5 - a_1$
$-4 - a_3$
$3.1 - a_2$
$-3.7 - a_1$
...
$-1.4 - a_2$
$2.7 - a_1$



Boosting fits ensemble of trees using recursive partitioning

r_1
2.5
-4
3.1
-3.7
…
-1.4
2.7

r_2
$2.5 - a_1$
$-4 - a_3$
$3.1 - a_2$
$-3.7 - a_1$
…
$-1.4 - a_2$
$2.7 - a_1$

Subtract predictions from first $m - 1$ trees, use residuals as training data for tree m

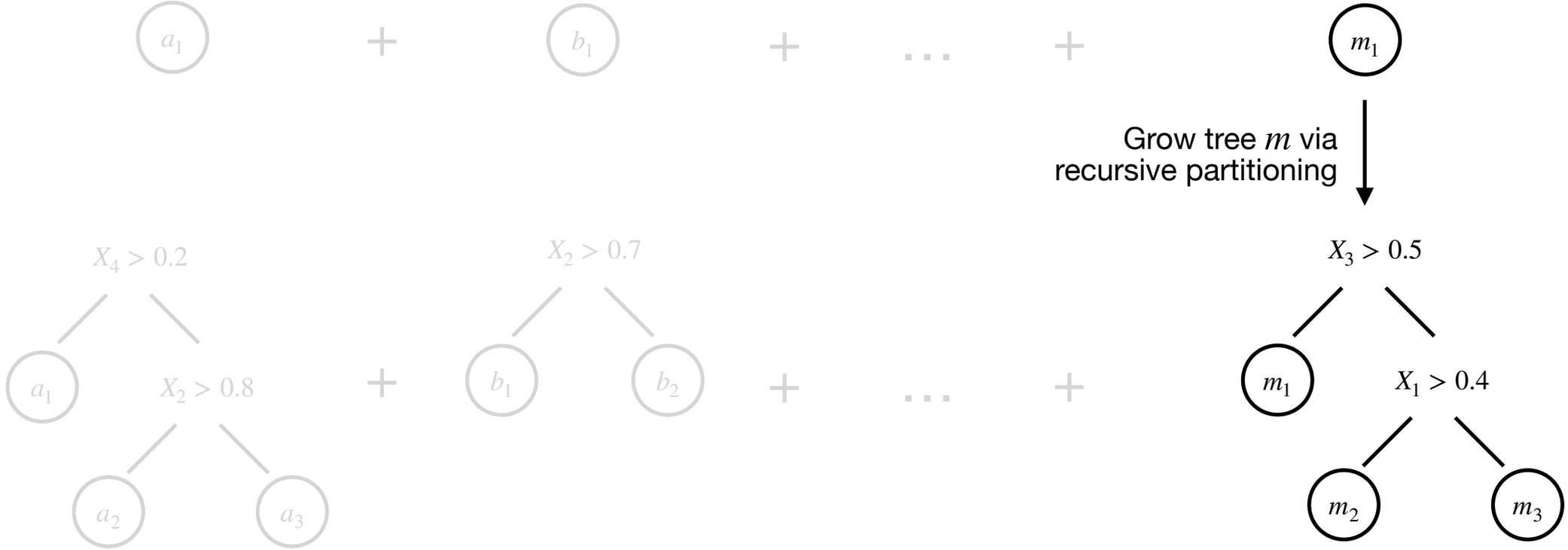
r_m
$2.5 - a_1 - \ldots$
$-4 - a_3 - \ldots$
$3.1 - a_2 - \ldots$
$-3.7 - a_1 - \ldots$
…
$-1.4 - a_2 - \ldots$
$2.7 - a_1 - \ldots$

Boosting fits ensemble of trees using recursive partitioning

r_1
2.5
-4
3.1
-3.7
…
-1.4
2.7

r_2
$2.5 - a_1$
$-4 - a_3$
$3.1 - a_2$
$-3.7 - a_1$
…
$-1.4 - a_2$
$2.7 - a_1$

r_m
$2.5 - a_1 - \ldots$
$-4 - a_3 - \ldots$
$3.1 - a_2 - \ldots$
$-3.7 - a_1 - \ldots$
…
$-1.4 - a_2 - \ldots$
$2.7 - a_1 - \ldots$



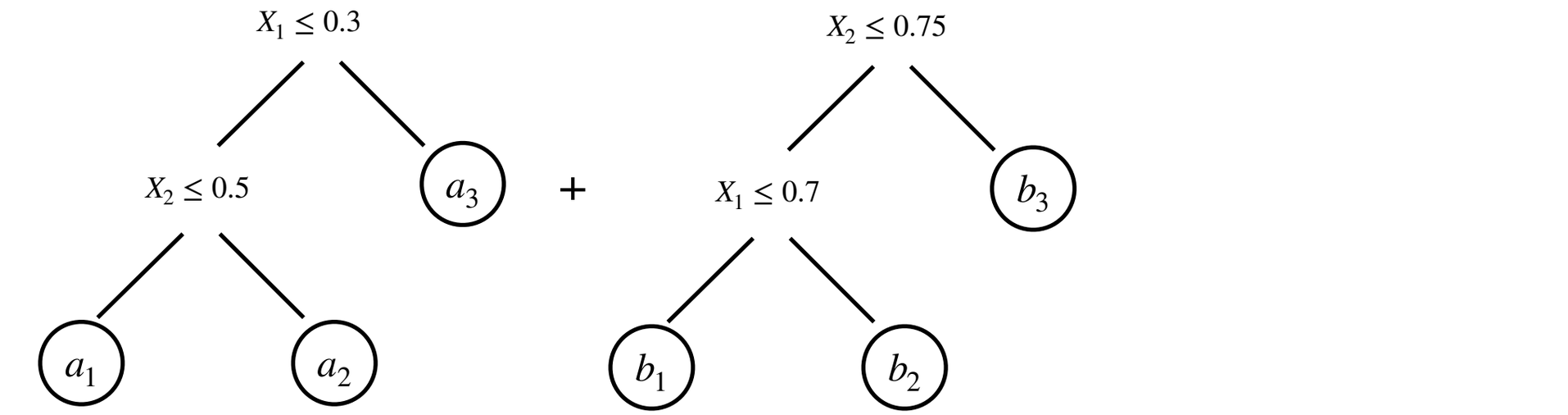
Two important ideas create “weak learners”:
“Stubby” trees and “squished” predictions

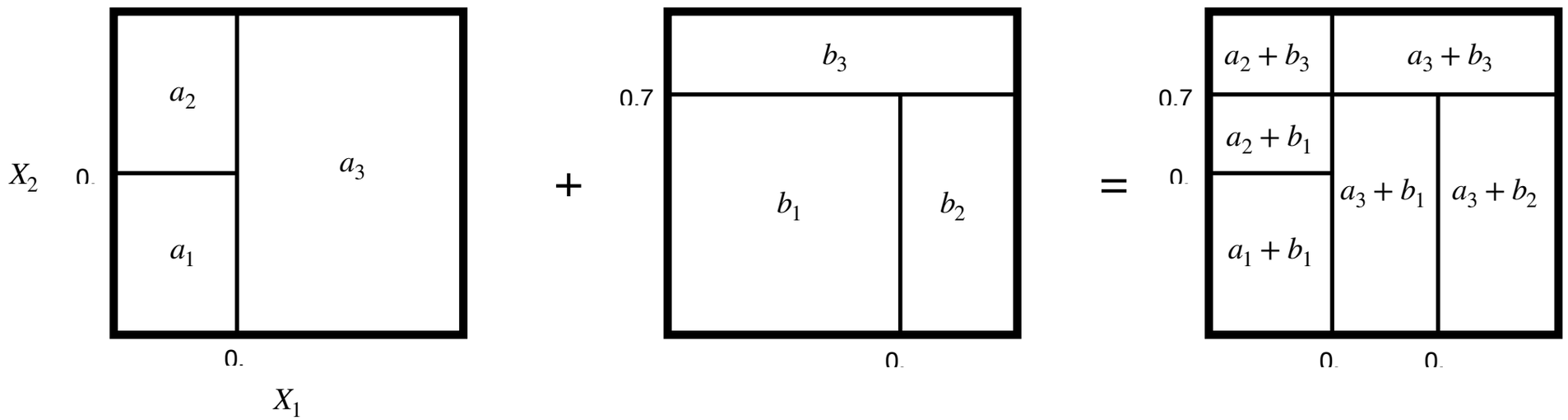
BART ≈ Boosting (Kind of)

The BART prior

- BART isn't a model, it's a prior distribution over functions:

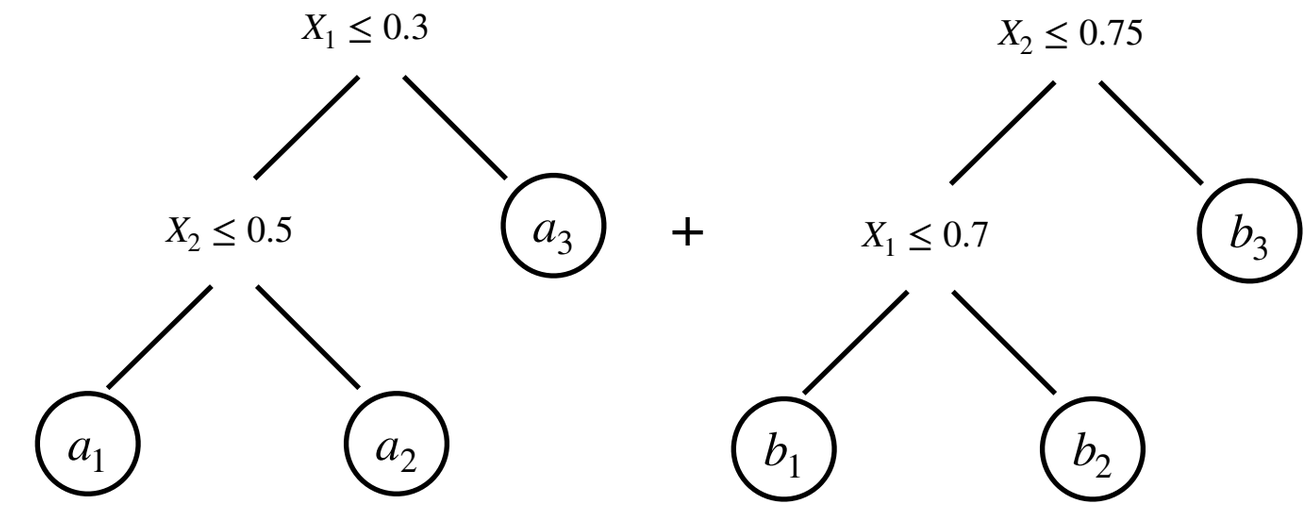
$$f(X_i) = \sum_{j=1}^{m} g(X_i, T_j, M_j)$$



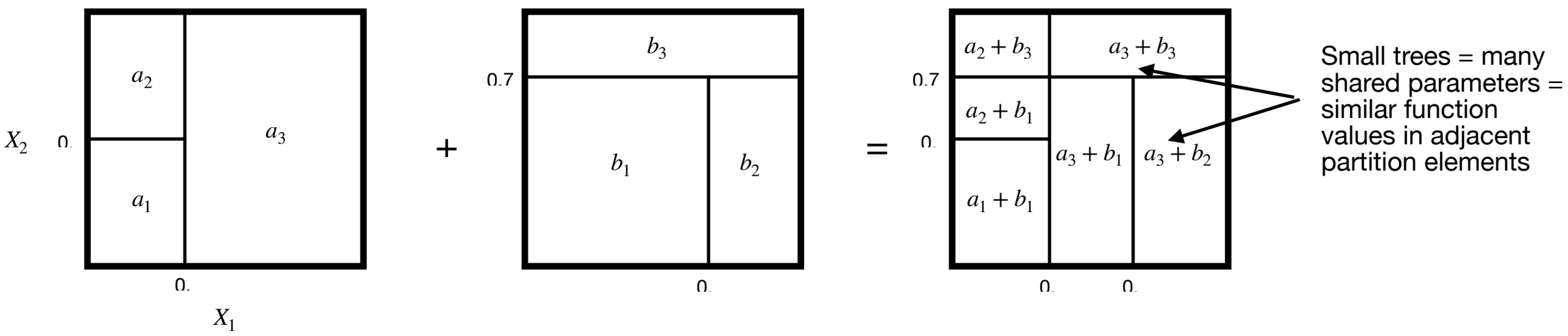


The BART prior

- The prior distribution discourages (penalizes) complexity



- Trees should be small (simple basis functions, few jumps)
- End-node parameters should be similar (small jumps)
- (Optionally) Splittting rules should use few variables
- The result is a relatively smooth function



The BART prior

$$g(X_i, T, \mu) = \sum_{\ell \in \text{leaves}(T)} 1\left(X_i \in \ell\right) \times \mu_\ell$$

Decision tree Leaf parameters Mutually exclusive partition

Leaf parameters have normal priors with calibrated scales:

$$\mu_\ell \sim N(0, v_f/m) \rightarrow f(x) \sim N(0, v_f)$$

Decision tree T is given a prior $p(T)$ over its structure by specifying, for each node η of depth d_η,

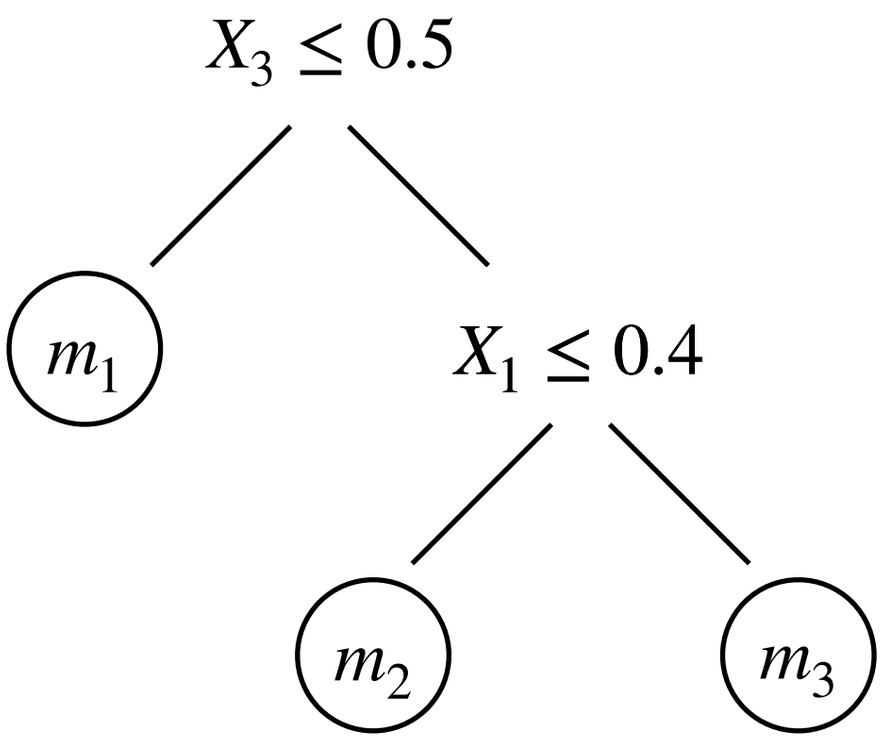
$$p_{\text{split}}(\eta, T) = \alpha(1 + d_\eta)^{-\beta}, \qquad p_{\text{rule}}(\eta, T) \propto 1$$

Probability a tree splits at least once

Split probability decays geometrically in depth

Sampling from the BART posterior

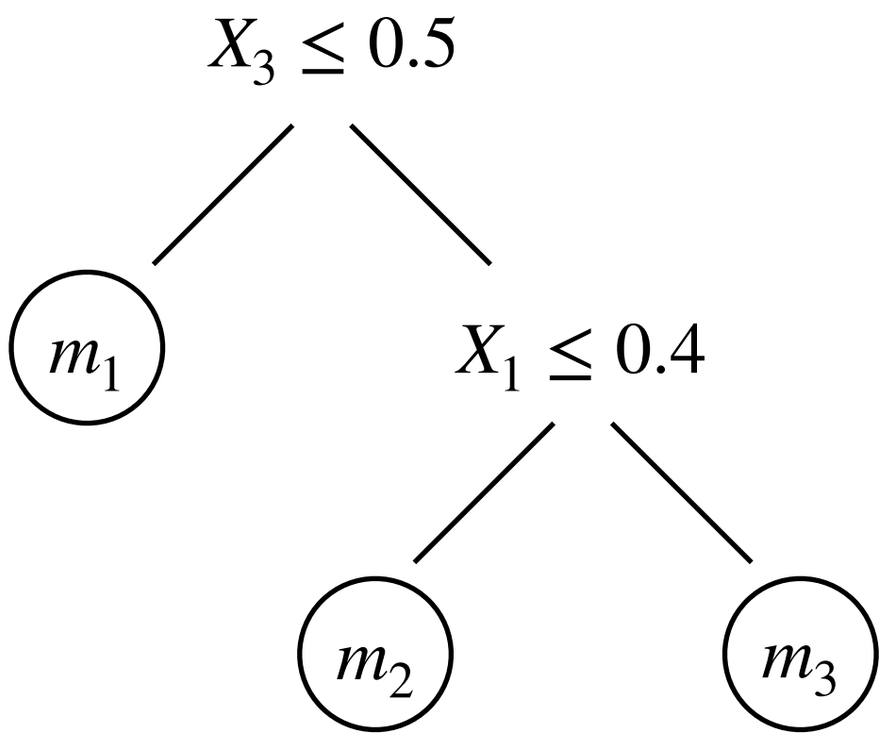
Consider small changes to an individual tree



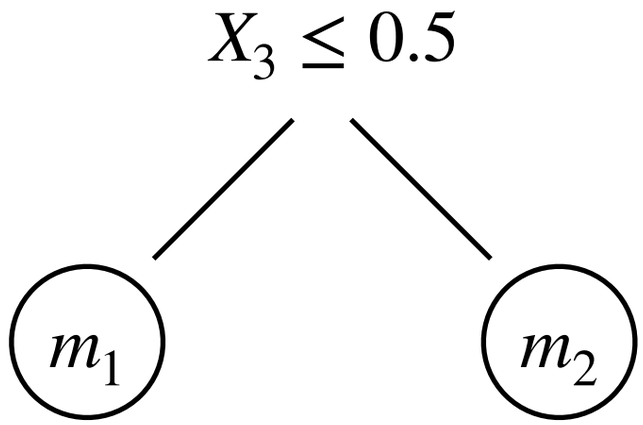
For this tree, we can either

- **Grow**: pick any of the three leaf nodes and split it
- **Prune**: pick the "leaf parent" node that splits X_1 at 0.4 and convert it back to a leaf

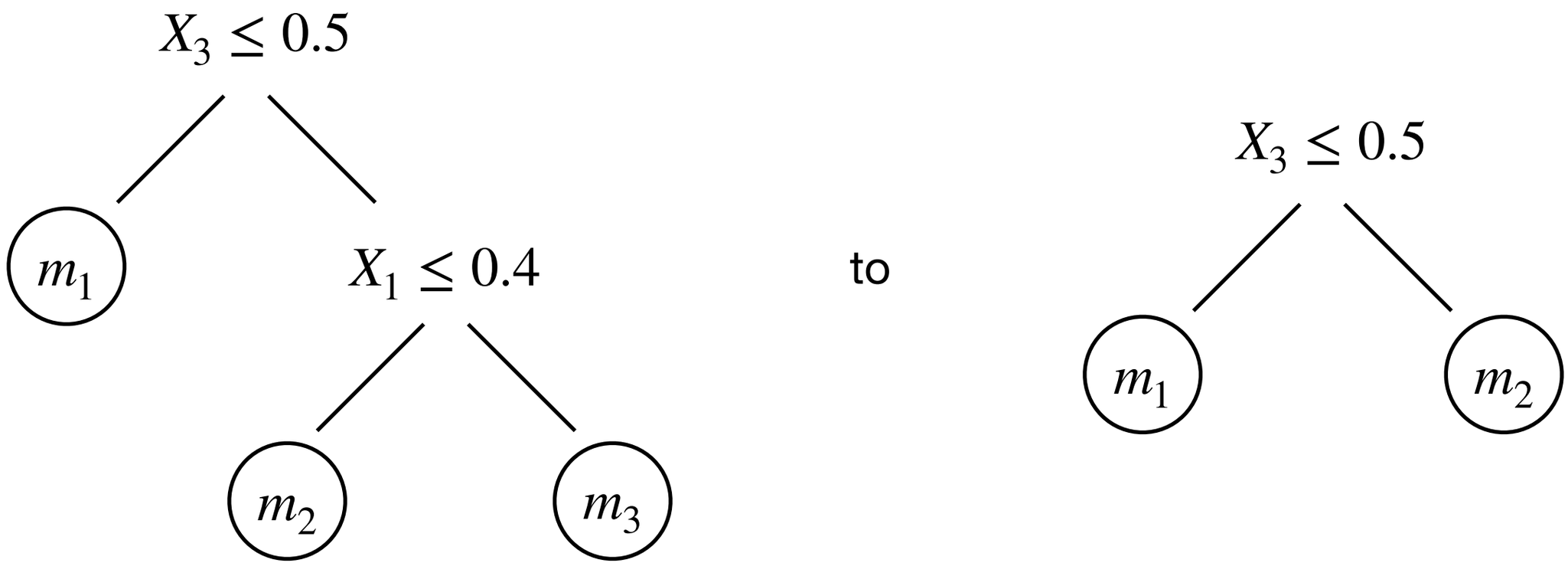
Pruning proposes a smaller tree



to

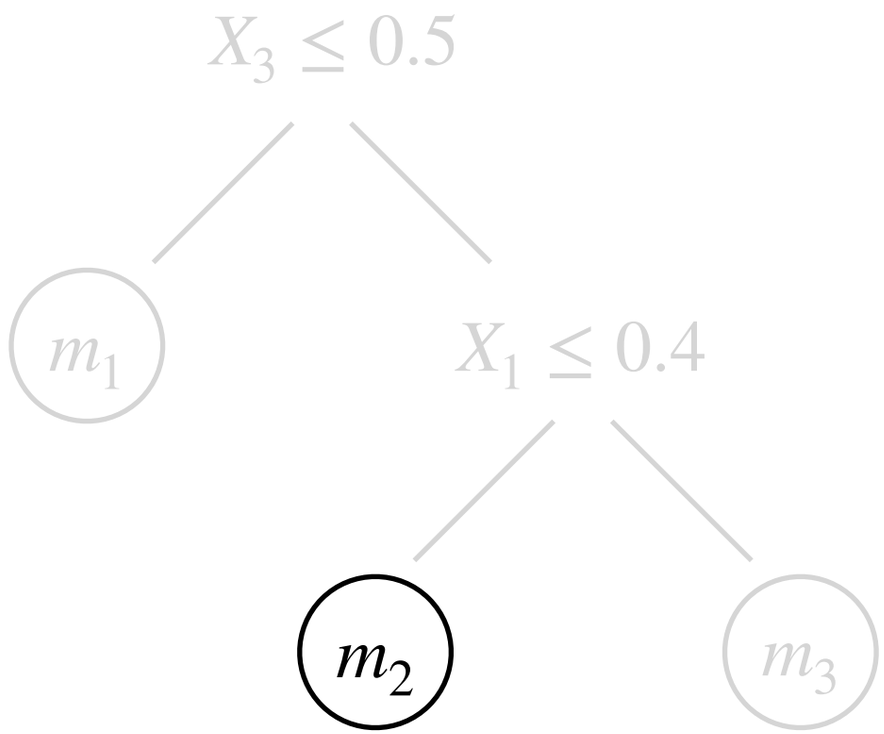


Pruning proposes a smaller tree



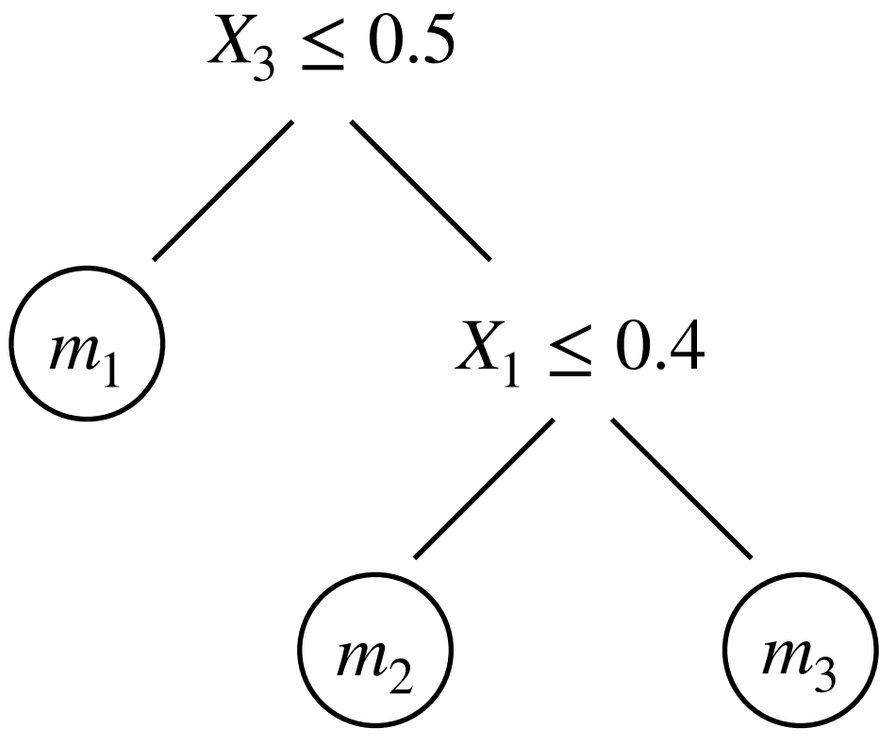
Next tree sample is chosen via the Metropolis-Hastings algorithm: randomly accept or reject based on how well the data supports the pruned tree (likelihood + prior)

Growing proposes a larger tree … and a split rule!



First, pick a leaf at random to split

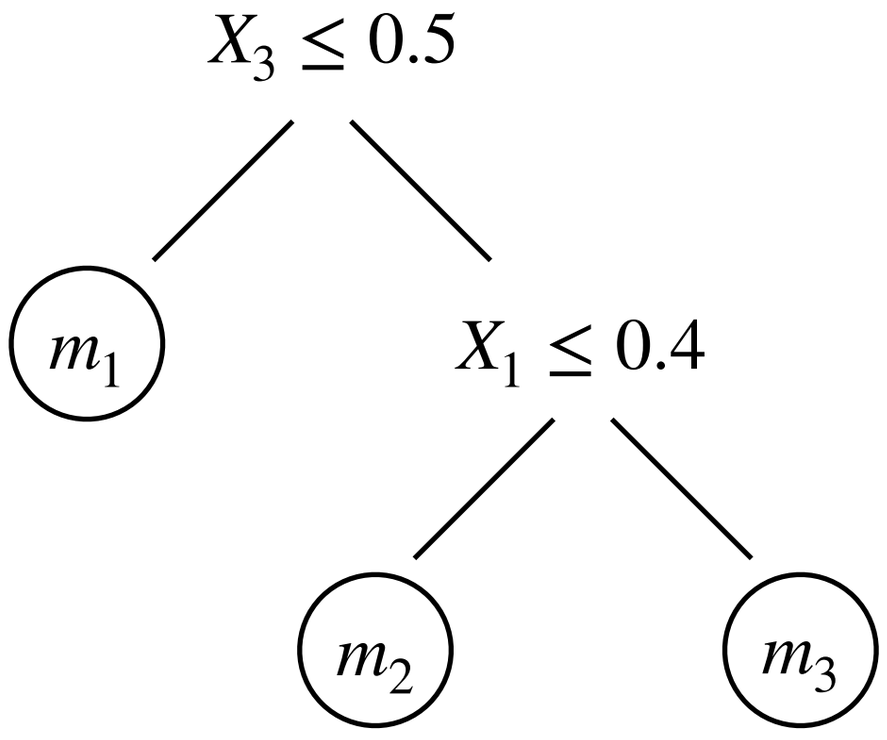
Growing proposes a larger tree … and a split rule!



X_1	X_2	X_3	X_4
4.3	5.2	-7.6	0.5
2.7	0.6	-1.2	3.2
…	…	…	…
-5.2	4.1	0.9	4.1

Then, pick a feature at random to define the split rule

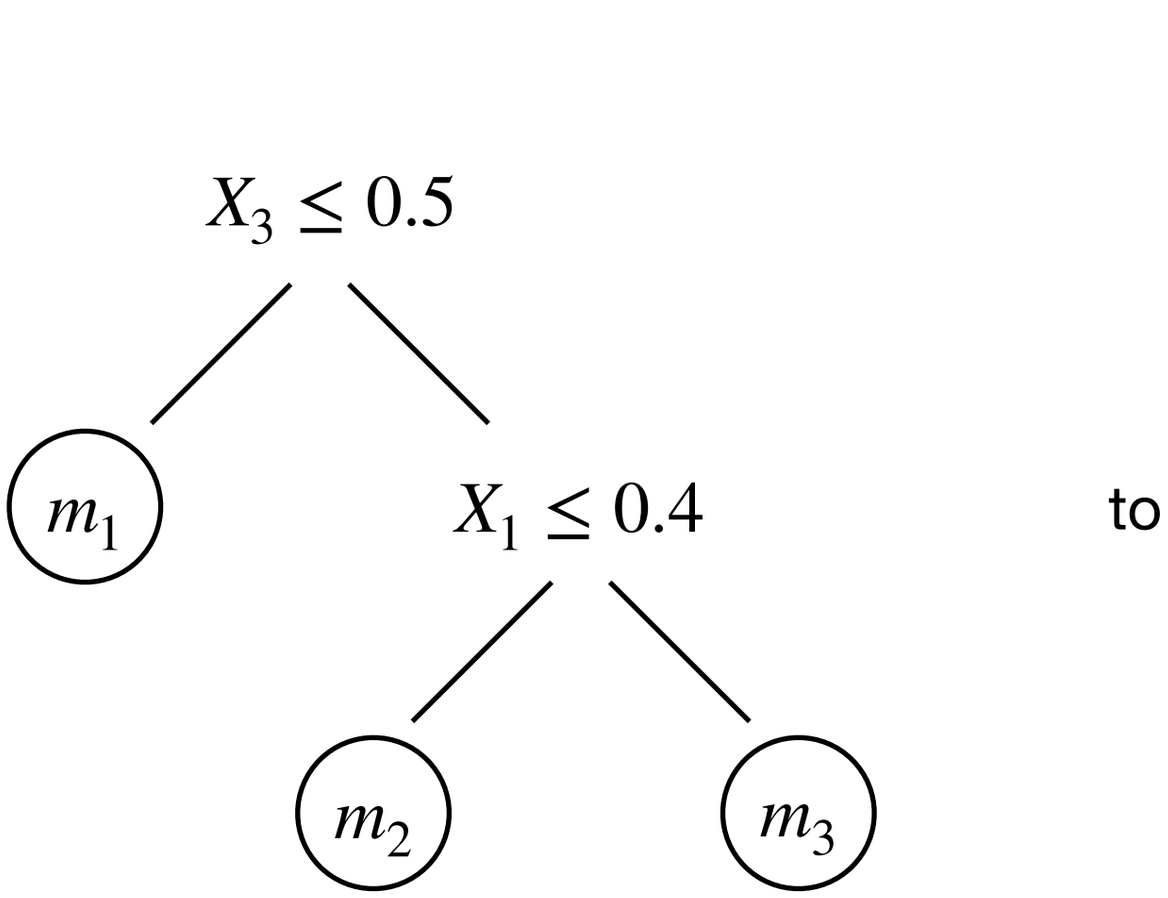
Growing proposes a larger tree … and a split rule!



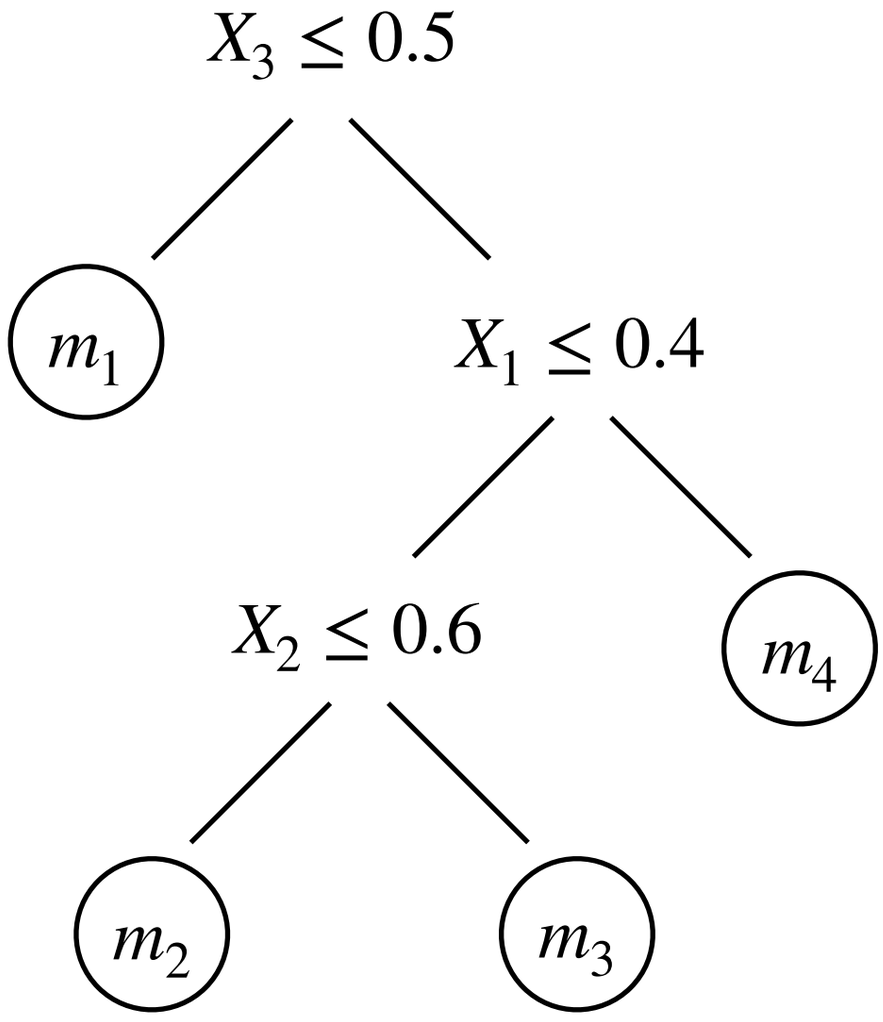
X_1	X_2	X_3	X_4
4.3	5.2	-7.6	0.5
2.7	0.6	-1.2	3.2
…	…	…	…
-5.2	4.1	0.9	4.1

Then, pick an available split point for that feature

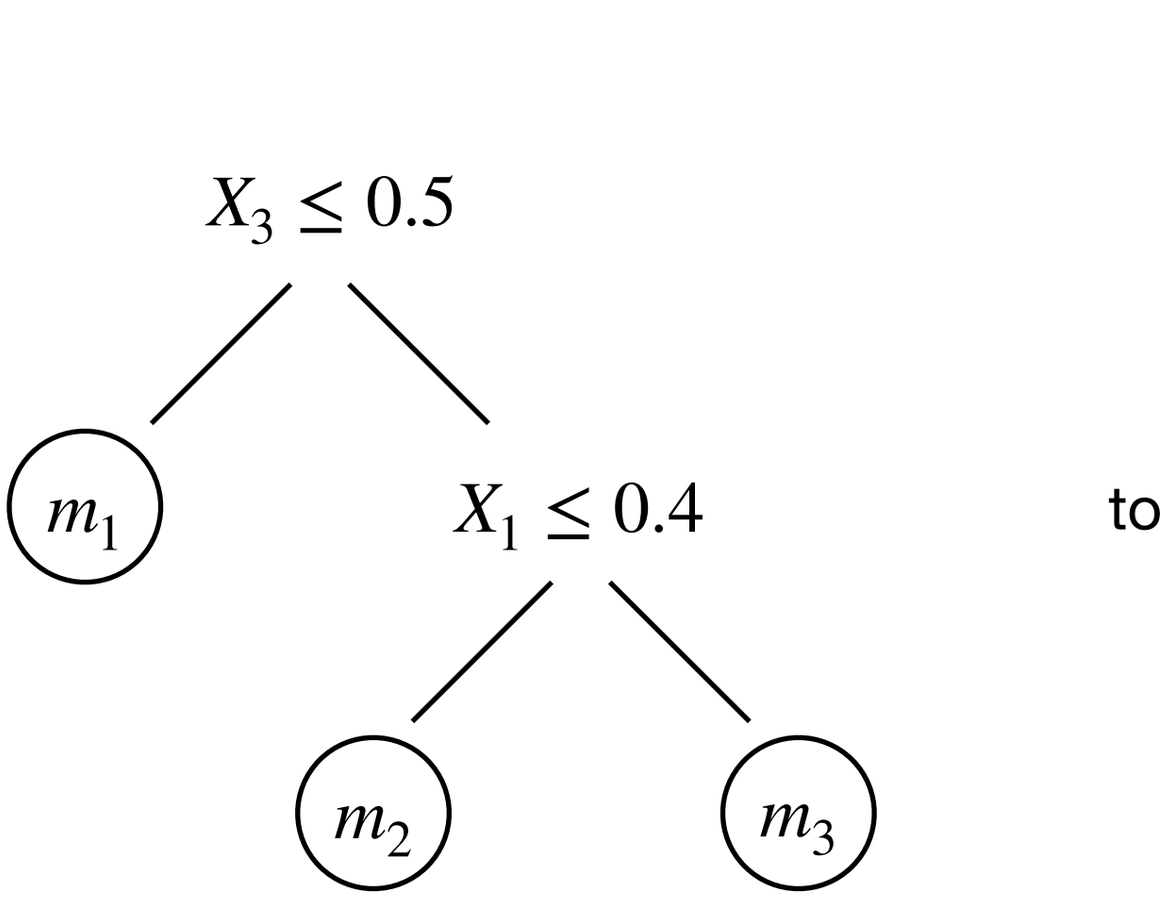
Growing proposes a larger tree … and a split rule!



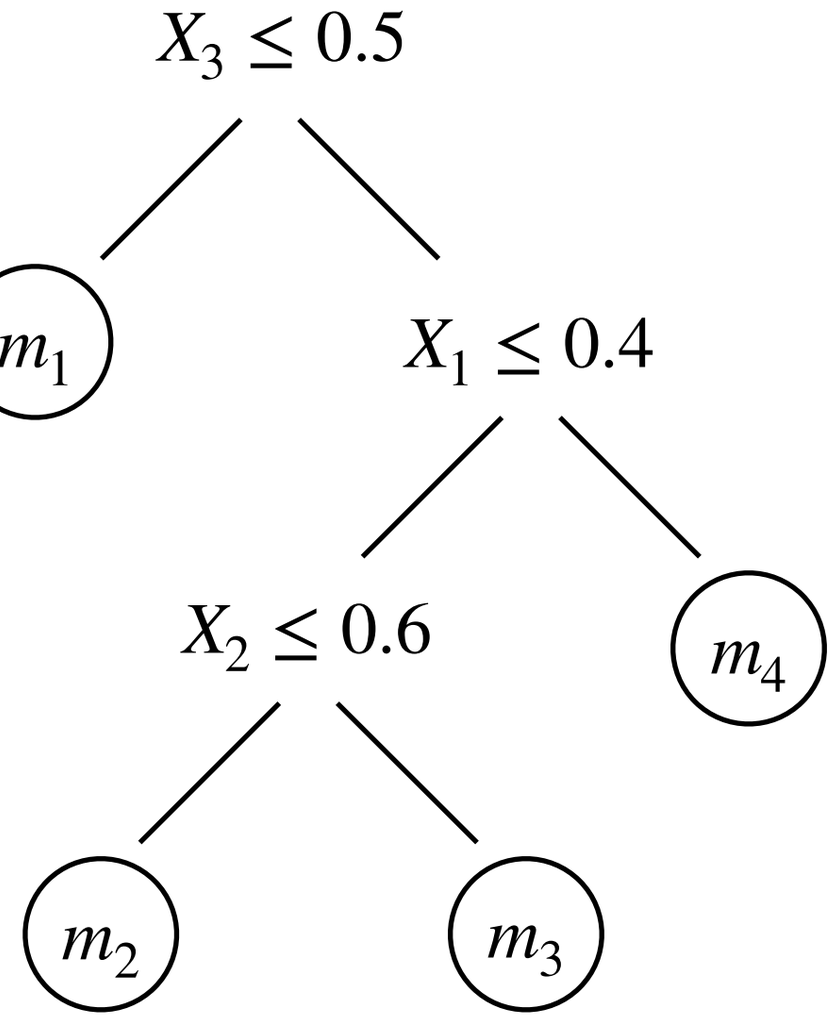
to



Growing proposes a larger tree … and a split rule!



to



Tree is accepted or rejected based on the same algorithm as in the “prune” step

BD-MCMC samples from the function posterior

- The full BART sampler sampler proceeds for each T_j as

  1. Draw tree from $p(T_j \mid r, X, \Psi)$ (as in previous slides)

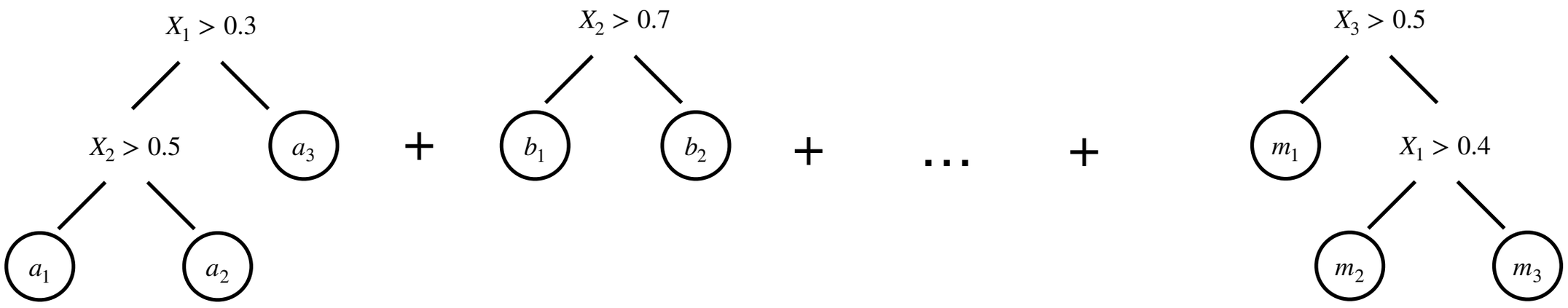
  2. Draw leaf node parameters from $p(\mu_j \mid T_j, r, X, \Psi)$

Usually r is the partial residual (outcome y net of all other model predictions (e.g. trees not including T_j))

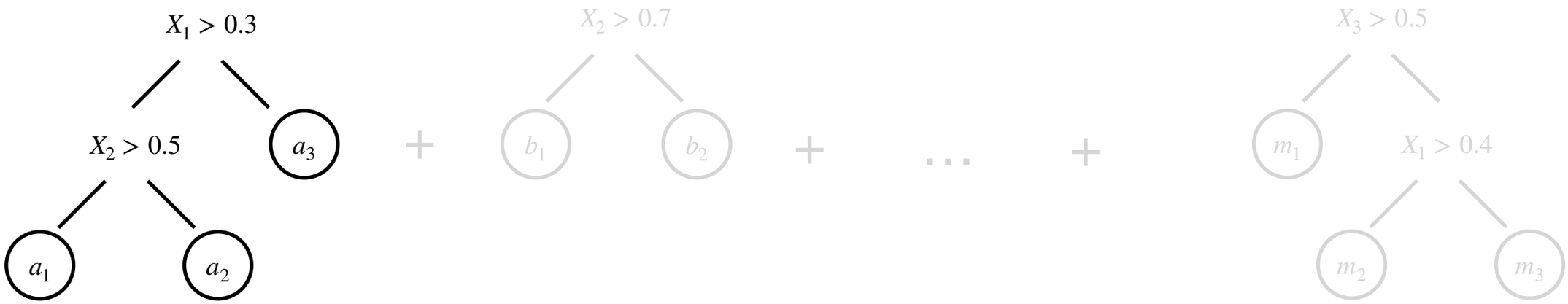
Ψ includes all other (non-tree) model parameters

This process is known as “Birth-Death” MCMC

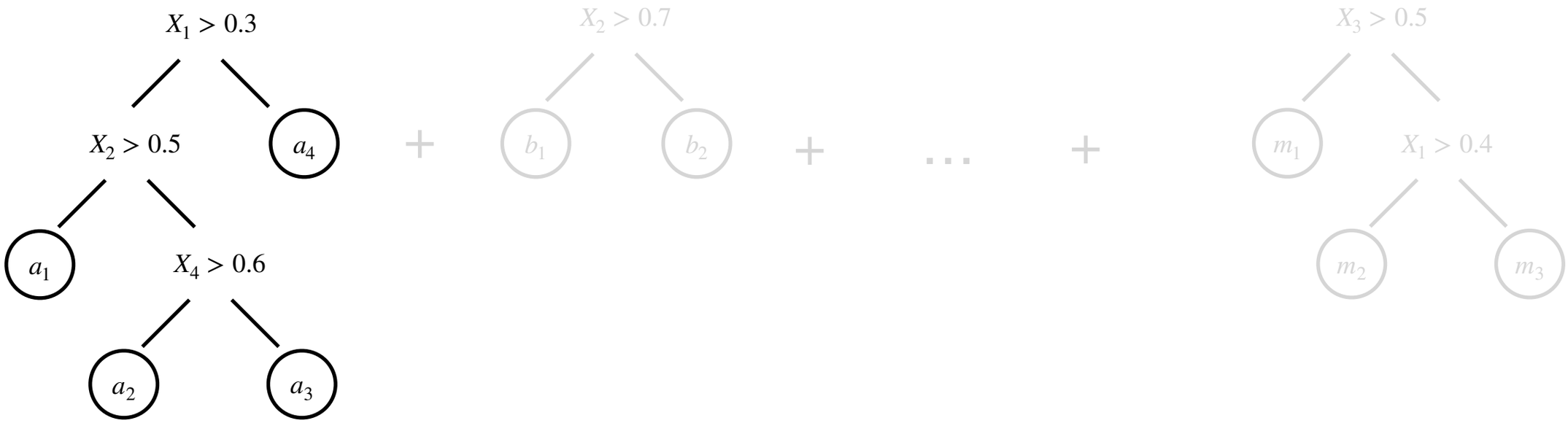
One MCMC iteration repeats this process for each tree



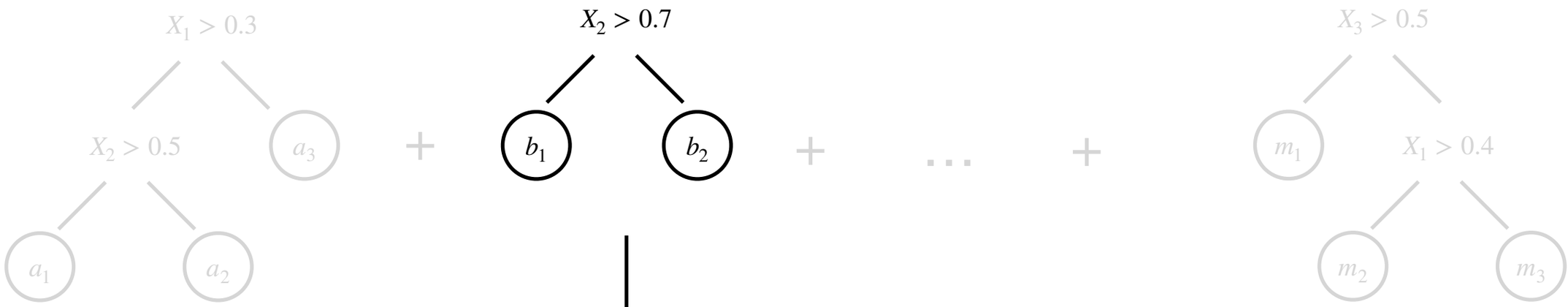
One MCMC iteration repeats this process for each tree



Step 1a: Birth/death MCMC for tree 1

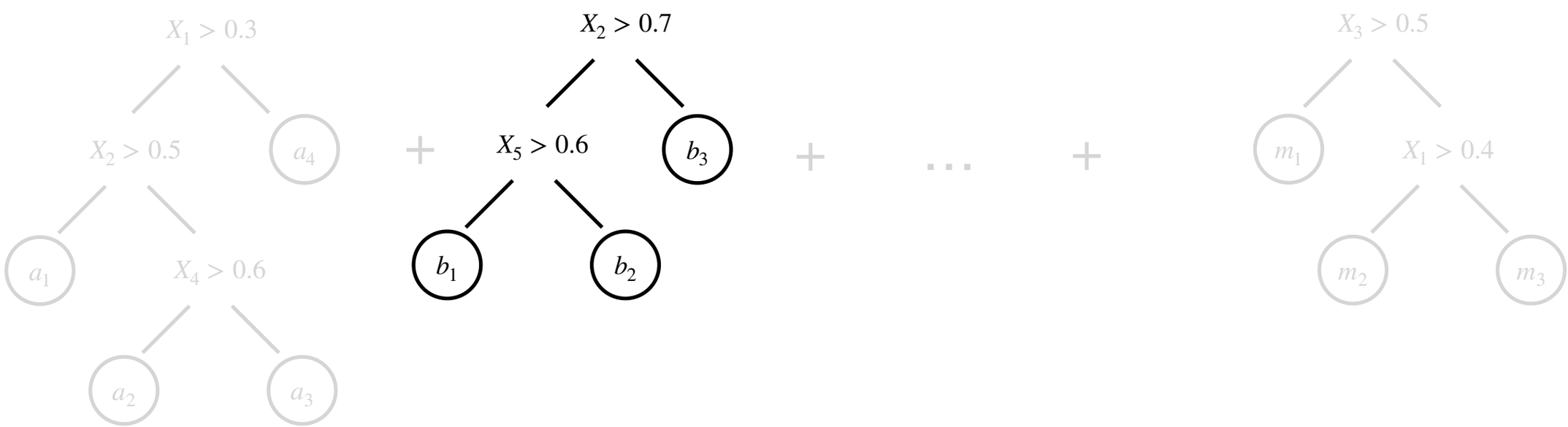


One MCMC iteration repeats this process for each tree



Step 1a: Birth/death MCMC for tree 1

Step 1b: Birth/death MCMC for tree 2



One MCMC iteration repeats this process for each tree

Step 1a: Birth/death MCMC for tree 1

Step 1b: Birth/death MCMC for tree 2

Step 1m: Birth/death MCMC for tree m

Ensemble repeats this process for each tree

Step 2: Sample other model parameters

Summary

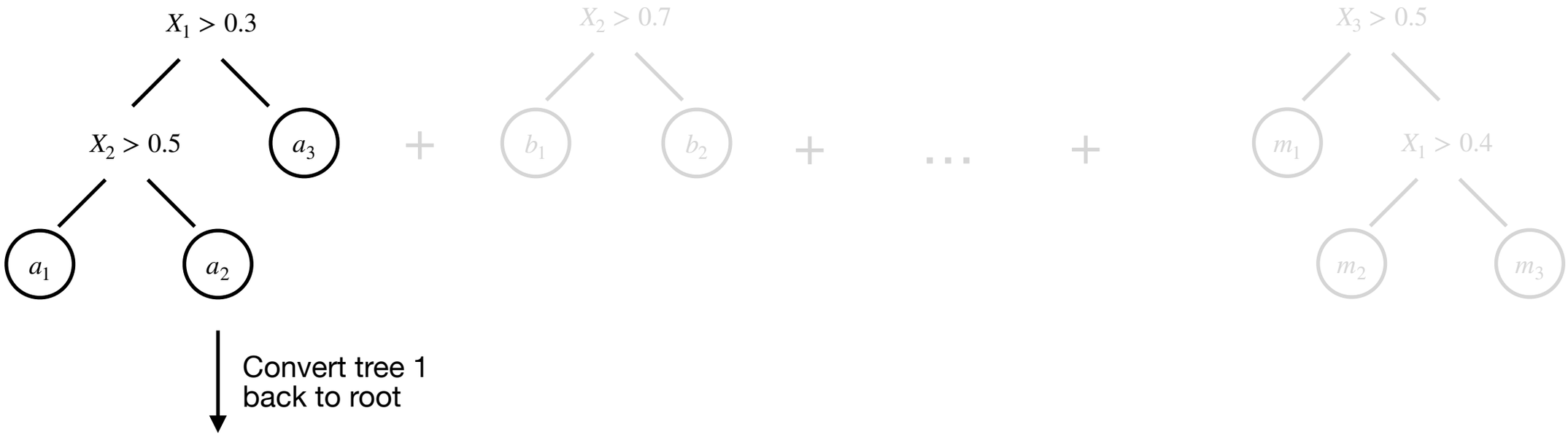
- After proposing a move to each tree, the result is a single posterior sample of f from its posterior
- Automatically generates posteriors over evaluations/ transformations/aggregations/functionals of f
- Subsequent steps sample additional parameters, e.g. $\sigma^2 = Var[\epsilon_i]$ in the regression model

$$y_i = f(X_i) + \epsilon_i = \sum_{j=1}^{m} g(X_i, T_j, M_j) + \epsilon_i$$

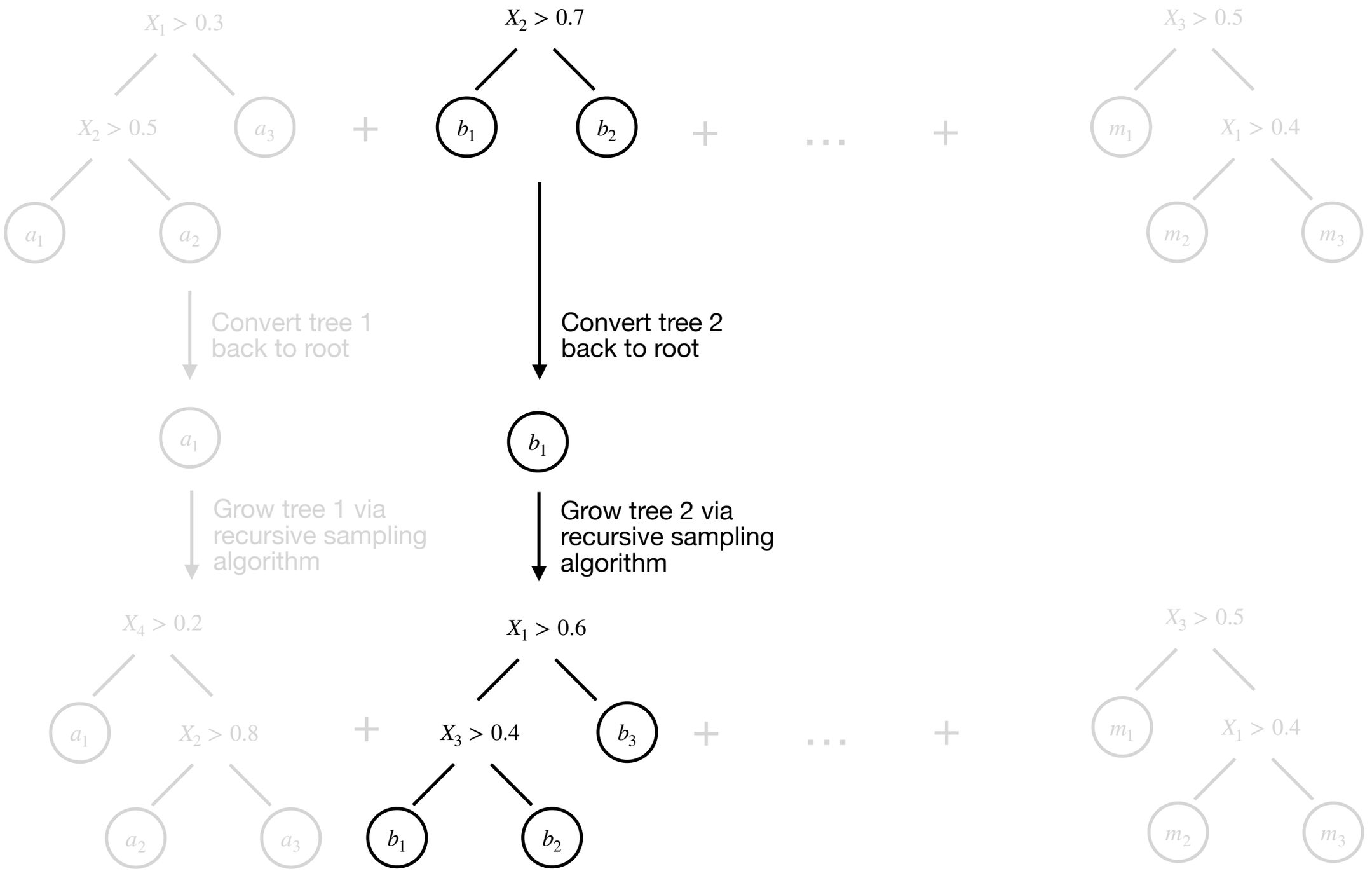
This can be slow!

XBART replaces BART birth/death MCMC with recursive algorithm that grows new tree from scratch in each sample

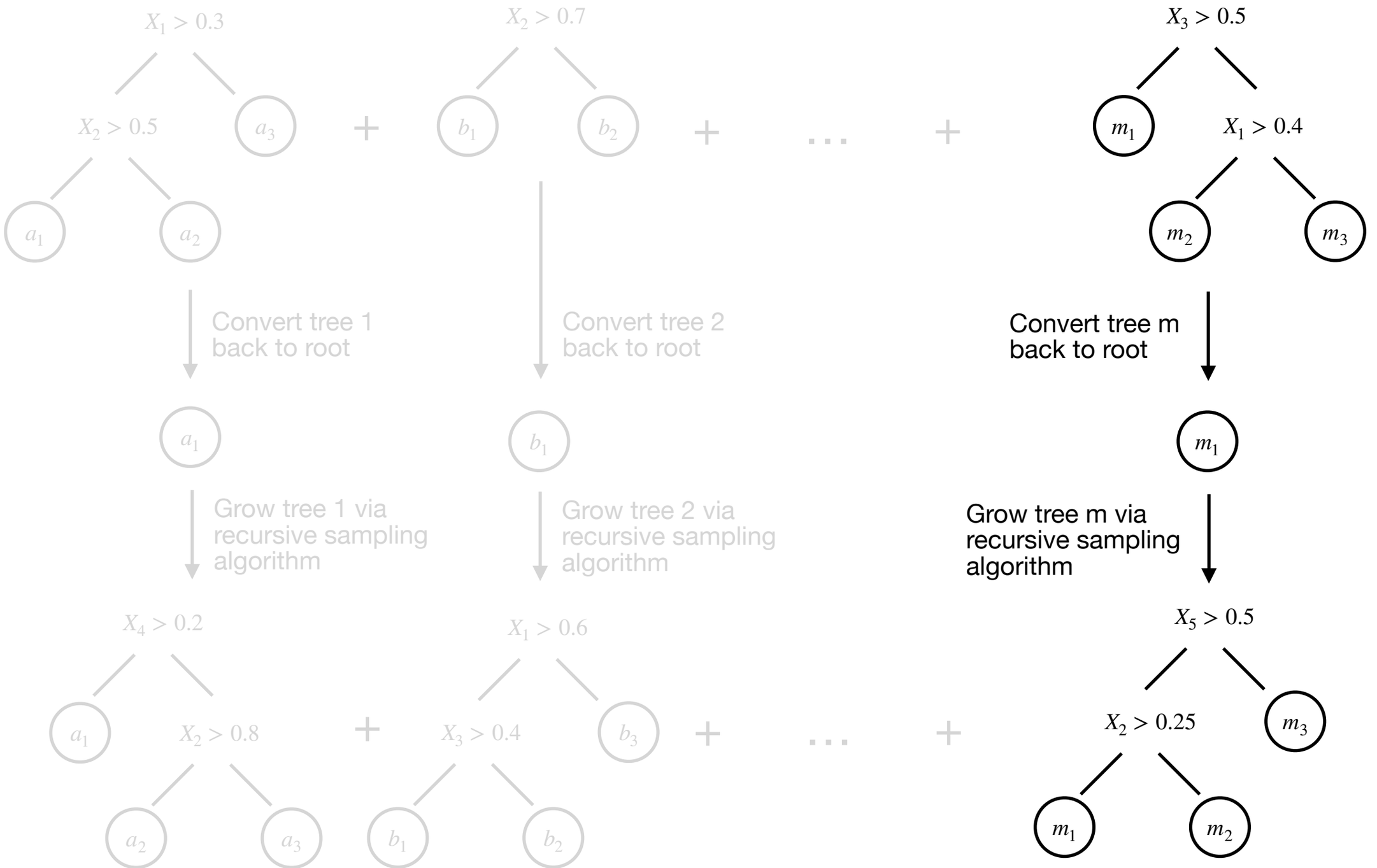
XBART re-samples trees recursively from root



XBART re-samples trees recursively from root



XBART re-samples trees recursively from root



XBART re-samples trees recursively from root

Step 2: Sample other XBART parameters

Recursive “grow-from-root” algorithm, in detail

At any given node k for tree i, we have:

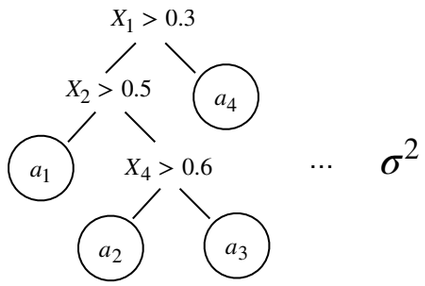
Outcome

r
-4.4
6.1
…
2.4

Covariates

X1	X2	…	Xp
0.75	2.8		1.4
0.4	1.5	…	2.0
…	…		…
0.65	-2.1		0.9

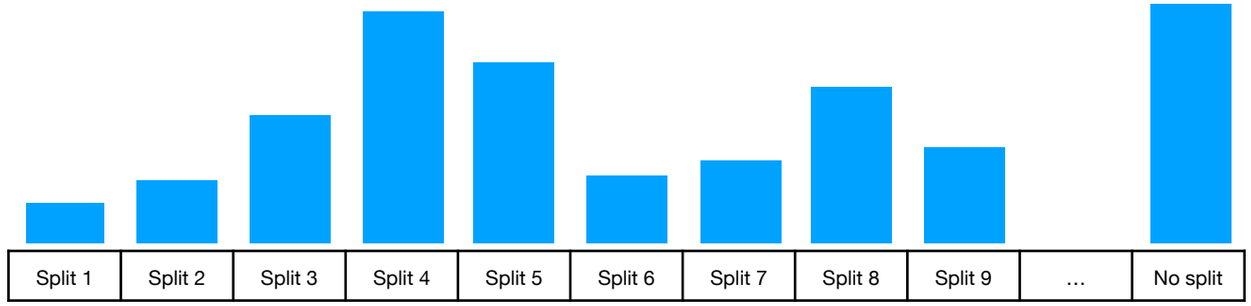
Global model parameters and trees other than j



(Sorted) cutpoint candidates

X1	X2	…	Xp
0.1	-5		0.8
…	…	…	…
0.95	3.8		2.1

Evaluate split criteria of each cutpoint candidate as well as “no split”



Sample next move based on enumerated split criteria

“No split” selected?

No

Sift outcome and covariates into left and right “children” of node k

Determine cutpoint candidates for left and right children

Call “grow-from-root” recursively on node k’s left child

Call “grow-from-root” recursively on node k’s right child

Yes

Sample leaf parameters for node k and stop recursing

Split criterion based on BART model

For observations in node k,

$$m(y_k \mid \psi, \phi) = \int \ell(y_k \mid \mu, \psi)\pi(\mu \mid \phi)d\mu$$

Outcome parameters

Prior parameters

Leaf parameter

Cutpoint j in node k divides y_k into y_{jk}^l and y_{jk}^r with marginal likelihoods $m(y_{jk}^l \mid \psi, \phi)$ and $m(y_{jk}^r \mid \psi, \phi)$, and split criteria are defined as

$$L(y_{jk}) = m(y_{jk}^l \mid \psi, \phi)m(y_{jk}^r \mid \psi, \phi)$$

"No split" criterion is defined as

Depth of node k

$$L(\emptyset) = |\mathcal{C}|\left(\frac{(1+d_k)^\beta}{\alpha} - 1\right) m(y_k \mid \psi, \phi)$$

Number of cutpoints

This no split correction implies a prior split probability (ignoring marginal likelihood) of $\alpha(1+d_k)^{-\beta}$

Leaf model is (almost) the same

BART model:

$$Y \mid T, X, \sigma^2 \sim \mathcal{N}\left(f(X), \sigma^2\right)$$

$$f(X) = \sum_{t \in T} g_t(X) = \sum_{t \in T} \sum_{j} \mu_{jt} \mathbf{1}\left\{X \in n_{jt}\right\}$$

$$\mu_{jt} \sim \mathcal{N}\left(\mu_0, \tau\right)$$

$$\sigma^2 \sim \text{IG}\left(\frac{\nu}{2}, \frac{\nu\lambda}{2}\right)$$

XBART model:

$$Y \mid T, X, \sigma^2, \tau^2 \sim \mathcal{N}\left(f(X), \sigma^2\right)$$

$$f(X) = \sum_{t \in T} g_t(X) = \sum_{t \in T} \sum_{j} \mu_{jt} \mathbf{1}\left\{X \in n_{jt}\right\}$$

$$\mu_{jt} \sim \mathcal{N}\left(\mu_0, \tau\right)$$

$$\sigma^2 \sim \text{IG}\left(\frac{\nu}{2}, \frac{\nu\lambda}{2}\right)$$

$$\tau \sim \text{IG}\left(a_\mu, b_\mu\right)$$

Leaf node scale parameter has prior

In practice, data are centered and scaled so that $\mu_0 = 0$ and $\sigma_0, \nu, \lambda, a_\mu, b_\mu$ are easy to calibrate

Tradeoffs: BART vs XBART

XBART converges faster …

… but is no longer a valid Bayesian procedure

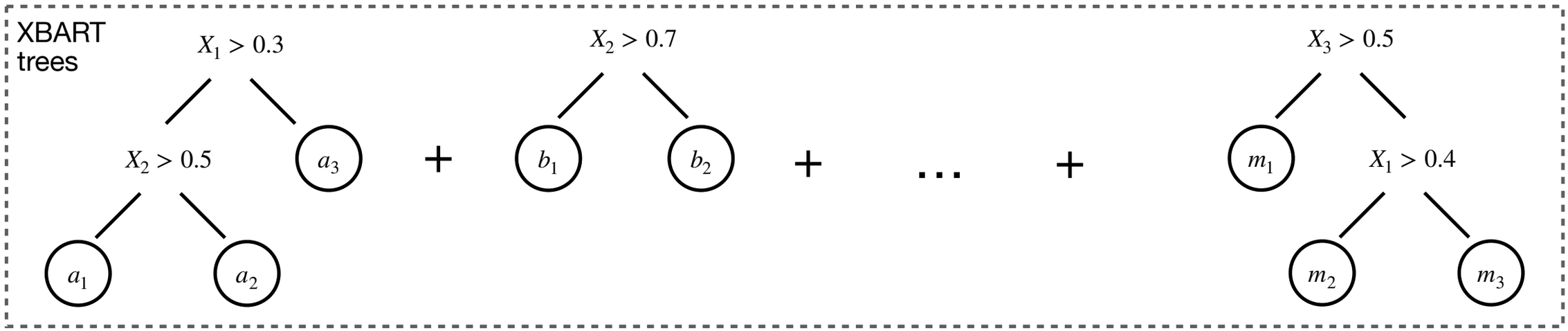
- BART MCMC ensures that samples converge to desired posterior
- XBART's recursive sampling procedure **does** define a Markov chain that converges to a stationary distribution …

  … but it's certainly not the BART posterior …

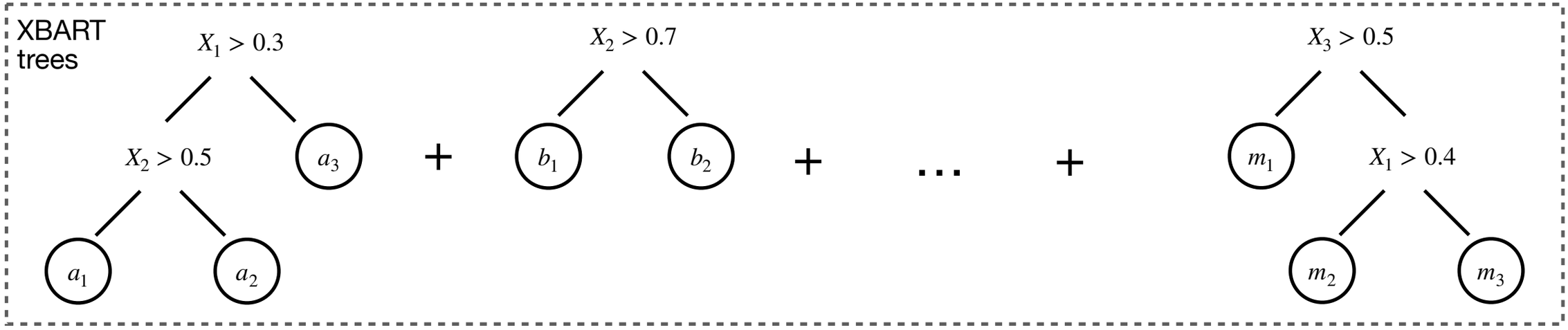
  … and unclear whether it constitutes a posterior of **any** model
- XBART tends to under-represent uncertainty

There is a solution!

BART MCMC need not start at root

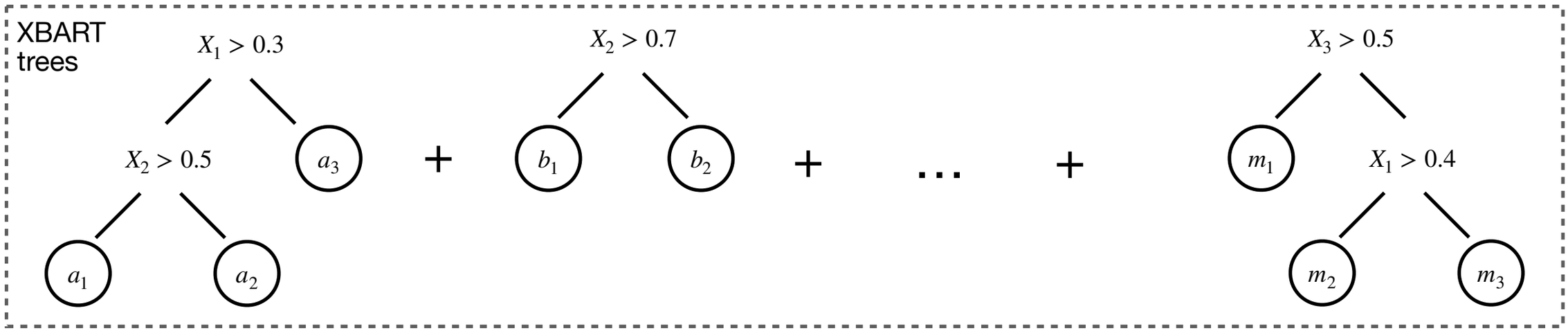


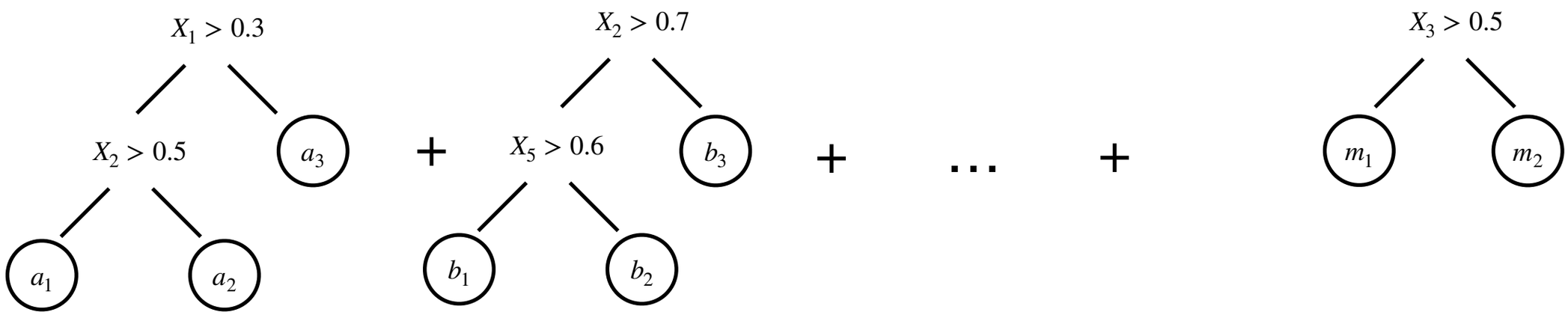
BART MCMC need not start at root



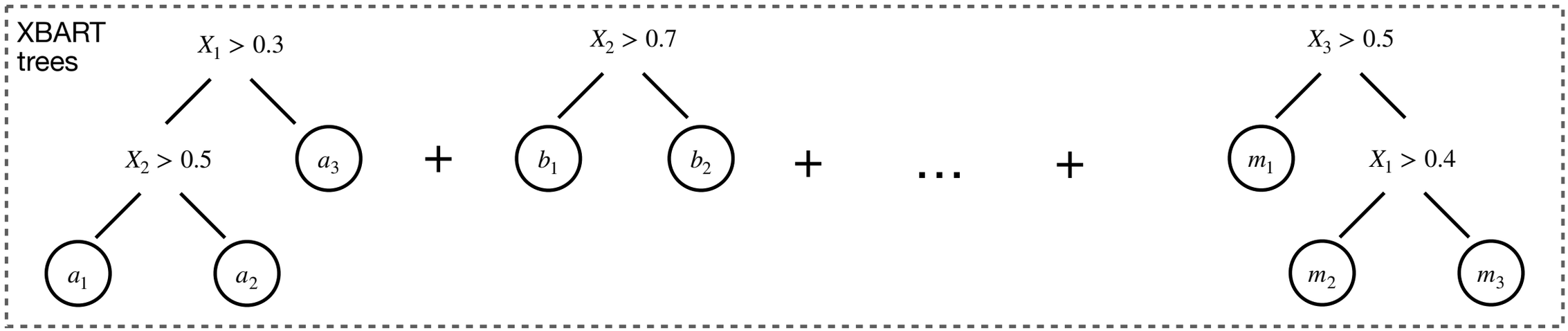
Treat as initial state for BART MCMC

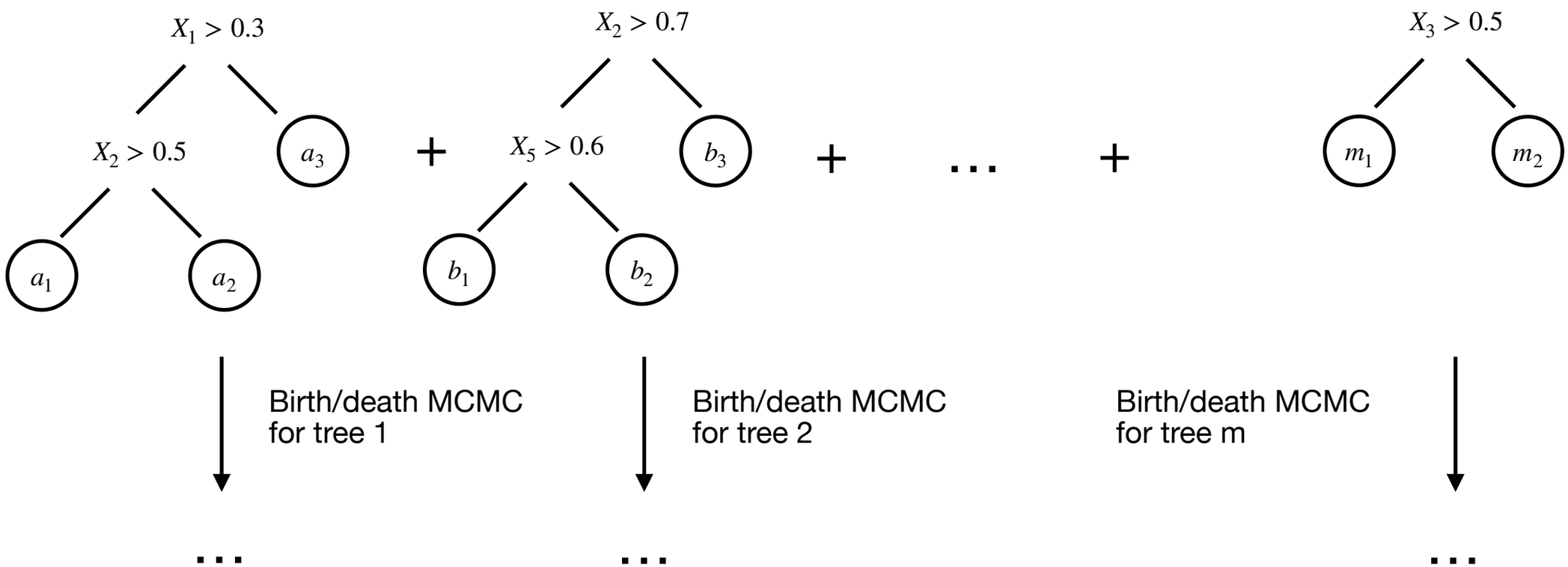
BART MCMC need not start at root



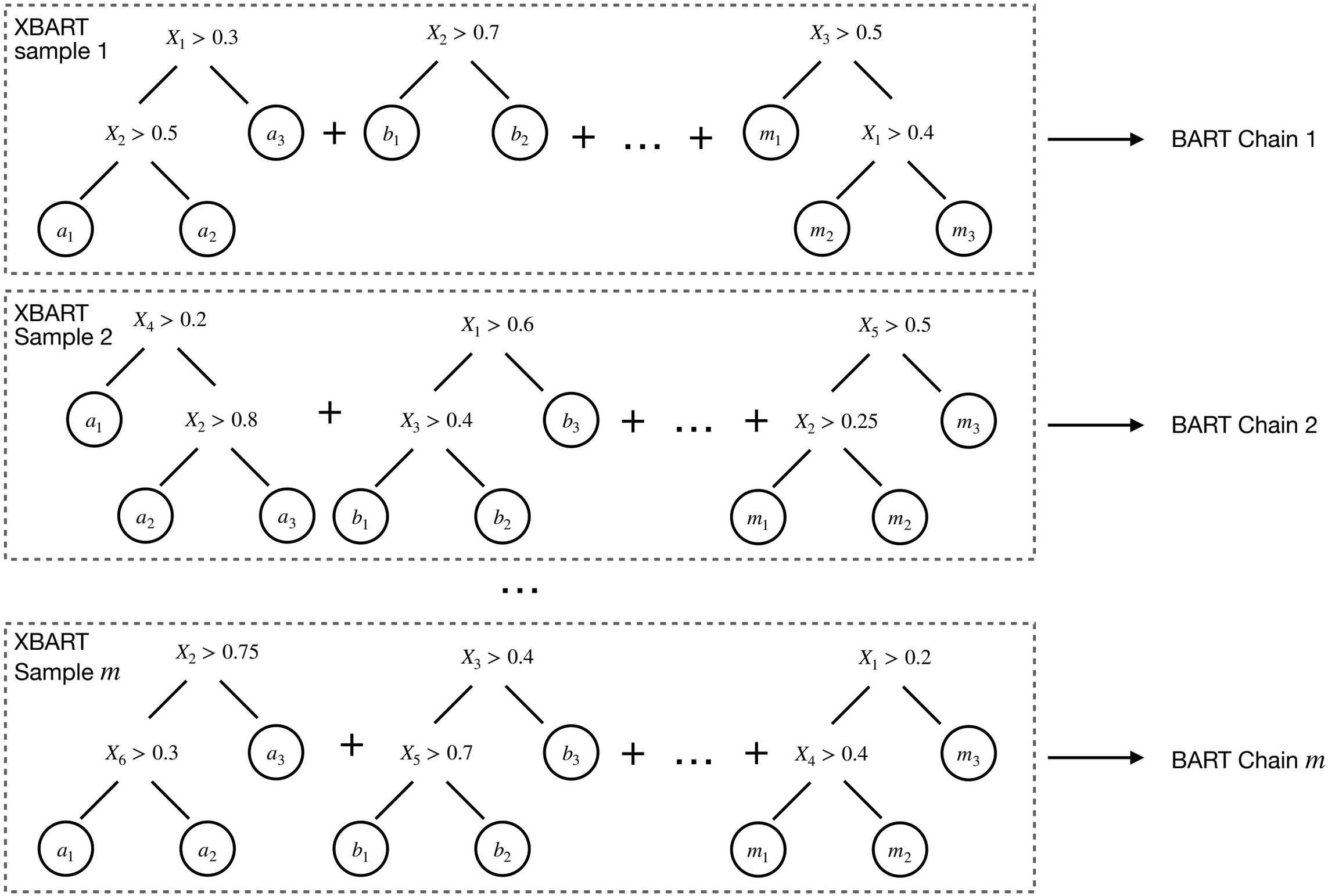


BART MCMC need not start at root





Multiple XBART draws initialize separate BART “chains”



Simulations show that “warm-start” BART can give better RMSE and coverage than either XBART or BART alone

DGP (details in appendix)	Metric	XBART	BART	Warm Start BART
Linear model	Coverage	0.78	0.77	0.99
	Interval length	7.82	6.14	9.92
	Run time (s)	5.61	131.17	9.46* (101.86**)
	RMSE	3.11	2.51	1.81
Trig / polynomial	Coverage	0.90	0.74	0.96
	Interval length	3.61	2.89	4.23
	Run time (s)	4.68	92.75	7.7* (80.18**)
	RMSE	1.03	1.27	1.01

* Each warm start BART chain running in parallel
** Each warm start BART chain running sequentially

BART / BCF workflow

- Use XBART / XBCF algorithms for iterative model exploration
- Use XBART / XBCF for large datasets
- “Warm-start” BART / BCF samplers from initial XBART/XBCF fits (default in `stochtree`)
    - XBART produces overdispersed and reasonable starting values for MCMC
    - My personal experience: Convergence in minutes for models I needed hours or days to fit!

Bayesian Tree Models for Causal Inference

Bayesian Tree Models for Causal Inference

- Hopefully we've motivated tree-based methods for learning functions
- Why **Bayesian** tree models?
  - Robust regularization through prior distributions with sensible defaults — no "dark art" of tuning parameters
  - Inference via posterior distributions is simple and (empirically) often yields good frequentist properties
  - Embedding tree methods in a *model* makes extensions straightforward — for example, to multilevel data

A simple multilevel linear model

Group-specific intercepts/
fixed/random effects

Group-specific “unexplained”
heterogeneity

$$y_{ij} = \alpha_j + \mathbf{x}'_{ij}\beta + \left[\mathbf{w}'_{ij}\tau + \phi_j\right] z_{ij} + \epsilon_{ij}$$

Controls at the individual
and/or group level

Moderators at the individual
and/or group level

Linear -> Nonlinear models

Group-specific intercepts/
fixed/random effects

Group-specific “unexplained”
heterogeneity

$$y_{ij} = \alpha_j + \beta(\mathbf{x}_{ij}) + [\tau(\mathbf{w}_{ij}) + \phi_j]\, z_{ij} + \epsilon_{ij}$$

Controls at the individual
and/or group level

Moderators at the individual
and/or group level

Multilevel Bayesian Causal Forests

$$y_{ij} = \alpha_j + \beta(\mathbf{x}_{ij}) + [\tau(\mathbf{w}_{ij}) + \phi_j]\, z_{ij} + \epsilon_{ij}$$

Functions get BART priors

Standard priors for other model parameters

First implemented for analysis in Yeager et. al. (2019)

But user-friendly, (relatively) efficient public software has lagged!

Let’s see a demo!

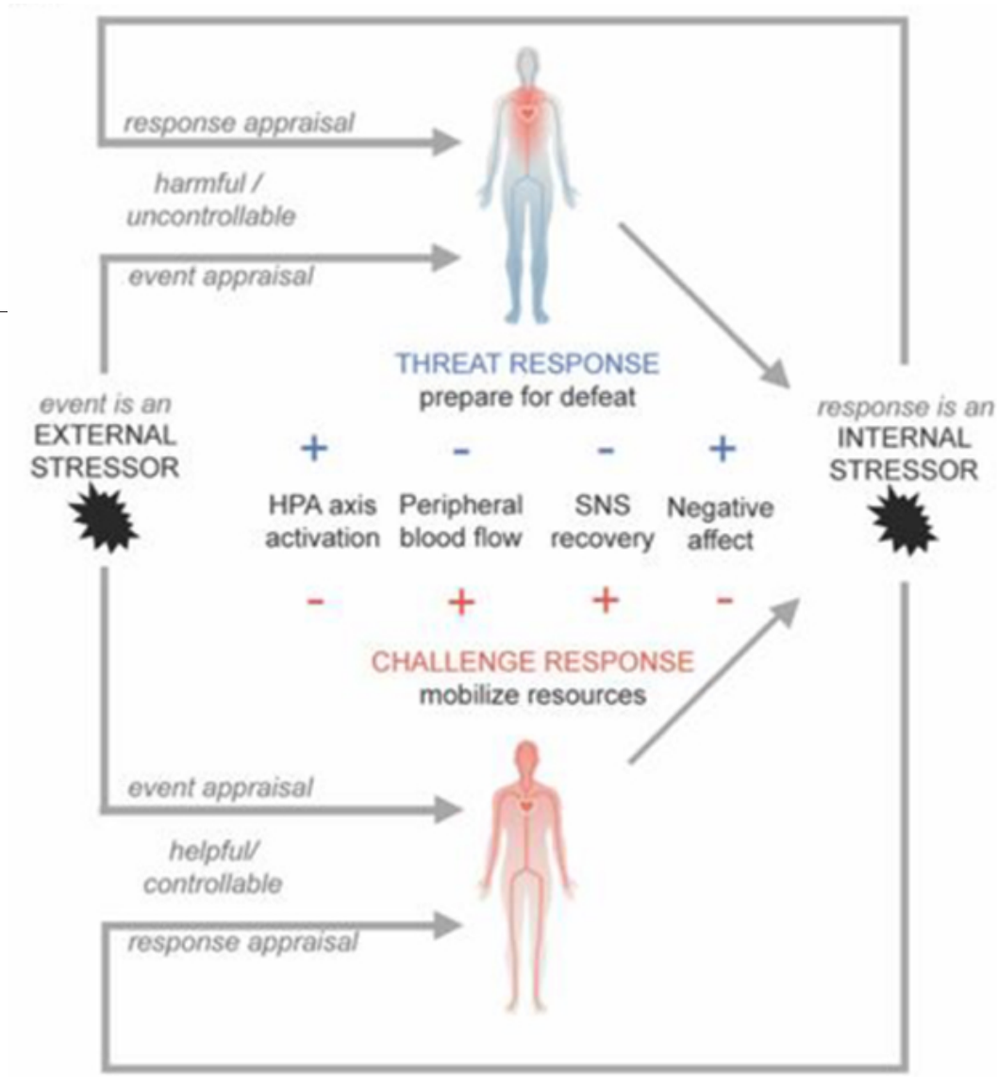
https://bit.ly/acic_bcf

Article

A Synergistic Mindsets Intervention Protects Adolescents from Social-Evaluative Stress

David S. Yeager[1*], Christopher J. Bryan[1*], James J. Gross[2], Jared Murray[1], Danielle Krettek[3], Pedro Santos[1], Hannah Graveling[4], Meghann Johnson[1], Jeremy P. Jamieson[4*]

Studies (Sample Size)	Population	Stressor	Measures of Threat-Type Stress Response
1 ($N = 2,717$)	13-18 y/o U.S. public school students during the COVID-19 pandemic.	Anticipated timed assignment	Event- and response-focused appraisals
2 ($N = 755$)	Diverse undergraduate students attending a public university	Experienced timed assignment	Cognitive appraisals at 1-3 days and 3 weeks post-test
3 and 4 (3: $N = 160$; 4: $N = 200$)	Undergraduate students at a private university	Trier Social Stress Test	Peripheral blood flow
5 ($N = 118$, n=1213 observations)	14-16 y/o adolescents from racial/ethnic minority groups, facing economic disadvantages	Daily stressors in high school	Daily negative self-regard and HPA-axis activation
6 ($N = 341$)	Same as study 2 but during the onset of the COVID-19 pandemic in Spring 2020	Ongoing academic demands during COVID-19 quarantines	Generalized internalizing symptoms



The Synergistic Mindsets Intervention

Growth mindset

e.g. Yeager & Dweck (2012)

+

Stress-can-be-enhancing mindset

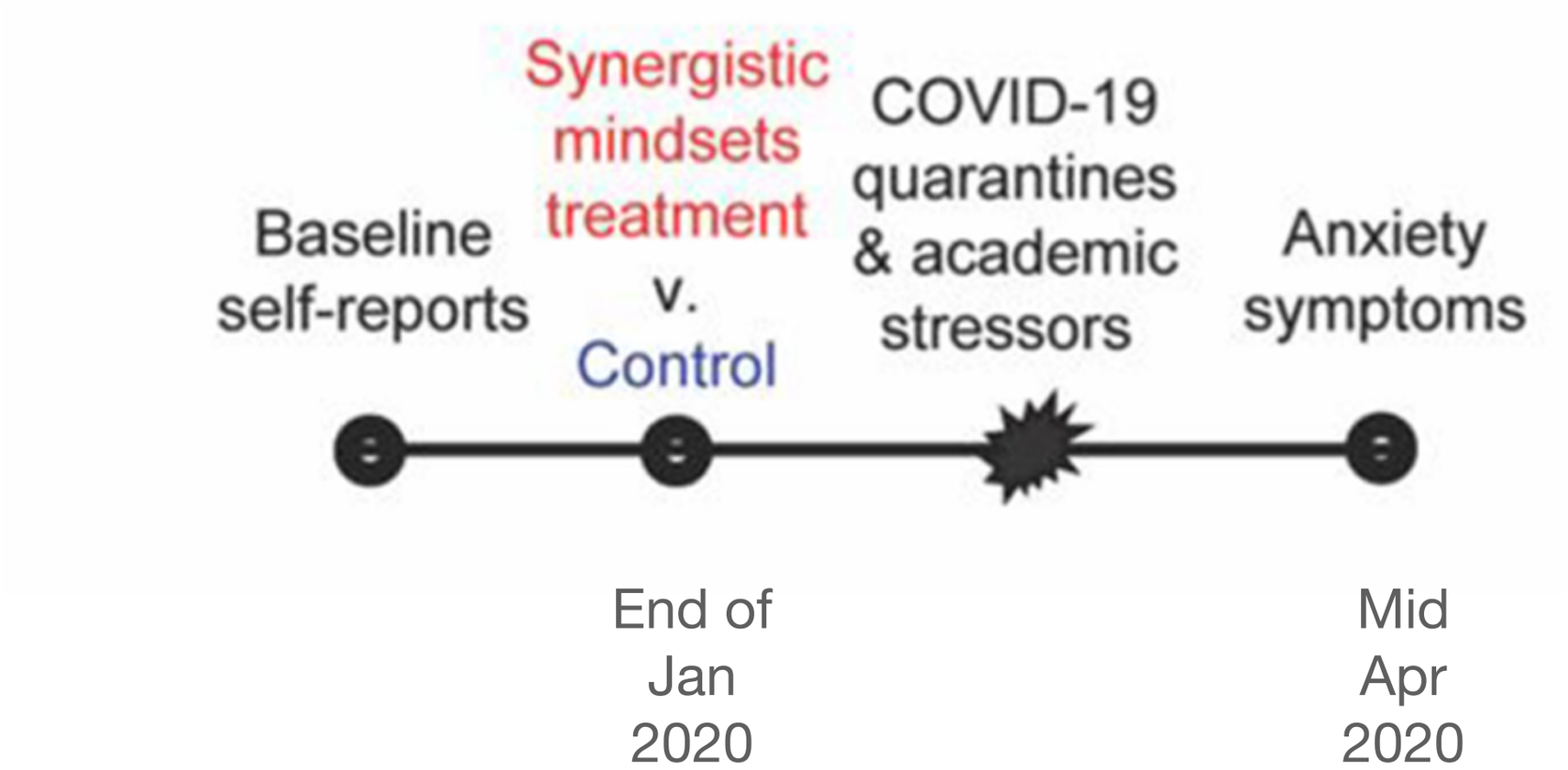
e.g. Crum et al. (2013)

- *Your abilities can be developed;*
- Understanding that helps you see struggles as a path to improvement (not a sign of inability).

- *Your stress responses can be useful for enhancing performance;*
- Understanding that allows you to harness the stress responses that inevitably follow from challenging goal-pursuit.

Expected intervention effects would be moderated by pre-intervention mindsets (largest effects in poor-mindset students) and the strength of the stressor

Study 6: Anxiety symptoms post-COVID-19

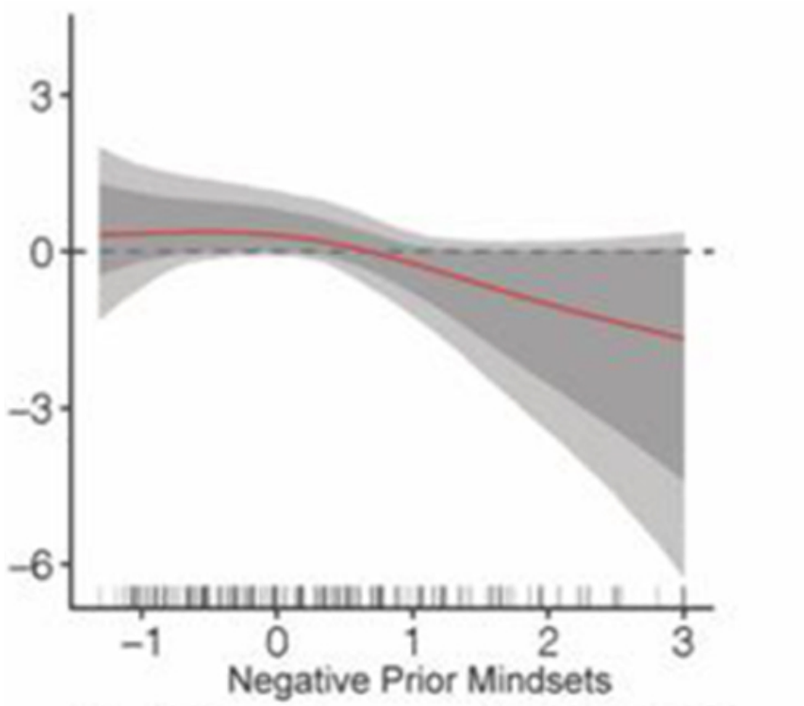


Interpreting and Presenting BCF Fits

Two (incompatible) descriptions of heterogeneity

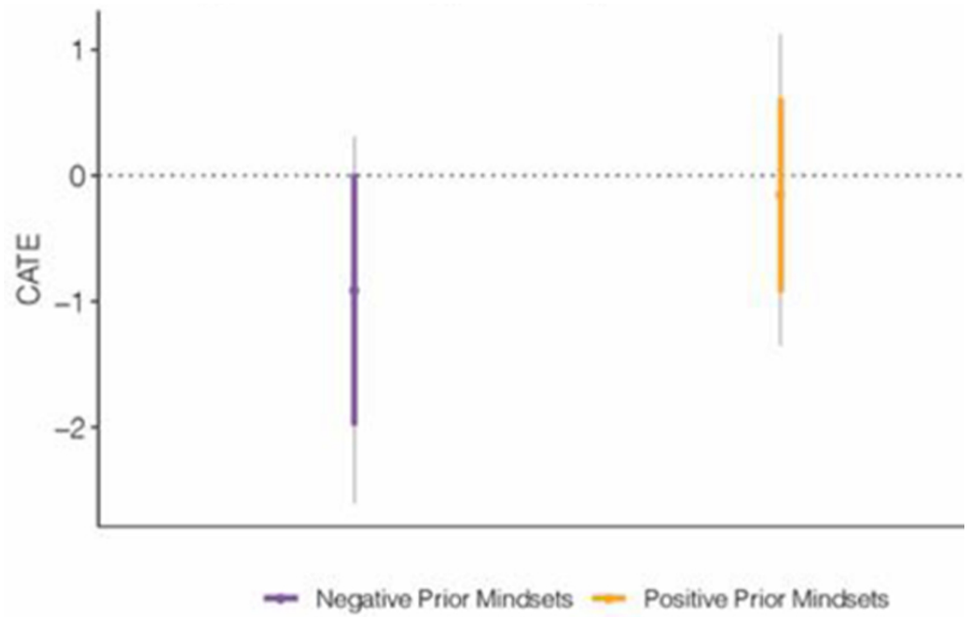
Partial effect of NPM on Trt Effect

(Adjusting for sex)



Prior Mindset index

Subgroup ATEs

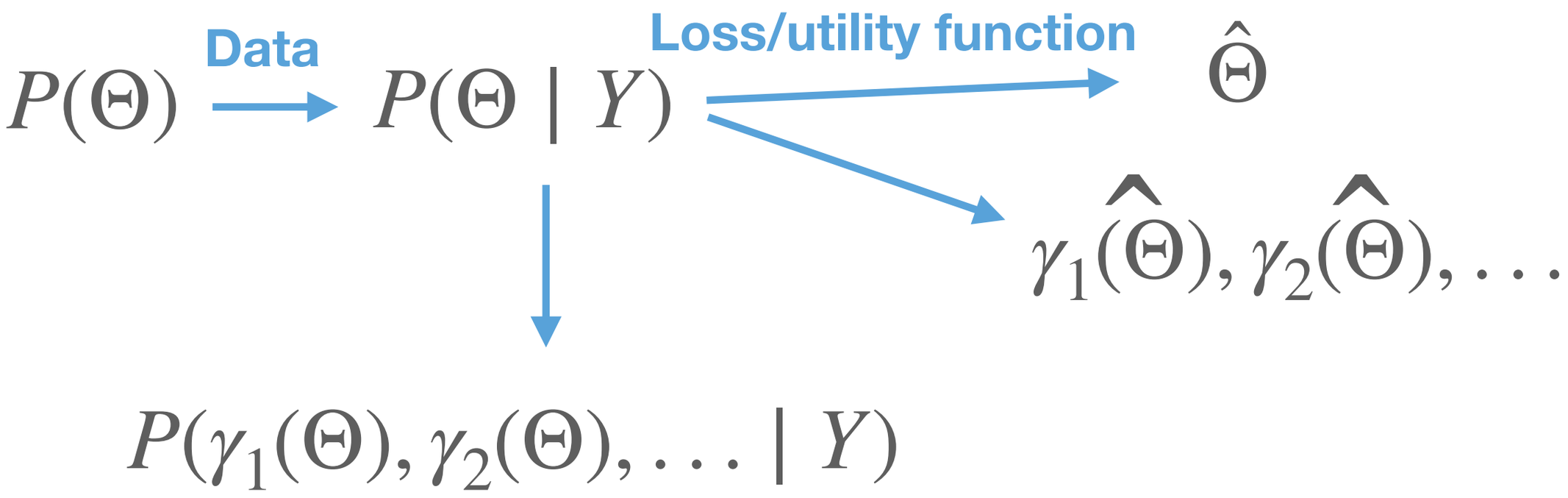


Subgroups

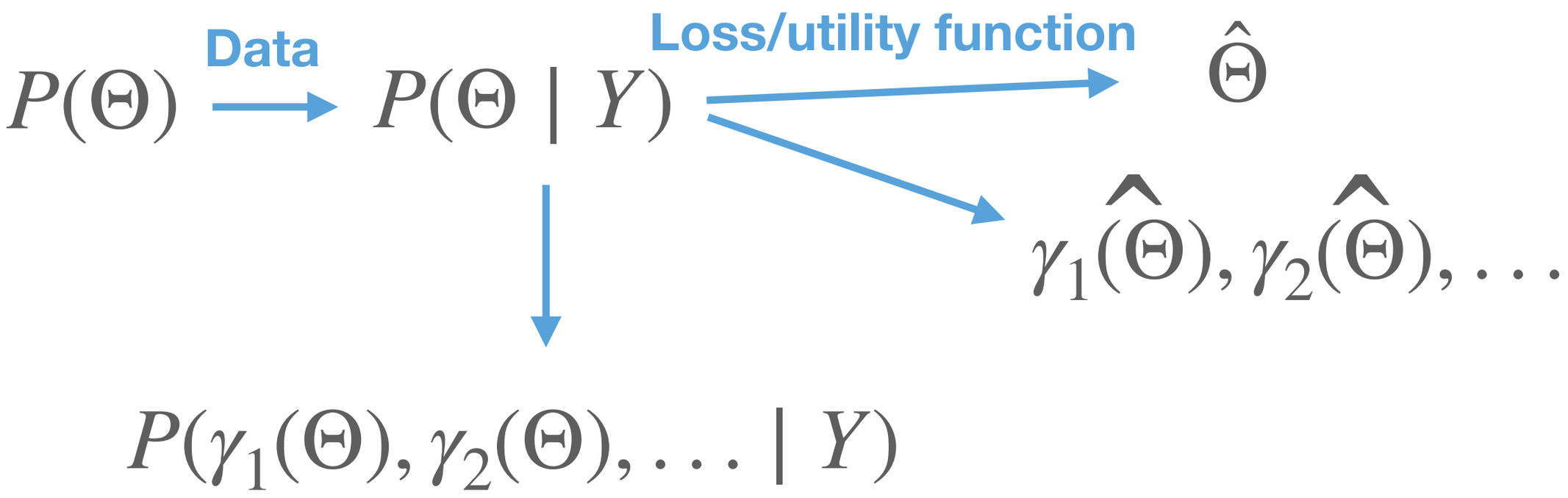
The Bayesian workflow

$$P(\Theta) \xrightarrow{\text{Data}} P(\Theta \mid Y)$$

The Bayesian workflow



The Bayesian workflow



Posterior summarization: **Approximate a function $g(x) \in \Theta$ by a simpler function $\gamma(x)$**

Posterior Summarization

Woody, Carvalho, and Murray (2020)

Given samples of a function $g(\mathbf{x})$,

1. Consider a class of simple/interpretable approximations Γ to g
2. Make inference on

$$\gamma = \underset{\tilde{\gamma} \in \Gamma}{\arg\min}\, d(g, \tilde{\gamma}, \tilde{\mathbf{X}}) + p(\tilde{\gamma})$$

for an appropriate distance function d and (optional) complexity penalty $p(\gamma)$

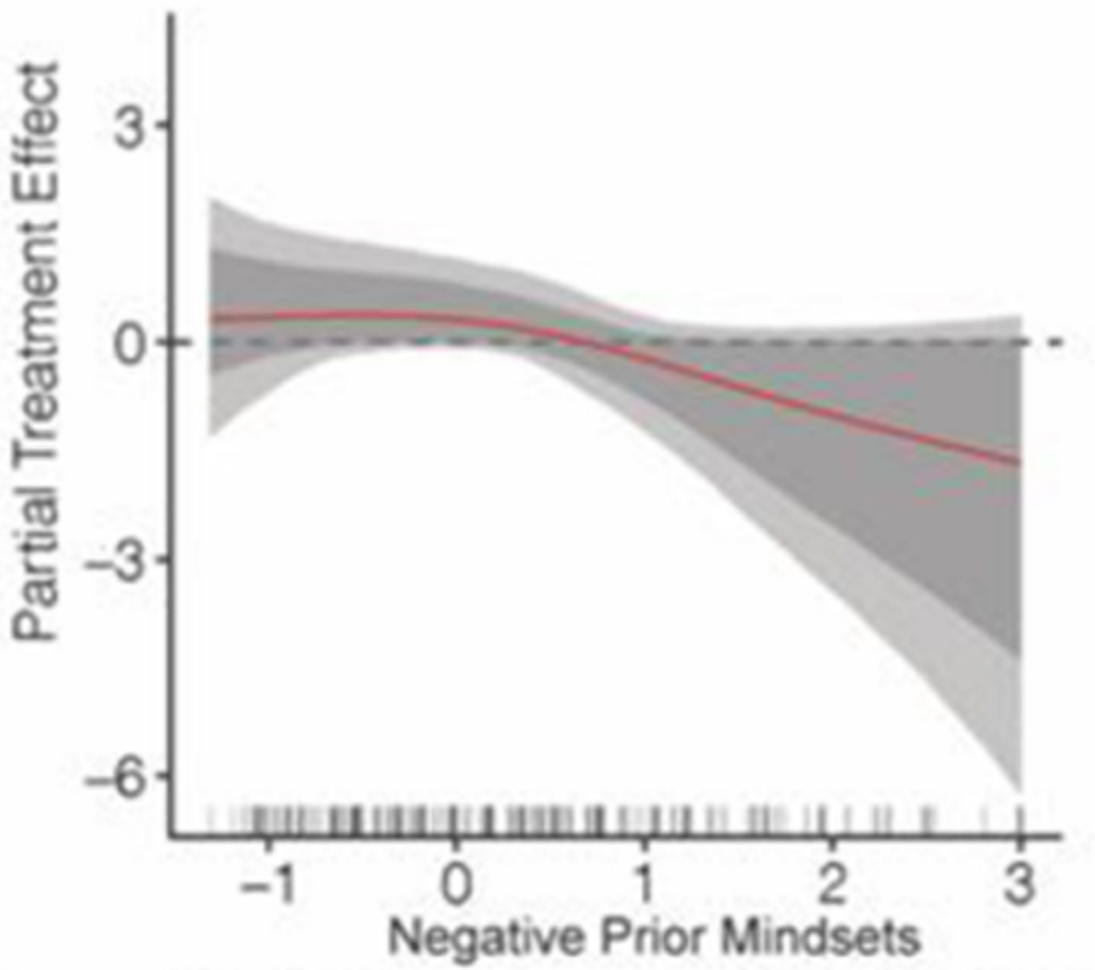
Get *draws* of γ by solving the optimization for each draw of g. Get point estimates by solving

$$\hat{\gamma} = \underset{\tilde{\gamma} \in \Gamma}{\arg\min}\, E_g[d(g, \tilde{\gamma}, \tilde{\mathbf{X}}) + p(\tilde{\gamma}) \mid Y, \mathbf{x}]$$

- d is typically squared error
- $\tilde{X}$ is typically the sample design matrix, but could be e.g. pop. values
- "Complexity" could be number/size of subgroups, (lack of) smoothness, interactions,…
- Free to vary any of these post-hoc — the model fit is the model fit

Additive summaries

$$\tau(w) \approx \gamma(w) = \gamma_0 + \gamma_1(w_1) + \gamma_2(w_2) + \ldots + \gamma_p(w_p)$$



$\gamma_j(w_j)$ is approx partial effect of moderator j

(Posteriors of) diagnostics for free, e.g.

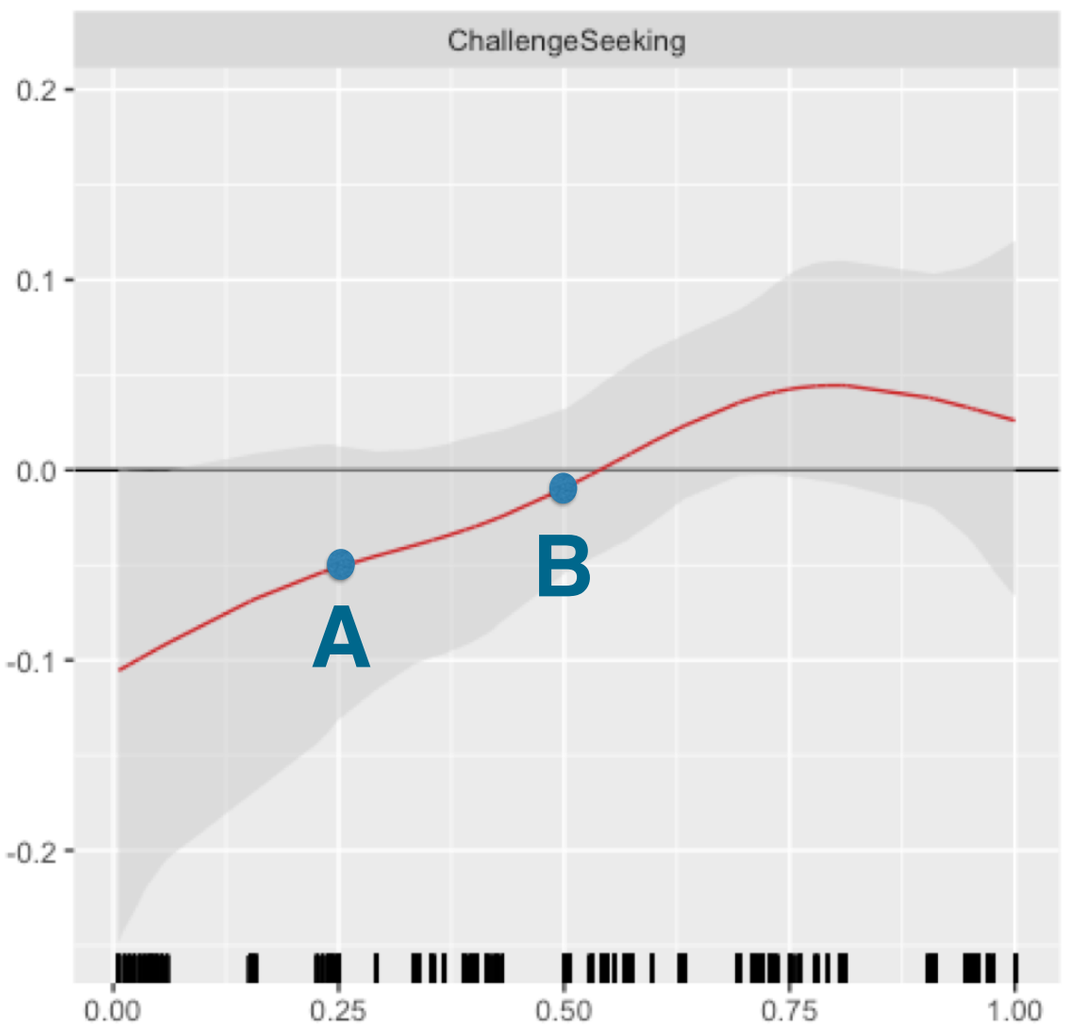
$$R^2_\gamma = \text{corr}[\tau(w), \gamma(w)]^2$$

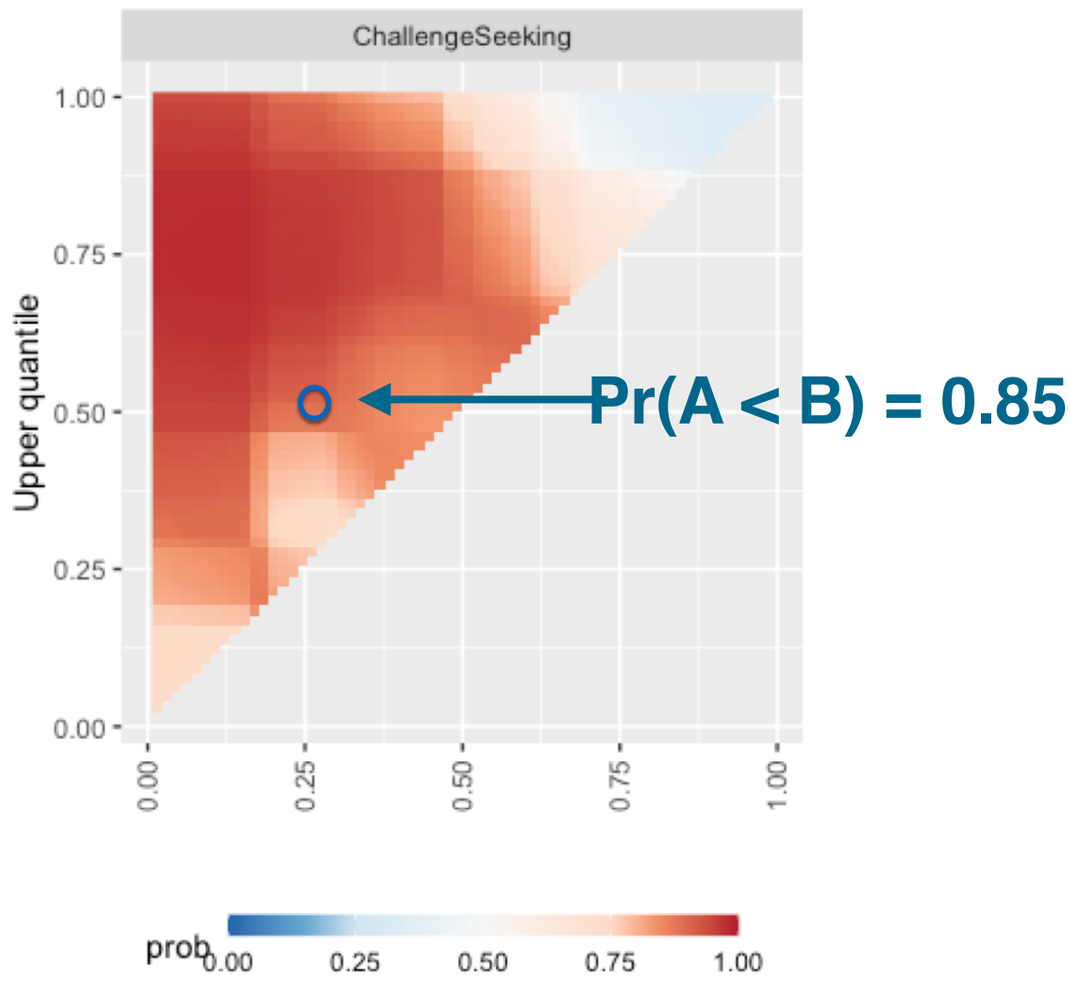
Diagnostics are informative!

(Maybe τ includes an interaction)

We can iterate, trying new summaries, without compromising (Bayesian) inference

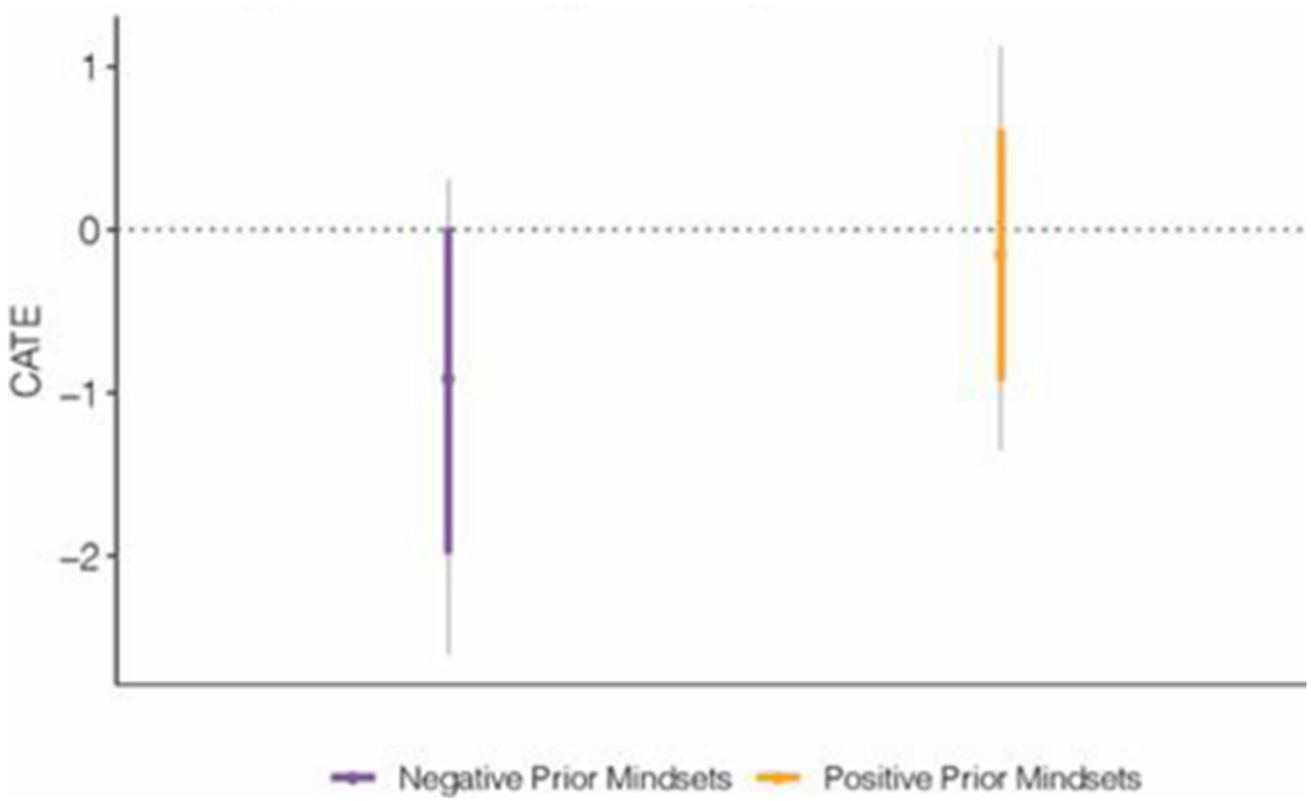
Uncertainty in additive summaries





Subgroup summaries

$$\tau(w) \approx \gamma(w) = \gamma_1(w)\mathbf{1}(w \text{ in neg}) + \gamma_2(w)\mathbf{1}(w \text{ in pos})$$



Choose subgroup definitions by maximizing a utility function

A hybrid Bayesian workflow

$$P(\Theta) \xrightarrow{\text{Data}} P(\Theta \mid Y)$$

A hybrid Bayesian workflow

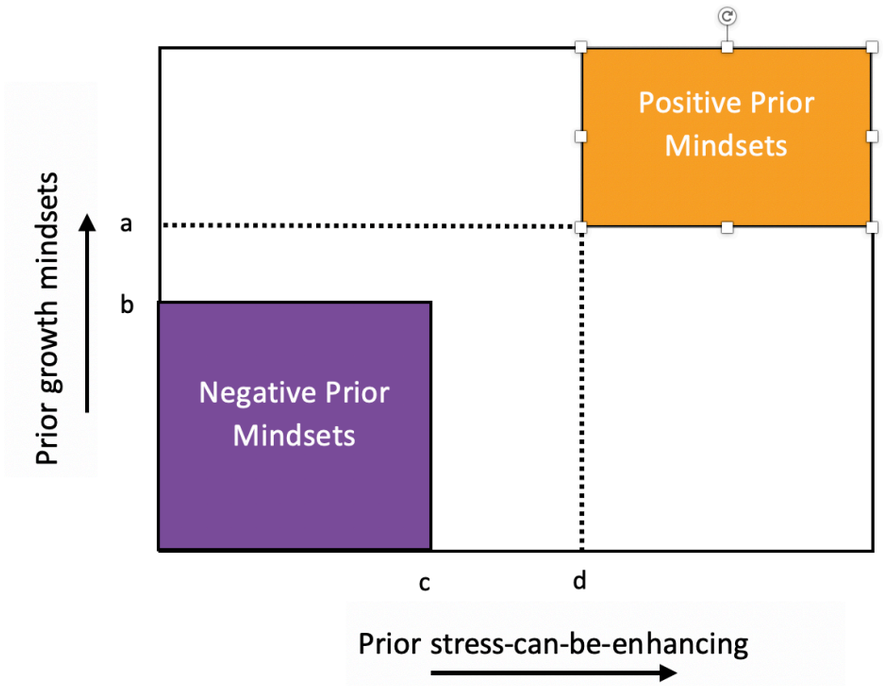
$$P(\Theta) \xrightarrow{\text{Data}} P(\Theta \mid Y) \xrightarrow{\text{Loss/utility function}} \Gamma$$ (e.g. selecting subgroups)

A hybrid Bayesian workflow

$P(\Theta)$ —Data→ $P(\Theta \mid Y)$ —Loss/utility function→ Γ

$$P(\gamma(\Theta), \ldots \mid Y)$$

The action in the decision analysis is choosing a family of posterior summaries

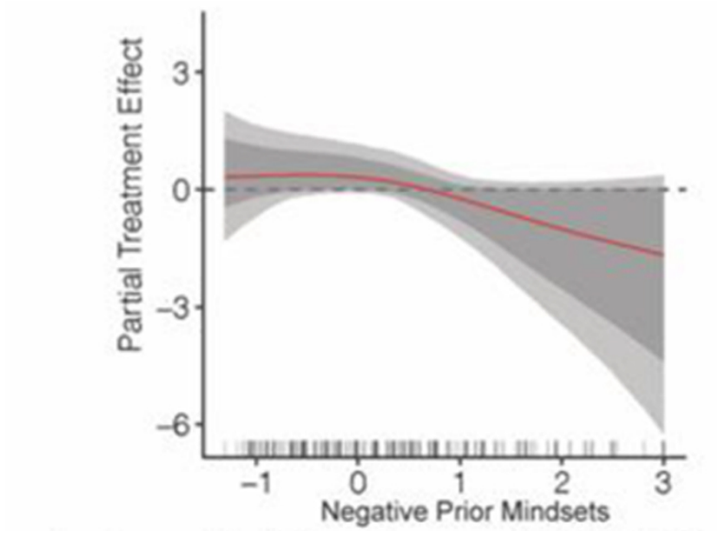


Summarizing posterior summarization

$$\tau(w) \approx \gamma(w) = \gamma_0 + \gamma_1(w_1) + \gamma_2(w_2) + \ldots + \gamma_p(w_p)$$

Summaries come from **one** model fit

The model didn't specify functional form of moderation by negative mindsets.



The subgroups were chosen algorithmically via decision analysis.

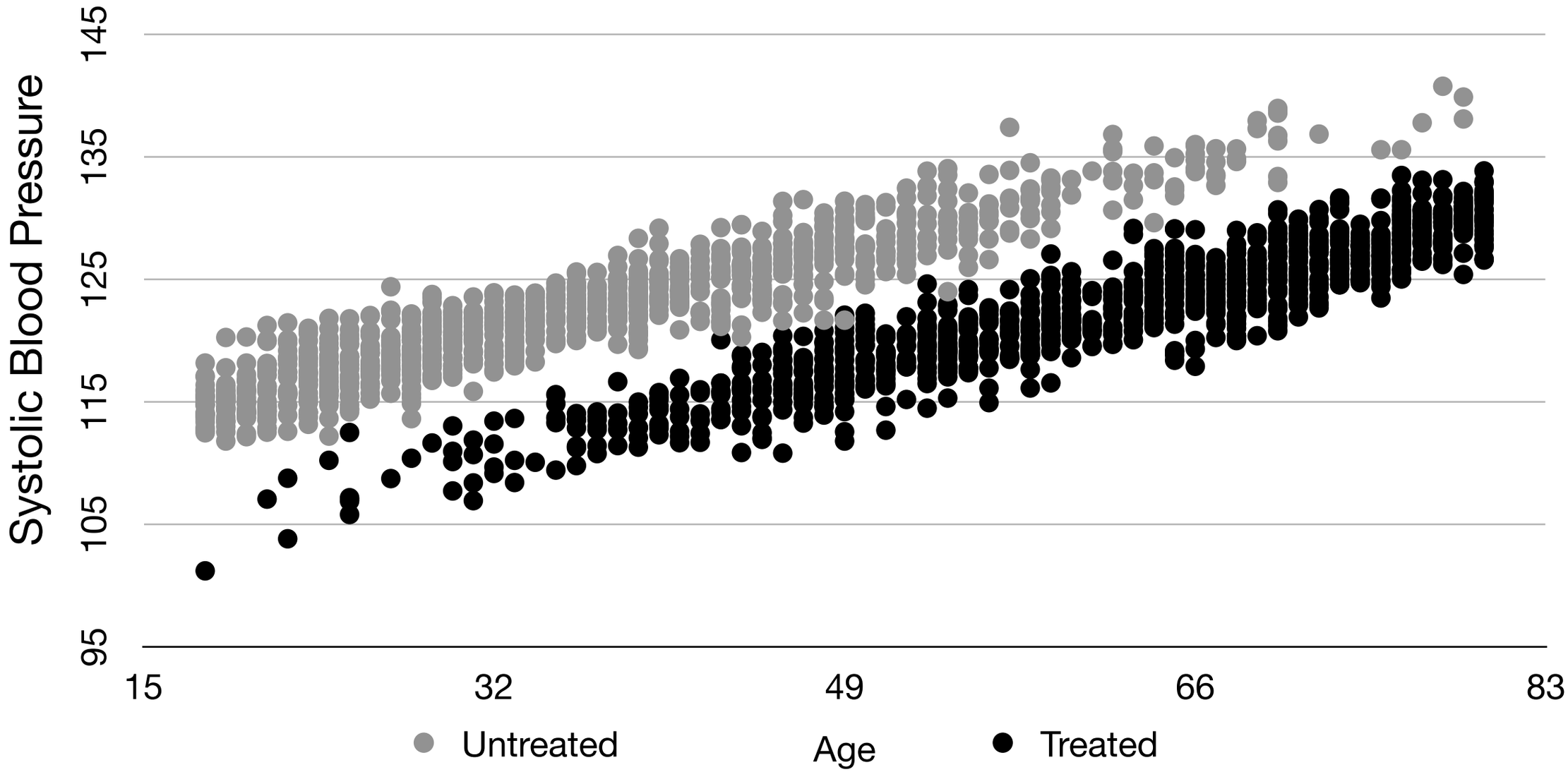
Fewer researcher d.o.f. without limiting model complexity or pre-reg dozens of models

$$\tau(w) \approx \gamma(w) = \gamma_1(w)\mathbf{1}(w \text{ in neg}) + \gamma_2(w)\mathbf{1}(w \text{ in pos})$$

BCF in Observational Studies

Why selection matters in observational data

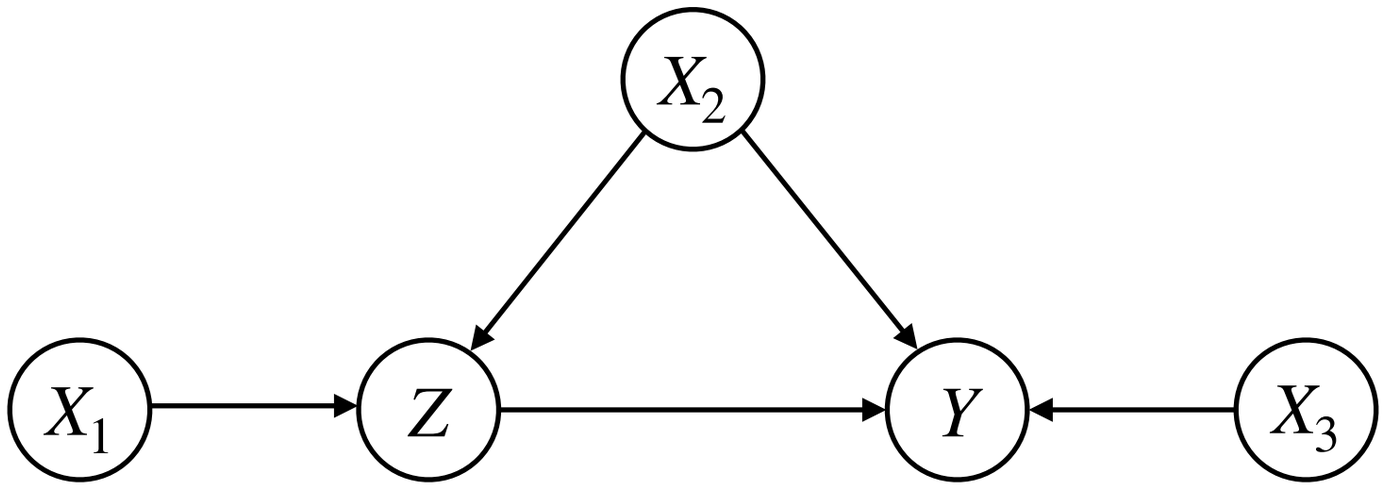
A stylized motivating example



Treated and untreated group means both 122.5…

… but treatment is clearly effective if we adjust for age

Selection problems = confounded estimates



Consider this model…

Condition on the data “as observed” and split into $Z = 1$ and $Z = 0$

$$\mathbb{E}\left[Y \mid Z = 1\right] = \mathbb{E}\left[F_y(Z, X_2, X_3, \epsilon_y) \mid Z = 1\right]$$

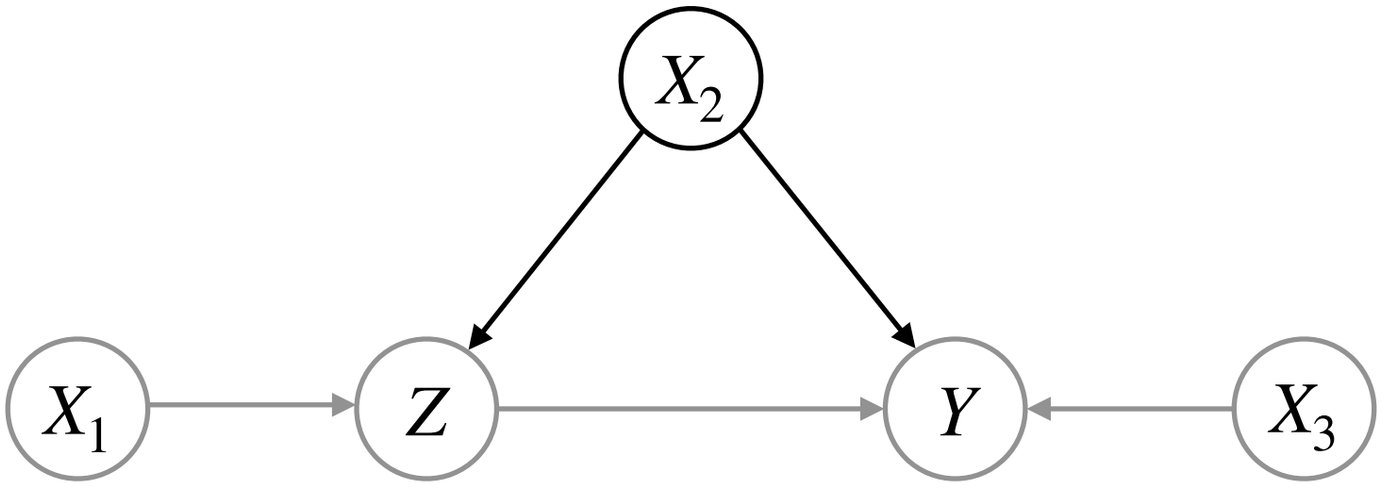
$$\mathbb{E}\left[Y \mid Z = 0\right] = \mathbb{E}\left[F_y(Z, X_2, X_3, \epsilon_y) \mid Z = 0\right]$$

Averaging over two different distributions!

$$X_2 \mid Z = 1$$

$$X_2 \mid Z = 0$$

Addressed by controlling for confounders



X_2 is a "confounding" variable — we must adjust for it

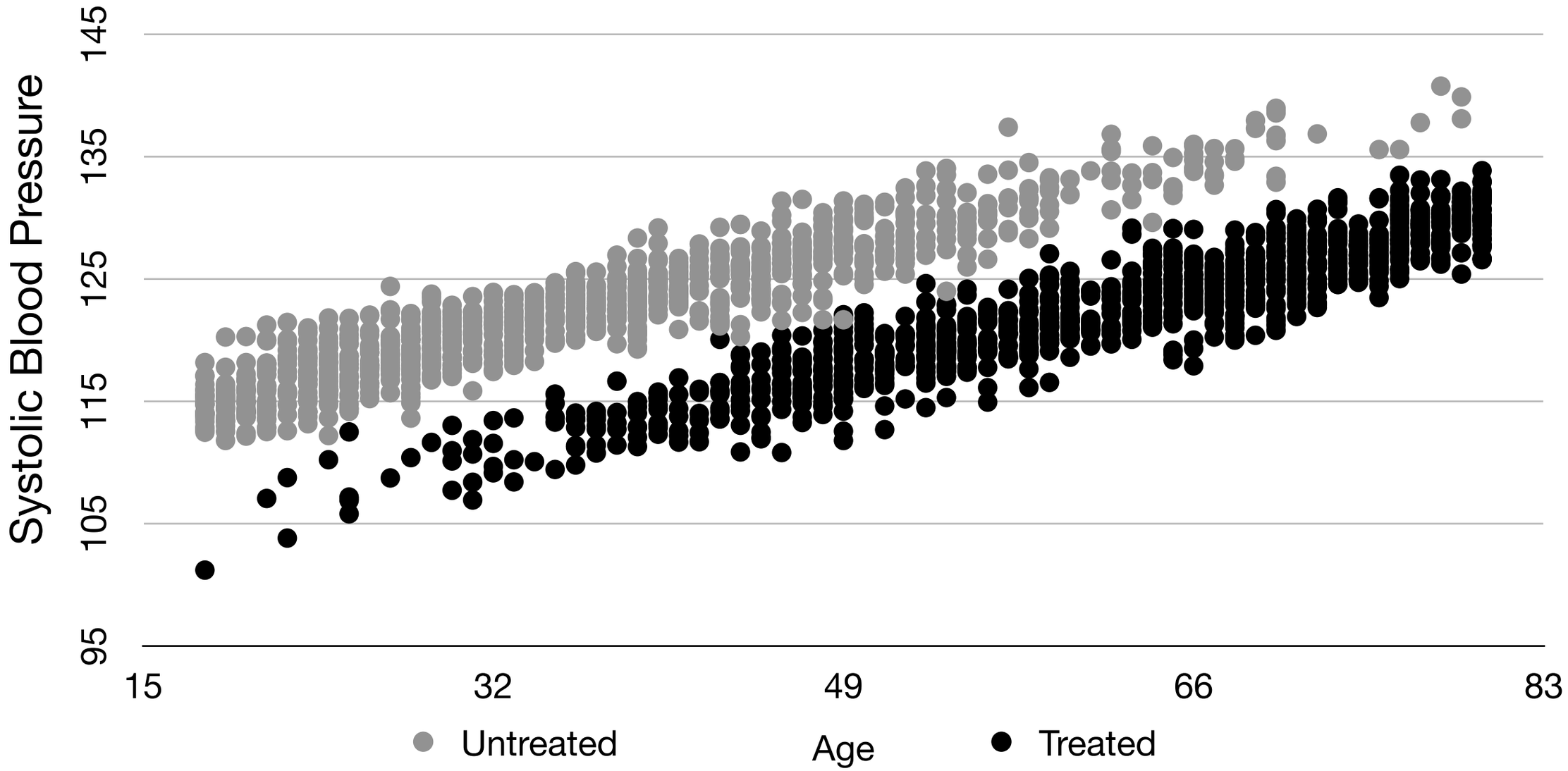
Solution: compute the same contrast conditional on X_2, then average

$$\mathbb{E}_{X_2}\left[\mathbb{E}\left[Y \mid X_2, Z=1\right] - \mathbb{E}\left[Y \mid X_2, Z=0\right]\right]$$

Several ways to do this:

- Outcome modeling
- IPW
- Both (AIPW/ One-step corrections/TMLE…)

Why selection matters in (causal) outcome models



Outcome modeling with OLS will do just fine here; reality usually demands more complex models. Complexity = regularization = bias

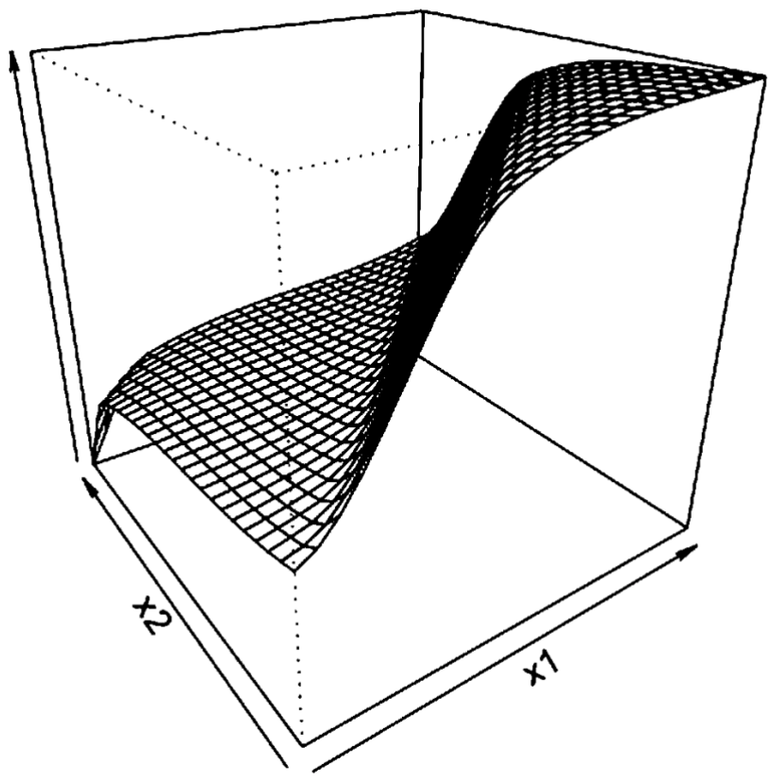
Why selection matters in (causal) outcome models

Regularization bias

$$\mathbb{E}_X\left[\mathbb{E}\left[Y \mid X, Z=1\right] - \mathbb{E}\left[Y \mid X, Z=0\right]\right] \approx \frac{1}{n}\sum_{i=1}^{n} \hat{f}(X,1) - \hat{f}(X,0)$$

It's well-known that plug-in estimates from regularized outcome models can be badly biased.

$\mu(x_1, x_2)$



$$Y_i = \mu(x_1, x_2) - \tau Z_i + \epsilon_i$$

$$\epsilon_i \overset{\text{iid}}{\sim} \text{N}(0,1), \quad x_{i1}, x_{i2} \overset{\text{iid}}{\sim} \text{Uniform}(0,1).$$

Prior	bias	coverage	rmse
BART	0.27	65%	0.31

(Hahn, Murray and Carvalho (2020))

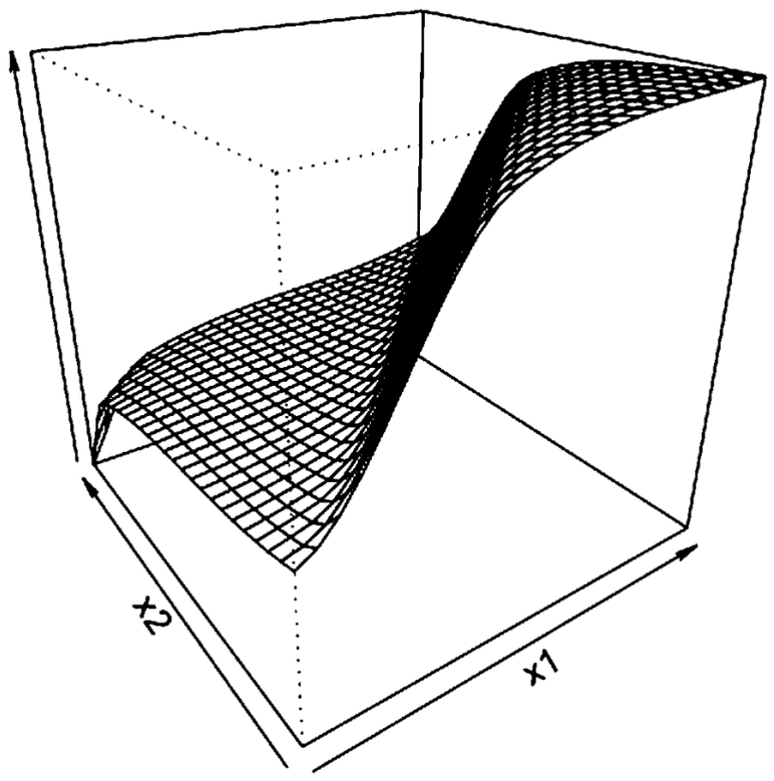
Why selection matters in (causal) outcome models

Regularization bias

$$\mathbb{E}_X\left[\mathbb{E}\left[Y \mid X, Z=1\right] - \mathbb{E}\left[Y \mid X, Z=0\right]\right] \approx \frac{1}{n}\sum_{i=1}^{n} \hat{f}(X,1) - \hat{f}(X,0)$$

Simple adjustments to account for selection can often mitigate the bias.

$\mu(x_1, x_2)$



$$Y_i = \mu(x_1, x_2) - \tau Z_i + \epsilon_i$$

$$\epsilon_i \overset{\text{iid}}{\sim} \text{N}(0,1), \quad x_{i1}, x_{i2} \overset{\text{iid}}{\sim} \text{Uniform}(0,1).$$

Prior	bias	coverage	rmse
BART	0.27	65%	0.31
BCF	0.14	95%	0.21

(Hahn, Murray and Carvalho (2020))

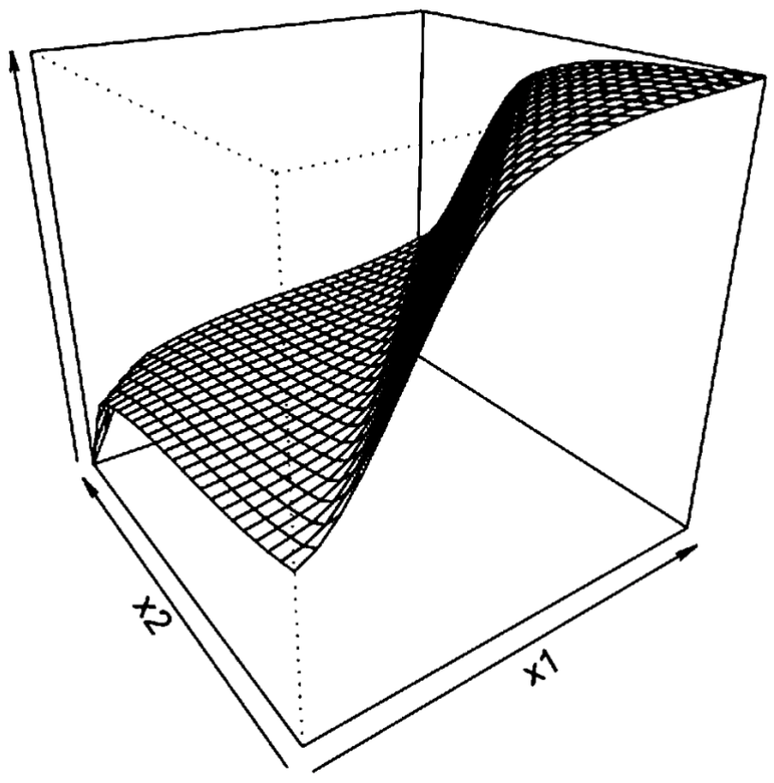
Why selection matters in (causal) outcome models

Regularization bias

$$\mathbb{E}_X\left[\mathbb{E}\left[Y \mid X, Z=1\right] - \mathbb{E}\left[Y \mid X, Z=0\right]\right] \approx \frac{1}{n}\sum_{i=1}^{n} \hat{f}(X,1) - \hat{f}(X,0)$$

These adjustments are useful ***with or without*** other debiasing techniques (AIPW/DoubleML/One-Step/TMLE/…)

$\mu(x_1, x_2)$



$$Y_i = \mu(x_1, x_2) - \tau Z_i + \epsilon_i$$

$$\epsilon_i \overset{\text{iid}}{\sim} \text{N}(0,1), \quad x_{i1}, x_{i2} \overset{\text{iid}}{\sim} \text{Uniform}(0,1).$$

Prior	bias	coverage	rmse
BART	0.27	65%	0.31
BCF	0.14	95%	0.21

(Hahn, Murray and Carvalho (2020))

Adjustments to mitigate regularization bias

Adjustment

1. Include estimated propensity score $\hat{e}(x)$ as a covariate
2. Regress on "residuals" $z - \hat{e}(x)$ instead of z
3. Add a linear term in the "clever covariate", e.g. $\dfrac{z - \hat{e}(x)}{\hat{e}(x)(1 - \hat{e}(x))}$ for the ATE

Adjustments to mitigate regularization bias

Adjustment

1. **Include estimated propensity score $\hat{e}(x)$ as a covariate**
2. Regress on "residuals" $z - \hat{e}(x)$ instead of z
3. Add a linear term in the "clever covariate", e.g. $\dfrac{z - \hat{e}(x)}{\hat{e}(x)(1 - \hat{e}(x))}$ for the ATE

Comments

- For BCF, include in μ,τ, or both
- Simple, low-cost, easy to do with existing software
- Free of outcomes — just like any other covariate transformation
- Valid "Zellner" prior for the outcome model

Adjustments to mitigate regularization bias

Adjustment

1. Include estimated propensity score $\hat{e}(x)$ as a covariate
2. **Regress on "residuals"** $z - \hat{e}(x)$ **instead of** z
3. Add a linear term in the "clever covariate", e.g. $\frac{z - \hat{e}(x)}{\hat{e}(x)(1 - \hat{e}(x))}$ for the ATE

Comments

- Pro: With Gaussian likelihood, you get desirable orthogonality properties
- Con: Can be expensive in terms of variance (e.g. accidentally including instruments)
- Requires modification to software/methods (possible in stochtree!)
- Free of outcomes — just like any other covariate transformation
- Valid "Zellner" prior for the outcome model

Adjustments to mitigate regularization bias

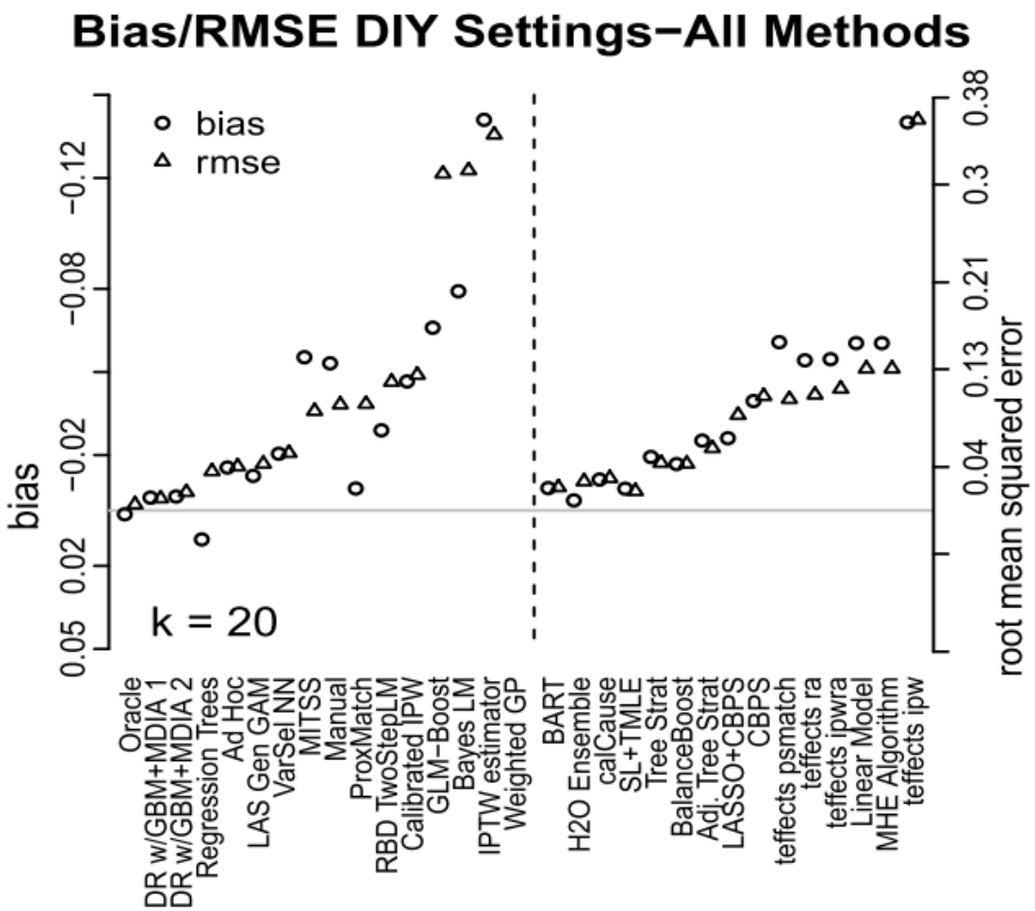
Adjustment

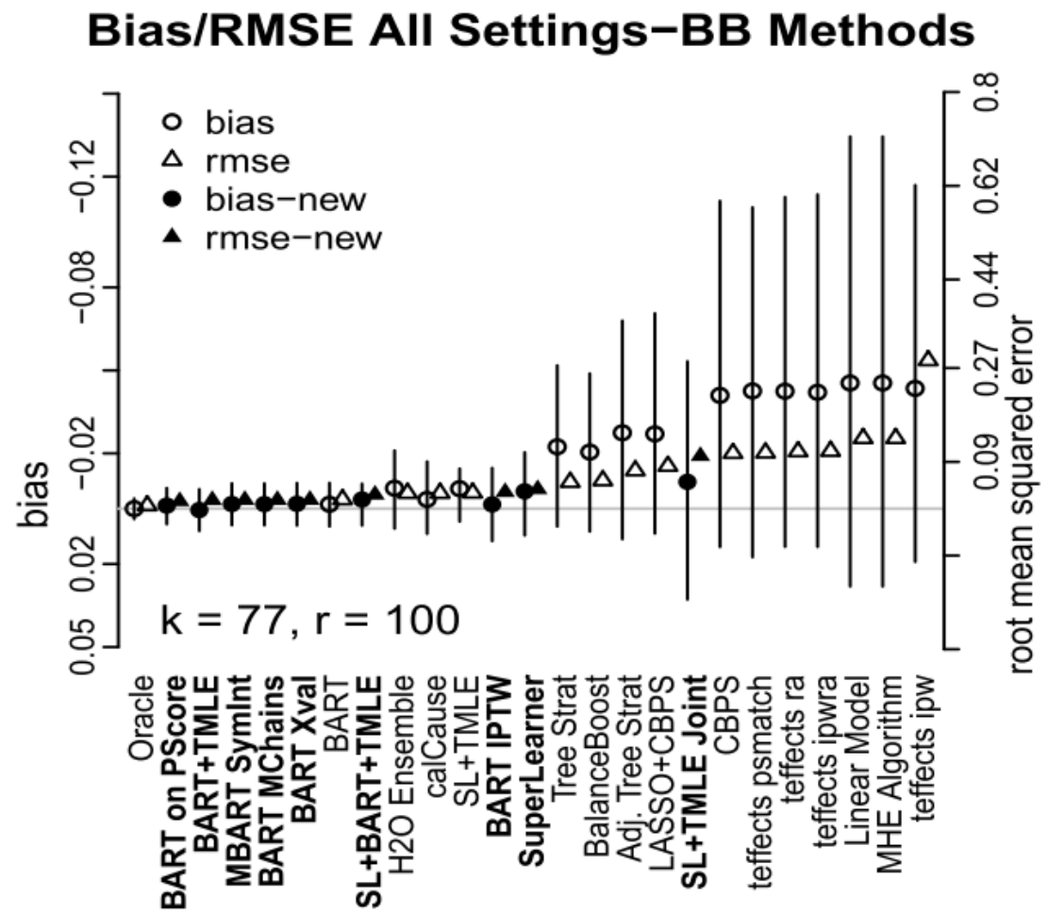
1. Include estimated propensity score $\hat{e}(x)$ as a covariate
2. Regress on "residuals" $z - \hat{e}(x)$ instead of z
3. **Add a linear term in the "clever covariate", e.g.** $\frac{z - \hat{e}(x)}{\hat{e}(x)(1 - \hat{e}(x))}$ **for the ATE**

Comments

- Adjusts only for a *single* target
  - Could make bias *worse* for other targets
- Calibrating the prior on the regression coefficient is difficult
- Forces effect modification by the propensity score
- Not recommended, but possible using stochtree

Adjustments are effective empirically





"Automated versus Do-It-Yourself Methods for Causal Inference: Lessons Learned from a Data Analysis Competition", Dorie et al. (2019)

Questions?

Appendix

Posterior summarization

Additive summaries

$$\tau(w) \approx \tau_0 + \gamma_1(w_1) + \gamma_2(w_2) + \ldots + \gamma_p(w_p)$$

Sample ATE

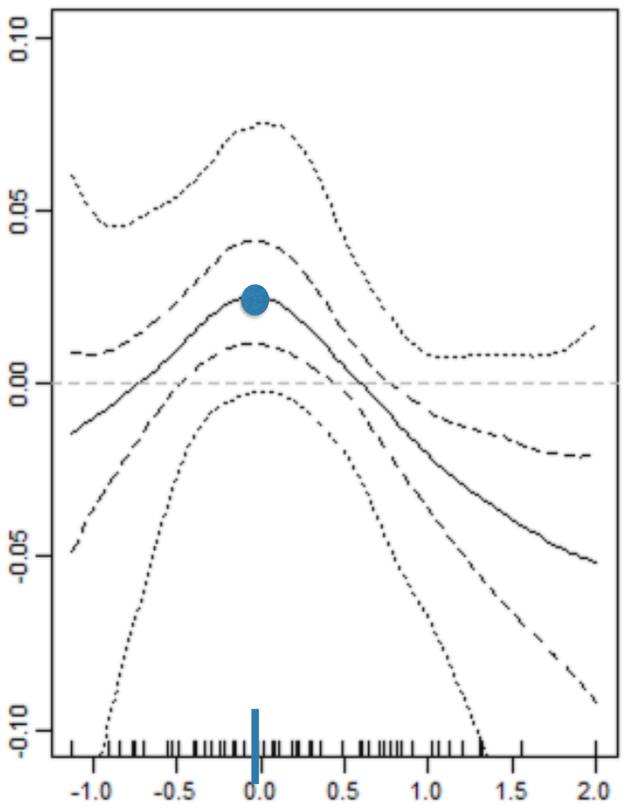
Offset due to each moderator

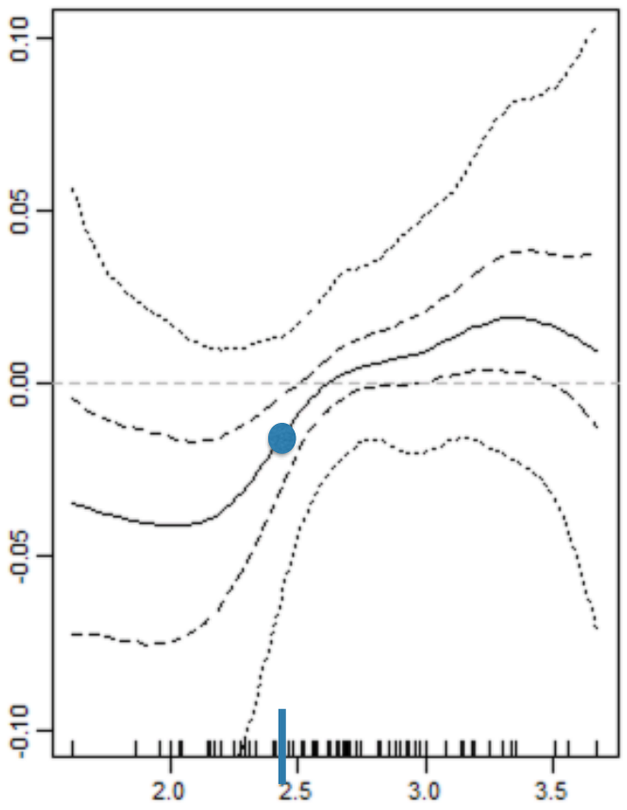
Achievement

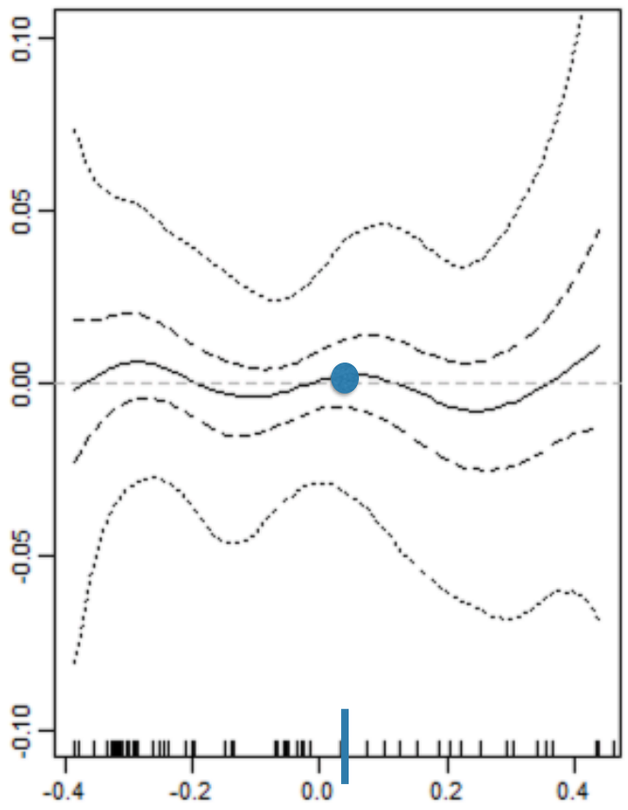
Challenge-Seeking Norms

% Stereotyped Minority (centered)

= 0.09+







0.095 = 0.09 + 0.025

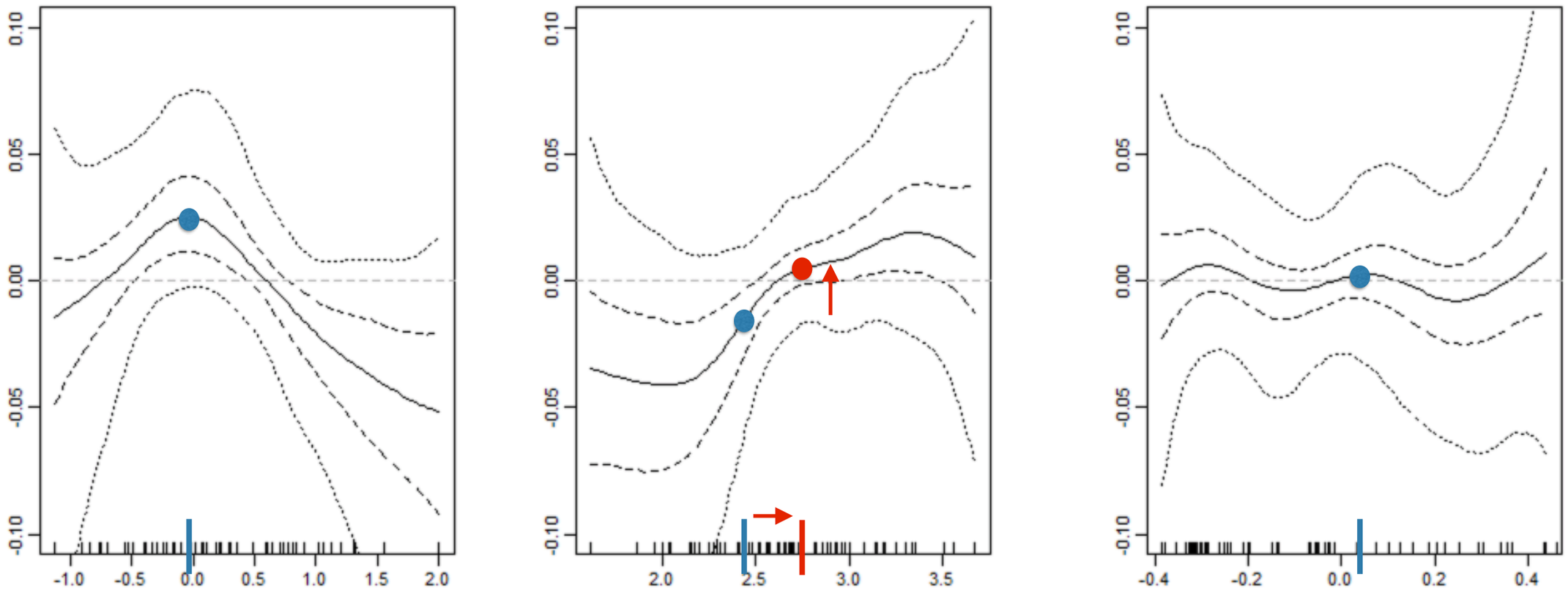
- 0.02

+ 0.0

Achievement

Challenge-Seeking Norms

% Stereotyped Minority (centered)



= 0.09+

0.095 = 0.09 + 0.025 **- 0.02** **+ 0.0**

0.12 = 0.09 + 0.025 **+ 0.005** **+ 0.0**

Adjusting for Ach & Pct Minority, this change in norms is associated with an increase of 0.005 - (-0.02) = 0.025 in the treatment effect

Reading Additive Summary Plots

- Remember: These represent offsets, and the offset from *what* is arbitrary. Don't worry about the sign.

- Look for *variability* and the *shape*
  - Low variability (flat line) means no effect of that moderator after adjusting for the others
  - The shape tells us what happens as we imagine one moderator varying while the others don't.