Macro Random Forest and Related Topics

Philippe Goulet Coulombe
Université du Québec à Montréal

2023 SIDE Summer School

Outline

1. Random Forest (RF) Crash Course
2. Macroeconomic RF
3. Why does RF always seem to avoid catastrophic overfitting failure?

I.

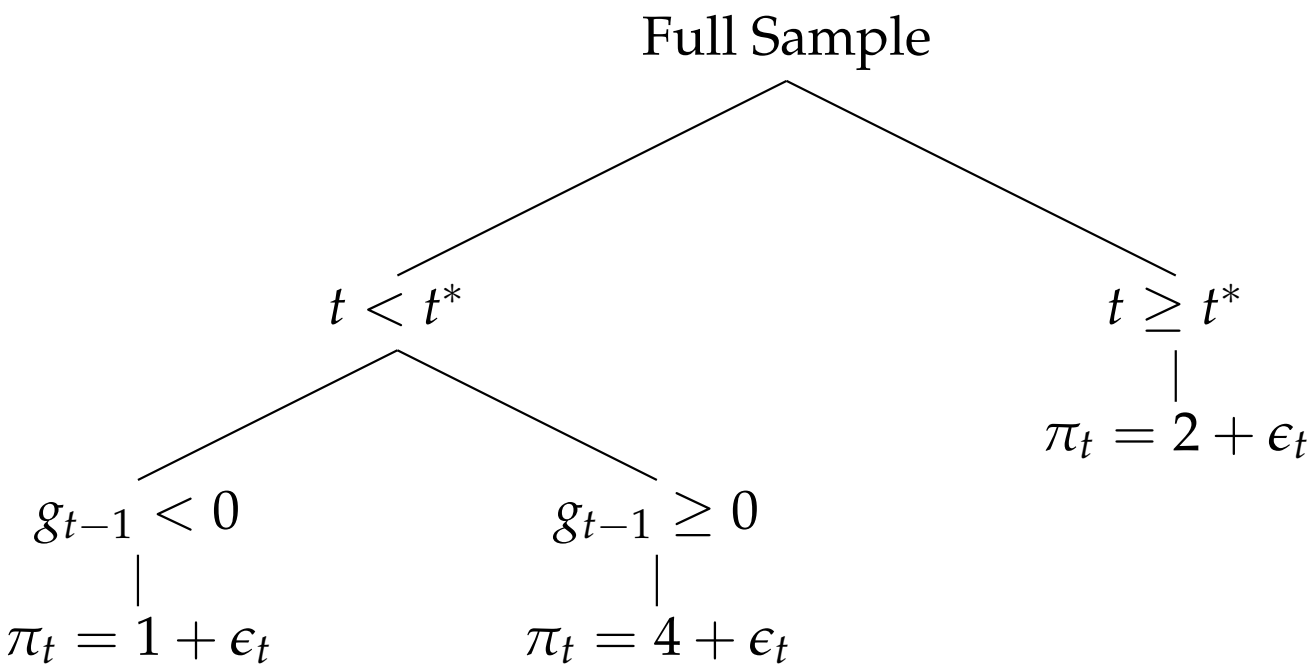
Random Forest Crash Course

Random Forest Crash Course

What is a tree?

RF is a diversified ensemble of regression trees. What is a tree?

- Let π_t be inflation at time t.
- t^* is inflation targeting implementation date.
- Let g_t be some measure of output gap.



$$y_i = \mathcal{T}(X_i) + \epsilon_i$$

- A regression tree is an algorithm that recursively partitions the data until some stopping criterion is met. A *greedy* algorithm is used:

$$\min_{k \in \mathcal{K}, c \in \mathbb{R}} \left[\min_{\mu_1} \sum_{\{i \in L | X_i^k \leq c\}} (y_i - \mu_1)^2 + \min_{\mu_2} \sum_{\{i \in L | X_i^k > c\}} (y_i - \mu_2)^2 \right] \quad (1)$$

- The prediction for j is the average of y_i for all i that are members of the same "leaf" as j.
- A single tree typically has low bias and very high variance.
- There exists ways to decrease tree's variance by "pruning", which means stopping the greedy algorithm "early".

RF Crash Course

3 ingredients to go from a single tree to a forest

For each tree:

1. **Let the trees run deep**: even though that would surely imply overfitting for a single tree, let each tree run until leafs contain very few observations (usually < 5).

Diversifying the Portfolio (i.e., creating the ensemble)

2. **Bagging**: Create B nonparametric bootstrap samples of the data. That is, we are picking $[y_i \;\; X_i]$ pairs with replacement.

3. **Perturbation**: At each splitting point, we only consider a subset of all predictors ($\mathcal{J}^- \subset \mathcal{J}$) for the split.

RF prediction is the simple average of all the B tree predictions.

RF Crash Course

Why do we like it

- It works tremendously well on all sorts of data, even macro data (Chen et al., 2019; Goulet Coulombe et al., 2022; Medeiros et al., 2021; Goulet Coulombe, 2020; Goulet Coulombe et al., 2021a).
- More often than not, it's better than Neural Networks for tabular data (Grinsztajn et al., 2022), and the latter requires careful tuning.
- Can approximate a wide range of nonlinearities
- Tuning parameters do not alter prediction much
- Can easily deal with a very large X (no matrix operation involved)
- **Most importantly, it does not seem to overfit.** We'll get back to that.

II.

Macroeconomic Random Forest

Final Destination

Modeling *flexibly* macro relationships without assuming what flexible means first. Take something fundamental: a Phillips' curve.

$$u_t^{\text{gap}} \rightarrow \pi_t$$

The statistical characterization of "$\rightarrow$" has forecasting, policy and theoretical (!) implications. Better get it right.

One way out is getting "$\rightarrow$" from off-the-shelf nonparametric Machine Learning (ML) techniques. **But:**

- Likely too flexible and wildly inefficient for the short *time series* we have.
- No obvious parameter(s) to look at — interpretation is fuzzy.

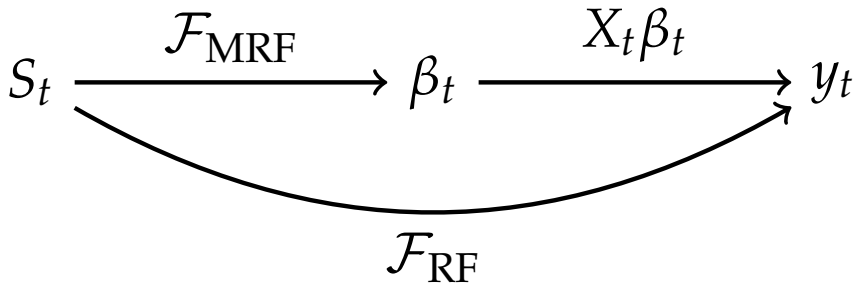
Another is assuming $\pi_t = \beta_t u_t^{\text{gap}} + \text{stuff}_t$. **But:**

- Rigid
- In-sample fit notoriously don't translate in out-of-sample gains.

Solution: ***Generalized*** **Time-Varying Parameters** via Random Forests.

(Machine) Learning β_t's

- I propose *Macroeconomic Random Forests* (MRF): fix the linear part X_t and let the coefficients β_t vary trough time according to a Random Forest.



- The core "mechanical" modification wrt plain RF is fitting an ensemble of trees which have a linear model in each leaf rather than a constant.

- MRF is nice "meeting halfway"
  - $\Rightarrow$ Brings macro closer to ML by squashing many popular nonlinearities (structural change/breaks, thresholds, regime-switching, etc.) into an arbitrarily large S_t, handled easily by RF.
  - $\Rightarrow$ The core output are β_t's, *Generalized Time-Varying Parameters* (GTVPs).
  - $\Leftarrow$ Brings ML closer to macro by adapting RF to the reality of economic time series. MRF $\succ$ RF if the linear part is pervasive (like in a (V)AR).

Philosophical Detour

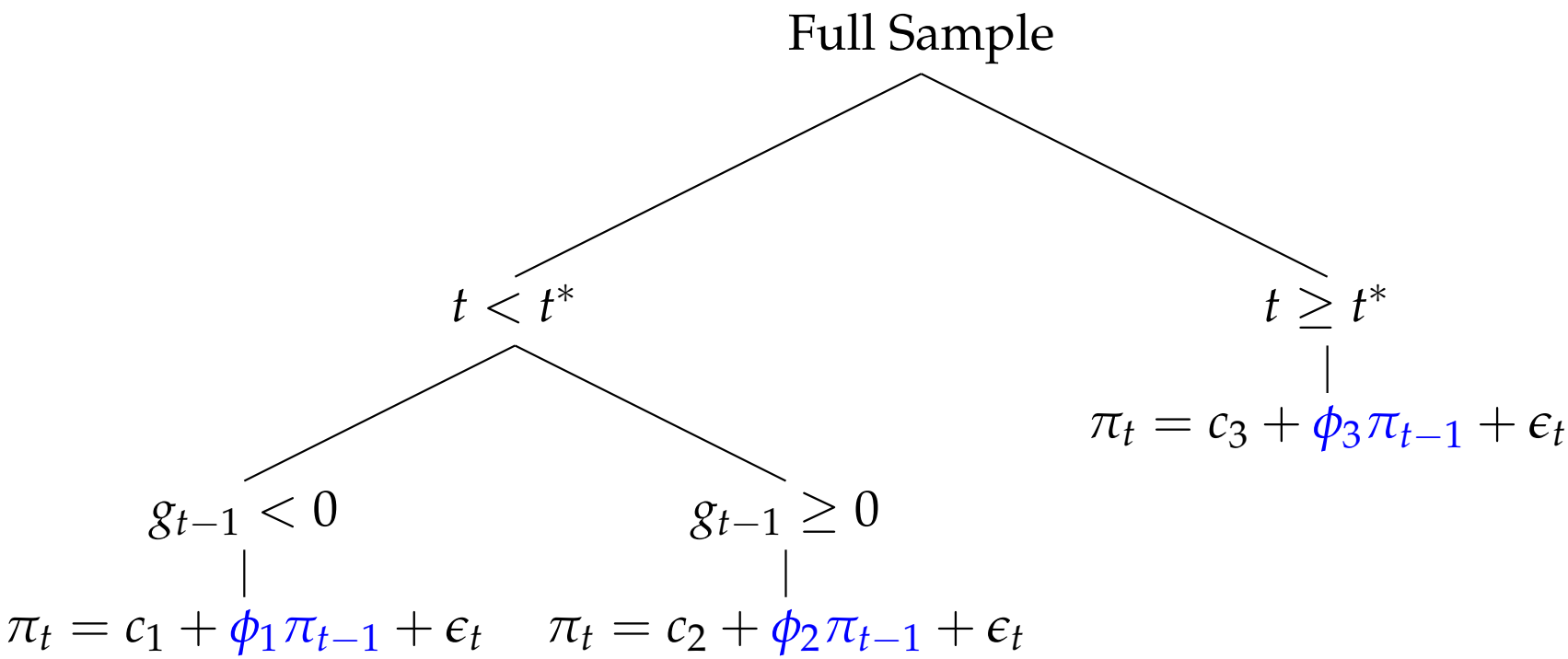
Why Inherently Interpretable Models?

- Because post-estimation black box interpretation is a far cry from the kind of interpretability provided by small linear econometric models
- Nonetheless, Shapley Values can go a long way, especially in low-dimensional configurations (Buckmann et al., 2022; Borup et al., 2022).
- Thus, the problem is not necessarily nonlinearity and interaction terms, but *high-dimensionality*
- Even a linear SVAR with 100 beautifully orthogonalized shocks is practically meaningless (what is the meaning of changing $X_{t,k}$ keeping fixed 99 other things that usually co-move together?)
- Going for statistical factors, through PCA or DFMs, certainly reduces dimensionality but frames everything in a mostly uninterpretable latent space
- It appears there is more hope in choosing a few relevant things (through a linear part in MRF) and letting the high-dimensional nonlinear parts patch the holes in the fit

Generalized Time-Varying Parameters

Why Trees Make Sense (in Macro/Finance)

- Let π_t be inflation at time t.
- t^* is inflation targeting implementation date.
- Let g_t be some measure of output gap.



Generalized Time-Varying Parameters

- The general model is

$$y_t = X_t\beta_t + \epsilon_t$$
$$\beta_t = \mathcal{F}(S_t)$$

  where S_t are the state variables that determine time-variation.
- If we know the threshold variables ($S_t = [t,\ g_{t-1}]$) and values ($\boldsymbol{c} = [t^*,\ 0]$): run OLS on subsamples.
- **But we don't.** So we need an algorithm to find out:

$$\min_{j\in\mathcal{J}^-,\ c\in\mathbb{R}}\left[\min_{\beta_1}\sum_{\{t\in l|S_{j,t}\le c\}}(y_t - X_t\beta_1)^2 + \lambda\|\beta_1\|_2\right.$$
$$\left.+\min_{\beta_2}\sum_{\{t\in l|S_{j,t}>c\}}(y_t - X_t\beta_2)^2 + \lambda\|\beta_2\|_2\right].$$

Generalized Time-Varying Parameters

3 ingredients to go from a single tree to a forest

For each tree:

1. **Let the trees run deep**: even though that would surely imply overfitting for a single tree, let each tree run until leafs contain very few observations (usually < 5).

Diversifying the Portfolio (i.e., creating the ensemble)

2. **Bagging**: Create B nonparametric bootstrap samples of the data. That is, we are picking $[y_t \;\; X_t]$ pairs with replacement.

3. **De-correlated trees**: At each splitting point, we only consider a subset of all predictors ($\mathcal{J}^- \subset \mathcal{J}$) for the split.

(M)RF prediction is the simple average of all the B tree predictions.

Generalized Time-Varying Parameters

Useful Addition: Random Walk Regularization

- The above implements the prior $\beta_t \sim \mathcal{N}(0, .)$.
- However, $\beta_t \sim \mathcal{N}(\beta_{t-1}, .)$, i.e., time-smoothness, makes more sense.
- I implement it via WLS with rudimentary egalitarian Olympic podium weights $w(t;\zeta)$, where $\zeta < 1$ is a tuning parameter.
- The splitting rule becomes

$$\min_{j \in \mathcal{J}^-,\ c \in \mathbb{R}} \left[\min_{\beta_1} \sum_{t \in I_1^{RW}(j,c)} w(t;\zeta) \left(y_t - X_t \beta_1\right)^2 + \lambda \|\beta_1\|_2 \right.$$

$$\left. + \min_{\beta_2} \sum_{t \in I_2^{RW}(j,c)} w(t;\zeta) \left(y_t - X_t \beta_2\right)^2 + \lambda \|\beta_2\|_2 \right].$$

Generalized Time-Varying Parameters

Inference

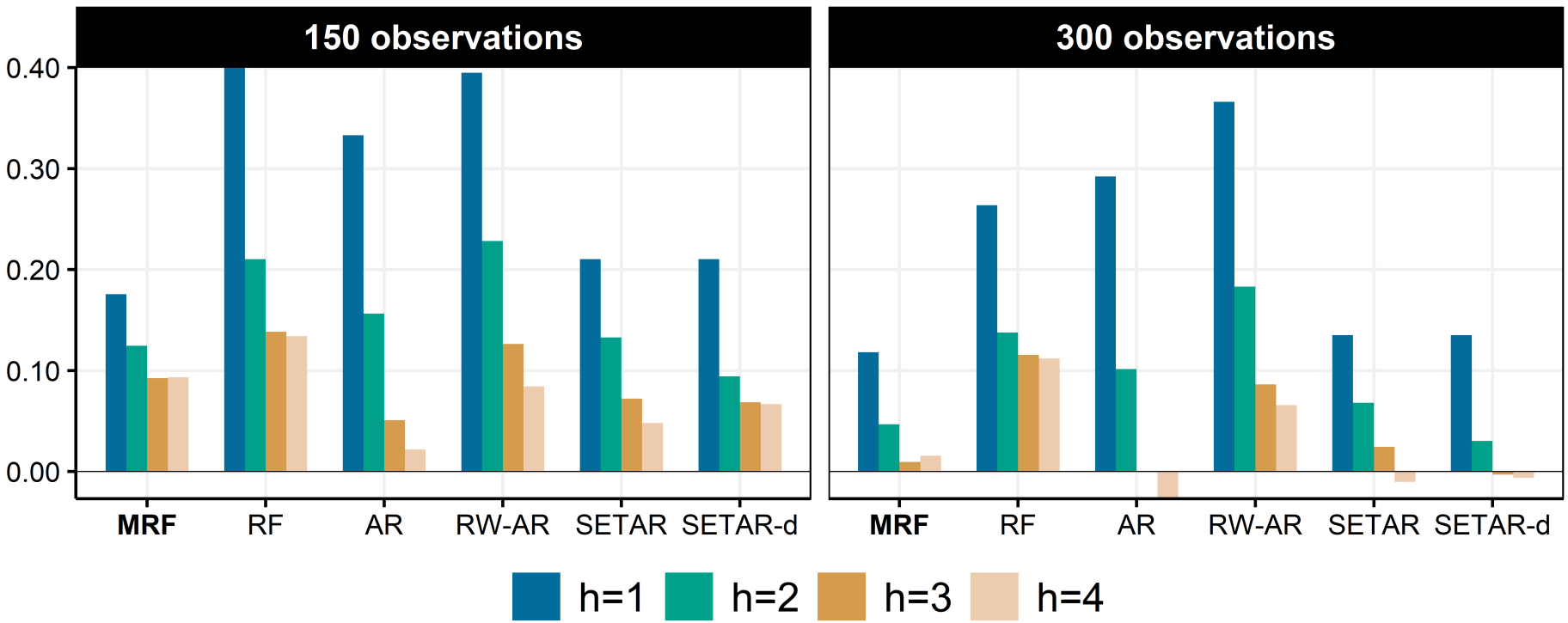
- Following (Taddy et al., 2015), *interpret* $\mathcal{F}$ as a the posterior mean of tree $\mathcal{T}$ which posterior distribution was obtained by (Rubin, 1981)'s Bayesian Bootstrap.
- Crucial advantage: no additional computations required, quantiles computed straight from the "bag" of trees.
- (Taddy et al., 2015)'s approach requires *iid* data: the bayesian model is multinomial with Dirichlet conjugate prior.
- I propose to rather use a Block Bayesian Bootstrap (BBB)

Simulations

DGP 3: Persistent SETAR

$$y_t = \phi_{0,t} + \phi_{1,t} y_{t-1} + \phi_{2,t} y_{t-2} + \epsilon_t, \quad \epsilon_t \sim N(0, 0.5^2)$$

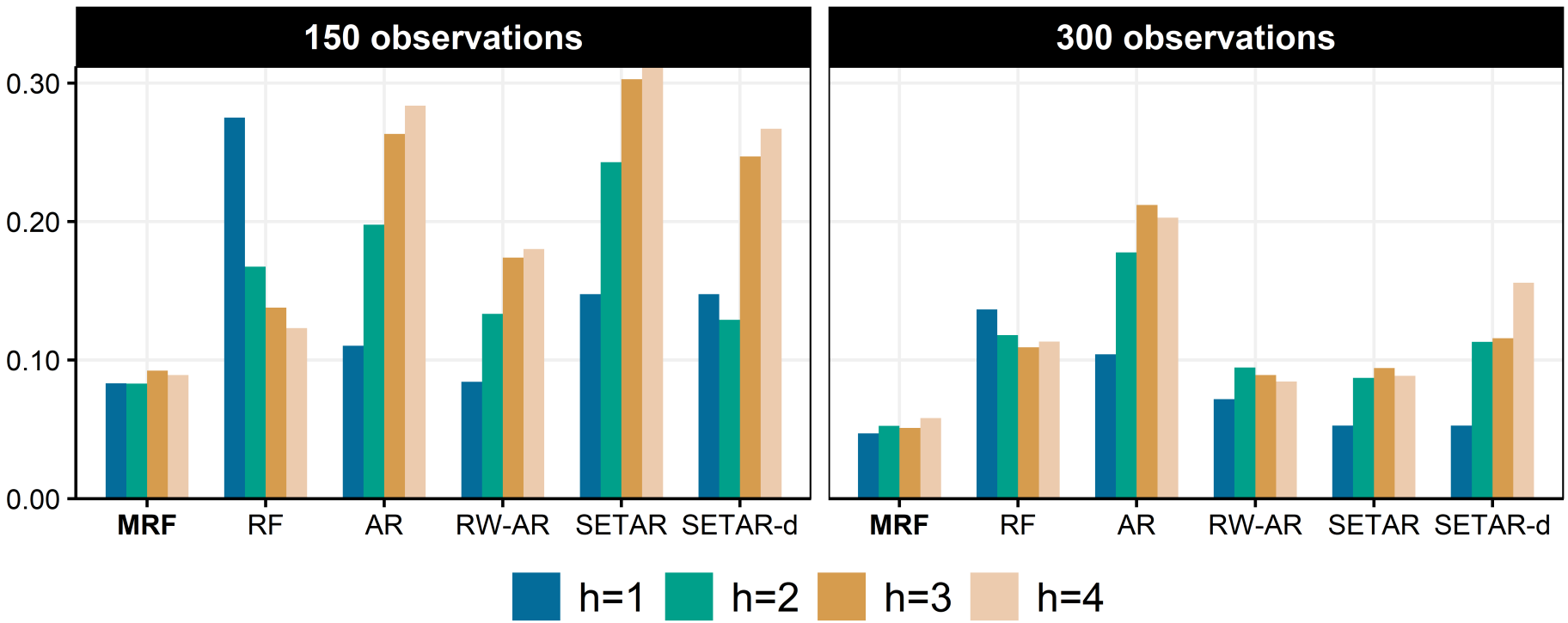
$$\beta_t = [\phi_{0,t} \;\; \phi_{1,t} \;\; \phi_{2,t}] = \begin{cases} [2 \;\; 0.8 \;\; -0.2], & \text{if } y_{t-1} \geq 0 \\ [0.25 \;\; 1.1 \;\; -0.4], & \text{otherwise} \end{cases}$$



Simulations

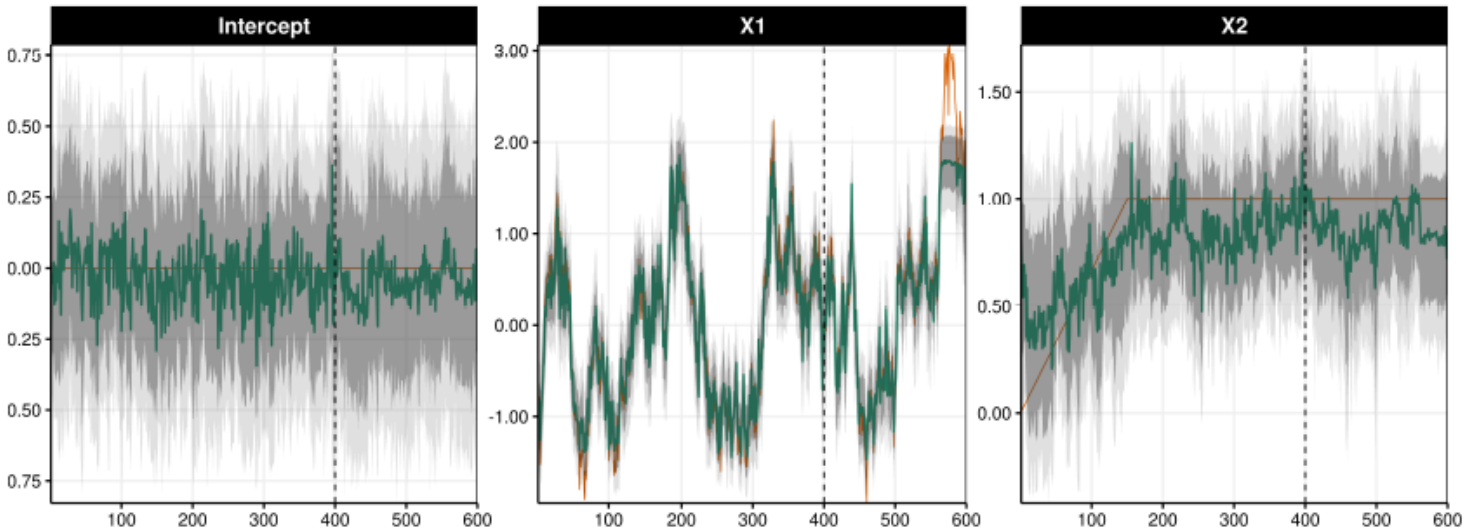
DGP 6: SETAR that morphs instantly in AR(2)

$$\text{DGP 6} = \begin{cases} \text{SETAR}, & \text{if } t < T/2 \\ \text{Plain AR(2)}, & \text{otherwise} \end{cases}$$

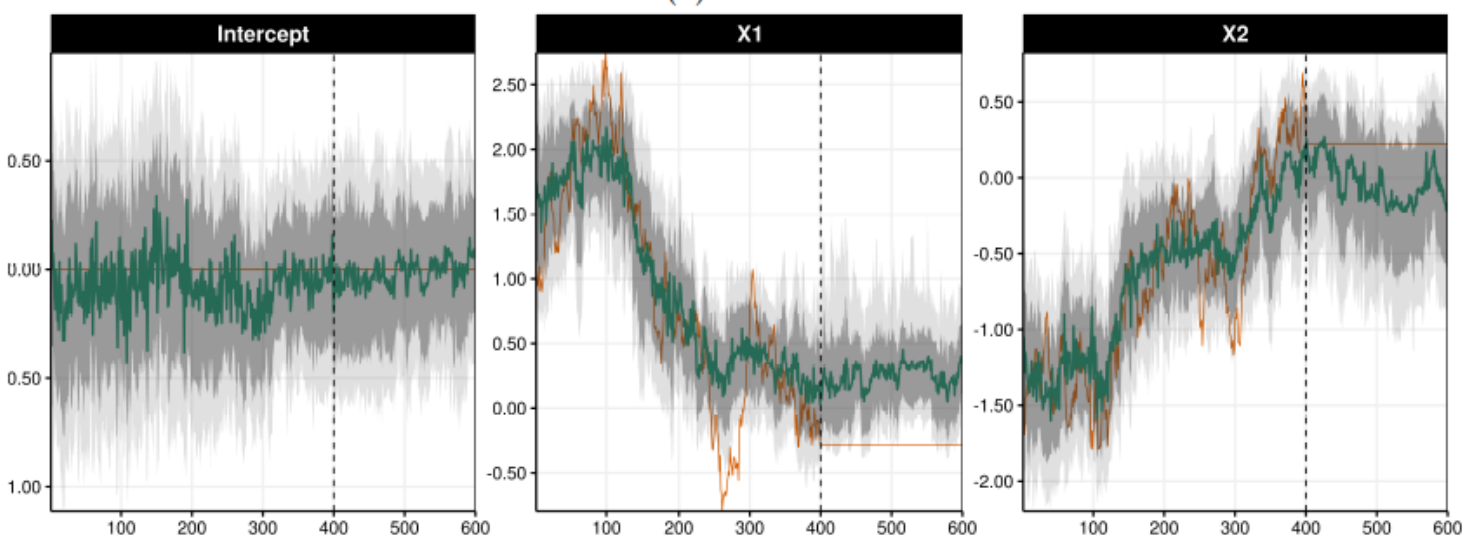


Simulations

A look at GTVPs under Different Contexts, when S_t is large



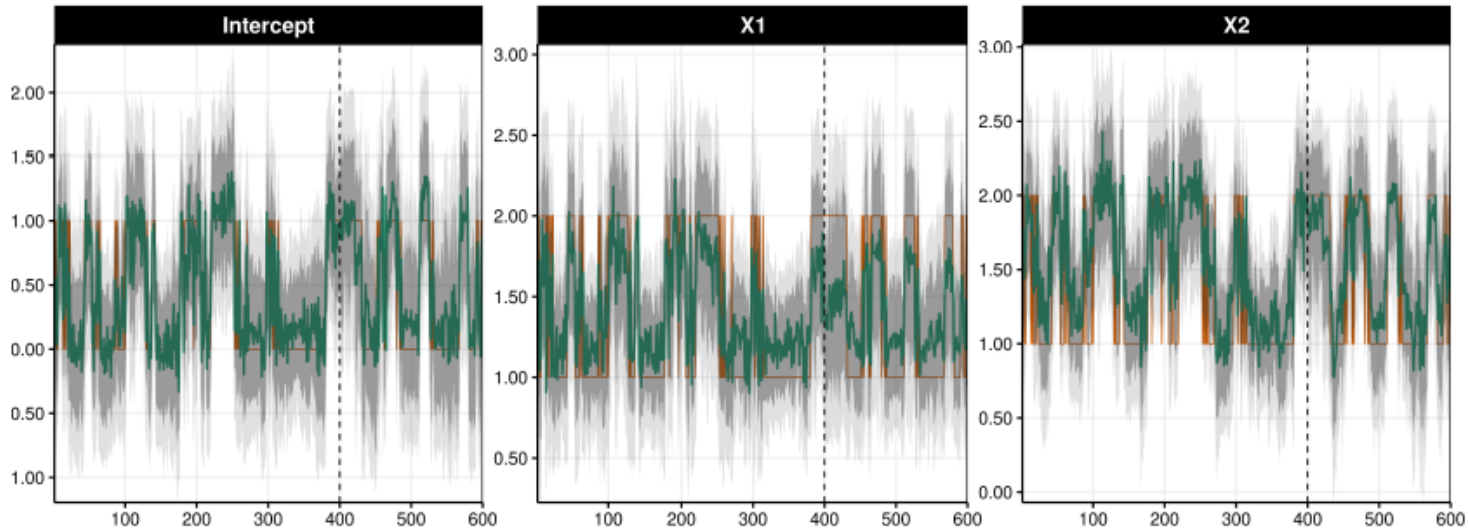
(d) DGP 4



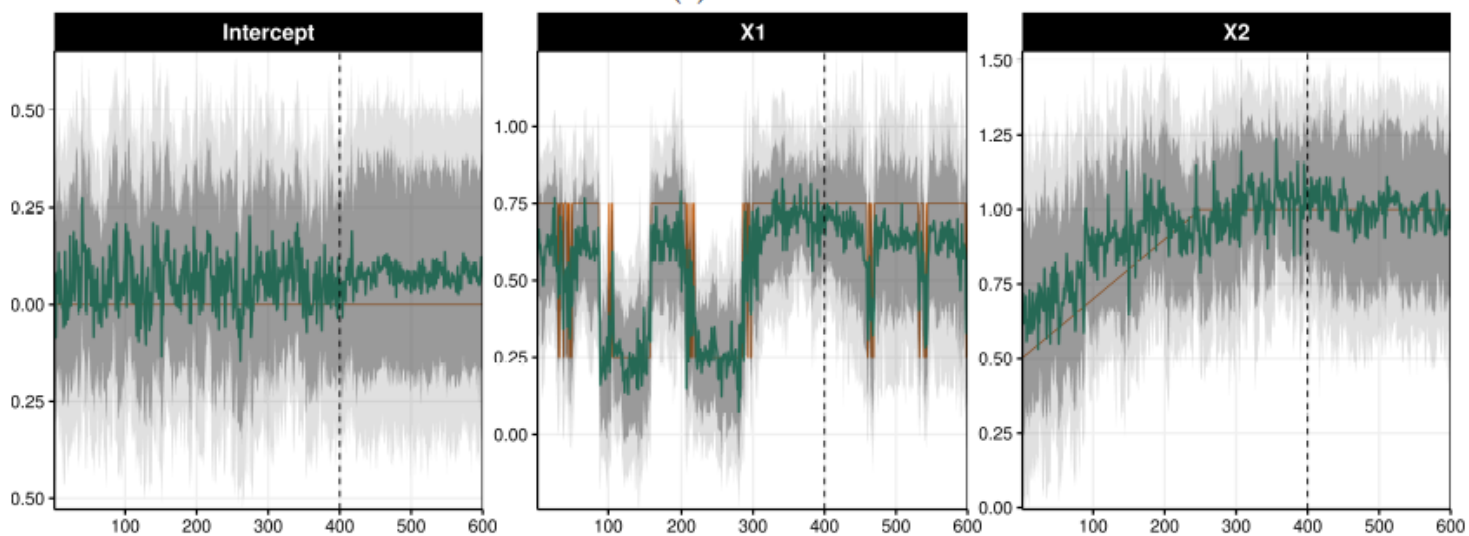
(e) DGP 5

Simulations

A look at GTVPs under Different Contexts, when S_t is large



(a) DGP 1



(b) DGP 2

Forecasting

Setup

- Data: FRED-QD, the SW data set update by (McCracken and Ng, 2016), 260 series
- POOS period starts on 2002Q1 and ends 2014Q1. *Expanding* window estimation from 1959Q3.
- Horizons: $h \in \{1, 2, 4, 6, 8\}$ quarters
- 6 variables of interest: GDP growth, Unemployment Rate (UNRATE) growth, Interest Rate (GS1), Inflation (Δlog(CPIAUCSL)), Housing Starts (HOUST) and some spread (T10YFFM).
- Evaluation metric is $RMSPE_{v,h,m} = \sqrt{\sum_{t \in OOS} (y_t^v - \hat{y}_{t-h}^{v,h,m})^2}$

Forecasting

About the composition of S_t

1. 8 lags of y_t
2. t for structural breaks/exogenous time-variation
3. 2 lags of all variables in FRED
4. F to summarize the cross-section variation: 8 lags of 5 factors extracted from FRED by PCA

Most importantly

5. For each variable j, I generate two $\text{MAF}_{t,j}$ (Moving Average Factors) summarizing the information contained in its distributed lags.
   - Bypasses the need to penalize explicitly a nonexistent lag polynomial.
   - Done by PCA on 8 lags.
   - Further studied for many ML models (along with other transformations) in (Goulet Coulombe et al., 2021a).

Forecasting

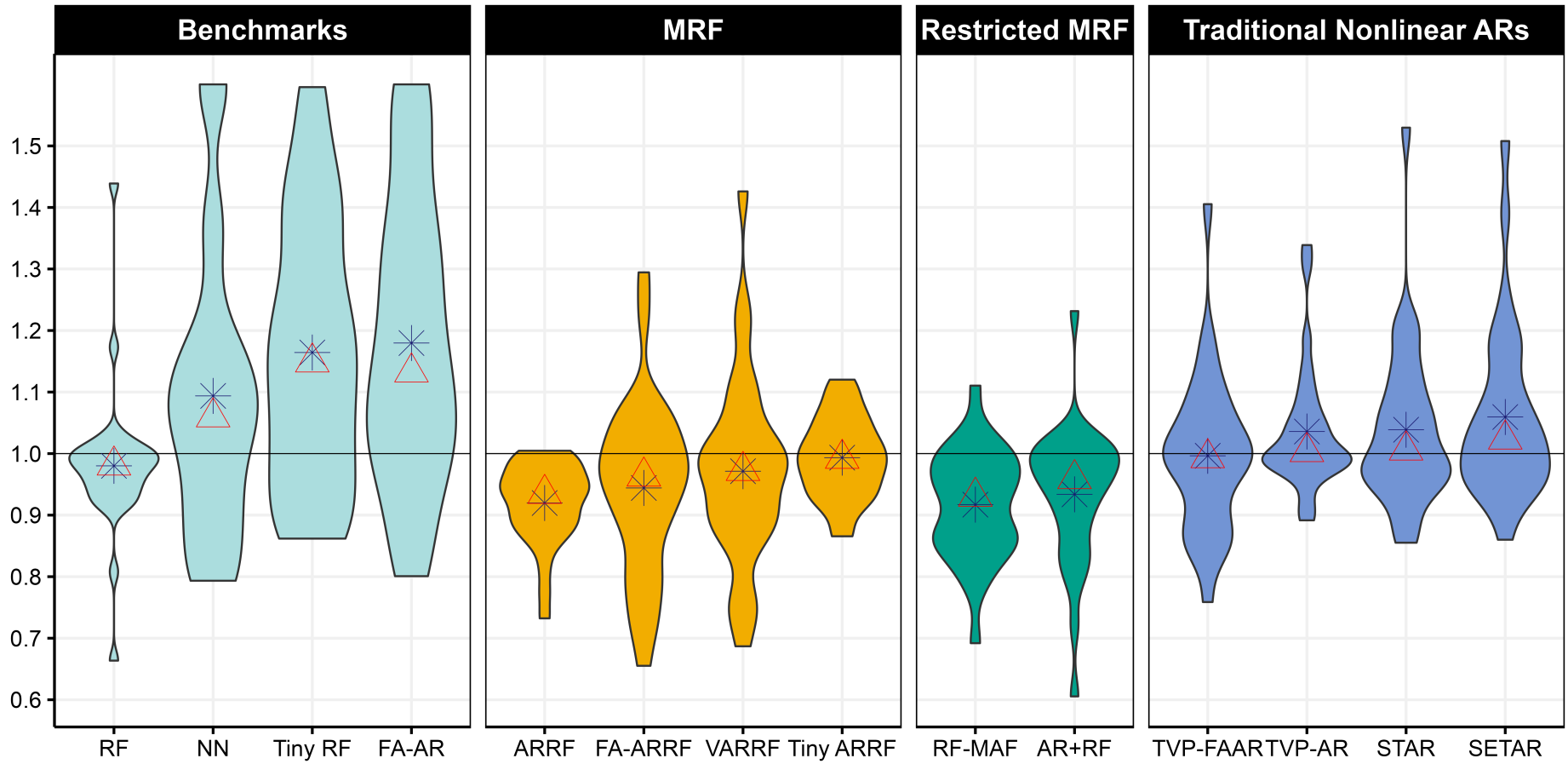
Main Models

Table: Wild Horses

Acronym	Linear Part (X_t^m)	RF part
AR	$[1,\ y_{t-\{1:4\}}]$	$\emptyset$
FA-AR	$[1,\ y_{t-\{1:4\}},\ F_{1,t-\{1:2\}},\ F_{2,t-\{1:2\}}]$	$\emptyset$
RF	$\emptyset$	8 lags of all raw data
Tiny RF	$\emptyset$	$[y_{t-\{1:8\}},\ t]$
RF-MAF	$\emptyset$	S_t
AR+RF	Filter y_t first with an AR(4)	S_t
ARRF	$[1,\ y_{t-\{1:2\}}]$	S_t
Tiny ARRF	$[1,\ y_{t-\{1:2\}}]$	$[y_{t-\{1:8\}},\ t]$
FAARRF	$[1,\ y_{t-\{1:2\}},\ F_{1,t-1},\ F_{2,t-1}]$	S_t
VARRF	$[1,\ y_{t-\{1:2\}},\ GDP_{t-1},\ IR_{t-1},\ INF_{t-1}]$	S_t

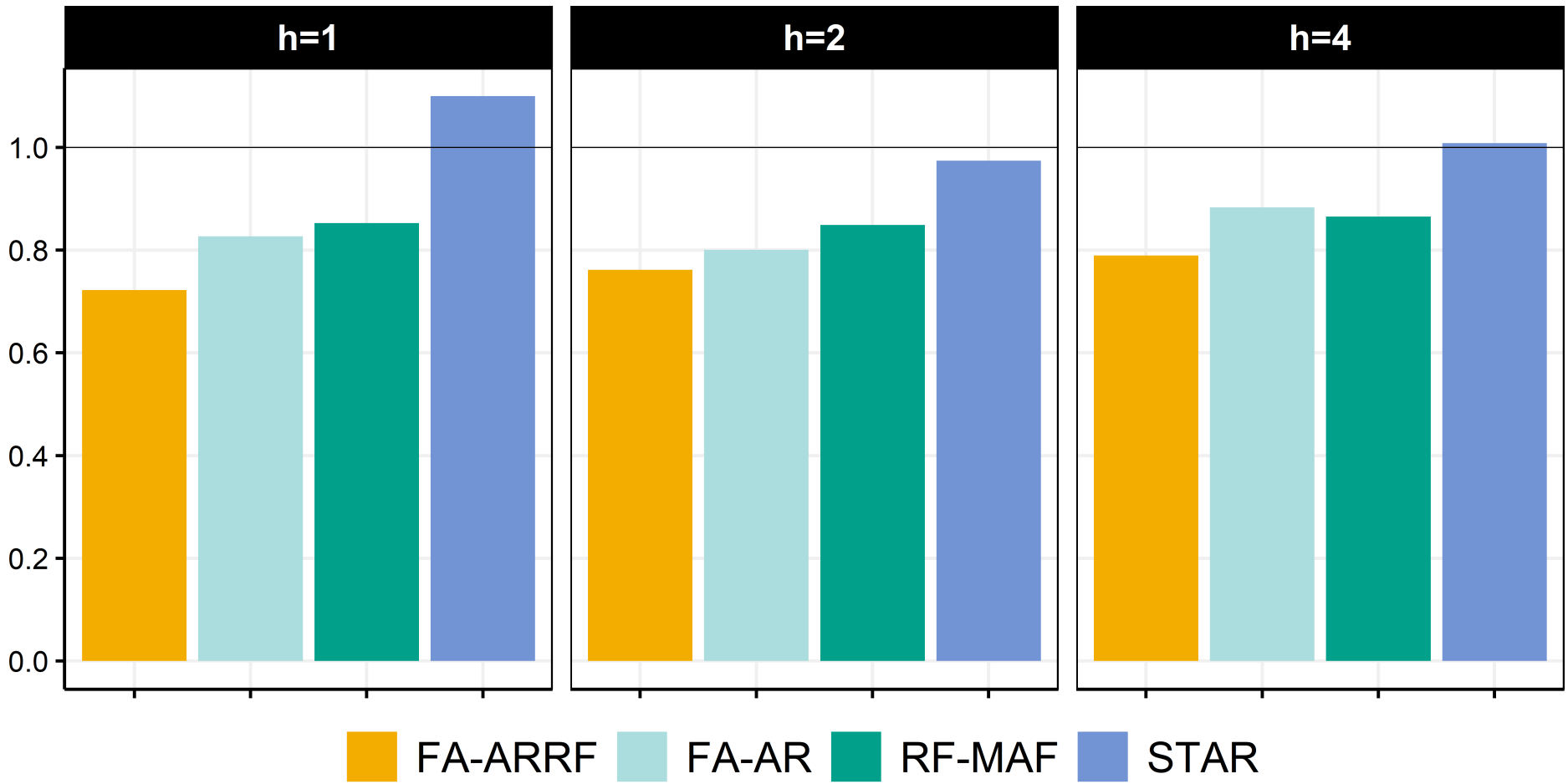
Forecasting

Visualizing the distribution of $RMSPE_{v,h,m}/RMSPE_{v,h,AR}$



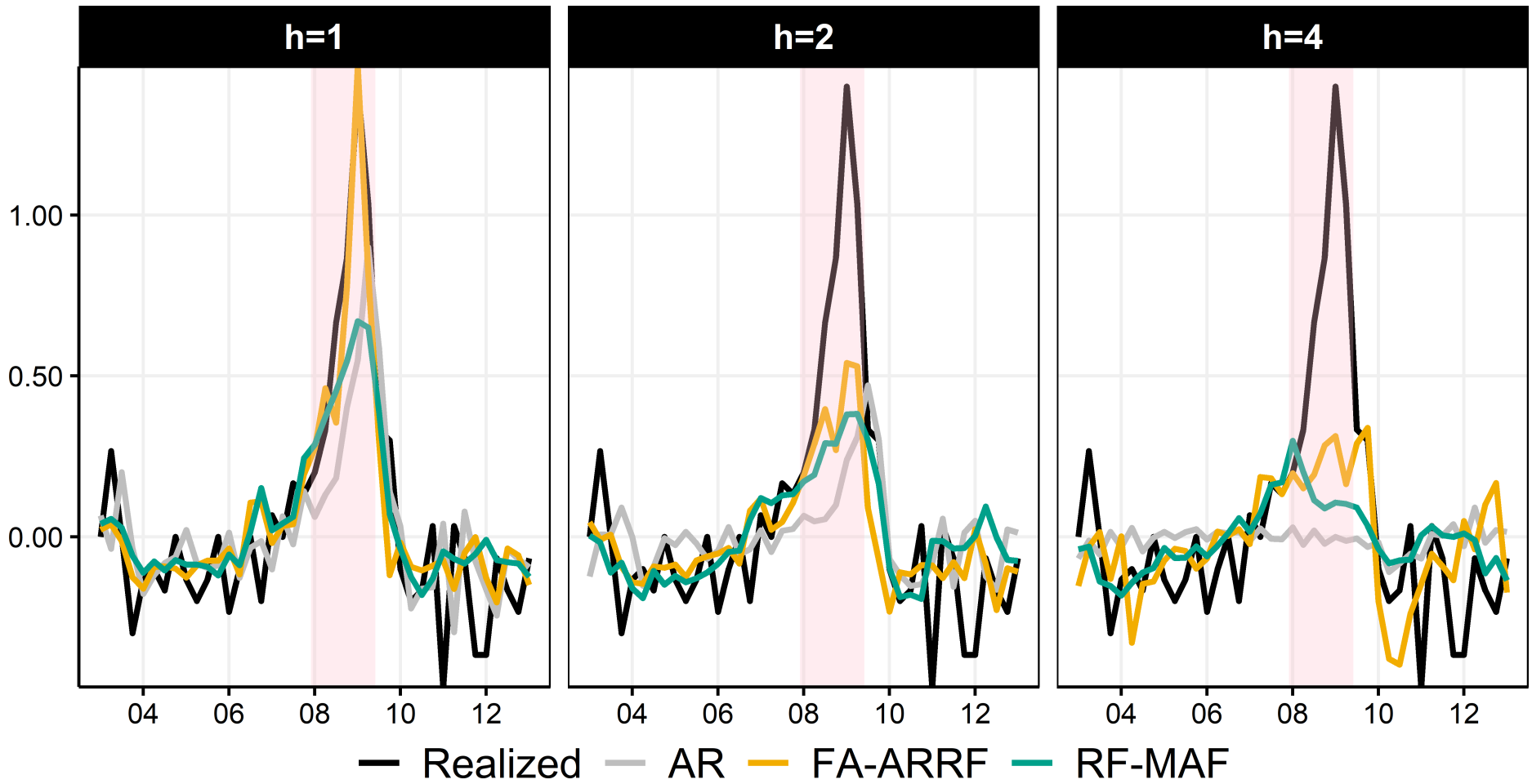
Forecasting

$RMSPE_{UR,h,m}$ / $RMSPE_{UR,h,AR}$ in more detail



Forecasting

What do forecasts look like for UR? $\rightarrow R^2_{OOS}$ 80% for $h = 1$



GTVPs of the one-quarter ahead UR forecast

$$\Delta UR_{t+1} = \mu_t + \phi_t^1 y_t + \phi_t^2 y_{t-1} + \gamma_t^1 F_t^1 + \gamma_t^2 F_t^2 + e_{t+1}.$$

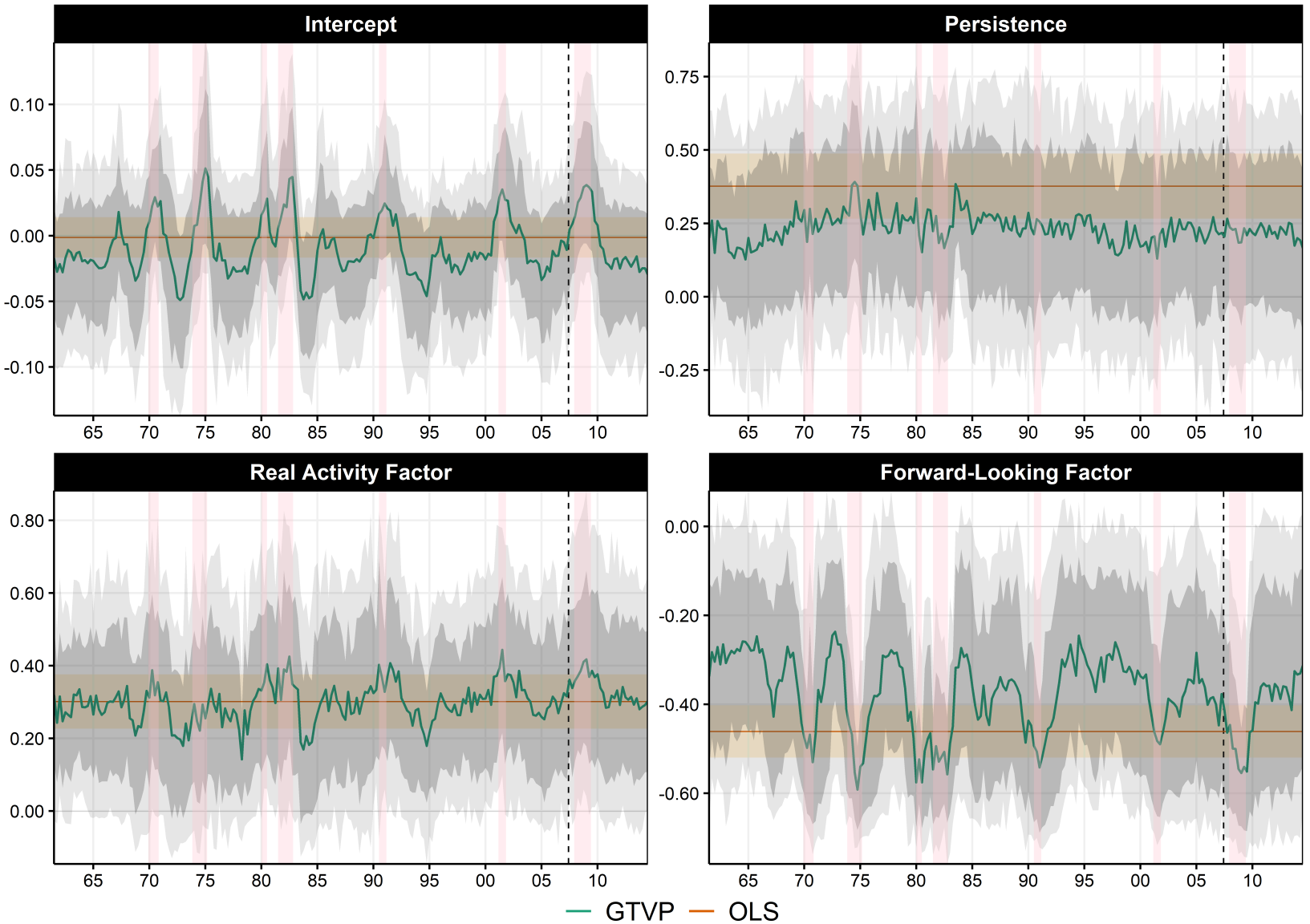


Figure: GTVPs of the one-quarter ahead UR forecast. The grey bands are the 68% and 90% credible region. The pale orange region is the OLS coefficient ± one standard error. The vertical dotted blue line is the end of the training sample. Pink shading corresponds to NBER recessions.

Dynamic β_t Learning

$$\Delta UR_{t+1} = \mu_t + \phi_t^1 y_t + \phi_t^2 y_{t-1} + \gamma_t^1 F_t^1 + \gamma_t^2 F_t^2 + e_{t+1}.$$

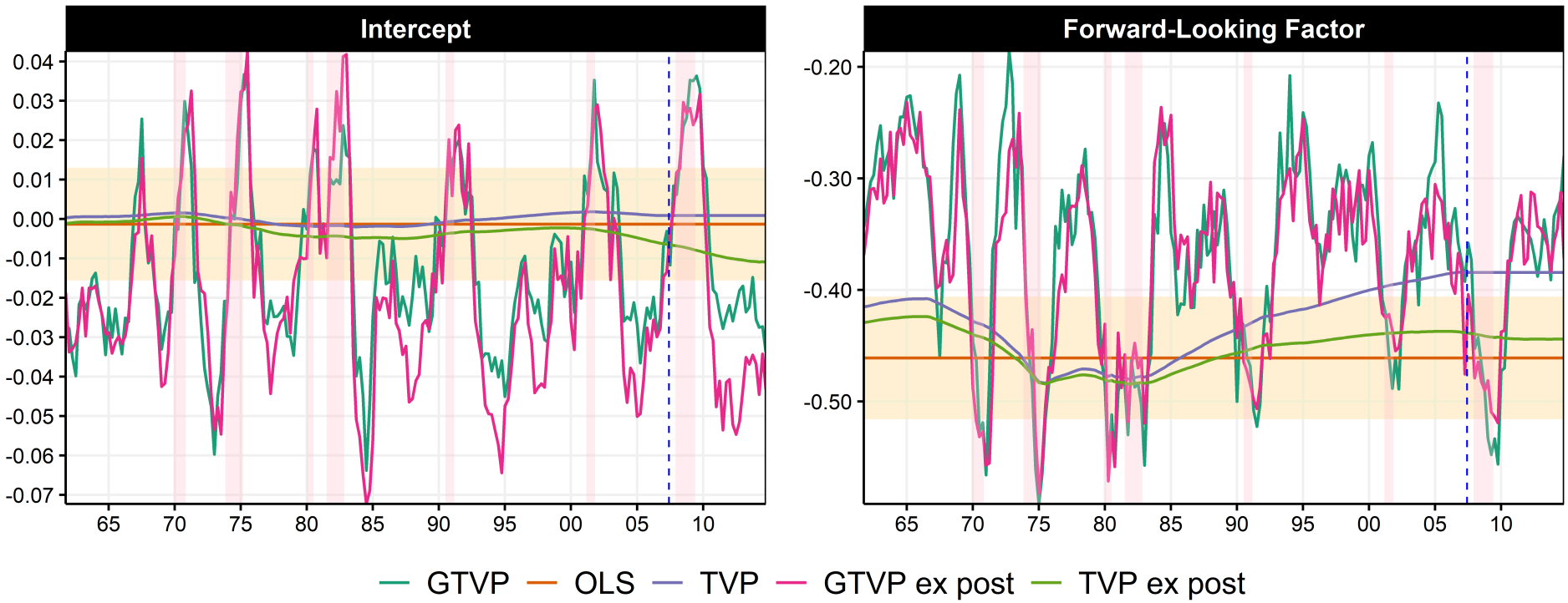
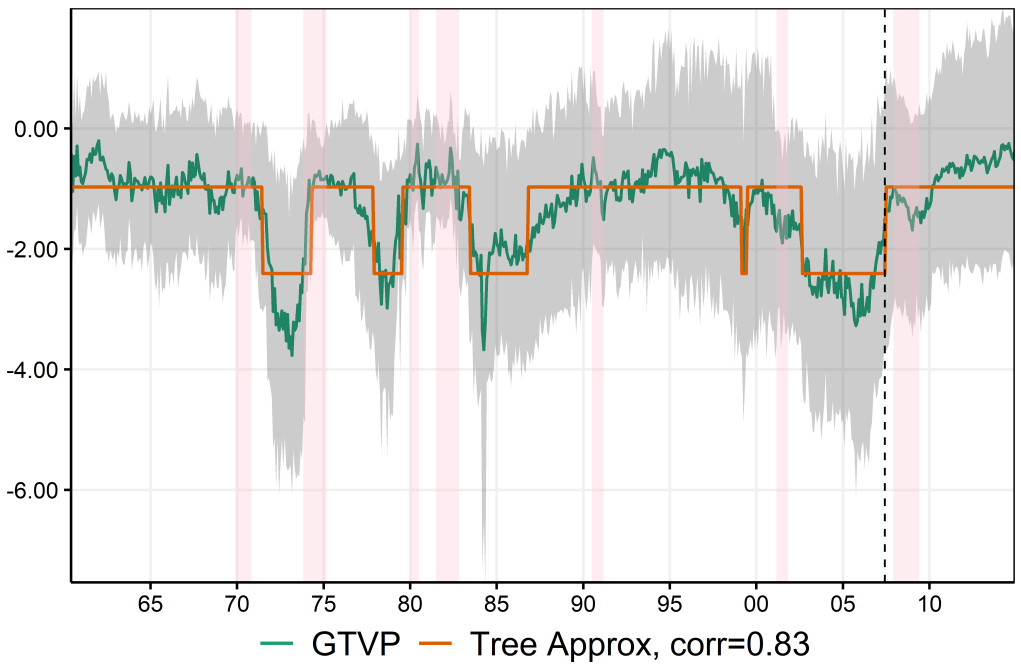


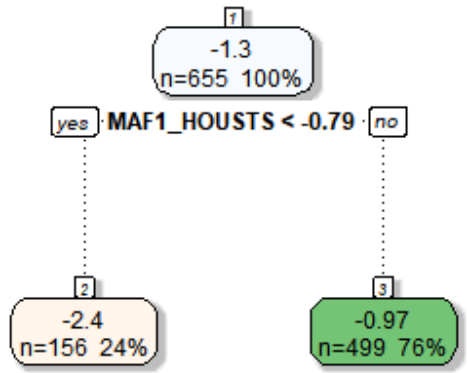
Figure: Comparing TVPs and GTVPs, ex-ante and ex-post.

An Interesting Observation for (monthly) *Inflation*

$$\pi_{t+1} = \mu_t + \phi_t^1 y_t + \phi_t^2 y_{t-1} + \gamma_t^1 F_t^1 + \gamma_t^2 F_t^2 + e_{t+1}.$$



(a) Surrogate Model Replication

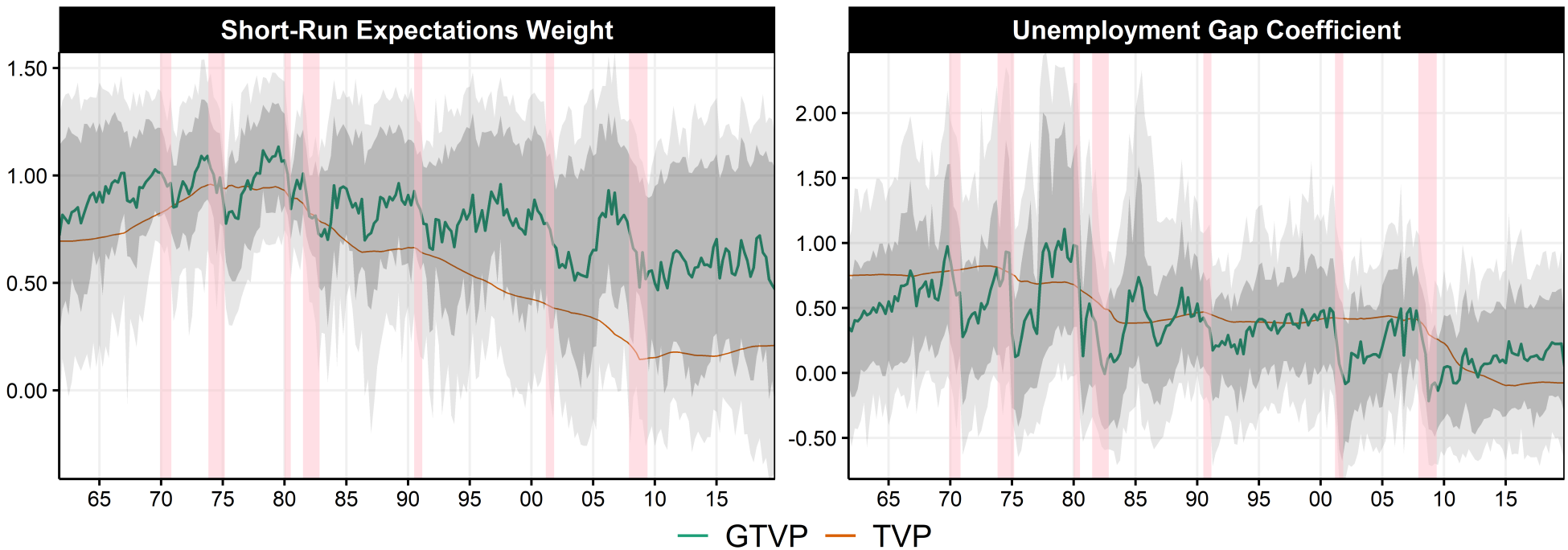


(b) Corresponding Tree

A more traditional Phillips' Curve

À la (Blanchard et al., 2015) and many others

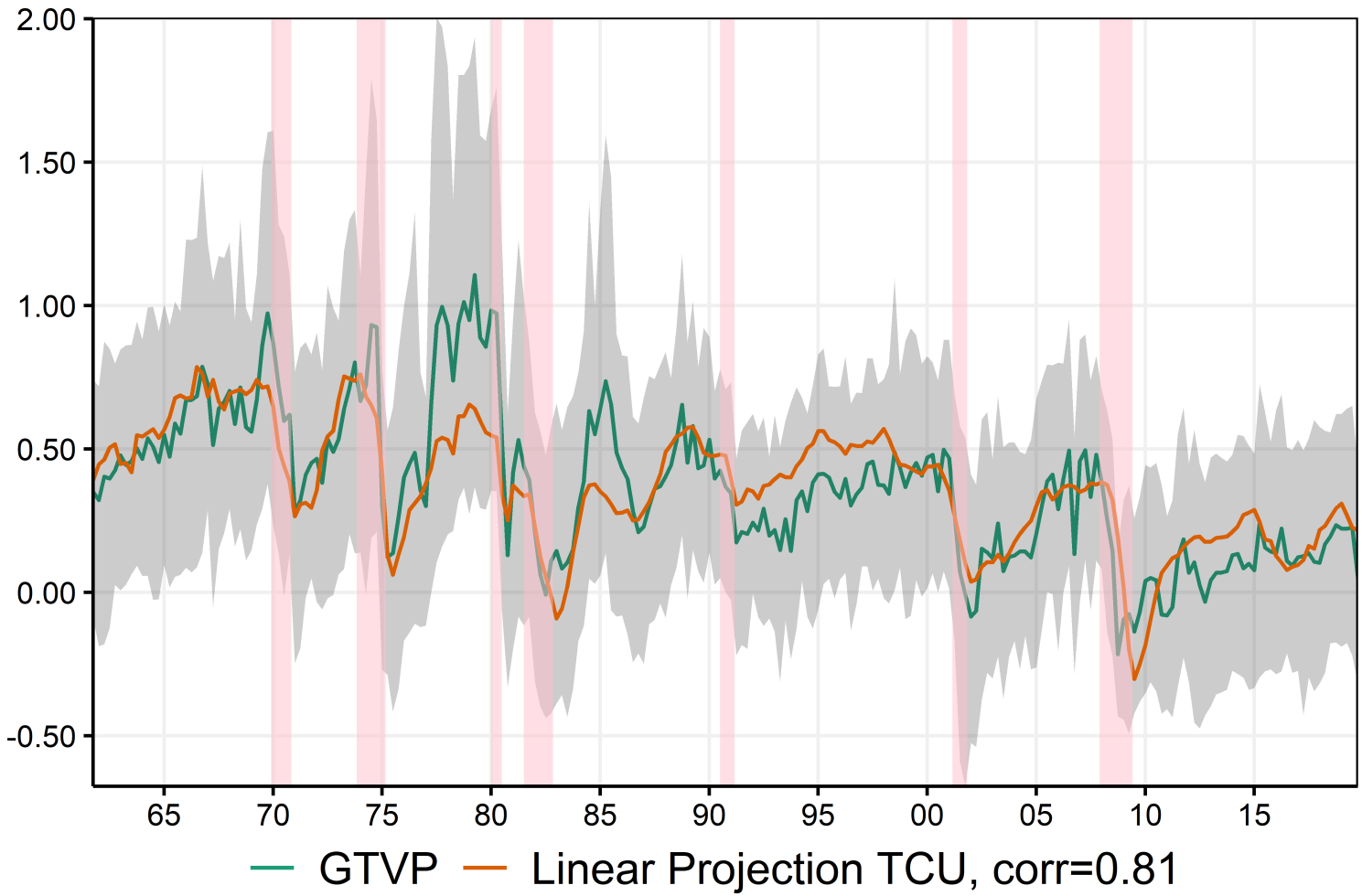
$$\pi_t = \mu_t + \beta_{1,t}\hat{\pi}_t^{SR} + \beta_{2,t}u_t^{GAP} + \beta_{3,t}\pi_t^{IMP} + \varepsilon_t$$



A more traditional Phillips' Curve

$\beta_{2,t}$ looks like Total Capacity Utilization (TCU)

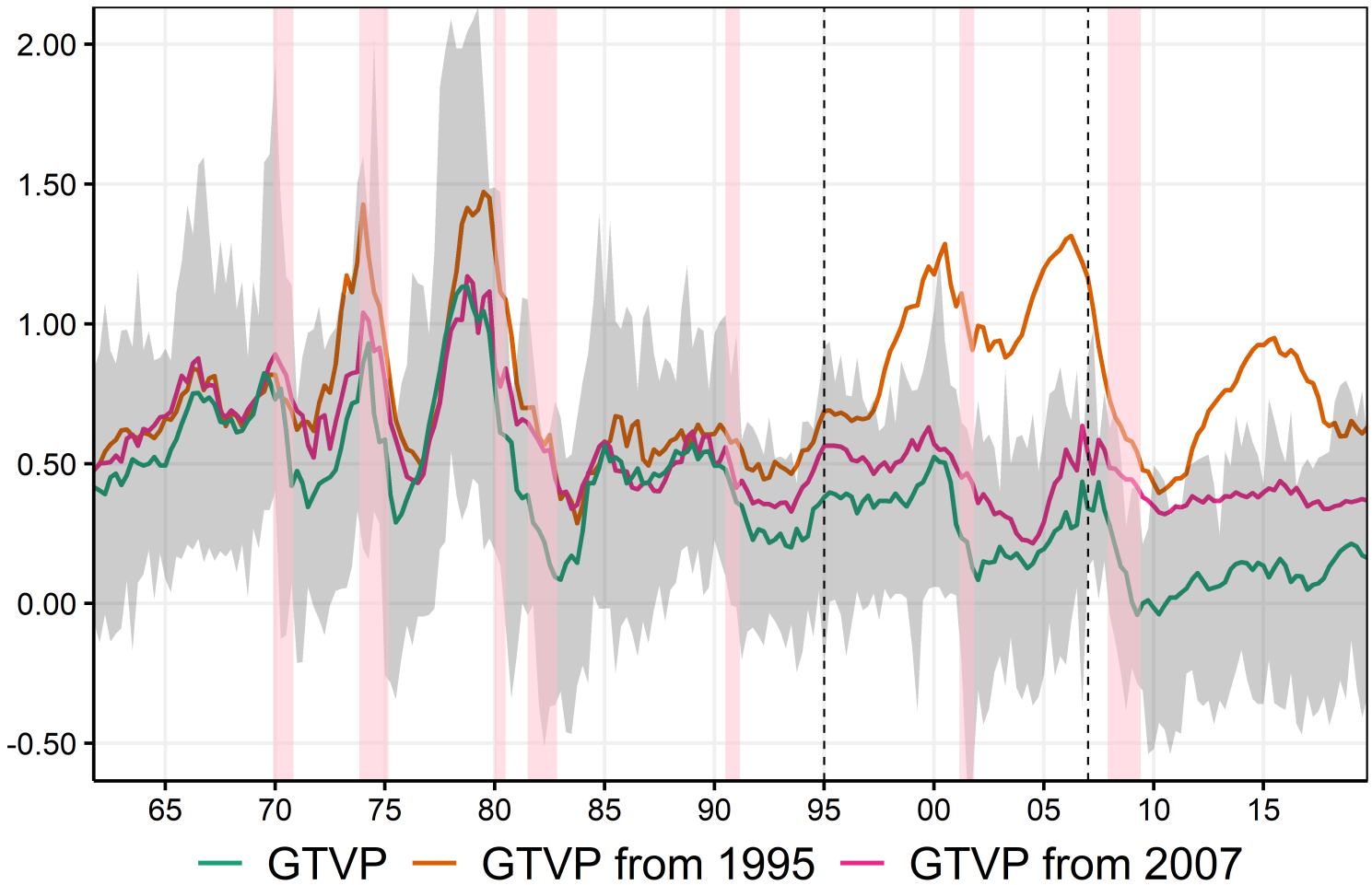
$$\pi_t = \mu_t + \beta_{1,t}\hat{\pi}_t^{SR} + \beta_{2,t}u_t^{GAP} + \beta_{3,t}\pi_t^{IMP} + \varepsilon_t$$



Dynamic Phillips' Curve Learning

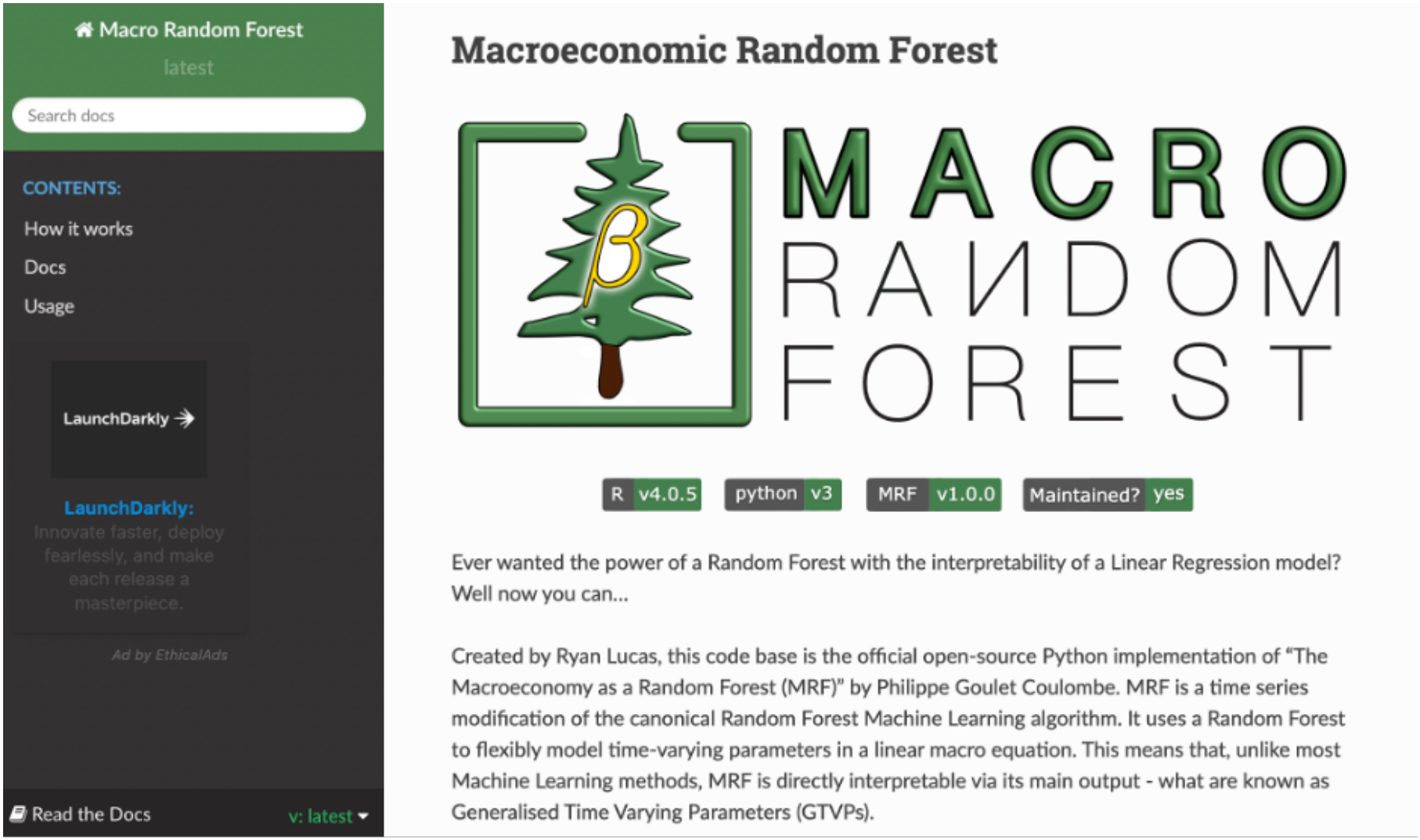
Comparing "out-of-sample" predictions of GTVPs at different points in time

$$\pi_t = \mu_t + \beta_{1,t}\hat{\pi}_t^{SR} + \beta_{2,t}u_t^{GAP} + \beta_{3,t}\pi_t^{IMP} + \varepsilon_t$$



Packages

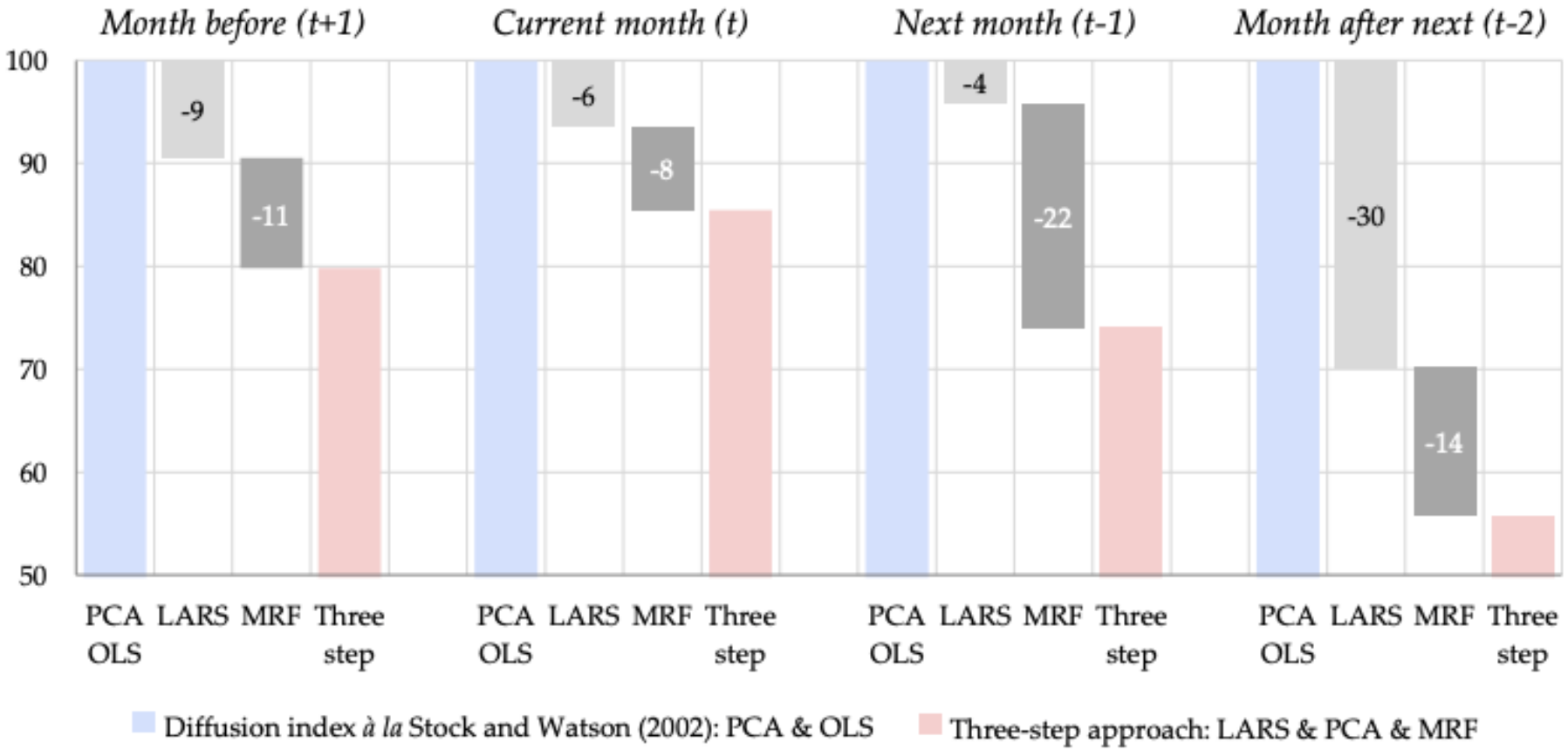
There is a `R package` and a `Python` one coded by Ryan Lucas (MIT).



Some Applications

- Arctic Sea Ice (Diebold et al., 2023)
- Macro forecasting for UK (Goulet Coulombe et al., 2021b)
- World Trade Nowcasting (Chinn et al., 2023)

Figure N1. Decomposition of accuracy gains relative to PCA-OLS



III.

On the Unreasonable Reliability of Random Forest

The R^2_{test} vs R^2_{train} Puzzle

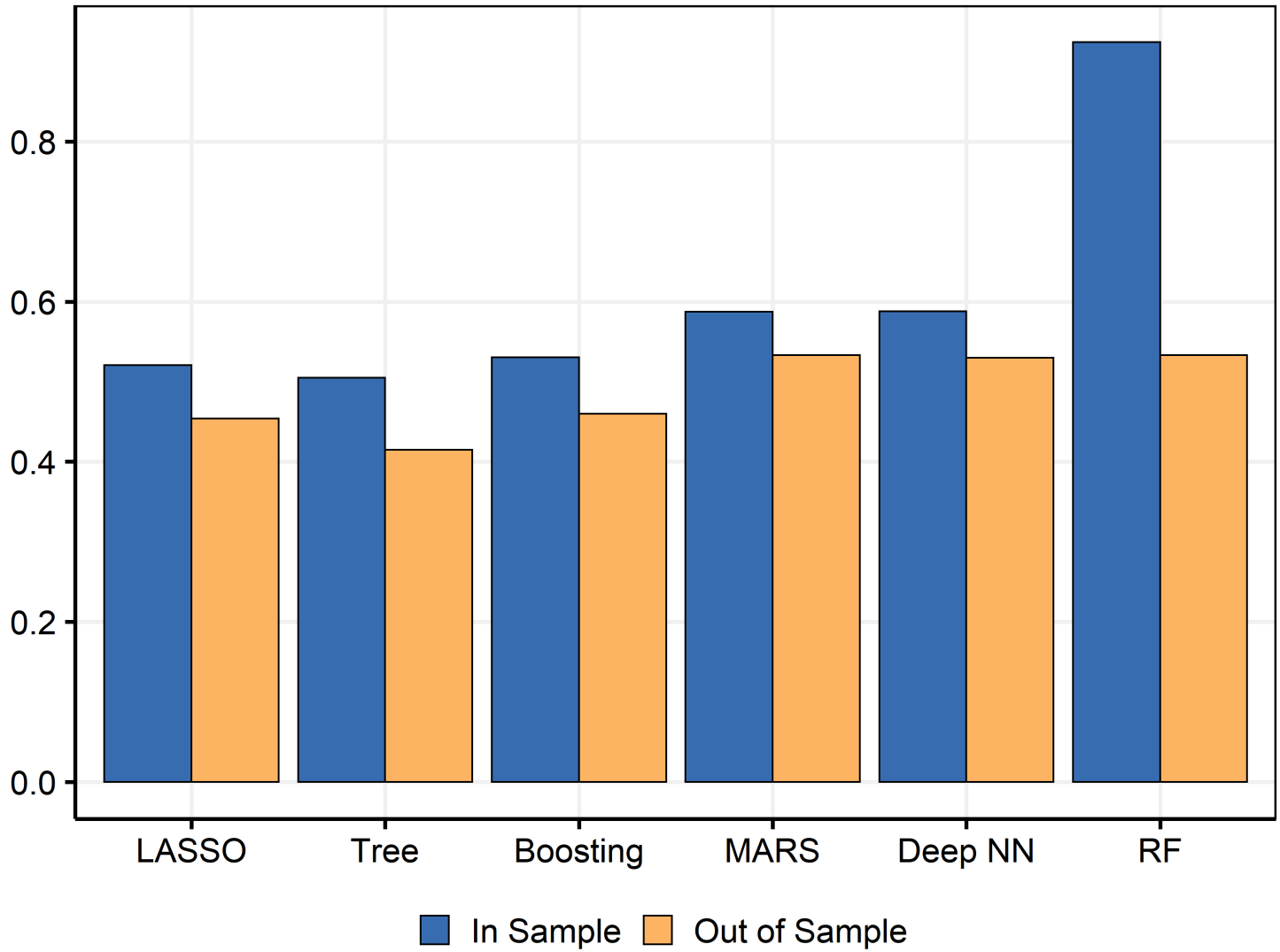


Figure: *Abalone* data set example

Usual explanations for RF's success don't explain it

- (Breiman, 2001) originally derived an upper bound on the generalization error of RF — it decreases as $\mathcal{T}$ strength increases, and increases as correlation between them increases.
  - $\Rightarrow$ Nice to have, but it does not say much about results obtained in practice.
- (Bühlmann and Yu, 2002): bagging brings smoothness (hence regularization)
  - $\Rightarrow$ If that was just that, then $R^2_{\text{test}} \approx R^2_{\text{train}}$ like for any usual smoothing method
- (Mentch and Zhou, 2019) (and ESL): randomization implies a ridge-like regularization obtained by model averaging – an adequate argument for *global linear* models (reminiscent of (Elliott et al., 2013)'s CSR)
  - $\Rightarrow$ If that was just that, then $R^2_{\text{test}} \approx R^2_{\text{train}}$ like for Ridge
- (Belkin et al., 2019) claims RF has a "double-descent" risk curve, like Neural Nets.
  - $\Rightarrow$ Their construction confused additional trees with additional complexity, which is true for Boosted Trees, but not RF. In fact, RF has a single, never-ascending, descent.

Roadmap

- Why the puzzle occurs and what it tells us about RF's legendary robustness to overfitting
  1. What happens in the overfitting zone stays in the overfitting zone
  2. Bagging + Perturbation (`B` & `P`) as an approximation to population sampling (and a *Perfectly* Random Forest)
- Those ideas should apply to any *randomized greedy algorithm* $\rightarrow$ leverage those to develop two new "self-tuning" algorithms
  1. Booging
  2. MARSquake
- Slow-Growing Trees

Greed is Good

The Key: *A greedy algorithm treats what has already happened as given and what comes next as if it will never happen.*

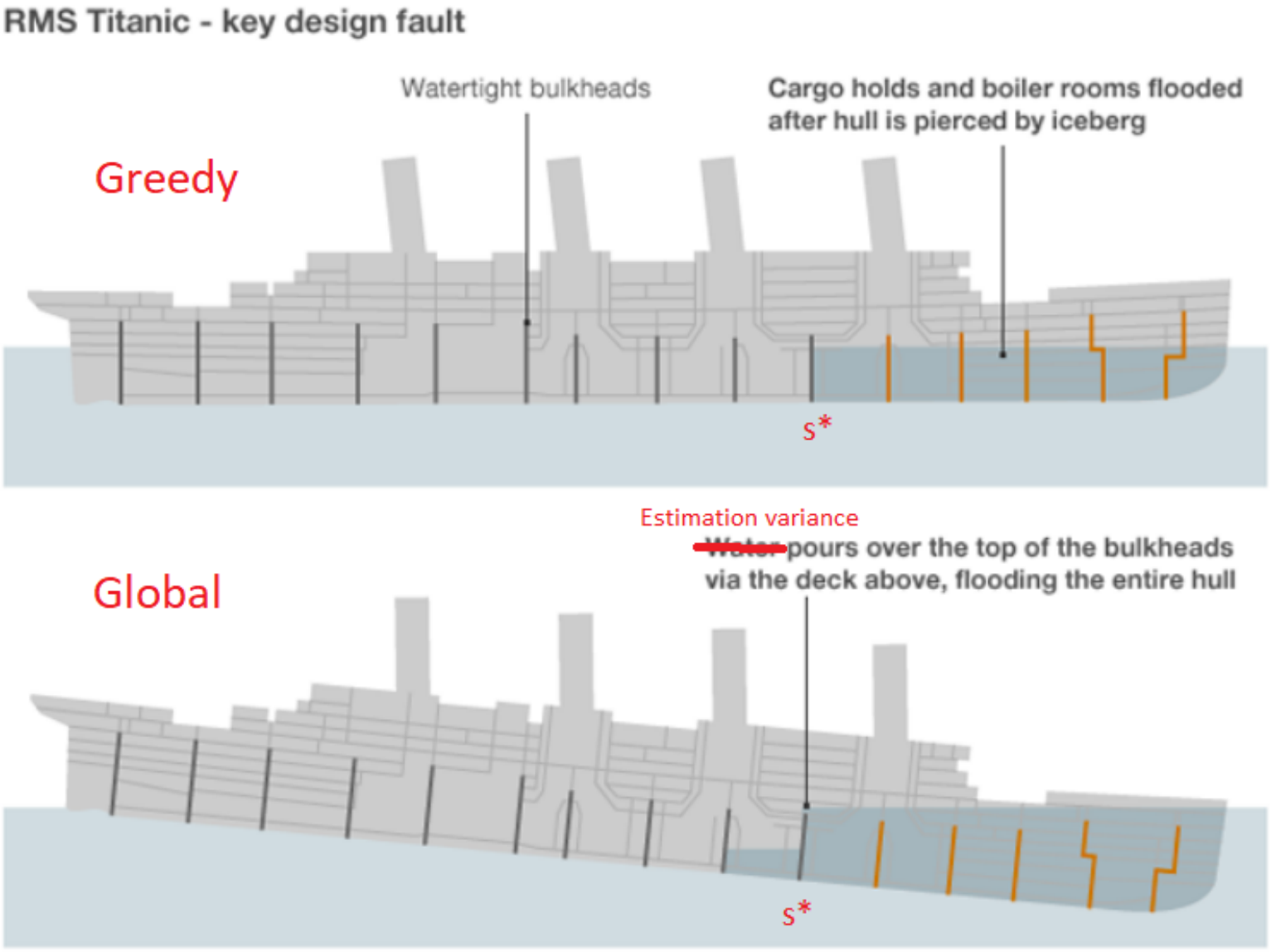
- **Old song**: greedy optimization is an inevitable (but suboptimal) practical approach in the face of computational adversity (see ESL) — bad because no guarantee to get the "optimal" tree.
- **New song**: by building recursively a model of increasing complexity (when true complexity s^* is unknown) in a stepwise fashion, what is estimated in early steps is immune to the "pollution" brought by the latter steps (which are likely overfitting).

What happens past s^* stays past s^*

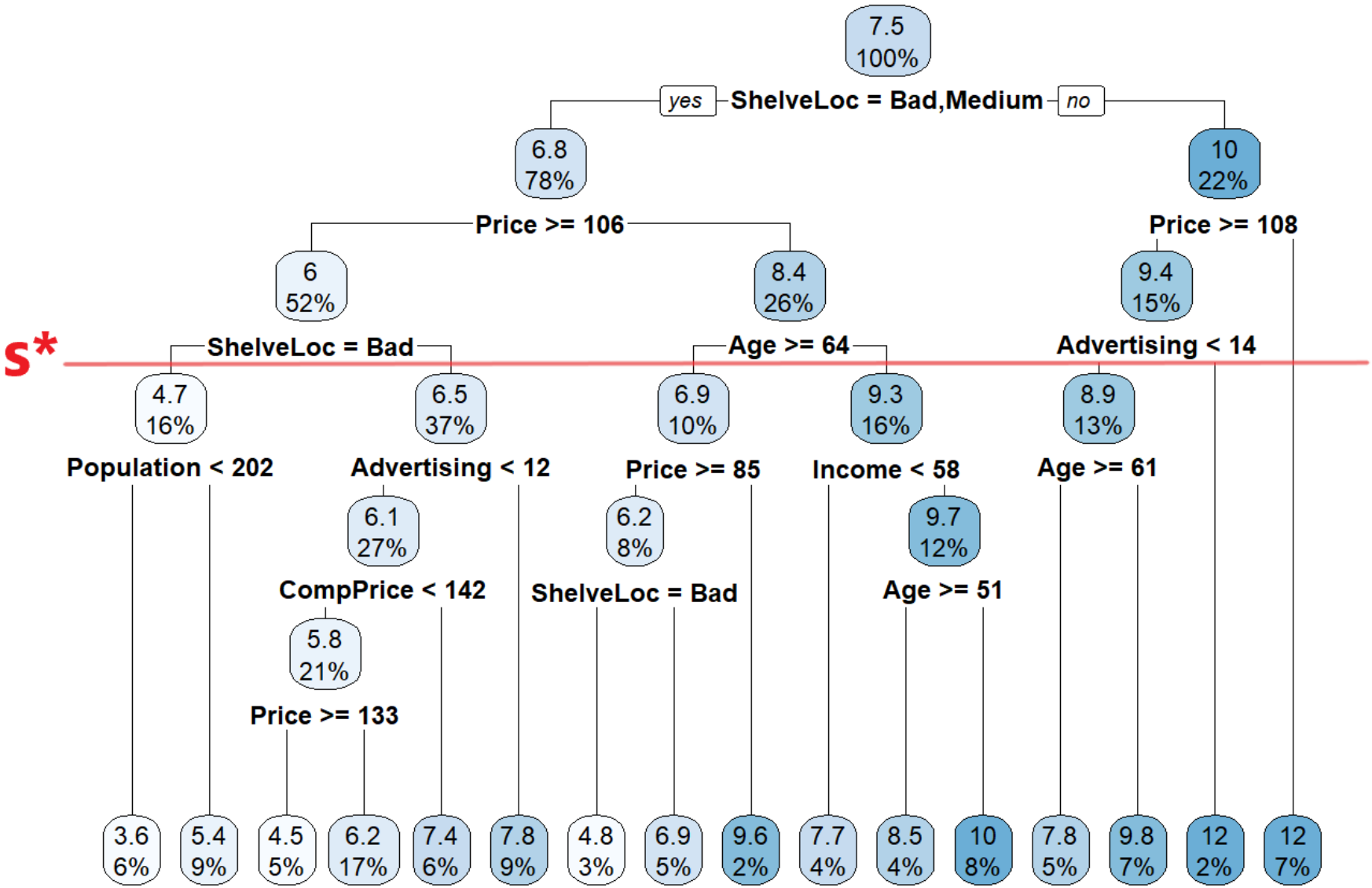
$$\hat{y}_i = \underbrace{\underbrace{\beta_1 x_{1,i}}_{s=1} + \underbrace{\beta_2 x_{2,i}}_{s=2} + \underbrace{\beta_3 x_{3,i}}_{s=3}}_{OLS}$$

- **Global** optimization (think OLS): overfitting weakens the whole prediction function
  - $\Rightarrow$ estimating many useless coefficients inflates the generalization error by increasing the variance of *both* the useful coefficients and the useless ones.
- **Greedy** optimization (think tree, or boosting): the function estimated before s is *treated as given*.
  - $\Rightarrow$ the algorithm eventually reach s^* where the only thing left to fit is the unshrinkable "true" error $\epsilon_i = y_i - \hat{f}_{s^*}(x_i)$, i.e., overfitting.
  - $\Rightarrow$ But this does not alter $\hat{f}_{s-1}$ since it is not re-evaluated. Only useless stuff is added on "top" of it.
  - $\Rightarrow$ More concretely, $\hat{\beta}$'s estimated or tree splits estimated before s^* cannot be revoked, and the predictive structure attached to them cannot weaken by ulterior steps.

RMS Titanic, Compartments, and ML Algorithms



Visually, for a Tree



What is happening beyond s^*?

- At s^*, the unknown point of optimal early stopping (aka the *true* terminal node in the case of a tree), the DGP is

$$y_i = \mu + \epsilon_i. \tag{2}$$

  and the best prediction is clearly the sample average. And yet, the algorithm continues to fit beyond s^*.

- Two questions:
  1. What is the prediction of a "perfectly random forest"? That is, one where we replaced `B` & `P` by population sampling – fitting fully overfitted greedy trees on *non-overlapping* samples of the same DGP?
  2. Can `B` & `P` provide a good approximation to the ideal PRF when applied to trees? (This is an empirical matter.)

The Perks of a *Perfectly* Random Forest

- We are looking at the prediction for a new data point j using f trained on observations $i \neq j$.
- Assume fully grown trees – terminal nodes include one observation.
- Since the tree is fitting noise, each out-of-sample tree prediction is a randomly chosen y_i for each b.
- Define $r = {}^{B}/_{N}$ where N is the number of training observations and r will eventually stand for "replicas".
- Since the $y_{i(b)}$'s amount to random draws of $y_{1:N}$, for a large enough B, we know with certainty that the vector to be averaged will contain r times the same observation y_i.
- Remembering that $r = {}^{B}/_{N}$, the prediction is

$$\hat{\mu}_j^{\text{RF}} = \frac{1}{B}\sum_{b=1}^{B} y_{i(b)} = \frac{1}{B}\sum_{i=1}^{N}\sum_{r'=1}^{r} y_{i,r'} = \frac{1}{B}\sum_{i=1}^{N}\sum_{r'=1}^{r} y_i = \frac{r}{B}\sum_{i=1}^{N} y_i = \frac{1}{N}\sum_{i=1}^{N} y_i$$

- When a PRF is starting to fit pure noise, its out-of-sample prediction collapses to $\bar{y}$, which is *optimal*.

B & P as an Approximation to Population Sampling

- Intuitively, at s^*, the test set behavior is identical to that of doing (random) subsampling with subsamples containing one observation.
- Averaging the results of the latter (over a large B) is just a complicated way to compute an ***average*** — equivalent to stopping at s^*.

Let's recapitulate

- For the prediction function to be close to optimal without tuning it, we needed stuff past s^* to *efficiently* averages out to 0 in *the hold-out sample*.
- We also needed the estimated function before s^* to be protected against what comes next. We have both.
  - $\Rightarrow$ Immediate implication: there is no need to find s^* through cross-validation to obtain optimal predictions.
- Simulations will ask "How close to population sampling are we when fitting B & P trees?" and the answer will be "surprisingly close".

Not your Average Model Averaging

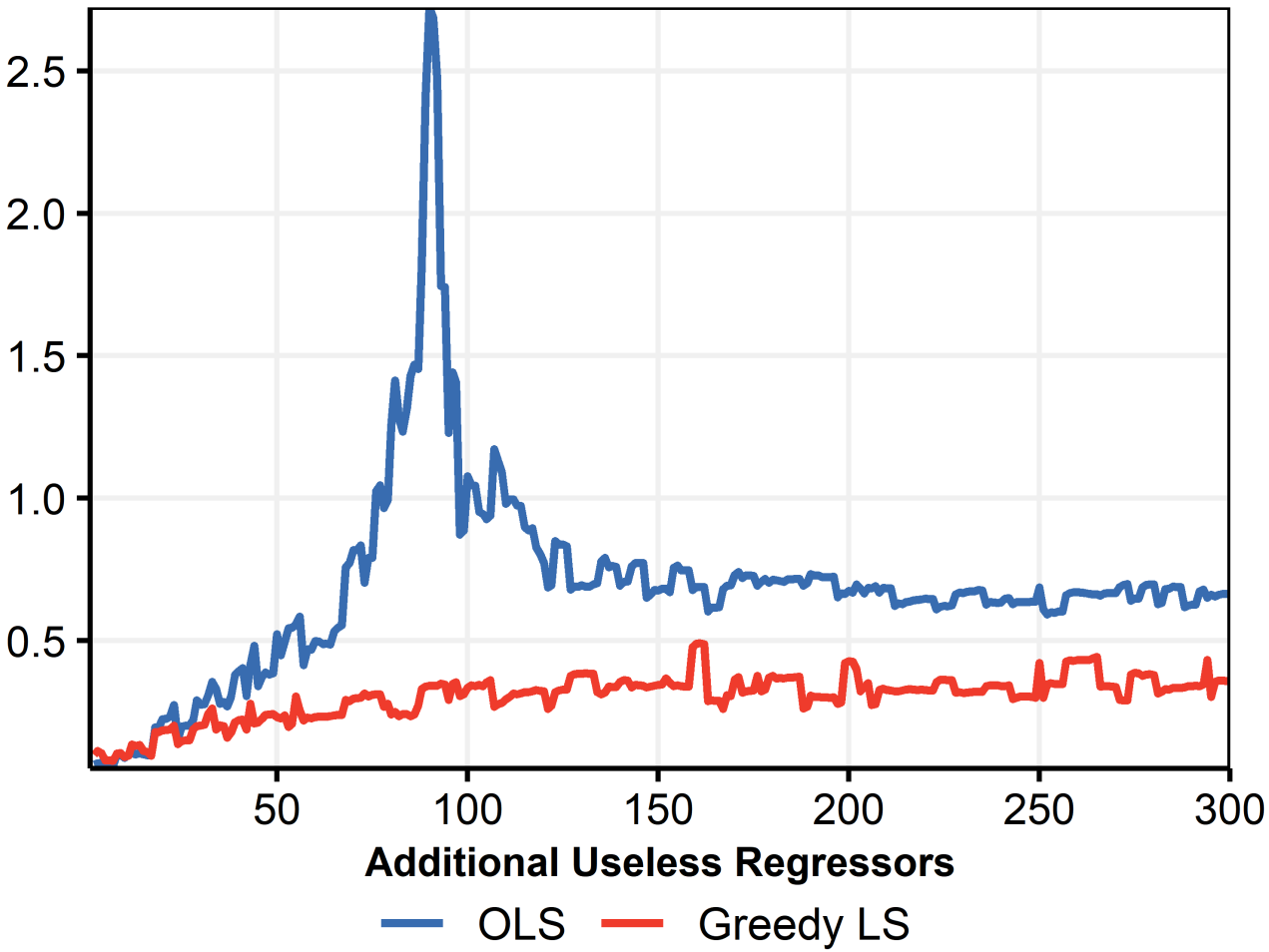


Figure: Model averaging/bagging different **linear** base learners with increasingly many useless features. Units are ln $\left(\text{MSE}_{\text{model}}/\text{MSE}_{\text{Oracle}}\right)$. Oracle has 10 regressors, SNR=2, and $N = 100$.

RF behaves very differently from Neural Networks

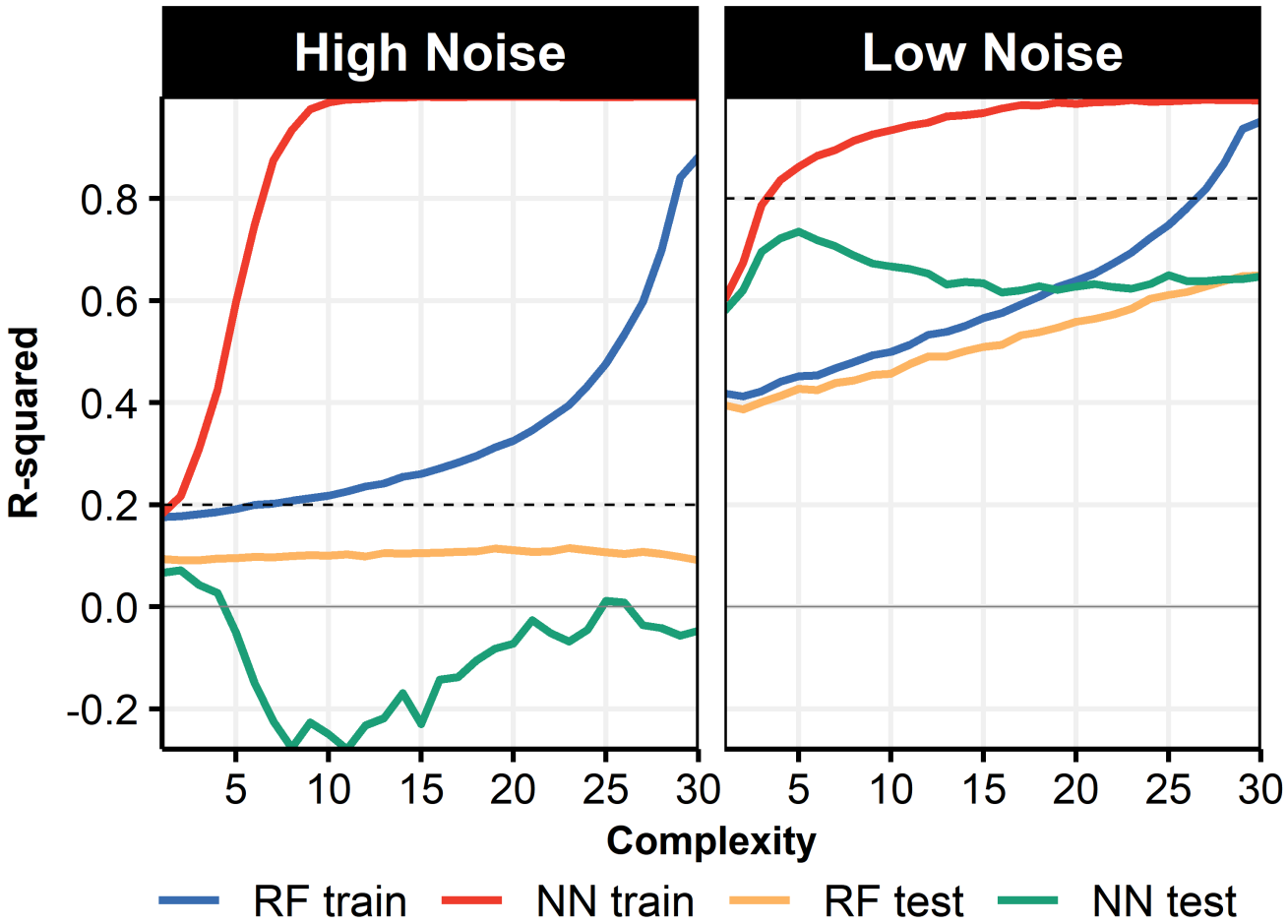


Figure: Dashed lines are true R^2. DGP is Friedman 1 (Friedman, 1991). The x-axis is an index of complexity/depth. For RF, it is a decreasing minimal size node from 200 to 1 in 30 steps, and for NN, an increasing number of layers from 1 to 30. The NN is 50 neurons wide and RF's $\texttt{mtry} = 1/3$.

Taking it Seriously: *Booging* and *MARSquake*

- The key ingredients for an ensemble to completely overfit in-sample while maintaining a stellar generalization error are
  - (i) the base learner prediction function is obtained by greedy/recursive optimization and
  - (ii) enough randomization in the fitting process.
- (i) means `B` & `P` variants of Boosted Trees and MARS are eligible for self-pruning.
- (ii) means its success depends on the capacity of the algorithm for randomization
- Tree constructions are completely irrevocable, additive structure are partly revocable (making (i) and (ii) maybe not as applicable as for RF)
- Let the data decide.

Simulations — Results

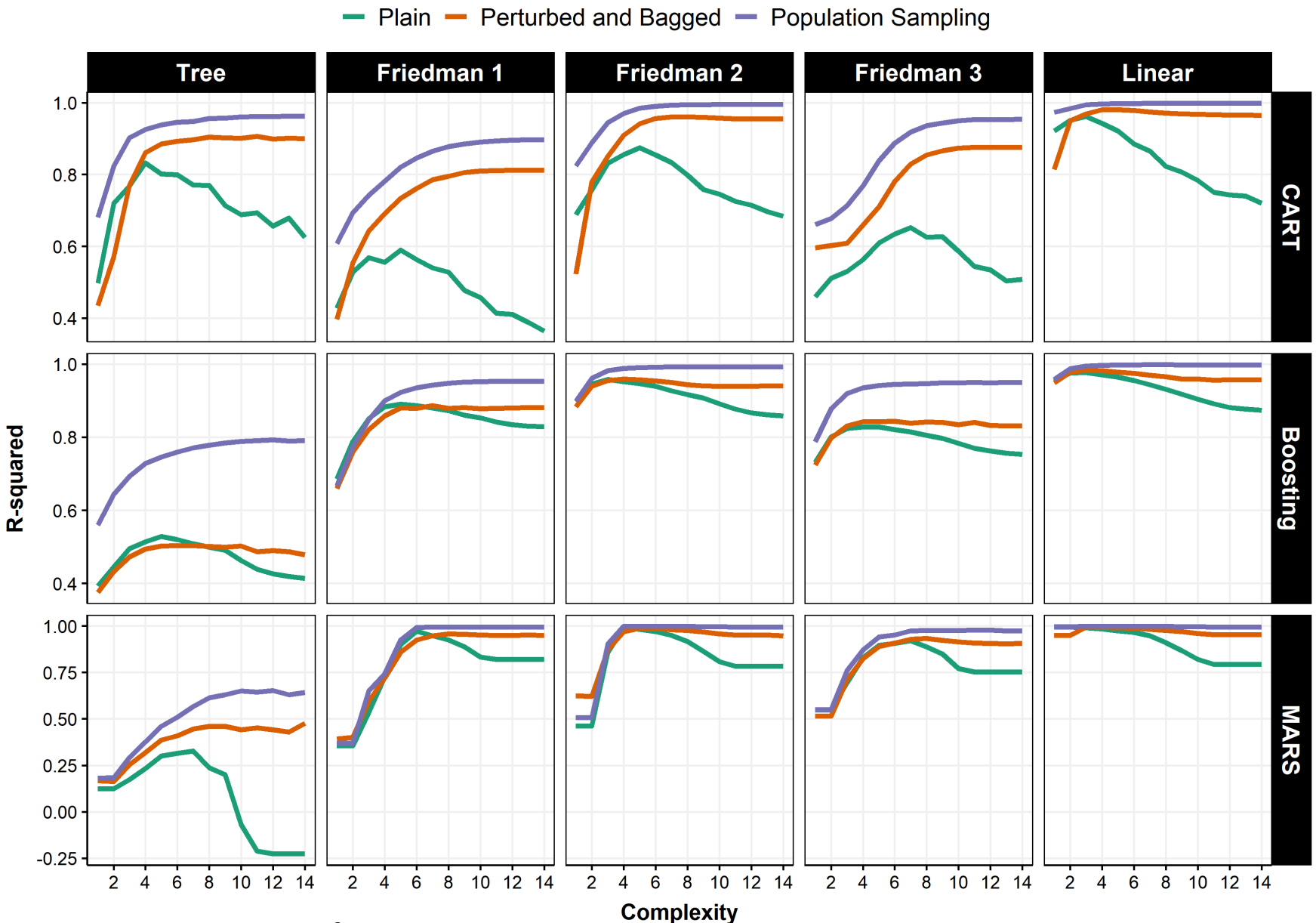
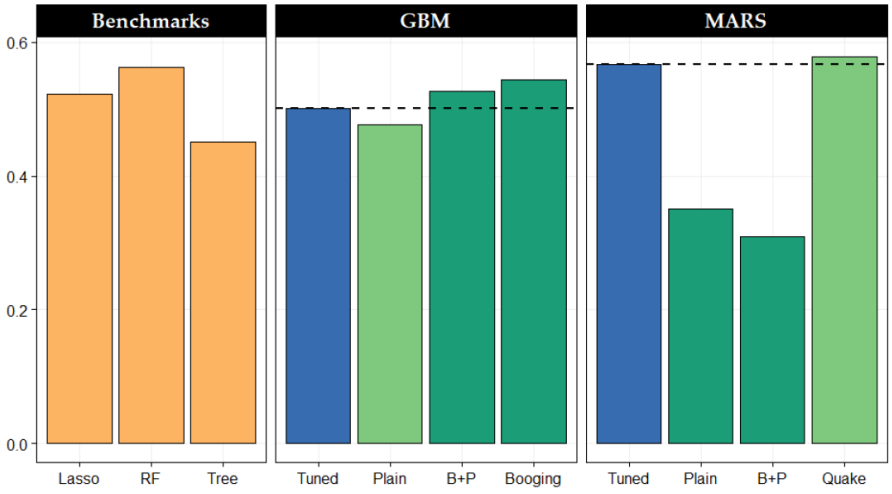
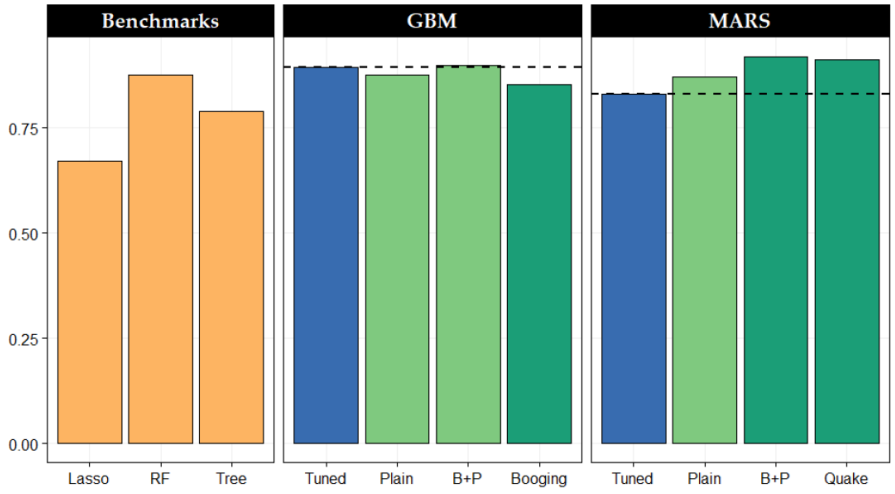


Figure: This plots hold-out sample R^2's between the prediction **and the true conditional mean**. The level of noise is calibrated so the signal-to-noise ratio is 4. Column facets are DGPs and row facets are base learners. The x-axis is an index of depth. For CART, it is a decreasing minimal size node $\in 1.4^{\{16,..,2\}}$, for Boosting, an increasing number of steps $\in 1.5^{\{4,..,18\}}$ and for MARS, it is an increasing number of included terms $\in 1.4^{\{2,..,16\}}$.

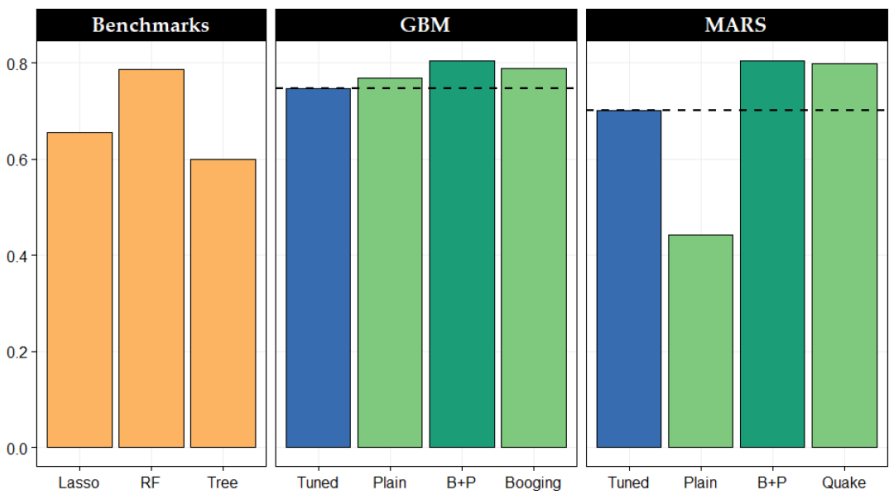
Real Data Results (1)



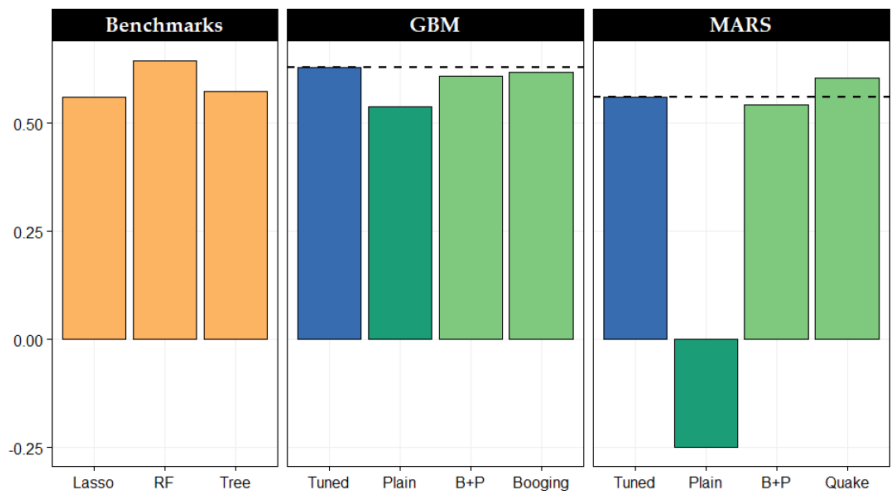
(a) *Abalone*



(b) *Boston Housing*



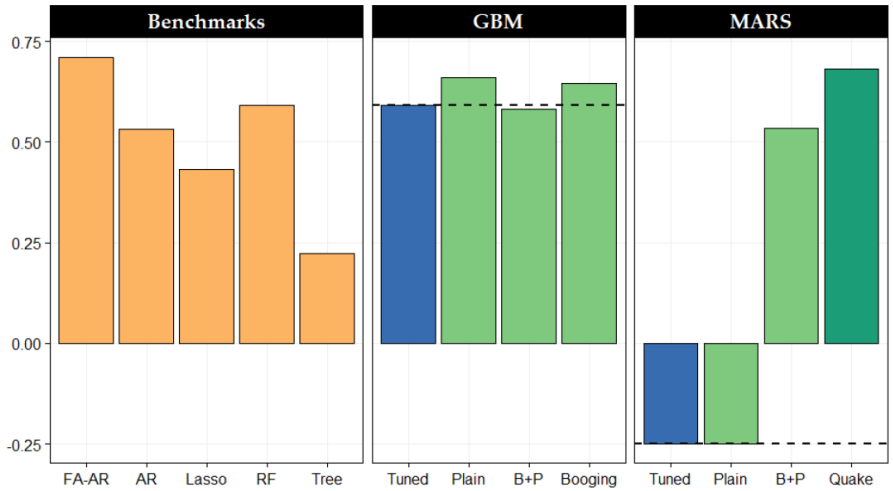
(c) *Crime Florida*



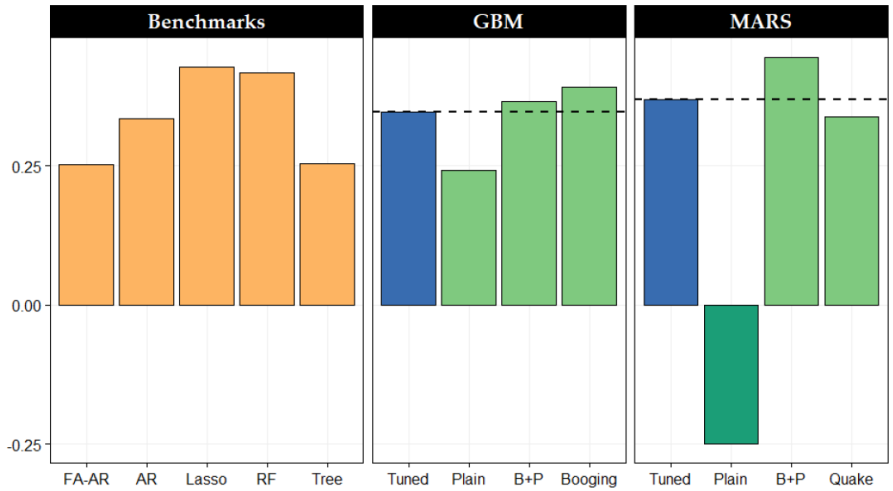
(d) *Fish Toxicity*

Figure: Performance metric is R^2_{test}. Darker green bars means the performance differential between the tuned version and the three others is significant at the 5% level.

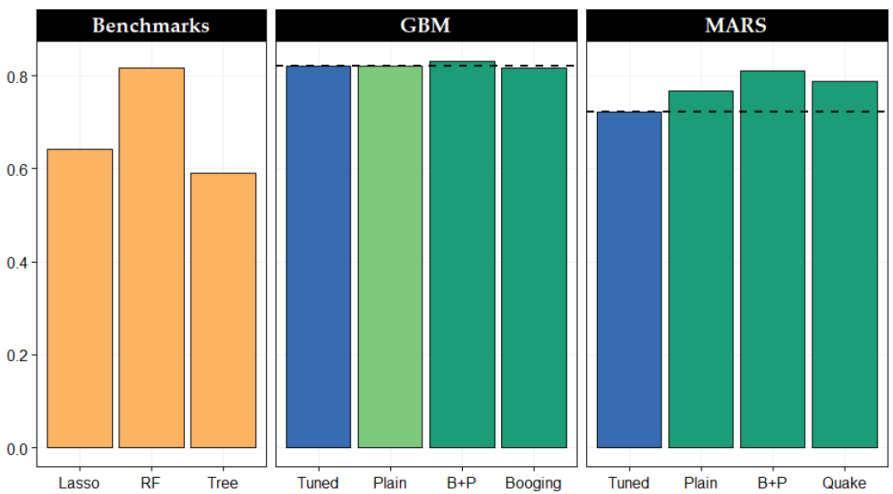
Real Data Results (2)



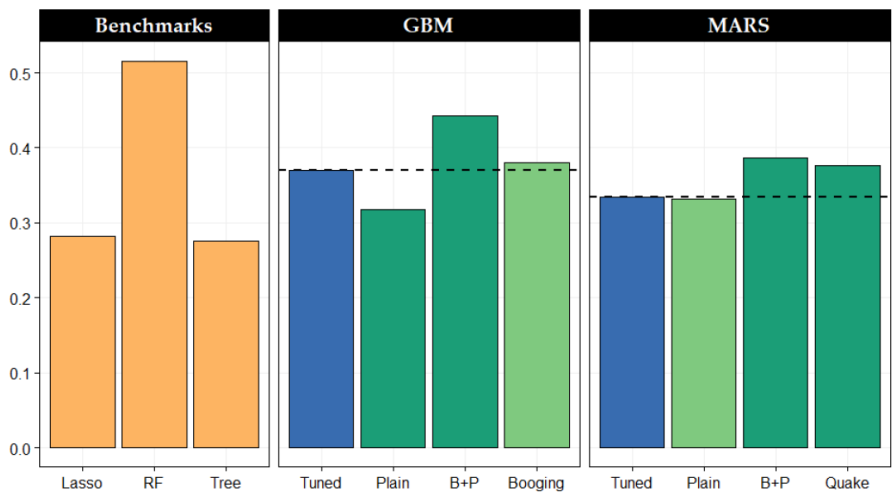
(e) *US Unemployment Rate* ($h = 1$)



(f) *US Inflation* ($h = 1$)



(g) *California Housing*



(h) *White Wine*

Figure: Performance metric is R^2_{test}. Darker green bars means the performance differential between the tuned version and the three others is significant at the 5% level.

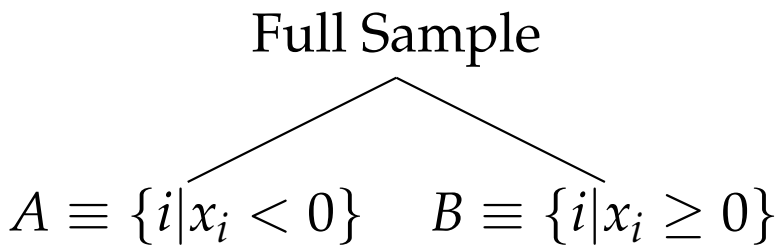
Filling the Missing Corner

Table: A Tree Ensemble Quaternity

Regularizer \ Model Structure	Additive Shallow Trees	One Deep Tree
Slow Learning	Boosting	**Slow-Growing Tree**
B & P	Booging	Random Forest

Implementation

Consider the deceptively simple tree below, which is obtained after one recursion of CART.



The subsequent problem of finding k_A^* and c_A^* to further grow the tree on the A side is

$$\min_{k \in \mathcal{K}, c \in \mathbb{R}} \left[\min_{\mu_1} \sum_{\{i|X_i^k \leq c\}} \omega_i^A \left(y_i - \mu_1\right)^2 + \min_{\mu_2} \sum_{\{i|X_i^k > c\}} \omega_i^A \left(y_i - \mu_2\right)^2 \right] \tag{3}$$

where $\omega_i = I(i \in A)$. This can be generalized to

$$\omega_i = I(i \in A) + (1 - \eta) I(i \in B)$$

where $\eta \in (0,1]$ is a learning rate. ω_i's collapse to CART when $\eta = 1$.

SGT Algorithm

Algorithm 1 Slow-Growing Tree

Input: Training data $[y_i \;\; \boldsymbol{X}_i]$, test set predictors $\boldsymbol{X}_j$, learning rate $\eta \in (0, 1]$, maximal Gini coefficient $\bar{G}$

Initialize $\omega_i^0 = 1 \;\; \forall i$.

for l's such that $G_l < \bar{G}$ **do**

$$(k_l^*, c_l^*) = \underset{k \in \mathcal{K}, c \in \mathbb{R}}{\arg\min} \left[\min_{\mu_1} \sum_{\{i | X_i^k \leq c\}} \omega_i^l \, (y_i - \mu_1)^2 + \min_{\mu_2} \sum_{\{i | X_i^k > c\}} \omega_i^l \, (y_i - \mu_2)^2 \right]$$

Create 2 children nodes with $\omega_i^{l,+} = \omega_i^l (1 - \eta I(X_i^{k_l^*} \leq c_l^*))$ and $\omega_i^{l,-} = \omega_i^l (1 - \eta I(X_i^{k_l^*} > c_l^*))$.

end for

Return: $\hat{y}_j = \sum_{l=1}^{L} \left(w_j^l(\boldsymbol{X}_j) \sum_{i=1}^{N} \omega_i^l y_i \right)$ where $w_j^l(\boldsymbol{X}_j) = \dfrac{\omega_j^l(\boldsymbol{X}_j)}{\sum_{l=1}^{L} \omega_j^l(\boldsymbol{X}_j)}$

Simulations

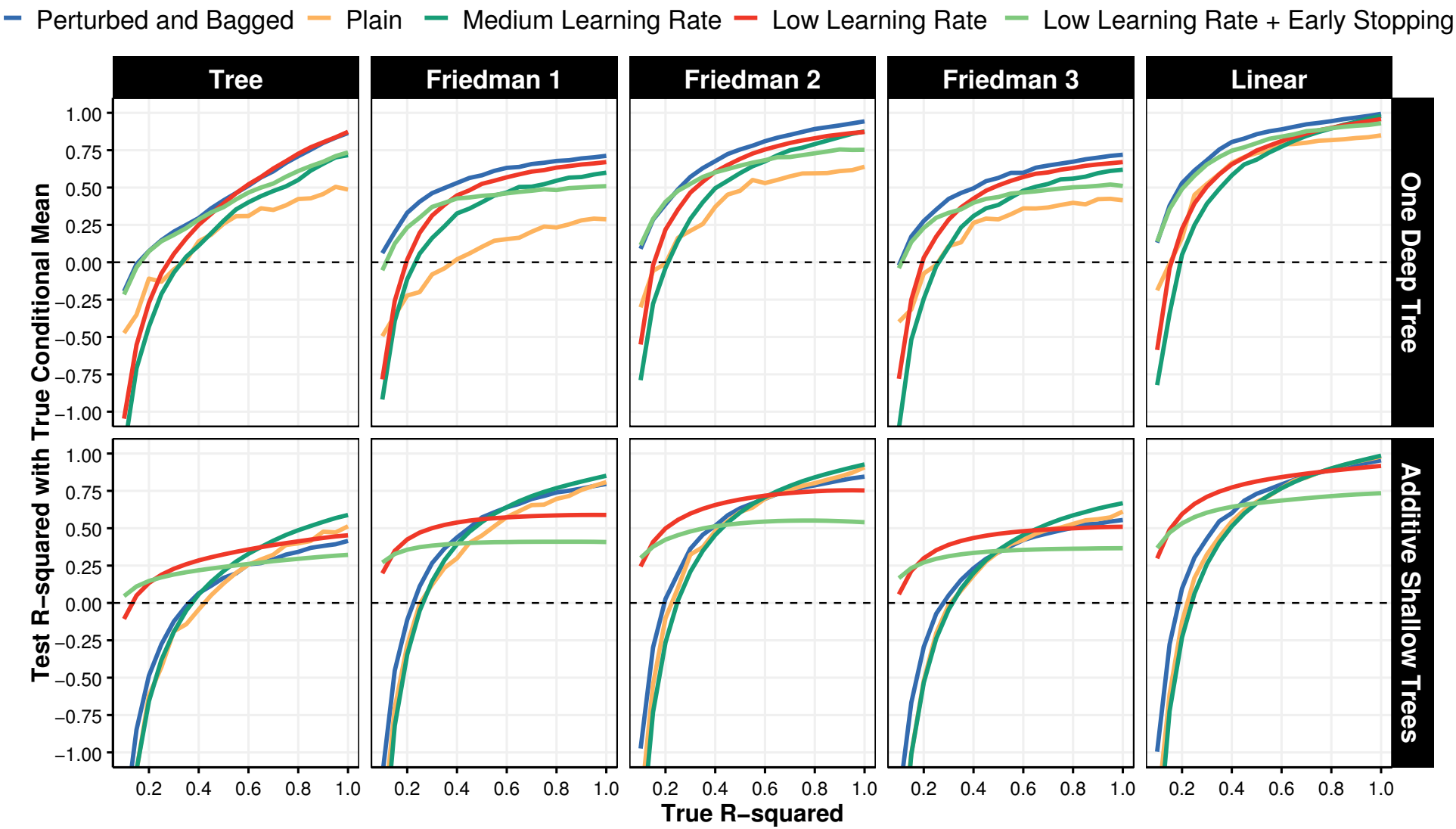


Figure: This plots the hold-out sample R^2 between the prediction **and the true conditional mean**. The level of noise is decreasing along the x-axis. Column facets are DGPs and row facets are "models". The y-axis is cut at -1 to favor readability because a few models go largely below it for the lowest true R^2 case.

Conclusion

1. `B` & `P` as implemented by RF automatically *prune* a (latent) true underlying tree.
2. This gives rise to the R^2_{test} vs R^2_{train} puzzle, which traditional explanations do not account for
3. More generally, there is no need to tune the stopping point of a properly randomized ensemble of greedily optimized base learners.
4. Boosting and MARS are also eligible for automatic (implicit) tuning.
5. Stabilizing the greedy algorithm can also be done with slow-learning (the traditional boosting way) and is developed in (Goulet Coulombe, 2021) where a single "Slow-Growing" Tree can match RF.

Based of the following papers:

- To Bag is to Prune
- Slow-Growing Trees