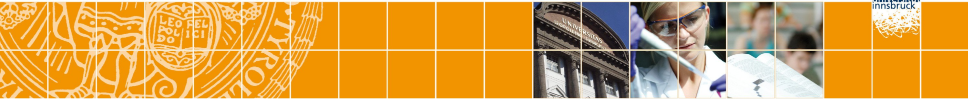
Universität Innsbruck

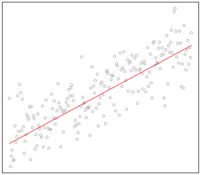


Distributional Trees and Forests

Lisa Schlosser, Torsten Hothorn, Achim Zeileis

`https://R-Forge.R-project.org/projects/partykit/`

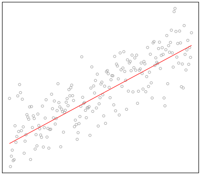
Motivation



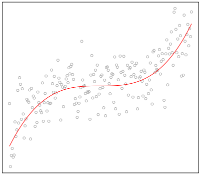
LM, GLM

`lm`
`glm`

Motivation



→



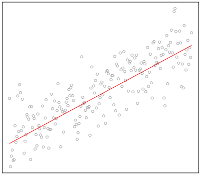
LM, GLM

`lm`
`glm`

GAM

`mgcv`
`VGAM`
...

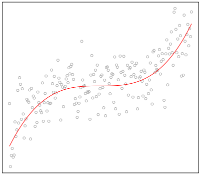
Motivation



LM, GLM

`lm`
`glm`

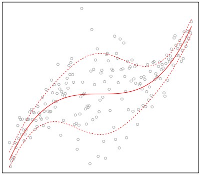
→



GAM

`mgcv`
`VGAM`
`...`

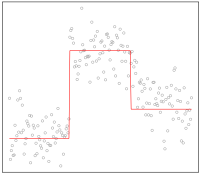
→



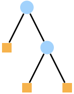
GAMLSS

`gamlss`
`mgcv`
`VGAM`
`gamboostLSS`
`...`

Motivation

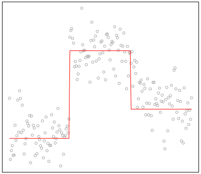


Regression Tree

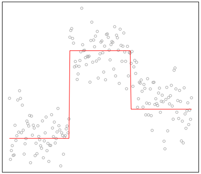


`rpart`
`party(kit)`

Motivation

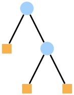


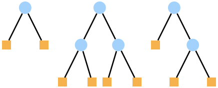
→



Regression Tree

Random Forest

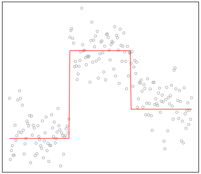




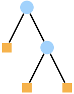
`rpart`
`party(kit)`

`randomForest`
`ranger`
`party(kit)`
...

Motivation

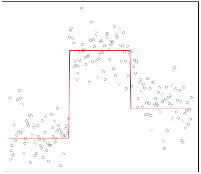


Regression Tree

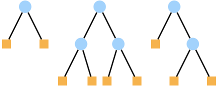


`rpart`
`party(kit)`

→



Random Forest



`randomForest`
`ranger`
`party(kit)`
...

→

?

Distributional Trees and Forests

`disttree`
based on
`partykit`

Goals

Tree:

- Specify the complete distribution in each subgroup.
  (location, scale and shape)
- Automatic detection of steps and abrupt changes.
- Capture non-linear and non-additive effects and interactions.

Forest:

- Smoother effects.
- Stabelization of the model.

Building Distributional Trees and Forests

Tree:

1. Specify a distribution with log-likelihood function $\ell(\theta; y)$.
2. Estimate $\hat{\theta}$ via maximum likelihood.
3. Test for associations or instabilities of the scores $\frac{\partial \ell}{\partial \theta}(\hat{\theta}; y_i)$ and each partitioning variable x_i.
4. Split the sample along the partitioning variable with the strongest association or instability. Choose breakpoint with highest improvement in log-likelihood.
5. Repeat steps 2–4 recursively in the subgroups until some stopping criterion is met.

Forest: Ensemble of trees.

- Bootstrap or subsamples.
- Random input variable sampling.

Prediction

Tree:

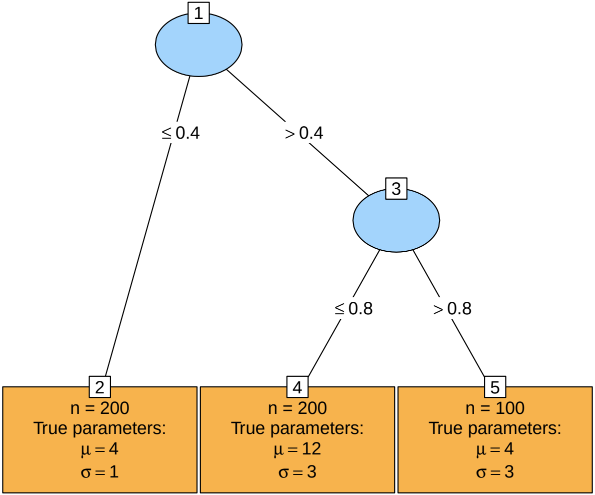
Estimate $\hat{\theta}$ on the subsample of the learning data which ends up in the same terminal node as the new observation.

Forest:

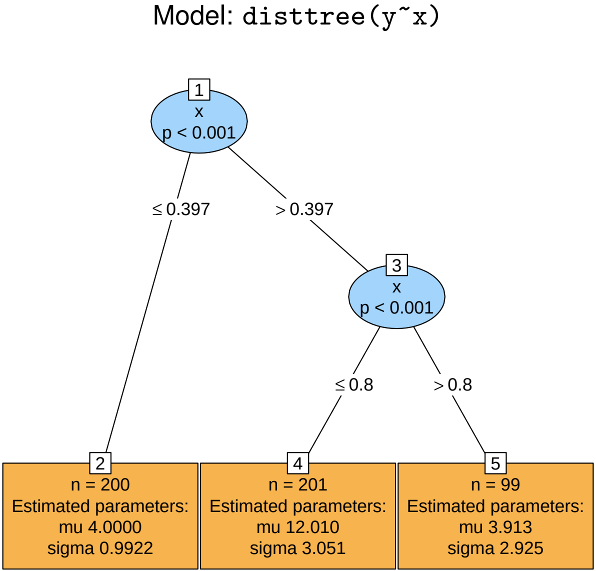
Estimate $\hat{\theta}$ on the whole learning data but weighted by the number of trees in which a learning observation ends up in the same terminal node as the new observation.

Fitting a Tree

DGP: $Y \sim \mathcal{N}(\mu(X), \sigma(X))$

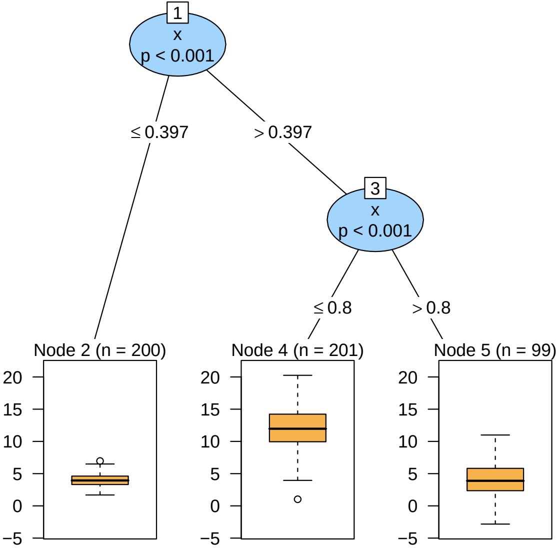


Fitting a Tree

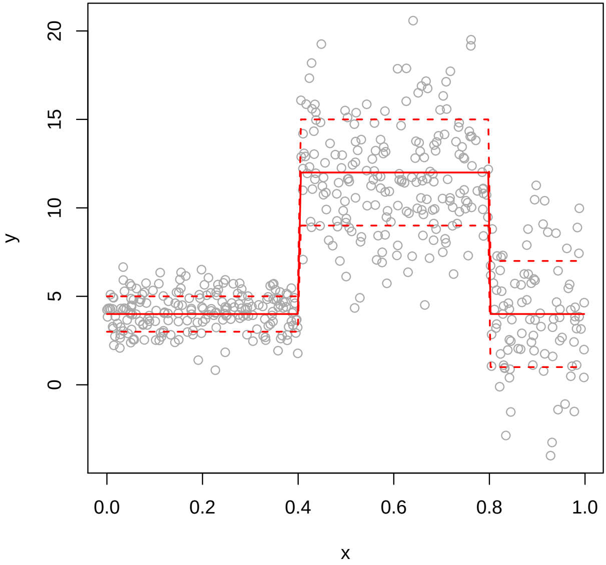


Fitting a Tree

Model: `disttree(y~x)`

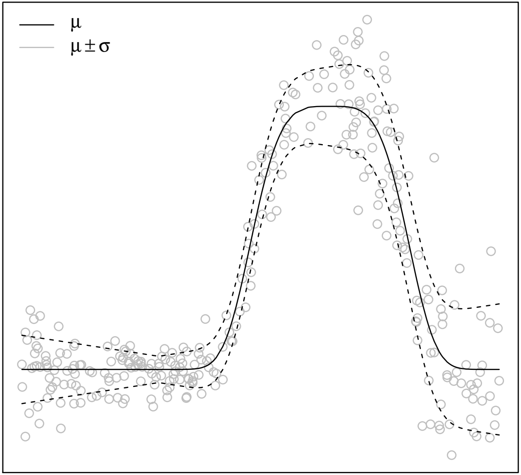


Fitting a Tree

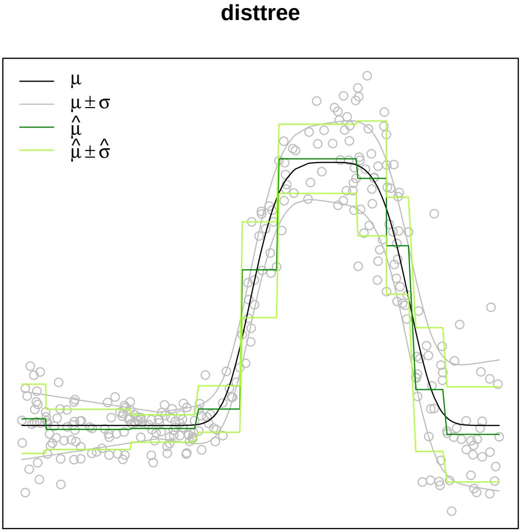


Simulation

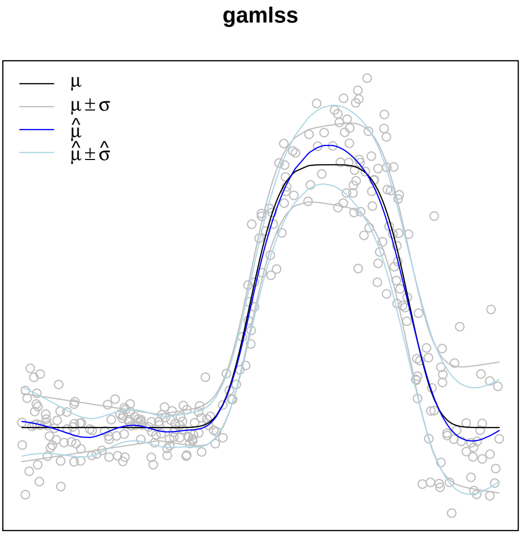
true parameters



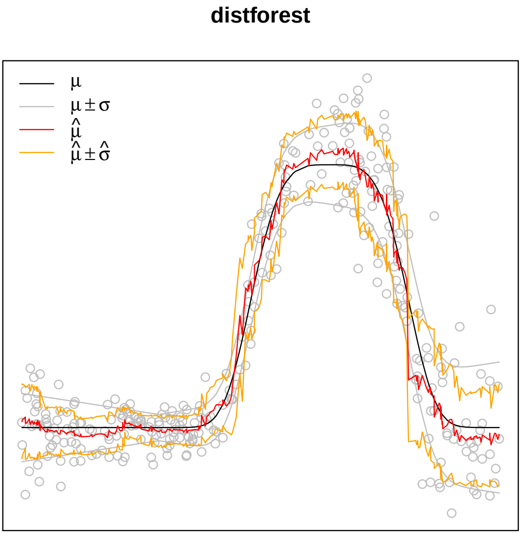
Simulation



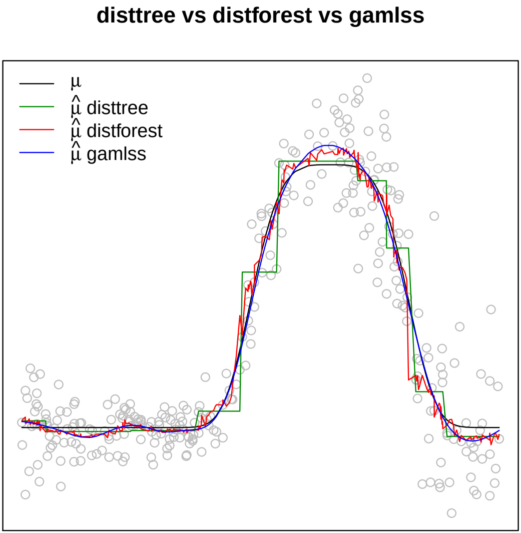
Simulation



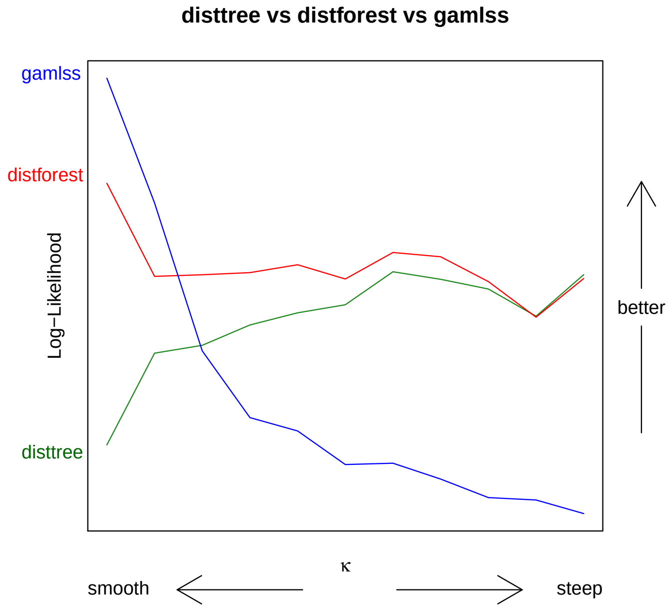
Simulation



Simulation


disttree vs distforest vs gamlss
μ
$\hat{\mu}$ disttree
$\hat{\mu}$ distforest
$\hat{\mu}$ gamlss

Simulation


disttree vs distforest vs gamlss
gamlss
distforest
disttree
Log−Likelihood
better
κ
smooth
steep

Software

R-package **disttree** available on R-Forge:

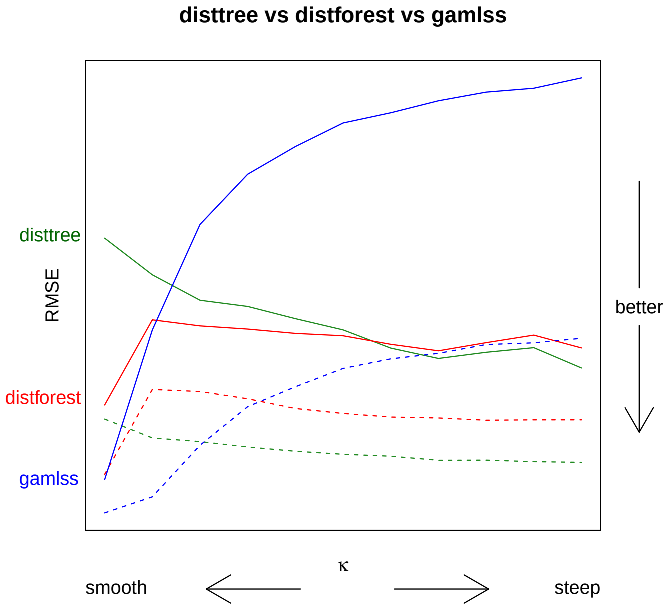
`https://R-Forge.R-project.org/projects/partykit/` Main

functions:

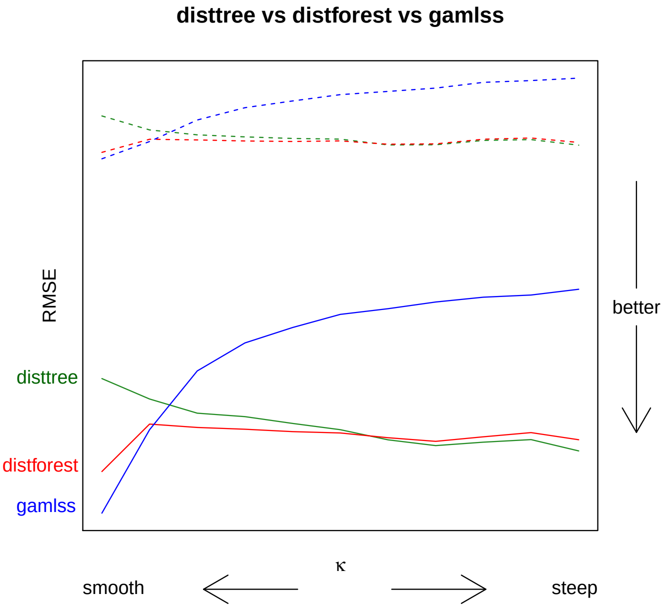
- `distfit()`
- `disttree()`
- `distforest()`

References

Simulation


disttree vs distforest vs gamlss
disttree
distforest
gamlss
RMSE
better
κ
smooth
steep

Simulation


disttree vs distforest vs gamlss
RMSE
disttree
distforest
gamlss
better
κ
smooth
steep