Data-driven methods for predictive modelling

D G Rossiter

Cornell University, Soil & Crop Sciences Section

Nanjing Normal University, Geographic Sciences Department
南京师范大学地理学学院

April 9, 2020

Statistical modelling

- Statistics starts with **data**: something we have measured
- Data is **generated** by some (unknown) **mechanism**: input (stimulus) x, output (response) y
- Before analysis this is a **black box** to us, we only have the data itself
- **Two goals** of analysis:
  1. **Prediction** of future responses, given known inputs
  2. **Explanation**, **Understanding** of what is in the “black box” (i.e., make it “white” or at least “some shade of grey”).

Modelling cultures

Data modelling (also called "**model-based**")

- *assume* an empirical-statistical (stochastic) data model for the inside of the black box, e.g., a functional form such as multiple linear, exponential, hierarchical ...
- *parameterize* the model from the data
- *evaluate* the model using model diagnostics

Algorithmic modelling (also called "**data-driven**")

- *find* an algorithm that produces y given x
- *evaluate* by **predictive** accuracy (note: *not* internal accuracy)

Reference: Breiman, L. (2001). *Statistical Modeling: The Two Cultures* (with comments and a rejoinder by the author). **Statistical Science**, 16(3), 199–231. https://doi.org/10.1214/ss/1009213726

Explanation vs. prediction

- **Explanation**
  - Testing a **causal theory** – why are things the way they are?
  - Emphasis is on **correct model specification** and **coefficient estimation**
  - Uses **conceptual** variables based on theory, which are represented by **measureable** variables
- **Prediction**
  - Predicting **new** (space, members of population) or **future** (time) **observations**.
  - Uses **measureable** variables only, no need for concepts

Reference: Shmueli, G. (2010). *To Explain or to Predict?* **Statistical Science**, 25(3), 289–310. `https://doi.org/10.1214/10-STS330`

Bias/variance tradeoff

The expected prediction error (EPE) for a new observation with value x is:

$$
\begin{aligned}
\mathrm{EPE} &= \mathrm{E}\{Y - \hat{f}(x)\}^2 \\
&= \mathrm{E}\{Y - f(x)\}^2 + \{\mathrm{E}(\hat{f}(x)) - f(x)\}^2 \\
&\quad + \mathrm{E}\{\hat{f}(x) - \mathrm{E}(\hat{f}(x))\}^2 \\
&= \mathrm{Var}(Y) + \mathrm{Bias}^2 + \mathrm{Var}(\hat{f}(x))
\end{aligned}
$$

Model variance: residual error with perfect model specification (i.e., noise in the relation)

Bias: mis-specification of the statistical model: $\hat{f}(x) \neq f(x)$

Estimation variance: the result of using a sample to estimate f as $\hat{f}(x)$

Bias/variance tradeoff: explanation vs. prediction

Explanation **Bias** should be minimized

- correct model specification and correct coefficients → correct conclusions about the theory (e.g., causal relation)

Prediction **Total EPE** should be minimized.

- accept some bias if that reduces the estimation variance
- a simpler model (omitting less important predictors) often has better fit to the data

When does an underspecified model better predict than a full model?

- the data are very noisy (large σ);
- the true absolute values of the left-out parameters are small;
- the **predictors are highly correlated**; and
- the sample size is small or the range of left-out variables is narrow.

Problems with data modelling

- Mosteller and Tukey(1977): "The whole area of guided regression [an example of, model-based inference] is fraught with intellectual, statistical, computational, and subject matter difficulties."
- It seems we understand nature if we fit a model form, but in fact our conclusions are about the **model's** mechanism, and not necessarily about **nature's** mechanism.
- So, if the model is a poor emulation of nature, the conclusions about nature may be wrong ...
- ... and of course the predictions may be wrong – we are incorrectly **extrapolating**.

The philosophy of data-driven methods

- Also called "statistical learning", "machine learning"
- Build structures to represent the "black box" *without* using a statistical model
- Model quality is evaluated by **predictive accuracy on test sets** covering the target population
  - **cross-validation** methods can use (part of) the original data set if an independent set is not available

Some data-driven methods

1. Covered in this lecture
   - Classification & Regression Trees (CART) 分类与回归树
   - Random Forests (RF) 随 机森林
   - Cubist
2. Others
   - Artificial Neural Networks (ANN) 人工神经网络
   - Support Vector Machines
   - Gradient Boosting

Key references – texts

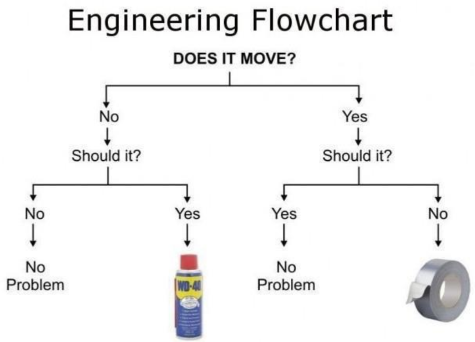
- Hastie, T., Tibshirani, R., & Friedman J. H. (2009). The elements of statistical learning data mining, inference, and prediction (2nd ed). New York: Springer. https://doi.org/10.1007%2F978-0-387-84858-7
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning: with applications in R. New York: Springer. https://doi.org/10.1007%2F978-1-4614-7138-7
- Statistical Learning on-line course (based on James et al. book): https://lagunita.stanford.edu/courses/HumanitiesSciences/StatLearning/Winter2016/about
- Kuhn, M., & Johnson, K. (2013). Applied Predictive Modeling (2013 edition). New York: Springer. https://doi.org/10.1007/978-1-4614-6849-3

Key references – papers

- Shmueli, G. (2010). *To Explain or to Predict?* **Statistical Science**, 25(3), 289–310. https://doi.org/10.1214/10-STS330
- Breiman, L. (2001). *Statistical Modeling: The Two Cultures* (with comments and a rejoinder by the author). **Statistical Science**, 16(3), 199–231. https://doi.org/10.1214/ss/1009213726
- Breiman, L. (2001). Random forests. **Machine Learning**, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324
- Kuhn, M. (2008). *Building Predictive Models in R Using the caret Package*. **Journal of Statistical Software**, 28(5), 1–26.

Decision trees 决策树

- Typical uses in diagnostics (medical, automotive . . .)
- Begin with the full set of possible decisions
- Split into two (*binary*) subsets based on the values of some **decision criterion**
- Each branch has a more limited set of decisions, or at least has more information to help make a decision
- Continue **recursively** on both branches until there is enough information to make a decision



Classification & Regression Trees 分类与回归树

- A type of decision tree; decision is “what is the predicted response, given values of predictors”?
- Aim is to predict the **response** (target) variable from one or more **predictor** variables
- If *response* is **categorical** (class, factor) we build a **classification tree**
- If *response* is **continuous** we build a **regression tree**
- *Predictors* can be any combination of categorical or continuous

Advantages of CART

- A simple model, **no statistical assumptions** other than between/within class variance to decide on splits
  - For example, no assumptions of the distribution of residuals
  - So can deal with non-linear and threshold relations
- No need to transform predictors or response variable
- **Predictive power** is quantified by **cross-validation**; this also controls **complexity** to avoid **over-fitting**

Disadvantages of CART

- No model to interpret (although we can see variable importance)
- Predictive power over a **population** depends on a **sample** that is **representative** of that population
- Quite **sensitive** to the **sample**, even when pruned
- Pruning to a complexity parameter depends on 10-fold cross-validation, which is sensitive to the choice of observations in each fold
- Typically makes only a small number of different predictions (“boxes”), so maps made with it show **discontinuities** (“jumps”)

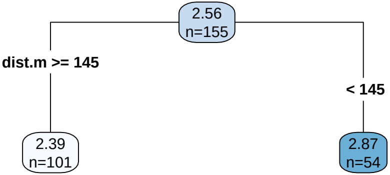
Tree terminology

- **splitting variable** variable to examine, to decide which branch of the tree to follow
- **root node** 根部节点 variable used for first split; overall mean and total number of observations
- **interior node** 非叶子节点 splitting variable, value on which to split, mean and number to be split
- **leaf** 叶子点 predicted value, number of observations contributing to it
- **cutpoint** of the splitting variable: value used to decide which branch to follow
- **growing** the tree
- **pruning** the tree

Example regression tree

- Meuse River soil heavy metals dataset
- **Response** variable: log(Zn) concentration in topsoil
- **Predictor** variables
  1. distance to Meuse river (continuous)
  2. elevation above sea level (continuous)
  3. flood frequency class (categorical, 3 classes)

Example regression tree – first split



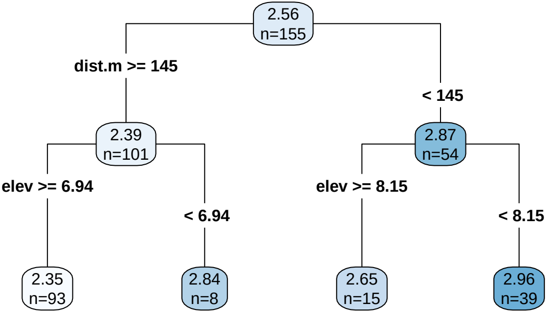
Splitting variable: distance to river

Is the point closer or further than 145 m from the river? 101 points *yes*, 54 points *no*.

Explanation of first split

- **root**: average log(Zn) of whole dataset 2.56 log(mg kg^{-1}) fine soil; based on all 155 observations
- **splitting variable at root**: distance to river
- **cutpoint at root**: 145 m
- **leaves**
  - distance $<$ 145 m: 54 observations, their mean is 2.87 log(mg kg^{-1})
  - distance $\geq$ 145 m: 101 observations, their mean is 2.39 log(mg kg^{-1})
  - full dataset has been *split* into two *more homogeneous* subsets

Example regression tree – second split



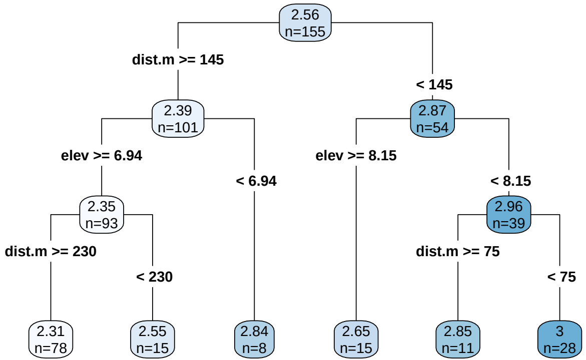
For both branches, what is the elevation of the point?

Note: this is a coincidence in this case, different splitting variables can be used on different branches.

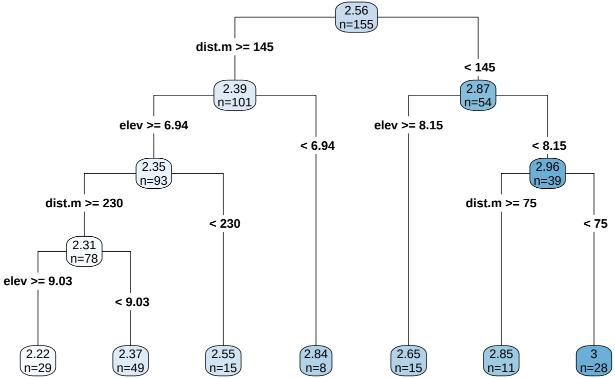
Explanation of second split

- **interior nodes** were **leaves** after the first split, now 'roots' of subtrees
  - *left*: distance $\geq$ 145 m: 101 observations, their mean is 2.39 log(mg kg^{-1}) – note smaller mean on left
  - *right*: distance $<$ 145 m: 54 observations, their mean is 2.87 log(mg kg^{-1})
- **splitting variable at interior node** for $<$ 145 m: elevation
- **cutpoint at interior node** for $<$ 145 m: 8.15 m.a.s.l.
- **splitting variable at interior node** for $\geq$ 145 m: elevation
- **cutpoint at interior node** for $\geq$ 145 m: 6.95 m.a.s.l.
- **leaves** 93, 8, 15, 39 observations; means 2.35, 2.84, 2.65, 2.96 log(mg kg^{-1})
- These leaves are now more homogeneous than the interior nodes.

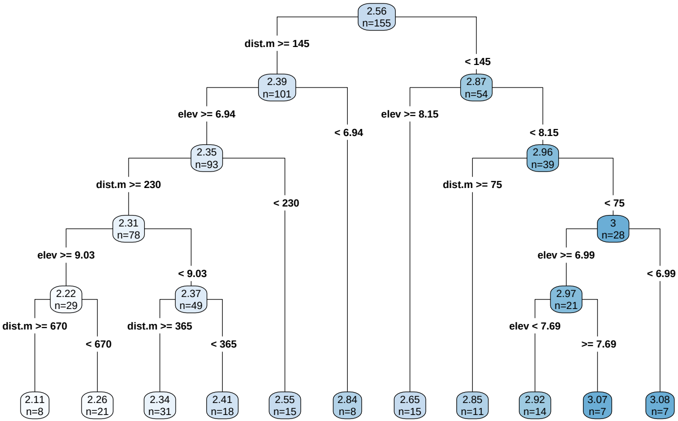
Example regression tree – third split



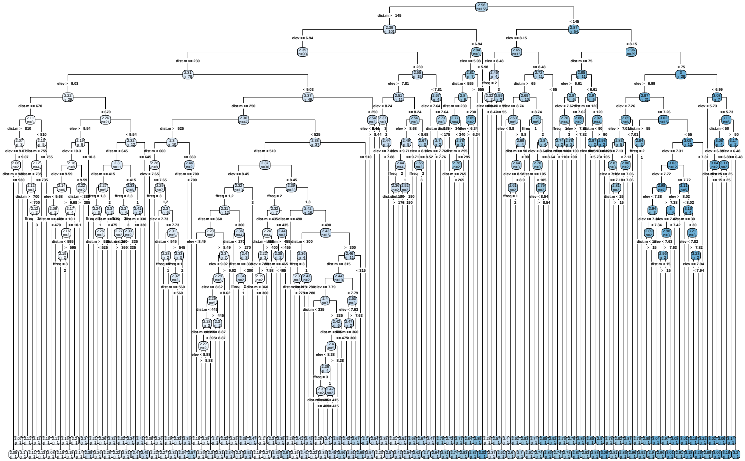
Example regression tree – fourth split



Example regression tree – fifth split



Example regression tree – maximum possible splits



How are splits decided?

1. Take all possible *predictors* and all possible *cutpoints*
2. Split the data(sub)set at *all combinations*
3. Compute some **measure of discrimination** for all these – i.e., a measure which determine which split is "best"
4. Select the predictor/split that most discriminates

Criteria for **continuous** and **categorical** response variables: see next slides

How are splits decided? – Continuous response

Select the predictor/split that most increases *between-class* variance (this decreases *pooled within-class* variance):

$$\sum_{\ell} \sum_{i} (y_{\ell,i} - \overline{y_l})^2$$

- $y_{\ell,i}$ value i of the target in leaf ℓ
- $\overline{y_l}$ is the mean value of the target in leaf ℓ

So the set of leaves are **more homogeneous**, on average, than the root.

How are splits decided? – Categorical response

Select the predictor/split that minimizes the *impurity* of the set of leaves:

- Misclassification rate: $\frac{1}{N_m} \sum_{i \in R} I(y_i \neq k(m))$
  - N_m: number of observations at node m
  - R_m: the set of observations
  - $k(m)$ is the majority class; I is the logical T/F function
- Impurity is maximal when all classes have same frequency, and minimal when only one class has any observations in the leaf

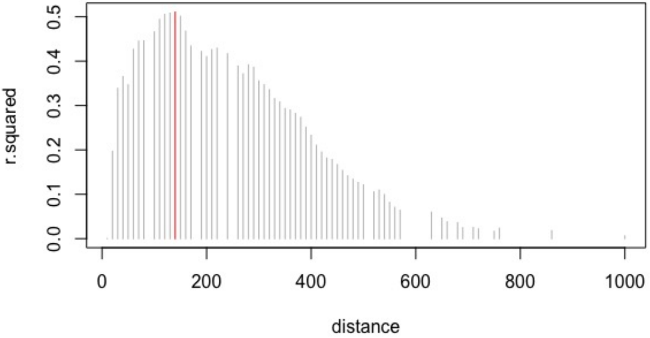
So the set of leaves are purer (less confusion), on average, than the root.

Example split (1)

```
> # all the possible cutpoints for distance to river
> (distances <- sort(unique(meuse$dist.m)))
 [1]   10   20   30   40   50   60   70   80  100  110  120  130  140  15
[15]  160  170  190  200  210  220  240  260  270  280  290  300  310  32
[29]  330  340  350  360  370  380  390  400  410  420  430  440  450  46
[43]  470  480  490  500  520  530  540  550  560  570  630  650  660  68
[57]  690  710  720  750  760  860 1000
> for (i in 1:nd) {  # try them all
  branch.less <- meuse$zinc[meuse$dist.m < distances[i]]
  branch.more <- meuse$zinc[meuse$dist.m >= distances[i]]
  rss.less <- sum((branch.less-mean(branch.less))^2)
  rss.more <- sum((branch.more-mean(branch.more))^2)
  rss <- sum(rss.less + rss.more)
  results.df[i,2:5] <- c(rss.less, rss.more, rss, 1-rss/tss)
  }
> # find the best split
> ix.r.squared.max <- which.max(results.df$r.squared)
print(results.df[ix.r.squared.max,])
> print(results.df[ix.r.squared.max,])
   distance rss.less rss.more      rss r.squared
13      140  7127795  3030296 10158091  0.510464
> # plot the results
plot(r.squared ~ distance, data=results.df, type="h",
     col=ifelse(distance==d.threshold,"red","gray"))
```

Example split (2): R^2 vs. cutpoint – distance to river

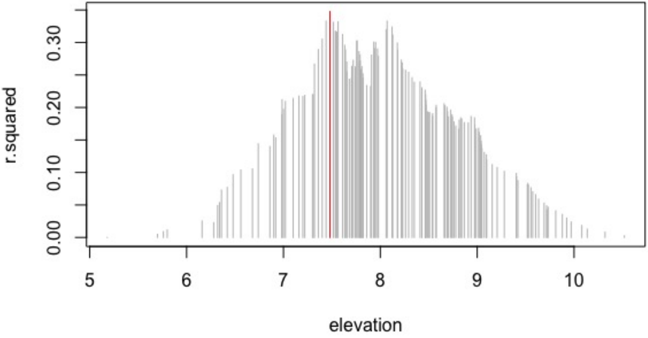
Try to split the **root node** on this predictor:



Best cutpoint is 140 m; this explains 51% of the total variance

Example split (3): R^2 vs. cutpoint – elevation

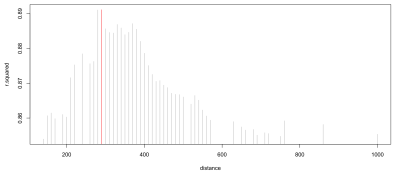
Try to split the **root node** on this predictor:

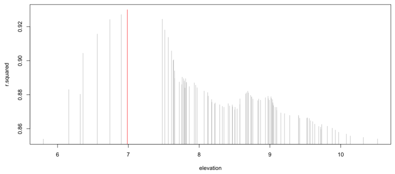


Best cutpoint is 7.48 m.a.s.l.; this only explains 35% of the total variance; so use the distance to river as the first split

Example split (4a): left first-level leaf

Try to split the **left first-level leaf** (101 observations):



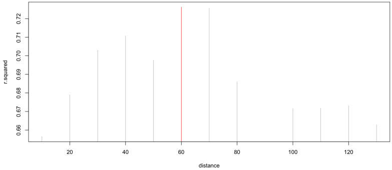


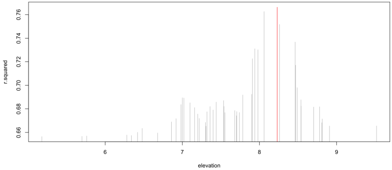
Best cutpoint is 6.99 m.a.s.l.; this explains 93.0% of the variance *in this group*. Splitting at 290 m distance would explain 89.1%.

So split this leaf on *elevation* – it becomes an *interior node*

Example split (4b): right first-level leaf

Try to split the **right first-level leaf** (54 observations):





Best cutpoint is 8.23 m.a.s.l.; this explains 76.6% of the variance *in this group*. Splitting at 60 m distance would explain 72.6%.

So split on *elevation* – it becomes an *interior node*.

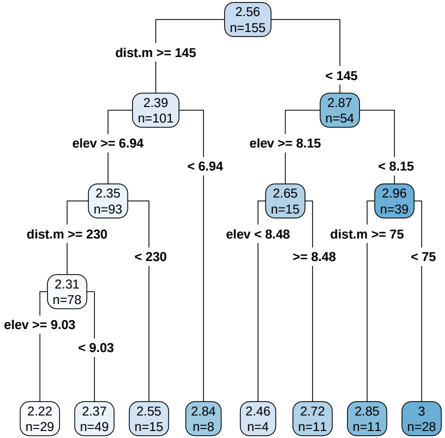
Controlling tree complexity

- Fitting a full tree, until there is only one observation per leaf, is always **over-fitting** to the sample set, and will not be a good **predictor** of the population.
- A full tree fits some **noise** as well as **structure**.
- Can control by the **analyst** or automatically by **pruning** (see below).
- Analyst can specify:
  - Minimum number of observations in a leaf (fewer: no split is attempted): `minsplit`
  - Maximum depth of tree: `maxdepth`
  - Minimum improvement in pooled within-class vs. between-class variance: `cp` (see below)

Predicting with the fitted tree

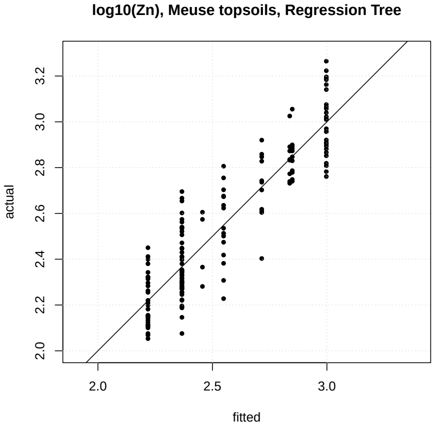
- A simple ‘model’ is applied to each leaf:
  - Response variable continuous numeric: mean of observed data in leaf
  - Categorical variable: most frequent category in leaf
- Value at new location is predicted by running the covariate data down the tree

Fitted regression tree



Question: What is the predicted value for a point 100 m from the river and 9 m.a.s.l. elevation?

Predictions at known points



Note only one prediction per leaf, applies to all points falling in the leaf.

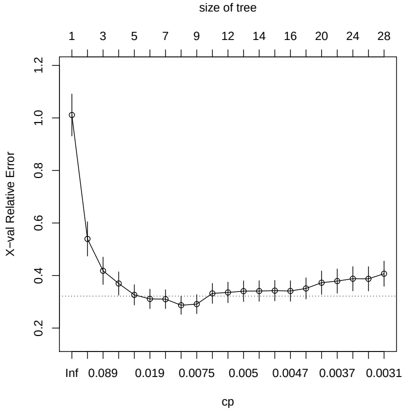
Pruning – why?

- The splitting can continue until each calibration observation is in its own leaf
- This is almost always **over-fitting** to the current dataset
- What we want is a tree for the best **prediction**
- Solution: **grow** a full tree; then **prune** it back to a simpler tree with the best **predictive** power
  - Similar to using the **adjusted** R^2 to avoid over-fitting a multiple linear regression

Pruning – how?

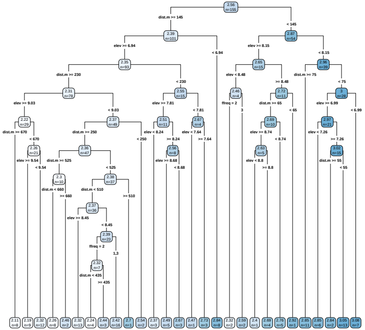
- The cp "complexity parameter" value: Any split that does not decrease the overall lack of fit by a factor of cp is not used.
  - Default value is 0.01 (1% increase in R^2)
  - Can be set by the analyst during **growing**
  - Can also be used as a target for **pruning**
- Q: How to decide on the value of cp that gives the best **predictive** tree?
- A: Use the **cross-validation error**, also called the **out-of-bag** error.
  - apply the model to the original data split K-fold (default 10), each time excluding some observations; compare predictions to actual values
  - Note how this fits the philosophy of data-driven approaches: **predictive** accuracy is the criterion

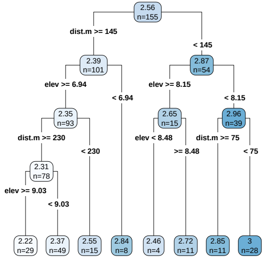
X-validation error vs. complexity parameter



Horizontal line is 1 standard error above the minimum error. Usually choose the largest cp below this; here cp=0.01299 (about 1.3% improvement in R^2).

Full and pruned trees





Full tree built with cp=0.003 = 0.3%; 27 leaves; pruned to 8 (cp=0.013 = 1.3%)

Interpretation: a noisy dataset if using these two predictors

Variable importance

- Unlike with regression we do not get any coefficient or its standard error for each predictor
- So to evaluate the importance of each predictor we see how much it's used in the tree
  - simple:
    - sum of gain in R^2 over all splits based on the predictor
  - complicated;
    - permute predictor values;
    - use these to re-build the tree;
    - compute cross-validation error;
    - the larger the difference, the more important

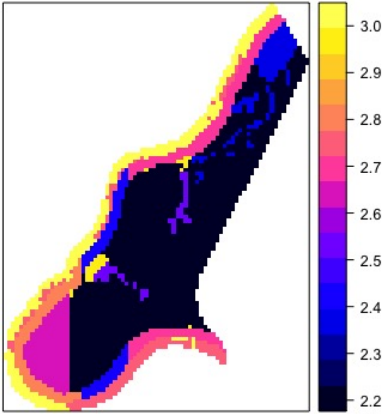
Variable importance – example

```
        variableImportance
dist.m     55.5876
elev       38.9996
ffreq       5.4128
```

Normalized to sum to 100% of the gain in R^2

Distance to river is most important.

Map predicted from Regression Tree

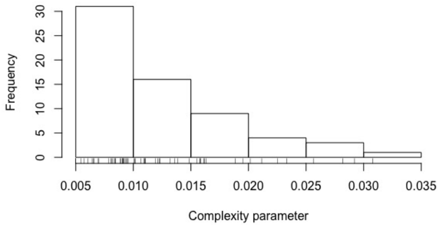


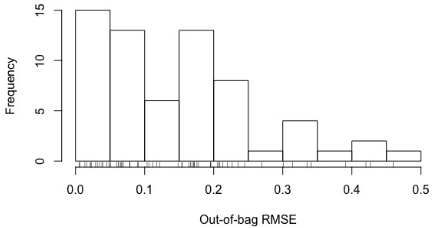
This tree: log(Zn) predicted from `dist` (45% importance); E (17%); `soil` (15%); N (11%); `ffreq`. (11%).

Sensitivity of Regression Trees to sample

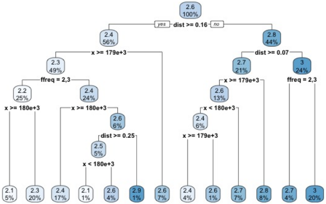
- **Question**: how sensitive are Regression Trees to the sample?
- **Experiment**: build trees from random samples of 140 of the 155 observations (only 10% not used!)
  - How different are the optimized **trees** and the predictive **maps**?
  - What is the distribution of the optimal **complexity parameter** and the **out-of-bag** (predictive) error?

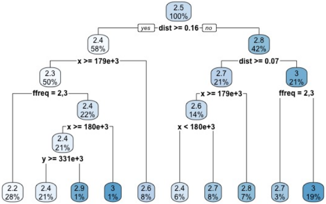
Sensitivity: complexity and out-of-bag error

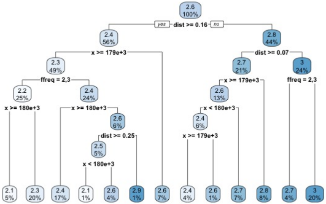




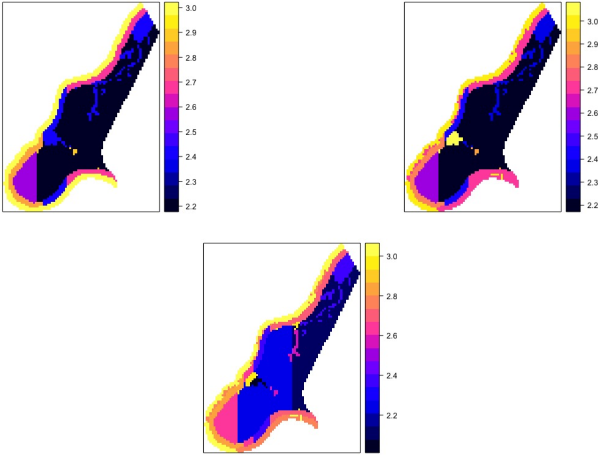
Sensitivity: trees







Sensitivity: predictive maps

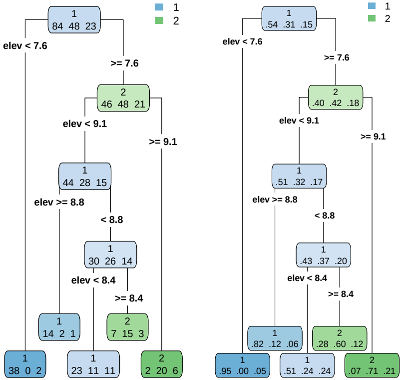


Regression trees are sensitive to the observations

- This is a problem!
- **Solution**: why have one tree when you can have a **forest**?

Classification trees

- Target variable is a **categorical** variable
- *Example* (Meuse river): flood frequency **class** (3 levels) predicted from distance to river and elevation
- Result (pruned): number of observations in each class (left); proportion (right) – note class 3 not predicted!



Random forests – motivation

- Instead of relying on a *single* (hopefully best) tree, maybe it is better to fit *many* trees.
- But. . . how to obtain *multiple* regression trees if we have only *one* data set?
  - Go into field and collect *new* sample data? too expensive and impractical.
  - *Split* the dataset and fit trees to the separate parts? Too few observations to build a reliable tree.
  - **Solution**: Use the *single* sample to generate an *ensemble* (group) of trees; use these together to predict.

Bagging (1)

- “Bag” = a group of samples “in the bag”; others “out-of-bag”
- Suppose we have a large sample that is a good **representation** of the study area
  - i.e., *sample* frequency distribution is close to *population* frequency distribution
- Generate a new sample is generated by **sampling from the sample**!

Bagging (2) – Boostrapping

Standard method for sampling in bagging is called **bootstrapping**[1]

- Select **same number of points** as in sample
- Sample **with replacement** (otherwise you get the same sample)
- So *some observations are used more than once!*
- But, **the sample is supposed to represent the population**, so these could be values that would have been obtained in a new field sample.

[1] for historical reasons

Sampling with replacement

```
> # sample 20 times from (1, 2,... 20) with replacement
> (my.sample <- sample(1:20, 20, replace=TRUE))
 [1]  7 13  5  2  1  9 19  1  6  2  9  9 12  4 11  9  5 20 20 11
> sort(my.sample)
 [1]  1  1  2  2  4  5  5  6  7  9  9  9  9 11 11 12 13 19 20 20
> (1:20) %in% my.sample     # in bag
 [1]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
[10] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE
[19]  TRUE  TRUE
> !((1:20) %in% my.sample)  # Out-of-bag
 [1] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE
[10]  TRUE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
[19] FALSE FALSE
```

Example: 10 bootstrap samples from the integers 1 … 20 – sorted

	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10
1	1	2	1	1	2	4	2	1	1	3
2	3	3	3	2	3	6	3	2	2	3
3	5	3	3	2	4	6	3	4	3	5
4	6	5	6	4	4	7	4	5	3	10
5	7	5	6	5	7	8	6	6	5	10
6	8	5	7	5	8	10	7	6	6	11
7	11	7	8	7	8	10	7	6	6	13
8	15	7	9	8	8	11	9	7	7	13
9	15	8	13	10	9	12	10	7	8	13
10	16	8	15	10	9	13	10	8	8	14
11	16	9	15	10	11	13	13	8	9	14
12	17	12	16	10	13	14	13	10	12	14
13	17	14	16	14	13	15	14	14	12	15
14	18	14	17	16	14	16	15	17	13	16
15	18	15	17	16	16	18	15	17	13	16
16	19	15	18	17	18	18	15	18	14	16
17	19	16	19	17	19	18	16	19	14	17
18	19	17	19	19	19	19	17	20	17	19
19	19	18	20	19	19	20	17	20	19	20
20	19	18	20	19	19	20	19	20	20	20

Forests with bagging – method

- Fit a **full regression tree** to each bootstrap sample; *do not prune*
- Each bootstrap sample results in a **tree** and in a **predicted value** for any combination values of the predictors
- Prediction is the **average** of the individual predictions from the "forest" of regression trees
- Jumps in predictions are **smoothed**; more precise predictions

Forest with bagging – limitations

- All predictors are tried at each split, so **trees tend to be similar**
- Some predictors may never enter into the trees → **missing source of diversity**
- Solution: **random forest** variation of bagging – **two sources of randomness**
  - *Random 1*: sampling by bagging
  - *Random 2*: choice of predictors at each split (see next)

Random forests

- Multiple samples obtained by bootstrapping, used to build trees (as in bagging)
- Average predictions over all trees (as in bagging)
- Besides, in each internal node a **random subset of splitting variables** (predictors) is used
    - Extra source of diversity among trees
    - Predictors that are “outcompeted” in bagging by stronger competitors may now enter the group of trees

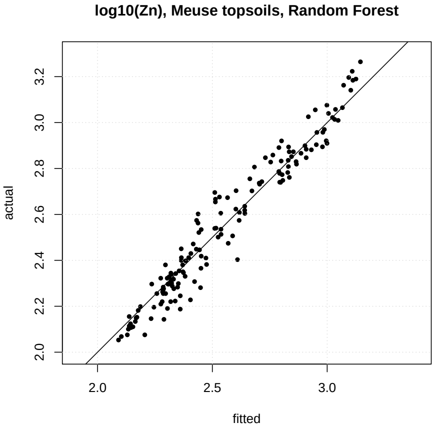
Selecting predictors at each split

- `randomForest`, `ranger` parameter **mtry**: Number of variables randomly sampled as candidates at each split.
  - `ranger` default $\lfloor \sqrt{p} \rfloor$, where p is number of possible predictors
  - example: 60 predictors $\rightarrow \lfloor \sqrt{60} \rfloor = \lfloor 7.74 \rfloor = 7$ tried at each split
  - `randomForest` default $\lfloor p/3 \rfloor$
  - example: 60 predictors $\rightarrow \lfloor 60/3 \rfloor = \lfloor 20 \rfloor = 20$ tried at each split
- Can be **tuned**, see below.

Other control parameters

- number of trees in the forest
  - **ranger** parameter **min.node.size**
  - **randomForest** parameter **ntree**
  - default = 500
- minimal node size
  - **ranger** parameter **min.node.size**
  - **randomForest** parameter **nodesize**
  - default = 5
- (*optional*) names of variables to always try at each split; weights for sampling of training observations (to compensate for unbalanced samples)

Fitted by RF vs. observed

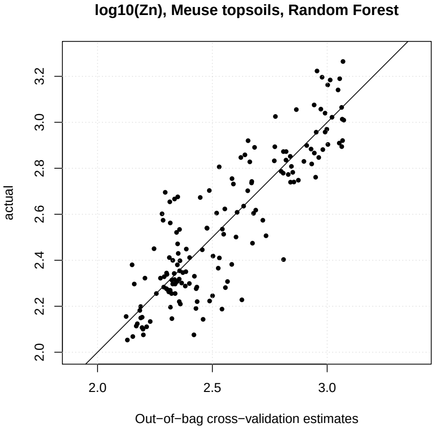


Average prediction of many trees, comes close to actual value

Out-of-bag ("OOB") evaluation

- In a bootstrap sample not all samples are present: sampling is with *replacement*.
- Sample data not in bootstrap sample: **out-of-bag** sample: these were *not* used to build the tree.
- These data can be used for **evaluation** ("validation"):
  - Use the tree fitted on the bootstrap sample to predict at out-of-bag data, i.e., observations *not* used in that bootstrap sample.
  - Compute **squared prediction error** for out-of-bag data.
- This gives a very good estimate of the true prediction error *if* the sample was **representative** of the population.

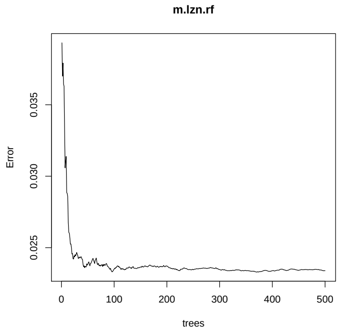
Out-of-bag RF predictions vs. observed



Average prediction of many trees *not* using an observation. Further from actual value; **better estimate of predictive power**

How many trees are needed to make a forest?

- Plot mean squared out-of-bag error against number of trees
- Check whether this is stable
- If not, increase number of trees



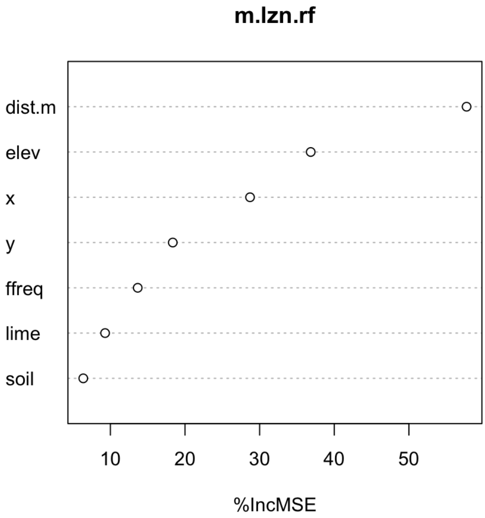
Variable importance

Importance quantified by permutation accuracy:

- *randomize* (permute) values of a predictor
  - so the predictor can not have any relation with the target
- build a random forest with this randomized predictors and the other (non-randomized) ones
- compute OOB error; compare with OOB error *without* randomization
  - the larger the difference, the more important
- Example:

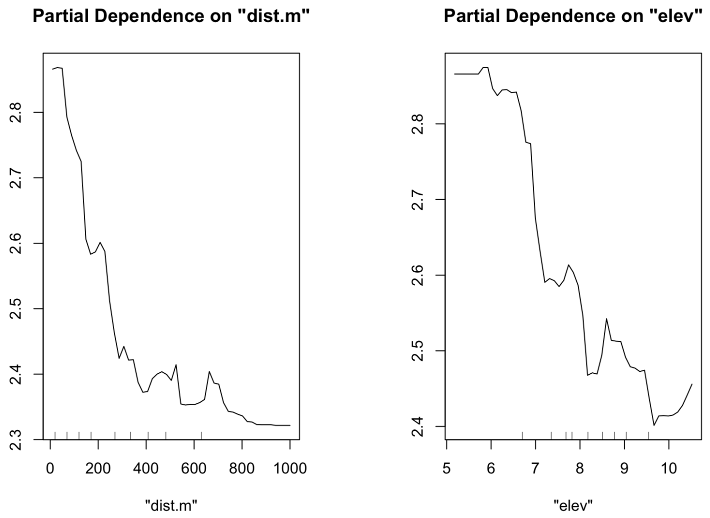
```
       % Increase in MSE under randomization
ffreq    9.4
dist.m 67.5
elev   54.0
```

Variable importance plot

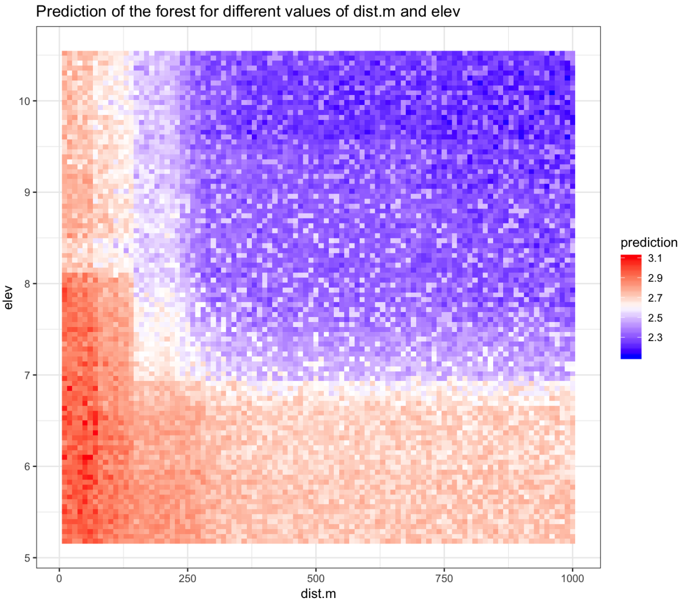


Partial dependence plots

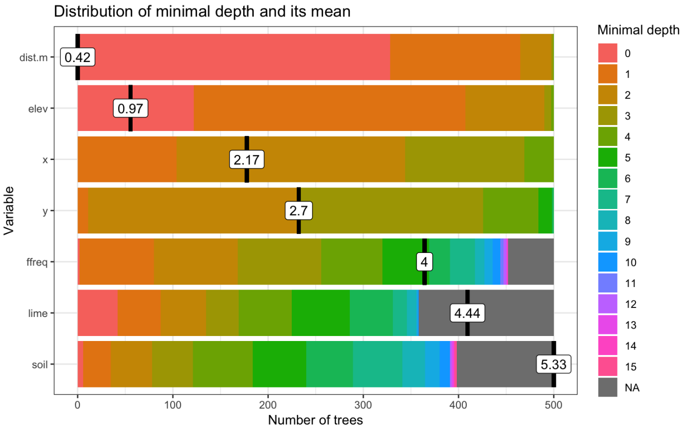
The effect of each variable, with the others held **constant** at their means/most common class.



Two-way partial dependence



Examining the forest – at what depth in the trees are predictors used?

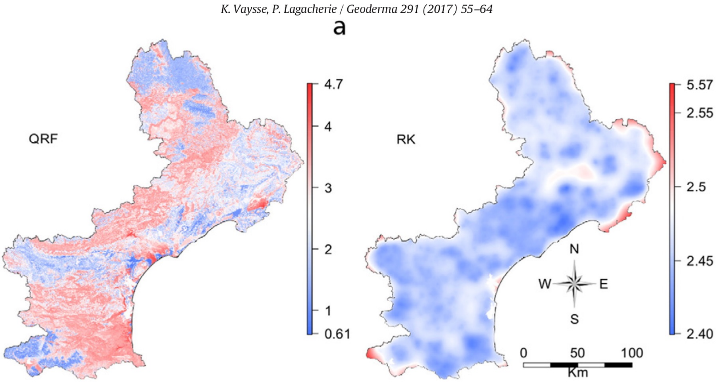


Earlier in tree → most discriminating

Uncertainty of RF maps

- Recall: RF is built from many trees, each tree makes a prediction at each location
- These are **averaged** to get a “best” predictive map
- However, the *set* of predictions can be considered a **probability distribution** of the true value
- From this we can make a map of any **quantile**, e.g., 5% and 95% confidence limits, or prediction interval width

RF uncertainty vs. RK uncertainty



95% prediction interval for topsoil pH
prediction from 2 024 point observations and 18 covariates
Languedoc-Roussillon region (F)

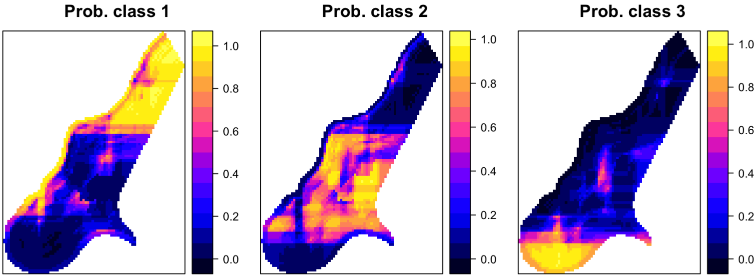
References for quantile random forests

- Meinshausen, N. (2006). *Quantile regression forests*. **Journal of Machine Learning Research**, 7, 983–999.
- Meinshausen, N., & Schiesser, L., 2015. *Quantregforest: Quantile Regression Forests. R package*. https://cran.r-project.org
- Vaysse, K., & Lagacherie, P. (2017). *Using quantile regression forest to estimate uncertainty of digital soil mapping products*. **Geoderma**, 291, 55–64. https://doi.org/10.1016/j.geoderma.2016.12.017

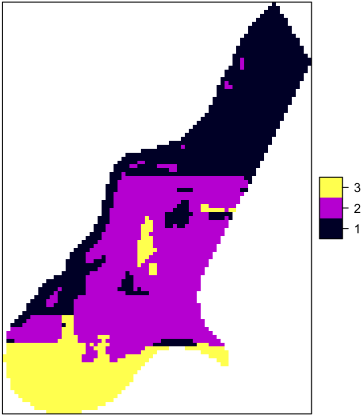
Random forests for categorical variables

- Target variable is **categorical**, i.e., a class
  - Example: Meuse river flooding frequency classes (every year, every 2–5 years, rare or none)
- Final prediction is the class predicted by the **majority** of the regression trees in the forest
- Can also see the **probabilty** for each class, by predicting with the model with the **type="prob"** argument to `predict.randomForest`.

Predicted class probabilty



Predicted most probable class



Accuracy measures

- naïve agreement: how often a class in the training set is correctly predicted – see with a **confusion matrix** ("cross-classification")
- Out-of-bag (OOB) estimate of error rate
- **Gini impurity**: how often a *randomly chosen* training observation would be *incorrectly* assigned ...
  ... if it were *randomly labeled* according to the *frequency distribution* of labels in the subset.

Cross-classification matrix

A **confusion matrix** (a.k.a. cross-classification matrix) of actual (columns) vs. predicted (rows) classes:

```
Confusion matrix:
   1  2  3 class.error
1 77  7  0  0.08333333
2  3 40  5  0.16666667
3  1  9 13  0.43478261
```

Predictor selection

- **Problem**: large number of possible predictors, can lead to …
    - Computational inefficiency
    - Difficult interpretation of variable importance
    - Meaningless good fits, even if using cross-validation[2]
- **Solution 1**: expert selection from “known” relations
    - this is then not pure “data mining” for unsuspected relations
- **Solution 2**: (semi-)automatic feature selection, see next.

[2] Wadoux, A. M. J.-C., *et al.* (2019). **A note on knowledge discovery and machine learning in digital soil mapping**. **European Journal of Soil Science**, 71, 133–136. https://doi.org/10.1111/ejss.12909

Feature selection methods

Wrapper methods: "evaluate multiple models using procedures that add and/or remove predictors to find the optimal combination that **maximizes model performance.**"

- risk of over-fitting
- high computational load

Filter methods: "evaluate the relevance of the predictors **outside of the predictive models** and subsequently model only the predictors that **pass some criterion"**

- does not account for correlation among predictors
- does not directly assess model performance

Recursive feature elimination

- A “wrapper” method
- Implemented in `caret::rfe` “Backwards Feature Selection” function
- Algorithm: “Recursive Feature Elimination (RFE) incorporating resampling”
  1. Partition data into training/test sets via resampling
  2. Start with **full model**, compute variable importance
  3. **for** each **proposed subset size**
     1. Re-compute model with **reduced variable sets**
     2. **Calculate performance profiles** using **test samples**
  4. Determine **optimum number** of predictors

Reference for feature selection

- From the documentation of the caret package (§5).
- **Feature selection**: `https://topepo.github.io/caret/feature-selection-overview.html`
- **Recursive feature elimination**: `https://topepo.github.io/caret/recursive-feature-elimination.html`

Cubist

- Similar to CART, but instead of **single values** at leaves it creates a **multivariate linear regression** for the cases in the leaf
- **Advantage vs. CART**: predictions are continuous, not discrete values equal to the number of leaves in the regression tree.
  - Also can be improved with nearest-neighbours, see below
- **Advantage vs. RF**: the model can be interpreted, to a certain extent.
- **Disadvantage**: its algorithm is not easy to understand; however its results are generally quite good.

Refinements to Cubist

- **"Committees"** of models: a sequence of models, where each corrects the errors in the previous one
- **nearest-neighbours adjustment**: modify model result at a prediction point from some number of neighbours **in feature** (predictor) **space.**

$$\hat{y}' = \frac{1}{K} \sum_{i=1}^{K} w_i \left[t_i + (\hat{y} - \hat{t}_i) \right] \quad (1)$$

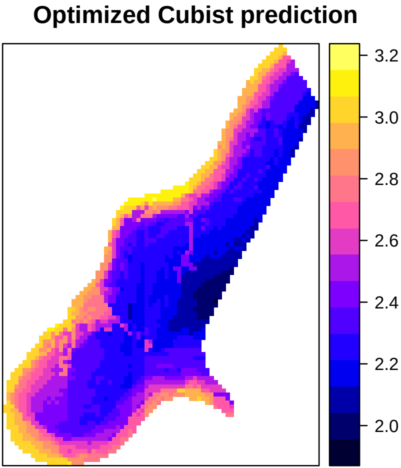
where t_i is the actual value of the neighbour, $\hat{t}_i$ is its value predicted by the model tree(s), and w_i is the weight given to this neighbour for the adjustment, based on its distance D_i from the target point. These are computed as $w_i = 1/(D_i + 0.5)$ and normalized to sum to one.

Example cubist model

```
Rule 1/1: [66 cases, mean 2.288309, range 2.053078 to 2.89098, err 0.1036
  if x > 179095, dist > 0.211846
  then outcome = 2.406759 - 0.32 dist
Rule 1/2: [9 cases, mean 2.596965, range 2.330414 to 2.832509, err 0.1163
  if x <= 179095, dist > 0.211846
  then outcome = -277.415278 + 0.000847 y + 0.56 dist
Rule 1/3: [80 cases, mean 2.772547, range 2.187521 to 3.264582, err 0.157
  if dist <= 0.211846
  then outcome = 2.632508 - 2.1 dist - 2.4e-05 x + 1.4e-05 y

Rule 2/1: [45 cases, mean 2.418724, range 2.10721 to 2.893762, err 0.1822
  if x <= 179826, ffreq in {2, 3}
  then outcome = 128.701732 - 0.000705 x
Rule 2/2: [121 cases, mean 2.443053, range 2.053078 to 3.055378, err 0.18
  if dist > 0.0703468
  then outcome = 30.512065 - 0.87 dist - 0.000154 x
Rule 2/3: [55 cases, mean 2.543648, range 2.075547 to 3.055378, err 0.125
  if dist > 0.0703468, ffreq = 1
  then outcome = 37.730889 - 0.000314 x - 0.35 dist + 6.5e-05 y
Rule 2/4: [34 cases, mean 2.958686, range 2.574031 to 3.264582, err 0.139
  if dist <= 0.0703468
  then outcome = 2.982852 - 0.36 dist
```

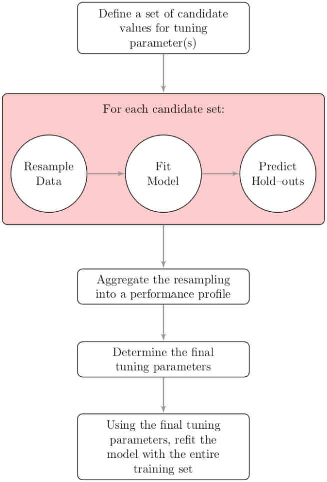
Map predicted by Cubist



Model tuning

- Data-driven models have **parameters** that control their behaviour and can significantly affect their **predictive power.**
    - **CART**: complexity parameter
    - **randomForest**: number of predictors to try at each split; minimum number of observations in a leaf; number of trees in the forest
        - too many predictors → trees too uniform, loss of diversity; too few → highly-variable trees, poor predictions
        - too few observations per leaf *to* imprecise prediction; too many → over-fitting
        - too few trees → sub-optimal model; too many trees → wasted computation
    - **Cubist**: number of committees; number of nearest neighbours
- The model can be **tuned** to **optimize** the selection of these.

Model tuning – flow chart



source: Kuhn, M., & Johnson, K. (2013). *Applied Predictive Modeling* (2013 edition). New York: Springer; figure 4.4

Model tuning – algorithm

1. For each combination of parameters to be optimized:
   1. Split the dataset into some disjunct subsets, for example 10, by random sampling.
   2. For each subset:
      1. Fit the model with the selected parameters on all but one of the subsets (**train** subset).
      2. Predict at the remaining subset, i.e., the one not used for model building, with the fitted model.
      3. Compute the **goodness-of-fit** statistics of fitting to the **test** subset
         e.g., root mean square error (RMSE) of prediction; squared correlation coefficient between the actual and fitted values, i.e., R^2 against a 1:1 line.
   3. Average the statistics for the disjunct test subsets.
2. Search the table of results for the best results
   e.g., lowest RMSE, highest R^2.

Model tuning – R implementation

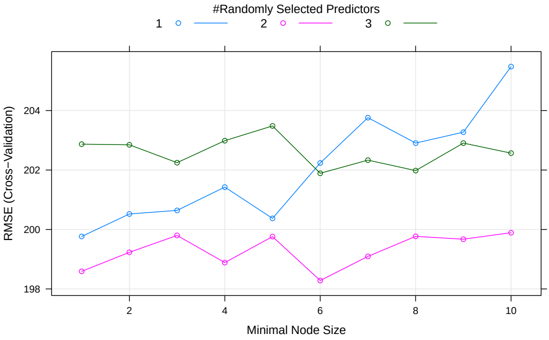
- caret "**C**lassification **A**nd **RE**gression **T**raining" package
  - Kuhn, M. (2008). *Building predictive models in R using the caret package*. Journal of Statistical Software, 28(5), 1–26.
  - `https://topepo.github.io/caret/index.html`
  - can tune 200+ models; some built-in, some by calling the appropriate package
- method:
  1. set up a vector or matrix with the parameter values to test, e.g, all combinations of 1 . . . 3 splitting variables to try, and 1 . . . 10 observations per leaf
  2. run the model for all of these and collect the cross-validation statistics
  3. select the best one and build a final model

Model tuning example – random forest (1)

```
> ranger.tune <- train(x = preds, y = response, method="ranger",
             tuneGrid = expand.grid(.mtry = 1:3,
                                    .splitrule = "variance",
                                    .min.node.size = 1:10),
             trControl = trainControl(method = 'cv'))
> print(ranger.tune)

## Resampling: Cross-Validated (10 fold)
## Resampling results across tuning parameters:
##
##   mtry  min.node.size  RMSE      Rsquared   MAE
##   1      1             199.7651  0.8862826  156.1662
##   1      2             200.5215  0.8851154  156.3225
##   1      3             200.6421  0.8854146  156.2801
...
##   3      8             201.9809  0.8793349  158.7097
##   3      9             202.9065  0.8781754  159.7739
##   3     10             202.5687  0.8788200  159.5980
##
## RMSE was used to select the optimal model
## Final values: mtry = 2, min.node.size = 6.
```

Model tuning example – random forest (2)



Find the minimum RMSE; but favour simpler models (fewer predictors, larger nodes) if not too much difference

Model tuning example – Cubist (1)

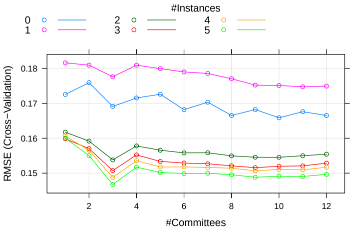
```
> cubist.tune <- train(x = all.preds, y = all.resp, method="cubist",
                     tuneGrid = expand.grid(.committees = 1:12,
                                            .neighbors = 0:5),
                     trControl = trainControl(method = 'cv'))
```

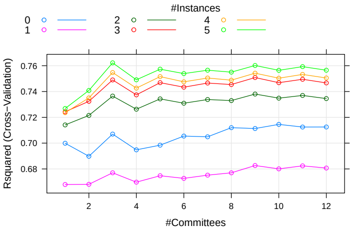
```
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 139, 139, 140, 139, 139, 139, ...
## Resampling results across tuning parameters:
##
##   committees  neighbors  RMSE       Rsquared   MAE
##    1          0          0.1898596  0.6678588  0.1405553
##    1          1          0.1764705  0.6953460  0.1189364
##    1          2          0.1654910  0.7296723  0.1163660
##    1          3          0.1623381  0.7425831  0.1163285
##    1          4          0.1631900  0.7453506  0.1192963
...
##   12          3          0.1599994  0.7533962  0.1139932
##   12          4          0.1584434  0.7617762  0.1153331
##   12          5          0.1589143  0.7622337  0.1165942
\##
## RMSE was used to select the optimal model using the smallest value.
## The final values: committees = 10, neighbors = 4.
```

Model tuning example – Cubist (2)

Criterion: RMSE

Criterion: R^2





Adding one neighbour reduces predictive power; adding 2 . . . increases it; 3 is close to optimum

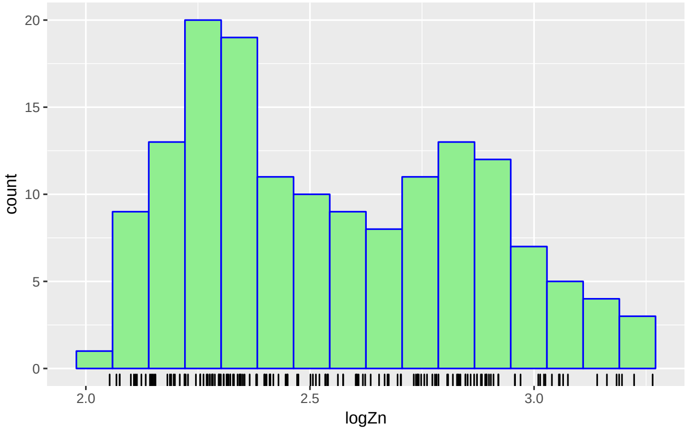
Committees improve predictive power; 3 is optimum

6 **Spatial random forests**

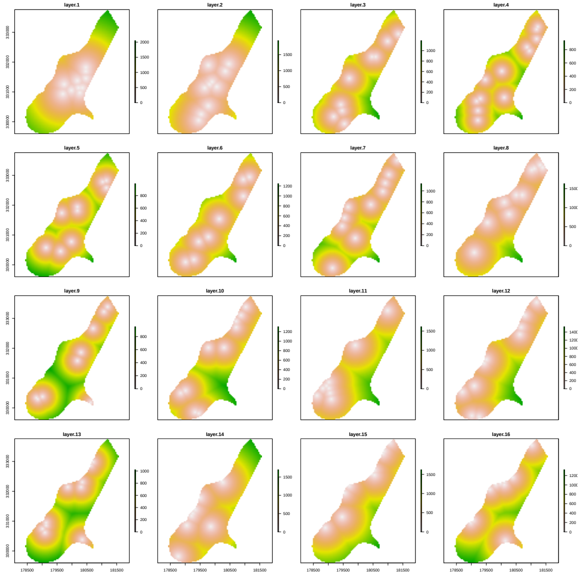
Spatial random forests

- Random forests can use **coördinates** and **distances** to geographic features as predictors
  - e.g., E, N, distance to river, distance to a single point . . .
- Can also use distances to **multiple points** as predictors
  - Distance **buffers**: distance to closest point with some range of values
  - Common approach: compute **quantiles** of the response variable and one buffer for each
  - Each sample point has a distance to the closest point in each quantile
- This uses **separation between point-pairs** of different values, but with *no* model.

log(Zn) distribution – 16 quantiles



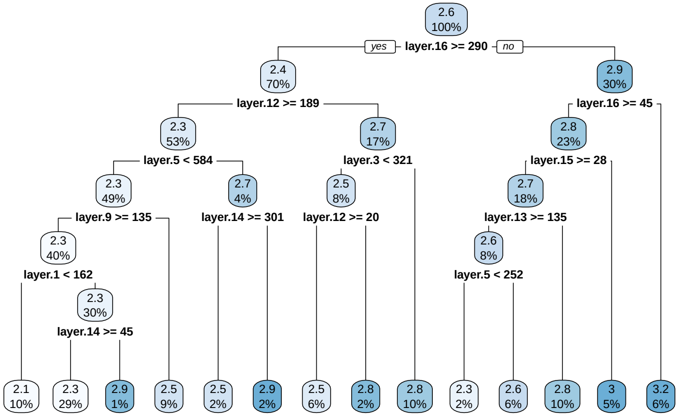
Distance to closest point in each quantile



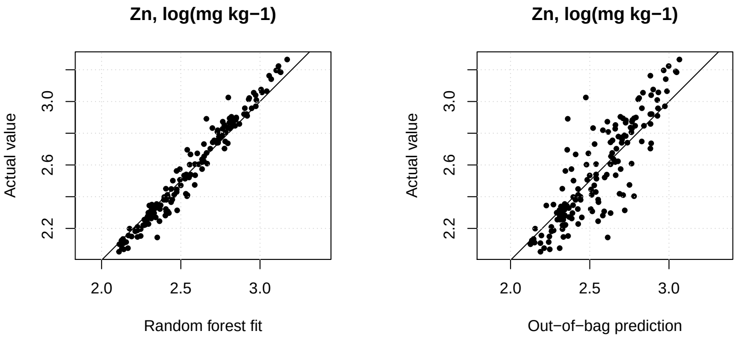
Data-driven methods for predictive modelling

DGR/罗大维

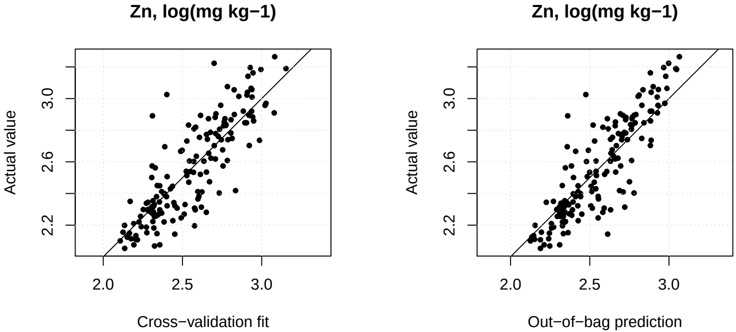
Regression tree on 16 distance buffers



Random forest prediction on 16 distance buffers



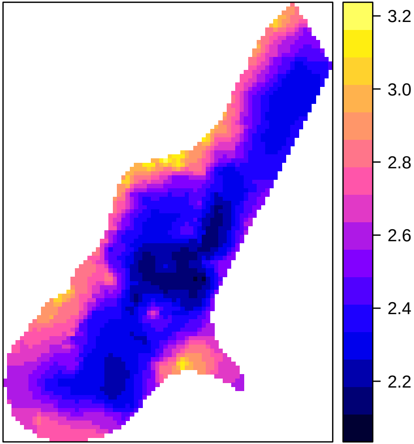
OOB error vs. OK cross-validation error



OK RF

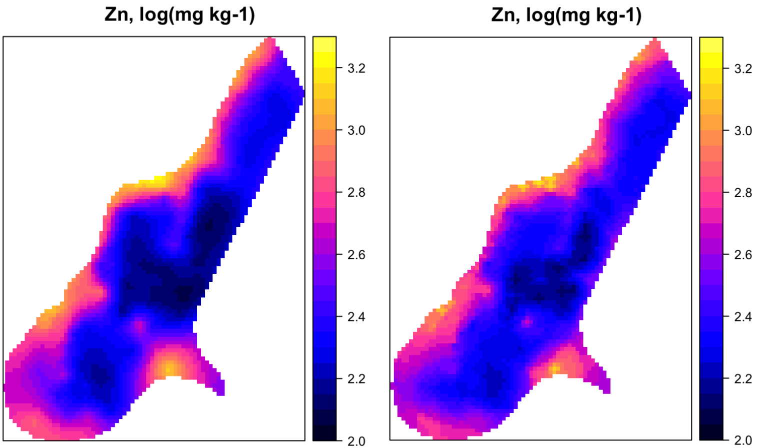
Note that RF does *not* use any *model* of spatial autocorrelation!

Random forest map on 16 distance buffers



Resembles OK map, but *no model* was used.

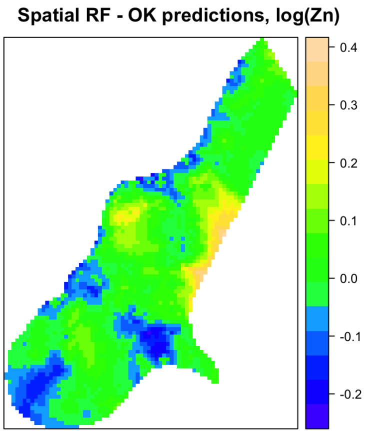
Compare with Ordinary Kriging



Ordinary Kriging

Random forest on distance buffers

Difference spatial RF - OK



Reference for spatial random forests

- Hengl, T., Nussbaum, M., Wright, M. N., Heuvelink, G. B. M., & Gräler, B. (2018). *Random forest as a generic framework for predictive modeling of spatial and spatio-temporal variables*. **PeerJ**, 6, e5518. `https://doi.org/10.7717/peerj.5518`

Conclusion: Data-driven vs. model-based methods

- **Data**-driven: main aim is **predictive power**
  - Individual trees can be interpreted, but forests can not (only can see variable importance, not choice or cutpoints)
- **Model**-based: main aim is **understanding processes**
  - We hope the model is a simplified representation of the process that produced the observations
  - If the model is correct, predictions will be accurate

Conclusion: limitations

- **Data**-driven methods depend on their **training observations**
  - They have no way to extrapolate or even interpolate to unobserved areas in feature space
  - So the observations should cover the entire range of the population
- **Model**-based methods depend on a correct **empirical-statistical model**
  - Model is derived from training observations, but many models are possible
  - Various model-selection techniques
  - Wrong model → poor predictions, incorrect understanding of processes

End