# Uncertainty in Machine Learning: Modeling and Visualizing Errors in Spatial Predictions Based on ML


tom.hengl@enviometrix.net
@tom_hengl
thengl
http://envirometrix.net

Preprint

View 6 tweets

**NOT PEER-REVIEWED**

"*PeerJ Preprints*" is a venue for early communication or feedback before peer review. Data may be preliminary.

Learn more about preprints or browse peer-reviewed articles instead.

# Random Forest as a generic framework for predictive modeling of spatial and spatio-temporal variables

Research article | Biogeography | Soil Science | Computational Science | Data Mining and Machine Learning | Spatial and Geographic Information Science

Tomislav Hengl[1], Madlene Nussbaum[2], Marvin N Wright[3], Gerard B.M. Heuvelink[4]

March 14, 2018

## Author and article information

## Abstract

Random forest and similar Machine Learning techniques are already used to generate spatial predictions, but spatial location of points (geography) is often ignored in the modeling process. Spatial auto-correlation, especially if still existent in the cross-validation residuals, indicates that the predictions are maybe biased, and this is suboptimal. This paper presents a random forest for spatial predictions framework (RFsp) where buffer distances from observation points are used as explanatory variables, thus incorporating geographical proximity effects into the prediction process. The RFsp framework is illustrated with examples that use textbook datasets and apply

README.md

# RFsp — Random Forest for spatial data (R tutorial)

Hengl, T., Nussbaum, M., and Wright, M.N.





# *Some basic definitions (spatial prediction and the mapping accuracy)*

Model-based spatial prediction algorithms commonly aim to minimize the prediction error variance $\sigma^2(\mathbf{s}_0)$ at a prediction location $\mathbf{s}_0$ under the constraint of unbiasedness (Christensen, 2001). Unbiasedness and prediction error variance are defined in terms of a statistical model $\mathbf{Y} = \{Y(\mathbf{s}),\ \mathbf{s} \in D\}$ of the measurements $y(\mathbf{s}_i)$. In mathematical terms, the prediction error variance:

$$\sigma^2(\mathbf{s}_0) = \mathrm{E}\left\{\left(\hat{Y}(\mathbf{s}_0) - Y(\mathbf{s}_0)\right)^2\right\} \tag{1}$$

is to be minimized while satisfying the (unbiasedness) constraint:

$$\mathrm{E}\left\{\hat{Y}(\mathbf{s}_0) - Y(\mathbf{s}_0)\right\} = 0 \tag{2}$$

where the predictor $\hat{Y}(\mathbf{s}_0)$ of $Y(\mathbf{s}_0)$ is typically taken as a function of covariates and the $Y(\mathbf{s}_i)$ which, upon substitution of the observations $y(\mathbf{s}_i)$, yields a (deterministic) prediction $\hat{y}(\mathbf{s}_0)$.

In the case of multiple linear regression (MLR), model assumptions state that at any location in $D$ the dependent variable is the sum of a linear combination of the covariates at that location and a zero-mean normally distributed residual. Thus, at the $n$ observation locations we have:

$$\mathbf{Y} = \mathbf{X}^{\mathbf{T}} \cdot \beta + \varepsilon \tag{3}$$

where $\mathbf{Y}$ is a vector of the target variable at the $n$ observation locations, $\mathbf{X}$ is an $n \times p$ matrix of covariates at the same locations and $\beta$ is a vector of $p$ regression coefficients. The stochastic residual $\varepsilon$ is assumed to be independently and identically distributed. The paired observations of the target variable and covariates ($\mathbf{y}$ and $\mathbf{X}$) are used to estimate the regression coefficients using, e.g., Ordinary Least Squares (Kutner et al., 2004):

$$\hat{\beta} = \left(\mathbf{X}^{\mathbf{T}} \cdot \mathbf{X}\right)^{-1} \cdot \mathbf{X}^{\mathbf{T}} \cdot \mathbf{y} \tag{4}$$

once the coefficients are estimated, these can be used to generate a prediction at $\mathbf{s}_0$:

$$\hat{y}(\mathbf{s}_0) = \mathbf{x}_0^{\mathbf{T}} \cdot \hat{\beta} \tag{5}$$

with associated prediction error variance:

$$\sigma^2(\mathbf{s}_0) = var\left[\varepsilon(\mathbf{s}_0)\right] \cdot \left[1 + \mathbf{x}_0^{\mathrm{T}} \cdot \left(\mathbf{X}^{\mathrm{T}} \cdot \mathbf{X}\right)^{-\mathbf{1}} \cdot \mathbf{x}_0\right] \tag{6}$$

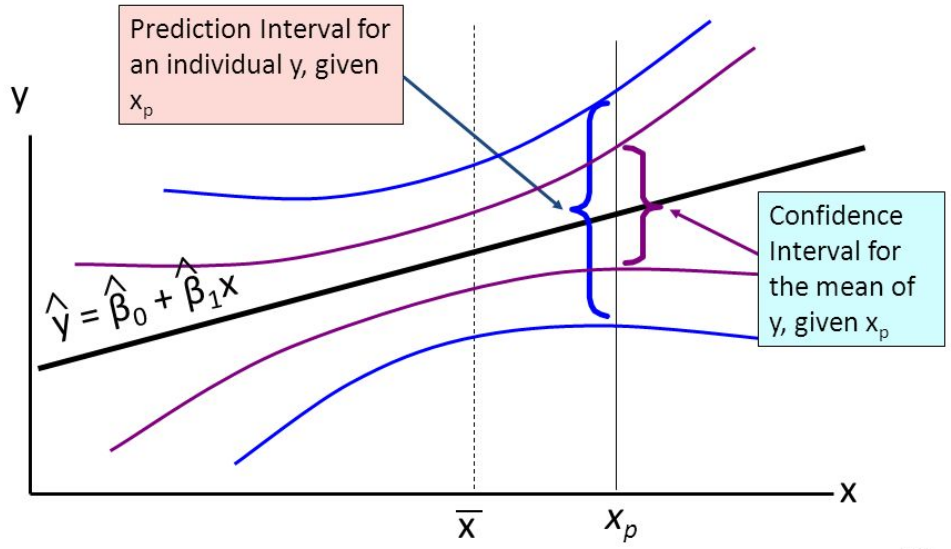
here, $\mathbf{x}_0$ is a vector with covariates at the prediction location and $var\left[\varepsilon(\mathbf{s}_0)\right]$ is the variance of the stochastic residual. The latter is usually estimated by the mean squared error (MSE):

$$\mathrm{MSE} = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n - p} \tag{7}$$

# *Spatial prediction using linear regression*

$\hat{y}(\mathbf{s}_0)$ = prediction

$\sigma^2(\mathbf{s}_0)$ = prediction error = fitting error (MSE) + uncertainty of new estimate (c.i.)



# *Difference between training and validation*

| *Training / model fitting** | *Validation* |
|---|---|
| Goodness of fit (OLS) | Model-free estimate of the mapping success (repeated refitting) |
| No special design requirements | Objective experimental designs |
| Artificial measure of the modeling success (over-fitting problems) | Actual accuracy |

*In the case of many machine learning methods, training is also based on bagging / boosting (hence internal validation).

$$\text{RMSE} = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(\hat{y}(\mathbf{s}_j) - y(\mathbf{s}_j))^2}$$

$$\text{ME} = \frac{1}{m}\sum_{j=1}^{m}(\hat{y}(\mathbf{s}_j) - y(\mathbf{s}_j))$$

In addition to R–square, we also derive Lin's Concordance Correlation Coefficient (CCC) (Steichen and Cox, 2002):

$$\rho_c = \frac{2 \cdot \rho \cdot \sigma_{\hat{y}} \cdot \sigma_y}{\sigma_{\hat{y}}^2 + \sigma_y^2 + (\mu_{\hat{y}} - \mu_y)^2} \tag{21}$$

where $\hat{y}$ are the predicted values and $y$ are actual values at cross-validation points, $\mu_{\hat{y}}$ and $\mu_y$ are predicted and observed means and $\rho$ is the correlation coefficient between predicted and observed values. CCC correctly quantifies how far the observed data deviate from the line of perfect concordance

# *Error of the prediction error*

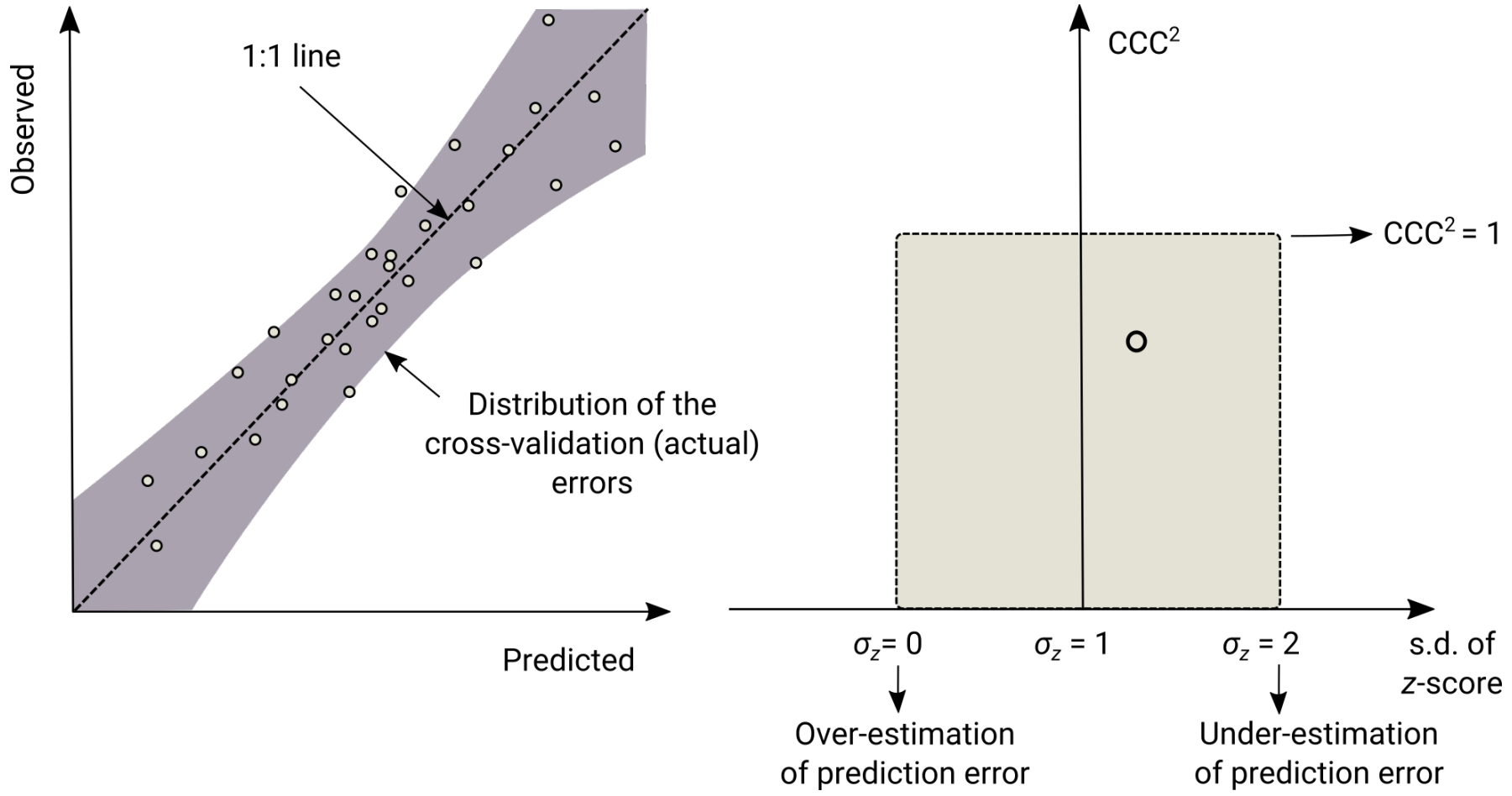
The error of estimating the variance of prediction errors can likewise be quantified via the $z$-score (Bivand et al., 2008):

$$z_{score}(\mathbf{s}_j) = \frac{\hat{y}(\mathbf{s}_j) - y(\mathbf{s}_j)}{\sigma(\mathbf{s}_j)} \tag{22}$$

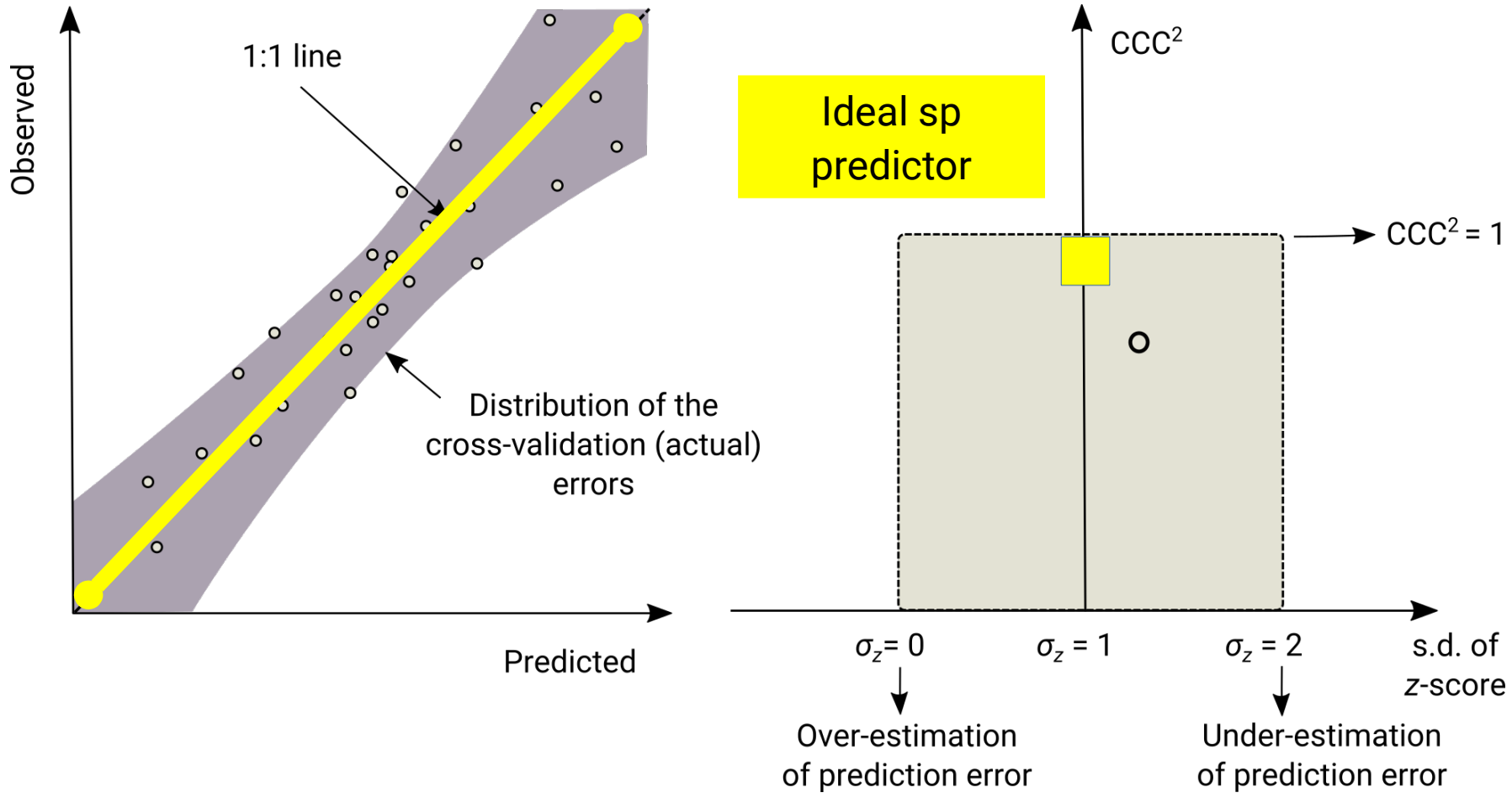
the $z$-score are expected to have a mean equal to 0 and variance equal to 1. If the $z$-score variance is substantially smaller than 1 then the model overestimates the actual prediction uncertainty. If the $z$-score variance is substantially greater than 1 then the model underestimates the prediction uncertainty.

Z-score is the measure of how realistic is our prediction error
(basically: uncertainty of uncertainty!!)

# *The two key measures of the mapping success*



# *The two key measures of the mapping success*



# *The CV residuals should show no sp. dep.*

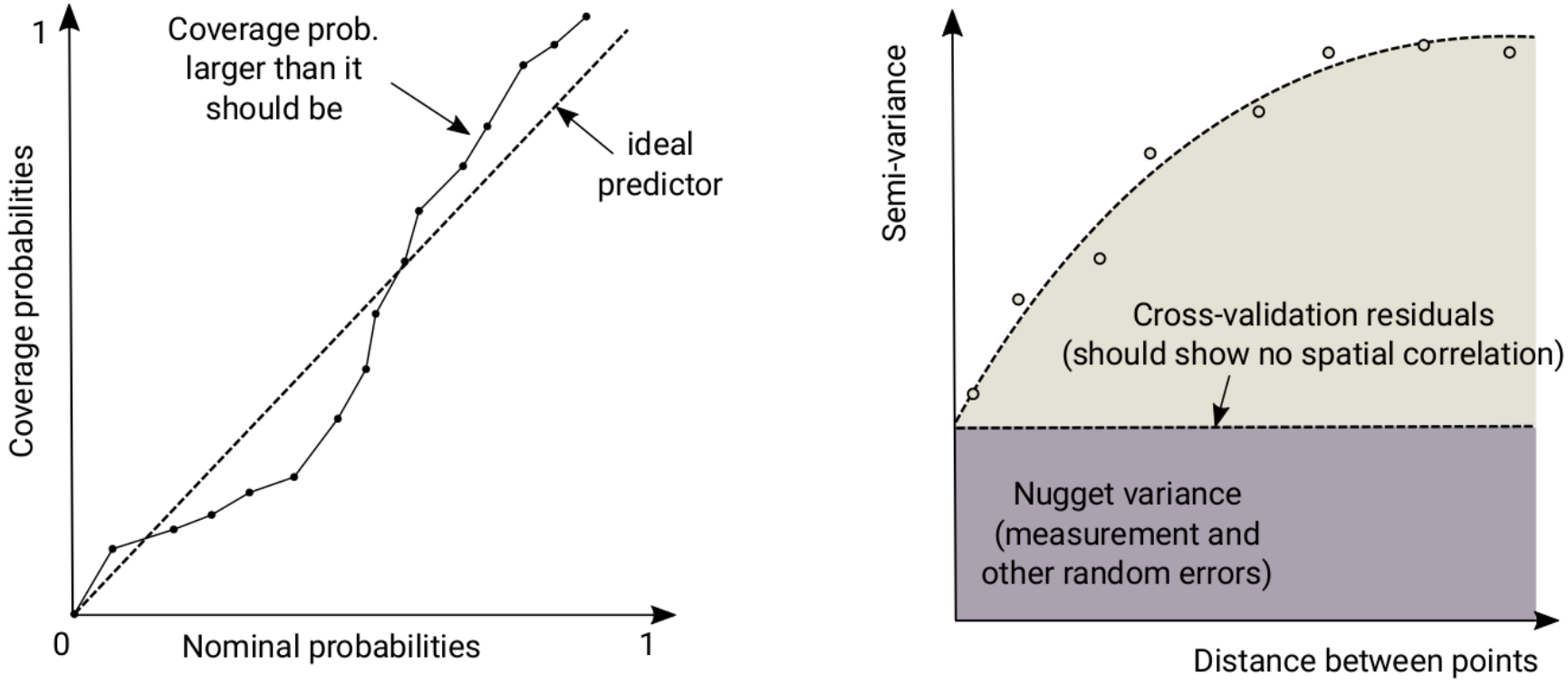



**Figure 3.** Schematic examples of standard mapping performance criteria used for evaluation of spatial prediction algorithms and their interpretation: (a) predicted vs. observed plot, (b) standardized accuracy vs. standard deviation of the *z*-scores, (c) *"accuracy plots"* (after Goovaerts (1999)), and (d) variogram of the target variable and the cross-validation residuals. MSE = Mean Squared residual Error. In principle, all plots and statistics reported in this paper are based on the results of *n*–fold cross-validation.

# *The CV residuals should show no sp. dep.*

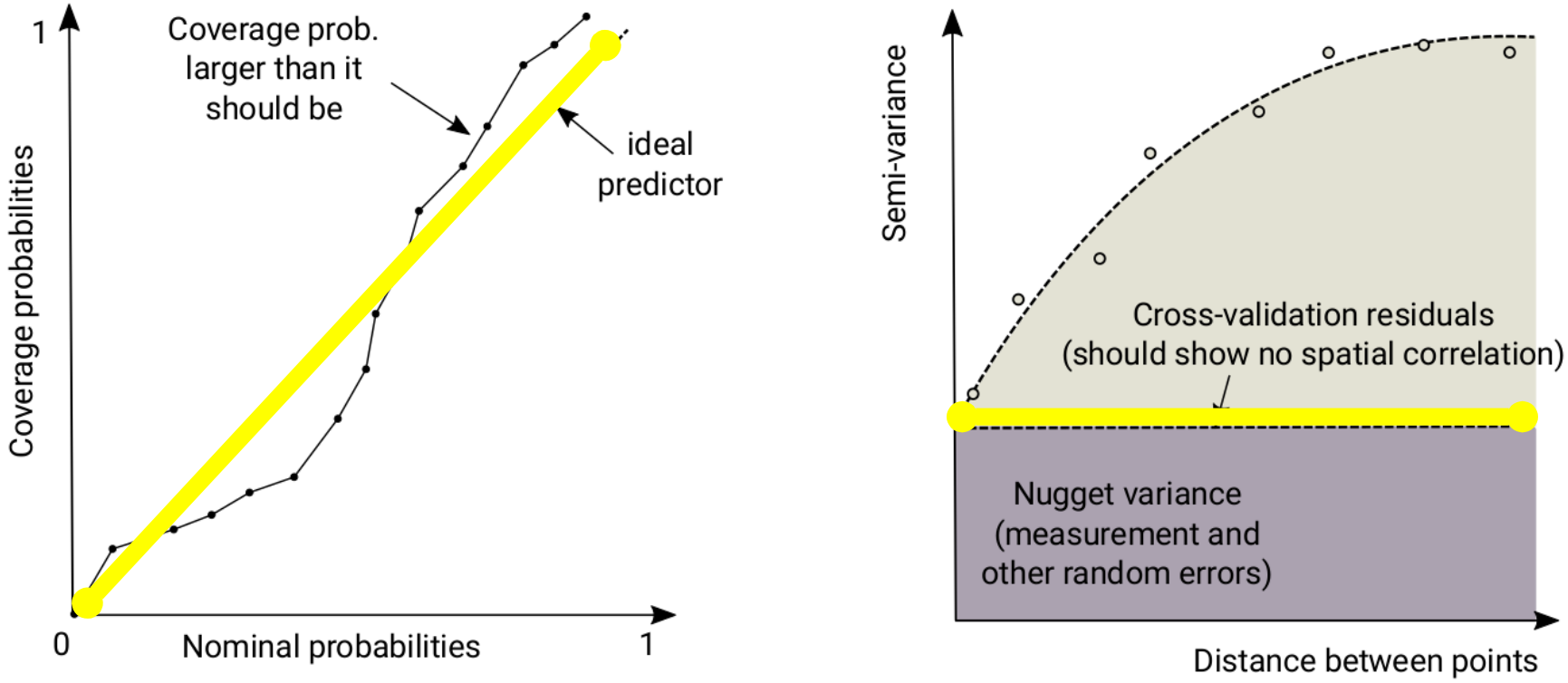



**Figure 3.** Schematic examples of standard mapping performance criteria used for evaluation of spatial prediction algorithms and their interpretation: (a) predicted vs. observed plot, (b) standardized accuracy vs. standard deviation of the *z*-scores, (c) "*accuracy plots*" (after Goovaerts (1999)), and (d) variogram of the target variable and the cross-validation residuals. MSE = Mean Squared residual Error. In principle, all plots and statistics reported in this paper are based on the results of *n*–fold cross-validation.

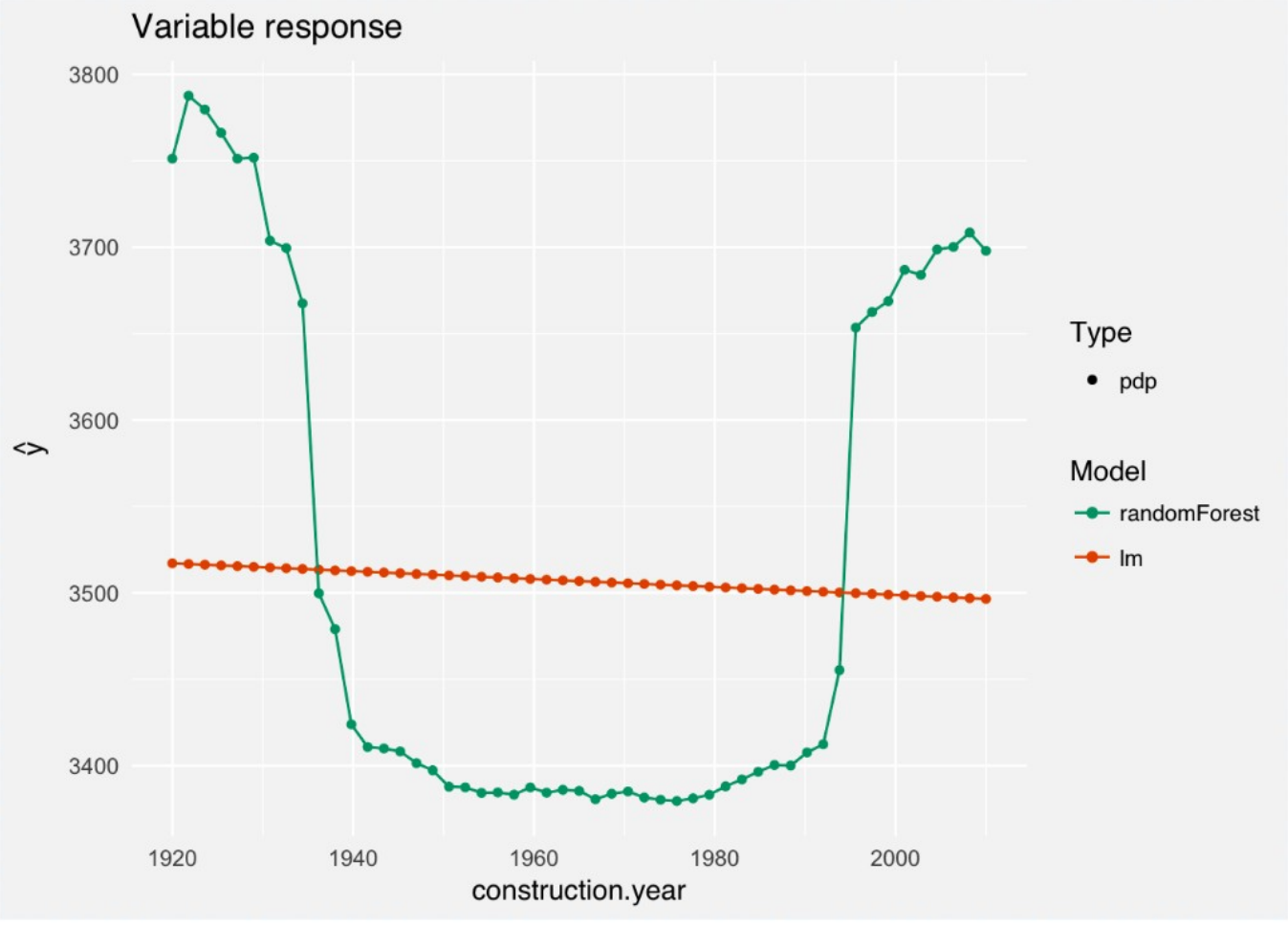
# *So in summary:*

- Spatial prediction model: $\mathbf{Y} = \mathbf{X}^{\mathbf{T}} \cdot \beta + \varepsilon$
- Model fitting algorithm: $\hat{\beta} = (\mathbf{X}^{\mathbf{T}} \cdot \mathbf{X})^{-1} \cdot \mathbf{X}^{\mathbf{T}} \cdot \mathbf{y}$
- Predictions: $\hat{y}(\mathbf{s}_0) = \mathbf{x}_0^{\mathbf{T}} \cdot \hat{\beta}$
- Prediction errors: $\sigma^2(\mathbf{s}_0) = var[\varepsilon(\mathbf{s}_0)] \cdot \left[1 + \mathbf{x}_0^{\mathrm{T}} \cdot (\mathbf{X}^{\mathrm{T}} \cdot \mathbf{X})^{-1} \cdot \mathbf{x}_0\right]$
- Prediction accuracy (CCC): $\rho_c = \dfrac{2 \cdot \rho \cdot \sigma_{\hat{y}} \cdot \sigma_y}{\sigma_{\hat{y}}^2 + \sigma_y^2 + (\mu_{\hat{y}} - \mu_y)^2}$
- Accuracy of prediction error: s.d. of $z_{score}(\mathbf{s}_j) = \dfrac{\hat{y}(\mathbf{s}_j) - y(\mathbf{s}_j)}{\sigma(\mathbf{s}_j)}$

# *How I met Machine Learning*

```r
sv_lm  <- single_variable(explainer_lm, variable =  "construction.year", type = "pdp")

plot(sv_rf, sv_lm)
```



Computing

# The Extraordinary Link Between Deep Neural Networks and the Nature of the Universe

Nobody understands why deep neural networks are so good at solving complex problems. Now physicists say the secret is buried in the laws of physics.

by Emerging Technology from the arXiv September 9, 2016

**In the last couple of years, deep learning techniques have transformed the**

# Anthony Goldbloom gives you the secret to winning Kaggle competitions

January 13, 2016 Andrew Fogg Big Data

Kaggle has become the premier Data Science competition where the best and the brightest turn out in droves – Kaggle has more than 400,000 users – to try and claim the glory. With so many Data Scientists vying to win each competition (around 100,000 entries/month), prospective entrants can use all the tips they can get.

And who better than Kaggle CEO and Founder, Anthony Goldbloom, to dish out that advice? We caught up with him at Extract SF 2015 in October to pick his brain about how best to approach a Kaggle competition.

## Handcrafted feature engineering

This approach works best if you already have an intuition as to what's in the data. First, a competitor will take the data and plot histograms and such to explore what's in it. Then they'll spend a lot time generating features and testing which ones really do correlate with the target variable.

For example, a chain of used car dealers wanted to predict which cars sold at a second-hand auction would be good buys and which ones would be lemons. After much trial and error, by many different applicants, it turned out that one of the most predictive features was the car color. By grouping standard color cars and unreliable colored cars, they found that unusual colored cars were more likely to be reliable.

The way they found this answer was to test lots and lots and lots of hypotheses. The vast majority of them didn't work out, but the one that did won them the competition.

As long as Kaggle has been around, Anthony says, it has almost always been ensembles of decision trees that have won competitions.

It used to be random forest that was the big winner, but over the last six months a new algorithm called XGboost has cropped up, and it's winning practically every competition in the structured data category.

OPEN ACCESS PEER-REVIEWED

RESEARCH ARTICLE

# SoilGrids250m: Global gridded soil information based on machine learning

Tomislav Hengl, Jorge Mendes de Jesus, Gerard B. M. Heuvelink, Maria Ruiperez Gonzalez, Milan Kilibarda, Aleksandar Blagotić, Wei Shangguan, Marvin N. Wright, Xiaoyuan Geng, Bernhard Bauer-Marschallinger, Mario Antonio Guevara, Rodrigo Vargas, Robert A. MacMillan, [ ... ], Bas Kempen [ view all ]

Published: February 16, 2017 • https://doi.org/10.1371/journal.pone.0169748







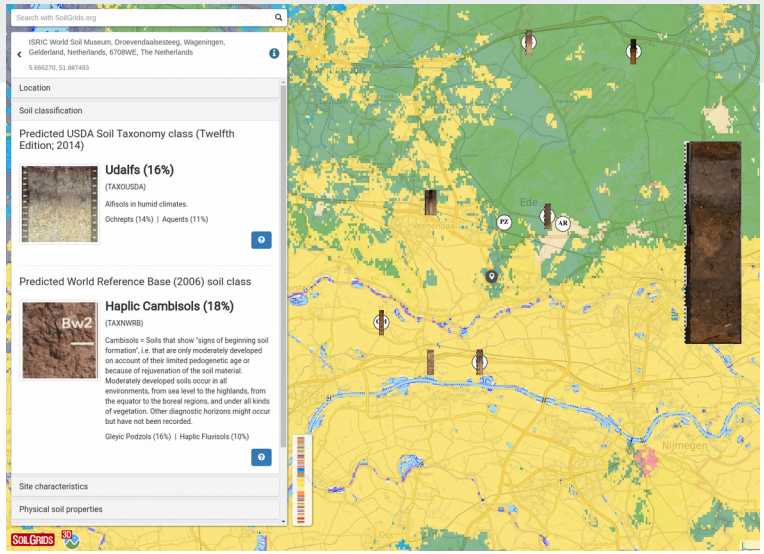
## Abstract

This paper describes the technical development and accuracy assessment of the most recent and improved version of the SoilGrids system at 250m resolution (June 2016 update). SoilGrids provides global predictions for standard numeric soil properties (organic carbon, bulk density, Cation Exchange Capacity (CEC), pH, soil texture fractions and coarse fragments) at seven standard depths (0, 5, 15, 30, 60, 100 and 200 cm), in addition to predictions of depth to bedrock and distribution of soil classes based on the World Reference Base (WRB) and USDA classification systems (ca. 280 raster layers in total). Predictions were based on ca. 150,000 soil profiles used for training and a stack of 158 remote sensing-based soil covariates (primarily derived from MODIS land products, SRTM DEM derivatives, climatic images and global landform and lithology maps), which were used to fit an ensemble of machine learning methods—random forest and gradient boosting and/or multinomial logistic regression—as implemented in the R packages `ranger`, `xgboost`, `nnet` and `caret`. The results of 10–fold cross-validation show that the ensemble models explain between 56% (coarse fragments) and 83% (pH) of variation with an overall average of 61%. Improvements in the relative accuracy considering the amount of variation explained, in comparison to the previous version of SoilGrids at 1 km spatial resolution, range from 60 to 230%. Improvements can be attributed to: (1) the use of machine learning instead of linear regression, (2) to considerable investments in preparing finer resolution covariate layers and (3) to insertion of additional soil profiles. Further development of



### Subject Areas

Shannon index | Forecasting | Machine learning | Trees | Agricultural soil sci... | Remote sensing | Glaciers | Soil ecology




# Global mapping of potential natural vegetation: an assessment of Machine Learning algorithms for estimating land potential

Research article | Biogeography | Computational Biology | Plant Science | Data Mining and Machine Learning | Spatial and Geographic Information Science

Tomislav Hengl[1], Markus G Walsh[2], Jonathan Sanderman[3], Ichsani Wheeler[4], Sandy P Harrison[5], Iain C Prentice[6]

March 30, 2018

## Author and article information

## Abstract

Potential Natural Vegetation (PNV) is the vegetation cover in equilibrium with climate, that would exist at a given location non-impacted by human activities. PNV is useful for raising public awareness about land degradation and for estimating land potential. This paper presents results of assessing Machine Learning Algorithms (MLA) for operational mapping of Potential Natural Vegetation (PNV). The following MLA were considered: neural

Journal of Statistical Software

March 2017, Volume 77, Issue 1. doi: 10.18637/jss.v077.i01

# ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R

**Marvin N. Wright**
Universität zu Lübeck

**Andreas Ziegler**
Universität zu Lübeck
University of KwaZulu-Natal

**Abstract**

We introduce the C++ application and R package **ranger**. The software is a fast implementation of random forests for high dimensional data. Ensembles of classification, regression and survival trees are supported. We describe the implementation, provide

Applications ▾ Terminal ▾ Don Jan 25, 16:13 9,2 °C hr ▾

Not secure | 145.239.142.3:8787

Bookmarks | Amazon RStudio | RStudio | VideoSta...

File Edit Code View Plots Session Build

LDN_SOC_change.R × | PNV_mapping.R ×

Source on Save

```
/Forest_TYPE_MAP_2006_250m.sdat -of \"SAGA\"
479 tile.pol.eu = spTransform(tile.pol, CRS("+pr
    +ellps=GRS80 +units=m +no_defs"))
480 writeOGR(tile.pol.eu, "/data/PNV/Data/EU_for
    ")
481 system(paste0('saga_cmd -c=64 shapes_grid 2
    m.sgrd\" -POLYGONS=\"/data/PNV/Data/EU_fores
    /Data/EU_forest/ov_tiles_eu.shp\"'))
482 ov_tiles_eu = readOGR("/data/PNV/Data/EU_for
483 str(ov_tiles_eu@data)
484 summary(sel$.eu <- (!ov_tiles_eu$Forest_TYPE
485 pr.dirs.eu = paste0("T", ov_tiles_eu$ID[whic
486 ## 995 tiles only
487
488 ## Takes 30 mins without errors
489 m.eu.trees = readRDS.gz("m.eu.trees.rds")
490 col.legend.for = data.frame(Group=eu.sp$sp,
491 cl <- parallel::makeCluster(8, type="FORK")
492 x = parallel::parLapply(cl, pr.dirs.eu, func
    .tbl, col.legend=col.legend.for, varn="EUfor
493 parallel::stopCluster(cl)
494
495
```

489:13 EU forest predict ÷

Console /data/LDN/

```
Copying nodata values from source /data/GEOG/OCSTHA
0...10...20...30...40...50...60...70...80...90...10
> save.image()

Restarting R session...

Microsoft R Open 3.4.1
The enhanced R distribution from Microsoft
Microsoft packages Copyright (C) 2017 Microsoft Cor

Using the Intel MKL for parallel mathematical compu

Default CRAN mirror snapshot taken on 2017-09-01.
See: https://mran.microsoft.com/.

> setwd("/data/PNV/R_code")
> load(".RData")
```

ubuntu@ip-172-31-42-90: ~

File Edit View Search Terminal Help

```
  1  [||||||||||||||100.0%]   17 [||||||||||||||100.0%]   33 [||||||||||||||100.0%]   49 [||||||||||||||100.0%]
  2  [||||||||||||||100.0%]   18 [||||||||||||||100.0%]   34 [||||||||||||||100.0%]   50 [||||||||||||||100.0%]
  3  [||||||||||||||100.0%]   19 [||||||||||||||100.0%]   35 [||||||||||||||100.0%]   51 [||||||||||||||100.0%]
  4  [||||||||||||||100.0%]   20 [||||||||||||||100.0%]   36 [||||||||||||||100.0%]   52 [||||||||||||||100.0%]
  5  [||||||||||||||100.0%]   21 [||||||||||||||100.0%]   37 [||||||||||||||100.0%]   53 [||||||||||||||100.0%]
  6  [||||||||||||||100.0%]   22 [||||||||||||||100.0%]   38 [||||||||||||||100.0%]   54 [||||||||||||||100.0%]
  7  [||||||||||||||100.0%]   23 [||||||||||||||100.0%]   39 [||||||||||||||100.0%]   55 [||||||||||||||100.0%]
  8  [||||||||||||||100.0%]   24 [||||||||||||||100.0%]   40 [||||||||||||||100.0%]   56 [||||||||||||||100.0%]
  9  [||||||||||||||100.0%]   25 [||||||||||||||100.0%]   41 [||||||||||||||100.0%]   57 [||||||||||||||100.0%]
 10  [||||||||||||||100.0%]   26 [||||||||||||||100.0%]   42 [||||||||||||||100.0%]   58 [||||||||||||||100.0%]
 11  [||||||||||||||100.0%]   27 [||||||||||||||100.0%]   43 [||||||||||||||100.0%]   59 [||||||||||||||100.0%]
 12  [||||||||||||||100.0%]   28 [||||||||||||||100.0%]   44 [||||||||||||||100.0%]   60 [||||||||||||||100.0%]
 13  [||||||||||||||100.0%]   29 [||||||||||||||100.0%]   45 [||||||||||||||100.0%]   61 [||||||||||||||100.0%]
 14  [||||||||||||||100.0%]   30 [||||||||||||||100.0%]   46 [||||||||||||||100.0%]   62 [||||||||||||||100.0%]
 15  [||||||||||||||100.0%]   31 [||||||||||||||100.0%]   47 [||||||||||||||100.0%]   63 [||||||||||||||100.0%]
 16  [||||||||||||||100.0%]   32 [||||||||||||||100.0%]   48 [||||||||||||||100.0%]   64 [||||||||||||||100.0%]
 Mem[|||||||||                                  31.7G/252G]  Tasks: 98, 53 thr; 65 running
 Swp[                                             0K/0K]  Load average: 63.67 39.60 18.64
                                                          Uptime: 06:39:28

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
58699 ubuntu     20   0 1138M  520M 43160 R 100.  0.2  4:44.17 /opt/microsoft/ropen/3.4.2/lib64/R/bin/exec/R --slave --no
58219 ubuntu     20   0 1135M  518M 43056 R 100.  0.2  4:43.98 /opt/microsoft/ropen/3.4.2/lib64/R/bin/exec/R --slave --no
58209 ubuntu     20   0 1135M  517M 42796 R 100.  0.2  4:44.36 /opt/microsoft/ropen/3.4.2/lib64/R/bin/exec/R --slave --no
58359 ubuntu     20   0 1134M  517M 43380 R 100.  0.2  4:44.43 /opt/microsoft/ropen/3.4.2/lib64/R/bin/exec/R --slave --no
58349 ubuntu     20   0 1134M  517M 42972 R 100.  0.2  4:44.29 /opt/microsoft/ropen/3.4.2/lib64/R/bin/exec/R --slave --no
58609 ubuntu     20   0 1137M  521M 43108 R 100.  0.2  4:44.50 /opt/microsoft/ropen/3.4.2/lib64/R/bin/exec/R --slave --no
58519 ubuntu     20   0 1128M  511M 43052 R 100.  0.2  4:44.06 /opt/microsoft/ropen/3.4.2/lib64/R/bin/exec/R --slave --no
58729 ubuntu     20   0 1177M  559M 43116 R 100.  0.2  4:44.43 /opt/microsoft/ropen/3.4.2/lib64/R/bin/exec/R --slave --no
58649 ubuntu     20   0 1138M  521M 43100 R 100.  0.2  4:44.41 /opt/microsoft/ropen/3.4.2/lib64/R/bin/exec/R --slave --no
58709 ubuntu     20   0 1137M  520M 43444 R 100.  0.2  4:43.96 /opt/microsoft/ropen/3.4.2/lib64/R/bin/exec/R --slave --no
58339 ubuntu     20   0 1135M  517M 42740 R 100.  0.2  4:44.40 /opt/microsoft/ropen/3.4.2/lib64/R/bin/exec/R --slave --no
58129 ubuntu     20   0 1135M  518M 43312 R 100.  0.2  4:43.15 /opt/microsoft/ropen/3.4.2/lib64/R/bin/exec/R --slave --no
58149 ubuntu     20   0 1134M  519M 42696 R 99.7  0.2  4:43.66 /opt/microsoft/ropen/3.4.2/lib64/R/bin/exec/R --slave --no
58459 ubuntu     20   0 1131M  516M 42808 R 99.7  0.2  4:44.25 /opt/microsoft/ropen/3.4.2/lib64/R/bin/exec/R --slave --no
58289 ubuntu     20   0 1109M  492M 43404 R 99.7  0.2  4:44.05 /opt/microsoft/ropen/3.4.2/lib64/R/bin/exec/R --slave --no
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

Documentation

**Arguments**

| | |
|---|---|
| cl | a cluster object, created by this package or by package snow. If NULL, use the registered default cluster. |

# *MLA's are interesting for generating sp, however...*

Spatial auto-correlation, especially if still existent in the cross-validation residuals, indicates that the predictions are maybe biased, and this is suboptimal.

To account for this, we developed a technique called RFsp (as implemented in the ranger package) which uses a combination of Reflectance (R), process-based (P) covs and geographical distances (G) to sampling locations:

$$Y(\mathbf{s}) = f(\mathbf{X}_G, \mathbf{X}_R, \mathbf{X}_P) \tag{18}$$

where $\mathbf{X}_G$ are covariates accounting for geographical proximity and spatial relations between observations

# *Meet a kriging alternative*

# *Standard steps in mb geostatistics*

Usually, about 5-6 inputs required from the analyst:

- Determine distribution of the target variable and appropriate transformation (normal, log-normal, zero-inflated, Gamma, Poissonic ...)
- Fit variogram (WLS, REML, ...), deal with multicolinearity (PCA?), non-stationary properties, support size, mixed effects…
- Predict (mean values and uncertainty) limit the search radius
- Validate predictions (mapping accuracy)

# *Vgm modeling and predictions (kriging)*

```
R> zinc.vgm <- likfit(zinc.geo, lambda = 0,
ini=c(var(log1p(zinc.geo$data)), 500), cov.model =
"exponential")
```

```
R> zinc.ok <- krige.conv(zinc.geo, locations = locs, krige
= krige.control(obj.m = zinc.vgm))
```

krige.conv: model with constant mean

krige.conv: performing the Box-Cox data transformation

krige.conv: back-transforming the predicted mean and variance

krige.conv: Kriging performed using global neighbourhood

*Journal of Statistical Software*

*January 2015, Volume 63, Issue 12.* http://www.jstatsoft.org/

# Model-Based Geostatistics the Easy Way

**Patrick E. Brown**
Cancer Care Ontario and University of Toronto

**Abstract**

This paper briefly describes geostatistical models for Gaussian and non-Gaussian data and demonstrates the **geostatsp** and **dieasemapping** packages for performing inference using these models. Making use of R's spatial data types, and raster objects in particular, makes spatial analyses using geostatistical models simple and convenient. Examples using real data are shown for Gaussian spatial data, binomially distributed spatial data, a log-

# *RFsp with geographical covariates only (OK-alt.)*

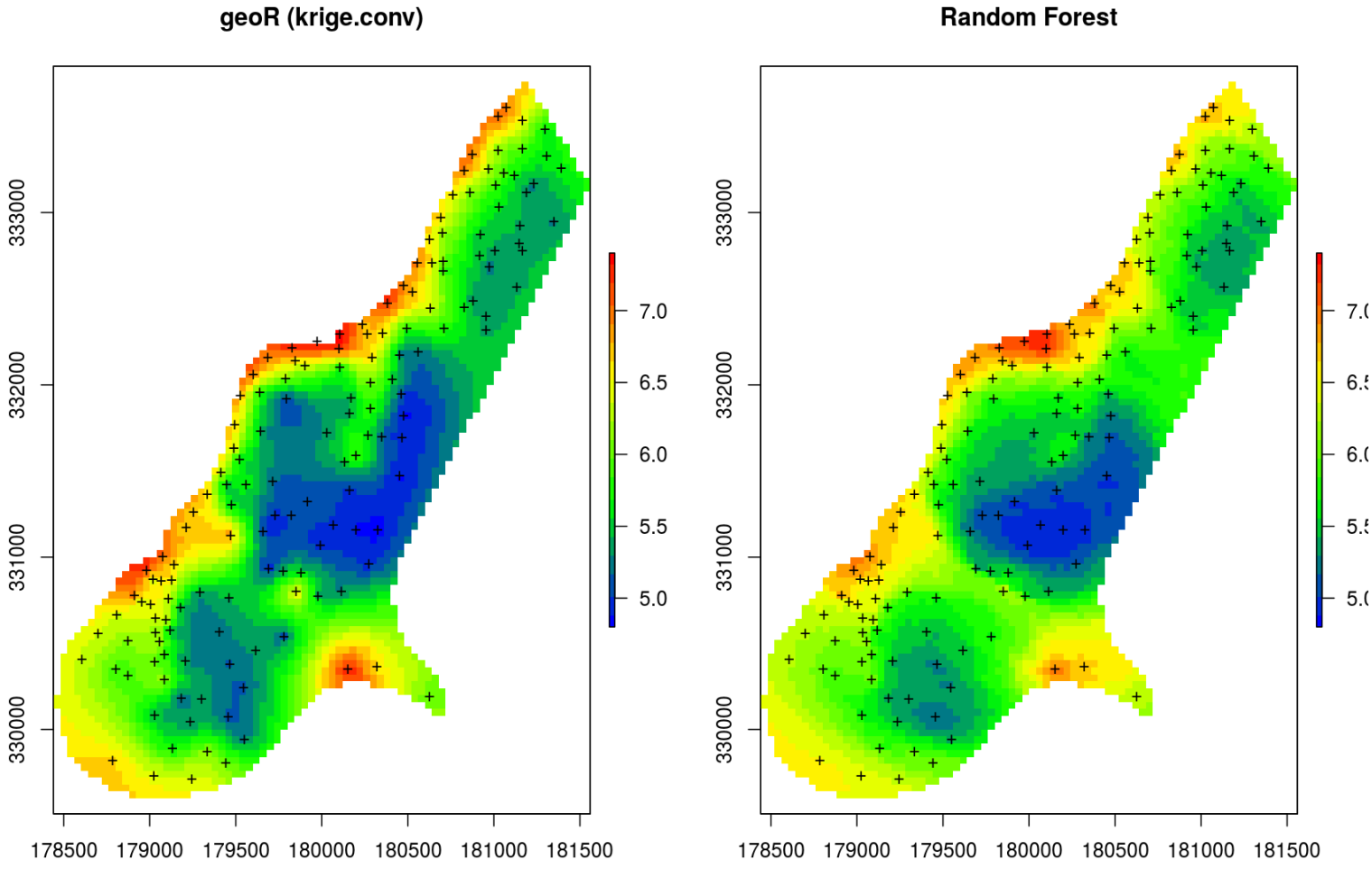
```
R> grid.dist0 <- buffer.dist(meuse["zinc"], meuse.grid[1],
as.factor(1:nrow(meuse)))
```

```
R> ov.zinc <- over(meuse["zinc"], grid.dist0)
```

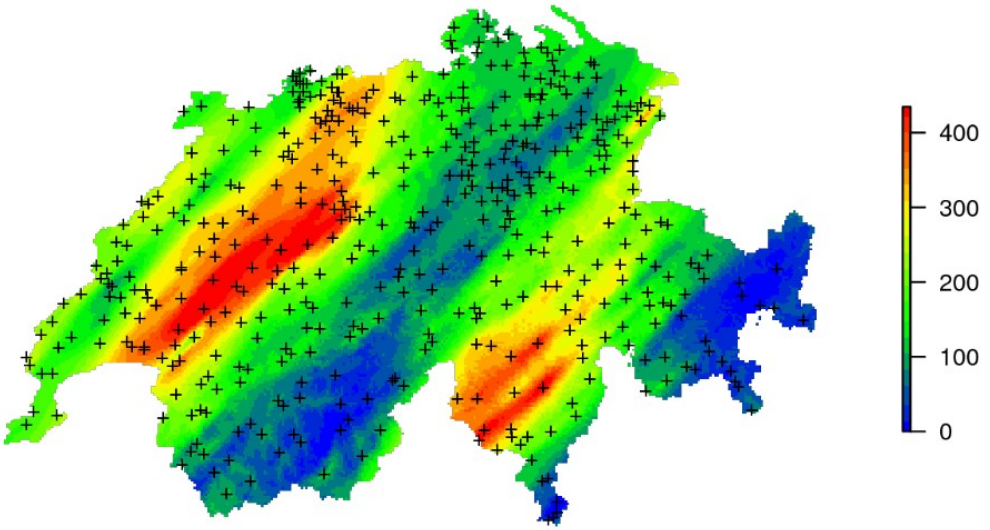
```
R> m.zinc <- ranger(as.formula(paste("zinc ~",
paste(names(grid.dist0), collapse="+")),
cbind(meuse@data["zinc"], ov.zinc))
## zinc ~ layer.1 + layer.2 + … + layer.155
```

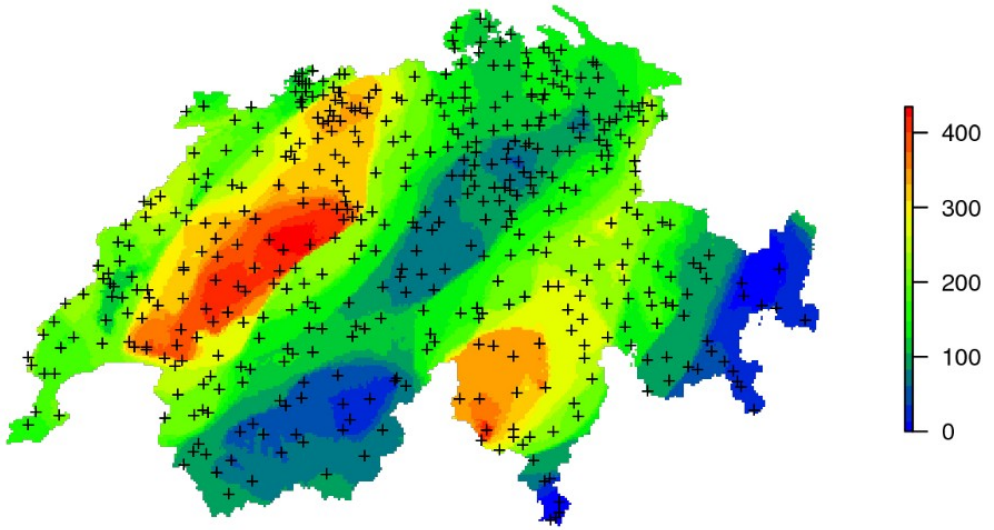
```
R> zinc.rfd <- predict(m.zinc, grid.dist0@data)
```

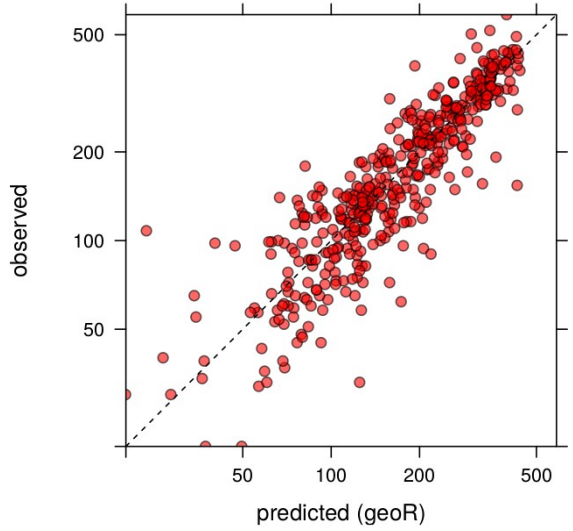
# *Meuse data set — Zinc*

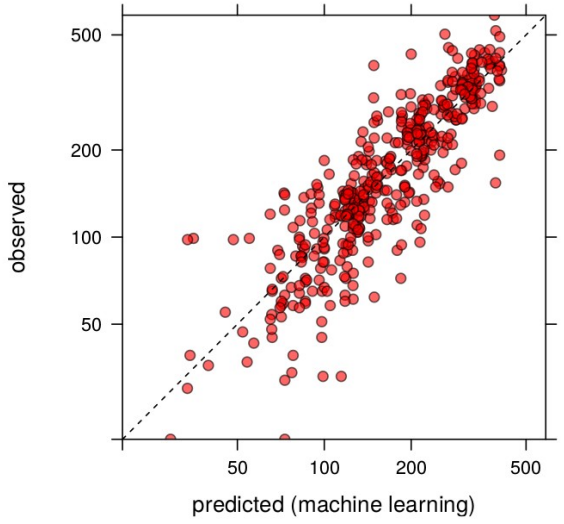


# *SIC97 data set — daily rainfall (UK vs RFsp)*

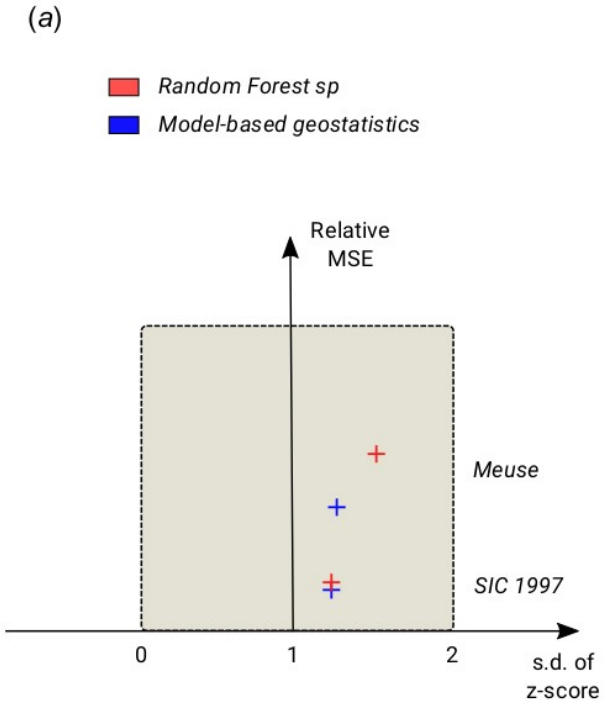


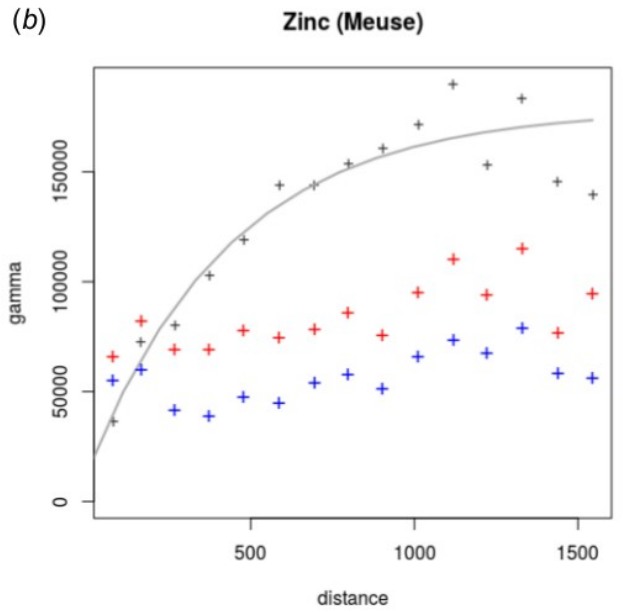



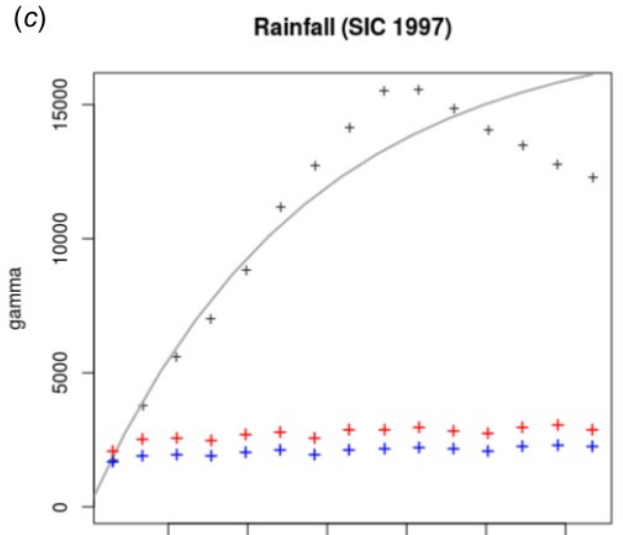



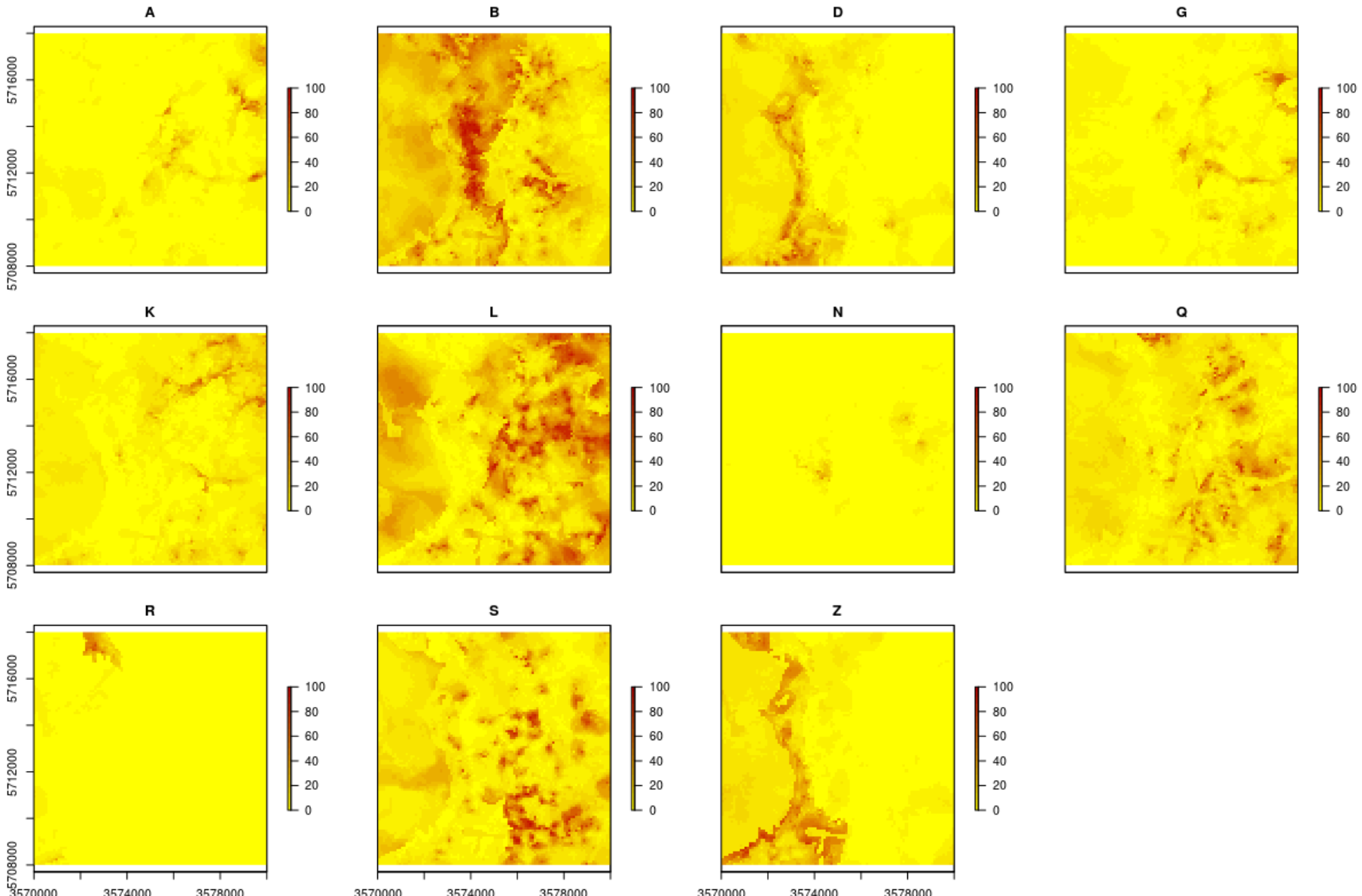



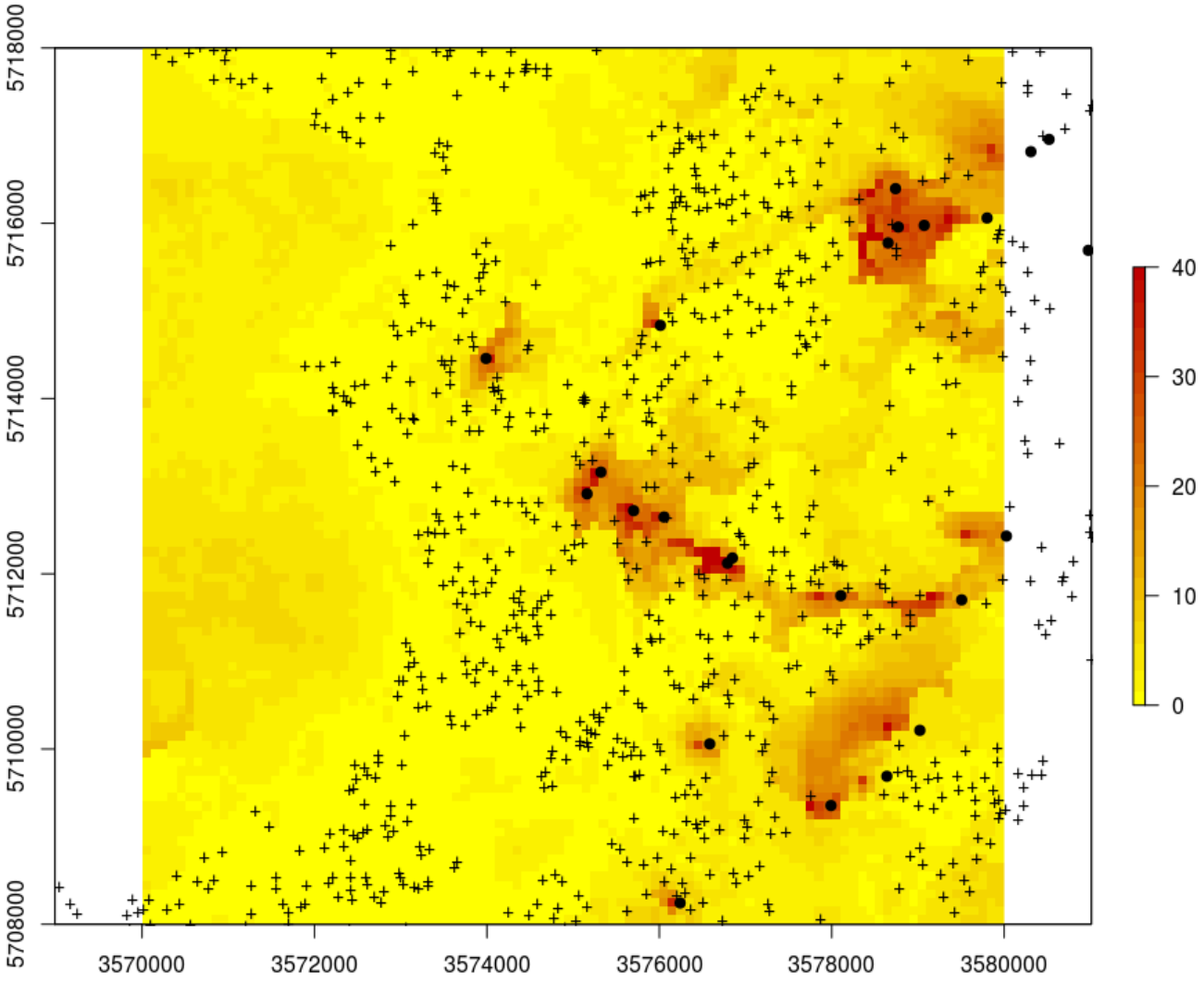
# Plots


(a)
Random Forest sp
Model-based geostatistics
Relative MSE
Meuse
SIC 1997
0
1
2
s.d. of z-score

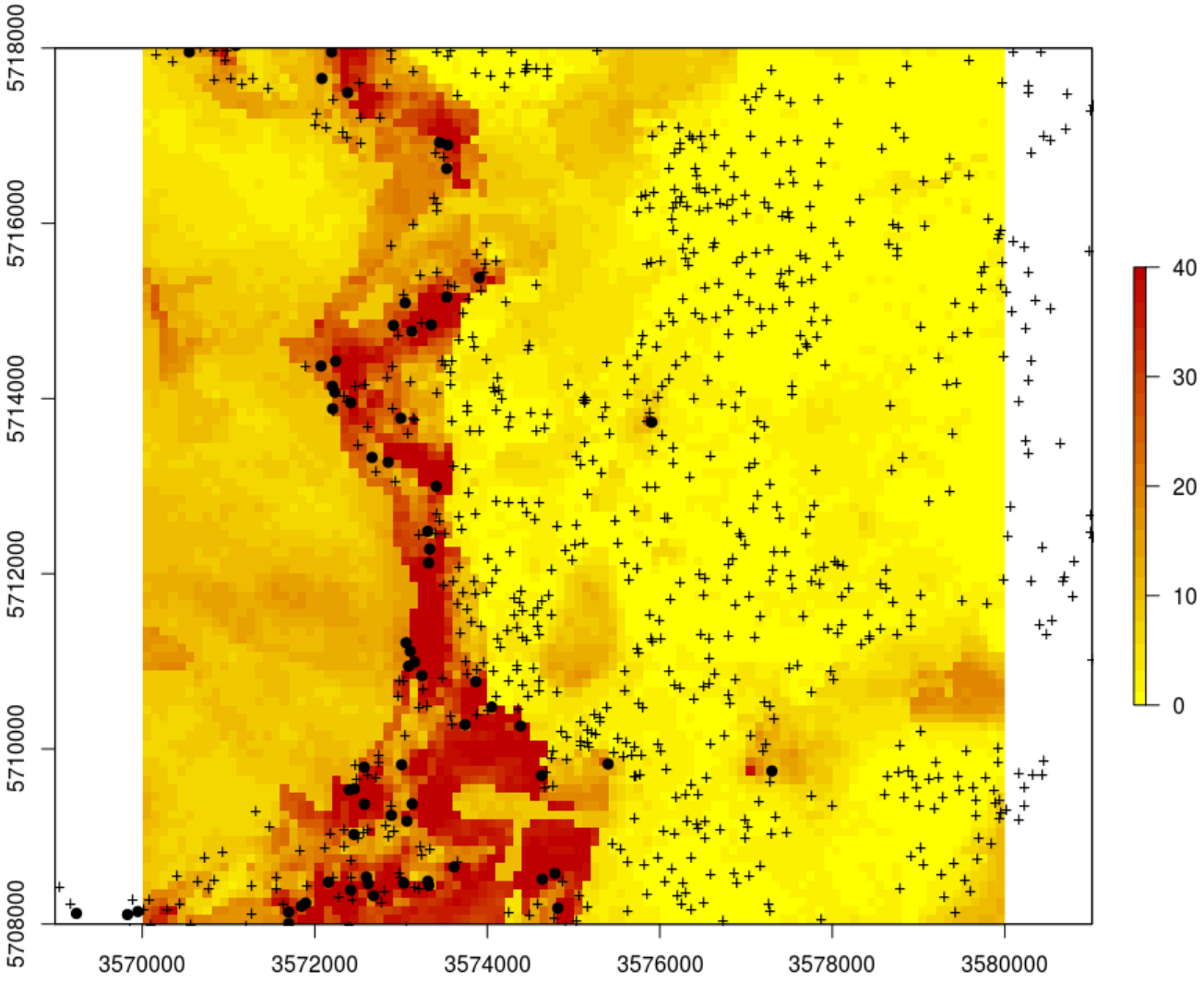

(b)
Zinc (Meuse)
gamma
0
50000
100000
150000
500
1000
1500
distance


(c)
Rainfall (SIC 1997)
gamma
0
5000
10000
15000


A
B
D
G
K
L
N
Q
R
S
Z
5716000
5712000
5708000
3570000
3574000
3578000
100
80
60
40
20
0

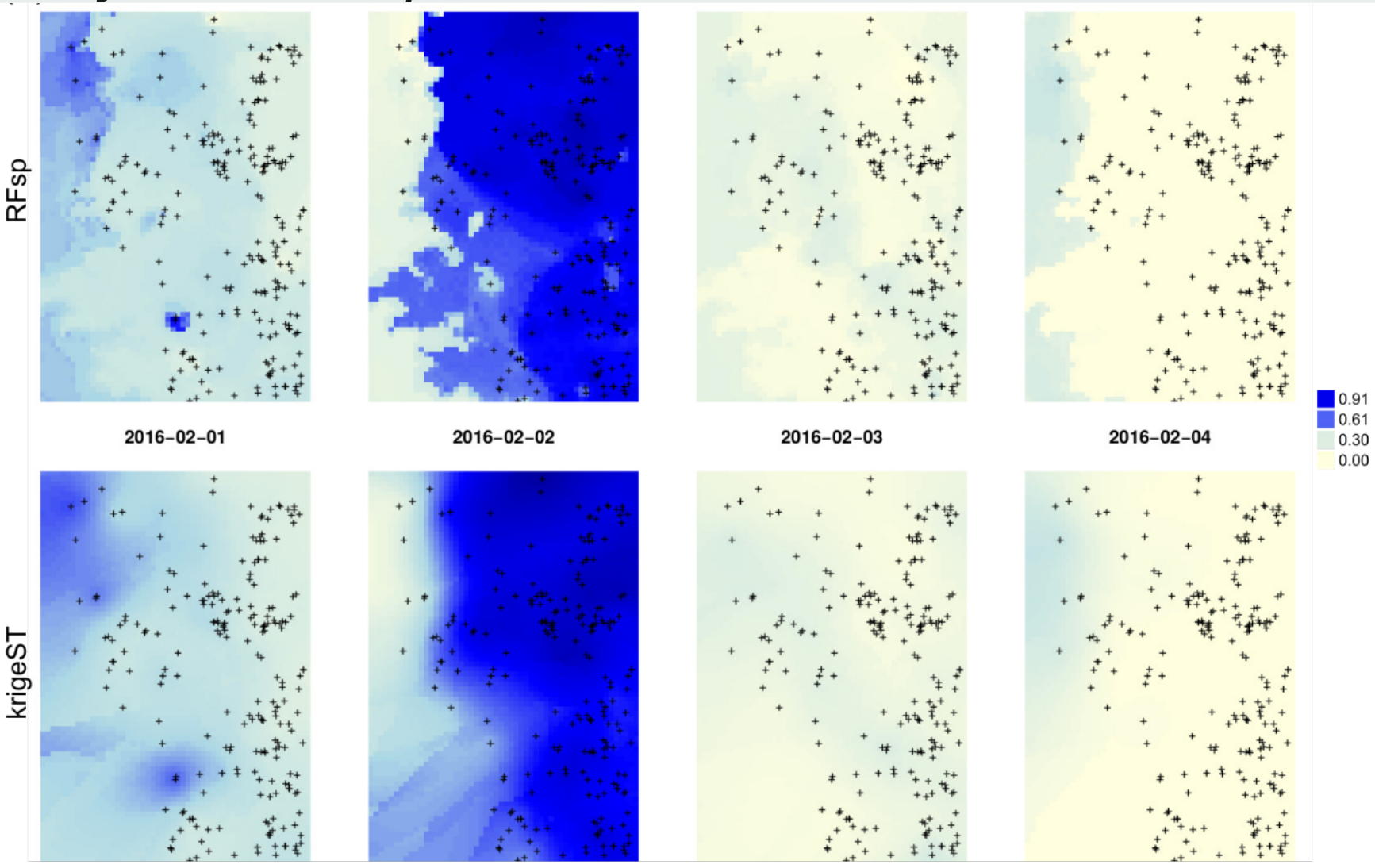

5718000
5716000
5714000
5712000
5710000
5708000
3570000
3572000
3574000
3576000
3578000
3580000
40
30
20
10
0

# Class G


5718000
5716000
5714000
5712000
5710000
5708000
3570000
3572000
3574000
3576000
3578000
3580000
40
30
20
10
0

# Class D

# *Daily rainfall spacetime*



# *But what about uncertainty in RFsp?*

# Thanks to R-sig-geo

**Tomislav Hengl-4** ▶Jun 23, 2013; 5:51pm **Prediction variance (map) for predictions derived using RandomForest pac** Reply | Threaded | More

153 posts

Dear list,

I have a question about the randomForest models. I'm trying to figure out a way to estimate the prediction variance (spatially) for the randomForest function (http://cran.r-project.org/web/packages/randomForest/).

If I run a GLM I can also derive the prediction variance using:

```
 > demo(meuse, echo=FALSE)
 > meuse.ov <- over(meuse, meuse.grid)
 > meuse.ov <- cbind(meuse.ov, meuse@data)
 > omm0 <- glm(log1p(om)~dist+ffreq, meuse.ov, family=gaussian())
 > om.glm <- predict.glm(omm0, meuse.grid, se.fit=TRUE)
 > str(om.glm)
List of 3
  $ fit          : Named num [1:3103] 2.34 2.34 2.32 2.29 2.34 ...
   ..- attr(*, "names")= chr [1:3103] "1" "2" "3" "4" ...
  $ se.fit       : Named num [1:3103] 0.0491 0.0491 0.0481 0.046 0.0491 ...
   ..- attr(*, "names")= chr [1:3103] "1" "2" "3" "4" ...
  $ residual.scale: num 0.357
```

when I fit a randomForest model, I do not get any estimate of the model uncertainty (for each pixel) but just the predictions:

```
 > meuse.ov <- meuse.ov[-omm0$na.action,]
```

# Thanks to Forrest Stevens

**Tomislav Hengl-4** Jun 25, 2013; 12:44am **Re: Prediction variance (map) for predictions derived using RandomForest** | Reply | Threaded | More

153 posts

In reply to this post by Forrest Stevens

Dear Forrest,

Thanks a lot for your tip. I think quantregForest is what we were looking for. It takes much more time to compute, but the method looks sound (http://jmlr.org/papers/volume7/meinshausen06a/meinshausen06a.pdf). I do simplify everything on the end and assume that I can derive upper and lower confidence limits for +/- 1 s.d. (0.15866, 1-0.15866) and then use this as the prediction variance, but this is probably as good as it goes. Here is the revised code:

https://code.google.com/p/gsif/source/browse/trunk/meuse/RK_vs_RandomForestK.R

Thank you all for your suggestions / opinions (very useful as usual).

cheers,

T. (Tom) Hengl
Url: http://www.wageningenur.nl/en/Persons/dr.-T-Tom-Hengl.htm
Network: http://profiles.google.com/tom.hengl
Publications: http://scholar.google.com/citations?user=2oYU7S8AAAAJ

On 23/06/2013 15:08, Forrest Stevens wrote:

> Hi Tom, I've done something similar in the past to visualize the
> distribution of the predictions attained for each observation across

# *There are in fact several approaches*

The uncertainty of the predictions of random forest for regression-type problems can be estimated using several approaches:

- The Jackknife-after-Bootstrap method (see e.g. Wager et al. (2014)).
- The U-statistics approach of Mentch and Hooker (2016).
- The Monte Carlo simulations (both target variable and covariates) approach of Coulston et al. (2016).
- The Quantile Regression Forests (QRF) method (Meinshausen, 2006).

Journal of Machine Learning Research 7 (2006) 983–999 Submitted 10/05; Revised 2/06; Published 6/06

# Quantile Regression Forests

**Nicolai Meinshausen** NICOLAI@STAT.MATH.ETHZ.CH
*Seminar für Statistik*
*ETH Zürich*
*8092 Zürich, Switzerland*

**Editor:** Greg Ridgeway

## Abstract

Random forests were introduced as a machine learning tool in Breiman (2001) and have since proven to be very popular and powerful for high-dimensional regression and classification. For regression, random forests give an accurate approximation of the conditional mean of a response variable. It is shown here that random forests provide information

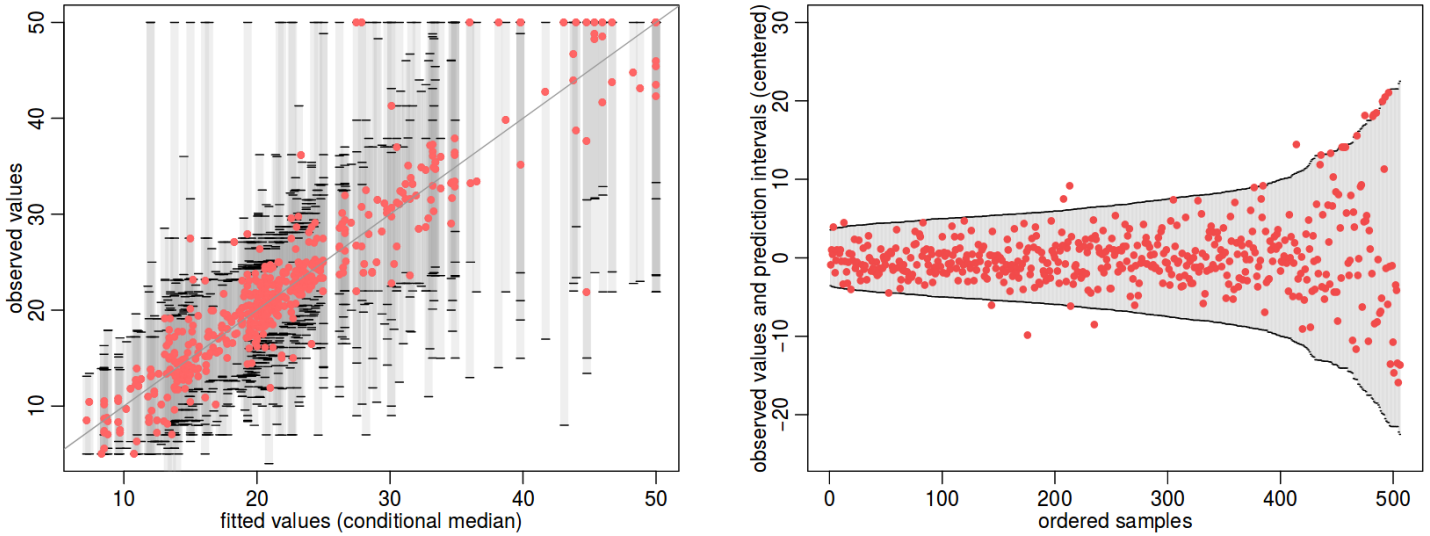
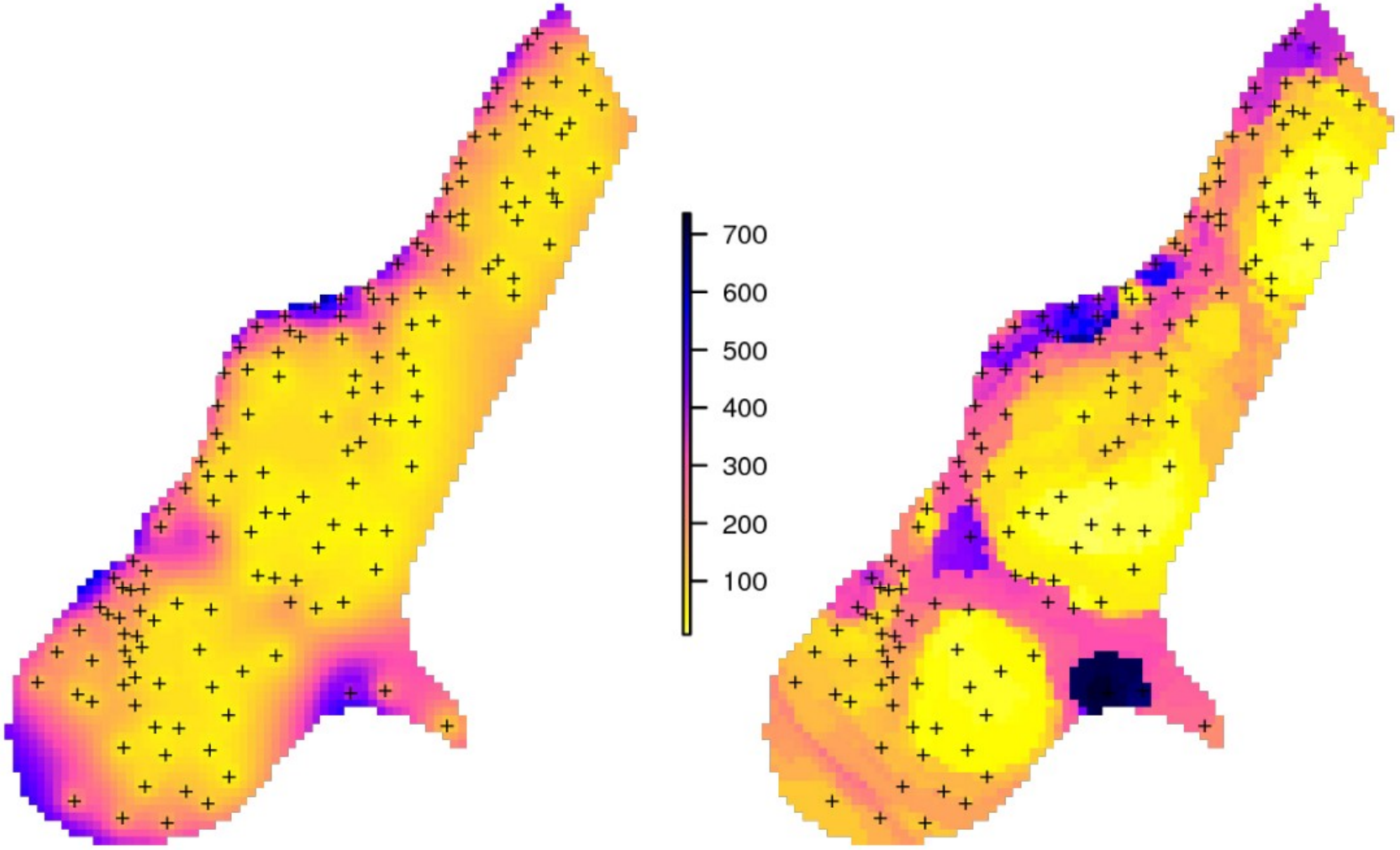


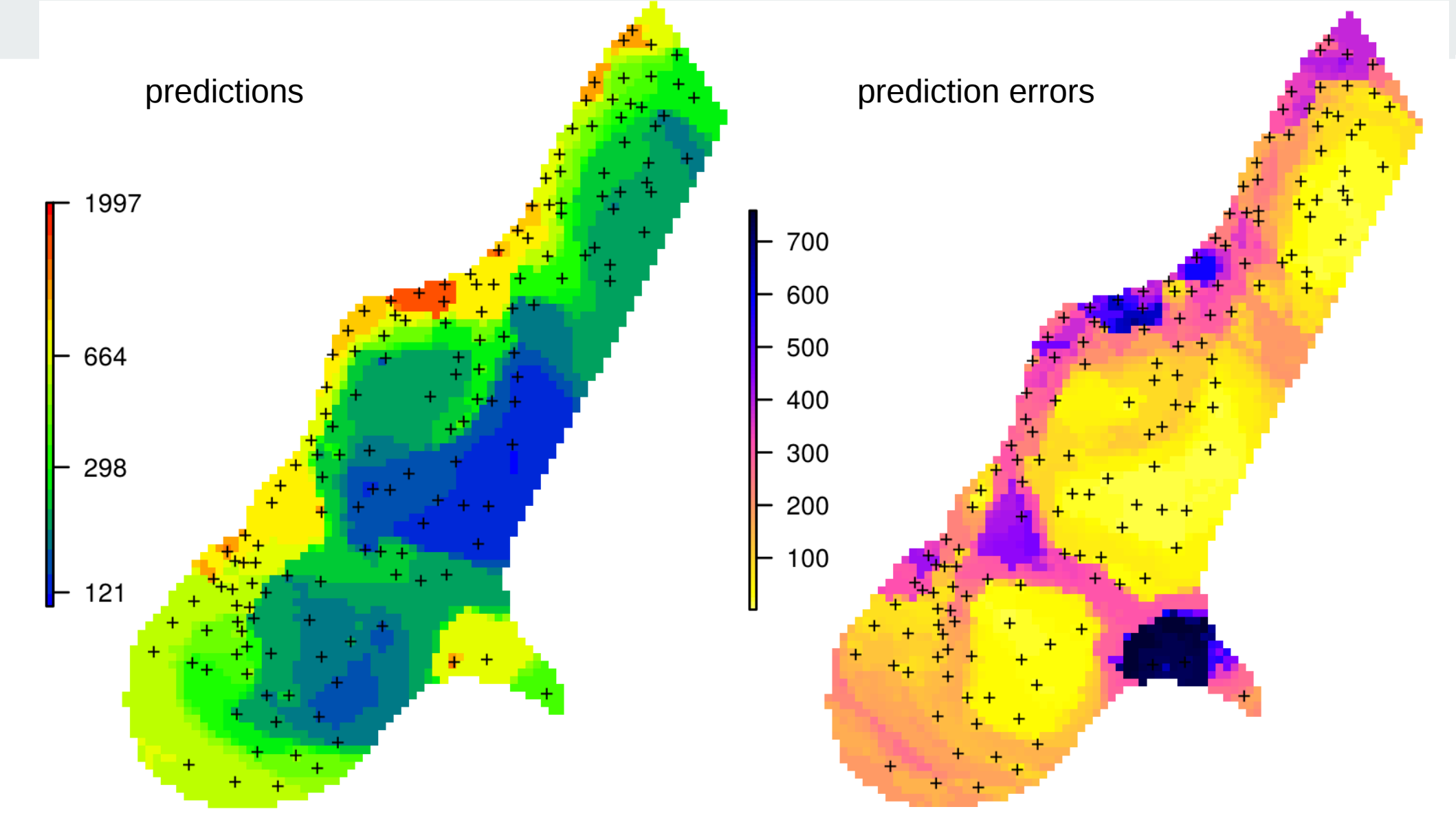

Figure 3: For each data point $i = 1, \ldots, n$ in the Boston Housing data set (with $n = 506$), conditional quantiles are estimated with QRF on a test set which does not include the $i$-th observation (5-fold cross-validation). Left panel: the observed values are

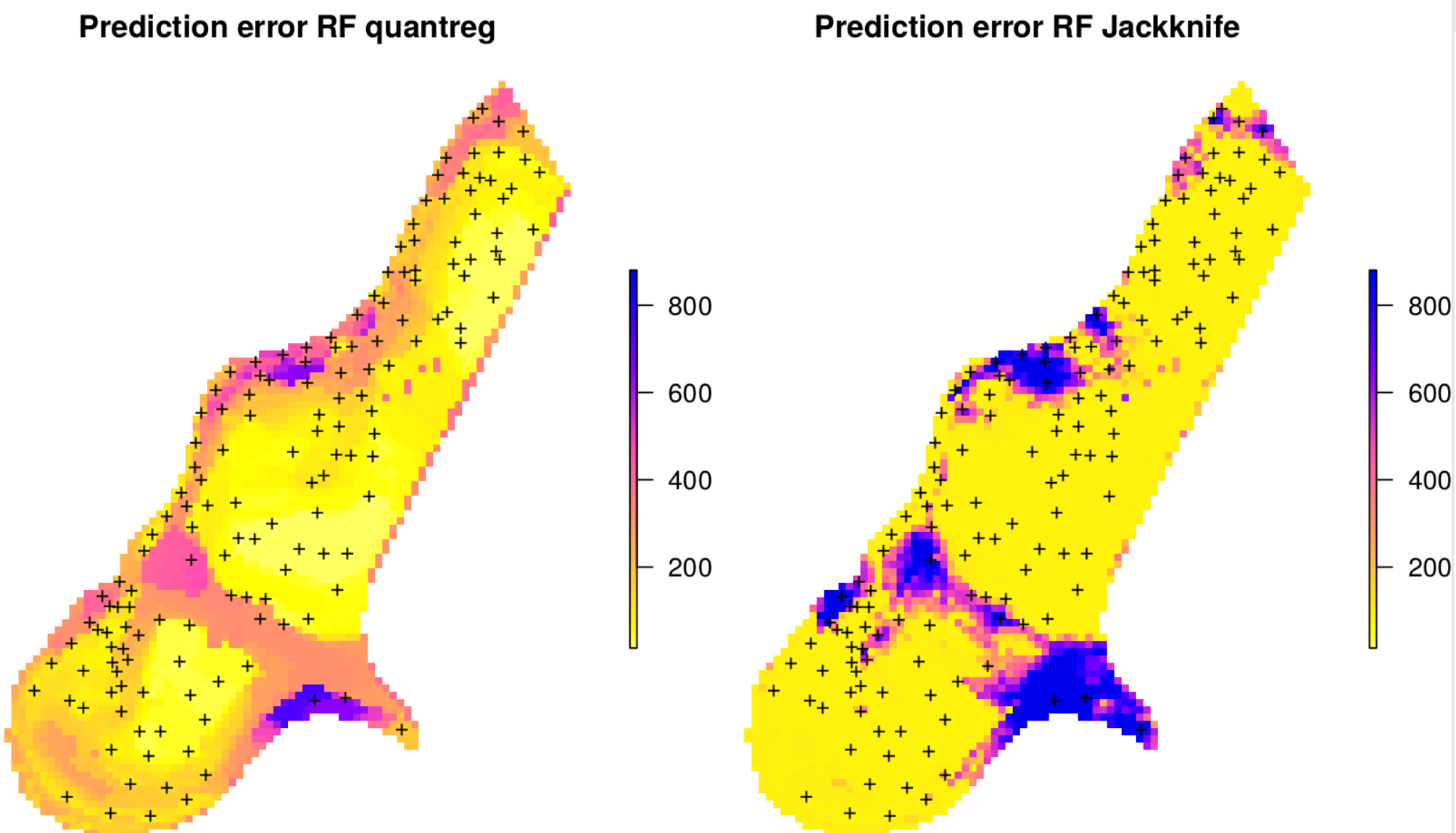
$$\sigma_{QRF}(\mathbf{s}_j) \approx \frac{\hat{y}_{q=0.841}(\mathbf{s}_j) - \hat{y}_{q=0.159}(\mathbf{s}_j)}{2} \tag{23}$$

This formula assumes that the prediction errors are symmetrical at each new prediction location, which might not always be the case.

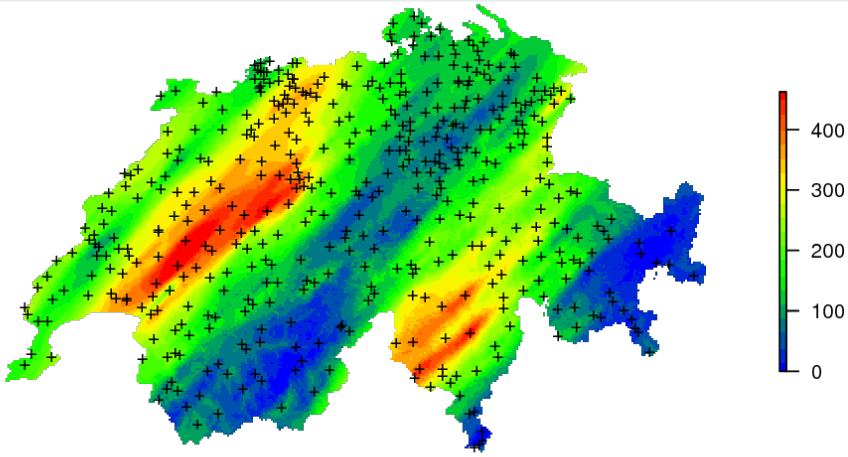
# *Meuse prediction error*



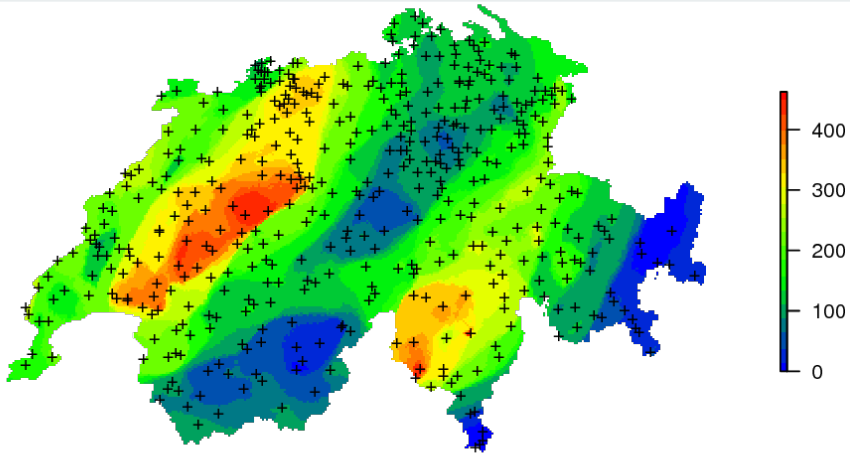

predictions
1997
664
298
121
prediction errors
700
600
500
400
300
200
100


Prediction error RF quantreg
800
600
400
200
Prediction error RF Jackknife
800
600
400
200

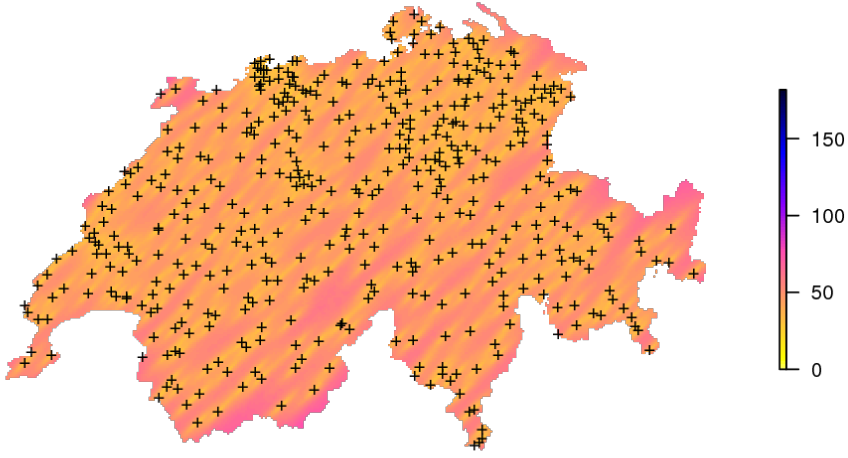
Universal kriging (UK)



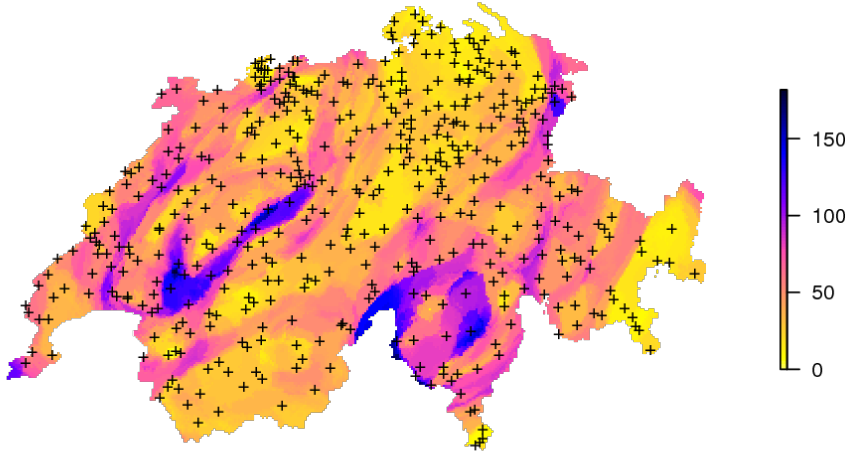

Random Forest (RF)




Universal kriging (UK) prediction error




Random Forest (RF) prediction error



# *Vgm modeling and predictions (kriging)*

- No stationarity requirements.
- No Normal distribution requirements.
- No problems with choosing the right variogram (in fact, RFsp does not need vgm at all!).
- No (serious) problems with hot-spots.
- More complex distances can be added.

# *Possibilities for incorporating more complex dist.*

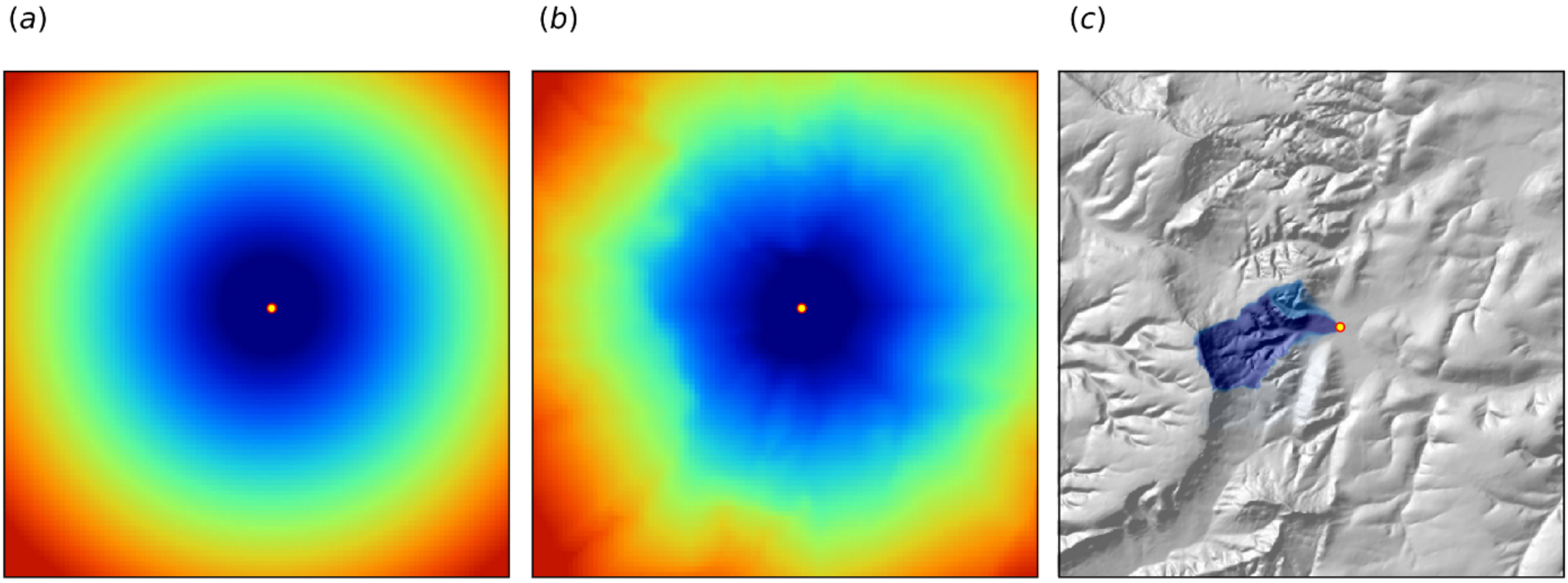



**Figure 2.** Examples of distance maps to some location in space (yellow dot) based on different derivation algorithms: (a) simple Euclidean distances, (b) complex speed-based distances based on the gdistance package and Digital Elevation Model (DEM) (van Etten, 2017), and (c) upslope area derived based on the DEM in SAGA GIS (Conrad et al., 2015). Case study: Ebergötzen (Böhner et al., 2006).

# *Some disadvantages RFsp*

- Sensitive to extrapolation = sensitive to data quality (**quality of sampling is ever more important**)
- RF often leads to (hiddden) over-fitting = it requires a reliable CV (spatial declustering)
- Computationally intensive… and the CI will likely to grow in the future

# *Probably #1 problem of RF is extrapolation*

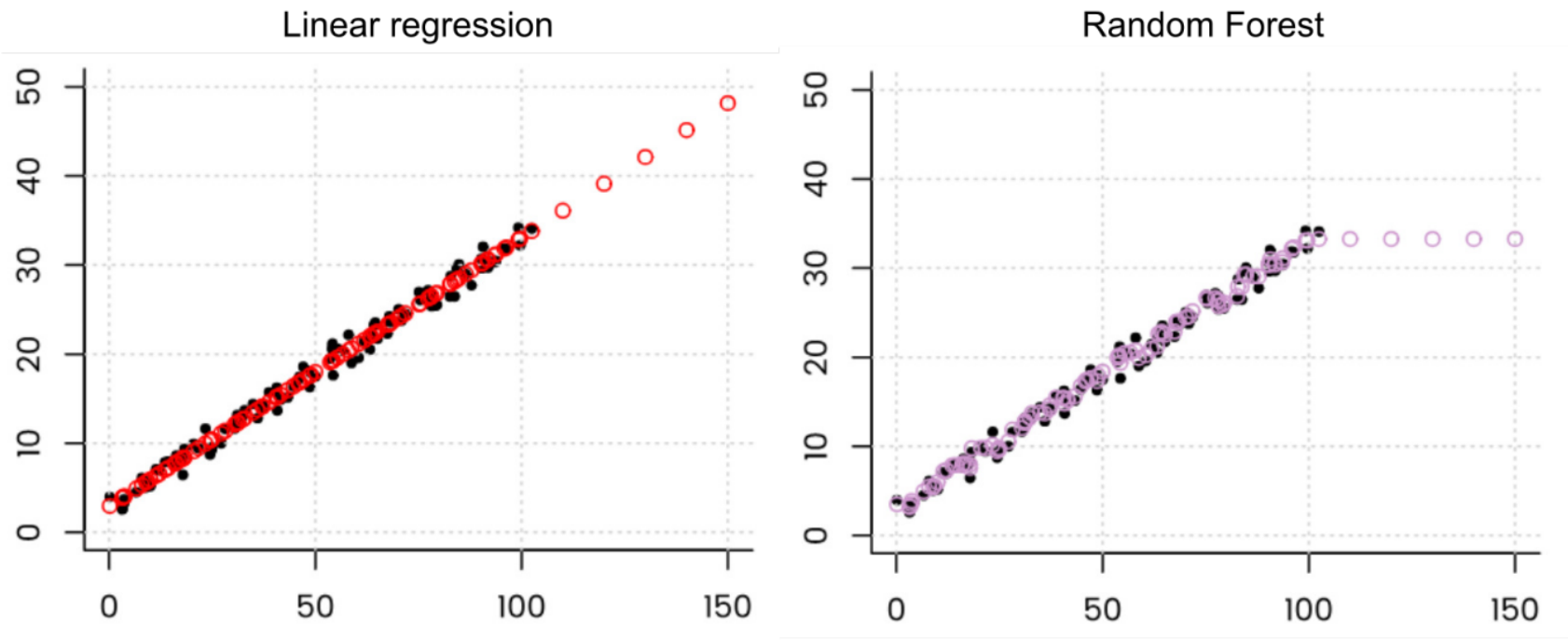
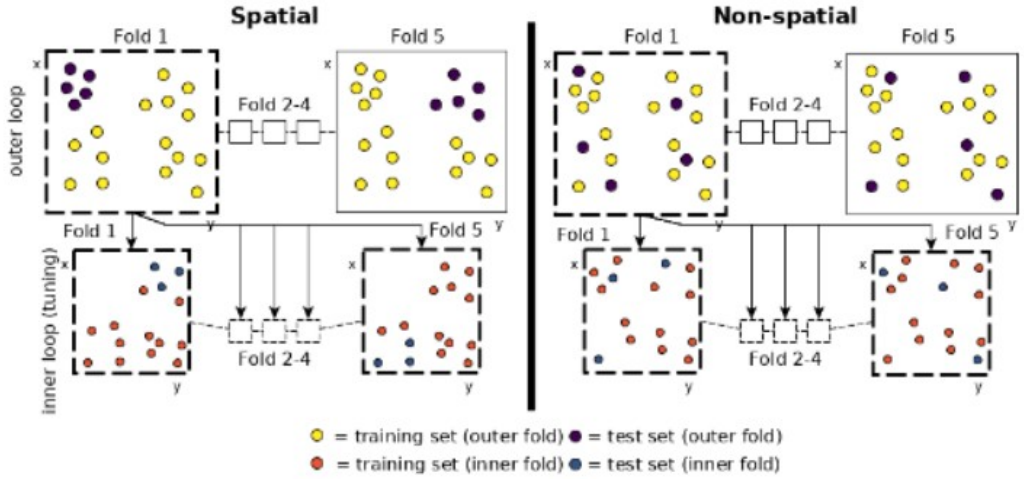



**Figure 14.** Illustration of the extrapolation problem of Random Forest based on the code examples from Peter Ellis (`http://freerangestats.info`). Even though Random Forest is more generic than linear regression and can be used also to fit complex non-linear problems, it can lead to completely nonsensical predictions if applied to extrapolation domains.

assumption is violated in the spatial case due to spatial autocorrelation. Subsequently, non-spatial cross-validation will fail to provide accurate performance estimates.




Nested Spatial and Non-Spatial Cross-Validation

By doing a random sampling of the data set (i.e., non-spatial sampling), training and test set data are often located directly next to each other (in geographical space). Hence, the test set will contain observations which are somewhat similar (due to spatial autocorrelation) to observations in the training set. This leads to the effect that the model, which was trained on the training set, performs quite well on the test data because it already knows it to some degree.

To reduce this bias on the resulting predictive accuracy estimate, Brenning 2005 suggested using spatial partitioning in favor of random partitioning (see Figure 1). Here, spatial clusters are equal to the number of folds chosen. These spatially disjoint subsets of the data introduce a spatial distance between training and test set. This reduces the influence of spatial

# *Looking into the crystal ball...*

- ➔ We might be moving from BIG DATA to BIG MODELS

- We discovered that, with RF and similar tree-based MLA's, whole scientitic fields could be put into a single model (for example, whole 600 million of GBIF records could be used to train a single global biodiversity model). **This model could then be used to generate predictions on-demand, becoming a knowledge engine**.

Center for Brains, Minds & Machines

**CBMM Memo No. 046** **April 1, 2016**

# Building Machines That Learn and Think Like People

by

**Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman**


## Abstract

Recent progress in artificial intelligence (AI) has renewed interest in building systems that learn and think like people. Many advances have come from using deep neural networks trained end-to-end in tasks such as object recognition, video games, and board games, achieving performance that equals or even beats humans in some respects. Despite their biological inspiration and performance achievements, these systems differ from human intelligence in crucial ways. We review progress in cognitive science suggesting that truly human-like learning and thinking machines will have to reach beyond current engineering trends in both what they learn, and how they learn it. Specifically, we argue that these machines should (a) build causal models of the world that support explanation and understanding, rather than merely solving pattern recognition problems; (b) ground learning in intuitive

# *Conclusions*

- The advantage of RFsp over model-based geostatistics is that RFsp requires much less statistical assumptions and is easier to automate (and scale up through parallelization). On the other hand, computational intensity of RFsp can increase exponentially as the number of training points and covariates increases. RF is also sensitive to extrapolation and often leads to (hidden) overfitting. RFsp is still an experimental method and application with large data sets (>>1000 points) is probably not yet recommended.

# Geocomputation with R




Robin Lovelace, Jakub Nowosad, Jannes Muenchow

An indication of the book's progress over time is illustrated below (to be updated roughly every week as the book progresses).