Friedrich-Schiller-Universität Jena

R Package ‘sperrorest’

Parallelized spatial error estimation and permutation-based variable importance assessment for geospatial machine learning

Patrick Schratz, Tobias Herrmann, Alexander Brenning,

GIScience group, University of Jena

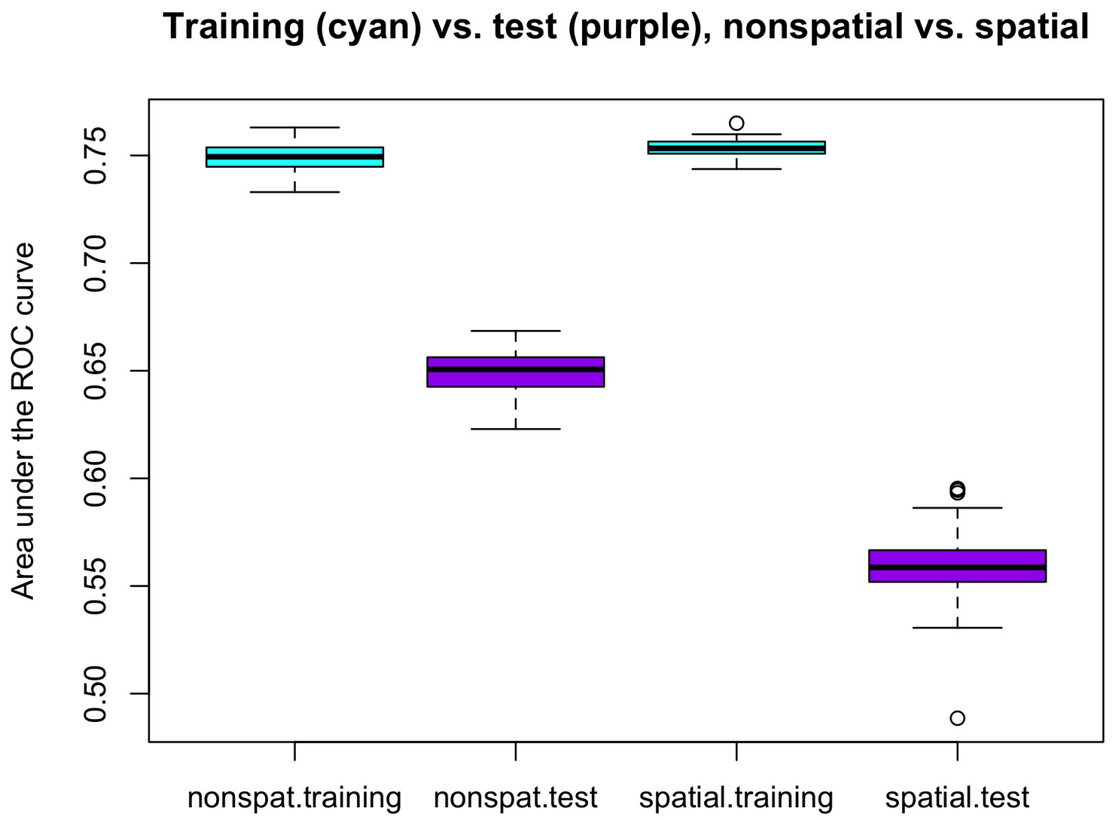
EGU Vienna, April 2017

seit 1558

Spatial cross-validation OR

"Are my model accuracies to good to be true?"*

Spatial vs. non-spatial model performance assessment example (classification tree)



If you are insterested in...

- Which spatial partitioning (there are many!) method to use

If you are insterested in...

- Which spatial partitioning (there are many!) method to use
- Performing (spatial) CV in parallel using `parsperrorest()`

If you are insterested in...

- Which spatial partitioning (there are many!) method to use
- Performing (spatial) CV in parallel using `parsperrorest()`
- Getting more accurate performances of your spatial models

If you are insterested in...

- Which spatial partitioning (there are many!) method to use
- Performing (spatial) CV in parallel using `parsperrorest()`
- Getting more accurate performances of your spatial models
- Getting permutation-based variable importance information from CV

If you are insterested in...

- Which spatial partitioning (there are many!) method to use
- Performing (spatial) CV in parallel using `parsperrorest()`
- Getting more accurate performances of your spatial models
- Getting permutation-based variable importance information from CV

-> Visit my PICO at 'PICOA.2'

Some facts about the package:

- Initial CRAN release: 2012 (v0.2-1)
- Developed by Alexander Brenning[1]
- Purpose: Provide an interface for **spatial** error estimation (cross-validation) and variable importance in R

New v1.0.0 (March 2017)

- Github repository: https://pat-s.github.io/sperrorest/index.html
- Parallelized function `parsperrorest()`
- Full changelog: https://github.com/pat-s/sperrorest/blob/master/NEWS.md

[1] Brenning, A. (2012). Spatial cross-validation and bootstrap for the assessment of prediction rules in Remote Sensing: The R package sperrorest. . doi:10.1109/IGARSS.2012.6352393

Function `parsperrorest()`

Two parallel modes (`par.mode = 1` OR `par.mode = 2`)

Mode 1

- apply-based (package `pbapply`), uses either `mclapply()` (Unix) or `parapply()` (Windows)
- elapsed/remaining time output to console
- faster than 'Mode 2'

Mode 2

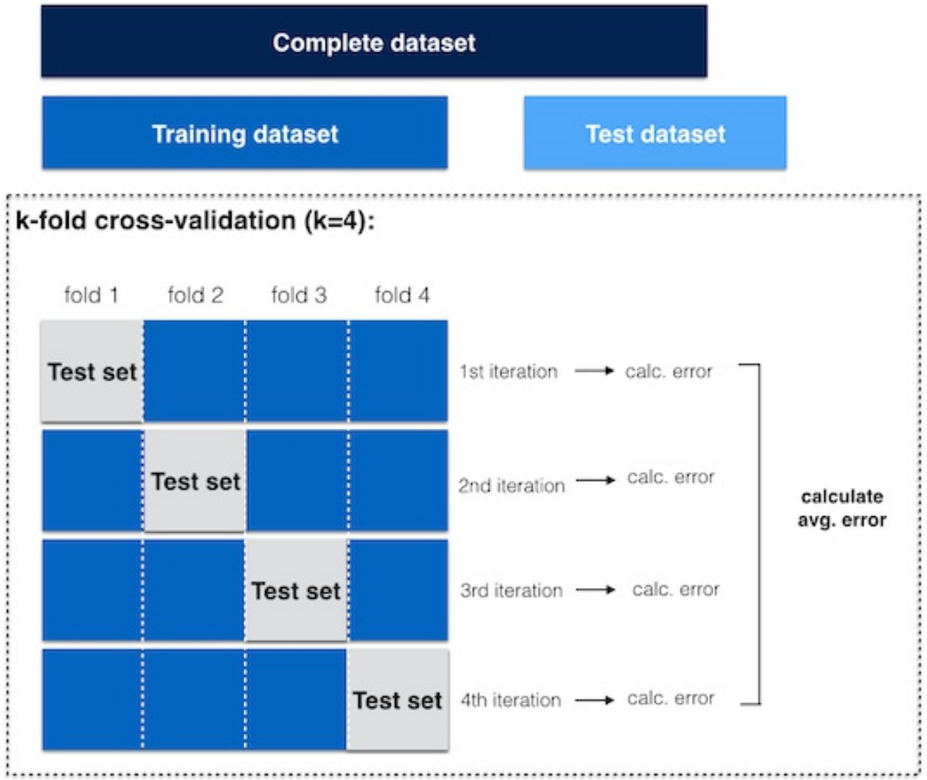
- foreach-based
- provides repetition/fold console output like `sperrorest()` does
- slower than 'Mode 1'
- somewhat more stable

What is the purpose of the package?

- `sperrorest` simplifies/enables (**spatial**) cross-validation (CV) of statistical- and machine learning models
- The package provides a complete framework to
  - set up (create training and test sets) -> `partition.*()`
  - perform (run CV) -> `sperrorest()` and `parsperrorest()`
  - analyze (summarize) cross-validation runs `summary.*()`

Why use cross-validation to assess model performance?

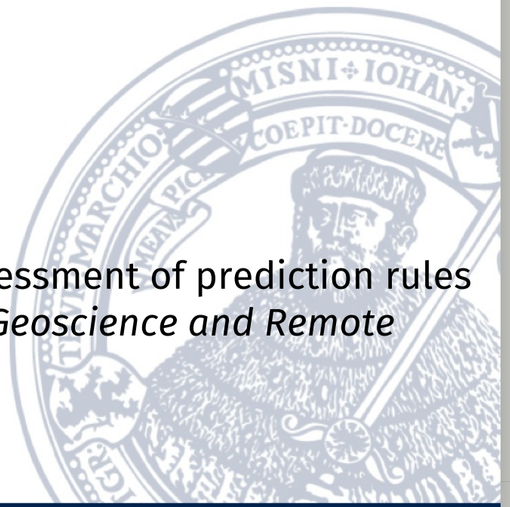
- Independent training and test data
- Every observation is used (at least) once for testing



img source: http://sebastianraschka.com/Articles/2014_intro_supervised_learning.html

Why is SPATIAL cross-validation important?

- When dealing with any kind of spatial data, spatial autocorrelation is present (with a varying magnitude)
- When doing a **non-spatial** cross-validation, usually a *random resampling* is applied. This resampling method assumes that the observations are independent. This is not the case for spatial data!
- The predicted model performances will be overoptimistic if spatial autocorrelation between the training and test data exists.[2]



[2] Brenning, A. (2012). Spatial cross-validation and bootstrap for the assessment of prediction rules in Remote Sensing: The R package sperrorest. *In 2012 IEEE International Geoscience and Remote Sensing Symposium (pp. 5372–5375)*. doi:10.1109/IGARSS.2012.6352393

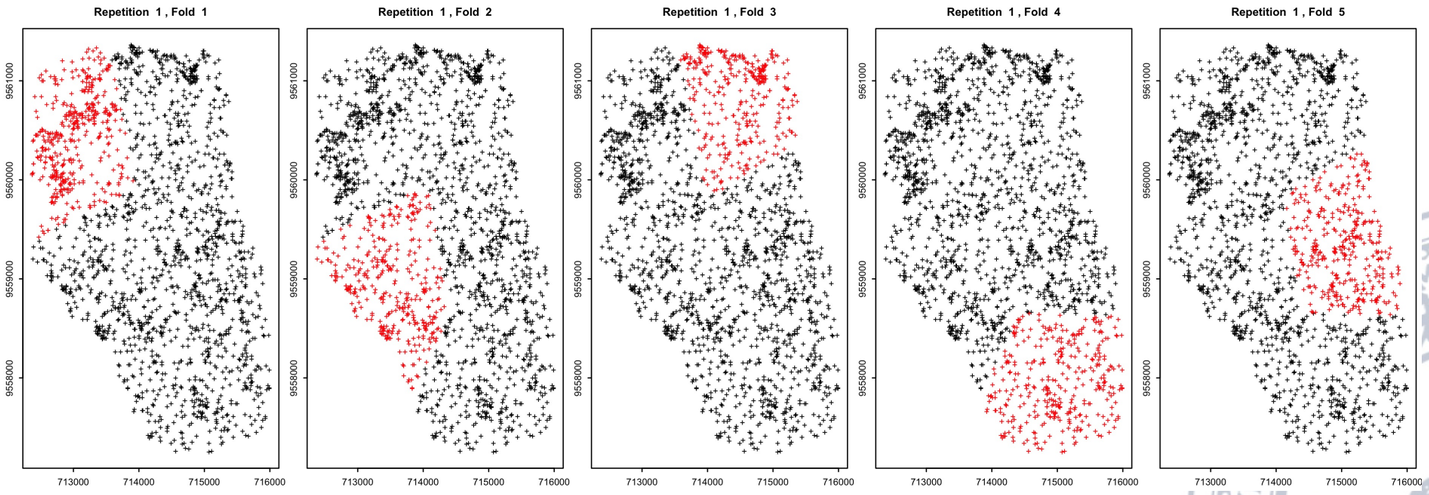
Every model *with spatial data* should be validated using spatial cross-validation to obtain unbiased[3] performance estimates!

[3] bias reduced ;-)

Spatial partitioning methods of sperrorest

`partition.kmeans()`: Versatile approach based on k-means clustering of coordinates

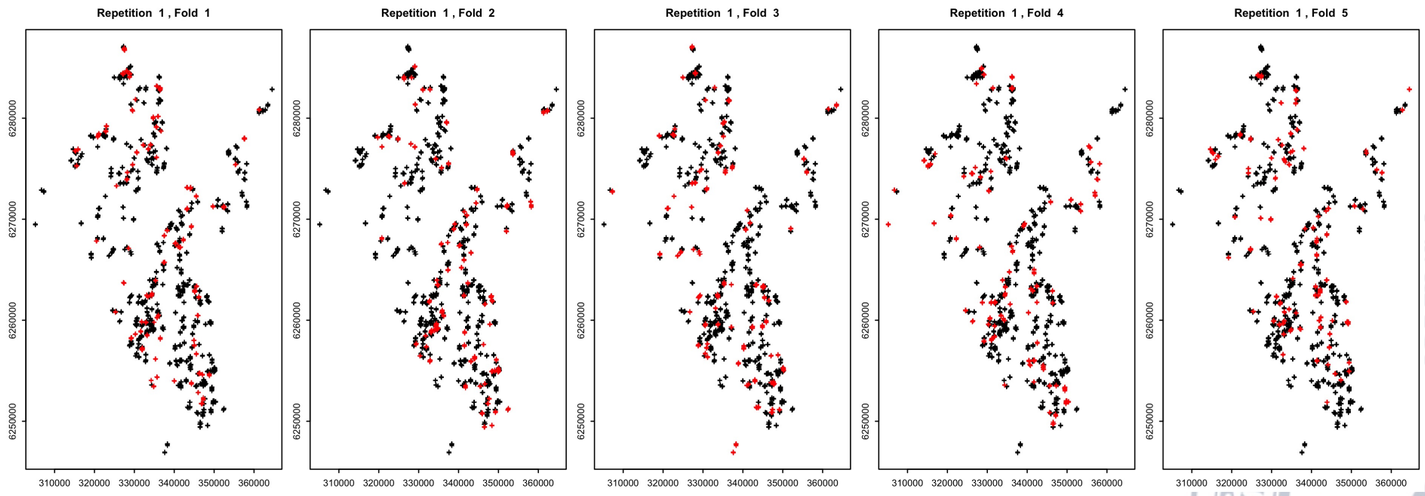
```
data(ecuador)
resamp <- partition.kmeans(ecuador, nfold = 5, repetition = 1:1)
plot(resamp, ecuador)
```



Spatial partitioning methods of sperrorest

`partition.factor.cv()`: Partitioning at grouping level (here: fields)

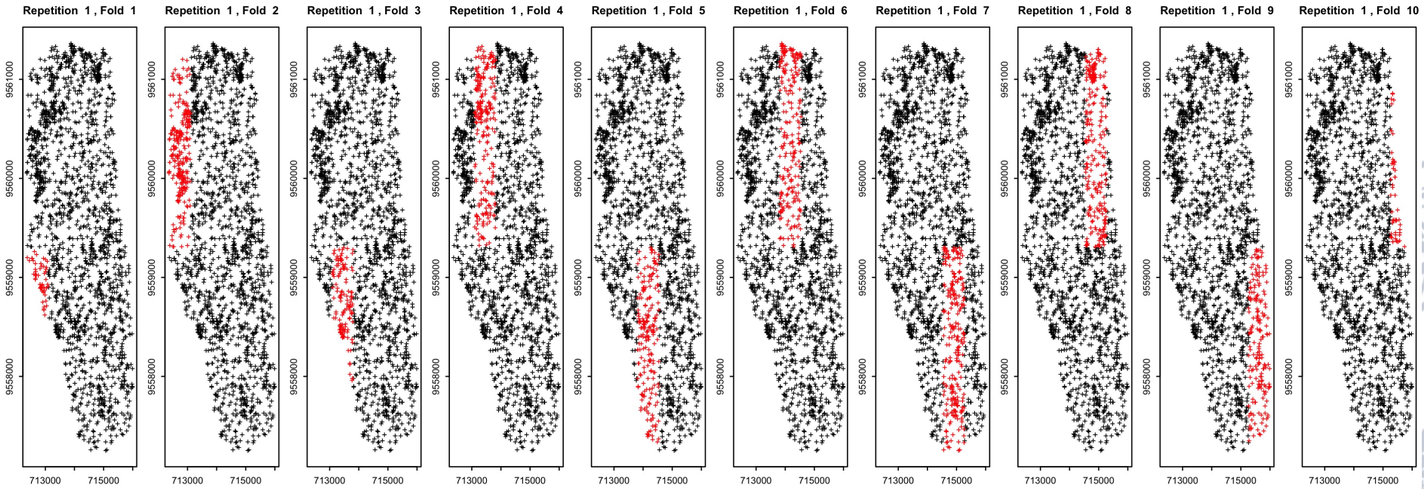
```
data(maipo)
resamp <- partition.factor.cv(maipo, nfold = 5, repetition = 1:1, fac = "field")
plot(resamp, maipo, coords = c("utmx","utmy"))
```



Spatial partitioning methods of sperrorest

`partition.tiles()`: Rectangular tile partitioning (here: 5x2 grid). **Non-equal train/test splits within folds!**

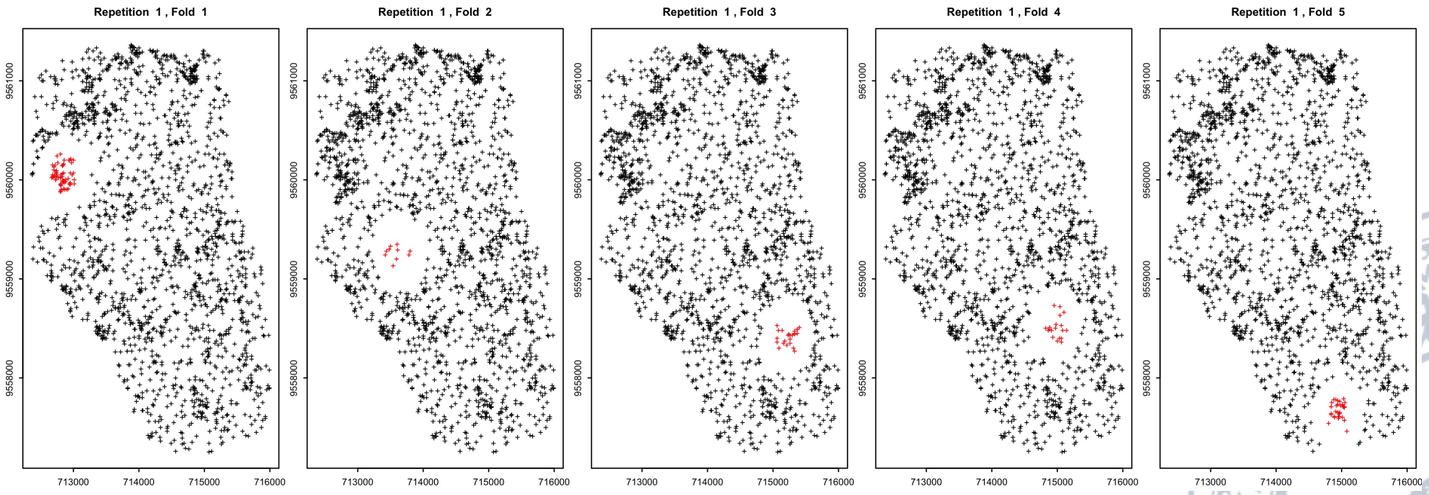
```
data(ecuador)
resamp <- partition.tiles(ecuador, nsplit = c(4,3), reassign = FALSE)
plot(resamp, ecuador)
```



Spatial partitioning methods of sperrorest

`partition.disc()`: Partitioning by circular test areas (with optional buffer)

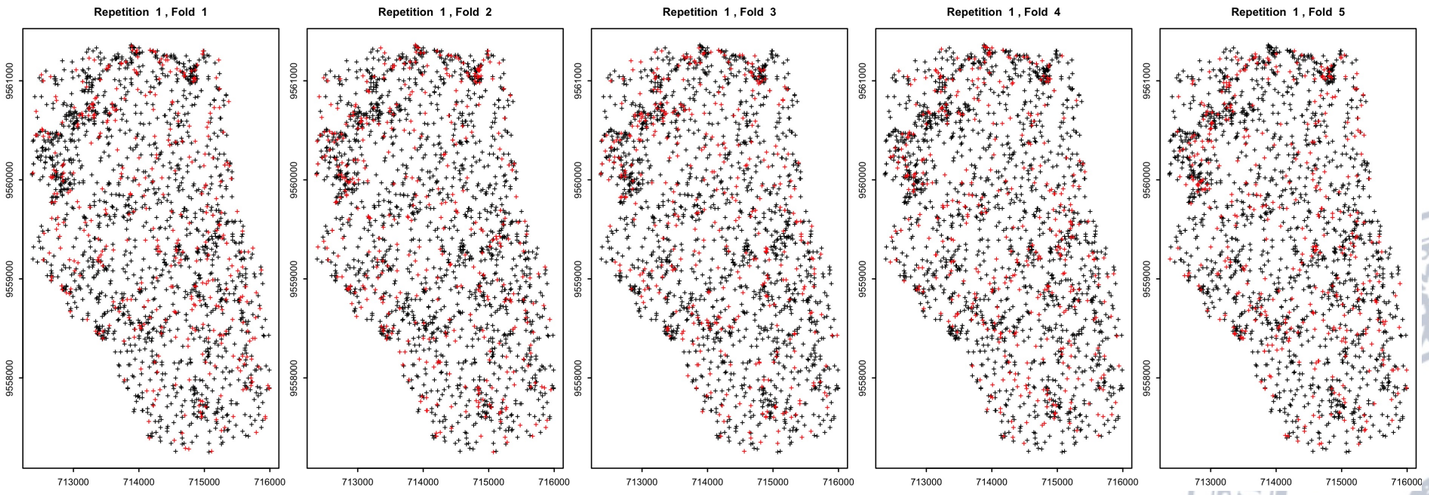
```
data(ecuador)
resamp <- partition.disc(ecuador, radius = 200, buffer = 200, ndisc = 5, repetition = 1:1)
plot(resamp, ecuador)
```



Non-spatial partitioning methods of sperrorest

`partition.cv()`: Random partitioning

```
data(ecuador)
resamp <- partition.cv(ecuador, nfold = 5, repetition = 1:1)
plot(resamp, ecuador)
```



More resampling methods

Non-spatial

- `partition.cv.strat()`: Similar to `partition.cv()`
- `partition.loo()`: Leave-one-out partitioning

Bootstrap based

- `represampling.bootstrap()`
- `represampling.disc.bootstrap()`
- `represampling.factor.bootstrap()`
- `represampling.kmeans.bootstrap()`
- `represampling.tile.bootstrap()`

* or build your own resampling or partitioning function for cross-validation or bootstrap!

Usage example (spatial vs. non-spatial)

First we create a formula `fo`, fit our model `fit` and create a custom predict function `mypred.rpart` which will work with `sperrorest()`. For most models you can use the generic `predict()` function.

In this example we are using classification trees from package `rpart` because they nicely reveal the overfitting of models in a non-spatial setting.

```
data(ecuador) # Muenchow et al. (2012), see ?ecuador
fo <- slides ~ dem + slope + hcurv + vcurv + log.carea + cslope

# Example of a classification tree fitted to this data:
library(rpart)
ctrl <- rpart.control(cp = 0.005) # show the effects of overfitting
fit <- rpart(fo, data = ecuador, control = ctrl)

# custom predict function
mypred.rpart <- function(object, newdata) predict(object, newdata)[, 2]
```

Usage example (spatial vs. non-spatial)

Non-spatial

```
# Non-spatial 100-repeated 10-fold cross-validation:
parsperrorest(data = ecuador, formula = fo,
              model.fun = rpart,
              model.args = list(control = ctrl),
              pred.fun = mypred.rpart,
              smp.fun = partition.cv,
              smp.args = list(repetition = 1:100,
                              nfold = 10),
              importance = TRUE, imp.permutations = 100,
              par.args = list(par.units = 4,
                              par.mode = 1)) -> nspres
```

smp.fun = partition.cv

Spatial

```
# Spatial 100-repeated 10-fold cross-validation:
parsperrorest(data = ecuador, formula = fo,
              model.fun = rpart,
              model.args = list(control = ctrl),
              pred.fun = mypred.rpart,
              smp.fun = partition.kmeans,
              smp.args = list(repetition = 1:100,
                              nfold = 10),
              importance = TRUE, imp.permutations = 100,
              par.args = list(par.units = 4,
                              par.mode = 1)) -> spres
```

smp.fun = partition.kmeans

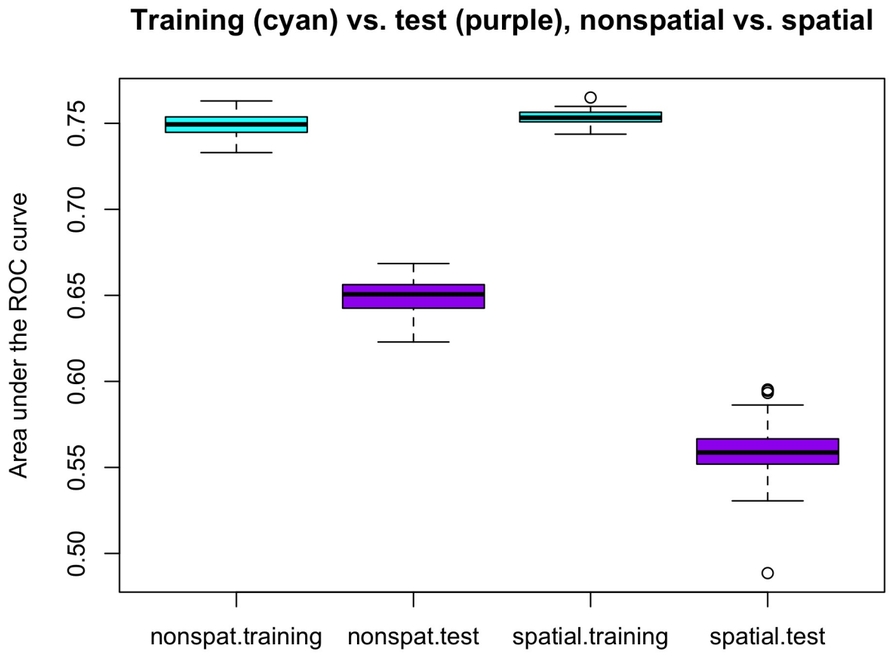
Usage example (spatial vs. non-spatial)

Compare both spatial and non-spatial error distributions of training and test sets.

We will use AUROC as the main error measure in this classification example.

```
smry <- data.frame(
  nonspat.training = unlist(summary(nspres$error.rep, level = 1)$train.auroc),
  nonspat.test     = unlist(summary(nspres$error.rep, level = 1)$test.auroc),
  spatial.training = unlist(summary(spres$error.rep,  level = 1)$train.auroc),
  spatial.test     = unlist(summary(spres$error.rep,  level = 1)$test.auroc))

boxplot(smry, col = c('cyan','purple','cyan','purple'),
        main = 'Training vs. test, nonspatial vs. spatial',
        ylab = 'Area under the ROC curve')
```



- Training-set AUROC much higher than cross-validation AUROC -> indicates overfitting
- **Non-spatial** cross-validation not fully able to detect overfitting -> over-optimistic assessment of spatial prediction performance

Summary methods (Repetition level)

Non-spatial

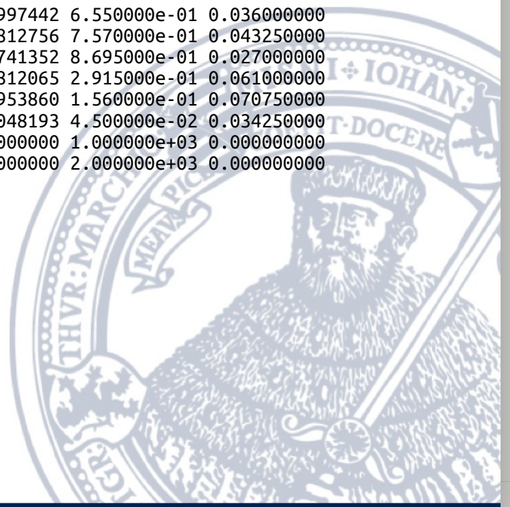
```
summary(nspres$error.rep)
```

```
##                          mean          sd       median         IQR
## train.auroc       7.491639e-01 0.006592398 7.493937e-01 0.008902046
## train.error       2.930411e-01 0.005281050 2.925000e-01 0.007930556
## train.accuracy    7.069589e-01 0.005281050 7.075000e-01 0.007930556
## train.sensitivity 6.844778e-01 0.019618506 6.847778e-01 0.026222222
## train.specificity 7.294400e-01 0.012622770 7.302778e-01 0.014666667
## train.fpr70       3.018522e-01 0.023683083 2.957222e-01 0.025361111
## train.fpr80       4.760267e-01 0.032224847 4.710556e-01 0.046833333
## train.fpr90       6.672656e-01 0.020639622 6.643889e-01 0.027388889
## train.tpr80       5.645089e-01 0.011603589 5.655556e-01 0.015666667
## train.tpr90       3.318267e-01 0.014408557 3.318333e-01 0.017722222
## train.tpr95       1.849667e-01 0.013059969 1.856111e-01 0.018750000
## train.events      9.000000e+03 0.000000000 9.000000e+03 0.000000000
## train.count       1.800000e+04 0.000000000 1.800000e+04 0.000000000
## test.auroc        6.492928e-01 0.009481132 6.506303e-01 0.013554750
## test.error        3.772300e-01 0.008292756 3.770000e-01 0.011500000
## test.accuracy     6.227700e-01 0.008292756 6.230000e-01 0.011500000
## test.sensitivity  6.012200e-01 0.018478806 6.030000e-01 0.023500000
## test.specificity  6.443200e-01 0.020083100 6.460000e-01 0.027500000
## test.fpr70        5.069800e-01 0.028486389 5.055000e-01 0.040500000
## test.fpr80        6.447900e-01 0.026780173 6.415000e-01 0.036500000
## test.fpr90        8.041400e-01 0.021611828 8.045000e-01 0.032000000
## test.tpr80        4.079800e-01 0.020300261 4.120000e-01 0.019500000
## test.tpr90        1.959900e-01 0.022569979 1.965000e-01 0.034250000
## test.tpr95        9.348000e-02 0.017725448 9.200000e-02 0.025500000
## test.events       1.000000e+03 0.000000000 1.000000e+03 0.000000000
## test.count        2.000000e+03 0.000000000 2.000000e+03 0.000000000
```

Spatial

```
summary(spres$error.rep)
```

```
##                          mean          sd       median         IQR
## train.auroc       7.533051e-01 0.003621755 7.532951e-01 0.005597159
## train.error       2.913928e-01 0.003389006 2.918056e-01 0.004444444
## train.accuracy    7.086072e-01 0.003389006 7.081944e-01 0.004444444
## train.sensitivity 6.954611e-01 0.012005458 6.920000e-01 0.016027778
## train.specificity 7.217533e-01 0.008780407 7.198889e-01 0.008000000
## train.fpr70       2.908456e-01 0.010933883 2.890000e-01 0.011111111
## train.fpr80       4.566378e-01 0.016625761 4.560000e-01 0.020444444
## train.fpr90       6.542111e-01 0.014874681 6.537778e-01 0.019472222
## train.tpr80       5.572822e-01 0.008801583 5.576667e-01 0.012250000
## train.tpr90       3.491878e-01 0.009069540 3.496667e-01 0.005083333
## train.tpr95       1.487700e-01 0.035201567 1.616667e-01 0.071583333
## train.events      9.000000e+03 0.000000000 9.000000e+03 0.000000000
## train.count       1.800000e+04 0.000000000 1.800000e+04 0.000000000
## test.auroc        5.599123e-01 0.014447499 5.586595e-01 0.014721500
## test.error        4.498300e-01 0.020412071 4.460000e-01 0.038250000
## test.accuracy     5.501700e-01 0.020412071 5.540000e-01 0.038250000
## test.sensitivity  4.770300e-01 0.028685496 4.830000e-01 0.032250000
## test.specificity  6.233100e-01 0.039167781 6.280000e-01 0.064250000
## test.fpr70        6.536100e-01 0.031997442 6.550000e-01 0.036000000
## test.fpr80        7.633100e-01 0.027812756 7.570000e-01 0.043250000
## test.fpr90        8.722900e-01 0.018741352 8.695000e-01 0.027000000
## test.tpr80        2.760000e-01 0.044812065 2.915000e-01 0.061000000
## test.tpr90        1.380700e-01 0.041953860 1.560000e-01 0.070750000
## test.tpr95        4.272000e-02 0.022048193 4.500000e-02 0.034250000
## test.events       1.000000e+03 0.000000000 1.000000e+03 0.000000000
## test.count        2.000000e+03 0.000000000 2.000000e+03 0.000000000
```



Permutation-based Variable Importance[4]

Non-spatial

```
summary(nspres$importance)[1:10]
```

```
##            mean.auroc   mean.error mean.accuracy mean.sensitivity
## dem       0.125296481 -0.096895750   0.096895750       0.12446769
## slope     0.005298123 -0.001661750   0.001661750       0.00413101
## hcurv     0.031417110 -0.024046375   0.024046375       0.05797876
## vcurv     0.009737792 -0.006681975   0.006681975       0.01568034
## log.carea 0.028480043 -0.027956225   0.027956225       0.03468735
## cslope    0.009218880 -0.005630575   0.005630575       0.02681362
##           mean.specificity  mean.fpr70  mean.fpr80  mean.fpr90
## dem            0.069643940 -0.13677964 -0.10271008 -0.07407327
## slope         -0.000770077 -0.01097616 -0.01847902 -0.01759824
## hcurv         -0.009793958 -0.05104078 -0.03595676 -0.03249064
## vcurv         -0.002269137 -0.01718063 -0.01947689 -0.01047334
## log.carea      0.021456550 -0.03457664 -0.02166920 -0.01094175
## cslope        -0.015654495 -0.01303756 -0.01093145 -0.01372432
##             mean.tpr80    mean.tpr90
## dem        0.219457801  0.0451892196
## slope      0.001129014 -0.0004273368
## hcurv      0.022841537  0.0065212811
## vcurv      0.007311441  0.0083051513
## log.carea  0.036446092  0.0142388108
## cslope    -0.002588373  0.0020895990
```

Spatial

```
summary(spres$importance)[1:10]
```

```
##            mean.auroc    mean.error mean.accuracy mean.sensitivity
## dem        0.027041166 -0.0241957993  0.0241957993      0.020557928
## slope     -0.003074978  0.0045348759 -0.0045348759     -0.006784491
## hcurv      0.004402262 -0.0014313317  0.0014313317      0.022176014
## vcurv     -0.001812538  0.0020414829 -0.0020414829      0.004826931
## log.carea  0.005765719  0.0009237096 -0.0009237096      0.008299803
## cslope     0.002854320  0.0004552682 -0.0004552682      0.008781445
##           mean.specificity   mean.fpr70   mean.fpr80   mean.fpr90
## dem            0.032459193 -0.051136819 -0.034294393 -0.031403251
## slope         -0.002016565  0.006498740  0.006521210 -0.001539555
## hcurv         -0.016737312 -0.009434742 -0.002674188 -0.003567928
## vcurv         -0.006819133 -0.005230117 -0.001395638  0.002148895
## log.carea     -0.002199538 -0.020309695  0.006230078 -0.009065979
## cslope        -0.008048156 -0.006911105  0.000521190 -0.003841377
##             mean.tpr80    mean.tpr90
## dem        0.0084642081 -0.0058246734
## slope     -0.0006432569 -0.0048475963
## hcurv      0.0005174410 -0.0036273013
## vcurv     -0.0021882795 -0.0009014397
## log.carea -0.0107712505 -0.0095970755
## cslope    -0.0105363532 -0.0060414542
```

[4] Russ, Georg, and Alexander Brenning. 2010b. “Spatial Variable Importance Assessment for Yield Prediction in Precision Agriculture.” *In Lecture Notes in Computer Science, 184–95*. Springer Science + Business Media. doi:10.1007/978-3-642-13062-5_18.

Miscellaneous

Other summaries:

- Fold-level error -> `summary(object$error.fold)`
- Resampling -> `summary(object$represampling)`
- Benchmarks -> `summary(object$benchmarks)`
- Package Info -> `summary(object$package.version)`

Workhorse functions:

- `sperrorest()` (sequential)
- `parsperrorest()` (parallelized)

Package vignette:

- https://pat-s.github.io/sperrorest/articles/sperrorest-vignette.html

Material & Contact

Web: https://pat-s.github.io

Twitter: @pjs_228

PDF-Slides: https://speakerdeck.com/pat_s/sperrorest-egu2017-presentation

GIScience group, University of Jena