Classification in R: Validation

By now, you have generated your classification map, great! But can you rely on the map's information? To underpin the meaningfulness of your results, a validation is needed.
The validation of remote sensing data is the last step in our workflow. The purpose of this chapter is to describe standard and advanced methods for validating a classification map.

Validation Intro

Training dataset: A model is initially fit on a training sample dataset. The model iteratively learns from those training samples and tries to map data *x* to output response *y*.

Testing dataset: During training, algorithms often use a testing dataset for an unbiased evaluation of a model fit while tuning the model's hyperparameter, e.g., *mtry* for RF, or *γ* and *C* for SVM. The testing dataset is generated internally, e.g., in the form of OOB samples in RF, or cross validation in SVM.

Validation dataset: Finally, a validation dataset is completely independent from the other two datasets and provides an unbiased evaluation of a model fit.

Alright, so what is the best validation practice for remote sensing studies?

1. automatically create multiple point coordinates all over your study area or your classification extent,
2. manually attribute the corresponding class labels to all of those point coordinates (labeling),
3. statistically examine the deviations and matches between the manually assigned class labels and the labels assigned by the classificator at any given point coordinates

In the following, we want to present a best practice workflow for a classification in detail.

Create Samples in R

Creating validation samples is a crucial step in validation workflows. Certainly, sampling is not a trivial task. Many publications deal with the optimal sampling method, trying to maintain heterogeneity and avoid autcorrelations within validation samples, e.g., KÖHL et al. 2006, MU et al. 2015, OLOFFSON et al. 2014.

In general, there are several random sampling strategies commonly found in remote sensing studies and software solutions:

- **random**: Validation samples were picked completely random. Each pixel within the study area has the same probability of being picked.
- **stratified random**: The proportions of the classes in the validation samples correspond to the area proportion on the classification map. That is, the larger the area of class on the map, the more samples will be drawn from it.
- **equalized stratified random**: Ensures, that each class's sample size is exactly the same regardless of area of class on the map.

We will use **equalized stratified random** sampling for this example.

All we need is the raster package, so make sure you've imported it.

```
#install.packages("raster")
library(raster)
```

Furthermore, you should have your classification map ready, lying in your working directory. Import it with the `raster()` function:

```
img.classified <- raster("RF_classification.tif")
```

The raster provides a function called `sampleStratified()`, which does all the work for us:

```
smp.test <- sampleStratified(x = img.classified,
                             size = 50,
                             na.rm = TRUE,
                             sp = TRUE)
```

This function needs a raster-object as `x` argument and a positive integer value as `size` argument. Latter is the number of sample points per class. Additionally, we can exclude all NA values, by setting `na.rm = TRUE`. NA Values can arise if your scene lies diagonally in space and points are placed in the border areas. Line 4 ensures, that the returned object is a SpatialPointDataFrame (which is easier to handle).

We can now check the class labels of our newly extracted validation points:

```
smp.test$RF_classification
```

```
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2
 [57] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3
[113] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
[169] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
[225] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
```

Note: the name of the column within `smp.test` depends on the file name of your classification map.

Due to the subsequent manual labeling in the next section, it makes sense to mix the samples so that the order is random. Again, we use the `sample()` function for this:

```
smp.test <- smp.test[sample(nrow(smp.test)), ]
```

By looking at the class labels again, we see that the order is now random:

```
  [1] 1 5 4 4 5 2 2 4 5 5 4 4 1 3 5 2 4 2 5 4 4 2 5 3 5 3 4 2 1 1 5 3 5 5 3
2 1 5 4 3 3 4 3 2 5 4 1 2 4 1 1 2 2 5 2 3
 [57] 3 2 5 3 2 4 2 4 1 1 4 2 2 3 2 1 5 1 2 4 3 2 4 5 2 1 3 5 1 3 3 5 4 4 5
4 1 1 1 1 2 2 1 3 3 3 1 3 5 4 5 2 1 2 3 4
```

```
[113] 1 3 4 5 5 3 3 4 3 4 4 2 2 2 3 3 1 1 5 3 1 1 3 5 4 4 5 1 4 5 2 1 4 1 4
5 1 2 4 4 4 3 2 3 5 3 2 3 5 3 1 5 4 5 1 2
[169] 1 4 4 4 4 2 5 2 2 2 3 3 3 1 2 5 5 1 5 2 2 2 5 1 4 5 2 1 3 4 3 1 1 5 4
4 1 5 4 5 2 5 1 1 4 4 5 2 1 3 3 3 1 5 3 2
[225] 4 2 1 5 2 3 3 4 1 1 2 3 2 5 5 1 4 3 1 3 3 3 1 5 5 2
```

In addition, we can delete all variables in our dataframe `smp.test` and append a consecutive `ID` variable called `ID`, which will then be displayed to us in QGIS:

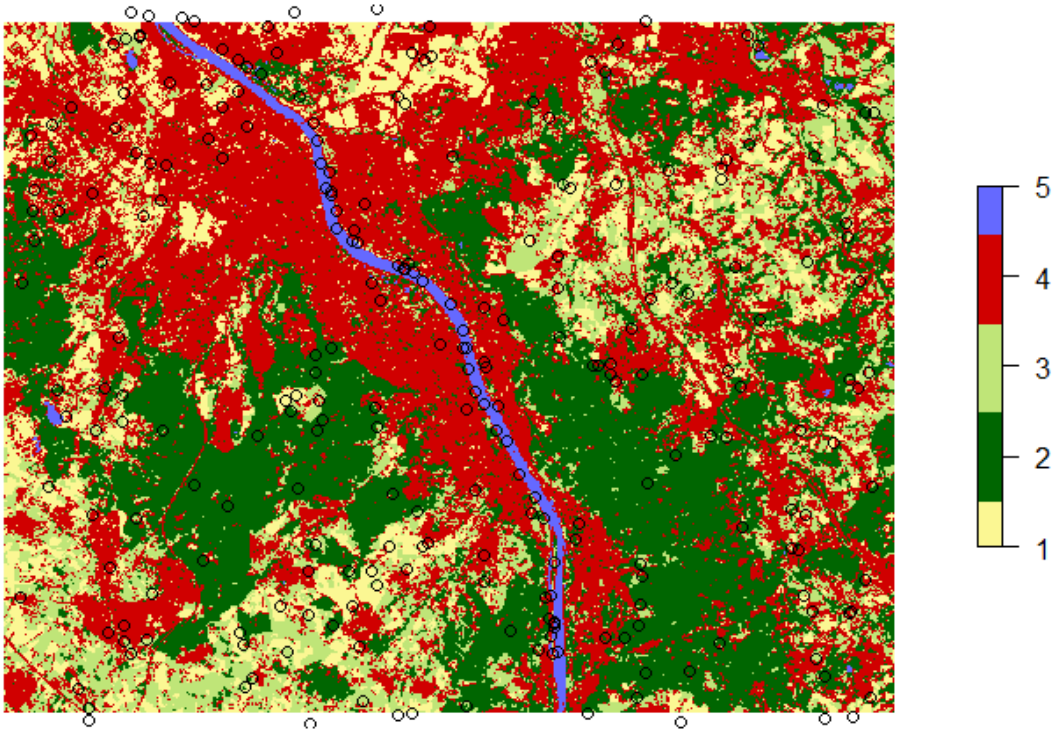
```
smp.test <- smp.test[, -c(1, 2)]
smp.test$ID <- 1:nrow(smp.test)
```

```
smp.test
class       : SpatialPointsDataFrame
features    : 250
extent      : 357210, 383400, 5607060, 5627340  (xmin, xmax, ymin, ymax)
crs         : +proj=utm +zone=32 +datum=WGS84 +units=m +no_defs
variables   : 1
names       :  ID
min values  :   1
max values  : 250
```

As you can see, there is 1 variable (`ID`) left that consecutively represents our samplings.

To visualize the distribution of our validation points, we can plot the SpatialPointDataFrame `smp.test` on top of our classification map in one plot:

```
plot(img.classified,
     axes = FALSE,
     box = FALSE,
     col = mycolors)
points(smp.test)
```



Last but not least, it is still necessary to save the SpatialPointDataFrame `smp.test` as a shapefile to your hard drive. This is also very easy with the `shapefile()` function from the raster package. Choose an appropriate filename for the new shapefile created:

```
shapefile(smp.test,
          filename = "RF_validation.shp",
          overwrite = TRUE
)
```

Questions / prove your knowledge:

- What is the purpose of validation in the land use classification workflow?
- Why is it necessary to create an independent validation dataset?
- Briefly discern random, stratified and equalized stratified random strategies!