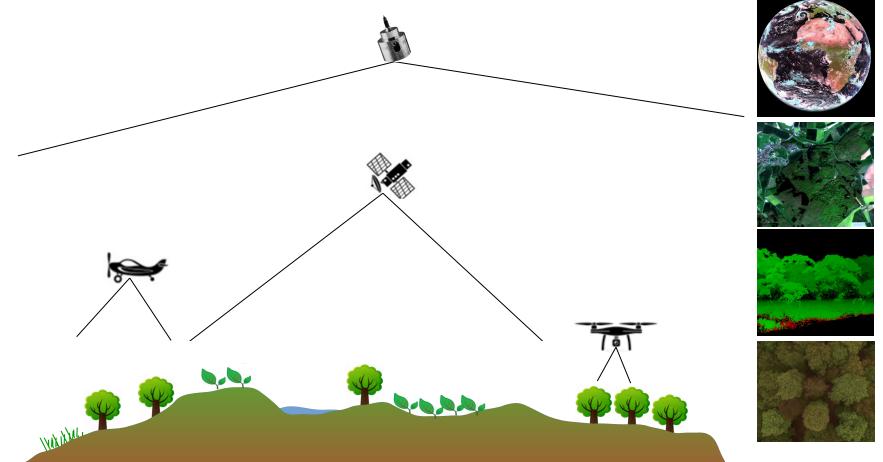


# Machine learning for remote sensing applications

Practice: Land cover classification

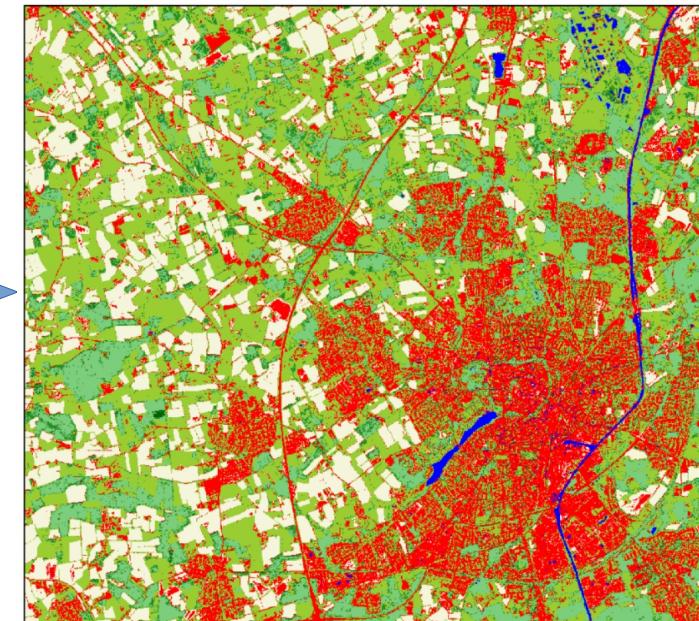
**Hanna Meyer**

Institute for Geoinformatics, WWU Münster

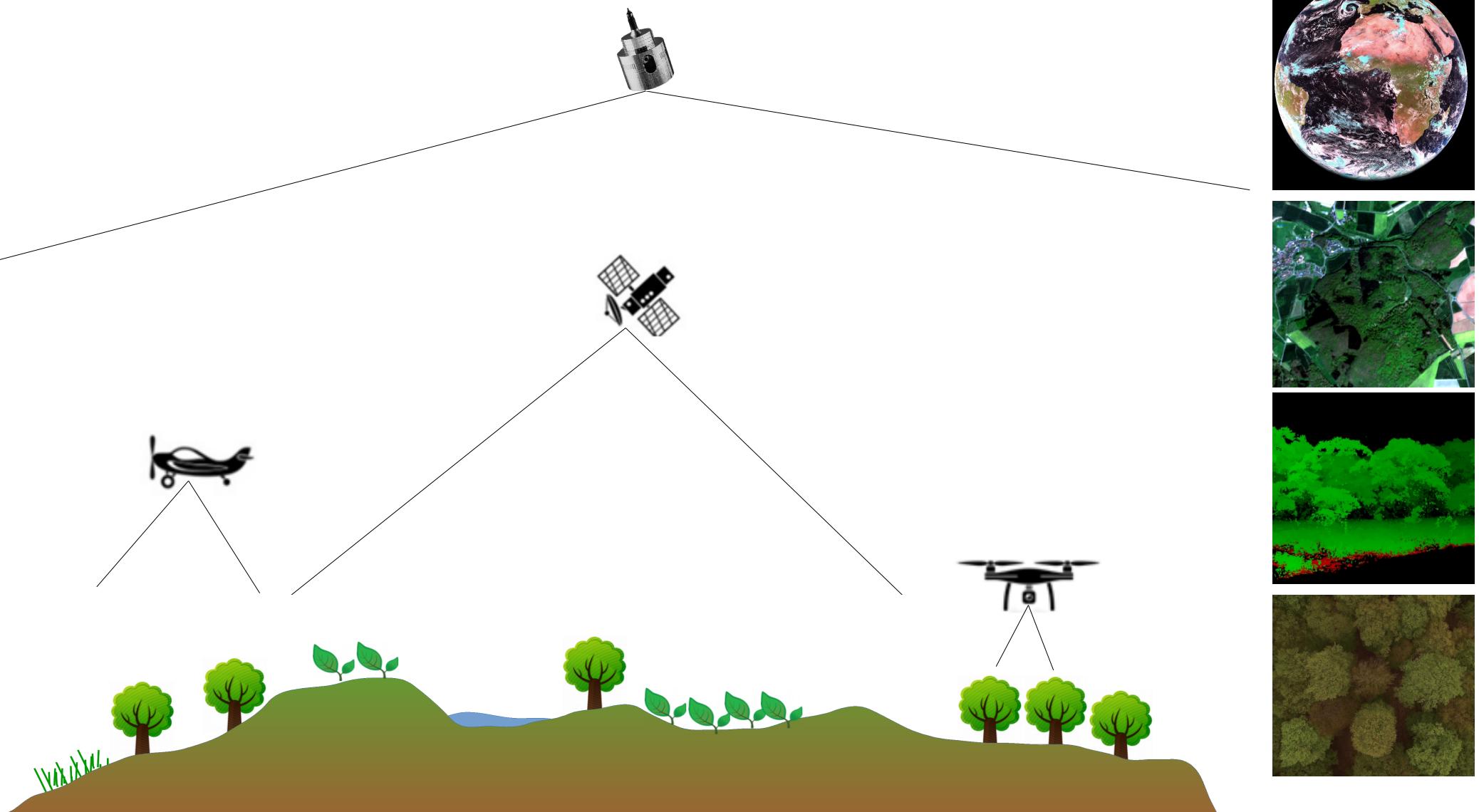


→ Slides and material: [https://github.com/HannaMeyer/OpenGeoHub\\_2019](https://github.com/HannaMeyer/OpenGeoHub_2019)

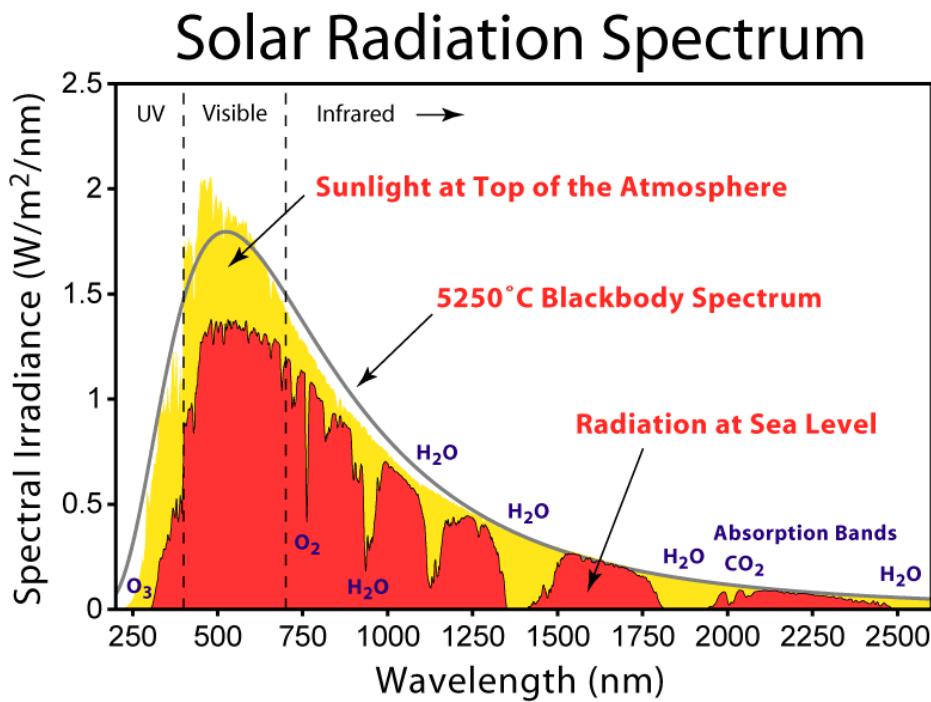
# Problem: Moving from field observations to maps of environmental variables



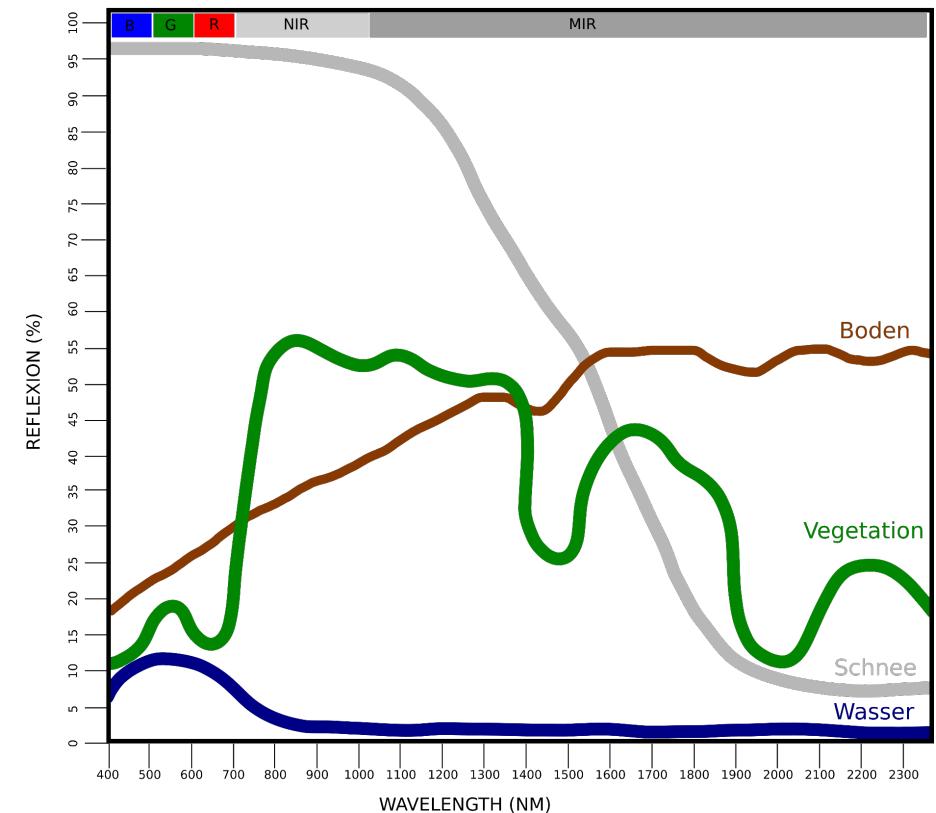
# Remote sensing data



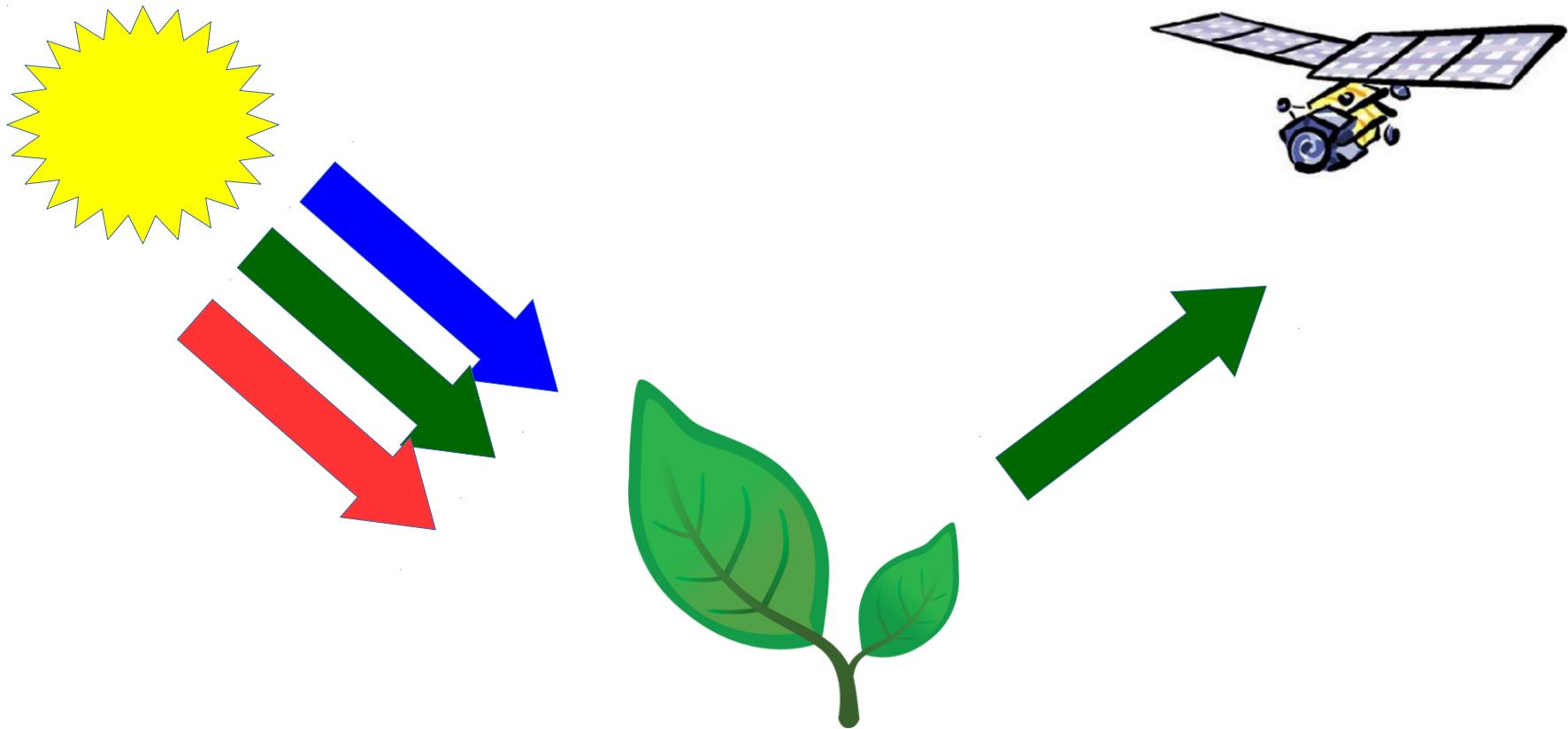
# Solar radiation and reflection properties



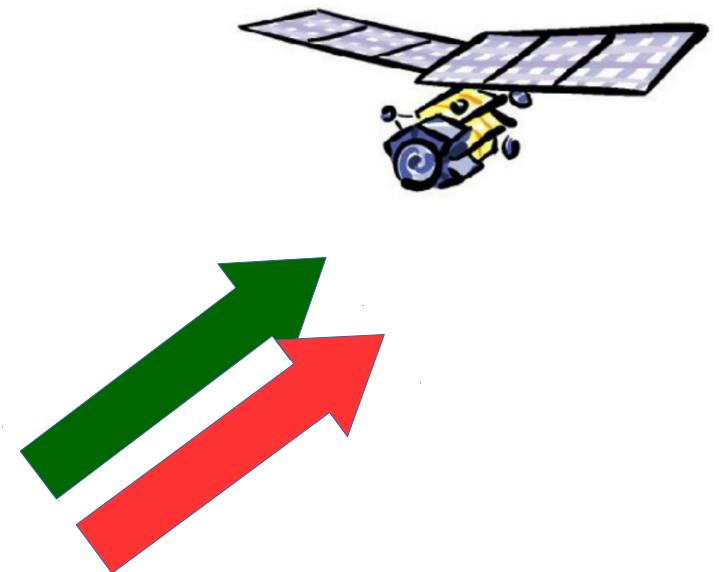
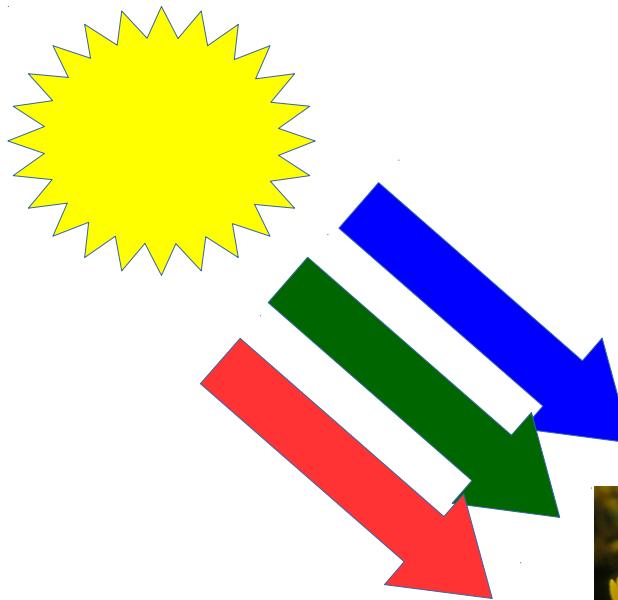
[https://commons.wikimedia.org/wiki/File:Solar\\_Spectrum.png](https://commons.wikimedia.org/wiki/File:Solar_Spectrum.png)



# Reflection properties



# Reflection properties





How can we make use of the different reflectance properties to map land cover?



# So let's get some satellite data

Platform/Sensor	Spatial resolution (m)	Temporal resolution	Availability
Landsat MSS	79	16 days	since 1972
Landsat TM	30	16 days	since 1982
Landsat ETM+	30	16 days	since 1999
Landsat 8 (OLI)	30	16 days	since 2013
Sentinel-2	10	5/10 days	since 2014
MODIS Terra/Aqua	250-1000	4 per day	since 2000
Meteosat Second Generation	3000	15 minutes	since 2002

# Sentinel-2 data

Spectral bands for the Sentinel-2 sensors<sup>[10]</sup>

Sentinel-2 bands	Sentinel-2A		Sentinel-2B		Spatial resolution (m)
	Central wavelength (nm)	Bandwidth (nm)	Central wavelength (nm)	Bandwidth (nm)	
Band 1 - Coastal aerosol	442.7	21	442.2	21	60
Band 2 - Blue	492.4	66	492.1	66	10
Band 3 - Green	559.8	36	559.0	36	10
Band 4 - Red	664.6	31	664.9	31	10
Band 5 - Vegetation red edge	704.1	15	703.8	16	20
Band 6 - Vegetation red edge	740.5	15	739.1	15	20
Band 7 - Vegetation red edge	782.8	20	779.7	20	20
Band 8 - NIR	832.8	106	832.9	106	10
Band 8A - Narrow NIR	864.7	21	864.0	22	20
Band 9 - Water vapour	945.1	20	943.2	21	60
Band 10 - SWIR - Cirrus	1373.5	31	1376.9	30	60
Band 11 - SWIR	1613.7	91	1610.4	94	20
Band 12 - SWIR	2202.4	175	2185.7	185	20

# Getting satellite data

USGS science for a changing world

EarthExplorer - Home Page Expires In 1:57:21

Home 1 New System Message Save Criteria Load Favorite Manage Criteria Item Basket (0) HannaM RSS Feedback Help

Search Criteria Data Sets Additional Criteria Results

4. Search Results

If you selected more than one data set to search, use the dropdown to see the search results for each specific data set.

Show Result Controls

Data Set Click here to export your results

Sentinel-2

54 ID:L1C\_T32UMC\_A016307\_20180806T104340 Acquisition Date:2018/08/06 Platform:SENTINEL-2B Tile Number:T32UMC

55 ID:L1C\_T32UMC\_A016307\_20180806T104340 Acquisition Date:2018/08/06 Platform:SENTINEL-2A Tile Number:T32UMC

56 ID:L1C\_T32ULC\_A016307\_20180806T104340 Acquisition Date:2018/08/06 Platform:SENTINEL-2A Tile Number:T32ULC

57 ID:L1C\_T32ULC\_A007370\_20180804T105022 Acquisition Date:2018/08/04 Platform:SENTINEL-2B Tile Number:T32ULC

ID:L1C\_T32ULC\_A016264\_20180803T103239 Acquisition Date:2018/08/03 Platform:SENTINEL-2A

View Item Basket Submit Standing Request

Map Satellite

Search Criteria Summary (Show)

(51° 51' 32" N, 009° 41' 39" E) Options Overlays

Clear Criteria

The up-to-date Google map is not for purchase or for download; it is to be used as a guide for reference and search purposes only.

Map data ©2019 GeoBasis-DE/BKG (©2009), Google Imagery ©2019 TerraMetrics | 10 Km Terms of Use

Detailed description: This screenshot shows the USGS EarthExplorer search results page. On the left, a sidebar displays five search results for Sentinel-2 and Sentinel-2A data from August 2018. Each result includes a thumbnail, ID, acquisition date, platform, tile number, and a set of download and sharing icons. To the right is a large satellite map of a rural area in Europe, likely the Netherlands and Germany, with a red marker indicating a specific location. The map includes place names like Harderwijk, Ede, Arnhem, Nijmegen, and Wuppertal. A legend at the top right of the map shows coordinates (51° 51' 32" N, 009° 41' 39" E) and provides options for 'Options' and 'Overlays'. The bottom of the map has copyright information for Google and GeoBasis-DE. At the very bottom of the page, there are links for DOI Privacy Policy, Legal, Accessibility, Site Map, and Contact USGS.

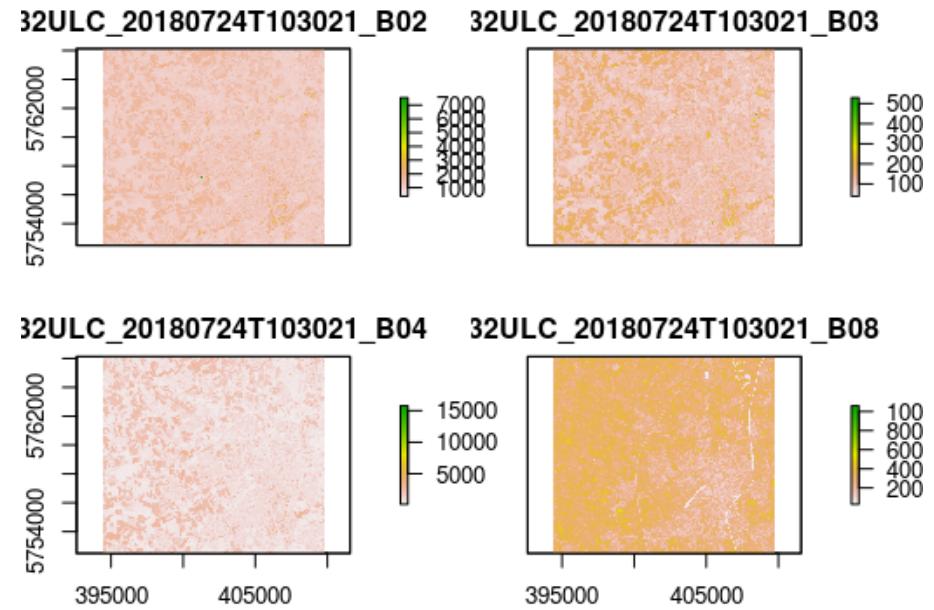
→ <https://earthexplorer.usgs.gov/>

# Load satellite data in R

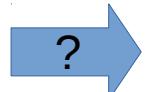
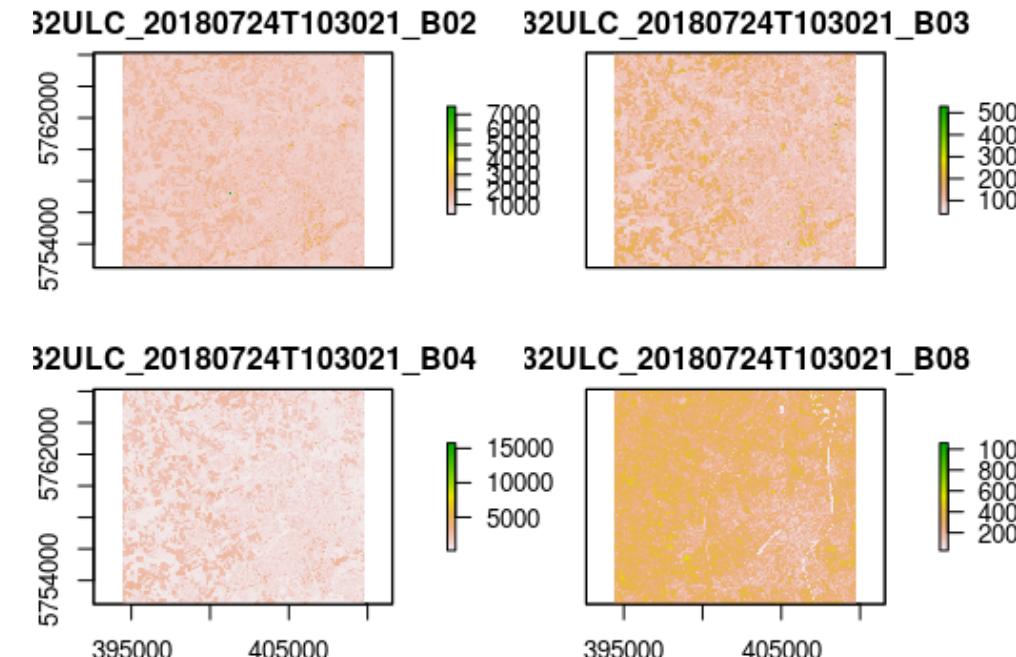
- Options: raster, stars, gdalcubes
- Focus for this session on raster

## How to import satellite data into R

```
library(raster)
sen <- stack("blue.tif","green.tif",
            "red.tif","NIR.tif")
plot(sen)
```



# How do we get the “color” in the satellite data?

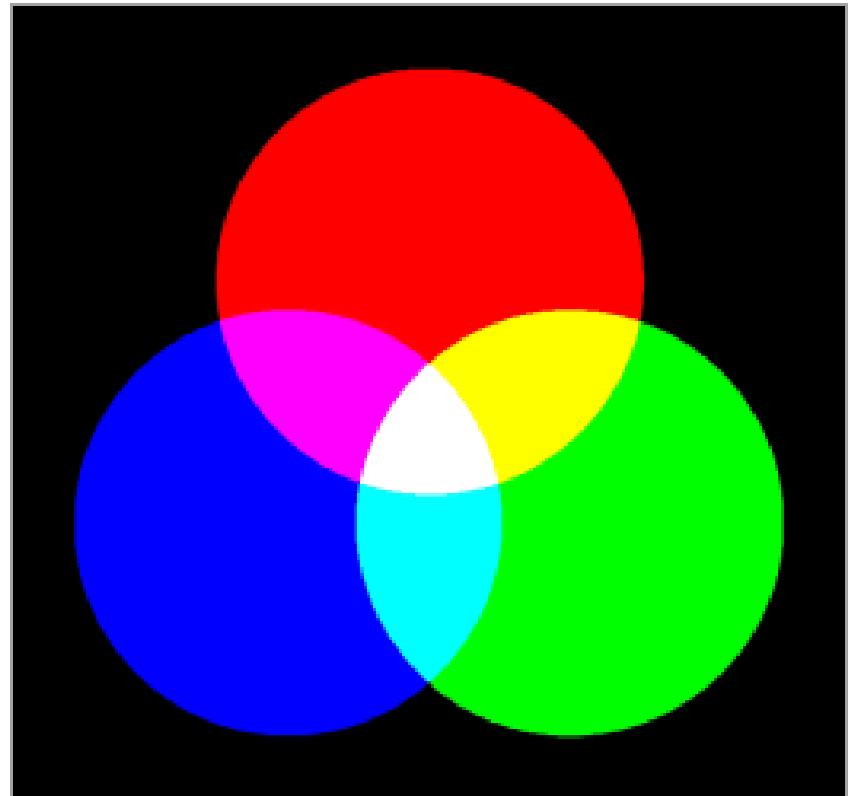


# How do we get the “color” in the satellite data?

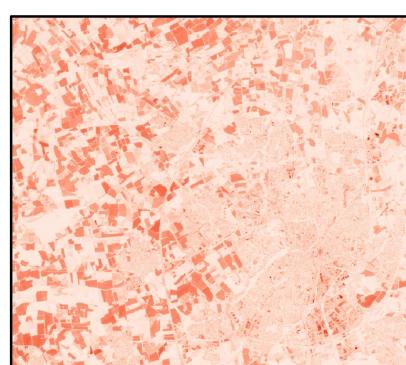
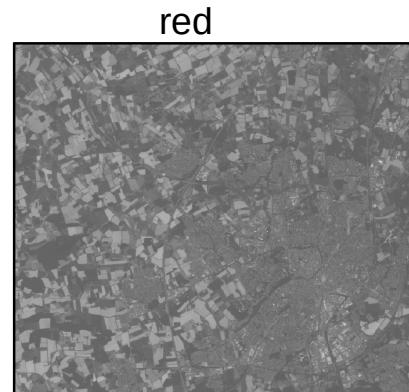
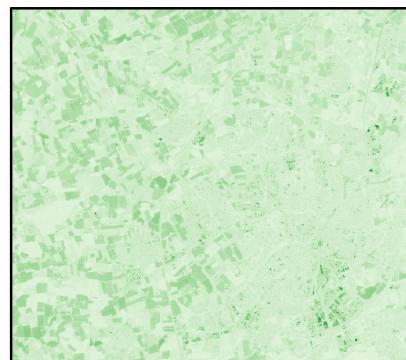
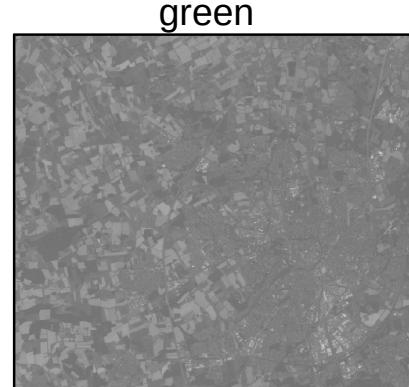
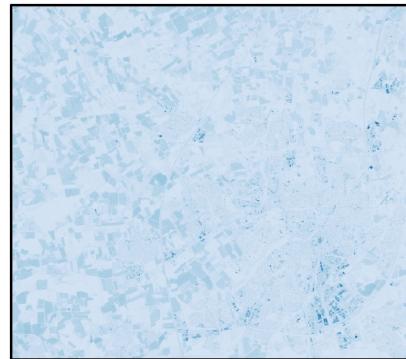
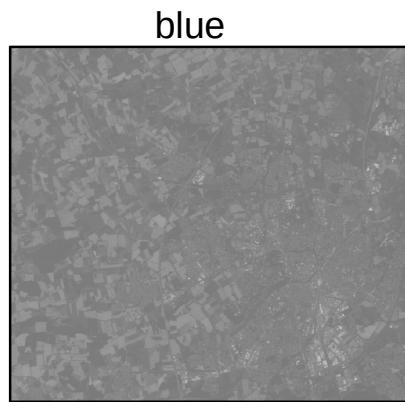
- Our eyes have three “sensors”: Blue, Green, Red
- Different colors through color mixing
- Combine visible channels to create a “true color composite”

## How to do a color composite in R

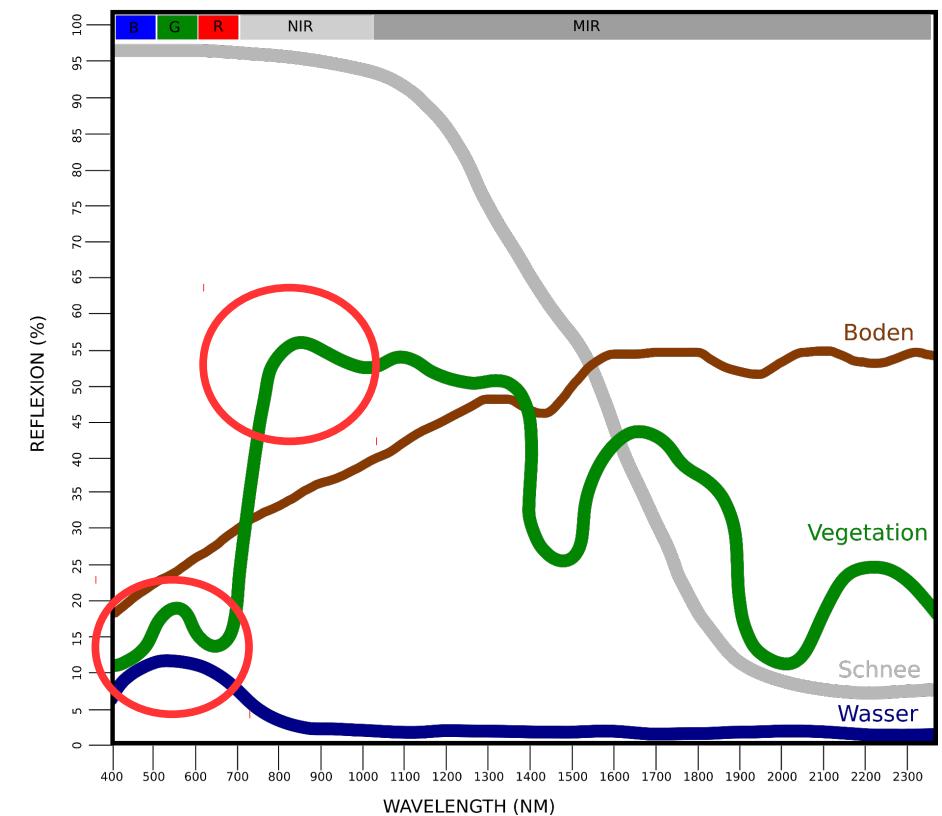
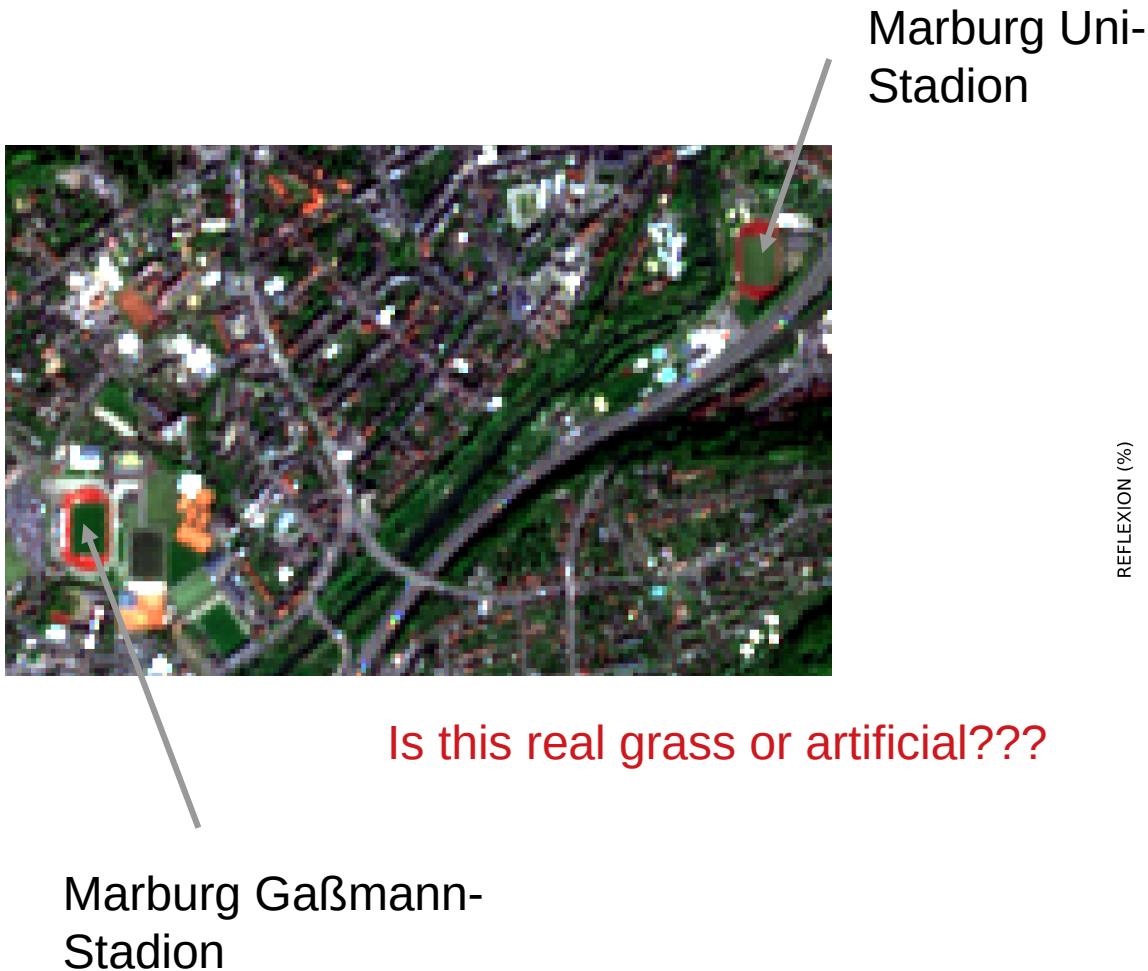
```
sen <- stack("blue.tif", "green.tif",
            "red.tif", "NIR.tif")
plotRGB(sen, r=3, g=2, b=1, stretch="lin")
```



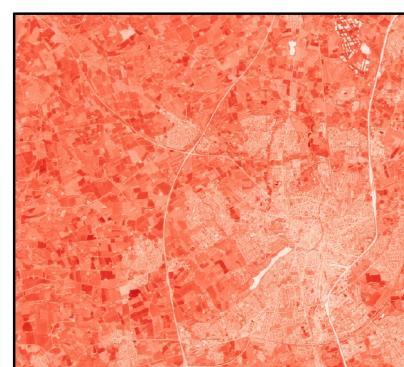
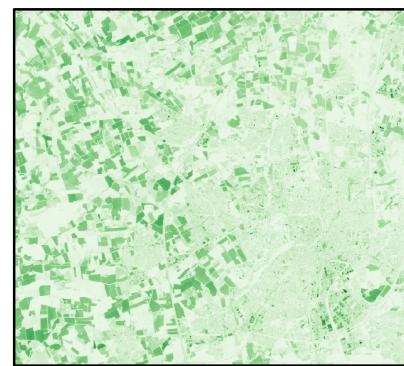
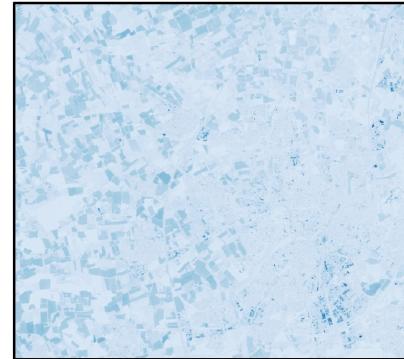
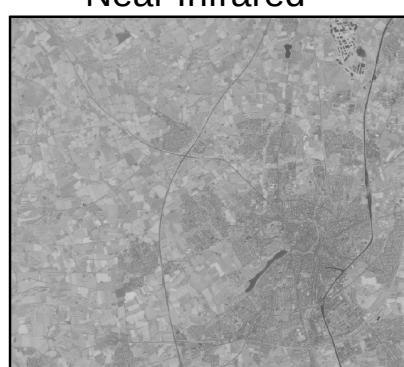
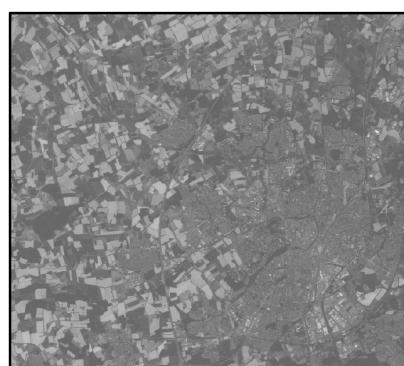
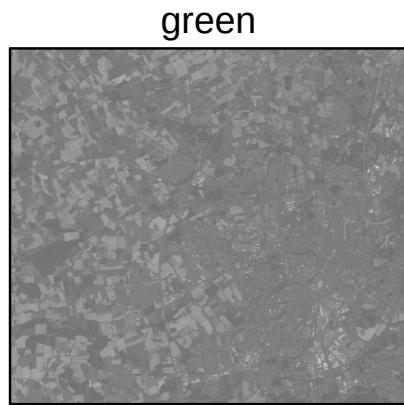
# How do we get the “color” in the satellite data?



# But satellite data contain more than what we can see...



# False color composite

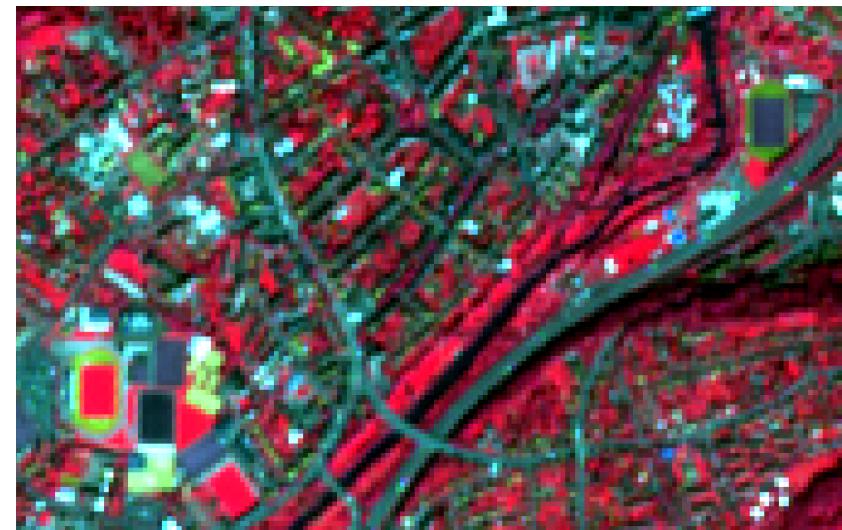


```
sen <- stack("blue.tif","green.tif",
             "red.tif","NIR.tif")
plotRGB(sen,r=4,g=3,b=2,stretch="lin")
```

# False color composite

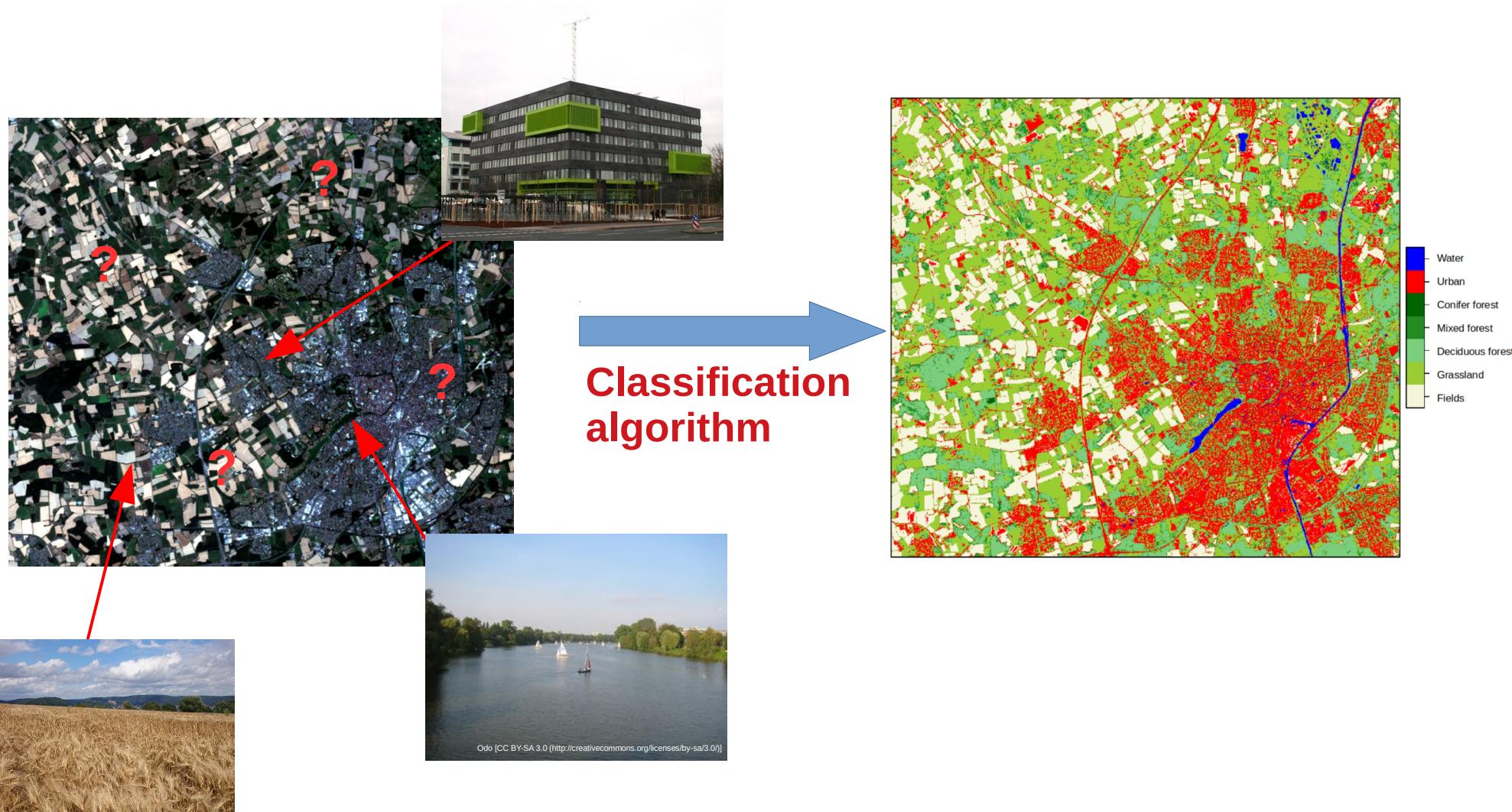


Marburg Uni-Stadion

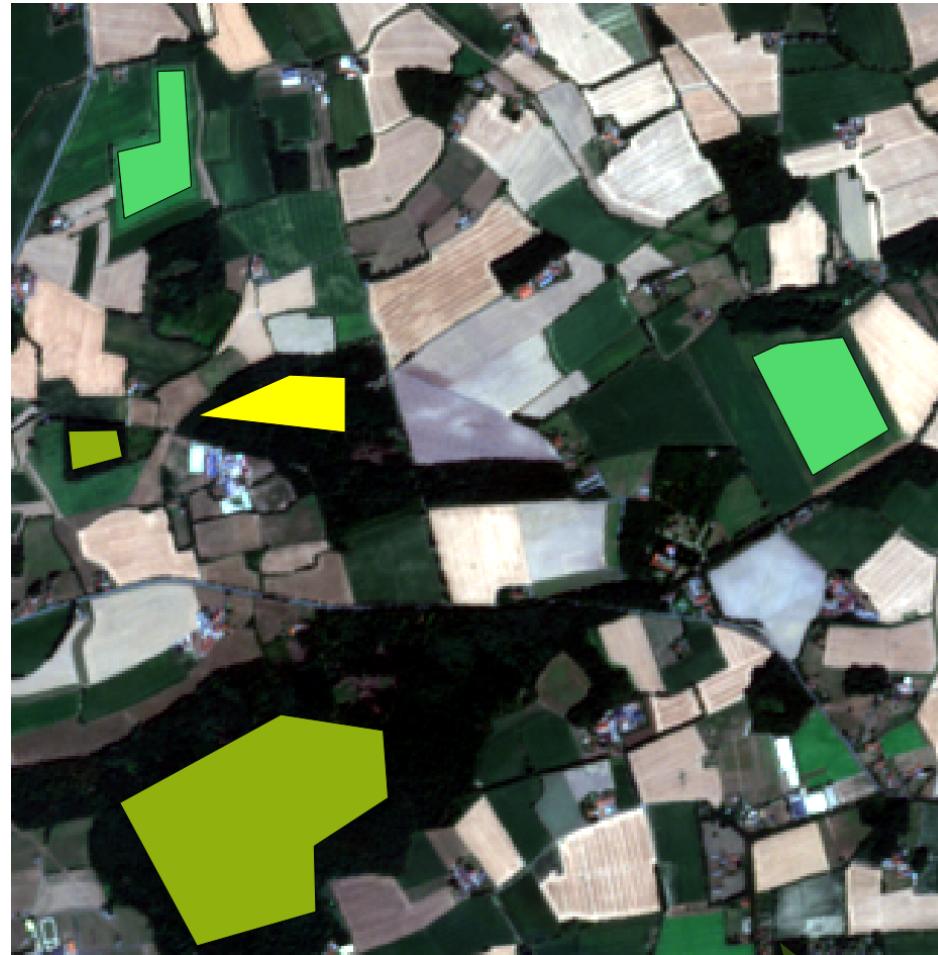


Marburg Gaßmann-Stadion

# How to relate the spectral properties to land cover classes?



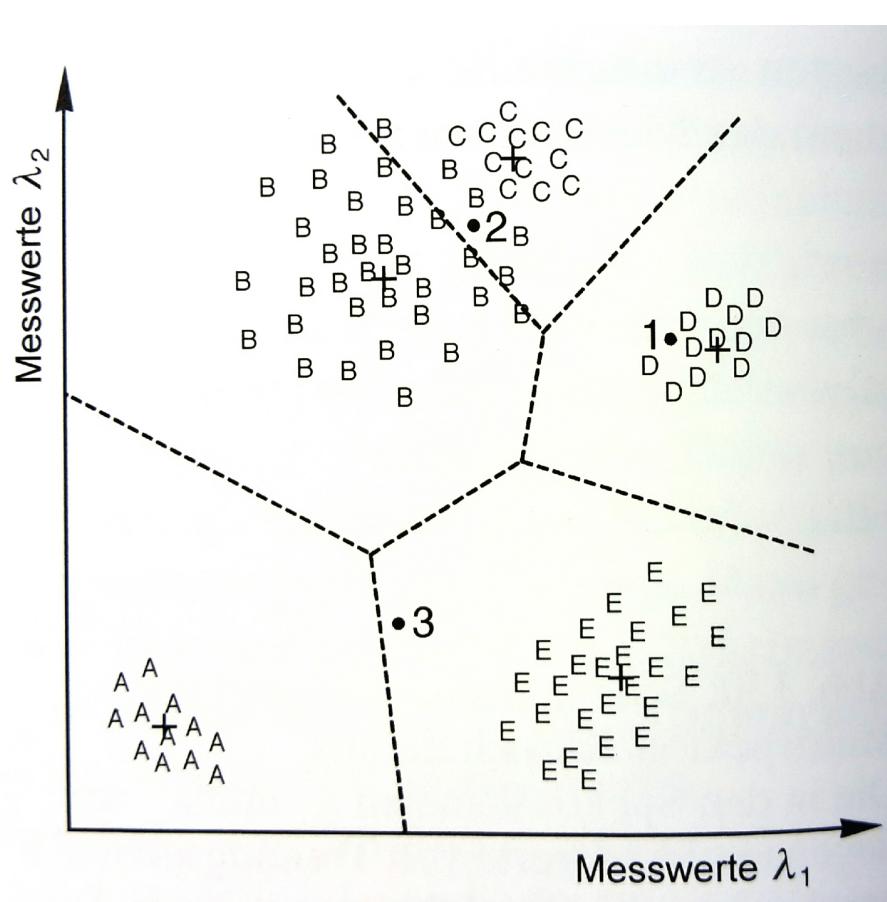
# Reference data



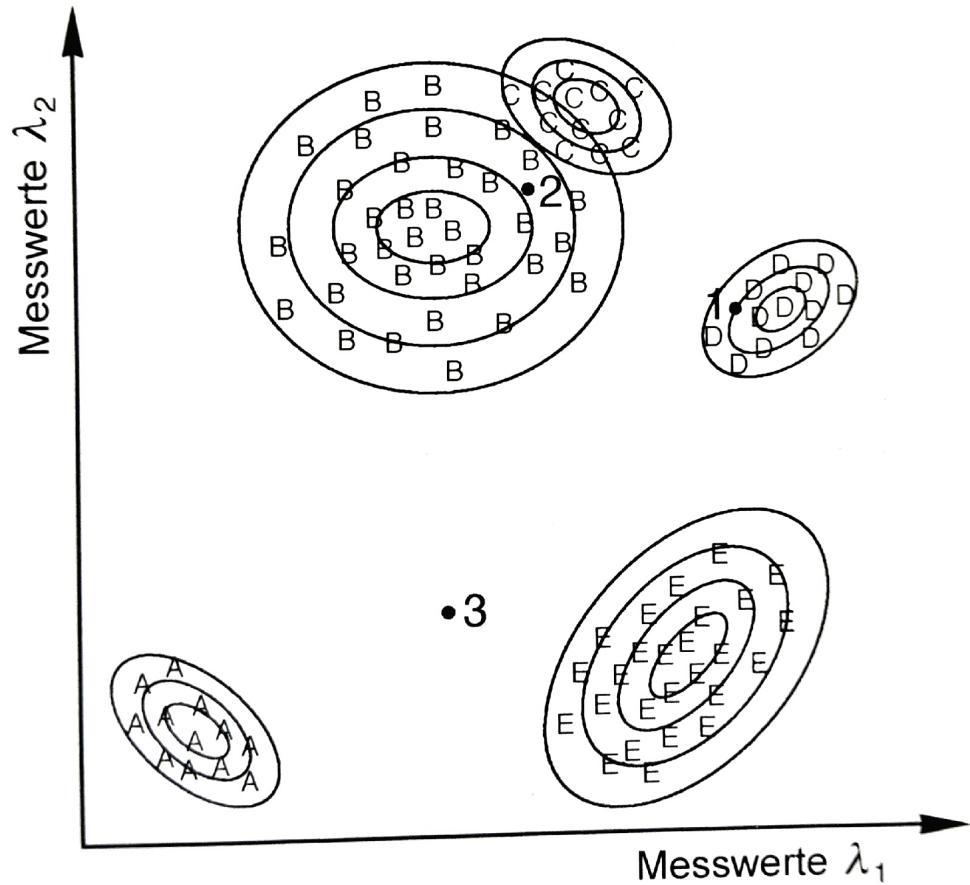
Training data from field work, expert knowledge,...  
either polygons taken with GPS or digitized e.g. using QGIS

# Traditional Supervised Classifiers

Minimum distance classifier

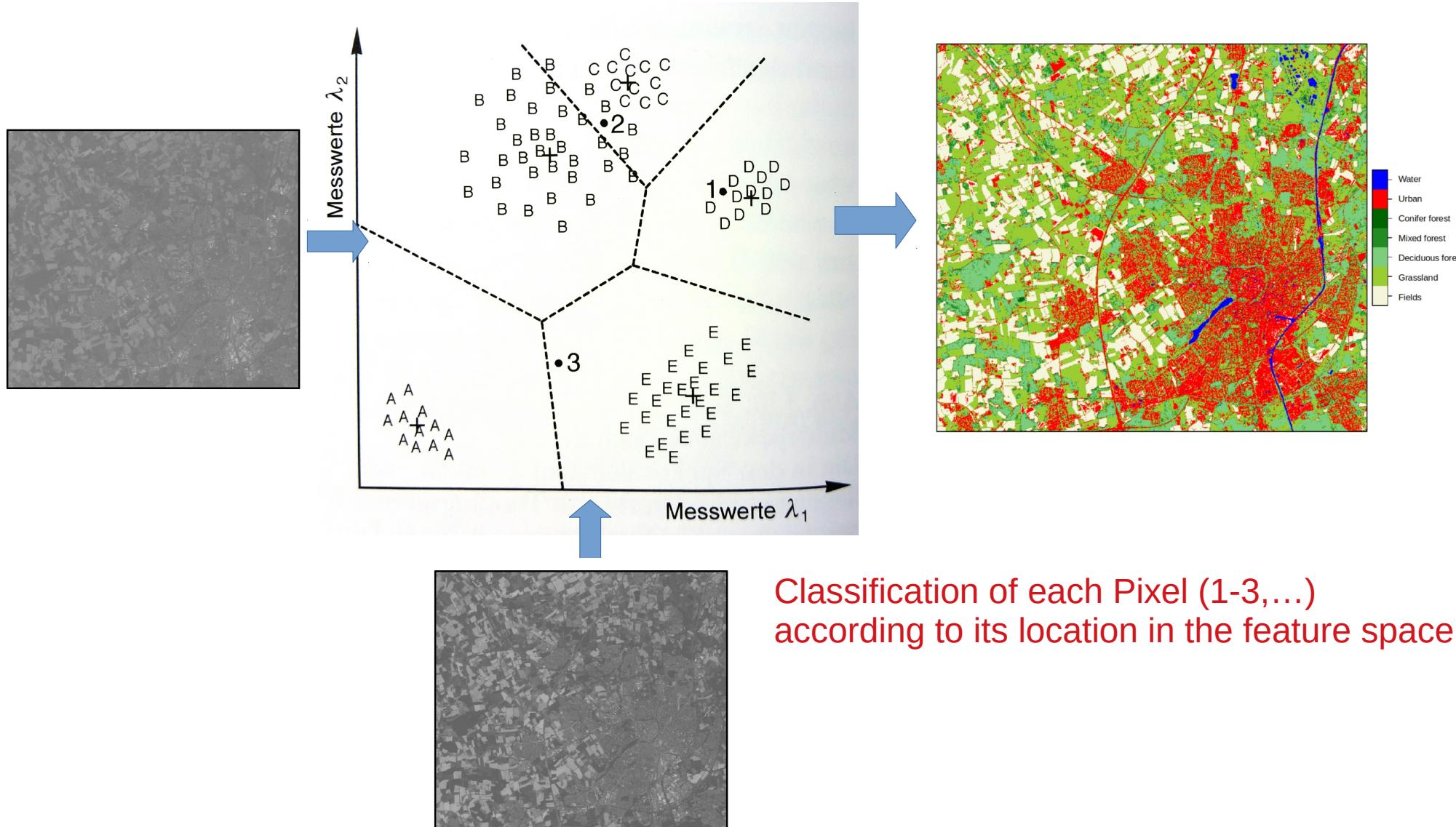


Maximum likelihood classifier



Albertz (2009): Einführung in die Fernerkundung. WBG, Darmstadt

# Image Classification

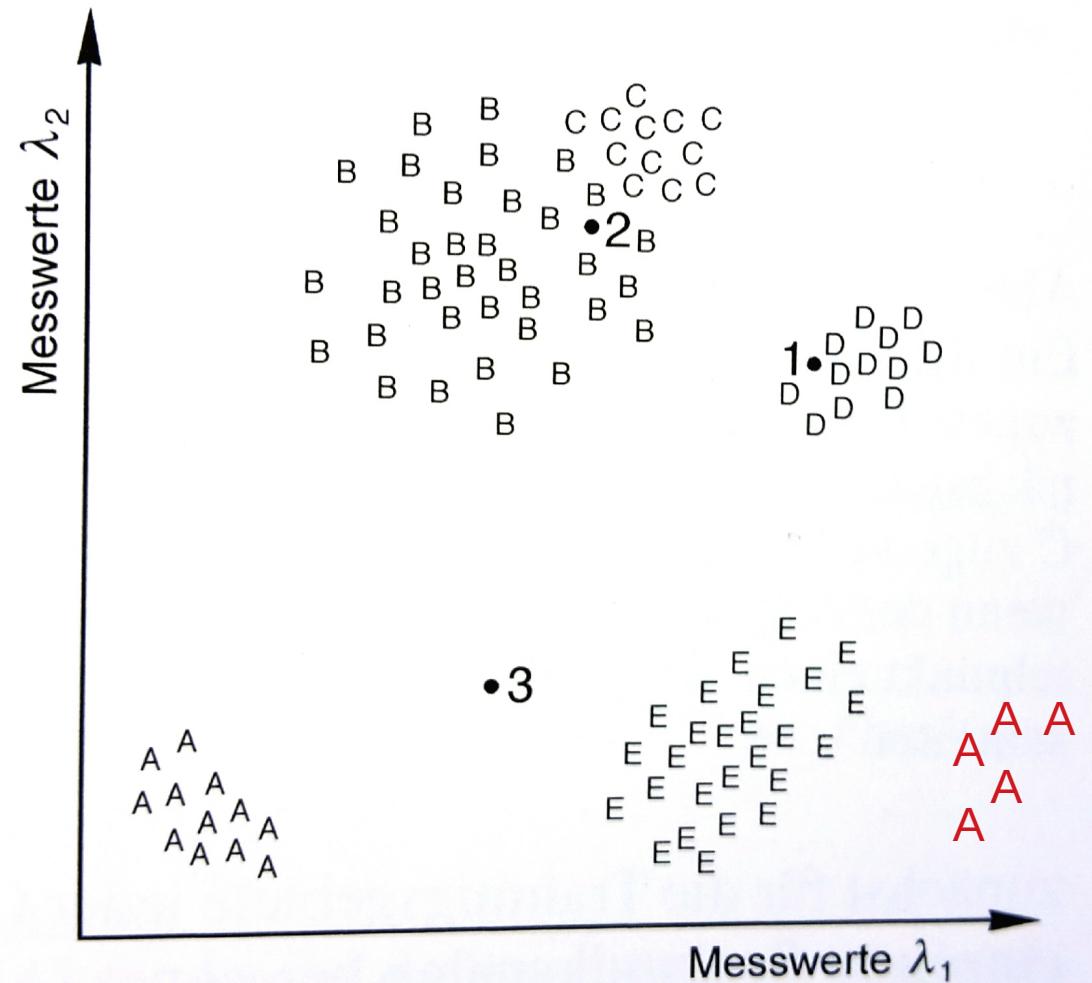


# Traditional Classifiers

**Complex patterns cannot be accounted for by these algorithms!**

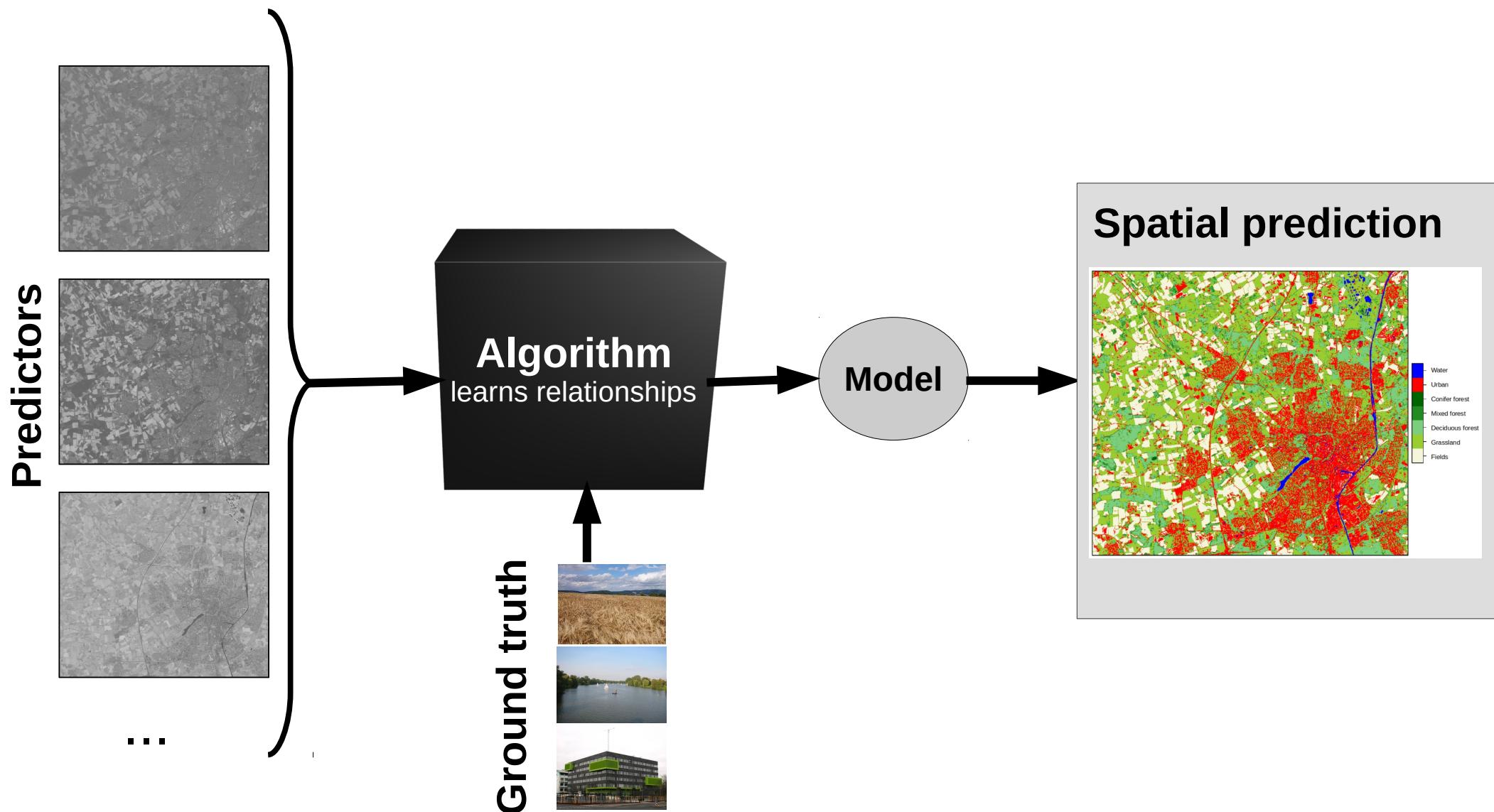
E.g. variations in the spectral characteristics depending on

- Forest on slopes with different aspect
- Crops that are irrigated/non irrigated
- Water in streams/lakes
- ...

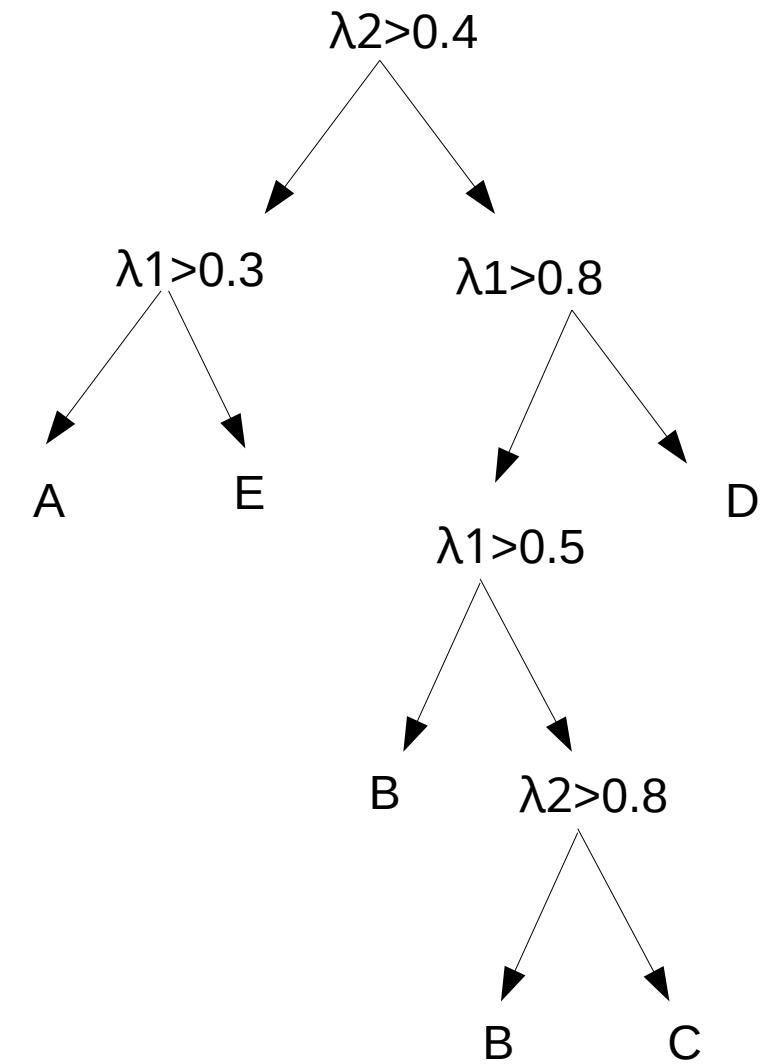
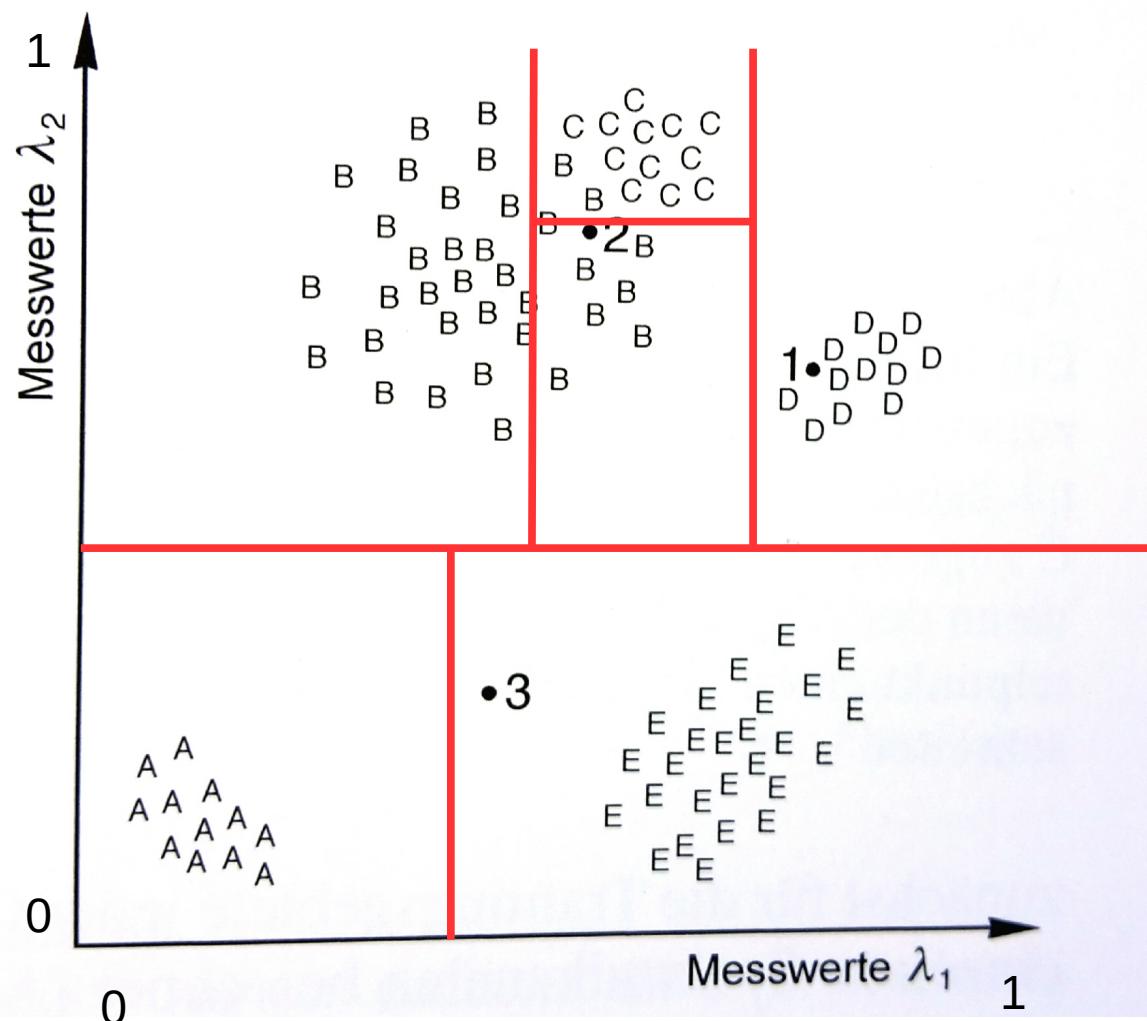


→ we need more complex models

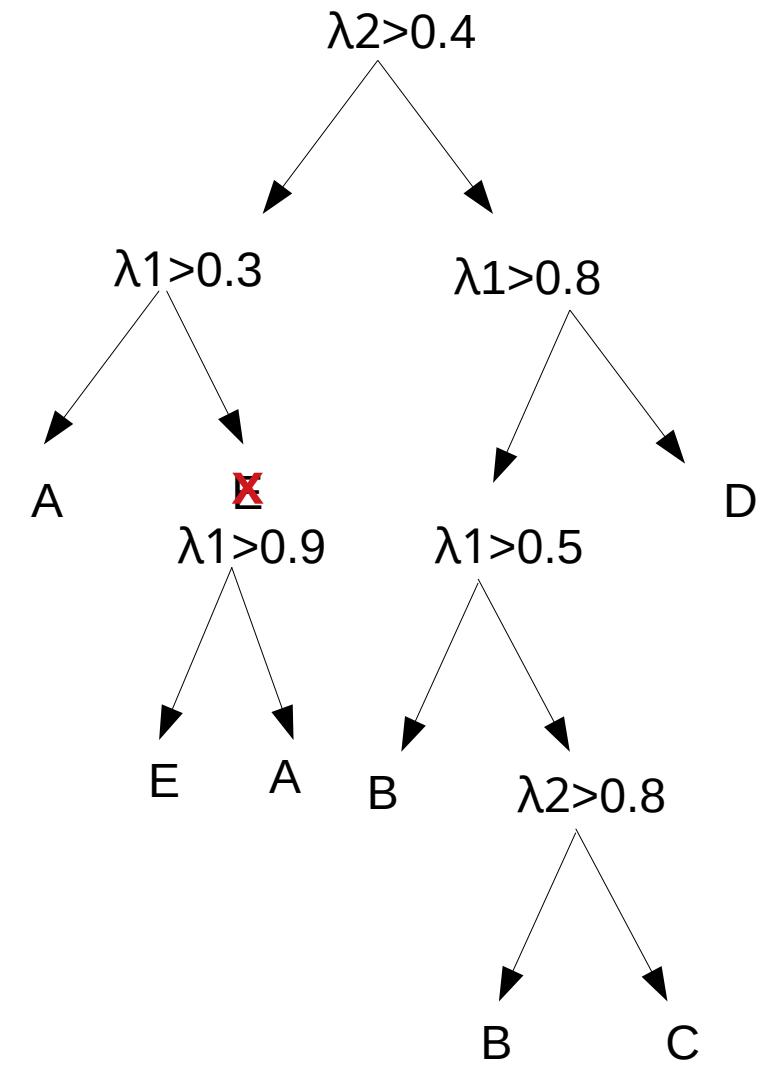
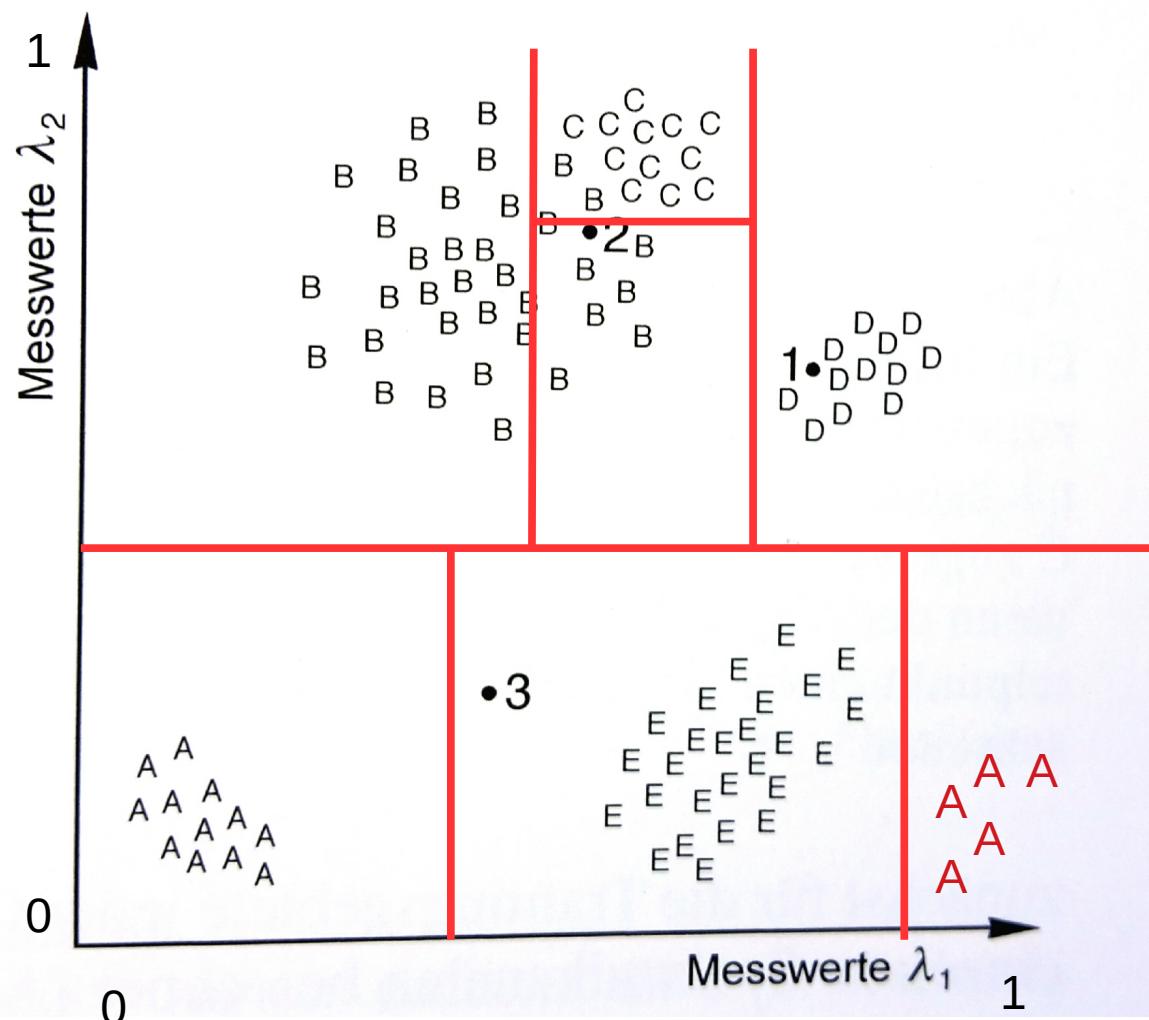
# Classification with machine learning



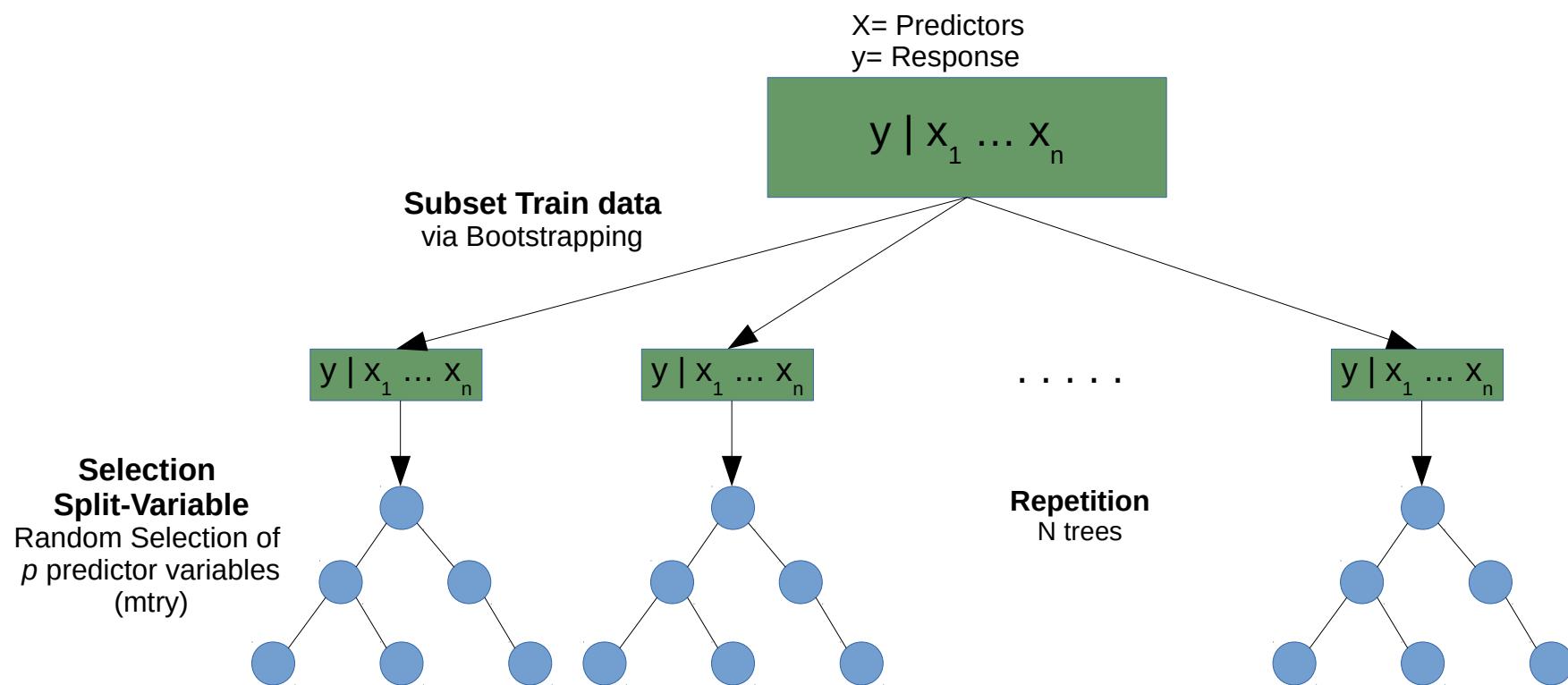
# Random Forest basics: Classification trees



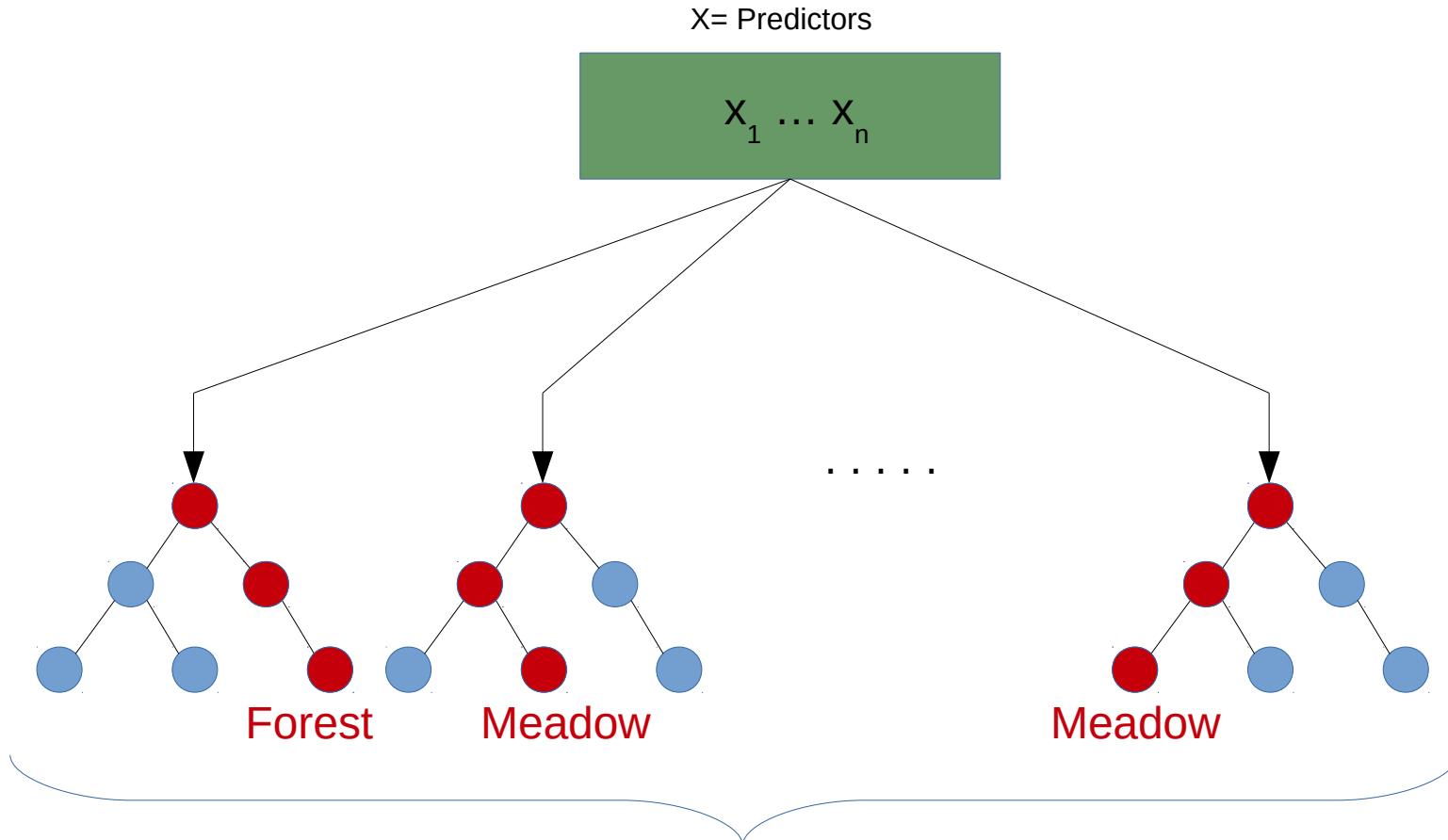
# Random Forest basics: Classification trees



# Random Forest: Model training



# Random Forest: Prediction

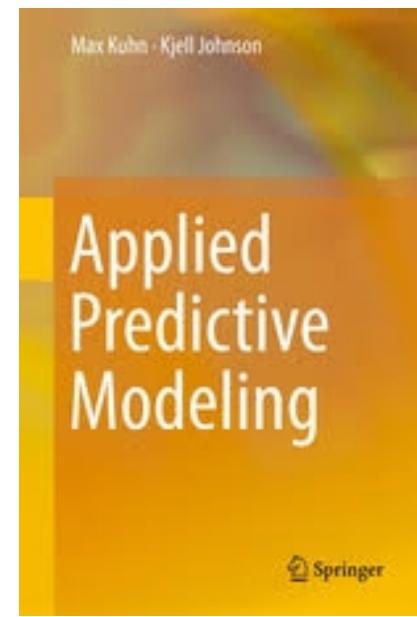


Decision made by combining the individual  
trees  
(Classification: Majority; Regression: Mean)

# Machine learning in R

- Many packages for different ML algorithms (e.g. Random Forests, Neural Networks, Support Vector Machines, ...)
- For classification and regression problems
- Wrapper packages
  - allowing access to many algorithms via a unified syntax
  - Supporting functionality for cross-validation etc.
  - **Caret (Classification And REgression Training)**
  - **Mlr (Machine Learning in R)**
  - **Tidymodels**

For today's session



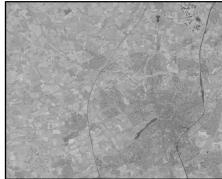
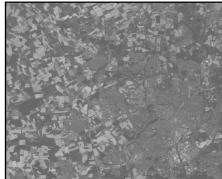
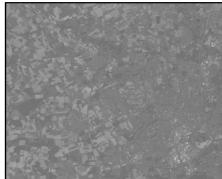
# Step 1: Model training in R

Predictors					Response
B02	B03	B04	B08	...	class
1	857	632	387	308	Water
2	848	633	389	312	Water
3	843	624	357	343	Water
4	854	630	360	333	Water
5	854	628	376	302	Water
6	859	615	364	350	Water

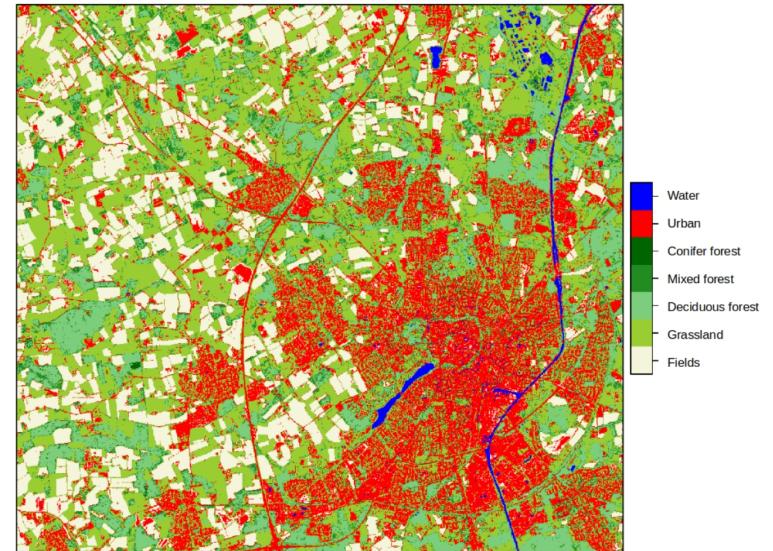
## How to do it in R

```
library(caret)
model <- train(predictors,
                 response,
                 method="rf")
```

# Step 2: Model prediction in R



+ trained model =

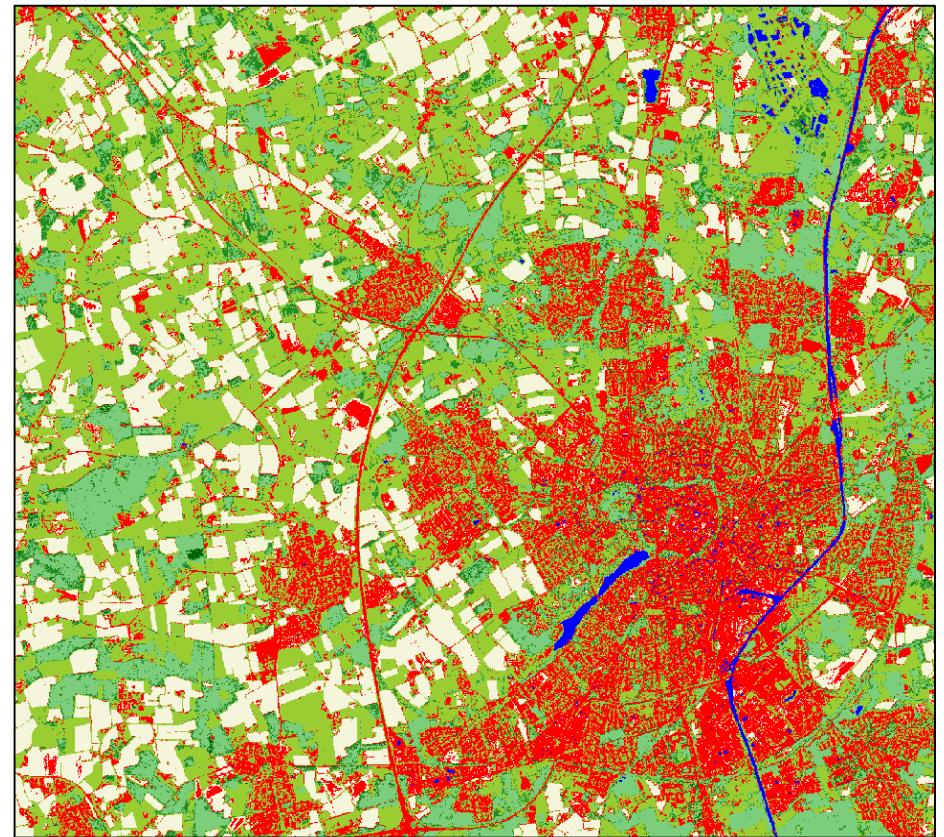


...

## How to do it in R

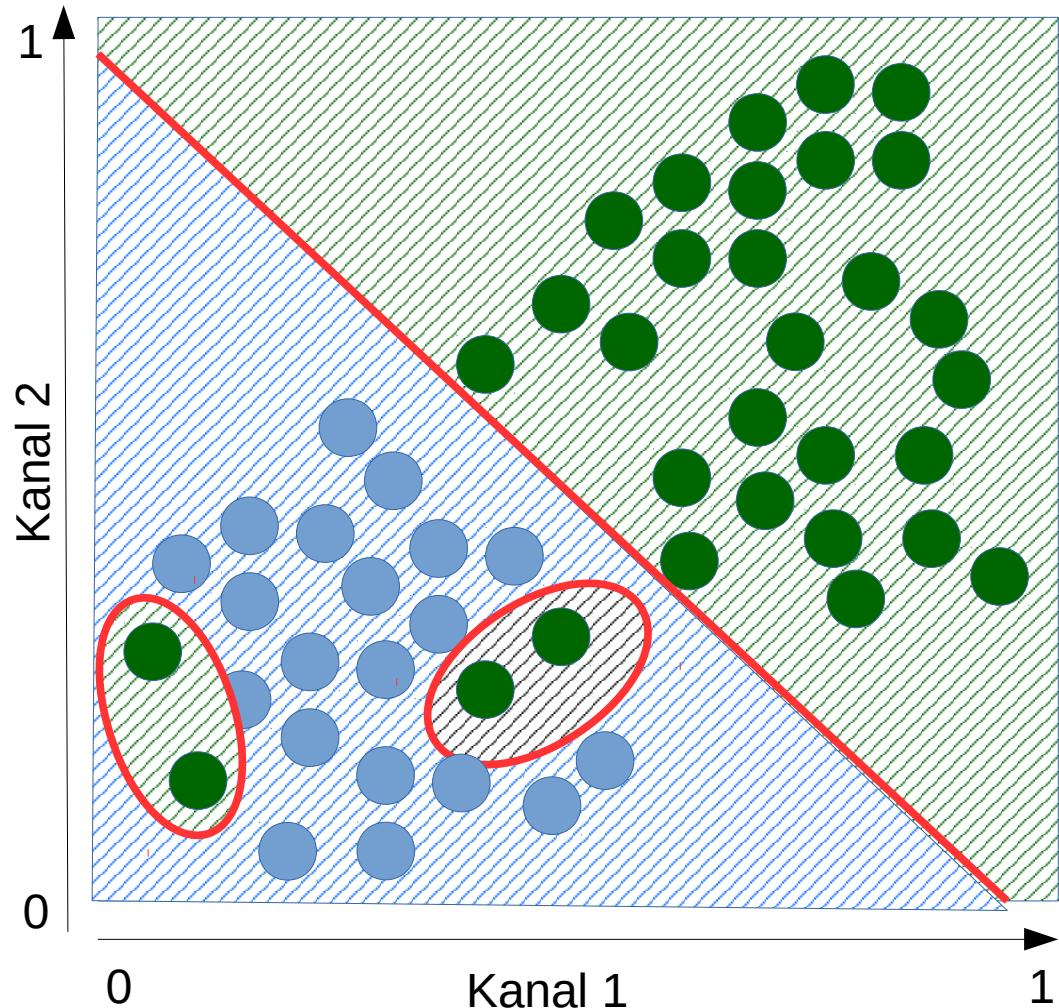
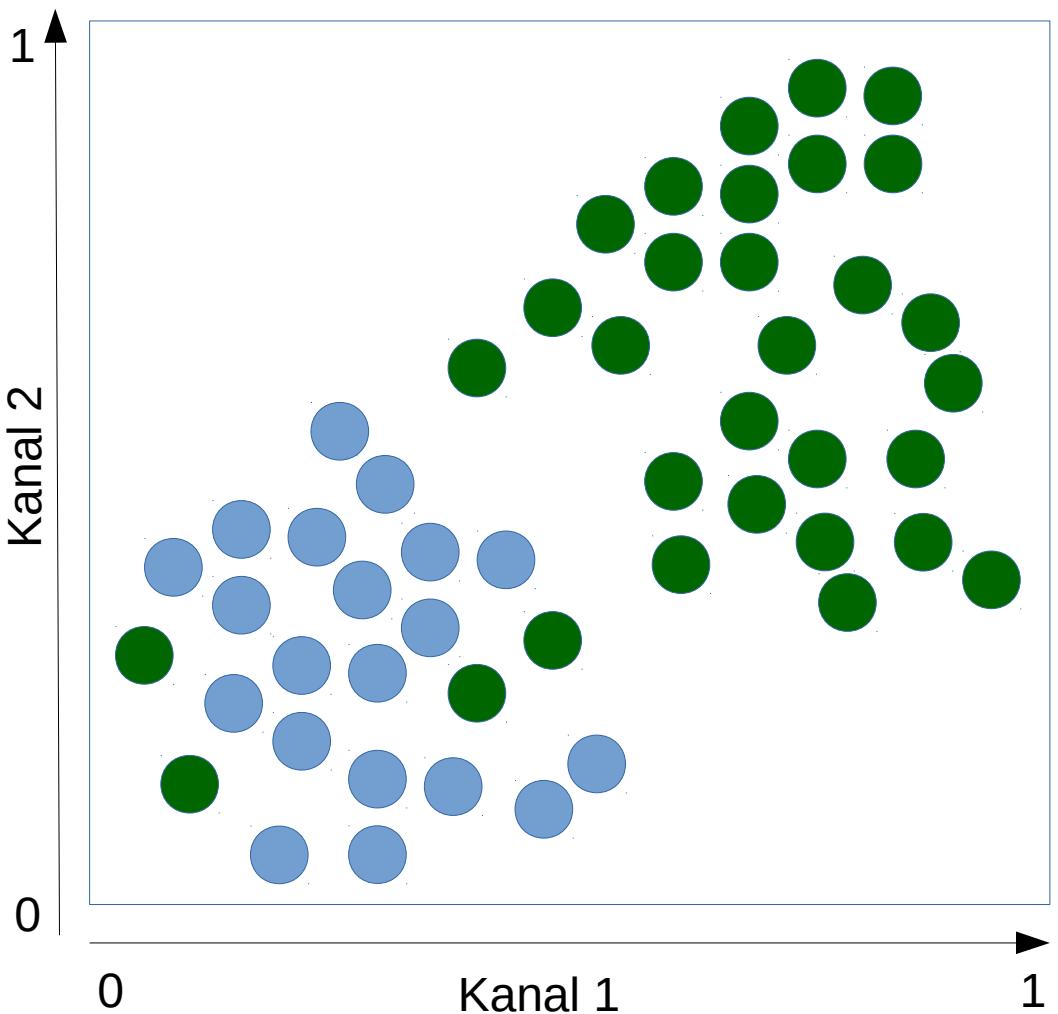
```
library(raster)
pred_sp <- stack(predictors)
prediction <- predict(pred_sp, model)
```

# Result

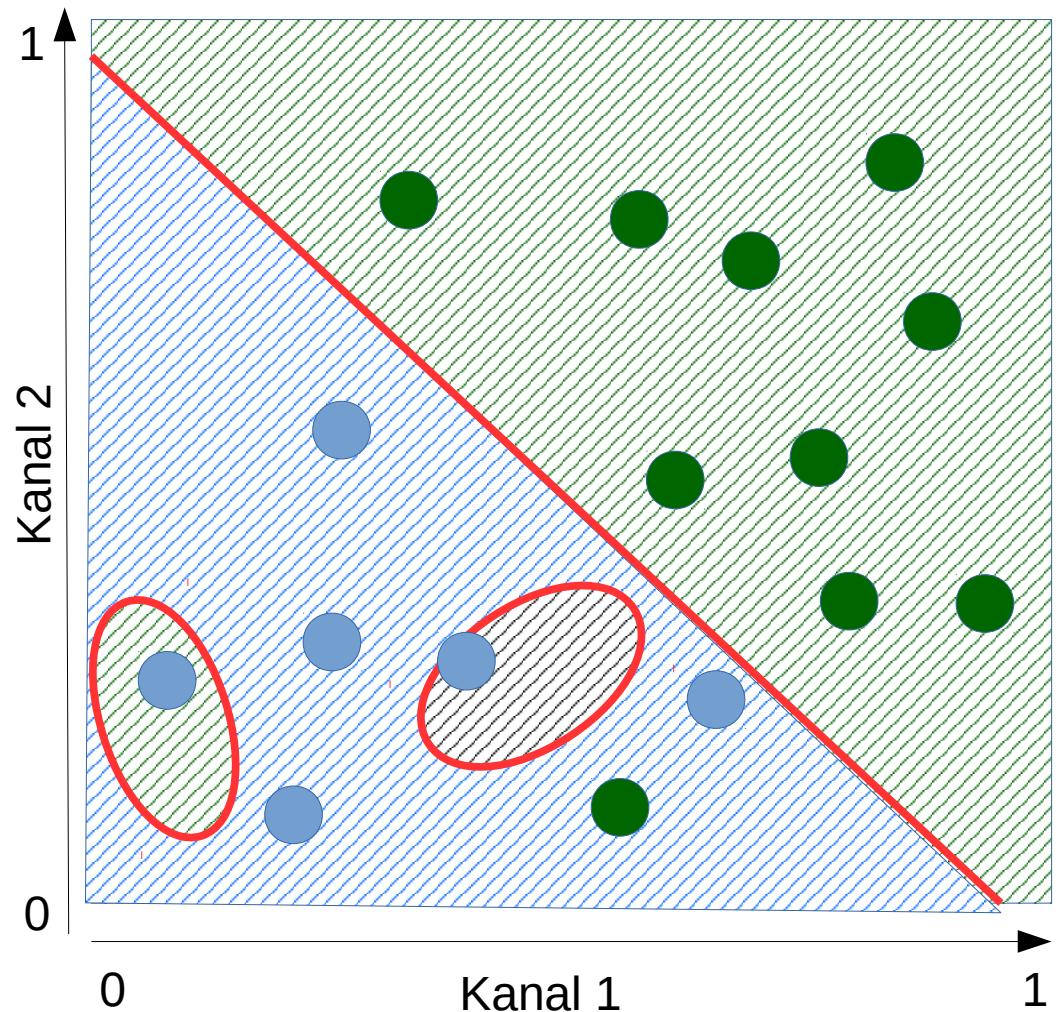
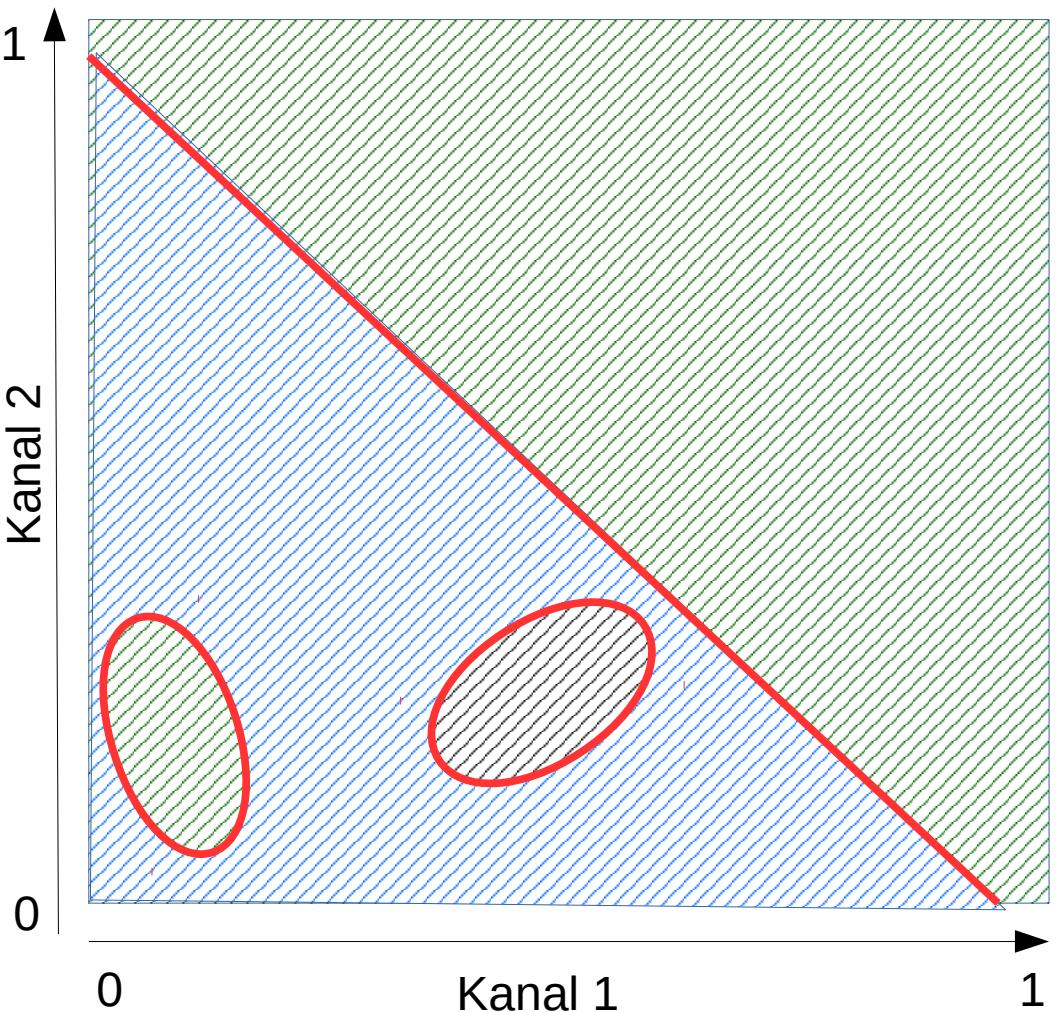


How good is this map?

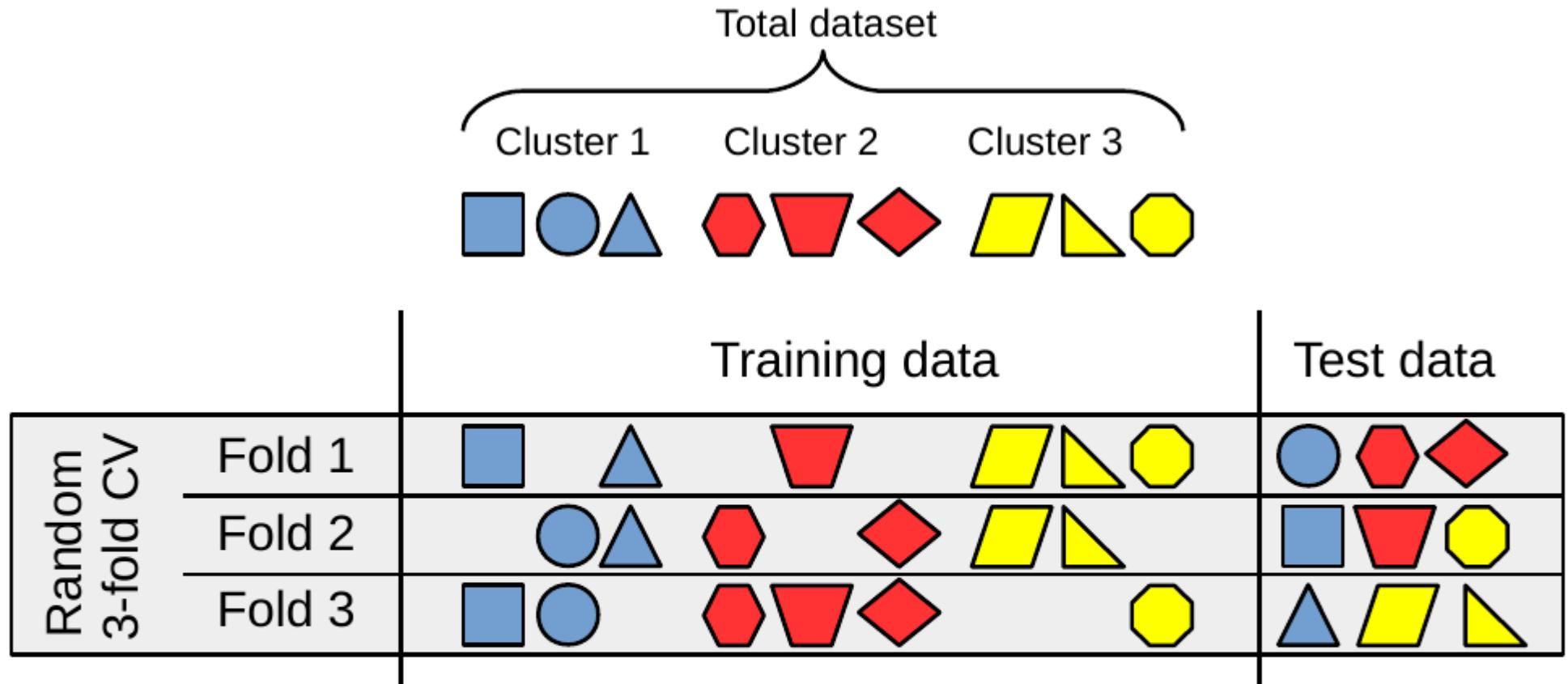
# Problem: Overfitting



# Problem: Overfitting



# “Default” random cross-validation



# “Default” random cross-validation

## How to do it in R

```
model <- train(predictors,
                 response,
                 method="rf",
                 trControl=trainControl(method="cv"))
```

```
> model
Random Forest

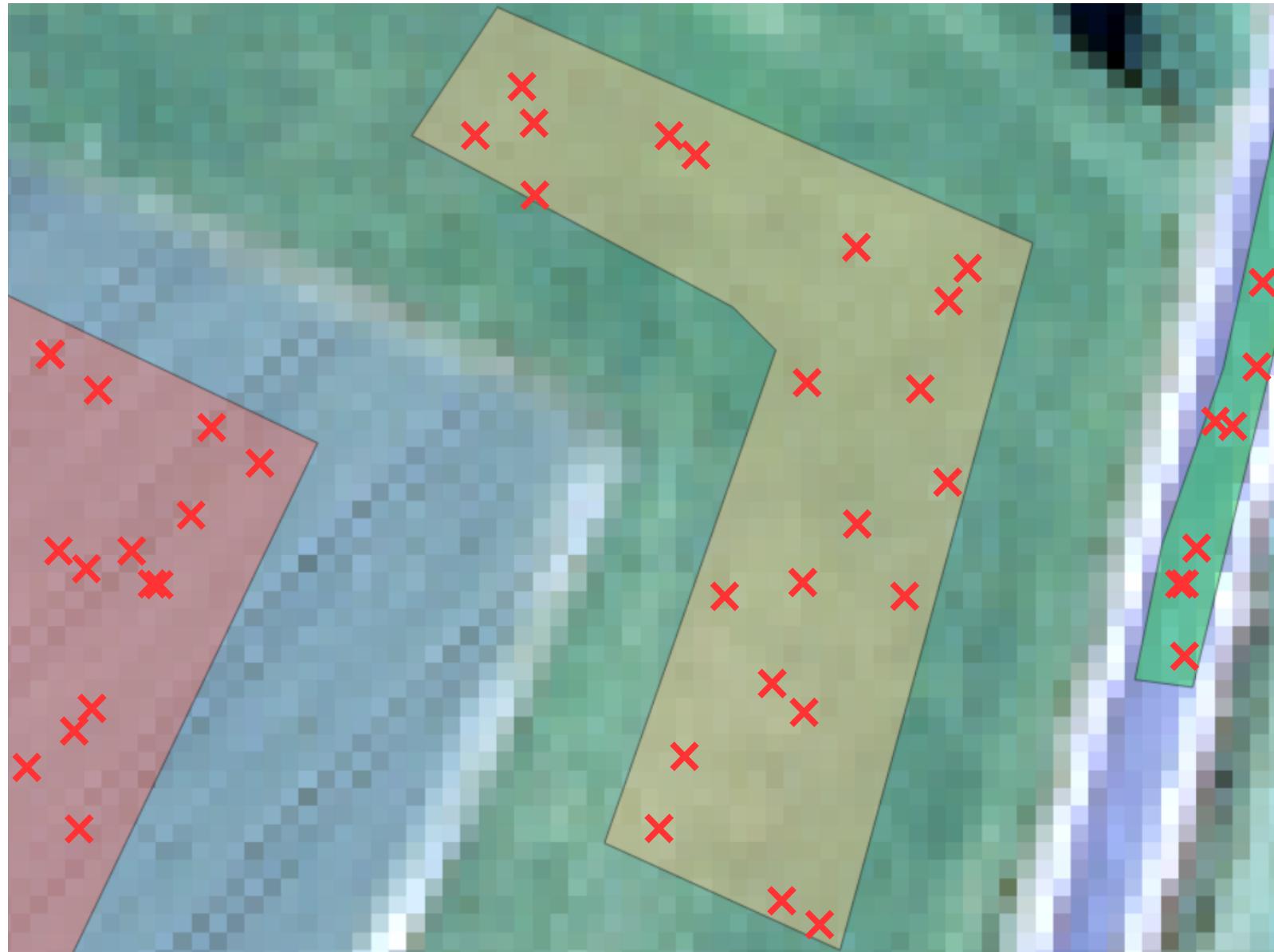
3962 samples
 16 predictor
  9 classes: 'Feld_bepfl', 'Feld_unbepfl', 'Gewaesser', 'Laubwald', 'Mischwald', 'Nat_Feuchtw', 'Siedlungsgebiet', 'StrGruenland'
```

```
No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3566, 3565, 3567, 3565, 3566, 3565, ...
Resampling results across tuning parameters:
```

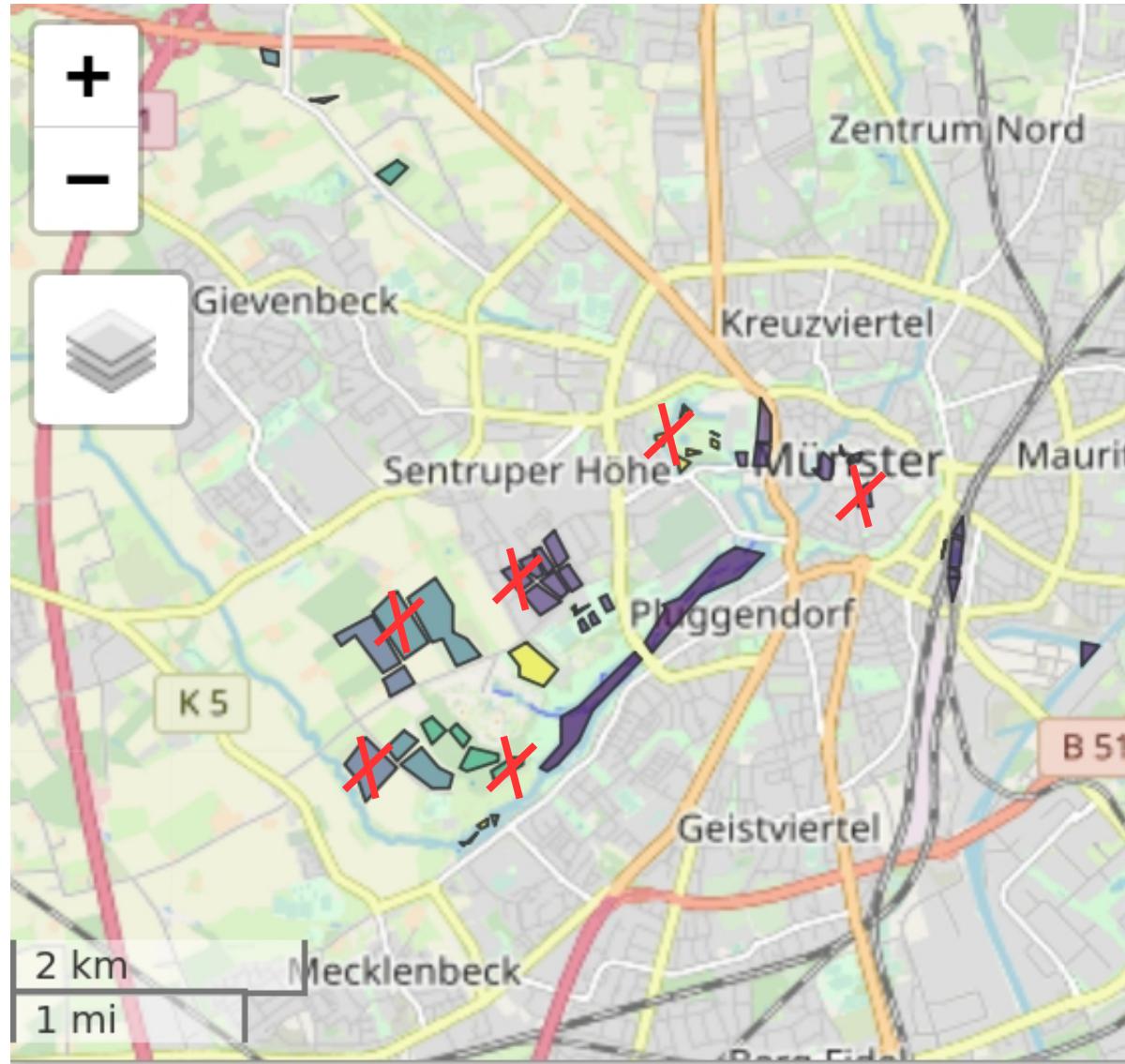
mtry	Accuracy	Kappa
2	0.9512871	0.9426287
9	0.9545617	0.9465151
16	0.9507795	0.9420838

Kappa was used to select the optimal model using the largest value.  
The final value used for the model was mtry = 9.

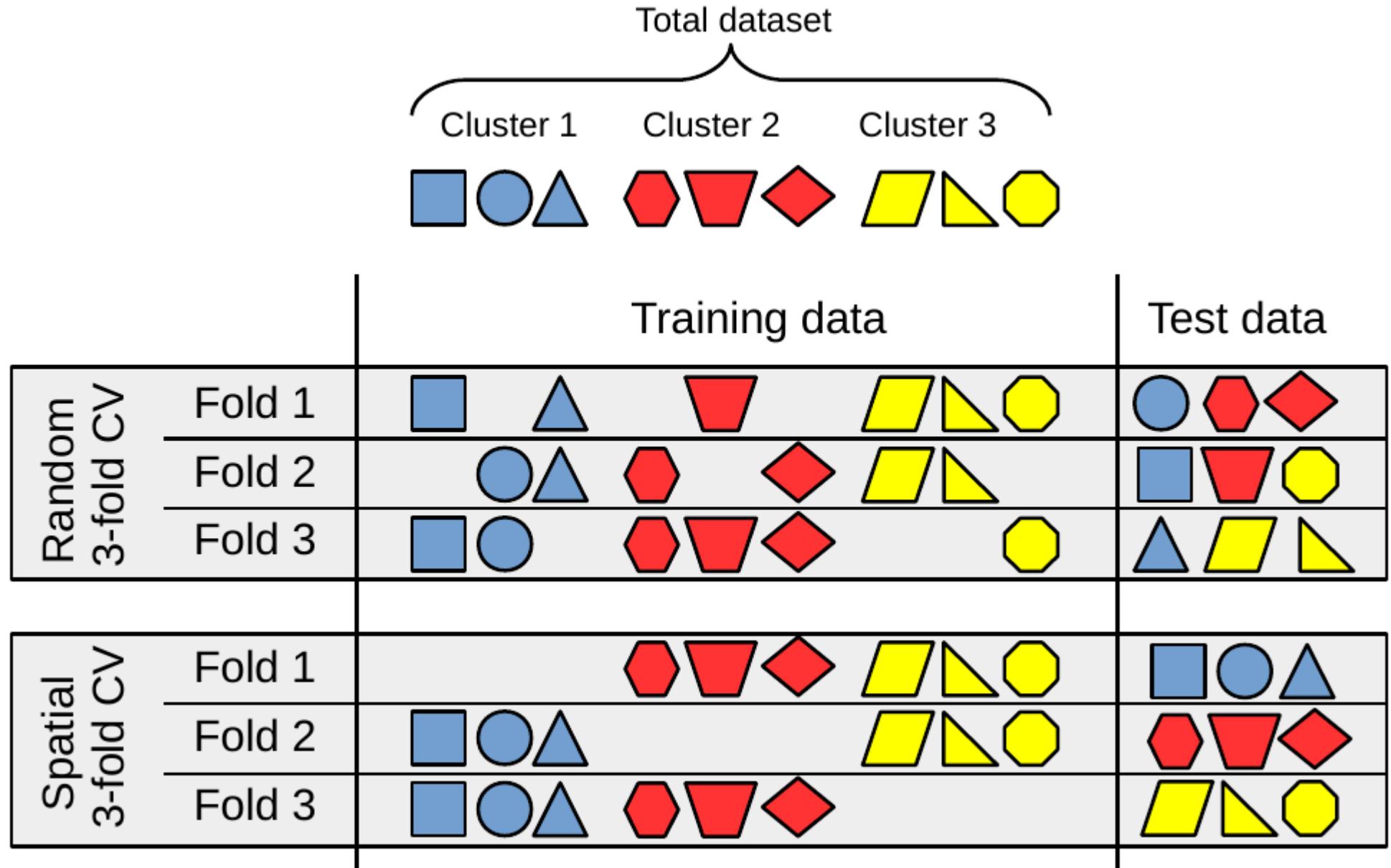
# “Default” random cross-validation



# Model validation



# Spatial cross-validation



# Spatial cross-validation

## How to do it in R

```
library(CAST)
indices <- CreateSpacetimeFolds(trainingData,
                                  spacevar="Station")
model <- train(predictors,
                response,
                method="rf",
                trControl=trainControl(method="cv",
                                        index = indices$index))

> model
Random Forest

3962 samples
 16 predictor
  9 classes: 'Feld_bepfl', 'Feld_unbepfl', 'Gewaesser', 'Laubwald',
'Mischwald', 'Nadelwald', 'Renat_Feuchtw', 'Siedlungsgebiet', 'StrGruenland'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3650, 3626, 3483, 3785, 3768, 3791, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  2     0.6110833  0.4846005
  9     0.6590675  0.5467704
  16    0.6568454  0.5481304
```

Kappa was used to select the optimal model using the largest value.  
The final value used for the model was mtry = 16.

PolygonID/  
Spatial block/...

# Overfitting due to “unsuitable predictors”



<https://gis.stackexchange.com/questions/111932/classified-images-of-randomforest-classification-look-clustered>

# Spatial variable selection

## How to do it in R

```
library(CAST)
indices <- CreateSpacetimeFolds(trainingData,
                                  spacevar="Station")
model <- ffs(predictors,
              response,
              method="rf",
              trControl=trainControl(method="cv",
                                      index = indices$index))
```

```
> model
```

```
Random Forest
```

```
3962 samples
5 predictor
9 classes: 'Feld_bepfl', 'Feld_unbepfl', 'Gewaesser', 'Laubwald', 'Mischwald',
nat_Feuchtw', 'Siedlungsgebiet', 'StrGruenland'
```

```
No pre-processing
```

```
Resampling: Cross-Validated (10 fold)
```

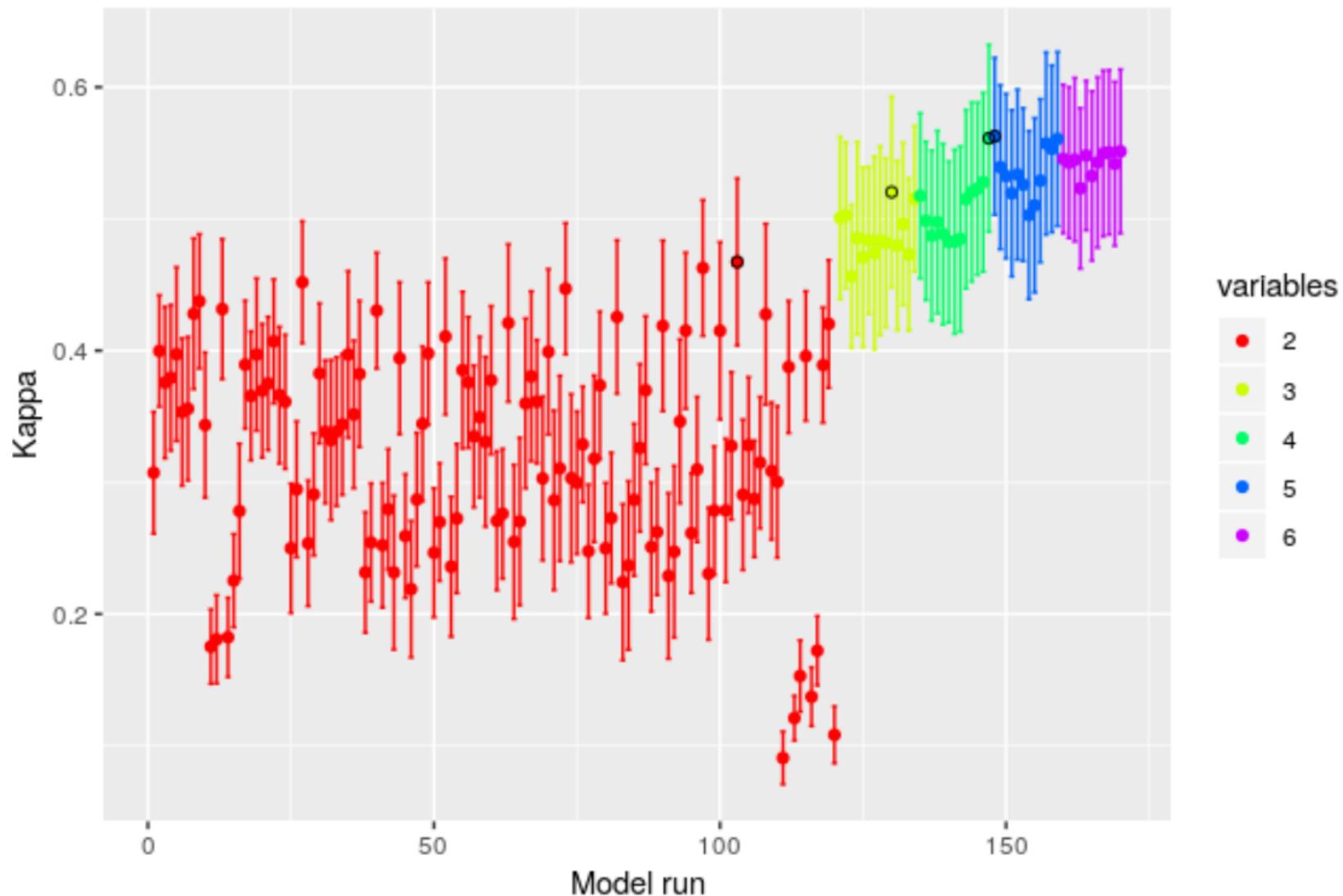
```
Summary of sample sizes: 3650, 3626, 3483, 3785, 3768, 3791, ...
```

```
Resampling results:
```

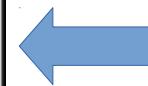
Accuracy	Kappa
0.6702505	0.5626794

```
Tuning parameter 'mtry' was held constant at a value of 1
```

# Spatial variable selection



# Summary



That's what we need to avoid!

<https://xkcd.com/1838/>

# From the theory to practice...

# Practice

- Basic model training for land cover classification
- Spatial cross-validation
- Spatial variable selection



→ Slides and material: [https://github.com/HannaMeyer/OpenGeoHub\\_2019](https://github.com/HannaMeyer/OpenGeoHub_2019)