TIME SERIES LAND COVER MAPPING OF BARINGO COUNTY IN 2022, 2012 AND 2002

This project was completed by John Paul Mbuthia at the Department of Resource Surveys and Remote Sensing (DRSRS). I am very thankful to my instructor, Mr. Robert, Mr. Geoffrey, and Mr. Otero for their patience and guidance as I undertook this project.

1. Introduction

Land plays an important role in the socioeconomic conditions of humans. Baringo county is known to have two lakes, Lake Baringo at 130 km^2, and Lake Bogoria an alkaline lake with an area of 34 km^2. Despite having two lakes in one county, Baringo also has vast large tracts of land which are forests, farmlands, grasslands, built-up areas, and other lands. Over the years with the effect of climate change, it has been quite difficult to monitor the land cover changes that have occurred in the ground. Using remote sensing technology helped by the statistical language of R, new land cover classes of Baringo have been created using the Intergovernmental Panel on Climate Change (IPCC) land cover classes which are wetlands, farmlands, grasslands, forests, built up areas and other lands. These classes will help outline area changes which will be in hectares of land cover of Baringo county over a 3-decade span. Not only does this project help in detecting changes in land cover but also contributes significantly to the mitigation of climate change, including the promotion of sustainable management of forests, water bodies, and terrestrial ecosystems.

2. Methodology

2.1 Study Area

This study was implemented within Baringo county as displayed on figure 1 below.

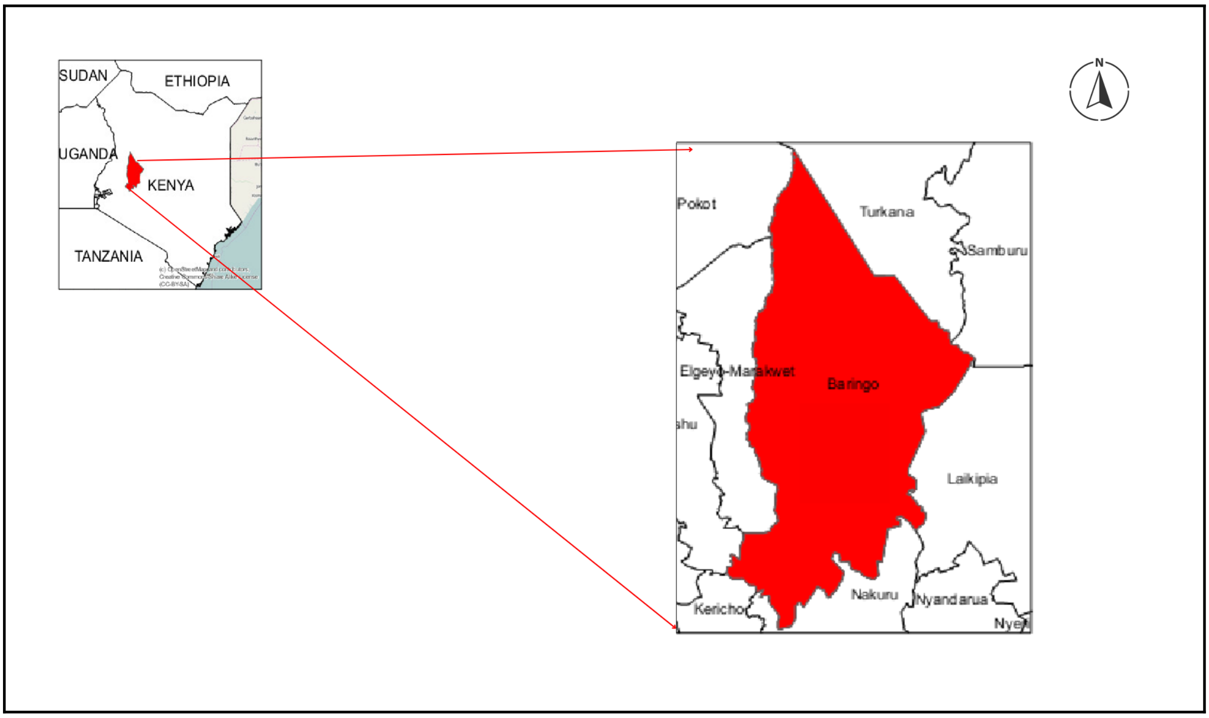


Figure 1: A map showing Baringo County as the study area

2.2 Flow Chart

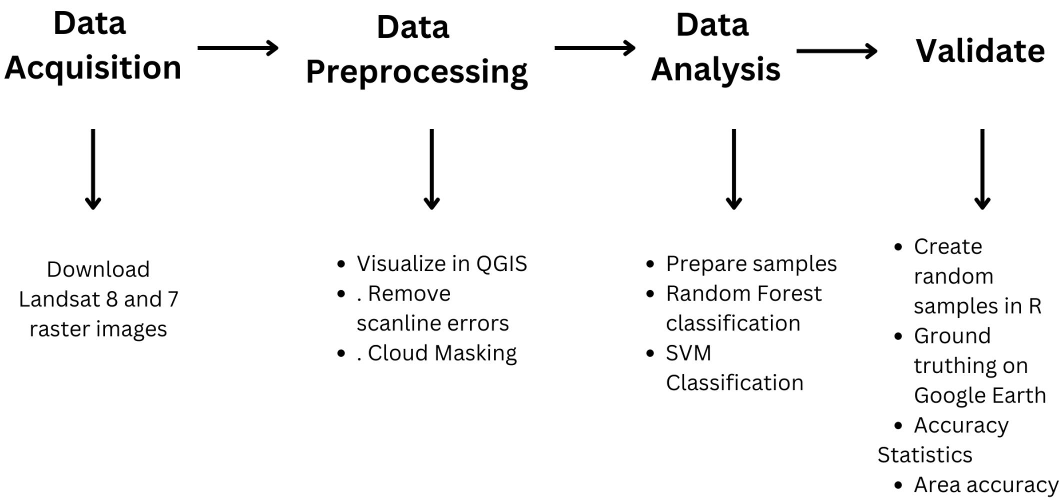


Figure 2: Methodology flow chart

2.3 Image processing

To develop IPCC land cover classes which are wetlands, croplands, forest land, grassland and settlements, landsat 8 and 7 data were used. Band 1, 2, 5, and 7 were used to perform the supervised classification of Baringo County using the IPCC classes. These 4 bands achieve the same classification results as using all bands in landsat 8 and 7.

2.3.1 Data Acquisition

The raster datasets were downloaded from usgs.gov at a resolution of 30 metres as shown in the table below.

Table 1: Data sources, period and resolution

Dataset	**Data Type**	**Resolution**	**Time**	**Source**
Landsat 8	Raster	30 metres	Feb, 2022	usgs.gov
Landsat 7	Raster	30 metres	Feb, 2012	usgs.gov
Landsat 7	Raster	30 metres	Feb, 2002	usgs.gov

USGS allows cloud filtering of the raster images. Setting the filter to 15% allows very minimal cloud cover in the raster tiles. After acquiring the required datasets, the three tiles representing Baringo County were visualized in QGIS. Mosaicking each band in the images was done, the three tiles would then end up as one large tile representing each band. This is followed by stacking the bands to form one raster by compositing the tile bands. For landsat 7, the scanline errors are removed using the mask layers available in USGS for the respective timeline. This is done before mosaicking or compositing the tiles. Once the raster images were composited, clipping was done using the Baringo County shapefile to have the Baringo County raster image of our interested years, 2022, 2012 and 2002.

Pixel based classification was done using QGIS. Each pixel represents a specific class. The six classes assigned to the pixels are wetland, forest, farmland, built-up, grassland and other lands. However, identifying pixels that represent built up areas was challenging due to spectral confusion with other land cover types. Depending on the user's accuracy, each pixel should be assigned correctly to its category. For example, an area inside Lake Baringo would be assigned to a wetland, and an area having a building or a forest each pixel would be assigned to its correct class respectively.

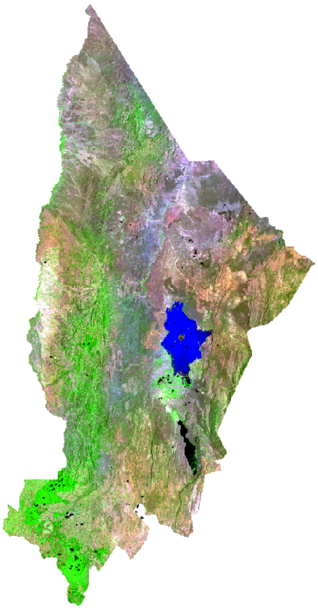


Figure 3: Baringo County landsat 8 image with training sites seen as small dark spots

Adding the new classes to the R script, a plot is made showing the reflectance value of each class per landsat 8 bands. All other classes have a spike reflectance value after band 2 except for class 1 which represents wetlands as shown in figure 4. These reflectance values play a very crucial role in the model training.

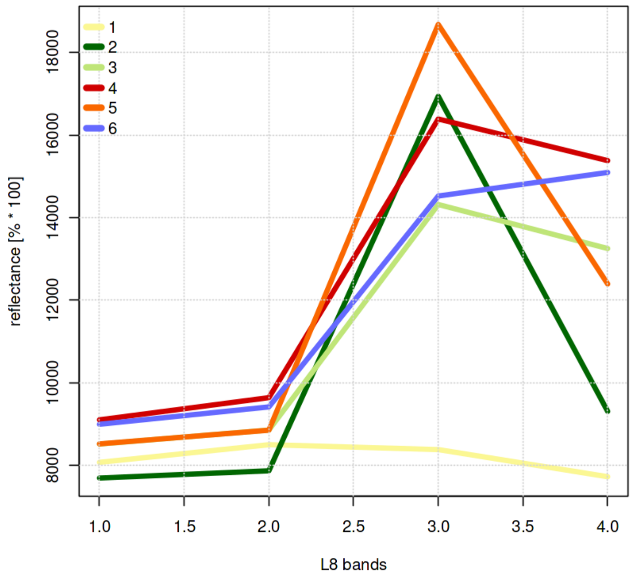


Figure 4: Reflectance value of training classes per bands

2.4 Model Building

In this part, the engineered features which are the samples with the reflectance value are used to build the machine learning model that does the classification. Two models are used which are the random forest model and support vector machine model. Comparison is then done between these two models to see how well they do the classification over the three decade span. The best performing model is finally used to do the supervised classification of Baringo County. In this study the best performing model is the random forest classification which is used to perform the classification.

Random Forest: An ensemble learning method that operates by constructing a multitude of decision trees.

Support Vector Machine (SVM): A powerful and versatile machine learning model capable of performing linear or nonlinear classification.

There is often an imbalance of training pixels in the training samples where one class is represented by a large number of pixels while the other is represented by a few samples. This often leads the classifier to over classify strongly represented values and under classify classes with small samples. To solve this error downsampling was done. Training samples are down-sampled to have an equal number of samples per each classification class. In this study each class was down-sampled to 30 values. Out Of Bag error (OOB) is estimated internally in the training phase as an unbiased estimate of the classification error. The random forest model below in figure 5 shows the relationship between the OOB error and the number of trees used. As the number of trees increases there is a decrease in the error which is a good sign for the random forest model.

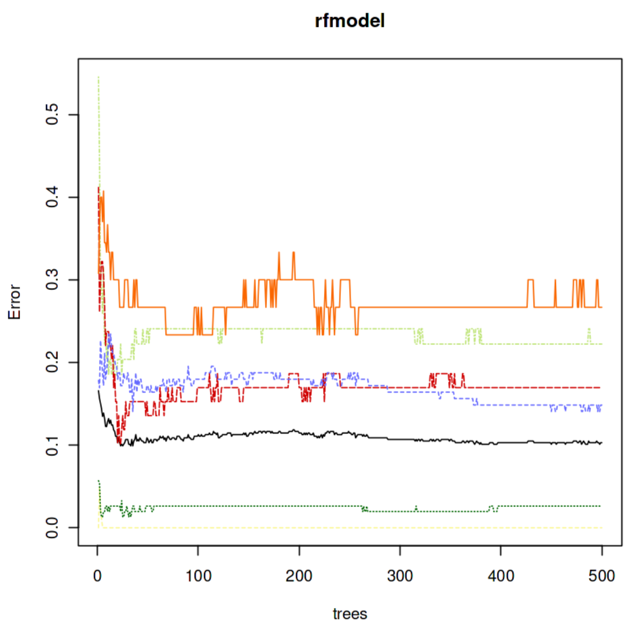


Figure 5: Random Forest model showing the error per number of trees

The support vector machine performs poorly with an error of 19%. This leads to sticking with the random forest model which performed better than the SVM to do the classification.

2.5 Ground validation

To validate the results, clear resolution satellite images in Google Earth are used, where random points shown in figure 7 are generated from the classified image and ground truthing is done.

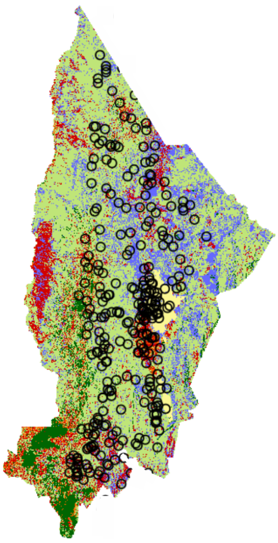
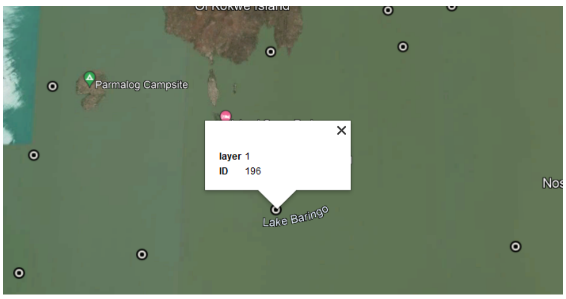
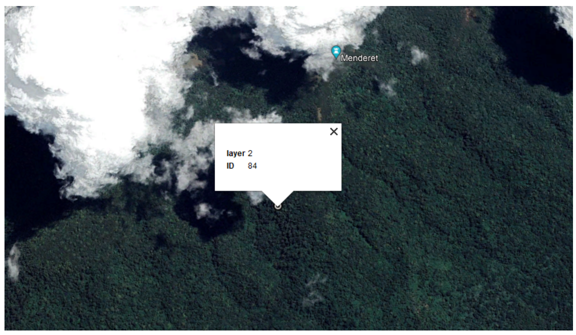


Figure 6: Random points from classified raster image of Baringo County 2022

While validating, each class is checked whether it belongs to the specific attribute it represents as seen in the below figures.



Class 1 represents wetland, as seen in the figure above the classification correctly classifies wetland area in Lake Baringo



Class 2 represents Forest and the algorithm correctly classifies it.

Class 3 represents Farmland (2015 Maxar Image)

Class 4 represents built up which correctly shows a homestead (2015 Maxar Image)

Class 5 represents grasslands

Class 6 represents other lands

Figure 7: Ground validation per each class of the supervised classification

2.6 Results

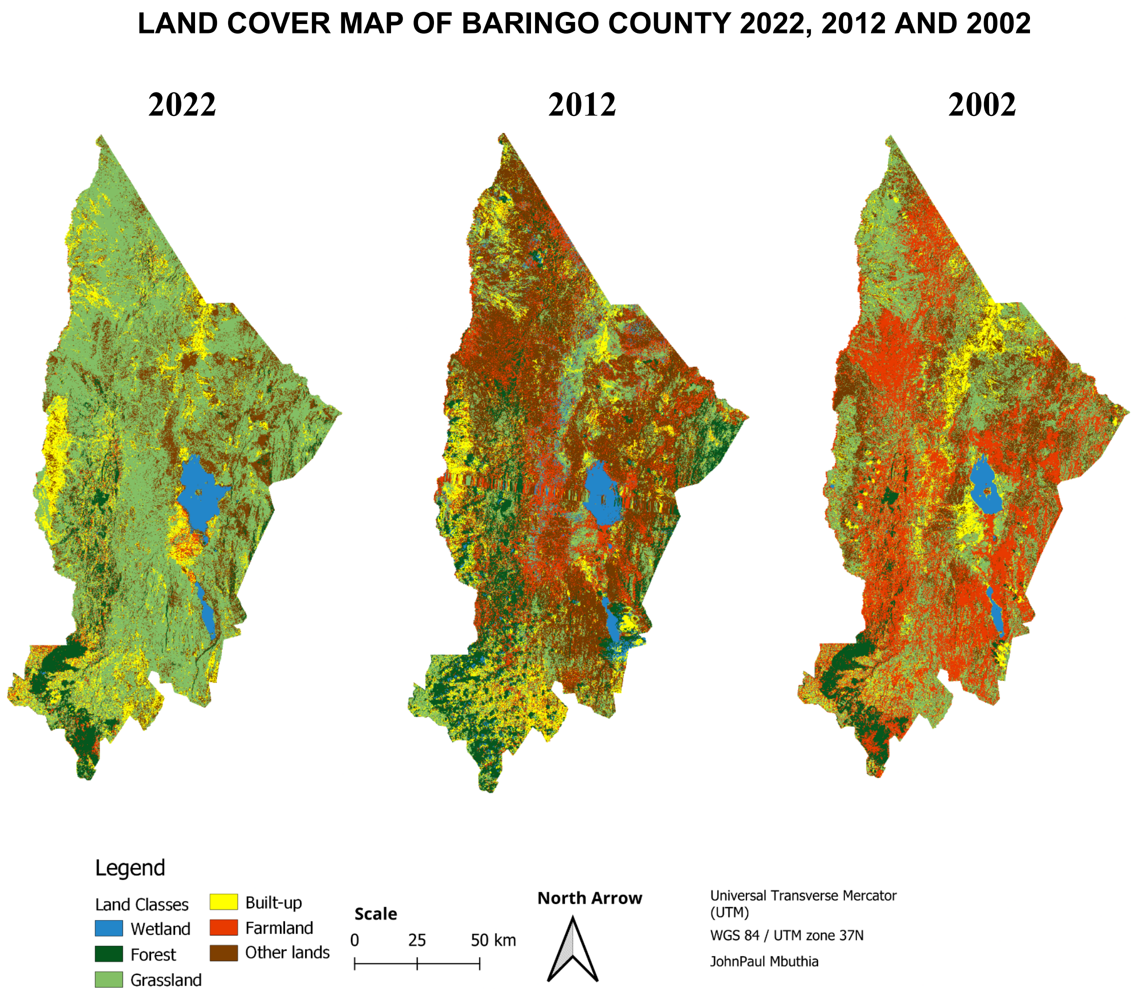


Figure 8: land cover classification map of Baringo County 2022, 2012 and 2002

Table 2: Area covered by each classified category

Land Use Type	**Area (Ha)**		
	2022	**2012**	**2002**
Wetlands	26124	49562	31638
Forest	69363	151105	162455
Farmland	656831	209543	265577
Built up	115435	109992	89512
Grasslands	23229	190001	236869
Other lands	190916	371670	295825

When the three results are compared, land cover in Baringo County has changed significantly. Wetland cover has reduced greatly while farmlands have been increasing since 2002. Built-up areas also increased from 2012.

After training the classification classes using random forest, classification is done on the target rasters. The first image to be classified was in February, 2022 as shown in figure 8. On the 2022 landsat 8 image, the random forest model had an accuracy of 88.93%. The Wetland category had the lowest class error at 0, while the grasslands category had the highest error at 0.3 as shown in the confusion matrix on figure 10. To reduce the large error in the grassland category more training samples/pixels can be corrected. Also the user accuracy has to be improved. Class 1 represents wetlands, class 2 forest, class 3 farmland, class 4 built up, class 5 grassland and class 6 other lands on the confusion matrix below.

118+989

```
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 11.07%
Confusion matrix:
   1   2  3  4  5   6 class.error
1 91   0  0  0  0   0   0.0000000
2  0 151  0  0  1   1   0.0130719
3  0   1 39  4  4   6   0.2777778
4  0   0  4 49  2   4   0.1694915
5  0   1  2  5 21   1   0.3000000
6  0   3 11  7  0 107   0.1640625
```

	Area km²
1	261.24
2	693.63
3	6568.31
4	1154.35
5	232.29
6	1909.16

Figure 9: Confusion matrix with the error rate on the left, on the right area in km^2 per classification class

From the classification results above of 2022, it's now possible to get the accurate area statistics of each land cover class in Baringo County. The above figure shows the size in kilometers squared of the area classified. Farmland had the largest land cover area, followed by Other lands while built up areas follow very closely. Wetland which contains both Lake Baringo and Lake Bogoria has an area of 26117 hectares.

The second image to be classified was in 2012 as shown in figure 8. The model had an accuracy 72.07% in 2012, the reduced accuracy may be attributed to the cloud masking done which did not mask all clouds.

In the year 2002, the random forest model had an accuracy of 94% as shown in figure 10.

```
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 1

        OOB estimate of  error rate: 6%
```

Area

```
     km²
1  316.38
2  844.55
3 2655.77
4 2462.74
5 2368.69
6 2170.63
```

Figure 10: Out of bag error rate on the left, on the right area in km^2 per classification class

2.7 Conclusion

The activities above have shown the land cover types and area coverage of Baringo County as per IPCC standards. It has also shown the changes in land cover over a 3 decade span, that is from the year 2002, 2012 and 2022. The greatest land cover change in Baringo County has been in wetlands. The government and local communities should come up with policies on how to conserve the remaining wetlands. The two main lakes in Baringo are home for many species of birds with the main birds being thousands of flamingos. Preserving the wetlands will not only provide water for the local community but also conserve the ecology of Baringo.In the next report more training samples will be collected for our target years. The aim will be 100% user accuracy in pixel identification of the land cover classes. Regression algorithms for classification will also be tried out and the differences compared.

Setup

In [1]:
```
library(tidyverse)
library(ggmap)
library(terra)
library(sp)
library(sf)
library(rgeos)
library(rpart)
library(raster)
library(randomForest)
library(e1071) # svm classification
library(ggplot2)
```

```
── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
✓ ggplot2 3.4.0      ✓ purrr   1.0.1
✓ tibble  3.1.8      ✓ dplyr   1.0.10
✓ tidyr   1.2.1      ✓ stringr 1.5.0
✓ readr   2.1.3      ✓ forcats 0.5.2
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✕ dplyr::filter() masks stats::filter()
✕ dplyr::lag()    masks stats::lag()
i Google's Terms of Service: <https://mapsplatform.google.com>

i Please cite ggmap if you use it! Use `citation("ggmap")` for details.

terra 1.6.53


Attaching package: 'terra'


The following object is masked from 'package:ggmap':

    inset


The following object is masked from 'package:tidyr':

    extract


Linking to GEOS 3.8.0, GDAL 3.0.4, PROJ 6.3.1; sf_use_s2() is TRUE

rgeos version: 0.6-1, (SVN revision 692)
 GEOS runtime version: 3.8.0-CAPI-1.13.1
 Please note that rgeos will be retired during 2023,
plan transition to sf functions using GEOS at your earliest convenience.
 Linking to sp version: 1.5-1
 Polygon checking: TRUE



Attaching package: 'raster'


The following object is masked from 'package:dplyr':

    select


randomForest 4.6-14

Type rfNews() to see new features/changes/bug fixes.


Attaching package: 'randomForest'


The following object is masked from 'package:dplyr':

    combine


The following object is masked from 'package:ggplot2':

    margin



Attaching package: 'e1071'


The following object is masked from 'package:raster':

    interpolate


The following object is masked from 'package:terra':

    interpolate
```

Data Exploration

In [9]:
```
raster <- paste0("/kaggle/input/final-images/bar_02_final.tif")
img <- brick("/kaggle/input/final-images/bar_02_final.tif")
train_samples <- st_read("/kaggle/input/train02/training sample.2002.shp")
# train_pixels <- st_read("/kaggle/input/train-pixels/training_sites.shp")
trainpix <- st_read("/kaggle/input/train02/training sample.2002.shp")
bar_county <- st_read("/kaggle/input/bar-county/baringo.shp")
```

```
Reading layer `training sample.2002' from data source
  `/kaggle/input/train02/training sample.2002.shp' using driver `ESRI Shapefile'
replacing null geometries with empty geometries
Simple feature collection with 255 features and 1 field (with 1 geometry empty)
Geometry type: LINESTRING
Dimension:     XY
Bounding box:  xmin: 122452.8 ymin: -8252.862 xmax: 206489.2 ymax: 137746.3
CRS:           NA
Reading layer `training sample.2002' from data source
  `/kaggle/input/train02/training sample.2002.shp' using driver `ESRI Shapefile'
replacing null geometries with empty geometries
Simple feature collection with 255 features and 1 field (with 1 geometry empty)
Geometry type: LINESTRING
Dimension:     XY
Bounding box:  xmin: 122452.8 ymin: -8252.862 xmax: 206489.2 ymax: 137746.3
CRS:           NA
Reading layer `baringo' from data source `/kaggle/input/bar-county/baringo.shp' using driver `ESRI Shapefile'
Simple feature collection with 1 feature and 1 field
Geometry type: POLYGON
Dimension:     XY
Bounding box:  xmin: 35.5228 ymin: -0.2263882 xmax: 36.48872 ymax: 1.663619
CRS:           NA
```

In [3]:
```
st_crs(trainpix) <- 32637
```

In [4]:
```
st_crs(bar_county) <- 32637

trainpix <- st_transform(trainpix, crs = st_crs(32637))
rast22 <- rast(raster)

tail(trainpix, 2)
```

```
Registered S3 method overwritten by 'geojsonsf':
  method        from
  print.geojson geojson
```

A sf: 2 × 2

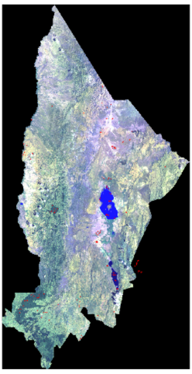
	Id	geometry
	<int>	<LINESTRING [m]>
254	6	LINESTRING (183973.7 65103....
255	6	LINESTRING (184031.9 64898....

In [5]:
```
corr_pixels <- st_make_valid(trainpix)
# sf_cent <- st_centroid(corr_pixels)

train_sts <- corr_pixels %>% filter(!st_is_empty(.))

tail(train_sts, 2)
```

A sf: 2 × 2

	Id	geometry
	<int>	<LINESTRING [m]>
253	6	LINESTRING (183973.7 65103....
254	6	LINESTRING (184031.9 64898....

In [10]:
```
plotRGB(img, r = 4, g = 3, b = 2, stretch = "lin")
plot(train_sts, col="red", add=TRUE)
```



Feature Engineering

In [11]:
```
levels(as.factor(train_sts$LULC_CODE))
```

In [12]:
```
names(img)
```

'bar_02_final_1' · 'bar_02_final_2' · 'bar_02_final_3' · 'bar_02_final_4'

In [13]:
```
names(img) <- c("b1", "b2", "b3", "b4")

names(img)
```

'b1' · 'b2' · 'b3' · 'b4'

Extract samples from raster image

In [14]:
```
smp <- extract(img, train_sts, df = TRUE)
```

In [20]:
```
tail(smp)
```

A data.frame: 6 × 5

	b1	b2	b3	b4	cl
	<dbl>	<dbl>	<dbl>	<dbl>	<fct>
1078	72	63	136	104	6
1079	72	61	142	112	6
1080	73	66	162	123	6
1081	73	64	153	117	6
1082	72	63	145	110	6
1083	73	61	131	100	6

In [19]:
```
smp$cl <- as.factor(train_sts$Id[match(smp$ID, seq(nrow(train_sts)))])
smp <- smp[-1]
```

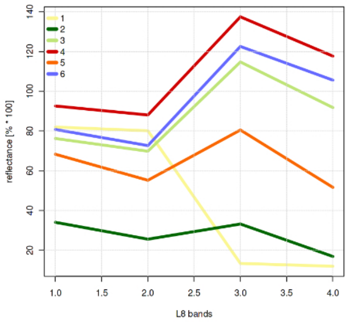
In [21]:
```
smp <- na.omit(smp)
```

In [22]:
```
which(is.na(smp$cl))
```

In [23]:
```
summary(smp$cl)
```

1: 238 **2:** 273 **3:** 156 **4:** 63 **5:** 113 **6:** 240

In [24]:
```
sp <- aggregate( . ~ cl, data = smp, FUN = mean, na.rm = TRUE )

# plot empty plot of a defined size
plot(0,
     ylim = c(min(sp[2:ncol(sp)]), max(sp[2:ncol(sp)])),
     xlim = c(1, ncol(smp)-1),
     type = 'n',
     xlab = "L8 bands",
     ylab = "reflectance [% * 100]"
     )

# define colors for class representation - one color per class necessary!
mycolors <- c("#fbf793", "#006601", "#bfe578", "#d00000", "#fa6700", "#6569ff")

# draw one line for each class
for (i in 1:nrow(sp)){
  lines(as.numeric(sp[i, -1]),
        lwd = 4,
        col = mycolors[i]
        )
  }

# add a grid
grid()

# add a legend
legend(as.character(sp$cl),
       x = "topleft",
       col = mycolors,
       lwd = 5,
       bty = "n"
       )
```



Model Building

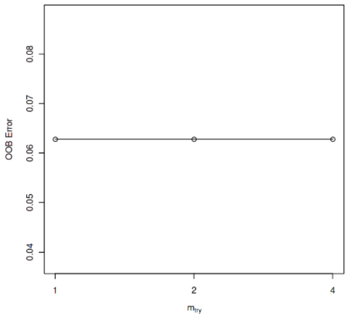
Random Forest Model

In [25]:
```
# down sampling
smp.size <- rep(min(summary(smp$cl)), nlevels(smp$cl))
smp.size
```

63 · 63 · 63 · 63 · 63 · 63

In [26]:
```
# train the model

rfmodel <- tuneRF(x = smp[-ncol(smp)],
                  y = smp$cl,
                  sampsize = smp.size,
                  strata = smp$cl,
                  ntree = 250,
                  importance = TRUE,
                  doBest = TRUE
                  )
```

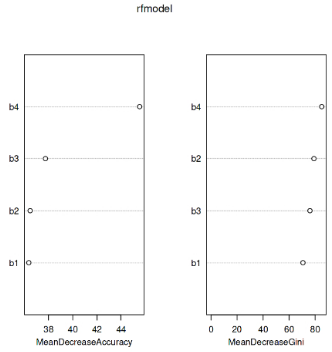
```
mtry = 2  OOB error = 6.28%
Searching left ...
mtry = 1        OOB error = 6.28%
0 0.05
Searching right ...
mtry = 4        OOB error = 6.28%
0 0.05
```



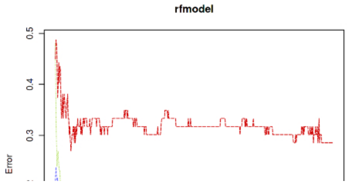
In [27]:
```
rfmodel
```

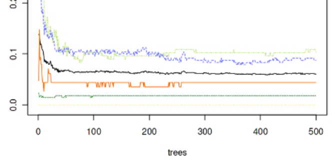
```
Call:
 randomForest(x = x, y = y, mtry = res[which.min(res[, 2]), 1],      strata = ..2, sampsize = ..1, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 1

        OOB estimate of  error rate: 6%
Confusion matrix:
    1   2   3  4   5   6 class.error
1 238   0   0  0   0   0  0.00000000
2   0 268   0  0   5   0  0.01831502
3   0   0 140  6   4   6  0.10256410
4   0   0  14 45   1   3  0.28571429
5   0   0   0  1 108   4  0.04424779
6   0   1   9  4   7 219  0.08750000
```

In [28]:
```
varImpPlot(rfmodel)
```



In [29]:
```
plot(rfmodel, col = c("#000000", "#fbf793", "#006601", "#bfe578", "#d00000", "#fa6700", "#6569ff"))
```





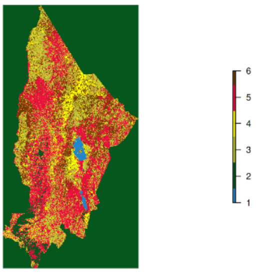
We can see a decrease in error as we increase the number of trees.

In [30]:
```
# save(rfmodel, file = "rfmodel.RData")
# load("rfmodel.RData")
```

In [31]:
```
# predict

result <- predict(img,
                  rfmodel,
                  filename = "classified02.tif",
                  overwrite = TRUE
                  )
```

In [32]:
```
writeRaster(result, 'classified2012.tif')
```

In [33]:
```
plot(result,
     axes = FALSE,
     box = FALSE,
     col = c("#2386c9", # Wetland
             "#05571d", # Forest
             "#c1c439", # Farmland
             "#ffff00", # Builtup
             "#ee0f3f", # Grassland
             "#733a00"  # Other lands
             )
     )
```



SVM Classification

In [120]:
```
head(smp)
```

A data.frame: 6 × 5

	b1	b2	b3	b4	cl
	<dbl>	<dbl>	<dbl>	<dbl>	<fct>
1	96	104	30	27	1
2	97	103	19	15	1
3	98	102	14	11	1
4	95	102	14	13	1
5	72	60	68	47	1
6	85	70	82	82	1

In [121]:
```
# shuffle to prevent spatial autocorrelation
smp <- smp[sample(nrow(smp)), ]
```

In [122]:
```
summary(smp$cl)
```

1: 105 **2:** 155 **3:** 121 **4:** 31 **5:** 21 **6:** 110

In [123]:
```
smp.maxsamplesize <- min(summary(smp$cl))
smp.maxsamplesize
```

21

In [124]:
```
smp <- smp[ave(1:(nrow(smp)), smp$cl, FUN = seq) <= smp.maxsamplesize, ]
```

In [125]:
```
summary(smp$cl)
```

1: 21 **2:** 21 **3:** 21 **4:** 21 **5:** 21 **6:** 21

In [126]:
```
gammas = 2^(-8:5)
gammas
```

0.00390625 · 0.0078125 · 0.015625 · 0.03125 · 0.0625 · 0.125 · 0.25 · 0.5 · 1 · 2 · 4 · 8 · 16 · 32

In [127]:
```
costs = 2^(-5:8)
costs
```

0.03125 · 0.0625 · 0.125 · 0.25 · 0.5 · 1 · 2 · 4 · 8 · 16 · 32 · 64 · 128 · 256

Gammas and costs are used to deteremine the best parameters for training the model

In [128]:
```
#Turn train.y to factor based on their type.
smp.y <- smp %>% mutate(across(where(is.numeric), factor))
```

In [129]:
```
# train

svmgs <- tune(svm,
              train.x = smp[-ncol(smp)],
              train.y = smp$cl,
#               train.y = smp.y$cl,
              type = "C-classification",
              kernel = "radial",
              scale = TRUE,
              ranges = list(gamma = gammas, cost = costs),
              tunecontrol = tune.control(cross = 5)
          )
```

In [130]:
```
svmgs
```

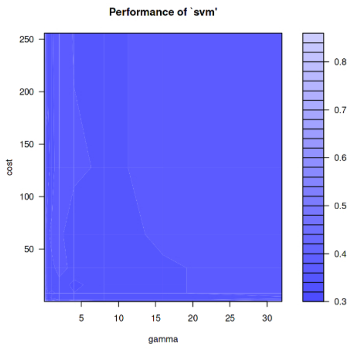
```
Parameter tuning of ‘svm’:

- sampling method: 5-fold cross validation
```

```
- best parameters:
 gamma cost
0.0625   16

- best performance: 0.3009231
```

In [131]:
```
svmmodel <- svmgs$best.model
svmmodel
```

```
Call:
best.tune(METHOD = svm, train.x = smp[-ncol(smp)], train.y = smp$cl,
    ranges = list(gamma = gammas, cost = costs), tunecontrol = tune.control(cross = 5),
    type = "C-classification", kernel = "radial", scale = TRUE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  16

Number of Support Vectors:  86
```

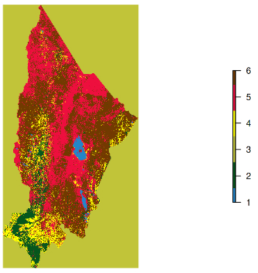
In [132]:
```
plot(svmgs)
```



In [133]:
```
# save(svmmodel, file = "svmmodel.RData")
# load("svmmodel.RData")
```

In [134]:
```
result <- predict(img,
                  svmmodel,
                  filename = "classification_svm.tif",
                  overwrite = TRUE
                  )
```

In [135]:
```
plot(result,
     axes = FALSE,
     box = FALSE,
     col = c("#2386c9", # Wetland
             "#05571d", # Forest
             "#c1c439", # Farmland
             "#ffff00", # Builtup
             "#ee0f3f", # Grassland
             "#733a00"  # Other lands
             )
     )
```



Validation

In [98]:
```
# classified random foret image
img.classified <- raster("/kaggle/input/classified02/2002class.tif")
```

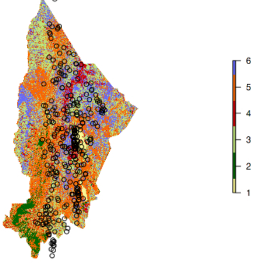
In [99]:
```
smp.test <- sampleStratified(img.classified,
                             size = 50,
                             na.rm = TRUE,
                             sp = TRUE)
```

In [100]:
```
smp.test <- smp.test[sample(nrow(smp.test)), ]
smp.test$layer
```

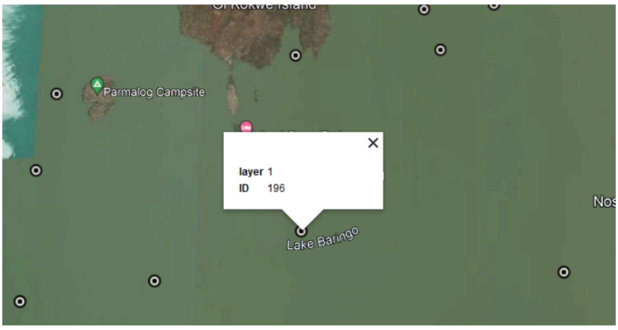
```
1· 6· 2· 2· 4· 2· 5· 1· 1· 5· 1· 5· 4· 4· 3· 2· 5· 2· 3· 3· 5· 1· 2· 4· 2· 2· 2· 2· 6· 2· 4· 3· 1·
5· 5· 5· 4· 6· 6· 4· 3· 3· 3· 6· 2· 5· 6· 1· 3· 2· 1· 4· 5· 4· 1· 1· 1· 5· 5· 1· 4· 3· 1· 3· 6· 1·
3· 4· 2· 6· 5· 6· 4· 5· 1· 5· 2· 6· 4· 3· 6· 4· 3· 3· 4· 4· 3· 6· 4· 6· 3· 6· 1· 3· 5· 3· 3· 5· 1·
3· 4· 6· 1· 6· 1· 6· 1· 6· 2· 2· 5· 3· 2· 5· 5· 2· 4· 1· 3· 2· 5· 3· 4· 6· 6· 4· 3· 4· 4· 6· 1· 6·
2· 3· 5· 3· 1· 4· 5· 6· 3· 4· 2· 4· 1· 3· 2· 4· 2· 6· 6· 5· 2· 1· 2· 5· 6· 3· 1· 6· 1· 1· 4· 4· 5·
2· 1· 6· 3· 3· 1· 5· 6· 1· 2· 6· 5· 1· 2· 5· 3· 6· 1· 5· 6· 5· 4· 5· 4· 6· 2· 4· 2· 2· 1· 3· 2· 4·
3· 2· 1· 4· 5· 1· 3· 3· 2· 2· 6· 1· 3· 5· 6· 1· 4· 4· 4· 5· 2· 1· 2· 3· 4· 5· 4· 5· 1· 6· 3· 3· 6·
3· 6· 4· 4· 2· 6· 1· 6· 4· 5· 3· 2· 2· 6· 6· 1· 4· 5· 2· 6· 5· 3· 5· 5· 5· 5· 5· 1· 1· 1· 2· 6· 1·
2· 6· 4· 3· 3· 1· 6· 2· 4· 2· 4· 1· 2· 5· 3· 5· 3· 4· 1· 3· 5· 3· 3· 5· 6· 6· 2· 4· 6· 2· 5· 6· 2·
1· 4· 4
```

In [101]:
```
# smp.test <- smp.test[, -c(1, 2)]
smp.test$ID <- 1:nrow(smp.test)
smp.test <- smp.test[c('layer', 'ID')]
# smp.test
```

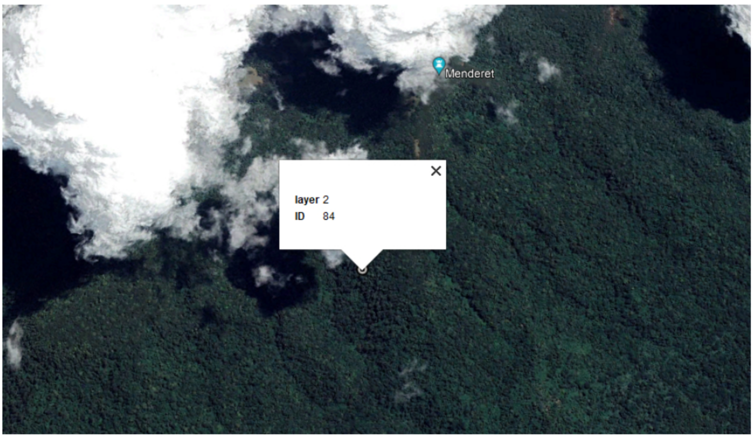
In [102]:
```
plot(img.classified,
     axes = FALSE,
     box = FALSE,
     col = c("#fbf793", "#006601", "#bfe578", "#d00000", "#fa6700", "#6569ff")
     )

points(smp.test)
```



Validate random points on Google Earth



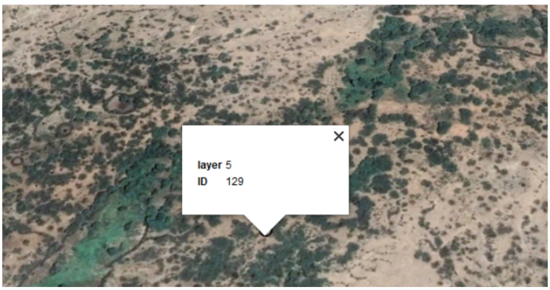
Class 1 represents wetland, as seen in the figure above the classfiication correctly classifies wetland area in Lake Baringo

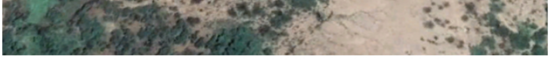


Class 2 represents Forest and the algorithm correctly classifies it.

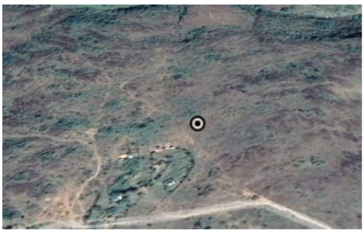
Class 3 represents Farmland (2015 Maxar Image)

Class 4 represents built up which correctly shows a homestead (2015 Maxar Image)





Class 5 represents graslands



Class 6 represents other lands

In [103]:
```
shapefile(smp.test,
          filename = "validation_RF.shp",
          overwrite = TRUE
          )
```

Accuracy Statistics

Accuracy Matrix

In [104]:
```
shp.valid <- shapefile("/kaggle/working/validation_RF.shp")
# shp.valid <-  smp.test
```

In [105]:
```
reference <- as.factor(shp.valid$layer)
```

In [106]:
```
reference
```

1 · 6 · 2 · 2 · 4 · 2 · 5 · 1 · 1 · 5 · 1 · 5 · 4 · 4 · 3 · 2 · 5 · 2 · 3 · 3 · 5 · 1 · 2 · 4 · 2 · 2 · 2 · 2 · 6 · 2 · 4 · 3 · 1 ·
5 · 5 · 5 · 4 · 6 · 6 · 4 · 3 · 3 · 3 · 6 · 2 · 5 · 6 · 1 · 3 · 2 · 1 · 4 · 5 · 4 · 1 · 1 · 1 · 5 · 5 · 1 · 4 · 3 · 1 · 3 · 6 · 1 ·
3 · 4 · 2 · 6 · 5 · 6 · 4 · 5 · 1 · 5 · 2 · 6 · 4 · 3 · 6 · 4 · 3 · 3 · 4 · 4 · 3 · 6 · 4 · 6 · 3 · 6 · 1 · 3 · 5 · 3 · 3 · 5 · 1 ·
3 · 4 · 6 · 1 · 6 · 1 · 6 · 1 · 6 · 2 · 2 · 5 · 3 · 2 · 5 · 5 · 2 · 4 · 1 · 3 · 2 · 5 · 3 · 4 · 6 · 6 · 4 · 3 · 4 · 4 · 6 · 1 · 6 ·
2 · 3 · 5 · 3 · 1 · 4 · 5 · 6 · 3 · 4 · 2 · 4 · 1 · 3 · 2 · 4 · 2 · 6 · 6 · 5 · 2 · 1 · 2 · 5 · 6 · 3 · 1 · 6 · 1 · 1 · 4 · 4 · 5 ·
2 · 1 · 6 · 3 · 3 · 1 · 5 · 6 · 1 · 2 · 6 · 5 · 1 · 2 · 5 · 3 · 6 · 1 · 5 · 6 · 5 · 4 · 5 · 4 · 6 · 2 · 4 · 2 · 2 · 1 · 3 · 2 · 4 ·
3 · 2 · 1 · 4 · 5 · 1 · 3 · 3 · 2 · 2 · 6 · 1 · 3 · 5 · 6 · 1 · 4 · 4 · 4 · 5 · 2 · 1 · 2 · 3 · 4 · 5 · 4 · 5 · 1 · 6 · 3 · 3 · 6 ·
3 · 6 · 4 · 4 · 2 · 6 · 1 · 6 · 4 · 5 · 3 · 2 · 2 · 6 · 6 · 1 · 4 · 5 · 2 · 6 · 5 · 3 · 5 · 5 · 5 · 5 · 5 · 1 · 1 · 1 · 2 · 6 · 1 ·
2 · 6 · 4 · 3 · 3 · 1 · 6 · 2 · 4 · 2 · 4 · 1 · 2 · 5 · 3 · 5 · 3 · 4 · 1 · 3 · 5 · 3 · 3 · 5 · 6 · 6 · 2 · 4 · 6 · 2 · 5 · 6 · 2 ·
1 · 4 · 4

▶ **Levels**:

In [107]:
```
predicted <- as.factor(extract(img.classified, shp.valid))
```

In [108]:
```
predicted
```

1 · 6 · 2 · 2 · 4 · 2 · 5 · 1 · 1 · 5 · 1 · 5 · 4 · 4 · 3 · 2 · 5 · 2 · 3 · 3 · 5 · 1 · 2 · 4 · 2 · 2 · 2 · 2 · 6 · 2 · 4 · 3 · 1 ·
5 · 5 · 5 · 4 · 6 · 6 · 4 · 3 · 3 · 3 · 6 · 2 · 5 · 6 · 1 · 3 · 2 · 1 · 4 · 5 · 4 · 1 · 1 · 1 · 5 · 5 · 1 · 4 · 3 · 1 · 3 · 6 · 1 ·
3 · 4 · 2 · 6 · 5 · 6 · 4 · 5 · 1 · 5 · 2 · 6 · 4 · 3 · 6 · 4 · 3 · 3 · 4 · 4 · 3 · 6 · 4 · 6 · 3 · 6 · 1 · 3 · 5 · 3 · 3 · 5 · 1 ·
3 · 4 · 6 · 1 · 6 · 1 · 6 · 1 · 6 · 2 · 2 · 5 · 3 · 2 · 5 · 5 · 2 · 4 · 1 · 3 · 2 · 5 · 3 · 4 · 6 · 6 · 4 · 3 · 4 · 4 · 6 · 1 · 6 ·
2 · 3 · 5 · 3 · 1 · 4 · 5 · 6 · 3 · 4 · 2 · 4 · 1 · 3 · 2 · 4 · 2 · 6 · 6 · 5 · 2 · 1 · 2 · 5 · 6 · 3 · 1 · 6 · 1 · 1 · 4 · 4 · 5 ·
2 · 1 · 6 · 3 · 3 · 1 · 5 · 6 · 1 · 2 · 6 · 5 · 1 · 2 · 5 · 3 · 6 · 1 · 5 · 6 · 5 · 4 · 5 · 4 · 6 · 2 · 4 · 2 · 2 · 1 · 3 · 2 · 4 ·
3 · 2 · 1 · 4 · 5 · 1 · 3 · 3 · 2 · 2 · 6 · 1 · 3 · 5 · 6 · 1 · 4 · 4 · 4 · 5 · 2 · 1 · 2 · 3 · 4 · 5 · 4 · 5 · 1 · 6 · 3 · 3 · 6 ·
3 · 6 · 4 · 4 · 2 · 6 · 1 · 6 · 4 · 5 · 3 · 2 · 2 · 6 · 6 · 1 · 4 · 5 · 2 · 6 · 5 · 3 · 5 · 5 · 5 · 5 · 5 · 1 · 1 · 1 · 2 · 6 · 1 ·
2 · 6 · 4 · 3 · 3 · 1 · 6 · 2 · 4 · 2 · 4 · 1 · 2 · 5 · 3 · 5 · 3 · 4 · 1 · 3 · 5 · 3 · 3 · 5 · 6 · 6 · 2 · 4 · 6 · 2 · 5 · 6 · 2 ·
1 · 4 · 4

▶ **Levels**:

In [109]:
```
accmat <- table("pred" = predicted, "ref" = reference)
accmat
```

```
    ref
pred  1  2  3  4  5  6
   1 50  0  0  0  0  0
   2  0 50  0  0  0  0
   3  0  0 50  0  0  0
   4  0  0  0 50  0  0
   5  0  0  0  0 50  0
   6  0  0  0  0  0 50
```

In [110]:
```
# User Accuracy
UA <- diag(accmat) / rowSums(accmat) * 100
UA
```

1: 100 **2:** 100 **3:** 100 **4:** 100 **5:** 100 **6:** 100

In [111]:
```
# Producer's Accuracy - how often real features are shown in the classification map
PA <- diag(accmat) / colSums(accmat) * 100
PA
```

1: 100 **2:** 100 **3:** 100 **4:** 100 **5:** 100 **6:** 100

In [112]:
```
# Overall Accuracy
OA <- sum(diag(accmat)) / sum(accmat) * 100
OA
```

100

In [113]:
```
accmat.ext <- addmargins(accmat)
accmat.ext <- rbind(accmat.ext, "Users" = c(PA, NA))
accmat.ext <- cbind(accmat.ext, "Producers" = c(UA, NA, OA))

colnames(accmat.ext) <- c(levels(as.factor(smp$cl)), "Sum", "PA")
rownames(accmat.ext) <- c(levels(as.factor(smp$cl)), "Sum", "UA")

accmat.ext <- round(accmat.ext, digits = 1)
dimnames(accmat.ext) <- list("Prediction" = colnames(accmat.ext),
                             "Reference" = rownames(accmat.ext))
class(accmat.ext) <- "table"
accmat.ext
```

```
          Reference
Prediction   1   2   3   4   5   6 Sum  UA
       1    50   0   0   0   0   0  50 100
       2     0  50   0   0   0   0  50 100
       3     0   0  50   0   0   0  50 100
       4     0   0   0  50   0   0  50 100
       5     0   0   0   0  50   0  50 100
       6     0   0   0   0   0  50  50 100
       Sum  50  50  50  50  50  50 300
       PA  100 100 100 100 100 100     100
```

Significance Test

A binomial test is used

In [114]:
```
sign <- binom.test(x = sum(diag(accmat)),
                   n = sum(accmat),
                   alternative = c("two.sided"),
                   conf.level = 0.95
```

```
       )

pvalue <- sign$p.value
pvalue
```

9.81818693059561e-91

In [115]:
```
CI95 <- sign$conf.int[1:2]
CI95
```

0.987779025305706 · 1

Area Adjusted Accuracy Assessment

In [116]:
```
length(shp.valid$layer)
```

300

In [117]:
```
# clipped by mask on qgis to prevent excess values on farmland, class 3
img.classified <- brick("/kaggle/input/masked/classifiedf_3_clear.tif")
```

In [118]:
```
# create regular accuracy matrix
confmat <- table(as.factor(extract(img.classified, shp.valid)), as.factor(shp.valid$layer))
```

In [119]:
```
# get number of pixels per class and convert in km²
imgVal <- getValues(img.classified)
```

In [120]:
```
sum(is.na(imgVal))
```

12991314

In [121]:
```
length(imgVal)
```

25012151

In [122]:
```
imgVal <- na.omit(imgVal)
```

In [123]:
```
print(sum(is.na(imgVal)))
print(length(imgVal))
```

```
[1] 0
[1] 12020837
```

In [124]:
```
nclass <- length(unique(smp$cl))
maparea <- sapply(1:nclass, function(x) sum(imgVal == x))
maparea <- maparea * res(img.classified)[1] ^ 2 / 1000000
```

In [125]:
```
sum(maparea)
```

10818.7533

In [126]:
```
# set confidence interval
conf <- 1.96

# total  map area
A <- sum(maparea)
```

In [127]:
```
A
```

10818.7533

Baringo county according to IEBC has 11,015 square kilometers

In [128]:
```
# proportion of area mapped as class i
W_i <- maparea / A
```

In [129]:
```
# number of reference points per class
n_i <- rowSums(confmat)
# population error matrix (Eq.4)
p <- W_i * confmat / n_i
p[is.na(p)] <- 0
```

In [130]:
```
# area estimation
p_area <- colSums(p) * A
# area estimation confidence interval (Eq.10)
p_area_CI <- conf * A * sqrt(colSums((W_i * p - p ^ 2) / (n_i - 1)))
```

In [131]:
```
# overall accuracy (Eq.1)
OA <- sum(diag(p))
# producers accuracy (Eq.2)
PA <- diag(p) / colSums(p)
# users accuracy (Eq.3)
UA <- diag(p) / rowSums(p)
```

In [132]:
```
print(OA)
print(PA)
print(UA)
```

```
[1] 0.4211413
        1         2         3         4         5         6 
1.0000000 0.9499969 0.1403461 0.1692216 0.3513092 0.8365815 
        1         2         3         4         5         6 
0.9433962 0.9423077 0.3181818 0.2058824 0.5000000 0.3805310 
```

In [133]:
```
# overall accuracy confidence interval (Eq.5)
OA_CI <- conf * sqrt(sum(W_i ^ 2 * UA * (1 - UA) / (n_i - 1)))
# user accuracy confidence interval (Eq.6)
UA_CI <- conf * sqrt(UA * (1 - UA) / (n_i - 1))
# producer accuracy confidence interval (Eq.7)
N_j <- sapply(1:nclass, function(x) sum(maparea / n_i * confmat[ , x]) )
tmp <- sapply(1:nclass, function(x) sum(maparea[-x] ^ 2 * confmat[-x, x] / n_i[-x] * ( 1 - confmat[-x, x] / n_i[-x]) / (n_i[-x]
- 1)) )
PA_CI <- conf * sqrt(1 / N_j ^ 2 * (maparea ^ 2 * ( 1 - PA ) ^ 2 * UA * (1 - UA) / (n_i - 1) + PA ^ 2 * tmp))
```

In [134]:
```
# gather results
result <- matrix(c(p_area, p_area_CI, PA * 100, PA_CI * 100, UA * 100, UA_CI * 100, c(OA * 100, rep(NA, nclass-1)), c(OA_CI * 10
0, rep(NA, nclass-1))), nrow = nclass)
result <- round(result, digits = 2)
rownames(result) <- levels(as.factor(smp$cl))
colnames(result) <- c("km²", "km²±", "PA", "PA±", "UA", "UA±", "OA", "OA±")
class(result) <- "table"
result
```

	km²	km²±	PA	PA±	UA	UA±	OA	OA±
1	316.38	21.06	100.00	0.00	94.34	6.28	42.11	6.04
2	844.55	99.10	95.00	9.32	94.23	6.40		
3	2655.77	593.97	14.03	8.09	31.82	19.92		
4	2462.74	590.67	16.92	10.08	20.59	13.80		
5	2368.69	567.00	35.13	11.27	50.00	19.60		
6	2170.63	503.08	83.66	10.62	38.05	8.99		

The above table shows the size in Km2 of the area classified.
The results show that:-

- Wetland class(1) has an area of 261.17 Km2
- Forest class(2) has an area of 708.89 Km2
- Farmland class(3) has an area of 6529.87 Km2
- Builtup class(4) has an area of 1177.71 Km2
- Grassland class(5) has an area of 229.58 Km2
- Other lands class(6) has an area of 1911.78 Km2

Farmland has the largest land cover mass, followed by Other lands while builtup areas follow very closely. Wetland which contains both Lake Baringo and Lake Bogoria has an area of 261.17 Km2

```
In [ ]:
```