Categorical Change Detection and Machine Learning using R

Ivan Innocent Sekibenga

2026-02-13

1 Introduction

Categorical change detection is a remote sensing technique used to identify and classify changes in land cover between two or more dates of satellite imagery. By combining change detection with machine learning in R, we can automate the classification of change types (e.g., urbanization, agricultural conversion) with greater accuracy.

This tutorial uses two Landsat images of California's Central Valley captured a decade apart:

- **Time 1 (2001)**: Landsat 7 ETM+ (`centralvalley-2001LE7.tif`)
- **Time 2 (2011)**: Landsat 5 TM (`centralvalley-2011LT5.tif`)

Both images have 6 spectral bands: B1 (Blue), B2 (Green), B3 (Red), B4 (NIR), B5 (SWIR1), and B7 (SWIR2).

Training polygons from `lcsamples.rds` provide labeled land cover classes: **cropland**, **fallow**, **built**, **open**, and **water**.

2 Load Required Packages

```
library(terra)       # Modern raster data handling
library(sf)          # Vector/spatial data
library(tidyverse)   # Data wrangling and visualization
library(tidymodels)  # Machine learning workflows
library(kableExtra)  # Table formatting
```

```
## systemfonts and textshaping have been compiled with different versions of Freetype. Because of this, textshaping
will not use the font cache provided by systemfonts
```

3 Load and Explore Raster Data

3.1 Load Multi-Temporal Imagery

```
# Load raster images for two time periods
raster_t1 <- rast("data/rs/centralvalley-2001LE7.tif")
raster_t2 <- rast("data/rs/centralvalley-2011LT5.tif")

# Inspect properties
raster_t1
```

```
## class       : SpatRaster
## size        : 1230, 1877, 6  (nrow, ncol, nlyr)
## resolution  : 0.0002694946, 0.0002694946  (x, y)
## extent      : -121.9258, -121.42, 37.85402, 38.1855  (xmin, xmax, ymin, ymax)
## coord. ref. : lon/lat WGS 84 (EPSG:4326)
## source      : centralvalley-2001LE7.tif
## names       : B1, B2, B3, B4, B5, B7
```

```
raster_t2
```

```
## class       : SpatRaster
## size        : 1230, 1877, 6  (nrow, ncol, nlyr)
## resolution  : 0.0002694946, 0.0002694946  (x, y)
## extent      : -121.9258, -121.42, 37.85402, 38.1855  (xmin, xmax, ymin, ymax)
## coord. ref. : lon/lat WGS 84 (EPSG:4326)
## source      : centralvalley-2011LT5.tif
## names       : B1, B2, B3, B4, B5, B7
```

```
# Check that both rasters have compatible geometry
compareGeom(raster_t1, raster_t2)
```

```
## [1] TRUE
```

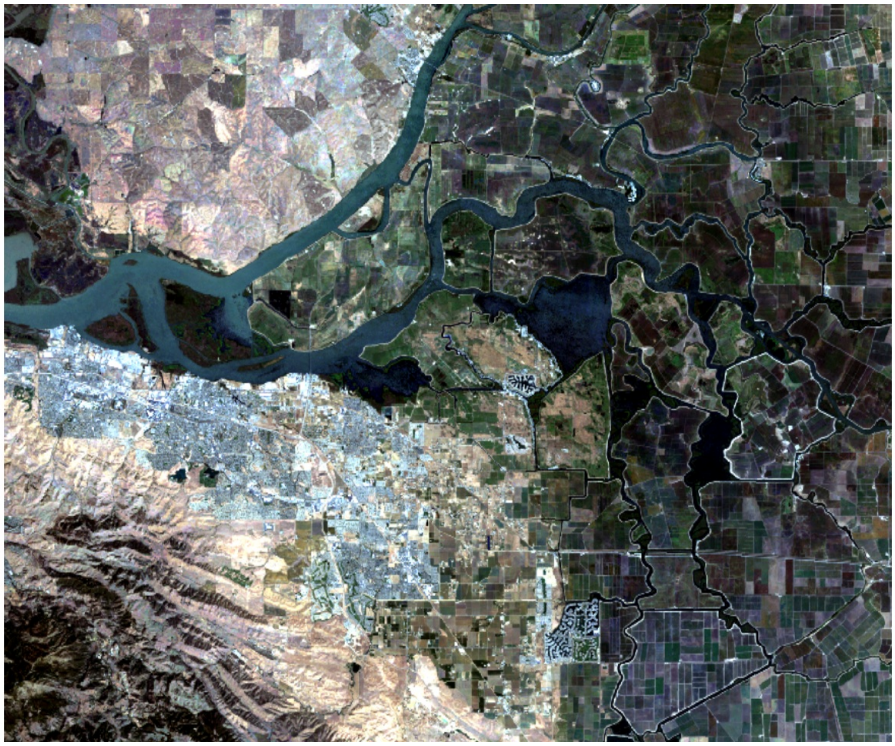
Both rasters share the same extent, resolution, and CRS (WGS 84), which is essential for pixel-by-pixel change detection.

3.2 Visualize the Imagery

```
# True color composite (R=B3, G=B2, B=B1) for Time 1
plotRGB(raster_t1, r = 3, g = 2, b = 1, stretch = "lin",
        main = "Central Valley 2001 (Landsat 7 ETM+)")
```

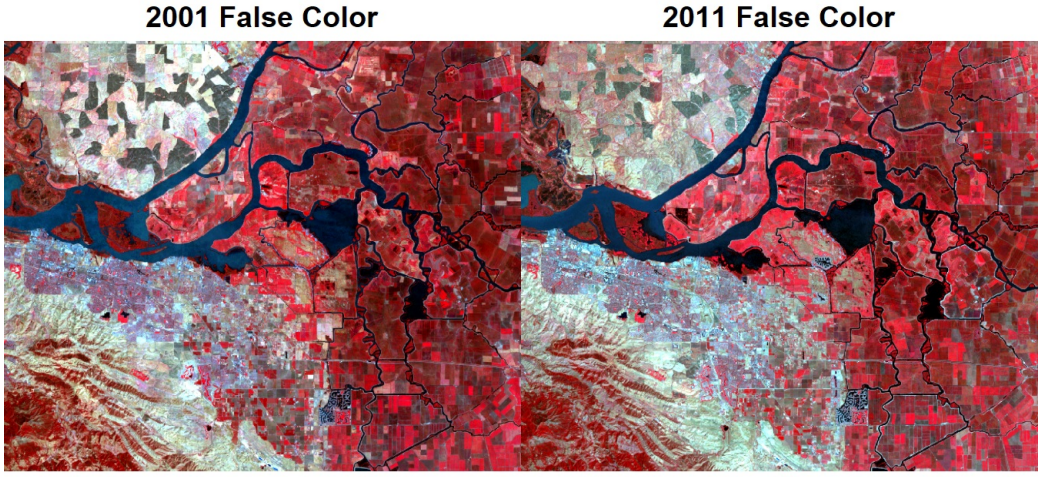
True color composite - 2001

```
# True color composite for Time 2
plotRGB(raster_t2, r = 3, g = 2, b = 1, stretch = "lin",
        main = "Central Valley 2011 (Landsat 5 TM)")
```



True color composite - 2011

```
# False color composites (NIR, Red, Green) to highlight vegetation
par(mfrow = c(1, 2))
plotRGB(raster_t1, r = 4, g = 3, b = 2, stretch = "lin",
        main = "2001 False Color")
plotRGB(raster_t2, r = 4, g = 3, b = 2, stretch = "lin",
        main = "2011 False Color")
```



False color composites highlighting vegetation

```
par(mfrow = c(1, 1))
```

4 Load and Prepare Training Data

4.1 Load Training Samples

The training data consists of 49 labeled polygons covering 5 land cover classes.

```
# Load training polygons (SpatVector)
training_polys <- readRDS("data/rs/lcsamples.rds")

# Preview the class distribution
training_df <- as.data.frame(training_polys)
training_df |> count(class) |>
  kable(caption = "Training Sample Distribution by Class") |>
  kable_styling(bootstrap_options = c("striped", "hover"), full_width = FALSE)
```

Training Sample Distribution by Class

class	n
built	14
cropland	12
fallow	11
open	9
water	3

4.2 Extract Spectral Values

We extract the pixel values from each raster at the training polygon locations. Since the training polygons are in UTM Zone 10 and the rasters are in WGS 84, we need to reproject the polygons first.

```
# Reproject training polygons to match raster CRS
training_reproj <- project(training_polys, crs(raster_t1))

# Extract spectral values for Time 1
extract_t1 <- terra::extract(raster_t1, training_reproj, df = TRUE)
extract_t1 <- extract_t1 |>
  left_join(
    tibble(ID = seq_len(nrow(training_polys)),
           class = training_polys$class),
    by = "ID"
  ) |>
  mutate(class = as.factor(class)) |>
  select(class, B1:B7) |>
  drop_na()

head(extract_t1)
```

```
##      class        B1        B2        B3        B4        B5        B7
## 1 cropland 0.1285658 0.1305083 0.1490986 0.2157305 0.2794398 0.2177061
## 2 cropland 0.1285658 0.1305083 0.1490986 0.2157305 0.2794398 0.2177061
## 3 cropland 0.1231868 0.1237071 0.1421871 0.2028163 0.2658510 0.1973636
## 4 cropland 0.1219357 0.1216776 0.1379288 0.1967401 0.2551887 0.1896176
## 5 cropland 0.1205397 0.1228051 0.1364371 0.1930361 0.2506213 0.1855069
## 6 cropland 0.1205397 0.1228051 0.1364371 0.1930361 0.2506213 0.1855069
```

```
# Extract spectral values for Time 2
extract_t2 <- terra::extract(raster_t2, training_reproj, df = TRUE)
extract_t2 <- extract_t2 |>
  left_join(
    tibble(ID = seq_len(nrow(training_polys)),
           class = training_polys$class),
    by = "ID"
  ) |>
  mutate(class = as.factor(class)) |>
  select(class, B1:B7) |>
  drop_na()

head(extract_t2)
```

```
##      class        B1        B2        B3        B4        B5        B7
## 1 cropland 0.1238642 0.1265728 0.1379537 0.2374454 0.2616852 0.2135262
## 2 cropland 0.1238642 0.1265728 0.1379537 0.2374454 0.2616852 0.2135262
## 3 cropland 0.1232517 0.1240913 0.1375854 0.2328425 0.2503225 0.2060709
## 4 cropland 0.1247514 0.1288536 0.1458697 0.2234812 0.2533683 0.2178184
## 5 cropland 0.1287331 0.1275539 0.1429229 0.2155746 0.2551467 0.2193725
## 6 cropland 0.1287331 0.1275539 0.1429229 0.2155746 0.2551467 0.2193725
```

```
cat("Time 1 extracted pixels:", nrow(extract_t1), "\n")
```

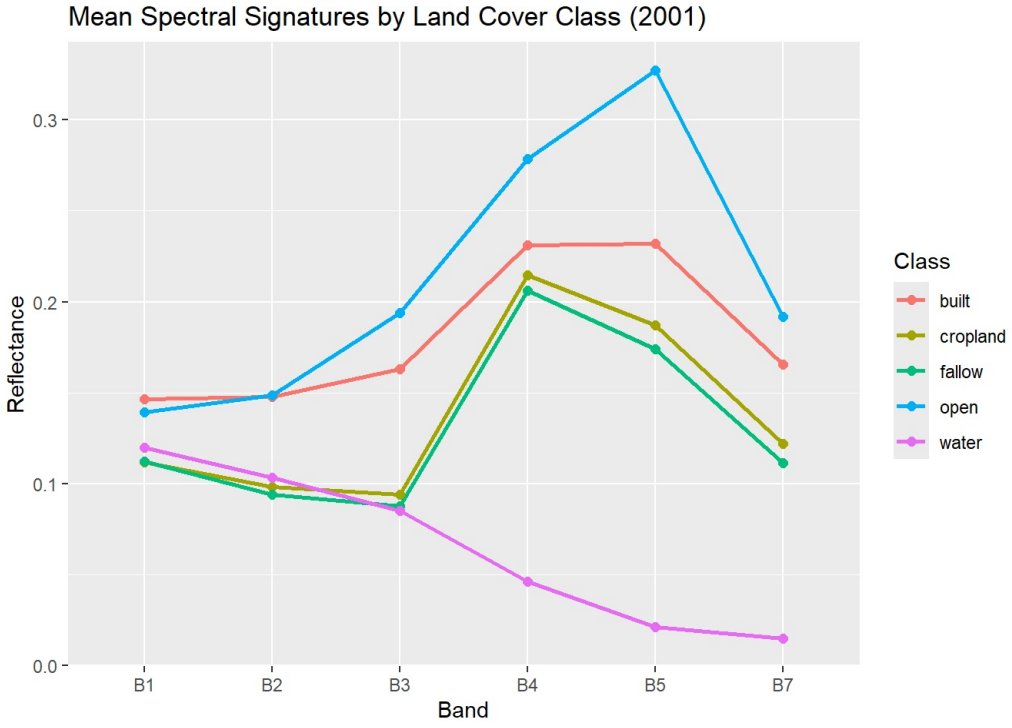
```
## Time 1 extracted pixels: 68831
```

```
cat("Time 2 extracted pixels:", nrow(extract_t2), "\n")
```

```
## Time 2 extracted pixels: 68831
```

4.3 Explore Spectral Signatures

```
# Compute mean spectral values per class for Time 1
spectral_means <- extract_t1 |>
  group_by(class) |>
  summarise(across(B1:B7, mean)) |>
  pivot_longer(cols = B1:B7, names_to = "band", values_to = "reflectance")

ggplot(spectral_means, aes(x = band, y = reflectance, color = class, group = class)) +
  geom_line(linewidth = 1) +
  geom_point(size = 2) +
  labs(title = "Mean Spectral Signatures by Land Cover Class (2001)",
       x = "Band", y = "Reflectance", color = "Class")
```



5 Land Cover Classification with Random Forest

5.1 Split Training Data

```
set.seed(6142)

# Split Time 1 training data
split_t1 <- initial_split(extract_t1, prop = 0.7, strata = class)
train_t1 <- training(split_t1)
test_t1 <- testing(split_t1)

cat("Time 1 - Training pixels:", nrow(train_t1), "\n")
```

```
## Time 1 - Training pixels: 48180
```

```
cat("Time 1 - Testing pixels:", nrow(test_t1), "\n")
```

```
## Time 1 - Testing pixels: 20651
```

```
train_t1 |> count(class)
```

```
##      class     n
## 1    built  3614
## 2 cropland  4940
## 3   fallow  6898
## 4     open 13008
## 5    water 19720
```

5.2 Define and Fit Random Forest Model

Random Forest is widely used in remote sensing classification due to its robustness with high-dimensional spectral data and resistance to overfitting.

```
# Define Random Forest model
rf_model <- rand_forest(trees = 500) |>
  set_engine("randomForest") |>
  set_mode("classification")

# Define preprocessing recipe
rf_recipe <- recipe(class ~ ., data = train_t1) |>
  step_normalize(all_predictors())

# Create and fit the workflow
rf_workflow <- workflow() |>
  add_recipe(rf_recipe) |>
  add_model(rf_model) |>
  fit(train_t1)

rf_workflow
```

```
## ══ Workflow [trained] ══════════════════════════════════════════════════════
## Preprocessor: Recipe
## Model: rand_forest()
##
## ── Preprocessor ────────────────────────────────────────────────────────────
## 1 Recipe Step
##
## • step_normalize()
##
## ── Model ───────────────────────────────────────────────────────────────────
##
## Call:
##  randomForest(x = maybe_data_frame(x), y = y, ntree = ~500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 1.93%
## Confusion matrix:
##          built cropland fallow  open water class.error
## built     3235       44     26   309     0 0.104869950
## cropland    21     4608    310     1     0 0.067206478
## fallow       8      172   6718     0     0 0.026094520
## open        32        6      3 12967     0 0.003151907
## water        0        0      0     0 19720 0.000000000
```

5.3 Accuracy Assessment

```
# Predict on the held-out test set
test_predictions <- augment(rf_workflow, test_t1)

# Confusion matrix
conf_mat(test_predictions, truth = class, estimate = .pred_class)
```

```
##           Truth
## Prediction built cropland fallow open water
##   built     1376       13      6   22     0
##   cropland    30     1983     60    0     0
##   fallow      19      129   2854    2     0
##   open       129        1      0 5579     0
##   water        0        0      0    0  8448
```

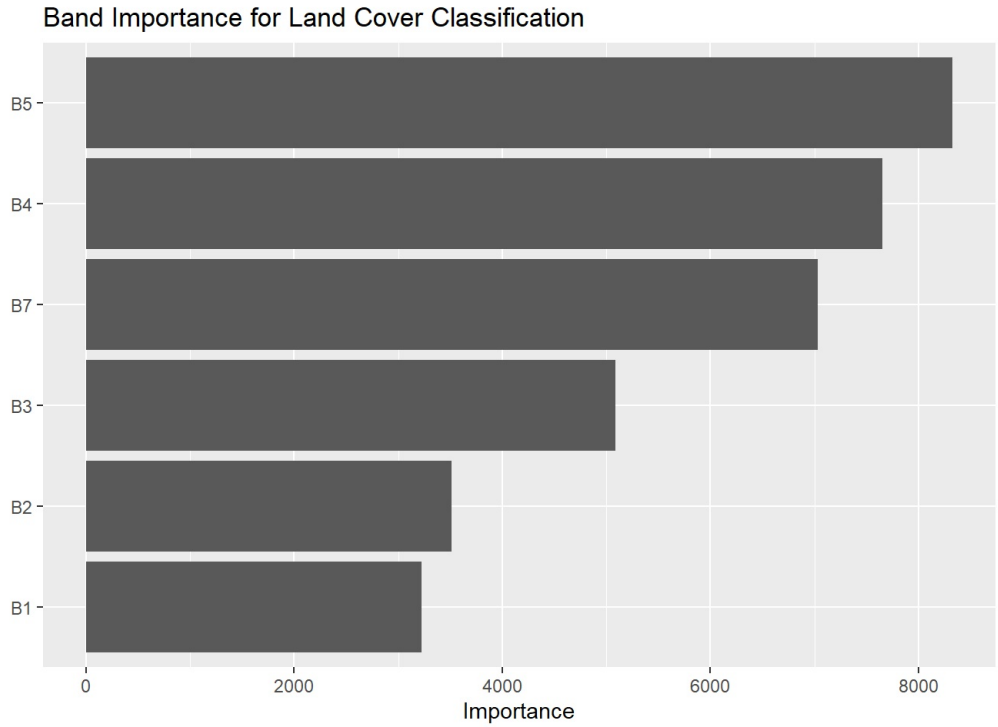
```
# Overall accuracy and Kappa
test_predictions |>
  metrics(truth = class, estimate = .pred_class) |>
  kable(caption = "Classification Accuracy Metrics") |>
  kable_styling(bootstrap_options = c("striped", "hover"), full_width = FALSE)
```

Classification Accuracy Metrics

.metric	.estimator	.estimate
accuracy	multiclass	0.9800978
kap	multiclass	0.9724185

5.4 Variable Importance

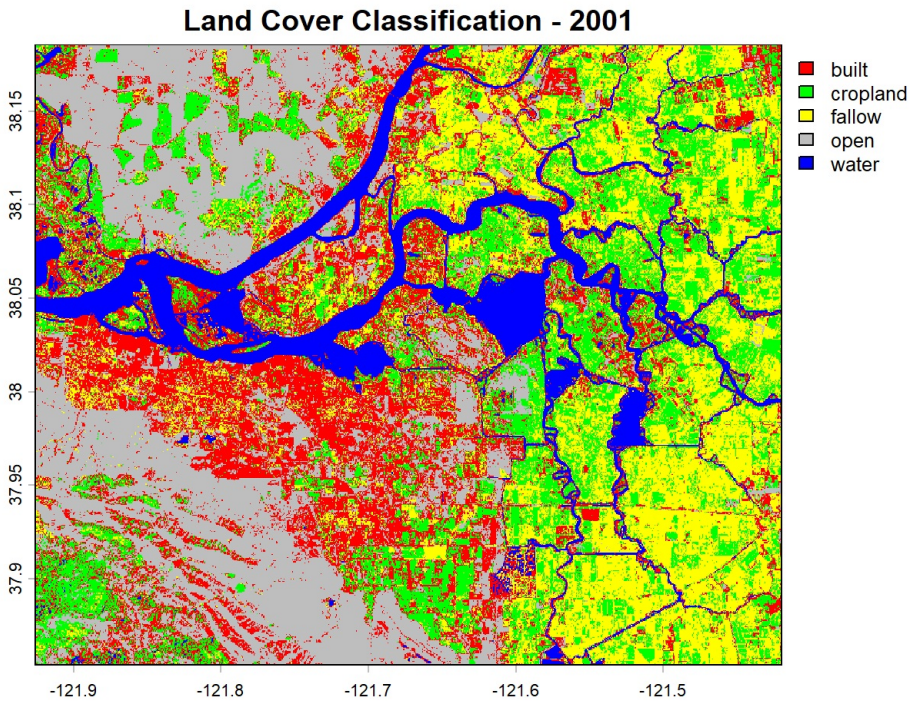
```
library(vip)

rf_workflow |>
  extract_fit_parsnip() |>
  vip(num_features = 6) +
  labs(title = "Band Importance for Land Cover Classification")
```



6 Classify Raster Images

6.1 Classify Time 1 (2001)

```
# Predict land cover for the entire 2001 image
classified_t1 <- terra::predict(raster_t1, rf_workflow, na.rm = TRUE)

# Define class colors
class_colors <- c("built" = "red", "cropland" = "green",
                  "fallow" = "yellow", "open" = "grey", "water" = "blue")

plot(classified_t1, main = "Land Cover Classification - 2001",
     col = class_colors, type = "classes")
```



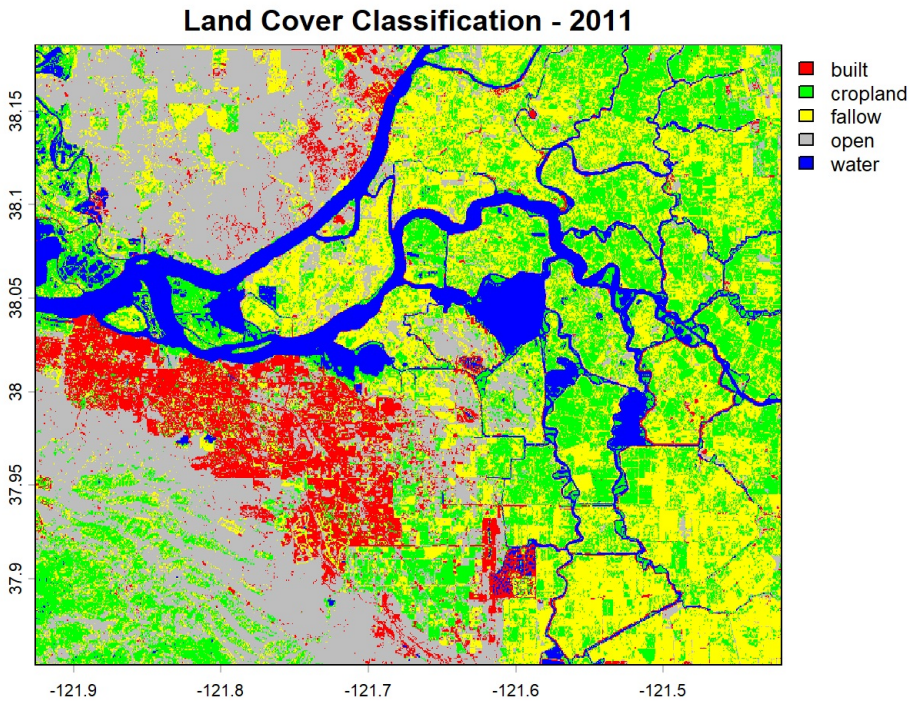
6.2 Train and Classify Time 2 (2011)

We train a separate model for Time 2 because the spectral characteristics may differ between sensors and dates.

```
set.seed(8371)

# Split Time 2 training data
split_t2 <- initial_split(extract_t2, prop = 0.7, strata = class)
train_t2 <- training(split_t2)
test_t2 <- testing(split_t2)

# Fit Random Forest for Time 2
rf_workflow_t2 <- workflow() |>
  add_recipe(recipe(class ~ ., data = train_t2) |>
               step_normalize(all_predictors())) |>
  add_model(rf_model) |>
  fit(train_t2)

# Accuracy for Time 2
test_pred_t2 <- augment(rf_workflow_t2, test_t2)
conf_mat(test_pred_t2, truth = class, estimate = .pred_class)
```

```
##           Truth
## Prediction built cropland fallow open water
##   built     1424        2      0   15     0
##   cropland     0     1951     81    8     0
##   fallow       0      154   2872   16     0
##   open       110        2      5 5547     0
##   water        0        0      0    0  8463
```

```
# Predict land cover for the entire 2011 image
classified_t2 <- terra::predict(raster_t2, rf_workflow_t2, na.rm = TRUE)

plot(classified_t2, main = "Land Cover Classification - 2011",
     col = class_colors, type = "classes")
```

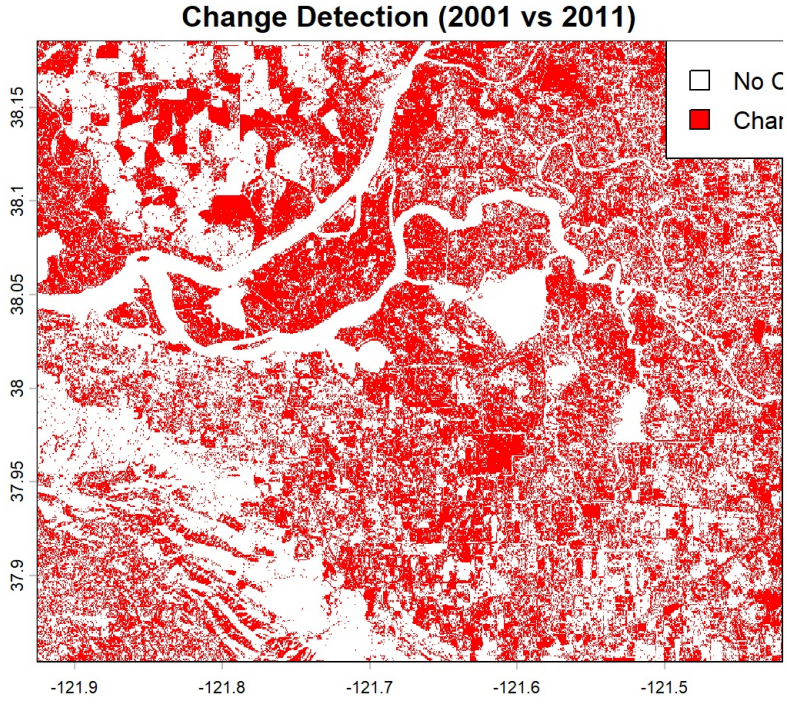


7 Post-Classification Change Detection

Post-classification change detection compares two independently classified maps to identify pixels where the land cover class has changed between the two dates.

7.1 Generate Change Map

```
# Identify changed pixels: TRUE where class differs between dates
change_binary <- classified_t1 != classified_t2

plot(change_binary, main = "Change Detection (2001 vs 2011)",
     col = c("white", "red"), legend = FALSE)
legend("topright", legend = c("No Change", "Change"),
       fill = c("white", "red"))
```



7.2 Build Transition Matrix

The transition matrix (cross-tabulation) shows how many pixels transitioned from each class in 2001 to each class in 2011.

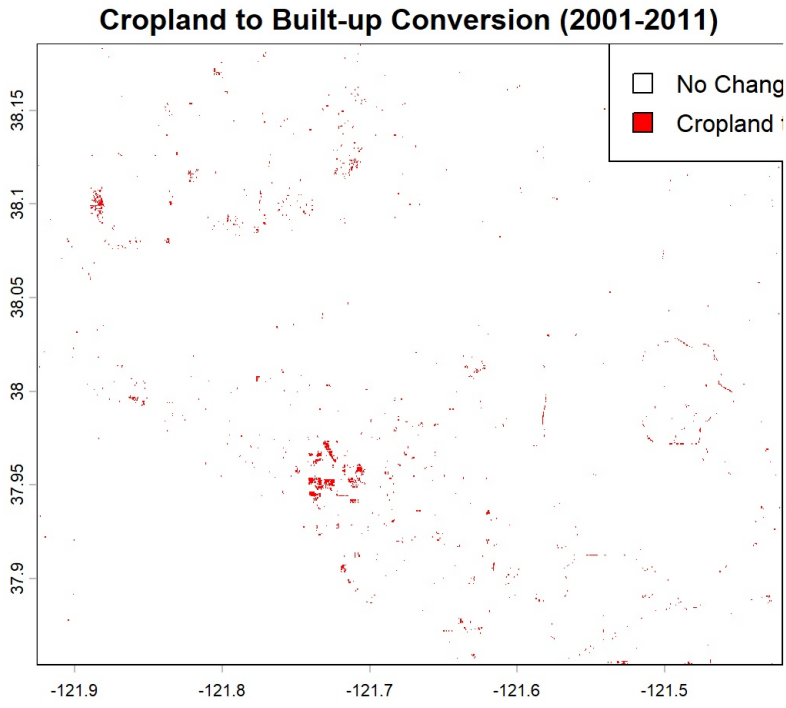
```
# Extract classified values
vals_t1 <- values(classified_t1) |> as.vector()
vals_t2 <- values(classified_t2) |> as.vector()

# Remove NA pairs
valid <- !is.na(vals_t1) & !is.na(vals_t2)

transition_table <- table(
  "From (2001)" = vals_t1[valid],
  "To (2011)" = vals_t2[valid]
)

transition_table |>
  as.data.frame.matrix() |>
  kable(caption = "Land Cover Transition Matrix (pixel counts): 2001 to 2011") |>
  kable_styling(bootstrap_options = c("striped", "hover"))
```

Land Cover Transition Matrix (pixel counts): 2001 to 2011

1	2	3	4	5
130960	134413	133875	95252	11599
9018	199669	197519	57650	5182
20278	175420	286987	34775	1891
44652	30925	57532	445557	1573
369	10728	585	150	222149

7.3 Identify Specific Transitions

```
# Example: Identify pixels that changed from cropland to built (urbanization)
cropland_to_built <- (classified_t1 == "cropland") & (classified_t2 == "built")

plot(cropland_to_built, main = "Cropland to Built-up Conversion (2001-2011)",
     col = c("white", "red"), legend = FALSE)
legend("topright", legend = c("No Change", "Cropland to Built"),
       fill = c("white", "red"))
```



7.4 Quantify Changes

```
# Compute area of each transition (in number of pixels)
transition_df <- as.data.frame(transition_table)
names(transition_df) <- c("From_2001", "To_2011", "Pixels")

# Show only transitions where change occurred
transition_df |>
  filter(From_2001 != To_2011, Pixels > 0) |>
  arrange(desc(Pixels)) |>
  kable(caption = "Land Cover Transitions (Changed Pixels Only)") |>
  kable_styling(bootstrap_options = c("striped", "hover"), full_width = FALSE)
```

Land Cover Transitions (Changed Pixels Only)

From_2001	To_2011	Pixels
2	3	197519
3	2	175420
1	2	134413
1	3	133875
1	4	95252
2	4	57650
4	3	57532
4	1	44652
3	4	34775
4	2	30925
3	1	20278
1	5	11599
5	2	10728
2	1	9018
2	5	5182
3	5	1891
4	5	1573
5	3	585
5	1	369
5	4	150

8 Summary

This tutorial demonstrated a complete categorical change detection workflow for the Central Valley, California using Landsat imagery from 2001 and 2011:

1. **Data loading** — Multi-temporal Landsat imagery and labeled training polygons
2. **Spectral exploration** — Visualized band signatures across 5 land cover classes
3. **Classification** — Random Forest via tidymodels for both time periods
4. **Accuracy assessment** — Confusion matrices and overall accuracy metrics
5. **Change detection** — Post-classification comparison to map land cover change
6. **Transition analysis** — Quantified pixel-level from-to transitions

Key considerations:

- **Sensor differences**: Landsat 7 ETM+ (2001) and Landsat 5 TM (2011) have similar but not identical spectral characteristics. Training separate models per date accounts for this.
- **Training sample quality**: With only 49 polygons across 5 classes, classification accuracy is dependent on sample representativeness. The water class (3 polygons) is particularly underrepresented.
- **Error propagation**: Post-classification change detection compounds errors from both independent classifications, so high per-class accuracy is

important for reliable change maps.