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
```

Interpretation: The output displays key properties of both rasters including dimensions (rows, columns, cells), resolution, extent (geographic boundaries), coordinate reference system (CRS), and the number of bands. Both images should show 6 spectral bands and identical spatial dimensions, confirming they cover the same geographic area. The WGS 84 (EPSG:4326) coordinate system indicates the data uses latitude/longitude coordinates.

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

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

Interpretation: A `TRUE` result confirms that both rasters have identical geometry (extent, resolution, CRS, and number of rows/columns). This is essential for pixel-by-pixel change detection analysis, as it ensures that each pixel in the 2001 image corresponds exactly to the same geographic location in the 2011 image.

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

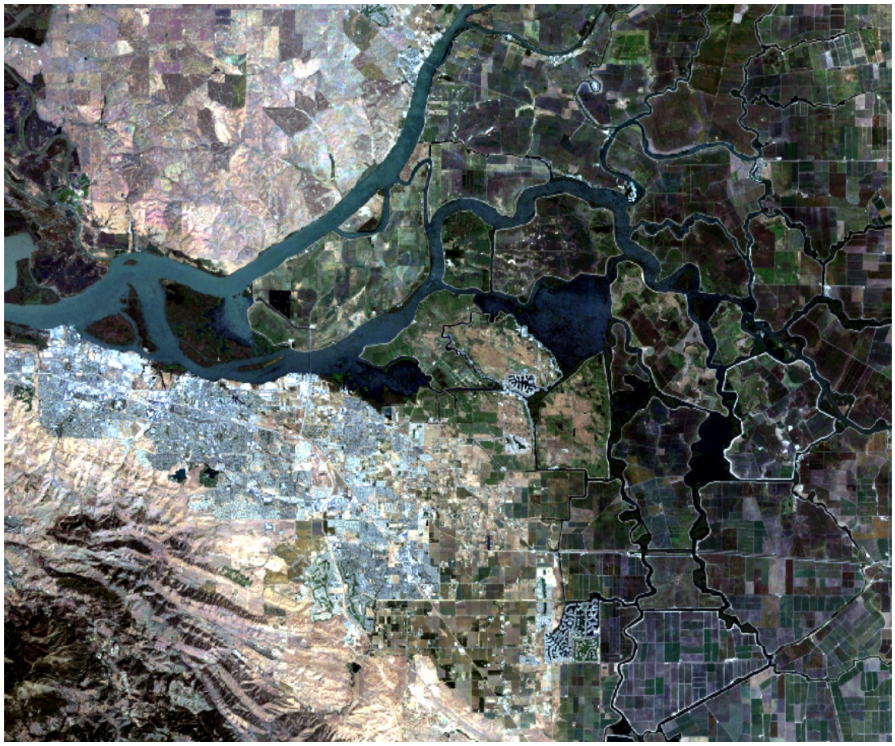
3.2 Visualize the Imagery

```
# 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

Interpretation: This true color composite displays the 2001 image as it would appear to the human eye, with red (B3), green (B2), and blue (B1) bands assigned to their natural colors. Vegetation typically appears in shades of green to brown, water in dark blue to black, built-up areas in grey to white, and bare soil in light brown. The linear stretch enhances contrast by spreading the pixel values across the full color range.

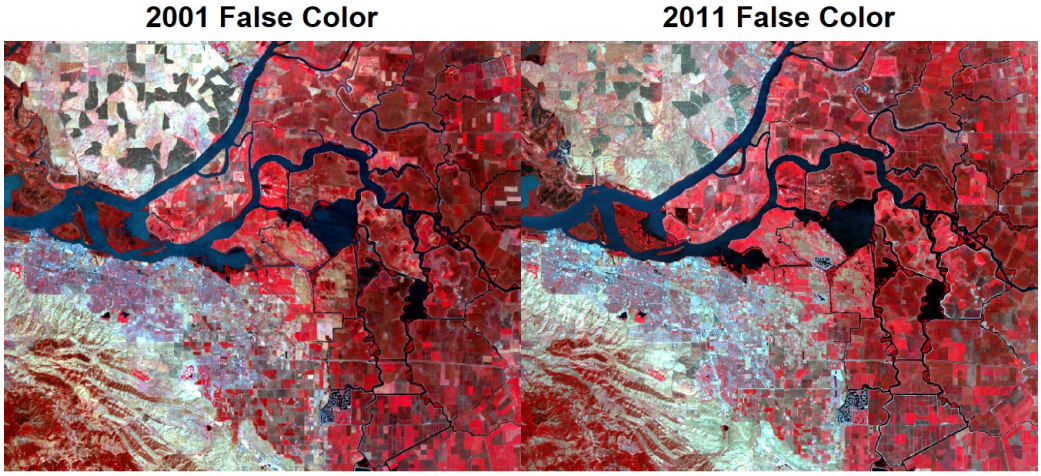
```
# True color composite for Time 2
plotRGB(raster_t2, r = 3, g = 2, b = 1, stretch = "lin",
        main = "Central Valley 2011 (Landsat 5 TM)")
```



True color composite - 2011

Interpretation: The 2011 true color composite allows visual comparison with the 2001 image. Notable changes in land cover may be visible, such as expansion of urban areas (grey), changes in agricultural patterns (green/brown variations), or water body alterations. Differences in overall tone between the two images may also reflect sensor variations between Landsat 7 ETM+ and Landsat 5 TM.

```
# 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))
```

Interpretation: False color composites using NIR (Near-Infrared), Red, and Green bands are particularly valuable for vegetation analysis. Healthy vegetation appears bright red or pink because it reflects strongly in the NIR band. Water appears dark blue or black (low NIR reflectance), bare soil appears brown or tan, and urban areas appear cyan or grey. Comparing the two side-by-side images helps identify changes in vegetation health, agricultural activity, or land use conversion between 2001 and 2011.

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

Interpretation: This table shows the distribution of training polygons across the five land cover classes. The counts reveal the balance (or imbalance) in the training data. Classes with fewer samples (e.g., water with only 3 polygons) may be underrepresented and could result in lower classification accuracy for those classes. The total of 49 polygons is relatively small for a machine learning classifier, making cross-validation particularly important for robust accuracy assessment.

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
```

Interpretation: The output shows the first few rows of extracted spectral data for Time 1. Each row represents a single pixel from within a training polygon. The columns show the land cover class and the reflectance values for all 6 spectral bands (B1-B7). These values typically range from 0 to ~10,000+ in Landsat imagery. Variations in band values across rows reflect different land cover types. This extracted data forms the training dataset that the Random Forest classifier will learn from.

```
# 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
```

Interpretation: Similar to Time 1, this shows the extracted spectral data for Time 2 (2011). Comparing the spectral values between the two time periods may reveal systematic differences due to sensor characteristics (Landsat 7 vs Landsat 5), atmospheric conditions, or phenological changes. Training separate models for each time period accounts for these differences.

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

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

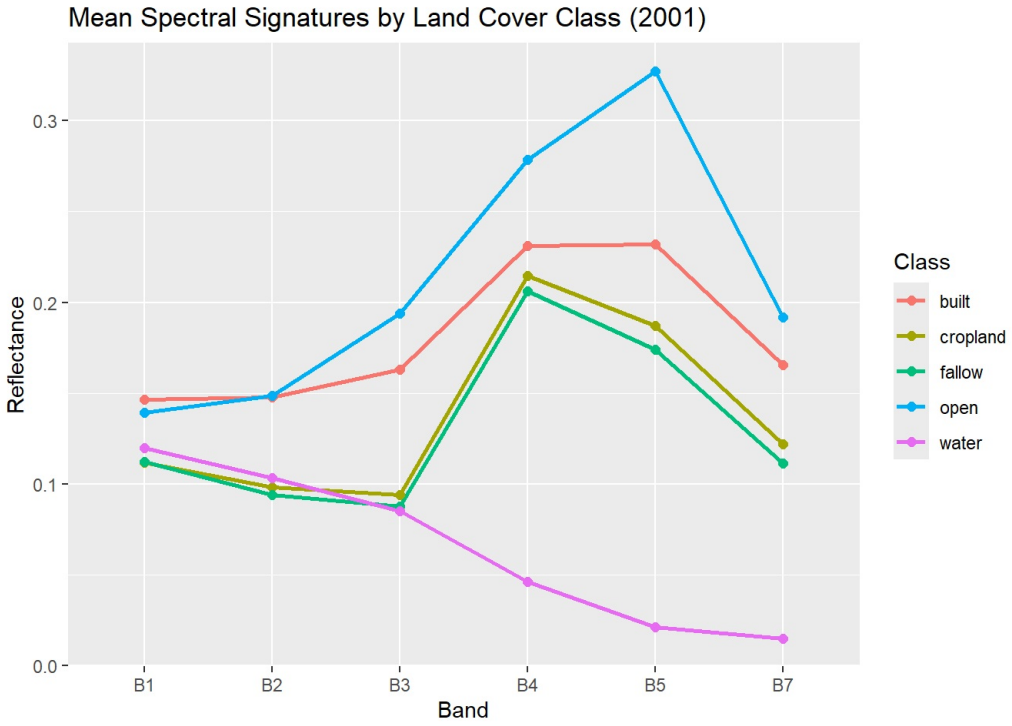
```
cat("Time 2 extracted pixels:", nrow(extract_t2), "\n")
```

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

Interpretation: These counts show the total number of pixels extracted from the training polygons. With 49 polygons, we typically obtain hundreds to thousands of individual pixel samples, which provides substantially more training data than just 49 samples. The similar counts between Time 1 and Time 2 confirm that the same training polygons were used for both dates. More pixels mean more robust model training, though spatial autocorrelation among nearby pixels within the same polygon means these samples are not fully independent.

4.3 Explore Spectral Signatures

```
# Compute mean spectral values per class for Time 1
spectral_means_t1 <- extract_t1 |>
  group_by(class) |>
  summarise(across(B1:B7, mean)) |>
  pivot_longer(cols = B1:B7, names_to = "band", values_to = "reflectance")

ggplot(spectral_means_t1, 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")
```

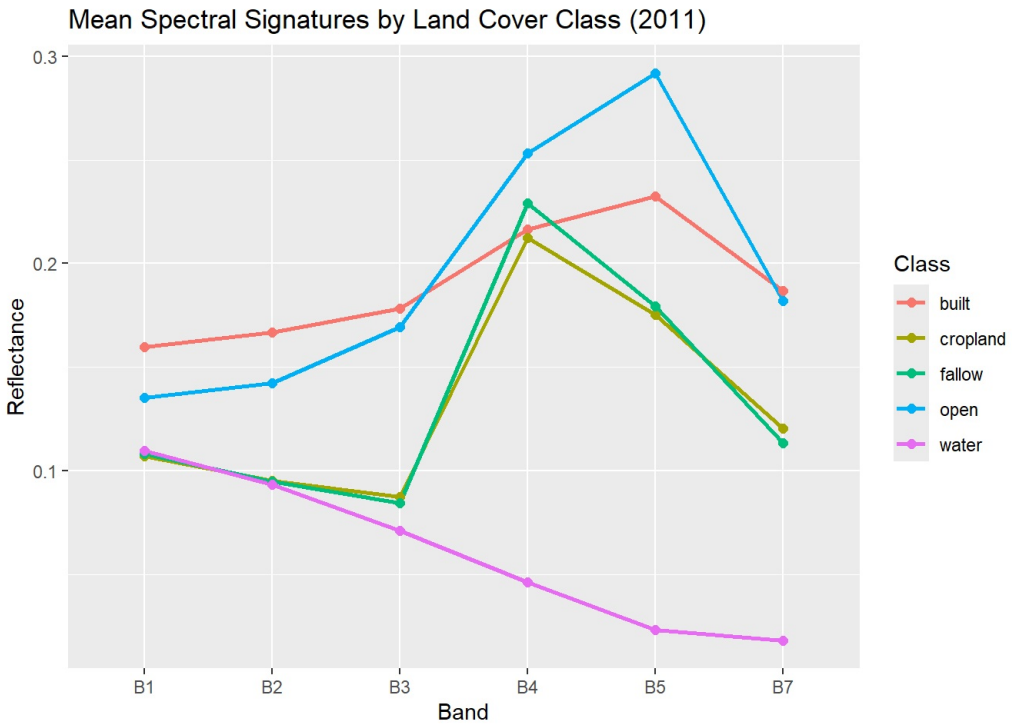


Interpretation: This plot shows the characteristic spectral signature of each land cover class across the six Landsat bands for 2001. Key patterns to look for:

- **Water**: Typically shows high reflectance in visible bands (B1-B3) and very low in NIR/SWIR (B4, B5, B7)
- **Vegetation (cropland)**: Low in visible bands, very high in NIR (B4), moderate in SWIR
- **Built-up areas**: Generally moderate and relatively flat across all bands
- **Bare soil/fallow**: Higher reflectance in SWIR bands (B5, B7) than vegetation
- **Open land**: May show intermediate values depending on cover type

Classes with distinct, well-separated signatures are easier to classify accurately. Overlapping signatures indicate potential confusion between classes.

```
# Compute mean spectral values per class for Time 2
spectral_means_t2 <- extract_t2 |>
  group_by(class) |>
  summarise(across(B1:B7, mean)) |>
  pivot_longer(cols = B1:B7, names_to = "band", values_to = "reflectance")

ggplot(spectral_means_t2, 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 (2011)",
       x = "Band", y = "Reflectance", color = "Class")
```



Interpretation: This plot shows the spectral signatures for 2011. Comparing with the 2001 signatures reveals:

- **Sensor differences**: Systematic shifts in reflectance values between Landsat 7 ETM+ and Landsat 5 TM
- **Temporal changes**: Changes in vegetation health, phenology (seasonal growth stage), or atmospheric conditions
- **Class separability**: Whether the classes remain well-distinguished in the 2011 data

Similar signature shapes across both time periods suggest consistent land cover characteristics, while shifts justify training separate models for each date.

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
```

Interpretation: The data is split into 70% training and 30% testing sets using stratified sampling (proportional class representation). The training set is used to build the Random Forest model, while the test set is held out to evaluate performance on unseen data. The class count table confirms that all five classes are represented in the training data. The stratification ensures that rare classes (like water) are present in both training and test sets despite having few samples.

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 (no normalization needed for tree-based models)
rf_recipe <- recipe(class ~ ., data = train_t1)

# 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 ────────────────────────────────────────────────────────────
## 0 Recipe Steps
##
## ── 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: 1.94%
## Confusion matrix:
##          built cropland fallow  open water class.error
## built     3232       44     28   310     0 0.105700055
## cropland    22     4611    306     1     0 0.066599190
## fallow       8      178   6712     0     0 0.026964337
## open        31        7      2 12968     0 0.003075031
## water        0        0      0     0 19720 0.000000000
```

Interpretation: The output confirms that the Random Forest model with 500 trees has been successfully trained on the training data. The workflow combines the preprocessing recipe and model specification. Random Forest builds an ensemble of 500 decision trees, each trained on a bootstrap sample of the data and a random subset of features. The final classification is determined by majority vote across all trees. The 500-tree ensemble balances computational efficiency with model stability and accuracy.

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     1381       14      5   21     0
##   cropland    29     1984     64    0     0
##   fallow      19      127   2851    2     0
##   open       125        1      0 5580     0
##   water        0        0      0    0  8448
```

Interpretation: The confusion matrix compares predicted classes (columns) against true classes (rows). Key insights:

- **Diagonal elements**: Correctly classified pixels for each class (higher is better)
- **Off-diagonal elements**: Misclassifications showing which classes are confused with each other
- **Class-specific accuracy**: Classes with high diagonal values relative to their row totals perform well
- **Common confusions**: Large off-diagonal values indicate problematic class pairs (e.g., cropland confused with fallow)

User's accuracy (row perspective) and producer's accuracy (column perspective) can be calculated from this matrix to assess class-specific performance.

```
# 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	0.9802915
kap	multiclass	0.9726884

Interpretation: This table presents overall classification metrics:

- **Accuracy**: The proportion of correctly classified pixels (ranges 0-1, higher is better). Values above 0.80 (80%) are generally considered acceptable for land cover classification.
- **Kappa**: Cohen's Kappa coefficient adjusts accuracy for chance agreement (ranges -1 to 1, higher is better). Values above 0.75 indicate

excellent agreement, 0.40-0.75 fair to good agreement, and below 0.40 poor agreement.

These single-split metrics provide an initial estimate of model performance, but cross-validation gives a more reliable assessment.

5.4 Cross-Validation for Robust Accuracy Estimates

With only 49 training polygons, a single 70/30 split can produce unstable accuracy estimates. We use 5-fold cross-validation (stratified by class) on the full extracted pixel data to get a more reliable picture of model performance.

```
set.seed(3917)

# Create 5-fold CV resamples stratified by class
cv_folds_t1 <- vfold_cv(extract_t1, v = 5, strata = class)

# Define workflow (unfitted) for resampling
rf_cv_workflow <- workflow() |>
  add_recipe(recipe(class ~ ., data = extract_t1)) |>
  add_model(rf_model)

# Fit across folds
cv_results_t1 <- fit_resamples(
  rf_cv_workflow,
  resamples = cv_folds_t1,
  metrics = metric_set(accuracy, kap)
)

# Summarize CV metrics
collect_metrics(cv_results_t1) |>
  kable(caption = "5-Fold Cross-Validation Metrics — Time 1 (2001)") |>
  kable_styling(bootstrap_options = c("striped", "hover"), full_width = FALSE)
```

5-Fold Cross-Validation Metrics — Time 1 (2001)

.metric	.estimator	mean	n	std_err	.config
accuracy	multiclass	0.9823771	5	0.0004127	pre0_mod0_post0
kap	multiclass	0.9755827	5	0.0005708	pre0_mod0_post0

Interpretation: The cross-validation results provide more robust performance estimates by averaging across 5 independent train-test splits:

- **mean**: The average accuracy and Kappa across all 5 folds, giving a more stable estimate than a single split
- **n**: Number of folds (5)
- **std_err**: Standard error of the mean, indicating variability in performance across folds. Lower values suggest stable performance across different data subsets.

If CV metrics are substantially lower than single-split metrics, it suggests the single split may have been overly optimistic. The standard error helps quantify uncertainty in the performance estimate.

```
set.seed(5284)

# Cross-validation for Time 2
cv_folds_t2 <- vfold_cv(extract_t2, v = 5, strata = class)

cv_results_t2 <- workflow() |>
  add_recipe(recipe(class ~ ., data = extract_t2)) |>
  add_model(rf_model) |>
  fit_resamples(
    resamples = cv_folds_t2,
    metrics = metric_set(accuracy, kap)
  )

collect_metrics(cv_results_t2) |>
  kable(caption = "5-Fold Cross-Validation Metrics — Time 2 (2011)") |>
  kable_styling(bootstrap_options = c("striped", "hover"), full_width = FALSE)
```

5-Fold Cross-Validation Metrics — Time 2 (2011)

.metric	.estimator	mean	n	std_err	.config
accuracy	multiclass	0.9827839	5	0.0003395	pre0_mod0_post0
kap	multiclass	0.9761587	5	0.0004690	pre0_mod0_post0

Interpretation: The Time 2 cross-validation metrics allow comparison with Time 1 performance. Similar accuracy between the two dates suggests consistent model performance despite sensor differences. Lower accuracy for Time 2 might indicate:

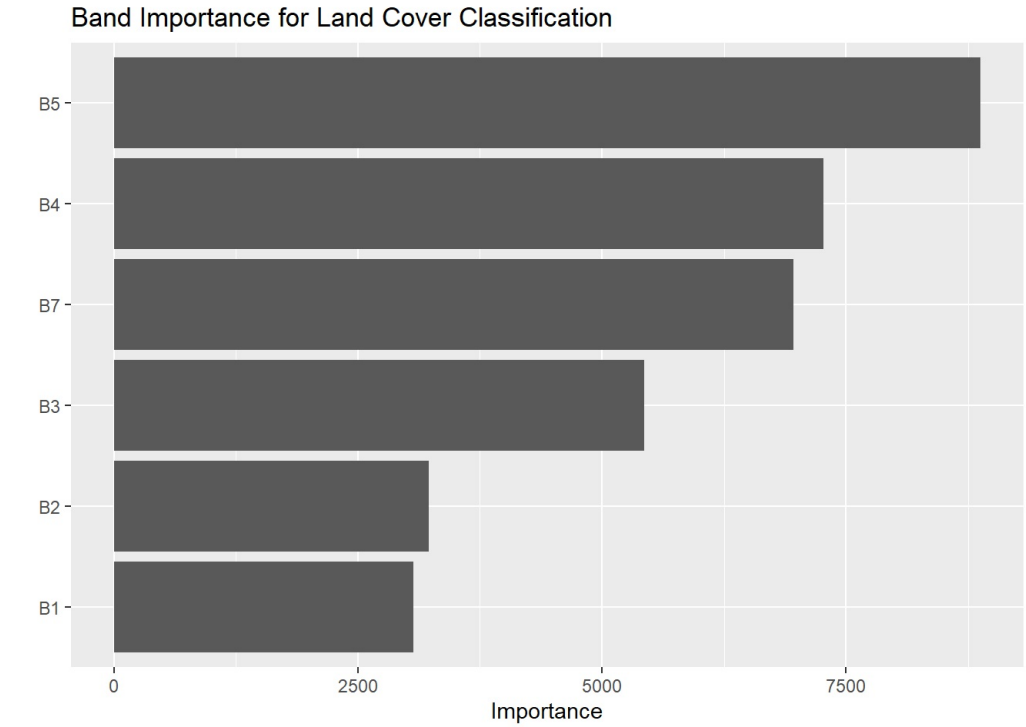
- More complex spectral patterns in the 2011 image
- Greater class overlap in spectral signatures
- Differences in atmospheric conditions or phenology

The standard error again quantifies uncertainty. Comparing standard errors between Time 1 and Time 2 indicates whether one time period has more variable performance across folds.

The cross-validated accuracy provides a more honest estimate of model performance than a single train/test split, especially given the small training sample. The standard error column indicates the variability across folds.

5.5 Variable Importance

```
library(vip)

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



Interpretation: The variable importance plot ranks spectral bands by their contribution to classification accuracy. Key insights:

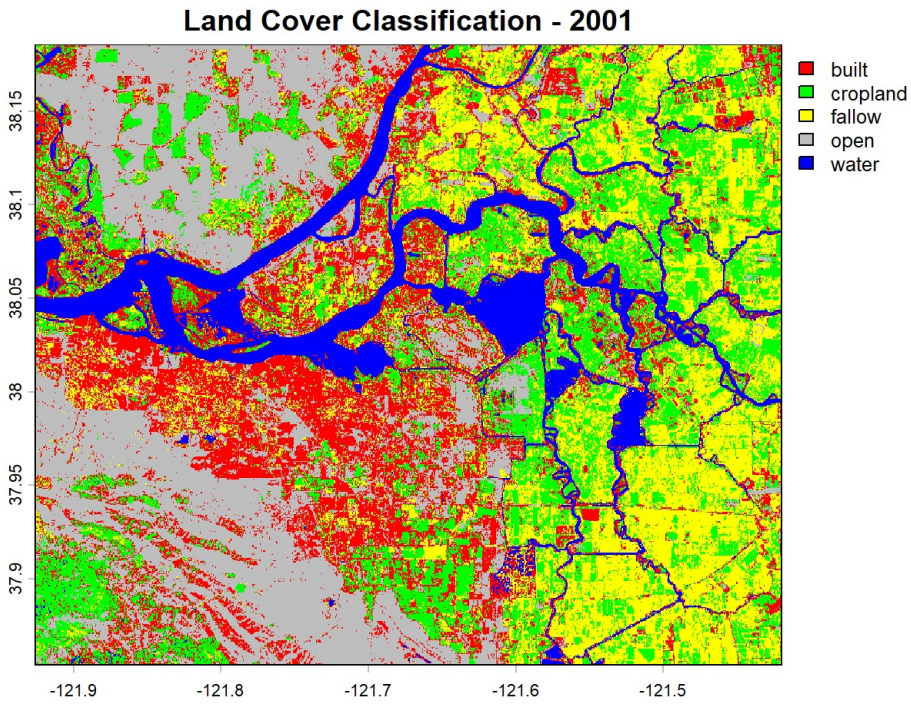
- **NIR (B4)**: Often most important for distinguishing vegetation from non-vegetation
- **SWIR bands (B5, B7)**: Critical for separating soil, built-up areas, and vegetation moisture content
- **Visible bands (B1-B3)**: May be less important if vegetation/non-vegetation is the primary distinction, but crucial for water identification

Bands with low importance could potentially be excluded to simplify the model, though Random Forest handles high dimensionality well. Understanding which bands drive classification helps interpret the model's decision-making and can inform feature selection for other algorithms.

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")
```



Interpretation: This map shows the classified land cover for the entire study area in 2001. Each pixel is assigned to one of the five classes based on the trained Random Forest model. Visual interpretation:

- **Spatial patterns**: Cropland (green) should cluster in agricultural areas, water (blue) in rivers/reservoirs
- **Reasonableness**: Does the classification match expected land use patterns for the Central Valley?
- **Artifacts**: Look for unusual patterns like isolated pixels or checkerboard effects that might indicate classification errors
- **Class distribution**: Relative abundance of each class visible by color dominance

This classified map serves as the baseline for change detection when compared with the 2011 classification.

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 (no normalization needed for tree-based models)
rf_workflow_t2 <- workflow() |>
  add_recipe(recipe(class ~ ., data = train_t2)) |>
  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)
```

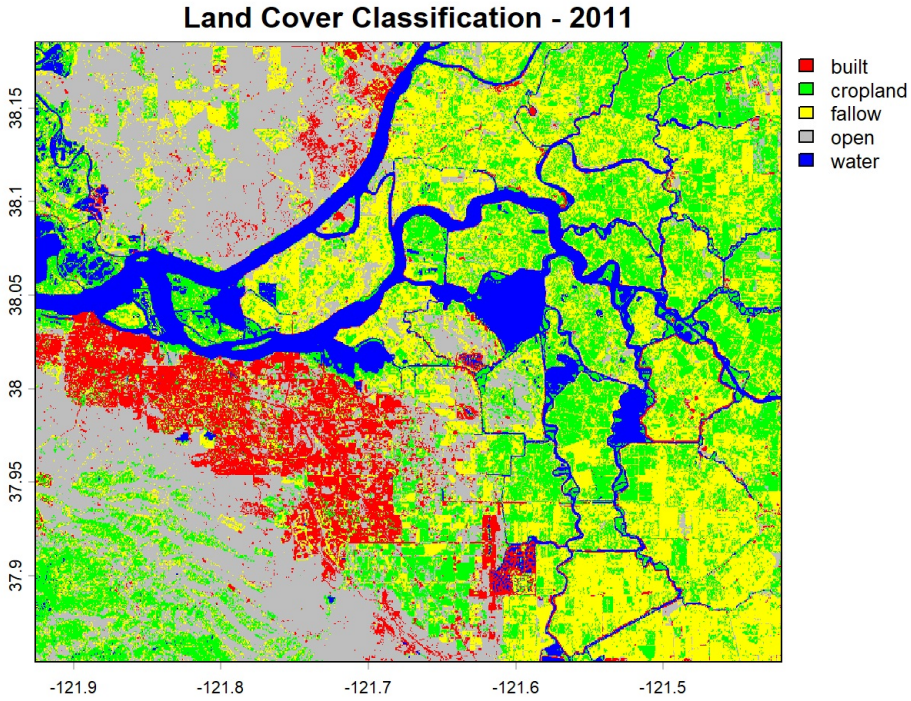
```
##           Truth
## Prediction built cropland fallow open water
##   built     1423        2      0   12     0
##   cropland     0     1952     85    9     0
##   fallow       0      153   2868   17     0
##   open       111        2      5 5548     0
##   water        0        0      0    0  8463
```

Interpretation: The Time 2 confusion matrix reveals the classification performance for the 2011 image. Compare this with the Time 1 confusion matrix to assess:

- **Consistency**: Are the same classes well-classified or problematic in both years?
- **Improvements/degradations**: Did certain class confusions improve or worsen?
- **Overall quality**: Does Time 2 classification appear as reliable as Time 1?

Since change detection relies on both classifications, classes with low accuracy in either time period will contribute to error propagation in the final change maps.

```
# 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")
```



Interpretation: The 2011 classified map provides the second time period for change detection. Visual comparison with the 2001 map may reveal:

- **Urban expansion**: Red (built) areas expanding into green (cropland) or yellow (fallow)
- **Agricultural changes**: Shifts between cropland and fallow lands
- **Water body changes**: Expansion or contraction of reservoirs or wetlands
- **Overall landscape change**: Major alterations in land use patterns

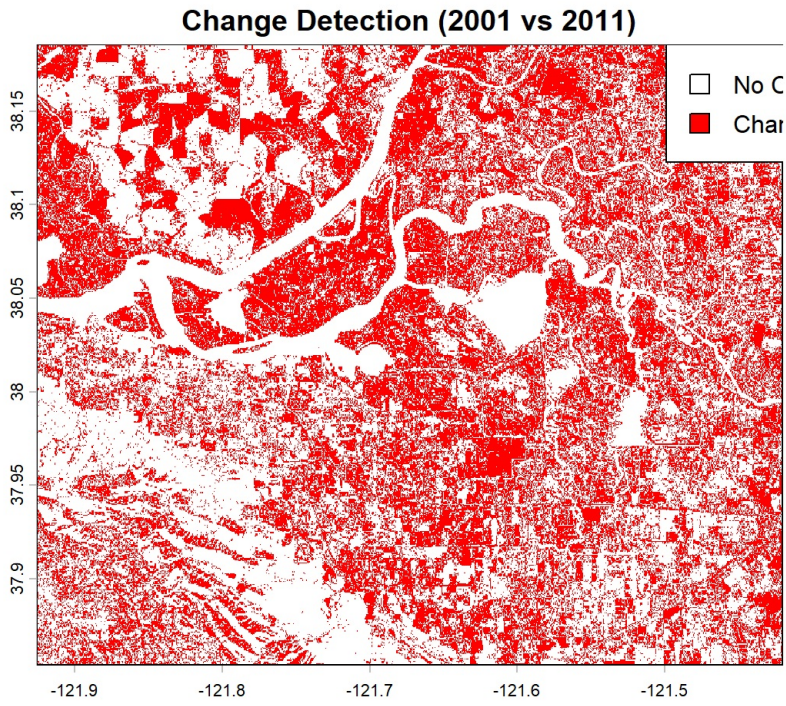
These visual changes will be quantified systematically in the change detection analysis.

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"))
```



Interpretation: This binary change map highlights all pixels where the classified land cover class changed between 2001 and 2011:

- **Red pixels**: Changed from one class to another (any transition)
- **White pixels**: Remained in the same class (stable areas)
- **Spatial patterns**: Clusters of change may indicate systematic processes like urban expansion fronts or agricultural conversion
- **Change extent**: Visual proportion of red to white indicates the overall magnitude of landscape change

Note that this map shows **all** changes but doesn't distinguish between different types of transitions. Some red pixels may represent true land cover change, while others may reflect classification errors (error propagation).

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()

# Map numeric codes to actual class names
class_names <- levels(extract_t1$class)  # "built", "cropland", "fallow", "open", "water"
vals_t1 <- class_names[vals_t1]
vals_t2 <- class_names[vals_t2]

# 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

	built	cropland	fallow	open	water
built	130463	128441	133795	95298	13587
cropland	8982	202859	200432	57561	6156
fallow	20496	172870	287175	34050	2020
open	44650	31153	57779	446183	1639
water	370	9140	575	142	222892

Interpretation: The transition matrix shows pixel counts for every possible class transition:

- **Diagonal elements** (e.g., built→built): Pixels that remained in the same class (stability)
- **Off-diagonal elements** (e.g., cropland→built): Pixels that changed from one class to another
- **Row totals**: Total area in each class in 2001
- **Column totals**: Total area in each class in 2011
- **Major transitions**: Large off-diagonal values indicate significant land cover changes (e.g., agricultural land converting to urban)
- **Net change**: Comparing row and column totals shows which classes gained or lost area

This matrix is fundamental for understanding the nature and magnitude of landscape change between the two dates.

We can also express the transition matrix in area units by multiplying pixel counts by the pixel area. The pixel dimensions come from the raster resolution.

```
# Compute pixel area in hectares
pixel_res <- res(raster_t1)  # pixel width and height in CRS units
pixel_area_m2 <- prod(pixel_res)  # area per pixel in m² (assumes projected CRS)

# If the CRS uses degrees (geographic), approximate area using mid-latitude
if (is.lonlat(raster_t1)) {
  mid_lat <- mean(ext(raster_t1)[3:4]) * pi / 180
  pixel_area_m2 <- prod(pixel_res * c(cos(mid_lat), 1) * 111320)
}

pixel_area_ha <- pixel_area_m2 / 10000  # convert m² to hectares

cat("Pixel resolution:", pixel_res[1], "x", pixel_res[2], "\n")
```

```
## Pixel resolution: 0.0002694946 x 0.0002694946
```

```
cat("Approximate pixel area:", round(pixel_area_m2, 1), "m² (",
    round(pixel_area_ha, 4), "ha)\n")
```

```
## Approximate pixel area: 709 m² ( 0.0709 ha)
```

```
# Transition matrix in hectares
transition_area <- as.data.frame.matrix(transition_table) * pixel_area_ha

transition_area |>
  mutate(across(everything(), ~round(.x, 1))) |>
  kable(caption = "Land Cover Transition Matrix (hectares): 2001 to 2011") |>
  kable_styling(bootstrap_options = c("striped", "hover"))
```

Land Cover Transition Matrix (hectares): 2001 to 2011

	built	cropland	fallow	open	water
built	9250.2	9106.8	9486.4	6756.9	963.4
cropland	636.8	14383.2	14211.1	4081.2	436.5
fallow	1453.2	12256.9	20361.4	2414.2	143.2
open	3165.8	2208.8	4096.7	31635.5	116.2
water	26.2	648.0	40.8	10.1	15803.6

Interpretation: The pixel resolution and area calculations convert the geographic coordinates to meaningful area measurements. For geographic coordinates (degrees), the conversion accounts for latitude-dependent distortion using the mid-latitude correction.

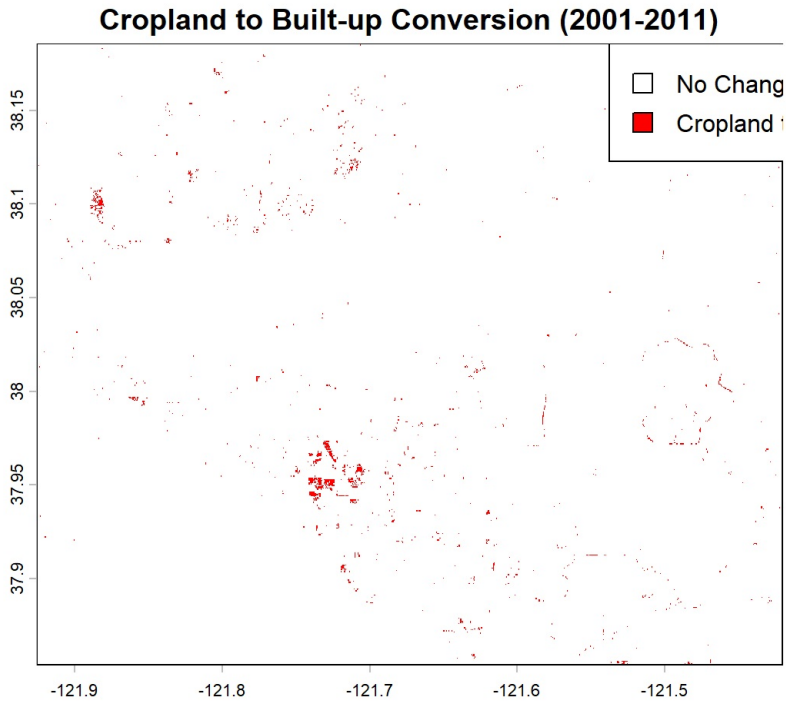
The area-based transition matrix provides the same information as the pixel counts but in hectares, making results more interpretable for land management and policy applications. For example:

- How many hectares of cropland were converted to built-up areas?
- What is the net gain/loss for each land cover type?
- Which transitions represent the largest area changes?

This format is more accessible to non-technical stakeholders and facilitates comparison with official land use statistics.

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"))
```



Interpretation: This map isolates a single transition type—cropland converting to built-up areas, representing urbanization or development. Red pixels show where agricultural land was developed between 2001 and 2011. Key observations:

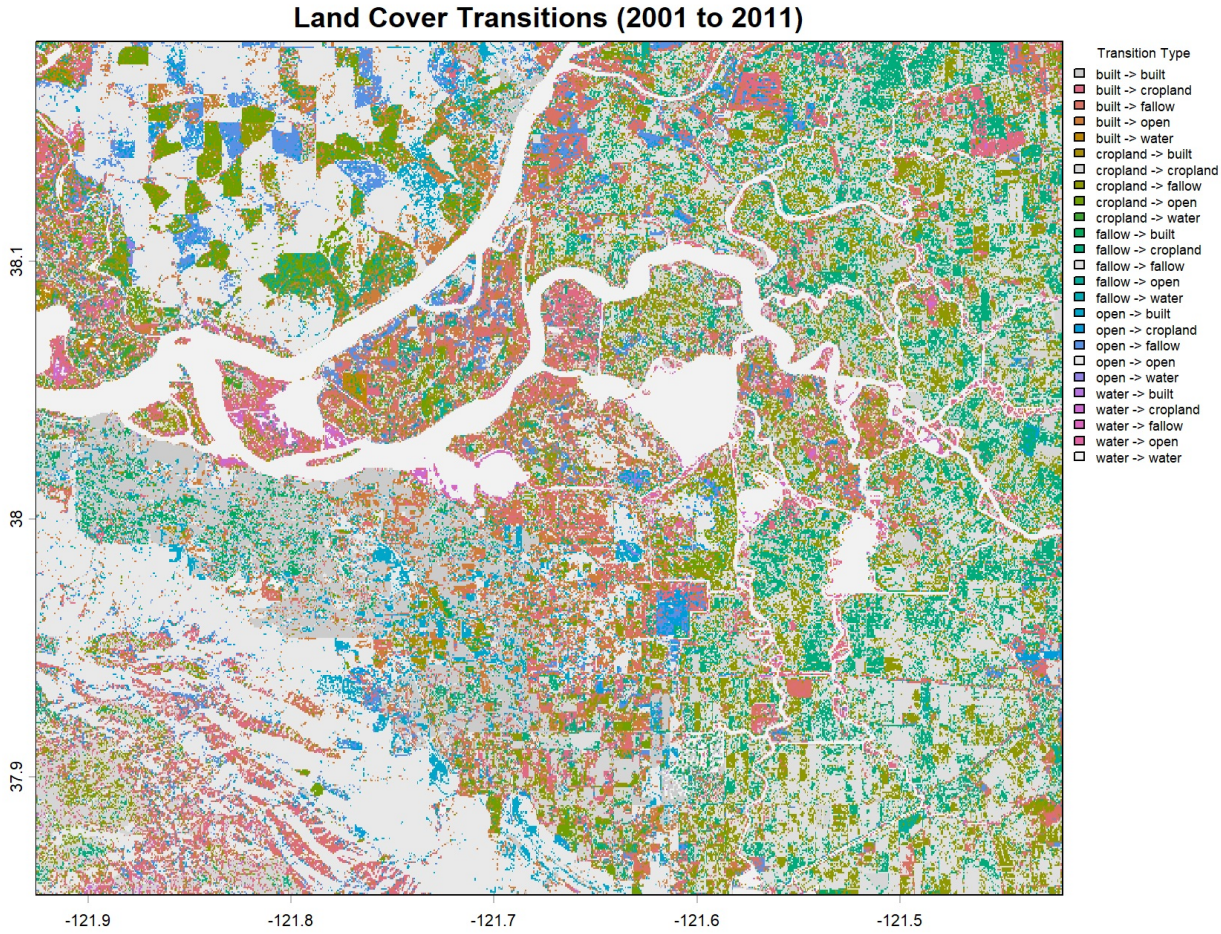
- **Spatial patterns**: Urbanization often occurs along existing development edges, transportation corridors, or near cities
- **Clustered vs. scattered**: Clustered patterns suggest planned development; scattered patterns may indicate sprawl or isolated developments
- **Hotspots**: Areas with concentrated red pixels are urbanization hotspots requiring land use planning attention
- **Magnitude**: The extent of red coverage indicates the scale of agricultural land loss to development

This type of specific transition map is valuable for focused policy questions about particular land use changes.

7.4 Map All Class Transitions

Beyond the binary change map and the single cropland-to-built example, we can create a comprehensive transition map that shows every from-to class combination spatially. Each pixel is labeled with its specific transition type (e.g., “cropland -> fallow”, “open -> built”).

```r
# Create a transition label for every pixel using class names
transition_labels <- ifelse(
  valid,
  paste0(vals_t1[valid], " -> ", vals_t2[valid]),
  NA
)

# Build a raster of transition codes
transition_rast <- classified_t1[[1]]  # copy structure
transition_ids <- as.integer(factor(transition_labels))
vals_full <- rep(NA_integer_, length(vals_t1))
vals_full[valid] <- transition_ids
values(transition_rast) <- vals_full

# Map integer codes back to transition labels (class names)
label_lookup <- levels(factor(transition_labels))
levels(transition_rast) <- data.frame(
  id = seq_along(label_lookup),
  label = label_lookup
)

# Separate "no change" vs "change" transitions for coloring
no_change_labels <- label_lookup[sapply(strsplit(label_lookup, " -> "), \(x) x[1] == x[2])]
change_labels <- setdiff(label_lookup, no_change_labels)

# Assign grey tones to "no change" and distinct colors to "change" transitions
n_change <- length(change_labels)
grey_palette <- grey.colors(length(no_change_labels), start = 0.8, end = 0.95)
change_palette <- hcl.colors(n_change, palette = "Dark 3")

# Combine into a named color vector matching label order
color_vec <- setNames(
  c(grey_palette, change_palette),
  c(no_change_labels, change_labels)
)
color_vec <- color_vec[label_lookup]  # reorder to match factor levels

plot(transition_rast, main = "Land Cover Transitions (2001 to 2011)",
     col = color_vec, type = "classes", mar = c(2, 2, 2, 12),
     plg = list(title = "Transition Type", cex = 0.7))
```



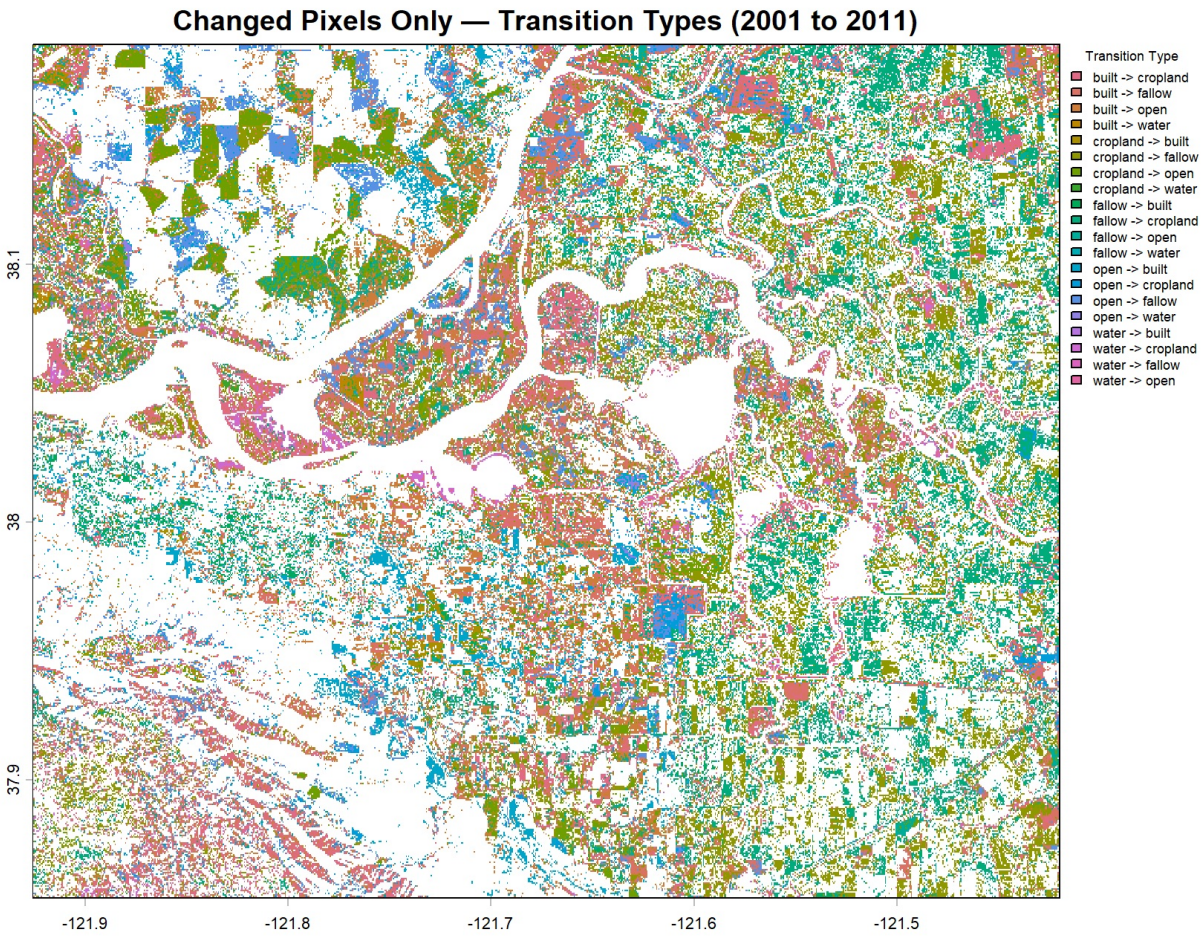
Interpretation: This comprehensive transition map displays all possible class transitions simultaneously:

- **Grey tones**: Pixels that remained stable (same class in both years)
- **Distinct colors**: Each specific transition type (e.g., cropland→built, fallow→cropland, open→water)
- **Spatial clustering**: Similar transition types clustering together indicate coherent change processes
- **Multiple change types**: Reveals the complexity and diversity of landscape change beyond simple binary change
- **Legend**: Identifies which color corresponds to which specific transition

This map provides the most complete spatial representation of landscape change, showing not just **where** change occurred but **what type** of change happened at each location.

Pixels that remained in the same class are shown in grey tones, while actual transitions are highlighted with distinct colors. This map makes it straightforward to see the spatial distribution of all change types at once.

```
# Map showing only pixels that changed class (mask out no-change)
change_only_rast <- transition_rast
no_change_ids <- which(label_lookup %in% no_change_labels)
change_only_rast[change_only_rast %in% no_change_ids] <- NA

# Colors for change-only labels (using class names)
change_only_labels <- label_lookup[!label_lookup %in% no_change_labels]
change_only_ids <- which(label_lookup %in% change_only_labels)
change_only_colors <- color_vec[change_only_ids]

plot(change_only_rast, main = "Changed Pixels Only — Transition Types (2001 to 2011)",
     col = change_only_colors, type = "classes", mar = c(2, 2, 2, 12),
     plg = list(title = "Transition Type", cex = 0.7))
```



Interpretation: This filtered version shows only pixels that actually changed, removing stable areas to focus attention on landscape dynamics:

- **Clearer visualization**: Without stable areas (grey), change patterns are more visible
- **Change concentration**: Areas of active land cover change stand out clearly
- **Transition diversity**: The variety of colors shows the diversity of change types
- **Hot spots**: Regions with multiple overlapping transition types indicate areas of complex landscape change
- **Background white**: Represents stable areas (no change) for spatial context

This map is particularly useful for identifying priority areas for detailed investigation or ground-truthing of detected changes.

7.5 Quantify Changes

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

# Add area columns
transition_df <- transition_df |>
  mutate(
    Area_ha = round(Pixels * pixel_area_ha, 1),
    Area_km2 = round(Area_ha / 100, 3)
  )

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

Land Cover Transitions with Area Estimates (Changed Pixels Only)

From_2001	To_2011	Pixels	Area_ha	Area_km2

cropland	fallow	200432	14211.1	142.111
fallow	cropland	172870	12256.9	122.569
built	fallow	133795	9486.4	94.864
built	cropland	128441	9106.8	91.068
built	open	95298	6756.9	67.569
open	fallow	57779	4096.7	40.967
cropland	open	57561	4081.2	40.812
open	built	44650	3165.8	31.658
fallow	open	34050	2414.2	24.142
open	cropland	31153	2208.8	22.088
fallow	built	20496	1453.2	14.532
built	water	13587	963.4	9.634
water	cropland	9140	648.0	6.480
cropland	built	8982	636.8	6.368
cropland	water	6156	436.5	4.365
fallow	water	2020	143.2	1.432
open	water	1639	116.2	1.162
water	fallow	575	40.8	0.408
water	built	370	26.2	0.262
water	open	142	10.1	0.101

Interpretation: This table quantifies all land cover transitions, sorted by magnitude:

- **From_2001 / To_2011**: The specific transition type
- **Pixels**: Number of pixels that underwent this transition
- **Area_ha / Area_km2**: Area in hectares and square kilometers

Key insights from this table:

- **Dominant transitions**: Top rows show the most extensive changes (e.g., if cropland→fallow is #1, agricultural abandonment is a major process)
- **Minor transitions**: Bottom rows may represent classification errors or localized changes
- **Unexpected transitions**: Illogical transitions (e.g., water→built) likely indicate classification errors
- **Change magnitude**: Total area changed can be summed across all transitions
- **Prioritization**: Largest transitions warrant detailed investigation and ground-truthing

This table is the quantitative summary that complements the spatial transition maps, essential for reporting and decision-making.

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, overall accuracy metrics, and 5-fold cross-validation for robust performance estimates
5. **Change detection** — Post-classification comparison to map land cover change
6. **Transition mapping** — Comprehensive spatial map of all from-to class transitions, with a change-only variant
7. **Transition analysis** — Quantified from-to transitions in both pixel counts and area (hectares / km²)

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. Cross-validation helps provide more stable accuracy estimates despite the small

sample.

- **Error propagation**: Post-classification change detection compounds errors from both independent classifications, so high per-class accuracy is important for reliable change maps.
- **Normalization**: Feature normalization (e.g., `step_normalize()`) is unnecessary for Random Forest and other tree-based methods, which are invariant to monotonic transformations of predictors.