Improving the quality of training samples

Alicia Valdés

27 February 2025

Load libraries

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tibble)
library(purrr)
library(tidyr)
library(sf)
```

```
## Linking to GEOS 3.12.2, GDAL 3.9.3, PROJ 9.4.1; sf_use_s2() is TRUE
```

```
library(sits)
```

```
## SITS - satellite image time series analysis.

## Loaded sits v1.5.2.
##         See ?sits for help, citation("sits") for use in publication.
##         Documentation avaliable in https://e-sensing.github.io/sitsbook/.

## Important: Please read "Release Notes for SITS 1.5.2" in
##                 https://github.com/e-sensing/sits.
```

```
library(kohonen)
```

```
##
## Attaching package: 'kohonen'
```

```
## The following object is masked from 'package:purrr':
##
##     map
```

```
library(here)
```

```
## here() starts at C:/Users/alici/OneDrive - Universidad de Oviedo/IMIB/Analyses/PAs/InSitu_curation
```

Load previously saved objects

These objects took a long time to generate!

```
samples_organized <- readRDS(file = here("r_objects", "samples_organized.rds"))
#cross_val <- readRDS(file = here("r_objects", "cross_val.rds"))
samples_SOM_cluster <- readRDS(here("r_objects", "samples_SOM_cluster.rds"))
samples_SOM_new_cluster <- readRDS(file = here("r_objects", "samples_SOM_new_cluster.rds"))
samples_RSI <- readRDS(here("r_objects", "samples_RSI.rds"))
som_cluster_bal <- readRDS(file = here("r_objects", "som_cluster_bal.rds"))
```

Data prep

Implementing csv to Rstudio format

No need to run again if using saved objects!

```
# Cargamos el csv en la ruta correspondiente.
samples <- read.csv(here("data", "AT_ALPENNINNE.csv"))

# Organizamos en el formato/estructura el csv para que RStudio acepte el formato.

samples_organized <- samples %>%
  pivot_wider(names_from = "band", values_from = "value") %>%  # Volver a formato ancho
  group_by(longitude, latitude, start_date, end_date, label, cube) %>%
  nest(time_series = c(Index, B02, B03, B04, B08, B09, B10))  # Anidar las bandas

samples_organized <- samples_organized %>%
  mutate(time_series = map(time_series, ~ .x %>%
                             # Convertir Index a formato Date
                             mutate(Index = as.Date(Index, format = "%Y-%m-%d"))))

# Accedemos a la primera time_series
samples_organized$time_series[[1]]

# Posiblemente en la integración de los datos puede llegar a haber en ocasiones
# errores en el formato de los campos por lo que nos aseguramos de que cada campo tenga su formato.

samples_organized <- samples_organized %>%
  mutate(
    start_date = as.Date(start_date, format = "%Y-%m-%d"),  # Convertir a formato Date
```

```
    end_date = as.Date(end_date, format = "%Y-%m-%d"),      # Convertir a formato Date
    label = as.character(label)  # Convertir a character
  )

saveRDS(samples_organized, file = here("r_objects", "samples_organized.rds"))
```

Ungroup the samples

We need ungroup() for some algorithms to work.

```
samples_organized_ungr <- samples_organized %>% ungroup()
```

Transform to sits class

```
# Se implementa como un conjunto de datos en formato 'group_dbf' y
# para aplicar los procesamientos que queremos tenemos que transformarlo a 'sits'.
class(samples_organized) <- c("sits", class(samples_organized))
class(samples_organized_ungr) <- c("sits", class(samples_organized_ungr))
```

Obtaining samples using Rstudio (do not use)

```
#####_________________ COMANDO PARA REALIZAR LA OBTENCIÓN DE LOS SAMPLES MEDIANTE RSTUDIO _____________
#####________ SIMPLEMENTE AGREGADO PARA CONOCER COMO SE HA OBTENIDO EL ARCHIVO DATA (NO USADO) _________

# Cargamos el datacube en el formato que admite SITS.

datacube <- sits_cube(
  source = "MPC",
  collection = "SENTINEL-2-L2A",
  data_dir = "RUTA/AL/DATACUBE"
)

# Cargamos el shapefile de los puntos que solo deben contener la etiqueta 'label'.

shp_file <- "RUTA/AL/SHAPEFILE.shp"
if (file.exists(shp_file)) {
  sf_shape <- st_read(shp_file)
  print(sf_shape)
} else {
  stop("El archivo no existe en la ruta especificada.")
}
# Obtenemos el samples con el que podemos empezar a realizar los analisis.

samples <- sits_get_data(
  cube         = datacube,
  samples      = sf_shape,
  start_date = "2021-01-01",
```

```
  end_date = "2021-12-31",
  progress = TRUE
)
```

Statistics of samples

Run on samples_organized.

```
summary(samples_organized)
```

```
## # A tibble: 16 x 3
##    label  count     prop
##    <chr>  <int>    <dbl>
##  1 1      11923 0.0243
##  2 14     14760 0.0300
##  3 15       423 0.000860
##  4 18      2385 0.00485
##  5 27    153596 0.312
##  6 30     12117 0.0246
##  7 40     19932 0.0405
##  8 41    166360 0.338
##  9 47       214 0.000435
## 10 48      4402 0.00895
## 11 49     15644 0.0318
## 12 51     66437 0.135
## 13 52      1118 0.00227
## 14 55     19188 0.0390
## 15 63       594 0.00121
## 16 73      2527 0.00514
```

Cross-validation - NEEDS RUNNING

```
# Cross-validation (uncertainties)
# Default: validation_split = 0.2 (proportion of original time series set to be used for validation)
# Default: Machine learning method (sits_rfor())
# There is also sits_kfold_validate
# https://e-sensing.github.io/sitsbook/improving-the-quality-of-training-samples.html#cross-validation-

cross_val <- sits_validate(samples_organized)
saveRDS(cross_val, file = here("r_objects", "cross_val.rds"))

# Shows ca. 80% accuracy
# However, this accuracy does not guarantee a good classification result.
# It only shows if the training data is internally consistent.
# (https://e-sensing.github.io/sitsbook/improving-the-quality-of-training-samples.html)
```

Show the result.

```
cross_val
```

Get labels, band, head and class of the samples

```
sits_labels(samples_organized)
```

```
##  [1] "1"  "14" "15" "18" "27" "30" "40" "41" "47" "48" "49" "51" "52" "55" "63"
## [16] "73"
```

```
sits_bands(samples_organized)
```

```
## [1] "B02" "B03" "B04" "B08" "B09" "B10"
```

```
head(samples_organized)
```

```
## # A tibble: 6 x 7
## # Groups:   longitude, latitude, start_date, end_date, label, cube [6]
##   longitude latitude start_date end_date   label cube           time_series
##       <dbl>    <dbl> <date>     <date>     <chr> <chr>          <list>
## 1   367476. 4696678. 2021-01-01 2021-12-01 41    SENTINEL-2-L2A <tibble>
## 2   367476. 4696681. 2021-01-01 2021-12-01 41    SENTINEL-2-L2A <tibble>
## 3   367477. 4696683. 2021-01-01 2021-12-01 41    SENTINEL-2-L2A <tibble>
## 4   367478. 4696678. 2021-01-01 2021-12-01 41    SENTINEL-2-L2A <tibble>
## 5   367479. 4696680. 2021-01-01 2021-12-01 41    SENTINEL-2-L2A <tibble>
## 6   367479. 4696683. 2021-01-01 2021-12-01 41    SENTINEL-2-L2A <tibble>
```

```
class(samples_organized)
```

```
## [1] "sits"       "sits"       "grouped_df" "tbl_df"     "tbl"
## [6] "data.frame"
```

Hierarchical cluster (HC) - ERROR

```
# Realizamos la agrupación que hace HC automáticamente.

samples_HC <- sits_cluster_dendro(samples = samples_organized_ungr,
                                  dist_method = "dtw_basic",
                                  linkage = "ward.D2")

# Error en matrix(0, x_len * (x_len + diagonal_factor)/2L, 1L):
# valor de 'nrow' no válido (demasiado grande o NA)
# Además: Aviso:
#  In x_len * (x_len + diagonal_factor) : NAs producidos por enteros excedidos
```

Error en matrix(0, x_len * (x_len + diagonal_factor)/2L, 1L): valor de 'nrow' no válido (demasiado grande o NA) Además: Aviso: In x_len * (x_len + diagonal_factor) : NAs producidos por enteros excedidos

Try with a subset of the data?

```
sits_cluster_frequency(samples_HC)

# How to know which cluster to remove? --> The ones with mixes of samples from different labels

# Mostramos la agrupación para observar como se ha clasificado
# y para eliminar el grupo realizado por el cluster sería el siguiente comando:

samples_HC_clean <- dplyr::filter(samples_HC, cluster != 4)
samples_HC_clean <- dplyr::filter(samples_HC, cluster != 5)
samples_HC_clean <- dplyr::filter(samples_HC, cluster != 6)

samples_HC_clean_clean <- sits_cluster_clean(samples_HC_clean)
```

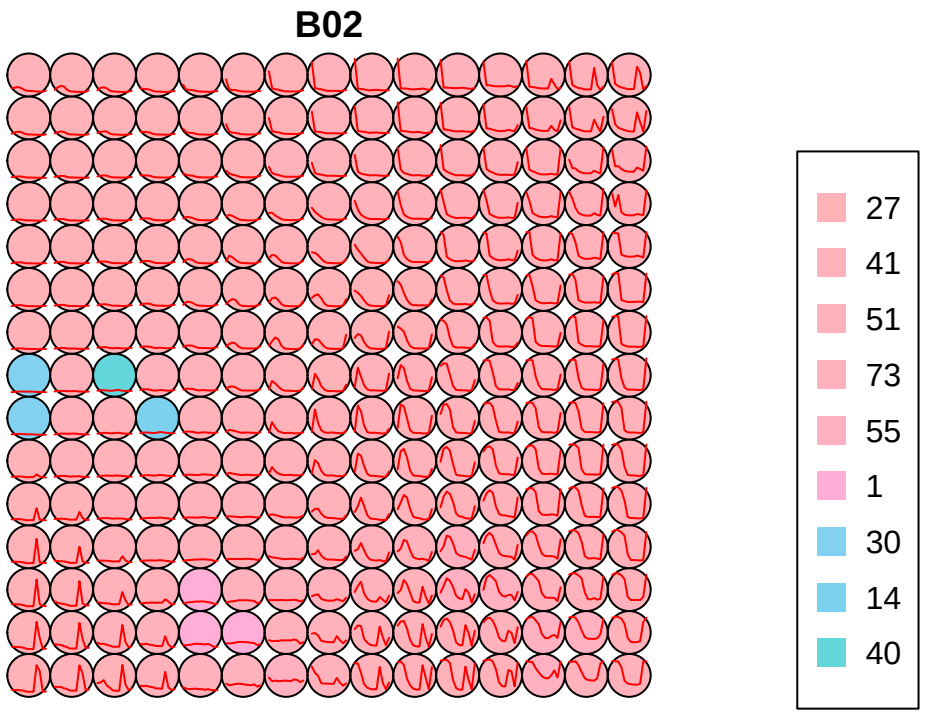
Self-organized map (SOM)

```
samples_SOM_cluster <- sits_som_map(samples_organized_ungr,
                                    grid_xdim = 15,
                                    grid_ydim = 15,
                                    alpha = 1.0,
                                    distance = "dtw",
                                    rlen = 20)
# Avisos:
#   1: In sits_som_map(samples_organized_ungr, grid_xdim = 15, grid_ydim = 15,  :
#      recommended values for grid_xdim and grid_ydim are (57 ...62)
#   2: In RcppSupersom(data = data_matrix, codes = init_matrix, numVars = nvar,  :
#      subscript out of bounds (index 1 >= vector size 1)
#   3: In .colors_get(labels = kohonen_obj[["neuron_label"]], legend = NULL,  :
#      missing colors for labels27, 41, 27, 51, 73, 51, 55, 51, 41, 41, 41, 41, 55, 51, 51, 27, 27, 41,
#   4: In .colors_get(labels = kohonen_obj[["neuron_label"]], legend = NULL,  : palette for missing col
saveRDS(samples_SOM_cluster, file = here("r_objects", "samples_SOM_cluster.rds"))
```

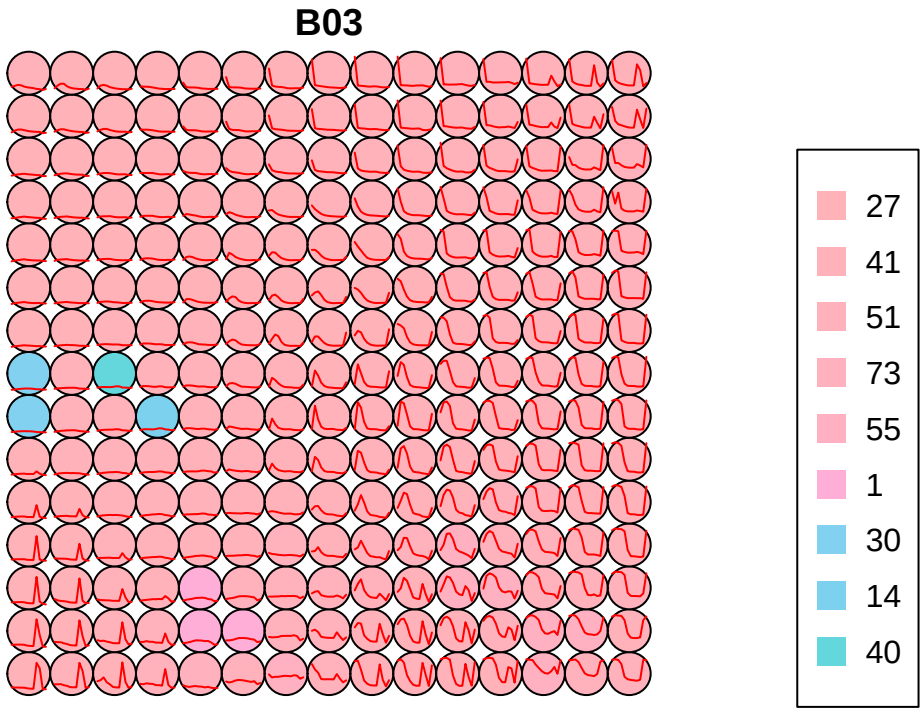
Avisos: 1: In sits_som_map(samples_organized_ungr, grid_xdim = 15, grid_ydim = 15, : recommended values for grid_xdim and grid_ydim are (57 ... 62) 2: In RcppSupersom(data = data_matrix, codes = init_matrix, numVars = nvar, : subscript out of bounds (index 1 >= vector size 1) 3: In .colors_get(labels = kohonen_obj[["neuron_label"]], legend = NULL, : missing colors for labels27, 41, 27, 51, 73, 51, 55, 51, 41, 41, 41, 41, 55, 51, 51, 27, 27, 41, 51, 1, 1, 51, 51, 41, 41, 41, 51, 55, 55, 41, 27, 27, 41, 41, 1, 51, 51, 51, 51, 41, 55, 55, 51, 51, 51, 27, 27, 41, 41, 41, 41, 51, 41, 51, 51, 55, 51, 51, 41, 41, 27, 27, 41, 41, 41, 41, 41, 41, 41, 41, 41, 51, 51, 41, 41, 27, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 30, 41, 41, 14, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 41, 30, 41, 40, 41, 41, 41, 41, 27, 41, 41, 41, 41, 41, 41, 41, 27, 27, 41, 41, 41, 41, 27, 27, 27, 27, 41, 41, 41, 41, 41, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 41, 41, 41, 41, 51, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 41, 41, 41, 41, 41, 51, 27, 27, 27, 27, 27, 27, 27, 41, 41, 41, 41, 41, 41, 41, 41, 51, 27, 27, 27, 27, 27, 27, 27, 41, 41, 41, 41, 41, 41, 41, 41, 41, 27, 27, 27, 27, 27, 27, 41, 41, 41, 51, 51, 41, 41, 41 4: In .colors_get(labels = kohonen_obj[["neuron_label"]], legend = NULL, : palette for missing colors isSet3

Plots for the SOM map (different bands).

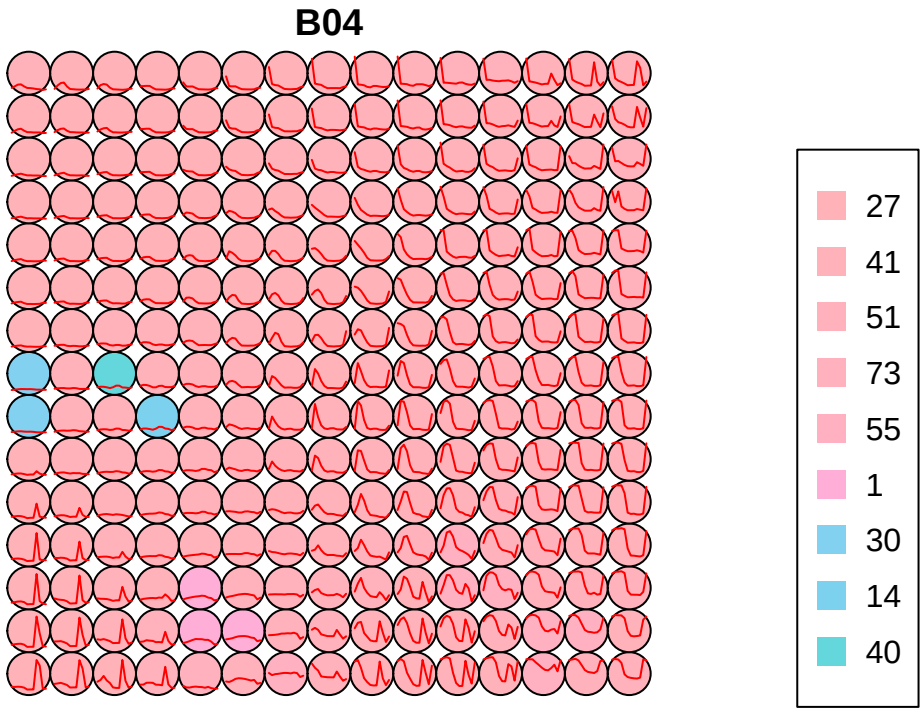
```
plot(samples_SOM_cluster, band = "B02")
```



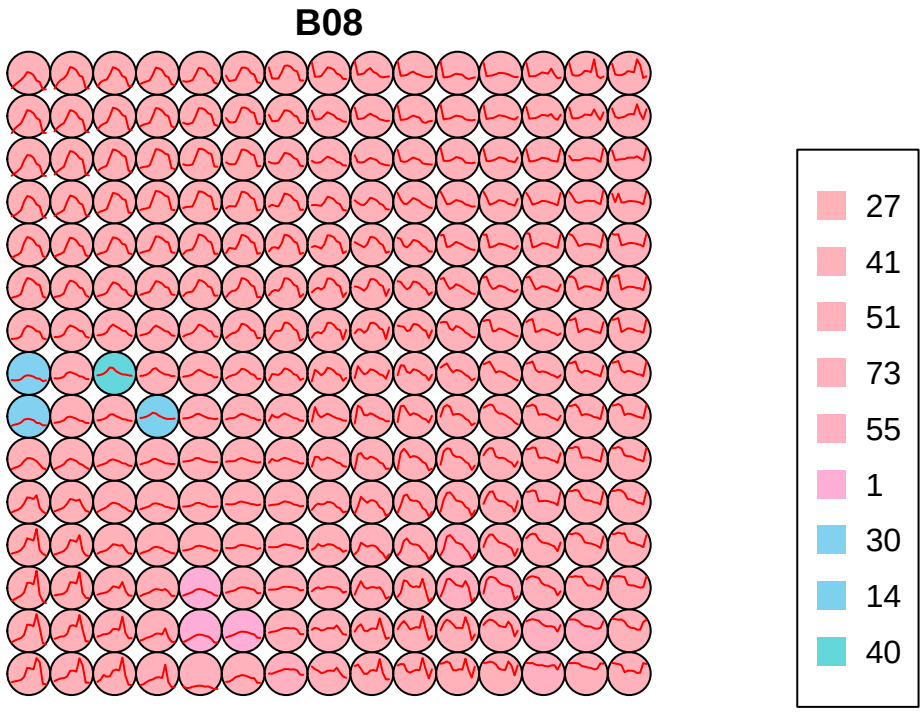

```
plot(samples_SOM_cluster, band = "B03")
```



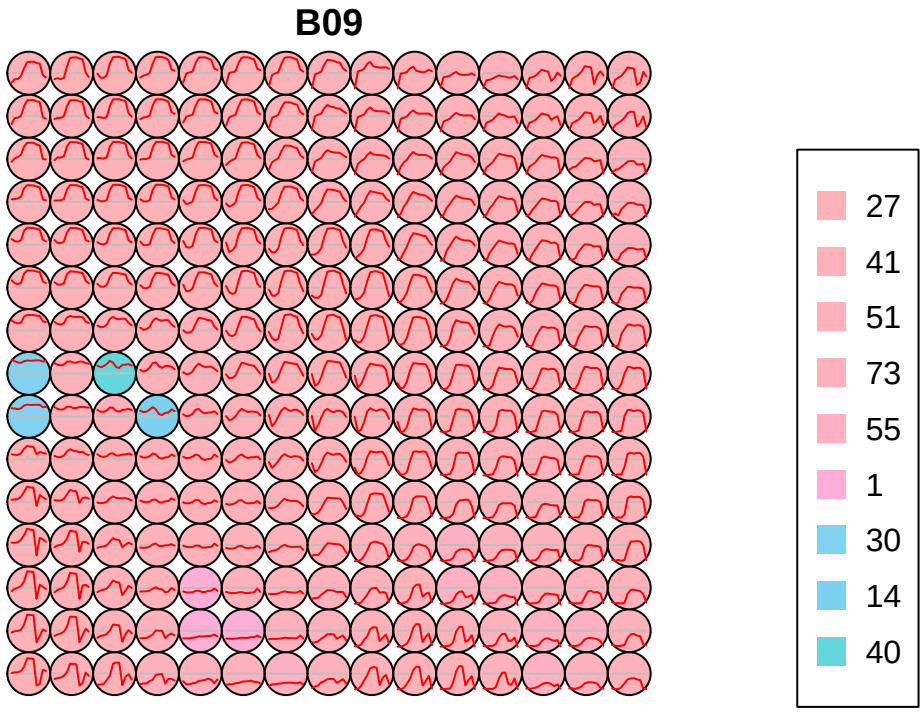

```
plot(samples_SOM_cluster, band = "B04")
```



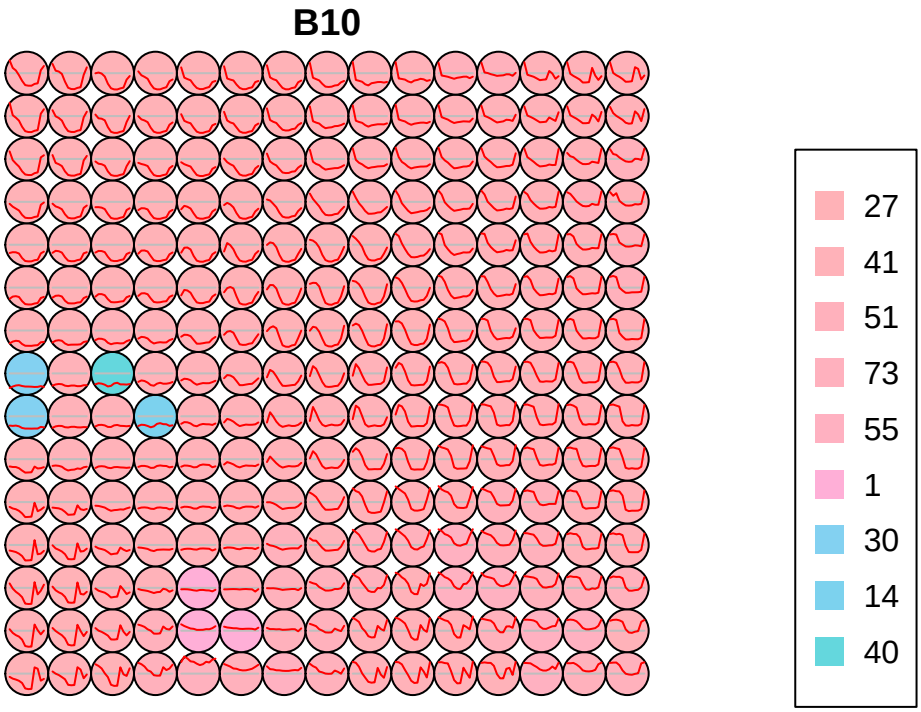

```
plot(samples_SOM_cluster, band = "B08")
```




```
plot(samples_SOM_cluster, band = "B09")
```




```
plot(samples_SOM_cluster, band = "B10")
```



See what to do with warning about the colors - not sure why the labels have so similar colors in the plot. Might need to change the color palette, but not sure how.

Evaluation

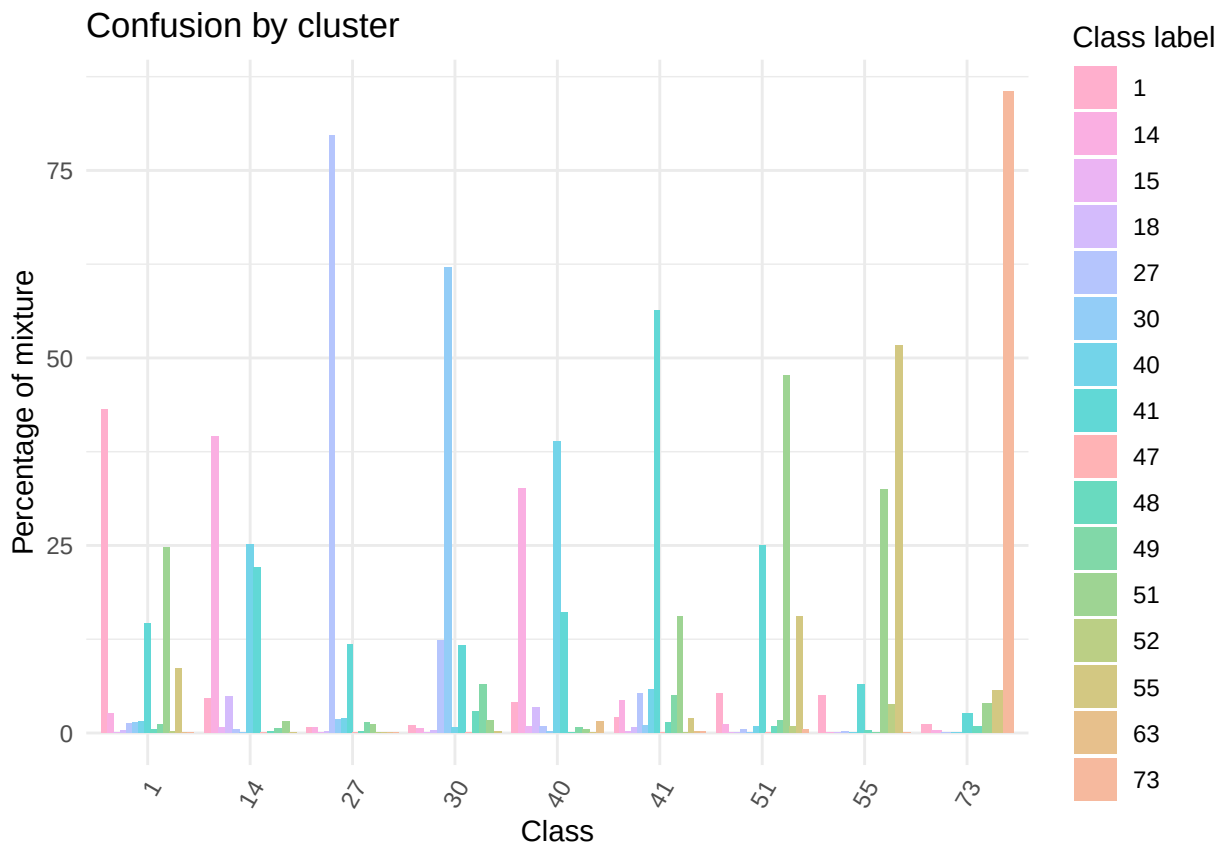
```
som_eval <- sits_som_evaluate_cluster(samples_SOM_cluster) # Lo evaluamos.
```

Plot confusion between clusters.

```
plot(som_eval)
```

```
## Warning in .colors_get(labels = labels, legend = NULL, palette = "Set3", :
## missing colors for labels1, 14, 15, 18, 27, 30, 40, 41, 48, 49, 51, 52, 55, 63,
## 73, 47
```

```
## Warning in .colors_get(labels = labels, legend = NULL, palette = "Set3", :
## palette for missing colors isSet3
```



Some classes missing in the above plot (there are 16 and only 9 are shown) - might have to do with warning about missing colors.

Detecting noisy samples

```
# Generamos los samples con el aprendizaje automatico de SOM.
samples_SOM <- sits_som_clean_samples(
  som_map = samples_SOM_cluster,
  prior_threshold = 0.6,
  posterior_threshold = 0.6,
  keep = c("clean", "analyze")
)
summary(samples_SOM)
```

```
## # A tibble: 5 x 3
##   label  count    prop
##   <chr>  <int>   <dbl>
## 1 27    129555 0.638
## 2 30      4982 0.0245
## 3 41     65620 0.323
## 4 51      1075 0.00529
## 5 73      1896 0.00933
```

This removes many labels! (keeps 5 out of 16).

How many to analyze / clean?

```
samples_SOM %>% count(eval)
```

```
## # A tibble: 2 x 2
##   eval         n
##   <chr>    <int>
## 1 analyze  81775
## 2 clean   121353
```

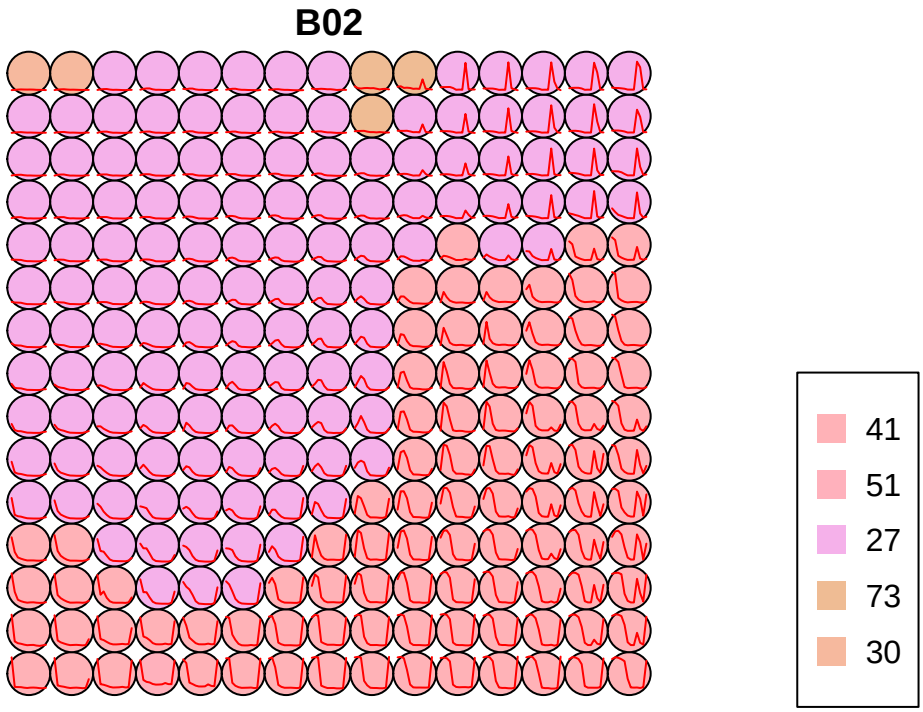
New SOM

Evaluate the mixture in the SOM clusters of new samples.

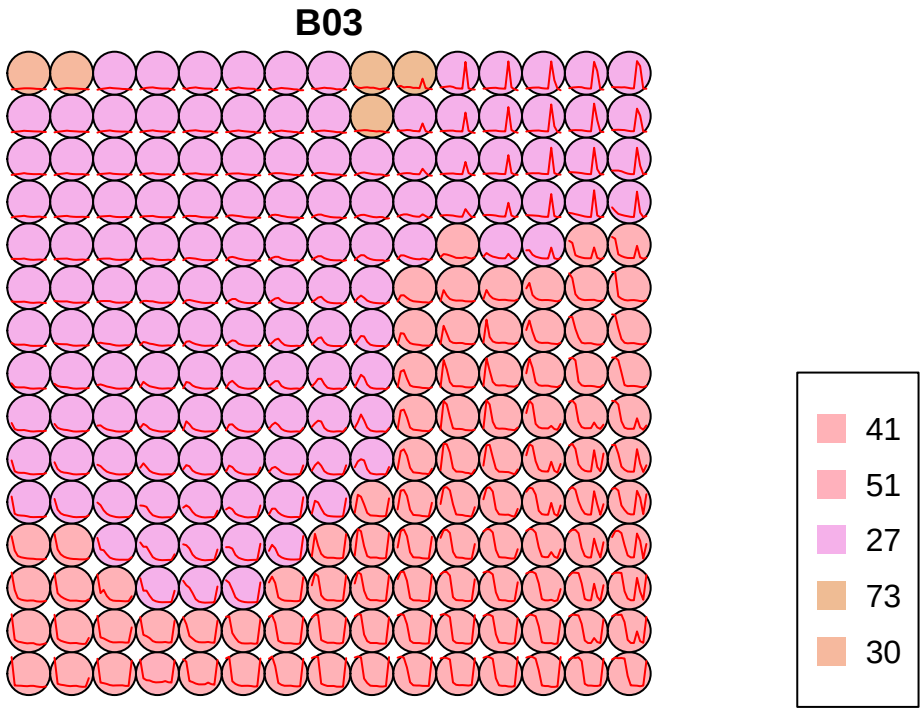
```
samples_SOM_new_cluster <- sits_som_map(
  data = samples_SOM,
  grid_xdim = 15,
  grid_ydim = 15,
  alpha = 1.0,
  rlen = 20,
  distance = "dtw"
)

saveRDS(samples_SOM_new_cluster, file = here("r_objects", "samples_SOM_new_cluster.rds"))
```

Plots for the new SOM map (different bands).

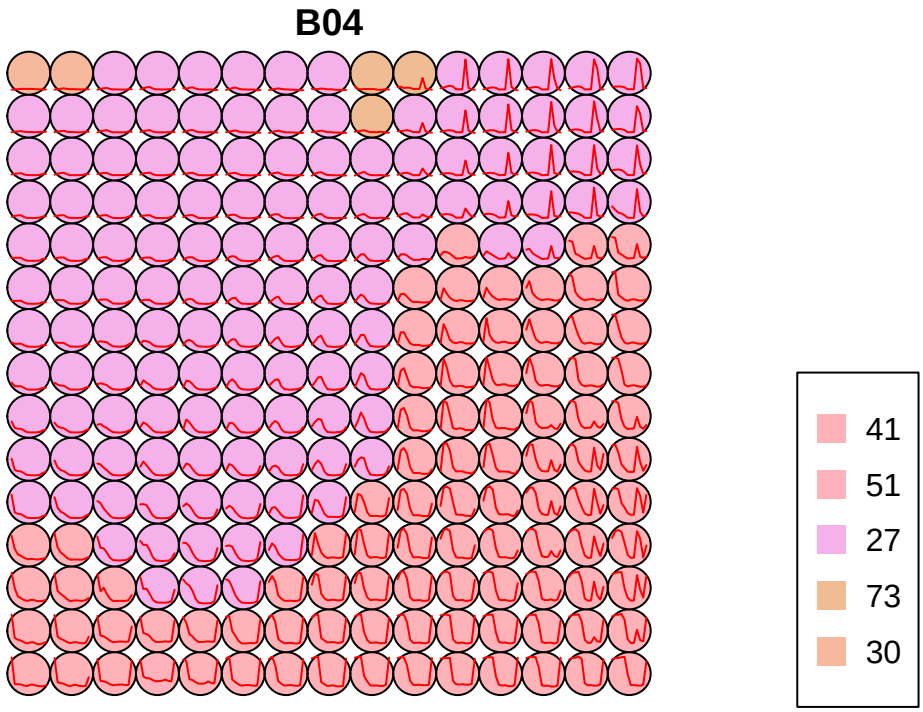
```
plot(samples_SOM_new_cluster, band = "B02")
```



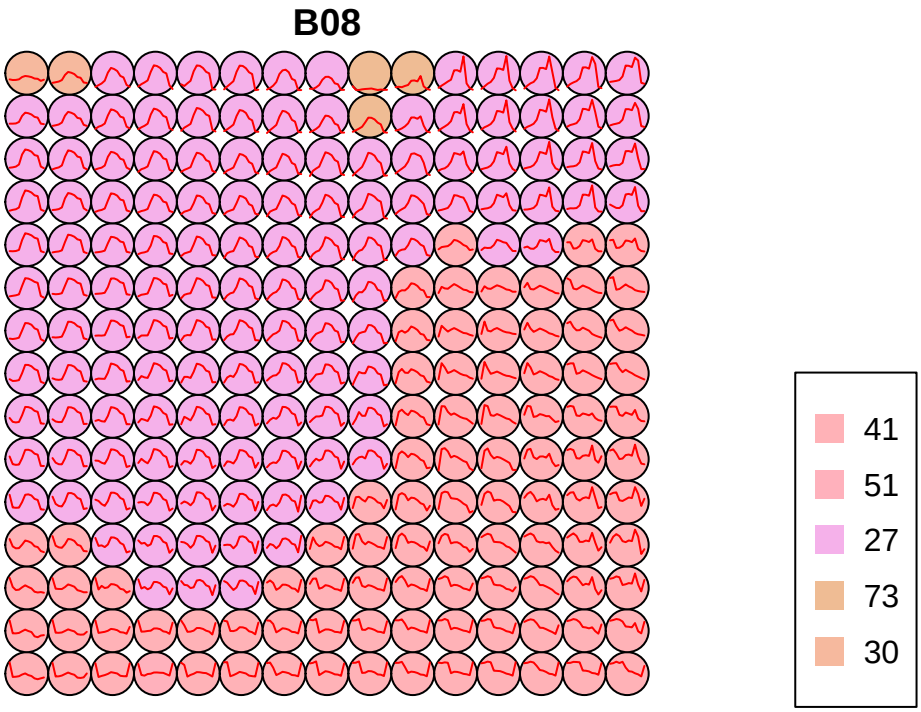

```
plot(samples_SOM_new_cluster, band = "B03")
```



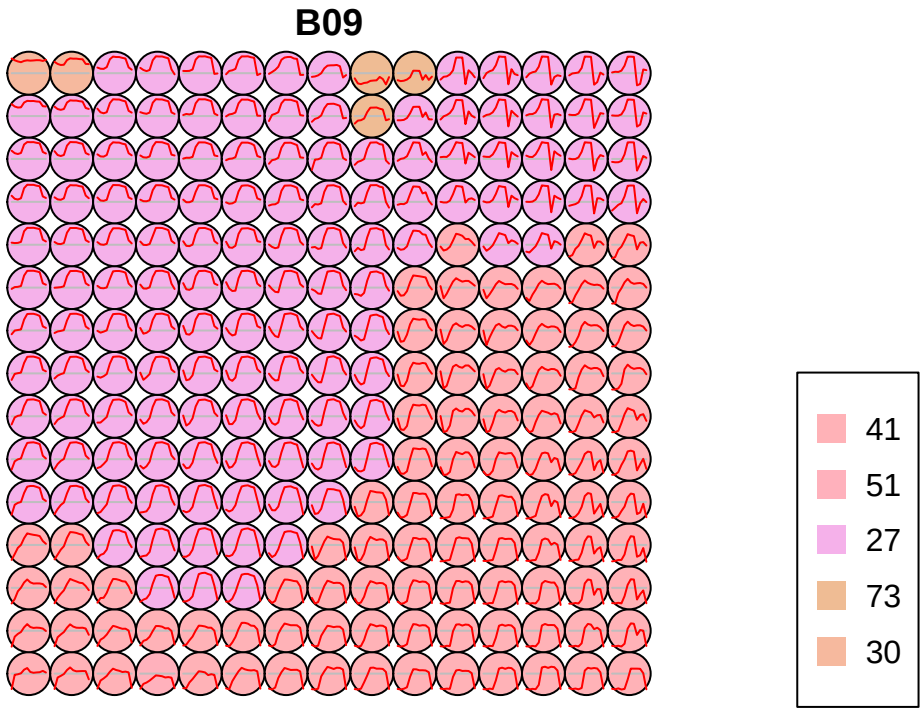

```
plot(samples_SOM_new_cluster, band = "B04")
```



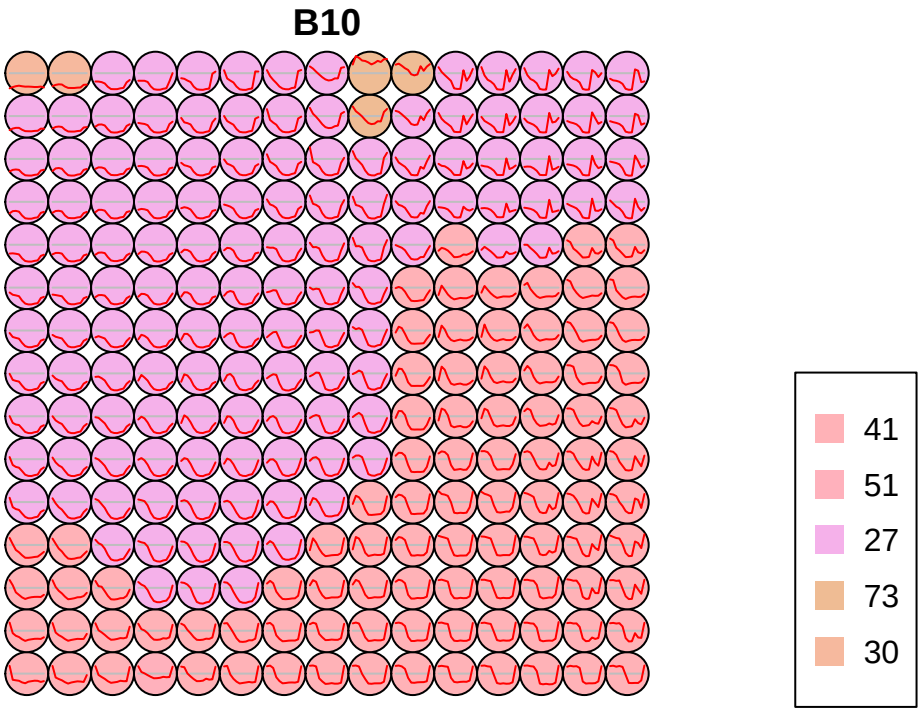

```
plot(samples_SOM_new_cluster, band = "B08")
```




```
plot(samples_SOM_new_cluster, band = "B09")
```




```
plot(samples_SOM_new_cluster, band = "B10")
```



Same problem with colors here.

New evaluation

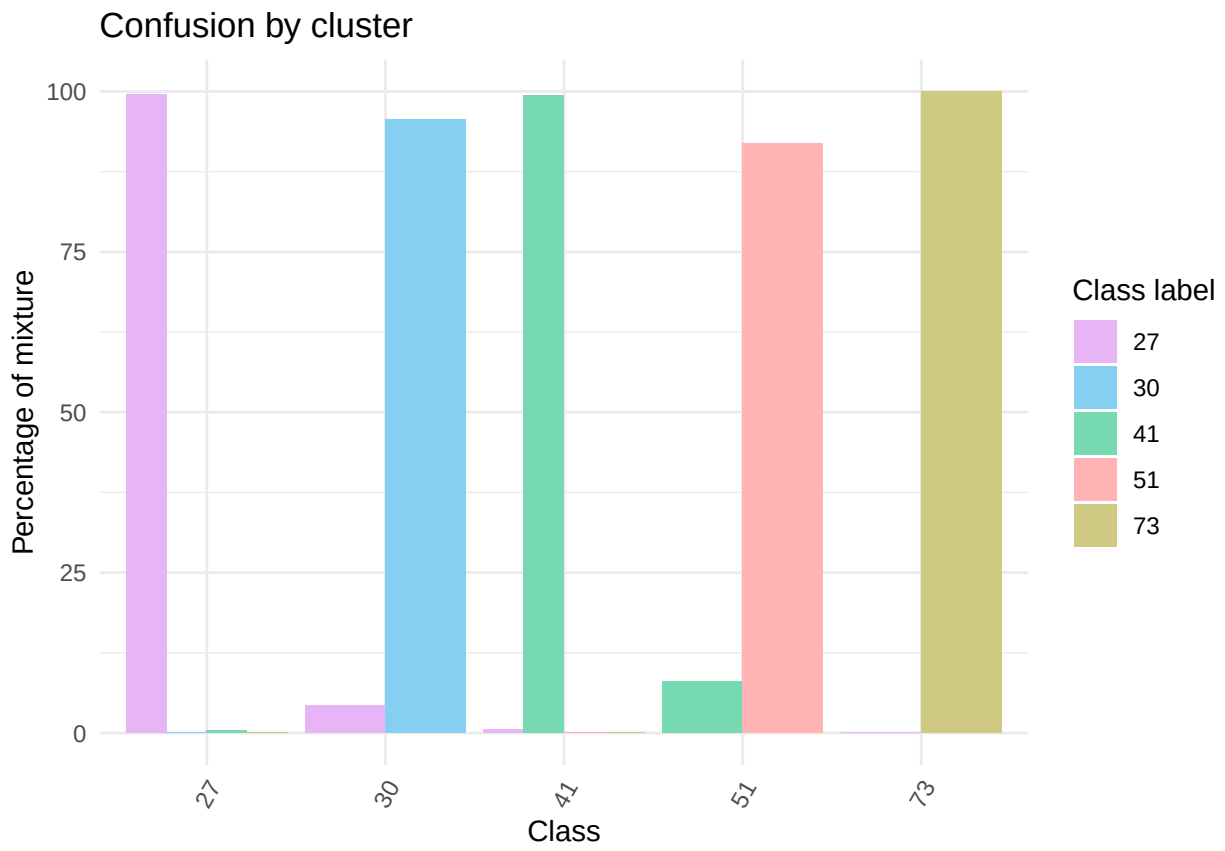
```
new_som_eval <- sits_som_evaluate_cluster(samples_SOM_new_cluster)
```

Plot confusion between clusters.

```
# Plot the confusion between clusters
plot(new_som_eval)
```

```
## Warning in .colors_get(labels = labels, legend = NULL, palette = "Set3", :
## missing colors for labels27, 30, 41, 73, 51
```

```
## Warning in .colors_get(labels = labels, legend = NULL, palette = "Set3", :
## palette for missing colors isSet3
```



5 remaining labels on the plot.

Reduce sample imbalance (RSI)

```
# Aplicamos los parámetros correctos de "máximo" y "mínimo" para reducir el samples.

samples_RSI <- sits_reduce_imbalance(
  samples = samples_organized_ungr,
  n_samples_over = 200, # Changed this value (min count was 214)
  n_samples_under = 2000, # Changed this value
  multicores = 12
)

saveRDS(samples_RSI, file = here("r_objects", "samples_RSI.rds"))
```

Print the balanced samples.

```
summary(samples_RSI)
```

```
## # A tibble: 16 x 3
##    label count    prop
##    <chr> <int>   <dbl>
##  1 1      2116 0.0802
```

```
##  2 14     2100 0.0796
##  3 15      423 0.0160
##  4 18     1936 0.0734
##  5 27     2116 0.0802
##  6 30     2088 0.0792
##  7 40     2096 0.0795
##  8 41     2116 0.0802
##  9 47      214 0.00811
## 10 48     2000 0.0758
## 11 49     2112 0.0801
## 12 51     2116 0.0802
## 13 52     1118 0.0424
## 14 55     2112 0.0801
## 15 63      594 0.0225
## 16 73     1116 0.0423
```

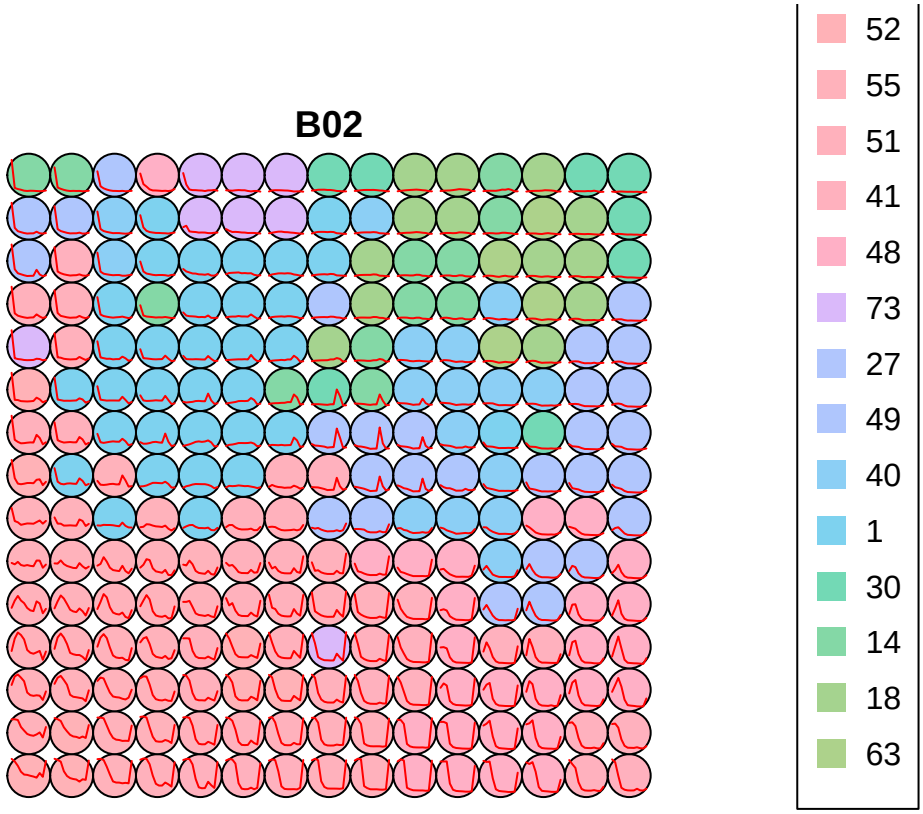
SOM

Clustering time series using SOM.

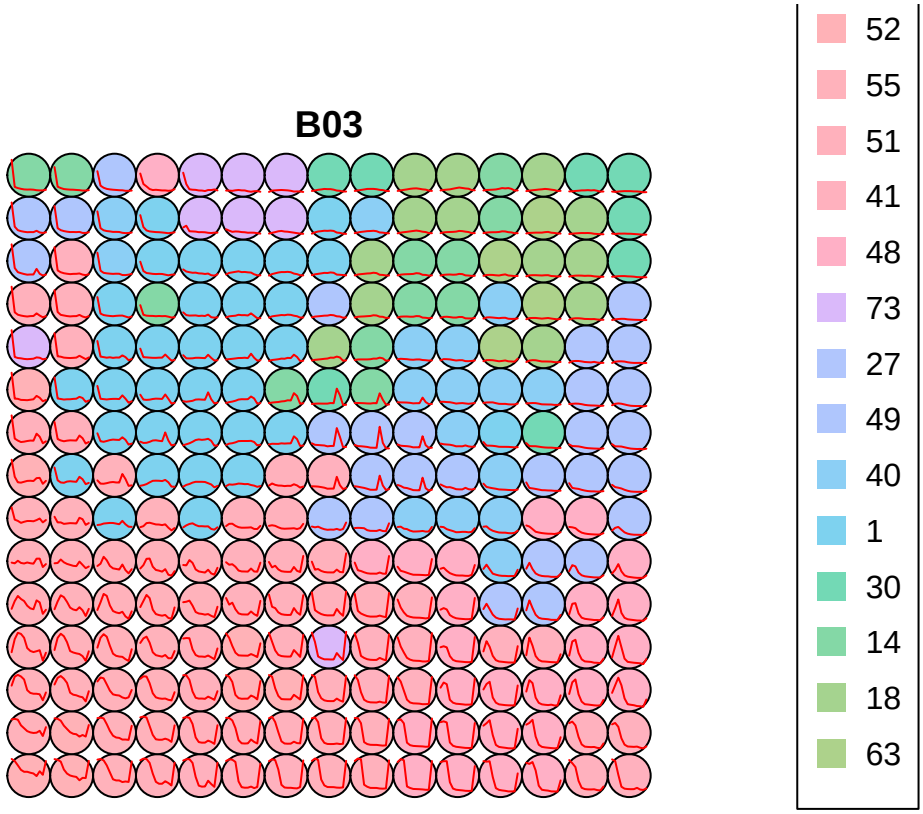
```
som_cluster_bal <- sits_som_map(
  data = samples_RSI,
  grid_xdim = 15,
  grid_ydim = 15,
  alpha = 1.0,
  distance = "dtw",
  rlen = 20,
  mode = "pbatch" # Not sure why this one, from https://e-sensing.github.io/sitsbook/improving-the-qual
  )

saveRDS(som_cluster_bal, file = here("r_objects", "som_cluster_bal.rds"))
```

Plots.

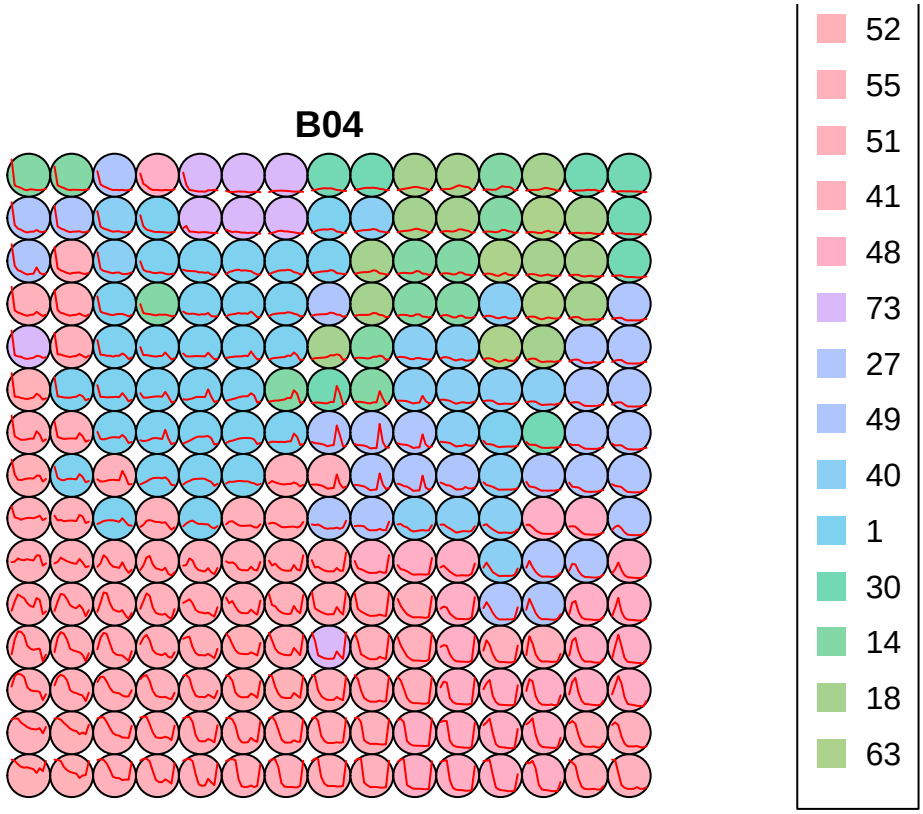
```
plot(som_cluster_bal, band = "B02")
```



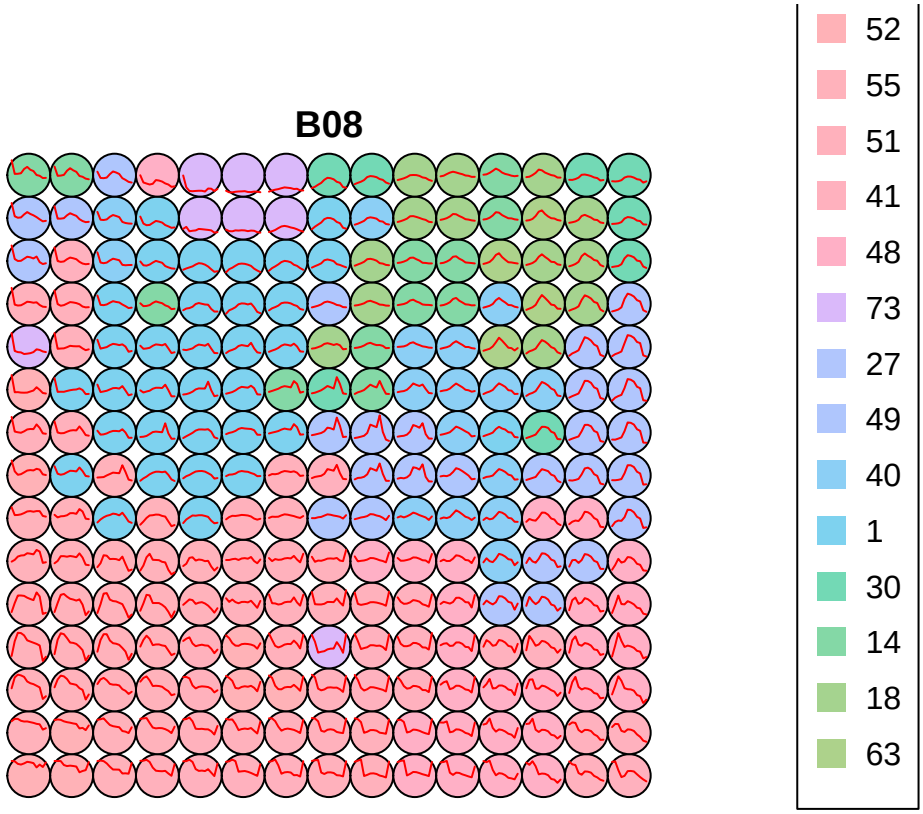

```
plot(som_cluster_bal, band = "B03")
```



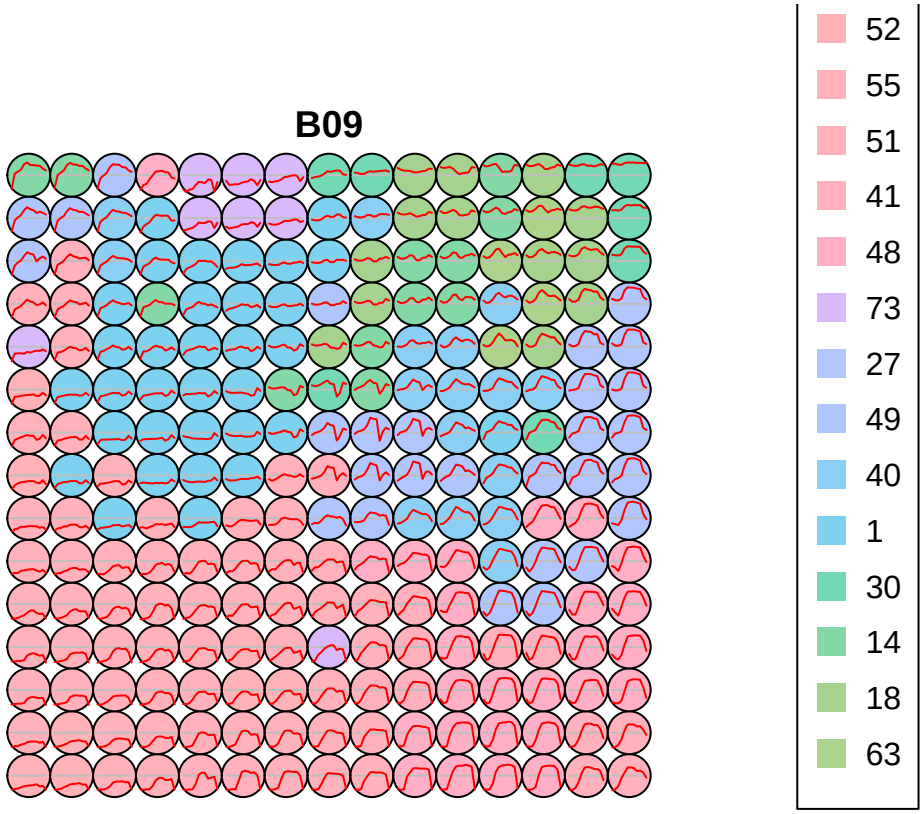

```
plot(som_cluster_bal, band = "B04")
```



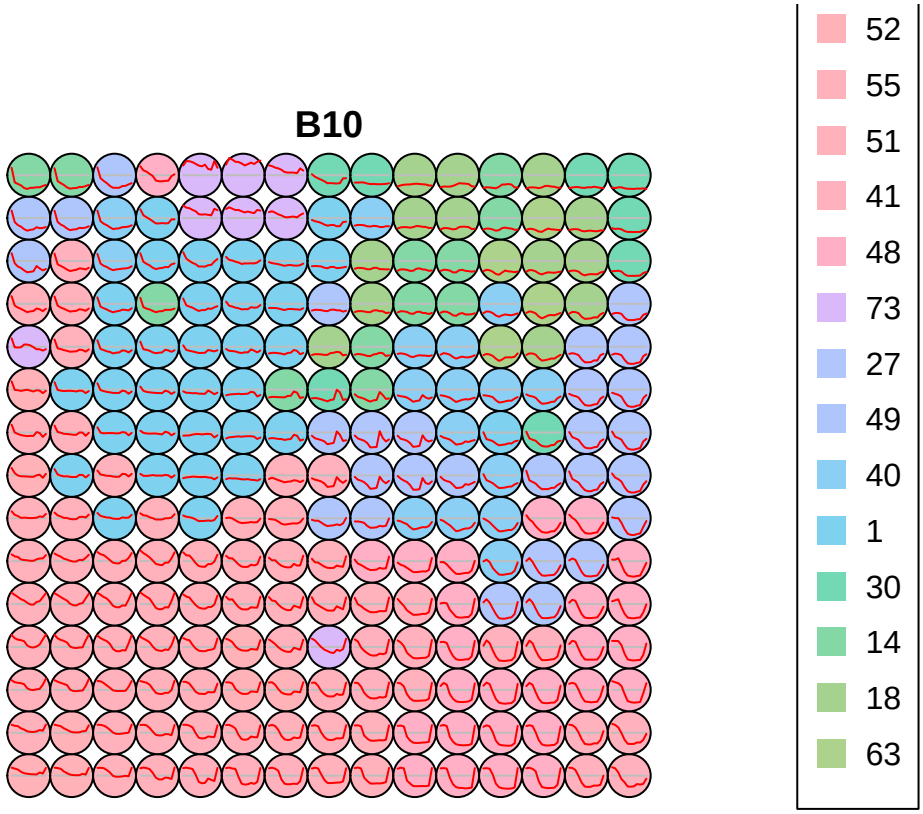

```
plot(som_cluster_bal, band = "B08")
```




```
plot(som_cluster_bal, band = "B09")
```




```
plot(som_cluster_bal, band = "B10")
```



Evaluation

Produce a tibble with a summary of the mixed labels.

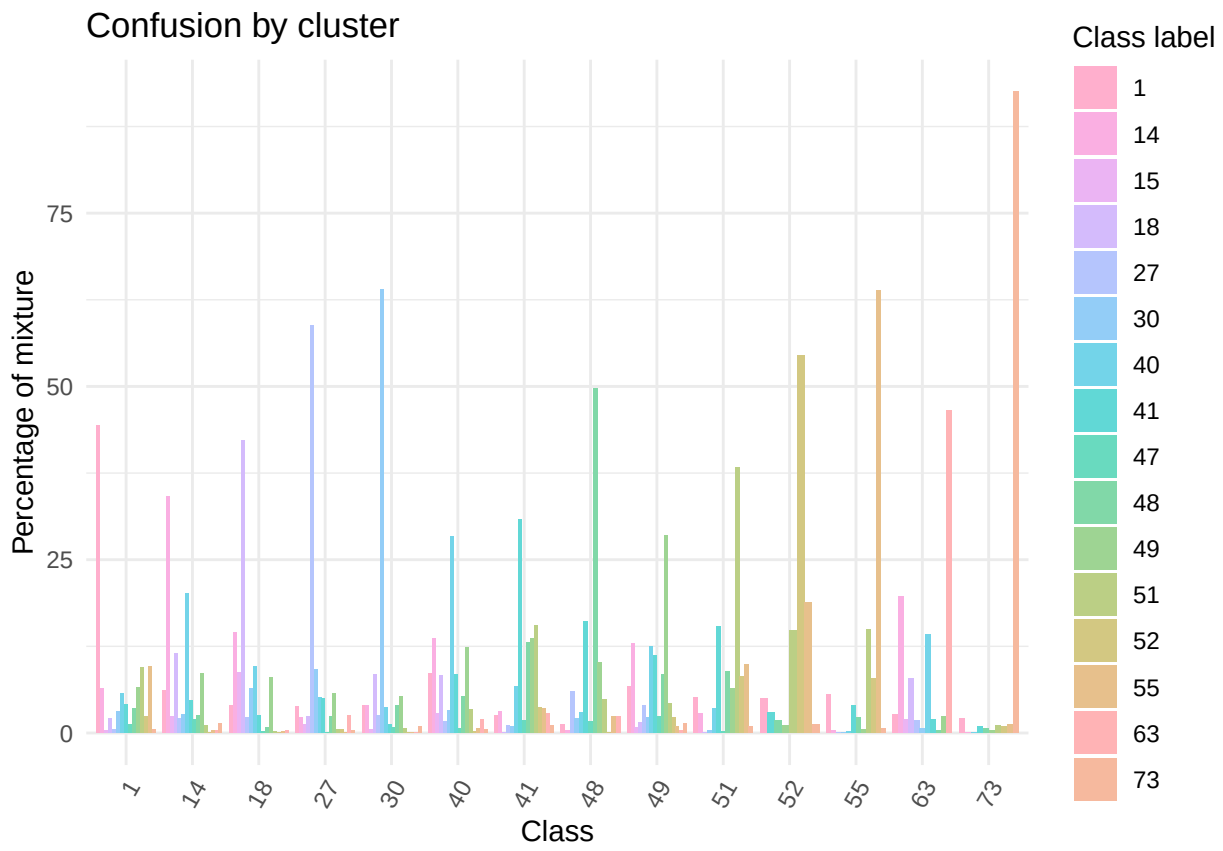
```
som_eval_RSI <- sits_som_evaluate_cluster(som_cluster_bal)
```

Plot confusion between clusters.

```
plot(som_eval_RSI)
```

```
## Warning in .colors_get(labels = labels, legend = NULL, palette = "Set3", :
## missing colors for labels1, 14, 15, 18, 27, 30, 40, 41, 47, 48, 49, 51, 52, 55,
## 73, 63
```

```
## Warning in .colors_get(labels = labels, legend = NULL, palette = "Set3", :
## palette for missing colors isSet3
```



Again warning about colors and missing labels on the graph.

Session info

```
sessionInfo()
```

```
## R version 4.4.2 (2024-10-31 ucrt)
## Platform: x86_64-w64-mingw32/x64
## Running under: Windows 11 x64 (build 22631)
##
## Matrix products: default
##
##
## locale:
## [1] LC_COLLATE=English_United States.utf8
## [2] LC_CTYPE=English_United States.utf8
## [3] LC_MONETARY=English_United States.utf8
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.utf8
##
## time zone: Europe/Madrid
## tzcode source: internal
##
```

```
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] here_1.0.1     kohonen_3.0.12 sits_1.5.2     sf_1.0-19      tidyr_1.3.1
## [6] purrr_1.0.4    tibble_3.2.1   dplyr_1.1.4
##
## loaded via a namespace (and not attached):
##  [1] utf8_1.2.4              generics_0.1.3          class_7.3-23
##  [4] KernSmooth_2.23-26      digest_0.6.37           magrittr_2.0.3
##  [7] evaluate_1.0.3          grid_4.4.2              timechange_0.3.0
## [10] fastmap_1.2.0           rprojroot_2.0.4         e1071_1.7-16
## [13] DBI_1.2.3               crosstalk_1.2.1         scales_1.3.0
## [16] jquerylib_0.1.4         cli_3.6.3               rlang_1.1.5
## [19] units_0.8-5             munsell_0.5.1           withr_3.0.2
## [22] yaml_2.3.10             tools_4.4.2             colorspace_2.1-1
## [25] ggplot2_3.5.1           vctrs_0.6.5             R6_2.6.1
## [28] proxy_0.4-27            lifecycle_1.0.4         lubridate_1.9.4
## [31] classInt_0.4-11         leaflet_2.2.2           leaflet.providers_2.0.0
## [34] htmlwidgets_1.6.4       pkgconfig_2.0.3         pillar_1.10.1
## [37] gtable_0.3.6            glue_1.8.0              Rcpp_1.0.14
## [40] xfun_0.50               tidyselect_1.2.1        rstudioapi_0.17.1
## [43] knitr_1.49              farver_2.1.2            htmltools_0.5.8.1
## [46] labeling_0.4.3          rmarkdown_2.29          compiler_4.4.2
```