Script to validate points in ReSurvey database using RS data (Satellite Embeddings Dataset)

Validation done with ALL points (all observations)

Alicia Valdés

30 septiembre 2025

This R script is used to validate the points in the ReSurvey database using only bands from the Google Satellite Embedding dataset: https://developers.google.com/earth-engine/datasets/catalog/GOOGLE_SATELLITE_EMBEDDING_V1_ANNUAL#description

Load libraries

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.2
## v ggplot2   4.0.0     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.1.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(here)
```

```
## here() starts at C:/Users/jimenezalfaro/OneDrive - Universidad de Oviedo/IMIB/Analyses/MOTIVATE/MOTIV
```

```
library(gridExtra)
```

```
##
## Adjuntando el paquete: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(readxl)
library(scales)
```

```
##
## Adjuntando el paquete: 'scales'
##
## The following object is masked from 'package:purrr':
##
##     discard
##
## The following object is masked from 'package:readr':
##
##     col_factor
```

```r
library(sf)
```

```
## Linking to GEOS 3.13.1, GDAL 3.11.0, PROJ 9.6.0; sf_use_s2() is TRUE
```

```r
library(rnaturalearth)
library(dtplyr)
library(lme4)
```

```
## Cargando paquete requerido: Matrix
##
## Adjuntando el paquete: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```r
library(lmerTest)
```

```
##
## Adjuntando el paquete: 'lmerTest'
##
## The following object is masked from 'package:lme4':
##
##     lmer
##
## The following object is masked from 'package:stats':
##
##     step
```

```r
library(car)
```

```
## Cargando paquete requerido: carData
##
## Adjuntando el paquete: 'car'
##
## The following object is masked from 'package:dplyr':
```

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

```
library(ggeffects)
library(party)
```

```
## Cargando paquete requerido: grid
## Cargando paquete requerido: mvtnorm
## Cargando paquete requerido: modeltools
## Cargando paquete requerido: stats4
##
## Adjuntando el paquete: 'modeltools'
##
## The following object is masked from 'package:car':
##
##     Predict
##
## The following object is masked from 'package:lme4':
##
##     refit
##
## Cargando paquete requerido: strucchange
## Cargando paquete requerido: zoo
##
## Adjuntando el paquete: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
## Cargando paquete requerido: sandwich
##
## Adjuntando el paquete: 'strucchange'
##
## The following object is masked from 'package:stringr':
##
##     boundary
##
##
## Adjuntando el paquete: 'party'
##
## The following object is masked from 'package:dplyr':
##
##     where
```

```
library(partykit)
```

```
## Cargando paquete requerido: libcoin
##
```

```
## Adjuntando el paquete: 'partykit'
##
## The following objects are masked from 'package:party':
##
##     cforest, ctree, ctree_control, edge_simple, mob, mob_control,
##     node_barplot, node_bivplot, node_boxplot, node_inner, node_surv,
##     node_terminal, varimp
```

```
library(moreparty)
library(doParallel)
```

```
## Cargando paquete requerido: foreach
##
## Adjuntando el paquete: 'foreach'
##
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
##
## Cargando paquete requerido: iterators
## Cargando paquete requerido: parallel
```

```
library(strucchange)
library(ggparty)
```

```
##
## Adjuntando el paquete: 'ggparty'
##
## The following object is masked from 'package:ggeffects':
##
##     get_predictions
```

```
library(caret)
```

```
## Cargando paquete requerido: lattice
##
## Adjuntando el paquete: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(moreparty)
library(randomForest)
```

```
## randomForest 4.7-1.2
## Type rfNews() to see new features/changes/bug fixes.
##
## Adjuntando el paquete: 'randomForest'
##
## The following object is masked from 'package:gridExtra':
```

```
##
##     combine
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
##
## Adjuntando el paquete: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(corrplot)
```

```
## corrplot 0.95 loaded
```

```
library(rlang)
```

```
##
## Adjuntando el paquete: 'rlang'
##
## The following objects are masked from 'package:purrr':
##
##     %@%, flatten, flatten_chr, flatten_dbl, flatten_int, flatten_lgl,
##     flatten_raw, invoke, splice
```

```
library(stringr)
library(beepr)
library(foreach)
library(permimp)
library(yardstick)
```

```
##
## Adjuntando el paquete: 'yardstick'
##
## The following objects are masked from 'package:caret':
##
##     precision, recall, sensitivity, specificity
##
## The following object is masked from 'package:readr':
##
##     spec
```

Define printall function

```
printall <- function(tibble) {
  print(tibble, width = Inf)
  }
```

Load geom_flat_violin plot

```
source("https://gist.githubusercontent.com/benmarwick/2a1bb0133ff568cbe28d/raw/fb53bd97121f7f9ce947837e
```

Load previously created objects

```
# Define the folder path
folder_path <- here("objects", "RF", "Satellite_Embeddings")

# List all .RData or .rda files in the folder
rdata_files <- list.files(folder_path, full.names = TRUE)

# Load each file
lapply(rdata_files, load, envir = .GlobalEnv)
```

```
## [[1]]
## [1] "rf1"
##
## [[2]]
## [1] "predictions_rf1"
##
## [[3]]
## [1] "varimp_rf1"
##
## [[4]]
## [1] "rf10"
##
## [[5]]
## [1] "predictions_rf10"
##
## [[6]]
## [1] "rf11"
##
## [[7]]
## [1] "predictions_rf11"
##
## [[8]]
## [1] "rf12"
##
## [[9]]
```

```
## [1] "predictions_rf12"
##
## [[10]]
## [1] "rf13"
##
## [[11]]
## [1] "predictions_rf13"
##
## [[12]]
## [1] "rf14"
##
## [[13]]
## [1] "predictions_rf14"
##
## [[14]]
## [1] "rf15"
##
## [[15]]
## [1] "predictions_rf15"
##
## [[16]]
## [1] "rf16"
##
## [[17]]
## [1] "predictions_rf16"
##
## [[18]]
## [1] "rf17"
##
## [[19]]
## [1] "predictions_rf17"
##
## [[20]]
## [1] "rf18"
##
## [[21]]
## [1] "predictions_rf18"
##
## [[22]]
## [1] "rf19"
##
## [[23]]
## [1] "predictions_rf19"
##
## [[24]]
## [1] "rf2"
##
## [[25]]
## [1] "predictions_rf2"
##
## [[26]]
## [1] "varimp_rf2"
##
## [[27]]
```

```
## [1] "rf20"
##
## [[28]]
## [1] "predictions_rf20"
##
## [[29]]
## [1] "rf3"
##
## [[30]]
## [1] "predictions_rf3"
##
## [[31]]
## [1] "varimp_rf3"
##
## [[32]]
## [1] "rf4"
##
## [[33]]
## [1] "predictions_rf4"
##
## [[34]]
## [1] "varimp_rf4"
##
## [[35]]
## [1] "rf5"
##
## [[36]]
## [1] "predictions_rf5"
##
## [[37]]
## [1] "rf6"
##
## [[38]]
## [1] "predictions_rf6"
##
## [[39]]
## [1] "rf7"
##
## [[40]]
## [1] "predictions_rf7"
##
## [[41]]
## [1] "rf8"
##
## [[42]]
## [1] "predictions_rf8"
##
## [[43]]
## [1] "rf9"
##
## [[44]]
## [1] "predictions_rf9"
```

Read data

```
data_validation <- read_tsv(here(
  "data", "clean","final_RS_data_SatEmb_20250922.csv"))
```

```
## Rows: 18048 Columns: 83
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (13): system:index, Lctnmth, RS_CODE, RSrvypl, RSrvyst, UNIT, .geo, biog...
## dbl (70): A00, A01, A02, A03, A04, A05, A06, A07, A08, A09, A10, A11, A12, A...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

No parsing issues!

Some data managenemt

TO-DO: Missing data checks

Do when all RS data is ready!

Distributions all bioregions

```
# Define a function to create histograms
plot_histogram <- function(data, x_var, x_label) {
  ggplot(data %>%
           dplyr::filter(EUNISa_1 %in% c("T", "R", "S", "Q")),
         aes(x = !!sym(x_var))) +
    geom_histogram(color = "black", fill = "white") +
    labs(x = x_label, y = "Frequency") +
    theme_bw()
}
```

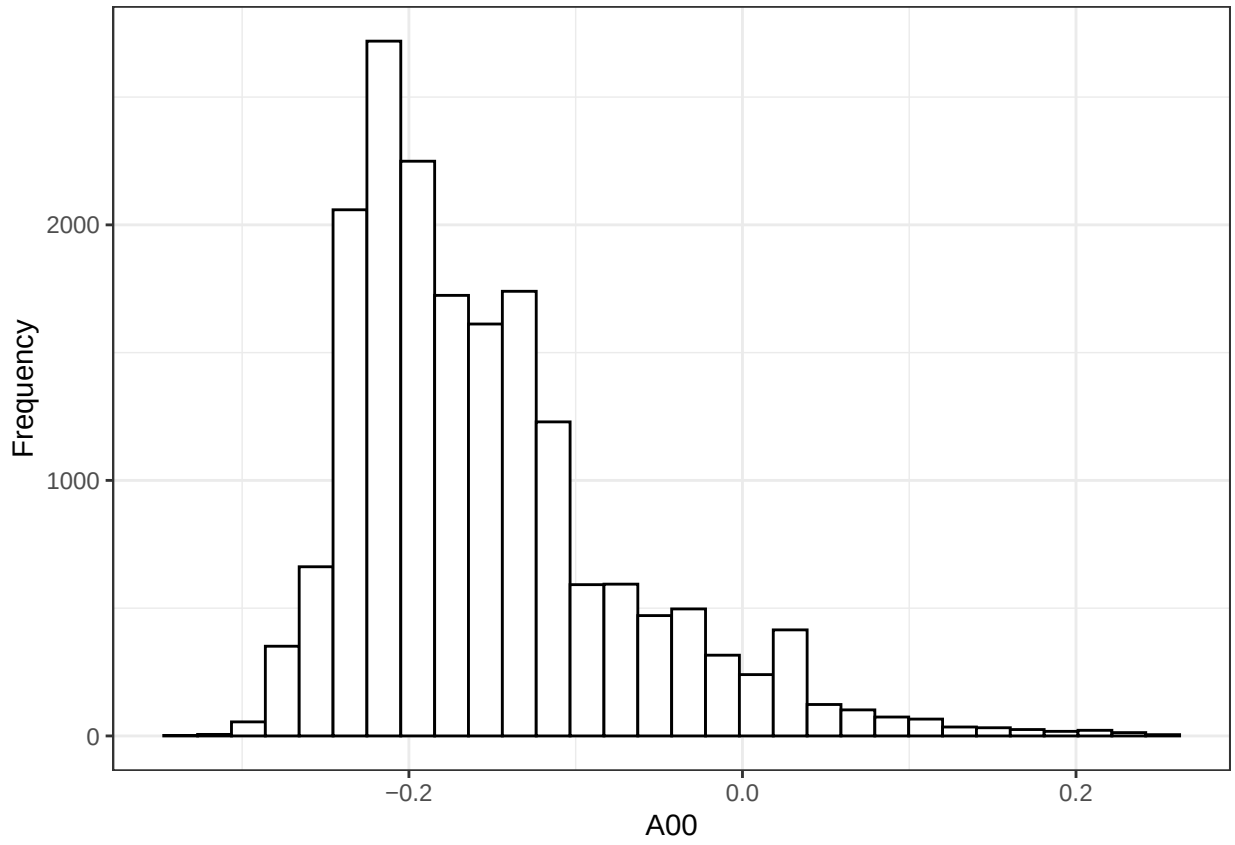
```
# Define a function to create plots with violin + boxplot + points
distr_plot <- function(data, y_vars, y_labels) {
  for (i in seq_along(y_vars)) {
    y_var <- y_vars[[i]]
    y_label <- y_labels[[i]]

    p <- ggplot(data = data %>%
                  dplyr::filter(EUNISa_1 %in% c("T", "R", "S", "Q")),
                aes(x = EUNISa_1_descr, y = !!sym(y_var), fill = EUNISa_1_descr)) +
      geom_flat_violin(position = position_nudge(x = 0.2, y = 0), alpha = 0.8) +
      geom_point(aes(y = !!sym(y_var), color = EUNISa_1_descr),
                 position = position_jitter(width = 0.15), size = 1, alpha = 0.25) +
      geom_boxplot(width = 0.2, outlier.shape = NA, alpha = 0.5) +
```

```
      stat_summary(fun.y = mean, geom = "point", shape = 20, size = 1) +
      stat_summary(fun.data = function(x) data.frame(y = max(x) + 0.1,
                                                     label = length(x)),
                   geom = "text", aes(label = ..label..), vjust = 0.5) +
      labs(y = y_label, x = "EUNIS level 1") +
      scale_x_discrete(labels = function(x) str_wrap(x, width = 15)) +
      guides(fill = FALSE, color = FALSE) +
      theme_bw() + coord_flip()

    print(p)
  }
}
```

Histograms to check that max and min values are ok:

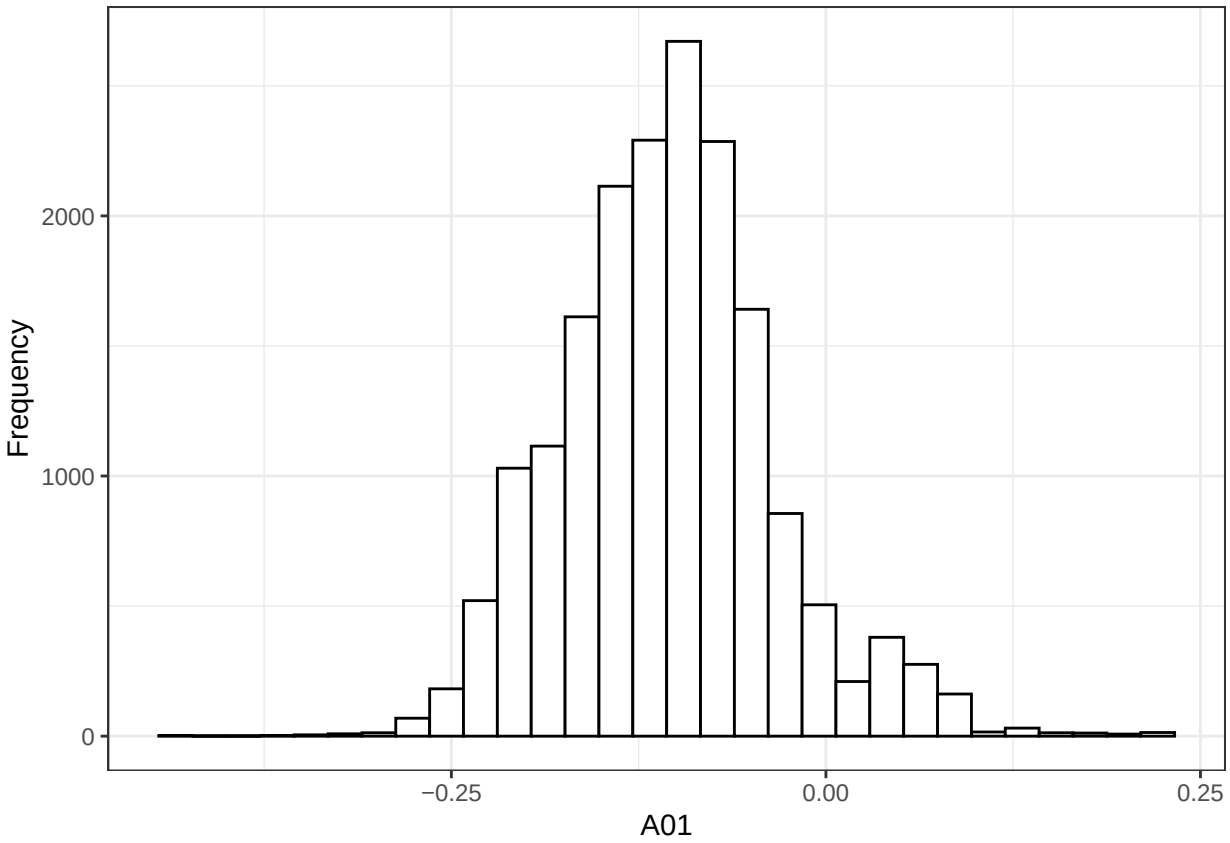
```
plot_histogram(data_validation, "A00", "A00")
```

```
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



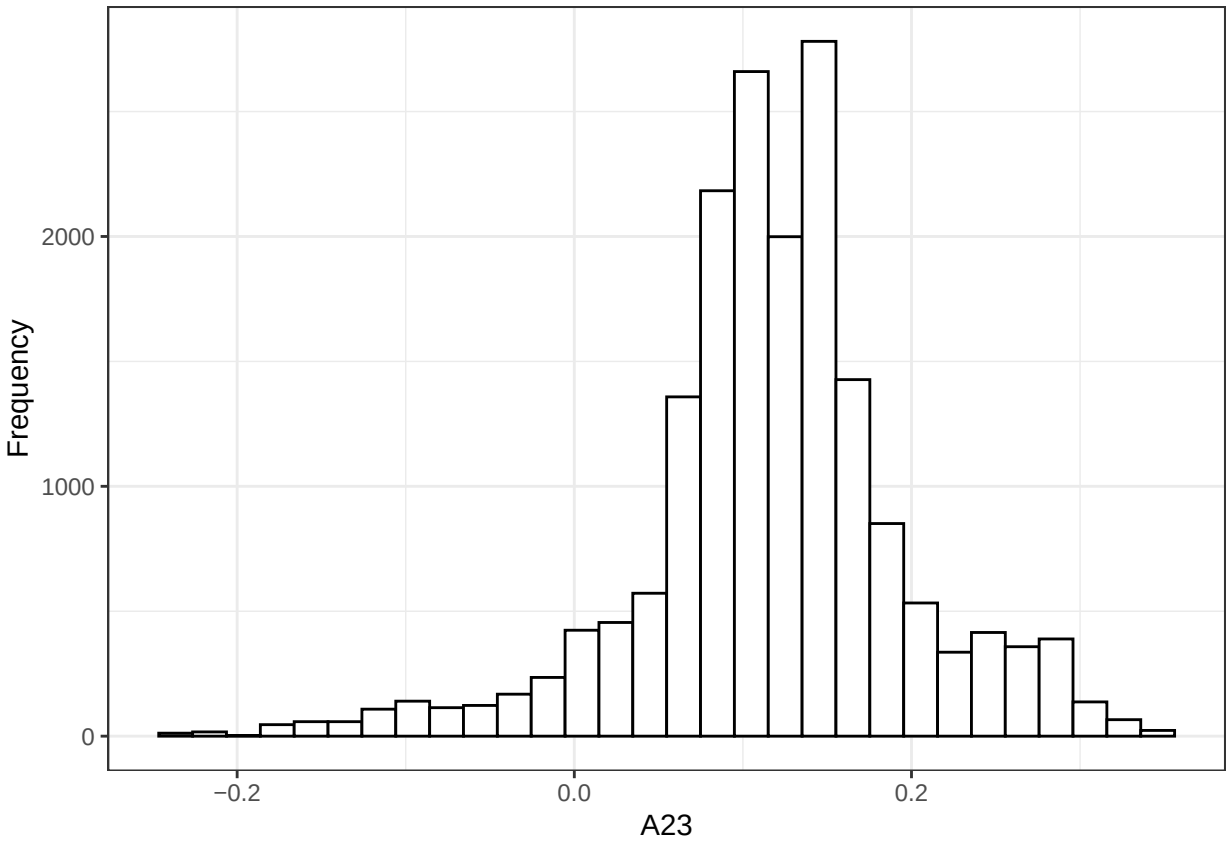

```
plot_histogram(data_validation, "A01", "A01")
```

```
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



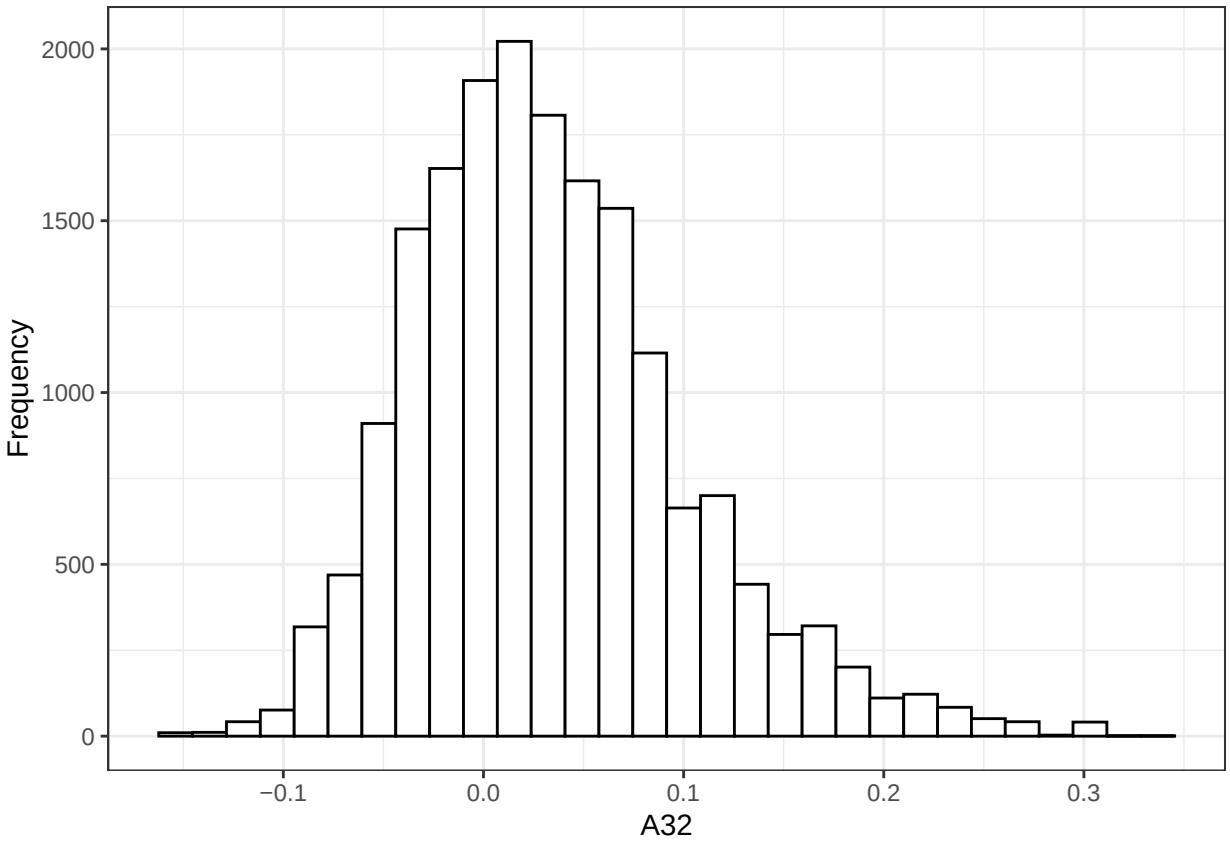

```
plot_histogram(data_validation, "A23", "A23")
```

```
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



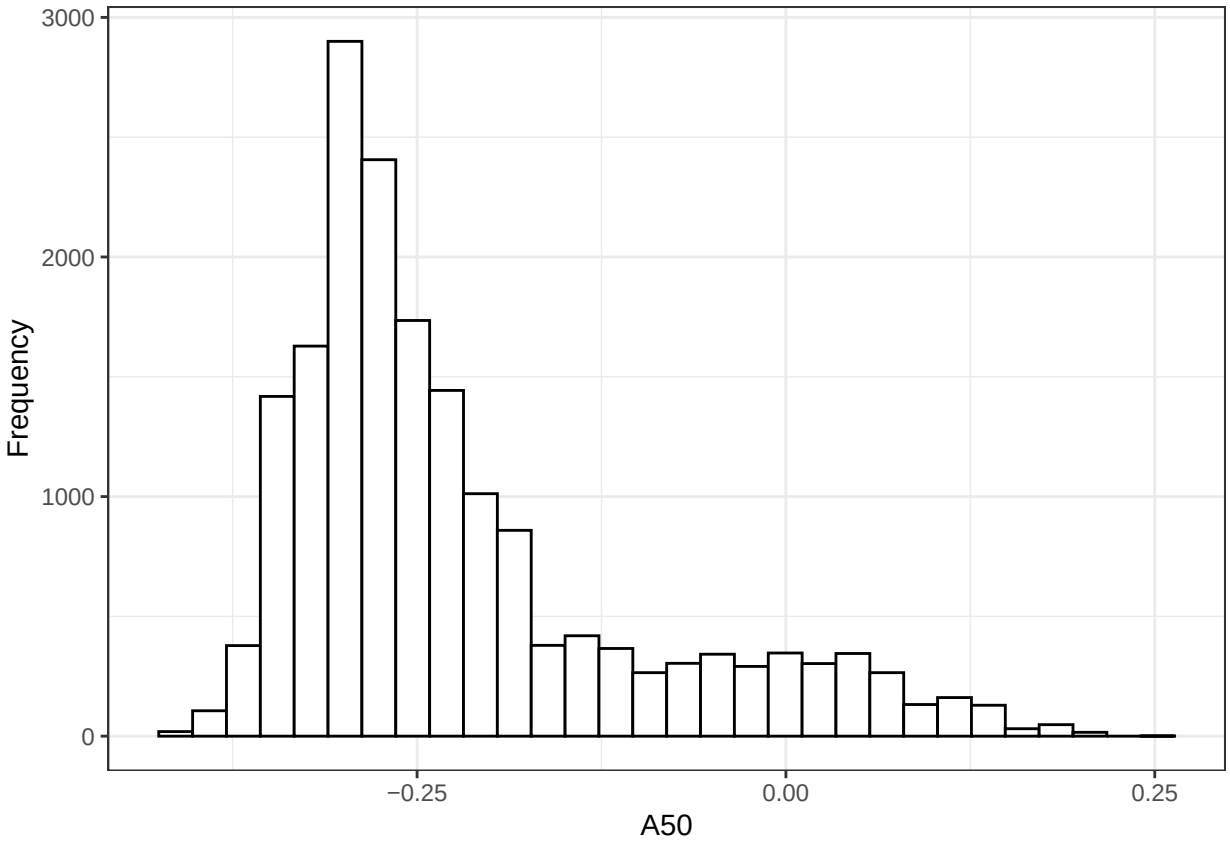

```
plot_histogram(data_validation, "A32", "A32")
```

```
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



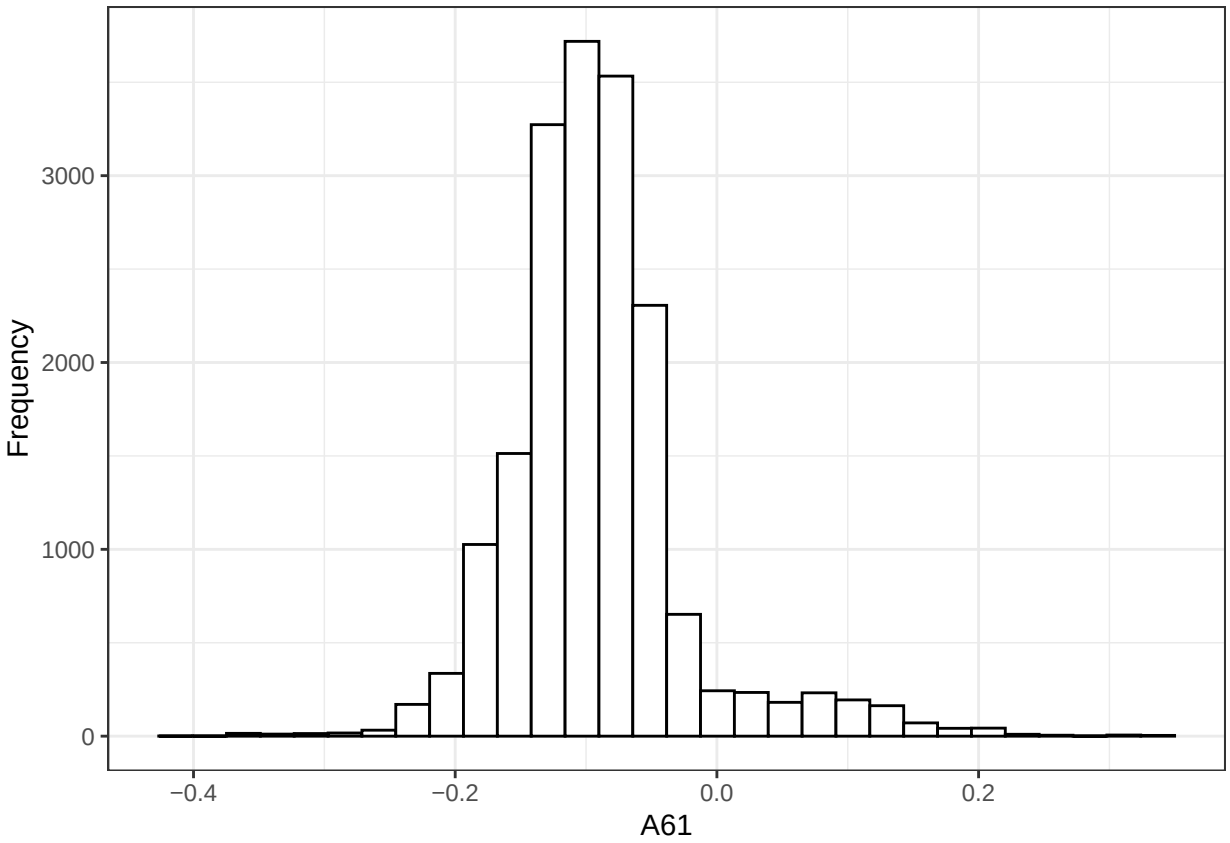

```
plot_histogram(data_validation, "A50", "A50")
```

```
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



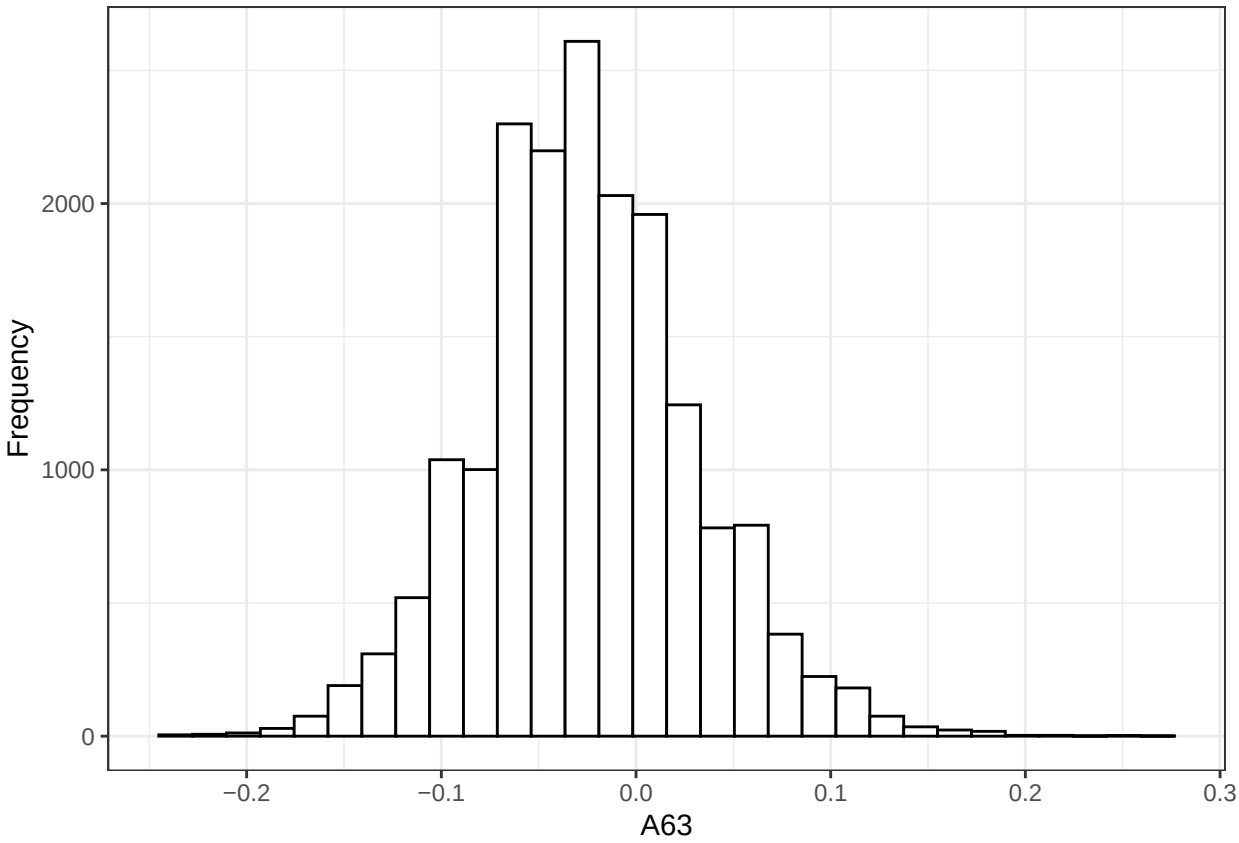

```
plot_histogram(data_validation, "A61", "A61")
```

```
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



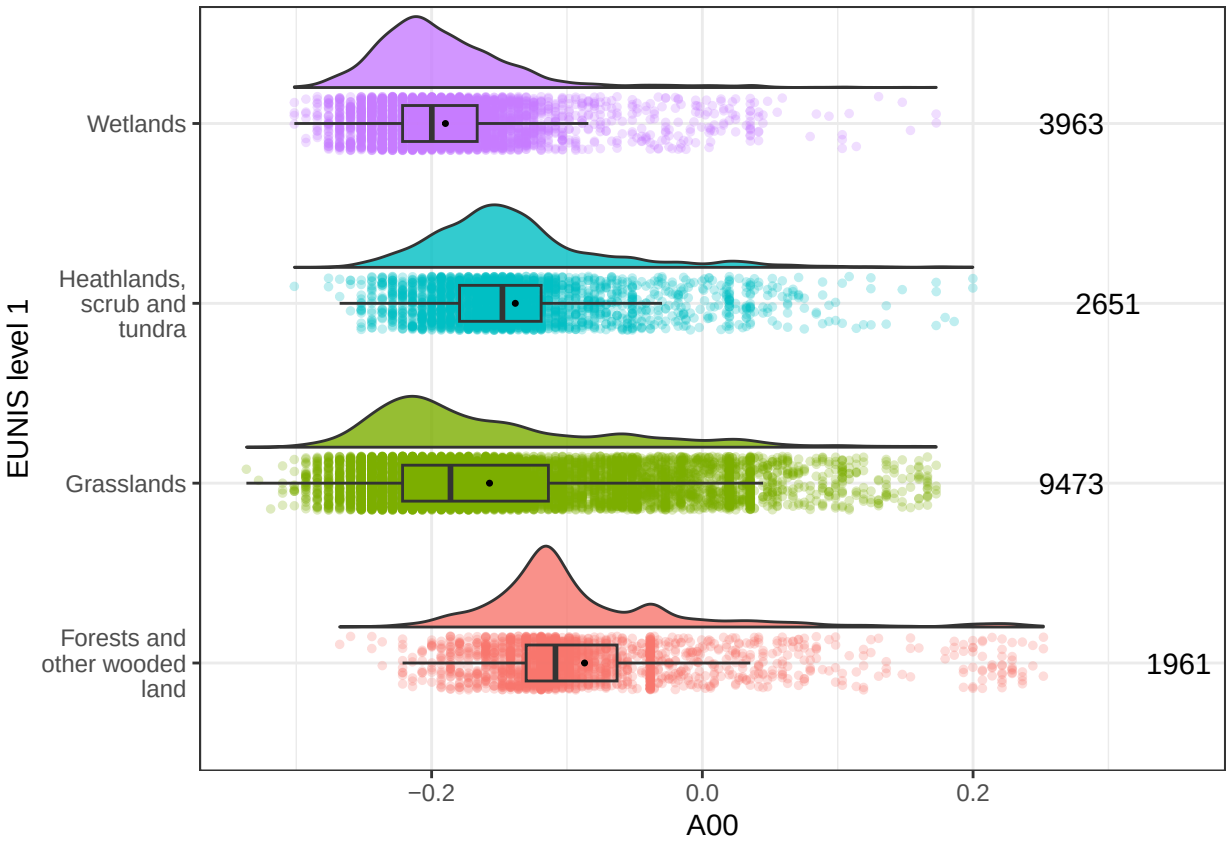

```
plot_histogram(data_validation, "A63", "A63")
```

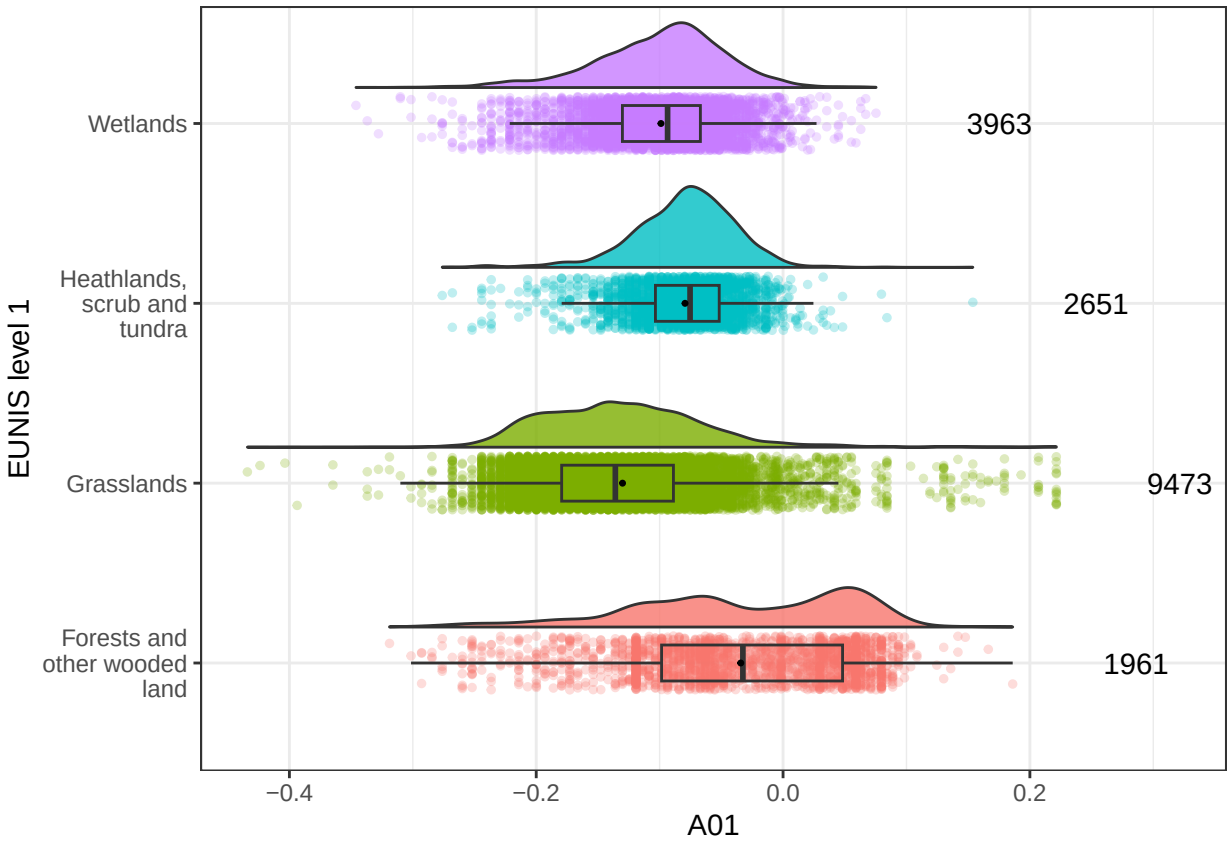
```
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```

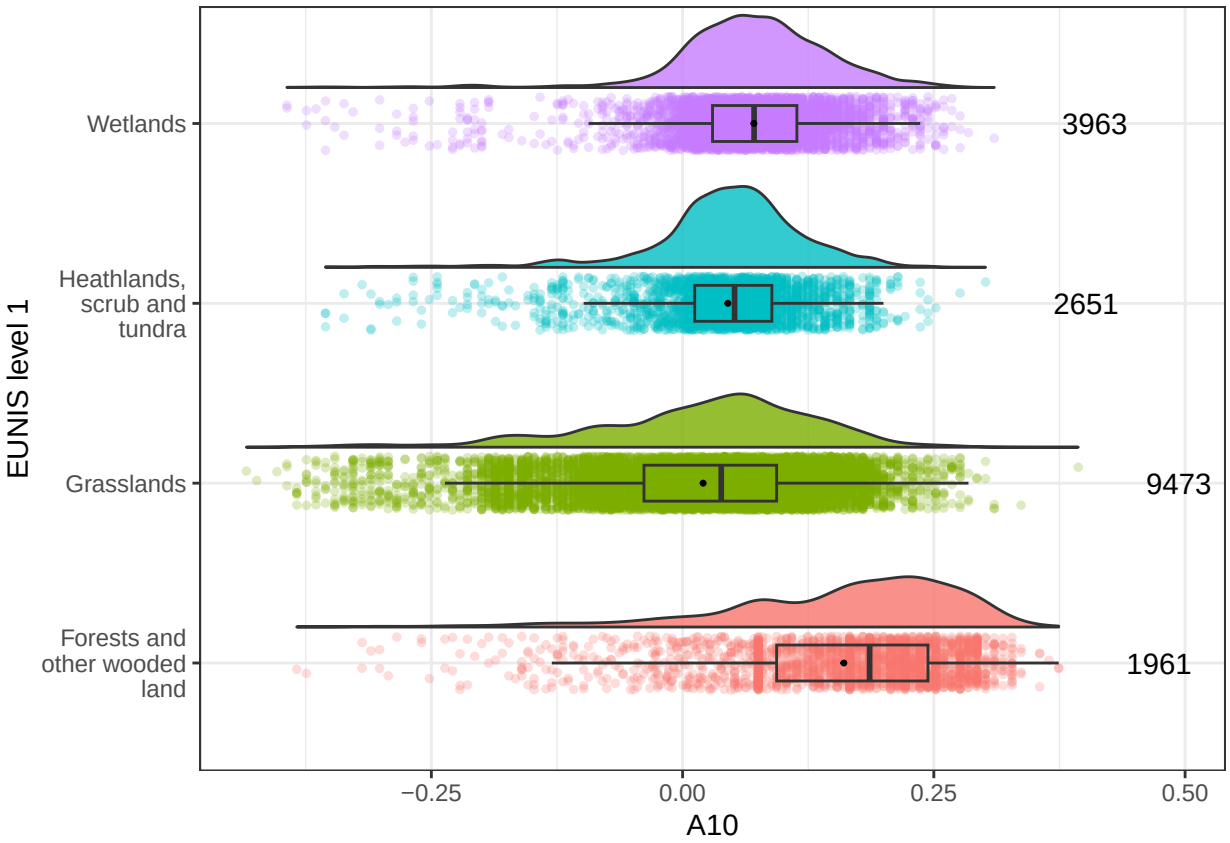


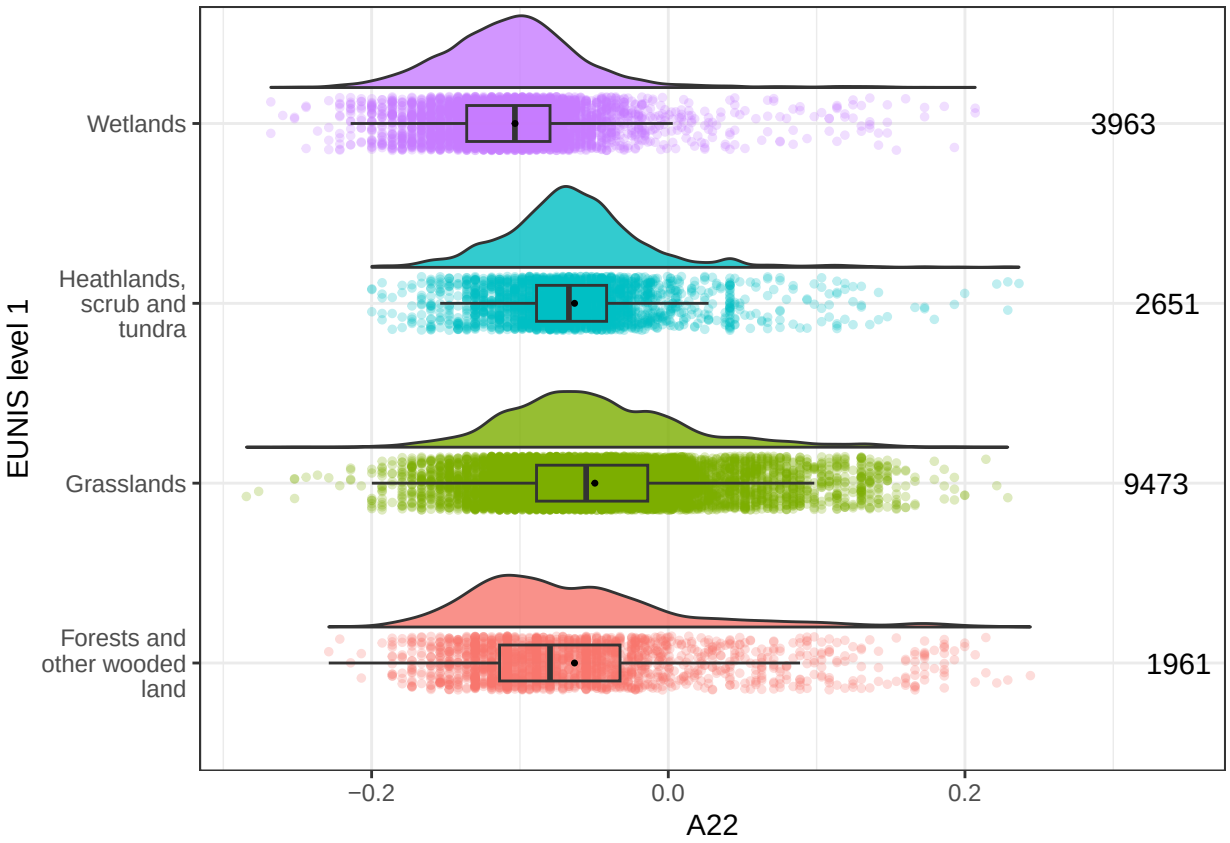
Distribution plots:

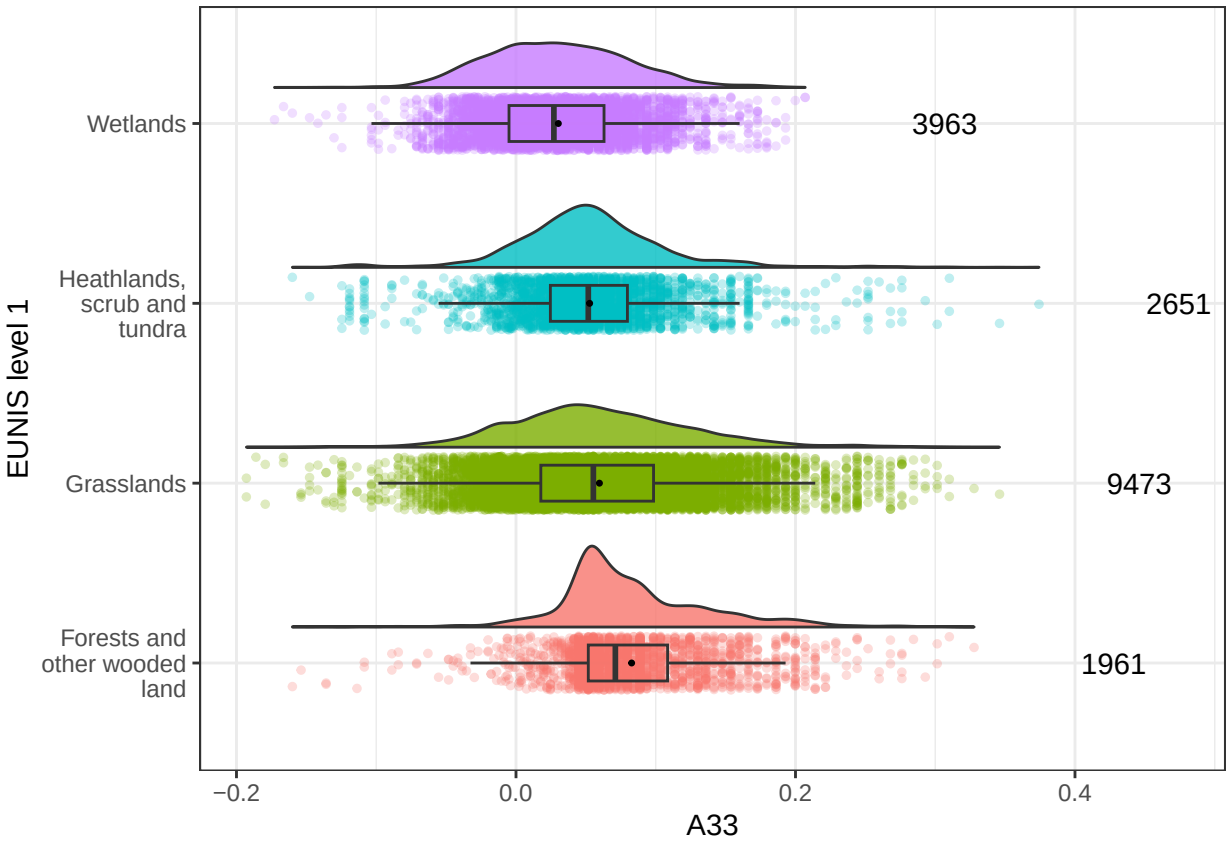
```
distr_plot(data_validation,
           c("A00", "A01", "A10", "A22", "A33",
             "A40", "A53", "A61", "A62", "A63"),
           c("A00", "A01", "A10", "A22", "A33",
             "A40", "A53", "A61", "A62", "A63"))
```

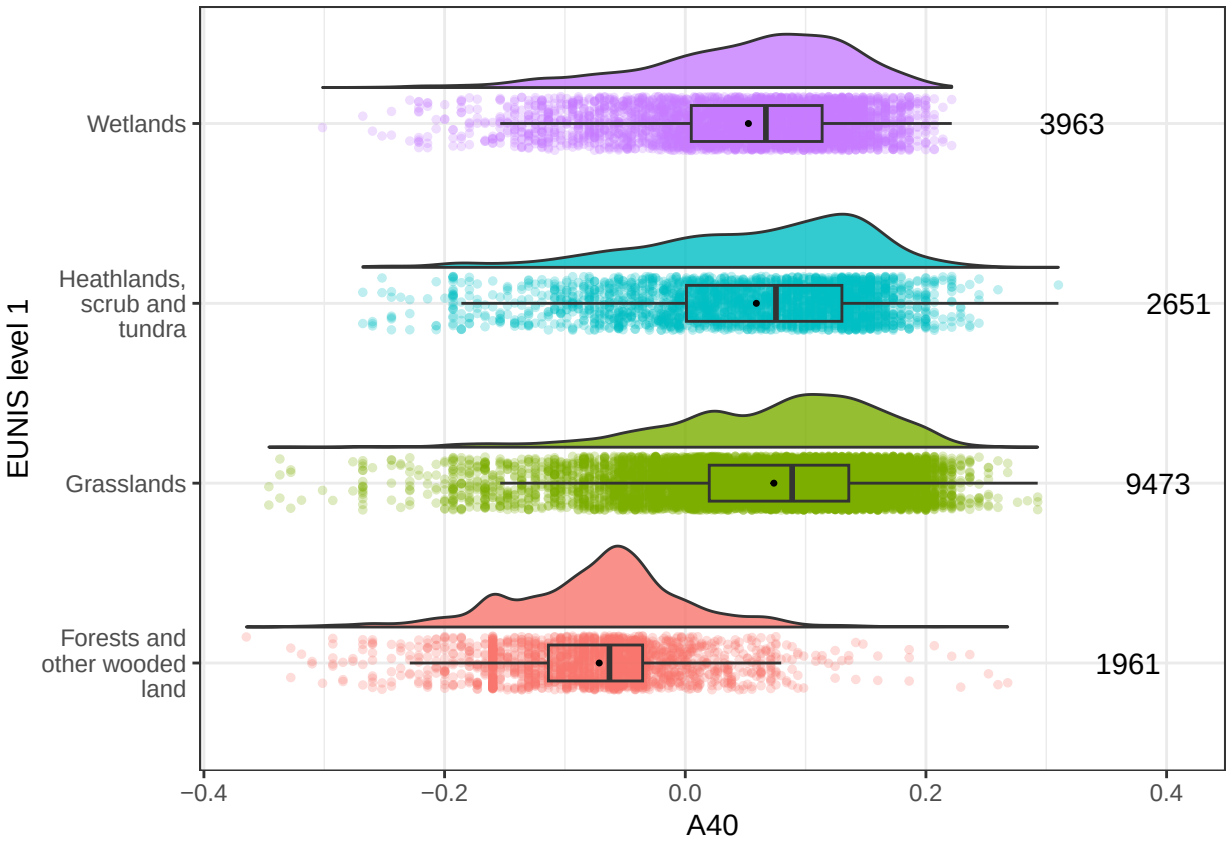

EUNIS level 1
Wetlands
Heathlands, scrub and tundra
Grasslands
Forests and other wooded land
3963
2651
9473
1961
−0.2
0.0
0.2
A00

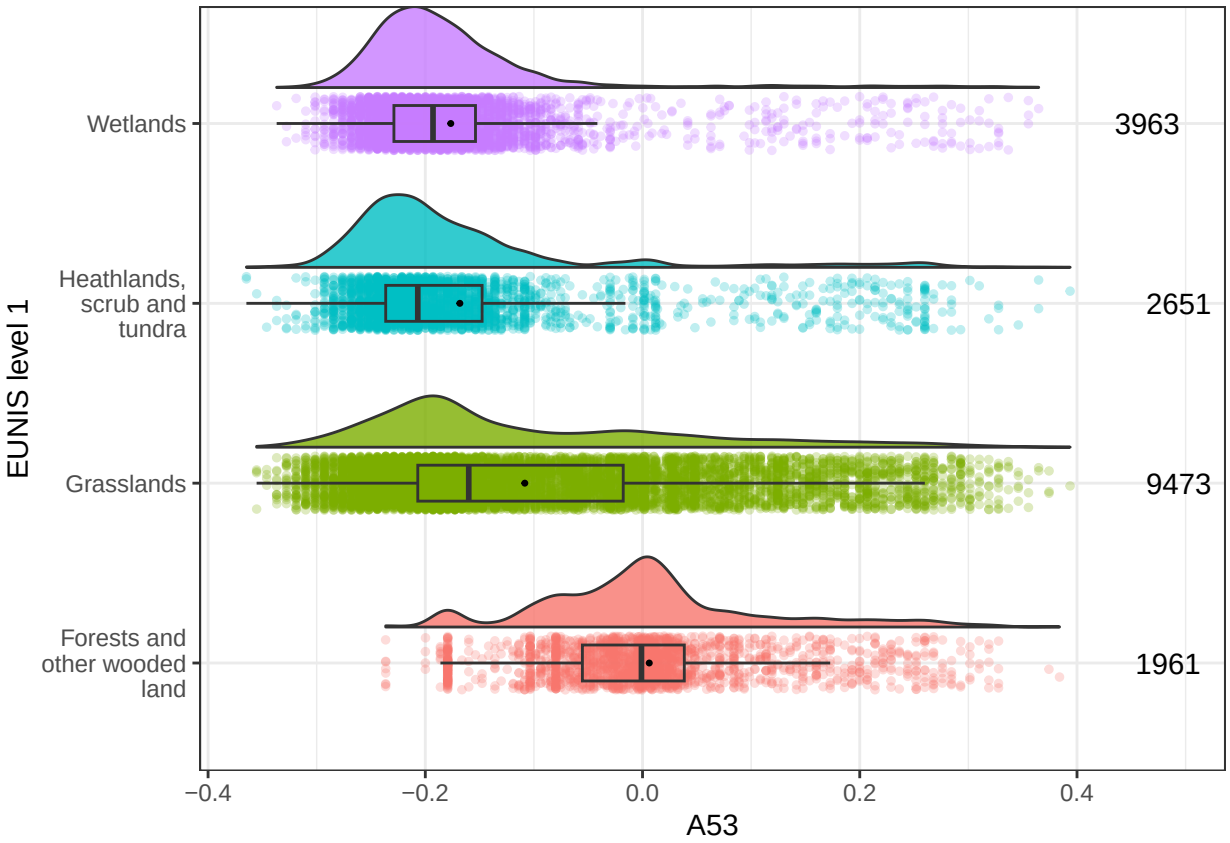

EUNIS level 1
Wetlands
Heathlands, scrub and tundra
Grasslands
Forests and other wooded land
3963
2651
9473
1961
−0.4
−0.2
0.0
0.2
A01

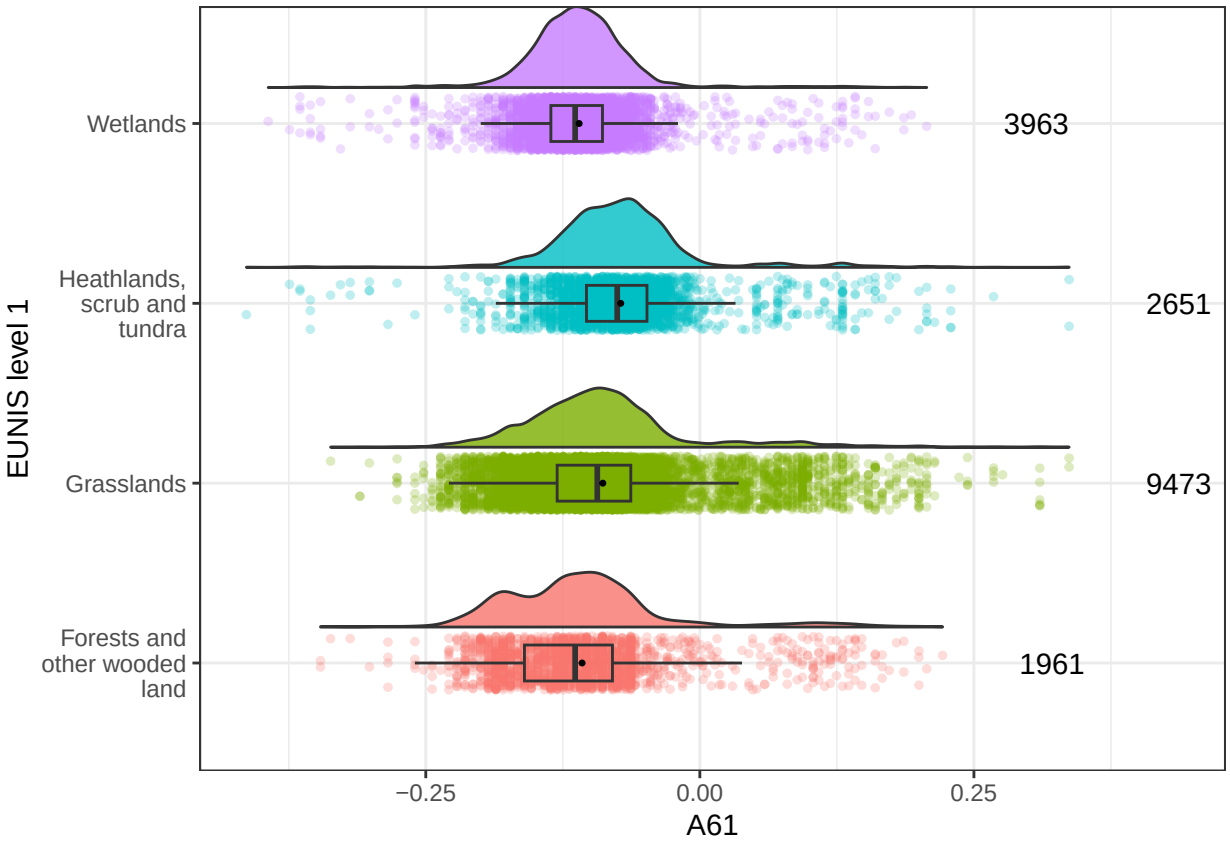

EUNIS level 1
Wetlands
Heathlands, scrub and tundra
Grasslands
Forests and other wooded land
3963
2651
9473
1961
−0.25
0.00
0.25
0.50
A10

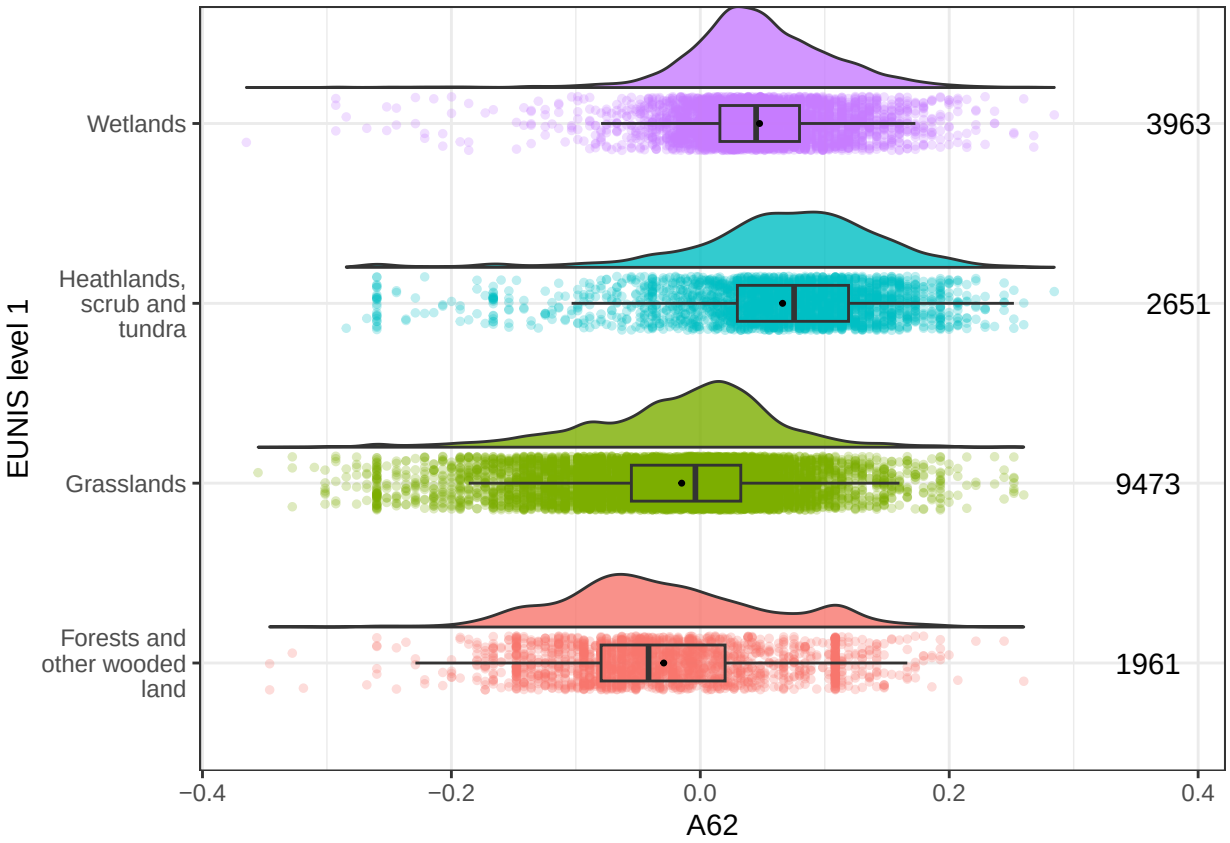

EUNIS level 1
Wetlands
Heathlands, scrub and tundra
Grasslands
Forests and other wooded land
3963
2651
9473
1961
−0.2
0.0
0.2
A22

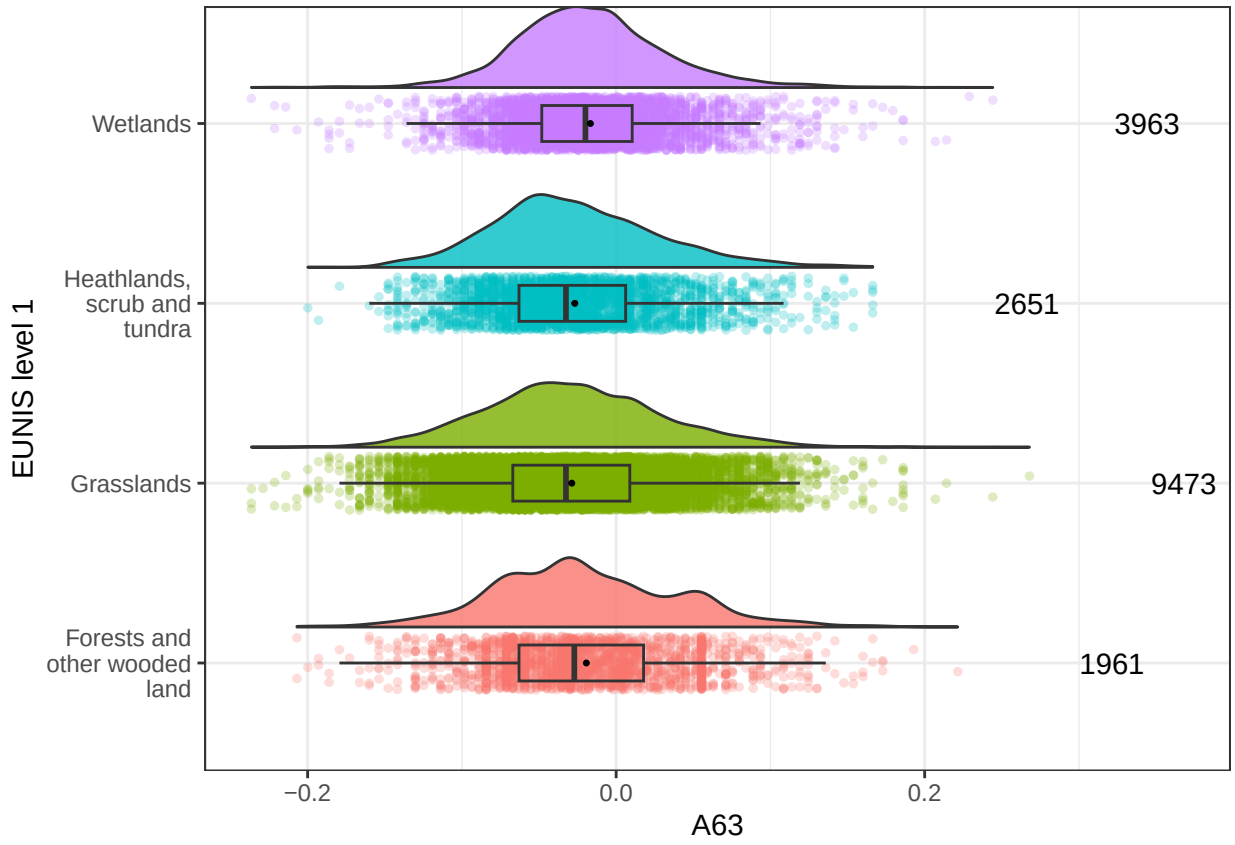

EUNIS level 1
Wetlands
Heathlands, scrub and tundra
Grasslands
Forests and other wooded land
3963
2651
9473
1961
−0.2
0.0
0.2
0.4
A33


EUNIS level 1
Wetlands
Heathlands, scrub and tundra
Grasslands
Forests and other wooded land
3963
2651
9473
1961
−0.4
−0.2
0.0
0.2
0.4
A40


EUNIS level 1
Wetlands
Heathlands, scrub and tundra
Grasslands
Forests and other wooded land
3963
2651
9473
1961
−0.4
−0.2
0.0
0.2
0.4
A53


EUNIS level 1
Wetlands
Heathlands, scrub and tundra
Grasslands
Forests and other wooded land
3963
2651
9473
1961
−0.25
0.00
0.25
A61


EUNIS level 1
Wetlands
Heathlands, scrub and tundra
Grasslands
Forests and other wooded land
3963
2651
9473
1961
−0.4
−0.2
0.0
0.2
0.4
A62



Distributions per bioregion

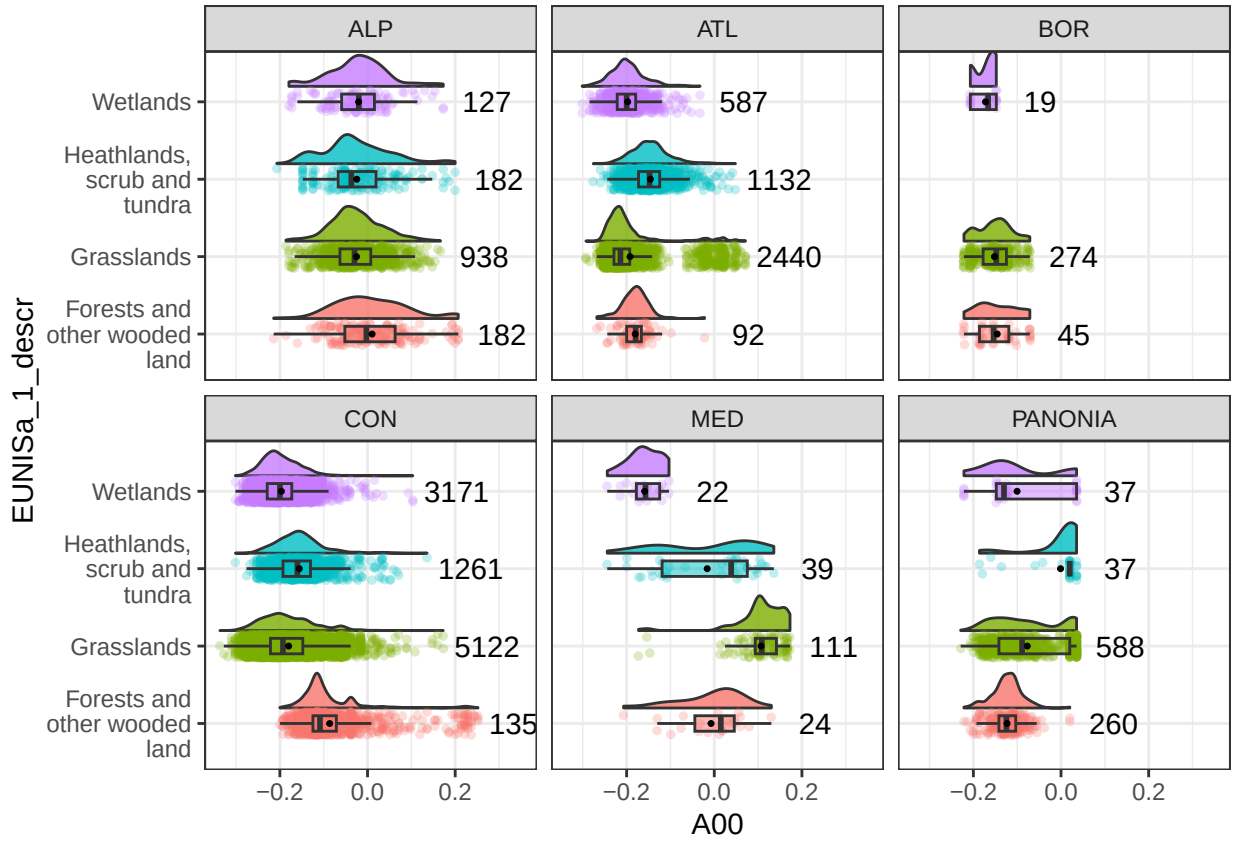
```
# Define a function to create plots with violin + boxplot + points
distr_plot_biogeo <- function(data, y_vars, y_labels) {
  plots <- list()

  for (i in seq_along(y_vars)) {
    y_var <- y_vars[[i]]
    y_label <- y_labels[[i]]

    p <- ggplot(data = data %>%
                  dplyr::filter(EUNISa_1 %in% c("T", "R", "S", "Q")),
                aes(x = EUNISa_1_descr, y = !!sym(y_var), fill = EUNISa_1_descr)) +
      geom_flat_violin(position = position_nudge(x = 0.2, y = 0), alpha = 0.8) +
      geom_point(aes(y = !!sym(y_var), color = EUNISa_1_descr),
                 position = position_jitter(width = 0.15), size = 1, alpha = 0.25) +
      geom_boxplot(width = 0.2, outlier.shape = NA, alpha = 0.5) +
      stat_summary(fun.y = mean, geom = "point", shape = 20, size = 1) +
      stat_summary(fun.data = function(x) data.frame(y = max(x) + 0.1,
                                                     label = length(x)),
                   geom = "text", aes(label = ..label..), vjust = 0.5) +
      labs(y = y_label, x = "EUNISa_1_descr") +
      scale_x_discrete(labels = function(x) str_wrap(x, width = 15)) +
```

```
      guides(fill = FALSE, color = FALSE) +
      theme_bw() + coord_flip() + facet_wrap(~ biogeo)

    plots[[y_var]] <- p
  }

  return(plots)
}
```

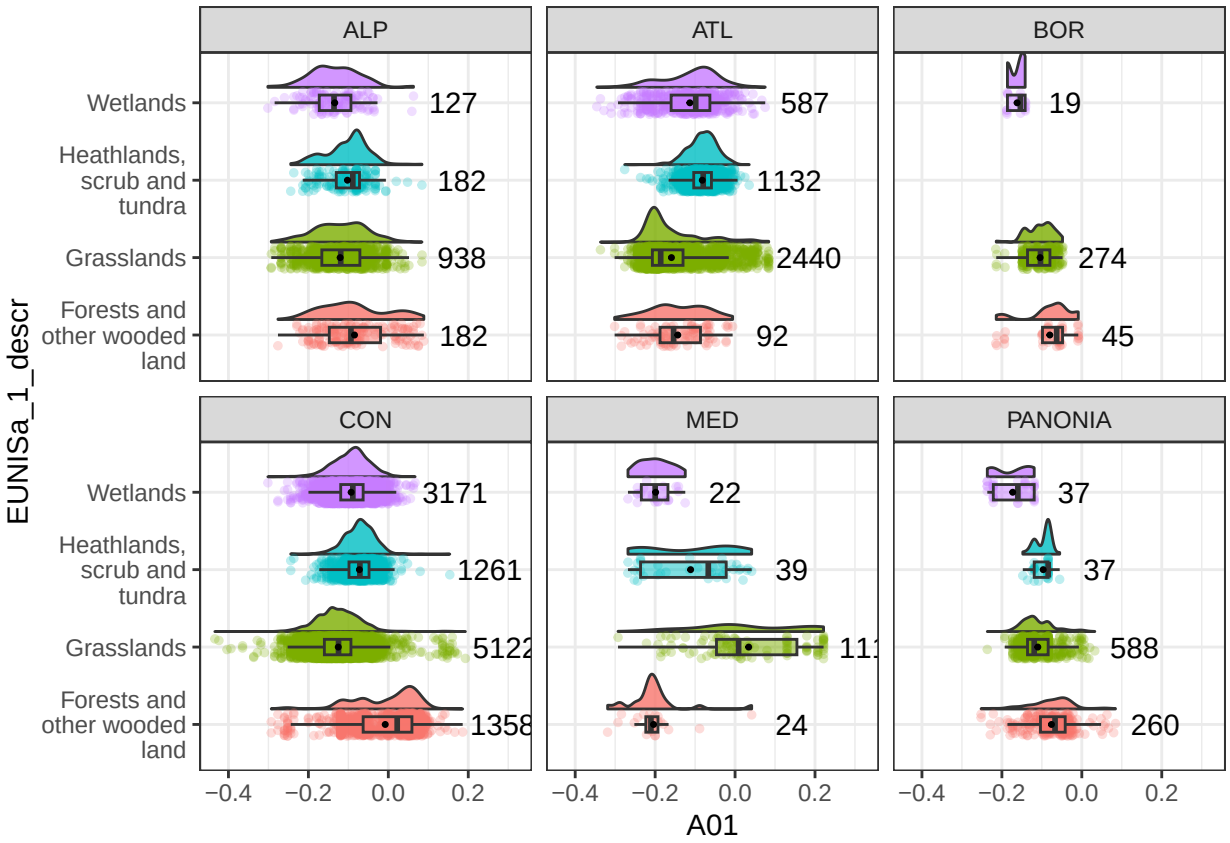
Distribution plots:

```
distr_plot_biogeo(data_validation,
            c("A00", "A01", "A10", "A22", "A33",
              "A40", "A53", "A61", "A62", "A63"),
            c("A00", "A01", "A10", "A22", "A33",
              "A40", "A53", "A61", "A62", "A63"))
```

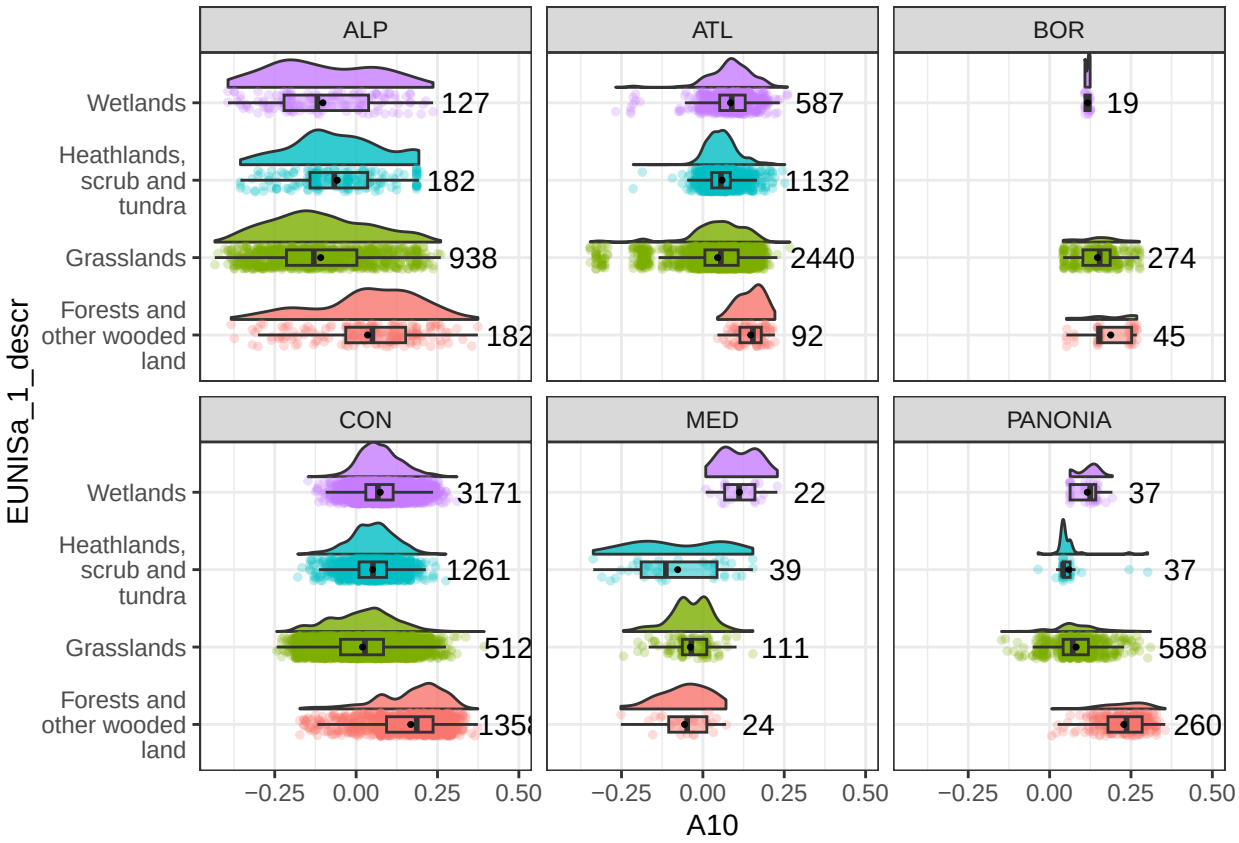
```
## $A00
```



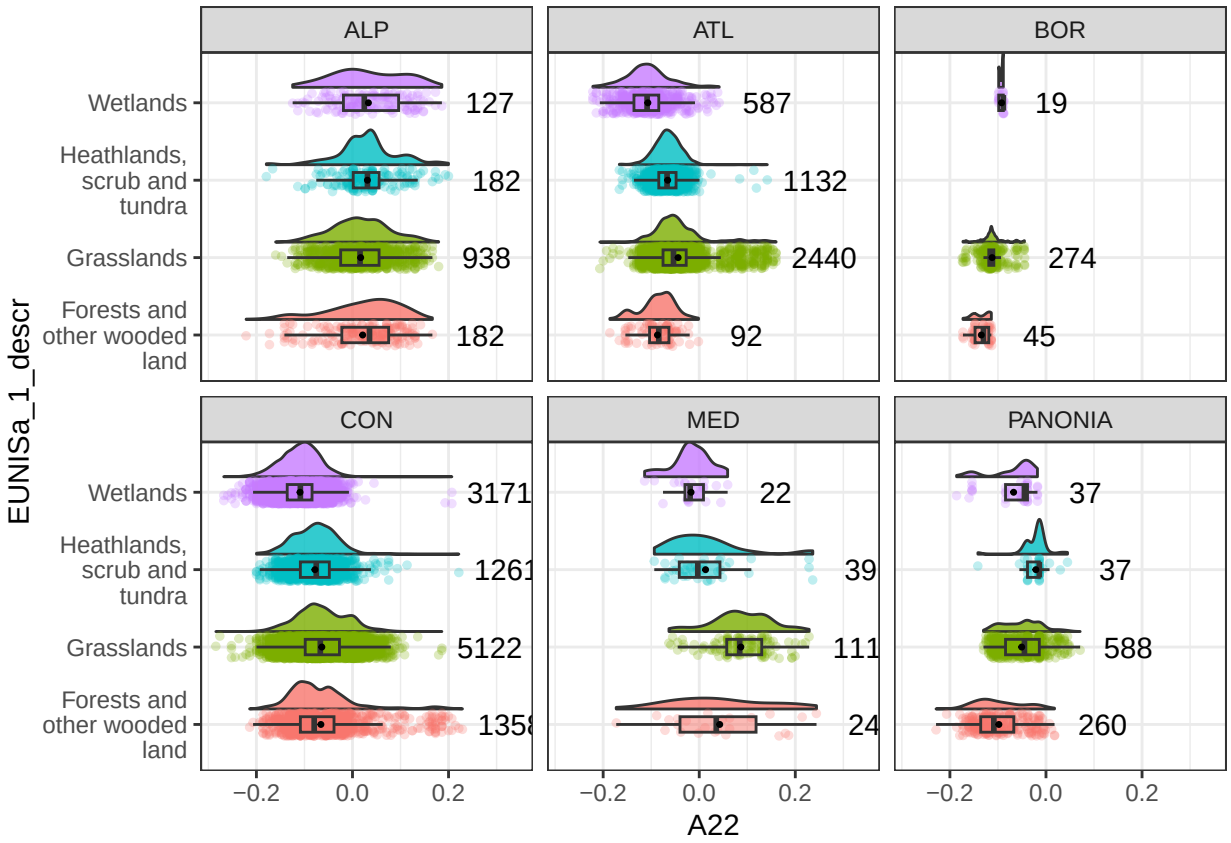

```
##
## $A01
```


ALP
ATL
BOR
CON
MED
PANONIA
Wetlands
Heathlands, scrub and tundra
Grasslands
Forests and other wooded land
EUNISa_1_descr
127
587
19
182
1132
938
2440
274
182
92
45
3171
22
37
1261
39
37
5122
111
588
1358
24
260
−0.4
−0.2
0.0
0.2
A01

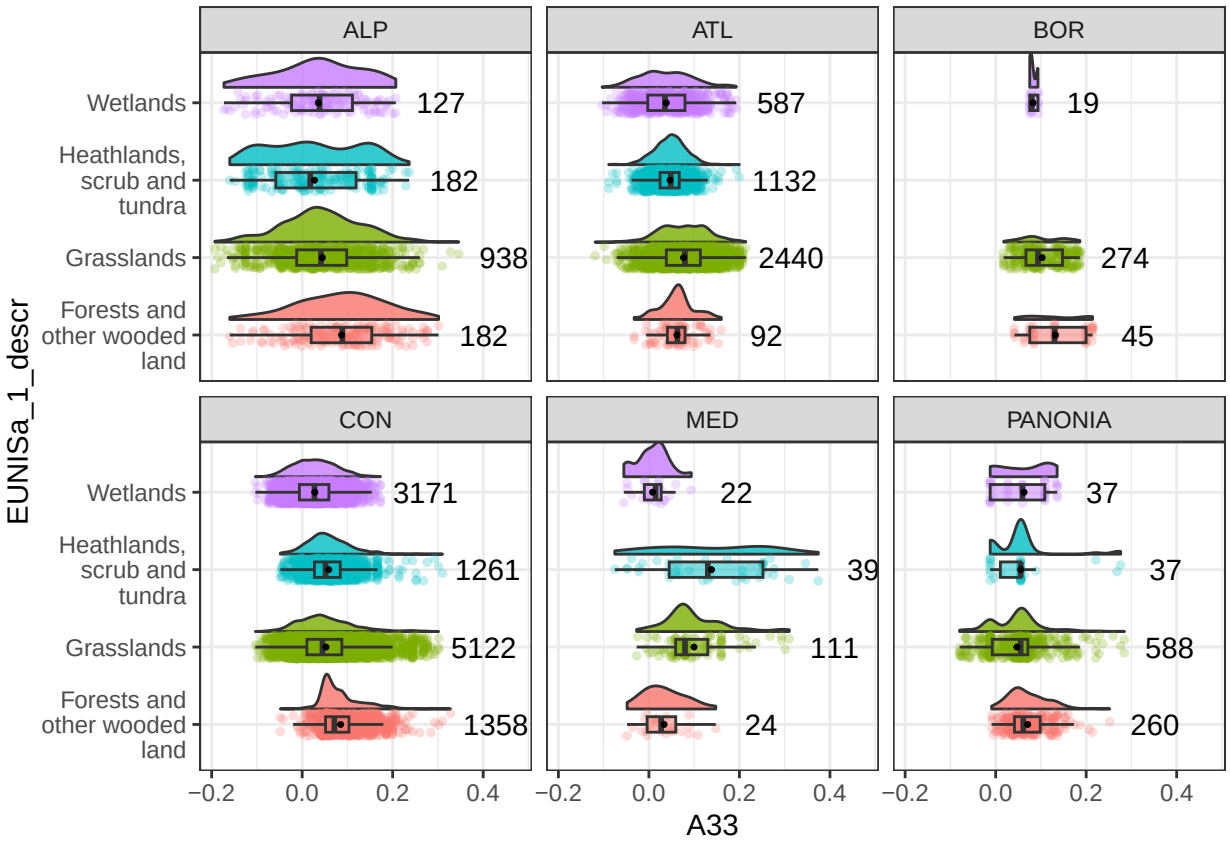
```
##
## $A10
```


ALP
ATL
BOR
CON
MED
PANONIA
Wetlands
Heathlands, scrub and tundra
Grasslands
Forests and other wooded land
127
182
938
182
587
1132
2440
92
19
274
45
3171
1261
512
1358
22
39
111
24
37
37
588
260
EUNISa_1_descr
−0.25
0.00
0.25
0.50
A10

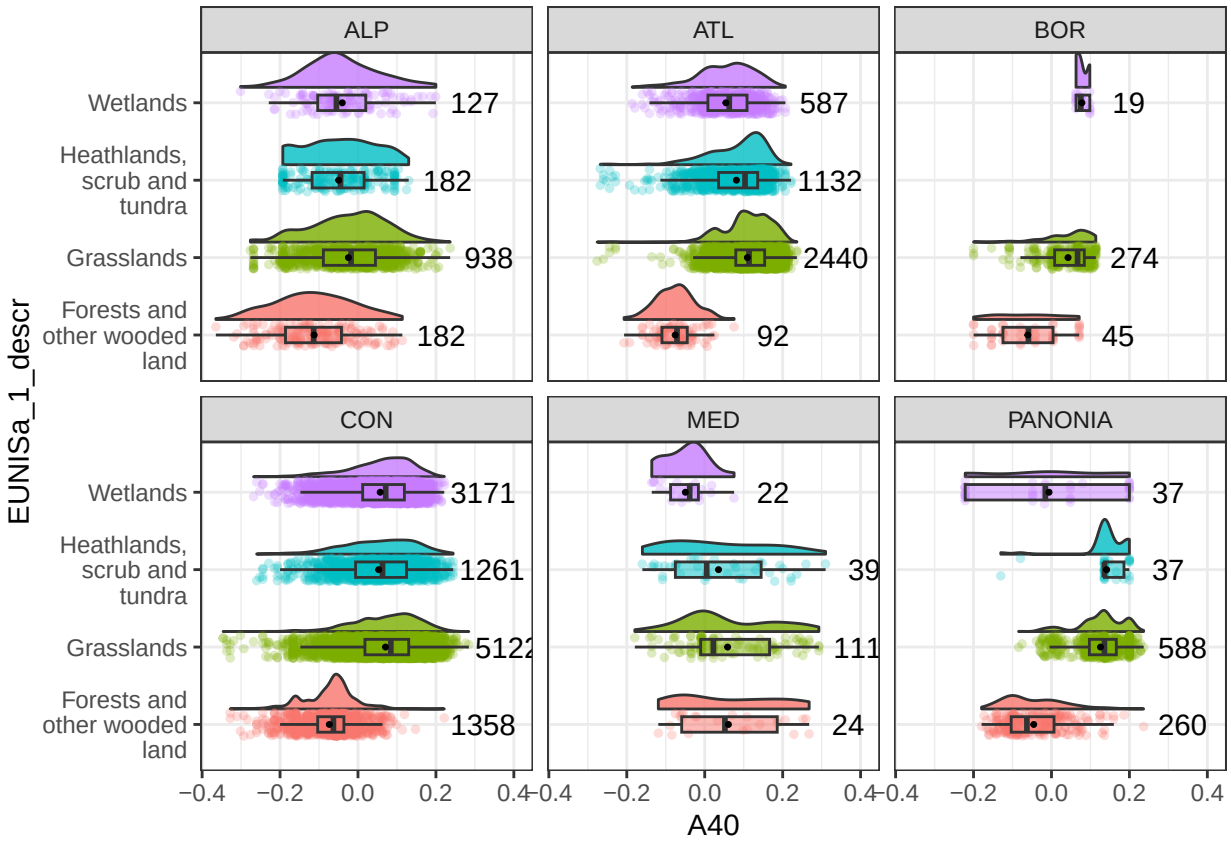
```
##
## $A22
```


ALP
ATL
BOR
CON
MED
PANONIA
Wetlands
Heathlands, scrub and tundra
Grasslands
Forests and other wooded land
EUNISa_1_descr
127
587
19
182
1132
938
2440
274
182
92
45
3171
22
37
1261
39
37
5122
111
588
1358
24
260
−0.2
0.0
0.2
A22

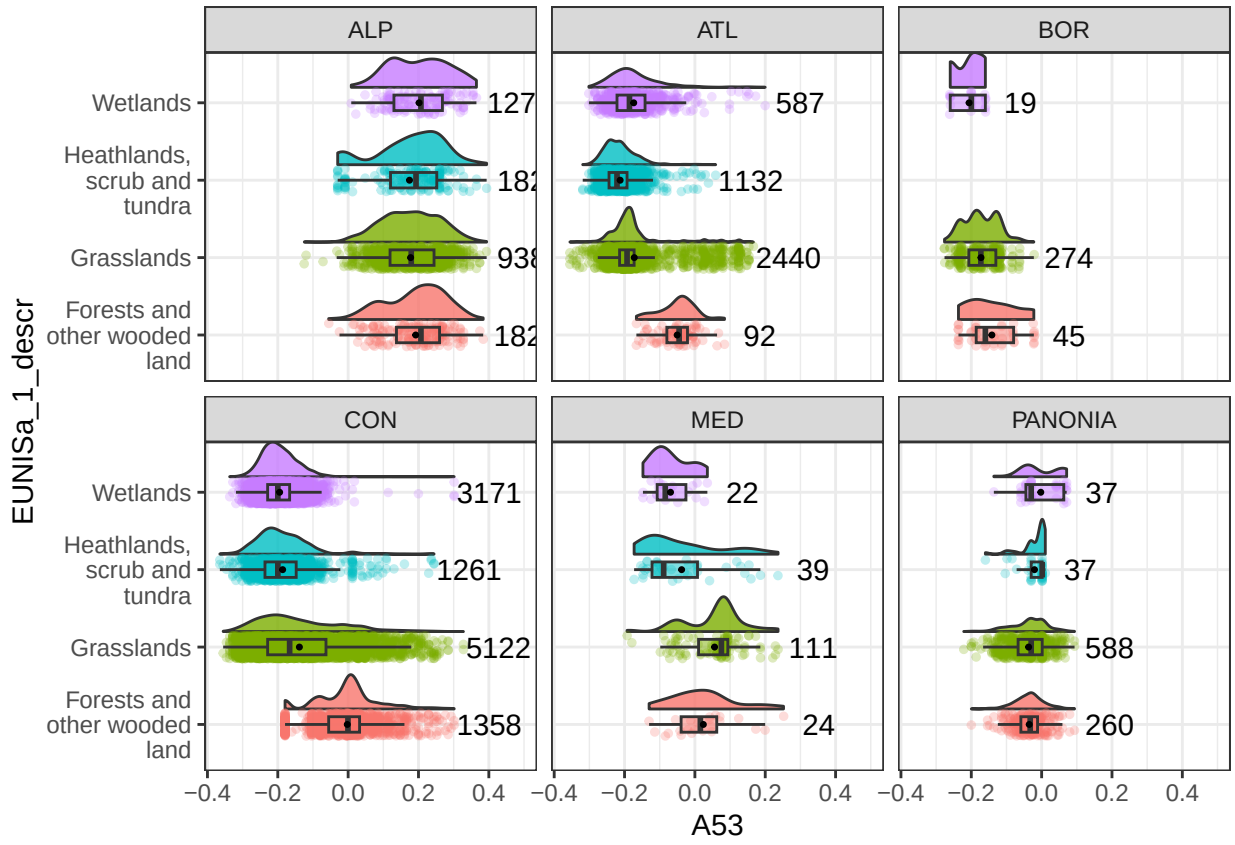
```
##
## $A33
```


ALP
ATL
BOR
CON
MED
PANONIA
EUNISa_1_descr
Wetlands
Heathlands, scrub and tundra
Grasslands
Forests and other wooded land
127
587
19
182
1132
938
2440
274
182
92
45
3171
22
37
1261
39
37
5122
111
588
1358
24
260
−0.2
0.0
0.2
0.4
A33

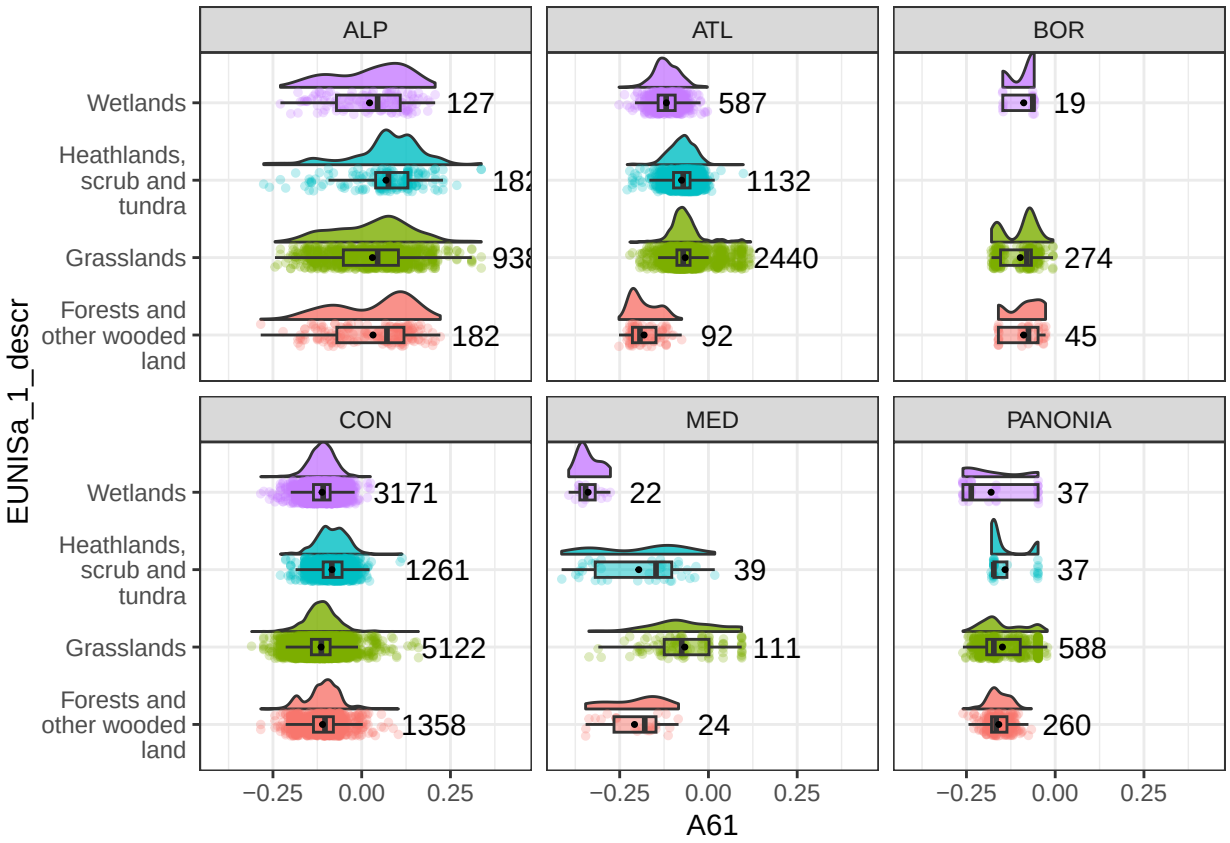
```
##
## $A40
```


ALP
ATL
BOR
CON
MED
PANONIA
Wetlands
Heathlands, scrub and tundra
Grasslands
Forests and other wooded land
EUNISa_1_descr
127
182
938
182
587
1132
2440
92
19
274
45
3171
1261
5122
1358
22
39
111
24
37
37
588
260
−0.4
−0.2
0.0
0.2
0.4
A40

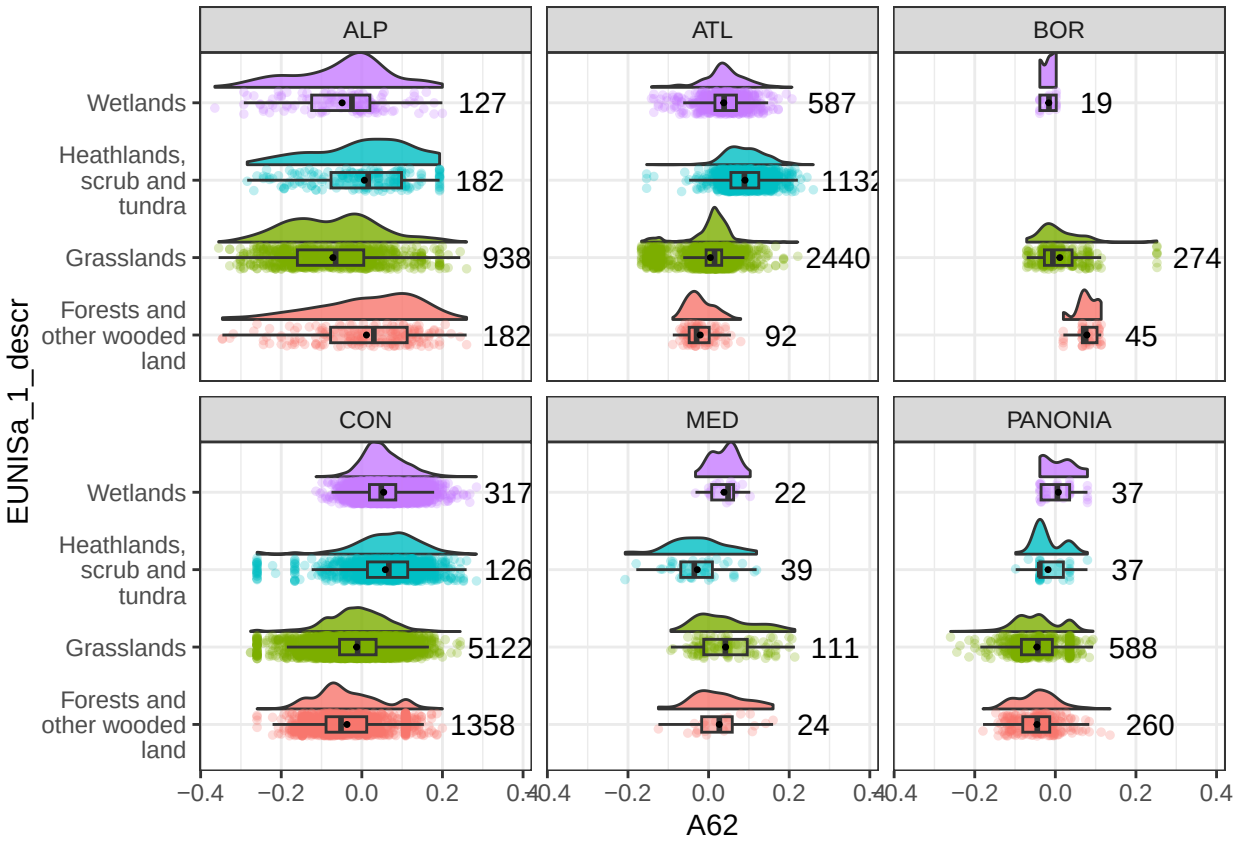
```
##
## $A53
```


ALP
ATL
BOR
CON
MED
PANONIA
EUNISa_1_descr
Wetlands
Heathlands, scrub and tundra
Grasslands
Forests and other wooded land
127
587
19
182
1132
938
2440
274
182
92
45
3171
22
37
1261
39
37
5122
111
588
1358
24
260
−0.4
−0.2
0.0
0.2
0.4
A53

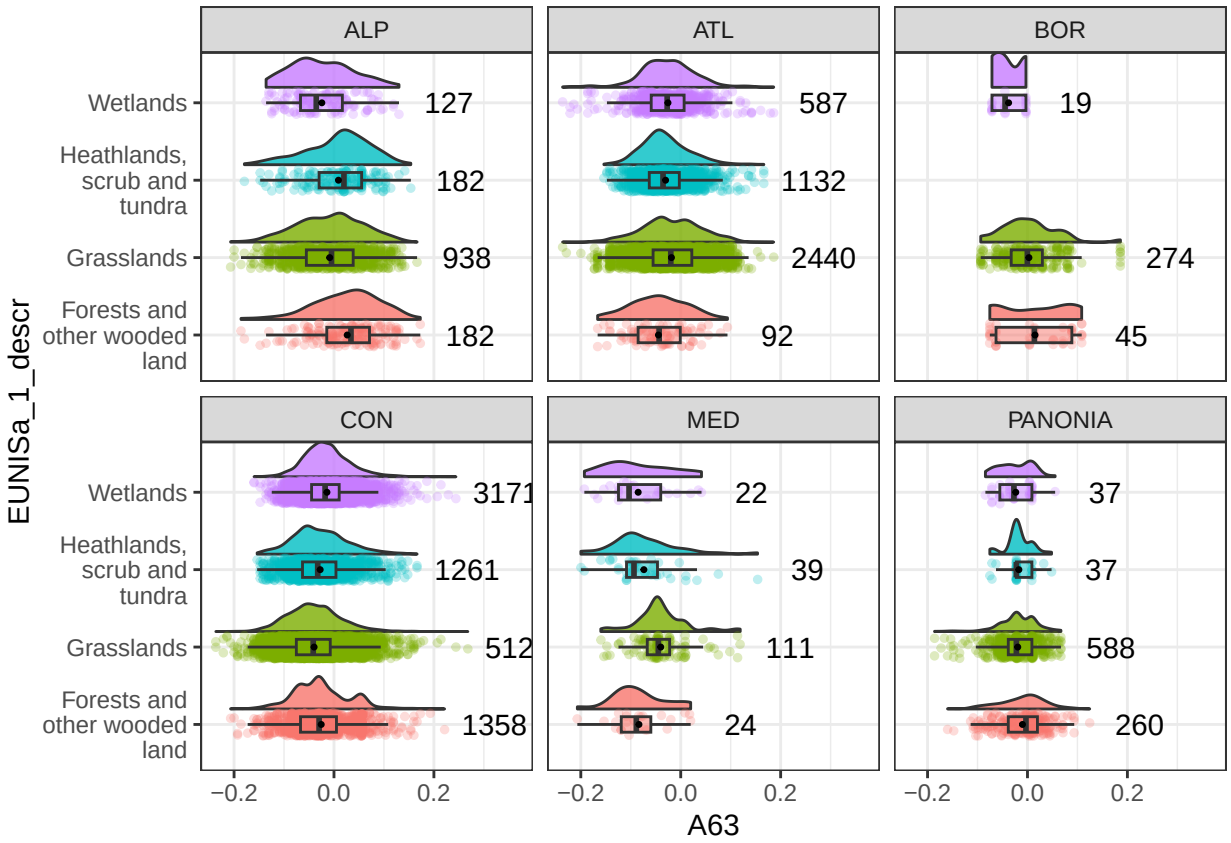
```
##
## $A61
```


ALP
ATL
BOR
CON
MED
PANONIA
Wetlands
Heathlands, scrub and tundra
Grasslands
Forests and other wooded land
EUNISa_1_descr
127
587
19
1132
2440
274
92
45
3171
22
37
1261
39
37
5122
111
588
1358
24
260
−0.25
0.00
0.25
A61

```
##
## $A62
```


ALP
ATL
BOR
CON
MED
PANONIA
EUNISa_1_descr
Wetlands
Heathlands, scrub and tundra
Grasslands
Forests and other wooded land
127
182
938
182
587
1132
2440
92
19
274
45
317
126
5122
1358
22
39
111
24
37
37
588
260
−0.4
−0.2
0.0
0.2
0.4
A62

```
##
## $A63
```



Define functions for RF models

Function for fitting RF models

RF models fitted using the conditional inference version of random forest (first cforest in package party, now fastcforest in package moreparty). Suggested if the data are highly correlated. Cforest is more stable in deriving variable importance values in the presence of highly correlated variables, thus providing better accuracy in calculating variable importance (ref below).

Hothorn, T., Hornik, K. and Zeileis, A. (2006) Unbiased Recursive Portioning: A Conditional Inference Framework. Journal of Computational and Graphical Statistics, 15, 651- 674. http://dx.doi.org/10.1198/106186006X133933

Choose the hyperparameter mtry based on the square root of the number of predictor variables:

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: Data mining, inference, and prediction. Springer Science & Business Media.

Maybe TO_DO: We variated ntree from 50 to 800 in steps of 50, leaving mtry constant at 2. Tis parameter variation showed that ntree=500 was optimal, while higher ntree led to no further model improvement (Supplementary Fig. S10). Subsequently, the hyperparameter mtry was varied from 2 to 8 with constant ntree=500. Here, mtry=3 led to the best results in almost all cases (Supplementary Fig. S11). Consequently, we chose ntree=500 and mtry=3 for our main analysis across all study sites.

Define a function to run fastcforest models:

```r
run_rf <- function(vars_RF, train_data, response_var, ntree = 500)
  {

  # Detect and register available cores (leave one free)
  n_cores <- parallel::detectCores() - 1
  cl <- makeCluster(n_cores)
  registerDoParallel(cl)

  train_name <- deparse(substitute(train_data))

  # Export necessary variables to the cluster
  clusterExport(cl, varlist = c("vars_RF", "train_data", "response_var"),
                envir = environment())


  # Set seed for reproducibility
  set.seed(123)

  # Measure execution time
  execution_time <- system.time({
    rf_model <- fastcforest(
      formula = reformulate(vars_RF, response = response_var),
      data = train_data,
      controls = party::cforest_control(
        mtry = round(sqrt(length(vars_RF))),
        ntree = ntree
      ),
      parallel = TRUE
    )
  })

  # Stop the cluster
  stopCluster(cl)

  # Return both the model and execution time
  list(model = rf_model, time = execution_time)
}
```

Function to compute variable importance

```r
# compute_varimp <- function(model, nperm = 100,
#                                  n_cores = parallel::detectCores() - 1) {
#   # Set up parallel backend
#   cl <- makeCluster(n_cores)
#   registerDoParallel(cl)
#
#   # Measure execution time
#   execution_time <- system.time({
#     varimp_list <- foreach(i = 1:nperm, .combine = '+',
#                            .packages = "party") %dopar% {
#       varimp(model, conditional = FALSE, nperm = 1)
```

```
#     }
#   })
#
#   stopCluster(cl)
#
#   # Average the results
#   varimp_avg <- varimp_list / nperm
#
#   return(list(varimp = varimp_avg, time = execution_time))
# }
```

Using permimp() en permimp package: https://cran.r-project.org/web/packages/permimp/vignettes/permimp-package.html#fn1

```
compute_varimp <- function(model, nperm = 100) {

  # Measure execution time
  execution_time <- system.time({
    varimp_result <- permimp(model, conditional = FALSE, progressBar = TRUE)
  })

  return(list(varimp = varimp_result, time = execution_time))
}
```

Function to compute CONDITIONAL variable importance

```
compute_varimp_cond <- function(model, nperm = 100) {

  # Measure execution time
  execution_time <- system.time({
    varimp_result <- permimp(model, conditional = TRUE, progressBar = TRUE)
  })

  return(list(varimp = varimp_result, time = execution_time))
}
```

Function to compute ROC (level 1)

```
compute_roc_level1 <- function(model, test_data) {
  # Measure execution time
  execution_time <- system.time({
    # Step 1: Predict probabilities
    probabilities <- predict(model, newdata = test_data, type = "prob")

    # Step 2: Convert list of matrices to a proper data frame
    prob_matrix <- t(sapply(probabilities, as.vector))
    prob_df <- as.data.frame(prob_matrix)
    colnames(prob_df) <- c("Q", "R", "S", "T")
```

```
    # Step 3: Prepare actual class labels
    actual <- factor(test_data$EUNISa_1, levels = c("Q", "R", "S", "T"))

    # Step 4: Binarize actual labels
    actual_bin <- model.matrix(~ actual - 1)
    colnames(actual_bin) <- gsub("actual", "", colnames(actual_bin))

    # Step 5: Compute ROC data for each class
    roc_data <- lapply(levels(actual), function(class) {
      roc_obj <- roc(actual_bin[, class], prob_df[[class]])
      auc_val <- round(auc(roc_obj), 3)
      data.frame(
        FPR = rev(roc_obj$specificities),
        TPR = rev(roc_obj$sensitivities),
        Class = paste0(class, " (AUC = ", auc_val, ")")
      )
    }) %>% bind_rows()
  })

  # Return both ROC data and execution time
  return(list(roc = roc_data, time = execution_time))
}
```

Function to compute ROC (level 2)

```
compute_roc_level2 <- function(model, test_data) {
  # Measure execution time
  execution_time <- system.time({
    # Step 1: Predict probabilities
    probabilities <- predict(model, newdata = test_data, type = "prob")

    # Step 2: Convert list of matrices to a proper data frame
    prob_matrix <- t(sapply(probabilities, as.vector))
    prob_df <- as.data.frame(prob_matrix)
    colnames(prob_df) <- c("Q1", "Q2", "Q4", "Q5", "R1", "R2", "R3", "R4", "R5",
                           "R6", "S3", "S4", "T1", "T3")

    # Step 3: Prepare actual class labels
    actual <- factor(test_data$EUNISa_2,
                     levels = c("Q1", "Q2", "Q4", "Q5", "R1", "R2", "R3", "R4",
                                "R5", "R6", "S3", "S4", "T1", "T3"))

    # Step 4: Binarize actual labels
    actual_bin <- model.matrix(~ actual - 1)
    colnames(actual_bin) <- gsub("actual", "", colnames(actual_bin))

    # Step 5: Compute ROC data for each class
    roc_data <- lapply(levels(actual), function(class) {
      roc_obj <- roc(actual_bin[, class], prob_df[[class]])
      auc_val <- round(auc(roc_obj), 3)
      data.frame(
        FPR = rev(roc_obj$specificities),
```

```
        TPR = rev(roc_obj$sensitivities),
        Class = paste0(class, " (AUC = ", auc_val, ")")
      )
    }) %>% bind_rows()
  })

  # Return both ROC data and execution time
  return(list(roc = roc_data, time = execution_time))
}
```

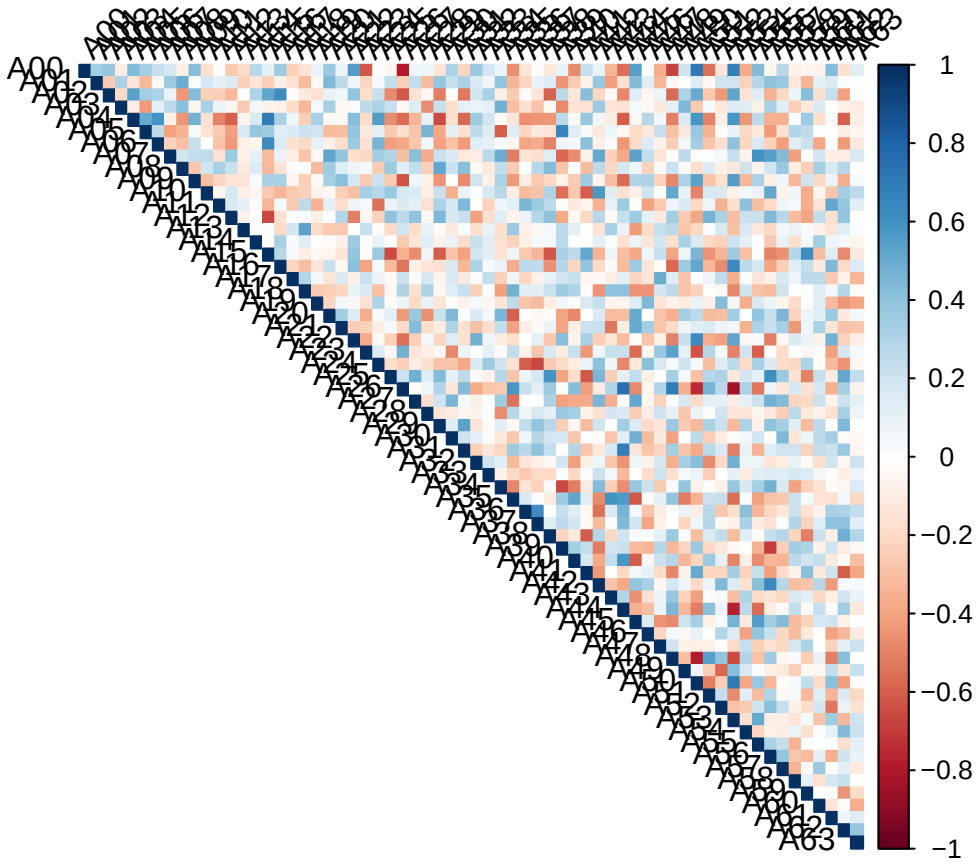
Define list of predictor vars

```
vars_RF_SatEmb <- colnames(
  data_validation)[startsWith(colnames(data_validation), "A")]
vars_RF_SatEmb_CH <- c(vars_RF_SatEmb, "canopy_height")
```

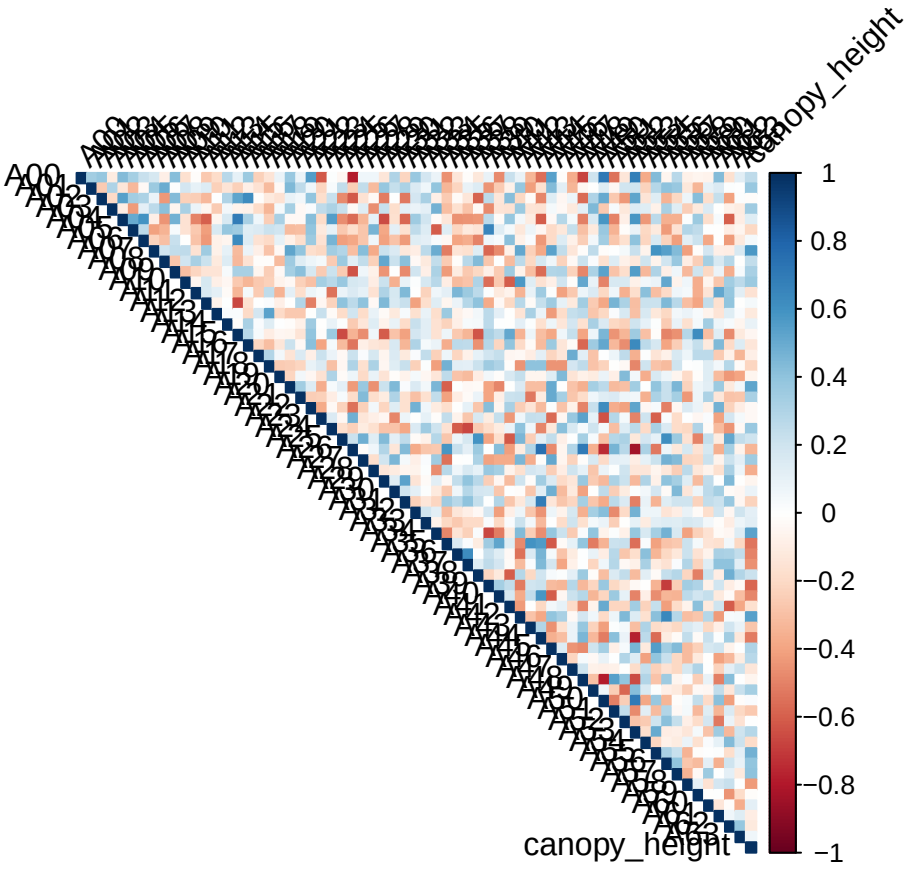
Correlation

Correlation of all variables to be included in RF models:

```
corrplot(data_validation %>%
           dplyr::select(starts_with("A")) %>%
           cor(use = "pairwise.complete.obs"),
         method = "color", type = "upper", tl.col = "black", tl.srt = 45)
```




```
corrplot(data_validation %>%
           dplyr::select(starts_with("A"), canopy_height) %>%
           cor(use = "pairwise.complete.obs"),
         method = "color", type = "upper", tl.col = "black", tl.srt = 45)
```



Rough validation

Define a set of rules for a first validation of ALL ReSurvey data ("Expert-based" rules). Not very ambitious, only taking out observations that are clearly wrong on the basis of indicator values.

Create column for first validation based on different indicators, where "wrong" is noted when the validation rule is not met.

Here only using validation based on canopy height, not NDWI.

Define rules:

```
data_validation <-
  data_validation %>%
  mutate(
    valid_1_CH = case_when(
      # R & Q points with high CH
      EUNISa_1 %in% c("R", "Q") & canopy_height > 2 ~ "wrong",
      # T points with low CH
      EUNISa_1 == "T" & canopy_height < 3 ~ "wrong",
      # S points with high CH
      EUNISa_1 == "S" & canopy_height > 3 ~ "wrong",
      TRUE ~ NA_character_),
    # Count how many validation rules are not met
    valid_1_count = rowSums(across(c(valid_1_CH), ~ . == "wrong"),
                            na.rm = TRUE),
```

```
    # Points where at least 1 rule not met
    valid_1 = if_else(valid_1_count > 0, "At least 1 rule broken",
                      "No rules broken so far")
    )
```

Fit RF models (level 1)

Without refinement

Get filtered data

```
# No validation
data_validation_novalid <- data_validation %>%
  # Select only GPS points
  dplyr::filter(Lctnmth == "Location with GPS" |
                  Lctnmth == "Location with differential GPS") %>%
  select(-Lctnmth) %>%
  mutate(EUNISa_1 = as.factor(EUNISa_1))

# Rough validation
data_validation_roughvalid <- data_validation %>%
  # Select only GPS points
  dplyr::filter(Lctnmth == "Location with GPS" |
                  Lctnmth == "Location with differential GPS") %>%
  # Filter out points that have not passed the rough validation
  dplyr::filter(valid_1 == "No rules broken so far") %>%
  select(-Lctnmth, -valid_1) %>%
  mutate(EUNISa_1 = as.factor(EUNISa_1))
```

N points per category

```
bind_rows(
  data_validation_novalid %>% count(EUNISa_1) %>%
    pivot_wider(names_from = EUNISa_1, values_from = n) %>%
    mutate(data = "data_validation_novalid") %>%
    select(data, Q, R, S, T),
  data_validation_roughvalid %>% count(EUNISa_1) %>%
    pivot_wider(names_from = EUNISa_1, values_from = n) %>%
    mutate(data = "data_validation_roughvalid") %>%
    select(data, Q, R, S, T),
  )
```

```
## # A tibble: 2 x 5
##   data                          Q     R     S     T
##   <chr>                     <int> <int> <int> <int>
## 1 data_validation_novalid    3814  6444  2364   347
## 2 data_validation_roughvalid 3522  6070  2097   263
```

Split into training and test data sets

```
set.seed(123)
```

```
train_indices_novalid <- sample(1:nrow(data_validation_novalid),
                          0.7 * nrow(data_validation_novalid))
train_indices_roughvalid <- sample(1:nrow(data_validation_roughvalid),
                          0.7 * nrow(data_validation_roughvalid))
```

```
train_data_novalid <- data_validation_novalid[train_indices_novalid, ]
train_data_roughvalid <- data_validation_roughvalid[train_indices_roughvalid, ]
```

```
test_data_novalid <- data_validation_novalid[-train_indices_novalid, ]
test_data_roughvalid <- data_validation_roughvalid[-train_indices_roughvalid, ]
```

Fit models

```
print(rf1$time)
```

```
##    user  system elapsed
##    9.89    4.89  110.48
```

```
print(rf2$time)
```

```
##    user  system elapsed
##    6.44    3.24  102.18
```

```
print(rf3$time)
```

```
##    user  system elapsed
##    4.73    2.67   86.47
```

```
print(rf4$time)
```

```
##    user  system elapsed
##    6.47    2.42  103.50
```

Predictions

Confusion matrices

```
# 1: No validation, SatEmb bands
confusionMatrix(predictions_rf1, test_data_novalid$EUNISa_1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    Q    R    S    T
##          Q  919  126   71    3
##          R  155 1753  114   48
##          S   48   70  508    0
##          T    0    1    6   69
##
## Overall Statistics
##
##                Accuracy : 0.835
##                  95% CI : (0.823, 0.8465)
##     No Information Rate : 0.5012
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7343
##
##  Mcnemar's Test P-Value : 1.558e-13
##
## Statistics by Class:
##
##                      Class: Q Class: R Class: S Class: T
## Sensitivity            0.8191   0.8990   0.7268  0.57500
## Specificity            0.9278   0.8367   0.9630  0.99814
## Pos Pred Value         0.8213   0.8469   0.8115  0.90789
## Neg Pred Value         0.9268   0.8918   0.9415  0.98663
## Prevalence             0.2884   0.5012   0.1796  0.03084
## Detection Rate         0.2362   0.4505   0.1306  0.01773
## Detection Prevalence   0.2876   0.5320   0.1609  0.01953
## Balanced Accuracy      0.8734   0.8678   0.8449  0.78657
```

```
# 2: No validation, SatEmb bands + CH
confusionMatrix(predictions_rf2, test_data_novalid$EUNISa_1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    Q    R    S    T
##          Q  918  124   74    5
##          R  160 1744  112   47
##          S   44   79  510    3
##          T    0    3    3   65
##
## Overall Statistics
##
##                Accuracy : 0.8319
##                  95% CI : (0.8198, 0.8435)
##     No Information Rate : 0.5012
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7295
##
##  Mcnemar's Test P-Value : 2.116e-11
```

```
##
## Statistics by Class:
##
##                      Class: Q Class: R Class: S Class: T
## Sensitivity            0.8182   0.8944   0.7296  0.54167
## Specificity            0.9267   0.8357   0.9605  0.99841
## Pos Pred Value         0.8189   0.8454   0.8019  0.91549
## Neg Pred Value         0.9264   0.8873   0.9419  0.98560
## Prevalence             0.2884   0.5012   0.1796  0.03084
## Detection Rate         0.2359   0.4482   0.1311  0.01671
## Detection Prevalence   0.2881   0.5302   0.1635  0.01825
## Balanced Accuracy      0.8724   0.8650   0.8451  0.77004
```

```
# 3: Rough validation, SatEmb bands
confusionMatrix(predictions_rf3, test_data_roughvalid$EUNISa_1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    Q    R    S    T
##          Q  865  133   76    3
##          R  116 1660  125   11
##          S   59   58  419    0
##          T    0    2    4   55
##
## Overall Statistics
##
##                Accuracy : 0.8363
##                  95% CI : (0.8238, 0.8483)
##     No Information Rate : 0.5167
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7321
##
##  Mcnemar's Test P-Value : 2.815e-07
##
## Statistics by Class:
##
##                      Class: Q Class: R Class: S Class: T
## Sensitivity            0.8317   0.8958   0.6715  0.79710
## Specificity            0.9167   0.8546   0.9605  0.99829
## Pos Pred Value         0.8032   0.8682   0.7817  0.90164
## Neg Pred Value         0.9303   0.8847   0.9328  0.99603
## Prevalence             0.2900   0.5167   0.1740  0.01924
## Detection Rate         0.2412   0.4629   0.1168  0.01534
## Detection Prevalence   0.3003   0.5332   0.1495  0.01701
## Balanced Accuracy      0.8742   0.8752   0.8160  0.89770
```

```
# 4: Rough validation, SatEmb bands + CH
confusionMatrix(predictions_rf4, test_data_roughvalid$EUNISa_1)
```

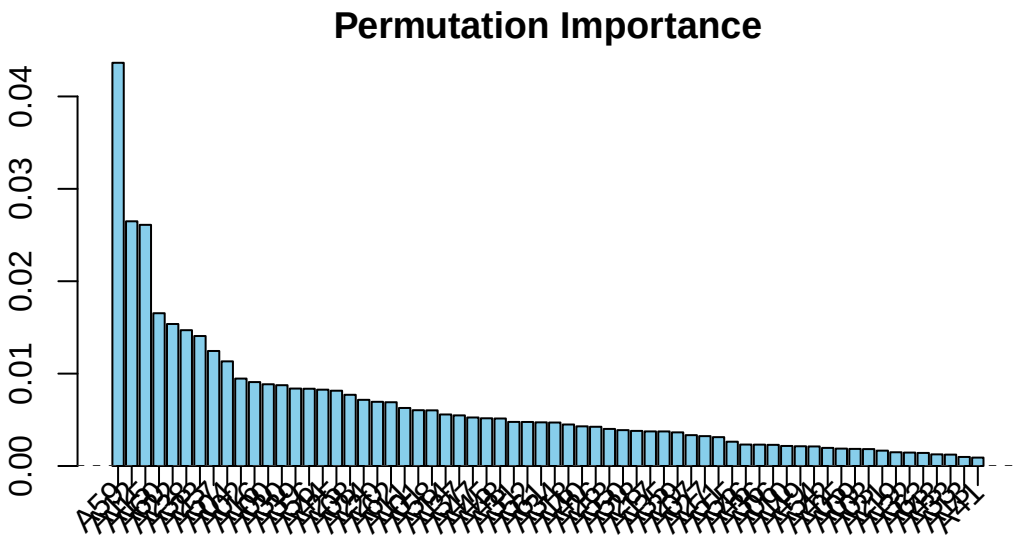
```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    Q    R    S    T
##          Q  854  131   69    1
##          R  131 1670  116    1
##          S   55   52  439    0
##          T    0    0    0   67
##
## Overall Statistics
##
##                Accuracy : 0.845
##                  95% CI : (0.8327, 0.8567)
##     No Information Rate : 0.5167
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7464
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Q Class: R Class: S Class: T
## Sensitivity            0.8212   0.9012   0.7035  0.97101
## Specificity            0.9211   0.8569   0.9639  1.00000
## Pos Pred Value         0.8095   0.8707   0.8040  1.00000
## Neg Pred Value         0.9265   0.8903   0.9391  0.99943
## Prevalence             0.2900   0.5167   0.1740  0.01924
## Detection Rate         0.2381   0.4657   0.1224  0.01868
## Detection Prevalence   0.2942   0.5349   0.1523  0.01868
## Balanced Accuracy      0.8711   0.8791   0.8337  0.98551
```

Variable importance

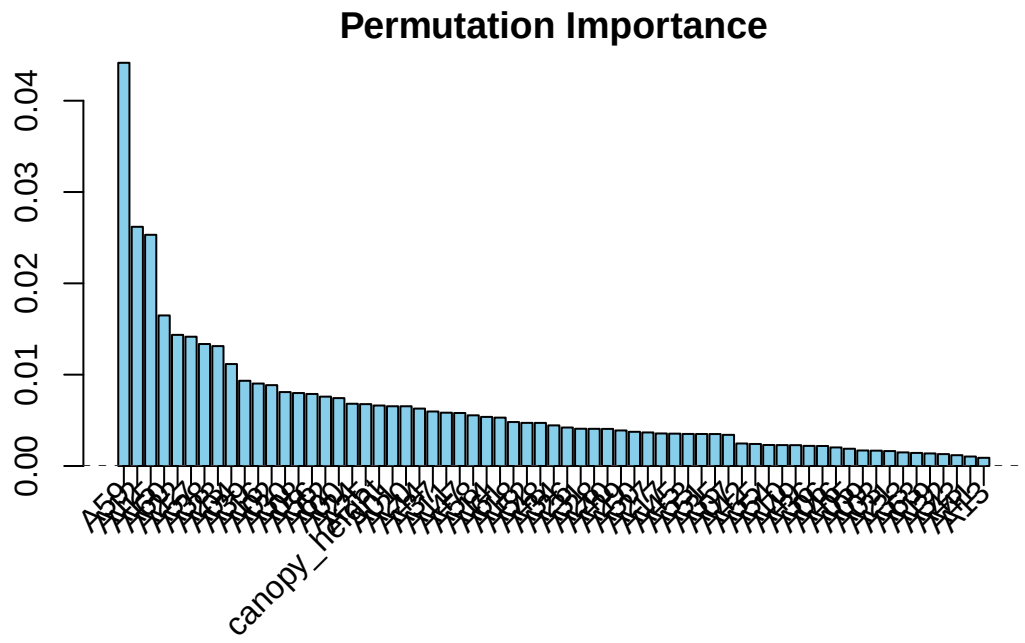
Unconditional

```
plot(varimp_rf1$varimp, margin = c(10, 6, 2, 2))
```

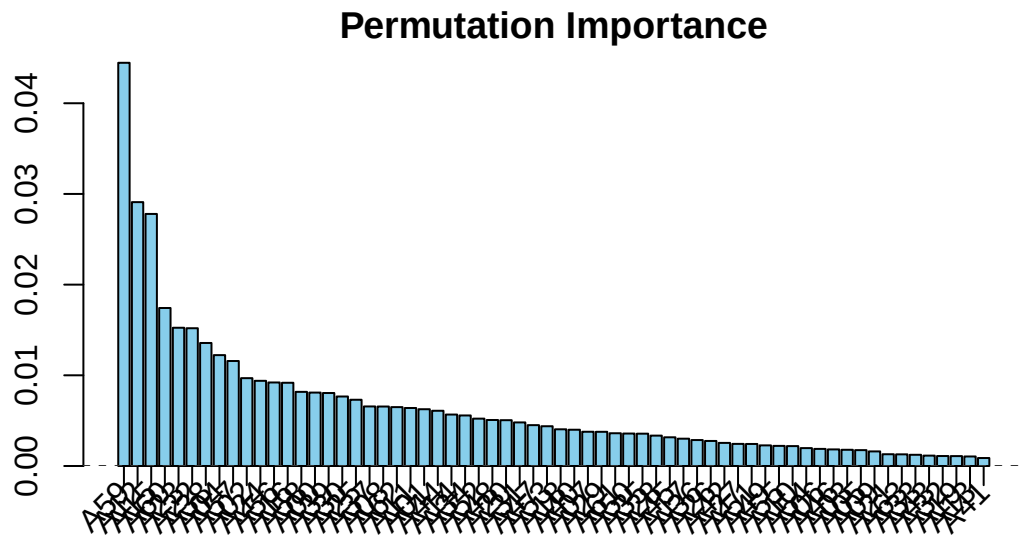


Plots

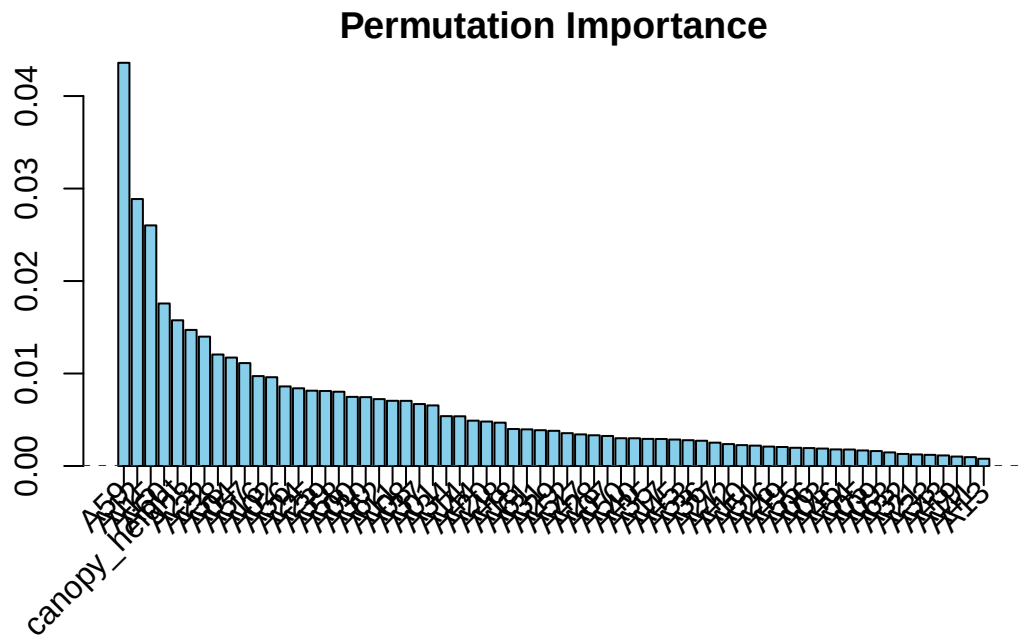
```
plot(varimp_rf2$varimp, margin = c(10, 6, 2, 2))
```




```
plot(varimp_rf3$varimp, margin = c(10, 6, 2, 2))
```




```
plot(varimp_rf4$varimp, margin = c(10, 6, 2, 2))
```



TBD: Conditional

With refinement

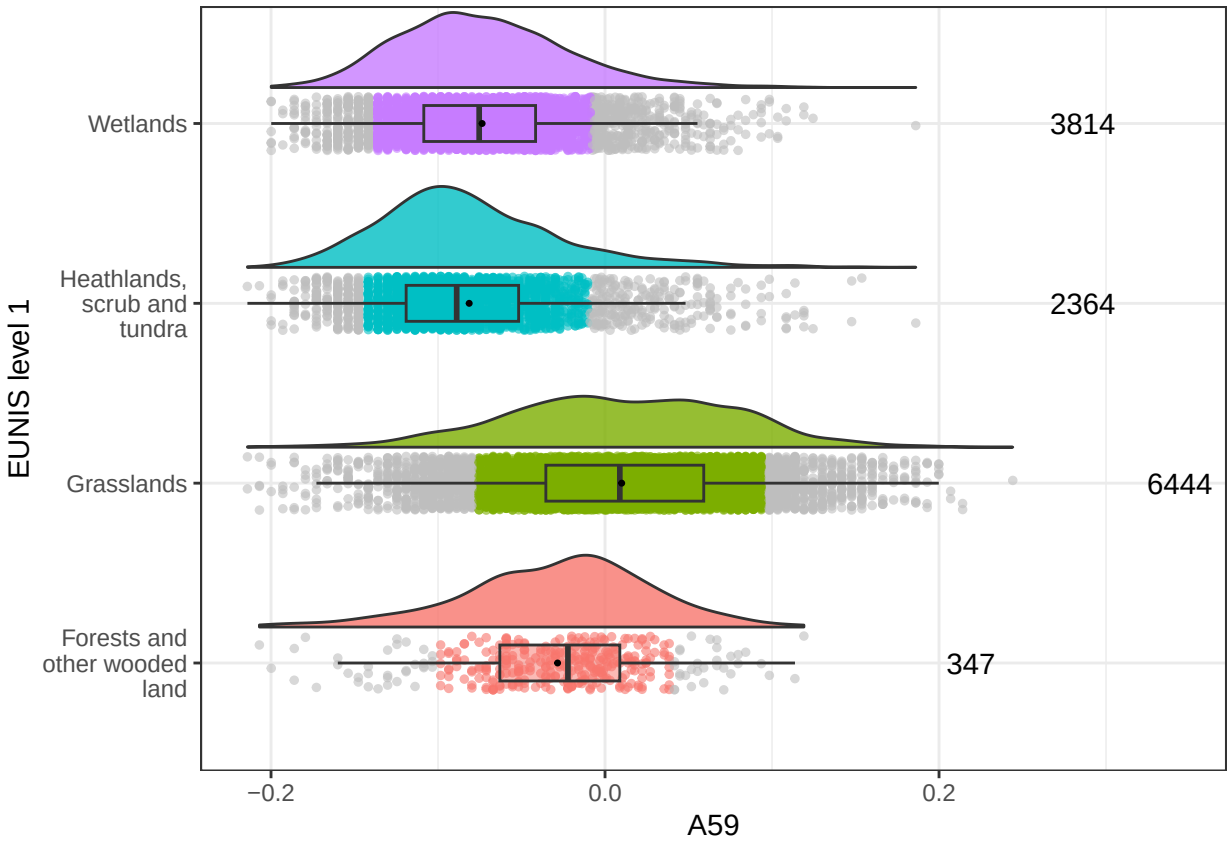
Refinement

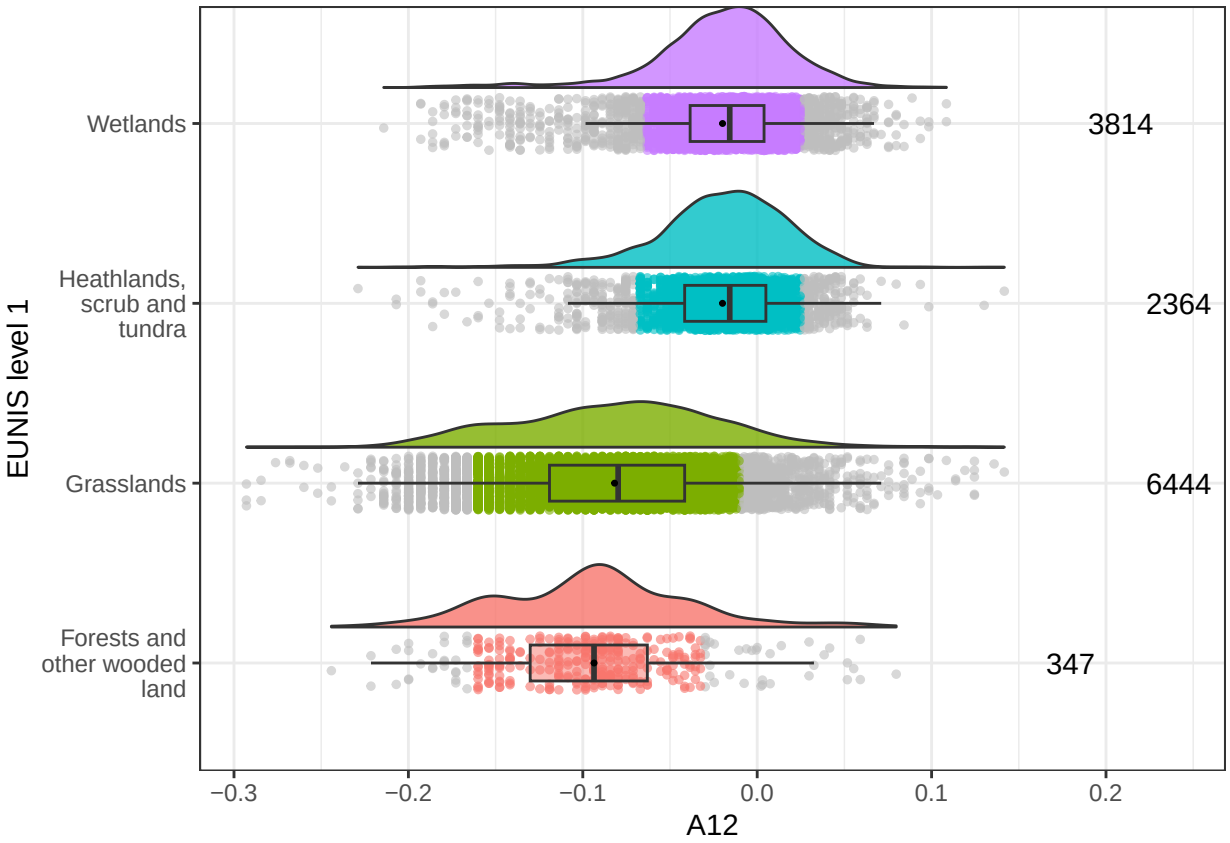
Based on the 1st, 2nd, 3rd, 4th... most important variables: A59, A12, A15, A60.

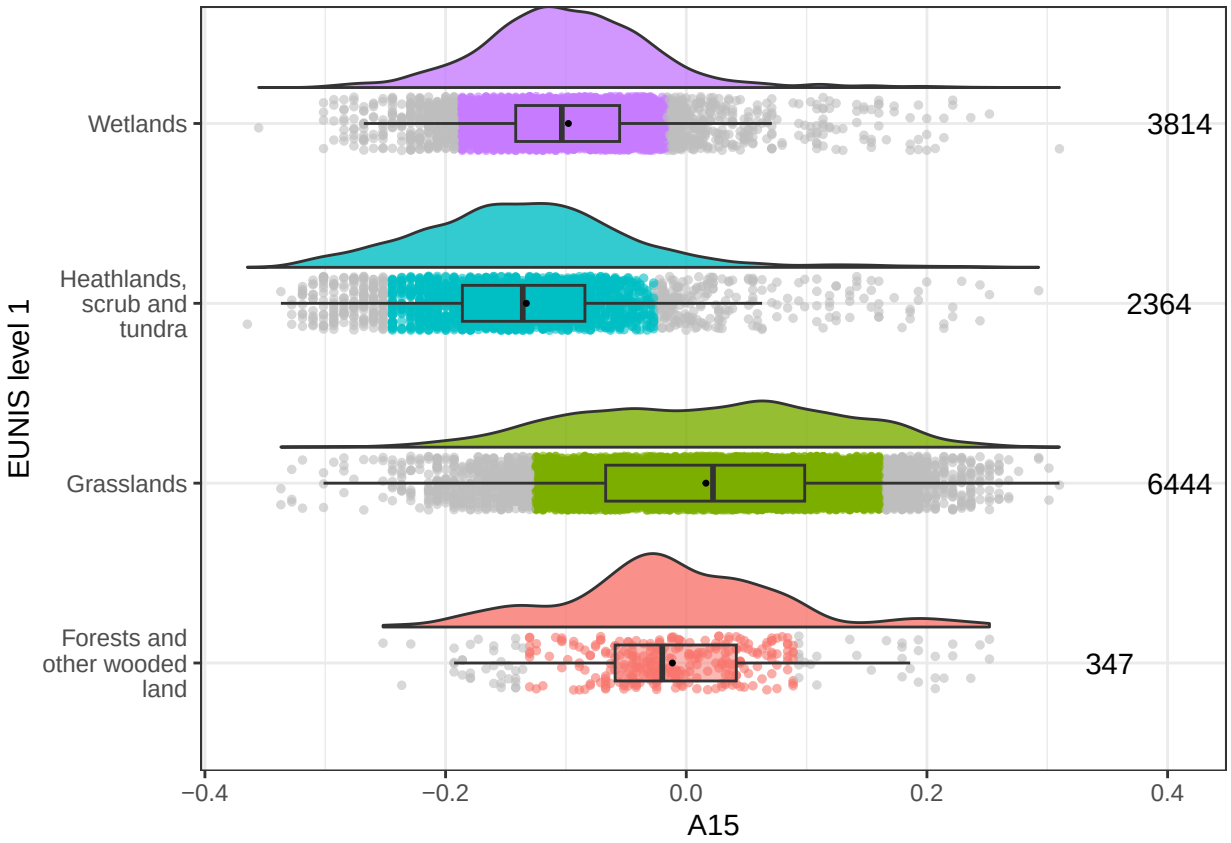
```
distr_plot_percentiles <- function(data, y_vars, y_labels) {
  for (i in seq_along(y_vars)) {
    y_var <- y_vars[[i]]
    y_label <- y_labels[[i]]

    # Calculate percentiles per EUNISa_1 group
    percentiles <- data %>%
      group_by(EUNISa_1) %>%
      summarise(
        p10 = quantile(.data[[y_var]], 0.1, na.rm = TRUE),
        p90 = quantile(.data[[y_var]], 0.9, na.rm = TRUE),
        .groups = "drop"
      )

    # Join percentiles back to data
    data_flagged <- data %>%
```

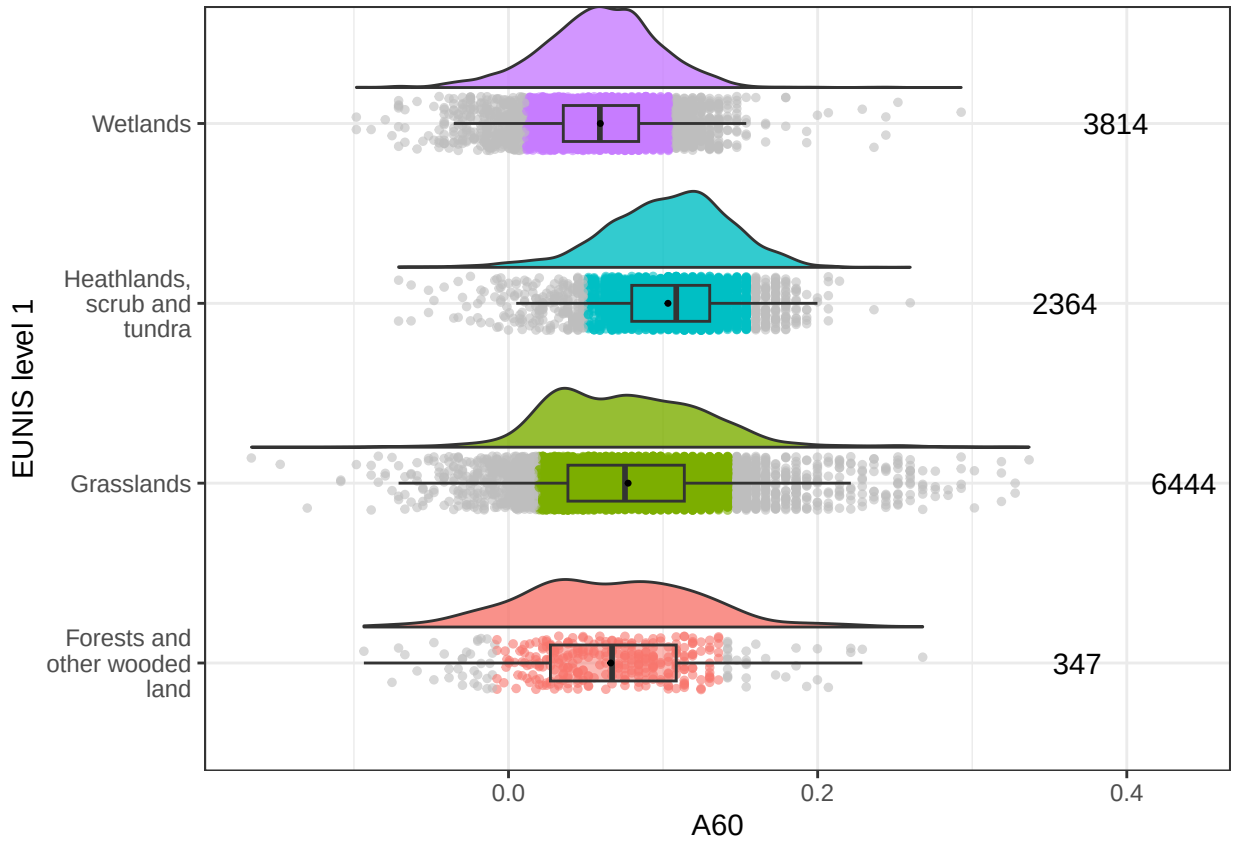
```
      left_join(percentiles, by = "EUNISa_1") %>%
      mutate(outlier_flag = case_when(
        .data[[y_var]] < p10 ~ "low",
        .data[[y_var]] > p90 ~ "high",
        TRUE ~ "mid"
      ))

    # Filter and plot
    p <- ggplot(data = data_flagged %>%
                  dplyr::filter(EUNISa_1 %in% c("T", "R", "S", "Q")),
                aes(x = EUNISa_1_descr, y = .data[[y_var]])) +
      geom_flat_violin(aes(fill = EUNISa_1_descr),
                       position = position_nudge(x = 0.2, y = 0), alpha = 0.8) +
      geom_point(aes(color = ifelse(outlier_flag == "mid",
                                    EUNISa_1_descr, "grey")),
                 position = position_jitter(width = 0.15), size = 1,
                 alpha = 0.6) +
      geom_boxplot(aes(fill = EUNISa_1_descr), width = 0.2, outlier.shape = NA,
                   alpha = 0.5) +
      stat_summary(fun = mean, geom = "point", shape = 20, size = 1) +
      stat_summary(fun.data = function(x) data.frame(y = max(x, na.rm = TRUE) +
                                                       0.1, label = length(x)),
                   geom = "text", aes(label = ..label..), vjust = 0.5) +
      labs(y = y_label, x = "EUNIS level 1") +
      scale_x_discrete(labels = function(x) str_wrap(x, width = 15)) +
      guides(fill = FALSE, color = FALSE) +
      theme_bw() + coord_flip() +
      scale_color_manual(values = c(
        "Forests and other wooded land" = "#F8766D",
        "Grasslands" = "#7CAE00",
        "Heathlands, scrub and tundra" = "#00BFC4",
        "Wetlands" = "#C77CFF",
        "grey" = "grey"))

    print(p)
  }
}
```

```
distr_plot_percentiles(
  # GPS points
  data_validation_novalid,
  c("A59", "A12", "A15", "A60"),
  c("A59", "A12", "A15", "A60"))
```

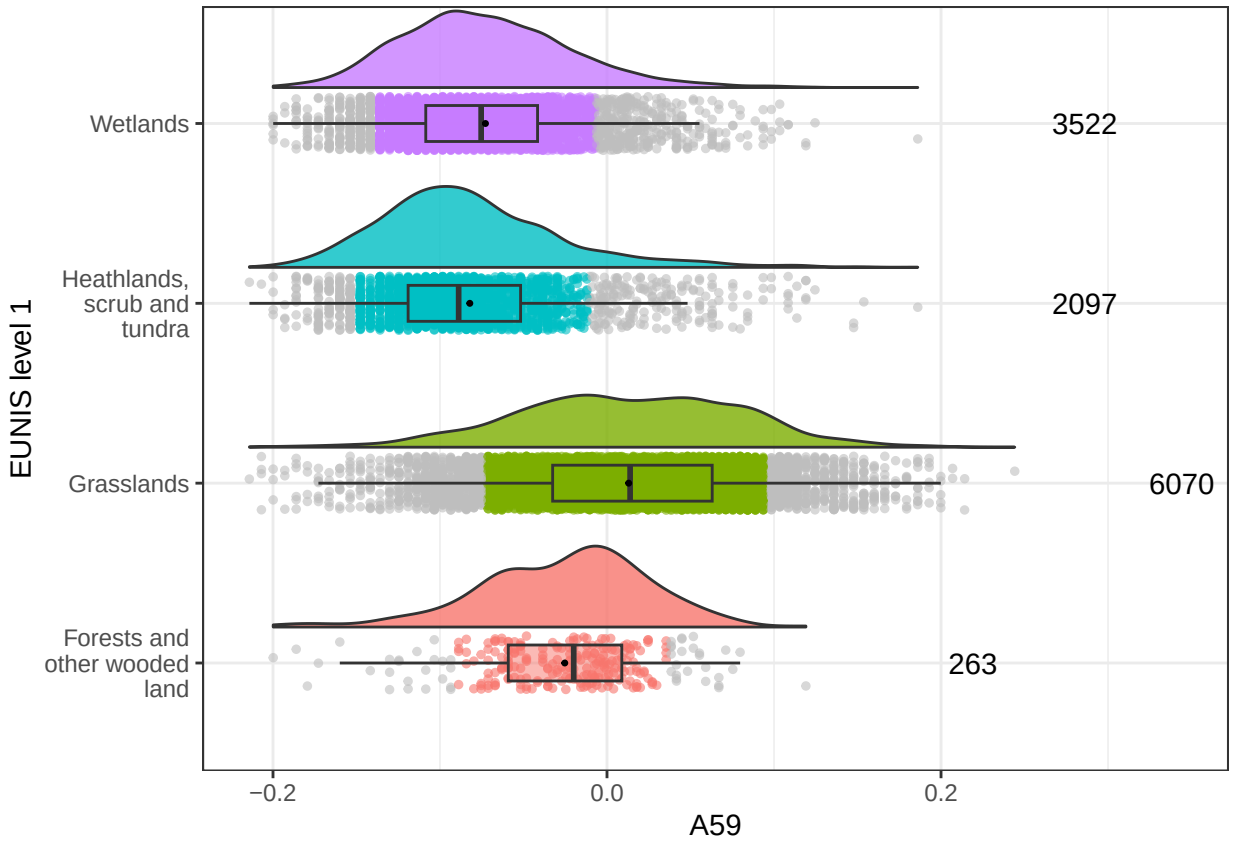

EUNIS level 1
Wetlands
Heathlands, scrub and tundra
Grasslands
Forests and other wooded land
3814
2364
6444
347
−0.2
0.0
0.2
A59

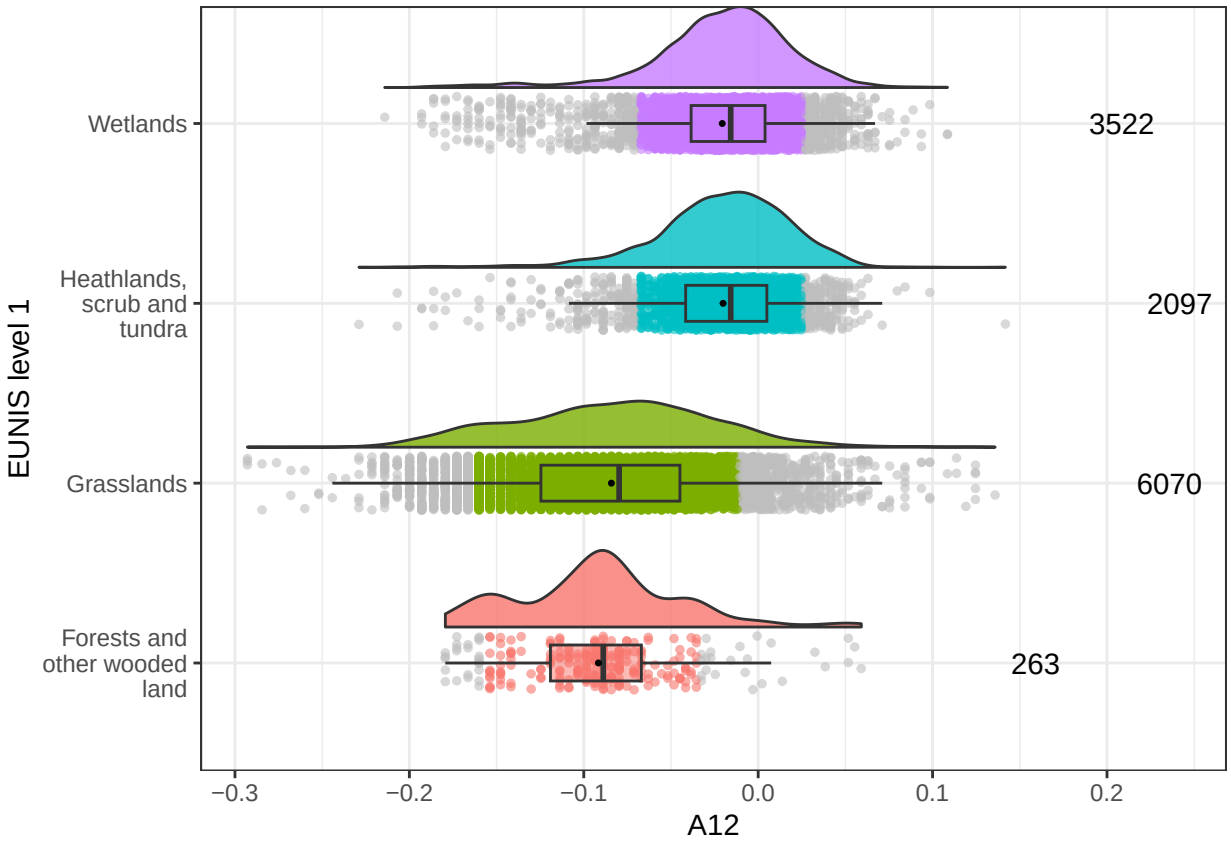

EUNIS level 1
Wetlands
Heathlands, scrub and tundra
Grasslands
Forests and other wooded land
3814
2364
6444
347
−0.3
−0.2
−0.1
0.0
0.1
0.2
A12

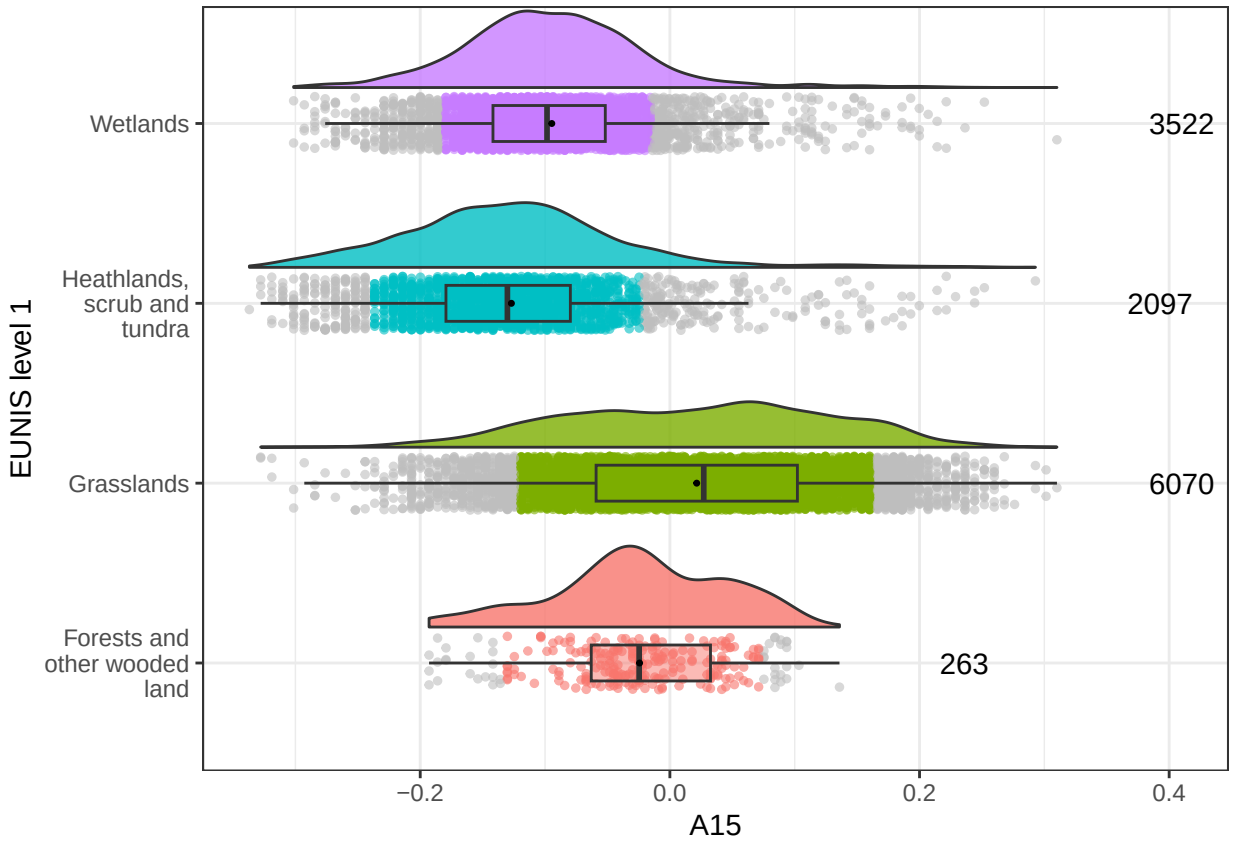

EUNIS level 1
Wetlands
Heathlands, scrub and tundra
Grasslands
Forests and other wooded land
3814
2364
6444
347
−0.4
−0.2
0.0
0.2
0.4
A15

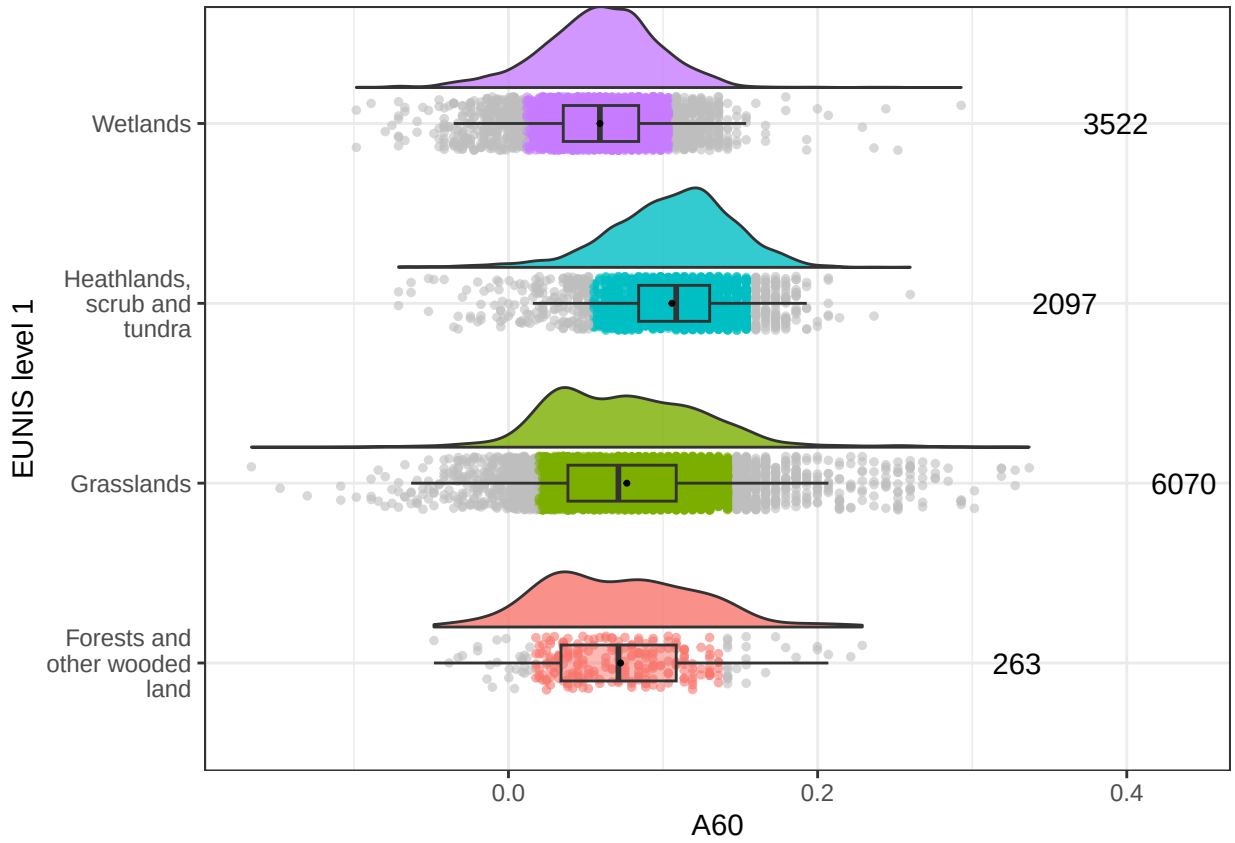



```
distr_plot_percentiles(
  # GPS points
  data_validation_roughvalid,
  c("A59", "A12", "A15", "A60"),
  c("A59", "A12", "A15", "A60"))
```


EUNIS level 1
Wetlands
Heathlands, scrub and tundra
Grasslands
Forests and other wooded land
3522
2097
6070
263
−0.2
0.0
0.2
A59


EUNIS level 1
Wetlands
Heathlands, scrub and tundra
Grasslands
Forests and other wooded land
3522
2097
6070
263
−0.3
−0.2
−0.1
0.0
0.1
0.2
A12


EUNIS level 1
Wetlands
Heathlands, scrub and tundra
Grasslands
Forests and other wooded land
3522
2097
6070
263
−0.2
0.0
0.2
0.4
A15



Calculate percentiles:

```
percentiles_novalid <-
  data_validation_novalid %>%
  group_by(EUNISa_1) %>%
  summarize(
    perc_10_A59 = quantile(A59, probs = 0.10, na.rm = T),
    perc_20_A59 = quantile(A59, probs = 0.20, na.rm = T),
    perc_80_A59 = quantile(A59, probs = 0.80, na.rm = T),
    perc_90_A59 = quantile(A59, probs = 0.90, na.rm = T),
    perc_10_A12 = quantile(A12, probs = 0.10, na.rm = T),
    perc_20_A12 = quantile(A12, probs = 0.20, na.rm = T),
    perc_80_A12 = quantile(A12, probs = 0.80, na.rm = T),
    perc_90_A12 = quantile(A12, probs = 0.90, na.rm = T),
    perc_10_A15 = quantile(A15, probs = 0.10, na.rm = T),
    perc_20_A15 = quantile(A15, probs = 0.20, na.rm = T),
    perc_80_A15 = quantile(A15, probs = 0.80, na.rm = T),
    perc_90_A15 = quantile(A15, probs = 0.90, na.rm = T),
    perc_10_A60 = quantile(A60, probs = 0.10, na.rm = T),
    perc_20_A60 = quantile(A60, probs = 0.20, na.rm = T),
    perc_80_A60 = quantile(A60, probs = 0.80, na.rm = T),
    perc_90_A60 = quantile(A60, probs = 0.90, na.rm = T),
    )
```

```
percentiles_roughvalid <-
  data_validation_roughvalid %>%
```

```
  group_by(EUNISa_1) %>%
  summarize(
    perc_10_A59 = quantile(A59, probs = 0.10, na.rm = T),
    perc_20_A59 = quantile(A59, probs = 0.20, na.rm = T),
    perc_80_A59 = quantile(A59, probs = 0.80, na.rm = T),
    perc_90_A59 = quantile(A59, probs = 0.90, na.rm = T),
    perc_10_A12 = quantile(A12, probs = 0.10, na.rm = T),
    perc_20_A12 = quantile(A12, probs = 0.20, na.rm = T),
    perc_80_A12 = quantile(A12, probs = 0.80, na.rm = T),
    perc_90_A12 = quantile(A12, probs = 0.90, na.rm = T),
    perc_10_A15 = quantile(A15, probs = 0.10, na.rm = T),
    perc_20_A15 = quantile(A15, probs = 0.20, na.rm = T),
    perc_80_A15 = quantile(A15, probs = 0.80, na.rm = T),
    perc_90_A15 = quantile(A15, probs = 0.90, na.rm = T),
    perc_10_A60 = quantile(A60, probs = 0.10, na.rm = T),
    perc_20_A60 = quantile(A60, probs = 0.20, na.rm = T),
    perc_80_A60 = quantile(A60, probs = 0.80, na.rm = T),
    perc_90_A60 = quantile(A60, probs = 0.90, na.rm = T),
    )
```

Get filtered data

```
# No validation

# Refin 10-90th, 1 variable
data_validation_novalid_refin1 <- data_validation_novalid %>%
  left_join(percentiles_novalid, by = "EUNISa_1") %>%
  mutate(EUNISa_1 = as.factor(EUNISa_1)) %>%
  dplyr::filter((A59 >= perc_10_A59 & A59 <= perc_90_A59))

# Refin 10-90th, 2 variables
data_validation_novalid_refin2 <- data_validation_novalid %>%
  left_join(percentiles_novalid, by = "EUNISa_1") %>%
  mutate(EUNISa_1 = as.factor(EUNISa_1)) %>%
  dplyr::filter((A59 >= perc_10_A59 & A59 <= perc_90_A59) &
                  (A12 >= perc_10_A12 & A12 <= perc_90_A12))

# Refin 10-90th, 3 variables
data_validation_novalid_refin3 <- data_validation_novalid %>%
  left_join(percentiles_novalid, by = "EUNISa_1") %>%
  mutate(EUNISa_1 = as.factor(EUNISa_1)) %>%
  dplyr::filter((A59 >= perc_10_A59 & A59 <= perc_90_A59) &
                  (A15 >= perc_10_A12 & A12 <= perc_90_A12) &
                  (A12 >= perc_10_A15 & A15 <= perc_90_A15))

# Refin 10-90th, 4 variables
data_validation_novalid_refin4 <- data_validation_novalid %>%
  left_join(percentiles_novalid, by = "EUNISa_1") %>%
  mutate(EUNISa_1 = as.factor(EUNISa_1)) %>%
  dplyr::filter((A59 >= perc_10_A59 & A59 <= perc_90_A59) &
                  (A15 >= perc_10_A12 & A12 <= perc_90_A12) &
                  (A12 >= perc_10_A15 & A15 <= perc_90_A15) &
```

```
                  (A60 >= perc_10_A60 & A60 <= perc_90_A60))

# Rough validation

# Refin 10-90th, 1 variable
data_validation_roughvalid_refin1 <- data_validation_roughvalid %>%
  left_join(percentiles_roughvalid, by = "EUNISa_1") %>%
  mutate(EUNISa_1 = as.factor(EUNISa_1)) %>%
  dplyr::filter((A59 >= perc_10_A59 & A59 <= perc_90_A59))

# Refin 10-90th, 2 variables
data_validation_roughvalid_refin2 <- data_validation_roughvalid %>%
  left_join(percentiles_roughvalid, by = "EUNISa_1") %>%
  mutate(EUNISa_1 = as.factor(EUNISa_1)) %>%
  dplyr::filter((A59 >= perc_10_A59 & A59 <= perc_90_A59) &
                  (A12 >= perc_10_A12 & A12 <= perc_90_A12))

# Refin 10-90th, 3 variables
data_validation_roughvalid_refin3 <- data_validation_roughvalid %>%
  left_join(percentiles_roughvalid, by = "EUNISa_1") %>%
  mutate(EUNISa_1 = as.factor(EUNISa_1)) %>%
  dplyr::filter((A59 >= perc_10_A59 & A59 <= perc_90_A59) &
                  (A15 >= perc_10_A12 & A12 <= perc_90_A12) &
                  (A12 >= perc_10_A15 & A15 <= perc_90_A15))

# Refin 10-90th, 4 variables
data_validation_roughvalid_refin4 <- data_validation_roughvalid %>%
  left_join(percentiles_roughvalid, by = "EUNISa_1") %>%
  mutate(EUNISa_1 = as.factor(EUNISa_1)) %>%
  dplyr::filter((A59 >= perc_10_A59 & A59 <= perc_90_A59) &
                  (A15 >= perc_10_A12 & A12 <= perc_90_A12) &
                  (A12 >= perc_10_A15 & A15 <= perc_90_A15) &
                  (A60 >= perc_10_A60 & A60 <= perc_90_A60))
```

N points per category

```
bind_rows(
  data_validation_novalid_refin1 %>% count(EUNISa_1) %>%
    pivot_wider(names_from = EUNISa_1, values_from = n) %>%
    mutate(data = "data_validation_novalid_refin1") %>%
    select(data, Q, R, S, T),
  data_validation_novalid_refin2 %>% count(EUNISa_1) %>%
    pivot_wider(names_from = EUNISa_1, values_from = n) %>%
    mutate(data = "data_validation_novalid_refin2") %>%
    select(data, Q, R, S, T),
  data_validation_novalid_refin3 %>% count(EUNISa_1) %>%
    pivot_wider(names_from = EUNISa_1, values_from = n) %>%
    mutate(data = "data_validation_novalid_refin3") %>%
    select(data, Q, R, S, T),
  data_validation_novalid_refin3 %>% count(EUNISa_1) %>%
    pivot_wider(names_from = EUNISa_1, values_from = n) %>%
```

```
    mutate(data = "data_validation_novalid_refin3") %>%
    select(data, Q, R, S, T),
  data_validation_roughvalid_refin1 %>% count(EUNISa_1) %>%
    pivot_wider(names_from = EUNISa_1, values_from = n) %>%
    mutate(data = "data_validation_roughvalid_refin1") %>%
    select(data, Q, R, S, T),
  data_validation_roughvalid_refin2 %>% count(EUNISa_1) %>%
    pivot_wider(names_from = EUNISa_1, values_from = n) %>%
    mutate(data = "data_validation_roughvalid_refin2") %>%
    select(data, Q, R, S, T),
  data_validation_roughvalid_refin3 %>% count(EUNISa_1) %>%
    pivot_wider(names_from = EUNISa_1, values_from = n) %>%
    mutate(data = "data_validation_roughvalid_refin3") %>%
    select(data, Q, R, S, T),
  data_validation_roughvalid_refin3 %>% count(EUNISa_1) %>%
    pivot_wider(names_from = EUNISa_1, values_from = n) %>%
    mutate(data = "data_validation_roughvalid_refin3") %>%
    select(data, Q, R, S, T),
  )
```

```
## # A tibble: 8 x 5
##   data                                Q     R     S     T
##   <chr>                           <int> <int> <int> <int>
## 1 data_validation_novalid_refin1   3126  5189  1894   281
## 2 data_validation_novalid_refin2   2612  4222  1588   234
## 3 data_validation_novalid_refin3    598  3164   193   172
## 4 data_validation_novalid_refin3    598  3164   193   172
## 5 data_validation_roughvalid_refin1  2896  4871  1737   213
## 6 data_validation_roughvalid_refin2  2448  3963  1459   179
## 7 data_validation_roughvalid_refin3   668  2848   188   129
## 8 data_validation_roughvalid_refin3   668  2848   188   129
```

Split into training and test data sets

```
set.seed(123)
```

```
train_indices_novalid_refin1 <- sample(1:nrow(data_validation_novalid_refin1),
                         0.7 * nrow(data_validation_novalid_refin1))
train_indices_novalid_refin2 <- sample(1:nrow(data_validation_novalid_refin2),
                         0.7 * nrow(data_validation_novalid_refin2))
train_indices_novalid_refin3 <- sample(1:nrow(data_validation_novalid_refin3),
                         0.7 * nrow(data_validation_novalid_refin3))
train_indices_novalid_refin4 <- sample(1:nrow(data_validation_novalid_refin4),
                         0.7 * nrow(data_validation_novalid_refin4))
train_indices_roughvalid_refin1 <- sample(1:nrow(data_validation_roughvalid_refin1),
                         0.7 * nrow(data_validation_roughvalid_refin1))
train_indices_roughvalid_refin2 <- sample(1:nrow(data_validation_roughvalid_refin2),
                         0.7 * nrow(data_validation_roughvalid_refin2))
train_indices_roughvalid_refin3 <- sample(1:nrow(data_validation_roughvalid_refin3),
                         0.7 * nrow(data_validation_roughvalid_refin3))
train_indices_roughvalid_refin4 <- sample(1:nrow(data_validation_roughvalid_refin4),
                         0.7 * nrow(data_validation_roughvalid_refin4))
```

```
train_data_novalid_refin1 <- data_validation_novalid_refin1[train_indices_novalid_refin1, ]
train_data_novalid_refin2 <- data_validation_novalid_refin2[train_indices_novalid_refin2, ]
train_data_novalid_refin3 <- data_validation_novalid_refin3[train_indices_novalid_refin3, ]
train_data_novalid_refin4 <- data_validation_novalid_refin4[train_indices_novalid_refin4, ]
train_data_roughvalid_refin1 <- data_validation_roughvalid_refin1[train_indices_roughvalid_refin1, ]
train_data_roughvalid_refin2 <- data_validation_roughvalid_refin2[train_indices_roughvalid_refin2, ]
train_data_roughvalid_refin3 <- data_validation_roughvalid_refin3[train_indices_roughvalid_refin3, ]
train_data_roughvalid_refin4 <- data_validation_roughvalid_refin4[train_indices_roughvalid_refin4, ]
```

```
test_data_novalid_refin1 <- data_validation_novalid_refin1[-train_indices_novalid_refin1, ]
test_data_novalid_refin2 <- data_validation_novalid_refin2[-train_indices_novalid_refin2, ]
test_data_novalid_refin3 <- data_validation_novalid_refin3[-train_indices_novalid_refin3, ]
test_data_novalid_refin4 <- data_validation_novalid_refin4[-train_indices_novalid_refin4, ]

test_data_roughvalid_refin1 <- data_validation_roughvalid_refin1[-train_indices_roughvalid_refin1, ]
test_data_roughvalid_refin2 <- data_validation_roughvalid_refin2[-train_indices_roughvalid_refin2, ]
test_data_roughvalid_refin3 <- data_validation_roughvalid_refin3[-train_indices_roughvalid_refin3, ]
test_data_roughvalid_refin4 <- data_validation_roughvalid_refin4[-train_indices_roughvalid_refin4, ]
```

Fit models

```
print(rf5$time)
```

```
##    user  system elapsed
##    5.34    2.25   69.94
```

```
print(rf6$time)
```

```
##    user  system elapsed
##    5.01    1.86   51.52
```

```
print(rf7$time)
```

```
##    user  system elapsed
##    4.02    2.09   52.42
```

```
print(rf8$time)
```

```
##    user  system elapsed
##    3.86    1.51   57.29
```

```
print(rf9$time)
```

```
##    user  system elapsed
##    2.09    0.73   38.61
```

```r
print(rf10$time)
```

```
##    user  system elapsed
##    1.96    0.60   36.39
```

```r
print(rf11$time)
```

```
##    user  system elapsed
##    1.56    0.56   38.33
```

```r
print(rf12$time)
```

```
##    user  system elapsed
##    1.39    0.58   34.85
```

```r
print(rf13$time)
```

```
##    user  system elapsed
##    4.75    1.89   61.17
```

```r
print(rf14$time)
```

```
##    user  system elapsed
##    4.53    1.64   54.57
```

```r
print(rf15$time)
```

```
##    user  system elapsed
##    3.69    1.24   44.80
```

```r
print(rf16$time)
```

```
##    user  system elapsed
##    3.51    1.41   45.45
```

```r
print(rf17$time)
```

```
##    user  system elapsed
##    1.73    0.87   35.81
```

```r
print(rf18$time)
```

```
##    user  system elapsed
##    1.98    0.78   36.81
```

```r
print(rf19$time)
```

```
##    user  system elapsed
##    1.46    0.46   38.72
```

```r
print(rf20$time)
```

```
##    user  system elapsed
##    1.61    0.50   38.31
```

Predictions

Confusion matrices

```r
# 5: No validation, refin1, SatEmb bands
confusionMatrix(predictions_rf5, test_data_novalid_refin1$EUNISa_1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    Q    R    S    T
##          Q  797   80   54    7
##          R   75 1466   53   23
##          S   65   38  425    0
##          T    0    2    3   60
##
## Overall Statistics
##
##                Accuracy : 0.8729
##                  95% CI : (0.8608, 0.8844)
##     No Information Rate : 0.5038
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7962
##
##  Mcnemar's Test P-Value : 2.231e-05
##
## Statistics by Class:
##
##                      Class: Q Class: R Class: S Class: T
## Sensitivity            0.8506   0.9243   0.7944  0.66667
## Specificity            0.9362   0.9033   0.9606  0.99836
## Pos Pred Value         0.8497   0.9066   0.8049  0.92308
## Neg Pred Value         0.9367   0.9216   0.9580  0.99027
## Prevalence             0.2976   0.5038   0.1699  0.02859
## Detection Rate         0.2532   0.4657   0.1350  0.01906
## Detection Prevalence   0.2980   0.5137   0.1677  0.02065
## Balanced Accuracy      0.8934   0.9138   0.8775  0.83252
```

```r
# 6: No validation, refin1, SatEmb bands + CH
confusionMatrix(predictions_rf6, test_data_novalid_refin1$EUNISa_1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    Q    R    S    T
##          Q  801   78   60    3
##          R   64 1472   58   26
##          S   72   32  415    1
##          T    0    4    2   60
##
## Overall Statistics
##
##                Accuracy : 0.8729
##                  95% CI : (0.8608, 0.8844)
##     No Information Rate : 0.5038
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.796
##
##  Mcnemar's Test P-Value : 5.002e-05
##
## Statistics by Class:
##
##                      Class: Q Class: R Class: S Class: T
## Sensitivity            0.8549   0.9281   0.7757  0.66667
## Specificity            0.9362   0.9052   0.9598  0.99804
## Pos Pred Value         0.8503   0.9086   0.7981  0.90909
## Neg Pred Value         0.9383   0.9254   0.9543  0.99027
## Prevalence             0.2976   0.5038   0.1699  0.02859
## Detection Rate         0.2544   0.4676   0.1318  0.01906
## Detection Prevalence   0.2992   0.5146   0.1652  0.02097
## Balanced Accuracy      0.8955   0.9167   0.8678  0.83235
```

```r
# 7: No validation, refin2, SatEmb bands
confusionMatrix(predictions_rf7, test_data_novalid_refin2$EUNISa_1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    Q    R    S    T
##          Q  695   55   61    0
##          R   29 1207   23   19
##          S   47   19  387    0
##          T    1    8    0   46
##
## Overall Statistics
##
##                Accuracy : 0.8991
##                  95% CI : (0.8869, 0.9104)
##     No Information Rate : 0.4963
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.8401
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Q Class: R Class: S Class: T
## Sensitivity            0.9003   0.9364   0.8217  0.70769
## Specificity            0.9364   0.9457   0.9690  0.99645
## Pos Pred Value         0.8570   0.9444   0.8543  0.83636
## Neg Pred Value         0.9569   0.9378   0.9608  0.99253
## Prevalence             0.2973   0.4963   0.1814  0.02503
## Detection Rate         0.2676   0.4648   0.1490  0.01771
## Detection Prevalence   0.3123   0.4921   0.1744  0.02118
## Balanced Accuracy      0.9183   0.9411   0.8953  0.85207
```

```
# 8: No validation, refin2, SatEmb bands + CH
confusionMatrix(predictions_rf8, test_data_novalid_refin2$EUNISa_1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    Q    R    S    T
##          Q  687   52   56    1
##          R   33 1206   29   15
##          S   52   23  386    0
##          T    0    8    0   49
##
## Overall Statistics
##
##                Accuracy : 0.8964
##                  95% CI : (0.8841, 0.9079)
##     No Information Rate : 0.4963
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8358
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Q Class: R Class: S Class: T
## Sensitivity            0.8899   0.9356   0.8195  0.75385
## Specificity            0.9403   0.9411   0.9647  0.99684
## Pos Pred Value         0.8631   0.9400   0.8373  0.85965
## Neg Pred Value         0.9528   0.9368   0.9602  0.99370
## Prevalence             0.2973   0.4963   0.1814  0.02503
## Detection Rate         0.2645   0.4644   0.1486  0.01887
## Detection Prevalence   0.3065   0.4940   0.1775  0.02195
## Balanced Accuracy      0.9151   0.9384   0.8921  0.87534
```

```r
# 9 : No validation, refin3, SatEmb bands
confusionMatrix(predictions_rf9, test_data_novalid_refin3$EUNISa_1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   Q   R   S   T
##          Q 123  13   4   0
##          R  65 910  26  16
##          S   1   8  23   0
##          T   3   1   1  45
##
## Overall Statistics
##
##                Accuracy : 0.8886
##                  95% CI : (0.8698, 0.9056)
##     No Information Rate : 0.7522
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6923
##
##  Mcnemar's Test P-Value : 9.902e-12
##
## Statistics by Class:
##
##                      Class: Q Class: R Class: S Class: T
## Sensitivity           0.64062   0.9764  0.42593  0.73770
## Specificity           0.98376   0.6515  0.99241  0.99576
## Pos Pred Value        0.87857   0.8948  0.71875  0.90000
## Neg Pred Value        0.93722   0.9009  0.97432  0.98654
## Prevalence            0.15496   0.7522  0.04358  0.04923
## Detection Rate        0.09927   0.7345  0.01856  0.03632
## Detection Prevalence  0.11299   0.8208  0.02583  0.04036
## Balanced Accuracy     0.81219   0.8139  0.70917  0.86673
```

```r
# 10 : No validation, refin3, SatEmb bands + CH
confusionMatrix(predictions_rf10, test_data_novalid_refin3$EUNISa_1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   Q   R   S   T
##          Q 119  12   5   0
##          R  69 911  21  15
##          S   2   6  27   0
##          T   2   3   1  46
##
## Overall Statistics
##
##                Accuracy : 0.8902
##                  95% CI : (0.8715, 0.9071)
##     No Information Rate : 0.7522
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                  Kappa : 0.6975
##
##  Mcnemar's Test P-Value : 3.198e-11
##
## Statistics by Class:
##
##                      Class: Q Class: R Class: S Class: T
## Sensitivity           0.61979   0.9775  0.50000  0.75410
## Specificity           0.98376   0.6580  0.99325  0.99491
## Pos Pred Value        0.87500   0.8967  0.77143  0.88462
## Neg Pred Value        0.93382   0.9058  0.97757  0.98736
## Prevalence            0.15496   0.7522  0.04358  0.04923
## Detection Rate        0.09605   0.7353  0.02179  0.03713
## Detection Prevalence  0.10977   0.8200  0.02825  0.04197
## Balanced Accuracy     0.80178   0.8177  0.74662  0.87450
```

```
# 11 : No validation, refin4, SatEmb bands
confusionMatrix(predictions_rf11, test_data_novalid_refin4$EUNISa_1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   Q   R   S   T
##          Q  96   7   3   1
##          R  51 775  17  18
##          S   1   4  41   0
##          T   0   7   1  30
##
## Overall Statistics
##
##                Accuracy : 0.8954
##                  95% CI : (0.8754, 0.9133)
##     No Information Rate : 0.7538
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7131
##
##  Mcnemar's Test P-Value : 6.593e-09
##
## Statistics by Class:
##
##                      Class: Q Class: R Class: S Class: T
## Sensitivity           0.64865   0.9773  0.66129  0.61224
## Specificity           0.98783   0.6680  0.99495  0.99202
## Pos Pred Value        0.89720   0.9001  0.89130  0.78947
## Neg Pred Value        0.94497   0.9058  0.97913  0.98126
## Prevalence            0.14068   0.7538  0.05894  0.04658
## Detection Rate        0.09125   0.7367  0.03897  0.02852
## Detection Prevalence  0.10171   0.8184  0.04373  0.03612
## Balanced Accuracy     0.81824   0.8226  0.82812  0.80213
```

```r
# 12 : No validation, refin4, SatEmb bands + CH
confusionMatrix(predictions_rf12, test_data_novalid_refin4$EUNISa_1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   Q   R   S   T
##          Q 103   9   2   0
##          R  43 771  23  12
##          S   2   7  36   0
##          T   0   6   1  37
##
## Overall Statistics
##
##                Accuracy : 0.9002
##                  95% CI : (0.8805, 0.9176)
##     No Information Rate : 0.7538
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7317
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Q Class: R Class: S Class: T
## Sensitivity           0.69595   0.9723  0.58065  0.75510
## Specificity           0.98783   0.6988  0.99091  0.99302
## Pos Pred Value        0.90351   0.9081  0.80000  0.84091
## Neg Pred Value        0.95203   0.8916  0.97418  0.98810
## Prevalence            0.14068   0.7538  0.05894  0.04658
## Detection Rate        0.09791   0.7329  0.03422  0.03517
## Detection Prevalence  0.10837   0.8070  0.04278  0.04183
## Balanced Accuracy     0.84189   0.8355  0.78578  0.87406
```

```r
# 13: Rough validation, refin1, SatEmb bands
confusionMatrix(predictions_rf13, test_data_roughvalid_refin1$EUNISa_1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    Q    R    S    T
##          Q  763   60   69    2
##          R   79 1366   53    9
##          S   57    9  392    0
##          T    1    2    2   52
##
## Overall Statistics
##
##                Accuracy : 0.8824
##                  95% CI : (0.8701, 0.8938)
##     No Information Rate : 0.4928
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                  Kappa : 0.811
##
##  Mcnemar's Test P-Value : 2.057e-07
##
## Statistics by Class:
##
##                      Class: Q Class: R Class: S Class: T
## Sensitivity            0.8478   0.9506   0.7597  0.82540
## Specificity            0.9350   0.9047   0.9725  0.99825
## Pos Pred Value         0.8535   0.9064   0.8559  0.91228
## Neg Pred Value         0.9322   0.9496   0.9496  0.99615
## Prevalence             0.3086   0.4928   0.1770  0.02160
## Detection Rate         0.2617   0.4684   0.1344  0.01783
## Detection Prevalence   0.3066   0.5168   0.1571  0.01955
## Balanced Accuracy      0.8914   0.9276   0.8661  0.91182
```

```
# 14: Rough validation, refin1, SatEmb bands + CH
confusionMatrix(predictions_rf14, test_data_roughvalid_refin1$EUNISa_1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    Q    R    S    T
##          Q  778   61   70    0
##          R   77 1362   50    2
##          S   45   13  396    0
##          T    0    1    0   61
##
## Overall Statistics
##
##                Accuracy : 0.8906
##                  95% CI : (0.8787, 0.9017)
##     No Information Rate : 0.4928
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8246
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Q Class: R Class: S Class: T
## Sensitivity            0.8644   0.9478   0.7674  0.96825
## Specificity            0.9350   0.9128   0.9758  0.99965
## Pos Pred Value         0.8559   0.9135   0.8722  0.98387
## Neg Pred Value         0.9392   0.9474   0.9513  0.99930
## Prevalence             0.3086   0.4928   0.1770  0.02160
## Detection Rate         0.2668   0.4671   0.1358  0.02092
## Detection Prevalence   0.3117   0.5113   0.1557  0.02126
## Balanced Accuracy      0.8997   0.9303   0.8716  0.98395
```

```r
# 15: Rough validation, refin2, SatEmb bands
confusionMatrix(predictions_rf15, test_data_roughvalid_refin2$EUNISa_1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    Q    R    S    T
##          Q  650   47   49    0
##          R   30 1158   21    3
##          S   43   18  345    0
##          T    0    2    0   49
##
## Overall Statistics
##
##                Accuracy : 0.9118
##                  95% CI : (0.8998, 0.9228)
##     No Information Rate : 0.5072
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8586
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Q Class: R Class: S Class: T
## Sensitivity            0.8990   0.9453   0.8313  0.94231
## Specificity            0.9433   0.9546   0.9695  0.99915
## Pos Pred Value         0.8713   0.9554   0.8498  0.96078
## Neg Pred Value         0.9563   0.9443   0.9652  0.99873
## Prevalence             0.2994   0.5072   0.1718  0.02153
## Detection Rate         0.2692   0.4795   0.1429  0.02029
## Detection Prevalence   0.3089   0.5019   0.1681  0.02112
## Balanced Accuracy      0.9211   0.9500   0.9004  0.97073
```

```r
# 16: Rough validation, refin2, SatEmb bands + CH
confusionMatrix(predictions_rf16, test_data_roughvalid_refin2$EUNISa_1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    Q    R    S    T
##          Q  651   36   56    0
##          R   36 1169   26    0
##          S   36   20  333    0
##          T    0    0    0   52
##
## Overall Statistics
##
##                Accuracy : 0.913
##                  95% CI : (0.9011, 0.924)
##     No Information Rate : 0.5072
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.86
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Q Class: R Class: S Class: T
## Sensitivity            0.9004   0.9543   0.8024  1.00000
## Specificity            0.9456   0.9479   0.9720  1.00000
## Pos Pred Value         0.8762   0.9496   0.8560  1.00000
## Neg Pred Value         0.9569   0.9527   0.9595  1.00000
## Prevalence             0.2994   0.5072   0.1718  0.02153
## Detection Rate         0.2696   0.4841   0.1379  0.02153
## Detection Prevalence   0.3077   0.5097   0.1611  0.02153
## Balanced Accuracy      0.9230   0.9511   0.8872  1.00000
```

```
# 17 : Rough validation, refin3, SatEmb bands
confusionMatrix(predictions_rf17, test_data_roughvalid_refin3$EUNISa_1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   Q   R   S   T
##          Q 133  11  10   0
##          R  39 865  20   5
##          S   1   3  26   0
##          T   1   1   0  35
##
## Overall Statistics
##
##                Accuracy : 0.9209
##                  95% CI : (0.9037, 0.9358)
##     No Information Rate : 0.7652
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7797
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Q Class: R Class: S Class: T
## Sensitivity            0.7644   0.9830  0.46429  0.87500
## Specificity            0.9785   0.7630  0.99634  0.99820
## Pos Pred Value         0.8636   0.9311  0.86667  0.94595
## Neg Pred Value         0.9588   0.9321  0.97321  0.99551
## Prevalence             0.1513   0.7652  0.04870  0.03478
## Detection Rate         0.1157   0.7522  0.02261  0.03043
## Detection Prevalence   0.1339   0.8078  0.02609  0.03217
## Balanced Accuracy      0.8714   0.8730  0.73031  0.93660
```

```r
# 18 : Rough validation, refin3, SatEmb bands + CH
confusionMatrix(predictions_rf18, test_data_roughvalid_refin3$EUNISa_1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   Q   R   S   T
##          Q 130   7   6   0
##          R  43 869  23   1
##          S   1   4  27   0
##          T   0   0   0  39
##
## Overall Statistics
##
##                Accuracy : 0.9261
##                  95% CI : (0.9094, 0.9405)
##     No Information Rate : 0.7652
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7923
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Q Class: R Class: S Class: T
## Sensitivity            0.7471   0.9875  0.48214  0.97500
## Specificity            0.9867   0.7519  0.99543  1.00000
## Pos Pred Value         0.9091   0.9284  0.84375  1.00000
## Neg Pred Value         0.9563   0.9486  0.97406  0.99910
## Prevalence             0.1513   0.7652  0.04870  0.03478
## Detection Rate         0.1130   0.7557  0.02348  0.03391
## Detection Prevalence   0.1243   0.8139  0.02783  0.03391
## Balanced Accuracy      0.8669   0.8697  0.73879  0.98750
```

```r
# 19 : Rough validation, refin4, SatEmb bands
confusionMatrix(predictions_rf19, test_data_roughvalid_refin4$EUNISa_1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   Q   R   S   T
##          Q 136   8   4   0
##          R  29 721   9   1
##          S   0   6  27   0
##          T   1   4   0  29
##
## Overall Statistics
##
##                Accuracy : 0.9364
##                  95% CI : (0.9192, 0.9509)
##     No Information Rate : 0.7579
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.833
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Q Class: R Class: S Class: T
## Sensitivity            0.8193   0.9756  0.67500  0.96667
## Specificity            0.9852   0.8347  0.99358  0.99471
## Pos Pred Value         0.9189   0.9487  0.81818  0.85294
## Neg Pred Value         0.9637   0.9163  0.98620  0.99894
## Prevalence             0.1703   0.7579  0.04103  0.03077
## Detection Rate         0.1395   0.7395  0.02769  0.02974
## Detection Prevalence   0.1518   0.7795  0.03385  0.03487
## Balanced Accuracy      0.9022   0.9052  0.83429  0.98069
```

```
# 20 : Rough validation, refin4, SatEmb bands + CH
confusionMatrix(predictions_rf20, test_data_roughvalid_refin4$EUNISa_1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   Q   R   S   T
##          Q 134  16   3   0
##          R  32 717  12   0
##          S   0   6  25   0
##          T   0   0   0  30
##
## Overall Statistics
##
##                Accuracy : 0.9292
##                  95% CI : (0.9113, 0.9445)
##     No Information Rate : 0.7579
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8135
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Q Class: R Class: S Class: T
## Sensitivity            0.8072   0.9702  0.62500  1.00000
## Specificity            0.9765   0.8136  0.99358  1.00000
## Pos Pred Value         0.8758   0.9422  0.80645  1.00000
## Neg Pred Value         0.9611   0.8972  0.98411  1.00000
## Prevalence             0.1703   0.7579  0.04103  0.03077
## Detection Rate         0.1374   0.7354  0.02564  0.03077
## Detection Prevalence   0.1569   0.7805  0.03179  0.03077
## Balanced Accuracy      0.8919   0.8919  0.80929  1.00000
```

TBD: Variable importance

Session info

```
sessionInfo()
```

```
## R version 4.5.1 (2025-06-13 ucrt)
## Platform: x86_64-w64-mingw32/x64
## Running under: Windows 11 x64 (build 26100)
##
## Matrix products: default
##   LAPACK version 3.12.1
##
## locale:
## [1] LC_COLLATE=Spanish_Spain.utf8  LC_CTYPE=Spanish_Spain.utf8
## [3] LC_MONETARY=Spanish_Spain.utf8 LC_NUMERIC=C
## [5] LC_TIME=Spanish_Spain.utf8
##
## time zone: Europe/Madrid
## tzcode source: internal
##
## attached base packages:
##  [1] parallel  stats4    grid      stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
## other attached packages:
##  [1] yardstick_1.3.2      permimp_1.1-0        beepr_2.0
##  [4] rlang_1.1.6          corrplot_0.95        pROC_1.19.0.1
##  [7] randomForest_4.7-1.2 caret_7.0-1          lattice_0.22-7
## [10] ggparty_1.0.0.1      doParallel_1.0.17    iterators_1.0.14
## [13] foreach_1.5.2        moreparty_0.4.2      partykit_1.2-24
## [16] libcoin_1.0-10       party_1.3-18         strucchange_1.5-4
## [19] sandwich_3.1-1       zoo_1.8-14           modeltools_0.2-24
## [22] mvtnorm_1.3-3        ggeffects_2.3.1      car_3.1-3
## [25] carData_3.0-5        lmerTest_3.1-3       lme4_1.1-37
## [28] Matrix_1.7-4         dtplyr_1.3.2         rnaturalearth_1.1.0
## [31] sf_1.0-21            scales_1.4.0         readxl_1.4.5
## [34] gridExtra_2.3        here_1.0.2           lubridate_1.9.4
## [37] forcats_1.0.0        stringr_1.5.2        dplyr_1.1.4
## [40] purrr_1.1.0          readr_2.1.5          tidyr_1.3.1
## [43] tibble_3.3.0         ggplot2_4.0.0        tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
##   [1] RColorBrewer_1.1-3   audio_0.1-11         rstudioapi_0.17.1
##   [4] jsonlite_2.0.0       magrittr_2.0.3       TH.data_1.1-4
##   [7] farver_2.1.2         nloptr_2.2.1         rmarkdown_2.29
##  [10] vctrs_0.6.5          minqa_1.2.8          htmltools_0.5.8.1
##  [13] varImp_0.4           cellranger_1.1.0     Formula_1.2-5
##  [16] parallelly_1.45.1    sass_0.4.10          KernSmooth_2.23-26
##  [19] bslib_0.9.0          htmlwidgets_1.6.4    plyr_1.8.9
##  [22] cachem_1.1.0         mime_0.13            lifecycle_1.0.4
##  [25] pkgconfig_2.0.3      R6_2.6.1             fastmap_1.2.0
```

```
##  [28] future_1.67.0        rbibutils_2.3        shiny_1.11.1
##  [31] digest_0.6.37        numDeriv_2016.8-1.1  rprojroot_2.1.1
##  [34] labeling_0.4.3       timechange_0.3.0     abind_1.4-8
##  [37] compiler_4.5.1       proxy_0.4-27         bit64_4.6.0-1
##  [40] withr_3.0.2          S7_0.2.0             backports_1.5.0
##  [43] DBI_1.2.3            lava_1.8.1           MASS_7.3-65
##  [46] classInt_0.4-11      ModelMetrics_1.2.2.2 tools_4.5.1
##  [49] units_0.8-7          httpuv_1.6.16        future.apply_1.20.0
##  [52] nnet_7.3-20          glue_1.8.0           nlme_3.1-168
##  [55] promises_1.3.3       inum_1.0-5           checkmate_2.3.3
##  [58] reshape2_1.4.4       generics_0.1.4       recipes_1.3.1
##  [61] gtable_0.3.6         tzdb_0.5.0           class_7.3-23
##  [64] data.table_1.17.8    hms_1.1.3            utf8_1.2.6
##  [67] coin_1.4-3           pillar_1.11.0        vroom_1.6.5
##  [70] later_1.4.4          splines_4.5.1        bit_4.6.0
##  [73] survival_3.8-3       tidyselect_1.2.1     knitr_1.50
##  [76] reformulas_0.4.1     xfun_0.53            measures_0.3
##  [79] hardhat_1.4.2        timeDate_4041.110    matrixStats_1.5.0
##  [82] DT_0.34.0            stringi_1.8.7        phosphoricons_0.2.1
##  [85] yaml_2.3.10          boot_1.3-32          shinyWidgets_0.9.0
##  [88] evaluate_1.0.5       codetools_0.2-20     cli_3.6.5
##  [91] rpart_4.1.24         xtable_1.8-4         Rdpack_2.6.4
##  [94] jquerylib_0.1.4      Rcpp_1.1.0           globals_0.18.0
##  [97] gower_1.0.2          rclipboard_0.2.1     listenv_0.9.1
## [100] ipred_0.9-15         prodlim_2025.04.28   e1071_1.7-16
## [103] crayon_1.5.3         insight_1.4.2        multcomp_1.4-28
```