Script to validate points in ReSurvey database using RS data

Adding ATL_BENELUX and CON_NORDIC S2 data, some Landsat data, adding canopy height data

Alicia Valdés

11 abril 2025

This R script is used to validate the points in the ReSurvey database using RS indicators (NDVI, NDMI, canopy height).

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   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.2
## -- 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
```

```
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
```

```
library(sf)
```

```
## Linking to GEOS 3.13.0, GDAL 3.10.1, PROJ 9.5.1; sf_use_s2() is TRUE
```

```
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
```

```
library(lmerTest)
```

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

```
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)
```

Define printall function

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

Load geom_flat_violin plot

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

Read ReSurvey data with RS indicators

```
db_resurv_RS<-read_tsv(
  here("data", "clean", "db_resurv_RS_20250408.csv"),
  col_types = cols(
    # Dynamically specify EUNIS columns as character
    .default = col_guess(),  # Default guessing for other columns
    EUNISa = col_character(),
    EUNISb = col_character(),
    EUNISc = col_character(),
    EUNISd = col_character(),
    EUNISa_1 = col_character(),
    EUNISa_2 = col_character(),
    EUNISa_3 = col_character(),
    EUNISa_4 = col_character(),
    EUNISb_1 = col_character(),
    EUNISb_2 = col_character(),
    EUNISb_3 = col_character(),
    EUNISb_4 = col_character(),
    EUNISc_1 = col_character(),
    EUNISc_2 = col_character(),
    EUNISc_3 = col_character(),
    EUNISc_4 = col_character(),
    EUNISd_1 = col_character(),
    EUNISd_2 = col_character(),
    EUNISd_3 = col_character(),
    EUNISd_4 = col_character(),
    EUNISa_1_descr = col_character(),
    EUNISb_1_descr = col_character(),
    EUNISc_1_descr = col_character(),
    EUNISd_1_descr = col_character(),
    EUNIS_assignation = col_character(),
    EUNISa_2_descr = col_character(),
```

```
    EUNISa_3_descr = col_character(),
    EUNISa_4_descr = col_character(),
    EUNISb_2_descr = col_character(),
    EUNISb_3_descr = col_character(),
    EUNISb_4_descr = col_character(),
    EUNISc_2_descr = col_character(),
    EUNISc_3_descr = col_character(),
    EUNISc_4_descr = col_character(),
    EUNISd_2_descr = col_character(),
    EUNISd_3_descr = col_character(),
    EUNISd_4_descr = col_character()
    )
  )
```

No parsing issues!

Some data managenemt

Several EUNIS level 1 assigned

Number of rows where there is more than one EUNIS 1 assigned, and they are different among them. See what to do with these later! So far I take EUNISa_1.

```
nrow(db_resurv_RS %>%
       # Rows with more than one EUNIS 1 assigned
       filter(!is.na(EUNISb_1)) %>%
       filter(EUNISa_1!=EUNISb_1 | EUNISb_1 != EUNISc_1 | EUNISa_1 != EUNISc_1))
```

```
## [1] 102
```

See "confusions":

```
db_resurv_RS %>%
  # Rows with more than one EUNIS 1 assigned
  filter(!is.na(EUNISb_1)) %>%
  filter(EUNISa_1!=EUNISb_1 | EUNISb_1 != EUNISc_1 | EUNISa_1 != EUNISc_1) %>%
  distinct(EUNISa_1, EUNISb_1, EUNISc_1, EUNISd_1)
```

```
## # A tibble: 5 x 4
##   EUNISa_1 EUNISb_1 EUNISc_1 EUNISd_1
##   <chr>    <chr>    <chr>    <chr>
## 1 R        S        <NA>     <NA>
## 2 S        T        <NA>     <NA>
## 3 R        R        S        <NA>
## 4 R        R        S        S
## 5 P        Q        <NA>     <NA>
```

Define "confusion" columns:

```
db_resurv_RS <- db_resurv_RS %>%
  mutate(EUNIS1_conf_type = case_when(
    EUNISa_1 == "R" & EUNISb_1 == "S" ~ "R/S",
    EUNISa_1 == "S" & EUNISb_1 == "T" ~ "S/T",
    EUNISa_1 == "R" & EUNISb_1 == "R" & EUNISc_1 == "S" ~ "R/S",
    EUNISa_1 == "R" & EUNISb_1 == "R" & EUNISc_1 == "S" & EUNISd_1 == "S" ~ "R/S",
    EUNISa_1 == "P" & EUNISb_1 == "Q" ~ "P/Q",
    TRUE ~ NA_character_),
    EUNIS1_conf = !is.na(EUNIS1_conf_type))
```

Tibble with selected columns

```
db_resurv_RS_short <- db_resurv_RS %>%
  select(PlotObservationID, Country, RS_CODE, `ReSurvey site`, `ReSurvey plot`,
         `Manipulate (y/n)`, `Type of manipulation`, Lon_updated, Lat_updated,
         `Location method`, `Location uncertainty (m)`, EUNISa_1,
         EUNISa_1_descr, EUNISa_2, EUNISa_2_descr, EUNISa_3, EUNISa_3_descr,
         EUNISa_4, EUNISa_4_descr, EUNIS1_conf, EUNIS1_conf_type,
         date, year, biogeo, unit, year_RS, Lon_RS, Lat_RS,
         starts_with("NDVI"), starts_with("NDMI"), starts_with("NDWI"),
         starts_with("EVI"), starts_with("SAVI"), canopy_height,
         SOS_DOY, SOS_date, Peak_DOY, Peak_date, EOS_DOY, EOS_date,
         S2_data, Landsat_data, CH_data, S2_phen_data)
```

TO-DO: Missing data checks

Do when all RS data is ready!

Flag when year is different between RS data and ReSurvey db

```
db_resurv_RS_short <- db_resurv_RS_short %>%
  mutate(year_diff = year != year_RS)
```

```
db_resurv_RS_short %>% count(year_diff)
```

```
## # A tibble: 3 x 2
##   year_diff      n
##   <lgl>      <int>
## 1 FALSE      58865
## 2 TRUE           2
## 3 NA        364713
```

2 with different year. RS indices would need to be calculated again for those.

Flag when coordinates are different between RS data and ReSurvey db

```r
db_resurv_RS_short <- db_resurv_RS_short %>%
  mutate(Lon_diff = case_when(Lon_updated == Lon_RS ~ "NO",
                              # Sometimes they are only slighly different
                              abs(Lon_updated - Lon_RS) < 0.01 ~ "SMALL",
                              is.na(Lon_updated) | is.na(Lon_RS) ~ NA,
                              TRUE ~ "LARGE"),
         Lat_diff = case_when(Lat_updated == Lat_RS ~ "NO",
                              # Sometimes they are only slighly different
                              abs(Lat_updated - Lat_RS) < 0.01 ~ "SMALL",
                              is.na(Lat_updated) | is.na(Lat_RS) ~ NA,
                              TRUE ~ "LARGE"))
```

```r
db_resurv_RS_short %>% count(Lon_diff)
```

```
## # A tibble: 4 x 2
##   Lon_diff      n
##   <chr>     <int>
## 1 LARGE         2
## 2 NO        58617
## 3 SMALL       248
## 4 <NA>     364713
```

```r
db_resurv_RS_short %>% count(Lat_diff)
```

```
## # A tibble: 4 x 2
##   Lat_diff      n
##   <chr>     <int>
## 1 LARGE         6
## 2 NO        57077
## 3 SMALL      1784
## 4 <NA>     364713
```

Very few with large differences (2 for longitude, 6 for latitude). RS indices would need to be calculated again for those.

If year_diff is TRUE or Lon_diff is LARGE or Lat_diff is LARGE –> RS indices need to be recalculated. So far, remove those rows from db_resurv_RS_short.

```r
db_resurv_RS_short <- db_resurv_RS_short %>%
  filter(year_diff == FALSE | is.na(year_diff)) %>%
  filter(Lon_diff == "NO" |Lon_diff == "SMALL" | is.na(Lon_diff)) %>%
  filter(Lat_diff == "NO" |Lat_diff == "SMALL" | is.na(Lat_diff))
```

TO-DO: Recalculate RS indices for those above

Handle plots that have more than one obs per year

Add column PLOT to data to identify unique plots:

```
db_resurv_RS_short_PLOT <- db_resurv_RS_short %>%
  # Original names give problems, create new vars
  mutate(RS_site = `ReSurvey site`, RS_plot = `ReSurvey plot`) %>%
  # Convert to data.table for faster processing
  lazy_dt() %>%
  # Group by the 3 vars that uniquely identify each pl ot
  group_by(RS_CODE, RS_site, RS_plot) %>%
  # Create a new variable PLOT for each group
  mutate(PLOT = .GRP) %>%
  # Convert back to tibble
  as_tibble() %>%
  # Remove unneeded vars
  select(-RS_site, -RS_plot)
```

There should be only one observation of each plot per year.

Plots where there is at least a year with more than one observation, and where those observations have a different EUNIS assigned:

```
plots_to_remove <- db_resurv_RS_short_PLOT %>%
  group_by(PLOT, year) %>%
  summarize(EUNISa_1_n = n_distinct(EUNISa_1, na.rm = TRUE)) %>%
  ungroup() %>%
  filter(EUNISa_1_n > 1) %>%
  distinct(PLOT)
```

```
## `summarise()` has grouped output by 'PLOT'. You can override using the
## `.groups` argument.
```

Remove plots_to_remove from the database:

```
db_resurv_RS_short_PLOT <- db_resurv_RS_short_PLOT %>%
  anti_join(plots_to_remove, by = "PLOT")
```

Plots and years where there is more than one observation:

```
plots_to_merge <- db_resurv_RS_short_PLOT %>%
  group_by(PLOT, year) %>%
  # Plots that have more than one observation per year
  filter(n() > 1) %>%
  ungroup() %>%
  distinct(PLOT)
```

Summarize plots_to_merge:

```
plots_to_merge_summ <- db_resurv_RS_short_PLOT %>%
  group_by(PLOT, year) %>%
  # Plots that have more than one observation per year
  filter(n() > 1) %>%
  mutate(obs_num = row_number()) %>%
  pivot_wider(
    names_from = obs_num,
```

```
    values_from = c(date, PlotObservationID),
    names_prefix = "obs_"
  ) %>%
  arrange(PLOT) %>%
  summarize(
    across(c(Country, RS_CODE, `ReSurvey site`, `ReSurvey plot`,
             `Manipulate (y/n)`, `Type of manipulation`, Lon_updated,
             Lat_updated, `Location method`, `Location uncertainty (m)`,
             EUNISa_1, EUNISa_1_descr, EUNISa_2, EUNISa_2_descr, EUNISa_3,
             EUNISa_3_descr, EUNISa_4, EUNISa_4_descr, EUNIS1_conf,
             EUNIS1_conf_type, biogeo, unit, year_RS, Lon_RS, Lat_RS, NDVI_max,
             NDVI_median, NDVI_min, NDVI_mode, NDVI_p10, NDVI_p90, NDMI_max,
             NDMI_median, NDMI_min, NDMI_mode, NDMI_p10, NDMI_p90, NDWI_max,
             NDWI_median, NDWI_min, NDWI_mode, NDWI_p10, NDWI_p90, EVI_max,
             EVI_median, EVI_min, EVI_mode, EVI_p10, EVI_p90, SAVI_max,
             SAVI_median, SAVI_min, SAVI_mode, SAVI_p10, SAVI_p90,
             canopy_height, SOS_DOY, SOS_date, Peak_DOY, Peak_date, EOS_DOY,
             EOS_date, S2_data, Landsat_data, CH_data, S2_phen_data, year_diff,
             Lon_diff, Lat_diff), first),
    across(starts_with("date_obs_"), min),
    across(starts_with("PlotObservationID_obs_"), min)
    ) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'PLOT'. You can override using the
## `.groups` argument.
```

Remove plots_to_merge from the database:

```
db_resurv_RS_short_PLOT <- db_resurv_RS_short_PLOT %>%
  anti_join(plots_to_merge, by = "PLOT")
```

And add plots_to_merge_summ, where each plot and year only has one row:

```
db_resurv_RS_short_PLOT <- bind_rows(db_resurv_RS_short_PLOT,
                                     plots_to_merge_summ)
```

Check that there is only one row per plot and per year:

```
db_resurv_RS_short_PLOT %>%
  group_by(PLOT, year) %>%
  # Plots that have more than one observation per year
  filter(n() > 1)
```

```
## # A tibble: 0 x 153
## # Groups:   PLOT, year [0]
## # i 153 variables: PlotObservationID <dbl>, Country <chr>, RS_CODE <chr>,
## #   ReSurvey site <chr>, ReSurvey plot <chr>, Manipulate (y/n) <chr>,
## #   Type of manipulation <chr>, Lon_updated <dbl>, Lat_updated <dbl>,
## #   Location method <chr>, Location uncertainty (m) <dbl>, EUNISa_1 <chr>,
## #   EUNISa_1_descr <chr>, EUNISa_2 <chr>, EUNISa_2_descr <chr>, EUNISa_3 <chr>,
## #   EUNISa_3_descr <chr>, EUNISa_4 <chr>, EUNISa_4_descr <chr>,
## #   EUNIS1_conf <lgl>, EUNIS1_conf_type <chr>, date <date>, year <dbl>, ...
```

So, to sum up what I have done:

- Plots where there is at least a year with more than one observation, and where those observations have a different EUNIS assigned: Plots REMOVED from the data
- Plots where there is more than one observation, but observations have the same EUNIS assigned: kept in the data. Merged so that there is only one row per year. Info about the different dates (when different) is kept in columns date_obs_1 - date_obs_40, and info about the different PlotObservationID is kept in the columns PlotObservationID_obs_1 - PlotObservationID_obs_40.

Save to clean data

Save clean file for analyses (to be updated continuously due to updates in ReSurvey database and updates on RS data).

```
write_tsv(db_resurv_RS_short_PLOT,
          here("data", "clean","db_resurv_RS_short_PLOT_20250408.csv"))
```

Distributions all bioregions

```
# Define a function to create histograms
plot_histogram <- function(data, x_var, x_label) {
  ggplot(data %>%
           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()
}
```

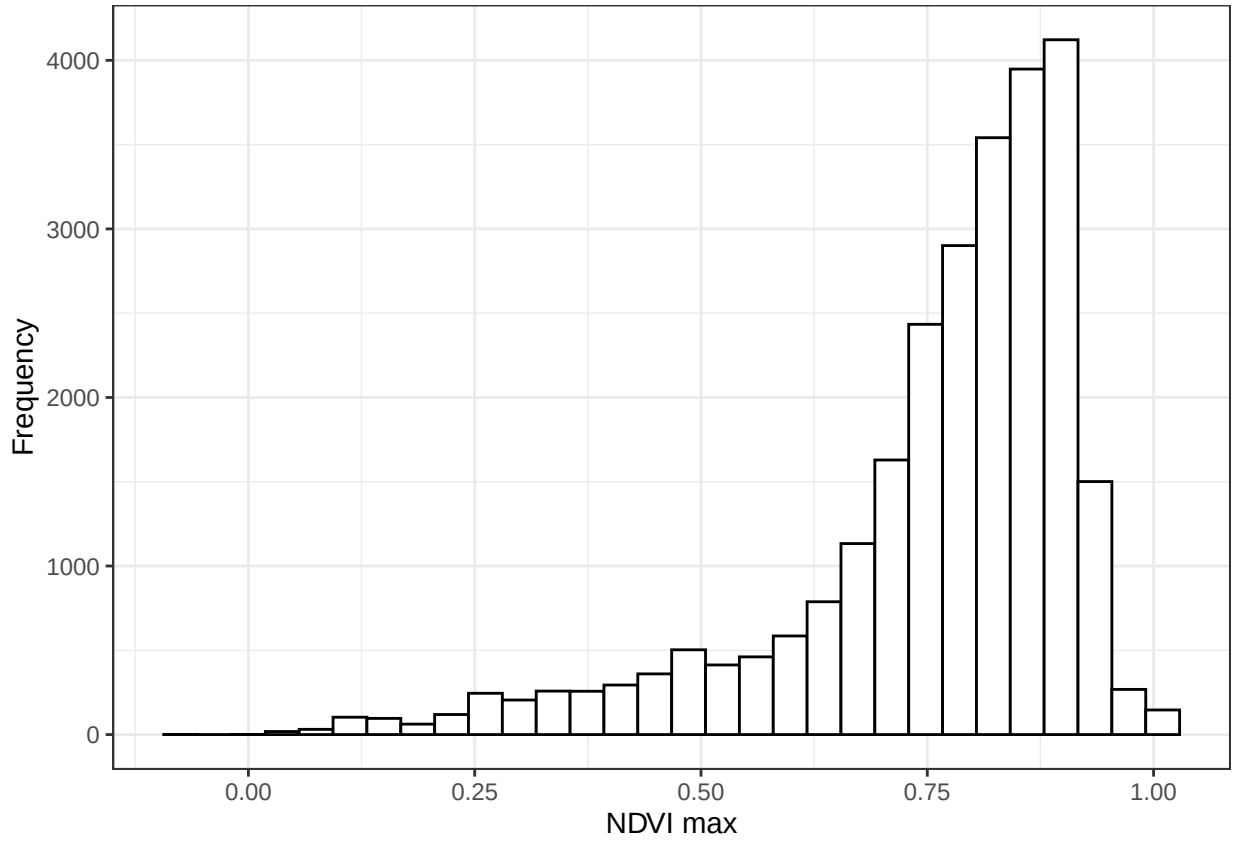
```
# Define a function to create plots with violin + boxplot + points
distr_plot <- function(data, y_var, y_label) {
  ggplot(data = data %>%
           filter(EUNISa_1 %in% c("T", "R", "S", "Q")),
         aes(x = EUNISa_1_descr, y = !!sym(y_var), fill = EUNISa_1_descr)) +
    geom_flat_violin(position = 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()
}
```

Max. NDVI, NDMI, NDWI, SAVI and EVI

Histograms to check that values are ok:

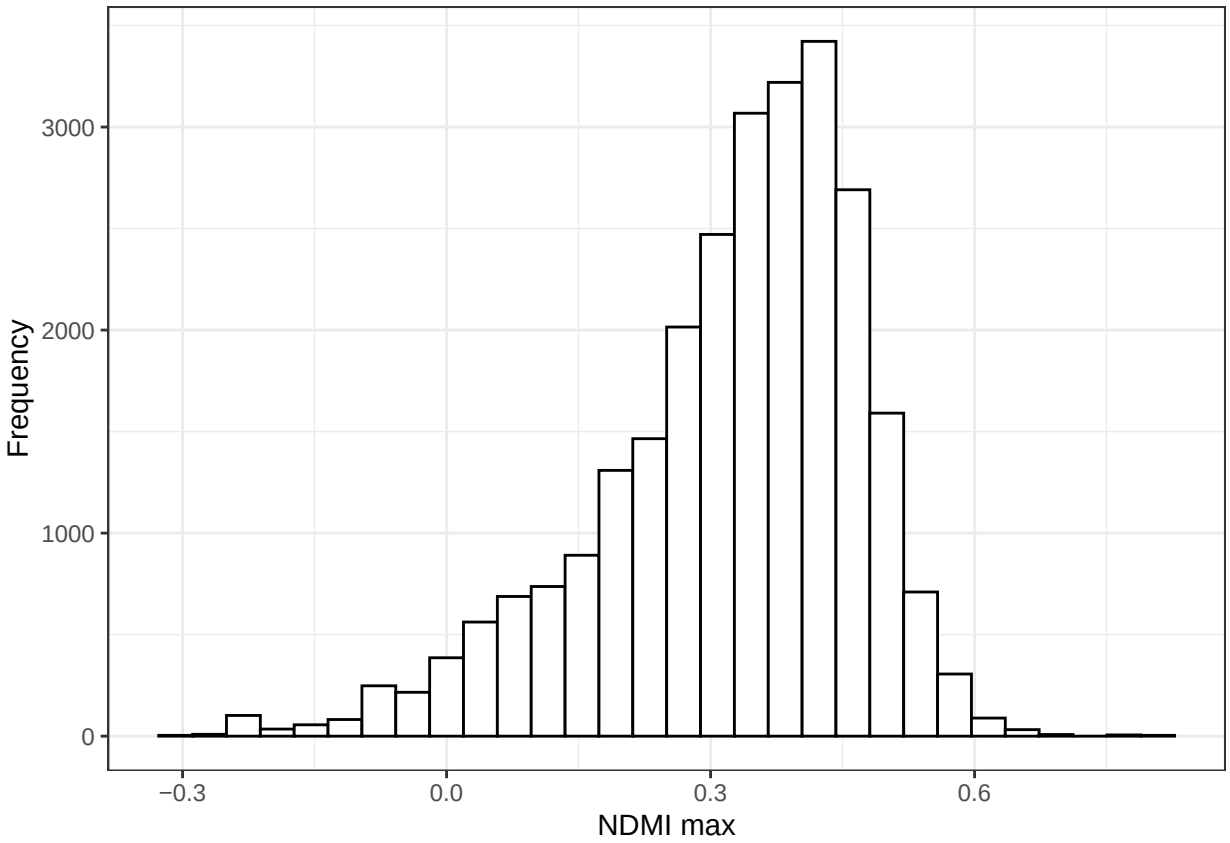
```
hist_NDVI <- plot_histogram(db_resurv_RS_short_PLOT, "NDVI_max", "NDVI max")
hist_NDMI <- plot_histogram(db_resurv_RS_short_PLOT, "NDMI_max", "NDMI max")
hist_NDWI <- plot_histogram(db_resurv_RS_short_PLOT, "NDWI_max", "NDWI max")
hist_SAVI <- plot_histogram(db_resurv_RS_short_PLOT, "SAVI_max", "SAVI max")
hist_EVI <- plot_histogram(db_resurv_RS_short_PLOT, "EVI_max", "EVI max")
hist_NDVI
```

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



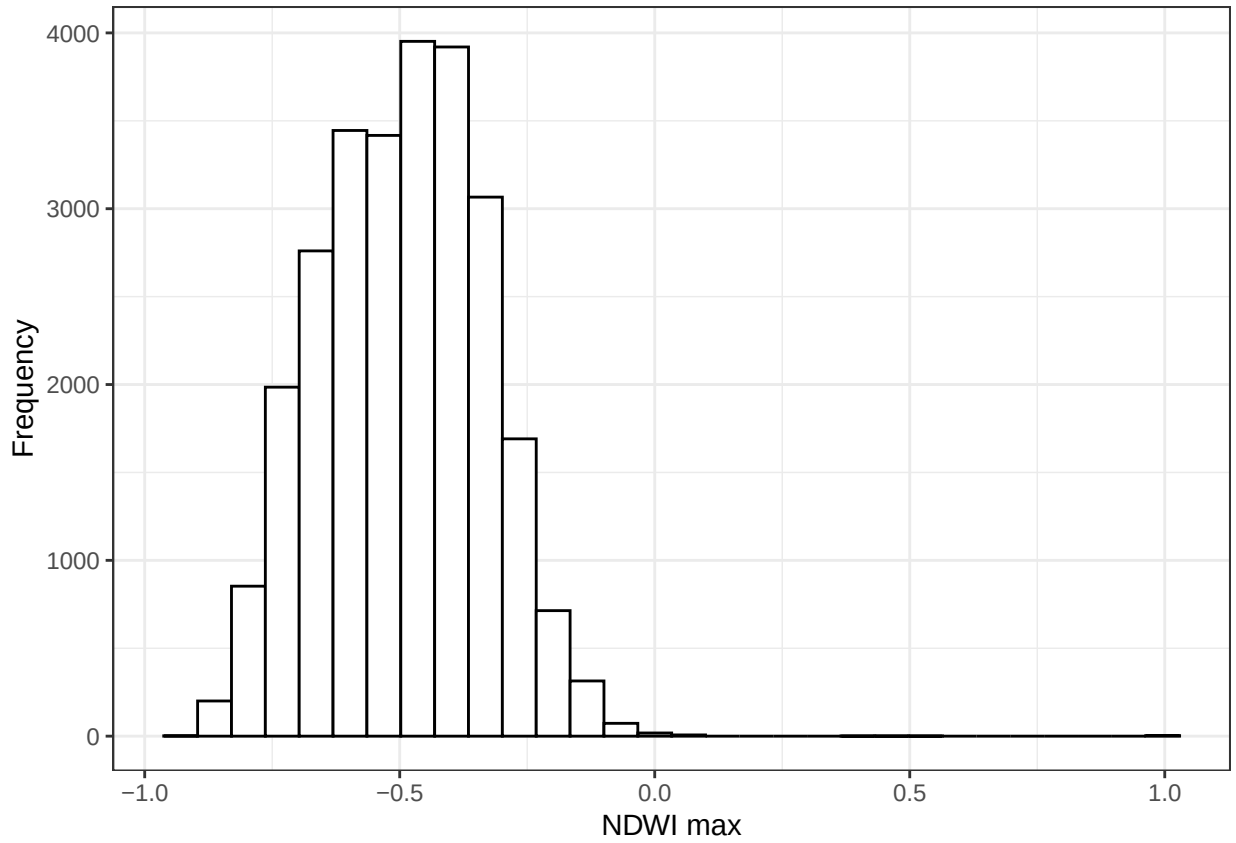

```
hist_NDMI
```

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



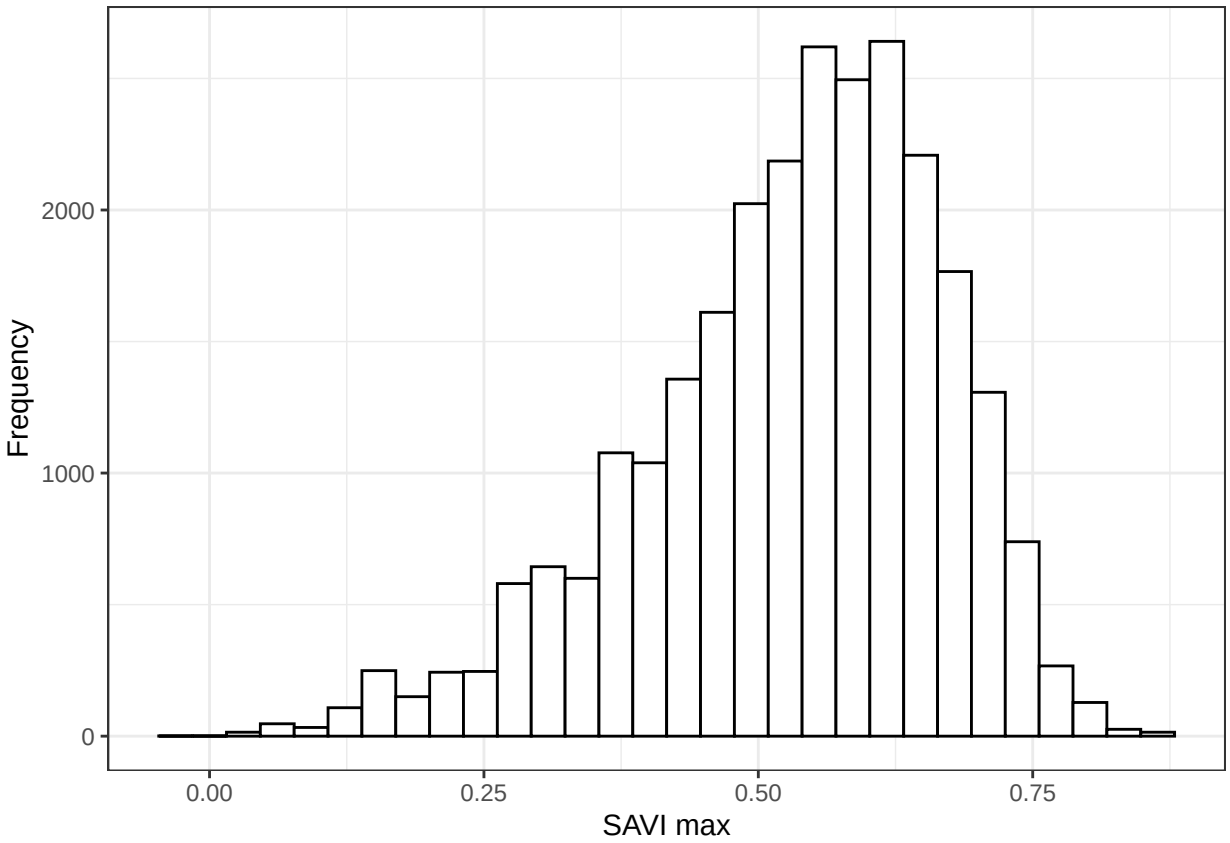

```
hist_NDWI
```

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



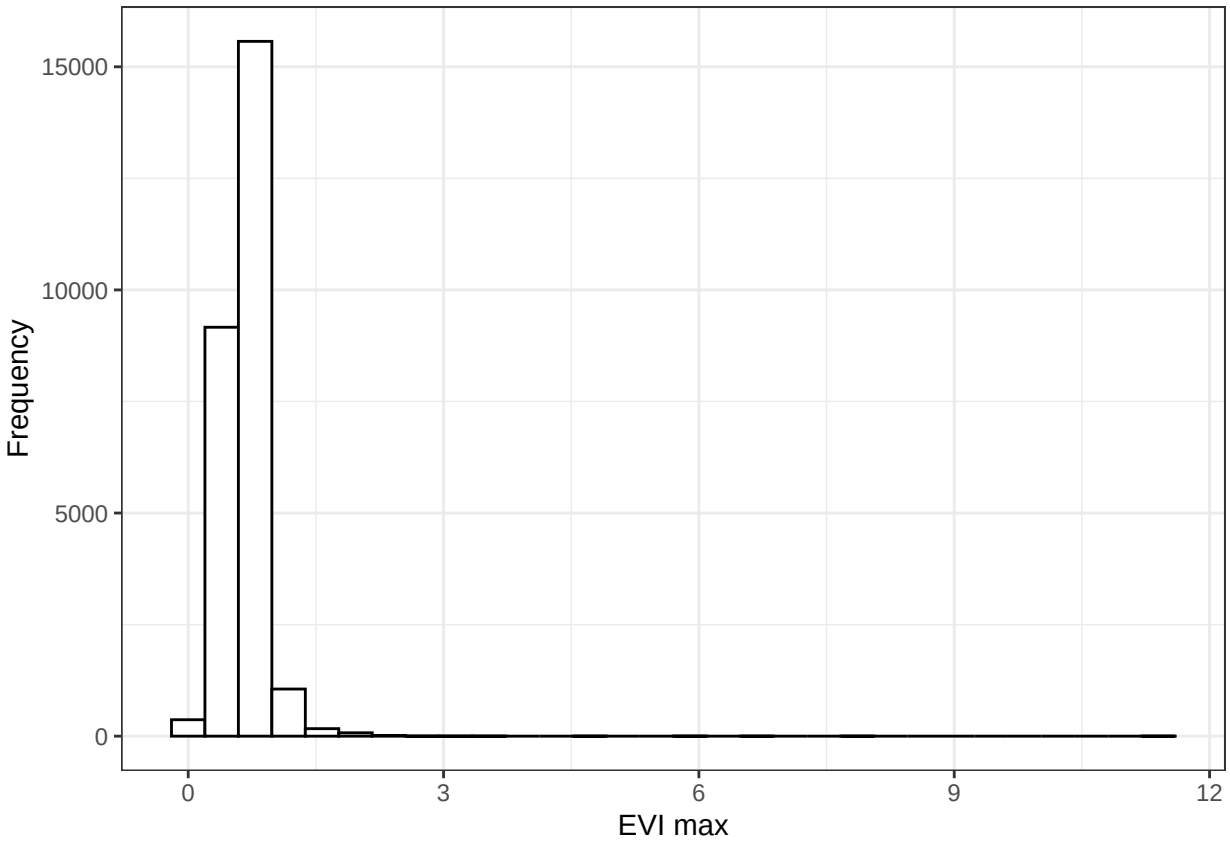

```
hist_SAVI
```

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



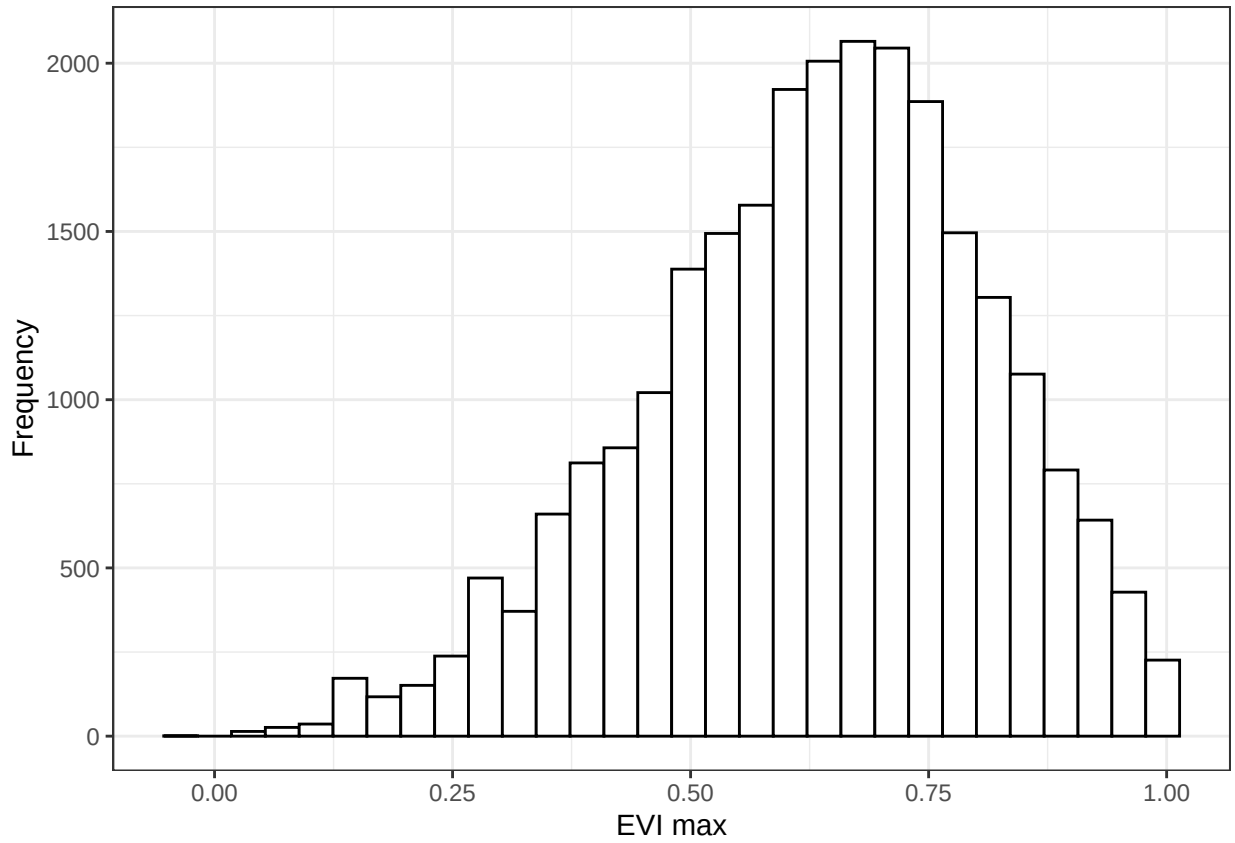

```
hist_EVI # Some values wrong!
```

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




```
hist_EVI_1 <- plot_histogram(db_resurv_RS_short_PLOT %>% filter(EVI_max <= 1),
                             "EVI_max", "EVI max")
hist_EVI_1
```

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




```
nrow(db_resurv_RS_short_PLOT %>%
       filter(EUNISa_1 %in% c("T", "R", "S", "Q")) %>%
       filter(EVI_max > 1))
```

```
## [1] 1130
```

```
db_resurv_RS_short_PLOT %>%
       filter(EUNISa_1 %in% c("T", "R", "S", "Q"))%>%
  filter(EVI_max > 1) %>%
  count(biogeo, unit)
```

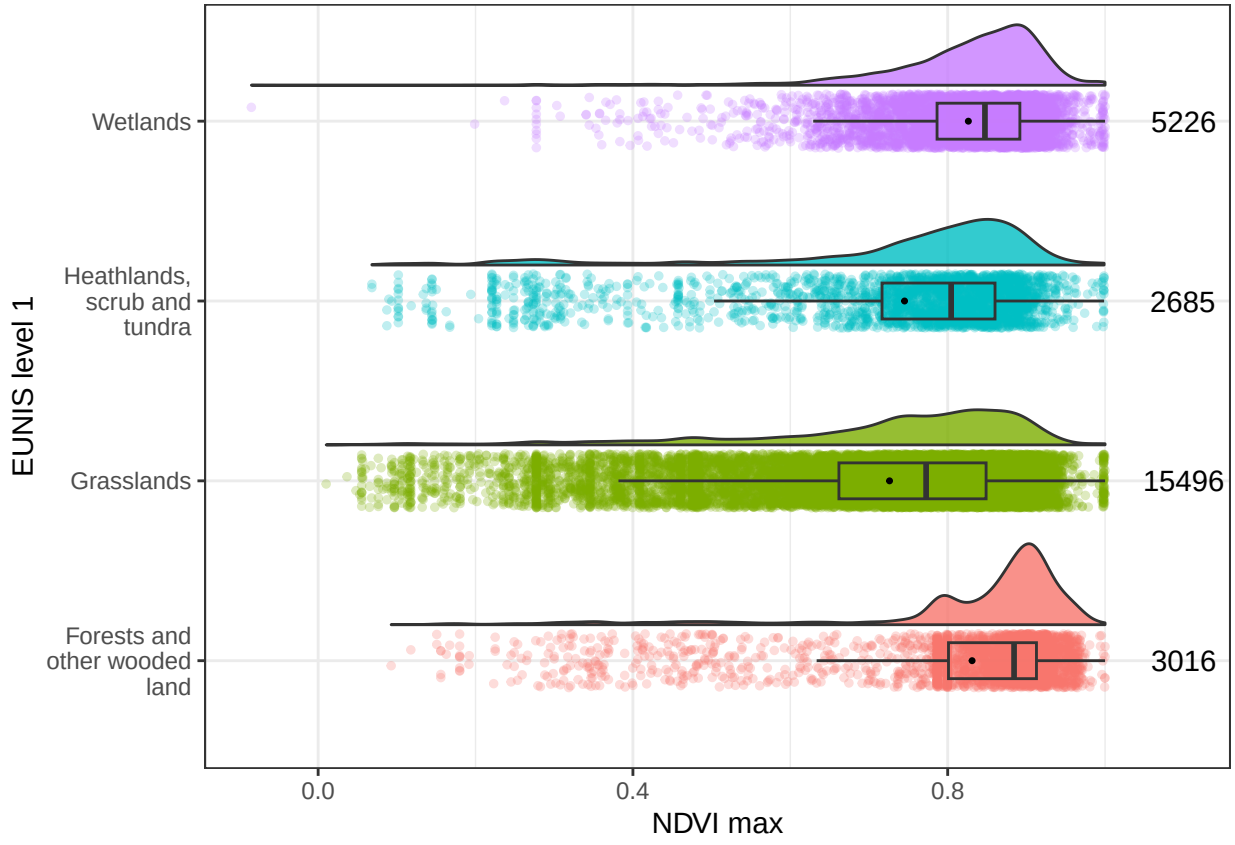
```
## # A tibble: 18 x 3
##    biogeo unit      n
##    <chr>  <chr> <int>
##  1 ALP    ALPS     31
##  2 ALP    EN       18
##  3 ALP    NOR       9
##  4 ALP    PYR      86
##  5 ALP    ROM      16
##  6 ATL    BRI     165
##  7 ATL    FRA      96
##  8 ATL    IBE     117
##  9 BOR    FIN     100
## 10 BOR    NOR      17
## 11 CON    AUS      39
```

```
## 12 CON    BAL       4
## 13 CON    FRA      15
## 14 CON    GER      32
## 15 CON    NOR     293
## 16 CON    POL      58
## 17 MED    POR      10
## 18 PAN    <NA>     24
```

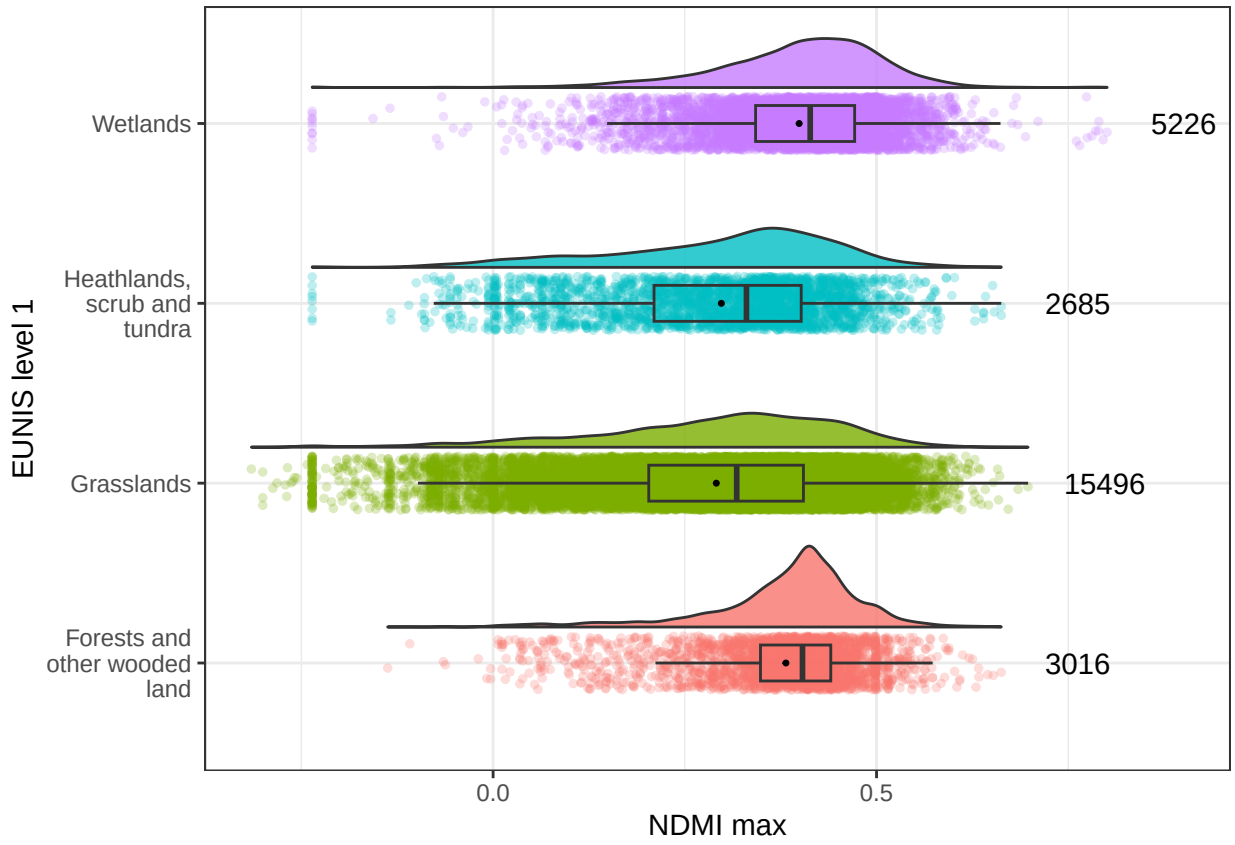
So far, remove observations with EVI_max > 1 from stuff using EVI values.

Distribution plots:

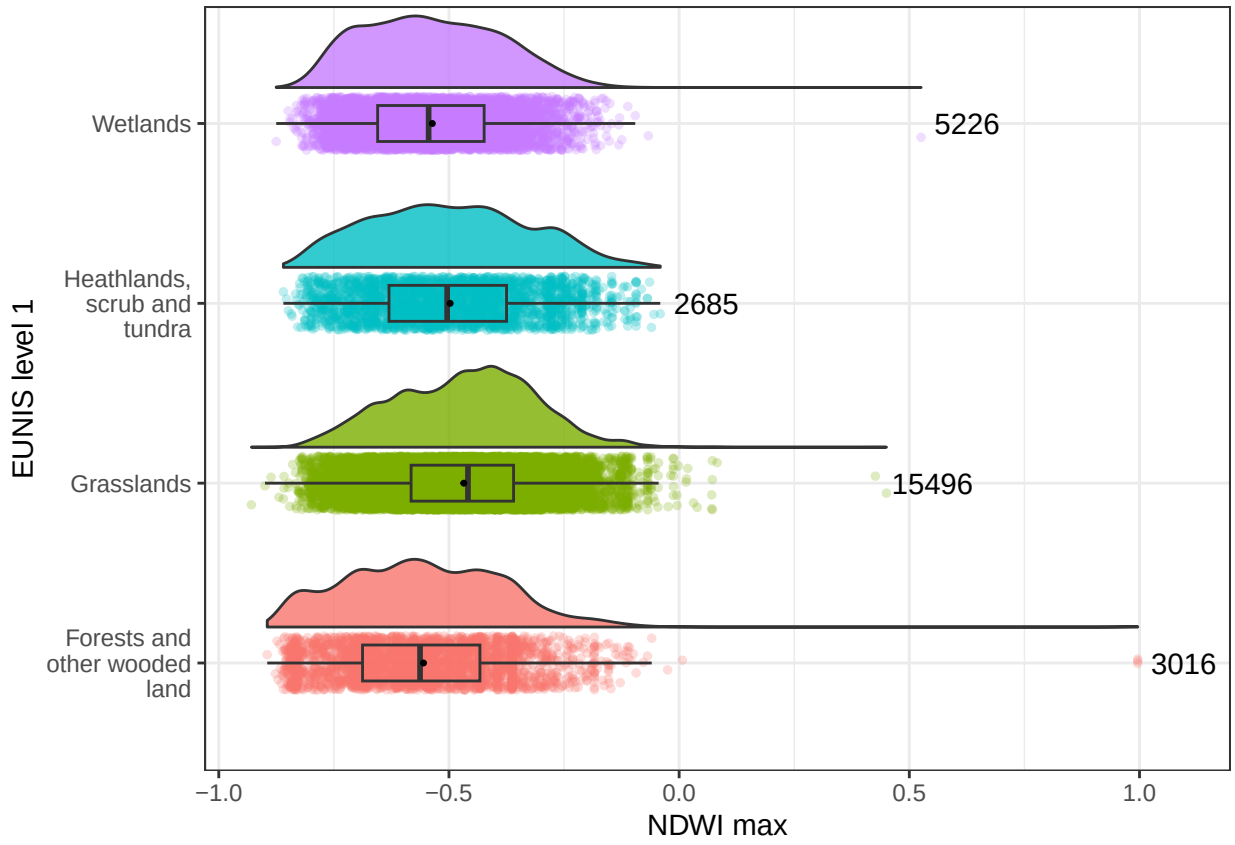
```
plot_distr_NDVI <- distr_plot(db_resurv_RS_short_PLOT, "NDVI_max", "NDVI max")
plot_distr_NDMI <- distr_plot(db_resurv_RS_short_PLOT, "NDMI_max", "NDMI max")
plot_distr_NDWI <- distr_plot(db_resurv_RS_short_PLOT, "NDWI_max", "NDWI max")
plot_distr_SAVI <- distr_plot(db_resurv_RS_short_PLOT, "SAVI_max", "SAVI max")
plot_distr_EVI <- distr_plot(db_resurv_RS_short_PLOT %>% filter(EVI_max <= 1),
                             "EVI_max", "EVI max")
plot_distr_NDVI
```



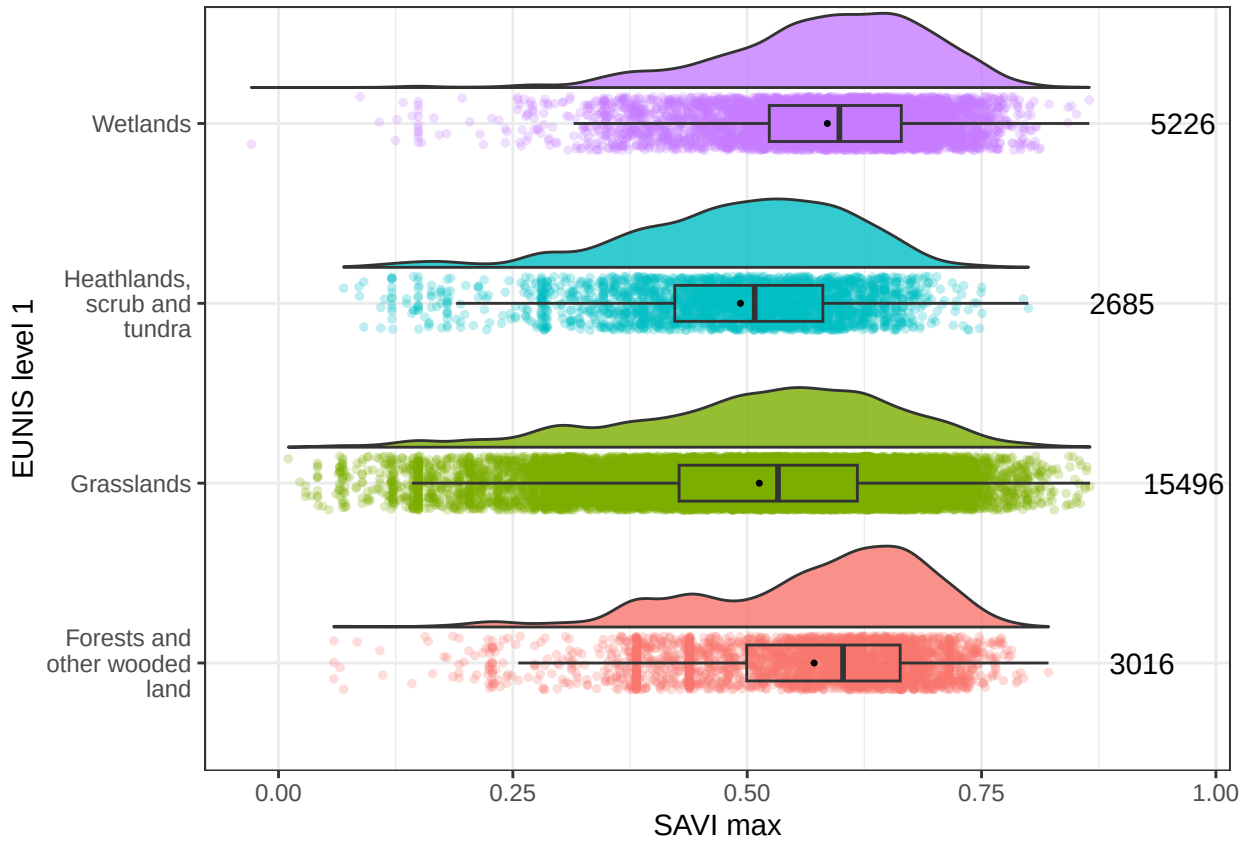

```
plot_distr_NDMI
```



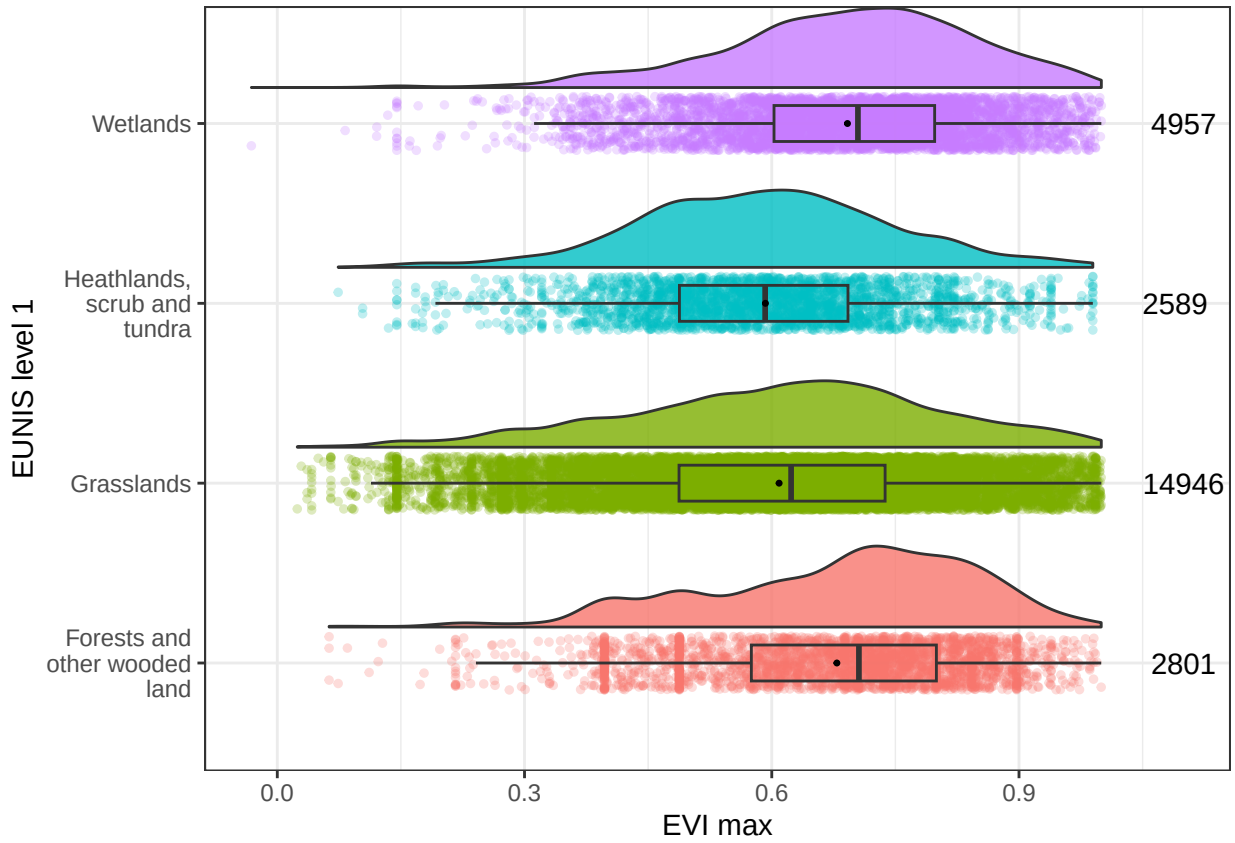

```
plot_distr_NDWI
```




```
plot_distr_SAVI
```

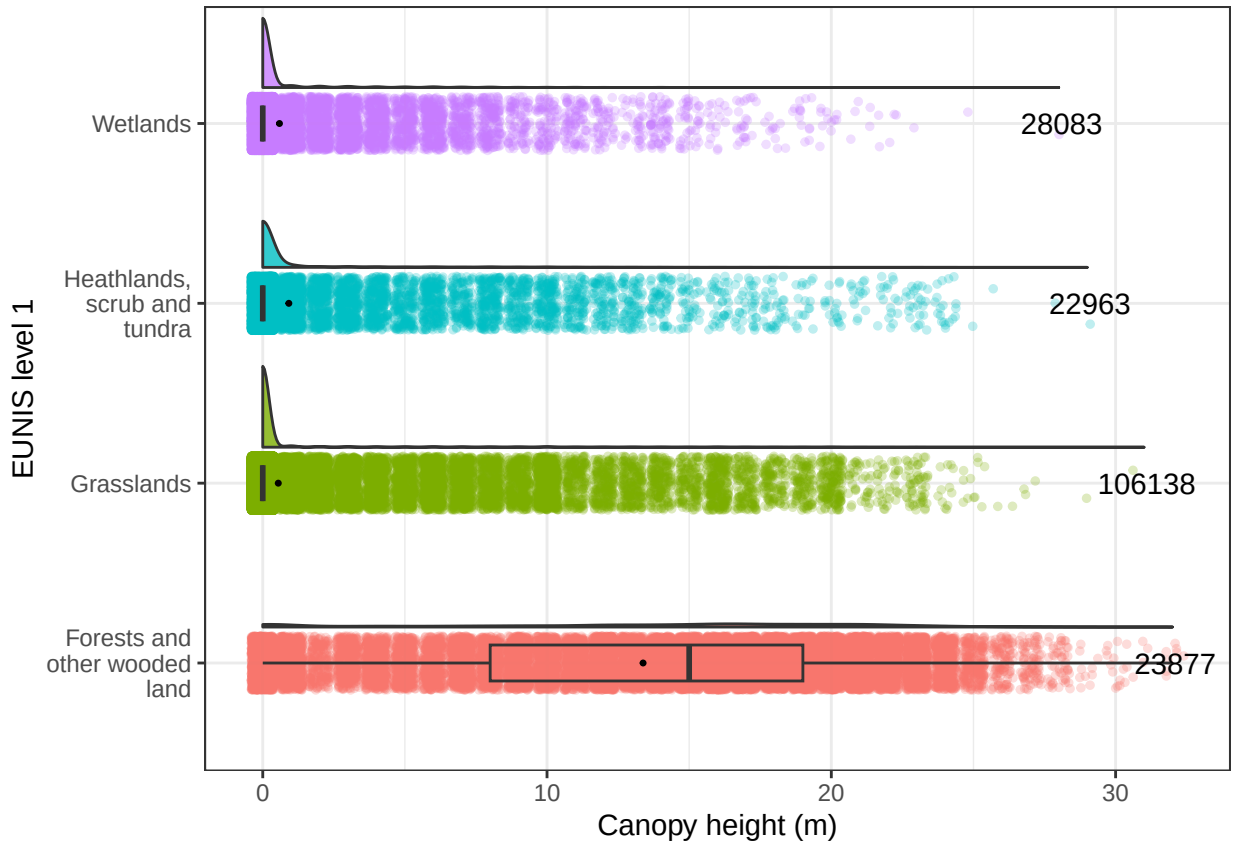



```
plot_distr_EVI
```



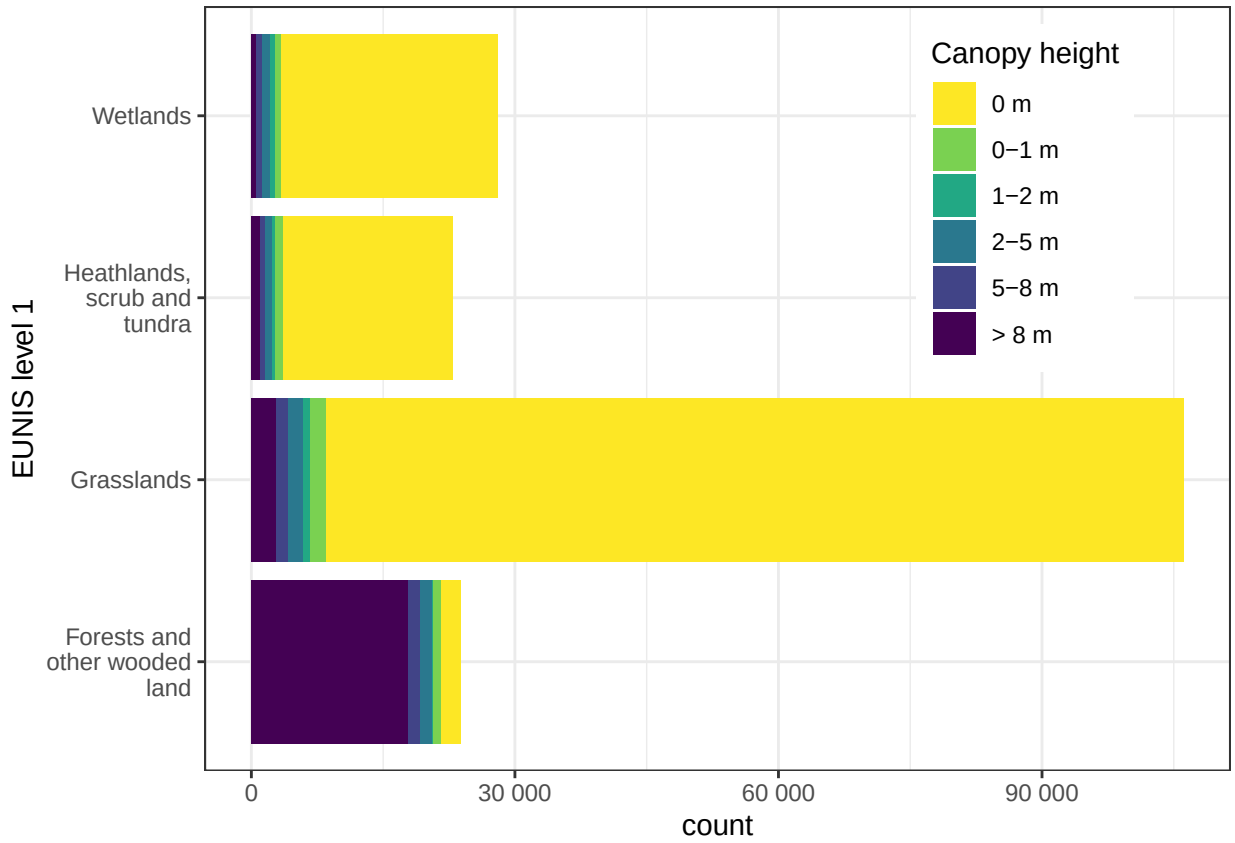
CH

```
plot_distr_CH <- distr_plot(db_resurv_RS_short_PLOT, "canopy_height",
                            "Canopy height (m)")
plot_distr_CH
```



Show habitats with CH categories

```
ggplot(db_resurv_RS_short_PLOT %>%
         # Keep only forests, grasslands, shrublands and wetlands
         filter(EUNISa_1 %in% c("T", "R", "S", "Q")) %>%
         mutate(CH_cat =
                  factor(
                    case_when(canopy_height == 0 ~ "0 m",
                              canopy_height > 0 & canopy_height <= 1 ~ "0-1 m",
                              canopy_height > 1 & canopy_height <=2 ~ "1-2 m",
                              canopy_height > 2 & canopy_height <=5 ~ "2-5 m",
                              canopy_height > 5 & canopy_height <=8 ~ "5-8 m",
                              canopy_height > 8 ~ "> 8 m",
                              is.na(canopy_height) ~ NA_character_),
                    levels = c(
                      "0 m", "0-1 m", "1-2 m", "2-5 m", "5-8 m", "> 8 m"))),
       aes(x = EUNISa_1_descr, fill = CH_cat)) +
  geom_bar() + theme_bw() + coord_flip() +
  scale_y_continuous(labels = label_number()) +
  scale_fill_viridis_d(direction = -1) +
  labs(x = "EUNIS level 1", fill = "Canopy height") +
  scale_x_discrete(labels = function(x) str_wrap(x, width = 15)) +
  theme(legend.position = c(0.8, 0.75),
        legend.direction = "vertical")
```



Stats per habitat type

```
db_resurv_RS_short_PLOT %>%
  # Keep only forests, grasslands, shrublands and wetlands
  filter(EUNISa_1 %in% c("T", "R", "S", "Q")) %>%
  group_by(EUNISa_1_descr) %>%
  summarise(across(canopy_height, list(
    mean = mean,
    median = median,
    sd = sd,
    min = min,
    max = max
    ), na.rm = TRUE))
```

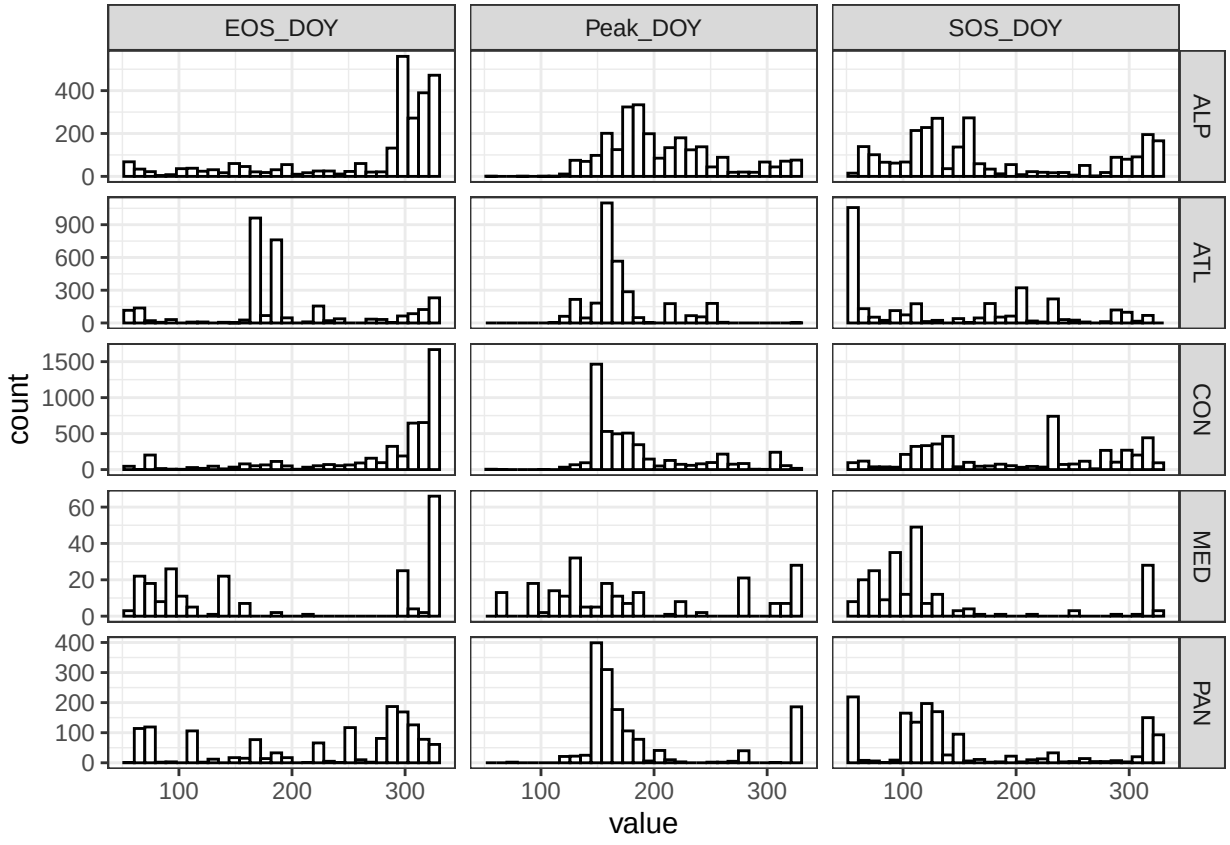
```
## # A tibble: 4 x 6
##   EUNISa_1_descr        canopy_height_mean canopy_height_median canopy_height_sd
##   <chr>                              <dbl>                <dbl>            <dbl>
## 1 Forests and other wo~              13.4                    15             7.38
## 2 Grasslands                          0.546                   0             2.46
## 3 Heathlands, scrub an~               0.918                   0             2.97
## 4 Wetlands                            0.588                   0             2.15
## # i 2 more variables: canopy_height_min <dbl>, canopy_height_max <dbl>
```

Phenology

Histograms phenology measures

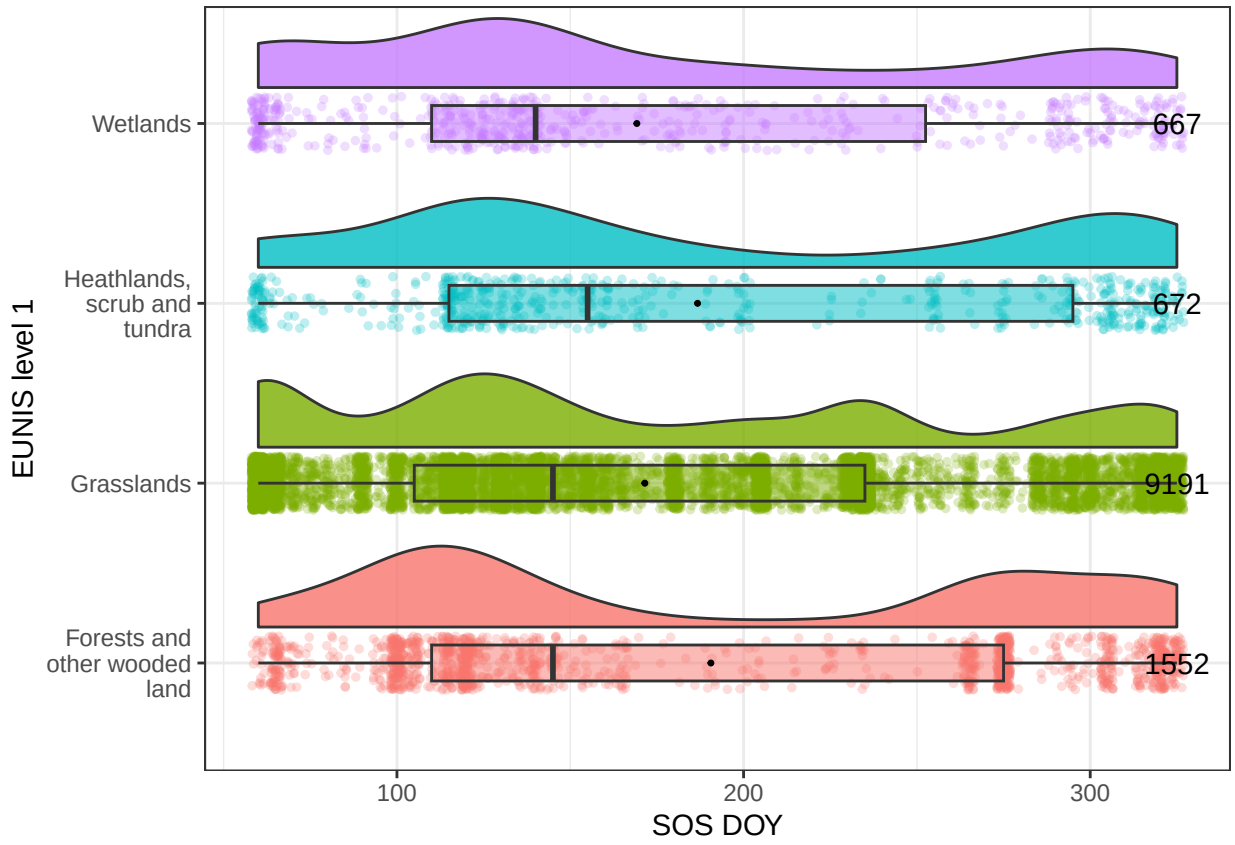
```
ggplot(data = db_resurv_RS_short_PLOT %>%
         # Keep only forests, grasslands, shrublands and wetlands
         filter(EUNISa_1 %in% c("T", "R", "S", "Q") & S2_phen_data == T) %>%
         pivot_longer(cols = c(SOS_DOY, Peak_DOY, EOS_DOY), names_to = "name",
                      values_to = "value"),
       aes(x = value)) +
  geom_histogram(fill = "white", color = "black") +
  facet_grid(biogeo ~ name, scales = "free_y") +
  theme_bw()
```

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

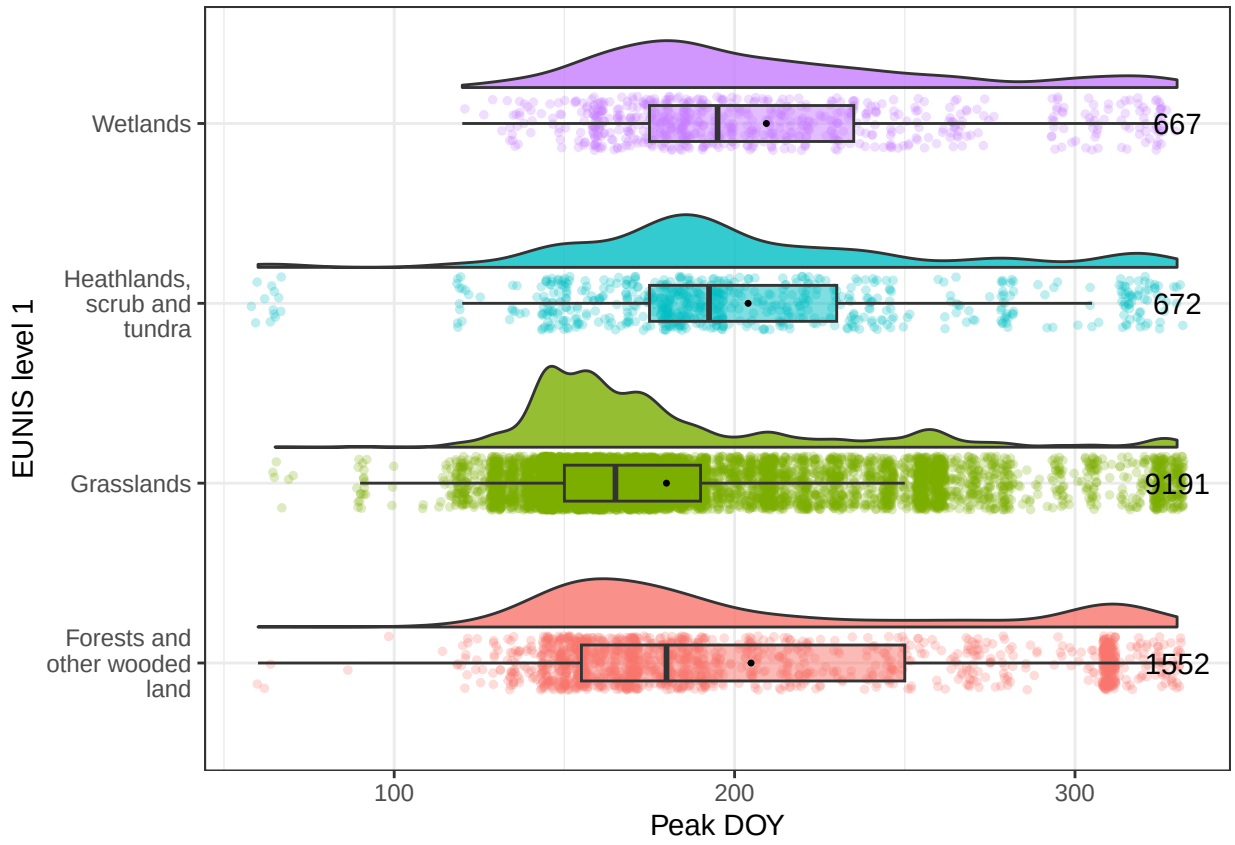


Distributions

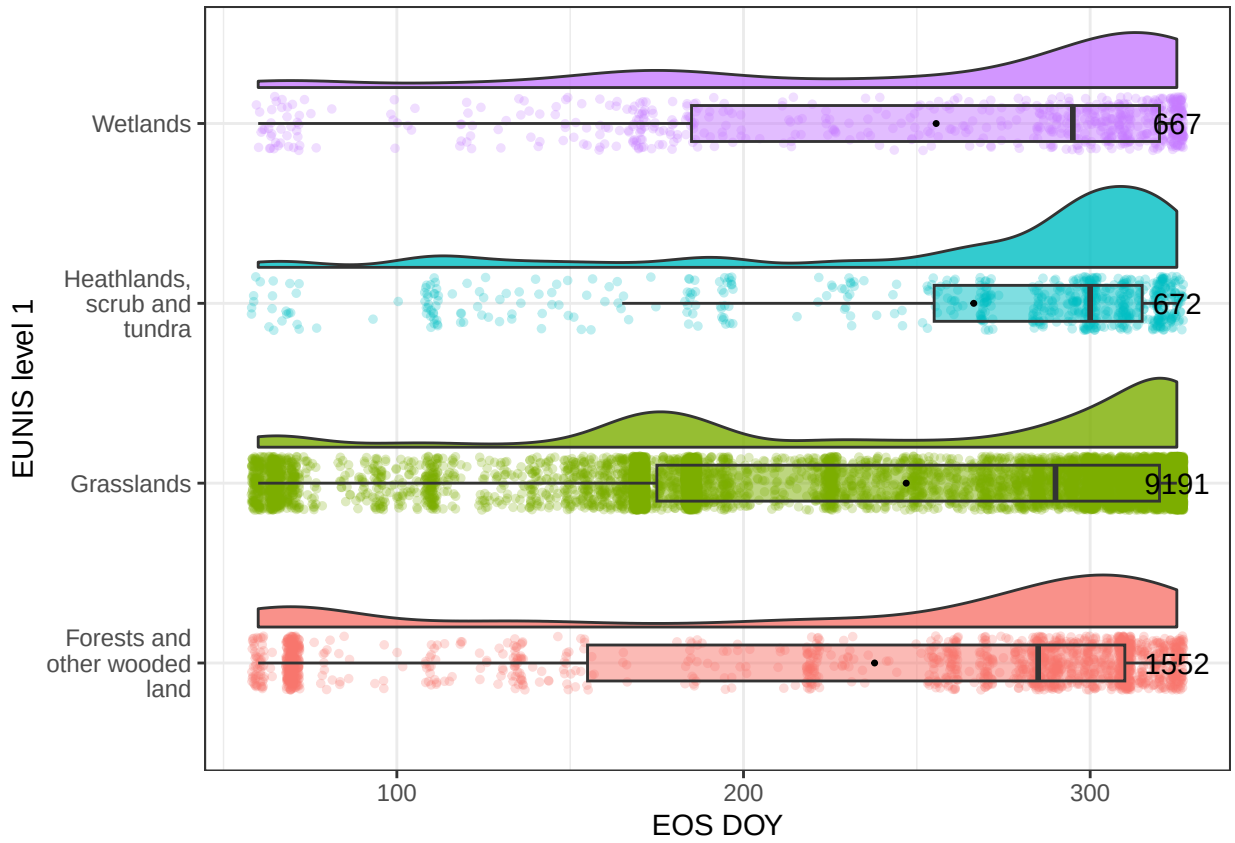
```
plot_distr_SOS_DOY <- distr_plot(db_resurv_RS_short_PLOT, "SOS_DOY", "SOS DOY")
plot_distr_Peak_DOY <- distr_plot(db_resurv_RS_short_PLOT, "Peak_DOY", "Peak DOY")
plot_distr_EOS_DOY <- distr_plot(db_resurv_RS_short_PLOT, "EOS_DOY", "EOS DOY")
plot_distr_SOS_DOY
```




```
plot_distr_Peak_DOY
```




```
plot_distr_EOS_DOY
```



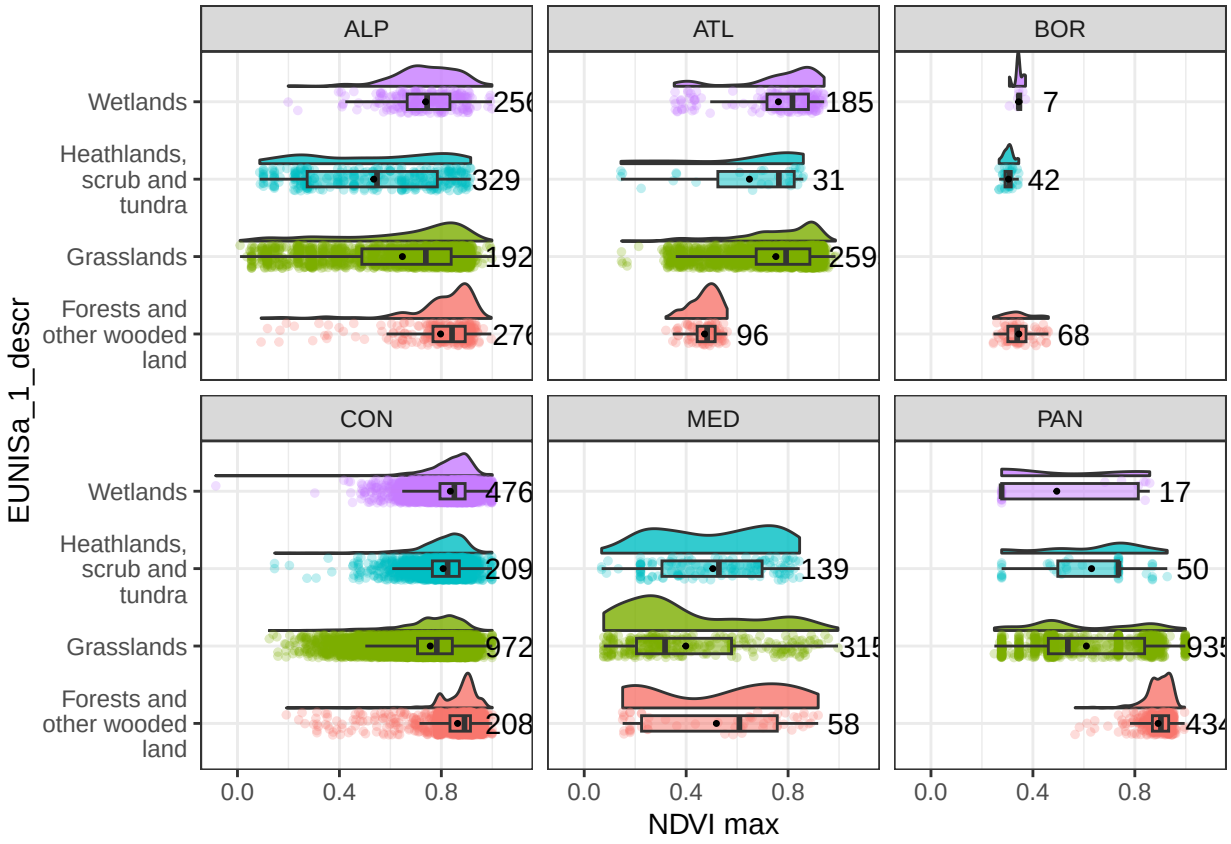
Distributions per bioregion

```
# Define a function to create plots with violin + boxplot + points
distr_plot_biogeo <- function(data, y_var, y_label) {
  ggplot(data = data %>%
           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)
}
```

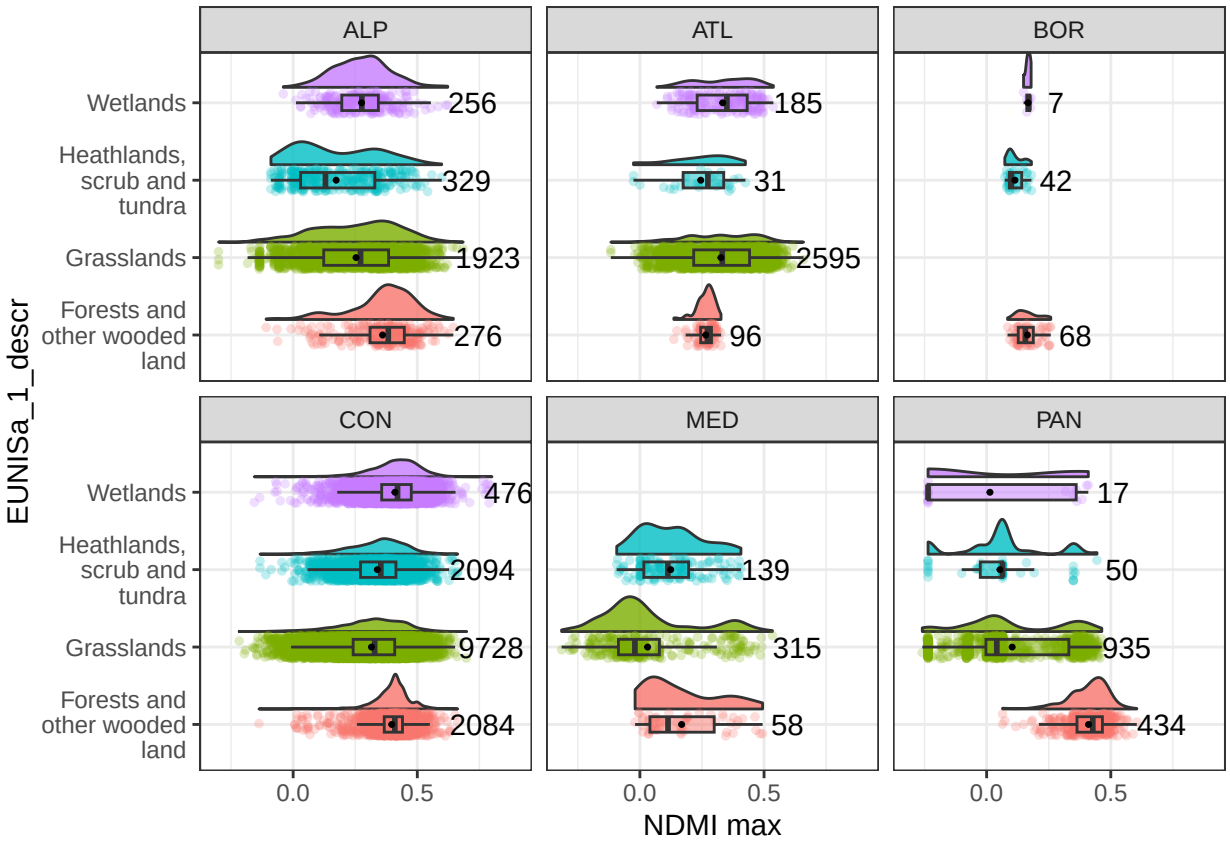
Max. NDVI, NDMI, NDWI, SAVI and EVI

Distribution plots:

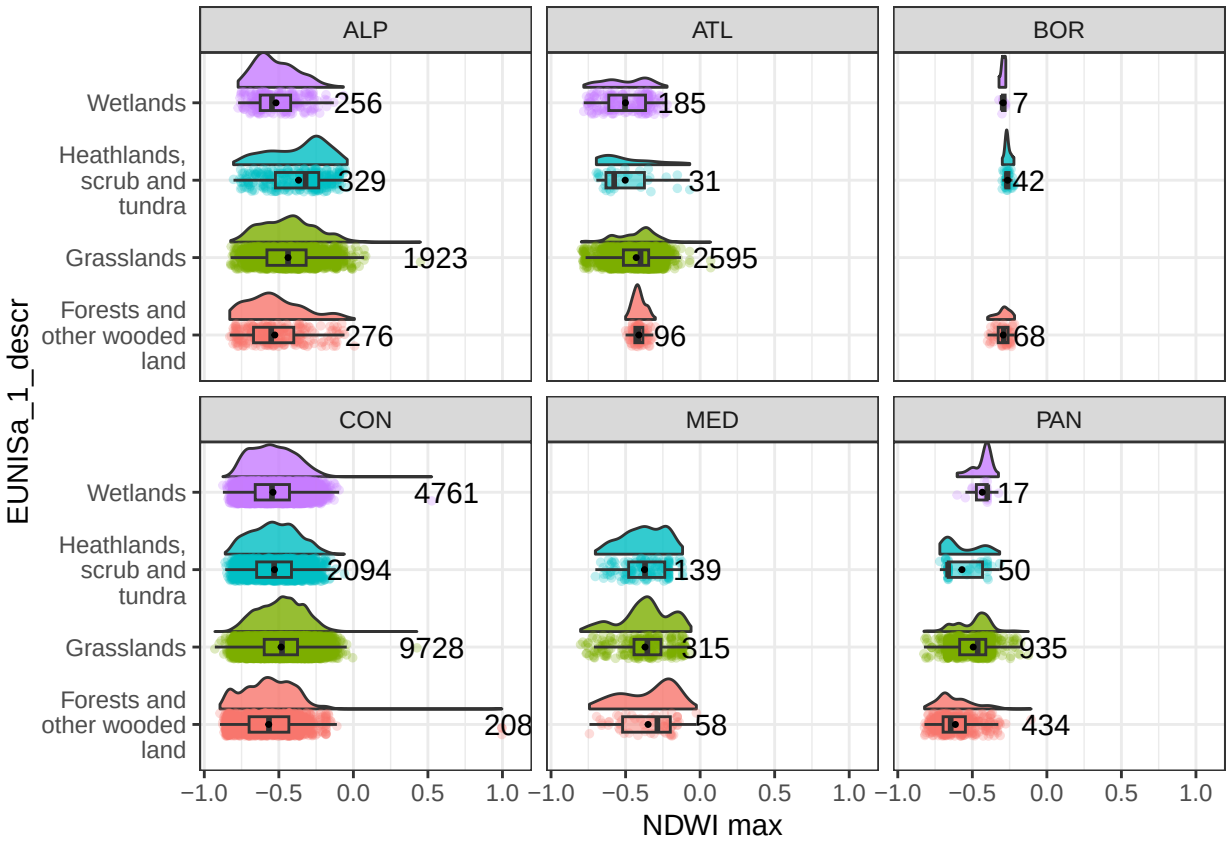
```
plot_distr_NDVI_biogeo <- distr_plot_biogeo(db_resurv_RS_short_PLOT %>%
                                              filter(!is.na(biogeo)),
                                            "NDVI_max", "NDVI max")
plot_distr_NDMI_biogeo <- distr_plot_biogeo(db_resurv_RS_short_PLOT %>%
                                              filter(!is.na(biogeo)),
                                            "NDMI_max", "NDMI max")
plot_distr_NDWI_biogeo <- distr_plot_biogeo(db_resurv_RS_short_PLOT %>%
                                              filter(!is.na(biogeo)),
                                            "NDWI_max", "NDWI max")
plot_distr_SAVI_biogeo <- distr_plot_biogeo(db_resurv_RS_short_PLOT %>%
                                              filter(!is.na(biogeo)),
                                            "SAVI_max", "SAVI max")
plot_distr_EVI_biogeo <- distr_plot_biogeo(db_resurv_RS_short_PLOT %>%
                                             filter(!is.na(biogeo)) %>%
                                             filter(EVI_max <= 1),
                                           "EVI_max", "EVI max")
plot_distr_NDVI_biogeo
```



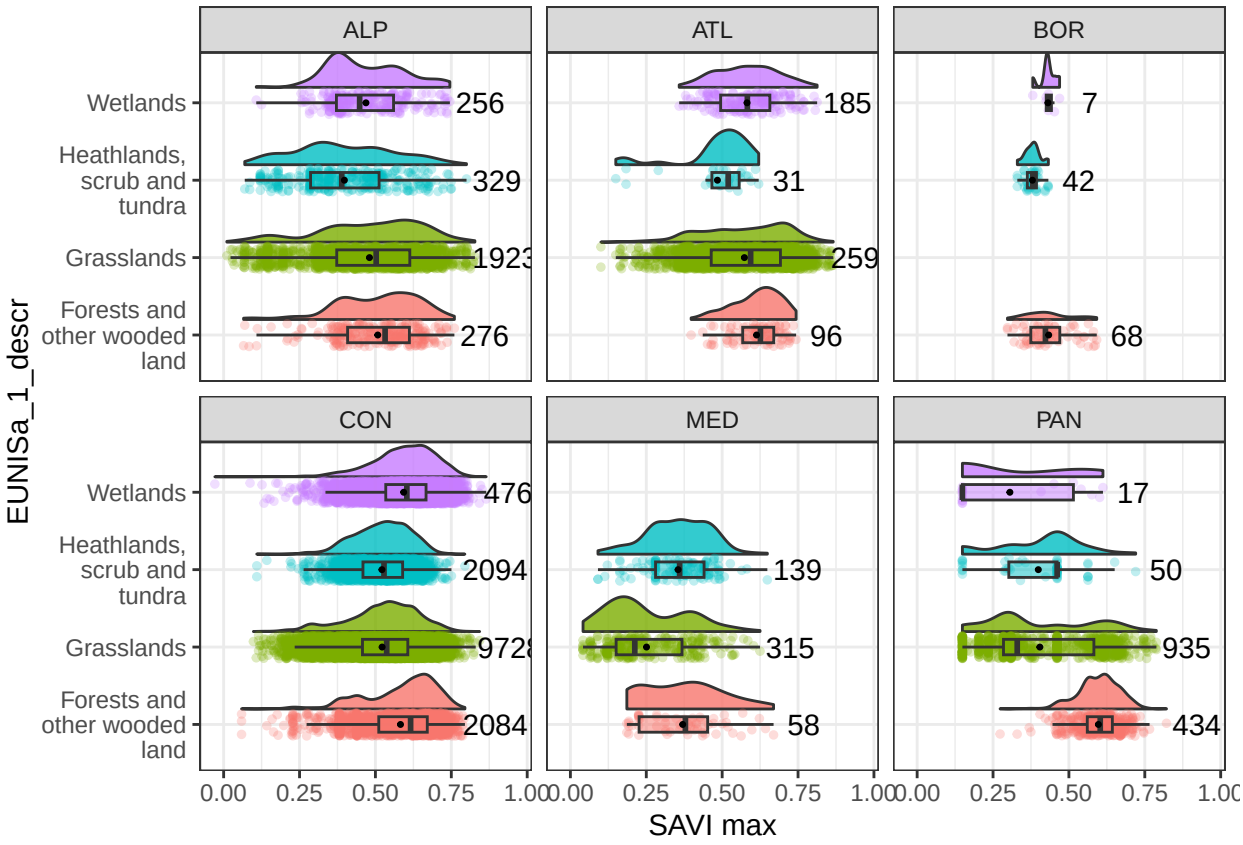

```
plot_distr_NDMI_biogeo
```



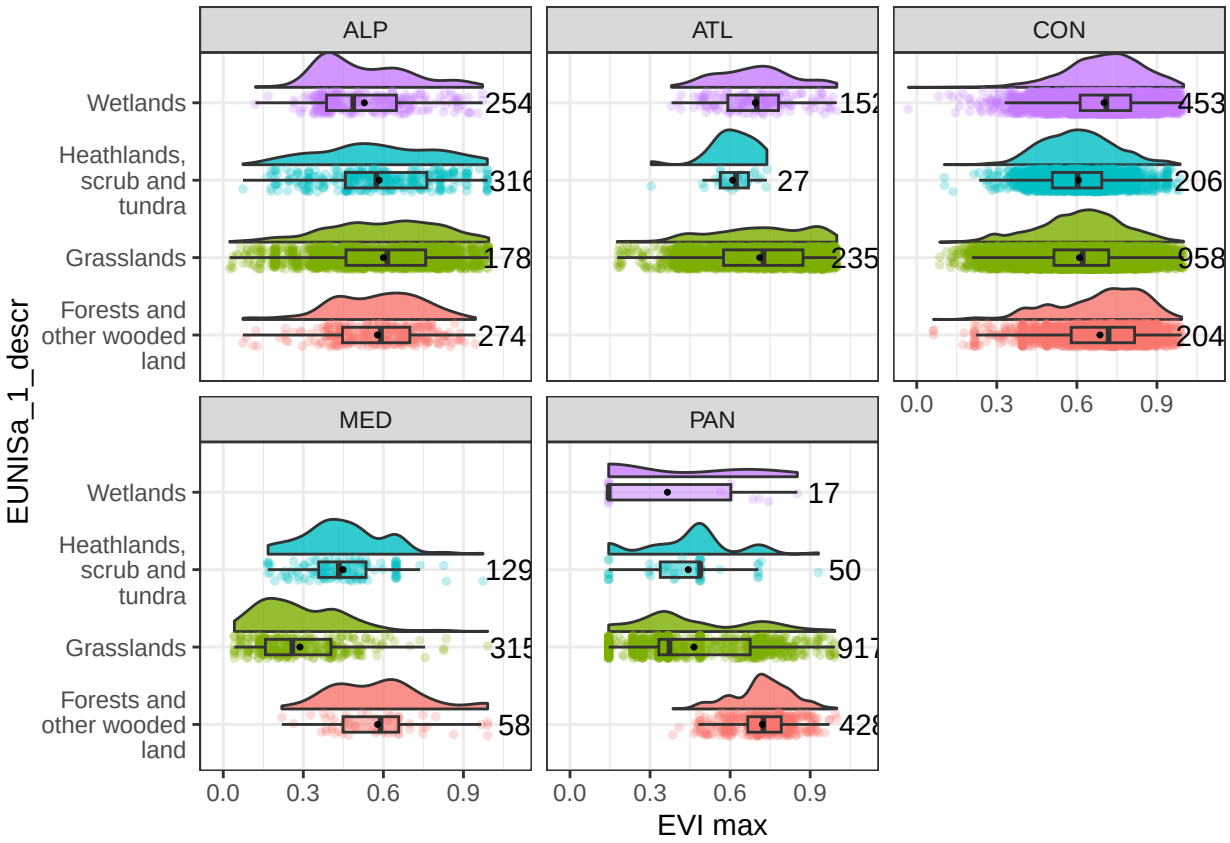

```
plot_distr_NDWI_biogeo
```




```
plot_distr_SAVI_biogeo
```

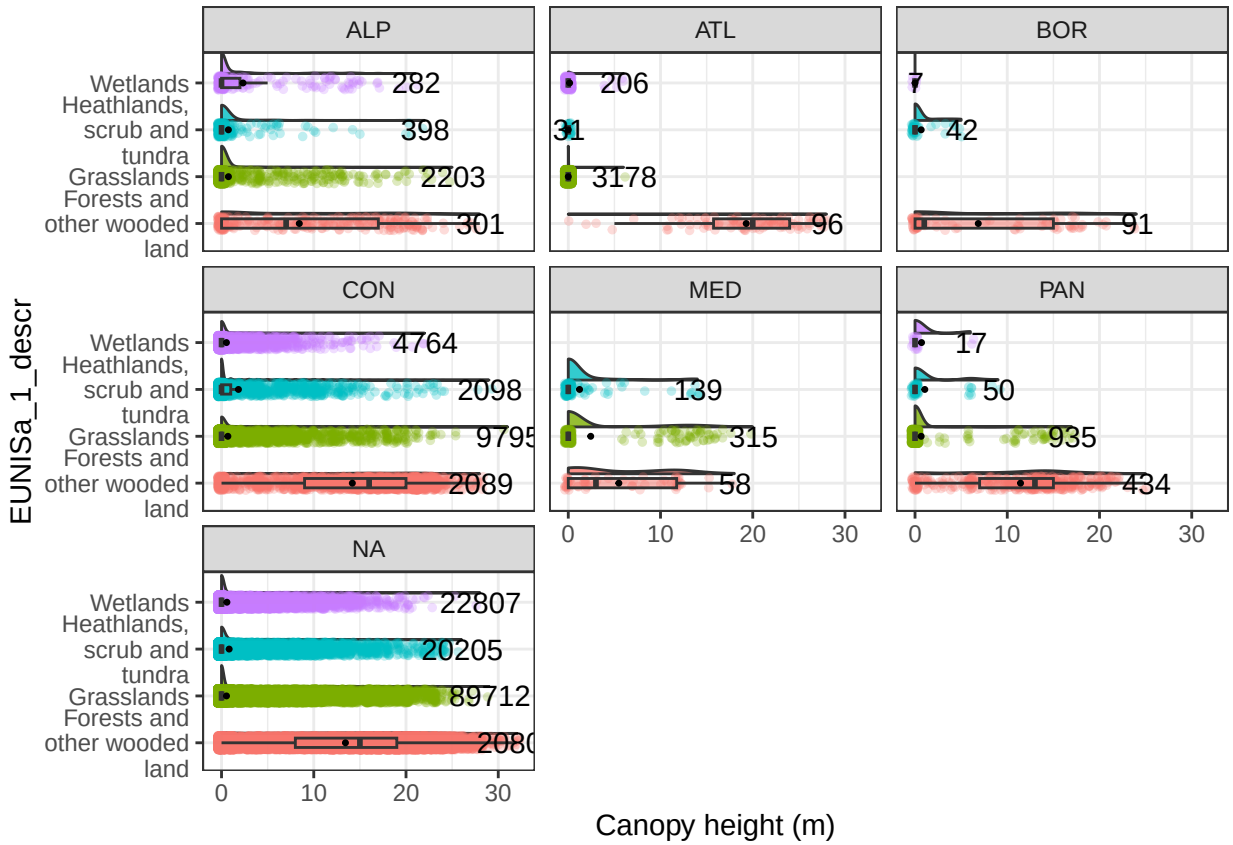



```
plot_distr_EVI_biogeo
```



CH

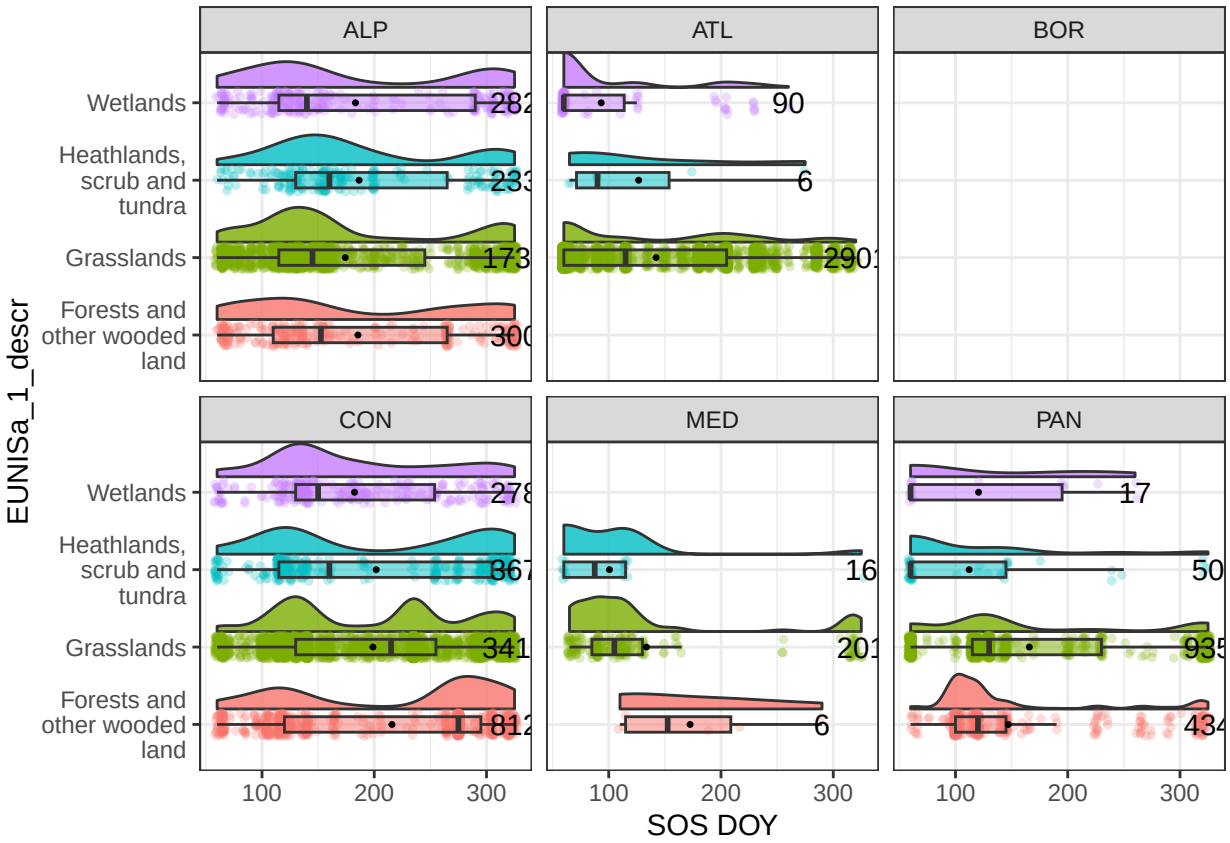
```
plot_distr_CH_biogeo <- distr_plot_biogeo(db_resurv_RS_short_PLOT,
                                          "canopy_height", "Canopy height (m)")
plot_distr_CH_biogeo
```



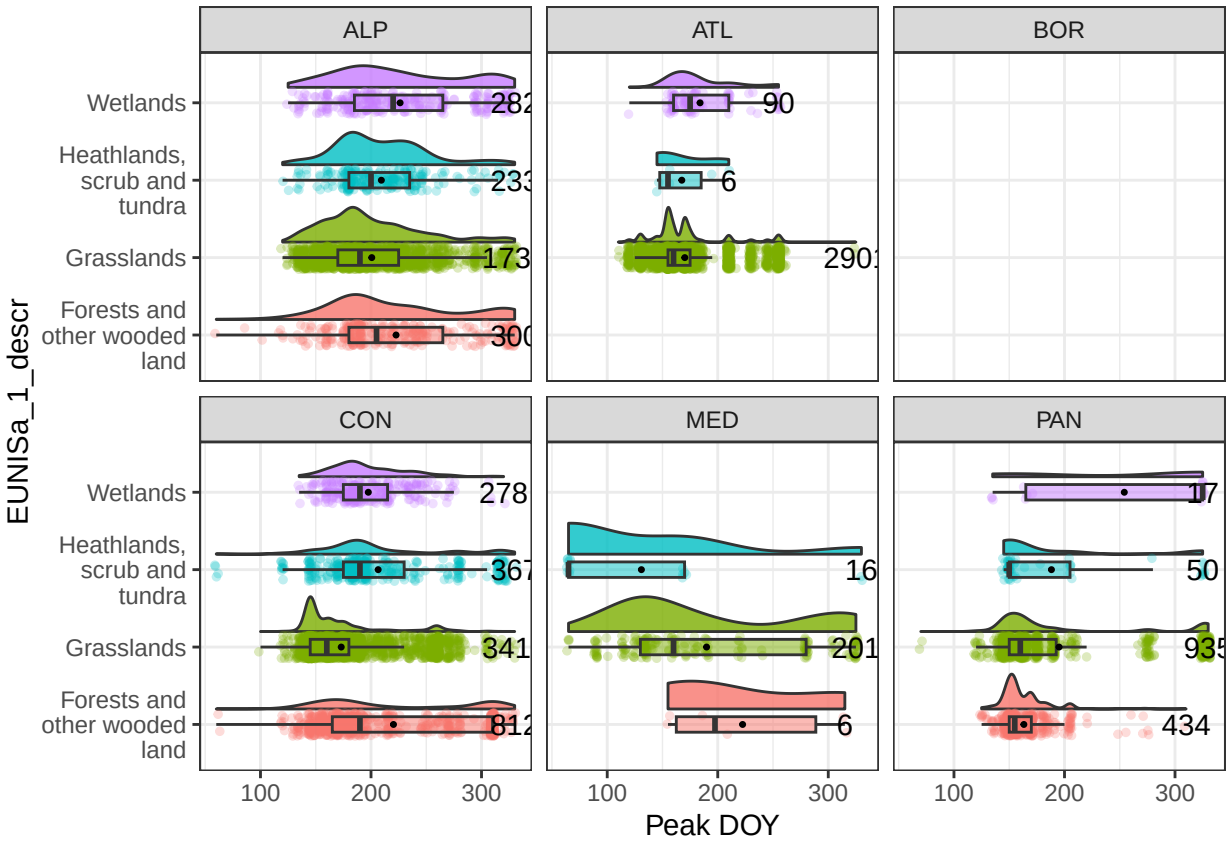
In this plot, those with biogeo = NA are those that do not have S2 or Landsat data (and thus biogeo has not been assigned), but have CH data. We should later assign a biogeo based on location.

Phenology

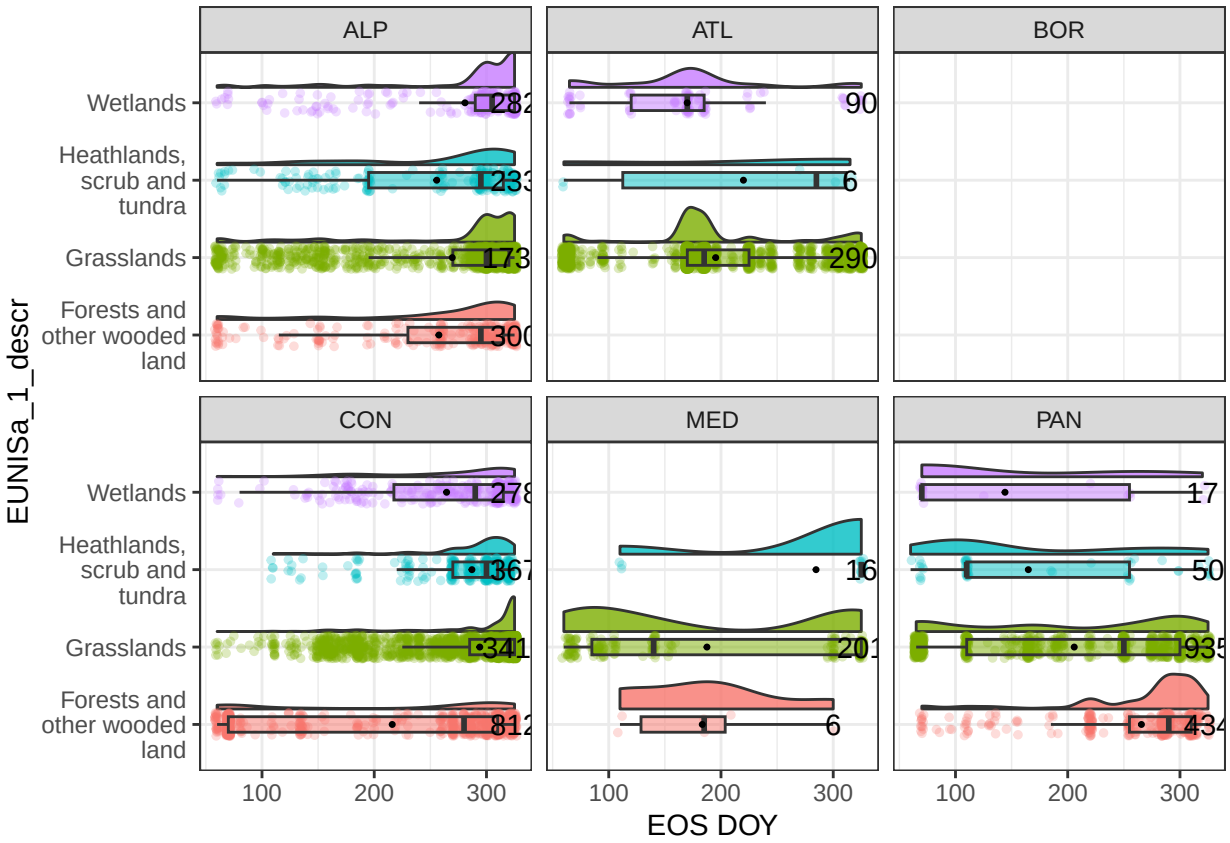
```
plot_distr_SOS_DOY_biogeo <- distr_plot_biogeo(db_resurv_RS_short_PLOT %>%
                                                 filter(!is.na(biogeo)),
                                                  "SOS_DOY", "SOS DOY")
plot_distr_Peak_DOY_biogeo <- distr_plot_biogeo(db_resurv_RS_short_PLOT %>%
                                                 filter(!is.na(biogeo)),
                                                   "Peak_DOY", "Peak DOY")
plot_distr_EOS_DOY_biogeo <- distr_plot_biogeo(db_resurv_RS_short_PLOT %>%
                                                 filter(!is.na(biogeo)),
                                                  "EOS_DOY", "EOS DOY")
plot_distr_SOS_DOY_biogeo
```




```
plot_distr_Peak_DOY_biogeo
```




```
plot_distr_EOS_DOY_biogeo
```



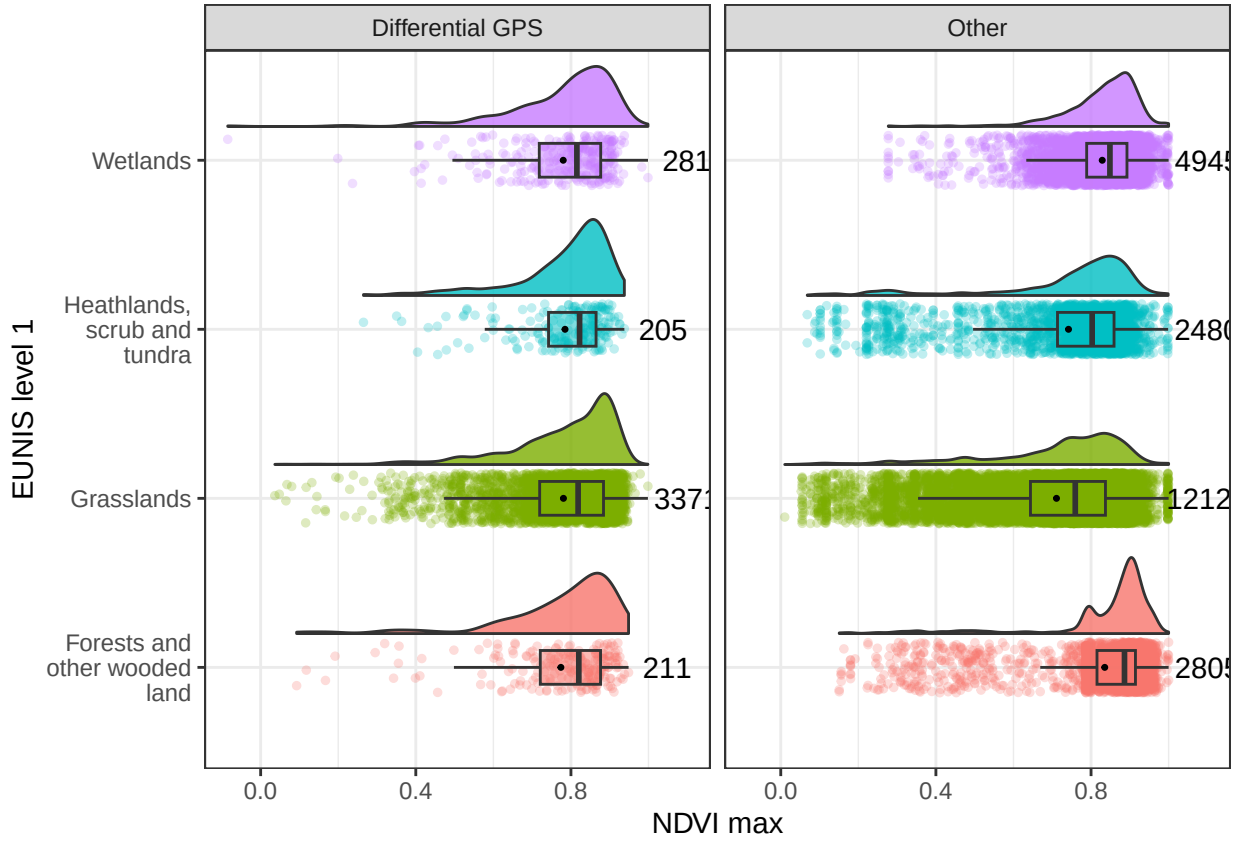
Why BOR missing?

Comparison differential GPS vs. other points

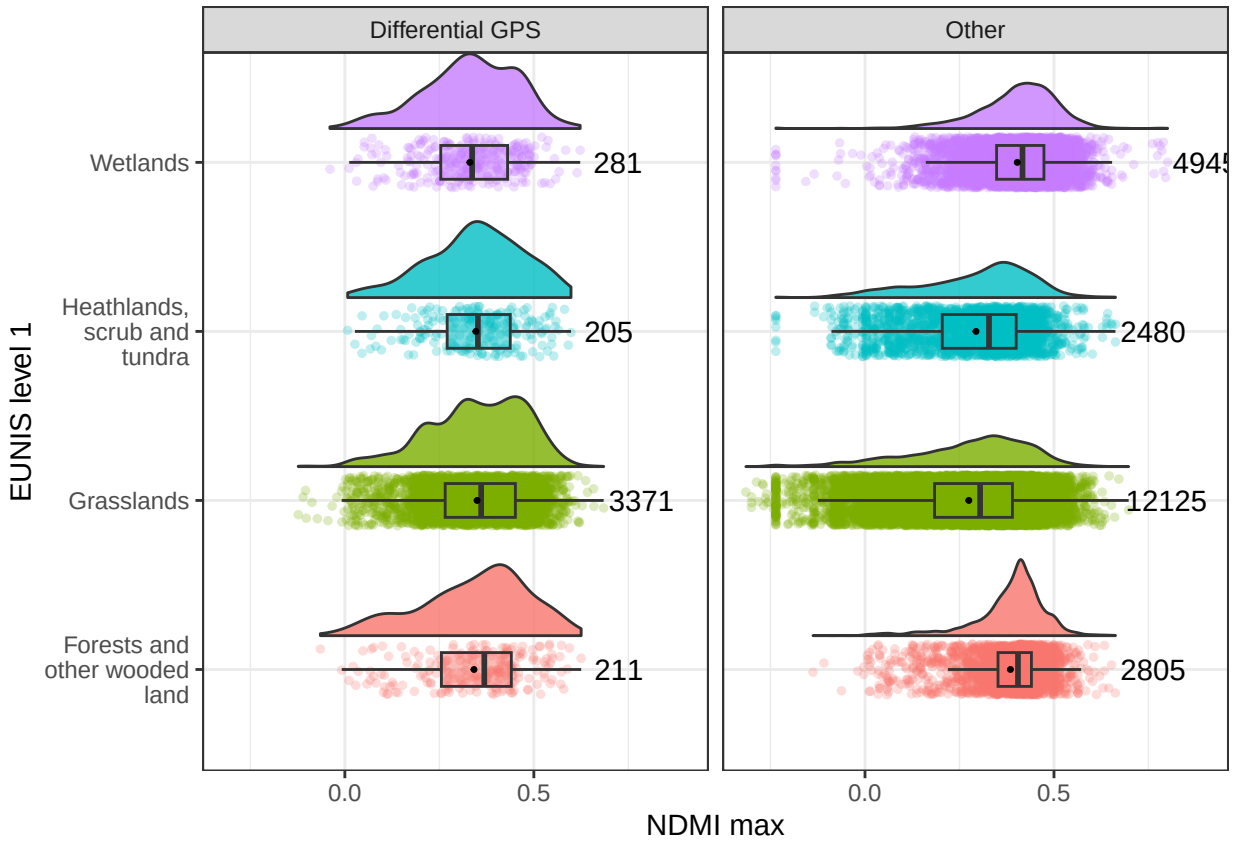
```
# Define function
distr_plot_GPS <- function(data, y_var, y_label) {
  ggplot(data = data %>%
           filter(EUNISa_1 %in% c("T", "R", "S", "Q")) %>%
           mutate(GPS =
                    ifelse(
                      is.na(`Location method`) |
                        `Location method` != "Location with differential GPS",
                      "Other",
                      "Differential GPS")),
         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() + facet_wrap(~ GPS)
}
```

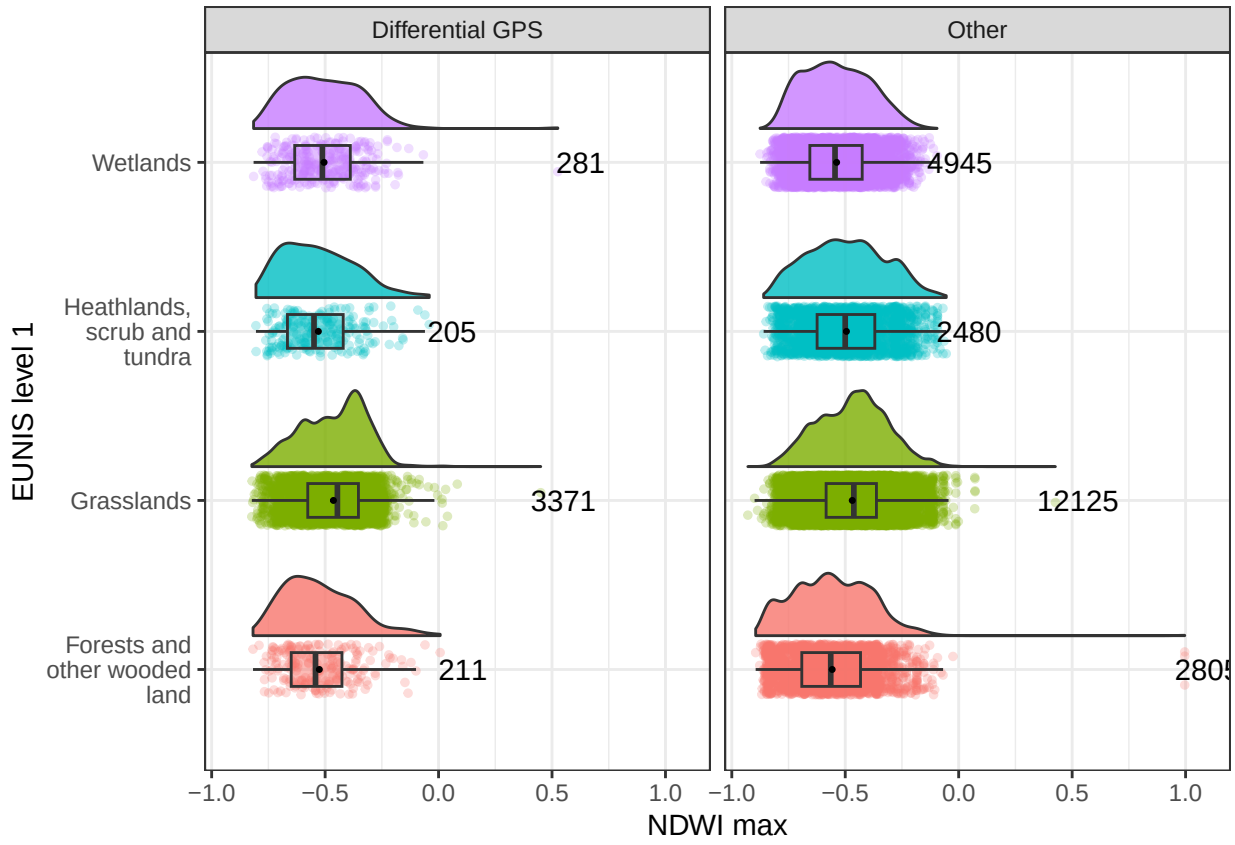
```
plot_distr_NDVI_GPS <- distr_plot_GPS(db_resurv_RS_short_PLOT,
                                      "NDVI_max", "NDVI max")
plot_distr_NDMI_GPS <- distr_plot_GPS(db_resurv_RS_short_PLOT,
                                      "NDMI_max", "NDMI max")
plot_distr_NDWI_GPS <- distr_plot_GPS(db_resurv_RS_short_PLOT,
                                      "NDWI_max", "NDWI max")
plot_distr_SAVI_GPS <- distr_plot_GPS(db_resurv_RS_short_PLOT,
                                      "SAVI_max", "SAVI max")
plot_distr_EVI_GPS <- distr_plot_GPS(db_resurv_RS_short_PLOT %>%
                                      filter(EVI_max <= 1),
                                     "EVI_max", "EVI max")
plot_distr_NDVI_GPS
```



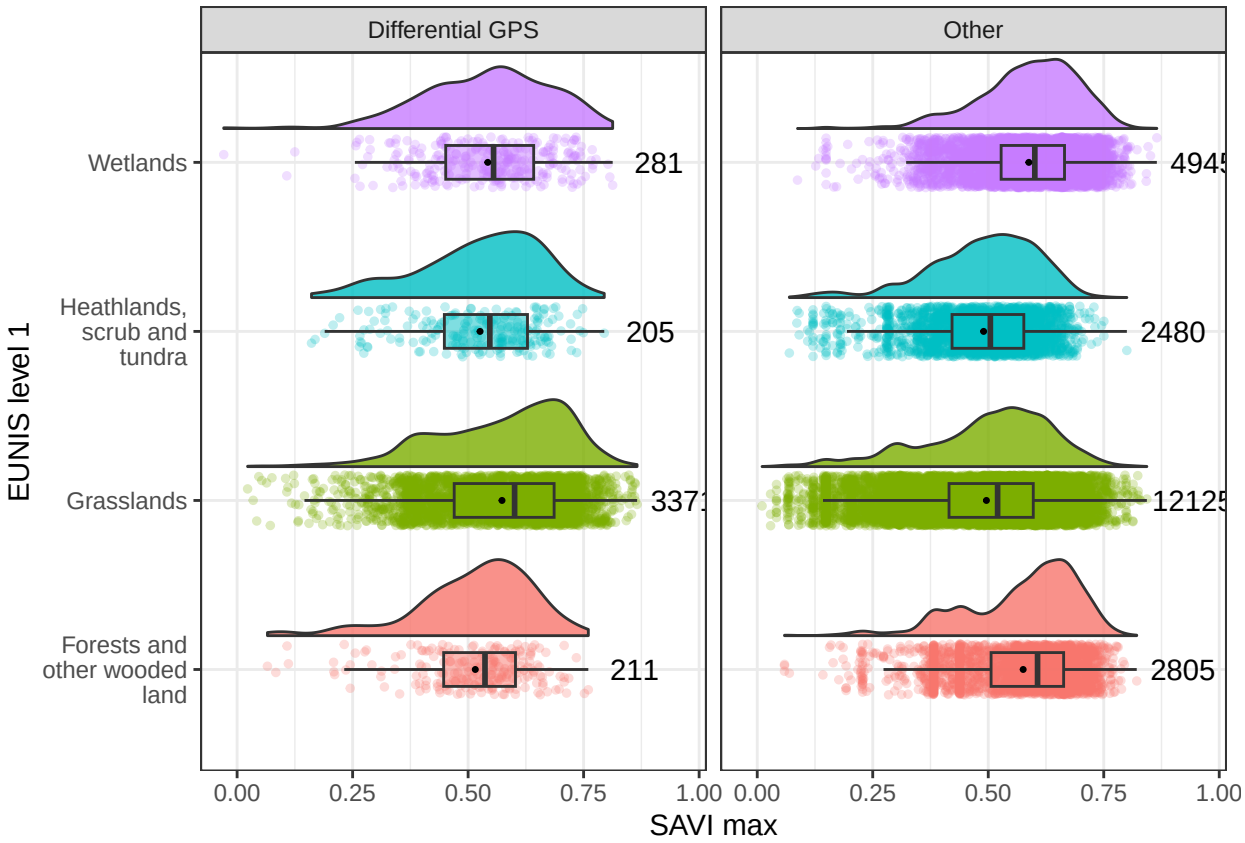

```
plot_distr_NDMI_GPS
```



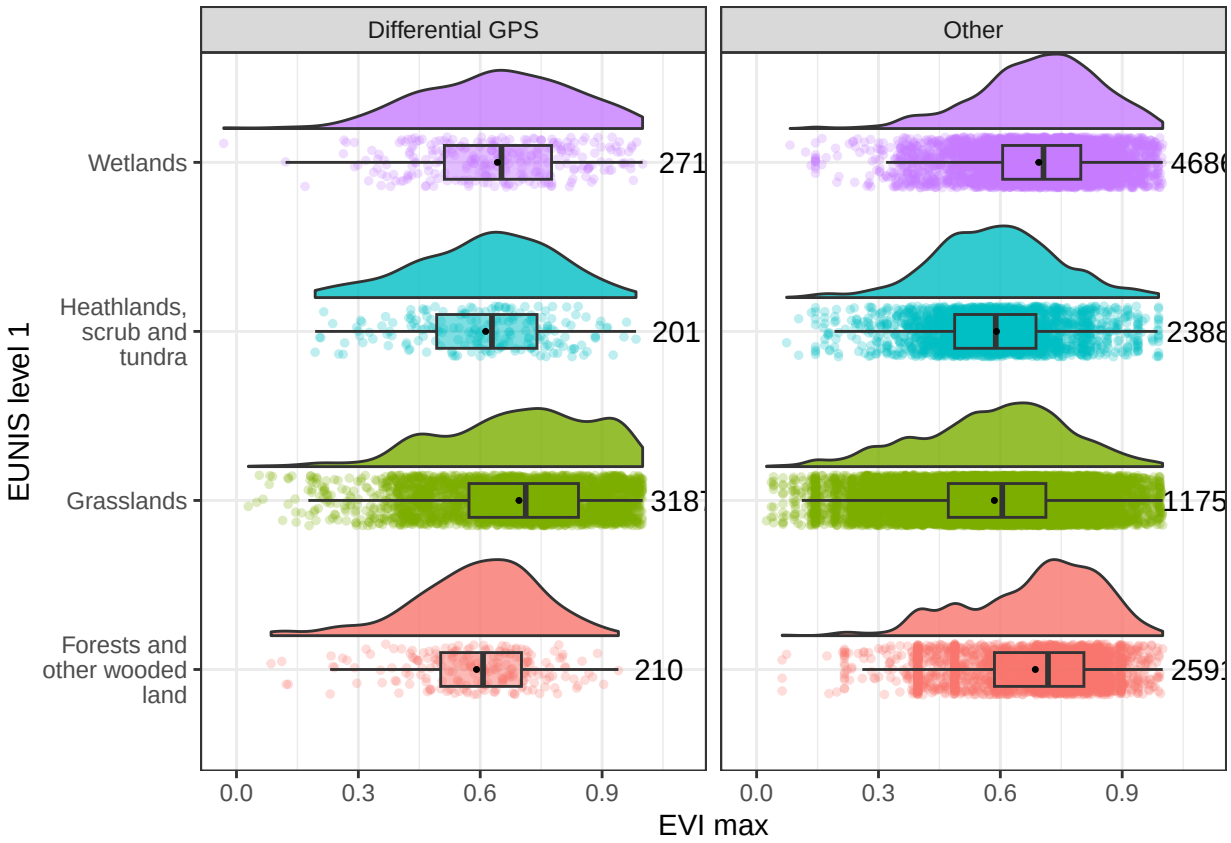

```
plot_distr_NDWI_GPS
```




```
plot_distr_SAVI_GPS
```




```
plot_distr_EVI_GPS
```



Including PLOT (USE LATER?)

Summarize variables by plot:

```
db_resurv_RS_short_PLOT_summ <- db_resurv_RS_short_PLOT %>%
  filter(EUNISa_1 %in% c("T", "R", "S", "Q")) %>%
  filter(S2_data == T | Landsat_data == T) %>%
  group_by(PLOT) %>%
  summarize(EUNIS1 = if_else(n_distinct(EUNISa_1) > 1, "Change",
                             unique(EUNISa_1)[1]),
            count = n(),
            across(starts_with("NDVI"), list(mean = mean, sd = sd),
                   .names = "{col}_{fn}"))
```

Maybe use later because now many plots have only one observation, probably because some Landsat data is missing?

First validation

For T, R, S, Q habitats.

Define a set of rules for a first validation of ALL ReSurvey data. We can call these “Expert-based” rules.

Number of observations in ReSurvey from the habitats of interest:

```
nrow(db_resurv_RS_short_PLOT %>%
       filter(EUNISa_1 %in% c("T", "R", "S", "Q")))
```

```
## [1] 181061
```

Number of observations in ReSurvey from the habitats of interest and with all RS data:

```
nrow(db_resurv_RS_short_PLOT %>%
       filter(EUNISa_1 %in% c("T", "R", "S", "Q")) %>%
       filter(CH_data == T) %>%
       filter(S2_data == T | Landsat_data ==T) %>%
       filter(S2_phen_data == T))
```

```
## [1] 11245
```

```
db_resurv_RS_short_PLOT_terrestrial <- db_resurv_RS_short_PLOT %>%
  filter(EUNISa_1 %in% c("T", "R", "S", "Q"))
```

Define rules

Create column for first validation based on different indicators, where "wrong" is noted when the validation rule is not met. Include EUNIS1 confusions.

```
db_resurv_RS_short_PLOT_terrestrial %>% count(EUNISa_1, EUNIS1_conf_type)
```

```
## # A tibble: 6 x 3
##   EUNISa_1 EUNIS1_conf_type      n
##   <chr>    <chr>             <int>
## 1 Q        <NA>              28083
## 2 R        R/S                  91
## 3 R        <NA>             106047
## 4 S        S/T                   9
## 5 S        <NA>              22954
## 6 T        <NA>              23877
```

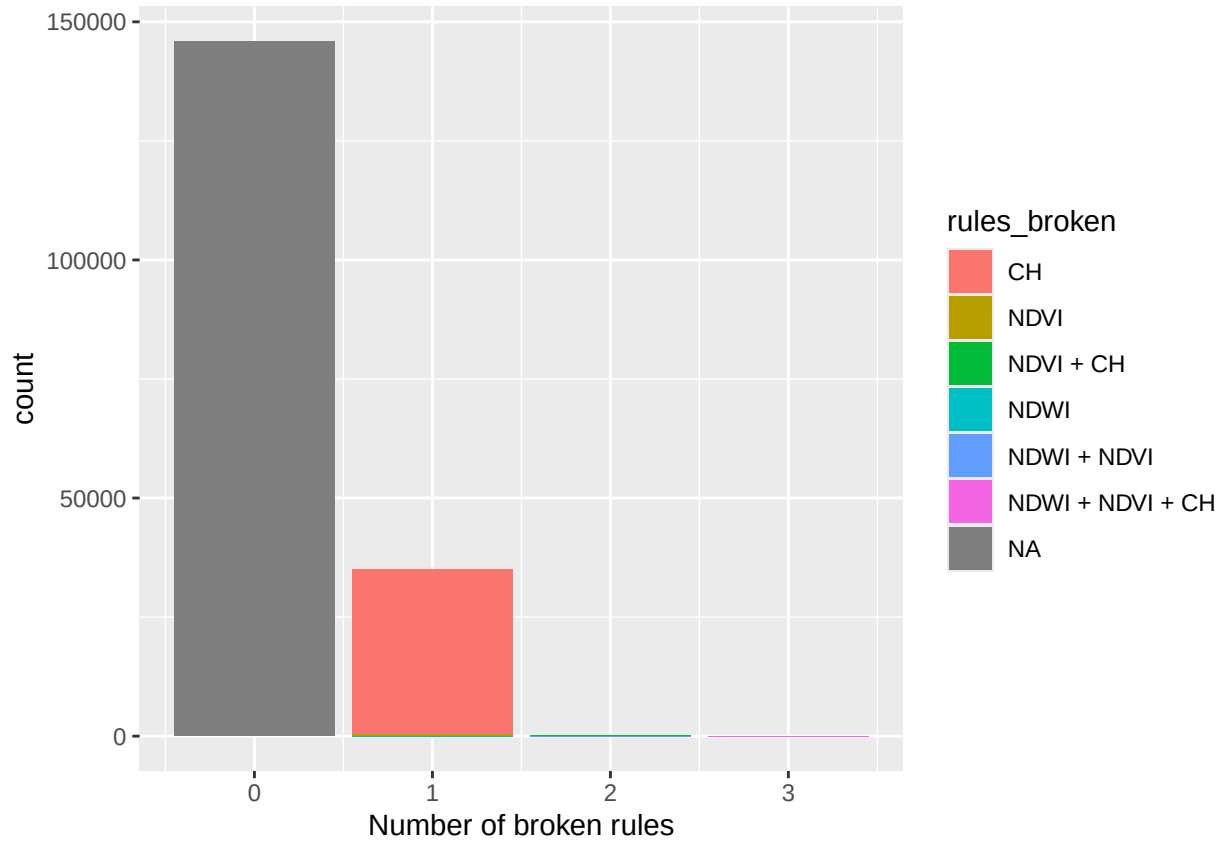
Define rules:

```
db_resurv_RS_short_PLOT_terrestrial <- db_resurv_RS_short_PLOT_terrestrial %>%
  mutate(
    valid_1_NDWI = case_when(
      # Points that are basically water
      NDWI_max > 0.3 ~ "wrong",
      TRUE ~ NA_character_),
    valid_1_CH = case_when(
      # T points with low CH
      EUNISa_1 == "T" & canopy_height < 8 ~ "wrong",
      # S points with low CH
      EUNISa_1 =="S" & canopy_height < 5 ~ "wrong",
      # R & Q points with high CH
      EUNISa_1 %in% c("R", "Q") & canopy_height > 2 ~ "wrong",
```

```
      TRUE ~ NA_character_),
    valid_1_NDVI = case_when(
      # T points with low NDVI_max
      EUNISa_1 == "T" & NDVI_max < 0.6 ~ "wrong",
      # S-R-Q points with low NDVI_max
      EUNISa_1 %in% c("R", "S", "Q") & NDVI_max < 0.2 ~ "wrong",
      TRUE ~ NA_character_),
    # Count how many validation rules are not met
    valid_1_count = rowSums(across(c(valid_1_NDWI, valid_1_CH, valid_1_NDVI),
                                   ~ . == "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")
    )
```

Plots first validation

```
ggplot(db_resurv_RS_short_PLOT_terrestrial%>%
         mutate(rules_broken = case_when(
           valid_1_count == 1 & valid_1_NDWI == "wrong" ~ "NDWI",
           valid_1_count == 1 & valid_1_NDVI == "wrong" ~ "NDVI",
           valid_1_count == 1 & valid_1_CH == "wrong" ~ "CH",
           valid_1_count == 2 &
             valid_1_NDWI == "wrong" & valid_1_NDVI == "wrong"~ "NDWI + NDVI",
           valid_1_count == 2 &
             valid_1_NDWI == "wrong" & valid_1_CH == "wrong"~ "NDWI + CH",
           valid_1_count == 2 &
             valid_1_NDVI == "wrong" & valid_1_CH == "wrong"~ "NDVI + CH",
           valid_1_count == 3 ~ "NDWI + NDVI + CH",
           TRUE ~ NA_character_
         )),
       aes(x = valid_1_count, fill = rules_broken)) +
  geom_bar() + labs(x = "Number of broken rules")
```




```
db_resurv_RS_short_PLOT_terrestrial %>%
         mutate(rules_broken = case_when(
           valid_1_count == 1 & valid_1_NDWI == "wrong" ~ "NDWI",
           valid_1_count == 1 & valid_1_NDVI == "wrong" ~ "NDVI",
           valid_1_count == 1 & valid_1_CH == "wrong" ~ "CH",
           valid_1_count == 2 &
             valid_1_NDWI == "wrong" & valid_1_NDVI == "wrong"~ "NDWI + NDVI",
           valid_1_count == 2 &
             valid_1_NDWI == "wrong" & valid_1_CH == "wrong"~ "NDWI + CH",
           valid_1_count == 2 &
             valid_1_NDVI == "wrong" & valid_1_CH == "wrong"~ "NDVI + CH",
           valid_1_count == 3 ~ "NDWI + NDVI + CH",
           TRUE ~ NA_character_
         )) %>%
  count(rules_broken, EUNIS1_conf_type)
```

```
## # A tibble: 12 x 3
##    rules_broken     EUNIS1_conf_type      n
##    <chr>            <chr>             <int>
##  1 CH               R/S                   3
##  2 CH               S/T                   4
##  3 CH               <NA>              34597
##  4 NDVI             R/S                   2
##  5 NDVI             <NA>                365
##  6 NDVI + CH        <NA>                180
```

```
##  7 NDWI             <NA>                 1
##  8 NDWI + NDVI      <NA>                 2
##  9 NDWI + NDVI + CH <NA>                 3
## 10 <NA>             R/S                 86
## 11 <NA>             S/T                  5
## 12 <NA>             <NA>            145813
```

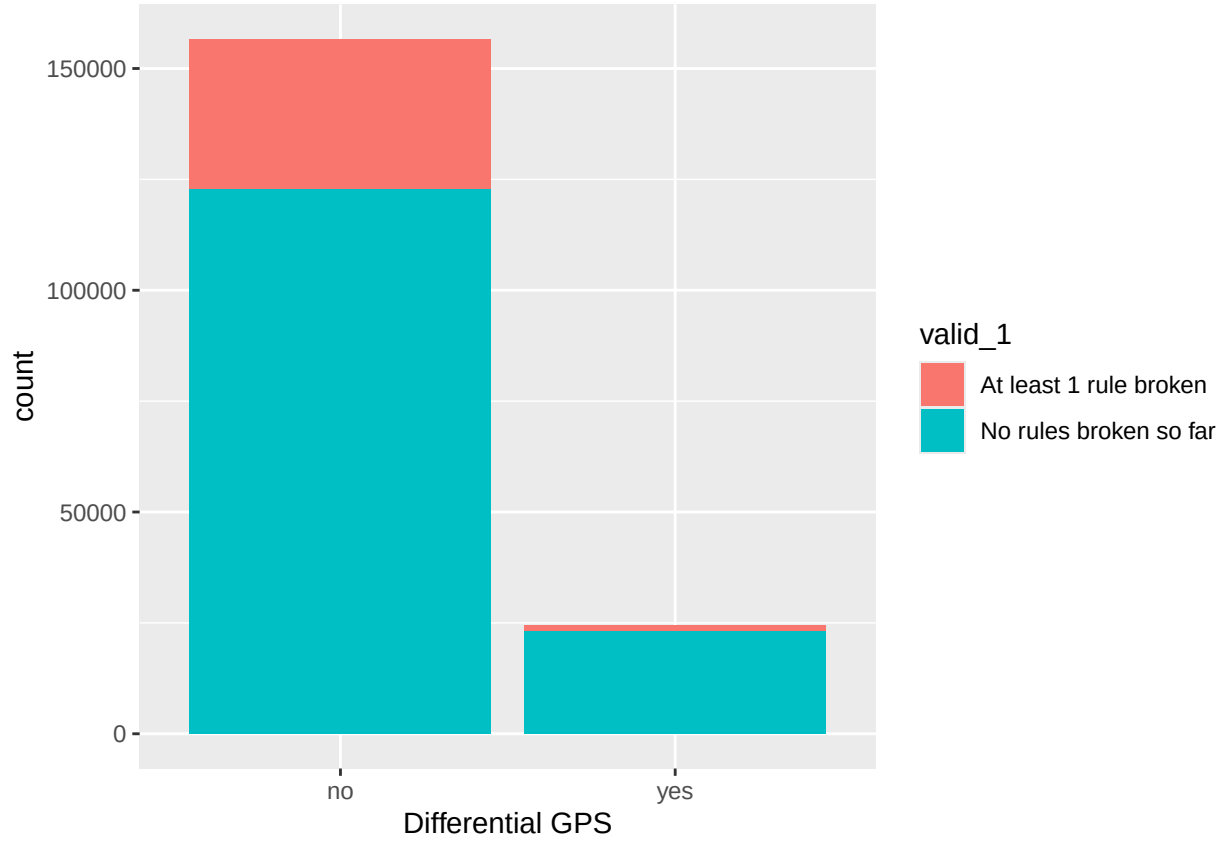
Proportion of observations not validated (so far):

```
nrow(db_resurv_RS_short_PLOT_terrestrial %>% filter(valid_1_count > 0))/
  nrow(db_resurv_RS_short_PLOT_terrestrial)
```

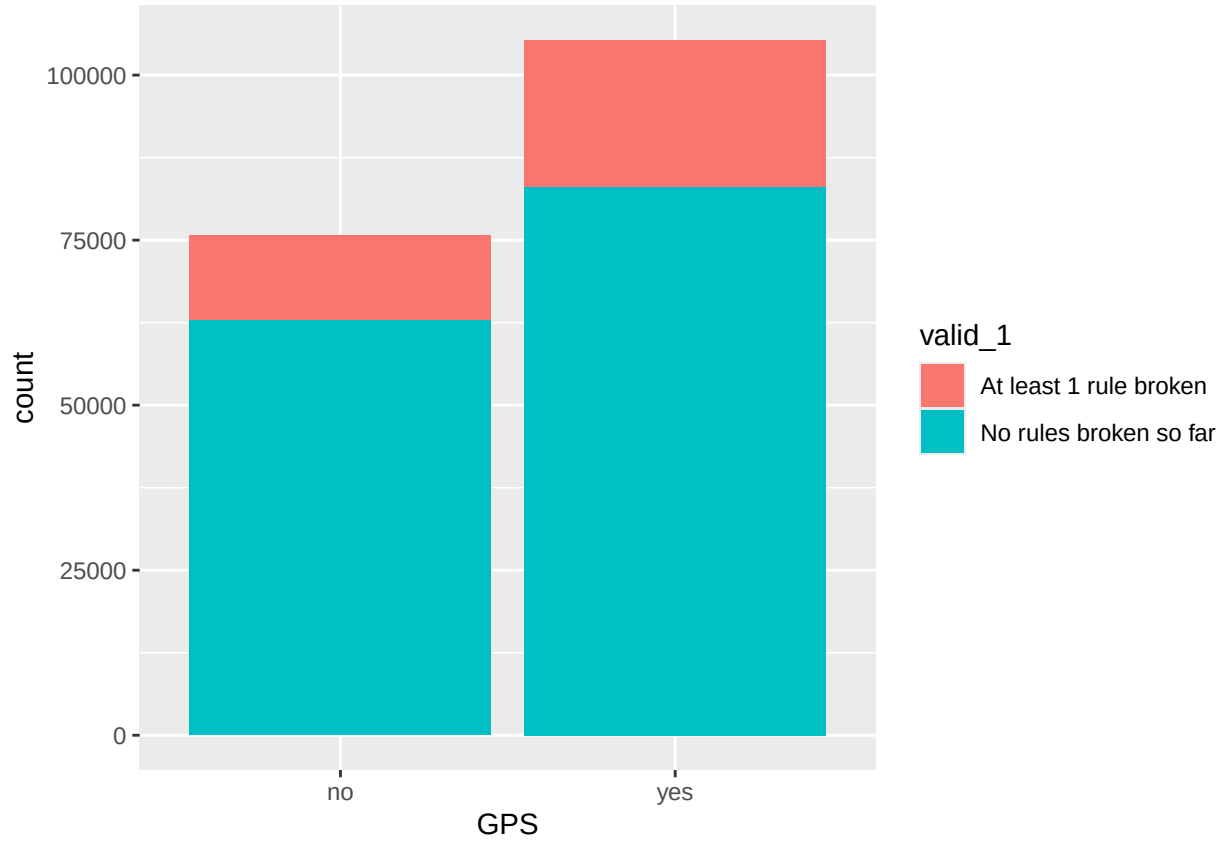
```
## [1] 0.1941721
```

But be aware that there are still many missing RS data, e.g. Landsat data for bioregion CON (many points).

```
ggplot(db_resurv_RS_short_PLOT_terrestrial %>%
         mutate(diff_GPS = if_else(
           `Location method` != "Location with differential GPS" |
             is.na(`Location method`), "no", "yes")),
       aes(x = diff_GPS, fill = valid_1)) +
  geom_bar() + labs(x = "Differential GPS")
```



```
ggplot(db_resurv_RS_short_PLOT_terrestrial %>%
         mutate(GPS = case_when(
           `Location method` == "Location with differential GPS" ~ "yes",
           `Location method` == "Location with GPS" ~ "yes",
           is.na(`Location method`) ~ "no",
           TRUE ~ "no"
         )),
       aes(x = GPS, fill = valid_1)) +
  geom_bar() + labs(x = "GPS")
```



Points with any rule broken and confusion between EUNIS:

```
nrow(db_resurv_RS_short_PLOT_terrestrial %>%
       filter(EUNIS1_conf == T & valid_1_count > 0))
```

```
## [1] 9
```

Convert to shp to look at these in GIS:

```
# st_write(db_resurv_RS_short_PLOT_terrestrial %>%
#            filter(EUNIS1_conf == T & valid_1_count > 0) %>%
#            st_as_sf(coords = c("Lon_updated", "Lat_updated"), crs = 4326),
#          "C:/GIS/MOTIVATE/shapefiles/resurv_not_val_EUNIS_conf.shp")
```

Checked and yes

How many points with differential GPS that have at least 1 rule broken?

```
nrow(db_resurv_RS_short_PLOT_terrestrial %>%
  filter(`Location method` == "Location with differential GPS" &
           valid_1 == "At least 1 rule broken"))
```

```
## [1] 1202
```

Convert to shp to look at these in GIS:

```
# st_write(db_resurv_RS_short_PLOT_terrestrial %>%
#            filter(`Location method` == "Location with differential GPS" &
#                     valid_1 == "At least 1 rule broken") %>%
#            st_as_sf(coords = c("Lon_updated", "Lat_updated"), crs = 4326),
#          "C:/GIS/MOTIVATE/shapefiles/resurv_not_val_diff_GPS.shp")
```

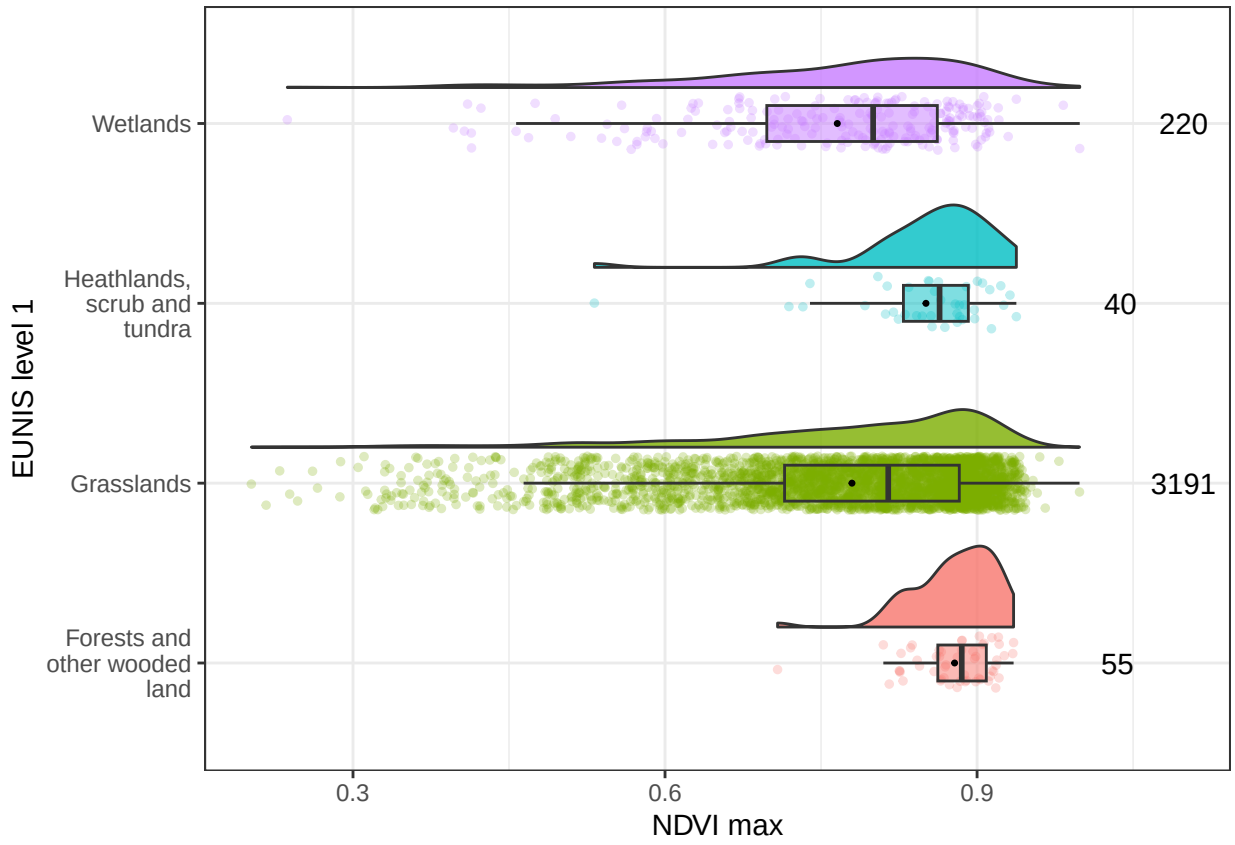
Distributions from diff GPS points without rules broken so far

Create tibble with differential GPS points without rules broken so far:

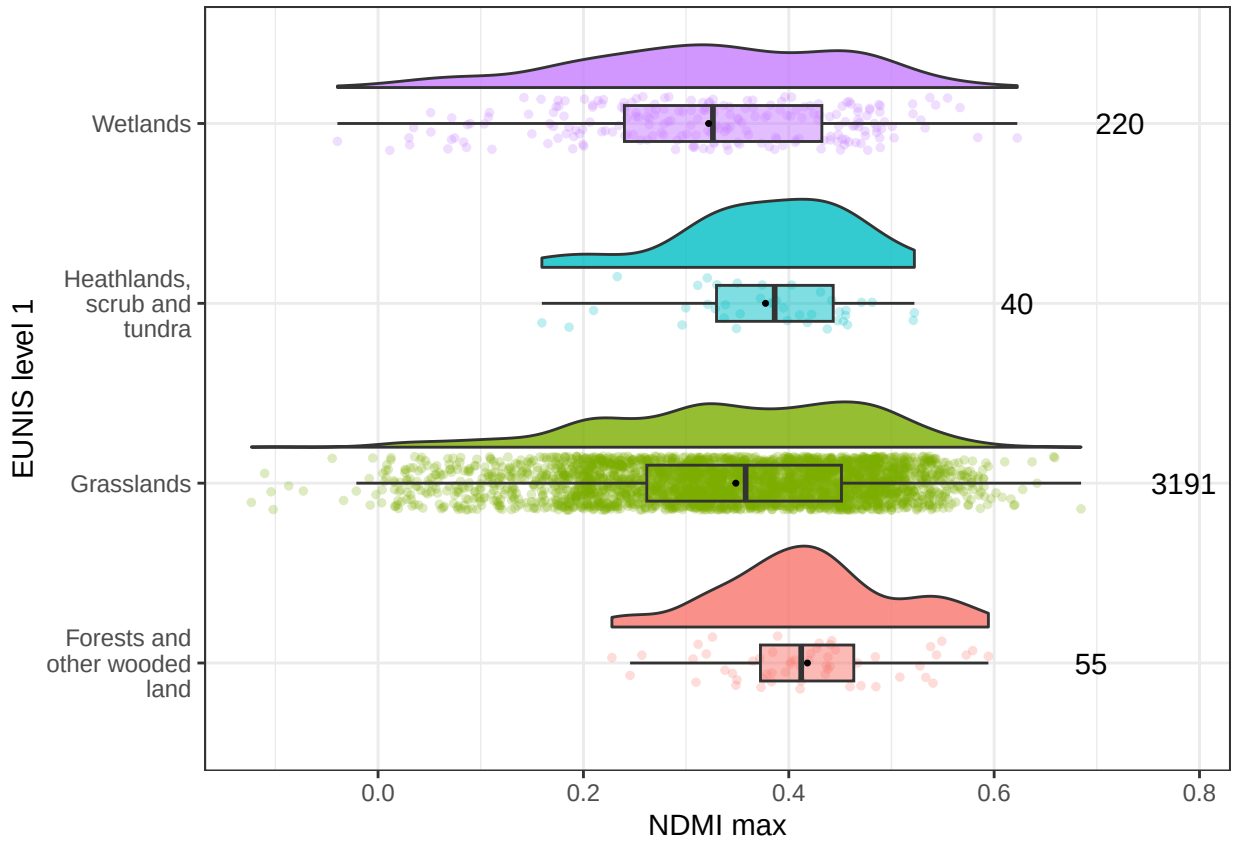
```
diff_GPS_valid <- db_resurv_RS_short_PLOT_terrestrial %>%
  filter(`Location method` == "Location with differential GPS" &
           valid_1 == "No rules broken so far")
```

Max. and min. NDVI, NDMI, NDWI, SAVI and EVI

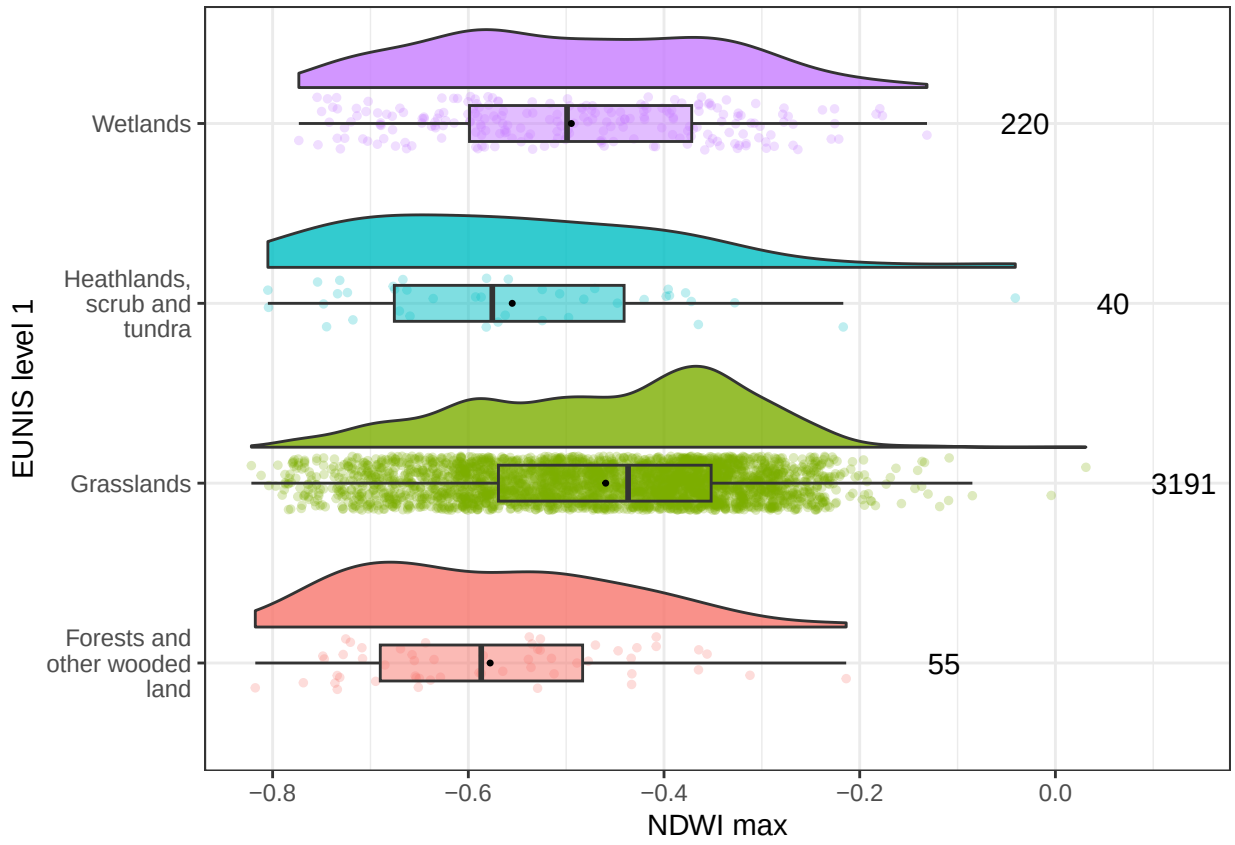
```
plot_distr_NDVI_max_diff_GPS_valid <- distr_plot(diff_GPS_valid,
                                                 "NDVI_max", "NDVI max")
plot_distr_NDMI_max_diff_GPS_valid <- distr_plot(diff_GPS_valid,
                                                 "NDMI_max", "NDMI max")
plot_distr_NDWI_max_diff_GPS_valid <- distr_plot(diff_GPS_valid,
                                                 "NDWI_max", "NDWI max")
plot_distr_SAVI_max_diff_GPS_valid <- distr_plot(diff_GPS_valid,
                                                 "SAVI_max", "SAVI max")
plot_distr_EVI_max_diff_GPS_valid <- distr_plot(diff_GPS_valid %>%
                                                  filter(EVI_max <= 1),
                                                "EVI_max", "EVI max")
plot_distr_NDVI_max_diff_GPS_valid
```



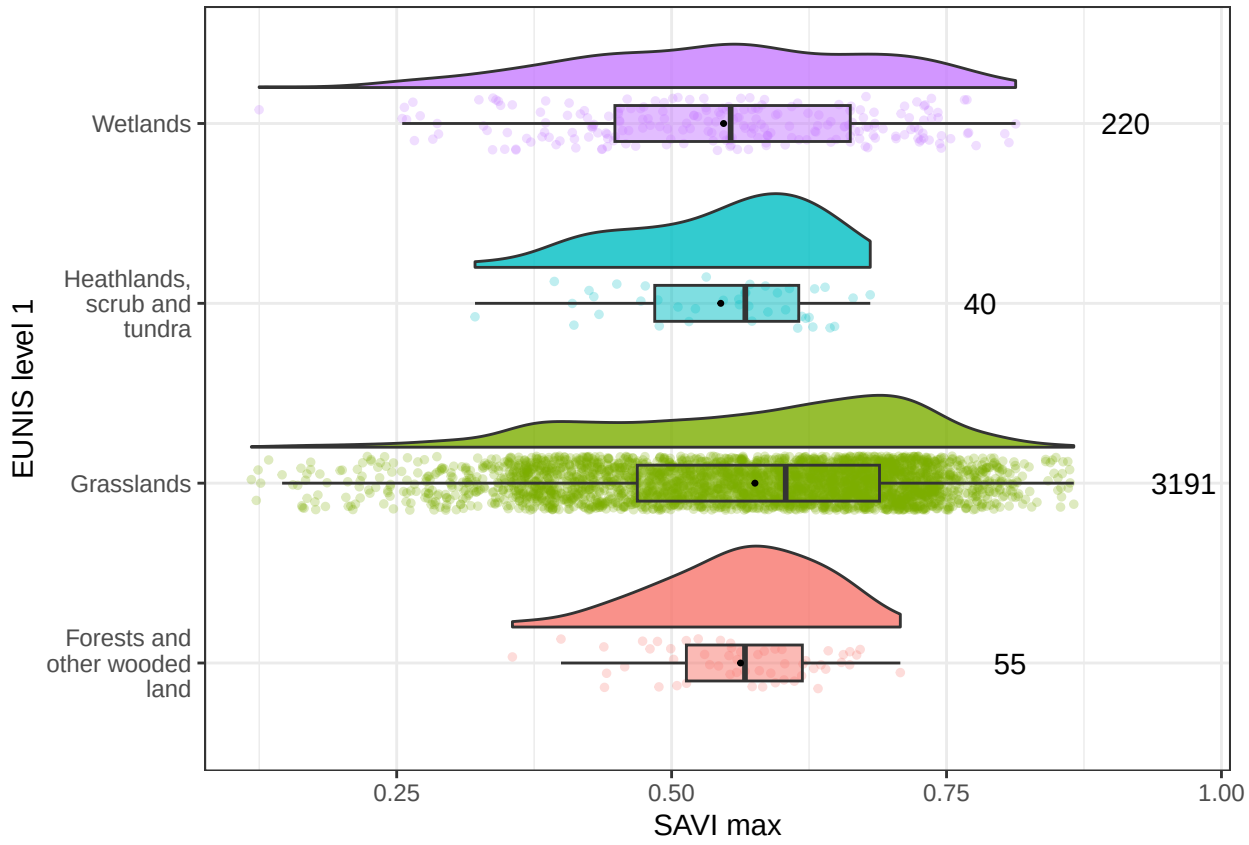

```
plot_distr_NDMI_max_diff_GPS_valid
```



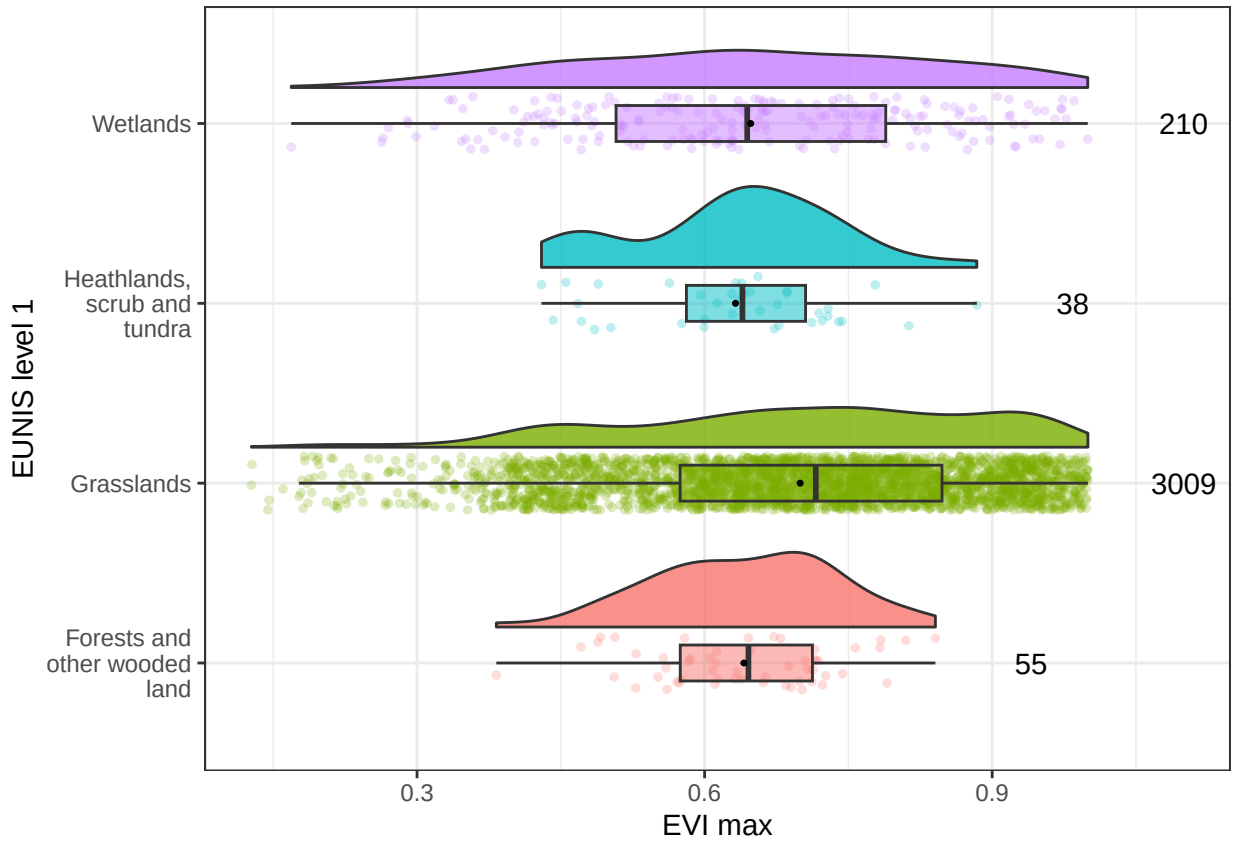

```
plot_distr_NDWI_max_diff_GPS_valid
```



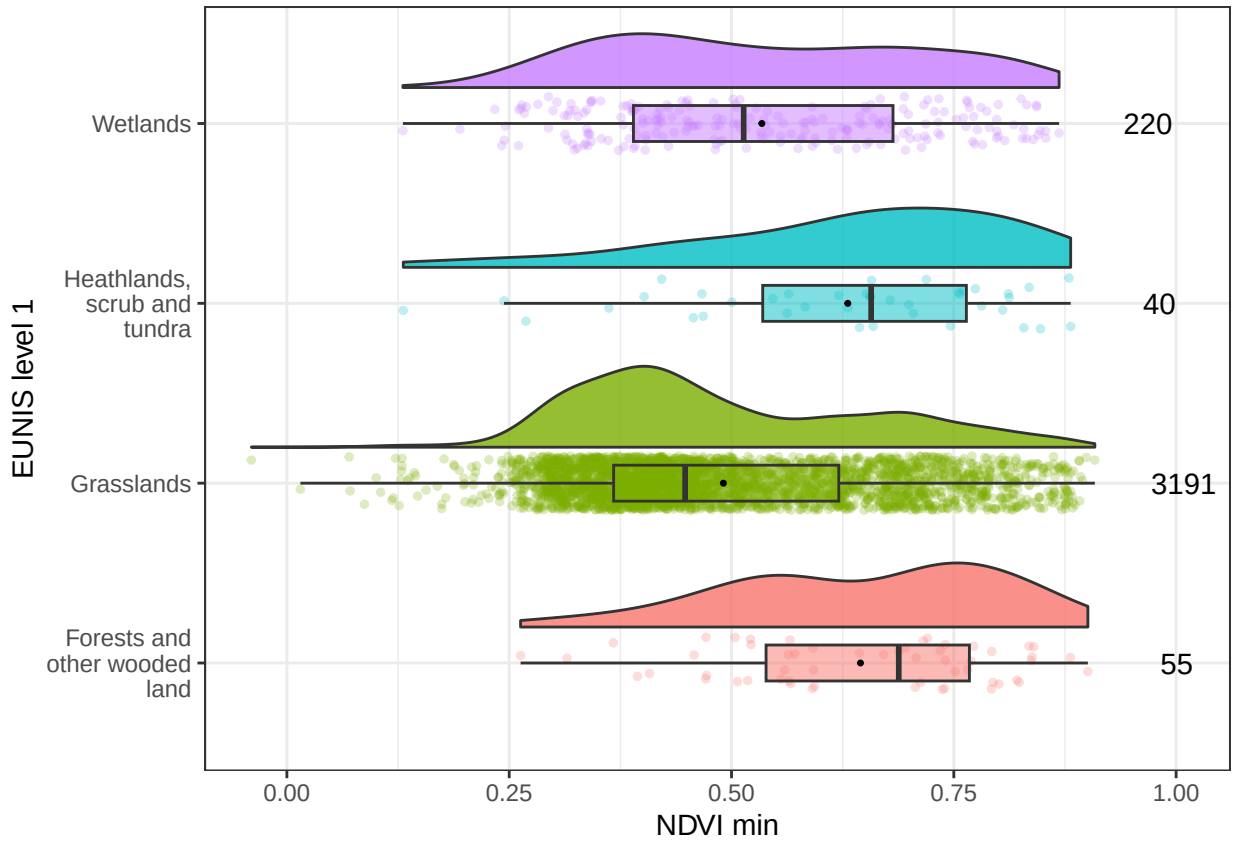

```
plot_distr_SAVI_max_diff_GPS_valid
```



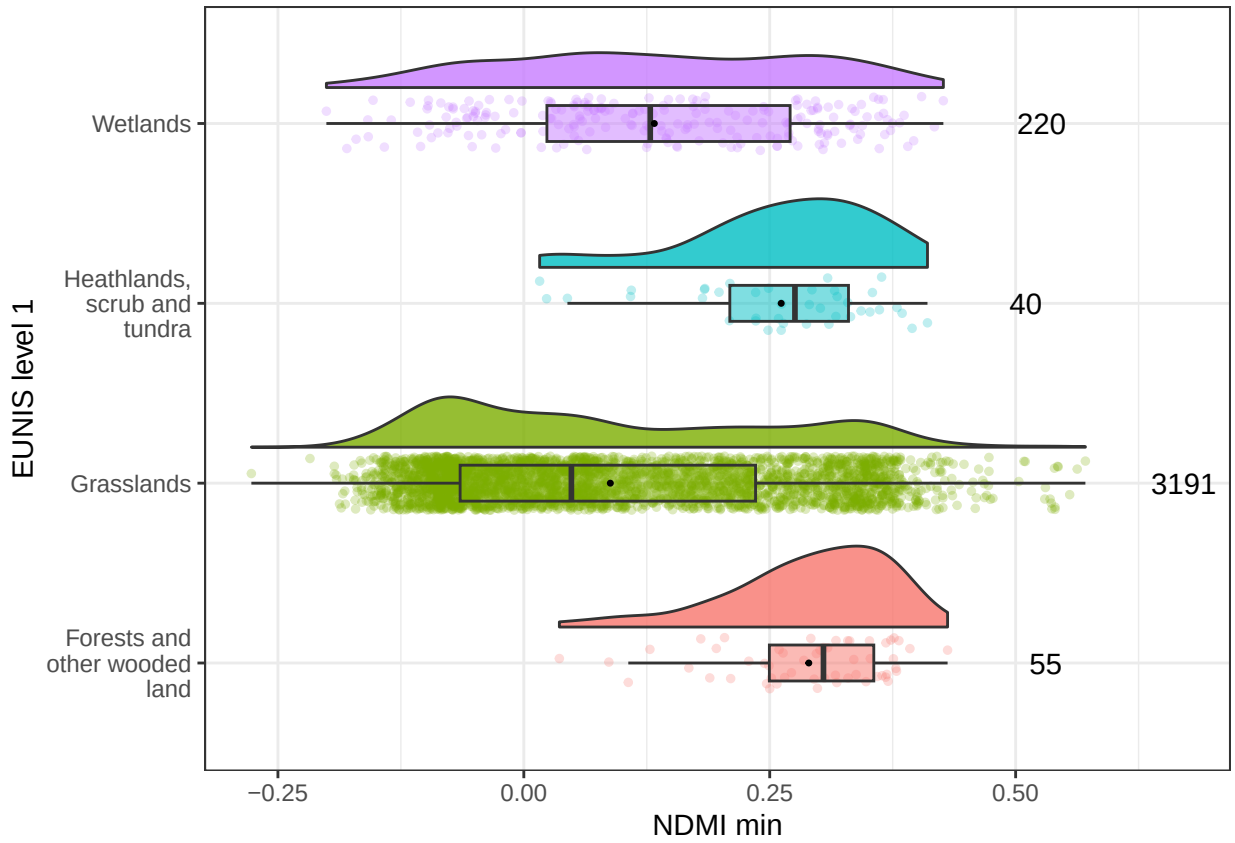

```
plot_distr_EVI_max_diff_GPS_valid
```



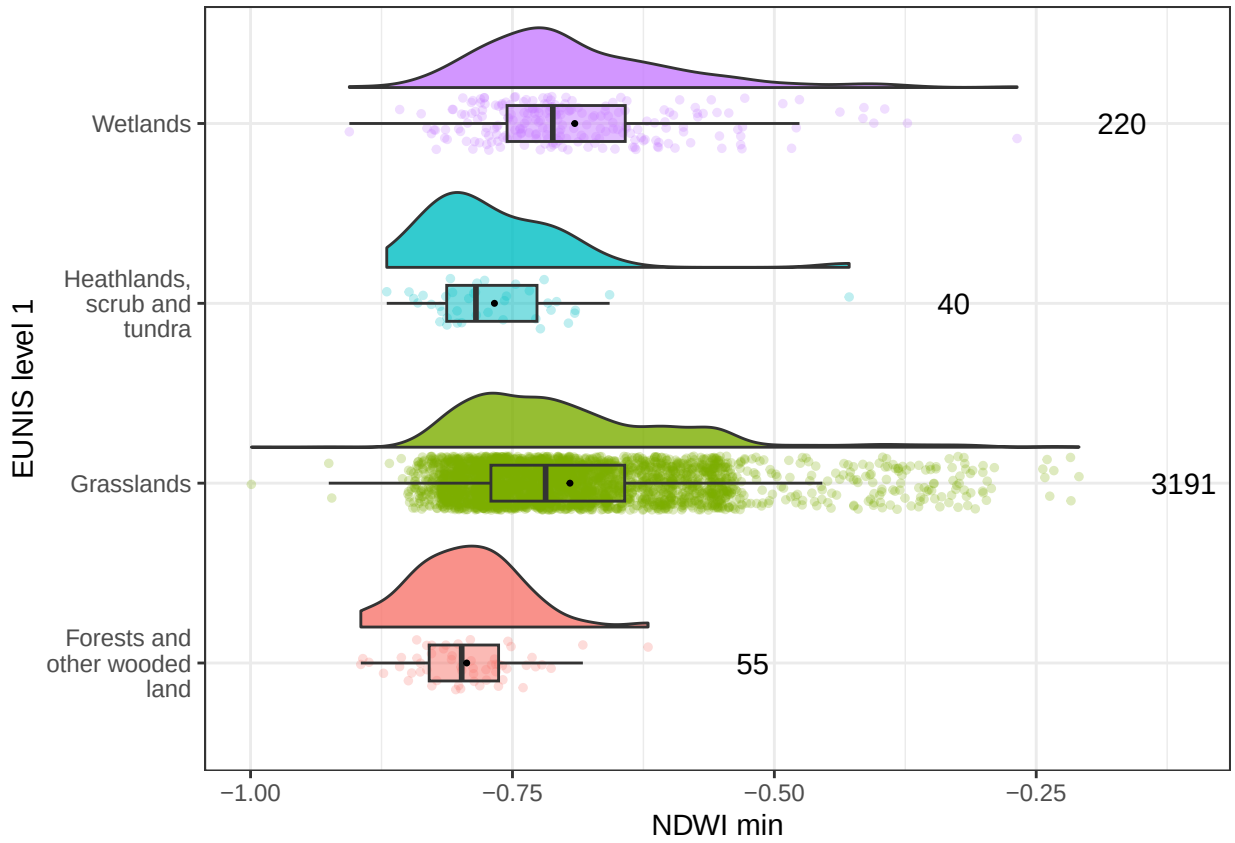

```
plot_distr_NDVI_min_diff_GPS_valid <- distr_plot(diff_GPS_valid,
                                                 "NDVI_min", "NDVI min")
plot_distr_NDMI_min_diff_GPS_valid <- distr_plot(diff_GPS_valid,
                                                 "NDMI_min", "NDMI min")
plot_distr_NDWI_min_diff_GPS_valid <- distr_plot(diff_GPS_valid,
                                                 "NDWI_min", "NDWI min")
plot_distr_SAVI_min_diff_GPS_valid <- distr_plot(diff_GPS_valid,
                                                 "SAVI_min", "SAVI min")
plot_distr_EVI_min_diff_GPS_valid <- distr_plot(diff_GPS_valid %>%
                                                  # Some values wrong!
                                                  # EVI should not be lower than -1
                                                  filter(EVI_min >= -1),
                                                "EVI_min", "EVI min")
plot_distr_NDVI_min_diff_GPS_valid
```



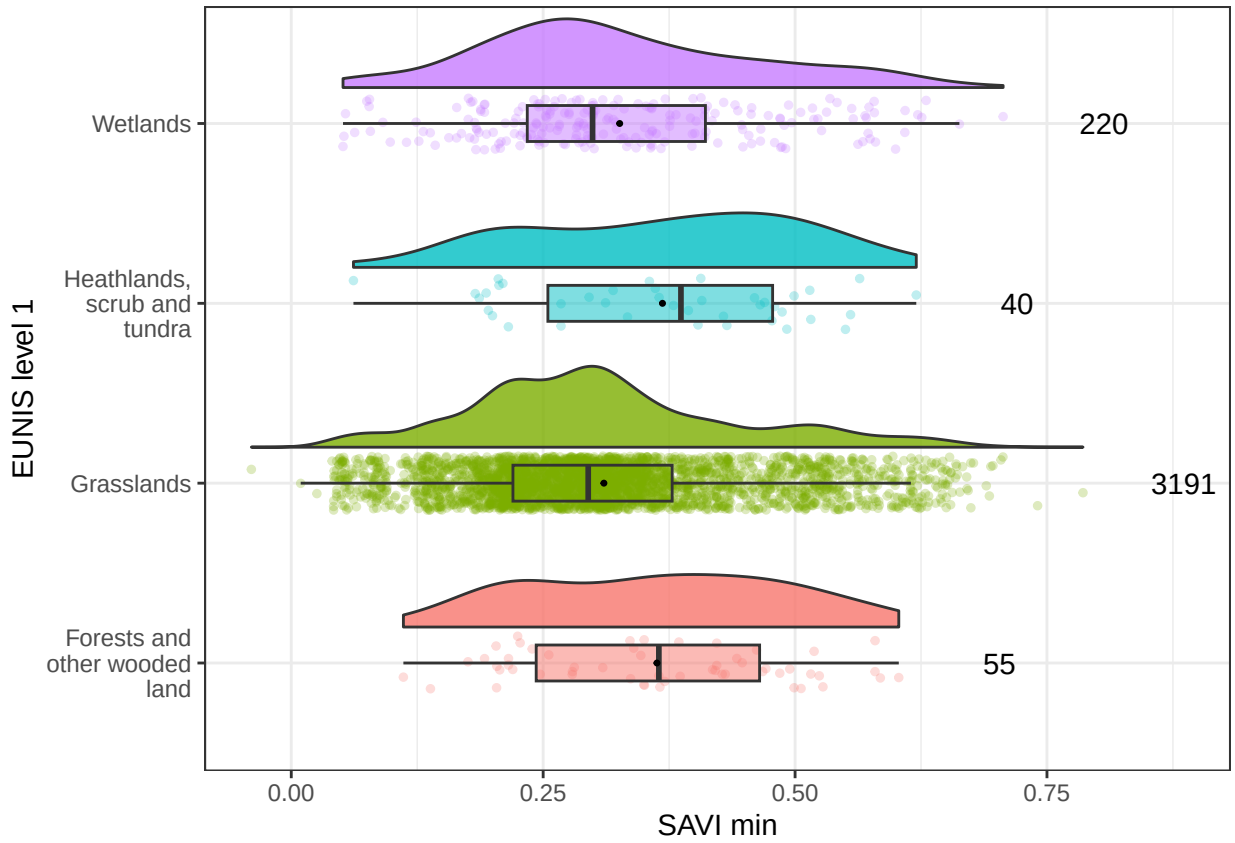

```
plot_distr_NDMI_min_diff_GPS_valid
```



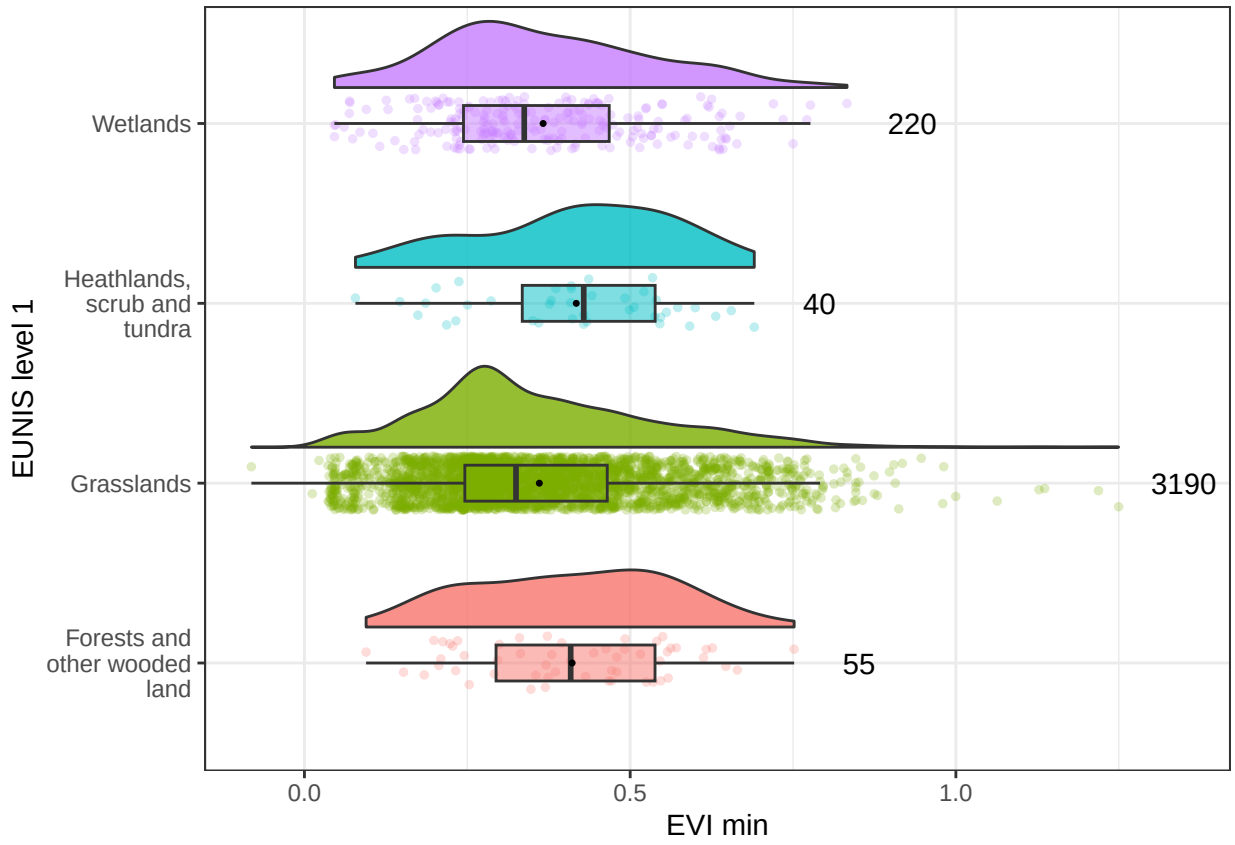

```
plot_distr_NDWI_min_diff_GPS_valid
```




```
plot_distr_SAVI_min_diff_GPS_valid
```

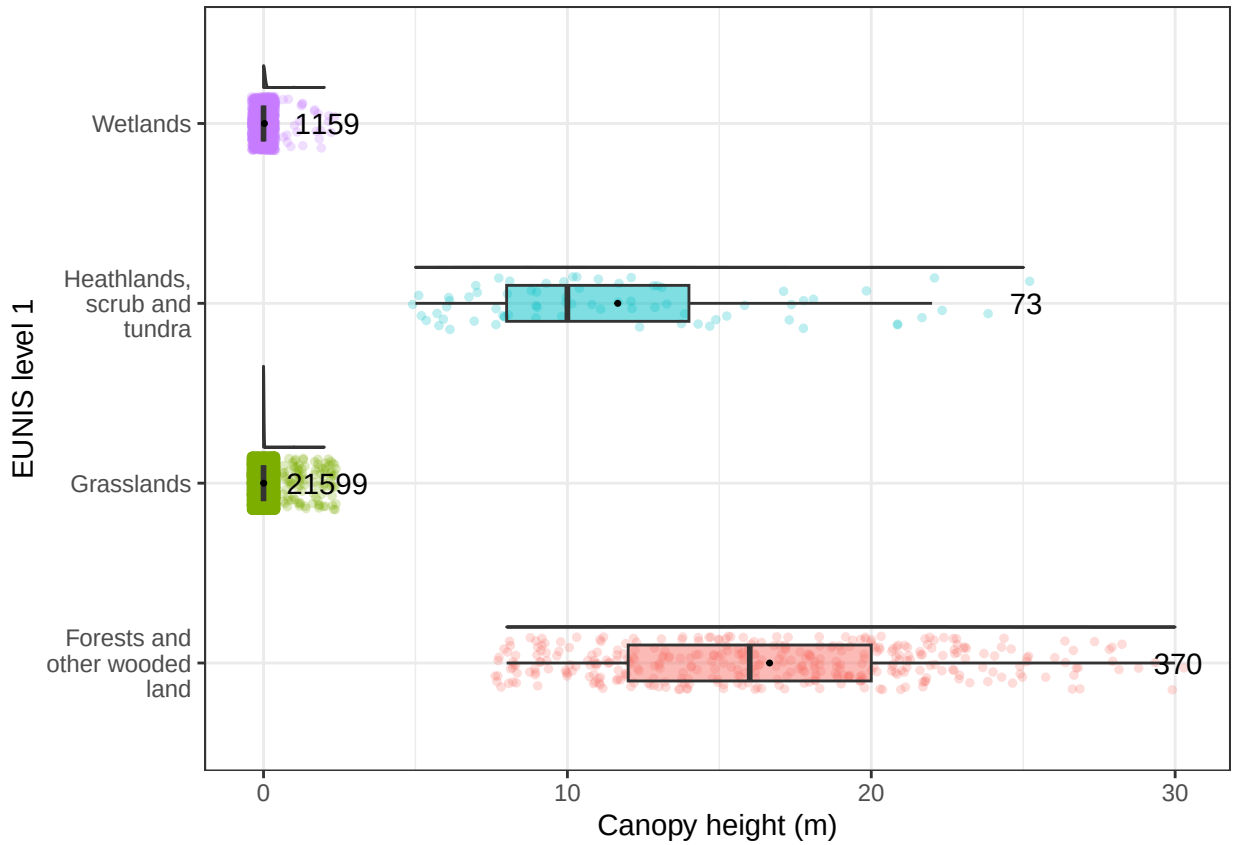



```
plot_distr_EVI_min_diff_GPS_valid
```



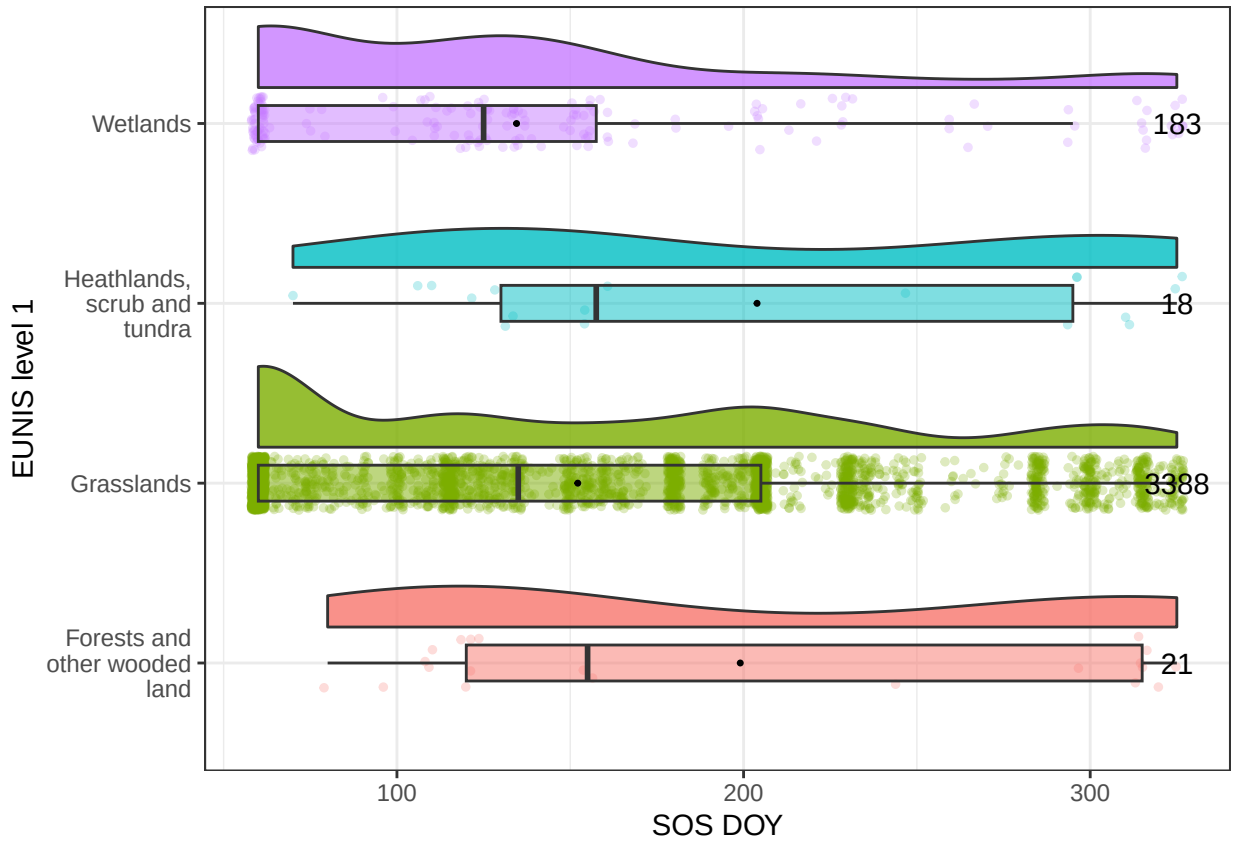
CH

```
plot_distr_CH_diff_GPS_valid <- distr_plot(diff_GPS_valid, "canopy_height",
                                           "Canopy height (m)")
plot_distr_CH_diff_GPS_valid
```

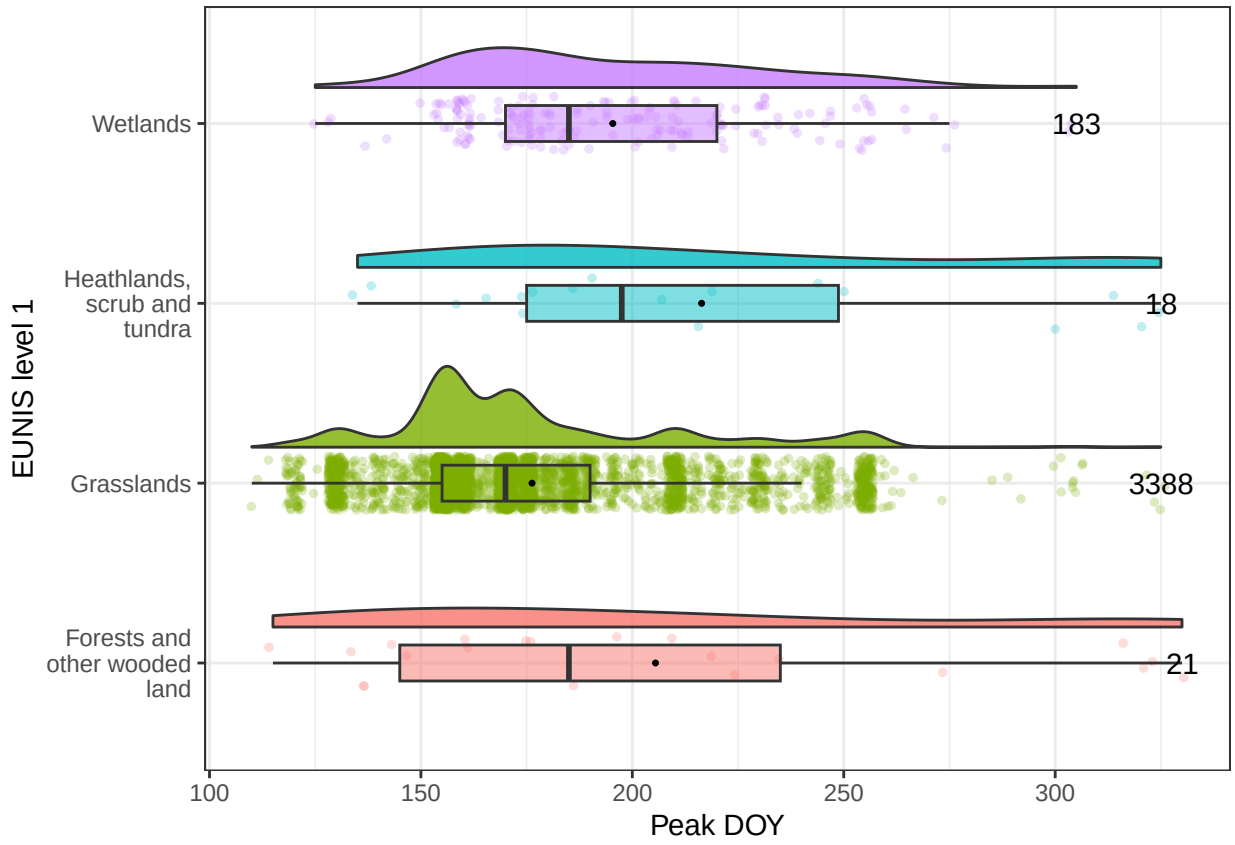


Phenology

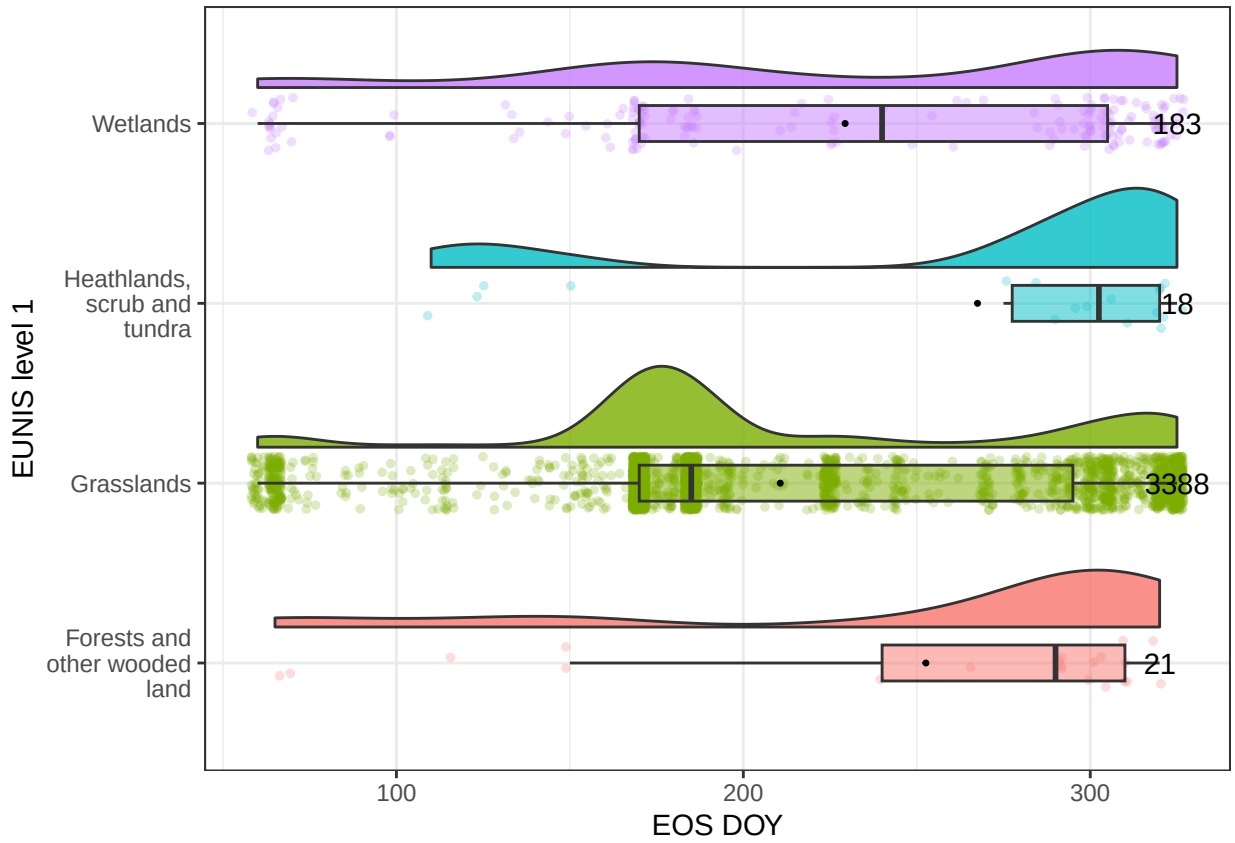
```
plot_distr_SOS_DOY_diff_GPS_valid <- distr_plot(diff_GPS_valid,
                                                "SOS_DOY", "SOS DOY")
plot_distr_Peak_DOY_diff_GPS_valid <- distr_plot(diff_GPS_valid,
                                                 "Peak_DOY", "Peak DOY")
plot_distr_EOS_DOY_diff_GPS_valid <- distr_plot(diff_GPS_valid,
                                                "EOS_DOY", "EOS DOY")
plot_distr_SOS_DOY_diff_GPS_valid
```




```
plot_distr_Peak_DOY_diff_GPS_valid
```




```
plot_distr_EOS_DOY_diff_GPS_valid
```



Not very conclusive...

Distributions from GPS (diff or not) points without rules broken so far

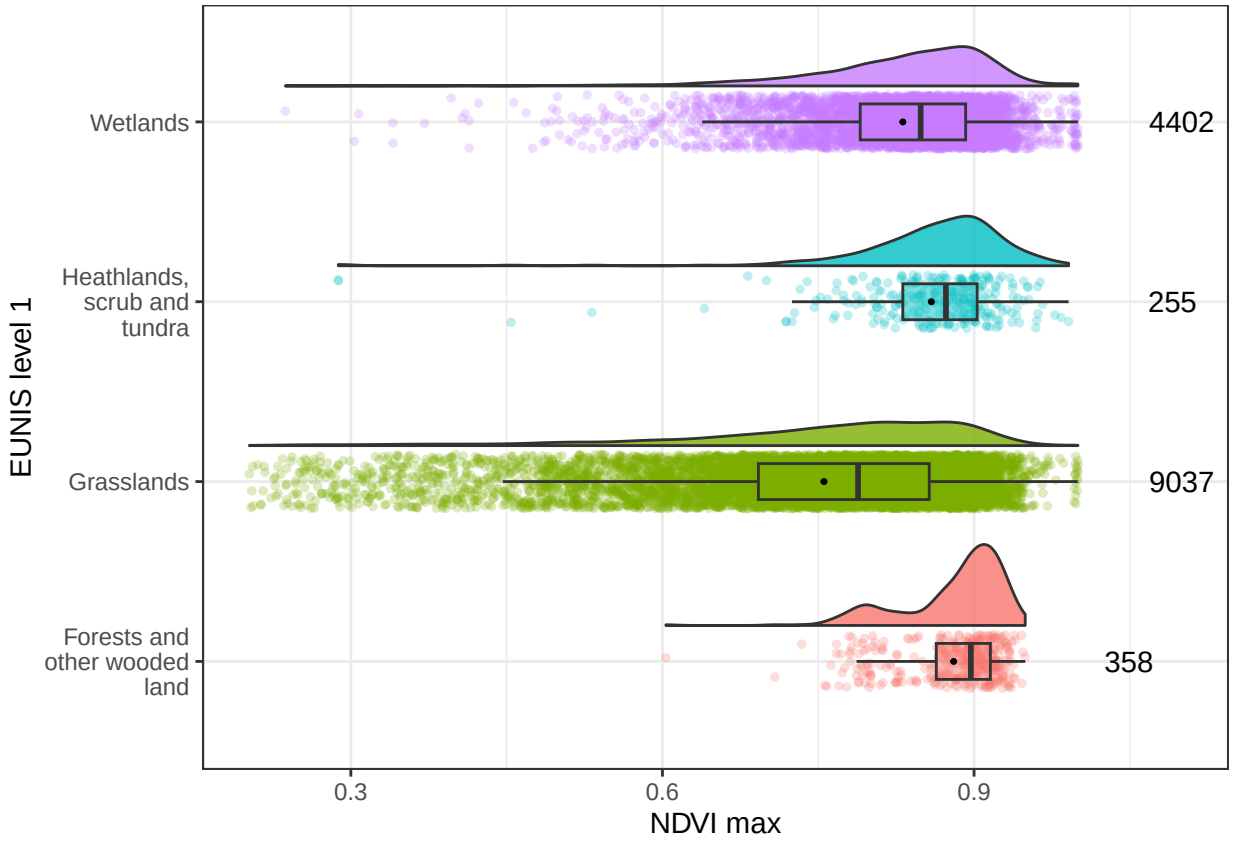
Create tibble with differential GPS points without rules broken so far:

```
all_GPS_valid <- db_resurv_RS_short_PLOT_terrestrial %>%
  filter((`Location method` == "Location with differential GPS" |
            `Location method` == "Location with GPS" ) &
           valid_1 == "No rules broken so far")
```

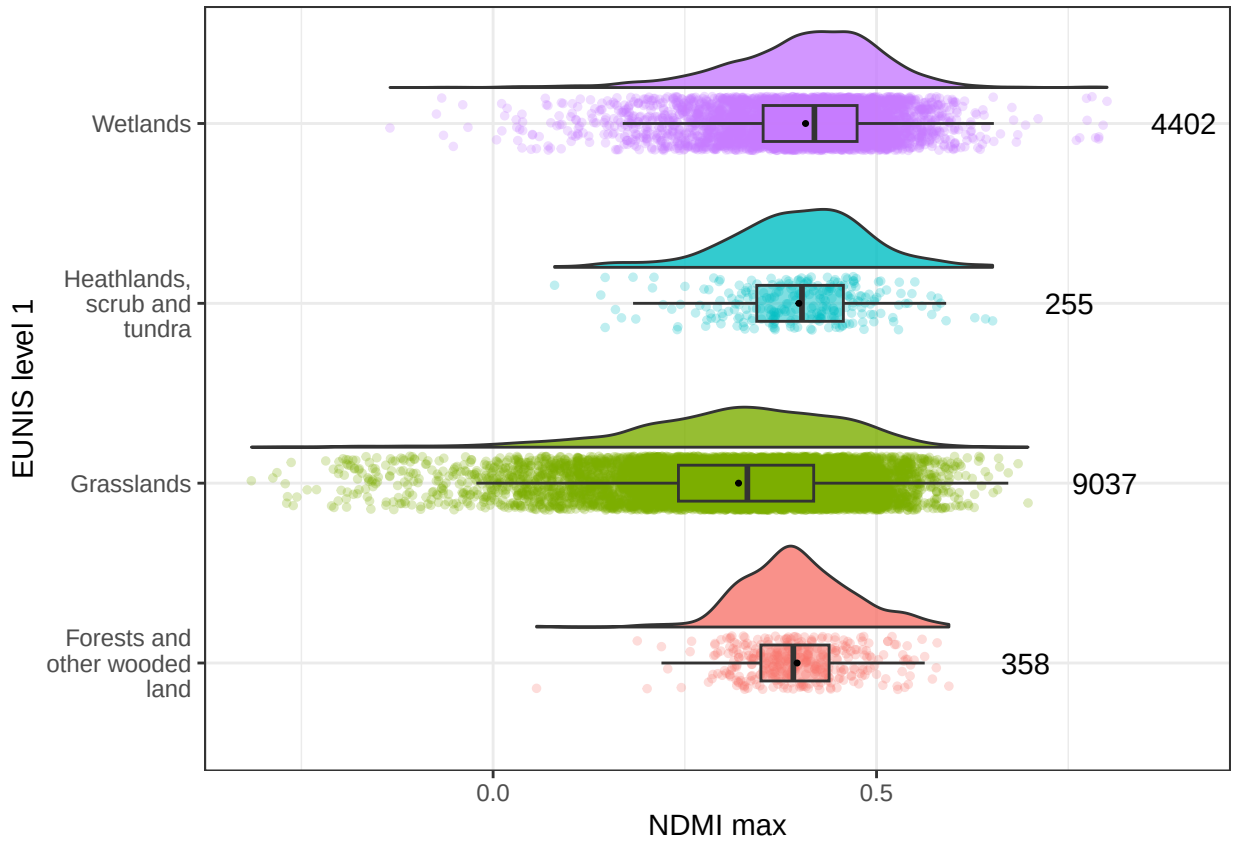
Max. and min. NDVI, NDMI, NDWI, SAVI and EVI

```
plot_distr_NDVI_max_all_GPS_valid <- distr_plot(all_GPS_valid,
                                                "NDVI_max", "NDVI max")
plot_distr_NDMI_max_all_GPS_valid <- distr_plot(all_GPS_valid,
                                                "NDMI_max", "NDMI max")
plot_distr_NDWI_max_all_GPS_valid <- distr_plot(all_GPS_valid,
                                                "NDWI_max", "NDWI max")
plot_distr_SAVI_max_all_GPS_valid <- distr_plot(all_GPS_valid,
                                                "SAVI_max", "SAVI max")
```

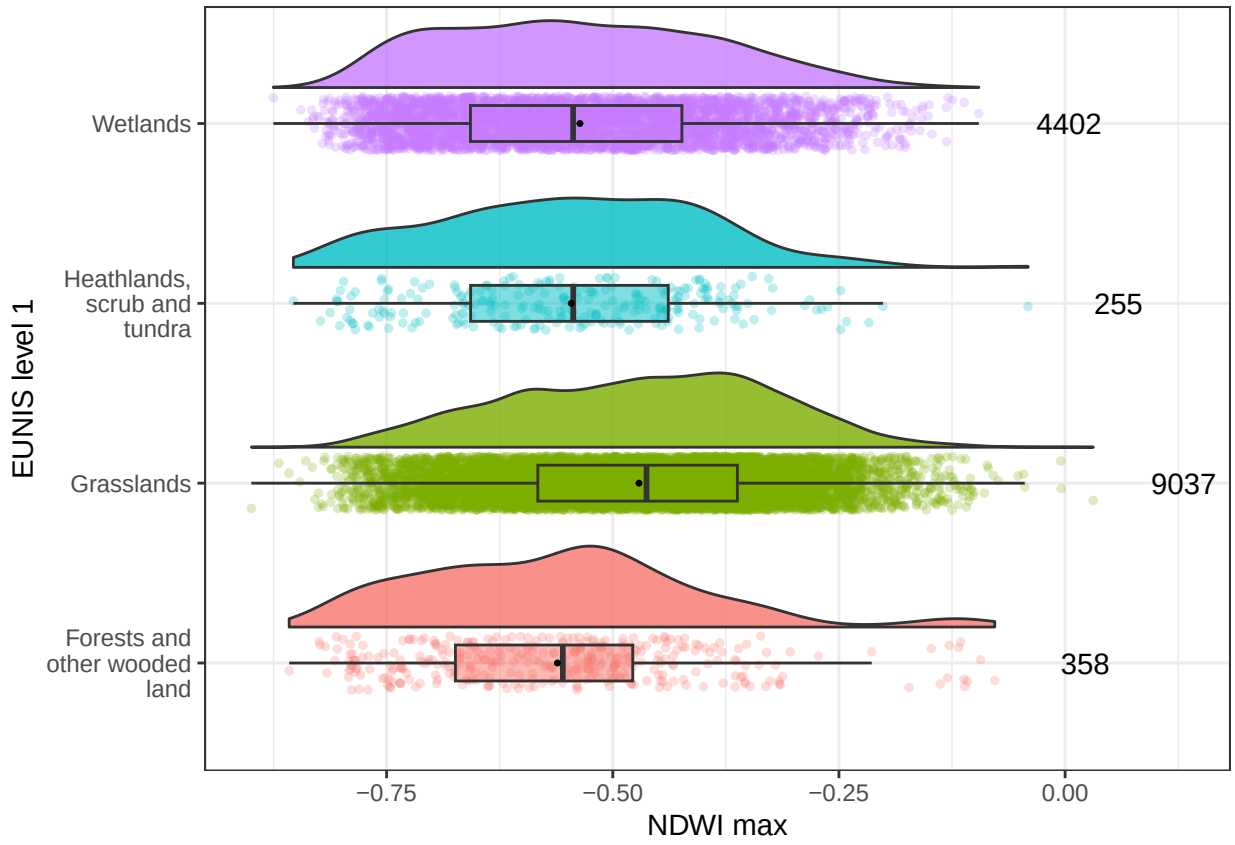
```
plot_distr_EVI_max_all_GPS_valid <- distr_plot(all_GPS_valid %>%
                                                 filter(EVI_max <= 1),
                                               "EVI_max", "EVI max")
plot_distr_NDVI_max_all_GPS_valid
```



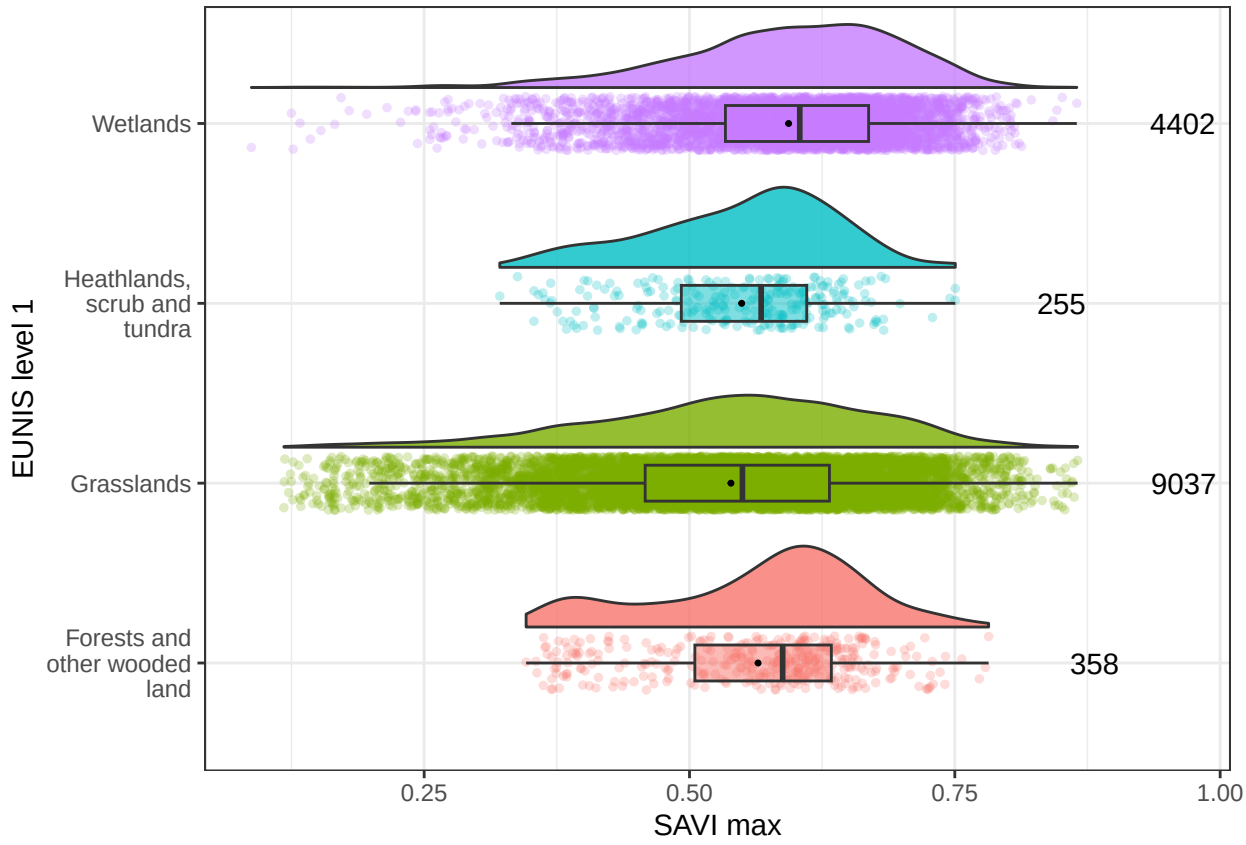

```
plot_distr_NDMI_max_all_GPS_valid
```



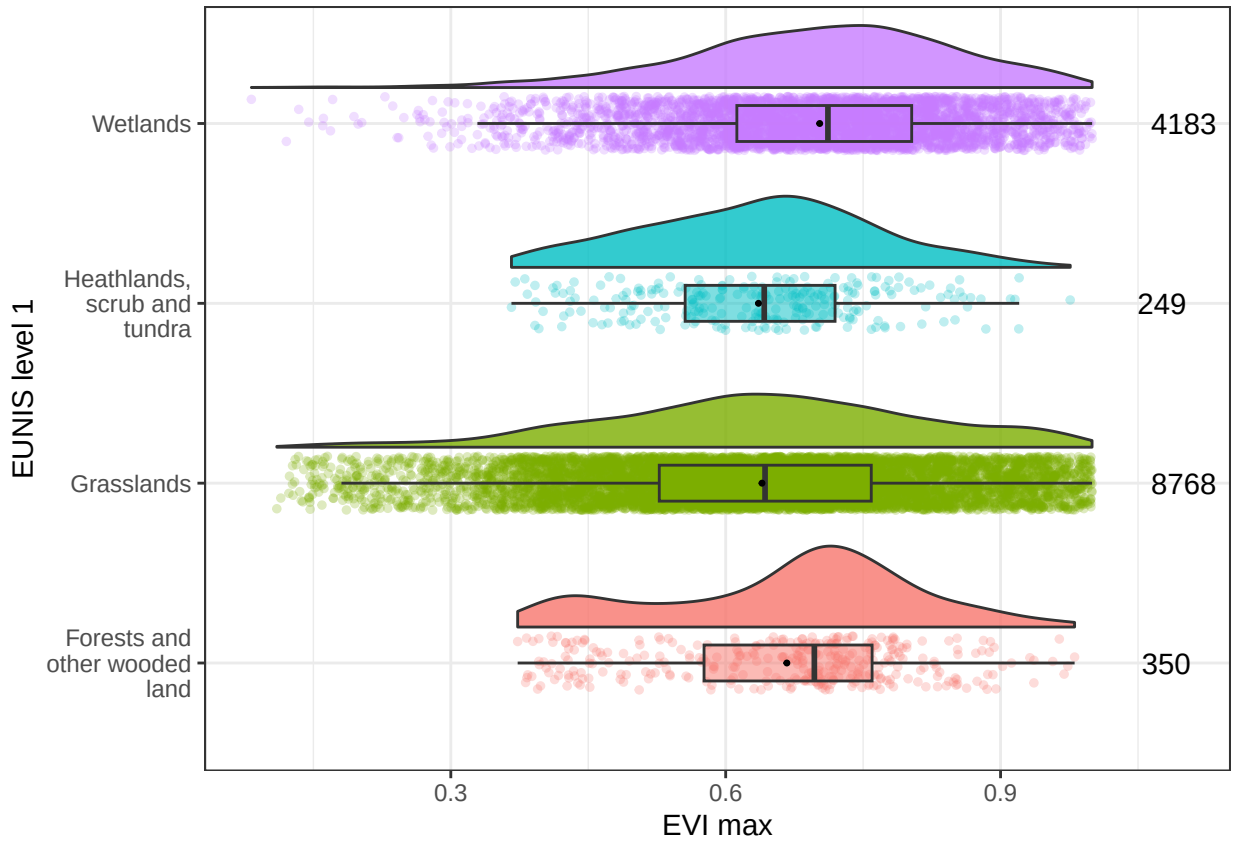

```
plot_distr_NDWI_max_all_GPS_valid
```



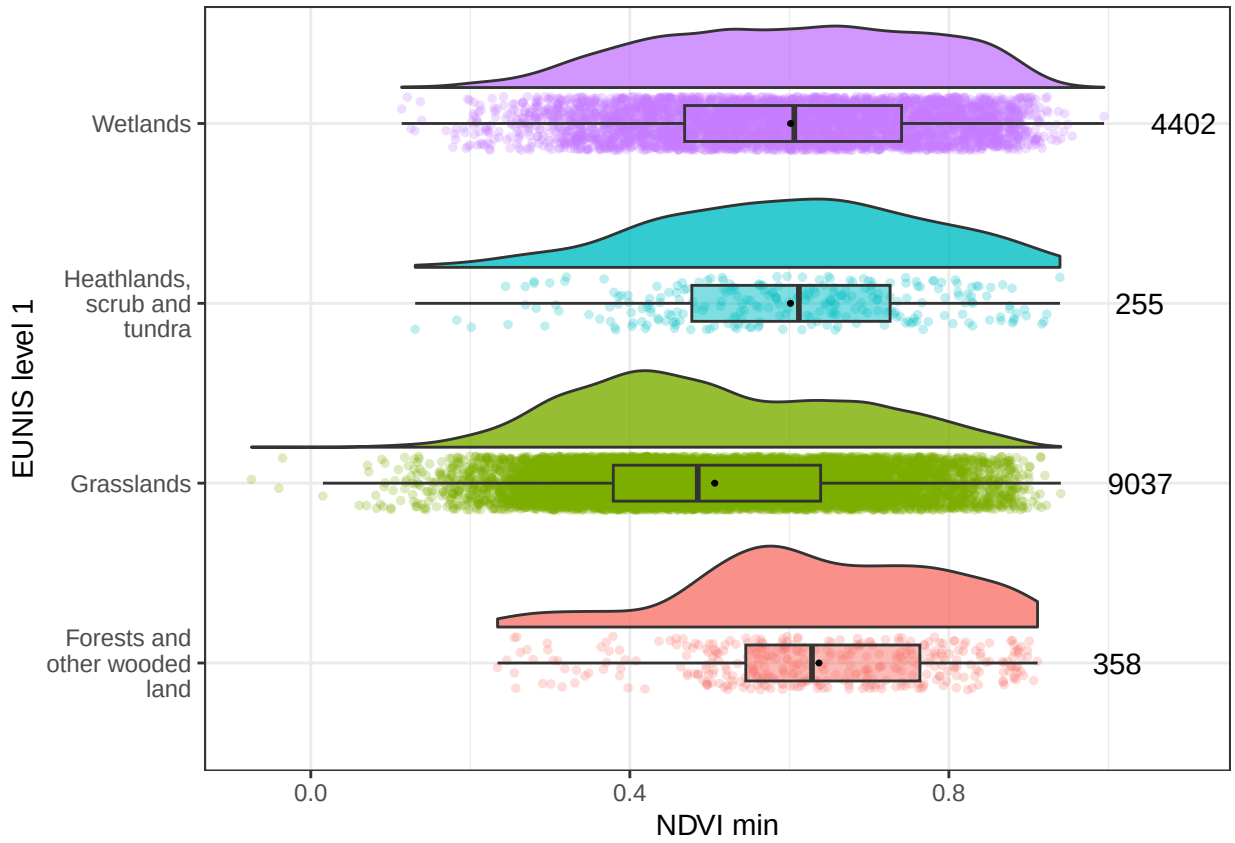

```
plot_distr_SAVI_max_all_GPS_valid
```



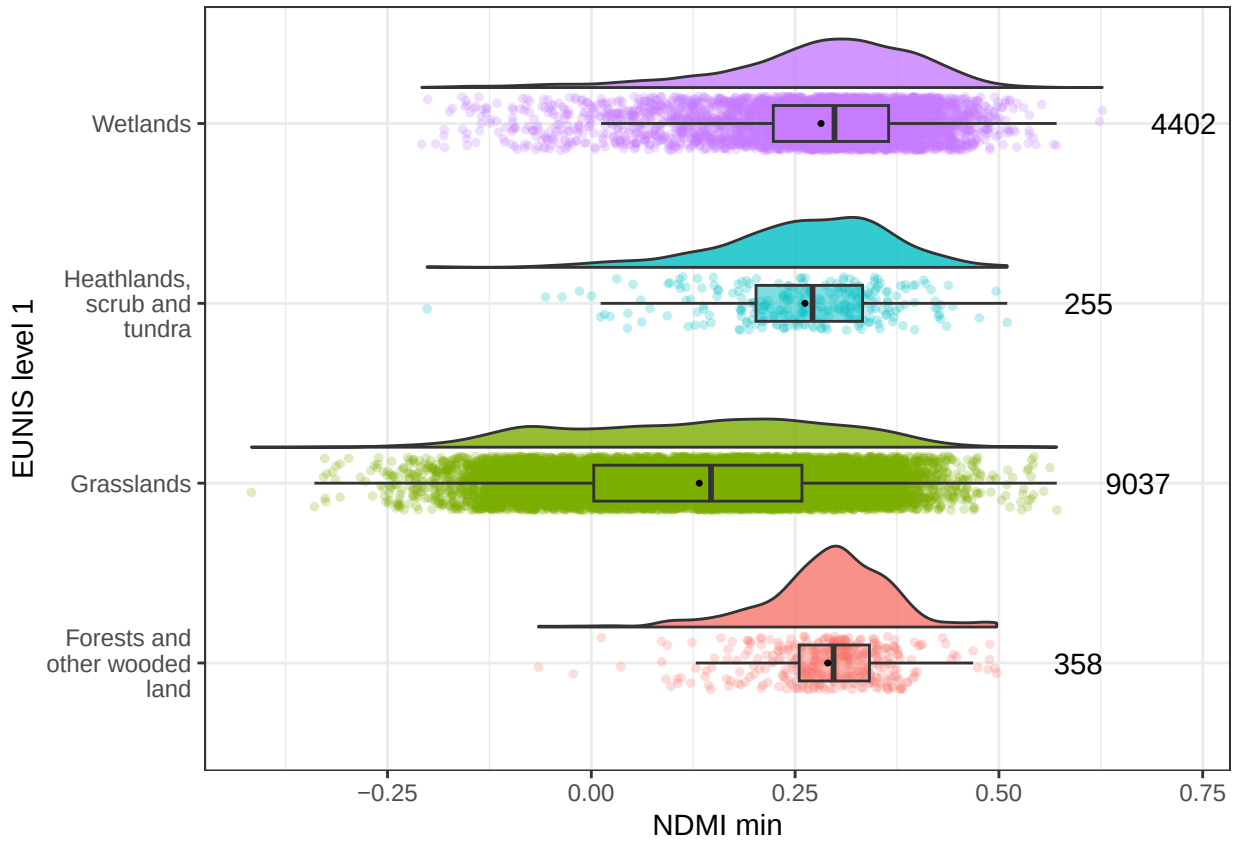

```
plot_distr_EVI_max_all_GPS_valid
```



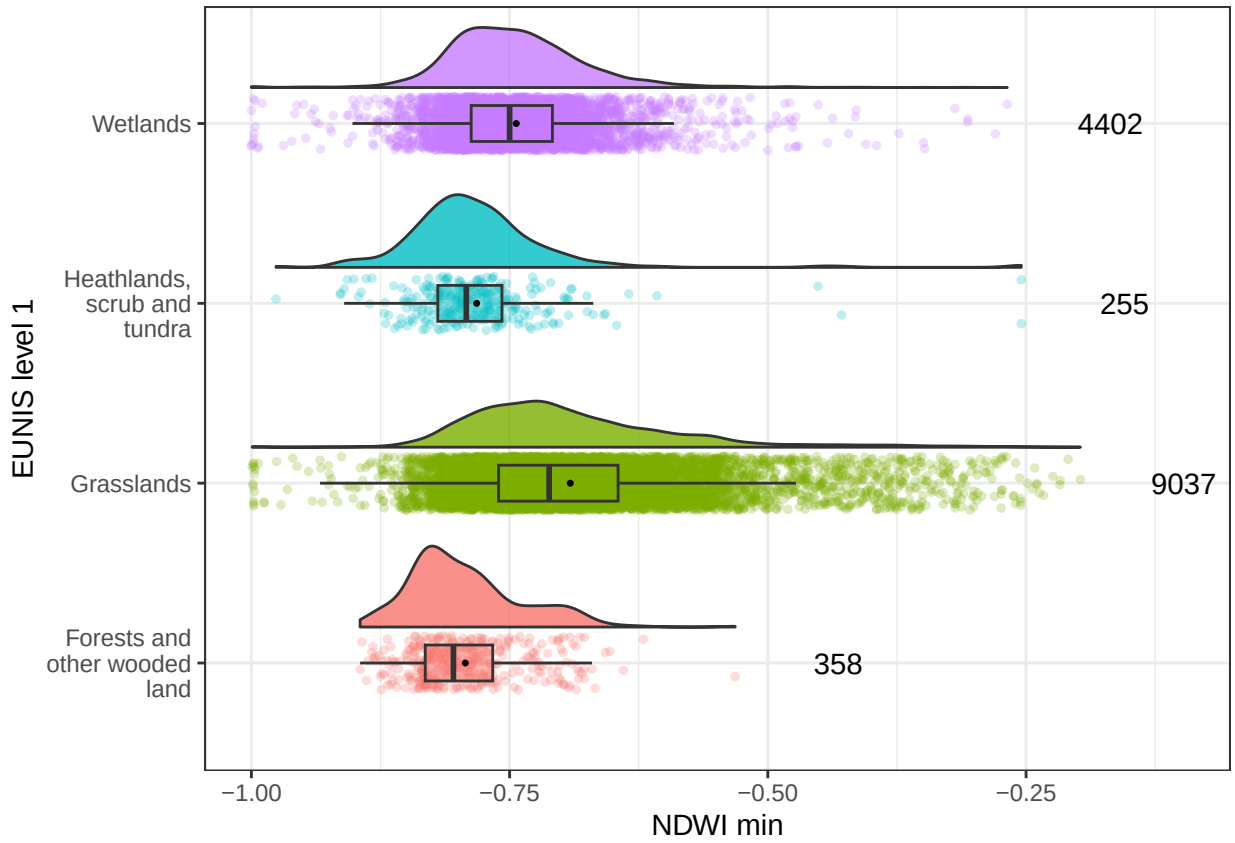

```
plot_distr_NDVI_min_all_GPS_valid <- distr_plot(all_GPS_valid,
                                                "NDVI_min", "NDVI min")
plot_distr_NDMI_min_all_GPS_valid <- distr_plot(all_GPS_valid,
                                                "NDMI_min", "NDMI min")
plot_distr_NDWI_min_all_GPS_valid <- distr_plot(all_GPS_valid,
                                                "NDWI_min", "NDWI min")
plot_distr_SAVI_min_all_GPS_valid <- distr_plot(all_GPS_valid,
                                                "SAVI_min", "SAVI min")
plot_distr_EVI_min_all_GPS_valid <- distr_plot(all_GPS_valid %>%
                                                  # Some values wrong!
                                                  # EVI should not be lower than -1
                                                  filter(EVI_min >= -1),
                                                "EVI_min", "EVI min")
plot_distr_NDVI_min_all_GPS_valid
```



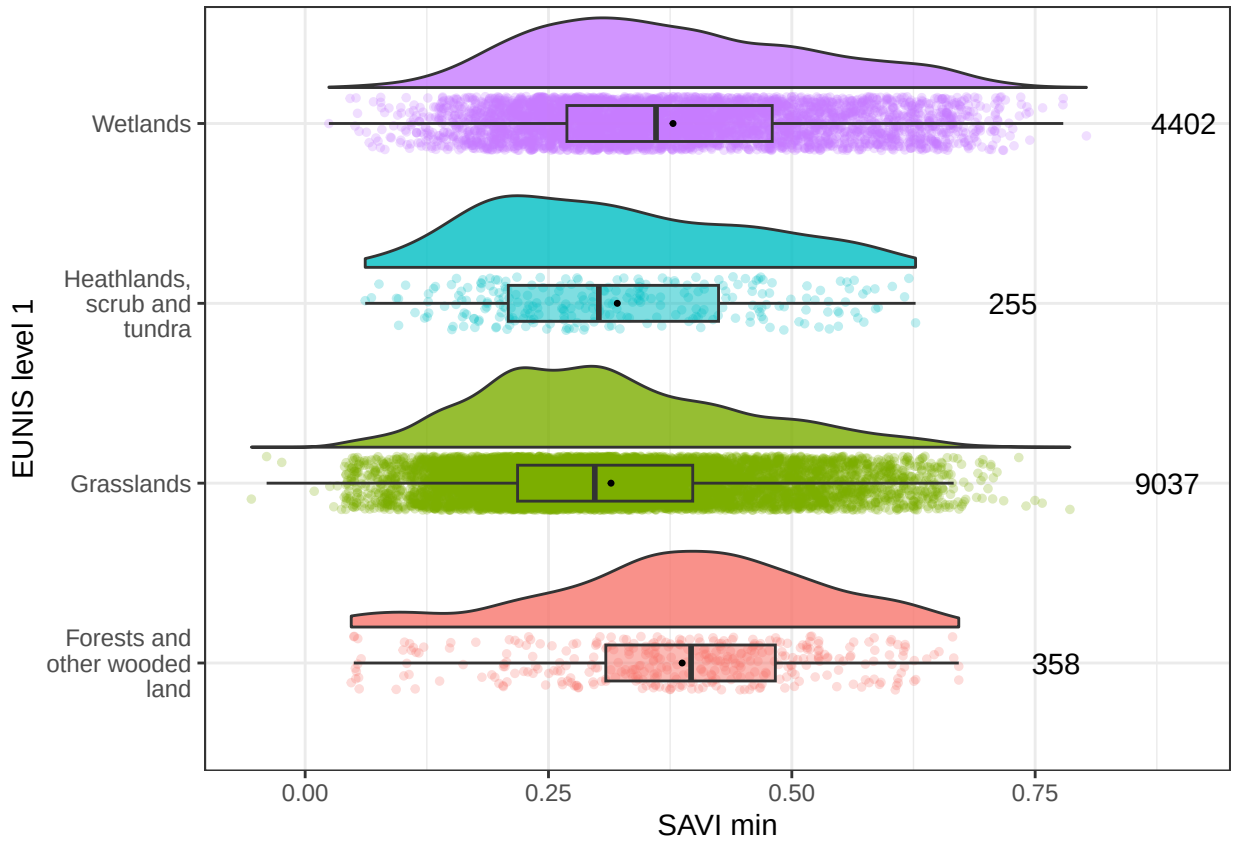

```
plot_distr_NDMI_min_all_GPS_valid
```



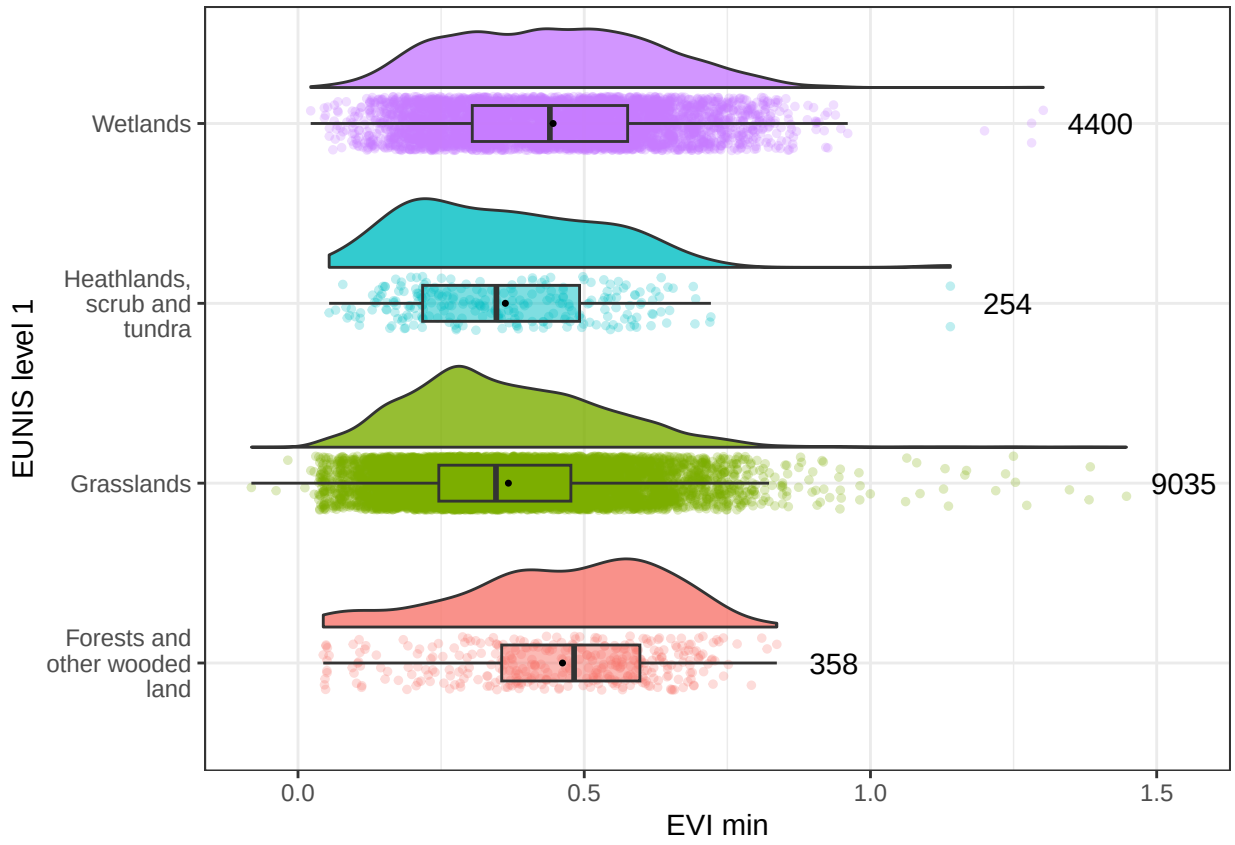

```
plot_distr_NDWI_min_all_GPS_valid
```




```
plot_distr_SAVI_min_all_GPS_valid
```

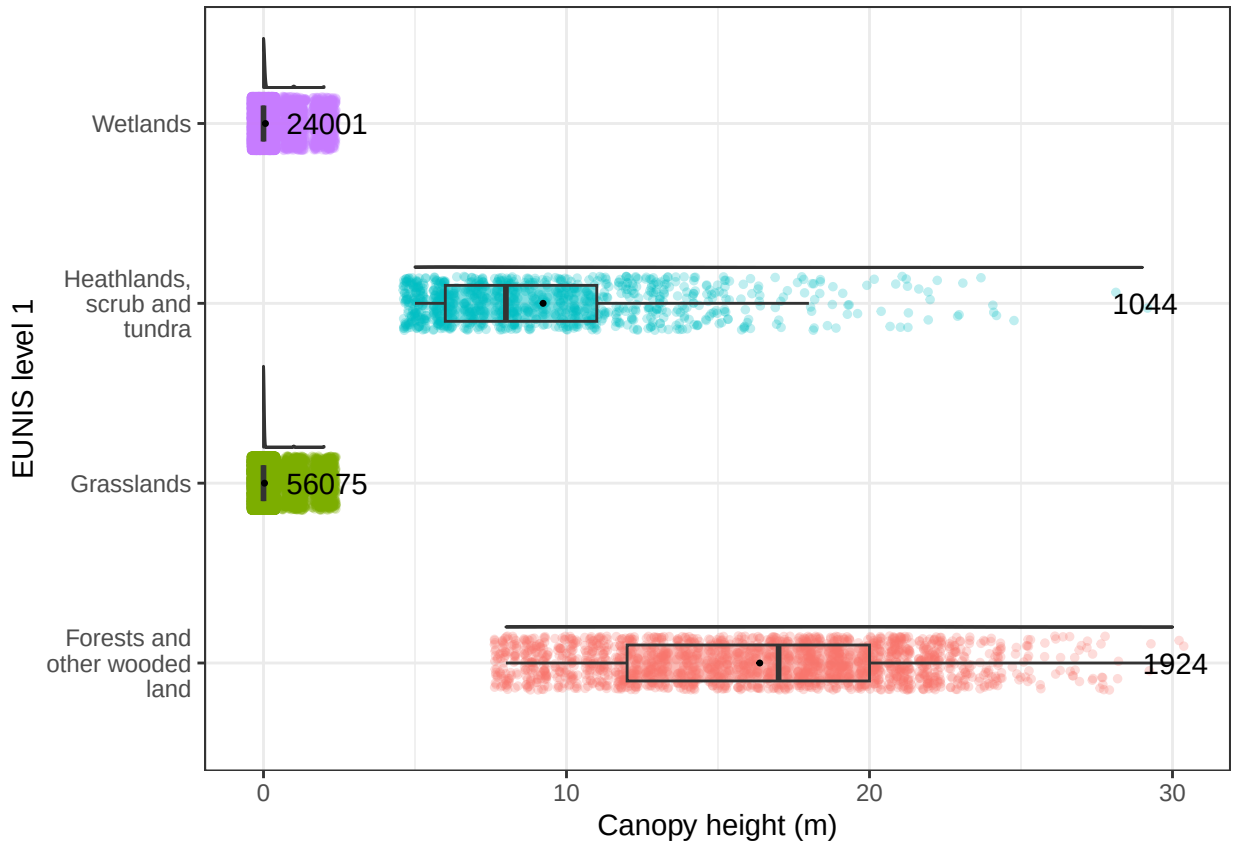



```
plot_distr_EVI_min_all_GPS_valid
```



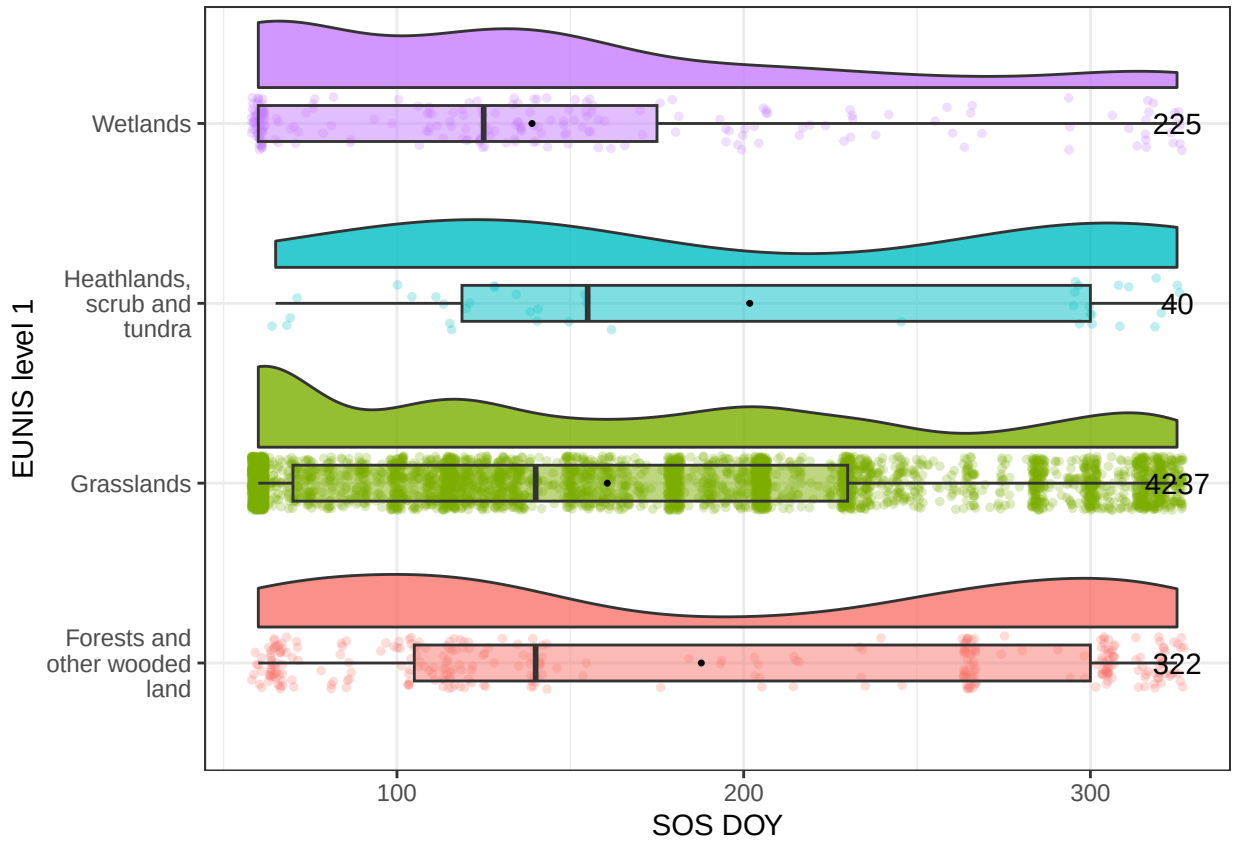
CH

```
plot_distr_CH_all_GPS_valid <- distr_plot(all_GPS_valid, "canopy_height",
                                          "Canopy height (m)")
plot_distr_CH_all_GPS_valid
```

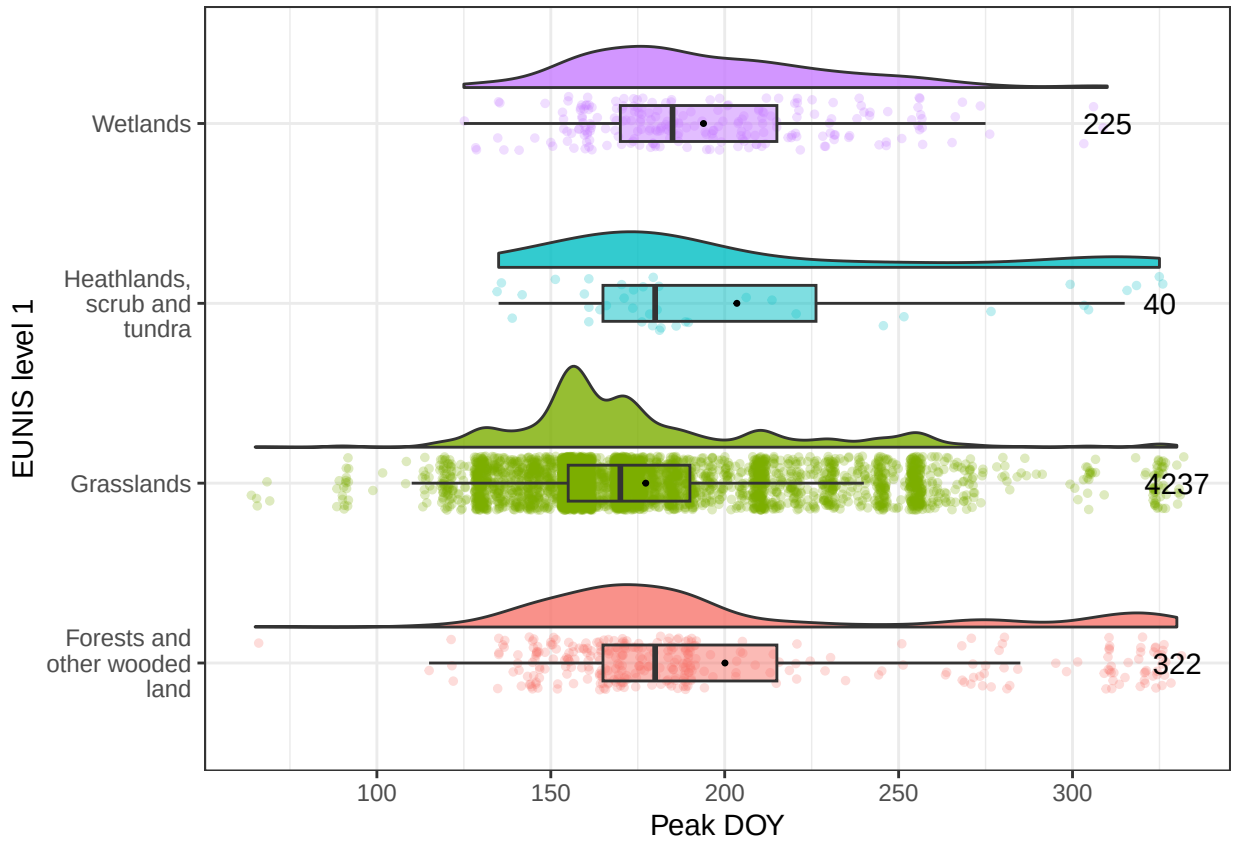


Phenology

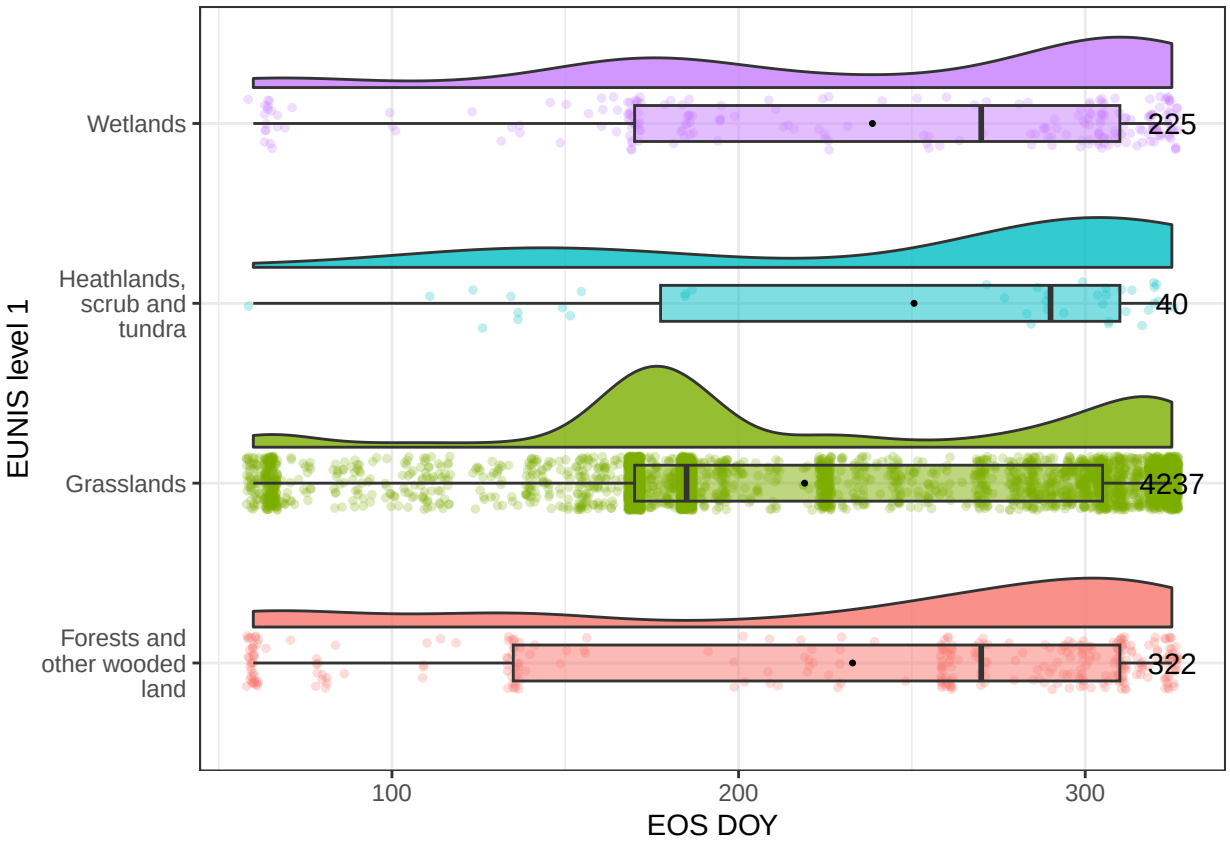
```
plot_distr_SOS_DOY_all_GPS_valid <- distr_plot(all_GPS_valid,
                                               "SOS_DOY", "SOS DOY")
plot_distr_Peak_DOY_all_GPS_valid <- distr_plot(all_GPS_valid,
                                                "Peak_DOY", "Peak DOY")
plot_distr_EOS_DOY_all_GPS_valid <- distr_plot(all_GPS_valid,
                                               "EOS_DOY", "EOS DOY")
plot_distr_SOS_DOY_all_GPS_valid
```




```
plot_distr_Peak_DOY_all_GPS_valid
```




```
plot_distr_EOS_DOY_all_GPS_valid
```


EUNIS level 1
Wetlands
Heathlands, scrub and tundra
Grasslands
Forests and other wooded land
225
40
4237
322
100
200
300
EOS DOY