

Analyses & results

GCFR vs SWAFR manuscript

Ruan van Mazijk

2019-08-28

```
# General programming
library(tidyverse)
library(here) # for more reliable file paths
library(glue) # better than paste()
library(magrittr) # for %<>% & %$$

# GIS
library(raster)
library(rgdal)

# Analyses
library(canprot) # for CLES
library(broom) # to tidy model outputs

# Figures
library(cowplot) # for panelling
library(ggfortify) # for autoplot() of PCAs

# Environmental variable names in nice order
var_names <- c(
  "Elevation",
  "MAP",
  "PDQ",
  "Surface T",
  "NDVI",
  "CEC",
  "Clay",
  "Soil C",
  "pH"
)

# Preserve clean plotting environment
op <- par()

# Global ggplot2 theme settings
theme_set(theme_bw() + theme(
  strip.background = element_blank(),
  panel.grid = element_blank()
))
```

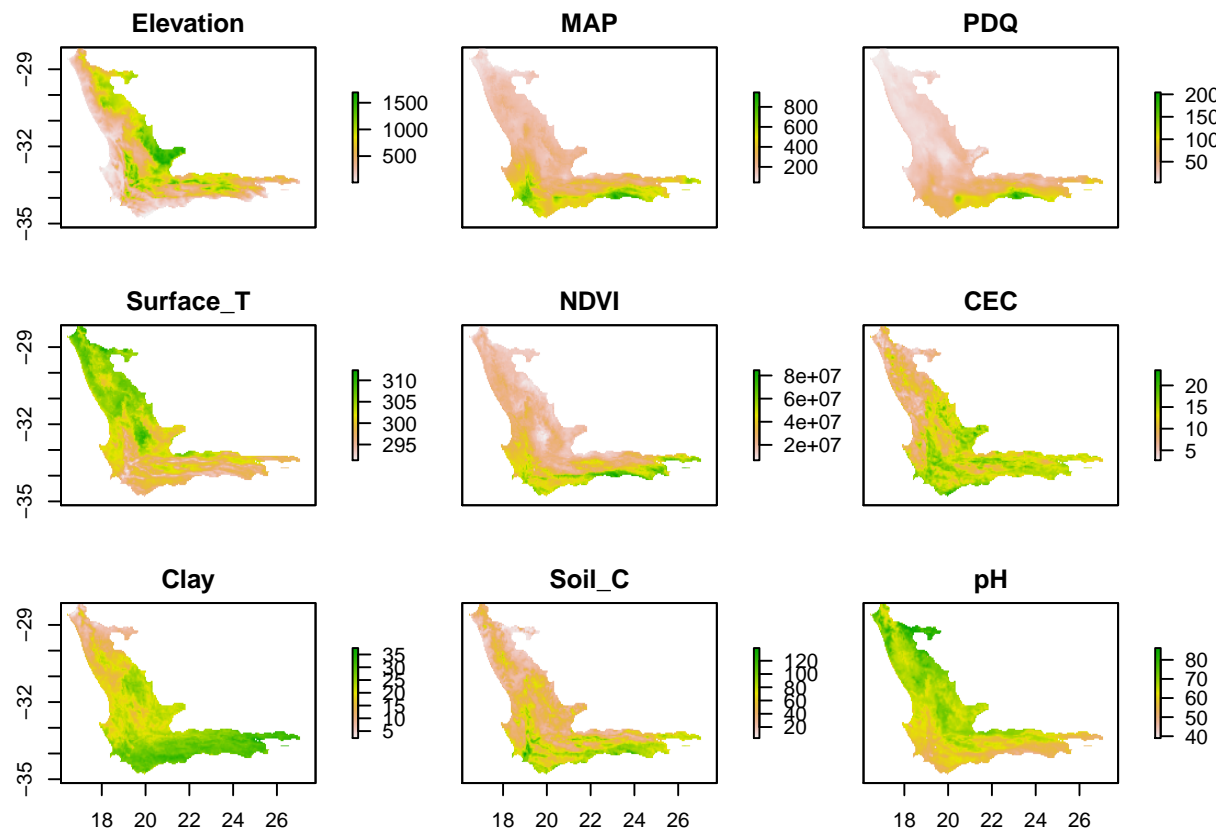
1. Comparing environmental heterogeneity

```
data_dir <- here("data/derived-data/May-2019")
GCFR_file_names <- glue("{data_dir}/GCFR_{var_names}_masked2.tif")
SWAFR_file_names <- glue("{data_dir}/SWAFR_{var_names}_masked2.tif")
GCFR_variables <- stack(GCFR_file_names)
SWAFR_variables <- stack(SWAFR_file_names)
names(GCFR_variables) <- str_replace_all(var_names, " ", "_")
```

```
names(SWAFR_variables) <- str_replace_all(var_names, " ", "_")
GCFR_variables
```

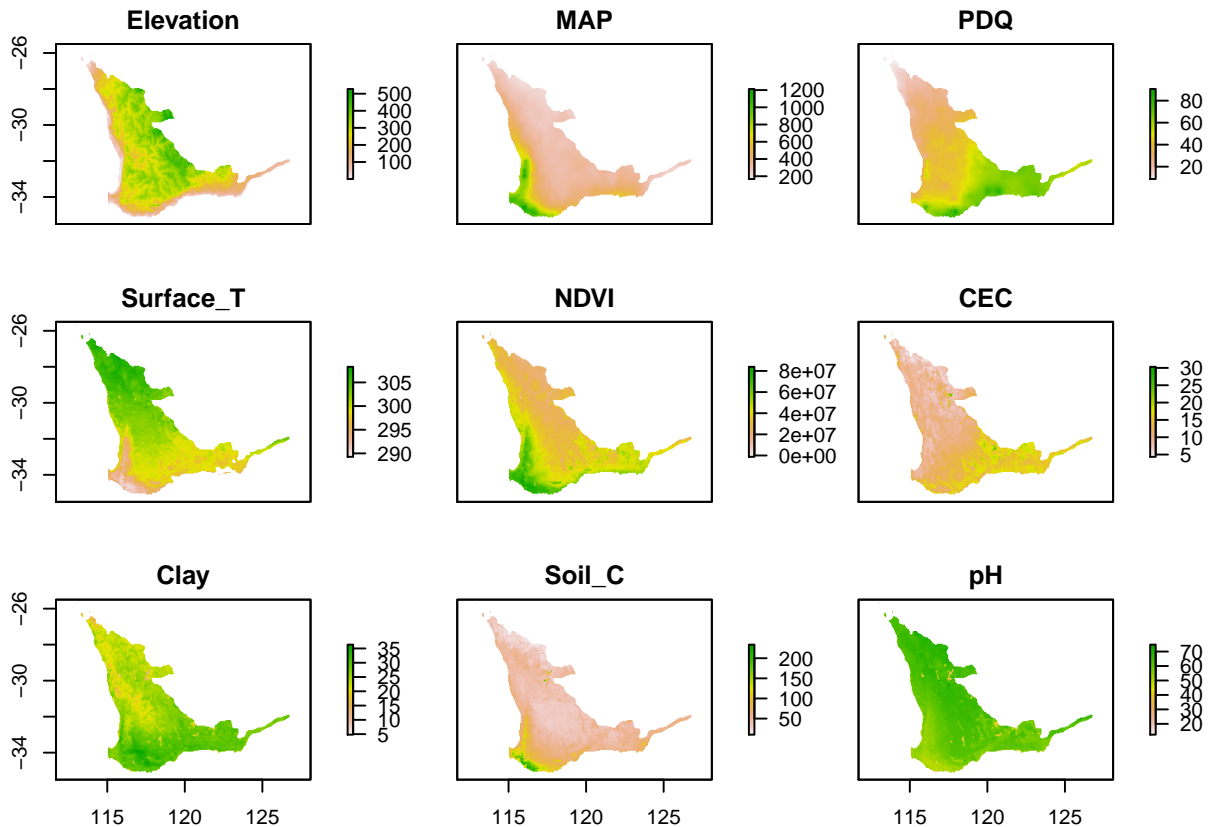
```
## class      : RasterStack
## dimensions  : 140, 220, 30800, 9  (nrow, ncol, ncell, nlayers)
## resolution  : 0.05, 0.05  (x, y)
## extent     : 16.45, 27.45, -35.15, -28.15  (xmin, xmax, ymin, ymax)
## crs        : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
## names      : Elevation, MAP, PDQ, Surface_T, NDVI, CEC,
## min values  : 6.366550e+00, 4.490771e+01, 3.845532e+00, 2.913266e+02, 6.709096e+06, 2.699161e+00,
## max values  : 1.693627e+03, 9.445777e+02, 2.042740e+02, 3.123608e+02, 8.453450e+07, 2.350940e+01,
SWAFR_variables
```

```
## class      : RasterStack
## dimensions  : 200, 280, 56000, 9  (nrow, ncol, ncell, nlayers)
## resolution  : 0.05, 0.05  (x, y)
## extent     : 112.9, 126.9, -35.5, -25.5  (xmin, xmax, ymin, ymax)
## crs        : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
## names      : Elevation, MAP, PDQ, Surface_T, NDVI,
## min values  : 4.407049e-01, 1.677904e+02, 8.773942e+00, 2.892730e+02, -1.272565e+06, 4.344898
## max values  : 5.276772e+02, 1.212685e+03, 9.061172e+01, 3.082984e+02, 8.391765e+07, 3.031978
par(mfrow = c(3, 3))
plot(GCFR_variables)
```



```
par(op)

par(mfrow = c(3, 3))
plot(SWAFR_variables)
```



```
par(op)
```

```
EDS_template_raster <- GCFR_variables$Elevation %>%
  aggregate(fact = 5) %>% # aggregate up to QDS
  disaggregate(fact = 2) # disaggregate down to EDS
GCFR_variables_EDS <- resample(
  GCFR_variables, EDS_template_raster,
  method = "bilinear"
)
GCFR_variables_EDS
```

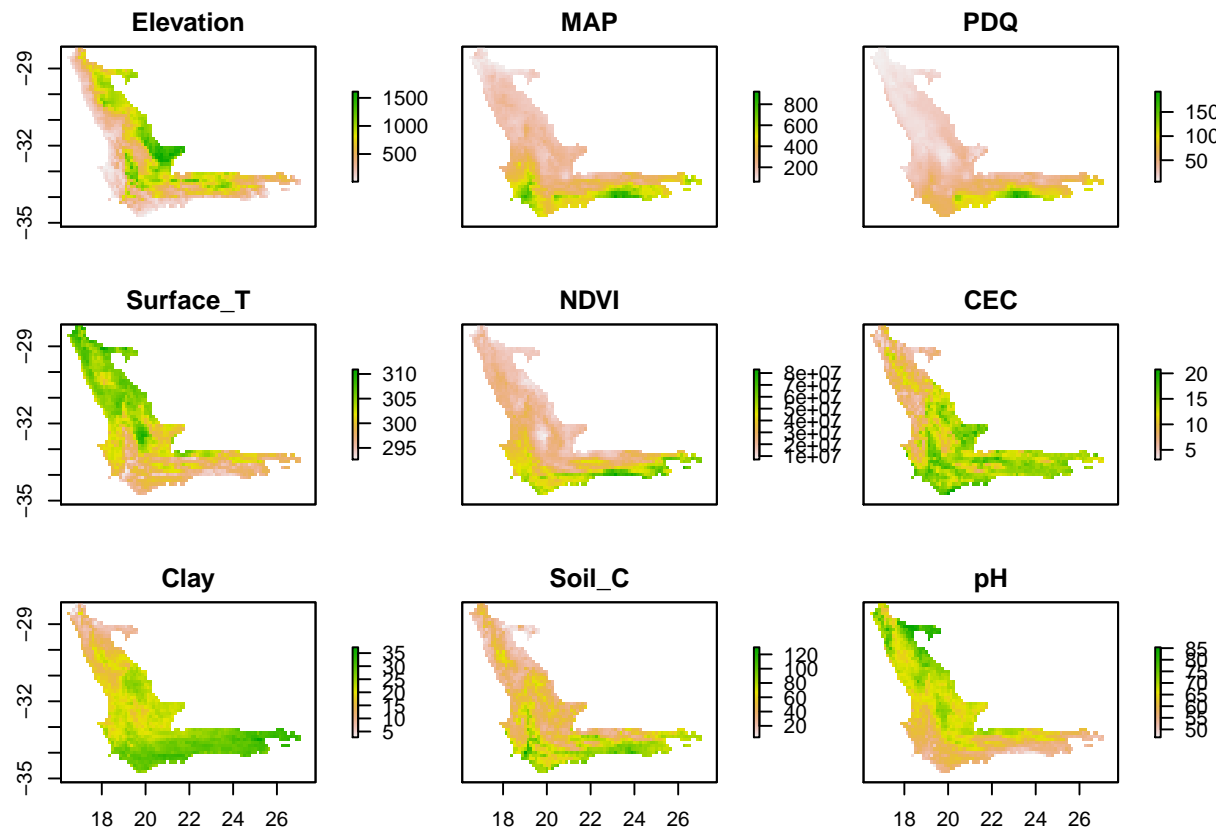
```
## class      : RasterBrick
## dimensions  : 56, 88, 4928, 9 (nrow, ncol, ncell, nlayers)
## resolution  : 0.125, 0.125 (x, y)
## extent     : 16.45, 27.45, -35.15, -28.15 (xmin, xmax, ymin, ymax)
## crs        : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
## source     : memory
## names      : Elevation, MAP, PDQ, Surface_T, NDVI, CEC,
## min values  : 7.740328e+00, 6.333686e+01, 5.568253e+00, 2.926559e+02, 7.119721e+06, 3.109218e+00,
## max values  : 1.611345e+03, 9.199011e+02, 1.921860e+02, 3.109162e+02, 8.301637e+07, 2.076085e+01,
```

```
EDS_template_raster <- SWAFR_variables$Elevation %>%
  aggregate(fact = 5) %>%
  disaggregate(fact = 2)
SWAFR_variables_EDS <- resample(
  SWAFR_variables, EDS_template_raster,
  method = "bilinear"
)
SWAFR_variables_EDS
```

```
## class      : RasterBrick
## dimensions  : 80, 112, 8960, 9 (nrow, ncol, ncell, nlayers)
## resolution  : 0.125, 0.125 (x, y)
```

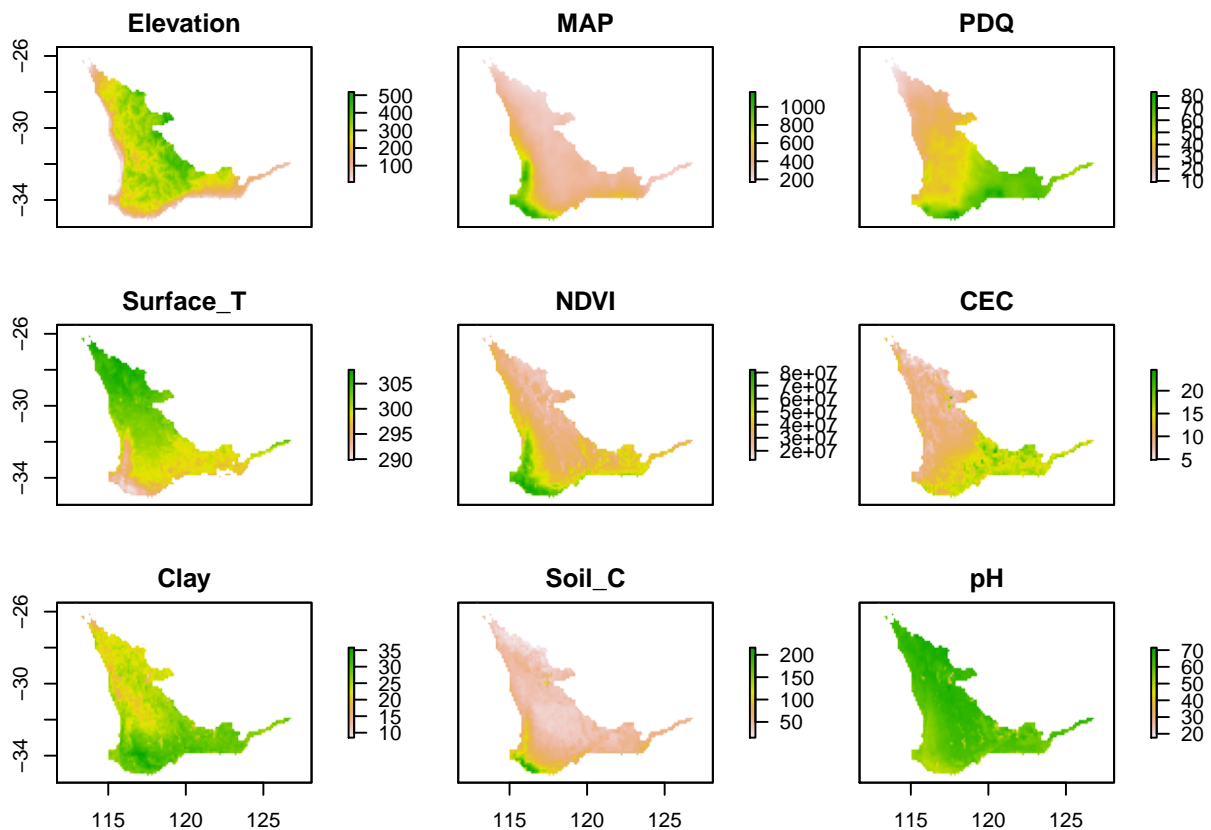
```
## extent      : 112.9, 126.9, -35.5, -25.5 (xmin, xmax, ymin, ymax)
## crs         : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
## source      : memory
## names       : Elevation, MAP, PDQ, Surface_T, NDVI, CEC,
## min values  : 7.149977e+00, 1.708365e+02, 9.094913e+00, 2.898587e+02, 1.308367e+07, 4.849654e+00,
## max values  : 5.180779e+02, 1.163782e+03, 8.326768e+01, 3.077511e+02, 8.209903e+07, 2.458403e+01,
```

```
par(mfrow = c(3, 3))
plot(GCFR_variables_EDS)
```



```
par(op)

par(mfrow = c(3, 3))
plot(SWAFR_variables_EDS)
```



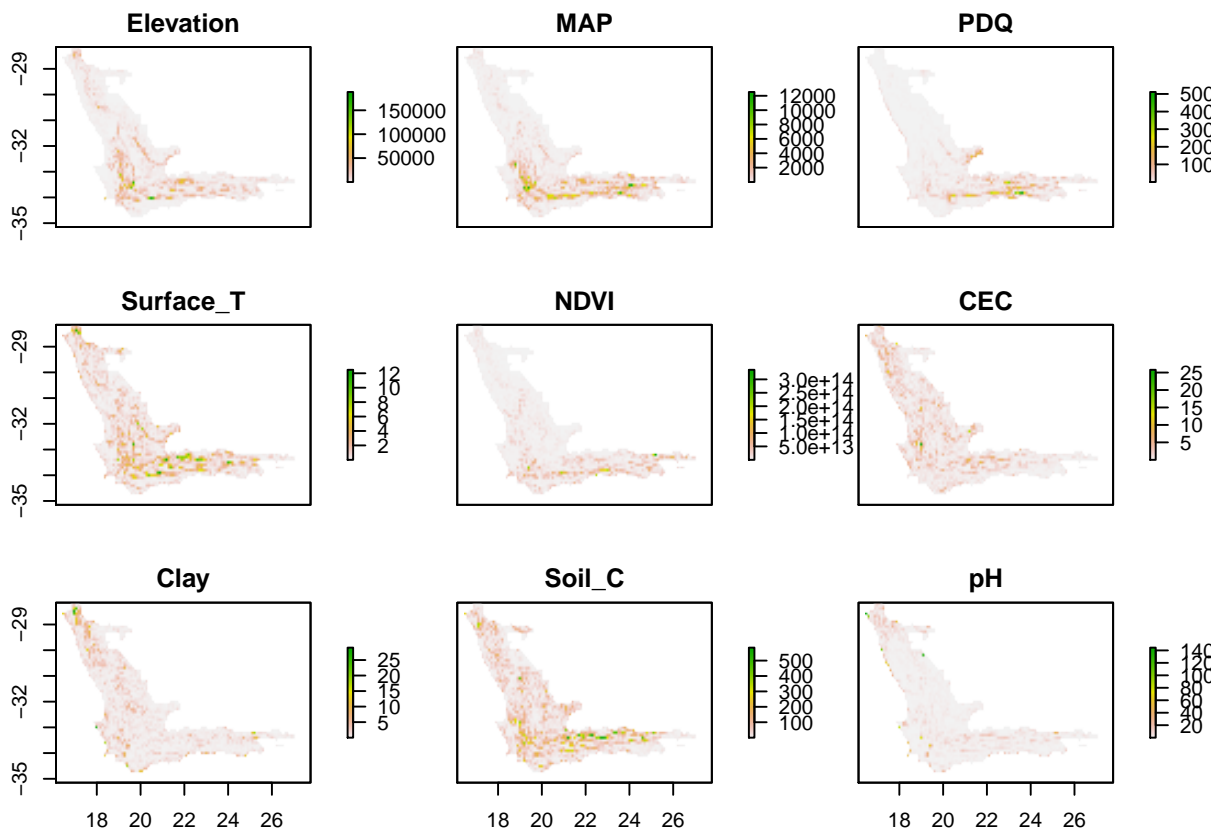
```
par(op)
```

```
scales <- c(1, 2, 4)
```

```
GCFR_heterogeneity <- map(scales,
  ~ GCFR_variables_EDS %>%
    aggregate(fact = .x) %>%
    aggregate(fun = var)
)
names(GCFR_heterogeneity) <- c("QDS", "HDS", "DS")
GCFR_heterogeneity %<>% {c(point1 = aggregate(GCFR_variables, fun = var), .)}
```

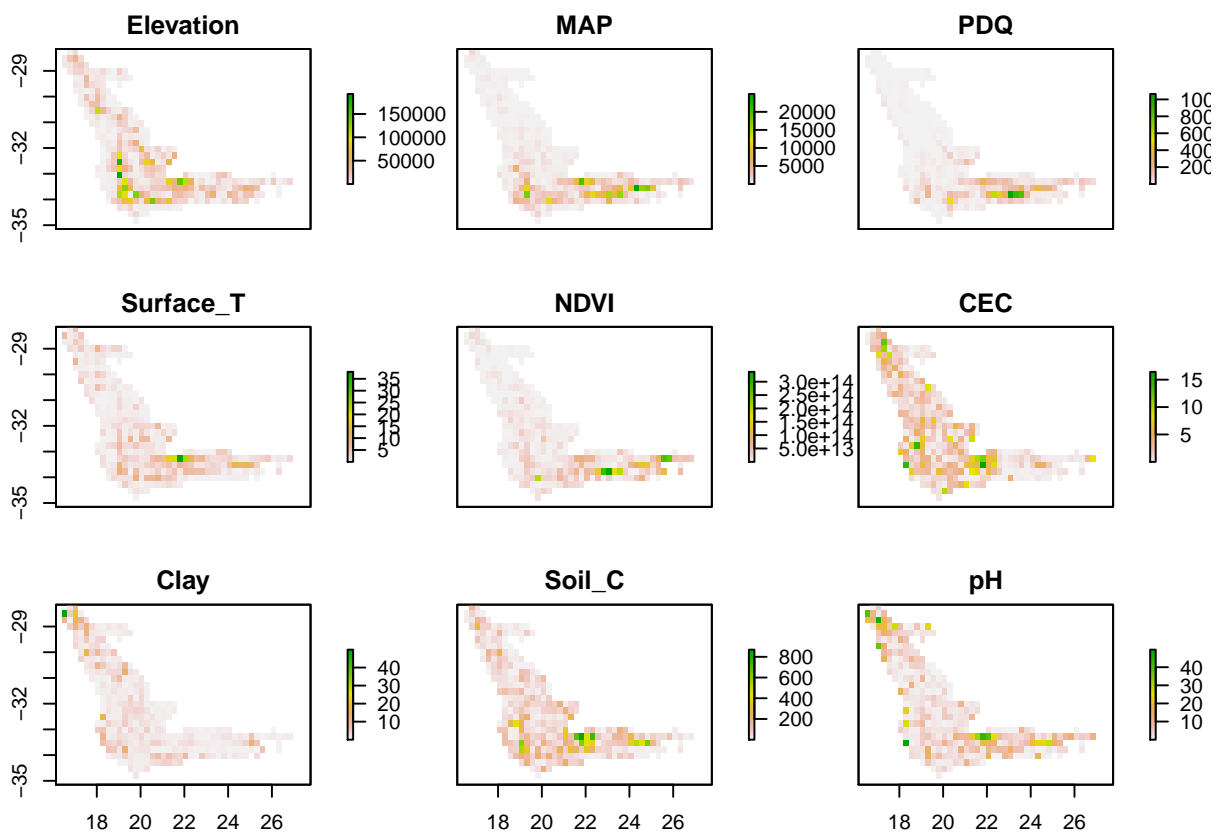
```
SWAFR_heterogeneity <- map(scales,
  ~ SWAFR_variables_EDS %>%
    aggregate(fact = .x) %>%
    aggregate(fun = var)
)
names(SWAFR_heterogeneity) <- c("QDS", "HDS", "DS")
SWAFR_heterogeneity %<>% {c(point1 = aggregate(SWAFR_variables, fun = var), .)}
```

```
par(mfrow = c(3, 3))
plot(GCFR_heterogeneity$point1)
```



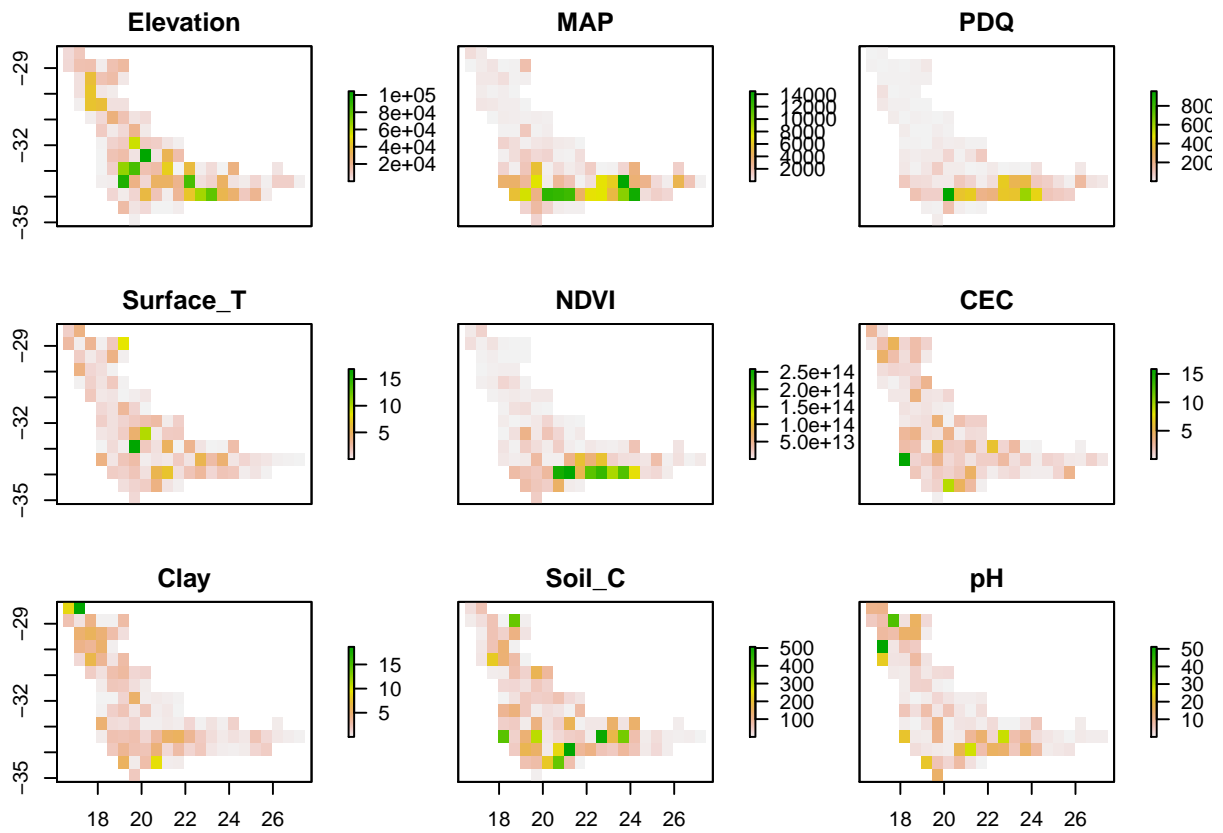
```
par(op)

par(mfrow = c(3, 3))
plot(GCFR_heterogeneity$QDS)
```



```
par(op)
```

```
par(mfrow = c(3, 3))
plot(GCFR_heterogeneity$HDS)
```



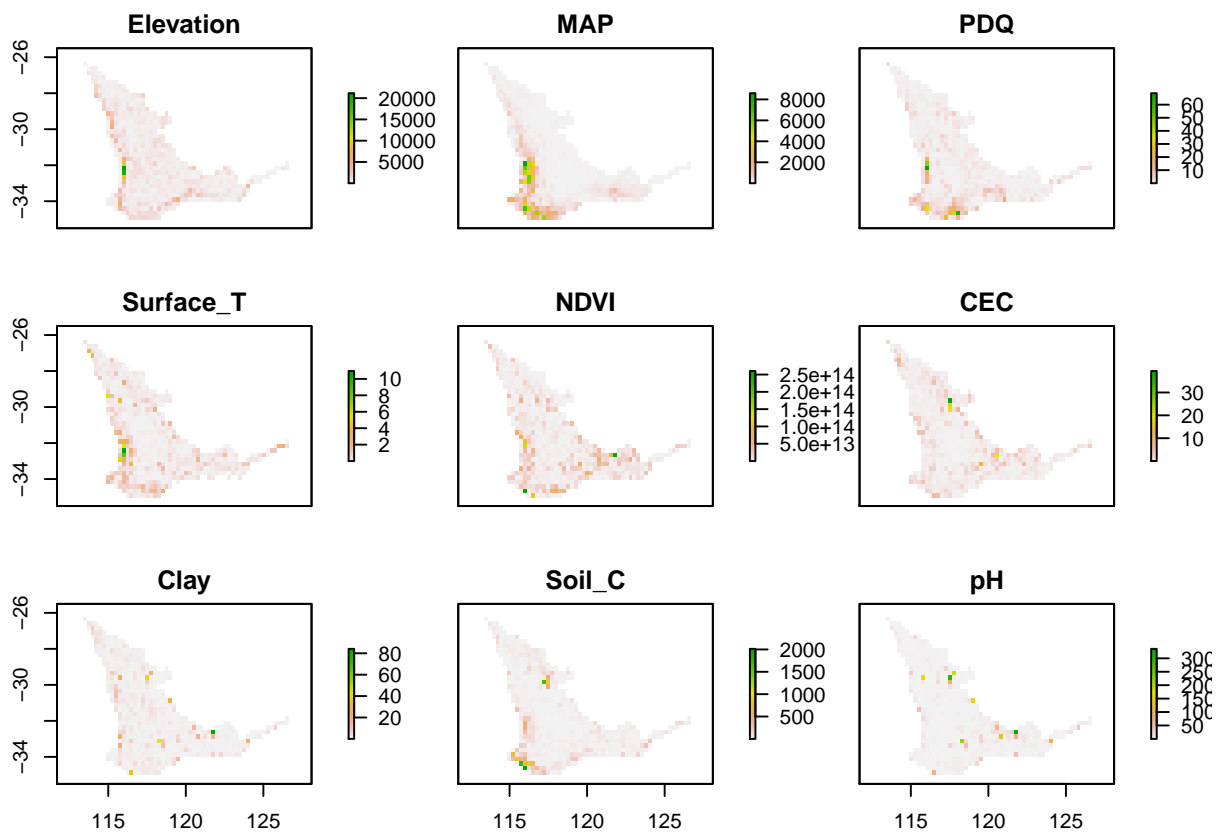
```
par(op)
```

```
par(mfrow = c(3, 3))
plot(GCFR_heterogeneity$DS)
```



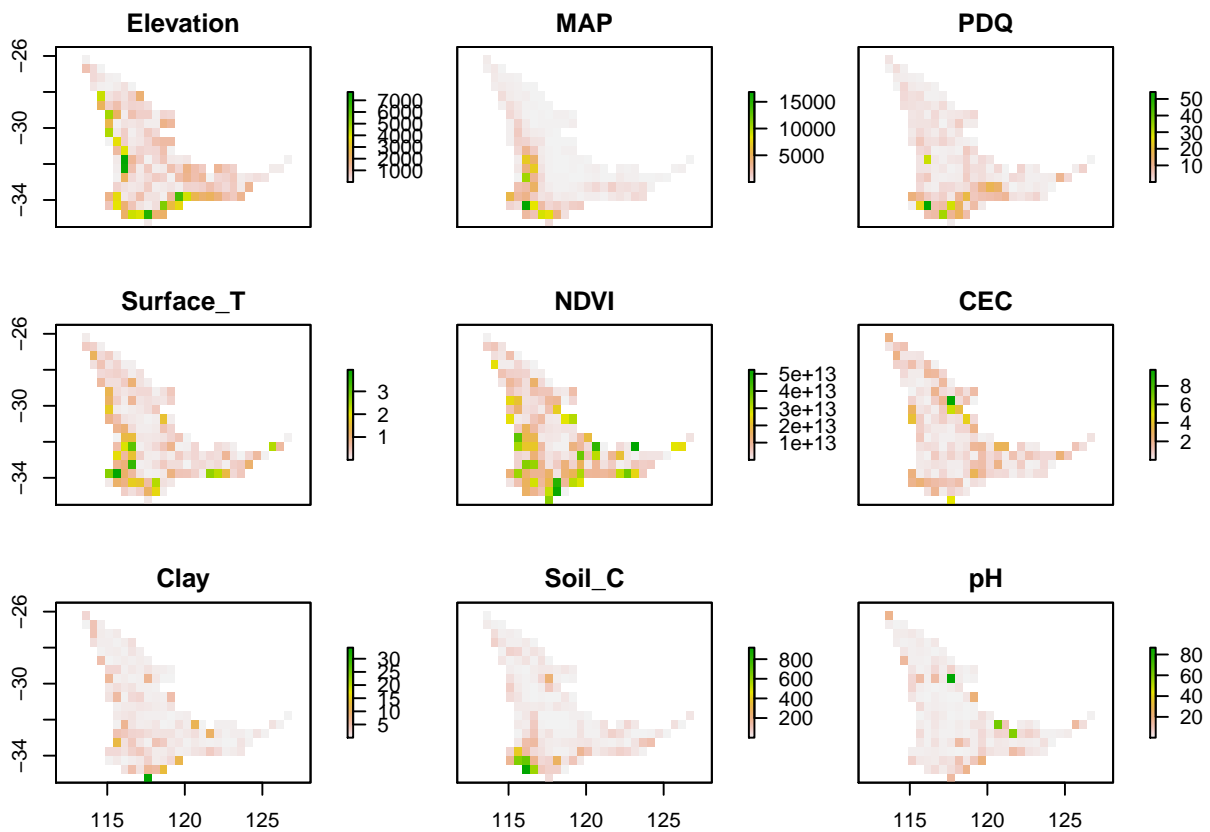
```
par(op)

par(mfrow = c(3, 3))
plot(SWAFR_heterogeneity$QDS)
```



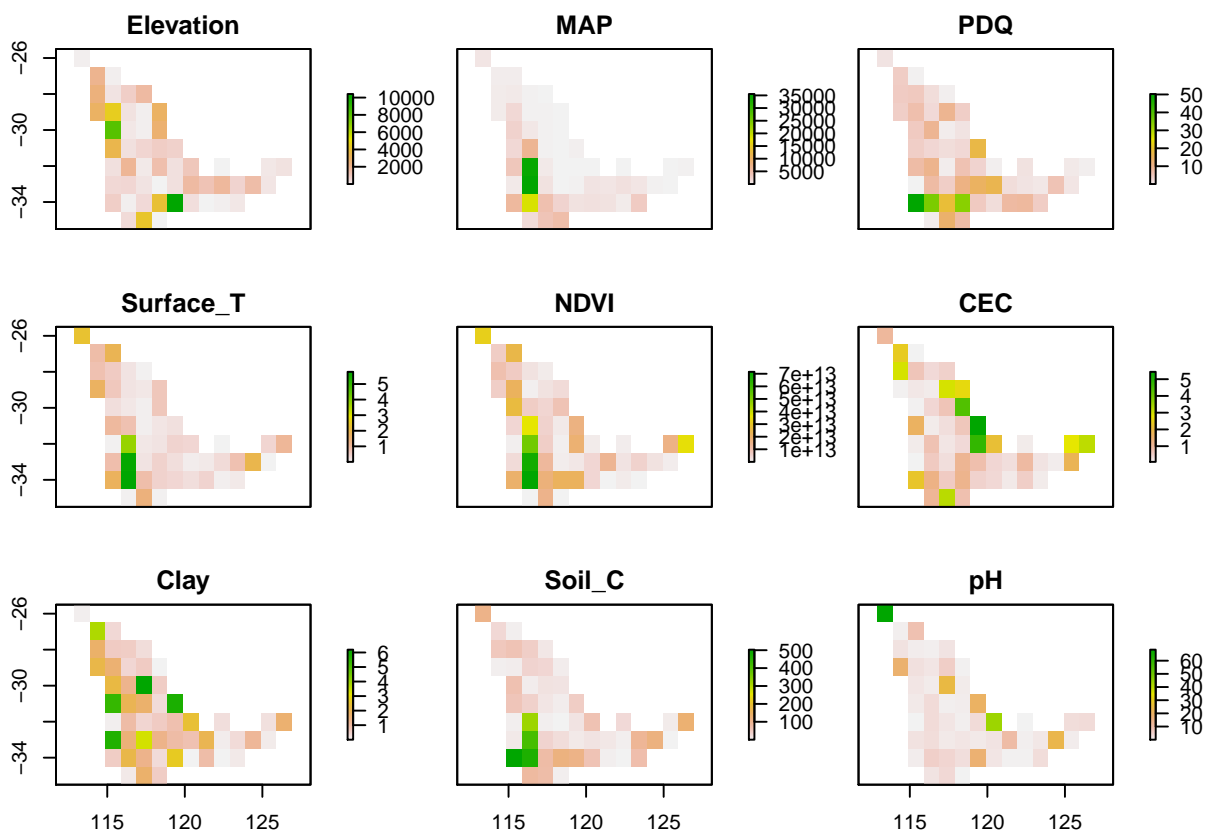
```
par(op)

par(mfrow = c(3, 3))
plot(SWAFR_heterogeneity$HDS)
```



```
par(op)

par(mfrow = c(3, 3))
plot(SWAFR_heterogeneity$DS)
```



```

par(op)

# Join regions' datasets
heterogeneity <- map2(GCFR_heterogeneity, SWAFR_heterogeneity,
  ~ na.exclude(rbind(
    cbind(region = "GCFR", as.data.frame(log10(.x))),
    cbind(region = "SWAFR", as.data.frame(log10(.y)))
  ))
)

heterogeneity_PCAs <- map(heterogeneity,
  ~ prcomp(
    .x[, -1],
    center = TRUE,
    scale. = TRUE
  )
)
map(heterogeneity_PCAs, summary)

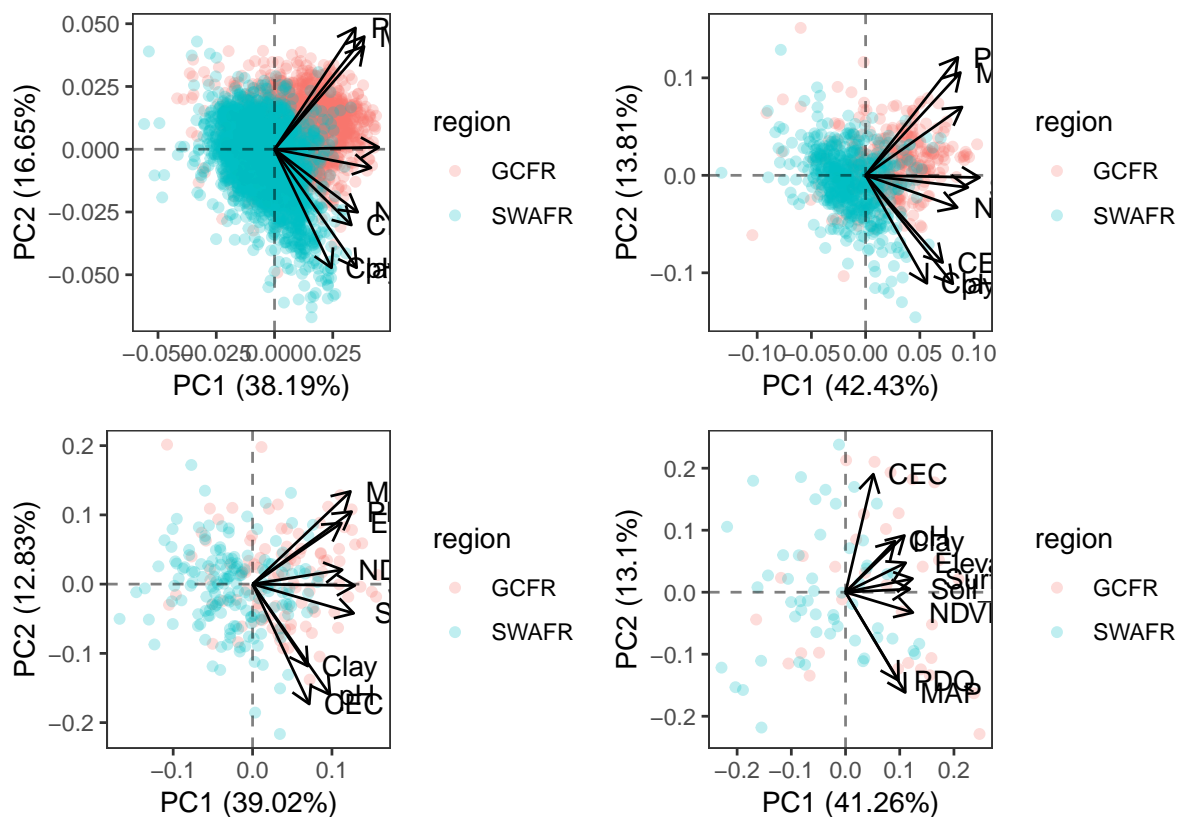
## $point1
## Importance of components:
##              PC1      PC2      PC3      PC4      PC5      PC6
## Standard deviation    1.8539 1.2243 0.89661 0.86925 0.75606 0.7470
## Proportion of Variance 0.3819 0.1665 0.08932 0.08396 0.06351 0.0620
## Cumulative Proportion 0.3819 0.5484 0.63775 0.72171 0.78522 0.8472
##              PC7      PC8      PC9
## Standard deviation    0.73289 0.66832 0.62552
## Proportion of Variance 0.05968 0.04963 0.04347
## Cumulative Proportion 0.90690 0.95653 1.00000
##
## $QDS
## Importance of components:
##              PC1      PC2      PC3      PC4      PC5      PC6
## Standard deviation    1.9543 1.1147 0.90436 0.88177 0.80313 0.71216
## Proportion of Variance 0.4244 0.1381 0.09087 0.08639 0.07167 0.05635
## Cumulative Proportion 0.4244 0.5624 0.65327 0.73966 0.81133 0.86768
##              PC7      PC8      PC9
## Standard deviation    0.66146 0.62594 0.60128
## Proportion of Variance 0.04861 0.04353 0.04017
## Cumulative Proportion 0.91630 0.95983 1.00000
##
## $HDS
## Importance of components:
##              PC1      PC2      PC3      PC4      PC5      PC6      PC7
## Standard deviation    1.8740 1.0746 0.9850 0.90862 0.82854 0.76507 0.7099
## Proportion of Variance 0.3902 0.1283 0.1078 0.09173 0.07628 0.06504 0.0560
## Cumulative Proportion 0.3902 0.5185 0.6263 0.71808 0.79435 0.85939 0.9154
##              PC8      PC9
## Standard deviation    0.65867 0.5724
## Proportion of Variance 0.04821 0.0364
## Cumulative Proportion 0.96360 1.0000
##
## $DS
## Importance of components:
##              PC1      PC2      PC3      PC4      PC5      PC6      PC7
## Standard deviation    1.9271 1.0859 0.9999 0.91028 0.79126 0.73452 0.6469
## Proportion of Variance 0.4126 0.1310 0.1111 0.09207 0.06957 0.05995 0.0465
## Cumulative Proportion 0.4126 0.5437 0.6547 0.74681 0.81637 0.87632 0.9228

```

```
##                                PC8      PC9
## Standard deviation      0.62256 0.55410
## Proportion of Variance 0.04306 0.03411
## Cumulative Proportion  0.96589 1.00000

# Force PC1 scores to be positive if all vars rotations are negative
heterogeneity_PCAs %<>% map(function(PCA) {
  if (all(PCA$rotation[, 1] <= 0)) {
    message("Multiplying this one by -1")
    PCA$rotation[, 1] %<>% multiply_by(-1)
    PCA$x[, 1] %<>% multiply_by(-1)
  }
  PCA
})
```

```
plot_grid(plotlist = map2(
  .x = heterogeneity_PCAs,
  .y = heterogeneity,
  .f =
    ~ autoplot(.x, data = .y, colour = "region",
      alpha = 0.25,
      loadings = TRUE, loadings.colour = "black",
      loadings.label = TRUE, loadings.label.colour = "black",
      loadings.label.hjust = -0.25
    ) +
    ggtitle(unique(.y$scale)) +
    geom_hline(yintercept = 0, linetype = "dashed", alpha = 0.5) +
    geom_vline(xintercept = 0, linetype = "dashed", alpha = 0.5)
))
```



```
PC1s <- map(heterogeneity_PCAs, ~tibble(PC1 = .x$x[, 1]))
heterogeneity %<>% map2(PC1s, ~cbind(.x, .y))
```

```

CLES_results <- map2_dfr(
  .x = heterogeneity %>%
    map(filter, region == "GCFR") %>%
    map(dplyr::select, -region),
  .y = heterogeneity %>%
    map(filter, region == "SWAFR") %>%
    map(dplyr::select, -region),
  .id = "scale", # for every spatial scale,
  ~ map2_df(
    .x = .x,
    .y = .y,
    .id = "variable", # for every variable in each region,
    ~ tibble(
      CLES_value = CLES(.y, .x), # calculate the CLES,
      U_test = wilcox.test(.x, .y, conf.int = TRUE) %>% # & Mann-Whitney U-test
        tidy() %>%
        list()
    )
  )
)
CLES_results %<>% mutate(
  variable = factor(variable, levels = var_names %>%
    str_replace_all(" ", "_") %>%
    c("PC1")
  ),
  scale = case_when(
    scale == "point1" ~ 0.10,
    scale == "QDS" ~ 0.25,
    scale == "HDS" ~ 0.50,
    scale == "DS" ~ 1.00
  ),
  diff = map_dbl(U_test, "estimate"),
  P_U = map_dbl(U_test, "p.value"),
  U_low = map_dbl(U_test, "conf.low"),
  U_upp = map_dbl(U_test, "conf.high")
)
CLES_results

```

```

## # A tibble: 40 x 8
##   scale variable CLES_value U_test diff P_U U_low U_upp
##   <dbl> <fct>      <dbl> <list> <dbl> <dbl> <dbl> <dbl>
## 1 0.1 Elevation 0.908 <tibble [1 x 7~ 1.24 0. 1.20 1.27
## 2 0.1 MAP 0.776 <tibble [1 x 7~ 0.927 1.57e-249 0.879 0.976
## 3 0.1 PDQ 0.801 <tibble [1 x 7~ 0.937 2.97e-296 0.891 0.983
## 4 0.1 Surface_T 0.744 <tibble [1 x 7~ 0.648 2.41e-196 0.609 0.687
## 5 0.1 NDVI 0.537 <tibble [1 x 7~ 0.0919 6.81e- 6 0.0520 0.132
## 6 0.1 CEC 0.646 <tibble [1 x 7~ 0.322 4.31e- 71 0.288 0.356
## 7 0.1 Clay 0.570 <tibble [1 x 7~ 0.151 1.02e- 17 0.117 0.185
## 8 0.1 Soil_C 0.693 <tibble [1 x 7~ 0.514 3.24e-123 0.475 0.554
## 9 0.1 pH 0.641 <tibble [1 x 7~ 0.386 2.39e- 66 0.344 0.428
## 10 0.1 PC1 0.819 <tibble [1 x 7~ 2.09 0. 2.00 2.18
## # ... with 30 more rows

```

```

CLES_models <- CLES_results %>%
  split(.$variable) %>%
  map(~lm(CLES_value ~ scale, .x))
CLES_models$Elevation

```

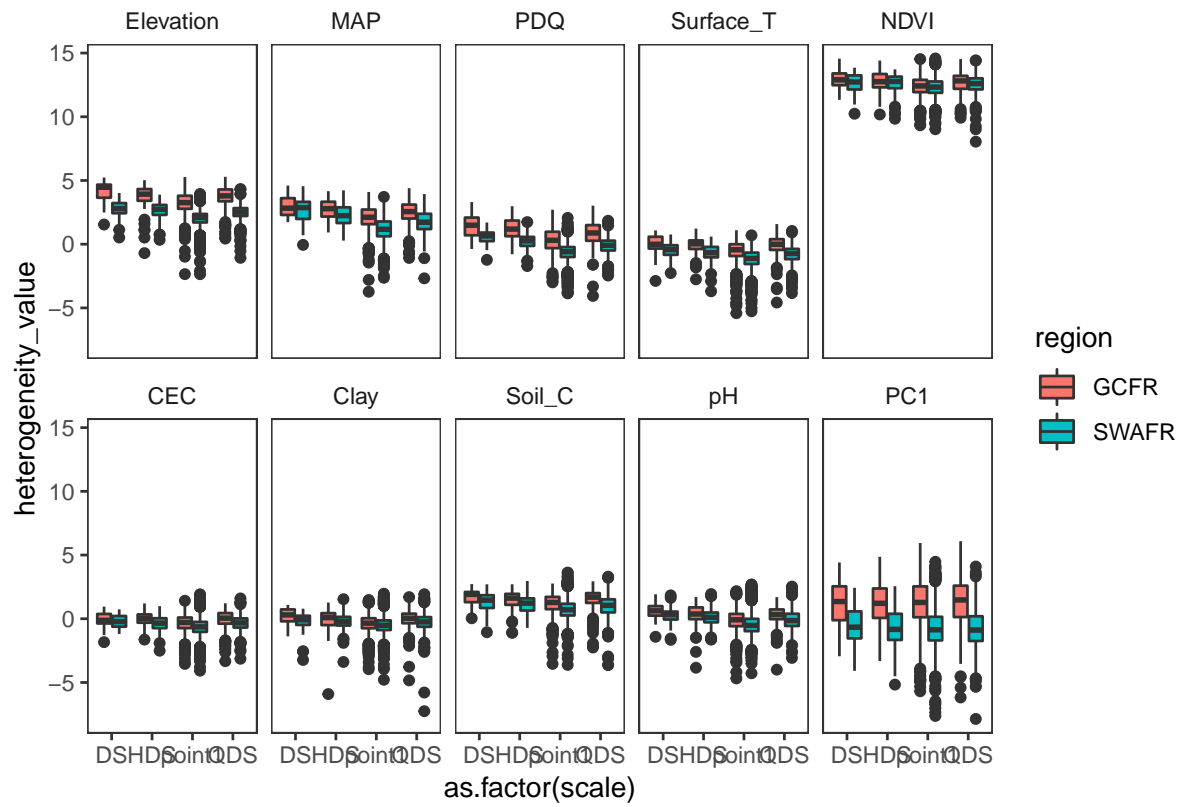
```
##
```

```
## Call:
## lm(formula = CLES_value ~ scale, data = .x)
##
## Coefficients:
## (Intercept)          scale
##      0.90749      -0.02919

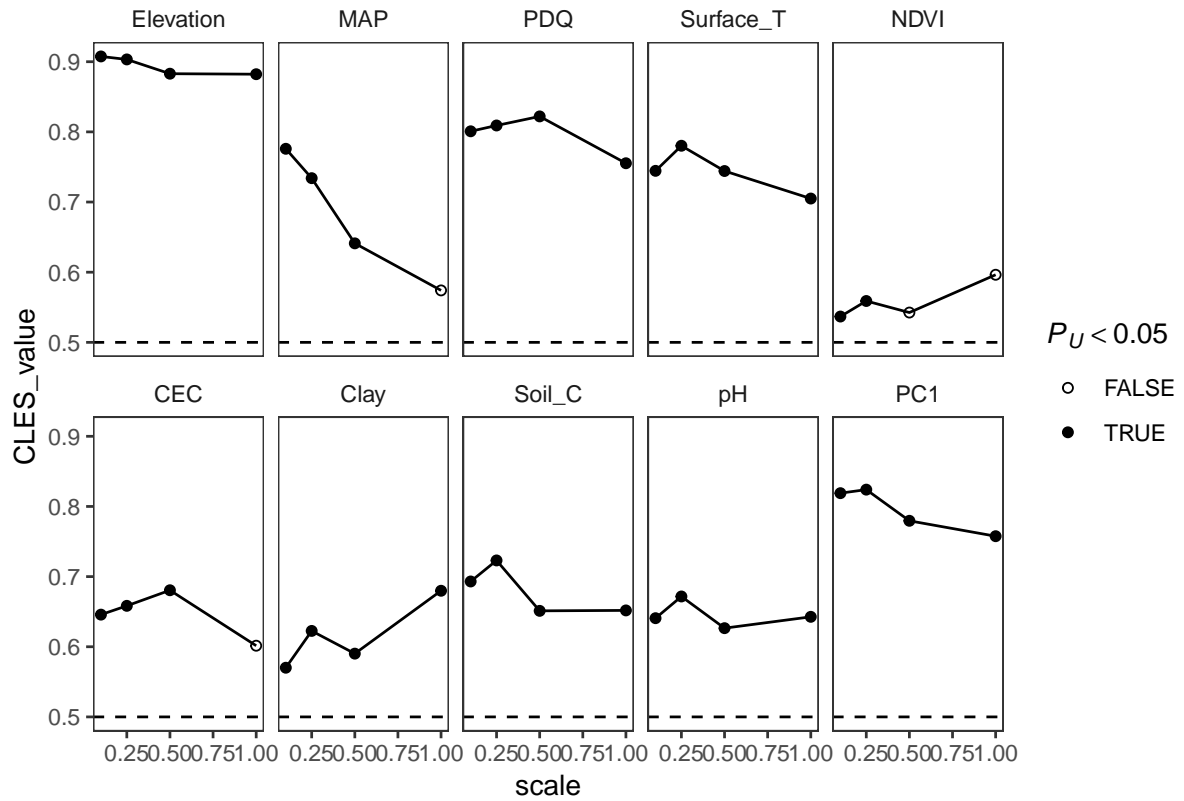
# Summarise those models
CLES_model_summaries <- CLES_models %>%
  map_df(.id = "variable", tidy) %>%
  filter(term != "(Intercept)") %>%
  mutate(sig = case_when(
    p.value <= 0.05 ~ "*",
    p.value <= 0.10 ~ ".",
    TRUE          ~ " ")
  ) %>%
  mutate(variable = factor(variable, levels = var_names %>%
    str_replace_all(" ", "_") %>%
    c("PC1")
  ) %>%
  mutate_if(is.numeric, round, digits = 3) %>%
  dplyr::select(variable, estimate, p.value, sig)
CLES_model_summaries
```

```
## # A tibble: 10 x 4
##   variable estimate p.value sig
##   <fct>      <dbl>   <dbl> <chr>
## 1 Elevation  -0.029   0.135 " "
## 2 MAP        -0.224   0.028 *
## 3 PDQ        -0.055   0.26  " "
## 4 Surface_T  -0.062   0.207 " "
## 5 NDVI        0.059   0.134 " "
## 6 CEC        -0.054   0.361 " "
## 7 Clay        0.104   0.143 " "
## 8 Soil_C     -0.065   0.27  " "
## 9 pH         -0.013   0.729 " "
## 10 PC1       -0.076   0.059 .
```

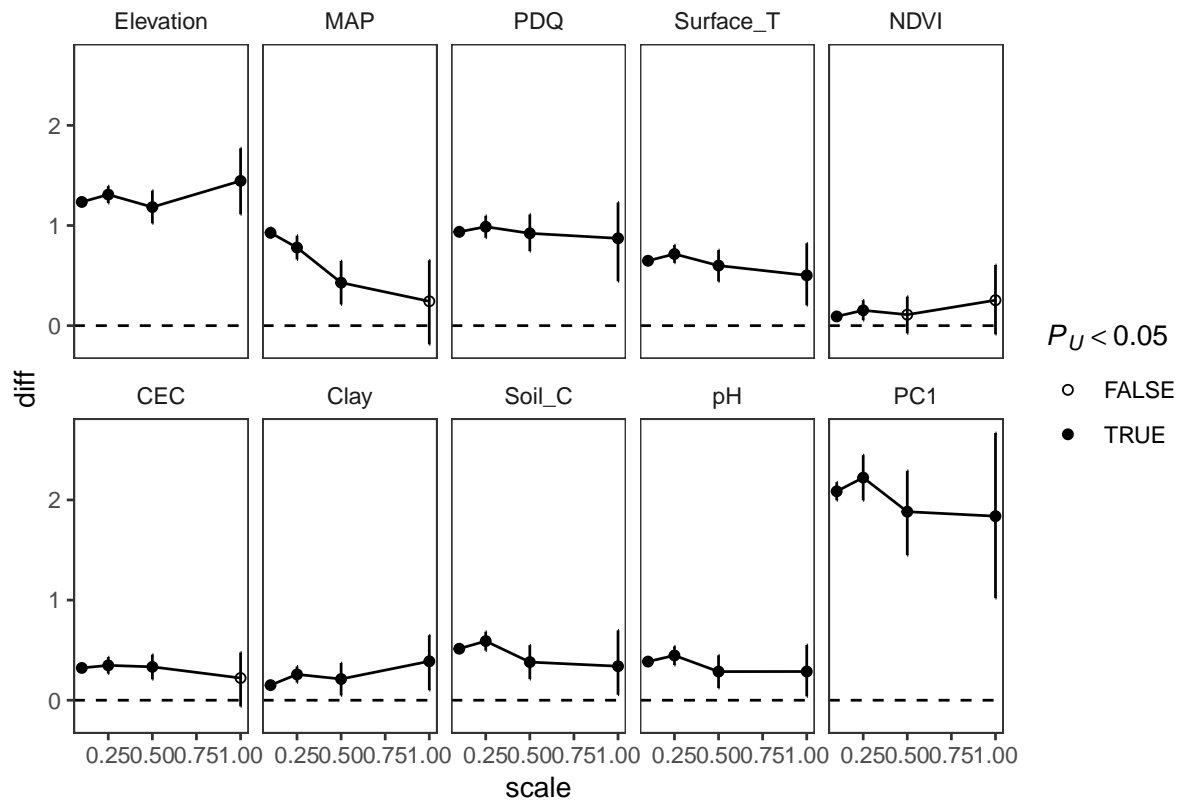
```
heterogeneity %>% #heterogeneity_df %>%
  bind_rows(.id = "scale") %>%
  gather(
    variable, heterogeneity_value,
    -region, -scale#, -lon, -lat
  ) %>%
  mutate(variable = factor(variable, levels = var_names %>%
    str_replace_all(" ", "_") %>%
    c("PC1")
  ) %>%
  ggplot(aes(as.factor(scale), heterogeneity_value, fill = region)) +
  geom_boxplot() +
  facet_wrap(~variable, nrow = 2)
```



```
ggplot(CLES_results) +
  aes(scale, CLES_value, group = variable) +
  geom_hline(yintercept = 0.5, lty = "dashed") +
  geom_line() +
  geom_point(aes(shape = P_U < 0.05)) +
  scale_shape_manual(name = bquote(italic("P"["U"]) < 0.05), values = c(1, 19)) +
  facet_wrap(~variable, nrow = 2)
```



```
ggplot(CLES_results) +
  aes(scale, diff, group = variable) +
  geom_hline(yintercept = 0.0, lty = "dashed") +
  geom_line() +
  geom_errorbar(aes(ymin = U_low, ymax = U_upp), width = 0) +
  geom_point(aes(shape = P_U < 0.05)) +
  scale_shape_manual(name = bquote(italic("P"["U"]) < 0.05), values = c(1, 19)) +
  facet_wrap(~variable, nrow = 2)
```

2. Comparing & decomposing regions' species richness

```
# Assign a variable to the global environment,
# not simply the parent environment (as with <-),
# with the same name
assign_global <- function(x) {
  assign(
    x      = deparse(substitute(x)),
    value = x,
    envir = .GlobalEnv
  )
}

# Read in and assign all polygon objects to global environment
import_region_polygons <- function(borders_dir =
  here("data/derived-data/borders")) {

  # GCFR -----

  GCFR_border      <- readOGR(glue("{borders_dir}/GCFR_border"))
  GCFR_border_buffered <- readOGR(glue("{borders_dir}/GCFR_border_buffered"))
  GCFR_box         <- readOGR(glue("{borders_dir}/GCFR_box"))
  GCFR_QDS         <- readOGR(glue("{borders_dir}/GCFR_QDS"))
  assign_global(GCFR_border)
  assign_global(GCFR_border_buffered)
  assign_global(GCFR_box)
  assign_global(GCFR_QDS)

  # SWAFR -----
```

```

SWAFR_border      <- readOGR(glue("{borders_dir}/SWBP_Mike-Cramer"))
SWAFR_border_buffered <- readOGR(glue("{borders_dir}/SWAFR_border_buffered"))
SWAFR_box         <- readOGR(glue("{borders_dir}/SWAFR_box"))
SWAFR_QDS         <- readOGR(glue("{borders_dir}/SWAFR_QDS"))
assign_global(SWAFR_border)
assign_global(SWAFR_border_buffered)
assign_global(SWAFR_box)
assign_global(SWAFR_QDS)

# FIXME: Why are these shapefile imports throwing non-fatal errors/warnings?
# TODO: Add GIS-std-checkers here too
}

```

```

# Include lon/lat when converting from Raster* to data.frame
raster2df <- function(r) {
  lon_lat <- xyFromCell(r, 1:ncell(r))
  colnames(lon_lat) <- c("lon", "lat")
  df <- as.data.frame(log10(r))
  df <- cbind(lon_lat, df)
  df
}

```

```
import_region_polygons()
```

```

## OGR data source with driver: ESRI Shapefile
## Source: "C:\Users\user\Desktop\Cape-vs-SWA\data\derived-data\borders\GCFR_border", layer: "LA_CUR
## with 2 features
## It has 1 fields
## OGR data source with driver: ESRI Shapefile
## Source: "C:\Users\user\Desktop\Cape-vs-SWA\data\derived-data\borders\GCFR_border_buffered", layer
## with 1 features
## It has 1 fields
## OGR data source with driver: ESRI Shapefile
## Source: "C:\Users\user\Desktop\Cape-vs-SWA\data\derived-data\borders\GCFR_box", layer: "value"
## with 1 features
## It has 1 fields
## OGR data source with driver: ESRI Shapefile
## Source: "C:\Users\user\Desktop\Cape-vs-SWA\data\derived-data\borders\GCFR_QDS", layer: "areakm2"
## with 1920 features
## It has 4 fields
## OGR data source with driver: ESRI Shapefile
## Source: "C:\Users\user\Desktop\Cape-vs-SWA\data\derived-data\borders\SWBP_Mike-Cramer", layer: "S
## with 1 features
## It has 1 fields
## OGR data source with driver: ESRI Shapefile
## Source: "C:\Users\user\Desktop\Cape-vs-SWA\data\derived-data\borders\SWAFR_border_buffered", laye
## with 1 features
## It has 1 fields
## OGR data source with driver: ESRI Shapefile
## Source: "C:\Users\user\Desktop\Cape-vs-SWA\data\derived-data\borders\SWAFR_box", layer: "value"
## with 1 features
## It has 1 fields
## OGR data source with driver: ESRI Shapefile
## Source: "C:\Users\user\Desktop\Cape-vs-SWA\data\derived-data\borders\SWAFR_QDS", layer: "areakm2"
## with 2464 features
## It has 4 fields

```

```

# Import EDS polygons
ZA_EDS <- readOGR(here("data/raw-data/QDGC/qdgc_zaf"), layer = "qdgc_03_zaf")

```

```

## OGR data source with driver: ESRI Shapefile
## Source: "C:\Users\user\Desktop\Cape-vs-SWA\data\raw-data\QDGC\qdgz_zaf", layer: "qdgz_03_zaf"
## with 14144 features
## It has 4 fields

AU_EDS <- readOGR(here("data/raw-data/QDGC/qdgc_au"), layer = "qdgz_03_au")

## OGR data source with driver: ESRI Shapefile
## Source: "C:\Users\user\Desktop\Cape-vs-SWA\data\raw-data\QDGC\qdgz_au", layer: "qdgz_03_au"
## with 89216 features
## It has 4 fields

# Crop to regions
GCFR_EDS <- crop(ZA_EDS, GCFR_box)
SWAFR_EDS <- crop(AU_EDS, SWAFR_box)

Larsen_grid <- rbind(GCFR_EDS, SWAFR_EDS)
Larsen_grid$edgc <- Larsen_grid$qdgc
Larsen_grid$qdgc <- str_remove(Larsen_grid$edgc, ".$")
Larsen_grid$hdgc <- str_remove(Larsen_grid$qdgc, ".$")
Larsen_grid$dgdc <- str_remove(Larsen_grid$hdgc, ".$")

heterogeneity_w_coords <- map2(GCFR_heterogeneity, SWAFR_heterogeneity,
  ~ na.exclude(rbind(
    cbind(region = "GCFR", raster2df(.x)),
    cbind(region = "SWAFR", raster2df(.y))
  ))
)
heterogeneity <- map2(heterogeneity, heterogeneity_w_coords, full_join)

heterogeneity$QDS$QDS <- heterogeneity$QDS %>%
  SpatialPoints(
    coords = data.frame(x = lon, y = lat),
    proj4string = crs(Larsen_grid)
  ) %over%
  Larsen_grid %>%
  pull(qdgc)

heterogeneity$HDS$HDS <- heterogeneity$HDS %>%
  SpatialPoints(
    coords = data.frame(x = lon, y = lat),
    proj4string = crs(Larsen_grid)
  ) %over%
  Larsen_grid %>%
  pull(hdgc)

heterogeneity$DS$DS <- heterogeneity$DS %>%
  SpatialPoints(
    coords = data.frame(x = lon, y = lat),
    proj4string = crs(Larsen_grid)
  ) %over%
  Larsen_grid %>%
  pull(dgc)

GCFR_species_occ <- read_rds(here(
  "data/derived-data/flora",
  "trimmed_GCFR_clean_flora_spdf_species"
))

```

```

SWAFR_species_occ <- read_rds(here(
  "data/derived-data/flora",
  "trimmed_SWAFR_clean_flora_spdf_species"
))

GCFR_species_occ@data %>%
  group_by(species) %>%
  summarise(n_collections = n()) %>%
  arrange(desc(n_collections)) %>%
  write_csv(here("draft-02/manuscript_ver3/GCFR-species.csv"))
GCFR_bad_species <-
  read_csv(here("draft-02/manuscript_ver3/GCFR-species.csv")) %>%
  filter(n_collections < 5) %>%
  pull(species)
SWAFR_species_occ@data %>%
  group_by(species) %>%
  summarise(n_collections = n()) %>%
  arrange(desc(n_collections)) %>%
  write_csv(here("draft-02/manuscript_ver3/SWAFR-species.csv"))
SWAFR_bad_species <-
  read_csv(here("draft-02/manuscript_ver3/SWAFR-species.csv")) %>%
  filter(n_collections < 5) %>%
  pull(species)

species_occ <- rbind(GCFR_species_occ, SWAFR_species_occ)

species_occ$EDS <- species_occ %over%
  Larsen_grid %>%
  pull(edgc)
species_occ@data$EDS %<>% as.character()
species_occ$QDS <- str_remove(species_occ$EDS, ".$")
species_occ$HDS <- str_remove(species_occ$QDS, ".$")
species_occ$DSD <- str_remove(species_occ$HDS, ".$")

species_occ_data <- species_occ@data %>%
  filter(!(species %in% c(GCFR_bad_species, SWAFR_bad_species)))

QDS_richness <- species_occ_data %>%
  group_by(QDS) %>%
  summarise(
    n_EDS = length(unique(EDS)),
    n_collections = length(species),
    QDS_richness = length(unique(species))
  )
mean_QDS_richness <- QDS_richness %>%
  mutate(HDS = str_remove(QDS, ".$")) %>%
  group_by(HDS) %>%
  summarise(
    mean_QDS_richness = mean(QDS_richness)
  )
HDS_richness <- species_occ_data %>%
  group_by(HDS) %>%
  summarise(
    n_QDS = length(unique(QDS)),
    n_collections = length(species),
    HDS_richness = length(unique(species))
  )

```

```

mean_HDS_richness <- HDS_richness %>%
  mutate(DS = str_remove(HDS, ".$")) %>%
  group_by(DS) %>%
  summarise(
    mean_HDS_richness = mean(HDS_richness)
  )
DS_richness <- species_occ_data %>%
  group_by(DS) %>%
  summarise(
    n_HDS = length(unique(HDS)),
    n_collections = length(species),
    DS_richness = length(unique(species))
  )

```

```

data <- heterogeneity[-1]
data$QDS %<>%
  as_tibble() %>%
  full_join(QDS_richness) %>%
  na.exclude() %>%
  filter(n_EDS == 4)
data$HDS %<>%
  as_tibble() %>%
  full_join(mean_QDS_richness) %>%
  full_join(HDS_richness) %>%
  na.exclude() %>%
  filter(n_QDS == 4) %>%
  mutate(
    QDS_turnover = HDS_richness - mean_QDS_richness,
    QDS_turnover_prop = QDS_turnover / HDS_richness
  )
data$DS %<>%
  as_tibble() %>%
  full_join(mean_HDS_richness) %>%
  full_join(DS_richness) %>%
  na.exclude() %>%
  filter(n_HDS == 4) %>%
  mutate(
    HDS_turnover = DS_richness - mean_HDS_richness,
    HDS_turnover_prop = HDS_turnover / DS_richness,
  )

```

```

m1 <- lm(QDS_richness ~ PC1, data$QDS)
m2 <- lm(log(QDS_richness) ~ PC1, data$QDS)
m3 <- lm(log10(QDS_richness) ~ PC1, data$QDS)
AIC(m1, m2, m3)

```

```

##      df      AIC
## m1   3 9655.6932
## m2   3 1626.4966
## m3   3  487.2083

```

```

# Choose m3
m4 <- lm(log10(QDS_richness) ~ PC1 + region, data$QDS)
m5 <- lm(log10(QDS_richness) ~ PC1 * region, data$QDS)
AIC(m3, m4, m5)

```

```

##      df      AIC
## m3   3  487.2083
## m4   4  488.4461

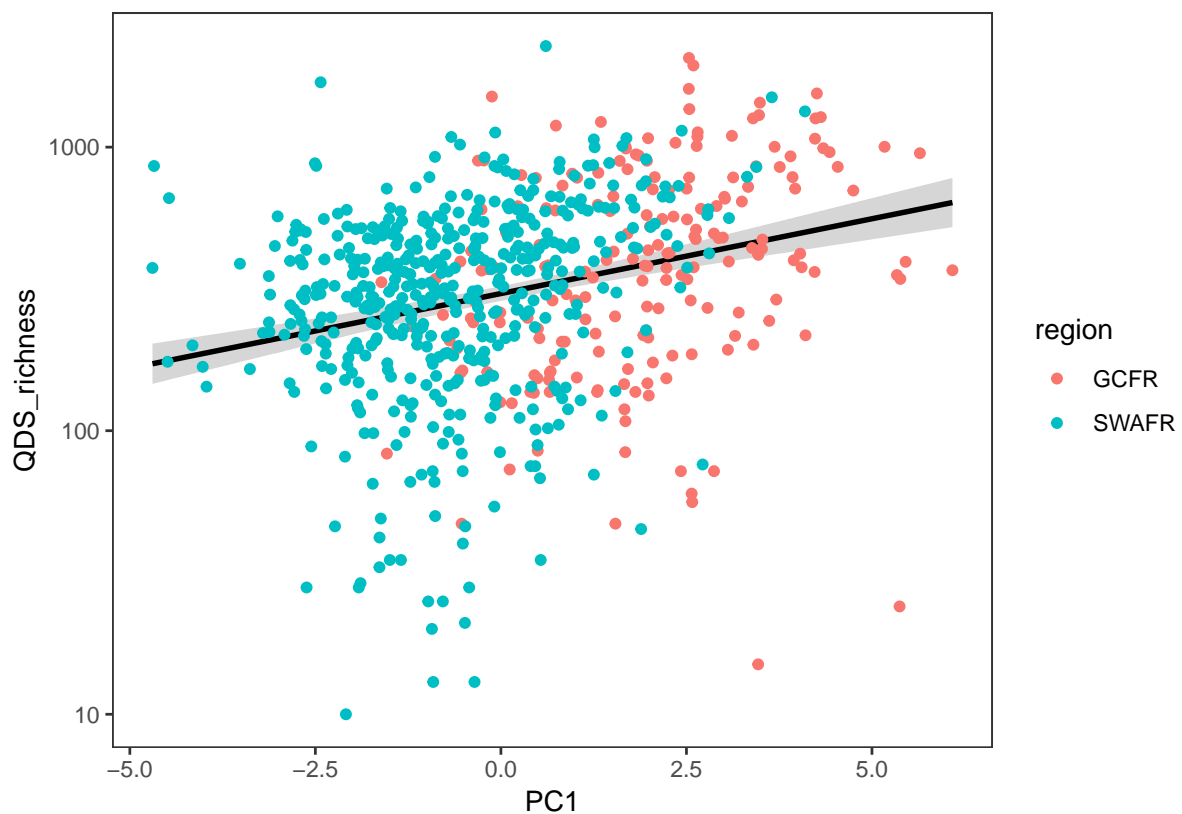
```

```
## m5 5 490.1653
```

```
# Choose m3  
summary(m3)
```

```
##  
## Call:  
## lm(formula = log10(QDS_richness) ~ PC1, data = data$QDS)  
##  
## Residuals:  
##      Min       1Q   Median       3Q      Max   
## -1.49045 -0.17294  0.05534  0.24052  0.87306   
##  
## Coefficients:  
##              Estimate Std. Error t value Pr(>|t|)      
## (Intercept)  2.483705   0.013202 188.133 < 2e-16 ***  
## PC1          0.052746   0.007038   7.494 2.08e-13 ***  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## Residual standard error: 0.3447 on 681 degrees of freedom  
## Multiple R-squared:  0.07619,    Adjusted R-squared:  0.07484   
## F-statistic: 56.17 on 1 and 681 DF,  p-value: 2.076e-13
```

```
ggplot(data$QDS, aes(PC1, QDS_richness)) +  
  geom_smooth(method = lm, colour = "black") +  
  geom_point(aes(colour = region)) +  
  scale_y_log10()
```



```
data$QDS$PC1_residual <- m3$residuals  
  
m1 <- lm(HDS_richness ~ PC1, data$HDS)  
m2 <- lm(log(HDS_richness) ~ PC1, data$HDS)
```

```

m3 <- lm(log10(HDS_richness) ~ PC1, data$HDS)
AIC(m1, m2, m3)

##      df      AIC
## m1   3 3031.5108
## m2   3  459.3537
## m3   3  125.7407

# Choose m3
m4 <- lm(log10(HDS_richness) ~ PC1 + region, data$HDS)
m5 <- lm(log10(HDS_richness) ~ PC1 * region, data$HDS)
AIC(m3, m4, m5)

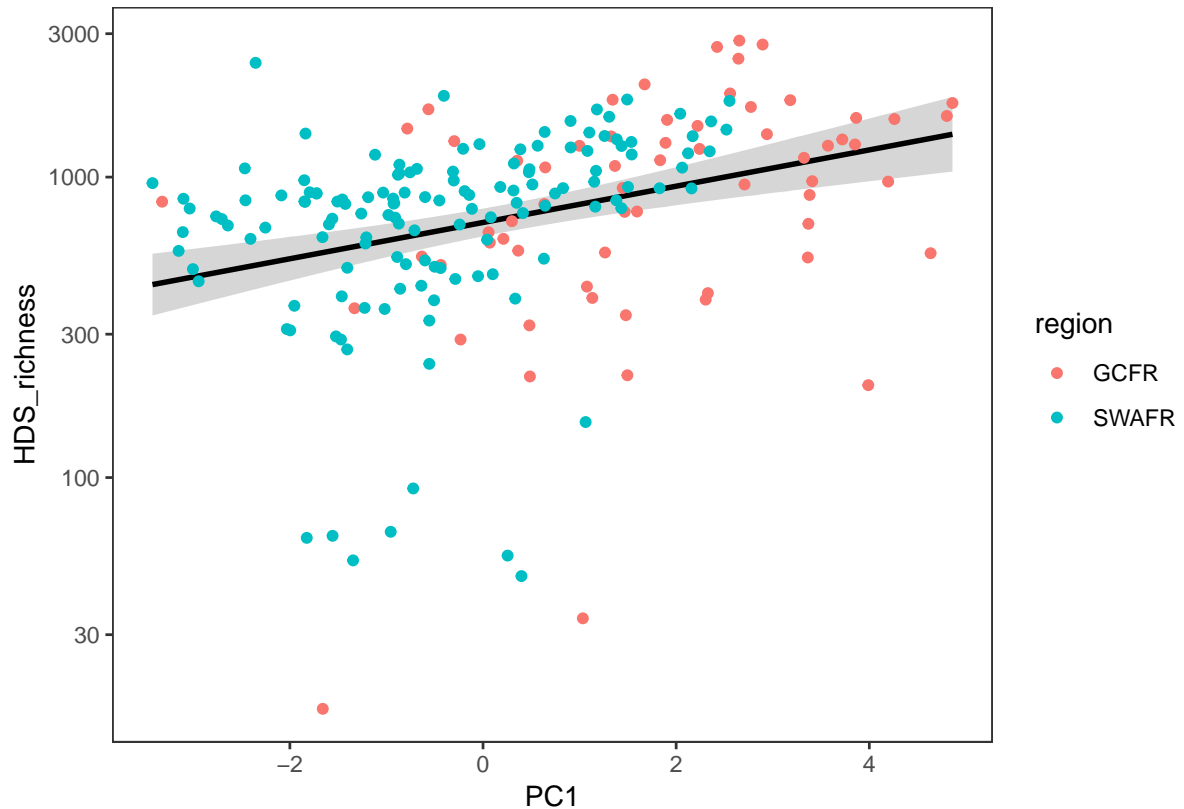
##      df      AIC
## m3   3 125.7407
## m4   4 125.1210
## m5   5 126.7163

# Choose m3
summary(m3)

##
## Call:
## lm(formula = log10(HDS_richness) ~ PC1, data = data$HDS)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.51825 -0.09608  0.09299  0.19342  0.67349
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.84889    0.02342 121.652 < 2e-16 ***
## PC1          0.06038    0.01271   4.752 3.86e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3281 on 198 degrees of freedom
## Multiple R-squared:  0.1024, Adjusted R-squared:  0.09786
## F-statistic: 22.59 on 1 and 198 DF, p-value: 3.858e-06

ggplot(data$HDS, aes(PC1, HDS_richness)) +
  geom_smooth(method = lm, colour = "black") +
  geom_point(aes(colour = region)) +
  scale_y_log10()

```



```
data$HDS$PC1_residual <- m3$residuals

m1 <- lm(DS_richness ~ PC1, data$DS)
m2 <- lm(log(DS_richness) ~ PC1, data$DS)
m3 <- lm(log10(DS_richness) ~ PC1, data$DS)
AIC(m1, m2, m3)

##      df      AIC
## m1   3 812.69955
## m2   3  98.49444
## m3   3  13.42313

# Choose m3
m4 <- lm(log10(DS_richness) ~ PC1 + region, data$DS)
m5 <- lm(log10(DS_richness) ~ PC1 * region, data$DS)
AIC(m3, m4, m5)

##      df      AIC
## m3   3 13.42313
## m4   4 15.05489
## m5   5 16.87791

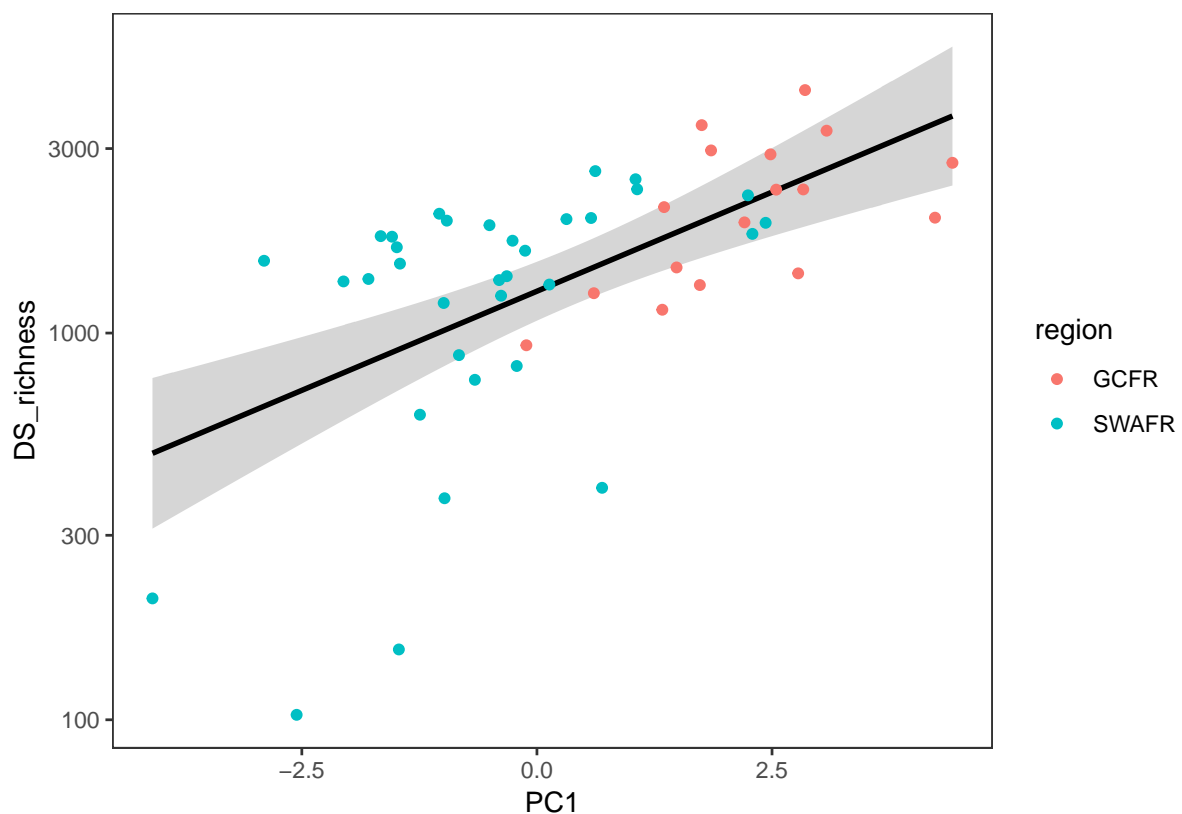
# Choose m3
summary(m3)

##
## Call:
## lm(formula = log10(DS_richness) ~ PC1, data = data$DS)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.83339 -0.12455  0.07006  0.19871  0.37622
##
```



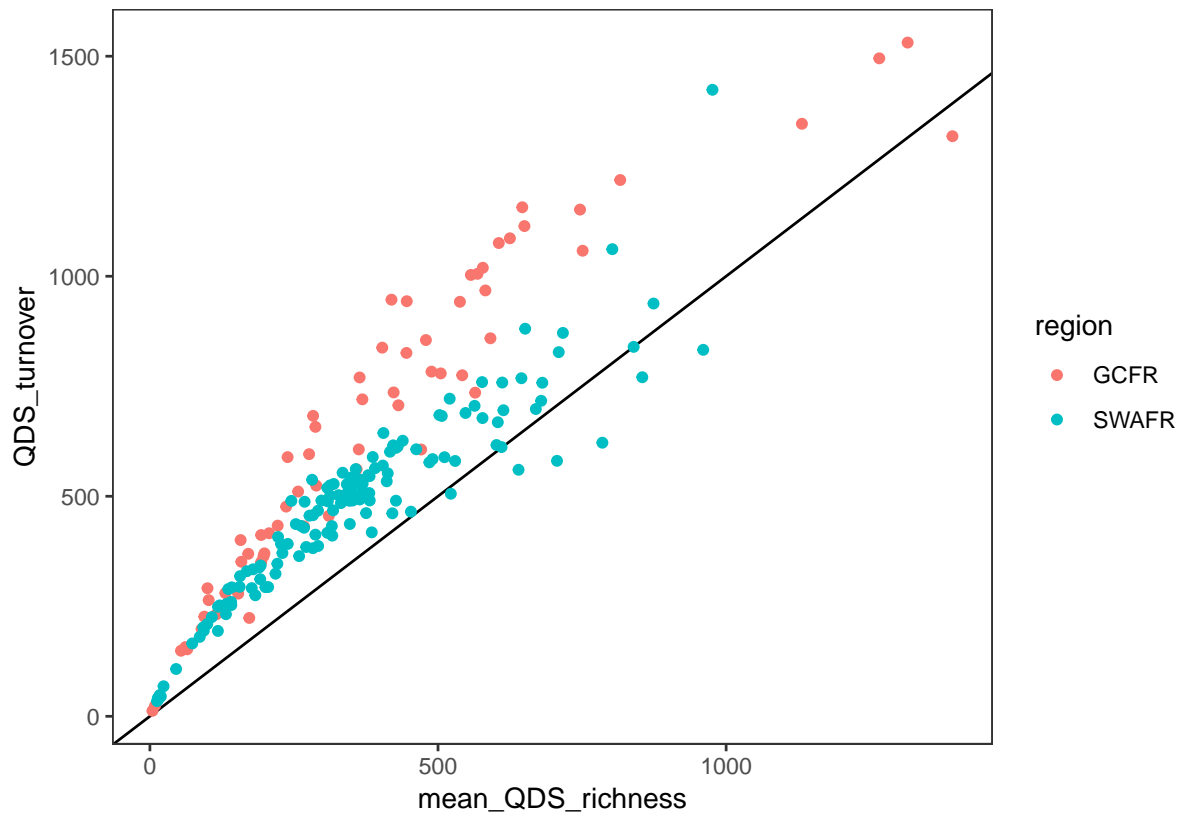
```
## Coefficients:
##           Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.10819    0.03792  81.975 < 2e-16 ***
## PC1          0.10252    0.02008   5.106 5.37e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2655 on 49 degrees of freedom
## Multiple R-squared:  0.3473, Adjusted R-squared:  0.334
## F-statistic: 26.08 on 1 and 49 DF,  p-value: 5.365e-06
```

```
ggplot(data$DS, aes(PC1, DS_richness)) +
  geom_smooth(method = lm, colour = "black") +
  geom_point(aes(colour = region)) +
  scale_y_log10()
```

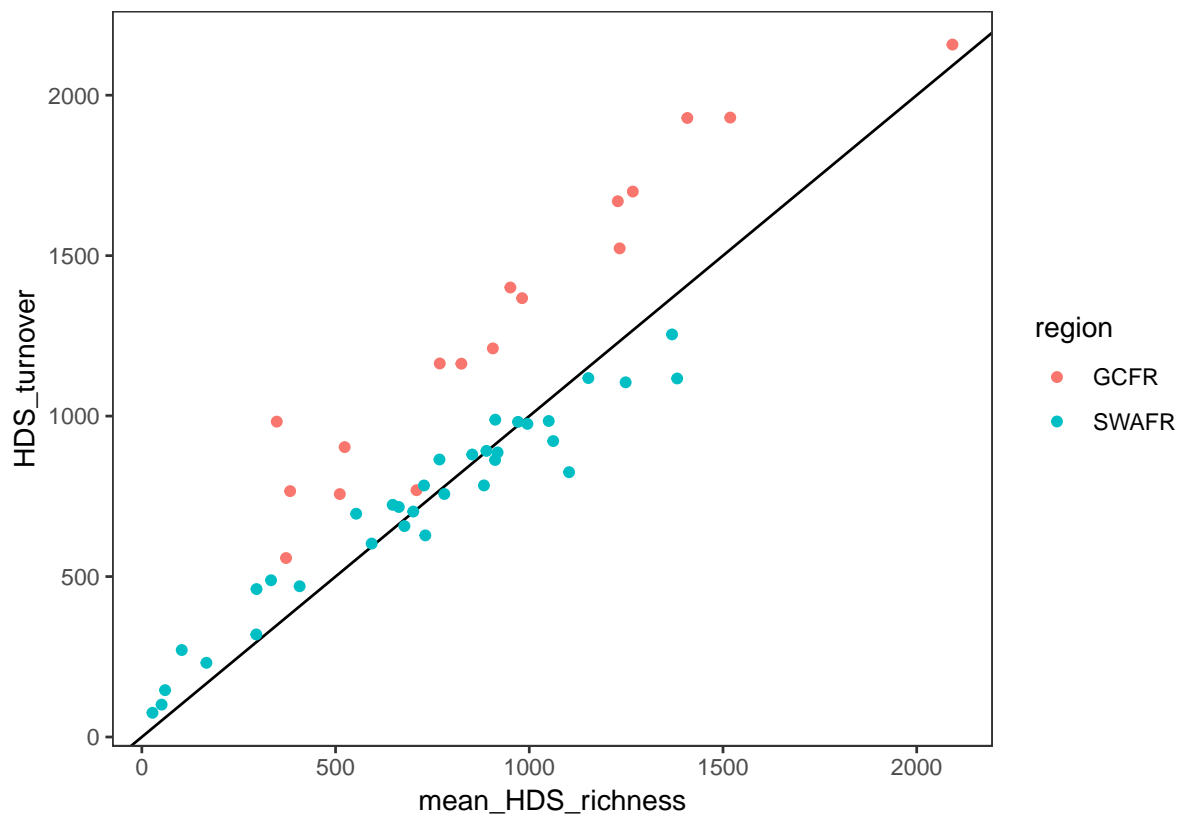


```
data$DS$PC1_residual <- m3$residuals
```

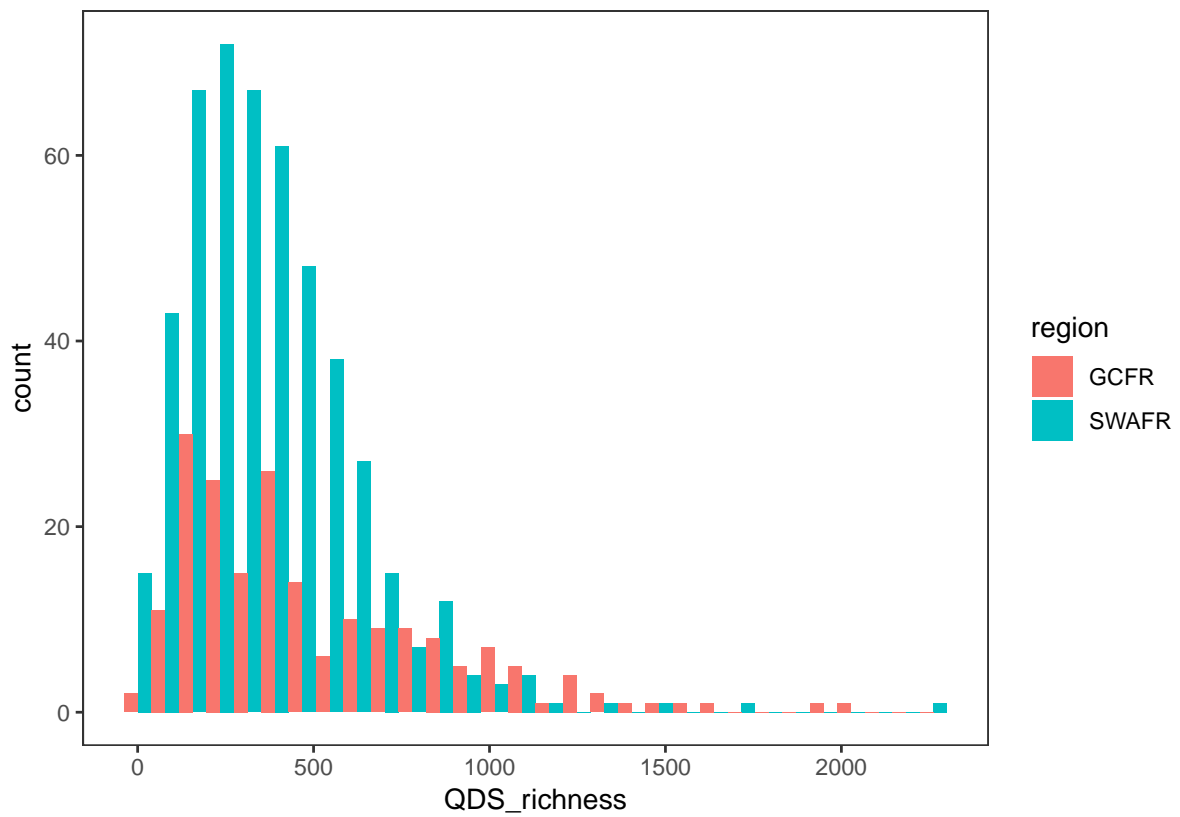
```
ggplot(data$HDS, aes(mean_QDS_richness, QDS_turnover, colour = region)) +
  geom_abline(intercept = 0, slope = 1) +
  geom_point()
```



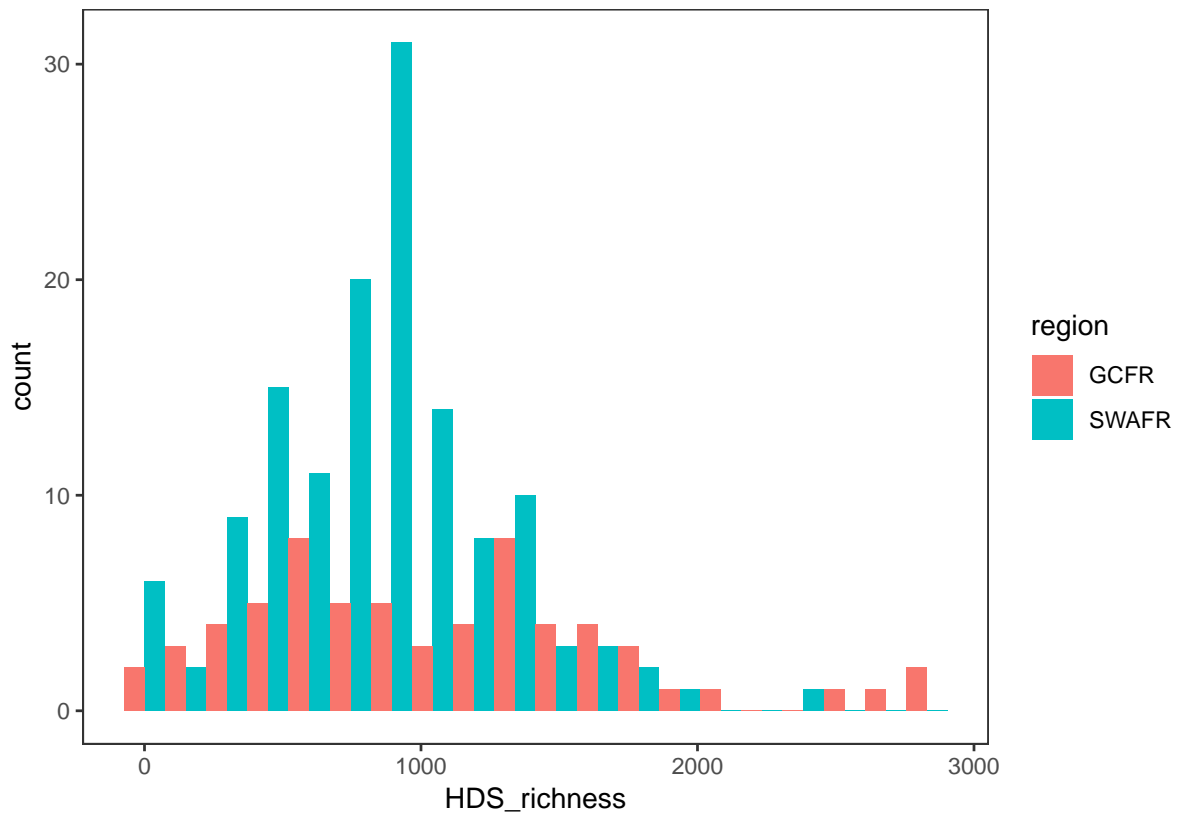
```
ggplot(data$DS, aes(mean_HDS_richness, HDS_turnover, colour = region)) +
  geom_abline(intercept = 0, slope = 1) +
  geom_point()
```



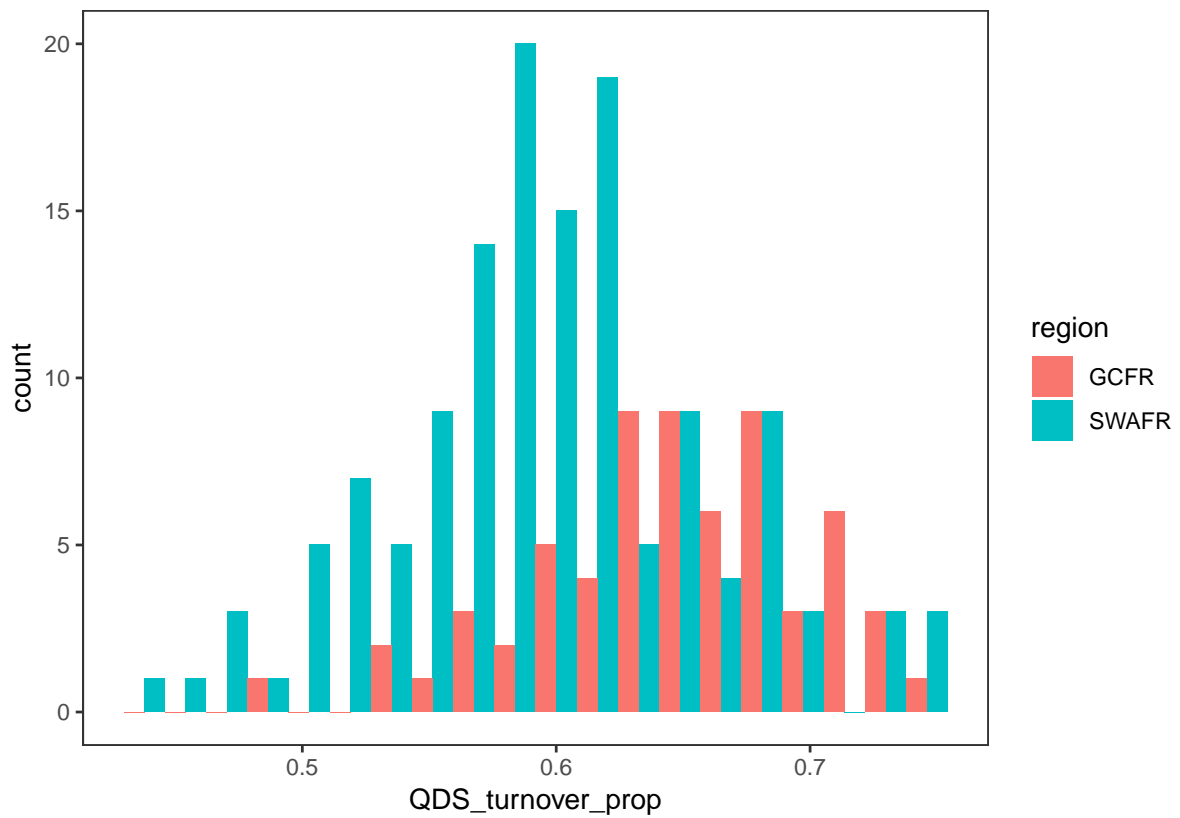
```
ggplot(data$QDS, aes(QDS_richness, fill = region)) +  
  geom_histogram(bins = 30, position = "dodge")
```



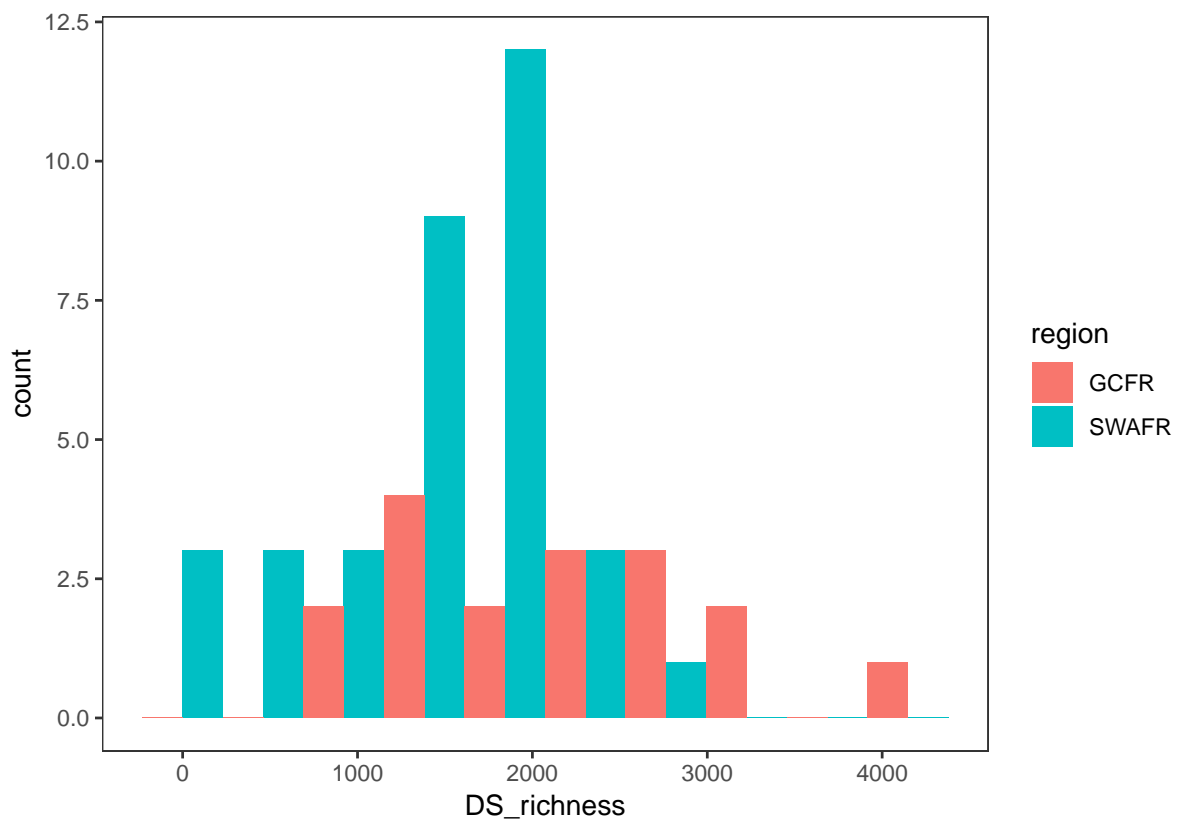
```
ggplot(data$HDS, aes(HDS_richness, fill = region)) +  
  geom_histogram(bins = 20, position = "dodge")
```



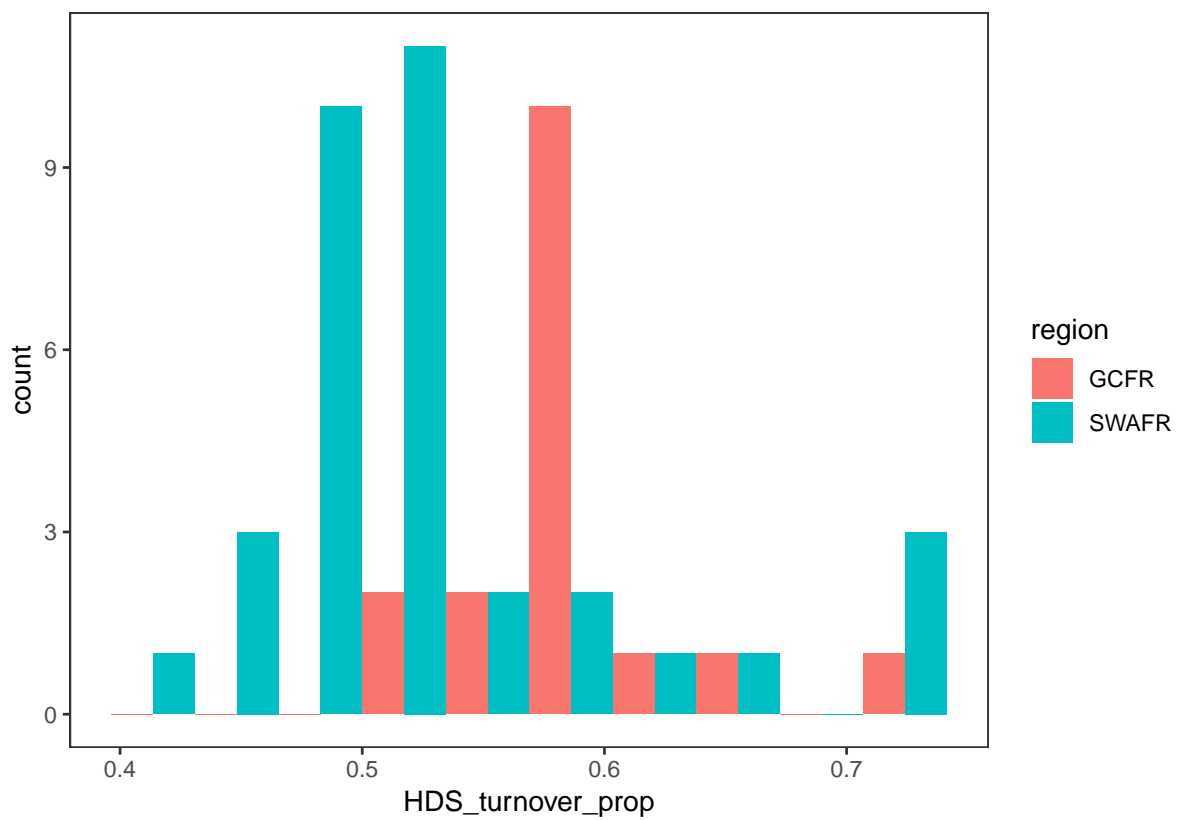
```
ggplot(data$HDS, aes(QDS_turnover_prop, fill = region)) +  
  geom_histogram(bins = 20, position = "dodge")
```



```
ggplot(data$DS, aes(DS_richness, fill = region)) +  
  geom_histogram(bins = 10, position = "dodge")
```



```
ggplot(data$DS, aes(HDS_turnover_prop, fill = region)) +  
  geom_histogram(bins = 10, position = "dodge")
```



```

wilcox.test(QDS_richness ~ region, data$QDS)

##
## Wilcoxon rank sum test with continuity correction
##
## data: QDS_richness by region
## W = 56036, p-value = 0.0002828
## alternative hypothesis: true location shift is not equal to 0
wilcox.test(HDS_richness ~ region, data$HDS)

##
## Wilcoxon rank sum test with continuity correction
##
## data: HDS_richness by region
## W = 5116.5, p-value = 0.0454
## alternative hypothesis: true location shift is not equal to 0
wilcox.test(DS_richness ~ region, data$DS)

##
## Wilcoxon rank sum test
##
## data: DS_richness by region
## W = 423, p-value = 0.006739
## alternative hypothesis: true location shift is not equal to 0
wilcox.test(QDS_turnover ~ region, data$HDS)

##
## Wilcoxon rank sum test with continuity correction
##
## data: QDS_turnover by region
## W = 5535, p-value = 0.001955
## alternative hypothesis: true location shift is not equal to 0
wilcox.test(QDS_turnover_prop ~ region, data$HDS)

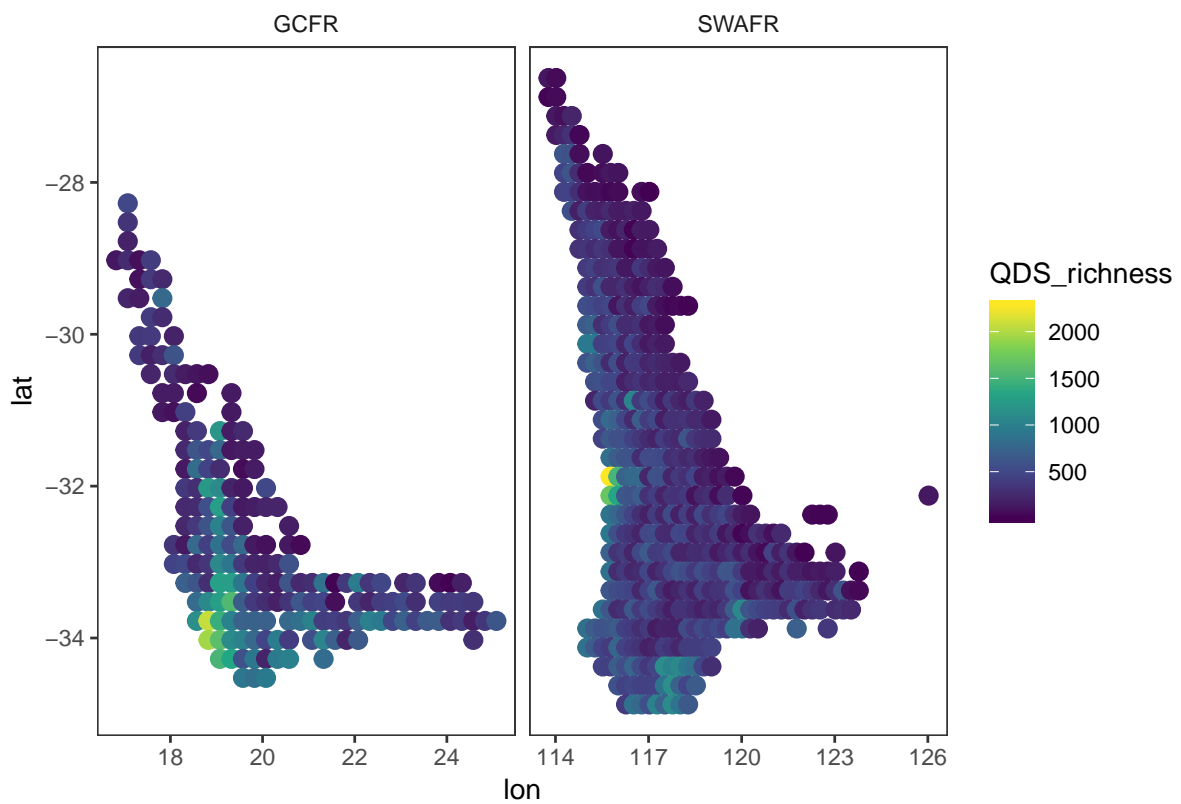
##
## Wilcoxon rank sum test with continuity correction
##
## data: QDS_turnover_prop by region
## W = 6441, p-value = 4.507e-08
## alternative hypothesis: true location shift is not equal to 0
wilcox.test(HDS_turnover ~ region, data$DS)

##
## Wilcoxon rank sum test
##
## data: HDS_turnover by region
## W = 483, p-value = 4.736e-05
## alternative hypothesis: true location shift is not equal to 0
wilcox.test(HDS_turnover_prop ~ region, data$DS)

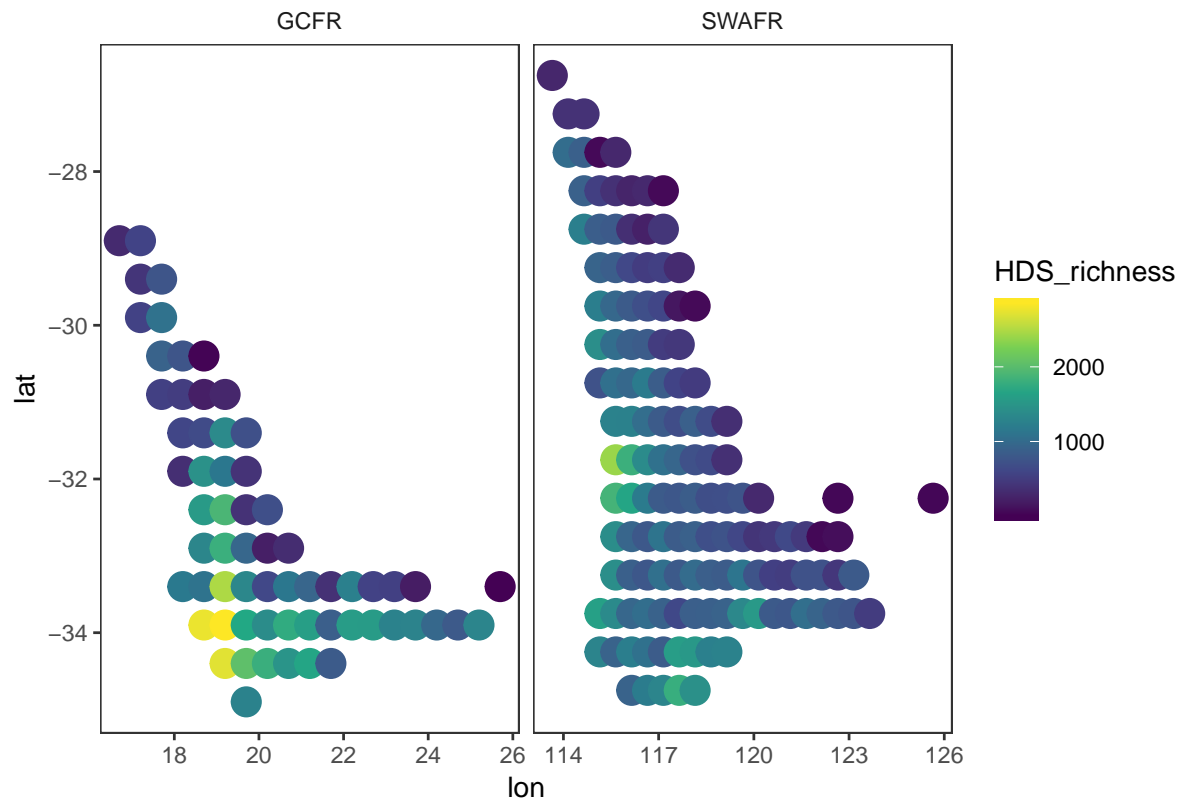
##
## Wilcoxon rank sum test
##
## data: HDS_turnover_prop by region
## W = 469, p-value = 0.0001875
## alternative hypothesis: true location shift is not equal to 0

```

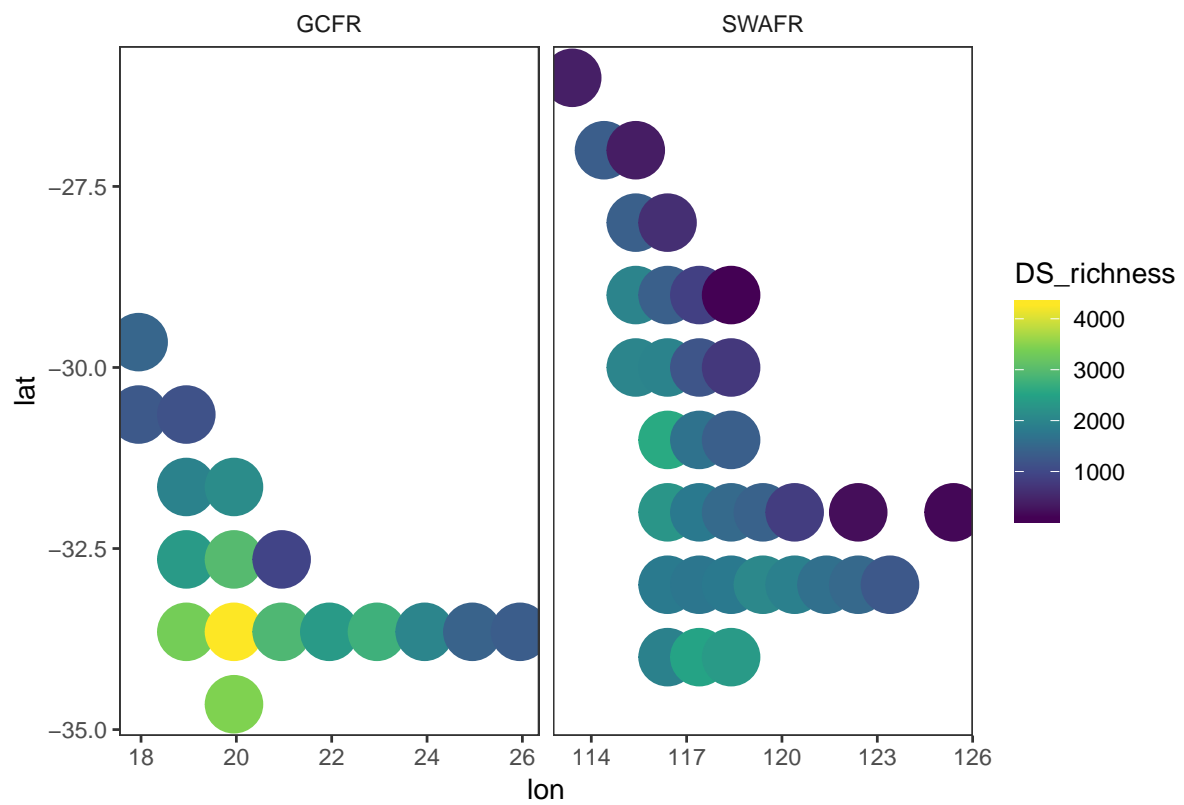
```
ggplot(data$QDS, aes(lon, lat, colour = QDS_richness)) +
  geom_point(size = 3) +
  facet_grid(~region, scales = "free_x") +
  scale_colour_viridis_c()
```



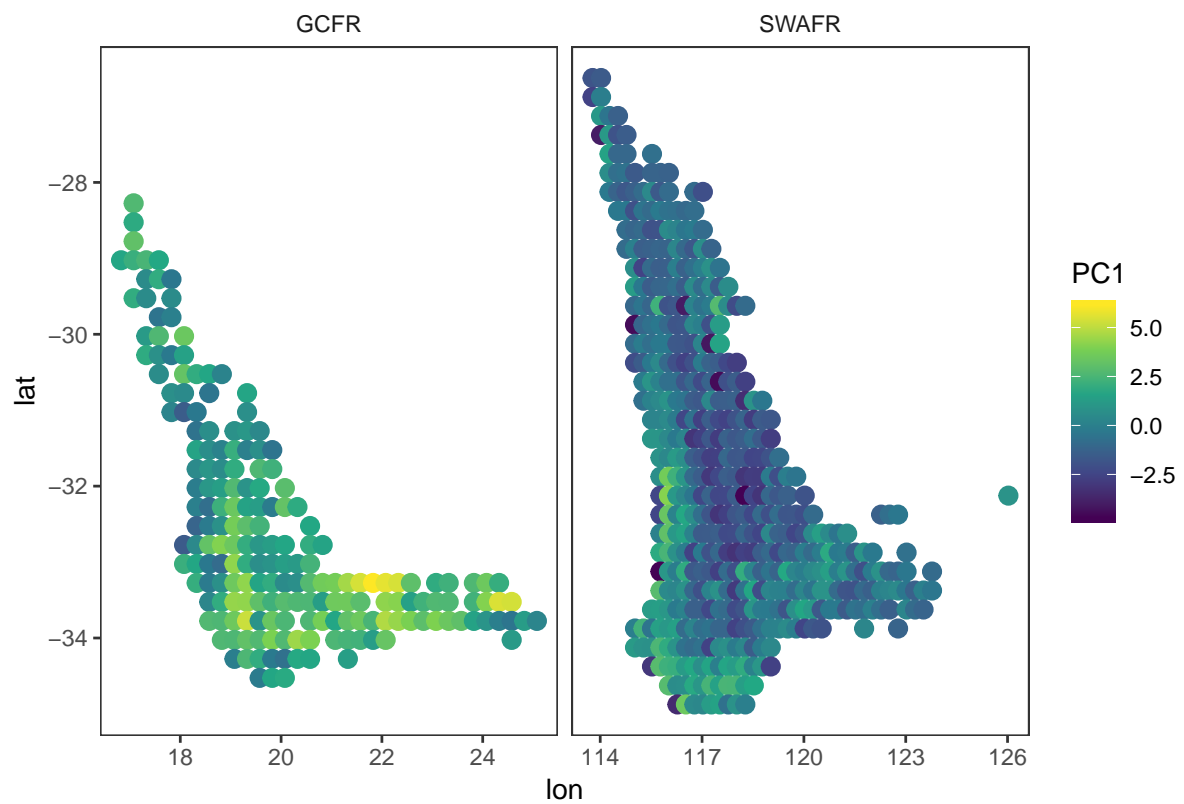
```
ggplot(data$HDS, aes(lon, lat, colour = HDS_richness)) +
  geom_point(size = 5) +
  facet_grid(~region, scales = "free_x") +
  scale_colour_viridis_c()
```



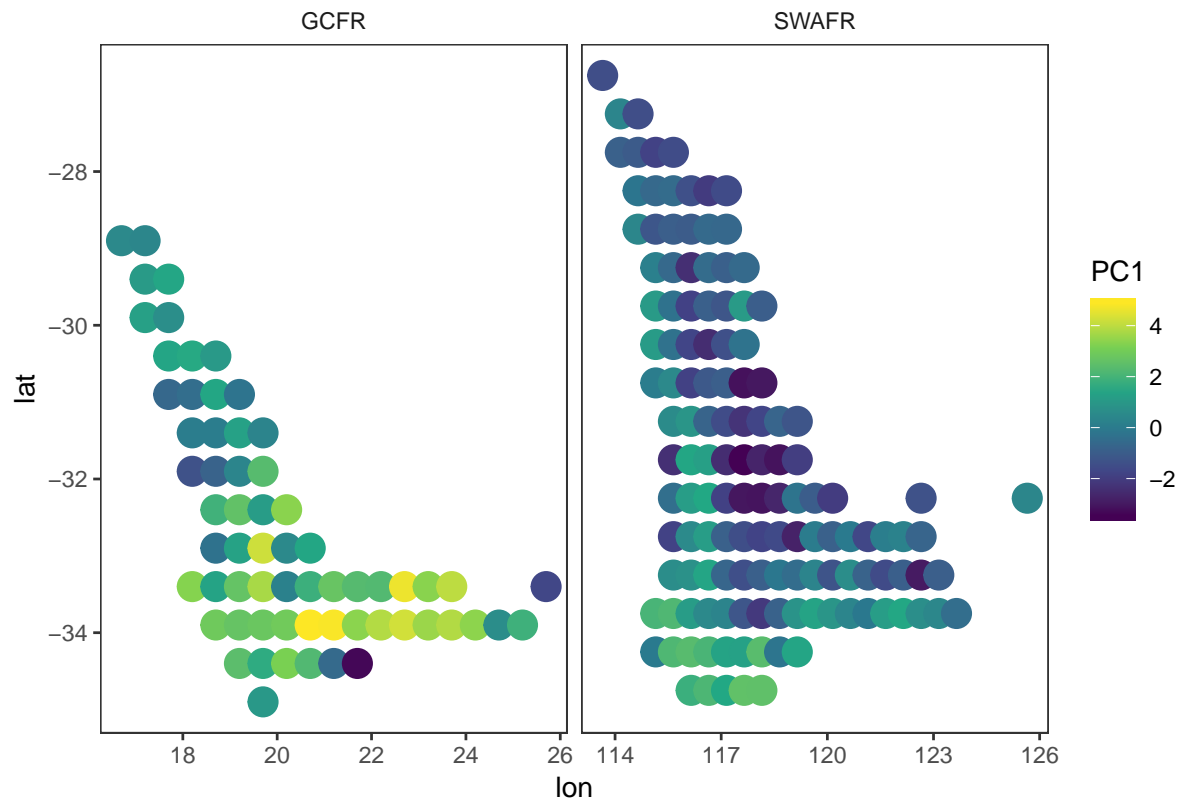
```
ggplot(data$DS, aes(lon, lat, colour = DS_richness)) +
  geom_point(size = 10) +
  facet_grid(~region, scales = "free_x") +
  scale_colour_viridis_c()
```



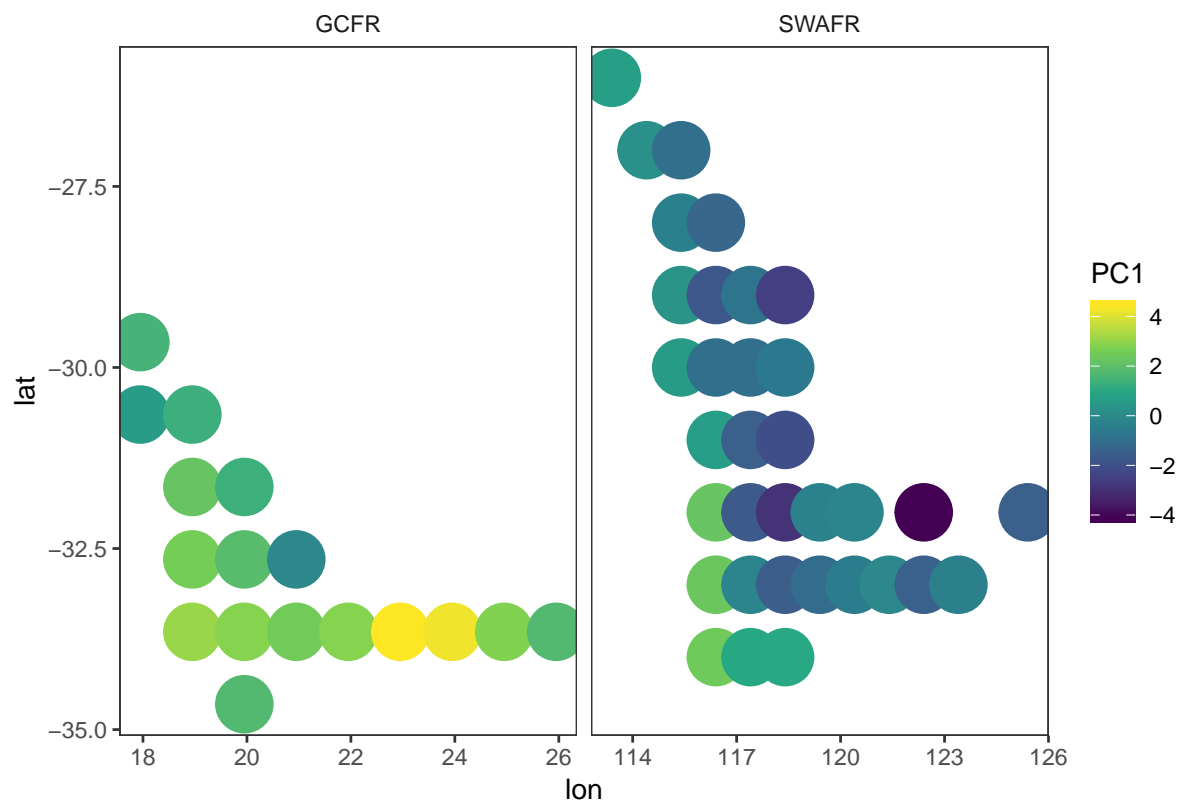

```
ggplot(data$QDS, aes(lon, lat, colour = PC1)) +
  geom_point(size = 3) +
  facet_grid(~region, scales = "free_x") +
  scale_colour_viridis_c()
```



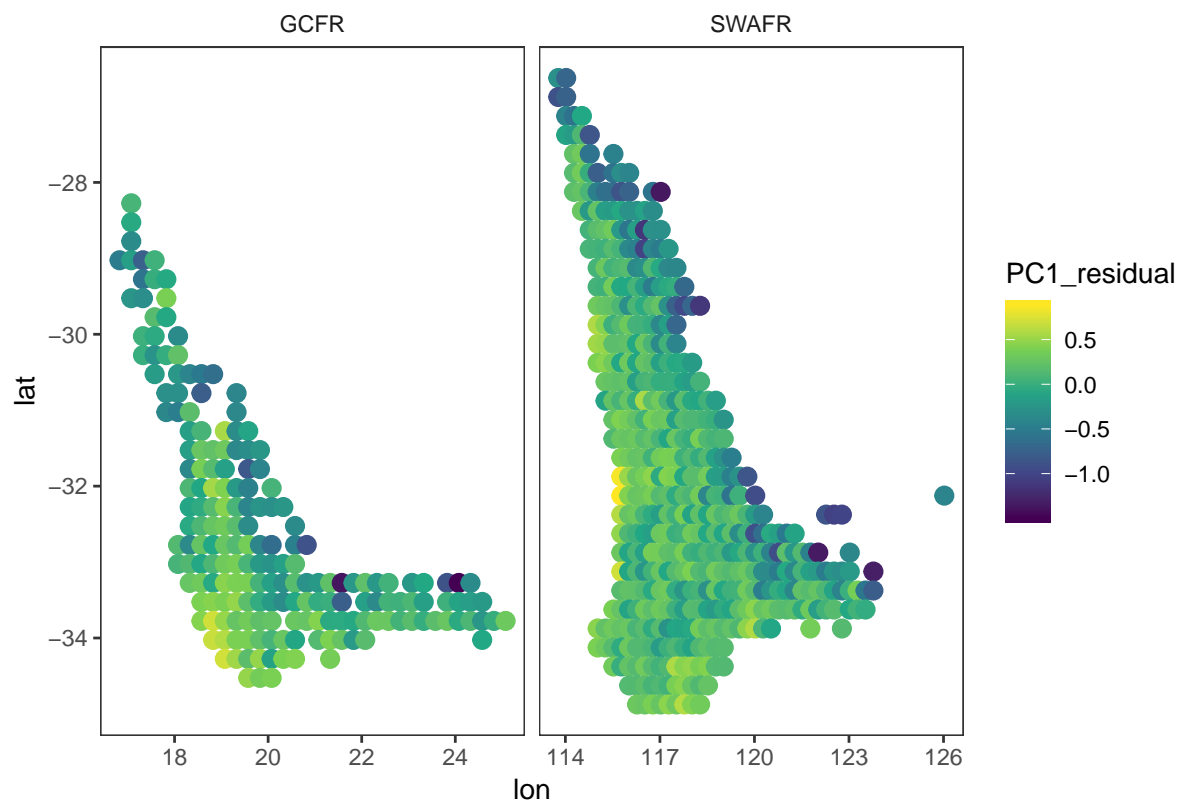
```
ggplot(data$HDS, aes(lon, lat, colour = PC1)) +
  geom_point(size = 5) +
  facet_grid(~region, scales = "free_x") +
  scale_colour_viridis_c()
```



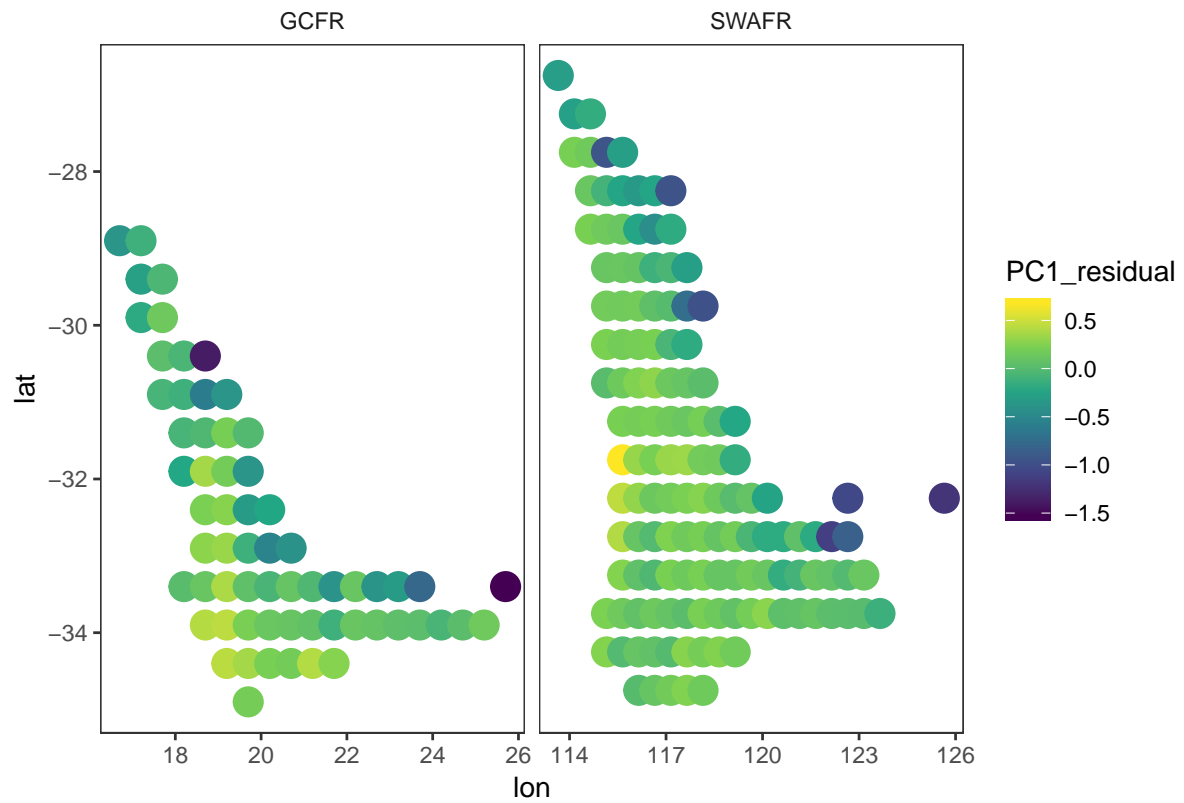
```
ggplot(data$DS, aes(lon, lat, colour = PC1)) +  
  geom_point(size = 10) +  
  facet_grid(~region, scales = "free_x") +  
  scale_colour_viridis_c()
```



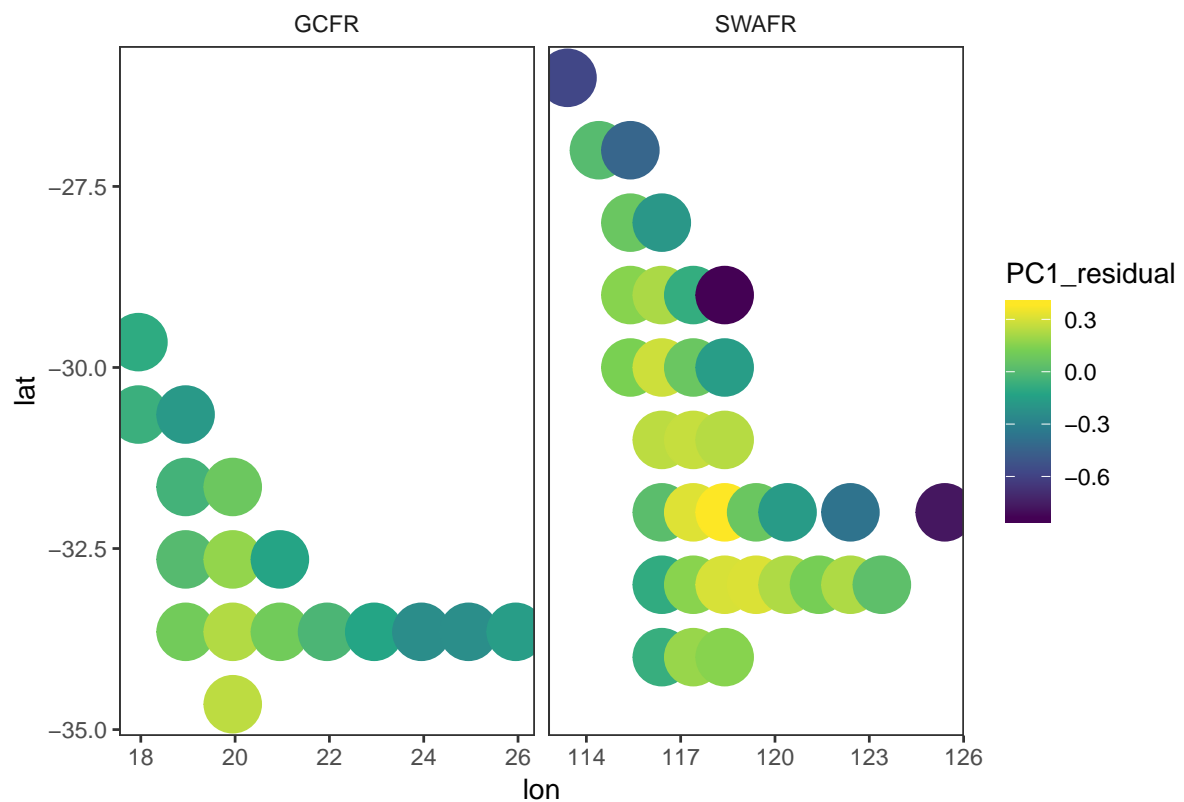
```
ggplot(data$QDS, aes(lon, lat, colour = PC1_residual)) +
  geom_point(size = 3) +
  facet_grid(~region, scales = "free_x") +
  scale_colour_viridis_c()
```



```
ggplot(data$HDS, aes(lon, lat, colour = PC1_residual)) +
  geom_point(size = 5) +
  facet_grid(~region, scales = "free_x") +
  scale_colour_viridis_c()
```



```
ggplot(data$DS, aes(lon, lat, colour = PC1_residual)) +
  geom_point(size = 10) +
  facet_grid(~region, scales = "free_x") +
  scale_colour_viridis_c()
```



3. Environmental heterogeneity as an explanation of species richness

3.1. Univariate models

```

predictor_names <- c(str_replace_all(var_names, " ", "_"), "PC1")

models_non_region <- map(predictor_names,
  ~lm(paste("log10(QDS_richness) ~", .x), data$QDS)
)
names(models_non_region) <- predictor_names
models_add_region <- map(predictor_names,
  ~lm(paste("log10(QDS_richness) ~", .x, "+ region"), data$QDS)
)
names(models_add_region) <- predictor_names
models_int_region <- map(predictor_names,
  ~lm(paste("log10(QDS_richness) ~", .x, "* region"), data$QDS)
)
names(models_int_region) <- predictor_names
knitr::kable(pmap_dfr(
  .l = list(models_non_region, models_add_region, models_int_region),
  .id = "variable",
  .f = ~ AIC(..1, ..2, ..3) %>%
    mutate(
      model_rank = 1:3,
      model_type = c(" ", "+", "x")[model_rank],
      delta_AIC = AIC - min(AIC),
      best_model = (model_rank == min(model_rank[delta_AIC < 2]))
    ) %>%
    filter(best_model) %>%
    dplyr::select(-df, -AIC, -model_rank, -best_model)
))

```

variable	model_type	delta_AIC
Elevation		0.0000000
MAP		0.0000000
PDQ		0.4844252
Surface_T		0.1758586
NDVI	+	1.8932475
CEC	+	0.0000000
Clay	+	0.0000000
Soil_C	+	0.0000000
pH	+	0.0000000
PC1		0.0000000

```

models_non_region <- map(predictor_names,
  ~lm(paste("log10(HDS_richness) ~", .x), data$HDS)
)
names(models_non_region) <- predictor_names
models_add_region <- map(predictor_names,
  ~lm(paste("log10(HDS_richness) ~", .x, "+ region"), data$HDS)
)
names(models_add_region) <- predictor_names
models_int_region <- map(predictor_names,
  ~lm(paste("log10(HDS_richness) ~", .x, "* region"), data$HDS)
)

```

```

names(models_int_region) <- predictor_names
pmap_dfr(
  .l = list(models_non_region, models_add_region, models_int_region),
  .id = "variable",
  .f = ~ AIC(..1, ..2, ..3) %>%
    mutate(
      model_rank = 1:3,
      model_type = c(" ", "+", "x")[model_rank],
      delta_AIC = AIC - min(AIC),
      best_model = (model_rank == min(model_rank[delta_AIC < 2]))
    ) %>%
  filter(best_model) %>%
  dplyr::select(-df, -AIC, -model_rank, -best_model)
)

```

```

##      variable model_type delta_AIC
## 1 Elevation              0.0000000
## 2      MAP              0.0000000
## 3      PDQ              0.9498597
## 4 Surface_T            0.0000000
## 5      NDVI            0.0000000
## 6      CEC            0.2008189
## 7      Clay            0.0000000
## 8    Soil_C            0.0000000
## 9       pH            0.7342033
## 10     PC1            0.6197164

```

```

models_non_region <- map(predictor_names,
  ~lm(glue("log10(DS_richness) ~", .x), data$DS)
)
names(models_non_region) <- predictor_names
models_add_region <- map(predictor_names,
  ~lm(paste("log10(DS_richness) ~", .x, "+ region"), data$DS)
)
names(models_add_region) <- predictor_names
models_int_region <- map(predictor_names,
  ~lm(paste("log10(DS_richness) ~", .x, "* region"), data$DS)
)
names(models_int_region) <- predictor_names
pmap_dfr(
  .l = list(models_non_region, models_add_region, models_int_region),
  .id = "variable",
  .f = ~ AIC(..1, ..2, ..3) %>%
    mutate(
      model_rank = 1:3,
      model_type = c(" ", "+", "x")[model_rank],
      delta_AIC = AIC - min(AIC),
      best_model = (model_rank == min(model_rank[delta_AIC < 2]))
    ) %>%
  filter(best_model) %>%
  dplyr::select(-df, -AIC, -model_rank, -best_model)
)

```

```

##      variable model_type delta_AIC
## 1 Elevation              0.000000
## 2      MAP              0.000000
## 3      PDQ              1.123590
## 4 Surface_T            + 0.000000
## 5      NDVI            + 0.000000

```

```
## 6      CEC      + 0.000000
## 7      Clay     0.000000
## 8      Soil_C   1.914403
## 9      pH       + 0.000000
## 10     PC1      0.000000
```

3.2. Multivariate models

...

```
data$QDS %<>% mutate(log10_QDS_richness = log10(QDS_richness))
data$HDS %<>% mutate(log10_HDS_richness = log10(HDS_richness))
data$DS  %<>% mutate(log10_DS_richness  = log10(DS_richness))

full_formula <- predictor_names[predictor_names != "PC1"] %>%
  {c(., paste(., "* region"))} %>%
  paste(collapse = " + ")

m_QDS_richness1 <- lm(glue("QDS_richness ~ {full_formula}"), data$QDS)
m_QDS_richness  <- lm(glue("log10_QDS_richness ~ {full_formula}"), data$QDS)
AIC(m_QDS_richness1, m_QDS_richness)
```

```
##           df      AIC
## m_QDS_richness1 21 9595.798
## m_QDS_richness  21  410.297

m_HDS_richness1 <- lm(glue("HDS_richness ~ {full_formula}"), data$HDS)
m_HDS_richness  <- lm(glue("log10_HDS_richness ~ {full_formula}"), data$HDS)
AIC(m_HDS_richness1, m_HDS_richness)
```

```
##           df      AIC
## m_HDS_richness1 21 3006.9797
## m_HDS_richness  21  122.1338

m_DS_richness1 <- lm(glue("DS_richness ~ {full_formula}"), data$DS)
m_DS_richness  <- lm(glue("log10_DS_richness ~ {full_formula}"), data$DS)
AIC(m_DS_richness1, m_DS_richness)
```

```
##           df      AIC
## m_DS_richness1 21 802.1799
## m_DS_richness  21 -20.9595

m_QDS_richness %<>% step(direction = "backward", trace = 0)
m_HDS_richness %<>% step(direction = "backward", trace = 0)
m_DS_richness  %<>% step(direction = "backward", trace = 0)
```

```
# Reparameterise models to {*}:regionGCFR & {*}:regionSWAFR
# a.o.t. {*}*region, so that the figure of the effects actually represents
# each region, not the baseline (GCFR) and "relative SWAFR"
# (and that would cause inconsistencies too when there is no interaction with
# region term for a roughness variable).
reparameterise <- function(m) {
  response <- colnames(m$model)[[1]]
  data <- data %$% {
    if (response == "log10_QDS_richness") QDS
    else if (response == "log10_HDS_richness") HDS
    else if (response == "log10_DS_richness") DS
  }
  preds_w_interactions <- m %$%
  coefficients %>%
  names() %>%
```

```

magrittr::extract(str_which(., ":regionSWAFR"))
reparameterisation <- preds_w_interactions %<>%
  str_remove(":regionSWAFR") %>%
  {glue("-{.}")} %>%
  paste(collapse = " ")
update(m,
  formula = glue(". ~ . {reparameterisation}"),
  data = data
)
}
# Test:
# a <- m_HDS_richness
# b <- reparameterise(m_HDS_richness)
# AIC(a, b) # same model! :)
m_QDS_richness %<>% reparameterise()
m_HDS_richness %<>% reparameterise()
m_DS_richness %<>% reparameterise()

models <- list(
  QDS_richness = m_QDS_richness,
  HDS_richness = m_HDS_richness,
  DS_richness = m_DS_richness
)
models_summary <- models %>%
  map_df(.id = "response", tidy, conf.int = TRUE) %>%
  dplyr::select(-std.error, -statistic) %>%
  filter(term != "(Intercept)")

models_R2 <- models %>%
  map_df(.id = "response", glance) %>%
  dplyr::select(response, adj.r.squared)

models_summary %<>% full_join(models_R2)

glance(m_QDS_richness)

## # A tibble: 1 x 11
##   r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC
##   <dbl>      <dbl> <dbl>    <dbl>    <dbl> <int>  <dbl> <dbl> <dbl>
## 1    0.212      0.203 0.320    22.7 7.94e-31     9  -186.  392.  438.
## # ... with 2 more variables: deviance <dbl>, df.residual <int>
glance(m_HDS_richness)

## # A tibble: 1 x 11
##   r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC
##   <dbl>      <dbl> <dbl>    <dbl>    <dbl> <int>  <dbl> <dbl> <dbl>
## 1    0.240      0.208 0.307     7.55 9.61e-9     9  -43.2  106.  139.
## # ... with 2 more variables: deviance <dbl>, df.residual <int>
glance(m_DS_richness)

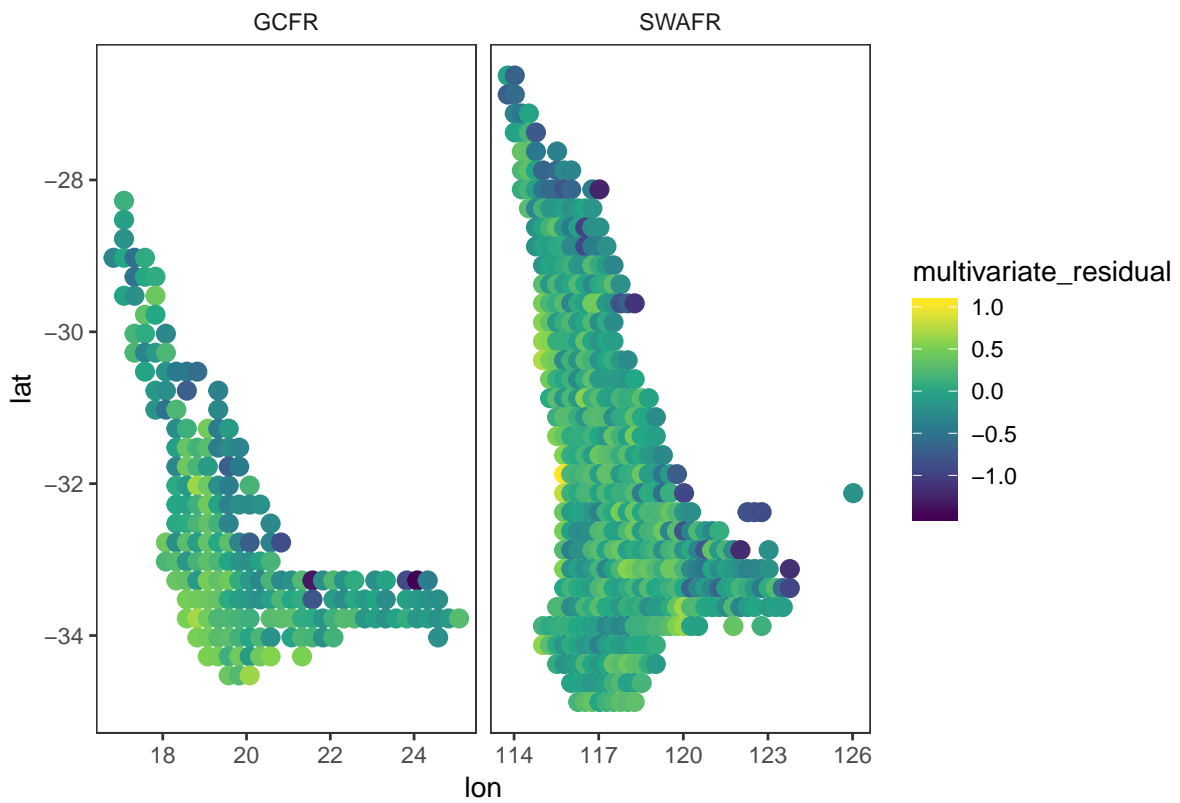
## # A tibble: 1 x 11
##   r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC
##   <dbl>      <dbl> <dbl>    <dbl>    <dbl> <int>  <dbl> <dbl> <dbl>
## 1    0.827      0.778 0.153    16.9 1.41e-11    12   30.1 -34.2 -9.06
## # ... with 2 more variables: deviance <dbl>, df.residual <int>
data$QDS$multivariate_residual <- m_QDS_richness$residuals
data$HDS$multivariate_residual <- m_HDS_richness$residuals

```

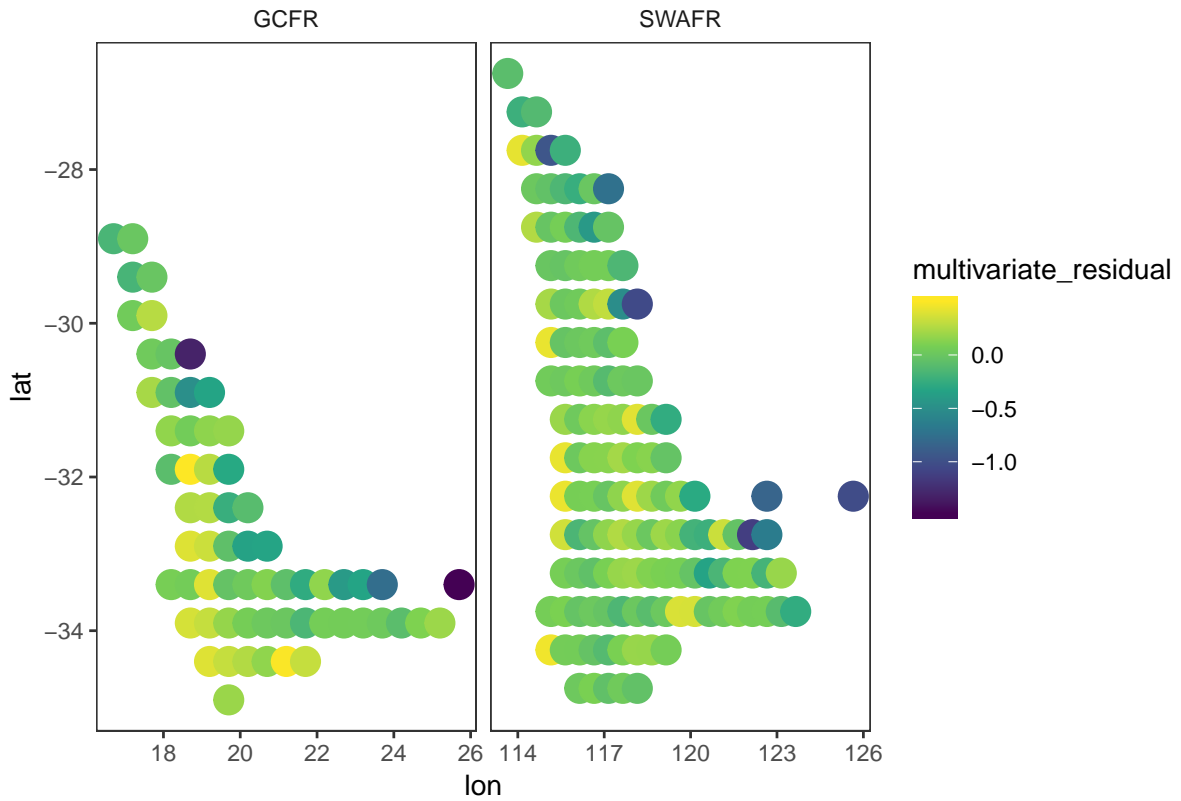


```
data$DS$multivariate_residual <- m_DS_richness$residuals
```

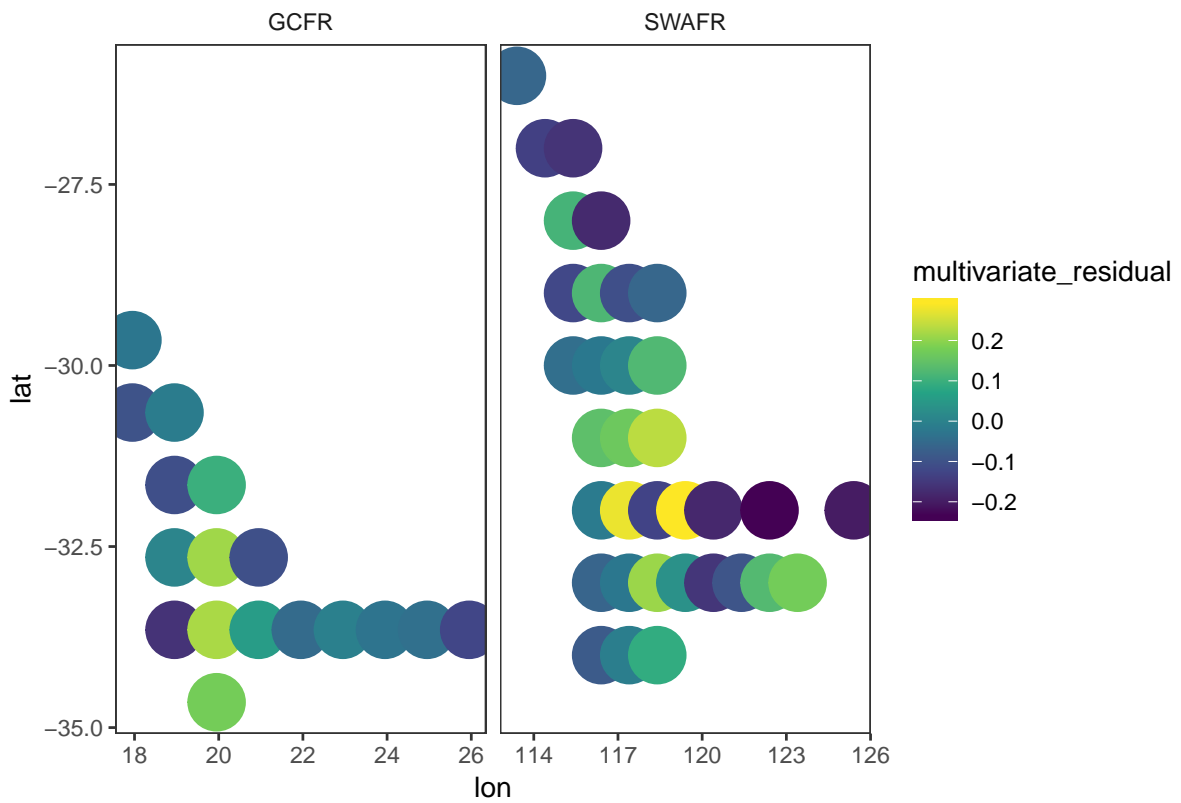
```
ggplot(data$QDS, aes(lon, lat, colour = multivariate_residual)) +  
  geom_point(size = 3) +  
  facet_grid(~region, scales = "free_x") +  
  scale_colour_viridis_c()
```



```
ggplot(data$HDS, aes(lon, lat, colour = multivariate_residual)) +  
  geom_point(size = 5) +  
  facet_grid(~region, scales = "free_x") +  
  scale_colour_viridis_c()
```



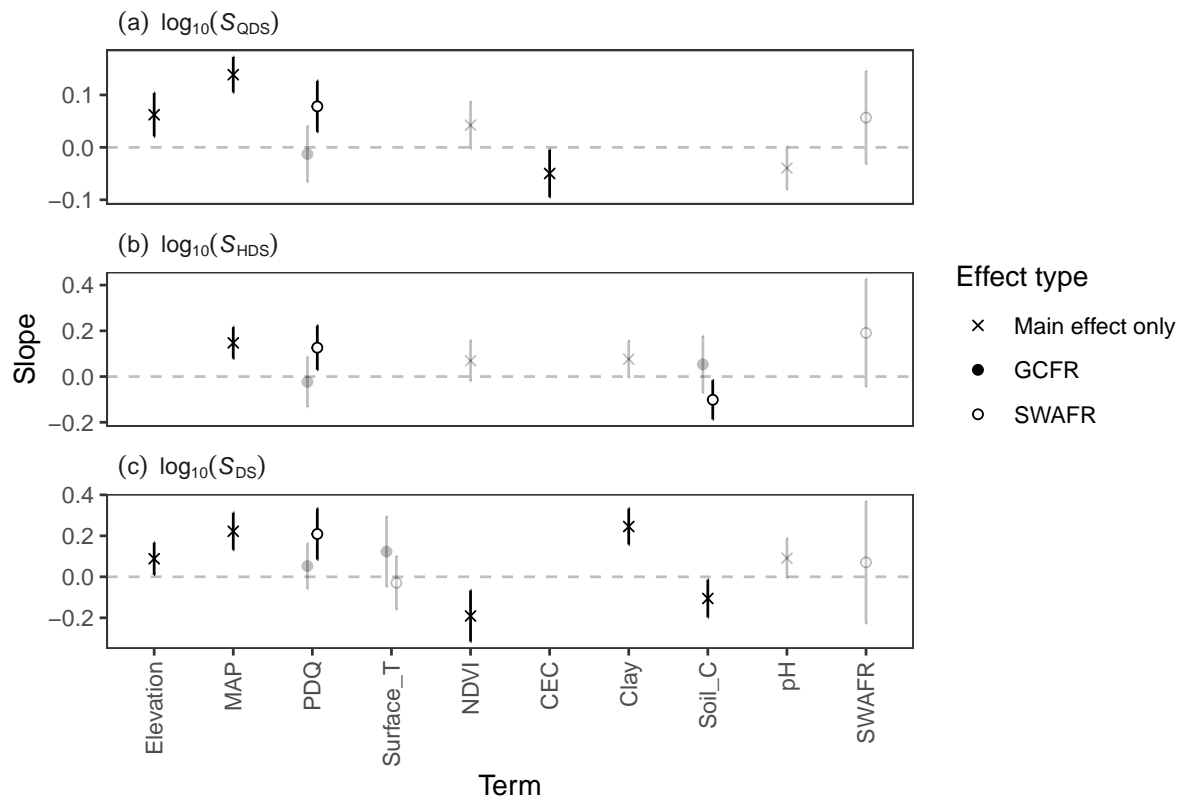
```
ggplot(data$DS, aes(lon, lat, colour = multivariate_residual)) +  
  geom_point(size = 10) +  
  facet_grid(~region, scales = "free_x") +  
  scale_colour_viridis_c()
```



```

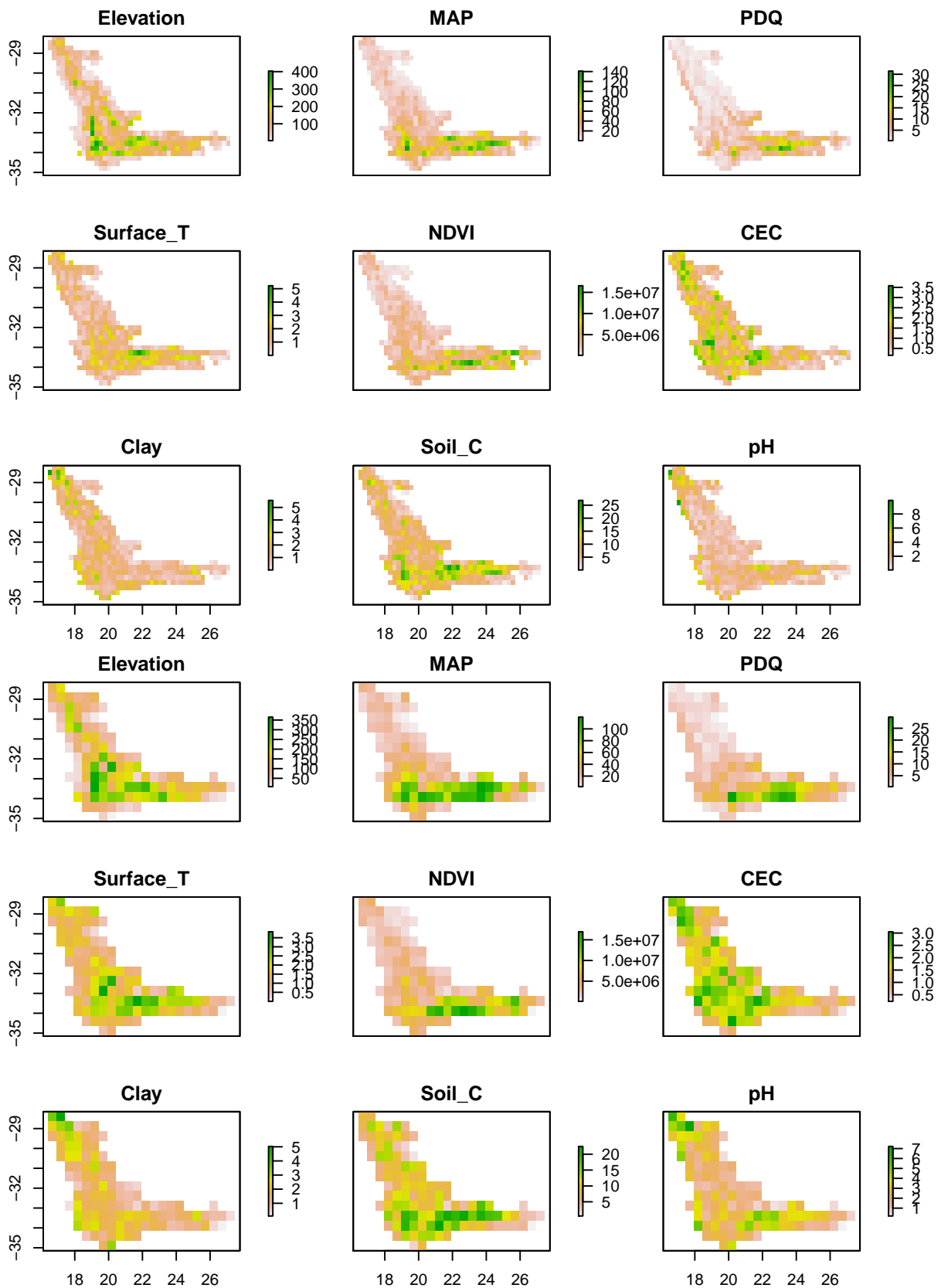
models_summary_for_plot <- models_summary %>%
  mutate(
    response = case_when(
      response == "QDS_richness" ~ "(a)~log[10](italic(S)[QDS])",
      response == "HDS_richness" ~ "(b)~log[10](italic(S)[HDS])",
      response == "DS_richness" ~ "(c)~log[10](italic(S)[DS])"
    ),
    region =
      case_when(
        str_detect(term, "regionSWAFR") ~ "SWAFR",
        str_detect(term, "regionGCFR") ~ "GCFR",
        TRUE ~ "Main effect only"
      ) %>%
    factor(levels = c("Main effect only", "GCFR", "SWAFR")),
    term = term %>%
    str_replace_all("\\.", " ") %>%
    str_remove_all("regionSWAFR:") %>%
    str_remove_all("regionGCFR:") %>%
    str_replace_all("regionSWAFR", "SWAFR") %>%
    factor(levels = c(str_replace_all(var_names, " ", "_"), "SWAFR")),
    sig = (p.value < 0.05)
  )
ggplot(models_summary_for_plot) +
  aes(
    term, estimate,
    fill = region, group = region, shape = region,
    alpha = sig
  ) +
  geom_hline(yintercept = 0, linetype = "dashed", colour = "grey75") +
  geom_errorbar(
    aes(ymin = conf.low, ymax = conf.high),
    position = position_dodge(width = 0.25),
    width = 0
  ) +
  geom_point(position = position_dodge(width = 0.25)) +
  labs(x = "Term", y = "Slope") +
  scale_fill_manual(values = c(NA, "black", "white")) +
  scale_shape_manual(values = c(4, 21, 21)) +
  scale_alpha_manual(values = c(0.25, 1)) +
  facet_wrap(~response, nrow = 3, scales = "free_y", labeller = label_parsed) +
  guides(
    fill = FALSE,
    shape = guide_legend(
      title = "Effect type",
      override.aes = list(fill = c(NA, "black", "white"))
    ),
    alpha = FALSE
  ) +
  theme(
    axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5),
    strip.text.x = element_text(angle = 0, hjust = 0)
  )

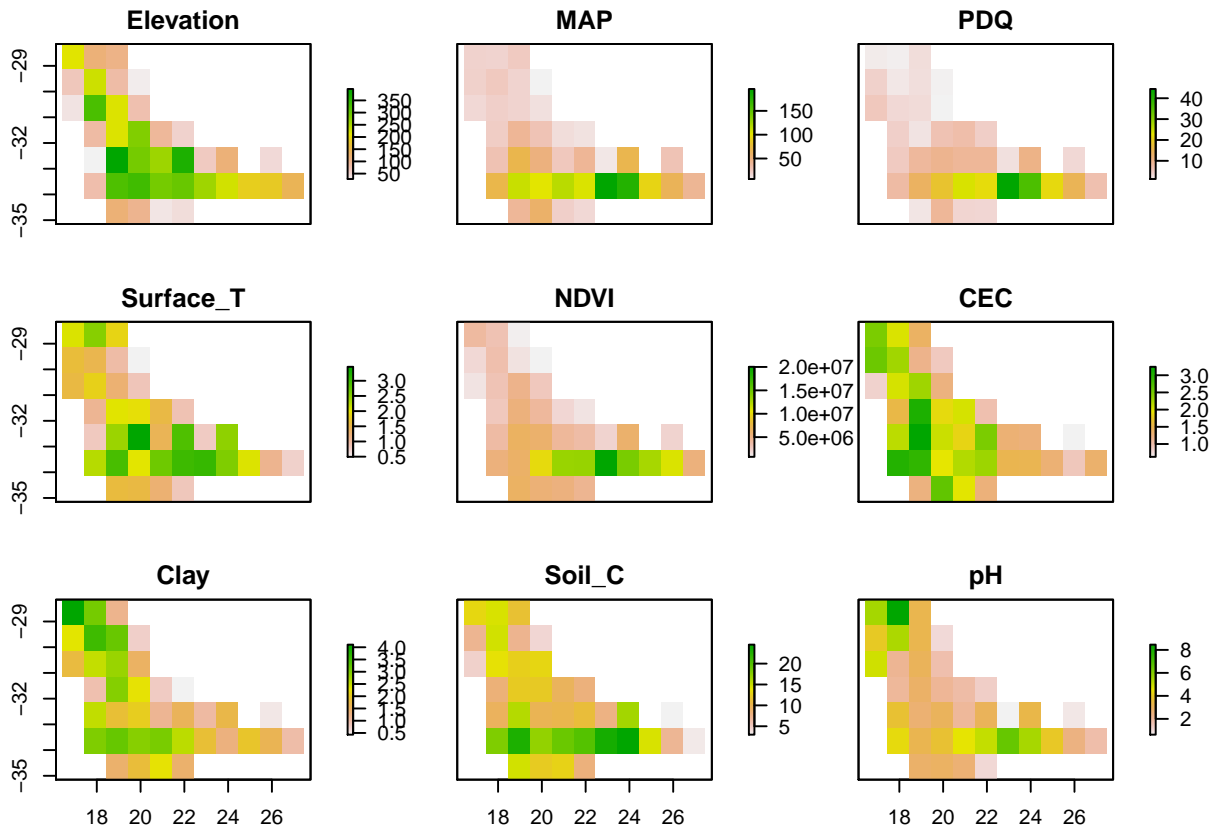
```



Repeat 1.–3. with EH_2

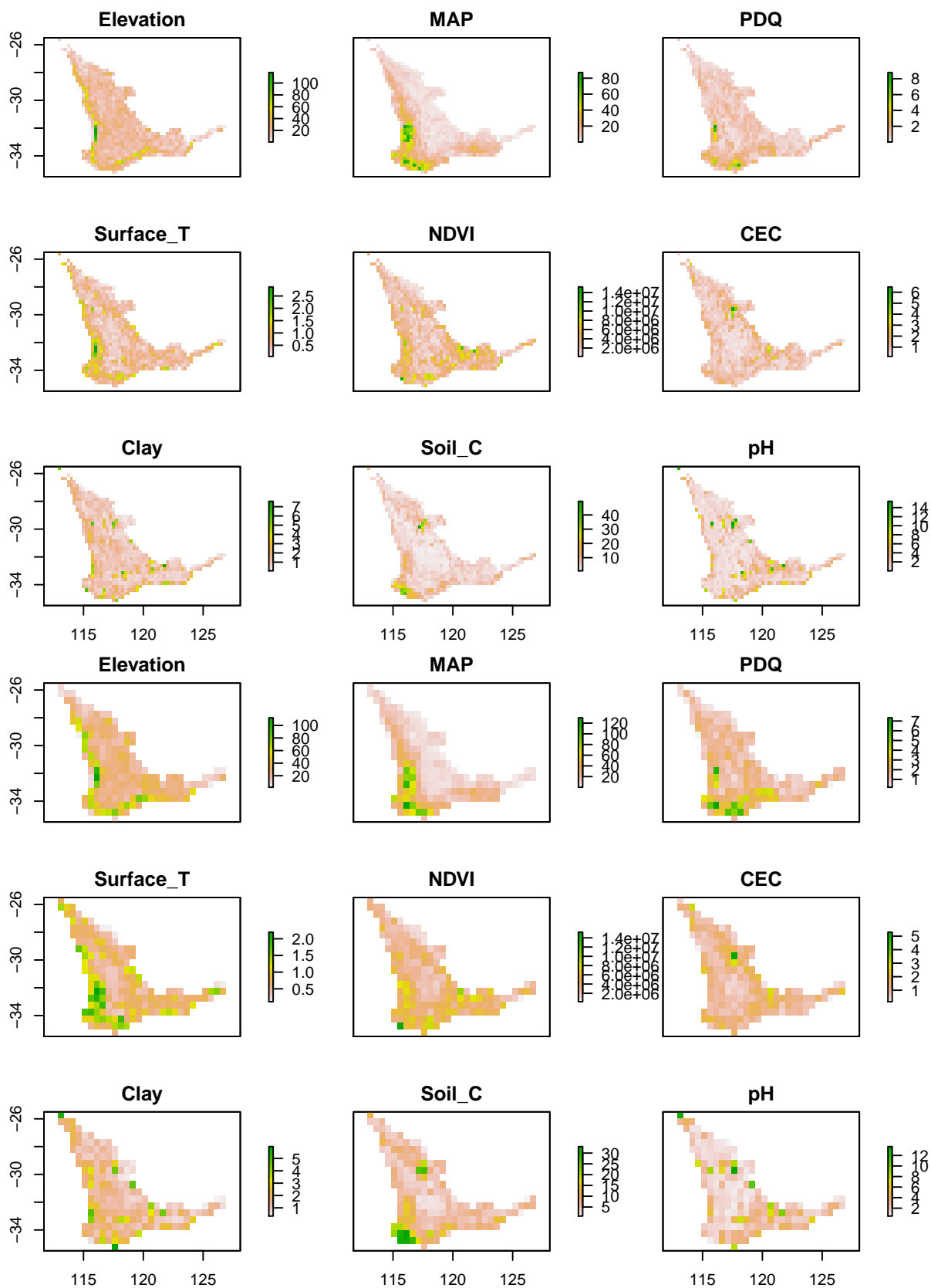
```
GCFR_heterogeneity2 <- list(
  QDS = aggregate(GCFR_variables, fact = 5, fun = sd),
  HDS = aggregate(GCFR_variables, fact = 10, fun = sd),
  DS = aggregate(GCFR_variables, fact = 20, fun = sd)
)
map(GCFR_heterogeneity2, plot)
```

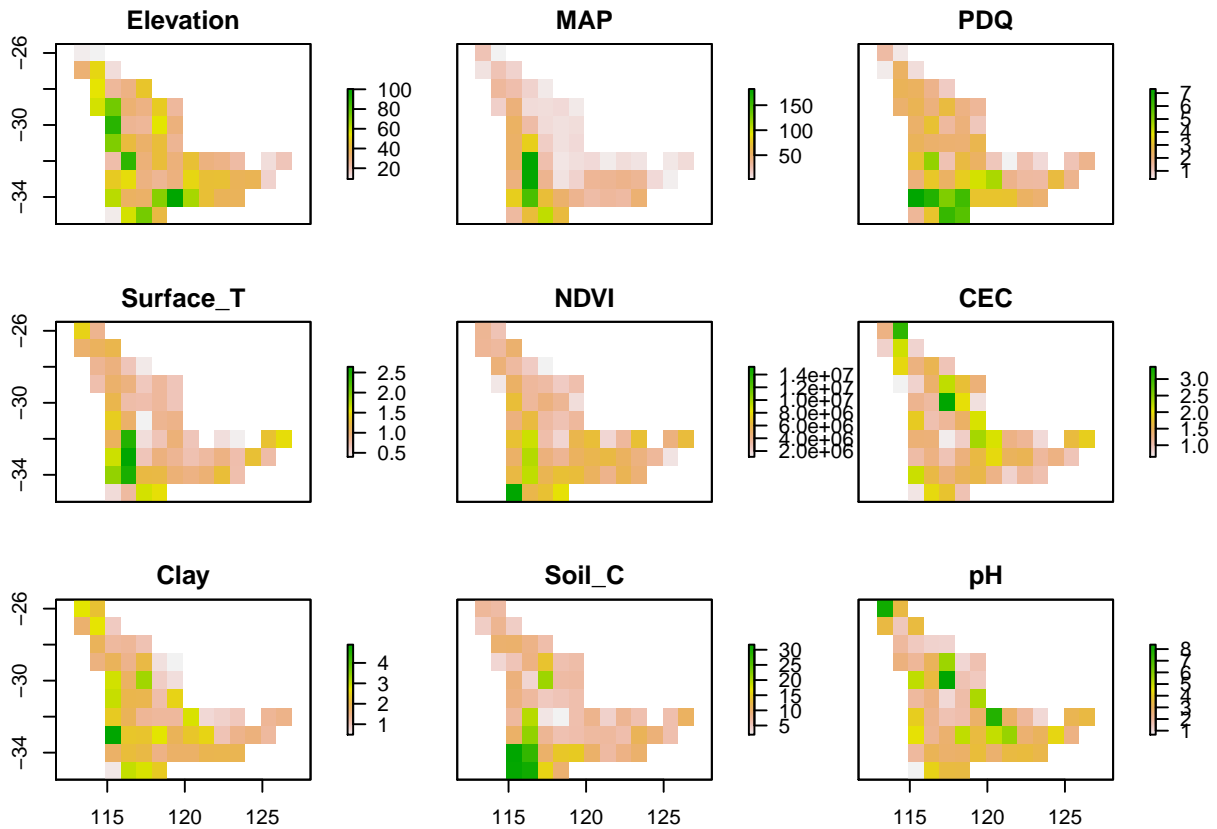




```
## $QDS
## NULL
##
## $HDS
## NULL
##
## $DS
## NULL

SWAFR_heterogeneity2 <- list(
  QDS = aggregate(SWAFR_variables, fact = 5, fun = sd),
  HDS = aggregate(SWAFR_variables, fact = 10, fun = sd),
  DS = aggregate(SWAFR_variables, fact = 20, fun = sd)
)
map(SWAFR_heterogeneity2, plot)
```





```
## $QDS
## NULL
##
## $HDS
## NULL
##
## $DS
## NULL

# Join regions' datasets
heterogeneity2 <- map2(GCFR_heterogeneity2, SWAFR_heterogeneity2,
  ~ na.exclude(rbind(
    cbind(region = "GCFR", as.data.frame(log10(.x))),
    cbind(region = "SWAFR", as.data.frame(log10(.y)))
  ))
)
```

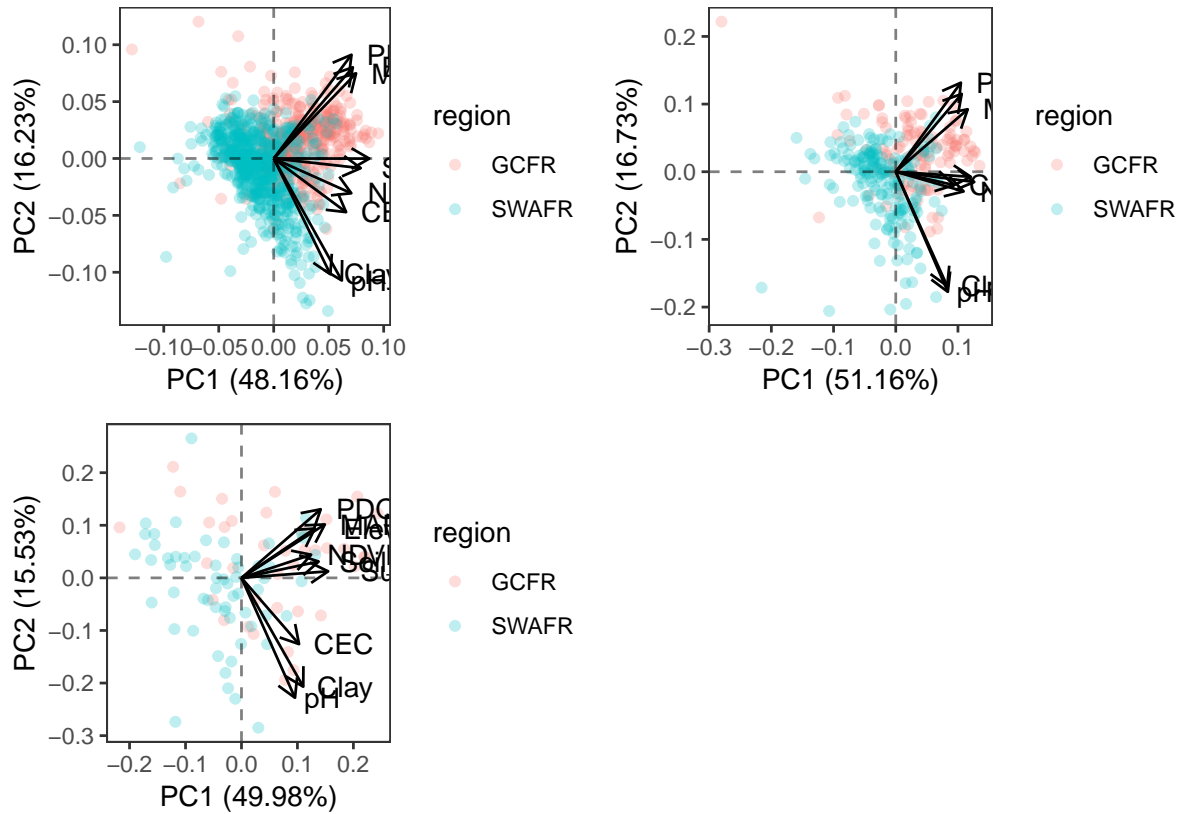
```
heterogeneity2_PCAs <- map(heterogeneity2,
  ~prcomp(.x[, -1], center = TRUE, scale. = TRUE)
)
map(heterogeneity2_PCAs, summary)
```

```
## $QDS
## Importance of components:
##
##          PC1      PC2      PC3      PC4      PC5      PC6
## Standard deviation    2.0820  1.2085  0.86871  0.80698  0.72347  0.63164
## Proportion of Variance 0.4816  0.1623  0.08385  0.07236  0.05816  0.04433
## Cumulative Proportion 0.4816  0.6439  0.72774  0.80009  0.85825  0.90258
##
##          PC7      PC8      PC9
## Standard deviation    0.59160  0.54606  0.4781
## Proportion of Variance 0.03889  0.03313  0.0254
## Cumulative Proportion 0.94147  0.97460  1.0000
```



```
##
## $HDS
## Importance of components:
##           PC1      PC2      PC3      PC4      PC5      PC6
## Standard deviation  2.1457 1.2270 0.89003 0.76637 0.66199 0.60262
## Proportion of Variance 0.5116 0.1673 0.08802 0.06526 0.04869 0.04035
## Cumulative Proportion 0.5116 0.6788 0.76687 0.83212 0.88082 0.92117
##           PC7      PC8      PC9
## Standard deviation  0.5450 0.49093 0.41413
## Proportion of Variance 0.0330 0.02678 0.01906
## Cumulative Proportion 0.9542 0.98094 1.00000
##
## $DS
## Importance of components:
##           PC1      PC2      PC3      PC4      PC5      PC6      PC7
## Standard deviation  2.1210 1.1822 0.9973 0.8469 0.63505 0.6206 0.50295
## Proportion of Variance 0.4999 0.1553 0.1105 0.0797 0.04481 0.0428 0.02811
## Cumulative Proportion 0.4999 0.6551 0.7657 0.8454 0.89016 0.9330 0.96106
##           PC8      PC9
## Standard deviation  0.47383 0.35484
## Proportion of Variance 0.02495 0.01399
## Cumulative Proportion 0.98601 1.00000

# Force PC1 scores to be positive if all vars rotations are negative
heterogeneity2_PCAs %<>% map(function(PCA) {
  if (all(PCA$rotation[, 1] <= 0)) {
    message("Multiplying this one by -1")
    PCA$rotation[, 1] %<>% multiply_by(-1)
    PCA$x[, 1] %<>% multiply_by(-1)
  }
  PCA
})
plot_grid(plotlist = map2(
  .x = heterogeneity2_PCAs,
  .y = heterogeneity2,
  .f =
    ~ autoplot(.x, data = .y, colour = "region",
      alpha = 0.25,
      loadings = TRUE, loadings.colour = "black",
      loadings.label = TRUE, loadings.label.colour = "black",
      loadings.label.hjust = -0.25
    ) +
    ggtitle(unique(.y$scale)) +
    geom_hline(yintercept = 0, linetype = "dashed", alpha = 0.5) +
    geom_vline(xintercept = 0, linetype = "dashed", alpha = 0.5)
))
```



```
PC1s2 <- map(heterogeneity2_PCAs, ~tibble(PC1 = .x$x[, 1]))
heterogeneity2 %<>% map2(PC1s2, ~cbind(.x, .y))
CLES2_results <- map2_dfr(
  .x = heterogeneity2 %>%
    map(filter, region == "GCFR") %>%
    map(dplyr::select, -region),
  .y = heterogeneity2 %>%
    map(filter, region == "SWAFR") %>%
    map(dplyr::select, -region),
  .id = "scale", # for every spatial scale,
  ~ map2_df(
    .x = .x,
    .y = .y,
    .id = "variable", # for every variable in each region,
    ~ tibble(
      CLES_value = CLES(.y, .x), # calculate the CLES,
      U_test = wilcox.test(.x, .y, conf.int = TRUE) %>% # & Mann-Whitney U-test
        tidy() %>%
        list()
    )
  )
)
CLES2_results %<>% mutate(
  variable = factor(variable, levels = var_names %>%
    str_replace_all(" ", "_") %>%
    c("PC1")
  ),
  scale = case_when(
    scale == "point1" ~ 0.10,
    scale == "QDS" ~ 0.25,
    scale == "HDS" ~ 0.50,
  )
)
```

```

    scale == "DS" ~ 1.00
  ),
  diff = map_dbl(U_test, "estimate"),
  P_U = map_dbl(U_test, "p.value"),
  U_low = map_dbl(U_test, "conf.low"),
  U_upp = map_dbl(U_test, "conf.high")
)
CLES2_results

## # A tibble: 30 x 8
##   scale variable CLES_value U_test      diff      P_U    U_low  U_upp
##   <dbl> <fct>      <dbl> <list>    <dbl>    <dbl>  <dbl>  <dbl>
## 1 0.25 Elevation    0.930 <tibble [1 x~ 0.652 8.05e-114 0.613 0.690
## 2 0.25 MAP          0.760 <tibble [1 x~ 0.415 5.90e- 43 0.363 0.467
## 3 0.25 PDQ          0.854 <tibble [1 x~ 0.504 5.74e- 78 0.458 0.551
## 4 0.25 Surface_T    0.809 <tibble [1 x~ 0.327 6.02e- 60 0.293 0.361
## 5 0.25 NDVI         0.545 <tibble [1 x~ 0.0460 1.85e- 2 0.00748 0.0844
## 6 0.25 CEC          0.689 <tibble [1 x~ 0.157 1.93e- 23 0.128 0.186
## 7 0.25 Clay         0.597 <tibble [1 x~ 0.0827 2.76e- 7 0.0522 0.113
## 8 0.25 Soil_C       0.726 <tibble [1 x~ 0.252 6.53e- 33 0.214 0.288
## 9 0.25 pH           0.655 <tibble [1 x~ 0.179 3.02e- 16 0.139 0.219
## 10 0.25 PC1         0.826 <tibble [1 x~ 2.38 2.06e- 66 2.15 2.60
## # ...with 20 more rows

CLES2_models <- CLES2_results %>%
  split(.$variable) %>%
  map(~lm(CLES_value ~ scale, .x))
CLES2_models$Elevation

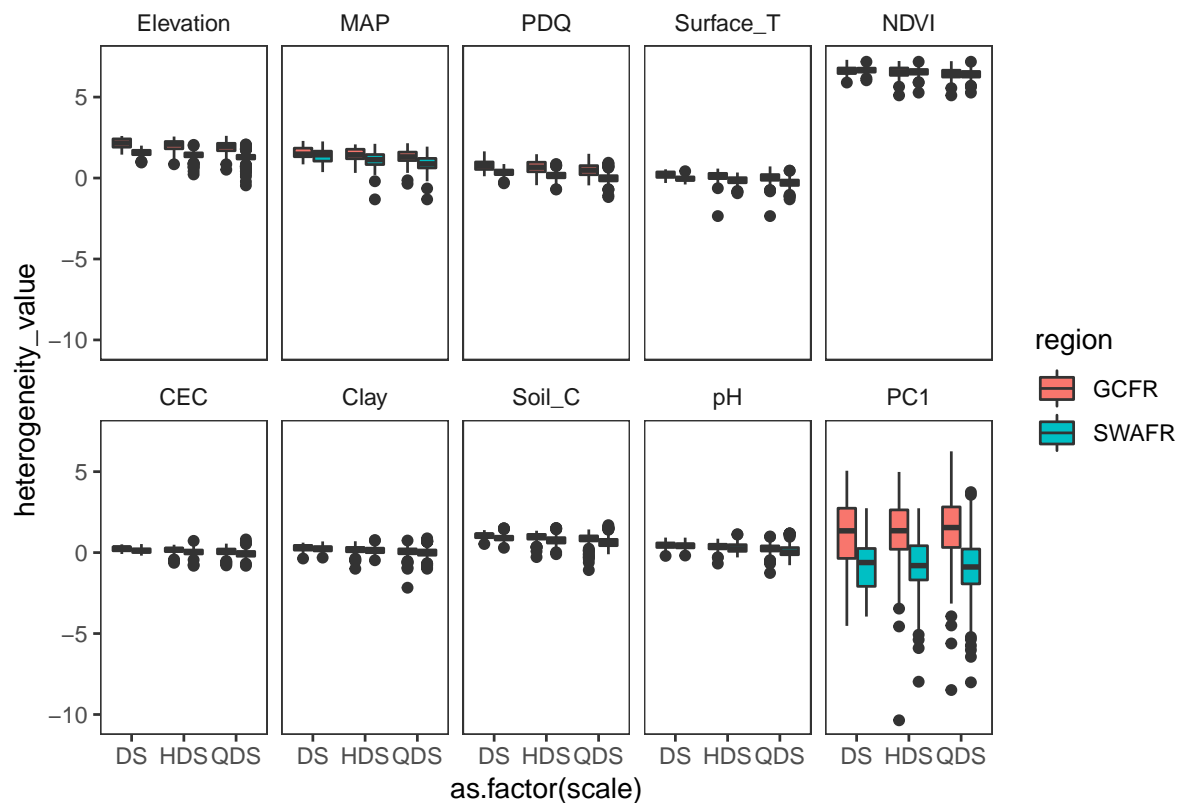
##
## Call:
## lm(formula = CLES_value ~ scale, data = .x)
##
## Coefficients:
## (Intercept)      scale
##    0.92611    -0.01698

# Summarise those models
CLES2_model_summaries <- CLES2_models %>%
  map_df(.id = "variable", tidy) %>%
  filter(term != "(Intercept)") %>%
  mutate(sig = case_when(
    p.value <= 0.05 ~ "*",
    p.value <= 0.10 ~ ".",
    TRUE ~ " ")
  ) %>%
  mutate(variable = factor(variable, levels = var_names %>%
    str_replace_all(" ", "_") %>%
    c("PC1")
  )
  ) %>%
  mutate_if(is.numeric, round, digits = 3) %>%
  dplyr::select(variable, estimate, p.value, sig)
CLES2_model_summaries

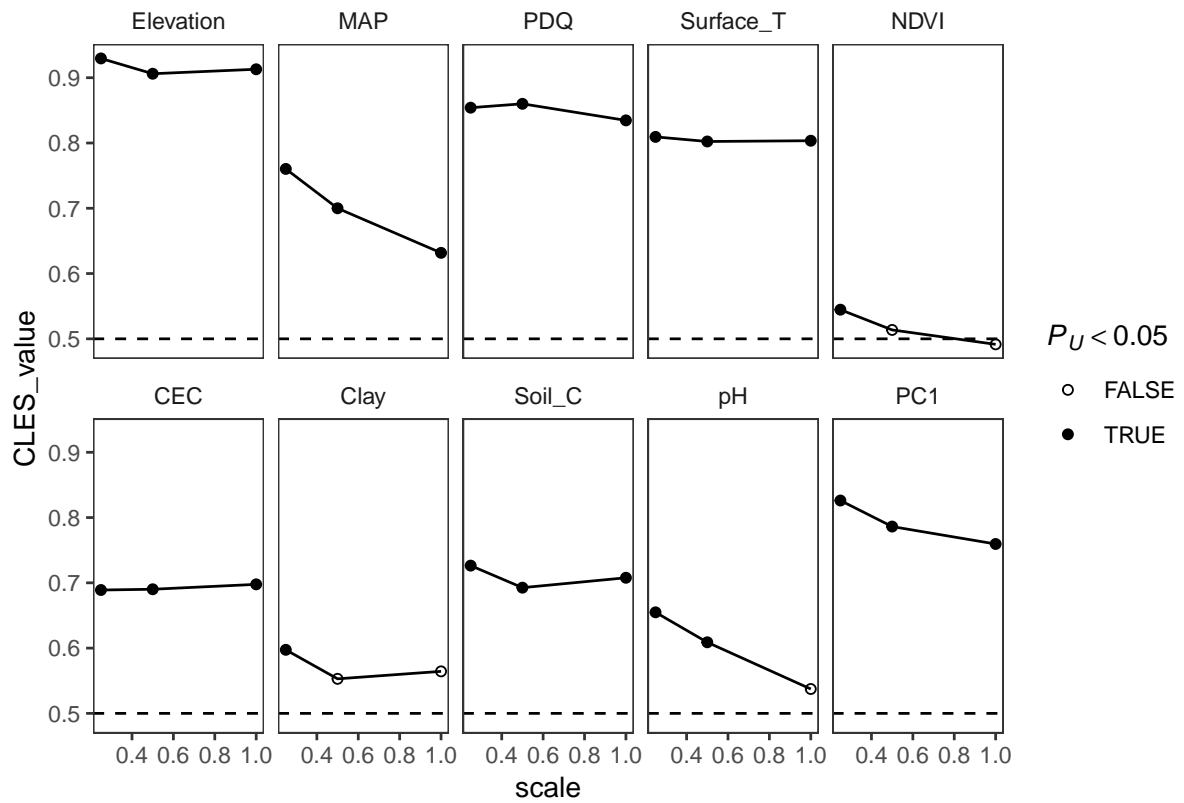
## # A tibble: 10 x 4
##   variable estimate p.value sig
##   <fct>      <dbl> <dbl> <chr>
## 1 Elevation  -0.017  0.638 " "
## 2 MAP        -0.166  0.098 .
## 3 PDQ        -0.03   0.355 " "
```

```
## 4 Surface_T -0.006 0.553 " "
## 5 NDVI -0.067 0.183 " "
## 6 CEC 0.012 0.134 " "
## 7 Clay -0.034 0.614 " "
## 8 Soil_C -0.017 0.75 " "
## 9 pH -0.155 0.041 *
## 10 PC1 -0.084 0.194 " "
```

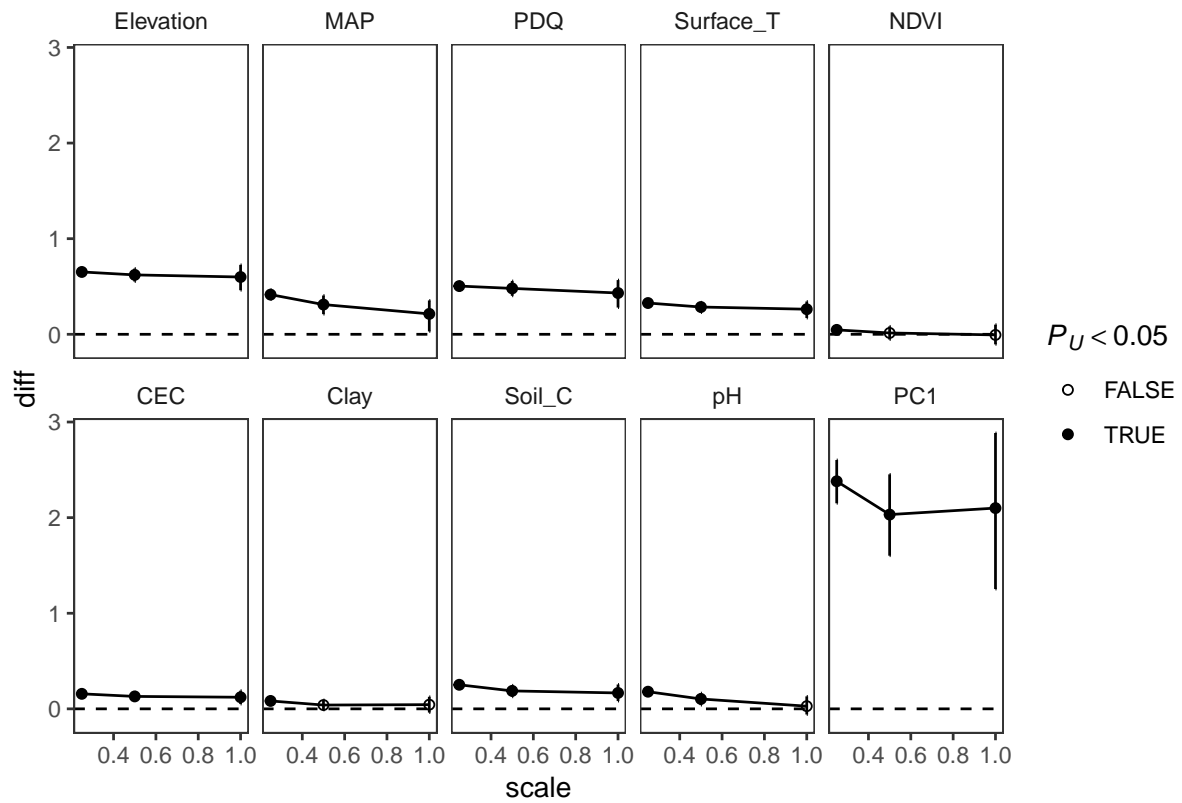
```
heterogeneity2 %>% #heterogeneity_df %>%
  bind_rows(.id = "scale") %>%
  gather(
    variable, heterogeneity_value,
    -region, -scale#, -lon, -lat
  ) %>%
  mutate(variable = factor(variable, levels = var_names %>%
    str_replace_all(" ", "_") %>%
    c("PC1")
  )) %>%
  ggplot(aes(as.factor(scale), heterogeneity_value, fill = region)) +
    geom_boxplot() +
    facet_wrap(~variable, nrow = 2)
```



```
ggplot(CLES2_results) +
  aes(scale, CLES_value, group = variable) +
  geom_hline(yintercept = 0.5, lty = "dashed") +
  geom_line() +
  geom_point(aes(shape = P_U < 0.05)) +
  scale_shape_manual(name = bquote(italic("P"["U"]) < 0.05), values = c(1, 19)) +
  facet_wrap(~variable, nrow = 2)
```



```
ggplot(CLES2_results) +
  aes(scale, diff, group = variable) +
  geom_hline(yintercept = 0.0, lty = "dashed") +
  geom_line() +
  geom_errorbar(aes(ymin = U_low, ymax = U_upp), width = 0) +
  geom_point(aes(shape = P_U < 0.05)) +
  scale_shape_manual(name = bquote(italic("P"["U"]) < 0.05), values = c(1, 19)) +
  facet_wrap(~variable, nrow = 2)
```



```

heterogeneity2_w_coords <- map2(GCFR_heterogeneity2, SWAFR_heterogeneity2,
  ~ na.exclude(rbind(
    cbind(region = "GCFR", raster2df(.x)),
    cbind(region = "SWAFR", raster2df(.y))
  ))
)
heterogeneity2 <- map2(heterogeneity2, heterogeneity2_w_coords, full_join)

heterogeneity2$QDS$QDS <- heterogeneity2$QDS %$%
  SpatialPoints(
    coords = data.frame(x = lon, y = lat),
    proj4string = crs(Larsen_grid)
  ) %over%
  Larsen_grid %>%
  pull(qdgc)

heterogeneity2$HDS$HDS <- heterogeneity2$HDS %$%
  SpatialPoints(
    coords = data.frame(x = lon, y = lat),
    proj4string = crs(Larsen_grid)
  ) %over%
  Larsen_grid %>%
  pull(hdgc)

heterogeneity2$DS$DS <- heterogeneity2$DS %$%
  SpatialPoints(
    coords = data.frame(x = lon, y = lat),
    proj4string = crs(Larsen_grid)
  ) %over%
  Larsen_grid %>%
  pull(dgc)

```

```

data2 <- heterogeneity2
data2$QDS %<>%
  as_tibble() %>%
  full_join(QDS_richness) %>%
  na.exclude() %>%
  filter(n_EDS == 4)
data2$HDS %<>%
  as_tibble() %>%
  full_join(mean_QDS_richness) %>%
  full_join(HDS_richness) %>%
  na.exclude() %>%
  filter(n_QDS == 4) %>%
  mutate(
    QDS_turnover      = HDS_richness - mean_QDS_richness,
    QDS_turnover_prop = QDS_turnover / HDS_richness
  )
data2$DS %<>%
  as_tibble() %>%
  full_join(mean_HDS_richness) %>%
  full_join(DS_richness) %>%
  na.exclude() %>%
  filter(n_HDS == 4) %>%
  mutate(
    HDS_turnover      = DS_richness - mean_HDS_richness,
    HDS_turnover_prop = HDS_turnover / DS_richness
  )

```

Univariate models

```

m1 <- lm(QDS_richness ~ PC1, data2$QDS)
m2 <- lm(log(QDS_richness) ~ PC1, data2$QDS)
m3 <- lm(log10(QDS_richness) ~ PC1, data2$QDS)
AIC(m1, m2, m3)

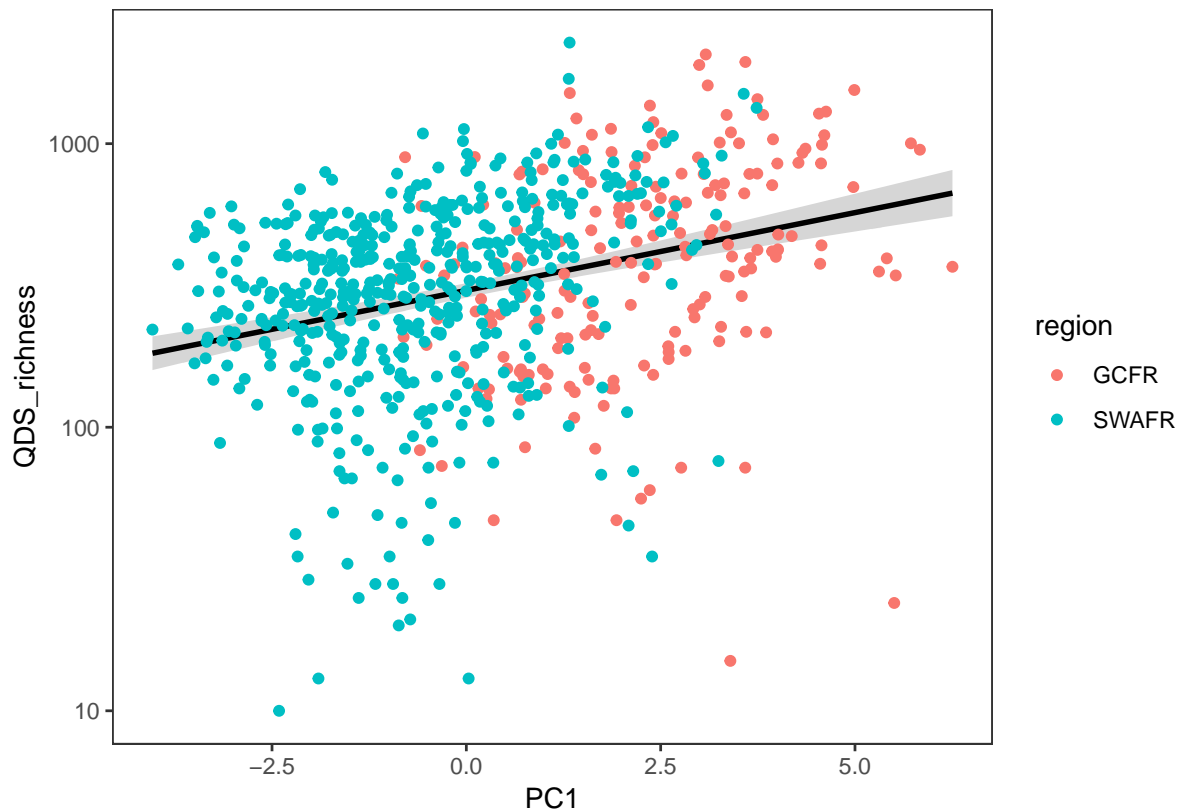
##      df      AIC
## m1   3 9829.8053
## m2   3 1645.6538
## m3   3  484.6806

# Choose m3
m4 <- lm(log10(QDS_richness) ~ PC1 + region, data2$QDS)
m5 <- lm(log10(QDS_richness) ~ PC1 * region, data2$QDS)
AIC(m3, m4, m5)

##      df      AIC
## m3   3  484.6806
## m4   4  482.9581
## m5   5  484.3936

# Choose m3
ggplot(data2$QDS, aes(PC1, QDS_richness)) +
  geom_smooth(method = lm, colour = "black") +
  geom_point(aes(colour = region)) +
  scale_y_log10()

```



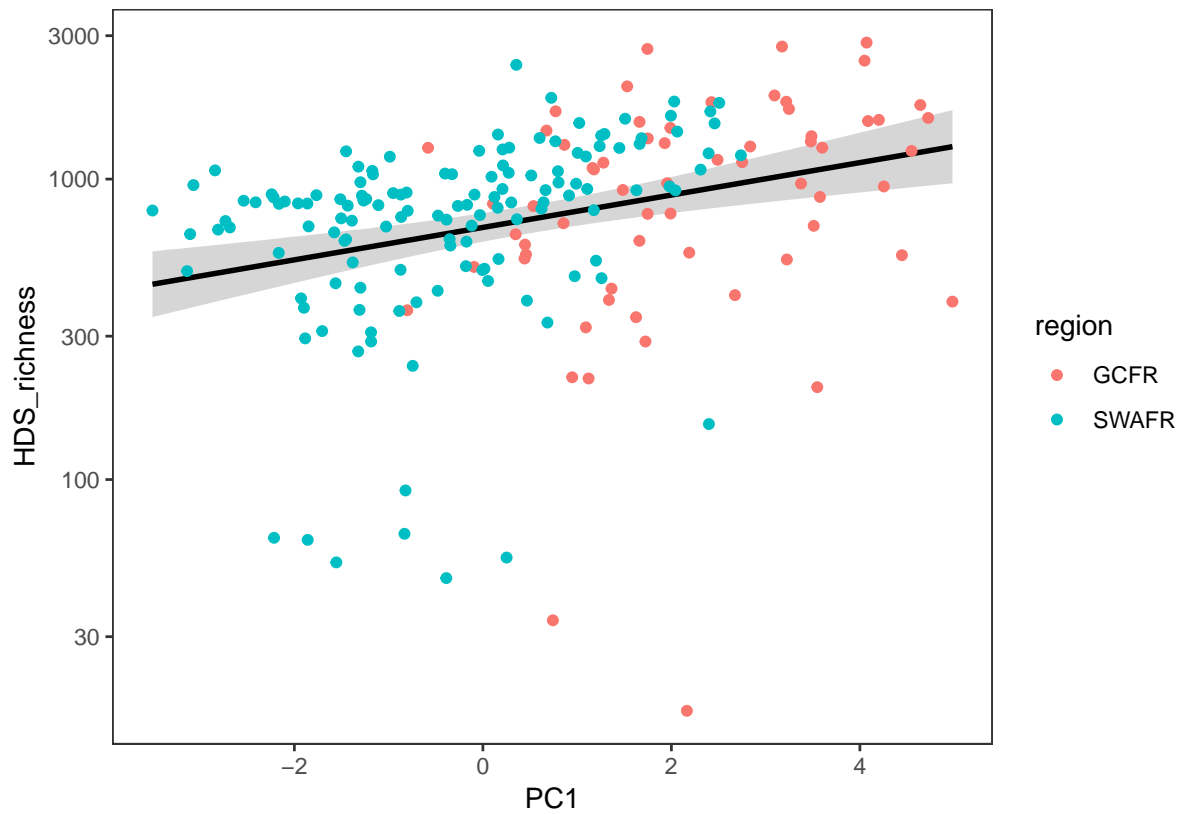
```
m1 <- lm(HDS_richness ~ PC1, data2$HDS)
m2 <- lm(log(HDS_richness) ~ PC1, data2$HDS)
m3 <- lm(log10(HDS_richness) ~ PC1, data2$HDS)
AIC(m1, m2, m3)

##      df      AIC
## m1   3 3030.2221
## m2   3  463.3749
## m3   3  129.7619

# Choose m3
m4 <- lm(log10(HDS_richness) ~ PC1 + region, data2$HDS)
m5 <- lm(log10(HDS_richness) ~ PC1 * region, data2$HDS)
AIC(m3, m4, m5)

##      df      AIC
## m3   3  129.7619
## m4   4  128.4571
## m5   5  130.4314

# Choose m3
ggplot(data2$HDS, aes(PC1, HDS_richness)) +
  geom_smooth(method = lm, colour = "black") +
  geom_point(aes(colour = region)) +
  scale_y_log10()
```

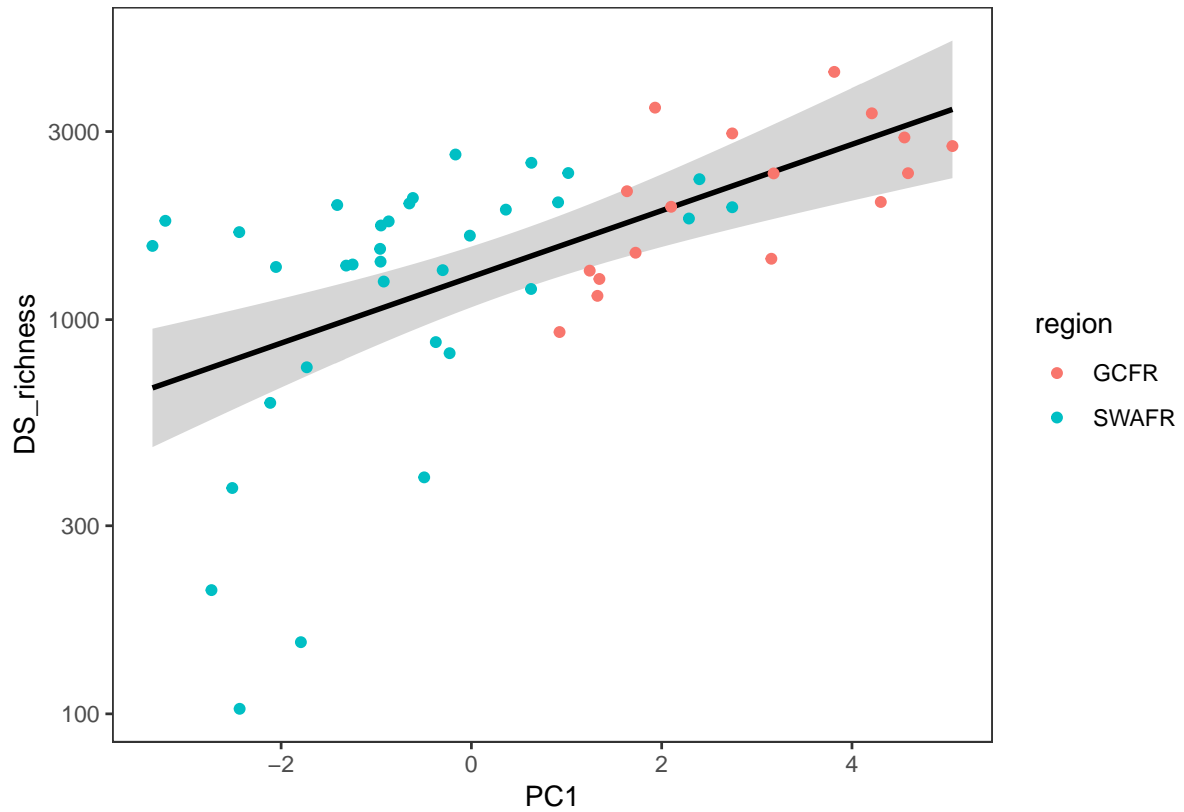
```
m1 <- lm(DS_richness ~ PC1, data2$DS)
m2 <- lm(log(DS_richness) ~ PC1, data2$DS)
m3 <- lm(log10(DS_richness) ~ PC1, data2$DS)
AIC(m1, m2, m3)

##      df      AIC
## m1   3 810.38969
## m2   3  99.60989
## m3   3  14.53858

# Choose m3
m4 <- lm(log10(DS_richness) ~ PC1 + region, data2$DS)
m5 <- lm(log10(DS_richness) ~ PC1 * region, data2$DS)
AIC(m3, m4, m5)

##      df      AIC
## m3   3 14.53858
## m4   4 15.58112
## m5   5 17.47984

# Choose m2
ggplot(data2$DS, aes(PC1, DS_richness)) +
  geom_smooth(method = lm, colour = "black") +
  geom_point(aes(colour = region)) +
  scale_y_log10()
```



```
models_non_region <- map(predictor_names,
  ~lm(paste("log10(QDS_richness) ~", .x), data2$QDS)
)
names(models_non_region) <- predictor_names
models_add_region <- map(predictor_names,
  ~lm(paste("log10(QDS_richness) ~", .x, "+ region"), data2$QDS)
)
names(models_add_region) <- predictor_names
models_int_region <- map(predictor_names,
  ~lm(paste("log10(QDS_richness) ~", .x, "* region"), data2$QDS)
)
names(models_int_region) <- predictor_names
knitr::kable(pmap_dfr(
  .l = list(models_non_region, models_add_region, models_int_region),
  .id = "variable",
  .f = ~ AIC(..1, ..2, ..3) %>%
    mutate(
      model_rank = 1:3,
      model_type = c(" ", "+", "x")[model_rank],
      delta_AIC = AIC - min(AIC),
      best_model = (model_rank == min(model_rank[delta_AIC < 2]))
    ) %>%
  filter(best_model) %>%
  dplyr::select(-df, -AIC, -model_rank, -best_model)
))
```

variable	model_type	delta_AIC
Elevation	+	0.000000
MAP	+	0.000000
PDQ	x	0.000000
Surface_T		0.000000

variable	model_type	delta_AIC
NDVI	x	0.000000
CEC	+	0.000000
Clay	+	0.000000
Soil_C	+	0.174429
pH	+	0.000000
PC1		1.722464

```
models_non_region <- map(predictor_names,
  ~lm(paste("log10(HDS_richness) ~", .x), data2$HDS)
)
names(models_non_region) <- predictor_names
models_add_region <- map(predictor_names,
  ~lm(paste("log10(HDS_richness) ~", .x, "+ region"), data2$HDS)
)
names(models_add_region) <- predictor_names
models_int_region <- map(predictor_names,
  ~lm(paste("log10(HDS_richness) ~", .x, "* region"), data2$HDS)
)
names(models_int_region) <- predictor_names
pmap_dfr(
  .l = list(models_non_region, models_add_region, models_int_region),
  .id = "variable",
  .f = ~ AIC(..1, ..2, ..3) %>%
    mutate(
      model_rank = 1:3,
      model_type = c(" ", "+", "x")[model_rank],
      delta_AIC = AIC - min(AIC),
      best_model = (model_rank == min(model_rank[delta_AIC < 2]))
    ) %>%
  filter(best_model) %>%
  dplyr::select(-df, -AIC, -model_rank, -best_model)
)
```

```
##      variable model_type delta_AIC
## 1 Elevation      + 1.13689775
## 2      MAP              0.07858797
## 3      PDQ      x 0.00000000
## 4 Surface_T          0.00000000
## 5      NDVI          0.00000000
## 6      CEC      x 0.00000000
## 7      Clay          0.00000000
## 8      Soil_C        0.00000000
## 9      pH           0.00000000
## 10     PC1          1.30479200
```

```
models_non_region <- map(predictor_names,
  ~lm(glue("log10(DS_richness) ~", .x), data2$DS)
)
names(models_non_region) <- predictor_names
models_add_region <- map(predictor_names,
  ~lm(paste("log10(DS_richness) ~", .x, "+ region"), data2$DS)
)
names(models_add_region) <- predictor_names
models_int_region <- map(predictor_names,
  ~lm(paste("log10(DS_richness) ~", .x, "* region"), data2$DS)
)
```

```

names(models_int_region) <- predictor_names
pmap_dfr(
  .l = list(models_non_region, models_add_region, models_int_region),
  .id = "variable",
  .f = ~ AIC(..1, ..2, ..3) %>%
    mutate(
      model_rank = 1:3,
      model_type = c(" ", "+", "x")[model_rank],
      delta_AIC = AIC - min(AIC),
      best_model = (model_rank == min(model_rank[delta_AIC < 2]))
    ) %>%
  filter(best_model) %>%
  dplyr::select(-df, -AIC, -model_rank, -best_model)
)

```

```

##      variable model_type delta_AIC
## 1  Elevation              0.8578615
## 2      MAP                0.0000000
## 3      PDQ                x 0.0000000
## 4 Surface_T              0.0000000
## 5      NDVI              0.0000000
## 6      CEC                + 0.0000000
## 7      Clay              0.9074850
## 8    Soil_C              0.7398216
## 9       pH                + 0.0000000
## 10     PC1              0.0000000

```

Multivariate models

```

data2$QDS %<>% mutate(log10_QDS_richness = log10(QDS_richness))
data2$HDS %<>% mutate(log10_HDS_richness = log10(HDS_richness))
data2$DS %<>% mutate(log10_DS_richness = log10(DS_richness))

m_QDS_richness <- lm(glue("log10_QDS_richness ~ {full_formula}"), data2$QDS)
m_HDS_richness <- lm(glue("log10_HDS_richness ~ {full_formula}"), data2$HDS)
m_DS_richness <- lm(glue("log10_DS_richness ~ {full_formula}"), data2$DS)

m_QDS_richness %<>% step(direction = "backward", trace = 0)
m_HDS_richness %<>% step(direction = "backward", trace = 0)
m_DS_richness %<>% step(direction = "backward", trace = 0)

```

```

# Reparameterise models to {*}:regionGCFR & {*}:regionSWAFR
# a.o.t. {*}*region, so that the figure of the effects actually represents
# each region, not the baseline (GCFR) and "relative SWAFR"
# (and that would cause inconsistencies too when there is no interaction with
# region term for a roughness variable).
reparameterise <- function(m) {
  response <- colnames(m$model)[[1]]
  data <- data2 %$% {
    if (response == "log10_QDS_richness") QDS
    else if (response == "log10_HDS_richness") HDS
    else if (response == "log10_DS_richness") DS
  }
  preds_w_interactions <- m %$%
  coefficients %>%
  names() %>%
  magrittr::extract(str_which(., ":regionSWAFR"))
  reparameterisation <- preds_w_interactions %<>%

```

```

    str_remove(":regionSWAFR") %>%
    {glue("-{.}")} %>%
    paste(collapse = " ")
  update(m,
    formula = glue(". ~ . {reparameterisation}"),
    data = data
  )
}
# Test:
# a <- m_HDS_richness
# b <- reparameterise(m_HDS_richness)
# AIC(a, b) # same model! :)
m_QDS_richness %<>% reparameterise()
m_HDS_richness %<>% reparameterise()
m_DS_richness %<>% reparameterise()

models <- list(
  QDS_richness = m_QDS_richness,
  HDS_richness = m_HDS_richness,
  DS_richness = m_DS_richness
)
models_summary <- models %>%
  map_df(.id = "response", tidy, conf.int = TRUE) %>%
  dplyr::select(-std.error, -statistic) %>%
  filter(term != "(Intercept)")

models_R2 <- models %>%
  map_df(.id = "response", glance) %>%
  dplyr::select(response, adj.r.squared)

models_summary %<>% full_join(models_R2)

glance(m_QDS_richness)

## # A tibble: 1 x 11
##   r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC
##   <dbl>      <dbl> <dbl>    <dbl>    <dbl> <int>  <dbl> <dbl> <dbl>
## 1    0.274        0.259 0.309    18.3 5.18e-39    15  -162.  356.  429.
## # ... with 2 more variables: deviance <dbl>, df.residual <int>

glance(m_HDS_richness)

## # A tibble: 1 x 11
##   r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC
##   <dbl>      <dbl> <dbl>    <dbl>    <dbl> <int>  <dbl> <dbl> <dbl>
## 1    0.377        0.330 0.283     8.00 3.42e-13    15  -23.4   78.7  131.
## # ... with 2 more variables: deviance <dbl>, df.residual <int>

glance(m_DS_richness)

## # A tibble: 1 x 11
##   r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC
##   <dbl>      <dbl> <dbl>    <dbl>    <dbl> <int>  <dbl> <dbl> <dbl>
## 1    0.705        0.622 0.200     8.48 2.25e-7    12   16.6  -7.12  18.0
## # ... with 2 more variables: deviance <dbl>, df.residual <int>

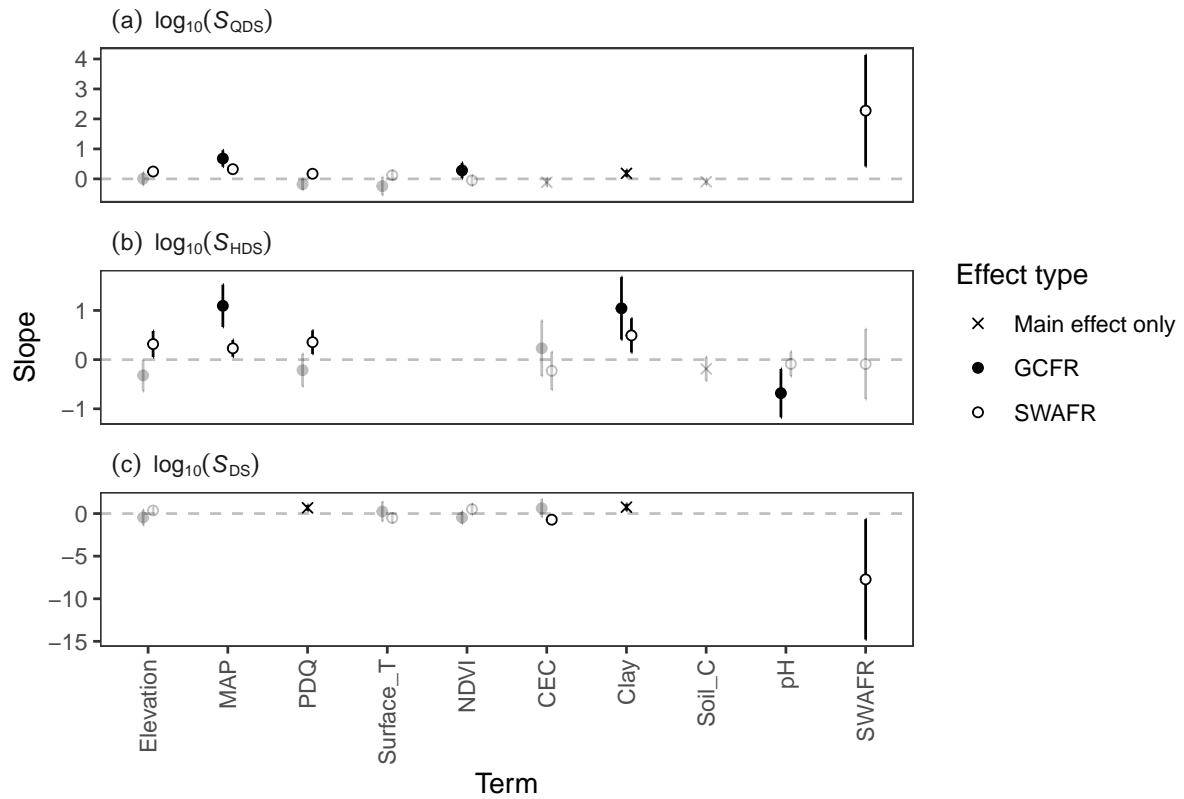
models_summary_for_plot <- models_summary %>%
  mutate(
    response = case_when(
      response == "QDS_richness" ~ "(a)~log[10](italic(S)[QDS])",

```

```

    response == "HDS_richness" ~ "(b)~log[10](italic(S)[HDS])",
    response == "DS_richness" ~ "(c)~log[10](italic(S)[DS])"
  ),
  region =
    case_when(
      str_detect(term, "regionSWAFR") ~ "SWAFR",
      str_detect(term, "regionGCFR") ~ "GCFR",
      TRUE ~ "Main effect only"
    ) %>%
    factor(levels = c("Main effect only", "GCFR", "SWAFR")),
  term = term %>%
    str_replace_all("\\.", " ") %>%
    str_remove_all("regionSWAFR:") %>%
    str_remove_all("regionGCFR:") %>%
    str_replace_all("regionSWAFR", "SWAFR") %>%
    factor(levels = c(str_replace_all(var_names, " ", "_"), "SWAFR")),
  sig = (p.value < 0.05)
)
ggplot(models_summary_for_plot) +
  aes(
    term, estimate,
    fill = region, group = region, shape = region,
    alpha = sig
  ) +
  geom_hline(yintercept = 0, linetype = "dashed", colour = "grey75") +
  geom_errorbar(
    aes(ymin = conf.low, ymax = conf.high),
    position = position_dodge(width = 0.25),
    width = 0
  ) +
  geom_point(position = position_dodge(width = 0.25)) +
  labs(x = "Term", y = "Slope") +
  scale_fill_manual(values = c(NA, "black", "white")) +
  scale_shape_manual(values = c(4, 21, 21)) +
  scale_alpha_manual(values = c(0.25, 1)) +
  facet_wrap(~response, nrow = 3, scales = "free_y", labeller = label_parsed) +
  guides(
    fill = FALSE,
    shape = guide_legend(
      title = "Effect type",
      override.aes = list(fill = c(NA, "black", "white"))
    ),
    alpha = FALSE
  ) +
  theme(
    axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5),
    strip.text.x = element_text(angle = 0, hjust = 0)
  )

```



```
ggplot(models_summary_for_plot[models_summary_for_plot$term != "SWAFR", ]) +
  aes(
    term, estimate,
    fill = region, group = region, shape = region,
    alpha = sig
  ) +
  geom_hline(yintercept = 0, linetype = "dashed", colour = "grey75") +
  geom_errorbar(
    aes(ymin = conf.low, ymax = conf.high),
    position = position_dodge(width = 0.25),
    width = 0
  ) +
  geom_point(position = position_dodge(width = 0.25)) +
  labs(x = "Term", y = "Slope") +
  scale_fill_manual(values = c(NA, "black", "white")) +
  scale_shape_manual(values = c(4, 21, 21)) +
  scale_alpha_manual(values = c(0.25, 1)) +
  facet_wrap(~response, nrow = 3, scales = "free_y", labeller = label_parsed) +
  guides(
    fill = FALSE,
    shape = guide_legend(
      title = "Effect type",
      override.aes = list(fill = c(NA, "black", "white"))
    ),
    alpha = FALSE
  ) +
  theme(
    axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5),
    strip.text.x = element_text(angle = 0, hjust = 0)
  )
```

