R Code: Continental-scale controls on soil organic carbon across sub-Saharan Africa

Sophie F. von Fromm

10/16/2020

1 AfSIS data

This document contains all the R code that were used to generate the statistical analysis and all figures for the manuscript von Fromm et al. (2020): *Continental-scale controls on soil organic carbon across sub-Saharan Africa*. If you have questions please contact Sophie F. von Fromm (sfromm@bgc-jena.mpg.de). The document also contains several links to websites from where some of the code used here have been adapted.

Note: For figures, only the code is provided. The figures themselves are shown in the manuscript and in the SI of the manuscript.

The following R packages are needed for the code:

```
library(tidyverse)
library(ggpubr)
library(raster)
library(sf)
library(e1071)
library(RColorBrewer)
library(tmap)
library(bestNormalize)
library(nlme)
library(Hmisc)
library(sjPlot)
library(rpart)
library(rpart.plot)
library(mlr)
library(pdp)
library(ranger)
library(cowplot)
library(car)
library(grid)
```

Create own theme for making figures with ggplot:

```
own_theme <- ggplot2::theme_bw(base_size = 14) +
  ggplot2::theme(rect = element_blank(),
                 axis.ticks = element_line(color = "black"),
                 axis.text = element_text(color = "black"),
                 axis.line = element_line(color = "black"),
                 panel.grid.minor = element_blank())
```

The data used in this project is from the African Soil Information Service (AfSIS: http://africasoils.net/) which is lead and maintained by the International Centre for Research in Agroforestry, Nairobi, Kenya. The lab meaurements were led by Steve P. McGrath and Stephan M. Haefele at the Department of Sustainable Agriculture Sciences, Rothamsted Research, Harpenden UK. For more information see also the *Methods* in von Fromm et al (2020).

Overview:
The following section gives a short overview about the data used for the publication.
45 variables and 2002 samples are included in the data file:

```
##  [1] "SSN"          "Longitude"    "Latitude"     "Region"       "Country"
##  [6] "Site"         "Cluster"      "Profile"      "Depth"        "Clay_8um"
## [11] "Total_C"      "CORG"         "CINORG"       "Total_N"      "pH"
## [16] "Am_Ox_Al"     "Am_Ox_Fe"     "Am_Ox_Mn"     "Am_Ox_P"      "Olsen_P"
## [21] "pbi"          "Caex"         "Kex"          "Mgex"         "Naex"
## [26] "eCEC"         "Al"           "Ca"           "Co"           "Cr"
## [31] "Cu"           "Fe"           "K"            "Mg"           "Mn"
## [36] "Na"           "Ni"           "P"            "Pb"           "S"
## [41] "Zn"           "Total_PSD"    "VegStructure" "PlotCultMgd"  "Notes"
```

Short explanation of the variables included in the dataset:

SSN – Sample code
Longitude/Latitude – Longitude/Latitude of sample location
Region – Sampling area within sub-Saharan Africa
Country – Country of origin of samples
Site – Site name
Cluster – Cluster number within each site
Profile – Profile number within each cluster
Depth – Sampling depth (categorial; topsoil: 0-20cm, subsoil: 20-50cm)
Clay_8um – All particles < 8 µm [%]
Total_C, CORG, CINORG – Total, organic, inorganic C content [wt-%]
Total_N – Total N content [wt-%]
pH – Soil pH_{H2O}
Am_Ox_Al/Fe/Mn/P – Amorphous oxalate aluminum/iron/manganese/phosphorus [mg/kg]
Olsen_P – Sodium bicarbonate extraction of available P [mg/kg]
pbi – Phosphorus buffer index
Caex, Kex, Mgex, Naex - Exchangeable Ca, K, Mg, Na [$cmol^+$/kg]
eCEC – Effective cation exchange capacity
Al–Zn – Element concentration (aqua regia) [mg/kg]
Total_PSD – Total particle size distribution [%]

VegStructure – Land-cover (based on LDSF field data)
PlotCultMgd – Is the plot cultivated? Yes/No
Notes - Field notes

Missing values within the data:

```
AfSIS_RefData_allSites %>%
  dplyr::select_if(is.numeric) %>%
  gather(key = Variable,value = value) %>%
  group_by(Variable) %>%
  summarise(n = n(), n_missing = sum(is.na(value))) %>%
  filter(n_missing > 0) %>%
  arrange(desc(n_missing)) %>%
  knitr::kable()
```

Variable	n	n_missing
Clay_8um	2002	363
Total_PSD	2002	363
PlotCultMgd	2002	159
pH	2002	30
Total_C	2002	10
Total_N	2002	10
CINORG	2002	4
CORG	2002	4

Negative values within the data:

```
AfSIS_RefData_allSites %>%
  dplyr::select_if(is.numeric) %>%
  dplyr::select(-Longitude, -Latitude) %>%
  gather(key = Variable, value = value) %>%
  group_by(Variable) %>%
  summarise(n = n(), n_negative = sum(value < 0)) %>%
  filter(n_negative > 0) %>%
  arrange(desc(n_negative)) %>%
  knitr::kable()
```

Variable	n	n_negative
Am_Ox_P	2002	493
Naex	2002	145
eCEC	2002	123
Pb	2002	7
Cu	2002	6
P	2002	4
Caex	2002	2

Variable	n	n_negative
Kex	2002	1
Mgex	2002	1

2 Global data

For the statistical analyses the AfSIS data was paired with global data (MAP, MAT, PET, land cover (only for gap-filling)). The follwing section provides the code for extracting the global data and merging it with the AfSIS dataset.
Note: You need to download and store the global data beforehand on your computer. Links to the data sources are provided below.

2.1 Climate data

Data sources:
WorldClim: http://worldclim.org/version2 and
PET: https://figshare.com/articles/Global_Aridity_Index_and_Potential_Evapotranspiration_ET0_Climate_Database_v2/7504448/3

MAT (= Mean annual temperature) raster:

```
MAT_directory <- "D:/Sophie/PhD/AfSIS_GlobalData/wc2.0_30s_bio/wc2.0_bio_30s_01.tif"
MAT_raster <- raster::raster(MAT_directory)
MAT <- raster::extract(MAT_raster, cbind(AfSIS_RefData_allSites$Longitude,
                                         AfSIS_RefData_allSites$Latitude))
```

MAP (= Mean annual precipitation) raster:

```
MAP_directory <- "D:/Sophie/PhD/AfSIS_GlobalData/wc2.0_30s_bio/wc2.0_bio_30s_12.tif"
MAP_raster <- raster::raster(MAP_directory)
MAP <- raster::extract(MAP_raster, cbind(AfSIS_RefData_allSites$Longitude,
                                         AfSIS_RefData_allSites$Latitude))
```

PET (= Potential annual evapotranspiration) raster:

```
PET_directory <- "D:/Sophie/PhD/AfSIS_GlobalData/global-et0_annual.tif/et0_yr/et0_yr.tif"
PET_raster <- raster::raster(PET_directory)
PET <- raster::extract(PET_raster, cbind(AfSIS_RefData_allSites$Longitude,
                                         AfSIS_RefData_allSites$Latitude))
```

Merge AfSIS data with extracted climate data:

```
AfSIS_RefData_allSites_Clima <- cbind(AfSIS_RefData_allSites, MAT, MAP, PET)
```

Calculating Aridity Index (PET/MAP):

```
AfSIS_RefData_allSites_Clima <- AfSIS_RefData_allSites_Clima %>%
  dplyr::mutate(AridityIndex = PET/MAP)
```

2.2 Monthly climate data

Data sources:
MAP: https://www.worldclim.org/data/worldclim21.html) and
PET: https://cgiarcsi.community/data/global-aridity-and-pet-database/

MAP (= monthly mean annual precipitation) raster

1 - January:

```
MAP_1_directory <- "D:/Sophie/PhD/AfSIS_GlobalData/wc2.1_30s_prec/wc2.1_30s_prec_01.tif"
MAP_1_raster <- raster::raster(MAP_1_directory)
MAP_1 <- raster::extract(MAP_1_raster, cbind(AfSIS_RefData_allSites_Clima$Longitude,
                                             AfSIS_RefData_allSites_Clima$Latitude))
```

2 - February:

```
MAP_2_directory <- "D:/Sophie/PhD/AfSIS_GlobalData/wc2.1_30s_prec/wc2.1_30s_prec_02.tif"
MAP_2_raster <- raster::raster(MAP_2_directory)
MAP_2 <- raster::extract(MAP_2_raster, cbind(AfSIS_RefData_allSites_Clima$Longitude,
                                             AfSIS_RefData_allSites_Clima$Latitude))
```

Code for the other months/variables are not shown since they are always the same.

Merge AfSIS data with extracted climate data and calculate monthly wetness index (MAP/PET):

```
AfSIS_RefData_allSites_Clima_month <- cbind(AfSIS_RefData_allSites_Clima,
                                            MAP_1, MAP_2, MAP_3, MAP_4, MAP_5,
                                            MAP_6, MAP_7, MAP_8, MAP_9, MAP_10,
                                            MAP_11, MAP_12, PET_1, PET_2, PET_3,
                                            PET_4, PET_5, PET_6, PET_7, PET_8,
                                            PET_9, PET_10, PET_11, PET_12)

AfSIS_RefData_allSites_Clima_month <- AfSIS_RefData_allSites_Clima_month %>%
  dplyr::mutate(WetnessIndex_1 = MAP_1/PET_1,
                WetnessIndex_2 = MAP_2/PET_2,
                WetnessIndex_3 = MAP_3/PET_3,
                WetnessIndex_4 = MAP_4/PET_4,
                WetnessIndex_5 = MAP_5/PET_5,
```

```
                WetnessIndex_6 = MAP_6/PET_6,
                WetnessIndex_7 = MAP_7/PET_7,
                WetnessIndex_8 = MAP_8/PET_8,
                WetnessIndex_9 = MAP_9/PET_9,
                WetnessIndex_10 = MAP_10/PET_10,
                WetnessIndex_11 = MAP_11/PET_11,
                WetnessIndex_12 = MAP_12/PET_12) %>%
  dplyr::select(-c(MAP_1:PET_12))
```

2.3 Africa land cover data

Source: http://www.esa.int/ESA_Multimedia/Images/2017/10/African_land_cover

```
LandCover_directory <- "D:/Sophie/PhD/AfSIS_GlobalData/ESA_AfricaLandCover_2015_16/ESACCI-LC-L4
LandCover_raster <- raster::raster(LandCover_directory)
LandCover <- raster::extract(LandCover_raster,
                             cbind(AfSIS_RefData_allSites_Clima_month$Longitude,
                                   AfSIS_RefData_allSites_Clima_month$Latitude))
AfSIS_RefData_allSites_Clima_LC <- cbind(AfSIS_RefData_allSites_Clima_month,
                                         LandCover)
```

Re-name land cover code:

```
AfSIS_RefData_allSites_Clima_LC <- AfSIS_RefData_allSites_Clima_LC %>%
  dplyr::mutate(LandCover_ESA = case_when(
    LandCover == 0 ~ "No data",
    LandCover == 1 ~ "Forest",
    LandCover == 2 ~ "Shrubland",
    LandCover == 3 ~ "Grassland",
    LandCover == 4 ~ "Cropland",
    LandCover == 5 ~ "Flooded",
    LandCover == 6 ~ "Sparse vegetation",
    LandCover == 7 ~ "Bare",
    LandCover == 8 ~ "Settlements",
    LandCover == 9 ~ "Snow/Ice",
    LandCover == 10 ~ "Open water"
  )) %>%
  dplyr::select(-LandCover)

#Save final (global) dataset
#write_csv(AfSIS_RefData_allSites_Clima_LC, "AfSIS_RefData_Roth_allSites_GlobalData.csv")
```

3 Preparation of data for analysis

The following chapter provides all the code in order to prepare the data for the actual analyses. This includes converting units and the calculation of the CIA (weathering), gap-filling and re-classification of the land cover data, excluding missing and negative values, as well as creating groups based on seasonality, pH_{H2O}, and CIA for the modelling.

Rename and filter for columns of interest.
Convert mg/kg in wt-%: (Al, Fe, Ca, K, Mg, Mn, Na, Al_ox, Fe_ox)/10000 = wt-%. Calculate the chemical index of alteration (CIA). For more details see *Methods* in von Fromm et al. (2020).

```
AfSIS_RefData_allSites_final <- AfSIS_RefData_allSites_GlobalData %>%
  dplyr::select(-c(18:21), -c(23:26), -c(29:32), -c(34:35), -c(37:41)) %>%
  dplyr::rename(Alox = Am_Ox_Al, Feox = Am_Ox_Fe) %>%
  dplyr::mutate(Al = Al/10000, Ca = Ca/10000,
                K = K/10000, Na = Na/10000,
                Alox = Alox/10000, Feox = Feox/10000,
                Ca_IC = case_when(
                  Total_C - CORG > 0 & Ca > Na ~ Na,
                  Total_C == CORG ~ Ca),
                CIA = 100*(Al/(Al + Ca_IC + Na + K)),
                CIA_uncor = 100*(Al/(Al + Ca + Na + K)))
```

Compare uncorrected CIA and corrected CIA:

```
# Uncorrected CIA
CIA_uncor <- AfSIS_RefData_allSites_final %>%
  drop_na(Total_C, CORG) %>%
  ggplot(aes(x = CIA_uncor, y = Total_C - CORG)) +
  geom_point(shape = 21) +
  own_theme +
  scale_y_continuous(expression("C"[total]~" - C"[org]~" [wt-%]"),
                     expand = c(0.01,0.01), limits = c(0,10)) +
  scale_x_continuous("uncorrected CIA [%]", expand = c(0.01,0.01),
                     limits = c(0,100))

# Corrected CIA
CIA <- AfSIS_RefData_allSites_final %>%
  drop_na(Total_C, CORG) %>%
  ggplot(aes(x = CIA, y = Total_C - CORG)) +
  geom_point(shape = 21) +
  own_theme +
  scale_y_continuous(expression("C"[total]~" - C"[org]~" [wt-%]"),
                     expand = c(0.01,0.01), limits = c(0,10)) +
  scale_x_continuous("corrected CIA [%]", expand = c(0.01,0.01),
                     limits = c(0,100))

# Plot both together
```

```
ggpubr::ggarrange(CIA_uncor, CIA)

ggsave("AfSIS_allData_CIA_correction.pdf", height = 4, width = 8.5, dpi = 300)
```

The corrected CIA (see Figure A1 in the manuscript) does not show an obvious trend with inorganic C content anymore. The corrected CIA will be used for all statistical analyses and only named CIA from here on.

3.1 Land cover

This section provides the code for gap-filling the land-cover data with the product from ESA and for the re-classification of the land cover groups.

Replace missing land cover data with ESA product:

```
AfSIS_RefData_allSites_final$VegStructure <-
  ifelse(is.na(AfSIS_RefData_allSites_final$VegStructure),
         AfSIS_RefData_allSites_final$LandCover_ESA,
         AfSIS_RefData_allSites_final$VegStructure)
```

Merge land cover data into four groups: Forest, Cropland, Grassland, and Other (including mainly shrubland, bushland, and woodland):

```
AfSIS_RefData_allSites_final_Veg <- AfSIS_RefData_allSites_final %>%
  mutate(LandCover = case_when(
    VegStructure == "Cropland" ~ "Cropland",
    VegStructure == "Forest" ~ "Forest",
    VegStructure == "Grassland" ~ "Grassland",
    VegStructure == "Other" & PlotCultMgd == 1 ~ "Cropland",
    #assignment based on field notes
    SSN == "icr005955" ~ "Cropland",
    SSN == "icr005995" ~ "Cropland",
    SSN == "icr006021" ~ "Cropland",
    SSN == "icr006034" ~ "Cropland",
    SSN == "icr006175" ~ "Cropland",
    SSN == "icr030334" ~ "Cropland",
    SSN == "icr030485" ~ "Cropland",
    SSN == "icr030486" ~ "Cropland",
    SSN == "icr030489" ~ "Cropland",
    SSN == "icr030490" ~ "Cropland",
    SSN == "icr033490" ~ "Grassland",
    SSN == "icr038239" ~ "Cropland",
    SSN == "icr054256" ~ "Grassland",
    SSN == "icr054257" ~ "Grassland",
    SSN == "icr054340" ~ "Cropland",
    SSN == "icr054395" ~ "Cropland",
```

```
    SSN == "icr054396" ~ "Cropland",
    SSN == "icr054402" ~ "Cropland",
    SSN == "icr054403" ~ "Cropland",
    SSN == "icr054531" ~ "Grassland",
    SSN == "icr054626" ~ "Grassland",
    SSN == "icr054709" ~ "Grassland",
    SSN == "icr054710" ~ "Grassland",
    SSN == "icr068199" ~ "Cropland",
    SSN == "icr073208" ~ "Cropland",
    TRUE ~ "Other",
  )) %>%
dplyr::select(-c(VegStructure, LandCover_ESA, PlotCultMgd, Notes))
```

3.2 Filtering

This section contains the code for filtering the data, so that only meaningful values (no NA's, no negative or unrealistic) are included in the final dataset for further analyses.

Drop samples with missing values:

```
AfSIS_RefData_noNA <- AfSIS_RefData_allSites_final_Veg %>%
  drop_na(CORG, pH, Clay_8um, CIA) %>%
  #only include realistic values for CORG and Caex
  filter(CORG <= 12 & CORG >= 0 & Caex >= 0)

# Check for negative/missing values
AfSIS_RefData_noNA %>%
  dplyr::select_if(is.numeric) %>%
  dplyr::select(-Longitude, -Latitude) %>%
  gather(key = Variable, value = value) %>%
  group_by(Variable) %>%
  summarise(n = n(), n_negative = sum(value < 0),
            n_missing = sum(is.na(value))) %>%
  knitr::kable()
```

Variable	n	n_negative	n_missing
Al	1601	0	0
Alox	1601	0	0
AridityIndex	1601	0	0
Ca	1601	0	0
Ca_IC	1601	0	0
Caex	1601	0	0
CIA	1601	0	0
CIA_uncor	1601	0	0
CINORG	1601	0	0
Clay_8um	1601	0	0

Variable	n	n_negative	n_missing
Cluster	1601	0	0
CORG	1601	0	0
Feox	1601	0	0
K	1601	0	0
MAP	1601	0	0
MAT	1601	0	0
Na	1601	0	0
PET	1601	0	0
pH	1601	0	0
Profile	1601	0	0
Total_C	1601	0	0
Total_N	1601	0	0
Total_PSD	1601	0	0
WetnessIndex_1	1601	0	0
WetnessIndex_10	1601	0	0
WetnessIndex_11	1601	0	0
WetnessIndex_12	1601	0	0
WetnessIndex_2	1601	0	0
WetnessIndex_3	1601	0	0
WetnessIndex_4	1601	0	0
WetnessIndex_5	1601	0	0
WetnessIndex_6	1601	0	0
WetnessIndex_7	1601	0	0
WetnessIndex_8	1601	0	0
WetnessIndex_9	1601	0	0

Overall, 1601 samples of the AfSIS reference dataset are available for further analysis. None of the selected variables contain negative values anymore.

3.3 Seasonality

For modelling, the data is grouped by the monthly wetness index to create more homogenous groups based on their climate.
Count number of wet months (based on monthly wetness index). Calculate mean for each month for each site and classify water regime:

```
AfSIS_avgWI_monthly <- AfSIS_RefData_noNA %>%
  tidyr::gather(key = "WetnessIndex_month", value = "WetnessIndex_value",
                WetnessIndex_1:WetnessIndex_12) %>%
  dplyr::group_by(Site, WetnessIndex_month) %>%
  summarise(WetnessIndex_value = mean(WetnessIndex_value)) %>%
  dplyr::mutate(WaterRegime_monthly = case_when(
    WetnessIndex_value >= 1 ~ "energy-limited",
    WetnessIndex_value < 1 ~ "water-limited"))
```

```
# count number of wet and dry month
AfSIS_WI_count <- data.frame(with(AfSIS_avgWI_monthly,
                                  table(Site, WaterRegime_monthly)))

# only keep number of wet month
AfSIS_nWM <- AfSIS_WI_count %>%
  dplyr::rename(n_wet_month = Freq) %>%
  dplyr::filter(WaterRegime_monthly == "energy-limited")

# merge with original dataset
AfSIS_RefData_noNA <- AfSIS_RefData_noNA %>%
  dplyr::left_join(AfSIS_nWM, by = "Site") %>%
  dplyr::select(-WaterRegime_monthly, -c(WetnessIndex_1:WetnessIndex_12))

# create groups
AfSIS_RefData_noNA <- AfSIS_RefData_noNA %>%
  dplyr::mutate(wet_month_Class = case_when(
    n_wet_month == 0 ~ "0",
    n_wet_month == 1 | n_wet_month == 2 | n_wet_month == 3 ~ "1-3",
    n_wet_month > 3 ~ "4-7"))
```

3.4 pH_{H2O}

For modelling, the data is grouped by the numeric pH_{H20} values to create more homogenous groups based on their pH_{H20} values.
Group soil pH_{H2O} into four groups with equal number of samples within each group.

```
table(ggplot2::cut_number(AfSIS_RefData_noNA$pH, 4)) # good group size + intervals
```

```
##
## [3.89,5.22]  (5.22,6.1]  (6.1,7.52] (7.52,9.92]
##         404         399         398         400
```

```
AfSIS_RefData_noNA$pH_Class <- as.numeric(ggplot2::cut_number(AfSIS_RefData_noNA$pH, 4))
```

Convert class numbers into meaningful pH classes:

```
AfSIS_RefData_noNA$pH_Class[AfSIS_RefData_noNA$pH_Class == 1] <- "strongly acid"
AfSIS_RefData_noNA$pH_Class[AfSIS_RefData_noNA$pH_Class == 2] <- "moderately acid"
AfSIS_RefData_noNA$pH_Class[AfSIS_RefData_noNA$pH_Class == 3] <- "neutral"
AfSIS_RefData_noNA$pH_Class[AfSIS_RefData_noNA$pH_Class == 4] <- "alkaline"
```

3.5 CIA

For modelling, the data is grouped by the numeric CIA (Chemical Index of Alteration) values to create more homogenous groups based on their CIA values.
Group CIA data into two groups with equal number of samples within each group.

```
table(ggplot2::cut_number(AfSIS_RefData_noNA$CIA, 2))  # good group size + intervals
```

```
##
## [10.3,88.1] (88.1,99.9]
##         801         800
```

```
AfSIS_RefData_noNA$CIA_Class <- as.numeric(ggplot2::cut_number(AfSIS_RefData_noNA$CIA, 2))
```

Convert class numbers into meaningful classes:

```
AfSIS_RefData_noNA$CIA_Class[AfSIS_RefData_noNA$CIA_Class == 1] <- "moderate"
AfSIS_RefData_noNA$CIA_Class[AfSIS_RefData_noNA$CIA_Class == 2] <- "high"
```

```
#Save final dataset with no missing values
AfSIS_RefData_noNA %>%
  dplyr::select(-11, -c(13:14), -c(19:23), -28, -30, -32) %>%
write_csv("AfSIS_RefData_Roth_noNA.csv")
```

4 Sampling locations

This chapter contains the code to plot the sampling locations with the R package *tmap*, including an aridity map (PET/MAP) as a background map and one detailed overview map from one site (Didy, Madagascar) as an example for the sampling scheme.

Convert dataset into spatial format:

```
AfSIS_sf <- sf::st_as_sf(AfSIS_RefData_noNA,
                         coords = c("Longitude", "Latitude"), crs = 4326)
```

Extract map of Africa:

```
data("World")
Africa_map <- World %>%
  dplyr::filter(continent == "Africa")
```

4.1 Aridity index map

The aridity map used here is based on the same global products used in the *Global data* chapter. And calculated for the African continent.
Create boundaries for African contintent (based on Long/Lat) and convert into spatial format:

```
Africa_bound <- data.frame(ID = 1:4, Longitude = c(-18, 60, -18, 60),
                           Latitude = c(20, 20, -36, -36))

Africa_sf <- sf::st_as_sf(Africa_bound, coords = c("Longitude", "Latitude"),
                          crs = 4326)
```

Load climate data and crop global maps based on Africa boundaries. Calculate Aridity Index (PET/MAP) for Africa:

```
MAP_raster <-
  raster::raster("D:/Sophie/PhD/AfSIS_GlobalData/wc2.0_30s_bio/wc2.0_bio_30s_12.tif") %>%
  #reduce size of raster file
  raster::aggregate(fact = 5, fun = mean)
MAP_africa <- raster::crop(MAP_raster, Africa_sf)

PET_raster <-
  raster::raster("D:/Sophie/PhD/AfSIS_GlobalData/global-et0_annual.tif/et0_yr/et0_yr.tif") %>%
  #reduce size of raster file
  raster::aggregate(fact = 5, fun = mean)
PET_africa <- raster::crop(PET_raster, Africa_sf)

AriInd_africa <- PET_africa/MAP_africa
```

4.2 Mapping

The final map can be found in the manuscript (*Figure 1*).

```
# Boundaries for example site (Didy, Madagascar)
Site_range <- AfSIS_sf %>%
  dplyr::filter(Site == "Didy") %>%
  sf::st_bbox(crs = 4326) %>%
  sf::st_as_sfc()

# Map of example Site
Site_map <- tmap::tm_grid(col = "grey", labels.size = 1, labels.col = "black",
                          n.x = 3, n.y = 3, labels.rot = c(0,90)) +
  tmap::tm_shape(AfSIS_sf, bbox = Site_range) +
  tmap::tm_dots(size = 0.1, col = "#525252") +
  tmap::tm_layout(main.title = "b)", main.title.position = 0.1)
```

```r
# Box around Didy sampling location
Site_range_large <- sf::st_bbox(c(xmin = 48, xmax = 49.7, ymin = -19.1, ymax = -16.9),
                                crs = 4326) %>% st_as_sfc()

# Map of all sampling locations with aridity index map in the background
AfSIS_map <- tmap::tm_grid(col = "grey", labels.size = 1.5, labels.cardinal = TRUE,
                           labels.col = "black") +
  tmap::tm_shape(AriInd_africa) +
  tmap::tm_raster(title = "Aridity Index\n(PET/MAP)", style = "fixed",
                  breaks = c(0,0.5,1.0,1.7,2.9,6.6,100), alpha = 0.8,
                  labels = c("0.0 - 0.5", "0.5 - 1.0", "1.0 - 1.7", "1.7 - 2.9",
                             "2.9 - 6.6", "> 6.6"),
                  palette = c("#4393c3", "#92c5de", "#d1e5f0",
                              "#fddbc7", "#f4a582", "#d6604d")) +
  tmap::tm_shape(Africa_map, projection = 4326) +
  tmap::tm_borders(col = "#969696", lwd = 0.5) +
  tmap::tm_shape(AfSIS_sf) +
  tmap::tm_dots(size = 0.5, col = "#525252", shape = 24) +
  tmap::tm_shape(Site_range_large) +
  tmap::tm_borders(lwd = 3) +
  tmap::tm_legend(position = c("0.03", "0.15"), legend.text.size = 1.3,
                  legend.title.size = 1.5) +
  tmap::tm_compass(type = "arrow", north = 0, position = c(0.01, 0.65), size = 2,
                   text.size = 1) +
  tmap::tm_scale_bar(position = c("left", "bottom"), text.size = 1.3)+
  tmap::tm_layout(main.title = "a)", main.title.position = 0.1,
                  title = "b)", title.position = c(0.84,0.37))

# Merge both maps
AfSIS_map
print(Site_map, vp = viewport(0.845, 0.746, width = 0.41, height = 0.41))

tmap::tmap_save(tm = AfSIS_map, filename = "AfSIS_RefData_SamplingLocations.jpeg",
                dpi = 300, outer.margins = c(0.01,0.01,0.01,0.15),
                units = "in", height = 7, width = 10, insets_tm = Site_map,
                insets_vp = viewport(0.845, 0.746, width = 0.41, height = 0.41))
```

5 Parameter distribution

This section provides code to get an overview about the data, including summary statistics, skweness, kurtosis, histograms and shapiro-wilk tests. Some of the results are shown in *Table 1* in the manuscript. In addition, it also contains the code for all the plots in the manuscript that show raw data (*Figure 2*, *Figure 3c,d*, *Figure 4b*, *Figure A3*).

5.1 Data distribution including skewness and kurtosis

Table 1 in the manuscript.

```
AfSIS_RefData_noNA %>%
  dplyr::select_if(is.factor) %>%
  gather(key = Variable, value = value) %>%
  group_by(Variable) %>%
  summarise(n_unique = length(unique(value))) %>%
  arrange(desc(n_unique)) %>%
  knitr::kable()
```

Variable	n_unique
Site	45
Country	17
Cluster	16
Profile	15
LandCover	4
pH_Class	4
Region	3
wet_month_Class	3
CIA_Class	2
Depth	2

```
AfSIS_RefData_noNA %>%
  dplyr::select_if(is.numeric) %>%
  gather(key = Variable, value = value) %>%
  group_by(Variable) %>%
  summarise(mean = round(mean(value), digits = 2),
            sd = round(sd(value), digits = 2),
            P0 = round(quantile(value,0), digits = 2),
            P25 = round(quantile(value,0.25), digits = 2),
            P50 = round(quantile(value,0.50), digits = 2),
            P75 = round(quantile(value,0.75), digits = 2),
            P100 = round(quantile(value,1), digits = 2),
            Skew = round(e1071::skewness(value), digits = 2),
            Kurt = round(e1071::kurtosis(value), digits = 2)) %>%
  knitr::kable()
```

Variable	mean	sd	P0	P25	P50	P75	P100	Skew	Kurt
Alox	0.28	0.36	0.01	0.12	0.20	0.29	3.71	4.52	25.29
AridityIndex	2.35	1.73	0.71	1.20	1.54	3.16	9.54	1.46	1.31
Caex	10.29	11.01	0.03	1.34	5.86	16.49	75.66	1.28	1.32
CIA	87.74	9.30	10.34	81.70	88.09	95.98	99.91	-1.04	3.88
Clay_8um	55.42	22.64	0.12	37.70	57.92	74.73	100.00	-0.26	-1.00

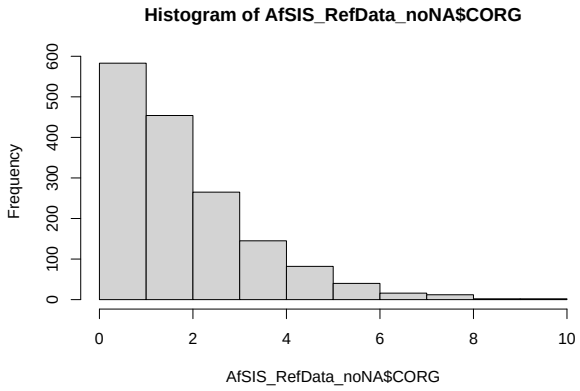
Variable	mean	sd	P0	P25	P50	P75	P100	Skew	Kurt
CORG	1.84	1.51	0.07	0.65	1.42	2.54	9.19	1.42	2.23
Feox	0.38	0.56	0.01	0.10	0.21	0.40	4.46	3.60	14.96
Latitude	-4.28	11.21	-29.98	-18.27	-1.71	4.19	14.88	-0.24	-1.17
Longitude	34.54	12.79	-13.43	32.27	36.51	39.63	48.76	-1.65	2.97
MAP	1070.35	487.21	255.00	648.00	1057.00	1432.00	2708.00	0.29	-0.63
MAT	21.67	3.24	13.74	19.80	21.52	22.95	29.82	0.17	-0.12
PET	1810.33	310.46	1350.00	1571.00	1759.00	1933.00	2949.00	1.19	1.96
pH	6.28	1.31	3.89	5.22	6.10	7.52	9.92	0.27	-1.11

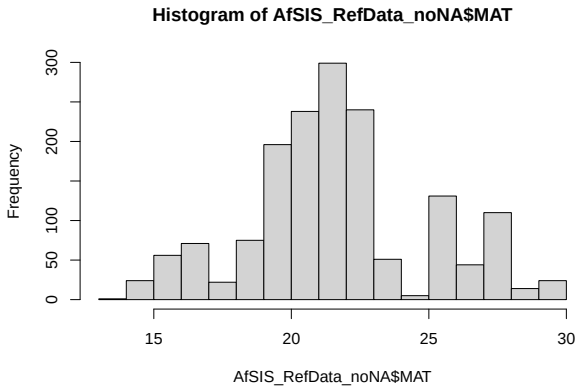
Shapiro-Wilk-Test:
Normal distribution: p > 0.05

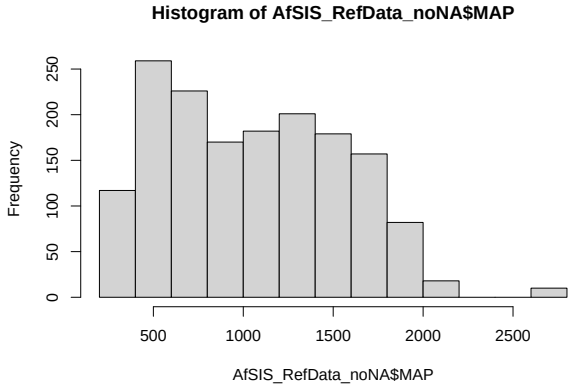
```
# Check if any p-value of the shapiro-wilk test is larger 0.5
AfSIS_RefData_noNA %>%
  dplyr::select_if(is.numeric) %>%
  purrr::map_dfr(~shapiro.test(.x)$p.value > 0.5) %>%
  gather(key = Variable, value = NormalDistributed) %>%
  knitr::kable()
```

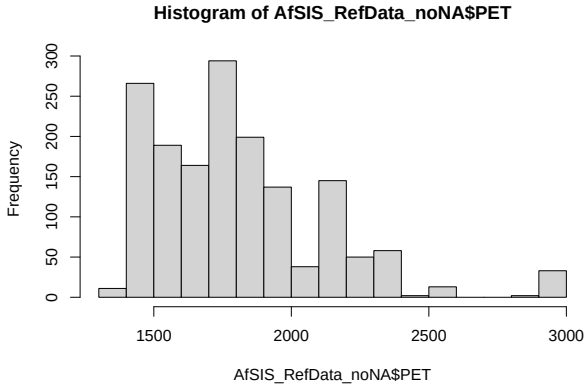
Variable	NormalDistributed
Longitude	FALSE
Latitude	FALSE
Clay_8um	FALSE
CORG	FALSE
pH	FALSE
Alox	FALSE
Feox	FALSE
Caex	FALSE
MAT	FALSE
MAP	FALSE
PET	FALSE
AridityIndex	FALSE
CIA	FALSE

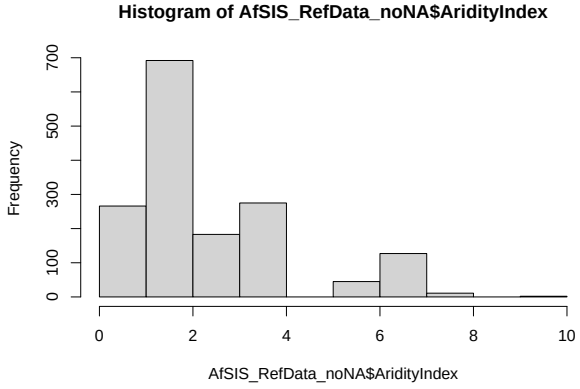
Histograms:

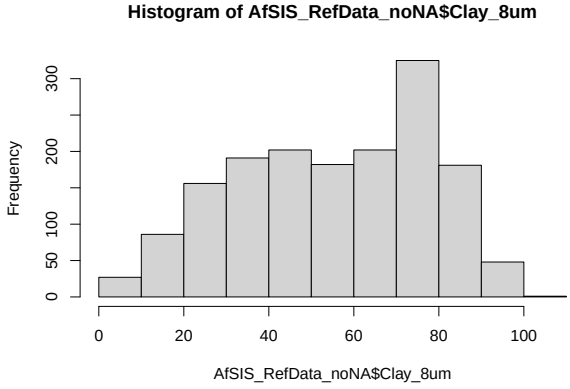

Histogram of AfSIS_RefData_noNA$CORG
Frequency
0
100
200
300
400
500
600
0
2
4
6
8
10
AfSIS_RefData_noNA$CORG

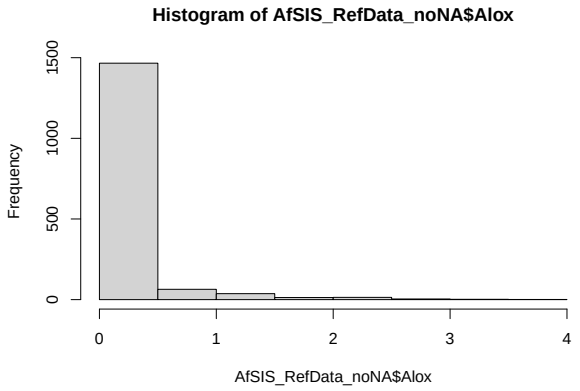

Histogram of AfSIS_RefData_noNA$MAT
Frequency
0
50
100
200
300
15
20
25
30
AfSIS_RefData_noNA$MAT

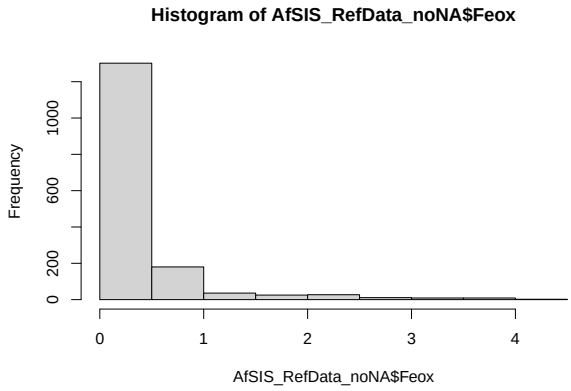

Histogram of AfSIS_RefData_noNA$MAP
Frequency
0
50
100
150
200
250
500
1000
1500
2000
2500
AfSIS_RefData_noNA$MAP

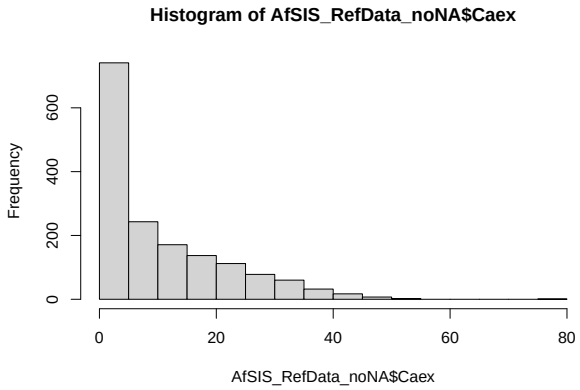

Histogram of AfSIS_RefData_noNA$PET
Frequency
0
50
100
150
200
250
300
1500
2000
2500
3000
AfSIS_RefData_noNA$PET

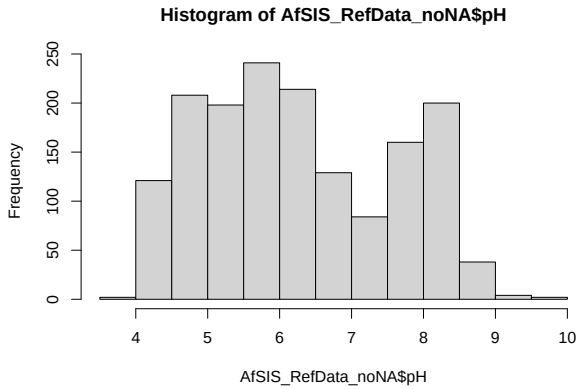

Histogram of AfSIS_RefData_noNA$AridityIndex
Frequency
0
100
300
500
700
0
2
4
6
8
10
AfSIS_RefData_noNA$AridityIndex

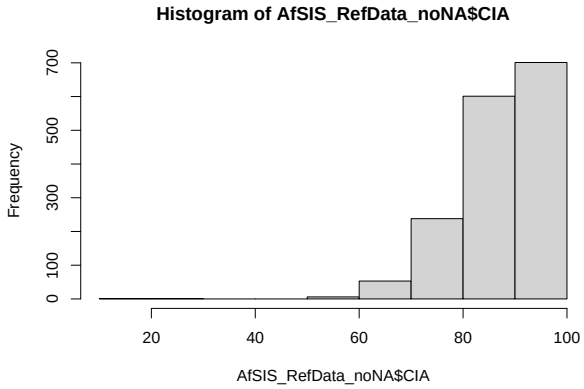

Histogram of AfSIS_RefData_noNA$Clay_8um
Frequency
0
50
100
200
300
0
20
40
60
80
100
AfSIS_RefData_noNA$Clay_8um











Based on Skweness, Kurtosis, Shapiro-Wilk-Test and Histograms none of the variables are normal distributed.

5.2 Plotting

Land cover and clay content vs CORG by depth:
Figure 2 in the manuscript.

```
ColorDepth <- brewer.pal(11, "BrBG")[c(1,3)]
```

```r
# Land-cover
AfSIS_LC_CORG <- AfSIS_RefData_noNA %>%
  ggplot(aes(x = LandCover, y = CORG, color = Depth)) +
  geom_boxplot(outlier.shape = 21, outlier.fill = "white", outlier.size = 3,
               size = 0.5) +
  own_theme +
  scale_y_continuous("", expand = c(0,0), breaks = seq(0,10,2)) +
  scale_x_discrete("") +
  coord_cartesian(ylim = c(0,10)) +
  scale_color_manual(values = ColorDepth)

# Clay content
AfSIS_Clay_CORG <- AfSIS_RefData_noNA %>%
  ggplot(aes(x = Clay_8um, y = CORG, color = Depth)) +
  geom_point(size = 3, fill = "white", shape = 21) +
  own_theme +
  geom_smooth(method = "lm") +
  theme(panel.spacing.x = unit(0.5, "cm"),
        strip.text = element_text(color = "black"),
        strip.background = element_rect(color = "grey")) +
  scale_y_continuous("", expand = c(0,0), breaks = seq(0,10,2)) +
  scale_x_continuous("", expand = c(0,0)) +
  coord_cartesian(ylim = c(0,10), xlim = c(0,100)) +
  scale_color_manual(values = ColorDepth)

# Clay content by site
# Selected site with different Clay-CORG relationships (neg,pos,none)
AfSIS_Clay_CORG_Site <- AfSIS_RefData_noNA %>%
  filter(Site == "Analavory" | Site == "Dambidolo" | Site == "Chinyanghuku") %>%
  ggplot(aes(x = Clay_8um, y = CORG, color = Depth)) +
  geom_point(size = 3, fill = "white", shape = 21) +
  facet_wrap(~Site, ncol = 1) +
  geom_smooth(method = "lm") +
  own_theme +
  theme(panel.spacing.x = unit(0.5, "cm"),
        strip.text = element_text(color = "black"),
        strip.background = element_rect(color = "grey")) +
  scale_y_continuous("", expand = c(0,0), breaks = seq(0,10,2)) +
  scale_x_continuous("", expand = c(0,0)) +
  coord_cartesian(ylim = c(0,10), xlim = c(0,101)) +
  scale_color_manual(values = ColorDepth)

# Plot both together
figure_arranged <- ggpubr::ggarrange(ggarrange(AfSIS_LC_CORG, AfSIS_Clay_CORG,
                                               nrow = 2, common.legend = TRUE,
                                               labels = list("a)", "b)")),
                                     AfSIS_Clay_CORG_Site, ncol = 2,
```

```
                                    widths = c(1.25,1), legend = "none",
                                    labels = list("", "c)"))

ggpubr::annotate_figure(figure_arranged,
                        left = text_grob("SOC [wt-%]", color = "black", rot = 90,
                                         size = 14),
                        bottom = text_grob("Clay and fine silt content [%]",
                                           color = "black", size = 14, vjust = -1))

ggsave("AfSIS_RefData_Clay_LC_SOC_Depth.jpeg", height = 5, width = 7, dpi = 300)
```

Clay content vs CORG by site:
Figure A3 in the supplement of the manuscript.

```
ColorDepth <- brewer.pal(11, "BrBG")[c(1,3)]

AfSIS_RefData_noNA %>%
  group_by(Site, Depth) %>%
  filter(n() > 1) %>%
  ungroup() %>%
  ggplot(aes(x = Clay_8um, y = CORG, color = Depth)) +
  geom_point(size = 3, fill = "white", shape = 21) +
  facet_wrap(~Site) +
  geom_smooth(method = "lm") +
  scale_x_continuous("Clay and fine silt content [%]", expand = c(0,0)) +
  scale_y_continuous("SOC [wt-%]", expand = c(0,0), breaks = seq(0,10,5)) +
  coord_cartesian(xlim = c(0,100), ylim = c(0,10)) +
  own_theme +
  theme(panel.spacing.x = unit(0.5, "cm"), legend.position = "top",
        strip.text = element_text(color = "black"),
        strip.background = element_rect(color = "grey")) +
  scale_color_manual(values = ColorDepth)

ggsave("AfSIS_RefData2_Clay_SOC_Site.jpeg", height = 8, width = 9)
```

pH_{H2O} vs Al_{ox}, Ca_{ex}, and Fe_{ox} by MAP:
Figure 3d in the manuscript.

Create new dataset for grouped pH_{H2O} (n = 20) and calculate the mean for each group for Al_{ox}, Ca_{ex}, and Fe_{ox}

```
AfSIS_pH_Alox_Caex <- AfSIS_RefData_noNA %>%
  dplyr::select(pH, Caex, Alox, Feox, MAP) %>%
  # convert units wt-% in g/kg (*10) for better visualization
  dplyr::mutate(Alox = Alox*10,
                Feox = Feox*10) %>%
  dplyr::mutate(pH_group = Hmisc::cut2(pH, g = 20, levels.mean = TRUE, digits = 2)) %>%
```

```
  group_by(pH_group) %>%
  summarise(Caex = mean(Caex),
            Alox = mean(Alox),
            Feox = mean(Feox),
            MAP = mean(MAP))

AfSIS_pH_Alox_Feox_Caex_MAP <- AfSIS_RefData_noNA %>%
  ggplot(aes(x = pH)) +
  geom_line(data = AfSIS_pH_Alox_Caex, color = "black",
            aes(y = Caex, x = as.numeric(paste(pH_group))), size = 1) +
  geom_point(data = AfSIS_pH_Alox_Caex,
             aes(y = Caex, x = as.numeric(paste(pH_group)), color = MAP),
             size = 4, shape = 15) +
  geom_line(data = AfSIS_pH_Alox_Caex, color = "black",
            aes(y = Alox, x = as.numeric(paste(pH_group))), size = 1) +
  geom_point(data = AfSIS_pH_Alox_Caex,
             aes(y = Alox, x = as.numeric(paste(pH_group)), color = MAP),
             size = 4, shape = 16)  +
  geom_line(data = AfSIS_pH_Alox_Caex, color = "black",
            aes(y = Feox, x = as.numeric(paste(pH_group))), size = 1) +
  geom_point(data = AfSIS_pH_Alox_Caex,
             aes(y = Feox, x = as.numeric(paste(pH_group)), color = MAP),
             size = 4, shape = 17) +
  own_theme +
  theme(legend.position = c(0.15,0.75)) +
  scale_y_continuous(expression("Ca"[ex]~", Al"[ox] ~ "and Fe"[ox]),
                     expand = c(0,0)) +
  scale_x_continuous(expand = c(0,0), breaks = seq(4,9,1)) +
  coord_cartesian(xlim = c(4,9), ylim = c(0,30)) +
  xlab(expression("pH"[H20])) +
  annotate(geom = "text", x = 7.0, y = 22,
           label = expression("Ca"[ex] ~ "[cmol/kg]"), size = 5) +
  annotate(geom = "text", x = 6.5, y = 6,
           label = expression("Al"[ox] ~ "[g/kg]"), size = 5) +
  annotate(geom = "text", x = 6.5, y = 0.79,
           label = expression("Fe"[ox] ~ "[g/kg]"), size = 5) +
  scale_color_distiller(type = "seq", palette = "Blues", direction = 1,
                        name = "MAP [mm]")
```

pH_{H2O} vs Al_{ox}, Ca_{ex}, and Fe_{ox} by CIA:
Figure 4b in the manuscript.

Create new dataset according to grouped pH_{H2O} (n = 20) and mean of Al_{ox}, Ca_{ex}, and Fe_{ox}

```
AfSIS_pH_Alox_Caex_CIA <- AfSIS_RefData_noNA %>%
  dplyr::select(pH, Caex, Alox, Feox, CIA) %>%
  # convert units wt-% in g/kg (*10)
  mutate(Alox = Alox*10,
```

```
         Feox = Feox*10) %>%
  mutate(pH_group = Hmisc::cut2(pH, g = 20, levels.mean = TRUE, digits = 2)) %>%
  group_by(pH_group) %>%
  summarise(Caex = mean(Caex),
            Alox = mean(Alox),
            Feox = mean(Feox),
            CIA = mean(CIA))

AfSIS_pH_Alox_Feox_Caex_CIA <- AfSIS_RefData_noNA %>%
  ggplot(aes(x = pH)) +
  geom_line(data = AfSIS_pH_Alox_Caex_CIA, color = "black",
            aes(y = Caex, x = as.numeric(paste(pH_group))), size = 1) +
  geom_point(data = AfSIS_pH_Alox_Caex_CIA,
             aes(y = Caex, x = as.numeric(paste(pH_group)), color = CIA),
             size = 4, shape = 15) +
  geom_line(data = AfSIS_pH_Alox_Caex_CIA, color = "black",
            aes(y = Alox, x = as.numeric(paste(pH_group))), size = 1) +
  geom_point(data = AfSIS_pH_Alox_Caex_CIA,
             aes(y = Alox, x = as.numeric(paste(pH_group)), color = CIA),
             size = 4, shape = 16)  +
  geom_line(data = AfSIS_pH_Alox_Caex_CIA, color = "black",
            aes(y = Feox, x = as.numeric(paste(pH_group))), size = 1) +
  geom_point(data = AfSIS_pH_Alox_Caex_CIA,
             aes(y = Feox, x = as.numeric(paste(pH_group)), color = CIA),
             size = 4, shape = 17) +
  own_theme +
  theme(legend.position = c(0.1,0.8)) +
  scale_y_continuous(expression("Ca"[ex]~", Al"[ox] ~ "and Fe"[ox]),
                     expand = c(0,0)) +
  scale_x_continuous(expand = c(0,0), breaks = seq(4,9,1)) +
  coord_cartesian(xlim = c(4,9), ylim = c(0,30)) +
  xlab(expression("pH"[H20])) +
  annotate(geom = "text", x = 6.5, y = 22,
           label = expression("Ca"[ex] ~ "[cmol/kg]"), size = 5) +
  annotate(geom = "text", x = 6.5, y = 6,
           label = expression("Al"[ox] ~ "[g/kg]"), size = 5) +
  annotate(geom = "text", x = 6.5, y = 0.79,
           label = expression("Fe"[ox] ~ "[g/kg]"), size = 5) +
  scale_color_distiller(type = "seq", palette = "YlOrBr", direction = 1,
                        name = "CIA [%]")
```

Ca_{ex} vs CORG by pH groups:
Figure 3c in the manuscript.

```
AfSIS_Caex <- AfSIS_RefData_noNA %>%
  dplyr::select(CORG, Caex, pH_Class) %>%
  group_by(pH_Class) %>%
```

```
  mutate(Caex_group = cut_number(Caex, n = 20)) %>%
  group_by(Caex_group, pH_Class) %>%
  summarise(CORG = mean(CORG),
            Caex = mean(Caex))

AfSIS_Caex_SOC_pH <- AfSIS_RefData_noNA %>%
  ggplot(aes(y = CORG, x = Caex)) +
  geom_point(size = 2, alpha = 0.1, aes(color = pH_Class)) +
  geom_line(data = AfSIS_Caex, aes(y = CORG, x = Caex,color = pH_Class), size = 1) +
  geom_point(data = AfSIS_Caex, aes(y = CORG, x = Caex,color = pH_Class), size = 3) +
  own_theme +
  theme(legend.position = c(0.85,0.8)) +
  scale_y_continuous("SOC [wt-%]", expand = c(0,0)) +
  scale_x_continuous(expression("Ca"[ex] ~ "[cmol/kg]"), expand = c(0,0)) +
  coord_cartesian(ylim = c(0,5), xlim = c(0,50)) +
  scale_color_brewer(palette = "RdBu", name = "pH classes")
```

6 Linear mixed modelling

This chapter contains all the code related to linear-mixed models, including normalization and standardization of the data, as well as all models for the entire dataset and for all the sub-groups (depth, pH_{H20}, seasonality, CIA). For each model, all model results are shown (including the full and all step-wise reduced models).

6.1 Entire dataset models

This section is dealing with the linear-models for the entire dataset (n = 1,601). The results of the reduced model can be found in *Table 2* and of full model in *Table A6* in the supplement of the manuscript.

Normalize and standardize data:

```
AfSIS_RefData_noNA_normal <-
  AfSIS_RefData_noNA %>%
  dplyr::select(-Longitude, -Latitude) %>%
  mutate_if(is.numeric, ~predict(bestNormalize::boxcox(.x)))
```

Build model:

```
LMM_full <- nlme::lme(CORG ~ MAT + AridityIndex + pH*Alox + LandCover +
                        Feox + Clay_8um + CIA + Caex + Depth,
                      random = ~1|Site/Cluster/Profile,
                      data = AfSIS_RefData_noNA_normal)
```

Autocorrelation (VIF):

```r
vif_full <- as.data.frame(car::vif(LMM_full))
print(vif_full)
```

```
##                   GVIF Df GVIF^(1/(2*Df))
## MAT           1.029834  1        1.014807
## AridityIndex 1.124626  1        1.060484
## pH            1.641419  1        1.281179
## Alox          1.983665  1        1.408426
## LandCover     1.054606  3        1.008901
## Feox          1.902116  1        1.379172
## Clay_8um      1.460759  1        1.208619
## CIA           1.288424  1        1.135088
## Caex          1.895088  1        1.376622
## Depth         1.166129  1        1.079874
## pH:Alox       1.079559  1        1.039018
```

```r
vif_full_test <- max(vif_full$"GVIF^(1/(2*Df))") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Step-wise reduction based on AIC and p-value:

```r
summary(LMM_full)$AIC
```

```
## [1] 1632.488
```

```r
summary(LMM_full)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.1419292	0.0645925	783	-2.1972999	0.0282913
MAT	-0.1520393	0.0459472	783	-3.3089981	0.0009791
AridityIndex	-0.4101138	0.0574596	783	-7.1374235	0.0000000
pH	-0.4065248	0.0302561	342	-13.4361173	0.0000000
Alox	0.2843170	0.0236496	342	12.0220620	0.0000000
LandCoverForest	0.2555406	0.0797027	783	3.2061709	0.0013997
LandCoverGrassland	0.0363020	0.0410789	783	0.8837152	0.3771212
LandCoverOther	0.0856768	0.0348220	783	2.4604240	0.0140922
Feox	0.0623942	0.0274867	342	2.2699782	0.0238308
Clay_8um	-0.0054301	0.0168912	342	-0.3214731	0.7480482
CIA	-0.2324502	0.0200465	342	-11.5955632	0.0000000
Caex	0.5405329	0.0298916	342	18.0830818	0.0000000
Depth.L	-0.3095306	0.0117708	342	-26.2964239	0.0000000
pH:Alox	-0.1597631	0.0169125	342	-9.4464663	0.0000000

```
LMM_reduced <- update(LMM_full, ~.-Clay_8um)
summary(LMM_reduced)$AIC
```

```
## [1] 1624.265
```

```
summary(LMM_reduced)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.1413159	0.0644593	783	-2.1923266	0.0286496
MAT	-0.1519753	0.0458961	783	-3.3112875	0.0009712
AridityIndex	-0.4101937	0.0573737	783	-7.1495127	0.0000000
pH	-0.4057695	0.0301780	343	-13.4458514	0.0000000
Alox	0.2830677	0.0234081	343	12.0927256	0.0000000
LandCoverForest	0.2553035	0.0796922	783	3.2036193	0.0014120
LandCoverGrassland	0.0368780	0.0410433	783	0.8985145	0.3691875
LandCoverOther	0.0862308	0.0347812	783	2.4792318	0.0133760
Feox	0.0614111	0.0272637	343	2.2524835	0.0249235
CIA	-0.2340102	0.0194657	343	-12.0216688	0.0000000
Caex	0.5380534	0.0288770	343	18.6326233	0.0000000
Depth.L	-0.3104809	0.0113850	343	-27.2710022	0.0000000
pH:Alox	-0.1596374	0.0169077	343	-9.4417132	0.0000000

```
LMM_reduced_1 <- update(LMM_reduced, ~.-Feox)
summary(LMM_reduced_1)$AIC
```

```
## [1] 1621.876
```

```
summary(LMM_reduced_1)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.1471614	0.0644720	783	-2.2825626	0.0227235
MAT	-0.1628994	0.0456497	783	-3.5684615	0.0003809
AridityIndex	-0.4187903	0.0572911	783	-7.3098626	0.0000000
pH	-0.4204675	0.0295950	344	-14.2073666	0.0000000
Alox	0.3140016	0.0191773	344	16.3735812	0.0000000
LandCoverForest	0.2537377	0.0795618	783	3.1891908	0.0014835
LandCoverGrassland	0.0376829	0.0409780	783	0.9195893	0.3580706
LandCoverOther	0.0837192	0.0347181	783	2.4114010	0.0161206
CIA	-0.2379565	0.0194082	344	-12.2606056	0.0000000
Caex	0.5490826	0.0284990	344	19.2667283	0.0000000
Depth.L	-0.3118909	0.0114405	344	-27.2618862	0.0000000
pH:Alox	-0.1569362	0.0168742	344	-9.3003643	0.0000000

```
LMM_reduced_2 <- update(LMM_reduced_1, ~.-LandCover)
summary(LMM_reduced_2)$AIC
```

```
## [1] 1616.767
```

```
summary(LMM_reduced_2)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.0782613	0.0604227	786	-1.295230	0.1956214
MAT	-0.1685294	0.0459085	786	-3.670984	0.0002580
AridityIndex	-0.4229021	0.0570996	786	-7.406393	0.0000000
pH	-0.4238894	0.0295945	344	-14.323260	0.0000000
Alox	0.3146437	0.0192463	344	16.348244	0.0000000
CIA	-0.2424201	0.0194347	344	-12.473568	0.0000000
Caex	0.5416717	0.0285135	344	18.997045	0.0000000
Depth.L	-0.3129596	0.0114556	344	-27.319247	0.0000000
pH:Alox	-0.1557690	0.0169164	344	-9.208143	0.0000000

No further improvements.

R^2 results of final model:

```
piecewiseSEM::rsquared(LMM_reduced_2)
```

```
##   Response   family     link method  Marginal Conditional
## 1     CORG gaussian identity   none 0.7100123    0.936259
```

```
#tab_model(LMM_reduced_2)
```

6.2 Clay and land cover models

This section provides the code for the clay and land cover-only models for the entire dataset. The R^2 results of those models can also be found in *Table A11* in the supplement of the manuscript.

Clay-only model:

```
LMM_Clay <- nlme::lme(CORG ~ Clay_8um, random = ~1|Site/Cluster/Profile,
                      data = AfSIS_RefData_noNA_normal)

summary(LMM_Clay)$AIC
```

```
## [1] 2987.603
```

```
summary(LMM_Clay)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.4042329	0.1141610	788	-3.540901	0.0004222
Clay_8um	0.0873278	0.0210313	349	4.152285	0.0000414

R^2 results of final model:

```
##   Response   family     link method    Marginal Conditional
## 1     CORG gaussian identity   none 0.008298685   0.7157964
```

Land cover-only model:

```
## [1] 3005.719
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.5054770	0.1264889	785	-3.9962173	0.0000704
LandCoverForest	0.2853913	0.1153787	785	2.4735186	0.0135896
LandCoverGrassland	0.0429743	0.0588116	785	0.7307122	0.4651730
LandCoverOther	0.0593606	0.0499651	785	1.1880400	0.2351770

R^2 results of final model:

```
##   Response   family     link method    Marginal Conditional
## 1     CORG gaussian identity   none 0.008046403   0.7538204
```

Clay + land cover-only model:

```
## [1] 2996.575
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.4742181	0.1189306	785	-3.9873531	0.0000731
Clay_8um	0.0905949	0.0210728	349	4.2991416	0.0000223
LandCoverForest	0.2940009	0.1134064	785	2.5924538	0.0097067
LandCoverGrassland	0.0530997	0.0580013	785	0.9154911	0.3602150
LandCoverOther	0.0761469	0.0494644	785	1.5394278	0.1241029

R^2 results of final model:

```
##   Response   family     link method   Marginal Conditional
## 1     CORG gaussian identity   none 0.02105504   0.7185834
```

6.3 Depth models

This section contains the two models for the topsoil and subsoil samples, respectively. The results of the two full models can also be found in *Table A7* in the supplement of the manuscript.

Normalize and standardize each sub-group:

```
AfSIS_depth_normal <- AfSIS_RefData_noNA %>%
  dplyr::select(-Longitude, -Latitude) %>%
  group_by(Depth) %>%
  mutate_if(is.numeric, ~predict(bestNormalize::boxcox(.x)))
```

Topsoil:

Build model:

```
LMM_full_TOP <- nlme::lme(CORG ~ MAT + AridityIndex + pH*Alox + LandCover +
                            Feox + Clay_8um + CIA + Caex, random = ~1|Site/Cluster,
                          data = AfSIS_depth_normal %>%
                            filter(Depth == "Topsoil"))
```

Autocorrelation:

```
vif_top <- as.data.frame(car::vif(LMM_full_TOP))
vif_top_test <- max(vif_top$"GVIF^(1/(2*Df))") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Step-wise reduction based on AIC:

```
summary(LMM_full_TOP)$AIC
```

```
## [1] 928.1354
```

```
summary(LMM_full_TOP)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.2050495	0.0712787	419	-2.8767292	0.0042235
MAT	-0.1495467	0.0513365	419	-2.9130650	0.0037705
AridityIndex	-0.3519458	0.0646002	419	-5.4480581	0.0000001
pH	-0.6128103	0.0446794	419	-13.7157294	0.0000000
Alox	0.3397157	0.0320271	419	10.6071275	0.0000000

	Value	Std.Error	DF	t-value	p-value
LandCoverForest	0.2667841	0.0983188	419	2.7134600	0.0069331
LandCoverGrassland	0.0537904	0.0529959	419	1.0149906	0.3106960
LandCoverOther	0.0854616	0.0441241	419	1.9368448	0.0534362
Feox	0.0021451	0.0355804	419	0.0602901	0.9519534
Clay_8um	-0.0725118	0.0233201	419	-3.1094067	0.0020026
CIA	-0.2772680	0.0279504	419	-9.9199979	0.0000000
Caex	0.6385578	0.0415344	419	15.3742010	0.0000000
pH:Alox	-0.1927627	0.0244570	419	-7.8816832	0.0000000

```
LMM_reduced_TOP <- update(LMM_full_TOP, ~.-Feox)
summary(LMM_reduced_TOP)$AIC
```

```
## [1] 921.3044
```

```
summary(LMM_reduced_TOP)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.2052017	0.0711801	420	-2.882852	0.0041433
MAT	-0.1498909	0.0509738	420	-2.940549	0.0034574
AridityIndex	-0.3522019	0.0643921	420	-5.469642	0.0000001
pH	-0.6134695	0.0433406	420	-14.154604	0.0000000
Alox	0.3407197	0.0274596	420	12.408022	0.0000000
LandCoverForest	0.2667365	0.0982473	420	2.714949	0.0069019
LandCoverGrassland	0.0538735	0.0529411	420	1.017611	0.3094486
LandCoverOther	0.0853276	0.0440416	420	1.937432	0.0533627
Clay_8um	-0.0723983	0.0232177	420	-3.118233	0.0019446
CIA	-0.2774901	0.0276763	420	-10.026282	0.0000000
Caex	0.6388713	0.0411916	420	15.509758	0.0000000
pH:Alox	-0.1926123	0.0242981	420	-7.927052	0.0000000

```
LMM_reduced_TOP_1 <- update(LMM_reduced_TOP, ~.-LandCover)
summary(LMM_reduced_TOP_1)$AIC
```

```
## [1] 913.0747
```

```
summary(LMM_reduced_TOP_1)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.1353352	0.0666487	423	-2.030576	0.0429230
MAT	-0.1567947	0.0518436	423	-3.024380	0.0026432

	Value	Std.Error	DF	t-value	p-value
AridityIndex	-0.3534063	0.0647416	423	-5.458717	0.0000001
pH	-0.6187162	0.0431640	423	-14.334071	0.0000000
Alox	0.3456221	0.0275308	423	12.554025	0.0000000
Clay_8um	-0.0741997	0.0232548	423	-3.190721	0.0015249
CIA	-0.2855869	0.0276718	423	-10.320493	0.0000000
Caex	0.6312843	0.0412975	423	15.286269	0.0000000
pH:Alox	-0.1912469	0.0243646	423	-7.849365	0.0000000

No further improvements.

R^2 results of final topsoil model:

```
##   Response   family     link method  Marginal Conditional
## 1     CORG gaussian identity   none 0.7098296   0.8744042
```

Subsoil:

Build model:

```
LMM_full_BOT <- nlme::lme(CORG ~ MAT + AridityIndex + pH*Alox + LandCover +
                            Feox + Clay_8um + CIA + Caex, random = ~1|Site/Cluster,
                          data = AfSIS_depth_normal %>%
                            filter(Depth == "Subsoil"))
```

Autocorrelation:

```
vif_bot <- as.data.frame(vif(LMM_full_BOT))
vif_bot_test <- max(vif_bot$"GVIF^(1/(2*Df))") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Step-wise reduction based on AIC:

```
summary(LMM_full_BOT)$AIC
```

```
## [1] 1089.782
```

```
summary(LMM_full_BOT)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.1004996	0.0782441	440	-1.2844371	0.1996649
MAT	-0.1639956	0.0546065	440	-3.0032253	0.0028237
AridityIndex	-0.3894825	0.0705728	440	-5.5188734	0.0000001
pH	-0.3393596	0.0462949	440	-7.3303892	0.0000000

	Value	Std.Error	DF	t-value	p-value
Alox	0.3443313	0.0340002	440	10.1273309	0.0000000
LandCoverForest	0.2159896	0.1114796	440	1.9374811	0.0533261
LandCoverGrassland	-0.0104995	0.0569850	440	-0.1842496	0.8539025
LandCoverOther	0.0487985	0.0502703	440	0.9707221	0.3322200
Feox	0.0514275	0.0387077	440	1.3286104	0.1846652
Clay_8um	0.0365188	0.0254182	440	1.4367193	0.1515084
CIA	-0.1371687	0.0314020	440	-4.3681453	0.0000156
Caex	0.4701045	0.0481272	440	9.7679523	0.0000000
pH:Alox	-0.1548247	0.0253546	440	-6.1063695	0.0000000

```
LMM_reduced_BOT <- update(LMM_full_BOT, ~.-LandCover)
summary(LMM_reduced_BOT)$AIC
```

```
## [1] 1077.946
```

```
summary(LMM_reduced_BOT)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.0643617	0.0715354	443	-0.8997193	0.3687584
MAT	-0.1686367	0.0545835	443	-3.0895185	0.0021310
AridityIndex	-0.3928469	0.0698689	443	-5.6226310	0.0000000
pH	-0.3405536	0.0461601	443	-7.3776597	0.0000000
Alox	0.3457936	0.0340477	443	10.1561446	0.0000000
Feox	0.0511256	0.0387318	443	1.3199899	0.1875201
Clay_8um	0.0384737	0.0254251	443	1.5132214	0.1309365
CIA	-0.1392432	0.0314140	443	-4.4325246	0.0000118
Caex	0.4579698	0.0478838	443	9.5641993	0.0000000
pH:Alox	-0.1560702	0.0253448	443	-6.1578727	0.0000000

```
LMM_reduced_BOT_1 <- update(LMM_reduced_BOT, ~.-Feox)
summary(LMM_reduced_BOT_1)$AIC
```

```
## [1] 1073.022
```

```
summary(LMM_reduced_BOT_1)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.0691394	0.0716285	444	-0.9652489	0.3349456
MAT	-0.1777087	0.0542767	444	-3.2741263	0.0011428
AridityIndex	-0.4023367	0.0696629	444	-5.7754822	0.0000000

	Value	Std.Error	DF	t-value	p-value
pH	-0.3503399	0.0455961	444	-7.6835479	0.0000000
Alox	0.3683899	0.0294920	444	12.4911621	0.0000000
Clay_8um	0.0436854	0.0251052	444	1.7400973	0.0825348
CIA	-0.1424806	0.0313219	444	-4.5489108	0.0000070
Caex	0.4675470	0.0473686	444	9.8704038	0.0000000
pH:Alox	-0.1523183	0.0251781	444	-6.0496243	0.0000000

```
LMM_reduced_BOT_2 <- update(LMM_reduced_BOT_1, ~.-Clay_8um)
summary(LMM_reduced_BOT_2)$AIC
```

```
## [1] 1068.439
```

```
summary(LMM_reduced_BOT_2)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.0767823	0.0727106	445	-1.055999	0.2915418
MAT	-0.1773914	0.0548858	445	-3.232006	0.0013206
AridityIndex	-0.4068978	0.0706016	445	-5.763298	0.0000000
pH	-0.3572395	0.0455001	445	-7.851401	0.0000000
Alox	0.3818116	0.0285372	445	13.379441	0.0000000
CIA	-0.1308342	0.0306900	445	-4.263089	0.0000246
Caex	0.4949417	0.0447114	445	11.069703	0.0000000
pH:Alox	-0.1529919	0.0251883	445	-6.073922	0.0000000

No further improvements.

R^2 results of final subsoil model:

```
##   Response   family     link method Marginal Conditional
## 1     CORG gaussian identity   none 0.656059   0.8536112
```

Plotting:
Figure A4 in the supplement of the manuscript.

```
ColorDepth <- brewer.pal(11, "BrBG")[c(1,3)]
plot_models(LMM_TOP_final, LMM_BOT_final, m.labels = c("Topsoil", "Subsoil"),
            ci.lvl = 0.95,
            legend.title = "Depth:", dot.size = 3, vline.color = "black",
            axis.labels = c(expression("pH*Al"[ox]),
                            expression("Ca"[ex]), "CIA", "Clay",
                            expression("Al"[ox]), "pH",
                            "PET/MAP",  "MAT")) +
```

```
  scale_y_continuous("Coefficient", expand = c(0,0), breaks = seq(-0.8,0.8,0.2),
                     limits = c(-0.75,0.75)) +
  own_theme +
  theme(legend.position = "top",
        legend.box.spacing = unit(0.1, "cm")) +
  scale_color_manual(values = ColorDepth)

ggsave("LMM_Depth.pdf", dpi = 300, height = 4, width = 8.5)
```

6.4 pH_{H2O} models

This section contains the code for the models for the four pH groups (strongly acid, moderately acid, neutral, and alkaline). The results of the full models can also be foung in *Table A8* in the supplement of the manuscript.

Normalize and standardize by sub-groups:

```
AfSIS_pH_normal <- AfSIS_RefData_noNA %>%
  dplyr::select(-Longitude, -Latitude) %>%
  group_by(pH_Class) %>%
  mutate_if(is.numeric, ~predict(bestNormalize::boxcox(.x)))
```

```
## `mutate_if()` ignored the following grouping variables:
## Column `pH_Class`
```

Strongly acid:

Build model:

```
LMM_full_acid <- lme(CORG ~ MAT + AridityIndex + pH*Alox + Depth +
                       Feox + Clay_8um + CIA + Caex, random = ~1|Site/Cluster/Profile,
                     data = AfSIS_pH_normal %>%
                       filter(pH_Class == "strongly acid"))
```

Autocorrelation:

```
vif_acid <- as.data.frame(vif(LMM_full_acid))
vif_acid_test <- max(vif_acid$"vif(LMM_full_acid)") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Step-wise reduction based on AIC:

```
summary(LMM_full_acid)$AIC
```

```
## [1] 548.7268
```

```
summary(LMM_full_acid)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.1349109	0.1253984	162	-1.075859	0.2835901
MAT	-0.1444447	0.0609793	162	-2.368751	0.0190268
AridityIndex	-0.2572387	0.0907309	162	-2.835182	0.0051639
pH	-0.1401279	0.0314355	91	-4.457630	0.0000236
Alox	0.5614090	0.0535419	91	10.485407	0.0000000
Depth.L	-0.4243057	0.0291088	91	-14.576558	0.0000000
Feox	0.1388158	0.0447434	91	3.102486	0.0025562
Clay_8um	-0.0854410	0.0323849	91	-2.638295	0.0098010
CIA	-0.1309855	0.0438195	91	-2.989203	0.0035965
Caex	0.1671935	0.0417725	91	4.002473	0.0001276
pH:Alox	-0.0347661	0.0304213	91	-1.142821	0.2561109

```
LMM_reduced_acid <- lme(CORG ~ MAT + AridityIndex + pH + Alox + Depth +
                          Feox + Clay_8um + CIA + Caex, random = ~1|Site/Cluster/Profile,
                        data = AfSIS_pH_normal %>%
                          filter(pH_Class == "strongly acid"))
summary(LMM_reduced_acid)$AIC
```

```
## [1] 542.878
```

```
summary(LMM_reduced_acid)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.1452897	0.1250431	162	-1.161917	0.2469781
MAT	-0.1420613	0.0609462	162	-2.330928	0.0209893
AridityIndex	-0.2507073	0.0905423	162	-2.768952	0.0062799
pH	-0.1339181	0.0309754	92	-4.323368	0.0000389
Alox	0.5399357	0.0500379	92	10.790545	0.0000000
Depth.L	-0.4325396	0.0282878	92	-15.290694	0.0000000
Feox	0.1385206	0.0447065	92	3.098447	0.0025804
Clay_8um	-0.0830429	0.0323105	92	-2.570151	0.0117718
CIA	-0.1389023	0.0432199	92	-3.213850	0.0018065
Caex	0.1574095	0.0409527	92	3.843694	0.0002229

```
LMM_reduced_acid_1 <- update(LMM_reduced_acid, ~.-MAT)
summary(LMM_reduced_acid_1)$AIC
```

```
## [1] 542.3544
```

```
summary(LMM_reduced_acid_1)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.2030562	0.1215998	163	-1.669873	0.0968638
AridityIndex	-0.2445468	0.0902613	163	-2.709322	0.0074629
pH	-0.1304306	0.0310437	92	-4.201514	0.0000613
Alox	0.5479614	0.0502037	92	10.914758	0.0000000
Depth.L	-0.4335382	0.0282574	92	-15.342441	0.0000000
Feox	0.1472282	0.0448613	92	3.281855	0.0014582
Clay_8um	-0.0812167	0.0324267	92	-2.504623	0.0140185
CIA	-0.1495213	0.0434677	92	-3.439827	0.0008767
Caex	0.1538033	0.0411128	92	3.741010	0.0003186

```
LMM_reduced_acid_2 <- update(LMM_reduced_acid_1, ~.-Clay_8um)
summary(LMM_reduced_acid_2)$AIC
```

```
## [1] 541.2526
```

```
summary(LMM_reduced_acid_2)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.1835693	0.1157275	163	-1.586220	0.1146274
AridityIndex	-0.2363839	0.0871302	163	-2.712996	0.0073842
pH	-0.1256666	0.0310870	93	-4.042421	0.0001090
Alox	0.5353042	0.0503744	93	10.626518	0.0000000
Depth.L	-0.4503139	0.0274289	93	-16.417504	0.0000000
Feox	0.1396632	0.0449519	93	3.106948	0.0025070
CIA	-0.1737092	0.0431292	93	-4.027650	0.0001150
Caex	0.1421833	0.0411160	93	3.458098	0.0008224

No further improvements.

R^2 results of final strongly acid model:

```
##   Response   family     link method  Marginal Conditional
## 1     CORG gaussian identity   none 0.6514953    0.923121
```

Moderately acid:

Build model:

```
LMM_full_macid <- lme(CORG ~ MAT + AridityIndex + pH*Alox + Depth +
                        Feox + Clay_8um + CIA + Caex, random = ~1|Site/Cluster/Profile,
                      data = AfSIS_pH_normal %>%
                        filter(pH_Class == "moderately acid"))
```

Autocorrelation:

```
vif_macid <- as.data.frame(vif(LMM_full_macid))
vif_macid_test <- max(vif_macid$"vif(LMM_full_macid)") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Step-wise reduction based on AIC:

```
summary(LMM_full_macid)$AIC
```

```
## [1] 527.3615
```

```
summary(LMM_full_macid)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	0.0000495	0.0628462	158	0.0007875	0.9993727
MAT	-0.1731060	0.0458041	158	-3.7792662	0.0002225
AridityIndex	-0.2711647	0.0451925	158	-6.0002137	0.0000000
pH	-0.1497996	0.0245689	45	-6.0971325	0.0000002
Alox	0.4926671	0.0513635	45	9.5917831	0.0000000
Depth.L	-0.3343285	0.0273668	45	-12.2165813	0.0000000
Feox	-0.0890557	0.0591217	45	-1.5063110	0.1389755
Clay_8um	-0.0255375	0.0406690	45	-0.6279347	0.5332206
CIA	-0.1258528	0.0323826	45	-3.8864292	0.0003315
Caex	0.4602467	0.0445374	45	10.3339451	0.0000000
pH:Alox	-0.0555997	0.0208476	45	-2.6669633	0.0105980

```
LMM_reduced_macid <- update(LMM_full_macid, ~.-Clay_8um)
summary(LMM_reduced_macid)$AIC
```

```
## [1] 521.164
```

```
summary(LMM_reduced_macid)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.0005518	0.0619752	158	-0.0089029	0.9929078
MAT	-0.1696636	0.0451946	158	-3.7540656	0.0002441
AridityIndex	-0.2704394	0.0447513	158	-6.0431662	0.0000000
pH	-0.1488739	0.0244909	46	-6.0787431	0.0000002
Alox	0.4879721	0.0511977	46	9.5311335	0.0000000
Depth.L	-0.3384406	0.0263757	46	-12.8315195	0.0000000
Feox	-0.0962543	0.0576999	46	-1.6681891	0.1020723
CIA	-0.1339061	0.0297236	46	-4.5050442	0.0000453
Caex	0.4516751	0.0416298	46	10.8497939	0.0000000
pH:Alox	-0.0556574	0.0208195	46	-2.6733312	0.0103588

```
LMM_reduced_macid_1 <- update(LMM_reduced_macid, ~.-Feox)
summary(LMM_reduced_macid_1)$AIC
```

```
## [1] 518.0483
```

```
summary(LMM_reduced_macid_1)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	0.0126554	0.0619815	158	0.2041797	0.8384759
MAT	-0.1571754	0.0449148	158	-3.4994085	0.0006060
AridityIndex	-0.2611441	0.0447099	158	-5.8408606	0.0000000
pH	-0.1448157	0.0243964	47	-5.9359342	0.0000003
Alox	0.4455984	0.0448267	47	9.9404581	0.0000000
Depth.L	-0.3386856	0.0263494	47	-12.8536263	0.0000000
CIA	-0.1367902	0.0297411	47	-4.5993581	0.0000321
Caex	0.4213876	0.0375914	47	11.2096783	0.0000000
pH:Alox	-0.0575387	0.0207845	47	-2.7683414	0.0080366

```
LMM_reduced_macid_2 <- lme(CORG ~ MAT + AridityIndex + pH + Alox + Depth +
                             CIA + Caex, random = ~1|Site/Cluster/Profile,
                           data = AfSIS_pH_normal %>%
                             filter(pH_Class == "moderately acid"))
summary(LMM_reduced_macid_2)$AIC
```

```
## [1] 517.683
```

```
summary(LMM_reduced_macid_2)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	0.0219002	0.0618894	158	0.3538608	0.7239150
MAT	-0.1551587	0.0450032	158	-3.4477246	0.0007247
AridityIndex	-0.2704647	0.0447048	158	-6.0500112	0.0000000
pH	-0.1440979	0.0246002	48	-5.8575830	0.0000004
Alox	0.4456531	0.0452343	48	9.8521034	0.0000000
Depth.L	-0.3433341	0.0265366	48	-12.9381484	0.0000000
CIA	-0.1386784	0.0299981	48	-4.6228987	0.0000287
Caex	0.4192525	0.0378481	48	11.0772331	0.0000000

No further improvements.

R^2 results of final moderately acid model:

```
##   Response   family     link method  Marginal Conditional
## 1     CORG gaussian identity   none 0.7461779   0.9287541
```

Neutral:

Build model:

```
LMM_full_neut <- lme(CORG ~ MAT + AridityIndex + pH*Alox + Depth +
                       Feox + Clay_8um + CIA + Caex, random = ~1|Site/Cluster/Profile,
                     data = AfSIS_pH_normal %>%
                       filter(pH_Class == "neutral"))
```

Autocorrelation:

```
vif_neut <- as.data.frame(vif(LMM_full_neut))
vif_neut_test <- max(vif_neut$"vif(LMM_full_neut)") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Step-wise reduction based on AIC:

```
summary(LMM_full_neut)$AIC
```

```
## [1] 475.2514
```

```
summary(LMM_full_neut)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.0547341	0.0794917	161	-0.6885510	0.4920969
MAT	-0.0391820	0.0642065	161	-0.6102495	0.5425570
AridityIndex	-0.3039517	0.0603266	161	-5.0384356	0.0000012

	Value	Std.Error	DF	t-value	p-value
pH	-0.1108360	0.0251544	38	-4.4062254	0.0000831
Alox	0.1914550	0.0476186	38	4.0205940	0.0002656
Depth.L	-0.3120614	0.0276524	38	-11.2851442	0.0000000
Feox	0.0027629	0.0518351	38	0.0533015	0.9577709
Clay_8um	0.1275336	0.0373811	38	3.4117121	0.0015451
CIA	-0.2749479	0.0263540	38	-10.4328736	0.0000000
Caex	0.3623386	0.0457182	38	7.9254800	0.0000000
pH:Alox	-0.0244850	0.0236300	38	-1.0361838	0.3066656

```
LMM_reduced_neut <- update(LMM_full_neut, ~.-Feox)
summary(LMM_reduced_neut)$AIC
```

```
## [1] 469.1713
```

```
summary(LMM_reduced_neut)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.0548439	0.0793398	161	-0.6912541	0.4904014
MAT	-0.0397710	0.0630422	161	-0.6308625	0.5290250
AridityIndex	-0.3043150	0.0598318	161	-5.0861737	0.0000010
pH	-0.1109947	0.0250169	39	-4.4367945	0.0000727
Alox	0.1930917	0.0366187	39	5.2730398	0.0000053
Depth.L	-0.3121255	0.0275977	39	-11.3098172	0.0000000
Clay_8um	0.1278544	0.0368087	39	3.4734846	0.0012731
CIA	-0.2752838	0.0255676	39	-10.7669176	0.0000000
Caex	0.3623918	0.0456571	39	7.9372413	0.0000000
pH:Alox	-0.0245126	0.0235935	39	-1.0389584	0.3052228

```
LMM_reduced_neut_1 <- update(LMM_reduced_neut, ~.-MAT)
summary(LMM_reduced_neut_1)$AIC
```

```
## [1] 463.8723
```

```
summary(LMM_reduced_neut_1)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.0688052	0.0755236	162	-0.9110416	0.3636271
AridityIndex	-0.3058605	0.0592260	162	-5.1642933	0.0000007
pH	-0.1130977	0.0248502	39	-4.5511725	0.0000510
Alox	0.1939518	0.0365327	39	5.3089864	0.0000047

	Value	Std.Error	DF	t-value	p-value
Depth.L	-0.3119690	0.0275974	39	-11.3043020	0.0000000
Clay_8um	0.1286171	0.0367667	39	3.4981909	0.0011867
CIA	-0.2767780	0.0254181	39	-10.8889936	0.0000000
Caex	0.3665539	0.0452149	39	8.1069334	0.0000000
pH:Alox	-0.0254925	0.0235075	39	-1.0844384	0.2848290

```
LMM_reduced_neut_2 <- lme(CORG ~ AridityIndex + pH + Alox + Depth +
                            Clay_8um + CIA + Caex, random = ~1|Site/Cluster/Profile,
                          data = AfSIS_pH_normal %>%
                           filter(pH_Class == "neutral"))
summary(LMM_reduced_neut_2)$AIC
```

```
## [1] 457.3689
```

```
summary(LMM_reduced_neut_2)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.0661349	0.0744219	162	-0.8886489	0.3755094
AridityIndex	-0.3054760	0.0585017	162	-5.2216553	0.0000005
pH	-0.1155922	0.0247627	40	-4.6679997	0.0000339
Alox	0.2029392	0.0354186	40	5.7297399	0.0000011
Depth.L	-0.3127829	0.0276208	40	-11.3241657	0.0000000
Clay_8um	0.1336928	0.0364739	40	3.6654437	0.0007175
CIA	-0.2762151	0.0254304	40	-10.8615904	0.0000000
Caex	0.3640407	0.0451106	40	8.0699559	0.0000000

No further improvements.

R^2 results of final neutral model:

```
##   Response   family     link method  Marginal Conditional
## 1     CORG gaussian identity   none 0.7545873   0.8959072
```

Alkaline:

Build model:

```
LMM_full_alk <- nlme::lme(CORG ~ MAT + AridityIndex + pH*Alox + Depth +
                            Feox + Clay_8um + CIA + Caex, random = ~1|Site/Cluster/Profile,
                          data = AfSIS_pH_normal %>%
                            filter(pH_Class == "alkaline"))
```

Autocorrelation:

```
vif_alk <- as.data.frame(vif(LMM_full_alk))
vif_alk_test <- max(vif_alk$"vif(LMM_full_alk)") < 3.0
```

VIF < 3.0: FALSE -> autocorrelation

Al_{ox} has a VIF of 3.1322418, which is still ok.

Step-wise reduction based on AIC:

```
summary(LMM_full_alk)$AIC
```

```
## [1] 515.5545
```

```
summary(LMM_full_alk)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.1754353	0.1673317	219	-1.0484285	0.2955969
MAT	-0.1556248	0.1489554	219	-1.0447740	0.2972792
AridityIndex	-0.0745645	0.1152549	219	-0.6469532	0.5183397
pH	-0.1167667	0.0259034	72	-4.5077698	0.0000248
Alox	0.0080072	0.0461796	72	0.1733934	0.8628287
Depth.L	-0.2421688	0.0225367	72	-10.7455428	0.0000000
Feox	0.1062214	0.0512996	72	2.0706088	0.0419818
Clay_8um	-0.0319038	0.0326222	72	-0.9779758	0.3313609
CIA	-0.0301993	0.0296115	72	-1.0198511	0.3112137
Caex	0.3139323	0.0435937	72	7.2013283	0.0000000
pH:Alox	-0.0035486	0.0160126	72	-0.2216115	0.8252437

```
LMM_reduced_alk <- update(LMM_full_alk, ~.-Alox)
summary(LMM_reduced_alk)$AIC
```

```
## [1] 509.2693
```

```
summary(LMM_reduced_alk)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.1752404	0.1677527	219	-1.0446354	0.2973432
MAT	-0.1543703	0.1489396	219	-1.0364621	0.3011297
AridityIndex	-0.0740060	0.1154891	219	-0.6408053	0.5223192
pH	-0.1156370	0.0252210	73	-4.5849486	0.0000184
Depth.L	-0.2423270	0.0224670	73	-10.7858972	0.0000000
Feox	0.1120713	0.0380846	73	2.9426910	0.0043591
Clay_8um	-0.0312208	0.0323990	73	-0.9636363	0.3384097

	Value	Std.Error	DF	t-value	p-value
CIA	-0.0303770	0.0295313	73	-1.0286386	0.3070458
Caex	0.3172722	0.0385159	73	8.2374292	0.0000000
pH:Alox	-0.0029374	0.0155847	73	-0.1884773	0.8510255

```
LMM_reduced_alk_1 <- lme(CORG ~ MAT + AridityIndex + pH + Depth + Feox +
                           Clay_8um + CIA + Caex, random = ~1|Site/Cluster/Profile,
                         data = AfSIS_pH_normal %>%
                           filter(pH_Class == "alkaline"))
summary(LMM_reduced_alk_1)$AIC
```

```
## [1] 500.8178
```

```
summary(LMM_reduced_alk_1)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.1742232	0.1678853	219	-1.0377512	0.3005304
MAT	-0.1541115	0.1491046	219	-1.0335797	0.3024727
AridityIndex	-0.0754222	0.1154114	219	-0.6535068	0.5141151
pH	-0.1146293	0.0247378	74	-4.6337633	0.0000151
Depth.L	-0.2427088	0.0223440	74	-10.8623913	0.0000000
Feox	0.1098486	0.0363940	74	3.0183167	0.0034851
Clay_8um	-0.0312704	0.0323169	74	-0.9676189	0.3363866
CIA	-0.0304920	0.0294822	74	-1.0342511	0.3043870
Caex	0.3178152	0.0382915	74	8.2998819	0.0000000

```
LMM_reduced_alk_2 <- update(LMM_reduced_alk_1, ~.-AridityIndex)
summary(LMM_reduced_alk_2)$AIC
```

```
## [1] 496.7544
```

```
summary(LMM_reduced_alk_2)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.1582130	0.1632031	220	-0.9694238	0.3333983
MAT	-0.1815194	0.1400948	220	-1.2956899	0.1964401
pH	-0.1150966	0.0247179	74	-4.6564152	0.0000138
Depth.L	-0.2422488	0.0223443	74	-10.8416448	0.0000000
Feox	0.1121151	0.0362459	74	3.0931791	0.0027932
Clay_8um	-0.0317332	0.0323085	74	-0.9821919	0.3292057
CIA	-0.0298402	0.0294496	74	-1.0132623	0.3142365
Caex	0.3180829	0.0382614	74	8.3134140	0.0000000

```
LMM_reduced_alk_3 <- update(LMM_reduced_alk_2, ~.-Clay_8um)
summary(LMM_reduced_alk_3)$AIC
```

```
## [1] 490.6211
```

```
summary(LMM_reduced_alk_3)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.1556856	0.1618351	220	-0.9620012	0.3371051
MAT	-0.1896306	0.1388802	220	-1.3654254	0.1735139
pH	-0.1133106	0.0246418	75	-4.5983186	0.0000169
Depth.L	-0.2475189	0.0215078	75	-11.5083435	0.0000000
Feox	0.1063483	0.0359841	75	2.9554213	0.0041710
CIA	-0.0312088	0.0294055	75	-1.0613254	0.2919478
Caex	0.2983133	0.0325407	75	9.1674049	0.0000000

```
LMM_reduced_alk_4 <- update(LMM_reduced_alk_3, ~.-CIA)
summary(LMM_reduced_alk_4)$AIC
```

```
## [1] 484.5192
```

```
summary(LMM_reduced_alk_4)$tTable
```

```
##                 Value  Std.Error  DF    t-value      p-value
## (Intercept) -0.1740727 0.16086995 220  -1.082071 2.804055e-01
## MAT         -0.1891420 0.13877823 220  -1.362908 1.743048e-01
## pH          -0.1133266 0.02459510  76  -4.607691 1.609931e-05
## Depth.L     -0.2488926 0.02139597  76 -11.632686 1.503284e-18
## Feox         0.1092344 0.03581110  76   3.050293 3.146781e-03
## Caex         0.2874581 0.03096474  76   9.283401 3.832476e-14
```

```
LMM_reduced_alk_5 <- update(LMM_reduced_alk_4, ~.-MAT)
summary(LMM_reduced_alk_5)$AIC
```

```
## [1] 482.2546
```

```
summary(LMM_reduced_alk_5)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.1076111	0.1531407	221	-0.7026939	0.4829860
pH	-0.1158791	0.0245821	76	-4.7139633	0.0000108
Depth.L	-0.2485380	0.0214418	76	-11.5912592	0.0000000
Feox	0.1126468	0.0358414	76	3.1429251	0.0023854
Caex	0.2902036	0.0310059	76	9.3596130	0.0000000

No further improvements.

R^2 results of final alkaline model:

```
##   Response   family     link method  Marginal Conditional
## 1     CORG gaussian identity   none 0.2784573   0.9389851
```

Plotting:
Figure 3a in the manuscript.

```
LMM_pH <- plot_models(LMM_acid_final, LMM_macid_final, LMM_neut_final,
                      LMM_alk_final, m.labels = c("str. acid", "mod. acid",
                                                  "neut.", "alk."),
                      axis.labels = c("Clay", "MAT", expression("Ca"[ex]), "CIA",
                                      expression("Fe"[ox]), "Depth",
                                      expression("Al"[ox], "pH", "PET/MAP")),
                      legend.title = "pH classes:", vline.color = "black",
                      dot.size = 3) +
  scale_y_continuous("Coefficient", expand = c(0,0), breaks = seq(-0.8,0.8,0.2),
                     limits = c(-0.75,0.75)) +
  own_theme +
  theme(legend.position = "top", legend.box.spacing = unit(0.1, "cm")) +
  scale_color_brewer(palette = "RdBu", direction = -1)
```

6.5 Seasonality models

This section provides all the code for the modelling of the three seasonality groups (0, 1-3, and 4-7 number of wet months). The results of the full models can also be found in *Table A9* in the supplement of the manuscript.

Normalize and standardize by sub-groups:

```
AfSIS_Seas_normal <- AfSIS_RefData_noNA %>%
  dplyr::select(-Longitude, -Latitude) %>%
  group_by(wet_month_Class) %>%
  mutate_if(is.numeric, ~predict(bestNormalize::boxcox(.x)))
```

0 number of wet months:

Build model:

```
LMM_full_0 <- nlme::lme(CORG ~ MAT + AridityIndex + pH*Alox + Depth +
                          Feox + Clay_8um + CIA + Caex, random = ~1|Site/Cluster/Profile,
                        data = AfSIS_Seas_normal %>%
                          filter(wet_month_Class == "0"))
```

Autocorrelation:

```
vif_0 <- as.data.frame(vif(LMM_full_0))
vif_0_test <- max(vif_0$"vif(LMM_full_0)") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Step-wise reduction based on AIC and p-value:

```
summary(LMM_full_0)$AIC
```

```
## [1] 712.2512
```

```
summary(LMM_full_0)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	0.0243566	0.0946565	267	0.2573155	0.7971334
MAT	-0.2606000	0.0798787	267	-3.2624450	0.0012483
AridityIndex	-0.2858323	0.0789145	267	-3.6220485	0.0003497
pH	-0.1120095	0.0380481	109	-2.9438906	0.0039617
Alox	0.0077581	0.0450699	109	0.1721354	0.8636503
Depth.L	-0.2775151	0.0184880	109	-15.0105794	0.0000000
Feox	0.1346050	0.0446234	109	3.0164637	0.0031831
Clay_8um	0.0212592	0.0330137	109	0.6439492	0.5209611
CIA	-0.0822891	0.0290217	109	-2.8354368	0.0054560
Caex	0.4788380	0.0477108	109	10.0362664	0.0000000
pH:Alox	-0.0205063	0.0236497	109	-0.8670838	0.3878009

```
LMM_reduced_0 <- update(LMM_full_0, ~.-Alox)
summary(LMM_reduced_0)$AIC
```

```
## [1] 705.9185
```

```
summary(LMM_reduced_0)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	0.0242814	0.0944003	267	0.2572173	0.7972092
MAT	-0.2588011	0.0789979	267	-3.2760491	0.0011919
AridityIndex	-0.2857689	0.0787134	267	-3.6304984	0.0003390
pH	-0.1117556	0.0379912	110	-2.9416215	0.0039818
Depth.L	-0.2775245	0.0184626	110	-15.0317140	0.0000000
Feox	0.1396126	0.0339172	110	4.1162795	0.0000747
Clay_8um	0.0220535	0.0326946	110	0.6745302	0.5013895
CIA	-0.0818500	0.0289077	110	-2.8314220	0.0055117
Caex	0.4818807	0.0439683	110	10.9597223	0.0000000
pH:Alox	-0.0204315	0.0236241	110	-0.8648598	0.3889979

```
LMM_reduced_0_1 <- update(LMM_reduced_0, ~.-Clay_8um)
summary(LMM_reduced_0_1)$AIC
```

```
## [1] 699.3575
```

```
summary(LMM_reduced_0_1)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	0.0253555	0.0949274	267	0.2671037	0.7895955
MAT	-0.2568925	0.0793345	267	-3.2380916	0.0013555
AridityIndex	-0.2887657	0.0790148	267	-3.6545756	0.0003101
pH	-0.1126079	0.0379927	111	-2.9639313	0.0037177
Depth.L	-0.2748104	0.0180609	111	-15.2157728	0.0000000
Feox	0.1446496	0.0330249	111	4.3800135	0.0000270
CIA	-0.0803350	0.0287220	111	-2.7969905	0.0060815
Caex	0.4967654	0.0383959	111	12.9379732	0.0000000
pH:Alox	-0.0206444	0.0236252	111	-0.8738284	0.3840983

```
LMM_reduced_0_2 <- lme(CORG ~ MAT + AridityIndex + pH + Depth +
                         Feox + CIA + Caex, random = ~1|Site/Cluster/Profile,
                       data = AfSIS_Seas_normal %>%
                         filter(wet_month_Class == "0"))
summary(LMM_reduced_0_2)$AIC
```

```
## [1] 692.4554
```

```
summary(LMM_reduced_0_2)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	0.0162833	0.0925949	267	0.175855	0.8605411
MAT	-0.2609585	0.0777946	267	-3.354457	0.0009105
AridityIndex	-0.2941867	0.0772788	267	-3.806821	0.0001746
pH	-0.1010320	0.0356121	112	-2.837014	0.0054065
Depth.L	-0.2743198	0.0180140	112	-15.228136	0.0000000
Feox	0.1433240	0.0329686	112	4.347284	0.0000305
CIA	-0.0789873	0.0286914	112	-2.752990	0.0068914
Caex	0.4923724	0.0380751	112	12.931606	0.0000000

```
LMM_reduced_0_3 <- update(LMM_reduced_0_2, ~.-CIA)
summary(LMM_reduced_0_3)$AIC
```

```
## [1] 692.4362
```

```
summary(LMM_reduced_0_3)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.0087966	0.0856697	267	-0.1026809	0.9182933
MAT	-0.2735194	0.0727328	267	-3.7606054	0.0002083
AridityIndex	-0.2822467	0.0724466	267	-3.8959272	0.0001237
pH	-0.0752337	0.0344258	113	-2.1853841	0.0309244
Depth.L	-0.2814357	0.0177314	113	-15.8721193	0.0000000
Feox	0.1513919	0.0330280	113	4.5837406	0.0000119
Caex	0.4718068	0.0374763	113	12.5894666	0.0000000

```
LMM_reduced_0_4 <- update(LMM_reduced_0_3, ~.-pH)
summary(LMM_reduced_0_4)$AIC
```

```
## [1] 690.1164
```

```
summary(LMM_reduced_0_4)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	0.0243046	0.0893074	267	0.2721457	0.7857203
MAT	-0.2709510	0.0766309	267	-3.5357913	0.0004791
AridityIndex	-0.2892826	0.0763204	267	-3.7903710	0.0001860
Depth.L	-0.2836820	0.0174948	114	-16.2151837	0.0000000
Feox	0.1627454	0.0328011	114	4.9615763	0.0000025
Caex	0.4474529	0.0361012	114	12.3944235	0.0000000

No further improvements.

R^2 results of final 0 wet months model:

```
##   Response   family     link method  Marginal Conditional
## 1     CORG gaussian identity   none 0.6989493   0.9532026
```

1-3 number of wet months:

Build model:

```
LMM_full_1_3 <- lme(CORG ~ MAT + AridityIndex + pH*Alox + Depth +
                      Feox + Clay_8um + CIA + Caex, random = ~1|Site/Cluster/Profile,
                    data = AfSIS_Seas_normal %>%
                      filter(wet_month_Class == "1-3"))
```

Autocorrelation:

```
vif_1_3 <- as.data.frame(vif(LMM_full_1_3))
vif_1_3_test <- max(vif_1_3$"vif(LMM_full_1_3)") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Step-wise reduction based on AIC and p-value:

```
summary(LMM_full_1_3)$AIC
```

```
## [1] 501
```

```
summary(LMM_full_1_3)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.0329654	0.1625880	179	-0.2027544	0.8395573
MAT	0.1501583	0.1161641	179	1.2926396	0.1978018
AridityIndex	-0.2422563	0.1099741	179	-2.2028484	0.0288824
pH	-0.3194298	0.0391776	69	-8.1533786	0.0000000
Alox	0.2697018	0.0441651	69	6.1066781	0.0000001
Depth.L	-0.4037579	0.0287593	69	-14.0392296	0.0000000
Feox	0.1436763	0.0483079	69	2.9741815	0.0040450
Clay_8um	0.0445039	0.0416953	69	1.0673593	0.2895299
CIA	-0.3195780	0.0279289	69	-11.4425743	0.0000000
Caex	0.5236901	0.0623330	69	8.4014924	0.0000000
pH:Alox	-0.0855703	0.0209466	69	-4.0851688	0.0001170

```
LMM_reduced_1_3 <- update(LMM_full_1_3, ~.-Clay_8um)
summary(LMM_reduced_1_3)$AIC
```

```
## [1] 495.57
```

```
summary(LMM_reduced_1_3)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.0398615	0.1641271	179	-0.2428699	0.8083843
MAT	0.1560099	0.1175636	179	1.3270251	0.1861906
AridityIndex	-0.2396716	0.1112656	179	-2.1540501	0.0325734
pH	-0.3223820	0.0390399	70	-8.2577591	0.0000000
Alox	0.2753479	0.0436867	70	6.3027795	0.0000000
Depth.L	-0.3940569	0.0272229	70	-14.4751795	0.0000000
Feox	0.1555507	0.0474161	70	3.2805439	0.0016169
CIA	-0.3113071	0.0266192	70	-11.6948508	0.0000000
Caex	0.5492504	0.0577355	70	9.5132220	0.0000000
pH:Alox	-0.0884761	0.0206448	70	-4.2856302	0.0000571

```
LMM_reduced_1_3_1 <- update(LMM_reduced_1_3, ~.-MAT)
summary(LMM_reduced_1_3_1)$AIC
```

```
## [1] 492.8483
```

```
summary(LMM_reduced_1_3_1)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.0219627	0.1578191	180	-0.1391636	0.8894765
AridityIndex	-0.1749834	0.0987491	180	-1.7719993	0.0780865
pH	-0.3206019	0.0390803	70	-8.2036770	0.0000000
Alox	0.2779766	0.0436432	70	6.3693036	0.0000000
Depth.L	-0.3962476	0.0272719	70	-14.5295445	0.0000000
Feox	0.1463584	0.0469808	70	3.1152785	0.0026637
CIA	-0.3102753	0.0266620	70	-11.6373643	0.0000000
Caex	0.5427328	0.0575498	70	9.4306658	0.0000000
pH:Alox	-0.0881951	0.0207095	70	-4.2586892	0.0000628

```
LMM_reduced_1_3_2 <- update(LMM_reduced_1_3_1, ~.-AridityIndex)
summary(LMM_reduced_1_3_2)$AIC
```

```
## [1] 491.1185
```

```
summary(LMM_reduced_1_3_2)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.1702087	0.1364493	181	-1.247413	0.2138575
pH	-0.3274333	0.0389294	70	-8.410962	0.0000000
Alox	0.2806378	0.0437048	70	6.421210	0.0000000
Depth.L	-0.3968003	0.0272353	70	-14.569361	0.0000000
Feox	0.1555939	0.0469910	70	3.311144	0.0014717
CIA	-0.3100260	0.0266519	70	-11.632399	0.0000000
Caex	0.5377971	0.0577455	70	9.313227	0.0000000
pH:Alox	-0.0885279	0.0206901	70	-4.278767	0.0000585

No further improvements.

R^2 results of final 1-3 wet months model:

```
##   Response   family     link method  Marginal Conditional
## 1     CORG gaussian identity   none 0.6989559   0.9476797
```

4-7 number of wet months:

Build model:

```
LMM_full_4_7 <- lme(CORG ~ MAT + AridityIndex + pH*Alox + Depth +
                      Feox + Clay_8um + CIA + Caex, random = ~1|Site/Cluster/Profile,
                    data = AfSIS_Seas_normal %>%
                      filter(wet_month_Class == "4-7"))
```

Autocorrelation:

```
vif_4_7 <- as.data.frame(vif(LMM_full_4_7))
vif_4_7_test <- max(vif_4_7$"vif(LMM_full_4_7)") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Step-wise reduction based on AIC and p-value:

```
summary(LMM_full_4_7)$AIC
```

```
## [1] 998.7499
```

```
summary(LMM_full_4_7)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.1153430	0.1564604	336	-0.7372028	0.4615135
MAT	-0.1917925	0.0758218	336	-2.5295156	0.0118787
AridityIndex	-0.3517372	0.1098418	336	-3.2022157	0.0014941
pH	-0.3593308	0.0439949	148	-8.1675560	0.0000000
Alox	0.5619649	0.0458498	148	12.2566373	0.0000000
Depth.L	-0.4199271	0.0238581	148	-17.6010307	0.0000000
Feox	-0.0302996	0.0471693	148	-0.6423594	0.5216337
Clay_8um	-0.0216246	0.0286960	148	-0.7535780	0.4522998
CIA	-0.2071472	0.0430281	148	-4.8142266	0.0000036
Caex	0.4809930	0.0538701	148	8.9287556	0.0000000
pH:Alox	-0.1185691	0.0320293	148	-3.7018950	0.0003013

```
LMM_reduced_4_7 <- update(LMM_full_4_7, ~.-Feox)
summary(LMM_reduced_4_7)$AIC
```

```
## [1] 992.8917
```

```
summary(LMM_reduced_4_7)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.1024803	0.1559021	336	-0.6573377	0.5114138
MAT	-0.1944609	0.0759266	336	-2.5611688	0.0108683
AridityIndex	-0.3537727	0.1101000	336	-3.2131930	0.0014398
pH	-0.3547645	0.0434083	149	-8.1727276	0.0000000
Alox	0.5487403	0.0410351	149	13.3724702	0.0000000
Depth.L	-0.4190092	0.0237991	149	-17.6060578	0.0000000
Clay_8um	-0.0240863	0.0284324	149	-0.8471437	0.3982737
CIA	-0.2049060	0.0428779	149	-4.7788284	0.0000042
Caex	0.4722442	0.0521257	149	9.0597254	0.0000000
pH:Alox	-0.1222609	0.0314927	149	-3.8821962	0.0001551

```
LMM_reduced_4_7_1 <- update(LMM_reduced_4_7, ~.-Clay_8um)
summary(LMM_reduced_4_7_1)$AIC
```

```
## [1] 986.3004
```

```
summary(LMM_reduced_4_7_1)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.0946000	0.1542073	336	-0.6134599	0.5399873
MAT	-0.1933473	0.0757672	336	-2.5518611	0.0111571
AridityIndex	-0.3595757	0.1092833	336	-3.2903073	0.0011069
pH	-0.3522155	0.0432897	150	-8.1362457	0.0000000
Alox	0.5457068	0.0409403	150	13.3293218	0.0000000
Depth.L	-0.4236944	0.0231539	150	-18.2990467	0.0000000
CIA	-0.2174622	0.0406215	150	-5.3533765	0.0000003
Caex	0.4666707	0.0517352	150	9.0203724	0.0000000
pH:Alox	-0.1220004	0.0314784	150	-3.8756869	0.0001585

No further improvements.

R^2 results of final 4-7 wet months model:

```
##   Response   family     link method Marginal Conditional
## 1     CORG gaussian identity   none 0.529278   0.9102283
```

Plotting:
Figure 3b in the manuscript.

```
WMColor <- brewer.pal(6, "Blues")[c(6,4,2)]
LMM_nwetmonth <- plot_models(LMM_0_final, LMM_1_3_final, LMM_4_7_final,
                             m.labels = c("0", "1-3", "4-7"), dot.size = 3,
                             legend.title = "n wet months:",vline.color = "black",
                             axis.labels = c(expression("pH*Al"[ox]),
                                             "CIA",expression("Al"[ox]),
                                             "pH", expression("Ca"[ex]),
                                             expression("Fe"[ox]), "Depth",
                                             "PET/MAP", "MAT")) +
  scale_y_continuous("Coefficient", expand = c(0,0), breaks = seq(-0.8,0.8,0.2),
                     limits = c(-0.75,0.75)) +
  own_theme +
  theme(legend.position = "top", legend.box.spacing = unit(0.1, "cm")) +
  scale_color_manual(values = WMColor)
```

6.6 CIA models

This section contains all the code for the modelling of the two CIA groups. Results of the final models can also be found in *Table A10* in the supplement of the manuscript.

Normalize and standardize by sub-groups:

```
AfSIS_CIA_normal <- AfSIS_RefData_noNA %>%
  dplyr::select(-Longitude, -Latitude) %>%
  group_by(CIA_Class) %>%
  mutate_if(is.numeric, ~predict(bestNormalize::boxcox(.x)))
```

Moderately weathered:

Build model:

```
LMM_full_modweath <- nlme::lme(CORG ~ MAT + AridityIndex + pH*Alox + Depth +
                                 Feox + Clay_8um + CIA + Caex,
                               random = ~1|Site/Cluster/Profile,
                               data = AfSIS_CIA_normal %>%
                                 filter(CIA_Class == "moderate"))
```

Autocorrelation:

```
vif_mod <- as.data.frame(vif(LMM_full_modweath))
vif_mod_test <- max(vif_mod$"vif(LMM_full_modweath)") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Step-wise reduction based on AIC and p-value:

```
summary(LMM_full_modweath)$AIC
```

```
## [1] 916.1969
```

```
summary(LMM_full_modweath)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.0245678	0.0753427	427	-0.3260805	0.7445232
MAT	-0.1085155	0.0679006	427	-1.5981510	0.1107491
AridityIndex	-0.2685272	0.0638504	427	-4.2055711	0.0000317
pH	-0.2189840	0.0318623	118	-6.8728191	0.0000000
Alox	0.0593317	0.0306263	118	1.9372764	0.0551015
Depth.L	-0.3016583	0.0184060	118	-16.3891439	0.0000000
Feox	0.1307222	0.0337614	118	3.8719479	0.0001776
Clay_8um	0.0108413	0.0260025	118	0.4169346	0.6774837
CIA	-0.1237801	0.0166931	118	-7.4150595	0.0000000
Caex	0.4513202	0.0320877	118	14.0651994	0.0000000
pH:Alox	-0.0752125	0.0160596	118	-4.6833420	0.0000076

```
LMM_reduced_modweath <- update(LMM_full_modweath, ~.-Clay_8um)
summary(LMM_reduced_modweath)$AIC
```

```
## [1] 908.9061
```

```
summary(LMM_reduced_modweath)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.0263310	0.0754506	427	-0.3489828	0.7272743
MAT	-0.1076507	0.0679953	427	-1.5832069	0.1141147
AridityIndex	-0.2678930	0.0639885	427	-4.1865828	0.0000344
pH	-0.2199392	0.0317928	119	-6.9178907	0.0000000
Alox	0.0608349	0.0304141	119	2.0002207	0.0477534
Depth.L	-0.3000601	0.0179792	119	-16.6892536	0.0000000
Feox	0.1319645	0.0335675	119	3.9313172	0.0001424
CIA	-0.1224353	0.0163636	119	-7.4821514	0.0000000
Caex	0.4572896	0.0288338	119	15.8595020	0.0000000
pH:Alox	-0.0748637	0.0160223	119	-4.6724810	0.0000079

```
LMM_reduced_modweath_1 <- update(LMM_reduced_modweath, ~.-MAT)
summary(LMM_reduced_modweath_1)$AIC
```

```
## [1] 905.8333
```

```
summary(LMM_reduced_modweath_1)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.0303318	0.0770997	428	-0.3934099	0.6942127
AridityIndex	-0.2735405	0.0651885	428	-4.1961472	0.0000330
pH	-0.2237451	0.0317360	119	-7.0501921	0.0000000
Alox	0.0597849	0.0304300	119	1.9646702	0.0517830
Depth.L	-0.2996162	0.0179767	119	-16.6669536	0.0000000
Feox	0.1382374	0.0333803	119	4.1412879	0.0000649
CIA	-0.1230864	0.0163706	119	-7.5187660	0.0000000
Caex	0.4596571	0.0288178	119	15.9504837	0.0000000
pH:Alox	-0.0767324	0.0159954	119	-4.7971530	0.0000047

```
LMM_reduced_modweath_2 <- update(LMM_reduced_modweath_1, ~.-Alox)
summary(LMM_reduced_modweath_2)$AIC
```

```
## [1] 902.5213
```

```
summary(LMM_reduced_modweath_2)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.0238516	0.0761436	428	-0.3132447	0.7542474
AridityIndex	-0.2737664	0.0645235	428	-4.2428926	0.0000271
pH	-0.2197351	0.0317427	120	-6.9223918	0.0000000
Depth.L	-0.2972110	0.0179539	120	-16.5541558	0.0000000
Feox	0.1776279	0.0268097	120	6.6255088	0.0000000
CIA	-0.1196064	0.0163110	120	-7.3328519	0.0000000
Caex	0.4820655	0.0264945	120	18.1949558	0.0000000
pH:Alox	-0.0821972	0.0157677	120	-5.2130202	0.0000008

No further improvements.

R^2 results of final moderate CIA model:

```
##   Response   family     link method  Marginal Conditional
## 1     CORG gaussian identity   none 0.6423543   0.9129375
```

Highly weathered:

Build model:

```
LMM_full_strweath <- lme(CORG ~ MAT + AridityIndex + pH*Alox + Depth +
                           Feox + Clay_8um + CIA + Caex, random = ~1|Site/Cluster/Profile,
                         data = AfSIS_CIA_normal %>%
                           filter(CIA_Class == "high"))
```

Autocorrelation:

```
vif_strong <- as.data.frame(vif(LMM_full_strweath))
vif_strong_test <- max(vif_strong$"vif(LMM_full_strweath)") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Step-wise reduction based on AIC and p-value:

```
summary(LMM_full_strweath)$AIC
```

```
## [1] 924.2311
```

```
summary(LMM_full_strweath)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.0803066	0.0908110	320	-0.8843265	0.3771839
MAT	-0.2308638	0.0540886	320	-4.2682532	0.0000260

	Value	Std.Error	DF	t-value	p-value
AridityIndex	-0.3582052	0.0707612	320	-5.0621663	0.0000007
pH	-0.3447164	0.0336826	169	-10.2342507	0.0000000
Alox	0.5496618	0.0364844	169	15.0656696	0.0000000
Depth.L	-0.3293189	0.0180743	169	-18.2202368	0.0000000
Feox	-0.0343163	0.0416642	169	-0.8236388	0.4113056
Clay_8um	-0.0235658	0.0238103	169	-0.9897290	0.3237215
CIA	-0.2042625	0.0322386	169	-6.3359581	0.0000000
Caex	0.3623454	0.0390064	169	9.2893902	0.0000000
pH:Alox	-0.1251513	0.0247314	169	-5.0604311	0.0000011

```
LMM_reduced_strweath <- update(LMM_full_strweath, ~.-Feox)
summary(LMM_reduced_strweath)$AIC
```

```
## [1] 918.3856
```

```
summary(LMM_reduced_strweath)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.0724995	0.0901906	320	-0.8038482	0.4220814
MAT	-0.2267812	0.0538536	320	-4.2110680	0.0000331
AridityIndex	-0.3533532	0.0704507	320	-5.0156079	0.0000009
pH	-0.3385511	0.0328452	170	-10.3074688	0.0000000
Alox	0.5352517	0.0322094	170	16.6178920	0.0000000
Depth.L	-0.3289017	0.0180315	170	-18.2404403	0.0000000
Clay_8um	-0.0270654	0.0234280	170	-1.1552597	0.2496053
CIA	-0.2002020	0.0318744	170	-6.2809630	0.0000000
Caex	0.3554017	0.0379518	170	9.3645489	0.0000000
pH:Alox	-0.1274904	0.0245527	170	-5.1925239	0.0000006

```
LMM_reduced_strweath_1 <- update(LMM_reduced_strweath, ~.-Clay_8um)
summary(LMM_reduced_strweath_1)$AIC
```

```
## [1] 911.9758
```

```
summary(LMM_reduced_strweath_1)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.0685767	0.0885840	320	-0.7741428	0.4394179
MAT	-0.2214626	0.0533220	320	-4.1533060	0.0000421
AridityIndex	-0.3533237	0.0696150	320	-5.0753938	0.0000007

	Value	Std.Error	DF	t-value	p-value
pH	-0.3374658	0.0328236	171	-10.2811878	0.0000000
Alox	0.5277073	0.0317301	171	16.6311119	0.0000000
Depth.L	-0.3339579	0.0175108	171	-19.0715875	0.0000000
CIA	-0.2100995	0.0307298	171	-6.8369918	0.0000000
Caex	0.3467363	0.0371362	171	9.3368862	0.0000000
pH:Alox	-0.1266286	0.0245575	171	-5.1564141	0.0000007

No further improvements.

R^2 results of final high CIA model:

```
##   Response   family     link method  Marginal Conditional
## 1     CORG gaussian identity   none 0.6846045    0.938205
```

Plotting:
Figure 4a in the manuscript.

```
LMM_CIA <- plot_models(LMM_modweath_final, LMM_strweath_final,
                       m.labels = c("moderate", "high"), dot.size = 3,
                       legend.title = "Weathering:", vline.color = "black",
                       axis.labels = c(expression("Al"[ox]), "MAT",
                                       expression("pH*Al"[ox]),
                                       expression("Ca"[ex]), "CIA",
                                       expression("Fe"[ox]), "Depth",
                                       "pH", "PET/MAP")) +
  scale_y_continuous("Coefficient", expand = c(0,0), breaks = seq(-0.8,0.8,0.2),
                     limits = c(-0.75,0.75)) +
  own_theme +
  theme(legend.position = "top", legend.box.spacing = unit(0.1, "cm")) +
  scale_color_manual(values = c("#993404", "#fec44f"))
```

```
## Plotting for manuscript
# Figure 3: pH and seasonality
ggarrange(LMM_pH, LMM_nwetmonth,
          AfSIS_Caex_SOC_pH,  AfSIS_pH_Alox_Feox_Caex_MAP,
          labels = list("a)", "b)", "c)", "d)"), heights = c(1.2,1))

ggsave("AfSIS_RefData2_Caex_fig.jpeg", height = 8, width = 11, dpi = 300)

# Figure 4: CIA
ggarrange(LMM_CIA, AfSIS_pH_Alox_Feox_Caex_CIA,
          labels = list("a)", "b)"), nrow = 1)

ggsave("AfSIS_RefData2_CIA_fig.jpeg", height = 5, width = 9, dpi = 300)
```

7 Regression tree analysis

This chapter contains all the code for the regression tree analyses, including the pruning, the spatial cross-validation and plotting of the trees. The modelling has been done for both depth intervals seperately.

7.1 Modelling

Two seperate trees will be build (topsoil vs subsoil). Otherwise, depth will be used most of the time to build the trees.

Topsoil:

Filter the dataset:

```
AfSIS_TOP <- AfSIS_RefData_noNA %>%
  dplyr::filter(Depth == "Topsoil") %>%
  # rename Aridity Index
  dplyr::rename(PET.MAP = AridityIndex)
```

Build regression tree:

```
RT_TOP <- rpart::rpart(formula = CORG ~ MAT + PET.MAP + pH + Alox + CIA + Caex +
                         Feox + Clay_8um + LandCover, data = AfSIS_TOP, method = "anova")
```

Prune model with grid search:
Code source: https://uc-r.github.io/regression_trees

```
set.seed(42)
hyper_grid_TOP <- expand.grid(
  # minsplit = number of splits from RT_TOP +/- 10; min = 5
  minsplit = seq(5, 21, 1),
  # maxdepth = size of tree from RT_TOP +/- 10; min = 1
  maxdepth = seq(1, 24, 1)
)

models <- list()
for (i in 1:nrow(hyper_grid_TOP)) {
  # get minsplit, maxdepth values at row i
  minsplit <- hyper_grid_TOP$minsplit[i]
  maxdepth <- hyper_grid_TOP$maxdepth[i]
  # train a model and store in the list
  models[[i]] <- rpart(formula = CORG ~ MAT + PET.MAP + pH + Alox +
      CIA + Caex + Feox + Clay_8um + LandCover,
    data = AfSIS_TOP, method = "anova",
    control = list(minsplit = minsplit, maxdepth = maxdepth)
  )
```

```
}

# function to get optimal cp
get_cp <- function(x) {
  min <- which.min(x$cptable[, "xerror"])
  cp <- x$cptable[min, "CP"]
}

# function to get minimum error
get_min_error <- function(x) {
  min <- which.min(x$cptable[, "xerror"])
  xerror <- x$cptable[min, "xerror"]
}

prun_param_TOP <- hyper_grid_TOP %>%
  mutate(cp = purrr::map_dbl(models, get_cp),
    error = purrr::map_dbl(models, get_min_error)) %>%
  arrange(error) %>%
  top_n(-5, wt = error)

prun_param_TOP
```

```
##   minsplit maxdepth   cp     error
## 1       14       24 0.01 0.4169800
## 2       17        7 0.01 0.4206282
## 3        8       16 0.01 0.4250738
## 4        6       16 0.01 0.4299029
## 5       18        8 0.01 0.4305540
```

Pruned regression tree:

```
RT_TOP_pruned <- rpart(formula = CORG ~ MAT + PET.MAP + pH + Alox + CIA + Caex +
                         Feox + Clay_8um + LandCover, data = AfSIS_TOP,
                       method = "anova", control = rpart.control(
                         cp = prun_param_TOP[1,3],
                         minsplit = prun_param_TOP[1,1],
                         maxdepth = prun_param_TOP[1,2]))
```

Subsoil:

Filter dataset:

```
AfSIS_BOT <- AfSIS_RefData_noNA %>%
  filter(Depth == "Subsoil") %>%
  # rename Aridity Index
  rename(PET.MAP = AridityIndex)
```

Build regression tree:

```
RT_BOT <- rpart(formula = CORG ~ MAT + PET.MAP + pH + Alox + CIA + Caex +
                  Feox + Clay_8um + LandCover, data = AfSIS_BOT, method = "anova")
```

Prune model with grid search:

```
set.seed(42)

hyper_grid_BOT <- expand.grid(
  # minsplit = number of splits from RT_BOT +/- 10; min = 5
  minsplit = seq(5, 20, 1),
  # maxdepth = size of tree from RT_TOP +/- 10; min = 1
  maxdepth = seq(1, 21, 1)
)

models <- list()
for (i in 1:nrow(hyper_grid_BOT)) {
  # get minsplit, maxdepth values at row i
  minsplit <- hyper_grid_BOT$minsplit[i]
  maxdepth <- hyper_grid_BOT$maxdepth[i]
  # train a model and store in the list
  models[[i]] <- rpart(formula = CORG ~ MAT + PET.MAP + pH + Alox +
      CIA + Caex + Feox + Clay_8um + LandCover,
    data = AfSIS_BOT, method  = "anova",
    control = list(minsplit = minsplit, maxdepth = maxdepth)
  )
}

# function to get optimal cp
get_cp <- function(x) {
  min <- which.min(x$cptable[, "xerror"])
  cp <- x$cptable[min, "CP"]
}

# function to get minimum error
get_min_error <- function(x) {
  min <- which.min(x$cptable[, "xerror"])
  xerror <- x$cptable[min, "xerror"]
}

prun_param_BOT <- hyper_grid_BOT %>%
  mutate(cp = purrr::map_dbl(models, get_cp),
    error = purrr::map_dbl(models, get_min_error)) %>%
  arrange(error) %>%
  top_n(-5, wt = error)

prun_param_BOT
```

```
##   minsplit maxdepth   cp     error
## 1       12       13 0.01 0.4134267
## 2       13        7 0.01 0.4147995
## 3       10       13 0.01 0.4152356
## 4       19       10 0.01 0.4164858
## 5       15        6 0.01 0.4170071
```

Pruned regression tree:

```
RT_BOT_pruned <- rpart(formula = CORG ~ MAT + PET.MAP + pH + Alox + CIA + Caex +
                         Feox + Clay_8um + LandCover, data = AfSIS_BOT,
                       method = "anova", control = rpart.control(
                         cp = prun_param_BOT[1,3],
                         minsplit = prun_param_BOT[1,1],
                         maxdepth = prun_param_BOT[1,2]))
```

Plotting:
Figure A5 in the supplement of the manuscript.

```
jpeg("AfSIS_RefData_RT_TOP_BOT.jpeg", width = 8, height = 7,
    units = "in", res = 500)
par(mfrow = c(2,1))
rpart.plot(RT_TOP_pruned, tweak = 1.5, main = "a) Topsoil")
rpart.plot(RT_BOT_pruned, tweak = 1.5, main = "b) Subsoil")
dev.off()
```

7.2 Spatial cross-validation

Code source: https://bookdown.org/robinlovelace/geocompr/spatial-cv.html

Extract coordinates:

```
coords_TOP <- AfSIS_TOP[,c("Longitude", "Latitude")]
coords_BOT <- AfSIS_BOT[,c("Longitude", "Latitude")]
```

Select response and predictor variables:

```
data_TOP <- AfSIS_TOP %>%
  dplyr::select(PET.MAP, Alox, Feox, Caex, LandCover, pH, CIA, Clay_8um, CORG, MAT)

data_BOT <- AfSIS_BOT %>%
  dplyr::select(PET.MAP, Alox, Feox, Caex, LandCover, pH, CIA, Clay_8um, CORG, MAT)
```

Create tasks:

```
task_TOP_RT <- mlr::makeRegrTask(data = data_TOP, target = "CORG", coordinates = coords_TOP)
task_BOT_RT <- mlr::makeRegrTask(data = data_BOT, target = "CORG", coordinates = coords_BOT)
```

Create learners with tuning parameters from pruned regression tree:

```
learner_RT_TOP <- mlr::makeLearner(cl = "regr.rpart", predict.type = "response",
                                   cp = prun_param_TOP[1,3],
                                   minsplit = prun_param_TOP[1,1],
                                   maxdepth = prun_param_TOP[1,2])
learner_RT_BOT <- mlr::makeLearner(cl = "regr.rpart", predict.type = "response",
                                   cp = prun_param_BOT[1,3],
                                   minsplit = prun_param_BOT[1,1],
                                   maxdepth = prun_param_BOT[1,2])
```

Define spatial partitioning:

```
rdesc <- mlr::makeResampleDesc(method = "SpRepCV", folds = 5, reps = 100)
```

Resample: Topsoil

```
set.seed(42)
sp_cv_TOP_RT <- mlr::resample(learner_RT_TOP, task_TOP_RT, resampling = rdesc,
                              measures = mlr::rmse, extract = getLearnerModel,
                              models = TRUE)
```

Summary RMSE and relative RMSE:

```
summary(sp_cv_TOP_RT$measures.test$rmse)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.7962  1.3713  1.4734  1.4897  1.7403  3.1142
```

```
# relative RMSE
median(sp_cv_TOP_RT$measures.test$rmse)/mean(AfSIS_TOP$CORG)
```

```
## [1] 0.6482931
```

Resample: Subsoil

```
set.seed(42)
sp_cv_BOT_RT <- mlr::resample(learner_RT_BOT, task_BOT_RT, resampling = rdesc,
                              measures = mlr::rmse, extract = getLearnerModel,
                              models = TRUE)
```

Summary RMSE and relative RMSE:

```
summary(sp_cv_BOT_RT$measures.test$rmse)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.4398  0.5356  0.6715  0.9079  1.2313  2.2626
```

```
# relative RMSE
median(sp_cv_BOT_RT$measures.test$rmse)/mean(AfSIS_BOT$CORG)
```

```
## [1] 0.4759412
```

Plotting:
Figure A2 in the supplement of the manuscript.

```
plot_TOP_RT <- mlr::createSpatialResamplingPlots(task_TOP_RT, sp_cv_TOP_RT, crs = 4326,
                                                 repetitions = 2, point.size = 2.7,
                                                 axis.text.size = 22,
                                                 x.axis.breaks = seq(0,40,20),
                                                 y.axis.breaks = seq(-40,0,20))

SP_CV_TOP <- cowplot::plot_grid(plotlist = plot_TOP_RT[["Plots"]], ncol = 5, nrow = 2,
                                labels = plot_TOP_RT[["Labels"]], label_size = 14,
                                label_fontfamily = "arial", label_colour = "black")

cowplot::save_plot("AfSIS_RefData2_spatialCV_RT_TOP.jpeg",
                   SP_CV_TOP, base_width = 20, base_height = 8)
```

8 Random forest

This section contains all the code for the regression tree analyses and the partial dependence plots. Same as for the regression trees, the two depth layers were modelled sperately.

Code source: https://bookdown.org/robinlovelace/geocompr/eco.html

8.1 Modelling

Topsoil:

Filter dataset:

```
AfSIS_TOP <- AfSIS_RefData_noNA %>%
  filter(Depth == "Topsoil")
AfSIS_BOT <- AfSIS_RefData_noNA %>%
  filter(Depth == "Subsoil")
```

Extract coordinates:

```
coords_TOP <- AfSIS_TOP[,c("Longitude", "Latitude")]
coords_BOT <- AfSIS_BOT[,c("Longitude", "Latitude")]
```

Select response and predictor variables (also for the clay- and land cover-only models):

```
data_TOP <- AfSIS_TOP %>%
  dplyr::select(AridityIndex, Alox, Feox, Caex, LandCover, pH, CIA, Clay_8um, CORG, MAT)
data_TOP_Clay <- AfSIS_TOP %>%
  dplyr::select(Clay_8um, CORG)
data_TOP_LC <- AfSIS_TOP %>%
  dplyr::select(LandCover, CORG)
data_TOP_Clay_LC <- AfSIS_TOP %>%
  dplyr::select(Clay_8um, LandCover, CORG)

data_BOT <- AfSIS_BOT %>%
  dplyr::select(AridityIndex, Alox, Feox, Caex, LandCover, pH, CIA, Clay_8um, CORG, MAT)
data_BOT_Clay <- AfSIS_BOT %>%
  dplyr::select(Clay_8um, CORG)
data_BOT_LC <- AfSIS_BOT %>%
  dplyr::select(LandCover, CORG)
data_BOT_Clay_LC <- AfSIS_BOT %>%
  dplyr::select(Clay_8um, LandCover, CORG)
```

Create tasks:

```
task_TOP_RF <- mlr::makeRegrTask(data = data_TOP, target = "CORG",
                                 coordinates = coords_TOP)
task_TOP_Clay_RF <- mlr::makeRegrTask(data = data_TOP_Clay, target = "CORG",
                                      coordinates = coords_TOP)
task_TOP_LC_RF <- mlr::makeRegrTask(data = data_TOP_LC, target = "CORG",
                                    coordinates = coords_TOP)
task_TOP_Clay_LC_RF <- mlr::makeRegrTask(data = data_TOP_Clay_LC, target = "CORG",
                                         coordinates = coords_TOP)

task_BOT_RF <- mlr::makeRegrTask(data = data_BOT, target = "CORG",
                                 coordinates = coords_BOT)
task_BOT_Clay_RF <- mlr::makeRegrTask(data = data_BOT_Clay, target = "CORG",
                                      coordinates = coords_BOT)
task_BOT_LC_RF <- mlr::makeRegrTask(data = data_BOT_LC, target = "CORG",
                                    coordinates = coords_BOT)
task_BOT_Clay_LC_RF <- mlr::makeRegrTask(data = data_BOT_Clay_LC, target = "CORG",
                                         coordinates = coords_BOT)
```

Create learner (based on the ranger R package):

```
lrn_rf <- mlr::makeLearner(cl = "regr.ranger", predict.type = "response")
```

Specifiy resampling (5 spatially disjoint partitions):

```
tune_level = mlr::makeResampleDesc("SpCV", iters = 5)
```

Specify number of random search:

```
ctrl <- mlr::makeTuneControlRandom(maxit = 50L)
```

The search for the best hyperparameters will be done 50x in each of the spatial disjoint partitions. This allows to take the clustering of the samples into account.

Specify hyperparameter search:

```
ps_TOP <- ParamHelpers::makeParamSet(
  makeIntegerParam("mtry", lower = 1, upper = ncol(data_TOP) - 1),
  makeNumericParam("sample.fraction", lower = 0.2, upper = 0.9),
  makeIntegerParam("min.node.size", lower = 1, upper = 10))
ps_TOP_1 <- ParamHelpers::makeParamSet(
  makeIntegerParam("mtry", lower = 1, upper = ncol(data_TOP_Clay) - 1),
  makeNumericParam("sample.fraction", lower = 0.2, upper = 0.9),
  makeIntegerParam("min.node.size", lower = 1, upper = 10))
ps_TOP_2 <- ParamHelpers::makeParamSet(
  makeIntegerParam("mtry", lower = 1, upper = ncol(data_TOP_Clay_LC) - 1),
  makeNumericParam("sample.fraction", lower = 0.2, upper = 0.9),
  makeIntegerParam("min.node.size", lower = 1, upper = 10))

ps_BOT <- ParamHelpers::makeParamSet(
  makeIntegerParam("mtry", lower = 1, upper = ncol(data_BOT) - 1),
  makeNumericParam("sample.fraction", lower = 0.2, upper = 0.9),
  makeIntegerParam("min.node.size", lower = 1, upper = 10))

ps_BOT_1 <- ParamHelpers::makeParamSet(
  makeIntegerParam("mtry", lower = 1, upper = ncol(data_BOT_Clay) - 1),
  makeNumericParam("sample.fraction", lower = 0.2, upper = 0.9),
  makeIntegerParam("min.node.size", lower = 1, upper = 10))

ps_BOT_2 <- ParamHelpers::makeParamSet(
  makeIntegerParam("mtry", lower = 1, upper = ncol(data_BOT_Clay_LC) - 1),
  makeNumericParam("sample.fraction", lower = 0.2, upper = 0.9),
  makeIntegerParam("min.node.size", lower = 1, upper = 10))
```

Tuning: Topsoil
Full model:

```
set.seed(42)
tune_TOP <- mlr::tuneParams(learner = lrn_rf,
                            task = task_TOP_RF,
                            resampling = tune_level,
                            par.set = ps_TOP,
                            control = ctrl,
                            measures = mlr::rmse)
```

```
# Create learner with best hyperparameter combination
set.seed(42)
lrn_rf_TOP <- mlr::setHyperPars(makeLearner("regr.ranger", predict.type = "response"),
                                par.vals = tune_TOP$x)

# Train model
set.seed(42)
model_rf_TOP <- mlr::train(lrn_rf_TOP, task_TOP_RF)

# retrieve the ranger output
mlr::getLearnerModel(model_rf_TOP)
```

```
## Ranger result
##
## Call:
##  ranger::ranger(formula = NULL, dependent.variable.name = tn,      data = getTaskData(.task
##
## Type:                             Regression
## Number of trees:                  500
## Sample size:                      791
## Number of independent variables:  9
## Mtry:                             3
## Target node size:                 1
## Variable importance mode:         none
## Splitrule:                        variance
## OOB prediction error (MSE):       0.8732137
## R squared (OOB):                  0.7022975
```

Clay-only model:

```
set.seed(42)
tune_TOP_Clay <- mlr::tuneParams(learner = lrn_rf,
                                 task = task_TOP_Clay_RF,
                                 resampling = tune_level,
                                 par.set = ps_TOP_1,
                                 control = ctrl,
                                 measures = mlr::rmse)
```

```
# Create learner with best hyperparameter combination
set.seed(42)
lrn_rf_TOP_Clay <- mlr::setHyperPars(makeLearner("regr.ranger", predict.type = "response"),
                                     par.vals = tune_TOP_Clay$x)

# Train model
set.seed(42)
model_rf_TOP_Clay <- mlr::train(lrn_rf_TOP_Clay, task_TOP_Clay_RF)

# retrieve the ranger output
mlr::getLearnerModel(model_rf_TOP_Clay)
```

```
## Ranger result
##
## Call:
##  ranger::ranger(formula = NULL, dependent.variable.name = tn,      data = getTaskData(.task
##
## Type:                             Regression
## Number of trees:                  500
## Sample size:                      791
## Number of independent variables:  1
## Mtry:                             1
## Target node size:                 8
## Variable importance mode:         none
## Splitrule:                        variance
## OOB prediction error (MSE):       2.59454
## R squared (OOB):                  0.1154503
```

Land cover-only model:

```
set.seed(42)
tune_TOP_LC <- mlr::tuneParams(learner = lrn_rf,
                               task = task_TOP_LC_RF,
                               resampling = tune_level,
                               par.set = ps_TOP_1,
                               control = ctrl,
                               measures = mlr::rmse)
```

```
# Create learner with best hyperparameter combination
set.seed(42)
lrn_rf_TOP_LC <- mlr::setHyperPars(makeLearner("regr.ranger", predict.type = "response"),
                                   par.vals = tune_TOP_LC$x)

# Train model
set.seed(42)
model_rf_TOP_LC <- mlr::train(lrn_rf_TOP_LC, task_TOP_LC_RF)
```

```
# retrieve the ranger output
mlr::getLearnerModel(model_rf_TOP_LC)
```

```
## Ranger result
##
## Call:
##  ranger::ranger(formula = NULL, dependent.variable.name = tn,      data = getTaskData(.task
##
## Type:                             Regression
## Number of trees:                  500
## Sample size:                      791
## Number of independent variables:  1
## Mtry:                             1
## Target node size:                 8
## Variable importance mode:         none
## Splitrule:                        variance
## OOB prediction error (MSE):       2.626248
## R squared (OOB):                  0.10464
```

Clay + land cover model:

```
set.seed(42)
tune_TOP_Clay_LC <- mlr::tuneParams(learner = lrn_rf,
                                    task = task_TOP_Clay_LC_RF,
                                    resampling = tune_level,
                                    par.set = ps_TOP_2,
                                    control = ctrl,
                                    measures = mlr::rmse)
```

```
# Create learner with best hyperparameter combination
set.seed(42)
lrn_rf_TOP_Clay_LC <- mlr::setHyperPars(makeLearner("regr.ranger", predict.type = "response"),
                                        par.vals = tune_TOP_Clay_LC$x)

# Train model
set.seed(42)
model_rf_TOP_Clay_LC <- mlr::train(lrn_rf_TOP_Clay_LC, task_TOP_Clay_LC_RF)

# retrieve the ranger output
mlr::getLearnerModel(model_rf_TOP_Clay_LC)
```

```
## Ranger result
##
## Call:
##  ranger::ranger(formula = NULL, dependent.variable.name = tn,      data = getTaskData(.task
```

```
##
## Type:                             Regression
## Number of trees:                  500
## Sample size:                      791
## Number of independent variables:  2
## Mtry:                             1
## Target node size:                 8
## Variable importance mode:         none
## Splitrule:                        variance
## OOB prediction error (MSE):       2.275919
## R squared (OOB):                  0.224077
```

Tuning: Subsoil
Full model:

```
set.seed(42)
tune_BOT <- mlr::tuneParams(learner = lrn_rf,
                            task = task_BOT_RF,
                            resampling = tune_level,
                            par.set = ps_BOT,
                            control = ctrl,
                            measures = mlr::rmse)
```

```
# learning using the best hyperparameter combination
set.seed(42)
lrn_rf_BOT <- mlr::setHyperPars(makeLearner("regr.ranger", predict.type = "response"),
                                par.vals = tune_BOT$x)

# train model
set.seed(42)
model_rf_BOT <- train(lrn_rf_BOT, task_BOT_RF)

# retrieve the ranger output
mlr::getLearnerModel(model_rf_BOT)
```

```
## Ranger result
##
## Call:
##  ranger::ranger(formula = NULL, dependent.variable.name = tn,      data = getTaskData(.task
##
## Type:                             Regression
## Number of trees:                  500
## Sample size:                      810
## Number of independent variables:  9
## Mtry:                             4
## Target node size:                 7
## Variable importance mode:         none
```

```
## Splitrule:                          variance
## OOB prediction error (MSE):         0.3520693
## R squared (OOB):                    0.7185627
```

Clay-only model:

```
set.seed(42)
tune_BOT_Clay <- mlr::tuneParams(learner = lrn_rf,
                                 task = task_BOT_Clay_RF,
                                 resampling = tune_level,
                                 par.set = ps_BOT_1,
                                 control = ctrl,
                                 measures = mlr::rmse)
```

```
# learning using the best hyperparameter combination
set.seed(42)
lrn_rf_BOT_Clay <- mlr::setHyperPars(makeLearner("regr.ranger", predict.type = "response"),
                                     par.vals = tune_BOT_Clay$x)

# train model
set.seed(42)
model_rf_BOT_Clay <- mlr::train(lrn_rf_BOT_Clay, task_BOT_Clay_RF)

# retrieve the ranger output
mlr::getLearnerModel(model_rf_BOT_Clay)
```

```
## Ranger result
##
## Call:
##  ranger::ranger(formula = NULL, dependent.variable.name = tn,      data = getTaskData(.task
##
## Type:                             Regression
## Number of trees:                  500
## Sample size:                      810
## Number of independent variables:  1
## Mtry:                             1
## Target node size:                 8
## Variable importance mode:         none
## Splitrule:                        variance
## OOB prediction error (MSE):       1.09732
## R squared (OOB):                  0.1228235
```

Land-cover-only model:

```
set.seed(42)
tune_BOT_LC <- mlr::tuneParams(learner = lrn_rf,
```

```
                                  task = task_BOT_LC_RF,
                                  resampling = tune_level,
                                  par.set = ps_BOT_1,
                                  control = ctrl,
                                  measures = mlr::rmse)
```

```
# learning using the best hyperparameter combination
set.seed(42)
lrn_rf_BOT_LC <- mlr::setHyperPars(makeLearner("regr.ranger", predict.type = "response"),
                                   par.vals = tune_BOT_LC$x)

# train model
set.seed(42)
model_rf_BOT_LC <- mlr::train(lrn_rf_BOT_LC, task_BOT_LC_RF)

# retrieve the ranger output
mlr::getLearnerModel(model_rf_BOT_LC)
```

```
## Ranger result
##
## Call:
##  ranger::ranger(formula = NULL, dependent.variable.name = tn,      data = getTaskData(.task
##
## Type:                             Regression
## Number of trees:                  500
## Sample size:                      810
## Number of independent variables:  1
## Mtry:                             1
## Target node size:                 5
## Variable importance mode:         none
## Splitrule:                        variance
## OOB prediction error (MSE):       1.055299
## R squared (OOB):                  0.1564147
```

Clay + land-cover model:

```
set.seed(42)
tune_BOT_Clay_LC <- mlr::tuneParams(learner = lrn_rf,
                                    task = task_BOT_Clay_LC_RF,
                                    resampling = tune_level,
                                    par.set = ps_BOT_2,
                                    control = ctrl,
                                    measures = mlr::rmse)
```

```
# learning using the best hyperparameter combination
set.seed(42)
lrn_rf_BOT_Clay_LC <- mlr::setHyperPars(makeLearner("regr.ranger", predict.type = "response"),
                                        par.vals = tune_BOT_Clay_LC$x)

# train model
set.seed(42)
model_rf_BOT_Clay_LC <- mlr::train(lrn_rf_BOT_Clay_LC, task_BOT_Clay_LC_RF)

# retrieve the ranger output
mlr::getLearnerModel(model_rf_BOT_Clay_LC)
```

```
## Ranger result
##
## Call:
##  ranger::ranger(formula = NULL, dependent.variable.name = tn,      data = getTaskData(.task
##
## Type:                             Regression
## Number of trees:                  500
## Sample size:                      810
## Number of independent variables:  2
## Mtry:                             1
## Target node size:                 8
## Variable importance mode:         none
## Splitrule:                        variance
## OOB prediction error (MSE):       0.9223568
## R squared (OOB):                  0.2626861
```

8.2 Partial Dependence Plots

Figure 5 in the manuscript.

```
set.seed(42)
RF_TOP <- ranger::ranger(formula = CORG ~ MAT + AridityIndex + pH + Alox +
                           CIA + Caex + Feox + Clay_8um + LandCover,
                         data = AfSIS_TOP, mtry = tune_TOP$x$mtry,
                         min.node.size = tune_TOP$x$min.node.size,
                         sample.fraction = tune_TOP$x$sample.fraction)

pdp_1_rf_TOP <- pdp::partial(RF_TOP, pred.var ="MAT")
pdp_2_rf_TOP <- pdp::partial(RF_TOP, pred.var = "AridityIndex")
pdp_3_rf_TOP <- pdp::partial(RF_TOP, pred.var = "pH")
pdp_4_rf_TOP <- pdp::partial(RF_TOP, pred.var = "Alox")
pdp_5_rf_TOP <- pdp::partial(RF_TOP, pred.var = "CIA")
pdp_6_rf_TOP <- pdp::partial(RF_TOP, pred.var = "Caex")
pdp_7_rf_TOP <- pdp::partial(RF_TOP, pred.var = "Feox")
```

```
pdp_8_rf_TOP <- pdp::partial(RF_TOP, pred.var = "Clay_8um")
pdp_9_rf_TOP <- pdp::partial(RF_TOP, pred.var = "LandCover")

set.seed(42)
RF_BOT <- ranger::ranger(formula = CORG ~ MAT + AridityIndex + pH + Alox +
                           CIA + Caex + Feox + Clay_8um + LandCover,
                         data = AfSIS_BOT, mtry = tune_BOT$x$mtry,
                         min.node.size = tune_BOT$x$min.node.size,
                         sample.fraction = tune_BOT$x$sample.fraction)

pdp_1_rf_BOT <- pdp::partial(RF_BOT, pred.var = "MAT")
pdp_2_rf_BOT <- pdp::partial(RF_BOT, pred.var = "AridityIndex")
pdp_3_rf_BOT <- pdp::partial(RF_BOT, pred.var = "pH")
pdp_4_rf_BOT <- pdp::partial(RF_BOT, pred.var = "Alox")
pdp_5_rf_BOT <- pdp::partial(RF_BOT, pred.var = "CIA")
pdp_6_rf_BOT <- pdp::partial(RF_BOT, pred.var = "Caex")
pdp_7_rf_BOT <- pdp::partial(RF_BOT, pred.var = "Feox")
pdp_8_rf_BOT <- pdp::partial(RF_BOT, pred.var = "Clay_8um")
pdp_9_rf_BOT <- pdp::partial(RF_BOT, pred.var = "LandCover")
```

Plotting:
Partial dependence plot: *Figure 5* in the manuscript.

```
# MAT
pdp_MAT_TOP_BOT <- ggplot() +
  geom_line(data = pdp_1_rf_TOP, aes(x = MAT, y = yhat), size = 1.3, color = "#00A9FF") +
  geom_segment(aes(x = 23, y = 2.87, xend = 23, yend = 2.27),
               arrow = arrow(length = unit(0.4, "cm")), color = "#00A9FF") +
  geom_segment(aes(x = 22, y = 2.97, xend = 22, yend = 2.37),
               arrow = arrow(length = unit(0.3, "cm")), color = "#00A9FF") +
  geom_line(data = pdp_1_rf_BOT, aes(x = MAT, y = yhat), size = 1.3, color = "#00A9FF",
            linetype = "twodash") +
  geom_rug(data = AfSIS_TOP, aes(x = MAT), color = "#00A9FF",
           length = unit(0.05, "npc")) +
  geom_rug(data = AfSIS_BOT, aes(x = MAT), color = "#00A9FF",
           linetype = "twodash", length = unit(0.05, "npc")) +
  own_theme +
  scale_x_continuous("", expand = c(0,0), breaks = seq(15,30,5)) +
  scale_y_continuous("", expand = c(0,0)) +
  coord_cartesian(xlim = c(13.7,30), ylim = c(0,4)) +
  annotate(geom = "text", x = 20, y = 3.5,
           label = "MAT [°C]", size = 4.5)

# AridityIndex
pdp_AriInd_TOP_BOT <- ggplot() +
  geom_line(data = pdp_2_rf_TOP, aes(x = AridityIndex, y = yhat),
            size = 1.3, color = "#C77CFF") +
```

```
  geom_segment(aes(x = 1.5, y = 3.05, xend = 1.5, yend = 2.45),
               arrow = arrow(length = unit(0.4, "cm")), color = "#C77CFF") +
  geom_segment(aes(x = 1.1, y = 3.36, xend = 1.1, yend = 2.76),
               arrow = arrow(length = unit(0.4, "cm")), color = "#C77CFF") +
  geom_line(data = pdp_2_rf_BOT, aes(x = AridityIndex, y = yhat),
            size = 1.3, color = "#C77CFF", linetype = "twodash") +
  geom_segment(aes(x = 1.5, y = 0.75, xend = 1.5, yend = 1.35),
               arrow = arrow(length = unit(0.4, "cm")), color = "#C77CFF") +
  geom_rug(data = AfSIS_TOP, aes(x = AridityIndex), color = "#C77CFF",
           length = unit(0.05, "npc")) +
  geom_rug(data = AfSIS_BOT, aes(x = AridityIndex), color = "#C77CFF",
           linetype = "twodash", length = unit(0.05, "npc")) +
  own_theme +
  scale_x_continuous("", expand = c(0,0), breaks = seq(1,9,2)) +
  scale_y_continuous("", expand = c(0,0)) +
  coord_cartesian(xlim = c(0.7,10), ylim = c(0,4)) +
  annotate(geom = "text", x = 4.5, y = 3.5,
           label = "PET/MAP", size = 4.5)

# pH
pdp_pH_TOP_BOT <- ggplot() +
  geom_line(data = pdp_3_rf_TOP, aes(x = pH, y = yhat),
            size = 1.3, color = "#FF61CC") +
  geom_segment(aes(x = 5.6, y = 2.9, xend = 5.6, yend = 2.3),
               arrow = arrow(length = unit(0.4, "cm")), color = "#FF61CC") +
  geom_line(data = pdp_3_rf_BOT, aes(x = pH, y = yhat),
            size = 1.3, color = "#FF61CC", linetype = "twodash") +
  geom_segment(aes(x = 5.5, y = 0.85, xend = 5.5, yend = 1.45),
               arrow = arrow(length = unit(0.4, "cm")), color = "#FF61CC") +
  geom_rug(data = AfSIS_TOP, aes(x = pH), color = "#FF61CC",
           length = unit(0.05, "npc")) +
  geom_rug(data = AfSIS_BOT, aes(x = pH), color = "#FF61CC",
           linetype = "twodash", length = unit(0.05, "npc")) +
  own_theme +
  scale_x_continuous("", expand = c(0,0), breaks = seq(3,10,1)) +
  scale_y_continuous("", expand = c(0,0)) +
  coord_cartesian(xlim = c(3.8,10), ylim = c(0,4)) +
  annotate(geom = "text", x = 7, y = 3.5,
           label = expression("pH"[H20]), size = 4.5)

# Al_ox
pdp_Alox_TOP_BOT <- ggplot() +
  geom_line(data = pdp_4_rf_TOP, aes(x = Alox, y = yhat),
            size = 1.3, color = "#F8766D") +
  geom_segment(aes(x = 0.27, y = 2.9, xend = 0.27, yend = 2.3),
               arrow = arrow(length = unit(0.4, "cm")), color = "#F8766D") +
  geom_line(data = pdp_4_rf_BOT, aes(x = Alox, y = yhat),
```

```
            size = 1.3, color = "#F8766D", linetype = "twodash") +
  geom_segment(aes(x = 0.31, y = 0.86, xend = 0.31, yend = 1.46),
               arrow = arrow(length = unit(0.4, "cm")), color = "#F8766D") +
  geom_segment(aes(x = 0.57, y = 1.35, xend = 0.57, yend = 1.95),
               arrow = arrow(length = unit(0.4, "cm")), color = "#F8766D") +
  geom_rug(data = AfSIS_TOP, aes(x = Alox), color = "#F8766D",
           length = unit(0.05, "npc")) +
  geom_rug(data = AfSIS_BOT, aes(x = Alox), color = "#F8766D",
           linetype = "twodash", length = unit(0.05, "npc")) +
  own_theme +
  scale_x_continuous("", expand = c(0,0), breaks = seq(0,4,1)) +
  scale_y_continuous("", expand = c(0,0)) +
  coord_cartesian(xlim = c(0,4), ylim = c(0,4)) +
  annotate(geom = "text", x = 2, y = 3.5,
           label = expression("Al"[ox] ~"[wt-%]"), size = 4.5)

# CIA
pdp_CIA_TOP_BOT <- ggplot() +
  geom_line(data = pdp_5_rf_TOP, aes(x = CIA, y = yhat),
            size = 1.3, color = "#7CAE00") +
  geom_segment(aes(x = 75, y = 2.95, xend = 75, yend = 2.35),
               arrow = arrow(length = unit(0.4, "cm")), color = "#7CAE00") +
  geom_line(data = pdp_5_rf_BOT, aes(x = CIA, y = yhat),
            size = 1.3, color = "#7CAE00", linetype = "twodash") +
  geom_segment(aes(x = 99, y = 0.85, xend = 99, yend = 1.45),
               arrow = arrow(length = unit(0.4, "cm")), color = "#7CAE00") +
  geom_rug(data = AfSIS_TOP, aes(x = CIA), color = "#7CAE00",
           length = unit(0.05, "npc")) +
  geom_rug(data = AfSIS_BOT, aes(x = CIA), color = "#7CAE00",
           linetype = "twodash", length = unit(0.05, "npc")) +
  own_theme +
  scale_x_continuous("", expand = c(0,0), breaks = seq(0,101,20)) +
  scale_y_continuous("", expand = c(0,0)) +
  coord_cartesian(xlim = c(10,101), ylim = c(0,4)) +
  annotate(geom = "text", x = 54, y = 3.5,
           label = "CIA [%]", size = 4.5)

# Caex
pdp_Caex_TOP_BOT <- ggplot() +
  geom_line(data = pdp_6_rf_TOP, aes(x = Caex, y = yhat),
            size = 1.3, color = "#CD9600") +
  geom_segment(aes(x = 9.3, y = 2.85, xend = 9.3, yend = 2.25),
               arrow = arrow(length = unit(0.4, "cm")), color = "#CD9600") +
  geom_segment(aes(x = 12, y = 3.08, xend = 12, yend = 2.48),
               arrow = arrow(length = unit(0.4, "cm")), color = "#CD9600") +
  geom_segment(aes(x = 20, y = 3.34, xend = 20, yend = 2.74),
               arrow = arrow(length = unit(0.4, "cm")), color = "#CD9600") +
```

```
  geom_segment(aes(x = 21, y = 3.4, xend = 21, yend = 2.8),
               arrow = arrow(length = unit(0.4, "cm")), color = "#CD9600") +
  geom_line(data = pdp_6_rf_BOT, aes(x = Caex, y = yhat),
            size = 1.3, color = "#CD9600", linetype = "twodash") +
  geom_segment(aes(x = 7.8, y = 0.85, xend = 7.8, yend = 1.45),
               arrow = arrow(length = unit(0.4, "cm")), color = "#CD9600") +
  geom_segment(aes(x = 21, y = 1.27, xend = 21, yend = 1.87),
               arrow = arrow(length = unit(0.4, "cm")), color = "#CD9600") +
  geom_rug(data = AfSIS_TOP, aes(x = Caex), color = "#CD9600",
           length = unit(0.05, "npc")) +
  geom_rug(data = AfSIS_BOT, aes(x = Caex), color = "#CD9600",
           linetype = "twodash", length = unit(0.05, "npc")) +
  own_theme +
  scale_x_continuous("", expand = c(0,0), breaks = seq(0,75,15)) +
  scale_y_continuous("", expand = c(0,0)) +
  coord_cartesian(xlim = c(0,76), ylim = c(0,4)) +
  annotate(geom = "text", x = 23, y = 3.6,
           label = expression("Ca"[ex] ~"[cmol/kg]"), size = 4.5)

# Feox
pdp_Feox_TOP_BOT <- ggplot() +
  geom_line(data = pdp_7_rf_TOP, aes(x = Feox, y = yhat),
            size = 1.3, color = "#00BFC4") +
  geom_segment(aes(x = 0.22, y = 3.02, xend = 0.22, yend = 2.42),
               arrow = arrow(length = unit(0.4, "cm")), color = "#00BFC4") +
  geom_line(data = pdp_7_rf_BOT, aes(x = Feox, y = yhat),
            size = 1.3, color = "#00BFC4", linetype = "twodash") +
  geom_segment(aes(x = 0.15, y = 0.7, xend = 0.15, yend = 1.3),
               arrow = arrow(length = unit(0.4, "cm")), color = "#00BFC4") +
  geom_segment(aes(x = 0.12, y = 0.55, xend = 0.12, yend = 1.15),
               arrow = arrow(length = unit(0.4, "cm")), color = "#00BFC4") +
  geom_segment(aes(x = 2.1, y = 1.05, xend = 2.1, yend = 1.65),
               arrow = arrow(length = unit(0.4, "cm")), color = "#00BFC4") +
  geom_rug(data = AfSIS_TOP, aes(x = Feox), color = "#00BFC4",
           length = unit(0.05, "npc")) +
  geom_rug(data = AfSIS_BOT, aes(x = Feox), color = "#00BFC4",
           linetype = "twodash", length = unit(0.05, "npc")) +
  own_theme +
  scale_x_continuous("", expand = c(0,0), breaks = seq(0,4.5,1)) +
  scale_y_continuous("", expand = c(0,0)) +
  coord_cartesian(xlim = c(0,4.5), ylim = c(0,4)) +
  annotate(geom = "text", x = 2, y = 3.5,
           label = expression("Fe"[ox] ~"[wt-%]"), size = 4.5)

# Clay
pdp_Clay_TOP_BOT <- ggplot() +
  geom_line(data = pdp_8_rf_TOP, aes(x = Clay_8um, y = yhat),
```

```
            size = 1.3, color = "#00BE67") +
  geom_line(data = pdp_8_rf_BOT, aes(x = Clay_8um, y = yhat),
            size = 1.3, color = "#00BE67", linetype = "twodash") +
  geom_rug(data = AfSIS_TOP, aes(x = Clay_8um), color = "#00BE67",
           length = unit(0.05, "npc")) +
  geom_rug(data = AfSIS_BOT, aes(x = Clay_8um), color = "#00BE67",
           linetype = "twodash", length = unit(0.05, "npc")) +
  own_theme +
  scale_x_continuous("", expand = c(0,0), breaks = seq(0,101,20)) +
  scale_y_continuous("", expand = c(0,0)) +
  coord_cartesian(xlim = c(0,101), ylim = c(0,4)) +
  annotate(geom = "text", x = 50, y = 3.5,
           label = "Clay and fine silt [%]", size = 4.5)

# LandCover
pdp_9_rf_TOP <- pdp_9_rf_TOP %>%
  tibble::add_column(Depth = "Topsoil")
pdp_9_rf_BOT <- pdp_9_rf_BOT %>%
  tibble::add_column(Depth = "Subsoil")

pdp_9_rf <- pdp_9_rf_TOP %>%
  full_join(pdp_9_rf_BOT)

pdp_9_rf$Depth <- factor(pdp_9_rf$Depth, levels = c("Topsoil", "Subsoil",
                                                    ordered = TRUE))

pdp_LandCover <- pdp_9_rf %>%
  ggplot(aes(x = LandCover, y = yhat, group = Depth)) +
  geom_line(aes(linetype = Depth), size = 1.3, color = "#737373") +
  geom_point(size = 3, fill = "white", pch = 21) +
  geom_segment(aes(x = 4, y = 0.78, xend = 4, yend = 1.38),
               arrow = arrow(length = unit(0.4, "cm")), color = "#737373") +
  own_theme +
  theme(legend.background = element_blank()) +
  scale_x_discrete("", expand = c(0.05,0.05)) +
  scale_y_continuous("", expand = c(0,0)) +
  coord_cartesian(ylim = c(0,4)) +
  scale_linetype_manual(values = c("solid", "twodash"), labels = ) +
  annotate(geom = "text", x = 2.5, y = 3.5,
           label = "Land cover", size = 4.5)

# Arrange all plots together
PDP_arranged <- ggarrange(pdp_Alox_TOP_BOT, pdp_Caex_TOP_BOT, pdp_CIA_TOP_BOT,
                          pdp_Clay_TOP_BOT, pdp_Feox_TOP_BOT, pdp_MAT_TOP_BOT,
                          pdp_AriInd_TOP_BOT, pdp_pH_TOP_BOT, pdp_LandCover,
                          common.legend = TRUE, labels = c("a)", "b)", "c)", "d)",
                                                           "e)", "f)", "g)", "h)",
```

```
                                                        "i)"))

annotate_figure(PDP_arranged,
                left = text_grob("Predicted SOC content [wt-%]", color = "black",
                                 rot = 90, size = 14, vjust = 0.7))

#Save
ggsave("AfSIS_RefData_PDP_TOP_BOT_RF.jpeg", width = 9, height = 7, dpi = 450)
```