Supplement of SOIL, 7, 305–332, 2021
https://doi.org/10.5194/soil-7-305-2021-supplement

Supplement of

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

Sophie F. von Fromm et al.

Correspondence to: Sophie F. von Fromm (sfromm@bgc-jena.mpg.de)

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

Sophie F. von Fromm

03/31/2021

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. (2021): *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)
library(gt)
library(MuMIn)
library(ggforce)
```

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 (2021).

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

```
##  [1] "SSN"          "Longitude"    "Latitude"     "Region"       "Country"
##  [6] "Site"         "Cluster"      "Profile"      "Depth"        "Clay_2um"
## [11] "Clay_8um"     "Clay_20um"    "Total_C"      "CORG"         "CINORG"
## [16] "Total_N"      "pH"           "Am_Ox_Al"     "Am_Ox_Fe"     "Am_Ox_Mn"
## [21] "Am_Ox_P"      "Olsen_P"      "pbi"          "Caex"         "Kex"
## [26] "Mgex"         "Naex"         "eCEC"         "Al"           "Ca"
## [31] "Co"           "Cr"           "Cu"           "Fe"           "K"
## [36] "Mg"           "Mn"           "Na"           "Ni"           "P"
## [41] "Pb"           "S"            "Zn"           "Total_PSD"    "VegStructure"
## [46] "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_2um – All particles < 8 µm [%]
Clay_8um – All particles < 8 µm [%]
Clay_20um – 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_20um	2002	363
Clay_2um	2002	363
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
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. (2021).

```
AfSIS_RefData_allSites_final <- AfSIS_RefData_allSites_GlobalData %>%
  dplyr::select(-10, -c(20:23), -c(25:28), -c(31:34), -c(36:37), -c(39:43)) %>%
  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(0,0), limits = c(0,10)) +
  scale_x_continuous("uncorrected CIA [%]", expand = c(0,0),
                     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("",expand = c(0,0), limits = c(0,10)) +
```

```
  scale_x_continuous("corrected CIA [%]", expand = c(0,0),
                     limits = c(0,100))

# Plot both together
ggpubr::ggarrange(CIA_uncor, CIA)

#ggsave("AfSIS_allData_CIA_correction.pdf", height = 4, width = 8, dpi = 300)
```

The corrected CIA (see Figure A2 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",
  )) %>%
  mutate(LandCover_ESA_cor = case_when(
    LandCover_ESA == "Bare" ~ "Other",
    LandCover_ESA == "Flooded" ~ "Other",
    LandCover_ESA == "Open water" ~ "Other",
    LandCover_ESA == "Sparse vegetation" ~ "Other",
    TRUE ~ LandCover_ESA
  )) %>%
  dplyr::select(-c(VegStructure, -LandCover_ESA, PlotCultMgd, Notes))
```

3.2 Filtering

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

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_20um	1601	0	0
Clay_8um	1601	0	0
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 acidic"
AfSIS_RefData_noNA$pH_Class[AfSIS_RefData_noNA$pH_Class == 2] <- "moderately acidic"
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(-12, -c(14:15), -29, -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*).

```r
# 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)

# 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.png",
                dpi = 400, 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 2* in the manuscript. In addition, it also contains the code for all the plots in the manuscript that show raw data (*Figure 2*, *Figure 5*, *Figure A1* and *Figure A4*).

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_SumTable <- AfSIS_RefData_noNA %>%
  dplyr::select_if(is.numeric) %>%
  dplyr::select(-Longitude, -Latitude, -Clay_20um, -Al, -Ca,
                -K, -Na, -Total_PSD, -Ca_IC, -n_wet_month) %>%
  pivot_longer(everything(), names_to = "Variable", values_to = "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)) %>%
  slice(6,9,8,10,2,5,11,4,1,7,3)

AfSIS_RefData_noNA_SumTable %>%
  knitr::kable()
```

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
MAT	21.67	3.24	13.74	19.80	21.52	22.95	29.82	0.17	-0.12
MAP	1070.35	487.21	255.00	648.00	1057.00	1432.00	2708.00	0.29	-0.63
PET	1810.33	310.46	1350.00	1571.00	1759.00	1933.00	2949.00	1.19	1.96
AridityIndex	2.35	1.73	0.71	1.20	1.54	3.16	9.54	1.46	1.31
Clay_8um	55.42	22.64	0.12	37.70	57.92	74.73	100.00	-0.26	-1.00
pH	6.28	1.31	3.89	5.22	6.10	7.52	9.92	0.27	-1.11
CIA	87.74	9.30	10.34	81.70	88.09	95.98	99.91	-1.04	3.88
Alox	0.28	0.36	0.01	0.12	0.20	0.29	3.71	4.52	25.29
Feox	0.38	0.56	0.01	0.10	0.21	0.40	4.46	3.60	14.96
Caex	10.29	11.01	0.03	1.34	5.86	16.49	75.66	1.28	1.32

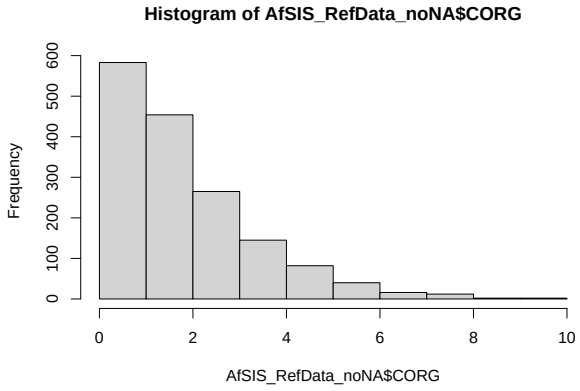
```
#Table as shown in the manuscript
AfSIS_RefData_noNA_SumTable_gt <-
  AfSIS_RefData_noNA_SumTable %>%
  gt()

# gtsave(AfSIS_RefData_noNA_SumTable_gt,
#        filename = "AfSIS_RefData_noNA_SumTable.rtf")
```

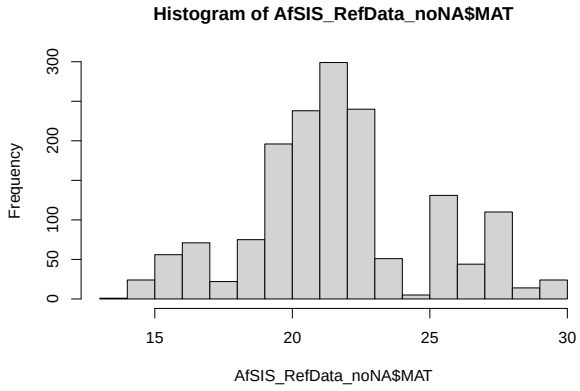
Shapiro-Wilk-Test:
Normal distribution: p > 0.05

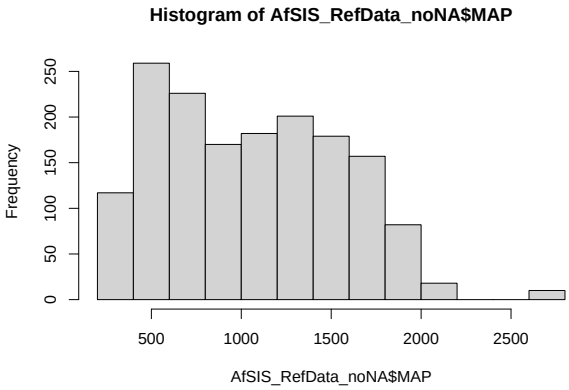
```
# Check if any p-value of the shapiro-wilk test is larger 0.5
AfSIS_RefData_noNA %>%
  dplyr::select_if(is.numeric) %>%
  purrr::map_dfr(~shapiro.test(.x)$p.value > 0.5) %>%
  pivot_longer(everything(), names_to = "Variable", values_to = "NormalDistributed") %>%
  knitr::kable()
```

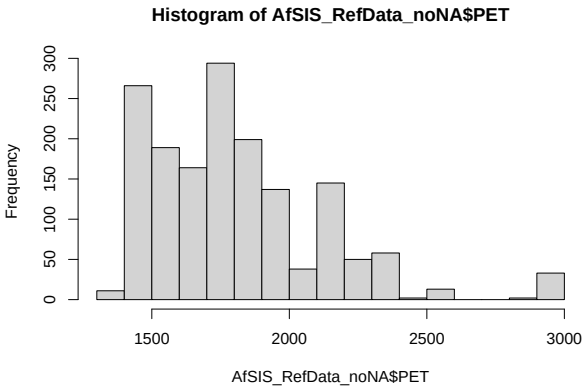
Variable	NormalDistributed
Longitude	FALSE
Latitude	FALSE
Clay_8um	FALSE
Clay_20um	FALSE
CORG	FALSE
pH	FALSE
Alox	FALSE
Feox	FALSE
Caex	FALSE
Al	FALSE
Ca	FALSE
K	FALSE
Na	FALSE
Total_PSD	FALSE
MAT	FALSE
MAP	FALSE
PET	FALSE
AridityIndex	FALSE
Ca_IC	FALSE
CIA	FALSE
n_wet_month	FALSE

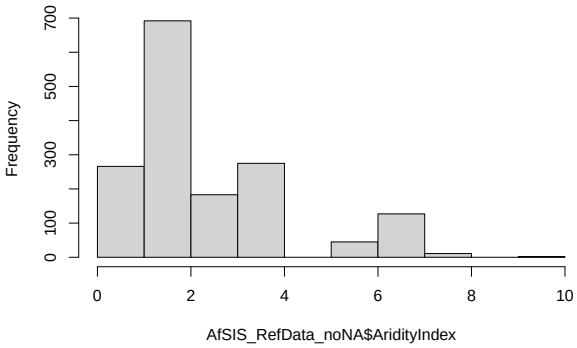
Histograms:

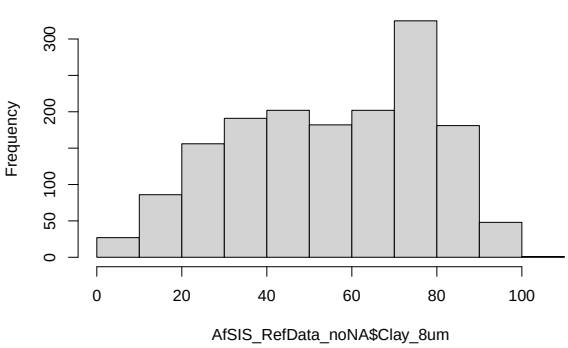


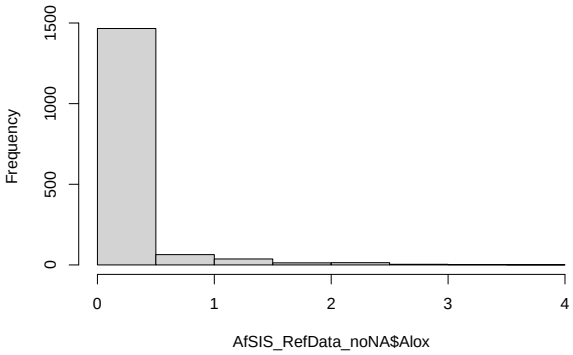


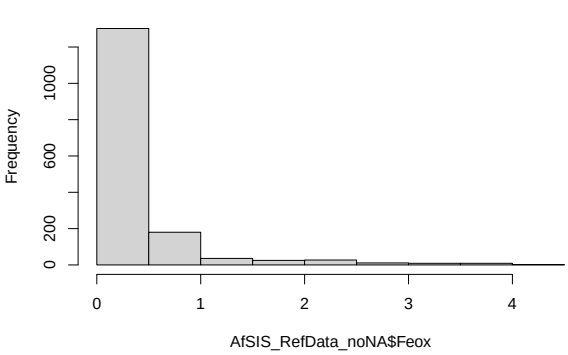

Histogram of AfSIS_RefData_noNA$MAP
Frequency
0
50
100
150
200
250
500
1000
1500
2000
2500
AfSIS_RefData_noNA$MAP

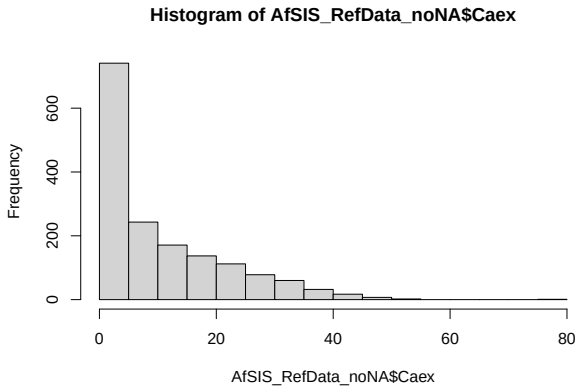

Histogram of AfSIS_RefData_noNA$PET
Frequency
0
50
100
150
200
250
300
1500
2000
2500
3000
AfSIS_RefData_noNA$PET

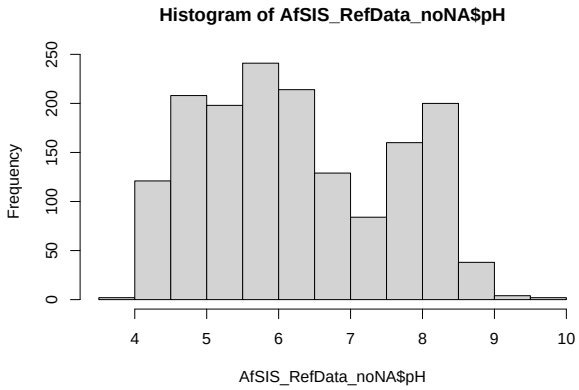

Histogram of AfSIS_RefData_noNA$AridityIndex
Frequency
0
100
300
500
700
0
2
4
6
8
10
AfSIS_RefData_noNA$AridityIndex

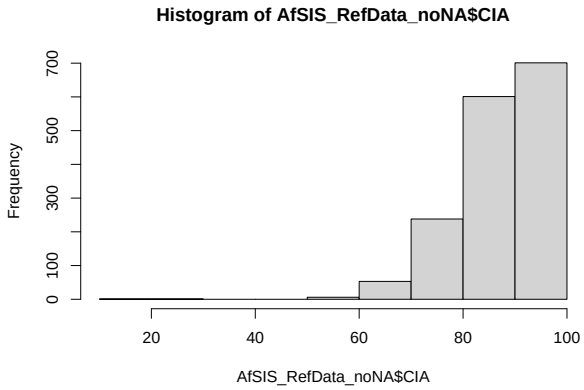

Histogram of AfSIS_RefData_noNA$Clay_8um
Frequency
0
50
100
200
300
0
20
40
60
80
100
AfSIS_RefData_noNA$Clay_8um


Histogram of AfSIS_RefData_noNA$Alox
Frequency
0
500
1000
1500
0
1
2
3
4
AfSIS_RefData_noNA$Alox


Histogram of AfSIS_RefData_noNA$Feox
Frequency
0
200
600
1000
0
1
2
3
4
AfSIS_RefData_noNA$Feox







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 and *Figure A1* in the supplement.

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

# 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 (< 8um)
```

```r
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 (< 20um)
AfSIS_Clay_CORG_20um <- AfSIS_RefData_noNA %>%
  ggplot(aes(x = Clay_20um, 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 LC and clay (< 8um) 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 + fine silt content [%]",
                                           color = "black", size = 14, vjust = -1))

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

# Plot clay (< 8um) and clay (< 20um) together
figure_arranged_clay <- ggarrange(AfSIS_Clay_CORG, AfSIS_Clay_CORG_20um,
                                  common.legend = TRUE, label.x = 0.051, label.y = 0.97,
                                  labels = c(" a) < 8 µm", "b) < 20 µm"))
annotate_figure(figure_arranged_clay,
                left = text_grob("SOC [wt-%]", color = "black", rot = 90,  size = 12),
                bottom = text_grob("Clay + silt content [%]",
                                   color = "black", size = 12, vjust = -1))

ggsave("AfSIS_RefData_Clay_SOC_Depth.jpeg", height = 4, width = 6, dpi = 300)
```

Clay content vs CORG by site:
Figure A4 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 + 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 5b 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)), fill = MAP),
             size = 4, shape = 22) +
  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)), fill = MAP),
             size = 4, shape = 21)  +
  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)), fill = MAP),
             size = 4, shape = 24) +
  own_theme +
  theme(legend.position = c(0.15,0.8)) +
  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 = 6.8, y = 22,
           label = expression(paste("Ca"[ex]," [cmol"^"+"," kg"^"-1","]")), size = 5) +
  annotate(geom = "text", x = 6.6, y = 6,
           label = expression(paste("Fe"[ox] ~ "[g kg"^"-1","]")), size = 5) +
  annotate(geom = "text", x = 5.9, y = 1,
           label = expression(paste("Al"[ox] ~ "[g kg"^"-1","]")), size = 5) +
```

```
  scale_fill_distiller(type = "seq", palette = "Blues", direction = 1,
                       name = "MAP [mm]")
```

Ca_{ex} vs CORG by pH groups:
Figure 5a 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, shape = 15,
             aes(y = CORG, x = Caex, color = pH_Class), size = 3) +
  own_theme +
  theme(legend.position = c(0.83,0.82)) +
  scale_y_continuous("SOC [wt-%]", expand = c(0,0)) +
  scale_x_continuous(expression(paste("Ca"[ex]," [cmol"^"+"," kg"^"-1","]")), expand = c(0,0)) +
  coord_cartesian(ylim = c(0,5), xlim = c(0,50)) +
  scale_color_brewer(palette = "RdBu", name = "pH classes")

ggarrange(AfSIS_Caex_SOC_pH, AfSIS_pH_Alox_Feox_Caex_MAP,
          labels = c("a)", "b)"))

ggsave("AfSIS_RefData2_pH_MAP_CaexFeoxAlox.jpeg", height = 5, width = 11, dpi = 300)
```

6 Linear mixed-effects models

This chapter contains all the code related to linear mixed-effects 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, land cover).

6.1 Entire dataset models

This section is dealing with the linear mixed-effects models for the entire dataset (n = 1,601). The results can be found in *Figure 3* and *Table B3 and B4* in the supplement of the manuscript.

6.1.1 All samples

Normalize and standardize data; merge Fe_{ox} and Al_{ox}:

```
AfSIS_RefData_noNA_normal <-
  AfSIS_RefData_noNA %>%
  mutate(Mox = 1/2 * Feox + Alox) %>%
  dplyr::select(-Longitude, -Latitude, -n_wet_month) %>%
  mutate_if(is.numeric, ~predict(bestNormalize::boxcox(.x)))
```

Build model (order of predictors based on a-priori knowledge):

```
mbx0 <- lme(CORG ~ 1,
            random = ~1|Site/Cluster/Profile, method = "ML",
            data = AfSIS_RefData_noNA_normal)
mbx1 <- update(mbx0, ~. + MAT)
mbx2 <- update(mbx1, ~. + AridityIndex)
mbx3 <- update(mbx2, ~. + Depth)
mbx4 <- update(mbx3, ~. + LandCover)
mbx5 <- update(mbx4, ~. + Clay_8um)
mbx6 <- update(mbx5, ~. + pH)
mbx7 <- update(mbx6, ~. + CIA)
mbx8 <- update(mbx7, ~. + Mox)
mbx9 <- update(mbx8, ~. + Caex)
mbx10 <- update(mbx9, ~. + pH*Mox)
```

Autocorrelation (VIF):

```
vif_full <- as.data.frame(car::vif(mbx10))
print(vif_full)
```

```
##                     GVIF Df GVIF^(1/(2*Df))
## MAT             1.022929  1        1.011399
## AridityIndex    1.127657  1        1.061912
## Depth           1.164122  1        1.078945
## LandCover       1.054299  3        1.008852
## Clay_8um        1.435020  1        1.197923
## pH              1.590136  1        1.261006
## CIA             1.244316  1        1.115489
## Mox             1.418831  1        1.191147
## Caex            1.865134  1        1.365699
## pH:Mox          1.044589  1        1.022051
```

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

VIF < 3.0: TRUE -> no autocorrelation

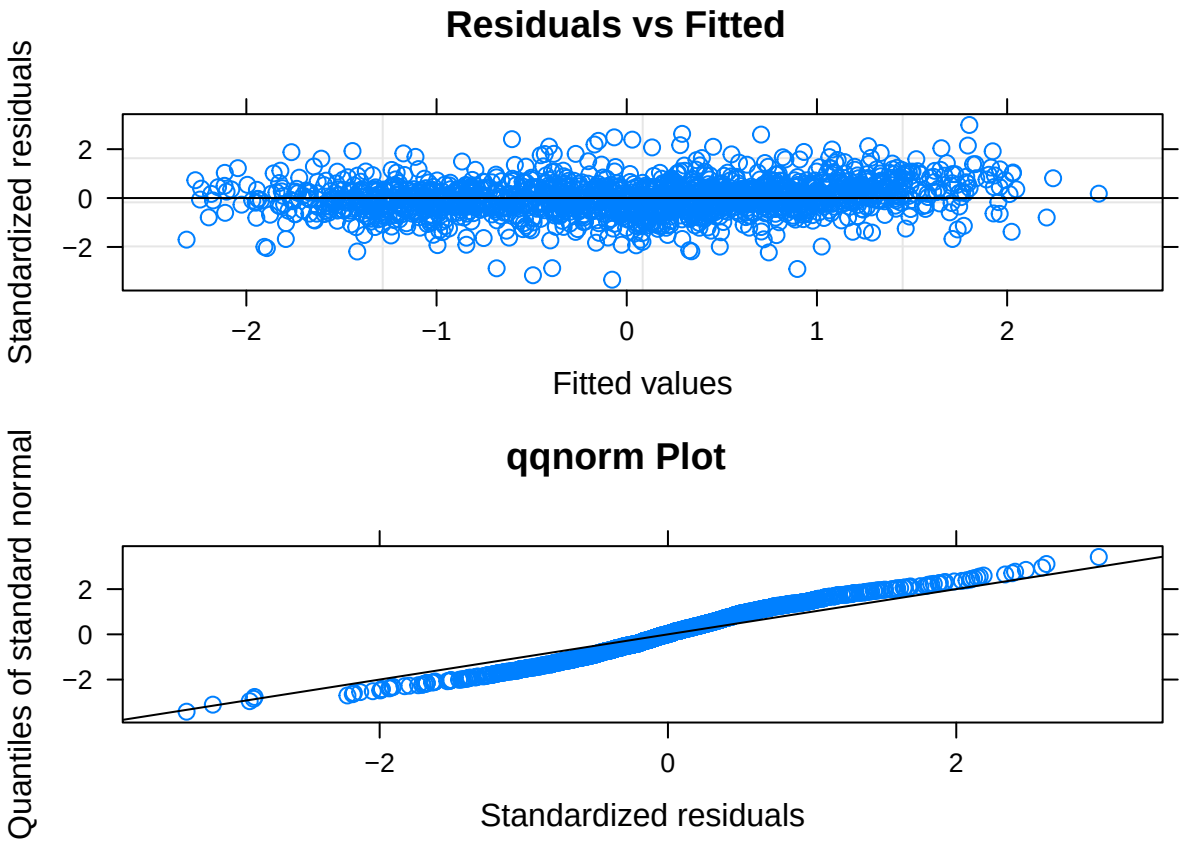
Summary output and diagnostic plots:

```
summary(mbx10)
```

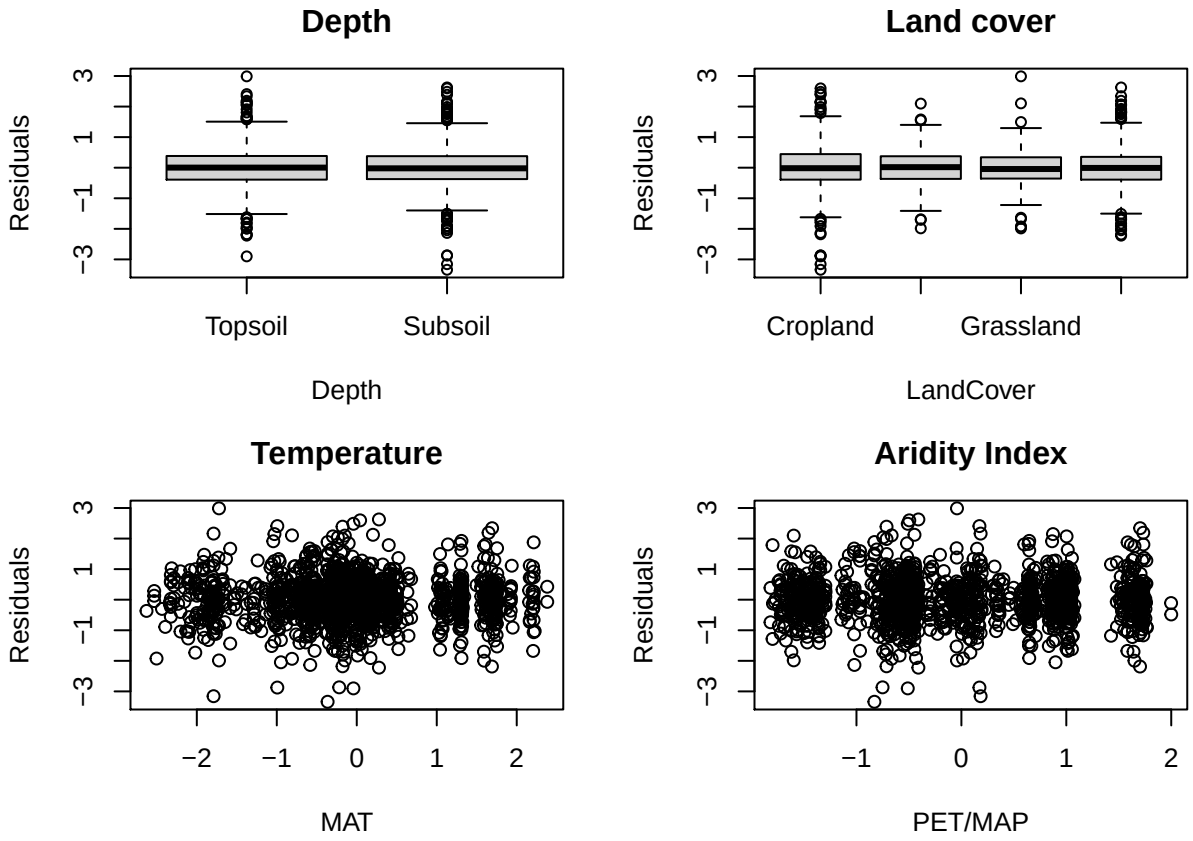
```
## Linear mixed-effects model fit by maximum likelihood
##  Data: AfSIS_RefData_noNA_normal
##        AIC      BIC    logLik
##   1599.155 1690.587 -782.5773
##
## Random effects:
##  Formula: ~1 | Site
##         (Intercept)
## StdDev:   0.3333004
##
##  Formula: ~1 | Cluster %in% Site
##         (Intercept)
## StdDev:   0.2256502
##
##  Formula: ~1 | Profile %in% Cluster %in% Site
##         (Intercept)  Residual
## StdDev:   0.2693615 0.2428774
##
## Fixed effects: CORG ~ MAT + AridityIndex + Depth + LandCover + Clay_8um + pH +      CIA + Mo
##                          Value  Std.Error  DF    t-value p-value
## (Intercept)        -0.1264070 0.06345588 783  -1.992045  0.0467
## MAT                -0.1414200 0.04566240 783  -3.097076  0.0020
## AridityIndex       -0.4273799 0.05648697 783  -7.565990  0.0000
## Depth.L            -0.3086967 0.01158359 343 -26.649481  0.0000
## LandCoverForest     0.2567720 0.08176803 783   3.140250  0.0018
## LandCoverGrassland  0.0249580 0.04219473 783   0.591496  0.5544
## LandCoverOther      0.0832766 0.03573248 783   2.330557  0.0200
## Clay_8um            0.0068470 0.01701617 343   0.402380  0.6877
## pH                 -0.3667555 0.03017725 343 -12.153378  0.0000
## CIA                -0.2174211 0.01991854 343 -10.915514  0.0000
## Mox                 0.3211101 0.02198865 343  14.603450  0.0000
## Caex                0.5262726 0.03003944 343  17.519387  0.0000
## pH:Mox             -0.1587035 0.01740751 343  -9.116952  0.0000
##  Correlation:
##                    (Intr) MAT    ArdtyI Dpth.L LndCvF LndCvG LndCvO Cly_8m
## MAT                -0.188
## AridityIndex       -0.188 -0.020
## Depth.L            -0.001  0.013 -0.045
## LandCoverForest    -0.188  0.026  0.084  0.011
## LandCoverGrassland -0.279  0.051 -0.073  0.001  0.152
## LandCoverOther     -0.338  0.038 -0.098  0.016  0.184  0.520
## Clay_8um            0.032 -0.006 -0.003 -0.254 -0.002  0.042  0.048
## pH                  0.020 -0.028 -0.139 -0.094  0.005  0.067  0.074  0.067
## CIA                -0.048 -0.026  0.065 -0.105  0.044  0.029  0.036 -0.243
## Mox                 0.028  0.086  0.161  0.022 -0.012 -0.031 -0.005 -0.263
```

```
## Caex                0.010  0.054 -0.072  0.244  0.040  0.015  0.050 -0.256
## pH:Mox              0.057 -0.025 -0.048 -0.070 -0.027 -0.021  0.011 -0.003
##                    pH     CIA    Mox    Caex
## MAT
## AridityIndex
## Depth.L
## LandCoverForest
## LandCoverGrassland
## LandCoverOther
## Clay_8um
## pH
## CIA                 0.139
## Mox                 0.199 -0.098
## Caex               -0.517  0.135 -0.328
## pH:Mox              0.175  0.018  0.019 -0.141
##
## Standardized Within-Group Residuals:
##          Min           Q1          Med           Q3          Max
## -3.339260158 -0.383272281 -0.006583472  0.379416920  2.988719988
##
## Number of Observations: 1601
## Number of Groups:
##                            Site              Cluster %in% Site
##                              45                            463
## Profile %in% Cluster %in% Site
##                            1251
```

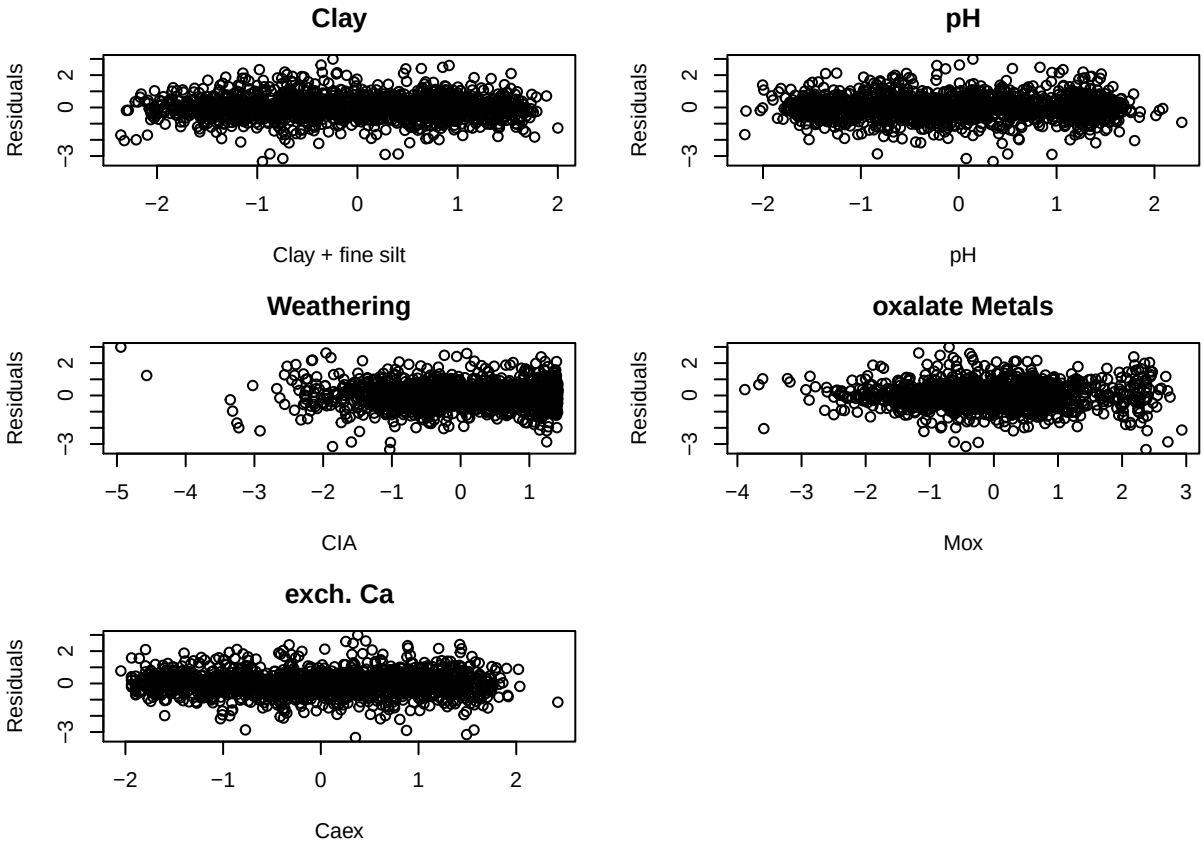
```
ggarrange(plot(mbx10, main = "Residuals vs Fitted"),
          qqnorm(mbx10, abline = c(0,1),
                 main = "qqnorm Plot"), nrow = 2)
```




```
#For each predictor
E2 <- resid(mbx10, type = "normalized")
F2 <- fitted(mbx10)
op <- par(mfrow = c(2,2), mar = c(4,4,3,2))
boxplot(E2 ~ Depth, data = AfSIS_RefData_noNA_normal,
        main = "Depth", ylab = "Residuals")
boxplot(E2 ~ LandCover, data = AfSIS_RefData_noNA_normal,
        main = "Land cover", ylab = "Residuals")
plot(x = AfSIS_RefData_noNA_normal$MAT,
     y = E2, ylab = "Residuals",
     main = "Temperature", xlab = "MAT")
plot(x = AfSIS_RefData_noNA_normal$AridityIndex,
     y = E2, ylab = "Residuals",
     main = "Aridity Index", xlab = "PET/MAP")
```




```
par(op)

op <- par(mfrow = c(3,2), mar = c(4,4,3,2))
plot(x = AfSIS_RefData_noNA_normal$Clay_8um,
     y = E2, ylab = "Residuals",
     main = "Clay", xlab = "Clay + fine silt")
plot(x = AfSIS_RefData_noNA_normal$pH,
     y = E2, ylab = "Residuals",
     main = "pH", xlab = "pH")
plot(x = AfSIS_RefData_noNA_normal$CIA,
     y = E2, ylab = "Residuals",
     main = "Weathering", xlab = "CIA")
plot(x = AfSIS_RefData_noNA_normal$Mox,
     y = E2, ylab = "Residuals",
     main = "oxalate Metals", xlab = "Mox")
plot(x = AfSIS_RefData_noNA_normal$Caex,
     y = E2, ylab = "Residuals",
     main = "exch. Ca", xlab = "Caex")
par(op)
```



Model assumptions are met.

Anova output all models (step-wise) and R^2 for full model:

```
ava_all <- anova(mbx0,mbx1,mbx2,mbx3,mbx4,mbx5,mbx6,mbx7,mbx8,mbx9,mbx10)
ava_all
```

```
##       Model df      AIC      BIC     logLik     Test  L.Ratio p-value
## mbx0      1  5 2993.221 3020.113 -1491.6107
## mbx1      2  6 2968.995 3001.266 -1478.4977   1 vs 2  26.2261  <.0001
## mbx2      3  7 2932.498 2970.147 -1459.2490   2 vs 3  38.4972  <.0001
## mbx3      4  8 2414.211 2457.238 -1199.1053   3 vs 4 520.2875  <.0001
## mbx4      5 11 2416.057 2475.219 -1197.0286   4 vs 5   4.1534  0.2454
## mbx5      6 12 2340.403 2404.943 -1158.2012   5 vs 6  77.6547  <.0001
## mbx6      7 13 2342.005 2411.923 -1158.0023   6 vs 7   0.3980  0.5281
## mbx7      8 14 2248.878 2324.176 -1110.4392   7 vs 8  95.1260  <.0001
## mbx8      9 15 1915.316 1995.992  -942.6582   8 vs 9 335.5620  <.0001
## mbx9     10 16 1678.088 1764.142  -823.0440  9 vs 10 239.2284  <.0001
## mbx10    11 17 1599.155 1690.587  -782.5773 10 vs 11  80.9334  <.0001
```

```
#Full model
r.squaredGLMM(mbx10)
```

```
##             R2m       R2c
## [1,] 0.7162272 0.9429757
```

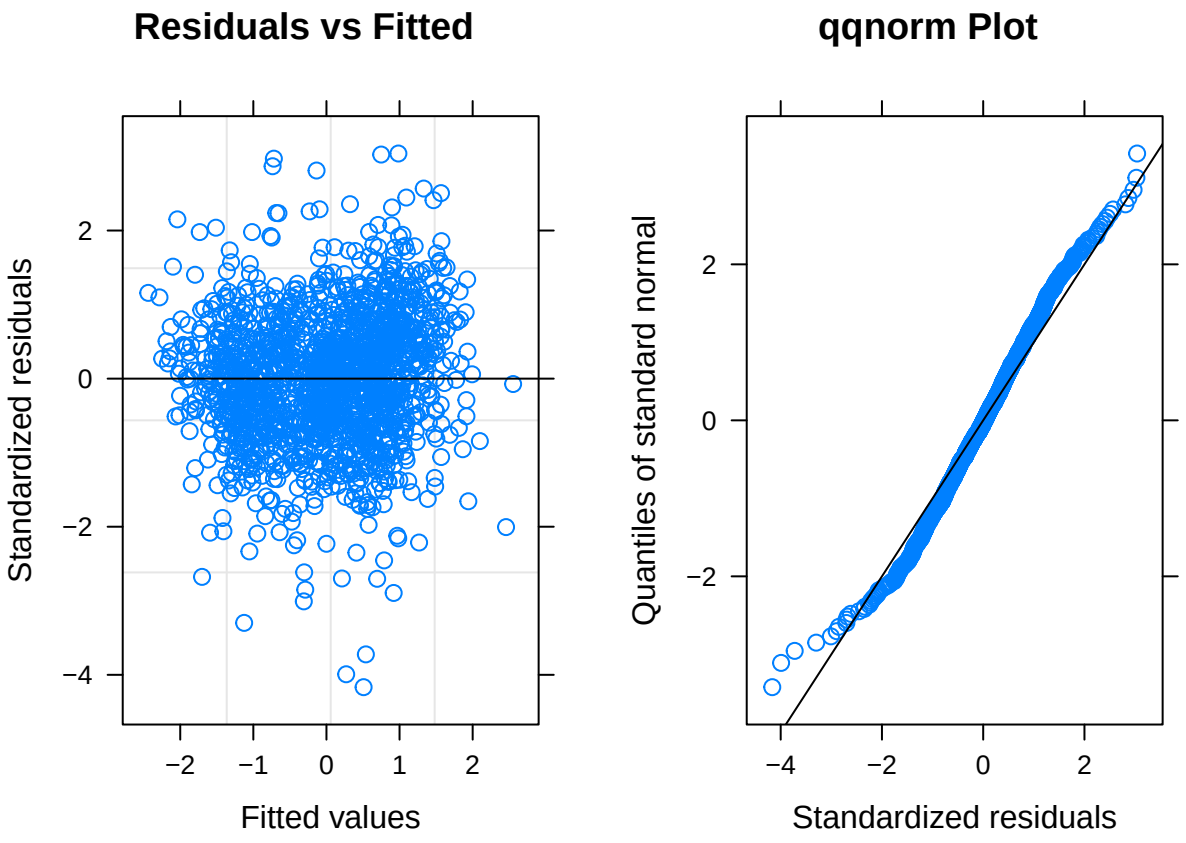
6.1.2 Geochemistry model

Build models:

```
mbx0.geo <- lme(CORG ~ 1,
                random = ~1|Site/Cluster/Profile, method = "ML",
                data = AfSIS_RefData_noNA_normal)
mbx1.geo <- update(mbx0.geo, ~. + Clay_8um)
mbx2.geo <- update(mbx1.geo, ~. + pH)
mbx3.geo <- update(mbx2.geo, ~. + CIA)
mbx4.geo <- update(mbx3.geo, ~. + Mox)
mbx5.geo <- update(mbx4.geo, ~. + Caex)
mbx6.geo <- update(mbx5.geo, ~. + pH*Mox)
```

Diagnostic plots:

```
ggarrange(plot(mbx6.geo, main = "Residuals vs Fitted"),
          qqnorm(mbx6.geo, abline = c(0,1),
                 main = "qqnorm Plot"))
```



Model assumptions are met.

Anova output for all geochemistry models (step-wise):

```
summary(mbx6.geo)
```

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: AfSIS_RefData_noNA_normal
##        AIC      BIC   logLik
##   2170.661 2229.823 -1074.33
##
## Random effects:
##  Formula: ~1 | Site
##         (Intercept)
## StdDev:    0.531758
##
##  Formula: ~1 | Cluster %in% Site
##         (Intercept)
## StdDev:   0.2350316
##
##  Formula: ~1 | Profile %in% Cluster %in% Site
##         (Intercept)  Residual
## StdDev:   0.1672964 0.3832526
##
## Fixed effects: CORG ~ Clay_8um + pH + CIA + Mox + Caex + pH:Mox
##                  Value  Std.Error  DF    t-value p-value
## (Intercept) -0.2123858 0.08503575 788  -2.497606  0.0127
## Clay_8um    -0.0916701 0.01938135 344  -4.729808  0.0000
## pH          -0.4746549 0.03513753 344 -13.508488  0.0000
## CIA         -0.2624443 0.02355414 344 -11.142171  0.0000
## Mox          0.3798990 0.02562432 344  14.825723  0.0000
## Caex         0.6553652 0.03485612 344  18.802014  0.0000
## pH:Mox      -0.1865062 0.02038018 344  -9.151351  0.0000
##  Correlation:
##          (Intr) Cly_8m pH     CIA    Mox    Caex
## Clay_8um  0.041
## pH        0.010  0.040
## CIA      -0.020 -0.291  0.143
## Mox       0.067 -0.261  0.236 -0.119
## Caex      0.021 -0.243 -0.539  0.155 -0.353
## pH:Mox    0.031 -0.031  0.179  0.025  0.021 -0.140
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -4.16659325 -0.54403254  0.02543385  0.54089760  3.04008827
##
## Number of Observations: 1601
## Number of Groups:
```

```
##                            Site              Cluster %in% Site
##                              45                           463
## Profile %in% Cluster %in% Site
##                            1251
```

```
ava_geo <- anova(mbx0.geo,mbx1.geo,mbx2.geo,mbx3.geo,mbx4.geo,mbx5.geo,mbx6.geo)
ava_geo
```

```
##          Model df      AIC      BIC    logLik   Test  L.Ratio p-value
## mbx0.geo     1  5 2993.221 3020.113 -1491.611
## mbx1.geo     2  6 2979.196 3011.466 -1483.598 1 vs 2  16.0257   1e-04
## mbx2.geo     3  7 2980.121 3017.770 -1483.061 2 vs 3   1.0743   3e-01
## mbx3.geo     4  8 2882.132 2925.159 -1433.066 3 vs 4  99.9893  <.0001
## mbx4.geo     5  9 2515.812 2564.218 -1248.906 4 vs 5 368.3196  <.0001
## mbx5.geo     6 10 2249.950 2303.733 -1114.975 5 vs 6 267.8629  <.0001
## mbx6.geo     7 11 2170.661 2229.823 -1074.330 6 vs 7  81.2889  <.0001
```

```
r.squaredGLMM(mbx6.geo)
```

```
##            R2m       R2c
## [1,] 0.4559481 0.8441893
```

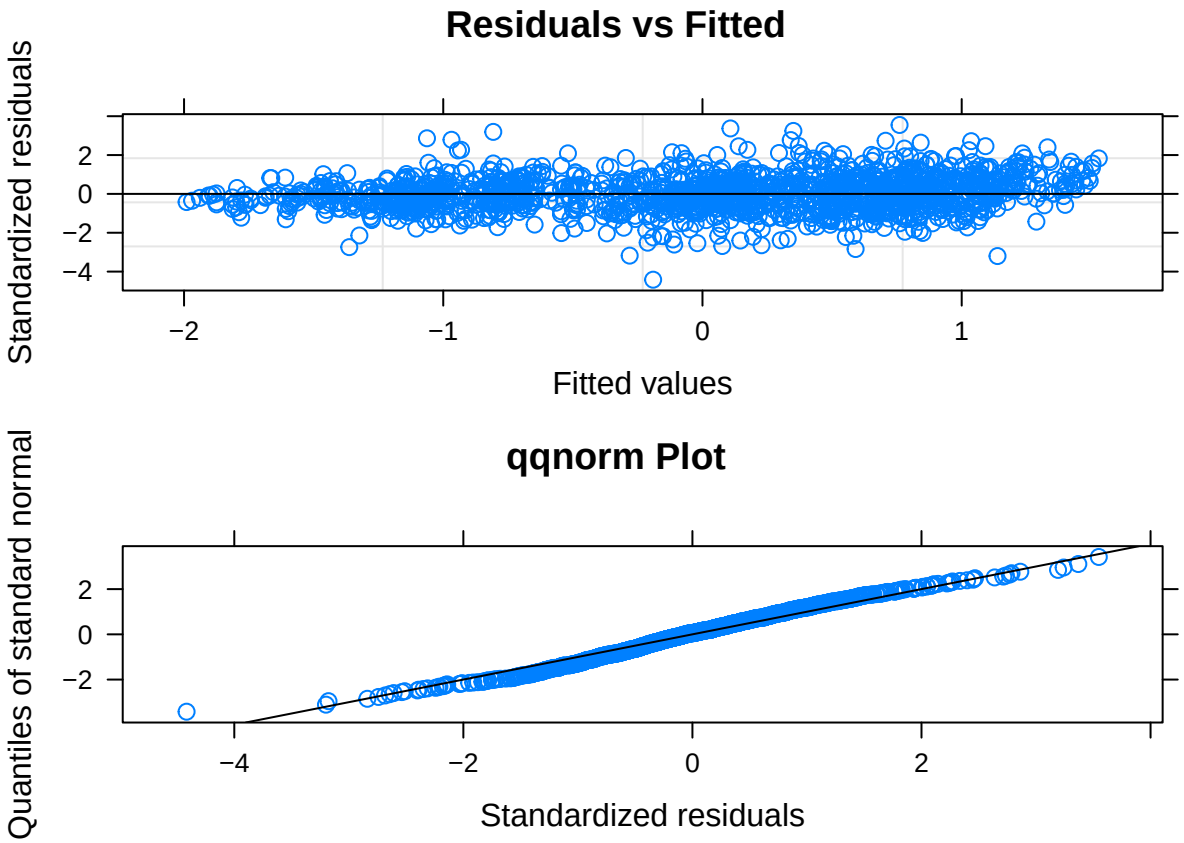
6.1.3 Climate model

Build models:

```
mbx0.clim <- lme(CORG ~ 1,
                 random = ~1|Site/Cluster/Profile, method = "ML",
                 data = AfSIS_RefData_noNA_normal)
mbx1.clim <- update(mbx0.clim, ~. + MAT)
mbx2.clim <- update(mbx1.clim, ~. + AridityIndex)
```

Diagnostic plots:

```
ggarrange(plot(mbx2.clim, main = "Residuals vs Fitted"),
          qqnorm(mbx2.clim, abline = c(0,1),
                 main = "qqnorm Plot"), nrow = 2)
```



Model assumptions are met.

Anova output for all climate models (step-wise):

```
summary(mbx2.clim)
```

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: AfSIS_RefData_noNA_normal
##        AIC      BIC    logLik
##   2932.498 2970.147 -1459.249
##
## Random effects:
##  Formula: ~1 | Site
##         (Intercept)
## StdDev:   0.5021366
##
##  Formula: ~1 | Cluster %in% Site
##         (Intercept)
## StdDev:   0.3043013
##
##  Formula: ~1 | Profile %in% Cluster %in% Site
##         (Intercept) Residual
## StdDev:   0.1930623 0.497936
```

```
##
## Fixed effects: CORG ~ MAT + AridityIndex
##                    Value  Std.Error  DF   t-value p-value
## (Intercept)  -0.1857930 0.08635738 786 -2.151443  0.0317
## MAT          -0.3633289 0.06589536 786 -5.513724  0.0000
## AridityIndex -0.5623186 0.07869964 786 -7.145124  0.0000
##  Correlation:
##              (Intr) MAT
## MAT          -0.195
## AridityIndex -0.226 -0.039
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -4.41657570 -0.53834836 -0.05422915  0.52772385  3.54806363
##
## Number of Observations: 1601
## Number of Groups:
##                            Site              Cluster %in% Site
##                              45                            463
## Profile %in% Cluster %in% Site
##                            1251
```

```
ava_clim <- anova(mbx0.clim,mbx1.clim,mbx2.clim)
ava_clim
```

```
##           Model df      AIC      BIC    logLik   Test  L.Ratio p-value
## mbx0.clim     1  5 2993.221 3020.113 -1491.611
## mbx1.clim     2  6 2968.995 3001.266 -1478.498 1 vs 2 26.22608  <.0001
## mbx2.clim     3  7 2932.498 2970.147 -1459.249 2 vs 3 38.49724  <.0001
```

```
r.squaredGLMM(mbx2.clim)
```

```
##            R2m       R2c
## [1,] 0.4758496 0.7937023
```

6.1.4 Summary tables

The summary tables can be found in *Figure 3* and *Table B4-B8*.

```
##R²
#All samples
MAT <- r.squaredGLMM(mbx1)[1,1]
AridityIndex <- r.squaredGLMM(mbx2)[1,1]-r.squaredGLMM(mbx1)[1,1]
Depth <- r.squaredGLMM(mbx3)[1,1]-r.squaredGLMM(mbx2)[1,1]
LandCover <- r.squaredGLMM(mbx4)[1,1]-r.squaredGLMM(mbx3)[1,1]
```

```
Clay <- r.squaredGLMM(mbx5)[1,1]-r.squaredGLMM(mbx4)[1,1]
pH <- r.squaredGLMM(mbx6)[1,1]-r.squaredGLMM(mbx5)[1,1]
CIA <- r.squaredGLMM(mbx7)[1,1]-r.squaredGLMM(mbx6)[1,1]
Mox <- r.squaredGLMM(mbx8)[1,1]-r.squaredGLMM(mbx7)[1,1]
Caex <- r.squaredGLMM(mbx9)[1,1]-r.squaredGLMM(mbx8)[1,1]
pH_Mox <- r.squaredGLMM(mbx10)[1,1]-r.squaredGLMM(mbx9)[1,1]

R2m.all <- tibble(MAT, AridityIndex, Depth, LandCover,
                  Clay, pH, CIA, Mox, Caex, pH_Mox) %>%
  mutate(Models = "All samples (R² = 0.72)") %>%
  dplyr::select(Models, everything())

#Geochemistry
MAT <- NA
AridityIndex <- NA
Depth <- NA
LandCover <- NA
Clay <- r.squaredGLMM(mbx1.geo)[1,1]
pH <- r.squaredGLMM(mbx2.geo)[1,1]-r.squaredGLMM(mbx1.geo)[1,1]
CIA <- r.squaredGLMM(mbx3.geo)[1,1]-r.squaredGLMM(mbx2.geo)[1,1]
Mox <- r.squaredGLMM(mbx4.geo)[1,1]-r.squaredGLMM(mbx3.geo)[1,1]
Caex <- r.squaredGLMM(mbx5.geo)[1,1]-r.squaredGLMM(mbx4.geo)[1,1]
pH_Mox <- r.squaredGLMM(mbx6.geo)[1,1]-r.squaredGLMM(mbx5.geo)[1,1]

R2m.geo <- tibble(MAT, AridityIndex, Depth, LandCover,
                  Clay, pH, CIA, Mox, Caex, pH_Mox) %>%
  mutate(Models = "Geochemistry (R² = 0.46)") %>%
  dplyr::select(Models, everything())

#Climate
MAT <- r.squaredGLMM(mbx1.clim)[1,1]
AridityIndex <- r.squaredGLMM(mbx2.clim)[1,1]-r.squaredGLMM(mbx1.clim)[1,1]
Clay <- NA
pH <- NA
CIA <- NA
Mox <- NA
Caex <- NA
pH_Mox <- NA

R2m.clim <- tibble(MAT, AridityIndex, Depth, LandCover,
                   Clay, pH, CIA, Mox, Caex, pH_Mox) %>%
  mutate(Models = "Climate (R² = 0.48)") %>%
  dplyr::select(Models, everything())

R2m.table <- base::rbind(R2m.all, R2m.geo, R2m.clim) %>%
  gt() %>%
  fmt_number(columns = c(2:11), decimals = 2) %>%
```

```
  cols_label(AridityIndex = "PET/MAP",
             LandCover = "Land cover",
             Clay = "Clay + fine silt",
             pH_Mox = "pH*Mox") %>%
  cols_align(align = "center", columns = c(2:11)) %>%
  fmt_missing(columns = c(2:11), missing_text = "-")

#gtsave(R2m.table, filename = "AfSIS_RefData2_R2m.rtf")

##Anova
ava_all_gt <- rbind(ava_all, ava_geo, ava_clim) %>%
  dplyr::select(-call) %>%
  gt(rowname_col = "Model") %>%
  tab_row_group(group = "All predictors", rows = 1:11) %>%
  tab_row_group(group = "Geochemistry-only", rows = 12:18) %>%
  tab_row_group(group = "Climate-only", rows = 19:21) %>%
  row_group_order(
    groups = c("All predictors", "Geochemistry-only", "Climate-only")) %>%
  fmt_number(columns = c(3:5,7), decimals = 2) %>%
  fmt_number(columns = 8, decimals = 4) %>%
  cols_align(align = "center", columns = c(2:8))

#gtsave(ava_all_gt, filename = "AfSIS_RefData2_AnovaAll.rtf")
```

6.1.5 Variation partitioning

```
## Without Depth and land cover
#Geo.Clim model: a+b+c
mbx10.geo.clim <- lme(CORG ~ MAT + AridityIndex + Clay_8um + pH + CIA +
                        Mox + Caex + pH*Mox, random = ~1|Site/Cluster/Profile,
                      method = "ML", data = AfSIS_RefData_noNA_normal)
r.squaredGLMM(mbx10.geo.clim)

#Geochemistry model: a+b
mbx6.geo$call
r.squaredGLMM(mbx6.geo)

#Climate model: b+c
mbx2.clim$call
r.squaredGLMM(mbx2.clim)

#Marginal R² for each model
abc <- round(r.squaredGLMM(mbx10.geo.clim)[1,1], digits = 2)
ab <- round(r.squaredGLMM(mbx6.geo)[1,1], digits = 2)
bc <- round(r.squaredGLMM(mbx2.clim)[1,1], digits = 2)
```

```
b <- ab + bc - abc
a <- ab - b
c <- bc - b

##Plotting
ggplot() +
  ggforce::geom_circle(aes(x0 = 0.565, y0 = 0.5, r = 0.048), fill = "#fe9929",
                       alpha = 0.5, n = 500, color = "#fe9929", size = 0.1) +
  geom_circle(aes(x0 = 0.6, y0 = 0.5, r = 0.05), fill = "#0570b0",
              alpha = 0.5, n = 500, color = "#0570b0", size = 0.1) +
  annotate(geom = "text", color = "black",
           label = "Geochemistry", y = 0.573, x = 0.559, size = 4) +
  annotate(geom = "text", color = "darkgrey",
           label = expression(paste("Clay + fine silt, ","pH"[H2O],",")),
           size = 3, y = 0.562, x = 0.559) +
  annotate(geom = "text", color = "darkgrey",
           label = expression(paste("CIA, ","M"[ox],", ","Ca"[ex])),
           size = 3, y = 0.555, x = 0.559) +
  annotate(geom = "text", color = "black",
           label = "19%", size = 4, y = 0.5, x = 0.535) +
  annotate(geom = "text", color = "black",
           label = "Climate", y = 0.573, x = 0.605, size = 4) +
  annotate(geom = "text", color = "darkgrey",
           label = "MAT,", size = 3, y = 0.563, x = 0.605) +
  annotate(geom = "text", color = "darkgrey",
           label = "PET/MAP", size = 3, y = 0.555, x = 0.605) +
  annotate(geom = "text", color = "black",
           label = "Geochemistry + Climate: 67%",
           size = 4, y = 0.44, x = 0.585) +
  annotate(geom = "text", color = "black",
           label = "21%", size = 4, y = 0.5, x = 0.63) +
  annotate(geom = "text", color = "black",
           label = "27%", size = 4, y = 0.5, x = 0.585) +
  theme_no_axes() +
  theme(rect = element_blank())

#ggsave("AfSIS_RefData2_VariationPartitioning_Manually.tiff",
#       width = 3.5, height = 2.5, dpi = 1200)
```

6.1.6 Clay and land cover models

Clay-only model (< 8um):

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

```
summary(LMM_Clay)$AIC
```

```
## [1] 2979.196
```

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

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.4036670	0.1129380	788	-3.574237	0.0003726
Clay_8um	0.0877243	0.0210309	349	4.171208	0.0000383

R^2 results of final model:

```
##               R2m       R2c
## [1,] 0.008495354 0.7116614
```

Clay-only model (< 20um):

```
LMM_Clay_20 <- nlme::lme(CORG ~ Clay_20um, random = ~1|Site/Cluster/Profile,
                         method = "ML", data = AfSIS_RefData_noNA_normal)

summary(LMM_Clay_20)$AIC
```

```
## [1] 2942.657
```

```
summary(LMM_Clay_20)$tTable %>% knitr::kable()
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.3641492	0.1049396	788	-3.470082	0.0005486
Clay_20um	0.1647142	0.0217156	349	7.585056	0.0000000

R^2 results of final model:

```
##              R2m       R2c
## [1,] 0.03233415 0.6836811
```

Land cover-only model:

```
LMM_LC <- nlme::lme(CORG ~ LandCover, random = ~1|Site/Cluster/Profile,
                    method = "ML", data = AfSIS_RefData_noNA_normal)
summary(LMM_LC)$AIC
```

```
## [1] 2992.522
```

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

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.5048850	0.1252783	785	-4.0301074	0.0000612
LandCoverForest	0.2855258	0.1153232	785	2.4758744	0.0135009
LandCoverGrassland	0.0428359	0.0588019	785	0.7284775	0.4665386
LandCoverOther	0.0590269	0.0499577	785	1.1815377	0.2377470

R^2 results of final model:

```
##               R2m       R2c
## [1,] 0.008188464 0.7498529
```

Clay + land cover-only model:

```
LMM_Clay_LC <- nlme::lme(CORG ~ Clay_8um + LandCover, random = ~1|Site/Cluster/Profile,
                         method = "ML", data = AfSIS_RefData_noNA_normal)
summary(LMM_Clay_LC)$AIC
```

```
## [1] 2977.265
```

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

	Value	Std.Error	DF	t-value	p-value
(Intercept)	-0.4734388	0.1177611	785	-4.0203314	0.0000637
Clay_8um	0.0911308	0.0210673	349	4.3256952	0.0000199
LandCoverForest	0.2941654	0.1133265	785	2.5957335	0.0096153
LandCoverGrassland	0.0529993	0.0579815	785	0.9140715	0.3609600
LandCoverOther	0.0758906	0.0494496	785	1.5347039	0.1252595

R^2 results of final model:

```
##              R2m       R2c
## [1,] 0.02154288 0.713932
```

6.2 Sub-models

This section contains the code for all sub-models. The results can be found in *Figure 4* and *Table B2*.

6.2.1 Depth models

This section contains the two models for the topsoil and subsoil samples, respectively.

Normalize and standardize each sub-group:

```
AfSIS_depth_normal <- AfSIS_RefData_noNA %>%
  dplyr::mutate(Mox = 1/2 * Feox + Alox) %>%
  dplyr::select(-Longitude, -Latitude, -n_wet_month) %>%
  group_by(Depth) %>%
  mutate_if(is.numeric, ~predict(bestNormalize::boxcox(.x)))
```

Topsoil:

Build models:

```
mbx0.TOP <- lme(CORG ~ 1,
                random = ~1|Site/Cluster, method = "ML",
                data = AfSIS_depth_normal %>% filter(Depth == "Topsoil"))
mbx1.TOP <- update(mbx0.TOP, ~. + Clay_8um)
mbx2.TOP <- update(mbx1.TOP, ~. + pH)
mbx3.TOP <- update(mbx2.TOP, ~. + CIA)
mbx4.TOP <- update(mbx3.TOP, ~. + Mox)
mbx5.TOP <- update(mbx4.TOP, ~. + Caex)
mbx6.TOP <- update(mbx5.TOP, ~. + pH*Mox)
```

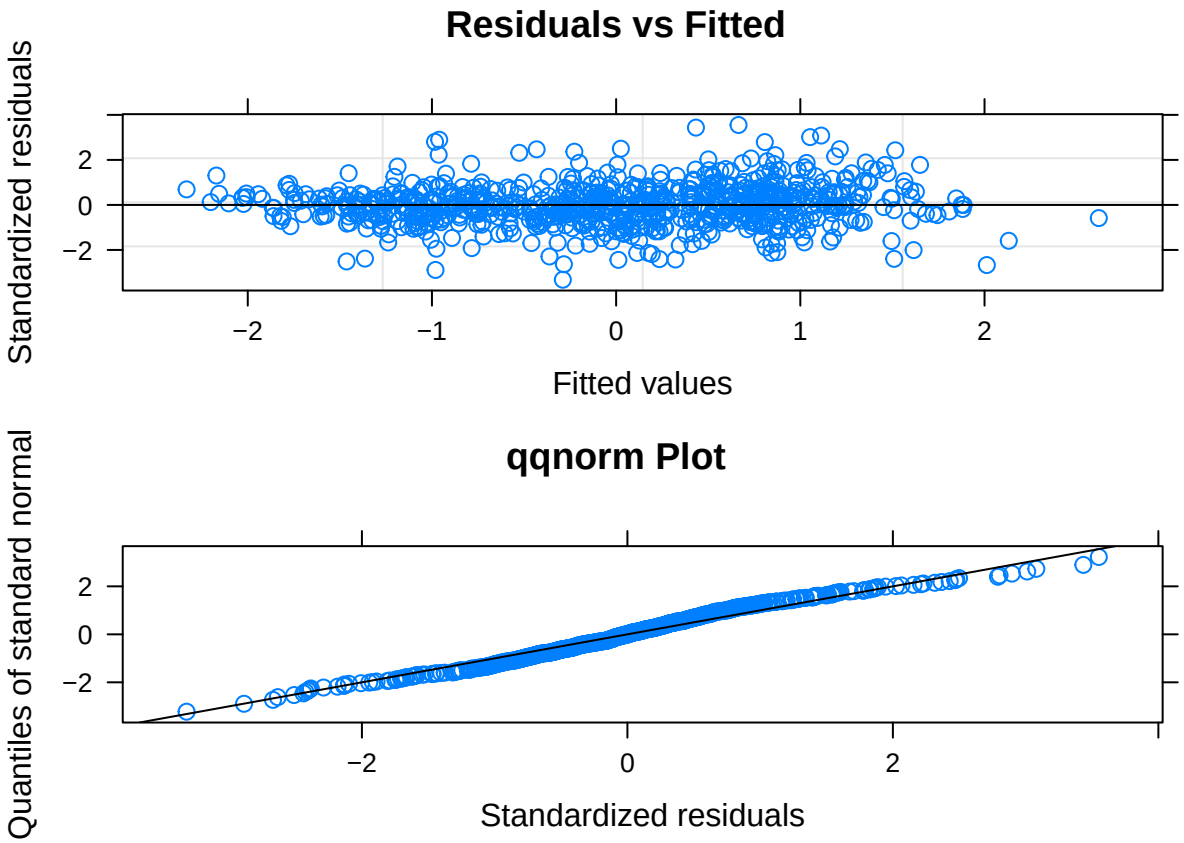
Autocorrelation:

```
vif_top <- as.data.frame(car::vif(mbx6.TOP))
vif_top_test <- max(vif_top$"car::vif(mbx6.TOP)") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Diagnostic plots:

```
ggarrange(plot(mbx6.TOP, main = "Residuals vs Fitted"),
          qqnorm(mbx6.TOP, abline = c(0,1),
                 main = "qqnorm Plot"), nrow = 2)
```



Model assumptions are met.

Anova output for all topsoil models (step-wise):

```
summary(mbx6.TOP)
```

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: AfSIS_depth_normal %>% filter(Depth == "Topsoil")
##        AIC     BIC    logLik
##   967.1072 1013.84 -473.5536
##
## Random effects:
##  Formula: ~1 | Site
##         (Intercept)
## StdDev:   0.4680077
##
##  Formula: ~1 | Cluster %in% Site
##         (Intercept)  Residual
## StdDev:   0.2141417 0.3675949
##
## Fixed effects: CORG ~ Clay_8um + pH + CIA + Mox + Caex + pH:Mox
##                  Value  Std.Error  DF    t-value p-value
## (Intercept) -0.2490844 0.07838700 425  -3.177624  0.0016
```

```
## Clay_8um    -0.0553626 0.02458803 425  -2.251608  0.0249
## pH          -0.5662012 0.04565521 425 -12.401678  0.0000
## CIA         -0.2558788 0.02913707 425  -8.781901  0.0000
## Mox          0.3679057 0.03198528 425  11.502345  0.0000
## Caex         0.5887718 0.04379886 425  13.442626  0.0000
## pH:Mox      -0.1908864 0.02622160 425  -7.279738  0.0000
##  Correlation:
##          (Intr) Cly_8m pH     CIA    Mox    Caex
## Clay_8um  0.049
## pH        0.027  0.091
## CIA      -0.028 -0.284  0.125
## Mox       0.084 -0.250  0.285 -0.102
## Caex      0.020 -0.308 -0.563  0.157 -0.361
## pH:Mox    0.038 -0.043  0.155  0.063  0.060 -0.160
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -3.32079495 -0.52349907  0.00628827  0.49126882  3.55111101
##
## Number of Observations: 791
## Number of Groups:
##              Site Cluster %in% Site
##                43               360
```

```
ava_TOP <- anova(mbx0.TOP,mbx1.TOP,mbx2.TOP,mbx3.TOP,mbx4.TOP,mbx5.TOP,mbx6.TOP)
ava_TOP
```

```
##          Model df       AIC      BIC    logLik   Test   L.Ratio p-value
## mbx0.TOP     1  4 1440.7243 1459.418 -716.3622
## mbx1.TOP     2  5 1418.8804 1442.247 -704.4402 1 vs 2  23.84389  <.0001
## mbx2.TOP     3  6 1408.7376 1436.777 -698.3688 2 vs 3  12.14285   5e-04
## mbx3.TOP     4  7 1350.4067 1383.120 -668.2033 3 vs 4  60.33092  <.0001
## mbx4.TOP     5  8 1148.8705 1186.257 -566.4352 4 vs 5 203.53620  <.0001
## mbx5.TOP     6  9 1016.1387 1058.198 -499.0694 5 vs 6 134.73174  <.0001
## mbx6.TOP     7 10  967.1072 1013.840 -473.5536 6 vs 7  51.03154  <.0001
```

```
r.squaredGLMM(mbx6.TOP)
```

```
##            R2m       R2c
## [1,] 0.5056475 0.8330061
```

Subsoil:

Build model:

```
mbx0.BOT <- lme(CORG ~ 1,
                random = ~1|Site/Cluster, method = "ML",
                data = AfSIS_depth_normal %>% filter(Depth == "Subsoil"))
mbx1.BOT <- update(mbx0.BOT, ~. + Clay_8um)
mbx2.BOT <- update(mbx1.BOT, ~. + pH)
mbx3.BOT <- update(mbx2.BOT, ~. + CIA)
mbx4.BOT <- update(mbx3.BOT, ~. + Mox)
mbx5.BOT <- update(mbx4.BOT, ~. + Caex)
mbx6.BOT <- update(mbx5.BOT, ~. + pH*Mox)
```

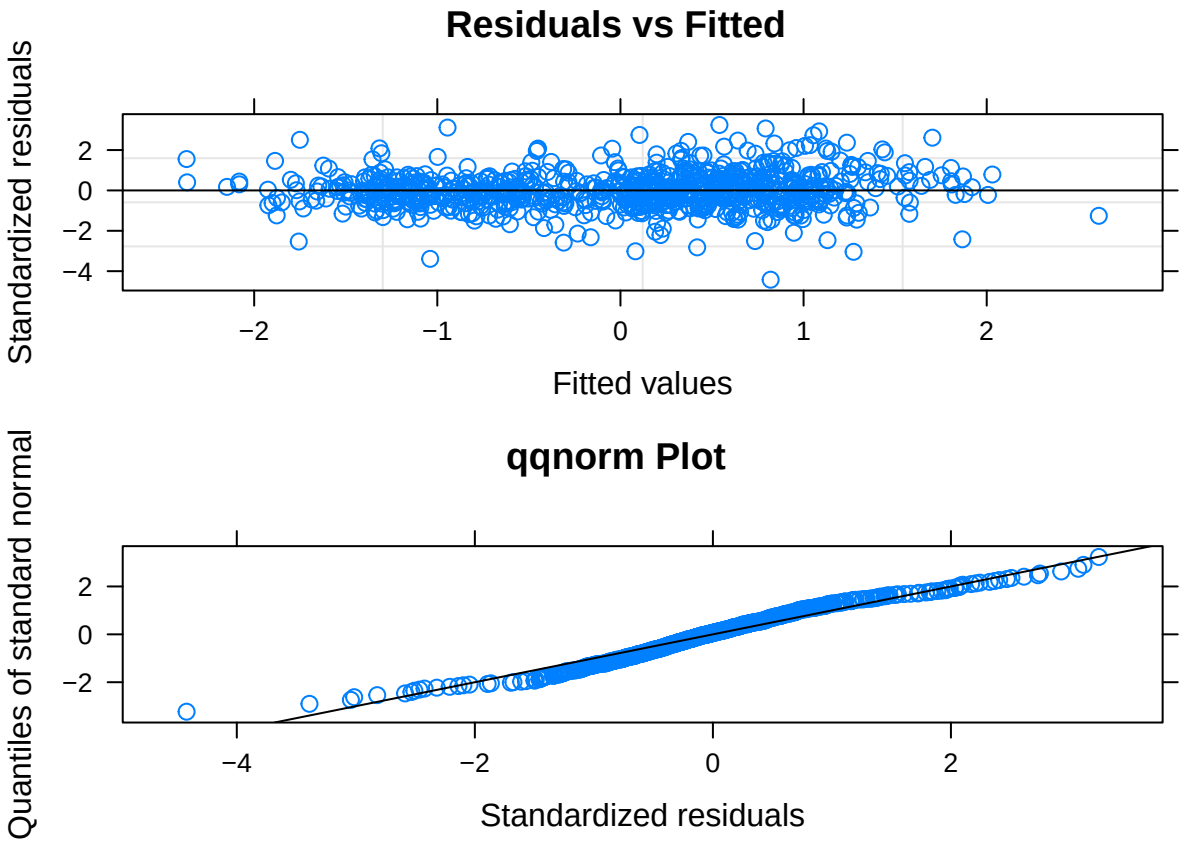
Autocorrelation:

```
vif_bot <- as.data.frame(vif(mbx6.BOT))
vif_bot_test <- max(vif_bot$"vif(mbx6.BOT)") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Diagnostic plots:

```
ggarrange(plot(mbx6.BOT, main = "Residuals vs Fitted"),
          qqnorm(mbx6.BOT, abline = c(0,1),
                 main = "qqnorm Plot"), nrow = 2)
```



Model assumptions are met.

Anova output for all subsoil models (step-wise):

```
summary(mbx6.BOT)
```

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: AfSIS_depth_normal %>% filter(Depth == "Subsoil")
##        AIC      BIC    logLik
##   1106.115 1153.085 -543.0574
##
## Random effects:
##  Formula: ~1 | Site
##         (Intercept)
## StdDev:   0.4897167
##
##  Formula: ~1 | Cluster %in% Site
##         (Intercept)  Residual
## StdDev:   0.2542965 0.3891222
##
## Fixed effects: CORG ~ Clay_8um + pH + CIA + Mox + Caex + pH:Mox
##                  Value  Std.Error  DF   t-value p-value
## (Intercept) -0.1983574 0.08284013 446 -2.394461  0.0171
## Clay_8um     0.0455868 0.02635944 446  1.729428  0.0844
## pH          -0.3290670 0.04712241 446 -6.983238  0.0000
## CIA         -0.1040680 0.03215945 446 -3.236000  0.0013
## Mox          0.4147492 0.03339600 446 12.419131  0.0000
## Caex         0.4186212 0.04937509 446  8.478390  0.0000
## pH:Mox      -0.1517777 0.02667293 446 -5.690326  0.0000
##  Correlation:
##          (Intr) Cly_8m pH     CIA    Mox    Caex
## Clay_8um  0.055
## pH       -0.010  0.054
## CIA      -0.041 -0.222  0.205
## Mox       0.075 -0.283  0.177 -0.202
## Caex      0.029 -0.303 -0.558  0.101 -0.298
## pH:Mox    0.041  0.005  0.261  0.034 -0.027 -0.166
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -4.42191470 -0.51362238 -0.04447791  0.48136537  3.24220899
##
## Number of Observations: 810
## Number of Groups:
##              Site Cluster %in% Site
##                42                358
```

```
ava_BOT <- anova(mbx0.BOT,mbx1.BOT,mbx2.BOT,mbx3.BOT,mbx4.BOT,mbx5.BOT,mbx6.BOT)
ava_BOT
```

```
##          Model df      AIC      BIC    logLik   Test   L.Ratio p-value
## mbx0.BOT     1  4 1460.723 1479.512 -726.3617
## mbx1.BOT     2  5 1373.421 1396.907 -681.7107 1 vs 2  89.30198  <.0001
## mbx2.BOT     3  6 1372.978 1401.160 -680.4888 2 vs 3   2.44387  0.1180
## mbx3.BOT     4  7 1373.422 1406.301 -679.7109 3 vs 4   1.55580  0.2123
## mbx4.BOT     5  8 1188.602 1226.178 -586.3011 4 vs 5 186.81959  <.0001
## mbx5.BOT     6  9 1135.713 1177.987 -558.8567 5 vs 6  54.88874  <.0001
## mbx6.BOT     7 10 1106.115 1153.085 -543.0574 6 vs 7  31.59862  <.0001
```

```
r.squaredGLMM(mbx6.BOT)
```

```
##             R2m       R2c
## [1,] 0.4320004 0.8113549
```

Anova Summary table:
Table B5 in the supplement of the manuscript.

```
ava_Depth_gt <- rbind(ava_TOP, ava_BOT) %>%
  dplyr::select(-call) %>%
  gt(rowname_col = "Model") %>%
  tab_row_group(group = "Topsoil", rows = 1:7) %>%
  tab_row_group(group = "Subsoil", rows = 8:14) %>%
  row_group_order(
    groups = c("Topsoil", "Subsoil")) %>%
  fmt_number(columns = c(3:5,7), decimals = 2) %>%
  fmt_number(columns = 8, decimals = 4) %>%
  cols_align(align = "center", columns = c(2:8))

gtsave(ava_Depth_gt, filename = "AfSIS_RefData2_AnovaDepth.rtf")
```

6.2.2 pH_{H2O} models

This section contains the code for the four pH models.

Normalize and standardize each sub-group:

```
AfSIS_pH_normal <- AfSIS_RefData_noNA %>%
  dplyr::select(-Longitude, -Latitude,-n_wet_month) %>%
  dplyr::mutate(Mox = 1/2 * Feox + Alox) %>%
  group_by(pH_Class) %>%
  mutate_if(is.numeric, ~predict(bestNormalize::boxcox(.x)))
```

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

Strongly acidic:

Build models:

```
mbx0.sacid <- lme(CORG ~ 1,
                  random = ~1|Site/Cluster/Profile, method = "ML",
                  data = AfSIS_pH_normal %>% filter(pH_Class == "strongly acidic"))
mbx1.sacid <- update(mbx0.sacid, ~. + Clay_8um)
mbx2.sacid <- update(mbx1.sacid, ~. + CIA)
mbx3.sacid <- update(mbx2.sacid, ~. + Mox)
mbx4.sacid <- update(mbx3.sacid, ~. + Caex)
```

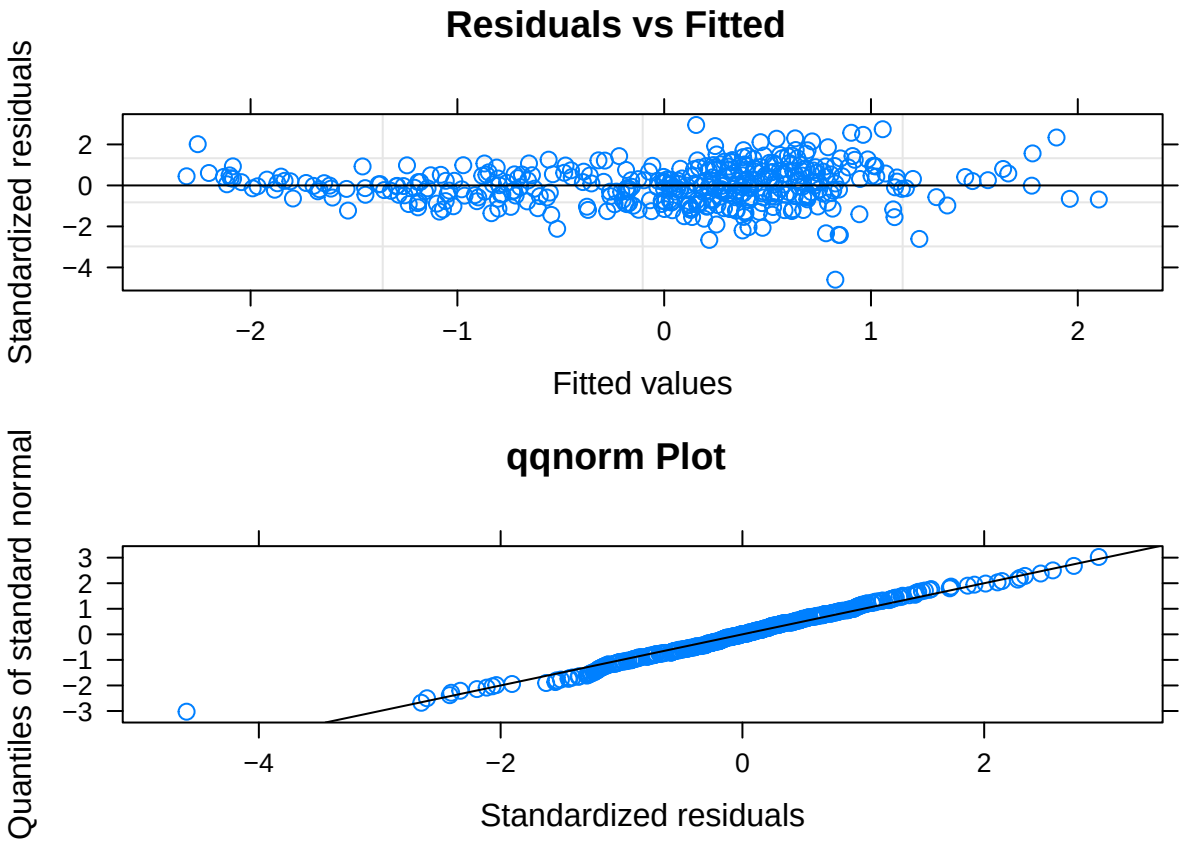
Autocorrelation:

```
vif_sacid <- as.data.frame(vif(mbx4.sacid))
vif_sacid_test <- max(vif_sacid$"vif(mbx4.sacid)") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Diagnostic plots:

```
ggarrange(plot(mbx4.sacid, main = "Residuals vs Fitted"),
          qqnorm(mbx4.sacid, abline = c(0,1),
                 main = "qqnorm Plot"), nrow = 2)
```



Model assumptions are met. The first plot is a little bit borderline.

Anova output for all strongly acid models (step-wise):

```
summary(mbx4.sacid)
```

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: AfSIS_pH_normal %>% filter(pH_Class == "strongly acidic")
##        AIC     BIC    logLik
##   690.6762 726.689 -336.3381
##
## Random effects:
##  Formula: ~1 | Site
##         (Intercept)
## StdDev:   0.4719538
##
##  Formula: ~1 | Cluster %in% Site
##         (Intercept)
## StdDev:   0.2024134
##
##  Formula: ~1 | Profile %in% Cluster %in% Site
##          (Intercept)  Residual
## StdDev: 7.007453e-05 0.4991287
```

```
##
## Fixed effects: CORG ~ Clay_8um + CIA + Mox + Caex
##                  Value  Std.Error  DF   t-value p-value
## (Intercept) -0.4995285 0.11499695 164 -4.343841  0.0000
## Clay_8um    -0.1628954 0.03896771  95 -4.180267  0.0001
## CIA         -0.0489030 0.04901814  95 -0.997652  0.3210
## Mox          0.7661011 0.05107408  95 14.999803  0.0000
## Caex         0.2156531 0.04377426  95  4.926482  0.0000
##  Correlation:
##          (Intr) Cly_8m CIA    Mox
## Clay_8um  0.129
## CIA       0.114 -0.196
## Mox       0.124 -0.123 -0.268
## Caex     -0.167 -0.079  0.294 -0.283
##
## Standardized Within-Group Residuals:
##          Min           Q1          Med           Q3          Max
## -4.597917771 -0.571913853 -0.009348452  0.560737291  2.947949226
##
## Number of Observations: 404
## Number of Groups:
##                            Site              Cluster %in% Site
##                              27                            141
## Profile %in% Cluster %in% Site
##                             305
```

```
ava_sacid <- anova(mbx0.sacid,mbx1.sacid,mbx2.sacid,mbx3.sacid,mbx4.sacid)
ava_sacid
```

```
##            Model df      AIC      BIC    logLik   Test   L.Ratio p-value
## mbx0.sacid     1  5 909.2250 929.2321 -449.6125
## mbx1.sacid     2  6 909.3178 933.3263 -448.6589 1 vs 2   1.90719  0.1673
## mbx2.sacid     3  7 911.3099 939.3198 -448.6550 2 vs 3   0.00788  0.9293
## mbx3.sacid     4  8 712.1829 744.1942 -348.0915 3 vs 4 201.12701  <.0001
## mbx4.sacid     5  9 690.6762 726.6890 -336.3381 4 vs 5  23.50670  <.0001
```

```
r.squaredGLMM(mbx4.sacid)
```

```
##           R2m       R2c
## [1,] 0.564682 0.7885299
```

Moderately acidic:

Build models:

```
mbx0.macid <- lme(CORG ~ 1,
                  random = ~1|Site/Cluster/Profile, method = "ML",
                  data = AfSIS_pH_normal %>% filter(pH_Class == "moderately acidic"))
mbx1.macid <- update(mbx0.macid, ~. + Clay_8um)
mbx2.macid <- update(mbx1.macid, ~. + CIA)
mbx3.macid <- update(mbx2.macid, ~. + Mox)
mbx4.macid <- update(mbx3.macid, ~. + Caex)
```

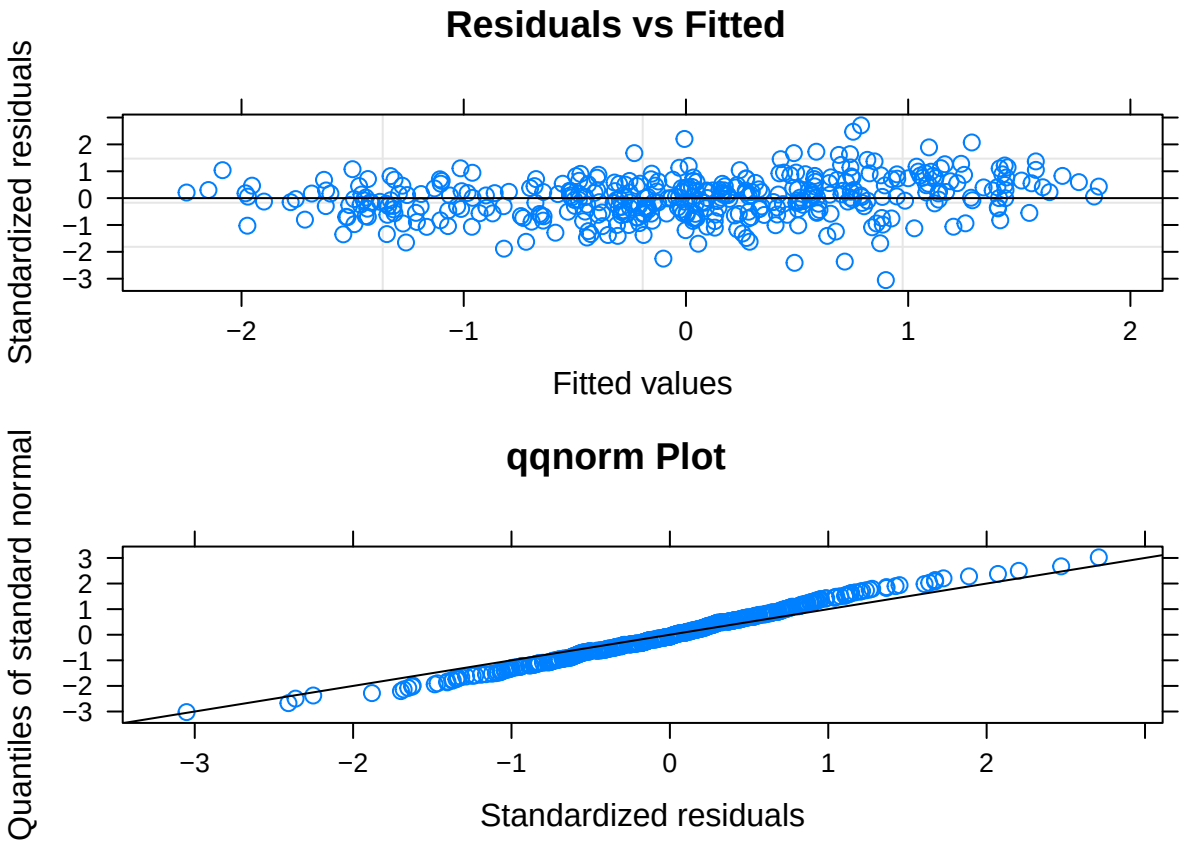
Autocorrelation:

```
vif_macid <- as.data.frame(vif(mbx4.macid))
vif_macid_test <- max(vif_macid$"vif(mbx4.macid)") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Diagnostic plots:

```
ggarrange(plot(mbx4.macid, main = "Residuals vs Fitted"),
          qqnorm(mbx4.macid, abline = c(0,1),
                 main = "qqnorm Plot"), nrow = 2)
```



Model assumptions are met.

Anova output for all moderately acid models (step-wise):

```r
summary(mbx4.macid)
```

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: AfSIS_pH_normal %>% filter(pH_Class == "moderately acidic")
##       AIC      BIC    logLik
##   679.029 714.9296 -330.5145
##
## Random effects:
##  Formula: ~1 | Site
##         (Intercept)
## StdDev:   0.4094074
##
##  Formula: ~1 | Cluster %in% Site
##         (Intercept)
## StdDev:   0.3211145
##
##  Formula: ~1 | Profile %in% Cluster %in% Site
##         (Intercept)  Residual
## StdDev:   0.2182148 0.3941462
##
## Fixed effects: CORG ~ Clay_8um + CIA + Mox + Caex
##                  Value  Std.Error  DF   t-value p-value
## (Intercept) -0.0991955 0.08865231 160 -1.118928  0.2648
## Clay_8um    -0.1358997 0.05000271  49 -2.717847  0.0091
## CIA         -0.0842539 0.04225970  49 -1.993717  0.0518
## Mox          0.5714311 0.06371937  49  8.967934  0.0000
## Caex         0.4100842 0.05211805  49  7.868371  0.0000
##  Correlation:
##          (Intr) Cly_8m CIA    Mox
## Clay_8um  0.026
## CIA      -0.006 -0.417
## Mox       0.164 -0.280 -0.214
## Caex      0.048 -0.363  0.404 -0.376
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -3.05150915 -0.52754479  0.03068403  0.48878160  2.70808360
##
## Number of Observations: 399
## Number of Groups:
##                            Site              Cluster %in% Site
##                              34                            186
## Profile %in% Cluster %in% Site
##                             346
```

```
ava_macid <- anova(mbx0.macid,mbx1.macid,mbx2.macid,mbx3.macid,mbx4.macid)
ava_macid
```

```
##            Model df      AIC      BIC    logLik   Test   L.Ratio p-value
## mbx0.macid     1  5 876.3942 896.3390 -433.1971
## mbx1.macid     2  6 864.4234 888.3571 -426.2117 1 vs 2  13.97086   2e-04
## mbx2.macid     3  7 849.8214 877.7441 -417.9107 2 vs 3  16.60196  <.0001
## mbx3.macid     4  8 734.6006 766.5123 -359.3003 3 vs 4 117.22076  <.0001
## mbx4.macid     5  9 679.0290 714.9296 -330.5145 4 vs 5  57.57167  <.0001
```

```
r.squaredGLMM(mbx4.macid)
```

```
##            R2m       R2c
## [1,] 0.5175705 0.8417852
```

Neutral:

Build models:

```
mbx0.neut <- lme(CORG ~ 1,
                 random = ~1|Site/Cluster/Profile, method = "ML",
                 data = AfSIS_pH_normal %>% filter(pH_Class == "neutral"))
mbx1.neut <- update(mbx0.neut, ~. + Clay_8um)
mbx2.neut <- update(mbx1.neut, ~. + CIA)
mbx3.neut <- update(mbx2.neut, ~. + Mox)
mbx4.neut <- update(mbx3.neut, ~. + Caex)
```

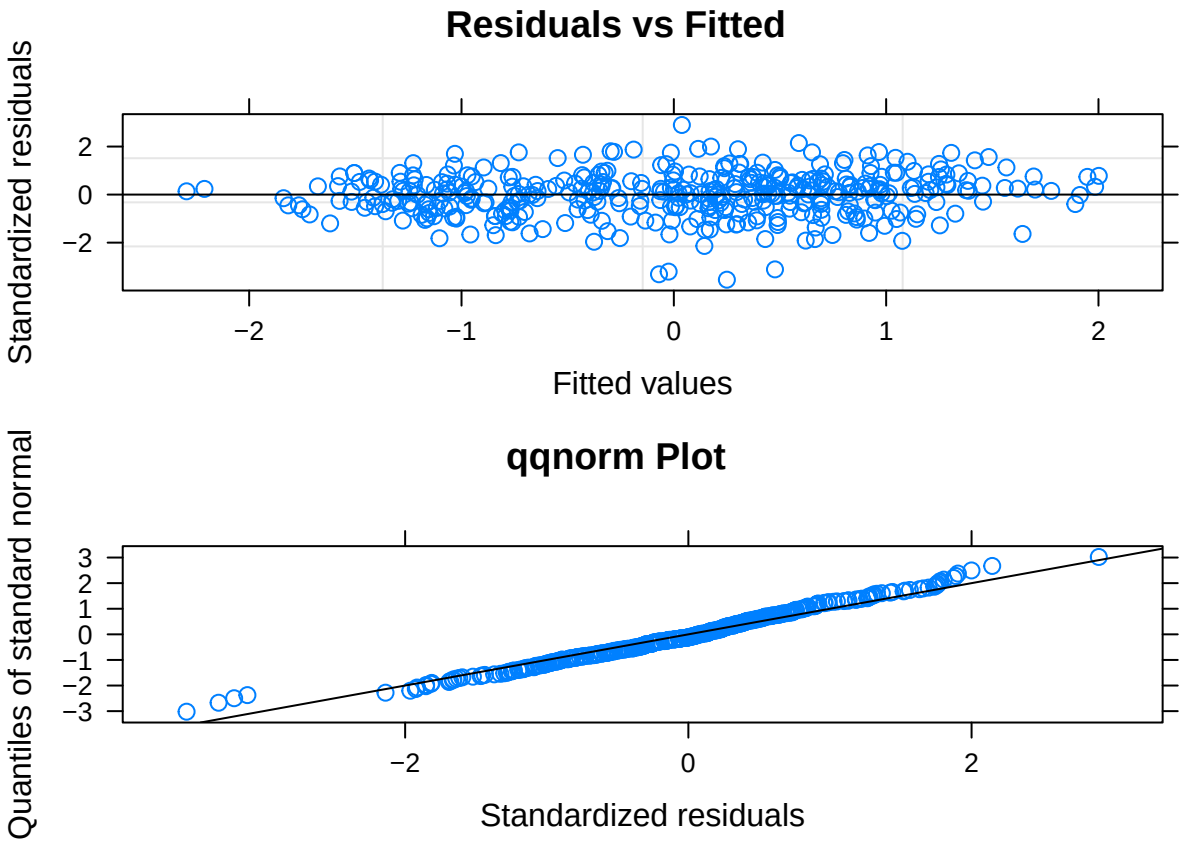
Autocorrelation:

```
vif_neut <- as.data.frame(vif(mbx4.neut))
vif_neut_test <- max(vif_neut$"vif(mbx4.neut)") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Diagnostic plots:

```
ggarrange(plot(mbx4.neut, main = "Residuals vs Fitted"),
          qqnorm(mbx4.neut, abline = c(0,1),
                 main = "qqnorm Plot"), nrow = 2)
```



Model assumptions are met.

Anova output for all neutral models (step-wise):

```
summary(mbx4.neut)
```

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: AfSIS_pH_normal %>% filter(pH_Class == "neutral")
##        AIC      BIC    logLik
##   581.0317 616.9097 -281.5158
##
## Random effects:
##  Formula: ~1 | Site
##         (Intercept)
## StdDev:    0.477398
##
##  Formula: ~1 | Cluster %in% Site
##         (Intercept)
## StdDev:   0.2130193
##
##  Formula: ~1 | Profile %in% Cluster %in% Site
##          (Intercept)  Residual
## StdDev: 1.820411e-05 0.4118728
```

```
##
## Fixed effects: CORG ~ Clay_8um + CIA + Mox + Caex
##                  Value  Std.Error  DF   t-value p-value
## (Intercept) -0.1087206 0.09927213 163 -1.095177  0.2751
## Clay_8um     0.0668009 0.04433256  42  1.506814  0.1393
## CIA         -0.2976458 0.02987280  42 -9.963773  0.0000
## Mox          0.3021391 0.04452794  42  6.785382  0.0000
## Caex         0.3329591 0.05087764  42  6.544310  0.0000
##  Correlation:
##          (Intr) Cly_8m CIA    Mox
## Clay_8um  0.045
## CIA      -0.071 -0.276
## Mox       0.057 -0.240 -0.145
## Caex      0.180 -0.349  0.244 -0.295
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -3.54342589 -0.53611649  0.08255728  0.53741423  2.89833967
##
## Number of Observations: 398
## Number of Groups:
##                            Site              Cluster %in% Site
##                              33                            189
## Profile %in% Cluster %in% Site
##                             352
```

```
ava_neut <- anova(mbx0.neut,mbx1.neut,mbx2.neut,mbx3.neut,mbx4.neut)
ava_neut
```

```
##           Model df      AIC      BIC    logLik   Test  L.Ratio p-value
## mbx0.neut     1  5 785.8673 805.7996 -387.9337
## mbx1.neut     2  6 772.2221 796.1408 -380.1110 1 vs 2 15.64522   1e-04
## mbx2.neut     3  7 686.0611 713.9663 -336.0306 2 vs 3 88.16097  <.0001
## mbx3.neut     4  8 620.1642 652.0558 -302.0821 3 vs 4 67.89696  <.0001
## mbx4.neut     5  9 581.0317 616.9097 -281.5158 4 vs 5 41.13250  <.0001
```

```
r.squaredGLMM(mbx4.neut)
```

```
##            R2m       R2c
## [1,] 0.5224481 0.8170989
```

Alkaline:

Build models:

```
mbx0.alk <- lme(CORG ~ 1,
                random = ~1|Site/Cluster/Profile, method = "ML",
                data = AfSIS_pH_normal %>% filter(pH_Class == "alkaline"))
mbx1.alk <- update(mbx0.alk, ~. + Clay_8um)
mbx2.alk <- update(mbx1.alk, ~. + CIA)
mbx3.alk <- update(mbx2.alk, ~. + Mox)
mbx4.alk <- update(mbx3.alk, ~. + Caex)
```

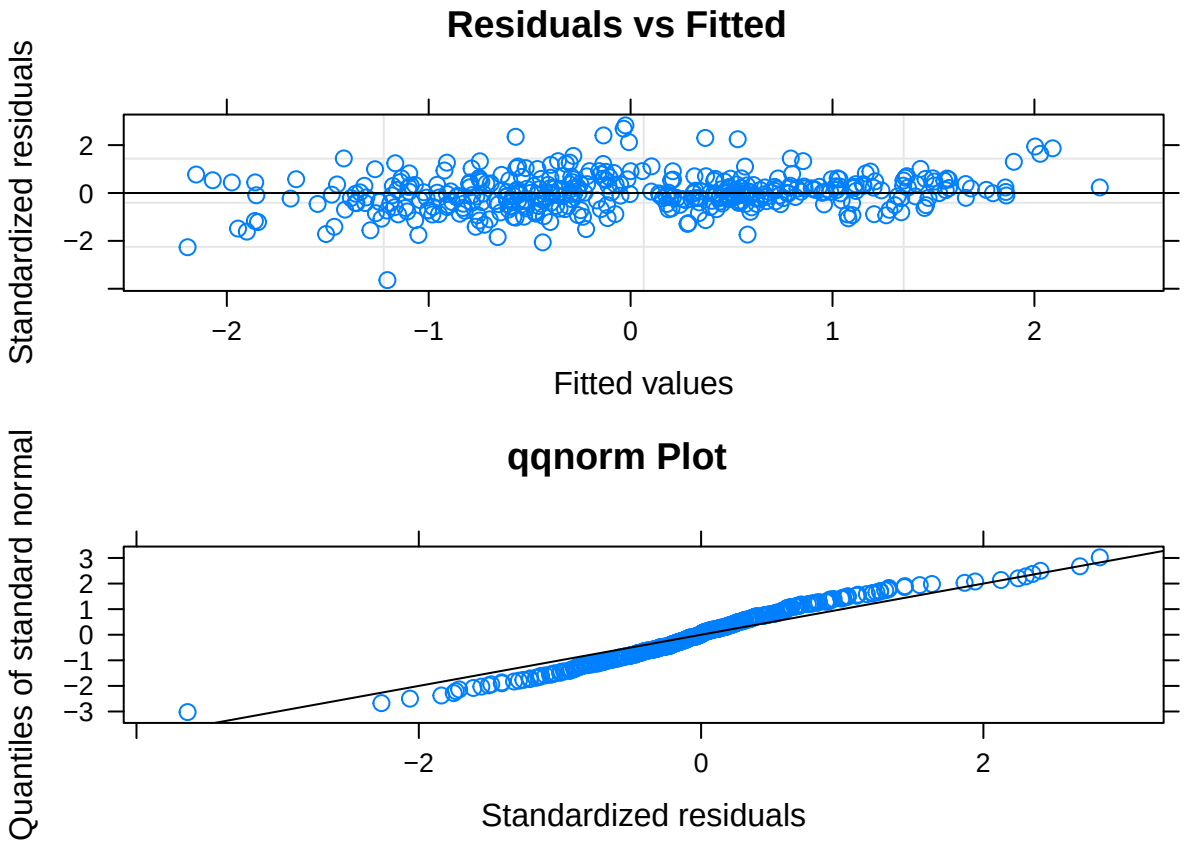
Autocorrelation:

```
vif_alk <- as.data.frame(vif(mbx4.alk))
vif_alk_test <- max(vif_alk$"vif(mbx4.alk)") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Diagnostic plots:

```
ggarrange(plot(mbx4.alk, main = "Residuals vs Fitted"),
          qqnorm(mbx4.alk, abline = c(0,1),
                 main = "qqnorm Plot"), nrow = 2)
```



Model assumptions are met.

Anova output for all alkaline models (step-wise):

```r
summary(mbx4.alk)
```

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: AfSIS_pH_normal %>% filter(pH_Class == "alkaline")
##        AIC      BIC    logLik
##   592.5831 628.5063 -287.2915
##
## Random effects:
##  Formula: ~1 | Site
##         (Intercept)
## StdDev:    0.595955
##
##  Formula: ~1 | Cluster %in% Site
##         (Intercept)
## StdDev:   0.1441046
##
##  Formula: ~1 | Profile %in% Cluster %in% Site
##         (Intercept)  Residual
## StdDev:    0.317386 0.3438606
##
## Fixed effects: CORG ~ Clay_8um + CIA + Mox + Caex
##                  Value  Std.Error  DF   t-value p-value
## (Intercept) -0.0826417 0.16256046 221 -0.508375  0.6117
## Clay_8um    -0.1350680 0.03947549  76 -3.421567  0.0010
## CIA         -0.0458182 0.03441432  76 -1.331372  0.1870
## Mox          0.1186411 0.03966816  76  2.990840  0.0037
## Caex         0.3845744 0.04763336  76  8.073636  0.0000
##  Correlation:
##          (Intr) Cly_8m CIA    Mox
## Clay_8um  0.024
## CIA      -0.125 -0.102
## Mox       0.003 -0.185  0.094
## Caex      0.066 -0.500 -0.201 -0.360
##
## Standardized Within-Group Residuals:
##          Min           Q1          Med           Q3          Max
## -3.638093299 -0.411829300 -0.005296146  0.370981139  2.824614787
##
## Number of Observations: 400
## Number of Groups:
##                            Site              Cluster %in% Site
##                              17                             99
## Profile %in% Cluster %in% Site
##                             320
```

```
ava_alk <- anova(mbx0.alk,mbx1.alk,mbx2.alk,mbx3.alk,mbx4.alk)
ava_alk
```

```
##          Model df      AIC      BIC    logLik   Test  L.Ratio p-value
## mbx0.alk     1  5 688.7103 708.6676 -339.3552
## mbx1.alk     2  6 679.0666 703.0154 -333.5333 1 vs 2 11.64370  0.0006
## mbx2.alk     3  7 681.0425 708.9827 -333.5212 2 vs 3  0.02417  0.8765
## mbx3.alk     4  8 651.4482 683.3799 -317.7241 3 vs 4 31.59425  <.0001
## mbx4.alk     5  9 592.5831 628.5063 -287.2915 4 vs 5 60.86514  <.0001
```

```
r.squaredGLMM(mbx4.alk)
```

```
##             R2m       R2c
## [1,] 0.1885064 0.8387112
```

Anova Summary table:
Table B6 in the supplement of the manuscript.

```
ava_pH_gt <- rbind(ava_sacid, ava_macid, ava_neut, ava_alk) %>%
  dplyr::select(-call) %>%
  gt(rowname_col = "Model") %>%
  tab_row_group(group = "Strongly acidic (3.9-5.2 pH)", rows = 1:5) %>%
  tab_row_group(group = "Moderately acidic (5.2-6.1 pH)", rows = 6:10) %>%
  tab_row_group(group = "Neutral (6.1-7.5 pH)", rows = 11:15) %>%
  tab_row_group(group = "Alkaline (7.5-9.9 pH)", rows = 16:20) %>%
  row_group_order(
    groups = c("Strongly acidic (3.9-5.2 pH)",
               "Moderately acidic (5.2-6.1 pH)",
               "Neutral (6.1-7.5 pH)",
               "Alkaline (7.5-9.9 pH)")) %>%
  fmt_number(columns = c(3:5,7), decimals = 2) %>%
  fmt_number(columns = 8, decimals = 4) %>%
  cols_align(align = "center", columns = c(2:8))

gtsave(ava_pH_gt, filename = "AfSIS_RefData2_AnovapH.rtf")
```

6.2.2.1 CIA models

This section is providing the code for the moderately and highly weathered models, respectively.

Normalize and standardize each sub-group:

```
AfSIS_CIA_normal <- AfSIS_RefData_noNA %>%
dplyr::select(-Longitude, -Latitude, -n_wet_month) %>%
  mutate(Mox = 1/2 * Feox + Alox) %>%
  group_by(CIA_Class) %>%
  mutate_if(is.numeric, ~predict(bestNormalize::boxcox(.x)))
```

Moderately weathered:

Build models:

```
mbx0.CIAmod <- lme(CORG ~ 1,
                   random = ~1|Site/Cluster/Profile, method = "ML",
                   data = AfSIS_CIA_normal %>% filter(CIA_Class == "moderate"))
mbx1.CIAmod <- update(mbx0.CIAmod, ~. + Clay_8um)
mbx2.CIAmod <- update(mbx1.CIAmod, ~. + pH)
mbx3.CIAmod <- update(mbx2.CIAmod, ~. + Mox)
mbx4.CIAmod <- update(mbx3.CIAmod, ~. + Caex)
mbx5.CIAmod <- update(mbx4.CIAmod, ~. + pH*Mox)
```

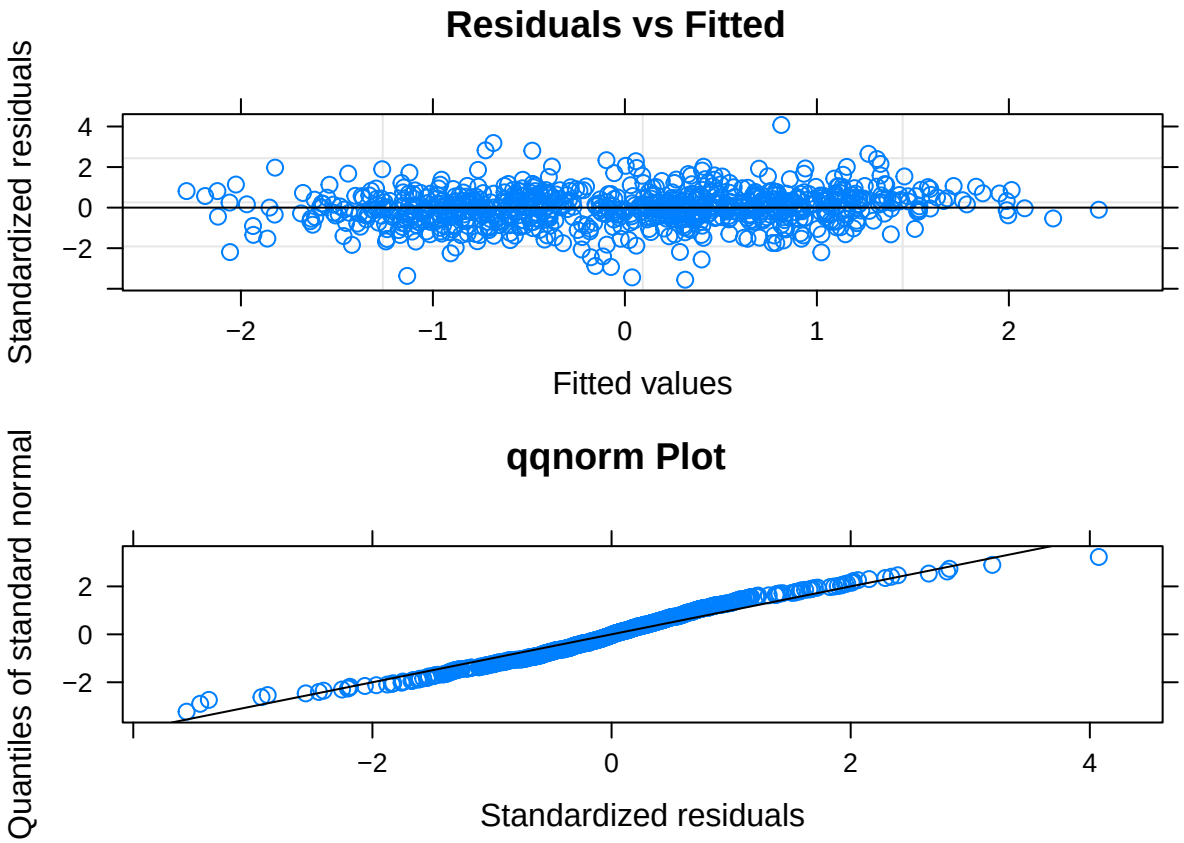
Autocorrelation:

```
vif_mod <- as.data.frame(vif(mbx5.CIAmod))
vif_mod_test <- max(vif_mod$"vif(mbx5.CIAmod)") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Diagnostic plots:

```
ggarrange(plot(mbx5.CIAmod, main = "Residuals vs Fitted"),
          qqnorm(mbx5.CIAmod, abline = c(0,1),
                 main = "qqnorm Plot"), nrow = 2)
```



Model assumptions are met.

Anova output for all moderately weathered models (step-wise):

```
summary(mbx5.CIAmod)
```

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: AfSIS_CIA_normal %>% filter(CIA_Class == "moderate")
##        AIC     BIC    logLik
##   1151.672 1198.53 -565.8358
##
## Random effects:
##  Formula: ~1 | Site
##         (Intercept)
## StdDev:   0.4759728
##
##  Formula: ~1 | Cluster %in% Site
##         (Intercept)
## StdDev:   0.1941322
##
##  Formula: ~1 | Profile %in% Cluster %in% Site
##         (Intercept)  Residual
## StdDev:    0.191248 0.3958591
##
## Fixed effects: CORG ~ Clay_8um + pH + Mox + Caex + pH:Mox
##                  Value  Std.Error  DF   t-value p-value
## (Intercept) -0.0577251 0.09352627 429 -0.617207  0.5374
## Clay_8um    -0.1262774 0.02949055 121 -4.281961  0.0000
## pH          -0.3048058 0.03623837 121 -8.411136  0.0000
## Mox          0.1752278 0.03131202 121  5.596184  0.0000
## Caex         0.5624563 0.03758580 121 14.964595  0.0000
## pH:Mox      -0.0860093 0.01935993 121 -4.442646  0.0000
##  Correlation:
##          (Intr) Cly_8m pH     Mox    Caex
## Clay_8um  0.042
## pH        0.126  0.053
## Mox       0.013 -0.209  0.216
## Caex      0.101 -0.435 -0.323 -0.398
## pH:Mox   -0.015 -0.097  0.198  0.291 -0.099
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -3.55417923 -0.46816528  0.01375396  0.47170274  4.07478271
##
## Number of Observations: 801
## Number of Groups:
##                            Site              Cluster %in% Site
##                              35                             246
```

```
## Profile %in% Cluster %in% Site
##                            675
```

```
ava_CIAmod <- anova(mbx0.CIAmod,mbx1.CIAmod,mbx2.CIAmod,mbx3.CIAmod,mbx4.CIAmod,mbx5.CIAmod)
ava_CIAmod
```

```
##              Model df      AIC      BIC    logLik   Test   L.Ratio p-value
## mbx0.CIAmod      1  5 1535.350 1558.779 -762.6749
## mbx1.CIAmod      2  6 1495.429 1523.544 -741.7144 1 vs 2  41.92114  <.0001
## mbx2.CIAmod      3  7 1487.129 1519.930 -736.5645 2 vs 3  10.29961  0.0013
## mbx3.CIAmod      4  8 1352.694 1390.181 -668.3468 3 vs 4 136.43540  <.0001
## mbx4.CIAmod      5  9 1169.173 1211.346 -575.5865 4 vs 5 185.52061  <.0001
## mbx5.CIAmod      6 10 1151.671 1198.530 -565.8358 5 vs 6  19.50155  <.0001
```

```
r.squaredGLMM(mbx5.CIAmod)
```

```
##           R2m       R2c
## [1,] 0.434201 0.8062081
```

Higly weathered:

Build models:

```
mbx0.CIAhigh <- lme(CORG ~ 1,
                    random = ~1|Site/Cluster/Profile, method = "ML",
                    data = AfSIS_CIA_normal %>% filter(CIA_Class == "high"))
mbx1.CIAhigh <- update(mbx0.CIAhigh, ~. + Clay_8um)
mbx2.CIAhigh <- update(mbx1.CIAhigh, ~. + pH)
mbx3.CIAhigh <- update(mbx2.CIAhigh, ~. + Mox)
mbx4.CIAhigh <- update(mbx3.CIAhigh, ~. + Caex)
mbx5.CIAhigh <- update(mbx4.CIAhigh, ~. + pH*Mox)
```

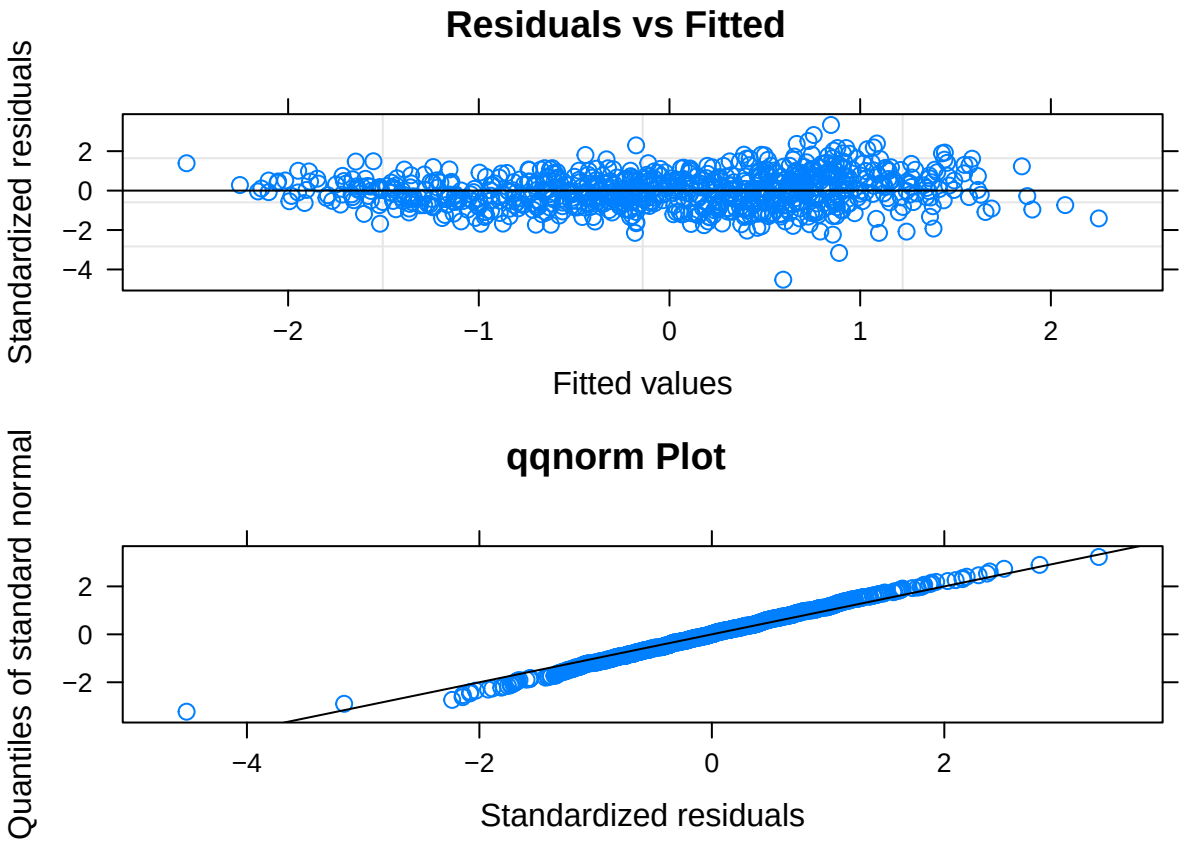
Autocorrelation:

```
vif_high <- as.data.frame(vif(mbx5.CIAhigh))
vif_high_test <- max(vif_high$"vif(mbx5.CIAhigh)") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Diagnostic plots:

```
ggarrange(plot(mbx5.CIAhigh, main = "Residuals vs Fitted"),
          qqnorm(mbx5.CIAhigh, abline = c(0,1),
                 main = "qqnorm Plot"), nrow = 2)
```



Model assumptions are met.

Anova output for all highly weathered models (step-wise):

```
summary(mbx5.CIAhigh)
```

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: AfSIS_CIA_normal %>% filter(CIA_Class == "high")
##        AIC      BIC    logLik
##   1215.271 1262.117 -597.6357
##
## Random effects:
##  Formula: ~1 | Site
##         (Intercept)
## StdDev:   0.5005484
##
##  Formula: ~1 | Cluster %in% Site
##         (Intercept)
## StdDev:   0.2307733
##
##  Formula: ~1 | Profile %in% Cluster %in% Site
##         (Intercept)  Residual
## StdDev:   0.1592132 0.4173483
```

```
##
## Fixed effects: CORG ~ Clay_8um + pH + Mox + Caex + pH:Mox
##                  Value  Std.Error  DF   t-value p-value
## (Intercept) -0.3519859 0.09307204 322 -3.781865   2e-04
## Clay_8um    -0.1464885 0.02660293 172 -5.506479   0e+00
## pH          -0.3267670 0.03934894 172 -8.304340   0e+00
## Mox          0.6055465 0.04208399 172 14.388999   0e+00
## Caex         0.4676428 0.04402820 172 10.621440   0e+00
## pH:Mox      -0.1822706 0.03011508 172 -6.052468   0e+00
##  Correlation:
##          (Intr) Cly_8m pH     Mox    Caex
## Clay_8um  0.071
## pH       -0.055  0.016
## Mox       0.182 -0.289  0.171
## Caex     -0.073 -0.084 -0.595 -0.245
## pH:Mox    0.029 -0.010  0.195 -0.198  0.004
##
## Standardized Within-Group Residuals:
##          Min           Q1          Med           Q3          Max
## -4.519200944 -0.580044863 -0.006732156  0.551659339  3.328923525
##
## Number of Observations: 800
## Number of Groups:
##                            Site              Cluster %in% Site
##                              38                            301
## Profile %in% Cluster %in% Site
##                             623
```

```
ava_CIAhigh <- anova(mbx0.CIAhigh,mbx1.CIAhigh,mbx2.CIAhigh,mbx3.CIAhigh,mbx4.CIAhigh,mbx5.CIA
ava_CIAhigh
```

```
##              Model df      AIC      BIC    logLik   Test   L.Ratio p-value
## mbx0.CIAhigh     1  5 1536.251 1559.674 -763.1255
## mbx1.CIAhigh     2  6 1538.148 1566.256 -763.0740 1 vs 2   0.10300  0.7483
## mbx2.CIAhigh     3  7 1535.929 1568.721 -760.9643 2 vs 3   4.21924  0.0400
## mbx3.CIAhigh     4  8 1343.695 1381.172 -663.8476 3 vs 4 194.23352  <.0001
## mbx4.CIAhigh     5  9 1248.824 1290.985 -615.4120 4 vs 5  96.87124  <.0001
## mbx5.CIAhigh     6 10 1215.271 1262.118 -597.6357 5 vs 6  35.55256  <.0001
```

```
r.squaredGLMM(mbx5.CIAhigh)
```

```
##            R2m       R2c
## [1,] 0.4664314 0.8153575
```

Anova Summary table:
Table B8 in the supplement of the manuscript.

```
ava_CIA_gt <- rbind(ava_CIAmod, ava_CIAhigh) %>%
  dplyr::select(-call) %>%
  gt(rowname_col = "Model") %>%
  tab_row_group(group = "Moderate weathering (10.3-88.1% CIA)", rows = 1:6) %>%
  tab_row_group(group = "High weathering (88.2-99.9% CIA)", rows = 7:12) %>%
  row_group_order(
    groups = c("Moderate weathering (10.3-88.1% CIA)",
               "High weathering (88.2-99.9% CIA)")) %>%
  fmt_number(columns = c(3:5,7), decimals = 2) %>%
  fmt_number(columns = 8, decimals = 4) %>%
  cols_align(align = "center", columns = c(2:8))

gtsave(ava_CIA_gt, filename = "AfSIS_RefData2_AnovaCIA.rtf")
```

6.2.3 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).

Normalize and standardize by sub-groups:

```
AfSIS_Seas_normal <- AfSIS_RefData_noNA %>%
  dplyr::select(-Longitude, -Latitude, -n_wet_month) %>%
  dplyr::mutate(Mox = 1/2 * Feox + Alox) %>%
  group_by(wet_month_Class) %>%
  mutate_if(is.numeric, ~predict(bestNormalize::boxcox(.x)))
```

0 number of wet months:

Build models:

```
mbx0.0wet <- lme(CORG ~ 1,
                 random = ~1|Site/Cluster/Profile, method = "ML",
                 data = AfSIS_Seas_normal %>% filter(wet_month_Class == "0"))
mbx1.0wet <- update(mbx0.0wet, ~. + Clay_8um)
mbx2.0wet <- update(mbx1.0wet, ~. + pH)
mbx3.0wet <- update(mbx2.0wet, ~. + CIA)
mbx4.0wet <- update(mbx3.0wet, ~. + Mox)
mbx5.0wet <- update(mbx4.0wet, ~. + Caex)
mbx6.0wet <- update(mbx5.0wet, ~. + pH*Mox)
```

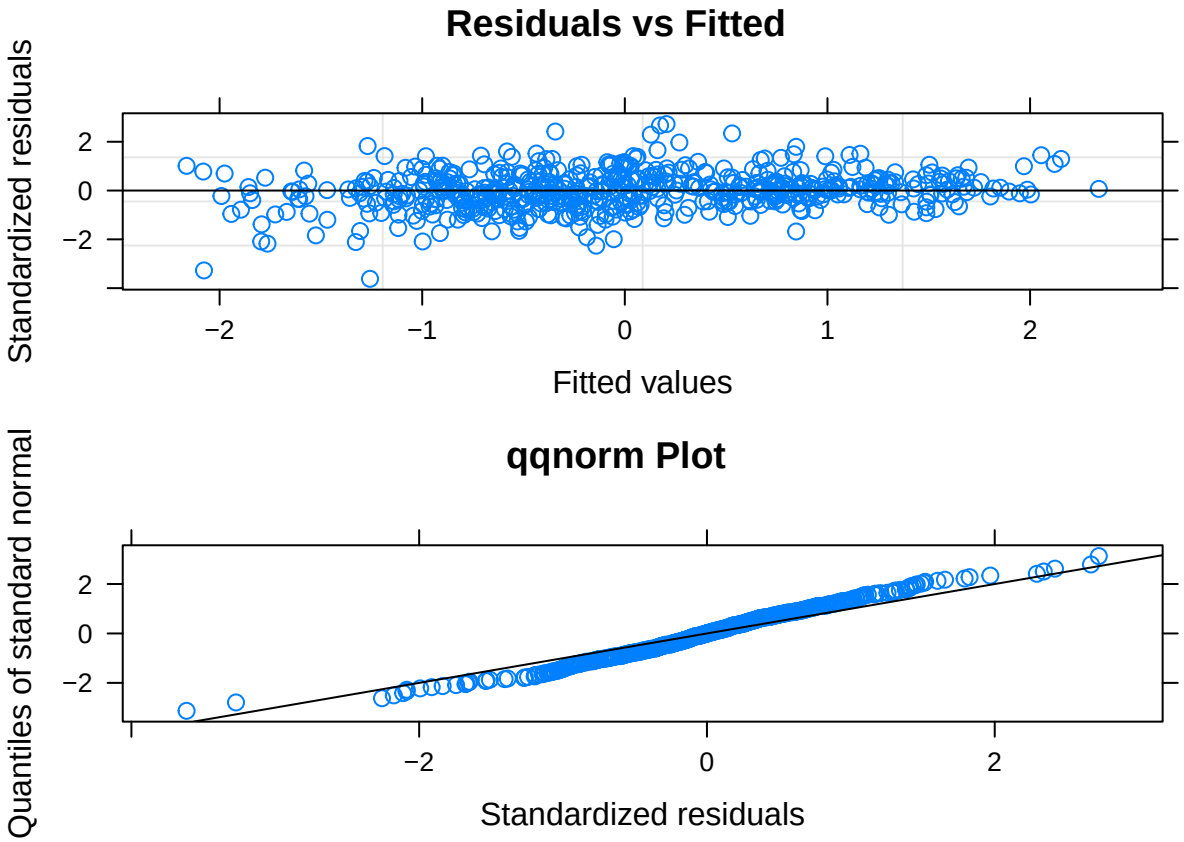
Autocorrelation:

```
vif_0wet <- as.data.frame(vif(mbx6.0wet))
vif_0wet_test <- max(vif_0wet$"vif(mbx6.0wet)") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Diagnostic plots:

```
ggarrange(plot(mbx6.0wet, main = "Residuals vs Fitted"),
          qqnorm(mbx6.0wet, abline = c(0,1),
                 main = "qqnorm Plot"), nrow = 2)
```



Model assumptions are met.

Anova output for all 0 number of wet months models (step-wise):

```
summary(mbx6.0wet)
```

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: AfSIS_Seas_normal %>% filter(wet_month_Class == "0")
##        AIC      BIC    logLik
##   840.0764 887.9169 -409.0382
##
## Random effects:
##  Formula: ~1 | Site
##         (Intercept)
## StdDev:   0.5621932
##
```

```
##  Formula: ~1 | Cluster %in% Site
##         (Intercept)
## StdDev:   0.1714326
##
##  Formula: ~1 | Profile %in% Cluster %in% Site
##         (Intercept)  Residual
## StdDev:   0.2980618 0.3512527
##
## Fixed effects: CORG ~ Clay_8um + pH + CIA + Mox + Caex + pH:Mox
##                 Value  Std.Error  DF   t-value p-value
## (Intercept)  0.0163270 0.14929408 269  0.109362  0.9130
## Clay_8um    -0.0915667 0.03949321 111 -2.318543  0.0223
## pH          -0.2164292 0.04409256 111 -4.908519  0.0000
## CIA         -0.1554676 0.03261935 111 -4.766116  0.0000
## Mox          0.1515214 0.04038214 111  3.752187  0.0003
## Caex         0.5700656 0.05532753 111 10.303470  0.0000
## pH:Mox      -0.0418532 0.02727945 111 -1.534239  0.1278
##  Correlation:
##          (Intr) Cly_8m pH     CIA    Mox    Caex
## Clay_8um -0.011
## pH        0.080  0.007
## CIA      -0.060 -0.136  0.196
## Mox       0.021 -0.263  0.102  0.064
## Caex      0.088 -0.449 -0.291 -0.086 -0.353
## pH:Mox   -0.059  0.005  0.339 -0.009 -0.114 -0.048
##
## Standardized Within-Group Residuals:
##          Min           Q1          Med           Q3          Max
## -3.616512598 -0.417120620 -0.002277899  0.435416449  2.723274893
##
## Number of Observations: 572
## Number of Groups:
##                            Site              Cluster %in% Site
##                              16                            186
## Profile %in% Cluster %in% Site
##                             455
```

```
ava_0wet <- anova(mbx0.0wet,mbx1.0wet,mbx2.0wet,mbx3.0wet,mbx4.0wet,mbx5.0wet,mbx6.0wet)
ava_0wet
```

```
##           Model df       AIC       BIC    logLik   Test  L.Ratio p-value
## mbx0.0wet     1  5 1016.2822 1038.0279 -503.1411
## mbx1.0wet     2  6  989.8852 1015.9800 -488.9426 1 vs 2 28.39700  <.0001
## mbx2.0wet     3  7  990.4100 1020.8540 -488.2050 2 vs 3  1.47523  0.2245
## mbx3.0wet     4  8  980.6457 1015.4388 -482.3228 3 vs 4 11.76432  0.0006
## mbx4.0wet     5  9  934.8227  973.9650 -458.4114 4 vs 5 47.82293  <.0001
## mbx5.0wet     6 10  840.4025  883.8939 -410.2012 5 vs 6 96.42024  <.0001
```

```
## mbx6.0wet     7 11  840.0764  887.9169 -409.0382 6 vs 7  2.32613  0.1272
```

```
r.squaredGLMM(mbx5.0wet)
```

```
##           R2m       R2c
## [1,] 0.330346 0.8595547
```

1-3 number of wet months:

Build models:

```
mbx0.1_3wet <- lme(CORG ~ 1,
                random = ~1|Site/Cluster/Profile, method = "ML",
                data = AfSIS_Seas_normal %>% filter(wet_month_Class == "1-3"))
mbx1.1_3wet <- update(mbx0.1_3wet, ~. + Clay_8um)
mbx2.1_3wet <- update(mbx1.1_3wet, ~. + pH)
mbx3.1_3wet <- update(mbx2.1_3wet, ~. + CIA)
mbx4.1_3wet <- update(mbx3.1_3wet, ~. + Mox)
mbx5.1_3wet <- update(mbx4.1_3wet, ~. + Caex)
mbx6.1_3wet <- update(mbx5.1_3wet, ~. + pH*Mox)
```

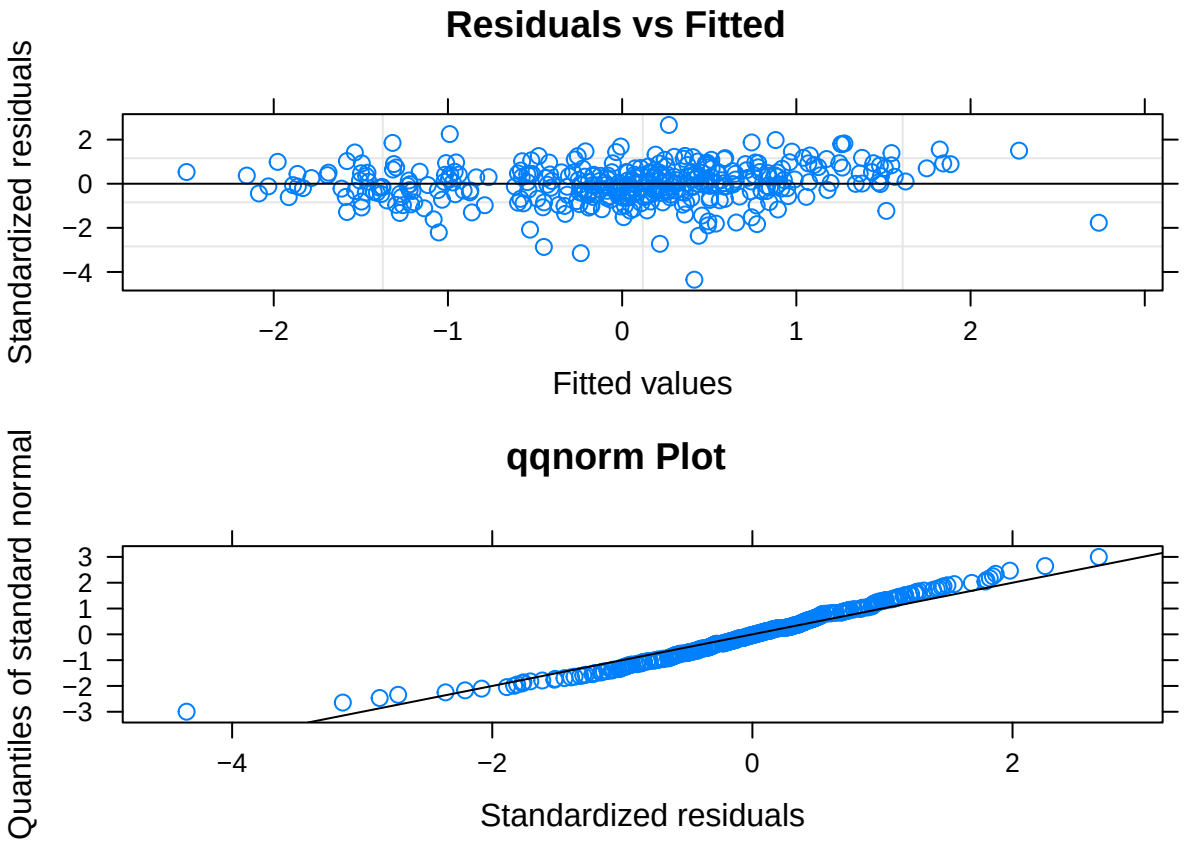
Autocorrelation:

```
vif_1_3wet <- as.data.frame(vif(mbx6.1_3wet))
vif_1_3wet_test <- max(vif_1_3wet$"vif(mbx6.1_3wet)") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Diagnostic plots:

```
ggarrange(plot(mbx6.1_3wet, main = "Residuals vs Fitted"),
          qqnorm(mbx6.1_3wet, abline = c(0,1),
                 main = "qqnorm Plot"), nrow = 2)
```



Model assumptions are met.

Anova output for all 1-3 number of wet months models (step-wise):

```
summary(mbx6.1_3wet)
```

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: AfSIS_Seas_normal %>% filter(wet_month_Class == "1-3")
##        AIC      BIC    logLik
##   599.7013 642.6603 -288.8506
##
## Random effects:
##  Formula: ~1 | Site
##         (Intercept)
## StdDev:   0.4256601
##
##  Formula: ~1 | Cluster %in% Site
##         (Intercept)
## StdDev:   0.2880689
##
##  Formula: ~1 | Profile %in% Cluster %in% Site
##         (Intercept)  Residual
## StdDev:   0.1806791 0.4273154
```

```
##
## Fixed effects: CORG ~ Clay_8um + pH + CIA + Mox + Caex + pH:Mox
##                  Value  Std.Error  DF    t-value p-value
## (Intercept) -0.1253104 0.13911333 181  -0.900779  0.3689
## Clay_8um    -0.0754912 0.04705944  71  -1.604167  0.1131
## pH          -0.3804571 0.04457801  71  -8.534637  0.0000
## CIA         -0.3592248 0.03407481  71 -10.542237  0.0000
## Mox          0.3748918 0.04762600  71   7.871578  0.0000
## Caex         0.7214074 0.07022050  71  10.273458  0.0000
## pH:Mox      -0.1427217 0.03083904  71  -4.627957  0.0000
##  Correlation:
##          (Intr) Cly_8m pH     CIA    Mox    Caex
## Clay_8um  0.036
## pH       -0.077  0.021
## CIA      -0.019 -0.354 -0.032
## Mox      -0.025 -0.305  0.335 -0.076
## Caex      0.135 -0.346 -0.573  0.286 -0.368
## pH:Mox    0.024  0.076  0.002 -0.051  0.066  0.053
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -4.35016241 -0.46313613  0.02137685  0.50180341  2.66320010
##
## Number of Observations: 367
## Number of Groups:
##                            Site              Cluster %in% Site
##                              12                            109
## Profile %in% Cluster %in% Site
##                             290
```

```
ava_1_3wet <- anova(mbx0.1_3wet,mbx1.1_3wet,mbx2.1_3wet,mbx3.1_3wet,mbx4.1_3wet,mbx5.1_3wet,mbx
ava_1_3wet
```

```
##             Model df      AIC      BIC    logLik   Test   L.Ratio p-value
## mbx0.1_3wet     1  5 933.0069 952.5337 -461.5034
## mbx1.1_3wet     2  6 912.8615 936.2937 -450.4308 1 vs 2  22.14535  <.0001
## mbx2.1_3wet     3  7 910.0750 937.4125 -448.0375 2 vs 3   4.78652  0.0287
## mbx3.1_3wet     4  8 811.9126 843.1555 -397.9563 3 vs 4 100.16242  <.0001
## mbx4.1_3wet     5  9 708.6976 743.8458 -345.3488 4 vs 5 105.21499  <.0001
## mbx5.1_3wet     6 10 618.4413 657.4949 -299.2206 5 vs 6  92.25629  <.0001
## mbx6.1_3wet     7 11 599.7013 642.6603 -288.8506 6 vs 7  20.74001  <.0001
```

```
r.squaredGLMM(mbx5.1_3wet)
```

```
##            R2m       R2c
## [1,] 0.6607345 0.8620473
```

4-7 number of wet months:

Build models:

```
mbx0.4_7wet <- lme(CORG ~ 1,
                   random = ~1|Site/Cluster/Profile, method = "ML",
                   data = AfSIS_Seas_normal %>% filter(wet_month_Class == "4-7"))
mbx1.4_7wet <- update(mbx0.4_7wet, ~. + Clay_8um)
mbx2.4_7wet <- update(mbx1.4_7wet, ~. + pH)
mbx3.4_7wet <- update(mbx2.4_7wet, ~. + CIA)
mbx4.4_7wet <- update(mbx3.4_7wet, ~. + Mox)
mbx5.4_7wet <- update(mbx4.4_7wet, ~. + Caex)
mbx6.4_7wet <- update(mbx5.4_7wet, ~. + pH*Mox)
```

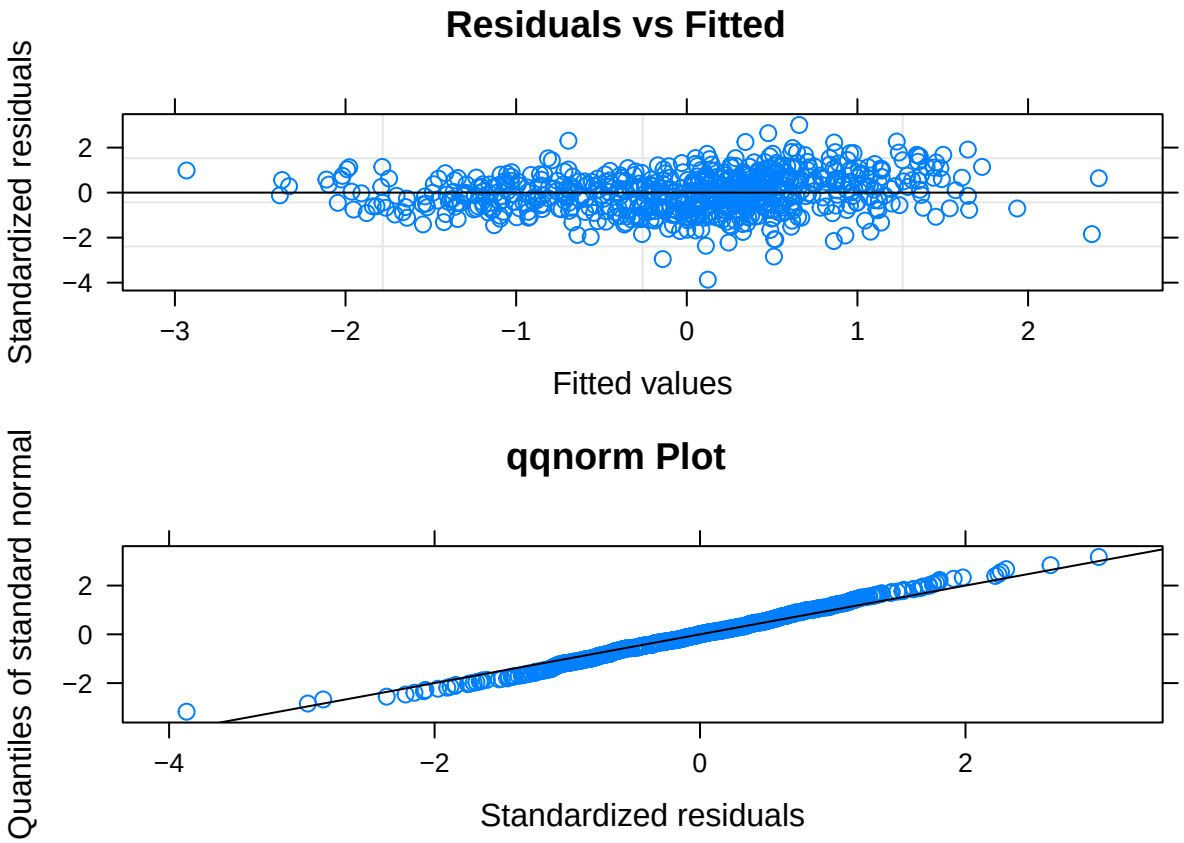
Autocorrelation:

```
vif_4_7wet <- as.data.frame(vif(mbx6.4_7wet))
vif_4_7wet_test <- max(vif_4_7wet$"vif(mbx6.4_7wet)") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Diagnostic plots:

```
ggarrange(plot(mbx6.4_7wet, main = "Residuals vs Fitted"),
          qqnorm(mbx6.4_7wet, abline = c(0,1),
                 main = "qqnorm Plot"), nrow = 2)
```



Model assumptions are met.

Anova output for all 4-7 number of wet months models (step-wise):

```
summary(mbx6.4_7wet)
```

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: AfSIS_Seas_normal %>% filter(wet_month_Class == "4-7")
##        AIC      BIC    logLik
##   1237.137 1286.585 -607.5683
##
## Random effects:
##  Formula: ~1 | Site
##         (Intercept)
## StdDev:   0.4988179
##
##  Formula: ~1 | Cluster %in% Site
##         (Intercept)
## StdDev:   0.3261345
##
##  Formula: ~1 | Profile %in% Cluster %in% Site
##         (Intercept)  Residual
## StdDev:   0.2217827 0.4942388
```

```
##
## Fixed effects: CORG ~ Clay_8um + pH + CIA + Mox + Caex + pH:Mox
##                 Value  Std.Error  DF   t-value p-value
## (Intercept) -0.2878181 0.14172641 338 -2.030801  0.0431
## Clay_8um    -0.1242916 0.03374543 150 -3.683212  0.0003
## pH          -0.3796374 0.05320083 150 -7.135930  0.0000
## CIA         -0.1735564 0.05047168 150 -3.438688  0.0008
## Mox          0.6265119 0.05279922 150 11.865931  0.0000
## Caex         0.6008705 0.06240118 150  9.629153  0.0000
## pH:Mox      -0.1796765 0.03877198 150 -4.634183  0.0000
##  Correlation:
##          (Intr) Cly_8m pH     CIA    Mox    Caex
## Clay_8um  0.072
## pH       -0.037  0.046
## CIA      -0.035 -0.350  0.196
## Mox       0.096 -0.050  0.097 -0.275
## Caex      0.019 -0.089 -0.525  0.198 -0.281
## pH:Mox    0.032 -0.069  0.055 -0.006 -0.427  0.067
##
## Standardized Within-Group Residuals:
##          Min           Q1          Med           Q3          Max
## -3.868495550 -0.623017815 -0.002887143  0.582630182  3.004391090
##
## Number of Observations: 662
## Number of Groups:
##                            Site              Cluster %in% Site
##                              17                            168
## Profile %in% Cluster %in% Site
##                             506
```

```
ava_4_7wet <- anova(mbx0.4_7wet,mbx1.4_7wet,mbx2.4_7wet,mbx3.4_7wet,mbx4.4_7wet,mbx5.4_7wet,mbx
ava_4_7wet
```

```
##             Model df      AIC      BIC    logLik   Test   L.Ratio p-value
## mbx0.4_7wet     1  5 1489.121 1511.598 -739.5607
## mbx1.4_7wet     2  6 1487.464 1514.435 -737.7318 1 vs 2   3.65766  0.0558
## mbx2.4_7wet     3  7 1488.855 1520.322 -737.4277 2 vs 3   0.60821  0.4355
## mbx3.4_7wet     4  8 1486.228 1522.190 -735.1140 3 vs 4   4.62743  0.0315
## mbx4.4_7wet     5  9 1339.022 1379.479 -660.5110 4 vs 5 149.20599  <.0001
## mbx5.4_7wet     6 10 1256.200 1301.153 -618.1003 5 vs 6  84.82156  <.0001
## mbx6.4_7wet     7 11 1237.137 1286.585 -607.5683 6 vs 7  21.06386  <.0001
```

```
r.squaredGLMM(mbx5.4_7wet)
```

```
##            R2m       R2c
## [1,] 0.4674492 0.7995353
```

Anova Summary table:
Table B7 in the supplement of the manuscript.

```
ava_nWet_gt <- rbind(ava_0wet, ava_1_3wet, ava_4_7wet) %>%
  dplyr::select(-call) %>%
  gt(rowname_col = "Model") %>%
  tab_row_group(group = "0 number of wet months", rows = 1:7) %>%
  tab_row_group(group = "1-3 number of wet months", rows = 8:14) %>%
  tab_row_group(group = "4-7 number of wet months", rows = 15:21) %>%
  row_group_order(
    groups = c("0 number of wet months",
               "1-3 number of wet months",
               "4-7 number of wet months")) %>%
  fmt_number(columns = c(3:5,7), decimals = 2) %>%
  fmt_number(columns = 8, decimals = 4) %>%
  cols_align(align = "center", columns = c(2:8))

gtsave(ava_nWet_gt, filename = "AfSIS_RefData2_AnovaWet.rtf")
```

6.2.4 Land cover models

This section contains the four models for the forest, cropland, grassland and other samples, respectively.

Normalize and standardize each sub-group:

```
AfSIS_landcover_normal <- AfSIS_RefData_noNA %>%
  dplyr::select(-Longitude, -Latitude, -n_wet_month) %>%
  dplyr::mutate(Mox = 1/2 * Feox + Alox) %>%
  group_by(LandCover) %>%
  mutate_if(is.numeric, ~predict(bestNormalize::boxcox(.x)))
```

Cropland:

Build models:

```
mbx0.Crop <- lme(CORG ~ 1,
                 random = ~1|Site/Cluster/Profile, method = "ML",
                 data = AfSIS_landcover_normal %>%
                   filter(LandCover == "Cropland"))
mbx1.Crop <- update(mbx0.Crop, ~. + Clay_8um)
mbx2.Crop <- update(mbx1.Crop, ~. + pH)
mbx3.Crop <- update(mbx2.Crop, ~. + CIA)
mbx4.Crop <- update(mbx3.Crop, ~. + Mox)
mbx5.Crop <- update(mbx4.Crop, ~. + Caex)
mbx6.Crop <- update(mbx5.Crop, ~. + pH*Mox)
```

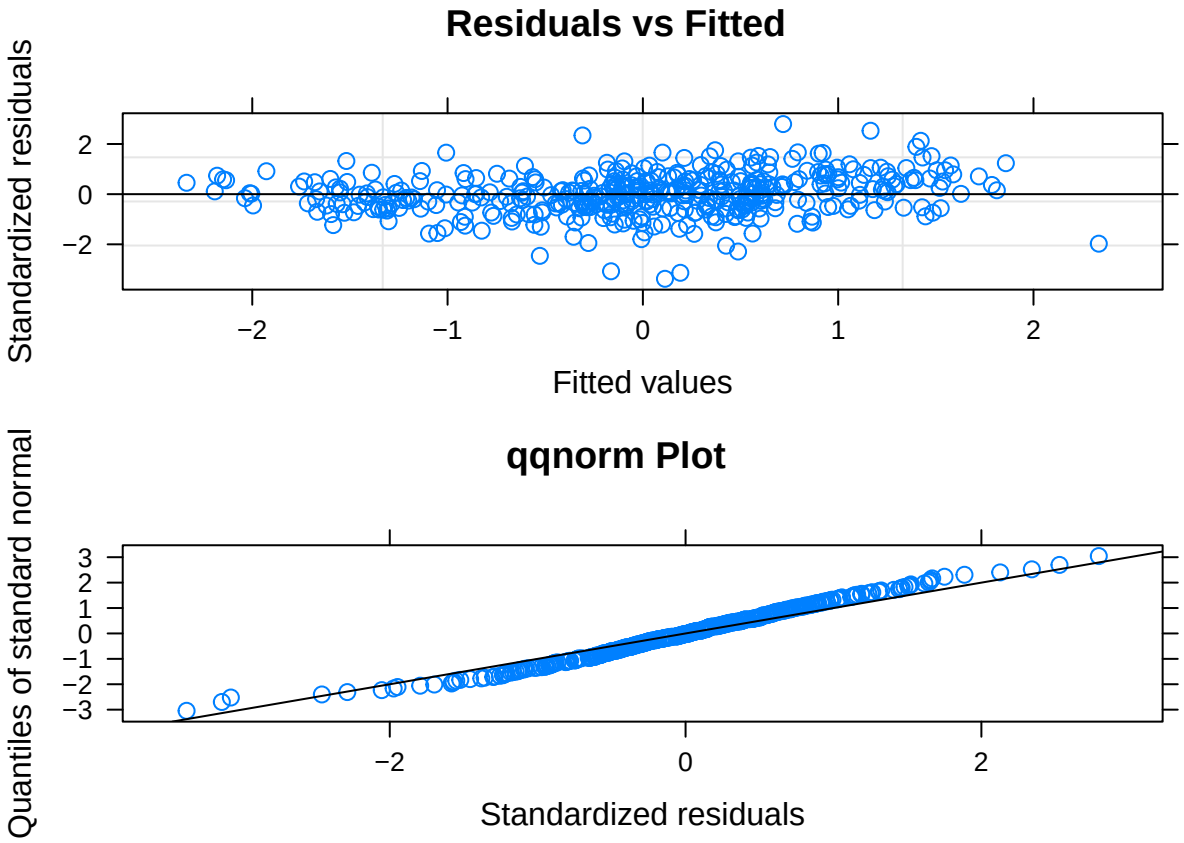
Autocorrelation:

```
vif_Crop <- as.data.frame(vif(mbx6.Crop))
vif_Crop_test <- max(vif_Crop$"vif(mbx6.Crop)") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Diagnostic plots:

```
ggarrange(plot(mbx6.Crop, main = "Residuals vs Fitted"),
          qqnorm(mbx6.Crop, abline = c(0,1),
                 main = "qqnorm Plot"), nrow = 2)
```



Model assumptions are met.

Anova output for all cropland models (step-wise):

```
summary(mbx6.Crop)
```

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: AfSIS_landcover_normal %>% filter(LandCover == "Cropland")
##        AIC      BIC    logLik
##   736.8044 781.4804 -357.4022
```

```
##
## Random effects:
##  Formula: ~1 | Site
##         (Intercept)
## StdDev:   0.4706105
##
##  Formula: ~1 | Cluster %in% Site
##         (Intercept)
## StdDev:   0.2618095
##
##  Formula: ~1 | Profile %in% Cluster %in% Site
##         (Intercept)  Residual
## StdDev:   0.2415631 0.4215576
##
## Fixed effects: CORG ~ Clay_8um + pH + CIA + Mox + Caex + pH:Mox
##                  Value  Std.Error  DF   t-value p-value
## (Intercept) -0.0178095 0.11091494 192 -0.160569  0.8726
## Clay_8um    -0.0584636 0.04568364  84 -1.279749  0.2042
## pH          -0.3061443 0.05440641  84 -5.626989  0.0000
## CIA         -0.2296943 0.04540893  84 -5.058349  0.0000
## Mox          0.4718395 0.06084963  84  7.754188  0.0000
## Caex         0.5831230 0.07070011  84  8.247838  0.0000
## pH:Mox      -0.2041613 0.04396248  84 -4.643990  0.0000
##  Correlation:
##          (Intr) Cly_8m pH     CIA    Mox    Caex
## Clay_8um  0.033
## pH       -0.041  0.022
## CIA      -0.065 -0.379  0.091
## Mox       0.160 -0.221  0.259 -0.193
## Caex      0.138 -0.242 -0.516  0.294 -0.351
## pH:Mox    0.119 -0.027  0.205  0.017  0.060  0.055
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -3.37365583 -0.46881915  0.01861612  0.51578215  2.79371638
##
## Number of Observations: 429
## Number of Groups:
##                            Site              Cluster %in% Site
##                              28                            147
## Profile %in% Cluster %in% Site
##                             339
```

```
ava_Crop <- anova(mbx0.Crop,mbx1.Crop,mbx2.Crop,mbx3.Crop,mbx4.Crop,mbx5.Crop,mbx6.Crop)
ava_Crop
```

```
##           Model df      AIC      BIC    logLik   Test  L.Ratio p-value
```

```
## mbx0.Crop     1  5 942.5679 962.8752 -466.2840
## mbx1.Crop     2  6 942.7653 967.1341 -465.3827 1 vs 2  1.80258  0.1794
## mbx2.Crop     3  7 943.7285 972.1587 -464.8642 2 vs 3  1.03689  0.3085
## mbx3.Crop     4  8 911.7222 944.2138 -447.8611 3 vs 4 34.00626  <.0001
## mbx4.Crop     5  9 817.9571 854.5102 -399.9785 4 vs 5 95.76511  <.0001
## mbx5.Crop     6 10 755.4934 796.1080 -367.7467 5 vs 6 64.46363  <.0001
## mbx6.Crop     7 11 736.8044 781.4804 -357.4022 6 vs 7 20.68910  <.0001
```

```
r.squaredGLMM(mbx6.Crop)
```

```
##             R2m       R2c
## [1,] 0.5512666 0.8484176
```

Forest:

Build models:

```
mbx0.For <- lme(CORG ~ 1,
                random = ~1|Site/Cluster/Profile, method = "ML",
                data = AfSIS_landcover_normal %>%
                  filter(LandCover == "Forest"))
mbx1.For <- update(mbx0.For, ~. + Clay_8um)
mbx2.For <- update(mbx1.For, ~. + pH)
mbx3.For <- update(mbx2.For, ~. + CIA)
mbx4.For <- update(mbx3.For, ~. + Mox)
mbx5.For <- update(mbx4.For, ~. + Caex)
mbx6.For <- update(mbx5.For, ~. + pH*Mox)
```

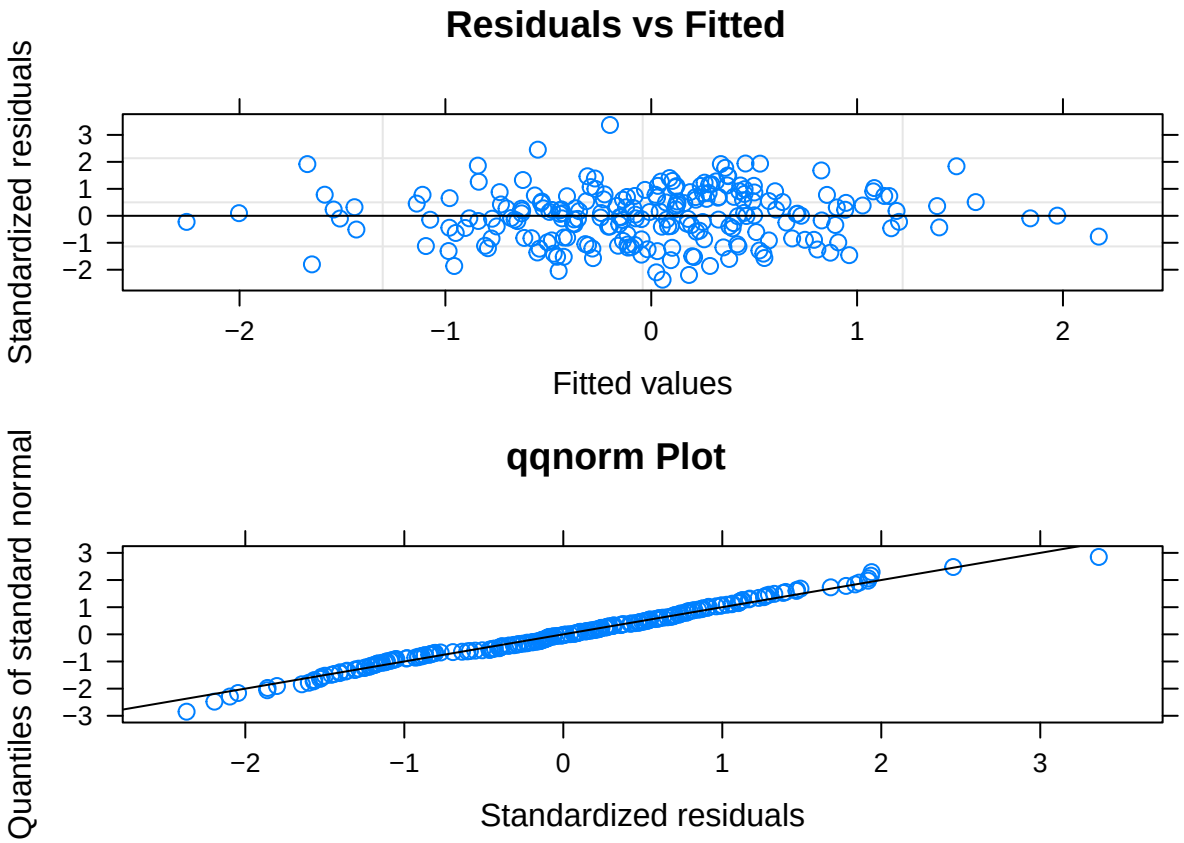
Autocorrelation:

```
vif_For <- as.data.frame(vif(mbx6.For))
vif_For_test <- max(vif_For$"vif(mbx6.For)") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Diagnostic plots:

```
ggarrange(plot(mbx6.For, main = "Residuals vs Fitted"),
          qqnorm(mbx6.For, abline = c(0,1),
                 main = "qqnorm Plot"), nrow = 2)
```



Model assumptions are met.

Anova output for all forest models (step-wise):

```
summary(mbx6.For)
```

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: AfSIS_landcover_normal %>% filter(LandCover == "Forest")
##        AIC      BIC    logLik
##   532.3299 570.0527 -255.1649
##
## Random effects:
##  Formula: ~1 | Site
##         (Intercept)
## StdDev:   0.4733846
##
##  Formula: ~1 | Cluster %in% Site
##         (Intercept)
## StdDev:   0.2334054
##
##  Formula: ~1 | Profile %in% Cluster %in% Site
##          (Intercept) Residual
## StdDev: 5.099371e-05 0.698566
```

```
##
## Fixed effects: CORG ~ Clay_8um + pH + CIA + Mox + Caex + pH:Mox
##                 Value  Std.Error  DF   t-value p-value
## (Intercept) -0.4950451 0.25361661 119 -1.951943  0.0533
## Clay_8um    -0.0995654 0.05744259  69 -1.733303  0.0875
## pH          -0.5779681 0.10104788  69 -5.719745  0.0000
## CIA         -0.2010908 0.10535826  69 -1.908638  0.0605
## Mox          0.5870801 0.06683457  69  8.784079  0.0000
## Caex         0.3286455 0.07228088  69  4.546783  0.0000
## pH:Mox      -0.1324029 0.04203640  69 -3.149720  0.0024
##  Correlation:
##          (Intr) Cly_8m pH     CIA    Mox    Caex
## Clay_8um  0.034
## pH       -0.214 -0.031
## CIA       0.345 -0.195  0.388
## Mox       0.042  0.025  0.085 -0.038
## Caex      0.009  0.068 -0.422  0.128 -0.090
## pH:Mox    0.007 -0.160 -0.002 -0.164 -0.426 -0.026
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -2.36941558 -0.78018867  0.02936595  0.69463304  3.36807715
##
## Number of Observations: 228
## Number of Groups:
##                            Site              Cluster %in% Site
##                              14                             34
## Profile %in% Cluster %in% Site
##                             153
```

```
ava_For <- anova(mbx0.For,mbx1.For,mbx2.For,mbx3.For,mbx4.For,mbx5.For,mbx6.For)
ava_For
```

```
##          Model df      AIC      BIC    logLik   Test  L.Ratio p-value
## mbx0.For     1  5 627.9841 645.1308 -308.9921
## mbx1.For     2  6 626.0641 646.6402 -307.0321 1 vs 2  3.92001  0.0477
## mbx2.For     3  7 615.7891 639.7946 -300.8946 2 vs 3 12.27494  0.0005
## mbx3.For     4  8 614.9410 642.3757 -299.4705 3 vs 4  2.84819  0.0915
## mbx4.For     5  9 556.7742 587.6383 -269.3871 4 vs 5 60.16678  <.0001
## mbx5.For     6 10 538.3499 572.6433 -259.1749 5 vs 6 20.42432  <.0001
## mbx6.For     7 11 532.3299 570.0527 -255.1649 6 vs 7  8.02000  0.0046
```

```
r.squaredGLMM(mbx6.For)
```

```
##            R2m       R2c
## [1,] 0.3402344 0.5799942
```

Grassland:

Build models:

```
mbx0.Gras <- lme(CORG ~ 1,
                 random = ~1|Site/Cluster/Profile, method = "ML",
                 data = AfSIS_landcover_normal %>%
                   filter(LandCover == "Grassland"))
mbx1.Gras <- update(mbx0.Gras, ~. + Clay_8um)
mbx2.Gras <- update(mbx1.Gras, ~. + pH)
mbx3.Gras <- update(mbx2.Gras, ~. + CIA)
mbx4.Gras <- update(mbx3.Gras, ~. + Mox)
mbx5.Gras <- update(mbx4.Gras, ~. + Caex)
mbx6.Gras <- update(mbx5.Gras, ~. + pH*Mox)
```

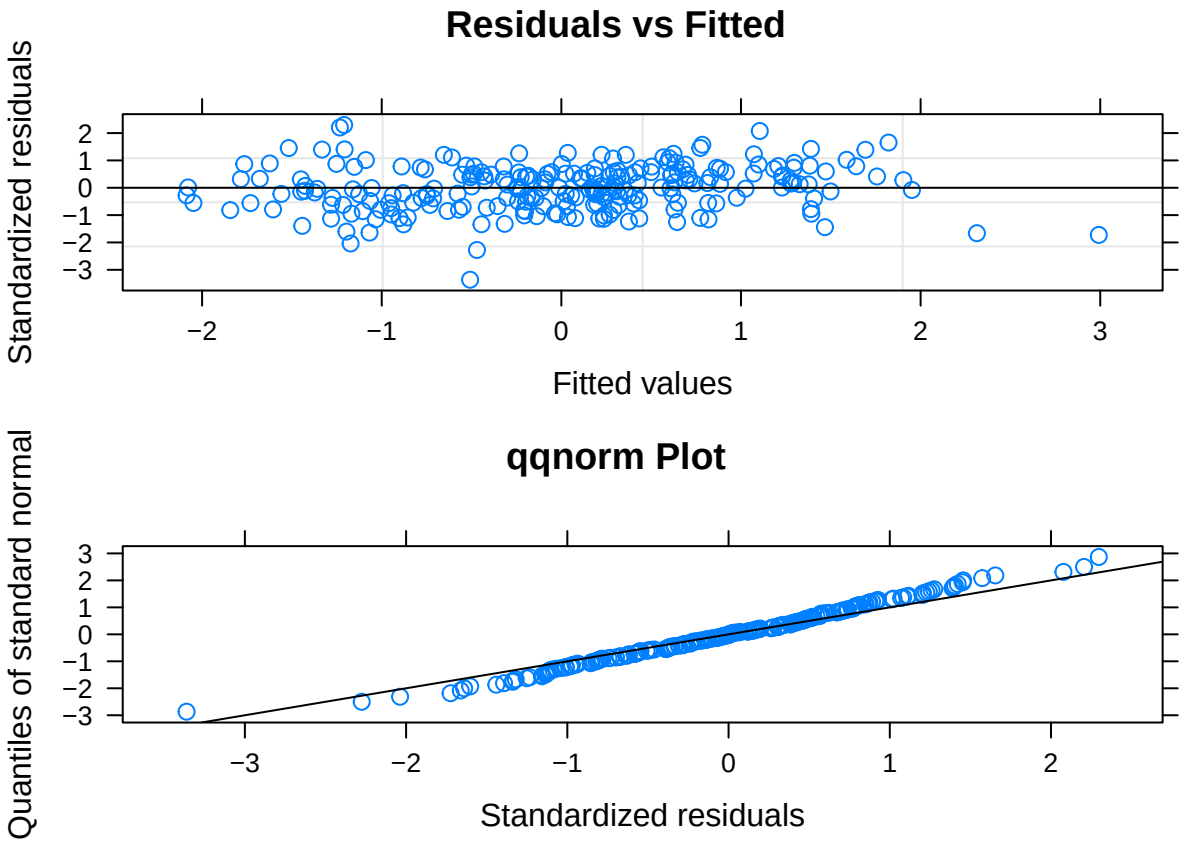
Autocorrelation:

```
vif_Gras <- as.data.frame(vif(mbx6.Gras))
vif_Gras_test <- max(vif_Gras$"vif(mbx6.Gras)") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Diagnostic plots:

```
ggarrange(plot(mbx6.Gras, main = "Residuals vs Fitted"),
          qqnorm(mbx6.Gras, abline = c(0,1),
                 main = "qqnorm Plot"), nrow = 2)
```



Model assumptions are met.

Anova output for all grassland models (step-wise):

```
summary(mbx6.Gras)
```

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: AfSIS_landcover_normal %>% filter(LandCover == "Grassland")
##        AIC      BIC    logLik
##   352.6606 391.0389 -165.3303
##
## Random effects:
##  Formula: ~1 | Site
##         (Intercept)
## StdDev:   0.4777654
##
##  Formula: ~1 | Cluster %in% Site
##         (Intercept)
## StdDev:   0.3017503
##
##  Formula: ~1 | Profile %in% Cluster %in% Site
##          (Intercept)  Residual
## StdDev: 2.171951e-05 0.3467708
```

```
##
## Fixed effects: CORG ~ Clay_8um + pH + CIA + Mox + Caex + pH:Mox
##                 Value  Std.Error DF   t-value p-value
## (Intercept) -0.1000498 0.10242335 97 -0.976826  0.3311
## Clay_8um    -0.0293389 0.04942236 44 -0.593636  0.5558
## pH          -0.5888785 0.06185046 44 -9.521004  0.0000
## CIA         -0.3158882 0.03856010 44 -8.192099  0.0000
## Mox          0.3077677 0.05673453 44  5.424698  0.0000
## Caex         0.6004530 0.07662817 44  7.835930  0.0000
## pH:Mox      -0.2624483 0.04569732 44 -5.743187  0.0000
##  Correlation:
##          (Intr) Cly_8m pH     CIA    Mox    Caex
## Clay_8um -0.011
## pH        0.003  0.109
## CIA      -0.045 -0.286 -0.055
## Mox       0.022 -0.224  0.300 -0.081
## Caex      0.107 -0.456 -0.557  0.235 -0.326
## pH:Mox    0.060 -0.054 -0.036  0.001  0.161  0.034
##
## Standardized Within-Group Residuals:
##          Min           Q1          Med           Q3          Max
## -3.361503793 -0.560058915  0.008157153  0.557501365  2.296283360
##
## Number of Observations: 242
## Number of Groups:
##                            Site              Cluster %in% Site
##                              30                            127
## Profile %in% Cluster %in% Site
##                             192
```

```
ava_Gras <- anova(mbx0.Gras,mbx1.Gras,mbx2.Gras,mbx3.Gras,mbx4.Gras,mbx5.Gras,mbx6.Gras)
ava_Gras
```

```
##           Model df      AIC      BIC    logLik   Test  L.Ratio p-value
## mbx0.Gras     1  5 570.2328 587.6775 -280.1164
## mbx1.Gras     2  6 561.0550 581.9887 -274.5275 1 vs 2 11.17772   8e-04
## mbx2.Gras     3  7 542.4526 566.8752 -264.2263 2 vs 3 20.60241  <.0001
## mbx3.Gras     4  8 484.6580 512.5695 -234.3290 3 vs 4 59.79467  <.0001
## mbx4.Gras     5  9 430.9453 462.3458 -206.4727 4 vs 5 55.71263  <.0001
## mbx5.Gras     6 10 381.4943 416.3837 -180.7471 5 vs 6 51.45106  <.0001
## mbx6.Gras     7 11 352.6606 391.0389 -165.3303 6 vs 7 30.83372  <.0001
```

```
r.squaredGLMM(mbx6.Gras)
```

```
##           R2m      R2c
## [1,] 0.566461 0.881398
```

Other:

Build models:

```
mbx0.Oth <- lme(CORG ~ 1,
                random = ~1|Site/Cluster/Profile, method = "ML",
                data = AfSIS_landcover_normal %>%
                  filter(LandCover == "Other"))
mbx1.Oth <- update(mbx0.Oth, ~. + Clay_8um)
mbx2.Oth <- update(mbx1.Oth, ~. + pH)
mbx3.Oth <- update(mbx2.Oth, ~. + CIA)
mbx4.Oth <- update(mbx3.Oth, ~. + Mox)
mbx5.Oth <- update(mbx4.Oth, ~. + Caex)
mbx6.Oth <- update(mbx5.Oth, ~. + pH*Mox)
```

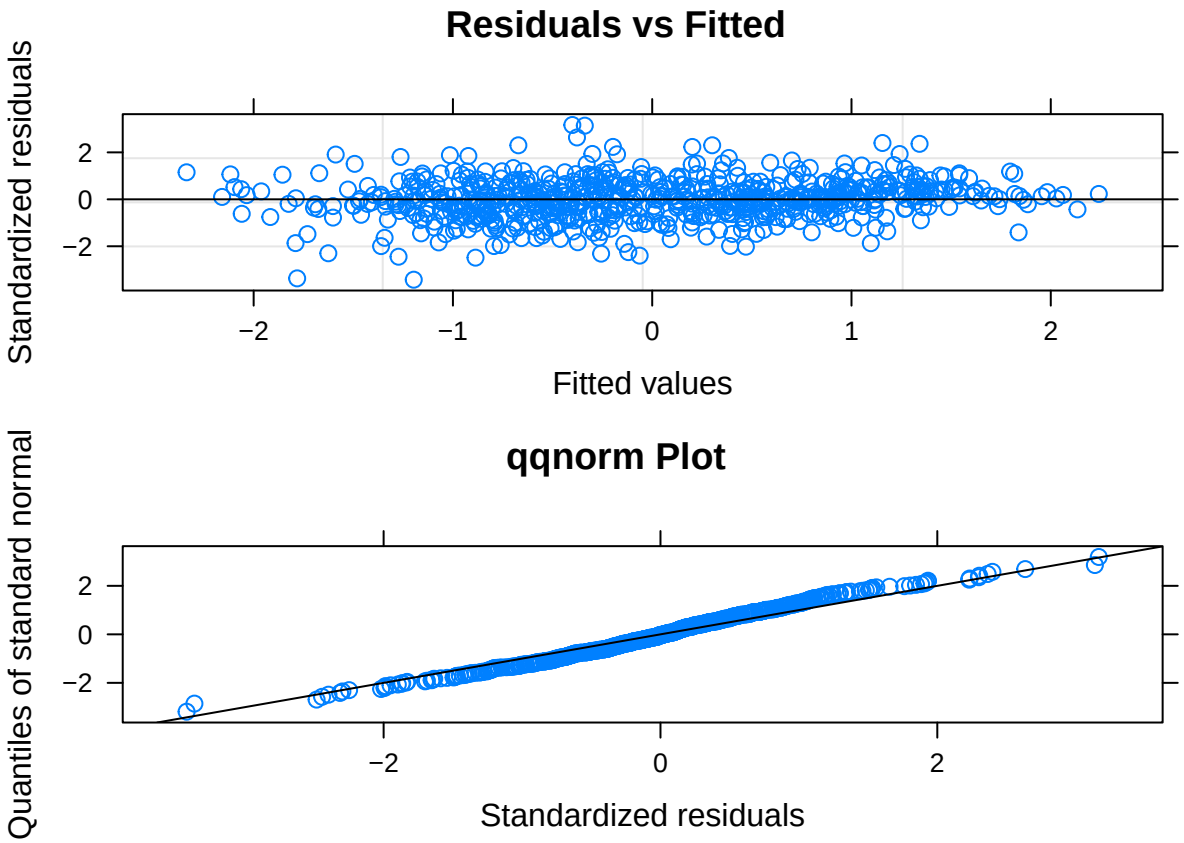
Autocorrelation:

```
vif_Oth <- as.data.frame(vif(mbx6.Oth))
vif_Oth_test <- max(vif_Oth$"vif(mbx6.Oth)") < 3.0
```

VIF < 3.0: TRUE -> no autocorrelation

Diagnostic plots:

```
ggarrange(plot(mbx6.Oth, main = "Residuals vs Fitted"),
          qqnorm(mbx6.Oth, abline = c(0,1),
                 main = "qqnorm Plot"), nrow = 2)
```



Model assumptions are met.

Anova output for all other models (step-wise):

```
summary(mbx6.Oth)
```

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: AfSIS_landcover_normal %>% filter(LandCover == "Other")
##        AIC      BIC    logLik
##   1011.661 1061.754 -494.8305
##
## Random effects:
##  Formula: ~1 | Site
##         (Intercept)
## StdDev:   0.5366225
##
##  Formula: ~1 | Cluster %in% Site
##         (Intercept)
## StdDev:   0.2049804
##
##  Formula: ~1 | Profile %in% Cluster %in% Site
##         (Intercept)  Residual
## StdDev:   0.1905183 0.3808144
```

```
##
## Fixed effects: CORG ~ Clay_8um + pH + CIA + Mox + Caex + pH:Mox
##                  Value  Std.Error  DF   t-value p-value
## (Intercept)  0.0272261 0.09717063 296  0.280189  0.7795
## Clay_8um    -0.1008798 0.03147209 129 -3.205373  0.0017
## pH          -0.3332253 0.05133679 129 -6.490966  0.0000
## CIA         -0.1300941 0.03312490 129 -3.927381  0.0001
## Mox          0.2960367 0.03609678 129  8.201195  0.0000
## Caex         0.5627922 0.04555745 129 12.353462  0.0000
## pH:Mox      -0.0837357 0.02544215 129 -3.291220  0.0013
##  Correlation:
##          (Intr) Cly_8m pH     CIA    Mox    Caex
## Clay_8um  0.046
## pH        0.150  0.055
## CIA      -0.049 -0.310  0.225
## Mox       0.001 -0.308  0.242 -0.068
## Caex      0.057 -0.315 -0.499  0.125 -0.348
## pH:Mox    0.049  0.034  0.331  0.106  0.200 -0.301
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -3.42347192 -0.47120690  0.02301884  0.49948350  3.16655136
##
## Number of Observations: 702
## Number of Groups:
##                            Site              Cluster %in% Site
##                              38                            271
## Profile %in% Cluster %in% Site
##                             567
```

```
ava_Oth <- anova(mbx0.Oth,mbx1.Oth,mbx2.Oth,mbx3.Oth,mbx4.Oth,mbx5.Oth,mbx6.Oth)
ava_Oth
```

```
##          Model df      AIC      BIC    logLik   Test   L.Ratio p-value
## mbx0.Oth     1  5 1313.240 1336.009 -651.6198
## mbx1.Oth     2  6 1291.221 1318.544 -639.6104 1 vs 2  24.01876  <.0001
## mbx2.Oth     3  7 1293.101 1324.979 -639.5506 2 vs 3   0.11963  0.7294
## mbx3.Oth     4  8 1277.315 1313.746 -630.6575 3 vs 4  17.78624  <.0001
## mbx4.Oth     5  9 1146.616 1187.601 -564.3078 4 vs 5 132.69931  <.0001
## mbx5.Oth     6 10 1020.268 1065.807 -500.1341 5 vs 6 128.34746  <.0001
## mbx6.Oth     7 11 1011.661 1061.754 -494.8305 6 vs 7  10.60726  0.0011
```

```
r.squaredGLMM(mbx6.Oth)
```

```
##            R2m       R2c
## [1,] 0.3727155 0.8220829
```

Anova Summary table:
Table B9 in the supplement of the manuscript.

```
ava_LC_gt <- rbind(ava_Crop, ava_For, ava_Gras, ava_Oth) %>%
  dplyr::select(-call) %>%
  gt(rowname_col = "Model") %>%
  tab_row_group(group = "Cropland", rows = 1:7) %>%
  tab_row_group(group = "Forest", rows = 8:14) %>%
  tab_row_group(group = "Grassland", rows = 15:21) %>%
  tab_row_group(group = "Other", rows = 22:28) %>%
  row_group_order(
    groups = c("Cropland",
               "Forest",
               "Grassland",
               "Other")) %>%
  fmt_number(columns = c(3:5,7), decimals = 2) %>%
  fmt_number(columns = 8, decimals = 4) %>%
  cols_align(align = "center", columns = c(2:8))

gtsave(ava_LC_gt, filename = "AfSIS_RefData2_AnovaLC.rtf")
```

6.2.5 Plotting

This section contains the code to reproduce *Figure 4*.

```
Clay <- r.squaredGLMM(mbx1.TOP)[1,1]*100
pH <- (r.squaredGLMM(mbx2.TOP)[1,1] - r.squaredGLMM(mbx1.TOP)[1,1])*100
CIA <- (r.squaredGLMM(mbx3.TOP)[1,1] - r.squaredGLMM(mbx2.TOP)[1,1])*100
Mox <- (r.squaredGLMM(mbx4.TOP)[1,1] - r.squaredGLMM(mbx3.TOP)[1,1])*100
Caex <- (r.squaredGLMM(mbx5.TOP)[1,1] - r.squaredGLMM(mbx4.TOP)[1,1])*100
pH_Mox <- (r.squaredGLMM(mbx6.TOP)[1,1] - r.squaredGLMM(mbx5.TOP)[1,1])*100

R2m.TOP <- tibble(Clay, pH, CIA, Mox, Caex, pH_Mox) %>%
  pivot_longer(everything(), names_to = "Predictor", values_to = "Percentage") %>%
  mutate(Group = "Topsoil") %>%
  mutate(Percentage = replace(Percentage, which(Percentage < 0), 0))

Clay <- r.squaredGLMM(mbx1.BOT)[1,1]*100
pH <- (r.squaredGLMM(mbx2.BOT)[1,1] - r.squaredGLMM(mbx1.BOT)[1,1])*100
CIA <- (r.squaredGLMM(mbx3.BOT)[1,1] - r.squaredGLMM(mbx2.BOT)[1,1])*100
Mox <- (r.squaredGLMM(mbx4.BOT)[1,1] - r.squaredGLMM(mbx3.BOT)[1,1])*100
Caex <- (r.squaredGLMM(mbx5.BOT)[1,1] - r.squaredGLMM(mbx4.BOT)[1,1])*100
pH_Mox <- (r.squaredGLMM(mbx6.BOT)[1,1] - r.squaredGLMM(mbx5.BOT)[1,1])*100

R2m.BOT <- tibble(Clay, pH, CIA, Mox, Caex, pH_Mox) %>%
  pivot_longer(everything(), names_to = "Predictor", values_to = "Percentage") %>%
```

```
  mutate(Group = "Subsoil") %>%
  mutate(Percentage = replace(Percentage, which(Percentage < 0), 0))

R2m.Depth <- rbind(R2m.TOP, R2m.BOT)

R2m.Depth$Group <- factor(R2m.Depth$Group, levels = c("Topsoil", "Subsoil"))
R2m.Depth$Predictor <- factor(R2m.Depth$Predictor,
                              levels = c("Clay", "pH", "CIA", "Mox", "Caex",
                                         "pH_Mox"))

### pH
pH <- 0
pH_Mox <- 0
Clay <- r.squaredGLMM(mbx1.sacid)[1,1]*100
CIA <- (r.squaredGLMM(mbx2.sacid)[1,1] - r.squaredGLMM(mbx1.sacid)[1,1])*100
Mox <- (r.squaredGLMM(mbx3.sacid)[1,1] - r.squaredGLMM(mbx2.sacid)[1,1])*100
Caex <- (r.squaredGLMM(mbx4.sacid)[1,1] - r.squaredGLMM(mbx3.sacid)[1,1])*100

R2m.sacid <- tibble(Clay, pH, CIA, Mox, Caex, pH_Mox) %>%
  pivot_longer(everything(), names_to = "Predictor", values_to = "Percentage") %>%
  mutate(Group = "strongly acidic") %>%
  mutate(Percentage = replace(Percentage, which(Percentage < 0), 0))

Clay <- r.squaredGLMM(mbx1.macid)[1,1]*100
CIA <- (r.squaredGLMM(mbx2.macid)[1,1] - r.squaredGLMM(mbx1.macid)[1,1])*100
Mox <- (r.squaredGLMM(mbx3.macid)[1,1] - r.squaredGLMM(mbx2.macid)[1,1])*100
Caex <- (r.squaredGLMM(mbx4.macid)[1,1] - r.squaredGLMM(mbx3.macid)[1,1])*100

R2m.macid <- tibble(Clay, pH, CIA, Mox, Caex, pH_Mox) %>%
  pivot_longer(everything(), names_to = "Predictor", values_to = "Percentage") %>%
  mutate(Group = "moderately acidic") %>%
  mutate(Percentage = replace(Percentage, which(Percentage < 0), 0))

Clay <- r.squaredGLMM(mbx1.neut)[1,1]*100
CIA <- (r.squaredGLMM(mbx2.neut)[1,1] - r.squaredGLMM(mbx1.neut)[1,1])*100
Mox <- (r.squaredGLMM(mbx3.neut)[1,1] - r.squaredGLMM(mbx2.neut)[1,1])*100
Caex <- (r.squaredGLMM(mbx4.neut)[1,1] - r.squaredGLMM(mbx3.neut)[1,1])*100

R2m.neut <- tibble(Clay, pH, CIA, Mox, Caex, pH_Mox) %>%
  pivot_longer(everything(), names_to = "Predictor", values_to = "Percentage") %>%
  mutate(Group = "neutral") %>%
  mutate(Percentage = replace(Percentage, which(Percentage < 0), 0))

Clay <- r.squaredGLMM(mbx1.alk)[1,1]*100
CIA <- (r.squaredGLMM(mbx2.alk)[1,1] - r.squaredGLMM(mbx1.alk)[1,1])*100
Mox <- (r.squaredGLMM(mbx3.alk)[1,1] - r.squaredGLMM(mbx2.alk)[1,1])*100
Caex <- (r.squaredGLMM(mbx4.alk)[1,1] - r.squaredGLMM(mbx3.alk)[1,1])*100
```

```
R2m.alk <- tibble(Clay, pH, CIA, Mox, Caex, pH_Mox) %>%
  pivot_longer(everything(), names_to = "Predictor", values_to = "Percentage") %>%
  mutate(Group = "alkaline") %>%
  mutate(Percentage = replace(Percentage, which(Percentage < 0), 0))

R2m.pH <- rbind(R2m.sacid, R2m.macid, R2m.neut, R2m.alk)

R2m.pH$Group <- factor(R2m.pH$Group,
                       levels = c("strongly acidic", "moderately acidic",
                                  "neutral", "alkaline"))
R2m.pH$Predictor <- factor(R2m.pH$Predictor,
                           levels = c("Depth","LandCover",
                                      "Clay", "pH", "CIA", "Mox", "Caex", "pH_Mox"))

### CIA
CIA <- 0
Clay <- r.squaredGLMM(mbx1.CIAmod)[1,1]*100
pH <- (r.squaredGLMM(mbx2.CIAmod)[1,1] - r.squaredGLMM(mbx1.CIAmod)[1,1])*100
Mox <- (r.squaredGLMM(mbx3.CIAmod)[1,1] - r.squaredGLMM(mbx2.CIAmod)[1,1])*100
Caex <- (r.squaredGLMM(mbx4.CIAmod)[1,1] - r.squaredGLMM(mbx3.CIAmod)[1,1])*100
pH_Mox <- (r.squaredGLMM(mbx5.CIAmod)[1,1] - r.squaredGLMM(mbx4.CIAmod)[1,1])*100

R2m.CIAmod <- tibble(Clay, pH, CIA, Mox, Caex, pH_Mox) %>%
  pivot_longer(everything(), names_to = "Predictor", values_to = "Percentage") %>%
  mutate(Group = "moderate") %>%
  mutate(Percentage = replace(Percentage, which(Percentage < 0), 0))

Clay <- r.squaredGLMM(mbx1.CIAhigh)[1,1]*100
pH <- (r.squaredGLMM(mbx2.CIAhigh)[1,1] - r.squaredGLMM(mbx1.CIAhigh)[1,1])*100
Mox <- (r.squaredGLMM(mbx3.CIAhigh)[1,1] - r.squaredGLMM(mbx2.CIAhigh)[1,1])*100
Caex <- (r.squaredGLMM(mbx4.CIAhigh)[1,1] - r.squaredGLMM(mbx3.CIAhigh)[1,1])*100
pH_Mox <- (r.squaredGLMM(mbx5.CIAhigh)[1,1] - r.squaredGLMM(mbx4.CIAhigh)[1,1])*100

R2m.CIAhigh <- tibble(Clay, pH, CIA, Mox, Caex, pH_Mox) %>%
  pivot_longer(everything(), names_to = "Predictor", values_to = "Percentage") %>%
  mutate(Group = "high") %>%
  mutate(Percentage = replace(Percentage, which(Percentage < 0), 0))

R2m.CIA <- rbind(R2m.CIAmod, R2m.CIAhigh)

R2m.CIA$Group <- factor(R2m.CIA$Group,
                        levels = c("moderate", "high"))
R2m.CIA$Predictor <- factor(R2m.CIA$Predictor,
                            levels = c("Clay", "pH", "CIA", "Mox", "Caex",
                                       "pH_Mox"))

### Number of wet month
```

```
Clay <- r.squaredGLMM(mbx1.0wet)[1,1]*100
pH <- (r.squaredGLMM(mbx2.0wet)[1,1] - r.squaredGLMM(mbx1.0wet)[1,1])*100
CIA <- (r.squaredGLMM(mbx3.0wet)[1,1] - r.squaredGLMM(mbx2.0wet)[1,1])*100
Mox <- (r.squaredGLMM(mbx4.0wet)[1,1] - r.squaredGLMM(mbx3.0wet)[1,1])*100
Caex <- (r.squaredGLMM(mbx5.0wet)[1,1] - r.squaredGLMM(mbx4.0wet)[1,1])*100
pH_Mox <- (r.squaredGLMM(mbx6.0wet)[1,1] - r.squaredGLMM(mbx5.0wet)[1,1])*100

R2m.0wt <- tibble(Clay, pH, CIA, Mox, Caex, pH_Mox) %>%
  pivot_longer(everything(), names_to = "Predictor", values_to = "Percentage") %>%
  mutate(Group = "0") %>%
  mutate(Percentage = replace(Percentage, which(Percentage < 0), 0))

Clay <- r.squaredGLMM(mbx1.1_3wet)[1,1]*100
pH <- (r.squaredGLMM(mbx2.1_3wet)[1,1] - r.squaredGLMM(mbx1.1_3wet)[1,1])*100
CIA <- (r.squaredGLMM(mbx3.1_3wet)[1,1] - r.squaredGLMM(mbx2.1_3wet)[1,1])*100
Mox <- (r.squaredGLMM(mbx4.1_3wet)[1,1] - r.squaredGLMM(mbx3.1_3wet)[1,1])*100
Caex <- (r.squaredGLMM(mbx5.1_3wet)[1,1] - r.squaredGLMM(mbx4.1_3wet)[1,1])*100
pH_Mox <- (r.squaredGLMM(mbx6.1_3wet)[1,1] - r.squaredGLMM(mbx5.1_3wet)[1,1])*100

R2m.1_3wt <- tibble(Clay, pH, CIA, Mox, Caex, pH_Mox) %>%
  pivot_longer(everything(), names_to = "Predictor", values_to = "Percentage") %>%
  mutate(Group = "1-3") %>%
  mutate(Percentage = replace(Percentage, which(Percentage < 0), 0))

Clay <- r.squaredGLMM(mbx1.4_7wet)[1,1]*100
pH <- (r.squaredGLMM(mbx2.4_7wet)[1,1] - r.squaredGLMM(mbx1.4_7wet)[1,1])*100
CIA <- (r.squaredGLMM(mbx3.4_7wet)[1,1] - r.squaredGLMM(mbx2.4_7wet)[1,1])*100
Mox <- (r.squaredGLMM(mbx4.4_7wet)[1,1] - r.squaredGLMM(mbx3.4_7wet)[1,1])*100
Caex <- (r.squaredGLMM(mbx5.4_7wet)[1,1] - r.squaredGLMM(mbx4.4_7wet)[1,1])*100
pH_Mox <- (r.squaredGLMM(mbx6.4_7wet)[1,1] - r.squaredGLMM(mbx5.4_7wet)[1,1])*100

R2m.4_7wt <- tibble(Clay, pH, CIA, Mox, Caex, pH_Mox) %>%
  pivot_longer(everything(), names_to = "Predictor", values_to = "Percentage") %>%
  mutate(Group = "4-7") %>%
  mutate(Percentage = replace(Percentage, which(Percentage < 0), 0))

R2m.nWet <- rbind(R2m.0wt, R2m.1_3wt, R2m.4_7wt)

R2m.nWet$Group <- factor(R2m.nWet$Group,
                         levels = c("0", "1-3", "4-7"))
R2m.nWet$Predictor <- factor(R2m.nWet$Predictor,
                             levels = c("MAT", "AridityIndex", "Depth","LandCover",
                                        "Clay", "pH", "CIA","Mox", "Caex",
                                        "pH_Mox"))

### LandCover
Clay <- r.squaredGLMM(mbx1.Crop)[1,1]*100
```

```
pH <- (r.squaredGLMM(mbx2.Crop)[1,1] - r.squaredGLMM(mbx1.Crop)[1,1])*100
CIA <- (r.squaredGLMM(mbx3.Crop)[1,1] - r.squaredGLMM(mbx2.Crop)[1,1])*100
Mox <- (r.squaredGLMM(mbx4.Crop)[1,1] - r.squaredGLMM(mbx3.Crop)[1,1])*100
Caex <- (r.squaredGLMM(mbx5.Crop)[1,1] - r.squaredGLMM(mbx4.Crop)[1,1])*100
pH_Mox <- (r.squaredGLMM(mbx6.Crop)[1,1] - r.squaredGLMM(mbx5.Crop)[1,1])*100

R2m.Crop <- tibble(Clay, pH, CIA, Mox, Caex, pH_Mox) %>%
  pivot_longer(everything(), names_to = "Predictor", values_to = "Percentage") %>%
  mutate(Group = "Cropland") %>%
  mutate(Percentage = replace(Percentage, which(Percentage < 0), 0))

Clay <- r.squaredGLMM(mbx1.For)[1,1]*100
pH <- (r.squaredGLMM(mbx2.For)[1,1] - r.squaredGLMM(mbx1.For)[1,1])*100
CIA <- (r.squaredGLMM(mbx3.For)[1,1] - r.squaredGLMM(mbx2.For)[1,1])*100
Mox <- (r.squaredGLMM(mbx4.For)[1,1] - r.squaredGLMM(mbx3.For)[1,1])*100
Caex <- (r.squaredGLMM(mbx5.For)[1,1] - r.squaredGLMM(mbx4.For)[1,1])*100
pH_Mox <- (r.squaredGLMM(mbx6.For)[1,1] - r.squaredGLMM(mbx5.For)[1,1])*100

R2m.For <- tibble(Clay, pH, CIA, Mox, Caex, pH_Mox) %>%
  pivot_longer(everything(), names_to = "Predictor", values_to = "Percentage") %>%
  mutate(Group = "Forest") %>%
  mutate(Percentage = replace(Percentage, which(Percentage < 0), 0))

Clay <- r.squaredGLMM(mbx1.Gras)[1,1]*100
pH <- (r.squaredGLMM(mbx2.Gras)[1,1] - r.squaredGLMM(mbx1.Gras)[1,1])*100
CIA <- (r.squaredGLMM(mbx3.Gras)[1,1] - r.squaredGLMM(mbx2.Gras)[1,1])*100
Mox <- (r.squaredGLMM(mbx4.Gras)[1,1] - r.squaredGLMM(mbx3.Gras)[1,1])*100
Caex <- (r.squaredGLMM(mbx5.Gras)[1,1] - r.squaredGLMM(mbx4.Gras)[1,1])*100
pH_Mox <- (r.squaredGLMM(mbx6.Gras)[1,1] - r.squaredGLMM(mbx5.Gras)[1,1])*100

R2m.Gras <- tibble(Clay, pH, CIA, Mox, Caex, pH_Mox) %>%
  pivot_longer(everything(), names_to = "Predictor", values_to = "Percentage") %>%
  mutate(Group = "Grassland") %>%
  mutate(Percentage = replace(Percentage, which(Percentage < 0), 0))

Clay <- r.squaredGLMM(mbx1.Oth)[1,1]*100
pH <- (r.squaredGLMM(mbx2.Oth)[1,1] - r.squaredGLMM(mbx1.Oth)[1,1])*100
CIA <- (r.squaredGLMM(mbx3.Oth)[1,1] - r.squaredGLMM(mbx2.Oth)[1,1])*100
Mox <- (r.squaredGLMM(mbx4.Oth)[1,1] - r.squaredGLMM(mbx3.Oth)[1,1])*100
Caex <- (r.squaredGLMM(mbx5.Oth)[1,1] - r.squaredGLMM(mbx4.Oth)[1,1])*100
pH_Mox <- (r.squaredGLMM(mbx6.Oth)[1,1] - r.squaredGLMM(mbx5.Oth)[1,1])*100

R2m.Oth <- tibble(Clay, pH, CIA, Mox, Caex, pH_Mox) %>%
  pivot_longer(everything(), names_to = "Predictor", values_to = "Percentage") %>%
  mutate(Group = "Other") %>%
  mutate(Percentage = replace(Percentage, which(Percentage < 0), 0))
```

```
R2m.LC <- rbind(R2m.Crop, R2m.For, R2m.Gras, R2m.Oth)

R2m.LC$Group <- factor(R2m.LC$Group,
                       levels = c("Cropland", "Forest", "Grassland", "Other"))
R2m.LC$Predictor <- factor(R2m.LC$Predictor,
                           levels = c("Clay", "pH", "CIA","Mox", "Caex",
                                      "pH_Mox"))
```

```
###Stack bars for each sub-model
##Use funtion to create plots
GeoChemColor <- brewer.pal(6, "YlOrBr")

plotStack_fun = function(df){
  df %>%
    ggplot(aes(y = reorder(Group, desc(Group)),
               x = Percentage, fill = Predictor)) +
    geom_bar(stat = "identity", color = "black",
             position = position_stack(reverse = TRUE)) +
    theme_bw(base_size = 14) +
    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(),
          panel.grid.major = element_blank(),
          legend.position = "top",
          legend.text.align = 0) +
    scale_y_discrete("", expand = c(0,0)) +
    scale_x_continuous("", expand = c(0,0), limits = c(0,70),
                       breaks = seq(0,70,10)) +
    scale_fill_manual(values = GeoChemColor,
                      labels = c("Clay + fine silt",
                                 expression("pH"[H2O]), "CIA",
                                 expression("M"[ox]),
                                 expression("Ca"[ex]),
                                 expression(paste("pH"[H2O]," * M"[ox])))) +
    guides(fill = guide_legend(nrow = 1))
}

R2m.Depth.stack <- plotStack_fun(R2m.Depth) +
  theme(axis.text.x = element_blank())
R2m.pH.stack <- plotStack_fun(R2m.pH) +
  theme(axis.text.x = element_blank())
R2m.CIA.stack <- plotStack_fun(R2m.CIA) +
  theme(axis.text.x = element_blank())
R2m.nWet.stack <- plotStack_fun(R2m.nWet) +
  theme(axis.text.x = element_blank())
```

```
R2m.LC.stack <- plotStack_fun(R2m.LC) +
  scale_x_continuous("Explained variation [%]",
                     expand = c(0,0), limits = c(0,70),
                     breaks = seq(0,70,10))

ggarrange(R2m.Depth.stack, R2m.pH.stack, R2m.nWet.stack,
          R2m.CIA.stack, R2m.LC.stack, ncol = 1,
          common.legend = TRUE, legend = "top",
          align = "v", heights = c(2.2/15,3.9/15,3/15,2.2/15,4/15),
          labels = c("a) Depth", "b) pH", "c) Wetness", "d) Weathering", "e) Land cover"),
          vjust = c(0.7,0.9,0.9,0.3,0.5),
          hjust = c(-0.3,-0.5,-0.2,-0.15,-0.15),
          font.label = list(size = 12))

ggsave("AfSIS_RefData2_R2m_allStacked.jpeg", height = 7, width = 10)
```

6.2.6 Summary table

This section contains the code to reproduce *Table B2*.

```
R2m.table_subM <- rbind(R2m.Depth, R2m.pH, R2m.nWet,
      R2m.CIA, R2m.LC) %>%
  mutate(R2m = Percentage/100) %>%
  dplyr::select(Predictor, Group, R2m) %>%
  pivot_wider(names_from = Predictor, values_from = R2m) %>%
  gt(rowname_col = "Group") %>%
  tab_row_group(group = "Depth", rows = 1:2) %>%
  tab_row_group(group = "pH classes", rows = 3:6) %>%
  tab_row_group(group = "Number of wet months", rows = 7:9) %>%
  tab_row_group(group = "Weathering", rows = 10:11) %>%
  tab_row_group(group = "Land cover", rows = 12:15) %>%
  row_group_order(
    groups = c("Depth", "pH classes",
               "Number of wet months", "Weathering",
               "Land cover")) %>%
  fmt_number(columns = c(2:7), decimals = 2) %>%
  cols_align(align = "center", columns = c(2:7)) %>%
  cols_label(Clay = "Clay + fine silt",
             pH_Mox = "pH*Mox") %>%
  fmt_missing(columns = c(2:7), missing_text = "-")

gtsave(R2m.table_subM, filename = "AfSIS_RefData2_R2m_subModels.rtf")
```

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 A3 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_Clay_20 <- AfSIS_TOP %>%
  dplyr::select(Clay_20um, 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_Clay_20 <- AfSIS_BOT %>%
  dplyr::select(Clay_20um, 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_Clay20_RF <- mlr::makeRegrTask(data = data_TOP_Clay_20, 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_Clay20_RF <- mlr::makeRegrTask(data = data_BOT_Clay_20, 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_TOP_3 <- ParamHelpers::makeParamSet(
  makeIntegerParam("mtry", lower = 1, upper = ncol(data_TOP_Clay_20) - 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))
ps_BOT_3 <- ParamHelpers::makeParamSet(
  makeIntegerParam("mtry", lower = 1, upper = ncol(data_BOT_Clay_20) - 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 (< 8um):

```
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):       2.594557
## R squared (OOB):                  0.1154444
```

Clay-only model (< 20um):

```
set.seed(42)
tune_TOP_Clay20 <- mlr::tuneParams(learner = lrn_rf,
```

```
                                    task = task_TOP_Clay20_RF,
                                    resampling = tune_level,
                                    par.set = ps_TOP_3,
                                    control = ctrl,
                                    measures = mlr::rmse)
```

```
# Create learner with best hyperparameter combination
set.seed(42)
lrn_rf_TOP_Clay20 <- mlr::setHyperPars(makeLearner("regr.ranger", predict.type = "response"),
                                       par.vals = tune_TOP_Clay20$x)

# Train model
set.seed(42)
model_rf_TOP_Clay20 <- mlr::train(lrn_rf_TOP_Clay20, task_TOP_Clay20_RF)

# retrieve the ranger output
mlr::getLearnerModel(model_rf_TOP_Clay20)
```

```
## 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):       2.438074
## R squared (OOB):                  0.1687936
```

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):       2.275918
## R squared (OOB):                  0.2240772
```

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 (< 8um):

```
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):         1.097358
## R squared (OOB):                    0.1227935
```

Clay-only model (< 8um):

```
set.seed(42)
tune_BOT_Clay20 <- mlr::tuneParams(learner = lrn_rf,
                                   task = task_BOT_Clay20_RF,
                                   resampling = tune_level,
                                   par.set = ps_BOT_3,
                                   control = ctrl,
                                   measures = mlr::rmse)
```

```
# learning using the best hyperparameter combination
set.seed(42)
lrn_rf_BOT_Clay20 <- mlr::setHyperPars(makeLearner("regr.ranger", predict.type = "response"),
                                       par.vals = tune_BOT_Clay20$x)

# train model
set.seed(42)
model_rf_BOT_Clay20 <- mlr::train(lrn_rf_BOT_Clay20, task_BOT_Clay20_RF)

# retrieve the ranger output
mlr::getLearnerModel(model_rf_BOT_Clay20)
```

```
## 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):       1.015081
## R squared (OOB):                  0.1885645
```

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 6 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 = 3.7,
           label = expression(paste("Ca"[ex]," [cmol"^"+"," kg"^"-1","]")), 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")) +
  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)
```