Umweltstatistik Bericht - R

Milena Mensching und Justus Weyers

2023-10-08

Contents

Technisches

Pakete laden

```
# Das Tidyverse
library(tidyverse)

# Plots
library(GGally)
library(corrplot)
library(patchwork)
library(wesanderson)

# Statistik
library(Metrics)
library(rstatix)

# Geo-Pakete
library(sf)
library(tmap)

# Lineare Modelle
library(glmulti)
library(car)
library(visreg)

# Baumbasierte Modelle
library(tree)
library(rpart)
library(rpart.plot)
library(randomForest)
library(quantregForest)

# LaTex
library(xtable)
```

Einstellungen

```
# Seed festlegen
set.seed(1)
# Interaktive Kartenansicht
```

```r
tmap::tmap_mode("view")
# Farbauswahl - https://github.com/karthik/wesanderson
colouring = c("#D8B70A", "#05612a", "#A2A475") # wes_palette("Cavalcanti1", n = 3)
```

Funktionsdefinitionen

```r
# Funktion, um Modellvorhersage und Fehler an
# den zentralen Dataframe 'data' anzubinden
pred = function(name, model, predictors = NA, data, response,
                predictions = NULL) {
  # Vorhersagespalte anlegen
  data[,name] = rep(NA, nrow(data))
  # Fehlerspalte anlegen
  data[,paste0("err", name)] = rep(NA, nrow(data))

  if (!is.null(predictions)) {
    data[as.double(names(predictions)),name] = predictions
  } else {
    # Vorhersagewerte bestimmen
    predictions = predict(model, newdata = predictors)

    # Vorhersage anbinden
    data[rownames(predictors),name] = predictions
  }

  # Eigentliche Werte
  actual = data[!is.na(data[,name]), response]
  # Fehler anbinden
  data[!is.na(data[,name]),paste0("err", name)] = sqrt((predictions-actual)**2)

  return(data)
}

# Funktion zur Berechnung von Gesamtvarianz (SST),
# nicht-erklärter Varianz (SSE) und R²
var = function(yobs, ymod) {
  SST = sum((yobs - mean(yobs))^2)
  SSE = sum((yobs - ymod)^2)
  R2 = 1 - SSE / SST
  vec = setNames(c(SST, SSE, R2), c("SST", "SSE", "R2"))
  return(vec)
}

# Funktion, um aus einem Dataframe ein
# simple feature zu erstellen
sf = function(df, epsg = 25833, coords = c("x_UTM", "y_UTM")) {
  return(st_as_sf(df, coords = coords, crs = st_crs(epsg)))
}

# Funktion um die Wichtigkeit von Prädiktoren
# in einem linearen Modell 'x' zu erhalten
# Code: {glmulti} Vincent Calcagno, McGill University
# (modifiziert)
```

```
getImportance = function(x) {
  # plots variable (i.e. terms) importances
    ww = exp(-(x@crits - x@crits[1]) / 2)
    ww = ww / sum(ww)
    # handle synonymies for interactions
    # this translates to unique notations (e.g. x:y and y:x are the same)
    clartou = function(x) {
        sort(strsplit(x, ":")[[1]]) -> pieces
        if (length(pieces) > 1) paste(pieces[1], ":", pieces[2], sep="")
        else x
        }
    # list terms in models
    tet = lapply(x@formulas, function(x) sapply(attr(delete.response(terms(x)),
                                                     "term.labels"), clartou))
    # all unique terms
    unique(unlist(tet)) -> allt
    # importances
    sapply(allt, function(x) sum(ww[sapply(tet, function(t) x%in%t)])) -> imp
    return(imp)
}

# Funktionen um Daten mit Hilfe der
# Box-Cox-Transformation zu transformieren
BCTransform <- function(y, lambda=0) {
    if (lambda == 0L) log(y)
    else ((y^lambda - 1) / lambda)
}
# Rücktransformation mit Box-Cox
BCTransformInverse <- function(yt, lambda=0) {
    if (lambda == 0L) exp(yt)
    else exp(log(1 + lambda * yt)/lambda)
}

# Funktion zum Anlegen von Ordnern einer Datei
mk = function(name) {
  targetdir = dirname(file.path(getwd(), imgpath, name))
  if (!file.exists(targetdir)) dir.create(targetdir, recursive = TRUE)
  return(file.path(targetdir, basename(name)))
}

# Funktion um Plots als SVG zu speichern
saveAsSVG = function(plot, name, h = 7, w = 15.5) {
  # Eventuell Ordner anlegen
  mk(name)
  # Abspeichern als Vektorgrafik
  plot |> ggsave(filename = file.path(imgpath, name),
                 width = w, height = h, units = "cm")
}
```

Konstanten

```
# Inputdatei
inputdata = "Data/data.txt"
```

```r
# Abbildungsordner
imgpath = "Images/"

# Response-Variable
response = "Bodenfeuchte_Prozent"
# Name der Response-Variable
responseName = "Bodenfeuchte [%]"

# Prädiktoren
predicts = c(
  "Tiefe_zum_Oxidationshorizont_cm",
  "Maechtigkeit_Oxidationshorizont_cm",
  "Maechtigkeit_A_Horizont_cm",
  "Maechtigkeit_AL_Schicht_cm",
  "Gelaendehoehe_m"
  )
# Namen der Prädiktoren
predictsNames = c(
  "Tiefe zum Oxidationshorizont [cm]",
  "Mächtigkeit Oxidationshorizont [cm]",
  "Mächtigkeit A-Horizont [cm]",
  "Mächtigkeit AL-Schicht [cm]",
  "Geländehöhe [m]"
  )
```

Daten einlesen

```r
# Alles Einlesen
readin = read.table(inputdata, header = TRUE)

# Variablen von Interesse auswählen
data = readin |>
  select(all_of(c("Punkt", "Jahr", "Stratum", "x_UTM", "y_UTM",
                  predicts, response)))

# Neue Spalte anlegen mit Namen der Straten
data = data %>%
  mutate(StratumName = case_when(
    Stratum == 1 ~ "Talsand",
    Stratum == 2 ~ "Übergang",
    Stratum == 3 ~ "Feuchtwiese"
    ))
```

Modelldaten selektieren

```r
# Daten für die Modellerstellung
modeldata = data |>
  select(all_of(c(predicts, response))) |>
  na.omit()

modeldata_scaled = apply(X = modeldata, FUN = scale, MARGIN = 2)

# Prädiktoren selektieren
```

```r
predictors = modeldata |>
  select(-one_of(response))

# Kartenansicht der Daten
# tm_shape(sf(data)) + tm_dots(col = response)
```

Datenexploration

Korrelationsuntersuchung

```r
# Korrelationsmatrix mit ggpairs
corplot = ggpairs(modeldata) +
  # Layout und theme
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  theme_bw()

# Export von 'corplot' als SVG
saveAsSVG(corplot, name = "sonstige/Korrelationsmatrix.svg", h = 15.5)
```

Einfaches lineares Modell

Modellerstellung

```r
# Modellerstellung
linModel = lm(Bodenfeuchte_Prozent ~ ., data = modeldata)

# Vorhersage und Fehler speichern
data = pred(name = "linModel", model = linModel, predictors = predictors,
            data = data, response = response)
```

Modellbewertung

```r
# Default plots
# plot(linModel)

# Residuen des einfachen LMs
reslin <- resid(linModel)
# Sind die Residuen normalverteilt?
shapiro.test(reslin)
# Dichtekurve der Residuen
# plot(density(reslin))

# Modellbeschreibung
summary(linModel)

# Model-Wert für das lineare Modell
AIC(linModel)

# RMSE
rmse(actual = data[!is.na(data$linModel), response],
     predicted = data[!is.na(data$linModel), "linModel"])
```

```
# Varianz
var(yobs = data[!is.na(data$linModel), response], ymod = na.omit(data$linModel))

# VIF - Dataframe
vifdf = data.frame(vif = vif(linModel), predictors = predictsNames)

# ANOVA für Testung, ob sich Vorhersagen für Straten unterscheiden
bartlett.test(data$errlinModel~ data$StratumName)
# => p-value = 0.8407: Varianzen gleich
results = aov(errlinModel ~ StratumName, data=data)
summary(results)
# => p-value: 0.751: Ergebnis nicht signifikant,
# Vorhersagen unterscheiden sich nicht
```

Visualisierung

```
# Vorhersage Einfaches lineares Modell
plot_linModelPrediction = data |>
  # Zeilen ohne Vorhersagewert löschen
  drop_na(linModel) |>
  # ggplot initialisieren
  ggplot(aes(x = Bodenfeuchte_Prozent, y = linModel)) +
  # Scatterplot
  geom_point(aes(col = StratumName)) +
  scale_color_manual(values = colouring) +
  # Ursprungsgerade mit Steigung 1
  geom_abline(slope = 1) +
  # Beschriftungen und theme
  labs(x = r"($y$)", y = r"($y_{linModel}$)", colour = "Stratum")+
  annotate("text", x = 52, y = 25, label = "y = 1x") +
  theme_bw()

# Export als SVG
saveAsSVG(plot_linModelPrediction, "linModel/linModelPrediction.svg")

# Fehler ~ Stratum
plot_linModelError = data |>
  # Zeilen ohne Fehlerwert löschen
  drop_na(errlinModel) |>
  # ggplot initialisieren
  ggplot(aes(x = StratumName, y = errlinModel)) +
  # Boxplot anlegen
  geom_boxplot(aes(fill = StratumName)) +
  scale_fill_manual(values = colouring) +
  # Beschriftungen, Layout und theme
  labs(x = "", y = r"($\sqrt{(y-y_{linModel})^2}$)")+
  theme_bw() +
  theme(legend.position="none")

# Export als SVG
saveAsSVG(plot_linModelError, "linModel/linModelError.svg")
# Export der Kombination aus plot_linModelPrediction und plot_linModelError
```

```r
saveAsSVG((plot_linModelPrediction | plot_linModelError),
          "linModel/linModel_page.svg")

# Regressionsgeraden mit VisReg
p1 <- visreg(linModel, xvar = "Gelaendehoehe_m",
             gg = TRUE, ylab = "")
p2 <- visreg(linModel, xvar = "Tiefe_zum_Oxidationshorizont_cm",
             gg = TRUE, ylab = "")
p3 <- visreg(linModel, xvar = "Maechtigkeit_Oxidationshorizont_cm",
             gg = TRUE, ylab = "Bodenfeuchte[%]")
p4 <- visreg(linModel, xvar = "Maechtigkeit_A_Horizont_cm",
             gg = TRUE, ylab = "")
p5 <- visreg(linModel, xvar = "Maechtigkeit_AL_Schicht_cm",
             gg = TRUE, ylab = "")
# Export der Regressionsgeraden als SVG
saveAsSVG((p1+p2)/(p3+p4)/(p5+ plot_spacer()), h = 10,
          "linModel/linModel_Regressionsgeraden.svg")

# VIF - Plot
vifplot = vifdf |>
  # ggplot initialisieren
  ggplot(aes(x = predictors, y = vif)) +
  # Balkendiagramm
  geom_bar(stat = "identity", fill = "white", col = "black") +
  # Balkenbeschriftung
  geom_text(aes(label = format(round(vif, 2))), vjust = 2, size = 3.5) +
  # Beschriftung, Layout und theme
  labs(x = "Prädiktoren", y = "VIF") +
  scale_x_discrete(labels = function(x) {
    # Funktion für Zeilenumbrüche in  x-Achsenbeschriftung
    gsub("\n", "\n\n", str_wrap(x, width = 10))
  }) +
  theme_bw()

# Abspeichern des VIF - Plots als Vektorgrafik
saveAsSVG(vifplot, "sonstige/VIF.svg", h = 7)

# Barplot zur Cooks-Distance
{svg(mk("linModel/linModel_cooksdistance.svg"), width = 15.5, height = 7)
plot(linModel, 4, main = "", xaxt = "n", ann = F)
title(xlab="Datenpunkt", ylab ="Cook´sche Distanz")
text(x= 40, y= 0.3,label = "max = 0,326", pos = 4, cex = 0.8)
text(x= 40, y= 0.25,label = "mean = 0,023", pos = 4,cex = 0.8)
text(x= 40, y= 0.2,label = "min = 6,00e-05", pos = 4, cex = 0.8)
dev.off()}

max(cooks.distance(linModel))
# => 0.3255527
min(cooks.distance(linModel))
# => 6.002479e-05
mean(cooks.distance(linModel))
# => 0.02328232
```

```r
# Dichtekurve mit ggplot
normdist = data.frame(res = linModel$residuals) |>
  # ggplot initialisieren
  ggplot(aes(x = res)) +
  # Verteilung
  geom_density() +
  # Beschriftung und theme
  labs(x = r"($y-y_{linModel}$)") +
  theme_classic()

# Export als SVG
saveAsSVG(normdist, "linModel/linModel_ResidualsDensity.svg")
```

Autom. Modellselektion (LM)

Modellerstellung

```r
# Mögliche lineare Modelle nach Güte untersuchen und ordnen
res = glmulti(linModel, level = 1, method = "h", confsetsize = 8, crit = "AIC")
res_scaled = glmulti(lm(Bodenfeuchte_Prozent ~ ., data = modeldata), level = 1, method = "h", confsetsi
```

```r
# Das beste Modell übernehmen
bestlinModel = res@objects[[1]]
# Vorhersage und Fehler speichern
data = pred(name = "bestlinModel", model = bestlinModel, response = response,
            data = data, predictors = predictors)
```

Modellbewertung

```r
# Beschreibung des besten Modells
summary(bestlinModel)

# AIC des besten linearen Modells
res@crits[1]

# AICs der ersten 8 Modelle
mean(res@crits)
svg(mk("linModelAuto/AICs.svg"));plot(res);dev.off()

# Varianz
var(yobs = data[!is.na(data$bestlinModel), response],
    ymod = na.omit(data$bestlinModel))

# RMSE
rmse(actual = data[!is.na(data$bestlinModel), response],
     predicted = data[!is.na(data$bestlinModel), "bestlinModel"])

# Wichtigkeit von Prädiktoren
imp = data.frame(importance = getImportance(res))
plot(res, type = "s")
```

```
#ANOVA für Testung, ob sich Vorhersagen für Straten unterscheiden
bartlett.test(data$errbestlinModel~ data$StratumName)
# => p-value = 0.9685: Varianzen gleich
results = aov(errbestlinModel ~ StratumName, data=data)
summary(results)
# => p-value: 0.55: Ergebnis nicht signifikant.
# Vorhersagen unterscheiden sich nicht
```

Visualisierung

```
# Default Plots
# plot(bestlinModel)

# Vorhersage bestes lineares Model
plot_linModelAutoPrediction = data |>
  # Zeilen ohne Vorhersagewerte löschen
  drop_na(bestlinModel) |>
  # ggplot initialisieren
  ggplot(aes(x = Bodenfeuchte_Prozent, y = bestlinModel)) +
  # Scatterplot
  geom_point(aes(col = StratumName)) +
  scale_color_manual(values = colouring) +
  # Ursprungsgerade mit Steigung 1
  geom_abline(slope = 1) +
  # Beschriftungen
  labs(x = r"($y$)", y = r"($y_{linModelAuto}$)", colour = "Stratum")+
  annotate("text", x = 52, y = 25, label = "y = 1x") +
  theme_bw()

# Export als SVG
saveAsSVG(plot_linModelAutoPrediction,
          "linModelAuto/linModelAutoPrediction.svg")

# Fehler ~ Stratum
plot_linModelAutoError = data |>
  # Zeilen ohne Fehlerwert löschen
  drop_na(errbestlinModel) |>
  # ggplot initialisieren
  ggplot(aes(x = StratumName, y = errbestlinModel)) +
  # Boxplot
  geom_boxplot(aes(fill = StratumName)) +
  scale_fill_manual(values = colouring) +
  # Beschriftungen, Layout und theme
  labs(x = "", y = r"($\sqrt{(y-y_{linModelAuto})^2}$)")+
  theme_bw() +
  theme(legend.position="none")

# Export als SVG
saveAsSVG(plot_linModelAutoError, "linModelAuto/linModelAutoError.svg")
# Export der Kombination plot_linModelAutoPrediction und plot_linModelAutoError
saveAsSVG((plot_linModelAutoPrediction | plot_linModelAutoError),
          "linModelAuto/linModelAuto_page.svg")
```

```r
# Plot zur Wichtigkeit von Prädiktoren
impplot = imp |>
  # ggplot initialisieren
  ggplot(aes(x = importance, y = reorder(rownames(imp), importance))) +
  # Barplot
  geom_bar(stat="identity", fill = "white", col = "black") +
  # Vertikale Linie bei 0.8
  geom_vline(xintercept = 0.8, color = "red") +
  # Beschriftungen und theme
  labs(x = "", y = "") +
  theme_bw()

# Export als SVG
saveAsSVG(impplot, "sonstige/Importance.svg", h = 3.5)

# Cooks Distance
{svg(mk("linModelAuto/linModelAuto_cooksdistance.svg"), width = 15.5,
     height = 7)
plot(bestlinModel, 4, main = "", xaxt = "n", ann = F)
title(xlab="Datenpunkt", ylab ="Cook´sche Distanz")
text(x= 40, y= 0.2,label = "max = 0,199", pos = 4, cex = 0.8)
text(x= 40, y= 0.18,label = "mean = 0,022", pos = 4,cex = 0.8)
text(x= 40, y= 0.16,label = "min = 5,37e-05", pos = 4, cex = 0.8)
dev.off()}

max(cooks.distance(bestlinModel))
# => 0.1993562
min(cooks.distance(bestlinModel))
# => 5.374797e-06
mean(cooks.distance(bestlinModel))
# => 0.02204445
```

GLM - GAMMA

Modellerstellung

```r
# Arbeitskopie von 'modeldata' erstellen
modeldata_transf = data.frame(modeldata)

# Box-Cox-Transformation der Response-Variable
bc = MASS::boxcox(linModel)
```

```r
lambda = bc$x[which.max(bc$y)]
# Anwendung der Transformation auf die Arbeitskopie
modeldata_transf = modeldata_transf |>
  # Spalte 'Bodenfeuchte_Prozent' überschreiben
  mutate(Bodenfeuchte_Prozent = BCTransform(
    y = modeldata_transf$Bodenfeuchte_Prozent,
    lambda = lambda)
    )

# Aufstellen eines GLM. Linkfunktion: gamma
glm_gamma = glm(Bodenfeuchte_Prozent ~ ., data = modeldata_transf,
```

```
                family = gaussian(link = "inverse"))

# Rücktransformation der Vorhersage
glm_gamma_predictions = BCTransformInverse(predict(glm_gamma, type="response"),
                                           lambda = lambda)

# Abspechern und Fehler berechnen
data = pred(name = "glm_gamma", predictions = glm_gamma_predictions,
            data = data, response = response)
```

Modellbewertung

```
# Modellbericht
summary(glm_gamma)
# Default plots
# plot(glm_gamma)

# AIC des GLMS
AIC(glm_gamma)

# RMSE
rmse(actual = data[!is.na(data$glm_gamma), response],
     predicted = data[!is.na(data$glm_gamma), "glm_gamma"])

# Varianz
var(yobs = data[!is.na(data$glm_gamma), response],
    ymod = na.omit(data$glm_gamma))

# Cook´s Distance
{svg(mk("glm/glm_cooksdistance.svg"), width = 15.5, height = 7)
plot(glm_gamma, 4, main = "", xaxt = "n", ann = F)
title(xlab="Datenpunkt", ylab ="Cook´sche Distanz")
text(x= 33, y= 0.09,label = "max = 0,101", pos = 4, cex = 0.8)
text(x= 33, y= 0.08,label = "mean = 0,019", pos = 4,cex = 0.8)
text(x= 33, y= 0.07,label = "min = 9,22e-07", pos = 4, cex = 0.8)
dev.off()}

max(cooks.distance(glm_gamma))
# => 0.1013789
min(cooks.distance(glm_gamma))
# => 9.217823e-07
mean(cooks.distance(glm_gamma))
# => 0.0185927

# ANOVA für Testung, ob sich Vorhersagen für Straten unterscheiden
bartlett.test(data$errglm_gamma~ data$StratumName)
# => p-value = 0.2924: Varianzen gleich
results = aov(errglm_gamma ~ StratumName, data=data)
summary(results)
# => p-value = 0.437: Ergebnis nicht signifikant.
# Vorhersagen unterscheiden sich nicht
```

Visualisierung

```
# Vorhersage GLM
plot_glmPrediction = data |>
  # Zeilen ohne GLM Vorhersagen löschen
  drop_na(glm_gamma) |>
  # ggplot initialisieren
  ggplot(aes(x = Bodenfeuchte_Prozent, y = glm_gamma)) +
  # Scatterplot
  geom_point(aes(col = StratumName)) +
  scale_color_manual(values = colouring) +
  # Ursprungsgerade mit Steigung 1
  geom_abline(slope = 1) +
  # Beschriftung und theme
  labs(x = r"($y$)", y = r"($y_{glm}$)", colour = "Stratum") +
  annotate("text", x = 52, y = 25, label = "y = 1x") +
  theme_bw()

# Export als SVG
saveAsSVG(plot_glmPrediction, "glm/glmPrediction.svg")

# Fehler ~ Stratum
plot_glmError = data |>
  # Zeilen ohne Fehlerwerte löschen
  drop_na(errglm_gamma) |>
  # ggplot initialisieren
  ggplot(aes(x = StratumName, y = errglm_gamma)) +
  # Boxplot
  geom_boxplot(aes(fill = StratumName)) +
  scale_fill_manual(values = colouring) +
  # Beschriftung, Layout und theme
  labs(x = "", y = r"($\sqrt{(y-y_{glm})^2}$)")+
  theme_bw() +
  theme(legend.position="none")

# Export als SVG
saveAsSVG(plot_glmError, "glm/glmError.svg")
# Export als Kombination von Vorhersage und Fehler
saveAsSVG((plot_glmPrediction | plot_glmError), "glm/glm_page.svg")
```

Autom. Modellselektion (GLM)

Modellerstellung

```
# Mögliche GLMs nach Güte untersuchen und ordnen
resglm = glmulti(glm_gamma, level = 1, method = "h", confsetsize = 8,
                 crit = "AIC")
```

```
# Das beste Modell übernehmen
bestglm = resglm@objects[[1]]
# Vorhersage mit Rücktransformation
bestglm_pred = BCTransformInverse(predict(bestglm, newdata = predictors),
                                  lambda = lambda)
```

```
# Speichern und Fehlerberechnung
data = pred(name = "bestglm", predictions = bestglm_pred, data = data,
            response = response)
```

Modellbewertung

```
# Beschreibung des besten GLMs
summary(bestglm)

# AIC des besten GLMs
resglm@crits[1]
# AICs der ersten 8 Modelle
mean(resglm@crits)
plot(resglm)
```

```
# Varianz
var(yobs = data[!is.na(data$bestglm), response], ymod = na.omit(data$bestglm))

# RMSE
rmse(actual = data[!is.na(data$bestglm), response],
     predicted = data[!is.na(data$bestglm), "bestglm"])

#Cook´s Distance
{svg(mk("glmAuto/glmAuto_cooksdistance.svg"), width = 15.5, height = 7)
plot(bestglm, 4, main = "", xaxt = "n", ann = F)
title(xlab="Datenpunkt", ylab ="Cooksche Distanz")
text(x= 37, y= 0.13,label = "max = 0,152", pos = 4, cex = 0.8)
text(x= 37, y= 0.12,label = "mean = 0,021", pos = 4,cex = 0.8)
text(x= 37, y= 0.11,label = "min = 9,41e-05", pos = 4, cex = 0.8)
dev.off()}

max(cooks.distance(bestglm))
# => 0.1523492
min(cooks.distance(bestglm))
# => 9.41153e-05
mean(cooks.distance(bestglm))
# => 0.02075966

# ANOVA für Testung, ob sich Vorhersagen für Straten unterscheiden
bartlett.test(data$errbestglm~ data$StratumName)
# => p-value = 0.305: Varianzen gleich
results = aov(data$errbestglm~data$StratumName)
summary(results)
# => p-value = 0.123: Ergebnis nicht signifikant.
# Vorhersagen unterscheiden sich nicht
```

Visualisierung

```
# Vorhersage bestes GLM
plot_glmAutoPrediction = data |>
  # Zeilen ohne Vorhersage des besten GLMs löschen
  drop_na(bestglm) |>
  # ggplot initialisieren
```

```
  ggplot(aes(x = Bodenfeuchte_Prozent, y = bestglm)) +
  # Scatterplot
  geom_point(aes(col = StratumName)) +
  scale_color_manual(values = colouring) +
  # Ursprungsgerade mit Steigung 1
  geom_abline(slope = 1) +
  # Beschriftung und theme
  labs(x = r"($y$)", y = r"($y_{glmAuto}$)", colour = "Stratum")+
  annotate("text", x = 52, y = 25, label = "y = 1x") +
  theme_bw()

# Export als SVG
saveAsSVG(plot_glmAutoPrediction, "glmAuto/glmAutoPrediction.svg")

# Fehler ~ Stratum
plot_glmAutoError = data |>
  # Zeilen ohne Fehlerangabe löschen
  drop_na(errbestglm) |>
  # ggplot initialisieren
  ggplot(aes(x = StratumName, y = errbestglm)) +
  # Boxplot
  geom_boxplot(aes(fill = StratumName)) +
  scale_fill_manual(values = colouring) +
  # Beschriftung, Layout und theme
  labs(x = "", y = r"($\sqrt{(y-y_{glmAuto})^2}$)")+
  theme_bw() +
  theme(legend.position="none")

# Export als SVG
saveAsSVG(plot_glmAutoError, "glmAuto/glmAutoError.svg")
# Export der kombinierten plots
saveAsSVG((plot_glmAutoPrediction|plot_glmAutoError),
          "glmAuto/glmAuto_page.svg")
```

Vergleich lineare Modelle

```
# Scatterplot mit allen Vorhersagen
scatterLinPreds = data |>
  # Dataframe umgestalten. Spalte mit modelltypen hinzufügen.
  pivot_longer(cols=c("linModel",
                      "bestlinModel",
                      "glm_gamma",
                      "bestglm"),
               names_to='Modeltype',
               values_to='predictions') |>
  # Entfernen von Zeilen ohne Fehlerangaben
  drop_na(predictions) |>
  # ggplot initialisieren
  ggplot(aes(x = Bodenfeuchte_Prozent, y = predictions, colour = Modeltype)) +
  # Scatterplot
  geom_point(aes(color = Modeltype)) +
  # Ursprungsgerade mit Steigung 1
```

```
  geom_abline(slope = 1) +
  # Beschriftungen und theme
  labs(x = "Messwerte", y = "Vorhersage") +
  annotate("text", x = 13, y = 7, label = "y = 1x") +
  theme_bw()

# Boxplot mit Fehlern nach Modell
vergLinModels = data |>
  # Dataframe umgestalten. Spalte mit modelltyp hinzufügen.
  pivot_longer(cols=c("errlinModel",
                      "errbestlinModel",
                      "errglm_gamma",
                      "errbestglm",
                      ),
               names_to='Modeltype',
               values_to='RootSquaredError') |>
  # Entfernen von Zeilen ohne Fehlerangaben
  drop_na(RootSquaredError) |>
  # Umbenennung der Modelltypen
  mutate(Modeltype = case_when(
    Modeltype == "errlinModel" ~ r"((1) $linModel$)",
    Modeltype == "errbestlinModel" ~ r"((2) $linModelAuto$)",
    Modeltype == "errglm_gamma" ~ r"((3) $glm$)",
    Modeltype == "errbestglm" ~ r"((4) $glmAuto$)"
    )) |>
  # ggplot initialisieren
  ggplot(aes(x = Modeltype, y = RootSquaredError)) +
  # Boxplot
  geom_boxplot() +
  # Beschriftung, Layout und theme
  labs(x = "", y = r"($\sqrt{(y-y_{...})^2}$)", fill = "Modelltyp") +
  theme_bw()
# Export als SVG Vektorgrafik
saveAsSVG(vergLinModels, "diskussion/vergleichLineareModelle.svg")
```

CART

Modellerstellung

```
# Baum erstellen
treeModel <- rpart(Bodenfeuchte_Prozent ~ ., data = modeldata)
# Vorhersage und Fehler speichern
data = pred(name = "treeModel", model = treeModel,
            predictors = predictors, data = data,
            response = response)
```

Modellbewertung

```
treeModel
rmse(data$Bodenfeuchte_Prozent[!is.na(data$treeModel)], na.omit(data$treeModel))

# ANOVA für Testung, ob sich Vorhersagen für Straten unterscheiden
```

```r
bartlett.test(data$errtreeModel~ data$StratumName)
# => p-value = 0.1116: Varianzen gleich
results = aov(errtreeModel ~ StratumName, data=data)
summary(results)
# => p-value: 0.0568: Ergebnis (knapp) nicht signifikant,
# Vorhersagen unterscheiden sich nicht
```

Visualisierung

```r
# Baumplot
treePlot = rpart.plot(treeModel, digits = 4)
```

```r
# Stufen plot in Eingabedaten
treeLine = data |>
  # Zeilen ohne Vorhersagewerte löschen
  drop_na(treeModel) |>
  # ggplot initialisieren
  ggplot(aes(x = Gelaendehoehe_m, y = Bodenfeuchte_Prozent)) +
  # Scatterplot
  geom_point(aes(col = StratumName)) +
  scale_color_manual(values = colouring, name = "Stratum") +
  # Stufenlinie hinzufügen
  geom_line(aes(y = treeModel), color = "red", linewidth = .75) +
  # BEschriftungen, Layout und theme
  labs(x = "Geländehoehe [m]", y = "Bodenfeuchte [%]") +
  theme_bw() +
  theme(legend.position = "none")

# Vorhersageplot für tree
treePred = data  |>
  # Zeilen ohne Vorhersagewerte löschen
  drop_na(treeModel) |>
  # ggplot initialisieren
  ggplot(aes(x = Bodenfeuchte_Prozent, y = treeModel)) +
  # Scatterplot
  geom_point(aes(col = StratumName)) +
  scale_color_manual(values = colouring, name = "Stratum") +
  # Ursprungsgerade mit Steigung 1
  geom_abline(slope = 1) +
  # Beschriftung, Layout und theme
  labs(x = r"($y$)", y = r"($y_{tree}$)") +
  annotate("text", x = 60, y = 40, label = "y = 1x") +
  theme_bw() +
  theme(legend.position = "bottom")

# Fehler des tree über die drei Straten
plot_treeError = data |>
  # Zeilen ohne Vorhersagewerte löschen
  drop_na(errtreeModel) |>
  # ggplot initialisieren
  ggplot(aes(x = StratumName, y = errtreeModel)) +
  # Boxplot
  geom_boxplot(aes(fill = StratumName)) +
```

```
  scale_fill_manual(values = colouring) +
  # Beschriftung, Layout und theme
  labs(x = r"($y$)", y = r"($\sqrt{(y-y_{cart})^2}$)")+
  scale_x_discrete(guide = guide_axis(n.dodge=2)) +
  theme_bw() +
  theme(legend.position="none", rect = element_rect(fill = "transparent"))

# Export als dreierkombi SVG
saveAsSVG((treeLine | treePred | plot_treeError), "CART/cart_page.svg")
```

Random Forest

Modellerstellung

```
# Wald anlegen
randomForest = randomForest(Bodenfeuchte_Prozent ~ ., data = modeldata)

# INB Vorhersage
inb = predict(randomForest, newdata = predictors)
data = pred(name = "INB", predictions = inb, data = data, response = response)

# OOB Vorhersage
oob = predict(randomForest)
data = pred(name = "OOB", predictions = oob, data = data, response = response)
```

Modellbewertung

```
randomForest

# Varianz
var(yobs = data[!is.na(data$OOB), response], ymod = na.omit(data$OOB))
var(yobs = data[!is.na(data$INB), response], ymod = na.omit(data$INB))

rmse(data$Bodenfeuchte_Prozent[!is.na(data$OOB)], na.omit(data$OOB))
rmse(data$Bodenfeuchte_Prozent[!is.na(data$INB)], na.omit(data$INB))

# ANOVA für Testung, ob sich Vorhersagen für Straten unterscheiden
bartlett.test(data$errOOB~ data$StratumName)
# => p-value = 0.3532: Varianzen gleich
results = aov(errOOB~ StratumName, data=data)
summary(results)
# => p-value: 0.0242: Ergebnis signifikant,
# Vorhersagen unterscheiden sich
```

Visualisierung

```
# Quadrierte Fehler im RF
{svg(mk("RF/rfs.svg"), width = 15.5, height = 7)
plot(randomForest)
dev.off()}
```

```
RFplot= data |>
  # Spalte mit OOB/INB hinzufügen.
  pivot_longer(cols=c("OOB", "INB"),
               names_to='Vorhersagetyp',
               values_to='Vorhersage') |>
  mutate(Vorhersagetyp = case_when(
    Vorhersagetyp == "INB" ~ "Normal",
    Vorhersagetyp == "OOB" ~ "Out-of-bag")) |>
  # Zeilen ohne vorhersagewerte löschen
  drop_na(Vorhersage) |>
  # ggplot initialisieren
  ggplot(aes(x = Bodenfeuchte_Prozent, y = Vorhersage)) +
  # Scatterplot
  geom_point(aes(alpha = Vorhersagetyp, col = StratumName)) +
  scale_color_manual(values = colouring) +
  scale_alpha_discrete(range = c(1.0, 0.17), guide = 'none') +
  geom_abline(slope = 1) +
  annotate("text", x = 60, y = 40, label = "y = 1x") +
  labs(x = r"($y$)", y = r"($y_{RF}$)") +
  # Theme
  theme_bw()

saveAsSVG(RFplot, "RF/RFplot.svg")

RFplotError = data |>
  # Zeilen ohne Fehlerangabe löschen
  drop_na(errINB) |>
  # ggplot initialisieren
  ggplot(aes(x = StratumName, y = errINB)) +
  # Boxplot
  geom_boxplot(aes(fill = StratumName)) +
  scale_fill_manual(values = colouring) +
  # Beschriftung, Layout und theme
  labs(x = "", y = r"($\sqrt{(y-y_{RF})^2}$)")+
  theme_bw() +
  theme(legend.position="none")

saveAsSVG(RFplotError, "RF/RFplotError.svg")

saveAsSVG((RFplot | RFplotError), "RF/RF_page.svg")

# Variableneinfluss
{svg(mk("RF/varImportance.svg"), width = 15.5, height = 7)
varImpPlot(randomForest(Bodenfeuchte_Prozent ~ ., data = modeldata,
                        importance = TRUE), type = "1")
dev.off()}
```

Tuned RF

Modellerstellung

```
{svg(mk("tunedRF/mtry.svg"), width = 15.5, height = 7)
tunedRF = tuneRF(x = predictors, y=modeldata$Bodenfeuchte_Prozent, doBest=TRUE,
                 ntreeTry=5000, plot = TRUE)
dev.off()}

tunedPrediction = predict(tunedRF, newdata = predictors)
data = pred(name = "tunedRF", predictions = tunedPrediction, data = data,
            response = response)
```

Modellbewertung

```
tunedRF

rmse(data$Bodenfeuchte_Prozent[!is.na(data$tunedRF)],
     na.omit(data$tunedRF))
mean(data$errtunedRF, na.rm = TRUE)
var(yobs = data[!is.na(data$INB), response], ymod = na.omit(data$INB))

# ANOVA für Testung, ob sich Vorhersagen für Straten unterscheiden
bartlett.test(data$errtunedRF~ data$StratumName)
# => p-value = 0.8096: Varianzen gleich
results = aov(errtunedRF~ StratumName, data=data)
summary(results)
# => p-value: 0.115: Ergebnis nicht signifikant,
# Vorhersagen unterscheiden sich nicht
```

Visualisierung

```
{svg(mk("tunedRF/tunedRFrfs.svg"), width = 15.5, height = 7)
plot(tunedRF)
dev.off()}

# Vorhersageplot für tree
tunedRFplot = data  |>
  # Zeilen ohne Vorhersagewerte löschen
  drop_na(tunedRF) |>
  # ggplot initialisieren
  ggplot(aes(x = Bodenfeuchte_Prozent, y = tunedRF)) +
  # Scatterplot
  geom_point(aes(col = StratumName)) +
  scale_color_manual(values = colouring, name = "Stratum") +
  # Ursprungsgerade mit Steigung 1
  geom_abline(slope = 1) +
  # Beschriftung, Layout und theme
  labs(x = r"($y$)", y = r"($y_{tunedRF}$)") +
  annotate("text", x = 60, y = 40, label = "y = 1x") +
  theme_bw()

# Fehler des tree über die drei Straten
tunedRF_Error = data |>
  # Zeilen ohne Vorhersagewerte löschen
  drop_na(errtunedRF) |>
```

```
  # ggplot initialisieren
  ggplot(aes(x = StratumName, y = errtunedRF)) +
  # Boxplot
  geom_boxplot(aes(fill = StratumName)) +
  scale_fill_manual(values = colouring) +
  # Beschriftung, Layout und theme
  labs(x = r"($y$)", y = r"($\sqrt{(y-y_{tunedRF})^2}$)")+
  scale_x_discrete(guide = guide_axis(n.dodge=2)) +
  theme_bw() +
  theme(legend.position="none", rect = element_rect(fill = "transparent"))


# Export als dreierkombi SVG
saveAsSVG((tunedRFplot | tunedRF_Error), "tunedRF/tunedRF_page.svg")
```

Diskussion

Vergleich Fehler

Vergleich der Fehler der verwendeten Modelle in Abhängigkeit vom Stratum.

```
data |>
  # Spalte mit allen 6 Modelltypen hinzufügen
  pivot_longer(cols=c("errlinModel",
                      "errbestlinModel",
                      "errglm_gamma",
                      "errbestglm",
                      "errtreeModel",
                      "errINB",
                      "errtunedRF"
                      ),
               names_to='Modeltype',
               values_to='RootSquaredError') |>
  # Entfernen von Zeilen ohne Fehlerwerte
  drop_na(RootSquaredError) |>
  # ggplot initialisieren
  ggplot(aes(x = StratumName, y = RootSquaredError)) +
  # Plot-Elemente
  geom_boxplot(aes(fill = Modeltype)) +
  # Beschriftung und theme
  labs(x = "", y = "Wurzel aus Fehlerquadratsumme", fill = "Modelltyp") +
  theme_bw()
```

```
# Boxplot mit Fehlern nach Modell für alle Modelle
vergAllModels1 = data |>
  # Dataframe umgestalten. Spalte mit modelltyp hinzufügen.
  pivot_longer(cols=c("errlinModel",
                      "errbestlinModel",
                      "errglm_gamma",
                      "errbestglm"
                      ),
               names_to='Modeltype',
               values_to='RootSquaredError') |>
  # Entfernen von Zeilen ohne Fehlerangaben
```

```
  drop_na(RootSquaredError) |>
  # Umbenennung der Modelltypen
  mutate(Modeltype = case_when(
    Modeltype == "errlinModel" ~ r"((1) $linModel$)",
    Modeltype == "errbestlinModel" ~ r"((2) $linModelAuto$)",
    Modeltype == "errglm_gamma" ~ r"((3) $glm$)",
    Modeltype == "errbestglm" ~ r"((4) $glmAuto$)"
    )) |>
  # ggplot initialisieren
  ggplot(aes(x = Modeltype, y = RootSquaredError)) +
  # Boxplot
  geom_boxplot() +
  # Beschriftung, Layout und theme
  labs(x = "", y = r"($\sqrt{(y-y_{...})^2}$)", fill = "Modelltyp") +
  scale_x_discrete(guide = guide_axis(n.dodge=2)) +
  ylim(0, 22) +
  theme_bw()

vergAllModels2 = data |>
  # Dataframe umgestalten. Spalte mit modelltyp hinzufügen.
  pivot_longer(cols=c("errtreeModel",
                      "errINB",
                      "errtunedRF"
                      ),
               names_to='Modeltype',
               values_to='RootSquaredError') |>
  # Entfernen von Zeilen ohne Fehlerangaben
  drop_na(RootSquaredError) |>
  # Umbenennung der Modelltypen
  mutate(Modeltype = case_when(
    Modeltype == "errtreeModel" ~ r"((5) $cart$)",
    Modeltype == "errINB" ~ r"((6) $RF$)",
    Modeltype == "errtunedRF" ~ r"((t) $tunedRF$)"
    )) |>
  # ggplot initialisieren
  ggplot(aes(x = Modeltype, y = RootSquaredError)) +
  # Boxplot
  geom_boxplot() +
  # Beschriftung, Layout und theme
  labs(x = "", y = r"($\sqrt{(y-y_{...})^2}$)", fill = "Modelltyp") +
  ylim(0, 22) +
  theme_bw()

vergAllModels = (vergAllModels1|vergAllModels2)
# Export als SVG
saveAsSVG(vergAllModels, "diskussion/VergleichAlleModelle.svg")
```

Vergleich der Abweichung von der Diagonalen.

```
data |>
  # Dataframe umgestalten. Spalte mit modelltyp hinzufügen.
  pivot_longer(cols=c("linModel",
                      "bestlinModel",
                      "glm_gamma",
```

```
                      "bestglm",
                      "treeModel",
                      "INB"
                      ),
               names_to='Modeltyp',
               values_to='Vorhersage') |>
  # Entfernen von Zeilen ohne angabe von Fehlern
  drop_na(Vorhersage) |>
  # ggplot initialisieren
  ggplot(aes(x = Bodenfeuchte_Prozent, y = Vorhersage)) +
  # Scatterplot
  geom_point(aes(color = Modeltyp)) +
  # Ursprungsgerade mit Steigung 1
  geom_abline(slope = 1) +
  # Beschriftungen und theme
  labs(x = "Messwerte", y = "Vorhersage") +
  annotate("text", x = 13, y = 7, label = "y = 1x") +
  theme_bw()
```

Finale

```
# Abspeichern des data-Dataframes als csv-Datei
write.csv(data, file.path("Data", "DATA.csv"))
# Abspeichern des data-Dataframes als shape-Datei
write_sf(sf(data), file.path("shapefiles", "points.shp"), delete_layer = TRUE)
```