

# Umweltstatistik Bericht - R

Milena Mensching und Justus Weyers

2023-10-08

## Contents

<b>Technisches</b>	<b>2</b>
Pakete laden . . . . .	2
Einstellungen . . . . .	2
Funktionsdefinitionen . . . . .	3
Konstanten . . . . .	4
Daten einlesen . . . . .	5
Modelldaten selektieren . . . . .	5
<b>Datenexploration</b>	<b>6</b>
Korrelationsuntersuchung . . . . .	6
<b>Einfaches lineares Modell</b>	<b>6</b>
Modellerstellung . . . . .	6
Modellbewertung . . . . .	6
Visualisierung . . . . .	8
<b>Autom. Modellselektion (LM)</b>	<b>10</b>
Modellerstellung . . . . .	10
Modellbewertung . . . . .	11
Visualisierung . . . . .	13
<b>GLM - GAMMA</b>	<b>15</b>
Modellerstellung . . . . .	15
Modellbewertung . . . . .	16
Visualisierung . . . . .	18
<b>Autom. Modellselektion (GLM)</b>	<b>19</b>
Modellerstellung . . . . .	19
Modellbewertung . . . . .	20
Visualisierung . . . . .	23
<b>Vergleich lineare Modelle</b>	<b>23</b>
<b>CART</b>	<b>24</b>
Modellerstellung . . . . .	24
Modellbewertung . . . . .	25
Visualisierung . . . . .	25
<b>Random Forest</b>	<b>27</b>
Modellerstellung . . . . .	27
Modellbewertung . . . . .	27

Visualisierung . . . . .	28
<b>Tuned RF</b>	<b>29</b>
Modellerstellung . . . . .	29
Modellbewertung . . . . .	30
Visualisierung . . . . .	31
<b>Diskussion</b>	<b>32</b>
Vergleich Fehler . . . . .	32
<b>Finale</b>	<b>35</b>

## 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
```

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

## Funktionsdefinitionen

```
# 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 Calcano, 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"

```

```

# Abbildungsordner
imgpath = "Images/"

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

# Prädiktoren
predicts = c(
  "Tiefe_zum_Oxidationshorizont_cm",
  "Mächtigkeit_Oxidationshorizont_cm",
  "Mächtigkeit_A_Horizont_cm",
  "Mächtigkeit_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

```

# 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

```

# 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

```

```

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

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

```

## Datenexploration

### Korrelationsuntersuchung

```

# 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

```

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

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

```

### Modellbewertung

```

# Default plots
# plot(linModel)

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

##
##  Shapiro-Wilk normality test
##
## data:  reslin
## W = 0.98013, p-value = 0.4692

# Dichtekurve der Residuen
# plot(density(reslin))

# Modellbeschreibung
summary(linModel)

```

```
##
## Call:
## lm(formula = Bodenfeuchte_Prozent ~ ., data = modeldata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.6046  -4.8015  -0.5561   4.9916  21.2343
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)      3396.96256    249.54917   13.612  <2e-16 ***
## Tiefe_zum_Oxidationshorizont_cm      0.09613     0.04801    2.002  0.0506 .
## Maechtigkeit_Oxidationshorizont_cm    0.01117     0.06719    0.166  0.8686
## Maechtigkeit_A_Horizont_cm     -0.04807     0.09788   -0.491  0.6255
## Maechtigkeit_AL_Schicht_cm     -0.02232     0.03740   -0.597  0.5533
## Gelaendehoehe_m      -34.05902     2.55253  -13.343  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.947 on 51 degrees of freedom
## Multiple R-squared:  0.8668, Adjusted R-squared:  0.8538
## F-statistic: 66.4 on 5 and 51 DF,  p-value: < 2.2e-16
# Model-Wert für das lineare Modell
AIC(linModel)

## [1] 405.7214

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

## [1] 7.517353

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

##          SST          SSE          R2
## 2.418986e+04 3.221104e+03 8.668407e-01

# 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)

##
## Bartlett test of homogeneity of variances
##
## data:  data$errlinModel by data$StratumName
## Bartlett's K-squared = 0.34713, df = 2, p-value = 0.8407
# => p-value = 0.8407: Varianzen gleich
results = aov(errlinModel ~ StratumName, data=data)
summary(results)

##          Df Sum Sq Mean Sq F value Pr(>F)
## StratumName  2      11    5.475    0.287  0.751
```

```
## Residuals    54    1029    19.054
## 152 Beobachtungen als fehlend gelöscht

# => 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
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()}
```

```

## pdf
## 2

max(cooks.distance(linModel))

## [1] 0.3255527
# => 0.3255527
min(cooks.distance(linModel))

## [1] 6.002479e-05
# => 6.002479e-05
mean(cooks.distance(linModel))

## [1] 0.02328232
# => 0.02328232

# 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

```

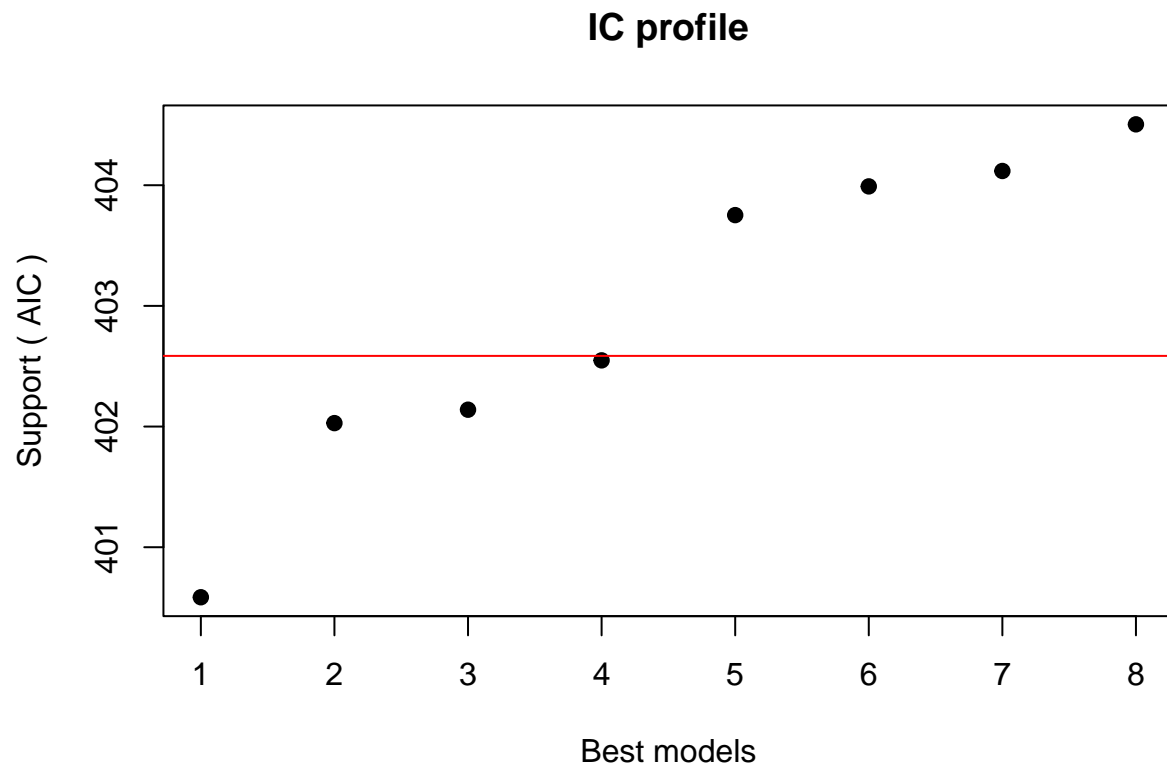
# Mögliche lineare Modelle nach Güte untersuchen und ordnen
res = glmulti(linModel, level = 1, method = "h", confsetsize = 8, crit = "AIC")

## Initialization...
## TASK: Exhaustive screening of candidate set.
## Fitting...
##
## After 50 models:
## Best model: Bodenfeuchte_Prozent~1+Tiefe_zum_Oxidationshorizont_cm+Gelaendehoehe_m
## Crit= 400.585517831168
## Mean crit= 402.958382257635
## Completed.

res_scaled = glmulti(lm(Bodenfeuchte_Prozent ~ ., data = modeldata), level = 1, method = "h", confsetsize = 8, crit = "AIC")

## Initialization...
## TASK: Exhaustive screening of candidate set.
## Fitting...
##
## After 50 models:
## Best model: Bodenfeuchte_Prozent~1+Tiefe_zum_Oxidationshorizont_cm+Gelaendehoehe_m
## Crit= 400.585517831168
## Mean crit= 402.958382257635

```



```
## Completed.
```

```
# 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

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

```
##
## Call:
## fitfunc(formula = as.formula(x), data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.6475  -5.6006   0.1699   4.0622  20.6839
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    3415.37606    181.34324   18.834 < 2e-16 ***
## Tiefe_zum_Oxidationshorizont_cm    0.09863     0.03105    3.176  0.00247 **
## Gelaendehoehe_m   -34.27033     1.84397  -18.585 < 2e-16 ***
```

```

## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.782 on 54 degrees of freedom
## Multiple R-squared:  0.8648, Adjusted R-squared:  0.8598
## F-statistic: 172.7 on 2 and 54 DF,  p-value: < 2.2e-16

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

## [1] 400.5855

# AICs der ersten 8 Modelle
mean(res@crits)

## [1] 402.9584

svg(mk("linModelAuto/AICs.svg"));plot(res);dev.off()

## pdf
## 2

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

##          SST          SSE          R2
## 2.418986e+04 3.270308e+03 8.648067e-01

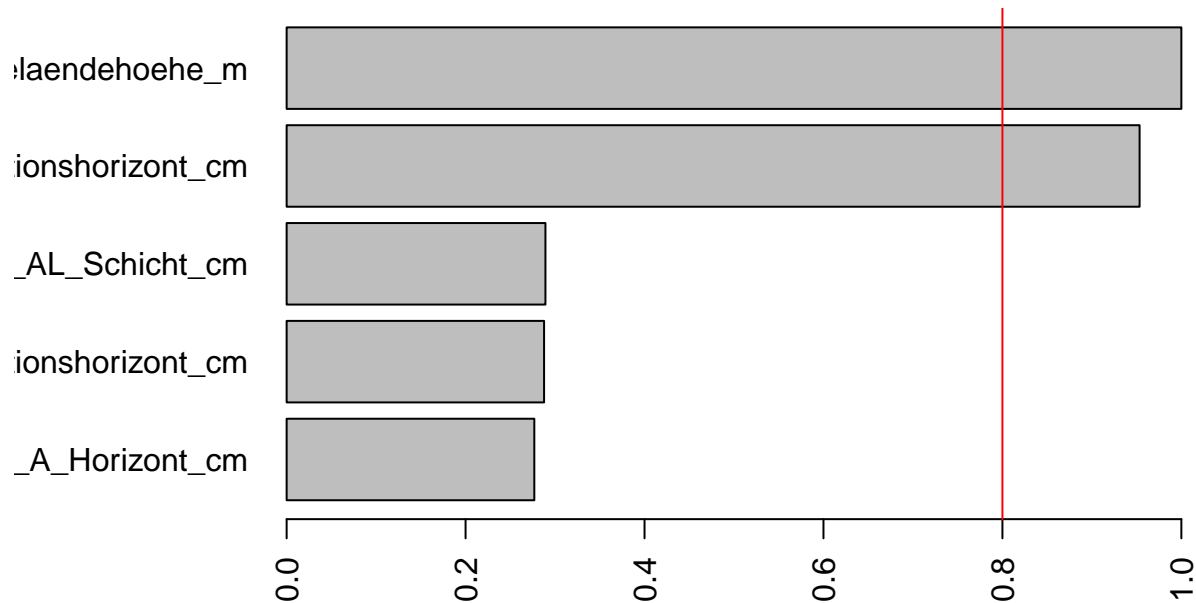
# RMSE
rmse(actual = data[!is.na(data$bestlinModel), response],
      predicted = data[!is.na(data$bestlinModel), "bestlinModel"])

## [1] 7.574551

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

```

## Model-averaged importance of terms



```
#ANOVA für Testung, ob sich Vorhersagen für Straten unterscheiden
bartlett.test(data$errbestlinModel~ data$StratumName)
```

```
##
## Bartlett test of homogeneity of variances
##
## data: data$errbestlinModel by data$StratumName
## Bartlett's K-squared = 0.064032, df = 2, p-value = 0.9685
```

```
# => p-value = 0.9685: Varianzen gleich
results = aov(errbestlinModel ~ StratumName, data=data)
summary(results)
```

```
##           Df Sum Sq Mean Sq F value Pr(>F)
## StratumName  2   24.2    12.1   0.605   0.55
## Residuals   54 1080.1    20.0
## 152 Beobachtungen als fehlend gelöscht
```

```
# => 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")

# 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()}
```

```
## pdf
## 2
```

```
max(cooks.distance(bestlinModel))
```

```
## [1] 0.1993562
```

```
# => 0.1993562
```

```
min(cooks.distance(bestlinModel))
```

```
## [1] 5.374797e-06
```

```
# => 5.374797e-06
```

```
mean(cooks.distance(bestlinModel))
```

```
## [1] 0.02204445
```

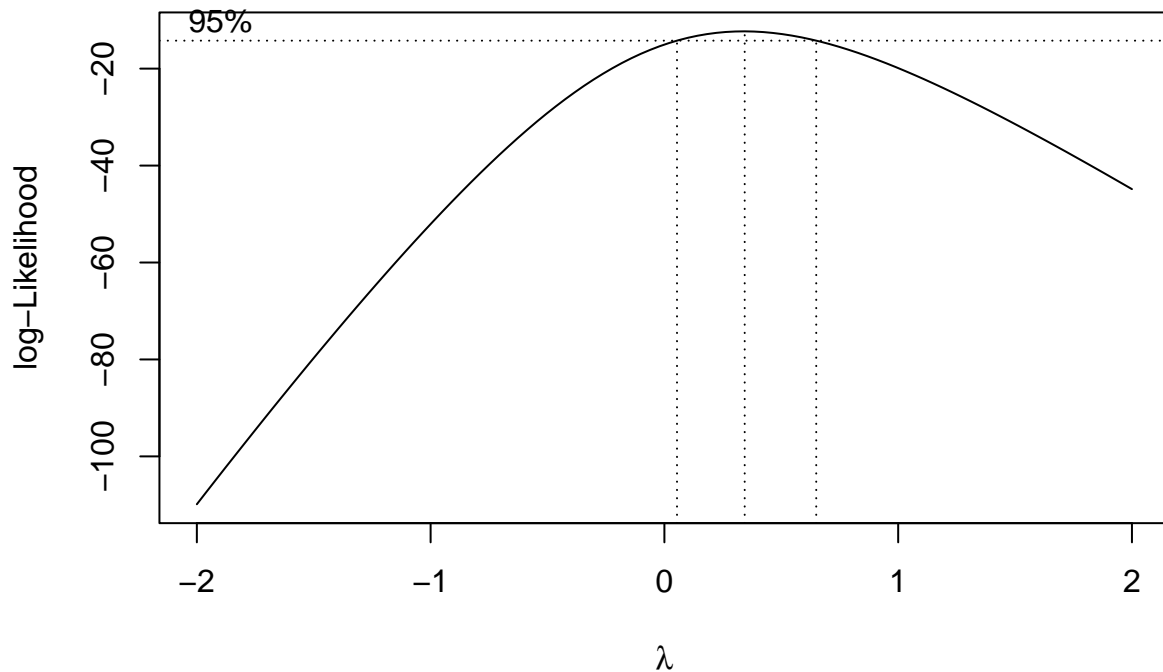
```
# => 0.02204445
```

## GLM - GAMMA

### Modellerstellung

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

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



```
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)

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

## Modellbewertung

```
# Modellbericht
summary(glm_gamma)
```



```
##
## Call:
## glm(formula = Bodenfeuchte_Prozent ~ ., family = gaussian(link = "inverse"),
##      data = modeldata_transf)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -1.47329  -0.34101  -0.01659   0.41201   1.50178
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -5.484e+00  3.820e-01 -14.358  <2e-16 ***
## Tiefe_zum_Oxidationshorizont_cm  -5.956e-05  4.895e-05  -1.217   0.229
## Maechtigkeit_Oxidationshorizont_cm -1.256e-04  7.752e-05  -1.621   0.111
## Maechtigkeit_A_Horizont_cm      -6.291e-05  1.307e-04  -0.481   0.632
## Maechtigkeit_AL_Schicht_cm       3.151e-05  4.273e-05   0.737   0.464
## Gelaendehoehe_m       5.716e-02  3.911e-03  14.616  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.3715144)
##
##      Null deviance: 176.298  on 56  degrees of freedom
## Residual deviance:  18.947  on 51  degrees of freedom
## AIC: 112.98
##
## Number of Fisher Scoring iterations: 5

# Default plots
# plot(glm_gamma)

# AIC des GLMS
AIC(glm_gamma)

## [1] 112.9799

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

## [1] 6.342931

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

##          SST          SSE          R2
## 2.418986e+04 2.293268e+03 9.051971e-01

# 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()}
```

```
## pdf
## 2
max(cooks.distance(glm_gamma))

## [1] 0.1013789
# => 0.1013789
min(cooks.distance(glm_gamma))

## [1] 9.217823e-07
# => 9.217823e-07
mean(cooks.distance(glm_gamma))

## [1] 0.0185927
# => 0.0185927

# ANOVA für Testung, ob sich Vorhersagen für Straten unterscheiden
bartlett.test(data$errglm_gamma ~ data$StratumName)

##
## Bartlett test of homogeneity of variances
##
## data: data$errglm_gamma by data$StratumName
## Bartlett's K-squared = 2.4592, df = 2, p-value = 0.2924
# => p-value = 0.2924: Varianzen gleich
results = aov(errglm_gamma ~ StratumName, data=data)
summary(results)

##           Df Sum Sq Mean Sq F value Pr(>F)
## StratumName  2   28.1   14.07   0.841  0.437
## Residuals  54  903.4   16.73
## 152 Beobachtungen als fehlend gelöscht
# => 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

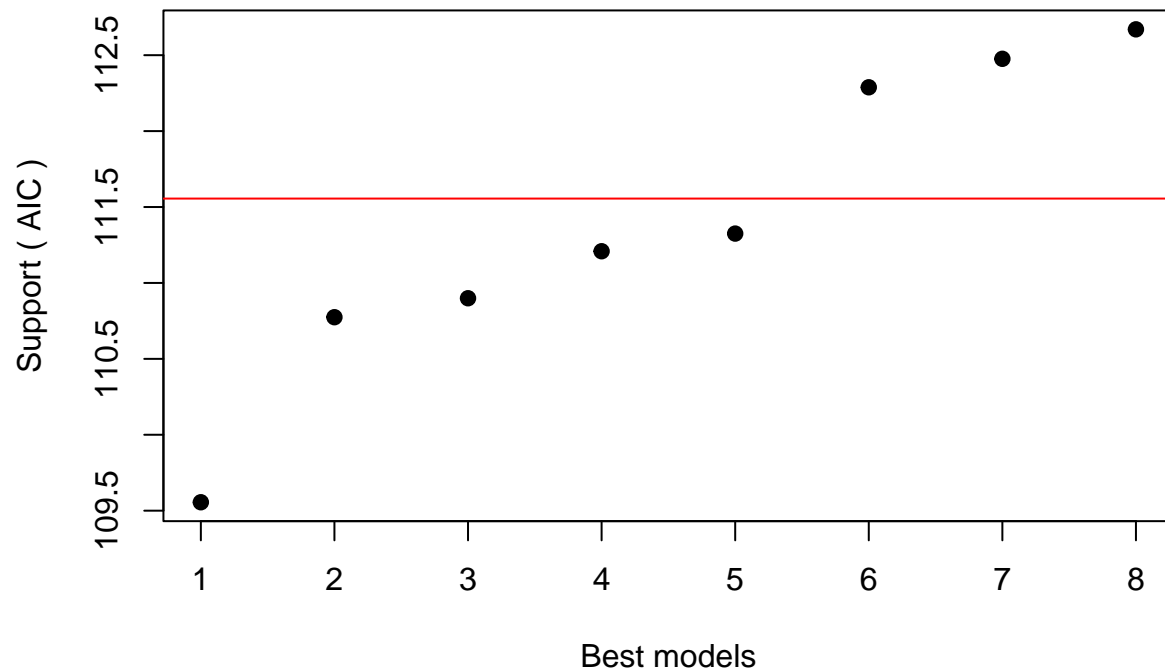
```

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

## Initialization...
## TASK: Exhaustive screening of candidate set.
## Fitting...
##
## After 50 models:
## Best model: Bodenfeuchte_Prozent~1+Tiefe_zum_Oxidationshorizont_cm+Gelaendehoehe_m
## Crit= 109.555594612136
## Mean crit= 111.399728708818

```

## IC profile



```
## Completed.
```

```
# 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)
```

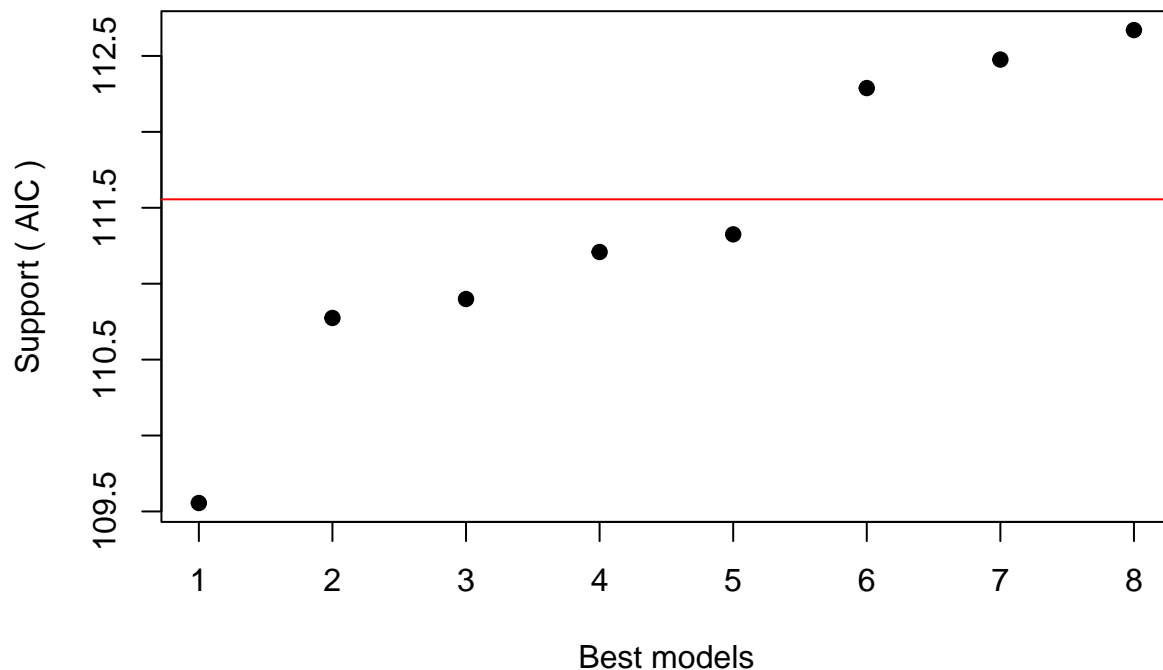
```
##
## Call:
## fitfunc(formula = as.formula(x), data = data)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -1.12302  -0.41725  -0.06289   0.35739   1.32455
##
## Coefficients:
##                                     Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)                298.058311  14.118663  21.111  <2e-16 ***
## Tiefe_zum_Oxidationshorizont_cm  0.004367  0.002418  1.806  0.0765 .
## Gelaendehoehe_m             -2.951432  0.143564 -20.558  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.3670957)
##
## Null deviance: 176.298  on 56  degrees of freedom
## Residual deviance:  19.823  on 54  degrees of freedom
## AIC: 109.56
##
## Number of Fisher Scoring iterations: 2
# AIC des besten GLMs
resglm@crits[1]

## [1] 109.5556
# AICs der ersten 8 Modelle
mean(resglm@crits)

## [1] 111.3997
plot(resglm)
```

## IC profile



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

```
##          SST          SSE          R2
## 2.418986e+04 2.688117e+03 8.888742e-01

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

## [1] 6.86731

#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()

## pdf
## 2

max(cooks.distance(bestglm))

## [1] 0.1523492
# => 0.1523492
min(cooks.distance(bestglm))

## [1] 9.41153e-05
# => 9.41153e-05
mean(cooks.distance(bestglm))

## [1] 0.02075966
# => 0.02075966

# ANOVA für Testung, ob sich Vorhersagen für Straten unterscheiden
bartlett.test(data$errbestglm~ data$StratumName)

##
## Bartlett test of homogeneity of variances
##
## data: data$errbestglm by data$StratumName
## Bartlett's K-squared = 2.3746, df = 2, p-value = 0.305
# => p-value = 0.305: Varianzen gleich
results = aov(data$errbestglm~data$StratumName)
summary(results)

##          Df Sum Sq Mean Sq F value Pr(>F)
## data$StratumName 2 67.8 33.91 2.178 0.123
## Residuals 54 840.9 15.57
## 152 Beobachtungen als fehlend gelöscht
# => 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"($\sqrt{(y-y_{\dots})^2}$)", 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

## n= 57
##
## node), split, n, deviance, yval
##      * denotes terminal node
##
## 1) root 57 24189.8600 45.04211
##    2) Gelaendehoehe_m>=98.209 33  2837.2390 29.96061
##      4) Gelaendehoehe_m>=99.1155 9   134.4000 18.00000 *
##      5) Gelaendehoehe_m< 99.1155 24   932.5196 34.44583
##        10) Gelaendehoehe_m>=98.8805 8   205.9187 29.46250 *
##        11) Gelaendehoehe_m< 98.8805 16   428.5975 36.93750 *
##    3) Gelaendehoehe_m< 98.209 24  3526.1000 65.77917
##      6) Gelaendehoehe_m>=97.909 10   207.4650 53.95000 *
##      7) Gelaendehoehe_m< 97.909 14   919.8486 74.22857 *

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

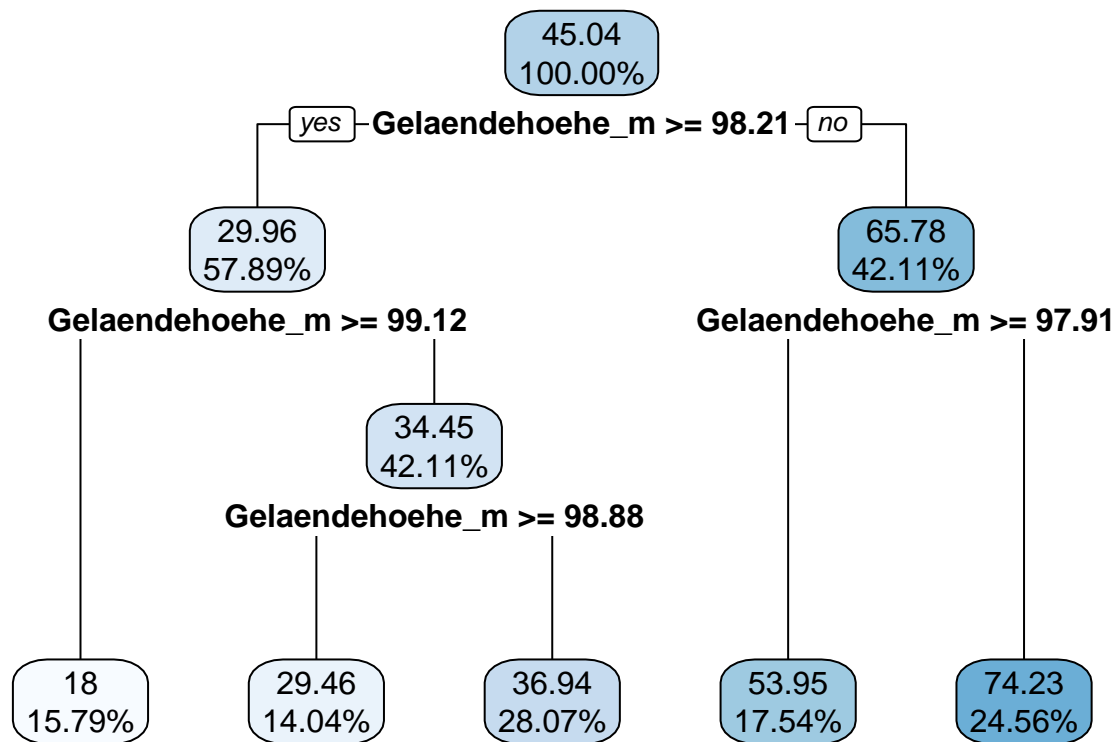
## [1] 5.767772
# ANOVA für Testung, ob sich Vorhersagen für Straten unterscheiden
bartlett.test(data$errtreeModel~ data$StratumName)

##
## Bartlett test of homogeneity of variances
##
## data:  data$errtreeModel by data$StratumName
## Bartlett's K-squared = 4.3859, df = 2, p-value = 0.1116
# => p-value = 0.1116: Varianzen gleich
results = aov(errtreeModel ~ StratumName, data=data)
summary(results)

##           Df Sum Sq Mean Sq F value Pr(>F)
## StratumName  2   63.5   31.76   3.027 0.0568 .
## Residuals  54  566.6   10.49
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 152 Beobachtungen als fehlend gelöscht
# => p-value: 0.0568: Ergebnis (knapp) nicht signifikant,
# Vorhersagen unterscheiden sich nicht
```

## Visualisierung

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



```

# 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

##
## Call:
## randomForest(formula = Bodenfeuchte_Prozent ~ ., data = modeldata)
##           Type of random forest: regression
##           Number of trees: 500
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 114.6442

```

```
##                                % Var explained: 72.99
# Varianz
var(yobs = data[!is.na(data$OOB), response], ymod = na.omit(data$OOB))

##          SST          SSE          R2
## 2.418986e+04 6.534719e+03 7.298571e-01

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

##          SST          SSE          R2
## 2.418986e+04 1.959610e+03 9.189904e-01

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

## [1] 10.7072

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

## [1] 5.863372

# ANOVA für Testung, ob sich Vorhersagen für Straten unterscheiden
bartlett.test(data$errOOB~ data$StratumName)

##
## Bartlett test of homogeneity of variances
##
## data: data$errOOB by data$StratumName
## Bartlett's K-squared = 2.0817, df = 2, p-value = 0.3532

# => p-value = 0.3532: Varianzen gleich
results = aov(errOOB~ StratumName, data=data)
summary(results)

##          Df Sum Sq Mean Sq F value Pr(>F)
## StratumName  2  300.5  150.26   3.988 0.0242 *
## Residuals   54 2034.5   37.68
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 152 Beobachtungen als fehlend gelöscht

# => 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()}

## pdf
## 2

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()

```

## Warning: Using alpha for a discrete variable is not advised.

```

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()}

```

```

## pdf
## 2

```

## 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()}

```

```
## mtry = 1   OOB error = 114.0413
## Searching left ...
## Searching right ...
## mtry = 2   OOB error = 72.82938
## 0.3613769 0.05
## mtry = 4   OOB error = 41.93734
## 0.42417 0.05
## mtry = 5   OOB error = 36.56692
## 0.1280582 0.05

## pdf
## 2

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

## Modellbewertung

```
tunedRF

##
## Call:
## randomForest(x = x, y = y, mtry = res[which.min(res[, 2]), 1])
##           Type of random forest: regression
##           Number of trees: 500
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 36.12484
##           % Var explained: 91.49

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

## [1] 2.689268

mean(data$errtunedRF, na.rm = TRUE)

## [1] 2.198602

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

##           SST           SSE           R2
## 2.418986e+04 1.959610e+03 9.189904e-01
# ANOVA für Testung, ob sich Vorhersagen für Straten unterscheiden
bartlett.test(data$errtunedRF~ data$StratumName)

##
## Bartlett test of homogeneity of variances
##
## data: data$errtunedRF by data$StratumName
## Bartlett's K-squared = 0.42249, df = 2, p-value = 0.8096
# => p-value = 0.8096: Varianzen gleich
results = aov(errtunedRF~ StratumName, data=data)
summary(results)

##           Df Sum Sq Mean Sq F value Pr(>F)
```

```
## StratumName  2  10.52   5.258    2.25  0.115
## Residuals    54 126.19   2.337
## 152 Beobachtungen als fehlend gelöscht
# => 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()}

## pdf
## 2

# 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"))

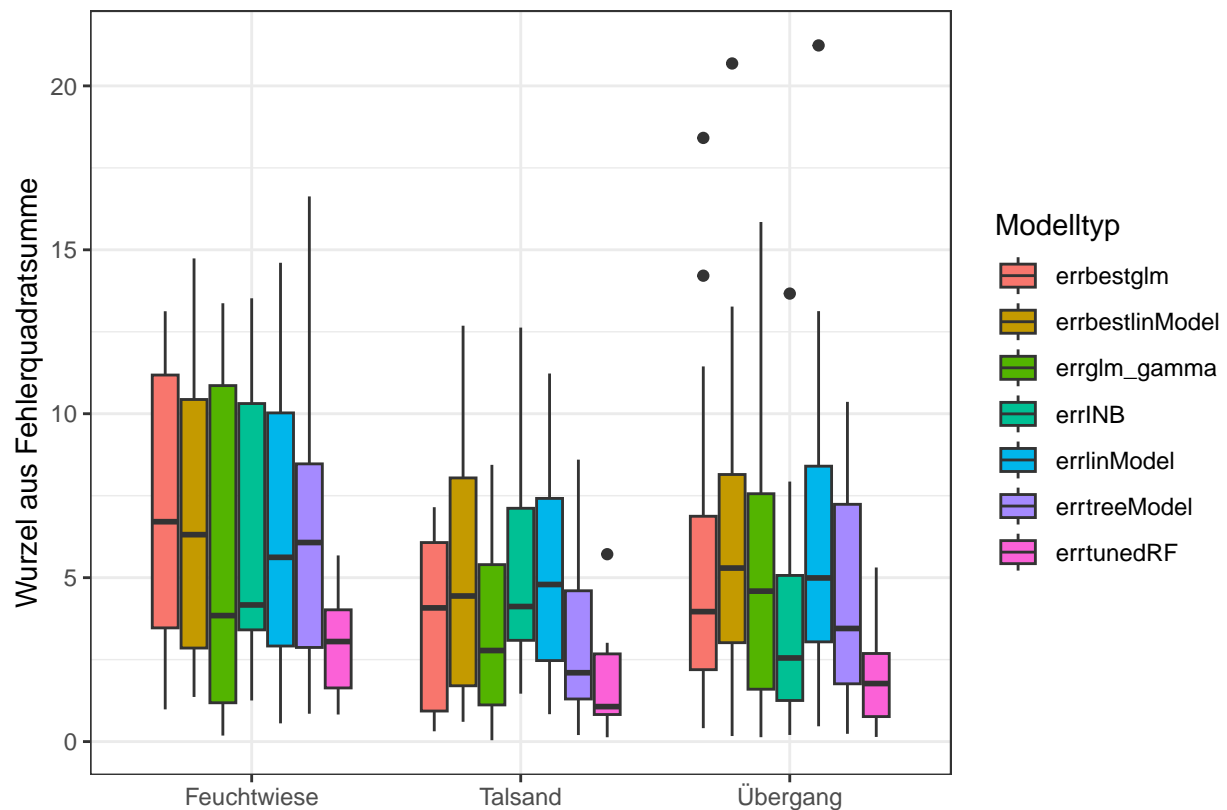
# Export als dreierkombi SVG
saveAsSVG((tunedRFplot | tunedRF_Error), "tunedRF/tunedRF_page.svg")
```

# 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")

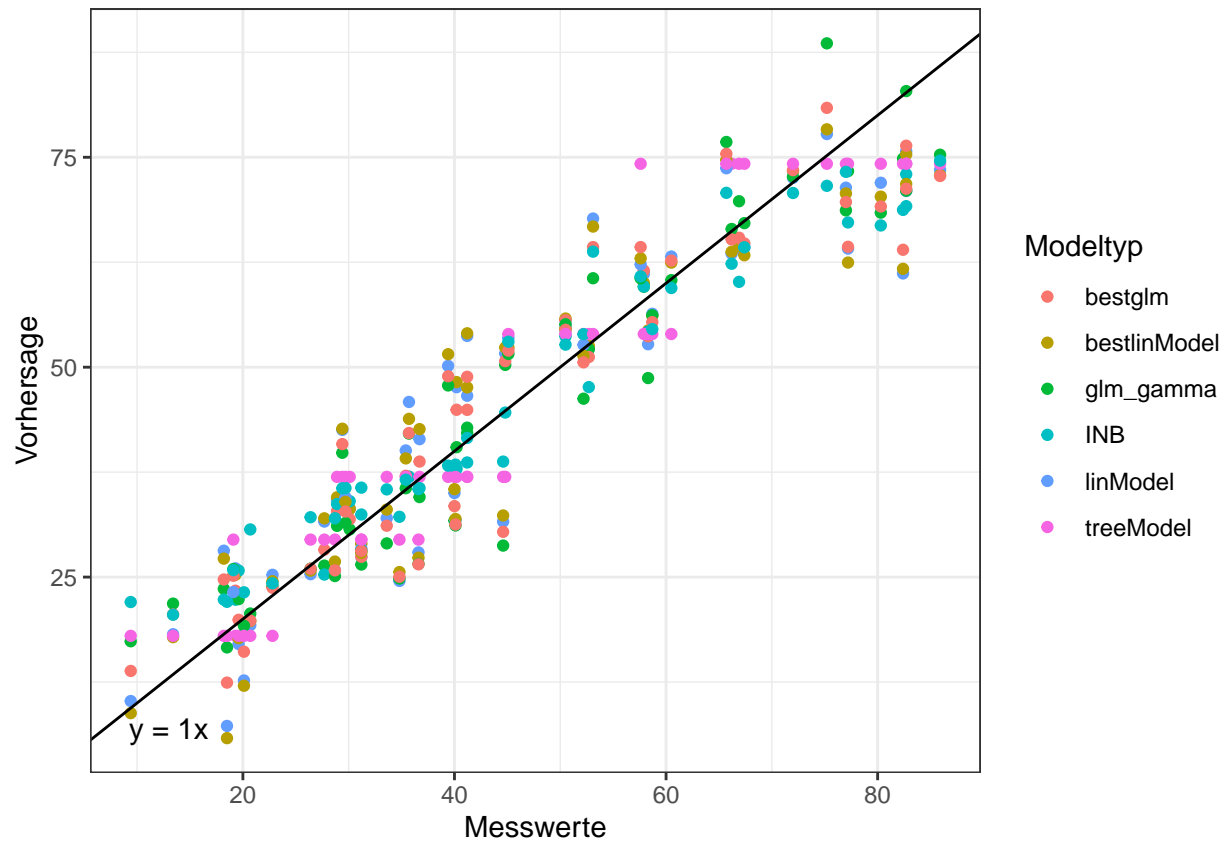
```

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)
```