Eine Analyse der ökologischen Nische der Fransenfledermaus *Myotis nattereri* in Europa unter Berücksichtigung der Klimaerwärmung

Fakultät für Umwelt und Natürliche Ressourcen
der Albert-Ludwigs-Universität Freiburg i. Br.

Dozent: Carsten Dormann

Ausgearbeitet von Gabriel Holz
Matrikelnummer: 3535995

30. Juni 2017

Inhaltsverzeichnis

1 Einleitung

Durch die globale Klimaerwärmung verschieben sich weltweit die Habitate von Lebewesen. Aus Sicht des Artenschutzes und der Schädlingsbekämpfung sind Szenarien über die Zukunft von großer Bedeutung, um Risiken abschätzen zu können. Um Szenarien modellieren zu können, muss die aktuelle Nische der jeweiligen Art möglichst gut beschrieben sein. In diesem Projekt wird die Fransenfledermaus Myotis nattereri genauer untersucht. Verwendet wurden Daten für Europa, erhoben vom Global Species Programme Red List Unit (IUCN). Die Daten bestehen aus einem binären georeferenzierten Raster. Die Zahl Eins indiziert, dass die Art vorhanden ist, die Zahl Null steht für das Fehlen der Art. In der folgenden Analyse werden die Habitatsansprüche der Fransenfledermaus Myotis nattereri anhand der folgenden Prädiktoren modelliert: mittlere Monatsminimaltemperatur (Tmin), Höhe über NN (ALTITUDE), mittlere Hangneigung (SLOPE), Jahrestemperatursumme (/> 5 C) (GDD), langjähriger Jahresniederschlag (PREYEAR), langjähriger Sommerniederschlag (PRESUMMER), Differenz zwischen Sommer- und Winterdurchschnittstemperatur (TSEASON) und folgende Landbedeckungsklassenanteile der Raster: BARE, CROP, GRASS, ICE, LAKE, MOSAIC, SEA, SHRUB, URBAN und WOOD. In einem zweiten Schritt wird die Temperatur (ausgedrückt in GDD) um 20 % erhöht und die Habitatsausdehnung mit dem aktuellen Zustand verglichen. Als Hintergrundinformationen sind noch folgende biologische Eigenschaften der Myotis nattereri zu nennen: Der Artname geht auf die Haare zurück, die sich auf der Schwanzflughaut befindenden, welche so wie Fransen aussehen. Der lateinische Name bezieht sich allerdings auf einen Zoologe namens Johann Natterer. Das Ausbreitungsgebiet in Europa erstreckt sich von Turkmenistan über den nahen Osten und Nord-West-Afrika bis an den 63. nördlichen Breitengrad. Auf Sardinen (Kuhl 1817) und einigen kleinen Inseln fehlt die Art. Die Fransenfledermaus ernährt sich, so wie viele andere Fledermäuse, durch Jagt auf Insekten und Spinnen. Dabei bevorzugt sie Wälder, insbesondere solche, die an Feuchtgebiete angrenzen. Im Sommer quartiert sie in kleinen Höhlen und löchern, wobei sie ihren Standort häufig wechselt. Auch die Fransenfledermaus hält einen Winterschlaf, meist in unterirdischen Höhlen oder in Gebäuden, in Gebieten des wärmeren Klimas auch oberirdisch. Das Winterquartier kann bis zu 185 km vom Sommerquartier entfernt liegen. In Europa nimmt die Art starkt an Abundanz ab, wodurch sie auf der roten Liste gefährdeter Tierarten steht. Gründe dafür sind der Mangel an geeigneten Quartieren im Sommer sowie im Winter. Schuld daran sind Fassadensanierungen und die Intensivierung der Landwirtschaft.

2 Methodik

Es wurden für die Vorhersage eines wärmeren Klimas zwei Modelle erstellt: Eines mit dem Random Forest Algorithmus und das zweite mit einem GLM. Wegen technischen Schwierigkeiten wurde das bessere Modell aus der Shrinkageanalyse nicht weiter verwendet.

2.1 Analyse der Prädiktoren

Durch die Analyse der Prädiktoren wird geschaut, ob diese möglichst uniform Verteilt sind, damit Ausreißer im Modell nicht höher gewichtet werden. Landklassen wurden bei sehr schiefen Verteilungen in binäre Klassen eingeteilt. Dabei wurde der Grenzwert der Klasseneinteilung auf 0% gesetzt. Transformiert wurden die Daten mittels Logarithmus oder Wurzel. Der Erfolg wurde anhand der folgenden Histogramme visuell quantifiziert. Mit dem Befehlt scale wurden die Daten standardisiert.

```
> #Datenladen
> load("/home/gabriel/Dokumente/Umwelt_statistik/SDM Rohdaten.rdata")
> allspec <- mamsamphsrepts
> art <- allspec$myotis_natter #Spalt des Auftretens der Fransenfledermaus
> art_predi <- allspec[ ,1:23]  #Spalten der Prädiktoren
```

```
> par(mfrow =c(5,4), mar=c(4,5,1,1)) #height=4, width=6
> for(i in 4:ncol(predi_trans)){
+   hist(predi_trans[,i], main=names(predi_trans)[i], xlab=NULL, ylab=NULL)
+ }
```

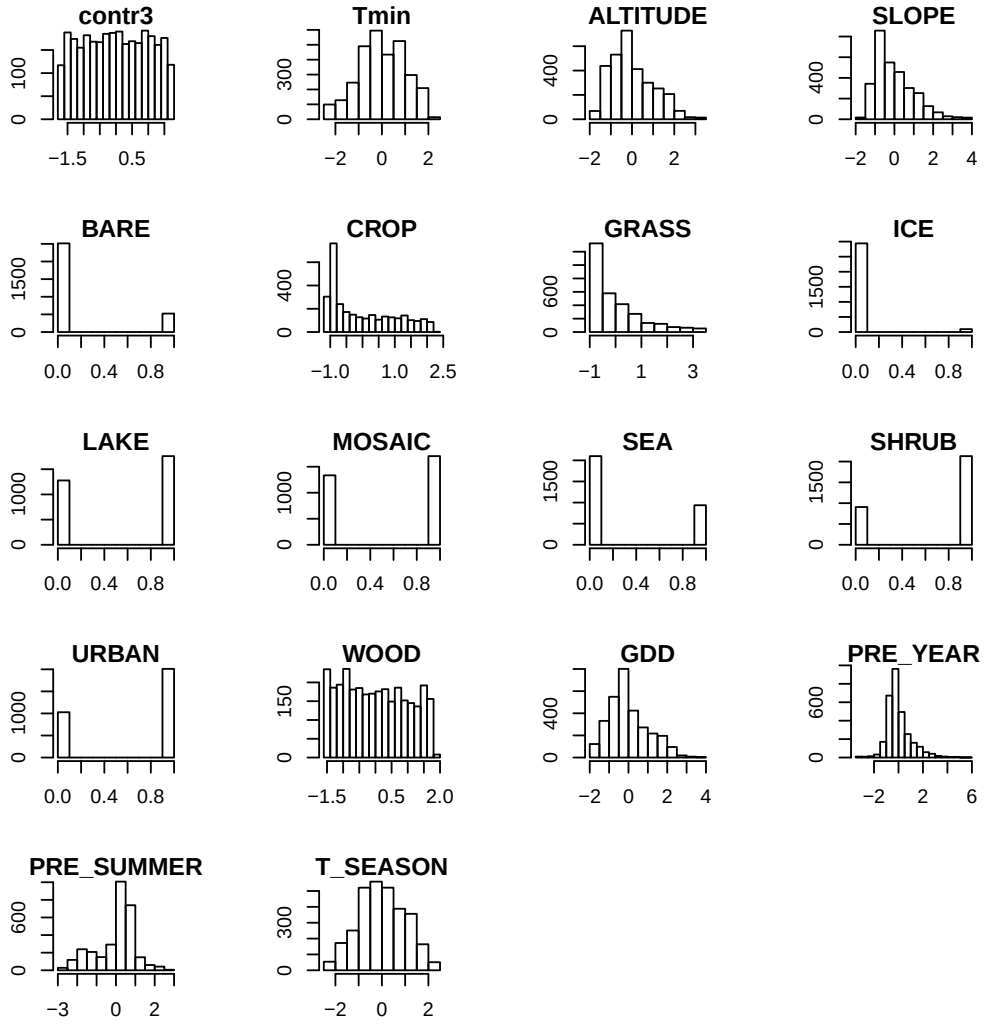

Abbildung 1: Histogramme der transformierten Prädiktoren

```
> #Packages
> library(psych)
> library(MuMIn)
> library(Hmisc)
```

```
[1] 0
```

```
[1] 523
```

2.2 Korrelierte Prädiktoren finden mit Hilfe der PCA und Cluster

Wenn zwei Prädiktoren miteinander korrelieren, sind die jeweiligen Schätzer instabil. Daher ist es sinnvoll sich auf einen der korrelierten Schätzer im Modell zu beschränken. Eine mögliche Herangehensweise ist durch eine Cluster

Analyse korrelierte Prädiktoren zu finden und mit diesen eine PCA (Hauptkomponentenanalyse) durchzuführen. Durch die Auswahl der wichtigen Hauptkomponenten bekommt man eine Dimensionsreduktion. Über die Ladungsmatrix können immer noch Rückschlüsse auf die wichtigen Prädiktoren gezogen werden. Die Clusteranalyse wurde nach Spearman berechnet (Abb. 3). Das Ergebnis der Clusteranalyse zeigt, dass die Steigung mit der Höhe korreliert. Die Steigung ist für das Habitat der Fransenfledermaus von größerer Bedeutung als die Höhe und wird darum in der weiteren Analyse verwendet.

```
> ##
> #PCA Berechnung
> ##
> pca <- prcomp(predi_trans[,-1], scale = F)
> names(pca$sdev) <- as.character(1:17) #Hinzufügen der Namen
> screeplot(pca, las=1, main = "", xlab="Hauptkomponenten") #Wie viel Varianz je PC erklärt werden kann
> biplot(pca) #welche PC korrelieren mit einander? PRE_SUMMER und WOOD  und GGD mit Tmin
> #Cluster Analyse mit transformierten Daten
> plot(varclus(~.,data= predi_trans[,-1], similarity = c("spearman")))
> abline(0.5, 0)
> ##
> #GLM mit relevanten PCA aufgeteilt jeweils errechnet für vier Gruppen, welche
> #die Clusteranalyse identifiziert hat
> ##
> clus_pca <- data.frame(1:nrow(predi_trans)) #Neuer Dataframe
> #Variabeln aus dem Cluster 1
> pca1 <- prcomp( ~ SHRUB + CROP + URBAN, data = predi_trans, scale = F)
> summary(pca1) #Die PC1 erklärt 74 prozent der Varianz

Importance of components%s:
                          PC1    PC2    PC3
Standard deviation     1.0321 0.4679 0.3862
Proportion of Variance 0.7432 0.1528 0.1041
Cumulative Proportion  0.7432 0.8959 1.0000

> new1 <- as.matrix(pca1$x) #nur die pca1 kommt für das GLM in Frage
> clus_pca$pca11 <- new1[,1]
> #cluster 2
> pca2<- prcomp( ~ PRE_YEAR + T_SEASON + SEA + ALTITUDE, data = predi_trans, scale = F)
> summary(pca2) #Die ersten beiden PC erklären 81 prozent der Varianz und kommen in die

Importance of components%s:
                          PC1    PC2    PC3     PC4
Standard deviation     1.2335 1.0434 0.7013 0.33361
Proportion of Variance 0.4735 0.3388 0.1531 0.03464
Cumulative Proportion  0.4735 0.8123 0.9654 1.00000

> #weitere Analyse
> new2 <- as.matrix(pca2$x)
> clus_pca$pca21<- new2[,1]
> clus_pca$pca22<- new2[,2]
```

```
> #cluster 3
> pca3 <- prcomp( ~ ICE + BARE + GRASS, data = predi_trans, scale = F)
> summary(pca3) #die erste PC erklärt schon 87 %

Importance of components%s:
                          PC1    PC2     PC3
Standard deviation     1.0112 0.3504 0.16725
Proportion of Variance 0.8715 0.1046 0.02384
Cumulative Proportion  0.8715 0.9762 1.00000

> new3 <- as.matrix(pca3$x)
> clus_pca$pca31<- new3[,1]
> clus_pca$art <- art
> #cluster 4
> pca4 <- prcomp( ~ WOOD + PRE_SUMMER + Tmin + GDD + LAKE + MOSAIC, data = predi_trans, scale = F)
> summary(pca4)#mit den ersten beiden PCA, welche 90 prozent der Varianz erklären wird im GLM

Importance of components%s:
                          PC1    PC2     PC3     PC4    PC5     PC6
Standard deviation     1.7335 0.8140 0.63032 0.41673 0.3890 0.31688
Proportion of Variance 0.6692 0.1476 0.08848 0.03868 0.0337 0.02236
Cumulative Proportion  0.6692 0.8168 0.90526 0.94394 0.9776 1.00000

> #weiter gerechnet
> new4 <- as.matrix(pca4$x)
> clus_pca$pca41<- new4[,1]
> clus_pca$pca42<- new4[,2]
> clus_pca$pca43<- new4[,3]
```

2.3 GLM

Ein GLM (generalized linear model) wurde mit den Daten der PCA gerechnet, die über 90 Prozent der Varianz erklären können und zuvor mit der Clusteranalyse in vier Gruppen unterteilt wurden. Ein zweites GLM wurde mit allen Daten berechnet, mit der ersten Interaktionsstufe. Durch die vielen Parameter würde das Modell keine gute Vorhersage machen können. Für die Vorhersage eines wärmeren Klimas sollte das Modell aber gut voraussagen können. Um dies zu erreichen wurde wie folgt vorgegangen: Durch stepwise- backwards wurden die am wenigst signifikanten Interaktionen herausgenommen. Dies wurde so lange wiederholt, bis 26 Prädiktoren übrig blieben. Im nächsten Schritt wurden mit der step Funktion in R, anhand des ACI weitere Prädiktoren heraus genommen.

```
> #
> #Modell mit Hilfe der PCA als erklärenden Variabeln
> #
> fmpca <- glm(art ~ (pca11 + pca21 + pca22 + pca31 + pca41 + pca42 +pca43)^2,
+              data = clus_pca, family = "binomial")
> fmpcastep <- step(fmpca)  #Vereinfachung mit Hife des AIC
> ##
> #GLM mit allen Parametern in erster Interaktion
```

```
> par(mfrow =c(1,2))
> screeplot(pca, las=1, main = "", xlab="Hauptkomponenten")
> biplot(pca)
```

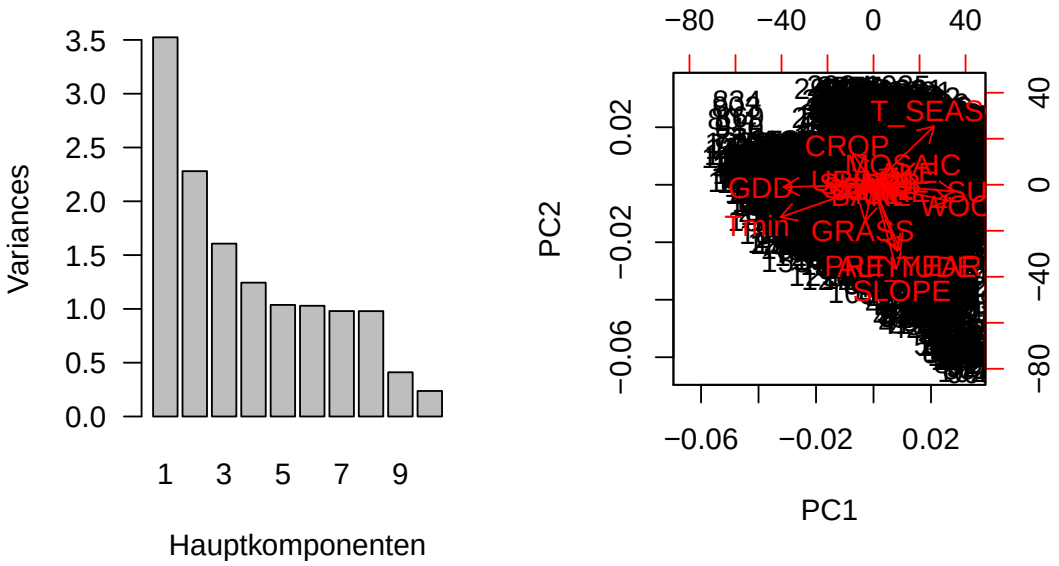

Abbildung 2: Links: Der Sceenplot zeigt, wieviel Varianz die einzelnen Hauptkomponenten erklären können. Rechts: Ein unübersichtlicher Biplot: die rote Achse entspricht den X-Werten der PCA und die schwarze Achse den Ladungen der einzelnen Prädiktoren

```
> par(mfrow =c(1,1))
> plot(varclus(~.,data= predi_trans[,-1], similarity = c("spearman")))
> abline(0.5, 0)
```

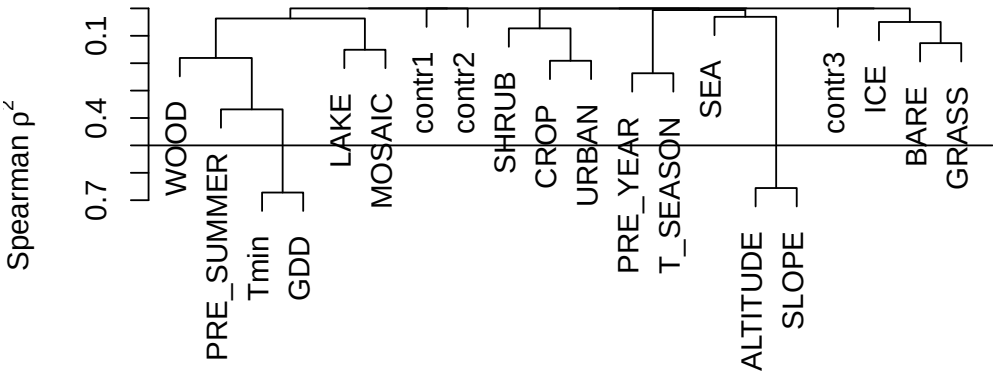

Abbildung 3: Clusteranalyse: GDD korreliert mit Tmin, sowie die Höhe mit der Steigung. Die waagrechte Linie ist der Schwellenwert von 0.5, welcher nicht überschritten werden sollte. Außerdem kann der Baum in vier nahezu unabhängige Gruppen geteilt werden

```
> ##
> fm1 <- glm(art ~ (GRASS + PRE_YEAR + T_SEASON + SEA + ICE + SLOPE + GDD + URBAN + BARE +
+                     SHRUB + WOOD + PRE_SUMMER + PRE_YEAR + CROP + LAKE + MOSAIC)^2, data =
+                     predi_trans,family = binomial)
> summary(fm1) #Sehr viele nicht Signifikanten Interaktionen
> #Herausnehmen der nicht signifikanten Interaktionen
> repeat{
+   neg_names <- names(which(coef(summary(fm1))[,4] > quantile(coef(summary(fm1))[,4],
+                                                              names = F)[4]))
+   #in neg_names werden die Namen gespeichert, welche einen Signifikantwert < 3/4
+   # aller Werte hat (willkürlich gewählt)
+   inter <- grepl('^[A-Za-z()_]+$', neg_names) # in inter werden die Datenpunkte, welche
+   #einen Doppelpunkt haben (also die Interaktionen mit F gespeichert)
+   neg_names <-  paste("-", neg_names[which(inter==F)], collapse="")
+   #Die neue Formula wird erstellt mit den Interaktionen
+   myformula <- paste0( ".~.", neg_names, collapse= "")
+   #Modell wird neu berechnet
+   fm2 <- update(fm1, as.formula(myformula))
+   #wenn es weniger als 20 Variabeln gibt oder die Signifikantswerte sehr klein sind, wird abgebrochen
+ if(length(as.numeric(fm1$coefficients))<20) break
+ fm1 <- fm2
+ #print(length(as.numeric(fm1$coefficients)))
+ }
> summary(fm2) #Viel weniger Werte
> fmst<- step(fm2) #jetzt kann mit der Step-Funktion noch ein
> #paar Hauptparameter heraus gelöscht werden zum Beispiel Eis
> summary(fmst)
```

2.4 Random Forest

Ein zweites Modell wurde mit dem Random Forest Algorithmus erstellt. Hierbei werden viele Bäume produziert nach dem Prinzip der rekursive Partitionierung. Ein großer Vorteil ist, dass die Interaktion zwischen den Prädiktoren automatisch mit einbezogen werden. Eine Verzweigung im Entscheidungsbaum finden dann statt, wenn die Varianz zwischen den Gruppen maximal ist, im Vergleich zur gesamt Varianz. Beim Random Forest werden viele Bäume erstellt, die sich immer weiter verbessern, da der nächste Baum durch die gewichteten Residuen vom vorigem Baum verbessert wird. Durch viele Wiederholungen entstehen somit immer bessere Modelle. Verglichen werden können diese durch Crossvalidierung oder Bootstrapping. Im Folgenden wird erst ein Baum errechnet und mit den wichtigen Variablen ein Modell erstellt. Dieses wird mit dem vollen Modell, welches wahrscheinlich überfittet ist, verglichen.

```
> ##
> #Orginal Daten mit den Kontroll-Zufallsvariablen
> ##
> art_predi_t <- as.data.frame(cbind(art_predi[,-c(1:6)], predi_trans$contr1,
+                     predi_trans$contr2, predi_trans$contr3, art))
> #Berechung eines einzelden Baumes um einflussreiche Prädiktoren auszumachen
> fit <- rpart(as.factor(art) ~ ICE + GRASS + PRE_YEAR + T_SEASON + SEA  + SLOPE
+              + CROP  + URBAN + BARE + SHRUB + WOOD + PRE_SUMMER + PRE_YEAR
```

```
+              + GDD + LAKE + MOSAIC, data = art_predi_t)
> #Welche Parameter sind besonders Einflussreich?
> ##
> #Random Forest mit allen Parametern
> ##
> rf_full <- randomForest(as.factor(art) ~ ICE  + GRASS + PRE_YEAR + T_SEASON
+                         + SEA  + SLOPE + CROP  + URBAN + BARE + SHRUB + WOOD + PRE_SUMMER
+                         + PRE_YEAR + GDD + LAKE + MOSAIC, data = art_predi_t, type="prob")
> #Fit nur mit den wichtigen Parametern
> rf_selc <- randomForest(as.factor(art) ~  GRASS  + T_SEASON  + CROP  + URBAN +
+                           BARE + SHRUB + WOOD + PRE_SUMMER + PRE_YEAR + GDD +
+                           LAKE, data = art_predi_t, type="prob")
> #Fit für die Partialplot Funktionen, welche allerdings nicht Korekt ist das sie eine Regression annimt
> rf_selc_part <- randomForest(art ~  + GRASS  + T_SEASON  + CROP  + URBAN +
+                                BARE + SHRUB + WOOD + PRE_SUMMER + PRE_YEAR +
+                                GDD + LAKE, data = art_predi_t)
> #Darstellung
> getwd()
> pdf("tree.pdf",width=9,height=7)
> plot(fit)
> text(fit, use.n = TRUE, xpd=T)
> dev.off()
> pdf("tree_imp.pdf",width=9,height=9)
> par(mfrow =c(2,2))
> plot(rf_selc, main="Select")
> plot(rf_full, main="Full")
> varImpPlot(rf_full, main="Modell full")
> varImpPlot(rf_selc, main="Modell full")
> dev.off()
```

2.5 GAM

```
> require(mgcv)
> require(effects)
> attach(predi_trans)
> ##
> #Idee: Die Einflussreichsten Prädiktoren aus dem Random Forest in das GAM
> # einbauen und mit der Sooth Funktion glätten. Durch bs="cs" den Shrinkage
> #  Algorithmus aktivieren um die Parameter Zahl zu minimieren. Inderaktionen
> #   einbauen, mit der Information aus dem backward stepwise GLM
> #   . Zur Kontrolle den Conditional plot vom GDD plotten mit den Konfidenzintervallen
> #
> ##
> #!!!bei kategorial Interaktionen, werden die y Werte mit
> #by angehängt ansonsten mit Komma
>
```

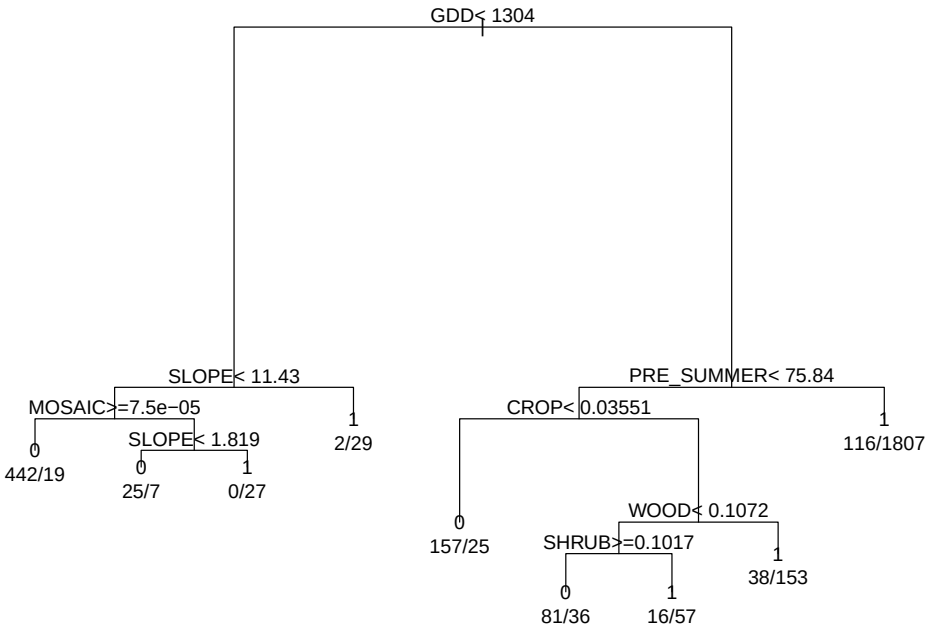

Abbildung 4: Klassifikation des gesamten Datensatzes. Zu sehen ist dass GDD mit Abstand die wichtigste Variabel ist. Gefolgt von SLOPE und PRESUMMER. Wenn PRESUMMER einen kleineren Wert von 75 hat ist eine Unterteilung nicht mehr weitere Unterteilung nicht mehr sinnvoll. In diesem Subset befinden sich 1807 also über die Hälfte aller Zellen in der die Fledermaus vorkommt.

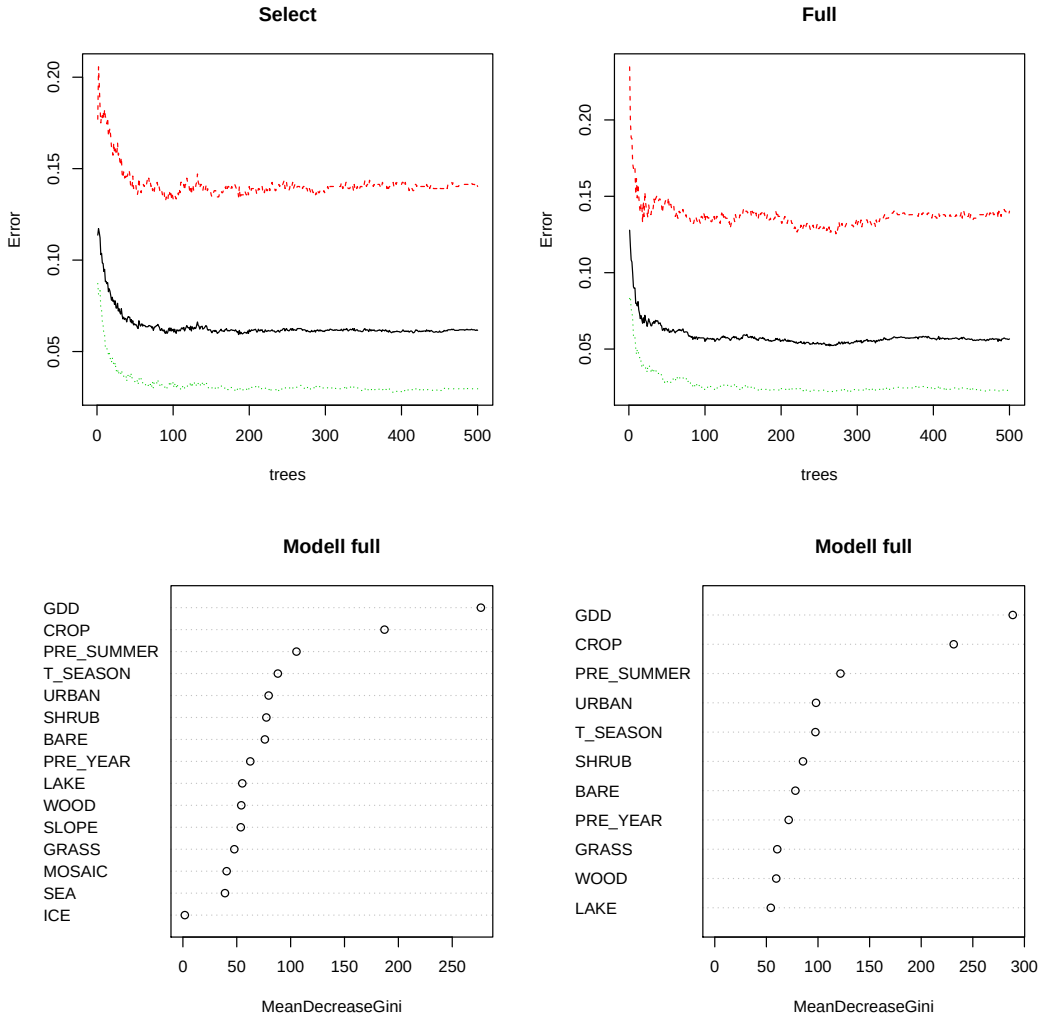

Abbildung 5: Die erste Zeile zeigt das einfache Modell und das komplette Modell. Die X-Achse sind die Bäume die benötigt werden um den Fehler auf der Y-Achse zu reduzieren. Nach ca 50 Bäumen gibt es kaum noch Verbesserung. In der zweiten Zeile sind die Parameter nach ihrer Erklärungsstärke sortiert. Ganz klar sind GDD und CROP die wichtigsten Parameter in beiden Modellen

```
> gam_fm <- gam(art ~ s(GDD, SLOPE) +  s(PRE_SUMMER, bs="cs") + s(CROP, bs="cs")
+               + SHRUB + s(PRE_YEAR, bs = "cs") + T_SEASON+ s(SLOPE, PRE_SUMMER),
+               data=predi_trans, family=binomial)
> summary(gam_fm)
> #GDD Predicten unter der Annahme das alle anderen Parameter auf dem Median liegen.
> newvalues <- data.frame("GDD"=seq(min(predi_trans$GDD), max(predi_trans$GDD),
+                                   len=100), t(apply(predi_trans, 2, median)))
> preds <- predict(gam_fm, newdata=newvalues, se.fit=T)
> #Funktion um die X-Achse retransformieren zu können
> zScore <- function(x, retrans = F, daten = NA) {
+   z <- c(1:length(x))
+   for (i in 1:length(x)){
+     if (retrans ==T) {
+       z[i] <- x[i]* sd(daten) + mean(daten)
+     }else{
+       z[i] <- (x[i] - mean(x))/sd(x)
+     }
+   }
+   return(z)
+ }
> #Darstellung auf transformierter X-Achse
> plot(gam_fm, trans=plogis)#Plot der Interaktionen
> pdf("gam_gdd.pdf")
> plot(zScore(newGDD, retrans = T, art_predi$GDD), plogis(preds$fit), type="l",
+      lwd=2, xlab="GDD", ylab="Fransen Fledermaus",
+      main="conditional plot mit dem GAM,
+      für die weiteren Variabeln wurde der Median verwendet", las=1)
> lines(zScore(newGDD, retrans = T, art_predi$GDD), plogis(preds$fit + 2*predsCond$se.fit), lwd=2, lty=2)
> lines(zScore(newGDD, retrans = T, art_predi$GDD), plogis(preds$fit - 2*predsCond$se.fit), lwd=2, lty=2)
> dev.off()
> summary(gam_fm)
>
```

2.6 Shrinkage

Beim Shrinkage wird auch versucht die Prädiktoren zu reduzieren, so dass eine Überfittung des Modelles gering bleibt. Anders als mit stepwise backwards aus dem GLM, wird beim Shrinkage versucht die Varianz in den Modellvorhersagen zu reduzieren, indem die Modellparameter in Richtung 0 gezwungen werden, wodurch sie möglicherweise ausgeschlossen werden. Dies geschieht nicht Stufenweise, sondern kontinuierlich. Mit der Methode Lasso wird versucht den Vorhersage - Fehler zu minimieren.

```
> library(biomod2)
> library(glmnet)
> #Alle möglichen Interaktionen ersten Grades und quadratische Terme in
> #das anfangs Modell mit einbezihen (Formelschreibweise)
> f2 <- makeFormula("art",   predi_trans[,-3],
```

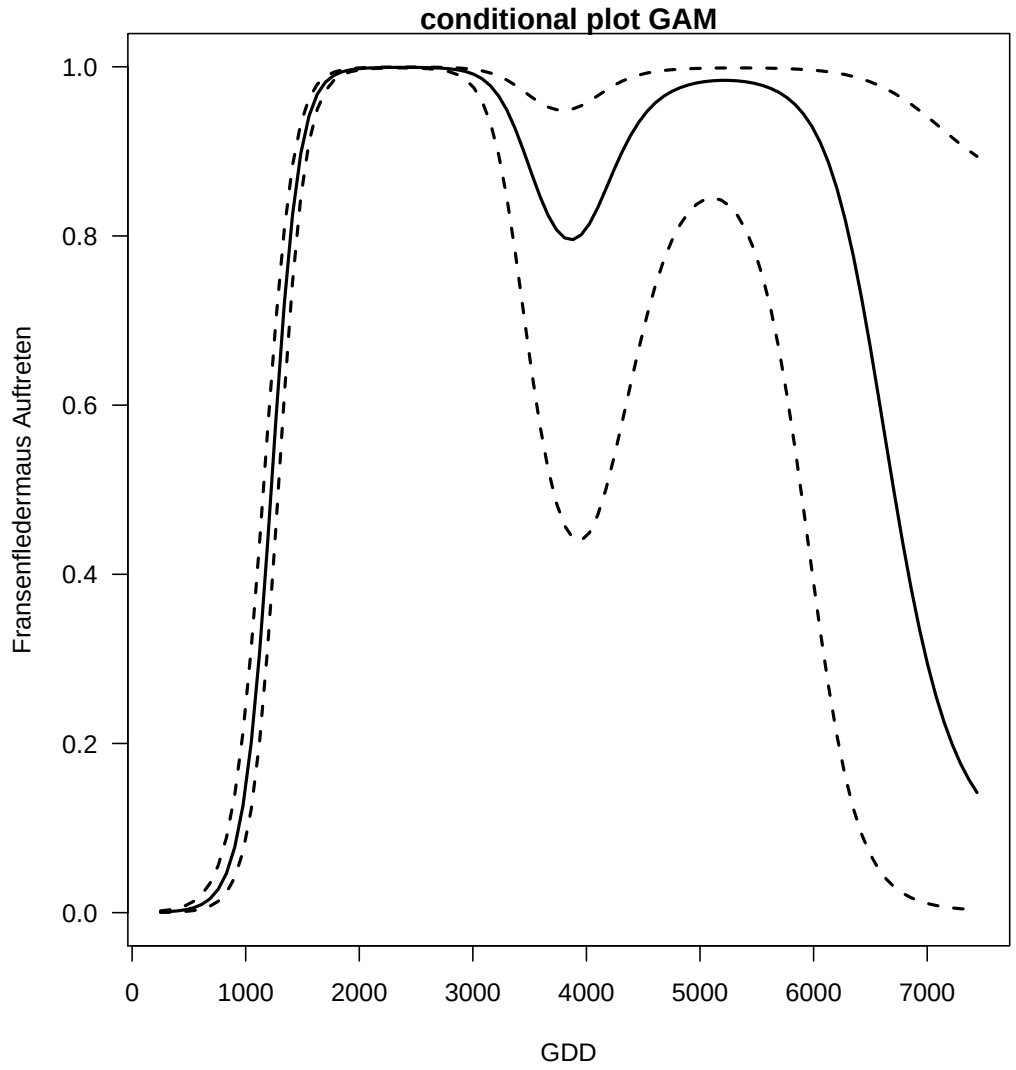

Abbildung 6: Konditional Plot für das GDD: Der Effekt von GDD, bei der Annahme, dass alle Prädiktoren auf dem Median liegen, ist zwischen 500 und 2000 GDD sehr hoch. Bei noch höherem GDD ist die Wahrscheinlichkeit mit einem hohen Fehler behaftet. Zusammengefasst kann also gesagt werden, dass es einen Effekt gibt dieser aber bei sehr hohen GDD nicht sicher vorhergesagt werden kann

```
+                   type = "quadratic", interaction.level = 1)
> ##
> #Shrinkage mit Methode lasso alpha =1
> ##
> #Modell Matrix erstellen ohne Intercept
> dataM2 <- model.matrix(f2, data=cbind.data.frame(predi_trans)[,-1])
> lasso.cv <- cv.glmnet(x=dataM2, y=art, alpha=1,
+                       family=c("binomial"),
+                       standardize=FALSE)
> #Modell berechnen mit den minimalen Lambda Werten
> lasso <- glmnet(x=dataM2, y=art, lambda=lasso.cv$lambda.min,
+                 alpha=1,
+                 family=c("binomial"), standardize=FALSE)
> #Koeffizenten bei den der Lamda-Wert am geringsten ist
> # anschauen (sind viele)
> tmp_coeffs <- coef(lasso.cv, s = "lambda.min")
> #Modell berechnen mit allen Variabeln
> # Lambda-Werten für die graphische Darstellung
> lasso.full <- glmnet(x=dataM2, y=art,family=c("binomial"))
> pdf("shrik.pdf")
> par(mfrow =c(1,2))
> plot(lasso.cv, las=1)
> plot(lasso.full, "lambda", las=1)
> dev.off()
> summary(lasso)
```

2.7 Crossvalidation

Durch die Crossvalidierung werden die Daten in zehn etwa gleich große Gruppen geteilt. In diesen Gruppen sollte das Verhältnis von Null- und Eins-werten der Antwortvariabel ausgeglichen sein. Im nächsten Schritt wird das Modell mit Werten aus neun Gruppen berechnet und eine Vorhersage auf die Werte der zehnten Gruppe gemacht. Dies wird für alle Gruppen durchgeführt. Anhand von gemittelten Werten des Root Mean Square Error (RMSE,) dem R2 und der LogLikelihood können Modelle miteinander verglichen werden. Ziel dieses Kapitels ist es ein Randomforest und ein Parametrisches Modell, aus den bis hierhin entwickelten Modellen, für die weitere Analyse auszuwählen. In der Abbildung 8 und 9 ist deutlich zuerkennen, dass das Shrinkage-Modell mit Abstand am besten ist. Leider habe ich es nicht geschafft für dieses Modell einen Marginal-Effekt-Plot zu erstellen und Bootstrapping durchzuführen. Dadurch habe ich das schlechter back ward stepwise Modell (fmst) für die weiter Analyse verwendet. Von den Random Forest Modellen wurde das mit den wenigeren Parametern verwendet (rfselc), da es einen besseren RMSE Wert hat. Das Hauptauswahlkriterium war der RMSE Wert, welcher ein Mass für den Bias und der Varianz ist.

```
> ##
> #Zwei Funktionen für die Crossvalidierung:
> ##
> #1.Likelihood-Funktion
> ell <- function(test, predsCV)  sum(dbinom(test, size=1, prob=predsCV, log=T), na.rm=T)
> #Crossvalidierung für GLM GAM GLMNET und RandomForest mit 10 subsampel
```

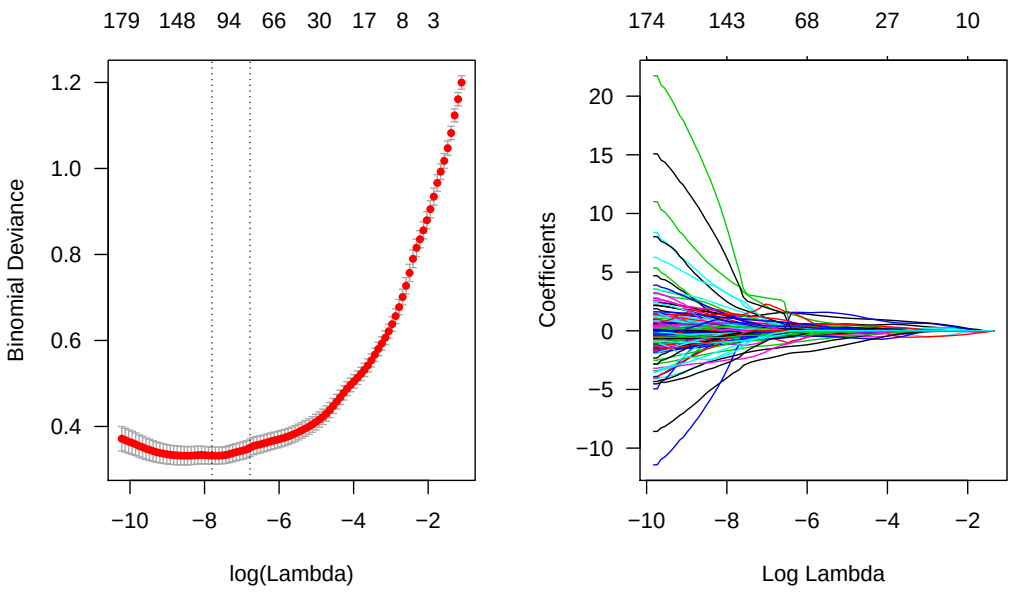

Abbildung 7: Die linke Graphik zeigt wie sich die Deviance gegenüber den reduzierten Parameter (Lambda) verhält. Gesucht wird das Lamda mit der minimalsten Deviance. Diese Stelle ist mit den vertikalen Linien dargestellt. In der rechten Graphik sind die Koeffizienten, welche immer weiter reduziert dargestellt. Wenn eine Koeffizient die Null-Linie erreicht ist er ganz aus dem Modell draußen. Bei -7.802476 log(Lambda) ist das Modell für eine Vorhersage perfekt

```
> cross_val <- function(k, model, antw_var, daten, fun=2){
+   #Aufteilung in gleich Gruppen mit jeweils gleich vielen 0 und 1 Werten
+   zero <- which(antw_var==0)
+   ones <- which(antw_var==1)
+   ind <- 1
+   ind[zero] <- sample(10, length(zero), replace = T)
+   ind[ones] <- sample(10, length(ones), replace = T)
+   #Erstellung von Output Matrix
+   crossmat <- as.data.frame(matrix(NA, k, 3))
+   colnames(crossmat) <- c("RMSE", "R2", "ll")
+   for(i in 1:k){
+     if(fun==4){
+       #Random Forest
+       ##Berechnung von 9 Gruppen
+       fm_forest <- randomForest(formula(model)[1:3], data = daten[!(ind==i),], type="prob")
+       #Vorhersage für die fehlende Gruppe
+       preds <- predict(fm_forest, data = daten[ind==i,], type="prob")[2]
+
+       rmse <- sqrt(mean((art[ind==i] - preds)^2))
+       r2 <- NA
+       ll <- ell(art[ind == i], predict(lasso, newx=daten[ind==i,]))
+     }else{
+       if(fun==1){
+         #Für glmnet
```

```
+       lasso.cv <- cv.glmnet(x=daten[!(ind==i),], y=art[!(ind==i)],
+                             alpha=1, family=c("binomial"), standardize=FALSE)
+       lasso <- glmnet(x= daten[!(ind==i),], y=art[!(ind==i)],
+                       lambda=lasso.cv$lambda.min, alpha=1, family=c("binomial"),
+                       standardize=FALSE)
+       rmse <- sqrt(mean((art[ind==i]- predict(lasso, newx=daten[ind==i,],
+                                               type="response"))^2))
+       r2 <- cor(predict(lasso, newx=daten[ind==i,]), art[ind==i])^2
+       ll <- ell(art[ind == i], predict(lasso, newx=daten[ind==i,]))
+     }else{
+     if(fun==2){
+     #für GLM
+     fm_cv <- glm(formula(model), data = daten[!(ind==i),], family = binomial)
+     }
+     if(fun==3){
+     #für GAM
+     fm_cv <- gam(formula(model), data = daten[!(ind==i),], family = binomial)
+     }
+     preds <- predict(fm_cv, newdata = daten[ind==i,], type="response")
+     rmse <- sqrt(mean((art[ind==i] - preds)^2))
+     r2 <- cor(preds, art[ind==i])^2
+     ll <- ell(art[ind == i], preds)
+     }
+   }
+     crossmat[i, ] <- c(rmse, r2, ll)
+   }
+   return(crossmat)
+ }
> ##
> #Für das Shrinkage Modell braucht es viel rechen Zeit, d.h.
> #wurde die Funktion auf 3 Cores ausgelagert und dann als csv. abgespeichert
> ##
> nCores <- detectCores()
> cl <- makeCluster(nCores - 1)
> system.time(rpartBScluster <- mclapply(10, NA, art, dataM2, fun=1,
+                                        FUN=cross_val, mc.cores=nCores))
> # approx. twice the speed (due to overhead)
> write.table(rpartBScluster, "/home/gabriel/Dokumente/Umwelt_statistik/shrink_rms.csv")
> ##
> #Anwendung der Funktion auf alle weiteren  Modelle
> ##
> #1. Erstellen eines Arrays mit 10 Zeilen (Gruppen), 6 Spalten (Modelle) und
> #drei Ebenen
> box_arr<- array(NA,dim = c(10,6,3)) #Zehn Crossvalidierngen fünf Modelle und
> #2. Einsortierung der Werte #Braucht etwa 10 min
```

```
> par(mfrow =c(1,2))
> new_arr <- readRDS(file="/home/gabriel/Dokumente/Umwelt_statistik/new_arr.Rda")
> boxplot(new_arr[,,2], names = colnames(new_arr), main ="R^2")
> boxplot(new_arr[,,1], names = colnames(new_arr), main="RMSE")
```

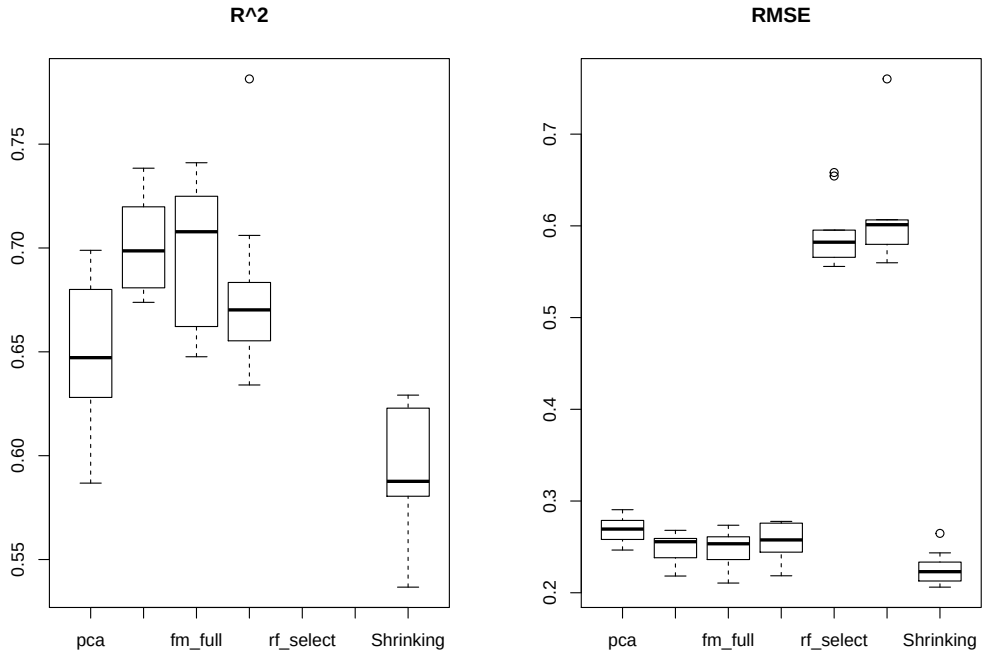

Abbildung 8: Die Modelle sind in der folgenden Reihenfolge abgebildet: PCA, fm.step, fm.full, fm.gam, rf.select, rf.full, Shrinkage

```
> for(i in 1:3){
+ box_arr[,,i] <- cbind(cross_val(10, fmpcastep, art, clus_pca)[,i],
+                        cross_val(10, fmst, art, predi_trans)[,i],
+                        cross_val(10, fm1, art, predi_trans)[,i],
+                        cross_val(10, gam_fm, art, predi_trans, fun=3)[,i],
+                        cross_val(10, rf_selc, art, art_predi_t, fun=4)[,i],
+                        cross_val(10, rf_full, art, art_predi_t, fun=4)[,i])
+ print(i)
+ }
> #3.Laden der Shrinkage Werte
> altmat<-read.table("/home/gabriel/Dokumente/Umwelt_statistik/shrink_rms.csv")
> #4.Einsorteirung der Shrinkage Werte
> new_arr<-array(c(box_arr[,,1],altmat[,1], box_arr[,,2],altmat[,2],
+                  box_arr[,,3],altmat[,3]), dim=c(10,7,3))
> colnames(new_arr)<- c("pca", "fm_step", "fm_full",
+                       "fm_gam", "rf_select", "rf_full", "Shrinkage")
> #Abspeichern der Daten, damit sie direkt in R-Sweave dargestellt werden können
> saveRDS(new_arr, "/home/gabriel/Dokumente/Umwelt_statistik/new_arr.Rda")
```

```
> par(mfrow =c(1,2))
> boxplot(new_arr[,,3], names = colnames(new_arr), main ="LogLike")
```

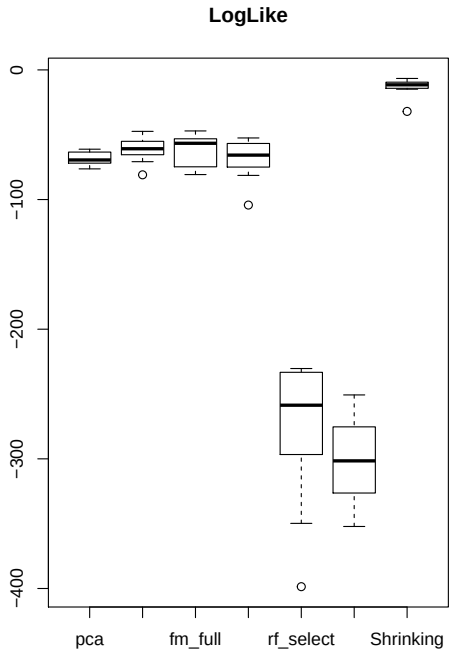

Abbildung 9: Die Modelle sind in der folgenden Reihenfolge abgebildet: PCA, fm.step, fm.full, fm.gam, rf.select, rf.full, Shrinkage

2.8 Marginal-Effekt-Plots von RandomForest und GLM

Das GDD hat im Random-Forest-Modell einen starken Effekt auf die Abundanz der Fransenfledermaus. Dies ist aus der Tabelle vom GLM nicht ersichtlich. Im GLM wird der Haupteffekt überlagert durch die beteiligten Interaktionen. In der Tabelle 1 ist zum Beispiel zu sehen, dass SEA:GDD und MOSAIK:GDD eine hoch signifikante Interaktionen darstellen. Die Effekte sind in den Abbildungen **??** und 11 beschrieben.

```
> ##
> # Marginal Plot mit den Convidenzinterfallen aus dem Bootstrapp für GDD
> #  mit dem randomForest Modell
> # #
> #1.neuer Array mit Zeilden der Datenpunkte Spalten, des GDD-Wertes und Ebenden für die bootstarps
> rf.pred.array <- array(dim=c(nrow(art_predi_t), 100, 25))
> #GDD-Werte
> newageseq <- seq(min(art_predi_t$GDD), max(art_predi_t$GDD), len=100)
> ## 2. Forschleife um die Bootstraps einzusortieren
> for (i in 1:25){
+   data.bs <-  art_predi_t[sample(nrow(art_predi_t), replace=T),] #neue Datenpunkte
+   rf.bs <- randomForest(as.factor(art) ~  GRASS  + T_SEASON  + CROP  + URBAN
+                         + BARE + SHRUB + WOOD + PRE_SUMMER
+                         + PRE_YEAR + GDD + LAKE, data = art_predi_t, type="prob")
+   # 3. erstellen von neuem marginal dataset (repeating 100 times GDD for each data point)
+   newDatsMargbs <- data.bs[rep(seq_len(nrow(data.bs)), length(newageseq)),]
+   # neues Datenset, 100 mal wiederholt
+   newDatsMargbs$GDD <- rep(newageseq, each = nrow(data.bs))
+   #head(newDatsMargbs)
+   # 4. predict das randomForest modell mit dem neuen Marginal Dataset
+   # (Ergebnis = Wahrscheinlichkeit)
+   preds.bs <- predict(rf.bs, newdata=newDatsMargbs, type="prob")[,2]
+   pred.mat.bs <- matrix(preds.bs, nrow=nrow(art_predi_t), ncol=100, byrow=F)
+   # byrow: wie werden die Daten einsortiert?
+   # 5. Im Array speichern
+   rf.pred.array[,,i] <- pred.mat.bs
+   print(paste("Bootstrap", i, "completed."))
+   #löschen um zuvermeiden, dass beim Fehler die alten Daten gespeichert werden
+   rm(rf.bs, newDatsMargbs, preds.bs, pred.mat.bs)
+ }
> # summarise array over dimensions 1 and 3 by computing 0.025, 0.5 and 0.975 quantile
> pred.age.mat <- apply(rf.pred.array, 2, quantile, c(0.025, 0.25, 0.5, 0.75, 0.975))
> # Marginal Plot mit Quantilen errechnet aus den Bootstrappsd
> matplot(newageseq, t(pred.age.mat), type="l", lwd=2, lty=c(3,2,1,2,3),
+         col="grey20", las=1, xlab="GDD", ylab="Myotis nattereri",
+         main="marginal plot des GDD Effekt")
> legend("topright", lty=c(1,2,3), lwd=2, col="grey20",
+        legend=c("median", "50% CI", "95% CI"), bty="n", cex=1.5)
> #####
> #Das gleiche für CROP
```

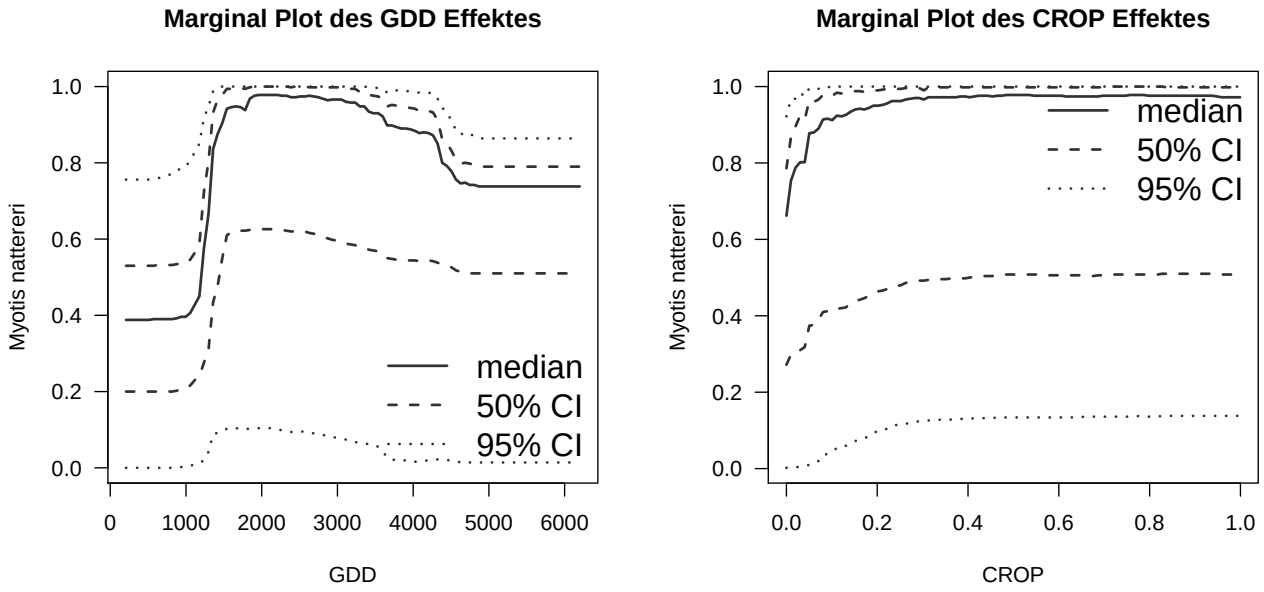

Abbildung 10: Der Effekt von den beiden Einflussreichsten Parametern auf das Vorkommen der Fransenfledermaus. Beim GDD ist der Effekt zwischen den Werten 1000 und 2000 sehr deutlich. Bei CROPE ist der Effekt nicht so eindeutig. Zwischen einem Anteil von 0 und 20 % von CROP in einer Zelle, steigt die Fransenfledermaus Abundanz an.

m1)

```
> ####
> rf.pred.array <- array(dim=c(nrow(art_predi_t), 100, 25))
> newageseq <- seq(min(art_predi_t$CROP), max(art_predi_t$CROP), len=100)
> for (i in 1:25){
+   data.bs <-  art_predi_t[sample(nrow(art_predi_t), replace=T),]
+   rf.bs <- randomForest(as.factor(art) ~  GRASS  + T_SEASON  + GDD  + URBAN
+                         + BARE + SHRUB + WOOD + PRE_SUMMER
+                         + PRE_YEAR + CROP + LAKE, data = art_predi_t, type="prob")
+   newDatsMargbs <- data.bs[rep(seq_len(nrow(data.bs)), length(newageseq)),]
+   newDatsMargbs$CROP <- rep(newageseq, each = nrow(data.bs))
+   preds.bs <- predict(rf.bs, newdata=newDatsMargbs, type="prob")[,2]
+   pred.mat.bs <- matrix(preds.bs, nrow=nrow(art_predi_t), ncol=100, byrow=F)
+   rf.pred.array[,,i] <- pred.mat.bs
+   print(paste("Bootstrap", i, "completed."))
+   rm(rf.bs, newDatsMargbs, preds.bs, pred.mat.bs)
+ }
> pred.age.mat <- apply(rf.pred.array, 2, quantile, c(0.025, 0.25, 0.5, 0.75, 0.975))
> pdf("CROPrf",height = 5, width=5)
> matplot(newageseq, t(pred.age.mat), type="l", lwd=2, lty=c(3,2,1,2,3), col="grey20",
+         las=1, xlab="GDD", ylab="Myotis nattereri", main="Marginal Plot des CROP Effektes")
> legend("topright", lty=c(1,2,3), lwd=2, col="grey20",
+        legend=c("median", "50% CI", "95% CI"), bty="n", cex=1.5)
> dev.off()
```

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	5.3462	0.3907	13.68	0.0000
GRASS	0.3707	0.1353	2.74	0.0061
PRE_YEAR	-0.1097	0.1946	-0.56	0.5730
T_SEASON	-0.5635	0.1582	-3.56	0.0004
SEA	-1.1811	0.2563	-4.61	0.0000
SLOPE	-1.2399	0.1439	-8.62	0.0000
GDD	1.9316	0.3098	6.24	0.0000
URBAN	1.3022	0.1780	7.31	0.0000
BARE	-0.5069	0.2381	-2.13	0.0333
WOOD	0.9819	0.1433	6.85	0.0000
PRE_SUMMER	4.9684	0.3787	13.12	0.0000
CROP	1.3080	0.1573	8.31	0.0000
LAKE	-1.0474	0.2747	-3.81	0.0001
MOSAIC	-2.0536	0.2767	-7.42	0.0000
GRASS:PRE_SUMMER	0.6412	0.1042	6.15	0.0000
PRE_YEAR:T_SEASON	0.9909	0.1218	8.13	0.0000
PRE_YEAR:SLOPE	0.7489	0.1093	6.85	0.0000
T_SEASON:PRE_SUMMER	-0.9778	0.1624	-6.02	0.0000
SEA:GDD	-2.5904	0.3845	-6.74	0.0000
SEA:PRE_SUMMER	-3.5290	0.4409	-8.00	0.0000
SLOPE:GDD	-0.9729	0.1723	-5.65	0.0000
SLOPE:PRE_SUMMER	-1.4979	0.1851	-8.09	0.0000
GDD:MOSAIC	2.7778	0.2308	12.04	0.0000
PRE_SUMMER:LAKE	-1.1297	0.1947	-5.80	0.0000

Tabelle 1: Summary-Ausgabe von dem GLM-Step-Modell

```
> #Marginalplot mit bootsrapping gleiche Prinzip wie beim randomForest
> #für GDD
> rf.pred.array <- array(dim=c(nrow(predi_trans), 100, 25))
> newageseq <- seq(min(predi_trans$GDD), max(predi_trans$GDD), len=100)
> for (i in 1:25){
+   data.bs <-  predi_trans[sample(nrow(predi_trans), replace=T),]
+   fmst.bs <- glm(formula(fmst), data = predi_trans, family = binomial)
+   newDatsMargbs <- data.bs[rep(seq_len(nrow(data.bs)), length(newageseq)),]
+   newDatsMargbs$GDD <- rep(newageseq, each = nrow(data.bs))
+   preds.bs <- predict(fmst.bs, newdata=newDatsMargbs)
+   pred.mat.bs <- matrix(preds.bs, nrow=nrow(predi_trans), ncol=100, byrow=F)
+   rf.pred.array[,,i] <- pred.mat.bs
+   print(paste("Bootstrap", i, "completed."))
+   rm(rf.bs, newDatsMargbs, preds.bs, pred.mat.bs)
+ }
> pred.age.mat <- apply(rf.pred.array, 2, quantile, c(0.025, 0.25, 0.5, 0.75, 0.975))
> pdf("GDDglm.pdf",height = 5, width=5)
> matplot(zScore(newageseq,retrans = T, daten = predi_trans$GDD),
+         plogis(t(pred.age.mat)), type="l", lwd=2, lty=c(3,2,1,2,3),
+         col="grey20", las=1, xlab="GDD", ylab="Myotis nattereri",
+         main="Marginal Plot des GDD Effektes im GLM-Step")
> legend("bottomright", lty=c(1,2,3), lwd=2, col="grey20",
```

```
+        legend=c("median", "50% CI", "95% CI"), bty="n", cex=1.5)
> dev.off()
> summary(fmst)
> ##
> #Same für CROP
> ##
> rf.pred.array <- array(dim=c(nrow(predi_trans), 100, 25))
> newageseq <- seq(min(predi_trans$CROP), max(predi_trans$CROP), len=100)
> for (i in 1:25){
+   data.bs <-  predi_trans[sample(nrow(predi_trans), replace=T),]
+   fmst.bs <- glm(formula(fmst), data = predi_trans, family = binomial)
+   newDatsMargbs <- data.bs[rep(seq_len(nrow(data.bs)), length(newageseq)),]
+   newDatsMargbs$CROP <- rep(newageseq, each = nrow(data.bs))
+   preds.bs <- predict(fmst.bs, newdata=newDatsMargbs)
+   pred.mat.bs <- matrix(preds.bs, nrow=nrow(predi_trans), ncol=100, byrow=F)
+   rf.pred.array[,,i] <- pred.mat.bs
+   print(paste("Bootstrap", i, "completed."))
+   rm(rf.bs, newDatsMargbs, preds.bs, pred.mat.bs)
+ }
> pred.age.mat <- apply(rf.pred.array, 2, quantile, c(0.025, 0.25, 0.5, 0.75, 0.975))
> pdf("CROPglm.pdf",height = 5, width=5)
> matplot(zScore(newageseq,retrans = T, daten = predi_trans$CROP),
+         plogis(t(pred.age.mat)), type="l", lwd=2, lty=c(3,2,1,2,3),
+         col="grey20", las=1, xlab="CROP", ylab="Myotis nattereri",
+         main="Marginal Plot des CROP Effektes im GLM-Step")
> legend("bottomright", lty=c(1,2,3), lwd=2, col="grey20", legend=c("median", "50% CI", "95% CI"), bty="n"
> dev.off()
> ##
> #Vergleich mit Conditional Plot und Ausgabe von se.fit beim GLM
> ##
> #Darstellung vom besten Modell GLM das gleiche auch für
> # CROP (Code nicht abgelbildet)
> newGDD <- seq(min(predi_trans$GDD), max(predi_trans$GDD), len=100)
> newDaten <- data.frame("GDD"= newGDD, t(apply(predi_trans, 2, median)))
> predsCond <- predict(fmst, newdata = newDaten, se.fit=T)
> plot(zScore(newGDD, retrans = T, art_predi$GDD), plogis(predsCond$fit),
+      type="l", lwd=2, xlab="GDD", ylab="Fransen Fledermaus",
+      main="conditional plot GDD, für die weiteren Variabeln
+      wurde der Median verwendet", las=1)
> lines(zScore(newGDD, retrans = T, art_predi$GDD),
+       plogis(predsCond$fit + 2*predsCond$se.fit), lwd=2, lty=2)
> lines(zScore(newGDD, retrans = T, art_predi$GDD),
+       plogis(predsCond$fit - 2*predsCond$se.fit), lwd=2, lty=2)

> ##
> #Erstellen von neuem Datensatz und GDD dabei um 20% erhöhen
```

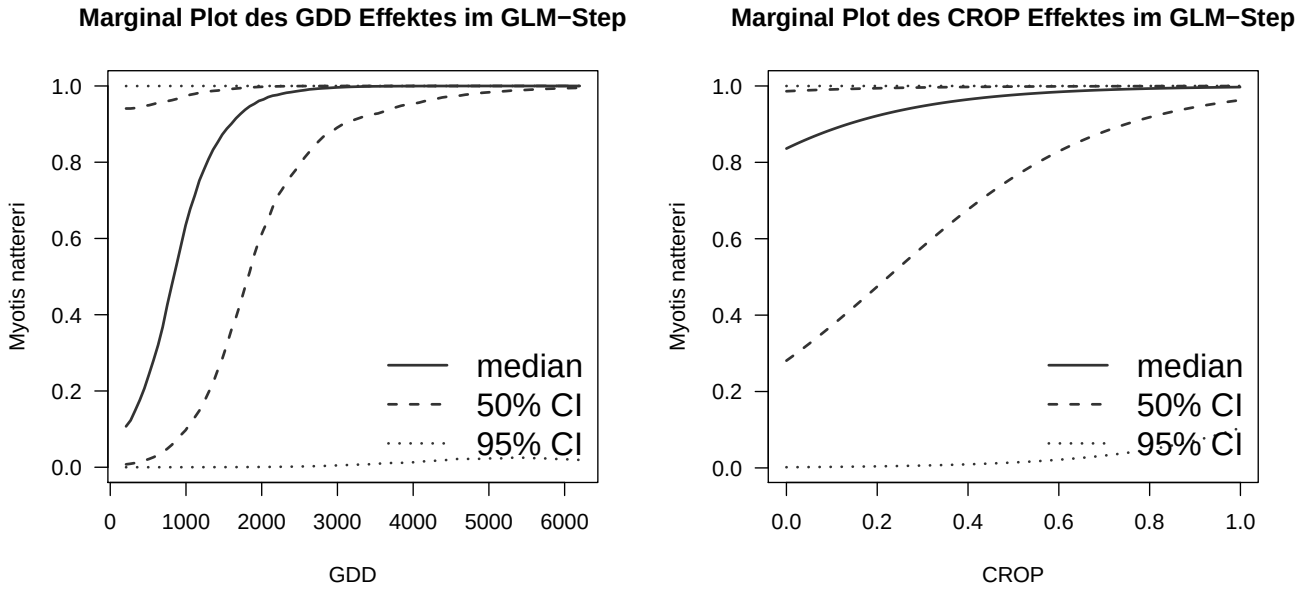

Abbildung 11: Marginalplot GLM: Durch 25-maliges Bootstrapping wurde der Standardfehler für jeden X-Wert als Konfidenzintervall dargestellt. Es wurden keine Annahmen der weiteren Parametern getroffen. Der Effekt durch das GDD auf die Fransenfledermaus Abundanz ist deutlich. Das 95 % Konvidenzinterfall zeigt dagegen keinen Effekt mehr. Zusammengefast: Zwischen einem GDD-Wert von 0 und 2000 haben wir einen deutlichen Effekt auf vom GDD auf das Auftreten der Fransenfledermaus. Bei CROP ist der Effect nicht ganz so eindeutig.

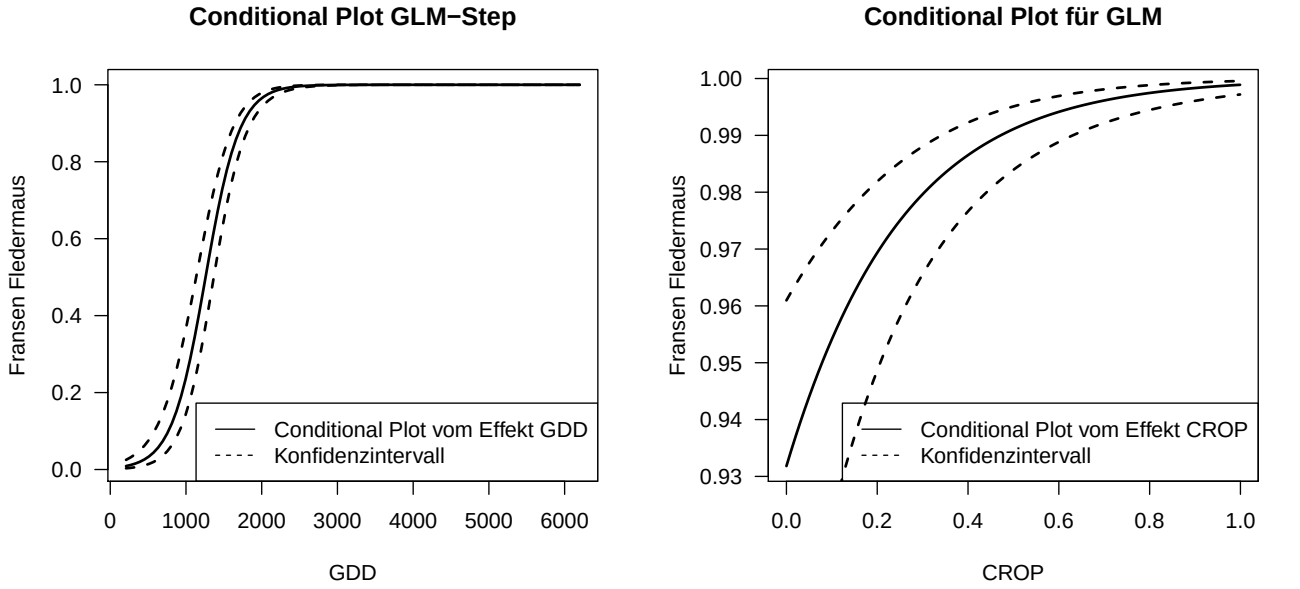

Abbildung 12: Der Conditionalplot vom GDD und CROP unter der Annahme, dass alle Parameter auf ihren Median liegen. Zu sehen ist im Gegensatz zudem Marginalplot eine sehr sichere Vorhersage und einen eindeutigen Effekt vom GDD auf die Abundanz der Fransenfledermaus. Mit steigenden CROP steigt die Wahrscheinlichkeit über das Auftreten der Fransenfledermaus. Auch hier sind die Fehlerbalken viel kleiner als die beim Marginalplot. Anmerkung: Das Konfidenzintervall errechnet sich in diesem Fall aus der Standardfehler und nicht aus den Bootstrapping Daten, sowie es im Marginalplot der Fall ist

```
> ##
> predi_trans02 <- predi_trans
> art_predi$GDD02 <- art_predi$GDD * 1.2
> #Standardisieren mit der Standard- Abweichung und dem Mittelwert vom orginal GDD
> predi_trans02$GDD <- ((art_predi$GDD*1.2) -
+                         mean(art_predi$GDD))/sd(art_predi$GDD)
```

2.9 Verteilung Fitting

Es wurden die Vorhersage-Werte mit Hilfe der Maximum-Likelihood-Methode an eine Passende Verteilung gefittet. Die Normalverteilung ist nur für die Linksakale geeinget da die Werte Gamma transoformiert wurden. Auf der Response-Skala wurde die Beta Verteilung angewendet. Das Ergebniss ist in der Abbildung 13 zu sehen.

```
> #Verteilung Fitten an den GDD + 20 %------------------
> prdicted_value <-predict(fmst, data = predi_trans02, type="response")
> link_value <-predict(fmst, data = predi_trans02)
> str(prdicted_value)
> #Rücktransformierung
> distd <-zScore(prdicted_value, retrans = T, daten = art_predi$GDD)
> library(MASS)
> par(mfrow=c(2,2), mar=c(4,5,4,0.5))
> ##
> #Logliklyhood Ansatz um an die Verteilungen zu fitten
> ##
> pdf("logfit.pdf")
> fitnorm <- fitdistr(distd, "normal")
> plot(density(distd), main="Normalverteilung")
> curve(dnorm(x, fitnorm$estimate[1],fitnorm$estimate[2]), add=T, col="red")
> fitnorm <- fitdistr(link_value , "normal")
> plot(density(link_value ), main="Normalverteilung Link")
> curve(dnorm(x, fitnorm$estimate[1],fitnorm$estimate[2]), add=T, col="blue")
> fitbeta <- fitdistr(prdicted_value, list(shape1=1, shape2=2), densfun="beta")
> plot(density(prdicted_value), main="Betaverteilung")
> curve(dbeta(x, fitbeta$estimate[1] , fitbeta$estimate[2]), add=T, col ="red")
> dev.off()
> #Abbildung 13
```

2.10 Residuen Analyse

Die Analyse der Residuen soll zeigen, ob diese normal verteilt sind, um die Unabhängigkeit der Datenpunkte sicherzustellen und dadurch zu klären, ob alle wichtigen Effekte im Modell mit einbezogen wurden. Das Ergebnis ist in der Abbildung 14 zusehen. Es sieht so aus, als ob die Residuen normal verteilt sind und somit keine Struktur aufweisen. In der Karte (Abbildung **??**), sind dagegen die Residuen in den Bereichen am höchsten, bei denen die Wahrscheinlichkeit weit von 0 und 1 entfernt ist . Dies ist in den Randgebieten des Ausbreitungsgebietes von der Fransenfledermaus. Hier kann das Modell keine eindeutige null oder eins Vorhersage machen. Das kann mit der Eigenschaft der Binomialverteilung erklärt werden, welche nur zwischen Werte von null und eins gültig ist.

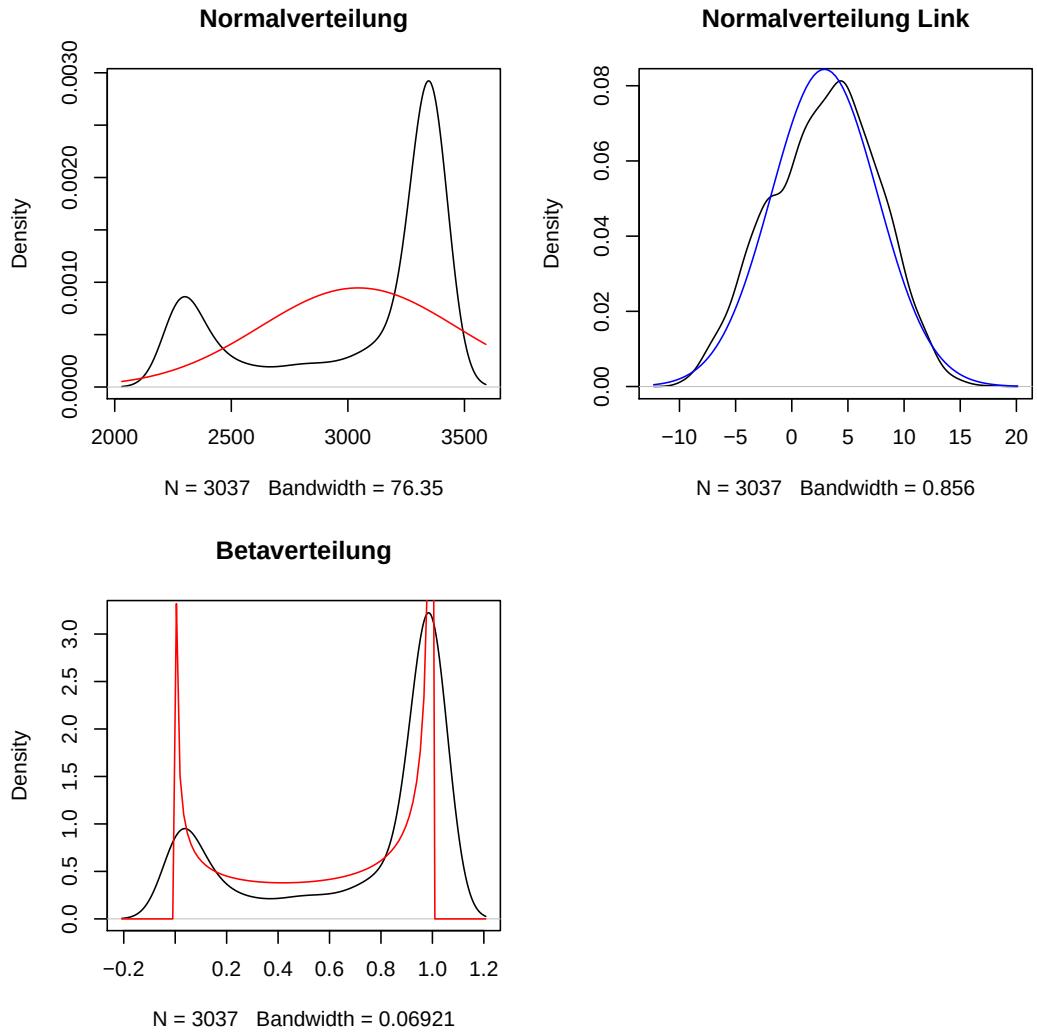

Abbildung 13: Der erste Plot zeigt die Normalverteilung (rot) gefittet an die Daten auf der Response-Skale. Der Fit ist eindeutig nicht befriedigend. In der zweiten Abbildung ist die Normalverteilung (blau) an die Daten auf der Link-Skale gefittet. Dies sieht schon besser aus. Der dritte Plot zeigt die Beta-Verteilung (rot) gefittet an die Daten auf der standardisierten Response-Skale. Dies ist zwischen den Werten null und eins ansatzweise geglückt, außerhalb diese Bereiches ist der Fit sehr schlecht. Das beste Ergebnis ist also die Normalverteilung auf der Link-Skale

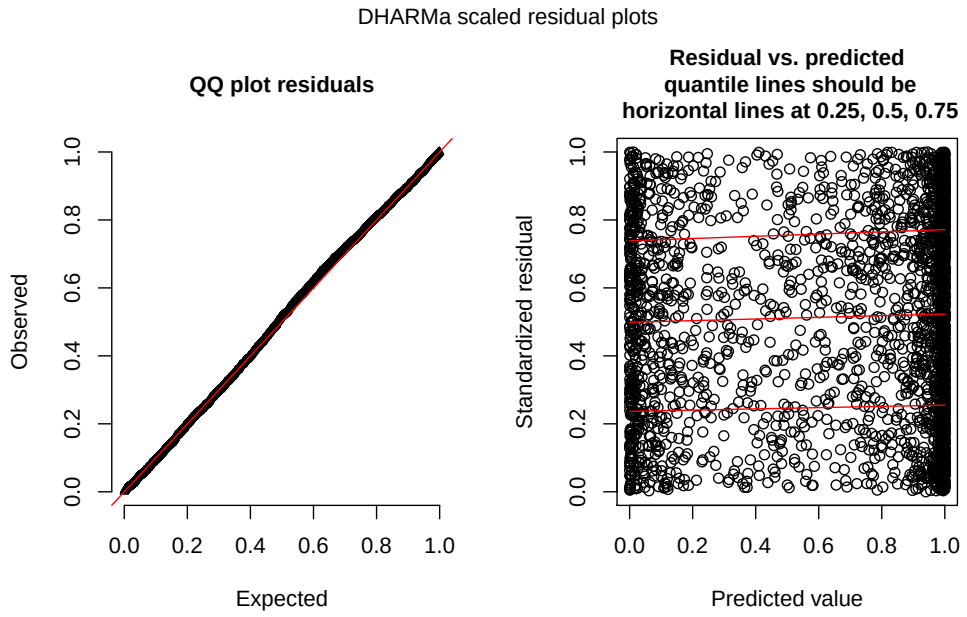

Abbildung 14: Im rechten Plot wurden die Residuen-Werte standardisiert und aufsummiert und gegen die Werte welche durch eine Binomialverteilung anzunehmen wären, geplottet. Da sie perfekt auf einer Linie liegen, ist kein systematischer Fehler zu erwarten. Im Rechtenplot sind die Residuen gegen die Vorhersagewerte aufgetragen. Hier ist auch keine Struktur zuerkennen und somit kann von einer zufälligen Residuenverteilung ausgegangen werden.

Dadurch haben wird eine räumliche Autokorrelation der Residuen. Plottet man die Residuen gegen die Vorhersagewerte bekommt man eine eindeutige Struktur, welche auch auf die Eigenschaften der Binomialverteilung und der transponierten Werte auf der Link-Skala zurück zuführt ist. Werden dagegen die Fehler für jeden einzelnen Datenpunkt simuliert, bekommt man eine Darstellung (rechte Abbildung 14), bei welcher keine Struktur in den Residuen zuerkennt ist.

```
> library(DHARMa)
> sim1 <- simulateResiduals(fmst)
> pdf("residualDARMa.pdf")
> plotSimulatedResiduals(sim1)
> dev.off()
> #Abbildung 14
```

2.11 Darstellungen im Raum

```
> ##
> #Bootstrapppen für den Fehler in der Prognode
> ##
>
> steps <- 25
> bot_mat <-matrix(NA, nrow(predi_trans02), steps)
>   for(i in 1:steps){
+     indi <- sample(1: nrow(predi_trans02), replace = T)
+     bfm <- glm(fmst, data = predi_trans02[indi,],
```

```
+                family=binomial)
+     bot_mat[,i] <- predict(bfm,predi_trans02,
+                            type="response")
+   }
> bot_quantil <- t(apply(bot_mat, 1,quantile,
+                        c(0.25, 0.5, 0.75)))
> mat_02_fehler <- predict(fmst,predi_trans02,
+                          type="response", se.fit = T)
> boot_fehler <- abs(bot_quantil[2] - mat_02_fehler$fit)
> ##
> #Maps mit ggplot
> ##
> #Matrix mit unterschiedlichen Prediktvalues erstellen
> library(ggplot2)
> art_predi02 <- art_predi_t
> art_predi02$GDD <- art_predi$GDD02
> d1 <- as.data.frame(cbind(art_predi$LON, art_predi$LAT))
> dataM2_pro <- model.matrix(f2, data=cbind.data.frame(predi_trans02)[,-1])
> lassomap<- predict(lasso, newx = as.matrix(dataM2), type="response")
> lassomap02<- predict(lasso, newx = as.matrix(dataM2_pro), type="response")
> rf_mat02 <- predict(rf_selc, art_predi02, type="prob")[,2]
> rf_mat <- predict(rf_selc, art_predi02, type="prob")[,2]
> mat_02 <- predict(fmst,predi_trans02, type="response")
> mat_01 <- predict(fmst,predi_trans, type="response")
> #Unterschiede zwischen der Prognose und den Originalwerten bzw.
> #den Modellwerten erstellen  (für das GLM)
> diff_mat <- mat_02 - mat_01
> diff_org <- mat_02 - art
> mat_org <- art
> res <- residuals(fmst)
> d <- cbind(d1, mat_02, mat_01, mat_org, res, diff_mat, diff_org,
+            rf_mat, rf_mat02, lassomap, lassomap02)
> colnames(d) <- c("lon", "lat", "mat_02","mat_01", "art",
+                  "res", "diff_mat", "diff_org", "rf_mat", "rf_mat02",
+                  "lasso_map", "lasso_map20")
> library(ggmap)
> library(ggplot2)
> ##
> #Basemap von google laden
> ##
> map <- get_map(location = 'Europe',  zoom = 3,  color ="bw")
> ##
> ##Darstellungen:
> ##
> map_art_org <- ggmap(map) +
```

```
+   geom_point(aes(x = lon, y = lat, color=lasso_map20), data = d,
+              shape=15, alpha = 0.5, size=1.3)+
+   scale_colour_gradient2(low = "white", mid="darkred",
+                          high = "darkgreen" , midpoint = 0.5, name =
+                            "Vorkommens- \nwahrscheinlichkeit")+
+   ggtitle("Shrinkage Modell ") +
+   scale_y_continuous(limits = c(33, 72), expand = c(0, 0)) +
+   scale_x_continuous(limits = c(-15, 33), expand = c(0, 0))
> png("lasso_map20.png")
> map_art_org
> dev.off()
> map_art_org <- ggmap(map) +
+   geom_point(aes(x = lon, y = lat, color=lasso_map), data = d,
+              shape=15, alpha = 0.5, size=1.3)+
+   scale_colour_gradient2(low = "white", mid="darkred",
+                          high = "darkgreen" , midpoint = 0.5,
+                          name = "Vorkommens- \nwahrscheinlichkeit")+
+   ggtitle("Shrinkage Modell bei Klimaerwärmung") +
+   scale_y_continuous(limits = c(33, 72), expand = c(0, 0)) +
+   scale_x_continuous(limits = c(-15, 33), expand = c(0, 0))
> png("lasso_map.png")
> map_art_org
> dev.off()
> map_art_org <- ggmap(map) +
+   geom_point(aes(x = lon, y = lat, color=rf_mat), data = d,
+              shape=15, alpha = 0.5, size=1.3)+
+   scale_colour_gradient2(low = "white", mid="darkred",
+                          high = "darkgreen" , midpoint = 0.5,
+                          name = "Vorkommens- \nwahrscheinlichkeit")+
+   ggtitle("Selected Random Forest Modell") +
+   scale_y_continuous(limits = c(33, 72), expand = c(0, 0)) +
+   scale_x_continuous(limits = c(-15, 33), expand = c(0, 0))
> png("rand_for.png")
> map_art_org
> dev.off()
> map_art_org <- ggmap(map) +
+   geom_point(aes(x = lon, y = lat, color=rf_mat02),
+              data = d, shape=15, alpha = 0.5, size=1.3)+
+   scale_colour_gradient2(low = "white", mid="darkred",
+                          high = "darkgreen" , midpoint =0.5,
+                          name = "Vorkommens- \nwahrscheinlichkeit")+
+   ggtitle("Selected Random Forest Modells bei Klimaerwärmung") +
+   scale_y_continuous(limits = c(33, 72), expand = c(0, 0)) +
+   scale_x_continuous(limits = c(-15, 33), expand = c(0, 0))
> png("rand_for20.png")
```

```
> map_art_org
> dev.off()
> map_art_org <- ggmap(map) +
+   geom_point(aes(x = lon, y = lat, color=art), data = d,
+              shape=15, alpha = 0.5, size=1.3)+
+   scale_colour_gradient2(low = "white", mid="darkred",
+                          high = "darkgreen" , midpoint =0.5,
+                          name = "Vorkommens-\nwahrscheinlichkeit")+
+ ggtitle("Originaldaten über das Vorkommen \n von Myotis nattereri") +
+   scale_y_continuous(limits = c(33, 72), expand = c(0, 0)) +
+   scale_x_continuous(limits = c(-15, 33), expand = c(0, 0))
> png("org_map.png")
> map_art_org
> dev.off()
> #Karte für Residuen
> map_art_res <- ggmap(map) +
+   geom_point(aes(x = lon, y = lat, color=res), data = d,shape=15,
+              alpha = 0.5, size=1.3)+
+   scale_colour_gradient2(low = "darkblue", mid="white",
+                          high = "darkred", name = "Residuen")+
+   ggtitle("Residuen aus dem backwards
+           stepwise\n GLM Model, räumlich Verteilt") +
+ scale_y_continuous(limits = c(33, 72), expand = c(0, 0)) +
+   scale_x_continuous(limits = c(-15, 33), expand = c(0, 0))
> png("res_map.png")
>  map_art_res
> dev.off()
> #Karte für die Wahrscheinlichkeiten die das Modell berechnte
> map_art_mat_01 <- ggmap(map) +
+   geom_point(aes(x = lon, y = lat, color=mat_01),
+              data = d,shape=15, alpha = 0.5, size=1.3)+
+   scale_colour_gradient2(low = "white", mid="darkred",
+                          high = "darkgreen" , midpoint =0.5,
+                          name = "Vorkommens- \nwahrscheinlichkeit")+
+   ggtitle("Myotis nattereri modeliert
+           \nmit dem backwards stepwise\n GLM Models") +
+   scale_y_continuous(limits = c(33, 72), expand = c(0, 0)) +
+   scale_x_continuous(limits = c(-15, 33), expand = c(0, 0))
> png("glm_map.png")
> map_art_mat_01
> dev.off()
> #GLM bei Klimaerwärmung
> map_art_mat_02 <- ggmap(map) +
+   geom_point(aes(x = lon, y = lat, color=mat_02),
+              data = d,shape=15, alpha = 0.5, size=1.3)+
```

```
+   scale_colour_gradient2(low = "white", mid="darkred",
+                          high = "darkgreen" , midpoint =0.5,
+                          name = "Vorkommens \nWahrscheinlichkeit")+
+   ggtitle("Myotis nattereri modeliert
+           \nmit dem backwards stepwise\n GLM bei Klimaerwärmung") +
+   scale_y_continuous(limits = c(33, 72), expand = c(0, 0)) +
+   scale_x_continuous(limits = c(-15, 33), expand = c(0, 0))
> png("glm20_map.png")
> map_art_mat_02
> dev.off()
> #Karte für die Unterschiede zwischen dem Jetzt
> #Zustand(Modell) und der Prognose
> map_art_diff <- ggmap(map) +
+   geom_point(aes(x = lon, y = lat, color=diff_mat),
+              data = d,shape=15, alpha = 0.5, size=1.3)+
+   scale_colour_gradient2(low = "darkred", mid="white",
+                          high = "darkgreen" , midpoint =0,
+                          name = "Vorkommens- \nwahrscheinlichkeit")+
+   ggtitle("Unterschied zwischen dem
+           \nGLM Modell im jetzigem Zustand und dem Szenario") +
+   scale_y_continuous(limits = c(33, 72), expand = c(0, 0)) +
+   scale_x_continuous(limits = c(-15, 33), expand = c(0, 0))
> png("div_glm_map.png")
> map_art_diff
> dev.off()
> #Karte für die Unterschiede zwischen dem Jetzt Zustand und der Prognose
> map_art_org <- ggmap(map) +
+   geom_point(aes(x = lon, y = lat, color=diff_org),
+              data = d, shape=15, alpha = 0.5, size=1.3)+
+   scale_colour_gradient2(low = "darkred",
+                          mid="white", high = "darkgreen", name = "Vorkommens \nWahrscheinlichkeit")+
+   ggtitle("Unterschied zwischem \nden Originaldaten und dem Szenario") +
+   scale_y_continuous(limits = c(33, 72), expand = c(0, 0)) +
+   scale_x_continuous(limits = c(-15, 33), expand = c(0, 0))
> png("diff_org.png")
> map_art_org
> dev.off()
```

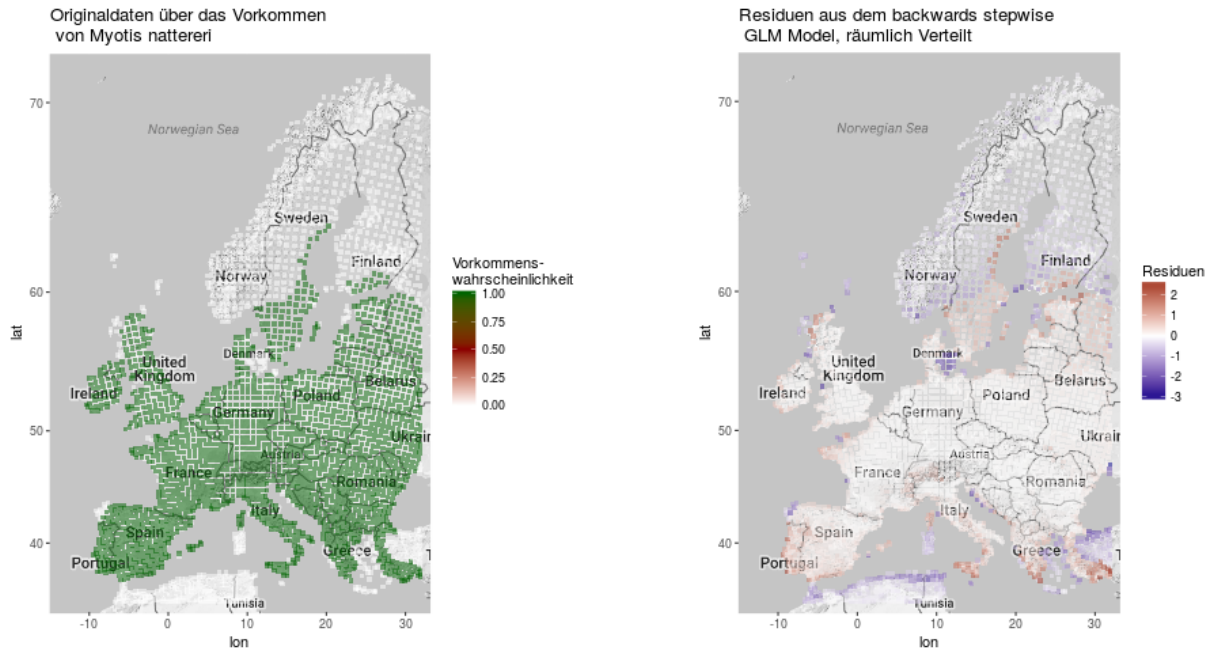

Abbildung 15: Die linke Karte zeigt die Originaldaten von Myotis nattereri. Die Ausbreitung stimmt überein mit der aus Literaturangaben. Die linke Karte zeigt die Verteilung der Residuen. Es fällt auf, dass die Randgebiete höhere Residuen haben. Erklärt werden kann dies durch die Transformation der Daten auf die Link-Skala.

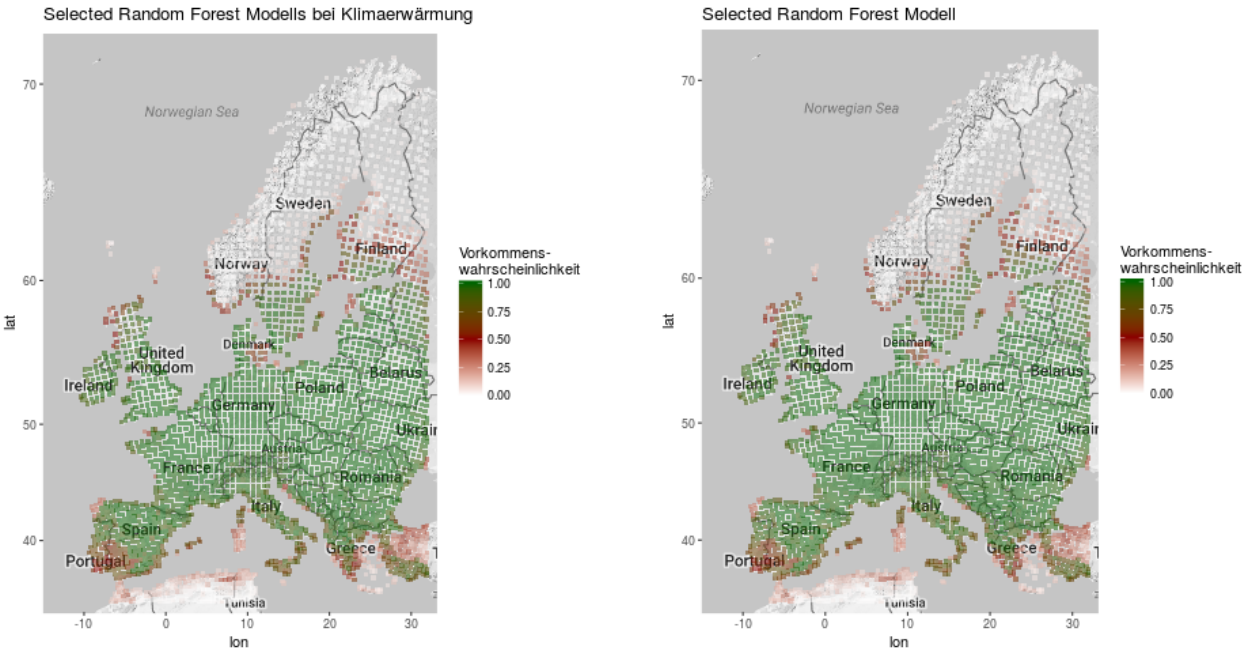

Abbildung 16: Der Unterschied zwischen der Prognose und den jetzigem Zustand ist relativ gering. Dennoch breitet sich die Art, laut dem Modell Richtung Norden weiter aus. Die Randgebiete sind auch in dieser Darstellung mit Unsicherheiten behaftet, so dass ich der Prognose für Nord-Afrika nicht vertrauen würde

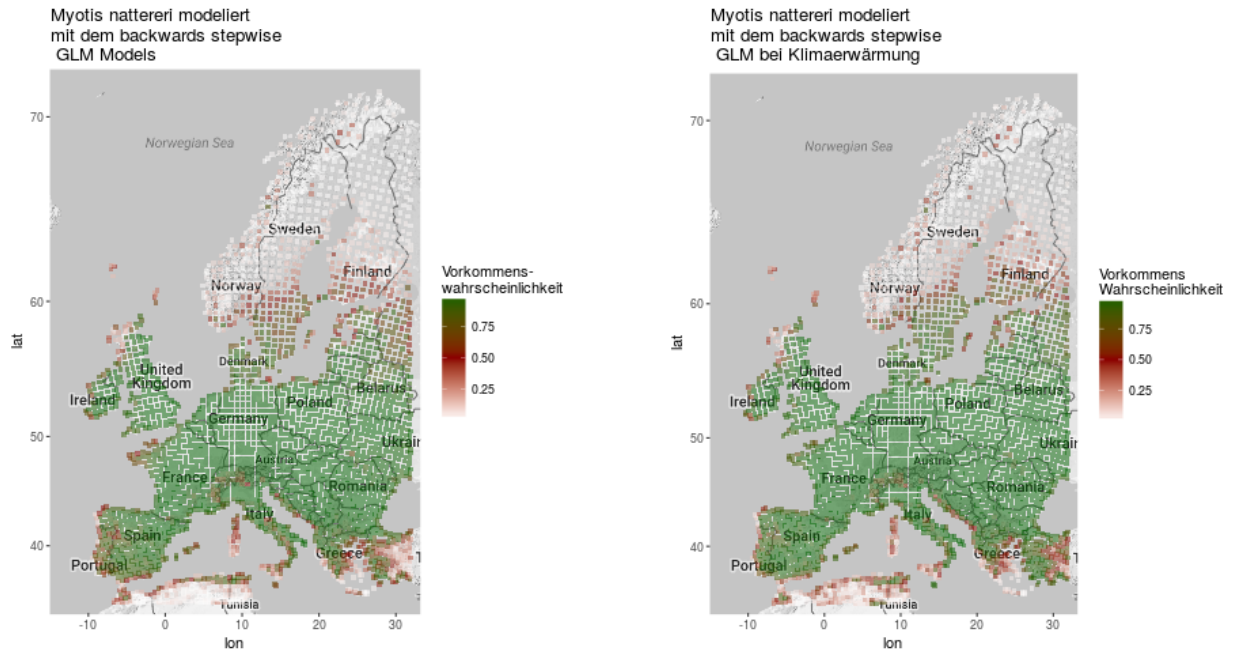

Abbildung 17: Bei dem GLM Modell ist der Unterschied zwischen der Prognose und dem jetzigem Zustand deutlich zu erkennen: Die Fransenfledermaus wird sich nach Norden weiter ausbreiten. Auch Südosten und sogar in Afrika wird sie ihr Habitat ausbauen können. Dies ist aus Sicht der Biologie der Fransenfledermaus nicht nachzuvollziehen.

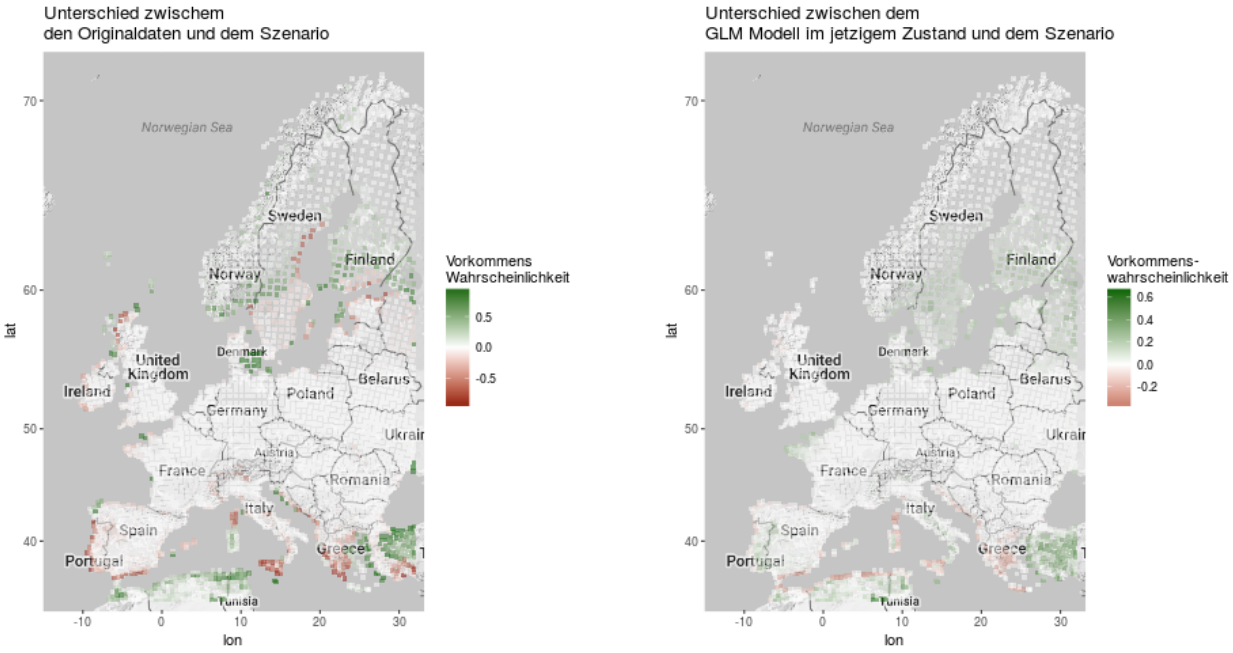

Abbildung 18: In dieser Darstellung sind die Differenzen farblich hervorgehoben. Die Wahrscheinlichkeit in Nord-Afrika erhöht sich, in Italien und auf einigen Inseln verringert sie, sich laut Prognose. Besonders der Unterschied zwischen Korsika und Sardinien kann nicht biologisch erklärt werden. Auf Sardinen tritt die Fransenfledermaus laut Literaturangaben (Kuhn) nicht auf, das Modell prognostiziert aber eine höhere Wahrscheinlichkeit als auf Korsika, wo die Art heutzutage heimisch ist.

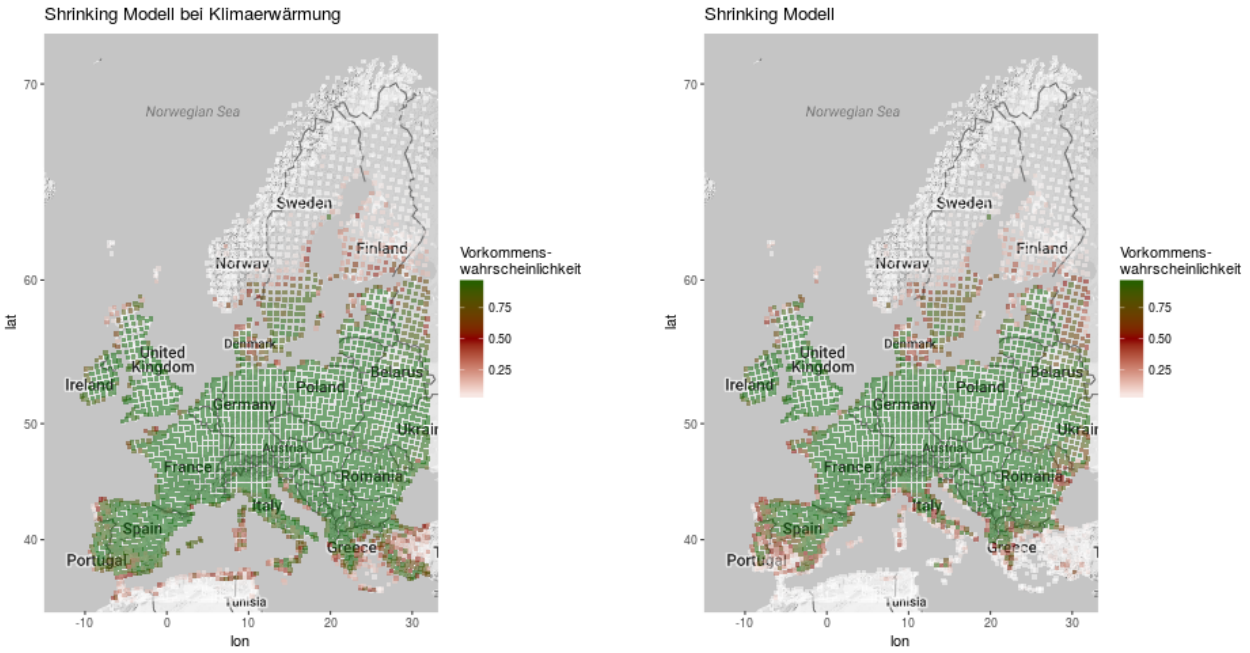

Abbildung 19: Betrachtet man den RMSE-Wert ist das Shrinkage-Modell mit Abstand am besten. Auch die Karte stimmt mit den Originaldaten gut überein. Die Prognose sieht auch für Nord-Afrika sinnvoll aus.

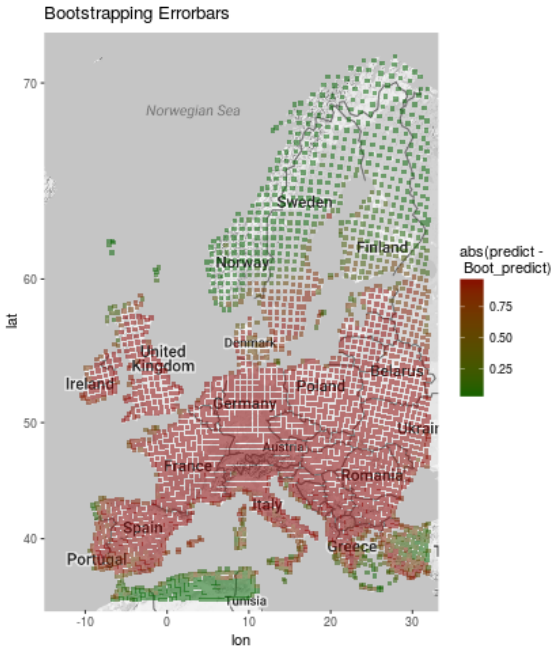

Abbildung 20: Die Unsicherheit, welche sich aus den Boostrapping der Prognose ergibt. Besonders in den Randgebieten ist dieser sehr niedrig.

3 Ergebnis und Diskussion

Die Analyse zeigt, dass sich das potentielle Ausbreitungsgebiet bei steigenden mittleren Jahrestemperaturen von Myotis nattereri weiter in den Norden verschieben wird. Der limitierende Faktor für die Ausbreitung in den Norden (ab 63. Breitengrad), ist die absolute Kälte während des Winterschlafes. Verschiebt sich die Grenze von extremen Kälteeinbrüchen im Winter weiter nach Norden, so kann die Myotis nattereri dieser potentiell folgen. Die Analyse konnte keinen eindeutigen generellen Effekt von weiteren Prädiktoren ausmachen. Der allgemeine Rückgang in den letzen Jahren von Myotis nattereri hat andere Gründe, welche nicht in die Analyse mit eingeflossen sind. Dazu gehört besonders der Rückgang von geeigneten Quartieren. Dadurch könnte trotz der Erwärmung die Myotis nattereri an Abundanz abnehmen, auch im Norden. Das GLM-Modell prognostiziert einen Anstieg der Abundanz in Nord-Afrika und der Türkei. Dafür gibt es aber keine biologische Erklärung, wodurch die Aussage sehr kritisch zu interpretieren ist. Das Shrinkage-Modell (Abb 19) prognostiziert eine realistischere Ausbreitung, bei der Myotis nattereri komplett aus Nord-Afrika verschwindet. Da sie hauptsächlich in feuchten Wäldern jagt, ist dies aus ökologischer Sicht plausibel. Die Schwäche des Modelles zeigt sich am deutlichsten auf den Inseln **??**. Auf Sardinen gibt es laut (Kuhl 1817) keine Vorkommen von Myotis nattereri und ist so auch in den Originaldaten abgebildet (Abb.15). Das Modell bildet dies aber nicht ab und prognostiziert sogar einen Anstieg von Myotis nattereri auf Sardinen und eine Abnahme auf Korsika, wo die Art heute schon lebt (Abbildung 18).
Folgende Punkte sind an der Methode der Analyse zu kritisieren bzw. könnten besser gemacht werden:

- Die Prädiktoren sind zu unspezifisch um Eigenschaften des Habitates von Myotis nattereri abzubilden (zum Beispiel potentielle Quartiersmöglichkeiten).
- Durch das step-backwards-GLM wurden nicht alle potentiellen Wege getestet, sondern nicht signifikante Interaktionen immer rausgeworfen. Dadurch werden mögliche Tade-Offs ausgeschlossen. Außerdem ging keine Funktion zweiten Grades mit in dieses Modell ein.
- Durch das Scheitern an der Darstellung der Shrinkage-Marginal-Effektplots ging das beste Modell in der Analyse nicht weiter behandelt.
- Durch die höheren Residuen an den Habitatsgrenzen, kann von einer räumlichen Auto-Korrelation ausgegangen werden, welche in keinem Modell berücksichtigt wurde.