Modelle mit Nutzungsintensität Fallstudie WPZ (fakultativ)

B.Sigrist

18. November 2019

Neue packages die wir fuer die Modelle und die Diagnostics brauchen

```
# neue Packages: DHARMa, car, MASS, ROCR, sjPlot, sjstats, rms, ggeffects,
# cowplot

ipak <- function(pkg) {
    new.pkg <- pkg[!(pkg %in% installed.packages()[, "Package"])]
    if (length(new.pkg))
        install.packages(new.pkg, repos = "http://cran.us.r-project.org", dependencies = T
    sapply(pkg, require, character.only = TRUE)
}

packages <- c("lme4", "bbmle", "MuMIn", "tidyverse", "DHARMa", "car", "MASS", "ROCR",
    "sjPlot", "rms", "ggeffects", "sjstats", "cowplot", "glmmTMB", "performance",
    "kableExtra")

ipak(packages)
```

```
DF_mod_day <- read_delim("data/Datensatz_Habitatnutzung_Modelle_20191101.csv", delim = ";"
    filter(time_of_day == "day") %>%
    mutate(slope_scaled = scale(slope), us_scaled = scale(us), os_scaled = scale(os),
        forest_prop_scaled = scale(forest_prop), dist_road_scaled = scale(dist_road_all),
        dist_road_only_scaled = scale(dist_road_only), dist_build_scaled = scale(dist_buil
        id = as.factor(id))
```

ursprüngliche Funktion und Modelformel

```
# glmer(formula, data = , family = binomial)

# 1) formula:
# Abhaengige Variable ~ Erklaerende Variable + Random Factor
# In unseren Modellen kontrollieren wir fuer individuelle Unterschiede bei den Rehen
# indem wir einen Random Factor definieren => (1 | id)

# 2) data:
# euer Datensatz

# 3) family:
# binomial

# Verteilung der abhängigen Variable bei der Nutzungsintensität aus?

ggplot(DF_mod_day, aes(nmb)) + geom_histogram()
```

Hinsichtlich der Nutzungsintensität müssen wir die Formel erweitern:

```
# Erweiterung um einen sog. Offset-Term, der hier gebraucht wird, um für die
# Anzahl der GPS Lokalisationen (in der Spalte GPStot aufgeführt) zu korrigeren
# (eigentlich eine Skalierung der abhängigen Variable um die relative
# Nutzungsintensität zu modellieren)

f_count <- nmb ~ slope_scaled + dist_road_scaled + forest_prop_scaled + os_scaled +
    us_scaled + dist_build_scaled + offset(log(GPStot)) + (1 | id)

### Für die Nutzungsintensität brauchen wir ein neues package (glmmTMB) um das
### GLMM fitten zu können. glmer kann leider mit der negativ binomial -
### Verteilung nicht in jedem Fall umgehen.

m <- glmmTMB(f_count, data = DF_mod_day, family = glmmTMB::nbinom2())

# Das Modell in die dredge-Funktion einfügen (siehe auch unbedingt ?dredge)

all_m <- dredge(m)

avgmodel <- model.avg(all_m, rank = "AICc", subset = delta < 2)
```

```r
summary(avgmodel)
```

Model testing for over/underdispersion, zeroinflation and spatial autocorrelation following the DHARMa package.

unbedingt die Vignette des DHARMa-Package konsultieren: https://cran.r-project.org/web/packages/DHARMa/vignettes/DHARMa.html

```r
f_count <- nmb ~ slope_scaled + dist_road_scaled + forest_prop_scaled + os_scaled +
    us_scaled + offset(log(GPStot)) + (1 | id)

### Für die Nutzungsintensität brauchen wir ein neues package (glmmTMB) um das
### GLMM fitten zu können. glmer kann leider mit der negativ binomial -
### Verteilung nicht in jedem Fall umgehen.

m_day_count <- glmmTMB(f_count, data = DF_mod_day, family = glmmTMB::nbinom2())

summary(m_day_count)

tab_model(m_day_count, transform = NULL, show.se = T)

# Residuals werden ueber eine Simulation auf eine Standard-Skala transformiert
# und kaennen anschliessend getestet werden. Dabei kann die Anzahl Simulationen
# eingestellt werden (dauert je nach dem sehr lange)

simulationOutput <- simulateResiduals(fittedModel = m_day_count, n = 10000)

# plotting and testing scaled residuals

plot(simulationOutput)

testResiduals(simulationOutput)

# The most common concern for GLMMs is overdispersion, underdispersion and
# zero-inflation.

# separate test for dispersion

testDispersion(simulationOutput)
```

```
# test for Zeroinflation

testZeroInflation(simulationOutput)

# test for spatial Autocorrelation

dM = as.matrix(dist(cbind(DF_mod_day$x, DF_mod_day$y)))

testSpatialAutocorrelation(simulationOutput, distMat = dM, plot = F)

# Testen auf Multicollinearitaet (dh zu starke Korrelationen im finalen Modell,
# zB falls auf Grund der oekologsichen Plausibilitaet stark korrelierte
# Variablen im Modell)
#--> funktioniert bei glmmTMB Modellen mit dieser Funktion aus dem performace package:

check_collinearity(m_day_count)
```

AUC funktioniert nicht bei nicht-binären abhängigen Variablen, daher müssen wir eine andere Möglichkeit finden um den Goodness-of-fit der Modelle abzuschätzen:

```
# Zitat B.Bokler 2013: 'GLMMs are still part of the statistical frontier, and
# not all of the answers about how to use them are known (even by experts)'

r2(m_day_count)
```

Plots der vorhergesagten relativen Nutzungsintensität funktionieren nach dem selben Prinzip das wir bereits kennen:

```
# graphische Darstellung der gesamten Modellresultate

plot_model(m_day_count, transform = NULL, show.values = TRUE, value.offset = 0.3)

# Plotten der vorhergesagten Wahrscheinlichkeit, dass ein Kreis besetzt ist, in
# Abhaengigkeit der erklaerenden Variable basierend auf den Modellresultaten.

plot_model(m_day_count, type = "pred", terms = "dist_road_scaled")
```

```r
# Problem: skalierte Variablen lassen sich nicht so ohne Weiteres plotten, hier
# ein quick-and-dirty hack um das Problem zu umgehen. Die Einstellungen muessen
# fuer jede Variable geaendert werden

p <- plot_model(m_day_count, type = "pred", terms = "dist_road_scaled")

labels <- round(seq(floor(min(DF_mod_day$dist_road_all)), ceiling(max(DF_mod_day$dist_road
    length.out = 6), 2)

p <- p + scale_x_continuous(breaks = c(-1, 0, 1, 2, 3, 4), labels = c(labels))

p
```