Predictive Models

Modeling butterfliy distribution

Prof. Dr. Loic Pellissier

August 2, 2016

Contents

1 Introduction

During the practical, you will focus only on modelling the distribution of a group of butterflies, from the *Lycaenidae* family in Switzerland. Species distribution models relate species occurrences to a set of environmental predictors. This approach allows extrapolating information on species occurrences from a few sites to an entire region.

1.1 GBIF

The Global Biodiversity Information Facility (GBIF, www.gbif.org) is an international organisation that focuses on making scientific data on biodiversity available via the Internet using web services. The data are provided by many institutions from around the world; GBIF's information architecture makes these data accessible and searchable through a single portal. Data available through the GBIF portal are primarily distribution data on plants, animals, fungi, and microbes accross earth, and scientific names data.

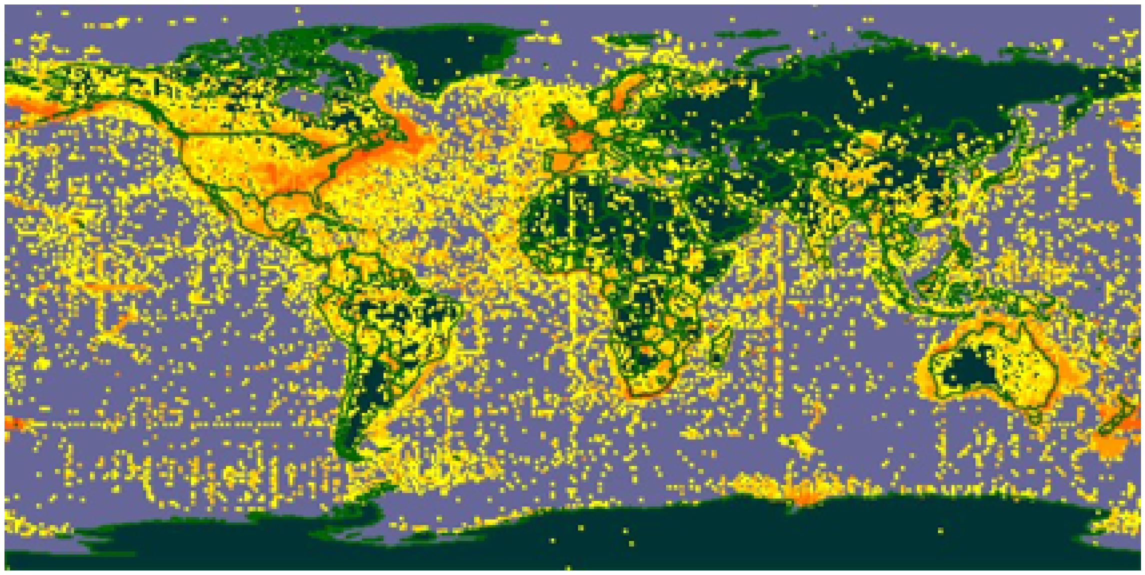


Figure 1: Map of species ocurrences available on GBIF

1.2 Study species

Lycaenidae is the second-largest family of butterflies (behind the Brush-footed butterflies), with over 5,000 species worldwide. Lycaenids are diverse in their food habits and aside from being phytophagous, some of them are also entomophagous, feeding on aphids and ant larvae. Some Lycaenids even exploit this association by inducing ants to feed them by regurgitation, a process called trophallaxis. Not all Lycaenid butterflies need ants, but about 75% of species associate with ants, a relationship called myrmecophily. These associations can be mutualistic, parasitic, or predatory depending on the species. In the case of some species, larvae are attended and protected by ants while they feed on the host plant and in turn, the ants receive sugar-rich honeydew from the larvae, throughout the larval life.

1.3 Study Area

2 Uploading data in R

Question 2.0.1 *Which environmental factor would you say best explains the distribution of the butterfly species?*

2.1 Load climatic data

Import the library containing required functions:

```
library(raster)
```

Load the three climatic rasters:

```
DDEG <- raster("../data/ddeg_1km")
MIND <- raster("../data/mind_1km")
SRAD <- raster("../data/srad_1km")
```

Where

- DDEG is the number of degrees above a threshold, a proxy for energy available.
- MIND is the moisture index,
- SRAD is radiation,

Vizualise the climatic maps that you have loaded:

```
library(RColorBrewer)
par(mar = c(3, 0.1, 1, 0.1), mfrow = c(2, 2))
plot(DDEG, col = rev(heat.colors(20)), box = F, axes = F, legend = F)
plot(DDEG, legend.only = T, horizontal = T, add = T, col = heat.colors(20),
    smallplot = c(0.25, 0.75, 0.08, 0.1))
plot(MIND, col = brewer.pal(20, "PuBu"), box = F, axes = F, legend = F)
```

```
## Warning in brewer.pal(20, "PuBu"): n too large, allowed maximum for palette PuBu is 9
## Returning the palette you asked for with that many colors
```

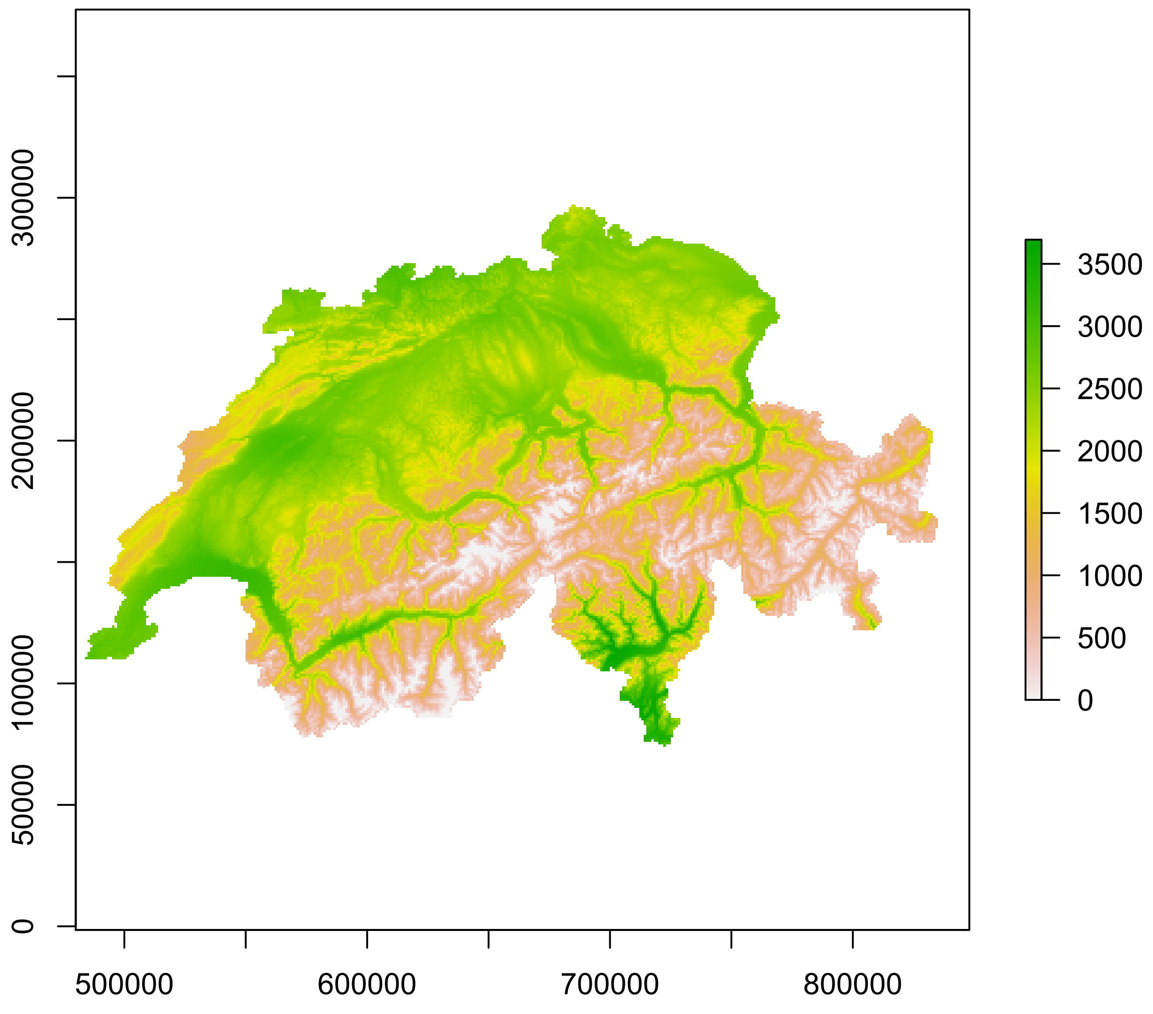


Figure 2: Study area in Switzerland with temperature as background representing the energy available for caterpillar development. The climatic maps were obtained by combining meteorological records from the Swiss meteorological stations and a digital elevation model mapping the elevation across Switzerland. Because temperature decrease linearly with elevation, it is possible to extrapolate the meteorological information from the station to the entire Switzerland.

Figure 3: In this practical, we will focus on modelling the species Glaucopsyche alexis (Picture from Wikimedia File:Glaucopsyche alexis (3722345118).jpg

```
plot(MIND, legend.only = T, horizontal = T, add = T, col = brewer.pal(20, "PuBu"),
    smallplot = c(0.25, 0.75, 0.08, 0.1))
```

```
## Warning in brewer.pal(20, "PuBu"): n too large, allowed maximum for palette PuBu is 9
## Returning the palette you asked for with that many colors
```

```
plot(SRAD, col = brewer.pal(20, "YlOrRd"), box = F, axes = F, legend = F)
```

```
## Warning in brewer.pal(20, "YlOrRd"): n too large, allowed maximum for palette YlOrRd is 9
## Returning the palette you asked for with that many colors
```

```
plot(SRAD, legend.only = T, horizontal = T, add = T, col = brewer.pal(20, "YlOrRd"),
    smallplot = c(0.25, 0.75, 0.14, 0.16))
```

```
## Warning in brewer.pal(20, "YlOrRd"): n too large, allowed maximum for palette YlOrRd is 9
## Returning the palette you asked for with that many colors
```

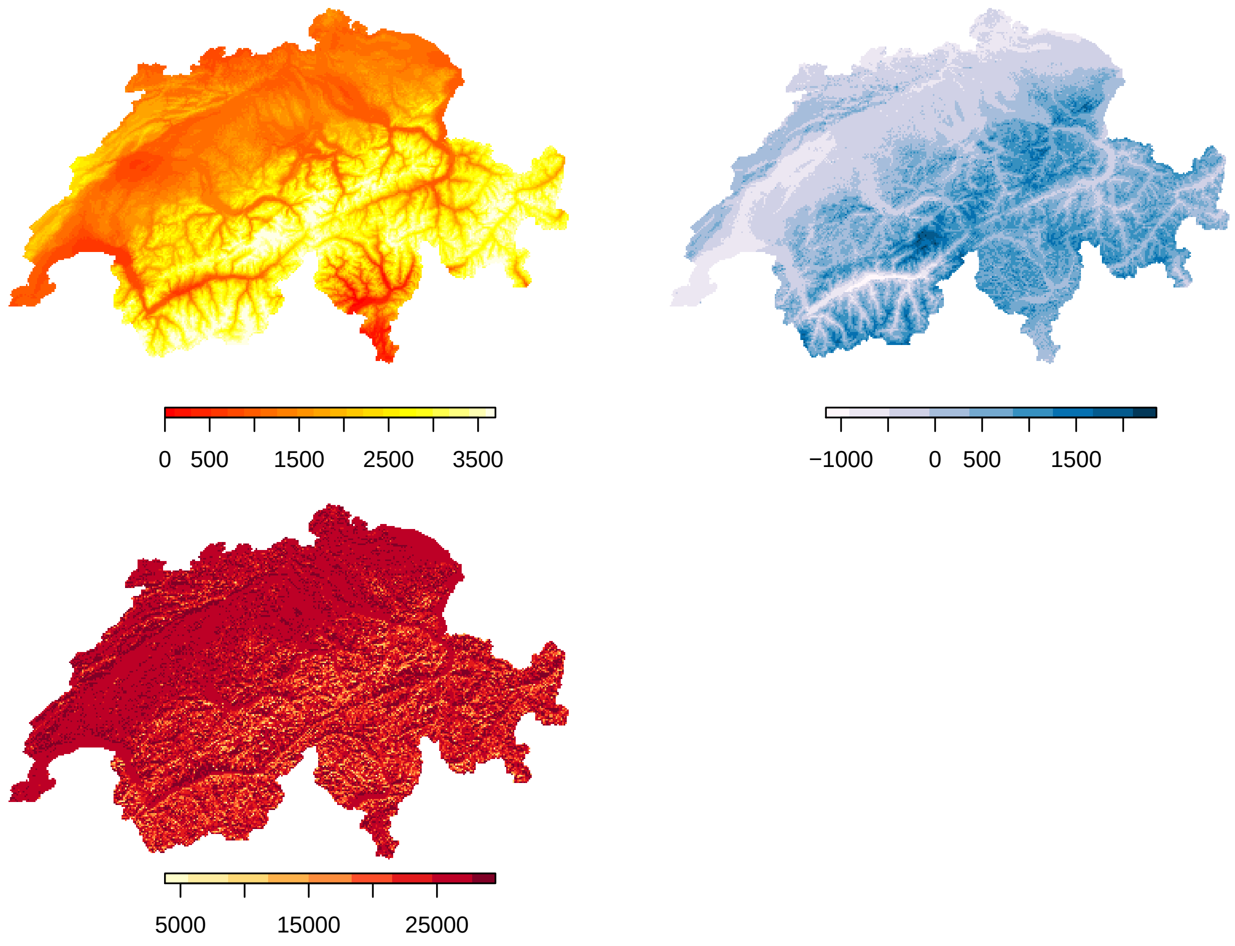


Figure 4: Climatic rasters

The three climatic rasters are *temperature*, *precipitation* and *solar radiation*.

Question 2.1.1 *Observe the differences in climate across Switzerland. Why do you think that temperature, precipitation and solar radiation are relevant to explain the biology and distribution of the butterfly species in Switzerland?*

2.2 Load species' data

In the following section, you will manipulate the geographic coordinate system of the occurrence points.

Load the occurrence data:

```
Occurrences <- read.table("../data/Occurrences_CH1903.txt", h = T)
```

Vizualise the data:

```
library(pander)
pander(head(Occurrences), caption = "Sample of occurrences data")
```

	Latitude	Longitude	Family	Genus	Species	Year
1	46.32	7.96	Lycaenidae	Polyommatus	Polyommatus_dama	1971
3	47.38	9.65	Lycaenidae	Plebejus	Plebejus_argus	1966
4	47.44	9.65	Lycaenidae	Polyommatus	Polyommatus_icarus	1968
5	47.16	9.48	Lycaenidae	Lycaeides	Lycaeides_idas	1973
8	46.89	9.474	Lycaenidae	Aricia	Aricia_artaxerxes	1986
9	46.46	7.895	Lycaenidae	Aricia	Aricia_artaxerxes	1988

Table 1: Sample of occurrences data (continued below)

	X	Y
1	640155	129978
3	766976	250038
4	766788	256707
5	754779	225239
8	755092	195146
9	635077	145140

```r
par(mfrow = c(1, 1))
plot(Occurrences$Longitude, Occurrences$Latitude)
```

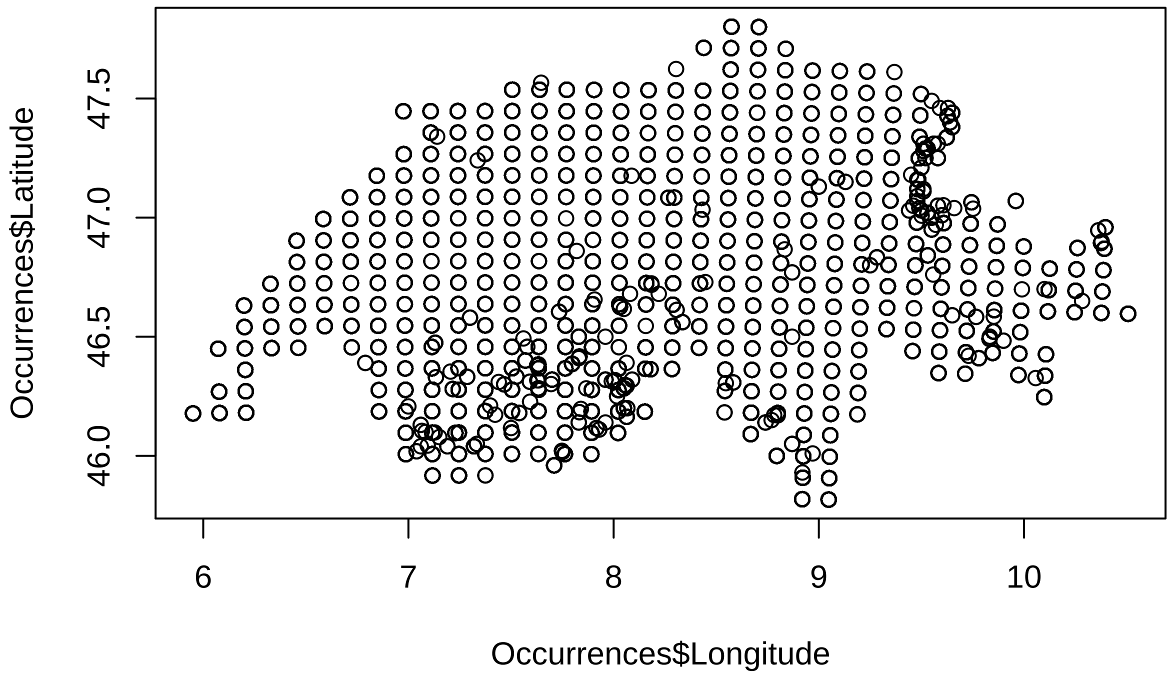


Figure 5: Butterfly occurrence observations

3 Link response variable to climatic predictors

3.1 Link predictors to *ocurrence* data points

This step is essential to link the information of your species presence to the variables explaining the distribution of the species.

```r
preds <- data.frame(extract(DDEG, Occurrences[, c("X", "Y")]), extract(MIND,
    Occurrences[, c("X", "Y")]), extract(SRAD, Occurrences[, c("X", "Y")]))
names(preds) <- c("DDEG", "MIND", "SRAD")
```

Add climate predictors data to the occurrence dataframe:

```r
Occurrences <- cbind(Occurrences, preds)
```

3.2 Complete training set by creating *pseudo-absences*

The species distribution model will discriminate climatic condition where a given species has been observed to the entire climatic environment that is available to the species (but where it may not be necessarily found). We will first generate 500 *pseudo-absence* points where we assume the species is not found. We will then extract the values of the three climatic rasters for each pseudoabsence points.

Extract all cells of the background environment:

```r
CellsXY <- na.omit(as.data.frame(DDEG, xy = T))
```

Select randomly 500 cells which are called pseudoabsences:

```r
set.seed(1)
id <- sample(1:nrow(CellsXY), 500)
CellsXY_red <- CellsXY[id, ]
```

Plot the selected pseudoabsences on a map:

```r
par(mar = c(0, 0, 0, 0))
plot(DDEG, col = rev(heat.colors(20)), box = F, axes = F, legend = T)
points(CellsXY_red[, c("x", "y")])
```

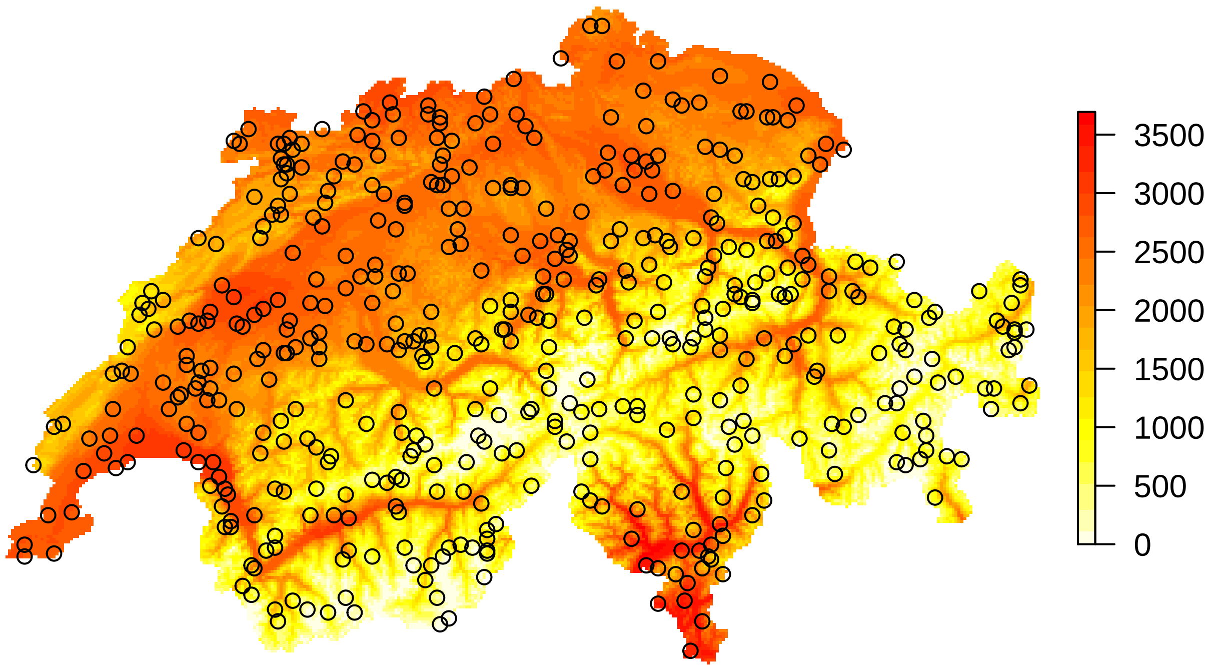


Figure 6: Degree-day raster and butterfly pseudo-absence observations

Obtain climate predictors for pseudo-absences locations:

```
preds <- data.frame(extract(DDEG, cbind(CellsXY_red[, c("x", "y")])), extract(MIND,
    cbind(CellsXY_red[, c("x", "y")])), extract(SRAD, cbind(CellsXY_red[, c("x",
    "y")])))
names(preds) <- c("DDEG", "MIND", "SRAD")
```

Add Climate data to the pseudoabsence dataframe:

```
CellsXY_red <- cbind(CellsXY_red, preds)
```

4 GLMs

The classic linear model is:

$$y_i = \mathbb{E}[y_i|x_i] + \epsilon_i = \beta^T x_i + \epsilon_i \tag{1}$$

where:

- y is the response variable,
- x is a vector of predictors, and
- ϵ is the error term, a random variable such that:
    - mean is zero: $\mathbb{E}[\epsilon] = 0$,

- variance is constant: $Var[\epsilon] = \sigma^2$,
- independent identically distributed,
- unbounded support,
- typically we assume $y_i \sim N(\mu_i, \sigma^2)$

Although this is a useful model it has two important restrictions:

1. The range of y must not be restricted since the range of $\mathbb{E}[y|x] = \beta^T x \in (-\infty, \infty)$ is not. This is why y and ϵ are assumed to have a distribution with unbounded support such as the normal distribution.
2. The variance of y must be constant.

GLMs address both these issues. Like the linear model, a generalized linear model consists of a linear predictor $f(x) = \beta^T x$ and, additionally, two functions:

- A **link** function $g(\mu)$ that describes how the mean depends on the linear predictor:

$$g(\mu) = g(\mathbb{E}[y|x]) = \beta^T x \tag{2}$$

  This allows us to have a response variable with a restricted range. The function g could, *a priori*, be any function such that if $\mathbb{E}[y|x] \in (a, b)$ then $g(\mathbb{E}[y|x]) \in (-\infty, \infty)$.
- A **variance** function $V(\mu)$ that describes how the variance depends on the mean:

$$Var[y_i] = \phi V(\mathbb{E}[y|x]) = \phi V(\mu) \tag{3}$$

  where ϕ is a constant.

4.1 Normal response

Assume our data is distributed normally such that:

- $y_i \sim N(\mu_i, \sigma^2)$
- $\mu_i = \mathbb{E}[y_i|x_i] = \beta^T x_i$
- $Var[y_i|x_i] = \sigma^2$

Since $\mu_i \in (-\infty, \infty)$, a valid choice for the link function is:

$$g(\mu) = \mu \in (-\infty, \infty) \tag{4}$$

Also since $Var[y_i|x_i] = \sigma^2$ we have that $V(\mu) = 1$. Since a normal variable y can be written as the sum of its expectation μ and a zero mean normal with the same variance σ^2 as y we see that for this choice of link function we recover the classic linear model:

$$y_i = \mu_i + \epsilon_i = g^{-1}(\beta^T x_i) + \epsilon_i = \beta^T x_i + \epsilon_i \tag{5}$$

where $\epsilon \sim N(0, \sigma^2)$.

4.2 Bernoulli response

Assume our data is distributed Bernoulli such that:

- $y_i \sim Bernoulli(p_i)$
- $p_i = \mathbb{E}[y_i|x_i] = \beta^T x_i$
- $Var[y_i|x_i] = p_i(1-p_i)$

Since $p_i \in (0,1)$, a valid choice for the link function is the *log-odds ratio* or *logit* function:

$$g(p) = \log \frac{p}{1-p} \in (-\infty, \infty) \tag{6}$$

Also since $Var[y_i|x_i] = p_i(1-p_i)$ we have that $V(p) = p(1-p)$. For this choice of link function we obtain the **logistic** regression model:

$$\mathbb{E}[y_i|x_i] = p_i = g^{-1}(\beta^T x_i) = \frac{e^{\beta^T x_i}}{1+e^{\beta^T x_i}} \tag{7}$$

$$Var[y_i|x_i] = p_i(1-p_i) \tag{8}$$

Another valid choice for the link function is the *probit* function:

$$g(p) = \Phi^{-1}(p) \in (-\infty, \infty) \tag{9}$$

where $\Phi(p)$ is the normal cumulitive distribution function. This choice of link function leads to the **probit** regression model.

4.3 Poisson response

Assume our data is distributed Poisson such that:

- $y_i \sim Poisson(\lambda_i)$
- $\lambda_i = \mathbb{E}[y_i|x_i] = \beta^T x_i$
- $Var[y_i|x_i] = \lambda_i$

Since $\lambda_i \in (0, \infty)$, a valid choice for the link function is the *log* function:

$$g(\lambda) = \log \lambda \in (-\infty, \infty) \tag{10}$$

Also since $Var[y_i|x_i] = \lambda_i$ we have that $V(\lambda) = \lambda$. For this choice of link function we obtain the **Poisson** regression model:

$$\mathbb{E}[y_i|x_i] = \lambda_i = g^{-1}(\beta^T x_i) = e^{\beta^T x_i} \tag{11}$$

$$Var[y_i|x_i] = \lambda_i = g^{-1}(\beta^T x_i) = e^{\beta^T x_i} \tag{12}$$

4.4 Residual analysis and diagnostics

In the classic linear regression setting the residuals are a *natural* quantity to study owing to the fact that the response variable can be expressed as a location model:

$$y_i = \mathbb{E}[y_i|x_i] + \epsilon = \mu_i + \epsilon \tag{13}$$

meaning that the residuals $r_i = y_i - \hat{\mu}_i$ are estimators of the error term ϵ. In the linear regression setting we can look at certain plots of the residuals to verify model assumptions:

- The Tukey-Anscombe plot helps check the unbiasedness of the model: $\mathbb{E}[r_i] = 0$
- The Scale-location plot helps check the homoscedasticity of errors: $Var[r_i] = k$
- The normal Q-Q plot helps check that error term is indeed distributed normally
- Cook and leverage plots help identify influential outliers: observations that possibly don't belong to the same dataset but which influence parameter estimation.

For general linear models, in general we don't know how the residual term $r_i = y_i - \hat{\mu}_i$ is distributed since non-normal response variables y cannot, in general, be expressed as a location model $y = \mu + \epsilon$. We do however know that if the variance of the response variable is non-constant, as in the Bernoulli and Poisson case, then the residuals will also have non-constant variance. We also know that the residuals need not be distributed normally so the Scale-location and normal Q-Q plots do not apply in the GLM setting.

In order to check for the unbiasedness of the model we look at a modified Tukey-Anscombe plot constructed with *Pearson* residuals which are the residuals standardized by dividing by the estimated standard deviation of the response variable. Standardizing the residuals helps us to visualize whether the expected value of the residuals is zero for any given level of the linear predictor. The Cook and leverage plots to help identify influential outliers also applies for GLMs.

4.5 GLMs in R

We use the glm() function to fit our generalized linear model. Like the lm() function the glm() function needs at least *formula* and *data* parameters where:

- **formula** could, for example, be y~poly(x,2)+ I(z^2) + I(w==0) + I(w>0):I(x<0) + I(w>0)*I(z<0), where,
  - the I() term is used to transform existing variables,
  - the poly(x,n) term, creates a polynomial of degree n: $x + x^2 + ... + x^n$,
  - I(w>0):I(x<0) term, creates an interaction term between the indicator functions $1_{w>0}(w)$ and $1_{x<0}(x)$, and
  - I(w>0)*I(z<0) term, creates the indicator functions $1_{w>0}(w)$ and $1_{z<0}(x)$ and additionally their interaction.
- **data** indicates the data.frame where the x,y,z,w variables are located.

Additionally glm() needs a *family* parameter which indicates the type of dependent variable and the link function to be used. For example:

- family= "binomial" or family=binomial("logit") indicates a binary dependent variable with logit link function,
- family= binomial("probit") indicates a binary dependent variable with probit link function, and

- family= poisson indicates a poisson dependent variable (values in 0,1,2,…) with a log link function.

As in the case with other R fitting functions which create *model* objects, the glm objects created by the fitting function *glm* has available certain standard functions:

- **summary**: gives a summary of estimated parameters, their significance and the overall significance of the model,
- **plot**: helps visualize he fitted model and/or gives model diagnostics to assess the validity of model assumptions, and
- **predict** applies the model to new data points, provided that the model predictors are available, to predict the dependent variable (and probabilites in the case of glm objects).

The script will also provide a graph of the response curves along the gradient of each predictor.

5 Fit Model

Here, we will select one of the butterfly species: *Glaucopsyche alexis.* You can also perform the steps for another species.

Question 5.0.1 *Observe to distribution points of Glaucopsyche alexis over Switzerland. What can you conclude on the ecology of this species? Is it a habitat specialist or a generalist? What environmental factors could be limiting its distribution?*

Run a GLM for each predictor separately. They will take the form:

$$g(p) = f(x) + \epsilon = \beta_0 + \beta_1 x + \beta_2 x^2 + ... + \beta_n x^n + \epsilon \tag{14}$$

where

- p is the response variable, in this case, the probability that a binary variable y takes the value 1, indicating the presence of the species Glaucopsyche alexis,
- $g(p)$ is a link function, in this case log-odds ratio $\frac{p}{1-p}$,
- x is the predictor, for example, temperature or precipitation,
- $f(x)$ is a polynomial in x, and
- n is the order of the polynomial used, in this case $n = 2$.

The code will also provide a graph of the response curves along the gradient of each predictor.

5.1 Prepare dataframe

We consider the species *Glaucopsyche_alexis.*

```
G_alexis <- Occurrences[Occurrences$Species == "Glaucopsyche_alexis", ]
```

Create a dataframe of presences for *Glaucopsyche alexis*:

```
Presences <- data.frame(cbind(G_alexis[, c("X", "Y", "DDEG", "MIND", "SRAD")],
    PRES = rep(1, nrow(G_alexis))))
```

Create a dataframe of pseudoabsences for *Glaucopsyche alexis*:

```
Absences <- data.frame(cbind(CellsXY_red[, c("x", "y", "DDEG", "MIND", "SRAD")],
    PRES = rep(0, nrow(CellsXY_red))))
colnames(Absences) <- colnames(Presences)
```

Collate the two dataframes:

```
Training <- na.omit(rbind(Presences, Absences))
```

5.2 Fit GLM models

Fit a General Linear Model (GLM) based on **temperature** only:

```
glm.uni <- glm(PRES ~ poly(DDEG, 2), data = Training, family = binomial, maxit = 100)
pander(summary(glm.uni)$coefficients, caption = "Summary for model based on degree-days")
```

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-0.1187	0.06582	-1.804	0.07125
poly(DDEG, 2)1	-0.4212	2.063	-0.2041	0.8383
poly(DDEG, 2)2	-8.845	2.101	-4.21	2.552e-05

Table 3: Summary for model based on degree-days

Plot the model for probability of presence and data points: presences in red and absences in blue:

```
par(mfrow = c(1, 1))
data_plot <- data.frame(cbind(Training$DDEG, predict(glm.uni,
    type = "response")))
sort1 <- na.omit(data_plot[order(data_plot[, 1], decreasing = FALSE),
    ])
data_plot <- data.frame(cbind(sort1[, 1], sort1[, 2]))
plot(data_plot[, 1], data_plot[, 2], xlab = expression("Degree" -
    days ~ degree ~ C), ylab = "Probability of occurrence", frame.plot = F,
    type = "l", ylim = c(0, 1))
# Plot the presences in red and absences in blue
points(Training$DDEG[Training$PRES == 1], Training$PRES[Training$PRES ==
    1], col = "red")
points(Training$DDEG[Training$PRES == 0], Training$PRES[Training$PRES ==
    0], col = "blue")
```

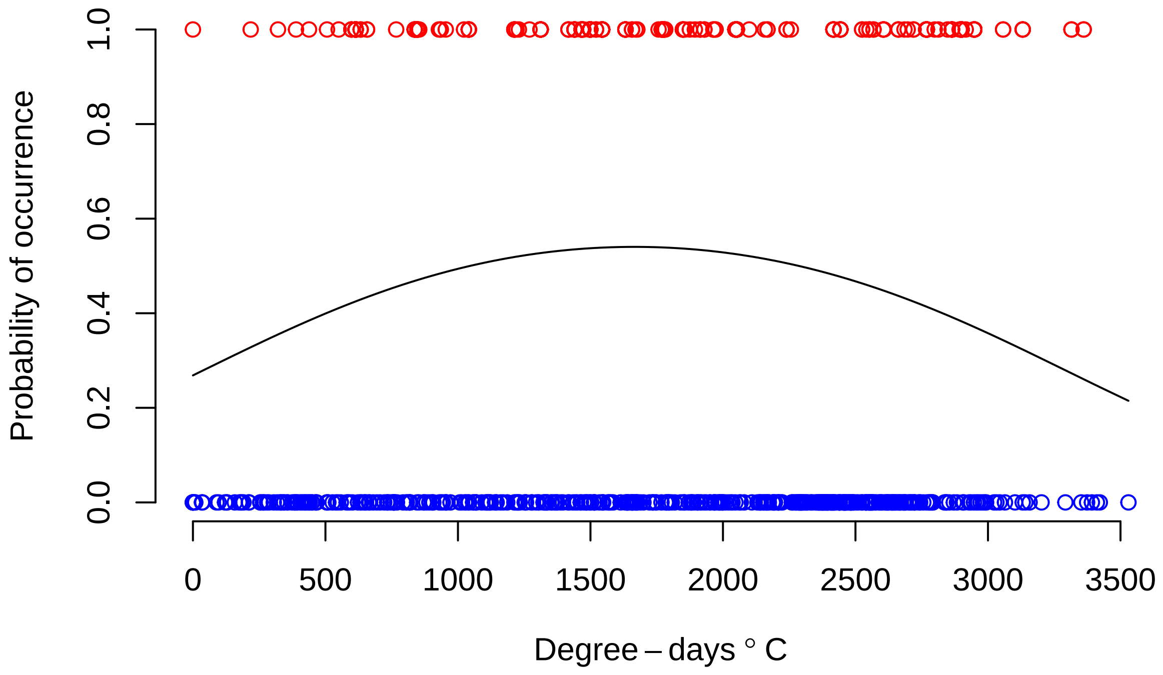


Figure 7: Response curve for degree-day model

Fit a General Linear Model (GLM) based on **precipitation** only:

```
glm.uni <- glm(PRES ~ poly(MIND, 2), data = Training, family = binomial, maxit = 100)
pander(summary(glm.uni)$coefficients, caption = "Summary for model based on moisture index")
```

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-0.1187	0.06661	-1.782	0.07483
poly(MIND, 2)1	-12.84	2.117	-6.065	1.322e-09
poly(MIND, 2)2	1.51	2.151	0.7017	0.4828

Table 4: Summary for model based on moisture index

Plot the model for probability of presence and data points: presences in red and absences in blue:

```
data_plot <- data.frame(cbind(Training$MIND, predict(glm.uni,
    type = "response")))
sort1 <- na.omit(data_plot[order(data_plot[, 1], decreasing = FALSE),
    ])
data_plot <- data.frame(cbind(sort1[, 1], sort1[, 2]))
plot(data_plot[, 1], data_plot[, 2], xlab = "Moisture index",
    ylab = "Probability of occurrence", frame.plot = F, type = "l",
    ylim = c(0, 1))
# Plot the presences in red and absences in blue
points(Training$MIND[Training$PRES == 1], Training$PRES[Training$PRES ==
    1], col = "red")
```

```r
points(Training$MIND[Training$PRES == 0], Training$PRES[Training$PRES ==
    0], col = "blue")
```

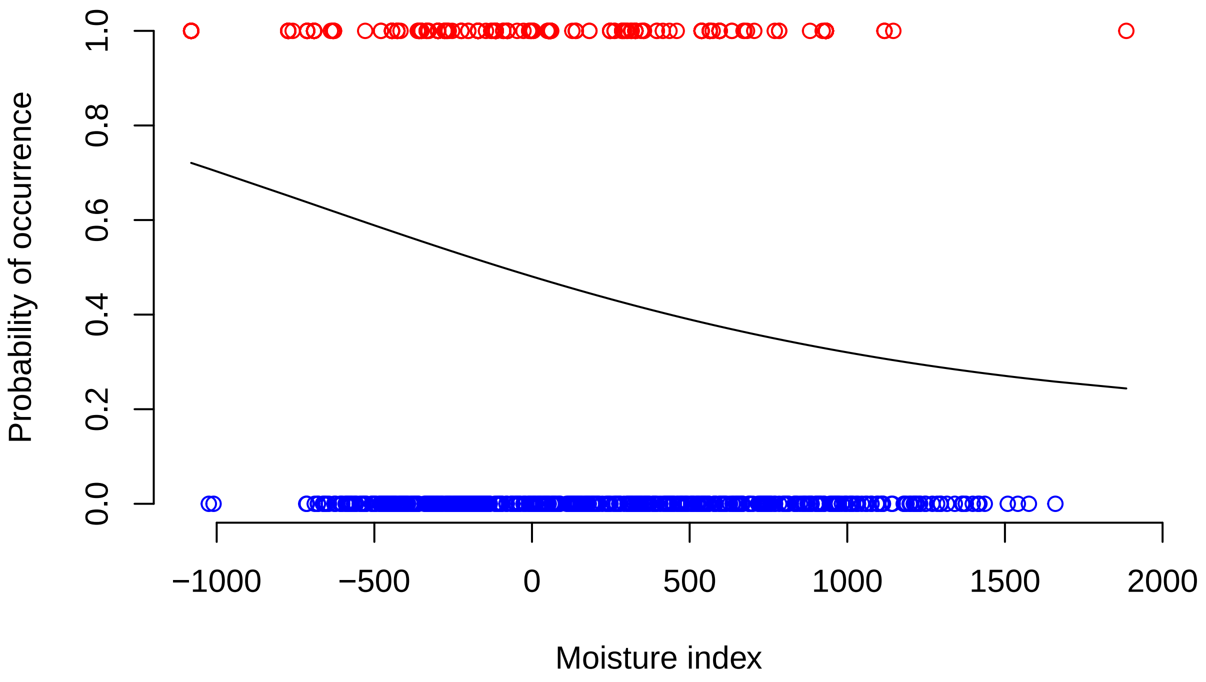


Figure 8: Response curve for moisture index model

Question 5.2.1 *Provide some general comments about the different response curves. Try to relate the shape to the distribution and the ecology of the species and the type of the predictors (temperature-, precipitation-based).*

Now run the GLM including the three climatic predictors using linear and quadratic terms.

Fit a multivariate General Linear Model (GLM):

```r
glm.multi <- glm(PRES ~ poly(DDEG, 2) + poly(MIND, 2) + poly(SRAD, 2), data = Training,
    family = binomial, maxit = 100)
pander(summary(glm.multi)$coefficients, caption = "Summary for multivariate model, climatic predictors"
```

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-0.1559	0.07167	-2.175	0.02962
poly(DDEG, 2)1	-30.71	4.231	-7.259	3.891e-13
poly(DDEG, 2)2	-5.805	3.356	-1.73	0.08366
poly(MIND, 2)1	-41.76	4.408	-9.474	2.686e-21
poly(MIND, 2)2	9.4	3.177	2.959	0.003089
poly(SRAD, 2)1	-12.98	2.503	-5.187	2.142e-07
poly(SRAD, 2)2	-8.617	2.364	-3.645	0.0002672

Table 5: Summary for multivariate model, climatic predictors

Perform model diagnostics (Tukey-Anscombe and outlier-leverage plots):

```
library(boot)
# glm diagnostics- in particular for cook and leverage plot
diags <- glm.diag(glm.multi)
n <- nrow(Training)
p <- glm.multi$df.null - glm.multi$df.residual + 1
hlin <- 8/(n - 2 * p)
vlin <- 2 * p/(n - 2 * p)

par(mfrow = c(1, 2))
# Tukey-Anscombe plot with pearson residuals
xx <- predict(glm.multi, type = "link")
yy <- residuals(glm.multi, type = "pearson")
plot(xx, yy, xlab = "linear predictor", ylab = "Pearson residuals")
ls <- loess.smooth(xx, yy, family = "gaussian", )
lines(ls$x, ls$y, col = "red")
abline(h = 0, lty = 3)
# Cook distance and leverage plot
plot(diags$h/(1 - diags$h), diags$cook, xlab = "h/(1-h)", ylab = "Cook statistic")
abline(h = hlin, v = vlin, lty = 2)
```

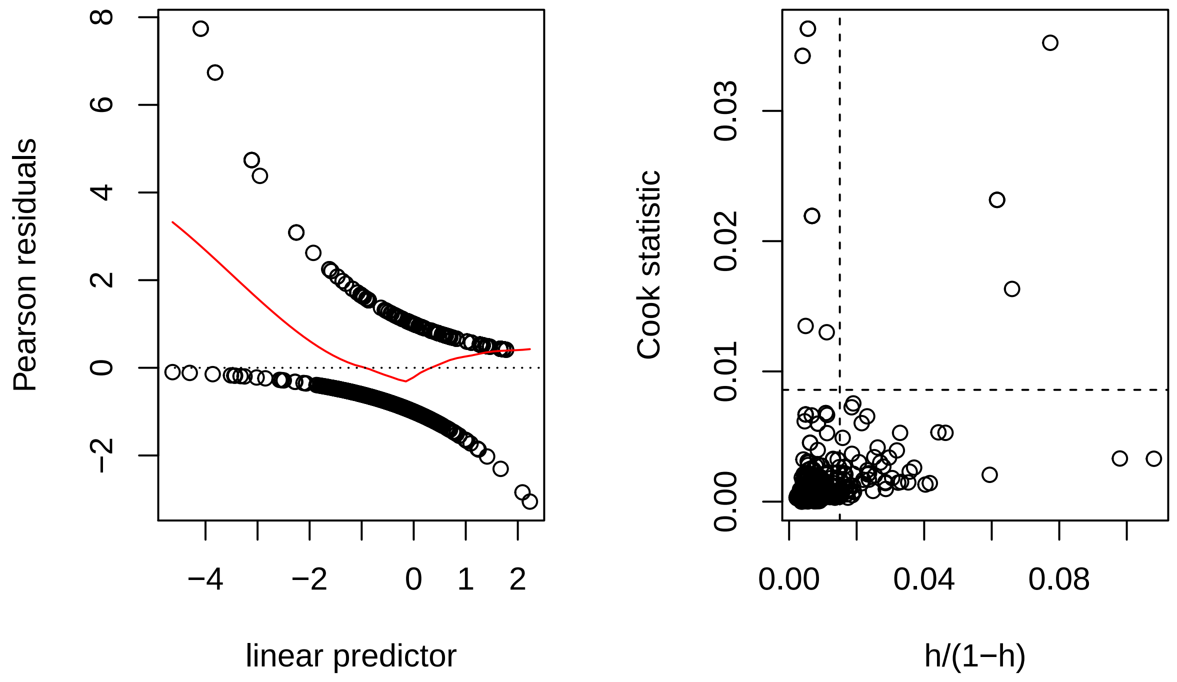


Figure 9: Model diagnostic plots, multivariate model, climatic predictors

We can see that the model is very biased meaning it will not give us a good spatial description of the butterfly distribution. This could be because we have left out an important predictor. We try to fix this by including elevation as a predictor.

Load elevation raster:

```r
dem <- raster("../data/mnt25_1km")
```

Plot elevation:

```r
par(mar = c(0, 0, 0, 0))
plot(dem, box = F, axes = F)
```

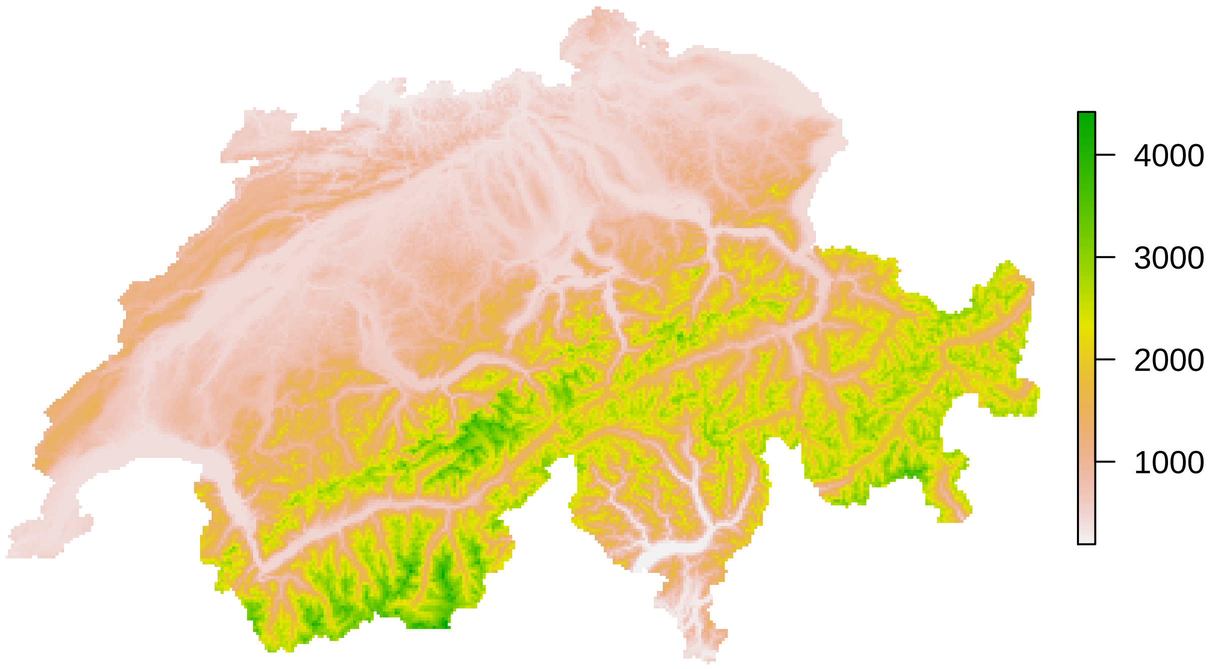


Figure 10: Elevation raster

Add elevation to training set:

```r
Training <- cbind(Training, elev = extract(dem, Training[, c("X", "Y")]))
```

Fit multivariate model with elevation:

```r
glm.multi <- glm(PRES ~ poly(DDEG, 2) + poly(MIND, 2) + poly(SRAD, 2) + poly(elev,
    2), data = Training, family = binomial, maxit = 100)
pander(summary(glm.multi)$coefficients, caption = "Summary for multivariate model, climatic predictors a
```

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-0.09637	0.08322	-1.158	0.2468
poly(DDEG, 2)1	122.3	12.75	9.588	9.019e-22
poly(DDEG, 2)2	1.847	5.279	0.3499	0.7264
poly(MIND, 2)1	-46.01	5.154	-8.928	4.347e-19
poly(MIND, 2)2	11.1	4.001	2.775	0.005523

	Estimate	Std. Error	z value	Pr(>|z|)
poly(SRAD, 2)1	-10.29	2.785	-3.696	0.0002187
poly(SRAD, 2)2	-7.349	2.655	-2.768	0.005647
poly(elev, 2)1	158.1	13.36	11.84	2.414e-32
poly(elev, 2)2	-27.43	5.44	-5.042	4.601e-07

Table 6: Summary for multivariate model, climatic predictors and elevation

Perform diagnostics for new model:

```r
diags <- glm.diag(glm.multi)
n <- nrow(Training)
p <- glm.multi$df.null - glm.multi$df.residual + 1
hlin <- 8/(n - 2 * p)
vlin <- 2 * p/(n - 2 * p)

par(mfrow = c(1, 2))
# Tukey-Anscombe plot with pearson residuals
xx <- predict(glm.multi, type = "link")
yy <- residuals(glm.multi, type = "pearson")
plot(xx, yy, xlab = "linear predictor", ylab = "Pearson residuals")
ls <- loess.smooth(xx, yy, family = "gaussian", )
lines(ls$x, ls$y, col = "red")
abline(h = 0, lty = 3)
# Cook distance and leverage plot
plot(diags$h/(1 - diags$h), diags$cook, xlab = "h/(1-h)", ylab = "Cook statistic")
abline(h = hlin, v = vlin, lty = 2)
```

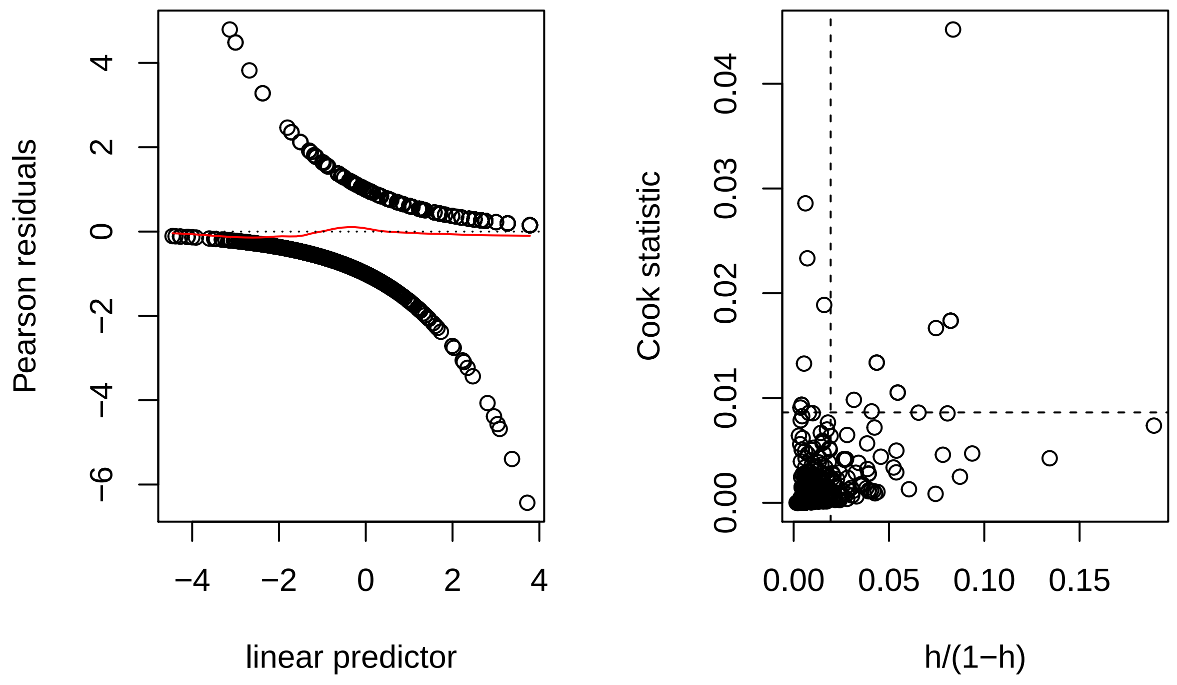


Figure 11: Model diagnostic plots, multivariate model, climatic predictors and elevation

We can see that the new model is acceptable in terms of bias: there is no longer serious systematic error in estimation. It is somewhat surprising that elevation worked in removing the bias from the model since we had already included degree-days and we know that a lot of the information contained in elevation is also contained in temperature variables. This can be seen from scatter plot and correlations between the two variables.

Plot elevation versus degree days and compute pearson correlation between the two:

```
plot(values(dem), values(DDEG))
```

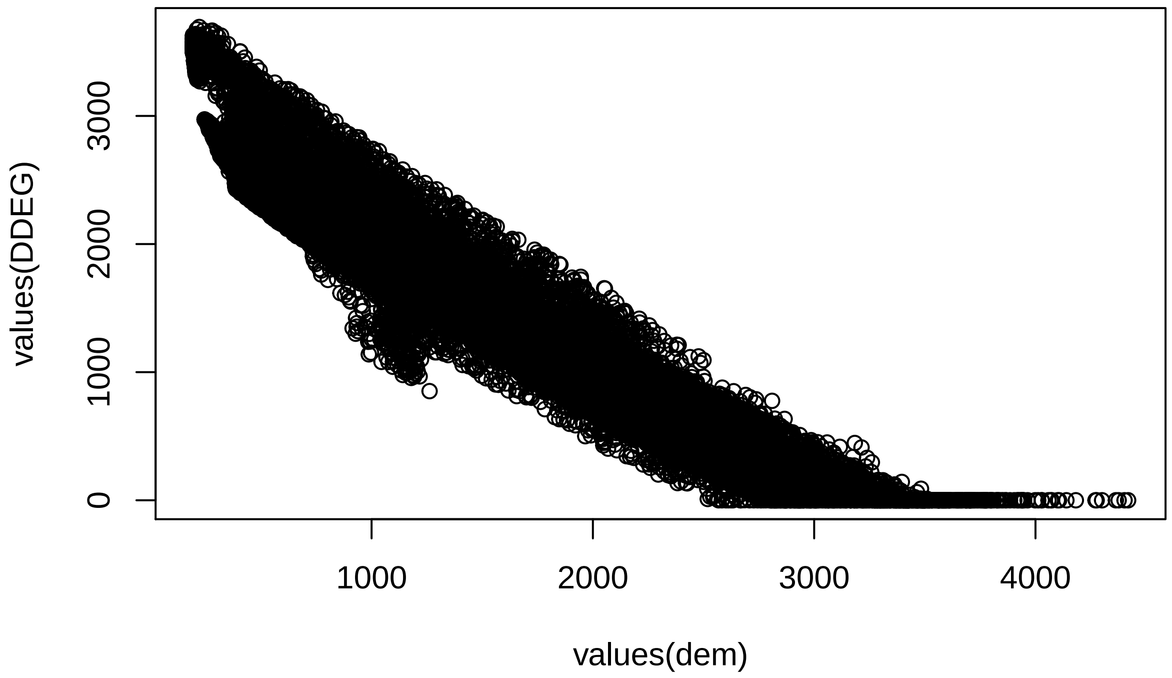


Figure 12: Scatterplot for elevation vs. degree-days

```r
cor(values(dem), values(DDEG), use = "pairwise.complete")
```

```
## [1] -0.9699703
```

We corroborate that the absolute correlation is high. However, we have looked at the correlation between the variables in the whole of Switzerland. Perhaps for the places where the butterflies were observed the relationship between degree-days and elevation is different.. We will verify this by constructing a model to explain how degree-days depends on elevation and comparing the spatial distribution of the residuals of this model with the distribution of observed butterflies. If we see that butterflies were observed at sites with high residuals we will know why we need both degree-days and elevation in the model.

Fit model of degree-days based on elevation:

```r
DF <- na.omit(data.frame(coordinates(dem), temp = values(DDEG), elev = values(dem)))
fit <- lm(temp ~ poly(elev, 2), data = DF)
pander(summary(fit), caption = "Summary of model for degree-days based on elevation")
```

	Estimate	Std. Error	t value	Pr(>|t|)
poly(elev, 2)1	-170817	188.5	-906.4	0
poly(elev, 2)2	17516	188.5	92.94	0
(Intercept)	1729	0.9087	1903	0

Observations	Residual Std. Error	R^2	Adjusted R^2
43018	188.5	0.9507	0.9507

Observations	Residual Std. Error	R^2	Adjusted R^2

Table 8: Summary of model for degree-days based on elevation

Create a raster of residuals

```r
res <- cbind(DF[, c("x", "y")], z = residuals(fit))
res.r <- merge(coordinates(dem), res, by = c("x", "y"), all = T)
res.r <- rasterFromXYZ(res.r)
```

Interpolate residuals at the location of observed butterflies

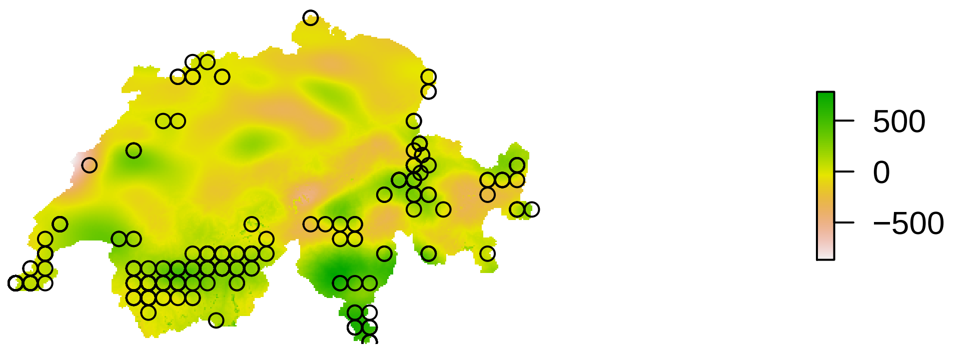
```r
library(gstat)
int <- idw(formula = z ~ 1, locations = ~x + y, data = res, newdata = data.frame(x = Presences$X,
    y = Presences$Y), idp = 2)
```

```
## [inverse distance weighted interpolation]
```

Compare residuals at locations of observed butterflies to residuals in general

```r
df.barplot <- data.frame(origin = c(rep("Switzerland", nrow(DF)), rep("Butterfly occurrence locations",
    nrow(Presences))), res = c(res$z, int$var1.pred))

par(mar = c(2.5, 1, 0, 1), mfrow = c(2, 1))
plot(res.r, axes = F, box = F)
points(Presences[, c("X", "Y")], col = "black", pch = 1)
boxplot(df.barplot$res ~ df.barplot$origin)
```



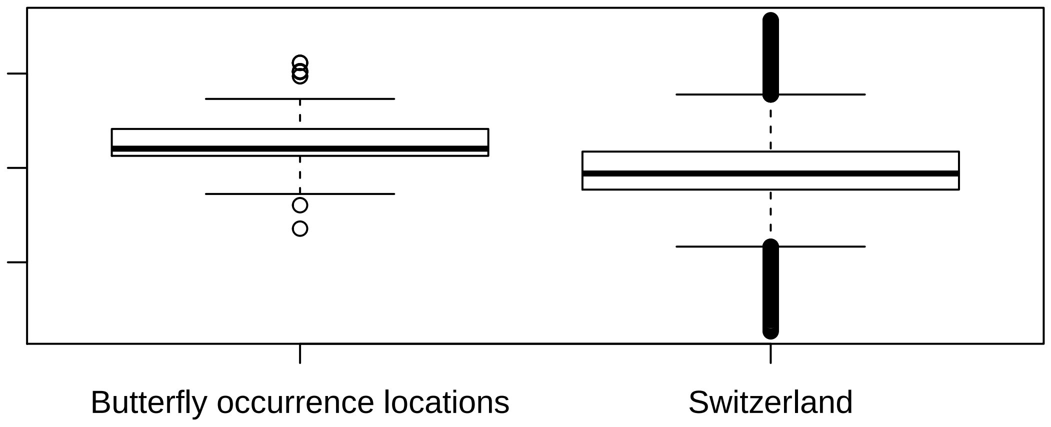


Figure 13: Analysis of spatial distribution of residuals (top) and comparison between residuals in the whole of Switzerland and those at observation locations (bottom)

We confirm that at observation locations elevation and degree-days tend to have a different relationship to that in the rest of Switzerland.

We can now have a look at where the outliers of the GLM model are located.

Identify influential outliers:

```r
indx.out <- which((diags$h/(1 - diags$h)) > vlin & diags$cook > hlin)
outs <- Training[indx.out, ]
```

Locate outliers:

```r
par(mar = c(0, 0, 0, 0))
plot(DDEG, col = rev(heat.colors(20)), legend = T, axes = F, box = F)
points(Training[, c("X", "Y")], col = c("black", "blue")[Training$PRES + 1],
    pch = 3, cex = 0.5)
points(outs[, c("X", "Y")], cex = 3, col = "blue")
legend(740000, 1e+05, legend = c("occurrence", "pseudo-absence", "outlier"),
    col = c("blue", "black", "blue"), pch = c(3, 3, 1), cex = c(1, 1, 1))
```

```
## Warning in if (xc < 0) text.width <- -text.width: the condition has length
## > 1 and only the first element will be used
```

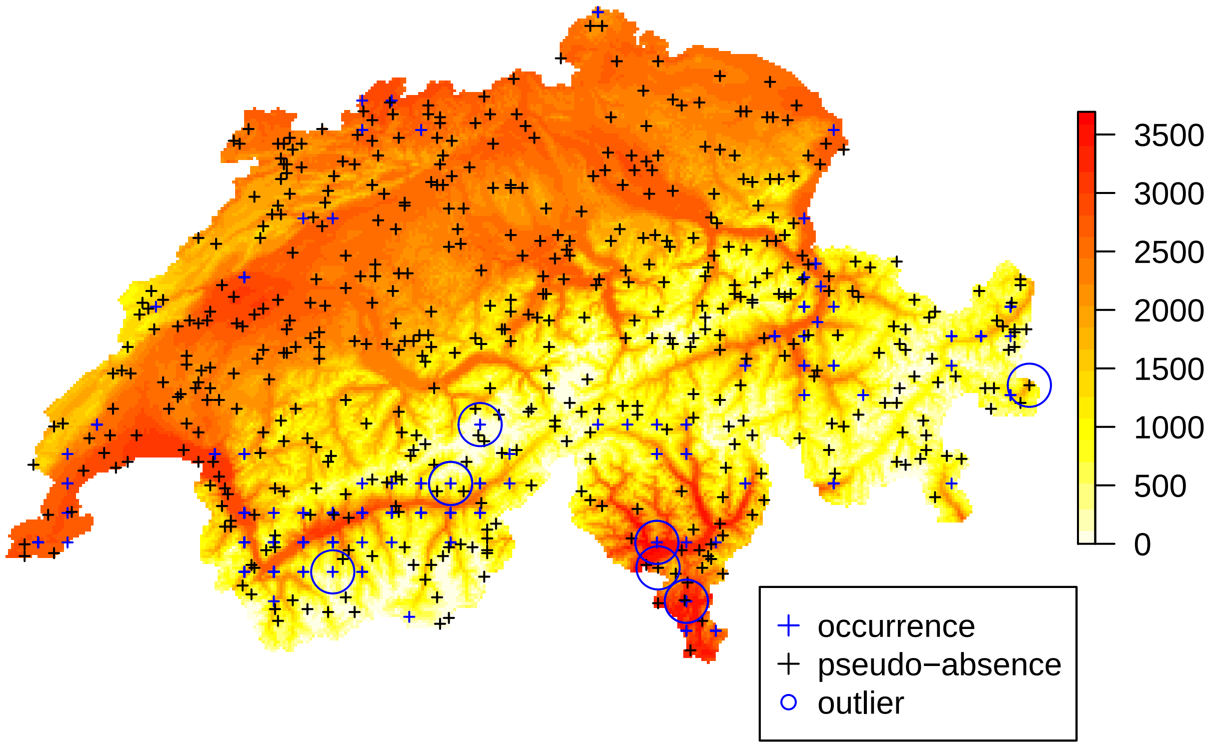


Figure 14: Degree-day raster and outlier location

We could investigate outliers further based on their location. What is special about these locations in terms of butterfly distribution and the factors that influence it? For the purpose of this practical we move on without taking further action regarding outliers.

6 Projection of the species distribution model over Switzerland

Now you can visualize the values expected from the *multivariate* model for all Switzerland.

Create the projection dataframe:

```
Projection <- na.omit(data.frame(CellsXY$x, CellsXY$y, extract(DDEG, CellsXY[,
    c("x", "y")]), extract(MIND, CellsXY[, c("x", "y")]), extract(SRAD, CellsXY[,
    c("x", "y")]), extract(dem, CellsXY[, c("x", "y")])))
names(Projection) <- c("X", "Y", "DDEG", "MIND", "SRAD", "elev")

ProjectionYX <- Projection[, 2:1]
Projection <- Projection[, 3:6]
```

Project multivariate model over Switzerland:

```
library(SDMTools)
```

```
pred.glm.multi <- predict(glm.multi, Projection, type = "response")
proj.glm.multi <- cbind(ProjectionYX, pred.glm.multi)
dataframe2asc(proj.glm.multi, filenames = "Glaucopsyche_alexis", outdir = "../data")
```

```r
Glaucopsyche_alexis <- raster("../data/Glaucopsyche_alexis.asc")
par(mar = c(0.5, 0.5, 0.5, 0.5))
plot(Glaucopsyche_alexis, axes = F, box = F, col = brewer.pal(9, "PuRd"))
points(G_alexis$X, G_alexis$Y, col = "blue")
```

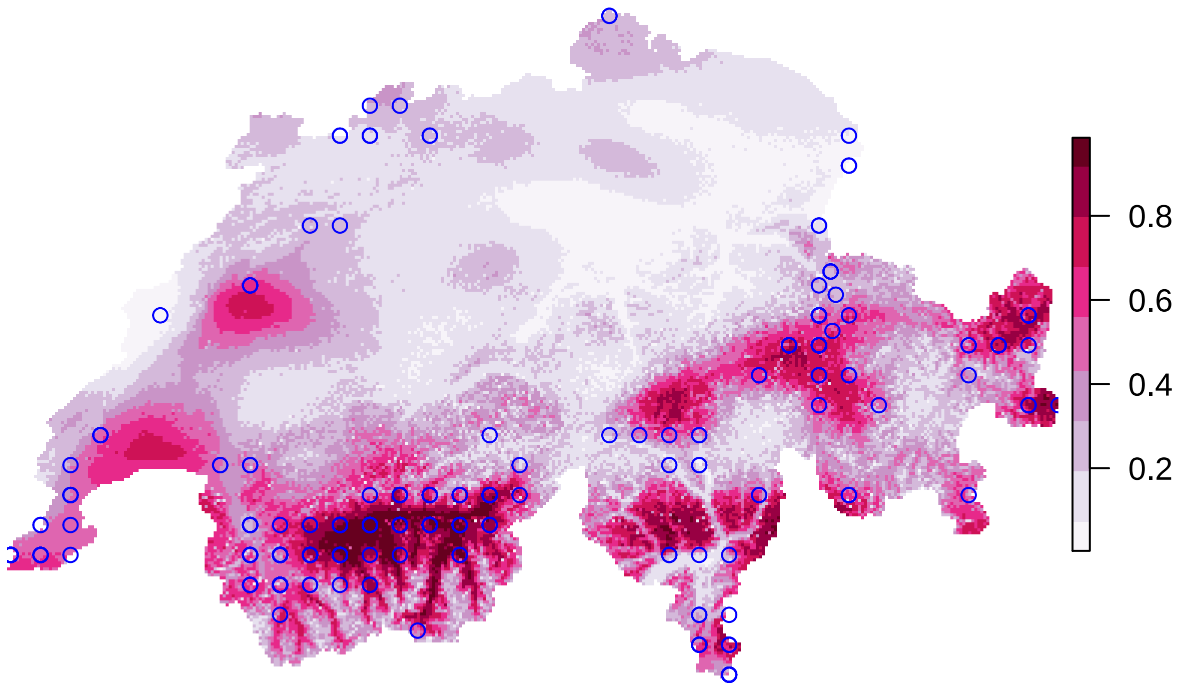


Figure 15: Projected probability of occurrence

Question 6.0.2 *Based on the distribution map, what can you conclude about the biology of the species?*

7 Model evaluation

You can use a contingency table to calculate a number of useful metrics (see Fielding & Bell 1997 and Allouche et al. 2006 for further details). The structure of a contingency table is usually formalized as follows:

predicted	Observed		Total
Null/Alternative	0	1	
0	U	T	W
1	V	S	R
Total	N_0	N_1	N

Table 9: Contingency table

In the following table some useful metrics based on the contingency table are summarized.

Metric	Formula
Prevalence	$\frac{N_1}{N}$
Correct classification rate	$\frac{S+U}{N}$
Mis-classification rate	$\frac{V+T}{N}$
Positive predictive power	$\frac{S}{R}$
Negative predictive power	$\frac{U}{W}$
Sensitivity (True positive rate)	$\frac{S}{N_1}$
Specificity (True negative rate)	$\frac{U}{N_0}$
False positive rate	$\frac{V}{N_0}$
False negative rate	$\frac{T}{N_1}$
Odds ratio	$\frac{U*S}{T*V}$
Kappa	$\frac{\frac{U+S}{N}-\frac{R*N_1+W*N_0}{N^2}}{1-\frac{R*N_1+W*N_0}{N^2}}$
True Skill Statistic (TSS)	$\frac{U*S-T*V}{(S+T)*(U+V)}$ = sensitivity + specificity -1

Table 10: Accuracy metrics

Kappa and TSS are among the most widely used metrics. They both give equal weight to correct prediction of presence and absence and also correct for chance performance. All the measures above can be easily obtained using the custom function meva.table() that gives you both the contingency table and metrics. In the example below we consider the multivariate GLM model and an initial threshold of 0.5 for transforming probabilities into binary response variables. The custom function meva.table() takes 3 arguments: a vector with the predicted values, a vector with the observed values and a threshold value used to transform the predicted values into binaries.

```
data_validation <- data.frame(cbind(Training$PRES, predict(glm.multi, type = "response")))
colnames(data_validation) <- c("Observed", "Projected")
```

Predictions generated by most modeling techniques are on a continuous scale between 0 and 1, while the observations are binary (i.e. 0 or 1). To compare the values we thus need to change predictions from continuous to binary meaning we need a threshold to change the probabilities to zeros or ones. Let's start with a simple arbitrary threshold of 0.5: everything that is smaller or equal to 0.5 becomes 0 and everything that is bigger than 0.5 becomes 1. You can use the simple standard command table() to quickly build such a contingency table:

The Functions.r file contains necessary functions for model evaluation. The meva.table() function takes three arguments:

- Projected probability of occurrence,
- Observed occurrence/absence, and,
- threshold value.

It first transforms the probabilites into occurrences or absences using the threshold value, builds a contingency table as in table 5 and calculates the accuracy metrics of table 6.

Load Functions.r file with necessary functions for model evaluation. Compute the table of accuracy statistics with an arbitrary threshold of 0.5:

```r
source("../R/Functions.r")
meva.table(data_validation[, 2], data_validation[, 1], 0.5)
```

```
## $CONTINGENCY_TABLE
##                  Observed values
## Predicted values   0   1
##            FALSE 397 144
##            TRUE  103 302
##
## $EVALUATION_METRICS
##    Metric                           Value
## 1  "Prevalence"                     "0.4715"
## 2  "Correct classification rate"    "0.7389"
## 3  "Mis-classification rate"        "0.2611"
## 4  "Sensitivity"                    "0.6771"
## 5  "Specificity"                    "0.794"
## 6  "Positive predictive power"      "0.206"
## 7  "Negative predictive power"      "0.3229"
## 8  "False positive rate"            "0.7457"
## 9  "False negative rate"            "0.7338"
## 10 "Odds Ratio"                     "8.0835"
## 11 "Kappa"                          "0.4735"
## 12 "Normalized mutual information" "0.8302"
## 13 "True skill statistic"           "0.4711"
```

Now you can calculate evaluation metrics for a series of possible thresholds on the training dataset. We then choose the threshold that optimizes a given metric on this set and use it to calculate the same metric on the independent evaluation dataset. For example, we may want to maximize TSS. The custom function max.TSS calculates TSS for all threshold values between 0 and 1, for 0.01 increment steps and automatically returns the maximum value of TSS and the corresponding threshold. Similarly, the plot.tss() function calculates the TSS for all threshold values between 0 and 1, for 0.01 increment steps and plots the corresponding curve.

```r
plot.tss(data_validation[, 2], data_validation[, 1])
```

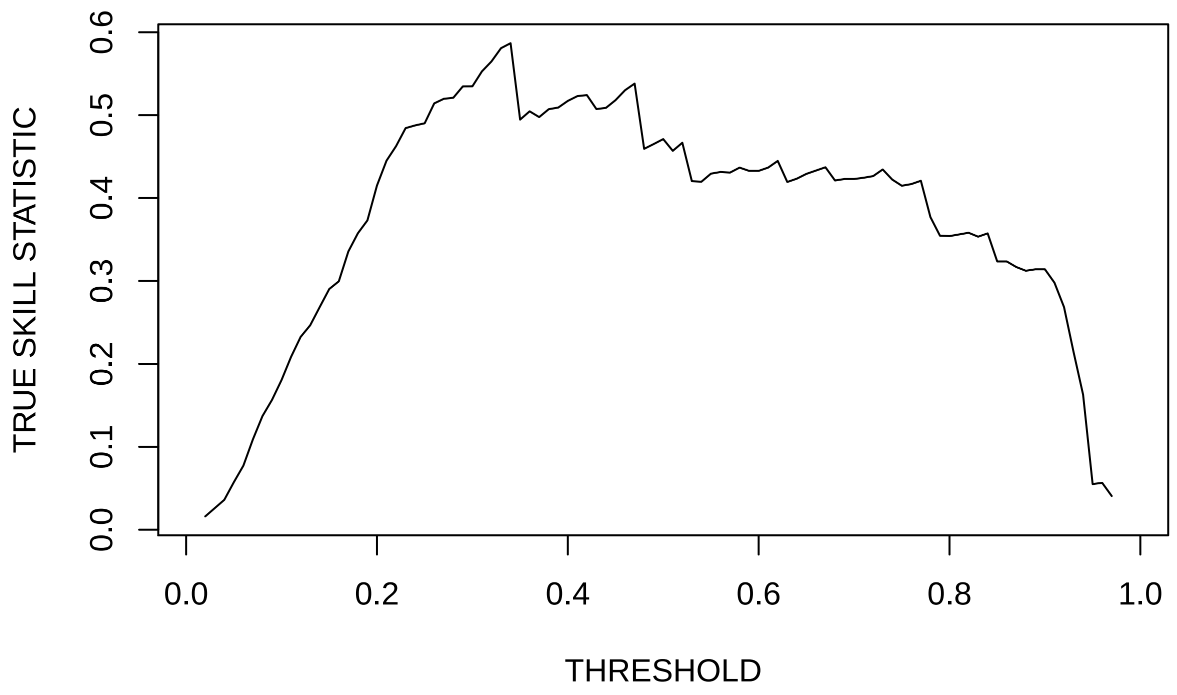


Figure 16: TSS vs. threshold

```
max_TSS <- max.TSS(data_validation[, 2], data_validation[, 1])
max_TSS[[2]]
```

```
##      [,1]                       [,2]
## [1,] "Maximum TSS"              "0.5868"
## [2,] "Correspondent threshold"  "0.34"
```

Compute the table of accuracy statistics with the optimized threshold of 0.34:

```
meva.table(data_validation[, 2], data_validation[, 1], as.numeric(max_TSS[[2]][2,
    2]))
```

```
## $CONTINGENCY_TABLE
##                 Observed values
## Predicted values   0   1
##            FALSE 336  38
##            TRUE  164 408
##
## $EVALUATION_METRICS
##    Metric                          Value
## 1  "Prevalence"                    "0.4715"
## 2  "Correct classification rate"   "0.7865"
## 3  "Mis-classification rate"       "0.2135"
## 4  "Sensitivity"                   "0.9148"
## 5  "Specificity"                   "0.672"
## 6  "Positive predictive power"     "0.328"
## 7  "Negative predictive power"     "0.0852"
## 8  "False positive rate"           "0.7133"
## 9  "False negative rate"           "0.8984"
## 10 "Odds Ratio"                    "21.9974"
## 11 "Kappa"                         "0.578"
## 12 "Normalized mutual information" "0.7118"
## 13 "True skill statistic"          "0.5868"
```

Now you can compute the binary distribution map by converting the probabilities to binary values using the optimal threshold computed above.

Convert probabilities in presence and absence:

```
Glaucopsyche_alexis_binary <- Glaucopsyche_alexis > as.numeric(max_TSS[[2]][2,
    2])
par(mar = c(0, 0, 0, 0), xpd = F)
plot(Glaucopsyche_alexis_binary, box = F, axes = F, legend = F, col = c("grey",
    "green4"))
points(G_alexis[, c("X", "Y")], col = "blue")
legend(680000, 85000, legend = c("predicted occurrence", "predicted absence"),
    fill = c("green4", "grey"), bg = "white")
legend(530000, 85000, legend = c("observed occurrence"), col = c("blue"), pch = 1,
    bg = "white")
```

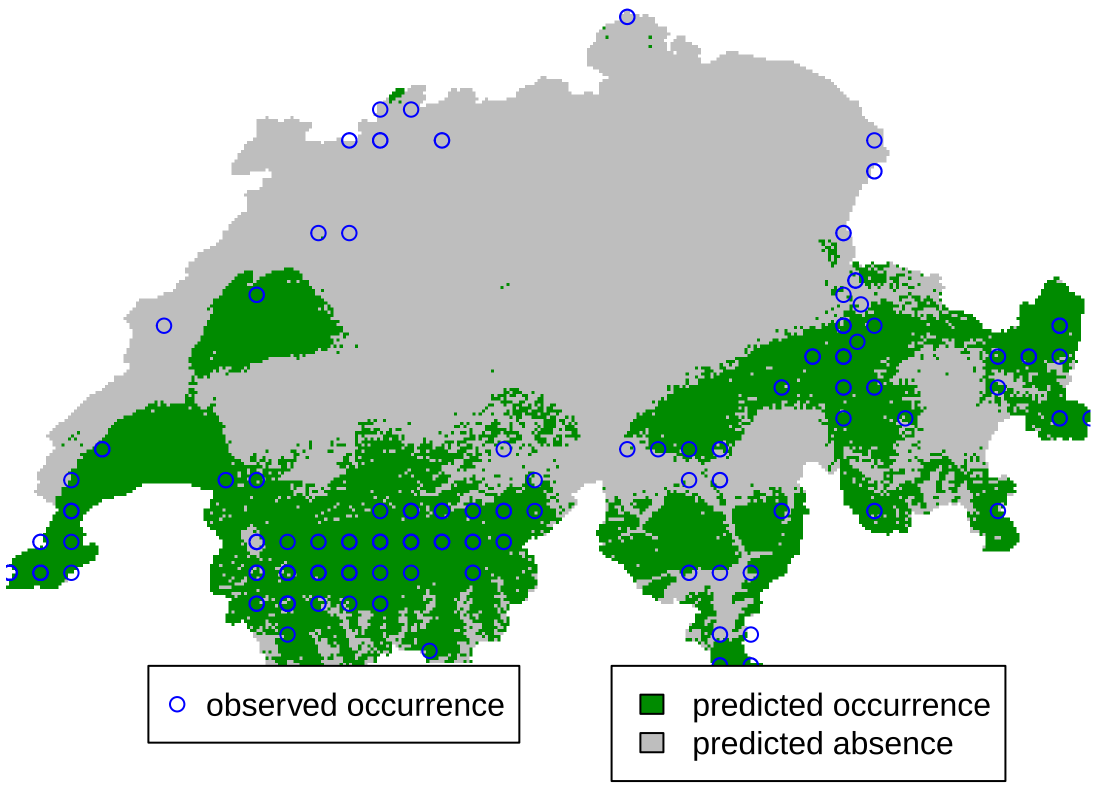


Figure 17: Projected presence/absence

Question 7.0.3 *Why is it useful to convert species probability of occurrence to presence and absence?*