Calibration of machine-learning probability predictions

Carsten F. Dormann
Biometry & Environmental System Analysis
University of Freiburg, Germany

December 30, 2019

Contents

1 *The analysis of Southern Whiteface (*Aphelocephala leucopsis*)*

1.1 *The data*

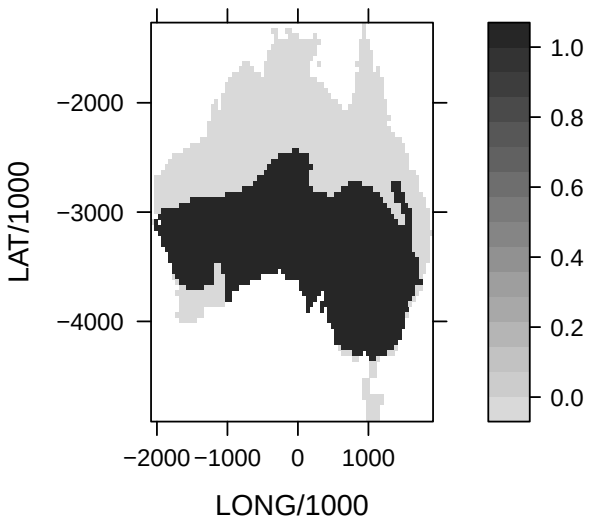
First, we load the data and plot the map of the whiteface's distribution; it becomes clear why it is called the "southern" whiteface.

```
load("SouthernWhiteface.Rdata")  # loads 'dats'
summary(dats)
```

```
      LONG                LAT               T_SEASON          TDIFF
Min.   :-2058250   Min.   :-4891404   Min.   :108.8   Min.   :10.18
1st Qu.: -808250   1st Qu.:-3441404   1st Qu.:424.1   1st Qu.:26.02
Median :   91750   Median :-2941404   Median :523.5   Median :29.24
Mean   :   34962   Mean   :-2909716   Mean   :495.5   Mean   :28.06
3rd Qu.:  891750   3rd Qu.:-2391404   3rd Qu.:589.5   3rd Qu.:31.10
Max.   : 1891750   Max.   :-1291404   Max.   :656.0   Max.   :34.10
   T_SUMMER         T_WINTER         PRE_YEAR        PRE_SUMMER
Min.   : 9.93   Min.   : 0.17   Min.   : 130.7   Min.   :  29.41
1st Qu.:25.51   1st Qu.:11.98   1st Qu.: 233.0   1st Qu.:  67.00
Median :28.57   Median :13.90   Median : 347.3   Median : 131.09
Mean   :27.49   Mean   :14.88   Mean   : 473.0   Mean   : 174.36
3rd Qu.:30.40   3rd Qu.:17.98   3rd Qu.: 616.8   3rd Qu.: 240.12
Max.   :33.20   Max.   :25.19   Max.   :2742.1   Max.   :1213.83
  PRE_WINTER          ELEV              SLOPE              ASPECT
Min.   :  2.06   Min.   : -12.3   Min.   :0.00000   Min.   :  0.07
1st Qu.: 23.00   1st Qu.: 130.6   1st Qu.:0.03000   1st Qu.:107.15
```

```
Median : 41.20   Median : 247.2   Median :0.05000   Median :198.67
Mean   : 66.40   Mean   : 278.0   Mean   :0.07403   Mean   :190.78
3rd Qu.: 79.49   3rd Qu.: 399.2   3rd Qu.:0.09000   3rd Qu.:279.63
Max.   :814.43   Max.   :1333.1   Max.   :0.59000   Max.   :360.00
   TMIN_JUL         TMIN_JAN        PERC_SEA          SHRUB
Min.   :-4.360   Min.   : 4.66   Min.   : 0.000   Min.   :0.0000
1st Qu.: 4.643   1st Qu.:18.91   1st Qu.: 0.000   1st Qu.:0.0600
Median : 6.035   Median :22.28   Median : 0.000   Median :1.0000
Mean   : 7.228   Mean   :21.10   Mean   : 1.599   Mean   :0.6719
3rd Qu.: 9.580   3rd Qu.:24.01   3rd Qu.: 0.000   3rd Qu.:1.0000
Max.   :21.600   Max.   :26.20   Max.   :96.180   Max.   :1.0000
   SAVANNA           GRASS              LAKE                CROP
Min.   :0.0000   Min.   :0.00000   Min.   :0.0000000   Min.   :0.00000
1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.0000000   1st Qu.:0.00000
Median :0.0000   Median :0.00000   Median :0.0000000   Median :0.00000
Mean   :0.1653   Mean   :0.04802   Mean   :0.0005804   Mean   :0.05754
3rd Qu.:0.0600   3rd Qu.:0.00000   3rd Qu.:0.0000000   3rd Qu.:0.00000
Max.   :1.0000   Max.   :1.00000   Max.   :0.2000000   Max.   :1.00000
    URBAN              MOSAIC               BARE
Min.   :0.0000000   Min.   :0.0000000   Min.   :0.000000
1st Qu.:0.0000000   1st Qu.:0.0000000   1st Qu.:0.000000
Median :0.0000000   Median :0.0000000   Median :0.000000
Mean   :0.0008606   Mean   :0.0007638   Mean   :0.007428
3rd Qu.:0.0000000   3rd Qu.:0.0000000   3rd Qu.:0.000000
Max.   :0.4300000   Max.   :0.4000000   Max.   :1.000000
    FOREST              ID        Aphelocephala.leucopsis
Min.   :0.00000   Min.   :1013   Min.   :0.0000
1st Qu.:0.00000   1st Qu.:1419   1st Qu.:0.0000
Median :0.00000   Median :2516   Median :1.0000
Mean   :0.03167   Mean   :3949   Mean   :0.5127
3rd Qu.:0.00000   3rd Qu.:6517   3rd Qu.:1.0000
Max.   :1.00000   Max.   :9948   Max.   :1.0000
```

```
library(lattice)
levelplot(Aphelocephala.leucopsis ~ LONG/1000 + LAT/1000, data = dats,
    col.regions = rev(gray(15:85/100)))   # map of summer rainfall (log)
```



1.2 *GLM*

Next, I prescribe a set of predictors to be used in all models. The aim is not the best possible model, but an illustration of the problem.

```
fglm ← step(glm(Aphelocephala.leucopsis ~ (TDIFF + PRE.SUMMER +
    T.SUMMER)^2 + I(TDIFF^2) + I(PRE.SUMMER^2) + I(T.SUMMER^2),
```

```
    family = binomial, data = dats), trace = F)
summary(fglm)
```

```
Call:
glm(formula = Aphelocephala.leucopsis ~ PRE_SUMMER + T_SUMMER +
    I(TDIFF^2) + I(PRE_SUMMER^2) + I(T_SUMMER^2) + PRE_SUMMER:T_SUMMER,
    family = binomial, data = dats)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.8958  -0.0741   0.1170   0.2957   3.3679

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)         -5.387e+01  4.488e+00 -12.003  < 2e-16 ***
PRE_SUMMER           1.048e-01  1.265e-02   8.278  < 2e-16 ***
T_SUMMER             4.038e+00  3.289e-01  12.278  < 2e-16 ***
I(TDIFF^2)           1.031e-02  9.030e-04  11.419  < 2e-16 ***
I(PRE_SUMMER^2)     -5.792e-05  1.243e-05  -4.661 3.14e-06 ***
I(T_SUMMER^2)       -8.315e-02  5.687e-03 -14.619  < 2e-16 ***
PRE_SUMMER:T_SUMMER -3.925e-03  3.770e-04 -10.409  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 4154.2  on 2997  degrees of freedom
Residual deviance: 1249.9  on 2991  degrees of freedom
AIC: 1263.9

Number of Fisher Scoring iterations: 9
```

```
library(verification)
roc.area(obs = dats$Aphelocephala.leucopsis, pred = predict(fglm,
    type = "response"))$A  # AUC
```

```
[1] 0.9712463
```

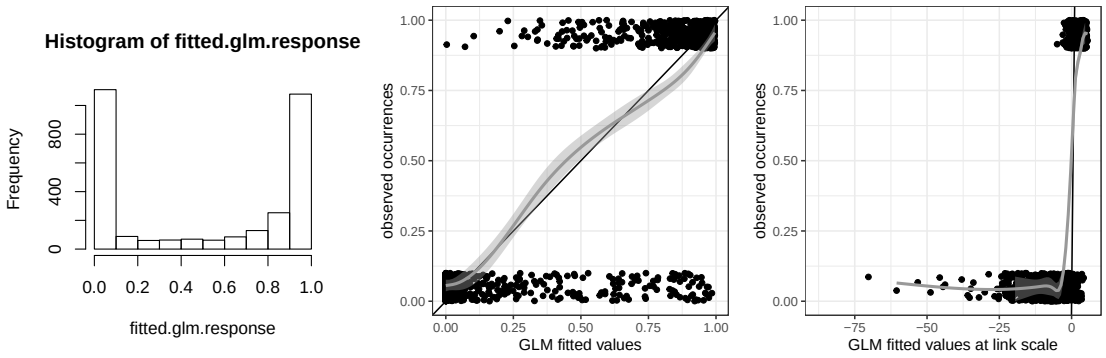
The model fits are stored and plotted against observed. A calibration line is computed and added.

```
fitted.glm.link <- predict(fglm)
fitted.glm.response <- plogis(fitted.glm.link)
hist(fitted.glm.response)
library(ggplot2)
library(gridExtra)  # for plot layout

pdata <- cbind.data.frame(fits = fitted.glm.response, dats)
a <- ggplot(pdata, aes(fits, jitter(Aphelocephala.leucopsis,
    0.5))) + scale_x_continuous("GLM fitted values") + scale_y_continuous("observed occurren
    limits = c(0, 1)) + geom_point() + geom_abline(slope = 1,
    intercept = 0, col = "black") + geom_smooth(col = "grey60") +
    theme_bw()

# same, but at link scale:
pdata2 <- cbind.data.frame(fits = fitted.glm.link, dats)
b <- ggplot(pdata2, aes(fits, jitter(Aphelocephala.leucopsis,
    0.5))) + scale_x_continuous("GLM fitted values at link scale") +
    scale_y_continuous("observed occurrences", limits = c(0,
        1)) + geom_point() + geom_abline(slope = 1, intercept = 0,
    col = "black") + geom_smooth(col = "grey60") + theme_bw()
a
```

```
`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

```
b
```

```
`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



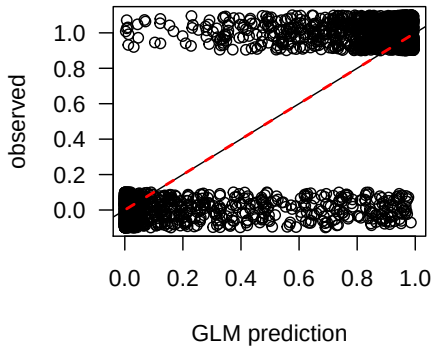
So the model predictions, grey line, are relatively close to the 1:1 line.

For calibration statistics, we now run a calibration regression. Note that we use fitted values at the link scale, so that now both the modelled and the predictor variable are at the link scale. Only if both are at the same scale can we expect a slope of 1 and an intercept of 0.

```
plot(jitter(dats$Aphelocephala.leucopsis, 0.5) ~ fitted.glm.response,
    las = 1, xlab = "GLM prediction", ylab = "observed")
abline(0, 1)  # ideal calibration is the 1:1 line
cali.glm ← glm(dats$Aphelocephala.leucopsis ~ fitted.glm.link,
    family = binomial)
summary(cali.glm)
```

```

Call:
glm(formula = dats$Aphelocephala.leucopsis ~ fitted.glm.link,
    family = binomial)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.8958  -0.0741   0.1170   0.2957   3.3679

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)     -7.437e-10  7.842e-02    0.00        1
fitted.glm.link  1.000e+00  4.169e-02   23.99   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 4154.2  on 2997  degrees of freedom
Residual deviance: 1249.9  on 2996  degrees of freedom
AIC: 1253.9

Number of Fisher Scoring iterations: 8
```

```
lines(plogis(seq(−5, 5, len = 100)), plogis(predict(cali.glm,
    newdata = data.frame(fitted.glm.link = seq(−5, 5, len = 100)))),
    col = "red", lty = 2, lwd = 2)
```



However, since this regression de facto assumes that a GLM can represent the deviations from the 1:1 line, its applicability is limited to diagnostics, not to fixing deviations.

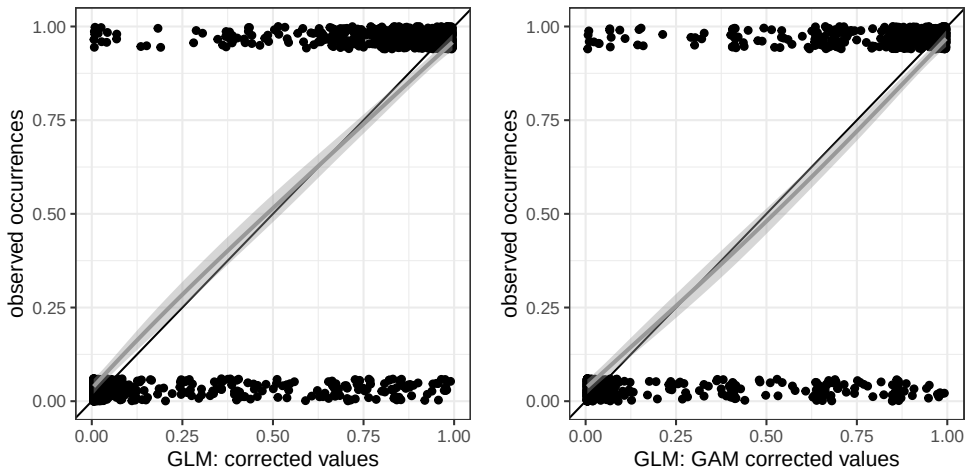
In this case, the slope is 1 and the intercept 0, indicating that the model predictions have no bias with respect to the observed data. That's how it should be. Calibration will not make much of a difference:

```
library(scam)
library(ggplot2)
cal.scam.glm ← scam(dats$Aphelocephala.leucopsis ~ s(fitted.glm.response,
    bs = "mpi"), family = binomial)
corr.fitted.glm.response ← predict(cal.scam.glm, type = "response")
pdata.corr ← cbind.data.frame(corr = corr.fitted.glm.response,
    dats)
ggplot(pdata.corr, aes(corr, jitter(Aphelocephala.leucopsis,
    0.3))) + scale_x_continuous("GLM: corrected values") + scale_y_continuous("observed occu
    limits = c(0, 1)) + geom_point() + geom_abline(slope = 1,
    intercept = 0, col = "black") + geom_smooth(col = "grey60") +
    theme_bw()
```

```
`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

```
# same with ordinary GAM, with slight shrinkage:
cal.gam.glm ← gam(dats$Aphelocephala.leucopsis ~ s(fitted.glm.response,
    bs = "ts", m = 3), family = binomial)
corr.fitted.glm.response ← predict(cal.gam.glm, type = "response")
pdata.corr ← cbind.data.frame(corr = corr.fitted.glm.response,
    dats)
ggplot(pdata.corr, aes(corr, jitter(Aphelocephala.leucopsis,
    0.3))) + scale_x_continuous("GLM: GAM corrected values") +
    scale_y_continuous("observed occurrences", limits = c(0,
        1)) + geom_point() + geom_abline(slope = 1, intercept = 0,
    col = "black") + geom_smooth(col = "grey60") + theme_bw()
```

```
`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



In this case it makes no difference, we'll have a look at it again in the next section.

1.3 *randomForest*

Now we do the same with a randomForest.

```
library(ranger)
frf <- ranger(as.factor(Aphelocephala.leucopsis) ~ TDIFF + PRE.SUMMER +
    T.SUMMER, data = dats, probability = T, importance = "impurity",
    keep.inbag = T)
frf
```

```
Ranger result

Call:
 ranger(as.factor(Aphelocephala.leucopsis) ~ TDIFF + PRE_SUMMER +      T_SUMMER, data = dat
keep.inbag = T)

Type:                             Probability estimation
Number of trees:                  500
Sample size:                      2998
Number of independent variables:  3
Mtry:                             1
Target node size:                 10
Variable importance mode:         impurity
Splitrule:                        gini
OOB prediction error (Brier s.):  0.03658797
```

```
roc.area(obs = dats$Aphelocephala.leucopsis, pred = predict(frf,
    data = dats)$prediction[, 2])$A   # AUC
```

```
[1] 0.9997497
```

The calibration analysis is slightly more involved here, as the randomForest returns probabilities, but the calibration analysis requires link-scale values. Some probabilities are 0 or 1, however, forcing us to nudge them a bit.

```
fitted.rf.response <- predict(frf, data = dats)$prediction[,
    2]  # probability of presence
geom_abline(slope = 1, intercept = 0, col = "red")
```

```
mapping: intercept = ~intercept, slope = ~slope
geom_abline: na.rm = FALSE
stat_identity: na.rm = FALSE
position_identity
```

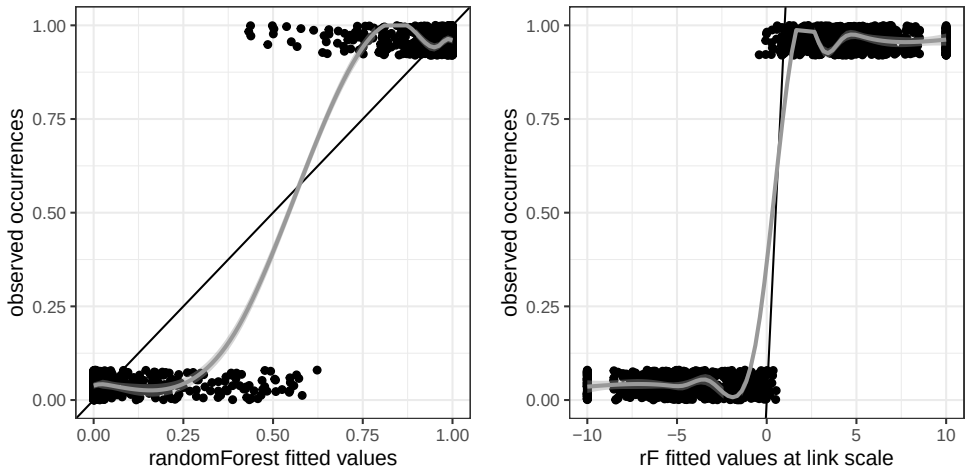
```
pdata <- cbind.data.frame(fits = fitted.rf.response, dats)
a <- ggplot(pdata, aes(fits, jitter(Aphelocephala.leucopsis,
    0.4))) + scale_x_continuous("randomForest fitted values") +
```

```
    scale_y_continuous("observed occurrences", limits = c(0,
        1)) + geom_point() + geom_abline(slope = 1, intercept = 0,
    col = "black") + geom_smooth(col = "grey60") + theme_bw()

fitted.rf.link <- qlogis(fitted.rf.response)
fitted.rf.link <- ifelse(fitted.rf.response == 0, -10, fitted.rf.link)  # set 0s to small va
fitted.rf.link <- ifelse(fitted.rf.response == 1, 10, fitted.rf.link)  # set 1s to large val
pdata2 <- cbind.data.frame(fits = fitted.rf.link , dats)
b <- ggplot(pdata2, aes(fits , jitter(Aphelocephala.leucopsis ,
    0.4))) + scale_x_continuous("rF fitted values at link scale") +
    scale_y_continuous("observed occurrences", limits = c(0,
        1)) + geom_point() + geom_abline(slope = 1, intercept = 0,
    col = "black") + geom_smooth(col = "grey60") + theme_bw()
a
```

```
`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

```
b
```

```
`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Compared to the GLM, the deviations are substantial.

Is this a consequence of using randomForest in classification mode, rather than as regression?

```
frf2 <- randomForest::randomForest(Aphelocephala.leucopsis ~
    TDIFF + PRE_SUMMER + T_SUMMER, data = dats)
frf2
```

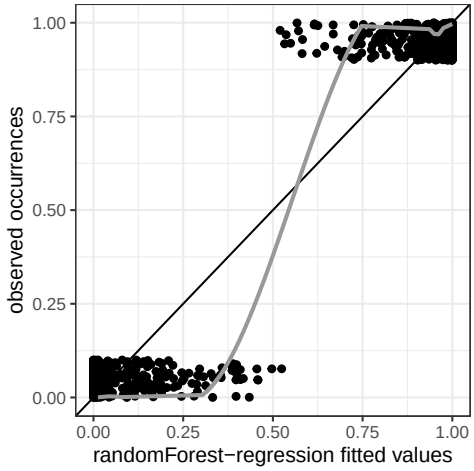
```

Call:
 randomForest(formula = Aphelocephala.leucopsis ~ TDIFF + PRE_SUMMER +      T_SUMMER, data
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 1

          Mean of squared residuals: 0.03554708
                    % Var explained: 85.77
```

```
roc.area(obs = dats$Aphelocephala.leucopsis, pred = predict(frf2,
    newdata = dats, type = "response"))$A  # AUC
```

```
[1] 0.9999955
```

```
pdata <- cbind.data.frame(fits = predict(frf2, newdata = dats,
    type = "response"), dats)
ggplot(pdata, aes(fits, Aphelocephala.leucopsis)) + scale_x_continuous("randomForest-regress
    scale_y_continuous("observed occurrences", limits = c(0,
        1)) + geom_jitter(width = 0, height = 0.1) + geom_abline(slope = 1,
    intercept = 0, col = "black") + geom_smooth(col = "grey60") +
    theme_bw()
```

```
`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

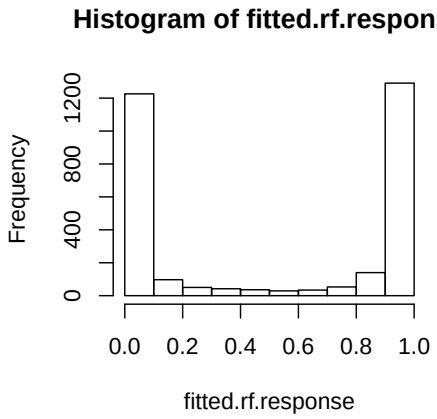


No: using randomForest in regression mode does not solve the issue of miscalibration. We will ignore this approach in the following.

We pick up where we left with the previous (regression) randomForest and carry out the actual calibration diagnostic using a GLM.

```
hist(fitted.rf.response)
cali.rf ← glm(dats$Aphelocephala.leucopsis ~ fitted.rf.link ,
    family = binomial)
summary(cali.rf)
```

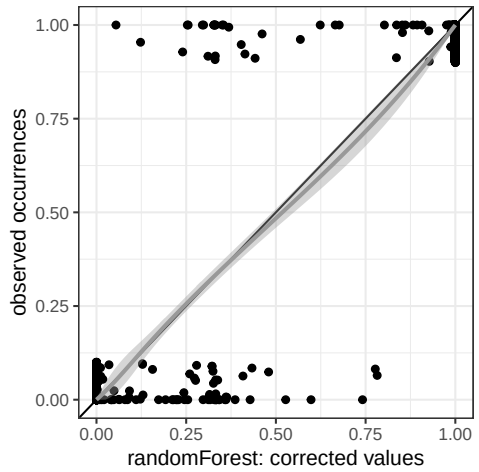
```

Call:
glm(formula = dats$Aphelocephala.leucopsis ~ fitted.rf.link,
    family = binomial)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.92322  -0.00001   0.00000   0.00011   2.46935

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)     -0.6969     0.2465  -2.827   0.0047 **
fitted.rf.link   4.4874     0.5443   8.245   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 4154.2  on 2997  degrees of freedom
Residual deviance:  119.6  on 2996  degrees of freedom
AIC: 123.6

Number of Fisher Scoring iterations: 12
```



In line with the above spline plot, the slope is much too steep and the intercept a bit off (negative value). While in this case it would be possible to use the inverse of the calibration regression to compute corrected predictions, we shall use the more generally suitable GAM:

```
cal.gam.rf ← gam(dats$Aphelocephala.leucopsis ~ s(fitted.rf.response,
    bs = "ts", m = 1), family = binomial)
corr.fitted.rf.response ← predict(cal.gam.rf, type = "response")
pdata.corr ← cbind.data.frame(corr = corr.fitted.rf.response,
    dats)
ggplot(pdata.corr, aes(corr, Aphelocephala.leucopsis)) + scale_x_continuous("randomForest: c
    scale_y_continuous("observed occurrences", limits = c(0,
        1), oob = scales::squish) + geom_jitter(width = 0, height = 0.1) +
    geom_abline(slope = 1, intercept = 0, col = "black") + geom_smooth(col = "grey60") +
    theme_bw()
```

```
`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



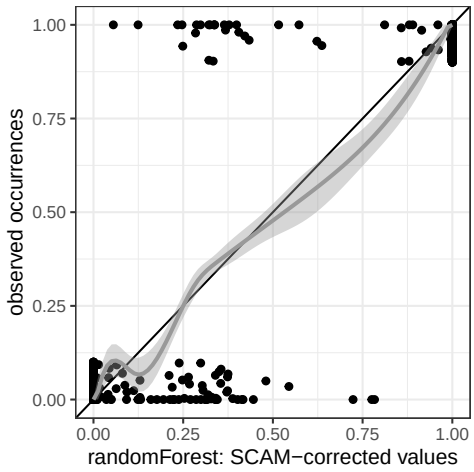
As we can see, the GAM-correction brings the fitted values close to the 1:1 line. Notice that the scatter of the observed data has increased, now spacing them out more widely along the x-axis. That is, in fact, what we want to see: that for an observed probability of, say, 0.1, 10% of the data points are actually 1s. In the previous graph, none of the data points was a 1, although the predictions were as high as 0.3! (Note also that the majority of data points sit on 0 and 1 (predicted value), which is not visible here.)

To make this statement explicit again: In the original plot, randomForest would claim a prediction of, say, 0.8, while in fact the observed values indicate already a value of 1. Hence, we need to increase the values at high predicted values, and decrease it at low predicted values. This will make the functional relationship more harsh and steep (see below).

Let's now constrain the GAM to be increasing monotonous and compare it with a (slightly penalised) ordinary GAM:

```
library(scam)
library(ggplot2)
cal.scam.rf ← scam(dats$Aphelocephala.leucopsis ~ s(fitted.rf.response,
    bs = "mpi", m = 3), family = binomial)
corr.fitted.rf.response ← predict(cal.scam.rf, type = "response")
pdata.corr ← cbind.data.frame(corr = corr.fitted.rf.response,
    dats)
ggplot(pdata.corr, aes(corr, Aphelocephala.leucopsis)) + scale_x_continuous("randomForest: S
    scale_y_continuous("observed occurrences", limits = c(0,
        1), oob = scales::squish) + geom_jitter(height = 0.1,
    width = 0) + geom_abline(slope = 1, intercept = 0, col = "black") +
    geom_smooth(col = "grey60") + theme_bw()
```

```
`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



In this case, the spline-argument `m=3` serves to penalise the spline. The GAM is more effective than the SCAM to get the calibration onto the 1:1 line. Overall, these two approaches are typically very similar, but it is worth checking the resulting calibration.

```
corr.fitted.rf.link ← qlogis(corr.fitted.rf.response)
corr.fitted.rf.link ← ifelse(corr.fitted.rf.response == 0, -40,
    corr.fitted.rf.link)  # set 0s to small value
corr.fitted.rf.link ← ifelse(corr.fitted.rf.response == 1, 40,
    corr.fitted.rf.link)  # set 1s to large value
cali.corr.rf ← glm(dats$Aphelocephala.leucopsis ~ corr.fitted.rf.link,
    family = binomial)
summary(cali.corr.rf)
```

```

Call:
glm(formula = dats$Aphelocephala.leucopsis ~ corr.fitted.rf.link,
    family = binomial)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.78824  -0.00020   0.00001   0.00003   2.45996

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)        0.02751    0.26399   0.104    0.917
corr.fitted.rf.link  1.05215    0.21856   4.814 1.48e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 4154.18  on 2997  degrees of freedom
Residual deviance:  108.25  on 2996  degrees of freedom
AIC: 112.25

```

```
Number of Fisher Scoring iterations: 12
```

So again, the calibrated predictions are close to the 1:1 line, as the calibration statistics indicate.

1.3.1 *A one-hidden-layer Neural Network*

Now we do the same with an Artificial Neural Network. There are plenty of different network algorithms, we use a rather simple one. While it does not matter for randomForest, ANNs seem to benefit from standardising the predictors.

```
library(nnet)
dats$sTDIFF ← scale(dats$TDIFF)
dats$sPRE_SUMMER ← scale(dats$PRE_SUMMER)
dats$sT_SUMMER ← scale(dats$T_SUMMER)
set.seed(1)
fnn ← nnet(as.factor(Aphelocephala.leucopsis) ~ sTDIFF + sPRE_SUMMER +
    sT_SUMMER, data = dats, size = 7, decay = 0.5, maxi = 500)
```

```
# weights:  36
initial  value 2186.894119
iter  10 value 852.636337
iter  20 value 695.119873
iter  30 value 625.919147
iter  40 value 608.252167
iter  50 value 601.686018
iter  60 value 595.782728
iter  70 value 590.650391
iter  80 value 589.349713
iter  90 value 588.879863
iter 100 value 588.823353
final  value 588.815705
converged
```

```
fnn
```

```
a 3-7-1 network with 36 weights
inputs: sTDIFF sPRE_SUMMER sT_SUMMER
output(s): as.factor(Aphelocephala.leucopsis)
options were - entropy fitting  decay=0.5
```

```
roc.area(obs = dats$Aphelocephala.leucopsis, pred = predict(fnn))$A  # AUC
```

```
[1] 0.983071
```

```
library(nnet)
library(ggplot2)
fitted.nn.response ← as.vector(predict(fnn, newdata = dats))
pdata ← cbind.data.frame(fits = fitted.nn.response, dats)
ggplot(pdata, aes(fits, Aphelocephala.leucopsis)) + scale_x_continuous("Neural Network: fitte
    scale_y_continuous("observed occurrences", limits = c(0,
        1), oob = scales::squish) + geom_jitter(width = 0, height = 0.1) +
    geom_abline(slope = 1, intercept = 0, col = "black") + geom_smooth(col = "grey60") +
    theme_bw()
```

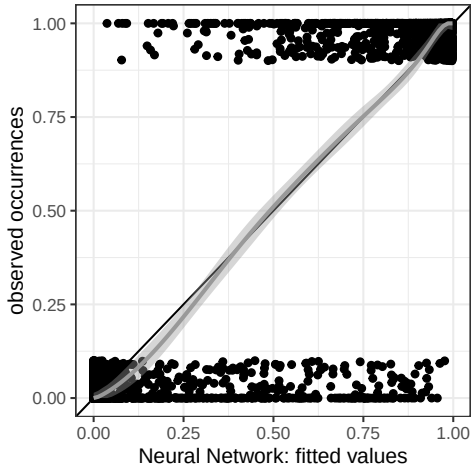
```
`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

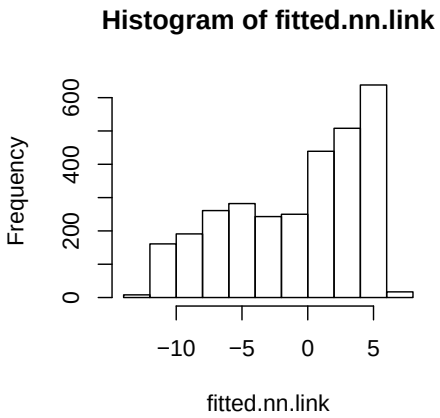
```
fitted.nn.link ← qlogis(fitted.nn.response)
range(fitted.nn.link)
```

```
[1] -12.235373   6.141623
```

```
# Since some values are Inf, we set those approx. to the
# largest value observed in fitted.nn.link:
hist(fitted.nn.link)
fitted.nn.link ← ifelse(fitted.nn.response == 0, -1e+15, fitted.nn.link)  # set 0s to large
fitted.nn.link ← ifelse(fitted.nn.response == 1, 5e+15, fitted.nn.link)  # set 1s to large
cali.nn ← glm(dats$Aphelocephala.leucopsis ~ fitted.nn.link,
    family = binomial)
summary(cali.nn)
```

```
Call:
glm(formula = dats$Aphelocephala.leucopsis ~ fitted.nn.link,
    family = binomial)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.96769  -0.05325   0.04150   0.18008   2.84620

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)    -0.09747    0.08615  -1.131    0.258
fitted.nn.link  1.20986    0.05604  21.590   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 4154.18  on 2997  degrees of freedom
Residual deviance:  978.19  on 2996  degrees of freedom
AIC: 982.19

Number of Fisher Scoring iterations: 8
```

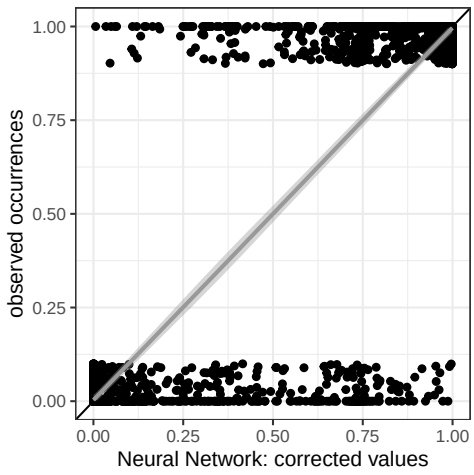




This fit is well calibrated. Note that this depends on the choise of the decay rate (smaller values cause more deviation).

```
cal.scam.nn ← scam(dats$Aphelocephala.leucopsis ~ s(fitted.nn.response,
    bs = "mpi"), family = binomial)
corr.fitted.nn.response ← predict(cal.scam.nn, type = "response")
pdata.corr ← cbind.data.frame(corr = corr.fitted.nn.response,
    dats)
ggplot(pdata.corr, aes(corr, Aphelocephala.leucopsis)) + scale_x_continuous("Neural Network:
    scale_y_continuous("observed occurrences", limits = c(0,
        1), oob = scales::squish) + geom_jitter(width = 0, height = 0.1) +
    geom_abline(slope = 1, intercept = 0, col = "black") + geom_smooth(col = "grey60") +
    theme_bw()
```

```
`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



1.4 *SVM*

Now we do the same with a support vector machine:

```
library(e1071)
fsvm ← svm(as.factor(Aphelocephala.leucopsis) ~ TDIFF + PRE_SUMMER +
    T_SUMMER, data = dats, type = "C-classification", probability = T,
    cost = 1)
fsvm
```

```
Call:
svm(formula = as.factor(Aphelocephala.leucopsis) ~ TDIFF + PRE_SUMMER +
    T_SUMMER, data = dats, type = "C-classification", probability = T,
    cost = 1)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  596
```

```
roc.area(obs = dats$Aphelocephala.leucopsis, pred = attr(predict(fsvm,
    newdata = dats, probability = T), "probabilities")[, 2])$A  # AUC
```

```
[1] 0.9835337
```

```
fitted.svm.response ← attr(predict(fsvm, newdata = dats, probability = T),
    "probabilities")[, 2]  # probability of presence
fitted.svm.link ← qlogis(fitted.svm.response)
range(fitted.svm.link)
```

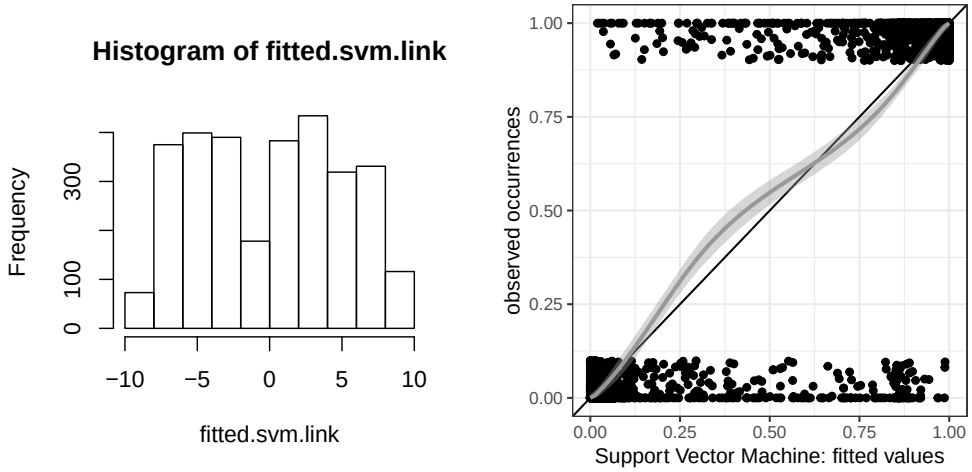
```
[1] -9.876540  8.747935
```

```
hist(fitted.svm.link)
pdata ← cbind.data.frame(fits = fitted.svm.response, dats)
ggplot(pdata, aes(fits, Aphelocephala.leucopsis)) + scale_x_continuous("Support Vector Machi
    scale_y_continuous("observed occurrences", limits = c(0,
        1), oob = scales::squish) + geom_jitter(width = 0, height = 0.1) +
    geom_abline(slope = 1, intercept = 0, col = "black") + geom_smooth(col = "grey60") +
    theme_bw()
```

```
`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

```
cali.svm ← glm(dats$Aphelocephala.leucopsis ~ fitted.svm.link,
    family = binomial)
summary(cali.svm)
```

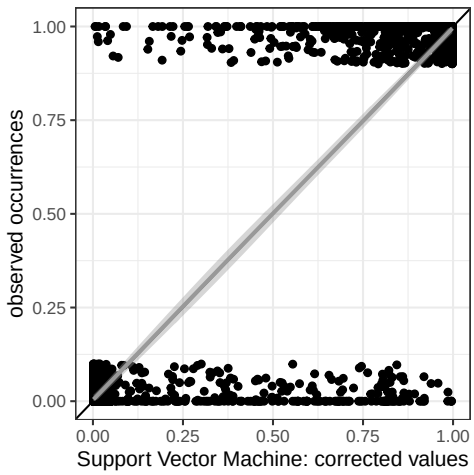
```
Call:
glm(formula = dats$Aphelocephala.leucopsis ~ fitted.svm.link,
    family = binomial)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-3.01221  -0.12534   0.01896   0.19535   2.84801

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)     -0.04483    0.08617   -0.52    0.603
fitted.svm.link  1.02673    0.04255   24.13   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 4154.2  on 2997  degrees of freedom
Residual deviance:  969.1  on 2996  degrees of freedom
AIC: 973.1

Number of Fisher Scoring iterations: 7
```



Clearly, this line follows the 1:1 calibration rather closely. This is setting-dependent. If we increase the 'cost'-parameter, the curve will deviate more and more; while a (default) value of 1 or less will yield a near-perfect match with the 1:1 line.

```
cal.scam.svm ← scam(dats$Aphelocephala.leucopsis ~ s(fitted.svm.response,
    bs = "mpi"), family = binomial)
corr.fitted.svm.response ← predict(cal.scam.svm, type = "response")
pdata.corr ← cbind.data.frame(corr = corr.fitted.svm.response,
    dats)
ggplot(pdata.corr, aes(corr, Aphelocephala.leucopsis)) + scale_x_continuous("Support Vector
    scale_y_continuous("observed occurrences", limits = c(0,
        1), oob = scales::squish) + geom_jitter(width = 0, height = 0.1) +
    geom_abline(slope = 1, intercept = 0, col = "black") + geom_smooth(col = "grey60") +
    theme_bw()
```

```
`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



1.5 BRT

```
library(gbm)
fbrt ← gbm(Aphelocephala.leucopsis ~ TDIFF + PRE.SUMMER + T.SUMMER,
    data = dats, distribution = "bernoulli", n.trees = 5000,
    interaction.depth = 3, shrinkage = 0.01, cv.fold = 5)
roc.area(obs = dats$Aphelocephala.leucopsis, pred = plogis(predict(fbrt,
    newdata = dats)))$A  # AUC
```

```
Using 4924 trees...
```

```
[1] 0.9985331
```

```
library(gbm)
fitted.brt.link ← predict(fbrt, newdata = dats)
```

```
Using 4924 trees...
```

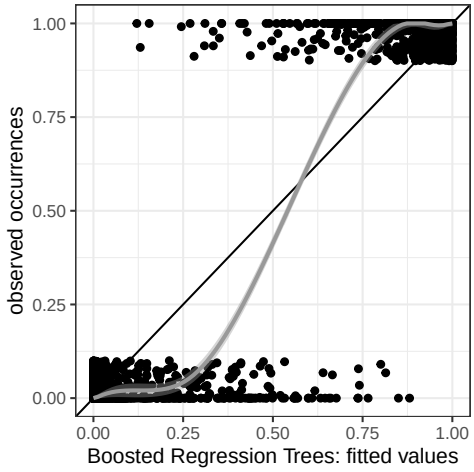
```
fitted.brt.response ← plogis(fitted.brt.link)

pdata ← cbind.data.frame(fits = fitted.brt.response, dats)
ggplot(pdata, aes(fits, Aphelocephala.leucopsis)) + scale_x_continuous("Boosted Regression T
    scale_y_continuous("observed occurrences", limits = c(0,
        1), oob = scales::squish) + geom_jitter(width = 0, height = 0.1) +
    geom_abline(slope = 1, intercept = 0, col = "black") + geom_smooth(col = "grey60") +
    theme_bw()
```

```
`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

```
cali.brt ← glm(dats$Aphelocephala.leucopsis ~ fitted.brt.link,
    family = binomial)
summary(cali.brt)
```

```

Call:
glm(formula = dats$Aphelocephala.leucopsis ~ fitted.brt.link,
    family = binomial)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.91350  -0.00156   0.00042   0.00648   3.10433

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)      -0.3097     0.1550  -1.997   0.0458 *
fitted.brt.link   2.2688     0.1731  13.107   <2e-16 ***
---
```

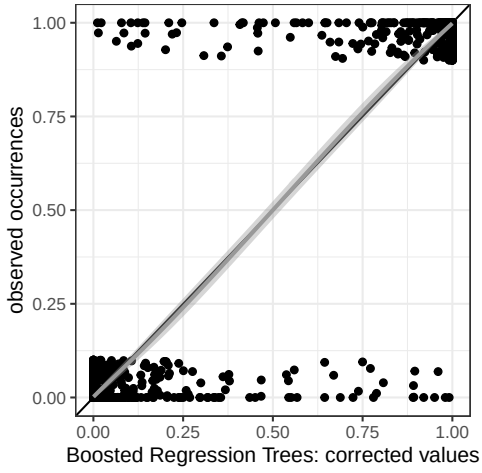
```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 4154.18  on 2997  degrees of freedom
Residual deviance:  293.21  on 2996  degrees of freedom
AIC: 297.21

Number of Fisher Scoring iterations: 10
```



Again, this line does not follow the 1:1 calibration, and the calibration regression is also indicating misfit.

```
cal.scam.brt ← scam(dats$Aphelocephala.leucopsis ~ s(fitted.brt.response,
    bs = "mpi"), family = binomial)
corr.fitted.brt.response ← predict(cal.scam.brt, type = "response")
pdata.corr ← cbind.data.frame(corr = corr.fitted.brt.response,
    dats)
ggplot(pdata.corr, aes(corr, Aphelocephala.leucopsis)) + scale_x_continuous("Boosted Regressi
    scale_y_continuous("observed occurrences", limits = c(0,
        1), oob = scales::squish) + geom_jitter(width = 0, height = 0.1) +
    geom_abline(slope = 1, intercept = 0, col = "black") + geom_smooth(col = "grey60") +
    theme_bw()
```

```
`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Nice!

2 *Cross-validation of corrected predictors*

We shall look at two cross-validations: first, 5-fold with unbalanced prevalence.

we can validate on geographic extrapolation, i.e. fit it to the left half of Australia and predict it to the right, and vice versa.

```
# block cross-validation: left vs right
k = 2
AUCmat ← LLmat ← RMSEmat ← matrix(ncol = k, nrow = 5)
rownames(AUCmat) ← rownames(LLmat) ← rownames(RMSEmat) ← c("GLM",
    "RF", "ANN", "SVM", "BRT")
AUCmatCal ← LLmatCal ← RMSEmatCal ← matrix(ncol = k, nrow = 5)
rownames(AUCmatCal) ← rownames(LLmatCal) ← rownames(RMSEmatCal) ← rownames(AUCmat)

ell ← function(obs, pred) {
    # compute likelihood
    predsCalTrunc ← ifelse(pred ≤ 0, 1e-08, ifelse(pred ≥
        1, 1 - 1e-08, pred))
    sum(dbinom(obs, prob = predsCalTrunc, size = 1, log = T))
}

for (i in 1:2) {
    # repeated resampling loop fit on left, predict to right half
    # of Australia
    if (i == 1)
        fold.nr ← ifelse(dats$LONG < 0, 1, 2)
    if (i == 2)
        fold.nr ← ifelse(dats$LONG < 0, 2, 1)
    # glm
    fglmcv ← step(glm(Aphelocephala.leucopsis ~ (TDIFF + PRE.SUMMER +
        T.SUMMER)^2 + I(TDIFF^2) + I(PRE.SUMMER^2) + I(T.SUMMER^2),
        family = binomial, data = dats[fold.nr != 1, ]), trace = F)
    predsGLM ← predict(fglmcv, type = "response", newdata = dats[fold.nr ==
        1, ])
    AUCmat[1, i] ← roc.area(obs = dats$Aphelocephala.leucopsis[fold.nr ==
        1], pred = predsGLM)$A
    LLmat[1, i] ← ell(dats$Aphelocephala.leucopsis[fold.nr ==
        1], predsGLM)
    RMSEmat[1, i] ← sqrt(mean((dats$Aphelocephala.leucopsis[fold.nr ==
        1] - predsGLM)^2))
    # glm calibrated:
    fitted.glm ← predict(fglmcv, type = "response", newdata = dats[fold.nr !=
        1, ])
    cal.scam.glmcv ← scam(dats$Aphelocephala.leucopsis[fold.nr !=
        1] ~ s(fitted.glm, bs = "mpi"), family = binomial)
    predsGLMcal ← predict(cal.scam.glmcv, newdata = data.frame(fitted.glm = predsGLM),
        type = "response")
    AUCmatCal[1, i] ← roc.area(obs = dats$Aphelocephala.leucopsis[fold.nr ==
        1], pred = predsGLMcal)$A
    LLmatCal[1, i] ← ell(obs = dats$Aphelocephala.leucopsis[fold.nr ==
        1], predsGLMcal)
    RMSEmatCal[1, i] ← sqrt(mean((dats$Aphelocephala.leucopsis[fold.nr ==
        1] - predsGLMcal)^2))

    # ranger
    frfcv ← ranger(as.factor(Aphelocephala.leucopsis) ~ TDIFF +
        PRE.SUMMER + T.SUMMER, data = dats[fold.nr != 1, ], probability = T)
    predsrf ← predict(frfcv, data = dats[fold.nr == 1, ])$prediction[,
        2]
    AUCmat[2, i] ← roc.area(obs = dats$Aphelocephala.leucopsis[fold.nr ==
        1], pred = predsrf)$A
    LLmat[2, i] ← ell(dats$Aphelocephala.leucopsis[fold.nr ==
        1], predsrf)
    RMSEmat[2, i] ← sqrt(mean((dats$Aphelocephala.leucopsis[fold.nr ==
        1] - predsrf)^2))
    # rf calibrated:
    fitted.rf ← predict(frfcv, data = dats[fold.nr != 1, ])$prediction[,
        2]
    cal.scam.rfcv ← scam(dats$Aphelocephala.leucopsis[fold.nr !=
        1] ~ s(fitted.rf, bs = "mpi"), family = binomial)
    predsRFcal ← predict(cal.scam.rfcv, newdata = data.frame(fitted.rf = predsrf),
        type = "response")
```

```
AUCmatCal[2, i] <- roc.area(obs = dats$Aphelocephala.leucopsis[fold.nr ==
    1], pred = predsRFcal)$A
LLmatCal[2, i] <- ell(dats$Aphelocephala.leucopsis[fold.nr ==
    1], predsRFcal)
RMSEmatCal[2, i] <- sqrt(mean((dats$Aphelocephala.leucopsis[fold.nr ==
    1] - predsRFcal)^2))

# ANN
invisible(capture.output(fanncv <- nnet(as.factor(Aphelocephala.leucopsis) ~
    sTDIFF + sPRE_SUMMER + sT_SUMMER, data = dats[fold.nr !=
    1, ], size = 7, decay = 0.5, maxi = 500)))
invisible(capture.output(predsann <- plogis(predict(fanncv,
    newdata = dats[fold.nr == 1, ])))) # to suppress text
AUCmat[3, i] <- roc.area(obs = dats$Aphelocephala.leucopsis[fold.nr ==
    1], pred = predsann)$A
LLmat[3, i] <- ell(dats$Aphelocephala.leucopsis[fold.nr ==
    1], predsann)
RMSEmat[3, i] <- sqrt(mean((dats$Aphelocephala.leucopsis[fold.nr ==
    1] - predsann)^2))
# ANN calibrated:
fitted.ann <- plogis(predict(fanncv, newdata = dats[fold.nr !=
    1, ]))
cal.scam.anncv <- scam(dats$Aphelocephala.leucopsis[fold.nr !=
    1] ~ s(fitted.ann, bs = "mpi"), family = binomial)
predsanncal <- predict(cal.scam.anncv, newdata = data.frame(fitted.ann = predsann),
    type = "response")
AUCmatCal[3, i] <- roc.area(obs = dats$Aphelocephala.leucopsis[fold.nr ==
    1], pred = predsanncal)$A
LLmatCal[3, i] <- ell(dats$Aphelocephala.leucopsis[fold.nr ==
    1], predsanncal)
RMSEmatCal[3, i] <- sqrt(mean((dats$Aphelocephala.leucopsis[fold.nr ==
    1] - predsanncal)^2))

# SVM
fsvmcv <- svm(as.factor(Aphelocephala.leucopsis) ~ TDIFF +
    PRE_SUMMER + T_SUMMER, data = dats[fold.nr != 1, ], type = "C-classification",
    probability = T, cost = 1)
predssvm <- attr(predict(fsvmcv, newdata = dats[fold.nr ==
    1, ], probability = T), "probabilities")[, 2]
AUCmat[4, i] <- roc.area(obs = dats$Aphelocephala.leucopsis[fold.nr ==
    1], pred = predssvm)$A
LLmat[4, i] <- ell(dats$Aphelocephala.leucopsis[fold.nr ==
    1], predssvm)
RMSEmat[4, i] <- sqrt(mean((dats$Aphelocephala.leucopsis[fold.nr ==
    1] - predssvm)^2))
# SVM calibrated:
fitted.svm <- attr(predict(fsvmcv, newdata = dats[fold.nr !=
    1, ], probability = T), "probabilities")[, 2]
cal.scam.svmcv <- scam(dats$Aphelocephala.leucopsis[fold.nr !=
    1] ~ s(fitted.svm, bs = "mpi"), family = binomial)
predssvmcal <- predict(cal.scam.svmcv, newdata = data.frame(fitted.svm = predssvm),
    type = "response")
AUCmatCal[4, i] <- roc.area(obs = dats$Aphelocephala.leucopsis[fold.nr ==
    1], pred = predssvmcal)$A
LLmatCal[4, i] <- ell(dats$Aphelocephala.leucopsis[fold.nr ==
    1], predssvmcal)
RMSEmatCal[4, i] <- sqrt(mean((dats$Aphelocephala.leucopsis[fold.nr ==
    1] - predssvmcal)^2))


# BRT
fbrtcv <- gbm(Aphelocephala.leucopsis ~ TDIFF + PRE_SUMMER +
    T_SUMMER, data = dats[fold.nr != 1, ], distribution = "bernoulli",
    n.trees = 5000, interaction.depth = 3, shrinkage = 0.01,
    cv.fold = 5)
invisible(capture.output(predsbrt <- plogis(predict(fbrtcv,
```

```
        newdata = dats[fold.nr == 1, ])))  # to suppress text
    AUCmat[5, i] <- roc.area(obs = dats$Aphelocephala.leucopsis[fold.nr ==
        1], pred = predsbrt)$A
    LLmat[5, i] <- ell(dats$Aphelocephala.leucopsis[fold.nr ==
        1], predsbrt)
    RMSEmat[5, i] <- sqrt(mean((dats$Aphelocephala.leucopsis[fold.nr ==
        1] - predsbrt)^2))
    # BRT calibrated:
    invisible(capture.output(fitted.brt <- plogis(predict(fbrtcv,
        newdata = dats[fold.nr != 1, ]))))
    cal.scam.brtcv <- scam(dats$Aphelocephala.leucopsis[fold.nr !=
        1] ~ s(fitted.brt, bs = "mpi"), family = binomial)
    predsBRTcal <- predict(cal.scam.brtcv, newdata = data.frame(fitted.brt = predsbrt),
        type = "response")
    AUCmatCal[5, i] <- roc.area(obs = dats$Aphelocephala.leucopsis[fold.nr ==
        1], pred = predsBRTcal)$A
    LLmatCal[5, i] <- ell(dats$Aphelocephala.leucopsis[fold.nr ==
        1], predsBRTcal)
    RMSEmatCal[5, i] <- sqrt(mean((dats$Aphelocephala.leucopsis[fold.nr ==
        1] - predsBRTcal)^2))

    # clean up:
    rm(fglmcv, predsGLM, fitted.glm, cal.scam.glmcv, predsGLMcal)
    rm(frfcv, predsrf, fitted.rf, cal.scam.rfcv, predsRFcal)
    rm(fanncv, predsann, fitted.ann, cal.scam.anncv, predsanncal)
    rm(fsvmcv, predssvm, fitted.svm, cal.scam.svmcv, predssvmcal)
    rm(fbrtcv, predsbrt, fitted.brt, cal.scam.brtcv, predsBRTcal)

}
```

```
signif(cbind(AUCraw = rowMeans(AUCmat, na.rm = T), AUCcal = rowMeans(AUCmatCal),
    llraw = rowSums(LLmat), llcal = rowSums(LLmatCal), RMSEraw = rowMeans(RMSEmat),
    RMSEcal = rowMeans(RMSEmatCal)), 3)
```

```
    AUCraw AUCcal llraw llcal RMSEraw RMSEcal
GLM  0.808  0.808 -5320 -2660   0.404   0.408
RF   0.929  0.929 -1250 -2850   0.325   0.363
ANN  0.957  0.949 -1690 -3340   0.435   0.306
SVM  0.901  0.901 -1680 -1940   0.333   0.336
BRT  0.895  0.904 -2000 -8390   0.352   0.367
```

3 *Why do some approaches yield uncalibrated fits?*

The stepwise-reduced GLM is `glm(formula = Aphelocephala.leucopsis ~ PRE_SUMMER + T_SUMMER + I(TDIFF^2) + I(PRE_SUMMER^2) + I(T_SUMMER^2) + PRE_SUMMER:T_SUMMER, family = binomial, data = dats)`. We can fit the same model, by hand if you like like so:

```
X <- model.matrix(~PRE_SUMMER + T_SUMMER + I(TDIFF^2) + I(PRE_SUMMER^2) +
    I(T_SUMMER^2) + PRE_SUMMER:T_SUMMER, data = as.data.frame(scale(dats)))
optfun <- function(parms) {
    p <- plogis(X %*% parms)
    -sum(dbinom(dats$Aphelocephala.leucopsis, prob = p, size = 1,
        log = T))
}
optfun(parms = rep(0.01, 7))
```

```
[1] 2105.853
```

```
(par <- optim(fn = optfun, par = rep(0.01, 7), control = list(maxit = 5000),
    method = "BFGS"))
```

```
$par
[1]  1.7493402 -4.3262291 -4.1001938 -0.3164058 -2.3372164 -1.4779296
[7] -3.2454675
```

```
$value
[1] 687.6579

$counts
function gradient
      84       25

$convergence
[1] 0

$message
NULL
```

Note that here the predictors were scaled before analysis (to facilitate optimisation).

Now we could use an alternative goal for our optimisation, one that focusses on getting the 0s and 1s right, rather than the likelihood. To do so, we threshold the predicted probabilities into 0s and 1s, and then optimise the proportion of correctly predicted 0s and 1s ("accuracy", or sum of proportion of true positives and true negatives):

Difficult to optimise, as the target function is very coarse through the integerisation:

```
set.seed(1)
optfun2 <- function(parms) {
    p <- plogis(X %*% parms)
    T <- 0.5126  #prevalence
    pred <- ifelse(p < T, 0, 1)
    CM <- table(pred, dats$Aphelocephala.leucopsis)
    -sum(diag(CM))/sum(CM)  # accuracy
}
optfun2(parms = runif(7))
```

```
[1] -0.4873249
```

```
(par2 <- optimx::optimx(fn = optfun2, par = runif(7), control = list(maxit = 5000,
    trace = 0), method = "hjkb"))  # good!
```

```
           p1        p2        p3        p4        p5        p6        p7
hjkb 3.629548 -6.370886 -5.875714 -1.794025 -5.323443 -1.437977 -4.490896
          value fevals gevals niter convcode kkt1  kkt2 xtime
hjkb -0.9269513    538     NA    19        0 TRUE FALSE  1.55
```

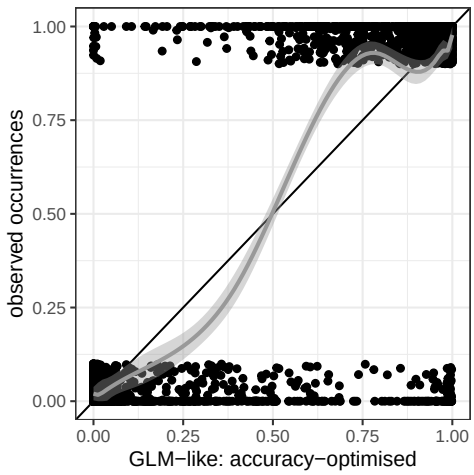
```
optfun2(parms = par$par)
```

```
[1] -0.9152769
```

With these optimised parameters we can generate model fits:

```
fitted.glm.acc <- plogis(X %*% unlist(par2[, 1:7]))
pdata <- cbind.data.frame(fits = fitted.glm.acc, dats)
ggplot(pdata, aes(fits, Aphelocephala.leucopsis)) + scale_x_continuous("GLM-like: accuracy-o
    scale_y_continuous("observed occurrences", limits = c(0,
        1), oob = scales::squish) + geom_jitter(width = 0, height = 0.1) +
    geom_abline(slope = 1, intercept = 0, col = "black") + geom_smooth(col = "grey60") +
    theme_bw()
```

```
`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

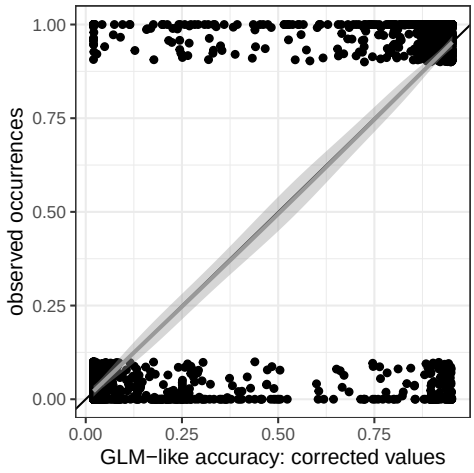


As we can see, using a different loss function will not only yield different parameter estimates, but also distortion from the 1:1 calibration line.

Let's reign this in with a `scam`-calibration:

```
cal.scam.glmacc ← scam(dats$Aphelocephala.leucopsis ~ s(fitted.glm.acc,
    bs = "mpi", m = 2), family = binomial)
corr.fitted.glmacc.response ← predict(cal.scam.glmacc, type = "response")
pdata.corr ← cbind.data.frame(corr = corr.fitted.glmacc.response,
    dats)
ggplot(pdata.corr, aes(corr, Aphelocephala.leucopsis)) + scale_x_continuous("GLM−like accurac
    scale_y_continuous("observed occurrences", limits = c(0,
        1), oob = scales::squish) + geom_jitter(width = 0, height = 0.1) +
    geom_abline(slope = 1, intercept = 0, col = "black") + geom_smooth(col = "grey60") +
    theme_bw()
```

```
`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

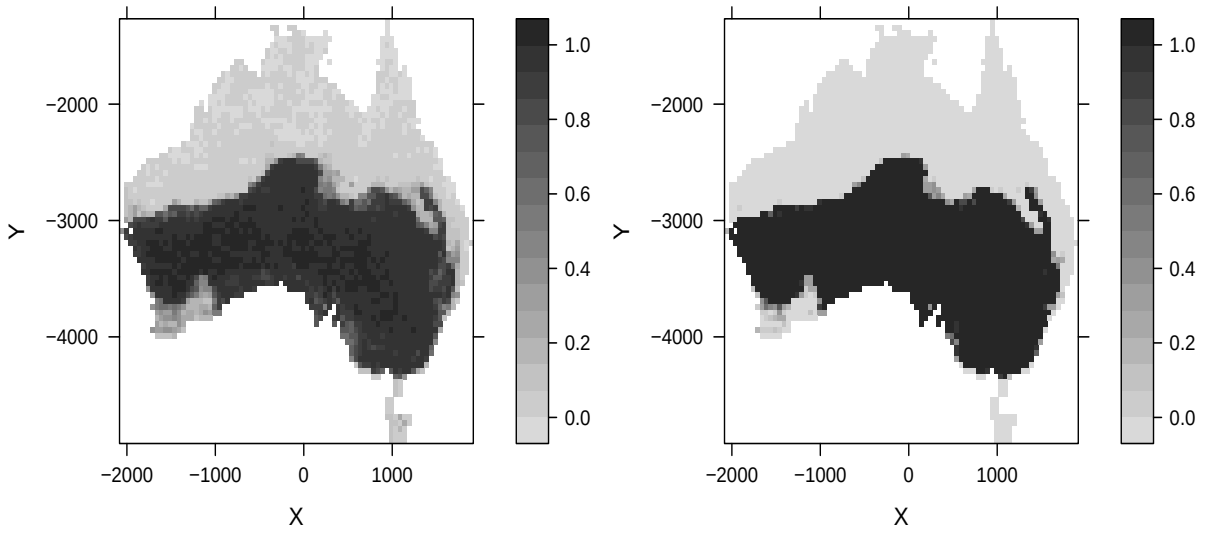


Nice!

4 Maps and effects

4.1 Maps

```
library(lattice)
dats$X ← dats$LONG/1000   # in km
dats$Y ← dats$LAT/1000   # in km
p1 ← levelplot(fitted.rf.response ~ X + Y, data = dats, col.regions = rev(gray(15:85/100)))
p2 ← levelplot(corr.fitted.rf.response ~ X + Y, data = dats,
    col.regions = rev(gray(15:85/100)))
print(p1, split = c(1, 1, 2, 1), more = T)
print(p2, split = c(2, 1, 2, 1))
```



Note that the randomForest predictions become much more crisp through the correction. The calibration plot in section 1.3 shows that the actual frequency in the observations is much higher than randomForest predictions above 0.5. Hence, when randomForest says "0.75' it actually means that the corrected probability are close to 1.

In this case, unless you happen to be interested in south western Australia, this difference may not bother you. The total occurrence of our target bird is typically very similar (indicating that both model predictions are in balance with the data):

```
mean(dats$Aphelocephala.leucopsis)  # prevalence in the data
```

```
[1] 0.5126751
```

```
mean(fitted.rf.response)  # randomForest-prelance
```

```
[1] 0.5128449
```

```
mean(corr.fitted.rf.response)  # corrected prevalence
```

```
[1] 0.5126751
```

Logically, if values become more extreme, and prevalence is the same, some pixels must move from grey to white or black, to keep the balance. In the map, this results in sharper boundaries between presence and absence.

4.2 *Effects*

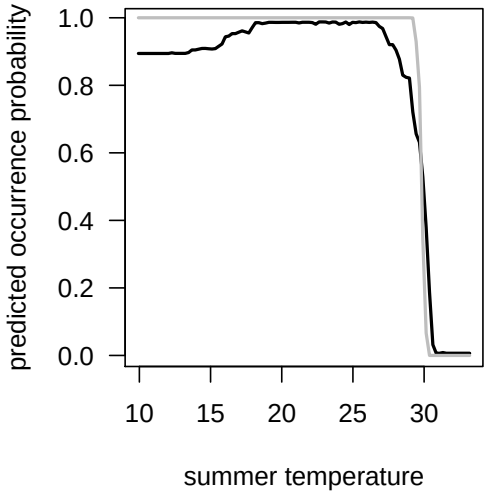
```
importance(frf)  # T_SUMMER most important
```

```
     TDIFF PRE_SUMMER   T_SUMMER
  303.4624   522.7069   591.0275
```

Conditional plot of T_SUMMER:

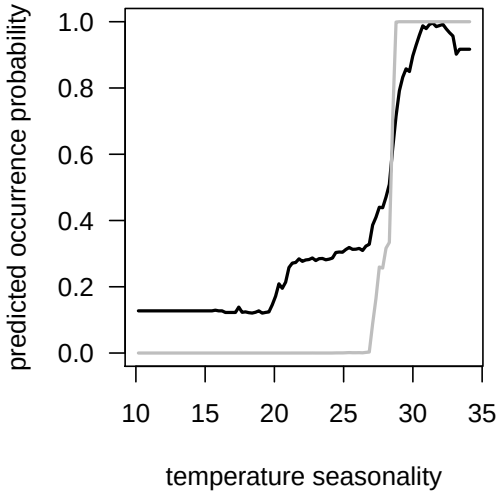
```
cal.gam.rf ← gam(dats$Aphelocephala.leucopsis ~ s(fitted.rf.response,
    bs = "ts", m = 1), family = binomial)  # just a recap from earlier

newT ← seq(min(dats$T_SUMMER), max(dats$T_SUMMER), len = 100)
newTdf ← data.frame(T_SUMMER = newT, TDIFF = median(dats$TDIFF),
    PRE_SUMMER = median(dats$PRE_SUMMER))
preds ← predict(frf, data = newTdf, type = "response")  # 'se' not implemented for classifi
preds.cor ← predict(cal.gam.rf, newdata = data.frame(fitted.rf.response = preds$predictions
    2]), type = "response")
```

```
par(mar = c(5, 5, 1, 1))
plot(newT, preds$predictions[, 2], type = "l", las = 1, lwd = 2,
    xlab = "summer temperature", ylab = "predicted occurrence probability")
lines(newT, preds.cor, lwd = 2, col = "grey")
```



Conditional plot of TDIFF:

```
newTDIFF ← seq(min(dats$TDIFF), max(dats$TDIFF), len = 100)
newTDIFFdf ← data.frame(TDIFF = newTDIFF, T.SUMMER = median(dats$T.SUMMER),
    PRE.SUMMER = median(dats$PRE.SUMMER))
preds ← predict(frf, data = newTDIFFdf, type = "response")  # 'se' not implemented for clas
preds.cor ← predict(cal.gam.rf, newdata = data.frame(fitted.rf.response = preds$predictions
    2]), type = "response")

par(mar = c(5, 5, 1, 1))
plot(newTDIFF, preds$predictions[, 2], type = "l", las = 1, lwd = 2,
    xlab = "temperature seasonality", ylab = "predicted occurrence probability",
    ylim = c(0, 1))
lines(newTDIFF, preds.cor, lwd = 2, col = "grey")
```



These figures illustrate that randomForest, and BRT alike, are drawing very sharp boundaries between where a species does or does not occur. Ecologically, that may be hard to defend. The correction amplifies this, but it is the model itself, not the correction, that fits a threshold. The correction "only" recalibrates it to probabilities. Not calibrating may look nicer, but the model predictions are then biased.