

**Supplementary material**

**2aaV\_UZ "**

# incR: a new R package to analyse incubation behaviour

## Appendix 1

*Pablo Capilla-Lasheras (p.capilla@exeter.ac.uk / pacapilla@gmail.com)*

*2018-04-24*

## Introduction

The use of the main function in `incR`, `incRscan`, requires to set argument values that can affect `incRscan` performance. The combination of video-recordings and nest temperatures can provide a powerful way of (i) assessing the general performance of `incRscan` on a specific data set and (ii) simulating how robust the results are to varying `incRscan` parameters, enabling the user to choose the optimal settings.

Here, using incubation temperatures from blue tits (*Cyanistes caeruleus*) and great tits (*Parus major*) recorded by iButtons (Maxim Technologies Inc.), I apply the approach described above. I combine nest temperatures and video-recordings to evaluate the position of the incubating individual (in or off the nest - incubation scores), and find the optimal parameters for `incRscan` for each data set.

## The data

See the main manuscript for details about data collection. I analyse twelve (total or partial) days of incubation coming from eight different nest-boxes (two blue tits and six great tit clutches) in two years of study, for which video-cameras were simultaneously employed along with iButtons devices that recorded nest temperature. The data were collected in spring 2015 and 2016 in Scotland and The Netherlands.

## Workflow

Following the recommendations in the main manuscript - that can also be found in an `incR` vignette:

## Data preparation `incRprep` and `incRenv`

I load data from ten nest-boxes as ten different files and then create a list in which every element is a data frame. Of course, if you wanted to reproduce the code using your own data, you will need to change the paths to files.

```
# install if not done before
## for developing version
devtools::install_github(repo = "PabloCapilla/incR")
## for CRAN stable version
install.packages("incR")

library(incR)
packageVersion("incR") # release package version 1.1.0 (March 2018)

# install and load these packages to run this vignette
library(ggplot2)
library(dplyr)
```

```

library(data.table)
library(extrafont)
loadfonts()

# environmental data for Scotland for 2015 and 2016
env_Data_Scotland_2015 <- read.csv("2-dataCalibration/envData/env_Data2015.csv")
env_Data_Scotland_2016 <- read.csv("2-dataCalibration/envData/2016/1510UT_FS_20160606.csv")
# environmental data for the Netherlands
env_Data_NIOO_2016 <- read.csv("2-dataCalibration/envData/weathersouth_NIOO.csv")

# a sample of these data
head(env_Data_NIOO_2016) # the other env files have the same structure and column names

# incubation data
incubation_files <- dir("./2-dataCalibration/data_R/", # files containing incubation data
                        full.names = TRUE,
                        pattern = ".csv")
rawdata_calibration <- lapply(X = as.list(incubation_files), # reads every file
                              # in incubation_files
                              FUN = function(X) read.csv(X) # produces a list
                              # of data frames)

# checking the data are correct
lapply(X = rawdata_calibration, FUN = head)

```

In order from left to right: date column, nest temperature, presence of the incubating bird (1 = YES, 0 = NO, or NA) based on video footage, nest-box code, site (either Scotland or the Netherlands).

Once a list of data frames has been created, it is easy and fast to use `lapply` to apply any function to every element of the list (as done above). To make things even faster, I use `parLapply` from the `doPARALLEL` package to do parallel computing in my PC - see below how this is set up for three computing threads. *Doing parallel computing in R might sound difficult but it is actually very straightforward. Have a look at this [link](#) to find out more*

```

library(doParallel)
ncores <- 3
cluster_incR <- makeCluster(ncores)
registerDoParallel(cluster_incR)
#getDoParWorkers() # check you actually have as many working threads as you want

clusterExport(cl = cluster_incR, c("env_Data_Scotland_2015",
                                   "env_Data_NIOO_2016",
                                   "env_Data_Scotland_2016",
                                   "rawdata_calibration"))
clusterEvalQ(cl = cluster_incR, library(incR))

```

Now, `parLapply` can be called.

```

# applying incRprep
calibration_prepdata <- parLapply(cl = cluster_incR,
                                  X = rawdata_calibration,
                                  fun = incRprep,
                                  date.name = "date00",
                                  temperature.name = "temp",
                                  date.format = "%d/%m/%Y %H:%M",
                                  timezone = "GMT")

```

You can check that `incRprep` has worked fine for every data frame (Note that the column `PRESENCE` gives incubation scores based on video footage). New columns `index`, `time`, `hour`, `minute`, `date`, `dec_time` and `temp1` should have been created. For information about these new variables check the documentation page for `incRprep`.

```
lapply(calibration_prepdata, head, n = 3)
```

The next step, and final data preparation step, is to calculate environmental temperatures for every line and data frame. This task is done by `incRenv` and I apply the function in a similar fashion as we did for `incRprep`.

The only difference now is that we need to deal with the fact that the source of environmental data changes for the different elements in `calibration_prepdata`: for the 1st and 2nd elements `env_Data_Scotland_2016` applies, `env_Data_Scotland_2015` for the 7th and 8th, and `env_Data_NIOO_2016` for 3rd to 6th one. I solve this issue in the code below, other coding choices are certainly possible.

```
clusterExport(cl = cluster_incR, c("calibration_prepdata"))
```

```
# the computation below might take between one and ten minutes
calibration_finaldata <- parLapply(cl = cluster_incR,
  X = as.list(c(1:length(calibration_prepdata))),
  fun = function(X){
    if(X <= 2){
      environmental_data <- env_Data_Scotland_2016
    } else {
      if (X >= 7){
        environmental_data <- env_Data_Scotland_2015
      } else {
        environmental_data <- env_Data_NIOO_2016
      }
    }
    output <- incRenv (data.env = environmental_data,
      data.nest =
        calibration_prepdata[[X]],
      env.temperature.name = "tempEnv",
      env.date.name = "dateEnv",
      env.date.format =
        "%d/%m/%Y %H:%M",
      env.timezone="GMT")
    return(output)
  }
)
#stopCluster(cluster_incR) # stop cluster
# checking the new column for environmental temperatures has been added:
lapply(calibration_finaldata, head, n = 3)

# range in differences between nest temperature and environmental temperature
temp_dif <- rbindlist(lapply(calibration_finaldata, function(x){
  df <- x %>% filter(PRESENCE == 1) # select only data used for calibration
  data.frame(box = unique(x$BOX),
    mean = mean(df$temp - df$env_temp),
    min = range(df$temp - df$env_temp)[1], # calculate minimum for every nest-box
    max = range(df$temp - df$env_temp)[2]) # calculate maximum for every nest-box
}))
)
```

```

# mean across nest-boxes
mean(temp_dif$mean)
# sd
sd(temp_dif$mean)

```

Now we have the data ready to use `incRscan` and assess its results over a number of different parameter values.

## Calibration of `incRscan`

For details about the parameters in `incRscan`, check the documentation (`?incR::incRscan`) and the main manuscript.

I run `incRscan` for varying values of `maxNightVariation`, `sensitivity` and `temp.diff.threshold`, as specified in the main text, while keeping the other parameters to fixed values: `lower.time = 22pm`, `upper.time = 3am`, `maxNightVariation = 1.5`, `sensitivity = 0.15`, `temp.diff.threshold = 3`.

I simply set parameter values for testing, by building a matrix that contains every single combination of parameter values under inspection and run `incRscan` for each of those rows (i.e. for every parameter combination). Then, I compare `incRscan`-based incubation scores (stored in a new column `incR_score`) with video-based incubation scores (in `PRESENCE`).

### Setting up matrix of parameter values:

```

# parameter values for simulation
sim.maxNightVariation <- seq (from = 0.5, to = 10, by = 0.5)
sim.sensitivity <- seq (from = 0, to = 1, by = 0.1)
sim.temp.diff.threshold <- seq (from = 0.5, to = 10, by = 0.5)

# parameter combinations for maxNightVariation
sim.maxNightVariation_table <- data.frame(sim_maxNightVariation = sim.maxNightVariation,
                                           sim_sensitivity = 0.15,
                                           sim_temp.diff.threshold = 3,
                                           batch = 1)

# parameter combinations for sensitivity
sim.sensitivity_table <- data.frame(sim_maxNightVariation = 1.5,
                                   sim_sensitivity = sim.sensitivity,
                                   sim_temp.diff.threshold = 3,
                                   batch = 2)

# parameter combinations for temp.diff.threshold
sim.temp.diff.threshold_table <- data.frame(sim_maxNightVariation = 1.5,
                                             sim_sensitivity = 0.15,
                                             sim_temp.diff.threshold = sim.temp.diff.threshold,
                                             batch = 3)

# table for every parameter combination of interest
parameter_table <- rbind(sim.maxNightVariation_table,
                         sim.sensitivity_table,
                         sim.temp.diff.threshold_table)

head(parameter_table)
nrow(parameter_table) # meaning that incRscan needs to run 51 times

```

```
clusterExport(cl = cluster_incrR, c("parameter_table", "calibration_finaldata"))
```

## Running incrRscan for every combination of parameter values

As well as running incrRscan, I compute the percentage of agreement between incrR\_score and PRESENCE for every iteration (i.e. parameter combination).

```
progressBar <- txtProgressBar(min = 0, max = nrow(parameter_table), style = 3) # use in PC
accuracy_results <- as.list(NA) # list to store results
```

```
for (i in 1:nrow(parameter_table)){
  clusterExport(cl = cluster_incrR, c("i"))
  incrRscan_list <- parLapply(cl = cluster_incrR,
                             X = calibration_finaldata,
                             fun = function(X){
                               output <- incrRscan(
                                 data = X,
                                 temp.name="temp",
                                 lower.time=22,
                                 upper.time=3,
                                 sensitivity =
                                   parameter_table[i,"sim_sensitivity"],
                                 temp.diff.threshold =
                                   parameter_table[i,"sim_temp.diff.threshold"],
                                 maxNightVariation =
                                   parameter_table[i,"sim_maxNightVariation"],
                                 env.temp = "env_temp"
                               )
                               return(output[[1]])
                             })
  # percentage of agreement between incrRscan and video footage
  accuracy <- parLapply(cl = cluster_incrR,
                       X = incrRscan_list,
                       fun = function (X) {
                         if(is.null(dim(X))){
                           return(NA)
                         } else {
                           X <- X[complete.cases(X$PRESENCE),]
                           1 - (sum(
                             abs(X$incrR_score-X$PRESENCE)) / length(X$incrR_score)
                           )
                         }
                       })
}

# results
accuracy_results[[i]] <- data.frame(box = do.call(
  args = lapply(X = calibration_finaldata,
               FUN = function(X) as.character(unique(X$BOX))),
  what = "rbind"),
  sensitivity = rep(parameter_table[i,"sim_sensitivity"],8),
  temp.diff.threshold = rep(parameter_table[i,"sim_temp.diff.threshold"],8),
  maxNightVariation = rep(parameter_table[i,"sim_maxNightVariation"],8),
```

```

    batch = rep(parameter_table[i,"batch"],8),
    accuracy = do.call(args = accuracy,
                      what = "rbind")
)

setTxtProgressBar(pb = progressBar, value = i)
}

# bind list in a data frame
calibrating_results <- rbindlist(accuracy_results)

```

## Visualising results of the calibration

Once the loop finishes the calculation, we can visualise which combination of parameters provides the highest agreement between incubation scores based on video footage and `incRscan`.

For simulation of `maxNightVariation` values:

```

calibration_plot1 <- ggplot(data = calibrating_results %>% filter(batch == 1),
                          aes(x = maxNightVariation, y = accuracy, color = box))+
  geom_point(size = 4) +
  geom_line(size = 1.5) +
  labs(x = "maxNightVariation", y = " ") +
  theme_bw() +
  theme(axis.title.x = element_text(family = "Courier New",
                                    colour="black", size=35),
        axis.text.x = element_text(size=25),
        axis.title.y = element_text(family = "Times New Roman", colour="black", size=35,
                                    margin = margin (0,25,0,0)),
        axis.text.y = element_text(size=25),
        panel.grid.minor.x=element_blank(),
        panel.grid.minor.y=element_blank(),
        panel.grid.major.y=element_blank(),
        panel.grid.major.x=element_blank(),
        legend.position="none",
        legend.text = element_text(family = "Times New Roman",
                                    size = 17))

```

For simulation of sensitivity values:

```

calibration_plot2 <- ggplot(data = calibrating_results %>% filter(batch == 2),
                          aes(x = sensitivity, y = accuracy, color = box))+
  geom_point(size = 4) +
  geom_line(size = 1.5) +
  theme_bw() +
  labs(x = "sensitivity", y = "Agreement incRscan - video recordings") +
  theme(axis.title.x = element_text(family = "Courier New",
                                    colour="black", size=35),
        axis.text.x = element_text(size=25),
        axis.title.y = element_text(family = "Times New Roman", colour="black", size=30,
                                    margin = margin (0,25,0,0)),
        axis.text.y = element_text(size=25),
        panel.grid.minor.x=element_blank(),

```

```

panel.grid.minor.y=element_blank(),
panel.grid.major.y=element_blank(),
panel.grid.major.x=element_blank(),
legend.position="none",
legend.text = element_text(family = "Times New Roman",
                             size = 17))

```

For simulation of temp.diff.threshold values:

```

calibration_plot3 <- ggplot(data = calibrating_results %>% filter(batch == 3),
  aes(x = temp.diff.threshold, y = accuracy, color = box))+
  geom_point(size = 4) +
  geom_line(size = 1.5) +
  theme_bw() +
  labs(x = "temp.diff.threshold", y = " ") +
  theme(axis.title.x = element_text(family = "Courier New",
    colour="black", size=35),
    axis.text.x = element_text(size=25),
    axis.title.y = element_text(family = "Times New Roman", colour="black", size=35,
    margin = margin (0,25,0,0)),
    axis.text.y = element_text(size=25),
    panel.grid.minor.x=element_blank(),
    panel.grid.minor.y=element_blank(),
    panel.grid.major.y=element_blank(),
    panel.grid.major.x=element_blank(),
    legend.position="bottom",
    legend.title = element_blank(),
    legend.text = element_text(family = "Times New Roman",
    size = 20),
    legend.key = element_rect(size = 5),
    legend.key.size = unit(1.7, 'lines'))

```

Averaging over nest-boxes for each set of simulations, maxNightVariation values above 1.5, sensitivity between 0 and 0.3 and temp.diff.threshold of 4 yielded the highest percentage of agreement between incRscan and video recordings (90.27%, 90.27% and 91.16% respectively - Figure below).

```

# for maxNightVariation
calibration_plot1_mean <- calibrating_results %>%
  filter(batch == 1) %>%
  group_by(maxNightVariation) %>%
  summarise(mean_accuracy = mean(accuracy, na.rm = TRUE)) %>%
  arrange(desc(mean_accuracy)) %>%
  ggplot(aes(y = mean_accuracy, x = maxNightVariation)) +
  geom_point(size = 4) +
  geom_line(size = 1.5)+
  theme_bw() +
  labs(x = "maxNightVariation", y = " ") +
  theme(axis.title.x = element_text(family = "Courier New",
    colour="black", size=35),
    axis.text.x = element_text(size=25),
    axis.title.y = element_text(family = "Times New Roman", colour="black", size=35,
    margin = margin (0,25,0,0)),
    axis.text.y = element_text(size=25),
    panel.grid.minor.x=element_blank(),
    panel.grid.minor.y=element_blank(),

```



```

panel.grid.major.y=element_blank(),
panel.grid.major.x=element_blank(),
legend.position="none",
legend.title = element_blank(),
legend.text = element_text(family = "Times New Roman",
                             size = 17))

# for sensitivity
calibration_plot2_mean <- calibrating_results %>%
  filter(batch == 2) %>%
  group_by(sensitivity) %>%
  summarise(mean_accuracy = mean(accuracy, na.rm = TRUE)) %>%
  arrange(desc(mean_accuracy)) %>%
  ggplot(aes(y = mean_accuracy, x = sensitivity)) +
  geom_point(size = 4) +
  geom_line(size = 1.5)+
  theme_bw() +
  labs(x = "sensitivity", y = "Agreement incRscan - video recordings")+
  theme(axis.title.x = element_text(family = "Courier New",
                                     colour="black", size=35),
        axis.text.x = element_text(size=25),
        axis.title.y = element_text(family = "Times New Roman", colour="black", size=30,
                                     margin = margin (0,25,0,0)),
        axis.text.y = element_text(size=25),
        panel.grid.minor.x=element_blank(),
        panel.grid.minor.y=element_blank(),
        panel.grid.major.y=element_blank(),
        panel.grid.major.x=element_blank(),
        legend.position="none",
        legend.title = element_blank(),
        legend.text = element_text(family = "Times New Roman",
                                    size = 30))

# for temp.diff.threshold
calibration_plot3_mean <- calibrating_results %>%
  filter(batch == 3) %>%
  group_by(temp.diff.threshold) %>%
  summarise(mean_accuracy = mean(accuracy, na.rm = TRUE)) %>%
  arrange(desc(mean_accuracy)) %>%
  ggplot(aes(y = mean_accuracy, x = temp.diff.threshold)) +
  geom_point(size = 4)+
  geom_line(size = 1.5)+
  theme_bw() +
  labs(x = "temp.diff.threshold", y = " ")+
  theme(axis.title.x = element_text(family = "Courier New",
                                     colour="black", size=35),
        axis.text.x = element_text(size=25),
        axis.title.y = element_text(family = "Times New Roman", colour="black", size=35,
                                     margin = margin (0,25,0,0)),
        axis.text.y = element_text(size=25),
        panel.grid.minor.x=element_blank(),
        panel.grid.minor.y=element_blank(),
        panel.grid.major.y=element_blank(),
        panel.grid.major.x=element_blank(),
        legend.position="none",

```

```

    legend.title = element_blank(),
    legend.text = element_text(family = "Times New Roman",
                               size = 17))

# plot to visualise best results per test by box
best_agreement <- calibrating_results %>%
  group_by(box, batch) %>%
  arrange(desc(accuracy)) %>%
  slice(1) %>%
  ggplot(aes(x = box, y = accuracy, fill = box)) +
  labs(x = "Nest-box", y = "Agreement incRscan - video recordings")+
  geom_point(position = position_jitter(width = 0.4),
             color = "black",
             pch = 21,
             alpha = 0.6,
             size = 6.5) +
  theme_bw()+
  theme(axis.title.x = element_text(family = "Times New Roman",
                                     colour="black",
                                     size=35),
        axis.text.x = element_text(size=25,
                                     angle = 45,
                                     family = "Times New Roman"),
        axis.title.y = element_text(family = "Times New Roman",
                                     colour="black",
                                     size=35,
                                     margin = margin (0,25,0,0)),
        axis.text.y = element_text(size=25),
        panel.grid.minor.x=element_blank(),
        panel.grid.minor.y=element_blank(),
        panel.grid.major.y=element_blank(),
        panel.grid.major.x=element_blank(),
        legend.position="none",
        legend.title = element_blank(),
        legend.text = element_text(family = "Times New Roman",
                                    size = 35))

ggsave(plot = best_agreement,
        file = "./plots/Figure 2.jpeg",
        device = "jpeg",
        height = 11, width = 13)

# creating the panel
# plotting the six plots together
source("http://peterhaschke.com/Code/multiplot.R")
ggsave(plot = multiplot(plotlist = list(calibration_plot1,
                                       calibration_plot2,
                                       calibration_plot3,
                                       calibration_plot1_mean,
                                       calibration_plot2_mean,
                                       calibration_plot3_mean),
                               cols = 2),

```

```
filename = "./plots/Supplementary_Figure_1.jpeg",  
device = "jpeg",  
height = 20, width = 15)
```

## Appendix 2

# incR: a new R package to analyse incubation behaviour

## Appendix 2

*Pablo Capilla-Lasheras (p.capilla@exeter.ac.uk / pacapilla@gmail.com)*

*2018-04-24*

## Introduction

This document provides R code to reproduce the analysis carried out in the main text of the manuscript. A brief explanation for the general coding strategy can be found in Appendix 1. The calibration of `incRscan` can be found in Appendix 1.

Change working directory and file paths according to your file system structure.

## R code to run the incR workflow

```
# install if not done before
## for developing version
devtools::install_github(repo = "incR", username = "PabloCapilla")
## for CRAN stable version
install.packages("incR")

library(incR)
packageVersion("incR") # release package version 1.1.0 (March 2018)

# install and load these packages to run the code below
library(doParallel)
library(ggplot2)
library(dplyr)
library(data.table)
library(extrafont)
loadfonts()

# environmental data for Scotland
env_Data_Scotland_2015 <-
  read.csv("2-dataCalibration/envData/env_Data2015.csv") # for 2015
env_Data_Scotland_2016 <-
  read.csv("2-dataCalibration/envData/2016/1510UT_FS_20160606.csv") # for 2016
# environmental data for the Netherlands
env_Data_NIOO_2016 <- read.csv("2-dataCalibration/envData/weathersouth_NIOO.csv")

# a sample of these data
head(env_Data_NIOO_2016)
# the other environmental temperature files have
# the same structure and column names

# nest data
incubation_files <- dir("./2-dataCalibration/data_R/", # files containing incubation data
  full.names = TRUE,
```

```

        pattern = ".csv")
rawdata_incubation <- lapply(X = as.list(incubation_files), # reads every file
                             # in incubation_files
                             FUN = function(X) read.csv(X) # produces a
                                                             # list of data frames

# checking the data are correct
lapply(X = rawdata_incubation, FUN = head)

```

In order from left to right: date column, nest temperature, presence of the incubating bird (1 = YES, 0 = NO) based on video footage, nest-box code, site (either Scotland or the Netherlands).

Once a list of data frames has been created, it is easy and fast to use `lapply` to apply any function to every element of the list (as done above). To make things even faster, I use `parLapply` from the `doPARALLEL` package to do parallel computation on three threads in my PC - see below how this is set up for 3 computing threads. *Doing parallel computing in R might sound difficult but it is actually very straightforward. Have a look at this [link](#) to find out more*

```

ncores <- 3
cluster_incR <- makeCluster(ncores)
registerDoParallel(cluster_incR)
#getDoParWorkers() # check you actually have as many working threads as you want

clusterExport(cl = cluster_incR, c("env_Data_Scotland_2015",
                                   "env_Data_NI00_2016",
                                   "env_Data_Scotland_2016",
                                   "rawdata_incubation"))

clusterEvalQ(cl = cluster_incR, library(incR))
clusterEvalQ(cl = cluster_incR, library(data.table))

```

Now, `parLapply` can be called. First, I use `incRprep`.

```

# applying incRprep
incubation_prepdata <- parLapply(cl = cluster_incR,
                                X = rawdata_incubation,
                                fun = incRprep,
                                date.name = "date00",
                                temperature.name = "temp",
                                date.format = "%d/%m/%Y %H:%M",
                                timezone = "GMT")

```

You can check that `incRprep` has worked fine for every data frame (Note that the column `PRESENCE` gives incubation scores based on video footage). New columns `index`, `time`, `hour`, `minute`, `date`, `dec_time` and `temp1` should have been created. For information about these new variables check the documentation page for `incRprep`.

```

lapply(incubation_prepdata, head, n = 3)

```

The next step, and final data preparation step, is to calculate environmental temperatures for every line and data frame. This task is done by `incRenv` and I apply the function in a similar fashion as we did for `incRprep`.

The only difference now is that we need to deal with the fact that the source of the environmental data changes for the different elements in `incubation_prepdata`: for the 1st and 2nd elements `env_Data_Scotland_2016` applies, `env_Data_Scotland_2015` for the 7th and 8th, and `env_Data_NI00_2016` for 3rd to 6th one. I solve this issue in the code below, other coding choices are certainly possible.

```

clusterExport(cl = cluster_incR, c("incubation_prepdata"))
# the computation below might take between one and ten minutes
incubation_finaldata <- parLapply(cl = cluster_incR,
  X = as.list(c(1:length(incubation_prepdata))),
  fun = function(X){
    if(X <= 2){
      environmental_data <- env_Data_Scotland_2016
    } else {
      if (X >= 7){
        environmental_data <- env_Data_Scotland_2015
      } else {
        environmental_data <- env_Data_NIOO_2016
      }
    }
    output <- incRenv (data.env =
      environmental_data,
      data.nest =
        incubation_prepdata[[X]],
      env.temperature.name =
        "tempEnv",
      env.date.name =
        "dateEnv",
      env.date.format =
        "%d/%m/%Y %H:%M",
      env.timezone=
        "GMT")

    return(output)
  }
)
# checking the new column for environmental temperatures has been added:
lapply(incubation_finaldata, head, n = 3)

```

Now we have the data ready to use `incRscan`. `incR_scan` argument values are chosen based on simulating different values and evaluating its performance (see main text for results, Appendix 1 and package vignette for detailed code to carry out such validation).

```

incRscan_list <- parLapply(cl = cluster_incR,
  X = incubation_finaldata,
  fun = function(X){
    output <- incRscan(data = X,
      temp.name="temp",
      lower.time=22,
      upper.time=3,
      sensitivity = 0.25,
      temp.diff.threshold = 4,
      maxNightVariation = 1.5,
      env.temp = "env_temp")

    return(output[[1]])
  })
# checking the tables with incubation scores based on temperatures:
lapply(incRscan_list, head, 5)

# percentage of agreement between incRscan and video footage

```

```

accuracy <- parLapply(cl = cluster_incr,
  X = incrscan_list,
  fun = function (X) {
    if(is.null(dim(X)){
      return(NA)
    } else {
      X <- X[complete.cases(X$PRESENCE),]
      1 - (sum(abs(X$incr_score-X$PRESENCE)) / length(X$incr_score))
    }
  }
)

# summary of accuracy
summary(do.call(what = "rbind", args = accuracy))

```

## Correlations between video-based and incr-based incubation attendance

```

attendance_results <- parLapply(cl = cluster_incr,
  X = incrscan_list,
  fun = function(X){
    X <- X[complete.cases(X$PRESENCE),]
    output_video <-
      incrRatt(data = X,
        vector.incubation = "PRESENCE")
    output_video$box <-
      rep(na.omit(unique(X$BOX)),
        length = nrow(output_video))
    output_incr <-
      incrRatt(data = X,
        vector.incubation = "incr_score")
    output_video$incr_scan <- output_incr[,2]
    return(output_video)
  })

attendance_results <- rbindlist(attendance_results)
cor.test(attendance_results$percentage_in, attendance_results$incr_scan)

attendance_plot <- ggplot(data = attendance_results,
  aes(x = incr_scan, y = percentage_in, color = box)) +
  geom_point(size = 7) +
  geom_abline(slope = 1, intercept = 0, linetype = 2, size = 1.5) +
  theme_bw() +
  labs(x = " ", y = " ") +
  theme(axis.title.x = element_text(family = "Courier New",
    colour="black", size=35),
    axis.text.x = element_text(size=25),
    axis.title.y = element_text(family = "Times New Roman", colour="black", size=35,
    margin = margin (0,25,0,0)),
    axis.text.y = element_text(size=25),

```



```

panel.grid.minor.x=element_blank(),
panel.grid.minor.y=element_blank(),
panel.grid.major.y=element_blank(),
panel.grid.major.x=element_blank(),
legend.position="none")

```

## Correlations between video-based and incR-based number and duration of off-bouts

```

offbout_results <- parLapply(cl = cluster_incR,
  X = incRscan_list,
  fun = function(X){
    X <- X[complete.cases(X$PRESENCE),]
    sampling_rate <- X$dec_time[5] - X$dec_time[4]

    output_bouts <- {
      incRbouts(data = X,
        vector.incubation = "PRESENCE",
        dec_time = "dec_time",
        temp = "temp",
        sampling.rate = sampling_rate)
    }$day_bouts
    output_bouts$box <- rep(na.omit(unique(X$BOX)),
      length = nrow(output_bouts))

    output_bouts$incR_scan_nbouts <- {
      incRbouts(data = X,
        vector.incubation = "incR_score",
        dec_time = "dec_time",
        temp = "temp",
        sampling.rate = sampling_rate)}$day_bouts[,3]
    output_bouts$incR_scan_timebouts <- {
      incRbouts(data = X,
        vector.incubation = "incR_score",
        dec_time = "dec_time",
        temp = "temp",
        sampling.rate = sampling_rate)}$day_bouts[,5]

    return(output_bouts)
  })

bouts_results <- rbindlist(offbout_results)
cor.test(bouts_results$number.off.bouts,bouts_results$incR_scan_nbouts)
cor.test(bouts_results$mean.time.off.bout,bouts_results$incR_scan_timebouts)

nbout_plot <- ggplot(data = bouts_results,
  aes(x = incR_scan_nbouts, y = number.off.bouts, color = box)) +
  geom_point(size = 7) +

```

```

geom_abline(slope = 1, intercept = 0, linetype = 2, size = 1.5) +
theme_bw() +
labs(x = " ", y = "Video-footage estimate") +
theme(axis.title.x = element_text(family = "Courier New",
                                   colour="black", size=35),
       axis.text.x = element_text(size=25),
       axis.title.y = element_text(family = "Times New Roman", colour="black", size=35,
                                   margin = margin (0,25,0,0)),
       axis.text.y = element_text(size=25),
       panel.grid.minor.x=element_blank(),
       panel.grid.minor.y=element_blank(),
       panel.grid.major.y=element_blank(),
       panel.grid.major.x=element_blank(),
       legend.position="none")

time_bout_plot <- ggplot(data = bouts_results,
                        aes(x = incR_scan_timebouts*60, y = mean.time.off.bout*60, color = box)) +
geom_point(size = 7) +
geom_abline(slope = 1, intercept = 0, linetype = 2, size = 1.5) +
theme_bw() +
labs(x = "incR estimate", y = " ") +
theme(axis.title.x = element_text(family = "Courier New",
                                   colour="black", size=35),
       axis.text.x = element_text(size=25),
       axis.title.y = element_text(family = "Times New Roman", colour="black", size=35,
                                   margin = margin (0,25,0,0)),
       axis.text.y = element_text(size=25),
       panel.grid.minor.x=element_blank(),
       panel.grid.minor.y=element_blank(),
       panel.grid.major.y=element_blank(),
       panel.grid.major.x=element_blank(),
       legend.position="bottom",
       legend.title = element_blank(),
       legend.text = element_text(family = "Times New Roman",
                                   size = 20),
       legend.key = element_rect(size = 5),
       legend.key.size = unit(1.7, 'lines'))

# creating the plot panel
source("http://peterhaschke.com/Code/multiplot.R")
ggsave(plot = multiplot(plotlist = list(attendance_plot, nbout_plot, time_bout_plot),
                        cols = 1), filename = "./plots/Figure 3.jpeg",
       device = "jpeg",
       height = 15, width = 10)

```

## Creating nest temperature traces

As an visual example of what the package performs, I represent the nest temperature trace of a nest box at different stages of the analysis.

```

inc_trace1 <- ggplot(data = incRscan_list[[8]] %>%
                    filter(date == "2015-05-07"),

```

```

        aes(x = dec_time, y = temp)) +
geom_point(size = 1.75, color = "grey") +
theme_bw() +
labs(y = "Temperature (°C)",
      x = "Time") +
scale_color_manual(name = "", labels = c("Absence", "Presence"),
                   values = c("#8da0cb", "#fc8d62")) +
theme(axis.title.x = element_text(family = "Times New Roman",
                                   colour="black", size=20),
      axis.text.x = element_text(size=17, family = "Times New Roman"),
      axis.title.y = element_text(family = "Times New Roman",
                                   colour="black", size=18,
                                   margin = margin (0,5,0,0)),
      axis.text.y = element_text(size=17,family = "Times New Roman"),
      panel.grid.minor.x=element_blank(),
      panel.grid.minor.y=element_blank(),
      panel.grid.major.y=element_blank(),
      panel.grid.major.x=element_blank()) +
scale_y_continuous(breaks = seq(10,38,4),
                  labels = seq(10,38,4)) +
scale_x_continuous(breaks = seq(2,24,2),
                  labels = seq(2,24,2))

ggsave(filename = "./plots/inc_trace1.jpeg", plot = inc_trace1,
        device = "jpeg", width = 5, height = 5)

inc_trace2 <- ggplot(data = incRscan_list[[8]] %>%
                    filter(date == "2015-05-07"),
                    aes(x = dec_time, y = temp)) +
geom_point(size = 1.75, color = "grey") +
geom_line(aes(y = env_temp), color = "#66c2a5", size = 1.5) +
theme_bw() +
labs(y = "Temperature (°C)",
      x = "Time") +
scale_color_manual(name = "", labels = c("Absence", "Presence"),
                   values = c("#8da0cb", "#fc8d62")) +
theme(axis.title.x = element_text(family = "Times New Roman",
                                   colour="black", size=20),
      axis.text.x = element_text(size=17, family = "Times New Roman"),
      axis.title.y = element_text(family = "Times New Roman",
                                   colour="black", size=18,
                                   margin = margin (0,5,0,0)),
      axis.text.y = element_text(size=17,family = "Times New Roman"),
      panel.grid.minor.x=element_blank(),
      panel.grid.minor.y=element_blank(),
      panel.grid.major.y=element_blank(),
      panel.grid.major.x=element_blank()) +
scale_y_continuous(breaks = seq(10,38,4),
                  labels = seq(10,38,4)) +
scale_x_continuous(breaks = seq(2,24,2),
                  labels = seq(2,24,2))

ggsave(filename = "./plots/inc_trace2.jpeg", plot = inc_trace2,

```

```

device = "jpeg", width = 5, height = 5)

inc_trace3 <- ggplot(data = incRscan_list[[8]] %>%
  filter(date == "2015-05-07"),
  aes(x = dec_time, y = temp, color = factor(incR_score)))+
  geom_point(size = 1.75) +
  geom_line(aes(y = env_temp), color = "#66c2a5", size = 1.5) +
  theme_bw() +
  labs(y = "Temperature (°C)",
  x = "Time") +
  scale_color_manual(name = "", labels = c("Absence", "Presence"),
  values = c("#8da0cb", "#fc8d62")) +
  theme(axis.title.x = element_text(family = "Times New Roman",
  colour="black", size=20),
  axis.text.x = element_text(size=17, family = "Times New Roman"),
  axis.title.y = element_text(family = "Times New Roman",
  colour="black", size=18,
  margin = margin (0,5,0,0)),
  axis.text.y = element_text(size=17,family = "Times New Roman"),
  panel.grid.minor.x=element_blank(),
  panel.grid.minor.y=element_blank(),
  panel.grid.major.y=element_blank(),
  panel.grid.major.x=element_blank(),
  legend.position="top",
  legend.text = element_text(family = "Times New Roman", size = 17))+
  scale_y_continuous(breaks = seq(10,38,4),
  labels = seq(10,38,4)) +
  scale_x_continuous(breaks = seq(2,24,2),
  labels = seq(2,24,2))

ggsave(filename = "./plots/inc_trace3.jpeg", plot = inc_trace3,
  device = "jpeg", width = 5, height = 5)

```

## Supplementary Figure 2

```

SupFig2 <- incRplot(data = data %>%
  filter(BOX == "G16_BT") %>%
  filter(date == "2015-05-09"),
  time.var = "dec_time",
  day.var = "date",
  inc.temperature.var = "temp",
  env.temperature.var = "env_temp",
  vector.incubation = "incR_score")

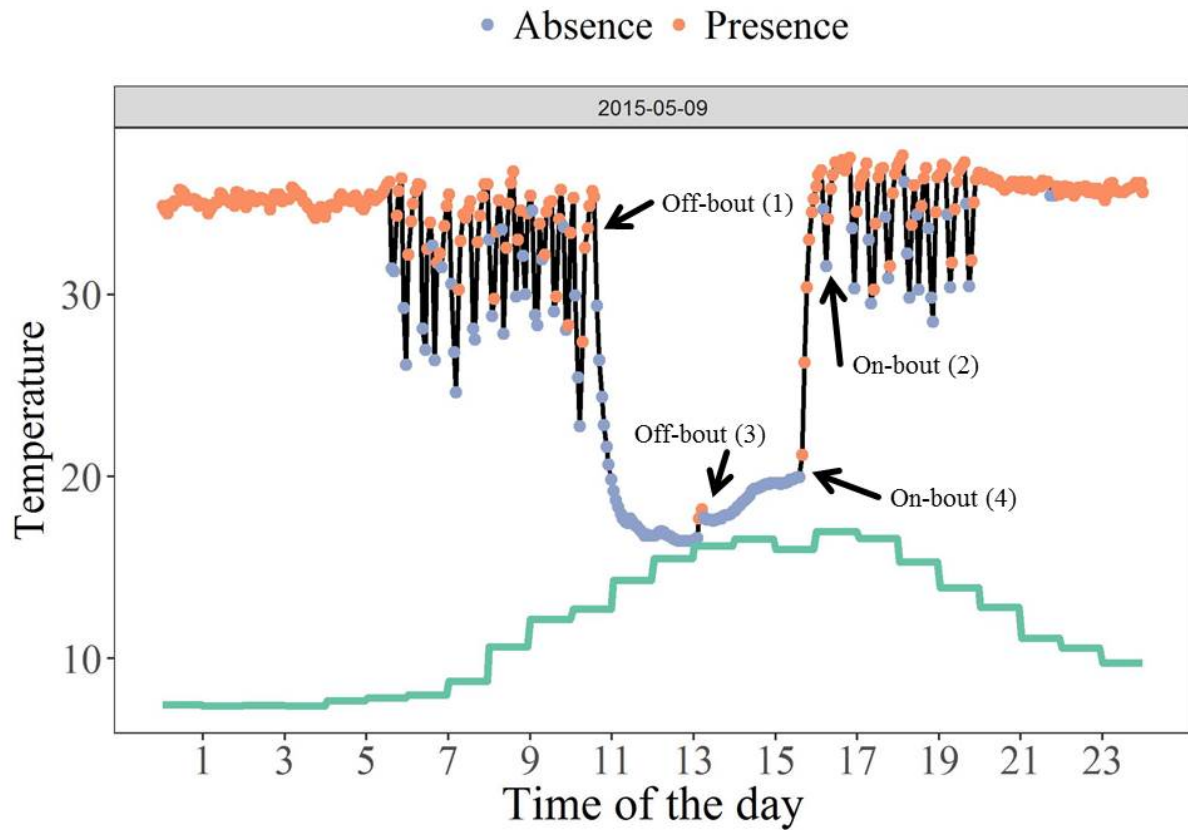
SupFig2 <- SupFig2 +
  theme(axis.title.x = element_text(family = "Times New Roman",
  colour="black", size=20),
  axis.text.x = element_text(size=17, family = "Times New Roman"),
  axis.title.y = element_text(family = "Times New Roman",
  colour="black", size=18,

```

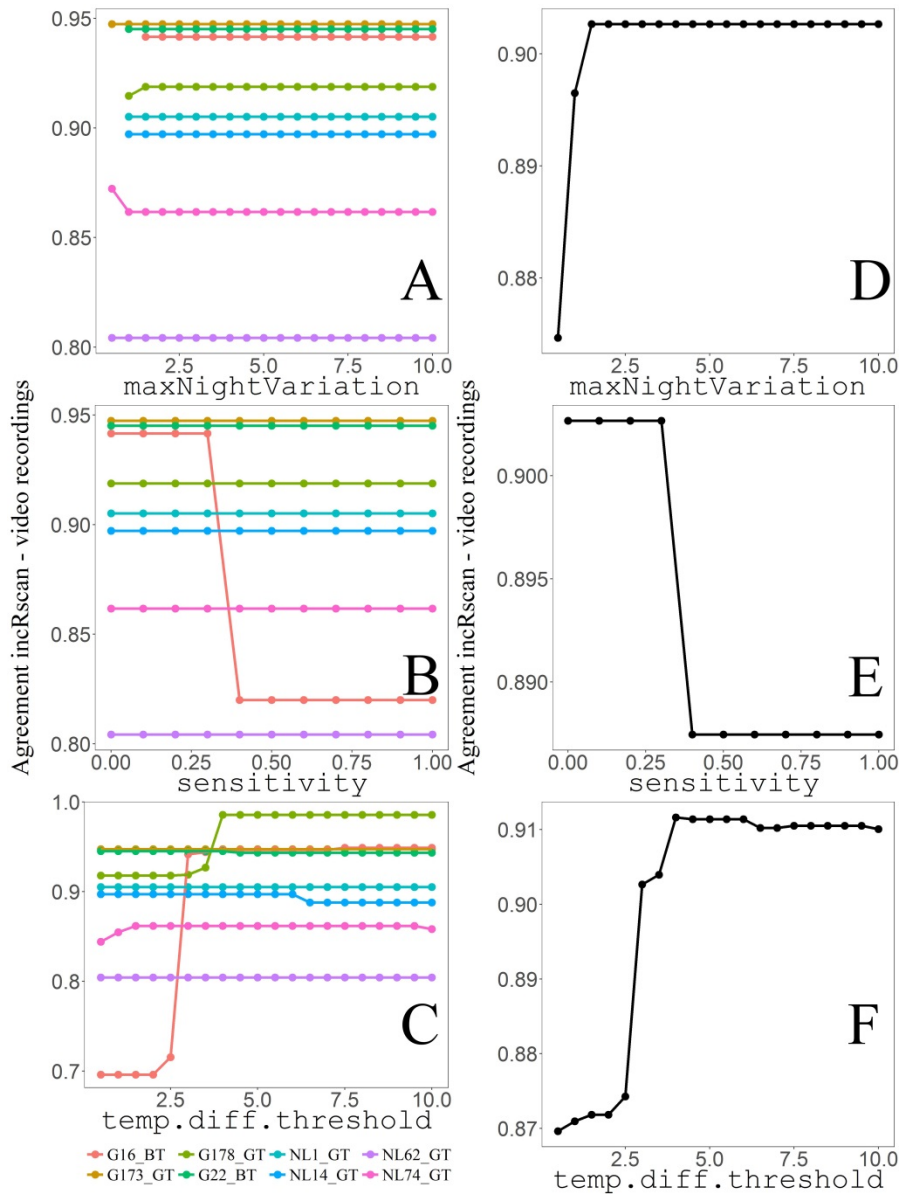
```
margin = margin (0,5,0,0),
axis.text.y = element_text(size=17,family = "Times New Roman"),
panel.grid.minor.x=element_blank(),
panel.grid.minor.y=element_blank(),
panel.grid.major.y=element_blank(),
panel.grid.major.x=element_blank(),
legend.position="top",
legend.text = element_text(family = "Times New Roman", size = 17))

ggsave(filename = "./plots/SupFig2.jpeg", plot = SupFig2,
device = "jpeg", width = 7, height = 5)
```

## Appendix 3



**Figure A1.** One day of incubation temperatures (dots connected by black line) for nest-box G16\_BT. `incR` provides the function `incRplot` to generate this plot, representing on-bouts (light-red) and off-bouts (purple) along with environmental temperatures (light-green line). Two on-bouts and off-bouts have been marked with arrows to represent examples of scenarios 1, 2, 3 and 4 (numbers in brackets) as explained in the “*Automated incubation scoring: incRscan*” section of the main text.



**Figure A2.** Results of the 1-dimensional grid search for individual nest-boxes (A, B and C) and averaging over nest-boxes (D, E, F). Plots show the percentage of agreement between incRscan-based and video-based incubation scores after varying values of maxNightVariation (A, D), sensitivity (B, E), temp.diff.threshold (C, F).

**Table A1.** Minimum, mean and maximum values (in °C) for the difference between nest and environmental temperatures across nest-boxes.

<b>Nest-box</b>	<b>Minimum</b>	<b>Mean</b>	<b>Maximum</b>
G173_GT	17.49	27.22	32.49
G178_GT	9.71	23.18	31.83
NL1_GT	8.44	17.88	19.92
NL14_GT	0.73	14.56	18.31
NL62_GT	10.61	16.82	19.44
NL74_GT	-0.98	13.80	19.41
G16_BT	1.50	25.33	28.50
G22_BT	4.04	23.94	27.39