2D distance sampling

Table of Contents

#### Load packages  
library(tidyverse)  
library(readxl)  
library(mvtnorm)  
# if(!"devtools" %in% rownames(installed.packages()))   
# {install.packages("devtools")}  
# devtools::install\_github('david-borchers/LT2D')  
library(LT2D)

#### Load 2D distance functions  
source("functions/com\_hfunctions.R")  
source("functions/com\_pifunctions.R")  
source("functions/com\_likelihoodutilities.R")  
source("functions/GoFy\_vlm.R") # custom GoFy function, modified by VLM 2022-11-11  
source("functions/plotfit.x.red.R") # custom function, modified by VLM 2023-08-31, to have a red line instead of a grey one

**Analyses for impala**

# Introduction and settings

In distance sampling surveys, the animals might avoid both the transects in the absence of observers, and the observers themselves. To correct for the effect of the behavioral responses of the animals to either the transects or the observers, we can estimate density and abundance using line transect survey data with both the forward and perpendicular distances to the observers (2D distance sampling - R LT2D package, Borchers and Cox 2017), not just the perpendicular distance. This analysis approach was also applied and recommended by Elenga et al. (2020).

Here, we rely on the functions from LT2D package (<https://github.com/david-borchers/LT2D>), as partly revised and applied in Elenga et al. (2020) (<https://github.com/cbonenfant/duikers-abundance>). With respect to the latter, we made additional minor changes to the code. The code, data and functions used are available on <https://github.com/Vale-LaMo/2D_DistanceSampling>

**To perform the analyses, initial parameters (species name, input file, percentage used for perpendicular distance data truncation, forward truncation distance in meters, etc.) are set at the beginning of this notebook (they can be customized manually, or via “Knit with parameters” in the Knit menu - except for the starting values that can only be customized manually, but normally they should not be edited - see below).** For the parameters trunc\_perp\_dist\_perc and trunc\_forw\_dist\_m, we recommend setting them respectively at 5, and at value >= than the largest forward distance. Then you can run the analyses and stop at the *Data cleaning and truncation* section to check the plots and eventually change these initial values.  
On the contrary, the parameters h.function and pi.function, and the corresponding number of parameters (n\_hpars and n\_pipars) are included in the header, but we recommend not editing them (unless you are an advanced user). The same applies for the starting\_values and the sd. All these parameters concern the function used to model the radial detection function and the density of animals vs. distance from the transect. The last parameter (n\_models) defines the number of models that are fitted with different starting values, and we recommend to set it to a large number (e.g., 100 or 200).

**Please also note that we assume that the folder in which this .Rmd file is stored includes the subfolders named *data* and *output*. The first one should contain the input data, while the second one must be created as an empty folder - output of the DS analyses will be saved there for (optional) subsequent analyses. The subfolder *functions* is also essential since it contains the customized functions used for the analyses. The whole structure can however be recreated effortlessly by forking and then cloning the GitHub repository on your local machine.** (see the README for detailed instructions)

# Detection function fitting

## Data import

First of all, we need to import the data. The dataset (Excel file) should include the following columns (order matters):

* area: surface of the study area, in km2
* transect: transect label (it could be a number or a letter)
* transect\_length: length of the transect, in km
* detected: a field whose value is 1 in case of detection of a group of animals, 0 otherwise
* object: a progressive number to identify each record (*i.e.*, 1,2,3,4,…)
* perp\_dist: perpendicular distance to the observer (only if the group has been detected)
* forw\_dist: forward distance to the observer (only if the group has been detected)
* cluster\_size: number of animals in the group (optional, only if the group has been detected)
* obs\_time: date and time stamp (optional)
* X\_observer: x-coord of the observer (optional)
* Y\_observer: y-coord of the observer (optional)

All optional fields and the distance columns can have empty cells (or *NA*) in the Excel file. On the contrary, *NA*s are not admitted in the fields: area, transect, transect\_length, detected, object (i.e., if the above order is respected, *NAs* are not admitted in the first 5 columns but can be present in the other ones). Values in the forw\_dist column can also be negative.  
In this file, we include all transects, even those for which there were no detections (and in this case, the detected column will be 0).

*Please note that when reading the file, we specify the column types, that is why the order is important. Make sure to maintain the recommended order of the columns to avoid errors in the procedure; additional columns could of course be added, or the order changed, but then you will have to modify the col\_types argument accordingly*

#### Load dataset  
data <- read\_excel(paste("data/",params$input\_file,sep=""), #sheet="template\_dataset",  
 col\_types = c(rep("numeric", 8),  
 "date",  
 "text", "text"))

## Warning: Expecting numeric in F94 / R94C6: got 'NA'

## Warning: Expecting numeric in G94 / R94C7: got 'NA'

## Warning: Expecting numeric in G97 / R97C7: got 'NA'

*The previous messages simply warn us on the presence of some NAs in the columns with numeric data. We will deal with them later but please go back and check your data if you did not expect this to happen.*

The data should look as follows:

# data$transect\_length <- data$transect\_length\*1000  
# data$area <- data$area\*1000000  
head(data)

## # A tibble: 6 × 11  
## area transect transect\_length detected object perp\_dist forw\_dist  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 487 1 16.8 1 1 55 41  
## 2 487 2 6.10 0 2 NA NA  
## 3 487 3 8.21 0 3 NA NA  
## 4 487 4 6.31 0 4 NA NA  
## 5 487 5 24.6 1 5 27 102  
## 6 487 5 24.6 1 6 69 -77  
## # ℹ 4 more variables: cluster\_size <dbl>, obs\_time <dttm>, X\_observer <chr>,  
## # Y\_observer <chr>

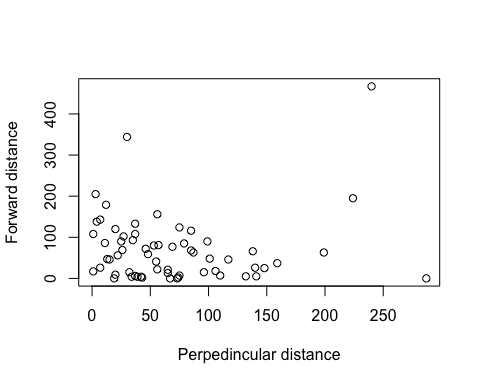
## Data cleaning and truncation

Second, we clean the data: we remove the lines of the transects without any observation (detected = 0) and we also exclude records for which distances are missing.

#### Dealing with NA and non-detections  
data\_clean <-   
 data %>%   
 filter(detected != 0, # we only include actual observations in the dataset used to fit the detection function  
 perp\_dist != "NA", # we remove lines with NA distances  
 forw\_dist != "NA")  
data\_clean$forw\_dist <- abs(data\_clean$forw\_dist) # we make sure all distances are positive (see Discussion for details)

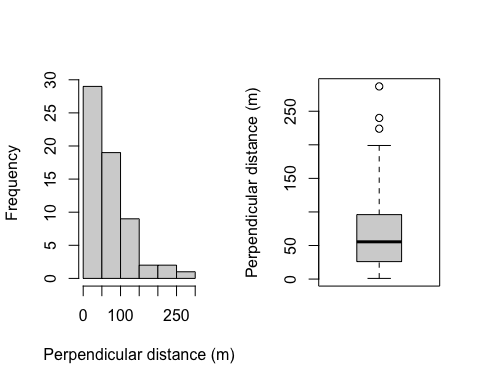
The following plot shows the distribution of forward distances with respect to perpendicular distances:

plot(data\_clean$perp\_dist, data\_clean$forw\_dist,  
 xlim=c(0,max(data\_clean$perp\_dist)),  
 ylim=c(0,max(data\_clean$forw\_dist)),  
 xlab = "Perpedincular distance",  
 ylab = "Forward distance")



We now select the truncation distances. For this purpose, we produce histograms and boxplots to identify outliers:

par(mfrow = c(1,2))  
hist(data\_clean$perp\_dist, main = "", xlab = "Perpendicular distance (m)")  
boxplot(data\_clean$perp\_dist, ylab = "Perpendicular distance (m)")



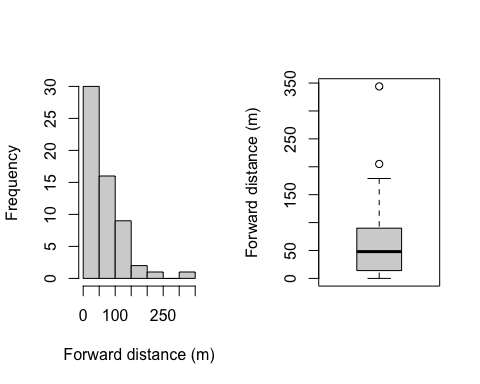
no\_data <- round(params$trunc\_perp\_dist\_perc\*length(data\_clean$perp\_dist)/100,0) # no. data to be deleted  
threshold <- sort(data\_clean$perp\_dist, decreasing = TRUE)[no\_data+1] # threshold  
data\_trunc <-   
 data\_clean %>%   
 filter(perp\_dist <= threshold)

By applying a standard truncation distance of 5%, we remove 3 record(s).

*Please note that the percentage of data can be changed by modifying the parameters of this notebook (or via the “Knit with parameters” option in the Knit menu).*

We also produce the histogram and the boxplot for the forward distances:

par(mfrow = c(1,2))  
hist(data\_trunc$forw\_dist, main = "", xlab = "Forward distance (m)")  
boxplot(data\_trunc$forw\_dist, ylab = "Forward distance (m)")



If outliers are detected in the forward distances, they can be removed by setting a custom trunc\_forw\_dist\_m parameter value. If truncation is not necessary, just set the trunc\_forw\_dist\_m to value >= than the largest forward distance.

# ystart = max(data\_trunc$forw\_dist) # change this to the desired truncation distance if necessary, e.g.  
ystart = params$trunc\_forw\_dist\_m  
data\_trunc <-   
 data\_trunc %>%   
 filter(forw\_dist <= ystart)

When all truncations have been applied, the truncated data are saved to a data\_trunc .RData file, to be used in the following (optional) DSM analyses.

save(data\_trunc, file = paste("output/data\_trunc\_", params$species\_name,".RData", sep = ""), compress = FALSE)

## Model fitting and estimation of the number of groups

We now fit 2D distance sampling model using multiple initial values to avoid local *minima* in the deviance (Elenga et al. 2020).  
 As in Elenga et al. (2020), we model the detection function in two dimensions using a radial exponential hazard risk ( under the notation of Borchers & Cox 2017), thereby making the same approximation as the half-normal detection function that is commonly found to describe the detection process in 1D. That is, we use the h.RE function of Elenga et al. (2020) for modeling the decay in detection rate with radial distance, and the pi.sigmo function of Elenga et al. (2020) for modeling the change in animal density with perpendicular distance to a line-transect (i.e., the behavioural response). See also Elenga et al. (2020), or the files “functions/com\_hfunctions.R” and “functions/com\_pifunctions.R” for alternative functions.  
Please note that the functions used are declared in the header of the document, together with the number of parameters that characterize each of them (different functions may have different number of parameters, e.g., 3 params for h.yTRE).  
200 (number set via the parameter n\_models) are fitted using parameter values extracted from a random distribution, with the initial values for the mean (starting\_values) and the sd (sd) declared in the header. The length of the starting\_values depends on the number of parameters that characterize the adopted functions (check them if you change function type).

#### Model fitting  
y = data\_trunc$forw\_dist  
x = data\_trunc$perp\_dist  
hr = params$h.function # h.yTRE not compatible with pi.sigmoI  
# these functions work: h.RE, h.IP, h.SS, h.okamura  
pi.x = params$pi.function # perpendicular distance function used  
# functions tested and working with h.RE: pi.sigmo, pi.CHN, pi.TN  
ystart = ceiling(max(y))  
w = ceiling(max(x))  
length.b = params$n\_hpars # pars for h function  
length.logphi = params$n\_pipars # pars for pi function  
length.pars = length.b + length.logphi  
debug=FALSE  
  
 FIT=list(); AICvalues=NULL  
 for (m in 1:params$n\_models) {  
 set.seed(m)  
 pars = rnorm(length.pars, # tot no. pars   
 params$starting\_values, params$sd)   
 set.seed(m)  
 tmp0 <- tryCatch.W.E (  
 fityx(y,x,pars[1:length.b],  
 hr,ystart,pi.x,  
 pars[(length.b+1):length(pars)],w,  
 control=list(),  
 hessian=TRUE,corrFlag=0.7,debug=FALSE)  
 )  
 fit = NA  
 if(! "error" %in% class(tmp0$value)) {  
 fit <- tmp0$value  
 fit$vcov <- matrix(Matrix::nearPD(fit$vcov)$mat,length.pars,length.pars)  
 }  
 FIT[[m]] = fit  
 # if(is.na(fit[1])) dev=c(dev, 1e12) else dev = c(dev, fit$AIC)  
   
 # with the funciton used in this analyses, we add the constraint that the pi.x pars should be negative  
 # to maintain the sigmoid shape  
 if(params$pi.function == "pi.sigmo" & params$h.function == "h.RE") {  
 if(!is.na(fit[1])) {  
 if(any(is.nan(fit$corr)) | any(fit$par[3:4] > 0)) {  
 AICvalues=c(AICvalues, 1e12)  
 } else {  
 AICvalues=c(AICvalues, fit$AIC)  
 }  
 } else {  
 AICvalues=c(AICvalues, 1e12)  
 }  
 } else {  
 if(!is.na(fit[1])) {  
 if(any(is.nan(fit$corr))) {  
 # if(all(fit$b > 0)) {  
 AICvalues=c(AICvalues, 1e12)  
 } else {  
 AICvalues=c(AICvalues, fit$AIC)  
 }  
 } else {  
 AICvalues=c(AICvalues, 1e12)  
 }  
 }  
 }  
# fitVU = FIT[[which.min(dev)]]  
# tabVU = matrix(NA,2,3)  
# if(is.na(fitVU[1])) tabVU = matrix(NA,2,3) else {  
# # set.seed(10)  
# tmp1 <- tryCatch.W.E (boot(fitVU))  
# if(! "error" %in% class(tmp1$value)) tabVU=tmp1$value  
# }  
# # tabVU # the CIs for the average p and the N of groups are generated by bootstrap  
# if(!is.numeric(unlist(tabVU))) print("error!")  
# save(fitVU, file = paste("output/fitVU\_", params$species\_name, ".RData", sep = ""), compress = FALSE)  
save(FIT, file = paste("output/FIT\_", params$species\_name, ".RData", sep = ""), compress = FALSE)

The best models are identified according to AIC values (AIC < 2):

data.frame(m = 1:params$n\_models, modAIC = AICvalues) -> df.AIC  
df.AIC %>%   
 arrange(modAIC) %>%   
 filter(modAIC <= min(df.AIC$modAIC) + 2) -> tab.AIC  
tab.AIC

## m modAIC  
## 1 141 1110.894  
## 2 76 1110.899

# # tryCatch.W.E(plotfit.x.red(x[x<=w],fitVU,nclass=20,nint=100));rug(x[x<=w])  
# # see https://github.com/david-borchers/LT2D/blob/master/inst/FitsForPaper.r  
# # the original plotfit.x function has been modified to customize the colors  
# # tryCatch.W.E(plotfit.x(x[x<=w],fitVU,nclass=20,nint=100));rug(x[x<=w]) # greyscale  
# for (i in 1:length(tab.AIC$m)) {  
# plotfit.x.red(x[x<=w],FIT[[tab.AIC$m[i]]],nclass=20,nint=100);rug(x[x<=w])  
# }

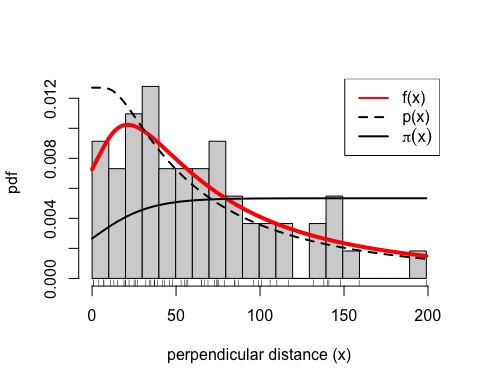
Among them, if more than one, we finally select the model with the lowest CV and we plot it. The figure shows the actual distribution of animals (continuous black line), the “observed” detection function (bold red line) and the “corrected” detection function (dashed black line), that takes into account the behavioural response

CV.phat.values <- vector("numeric", length(tab.AIC$m))  
for (i in 1:length(tab.AIC$m)) {  
 fName = params$h.function  
 CV.phat.values[i] <- phatModels(list(FIT[[tab.AIC$m[i]]]))$CV.phat  
 # LT2D::phatModels(modList = list(FIT[tab.AIC$m[i]]))$CV.phat  
}  
(tab.AIC %>%  
 mutate(CV.phat = CV.phat.values) %>%  
 filter(CV.phat == min(CV.phat)) -> best\_mod)

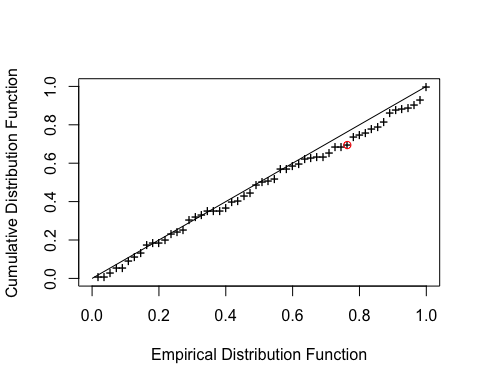
## m modAIC CV.phat  
## 1 141 1110.894 0.1885847

fitVU = FIT[[best\_mod$m]]  
save(fitVU, file = paste("output/fitVU\_", params$species\_name, ".RData", sep = ""), compress = FALSE)

# tryCatch.W.E(plotfit.x.red(x[x<=w],fitVU,nclass=20,nint=100));rug(x[x<=w])  
# see https://github.com/david-borchers/LT2D/blob/master/inst/FitsForPaper.r  
# the original plotfit.x function has been modified to customize the colors  
plotfit.x.red(x[x<=w],fitVU,nclass=20,nint=100);rug(x[x<=w])

 We now perform checks on the model, to verify the goodness of fit in the perpendicular dimension (Kolmogarov-Smirnov and Cramer-von Mises p-values are also reported):

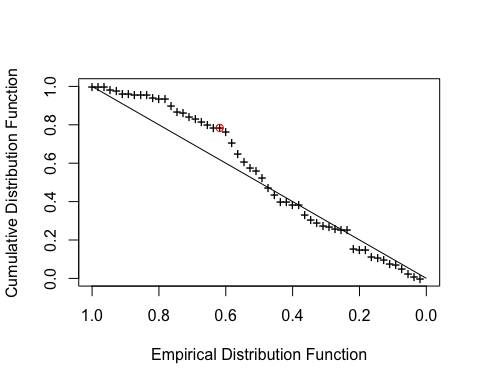
fName = params$h.function  
# GoF for perpendicular distances  
GoFx(fitVU,plot=TRUE)$pvals



## Cramer-von Mises Kolmogarov-Smirnov   
## 0.9100113 0.9561854

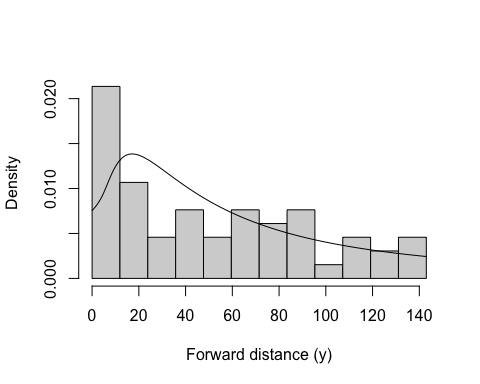
We also verify the goodness of fit in the forward dimension

# GoF for forward distances  
fName = params$pi.function  
GoFy\_vlm(fitVU,plot=TRUE)$pvals



## Cramer-von Mises Kolmogarov-Smirnov   
## 0.06775448 0.09714169

# plotfit.smoothfy(fitVU,nclass=32);rug(x=y[x<=w])  
# plotfit.y(y[x<=w & y<=ystart],x,fitVU,nclass=20);rug(x=y[x<=w])  
plotfit.smoothfy(fitVU,xmax=199)



Finally, we summarise the results on the **detection probability** and the **number of groups** in the surveyed region:

(LT2D::phatModels(modList = list(fitVU), # same as fitVU  
 n=length(na.omit(data\_trunc$cluster\_size))) -> stats\_df\_groups)

## phat CV.phat interval lower.bound upper.bound n Nhat NhatLower  
## 1 0.3651181 0.1885847 0.95 0.2531105 0.5266919 55 150.6362 104.4254  
## NhatUpper  
## 1 217.2964

# LT2D::phatModels(modList = list(FIT[76]))

Indeed, number of groups in the surveyed area is estimated by dividing the number of observed groups (n) for the detection probability. The following table shows the estimated number of groups and the density per square km:

length(na.omit(data\_trunc$cluster\_size))/(phatInterval(fitVU))[1] -> no\_groups  
names(no\_groups) <- "no\_groups"  
  
data %>%   
 mutate(transetto = factor(transect)) %>%  
 dplyr::group\_by(transetto) %>%   
 dplyr::summarise(no\_groups\_transect = sum(detected),  
 transect\_length = mean(transect\_length)) %>%   
 mutate(encounter\_rate = no\_groups\_transect/transect\_length) -> res  
# res  
  
(2\*(w/1000)\*sum(res$transect\_length)) -> surveyed\_area # here, the truncation distance is divided by 1000, to express the density in km2  
(no\_groups/surveyed\_area) -> dens\_groups\_km2   
names(dens\_groups\_km2) <- "dens\_groups\_km2"  
cbind(no\_groups, dens\_groups\_km2)

## no\_groups dens\_groups\_km2  
## 1 150.6362 0.5659583

## Cluster size stats and estimated number of individuals with CV

In order to estimate the number of individuals, we now consider the cluster size data, summarizing them and calculating the cluster size standard deviation:

data\_clustersize <-   
 data %>%   
 filter(detected != 0,  
 perp\_dist != "NA",  
 forw\_dist != "NA",  
 perp\_dist <= w,  
 forw\_dist <= ystart)  
data\_clustersize$forw\_dist <- abs(data\_clustersize$forw\_dist)  
print("Cluster size base stats:")

## [1] "Cluster size base stats:"

summary(data\_clustersize$cluster\_size)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.000 2.000 4.000 6.473 8.000 42.000

print("Cluster size standard deviation:")

## [1] "Cluster size standard deviation:"

sd(data\_clustersize$cluster\_size)

## [1] 7.809905

The estimated abundance of individual animals (abund\_survey\_individuals) is obtained by multiplying the estimated number of groups (no\_groups) for the mean cluster size (obviously, if all groups are made of 1 individual only, the estimates of the number of groups and animals are equal).

no\_groups\*mean(data\_clustersize$cluster\_size) -> abund\_survey\_individuals # estimated abundance, individuals  
data.frame(abund\_survey\_individuals[1],abund\_survey\_individuals/surveyed\_area) -> df  
names(df) <- c("no\_individuals","dens\_individuals\_km2")  
df

## no\_individuals dens\_individuals\_km2  
## 1 975.027 3.663293

Then, we can estimate the overall coefficient of variation of this estimate using the Delta method. According to this approximation, when two or more components are multiplied together, the squared CVs add. In this case the components of the formula to estimate the abundance (or density) are the encounter rate, the detection function and the cluster size.

cv\_encounterrate <- (sd(res$encounter\_rate)/mean(res$encounter\_rate))  
cv\_detfunc <- (phatInterval(fitVU)[2])  
cv\_clustersize <- (sd(data\_clustersize$cluster\_size)/mean(data\_clustersize$cluster\_size))  
  
cv\_tot <- sqrt(cv\_detfunc^2 + cv\_clustersize^2 + cv\_encounterrate^2)  
  
component = c("Encounter rate", "Cluster size", "Detection function", "Abundance")  
CV = c(cv\_encounterrate, cv\_clustersize, cv\_detfunc[[1]], cv\_tot[[1]])  
sd = c(cv\_encounterrate\*mean(res$encounter\_rate),  
 cv\_clustersize\*mean(data\_clustersize$cluster\_size),  
 (cv\_detfunc\*phatInterval(fitVU)[1])[[1]],  
 (cv\_tot\*abund\_survey\_individuals)[[1]])  
data.frame(component, CV = round(CV,3), sd = round(sd, 3))

## component CV sd  
## 1 Encounter rate 1.846 0.191  
## 2 Cluster size 1.207 7.810  
## 3 Detection function 0.189 0.069  
## 4 Abundance 2.214 2158.444

The table with the details of the CVs allows to identify potential issues, e.g. components that strongly affect the overall coefficient of variation.

# Results - summary

Finally, the essential results of the survey:

stats <- list(dim(data\_trunc)[1],  
 paste(min(data\_trunc$perp\_dist)," - ",max(data\_trunc$perp\_dist)),  
 paste(min(data\_trunc$forw\_dist)," - ",max(data\_trunc$forw\_dist)),  
 paste(params$h.function,"/",params$pi.function),  
 fitVU$AIC,  
 dim(res)[1],  
 sum(res$transect\_length),  
 surveyed\_area)  
names(stats) <- c("Number of oservations", "Perpendicular distance range (m)",  
 "Forward distance range (m)", "Model",  
 "AIC", "Number of transects",  
 "Effort (km)", "Surveyed area (km2)")  
as.data.frame(do.call(rbind, stats)) -> statistics  
colnames(statistics) <- NULL  
statistics

##   
## Number of oservations 55  
## Perpendicular distance range (m) 1 - 199  
## Forward distance range (m) 0 - 143  
## Model h.RE / pi.sigmo  
## AIC 1110.89361741092  
## Number of transects 71  
## Effort (km) 668.747  
## Surveyed area (km2) 266.161306

D <- (abund\_survey\_individuals[[1]]/surveyed\_area)  
varD <- (cv\_tot^2) \* (D^2)  
log(1+(varD/D^2)) -> var\_logD  
C <- exp(1.96\*sqrt(var\_logD))  
D\_min <- D/C[[1]]  
D\_max <- D\*C[[1]]  
ind\_min <- D\_min\*surveyed\_area  
ind\_max <- D\_max\*surveyed\_area  
# based on Buckland et al. 1993, Ch. 3, pp. 88-89  
  
# rows = c("Average p", "N groups", "N individuals")  
Estimate = c(stats\_df\_groups$phat,  
 stats\_df\_groups$Nhat,  
 abund\_survey\_individuals[[1]])  
Lower = c(stats\_df\_groups$lower.bound, stats\_df\_groups$NhatLower, ind\_min)  
Upper = c(stats\_df\_groups$upper.bound, stats\_df\_groups$NhatUpper, ind\_max)  
data.frame(Estimate = round(Estimate,3),  
 Lower = round(Lower,3),  
 Upper = round(Upper,3)) -> results  
row.names(results) <- c("Average p", "N groups", "N individuals")  
results

## Estimate Lower Upper  
## Average p 0.365 0.253 0.527  
## N groups 150.636 104.425 217.296  
## N individuals 975.027 71.603 13277.081

# References

Borchers DL, Cox MJ (2017) Distance sampling detection functions: 2D or not 2D? Biometrics 73(2):593-602. <https://doi.org/10.1111/biom.12581>  
Elenga G, Bonenfant C, Péron G (2020) Distance sampling of duikers in the rainforest: Dealing with transect avoidance. PLOS ONE 15(10): e0240049. <https://doi.org/10.1371/journal.pone.0240049>