Code for producing results and figures of Marques et al. 2023

Marques, T. A., Marques, C. S. and Gkikopoulou, K. C.

May 22, 2023

Contents

Introduction

This document presents the code required to reproduce the figures and resulting in the manuscript "A sperm whale cautionary tale about estimating acoustic cue rates for deep divers", submitted to The Journal of the Acoustical Society of America, by Marques, T. A., Marques, C. S. & and Gkikopoulou, K. C.

There are a couple of datasets considered:

1. a dataset used to produce the depth profile and the corresponding deep dive cycles in figure 1; this data is within object `depthprof1` in the file `data_article_cue_rate_depth_profile_sw02_254c.rda`, which is loaded as is as data;
2. a dataset with summaries of numbers of regular echolocation clicks per deep dive cycle, for each of the sperm whale tags considered on the manuscript, namely the cue rates per deep dive cycle, corresponding to `ddata1`, the single object in `data_4_article_clickrates_deep_dive.rda`.

Both `.rda` files named above were created via an internal ACCURATE document from the data that corresponds to the times of detections for each echolocation click from the focal animal detected in each tag, and those times were obtained from the DTAG raw sound files as described in the methods section of the paper. For future reference, the processing of the objects considered here was done in an RMarkdown dynamic report entitled `Cue_Rates_For_Sperm_Whales.Rmd`. This document itself is not shared because the corresponding data required to process it is not public yet. We are planning on making that data (i.e. the times of each echolocation click from the tagged animal found on each of the tags) public after the publication of a separate paper in preparation about the estimation of cue rates for sperm whales, where factors affecting the estimated cue rates will be explored.

Two different sections follow. In section 2 we plot an example depth profile, Figure 1 on the paper. In section 3 we produce the cue rate estimates using different approaches.

DDC data reading and processing

We begin by reading the deep dive cycle data in:

```
# files created in Cue_Rates_For_Sperm_Whales.Rmd
# Reading the data that contain the information per deep dive cycle - object ddata1
load("data_4_article_clickrates_deep_dive.rda")
#removing the tags for animals we know were exposed to sonar
DDCs<-ddata1[ddata1$sonar!="sonar",]
```

and from it, we aggregate data to create a per tag dataset,

```
# Creating the data per tag
tags<-DDCs%>%
  group_by(tag)%>%
  summarise(location=unique(location), year=unique(year), sex=unique(sex),
  duration= sum(durations,na.rm=T),nclicks=sum(nclick,na.rm=T),
  crate=sum(nclick,na.rm=T)/sum(durations,na.rm=T),ddc=max(absdives+1,na.rm = T))
```

We had 826 DDCs across a total of 104 whales, with a median number of 7 DDC per tag of, ranging from 1 to 31 DDC per tag. Tag durations ranged from 0.271 to 25.759 hours. The observed cue rates both per tag varied between 0.127 and 1.611, with a median value of 0.954, while in the case of DDC these ranged between 0 and 1.934, with a median value of 0.992.

Below we present a table where the locations and years covered by these tags:

```
kable(table(tags$location,tags$year))
```

	2001	2002	2003	2005	2009	2010	2013	2014	2015	2016	2017	2018	2019
Azores	0	0	0	0	0	8	0	0	0	0	0	3	5

	2001	2002	2003	2005	2009	2010	2013	2014	2015	2016	2017	2018	2019
DOMINICA	0	0	0	0	0	0	0	7	16	4	1	4	0
Gulf of Mexico	4	14	8	0	0	0	0	0	0	0	0	0	0
Kaikoura	0	0	0	0	0	0	6	0	0	0	0	0	0
Mediterranean	1	0	7	0	0	0	0	0	0	0	0	0	0
North Atlantic	0	0	7	0	0	0	0	0	0	0	0	0	0
Delaware													
Norway	0	0	0	0	1	3	0	0	0	2	0	0	2
Norway Andenes	0	0	0	1	0	0	0	0	0	0	0	0	0

The above we refer in the paper to as the complete dataset.

We also create corresponding reduced datasets. These correspond to removing all DDCs longer than 1 hour (i.e. 3600 seconds)

```
DDCsR<-DDCs[DDCs$durations<3600,]
# Obtaining the reduced data per tag;
tagsR<-DDCsR %>%
 dplyr:: group_by(tag) %>%
  dplyr::summarise(location=unique(location), year=unique(year), sex=unique(sex),
  duration= sum(durations,na.rm=T),nclicks=sum(nclick,na.rm=T),
  crate=sum(nclick,na.rm=T)/sum(durations,na.rm=T))
```

Figure 1: a depth profile and the corresponding deep dive cycles

Reading the data

```
# file created in Cue_Rates_For_Sperm_Whales.Rmd
# an example depth profile - in this case whale sw02_254c
# loads a single object: depthprof
load("data_article_cue_rate_depth_profile_sw02_254c.rda")
```

This is an example depth profile. This is the depth profile associated with tag sw02_254c, a whale tagged in the Gulf of Mexico in 2002. We thank Mark Johnson for sharing this full depth profile data for the purpose of this manuscript.

We first get an indicator for the start of each deep dive cycle:

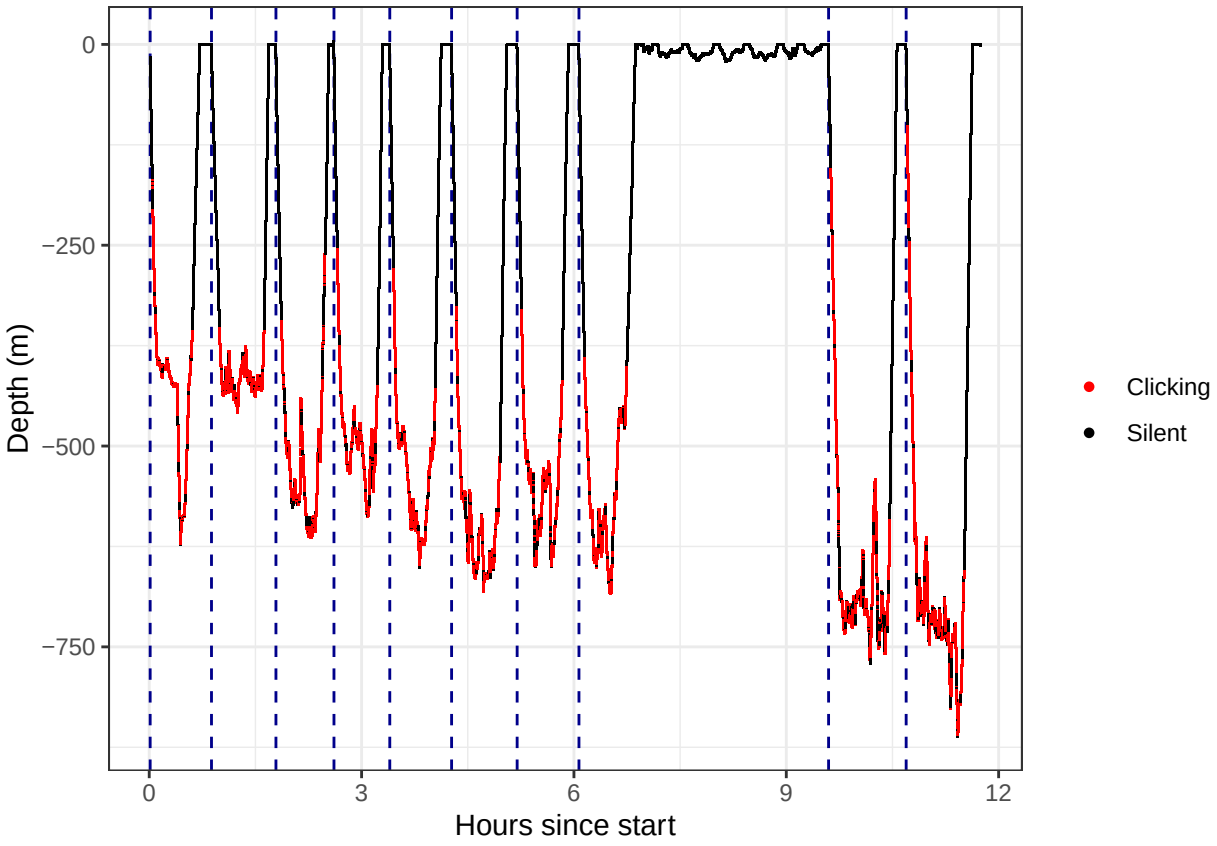
```
#Next we delete the beggining of the tag
depthprof1<-depthprof1[depthprof1$countdives!="after_tag",]
# Making the variable countdives numeric
depthprof1$countdives<-as.numeric(as.character(depthprof1$countdives))
# Get an indicator for the first second in each dive, to add vertical bars to plot
# note: if you do not explicit package "dplyr" you might get the "stats" version of "lag"
# obtain the difference between the current value
# for the dive number and the previous row value for the dive number
depthprof1$lag<-depthprof1$countdives-dplyr::lag(depthprof1$countdives)
#if the difference in the lag is NA it means it corresponds to
#the first observation of the column, aka the beginning of the first dive.
depthprof1$lag<-ifelse(is.na(depthprof1$lag),1,depthprof1$lag)
# select where the difference is 1, which correspond to the place where the dive begins
depthprofline<-depthprof1[depthprof1$lag==1,]
# remove unwanted values / shouldn't change the results
depthprofline<-depthprofline[!is.na(depthprofline$lag) &
    !is.na(depthprofline$countdives) & !is.na(depthprofline$nclick),]
```

We create a variable for coloring the plot according to whether the animal is clicking or not:

```
depthprof1$clickstatus<-ifelse(depthprof1$nclick==0,"Silent","Clicking")
```

Here we produce figure 1 in the paper

```
ggplot((depthprof1),aes(x=second/(60*60), y=-depth,color=clickstatus))+
  geom_vline(xintercept = depthprofline$second/(60*60),color="blue4",shape=".",
  linetype='dashed')+
  geom_point(shape=".")+
  ylab("Depth (m)")+xlab("Hours since start")+
  guides(colour = guide_legend(override.aes = list(size=1.2,shape=19)))+
  labs(color="")+
  scale_color_manual(values=c("red","black"))+theme_bw()
```




```
#this is the code to make a production ready version of figure 1
ggsave("Figures/Figure1.jpeg",dpi=600)
```

and finally we do not evaluate here but provide the code to create an animated gif that might be used to illustrate the depth profile highlighted in the paper:

```
# If you want to animate the previous figure
#change echo= FALSE,eval=FALSE to echo= TRUE,eval=TRUE
wake33<-ggplot(subset(depthprof,!is.na(clickstatus)),aes(x=second/(60*60),
  y=-depth,color=clickstatus))+
  geom_point()+
  ylab("Depth (m)")+xlab("Hours since start")+
```

```
  xlim(min(depthprofline$second/(60*60)),max(depthprofline$second/(60*60)))+
  guides(colour = guide_legend(override.aes = list(size=1.2,shape=19)))+
  labs(color="")+
  scale_color_manual(values=c("red","black"))+theme_bw()+
  transition_time(time = second/(60*60)) + shadow_wake(wake_length = 0.1)
wake33 %>% animate(detail = 5, nframes = 100, type = "cairo", duration=20)
```

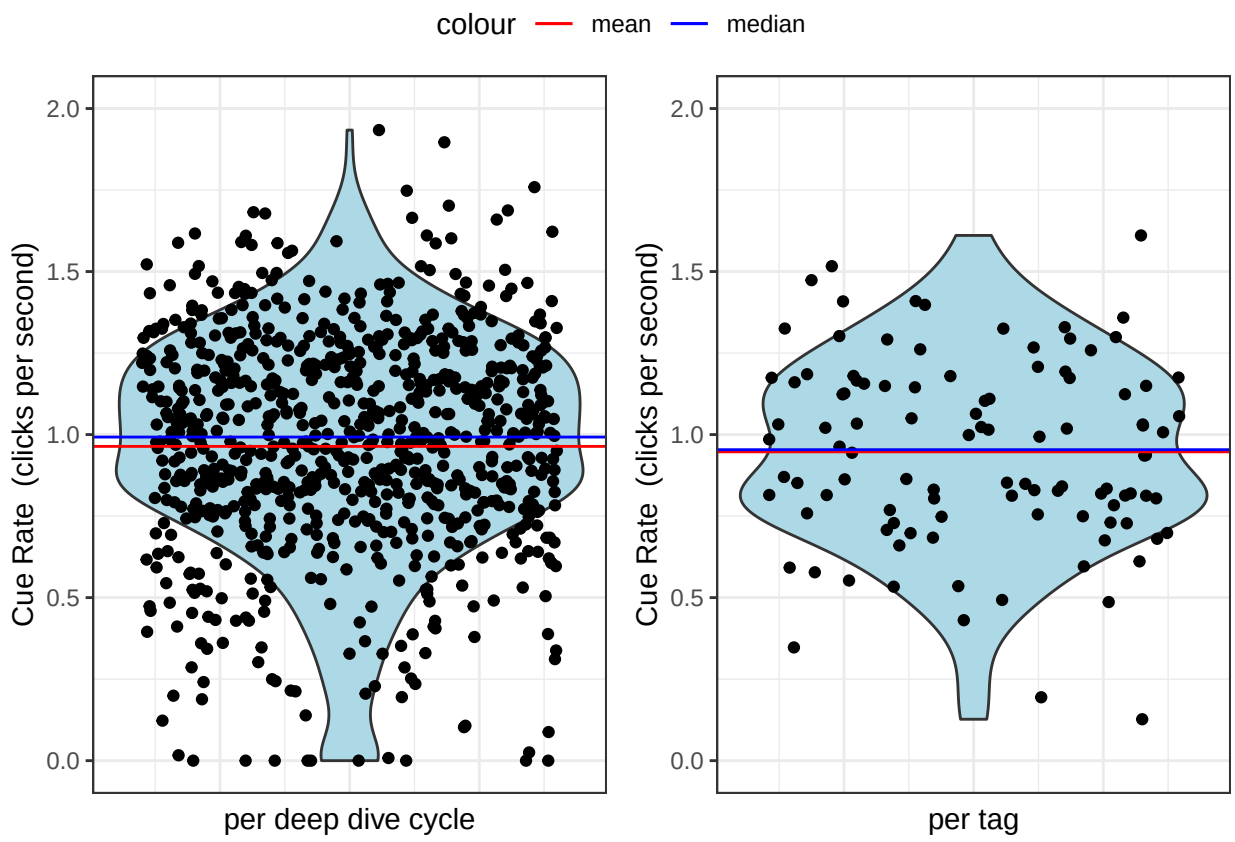
Figure 2 - cue rates per tag and per dive

We represent visually the cue rates when considering the deep dives as the sampling unit, and the corresponding cue rates when considering the tag as the sampling unit.

```
fig1A<-ggplot(subset(DDCs,!is.na(absdives)),aes(x=1,y=crate),fill="lightblue")+
  theme_bw()+geom_violin(fill="lightblue")+geom_jitter()+
  theme(axis.text.x = element_text(angle = 45, vjust = 0.5, hjust=0.5))+
  ylab("Cue Rate  (clicks per second)")+xlab("per deep dive cycle")+
  geom_hline(aes(yintercept = mean(crate,na.rm=T),color="mean"))+
  geom_hline(aes(yintercept = median(crate,na.rm=T), color="median"))+
 # stat_summary(fun.data = give.n, geom = "text", fun.y = median, colour = "red",size=4) +
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank())+
  scale_color_manual(values=c("red","blue"))+ylim(0,2)


fig1B<-ggplot(tags,aes(x=1,y=crate),fill="lightblue")+theme_bw()+
  geom_violin(fill="lightblue")+
  geom_jitter()+
  theme(axis.text.x = element_text(angle = 45, vjust = 0.5, hjust=0.5))+
  ylab("Cue Rate  (clicks per second)")+
  xlab("per tag")+geom_hline(aes(yintercept = mean(crate,na.rm=T),color="mean"))+
  geom_hline(aes(yintercept = median(crate,na.rm=T), color="median"))+
 # stat_summary(fun.data = give.n, geom = "text", fun.y = median, colour = "red",size=4) +
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank())+
  scale_color_manual(values=c("red","blue"))+ylim(0,2)

ggarrange(fig1A,fig1B,common.legend = TRUE)
```

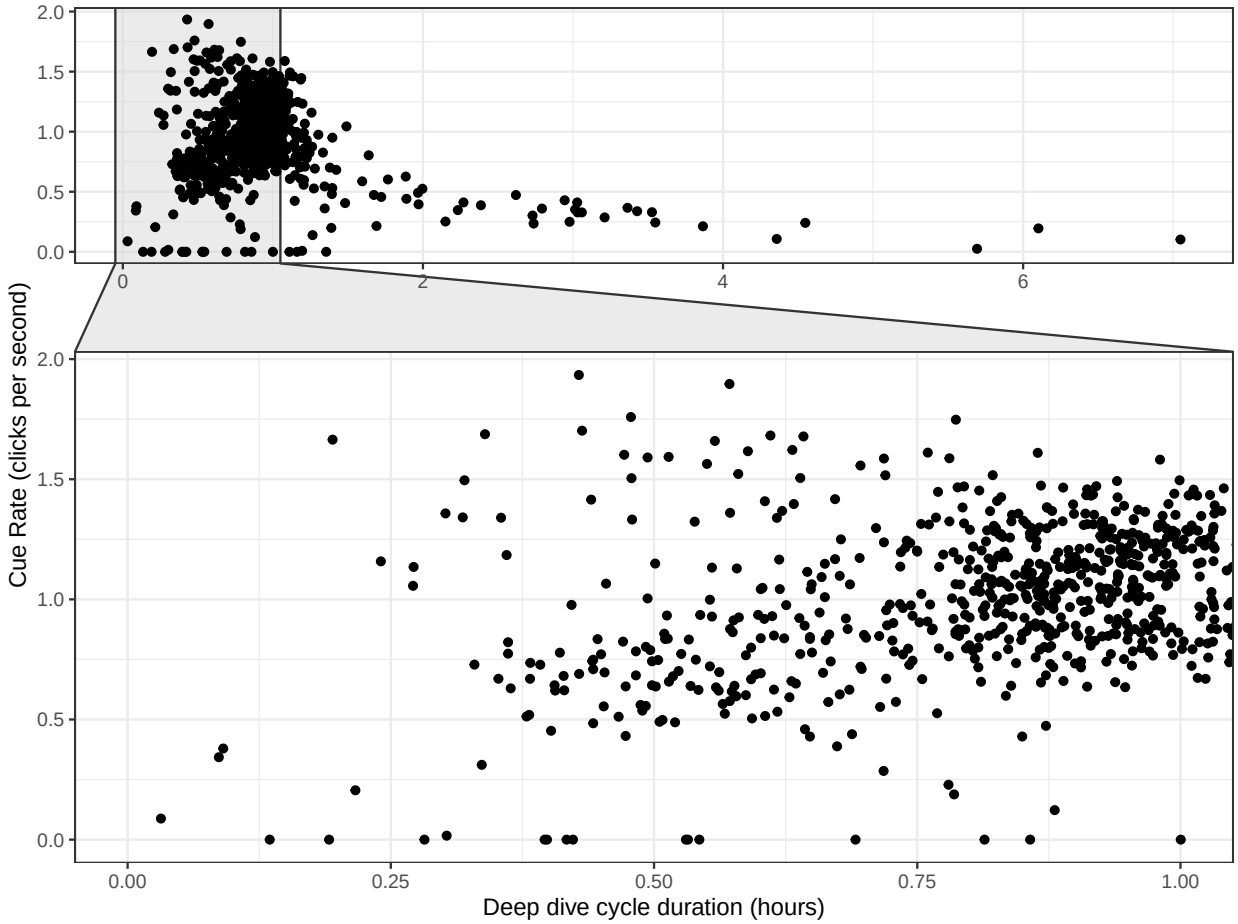



```
#this is the code to make a production ready version of figure 2
ggsave("Figures/Figure2.jpeg",dpi=600)
```

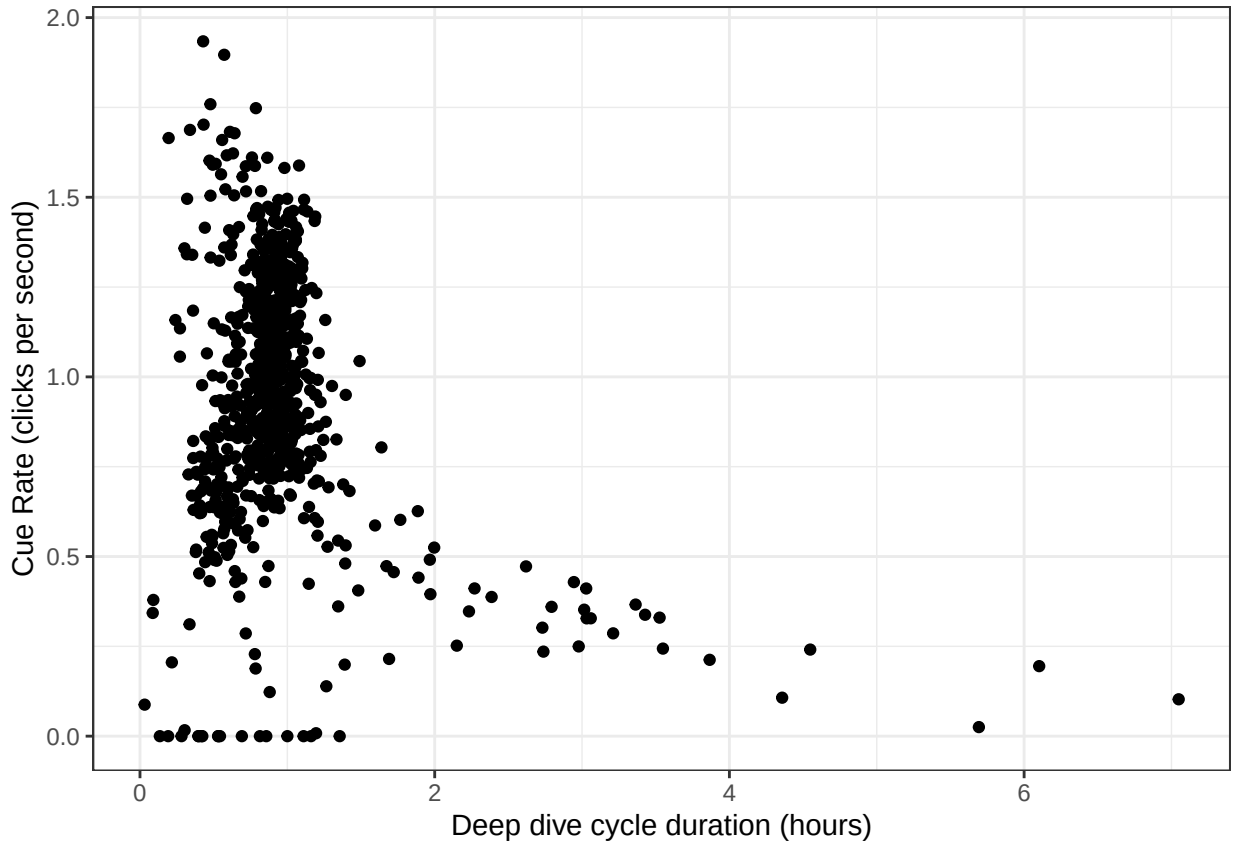
Figure 3: cue rates as a function of DDC duration

Below we represent the relationship between the cue rate and the deep dive cycle length, which will be helpful to interpret some of the results

```
ggfig1 <- ggplot(DDCs,aes(y=crate,x=durations/(60*60)))+
  geom_point()+
  xlab("Deep dive cycle duration (hours)")+
  ylab("Cue Rate (clicks per second)")+
  theme_bw()+
  ggforce::facet_zoom(xlim = c(0, 3600/(60*60)))
ggfig1
```

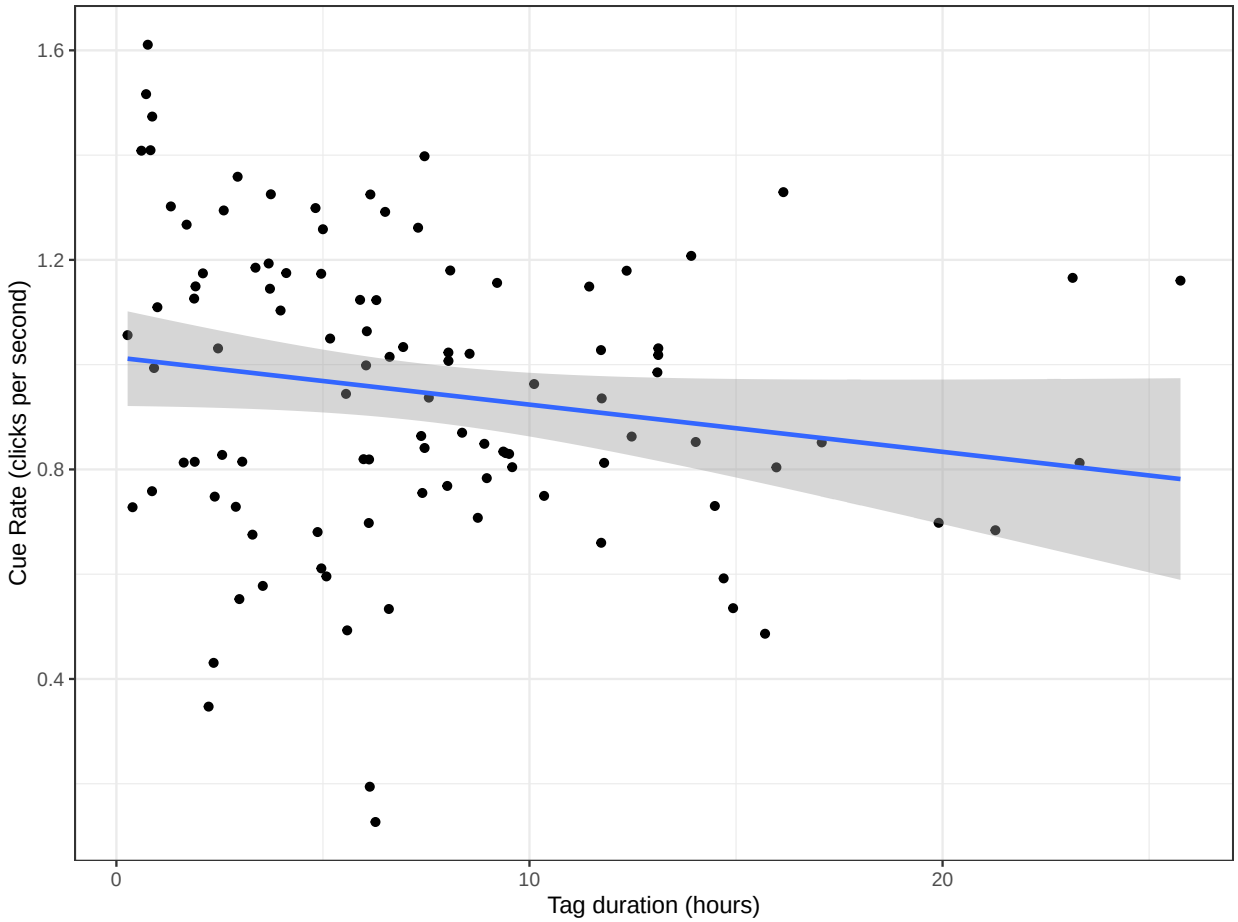



```
ggplot(DDCs,aes(y=crate,x=durations/(60*60)))+
  geom_point()+
  xlab("Deep dive cycle duration (hours)")+
  ylab("Cue Rate (clicks per second)")+
  theme_bw()
```



Looking at the same relation at the tag level:

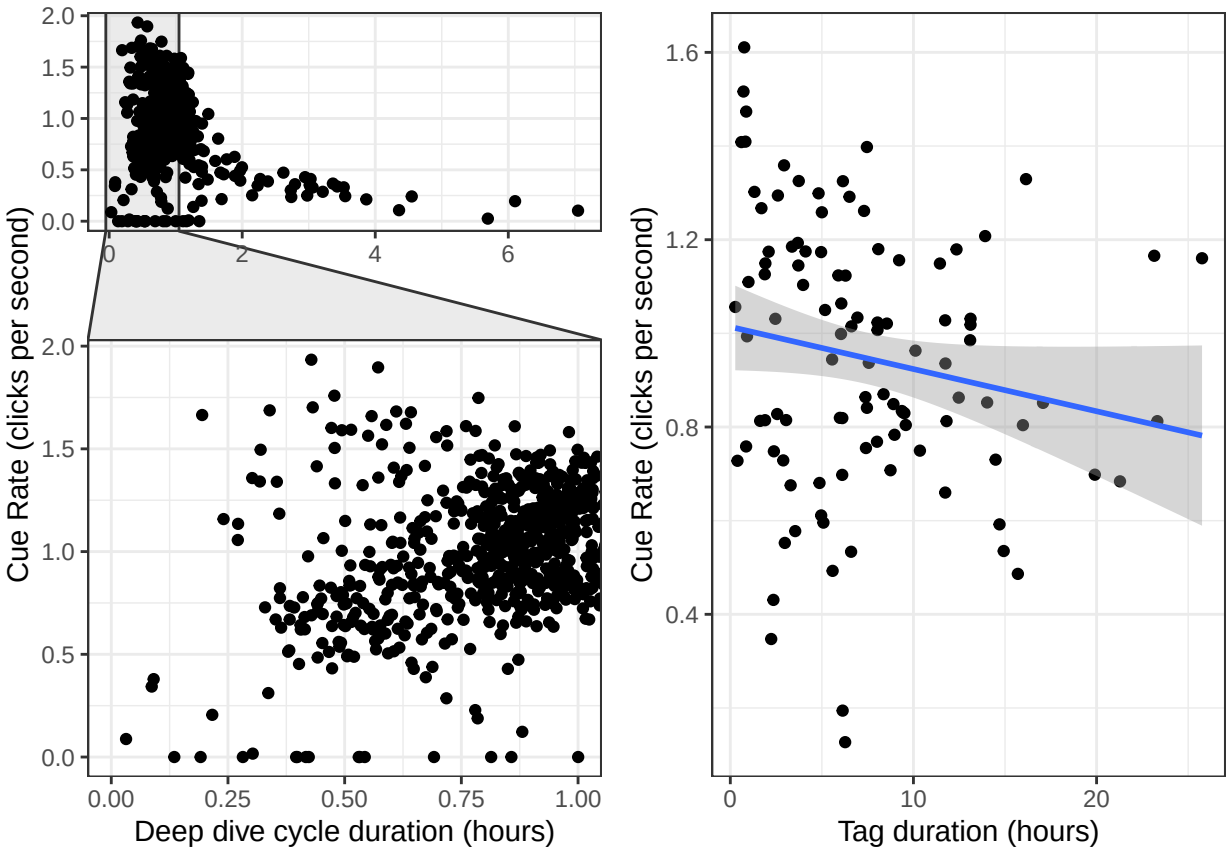
```
ggfig2 <- ggplot(tags,aes(y=crate,x=duration/(60*60)))+
  geom_point()+
  geom_smooth(method = "lm")+
  xlab("Tag duration (hours)")+
  ylab("Cue Rate (clicks per second)")+
  theme_bw()
ggfig2
```



A combined plot, figure 3 in the paper:

```
ggarrange(ggfig1, ggfig2,ncol = 2, nrow = 1)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
#this is the code to make a production ready version of figure 1
ggsave("Figures/Figure3.jpeg",dpi=600)
```

```
## Saving 6.5 x 4.5 in image
```

Cue rate analysis

Analysis with complete dataset

We check that we have the same number of tags both in the dataframe at the tag level and on the data frame at the deep dive cycle level. If so, this returns TRUE: TRUE. There are 826 deep dive cycles, over 104 tags (= individual whales). The tags durations range from 0.27 to 25.8 hours.

By tag

```
#input sample size, sample mean, and sample standard deviation
n.tags <- as.numeric(nrow(tags))
mean.tags <- mean(tags$crate,na.rm=T)
s.tags <- sd(tags$crate,na.rm=T)
mean.tags.CV=s.tags/mean.tags

#calculate margin of error
margin.tags <- qt(0.975,df=n.tags-1)*s.tags/sqrt(n.tags)

#calculate lower and upper bounds of confidence interval
low.tags <- mean.tags - margin.tags
```

```
high.tags <- mean.tags + margin.tags


#data frame with the information needed to create the plot
results<-data.frame(mean_cr=mean.tags,lci=low.tags,hci=high.tags,
        CV=mean.tags.CV, type="Average",unit="tag",label="tag: mean")
```

Mean The estimated mean for the cue rate, considering the tag as the sampling unit, is 0.947 and the median is 0.954. The maximum value for the cue rate is 1.611 and the minimum value is 0.127. The standard deviation for the cue rate is 0.285. The 95% CI for the mean is [0.892,1.003].

```
wmean.tags <- weighted.mean(tags$crate,tags$duration,na.rm=T)
wmean.tags.sd<-sqrt(var.wtd.mean.cochran(tags$crate,tags$duration))
wmean.tags.cv<-wmean.tags.sd/wmean.tags
#calculate lower and upper bounds of confidence interval
wmean.tags.margin <- qt(0.975,df=n.tags-1)*wmean.tags.sd

wmean.tags.low <- wmean.tags - wmean.tags.margin
wmean.tags.high <- wmean.tags + wmean.tags.margin

#data frame with the information needed to create the plot
results[2,]<-c(wmean.tags, wmean.tags.low,wmean.tags.high,wmean.tags.cv,
              type="Weighted average",unit="tag",label="tag: weighted mean")
```

Weighted mean The weighted mean for the cue rate for all tags is 0.911. The 95% CI for the weighted mean is [0.854, 0.968].

By deep dive cycle

```
#input sample size, sample mean, and sample standard deviation
n.DDCs <- as.numeric(nrow(DDCs))
mean.DDCs <- mean(DDCs$crate,na.rm=T)
s.DDCs <- sd(DDCs$crate,na.rm=T)
cv.mean.DDCs<-s.DDCs/mean.DDCs

#calculate margin of error
margin.DDCs <- qt(0.975,df=n.DDCs-1)*s.DDCs/sqrt(n.DDCs)

#calculate lower and upper bounds of confidence interval
low.DDCs <- mean.DDCs - margin.DDCs
high.DDCs <- mean.DDCs + margin.DDCs


#data frame with the information needed to create the plot
results[3,]<-c(mean.DDCs,low.DDCs,high.DDCs,cv.mean.DDCs,
            "Average","DDC",label="DDC: mean")
```

Mean The estimated mean for the cue rate, considering the DDCs as the sampling unit, is 0.964 and the median is 0.992. The maximum observed value for the cue rate is 1.934 and the minimum value is 0. The standard deviation for the cue rate is 0.344. The 95% CI for the mean is [0.94,0.987].

```
wmean.DDCs <- weighted.mean(DDCs$crate,DDCs$durations,na.rm=T)
wmean.DDCs.sd<-sqrt(var.wtd.mean.cochran(DDCs$crate,DDCs$durations))
wmean.DDCs.cv<-wmean.DDCs.sd/wmean.DDCs
#calculate lower and upper bounds of confidence interval
wmean.DDCs.margin <- qt(0.975,df=n.DDCs-1)*wmean.DDCs.sd

wmean.DDCs.low <- wmean.DDCs - wmean.DDCs.margin
wmean.DDCs.high <- wmean.DDCs + wmean.DDCs.margin

#data frame with the information needed to create the plot
results[4,]<-c(wmean.DDCs, wmean.DDCs.low,wmean.DDCs.high,wmean.DDCs.cv,
               type="Weighted average",unit="DDC",label="DDC: weighted mean")
```

Weighted mean The weighted mean for the cue rate for all tags is 0.911. The 95% CI for the weighted mean is [0.872, 0.949].

GEE Here we consider a regression approach to estimate cue rates, considering Generalized Estimating Equations (GEE).

Offset and Random Effect Here we implement the GEE model akin to that of Warren et al. 2017, which includes both a random effect and an offset, with an independence correlation structure.

```
#ori stands for offset + random effect + independence
gee.ori <- geeglm(formula =(nclick)~offset(log(durations)),
                  id = as.numeric(as.factor(DDCs$tag)),
                  data = DDCs,
                  family = poisson(link ="log"),
                  corstr = "independence")
```

Looking at the results of the model:

```
summary(gee.ori)
```

```
##
## Call:
## geeglm(formula = (nclick) ~ offset(log(durations)), family = poisson(link = "log"),
##     data = DDCs, id = as.numeric(as.factor(DDCs$tag)), corstr = "independence")
##
##  Coefficients:
##             Estimate  Std.err  Wald Pr(>|W|)
## (Intercept) -0.09349  0.03142 8.852  0.00293 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation structure = independence
## Estimated Scale Parameters:
##
##             Estimate Std.err
## (Intercept)    518.9    51.3
## Number of clusters:   104  Maximum cluster size: 31
```

Since the model considers a log link, we need to exponentiate the parameter to interpret it in the response (cue rate) scale.

```
# mean obtained by the model
coef.gee.ori<-exp(summary(gee.ori)$coefficients[, 1])
```

The estimated cue rate we get from the model is 0.9107.

```
#calculate lower and upper bounds of confidence interval
gee.ori.margin <- qt(0.975,df=n.DDCs-1)*summary(gee.ori)$coefficients[, 2]

gee.ori.low <- exp(summary(gee.ori)$coefficients[, 1] - gee.ori.margin)
gee.ori.high <- exp(summary(gee.ori)$coefficients[, 1] + gee.ori.margin)
```

For a 95% CI it is recommended to compute it in the link scale and then back transform the endpoints, leading to (0.856, 0.969).

```
# Adding to the results dataframe
results[5,]<-c(coef.gee.ori,gee.ori.low,gee.ori.high,NA,
type="GEE : o + r + i",unit="GEE : o + r + i",label="DDC: gee offset+re")
```

Random Effect only Here we implement the GEE model but without the offset, which is akin to the standard average at the DDC level.

```
#ri stands for random effect + independence
gee.ri <- geeglm(formula =((nclick+0.000001)/durations)~1,
                 id = as.numeric(as.factor(DDCs$tag)),
                 data = DDCs,
                 family = Gamma(link ="log"),
                 corstr = "independence")
```

Looking at the results of the model:

```
summary(gee.ri)
```

```
##
## Call:
## geeglm(formula = ((nclick + 1e-06)/durations) ~ 1, family = Gamma(link = "log"),
##     data = DDCs, id = as.numeric(as.factor(DDCs$tag)), corstr = "independence")
##
##  Coefficients:
##             Estimate Std.err Wald Pr(>|W|)
## (Intercept)  -0.0369  0.0299 1.53     0.22
##
## Correlation structure = independence
## Estimated Scale Parameters:
##
##             Estimate Std.err
## (Intercept)    0.127  0.0134
## Number of clusters:   104  Maximum cluster size: 31
```

Since the model considers a log link, we need to exponentiate the parameter to interpret it in the response (cue rate) scale.

```
# mean obtained by the model
coef.gee.ri<-exp(summary(gee.ri)$coefficients[, 1])
```

The estimated cue rate we get from the model is 0.964.

```
#calculate lower and upper bounds of confidence interval
gee.ri.margin <- qt(0.975,df=n.DDCs-1)*summary(gee.ri)$coefficients[, 2]
```

```
gee.ri.low <- exp(summary(gee.ri)$coefficients[, 1] - gee.ri.margin)
gee.ri.high <- exp(summary(gee.ri)$coefficients[, 1] + gee.ri.margin)
```

For a 95% CI it is recommended to compute it in the link scale and then back transform the endpoints, leading to (0.909, 1.022).

```
# Adding to the results dataframe
results[6,]<-c(coef.gee.ri,gee.ri.low,gee.ri.high,NA,
type="GEE : r + i",unit="GEE : r + i",label="DDC: gee re")
```

GLMM

```
glmm.ori <- lme4::glmer(nclick~offset(log(durations))+(1|tag),family=poisson(link = "log"),
          data=DDCs)
```

Offset and Random Effect Looking at the results of the model:

```
summary(glmm.ori)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula: nclick ~ offset(log(durations)) + (1 | tag)
##    Data: DDCs
##
##      AIC      BIC   logLik deviance df.resid
##   335769   335779  -167883   335765      824
##
## Scaled residuals:
##    Min     1Q Median     3Q    Max
## -94.20  -2.28   2.82  10.55 127.92
##
## Random effects:
##  Groups Name        Variance Std.Dev.
##  tag    (Intercept) 0.146    0.382
## Number of obs: 826, groups:  tag, 104
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.1136     0.0375   -3.03   0.0025 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# mean obtained by the model
coef.glmm.ori<-exp(summary(glmm.ori)$coefficients[, 1])
```

The estimated cue rate we get from the model is 0.893.

```
#calculate lower and upper bounds of confidence interval
glmm.ori.margin <- qt(0.975,df=n.DDCs-1)*summary(glmm.ori)$coefficients[, 2]

glmm.ori.low <- exp(summary(glmm.ori)$coefficients[, 1] - glmm.ori.margin)
glmm.ori.high <- exp(summary(glmm.ori)$coefficients[, 1] + glmm.ori.margin)
```

For a 95% CI it is recommended to compute it in the link scale and then back transform the endpoints,

leading to (0.829, 0.961).

```
# Adding to the results dataframe
results[7,]<-c(coef.glmm.ori,glmm.ori.low,glmm.ori.high,NA,
type="glmm : o + r + i",unit="glmm : o + r + i",label="DDC: glmm offset+re")
```

```
glmm.ri <- lme4::glmer((nclick+0.000001)/durations~(1|tag),family=Gamma(link = "log"),
          data=DDCs)
```

Random Effect only Looking at the results of the model:

```
summary(glmm.ri)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: Gamma  ( log )
## Formula: (nclick + 1e-06)/durations ~ (1 | tag)
##    Data: DDCs
##
##      AIC      BIC   logLik deviance df.resid
##     1598     1612     -796     1592      823
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.8072 -0.5571  0.0834  0.7111  2.8258
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  tag      (Intercept) 0.000    0.000
##  Residual             0.127    0.356
## Number of obs: 826, groups:  tag, 104
##
## Fixed effects:
##             Estimate Std. Error t value Pr(>|z|)
## (Intercept)  -0.0369     0.0323   -1.14     0.25
## optimizer (Nelder_Mead) convergence code: 0 (OK)
## boundary (singular) fit: see help('isSingular')
```

```
# mean obtained by the model
coef.glmm.ri<-exp(summary(glmm.ri)$coefficients[, 1])
```

The estimated cue rate we get from the model is 0.964.

```
#calculate lower and upper bounds of confidence interval
glmm.ri.margin <- qt(0.975,df=n.DDCs-1)*summary(glmm.ri)$coefficients[, 2]

glmm.ri.low <- exp(summary(glmm.ri)$coefficients[, 1] - glmm.ri.margin)
glmm.ri.high <- exp(summary(glmm.ri)$coefficients[, 1] + glmm.ri.margin)
```

For a 95% CI it is recommended to compute it in the link scale and then back transform the endpoints, leading to (0.905, 1.027).

```
# Adding to the results dataframe
results[8,]<-c(coef.glmm.ri,glmm.ri.low,glmm.ri.high,NA,
type="glmm : r + i",unit="glmm : r + i",label="DDC: glmm re")
```

Analysis with reduced dataset

We check that we have the same number of tags both in the data.frame at the tag level and on the data frame at the deep dive cycle level. If so, this returns TRUE: TRUE. There are 626 deep dive cycles, over 101 tags (= individual whales). The tags durations range from 0 to 18.9 hours.

By tag

```
#input sample size, sample mean, and sample standard deviation
n.tagsR <- as.numeric(nrow(tagsR))
mean.tagsR <- mean(tagsR$crate,na.rm=T)
s.tagsR <- sd(tagsR$crate,na.rm=T)
mean.tagsR.CV=s.tagsR/mean.tagsR

#calculate margin of error
margin.tagsR <- qt(0.975,df=n.tagsR-1)*s.tagsR/sqrt(n.tagsR)

#calculate lower and upper bounds of confidence interval
low.tagsR <- mean.tagsR - margin.tagsR
high.tagsR <- mean.tagsR + margin.tagsR


#data frame with the information needed to create the plot
resultsR<-data.frame(mean_cr=mean.tagsR,lci=low.tagsR,
        hci=high.tagsR,CV=mean.tagsR.CV, type="Average",unit="tag")
```

Mean The estimated mean for the cue rate, considering the tag as the sampling unit, is 1.07 and the median is 1.126. The maximum value for the cue rate is 2 and the minimum value is 0.088. The standard deviation for the cue rate is 0.262. The 95% CI for the mean is [1.018,1.121].

```
wmean.tagsR <- weighted.mean(tagsR$crate,tagsR$duration,na.rm=T)
wmean.tagsR.sd<-sqrt(var.wtd.mean.cochran(tagsR$crate,tagsR$duration))
wmean.tagsR.cv<-wmean.tagsR.sd/wmean.tagsR
#calculate lower and upper bounds of confidence interval
wmean.tagsR.margin <- qt(0.975,df=n.tagsR-1)*wmean.tagsR.sd

wmean.tagsR.low <- wmean.tagsR - wmean.tagsR.margin
wmean.tagsR.high <- wmean.tagsR + wmean.tagsR.margin

#data frame with the information needed to create the plot
resultsR[2,]<-c(wmean.tagsR, wmean.tagsR.low,wmean.tagsR.high,wmean.tagsR.cv,
                type="Weighted average",unit="tag")
```

Weighted mean The weighted mean for the cue rate for all tags is 1.023. The 95% CI for the weighted mean is [0.963, 1.082].

By deep dive cycle

```
#input sample size, sample mean, and sample standard deviation
n.DDCsR <- as.numeric(nrow(DDCsR))
mean.DDCsR <- mean(DDCsR$crate,na.rm=T)
s.DDCsR <- sd(DDCsR$crate,na.rm=T)
cv.mean.DDCsR<-s.DDCsR/mean.DDCsR
```

```
#calculate margin of error
margin.DDCsR <- qt(0.975,df=n.DDCsR-1)*s.DDCsR/sqrt(n.DDCsR)

#calculate lower and upper bounds of confidence interval
low.DDCsR <- mean.DDCsR - margin.DDCsR
high.DDCsR <- mean.DDCsR + margin.DDCsR


#data frame with the information needed to create the plot
resultsR[3,]<-c(mean.DDCsR,low.DDCsR,high.DDCsR,cv.mean.DDCsR,"Average","DDC")
```

Mean The estimated mean for the cue rate, considering the tag as the sampling unit, is 0.994 and the median is 1.01. The maximum value for the cue rate is 2 and the minimum value is 0. The standard deviation for the cue rate is 0.322. The 95% CI for the mean is [0.969,1.019].

```
wmean.DDCsR <- weighted.mean(DDCsR$crate,DDCsR$durations,na.rm=T)
wmean.DDCsR.sd<-sqrt(var.wtd.mean.cochran(DDCsR$crate,DDCsR$durations))
wmean.DDCsR.cv<-wmean.DDCsR.sd/wmean.DDCsR
#calculate lower and upper bounds of confidence interval
wmean.DDCsR.margin <- qt(0.975,df=n.DDCsR-1)*wmean.DDCsR.sd

wmean.DDCsR.low <- wmean.DDCsR - wmean.DDCsR.margin
wmean.DDCsR.high <- wmean.DDCsR + wmean.DDCsR.margin

#data frame with the information needed to create the plot
resultsR[4,]<-c(wmean.DDCsR, wmean.DDCsR.low,wmean.DDCsR.high,wmean.DDCsR.cv,
              type="Weighted average",unit="DDC")
```

Weighted Mean The weighted mean for the cue rate for all tags is 1.023. The 95% CI for the weighted mean is [1.001, 1.044].

GEE Here we consider a regression approach to estimate cue rates, considering Generalized Estimating Equations (GEE).

Offset and Random Effect Here we implement the GEE model akin to that of Warren et al. 2017, which includes both a random effect and an offset, with an independence correlation structure.

```
#ori stands for offset + random effect + independence
geeR.ori <- geeglm(formula =(nclick)~offset(log(durations)),
                   id = as.numeric(as.factor(DDCsR$tag)),
                   data = DDCsR,
                   family = poisson(link ="log"),
                   corstr = "independence")
```

Looking at the results of the model:

```
summary(geeR.ori)
```

```
##
## Call:
## geeglm(formula = (nclick) ~ offset(log(durations)), family = poisson(link = "log"),
##     data = DDCsR, id = as.numeric(as.factor(DDCsR$tag)), corstr = "independence")
##
##  Coefficients:
```

```
##             Estimate Std.err Wald Pr(>|W|)
## (Intercept)   0.0223  0.0291 0.59     0.44
##
## Correlation structure = independence
## Estimated Scale Parameters:
##
##             Estimate Std.err
## (Intercept)      228    21.9
## Number of clusters:   101  Maximum cluster size: 31
```

Since the model considers a log link, we need to exponentiate the parameter to interpret it in the response (cue rate) scale.

```
# mean obtained by the model
coef.geeR.ori<-exp(summary(geeR.ori)$coefficients[, 1])
```

The estimated cue rate we get from the model is 1.023.

```
#calculate lower and upper bounds of confidence interval
geeR.ori.margin <- qt(0.975,df=n.DDCsR-1)*summary(geeR.ori)$coefficients[, 2]

geeR.ori.low <- exp(summary(geeR.ori)$coefficients[, 1] - geeR.ori.margin)
geeR.ori.high <- exp(summary(geeR.ori)$coefficients[, 1] + geeR.ori.margin)
```

For a 95% CI it is recommended to compute it in the link scale and then back transform the endpoints, leading to (0.966, 1.083).

```
# Adding to the results dataframe
resultsR[5,]<-c(coef.geeR.ori,geeR.ori.low,geeR.ori.high,NA,
type="GEE : o + r + i",unit="GEE : o + r + i")
```

Random Effect only Here we implement the GEE model but without the offset, which is akin to the standard average at the DDC level.

```
#ri stands forrandom effect + independence
geeR.ri <- geeglm(formula =((nclick+0.000001)/durations)~1,
                  id = as.numeric(as.factor(DDCsR$tag)),
                  data = DDCsR,
                  family = Gamma(link ="log"),
                  corstr = "independence")
```

Looking at the results of the model:

```
summary(geeR.ri)
```

```
##
## Call:
## geeglm(formula = ((nclick + 1e-06)/durations) ~ 1, family = Gamma(link = "log"),
##     data = DDCsR, id = as.numeric(as.factor(DDCsR$tag)), corstr = "independence")
##
##  Coefficients:
##             Estimate  Std.err Wald Pr(>|W|)
## (Intercept) -0.00587  0.03423 0.03     0.86
##
## Correlation structure = independence
## Estimated Scale Parameters:
##
##             Estimate Std.err
```

```
## (Intercept)    0.105  0.0152
## Number of clusters:   101  Maximum cluster size: 31
```

Since the model considers a log link, we need to exponentiate the parameter to interpret it in the response (cue rate) scale.

```
# mean obtained by the model
coef.geeR.ri<-exp(summary(geeR.ri)$coefficients[, 1])
```

The estimated cue rate we get from the model is 0.994.

```
#calculate lower and upper bounds of confidence interval
geeR.ri.margin <- qt(0.975,df=n.DDCs-1)*summary(geeR.ri)$coefficients[, 2]

geeR.ri.low <- exp(summary(geeR.ri)$coefficients[, 1] - geeR.ri.margin)
geeR.ri.high <- exp(summary(geeR.ri)$coefficients[, 1] + geeR.ri.margin)
```

For a 95% CI it is recommended to compute it in the link scale and then back transform the endpoints, leading to (0.93, 1.063).

```
# Adding to the results dataframe
resultsR[6,]<-c(coef.geeR.ri,geeR.ri.low,geeR.ri.high,NA,
type="GEE : r + i",unit="GEE : r + i",label="DDC: gee re")
```

GLMM

```
glmmR.ori<- lme4::glmer(nclick~offset(log(durations))+(1|tag),
            family=poisson(link = "log"), data=DDCsR)
```

Offset and Random Effect Looking at the results of the model:

```
summary(glmmR.ori)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula: nclick ~ offset(log(durations)) + (1 | tag)
##    Data: DDCsR
##
##      AIC      BIC   logLik deviance df.resid
##    90849    90858   -45423    90845      624
##
## Scaled residuals:
##    Min     1Q Median     3Q    Max
## -54.02  -3.27   0.05   4.20  42.85
##
## Random effects:
##  Groups Name        Variance Std.Dev.
##  tag    (Intercept) 0.08     0.283
## Number of obs: 626, groups:  tag, 101
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.0341     0.0282    1.21     0.23
```

```
newdata<-data.frame(durations=1,nclick=DDCsR$nclick,duration=DDCsR$durations,
        tag=DDCsR$tag,location=DDCsR$location, year=DDCsR$year)

newdata$predictionss<-predict(glmmR.ori,newdata=newdata,type="response")

newdata$tpredictionss<-newdata$predictionss

coef5<-mean(newdata$tpredictionss)

#input sample size, sample mean, and sample standard deviation
nglmmR <- as.numeric(nrow(newdata))
xbarglmmR <- mean(newdata$tpredictionss,na.rm=T)
sglmmR <- sd(newdata$tpredictionss,na.rm=T)

#calculate margin of error
marginglmmR <- qt(0.975,df=nglmmR-1)*sglmmR/sqrt(nglmmR)

#calculate lower and upper bounds of confidence interval
minintglmmR <- xbarglmmR - marginglmmR
maxintglmmR <- xbarglmmR + marginglmmR
```

The estimated cue rate we get from the model is 0.999

```
# mean obtained by the model
coef.glmmR.ori<-exp(summary(glmmR.ori)$coefficients[, 1])
```

The estimated cue rate we get from the model is 1.035.

```
#calculate lower and upper bounds of confidence interval
glmmR.ori.margin <- qt(0.975,df=n.DDCs-1)*summary(glmmR.ori)$coefficients[, 2]

glmmR.ori.low <- exp(summary(glmmR.ori)$coefficients[, 1] - glmmR.ori.margin)
glmmR.ori.high <- exp(summary(glmmR.ori)$coefficients[, 1] + glmmR.ori.margin)
```

For a 95% CI it is recommended to compute it in the link scale and then back transform the endpoints, leading to (0.979, 1.094).

```
# Adding to the results dataframe
resultsR[7,]<-c(coef.glmmR.ori,glmmR.ori.low,glmmR.ori.high,NA,
type="glmmR : o + r + i",unit="glmmR : o + r + i",label="DDC: glmmR offset+re")
```

```
glmmR.ri<- lme4::glmer((nclick+0.000001)/durations~(1|tag),family=Gamma(link = "log"),
            data=DDCsR)
```

Random Effect only Looking at the results of the model:

```
summary(glmmR.ri)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: Gamma  ( log )
## Formula: (nclick + 1e-06)/durations ~ (1 | tag)
##    Data: DDCsR
##
##      AIC      BIC   logLik deviance df.resid
##     1246     1259     -620     1240      623
```

```
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.1035 -0.5694  0.0476  0.6380  2.9100
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  tag      (Intercept) 0.000262 0.0162
##  Residual             0.103825 0.3222
## Number of obs: 626, groups:  tag, 101
##
## Fixed effects:
##             Estimate Std. Error t value Pr(>|z|)
## (Intercept) -0.00476    0.03721   -0.13      0.9
```

```
# mean obtained by the model
coef.glmmR.ri<-exp(summary(glmmR.ri)$coefficients[, 1])
```

The estimated cue rate we get from the model is 0.995.

```
#calculate lower and upper bounds of confidence interval
glmmR.ri.margin <- qt(0.975,df=n.DDCsR-1)*summary(glmmR.ri)$coefficients[, 2]

glmmR.ri.low <- exp(summary(glmmR.ri)$coefficients[, 1] - glmmR.ri.margin)
glmmR.ri.high <- exp(summary(glmmR.ri)$coefficients[, 1] + glmmR.ri.margin)
```

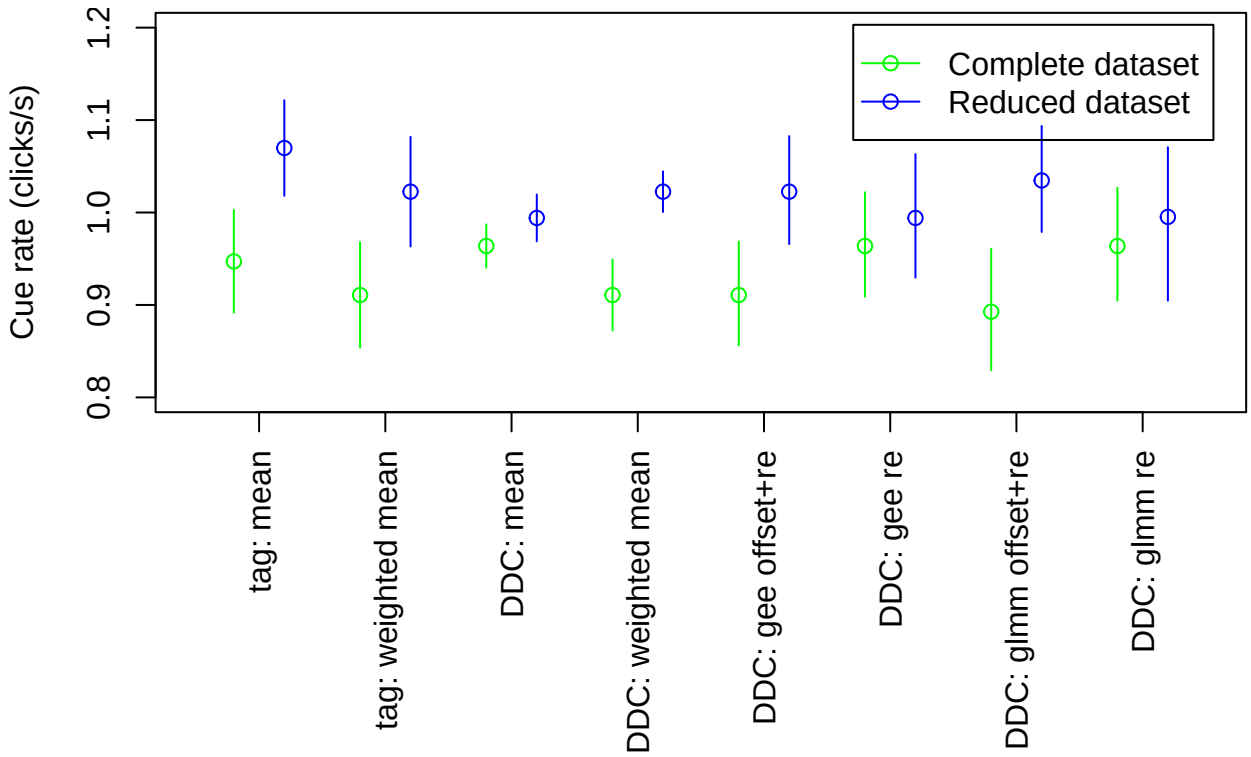
For a 95% CI it is recommended to compute it in the link scale and then back transform the endpoints, leading to (0.925, 1.071).

```
# Adding to the results dataframe
resultsR[8,]<-c(coef.glmmR.ri,glmm.ri.low,glmmR.ri.high,NA,
type="glmmR : r + i",unit="glmmR : r + i",label="DDC: glmmR re")
```

Figure 4: comparison of methods to estimate the cue rate

Plotting the results

```
#n.s=number of analysis to show on plot
n.s <- 8
par(mar=c(12,4,0.1,0.1))
with(results[1:n.s,],plot((1:n.s)-0.2,mean_cr,ylim=c(0.8,1.2),xlim=c(0.5,n.s+0.5),
  col="green",xaxt="n",ylab="Cue rate (clicks/s)",xlab=""))
axis(side=1,at=1:n.s,labels = results$label[1:n.s],las=2)
with(results[1:n.s,],segments(x0=(1:n.s)-0.2,y0=lci,x1=(1:n.s)-0.2,y1=hci,col="green"))
with(resultsR[1:n.s,],points((1:n.s)+0.2,mean_cr,col="blue"))
with(resultsR[1:n.s,],segments(x0=(1:n.s)+0.2,y0=lci,x1=(1:n.s)+0.2,y1=hci,col="blue"))
legend("topright",inset=0.03,legend=c("Complete dataset","Reduced dataset"),
  col=c("green","blue"),lty=1,pch=1)
```

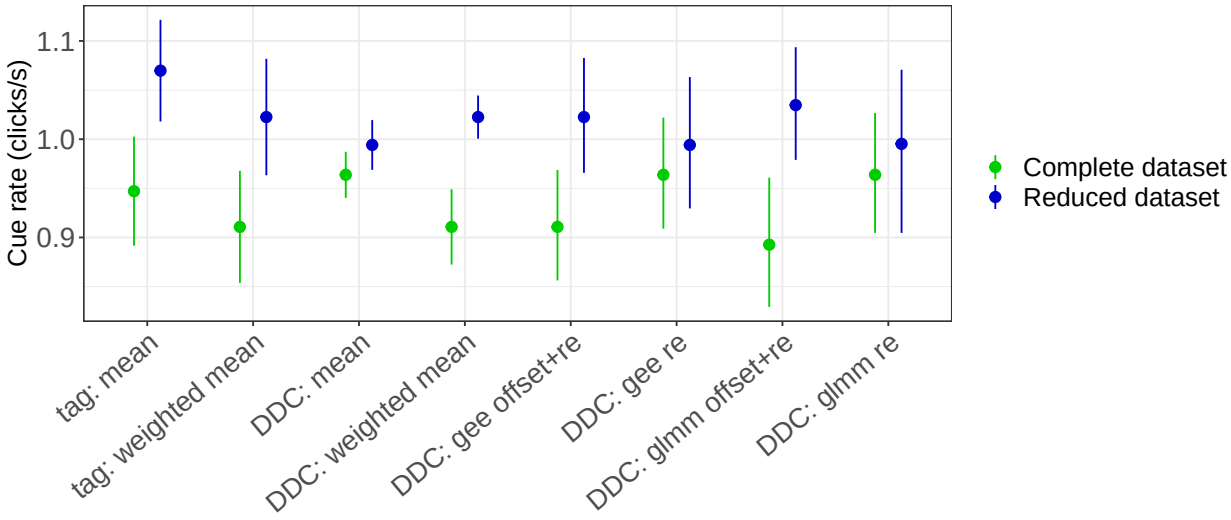


Joining info:

```
resultsR$label<-results$label
results$data<-"Complete dataset"
resultsR$data<-"Reduced dataset"
all_results<-full_join(results,resultsR)
```

```
## Joining, by = c("mean_cr", "lci", "hci", "CV", "type", "unit", "label", "data")
```

```
all_results$label<-factor(all_results$label,levels=c("tag: mean","tag: weighted mean",
"DDC: mean","DDC: weighted mean" ,"DDC: gee offset+re",
"DDC: gee re","DDC: glmm offset+re", "DDC: glmm re"  ))
ggplot(all_results,aes(x=label,y=mean_cr,color=data))+
  geom_pointrange(aes(ymin=lci,ymax=hci,color=data),
  position = position_dodge(width = 0.5), size=0.5,)+
 # geom_point()+
  xlab("")+theme_bw()+ylab("Cue rate (clicks/s)")+
  theme(axis.text.x=element_text(angle=40, hjust=1))+labs(color="")+
  scale_color_manual(values=c("green3","blue3"))+
  theme(
        axis.title.x = element_text(size = 15),
        axis.text.x = element_text(size = 16),
        axis.text.y = element_text(size = 16),
        axis.title.y = element_text(size = 15),
        legend.title = element_text(size=15),
        legend.text= element_text(size=15),
        plot.title = element_text(size=16))
```




```
ggsave("Figures/Figure4.jpeg",dpi=600)
```

```
## Saving 10 x 4.5 in image
```