Code for producing results and figures of "Estimating sperm whale cue rates to inform passive acoustic density estimation via cue counting"

Marques, T. A.

10 August 2023

Contents

Introduction

This document presents all the analysis that we've investigated while trying to estimate a cue rate, and in particular its precision, for a new location and year one might not have in the data. This ended up being a major endeavour, with lots of dead ends and open questions related to Gamma GLMMs and their implementation in `lme4::glmer` and in a Bayesian context.

This will also be a precursor, via considerable simplification, to the code required to reproduce the figures and results in the manuscript "Estimating sperm whale cue rates to inform passive acoustic density estimation via cue counting", submitted to The Journal of the Acoustical Society of America, by **add all names here**. In said paper we report on estimates of cue rates for sperm whales at a selection of time and year combinations that were sampled, but the main interest is in estimating a mean cue rate, and corresponding precision, for a time and location that might not have been observed.

These analysis and data to make it fully reproducible are hosted at https://github.com/TiagoAMarques/Report4BB

About the data

The dataset lies within `ddata1`, the single object in `data_4_article_clickrates_deep_dive.rda` that gets frontloaded below. This data was 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. This is combined with the information for the duration of the tag records to obtain cue production rates, number of sounds per animal per unit time. For future reference, the creation of the objects considered here was done in an RMarkdown dynamic report entitled `05_Cue_Rates_For_Sperm_Whales_per_DDC.Rmd`. This document is shared as part of a separate data paper, where the times of echolocation clicks in this unique DTAG dataset, along with the DTAGs depth profiles, are shared to be used by others.

The data consists of summaries of numbers of regular echolocation clicks per deep dive cycle, for each of the sperm whale tags considered on the manuscript. Note that despite having been recorded at the deep dive cycle level, to be used 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., the first step in the data pre-processing done below is to pool data for each tag record, as the tag is the fundamental (and more importantly independent) sampling unit.

Reading the data

We begin by reading the deep dive cycle data in:

```r
# file 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")
```

Data pre-processing

We actually have data from whales which have been subjected to SONAR exposure under controlled exposure experiments. We discard the data from those tags here, since evaluating the effect of SONAR is the focus of other research programs. Understanding the effects of sonar exposure on whale behaviour, and in particular cue rate production, is a separate research thread which requires information to be analysed at a much finer resolution, and for which context would have to be considered. For the majority of the tags the context information is unavailable to us within the ACCURATE project.

```
#removing the tags for animals we know were exposed to sonar
DDCs<-ddata1[ddata1$sonar!="sonar",]
```

Since we will treat tags, not deep dive cycles, as the independent sampling units, we aggregate the deep dive cycle data for each tag into a single tag record (object `tags`)

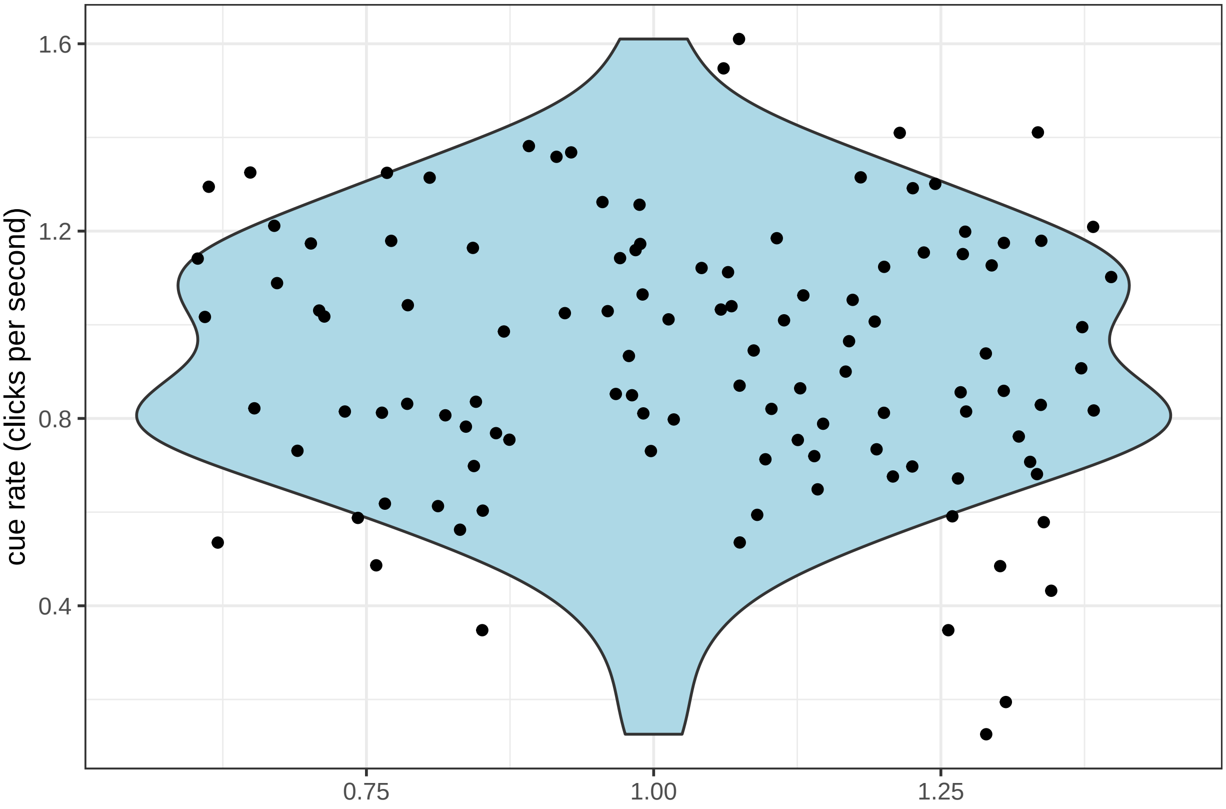
```
# Creating the data per tag
tags <- DDCs %>%
  group_by(tag) %>%
  summarise(location=unique(location), year=unique(year), sex=unique(sex),
  duration= sum(durations,na.rm=T),nclicks=sum(nclick,na.rm=T),
  crate=sum(nclick,na.rm=T)/sum(durations,na.rm=T),ddc=max(absdives+1,na.rm = T))
```

Exploratory data analysis

We have a total of 113 whales tagged which were not, knowingly, exposed to sonar.

Tag recording duration ranged from 0.144 to 25.781 hours. The observed cue rates per tag varied between 0.126 and 1.61, with a mean value of 0.93 and median value of 0.933 clicks per second. The individual cue rates per tag record, pooled across years and locations, are shown in the following figure:

```
ggplot(tags,aes(x=1,y=crate),fill="lightblue")+
  theme_bw()+geom_violin(fill="lightblue")+
  geom_jitter()+ylab("cue rate (clicks per second)")+xlab("")
```



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

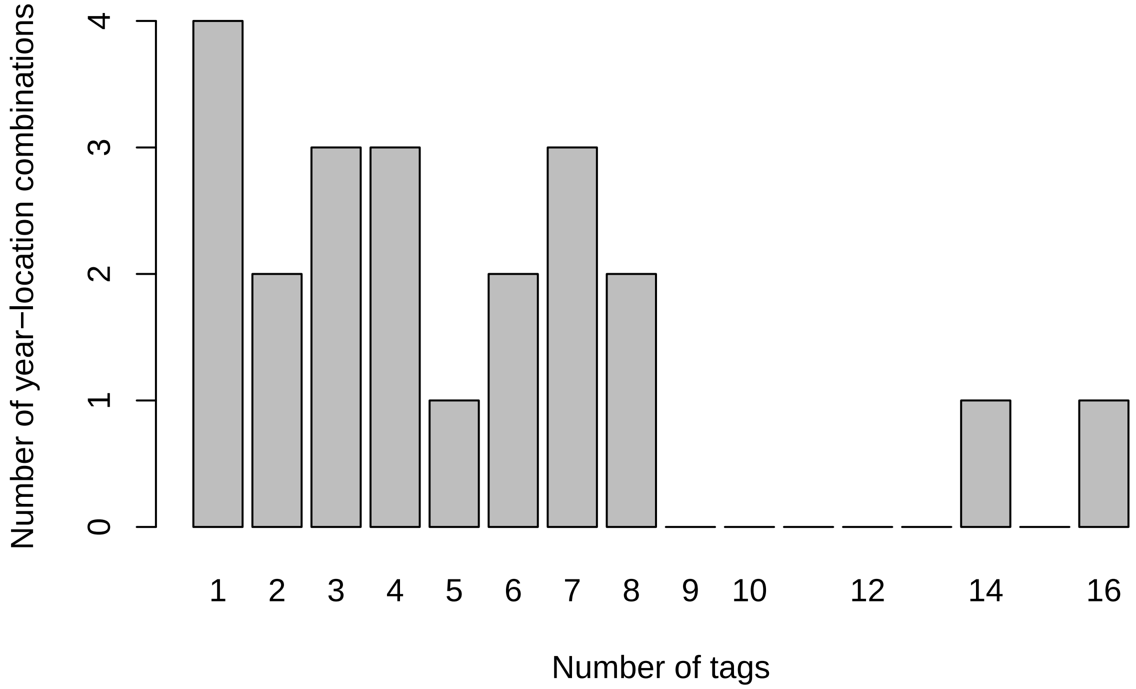
```
kable(table(tags$location,tags$year))
```

	2001	2002	2003	2005	2009	2010	2013	2014	2015	2016	2017	2018	2019	2020	2021
Azores	0	0	0	0	0	8	0	0	0	0	0	3	5	6	3
DOMINICA	0	0	0	0	0	0	0	7	16	4	1	4	0	0	0
Gulf of Mexico	4	14	8	0	0	0	0	0	0	0	0	0	0	0	0
Kaikoura	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0
Mediterranean	1	0	7	0	0	0	0	0	0	0	0	0	0	0	0
North Atlantic Delaware	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0
Norway	0	0	0	0	1	3	0	0	0	2	0	0	2	0	0
Norway Andenes	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

The number of tags per year-location combination varies considerably. Out of a total of 8 different locations and 15 different years, leading therefore to 120 possible year-location combinations, tags are not available for the majority of these possible combinations (98 combinations), with only 22 combinations having any tags associated with.

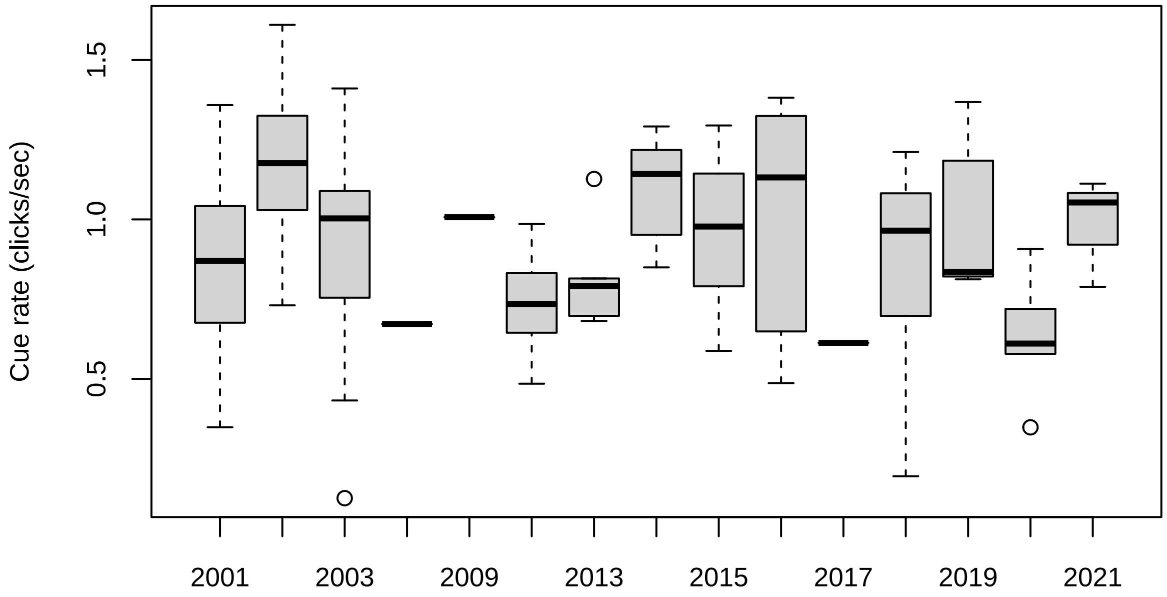
What might be the number of tag records required to obtain a reliable year-location cue rate estimate remains hard to evaluate, but several year-location combinations are certainly below that minimum, namely for those with less than a handful of tags. The distribution of the number of tags per year-location combination for which we have tags is represented in the image below:

```
#note the need to remove the first count
#corresponding to un-interesting year-location combinations
#with 0 tags
counts<-table(tags$location,tags$year)
maxtags<-max(table(tags$location,tags$year))
counts2<-numeric(maxtags)
for (i in 1:maxtags){
  counts2[i]<-sum(counts==i)
}
barplot(counts2,names.arg=1:maxtags,ylab="Number of year-location combinations",xlab="Number of tags")
```



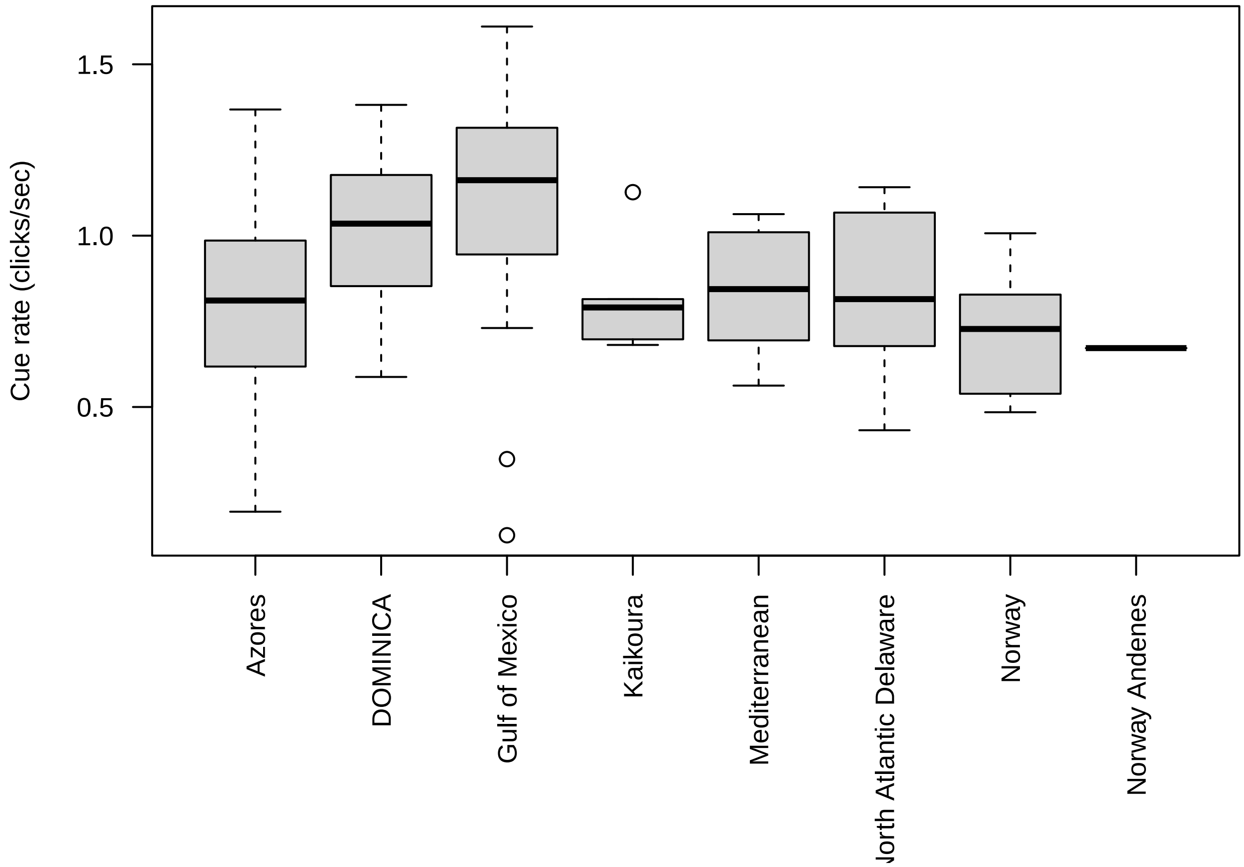
We can take a look at the cue rates (pooled across locations) per year

```
with(tags,boxplot(crate~year,ylab="Cue rate (clicks/sec)",xlab="",cex.lab=0.8,cex.axis=0.8))
```



and those (pooled across years) per location

```
par(mar=c(8,4,0.2,0.2))
with(tags,boxplot(crate~location,las=2,ylab="Cue rate (clicks/sec)",xlab="",cex.lab=0.8,cex.axis=0.8))
```



We are hoping to explain variability in cue rates as a function of year and location. To do so, year-location combinations with a small number of tags are difficult to deal with. In particular, for those with a single tag, a year-location effect would be hard to estimate: with a model with an interaction, the effect would be strictly unidentifiable. We therefore removed years and locations for which only a single tag existed, effectively removing the single tag from Norway Andenes (which was also the single tag from 2005). Additionally, we pooled tags from single-tag year-location combinations into an adjacent year for the same same location. Hence, we pooled

- the single tag from Norway in 2009 with the 3 Norway tags from 2010;
- the single tag from Dominica in 2017 with the remaining 4 Dominica tags from 2016;
- the single tag from the Mediterranean in 2001 with the 7 tags from Mediterranean from 2003.

```
#removing the location with a single tag
tags<-tags[tags$location!="Norway Andenes",]
#grouping single tag per year into adjacent years
tags$year[tags$location=="Norway" & tags$year==2009]<- 2010
tags$year[tags$location=="DOMINICA" & tags$year==2017]<- 2016
tags$year[tags$location=="Mediterranean" & tags$year==2001]<- 2003
```

After this assignment, we create a couple of factor covariates inside `tags`:

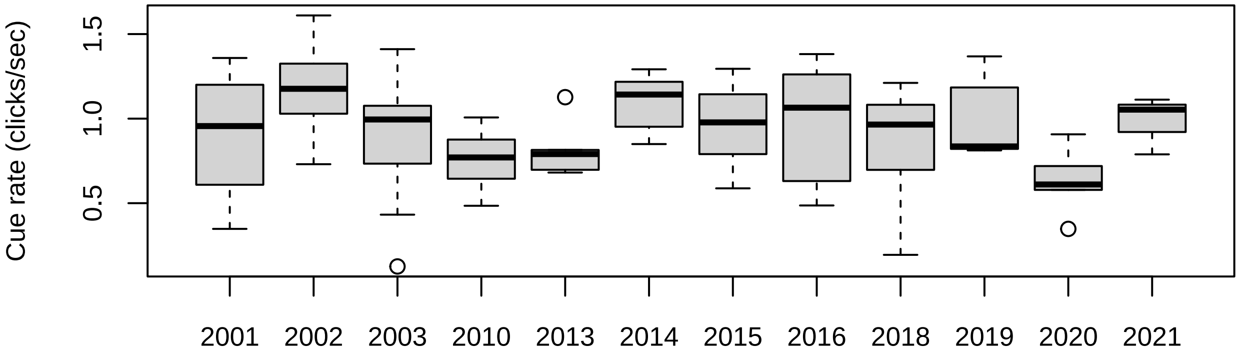
- fyear, representing year as a factor
- locyear, representing each year-location as a factor

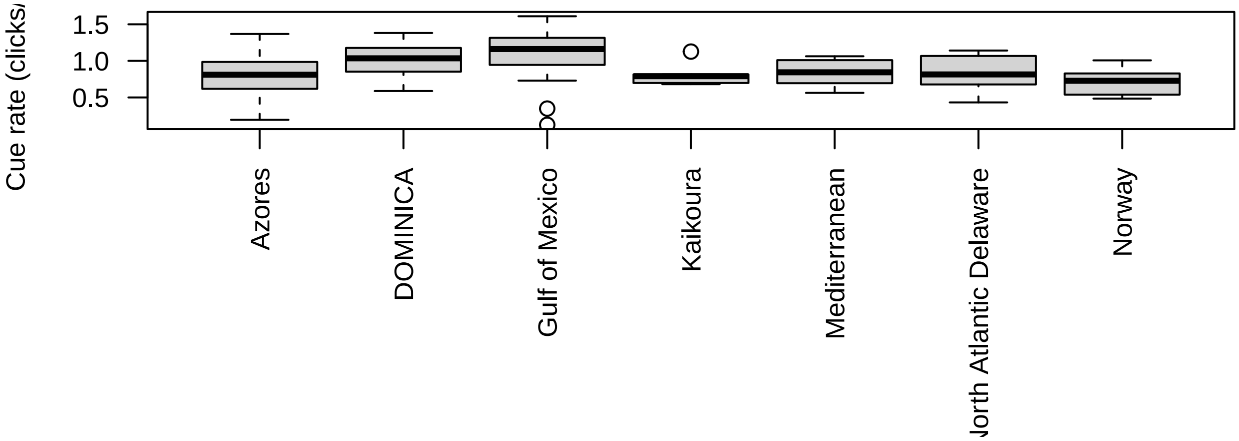
```
#building relevant factor covariates given this new assignment
#making a new variable, year as factor
tags$fyear<-as.factor(tags$year)
#adding a variable that represents each location-year combination
```

```r
tags$locyear<-as.factor(paste0(tags$location,tags$fyear))
```

Given this data pooling, the above plots become:

```r
par(mfrow=c(2,1),mar=c(4,4,0.2,0.2))
with(tags,boxplot(crate~year,ylab="Cue rate (clicks/sec)",xlab="",cex.lab=0.8,cex.axis=0.8))
par(mar=c(8,4,0.2,0.2))
with(tags,boxplot(crate~location,las=2,ylab="Cue rate (clicks/sec)",xlab="",cex.lab=0.8,cex.axis=0.8))
```





Estimating cue rates

Our objective is to obtain a cue rate estimate, and its desired precision to include in a cue counting density estimator. We assume cue rate might depend on a number of covariates, for which location and year might act as relevant proxies.

Estimation strategies

Ignoring what required predictions might be to begin with, from first principles, location might be a sensible fixed effects covariate, since different locations will present different depths and prey distributions, and hence foraging at different depths might occur, and consequently different cue rates per location (across years). On the other hand, it seems like the variability from year might be not driven by year itself, but as random fluctuations over time, perhaps more sensibly accounted for as factor (or a random effect). If one believes this to be the case, to predict cue rates for:

1. a sampled location and a new year, one could consider to use a model with location as a fixed effect, propagating the variability of year as a random effect;

2. for a new location and year, one could would use a model with both location and year as random effects.

In other words, intuitively year seems more sensibly modeled as a random effect, while location could be modelled either as as a fixed effect or as a random effect.

We can distinguish different levels of difficulty in terms of cue rate estimation with regards to the available information to do so. From easiest to hardest, we might want to estimate a cue rate for:

- a year-location combination we have data for;
- a location we have data for at a year we do not have data for;
- a year we have data for at a location we do not have data for;
- a completely new year-location combination.

From a conceptual point of view, if we had enough data across years and locations, one might want to:

1. treat as fixed effects those covariates for which we observed the level for which predictions are desired, but
2. treat as random effects those covariates for which the level at which we would like to predict were not observed

When considering a model with random effect(s) care must be had to propagate into predictions the variability associated with predicting for a new, previously unobserved, level of the random effect. As will be described below, this is not necessarily straightforward, and different alternatives are available to do so.

Note we will not be able to separate the effect of year and location for Kaikoura, since this location was only sampled in 2013, and no other location was sampled in that year. Additionally, we cannot separate effects for years 2020 and 2021 as we only have tags from a single site (the Azores) in those years.

Model fitting

We will assume that cue rates, a strictly positive quantity, follow a Gamma distribution, and a log-link function will be considered within a generalized linear model (GLM) or generalized linear mixed model (GLMM) to ensure positive predictions.

GLMs

We begin by looking at GLM models where both location and year are treated as fixed effects, first with year as a numerical covariate, including or not interaction terms between year and location and then the same couple of analysis but with year as a factor. The latter seems more sensible a priori, as noted above.

```r
#run models
CRglm1<-glm(crate~location+year,data=tags,family=Gamma(link="log"))
#summary(CRglm1)
CRglm2<-glm(crate~location+year+year:location,data=tags,family=Gamma(link="log"))
#summary(CRglm2)
CRglm3<-glm(crate~location+fyear,data=tags,family=Gamma(link="log"))
#summary(CRglm3)
CRglm4<-glm(crate~location+fyear+fyear:location,data=tags,family=Gamma(link="log"))
# summary(CRglm4)
# note this is the same model as CRglm4, just specified in a different way
# and which incidentally gets us a nicer output since the CRglm4
# is full of NA's for all the combinations of factors which do not exist
CRglm5<-glm(crate~locyear,data=tags,family=Gamma(link="log"))
#summary(CRglm5)
```

```r
kable(AIC(CRglm1,CRglm2,CRglm3,CRglm4,CRglm5))
```

	df	AIC
CRglm1	9	51.91597
CRglm2	12	57.84711
CRglm3	17	51.20435
CRglm4	19	54.76608
CRglm5	19	54.76608

According to AIC the best model considers both location and year as factors, while the interaction between these factors is not deemed relevant. Given the nature of year and expectations as noted above for temporal effects, using year as a factor seems indeed more sensible any way. We can look at the summary of said model

```r
summary(CRglm3)
```

```
##
## Call:
## glm(formula = crate ~ location + fyear, family = Gamma(link = "log"),
##     data = tags)
##
## Coefficients: (2 not defined because of singularities)
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      -0.01542    0.16082  -0.096   0.9238
## locationDOMINICA                  0.47196    0.16726   2.822   0.0058 **
## locationGulf of Mexico           -0.08485    0.21274  -0.399   0.6909
## locationKaikoura                 -0.18692    0.19696  -0.949   0.3450
## locationMediterranean            -0.27127    0.25427  -1.067   0.2887
## locationNorth Atlantic Delaware -0.27183    0.25698  -1.058   0.2928
## locationNorway                   -0.15665    0.12616  -1.242   0.2174
## fyear2002                         0.26997    0.15792   1.710   0.0906 .
## fyear2003                         0.11260    0.17057   0.660   0.5108
## fyear2010                        -0.20497    0.18465  -1.110   0.2698
## fyear2013                              NA         NA      NA       NA
## fyear2014                        -0.37130    0.25480  -1.457   0.1483
## fyear2015                        -0.49199    0.24225  -2.031   0.0450 *
## fyear2016                        -0.35990    0.23552  -1.528   0.1298
## fyear2018                        -0.44251    0.21466  -2.061   0.0420 *
## fyear2019                         0.05834    0.19556   0.298   0.7661
## fyear2020                        -0.44814    0.19696  -2.275   0.0251 *
## fyear2021                              NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Gamma family taken to be 0.07758609)
##
##     Null deviance: 13.5399  on 111  degrees of freedom
## Residual deviance:  9.7856  on  96  degrees of freedom
## AIC: 51.204
##
## Number of Fisher Scoring iterations: 5
```

We illustrate how to estimate the cue rate, and its precision, for all year location combinations for which we have more than 3 tags. The choice of 3 is arbitrary, but we considered that 3 or less tags would be unreliable. Can we perhaps look into it later?

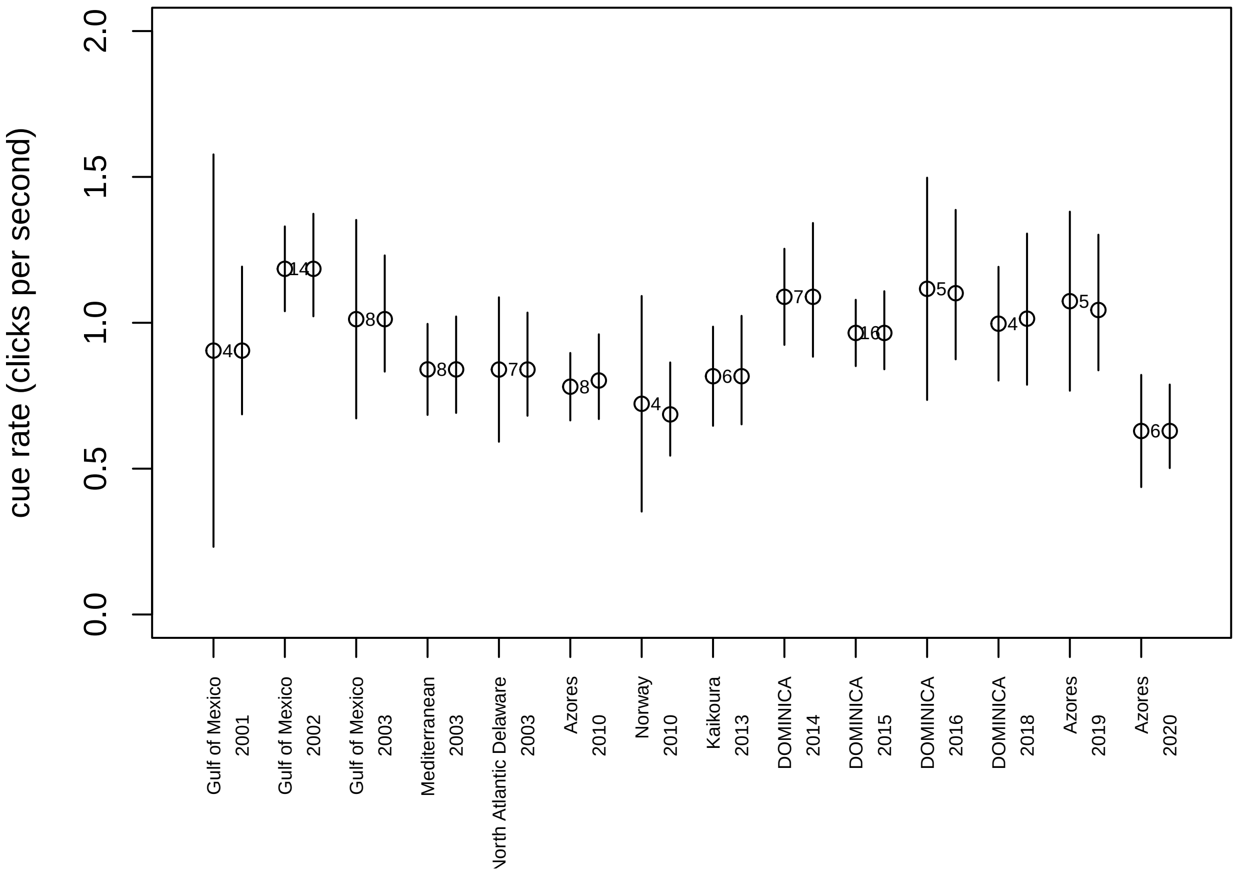
If we consider a GLM model with both year and location as fixed effects, the point estimates and respective

95% confidence intervals are easiest to obtain by recoding the year and location factors to have the desired levels of each factor as the intercept, and then exponentiating back to the linear scale. For confidence intervals, we create these on the link scale and then transform back the interval limits to the response scale.

```r
# create all possible unique location-year combinations
all.comb<-expand.grid(location=sort(unique(tags$location)),year=sort(unique(tags$year)))
# define minimum number of tags required to produce an estimate
nmin<-3
# select only those
index.min<-which(table(tags$location,tags$year)>nmin)
# create an object to hold results by location-year
# all.comb was created way above!
byly <- all.comb[index.min,]
# select only those year-location combinations for which we have tags
index.min<-which(table(tags$location,tags$year)!=0)
# create an object to hold all possible location-year's independent of the number of tags (1 is enough
byly.all <- all.comb[index.min,]




#now, for each combination
for(i in 1:nrow(byly)){
  index <- tags$location==byly$location[i] & tags$year==byly$year[i]
  #------------------------------------------------------------------
  #get the empirical average cue rate estimate
  #------------------------------------------------------------------
  #get the actual number of tags
  byly$ntags[i] <- sum(index)
  #calculate empirical cue rate
  byly$ecr[i] <- mean(tags$crate[index])
  #calculate empirical standard deviation
  byly$ecrsd[i] <- sd(tags$crate[index])
  #and the margin for a confidence interval
  byly$margin.ecr[i] <- qt(0.975,byly$ntags[i]-1)*byly$ecrsd[i]/sqrt(byly$ntags[i])
  #------------------------------------------------------------------
  #get the glm estimate
  #------------------------------------------------------------------
  # recode to get the current baseline
  # define the current year as baseline
  iyear <- which(byly$year==byly$year[i])
  #new levels
  levelsy<-c(byly$year[i],unique(byly.all$year[byly.all$year!=byly$year[i]]))
  #recode levels for year
  tags$fyear<-factor(tags$fyear,levels=levelsy)
  # define the current location as baseline
  ilocation <- which(byly$location==byly$location[i])
  #new levels
  levelsl<-c(byly$location[i],unique(byly.all$location[byly.all$location!=byly$location[i]]))
  tags$location<-factor(tags$location,levels=levelsl)
  #fit the model for the current year and location as baseline
  CRglm4yl<-glm(crate~location+fyear,data=tags,family=Gamma(link="log"))
  # get the cue rate
  byly$glm.cr[i] <- exp(summary(CRglm4yl)$coefficients[1,1])
  byly$glm.lci.cr[i] <- exp(summary(CRglm4yl)$coefficients[1,1]-qt(0.975,summary(CRglm4yl)$df.residual)
  byly$glm.uci.cr[i] <- exp(summary(CRglm4yl)$coefficients[1,1]+qt(0.975,summary(CRglm4yl)$df.residual)
}
# get the empirical lower and upper 95% CI
byly$lcl.ecr <- with(byly,ecr - margin.ecr)
byly$ucl.ecr <- with(byly,ecr + margin.ecr)
```

We compare below standard averages (left) against values obtained from the glm (right) for year-site combinations with more than 3 tags in terms of both point estimates (black points) and confidence intervals (solid vertical black lines). In between each such pair of estimates we added the number of tags available for each year-location combination.

```
#standard average
par(mfrow=c(1,1),mar=c(6,4,0.1,0.4))
plot(x=(1:nrow(byly))-0.2,y=byly$ecr,xaxt="n",ylim=c(0,2),xlim=c(0.5,nrow(byly)+0.5),xlab="",ylab="cue
with(byly,segments(x0=(1:nrow(byly))-0.2,x1=(1:nrow(byly))-0.2,y0=lcl.ecr,y1=ucl.ecr))
#GLM
points(x=(1:nrow(byly))+0.2,y=byly$glm.cr)
with(byly,segments(x0=(1:nrow(byly))+0.2,x1=(1:nrow(byly))+0.2,y0=glm.lci.cr,y1=glm.uci.cr))
#draw axis and annotations
axis(1, at=(1:nrow(byly))-0.2,byly$location,cex.axis=0.6,las=2)
axis(1, at=(1:nrow(byly))+0.2,byly$year,tick=FALSE,cex.axis=0.6,las=2,line=1)
text(x=(1:nrow(byly)),y=byly$ecr,labels=byly$ntags,cex=0.6)
```



These estimates illustrate that point estimates are not that different from each other using either means of the GLM, with considerable overlap in confidence intervals across year site combinations. In terms of cue rtaes, most location-year combinations are relatively consistent, with notable exceptions for the GoM in 2002, with higher estimated cue rate, and the Azores in 2020, with a considerably lower estimated cue rate.

Interestingly, the estimates from the GLM are similar or even slightly less precise then those from standard means, with notable exceptions being the most variable year-location combinations, where the `glm` returned narrower confidence intervals, namely for the GoM in 2001, Norway in 2010 and Dominica in 2016, where the strength borrowed from all the tags analysis means the model-based estimate is considerably more precise.

The GLM vs. empirical average comparison also illustrates how a naive cue rate confidence interval drawn from a reduced number of tags could result in inadmissible estimates for the latter (the naive 95% CI for the GoM in 2001 approaches 0, while negative values for the cue rate are not possible). From that perspective, estimates from the fitted model might be better, as these will avoid negative values by construction, induced by the log link.

GLMMs

Location as a fixed effect, year as random effect Attempting a model with location as a fixed effect, but with year as a random effect, to predict cue rate for a location we have data from, in a year we do not have data at that location. Therefore we consider a Gamma log-link GLMM for cue rate. Let us consider we are predicting the cue rate for Dominica and the GoM in the year 2012. Note the corresponding estimate would be the same in any non-observed year, it's a generic estimate for any non-observed level of the (year) random effect.

```
#define DOMINICA as baseline
tags$location<-factor(tags$location,levels=c("DOMINICA","Gulf of Mexico","Azores","Kaikoura","Mediterran
#run model
crglmerDom<-glmer(crate~location+(1|fyear),data=tags,family=Gamma(link="log"))
#define GoM as baseline
tags$location<-factor(tags$location,levels=c("Gulf of Mexico","DOMINICA","Azores","Kaikoura","Mediterran
# run model
crglmerGoM<-glmer(crate~location+(1|fyear),data=tags,family=Gamma(link="log"))
```

The empirical mean cue rate across all years for the GoM is 1.09 clicks per second and for Dominica is 1.02 clicks per second. On the other hand, considering the GLMM, those values are 1.05 clicks per second and 1.06 clicks per second for the GoM and Dominica, respectively. So while the observed cue rate was larger in the GoM, it is estimated by the GLMM as being (ever if ever just) higher in Dominica. This suggests that most of the variability that the model with two fixed effects (location and year) was attributing to changes per location are accounted for by random fluctuations in variability per year in the GLMM, and the random effect just happened to be higher for years in which the GoM was sampled (in particular 2002).

Note that we do not present here precision measures for the estimated cue rate in 2012. Obtaining the correct precision on the estimates for unobserved levels of the (year) random effect is not straightforward. We do so for a model with both year and location as random effects below.

Location and year as random effects The above results seem to suggest that it might be quite difficult to separate location and year effects.

A precautionary approach, when predicting a cue rate for new year and location combinations, might be to use both year and location as random effects in a random effects model and then propagate the uncertainty associated with the overall mean with respect to both random effects, year and location, that would be unobserved.

We can think of a few different ways to conceptualize the random effects. These include:

1. Independent random effects for year and location
2. A single random effect associated with each location-year combination
3. A random effect associated with location and a nested-within-location random effect associated with year. This means that effect of location is shared across years, but the effect of year is not shared across locations

This is implemented here:

```
# a model that considers each of year and location to be fully crossed and
#independent random effects
crglmer<-glmer(crate~(1|location)+(1|fyear),data=tags,family=Gamma(link="log"))
# a Pooled model not separating the effect of year and location
# so each year location combination is its own random effect level
# equates to saying there's no site effect not year effect
crglmerP<-glmer(crate~(1|locyear),data=tags,family=Gamma(link="log"))
# Random effect of year nested in location
crglmerN<-glmer(crate~(1|location/fyear),data=tags,family=Gamma(link="log"))
```

Interestingly, AIC - which might not be the best way to choose from different random effect stuctures - seems to favour the model where each year-site combination is a different level of a single year-site random effect.

```
AIC(crglmer,crglmerP,crglmerN)
```

```
##          df      AIC
## crglmer   4 50.48403
## crglmerP  3 48.86123
## crglmerN  4 49.70446
```

A summary of the model is

```
summary(crglmer)
```

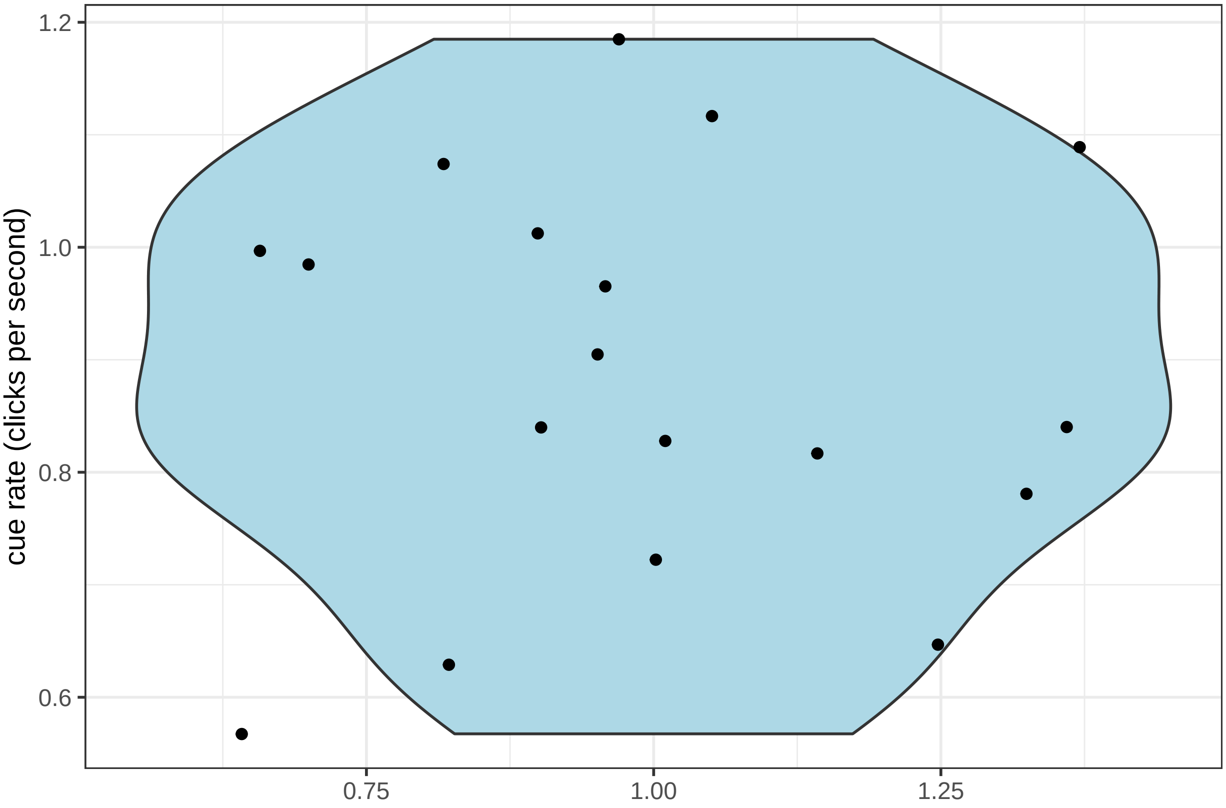
```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: Gamma  ( log )
## Formula: crate ~ (1 | location) + (1 | fyear)
##    Data: tags
##
##      AIC      BIC   logLik deviance df.resid
##     50.5     61.4    -21.2     42.5      108
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.0997 -0.5768  0.1072  0.7686  1.8868
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  fyear    (Intercept) 0.006517 0.08073
##  location (Intercept) 0.007293 0.08540
##  Residual             0.079031 0.28113
## Number of obs: 112, groups:  fyear, 12; location, 7
##
## Fixed effects:
##             Estimate Std. Error t value Pr(>|z|)
## (Intercept)  -0.1582     0.0961  -1.646   0.0997 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Under this representation, the cue rate would be estimated, for any new location-year combination as 0.85 clicks per second.

We note explicitly this is quite lower that the overall mean of all tags cue rates 0.93 clicks per second, a consequence of the tags being considerably unbalanced across years and locations, with more tags in year-location combinations which happened to have higher cue rates. This is nonetheless closer to, as would be expected, the mean of the average cue rates per year-location combination (0.89 clicks per second).

Note that the variability on the observed cue rates per year-location combination is not that large:

```
crates.psycomb<-data.frame(cr=tapply(X=tags$crate,INDEX = paste0(tags$location,tags$year),FUN=mean),sy=
ggplot(crates.psycomb,aes(x=1,y=cr),fill="lightblue")+
  theme_bw()+geom_violin(fill="lightblue")+
  geom_jitter()+ylab("cue rate (clicks per second)")+xlab("")
```



```
#boxplot(tapply(X=tags$crate,INDEX = paste0(tags$location,tags$year),FUN=mean))
```

Note that, given the model above, regarding the variance associated with the random effects, the percentage of the variation associated with year is 47.19 % and 52.81 % for location. In other words, there is more variation across time than across space, at least for the samples of space and time available to us.

Estimating the precision of the cue rate for a new year-location combination

Having estimated the mean cue rate at a new site-year to be 0.85 based on a GLMM with year and location as random effects, we need to quantify the precision around said mean.

To do so we need to propagate the variability associated with the random effects into the point estimate of the intercept of the model. This is not necessarily a straightforward thing to do.

The rationale for propagating uncertainty

Bolker et al. discuss it and note here under their GLMM FAQ section on Predictions and/or confidence (or prediction) intervals on predictions that "none of the following approaches takes the uncertainty of the random effects parameters into account…" and suggest that "if you want to take RE parameter uncertainty into account, a Bayesian approach is probably the easiest way to do it.".

Ben Bolker (BB) further provided several possible approaches to do so as a reply to a question TAM posted on stack exchange:

https://stats.stackexchange.com/questions/616697/how-to-estimate-precision-on-a-prediction-for-a-glmm-for-an-unobserved-level-of/

BB describes a possible approach via an (intuitive) parametric bootstrap. The suggested idea would be to resample values from the distribution of the overall mean, and then add sampled values from the variability associated with each of the random effects, and compute the variability of the resulting estimates. These have been referred to in the literature as "population prediction intervals". We note in passing that BB notes that this bootstrap procedure might be hard to justify from a theoretical point of view:

https://stats.stackexchange.com/questions/590595/justification-for-population-prediction-intervals

The 4 options listed by BB would be:

1. a procedure involving the analytic estimates of the variances of the parameters
2. a quick (parametric) bootstrap procedure
3. a true (parametric) bootstrap procedure (see below I actually implemented a possibly quite weird mix of a non-parametric+parametric bootstrap)
4. a full Bayesian implementation

Below I attempt to implement options 2-4.

Implementing uncertainty propagation

Option 2

A (simple) parametric bootstrap is the fastest to implement. In this case, the quick parametric bootstrap would correspond to resampling from Gaussian distributions for the intercept as well as the year and site random effects, based on the estimated model.

```r
# implementing option 2
# 2. a quick (parametric) bootstrap procedure
set.seed(123)
B<-9999
means<-rnorm(B,mean=summary(crglmer)$coefficients[1,1],sd=summary(crglmer)$coefficients[1,2])
desv.year<-rnorm(B,mean=0,sd=sqrt(as.numeric(summary(crglmer)$varcor[1])))
desv.loc<-rnorm(B,mean=0,sd=sqrt(as.numeric(summary(crglmer)$varcor[2])))

# need to include correlation on the variances of the random effects...

#estimates at a new year location combination, on the link scale
est.lmean.crs<-means+desv.year+desv.loc
#estimates on the scale of the response
est.mean.crs<-exp(est.lmean.crs)
```

Note that the above does not sample the right distributions, because it ignores the potential correlation between the variance of the random effects. However, when I looked at said correlation via the Bayesian implementation, option 4 below, there was no evidence for it to be present, so this might not be too bad as an approximation.

To get a 95% confidence interval (CI) we can simply use the percentile method, leading to a mean estimate of 0.85 and 95% confidence intervals of 0.64-1.15.

As BB noted, this option ignores variability in estimated random effects, since these are treated as fixed constants, not estimated values.

Note that I did not implement the methods strictly as described by BB, instead I did "almost" that, following what I had intuitively thought about originally. See below for the question that relates to this.

Option 3

As per the answer by BB (see also section 5.3 in Bolker 2008 https://math.mcmaster.ca/~bolker/emdbook/chap7A.pdf), while the above procedure is sensible, this approach will ignore the variability associated with the estimation of the random effects, and a "full" parametric bootstrap approach would first resample then re-fit the model".

Due to implementation decisions, I divide here option 3 into options 3a and 3b. To be fair, none is necessarily exactly what BB suggested.

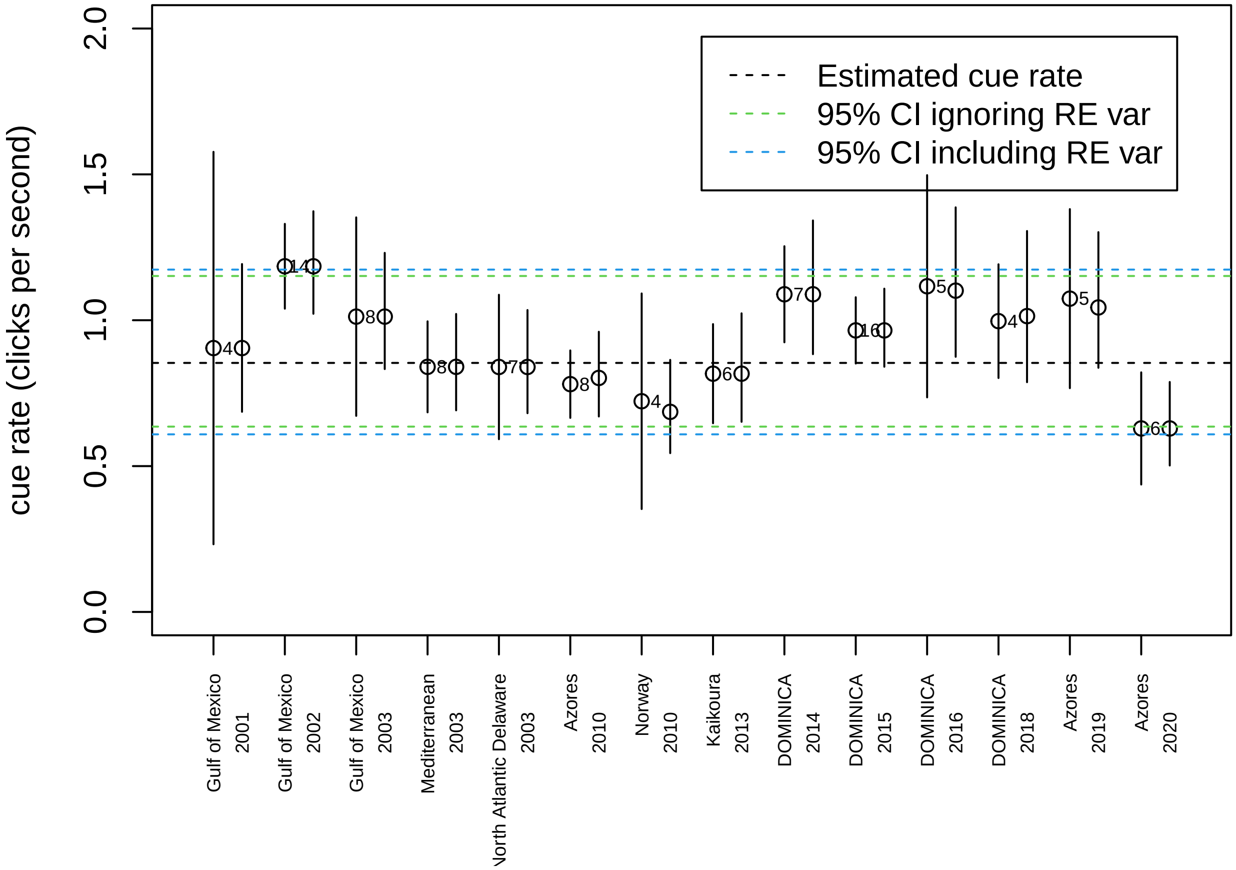
Option 3a I implement here below this second bootstrap, which would account for the variability in estimating the random effects, by sampling year-location combinations.

In hindsight, I might not have done just quite what BB was proposing, so I am coining this as option 3a. This option 3a might actually by a weird hybrid (incorporating notions from both non-parametric and parametric bootstrap) without any theoretical justification.

```
#unique location-year combinations
lys<-unique(tags$locyear)
#number of combinations
nlys<-length(lys)
B<-9999
res.boot<-numeric(B)
#for each bootstrapp resample
for(i in 1:B){
  #select a reasmple of location-year combinations
  ly1<-sample(lys,1)
  boot.tags<-tags[tags$locyear==ly1,]
  for(j in 2:nlys){
    boot.tags<-rbind(boot.tags,tags[tags$locyear==sample(lys,1),])
  }
  #fit the model
  # NOTE: while I silence the warning messages for the output of compiling the .Rmd
  # we get lots of "boundary (singular) fit: see help('isSingular')" warnings
  crglmer.boot<-glmer(crate~(1|location)+(1|fyear),data=boot.tags,family=Gamma(link="log"))
  means.b<-rnorm(1,mean=summary(crglmer.boot)$coefficients[1,1],sd=summary(crglmer.boot)$coefficients[1
  desv.year.b<-rnorm(1,mean=0,sd=sqrt(as.numeric(summary(crglmer.boot)$varcor[1])))
  desv.loc.b<-rnorm(1,mean=0,sd=sqrt(as.numeric(summary(crglmer.boot)$varcor[2])))
  #estimates at a new year location combination, on the link scale
  est.lmean.crs.b<-means.b+desv.year.b+desv.loc.b
  #estimates on the scale of the response
  res.boot[i]<-exp(est.lmean.crs.b)
}
```

As before we obtain a 95% CI using the percentile method: 0.61-1.17. As expected, this 95% CI is, even if just ever so slightly, wider than when ignoring the component of variation due to the random effects estimation. We can see that overlaid with the point and interval estimates presented before:

```
#standard average
par(mfrow=c(1,1),mar=c(6,4,0.1,0.4))
plot(x=(1:nrow(byly))-0.2,y=byly$ecr,xaxt="n",ylim=c(0,2),xlim=c(0.5,nrow(byly)+0.5),xlab="",ylab="cue r
with(byly,segments(x0=(1:nrow(byly))-0.2,x1=(1:nrow(byly))-0.2,y0=lcl.ecr,y1=ucl.ecr))
#GLM
points(x=(1:nrow(byly))+0.2,y=byly$glm.cr)
with(byly,segments(x0=(1:nrow(byly))+0.2,x1=(1:nrow(byly))+0.2,y0=glm.lci.cr,y1=glm.uci.cr))
#draw axis and annotations
axis(1, at=(1:nrow(byly))-0.2,byly$location,cex.axis=0.6,las=2)
axis(1, at=(1:nrow(byly))+0.2,byly$year,tick=FALSE,cex.axis=0.6,las=2,line=1)
text(x=(1:nrow(byly)),y=byly$ecr,labels=byly$ntags,cex=0.6)

#the estimated mean cue rate from GLMER
abline(h=exp(summary(crglmer)$coefficients[1,1]),lty=2)
#adding a new year location combination - 95% CI that ignore RE estimation variability
abline(h=quantile(est.mean.crs,probs=c(0.025,0.975)),col=3,lty=2)
#adding a new year location combination - 95% CI accounting for said variability
abline(h=quantile(res.boot,probs=c(0.025,0.975)),col=4,lty=2)
legend("topright",legend=c("Estimated cue rate","95% CI ignoring RE var","95% CI including RE var"),lty=
```



This result is quite interesting. As expected, if predicting for a new location-year not yet observed, we include within the confidence interval all but one of the cue rate point estimates obtained empirically for observed year-location combinations. The fact that we do not include the Gulf of Mexico in the year 2002, one of the year locations with the largest number of tags (14 tags) suggests that this year-location combination the cue rates were exceptionally high. The cause for that remains unresolved (**is sex info available? must check**).

Option 3b This section is Work in Progress and might become in fact deprecated. All the problems might have a common source and be down to an issue with either the `glmer` fit or to the way `simulate` works over a Gamma GLMM. More about this issue is below.

Notes on option 3 Some notes added on 8th June 2023, for BB benefit and for future reference. I've noticed that I might have mis-implemented BB's suggestions in (at least) two ways (above and beyond the "almost" bit I had already noted)

1. I now realize I got sidetracked and instead of what BB refers to as full parametric bootstrap, I actually created a mix of non-parametric with parametric bootstrap: first randomize independent sampling units (year-location combinations? individual tags? I lean towards the former, as that encompasses the second level of variability) observations, re-fit the model at each iteration, sampling from the variances associated with each random effect. **Note: While I can get an estimate of the mean at each bootstrap iteration, how do I get the variability on the random effects? At each iteration, there is none. Do I simply generate a single value for a possible mean of a year-location combination at each bootstrap resample? I guess so! (though this seems, intuitively, quite computationally ineffective). This question is actually hinting me that this is not what I am supposed to be doing!!**
2. For option 3, I might have ignored BB's comment "including the RE variance but not the FE sampling variance" since I sampled from the intercept distribution as well as for the random effects at each bootstrap resample (to be consistent with what I had done in option #2)

For option 3, regarding the non-parametric bootstrap, I actually resampled data from year-location combinations, not really year and location independently. The latter would mean one would end up with non-existing year-location combinations. However, said resampling scheme would be more consistent with a model where year-location combo was a single random effect. A downside of doing the modelling that way from the start is that we loose the ability to evaluate what is most important in explaining the variability in cue rates, year or location (here, as we saw above, most variability seems to come from year).

The main question remains. What is the full parametric procedure that BB suggested? How does one resample before re-fitting? What do I resample? Does that actually mean simulate new data, given the model, i.e.

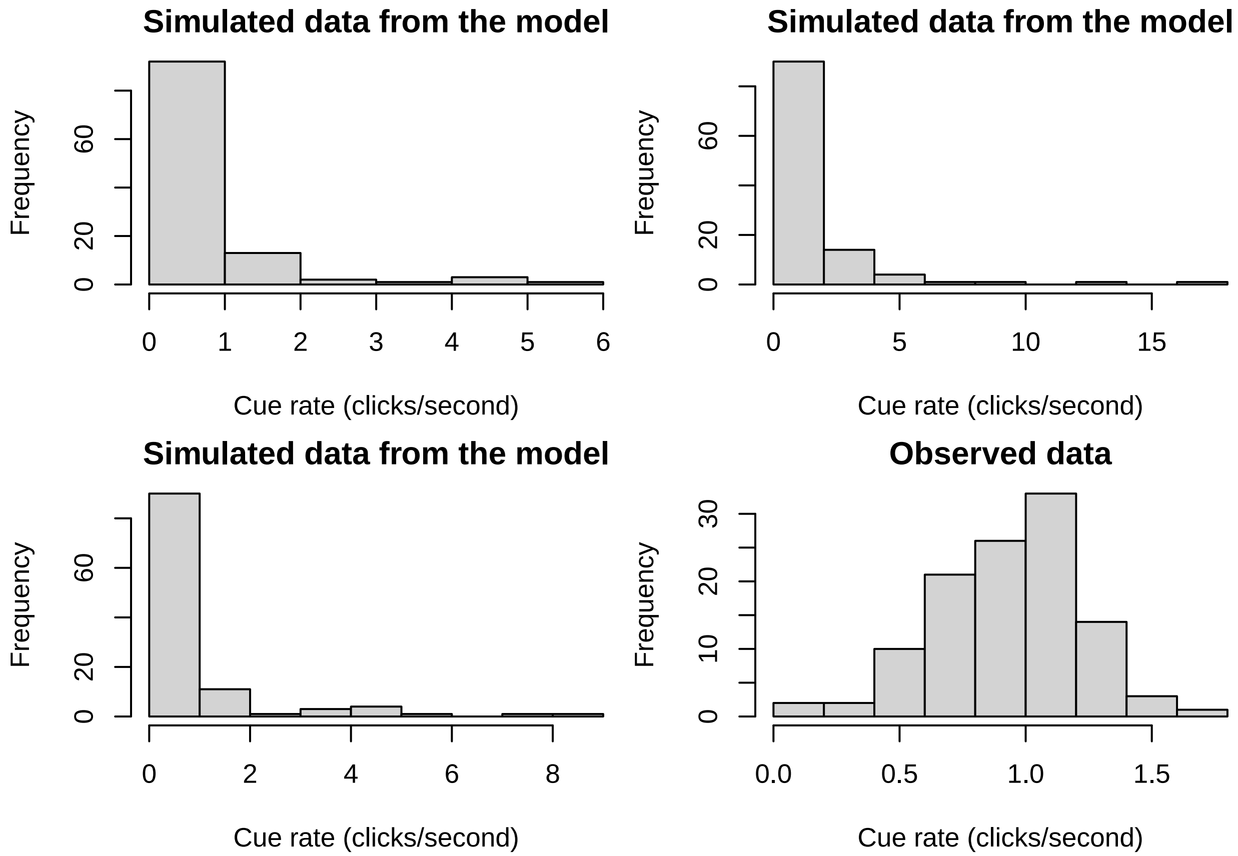
1. simulate data from the fitted model, that is, for each original data point (i.e. each tag)
    - simulate a draw for the estimated intercept (`int`)
    - add a draw from the estimated corresponding year value of the RE (`rey`)
    - add a draw from the estimated corresponding location value of the RE (`rel`)
    - draw a Gamma observation with a mean given by the `int+rey+rel` (but which variance? see also question below)
2. re-estimate the fitted model based on the simulated data
3. get the intercept value

The variance on these intercepts is then the variance one wants, which accounts for variability in the random effect estimation. **Nope, this can't be right. That is the variance on the overall mean, not the variance on the value of the mean of a possible new location-time combination.**

This resampling procedure seems somewhat circular to me, but that might be what one is actually supposed to do.

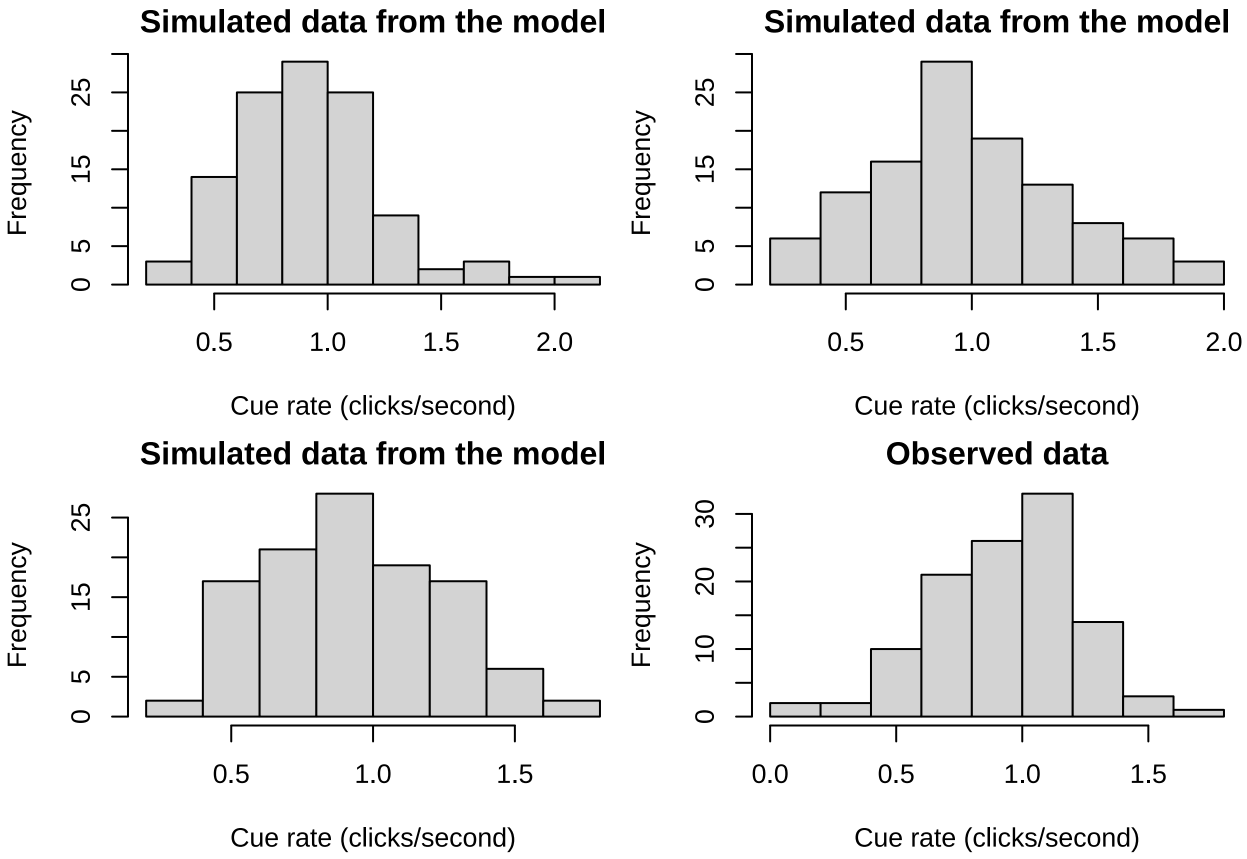
Presumably, one could easily simulate new data from this model via function `stats::simulate`. Unfortunately, the simulated data looks nothing like the real data, having considerable higher variability than the original data

```r
par(mfrow=c(2,2),mar=c(4,4,2,0.2))
hist(stats::simulate(crglmer)$sim_1,main="Simulated data from the model",xlab="Cue rate (clicks/second)
hist(stats::simulate(crglmer)$sim_1,main="Simulated data from the model",xlab="Cue rate (clicks/second)
hist(stats::simulate(crglmer)$sim_1,main="Simulated data from the model",xlab="Cue rate (clicks/second)
hist(tags$crate,main="Observed data",xlab="Cue rate (clicks/second)")
```



What does this tell us about the adequacy of the model? It implies that either the model's GOF is horrible or that there is something not working with `simulate`. I have been pointed to a git issue raised over `lme4` by Wei Zhang which seems to suggest issues with lme4 (https://github.com/lme4/lme4/issues/643). If one considers the model with year and location as fixed effects, `CRglm3` above, the simulated data via `simulate` looks much more like the real data, so maybe the issue is with the way `simulate` simulates from a random effects model? Might this be related just to the Gamma fitting in lme4?

```
par(mfrow=c(2,2),mar=c(4,4,2,0.2))
hist(stats::simulate(CRglm3)$sim_1,main="Simulated data from the model",xlab="Cue rate (clicks/second)"
hist(stats::simulate(CRglm3)$sim_1,main="Simulated data from the model",xlab="Cue rate (clicks/second)"
hist(stats::simulate(CRglm3)$sim_1,main="Simulated data from the model",xlab="Cue rate (clicks/second)"
hist(tags$crate,main="Observed data",xlab="Cue rate (clicks/second)")
```



Just realized that there are a couple of bespoke functions for `glmer` models:

- function called `simulate.merMod` that simulates from these models
- `bootMer` to resample and refit.

I tried `simulate.merMod` while specifying the random effect structure, but unfortunately, I get about the same strange results where simulated values from the model are not consistent with the data. The fact that when I simulate from a fixed effects model the simulated data is consistent with the observed data is confusing me.

From the help on `simulate.merMod` we see that

- `re.form` : formula for random effects to condition on. If `NULL`, condition on all random effects; if `NA` or `~0`, condition on no random effects. See Details.
- The `re.form` argument allows the user to specify how the random effects are incorporated in the simulation.
  - All of the random effects terms included in `re.form` will be conditioned on - that is, the conditional modes of those random effects will be included in the deterministic part of the simulation.
  - (If new levels are used (and `allow.new.levels` is TRUE), the conditional modes for these levels will be set to the population mode, i.e. values of zero will be used for the random effects.)
  - Conversely, the random effect terms that are not included in `re.form` will be simulated from - that is, new values will be chosen for each group based on the estimated random-effects variances.

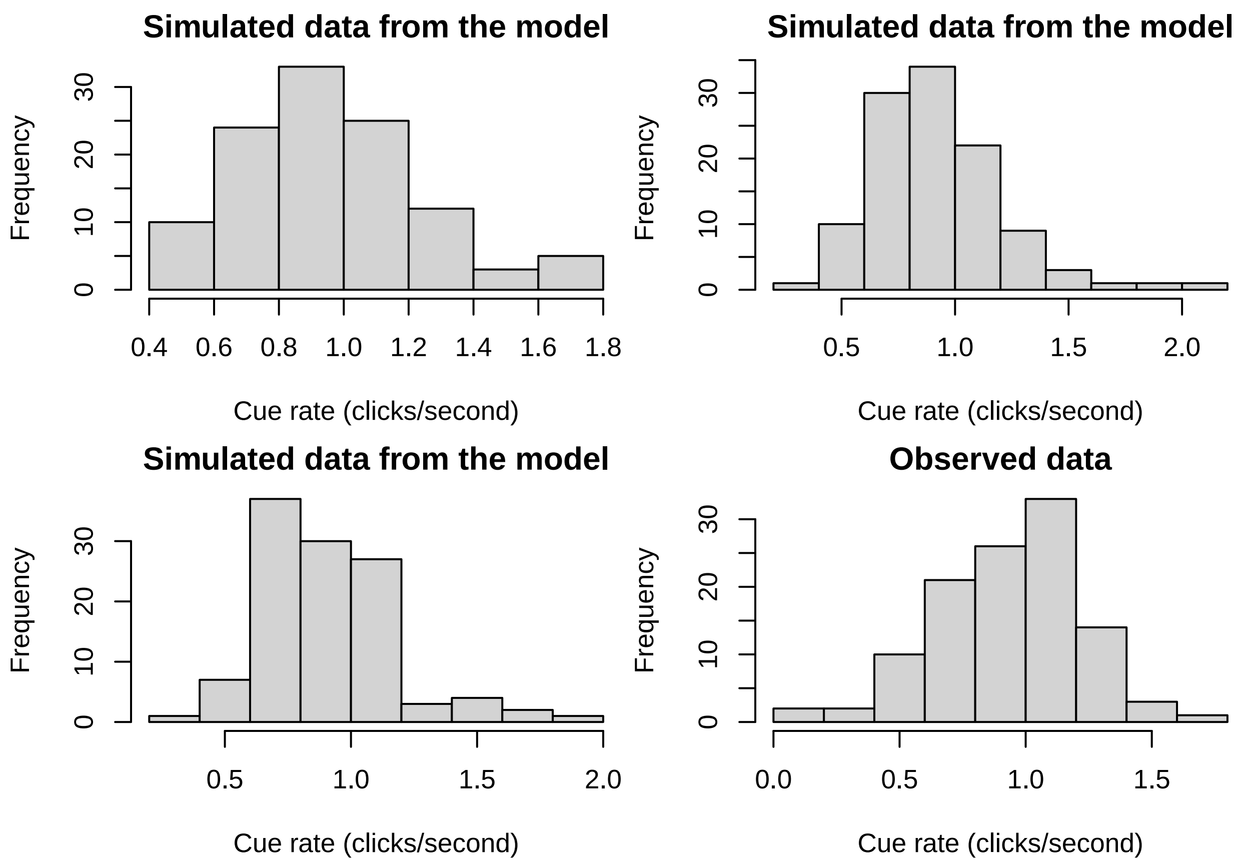
I actually tried all the options explicitly, `re.form=NULL`, `re.form=NA`, `re.form=0` and `re.form=~(1|location)+(1|fyear)`, but could not manage to get simulated data that looks like the observed data from any of these options.

Unrelated but still quite strange, the code with with anything but the default option of not explicitly specifying the argument `re.form` does not run in the .Rmd, returning the message Error in `mkNewReTrms()`, even though it does run on the command line. Said code is below for reference but with `eval=FALSE`.

```
par(mfrow=c(2,2),mar=c(4,4,2,0.2))
hist(stats::simulate(crglmer,re.form=NULL)$sim_1,main="Simulated data from the model",xlab="Cue rate (cl
hist(stats::simulate(crglmer,re.form=NA)$sim_1,main="Simulated data from the model",xlab="Cue rate (cli
hist(stats::simulate(crglmer,re.form=~(1|location)+(1|fyear))$sim_1,main="Simulated data from the model
hist(tags$crate,main="Observed data",xlab="Cue rate (clicks/second)")
```

If I simulate myself some data I obtain something which is far more sensible than the above, which again hints for the fact that `simulate` is doing gibberish.

```
par(mfrow=c(2,2),mar=c(4,4,2,0.2))
intercept<-summary(crglmer)$coefficients[1,1]
byyear <- ranef(crglmer)$fyear[,1]
yearnames<-rownames(coef(crglmer)$fyear)
byloc <- ranef(crglmer)$location[,1]
locnames <- rownames(coef(crglmer)$location)
disp<-summary(crglmer)$sigma^2
ntags<-nrow(tags)
new.data.mean<-numeric(ntags)
# example 1
for(i in 1:ntags){
new.data.mean[i]<-exp(intercept+byyear[which(yearnames==tags$fyear[i])]+byloc[which(locnames==tags$loca
}
new.data<-rgamma(ntags,shape=1/disp,scale=new.data.mean*disp)
hist(new.data,main="Simulated data from the model",xlab="Cue rate (clicks/second)")
# example 2
for(i in 1:ntags){
new.data.mean[i]<-exp(intercept+byyear[which(yearnames==tags$fyear[i])]+byloc[which(locnames==tags$loca
}
new.data<-rgamma(ntags,shape=1/disp,scale=new.data.mean*disp)
hist(new.data,main="Simulated data from the model",xlab="Cue rate (clicks/second)")
# example 3
for(i in 1:ntags){
new.data.mean[i]<-exp(intercept+byyear[which(yearnames==tags$fyear[i])]+byloc[which(locnames==tags$loca
}
new.data<-rgamma(ntags,shape=1/disp,scale=new.data.mean*disp)
hist(new.data,main="Simulated data from the model",xlab="Cue rate (clicks/second)")
hist(tags$crate,main="Observed data",xlab="Cue rate (clicks/second)")
```



Implementing option 3b the simulate way The results below make no sense, and this might be due to the way `simulate` is working, in particular given that if we `simulate` from a model with year and location as random effects, the data looks consistent with the real data, but if we `simulate` from the random effects model, it does not. Therefore, all the R code chucks in this section are with option `eval` det to `FALSE`.

```
B<-9999
res.boot.3b<-numeric(B)
boot.tags<-tags
for (i in 1:B){
  #get simulated data
  boot.tags$crate<-stats::simulate(crglmer)$sim_1
  #fit model
  crglmer.boot<-glmer(crate~(1|location)+(1|fyear),data=boot.tags,family=Gamma(link="log"))
  means.b<-rnorm(1,mean=summary(crglmer.boot)$coefficients[1,1],sd=summary(crglmer.boot)$coefficients[1
  desv.year.b<-rnorm(1,mean=0,sd=sqrt(as.numeric(summary(crglmer.boot)$varcor[1])))
  desv.loc.b<-rnorm(1,mean=0,sd=sqrt(as.numeric(summary(crglmer.boot)$varcor[2])))
  #estimates at a new year location combination, on the link scale
  est.lmean.crs.b<-means.b+desv.year.b+desv.loc.b
  #estimates on the scale of the response
  res.boot.3b[i]<-exp(est.lmean.crs.b)
}
#As before, we can obtain a 95% CI using the percentile method: `r round(quantile(res.boot.3b,0.025),2)
```

```
#standard average
par(mfrow=c(1,1),mar=c(6,4,0.1,0.4))
plot(x=(1:nrow(byly))-0.2,y=byly$ecr,xaxt="n",ylim=c(0,2),xlim=c(0.5,nrow(byly)+0.5),xlab="",ylab="cue 
with(byly,segments(x0=(1:nrow(byly))-0.2,x1=(1:nrow(byly))-0.2,y0=lcl.ecr,y1=ucl.ecr))
#GLM
points(x=(1:nrow(byly))+0.2,y=byly$glm.cr)
with(byly,segments(x0=(1:nrow(byly))+0.2,x1=(1:nrow(byly))+0.2,y0=glm.lci.cr,y1=glm.uci.cr))
#draw axis and annotations
axis(1, at=(1:nrow(byly))-0.2,byly$location,cex.axis=0.6,las=2)
axis(1, at=(1:nrow(byly))+0.2,byly$year,tick=FALSE,cex.axis=0.6,las=2,line=1)
text(x=(1:nrow(byly)),y=byly$ecr,labels=byly$ntags,cex=0.6)

#the estimated mean cue rate from GLMER
abline(h=exp(summary(crglmer)$coefficients[1,1]),lty=2)
#adding a new year location combination - 95% CI that ignore RE estimation variability
abline(h=quantile(est.mean.crs,probs=c(0.025,0.975)),col="blue",lty=2)
#adding a new year location combination - 95% CI accounting for said variability a la option 3a
abline(h=quantile(res.boot,probs=c(0.025,0.975)),col="green",lty=2)
#adding a new year location combination - 95% CI accounting for said variability a la option 3b
#abline(h=quantile(res.boot.3b,probs=c(0.025,0.975)),col=5,lty=2)
#hardwiring a line for the Bayesian implementation - option 4 - obtained in "GLMMinNimble.R"
abline(h=c(0.5229320,1.4197349),col="orange",lty=2,lwd=2)
legend("topright",legend=c("Estimated cue rate","option 2: 95% CI ignoring RE var","option 3a: 95% CI in
```

The confidence intervals make no sense. But then again, variances of the random effects are being routinely estimated to be 0… ?

As per the images below, the data simulated looks nothing like the observed data. While this was already hinted for in the histograms above, I was hopeful the "mean" values in each year-location combination would be somehow retrieved back - this seems not to be the case at all. If one looks at the values of the variances associated with the random effects and the value of the residual variance (but recall we are not sure this is a variance or the dispersion parameter) this might not be that surprising. These imply that much more variability (about an order of magnitude higher) remains unexplained by the residuals than that that is explained by the random effects. Where does this leave us?

```
par(mfrow=c(2,2))
with(tags,boxplot(crate~year,ylab="Cue rate (clicks/sec)",xlab="",cex.lab=0.8,cex.axis=0.8))
with(boot.tags,boxplot(crate~year,ylab="Cue rate (clicks/sec)",xlab="",cex.lab=0.8,cex.axis=0.8))
par(mar=c(8,4,0.2,0.2))
with(tags,boxplot(crate~location,las=2,ylab="Cue rate (clicks/sec)",xlab="",cex.lab=0.8,cex.axis=0.8))
with(boot.tags,boxplot(crate~location,las=2,ylab="Cue rate (clicks/sec)",xlab="",cex.lab=0.8,cex.axis=0
```

```
par(mfrow=c(1,2))
means.data.loc<-with(tags,tapply(X=crate,INDEX=location,FUN=mean))
means.boot.loc<-with(boot.tags,tapply(X=crate,INDEX=location,FUN=mean))
plot(means.data.loc,means.boot.loc)

means.data.y<-with(tags,tapply(X=crate,INDEX=year,FUN=mean))
means.boot.y<-with(boot.tags,tapply(X=crate,INDEX=year,FUN=mean))
plot(means.data.y,means.boot.y)
```

Implementing option 3b by hand Here we implement the simulation by hand.

```
B<-999
res.boot.3b<-numeric(B)
boot.tags<-tags
for (i in 1:B){
  #get simulated data
  for(m in 1:ntags){
    boot.tags$crate[m]<-exp(intercept+byyear[which(yearnames==tags$fyear[m])]+byloc[which(locnames==tag
  }
  #notice bad coding practice as I am using the same object to hold first mean on the link and then on
boot.tags$crate<-rgamma(ntags,shape=1/disp,scale=boot.tags$crate*disp)
  #fit model
  crglmer.boot<-glmer(crate~(1|location)+(1|fyear),data=boot.tags,family=Gamma(link="log"))
  means.b<-rnorm(1,mean=summary(crglmer.boot)$coefficients[1,1],sd=summary(crglmer.boot)$coefficients[1
  desv.year.b<-rnorm(1,mean=0,sd=sqrt(as.numeric(summary(crglmer.boot)$varcor[1])))
  desv.loc.b<-rnorm(1,mean=0,sd=sqrt(as.numeric(summary(crglmer.boot)$varcor[2])))
  #estimates at a new year location combination, on the link scale
  est.lmean.crs.b<-means.b+desv.year.b+desv.loc.b
  #estimates on the scale of the response
  res.boot.3b[i]<-exp(est.lmean.crs.b)
}
```

```
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.0039471 (tol = 0.002, component 1)
```

```
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.00514199 (tol = 0.002, component 1)
```

```
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.0081648 (tol = 0.002, component 1)
```

```
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.00696759 (tol = 0.002, component 1)
```

```
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.00523718 (tol = 0.002, component 1)
```

```
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
```

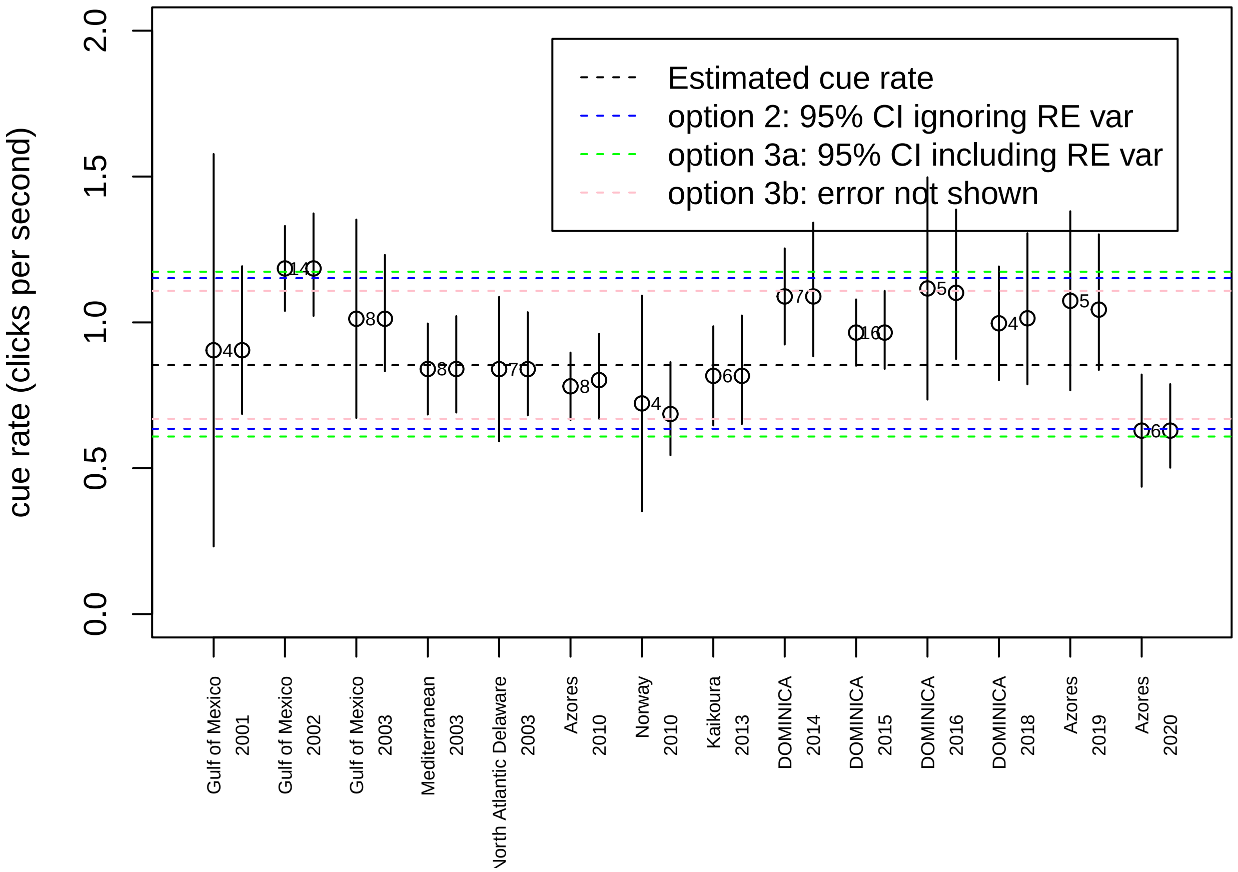
```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.00762247 (tol = 0.002, component 1)
```

```
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.00924901 (tol = 0.002, component 1)

## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.00786558 (tol = 0.002, component 1)

## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## unable to evaluate scaled gradient

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge: degenerate Hessian with 1 negative eigenvalues

## Warning in vcov(object, use.hessian = use.hessian): variance-covariance matrix computed from finite-
## not positive definite or contains NA values: falling back to var-cov estimated from RX

## Warning in vcov.merMod(object, correlation = correlation, sigm = sig): variance-covariance matrix co
## not positive definite or contains NA values: falling back to var-cov estimated from RX

## Warning in vcov(object, use.hessian = use.hessian): variance-covariance matrix computed from finite-
## not positive definite or contains NA values: falling back to var-cov estimated from RX

## Warning in vcov.merMod(object, correlation = correlation, sigm = sig): variance-covariance matrix co
## not positive definite or contains NA values: falling back to var-cov estimated from RX

## Warning in vcov(object, use.hessian = use.hessian): variance-covariance matrix computed from finite-
## not positive definite or contains NA values: falling back to var-cov estimated from RX

## Warning in vcov.merMod(object, correlation = correlation, sigm = sig): variance-covariance matrix co
## not positive definite or contains NA values: falling back to var-cov estimated from RX

## Warning in vcov(object, use.hessian = use.hessian): variance-covariance matrix computed from finite-
## not positive definite or contains NA values: falling back to var-cov estimated from RX

## Warning in vcov.merMod(object, correlation = correlation, sigm = sig): variance-covariance matrix co
## not positive definite or contains NA values: falling back to var-cov estimated from RX

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.00564348 (tol = 0.002, component 1)

## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.00912668 (tol = 0.002, component 1)

## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
```

```r
#As before, we can obtain a 95% CI using the percentile method: `r round(quantile(res.boot.3b,0.025),2)
```

Note that in this procedure there are a considerable number of warnings regarding the `glmer` fit which render this approach somewhat unreliable. I have not really investigated this deeper because I am leaning towards a Bayesian implementation in the end.

```r
#standard average
par(mfrow=c(1,1),mar=c(6,4,0.1,0.4))
plot(x=(1:nrow(byly))-0.2,y=byly$ecr,xaxt="n",ylim=c(0,2),xlim=c(0.5,nrow(byly)+0.5),xlab="",ylab="cue r
with(byly,segments(x0=(1:nrow(byly))-0.2,x1=(1:nrow(byly))-0.2,y0=lcl.ecr,y1=ucl.ecr))
#GLM
points(x=(1:nrow(byly))+0.2,y=byly$glm.cr)
with(byly,segments(x0=(1:nrow(byly))+0.2,x1=(1:nrow(byly))+0.2,y0=glm.lci.cr,y1=glm.uci.cr))
#draw axis and annotations
axis(1, at=(1:nrow(byly))-0.2,byly$location,cex.axis=0.6,las=2)
axis(1, at=(1:nrow(byly))+0.2,byly$year,tick=FALSE,cex.axis=0.6,las=2,line=1)
text(x=(1:nrow(byly)),y=byly$ecr,labels=byly$ntags,cex=0.6)

#the estimated mean cue rate from GLMER
abline(h=exp(summary(crglmer)$coefficients[1,1]),lty=2)
#adding a new year location combination - 95% CI that ignore RE estimation variability
abline(h=quantile(est.mean.crs,probs=c(0.025,0.975)),col="blue",lty=2)
#adding a new year location combination - 95% CI accounting for said variability a la option 3a
abline(h=quantile(res.boot,probs=c(0.025,0.975)),col="green",lty=2)
#adding a new year location combination - 95% CI accounting for said variability a la option 3b
abline(h=quantile(res.boot.3b,probs=c(0.025,0.975)),col="pink",lty=2)
legend("topright",legend=c("Estimated cue rate","option 2: 95% CI ignoring RE var","option 3a: 95% CI ir
```



Weirdly, we got a narrower 95% CI than for option 2, which makes no sense to me. But as I said, this might be unreliable given the issues in fitting over the simulated data mentioned above.

Option 4

Here I attempt to implement the GLMM model within a Bayesian context. I chose NIMBLE for implementation. Because we run into several issues with the results from the Bayesian implementation not adding up to the results from `lme4::glmer`, as noted in this question to the nimble users group

https://groups.google.com/g/nimble-users/c/7UHlKdCC8B4

we compare the results of:

1. The `lme4::glmer`
2. Nimble (MCMC)
3. `glmmTMB`
4. Nimble with Laplace approximation (MLE's) (see also https://r-nimble.org/html_manual/cha-AD.html)

First we define the Nimble model:

```
## define the model
GLMMcode <- nimbleCode({
  # the overall intercept
  beta0 ~ dnorm(0, sd = 10)
  # random effect standard deviation associated with location, a uniform, might change this to be somet
  sigmal_RE ~ dunif(0,10)
  # random effect standard deviation associated with year, a uniform, might change this to be something
  sigmay_RE ~ dunif(0, 10)
  # the gamma dispersion (or variance - see commented parametrization 1) parameter, a uniform, might ch
  #dispersion ~ dunif(0, 10)
  disp ~ dunif(0, 10)
  ## sd ~ dhalfflat()
  #get year random effects
  for(yy in 1:nyears){
    #REy[yy] ~ dnorm(0, sd = sigmay_RE)
    REy[yy] ~ dnorm(0, sd = sigmay_RE)
  }
  #get location random effects
  for(ll in 1:nlocs){
    #REl[ll] ~ dnorm(0, sd = sigmal_RE)
    REl[ll] ~ dnorm(0, sd = sigmal_RE)
  }
  for (i in 1:N){
    #get the linear predictor, consider a log link function
    log(mean[i]) <- beta0 + REy[year[i]] + REl[loc[i]]
    # now Using decentered parametrization, a suggestion by Ben Augustine
    # log(mean[i]) <- beta0 + REy[year[i]]*sigmay_RE + REl[loc[i]]*sigmal_RE
    # parametrization 1 - now I know that to not be what is to be used
    #left here for future reference
    # crate[i] ~ dgamma(shape=(mean[i]^2)/disp,scale=disp/mean[i])
    #parametrization 2
    crate[i] ~ dgamma(shape=1/disp, scale=mean[i]*disp)
  }
})
```

And then we define the required constants, data, and initial values, as well as the nodes to monitor in the MCMC

```
## constants, data, and initial values

#constant, sample size and number of levels for each of the random effects
#number of rows in tags
N<-nrow(tags)
#number of different years
nyears <- length(unique(tags$year))
#number of different locations
nlocs <- length(unique(tags$location))
#the year covariate is passed as a constant
year <- as.numeric(tags$fyear)
#the location covaraite is passed as a constant
loc <- as.numeric(tags$location)
#bundle all in a suitable object
constants <- list(N = N,nyears=nyears,nlocs=nlocs,year = year,loc = loc)

#data
data <- list(crate = tags$crate)

#initial values
#inits <- list(beta0 = 0, sigmal_RE = 1, sigmay_RE = 1, dispersion = 0.8,REy = rep(0,nyears),REl = rep(
inits <- list(beta0 = 0, sigmal_RE = 1, sigmay_RE = 1, disp = 1,REy = rep(0,nyears),REl = rep(0,nlocs))
```

```
## create the model object
myGLMMModel <- nimbleModel(code = GLMMcode, constants = constants, data = data,
                           inits = inits, check = FALSE, buildDerivs = TRUE) ## Add buildDerivs = TRUE for
```

```
## Defining model

## Building model

## Setting data and initial values

## Running calculate on model
##   [Note] Any error reports that follow may simply reflect missing values in model variables.

## Checking model sizes and dimensions
```

```r
cmyGLMMModel <- compileNimble(myGLMMModel)
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

```r
#things to monitor
#tomon<-c("beta0","dispersion","sigmay_RE","sigmal_RE","crate","REy","REl")
tomon<-c("beta0","disp","sigmay_RE","sigmal_RE","crate","REy","REl")
```

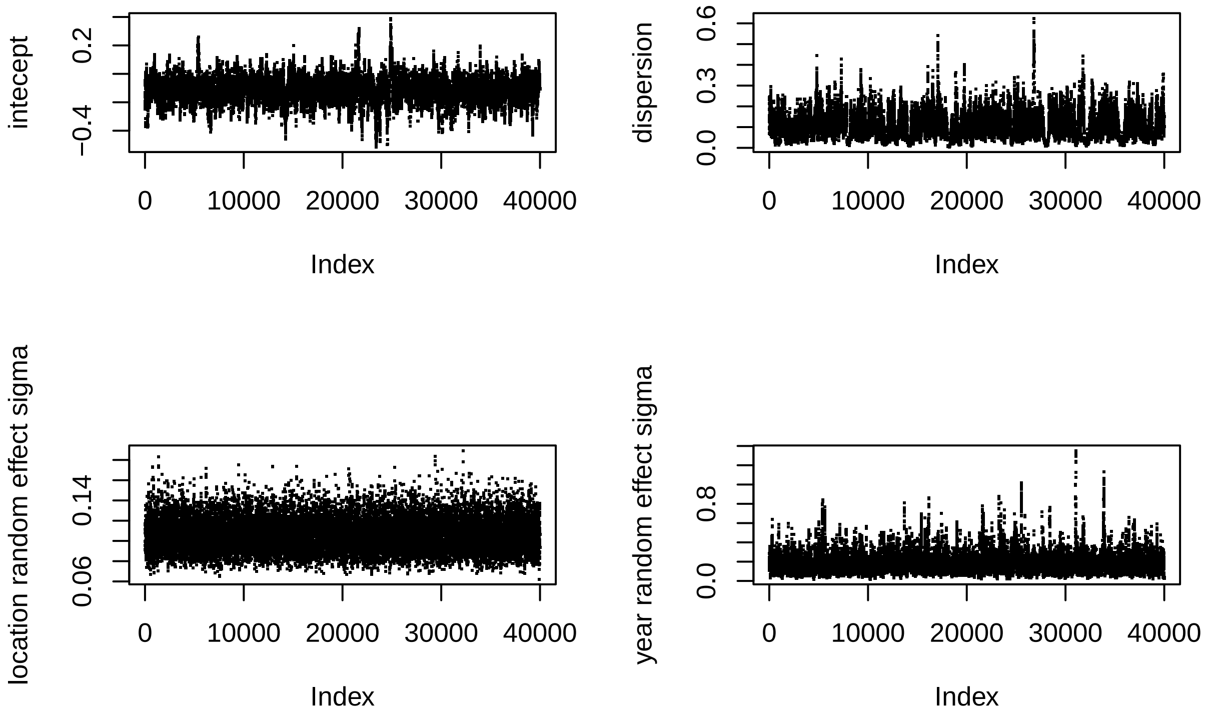
Then we run the code, considering 50000 iterations with a 10000 iterations burnin period, leaving 40000 iterations for inference.

```r
test<-nimbleMCMC(myGLMMModel,monitors=tomon,niter=50000,nburnin=10000,progressBar=TRUE,summary=TRUE)
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.

## running chain 1...

## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

There were no apparent issues with convergence/mixing of MCMC chains

```r
par(mfrow=c(2,2))
#trace plot intercept
plot(test$samples[,20],pch=".",ylab="intecept")
#trace plot dispersion
plot(test$samples[,135],pch=".",ylab="dispersion")
#trace plot year random effect standard deviation
plot(test$samples[,133],pch=".",ylab="location random effect sigma")
#trace plot location random effect standard deviation
plot(test$samples[,134],pch=".",ylab="year random effect sigma")
```



We can look at the main results, i.e. all top-level stochastic nodes of the model, namely the intercept, the dispersion parameter and the standard deviations of the random effects

```
#look at main results -
test$summary[c(20,133,134,135),]
```

```
##                  Mean      Median    St.Dev.   95%CI_low  95%CI_upp
## beta0      -0.1167985 -0.11394190 0.09016858 -0.30756200 0.05800892
## disp        0.1058820  0.10449963 0.01530254  0.08003455 0.14003796
## sigmal_RE   0.1717141  0.14898503 0.10350975  0.04518999 0.43860800
## sigmay_RE   0.1005263  0.09338585 0.05924980  0.01545019 0.23455656
```

However, the results are somewhat inconsistent with what we got in `glmer`. For additional comparisons, as noted above we also implemented a Nimble-Laplace approximation, which provides MLE's by integrating over the continous random effects distributios

```
## Build Laplace approximation in nimble
glmmNimLaplace <- buildLaplace(myGLMMModel)
cglmmNimLaplace <- compileNimble(glmmNimLaplace, project = myGLMMModel)
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

```
MLEres <- cglmmNimLaplace$findMLE()
#get MLE's
summ_MLEres <- cglmmNimLaplace$summary(MLEres)
#output with proper names - via code in https://r-nimble.org/html_manual/cha-AD.html
summaryLaplace(cglmmNimLaplace, MLEres)$params
```

```
##              estimate         se
## beta0      -0.1150270 0.06166998
## sigmal_RE   0.1120790 0.05036673
## sigmay_RE   0.0702499 0.06047423
## disp        0.1022385 0.01442276
```

and we also implement the same model using `glmmTMB` approach

```
# Use glmmTMB, parametrizes the model in the same way as the nimble model code below
crglmmTMB<-glmmTMB(crate~(1|location)+(1|fyear),data=tags,family=Gamma(link="log"))
```

Comparing the 4 implementations, we have for the intercept an estimate of -0.1168, -0.1582, -0.115 and -0.115 for Nimble MCMC, `glmer`, `glmmTMB` and Nimble-Laplace MLE, respectively.

```
#the beta0 matches OK-ish with the intercept from glmer
test$summary[20,1]
summary(crglmer)$coeff[1,1]
summary(crglmmTMB)$coefficients$cond[1]
summ_MLEres$params$estimates[1]
```

For the dispersion parameter we estimates of 0.1059, 0.079, 0.1022 and 0.1022 for Nimble MCMC, `glmer`, `glmmTMB` and Nimble-Laplace MLE, respectively.

```
# Note that parametrized as 2 the dispersion parameter does not seem to bear a direct relation to the v
#or... does it? see comment in glmmTMB output below
test$summary[133,1];
summary(crglmer)$sigma^2
#in glmmTMB output it is noted
#Dispersion estimate for Gamma family (sigma^2):
summary(crglmmTMB)$sigma^2 #Closer
#another way
#sd(resid(crglmer,type='pear'))^2
#sd(resid(crglmer,type='response'))^2
summ_MLEres$params$estimates[4]
```

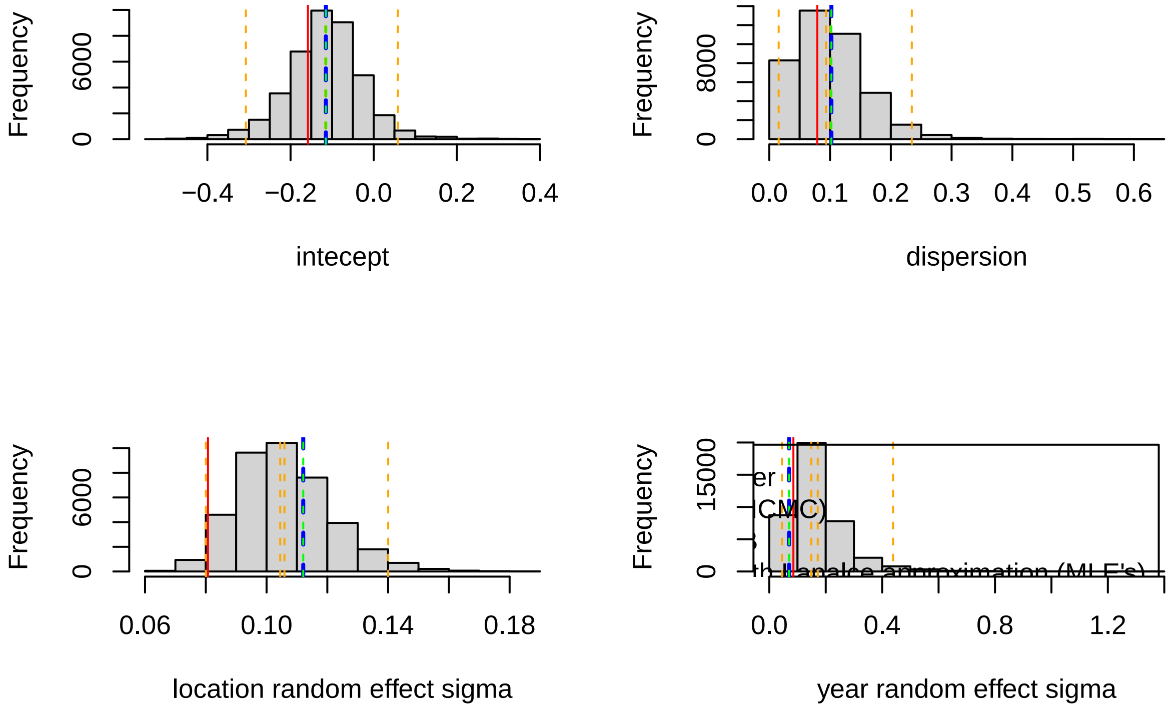
and for the random effects standard deviations, we have for location random effect estimates of 0.1717, 0.0807, 0.1121 and 0.1121 for Nimble MCMC, `glmer`, `glmmTMB` and Nimble-Laplace MLE, respectively, while for the year random effect estimates of 0.1005, 0.0854, and 0.0702 for Nimble MCMC, `glmer`, `glmmTMB` and Nimble-Laplace MLE, respectively.

```
test$summary[134:135,1]
summary(crglmer)$varcor
summary(crglmmTMB)$varcor ## Closer
summ_MLEres$params$estimates[2:3]
```

We can compare visually the estimated values under `glmer`, `glmmTMB`, NIMBLE MLE's via Laplace approximations and the posteriors from NIMBLE, and as noted, things do not add up

```
par(mfrow=c(2,2))
#posterior plot intercept
hist(test$samples[,20],pch=".",xlab="intecept",main="")
#the intercept from glmer
abline(v=summary(crglmer)$coeff[1,1],col="red")
abline(v=quantile(test$samples[,20],probs=c(0.025,0.5,0.975)),col="orange",lty=2)
abline(v=mean(test$samples[,20]),col="orange",lty=2)
abline(v=summary(crglmmTMB)$coefficients$cond[1],col="blue",lty=2,lwd=2)
```

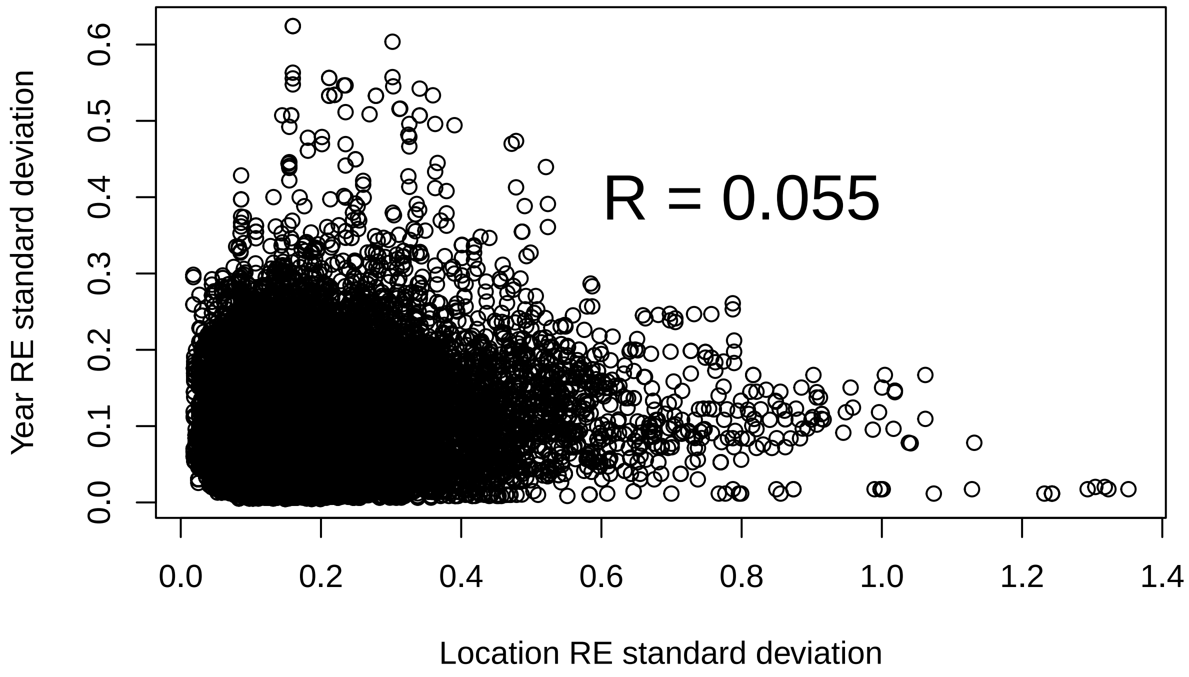
```r
abline(v=summ_MLEres$params$estimates[1],col="green",lty=2)
#posterior plot dispersion
hist(test$samples[,135],pch=".",xlab="dispersion",main="")
#the variance in glmer (I suspect that is the dispersion, waiting for BB to confirm)
abline(v=summary(crglmer)$sigma^2,col="red")
abline(v=quantile(test$samples[,135],probs=c(0.025,0.5,0.975)),col="orange",lty=2)
abline(v=mean(test$samples[,135]),col="orange",lty=2)
abline(v=summary(crglmmTMB)$sigma^2,col="blue",lty=2,lwd=2)
abline(v=summ_MLEres$params$estimates[4],col="green",lty=2)
#posterior plot location random effect standard deviation
hist(test$samples[,133],pch=".",xlab="location random effect sigma",main="")
abline(v=sqrt(as.numeric(summary(crglmer)$varcor))[1],col="red")
abline(v=quantile(test$samples[,133],probs=c(0.025,0.5,0.975)),col="orange",lty=2)
abline(v=mean(test$samples[,133]),col="orange",lty=2)
abline(v= sqrt(as.numeric(summary(crglmmTMB)$varcor$cond$location)),col="blue",lty=2,lwd=2)
abline(v=summ_MLEres$params$estimates[2],col="green",lty=2)
#posterior plot year random effect standard deviation
hist(test$samples[,134],pch=".",xlab="year random effect sigma",main="")
abline(v=sqrt(as.numeric(summary(crglmer)$varcor))[2],col="red")
abline(v=quantile(test$samples[,134],probs=c(0.025,0.5,0.975)),col="orange",lty=2)
abline(v=mean(test$samples[,134]),col="orange",lty=2)
abline(v= sqrt(as.numeric(summary(crglmmTMB)$varcor$cond$fyear)),col="blue",lty=2,lwd=2)
abline(v=summ_MLEres$params$estimates[3],col="green",lty=2)
legend("topright",inset=0.05,legend=c("lme4::glmer","Nimble (MCMC)","glmmTMB","Nimble with Lapalce appr
```



In summary, the `lme4::glmer` was off for both the intercept and the dispersion parameter, which were in turn consistent across the other 3 implementations. For the random effect variances the Nimble MLE's (via Laplace approximation) and `glmmTMB` were virtually identical, with Nimble MCMC and `lme4::glmer` show inconsistent results.

Just out of curiosity, I took a peak at what was the correlation between the random effects standard deviations, and it looks like there is none, even if one tests it formally (P-value=0). This links to a detail noted above.

```
plot(test$samples[,134],test$samples[,135],xlab="Location RE standard deviation",ylab="Year RE standard
text(0.8,0.4,labels =paste0("R = ",round(cor.test(test$samples[,134],test$samples[,135])$estimate,4)),c
```



Assuming that the MCMC approach, which does not use any approximations, is the one most reliable, we would finally obtain the variability aroun the mean by simulating from the relevant posterior distributions

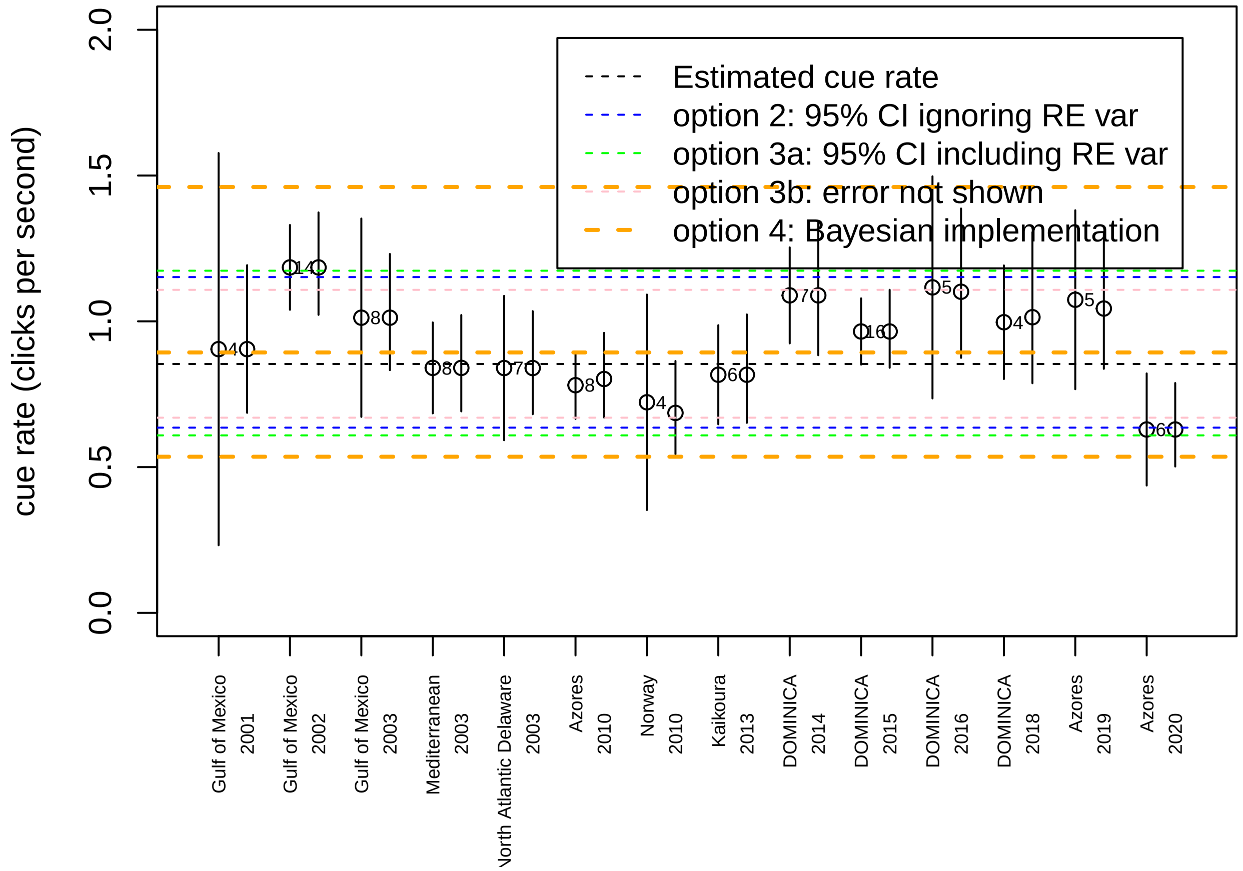
```
# getting the overall estimate for a new year and location and respective CI's we are looking for
# heuristically appealing but... is this kosher?
nsamples<-nrow(test$samples)
myquants95CI.Bayes<-quantile(exp(test$samples[,20]+rnorm(nsamples,mean=0,sd=test$samples[,134])+rnorm(ns
```

Comparing Options

Here we present the comparison of option 2, 3a and 4. 3b is not shown given the issues mentioned above. I think I prefer the Bayesian option, since that seems to be the only one that avoids approximations which might be the cause for discrepancies.

```
#standard average
par(mfrow=c(1,1),mar=c(6,4,0.1,0.4))
plot(x=(1:nrow(byly))-0.2,y=byly$ecr,xaxt="n",ylim=c(0,2),xlim=c(0.5,nrow(byly)+0.5),xlab="",ylab="cue
with(byly,segments(x0=(1:nrow(byly))-0.2,x1=(1:nrow(byly))-0.2,y0=lcl.ecr,y1=ucl.ecr))
#GLM
```

```
points(x=(1:nrow(byly))+0.2,y=byly$glm.cr)
with(byly,segments(x0=(1:nrow(byly))+0.2,x1=(1:nrow(byly))+0.2,y0=glm.lci.cr,y1=glm.uci.cr))
#draw axis and annotations
axis(1, at=(1:nrow(byly))-0.2,byly$location,cex.axis=0.6,las=2)
axis(1, at=(1:nrow(byly))+0.2,byly$year,tick=FALSE,cex.axis=0.6,las=2,line=1)
text(x=(1:nrow(byly)),y=byly$ecr,labels=byly$ntags,cex=0.6)
#the estimated mean cue rate from GLMER
abline(h=exp(summary(crglmer)$coefficients[1,1]),lty=2)
#adding a new year location combination - 95% CI that ignore RE estimation variability
abline(h=quantile(est.mean.crs,probs=c(0.025,0.975)),col="blue",lty=2)
#adding a new year location combination - 95% CI accounting for said variability a la option 3a
abline(h=quantile(res.boot,probs=c(0.025,0.975)),col="green",lty=2)
#adding a new year location combination - 95% CI accounting for said variability a la option 3b
abline(h=quantile(res.boot.3b,probs=c(0.025,0.975)),col="pink",lty=2)
#hardwiring a line for the Bayesian implementation - option 4 - obtained in "GLMMinNimble.R"
abline(h=myquants95CI.Bayes,col="orange",lty=2,lwd=2)
legend("topright",legend=c("Estimated cue rate","option 2: 95% CI ignoring RE var","option 3a: 95% CI i
```



Questions

Here are some pending questions that one might want to look into later:

For myself

- Given the variance across locations vs within locations, one might be better off MRSE-wise to estimate from other locations.

For BB

Question 1

Regarding option 2, BB suggested the following procedure:

- draw MVN samples (using `MASS::mvrnorm` or one of the other alternatives in the R ecosystem) from the distribution with the combined FE sampling variance + RE variance
- exponentiate them, and
- draw Gamma samples based on those mean values

but it is unclear to me why:

- one would want to implement the third step, draw from a Gamma, since what we are interested is in the variation on the mean of a new year-location combination, not the variance in observations from such a location;

Question 2

- (this is potentially a mute point/detail, but nonetheless I would like to know) what said Gamma would be, given via `glmer` we only get the mean value of the Gamma. Where in the output of `glmer` lies the estimate for the dispersion parameter of the corresponding Gamma (as happens to be reported by say a `glm` call with the argument `family=Gamma`)? I am working on the assumption that the reported variance is in fact the dispersion, but I have not received confirmation from BB.

Question 3

Not that this is that relevant given I might go for the Bayesian implementation, since as noted by BB from the start, that is the approach which more naturally allows us to propagate the uncertainty to the quantity we want to estimate, which is a cue rate at a new time and location not observed in the data, but implementation of option 4b indicates that there are issues with `simulate` over Gamma GLMM model objects. It would be interesting to hear from BB what that might be about. This might link to another issue raised on github about `lme4` Gamma GLMMs (https://github.com/lme4/lme4/issues/643).

Recycle bin

As the section name implies, this is just here temporarily before I decide to delete it for good.

We do note a few arbitrary combinations that might be interesting to compare:

```r
#define indexes of year-location combinations we are most interested in
iGoM2002<-tags$location=="Gulf of Mexico" & tags$year==2002
iGoM2001<-tags$location=="Gulf of Mexico" & tags$year==2001
iDom2015<-tags$location=="DOMINICA" & tags$year==2015
iDom2018<-tags$location=="DOMINICA" & tags$year==2018
```

- a location and year for which we have available data. We consider here the cases for which we have the most tags, namely
  - the Gulf of Mexico in 2002 (14 tags), and
  - Dominica in 2015 (16 tags), and
- for these same locations, two different years when we have less data, namely
  - 2001 for the Gulf of Mexico (4 tags) and
  - 2018 for Dominica (4 tags).

```r
#this might be safely deleted as it is all above
ntags.GoM2002 <- sum(iGoM2002)
ntags.GoM2001 <- sum(iGoM2001)
ntags.Dom2015 <- sum(iDom2015)
ntags.Dom2018 <- sum(iDom2018)
mean.cr.GoM2002 <- mean(tags$crate[iGoM2002])
mean.cr.GoM2001 <- mean(tags$crate[iGoM2001])
mean.cr.Dom2015 <- mean(tags$crate[iDom2015])
mean.cr.Dom2018 <- mean(tags$crate[iDom2018])
sd.cr.GoM2002 <- sd(tags$crate[iGoM2002])
sd.cr.GoM2001 <- sd(tags$crate[iGoM2001])
sd.cr.Dom2015 <- sd(tags$crate[iDom2015])
sd.cr.Dom2018 <- sd(tags$crate[iDom2018])
margin.cr.GoM2002 <- qt(0.975,ntags.GoM2002-1)*sd(tags$crate[iGoM2002])/sqrt(ntags.GoM2002)
margin.cr.GoM2001 <- qt(0.975,ntags.GoM2001-1)*sd(tags$crate[iGoM2001])/sqrt(ntags.GoM2001)
margin.cr.Dom2015 <- qt(0.975,ntags.Dom2015-1)*sd(tags$crate[iDom2015])/sqrt(ntags.Dom2015)
margin.cr.Dom2018 <- qt(0.975,ntags.Dom2018-1)*sd(tags$crate[iDom2018])/sqrt(ntags.Dom2018)
```

The point estimates and respective 95% confidence intervals using a standard average are:

- Gulf of Mexico in 2002 (14 tags): 1.18 (1.04-1.33)
- Gulf of Mexico in 2001 (4 tags): 0.9 (0.23-1.58)
- Dominica in 2015 (16 tags): 0.97 (0.85-1.08)
- Dominica in 2018 (4 tags): 1 (0.8-1.19)

Aknowledgements

This might be an internal document, but I like to show my gratitude as much as possible.

We thank Ben Bolker for helpful advice via the answer to our question on “Stack Exchange” at https://stats.stackexchange.com/questions/616697/how-to-estimate-precision-on-a-prediction-for-a-glmm-for-an-unobserved-level-of/. Also thanks to Ben Augustine and Perry du Valpine over email, regarding the comparison between Bayesian inferences in Nimble and `lme4`. Luca (in private) and Wei Zhang (both in private and publicly) provided helpful advice following the question on the nimble user group at https://groups.google.com/g/nimble-users/c/7UHlKdCC8B4 about the same comparison. Len Thomas provided useful discussion and advice.

All errors remaining are the sole responsability of the author.