Ships in ice infested waters

Exam, 15.12.2020, practical exercise

Load the needed libraries.

```
library(ggplot2)
library(StanHeaders)
library(rstan)
```

```
## rstan (Version 2.19.3, GitRev: 2e1f913d3ca3)

## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```
set.seed(123)

options(mc.cores = parallel::detectCores())
rstan_options(auto_write = TRUE)
```

Data

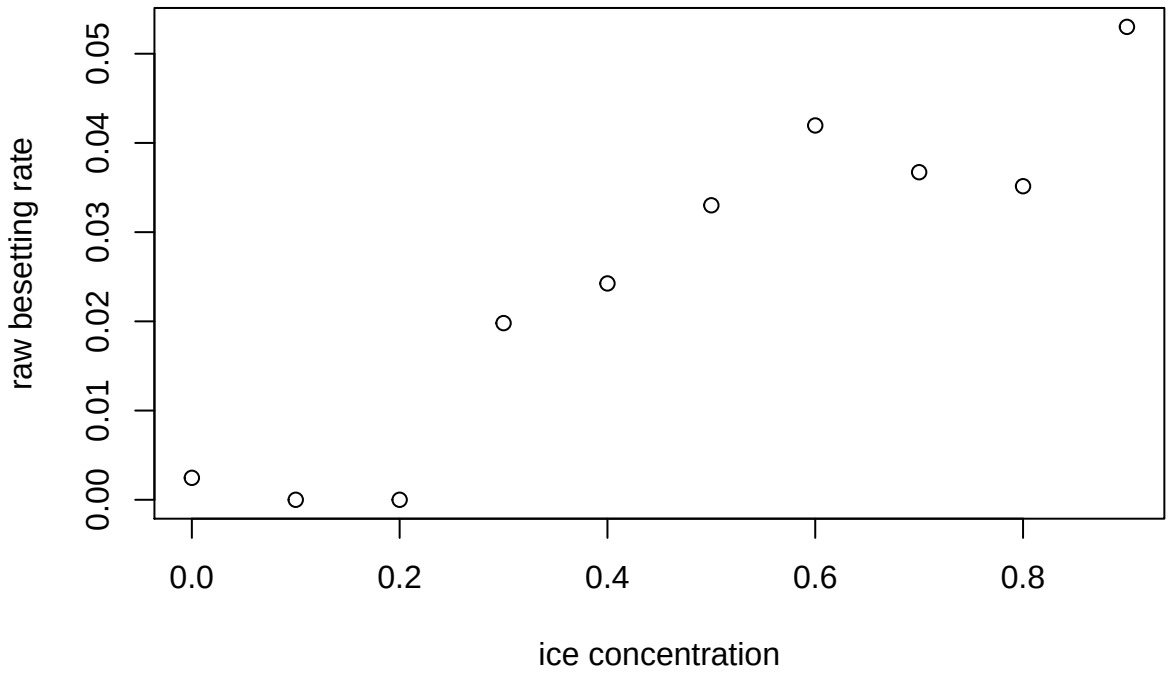
The aim of this exercise is to study how the probability of ice besetting events in the Arctic sea depends on ice concentration. A besetting event is an event where a ship gets stuck in ice and cannot move without assistance.

We collected data on distance traveled by Polar Class C ships in different ice concentrations and the number of besetting events that those ships experienced during those trips. The data is collected from the Kara Sea, a sea area in the northern sea route, and covers years 2013-2017. Let's first load the data and visualize the raw besetting event rate

```
data.icebesetting = read.table("kara_sea_ice_besetting_events_2013-2017.csv",header = TRUE,sep = ",")
print(data.icebesetting)
```

```
##    Minimum.ice.concentration distance..x1000nm. number.of.besetting.events
## 1                        0.0            406.728                          1
## 2                        0.1             42.317                          0
## 3                        0.2             83.345                          0
## 4                        0.3            100.998                          2
## 5                        0.4            164.952                          4
## 6                        0.5             90.883                          3
## 7                        0.6             71.497                          3
## 8                        0.7            136.173                          5
## 9                        0.8            256.039                          9
## 10                       0.9            358.427                         19
```

```
plot(data.icebesetting$Minimum.ice.concentration,
     data.icebesetting$number.of.besetting.events/data.icebesetting$distance..x1000nm.,
     ylab="raw besetting rate", xlab="ice concentration")
```



Note the ice concentration in the data has been categorized into 10 classes (0,0.1], (0.1,0.2],..,(0.9,1]. According to the above plot the besetting event rate seems to increase with ice concentration. However, let's analyze the data more properly so that we get better understanding what part of the increase is due to increase in the true besetting event rate and which part is just due to stochasticity in the process.

Let's denote by x_i the ice concentration in ith data row, by d_i the distance traveled in ice concentration x_i and by y_i the number of besetting events in that ice concentration. The statistical model to be used to analyze the data is based on classical exponential distribution for event analysis. That is, we assume that the "vulnerability" of a ship to get stuck in ice is described by a rate parameter

$$\lambda(x) = E\left[\frac{\text{number of besetting events in ice concentration } x}{\text{distance travelled in ice concentration } x}\right]$$

so that the probability distribution for distance between besetting events is $p(d|\lambda(x)) = \lambda(x)e^{-\lambda(x)d}$ (for $d > 0$). The cumulative distribution function of the exponential distribution gives the probability that there will be a besetting event within distance d in ice concentration x as

$$\Pr(\text{besetting event within distance } d|\lambda(x)) = 1 - e^{-d\times\lambda(x)}$$

The exponential distribution for distance between consecutive besetting events leads to a Poisson distribution for the number of accidents during a given distance. Hence, the model for our data, the numbers of accidents that have happened to ships in ice concentrations x_i during total distances d_i, is

$$y_i|d_i, \lambda(x_i) \sim \text{Poisson}(d_i \times \lambda(x_i))$$

Let's assume that the besetting rate changes continuously with ice concentration. To model this, we model the log rate as

$$\log \lambda(x_i) = \beta_0 + \beta_1 \times x_i$$

where β_0 is the log baseline rate and β_1 is the linear weight for ice concentration. Let's give vague prior distribution for the model parameters so that

$$\beta_0 \sim N(0, 5)$$
$$\beta_1 \sim N(0, 5)$$

Your tasks are the following:

1. Implement the model in Stan and sample from the posterior for the parameters β_0 and β_1. Check for convergence of the MCMC chain and examine the autocorrelation of the samples and discuss the results of the convergence check and autocorrelation.

The model

```
icebesetting.model  = "
data{
  int<lower=0> n; // number of observations
  int<lower=0> y[n];     // number of besetting events
  real<lower=0> d[n];    // distance
  real<lower=0,upper =1 > x[n];    // observed ice concentrations
}


parameters{
  real Beta0;
  real Beta1;

}

transformed parameters{


  real lambda[n];

  for( i in 1 : n ) {
    lambda[i] = d[i] * exp(Beta0 + Beta1 * x[i]);
  }

}

model{

Beta0 ~ normal(0,sqrt(5));
```

```
Beta1 ~ normal(0,sqrt(5));
 for( i in 1 : n ) {
 y[i] ~ poisson(lambda[i]);
 }

}
"
```

```
data.icebesetting
```

```
##    Minimum.ice.concentration distance..x1000nm. number.of.besetting.events
## 1                        0.0            406.728                          1
## 2                        0.1             42.317                          0
## 3                        0.2             83.345                          0
## 4                        0.3            100.998                          2
## 5                        0.4            164.952                          4
## 6                        0.5             90.883                          3
## 7                        0.6             71.497                          3
## 8                        0.7            136.173                          5
## 9                        0.8            256.039                          9
## 10                       0.9            358.427                         19
```

```
#data.icebesetting$Minimum.ice.concentration = as.factor(data.icebesetting$Minimum.ice.concentration)
x= data.icebesetting$Minimum.ice.concentration
y = data.icebesetting$number.of.besetting.events
d = data.icebesetting$distance..x1000nm.
```

```
data.stan <- list ("n"=length(y),"d" = d, "y" = y, "x" = x)

post=stan(model_code=icebesetting.model,data=data.stan,
                warmup=200,iter=400,chains=3)
```

```
## Trying to compile a simple C file

## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## gcc -std=gnu99 -I"/usr/share/R/include" -DNDEBUG   -I"/usr/lib/R/site-library/Rcpp/include/"  -I"/usr
## In file included from /usr/lib/R/site-library/RcppEigen/include/Eigen/Core:88,
##                  from /usr/lib/R/site-library/RcppEigen/include/Eigen/Dense:1,
##                  from /usr/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen.hpp:13
##                  from <command-line>:
## /usr/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:1: error: unknown type nam
##   613 | namespace Eigen {
##       | ^~~~~~~~
## /usr/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:17: error: expected '=',
##   613 | namespace Eigen {
##       |                 ^
## In file included from /usr/lib/R/site-library/RcppEigen/include/Eigen/Dense:1,
##                  from /usr/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen.hpp:13
##                  from <command-line>:
## /usr/lib/R/site-library/RcppEigen/include/Eigen/Core:96:10: fatal error: complex: No such file or di
##    96 | #include <complex>
##       |          ^~~~~~~~
```

```
## compilation terminated.
## make: *** [/usr/lib/R/etc/Makeconf:167: foo.o] Error 1
```

```
## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#bulk-ess
```

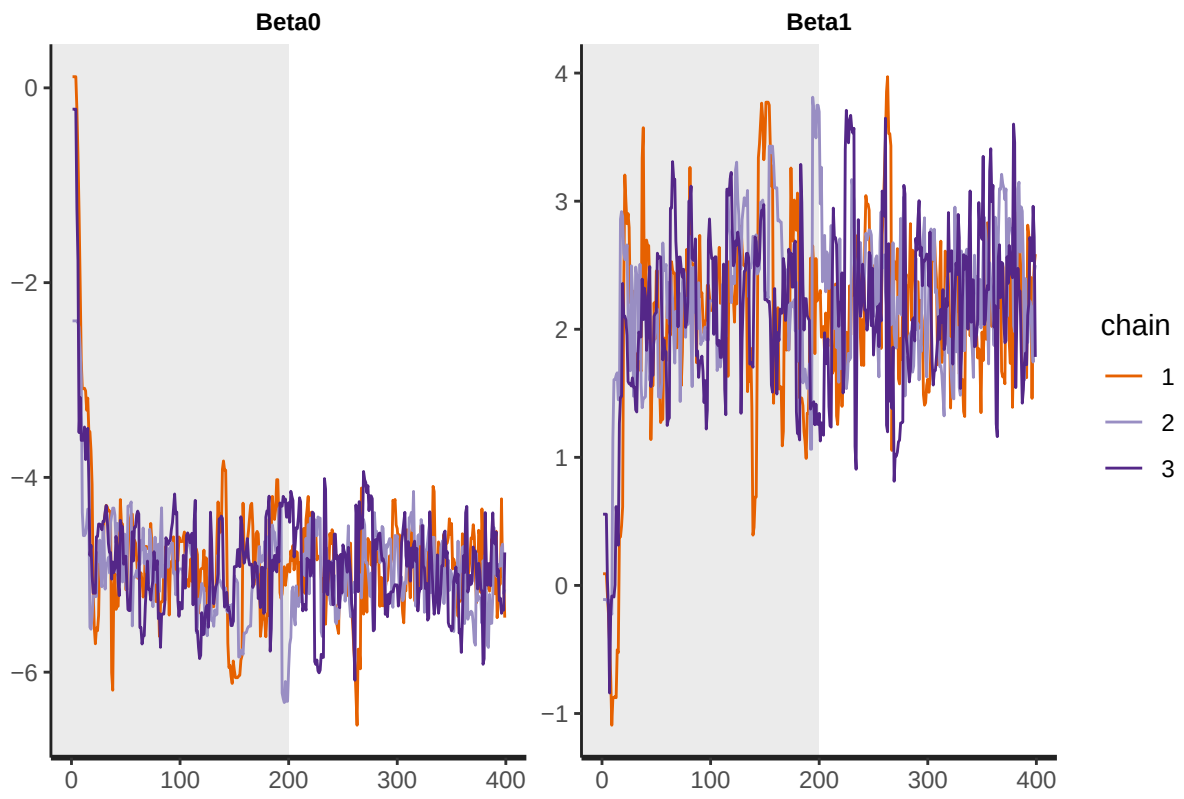
```
## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quanti
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#tail-ess
```

Posterior sampling and autocorrelation

```
print(post)
```

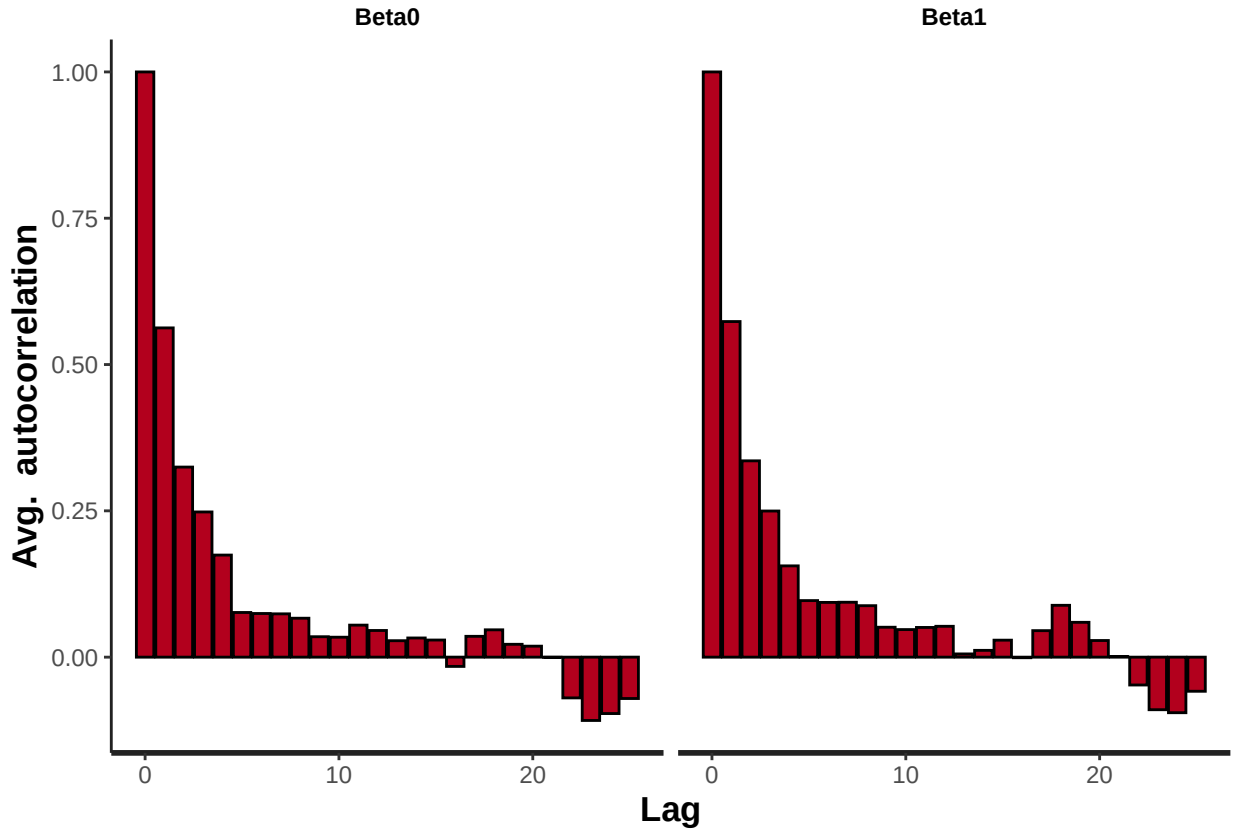
```
## Inference for Stan model: a28ddf62185ea1b1ba7698d94818c07a.
## 3 chains, each with iter=400; warmup=200; thin=1;
## post-warmup draws per chain=200, total post-warmup draws=600.
##
##             mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## Beta0      -4.96    0.03 0.40 -5.82 -5.20 -4.94 -4.67 -4.20   130 1.00
## Beta1       2.23    0.05 0.53  1.25  1.86  2.20  2.57  3.46   121 1.00
## lambda[1]   3.09    0.11 1.21  1.21  2.25  2.90  3.82  6.13   126 1.01
## lambda[2]   0.39    0.01 0.13  0.17  0.30  0.38  0.48  0.72   131 1.01
## lambda[3]   0.96    0.02 0.28  0.49  0.76  0.92  1.13  1.62   139 1.01
## lambda[4]   1.43    0.03 0.36  0.82  1.18  1.39  1.64  2.24   152 1.00
## lambda[5]   2.90    0.04 0.60  1.85  2.46  2.85  3.25  4.16   186 1.00
## lambda[6]   1.98    0.02 0.34  1.39  1.73  1.96  2.20  2.72   259 1.00
## lambda[7]   1.94    0.01 0.29  1.42  1.73  1.92  2.13  2.57   423 1.00
## lambda[8]   4.61    0.02 0.65  3.41  4.17  4.56  5.03  5.94   699 1.00
## lambda[9]  10.85    0.07 1.66  7.74  9.74 10.72 11.91 14.63   637 1.00
## lambda[10] 19.06    0.18 3.44 13.03 16.62 18.60 21.05 26.89   367 1.00
## lp__       44.27    0.08 0.97 41.82 43.96 44.62 44.95 45.21   158 1.01
##
## Samples were drawn using NUTS(diag_e) at Wed Dec 16 04:46:55 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
plot(post,pars=c("Beta0","Beta1"), plotfun="trace", inc_warmup = TRUE)
```




```
stan_ac(post,c("Beta0","Beta1"),inc_warmup = FALSE, lags = 25)
```

```
## Warning: Ignoring unknown parameters: fun.y

## No summary function supplied, defaulting to `mean_se()`
## No summary function supplied, defaulting to `mean_se()`
```



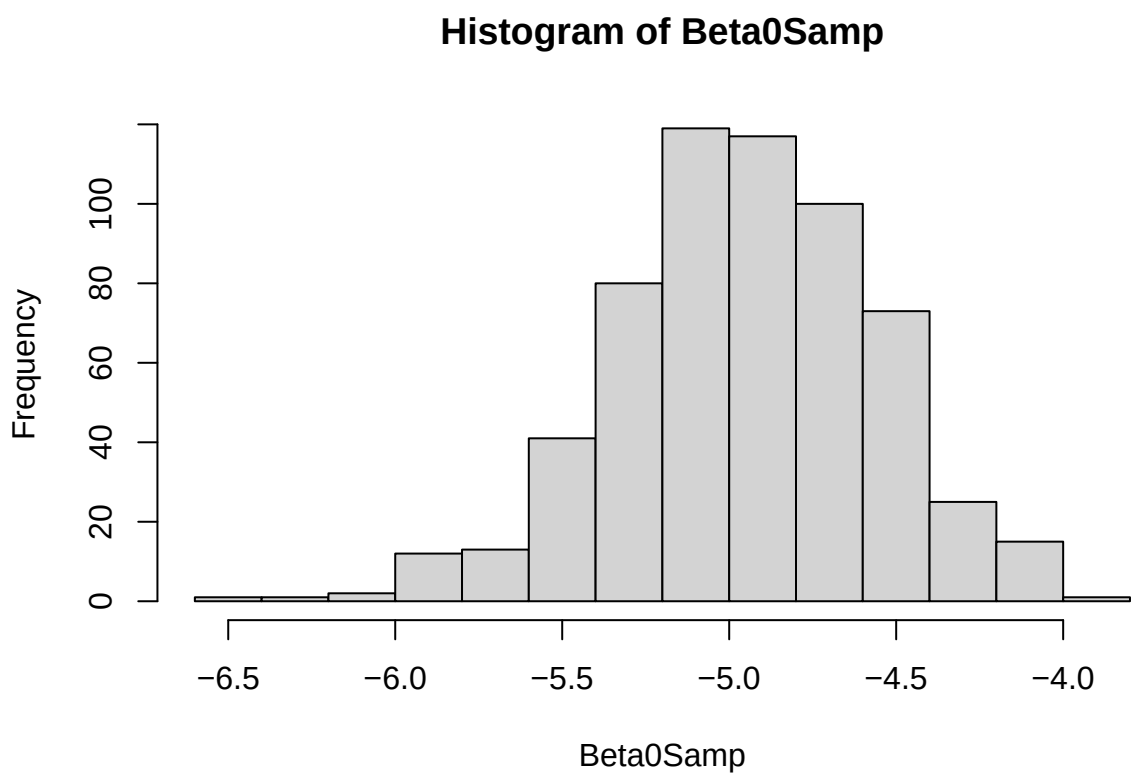
According to the Rhat summary and the plotted trace plots of sample chains the chains seem to have converged and the RHAT is <1.05 therefore we don't have any convergence problem. The autocorrelation of the Markov chain samples becomes very small in small time therefore we don't have any autocorrelation problem.

2. Visualize the posterior for β_0 and β_1 and calculate and report their posterior mean and variance. Calculate the covariance between β_0 and β_1. How does this differ from the prior covariance and why?

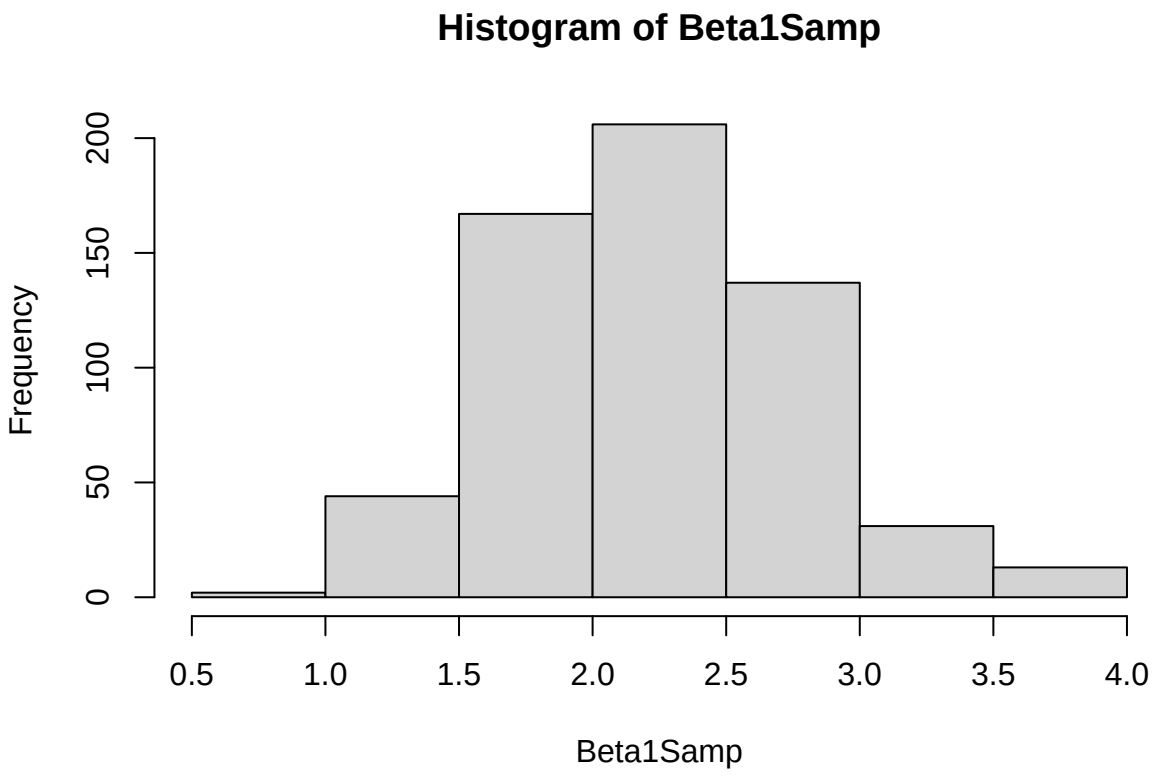
Posteriors of B0 and B1

```
Beta0Samp = as.matrix(post, pars ="Beta0")
Beta1Samp = as.matrix(post, pars ="Beta1")
```

```
hist(Beta0Samp)
```




```
hist(Beta1Samp)
```



mean Beta1

```
mean(Beta1Samp)
```

```
## [1] 2.227878
```

mean Beta0

```
mean(Beta0Samp)
```

```
## [1] -4.955117
```

var Beta1

```
var(Beta1Samp)
```

```
##           Beta1
## Beta1 0.2812269
```

var Beta0

```
var(Beta0Samp)
```

```
##           Beta0
## Beta0 0.1561941
```

cov Beta0 and Beta1

```
cov(Beta0Samp,Beta1Samp)
```

Poesterior

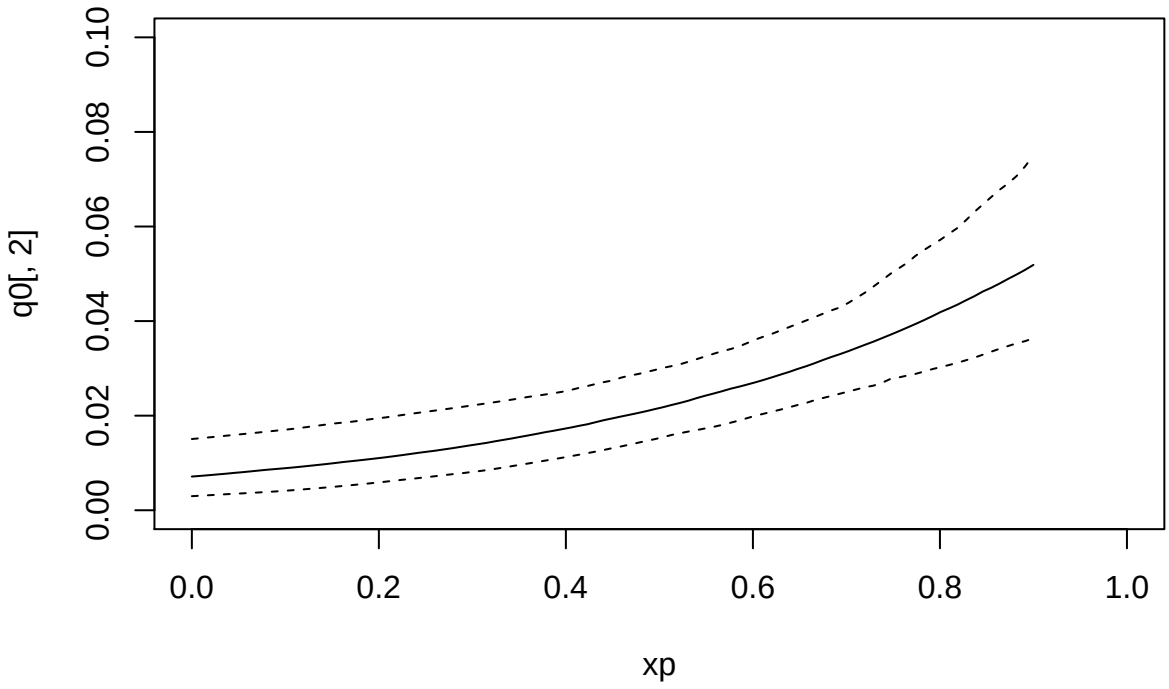
```
##            Beta1
## Beta0 -0.1955776
```

less than 0 (negative) #### Prior Since Beta0 and Beta1 have the same distribution their prior covariance is exactly 0

In the posterior parameters, the likelihood will add covariance between these parameters. This is the reason why the posterior covariance isn't exactly equal 0 but inferior to it unlike the prior one.

3. Visualize the posterior distribution of λ as a function of ice concentration. That is, draw the median and 95% credible interval of the prediction function at ice concentrations (0,0.1],...,(0.9,1].

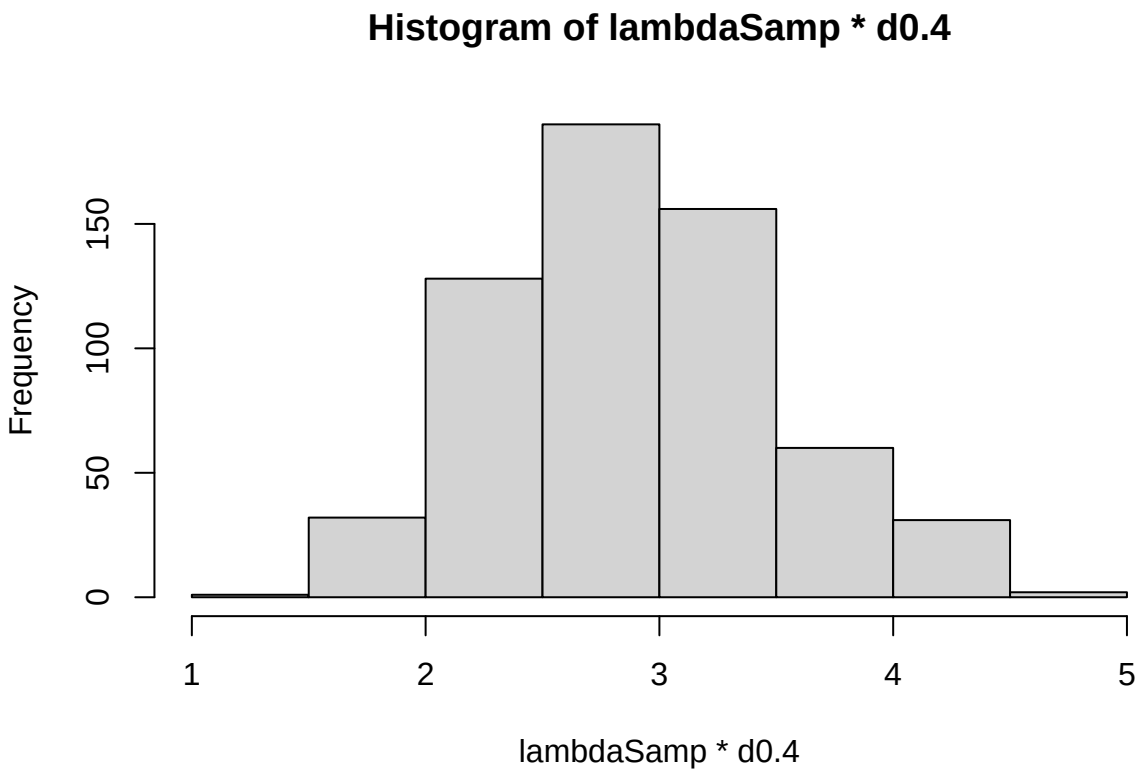
```
xp = seq(min(x), max(x), length=101)
q0 = matrix(nrow = 101,ncol=3)
# the evaluation points
for (i in 1:length(xp)) {
        f0 = Beta0Samp + Beta1Samp * xp[i]

th0 = exp(f0)
q0[i,] = quantile(th0,probs = c(0.025,0.5,0.975))
}
#plot(x,y/n)
plot(xp,q0[,2], type="l", col="black",xlim=c(0,1),ylim= c(0,0.1)) # mean
lines(xp,q0[,1], lty=2, col="black")
# 95% interval of f
lines(xp,q0[,3], lty=2, col="black")
```



4. Visualize the posterior distribution of $\lambda \times d$ and the posterior predictive distribution of $\tilde{y}$ in ice concentration (0.4,0.5] where $\tilde{y}$ is a predicted number of besetting events in $d = 200\ 000nm$. How do these differ and why?

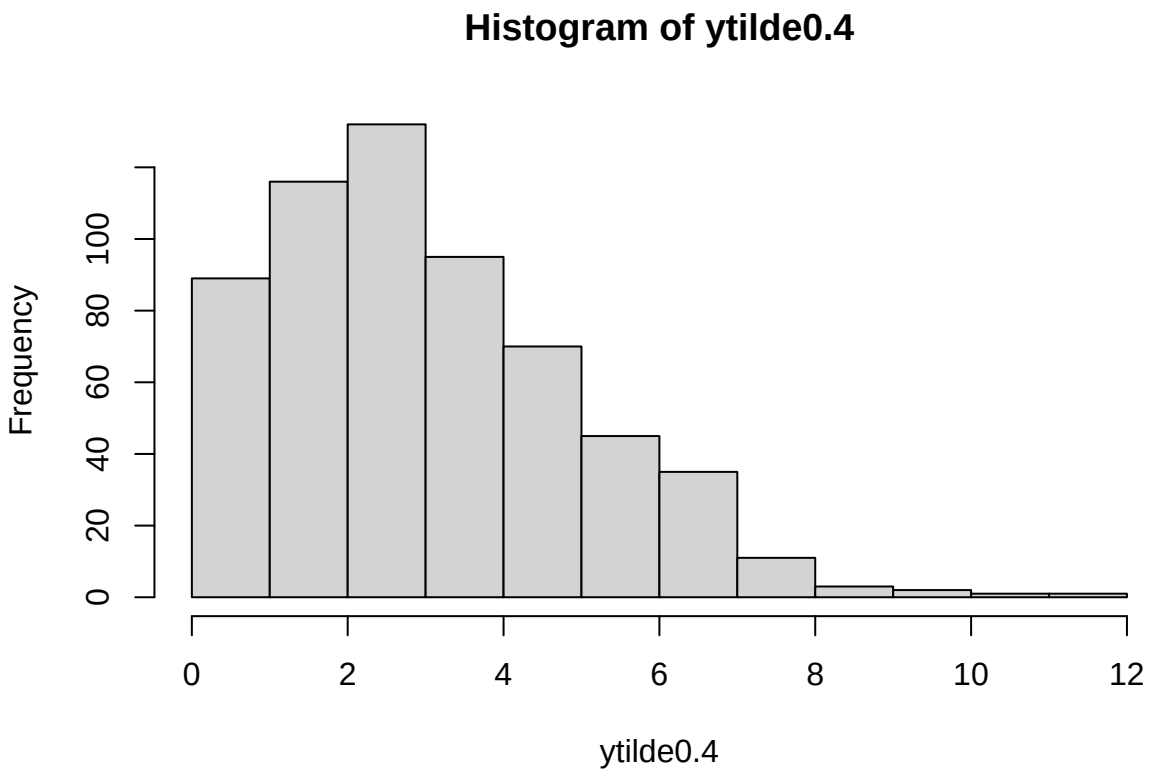
distribution of lambda * d with d = 164.952

```
d0.4 = data.icebesetting$distance..x1000nm.[data.icebesetting$Minimum.ice.concentration == 0.4]

lambdaSamp = exp(Beta0Samp + Beta1Samp * 0.4)  # lambda for x = 0.4
hist(lambdaSamp * d0.4)
```



distribution of y tilde with d = 200

```
ytilde0.4 = rep(0,length(lambdaSamp))
for (i in 1:length(ytilde0.4)) {
        ytilde0.4[i] = rpois(1,lambdaSamp[i] * 200)
}

hist(ytilde0.4)
```



Both graphs have the same mode however the first one is more symetric around the mode while the other one is more biased to the left. The reason behind this difference is that y tilde follows a poisson distribution while lambda d is a direct simulation and caculated by multiplying lambda with the distance

5. Calculate the *posterior predictive probability* that there will be a besetting event during $\tilde{d} = 200\ 000$nm in ice concentration $\tilde{x} = (0.4, 0.5]$. That is calculate

$$\Pr(\text{besetting event within distance } \tilde{d}|\tilde{x}, y, d, x)$$

where y, d, x collect all the data.

In order to get the posterior predictive probability we have 2 methods that could be used. The first one is by taking the predicted y vector that corresponds for the given interval and the given distance and check the portion of ytilde that are bigger or equal to 1. Therefore:

```
sum(ytilde0.4 >= 1) / length(ytilde0.4)
```

```
## [1] 0.97
```

The second method is by applying the formula and for this we will take the average of the lambda samples vector and use it in the formula. I prefer the first method this is why i will use this one in order to verify my answer and see if it gives close value.

```
1-exp(-200*mean(lambdaSamp))
```

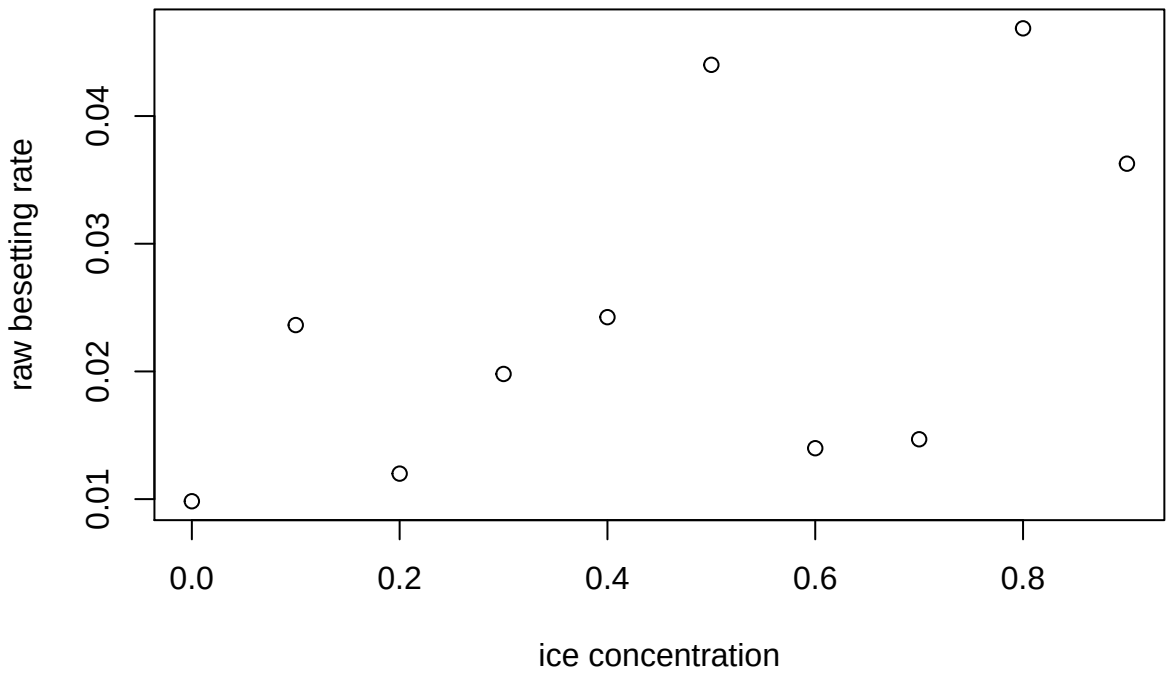
```
## [1] 0.9701759
```

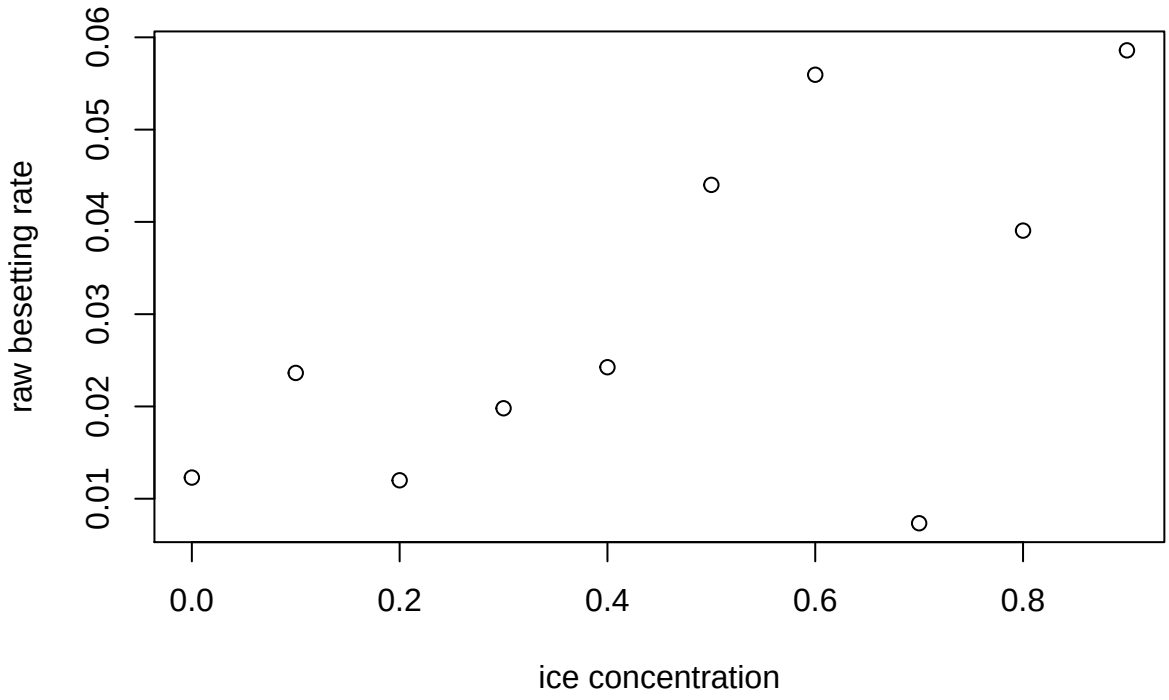
The answer is very close to the first one which is a good sign.

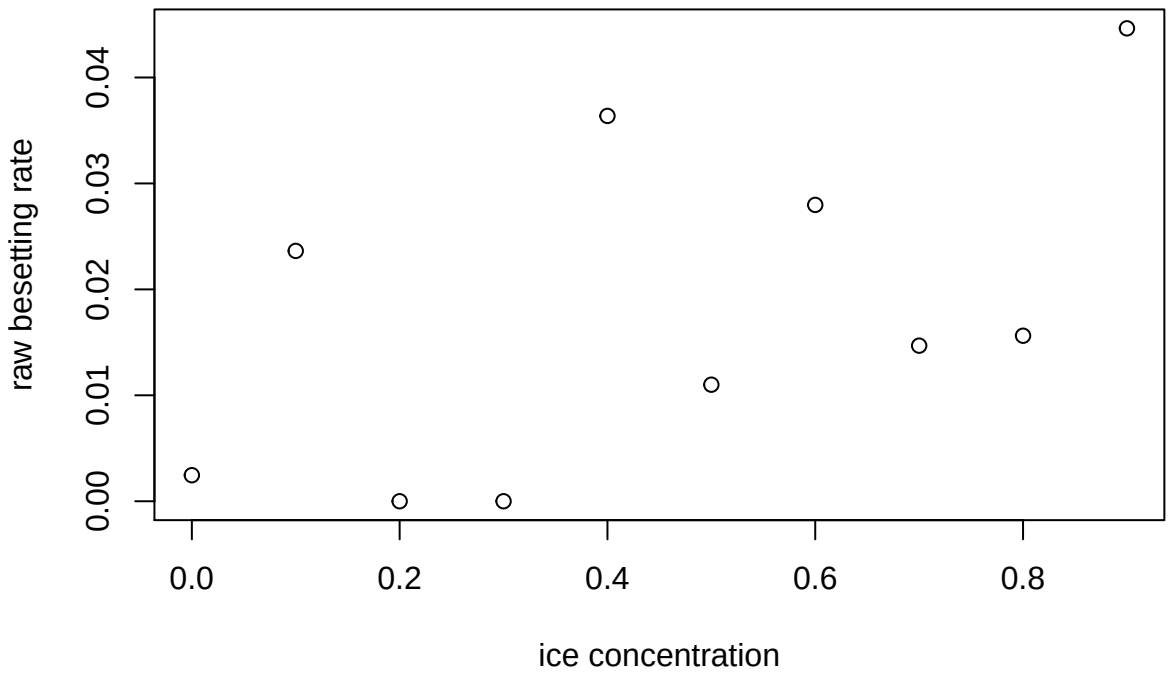
6. Conduct a posterior predictive check for your model by drawing 20 replicate data sets from your model and comparing their histogram and raw besetting rate vs ice concentration plot to those of the training data. Discuss the results of the posterior predictive check.

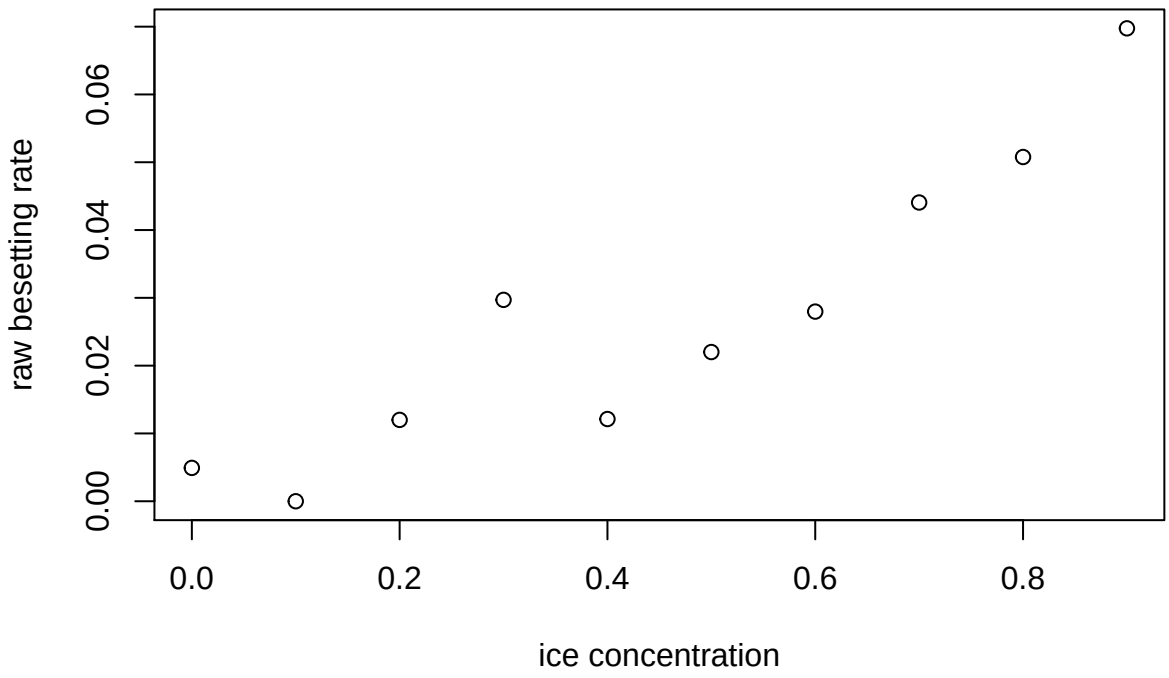
I am sorry for not using par(mfrow) for the plots but i am getting a strange error once i add it.

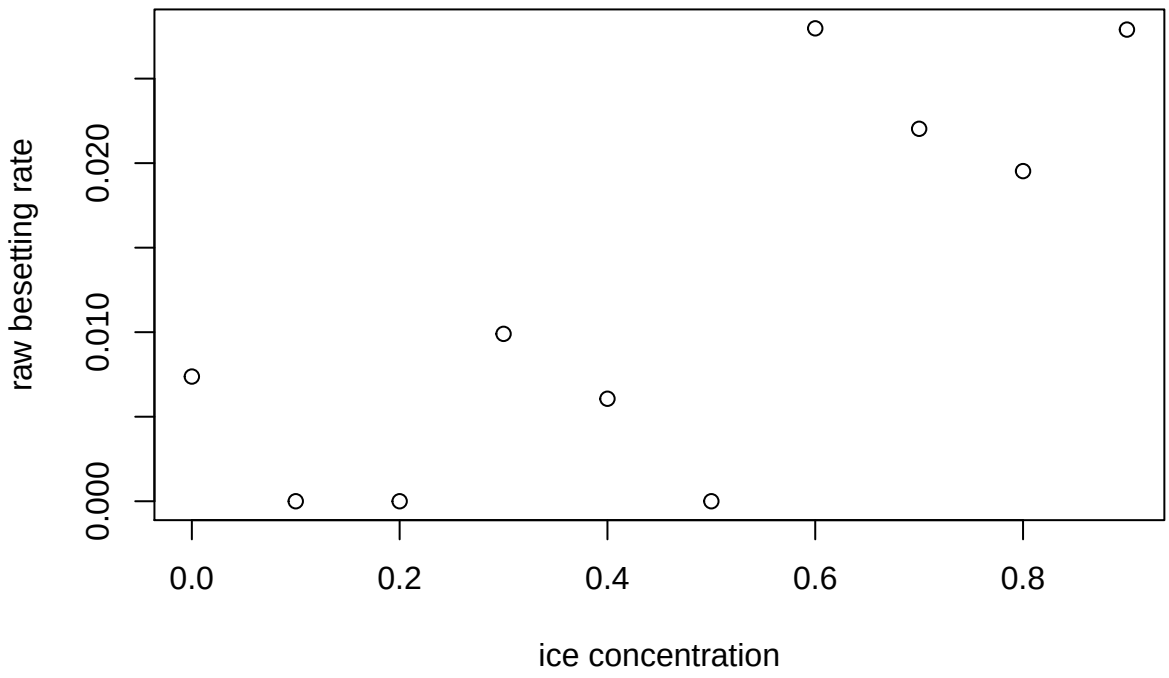
```
lambdaSampMatrix = as.matrix(post, pars =c("lambda[1]","lambda[2]","lambda[3]",
"lambda[4]","lambda[5]","lambda[6]","lambda[7]","lambda[8]","lambda[9]","lambda[10]"))


ytilde0.4[i] = rpois(1,lambdaSamp[i] * 200)
YTILDE = matrix(0L, nrow = 20, ncol =10)

#20 samples for every x and put them in a matrix
for (i in 1:20) {
  for (j in 1:10) {
  YTILDE[i,j] = rpois(1,lambdaSampMatrix[i,j] )
 }
}

for (i in 1:20) {
plot(data.icebesetting$Minimum.ice.concentration,
YTILDE[i,]/data.icebesetting$distance..x1000nm.,
ylab="raw besetting rate", xlab="ice concentration")
}
```

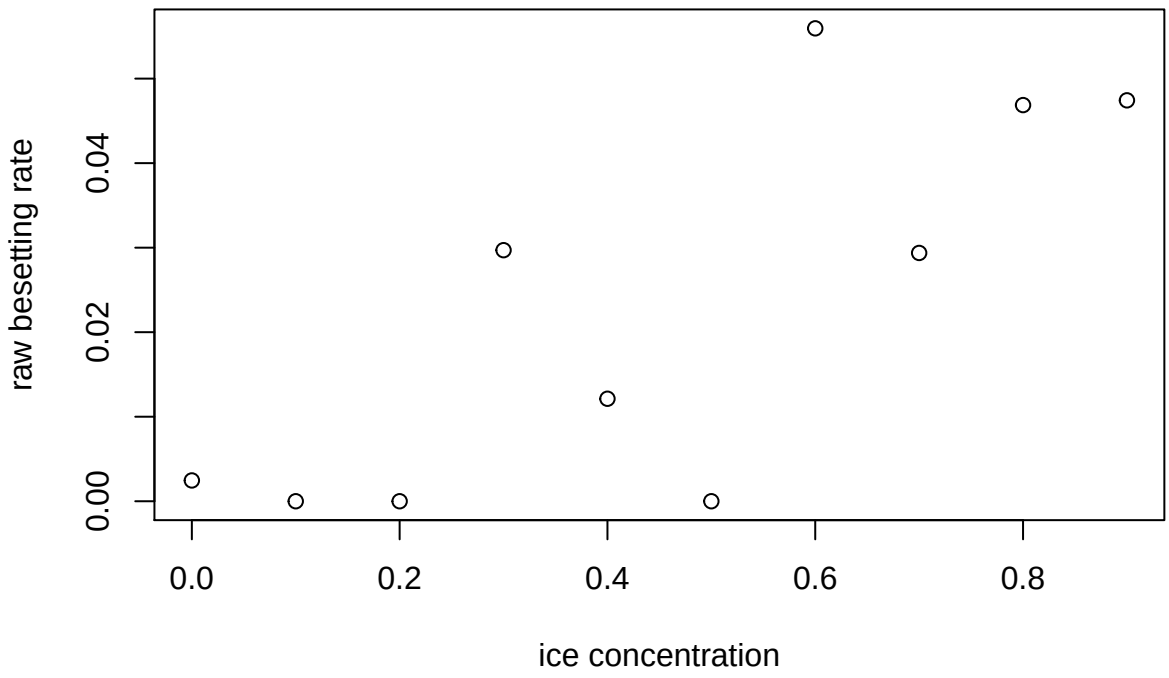

raw besetting rate
0.01
0.02
0.03
0.04
0.0
0.2
0.4
0.6
0.8
ice concentration

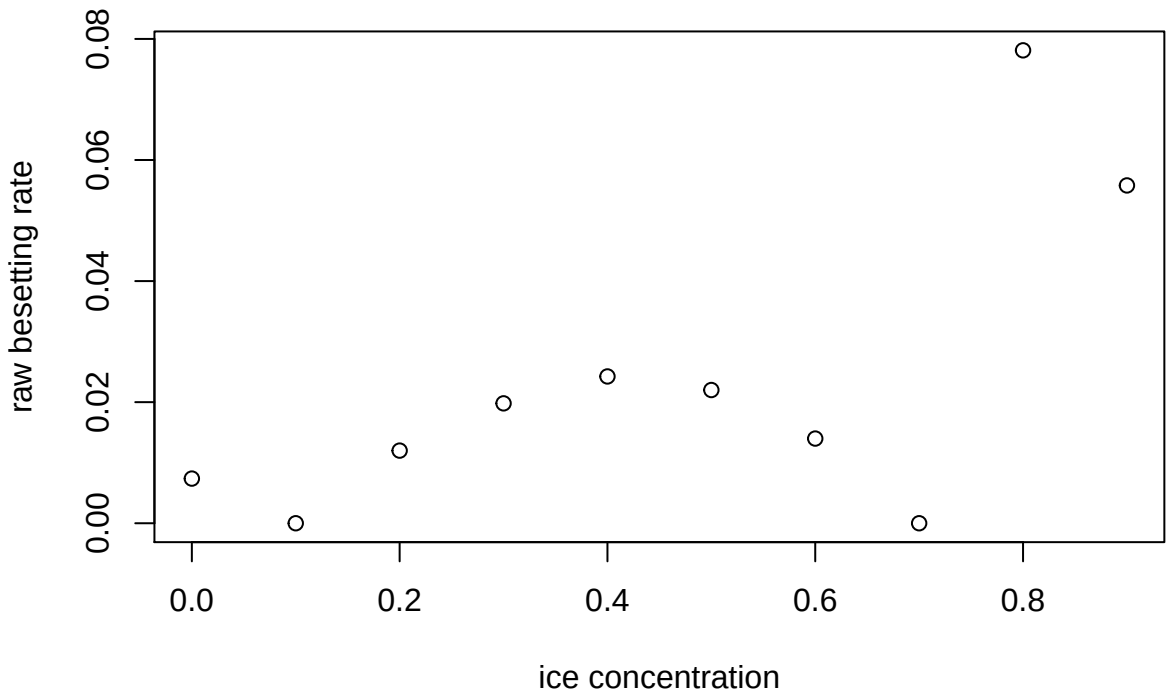

raw besetting rate
0.01
0.02
0.03
0.04
0.05
0.06
0.0
0.2
0.4
0.6
0.8
ice concentration

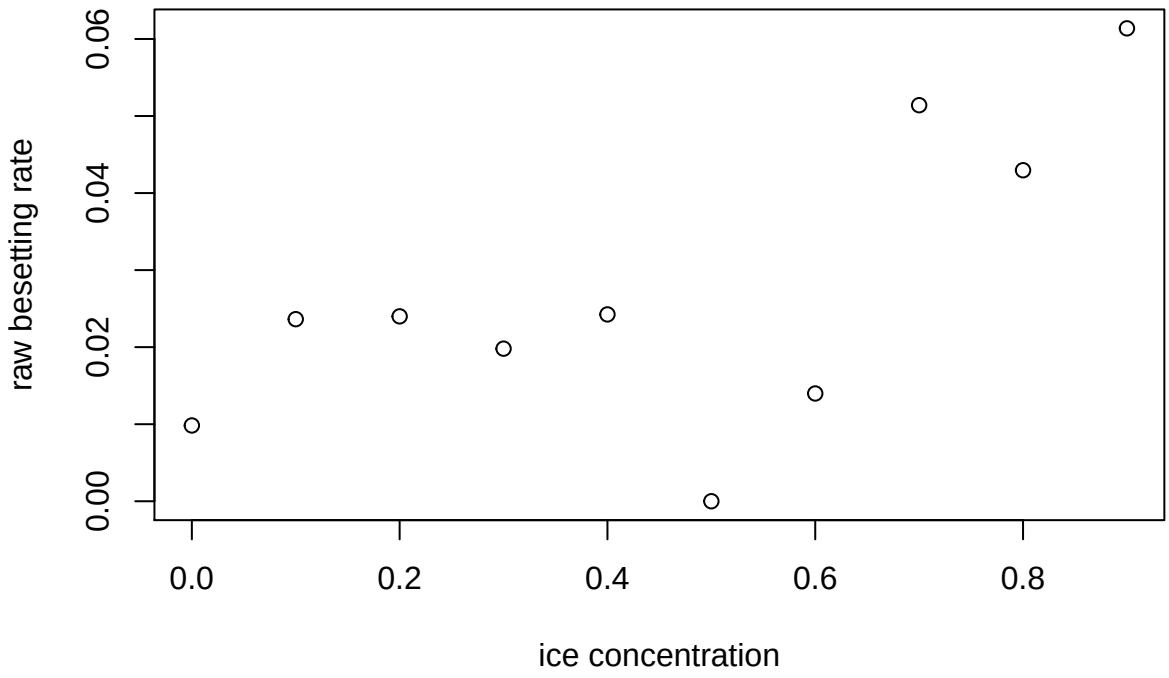

raw besetting rate
0.00
0.01
0.02
0.03
0.04
0.0
0.2
0.4
0.6
0.8
ice concentration

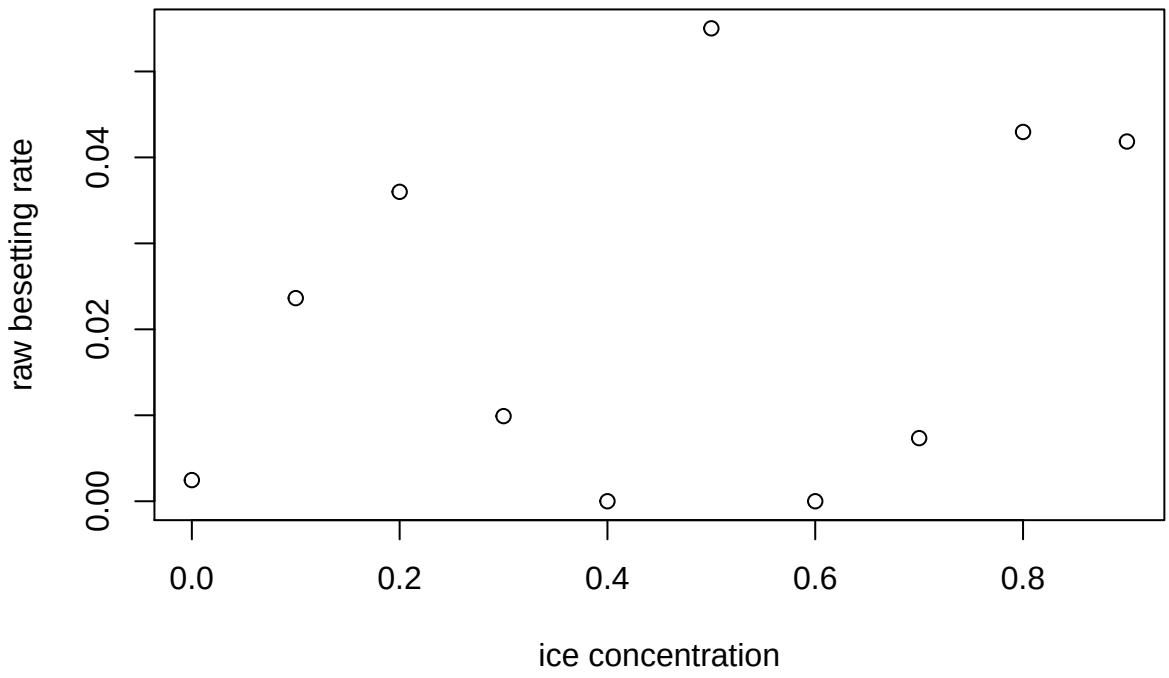

raw besetting rate
0.00
0.02
0.04
0.06
0.0
0.2
0.4
0.6
0.8
ice concentration

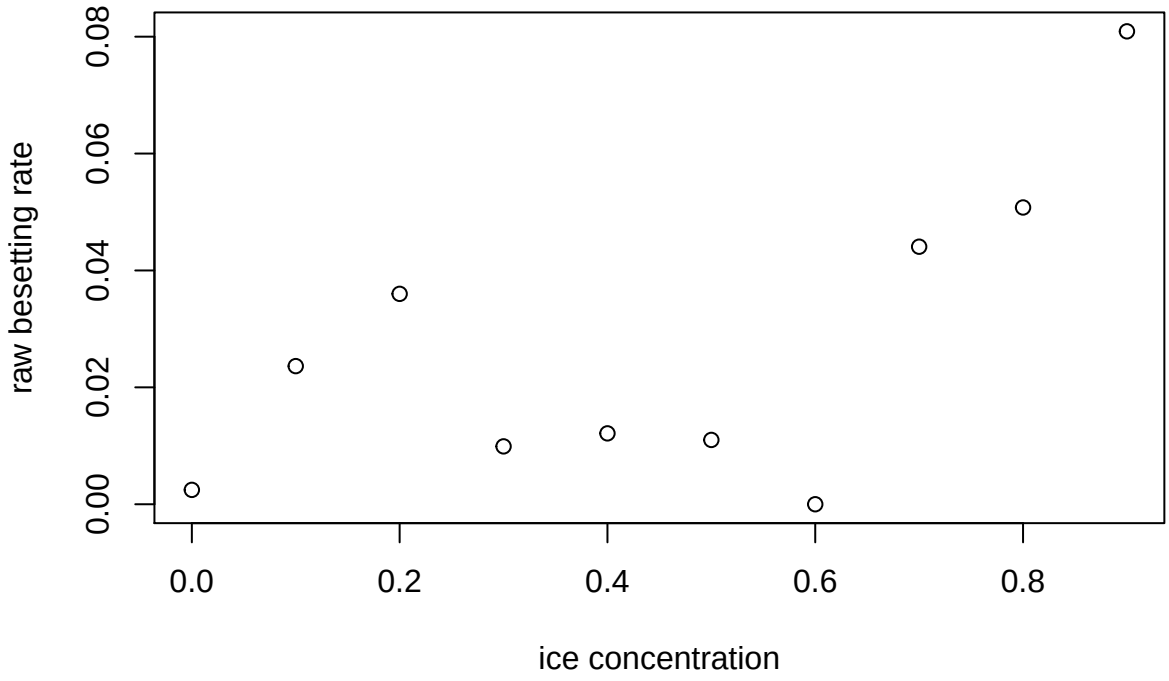

raw besetting rate
0.000
0.010
0.020
0.0
0.2
0.4
0.6
0.8
ice concentration

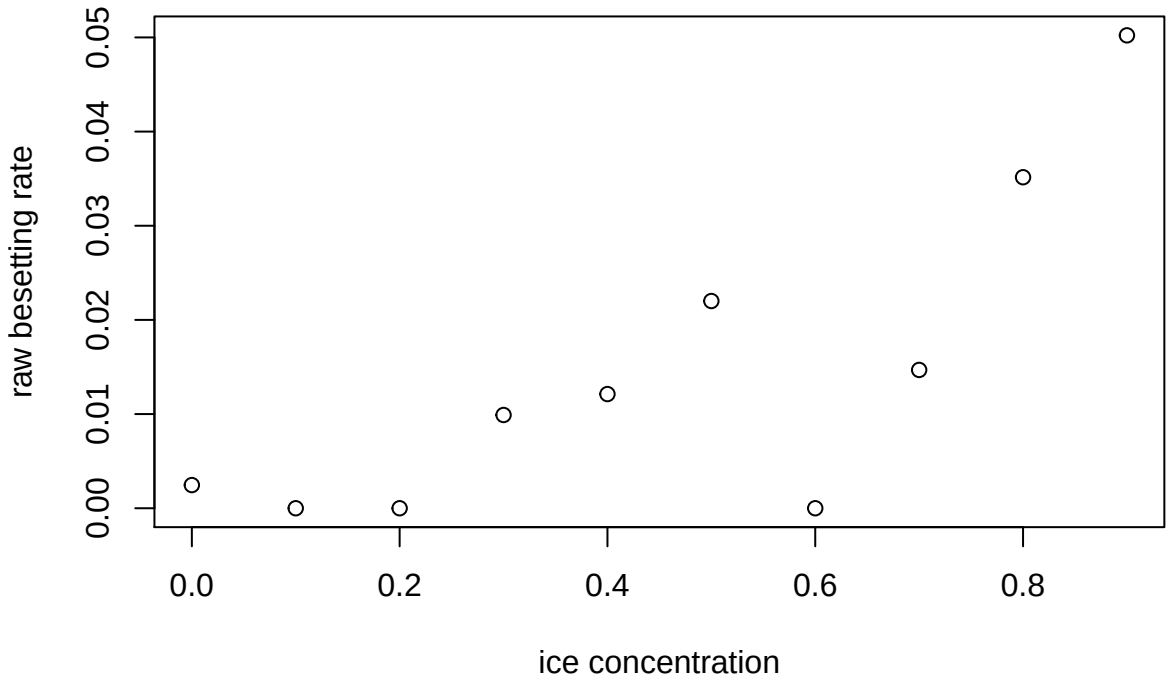

raw besetting rate
0.00
0.02
0.04
0.0
0.2
0.4
0.6
0.8
ice concentration

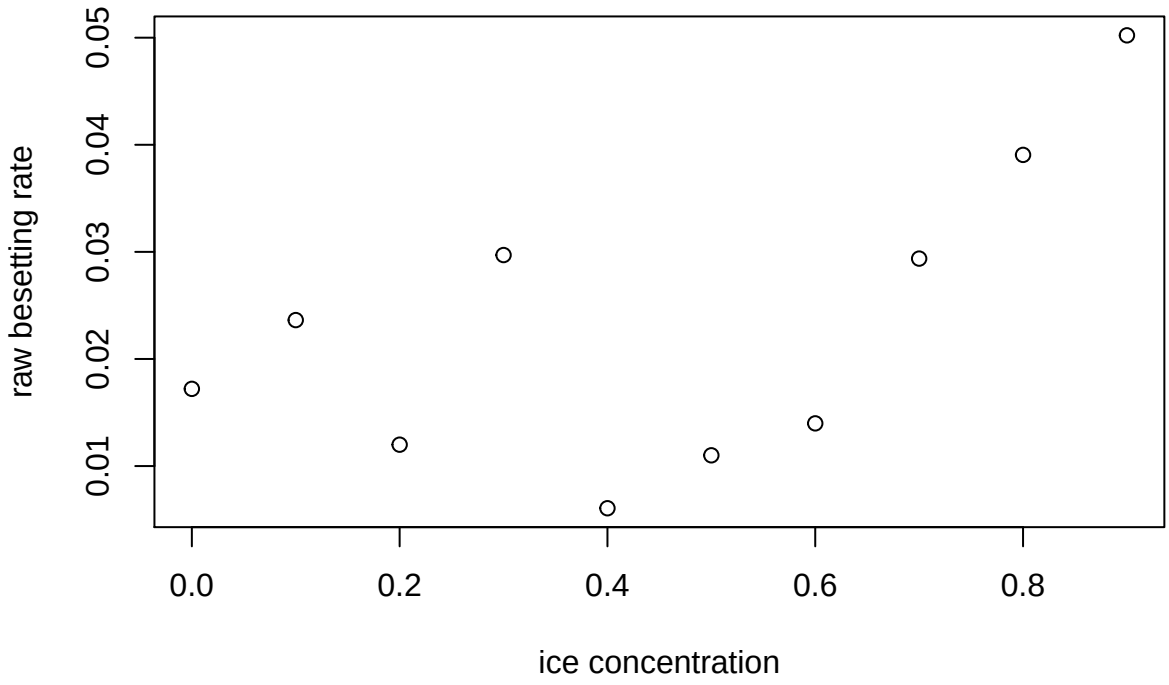

raw besetting rate
0.00
0.02
0.04
0.06
0.08
0.0
0.2
0.4
0.6
0.8
ice concentration

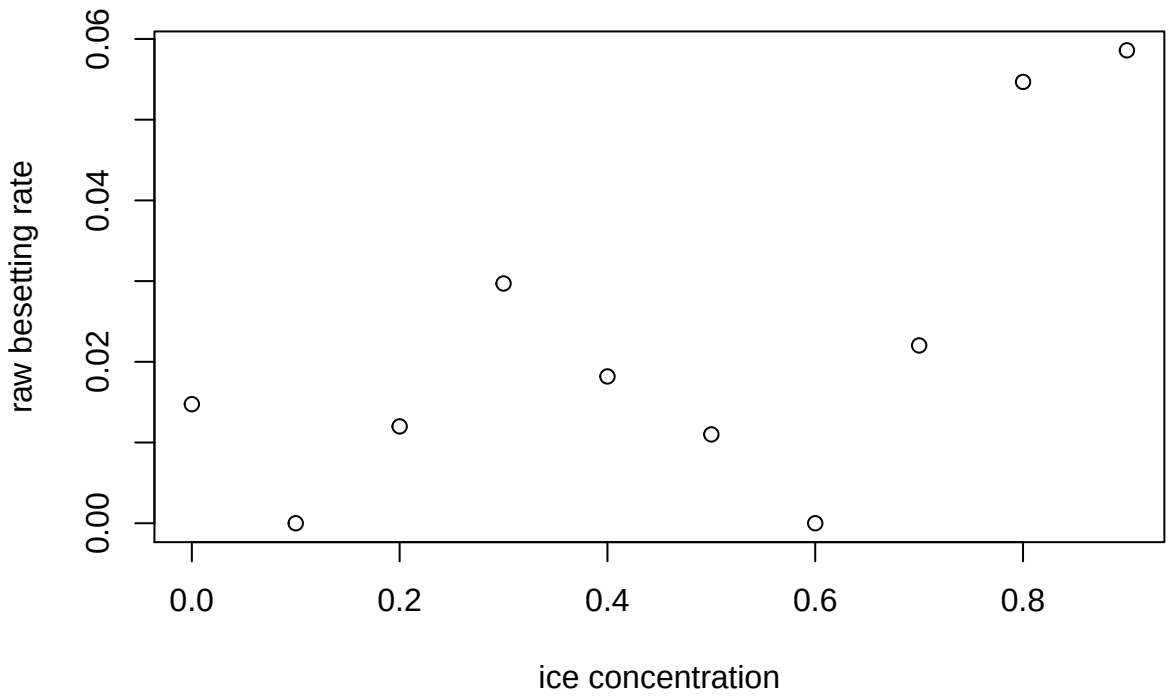

raw besetting rate
0.00
0.02
0.04
0.06
0.0
0.2
0.4
0.6
0.8
ice concentration

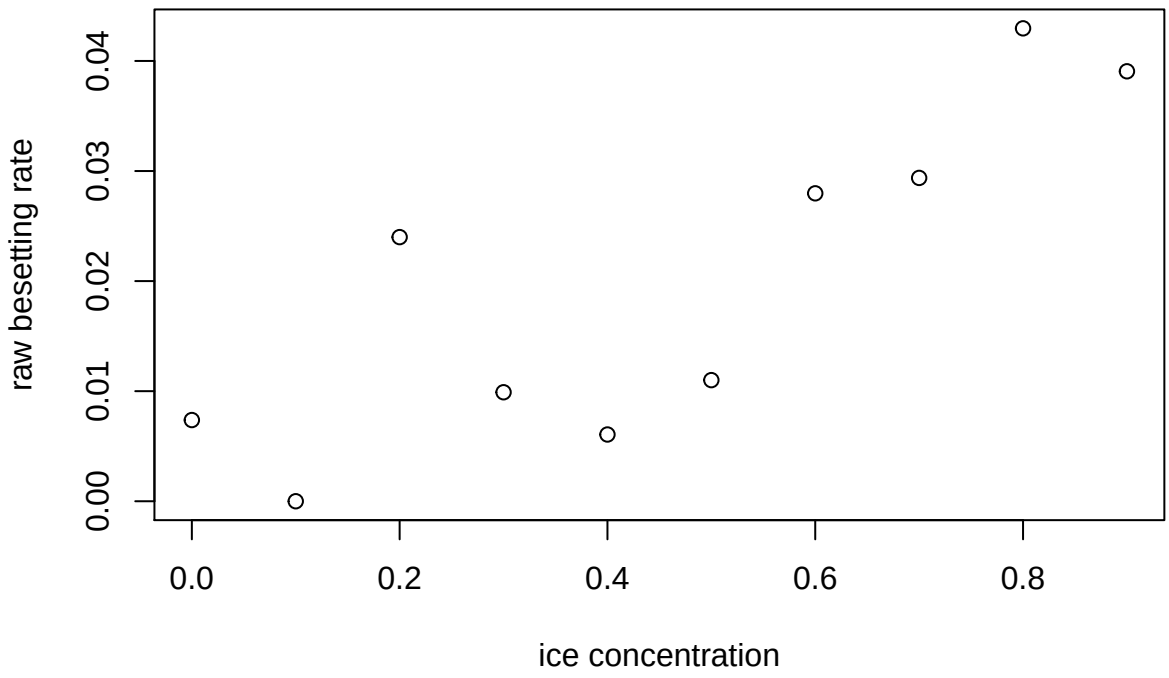

raw besetting rate
0.00
0.02
0.04
0.0
0.2
0.4
0.6
0.8
ice concentration

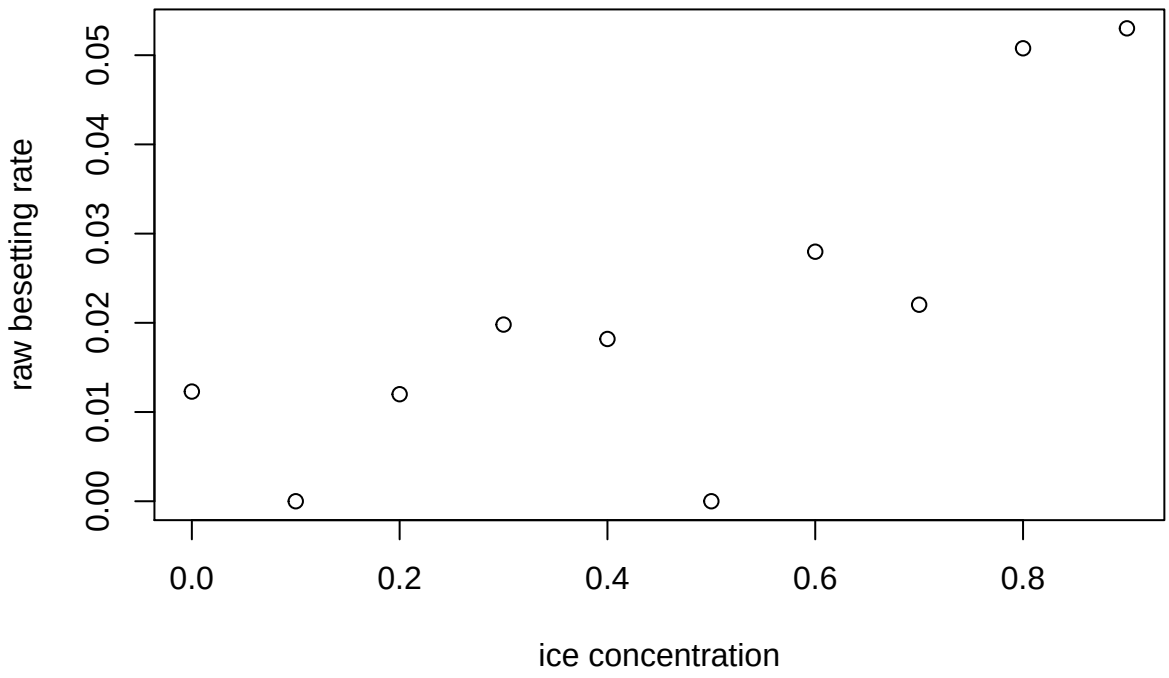

raw besetting rate
0.00
0.02
0.04
0.06
0.08
0.0
0.2
0.4
0.6
0.8
ice concentration

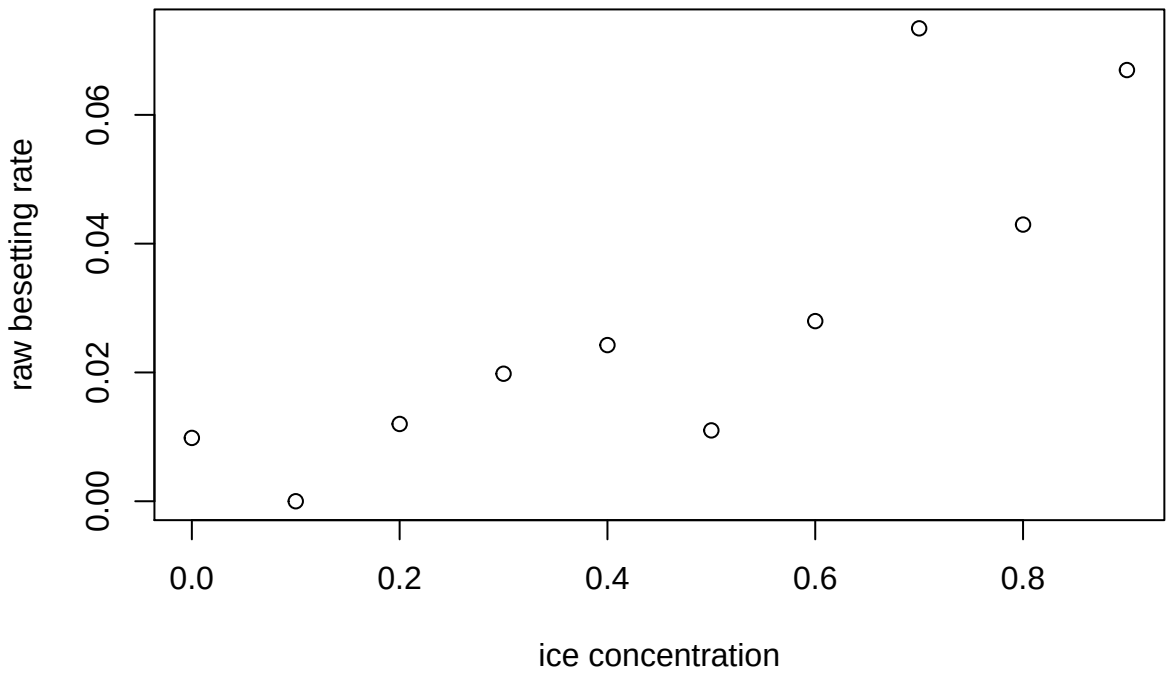

raw besetting rate
0.00
0.01
0.02
0.03
0.04
0.05
0.0
0.2
0.4
0.6
0.8
ice concentration

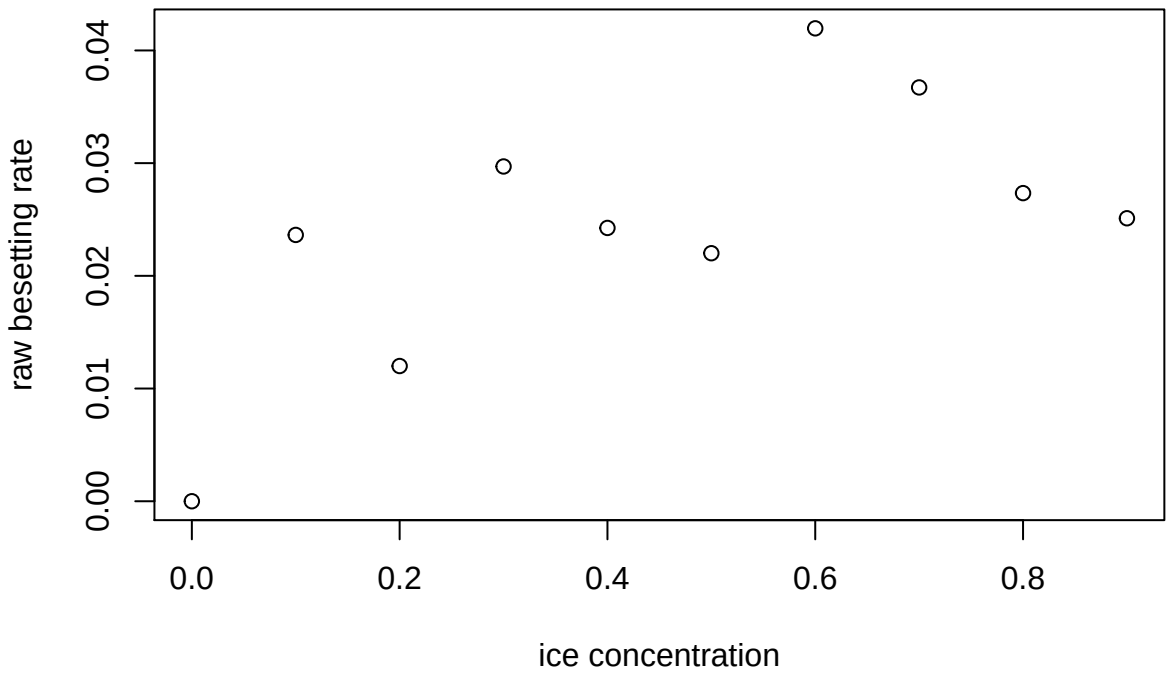

raw besetting rate
0.01
0.02
0.03
0.04
0.05
0.0
0.2
0.4
0.6
0.8
ice concentration

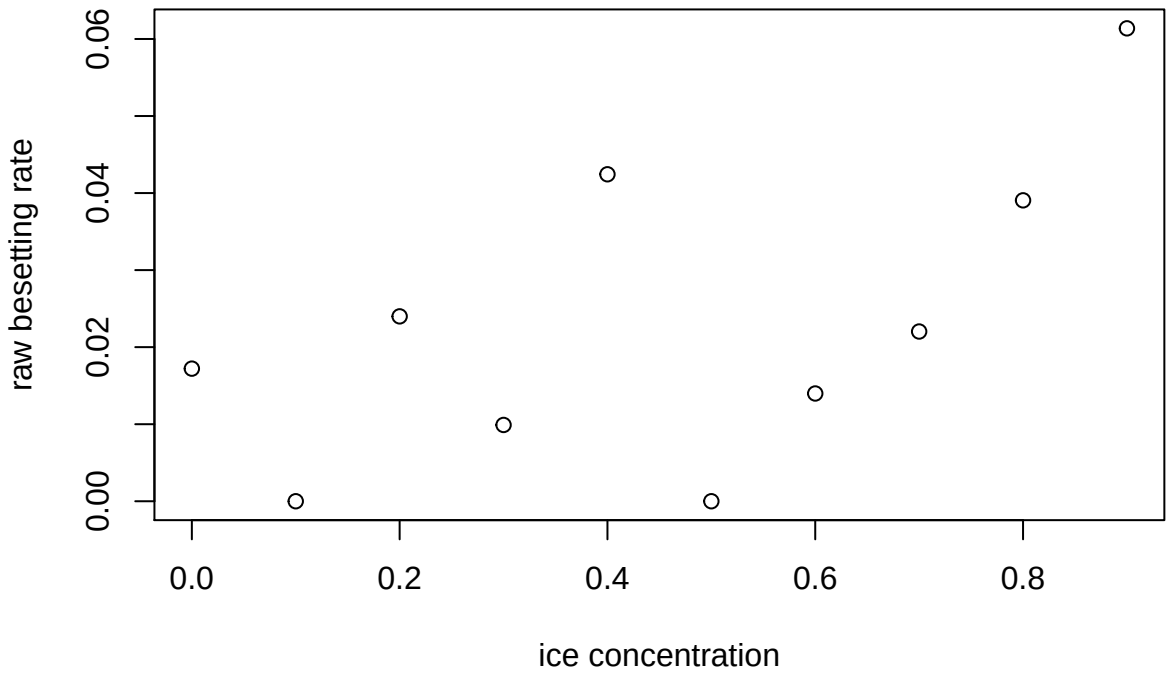

raw besetting rate
0.00
0.02
0.04
0.06
0.0
0.2
0.4
0.6
0.8
ice concentration

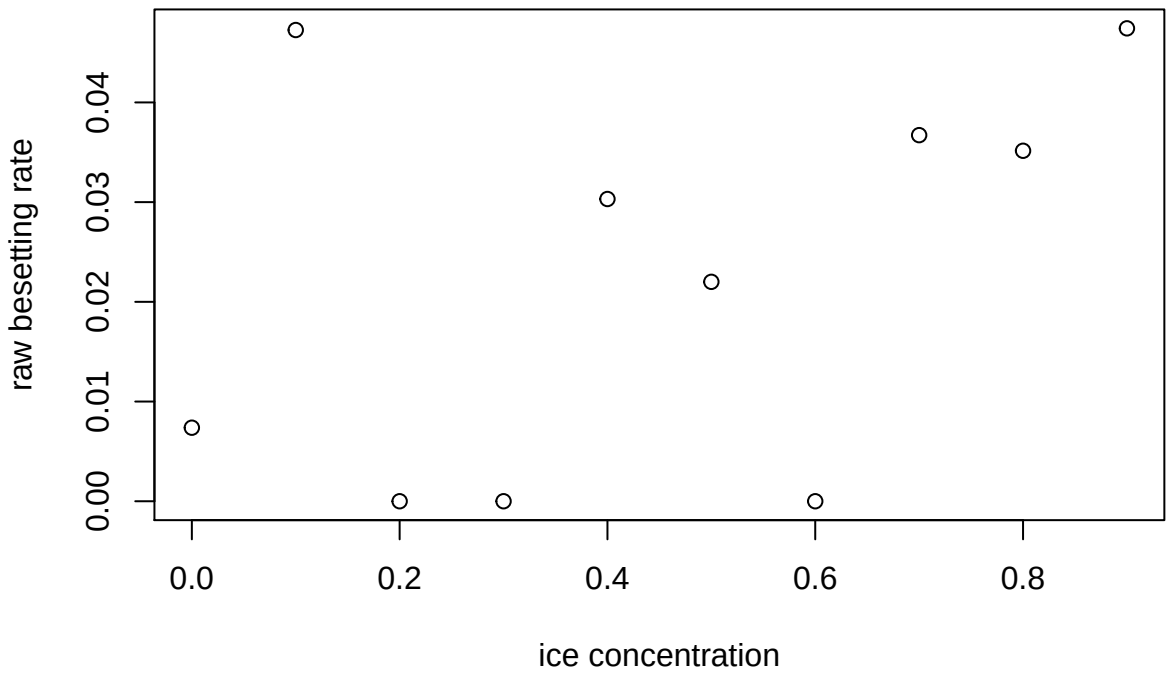

raw besetting rate
0.00
0.01
0.02
0.03
0.04
0.0
0.2
0.4
0.6
0.8
ice concentration

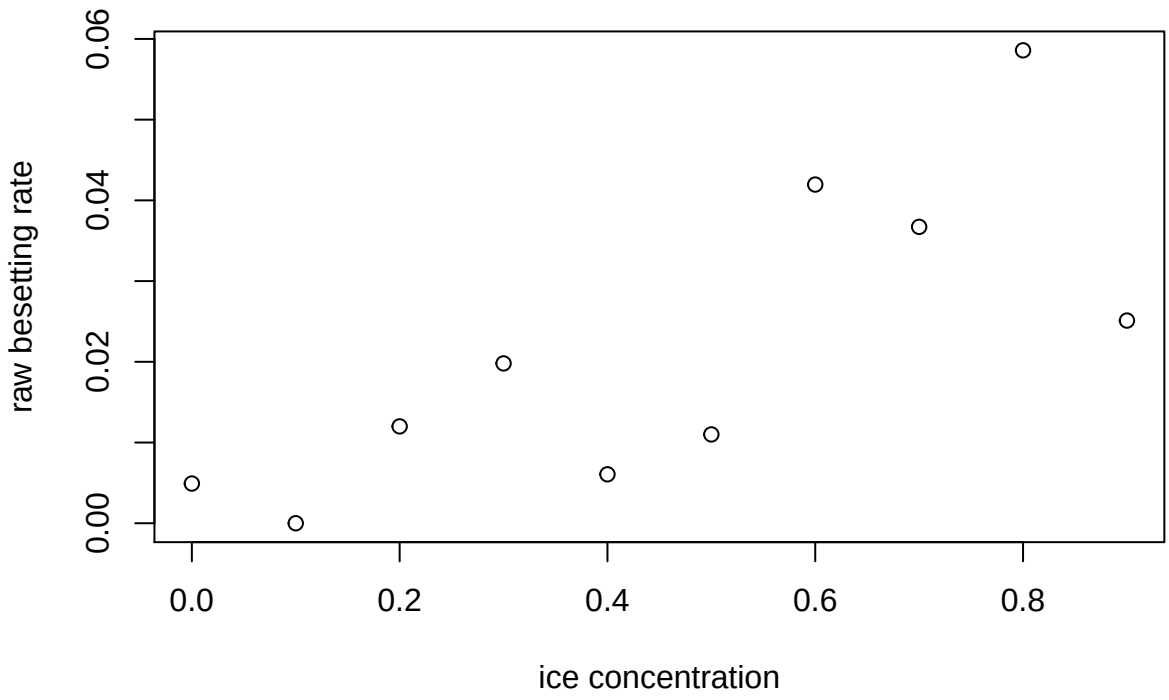

raw besetting rate
0.00
0.01
0.02
0.03
0.04
0.05
0.0
0.2
0.4
0.6
0.8
ice concentration


raw besetting rate
0.00
0.02
0.04
0.06
0.0
0.2
0.4
0.6
0.8
ice concentration

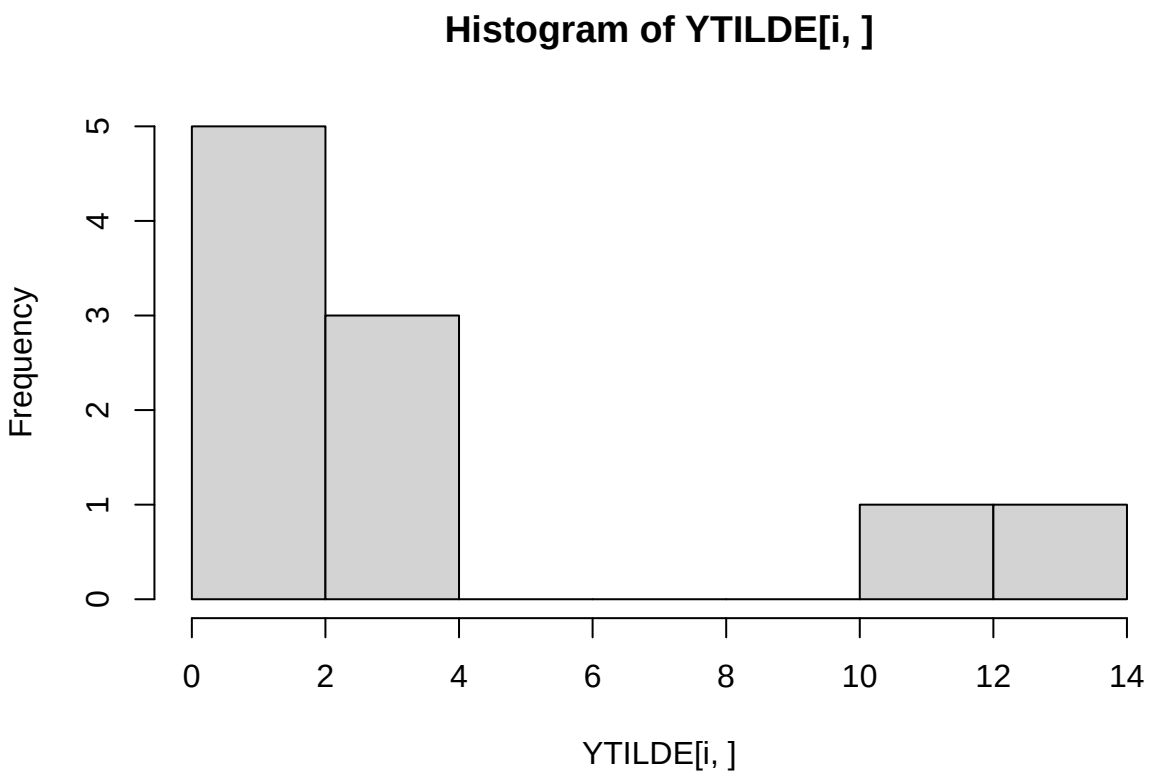

raw besetting rate
0.00
0.01
0.02
0.03
0.04
0.0
0.2
0.4
0.6
0.8
ice concentration

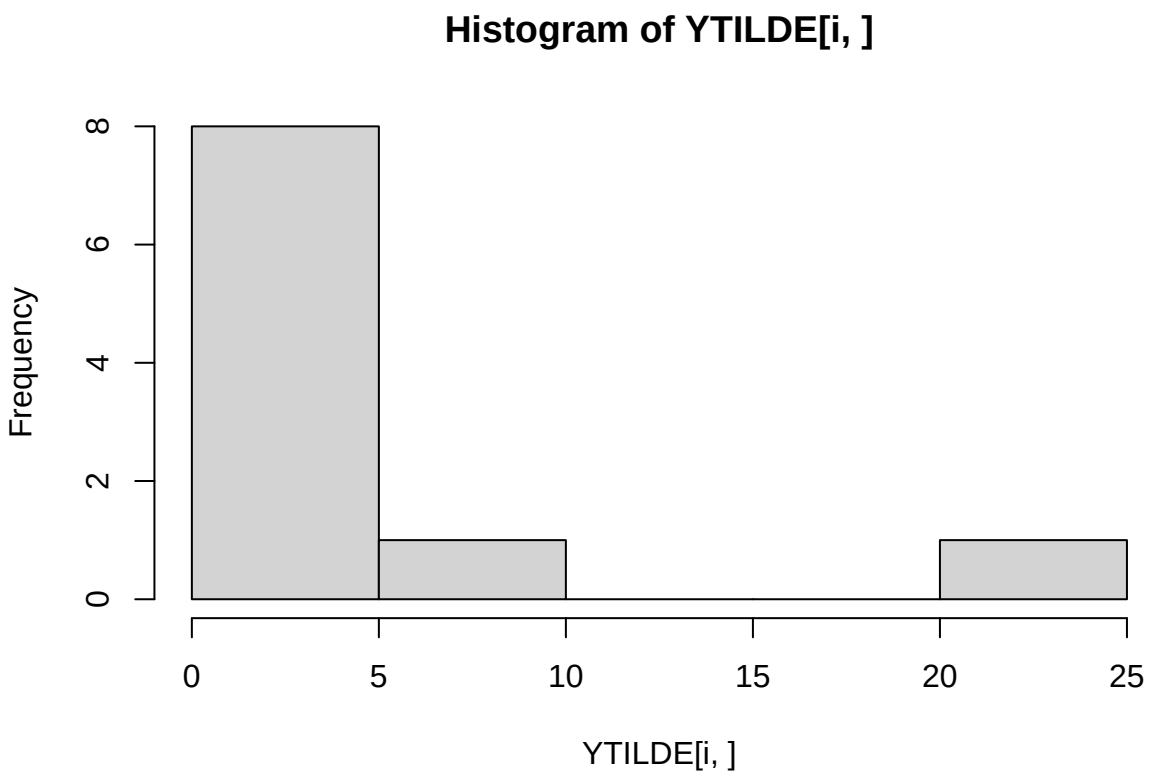

raw besetting rate
0.00
0.02
0.04
0.06
0.0
0.2
0.4
0.6
0.8
ice concentration

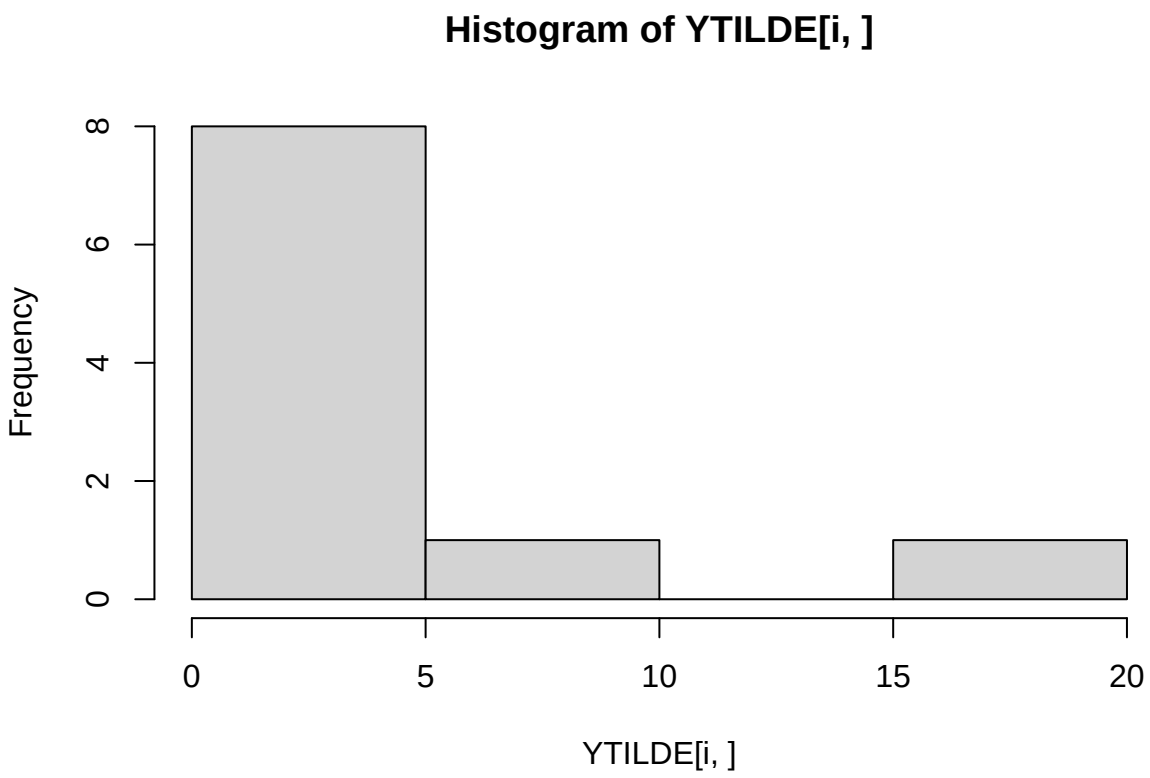

raw besetting rate
0.00
0.01
0.02
0.03
0.04
0.0
0.2
0.4
0.6
0.8
ice concentration

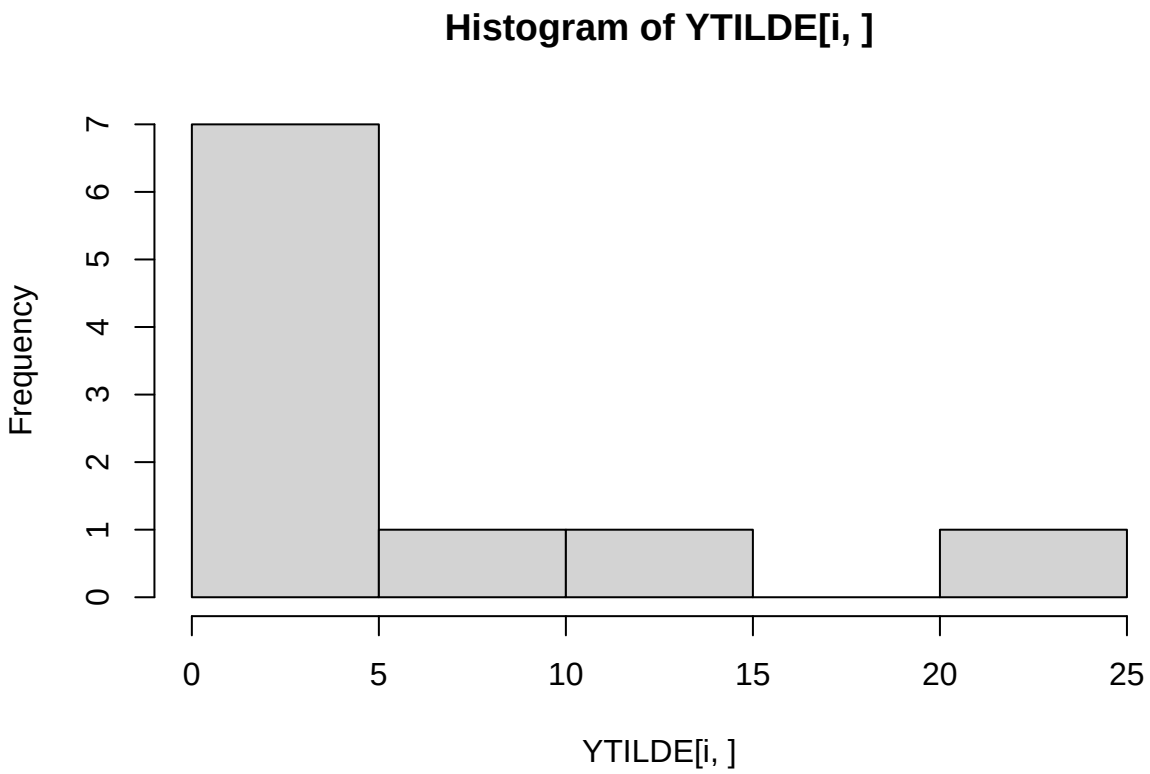


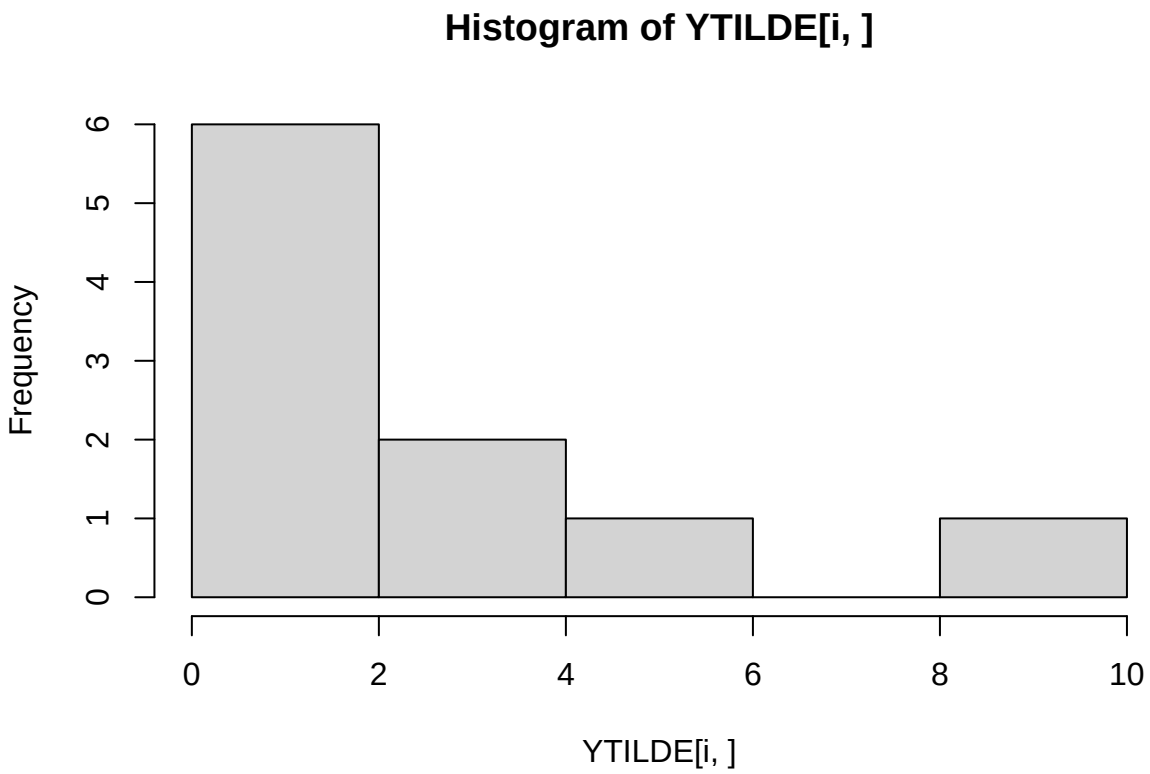

```
for (i in 1:20) {
hist(YTILDE[i,])
}
```

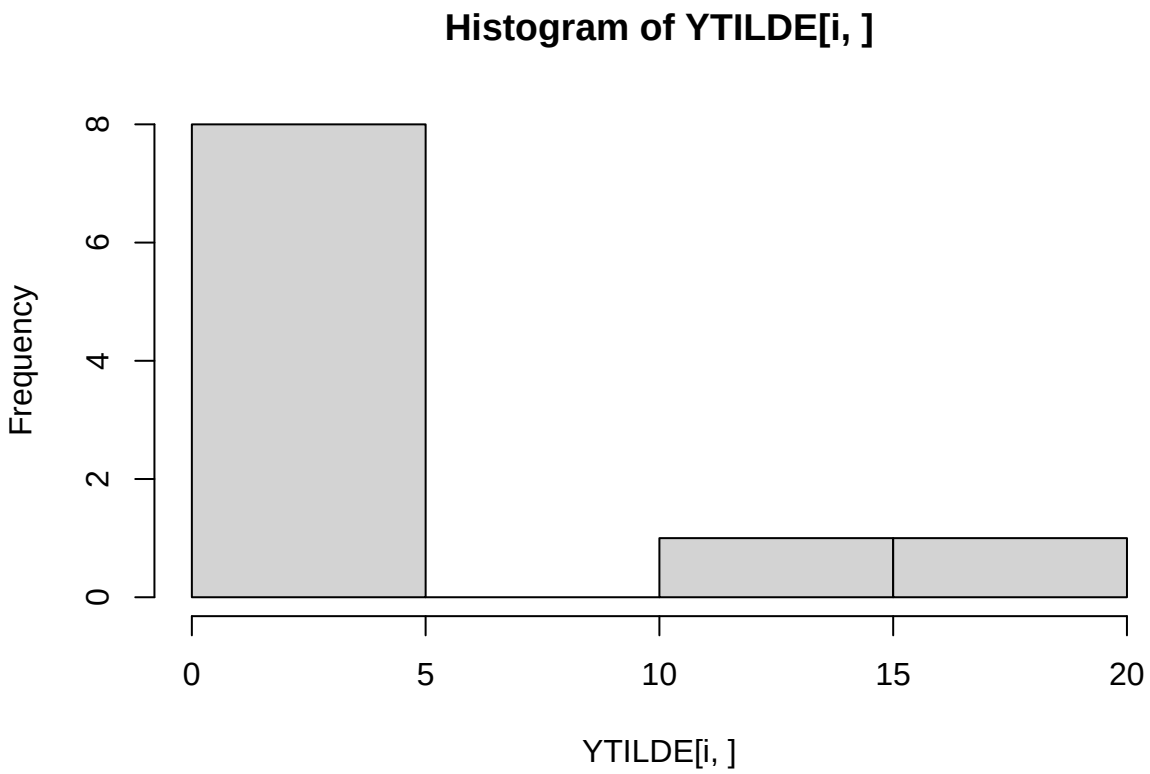

Histogram of YTILDE[i,]
Frequency
0
1
2
3
4
5
0
2
4
6
8
10
12
14
YTILDE[i,]

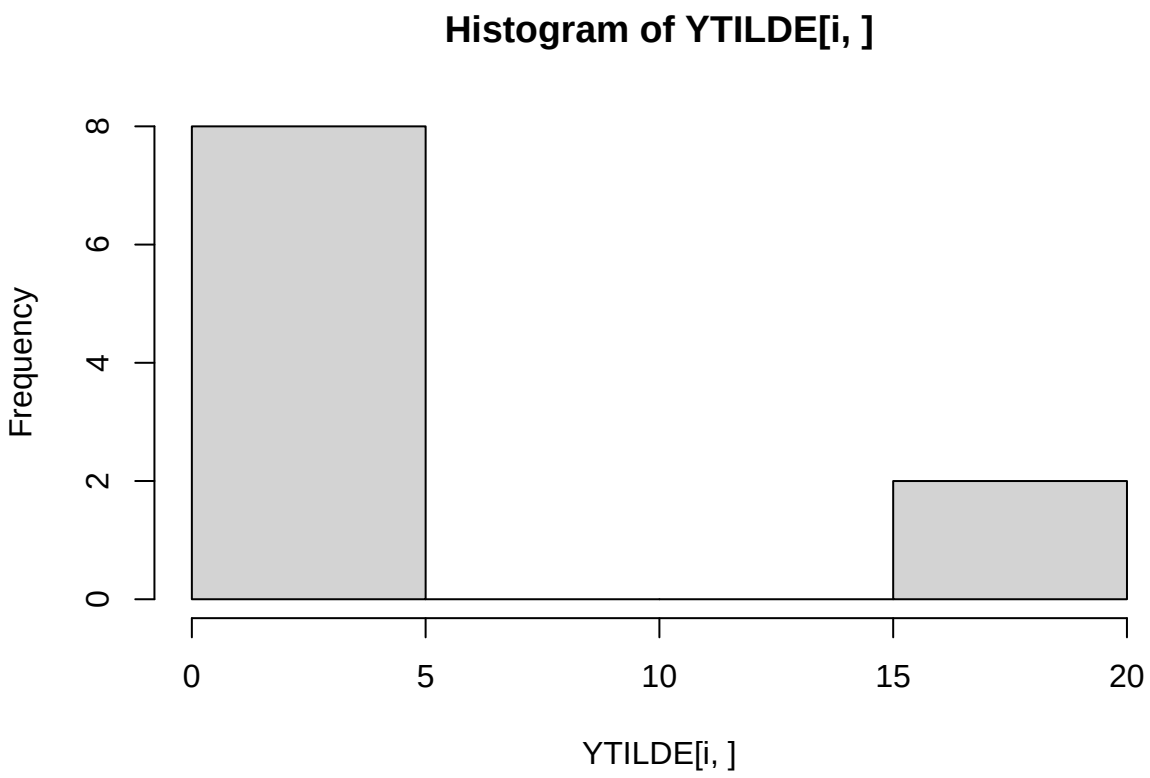

Histogram of YTILDE[i,]
Frequency
0
2
4
6
8
0
5
10
15
20
25
YTILDE[i,]

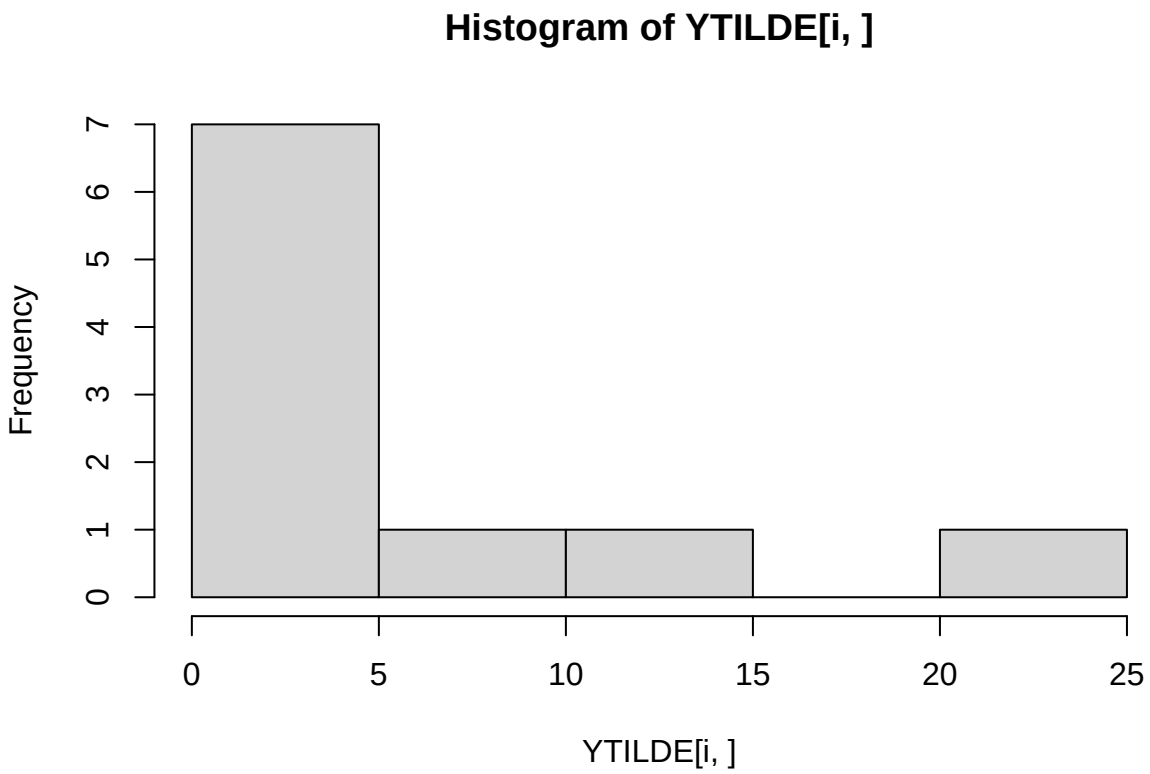

Histogram of YTILDE[i,]
Frequency
0
2
4
6
8
0
5
10
15
20
YTILDE[i,]

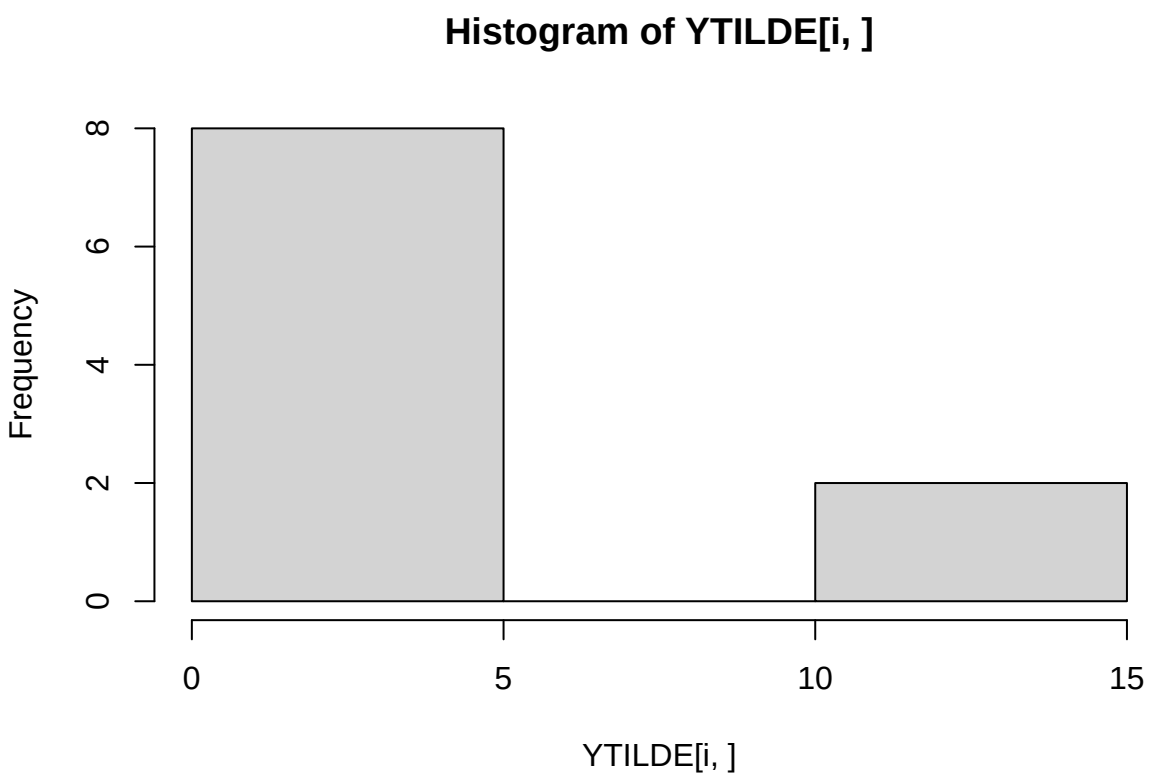

Histogram of YTILDE[i,]
Frequency
0
1
2
3
4
5
6
7
0
5
10
15
20
25
YTILDE[i,]

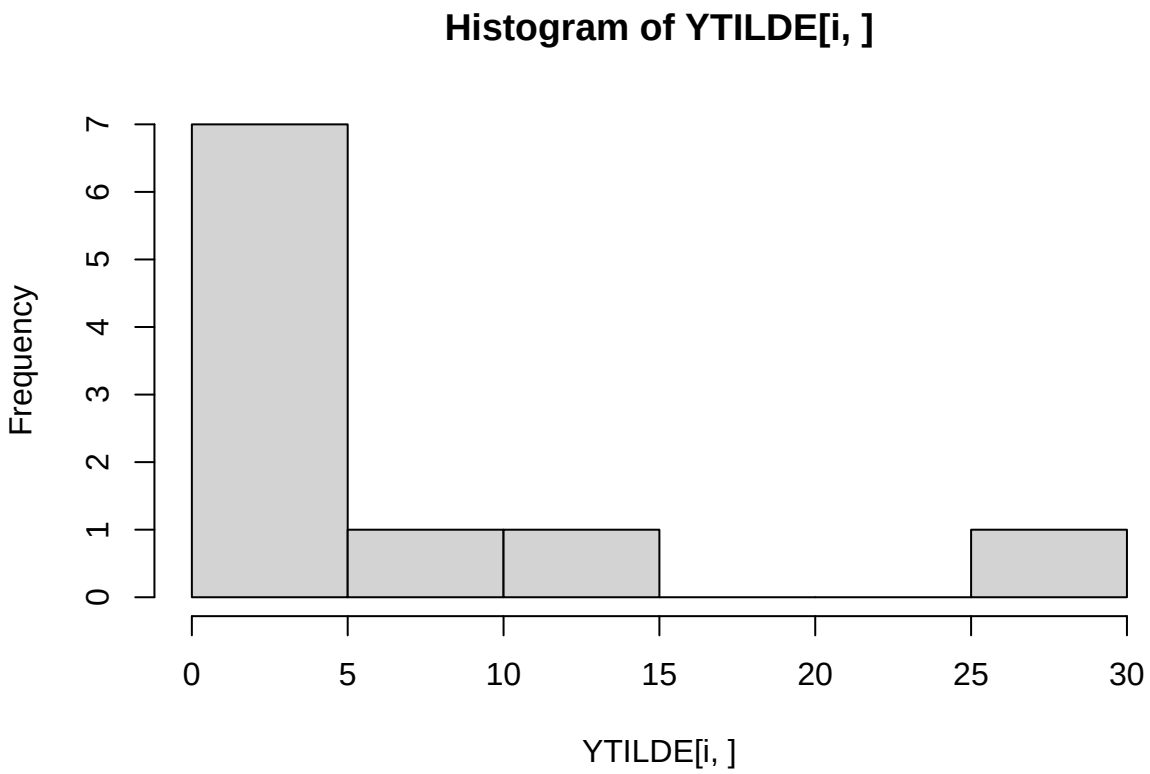

Histogram of YTILDE[i,]
Frequency
0
1
2
3
4
5
6
0
2
4
6
8
10
YTILDE[i,]

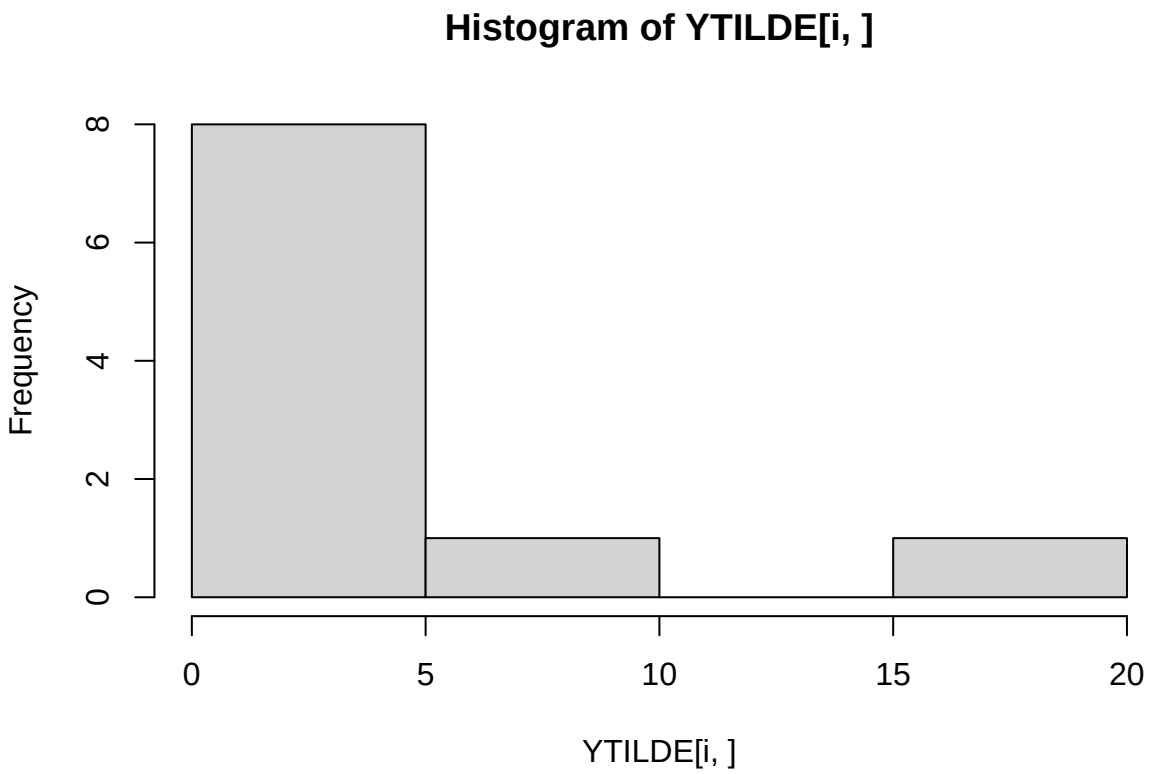

Histogram of YTILDE[i,]
Frequency
0
2
4
6
8
0
5
10
15
20
YTILDE[i,]

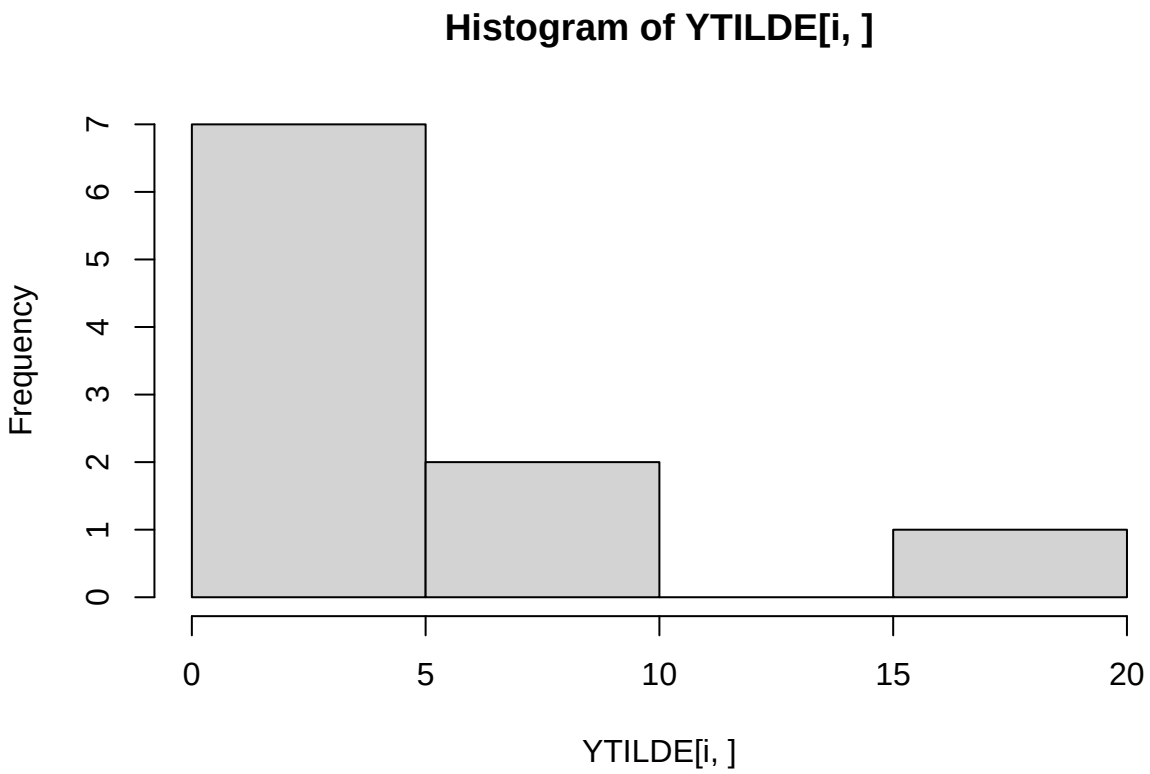

Histogram of YTILDE[i,]
Frequency
0
2
4
6
8
0
5
10
15
20
YTILDE[i,]

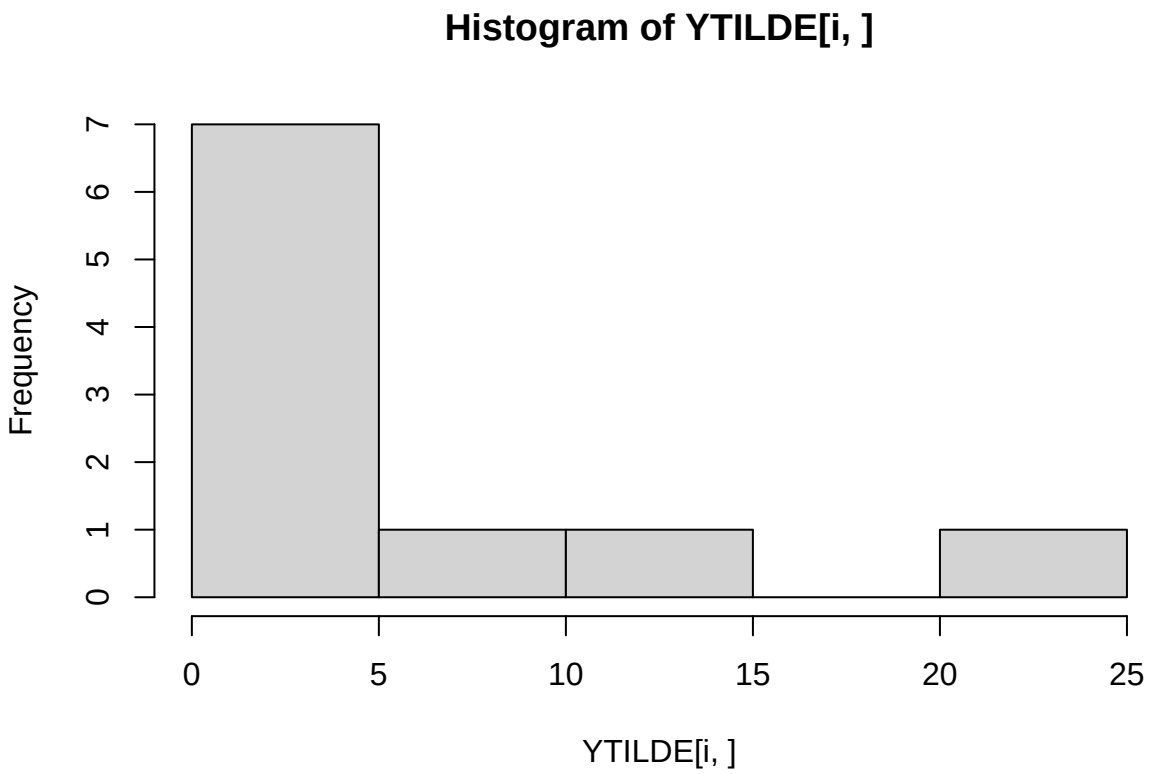

Histogram of YTILDE[i,]
Frequency
0
1
2
3
4
5
6
7
0
5
10
15
20
25
YTILDE[i,]

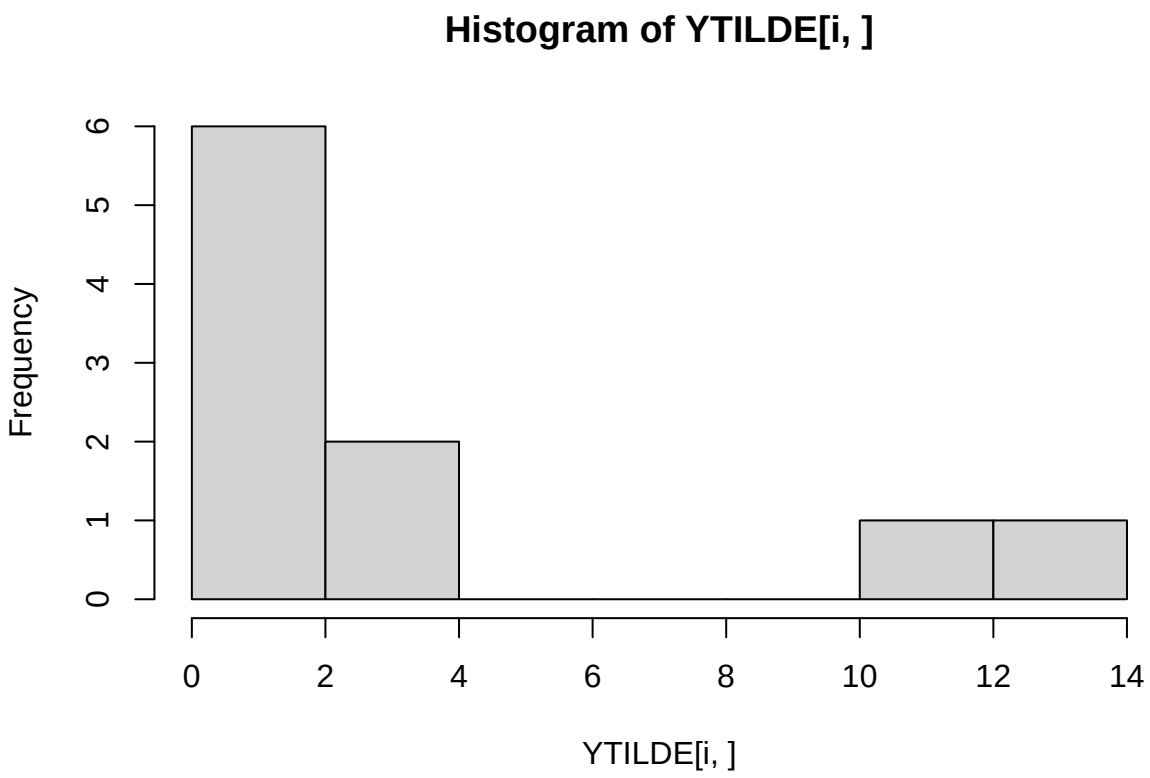

Histogram of YTILDE[i,]
Frequency
0
2
4
6
8
0
5
10
15
YTILDE[i,]

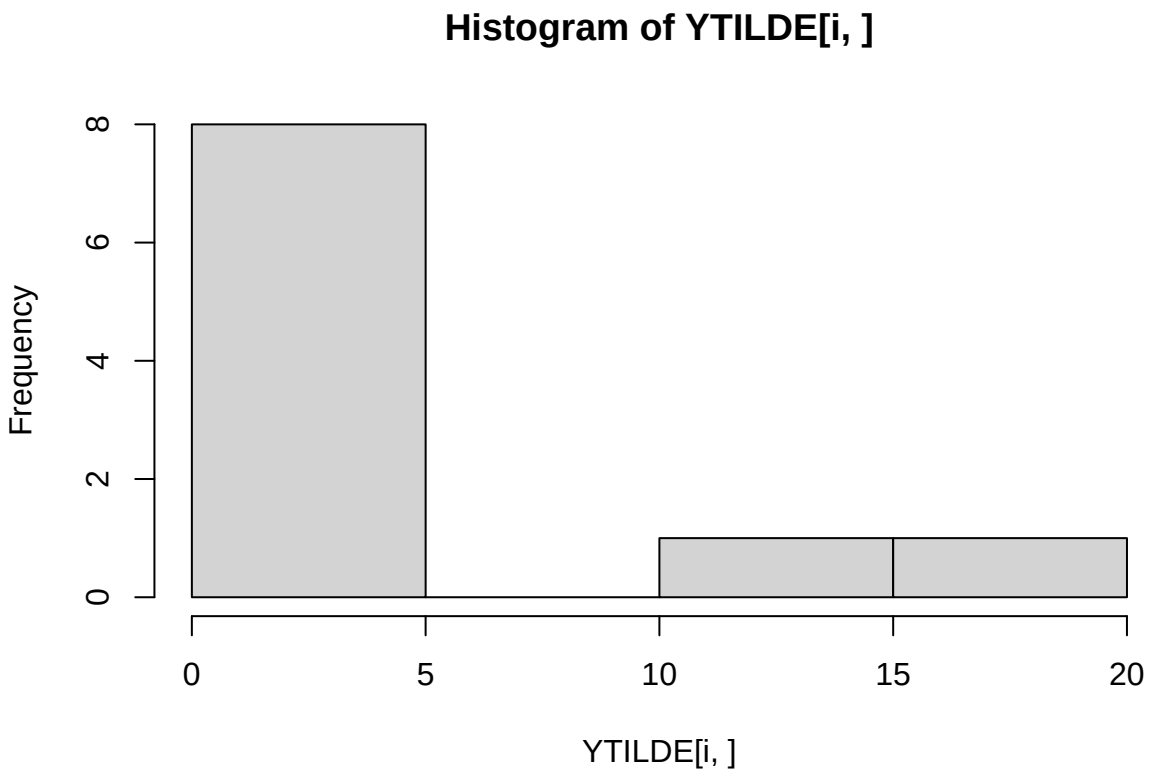

Histogram of YTILDE[i,]
Frequency
0
1
2
3
4
5
6
7
0
5
10
15
20
25
30
YTILDE[i,]

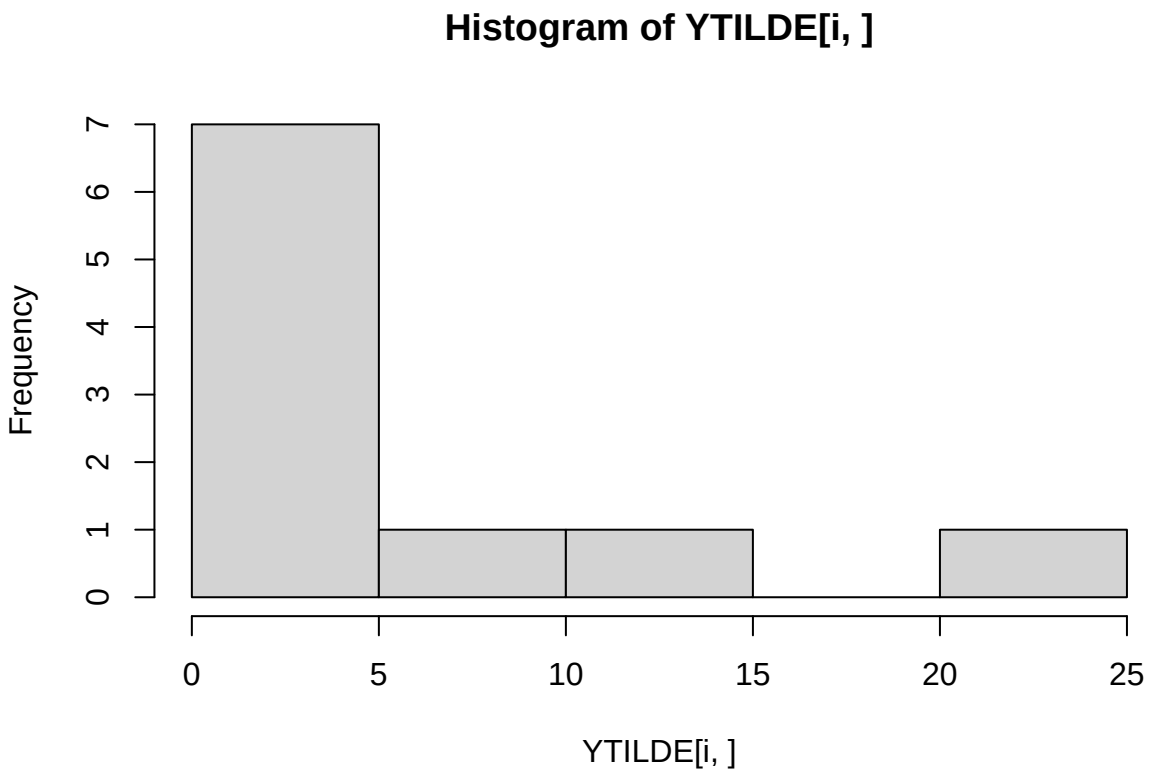

Histogram of YTILDE[i,]
Frequency
0
2
4
6
8
0
5
10
15
20
YTILDE[i,]

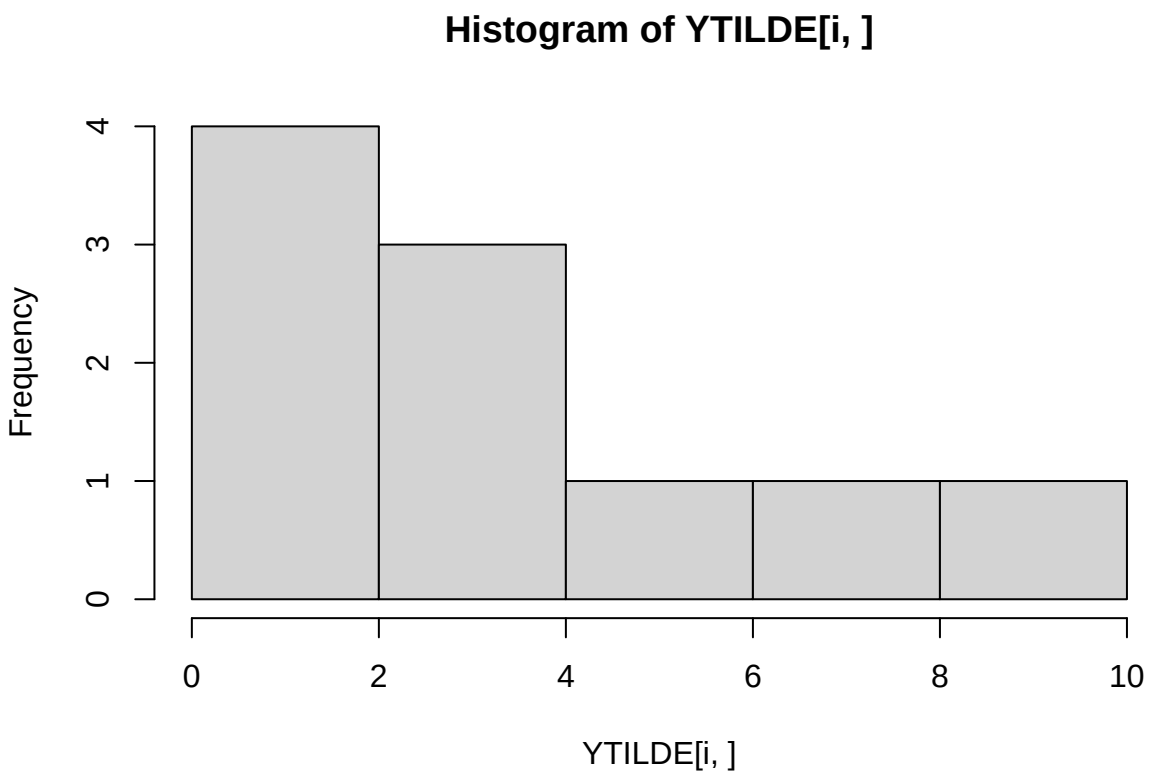

Histogram of YTILDE[i,]
Frequency
0
1
2
3
4
5
6
7
0
5
10
15
20
YTILDE[i,]

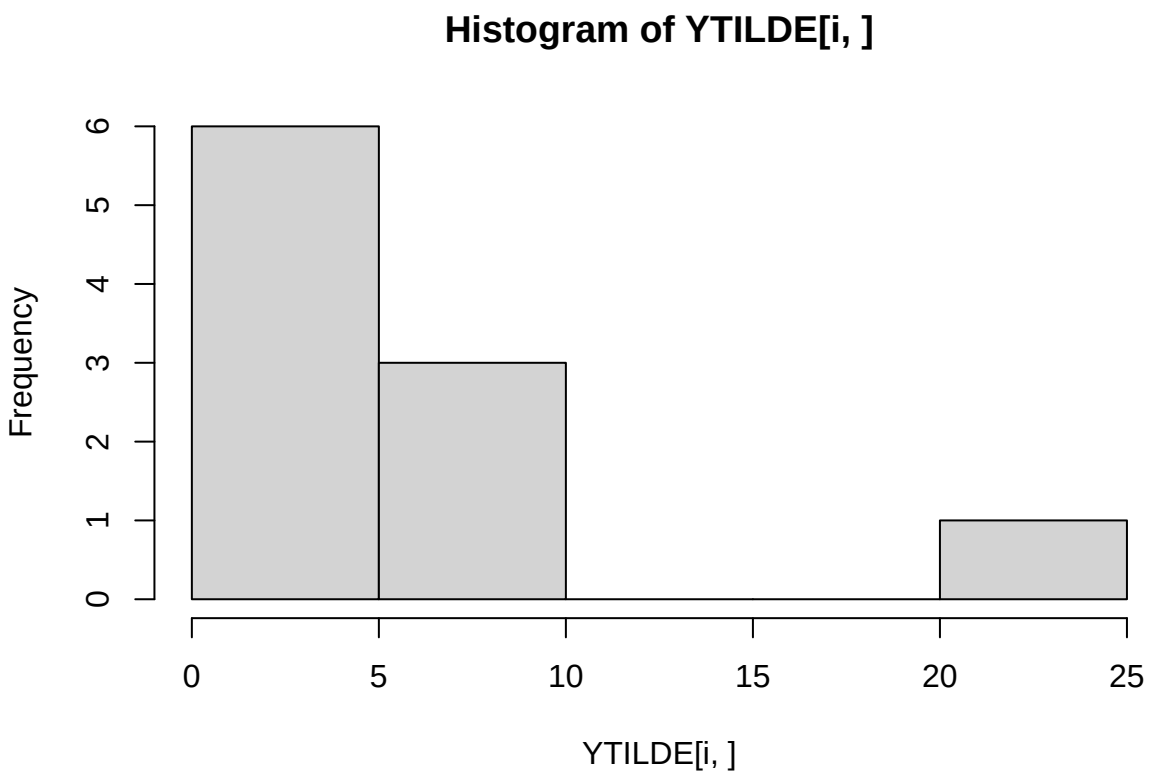

Histogram of YTILDE[i,]
Frequency
0
1
2
3
4
5
6
7
0
5
10
15
20
25
YTILDE[i,]

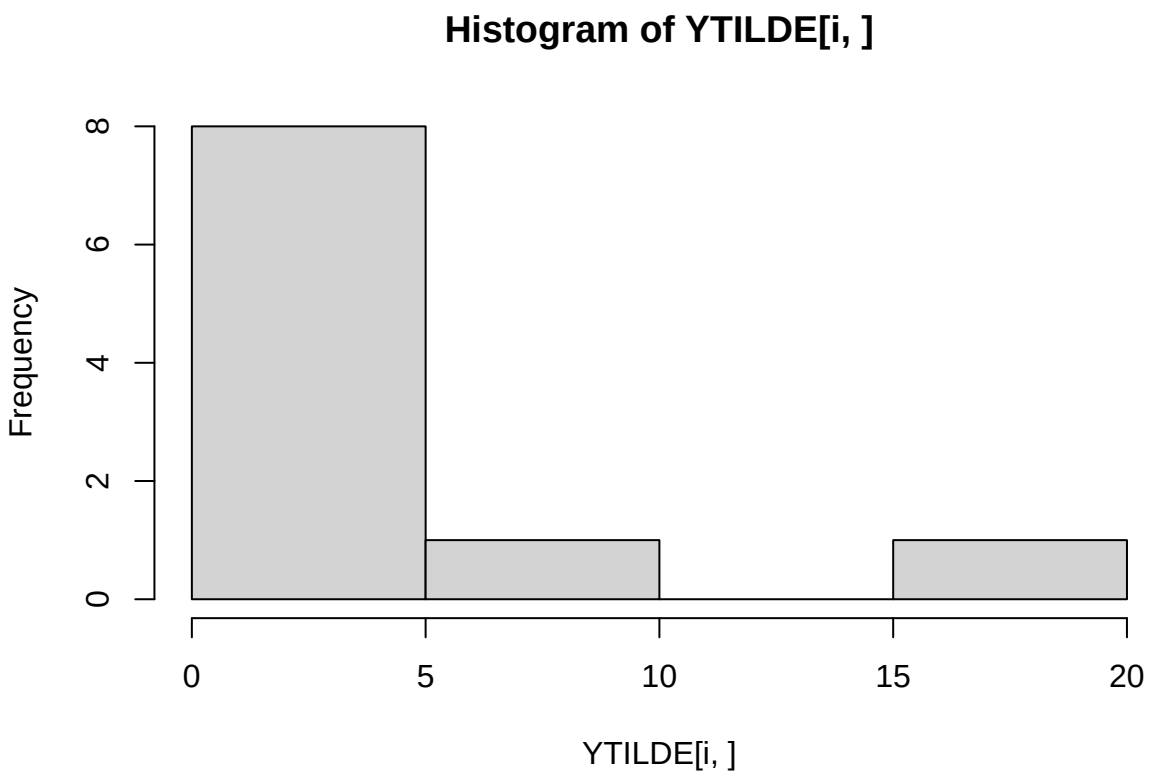

Histogram of YTILDE[i,]
Frequency
0
1
2
3
4
5
6
0
2
4
6
8
10
12
14
YTILDE[i,]

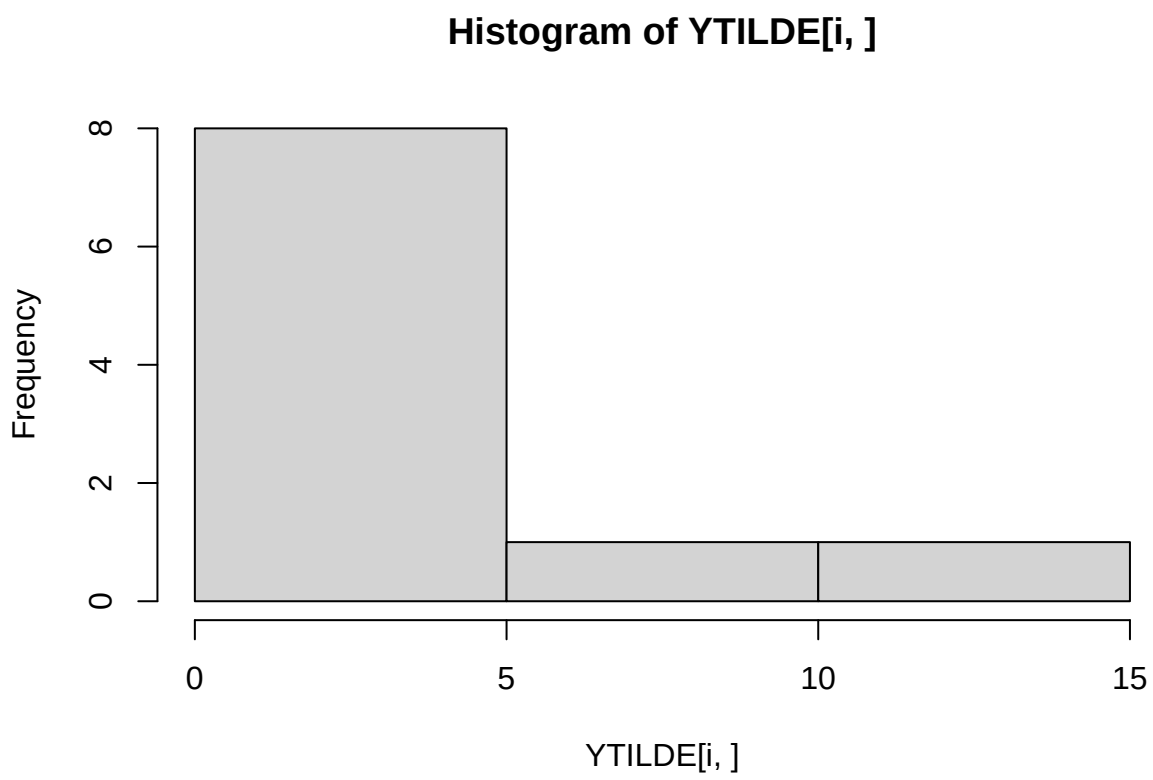

Histogram of YTILDE[i,]
Frequency
0
2
4
6
8
0
5
10
15
20
YTILDE[i,]


Histogram of YTILDE[i,]
Frequency
0
1
2
3
4
5
6
7
0
5
10
15
20
25
YTILDE[i,]

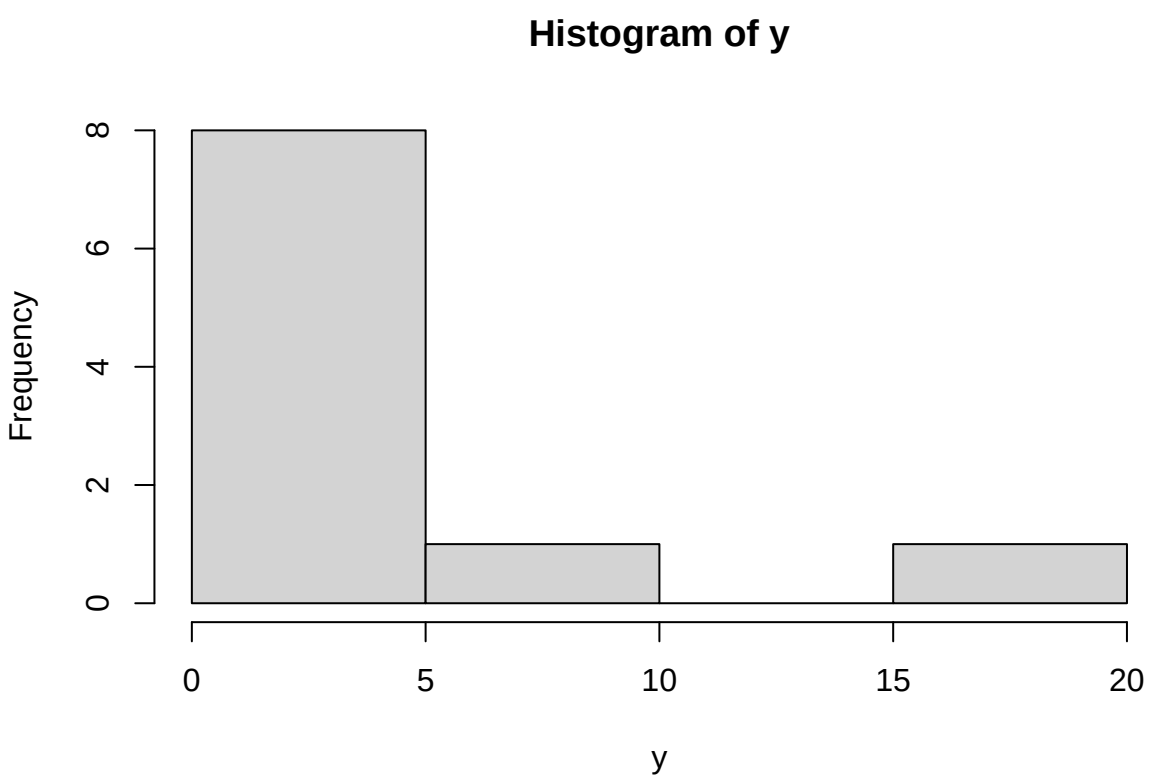

Histogram of YTILDE[i,]
Frequency
0
1
2
3
4
0
2
4
6
8
10
YTILDE[i,]


Histogram of YTILDE[i,]
Frequency
0
1
2
3
4
5
6
0
5
10
15
20
25
YTILDE[i,]


Histogram of YTILDE[i,]
Frequency
0
2
4
6
8
0
5
10
15
20
YTILDE[i,]




```
hist(y)
```



As we can see the first plots have a similar behavior as the plot shown in the exercise sheet. the points have ups and downs but at the overall shape is increasing. however it could still be improved a little bit so it matches perfectly the data but we can say that the model behaves good overall.

the histograms match sometimes the real histogram however by taking just one value of ytilde every time from a poisson distribution it would be hard to get exactly the same shape every time

Grading

Total 12 points Each of the above steps gives 2 points.