Dissertation Work for supervision 25th November

AUTHOR
Emma Bowen

For my dissertation, I have been working on different ways that acquisition functions could be implemented alongside Bayesian History Matching to speed up management strategy evaluation in fisheries.

I have based my work on the paper "Using history matching to speed up management strategy evaluation grid searches".

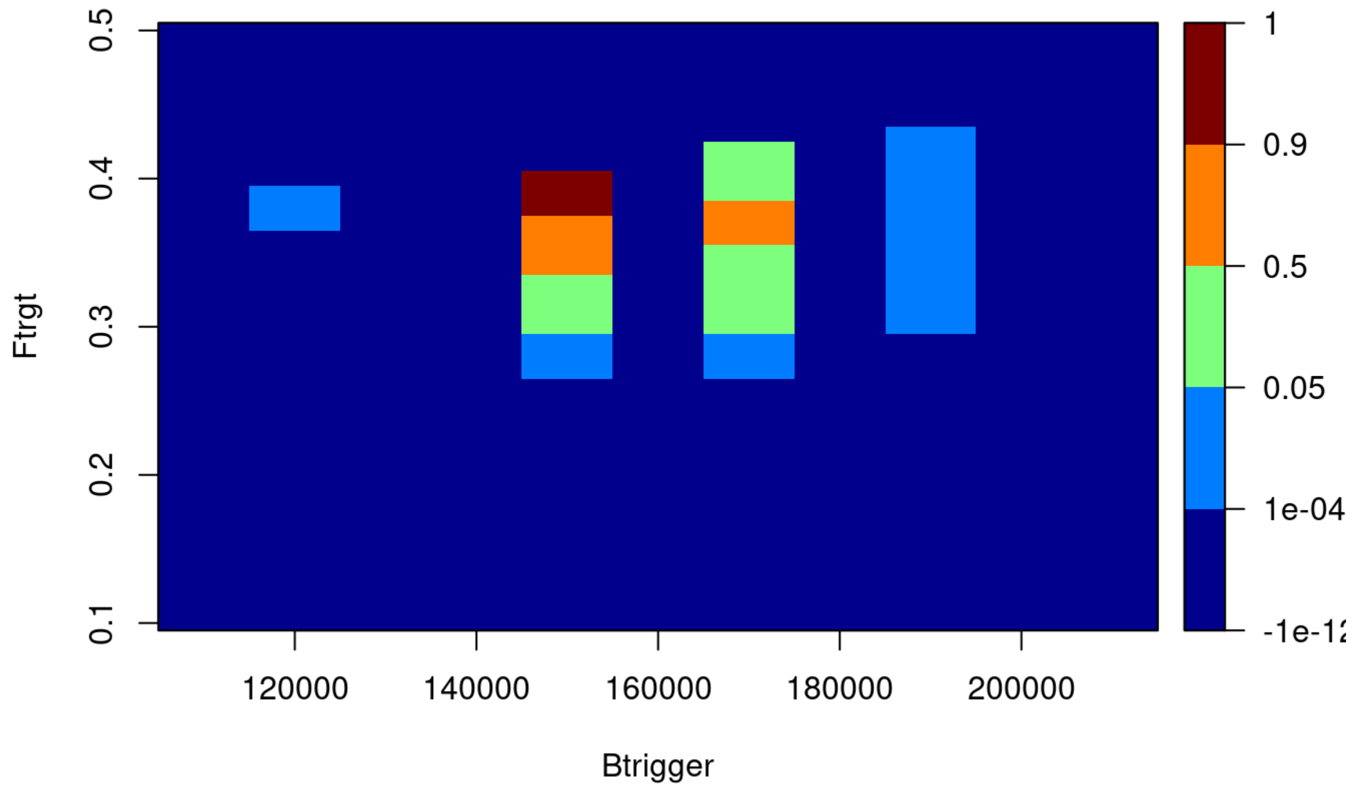
Currently, I am focusing on three files on my Github that are looped versions of the original code from this paper with acquisition functions added in. I will show the outputs from these files below and try to provide some explanation.

Looped version of case_study8.R with Expected Improvement

This is the case_study8.R file but now looped and with an expected improvement acquisition function added, which can be found here in my repo.

I will now add some of the outputs of this file. The number of rounds can be changed using the `max_rounds` variable which is how I have made the different plots.

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union

Joining with `by = join_by(Ftrgt, Btrigger)`

Plausible points remaining: 258

Current maximum 54596.5


 Round 2
Unevaluated candidates with EI > 0: 257
```



```
Plausible points remaining: 47
Best catch so far: 54596.5


 OPTIMIZATION COMPLETE

Completed 2 rounds

Total evaluations: 16


Optimal parameters:
```

The plot above presents the plausible points remaining in Round 2. For the points that are still plausible, it shows the probability that their median catch is higher than the current best median catch. We then use the expected improvement acquisition function to determine which points will be the best to sample at next.

First we use Bayesian History Matching to determine which points are still plausible. Then, we use the function below:

```r
#' @param xi is a scalar for exploration/exploitation trade off
expected_improvement <- function(mu, sigma, y_best, xi = 0.05, task = "max", pred_risk
{
  ei <- numeric(length(mu))
  safe_points <- pred_risk >= eps

  # Only calculate EI for safe points
  if (any(safe_points)) {
    if (task == "min")
        imp <- y_best - mu[safe_points] - xi
    if (task == "max")
        imp <- mu[safe_points] - y_best - xi
    else
        stop('task must be "min" or "max"')

    Z <- imp / sigma[safe_points]
    ei[safe_points] <- imp * pnorm(Z) + sigma[safe_points] * dnorm(Z)
    ei[safe_points][sigma[safe_points] == 0.0] <- 0.0
  }

  return(ei)
}
```

where you can see that we remove any points that have a probability less than `eps = 1e-4` of having `prisk<=0.05` removing any points with `pred_risk < eps`. Then, we only calculate EI for these `safe_points`. This saves us some computation time, especially as we move into later rounds. The calculation is done using the standard formula for EI.

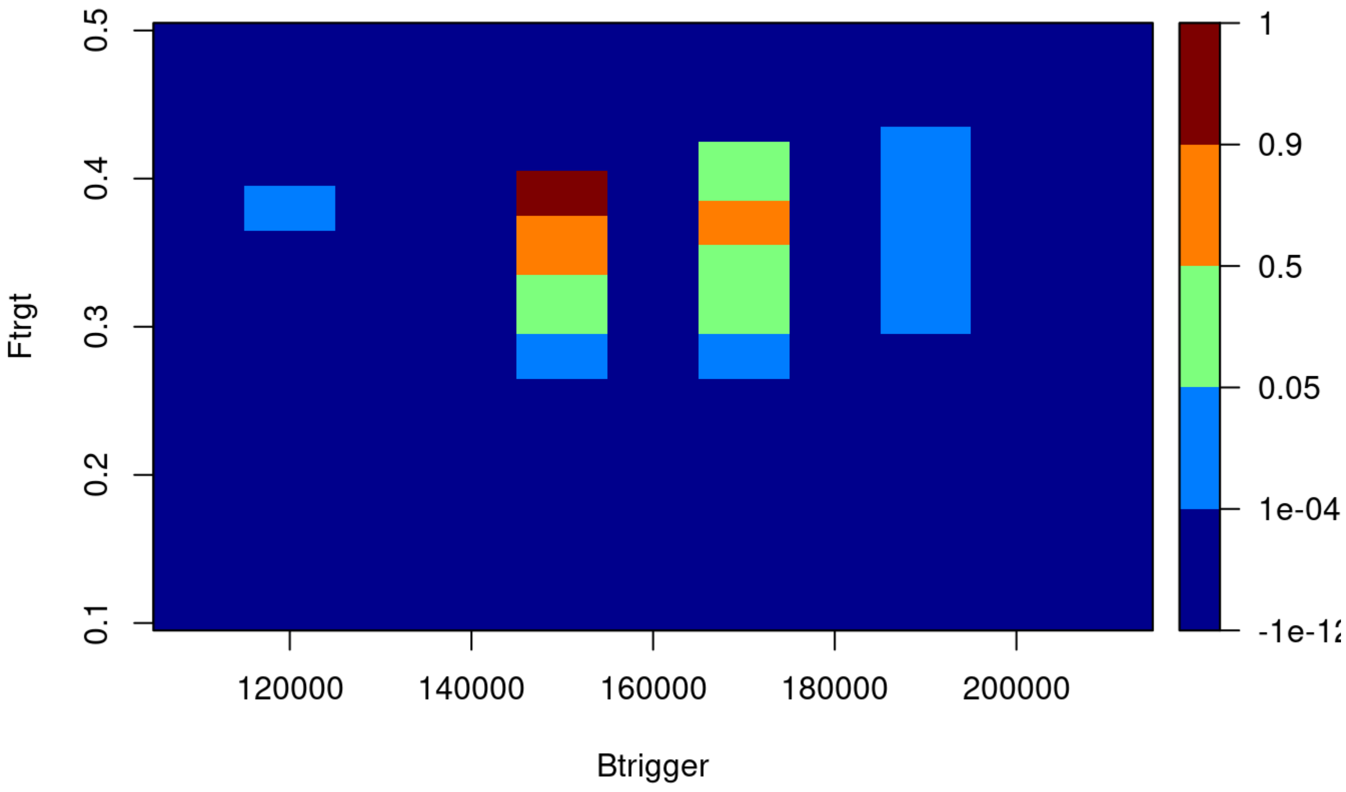
If we let $f_max = max(y_1, \dots, y_n)$ be the current best value, as we are maximising, and sample next at some point x_{n+1} with the emulator as $\hat{f}$, then we have:

$$EI[(x)] = (f_{min} - \hat{f})\Phi(\frac{f_{min} - \hat{f}}{s}) + s\phi(\frac{f_{min} - \hat{f}}{s})$$

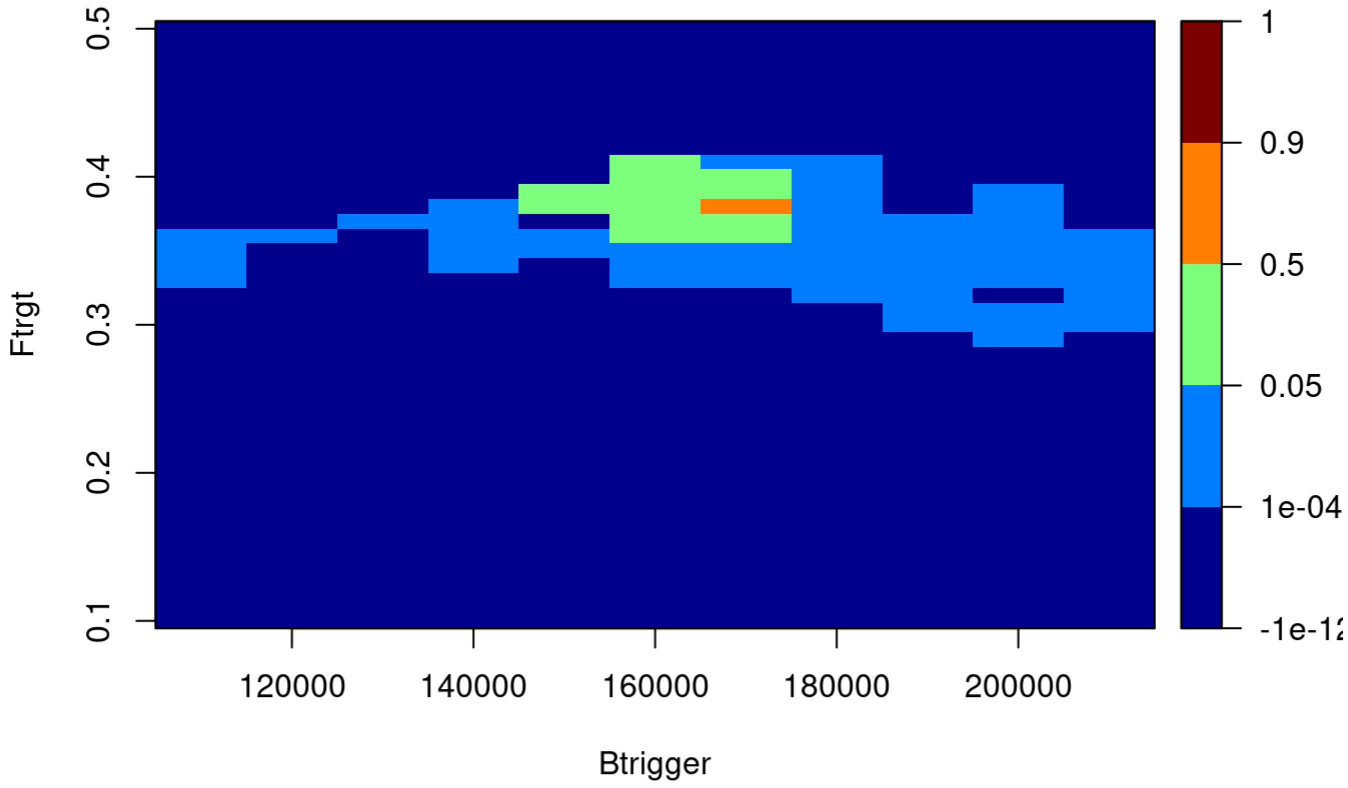
where s is the standard deviation of $\hat{f}$ at x and is the standard normal distribution function and is the standard normal density function.

We then repeat the code until we resolve to one point:

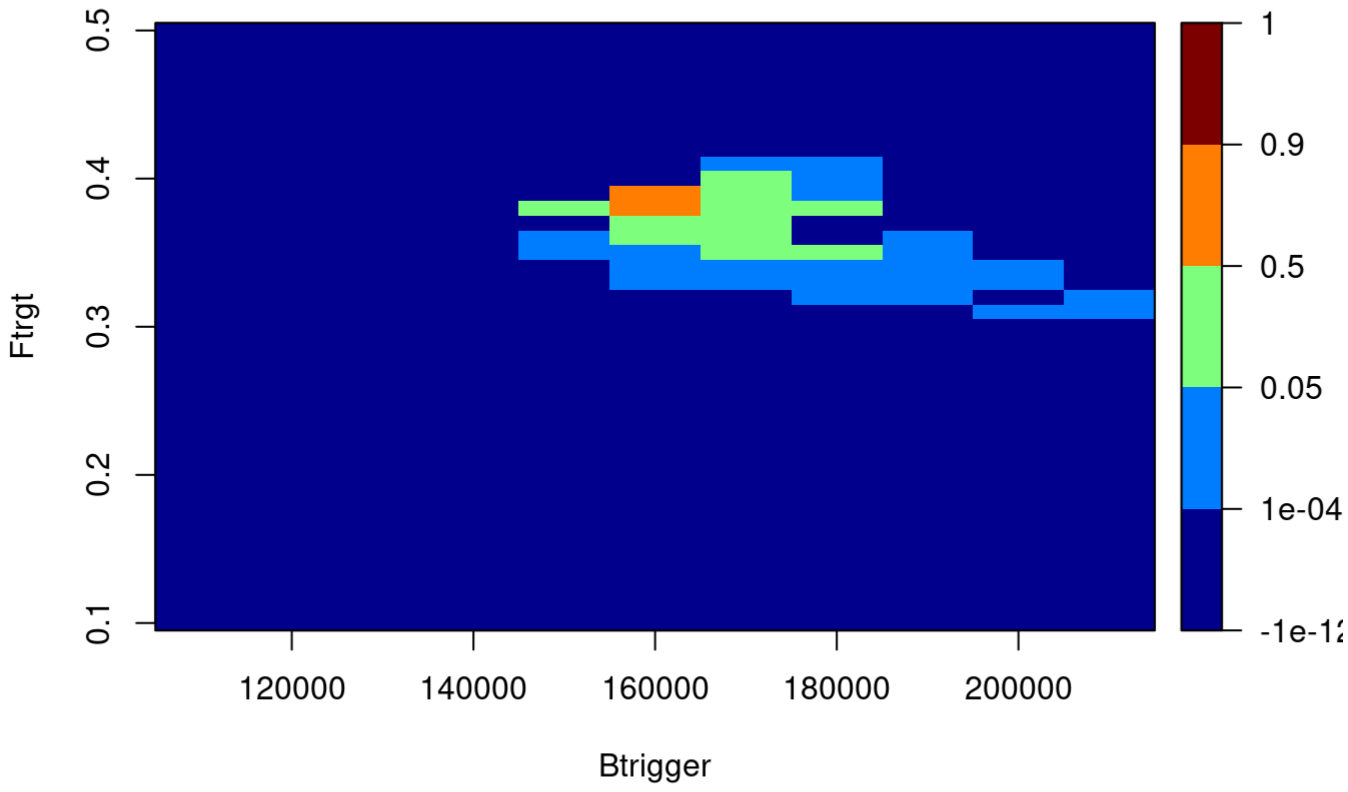
```
Joining with `by = join_by(Ftrgt, Btrigger)`

Plausible points remaining: 258

Current maximum 54596.5


 Round 2
Unevaluated candidates with EI > 0: 257
```



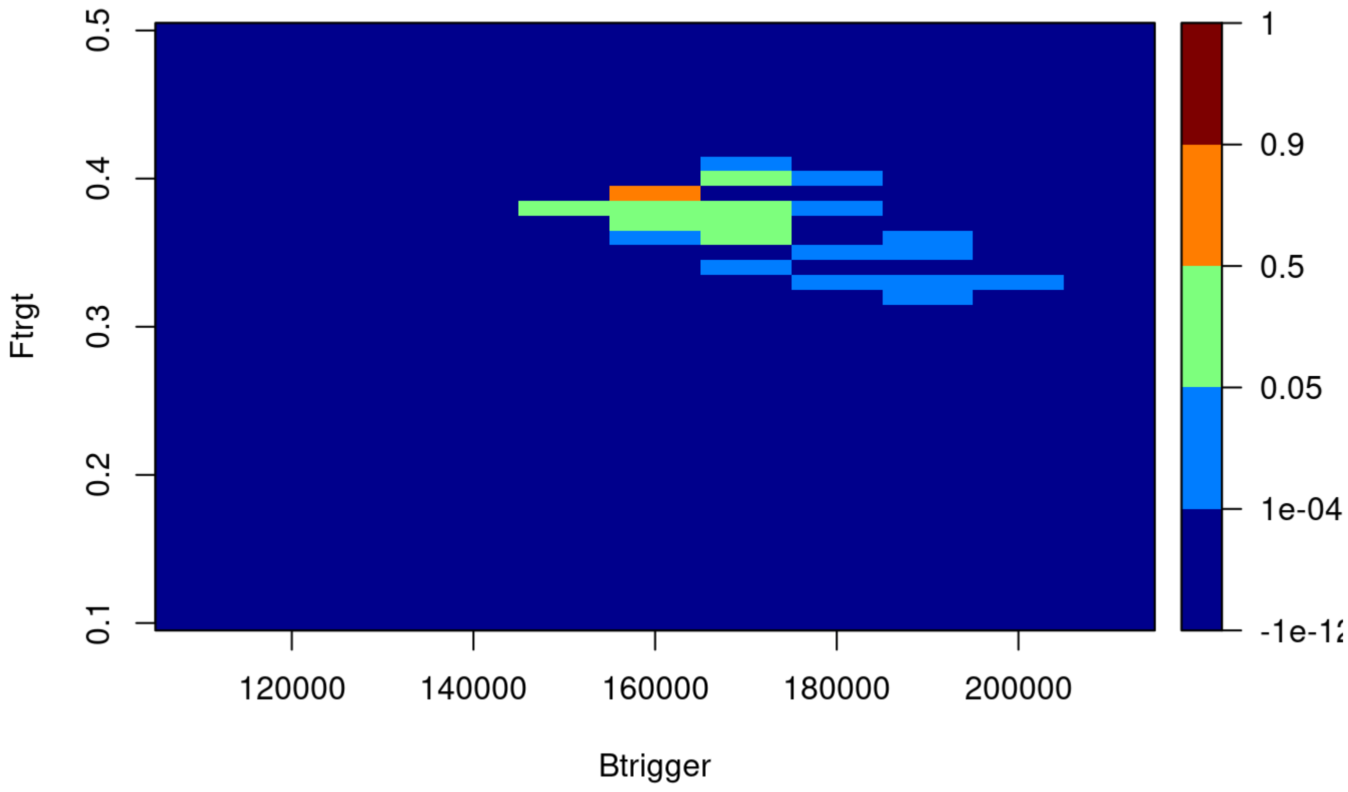
```
Plausible points remaining: 47
Best catch so far: 54596.5

 Round 3
Unevaluated candidates with EI > 0: 46
```



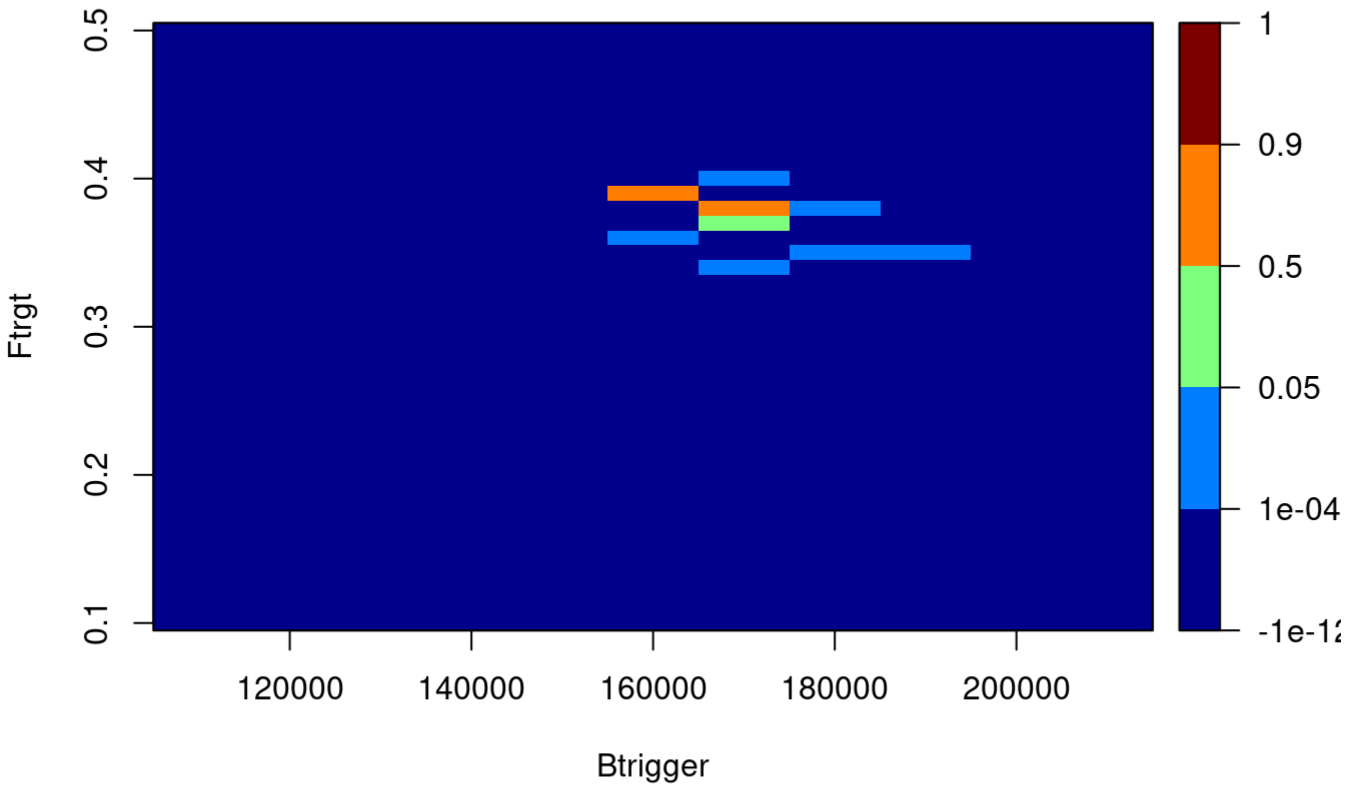
```
Plausible points remaining: 68
Best catch so far: 54596.5

 Round 4
Unevaluated candidates with EI > 0: 67
```



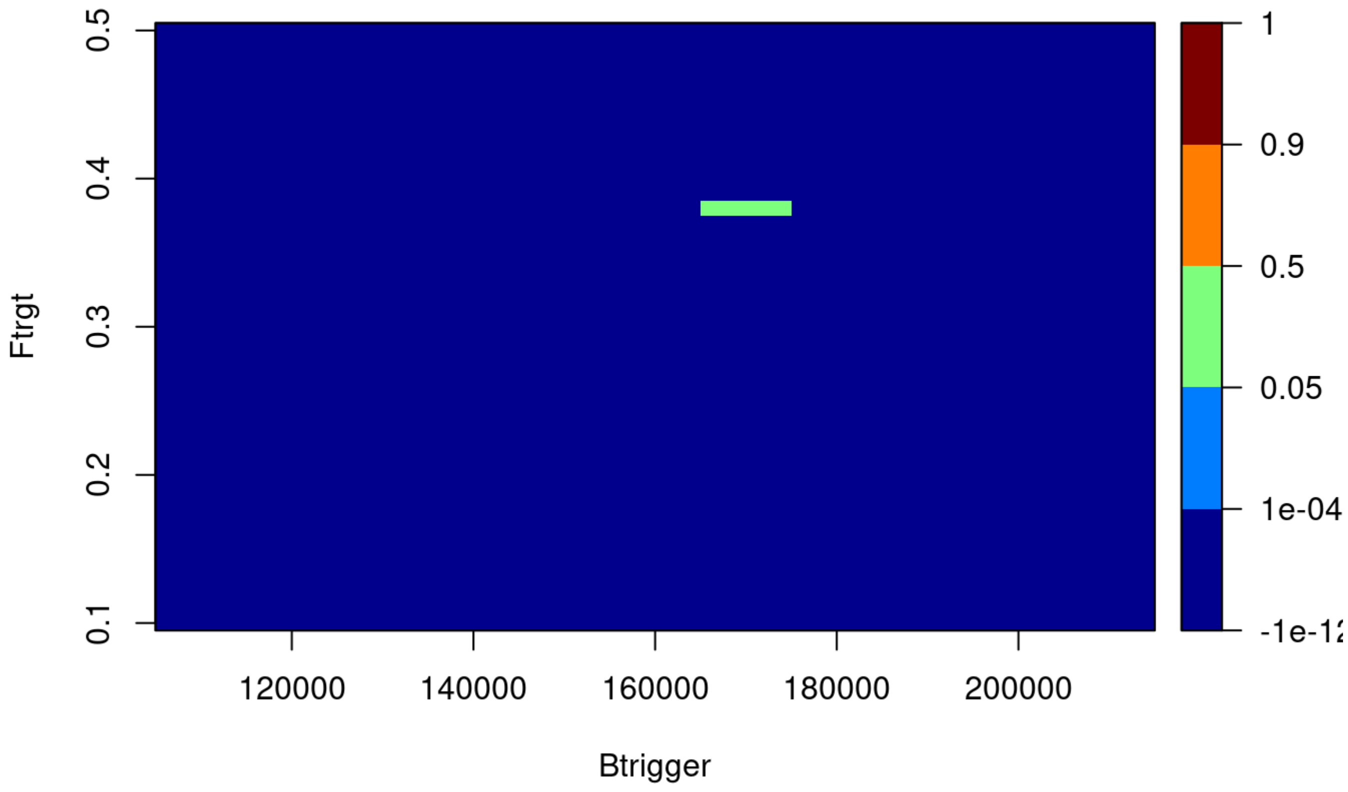
```
Plausible points remaining: 37
Best catch so far: 54596.5

 Round 5
Unevaluated candidates with EI > 0: 36
```



```
Plausible points remaining: 20
Best catch so far: 54596.5

 Round 6
Unevaluated candidates with EI > 0: 19
```



```
Plausible points remaining: 9
Best catch so far: 54596.5

 Round 7
Unevaluated candidates with EI > 0: 8
```



```
Plausible points remaining: 1
Best catch so far: 54596.5

 Round 8
No unevaluated candidates with positive EI. Stopping at round 8


 OPTIMIZATION COMPLETE

Completed 7 rounds

Total evaluations: 56


Optimal parameters:
    Ftarget Btrigger
275    0.38   170000
```

Looped version of case_study8.R with Augmented Expected Improvement

This is very similar to EI an the only change we make is to add an augmentation factor to help deal with very noisy observations:

```
{r}
# Augmentation factor to handle noise
```

```
        # When noise_var = 0, augmentation_factor = 1 (reduces to standard EI)
        augmentation_factor <- 1 - sqrt(noise_var / (noise_var + sigma^2))
        # OK to leave as this as 0*anything = 0
        aei <- ei * augmentation_factor
```

Then we get the whole augmented expected improvement function as:

```
#' @param xi is a scalar for exploration/exploitation trade off
augmented_expected_improvement <- function(mu, sigma, y_best, xi = 0.05, task = "max",
{
  ei <- numeric(length(mu))
  safe_points <- pred_risk >= eps

  # Only calculate AEI for safe points
  if (any(safe_points))
        if (task == "min")
            imp <- y_best - mu[safe_points] - xi
        if (task == "max")
            imp <- mu[safe_points] - y_best - xi
        else
            stop('task must be "min" or "max"')

        Z <- imp / sigma[safe_points]
        ei[safe_points] <- imp * pnorm(Z) + sigma[safe_points] * dnorm(Z)
        ei[safe_points][sigma[safe_points] == 0.0] <- 0.0

        # Augmentation factor to handle noise
        # When noise_var = 0, augmentation_factor = 1 (reduces to standard EI)
        augmentation_factor <- 1 - sqrt(noise_var / (noise_var + sigma^2))
        # OK to leave as this as 0*anything = 0
        aei <- ei * augmentation_factor

  # Giving points with prob(risk <= 0.05) < 0.01 an expected improvement of zero so we
  aei <- ifelse(pred_risk < eps, 0, aei)

  return(aei)
}
```

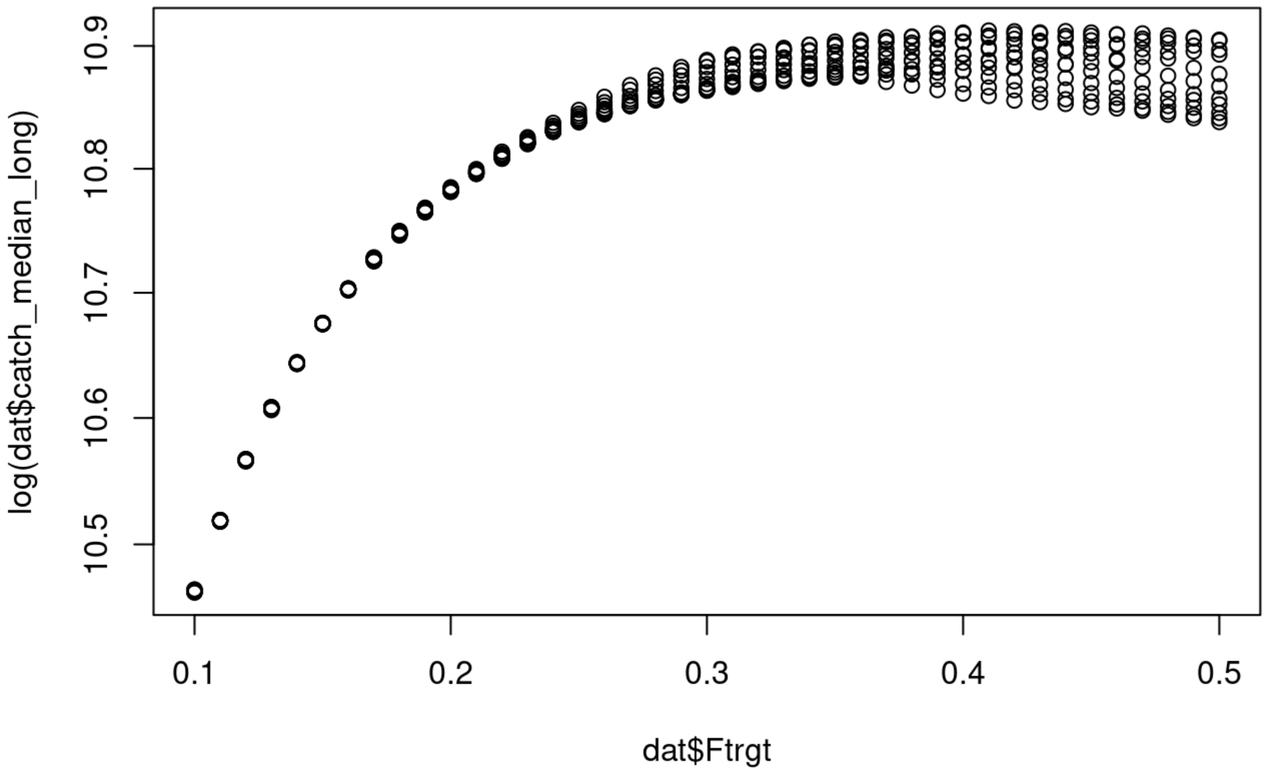
Looped ver of case_study8.R with Knowledge Gradient

This is detailed here but I will also enter the code below so that it can be run and the differences between the points KG and EI choose can be assessed:

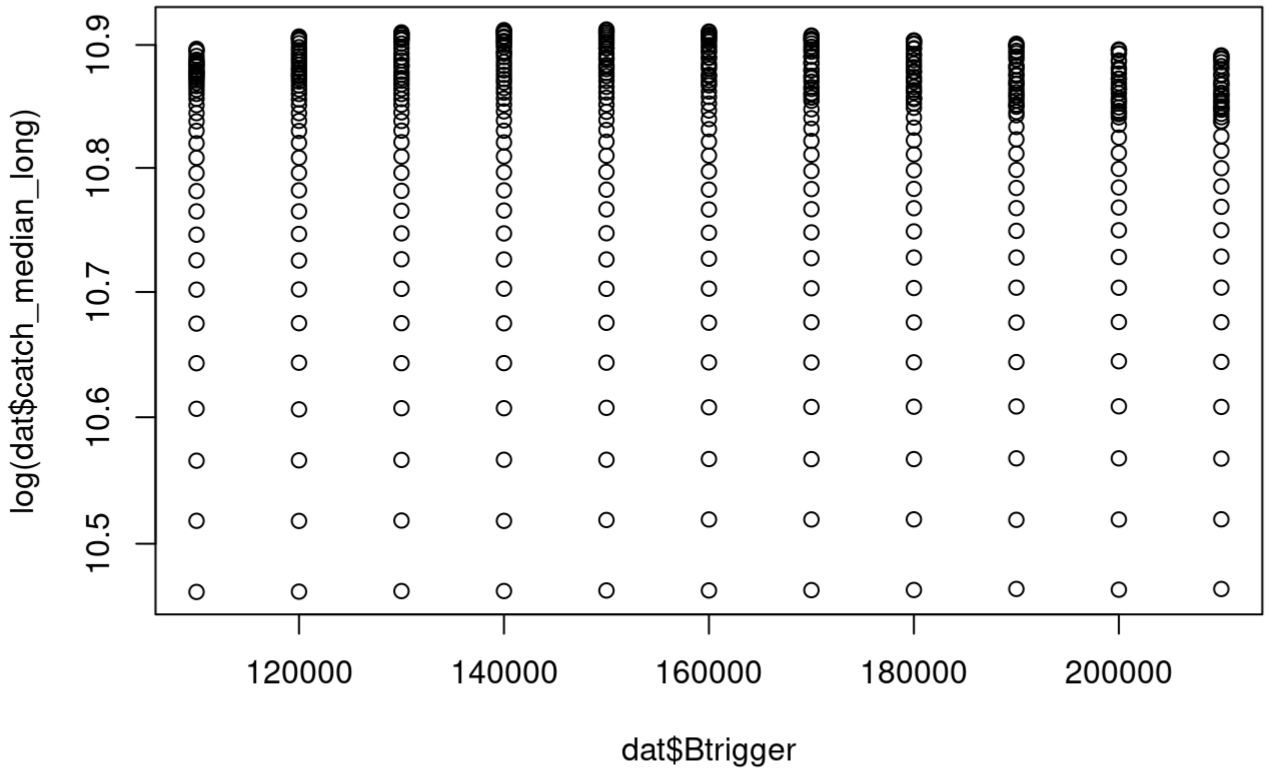
```
dat <- data.frame(
  Ftrgt =c(0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.11,0.11,0.11,0.11,0.11,0.11,0
  Btrigger =c(110000,120000,130000,140000,150000,160000,170000,180000,190000,210000,2e
  catch_median_long =c(34965,34973,34989.5,34990.5,35005,35005.5,35015,35019,35047.5,35
  risk1_full =c(0.01035,0.01025,0.01015,0.01015,0.0101,0.01005,0.01005,0.01,0.00995,0.0
)

rescale_Her <- function(runs,dat){
  # Rescale Ftarget and Btrigger to [0,1] using ranges from `dat`.
  # This makes optimisation / GP fitting numerically more stable.
  ret <- runs
  minF <- min(dat$Ftarget)
  rangeF <- diff(range(dat$Ftarget))
  ret$Ftarget <- (ret$Ftarget - minF)/rangeF
  minB <- min(dat$Btrigger)
  rangeB <- diff(range(dat$Btrigger))
  ret$Btrigger <- (ret$Btrigger - minB)/rangeB
  return(ret)
}

unrescale_Her <- function(runs,dat){
  # Undo the rescaling performed by `rescale_Her` to recover original units.
  ret <- runs
  minF <- min(dat$Ftarget)
  rangeF <- diff(range(dat$Ftarget))
  ret$Ftarget <- ret$Ftarget*rangeF + minF
  minB <- min(dat$Btrigger)
  rangeB <- diff(range(dat$Btrigger))
  ret$Btrigger <- ret$Btrigger*rangeB + minB
  return(ret)
}

library(DiceKriging)
library(dplyr)
library(plot3D)


cov_exp <- function(X1, X2, theta, sigma2) {
  n1 <- nrow(X1)
  n2 <- nrow(X2)
  # pairwise difference weighted by theta
  D <- array(0, dim = c(n1, n2))
  for (j in seq_along(theta)) {
    D <- D + theta[j] * abs(outer(X1[, j], X2[, j], "-"))
  }
  sigma2 * exp(-D)
}


knowledge_gradient_sim <- function(mu, sigma, model, obs_noise_var = 0, nsim = 100, pr
{
    X_pred <- dat[, c("Ftrgt", "Btrigger")]
    cov_grid <- cov_exp(X_pred, X_pred, theta = model@covariance@range.val, sigma2 = mo
    m <- length(mu)
    var <- sigma^2
    # Current best mean catch
    mu_best <- max(mu)
    kg <- numeric(m)

        # Loop over candidate points
        for (i in seq_len(m)) {
            # set knowledge gradient to 0 for any points with a predicted risk that is
            if (pred_risk[i] < eps) {
```

```r
                kg[i] <- 0
                next
            }

            mu_i <- mu[i]
            sigma_i <- sqrt(var[i] + obs_noise_var)
            # nsim simulated observations - using rnorm to sample from a normal distril
            # to model the catch, which is assumed to be normally distributed
            y_sim <- rnorm(nsim, mu_i, sigma_i)
            # the covariance between every point in the grid and the point x_i
            cov_xp_xi <- cov_grid[, i]
            # denominator term in the KG update formula
            denom <- var[i] + obs_noise_var
            # For each simulated y, compute updated max(mu)
            max_after <- numeric(nsim)
            for (s in seq_len(nsim)) {
                # says what would the new maximum mu be if we observed this simulated v
                # by implementing the standard formula for this in GP posterior updatin
                # This evaluates the posterior mean for every point in the space, upda
                # to the evaluated point by using the cov matrix
                mu_new <- mu + cov_xp_xi * (y_sim[s] - mu_i) / denom
                # takes this new maximum mu inot a vector for later averaging
                max_after[s] <- max(mu_new)
            }
            # Computes expected increase in the maximum posterior mean, mu
            # higher values mean we're getting better knowledge as to where and what tl
            kg[i] <- mean(max_after - mu_best)
        }
    # Returns the vector of knowledge gradient values for each point in the design spac
    kg
}

# Objective function - takes candidate points and returns their evaluated metrics
objective_func <- function(cand_points, lookup_data = dat) {
  # cand_points should have columns: Ftrgt, Btrigger
  # Returns: data frame with Ftrgt, Btrigger, catch_median_long, risk1_full

  evaluated <- left_join(cand_points, lookup_data, by = c("Ftrgt", "Btrigger"))

  # Select and rename to standard output format
  result <- data.frame(
    Ftarget = evaluated$Ftrgt,
    Btrigger = evaluated$Btrigger,
    C_long = evaluated$catch_median_long,
    risk3_long = evaluated$risk1_full
  )

  names(result) <- c("Ftarget", "Btrigger", "C_long", "risk3_long")

  return(result)
}

# Use a logarithmic y-axis for the plot, i.e. log="y" tells the plot to display the y-a
```

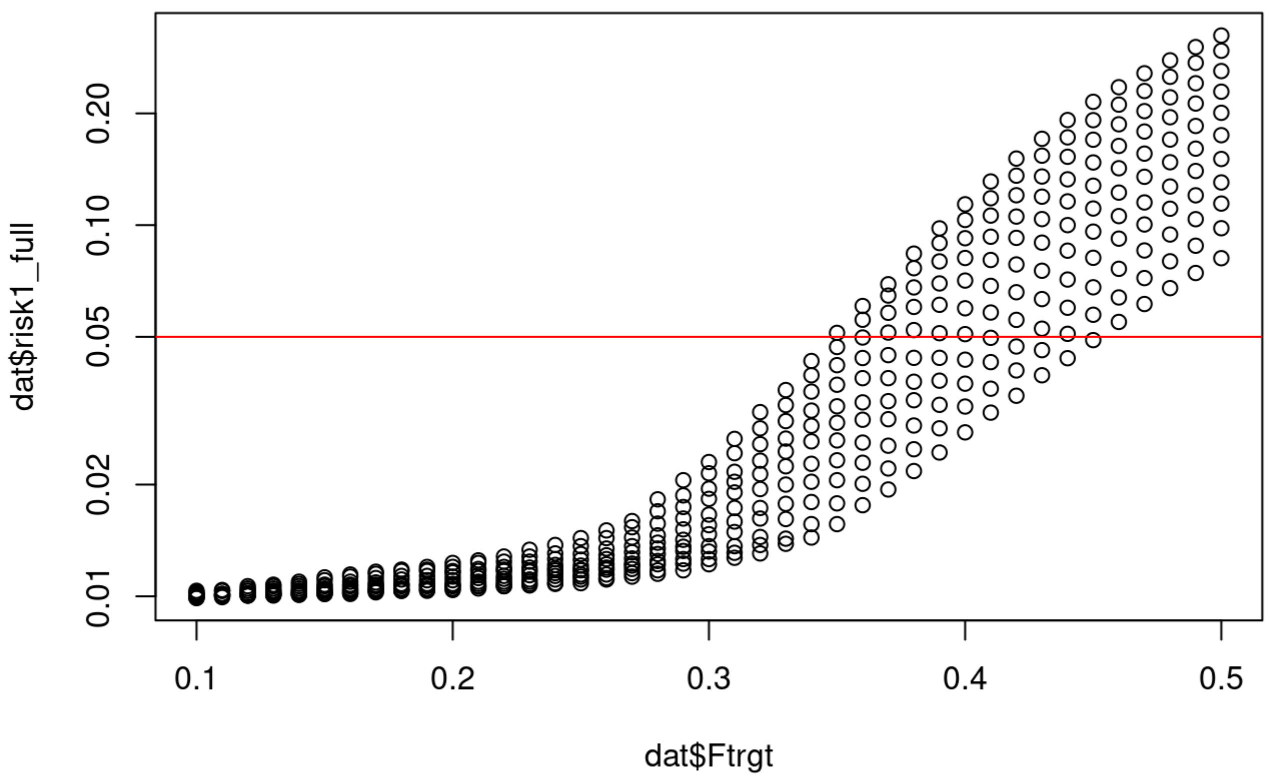
```
#CoPilot suggested this could result in a double transformation in lines 62 and 63, wh
plot(dat$Ftrgt,log(dat$catch_median_long),log="y")
```



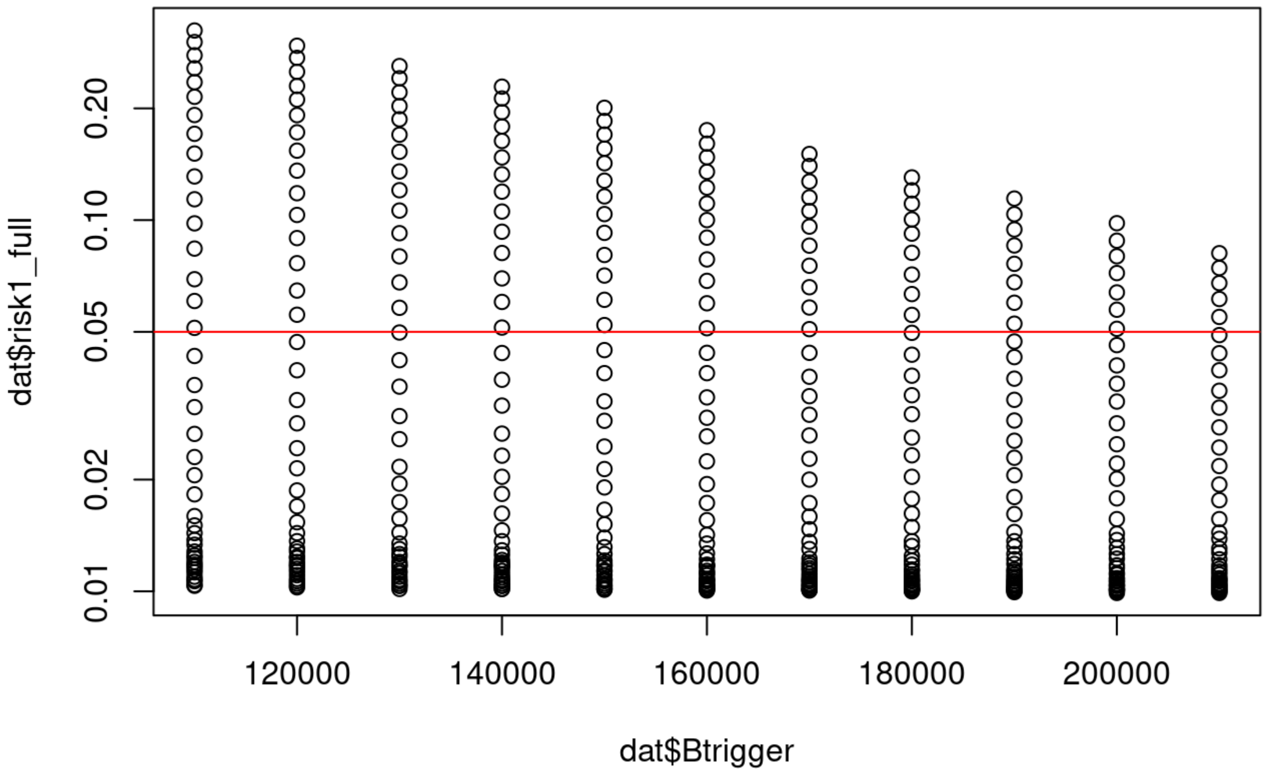
```
plot(dat$Btrigger,log(dat$catch_median_long),log="y")
```



```
plot(dat$Ftrgt,dat$risk1_full,log="y")
abline(h=0.05,col="red")
```



```r
plot(dat$Btrigger,dat$risk1_full,log="y")
abline(h=0.05,col="red")
```



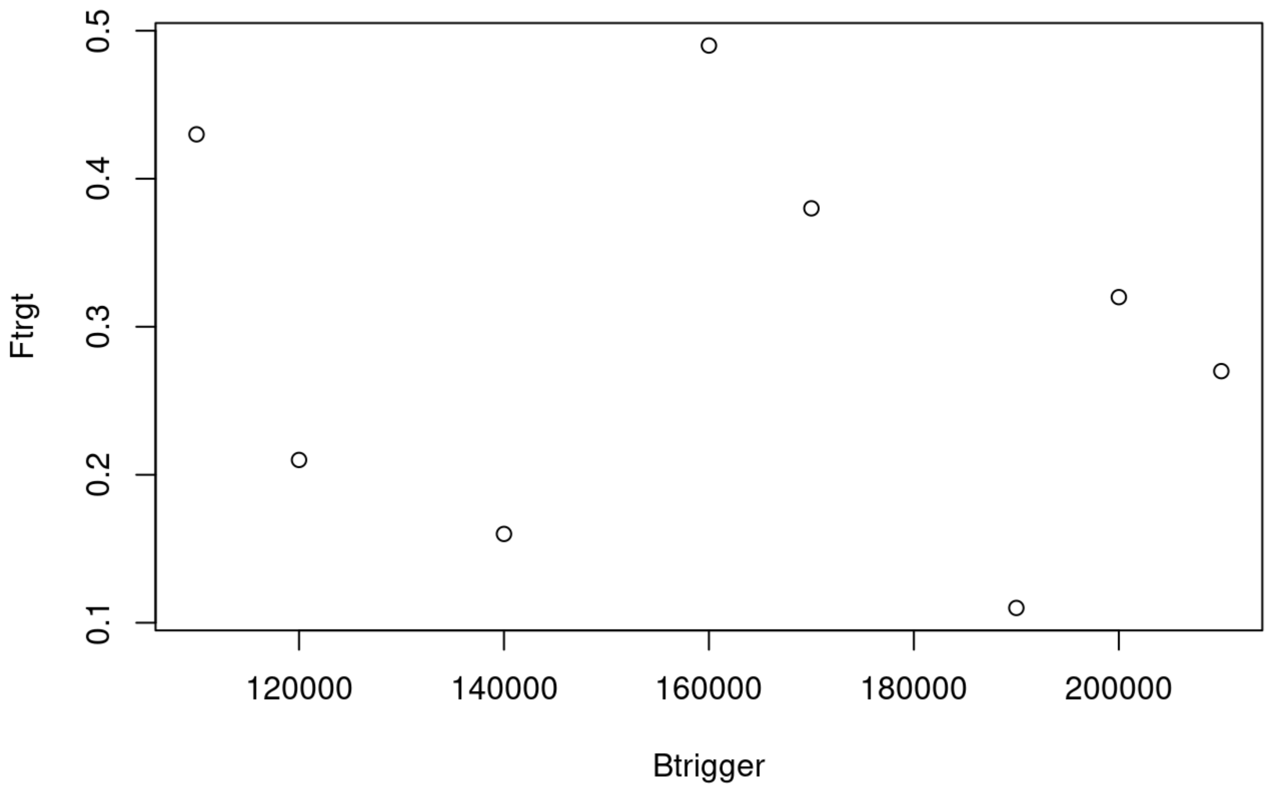
```
# Checks length of Ftrgt and Btrigger sets
length(unique(dat$Ftrgt))
```

```
[1] 41
```

```
length(unique(dat$Btrigger))
```

```
[1] 11
```

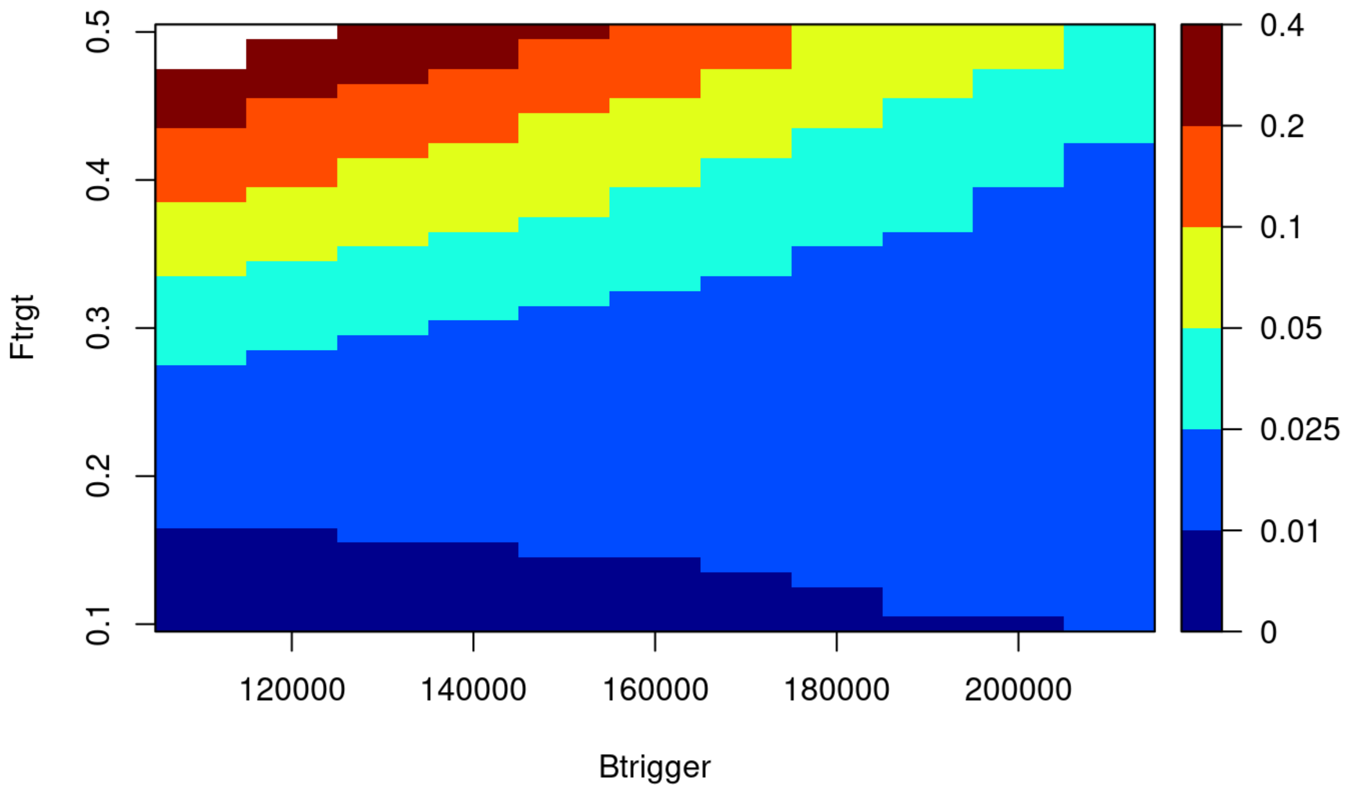
```
# Set Random Number Generator seed for reproducible random sampling.
# Copilot says I could change this to explore sensitivity of the process to the intial
set.seed(18)




#Round 1

# A space filling algorithm (seems simply chooses evenly spaced points) is used in the
# as detailed in Section 3 Case Study in the paper
round1 <- data.frame(Ftrgt=sample(unique(dat$Ftrgt)[floor(seq(2,40,l=8))]),Btrigger=sa
plot(round1[,2:1])
```



```
# Choose an initial small set of design points and fit GP emulators for log-catch and 
Ftarget <- sort(unique(dat$Ftrgt))
Btrigger <- sort(unique(dat$Btrigger))
dat1 <- data.frame(expand.grid(Ftarget,Btrigger))
names(dat1) <- c("Ftarget","Btrigger")

gridd <- dat1[,c("Ftarget","Btrigger")]
gridd<- rescale_Her(gridd[order(gridd$Ftarget,gridd$Btrigger),],dat=dat1)

dat_run <- left_join(round1,dat)
```

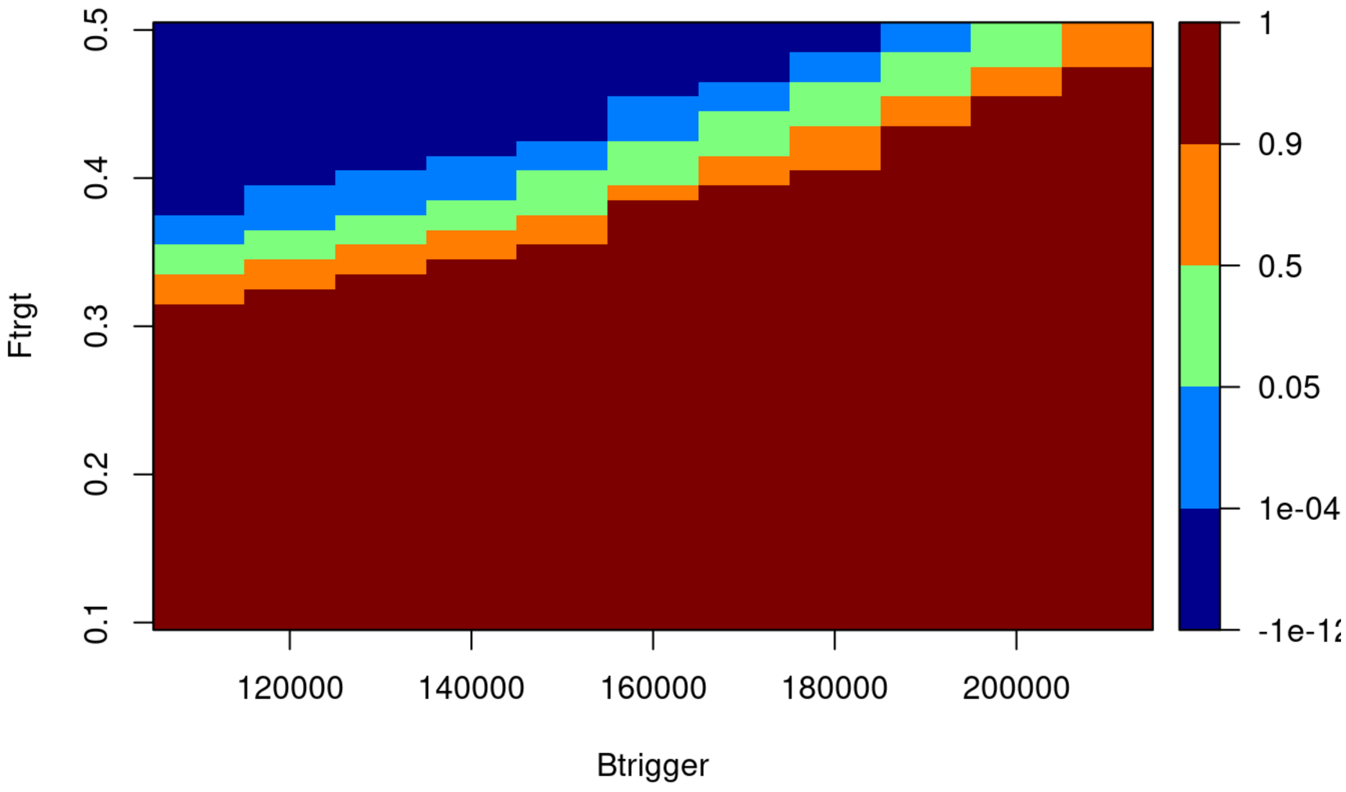
```
Joining with `by = join_by(Ftrgt, Btrigger)`
```

```
names(dat_run) <- c("Ftarget","Btrigger","C_long","risk3_long")
runs <- rescale_Her(dat_run,dat=dat1)

# build the emulator for median catch

# res_cat is the log of observed median catch at the design runs. The GP
# on line 109 models this response using polynomial terms in the formula.
res_cat <- log(runs$C_long)

# Looks like it is using maximum likelihood estimation below and mentions this nugget 
# CoPilot says "nugget: a tiny diagonal noise term for numerical stability."

# CoPilot also said "covtype = "exp": exponential correlation function (gives a less s
# which could maybe be something to follow up on

# Also has covariance type which is important for Gaussian Processes

# Below carries out the Kriging process for the two separate emulators, as detailed in
invisible(capture.output(
gp_cat <- km(~.^2,design=runs[,c("Ftarget","Btrigger")],estim.method="MLE",response = r
))
res_risk <- log(runs$risk3_long)

invisible(capture.output(
gp_risk <- km(~.^2,design=runs[,c("Ftarget","Btrigger")],estim.method="MLE",response =
))

# Gets the Gaussian Process to produce a prediciton for risk at every point in gridd
# gridd is the rescaled grid of Ftrgt and Btrigger
#This has a mean and standard deviation as we are unsure of the exact risk
pred_risk1_g <- predict(gp_risk,newdata=gridd,type="SK")


# Estimate median predicted risk
med_risk1 <- exp(pred_risk1_g$mean)
# plot it
image2D(matrix(med_risk1,nrow=11),breaks=c(0,0.01,0.025,0.05,0.1,0.2,0.4),y=sort(uniqu
```



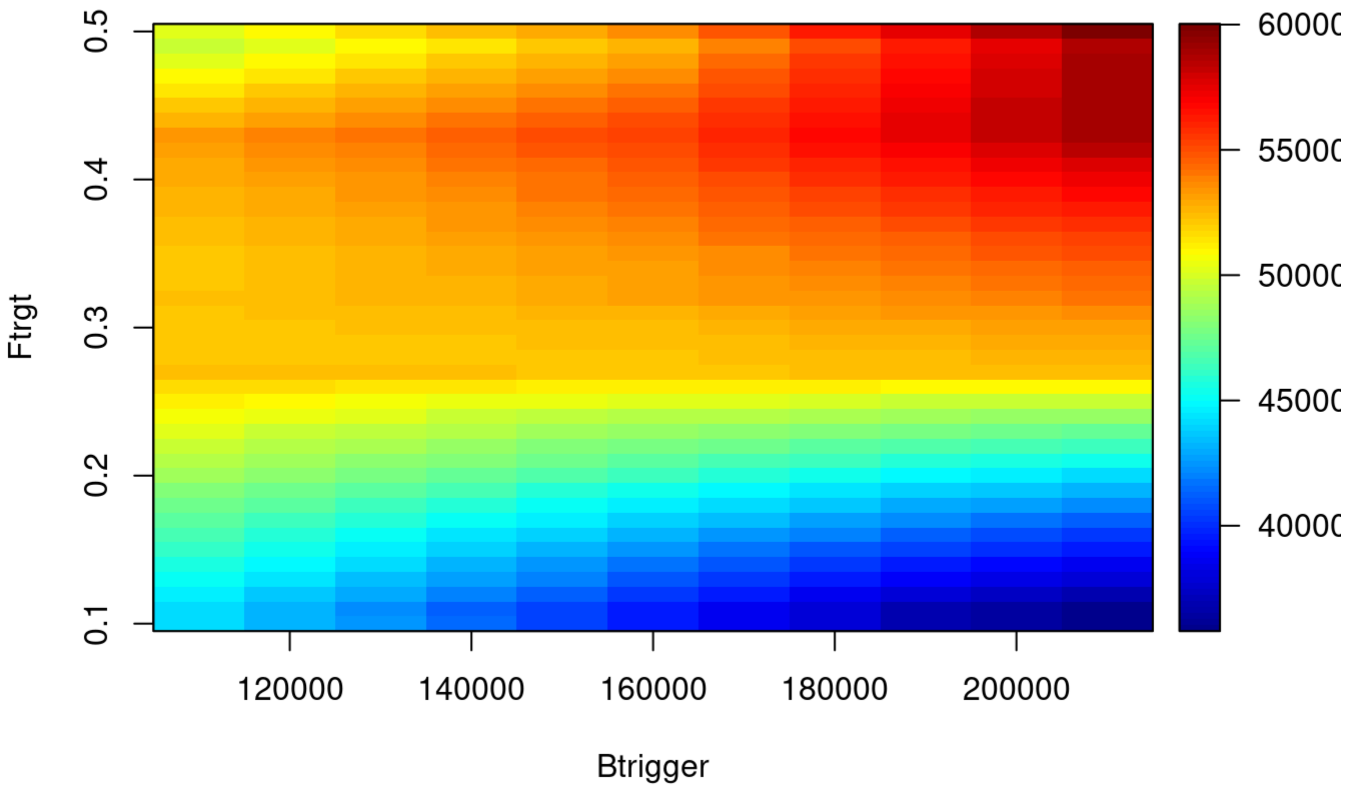
```
# now lets look at the probability that the risk is less than or equal to 0.05
prisk <- pnorm(log(0.05),pred_risk1_g$mean,pred_risk1_g$sd+1e-12)
```

```
# plot it
image2D(matrix(prisk,nrow=11),y=sort(unique(dat$Ftrgt)),x=sort(unique(dat$Btrigger)),x
```

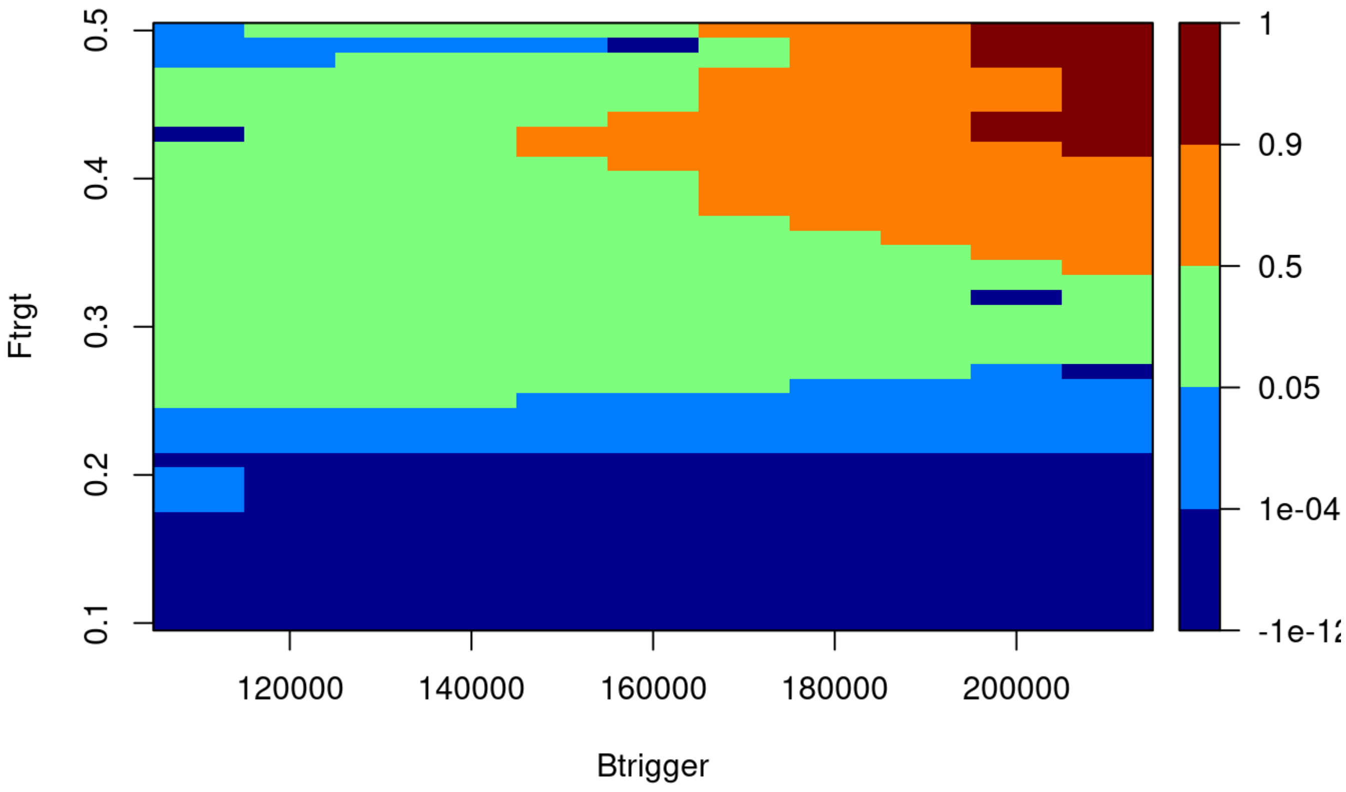


```
# Predicts mean and uncertainty for log(catch) at each parameter combination in gridd
# SK means simple kriging
pred_cat_g <- predict(gp_cat,newdata=gridd,type="SK")

# Best catch observed so far where risk is below 0.05
current_max <- max(runs$C_long[runs$risk3_long < 0.05])

#Convert to max1 log scale as emulator trained on log(catch)
# pcat1 is the probability that the catch is less than or equal to max1 (the best safe
pcat1 <- pnorm(log(current_max),pred_cat_g$mean,pred_cat_g$sd+1e-12)

# mark which points in gridd are still good candidates
eps <- 1e-4
possible <- (apply(cbind((1-pcat1) , prisk),1,min) >  eps)

# collect final points - this is so we know what to do in round 2
pot_points1 <- gridd[possible,]

#Removes already evaluated points from those that are still possible so don't evalute a
tmp <-setdiff(pot_points1,runs[,1:2])
pot_points1 <- tmp


# getting catch out of log(catch) for each point in gridd
```

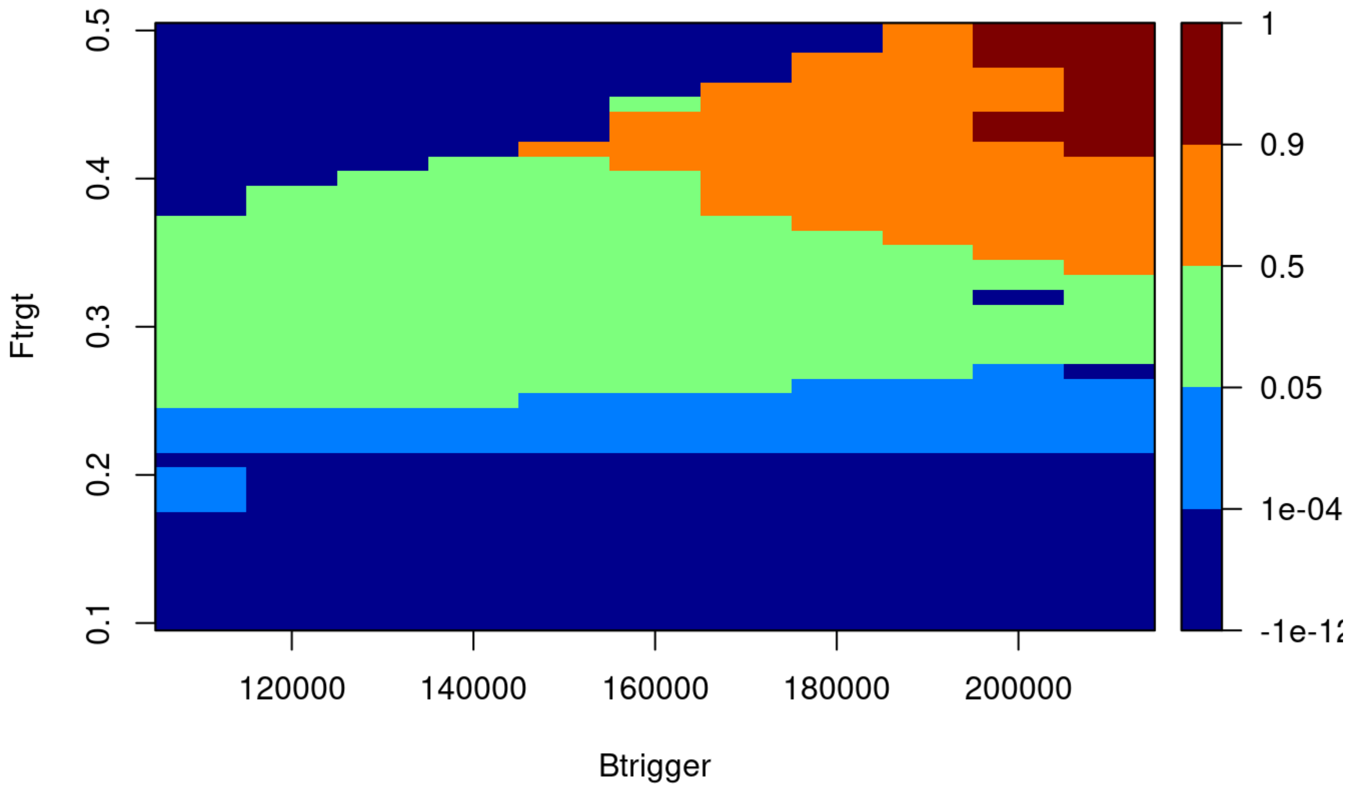
```
med_cat1 <- exp(pred_cat_g$mean)


# Below produces heat maps for round 1
image2D(matrix(med_cat1,nrow=11),y=sort(unique(dat$Ftrgt)),x=sort(unique(dat$Btrigger)
```



```
image2D(matrix(1-pcat1,nrow=11),y=sort(unique(dat$Ftrgt)),x=sort(unique(dat$Btrigger))
```



```
# Visualises region that has risk <0.05 and better catch than best evaluated so far
image2D(matrix(possible * (1-pcat1),nrow=11),y=sort(unique(dat$Ftrgt)),x=sort(unique(da
```



```r
# If no plausible points remain, stop the process
iteration <- 1
if (sum(possible) == 0) {
  stop("No plausible points remaining at round", iteration, "\n")
  ans<-unrescale_Her(pot_points1[1,],dat1)
  ans
}

cat("Plausible points remaining:", sum(possible), "\n")
```

```
Plausible points remaining: 258
```

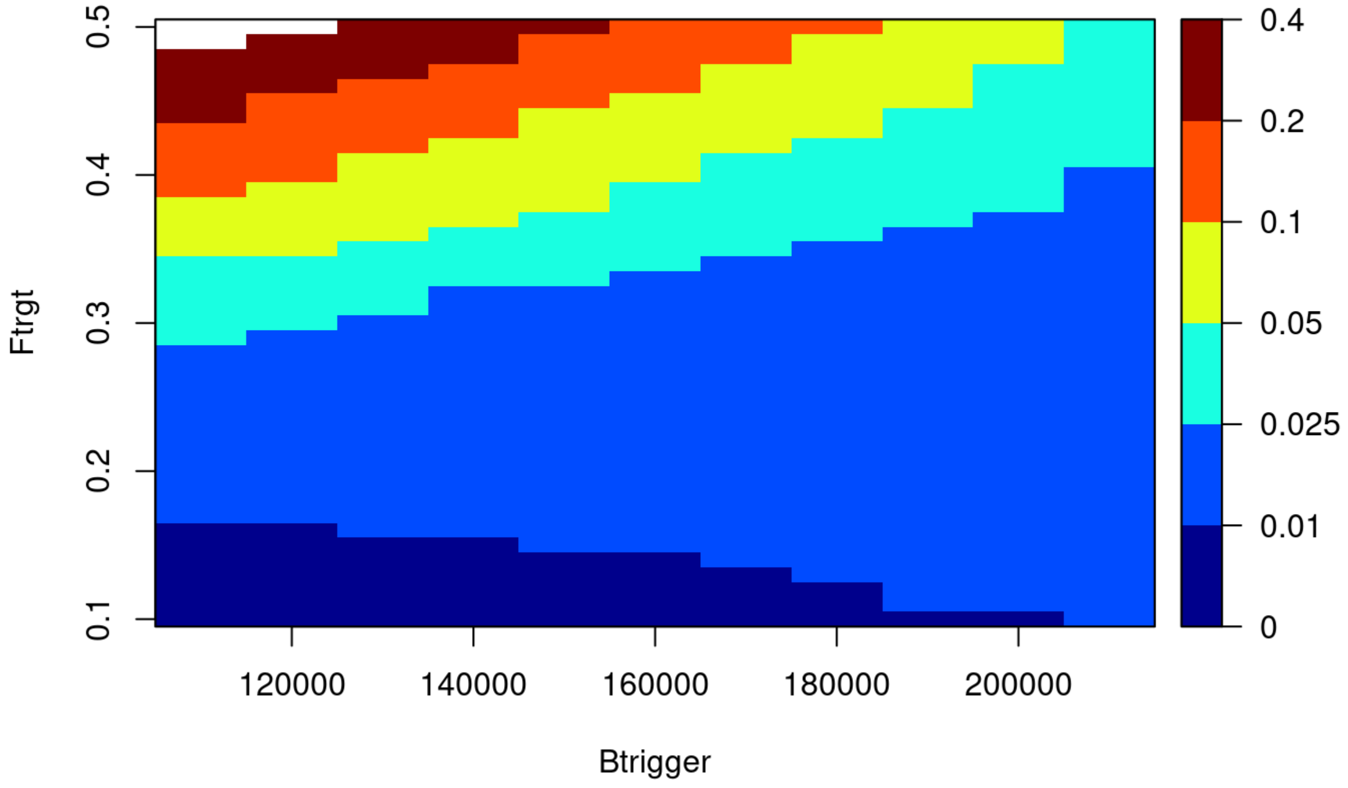
```r
cat("Current maximum", current_max)
```

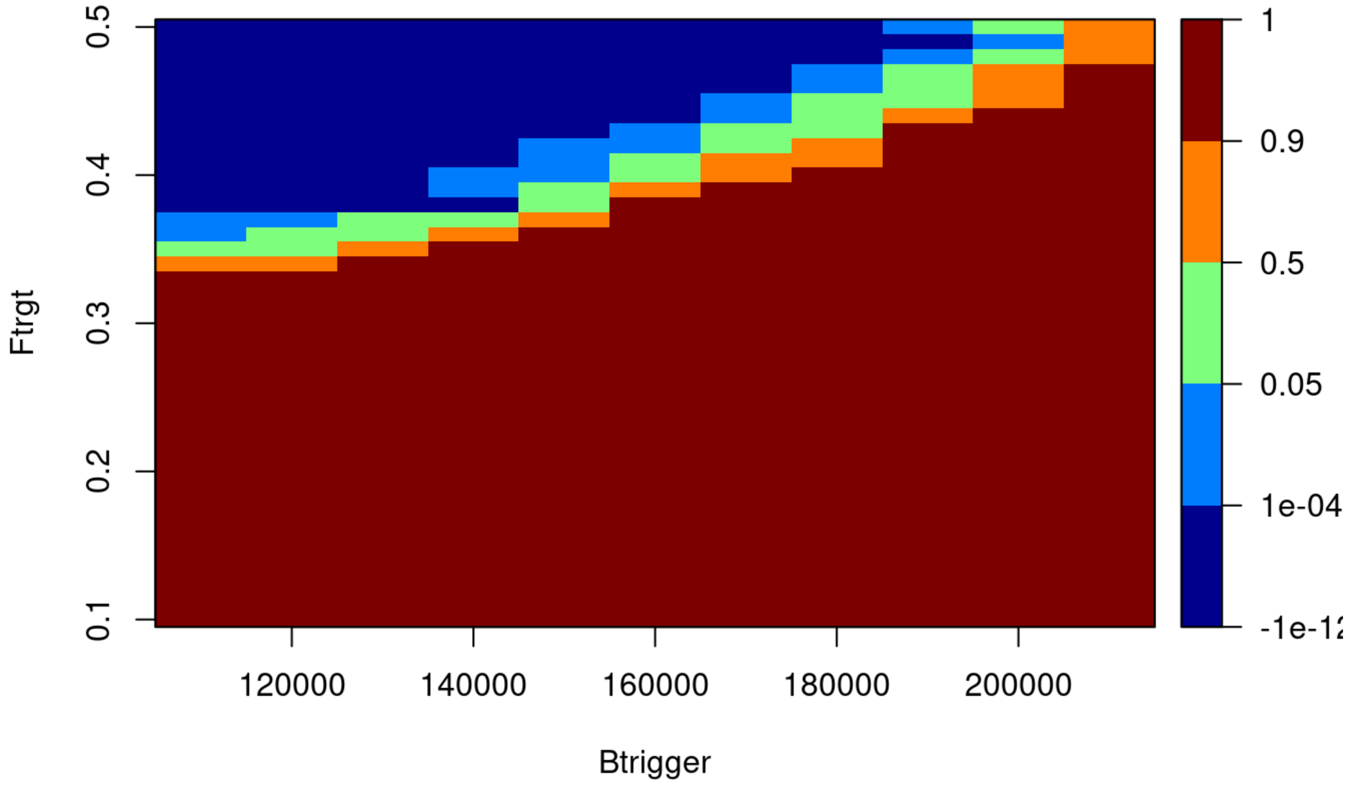
```
Current maximum 54596.5
```

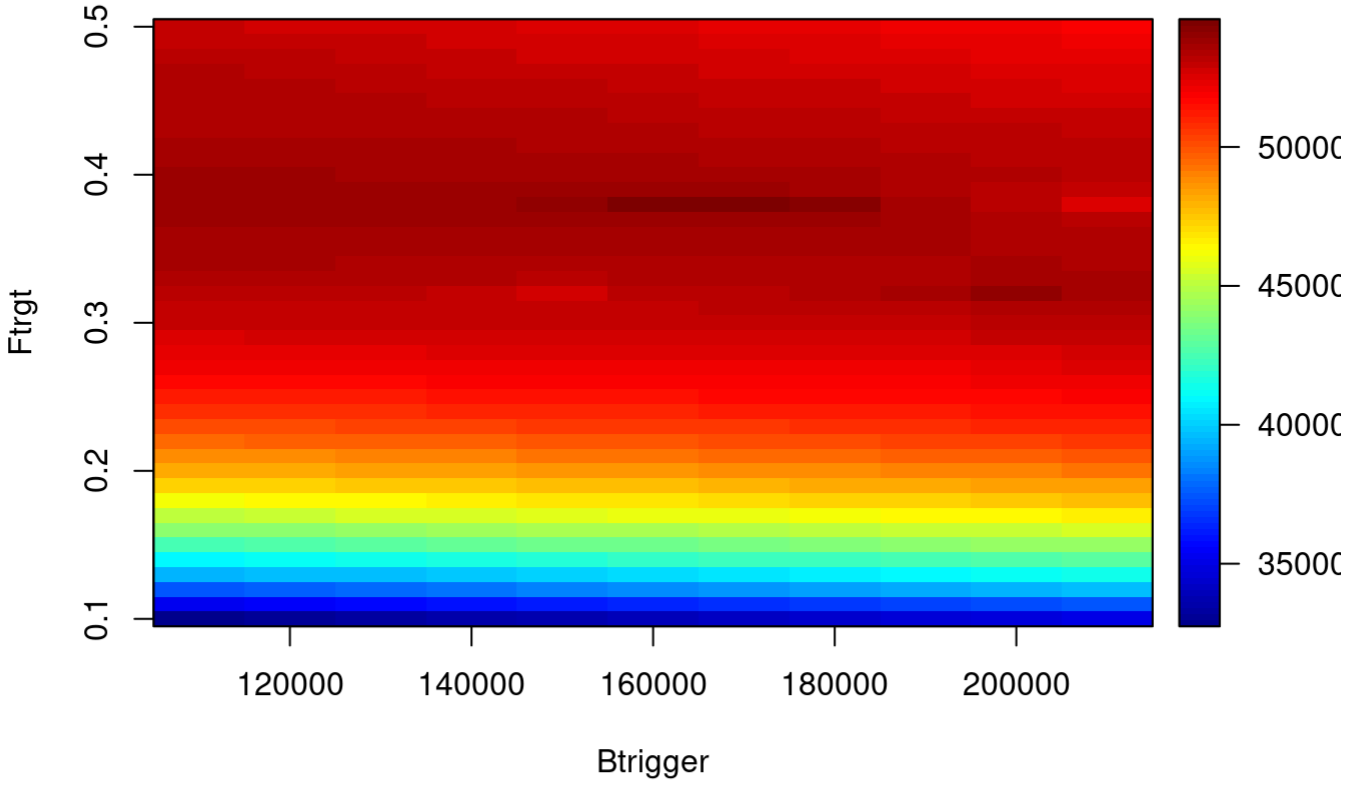
```r
# Store all rounds
all_rounds <- round1
all_rounds$Round <- 1

# ITERATIVE ROUNDS
max_rounds <- 20
round_num <- 1

#TODO: Can I rename the variables ion each round so I can clearly see the progression?
```

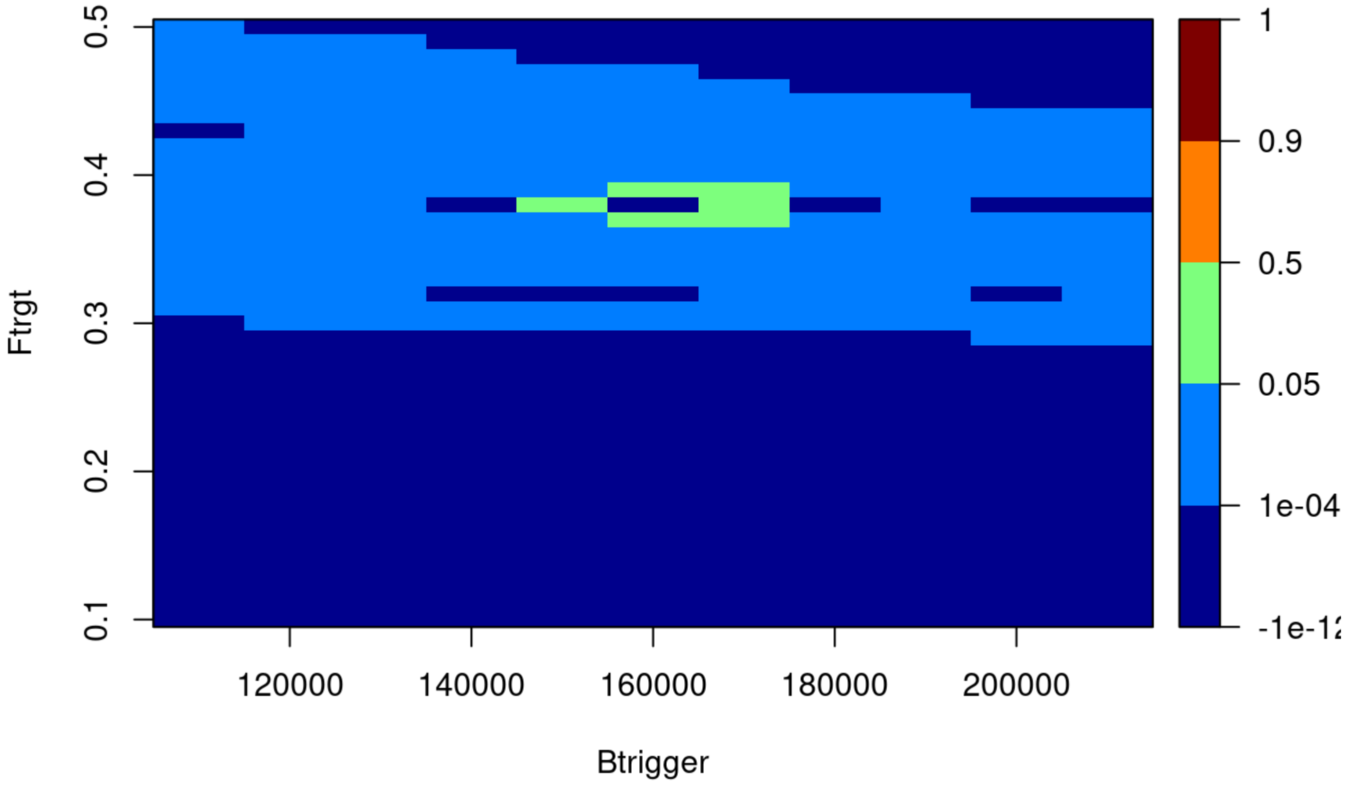
```
for (iteration in 2:max_rounds) {

  cat("\n Round", iteration, "\n")

  # Calculate KG
  mu <- pred_cat_g$mean
  sigma <- pred_cat_g$sd
  #TDOD: Put in correct X_pred
  ei <- knowledge_gradient_sim(mu, sigma, gp_cat, obs_noise_var = 0, nsim = 100, pred_

  # Create candidate set
  gridd_with_ei <- gridd
  gridd_with_ei$ei <- ei
  gridd_with_ei$possible <- possible

  cand <- subset(gridd_with_ei, possible == TRUE & ei > 0)

  # Create keys for filtering
  cand$key <- paste(cand$Ftarget, cand$Btrigger, sep = "_")
  runs$key <- paste(runs$Ftarget, runs$Btrigger, sep = "_")

  # Remove already evaluated points
  cand <- cand[!(cand$key %in% runs$key), ]

  # Check if candidates exhausted
  if (nrow(cand) == 0) {
    cat("No unevaluated candidates with positive EI. Stopping at round", iteration, "\
    break
  }

  cat("Unevaluated candidates with EI > 0:", nrow(cand), "\n")

  # Select next points
  if (nrow(cand) <= 8) {
    next_points <- cand[order(-cand$ei), c("Ftarget", "Btrigger")]
  } else {
    top_candidates <- cand[order(-cand$ei), ][1:nrow(cand), ]
    set.seed(123)
    km_result <- kmeans(top_candidates[, c("Ftarget", "Btrigger")], centers = 8)
    top_candidates$cluster <- km_result$cluster
    next_points <- do.call(rbind, lapply(split(top_candidates, top_candidates$cluster),
      df[which.max(df$ei), c("Ftarget", "Btrigger", "ei")]
    }))
  }

  # Unrescale and prepare for evaluation
  coords <- next_points[, c("Ftarget", "Btrigger")]
  new_points <- signif(unrescale_Her(coords, dat1), 2)
  names(new_points) <- c("Ftrgt", "Btrigger")
  new_points$Round <- iteration

  all_rounds <- rbind(all_rounds, new_points[, c("Ftrgt", "Btrigger", "Round")])

  new_round <- rbind(
    all_rounds[all_rounds$Round < iteration, c("Ftrgt", "Btrigger")],
    new_points[, c("Ftrgt", "Btrigger")]
  )

  dat_run <- objective_func(new_round, lookup_data = dat)
  runs <- rescale_Her(dat_run, dat = dat1)

  res_cat <- log(runs$C_long)
  invisible(capture.output(
  gp_cat <- km(~I(log(Ftarget+0.1)^2) + I(log(Ftarget+0.1)) + I(log(Ftarget+0.1)^3) + 
  ))

  res_risk <- log(runs$risk3_long)
  invisible(capture.output(
  gp_risk <- km(~.^2, design = runs[, c("Ftarget", "Btrigger")], estim.method = "MLE",
  ))


  pred_risk_g <- predict(gp_risk, newdata = gridd, type = "SK")
  pred_cat_g <- predict(gp_cat, newdata = gridd, type = "SK")

  # PLOTTING
  med_risk <- exp(pred_risk_g$mean)
  image2D(matrix(med_risk, nrow=11), breaks=c(0,0.01,0.025,0.05,0.1,0.2,0.4), y=sort(u

  prisk <- pnorm(log(0.05), pred_risk_g$mean, pred_risk_g$sd + 1e-12)
  image2D(matrix(prisk, nrow=11), y=sort(unique(dat$Ftrgt)), x=sort(unique(dat$Btrigge

  current_max <- max(runs$C_long[runs$risk3_long < 0.05])
  pcat <- pnorm(log(current_max), pred_cat_g$mean, pred_cat_g$sd + 1e-12)

  possible <- (apply(cbind((1 - pcat), prisk), 1, min) > eps)

  med_cat <- exp(pred_cat_g$mean)
  image2D(matrix(med_cat, nrow=11), y=sort(unique(dat$Ftrgt)), x=sort(unique(dat$Btrig
  image2D(matrix(1-pcat, nrow=11), y=sort(unique(dat$Ftrgt)), x=sort(unique(dat$Btrigg
  image2D(matrix(possible * (1-pcat), nrow=11), y=sort(unique(dat$Ftrgt)), x=sort(uniq

  #STOPPING CRITERION
  if (sum(possible) == 0) {
    cat("No plausible points remaining. Stopping at round", iteration, "\n")
    break
  }

  cat("Plausible points remaining:", sum(possible), "\n")
  cat("Best catch so far:", current_max, "\n")

  round_num <- iteration
}
```

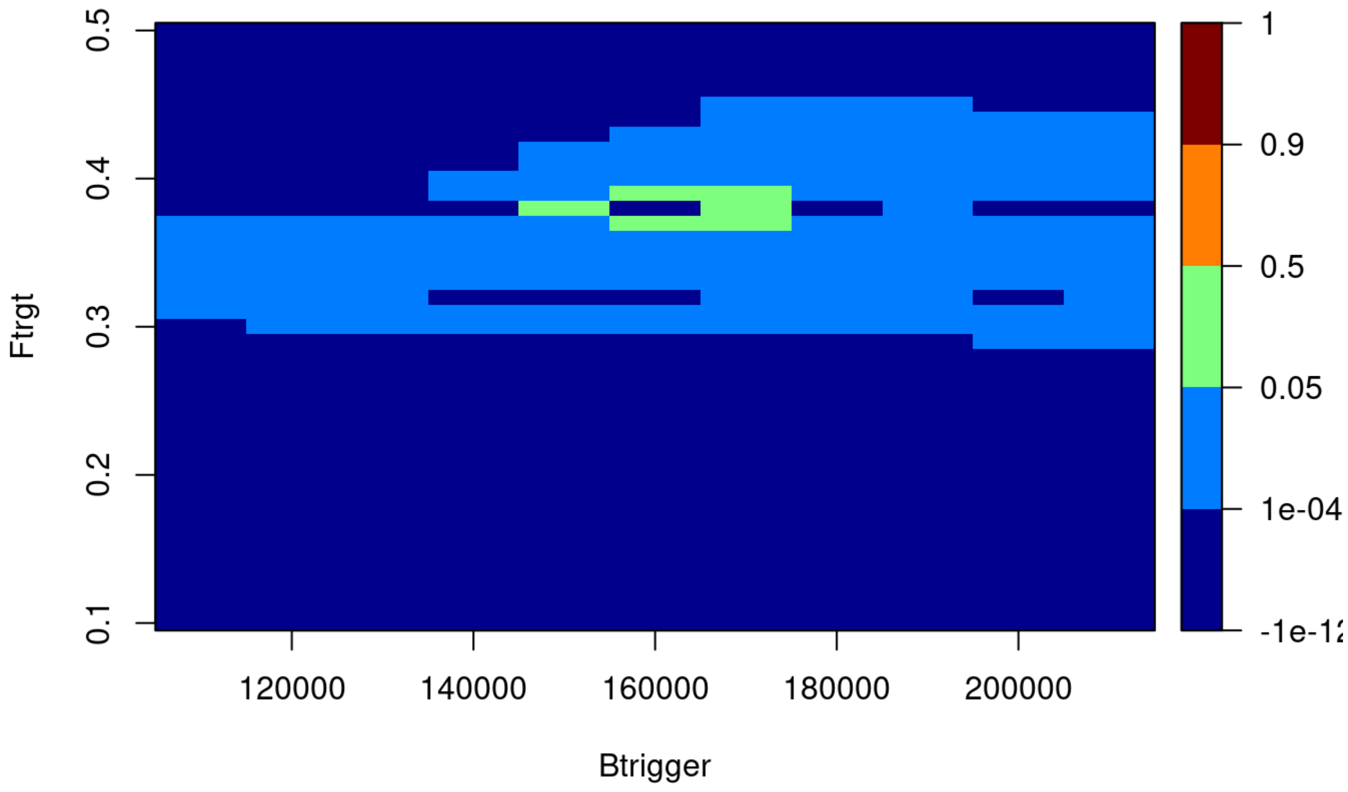
```
 Round 2
Unevaluated candidates with EI > 0: 257
```


Ftrgt
0.1
0.2
0.3
0.4
0.5
120000
140000
160000
180000
200000
Btrigger
0.4
0.2
0.1
0.05
0.025
0.01
0

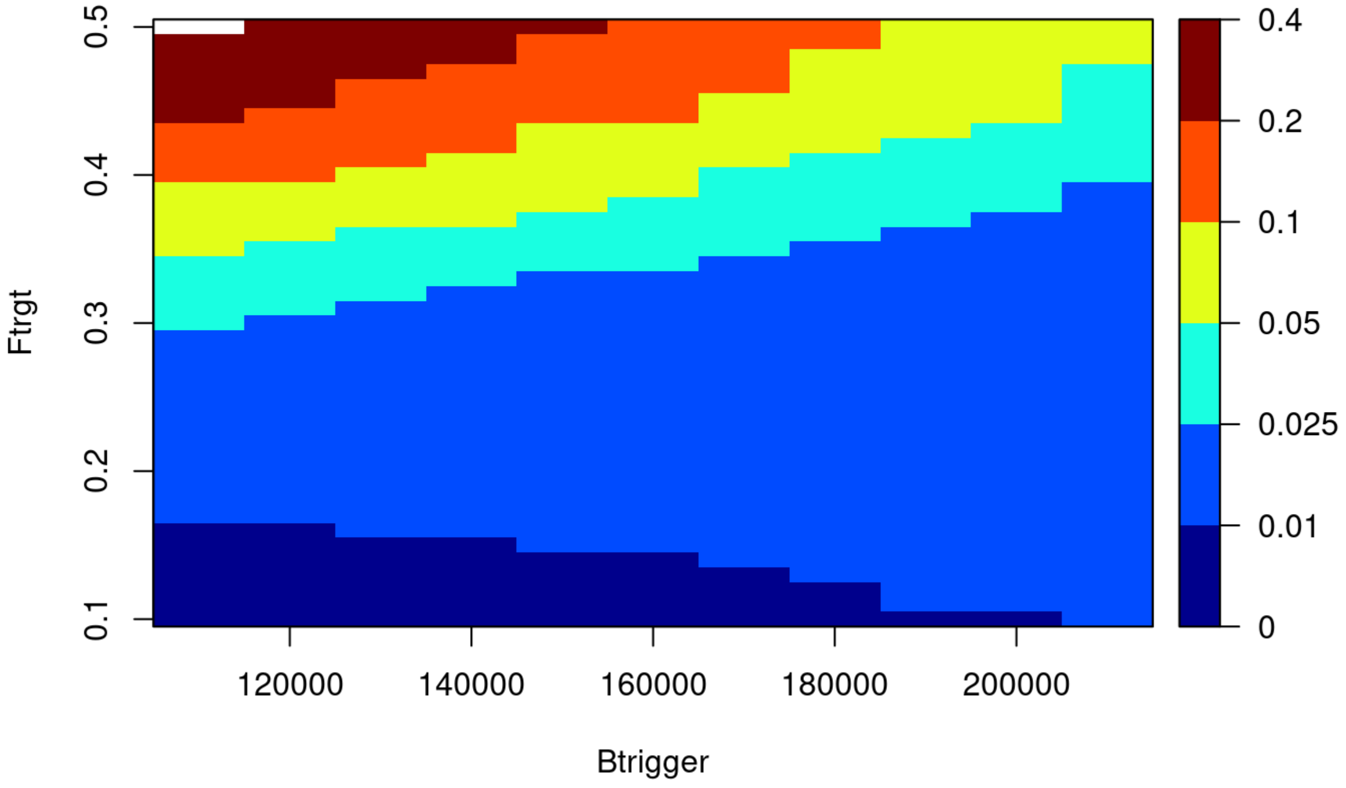

Ftrgt
0.1
0.2
0.3
0.4
0.5
120000
140000
160000
180000
200000
Btrigger
1
0.9
0.5
0.05
1e-04
-1e-12

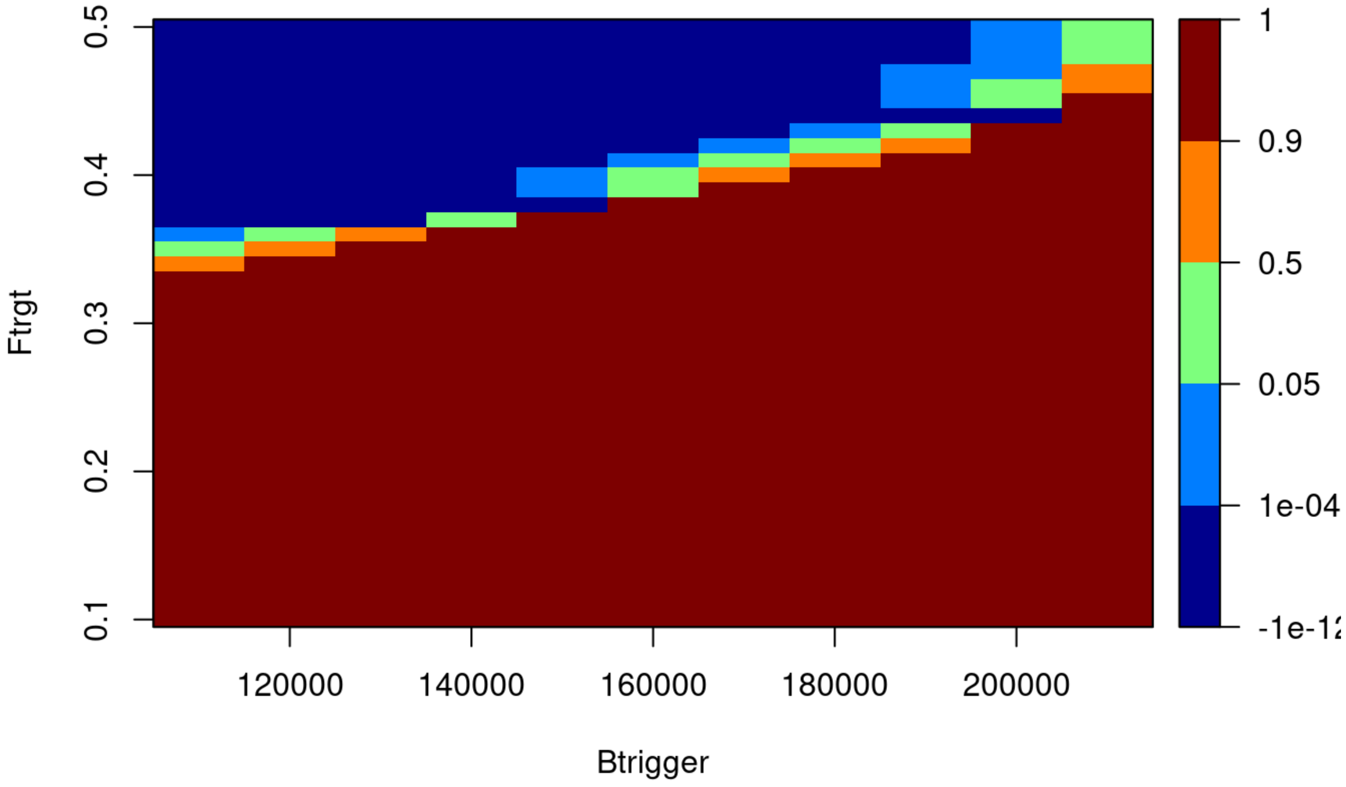

0.5
0.4
0.3
0.2
0.1
Ftrgt
120000
140000
160000
180000
200000
Btrigger
5000
4500
4000
3500

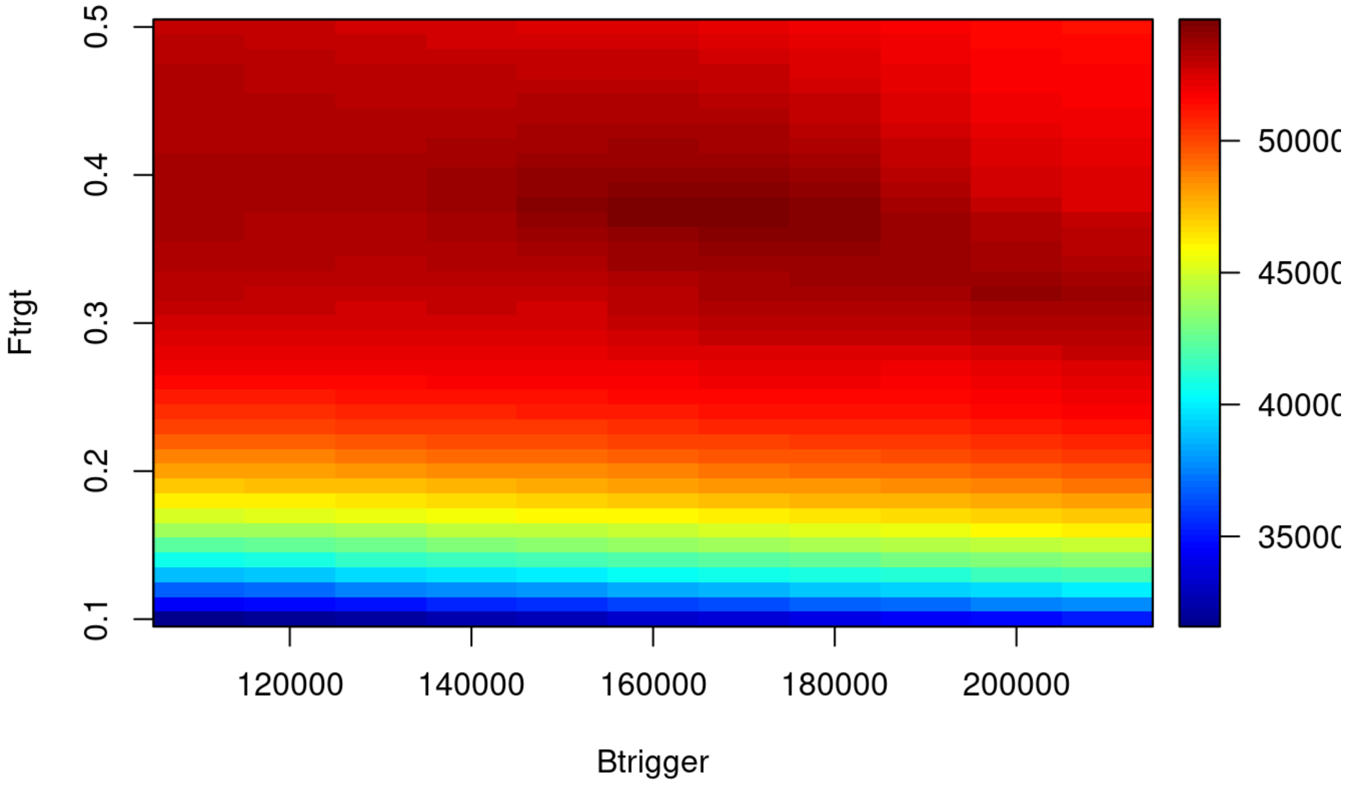

0.5
0.4
0.3
0.2
0.1
Ftrgt
120000
140000
160000
180000
200000
Btrigger
1
0.9
0.5
0.05
1e-04
-1e-1

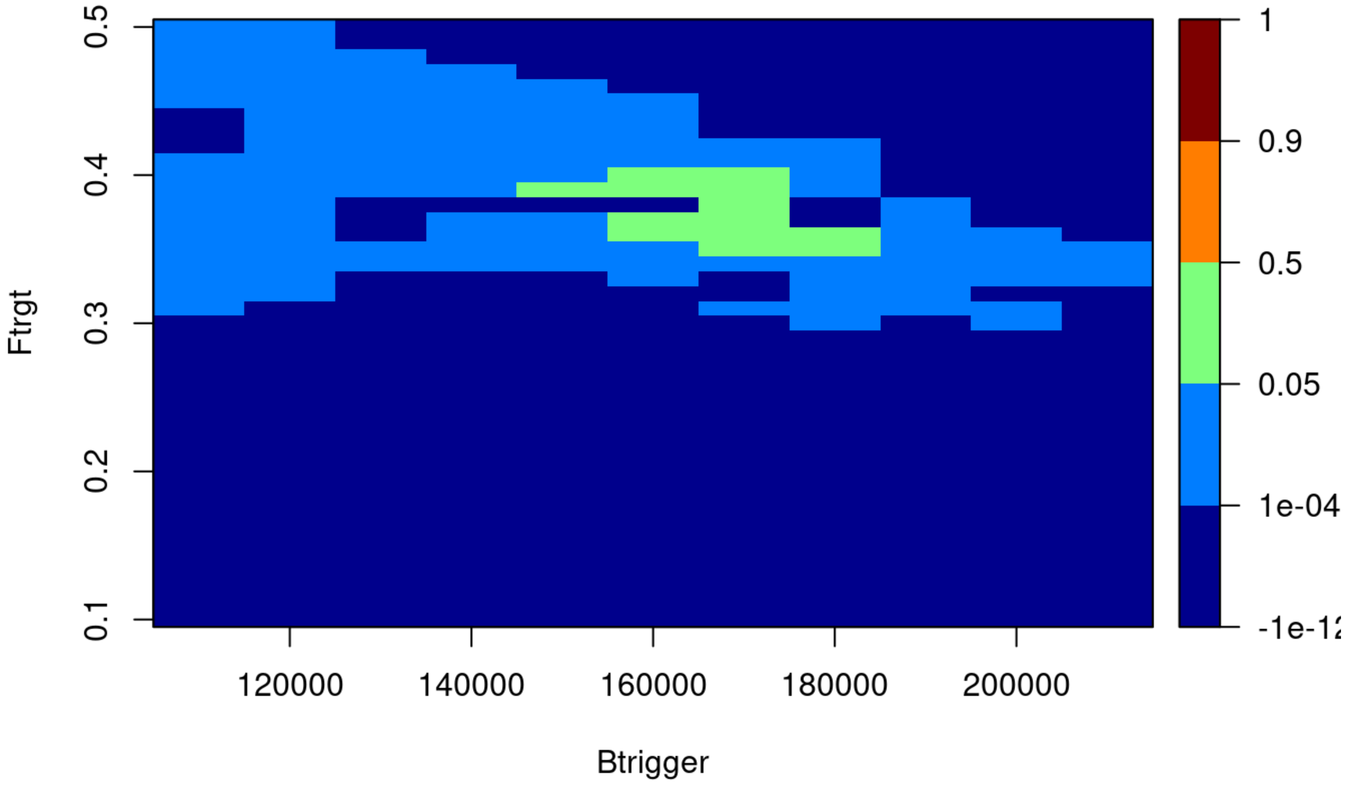


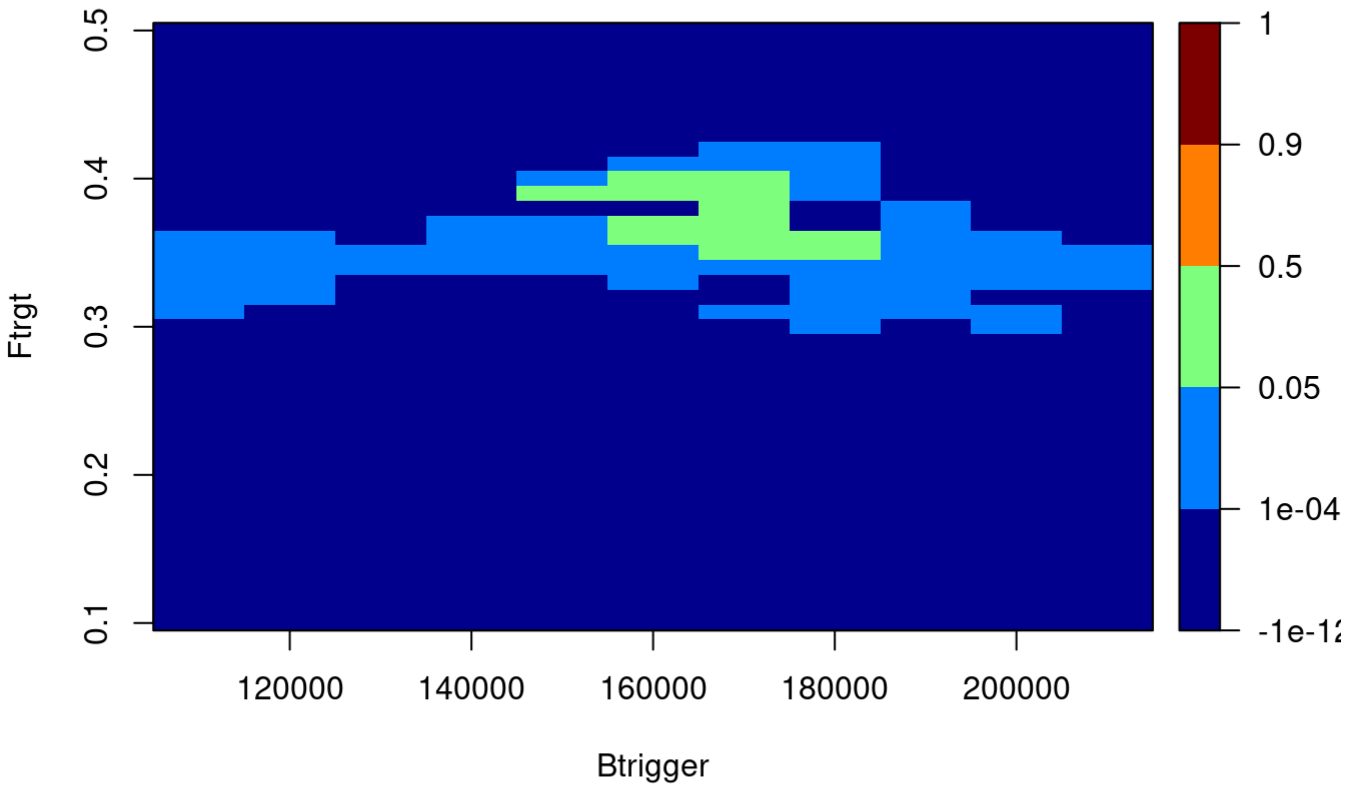
```
Plausible points remaining: 132
Best catch so far: 54596.5

 Round 3
Unevaluated candidates with EI > 0: 130
```


Ftrgt
0.1
0.2
0.3
0.4
0.5
120000
140000
160000
180000
200000
Btrigger
0.4
0.2
0.1
0.05
0.025
0.01
0

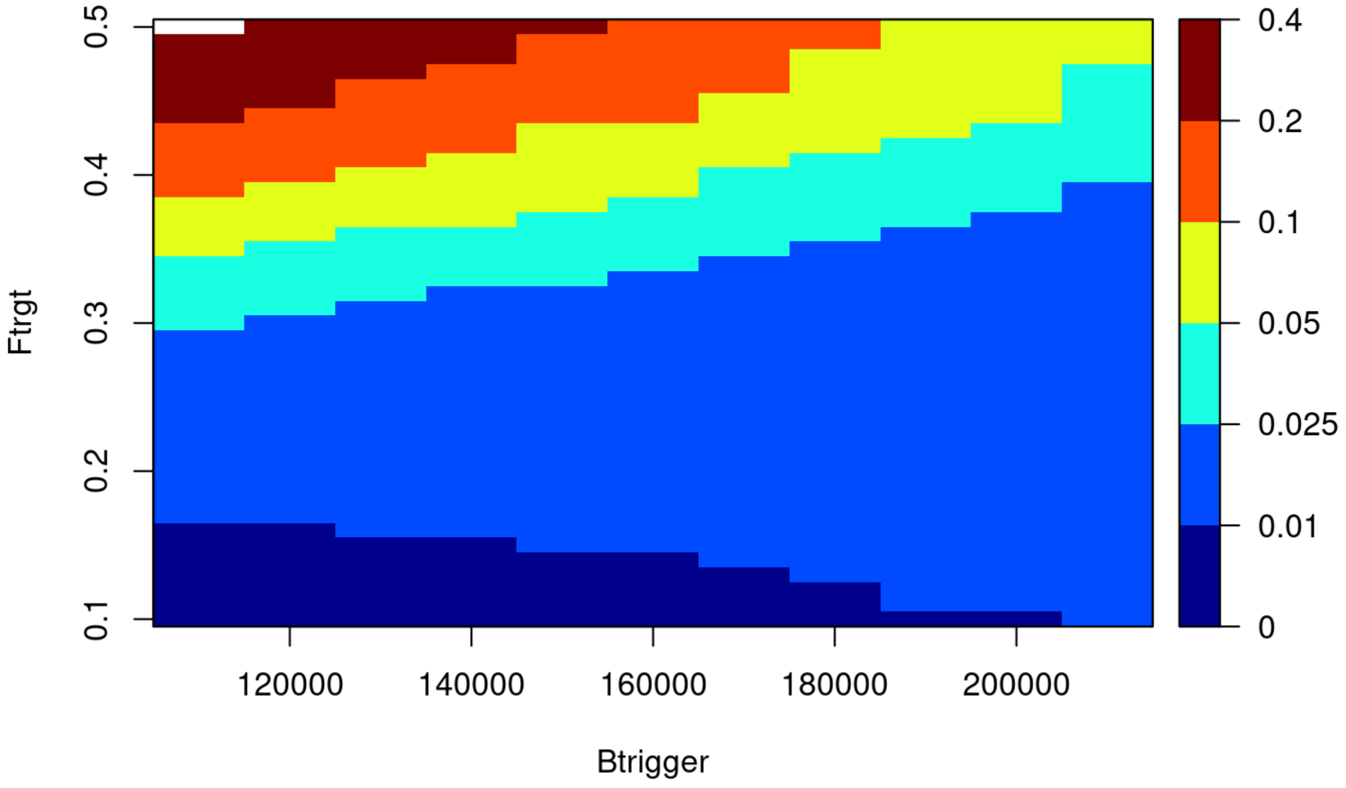

Ftrgt
0.1
0.2
0.3
0.4
0.5
120000
140000
160000
180000
200000
Btrigger
1
0.9
0.5
0.05
1e-04
-1e-12

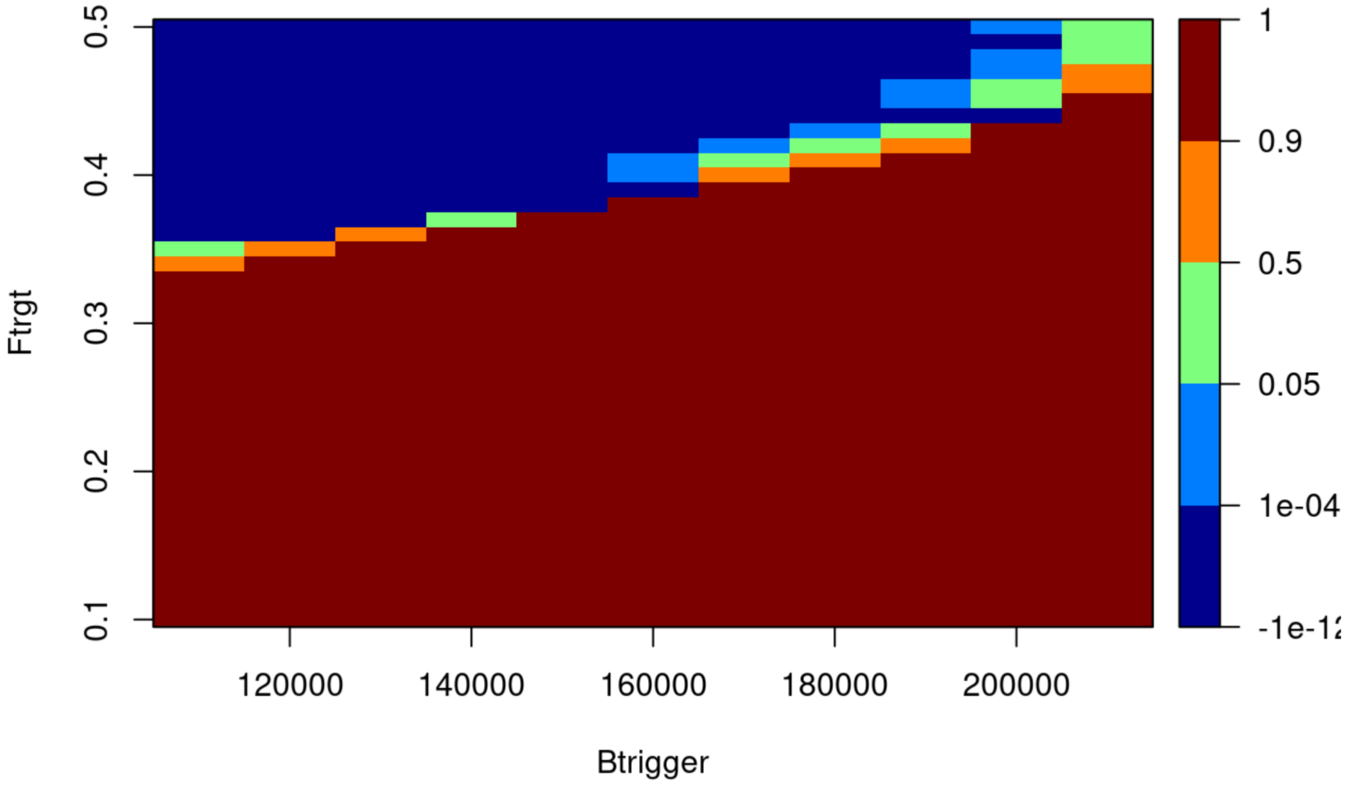

Ftrgt
0.1
0.2
0.3
0.4
0.5
120000
140000
160000
180000
200000
Btrigger
3500
4000
4500
5000

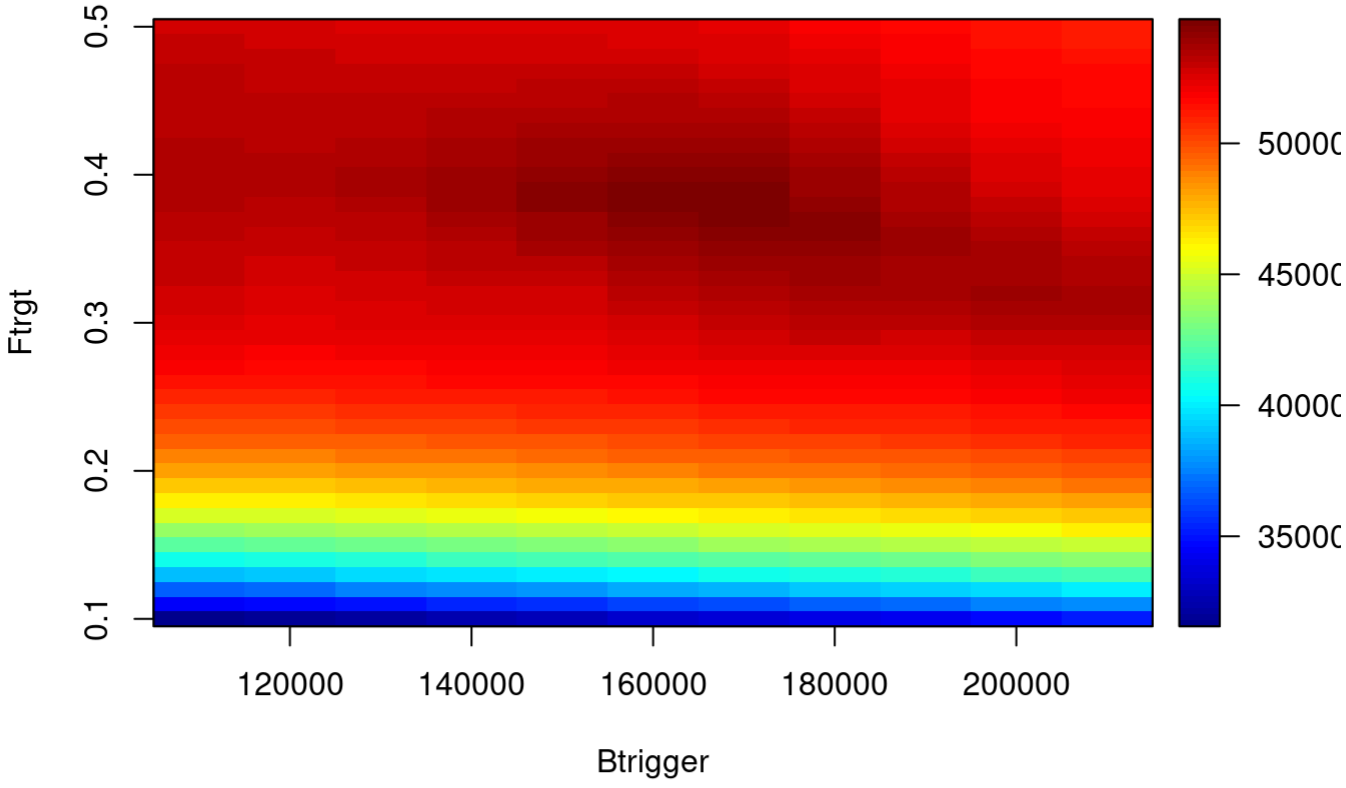

Ftrgt
0.1
0.2
0.3
0.4
0.5
120000
140000
160000
180000
200000
Btrigger
1
0.9
0.5
0.05
1e-04
-1e-1



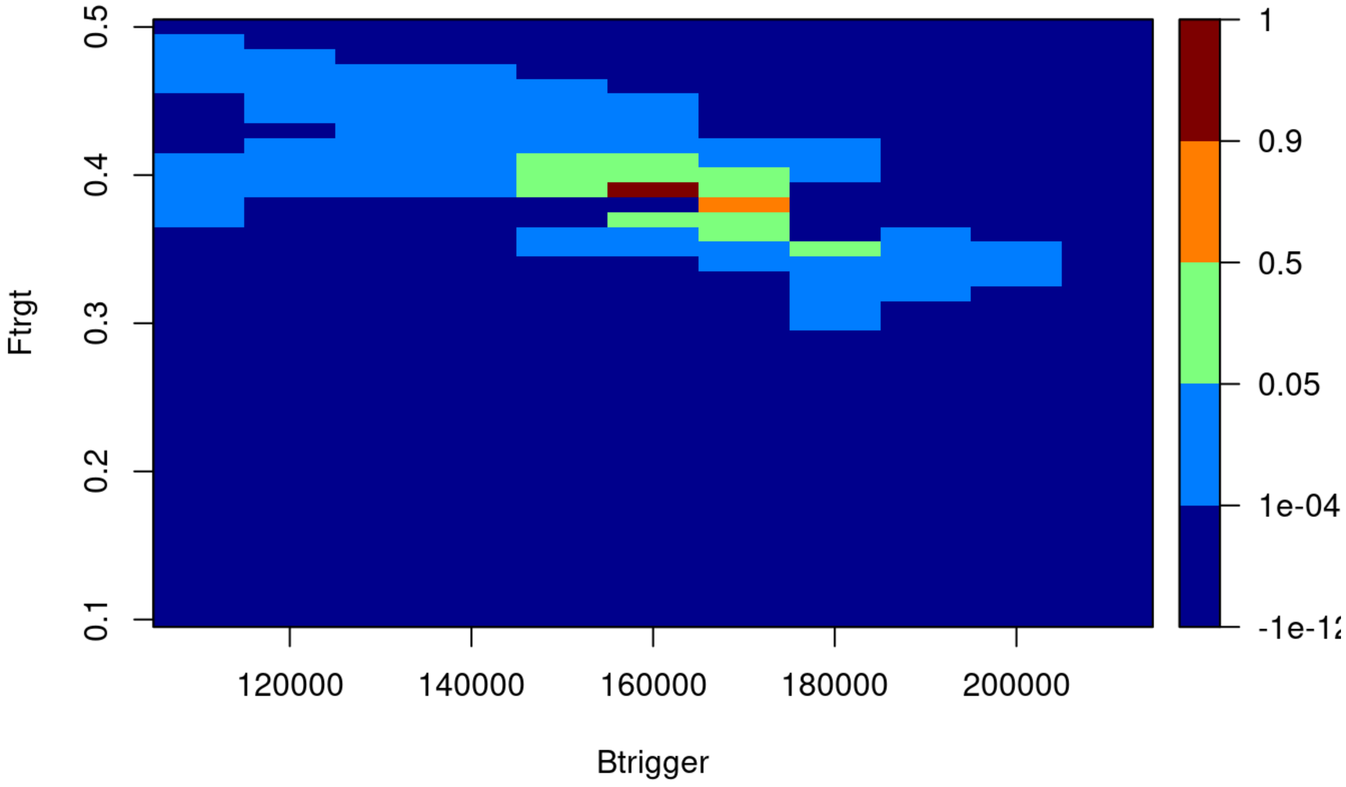
```
Plausible points remaining: 69
Best catch so far: 54596.5

 Round 4
Unevaluated candidates with EI > 0: 67
```


Ftrgt
0.1
0.2
0.3
0.4
0.5
120000
140000
160000
180000
200000
Btrigger
0.4
0.2
0.1
0.05
0.025
0.01
0

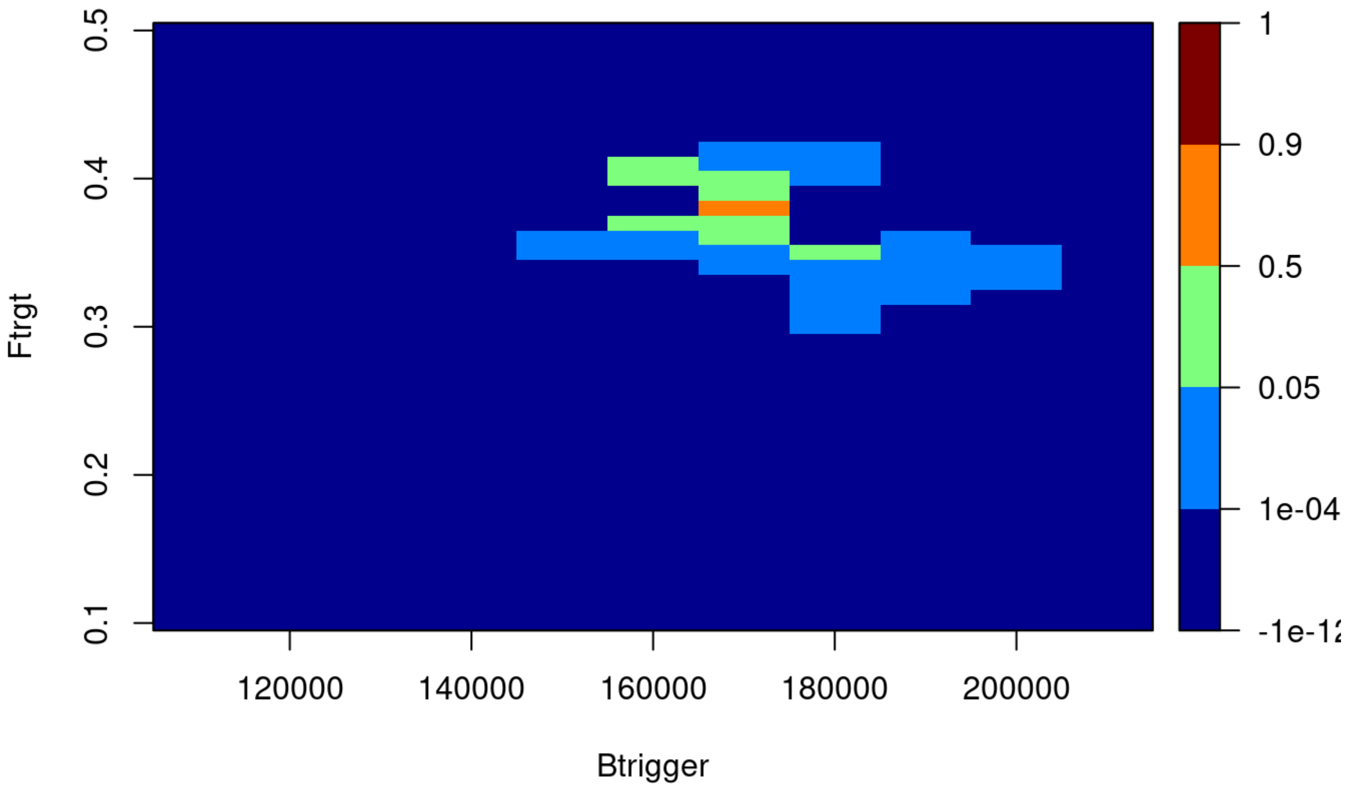

Ftrgt
0.1
0.2
0.3
0.4
0.5
120000
140000
160000
180000
200000
Btrigger
1
0.9
0.5
0.05
1e-04
-1e-12



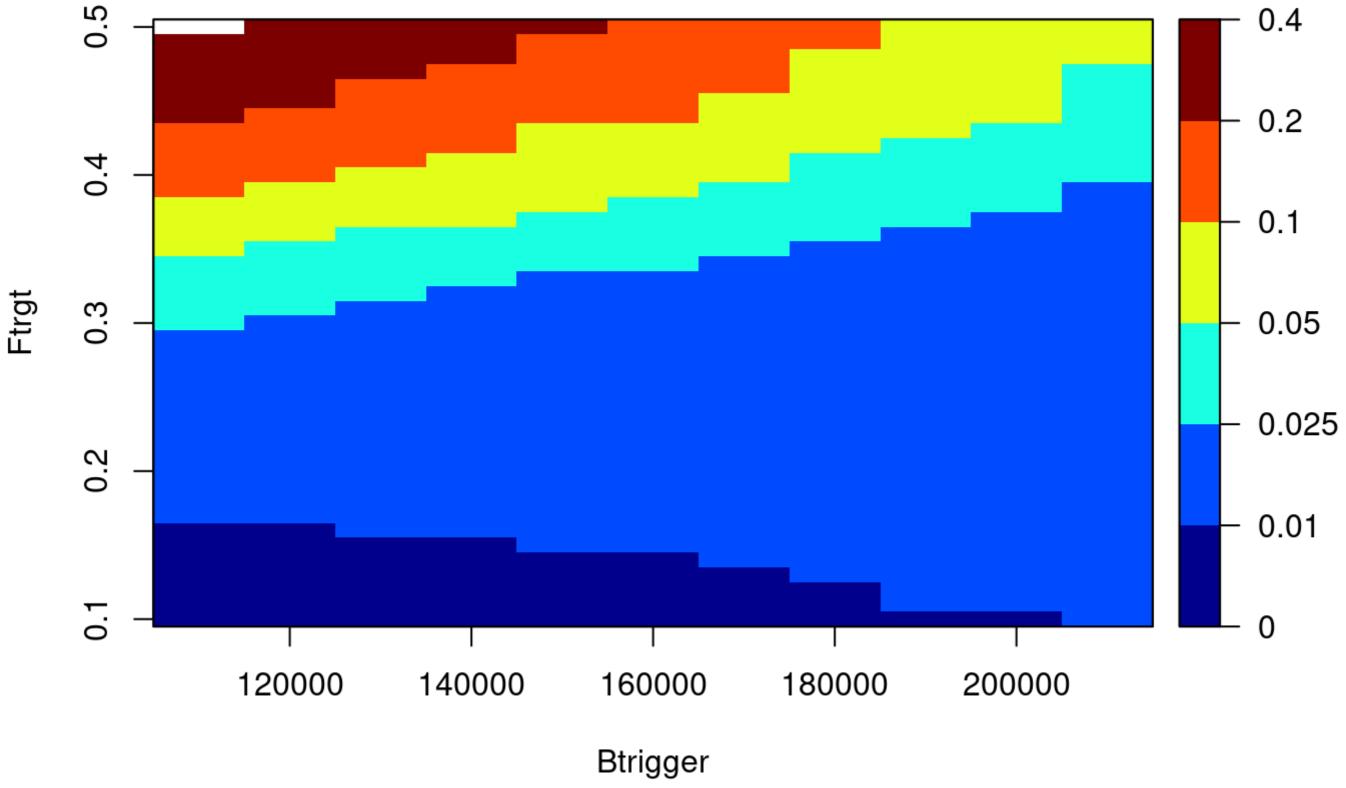
Ftrgt
0.1
0.2
0.3
0.4
0.5
120000
140000
160000
180000
200000
Btrigger
5000
4500
4000
3500

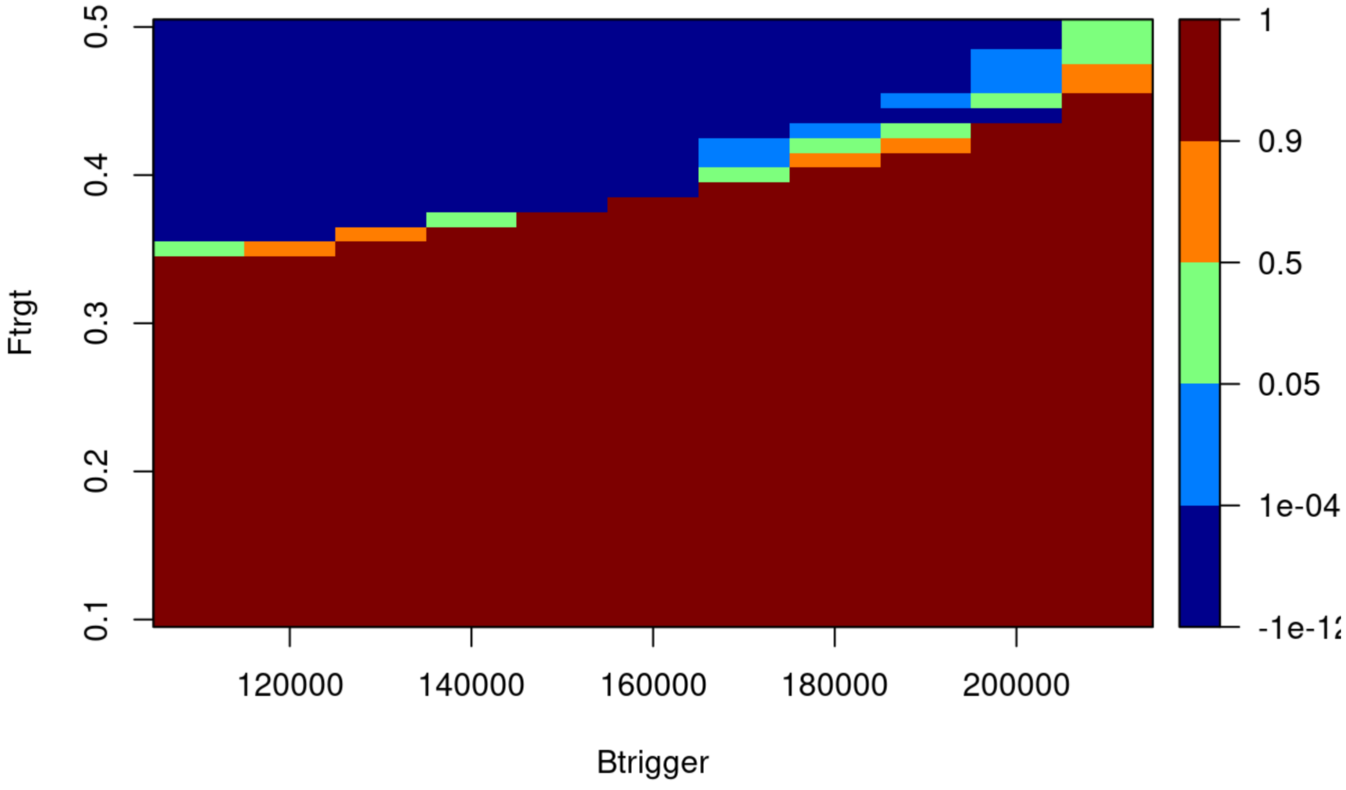


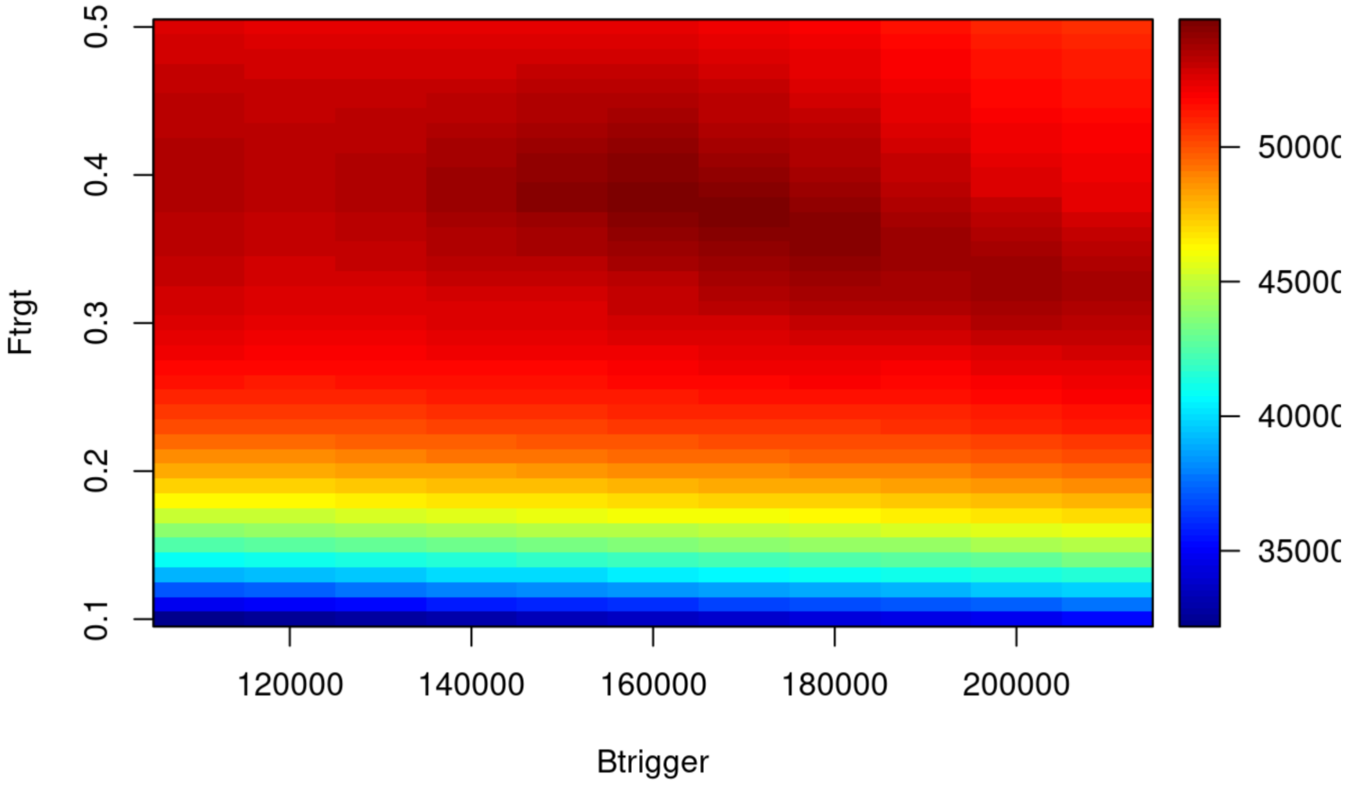
Ftrgt
0.1
0.2
0.3
0.4
0.5
120000
140000
160000
180000
200000
Btrigger
1
0.9
0.5
0.05
1e-04
-1e-1



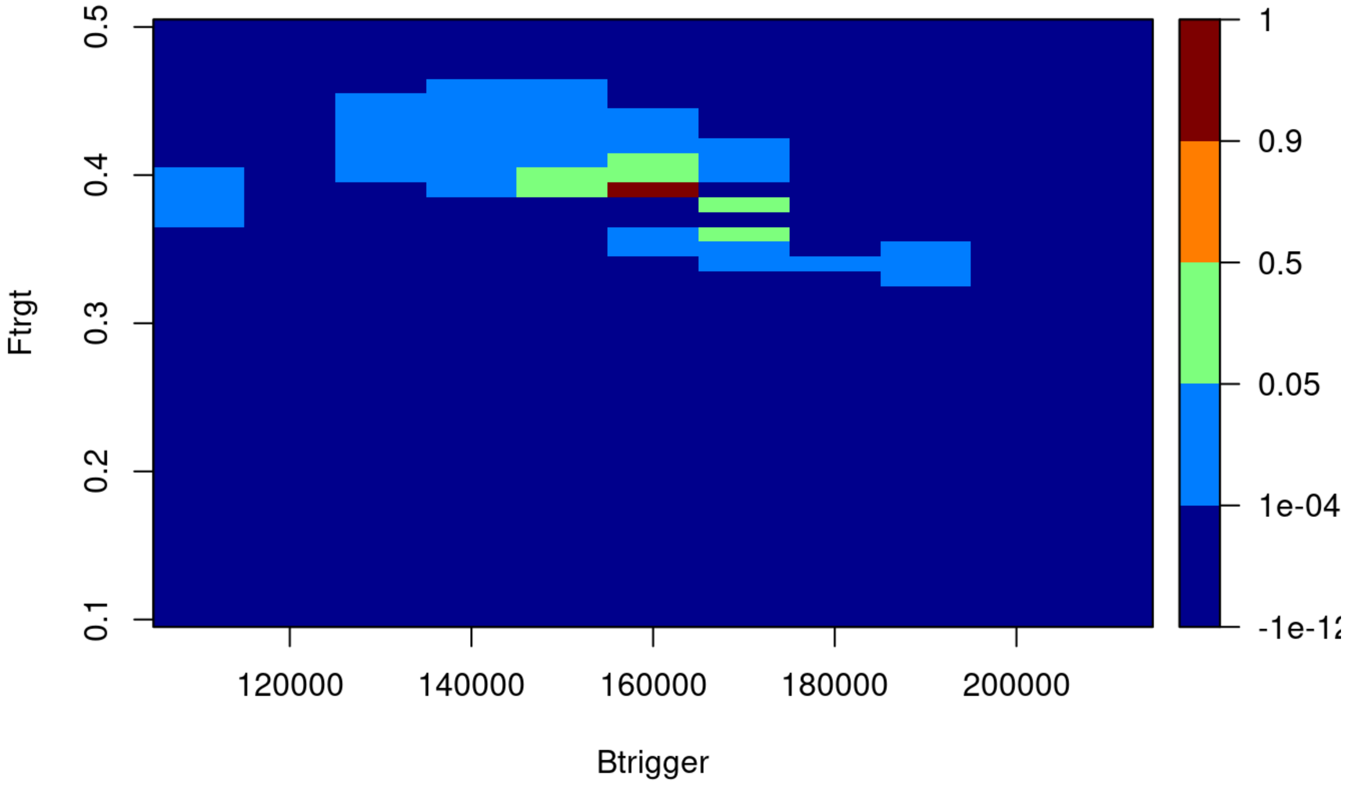
```
Plausible points remaining: 33
Best catch so far: 54596.5

 Round 5
Unevaluated candidates with EI > 0: 32
```


Ftrgt
0.1
0.2
0.3
0.4
0.5
120000
140000
160000
180000
200000
Btrigger
0.4
0.2
0.1
0.05
0.025
0.01
0

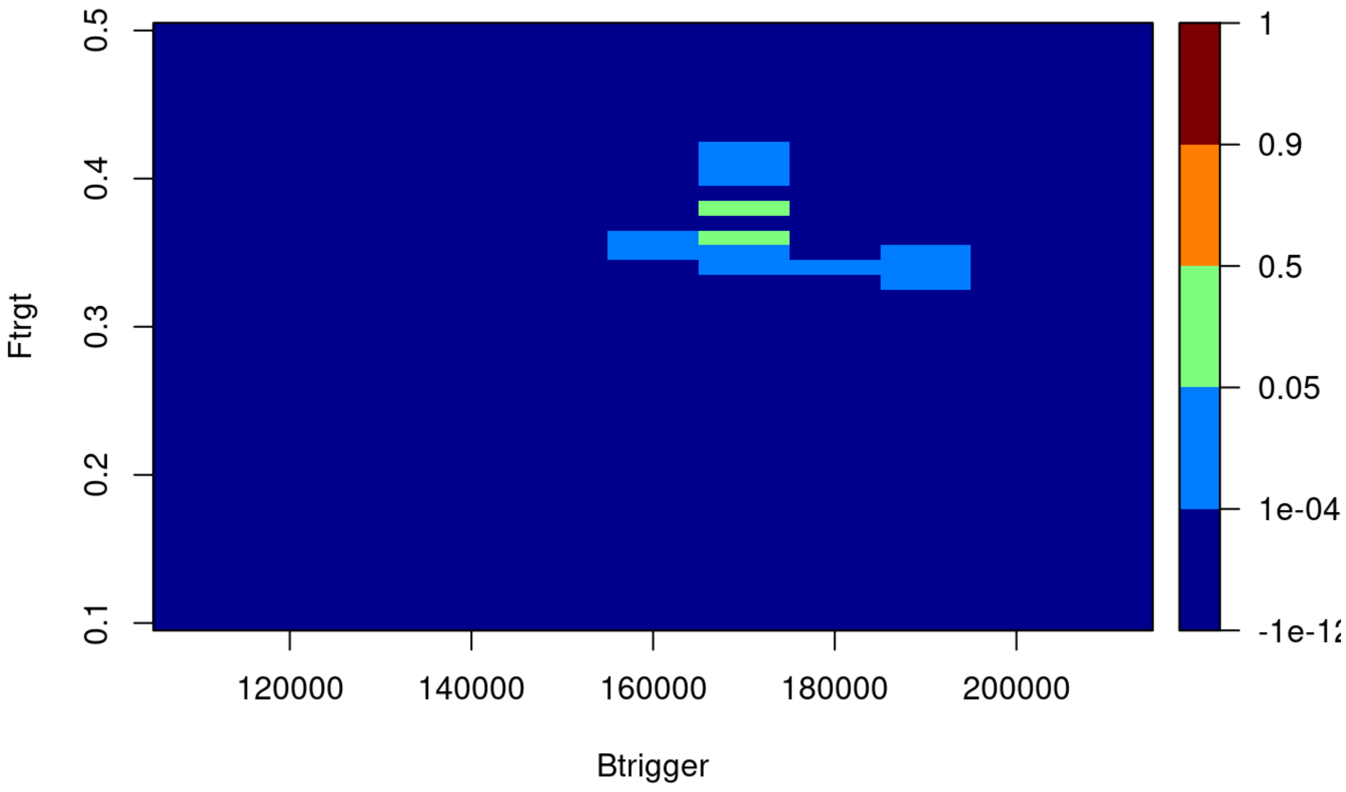

Ftrgt
0.1
0.2
0.3
0.4
0.5
120000
140000
160000
180000
200000
Btrigger
1
0.9
0.5
0.05
1e-04
-1e-12



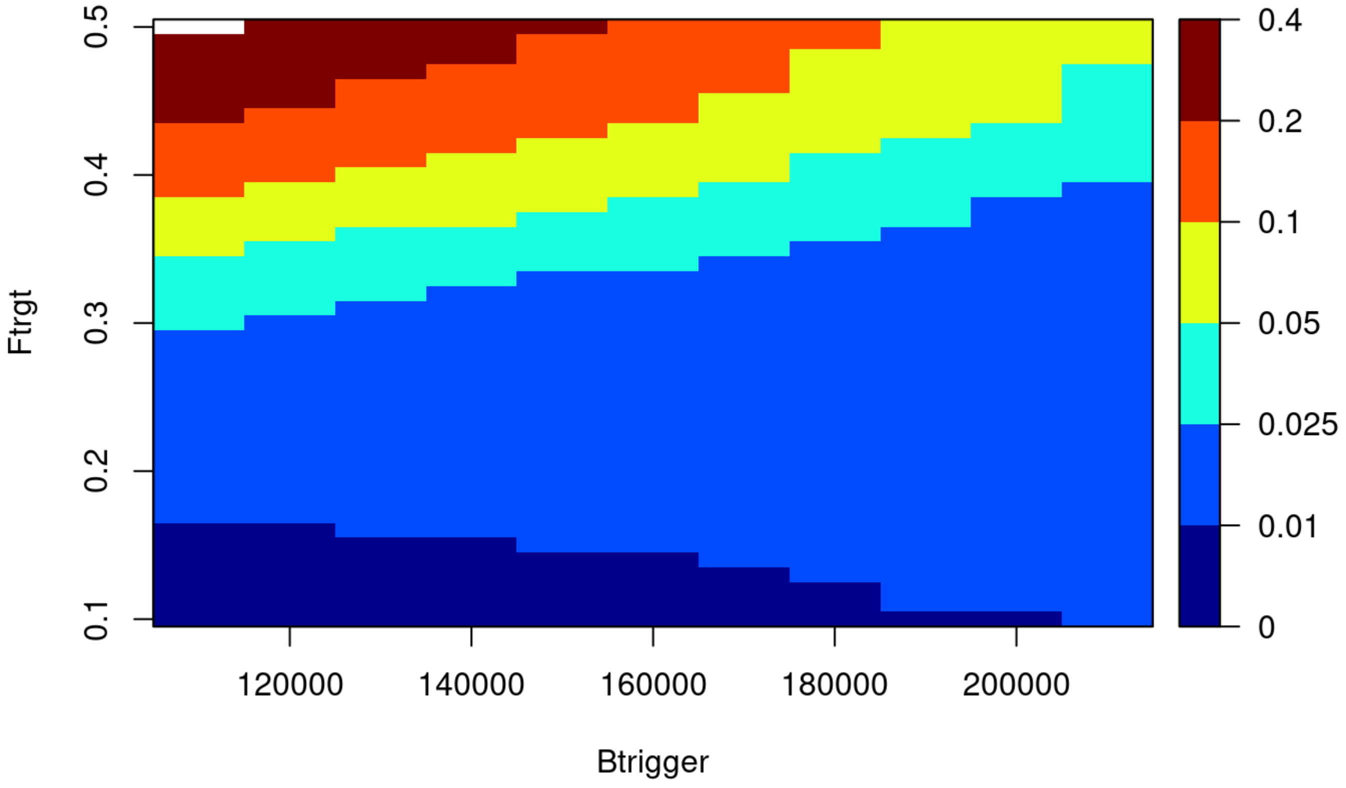
Ftrgt
0.1
0.2
0.3
0.4
0.5
120000
140000
160000
180000
200000
Btrigger
50000
45000
40000
35000

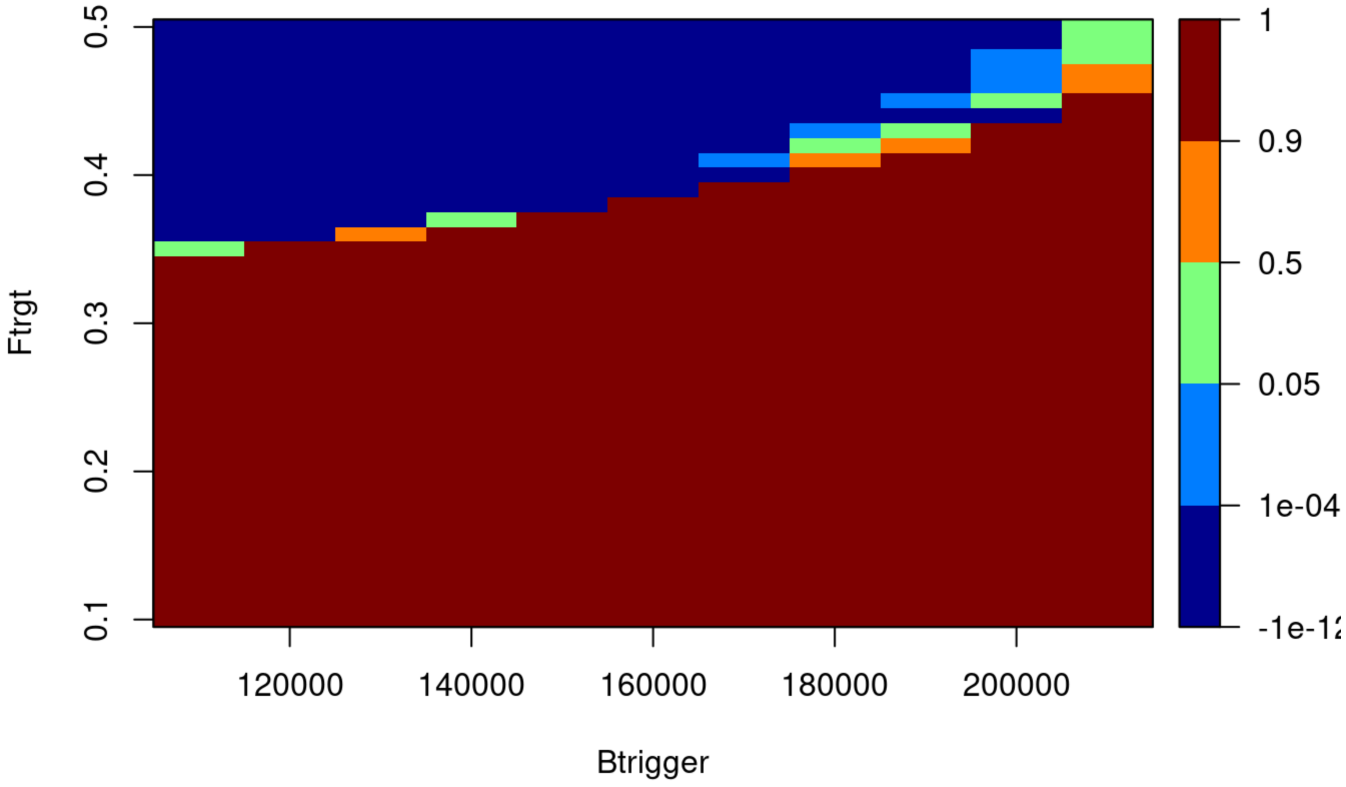


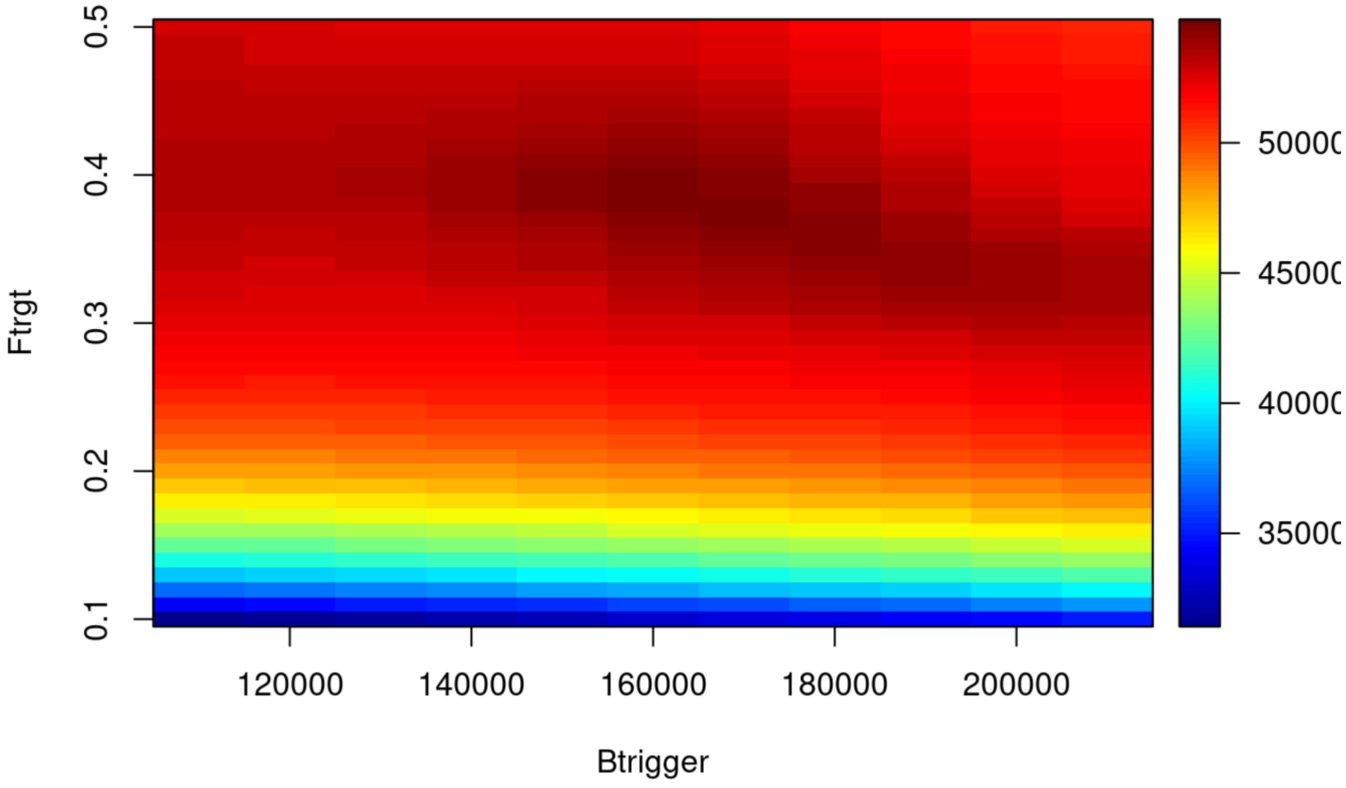
Ftrgt
0.1
0.2
0.3
0.4
0.5
120000
140000
160000
180000
200000
Btrigger
1
0.9
0.5
0.05
1e-04
-1e-12

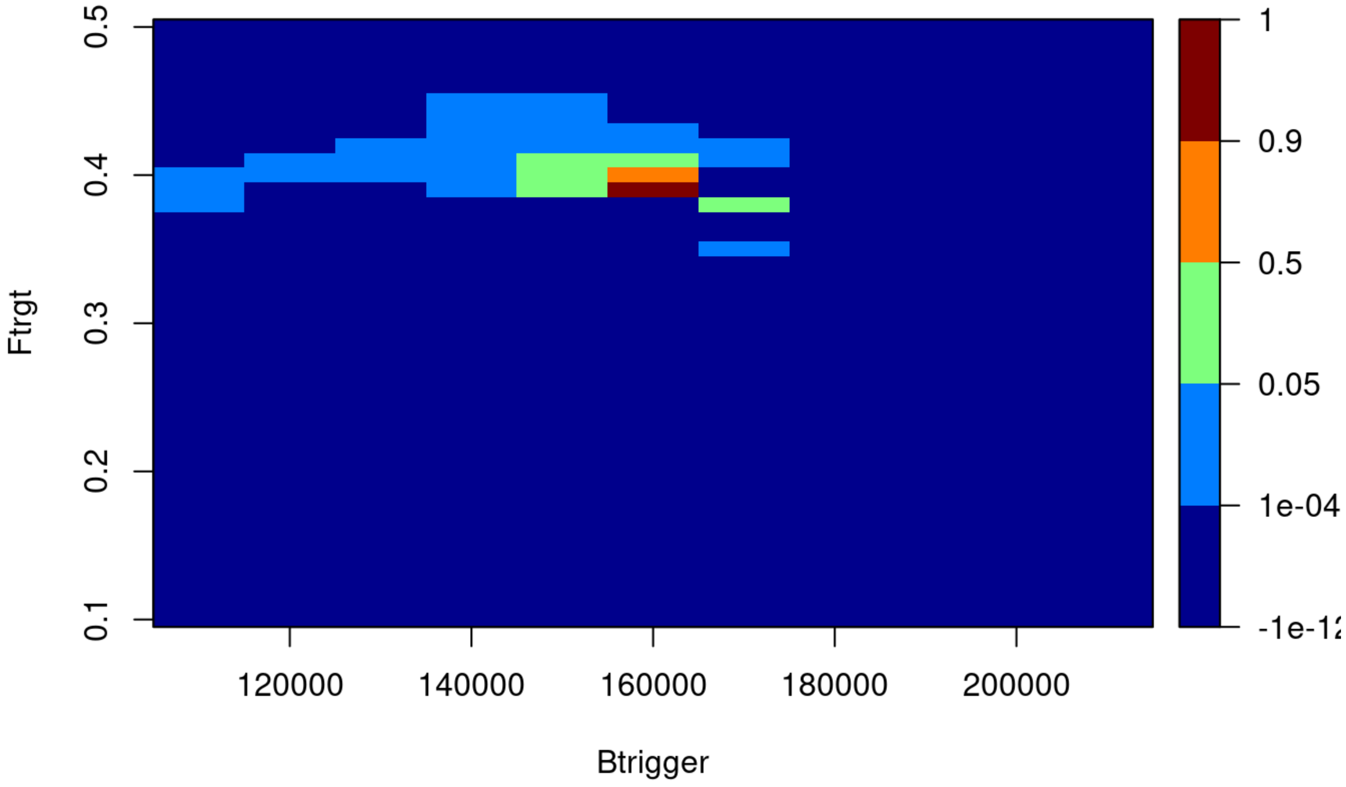


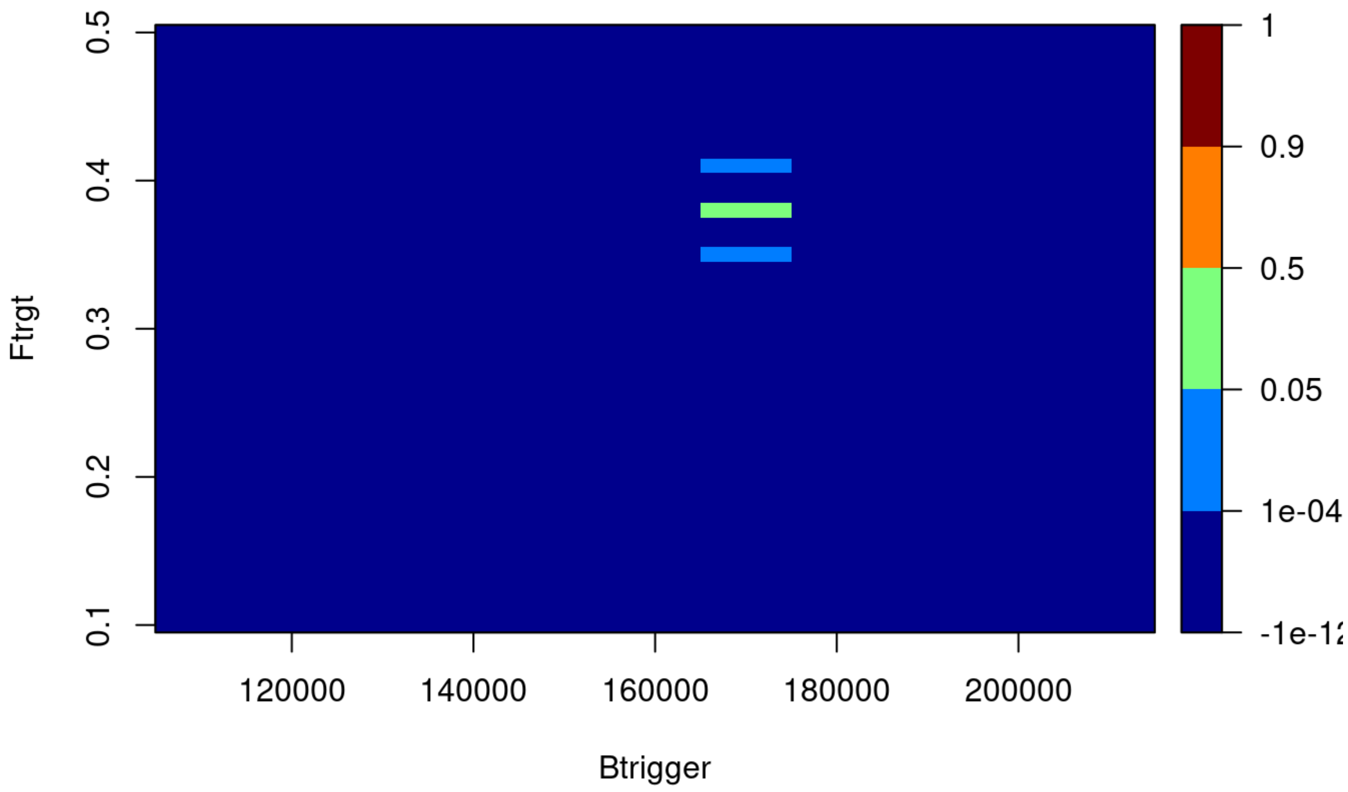
```
Plausible points remaining: 13
Best catch so far: 54596.5

 Round 6
Unevaluated candidates with EI > 0: 12
```


Ftrgt
0.1
0.2
0.3
0.4
0.5
120000
140000
160000
180000
200000
Btrigger
0.4
0.2
0.1
0.05
0.025
0.01
0

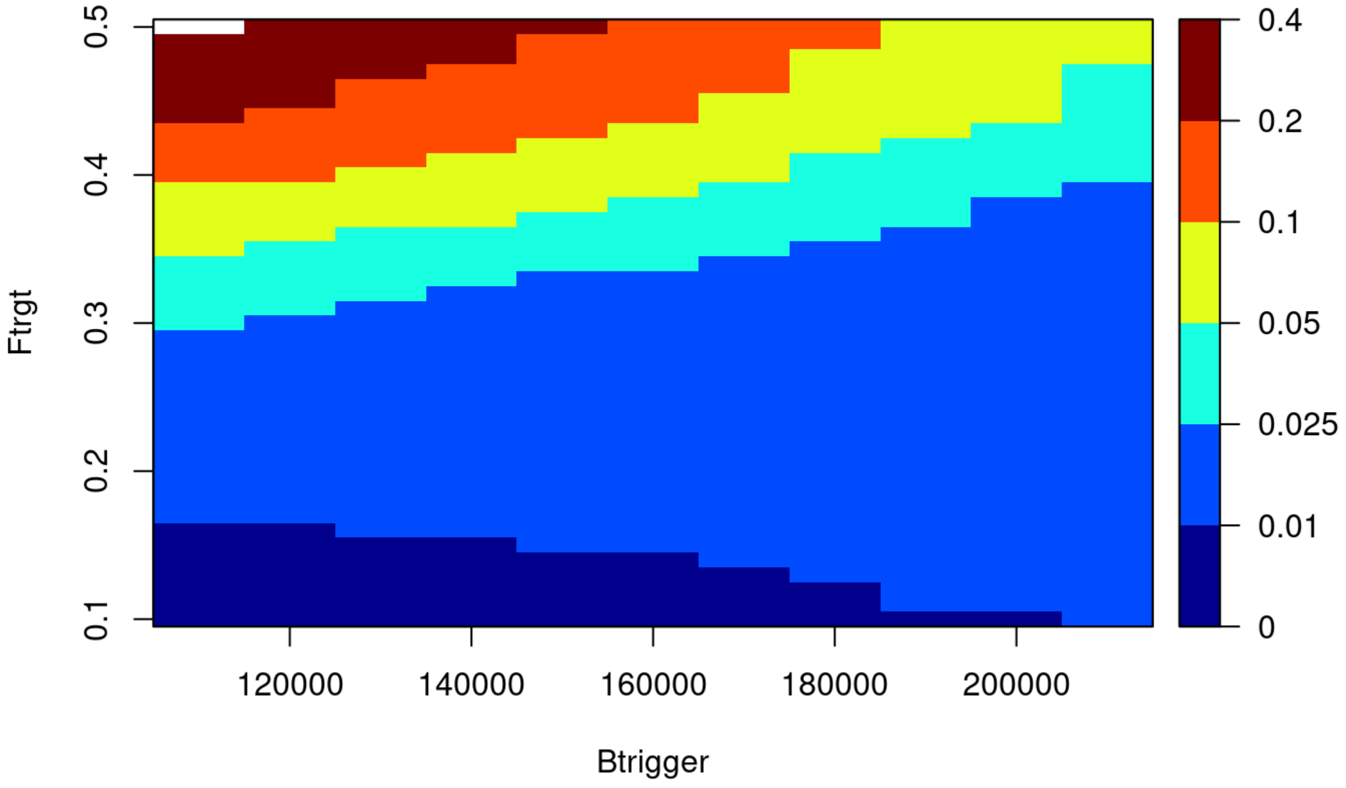

Ftrgt
0.1
0.2
0.3
0.4
0.5
120000
140000
160000
180000
200000
Btrigger
1
0.9
0.5
0.05
1e-04
-1e-12

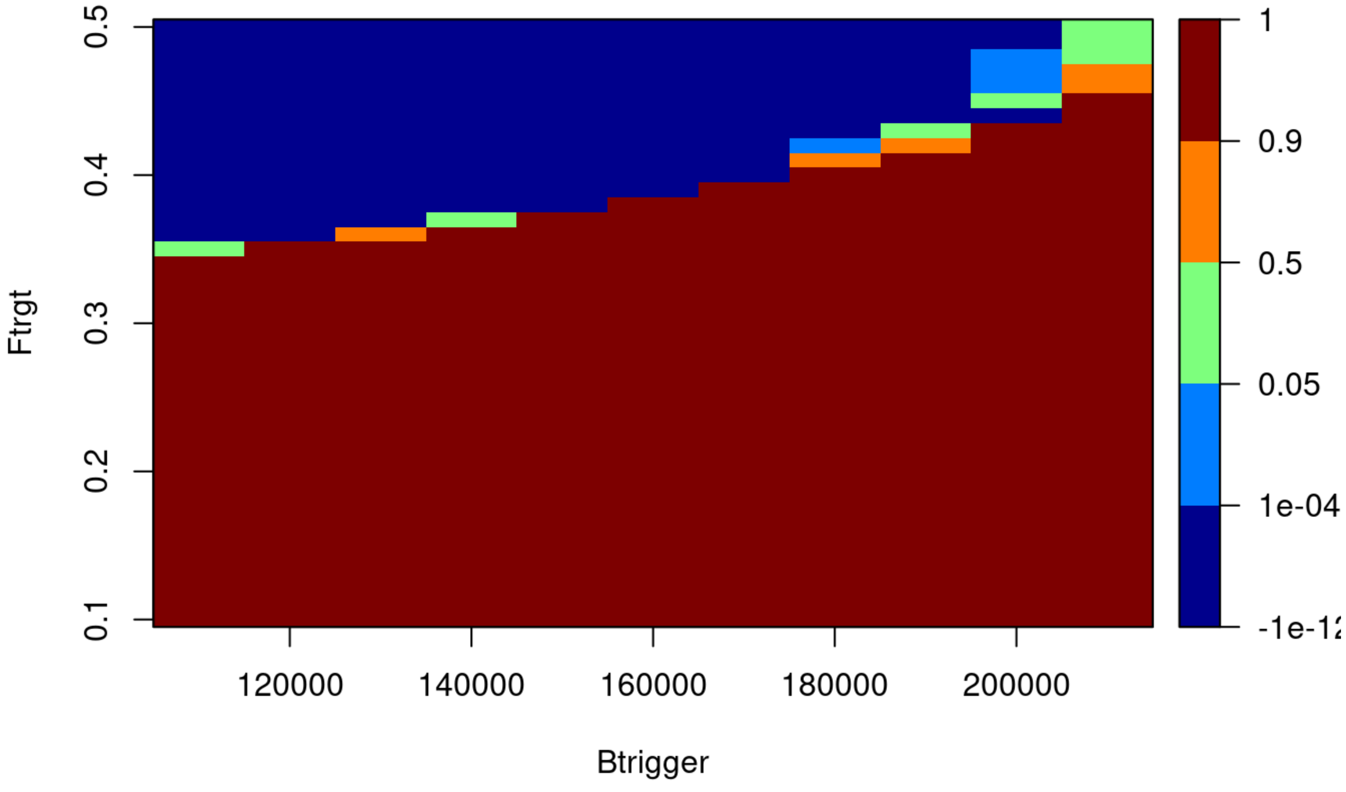

Ftrgt
0.1
0.2
0.3
0.4
0.5
120000
140000
160000
180000
200000
Btrigger
35000
40000
45000
50000

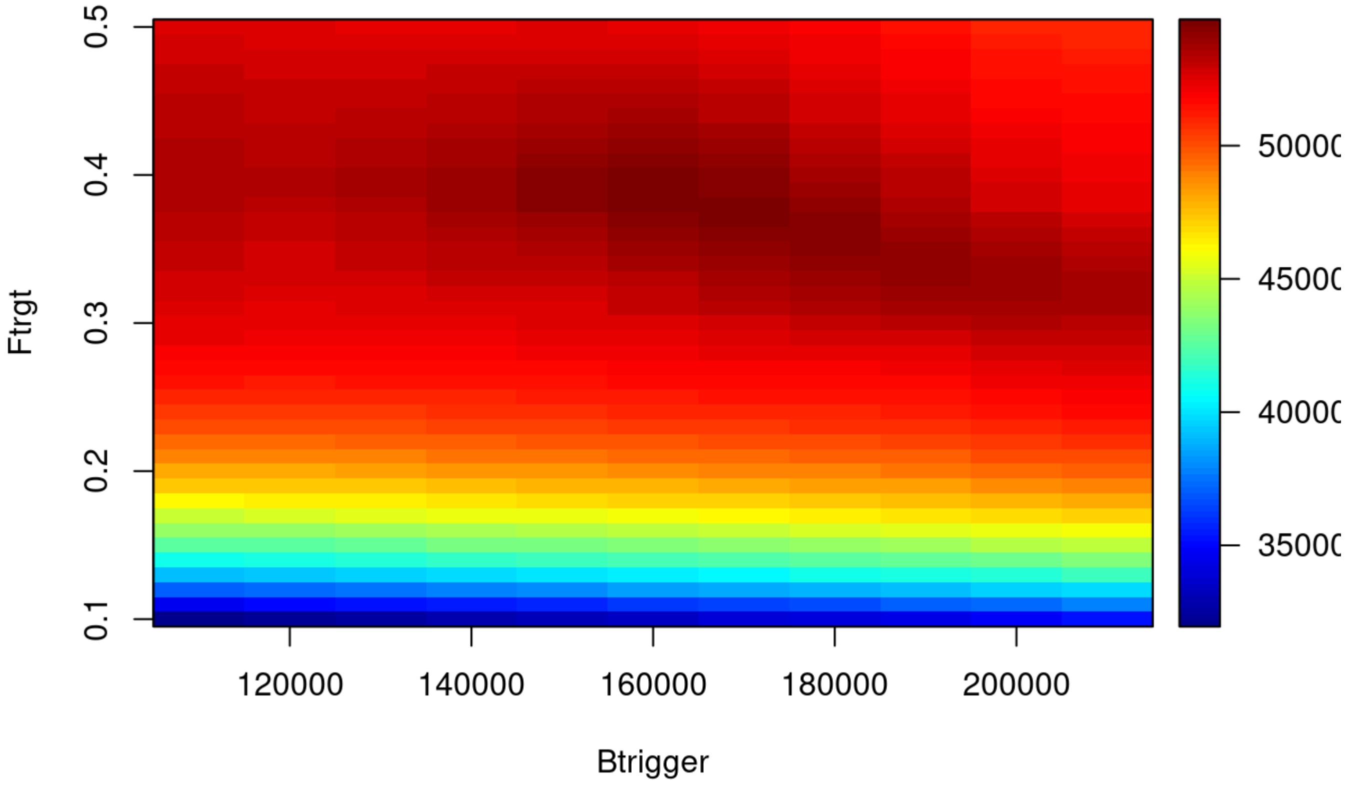

Ftrgt
0.1
0.2
0.3
0.4
0.5
120000
140000
160000
180000
200000
Btrigger
1
0.9
0.5
0.05
1e-04
-1e-12

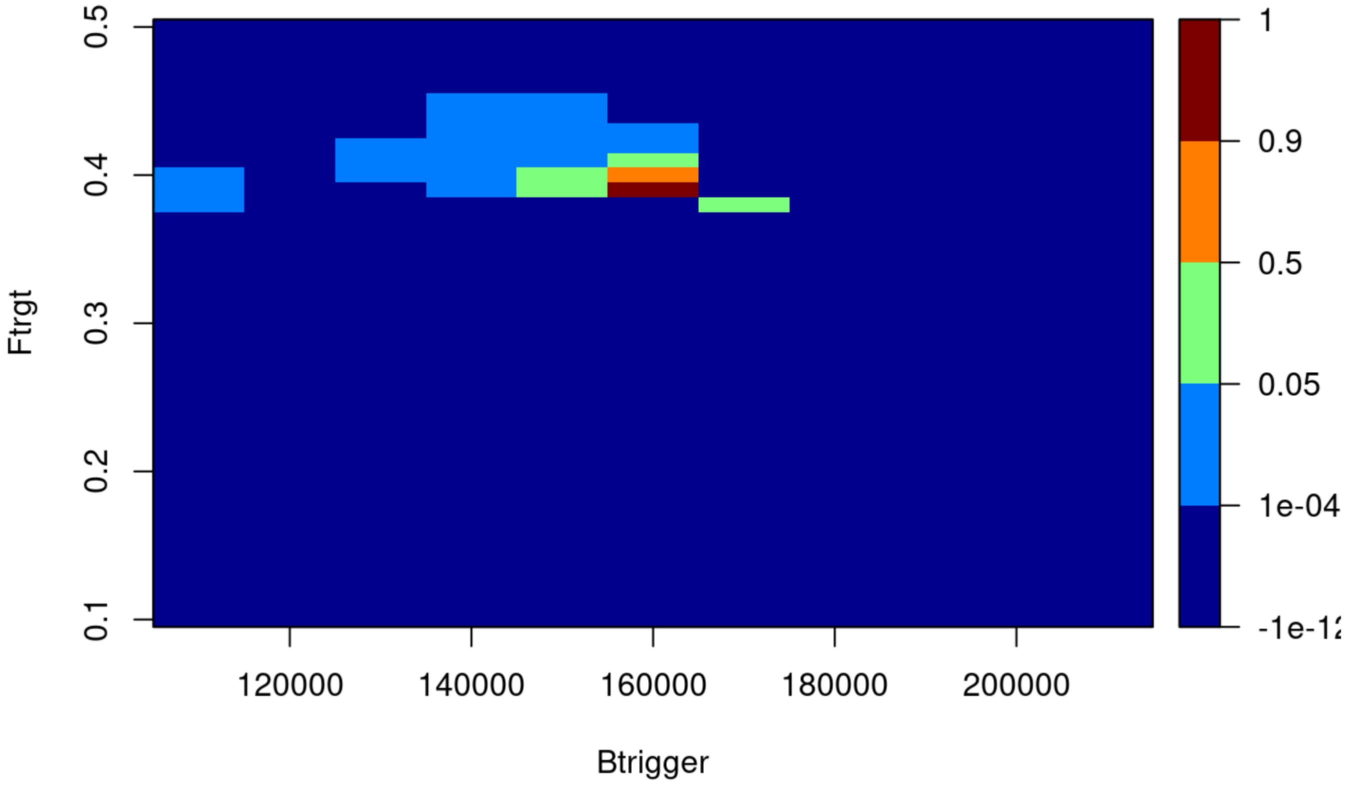


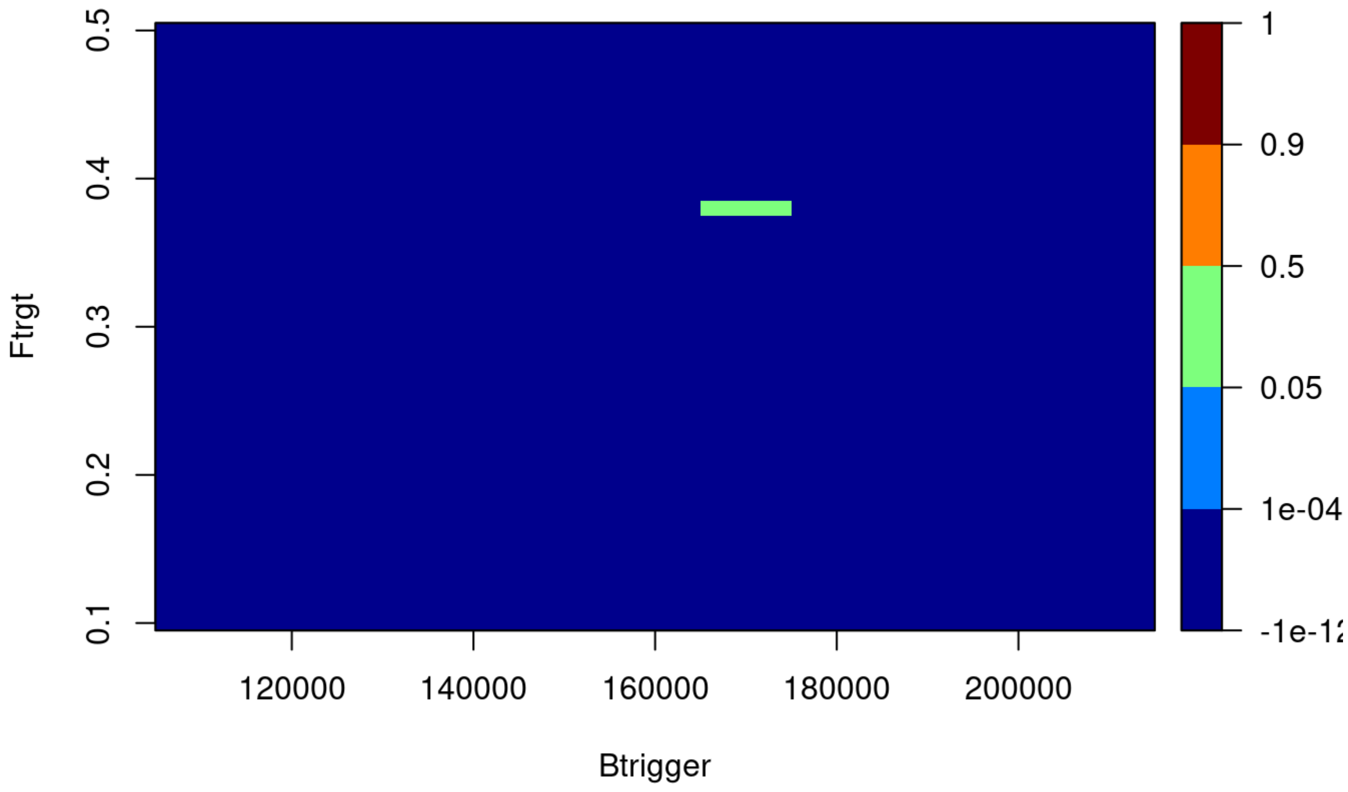
```
Plausible points remaining: 3
Best catch so far: 54596.5

 Round 7
Unevaluated candidates with EI > 0: 2
```


Ftrgt
0.1
0.2
0.3
0.4
0.5
120000
140000
160000
180000
200000
Btrigger
0
0.01
0.025
0.05
0.1
0.2
0.4


Ftrgt
0.1
0.2
0.3
0.4
0.5
120000
140000
160000
180000
200000
Btrigger
-1e-12
1e-04
0.05
0.5
0.9
1


0.5
0.4
0.3
0.2
0.1
Ftrgt
120000
140000
160000
180000
200000
Btrigger
50000
45000
40000
35000


0.5
0.4
0.3
0.2
0.1
Ftrgt
120000
140000
160000
180000
200000
Btrigger
1
0.9
0.5
0.05
1e-04
-1e-12



```
Plausible points remaining: 1
Best catch so far: 54596.5

 Round 8
No unevaluated candidates with positive EI. Stopping at round 8
```

```
#FINAL RESULTS
cat("\n OPTIMIZATION COMPLETE \n")
```

```

 OPTIMIZATION COMPLETE
```

```
cat("Completed", round_num, "rounds\n")
```

```
Completed 7 rounds
```

```
cat("Total evaluations:", nrow(runs), "\n")
```

```
Total evaluations: 50
```

```
# Get final answer
pot_points_final <- gridd[possible, ]
if (nrow(pot_points_final) > 0) {
  ans <- unrescale_Her(pot_points_final[1, ], dat1)
  cat("\nOptimal parameters:\n")
  ans
} else {
  cat("\nNo feasible points found.\n")
}
```

```

Optimal parameters:

    Ftarget Btrigger
275    0.38   170000
```

```
# Create final dataset for plotting
dat_round <- left_join(dat, all_rounds, by = c("Ftrgt", "Btrigger"))

# Color coding
col_round <- c("white", hcl.colors(max(all_rounds$Round), "viridis", rev = TRUE))
```