5.Gene_expression_analysis

Fay

2022-08-09

load libraries

```
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## v purrr   0.3.4
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(optimx)
```

Import data:

Here, we have the experimental / field data, inlcuding imputed data

```
lab <- read.csv("output_data/gene_expression/data_products/lab_imputed_gene_expression.csv")
field <- read.csv("output_data/gene_expression/data_products/field_imputed_gene_expression.csv")
```

Selecting genes

```
# vectors for selecting gene columns
Genes_lab <- c("IFNy", "CXCR3", "IL.6", "IL.10", "IL.13", "IL1RN", "CASP1",
               "CXCL9", "IDO1", "IRGM1", "MPO", "MUC2", "MUC5AC",
           "MYD88", "NCR1", "PRF1", "RETNLB", "SOCS1", "TICAM1", "TNF")

Genes_field   <- c("IFNy",  "CXCR3", "IL.6", #"GBP2", "IL.12", "IRG6",
                      "IL.10", "IL.13", "IL1RN",
                      "CXCR3", "CASP1", "CXCL9",
                      "IDO1", "IRGM1", "MPO", "MUC2", "MUC5AC", "MYD88",
                      "NCR1", "PRF1", "RETNLB", "SOCS1", "TICAM1", "TNF")
```

It is time to apply the package of Alice Balard et al. on our predictions!

Let's see if we indeed have differences across the hybrid index with our predicted weight loss.

Install the package

```
##
## * checking for file '/tmp/Rtmpy3a1fJ/remotes2e96174a7e12db/alicebalard-parasiteLoad-1b43216/DESCRIPTI
## * preparing 'parasiteLoad':
## * checking DESCRIPTION meta-information ... OK
## * checking for LF line-endings in source and make files and shell scripts
## * checking for empty or unneeded directories
## * building 'parasiteLoad_0.1.0.tar.gz'
```

Applying Alice's package on every gene

```
x <- field$IDO1



# Define function to be used to test, get the log lik and aic
tryDistrib <- function(x, distrib){
  # deals with fitdistr error:
  fit <- tryCatch(MASS::fitdistr(x, distrib), error=function(err) "fit failed")
  return(list(fit = fit,
              loglik = tryCatch(fit$loglik, error=function(err) "no loglik computed"),
              AIC = tryCatch(fit$aic, error=function(err) "no aic computed")))
}



findGoodDist <- function(x, distribs, distribs2){
  l =lapply(distribs, function(i) tryDistrib(x, i))
  names(l) <- distribs
  print(l)
  listDistr <- lapply(distribs2, function(i){
    if (i %in% "t"){
      fitdistrplus::fitdist(x, i, start = list(df =2))
    } else {
      fitdistrplus::fitdist(x,i)
    }}
  )
  par(mfrow=c(2,2))
  denscomp(listDistr, legendtext=distribs2)
  cdfcomp(listDistr, legendtext=distribs2)
  qqcomp(listDistr, legendtext=distribs2)
  ppcomp(listDistr, legendtext=distribs2)
  par(mfrow=c(1,1))
}
```

```
tryDistrib(x, "normal")
```

Functions for testing distributions

```
## $fit
##       mean          sd
##   15.1432029    4.3044978
##  ( 0.2278182) ( 0.1610918)
##
## $loglik
## [1] -1027.66
##
## $AIC
## NULL
```

```
tryDistrib(x, "binomial")
```

```
## $fit
## [1] "fit failed"
##
## $loglik
## [1] "no loglik computed"
##
## $AIC
## [1] "no aic computed"
```

```
tryDistrib(x, "student")
```

```
## $fit
## [1] "fit failed"
##
## $loglik
## [1] "no loglik computed"
##
## $AIC
## [1] "no aic computed"
```

```
tryDistrib(x, "weibull")
```

```
## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
```

```
## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
```

```
## $fit
##      shape        scale
##    3.6718724   16.7658776
##  ( 0.1428414) ( 0.2566000)
##
## $loglik
## [1] -1032.752
##
## $AIC
## NULL
```

```
tryDistrib(x, "weibullshifted")
```

```
## $fit
## [1] "fit failed"
##
## $loglik
```

```
## [1] "no loglik computed"
##
## $AIC
## [1] "no aic computed"
```

```
field$Sex <- as.factor(field$Sex)

speparam <- parasiteLoad::getParamBounds("weibull", data = field, response = "IFNy")

IFNy <- parasiteLoad::analyse(data = field,
                          response = "IFNy",
                          model = "weibull",
                          group = "Sex")
```

```
## [1] "Analysing data for response: IFNy"
## [1] "Fit for the response: IFNy"
## [1] "Fitting for all"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Fitting for groupA : F"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Fitting for groupB : M"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Testing H0 no alpha vs alpha"
##    dLL dDF     pvalue
## 1 2.22   1 0.03531073
## [1] "Testing H1 no alpha vs alpha"
##   dLL dDF     pvalue
## 1 3.2   1 0.01145293
## [1] "Testing H2 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.01   1 0.8704737
## [1] "Testing H2 groupB no alpha vs alpha"
##    dLL dDF       pvalue
## 1 7.87   1 7.270491e-05
```

```
## [1] "Testing H3 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.29   1 0.4482965
## [1] "Testing H3 groupB no alpha vs alpha"
##    dLL dDF      pvalue
## 1 7.69   1 8.80334e-05
## [1] "Testing H1 vs H0"
##    dLL dDF      pvalue
## 1 4.26   1 0.003524909
## [1] "Testing H2 vs H0"
##     dLL dDF       pvalue
## 1 11.04   3 6.250965e-05
## [1] "Testing H3 vs H1"
##     dLL dDF       pvalue
## 1 10.44   4 0.0003336357
## [1] "Testing H3 vs H2"
##    dLL dDF     pvalue
## 1 3.66   2 0.02585681
```

```
##All
print(IFNy)
```

```
## $H0
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1      alpha    myshape
## 33.1431666  0.6610428  1.2340092
##
## Log-likelihood: -1521.92
## Best method: bobyqa
##
## $H1
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1         L2      alpha    myshape
## 27.2414906 38.9834520  0.7464894  1.2538977
```

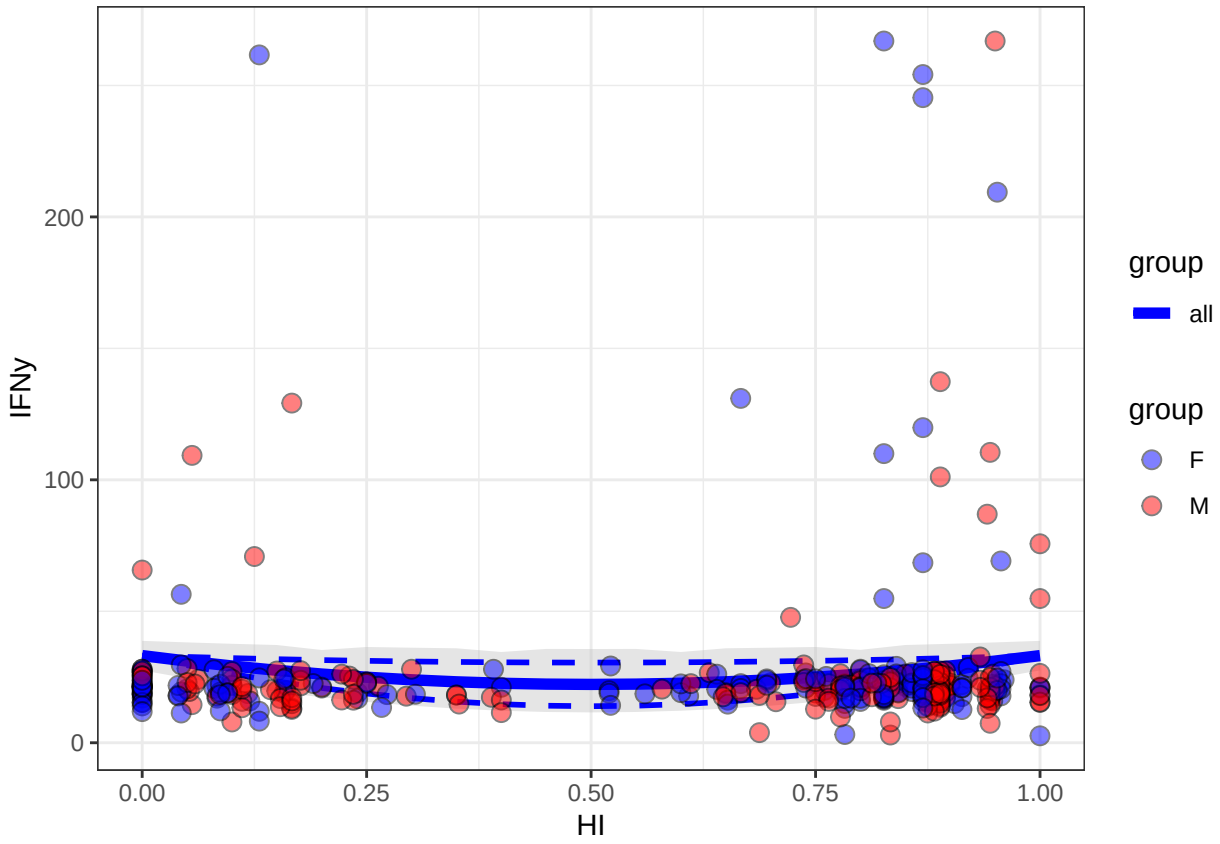
```
##
## Log-likelihood: -1517.66
## Best method: bobyqa
##
## $H2
## $H2$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1      alpha    myshape
## 29.3185104 -0.1066676  1.1502540
##
## Log-likelihood: -776.52
## Best method: bobyqa
##
## $H2$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##        L1     alpha   myshape
## 35.755998  1.170877  1.437375
##
## Log-likelihood: -734.35
## Best method: bobyqa
##
##
## $H3
## $H3$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
```

```
##
## Coefficients:
##         L1         L2      alpha    myshape
## 24.7435042 39.9424267  0.4267503  1.1727393
##
## Log-likelihood: -773.59
## Best method: bobyqa
##
## $H3$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##        L1        L2     alpha   myshape
## 31.911018 38.117710  1.151850  1.448096
##
## Log-likelihood: -733.63
## Best method: bobyqa
```

```
bananaPlot(mod = IFNy$H0,
             data = field,
             response = "IFNy",
             group = "Sex") +
  scale_fill_manual(values = c("blue", "red")) +
  scale_color_manual(values = c("blue", "red")) +
  theme_bw()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```




```
field$Sex <- as.factor(field$Sex)

speparam <- parasiteLoad::getParamBounds("weibull", data = field, response = "CXCR3")

IFNy <- parasiteLoad::analyse(data = field,
                          response = "CXCR3",
                          model = "weibull",
                          group = "Sex")
```

```
## [1] "Analysing data for response: CXCR3"
## [1] "Fit for the response: CXCR3"
## [1] "Fitting for all"
## [1] "Fitting model basic without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model basic with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
```

```
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting for groupA : F"
## [1] "Fitting model basic without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model basic with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting for groupB : M"
## [1] "Fitting model basic without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model basic with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Testing H0 no alpha vs alpha"
##   dLL dDF   pvalue
## 1 0.3   1 0.435069
## [1] "Testing H1 no alpha vs alpha"
##    dLL dDF   pvalue
## 1 0.41   1 0.362571
## [1] "Testing H2 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.05   1 0.7442424
## [1] "Testing H2 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.28   1 0.4512167
## [1] "Testing H3 groupA no alpha vs alpha"
##   dLL dDF    pvalue
## 1 0.2   1 0.5310769
## [1] "Testing H3 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.29   1 0.4490016
## [1] "Testing H1 vs H0"
##    dLL dDF   pvalue
## 1 0.28   1 0.451221
## [1] "Testing H2 vs H0"
##    dLL dDF   pvalue
## 1 0.37   3 0.863792
## [1] "Testing H3 vs H1"
##    dLL dDF    pvalue
## 1 0.59   4 0.8818897
## [1] "Testing H3 vs H2"
##   dLL dDF    pvalue
## 1 0.5   2 0.6051338
```

```
##All
print(IFNy)
```

```
## $H0
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 21.06964395  0.05978311  5.00000000
##
## Log-likelihood: -941.43
```

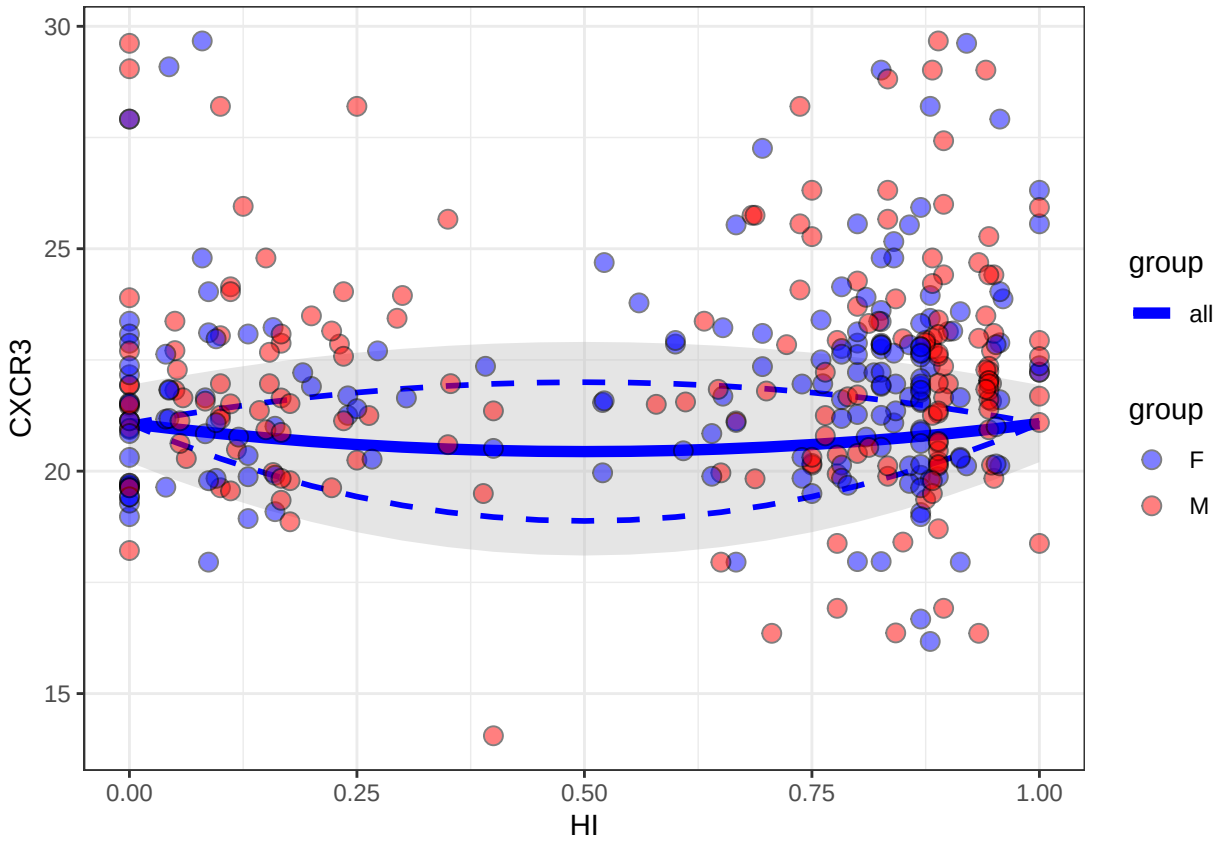
```
## Best method: bobyqa
##
## $H1
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 20.84145207 21.31893894  0.07048561  5.00000000
##
## Log-likelihood: -941.15
## Best method: bobyqa
##
## $H2
## $H2$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 20.76691119  0.03505679  5.00000000
##
## Log-likelihood: -465.22
## Best method: bobyqa
##
## $H2$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 21.35517115  0.08226393  5.00000000
```

```
##
## Log-likelihood: -475.84
## Best method: bobyqa
##
##
## $H3
## $H3$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1         L2      alpha    myshape
## 20.3935446 21.3147118  0.0697062  5.0000000
##
## Log-likelihood: -464.72
## Best method: bobyqa
##
## $H3$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 21.31744395 21.38636087  0.08281592  5.00000000
##
## Log-likelihood: -475.84
## Best method: bobyqa
```

```
bananaPlot(mod = IFNy$H0,
             data = field,
             response = "CXCR3",
             group = "Sex") +
  scale_fill_manual(values = c("blue", "red")) +
  scale_color_manual(values = c("blue", "red")) +
  theme_bw()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```




```
field$Sex <- as.factor(field$Sex)

speparam <- parasiteLoad::getParamBounds("weibull", data = field, response = "IL.6")

IFNy <- parasiteLoad::analyse(data = field,
                          response = "IL.6",
                          model = "weibull",
                          group = "Sex")
```

```
## [1] "Analysing data for response: IL.6"
## [1] "Fit for the response: IL.6"
## [1] "Fitting for all"
## [1] "Fitting model basic without alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting for groupA : F"
## [1] "Fitting model basic without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model basic with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting for groupB : M"
## [1] "Fitting model basic without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model basic with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Testing H0 no alpha vs alpha"
##    dLL dDF   pvalue
## 1 0.04   1 0.772892
## [1] "Testing H1 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.03   1 0.8182315
## [1] "Testing H2 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.02   1 0.8312212
## [1] "Testing H2 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.02   1 0.8435907
## [1] "Testing H3 groupA no alpha vs alpha"
##    dLL dDF   pvalue
## 1 0.01   1 0.887319
## [1] "Testing H3 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.02   1 0.8363586
## [1] "Testing H1 vs H0"
##    dLL dDF    pvalue
## 1 0.09   1 0.6757541
## [1] "Testing H2 vs H0"
##    dLL dDF    pvalue
## 1 0.02   3 0.9982955
## [1] "Testing H3 vs H1"
##    dLL dDF    pvalue
## 1 0.02   4 0.9997268
## [1] "Testing H3 vs H2"
##    dLL dDF    pvalue
## 1 0.09   2 0.9105931
```

```
##All
print(IFNy)
```

```
## $H0
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 22.62401299  0.02221795  5.00000000
```

```
##
## Log-likelihood: -967.18
## Best method: bobyqa
##
## $H1
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 22.76423147 22.48362866  0.01791773  5.00000000
##
## Log-likelihood: -967.1
## Best method: bobyqa
##
## $H2
## $H2$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1      alpha    myshape
## 22.6719470  0.0228637  5.0000000
##
## Log-likelihood: -484.34
## Best method: bobyqa
##
## $H2$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
```

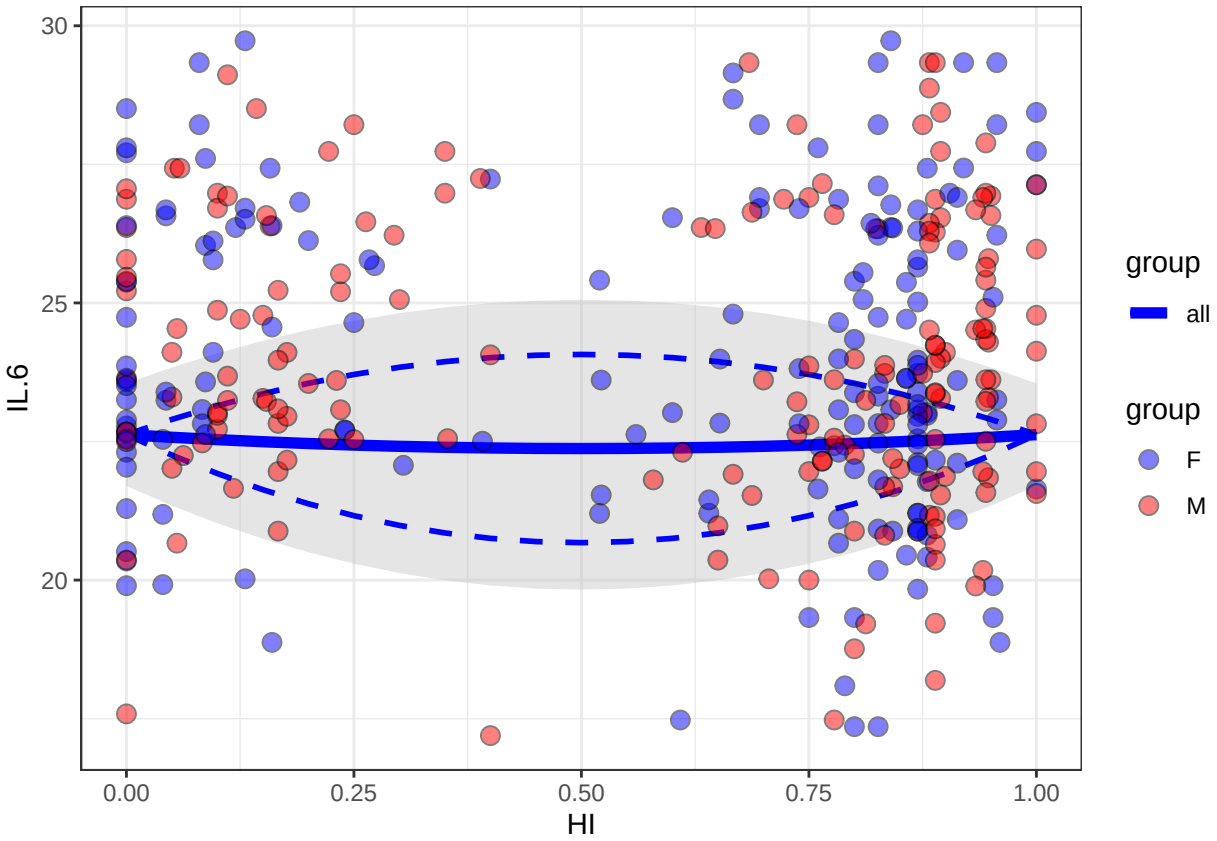
```
##          L1       alpha     myshape
## 22.57729903  0.02181845  5.00000000
##
## Log-likelihood: -482.83
## Best method: bobyqa
##
##
## $H3
## $H3$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 22.75917669 22.54465135  0.01591037  5.00000000
##
## Log-likelihood: -484.31
## Best method: bobyqa
##
## $H3$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 22.78737411 22.43638316  0.02292296  5.00000000
##
## Log-likelihood: -482.76
## Best method: bobyqa
```

```
bananaPlot(mod = IFNy$H0,
             data = field,
             response = "IL.6",
             group = "Sex") +
  scale_fill_manual(values = c("blue", "red")) +
  scale_color_manual(values = c("blue", "red")) +
  theme_bw()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
```

```
## will replace the existing scale.
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```




```
field$Sex <- as.factor(field$Sex)

speparam <- parasiteLoad::getParamBounds("weibull", data = field, response = "IL.10")

IFNy <- parasiteLoad::analyse(data = field,
                          response = "IL.10",
                          model = "weibull",
                          group = "Sex")
```

```
## [1] "Analysing data for response: IL.10"
## [1] "Fit for the response: IL.10"
## [1] "Fitting for all"
## [1] "Fitting model basic without alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting for groupA : F"
## [1] "Fitting model basic without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model basic with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting for groupB : M"
## [1] "Fitting model basic without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model basic with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
```

```
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Testing H0 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 1.13   1 0.1319113
## [1] "Testing H1 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.84   1 0.1949079
## [1] "Testing H2 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.99   1 0.1598326
## [1] "Testing H2 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.25   1 0.4813229
## [1] "Testing H3 groupA no alpha vs alpha"
##   dLL dDF    pvalue
## 1 0.7   1 0.2383206
## [1] "Testing H3 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.23   1 0.4981587
## [1] "Testing H1 vs H0"
##    dLL dDF    pvalue
## 1 0.38   1 0.3858744
## [1] "Testing H2 vs H0"
##    dLL dDF    pvalue
## 1 0.18   3 0.9493647
## [1] "Testing H3 vs H1"
##    dLL dDF    pvalue
## 1 0.17   4 0.9874975
## [1] "Testing H3 vs H2"
##    dLL dDF    pvalue
## 1 0.37   2 0.6938127
```

```
##All
print(IFNy)
```

```
## $H0
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
```

```
## Coefficients:
##         L1      alpha    myshape
## 23.9041161  0.1141012  5.0000000
##
## Log-likelihood: -983.29
## Best method: bobyqa
##
## $H1
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##        L1        L2     alpha   myshape
## 24.188503 23.568945  0.101235  5.000000
##
## Log-likelihood: -982.92
## Best method: bobyqa
##
## $H2
## $H2$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1      alpha    myshape
## 23.9618284  0.1430325  5.0000000
##
## Log-likelihood: -488.56
## Best method: bobyqa
##
## $H2$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
```

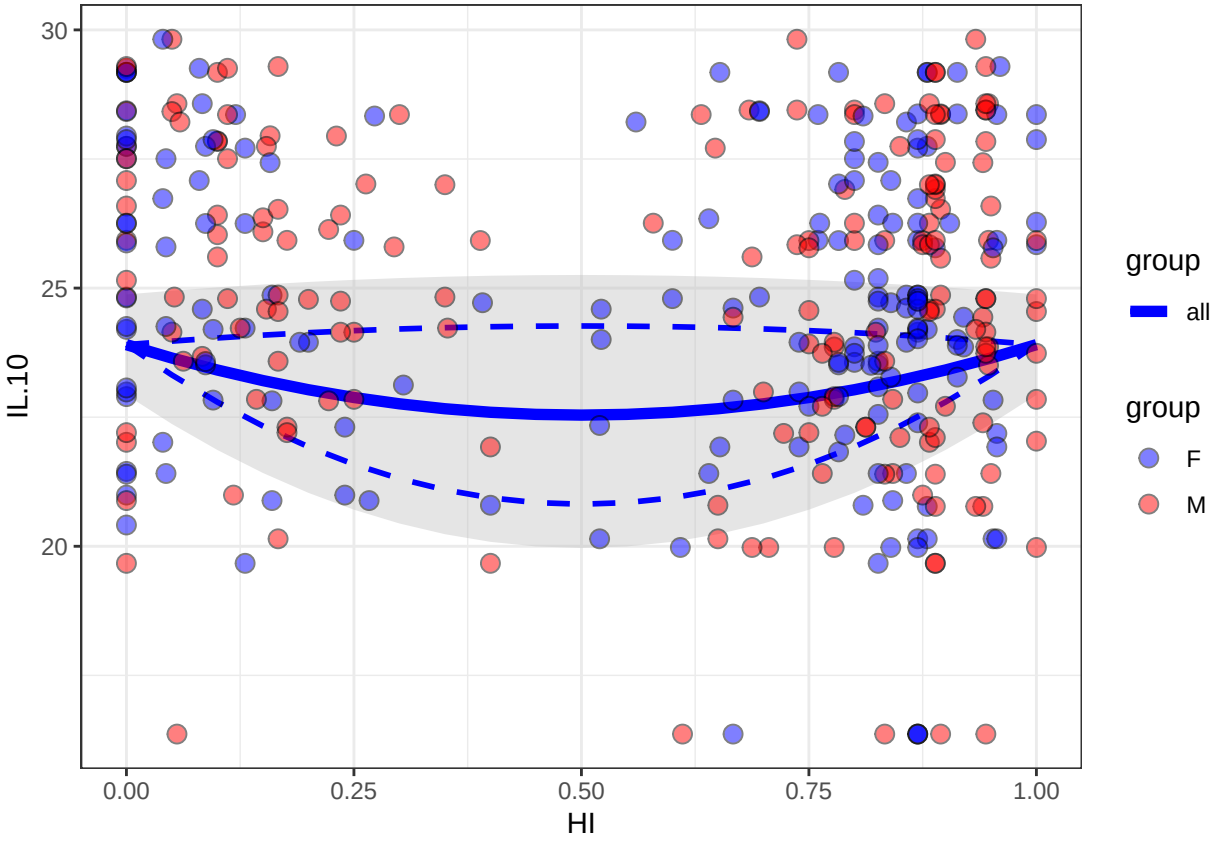
```
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 23.81497438  0.07976845  5.00000000
##
## Log-likelihood: -494.55
## Best method: bobyqa
##
##
## $H3
## $H3$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1         L2      alpha    myshape
## 24.0988699 23.7295619  0.1299026  5.0000000
##
## Log-likelihood: -488.5
## Best method: bobyqa
##
## $H3$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 24.24833902 23.47582522  0.07739981  5.00000000
##
## Log-likelihood: -494.25
## Best method: bobyqa
```

```
bananaPlot(mod = IFNy$H0,
             data = field,
             response = "IL.10",
             group = "Sex") +
  scale_fill_manual(values = c("blue", "red")) +
```

```
  scale_color_manual(values = c("blue", "red")) +
  theme_bw()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```




```
field$Sex <- as.factor(field$Sex)

speparam <- parasiteLoad::getParamBounds("weibull", data = field, response = "IL.13")

IFNy <- parasiteLoad::analyse(data = field,
                         response = "IL.13",
                         model = "weibull",
                         group = "Sex")
```

```
## [1] "Analysing data for response: IL.13"
## [1] "Fit for the response: IL.13"
## [1] "Fitting for all"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
```

```
## [1] "Did converge"
## [1] "Fitting for groupA : F"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Fitting for groupB : M"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Testing H0 no alpha vs alpha"
##    dLL dDF   pvalue
## 1 0.61   1 0.269707
## [1] "Testing H1 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.35   1 0.4040282
## [1] "Testing H2 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.42   1 0.3593852
## [1] "Testing H2 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.21   1 0.5161163
## [1] "Testing H3 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.38   1 0.3836652
## [1] "Testing H3 groupB no alpha vs alpha"
##    dLL dDF   pvalue
## 1 0.13   1 0.605722
## [1] "Testing H1 vs H0"
##    dLL dDF    pvalue
## 1 1.13   1 0.1327128
## [1] "Testing H2 vs H0"
##    dLL dDF    pvalue
## 1 1.44   3 0.4095516
## [1] "Testing H3 vs H1"
##    dLL dDF   pvalue
## 1 2.53   4 0.281628
## [1] "Testing H3 vs H2"
##    dLL dDF    pvalue
## 1 2.22   2 0.1091379
```

```
##All
print(IFNy)
```

```
## $H0
##
```

```
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1      alpha    myshape
## 16.9430224  0.1251227  3.2243289
##
## Log-likelihood: -1118.73
## Best method: bobyqa
##
## $H1
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 17.50789245 16.33902781  0.09726427  3.23125613
##
## Log-likelihood: -1117.6
## Best method: bobyqa
##
## $H2
## $H2$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1      alpha    myshape
## 17.5328781  0.1425032  3.3269127
##
## Log-likelihood: -558.67
## Best method: bobyqa
##
```

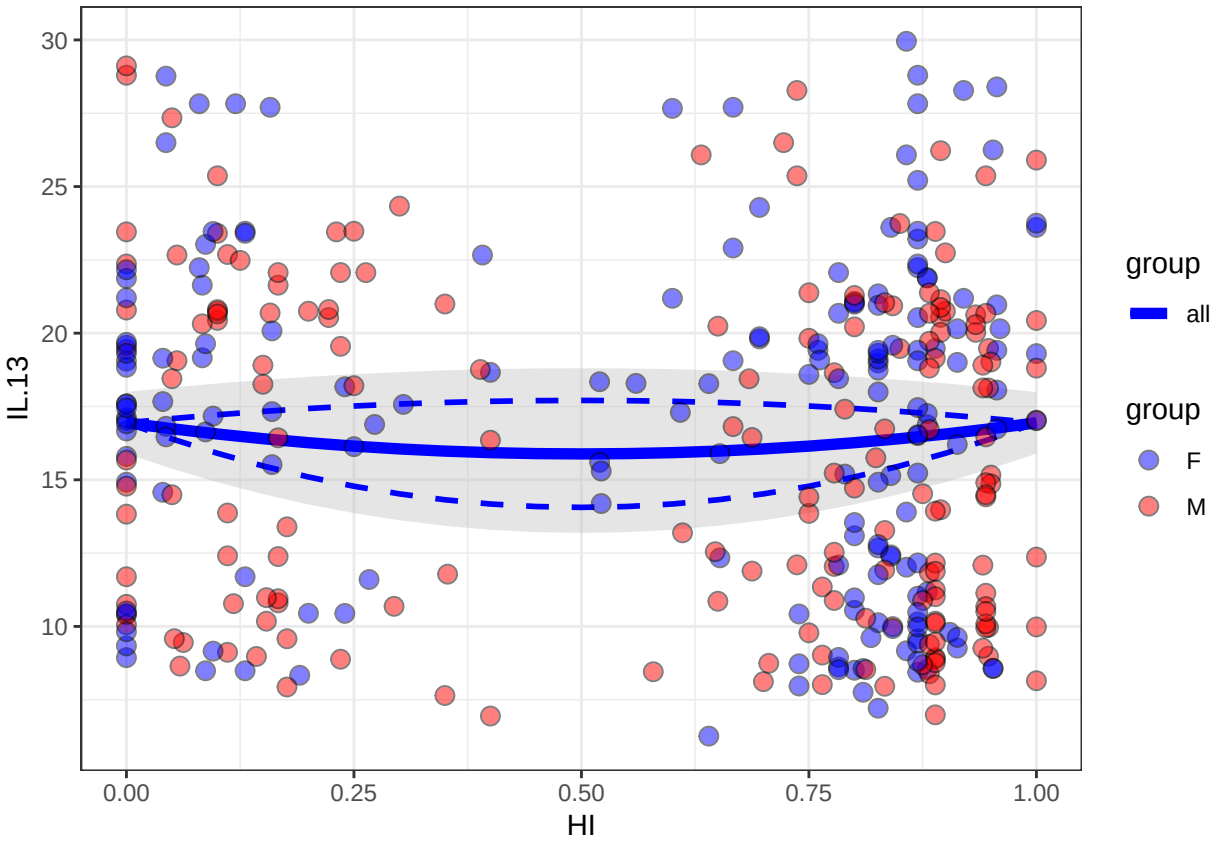
```
## $H2$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##        L1     alpha   myshape
## 16.356294  0.106893  3.148128
##
## Log-likelihood: -558.62
## Best method: bobyqa
##
##
## $H3
## $H3$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1         L2      alpha    myshape
## 17.5602669 17.4965735  0.1403354  3.3265980
##
## Log-likelihood: -558.67
## Best method: L-BFGS-B
##
## $H3$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 17.59224136 15.34317142  0.08582084  3.19159541
```

```
##
## Log-likelihood: -556.4
## Best method: bobyqa
```

```
bananaPlot(mod = IFNy$H0,
             data = field,
             response = "IL.13",
             group = "Sex") +
  scale_fill_manual(values = c("blue", "red")) +
  scale_color_manual(values = c("blue", "red")) +
  theme_bw()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```




```
field$Sex <- as.factor(field$Sex)

speparam <- parasiteLoad::getParamBounds("weibull", data = field, response = "IL1RN")

IFNy <- parasiteLoad::analyse(data = field,
                          response = "IL1RN",
                          model = "weibull",
                          group = "Sex")
```

```
## [1] "Analysing data for response: IL1RN"
## [1] "Fit for the response: IL1RN"
```

```
## [1] "Fitting for all"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Fitting for groupA : F"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Fitting for groupB : M"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Testing H0 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.29   1 0.4460508
## [1] "Testing H1 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.13   1 0.6047405
## [1] "Testing H2 groupA no alpha vs alpha"
##   dLL dDF    pvalue
## 1 0.1   1 0.6613392
## [1] "Testing H2 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.21   1 0.5198016
## [1] "Testing H3 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.01   1 0.9080646
## [1] "Testing H3 groupB no alpha vs alpha"
##    dLL dDF   pvalue
## 1 0.17   1 0.561718
## [1] "Testing H1 vs H0"
##    dLL dDF    pvalue
## 1 0.65   1 0.2545631
## [1] "Testing H2 vs H0"
##   dLL dDF    pvalue
## 1 1.8   3 0.3068796
## [1] "Testing H3 vs H1"
##    dLL dDF    pvalue
## 1 1.88   4 0.4407595
```

```
## [1] "Testing H3 vs H2"
##    dLL dDF    pvalue
## 1 0.72   2 0.4867802
```

```
##All
print(IFNy)
```

```
## $H0
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 14.20608660 -0.08178146  3.51503257
##
## Log-likelihood: -1033.51
## Best method: L-BFGS-B
##
## $H1
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 13.87671692 14.57381902 -0.05598649  3.51469421
##
## Log-likelihood: -1032.86
## Best method: bobyqa
##
## $H2
## $H2$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
```

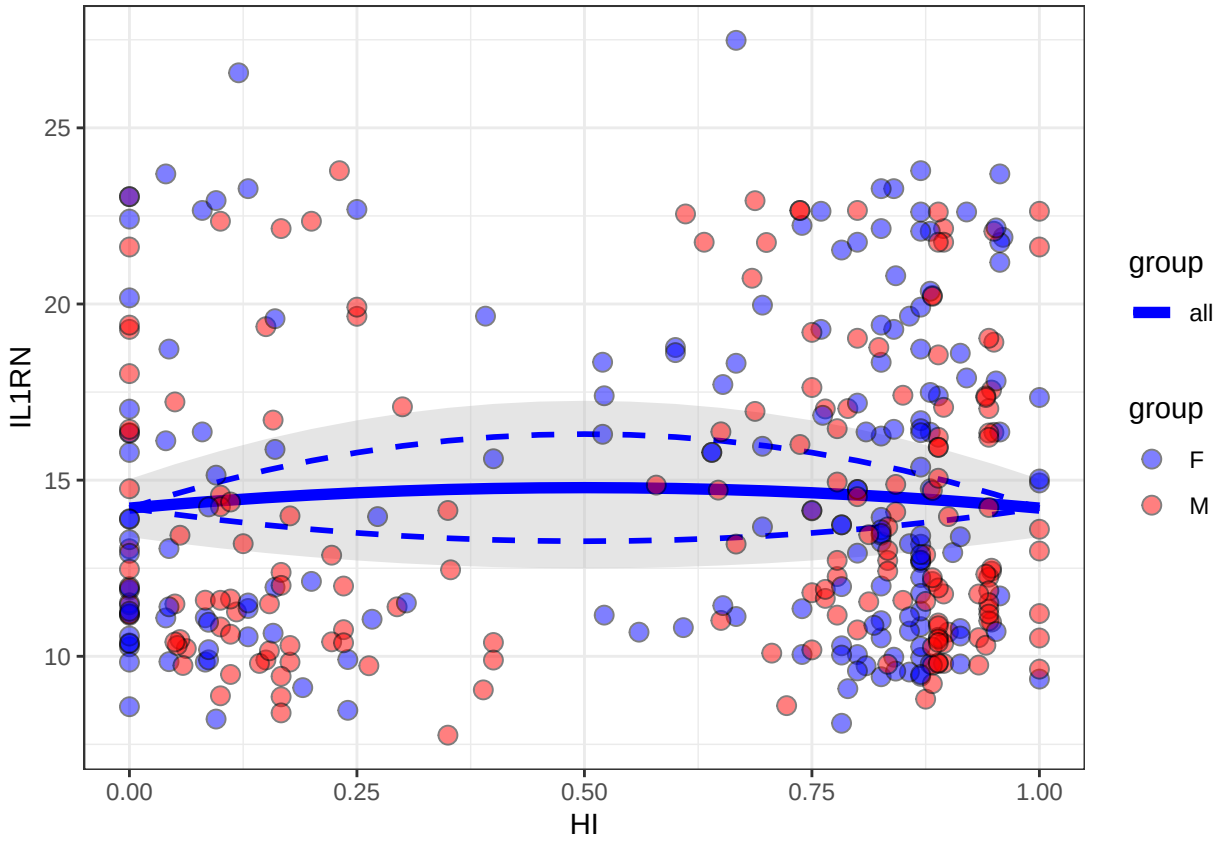
```
##
## Coefficients:
##          L1       alpha     myshape
## 14.59904776 -0.06987697  3.47926954
##
## Log-likelihood: -522.07
## Best method: bobyqa
##
## $H2$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 13.82579783 -0.09134731  3.59320483
##
## Log-likelihood: -509.64
## Best method: bobyqa
##
##
## $H3
## $H3$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 14.19802997 15.16970637 -0.01880956  3.47961721
##
## Log-likelihood: -521.52
## Best method: bobyqa
##
## $H3$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
```

```
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 13.57517204 14.04484807 -0.08226647  3.59270998
##
## Log-likelihood: -509.47
## Best method: bobyqa
```

```
bananaPlot(mod = IFNy$H0,
             data = field,
             response = "IL1RN",
             group = "Sex") +
  scale_fill_manual(values = c("blue", "red")) +
  scale_color_manual(values = c("blue", "red")) +
  theme_bw()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```




```
field$Sex <- as.factor(field$Sex)

speparam <- parasiteLoad::getParamBounds("weibull", data = field, response = "CXCR3")
```

```
IFNy <- parasiteLoad::analyse(data = field,
                          response = "CXCR3",
                          model = "weibull",
                          group = "Sex")
```

```
## [1] "Analysing data for response: CXCR3"
## [1] "Fit for the response: CXCR3"
## [1] "Fitting for all"
## [1] "Fitting model basic without alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting for groupA : F"
## [1] "Fitting model basic without alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting for groupB : M"
## [1] "Fitting model basic without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model basic with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Testing H0 no alpha vs alpha"
##   dLL dDF   pvalue
## 1 0.3   1 0.435069
## [1] "Testing H1 no alpha vs alpha"
##    dLL dDF   pvalue
## 1 0.41   1 0.362571
## [1] "Testing H2 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.05   1 0.7442424
## [1] "Testing H2 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.28   1 0.4512167
## [1] "Testing H3 groupA no alpha vs alpha"
##   dLL dDF    pvalue
## 1 0.2   1 0.5310769
## [1] "Testing H3 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.29   1 0.4490016
## [1] "Testing H1 vs H0"
##    dLL dDF   pvalue
## 1 0.28   1 0.451221
## [1] "Testing H2 vs H0"
##    dLL dDF   pvalue
## 1 0.37   3 0.863792
```

```
## [1] "Testing H3 vs H1"
##    dLL dDF    pvalue
## 1 0.59   4 0.8818897
## [1] "Testing H3 vs H2"
##   dLL dDF    pvalue
## 1 0.5   2 0.6051338
```

```
##All
print(IFNy)
```

```
## $H0
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 21.06964395  0.05978311  5.00000000
##
## Log-likelihood: -941.43
## Best method: bobyqa
##
## $H1
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 20.84145207 21.31893894  0.07048561  5.00000000
##
## Log-likelihood: -941.15
## Best method: bobyqa
##
## $H2
## $H2$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
```

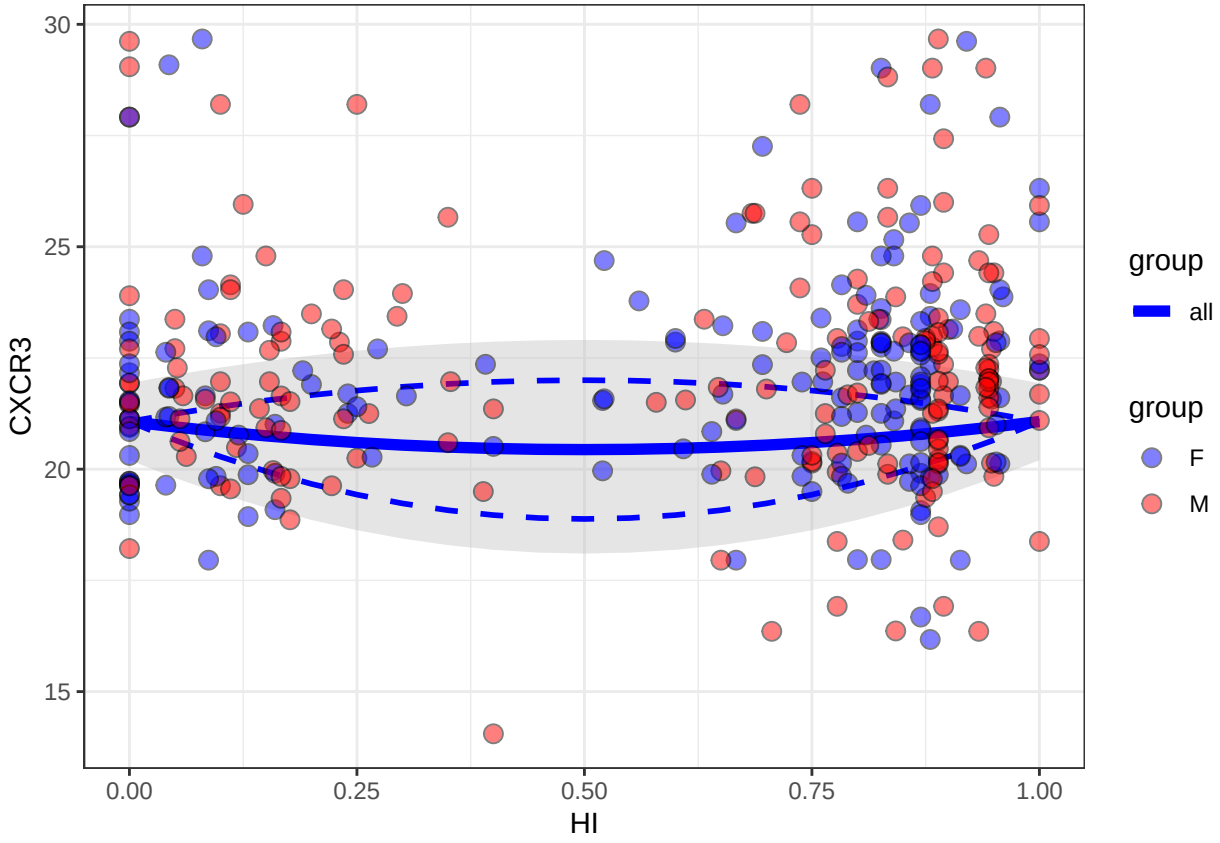
```
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 20.76691119  0.03505679  5.00000000
##
## Log-likelihood: -465.22
## Best method: bobyqa
##
## $H2$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 21.35517115  0.08226393  5.00000000
##
## Log-likelihood: -475.84
## Best method: bobyqa
##
##
## $H3
## $H3$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1         L2      alpha    myshape
## 20.3935446 21.3147118  0.0697062  5.0000000
##
## Log-likelihood: -464.72
## Best method: bobyqa
##
## $H3$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
```

```
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 21.31744395 21.38636087  0.08281592  5.00000000
##
## Log-likelihood: -475.84
## Best method: bobyqa
```

```
bananaPlot(mod = IFNy$H0,
             data = field,
             response = "CXCR3",
             group = "Sex") +
  scale_fill_manual(values = c("blue", "red")) +
  scale_color_manual(values = c("blue", "red")) +
  theme_bw()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```



```r
field$Sex <- as.factor(field$Sex)

speparam <- parasiteLoad::getParamBounds("weibull", data = field, response = "CASP1")

IFNy <- parasiteLoad::analyse(data = field,
                        response = "CASP1",
                        model = "weibull",
                        group = "Sex")
```

```
## [1] "Analysing data for response: CASP1"
## [1] "Fit for the response: CASP1"
## [1] "Fitting for all"
## [1] "Fitting model basic without alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting for groupA : F"
## [1] "Fitting model basic without alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting for groupB : M"
## [1] "Fitting model basic without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model basic with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Testing H0 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.07   1 0.7095437
## [1] "Testing H1 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.12   1 0.6176693
## [1] "Testing H2 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.03   1 0.7979228
## [1] "Testing H2 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.32   1 0.4219593
## [1] "Testing H3 groupA no alpha vs alpha"
##    dLL dDF   pvalue
## 1 0.01   1 0.918424
## [1] "Testing H3 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.34   1 0.4062534
## [1] "Testing H1 vs H0"
##    dLL dDF    pvalue
## 1 0.26   1 0.4672829
## [1] "Testing H2 vs H0"
```

```
##    dLL dDF   pvalue
## 1 0.29   3 0.899296
## [1] "Testing H3 vs H1"
##    dLL dDF    pvalue
## 1 0.24   4 0.9754971
## [1] "Testing H3 vs H2"
##    dLL dDF    pvalue
## 1 0.21   2 0.8105453
```

```
##All
print(IFNy)
```

```
## $H0
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 21.44597637  0.02768908  5.00000000
##
## Log-likelihood: -940.91
## Best method: bobyqa
##
## $H1
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 21.21918629 21.68746492  0.03748458  5.00000000
##
## Log-likelihood: -940.65
## Best method: bobyqa
##
## $H2
## $H2$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
```

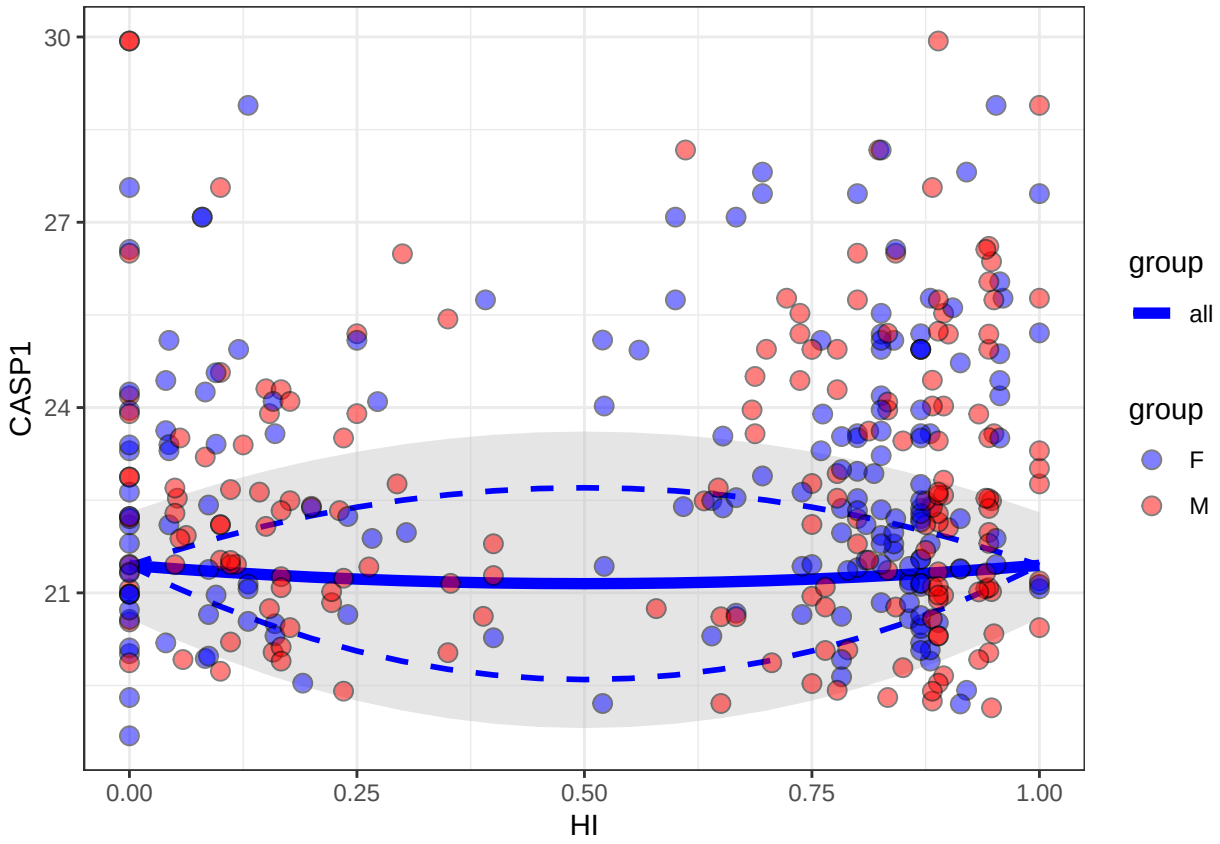
```
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 21.20758616 -0.02702379  5.00000000
##
## Log-likelihood: -468.37
## Best method: bobyqa
##
## $H2$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 21.69093740  0.08429205  5.00000000
##
## Log-likelihood: -472.25
## Best method: bobyqa
##
##
## $H3
## $H3$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1         L2      alpha    myshape
## 21.0271434 21.4677162 -0.0112422  5.0000000
##
## Log-likelihood: -468.25
## Best method: bobyqa
##
## $H3$groupB
##
```

```
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 21.46719243 21.86842933  0.08695199  5.00000000
##
## Log-likelihood: -472.15
## Best method: bobyqa
```

```
bananaPlot(mod = IFNy$H0,
             data = field,
             response = "CASP1",
             group = "Sex") +
  scale_fill_manual(values = c("blue", "red")) +
  scale_color_manual(values = c("blue", "red")) +
  theme_bw()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```




```
field$Sex <- as.factor(field$Sex)

speparam <- parasiteLoad::getParamBounds("weibull", data = field, response = "CXCL9")

IFNy <- parasiteLoad::analyse(data = field,
                          response = "CXCL9",
                          model = "weibull",
                          group = "Sex")
```

```
## [1] "Analysing data for response: CXCL9"
## [1] "Fit for the response: CXCL9"
## [1] "Fitting for all"
## [1] "Fitting model basic without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model basic with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
```

```
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting for groupA : F"
## [1] "Fitting model basic without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model basic with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting for groupB : M"
## [1] "Fitting model basic without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model basic with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Testing H0 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.29   1 0.4456462
## [1] "Testing H1 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.33   1 0.4138825
## [1] "Testing H2 groupA no alpha vs alpha"
##    dLL dDF   pvalue
## 1 0.01   1 0.917288
## [1] "Testing H2 groupB no alpha vs alpha"
##   dLL dDF    pvalue
## 1 0.8   1 0.2064454
## [1] "Testing H3 groupA no alpha vs alpha"
##   dLL dDF    pvalue
## 1   0   1 0.9980445
## [1] "Testing H3 groupB no alpha vs alpha"
##   dLL dDF    pvalue
## 1 0.8   1 0.2065262
## [1] "Testing H1 vs H0"
##    dLL dDF    pvalue
## 1 0.08   1 0.6968073
## [1] "Testing H2 vs H0"
##    dLL dDF    pvalue
## 1 0.87   3 0.6261238
## [1] "Testing H3 vs H1"
##    dLL dDF    pvalue
## 1 0.88   4 0.7806923
## [1] "Testing H3 vs H2"
##    dLL dDF    pvalue
## 1 0.08   2 0.9240358
```

```
##All
print(IFNy)
```

```
## $H0
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 20.33887663  0.05905836  5.00000000
##
## Log-likelihood: -968.17
```

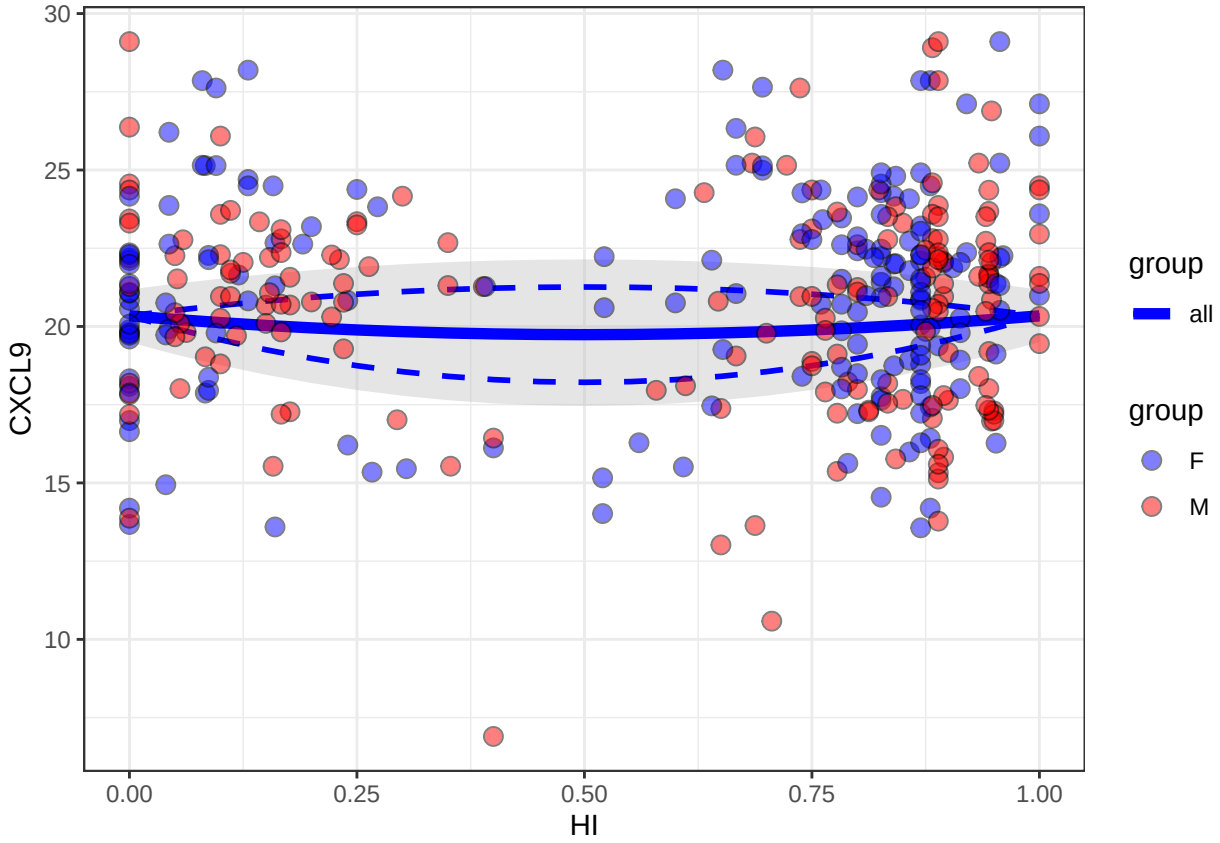
```
## Best method: bobyqa
##
## $H1
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 20.22351710 20.46051394  0.06399431  5.00000000
##
## Log-likelihood: -968.09
## Best method: bobyqa
##
## $H2
## $H2$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 20.19208231 -0.01154039  5.00000000
##
## Log-likelihood: -484.29
## Best method: bobyqa
##
## $H2$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1      alpha    myshape
## 20.5049387  0.1366603  5.0000000
```

```
##
## Log-likelihood: -483.01
## Best method: L-BFGS-B
##
##
## $H3
## $H3$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##           L1           L2        alpha      myshape
## 2.004487e+01 2.039149e+01 2.712052e-04 5.000000e+00
##
## Log-likelihood: -484.21
## Best method: bobyqa
##
## $H3$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1         L2      alpha    myshape
## 20.4871080 20.5193241  0.1368788  5.0000000
##
## Log-likelihood: -483.01
## Best method: bobyqa
```

```
bananaPlot(mod = IFNy$H0,
             data = field,
             response = "CXCL9",
             group = "Sex") +
  scale_fill_manual(values = c("blue", "red")) +
  scale_color_manual(values = c("blue", "red")) +
  theme_bw()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```




```
field$Sex <- as.factor(field$Sex)

speparam <- parasiteLoad::getParamBounds("weibull", data = field, response = "IDO1")

IFNy <- parasiteLoad::analyse(data = field,
                          response = "IDO1",
                          model = "weibull",
                          group = "Sex")
```

```
## [1] "Analysing data for response: IDO1"
## [1] "Fit for the response: IDO1"
## [1] "Fitting for all"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Fitting for groupA : F"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
```

```
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Fitting for groupB : M"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Testing H0 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.01   1 0.8748353
## [1] "Testing H1 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.09   1 0.6759373
## [1] "Testing H2 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.05   1 0.7573443
## [1] "Testing H2 groupB no alpha vs alpha"
##   dLL dDF   pvalue
## 1 0.1   1 0.660795
## [1] "Testing H3 groupA no alpha vs alpha"
##   dLL dDF   pvalue
## 1   0   1 0.969714
## [1] "Testing H3 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.13   1 0.6066322
## [1] "Testing H1 vs H0"
##    dLL dDF    pvalue
## 1 0.64   1 0.2581624
## [1] "Testing H2 vs H0"
##    dLL dDF    pvalue
## 1 1.58   3 0.3668558
## [1] "Testing H3 vs H1"
##    dLL dDF    pvalue
## 1 1.68   4 0.4995973
## [1] "Testing H3 vs H2"
##    dLL dDF    pvalue
## 1 0.74   2 0.4788598
```

```
##All
print(IFNy)
```

```
## $H0
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
```

```
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 15.17832328  0.01570283  3.67259140
##
## Log-likelihood: -1032.74
## Best method: bobyqa
##
## $H1
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 14.84999389 15.56234734  0.04229992  3.67087784
##
## Log-likelihood: -1032.1
## Best method: bobyqa
##
## $H2
## $H2$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 15.14011358 -0.04353434  3.94632263
##
## Log-likelihood: -506.41
## Best method: bobyqa
##
## $H2$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
```

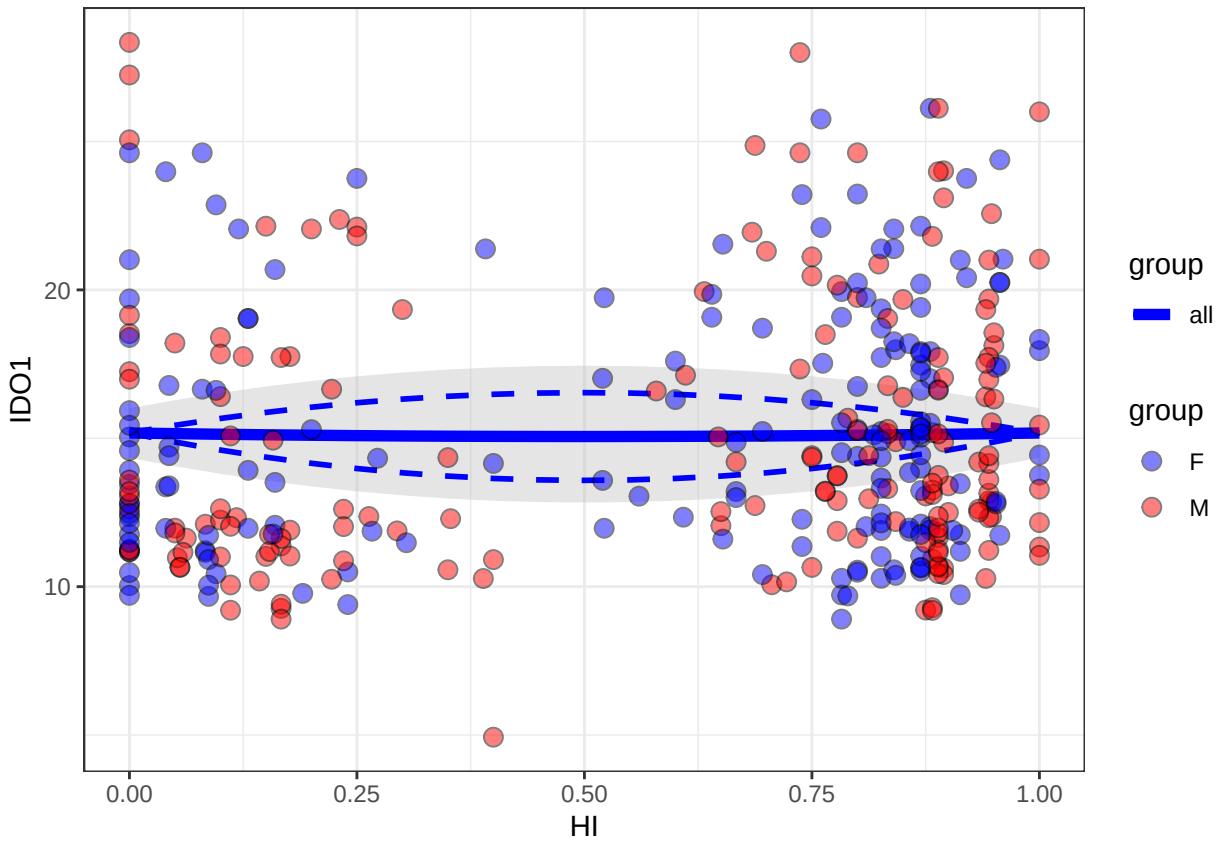
```
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 15.17803361  0.06249617  3.45903628
##
## Log-likelihood: -524.74
## Best method: bobyqa
##
##
## $H3
## $H3$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##           L1           L2        alpha      myshape
## 14.759481427 15.697332092  0.005473685  3.948013909
##
## Log-likelihood: -505.81
## Best method: bobyqa
##
## $H3$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 14.93962843 15.40727627  0.07387447  3.45669217
##
## Log-likelihood: -524.62
## Best method: bobyqa
```

```
bananaPlot(mod = IFNy$H0,
           data = field,
```

```
            response = "IDO1",
            group = "Sex") +
  scale_fill_manual(values = c("blue", "red")) +
  scale_color_manual(values = c("blue", "red")) +
  theme_bw()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```




```
field$Sex <- as.factor(field$Sex)

speparam <- parasiteLoad::getParamBounds("weibull", data = field, response = "IRGM1")

IFNy <- parasiteLoad::analyse(data = field,
                          response = "IRGM1",
                          model = "weibull",
                          group = "Sex")
```

```
## [1] "Analysing data for response: IRGM1"
## [1] "Fit for the response: IRGM1"
## [1] "Fitting for all"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
```

```
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Fitting for groupA : F"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Fitting for groupB : M"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Testing H0 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.32   1 0.4216119
## [1] "Testing H1 no alpha vs alpha"
##    dLL dDF   pvalue
## 1 0.41   1 0.364167
## [1] "Testing H2 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.26   1 0.4718911
## [1] "Testing H2 groupB no alpha vs alpha"
##   dLL dDF    pvalue
## 1   0   1 0.9798968
## [1] "Testing H3 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.63   1 0.2631179
## [1] "Testing H3 groupB no alpha vs alpha"
##   dLL dDF    pvalue
## 1   0   1 0.9803298
## [1] "Testing H1 vs H0"
##    dLL dDF    pvalue
## 1 0.21   1 0.5183066
## [1] "Testing H2 vs H0"
##    dLL dDF       pvalue
## 1 9.42   3 0.0002956138
## [1] "Testing H3 vs H1"
##     dLL dDF      pvalue
## 1 10.62   4 0.000282791
## [1] "Testing H3 vs H2"
##    dLL dDF    pvalue
## 1 1.41   2 0.2430339
```

```
##All
print(IFNy)
```

```
## $H0
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 11.22656174 -0.08936173  3.27063298
##
## Log-likelihood: -955.68
## Best method: bobyqa
##
## $H1
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1         L2      alpha    myshape
## 11.3880620 11.0568098 -0.1028669  3.2753228
##
## Log-likelihood: -955.47
## Best method: bobyqa
##
## $H2
## $H2$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1      alpha    myshape
```

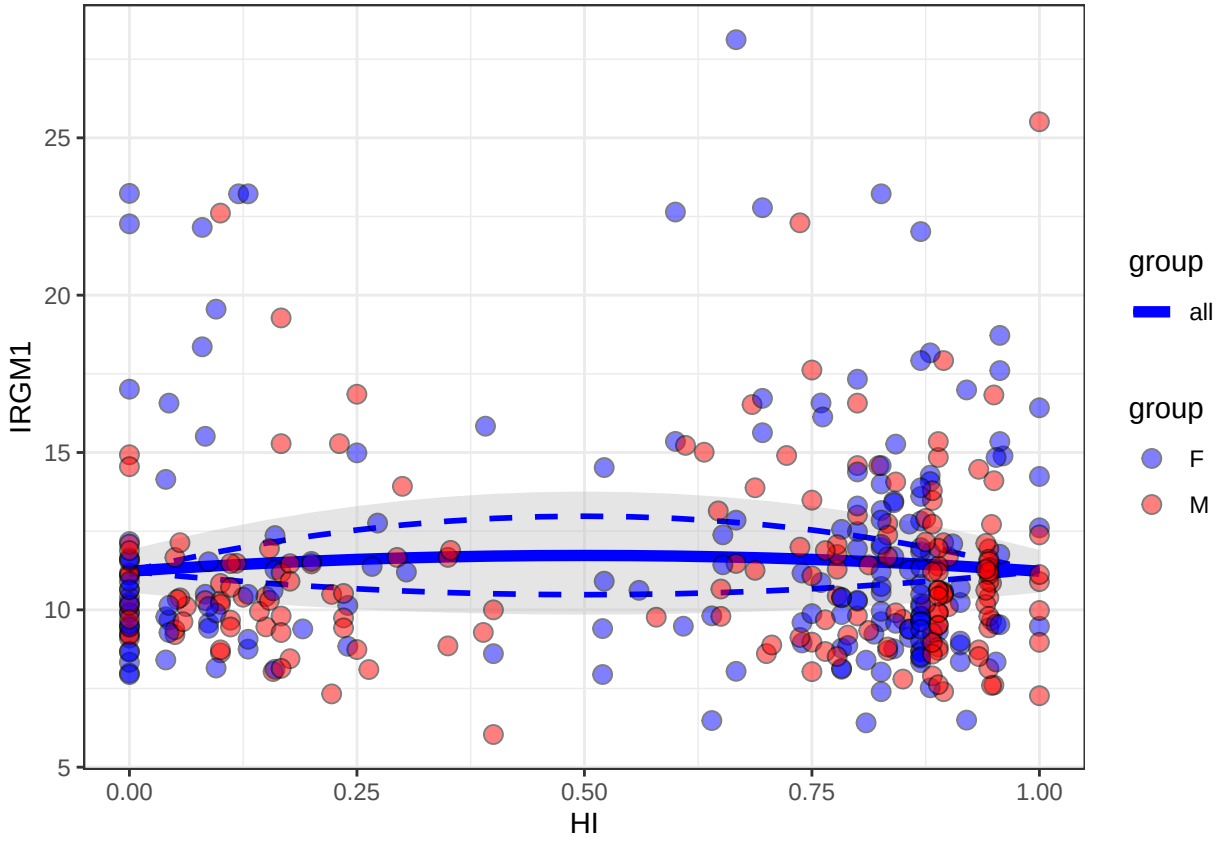
```
## 11.5322347 -0.1209123  3.0559623
##
## Log-likelihood: -495.52
## Best method: bobyqa
##
## $H2$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##           L1        alpha      myshape
## 11.066045510  0.003499006  3.696406427
##
## Log-likelihood: -450.75
## Best method: bobyqa
##
##
## $H3
## $H3$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1         L2      alpha    myshape
## 11.9670558 10.8855435 -0.2024564  3.0861690
##
## Log-likelihood: -494.65
## Best method: bobyqa
##
## $H3$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
```

```
##     control = config$control)
##
## Coefficients:
##           L1           L2        alpha      myshape
## 10.669511067 11.320659481 -0.003421731  3.711355847
##
## Log-likelihood: -450.2
## Best method: bobyqa
```

```
bananaPlot(mod = IFNy$H0,
             data = field,
             response = "IRGM1",
             group = "Sex") +
  scale_fill_manual(values = c("blue", "red")) +
  scale_color_manual(values = c("blue", "red")) +
  theme_bw()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```




```
field$Sex <- as.factor(field$Sex)

speparam <- parasiteLoad::getParamBounds("weibull", data = field, response = "MPO")

IFNy <- parasiteLoad::analyse(data = field,
                         response = "MPO",
```

```
                        model = "weibull",
                        group = "Sex")
```

```
## [1] "Analysing data for response: MPO"
## [1] "Fit for the response: MPO"
## [1] "Fitting for all"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Fitting for groupA : F"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Fitting for groupB : M"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Testing H0 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.27   1 0.4610341
## [1] "Testing H1 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.02   1 0.8427439
## [1] "Testing H2 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.07   1 0.6992651
## [1] "Testing H2 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.21   1 0.5135939
## [1] "Testing H3 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.05   1 0.7502735
## [1] "Testing H3 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.13   1 0.6124176
## [1] "Testing H1 vs H0"
##    dLL dDF     pvalue
## 1 3.26   1 0.01065468
## [1] "Testing H2 vs H0"
```

```
##    dLL dDF    pvalue
## 1 1.23   3 0.4840111
## [1] "Testing H3 vs H1"
##    dLL dDF    pvalue
## 1 1.81   4 0.4608187
## [1] "Testing H3 vs H2"
##    dLL dDF     pvalue
## 1 3.84   2 0.02145585
```

```
##All
print(IFNy)
```

```
## $H0
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 19.17550539 -0.06042347  4.64090412
##
## Log-likelihood: -1045.14
## Best method: bobyqa
##
## $H1
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 18.43201151 20.03697182 -0.01605671  4.68225355
##
## Log-likelihood: -1041.88
## Best method: bobyqa
##
## $H2
## $H2$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
```

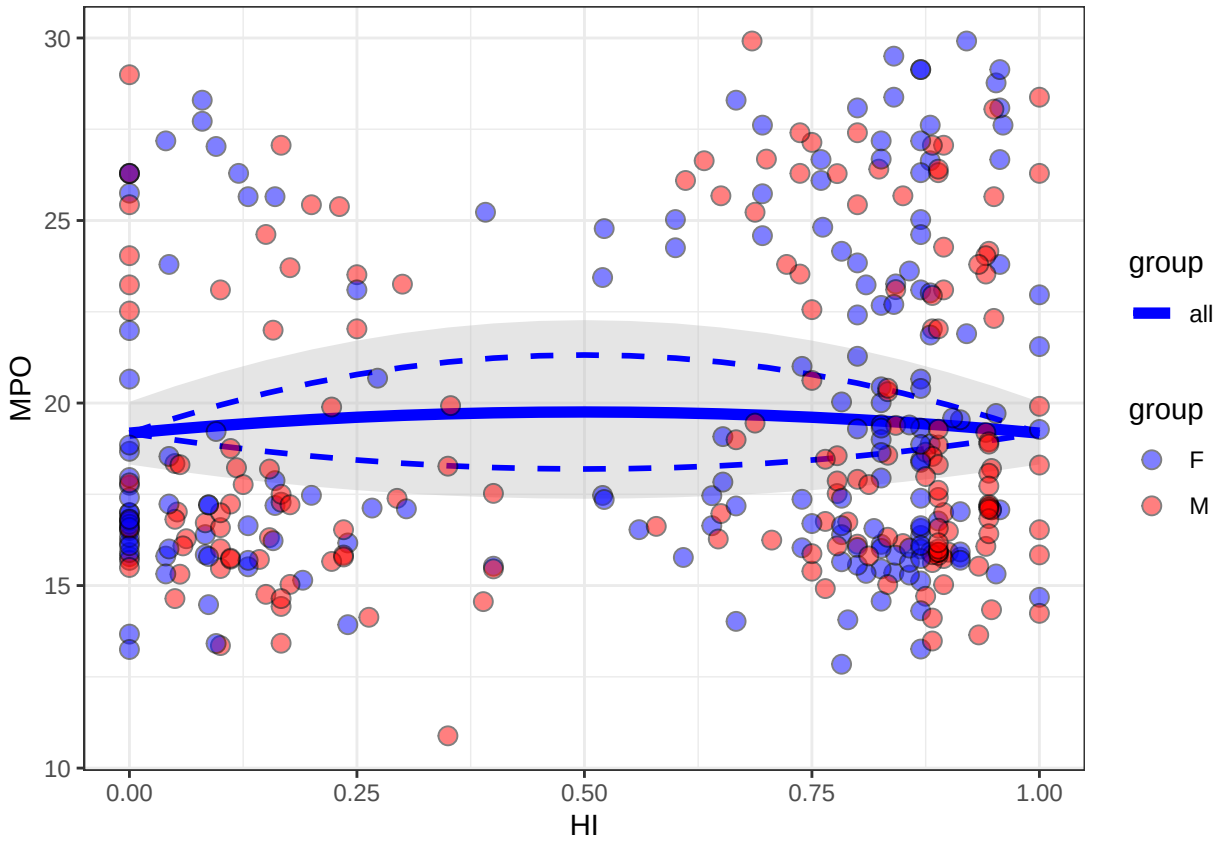
```
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 19.50478364 -0.04869596  4.58221592
##
## Log-likelihood: -526.39
## Best method: bobyqa
##
## $H2$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 18.86620648 -0.06880878  4.73850382
##
## Log-likelihood: -517.52
## Best method: bobyqa
##
##
## $H3
## $H3$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 18.48894157 20.92642485  0.03895738  4.66909307
##
## Log-likelihood: -523.01
## Best method: bobyqa
##
## $H3$groupB
##
```

```
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 18.45628993 19.26289416 -0.05328589  4.74755711
##
## Log-likelihood: -517.06
## Best method: bobyqa
```

```
bananaPlot(mod = IFNy$H0,
             data = field,
             response = "MPO",
             group = "Sex") +
  scale_fill_manual(values = c("blue", "red")) +
  scale_color_manual(values = c("blue", "red")) +
  theme_bw()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```




```
field$Sex <- as.factor(field$Sex)

speparam <- parasiteLoad::getParamBounds("weibull", data = field, response = "MUC2")

IFNy <- parasiteLoad::analyse(data = field,
                          response = "MUC2",
                          model = "weibull",
                          group = "Sex")
```

```
## [1] "Analysing data for response: MUC2"
## [1] "Fit for the response: MUC2"
## [1] "Fitting for all"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Fitting for groupA : F"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
```

```
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Fitting for groupB : M"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Testing H0 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.44   1 0.3494405
## [1] "Testing H1 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.45   1 0.3405081
## [1] "Testing H2 groupA no alpha vs alpha"
##   dLL dDF    pvalue
## 1   0   1 0.9373544
## [1] "Testing H2 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.91   1 0.1772277
## [1] "Testing H3 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.01   1 0.8713542
## [1] "Testing H3 groupB no alpha vs alpha"
##    dLL dDF  pvalue
## 1 0.87   1 0.18644
## [1] "Testing H1 vs H0"
##    dLL dDF    pvalue
## 1 0.02   1 0.8556694
## [1] "Testing H2 vs H0"
##    dLL dDF   pvalue
## 1 1.35   3 0.438556
## [1] "Testing H3 vs H1"
##    dLL dDF    pvalue
## 1 1.42   4 0.5861176
## [1] "Testing H3 vs H2"
##    dLL dDF    pvalue
## 1 0.08   2 0.9247695
```

```
##All
print(IFNy)
```

```
## $H0
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
```

```
##
## Coefficients:
##         L1      alpha    myshape
##  9.9469606 -0.1247795  2.6879567
##
## Log-likelihood: -980.67
## Best method: bobyqa
##
## $H1
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1         L2      alpha    myshape
##  9.9955568  9.8942658 -0.1298942  2.6889723
##
## Log-likelihood: -980.66
## Best method: bobyqa
##
## $H2
## $H2$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 10.43222132 -0.01450596  2.66264451
##
## Log-likelihood: -496.24
## Best method: bobyqa
##
## $H2$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
```

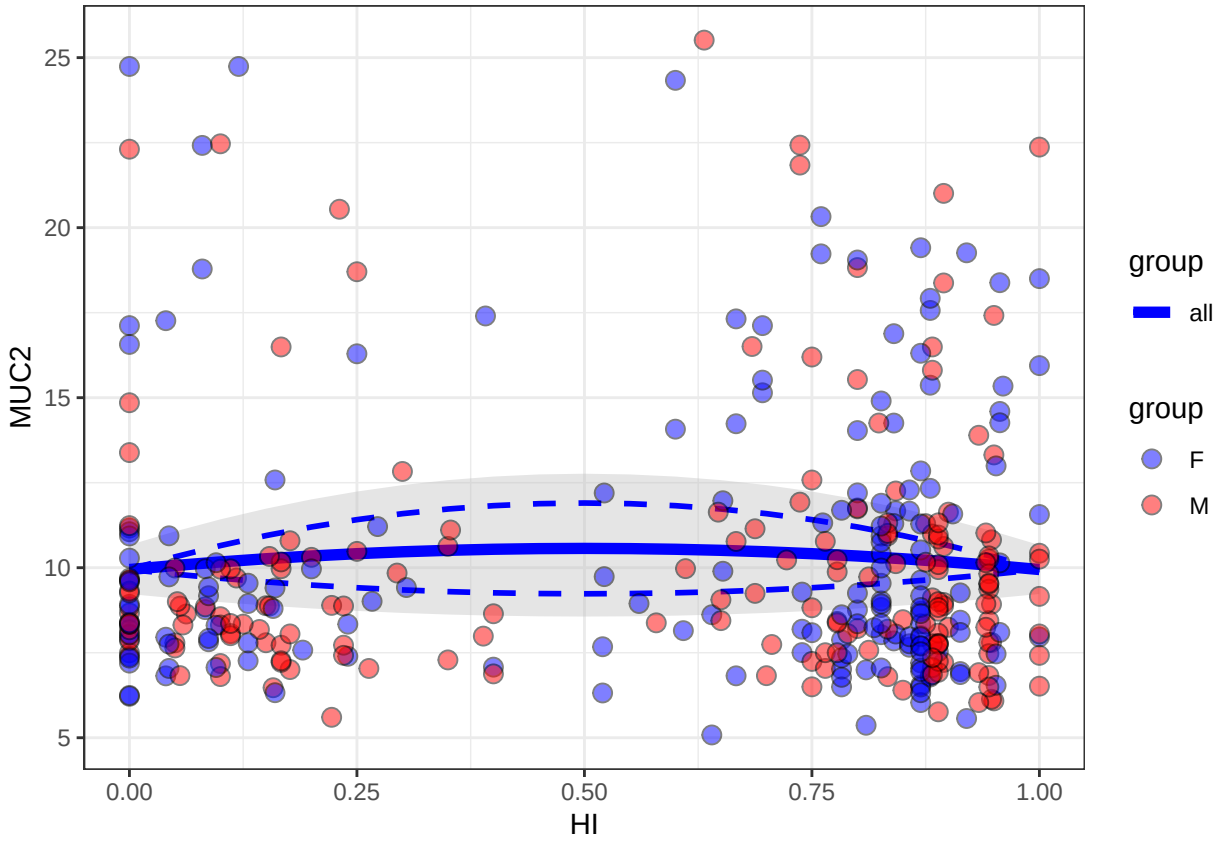
```
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1      alpha    myshape
##  9.4369379 -0.2595038  2.7319104
##
## Log-likelihood: -483.08
## Best method: bobyqa
##
##
## $H3
## $H3$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 10.54518538 10.28574104 -0.03130882  2.66643945
##
## Log-likelihood: -496.2
## Best method: bobyqa
##
## $H3$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1         L2      alpha    myshape
##  9.3391545  9.5218277 -0.2544648  2.7315536
##
## Log-likelihood: -483.05
## Best method: bobyqa
```

```
bananaPlot(mod = IFNy$H0,
           data = field,
           response = "MUC2",
           group = "Sex") +
```

```
  scale_fill_manual(values = c("blue", "red")) +
  scale_color_manual(values = c("blue", "red")) +
  theme_bw()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```




```
field$Sex <- as.factor(field$Sex)

speparam <- parasiteLoad::getParamBounds("weibull", data = field, response = "MUC5AC")

IFNy <- parasiteLoad::analyse(data = field,
                          response = "MUC5AC",
                          model = "weibull",
                          group = "Sex")
```

```
## [1] "Analysing data for response: MUC5AC"
## [1] "Fit for the response: MUC5AC"
## [1] "Fitting for all"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
```

```
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Fitting for groupA : F"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Fitting for groupB : M"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Testing H0 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.22   1 0.5074506
## [1] "Testing H1 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.11   1 0.6325418
## [1] "Testing H2 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.01   1 0.8667258
## [1] "Testing H2 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.33   1 0.4163933
## [1] "Testing H3 groupA no alpha vs alpha"
##   dLL dDF    pvalue
## 1   0   1 0.9912138
## [1] "Testing H3 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.27   1 0.4642135
## [1] "Testing H1 vs H0"
##    dLL dDF   pvalue
## 1 0.32   1 0.422614
## [1] "Testing H2 vs H0"
##    dLL dDF    pvalue
## 1 0.83   3 0.6478591
## [1] "Testing H3 vs H1"
##    dLL dDF    pvalue
## 1 0.96   4 0.7515028
## [1] "Testing H3 vs H2"
##    dLL dDF    pvalue
## 1 0.45   2 0.6356185
```

```
##All
print(IFNy)
```

```
## $H0
```

```
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1      alpha    myshape
## 10.8800554 -0.1019045  2.4583481
##
## Log-likelihood: -1039.9
## Best method: bobyqa
##
## $H1
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 10.63239552 11.17018364 -0.07420493  2.45903404
##
## Log-likelihood: -1039.57
## Best method: bobyqa
##
## $H2
## $H2$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 11.17872879 -0.03693447  2.39233194
##
## Log-likelihood: -525.46
## Best method: bobyqa
```

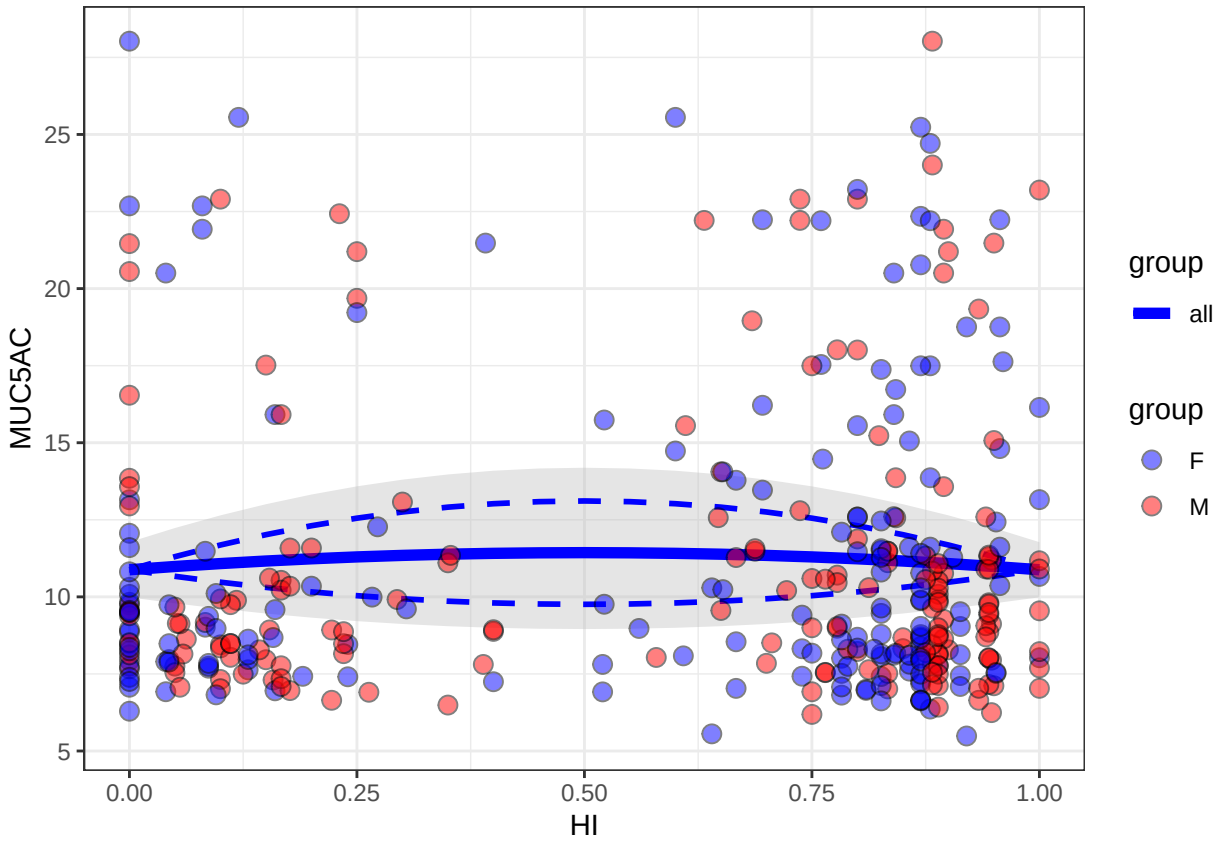
```
##
## $H2$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1      alpha    myshape
## 10.5746961 -0.1739501  2.5370220
##
## Log-likelihood: -513.61
## Best method: bobyqa
##
##
## $H3
## $H3$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##           L1           L2        alpha      myshape
## 10.982065381 11.468748390 -0.002510274  2.390792131
##
## Log-likelihood: -525.35
## Best method: bobyqa
##
## $H3$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1         L2      alpha    myshape
```

```
## 10.1961953 10.9195949 -0.1548015  2.5437995
##
## Log-likelihood: -513.27
## Best method: bobyqa
```

```
bananaPlot(mod = IFNy$H0,
             data = field,
             response = "MUC5AC",
             group = "Sex") +
  scale_fill_manual(values = c("blue", "red")) +
  scale_color_manual(values = c("blue", "red")) +
  theme_bw()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```




```
field$Sex <- as.factor(field$Sex)

speparam <- parasiteLoad::getParamBounds("weibull", data = field, response = "MYD88")

IFNy <- parasiteLoad::analyse(data = field,
                          response = "MYD88",
                          model = "weibull",
                          group = "Sex")
```

```
## [1] "Analysing data for response: MYD88"
```

```
## [1] "Fit for the response: MYD88"
## [1] "Fitting for all"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Fitting for groupA : F"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Fitting for groupB : M"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Testing H0 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.79   1 0.2097717
## [1] "Testing H1 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.34   1 0.4094372
## [1] "Testing H2 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 1.03   1 0.1514522
## [1] "Testing H2 groupB no alpha vs alpha"
##    dLL dDF   pvalue
## 1 0.06   1 0.737854
## [1] "Testing H3 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.29   1 0.4471382
## [1] "Testing H3 groupB no alpha vs alpha"
##    dLL dDF   pvalue
## 1 0.05   1 0.750039
## [1] "Testing H1 vs H0"
##    dLL dDF     pvalue
## 1 2.39   1 0.02882543
## [1] "Testing H2 vs H0"
##    dLL dDF    pvalue
## 1 0.96   3 0.5902559
## [1] "Testing H3 vs H1"
##    dLL dDF    pvalue
```

```
## 1 0.78   4 0.8146888
## [1] "Testing H3 vs H2"
##    dLL dDF    pvalue
## 1 2.22   2 0.1091409
```

```
##All
print(IFNy)
```

```
## $H0
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1      alpha    myshape
## 16.2028979 -0.1715168  3.2242197
##
## Log-likelihood: -1133.85
## Best method: bobyqa
##
## $H1
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1         L2      alpha    myshape
## 15.4279025 17.0953502 -0.1112175  3.2289018
##
## Log-likelihood: -1131.46
## Best method: bobyqa
##
## $H2
## $H2$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
```

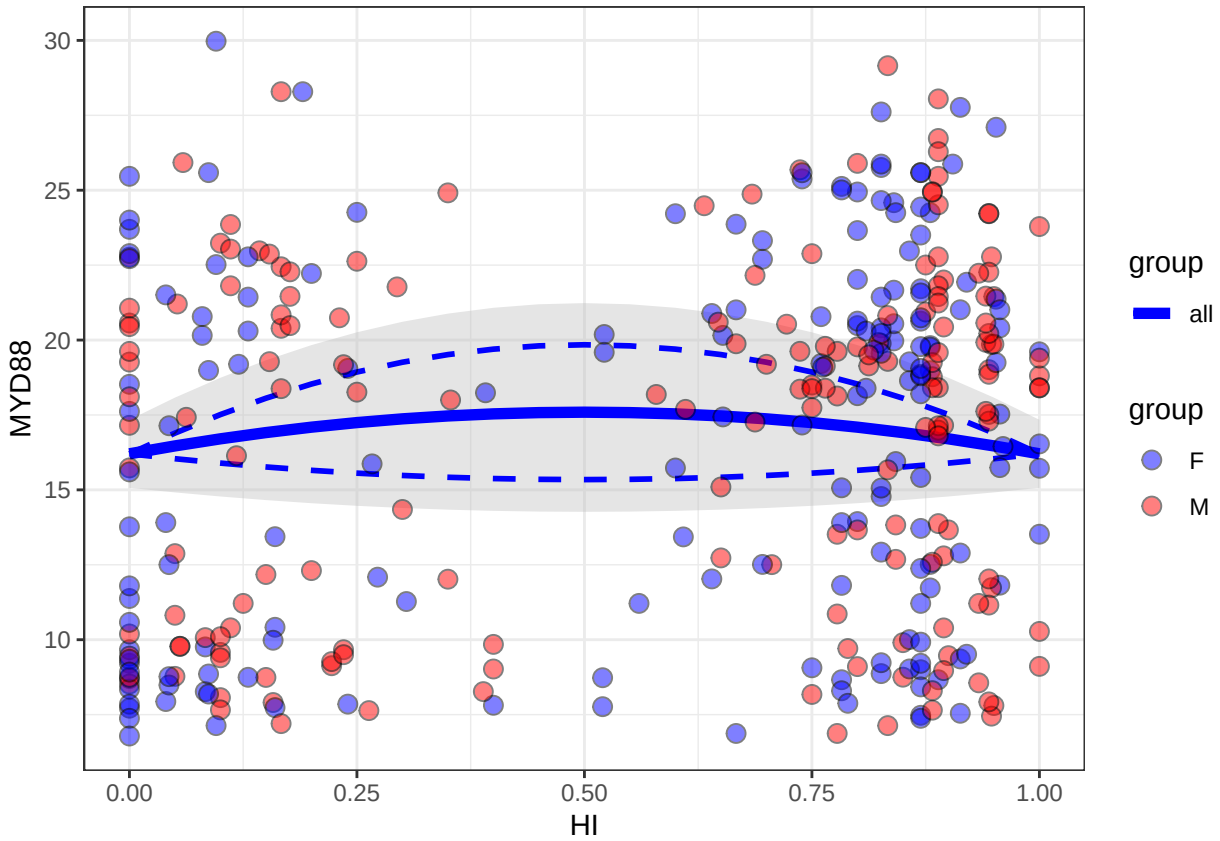
```
##     control = config$control)
##
## Coefficients:
##         L1      alpha    myshape
## 15.6723768 -0.3014729  3.0811816
##
## Log-likelihood: -570.63
## Best method: bobyqa
##
## $H2$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 16.68893205 -0.05934728  3.38790276
##
## Log-likelihood: -562.26
## Best method: bobyqa
##
##
## $H3
## $H3$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1         L2      alpha    myshape
## 15.0488790 16.8995836 -0.1635321  3.0811847
##
## Log-likelihood: -569.4
## Best method: bobyqa
##
## $H3$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
```

```
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 15.85227466 17.28434955 -0.05552599  3.39720867
##
## Log-likelihood: -561.28
## Best method: bobyqa
```

```
bananaPlot(mod = IFNy$H0,
             data = field,
             response = "MYD88",
             group = "Sex") +
  scale_fill_manual(values = c("blue", "red")) +
  scale_color_manual(values = c("blue", "red")) +
  theme_bw()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```




```
field$Sex <- as.factor(field$Sex)
```

```
speparam <- parasiteLoad::getParamBounds("weibull", data = field, response = "NCR1")

IFNy <- parasiteLoad::analyse(data = field,
                          response = "NCR1",
                          model = "weibull",
                          group = "Sex")
```

```
## [1] "Analysing data for response: NCR1"
## [1] "Fit for the response: NCR1"
## [1] "Fitting for all"
## [1] "Fitting model basic without alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting for groupA : F"
## [1] "Fitting model basic without alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting for groupB : M"
## [1] "Fitting model basic without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model basic with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Testing H0 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.76   1 0.2163148
## [1] "Testing H1 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.71   1 0.2332369
## [1] "Testing H2 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.27   1 0.4600683
## [1] "Testing H2 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.53   1 0.3040436
## [1] "Testing H3 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.34   1 0.4074304
## [1] "Testing H3 groupB no alpha vs alpha"
##   dLL dDF    pvalue
## 1 0.5   1 0.3184993
## [1] "Testing H1 vs H0"
##   dLL dDF    pvalue
## 1   0   1 0.9381331
## [1] "Testing H2 vs H0"
##    dLL dDF    pvalue
## 1 0.14   3 0.9639115
```

```
## [1] "Testing H3 vs H1"
##    dLL dDF    pvalue
## 1 0.35   4 0.9519389
## [1] "Testing H3 vs H2"
##    dLL dDF    pvalue
## 1 0.21   2 0.8097843
```

```
##All
print(IFNy)
```

```
## $H0
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 23.60790897  0.09354245  5.00000000
##
## Log-likelihood: -962.17
## Best method: bobyqa
##
## $H1
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 23.63335201 23.57825876  0.09234923  5.00000000
##
## Log-likelihood: -962.17
## Best method: bobyqa
##
## $H2
## $H2$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
```

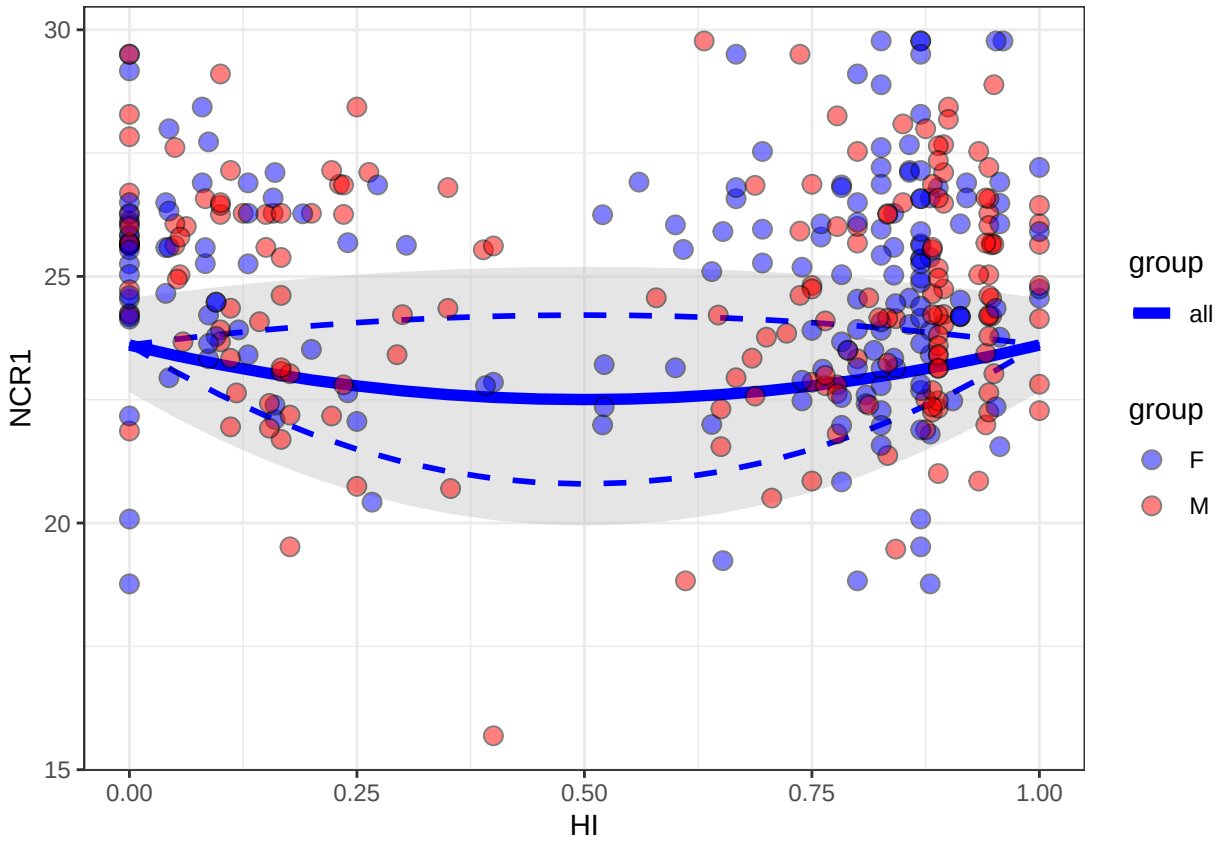
```
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 23.64154580  0.07831271  5.00000000
##
## Log-likelihood: -480.86
## Best method: bobyqa
##
## $H2$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##        L1     alpha   myshape
## 23.583430  0.111089  5.000000
##
## Log-likelihood: -481.18
## Best method: bobyqa
##
##
## $H3
## $H3$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1         L2      alpha    myshape
## 23.4868499 23.8962929  0.0932946  5.0000000
##
## Log-likelihood: -480.78
## Best method: bobyqa
##
## $H3$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
```

```
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1         L2      alpha    myshape
## 23.8664226 23.3554200  0.1085986  5.0000000
##
## Log-likelihood: -481.04
## Best method: bobyqa
```

```
bananaPlot(mod = IFNy$H0,
             data = field,
             response = "NCR1",
             group = "Sex") +
  scale_fill_manual(values = c("blue", "red")) +
  scale_color_manual(values = c("blue", "red")) +
  theme_bw()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```



```r
field$Sex <- as.factor(field$Sex)

speparam <- parasiteLoad::getParamBounds("weibull", data = field, response = "PRF1")

IFNy <- parasiteLoad::analyse(data = field,
                        response = "PRF1",
                        model = "weibull",
                        group = "Sex")
```

```
## [1] "Analysing data for response: PRF1"
## [1] "Fit for the response: PRF1"
## [1] "Fitting for all"
## [1] "Fitting model basic without alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting for groupA : F"
## [1] "Fitting model basic without alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting for groupB : M"
## [1] "Fitting model basic without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model basic with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Testing H0 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.26   1 0.4729116
## [1] "Testing H1 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.13   1 0.6056152
## [1] "Testing H2 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.26   1 0.4677997
## [1] "Testing H2 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.04   1 0.7896671
## [1] "Testing H3 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.12   1 0.6217862
## [1] "Testing H3 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.04   1 0.7912139
## [1] "Testing H1 vs H0"
##    dLL dDF    pvalue
## 1 0.56   1 0.2907995
## [1] "Testing H2 vs H0"
```

```
##    dLL dDF   pvalue
## 1 0.05   3 0.991903
## [1] "Testing H3 vs H1"
##    dLL dDF    pvalue
## 1 0.03   4 0.9994568
## [1] "Testing H3 vs H2"
##    dLL dDF    pvalue
## 1 0.54   2 0.5818403
```

```
##All
print(IFNy)
```

```
## $H0
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 22.88163888  0.05381747  5.00000000
##
## Log-likelihood: -964.65
## Best method: bobyqa
##
## $H1
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 23.22061491 22.50154864  0.03979383  5.00000000
##
## Log-likelihood: -964.09
## Best method: bobyqa
##
## $H2
## $H2$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
```

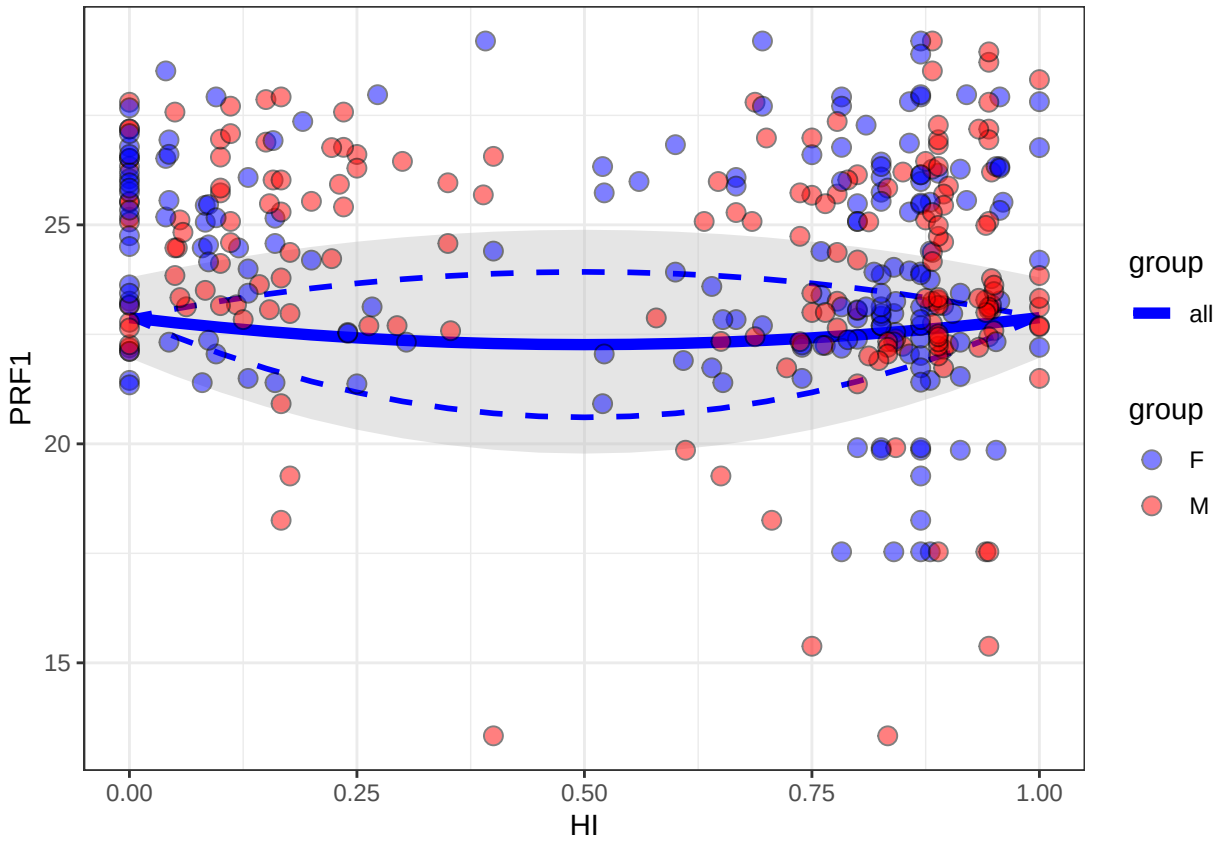
```
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 23.01514358  0.07322802  5.00000000
##
## Log-likelihood: -479.75
## Best method: bobyqa
##
## $H2$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 22.72645504  0.02993458  5.00000000
##
## Log-likelihood: -484.86
## Best method: bobyqa
##
##
## $H3
## $H3$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 23.23339196 22.66687326  0.05345807  5.00000000
##
## Log-likelihood: -479.59
## Best method: bobyqa
##
## $H3$groupB
##
```

```
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 23.20365139 22.37662943  0.02995278  5.00000000
##
## Log-likelihood: -484.47
## Best method: L-BFGS-B
```

```
bananaPlot(mod = IFNy$H0,
             data = field,
             response = "PRF1",
             group = "Sex") +
  scale_fill_manual(values = c("blue", "red")) +
  scale_color_manual(values = c("blue", "red")) +
  theme_bw()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```




```
field$Sex <- as.factor(field$Sex)

speparam <- parasiteLoad::getParamBounds("weibull", data = field, response = "RETNLB")

IFNy <- parasiteLoad::analyse(data = field,
                          response = "RETNLB",
                          model = "weibull",
                          group = "Sex")
```

```
## [1] "Analysing data for response: RETNLB"
## [1] "Fit for the response: RETNLB"
## [1] "Fitting for all"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Fitting for groupA : F"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
```

```
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Fitting for groupB : M"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Testing H0 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.43   1 0.3549992
## [1] "Testing H1 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.15   1 0.5893313
## [1] "Testing H2 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.19   1 0.5340153
## [1] "Testing H2 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.24   1 0.4895309
## [1] "Testing H3 groupA no alpha vs alpha"
##   dLL dDF   pvalue
## 1   0   1 0.962827
## [1] "Testing H3 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.21   1 0.5121995
## [1] "Testing H1 vs H0"
##    dLL dDF     pvalue
## 1 1.88   1 0.05233018
## [1] "Testing H2 vs H0"
##    dLL dDF    pvalue
## 1 1.67   3 0.3421117
## [1] "Testing H3 vs H1"
##    dLL dDF    pvalue
## 1 2.12   4 0.3734495
## [1] "Testing H3 vs H2"
##    dLL dDF     pvalue
## 1 2.34   2 0.09664512
```

```
##All
print(IFNy)
```

```
## $H0
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
```

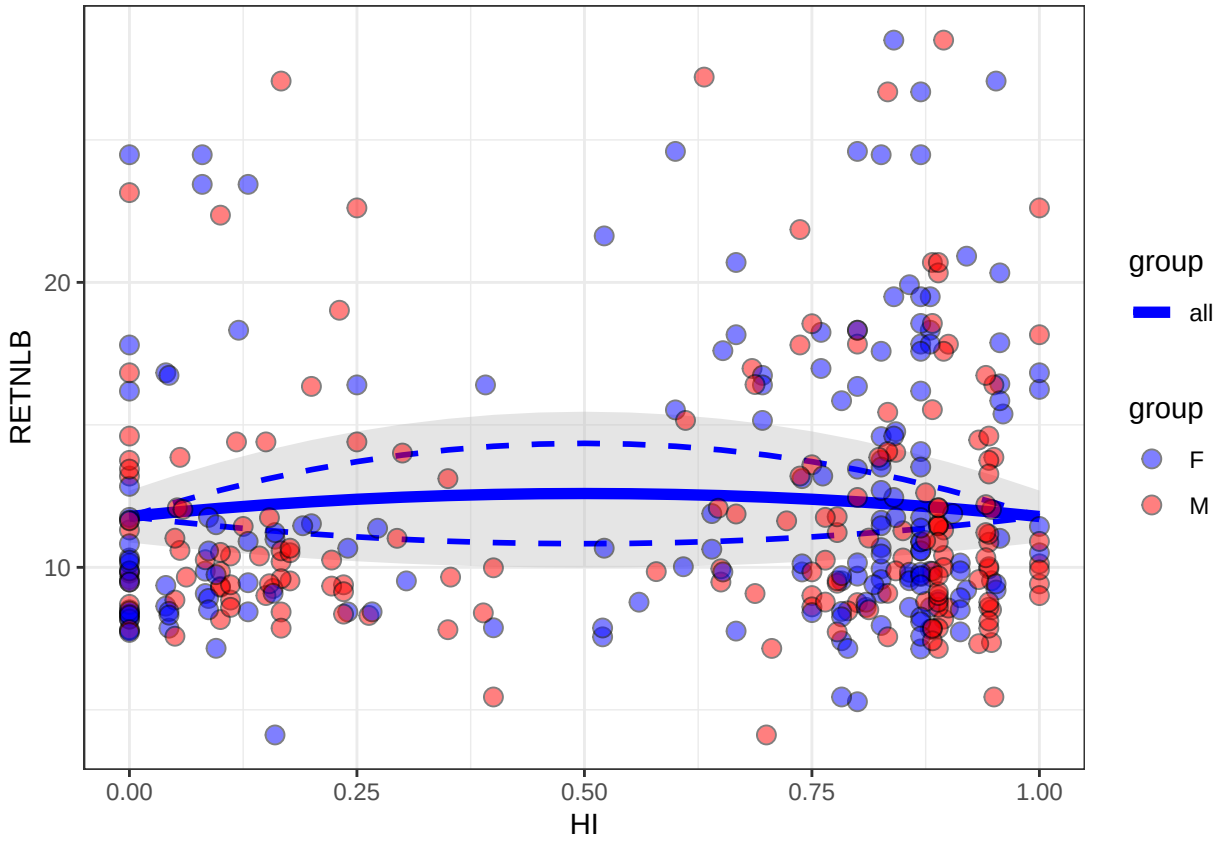
```
##
## Coefficients:
##         L1      alpha    myshape
## 11.7834882 -0.1369646  2.7148447
##
## Log-likelihood: -1043.42
## Best method: bobyqa
##
## $H1
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 11.18962827 12.46583439 -0.07890748  2.72667918
##
## Log-likelihood: -1041.54
## Best method: bobyqa
##
## $H2
## $H2$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1      alpha    myshape
## 12.1403690 -0.1339339  2.6732491
##
## Log-likelihood: -529.27
## Best method: bobyqa
##
## $H2$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
```

```
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1      alpha    myshape
## 11.4254856 -0.1394769  2.7794597
##
## Log-likelihood: -512.48
## Best method: bobyqa
##
##
## $H3
## $H3$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##           L1           L2        alpha      myshape
## 11.314885820 13.421334337  0.009858716  2.698337426
##
## Log-likelihood: -527.14
## Best method: bobyqa
##
## $H3$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1         L2      alpha    myshape
## 11.1214734 11.6747011 -0.1312826  2.7826987
##
## Log-likelihood: -512.27
## Best method: bobyqa
bananaPlot(mod = IFNy$H0,
           data = field,
           response = "RETNLB",
           group = "Sex") +
```

```
  scale_fill_manual(values = c("blue", "red")) +
  scale_color_manual(values = c("blue", "red")) +
  theme_bw()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```




```
field$Sex <- as.factor(field$Sex)

speparam <- parasiteLoad::getParamBounds("weibull", data = field, response = "SOCS1")

IFNy <- parasiteLoad::analyse(data = field,
                          response = "SOCS1",
                          model = "weibull",
                          group = "Sex")
```

```
## [1] "Analysing data for response: SOCS1"
## [1] "Fit for the response: SOCS1"
## [1] "Fitting for all"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
```

```
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Fitting for groupA : F"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Fitting for groupB : M"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
## [1] "Did converge"
## [1] "Testing H0 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.06   1 0.7355477
## [1] "Testing H1 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.03   1 0.8178395
## [1] "Testing H2 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.53   1 0.3038943
## [1] "Testing H2 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.44   1 0.3461543
## [1] "Testing H3 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.19   1 0.5349294
## [1] "Testing H3 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.47   1 0.3327884
## [1] "Testing H1 vs H0"
##    dLL dDF    pvalue
## 1 0.14   1 0.6027723
## [1] "Testing H2 vs H0"
##    dLL dDF     pvalue
## 1 4.28   3 0.03584161
## [1] "Testing H3 vs H1"
##    dLL dDF     pvalue
## 1 5.61   4 0.02414239
## [1] "Testing H3 vs H2"
##    dLL dDF    pvalue
## 1 1.47   2 0.2296914
```

```
##All
print(IFNy)
```

```
## $H0
```

```
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 12.39254617  0.03205676  3.59199127
##
## Log-likelihood: -950.39
## Best method: bobyqa
##
## $H1
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 12.52532286 12.25554898  0.02234286  3.59797296
##
## Log-likelihood: -950.26
## Best method: bobyqa
##
## $H2
## $H2$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1      alpha    myshape
## 12.9961677  0.1351202  3.4796588
##
## Log-likelihood: -484.99
## Best method: bobyqa
```

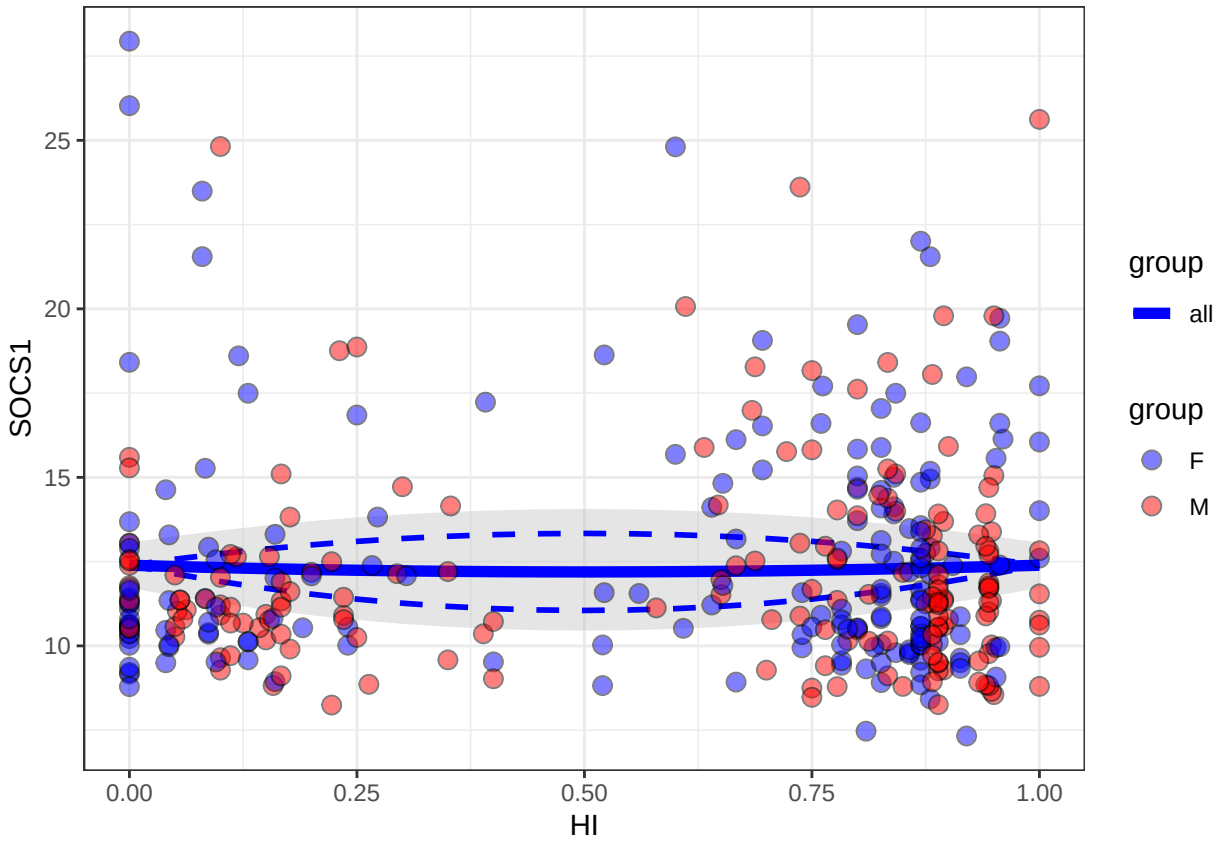
```
##
## $H2$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1      alpha    myshape
## 11.6652646 -0.1295653  3.7937587
##
## Log-likelihood: -461.13
## Best method: L-BFGS-B
##
##
## $H3
## $H3$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 13.40435891 12.47152011  0.08670974  3.51490342
##
## Log-likelihood: -484.34
## Best method: bobyqa
##
## $H3$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1         L2      alpha    myshape
```

```
## 11.1800494 11.9999689 -0.1323274  3.8143923
##
## Log-likelihood: -460.31
## Best method: bobyqa
```

```
bananaPlot(mod = IFNy$H0,
             data = field,
             response = "SOCS1",
             group = "Sex") +
  scale_fill_manual(values = c("blue", "red")) +
  scale_color_manual(values = c("blue", "red")) +
  theme_bw()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```




```
field$Sex <- as.factor(field$Sex)

speparam <- parasiteLoad::getParamBounds("weibull", data = field, response = "TICAM1")

IFNy <- parasiteLoad::analyse(data = field,
                          response = "TICAM1",
                          model = "weibull",
                          group = "Sex")
```

```
## [1] "Analysing data for response: TICAM1"
```

```
## [1] "Fit for the response: TICAM1"
## [1] "Fitting for all"
## [1] "Fitting model basic without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model basic with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting for groupA : F"
## [1] "Fitting model basic without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model basic with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting for groupB : M"
## [1] "Fitting model basic without alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model basic with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Testing H0 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.05   1 0.7430379
## [1] "Testing H1 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.08   1 0.6914911
## [1] "Testing H2 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.11   1 0.6341039
## [1] "Testing H2 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.41   1 0.3660989
## [1] "Testing H3 groupA no alpha vs alpha"
##   dLL dDF    pvalue
## 1   0   1 0.9863344
## [1] "Testing H3 groupB no alpha vs alpha"
##   dLL dDF    pvalue
## 1 0.4   1 0.3723029
## [1] "Testing H1 vs H0"
##   dLL dDF    pvalue
## 1 0.1   1 0.6571083
## [1] "Testing H2 vs H0"
##    dLL dDF   pvalue
## 1 0.47   3 0.814655
## [1] "Testing H3 vs H1"
##    dLL dDF    pvalue
## 1 1.78   4 0.4697213
## [1] "Testing H3 vs H2"
##   dLL dDF    pvalue
## 1 1.4   2 0.2458067
```

```
##All
print(IFNy)
```

```
## $H0
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 21.10711935  0.02683894  5.00000000
##
## Log-likelihood: -967.51
## Best method: bobyqa
##
## $H1
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 20.97193522 21.25222238  0.03280039  5.00000000
##
## Log-likelihood: -967.41
## Best method: bobyqa
##
## $H2
## $H2$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
```

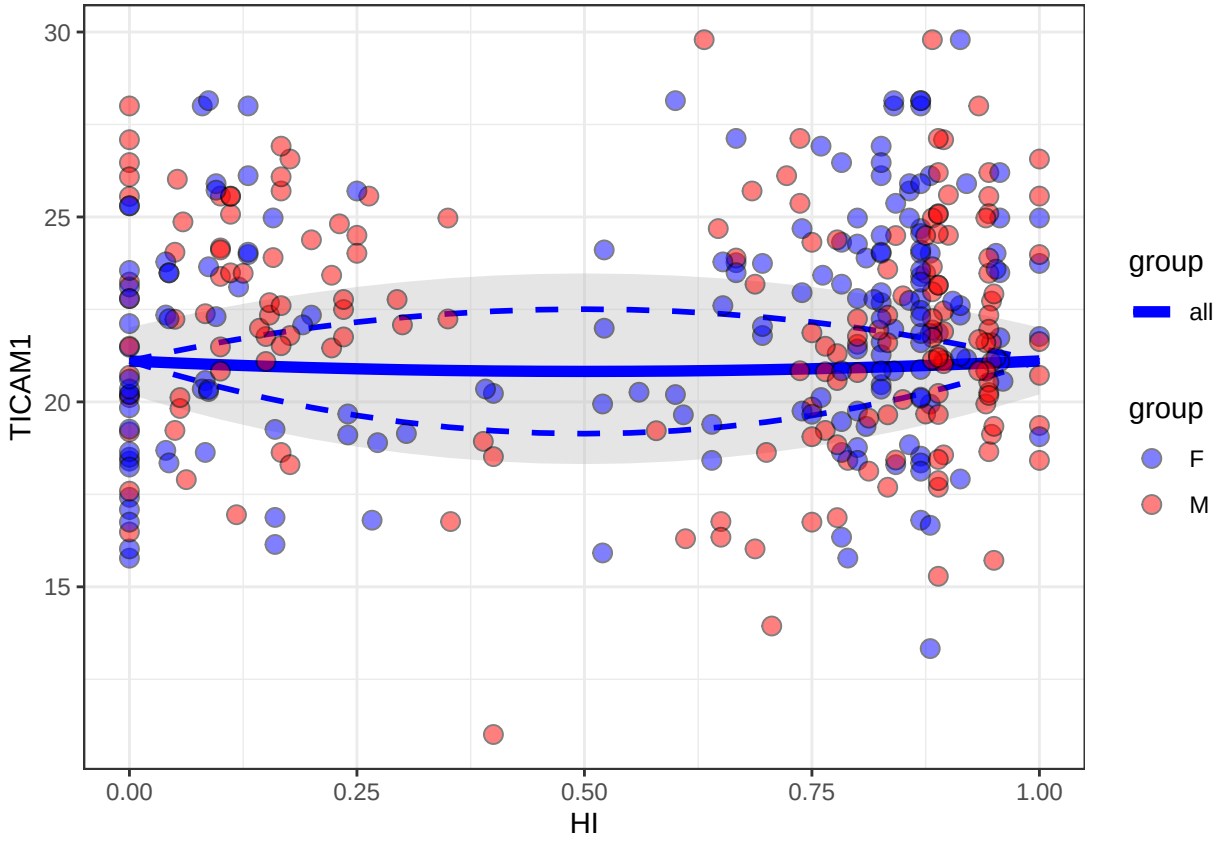
```
## 20.72060781 -0.05820441  5.00000000
##
## Log-likelihood: -482.78
## Best method: bobyqa
##
## $H2$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 21.43487691  0.09979881  5.00000000
##
## Log-likelihood: -484.26
## Best method: bobyqa
##
##
## $H3
## $H3$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##           L1           L2        alpha      myshape
## 20.222381184 21.547092478  0.002153487  5.000000000
##
## Log-likelihood: -481.77
## Best method: bobyqa
##
## $H3$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
```

```
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 21.88671743 21.09883081  0.09927843  5.00000000
##
## Log-likelihood: -483.87
## Best method: bobyqa
```

```
bananaPlot(mod = IFNy$H0,
             data = field,
             response = "TICAM1",
             group = "Sex") +
  scale_fill_manual(values = c("blue", "red")) +
  scale_color_manual(values = c("blue", "red")) +
  theme_bw()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```




```
field$Sex <- as.factor(field$Sex)

speparam <- parasiteLoad::getParamBounds("weibull", data = field, response = "TNF")

IFNy <- parasiteLoad::analyse(data = field,
                          response = "TNF",
```

```
                        model = "weibull",
                        group = "Sex")
```

```
## [1] "Analysing data for response: TNF"
## [1] "Fit for the response: TNF"
## [1] "Fitting for all"
## [1] "Fitting model basic without alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting for groupA : F"
## [1] "Fitting model basic without alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting model advanced with alpha"
```

```
## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable
```

```
## [1] "Did converge"
## [1] "Fitting for groupB : M"
## [1] "Fitting model basic without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model basic with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced without alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Fitting model advanced with alpha"

## Warning in bbmle::mle2(response ~ dweibull(shape = myshape, scale =
## MeanLoad(L1, : some parameters are on the boundary: variance-covariance
## calculations based on Hessian may be unreliable

## [1] "Did converge"
## [1] "Testing H0 no alpha vs alpha"
##   dLL dDF    pvalue
## 1 0.3   1 0.4409806
## [1] "Testing H1 no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.28   1 0.4520113
## [1] "Testing H2 groupA no alpha vs alpha"
##   dLL dDF    pvalue
## 1   0   1 0.9679167
## [1] "Testing H2 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.66   1 0.2512043
## [1] "Testing H3 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.02   1 0.8275115
## [1] "Testing H3 groupB no alpha vs alpha"
##    dLL dDF   pvalue
## 1 0.63   1 0.262373
## [1] "Testing H1 vs H0"
##   dLL dDF    pvalue
## 1   0   1 0.9762002
## [1] "Testing H2 vs H0"
##    dLL dDF    pvalue
## 1 0.44   3 0.8286785
## [1] "Testing H3 vs H1"
##    dLL dDF    pvalue
## 1 1.17   4 0.6743423
## [1] "Testing H3 vs H2"
```

```
##    dLL dDF    pvalue
## 1 0.72   2 0.4843956
```

```
##All
print(IFNy)
```

```
## $H0
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1       alpha     myshape
## 21.17789958  0.06016675  5.00000000
##
## Log-likelihood: -957.93
## Best method: bobyqa
##
## $H1
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 21.18688044 21.16797844  0.05975504  5.00000000
##
## Log-likelihood: -957.93
## Best method: bobyqa
##
## $H2
## $H2$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
```

```
## Coefficients:
##          L1       alpha     myshape
## 20.95671905 -0.00457202  5.00000000
##
## Log-likelihood: -475.34
## Best method: bobyqa
##
## $H2$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], alpha = paramBounds[["alphaLB"]],
##         myshape = paramBounds[["myshapeLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1      alpha    myshape
## 21.3862967  0.1233982  5.0000000
##
## Log-likelihood: -482.15
## Best method: bobyqa
##
##
## $H3
## $H3$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##          L1          L2       alpha     myshape
## 20.68615061 21.39990594  0.02591594  5.00000000
##
## Log-likelihood: -475.05
## Best method: bobyqa
##
## $H3$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L2, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$method, optimizer = config$optimizer,
##     data = data, lower = c(L1 = paramBounds[["L1LB"]], L2 = paramBounds[["L2LB"]],
##         alpha = paramBounds[["alphaLB"]], myshape = paramBounds[["myshapeLB"]]),
```

```
##     upper = c(L1 = paramBounds[["L1UB"]], L2 = paramBounds[["L2UB"]],
##         alpha = paramBounds[["alphaUB"]], myshape = paramBounds[["myshapeUB"]]),
##     control = config$control)
##
## Coefficients:
##         L1         L2      alpha    myshape
## 21.8558445 21.0294753  0.1220749  5.0000000
##
## Log-likelihood: -481.72
## Best method: bobyqa
```

```
bananaPlot(mod = IFNy$H0,
             data = field,
             response = "TNF",
             group = "Sex") +
  scale_fill_manual(values = c("blue", "red")) +
  scale_color_manual(values = c("blue", "red")) +
  theme_bw()
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```