

10. Applying random forest on field data - gene

Fay

2022-11-04

Aim:

- Applying the models established in the script: 9
- How are hybrid mice different to the parental species?

Load necessary libraries:

```
#install.packages("optimx", version = "2021-10.12") # this package is required for  
#the parasite load package to work  
library(tidyverse)  
library(tidyr)  
library(dplyr)  
library(cowplot)  
library(randomForest)  
library(ggplot2)  
library(VIM) # visualizing missing data  
library(mice) # imputing missing data without predictors  
library(ggpubr)  
library(optimx)  
library(rfUtilities) # Implements a permutation test cross-validation for  
# Random Forests models  
library(mice) #imputations  
library(fitdistrplus) #testing distributions  
library(logspline)  
library(caret)
```

Field data

Import field data

```
hm <- read.csv("output_data/imputed_mice.csv")
```

Clean data

```
Field <- hm %>%  
  filter(origin == "Field") %>%  
  drop_na(HI)
```

We have 1921 mice in total.

```
EqPCR.cols      <- c("delta_ct_cewe_MminusE", "MC.Eimeria", "Ct.Eimeria") #, "Ct.Mus" "delta_ct_ilwe_Mmin
Genes_wild      <- c("IFNy", "CXCR3", "IL.6", "IL.13", "IL.10",
                    "IL1RN", "CASP1", "CXCL9", "IDO1", "IRGM1", "MPO",
                    "MUC2", "MUC5AC", "MYD88", "NCR1", "PRF1", "RETNLB", "SOCS1",
                    "TICAM1", "TNF") #, "IL.12", "IRG6")
```

Prepare vectors for selecting

Actual Cleaning

```
#select the imputed gene columns
gene <- Field %>%
  dplyr::select(c(Mouse_ID, "IFNy", "CXCR3", "IL.6", "IL.13", "IL.10",
                  "IL1RN", "CASP1", "CXCL9", "IDO1", "IRGM1", "MPO",
                  "MUC2", "MUC5AC", "MYD88", "NCR1", "PRF1", "RETNLB", "SOCS1",
                  "TICAM1", "TNF"))

genes <- gene %>%
  dplyr::select(-Mouse_ID)

#remove rows with only nas
genes <- genes[,colSums(is.na(genes))<nrow(genes)]

#remove columns with only nas
genes <- genes[rowSums(is.na(genes)) != ncol(genes), ]

# select the same rows from the gene data
gene <- gene[row.names(genes),]

# select the same rows from the field data
Field <- Field[row.names(genes),]
```

Predicting weight loss in our imputed field data

Start by making the predictions for the field data.

```
# load predicting weight loss model
weight_loss_predict <- readRDS("r_scripts/models/predict_WL.rds")

set.seed(540)

#The predict() function in R is used to predict the values based on the input data.
predictions_field <- predict(weight_loss_predict, genes)

#make the vector positive so that the distributions further down work
predictions_field <- predictions_field * (-1)

# assign test.data to a new object, so that we can make changes
result_field <- genes
```

```
#add the new variable of predictions to the result object
result_field <- cbind(result_field, predictions_field)

# add it to the field data
Field <- cbind(Field, predictions_field)
```

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.

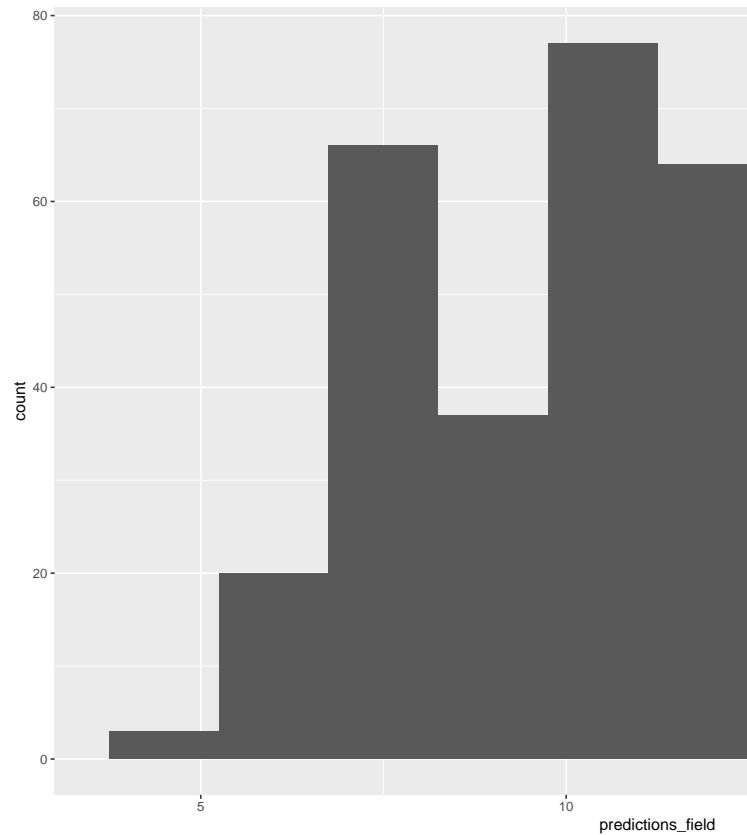
Install the package

```
##
## * checking for file '/tmp/RtmpIMuyF8/remotes115216751a8e33/alicebalard-parasiteLoad-1b43216/DESCRIPTION' ... OK
## * 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'
```

Data diagnostics

Visualizations

```
Field %>% ggplot(aes(x = predictions_field)) +
  geom_histogram(binwidth = 1.5)
```



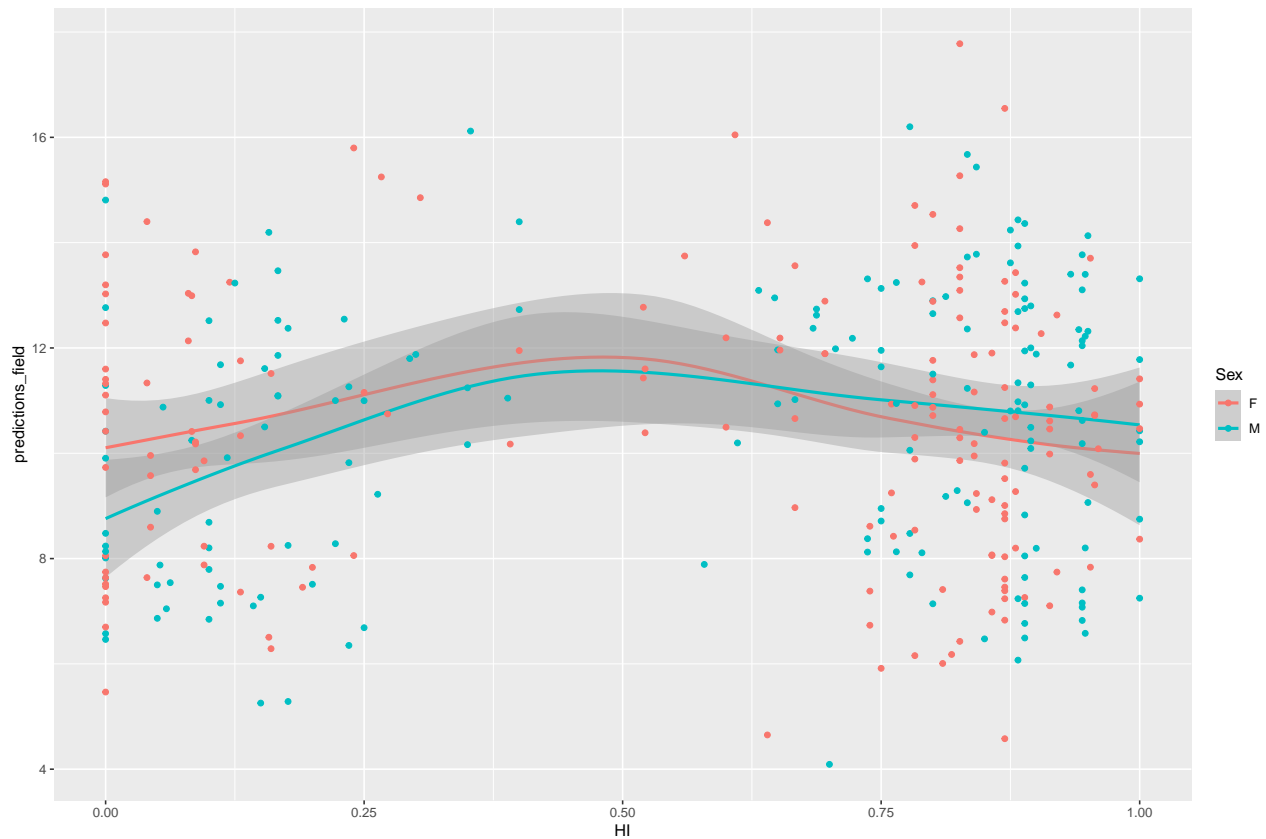
What is the distribution of the predicted weight loss?

Rough graph of our predictions against the hybrid index and against the

```
Field %>%
  ggplot(aes(x = HI , y = predictions_field , color = Sex)) +
  geom_smooth() +
  geom_point()
```

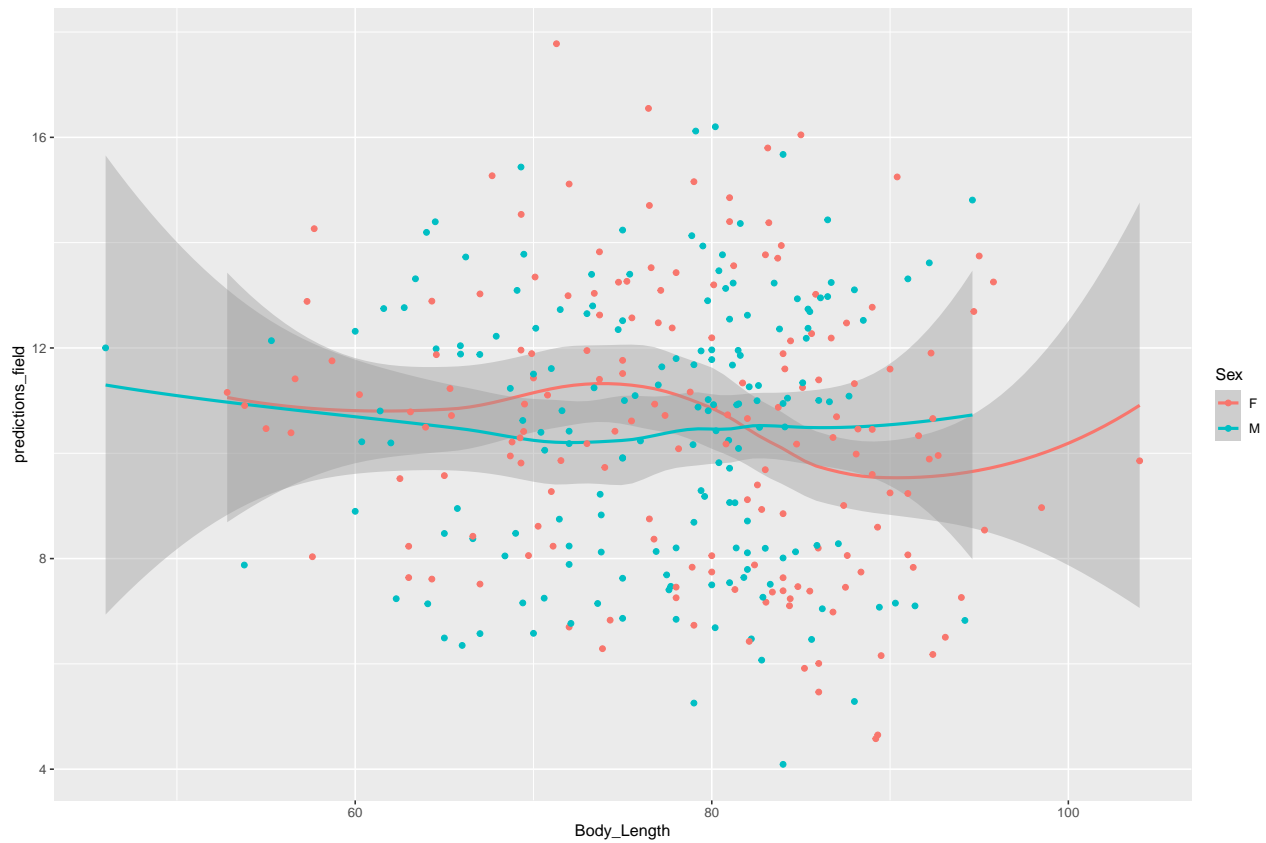
body length

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



```
Field %>%
  ggplot(aes(x = Body_Length , y = predictions_field , color = Sex)) +
  geom_smooth() +
  geom_point()
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
## Warning: Removed 1 rows containing non-finite values (`stat_smooth()`).
## Warning: Removed 1 rows containing missing values (`geom_point()`).
```



Fitting distributions??

Ratios / Percentages are not normally distributed. Weibull is a good distributions.

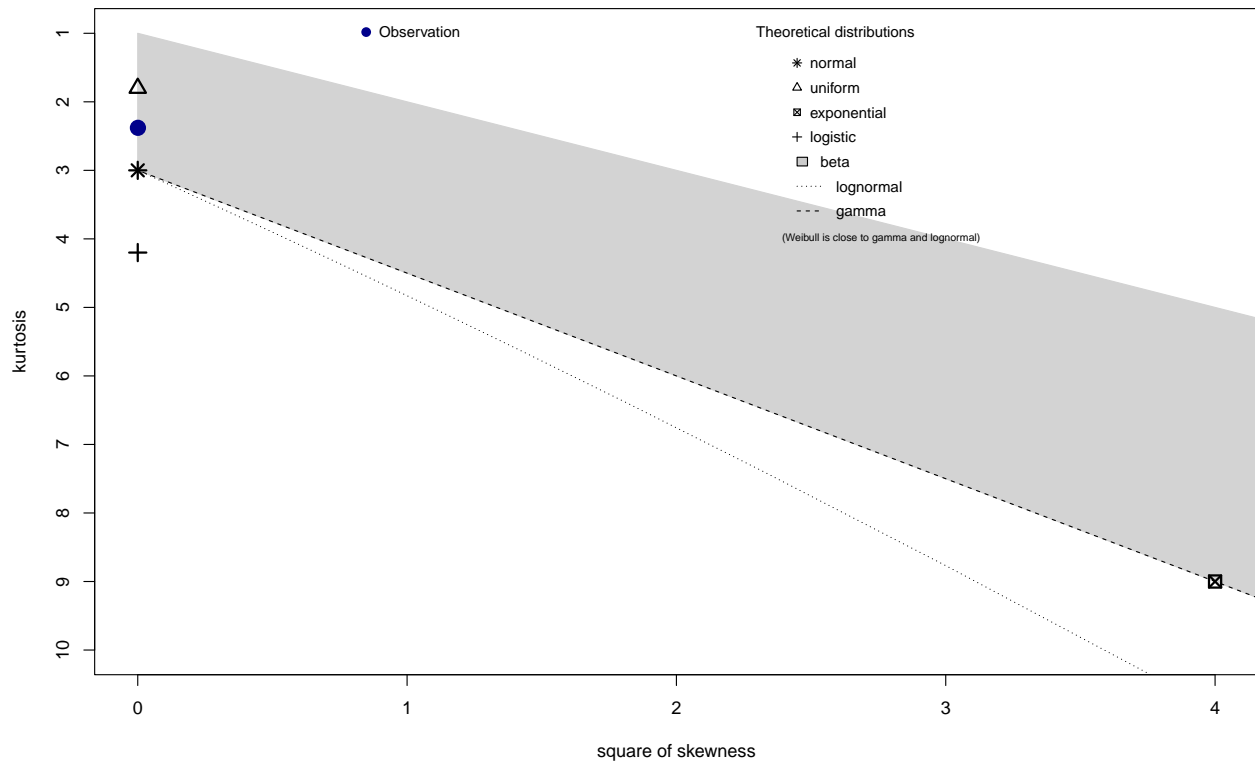
Alice used weibull for the qpcr data. (paper)

```
Field <- Field %>%
  dplyr::mutate(WL = predictions_field)

x <- Field$WL

descdist(data = x, discrete = FALSE)
```

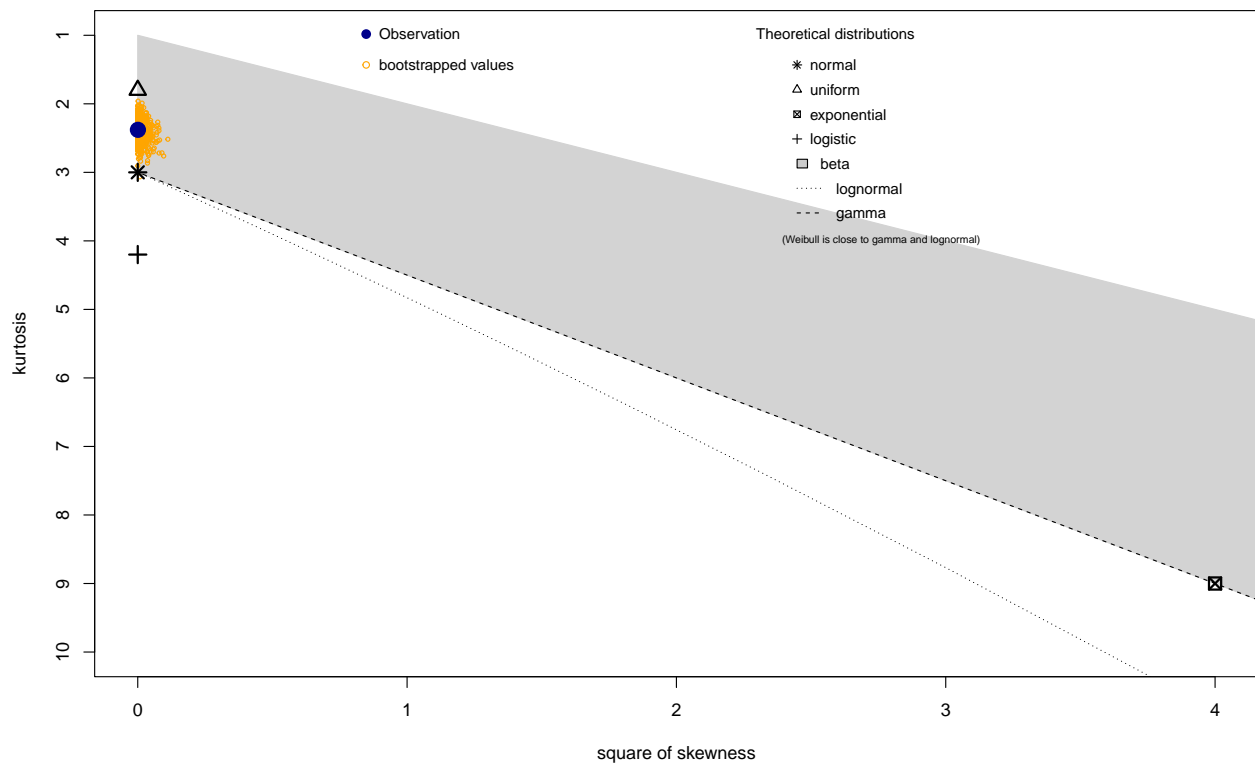
Cullen and Frey graph



```
## summary statistics
## -----
## min: 4.091834 max: 17.77793
## median: 10.65928
## mean: 10.4532
## estimated sd: 2.573586
## estimated skewness: 0.02495982
## estimated kurtosis: 2.379431
```

```
descdist(data = x, discrete = FALSE, #data is continuous
          boot = 1000)
```

Cullen and Frey graph



```
## summary statistics
## -----
## min: 4.091834 max: 17.77793
## median: 10.65928
## mean: 10.4532
## estimated sd: 2.573586
## estimated skewness: 0.02495982
## estimated kurtosis: 2.379431
```

Test for binomial distribution

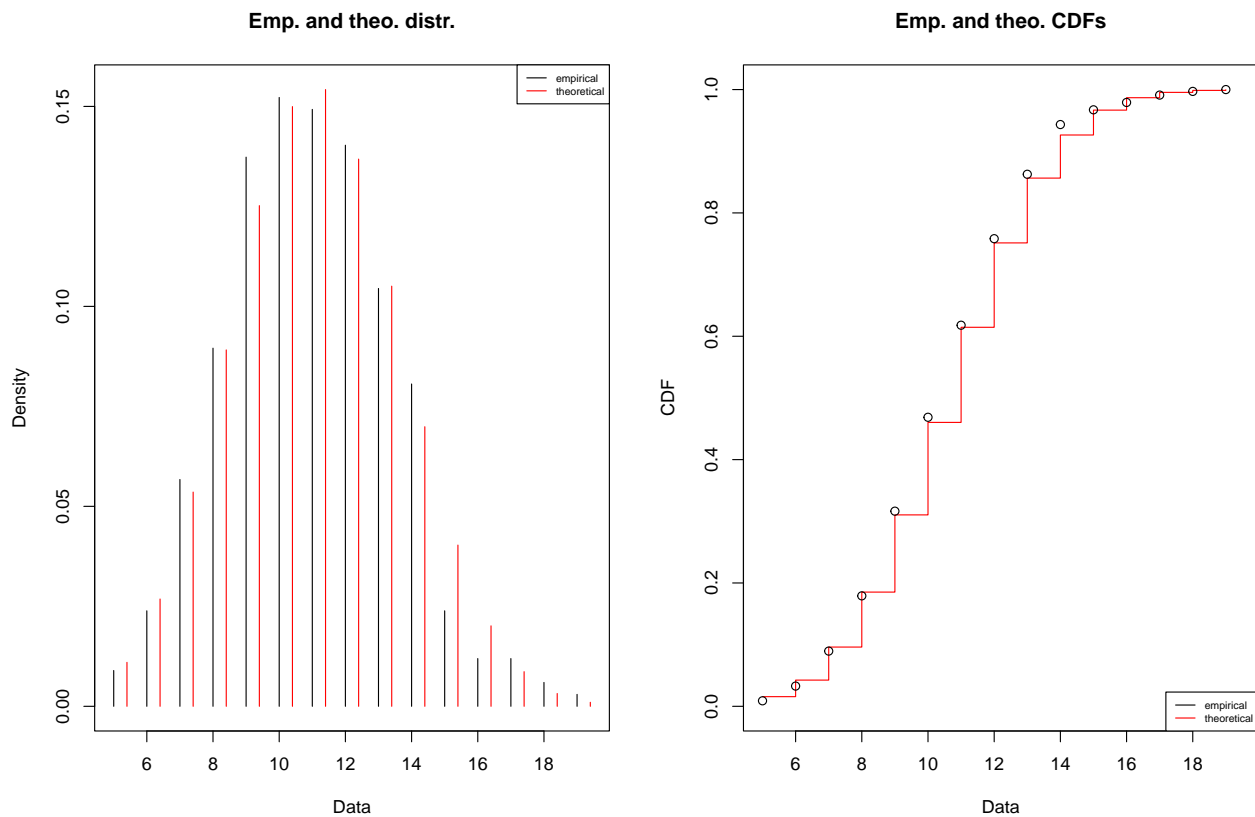
```
set.seed(10)
n = 25
size = 27
prob = .4
data = rbinom(x, size = size, prob = prob)
fit = fitdist(data = data, dist="binom",
              fix.arg=list(size = size),
              start=list(prob = 0.1))

summary(fit)
```

```
## Fitting of the distribution ' binom ' by maximum likelihood
## Parameters :
##      estimate Std. Error
## prob 0.399558 0.005150141
## Fixed parameters:
##      value
```



```
## size      27
## Loglikelihood: -779.317   AIC: 1560.634   BIC: 1564.448
plot(fit)
```



```
normal_ <- fitdist(x, "norm")
weibull_ <- fitdist(x, "weibull")
gamma_ <- fitdist(x, "gamma")

# 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)
  listDist <- 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=distrib2)
cdfcomp(listDistr, legendtext=distrib2)
qqcomp(listDistr, legendtext=distrib2)
ppcomp(listDistr, legendtext=distrib2)
par(mfrow=c(1,1))
}

```

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

Functions for testing distributions

```

## $fit
##      mean      sd
## 10.45320110  2.56974179
## ( 0.14039999) ( 0.09927779)
##
## $loglik
## [1] -791.5192
##
## $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")
```

```

## $fit
##      shape      scale
## 4.5546414 11.4515521
## ( 0.1927141) ( 0.1450042)

```

```

##
## $loglik
## [1] -791.1953
##
## $AIC
## NULL

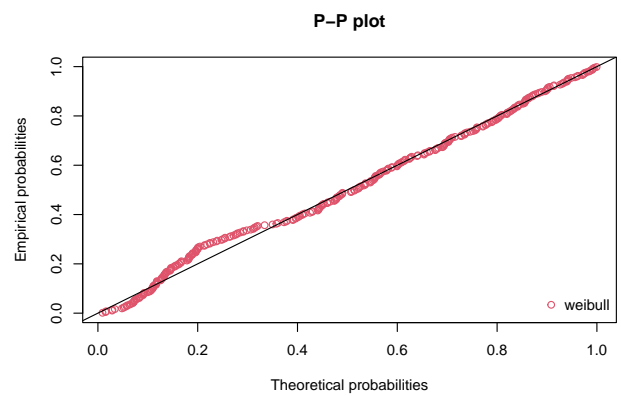
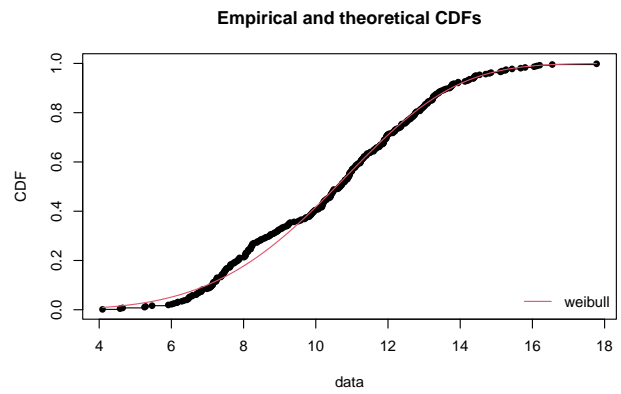
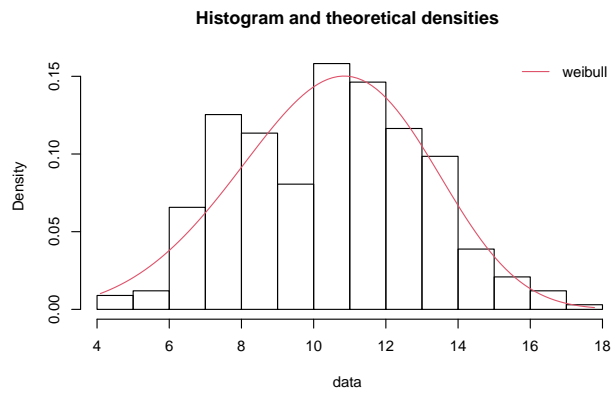
tryDistrib(x, "weibullshifted")

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

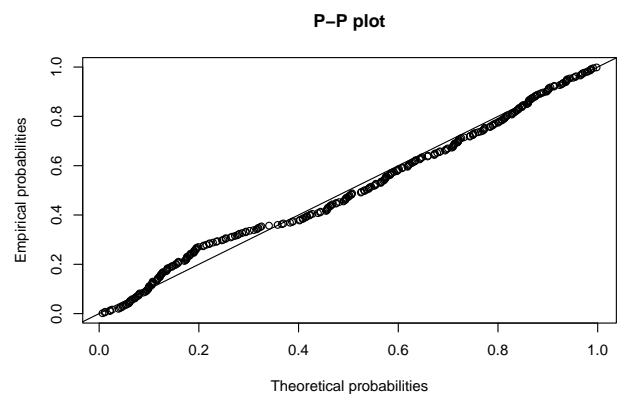
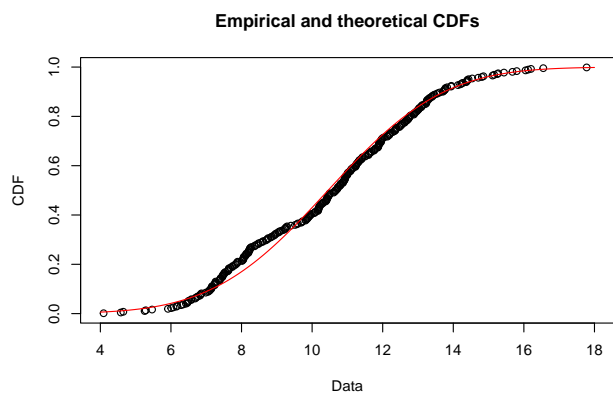
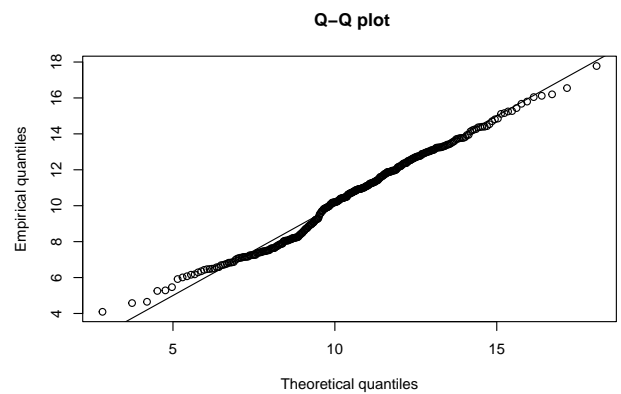
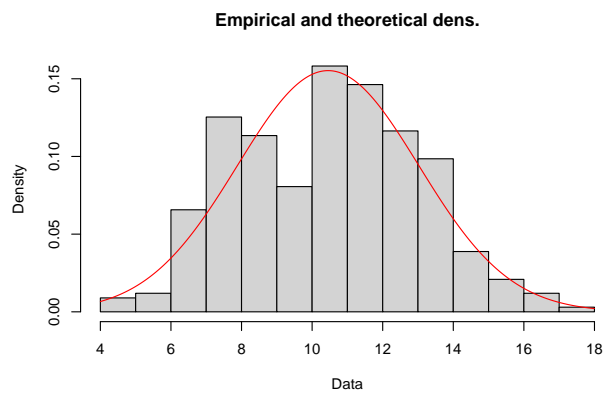
findGoodDist(x, "normal", "weibull")

## $normal
## $normal$fit
##      mean      sd
## 10.45320110 2.56974179
## ( 0.14039999) ( 0.09927779)
##
## $normal$loglik
## [1] -791.5192
##
## $normal$AIC
## NULL

```



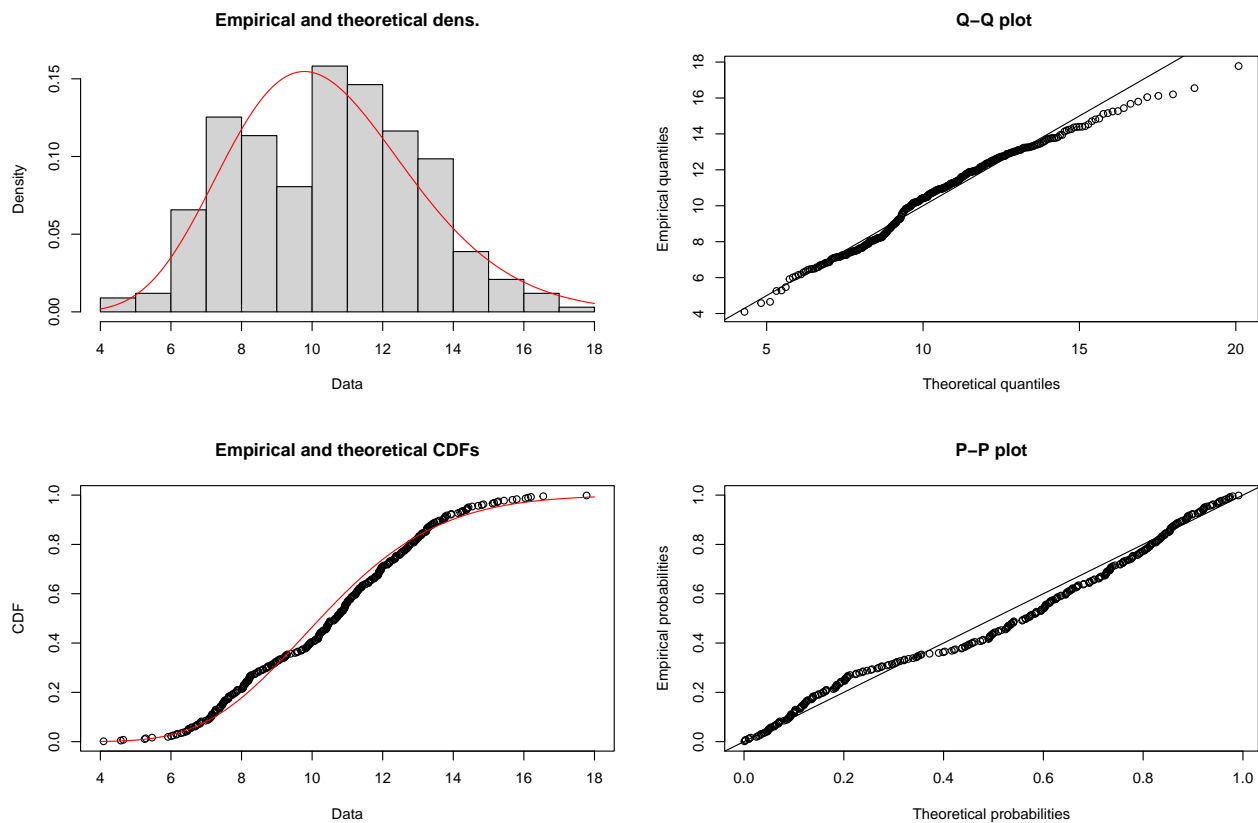
`plot(normal_)`



```
summary(normal_)
```

```
## Fitting of the distribution ' norm ' by maximum likelihood
## Parameters :
##      estimate Std. Error
## mean 10.453201 0.14039999
## sd    2.569742 0.09927772
## Loglikelihood: -791.5192   AIC: 1587.038   BIC: 1594.667
## Correlation matrix:
##      mean sd
## mean  1  0
## sd    0  1
```

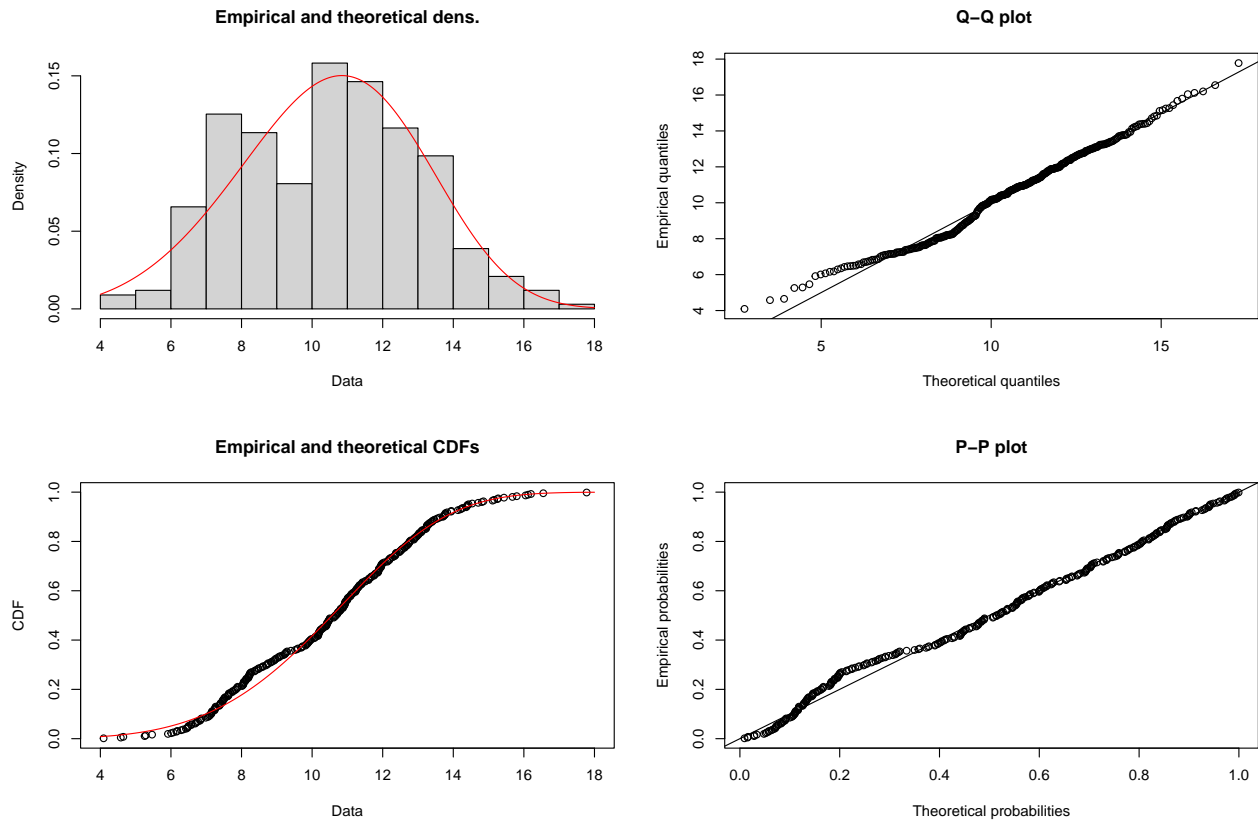
```
plot(gamma_)
```



```
summary(gamma_)
```

```
## Fitting of the distribution ' gamma ' by maximum likelihood
## Parameters :
##      estimate Std. Error
## shape 15.54936  1.1887735
## rate   1.48756  0.1155788
## Loglikelihood: -794.6766   AIC: 1593.353   BIC: 1600.981
## Correlation matrix:
##      shape      rate
## shape 1.0000000 0.9839713
## rate  0.9839713 1.0000000
```

```
plot(weibull_)
```



```
summary(weibull_)
```

```
## Fitting of the distribution ' weibull ' by maximum likelihood
## Parameters :
##      estimate Std. Error
## shape  4.554227  0.1927003
## scale  11.451692  0.1450187
## Loglikelihood: -791.1953   AIC:  1586.391   BIC:  1594.019
## Correlation matrix:
##      shape  scale
## shape 1.00000 0.32005
## scale 0.32005 1.00000
```

Is alpha significant for each hypothesis?

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

```
parasiteLoad::getParamBounds("normal", data = Field, response = "WL")
```

```
##      L1start      L1LB      L1UB      L2start      L2LB      L2UB
## 10.453201105  4.091833908 17.777929279 10.453201105  4.091833908 17.777929279
##      alphaStart      alphaLB      alphaUB      mysdStart      mysdLB      mysdUB
## 0.000000000 -5.000000000  5.000000000  1.000000000  0.000000001 10.000000000
```

```

speparam <- c(L1start = 10,
              L1LB = 1e-9,
              L1UB = 20,
              L2start = 10,
              L2LB = 1e-9,
              L2UB = 20,
              alphaStart = 0, alphaLB = -5, alphaUB = 5,
              myshapeStart = 1, myshapeLB = 1e-9, myshapeUB = 5)

```

```
##All
```

```

fitWL_Sex <- parasiteLoad::analyse(data = Field,
                                   response = "WL",
                                   model = "normal",
                                   group = "Sex")

```

```

## [1] "Analysing data for response: WL"
## [1] "Fit for the response: WL"
## [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 3.78   1 0.005963574
## [1] "Testing H1 no alpha vs alpha"
##      dLL dDF      pvalue
## 1 2.95   1 0.0151356
## [1] "Testing H2 groupA no alpha vs alpha"
##      dLL dDF      pvalue
## 1 1.37   1 0.09795211
## [1] "Testing H2 groupB no alpha vs alpha"

```

```
##      dLL dDF      pvalue
## 1 2.56    1 0.02360156
## [1] "Testing H3 groupA no alpha vs alpha"
##      dLL dDF      pvalue
## 1 1.61    1 0.07241473
## [1] "Testing H3 groupB no alpha vs alpha"
##      dLL dDF      pvalue
## 1 2.48    1 0.02590116
## [1] "Testing H1 vs H0"
##      dLL dDF      pvalue
## 1 0.95    1 0.168589
## [1] "Testing H2 vs H0"
##      dLL dDF      pvalue
## 1 0.19    3 0.9426523
## [1] "Testing H3 vs H1"
##      dLL dDF      pvalue
## 1 3.28    4 0.1608561
## [1] "Testing H3 vs H2"
##      dLL dDF      pvalue
## 1 4.04    2 0.01768312
```

```
parasiteLoad::analyse(data = Field,
                      response = "WL",
                      model = "normal",
                      group = "Sex")
```

```
## [1] "Analysing data for response: WL"
## [1] "Fit for the response: WL"
## [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 3.78   1 0.005963574
## [1] "Testing H1 no alpha vs alpha"
##      dLL dDF      pvalue
## 1 2.95   1 0.0151356
## [1] "Testing H2 groupA no alpha vs alpha"
##      dLL dDF      pvalue
## 1 1.37   1 0.09795211
## [1] "Testing H2 groupB no alpha vs alpha"
##      dLL dDF      pvalue
## 1 2.56   1 0.02360156
## [1] "Testing H3 groupA no alpha vs alpha"
##      dLL dDF      pvalue
## 1 1.61   1 0.07241473
## [1] "Testing H3 groupB no alpha vs alpha"
##      dLL dDF      pvalue
## 1 2.48   1 0.02590116
## [1] "Testing H1 vs H0"
##      dLL dDF      pvalue
## 1 0.95   1 0.168589
## [1] "Testing H2 vs H0"
##      dLL dDF      pvalue
## 1 0.19   3 0.9426523
## [1] "Testing H3 vs H1"
##      dLL dDF      pvalue
## 1 3.28   4 0.1608561
## [1] "Testing H3 vs H2"
##      dLL dDF      pvalue
## 1 4.04   2 0.01768312

## $H0
##
## Call:
## bbmle::mle2(minuslogl = response ~ dnorm(mean = MeanLoad(L1,
##      L1, alpha, HI), sd = mysd), start = start, method = config$method,
##      optimizer = config$optimizer, data = data, lower = c(L1 = paramBounds[["L1LB"]],
##      mysd = paramBounds[["mysdLB"]], alpha = paramBounds[["alphaLB"]]),
##      upper = c(L1 = paramBounds[["L1UB"]], mysd = paramBounds[["mysdUB"]],
##      alpha = paramBounds[["alphaUB"]]), control = config$control)
##
## Coefficients:
##      L1      mysd      alpha
## 9.814002 2.540905 -0.281860
##
## Log-likelihood: -787.74
## Best method: bobyqa
##
## $H1
##
## Call:
## bbmle::mle2(minuslogl = response ~ dnorm(mean = MeanLoad(L1,
##      L2, alpha, HI), sd = mysd), start = start, method = config$method,
##      optimizer = config$optimizer, data = data, lower = c(L1 = paramBounds[["L1LB"]],

```

```

##      mysd = paramBounds[["mysdLB"]], L2 = paramBounds[["L2LB"]],
##      alpha = paramBounds[["alphaLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##      mysd = paramBounds[["mysdUB"]], L2 = paramBounds[["L2UB"]],
##      alpha = paramBounds[["alphaUB"]]), control = config$control)
##
## Coefficients:
##      L1      L2      alpha      mysd
##  9.5753580 10.0978117 -0.2526079  2.5337254
##
## Log-likelihood: -786.79
## Best method: bobyqa
##
## $H2
## $H2$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dnorm(mean = MeanLoad(L1,
##      L1, alpha, HI), sd = mysd), start = start, method = config$method,
##      optimizer = config$optimizer, data = data, lower = c(L1 = paramBounds[["L1LB"]],
##      mysd = paramBounds[["mysdLB"]], alpha = paramBounds[["alphaLB"]]),
##      upper = c(L1 = paramBounds[["L1UB"]], mysd = paramBounds[["mysdUB"]],
##      alpha = paramBounds[["alphaUB"]]), control = config$control)
##
## Coefficients:
##      L1      mysd      alpha
##  9.9286873  2.5773333 -0.2326945
##
## Log-likelihood: -395.07
## Best method: bobyqa
##
## $H2$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dnorm(mean = MeanLoad(L1,
##      L1, alpha, HI), sd = mysd), start = start, method = config$method,
##      optimizer = config$optimizer, data = data, lower = c(L1 = paramBounds[["L1LB"]],
##      mysd = paramBounds[["mysdLB"]], alpha = paramBounds[["alphaLB"]]),
##      upper = c(L1 = paramBounds[["L1UB"]], mysd = paramBounds[["mysdUB"]],
##      alpha = paramBounds[["alphaUB"]]), control = config$control)
##
## Coefficients:
##      L1      mysd      alpha
##  9.6835336  2.5023082 -0.3393935
##
## Log-likelihood: -392.47
## Best method: bobyqa
##
## $H3
## $H3$groupA
##
## Call:
## bbmle::mle2(minuslogl = response ~ dnorm(mean = MeanLoad(L1,
##      L2, alpha, HI), sd = mysd), start = start, method = config$method,

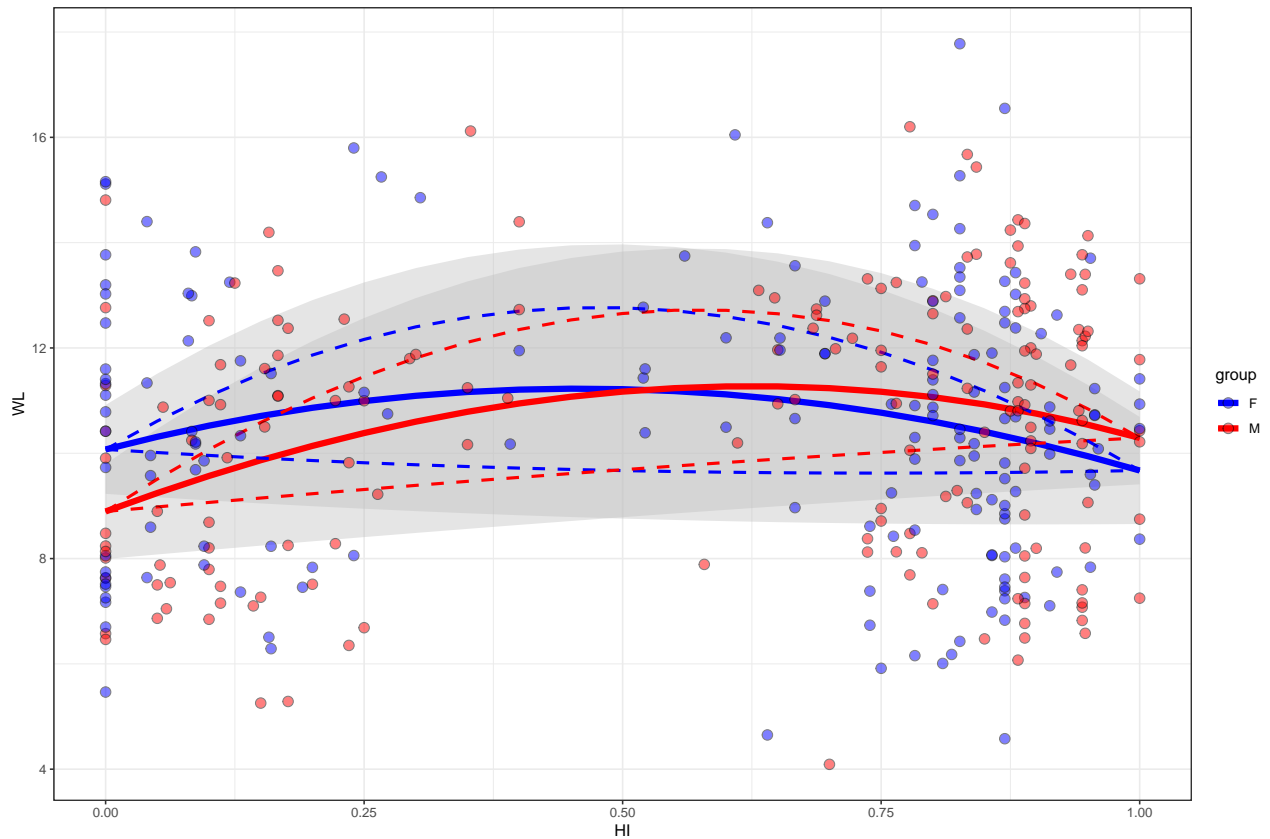
```

```

##      optimizer = config$optimizer, data = data, lower = c(L1 = paramBounds[["L1LB"]],
##      mysd = paramBounds[["mysdLB"]], L2 = paramBounds[["L2LB"]],
##      alpha = paramBounds[["alphaLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##      mysd = paramBounds[["mysdUB"]], L2 = paramBounds[["L2UB"]],
##      alpha = paramBounds[["alphaUB"]]), control = config$control)
##
## Coefficients:
##      L1      L2      alpha      mysd
## 10.0727199  9.6730112 -0.2723334  2.5735464
##
## Log-likelihood: -394.83
## Best method: bobyqa
##
## $H3$groupB
##
## Call:
## bbmle::mle2(minuslogl = response ~ dnorm(mean = MeanLoad(L1,
##      L2, alpha, HI), sd = mysd), start = start, method = config$method,
##      optimizer = config$optimizer, data = data, lower = c(L1 = paramBounds[["L1LB"]],
##      mysd = paramBounds[["mysdLB"]], L2 = paramBounds[["L2LB"]],
##      alpha = paramBounds[["alphaLB"]]), upper = c(L1 = paramBounds[["L1UB"]],
##      mysd = paramBounds[["mysdUB"]], L2 = paramBounds[["L2UB"]],
##      alpha = paramBounds[["alphaUB"]]), control = config$control)
##
## Coefficients:
##      L1      L2      alpha      mysd
##  8.8949407 10.2920779 -0.3288149  2.4464957
##
## Log-likelihood: -388.68
## Best method: bobyqa
plot_WL_Sex<- bananaPlot(mod = fitWL_Sex$H3,
      data = Field,
      response = "WL",
      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.
plot_WL_Sex

```



H0: the expected load for the subspecies and between 2 groups is the same

H1: the mean load across 2 groups is the same, but can differ across subspecies

H2: the mean load across subspecies is the same, but can differ between the 2 groups

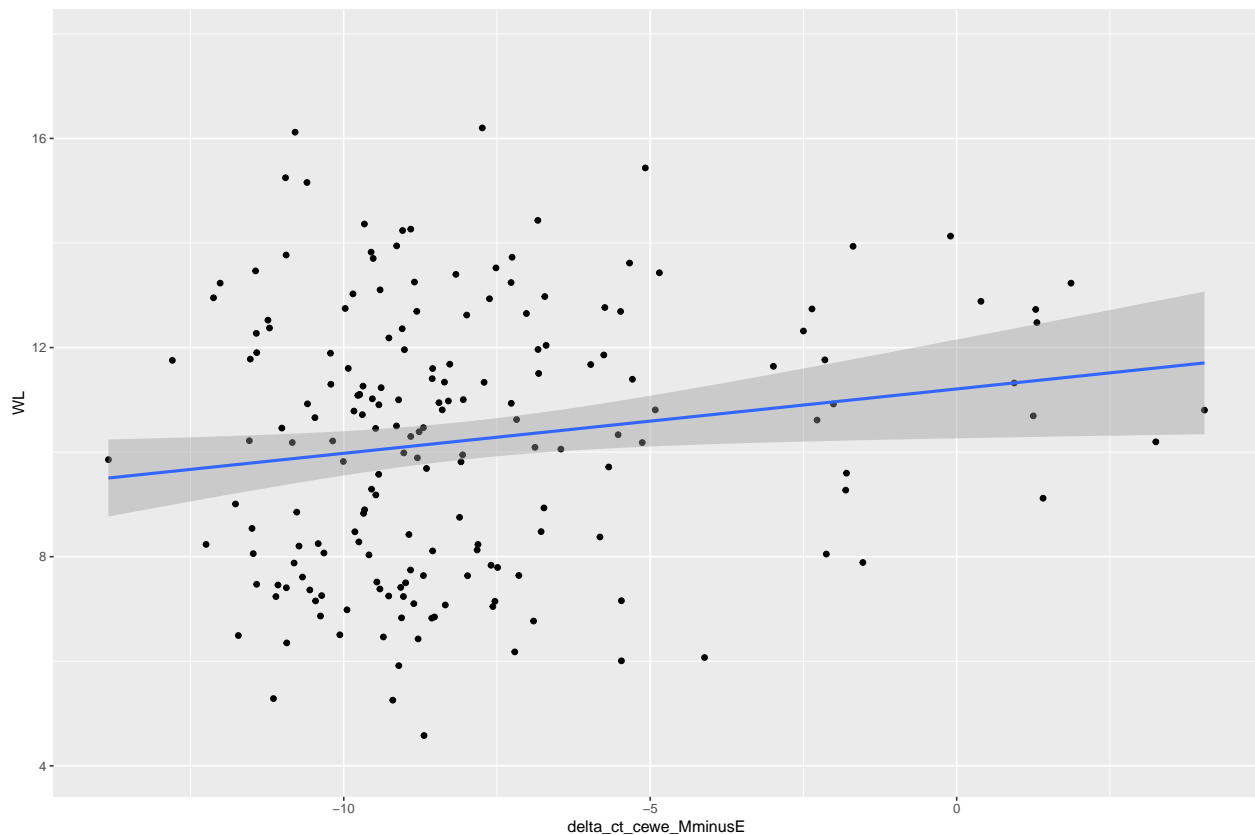
H3: the mean load can differ both across subspecies and between 2 groups

```
ggplot(data = Field, aes(x = delta_ct_cewe_MminusE, y = WL)) +
  geom_point() +
  stat_smooth(method= "lm")
```

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

```
## Warning: Removed 150 rows containing non-finite values (`stat_smooth()`).
```

```
## Warning: Removed 150 rows containing missing values (`geom_point()`).
```



```
Field2 <- Field %>%
  drop_na(delta_ct_cewe_MminusE)

cor(Field2$WL, Field2$delta_ct_cewe_MminusE)

## [1] 0.1617751

tolerance <- lm(WL ~ delta_ct_cewe_MminusE, data = Field)

summary(tolerance)

##
## Call:
## lm(formula = WL ~ delta_ct_cewe_MminusE, data = Field)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -5.5596 -2.1168  0.2586  1.7938  6.2395
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    11.20837    0.47860   23.419  <2e-16 ***
## delta_ct_cewe_MminusE  0.12310    0.05551    2.218  0.0278 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.486 on 183 degrees of freedom
```

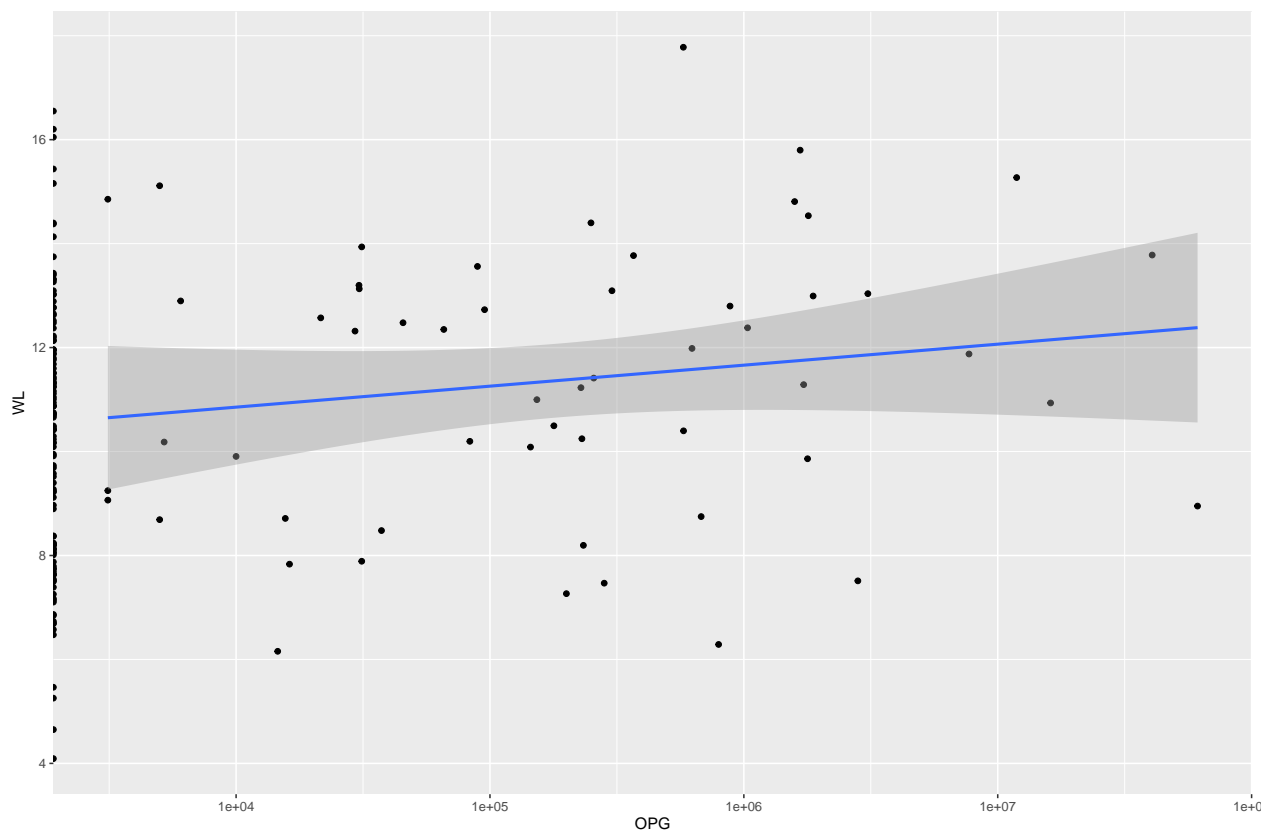
```
## (150 observations deleted due to missingness)
## Multiple R-squared:  0.02617,    Adjusted R-squared:  0.02085
## F-statistic: 4.918 on 1 and 183 DF,  p-value: 0.02781
```

```
confint(tolerance)
```

```
##                2.5 %    97.5 %
## (Intercept)      10.26408377 12.1526478
## delta_ct_cewe_MminusE 0.01358008 0.2326143
```

```
ggplot(data = Field, aes(x = OPG, y = WL)) +
  geom_point() +
  stat_smooth(method= "lm") +
  scale_x_log10()
```

```
## Warning: Transformation introduced infinite values in continuous x-axis
## Transformation introduced infinite values in continuous x-axis
## `geom_smooth()` using formula = 'y ~ x'
## Warning: Removed 280 rows containing non-finite values (`stat_smooth()`).
## Warning: Removed 157 rows containing missing values (`geom_point()`).
```



```
Field2 <- Field %>%
  drop_na(OPG)

cor(Field2$WL, Field2$OPG)
```

```
## [1] 0.04972871
```

```
tolerance <- lm(WL ~ OPG, data = Field)
```

```
summary(tolerance)
```

```
##
## Call:
## lm(formula = WL ~ OPG, data = Field)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -6.4814 -2.3628  0.1324  1.9028  7.1916
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.057e+01  1.986e-01  53.247  <2e-16 ***
## OPG          2.278e-08  3.449e-08   0.661    0.51
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.616 on 176 degrees of freedom
## (157 observations deleted due to missingness)
## Multiple R-squared:  0.002473, Adjusted R-squared: -0.003195
## F-statistic: 0.4363 on 1 and 176 DF, p-value: 0.5098
```

```
confint(tolerance)
```

```
##              2.5 %      97.5 %
## (Intercept) 1.018133e+01 1.096510e+01
## OPG          -4.527973e-08 9.083863e-08
```

```
tolerance <- lm(WL ~ OPG * delta_ct_cewe_MminusE, data = Field)
```

```
summary(tolerance)
```

```
##
## Call:
## lm(formula = WL ~ OPG * delta_ct_cewe_MminusE, data = Field)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -4.5013 -2.1298 -0.1818  1.6790  6.1191
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)      1.181e+01  8.669e-01  13.619  <2e-16 ***
## OPG              -9.065e-06  2.243e-05  -0.404  0.6881
## delta_ct_cewe_MminusE  2.227e-01  1.135e-01   1.962  0.0559 .
## OPG:delta_ct_cewe_MminusE  6.096e-07  6.573e-06   0.093  0.9265
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.469 on 45 degrees of freedom
## (286 observations deleted due to missingness)
```

```
## Multiple R-squared:  0.09644,    Adjusted R-squared:  0.0362
## F-statistic: 1.601 on 3 and 45 DF,  p-value: 0.2024
```

```
confint(tolerance)
```

```
##                2.5 %      97.5 %
## (Intercept)      1.005962e+01 1.355155e+01
## OPG              -5.424724e-05 3.611707e-05
## delta_ct_cewe_MminusE -5.889685e-03 4.513038e-01
## OPG:delta_ct_cewe_MminusE -1.262838e-05 1.384754e-05
```

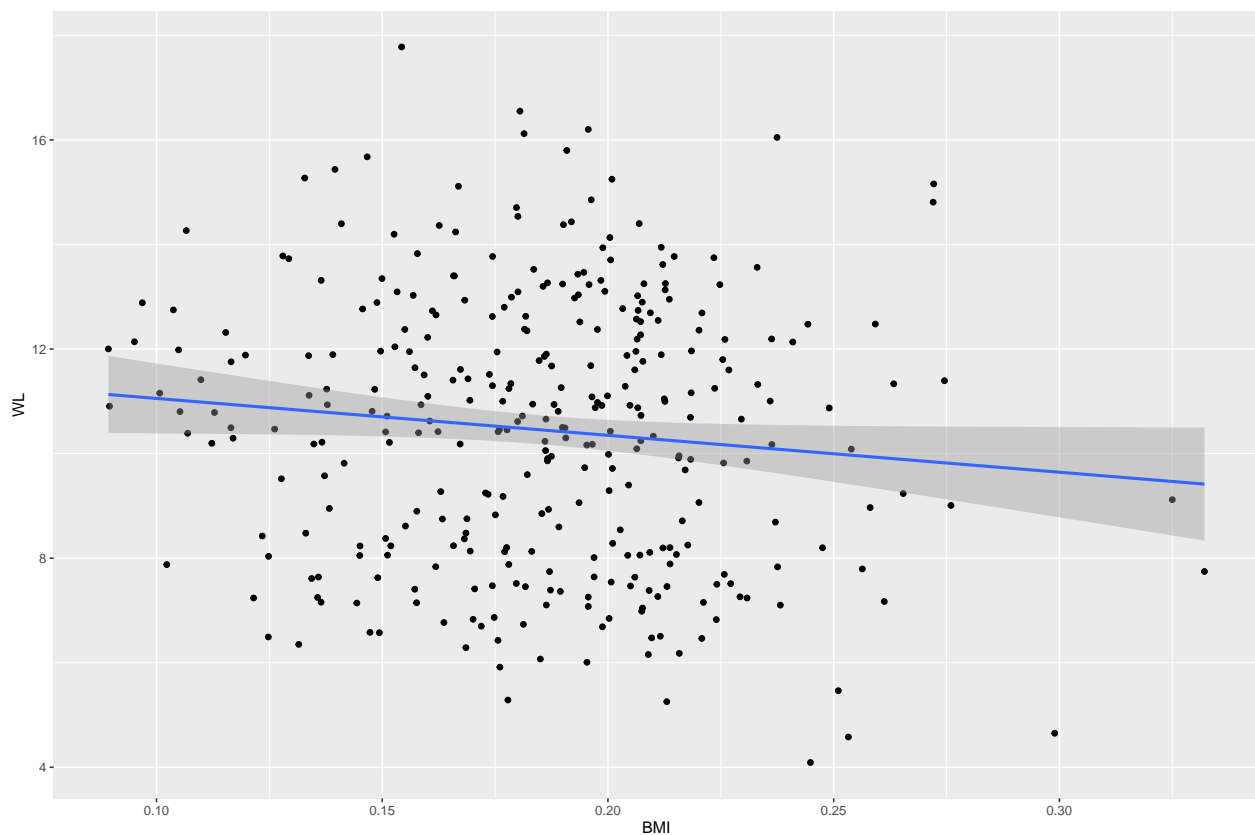
```
Field <- Field %>%
  dplyr::mutate(BMI = Body_Weight / Body_Length)
```

```
ggplot(data = Field, aes(x = BMI, y = WL)) +
  geom_point() +
  stat_smooth(method= "lm")
```

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

```
## Warning: Removed 1 rows containing non-finite values (`stat_smooth()`).
```

```
## Warning: Removed 1 rows containing missing values (`geom_point()`).
```



```
bmi <- lm(WL ~ BMI, data = Field)
```

```
cor(Field$BMI, Field$WL, use = "complete.obs")
```

```
## [1] -0.1062542
```



```
summary(bmi)
```

```
##
## Call:
## lm(formula = WL ~ BMI, data = Field)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -5.9409 -2.2486  0.0812  2.0151  7.1061
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  11.7608     0.6866  17.130  <2e-16 ***
## BMI         -7.0577     3.6248  -1.947   0.0524 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.567 on 332 degrees of freedom
## (1 observation deleted due to missingness)
## Multiple R-squared:  0.01129,    Adjusted R-squared:  0.008312
## F-statistic: 3.791 on 1 and 332 DF,  p-value: 0.05237
```

```
confint(bmi)
```

```
##              2.5 %      97.5 %
## (Intercept)  10.41027 13.11136206
## BMI         -14.18821  0.07274454
```