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/2.imputed_MICE_data_set.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 colums 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.

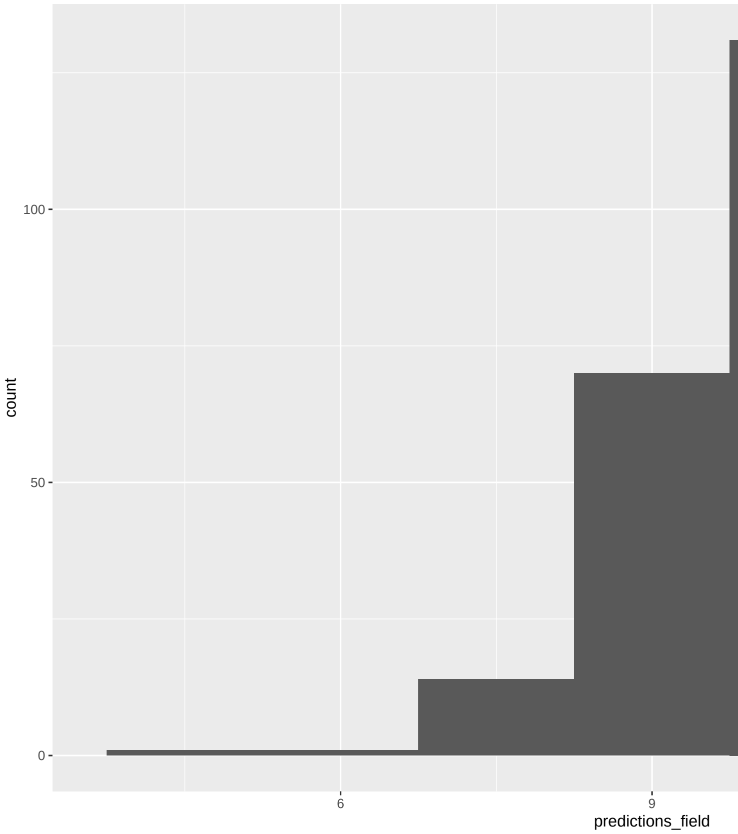
Install the package

```
##
## * checking for file ‘/tmp/RtmprtEvu4/remotesbb1c41dac9be9/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’
```

Data diagnostics

Visualizations

```

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



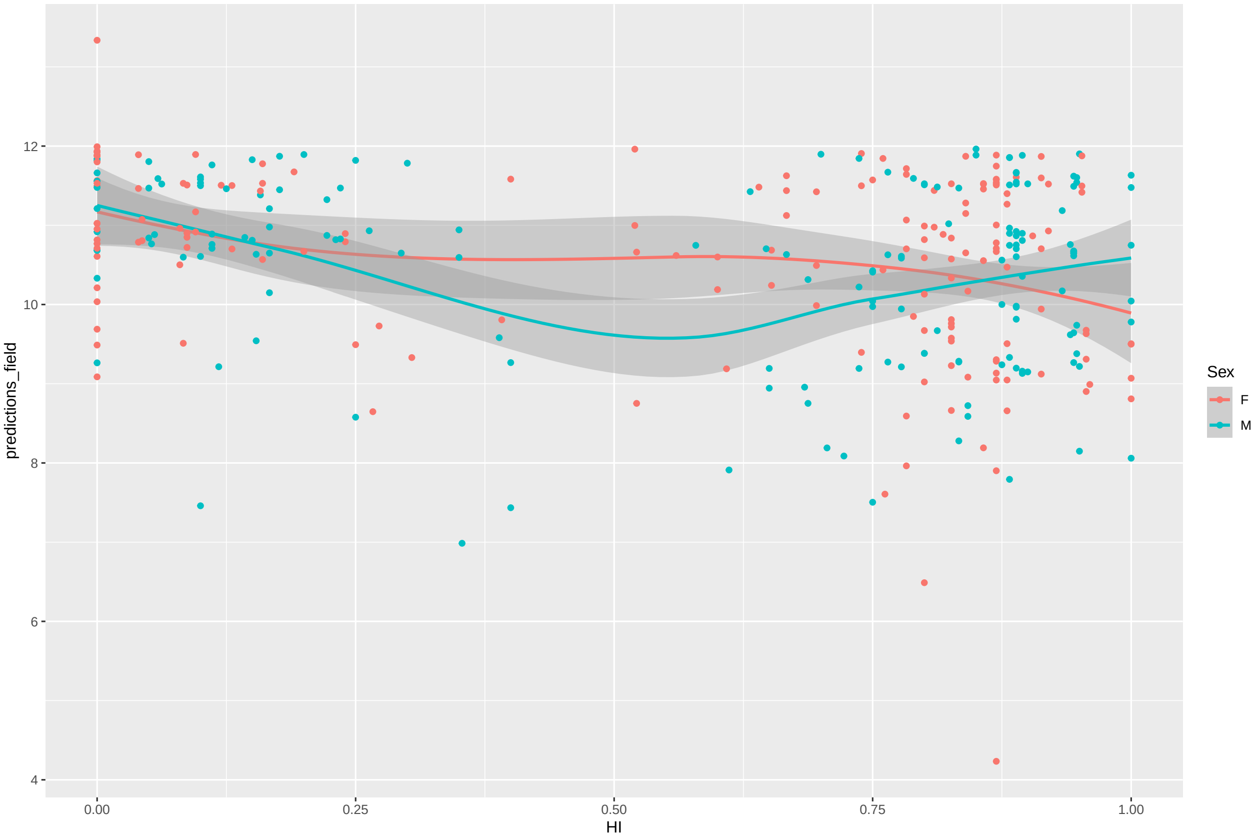
What is the distribution of the predicted weight loss?

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()
```

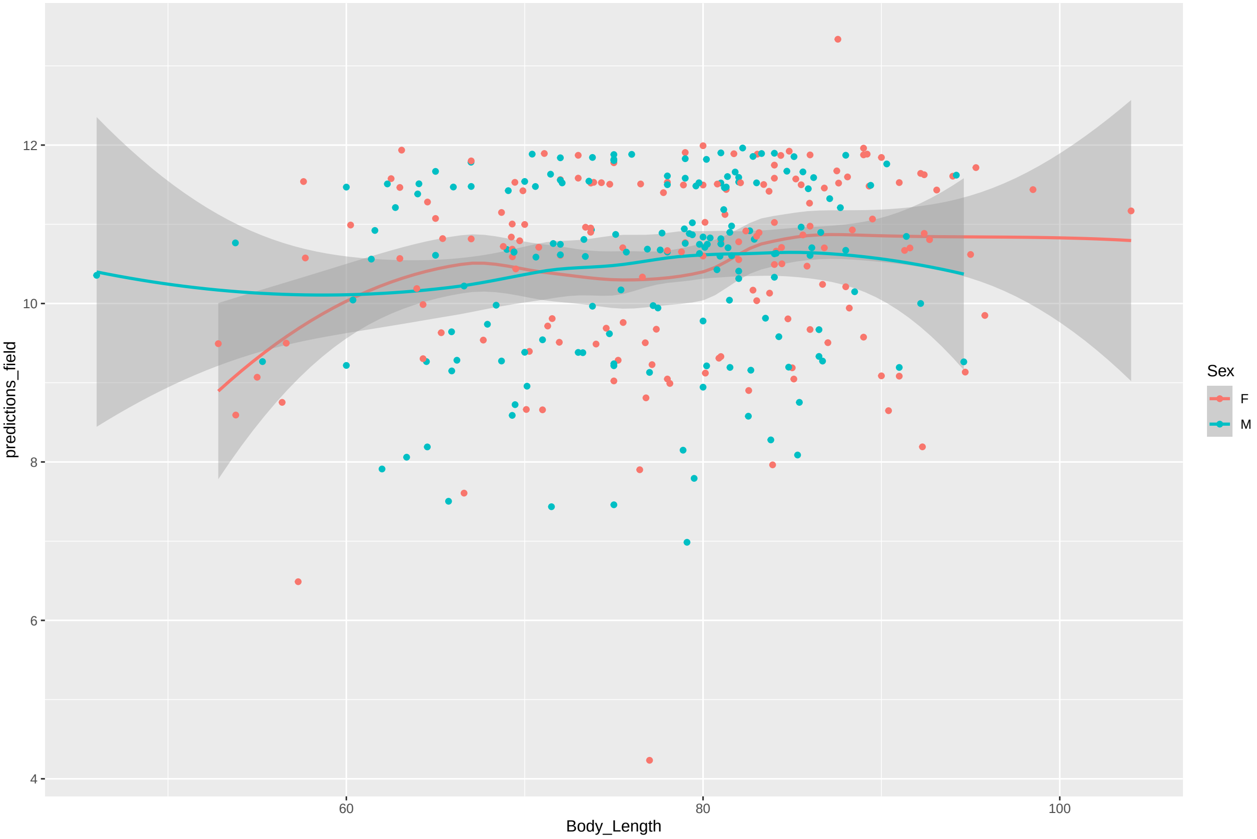
body length

```
## `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()`).
```

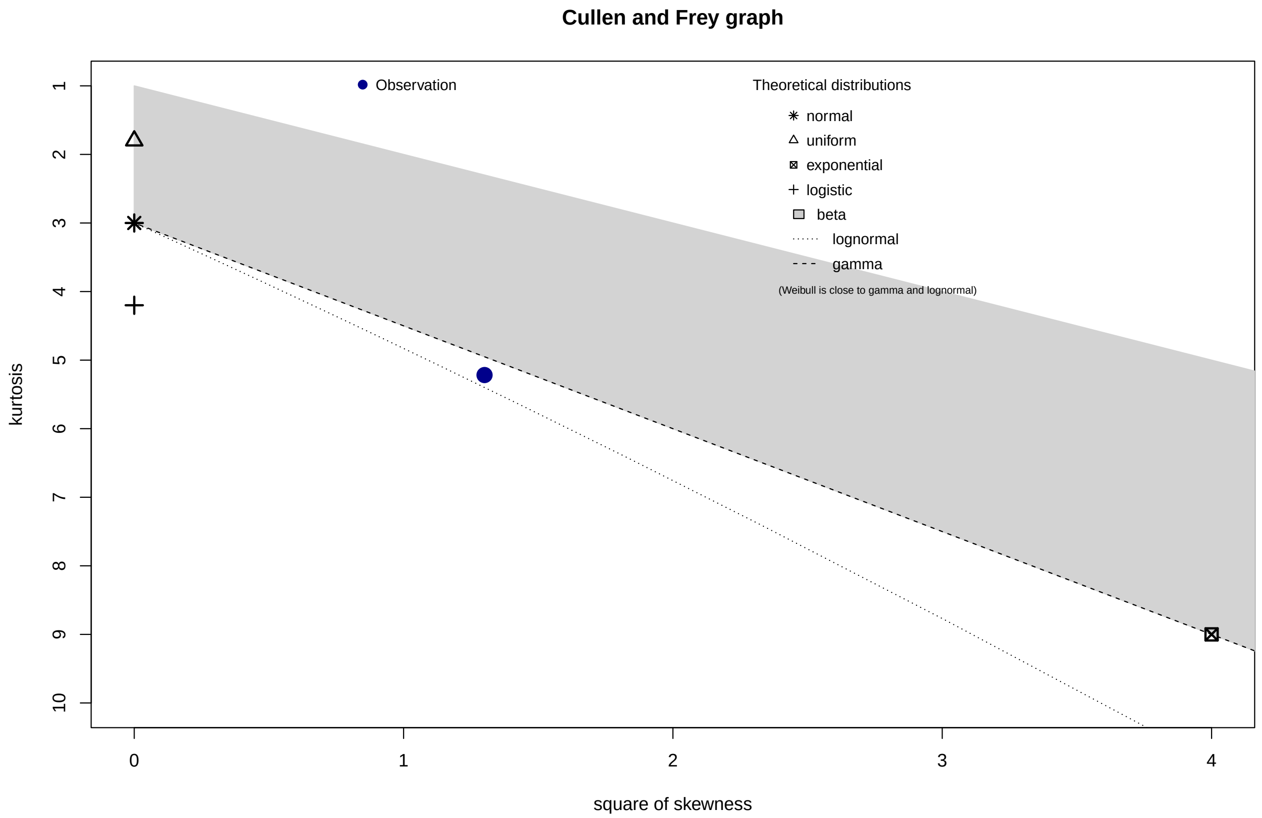


Fitting distributions??

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

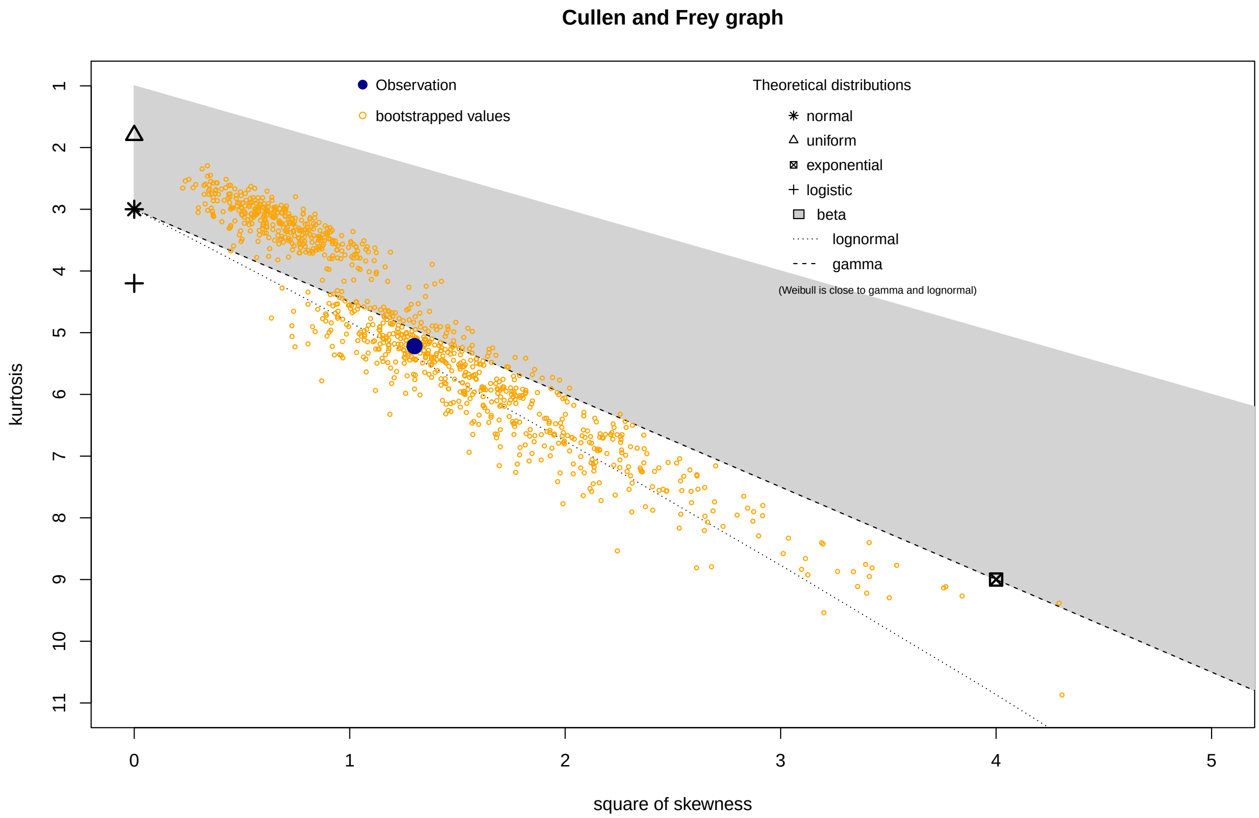
Alice used weibull for the qpcr data. (paper)

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

x <- Field$WL

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



```
## summary statistics
## ------
## min:  4.233544   max:  13.33598
## median:  10.74815
## mean:  10.51001
## estimated sd:  1.175876
## estimated skewness:  -1.140424
## estimated kurtosis:  5.2178
```

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



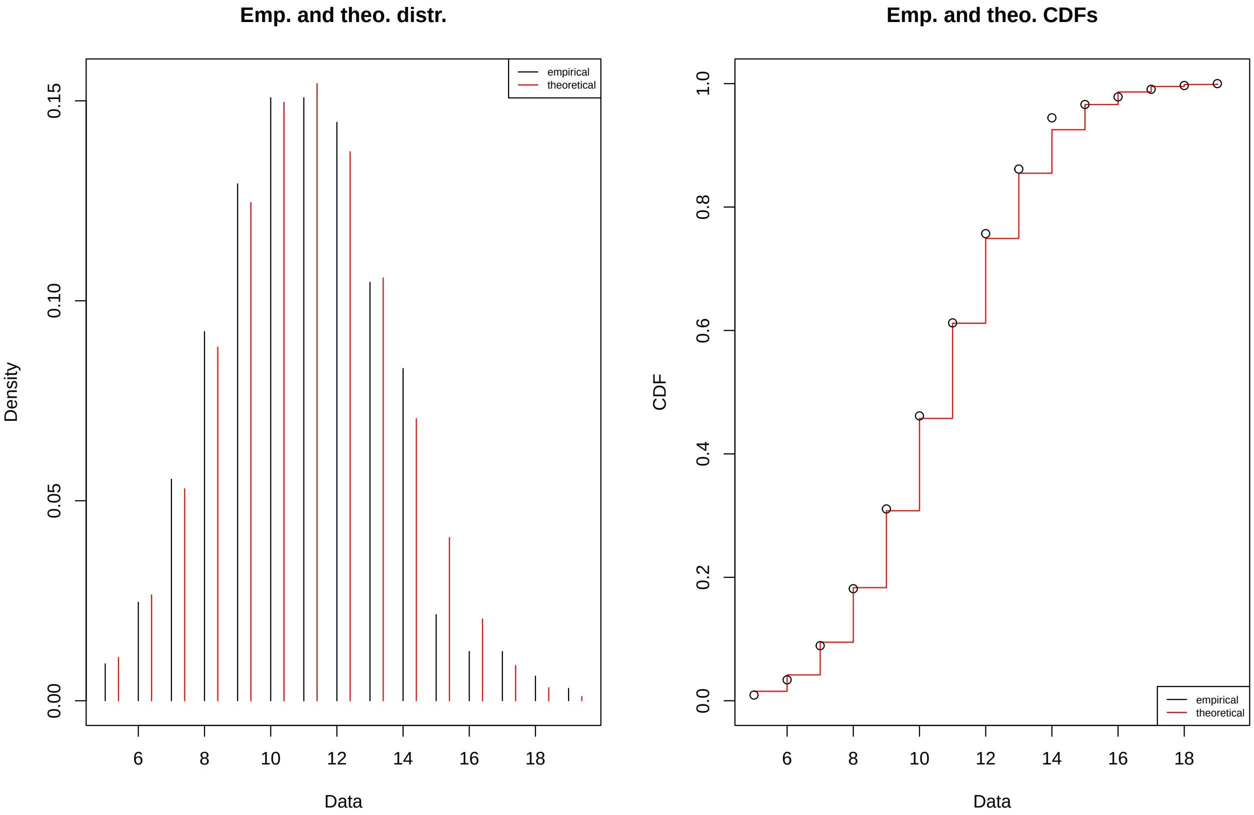
```
## summary statistics
## ------
## min:  4.233544   max:  13.33598
## median:  10.74815
## mean:  10.51001
## estimated sd:  1.175876
## estimated skewness:  -1.140424
## estimated kurtosis:  5.2178
```

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 0.400228 0.005230235
## Fixed parameters:
##      value
## size     27
## Loglikelihood:  -756.9419   AIC:  1515.884   BIC:  1519.668
```

```
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)
  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
##   10.51001208    1.17406552
##  ( 0.06512544) ( 0.04605064)
##
## $loglik
## [1] -513.3086
##
## $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
```

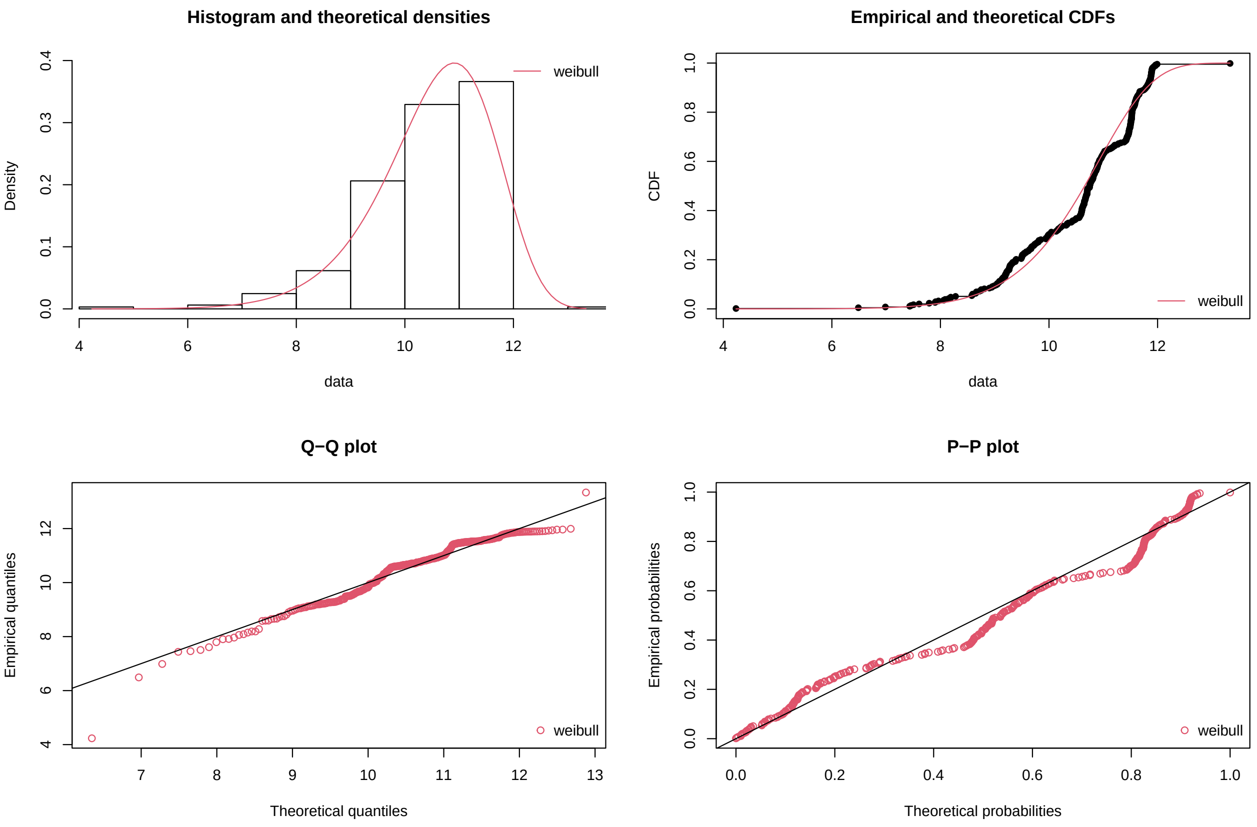
```
## $fit
##       shape         scale
##   11.78591645   10.99295508
##  ( 0.52918655) ( 0.05420501)
##
## $loglik
## [1] -484.6331
##
## $AIC
## NULL
```

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

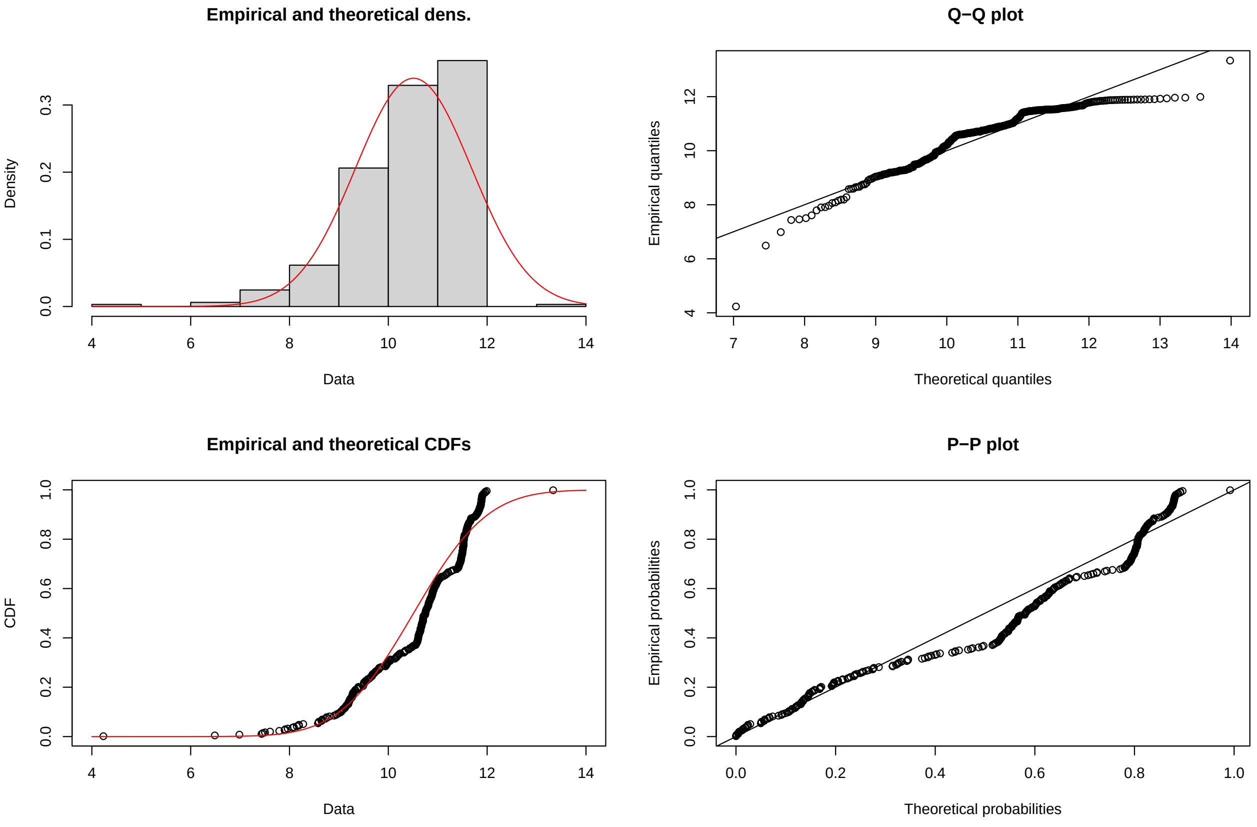
```
## $fit
## [1] "fit failed"
##
## $loglik
## [1] "no loglik computed"
##
## $AIC
## [1] "no aic computed"
```

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

```
## $normal
## $normal$fit
##       mean           sd
##   10.51001208    1.17406552
##  ( 0.06512544) ( 0.04605064)
##
## $normal$loglik
## [1] -513.3086
##
## $normal$AIC
## NULL
```


Histogram and theoretical densities
Density
0.0
0.1
0.2
0.3
0.4
weibull
4
6
8
10
12
data
Empirical and theoretical CDFs
CDF
0.0
0.2
0.4
0.6
0.8
1.0
weibull
4
6
8
10
12
data
Q−Q plot
Empirical quantiles
4
6
8
10
12
weibull
7
8
9
10
11
12
13
Theoretical quantiles
P−P plot
Empirical probabilities
0.0
0.2
0.4
0.6
0.8
1.0
weibull
0.0
0.2
0.4
0.6
0.8
1.0
Theoretical probabilities

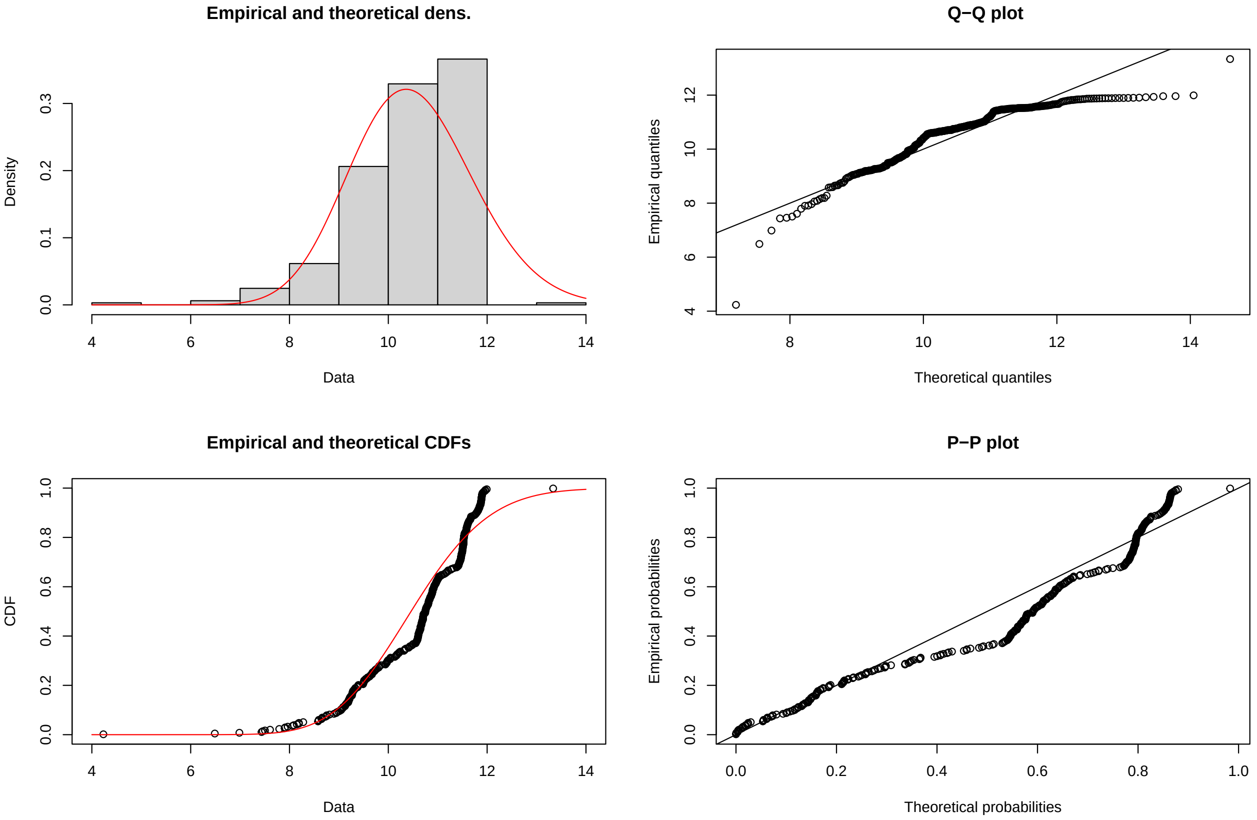
```
plot(normal_)
```


Empirical and theoretical dens.
Density
0.0
0.1
0.2
0.3
4
6
8
10
12
14
Data
Q−Q plot
Empirical quantiles
4
6
8
10
12
7
8
9
10
11
12
13
14
Theoretical quantiles
Empirical and theoretical CDFs
CDF
0.0
0.2
0.4
0.6
0.8
1.0
4
6
8
10
12
14
Data
P−P plot
Empirical probabilities
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
Theoretical probabilities

```
summary(normal_)
```

```
## Fitting of the distribution ' norm ' by maximum likelihood
## Parameters :
##       estimate Std. Error
## mean 10.510012 0.06512544
## sd    1.174066 0.04605049
## Loglikelihood:  -513.3086   AIC:  1030.617   BIC:  1038.185
## Correlation matrix:
##      mean sd
## mean    1  0
## sd      0  1
```

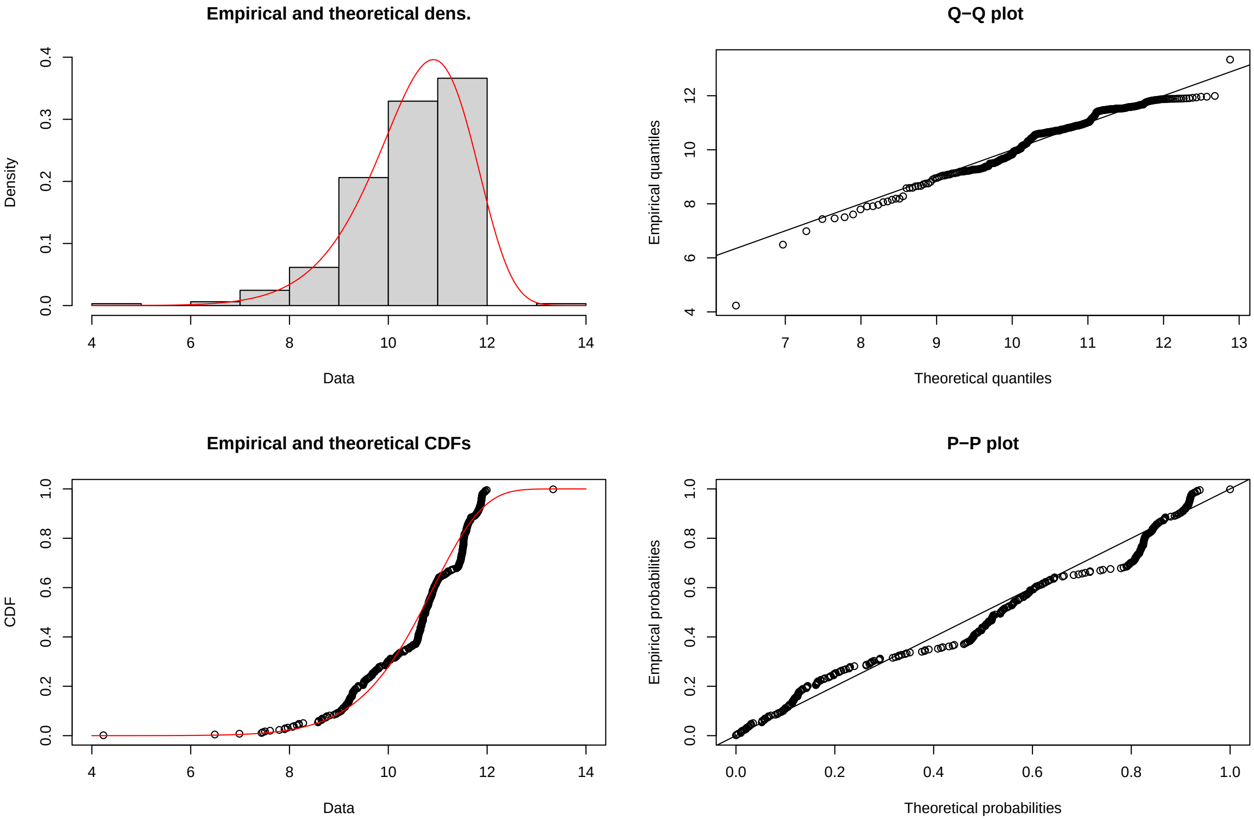
```
plot(gamma_)
```



```
summary(gamma_)
```

```
## Fitting of the distribution ' gamma ' by maximum likelihood
## Parameters :
##        estimate Std. Error
## shape 70.712716  5.5340919
## rate   6.728213  0.5284279
## Loglikelihood:  -532.0799   AIC:  1068.16   BIC:  1075.727
## Correlation matrix:
##           shape      rate
## shape 1.0000000 0.9964667
## rate  0.9964667 1.0000000
```

```
plot(weibull_)
```



```
summary(weibull_)
```

```
## Fitting of the distribution ' weibull ' by maximum likelihood
## Parameters :
##       estimate Std. Error
## shape 11.79385 0.52947312
## scale 10.99296 0.05417176
## Loglikelihood:  -484.633   AIC:  973.2659   BIC:  980.8336
## Correlation matrix:
##          shape    scale
## shape 1.000000 0.298565
## scale 0.298565 1.000000
```

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
## 10.510012081  4.233544127 13.335976577 10.510012081  4.233544127 13.335976577
##   alphaStart       alphaLB      alphaUB     mysdStart        mysdLB       mysdUB
##  0.000000000 -5.000000000  5.000000000  1.000000000  0.000000001 10.000000000
```

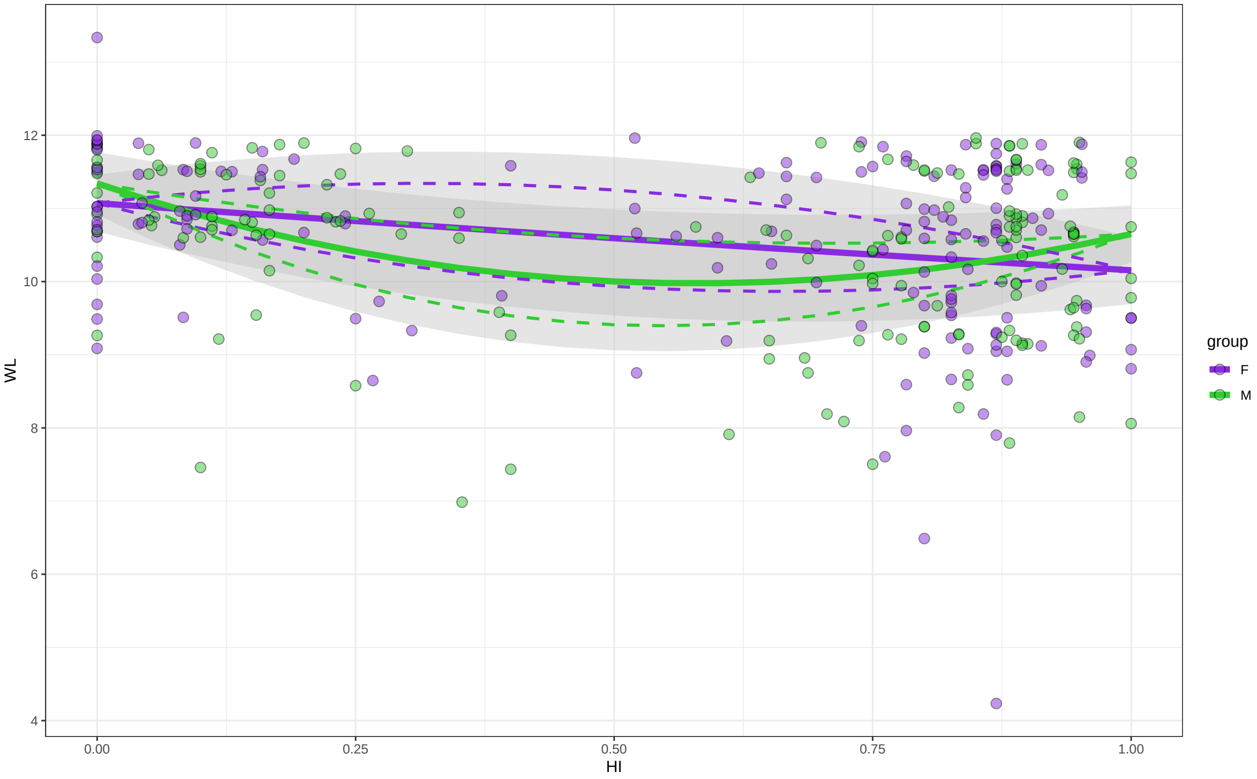
```
speparam <- c(L1start = 10,
                     L1LB = 1e-9,
                     L1UB = 20,
                     L2start = 10,
                     L2LB = 1e-9,
                     L2UB = 20,
                     alphaStart = 0, alphaLB = -5, alphaUB = 5,
                     myshapeStart = 1, 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 4.73   1 0.002099334
## [1] "Testing H1 no alpha vs alpha"
##    dLL dDF     pvalue
## 1 2.55   1 0.02395001
## [1] "Testing H2 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.84   1 0.1946722
## [1] "Testing H2 groupB no alpha vs alpha"
##    dLL dDF      pvalue
## 1 5.02   1 0.001539798
## [1] "Testing H3 groupA no alpha vs alpha"
##   dLL dDF    pvalue
## 1   0   1 0.9529976
## [1] "Testing H3 groupB no alpha vs alpha"
##    dLL dDF      pvalue
## 1 4.84   1 0.001871082
## [1] "Testing H1 vs H0"
##    dLL dDF       pvalue
## 1 8.31   1 4.575242e-05
## [1] "Testing H2 vs H0"
##    dLL dDF    pvalue
## 1 1.41   3 0.4201788
## [1] "Testing H3 vs H1"
##    dLL dDF    pvalue
## 1 2.64   4 0.2605042
## [1] "Testing H3 vs H2"
##    dLL dDF       pvalue
## 1 9.53   2 7.233994e-05
```

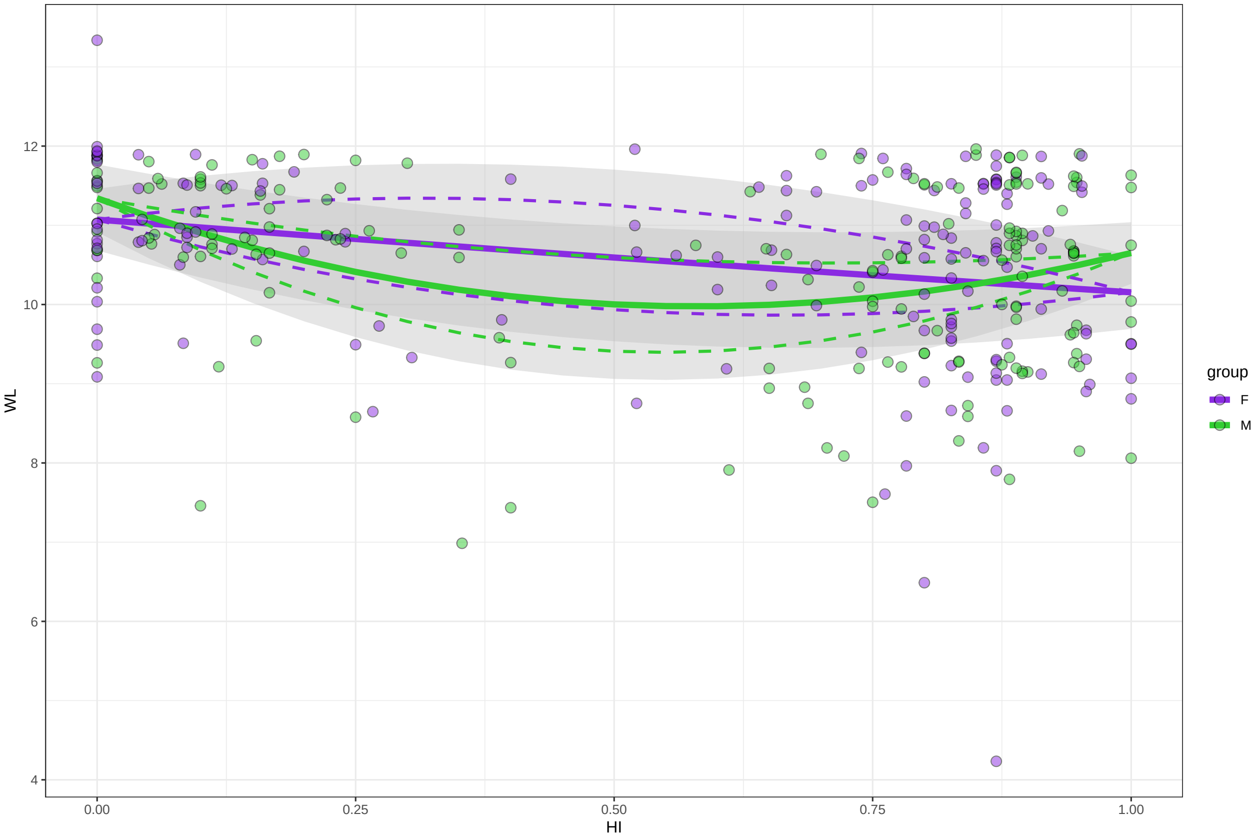
```
plot_WL_Sex<- bananaPlot(mod = fitWL_Sex$H3,
             data = Field,
             response = "WL",
             group = "Sex") +
    scale_fill_manual(values = c("blueviolet", "limegreen")) +
  scale_color_manual(values = c("blueviolet", "limegreen")) +
  theme_bw()
```

```
## 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.
```

```
# Create HI bar
HIgradientBar <- ggplot(data.frame(hi = seq(0,1,0.0001)),
                        aes(x=hi, y=1, fill = hi)) +
  geom_tile() +
  theme_void() +
  scale_fill_gradient(low = "blue", high = "red")  +
  scale_x_continuous(expand=c(.01,0)) +
  scale_y_continuous(expand=c(0,0)) +
  theme(legend.position = 'none')

plot_grid(plot_WL_Sex,
          HIgradientBar,
          nrow = 2,
          align = "v",
          axis = "tlr",
          rel_heights = c(13, 1))
```



```
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

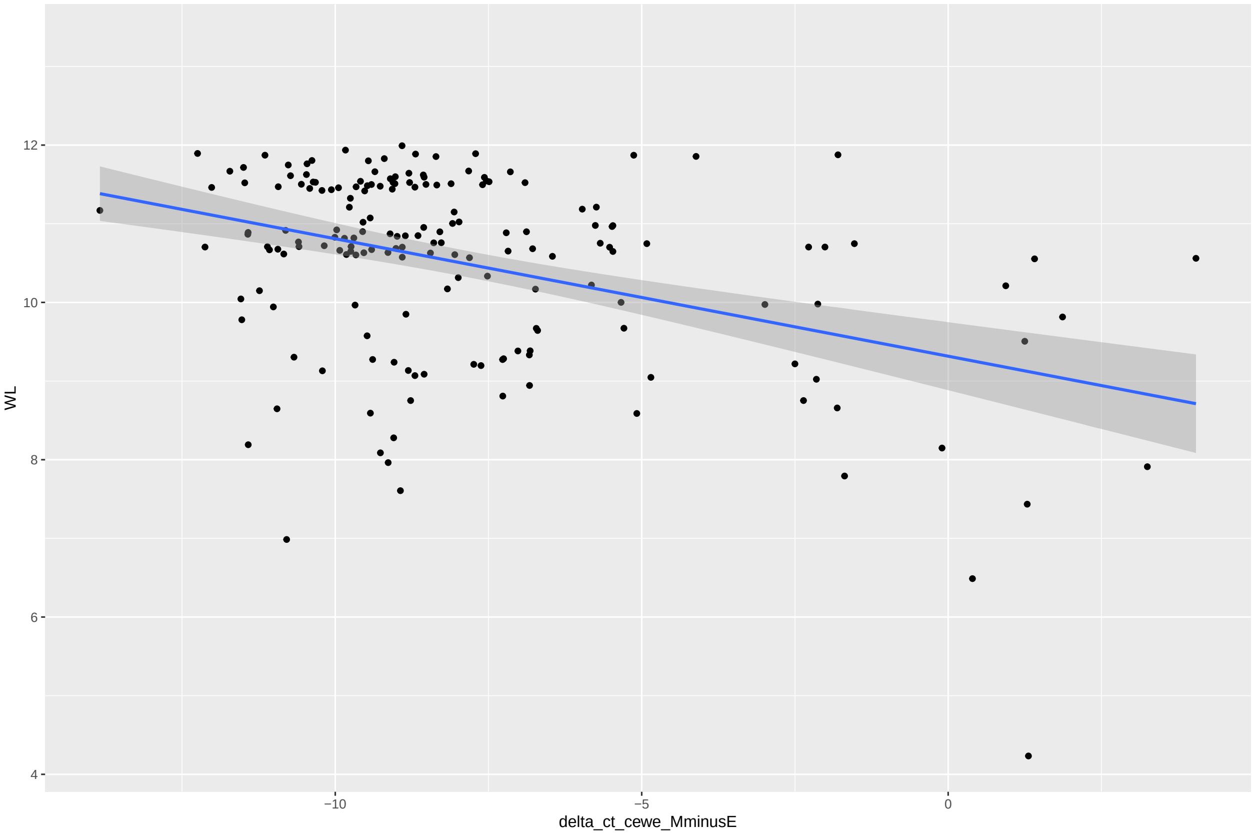
H3: the mean load can differ both across subspecies and between 2 groups

```
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 146 rows containing non-finite values (`stat_smooth()`).
```

```
## Warning: Removed 146 rows containing missing values (`geom_point()`).
```



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

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

```
## [1] -0.4000163
```

```
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
## -4.8869 -0.5427  0.2109  0.7972  2.2919
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)            9.31610    0.21901  42.538  < 2e-16 ***
## delta_ct_cewe_MminusE -0.14933    0.02572  -5.807 2.89e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.13 on 177 degrees of freedom
##   (146 observations deleted due to missingness)
## Multiple R-squared:   0.16,  Adjusted R-squared:  0.1553
## F-statistic: 33.72 on 1 and 177 DF,  p-value: 2.894e-08
```

```
confint(tolerance)
```

```
##                             2.5 %      97.5 %
## (Intercept)             8.8839025  9.74829863
## delta_ct_cewe_MminusE -0.2000837 -0.09857996
```

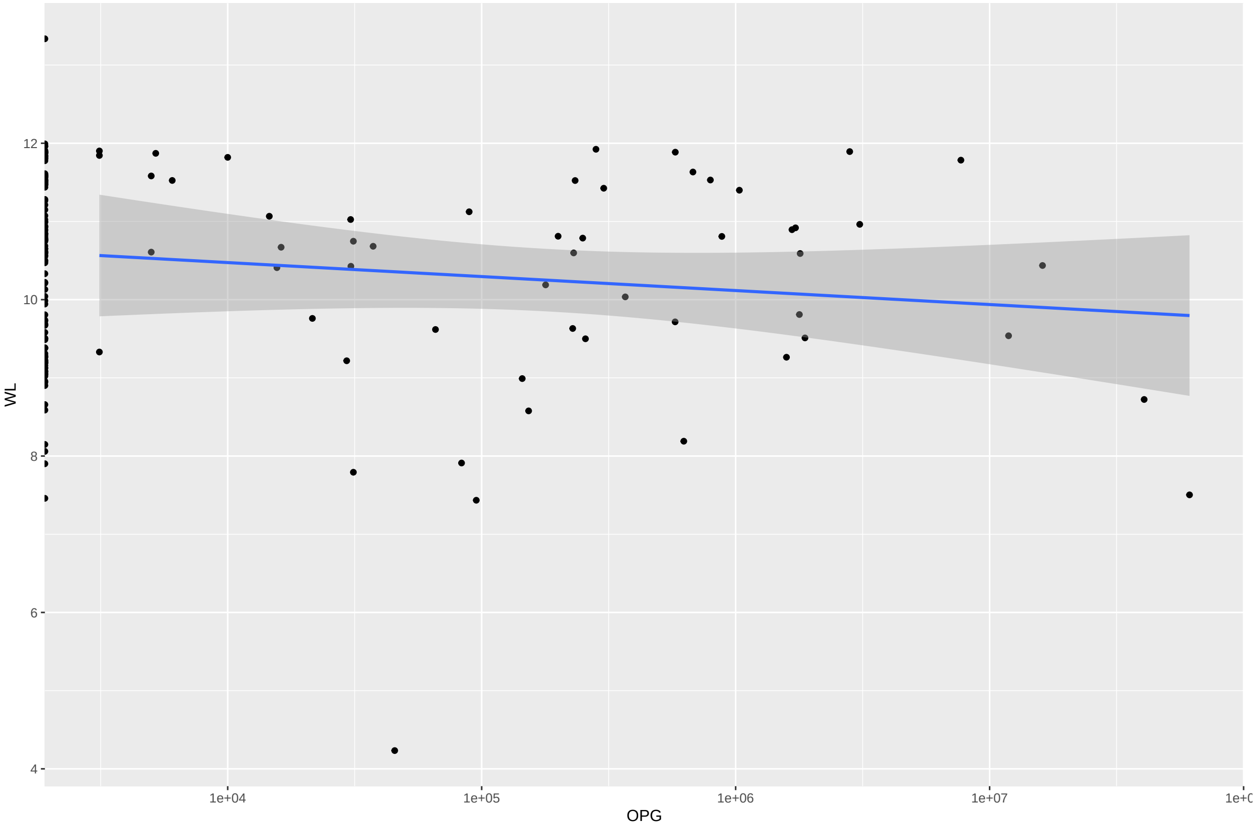
```
ggplot(data = Field, aes(x = OPG, y = WL)) +
  geom_point() +
  stat_smooth(method= "lm") +
  scale_x_log10()
```

```
## 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 270 rows containing non-finite values (`stat_smooth()`).
```

```
## Warning: Removed 148 rows containing missing values (`geom_point()`).
```



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

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

```
## [1] -0.1926576
```

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


summary(tolerance)
```

```
##
## Call:
## lm(formula = WL ~ OPG, data = Field)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.2266 -0.9518  0.2245  1.0493  2.8739
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.046e+01  9.398e-02 111.319   <2e-16 ***
## OPG         -4.227e-08  1.628e-08  -2.597   0.0102 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.234 on 175 degrees of freedom
##   (148 observations deleted due to missingness)
## Multiple R-squared:  0.03712,    Adjusted R-squared:  0.03161
## F-statistic: 6.746 on 1 and 175 DF,  p-value: 0.0102
```

```r
confint(tolerance)
```

```
##                     2.5 %        97.5 %
## (Intercept)  1.027658e+01  1.064754e+01
## OPG         -7.439688e-08 -1.015092e-08
```

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


summary(tolerance)
```

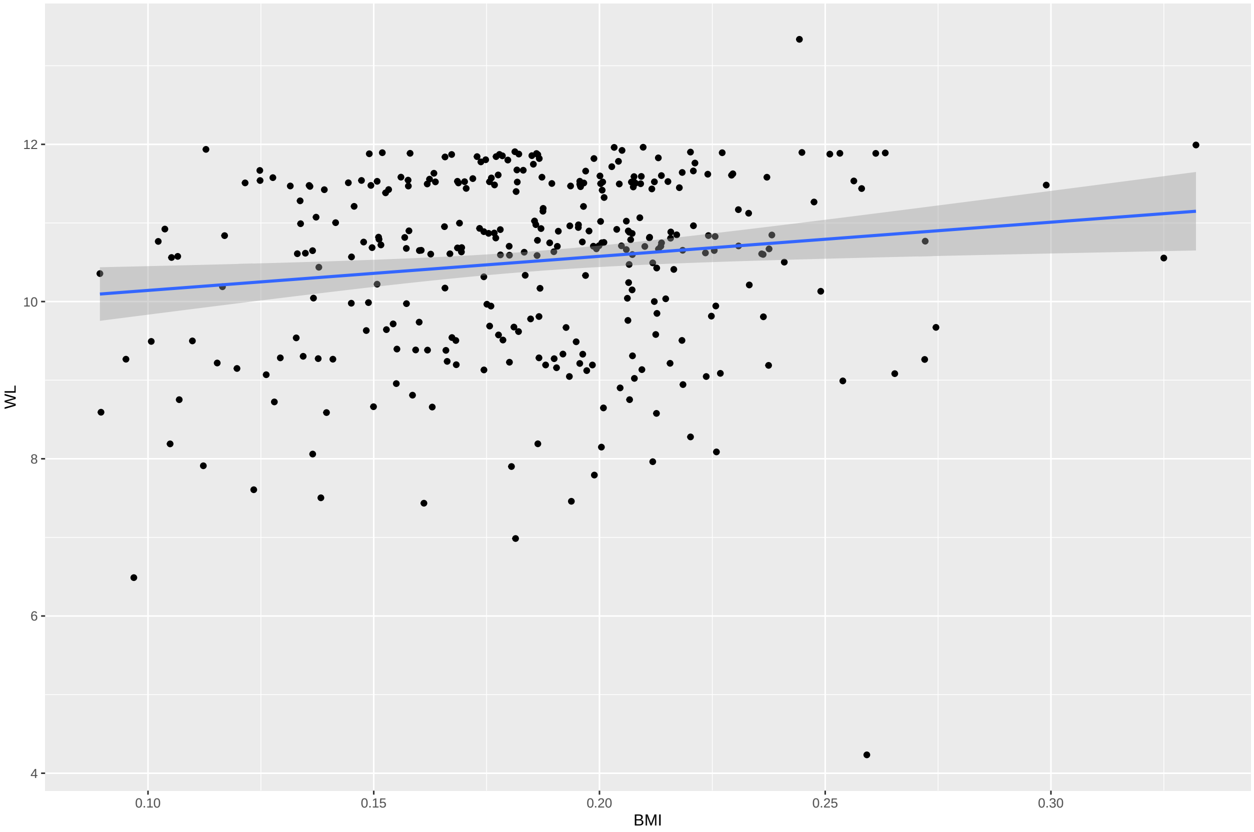
```
##
## Call:
## lm(formula = WL ~ OPG * delta_ct_cewe_MminusE, data = Field)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7718 -0.7674  0.1318  0.7395  1.8132
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  9.065e+00  4.071e-01  22.268  < 2e-16 ***
## OPG                         -1.511e-05  1.053e-05  -1.434 0.158457
## delta_ct_cewe_MminusE       -1.990e-01  5.330e-02  -3.734 0.000529 ***
## OPG:delta_ct_cewe_MminusE -1.889e-06  3.087e-06  -0.612 0.543525
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.16 on 45 degrees of freedom
##   (276 observations deleted due to missingness)
## Multiple R-squared:  0.4552, Adjusted R-squared:  0.4189
## F-statistic: 12.53 on 3 and 45 DF,  p-value: 4.369e-06
```

```
confint(tolerance)
```

```
##                                   2.5 %        97.5 %
## (Intercept)                8.245294e+00  9.885128e+00
## OPG                       -3.632549e-05  6.110134e-06
## delta_ct_cewe_MminusE     -3.063484e-01 -9.164759e-02
## OPG:delta_ct_cewe_MminusE -8.106031e-06  4.327224e-06
```

```
Field <- Field %>%
  dplyr::mutate(BMI = Body_Weight / (Body_Length)) #^2) which is the correct
# way to calculatebmi?

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] 0.1430957
```

```
summary(bmi)
```

```
##
## Call:
## lm(formula = WL ~ BMI, data = Field)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.5992 -0.7817  0.2431  0.9436  2.5680
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.7079     0.3168  30.645  < 2e-16 ***
## BMI           4.3394     1.6725   2.594  0.00991 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.167 on 322 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.02048,    Adjusted R-squared:  0.01743
## F-statistic: 6.731 on 1 and 322 DF,  p-value: 0.009907
```

```
confint(bmi)
```

```
##                2.5 %    97.5 %
## (Intercept) 9.084681 10.331149
## BMI         1.048858  7.629864
```