15. HI across facs data

Fay

2022-08-09

load libraries

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

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## v purrr   0.3.5
## -- 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

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

```
# Selecting facs
Gene_lab   <- 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")

#add a suffix to represent changes in data file
Gene_lab_imp <- paste(Gene_lab, "imp", sep = "_")

facs_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") #"IRG6")

facs_wild_imp <- paste(facs_wild, "imp", sep = "_")

Facs_lab <- c("Position", "CD4", "Treg", "Div_Treg", "Treg17", "Th1",
                    "Div_Th1", "Th17", "Div_Th17", "CD8", "Act_CD8",
                    "Div_Act_CD8", "IFNy_CD4", "IFNy_CD8") #,"Treg_prop",
                   # "IL17A_CD4")
```

```r
Facs_wild <- c( "Treg", "CD4", "Treg17", "Th1", "Th17", "CD8",
                     "Act_CD8", "IFNy_CD4",  "IFNy_CD8") #"IL17A_CD4",
```

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 across facs

```r
# Selecting the field samples

field <- hm %>%
  dplyr::filter(origin == "Field")

field <- unique(field)

#make a factor out of the melting curves (important for later visualization)
field <- field %>%
  dplyr::mutate(MC.Eimeria = as.factor(MC.Eimeria))

facs <- field %>%
  dplyr::select(all_of(Facs_wild))


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

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


##select same rows in the first table
field <- field[row.names(facs), ]
```

Install the package

```
##
## * checking for file '/tmp/RtmpRV4t5b/remotesdda52679d2c04/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

```r
x <- field$CD4



# 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
##   40.2010526   11.1986606
##  ( 1.1489583) ( 0.8124362)
##
## $loglik
## [1] -364.2996
##
## $AIC
## NULL
```

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

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

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

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

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

```
## $fit
##      shape        scale
##    3.9324346   44.4023805
##   ( 0.3090144) ( 1.2244501)
##
## $loglik
## [1] -364.467
##
## $AIC
## NULL
```

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

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

```r
# remove NA in HI
field <- field %>%
  drop_na(HI)

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

parasiteLoad::getParamBounds("weibull", data = field, response = "CD4")
```

```
##      L1start         L1LB         L1UB      L2start         L2LB         L2UB
## 40.201052632  0.000000001 68.500000000 40.201052632  0.000000001 68.500000000
##   alphaStart      alphaLB      alphaUB myshapeStart    myshapeLB    myshapeUB
##  0.000000000 -5.000000000  5.000000000  1.000000000  0.000000001  5.000000000
```

```r
#write a function to test parasite load on a facs cell proportion

facs_hypothesis <- function(y) {
  parasiteLoad::analyse(data = field,
                        response = y,
                        model = "weibull",
                        group = "Sex")
}
```

```r
parasite_load_facs <- function(x, y) {
  # x takes the cell without "" and y is the cell with ""
  x <- parasiteLoad::analyse(data = field,
                        response = y,
                        model = "weibull",
                        group = "Sex")


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

}

facs_hypothesis("CD4")
```

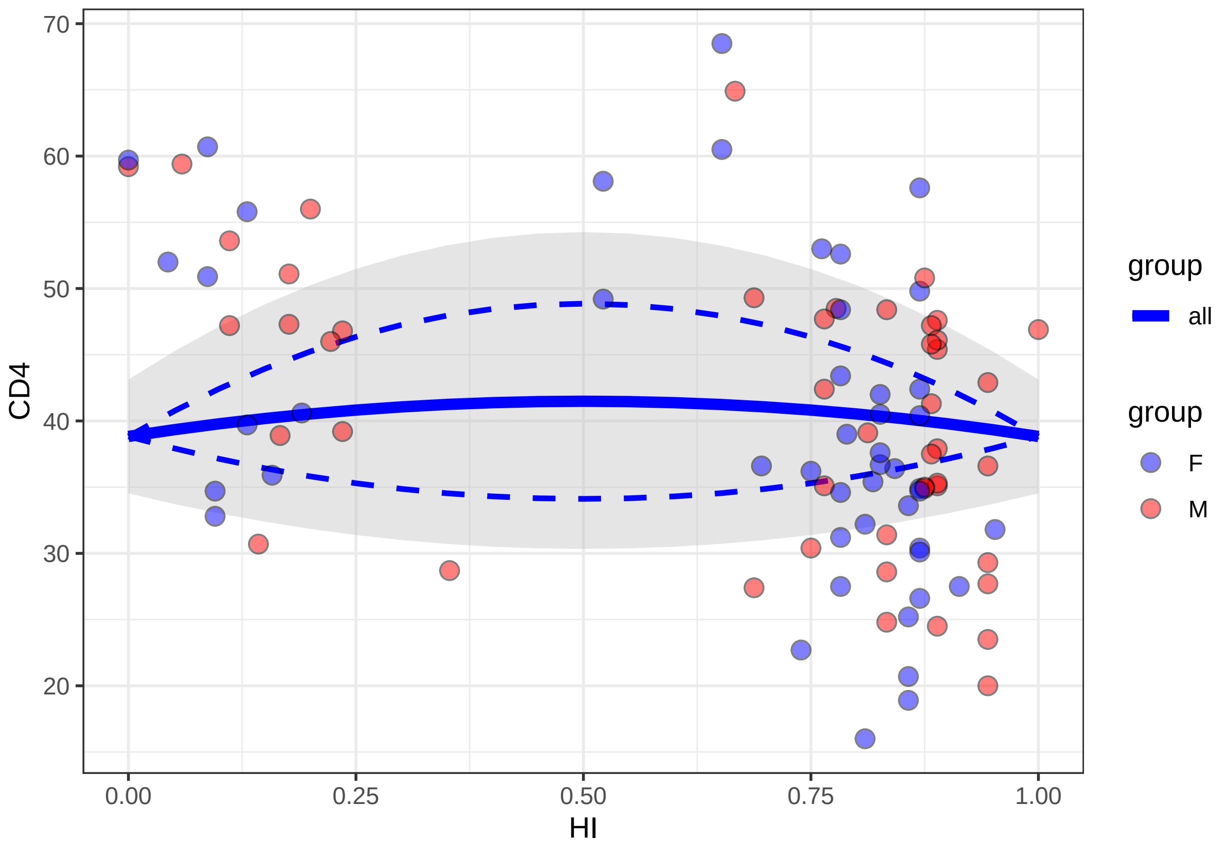
```
## [1] "Analysing data for response: CD4"
## [1] "Fit for the response: CD4"
## [1] "Fitting for all"
## [1] "Fitting model basic without alpha"
## [1] "Did converge"
## [1] "Fitting model basic with alpha"
## [1] "Did converge"
## [1] "Fitting model advanced without alpha"
## [1] "Did converge"
## [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 0.26   1 0.4676834
## [1] "Testing H1 no alpha vs alpha"
##   dLL dDF     pvalue
## 1 1.6   1 0.07405386
## [1] "Testing H2 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.24   1 0.4844657
## [1] "Testing H2 groupB no alpha vs alpha"
##    dLL dDF   pvalue
## 1 0.02   1 0.824134
## [1] "Testing H3 groupA no alpha vs alpha"
##    dLL dDF     pvalue
## 1 2.36   1 0.02983287
## [1] "Testing H3 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.16   1 0.5690839
## [1] "Testing H1 vs H0"
##    dLL dDF       pvalue
## 1 9.17   1 1.853683e-05
## [1] "Testing H2 vs H0"
##    dLL dDF    pvalue
## 1 0.85   3 0.6390205
## [1] "Testing H3 vs H1"
##    dLL dDF    pvalue
## 1 2.16   4 0.3646614
## [1] "Testing H3 vs H2"
##     dLL dDF       pvalue
## 1 10.48   2 2.807156e-05
```

```
## $H0
##
## Call:
## bbmle::mle2(minuslogl = response ~ dweibull(shape = myshape,
##     scale = MeanLoad(L1, L1, alpha, HI)/gamma(1 + (1/myshape))),
##     start = start, method = config$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
## 38.8303526 -0.1367112  3.9473623
##
## Log-likelihood: -364.2
## 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
## 45.4296434 32.4599266 -0.3466487  4.3684071
##
## Log-likelihood: -355.04
## 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
## 37.6506909 -0.1987171  3.6141515
##
## Log-likelihood: -187.06
## 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
## 40.1792486 -0.0558514  4.4045501
##
## Log-likelihood: -176.3
## 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
## 43.3293159 27.0701232 -0.7897272  4.0863711
##
## Log-likelihood: -181.15
## 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
## 47.1330155 35.4855422 -0.1360677  4.8941954
##
## Log-likelihood: -171.73
## Best method: bobyqa
```

```
parasite_load_facs(CD4, "CD4")
```

```
## [1] "Analysing data for response: CD4"
## [1] "Fit for the response: CD4"
## [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 0.26   1 0.4676834
## [1] "Testing H1 no alpha vs alpha"
##   dLL dDF     pvalue
## 1 1.6   1 0.07405386
## [1] "Testing H2 groupA no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.24   1 0.4844657
## [1] "Testing H2 groupB no alpha vs alpha"
##    dLL dDF   pvalue
## 1 0.02   1 0.824134
## [1] "Testing H3 groupA no alpha vs alpha"
##    dLL dDF     pvalue
## 1 2.36   1 0.02983287
## [1] "Testing H3 groupB no alpha vs alpha"
##    dLL dDF    pvalue
## 1 0.16   1 0.5690839
## [1] "Testing H1 vs H0"
##    dLL dDF       pvalue
## 1 9.17   1 1.853683e-05
## [1] "Testing H2 vs H0"
##    dLL dDF    pvalue
## 1 0.85   3 0.6390205
## [1] "Testing H3 vs H1"
##    dLL dDF    pvalue
## 1 2.16   4 0.3646614
## [1] "Testing H3 vs H2"
##     dLL dDF       pvalue
## 1 10.48   2 2.807156e-05
```

```
## 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.
```



70
60
50
40
30
20
CD4
0.00
0.25
0.50
0.75
1.00
HI
group
all
group
F
M