First draft of titer models

Menna

```
rm(list = ls())
library(nlme)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:nlme':
##
##     collapse

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(lattice)
library(hexbin)
```

```
## Warning: package 'hexbin' was built under R version 4.3.3
```

```
library(ggplot2)
```

```
# This function and others will be relocated to a "base_functions.R" file and source from it
calc_percent_parms_in_CI <- function(percents, intercept, slope, CIs) {

  # This function calculates percentages of intercept and/or slope values lying within the estimated CI
  # CIs is a 2x2 matrix where
  #             lower bound    upper bound
  # intercept         x             y
  # slope             z             k

  c1 = percents[1]; c2 = percents[2]; c3 = percents[3]
  if (CIs[1,1] <= intercept &  intercept <= CIs[1,2])
    c1 = c1 + 1
  if (CIs[2,1] <= slope & slope <= CIs[2,2])
    c2 = c2 + 1
  if (CIs[1,1] <= intercept & intercept <= CIs[1,2] & CIs[2,1] <= slope & slope <= CIs[2,2])
```

```
    c3 = c3 + 1

  return(c(c1, c2, c3))
}
```

functions to introduce censoring

```
interval_censoring <- function(titer_0) {
  base_dilution = 5   # lowest possible titer
  n_dilutions = 10  # number of dilutions
  ratio = 2         # denoting serum two-fold dilutions
  dilutions <- base_dilution * ratio ^ (0:(n_dilutions - 1))
  log2_dilutions <- log(dilutions, base = 2)
  n_individuals <- length(titer_0)

  # discretization of dilutions
  for (i in 1:n_individuals) {
    closest_dilution = rep(Inf, times = n_individuals)
    for (j in 2:n_dilutions-1) {
      if (abs(log2_dilutions[j] - titer_0[i]) < abs(closest_dilution[i] - titer_0[i])) {
        closest_dilution[i] = log2_dilutions[j]
      }
    }
    titer_0[i] = closest_dilution[i]
  }
return(titer_0)
}
```

```
left_censoring <- function(titer_0, lower_LoD){

  lower_limit = 5   # lowest possible titer
  n_dilutions = 10  # number of dilutions
  ratio = 2         # denoting serum two-fold dilutions
  dilutions <- lower_limit * ratio ^ (0:(n_dilutions - 1))
  log2_dilutions <- log(dilutions, base = 2)

  # lower limit of detection
  titer_0[titer_0 < log2_dilutions[2]] = log2_dilutions[lower_LoD]

return(titer_0)
}
```

```
right_censoring <- function(titer_0, upper_LoD){

  lower_limit = 5   # lowest possible titer
  n_dilutions = 10  # number of dilutions
  ratio = 2         # denoting serum two-fold dilutions
  dilutions <- lower_limit * ratio ^ (0:(n_dilutions - 1))
  log2_dilutions <- log(dilutions, base = 2)

  # upper limit of detection
  titer_0[titer_0 > log2_dilutions[9]] = log2_dilutions[upper_LoD]
```

```
return(titer_0)
}
```

```
run_lme <- function(IDs, strain, titer_0, titer_30) {

# testing dataset
test_dataset <- data.frame(IDs, strain, titer_0, titer_30) # titer 0, 30 change
test_dataset <- arrange(test_dataset, IDs)

# lme fitting
model <- lme(titer_30 ~  0 + strain + titer_0:strain, random = ~1|IDs, data = test_dataset, method = "RI

return(model)
}
# note to me: da a7san mn lamma kan kol shwaya m7taga tsmy variable gded w t2oly censored data censored
```

Inference from a linear model without individual effects

```
n_individuals = 50
IDs = c(1:n_individuals)
percents = c(0, 0, 0)

# here we do log of means not mean of logs, not the same bec log function isn't linear
intercept = log(rnorm(1, mean = 40, sd = 10), base = 2) # log(40, base = 2) = 5.3
slope = rnorm(1, mean = 0.3, sd = 0.07)
titer_0 = log(rnorm(n_individuals, mean = 88, sd = 28), base = 2) # log(88, base = 2) = 6.5

for (i in 1:100) {

  noise = log(rnorm(n_individuals, mean = 5, sd = 1.2), base = 2)

  titer_30 = intercept + slope * titer_0 + noise
  # plot(titer_0,titer_30)

  input_data = data.frame(IDs, titer_0, titer_30)

  model <- lm(titer_30 ~ titer_0, data = input_data)
  # summary(model)

  # par(mfrow = c(2,2))
  # plot(model)

  # Assessing confidence in parameters
  CIs = confint(model)  # calculates 95% CIs

  percents = calc_percent_parms_in_CI(percents, intercept, slope, CIs)
}

cat("Percentage of intercept value lying within the estimated CI is", percents[1] * 100 / i, "%", "\n")
```

```
## Percentage of intercept value lying within the estimated CI is 17 %
```

```r
cat("Percentage of slope value lying within the estimated CI is", percents[2] * 100 / i, "%", "\n")
```

```
## Percentage of slope value lying within the estimated CI is 95 %
```

```r
cat("Percentage of both intercept and slope values lying within the estimated CIs is", percents[3] * 10
```

```
## Percentage of both intercept and slope values lying within the estimated CIs is 14 %
```

Inference from a mixed effects model (with individual effects fixed across strains)

```r
n_individuals <- 50000
n_strains <- 2
IDs <- rep(c(1:n_individuals), n_strains)
strain <- c(rep("A",n_individuals), rep("B", n_individuals))

# Initialization of parameters
intercept_strain_A = runif(1, min = 0.1, max = 12) # log(40, base = 2) = 5.3
intercept_strain_B = runif(1, min = 5, max = 19) # log(40, base = 2) = 3

slope_strain_A <- runif(1, min = 0.3, max = 0.7)
slope_strain_B <- runif(1, min = 0.5, max = 0.8)

# Generating titer 0
titer_0_strain_A <- runif(n_individuals, min = 4.5, max = 8)  # log(88, base = 2) = 6.5
titer_0_strain_B <- runif(n_individuals, min = 2, max = 5) # log(16, base = 2) = 4

titer_0 <- c(titer_0_strain_A, titer_0_strain_B)
```

To calculate titer 30, we have two scenarios as follows:

alpha. The true scenario happening in human body: titer 30 shaped based on uncensored titer 0 beta. The scenario we have as modellers: titers 30 predicted from censored titer 0

So, I am considering those two scenarios. Here is scenario "alpha":

```r
# calculation of titer 30
individual_effect_sd <- 0.3
individual_effect <- rnorm(n_individuals, mean = 0, sd = individual_effect_sd)

noise_A = runif(n_individuals, min = 1, max = 2)
titer_30_strain_A <- intercept_strain_A + slope_strain_A * titer_0_strain_A + noise_A + individual_effe

noise_B =  runif(n_individuals, min = 1, max = 2)
titer_30_strain_B <- intercept_strain_B + slope_strain_B * titer_0_strain_B + noise_B + individual_effe

titer_30 <- c(titer_30_strain_A, titer_30_strain_B)
```

Introducing censoring

Three types of censoring events are involved:

1. left censoring: having a lower limit of detection (LoD)

2. right censoring: having an upper limit of detection (LoD)
3. interval censoring: measuring only serial fold dilutions i.e. discretization of titers

```
# applying interval censoring to titer 0
censored_titer_0_strain_A = interval_censoring(titer_0_strain_A)
censored_titer_0_strain_B = interval_censoring(titer_0_strain_B)
# grouping data from strain A and B
censored_titer_0 <- c(censored_titer_0_strain_A, censored_titer_0_strain_B)
```

Here is scenario "beta":

```
# calculation of titer 30
# individual_effect_sd <- 0.3
# individual_effect <- rnorm(n_individuals, mean = 0, sd = individual_effect_sd)
#
# noise_A = log(rnorm(n_individuals, mean = 5, sd = 1.2), base = 2)
# titer_30_strain_A <- intercept_strain_A + slope_strain_A * censored_titer_0_strain_A + noise_A + indi
#
# noise_B = log(rnorm(n_individuals, mean = 5, sd = 1.2), base = 2)
# titer_30_strain_B <- intercept_strain_B + slope_strain_B * censored_titer_0_strain_B + noise_B + indi
#
# censor_based_titer_30 <- c(titer_30_strain_A, titer_30_strain_B)
```

```
# I am applying censoring to titer 30
censored_titer_30_strain_A <- interval_censoring(titer_30_strain_A)
censored_titer_30_strain_B <- interval_censoring(titer_30_strain_B)

censored_titer_30 <- c(titer_30_strain_A, titer_30_strain_B)
# censor_based_titer_30 <- interval_censoring(c(censor_based_titer_30_strain_A, censor_based_titer_30_s
```

To test the effect of censoring according to different variables: dependent and independent, we have four scenarios to consider:

1. The true scenario happening in human body: uncensored titer 0, uncensored titer 30
2. Introducing censoring (measurement deficieny) in independent var: censored titer 0, uncensored titer 30
3. Introducing censoring (measurement deficieny) in dependent var: uncensored titer 0, censored titer 30
4. Introducing censoring (measurement deficieny) in both var: censored titer 0, censored titer 30

However, scenario 1 can't be observed by the current titer measures, and always scenario 4 is the realistic situation.

Here, I am exploring all the 4 scenarios to get a sense of the amount of bias introduced by each variable (also will consider the other two scenarios mentioned earlier calculating the titer 30, so we have total 8 scenarios)

```
# fitting lme models
model_1 = run_lme(IDs, strain, titer_0, titer_30)
# model_2 = run_lme(IDs, strain, censored_titer_0, titer_30)
# model_3 = run_lme(IDs, strain, titer_0, censored_titer_30)
# model_4 = run_lme(IDs, strain, censored_titer_0, censored_titer_30)

# summary(model)
# par(mfrow = c(2,2))
# plot(model)
```

Post-model analysis

```
# Assigning estimates in a dataframe
df_base = c(rep(0,1))
estimates <- data.frame(intercept_A = df_base , intercept_B = df_base, slope_A = df_base, slope_B = df_

estimates[1,] <- fixef(model_1)

# Assessing confidence in parameters
CIs <- intervals(model_1, which = "fixed")  # calculates 95% CIs
CIs <- CIs$fixed[, c("lower", "upper")]

# Percentages of intercept and/or slope values lying within the estimated CIs
# percents_strain_A = c(0, 0, 0); percents_strain_B = c(0, 0, 0)
#
# percents_strain_A = calc_percent_parms_in_CI(percents_strain_A, intercept_strain_A, slope_strain_A, C
# percents_strain_B = calc_percent_parms_in_CI(percents_strain_B, intercept_strain_B, slope_strain_B, C
#
# percents_of_parameters_in_CIs <- data.frame(percents_strain_A * 100 / i, percents_strain_B * 100 / i)
# rownames(percents_of_parameters_in_CIs) <- c("intercept", "slope", "both")
# colnames(percents_of_parameters_in_CIs) <- c("strain A", "strain B")
# print(percents_of_parameters_in_CIs)

# bias
true_parameters <- c(intercept_strain_A, intercept_strain_B, slope_strain_A, slope_strain_B)
estimate_means <- colMeans(estimates)
bias <- abs(estimate_means - true_parameters)
normalized_bias <-  bias / true_parameters

# print(bias)
print(normalized_bias)
```

```
## intercept_A intercept_B     slope_A     slope_B
## 0.760006224 0.279725158 0.002129727 0.001775935
```

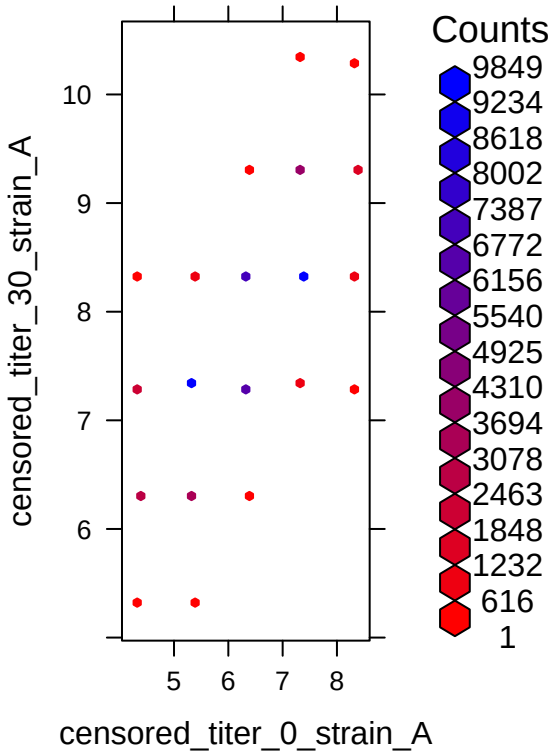
Generating several plots

```
# plot(titer_0_strain_A, titer_30_strain_A)
# plot(titer_0_strain_B, titer_30_strain_B)
# plot(censored_titer_0_strain_A, titer_30_strain_A)
# plot(censored_titer_0_strain_B, titer_30_strain_B)
# plot(censored_titer_0_strain_A, censored_titer_30_strain_A)
# plot(censored_titer_0_strain_B, censored_titer_30_strain_B)
```

```
### plot by strain
# xyplot(titer_30 ~ titer_0|strain, input_data)
# xyplot(censored_titer_30 ~ censored_titer_0|strain, censored_input_data)

### plot by individual
# xyplot(censored_titer_30 ~ censored_titer_0|IDs, censored_input_data)
# xyplot(titer_30 ~ titer_0|IDs, input_data)
```

```r
# create a hexbin plot
hexbinplot(censored_titer_30_strain_A ~ censored_titer_0_strain_A, colramp = function(n) colorRampPalet
```




```r
# hexbinplot(censored_titer_30_strain_B ~ censored_titer_0_strain_B, colramp = function(n) colorRampPal
```

```r
# plotting data (continuous and censored) of the two strains
# ggplot(data = input_data, aes(x = titer_0, y = titer_30, color = strain)) +
#   geom_point()
# ggplot(data = censored_input_data, aes(x = censored_titer_0, y = censored_titer_30, color = strain))
#   geom_point()
```