Project 1 Exploratory Analysis

Jack Cunningham

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.2     v tibble    3.2.1
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.0.4
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

Loading Data

```
data <- readxl::read_excel("Project2_S25_Data.xlsx")[,1:12]
```

```
New names:
* `` -> `...13`
* `` -> `...14`
* `` -> `...15`
* `` -> `...16`
* `` -> `...17`
* `` -> `...18`
* `` -> `...19`
* `` -> `...20`
* `` -> `...21`
```

```
data
```

```
# A tibble: 1,508 x 12
   Tick_ID Deer_ID Vaccine_Group Infestation Mortality Weight_Replete_Female(g~1
   <chr>     <dbl> <chr>         <chr>           <dbl>                      <dbl>
 1 792_01      792 P             PRE                 0                     0.0682
 2 792_02      792 P             PRE                 0                     0.178
 3 792_03      792 P             PRE                 0                     0.035
 4 792_04      792 P             PRE                 0                     0.0368
 5 792_05      792 P             PRE                 1                     0.0263
 6 792_06      792 P             PRE                 0                     0.195
 7 792_07      792 P             PRE                 1                     0.0648
 8 792_08      792 P             PRE                 0                     0.0607
 9 792_09      792 P             PRE                 1                     0.0689
10 792_10      792 P             PRE                 0                     0.181
# i 1,498 more rows
# i abbreviated name: 1: `Weight_Replete_Female(g)`
# i 6 more variables: `Weight_ Egg_Mass(g)` <chr>,
#   `Bloodmeal_Conversion(%)` <chr>, `%_Hatched` <chr>, Num_Eggs_laid <chr>,
#   Num_Larvae <chr>, Comments <chr>
```

Missing Data

```
na_count <- sapply(data, function(y) sum(length(which(is.na(y)))))
na_count
```

```
                Tick_ID                 Deer_ID           Vaccine_Group
                      0                       0                       0
             Infestation              Mortality Weight_Replete_Female(g)
                      0                       0                       0
    Weight_ Egg_Mass(g) Bloodmeal_Conversion(%)               %_Hatched
                      0                       0                       4
          Num_Eggs_laid              Num_Larvae                Comments
                      5                       6                     878
```

We can see that there does not appear to be a lot of missing data on first glance, but after looking at the excel we can see that there does appear to be many "N/A" values. I also filter out the Comments column, because after reviewing the excel it seems it was only populated when there were extreme circumstances.

```
data_no_coms <- data[,1:11]
na_count_manual <- sapply(data_no_coms, function(y) sum(length(which(y == "N/A"))))
na_count_manual
```

```
                Tick_ID                   Deer_ID            Vaccine_Group
                      0                         0                        0
             Infestation                Mortality Weight_Replete_Female(g)
                      0                         0                        0
    Weight_ Egg_Mass(g)  Bloodmeal_Conversion(%)                %_Hatched
                    552                       552                      560
          Num_Eggs_laid                Num_Larvae
                    557                       560
```

There appears to be a significant amount of data that is manually classified with "N/A". Lets see if they are often in the same row.

```
data$na_count <- apply(data_no_coms, 1, function(x) sum(x == "N/A"|is.na(x)))
table(data$na_count)
```

```
  0   1   2   3   5
942   1   4   9 552
```

We can see that almost always missing values are grouped in the same row.

When reviewing the excel document it appears that when the five values are all N/As its due to the tick dying. Let's see:

```
data |> group_by(Mortality, na_count) |>
  summarise(n = n())
```

```
`summarise()` has grouped output by 'Mortality'. You can override using the
`.groups` argument.
```

```
# A tibble: 7 x 3
# Groups:   Mortality [2]
  Mortality na_count     n
      <dbl>    <int> <int>
1         0        0   938
```

```
2         0        1     1
3         0        2     4
4         0        3     9
5         0        5     1
6         1        0     4
7         1        5   551
```

It appears this trend holds with the exception of 4 ticks who died but have no missing records. Lets look at those:

```
data |> filter(Mortality == 1 & na_count == 0)
```

```
# A tibble: 4 x 13
  Tick_ID Deer_ID Vaccine_Group Infestation Mortality `Weight_Replete_Female(g)`
  <chr>     <dbl> <chr>         <chr>           <dbl>                       <dbl>
1 800_14      800 P             PRE                 1                       0.258
2 803_02      803 P             PRE                 1                       0.252
3 805_34      805 P             PRE                 1                       0.148
4 805_35      805 P             PRE                 1                       0.270
# i 7 more variables: `Weight_ Egg_Mass(g)` <chr>,
#   `Bloodmeal_Conversion(%)` <chr>, `%_Hatched` <chr>, Num_Eggs_laid <chr>,
#   Num_Larvae <chr>, Comments <chr>, na_count <int>
```

Three out of the four "died during oviposition". That is a common comment in this data set. Lets see how many rows discuss a death during oviposition.

```
data |> filter(grepl('position|ovi&died', Comments, ignore.case =  TRUE))
```

```
# A tibble: 62 x 13
   Tick_ID Deer_ID Vaccine_Group Infestation Mortality Weight_Replete_Female(g~1
   <chr>     <dbl> <chr>         <chr>           <dbl>                       <dbl>
 1 793_05      793 P             PRE                 0                       0.189
 2 793_10      793 P             PRE                 0                       0.247
 3 793_13      793 P             PRE                 0                       0.205
 4 793_16      793 P             PRE                 0                       0.184
 5 793_17      793 P             PRE                 0                       0.187
 6 793_18      793 P             PRE                 0                       0.151
 7 793_24      793 P             PRE                 0                       0.194
 8 793_25      793 P             PRE                 0                       0.195
 9 800_15      800 P             PRE                 0                       0.241
```

```
10 800_29      800 P             PRE                 0                       0.157
# i 52 more rows
# i abbreviated name: 1: `Weight_Replete_Female(g)`
# i 7 more variables: `Weight_ Egg_Mass(g)` <chr>,
#   `Bloodmeal_Conversion(%)` <chr>, `%_Hatched` <chr>, Num_Eggs_laid <chr>,
#   Num_Larvae <chr>, Comments <chr>, na_count <int>
```

Oviposition

Oviposition is the process of laying or depositing eggs. I deem this as a middle ground between an outright death for a tick and the tick surviving. In an ideal world a vaccinated deer would leave a tick dead, in the worst world the tick would survive. Maybe when a tick dies during oviposition the number of hatched eggs is smaller.

```
data$ovi_position <- grepl('position|ovi', data$Comments, ignore.case = TRUE)
```

Redefining The Mortality Variable

Lets handle the pre-processing of the mortality variable. I create a "result" feature, it is set as "death", "oviposition death", "no death".

```
data$result <- case_when(
  data$ovi_position == TRUE & data$Mortality !=1 ~ "Oviposition Death",
  data$Mortality == 1 ~ "Death",
  data$Mortality == 0 ~ "Lives")
```

```
table(data$result)
```

```
           Death            Lives Oviposition Death
             555              889                64
```

Handling NAs

Now we have an idea of why we are getting NA values. They primarily come from situations where the tick dies and thus no outcome data is recorded. As seen below:

```
data |> filter(result == "Death") |>
  group_by(na_count) |>
  summarise(n = n())
```

```
# A tibble: 2 x 2
  na_count     n
     <int> <int>
1        0     4
2        5   551
```

There is still one pesky row, lets take a look at it:

```
data |> filter(result == "Death"& na_count == 0)
```

```
# A tibble: 4 x 15
  Tick_ID Deer_ID Vaccine_Group Infestation Mortality `Weight_Replete_Female(g)`
  <chr>     <dbl> <chr>         <chr>           <dbl>                       <dbl>
1 800_14      800 P             PRE                 1                       0.258
2 803_02      803 P             PRE                 1                       0.252
3 805_34      805 P             PRE                 1                       0.148
4 805_35      805 P             PRE                 1                       0.270
# i 9 more variables: `Weight_ Egg_Mass(g)` <chr>,
#   `Bloodmeal_Conversion(%)` <chr>, `%_Hatched` <chr>, Num_Eggs_laid <chr>,
#   Num_Larvae <chr>, Comments <chr>, na_count <int>, ovi_position <lgl>,
#   result <chr>
```

Based on the information here, I think this is best categorized as an "Oviposition death".

```
data$result <- ifelse(data$result == "Death" & data$na_count == 0,
                      "Oviposition Death",data$result)
```

Now lets set those NAs as zero:

```
data$`Weight_ Egg_Mass(g)` <- ifelse(data$result == "Death", 0, data$`Weight_ Egg_Mass(g)`
data$`Bloodmeal_Conversion(%)` <- ifelse(data$result == "Death", 0, data$`Bloodmeal_Conver
data$`%_Hatched` <- ifelse(data$result == "Death", 0, data$`%_Hatched`)
data$Num_Eggs_laid <- ifelse(data$result == "Death", 0, data$Num_Eggs_laid)
data$Num_Larvae <- ifelse(data$result == "Death", 0, data$Num_Larvae)
```

Now lets handle the remaining NAs. Lets first analyze those rows before deciding how to proceed:

```r
data |>
  filter(na_count %in%c(1,2,3,4))
```

```
# A tibble: 14 x 15
   Tick_ID Deer_ID Vaccine_Group Infestation Mortality Weight_Replete_Female(g~1
   <chr>     <dbl> <chr>         <chr>           <dbl>                      <dbl>
 1 800_04      800 P             PRE                 0                      0.197
 2 804_05      804 P             PRE                 0                      0.206
 3 804_17      804 P             PRE                 0                      0.203
 4 804_45      804 P             PRE                 0                      0.236
 5 793_57      793 C             FIRST               0                      0.257
 6 792_138     792 C             THIRD               0                      0.266
 7 803_132     803 C             THIRD               0                      0.201
 8 803_153     803 C             THIRD               0                      0.159
 9 801_47      801 L             FIRST               0                      0.196
10 801_55      801 L             FIRST               0                      0.321
11 801_70      801 L             FIRST               0                      0.224
12 801_79      801 L             FIRST               0                      0.159
13 801_92      801 L             SECOND              0                      0.315
14 790_59      790 H             FIRST               0                      0.245
# i abbreviated name: 1: `Weight_Replete_Female(g)`
# i 9 more variables: `Weight_ Egg_Mass(g)` <chr>,
#   `Bloodmeal_Conversion(%)` <chr>, `%_Hatched` <chr>, Num_Eggs_laid <chr>,
#   Num_Larvae <chr>, Comments <chr>, na_count <int>, ovi_position <lgl>,
#   result <chr>
```

Since there are only 15 rows I feel its worth taking a look at each situation.

The first situation is “dram spilled no hatch data collected”:

```r
data |>
  filter(na_count %in%c(1,2,3,4) & grepl('Dram', data$Comments, ignore.case = TRUE))
```

```
# A tibble: 4 x 15
  Tick_ID Deer_ID Vaccine_Group Infestation Mortality `Weight_Replete_Female(g)`
  <chr>     <dbl> <chr>         <chr>           <dbl>                      <dbl>
1 793_57      793 C             FIRST               0                      0.257
2 792_138     792 C             THIRD               0                      0.266
3 803_132     803 C             THIRD               0                      0.201
4 790_59      790 H             FIRST               0                      0.245
# i 9 more variables: `Weight_ Egg_Mass(g)` <chr>,
```

```
#   `Bloodmeal_Conversion(%)` <chr>, `%_Hatched` <chr>, Num_Eggs_laid <chr>,
#   Num_Larvae <chr>, Comments <chr>, na_count <int>, ovi_position <lgl>,
#   result <chr>
```

In this case I think an acceptable approach is to use median imputation for the missing values from the group of surviving ticks.

```
hatched_data_lives = data |>
  filter(result == "Lives") |>
  select(Num_Eggs_laid, Num_Larvae)
head(hatched_data_lives)
```

```
# A tibble: 6 x 2
  Num_Eggs_laid Num_Larvae
  <chr>         <chr>
1 636           452
2 1540          1344
3 108           4
4 128           8
5 772           12
6 480           212
```

```
hatched_data_median_lives = lapply(hatched_data_lives, function(x) median(as.numeric(x), n
```

```
Warning in median(as.numeric(x), na.rm = TRUE): NAs introduced by coercion
Warning in median(as.numeric(x), na.rm = TRUE): NAs introduced by coercion
```

```
hatched_data_median_lives
```

```
$Num_Eggs_laid
[1] 2152

$Num_Larvae
[1] 1514
```

```
hatched_data_median_hatched_percent = hatched_data_median_lives$Num_Larvae/hatched_data_me
```

Now we substitute in these values into the missing values from earlier:

```
data <- data |> mutate(Num_Eggs_laid = ifelse(na_count == "3" & grepl('Dram', data$Comment
  mutate(Num_Larvae = ifelse(na_count == "3" & grepl('Dram', data$Comments, ignore.case =
  mutate(`%_Hatched` = ifelse(na_count == "3" & grepl('Dram', data$Comments, ignore.case =
```

Let's see how many missing values still exist:

```
data$na_count_2 <- apply(data[,-12], 1, function(x) sum(x == "N/A"|is.na(x)))
table(data$na_count_2)
```

```
   0    1    2    3    5
1497    1    4    5    1
```

```
data |> filter(na_count_2 != 0) |>
  View()
```

% Hatched

Lets try to understand the % hatched variable. It appears it is a computation of num_larvae/num_eggs_laid.

```
data <- data |>
  mutate(percent_hatched = ifelse(Num_Eggs_laid != "0",100*as.numeric(Num_Larvae)/as.numer
```

```
Warning: There were 2 warnings in `mutate()`.
The first warning was:
i In argument: `percent_hatched = ifelse(...)`.
Caused by warning in `ifelse()`:
! NAs introduced by coercion
i Run `dplyr::last_dplyr_warnings()` to see the 1 remaining warning.
```

I want to see if these calculations always match up:

```
data |>
  filter(percent_hatched != 0) |>
  mutate(percent_compare = abs(percent_hatched - as.numeric(`%_Hatched`))) |>
  filter(percent_compare > .05) |>
  select(Num_Eggs_laid,Num_Larvae,percent_hatched,`%_Hatched`,percent_compare)
```

```
# A tibble: 43 x 5
   Num_Eggs_laid Num_Larvae percent_hatched `%_Hatched`        percent_compare
   <chr>         <chr>                <dbl> <chr>                        <dbl>
 1 1608          1469                  91.4 93.03                         1.67
 2 1992          1992                 100   97.59                         2.41
 3 572           412                   72.0 27.97                        44.1
 4 864           708                   81.9 84.26                         2.32
 5 2116          1768                  83.6 85.55                         2.00
 6 2636          3592                 136.  98.33                        37.9
 7 988           800                   81.0 80.900000000000006           0.0717
 8 1820          804                   44.2 41.18                         3.00
 9 2152          1514                  70.4 0.703531598513011            69.6
10 6204          2232                  36.0 69.66                        33.7
# i 33 more rows
```

We can see that generally speaking the calculated formula is in line with %_Hatched. There are some instances of large differences, but I'd rather trust the count of num larvae than rely on the percentage which could be a manual error.

With this in mind, there are some instances where the %hatched was estimate at 2 weeks instead of waiting. And it seems those values were not recorded. So I will take the median of our percent hatched variable and impute them into those values, and then calculate the num larvae.

```
percent_hatched_median_lives = median(data[data$percent_hatched !=0,]$percent_hatched,na.r
data <- data |> mutate(percent_hatched = ifelse(Num_Eggs_laid != "0" & na_count_2 == 2, pe
  mutate(Num_Larvae = ifelse(Num_Eggs_laid != "0" & na_count_2 == 2,
                             round(as.numeric(Num_Eggs_laid)*percent_hatched_median_lives/
                             Num_Larvae))
```

```
Warning: There was 1 warning in `mutate()`.
i In argument: `Num_Larvae = ifelse(...)`.
Caused by warning in `ifelse()`:
! NAs introduced by coercion
```

Lets now see the remaining NA values:

```
#data <- data |> select(-`%_Hatched`)
data$na_count_3 <- apply(data[,-c(9,12)], 1, function(x) sum(x == "N/A"|is.na(x)))
table(data$na_count_3)
```

```
   0    2    3    5
1500    1    6    1
```

```
data |>
  filter(na_count_3 != 0)
```

```
# A tibble: 8 x 18
  Tick_ID Deer_ID Vaccine_Group Infestation Mortality `Weight_Replete_Female(g)`
  <chr>     <dbl> <chr>         <chr>           <dbl>                       <dbl>
1 800_15      800 P             PRE                 0                       0.241
2 804_05      804 P             PRE                 0                       0.206
3 803_153     803 C             THIRD               0                       0.159
4 801_47      801 L             FIRST               0                       0.196
5 801_55      801 L             FIRST               0                       0.321
6 801_70      801 L             FIRST               0                       0.224
7 801_79      801 L             FIRST               0                       0.159
8 801_92      801 L             SECOND              0                       0.315
# i 12 more variables: `Weight_ Egg_Mass(g)` <chr>,
#   `Bloodmeal_Conversion(%)` <chr>, `%_Hatched` <chr>, Num_Eggs_laid <chr>,
#   Num_Larvae <chr>, Comments <chr>, na_count <int>, ovi_position <lgl>,
#   result <chr>, na_count_2 <int>, percent_hatched <dbl>, na_count_3 <int>
```

Since there is no mention of measurement error on the remaining missings that have no data on num eggs laid and num larve and no recorded % Hatched, I will set them to zero.

```
data <- data |>
  mutate(mark = (Num_Eggs_laid == "N/A"|is.na(Num_Eggs_laid))&(Num_Larvae == "N/A"|is.na(N
  mutate(Num_Eggs_laid = ifelse(mark,0,Num_Eggs_laid)) |>
  mutate(Num_Larvae = ifelse(mark,0,Num_Larvae)) |>
  mutate(percent_hatched = ifelse(mark,0,percent_hatched)) |>
  select(-mark)
```

Lets do a final check on NAs:

```
data$na_count_4 <- apply(data[,c(-9,-12)], 1, function(x) sum(x == "N/A"|is.na(x)))
table(data$na_count_4)
```

```
   0    2    3
1505    2    1
```

Looking at those rows:

```
data |> filter(na_count_4 != 0)
```

```
# A tibble: 3 x 19
  Tick_ID Deer_ID Vaccine_Group Infestation Mortality `Weight_Replete_Female(g)`
  <chr>     <dbl> <chr>         <chr>           <dbl>                       <dbl>
1 800_15      800 P             PRE                 0                       0.241
2 803_153     803 C             THIRD               0                       0.159
3 801_92      801 L             SECOND              0                       0.315
# i 13 more variables: `Weight_ Egg_Mass(g)` <chr>,
#   `Bloodmeal_Conversion(%)` <chr>, `%_Hatched` <chr>, Num_Eggs_laid <chr>,
#   Num_Larvae <chr>, Comments <chr>, na_count <int>, ovi_position <lgl>,
#   result <chr>, na_count_2 <int>, percent_hatched <dbl>, na_count_3 <int>,
#   na_count_4 <int>
```

There is one situation where we have % Hatched and Num Eggs Laid, but no record on larvae hatched and percent hatched. Lets use a computation to fix that:

```
data <- data |>
  mutate(mark = Tick_ID == "801_92") |>
  mutate(percent_hatched = ifelse(mark, as.numeric(`%_Hatched`),percent_hatched)) |>
  mutate(Num_Larvae = ifelse(mark,round(as.numeric(Num_Eggs_laid)*percent_hatched/100),
                             Num_Larvae))
```

```
Warning: There was 1 warning in `mutate()`.
i In argument: `percent_hatched = ifelse(mark, as.numeric(`%_Hatched`),
  percent_hatched)`.
Caused by warning in `ifelse()`:
! NAs introduced by coercion
```

Lets see whats left:

```
data$na_count_5 <- apply(data[,c(-9,-12)], 1, function(x) sum(x == "N/A"|is.na(x)))
table(data$na_count_5)
```

```
   0    2    3
1506    1    1
```

```
  data |>
    filter(na_count_5!=0)
```

```
# A tibble: 2 x 21
  Tick_ID Deer_ID Vaccine_Group Infestation Mortality `Weight_Replete_Female(g)`
  <chr>     <dbl> <chr>         <chr>           <dbl>                       <dbl>
1 800_15      800 P             PRE                 0                       0.241
2 803_153     803 C             THIRD               0                       0.159
# i 15 more variables: `Weight_ Egg_Mass(g)` <chr>,
#   `Bloodmeal_Conversion(%)` <chr>, `%_Hatched` <chr>, Num_Eggs_laid <chr>,
#   Num_Larvae <chr>, Comments <chr>, na_count <int>, ovi_position <lgl>,
#   result <chr>, na_count_2 <int>, percent_hatched <dbl>, na_count_3 <int>,
#   na_count_4 <int>, mark <lgl>, na_count_5 <int>
```

For tick 800_15 it appears this should be treated as a usual tick death, there is no evidence of eggs being laid.

```
  data <- data |>
    mutate(mark = Tick_ID == "800_15") |>
    mutate(`Weight_ Egg_Mass(g)` = ifelse(mark,0,`Weight_ Egg_Mass(g)`)) |>
    mutate(`Bloodmeal_Conversion(%)` = ifelse(mark,0,`Bloodmeal_Conversion(%)`))
```

The final NA check:

```
  data$na_count_6 <- apply(data[,c(-9,-12)], 1, function(x) sum(x == "N/A"|is.na(x)))
  table(data$na_count_6)
```

```
   0    3
1507    1
```

```
  data |>
    filter(na_count_6 != 0)
```

```
# A tibble: 1 x 22
  Tick_ID Deer_ID Vaccine_Group Infestation Mortality `Weight_Replete_Female(g)`
  <chr>     <dbl> <chr>         <chr>           <dbl>                       <dbl>
1 803_153     803 C             THIRD               0                       0.159
# i 16 more variables: `Weight_ Egg_Mass(g)` <chr>,
```

```
#   `Bloodmeal_Conversion(%)` <chr>, `%_Hatched` <chr>, Num_Eggs_laid <chr>,
#   Num_Larvae <chr>, Comments <chr>, na_count <int>, ovi_position <lgl>,
#   result <chr>, na_count_2 <int>, percent_hatched <dbl>, na_count_3 <int>,
#   na_count_4 <int>, mark <lgl>, na_count_5 <int>, na_count_6 <int>
```

In this case we have evidence of hatching, but we don't know the number of eggs laid or larvae. I choose a median imputation for num eggs laid given the tick lives, then computing the num larvae/

```
med_imput_data = data |> filter(Mortality == 0) |> select(Num_Eggs_laid)
med_imput = as.numeric(median(as.numeric(med_imput_data$Num_Eggs_laid),na.rm = TRUE))
```

```
data <- data |>
  mutate(Num_Eggs_laid = ifelse(na_count_6 != 0, med_imput, Num_Eggs_laid)) |>
  mutate(percent_hatched = ifelse(na_count_6 != 0,as.numeric(`%_Hatched`), percent_hatched
  mutate(Num_Larvae = ifelse(na_count_6 != 0, percent_hatched/100 * as.numeric(Num_Eggs_la
```

```
Warning: There was 1 warning in `mutate()`.
i In argument: `percent_hatched = ifelse(na_count_6 != 0,
  as.numeric(`%_Hatched`), percent_hatched)`.
Caused by warning in `ifelse()`:
! NAs introduced by coercion
```

Final NA check:

```
data$na_count_7 <- apply(data[,c(-9,-12)], 1, function(x) sum(x == "N/A"|is.na(x)))
table(data$na_count_7)
```

```
   0
1508
```

Great, there are no more missings. Lets convert our data to the correct types:

Data Conversion

First lets drop columns that won't be needed moving forward.

```
colnames(data)
```

```
 [1] "Tick_ID"                  "Deer_ID"
 [3] "Vaccine_Group"            "Infestation"
 [5] "Mortality"                "Weight_Replete_Female(g)"
 [7] "Weight_ Egg_Mass(g)"      "Bloodmeal_Conversion(%)"
 [9] "%_Hatched"                "Num_Eggs_laid"
[11] "Num_Larvae"               "Comments"
[13] "na_count"                 "ovi_position"
[15] "result"                   "na_count_2"
[17] "percent_hatched"          "na_count_3"
[19] "na_count_4"               "mark"
[21] "na_count_5"               "na_count_6"
[23] "na_count_7"
```

```
data <- data[,-c(9,13,16,18,19,20,21,22,23)]
```

Now we need to convert our data into the correct types per variable:

```
colnames(data)
```

```
 [1] "Tick_ID"                  "Deer_ID"
 [3] "Vaccine_Group"            "Infestation"
 [5] "Mortality"                "Weight_Replete_Female(g)"
 [7] "Weight_ Egg_Mass(g)"      "Bloodmeal_Conversion(%)"
 [9] "Num_Eggs_laid"            "Num_Larvae"
[11] "Comments"                 "ovi_position"
[13] "result"                   "percent_hatched"
```

```
numeric_variables = colnames(data)[c(6,7,8,9,10,14)]
numeric_variables
```

```
[1] "Weight_Replete_Female(g)" "Weight_ Egg_Mass(g)"
[3] "Bloodmeal_Conversion(%)"  "Num_Eggs_laid"
[5] "Num_Larvae"               "percent_hatched"
```

```
factor_variables = colnames(data)[c(2,3,4,5,13)]
factor_variables
```

```
[1] "Deer_ID"       "Vaccine_Group" "Infestation"   "Mortality"
[5] "result"
```

```
char_variables = colnames(data)[c(1,11)]
```

```
data_processed = data |>
  mutate(across(all_of(numeric_variables),as.numeric)) |>
  mutate(across(all_of(factor_variables),as.factor)) |>
  mutate(across(all_of(char_variables),as.character)) |>
  mutate(Comments = ifelse(is.na(Comments),"",Comments))
```

lets see if there are NAs that occurred here:

```
anyNA(data_processed)
```

```
[1] FALSE
```

Fixing when Number Larvae is greater Num Eggs Laid

```
data_processed = data_processed |>
  mutate(mark = Num_Larvae > Num_Eggs_laid) |>
  mutate(correct_larve = ifelse(mark,Num_Eggs_laid,Num_Larvae)) |>
  mutate(correct_eggs = ifelse(mark,Num_Larvae,Num_Eggs_laid)) |>
  mutate(Num_Larvae = correct_larve) |>
  mutate(Num_Eggs_laid = correct_eggs) |>
  mutate(percent_hatched = ifelse(mark,Num_Larvae/Num_Eggs_laid,percent_hatched)) |> selec
```

Leveling Factors

```
data_processed$Infestation <- factor(data_processed$Infestation,
                                     levels = c("PRE","FIRST","SECOND","THIRD"))
data_processed$result <- factor(data_processed$result,
                                levels = c("Death","Oviposition Death","Lives"))
data_processed$Vaccine_Group <- factor(data_processed$Vaccine_Group,
                                       levels = c("P","C","L","H"))
```

Saving down processed data:

```r
save(data_processed, file = "data_processed.RData")
```