Identification of eclosion time using R

Jun Ishigohoka

2026-02-11

This PDF was created from a notebook written in Rmarkdown `src/eclosion_detection.Rmd` using `rmarkdown::render`. The source notebook can be opened in `RStudio` and can be edited for your analysis. For tutorials on Rmarkdown, please see https://rmarkdown.rstudio.com/lesson-1.html.

1 Introduction

We will detect eclosion time using `R`. Before eclosion, flies in pupal cases are (or more acurately, they become) pigmented. After eclosion the pupal case is empty and becomes semi-transparent. Eclosion events will be detected based on the change in pixel intensity between time points.

2 Pre-requisites

- Install `R` and `RStudio`
- Install the following `R` libraries
  - `ggplot2`: For plotting
  - `dplyr`: To "pipe" commands one after the other
  - `lubridate`: To deal with date and time
  - `zoo`: Required for rolling median computation
- Image analysis in `FIJI` (`image_analysis.pdf`)
  - Pixel intensity over time of pupae
  - Centroid coordinates of pupae exported as csv
  - Coordinates of wells of the 96-well plate as csv

3 Start R and import libraries

```
library(ggplot2)
library(dplyr)
library(lubridate)
```

Load custom functions.

```
source("scripts/detect_eclosion.R")
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

4 Match well and FIJI ID

4.1 Overview

We will assign wells (A1, B1,...) to each pupa (ROI). We do this by

1. For each ROI, compute the distance to all wells.
2. For each ROI, assign the closest well.

4.2 Prepare ROI coordinates

Read coordinates of ROIs (i.e. pupae)

```
roi_coords <- read.csv("test_data/rec_016_2025_11_12_09_40_resized_threshold_rois_edited_coords.csv")
```

Check the first few rows

```
head(roi_coords)
```

```
##   X.1                                  Label        X       Y
## 1   1 2025_11_10_15_30_resized.jpg:0331-1139 1139.322 331.089
## 2   2 2025_11_10_15_30_resized.jpg:0352-1194 1193.967 352.600
## 3   3 2025_11_10_15_30_resized.jpg:0353-1331 1330.565 353.318
## 4   4 2025_11_10_15_30_resized.jpg:0358-1448 1448.600 358.050
## 5   5 2025_11_10_15_30_resized.jpg:0361-1384 1383.935 361.457
## 6   6 2025_11_10_15_30_resized.jpg:0364-1286 1285.557 364.270
```

The ID of ROIs is included in `Label`. Extract the ID from the `Label` column (e.g. 0331-1139).

```
roi_coords <- roi_coords %>%
  # Make a new column id by substituting all string (.*)
  # of column Label up to pattern ":" with ""
  mutate(id = gsub(".*:", "", Label)) %>%
  # Only output these columns
  select(id, X, Y)
```

4.3 Prepare coordinates of the 96-well plate.

Read coordicates of wells of the 96-well plate.

```
well_coords <- read.csv("test_data/rec_016_plate_1_coords.csv")
```

Check the first few rows.

```
head(well_coords)
```

```
##   X.1                                  Label        X       Y
## 1   1 2025_11_10_15_30_resized.jpg:0651-1275 1438.167 966.833
## 2   2 2025_11_10_15_30_resized.jpg:0651-1275 1382.167 957.500
## 3   3 2025_11_10_15_30_resized.jpg:0651-1275 1330.833 949.500
## 4   4 2025_11_10_15_30_resized.jpg:0651-1275 1276.000 945.000
## 5   5 2025_11_10_15_30_resized.jpg:0651-1275 1218.000 936.500
## 6   6 2025_11_10_15_30_resized.jpg:0651-1275 1166.000 927.500
```

This data frame does not contain well information. When you made these coordinates in `FIJI`, you followed A1, B1, C1, ..., G11, G12. Make these strings.

```
# make an outer product of the first 8 LETTERS and first 12 integers,
# and make it into a vector
```

```
wells <- c(outer(LETTERS[1:8],  1:12, paste0))
wells
```

```
##  [1] "A1"  "B1"  "C1"  "D1"  "E1"  "F1"  "G1"  "H1"  "A2"  "B2"  "C2"  "D2"
## [13] "E2"  "F2"  "G2"  "H2"  "A3"  "B3"  "C3"  "D3"  "E3"  "F3"  "G3"  "H3"
## [25] "A4"  "B4"  "C4"  "D4"  "E4"  "F4"  "G4"  "H4"  "A5"  "B5"  "C5"  "D5"
## [37] "E5"  "F5"  "G5"  "H5"  "A6"  "B6"  "C6"  "D6"  "E6"  "F6"  "G6"  "H6"
## [49] "A7"  "B7"  "C7"  "D7"  "E7"  "F7"  "G7"  "H7"  "A8"  "B8"  "C8"  "D8"
## [61] "E8"  "F8"  "G8"  "H8"  "A9"  "B9"  "C9"  "D9"  "E9"  "F9"  "G9"  "H9"
## [73] "A10" "B10" "C10" "D10" "E10" "F10" "G10" "H10" "A11" "B11" "C11" "D11"
## [85] "E11" "F11" "G11" "H11" "A12" "B12" "C12" "D12" "E12" "F12" "G12" "H12"
```

Because `well_coords` is sorted the same way (A1, B1, ..., G12, H12), we can bind this to `well_coords` as a new column.

```
well_coords <- well_coords %>%
  mutate(well = wells) %>% # make a new column well which has values in vector wells
  select(well, X, Y) # Only keep these columns
```

Confirm that a column `well` was added.

```
head(well_coords)
```

```
##   well        X       Y
## 1   A1 1438.167 966.833
## 2   B1 1382.167 957.500
## 3   C1 1330.833 949.500
## 4   D1 1276.000 945.000
## 5   E1 1218.000 936.500
## 6   F1 1166.000 927.500
```

Now, compute the distance between ROIs (pupae) and well centroids, then the closest well to each pupa.

```
          # Make all possible combinations between id and well
id_well <- expand.grid(id = roi_coords$id, well = well_coords$well) %>%
  mutate(
    # Get X and Y of the ROI
    x_roi = roi_coords$X[match(id, roi_coords$id)],
    y_roi = roi_coords$Y[match(id, roi_coords$id)],
    # Get X and Y of the well
    x_well = well_coords$X[match(well, well_coords$well)],
    y_well = well_coords$Y[match(well, well_coords$well)],
    # Compute distance
    D = sqrt((x_roi - x_well)^2 + (y_roi - y_well)^2)
  ) %>%
  # Do the following operation for each id
  group_by(id) %>%
  # For each id, keep only the row with the minimum D
  filter(D == min(D))
```

Check if one well has at maximum 1 ROI

```
table(id_well$well)
```

```
##
##  A1  B1  C1  D1  E1  F1  G1  H1  A2  B2  C2  D2  E2  F2  G2  H2  A3  B3  C3  D3
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  E3  F3  G3  H3  A4  B4  C4  D4  E4  F4  G4  H4  A5  B5  C5  D5  E5  F5  G5  H5
```

```
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  A6  B6  C6  D6  E6  F6  G6  H6  A7  B7  C7  D7  E7  F7  G7  H7  A8  B8  C8  D8
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  E8  F8  G8  H8  A9  B9  C9  D9  E9  F9  G9  H9 A10 B10 C10 D10 E10 F10 G10 H10
##   1   1   1   1   1   1   1   1   0   1   1   1   1   1   1   1   1   1   0   1
## A11 B11 C11 D11 E11 F11 G11 H11 A12 B12 C12 D12 E12 F12 G12 H12
##   1   1   1   1   1   1   1   1   1   1   1   1   1   0   1   1
```

4.4 Troubleshoot

If there are wells with more than 1 ROIs, identify the id of the ROIs. To get the id of e.g. H12

```
id_well %>%
  filter(well == "H12") %>%
  select(well,id)
```

```
## # A tibble: 1 x 2
## # Groups:   id [1]
##   well  id
##   <fct> <fct>
## 1 H12   0331-1139
```

Open the reference image and ROIs in `FIJI`, and check the ROIs based on the id to find out what is wrong.

5 Process pixel intensity data

5.1 Overview

We will apply the following heuristic to detect putative eclosion time:

- If the magnitude of difference in pixel intensity (after - before) is the largest (the most negative value) and
- if this difference is statistically significant

then, the time-point is a candidate eclosion time.

This heuristic is implemented in a custom function `get_ecl_time`.

In reality, instead of using the actual pixel intensity (median pixel intensity within the ROI), we will use rolling median of this intensity over 5 time-points around the focal time points. This is to reduce the noise in measurement. This rolling median computation as well as difference between before and after are implemented in another custom function `comp_roll_med`.

5.2 Preparation of data

Read the formatted data

```
d <- read.csv("test_data/rec_016_plate_1_results_formatted.csv")
d$datetime <- as.POSIXct(paste(d$date, d$time),  format = "%Y-%m-%d %H:%M", tz = "CET")
```

Check the first few rows

```
head(d)
```

```
##         date     time        id Area    Mean StdDev Mode Min Max Median
## 1 2025-11-10 15:30:00 0361-1384   69  60.174 25.273   71   0  97     68
## 2 2025-11-10 15:30:00 0455-1269   90 104.133 41.196    0   0 163    118
## 3 2025-11-10 15:30:00 0634-1429   89 102.247 30.163  105   0 148    110
## 4 2025-11-10 15:30:00 0372-1129   82  96.720 40.863  132   0 145    110
```

```
## 5 2025-11-10 15:30:00 0547-1144   86 106.988 48.002    0   0 186    121
## 6 2025-11-10 15:30:00 0479-1477   70  92.971 25.799  103   3 121    103
##              datetime
## 1 2025-11-10 15:30:00
## 2 2025-11-10 15:30:00
## 3 2025-11-10 15:30:00
## 4 2025-11-10 15:30:00
## 5 2025-11-10 15:30:00
## 6 2025-11-10 15:30:00
```

The column `Median` is the median pixel intensity of the ROI.

Merge data with the well position.

```
d <- merge(d, id_well, by = "id") %>%
  arrange(datetime, id) # Sort by date time then by id
```

Compute rolling median of Median pixel intensity and the difference in rolling median before and after each time point using `comp_roll_med`. `k` is the number of time points to use for rolling median. `width` is the number of time points to define "before" and "after". Because images were taken every 10 min, $24 * 6$ should specify 1 day. (Not exactly, though, because we remove some timepoints in image pre-processing)

```
d <- comp_roll_med(data = d, k = 5, width = 24 * 6)
```

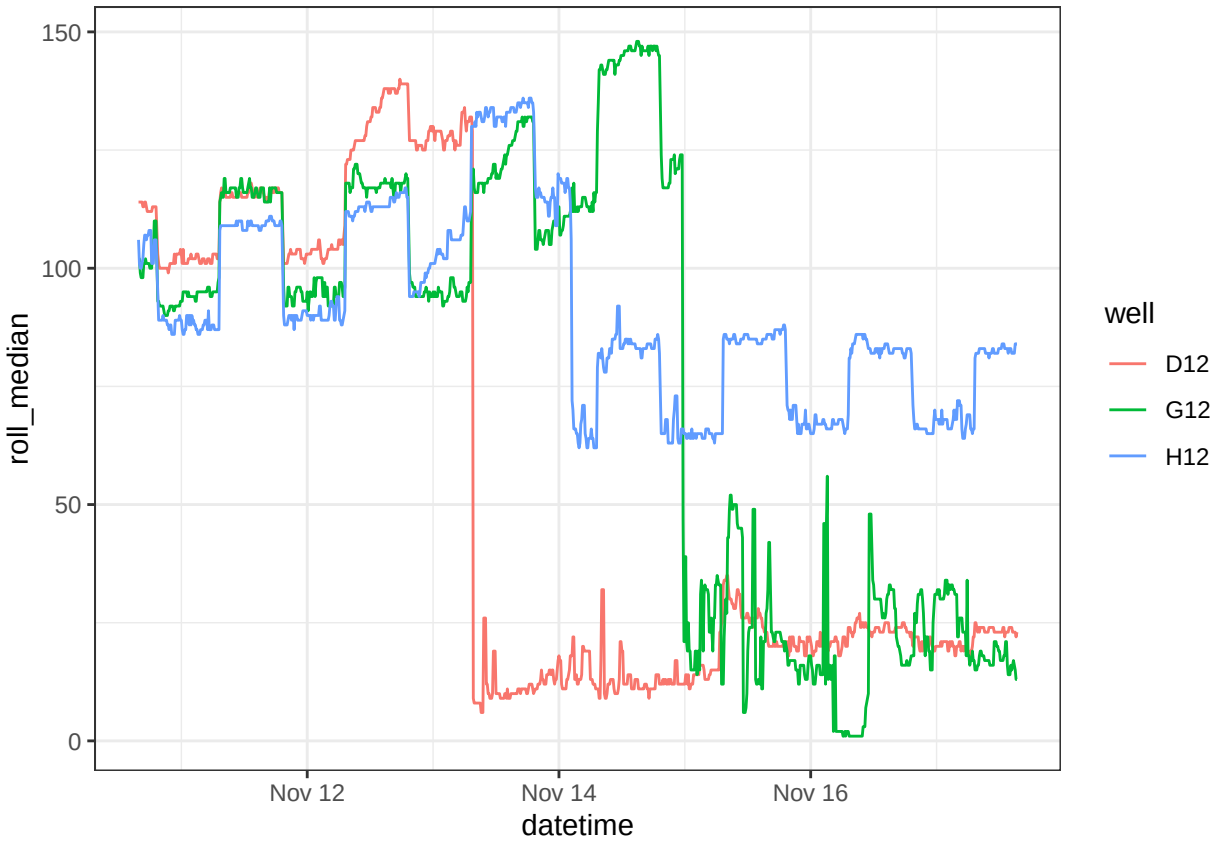
Confirm that columns `roll_median`, `med_before`, `med_after`, `diff_median` are added.

```
d %>%
  head %>%
  select(id, well, datetime, roll_median, med_before, med_after, diff_medians)
```

```
## # A tibble: 6 x 7
##   id     well  datetime            roll_median med_before med_after diff_medians
##   <chr>  <fct> <dttm>                    <dbl>      <dbl>     <dbl>        <dbl>
## 1 0331-~ H12   2025-11-10 15:50:00         106         NA      98.0           NA
## 2 0352-~ G12   2025-11-10 15:50:00         102         NA     103.            NA
## 3 0353-~ D12   2025-11-10 15:50:00         114         NA     108.            NA
## 4 0358-~ B12   2025-11-10 15:50:00          96         NA      99.2           NA
## 5 0361-~ C12   2025-11-10 15:50:00          71         NA      93.5           NA
## 6 0364-~ E12   2025-11-10 15:50:00         113         NA     104.            NA
```

Confirm that there is a timepoint at which `roll_median` drops visibly, which should correspond to the timing of eclosion.

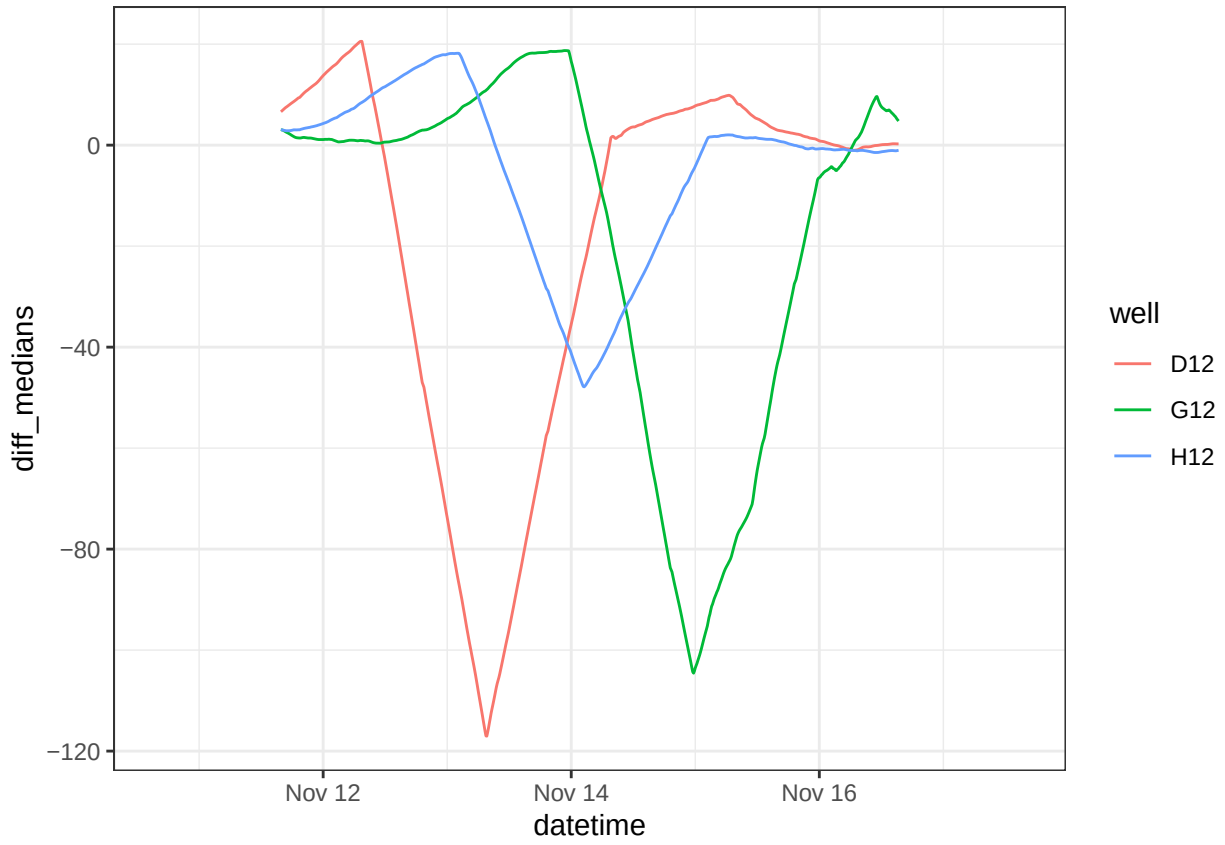
```
ggplot(data = subset(d, id %in% unique(d$id)[1:3]),
       mapping = aes(x = datetime,
                     y = roll_median ,
                     colour = well))+
        geom_line() +
        scale_x_datetime() +
        theme_bw()
```



This is usually the timepoint at which `diff_medians` is minimum.

```
ggplot(data = subset(d, id %in% unique(d$id)[1:3]),
       mapping = aes(x = datetime,
                     y = diff_medians ,
                     group = well,
                     colour = well))+
        geom_line() +
        scale_x_datetime() +
        theme_bw()
```

```
## Warning: Removed 858 rows containing missing values or values outside the scale
## range (`geom_line()`).
```



6 Detect putative eclosion time

For each id, identify putative eclosion time using `get_ecl_time`. `width` should be the same as the `width` used in `comp_roll_med`. `tz` is the time zone which output is formatted. To avoid confusion of the summer time, specify the GMT + 1, which is confusingly, `Etc/UTC-1`

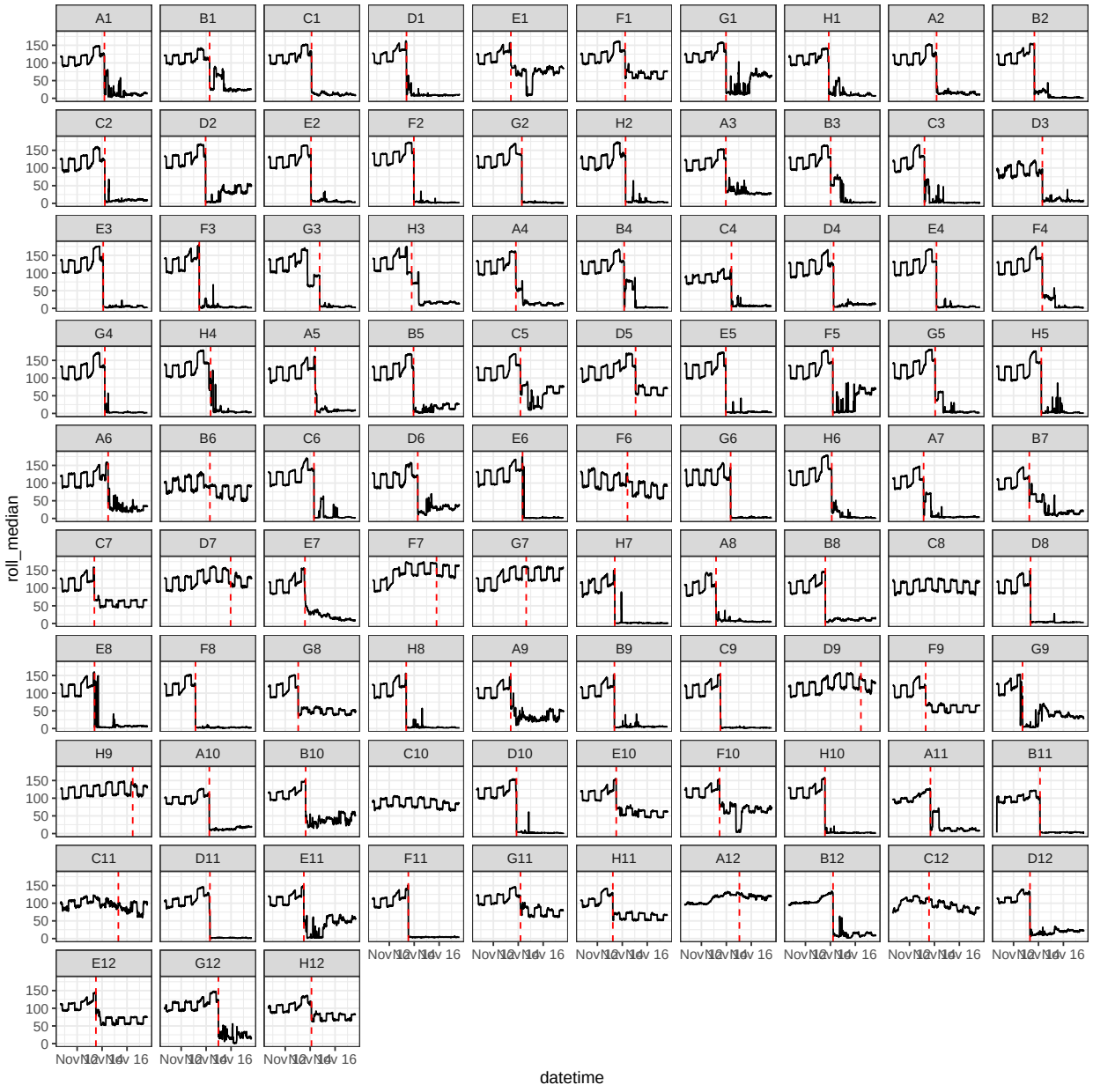
```
res <- get_ecl_time(data = d, width = 24 * 6, tz = "Etc/UTC-1")
```

7 Check

For each well, plot the rolling median pixel intensity over time with detected eclosion time.

```
merge(d, res) %>%
  ggplot(mapping = aes(x = datetime, y = roll_median))+
    geom_line() +
    geom_vline(aes(xintercept = ecl_time), linetype = 2, col = "red") +
    scale_x_datetime() +
    facet_wrap(.~well)+
    theme_bw()
```

```
## Warning: Removed 1998 rows containing missing values or values outside the
## scale range (`geom_vline()`).
```



The detected eclosion time (vertical red lines) seems wrong for B6, F6, D7, F7, G7, D9, H9, A12.

If you check the images in FIJI, pupae were not visible in B6 and F6. You can also confirm in images that eclosion did not happen in D7, F7, G7, D9, H9.

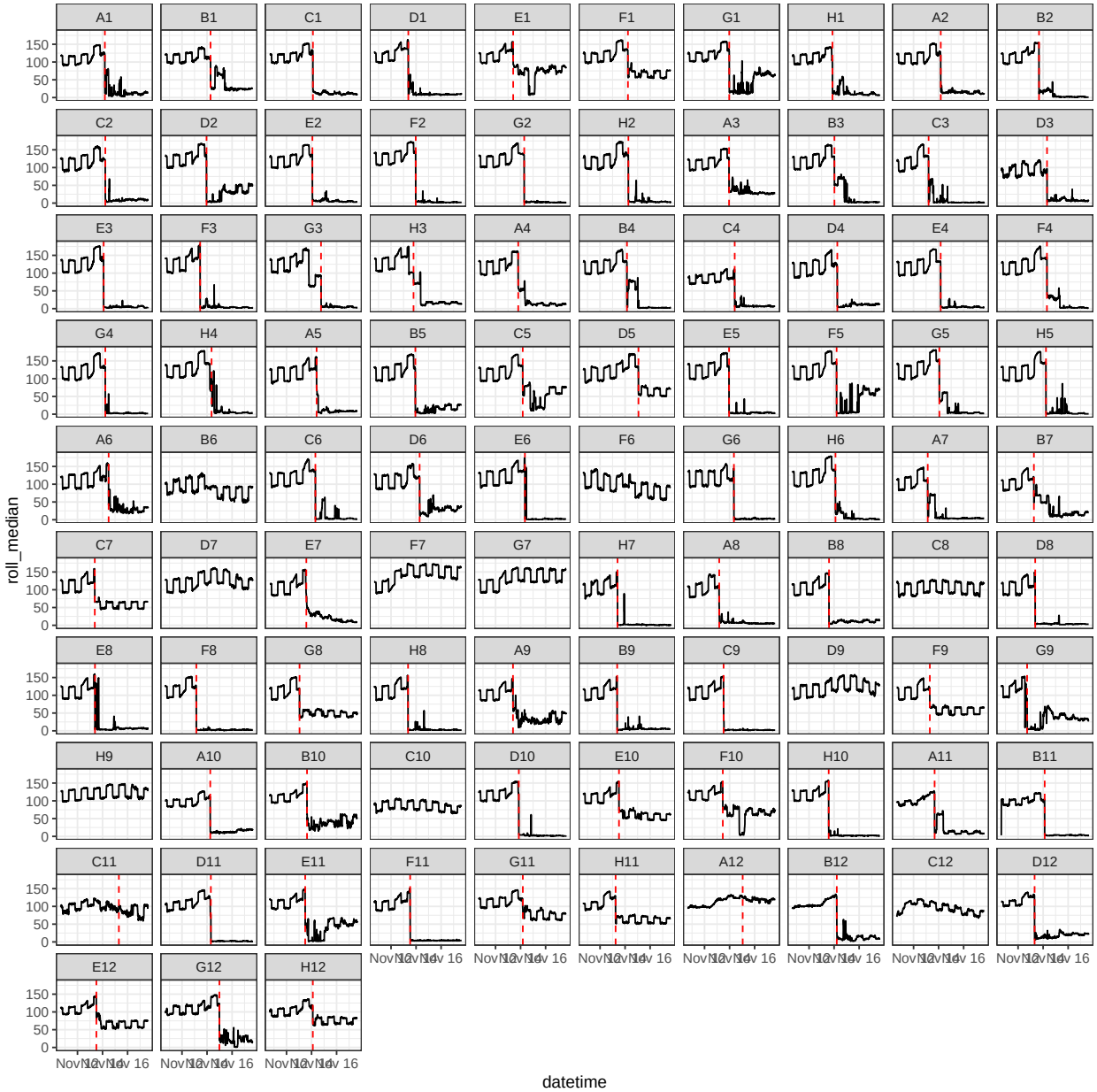
To correct these errors manually, substitute the incorrect eclosion time with NA.

```
res <- merge(res, id_well) %>%
  mutate(ecl_time = ifelse(well %in% c("B6", "F6", "D7", "F7", "G7", "D9", "H9", "C12"),
                           NA,
                           ecl_time),
         ecl_time = as.POSIXct(ecl_time, tz = "Etc/UTC-1"))
```

Plot the results again to confirm the correction.

```
merge(d, res) %>%
  ggplot(mapping = aes(x = datetime))+
    geom_line(aes(y = roll_median)) +
    geom_vline(aes(xintercept = ecl_time), linetype = 2, col = "red") +
    scale_x_datetime() +
    labs(aes(title = paste(as.POSIXct(ecl_time, tz = "Etc/UTC-1")), sep = ", "))+
    facet_wrap(.~well)+
    theme_bw()
```

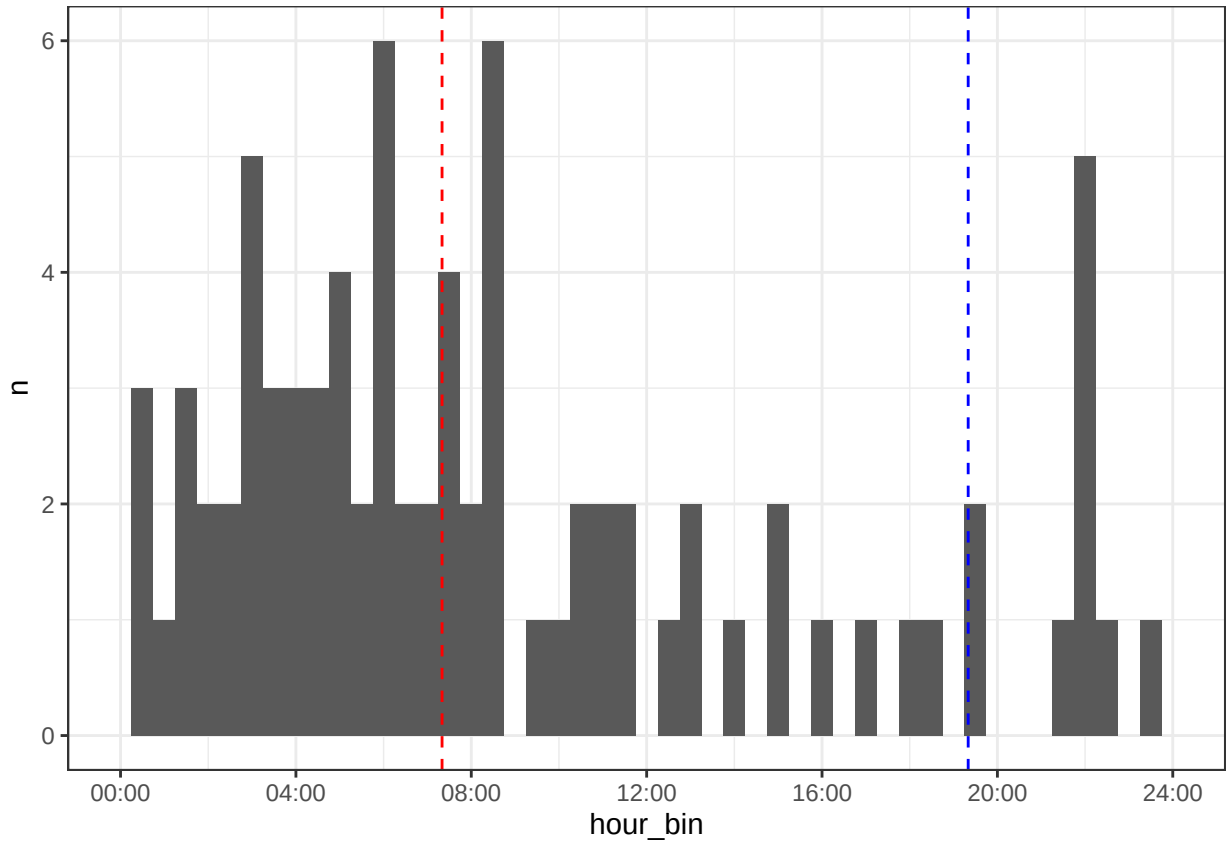
```
## Warning: Removed 9990 rows containing missing values or values outside the
## scale range (`geom_vline()`).
```



Plot the number of eclosion in each hour.

```r
bin_size <- 30/60
res %>%
  mutate(datetime = as.POSIXct(ecl_time, tz = "Etc/UTC-1"),
         hour_of_day = hour(datetime) + minute(datetime)/60,
         hour_bin = floor(hour_of_day / bin_size) * bin_size) %>%
  count(hour_bin) %>%
  ggplot(aes(x = hour_bin, y = n)) +
  geom_col(width = bin_size) +
  geom_vline(xintercept = 7 + 20/60, linetype = "dashed", color = "red") +
  geom_vline(xintercept = 19 + 20/60, linetype = "dashed", color = "blue") +
  scale_x_continuous(
    breaks = seq(0, 24, by = 4),
    labels = function(x) sprintf("%02d:00", x),
    limits = c(0,24)
  ) +
  theme_bw()
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale
## range (`geom_col()`).
```



Save the result

```r
res <- res %>%
  mutate(rec = "rec_016") %>%
  select(rec, well, id, ecl_time)
```

```
write.csv(res, "test_data/rec_016_result.csv", quote = F, row.names = F)
```

After genotyping, genotype column is added, and we can visualise the result for different genotypes separately.