ggplot2

Kevin.C

ggplot2 Introduction

ggplot2 is an open-source data visualization package for the statistical programming language R. Created by Hadley Wickham in 2005, ggplot2 is an implementation of Leland Wilkinson's Grammar of Graphics—a general scheme for data visualization which breaks up graphs into semantic components such as scales and layers. ggplot2 can serve as a replacement for the base graphics in R and contains a number of defaults for web and print display of common scales. Since 2005, ggplot2 has grown in use to become one of the most popular R packages. It's also the foundation of many graphic applications such as Vega-Lite and Tableau.

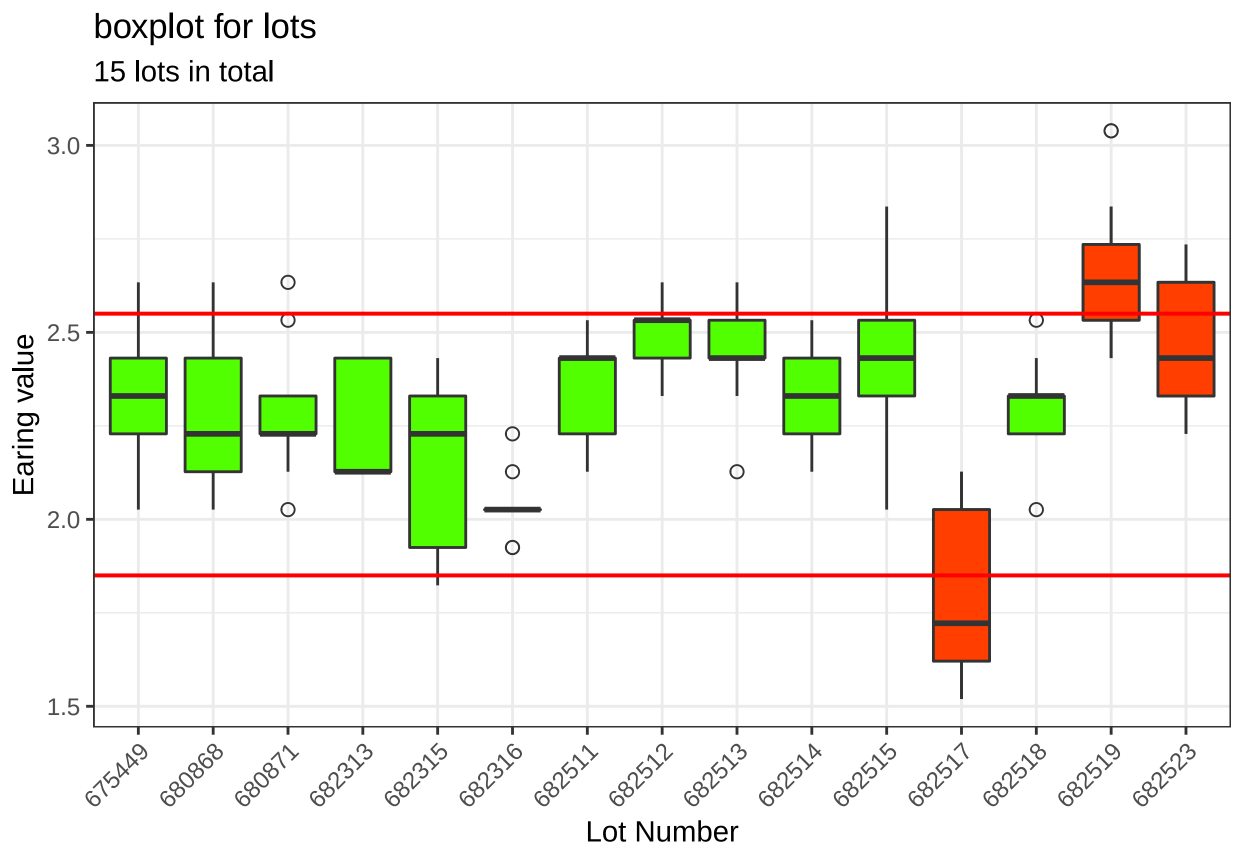
The grammar of graphics breaks up graphs into several components:

```
    -Data:
            Defines what can be done with it.Sometimes you might need to manipulate
            data first.
    -Mapping:
            Generic datasets to be understood by the graphic system. Aesthetic mapping
            would link variables in data to graphical properties in the geometry
    -Statistic:
            Transform input variables to displayed values
            Ex:
            • Count number of observations in each category for a bar chart
            • Calculate summary statistics for a boxplot.
    -Scale:
            A scale translate back and forth between variable ranges and property ranges
            Ex: Categories → Colour

    -Geometries:
            How to interpret aesthetics as graphical representations
    -Facets:
            Define the number of panels with equal logic and split data among them
    -Coordinates
            Defines the physical mapping of the aesthetics to the paper
    -Theme
            Theming spans every part of the graphic that is not linked to data
```

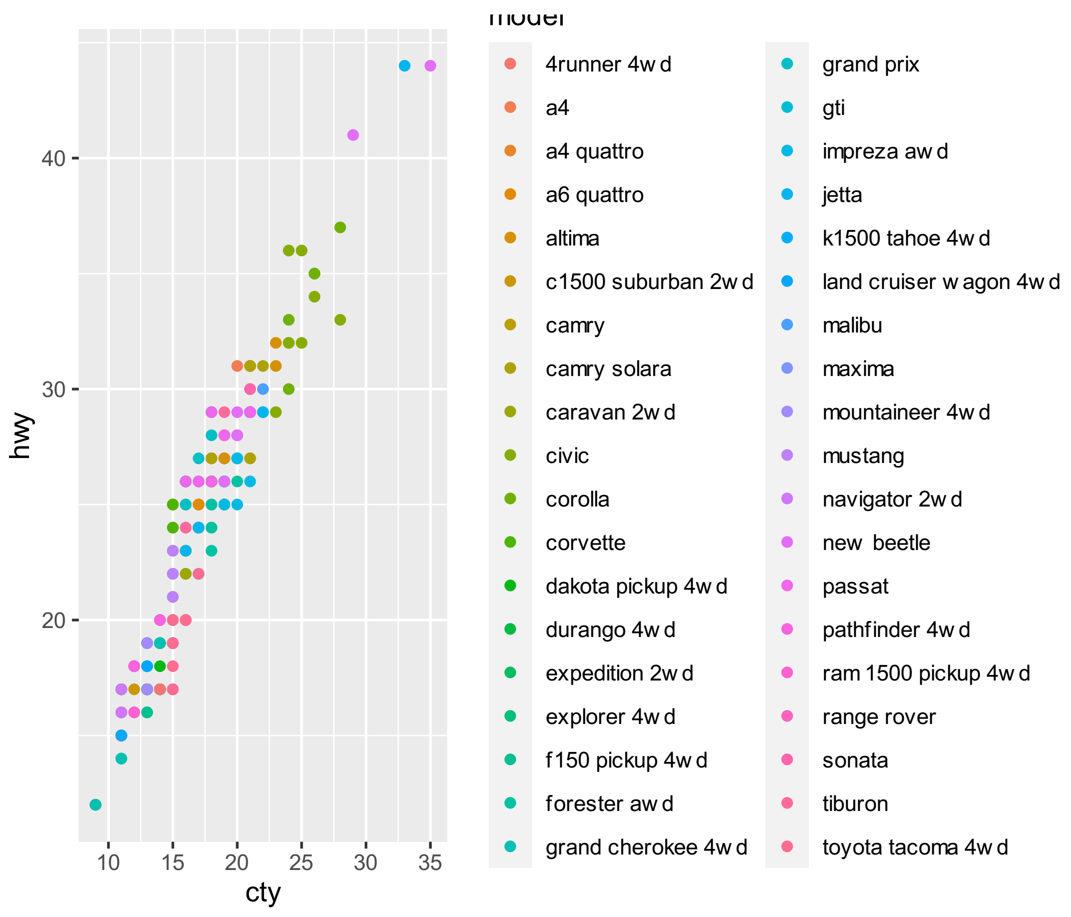
With that being said, the coding will be like stacking several layers to create a highly customized visualization as below.

```r
HL_lot_num <- c("682517","682519","682523")

Highlight <- Earing %>%
        filter(lot %in% HL_lot_num)
```

```r
# Stacking several functions to customize the vis.
 ggplot(Earing) +
        geom_boxplot(aes(lot,earing),fill="#52FF00",outlier.shape = 1,outlier.size = 2,) +
        geom_boxplot(aes(lot,earing),data = Highlight,fill="#FF3E00",
                     outlier.shape = 1,outlier.size = 2,) +
        geom_hline(aes(yintercept = 1.85),color = "red",size = 0.7)+
        geom_hline(aes(yintercept = 2.55),color = "red",size = 0.7)+
        theme_bw()+
        theme(axis.text.x = element_text(angle = 45, hjust = 1))+
        ggtitle("boxplot for lots",subtitle = "15 lots in total")+
        labs(x="Lot Number",y="Earing value")
```



```r
# Interaction example - it shows tooltip when hovering on the point.
interaction_plot <- ggplot(mpg)+
  geom_point_interactive(aes(cty,hwy,color = model,tooltip = model,data_id= model))
girafe(ggobj = interaction_plot)
```



```
# "ggiraph" for interaction
# https://davidgohel.github.io/ggiraph/
```

Although it looks complicated, you can actually create a simple chart with the data by mapping them into the corresponding geometry with only one or two lines of codes. We'll make some simple chart in the following.

Basics

ggplot2 is based on the grammar of graphics, the idea that you can build every graph from the same components: a data set, a coordinate system, and geoms—visual marks that represent data points. To display values, map variables in the data to visual properties of the geom (aesthetics) like size, color, and x. and y locations.

*** Note that we use "+" to add layers.**

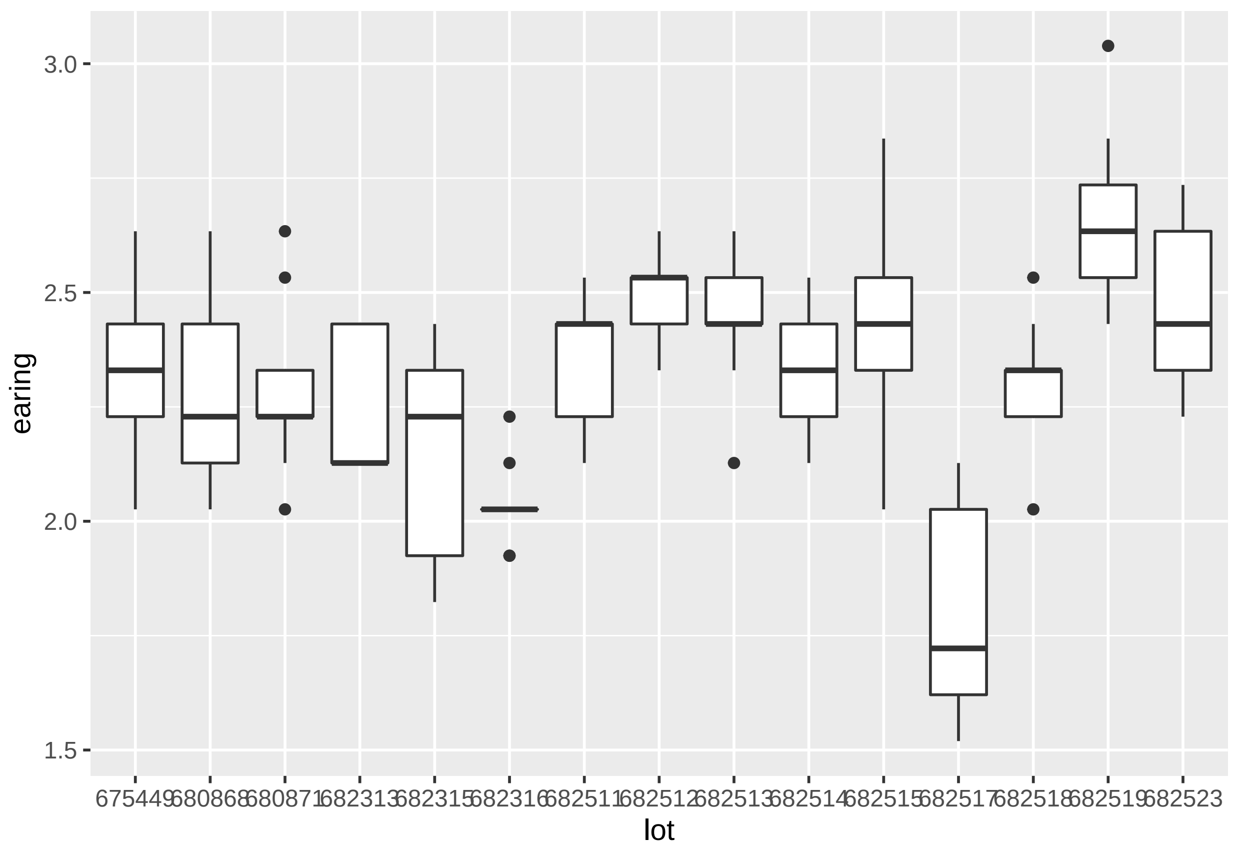
```
 ------- We have to provide <DATA>, choose <GEOM_FUNCTION> and <MAPPINGS>
 variables you want to map-------

ggplot (data = <DATA> ) +
 <GEOM_FUNCTION> (mapping = aes( <MAPPINGS> ),

 ------- Code below are not required, only use them when you want to add
 extra stuff ------

 stat = <STAT> , position = <POSITION> ) +
 <COORDINATE_FUNCTION> +
 <FACET_FUNCTION> +
 <SCALE_FUNCTION> +
 <THEME_FUNCTION>
```

We will continue to use jellybeans.dat, Earing.dat to make some plots

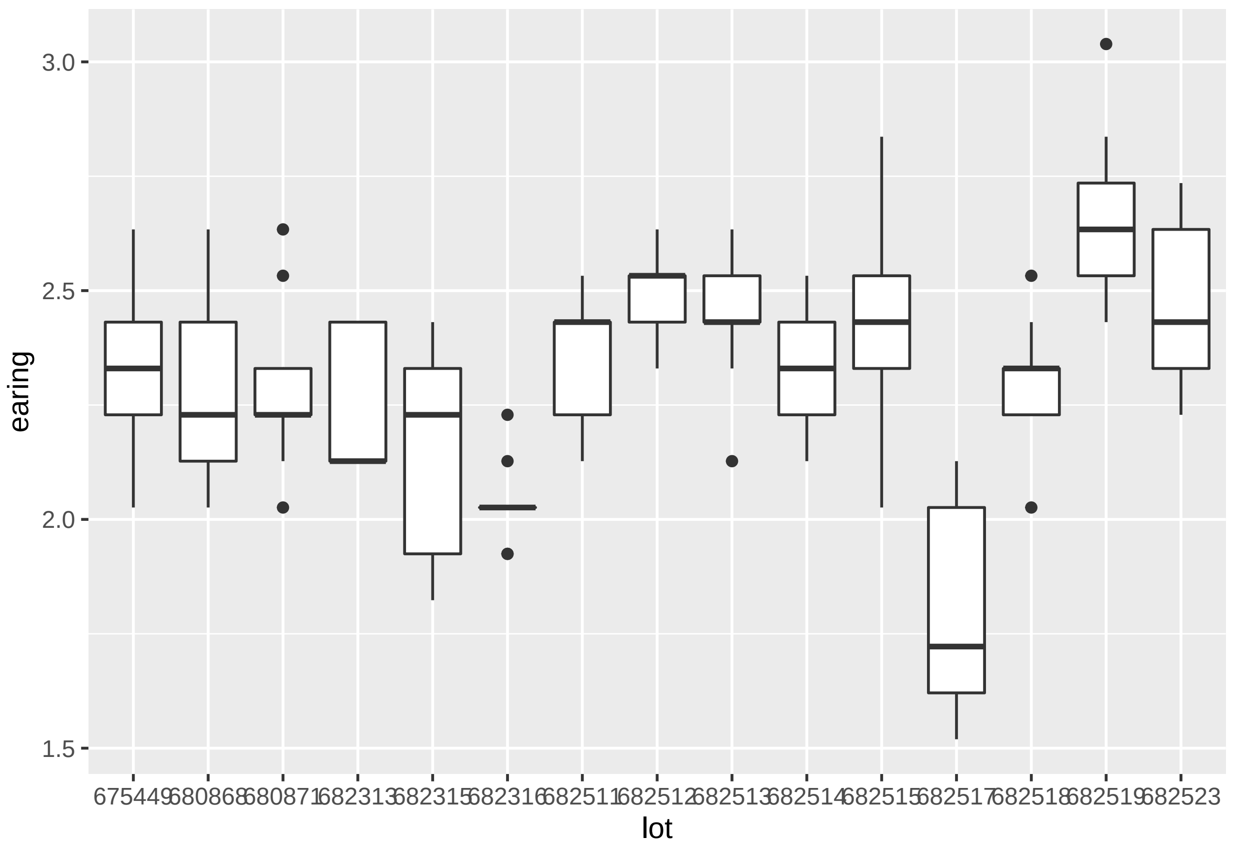
Boxplot / Histgram

You can define data sets and variables in ggplot() or geom_function(). Defining in the ggplot() would make it become a global setting, while the another would only effect in the certain layer.

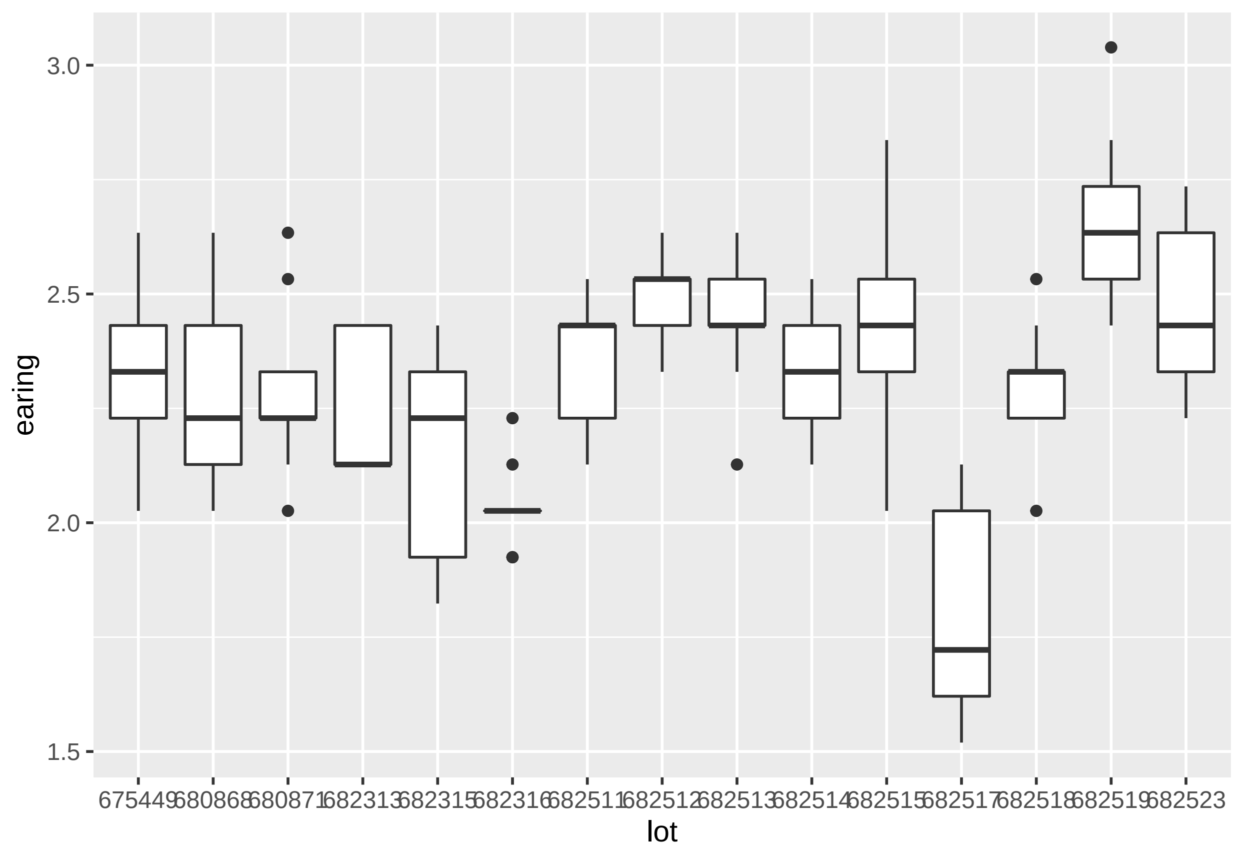
```
# Provide dataset, geom type and variables
ggplot(Earing) + geom_boxplot(aes(lot,earing))
```



```r
# The data and mapping setting can also be defined in ggplot()
ggplot(Earing,aes(lot,earing)) + geom_boxplot()
```

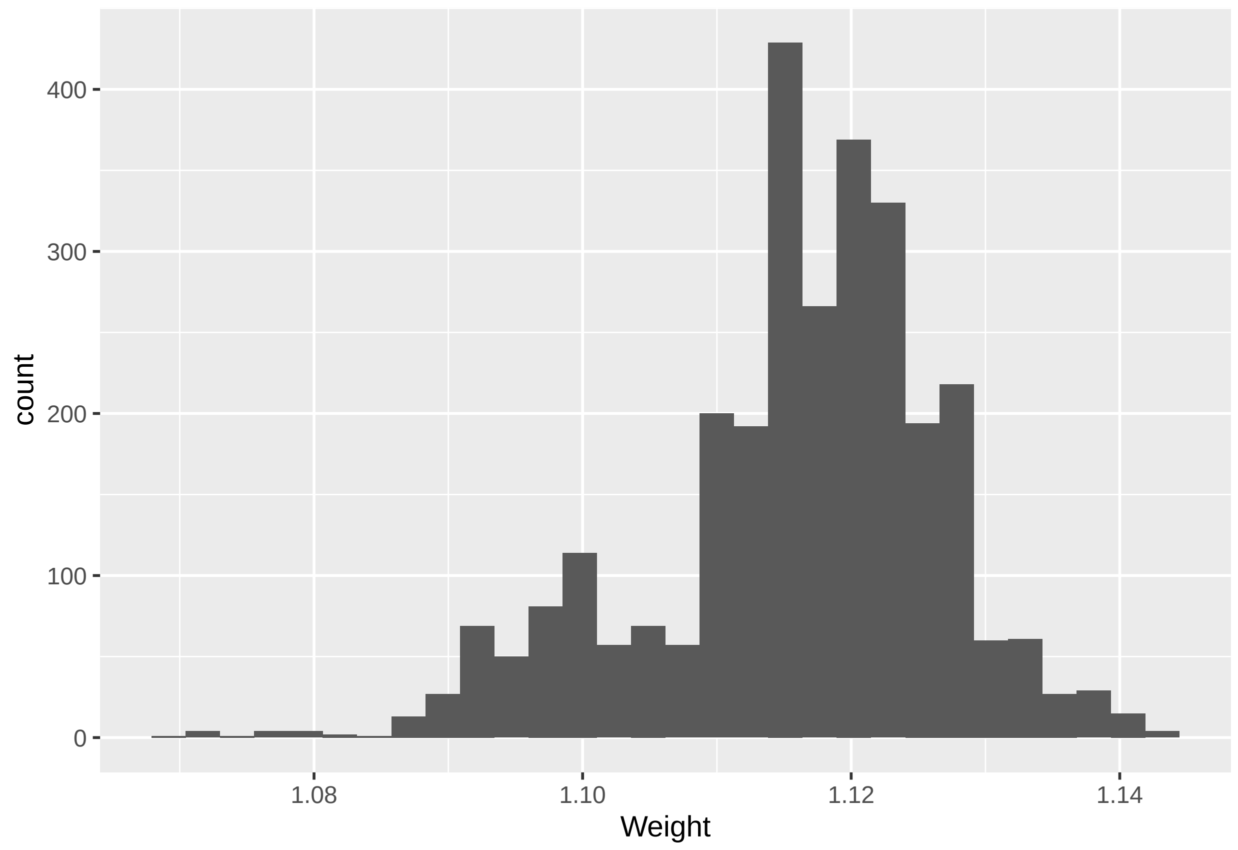


```r
# Also doable if you want to define all of it in geom_boxplot
ggplot() + geom_boxplot(aes(lot,earing),data = Earing)
```



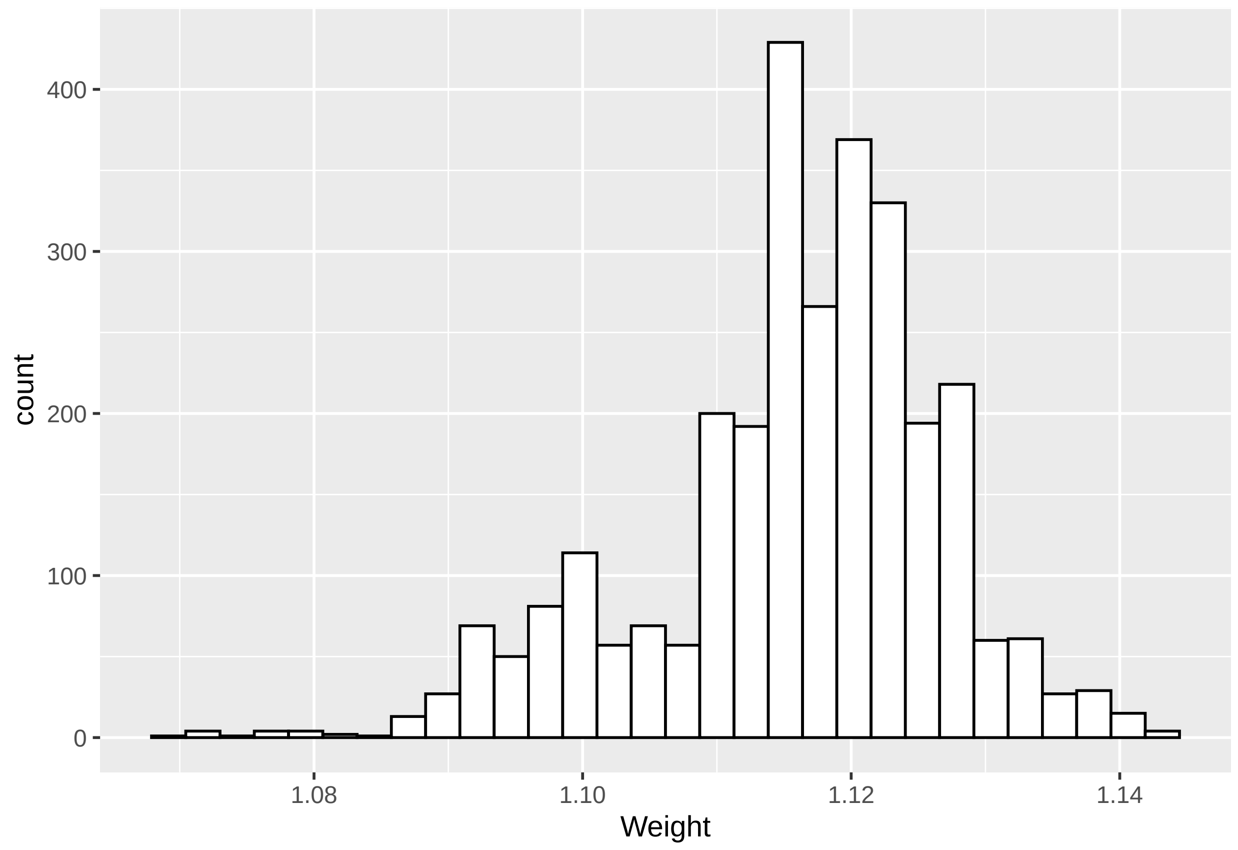
```
# Hist
ggplot(jellybeans)+geom_histogram(aes(Weight))
```

```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```

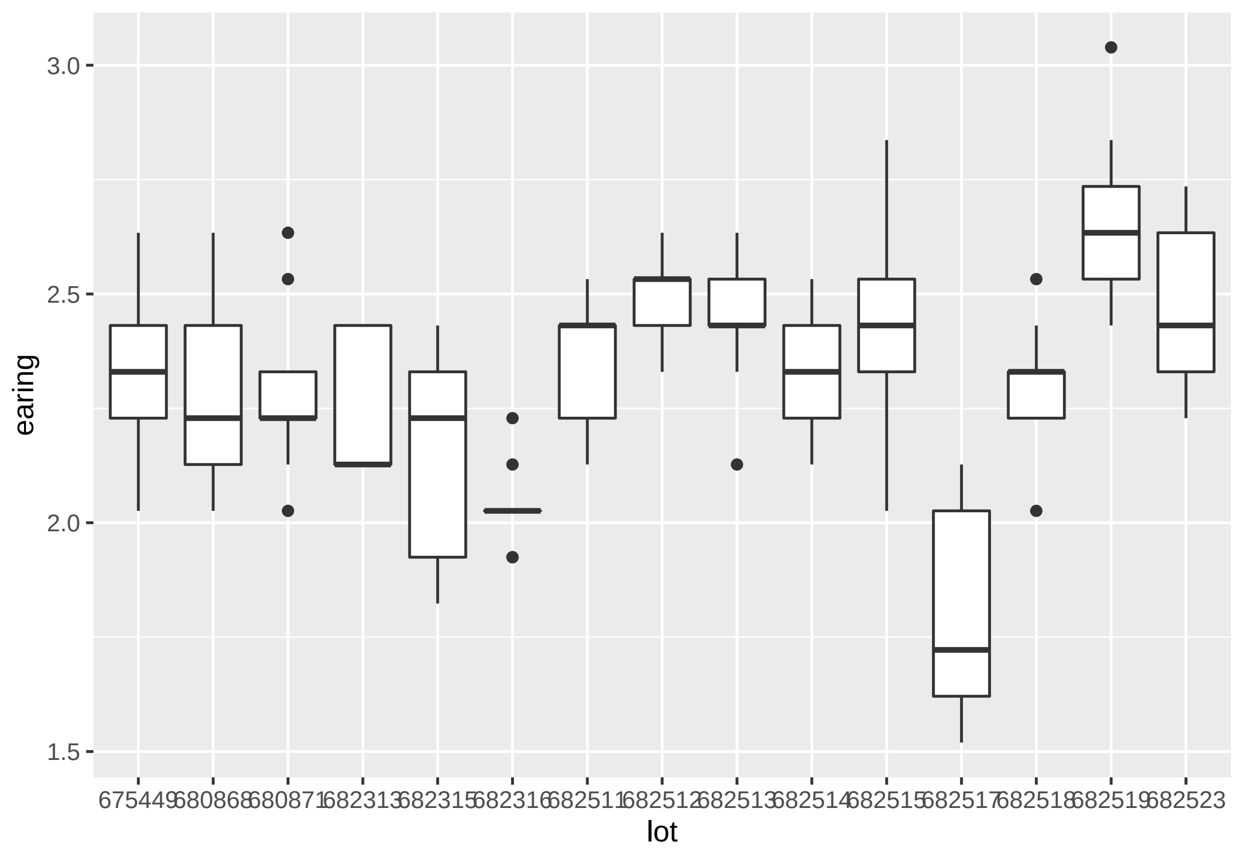


```
ggplot(jellybeans)+geom_histogram(aes(Weight),color = "black", fill = "white")
```

```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```



```r
# qplot - another way to use ggplot2, but I never use this one
qplot(lot,earing,data = Earing,geom = "boxplot")
```

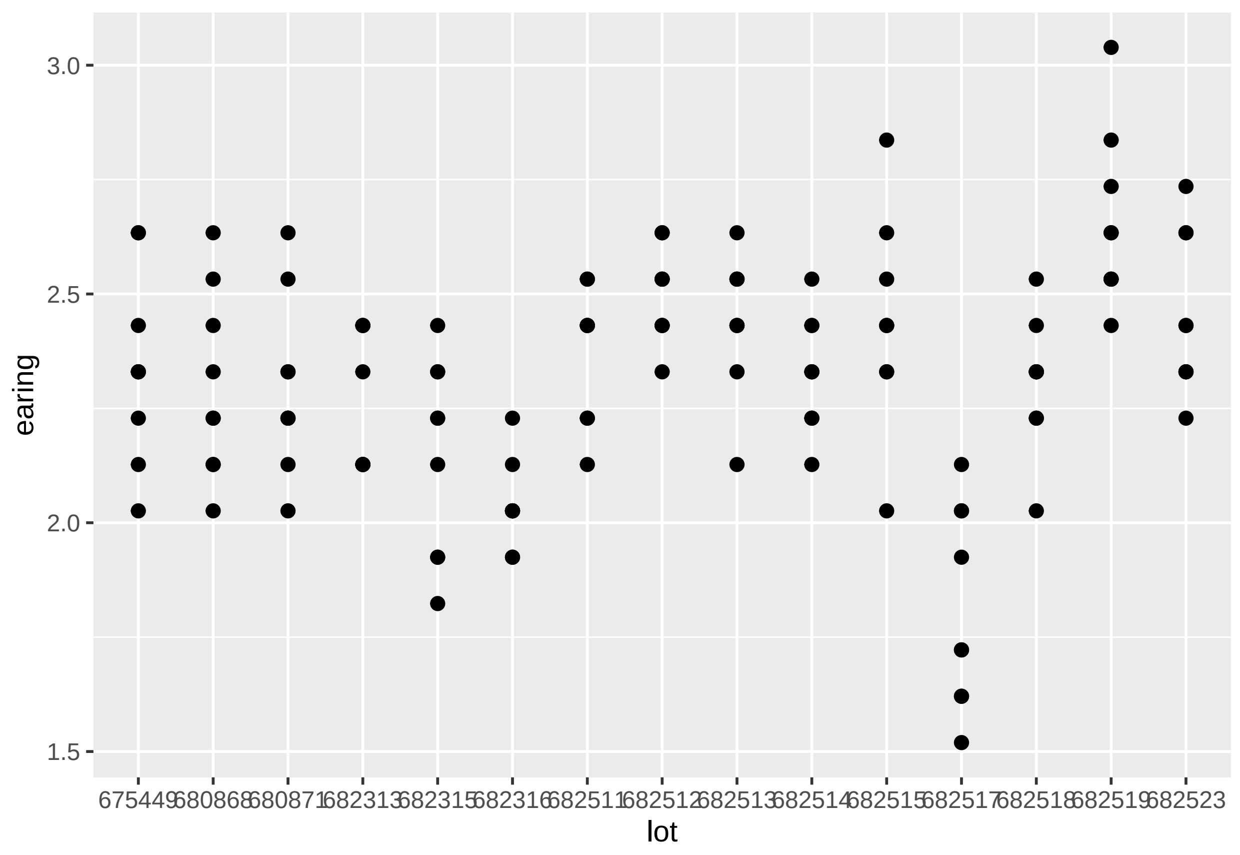


```
# Save plot to working directory
ggsave("lot.png", width = 5, height = 5)
```

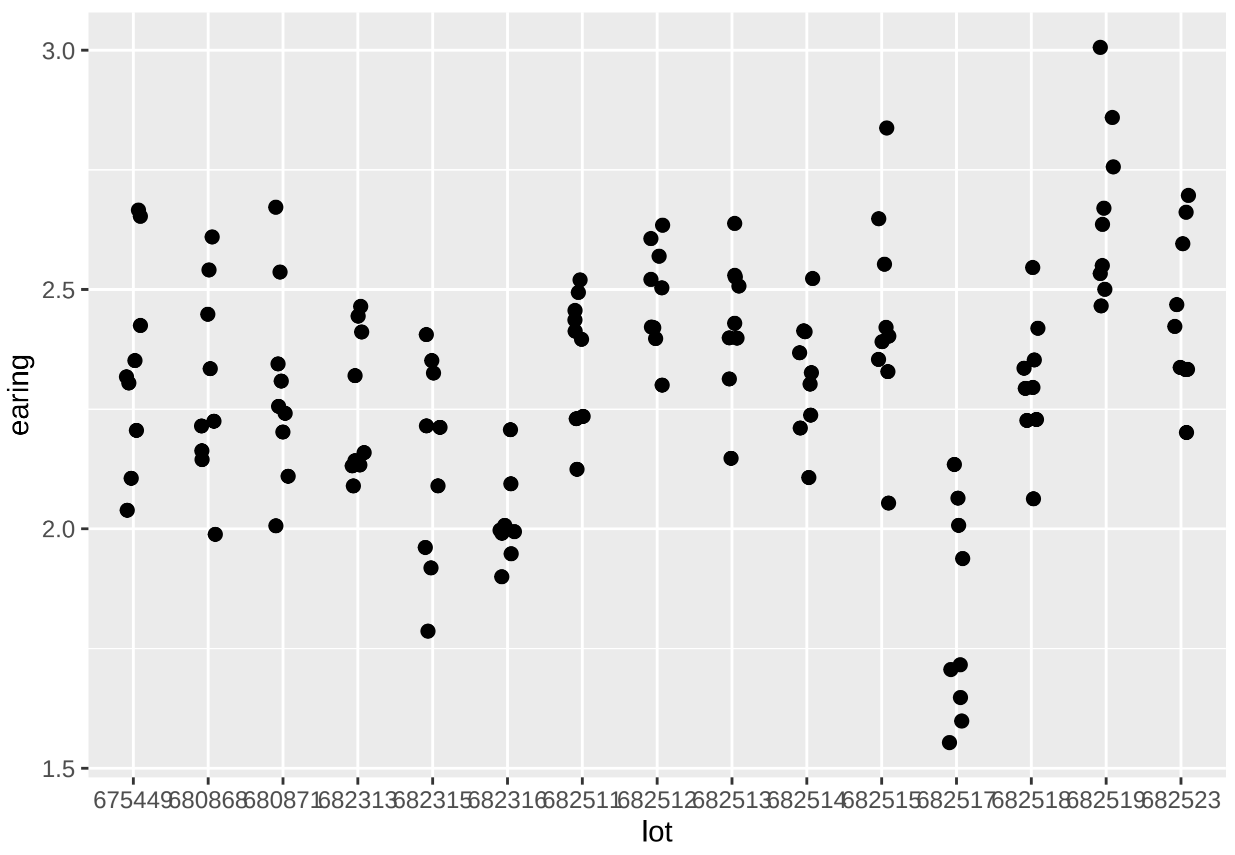
Point/Jitter

A jitter plot is a variant of the strip plot with a better view of overlapping data points, used to visualise the distribution of many individual one-dimensional values.

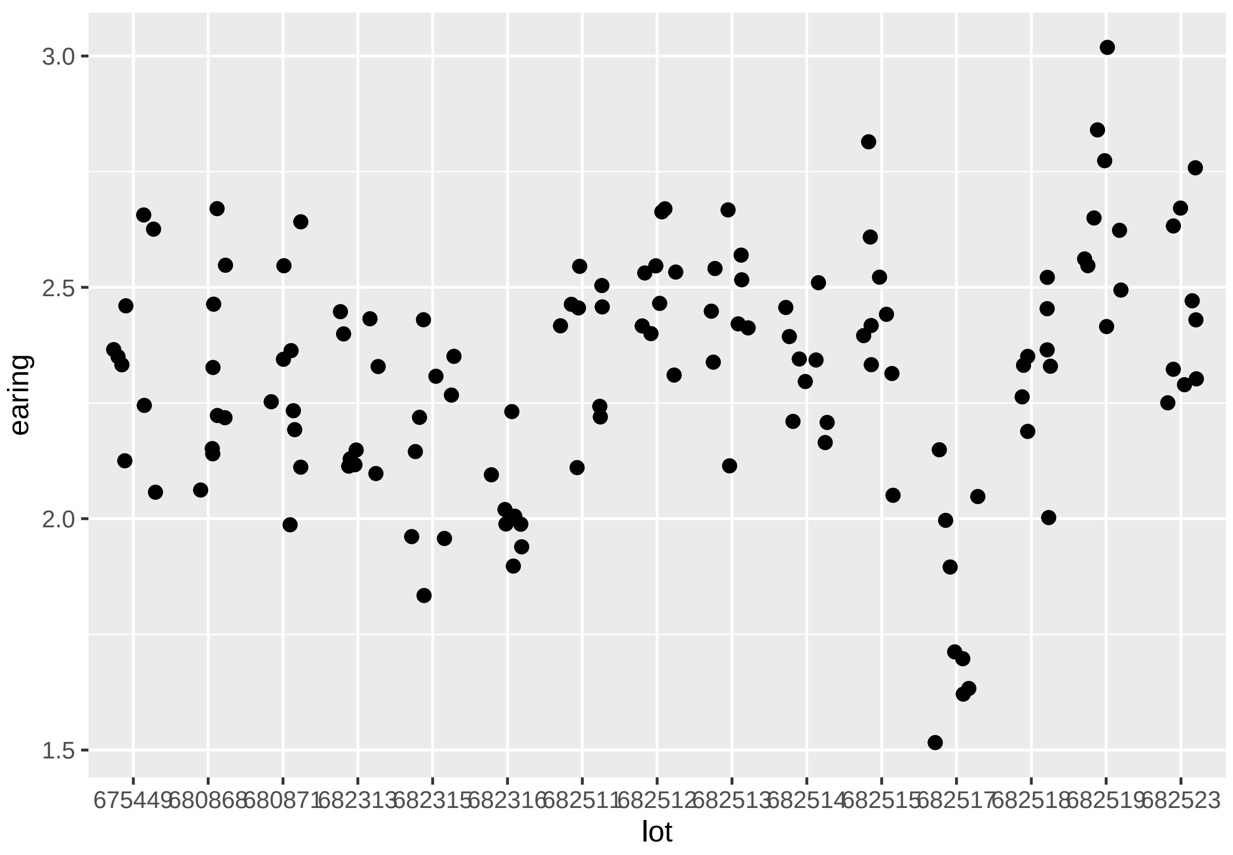
```
# Provide dataset, geom type and variables
ggplot(Earing) + geom_point(aes(lot,earing),size = 2)
```



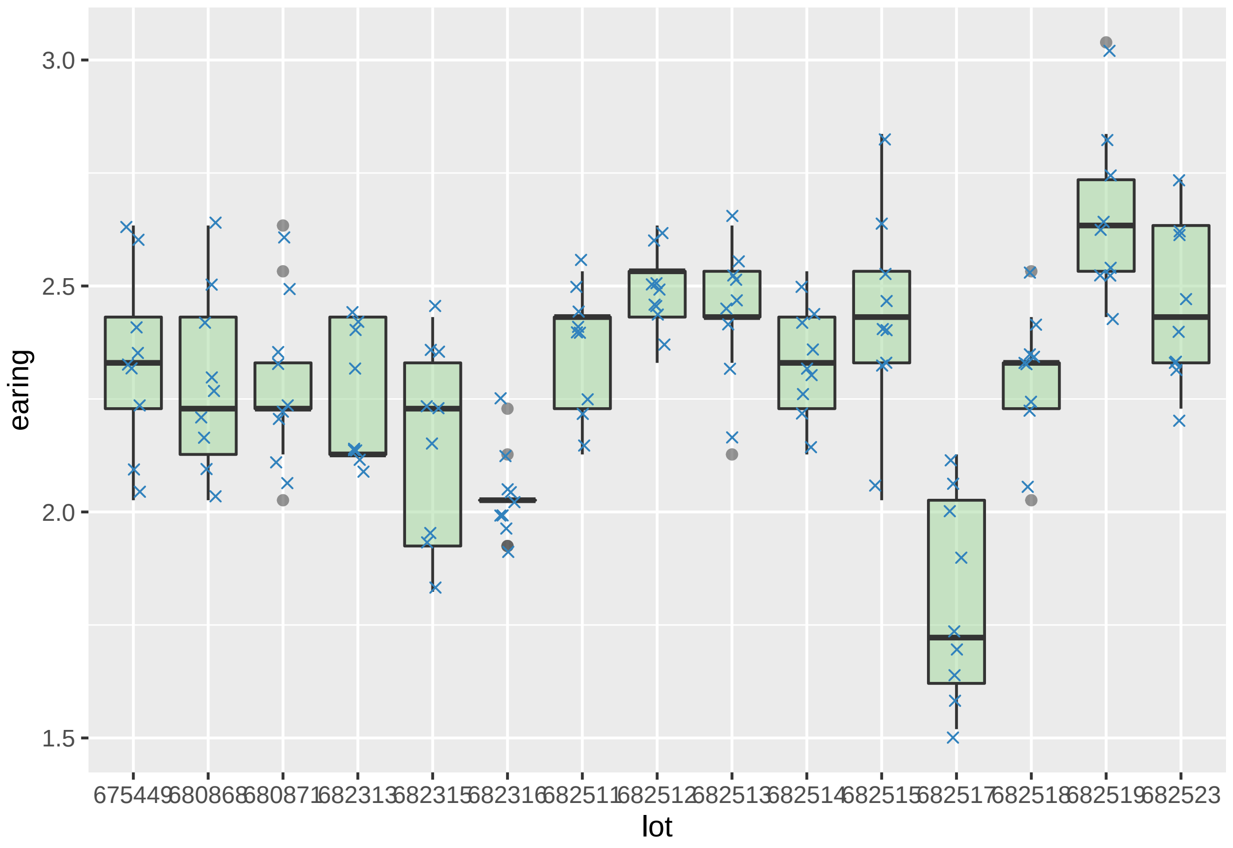
```r
# Adjust width - Recommend you adjust it and see how it goes.
ggplot(Earing) + geom_jitter(aes(lot,earing),width = 0.1,size = 2)
```



```
ggplot(Earing) + geom_jitter(aes(lot,earing),width = 0.3,size = 2)
```

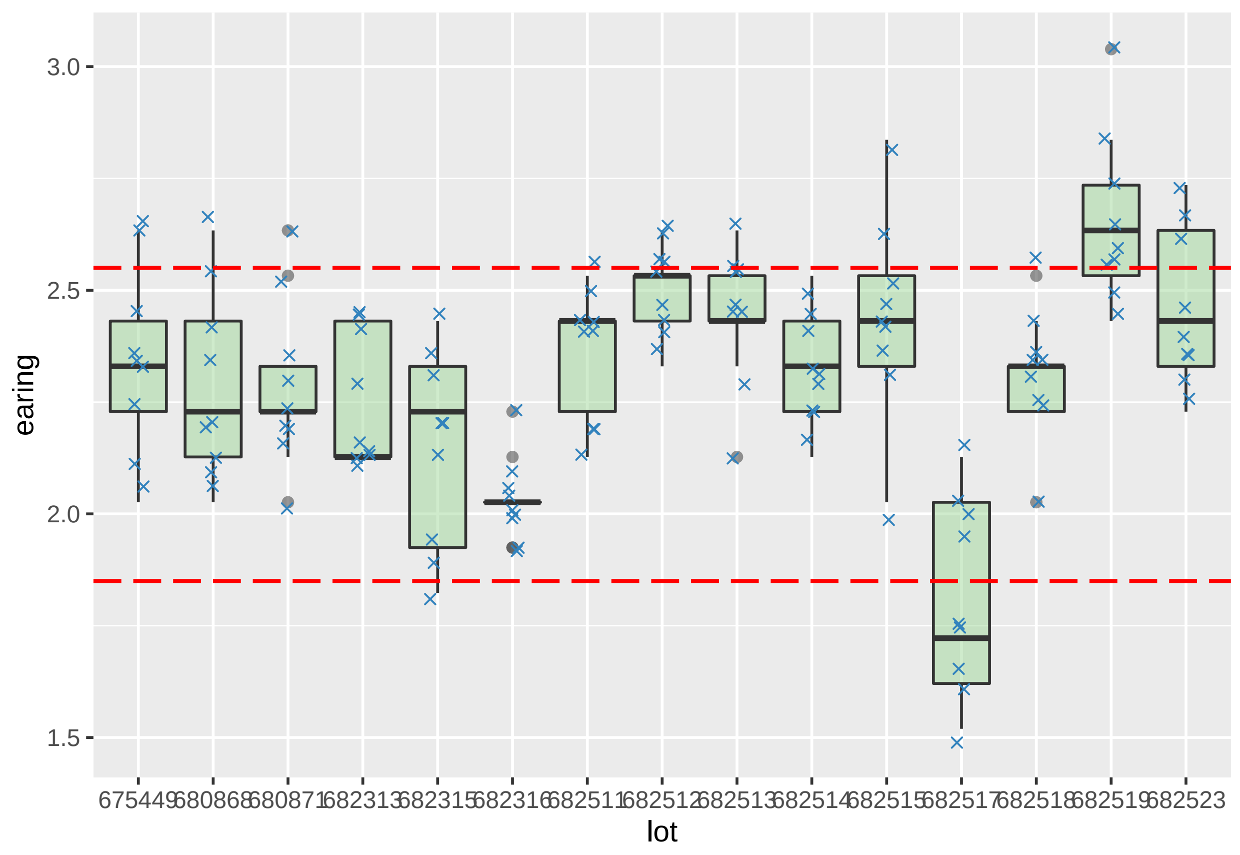


```
# Show it together / I changed the point type here.
plot <- ggplot(Earing) +
  geom_boxplot(aes(lot,earing),fill = "#a1d99b",alpha = 0.5)+
  geom_jitter(aes(lot,earing),width = 0.1,shape = 4,color = "#3182bd");plot
```

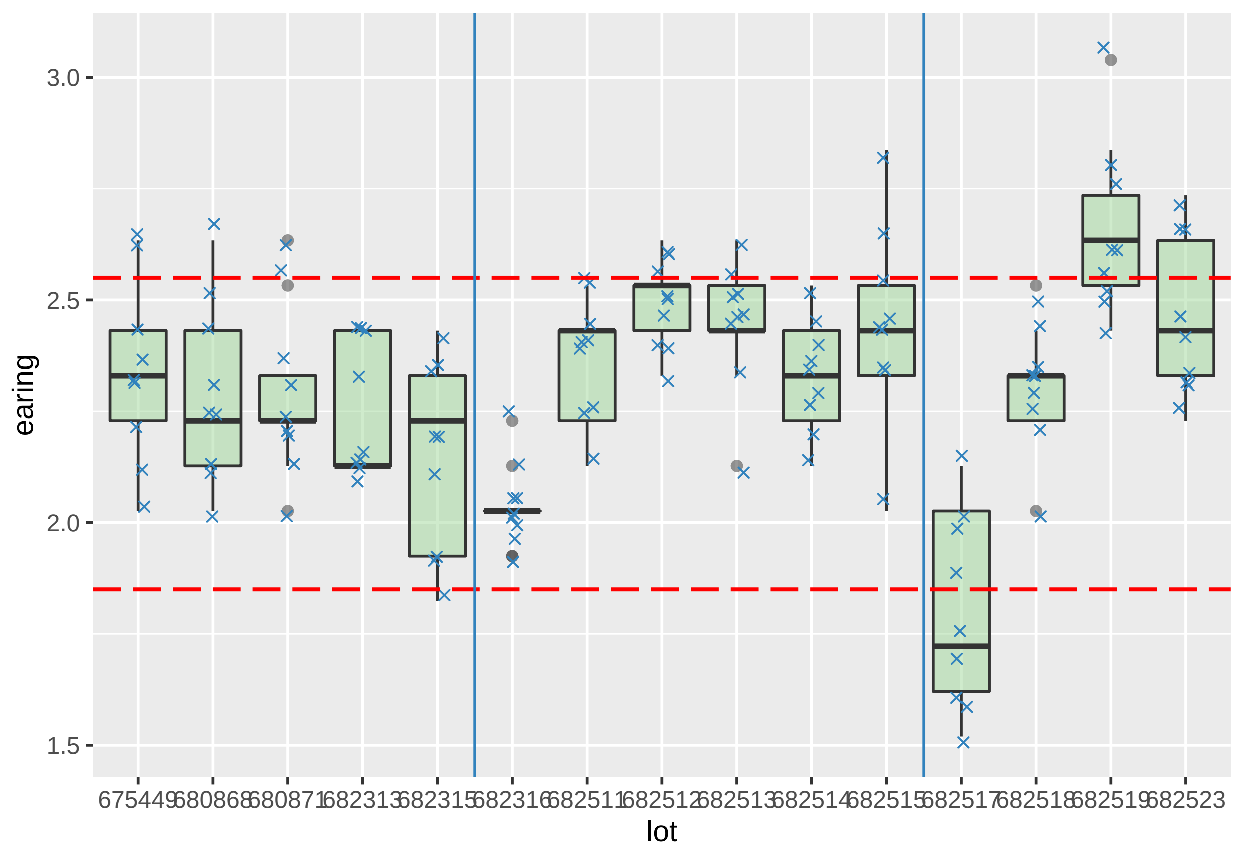


Add Line & Annotation

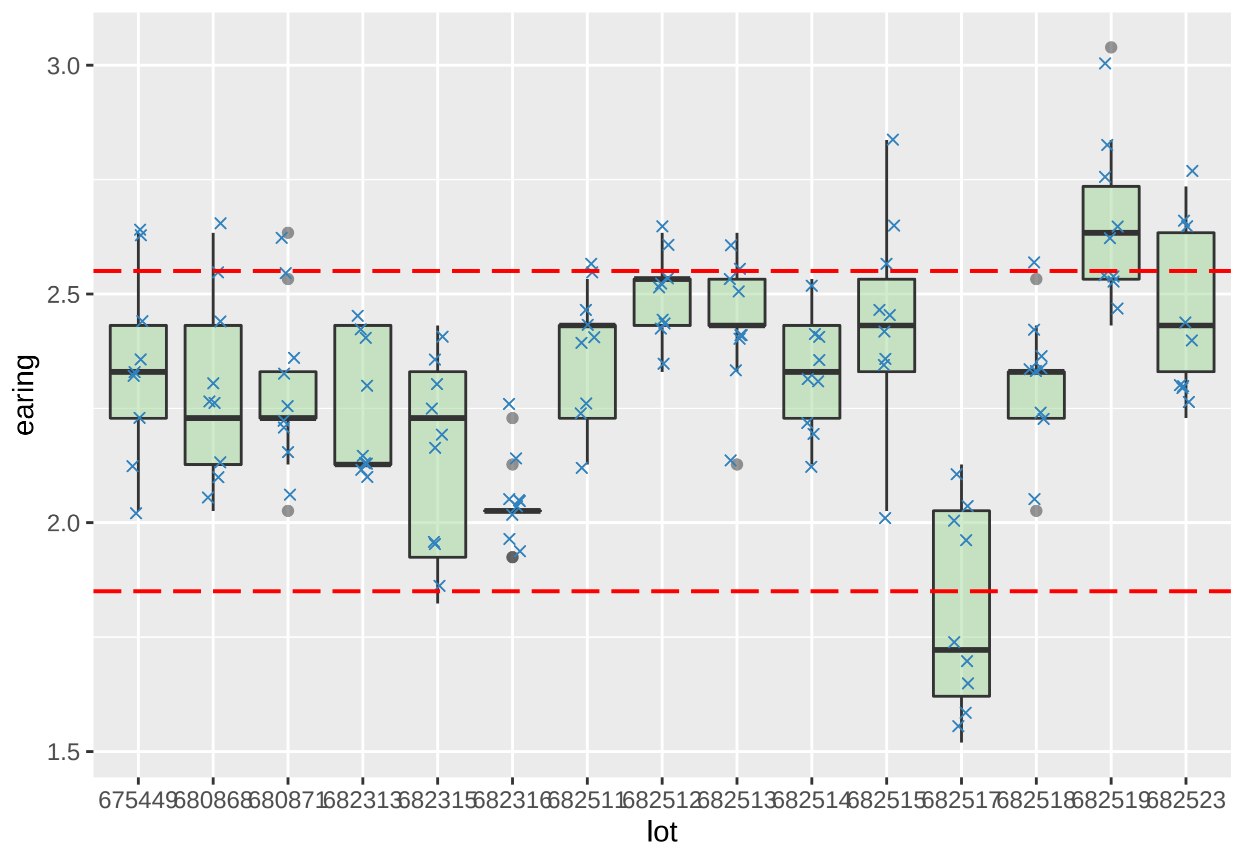
```
# Line type
# http://sape.inf.usi.ch/quick-reference/ggplot2/linetype

# Horizontal Line
plot <- ggplot(Earing) +
  geom_boxplot(aes(lot,earing),fill = "#a1d99b",alpha = 0.5)+
  geom_jitter(aes(lot,earing),width = 0.1,shape = 4,color = "#3182bd")+
  geom_hline(aes(yintercept = 1.85),color = "red",size = 0.7,linetype ="longdash")+
  geom_hline(aes(yintercept = 2.55),color = "red",size = 0.7,linetype ="longdash");plot
```



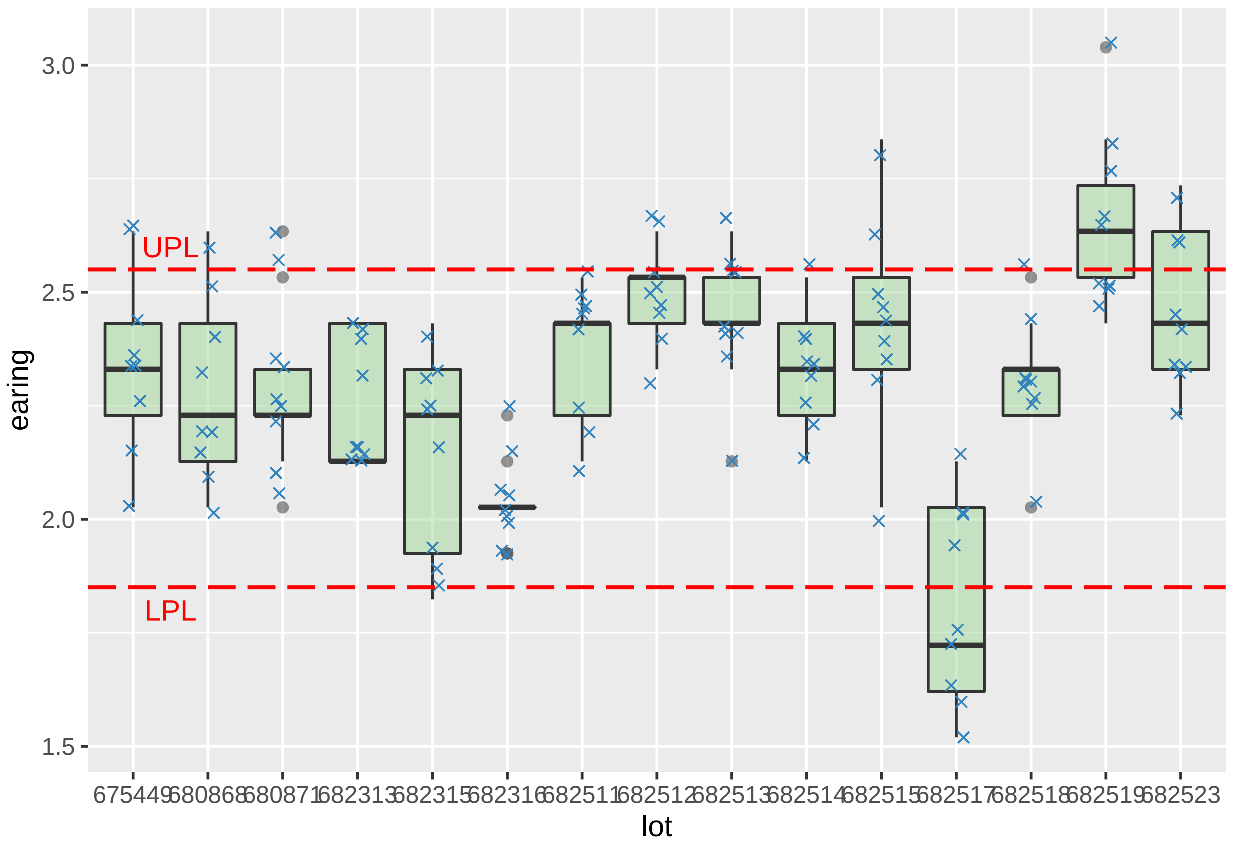
```
# Vertical Line
plot <- ggplot(Earing) +
  geom_boxplot(aes(lot,earing),fill = "#a1d99b",alpha = 0.5)+
  geom_jitter(aes(lot,earing),width = 0.1,shape = 4,color = "#3182bd")+
  geom_hline(aes(yintercept = 1.85),color = "red",size = 0.7,linetype ="longdash")+
  geom_hline(aes(yintercept = 2.55),color = "red",size = 0.7,linetype ="longdash")+
  geom_vline(aes(xintercept = 5.5),color = "#3182bd",size = 0.5)+
  geom_vline(aes(xintercept = 11.5),color = "#3182bd",size = 0.5);plot
```



```
# Make a df for limit value
limit = c(1.85,2.55)
Limit = as.data.frame(limit)

plot <- ggplot(Earing) +
  geom_boxplot(aes(lot,earing),fill = "#a1d99b",alpha = 0.5)+
  geom_jitter(aes(lot,earing),width = 0.1,shape = 4,color = "#3182bd")+
  geom_hline(data = Limit,aes(yintercept = limit),
             color = "red",size = 0.7,linetype ="longdash");plot
```

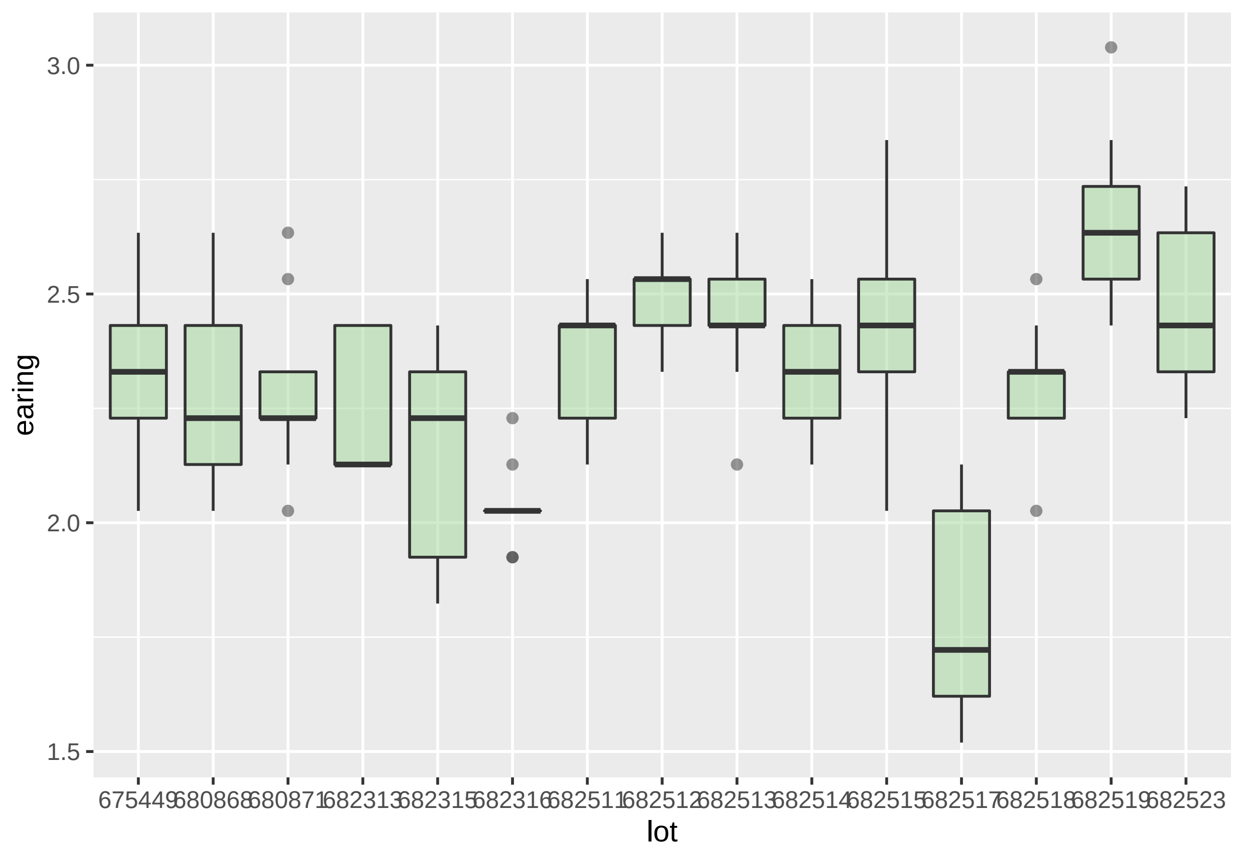


```
# Annotation
plot <- ggplot(Earing) +
  geom_boxplot(aes(lot,earing),fill = "#a1d99b",alpha = 0.5)+
  geom_jitter(aes(lot,earing),width = 0.1,shape = 4,color = "#3182bd")+
  geom_hline(data = Limit,aes(yintercept = limit),
             color = "red",size = 0.7,linetype ="longdash")+
  annotate(geom = "text",x = c(1.5,1.5),y = c(1.8,2.6),
           label = c("LPL","UPL"), color ="red");plot
```

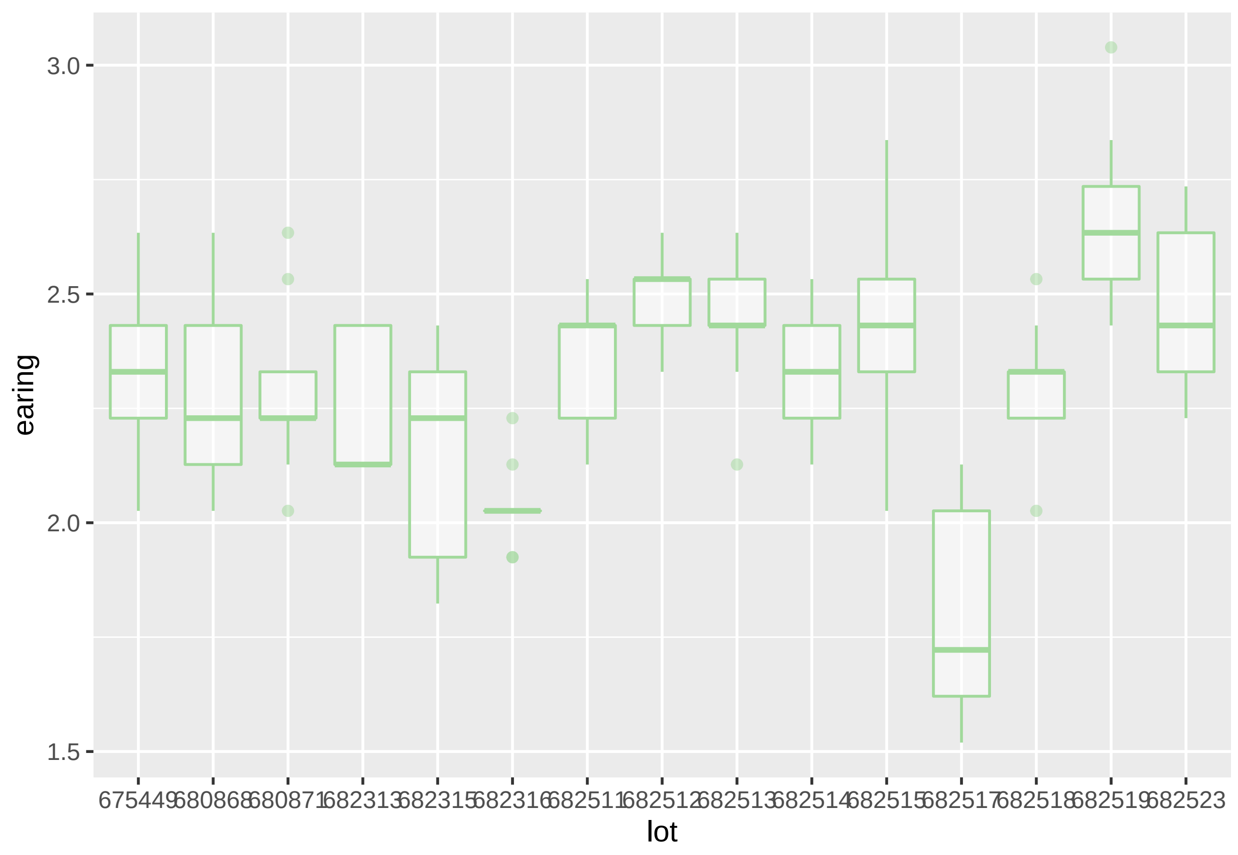


Color

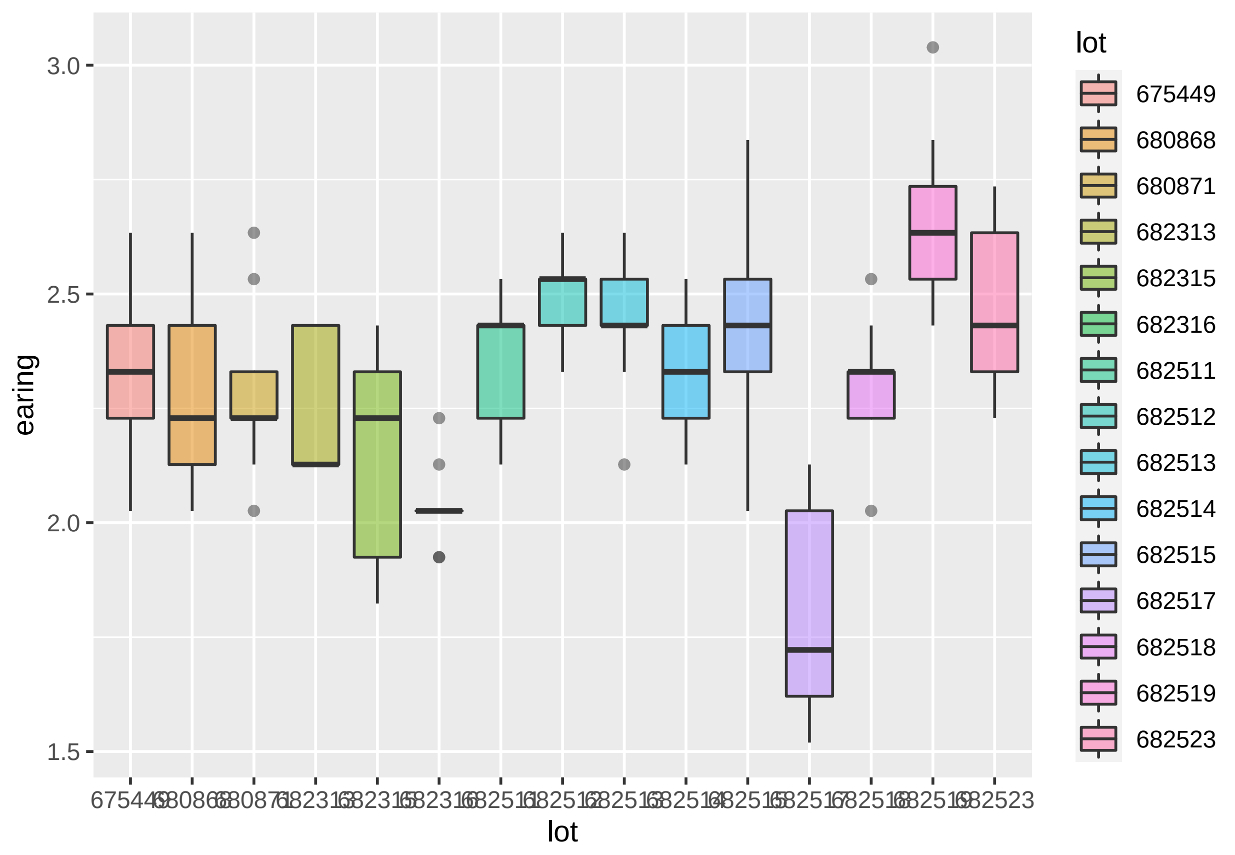
```
# https://colorbrewer2.org/#type=sequential&scheme=PuBuGn&n=8

# Fill vs color
# Fill
plot <- ggplot(Earing) +
  geom_boxplot(aes(lot,earing),fill = "#a1d99b",alpha = 0.5)  ; plot
```



```
# Color
plot <- ggplot(Earing) +
  geom_boxplot(aes(lot,earing),color = "#a1d99b",alpha = 0.5)  ; plot
```

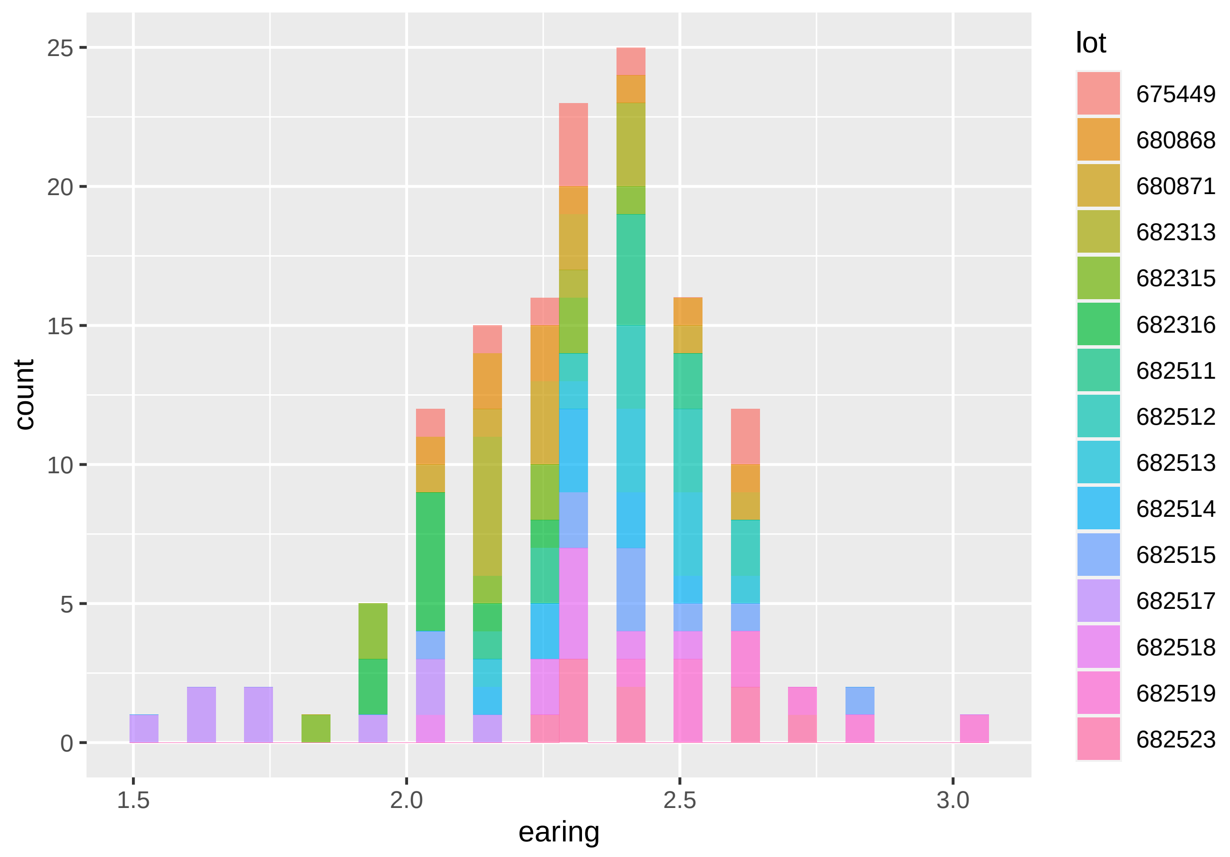


```
# Mapping fill with data
plot <- ggplot(Earing) +
  geom_boxplot(aes(lot,earing,fill=lot),alpha = 0.5)  ; plot
```

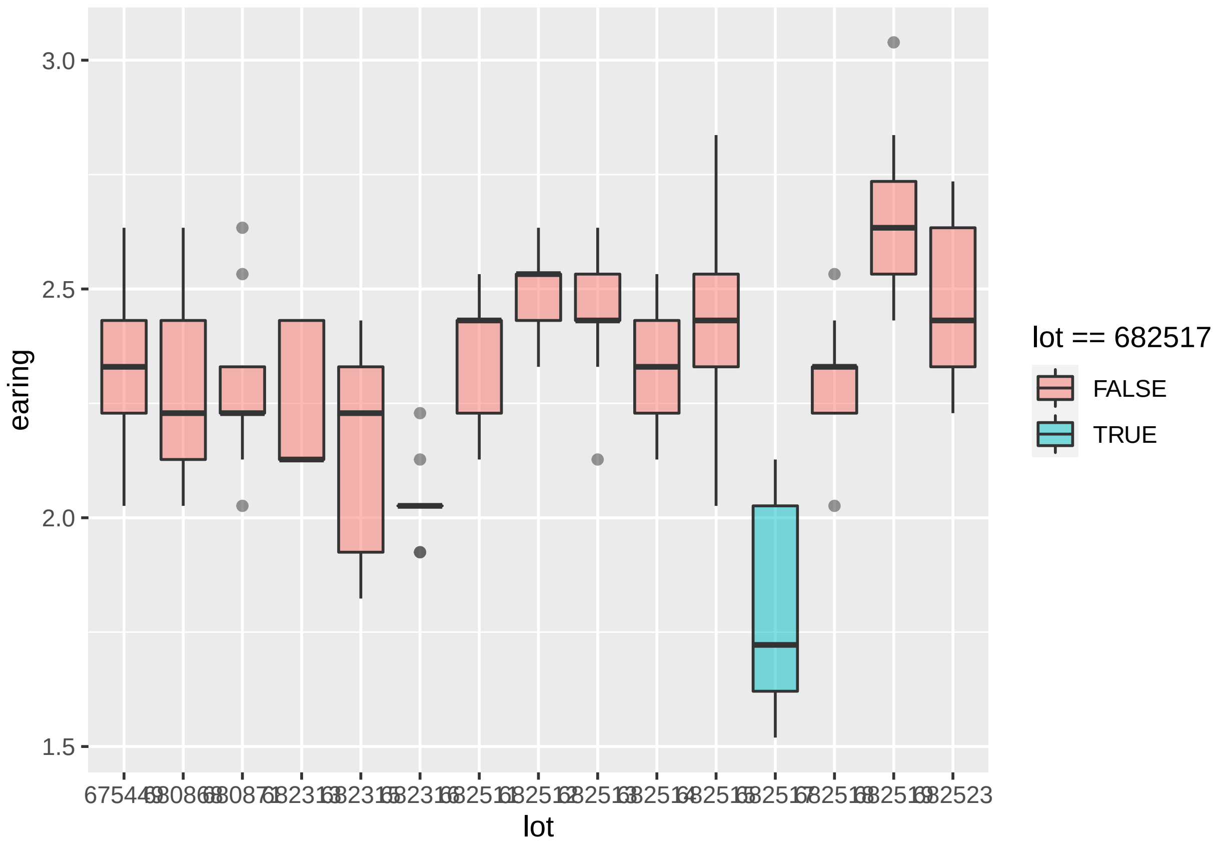


```
hist <- ggplot(Earing) +
  geom_histogram(aes(earing,fill=lot),alpha = 0.7) ; hist
```

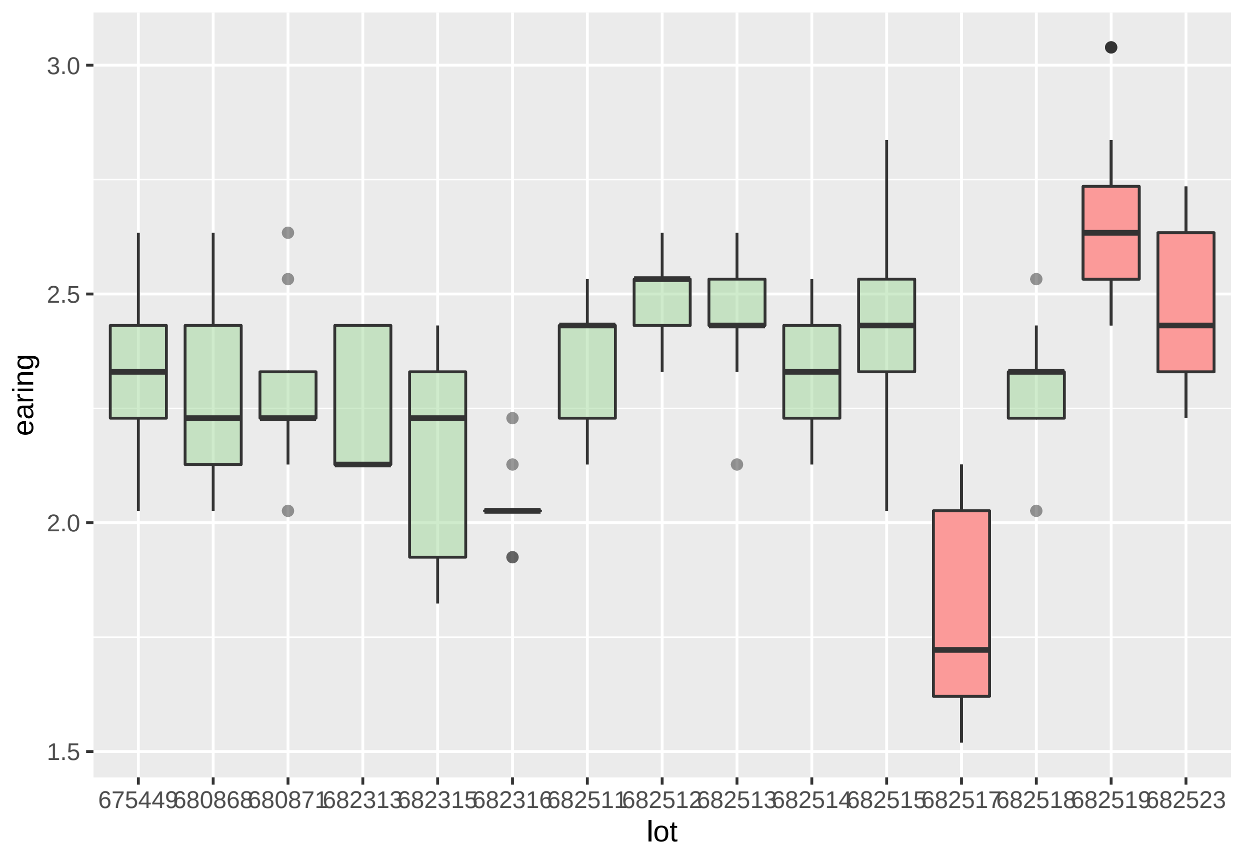
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



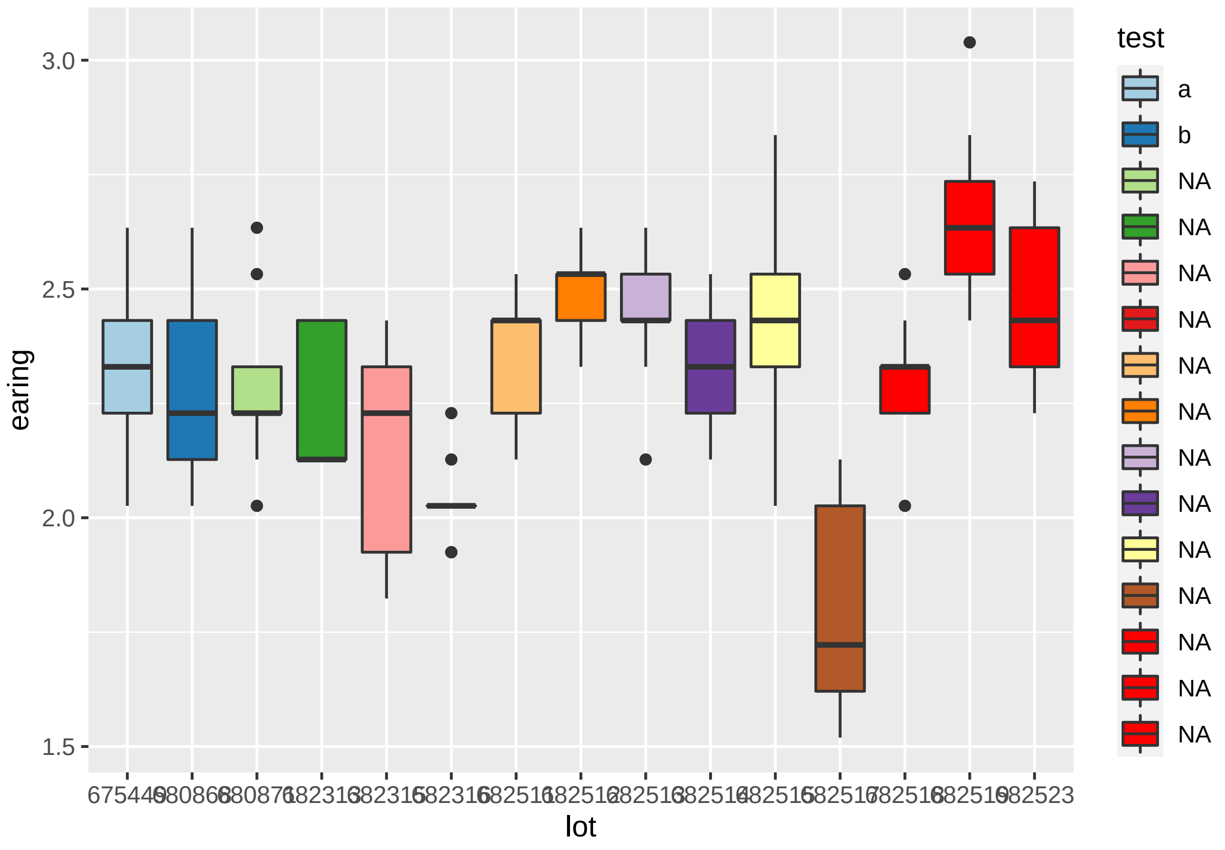
```r
# Highlight specific group
# (1) specify item in mapping
plot <- ggplot(Earing) +
  geom_boxplot(aes(lot,earing,fill = lot == 682517),alpha = 0.5) ; plot
```



```
# (2) add another layer include highlighted groups
HL_lot_num <- c("682517","682519","682523")

Highlight <- Earing %>%
        filter(lot %in% HL_lot_num)

plot <- ggplot(Earing) +
  geom_boxplot(aes(lot,earing),alpha = 0.5,fill = "#a1d99b") +
  geom_boxplot(aes(lot,earing),data = Highlight, fill = "#fb9a99"); plot
```

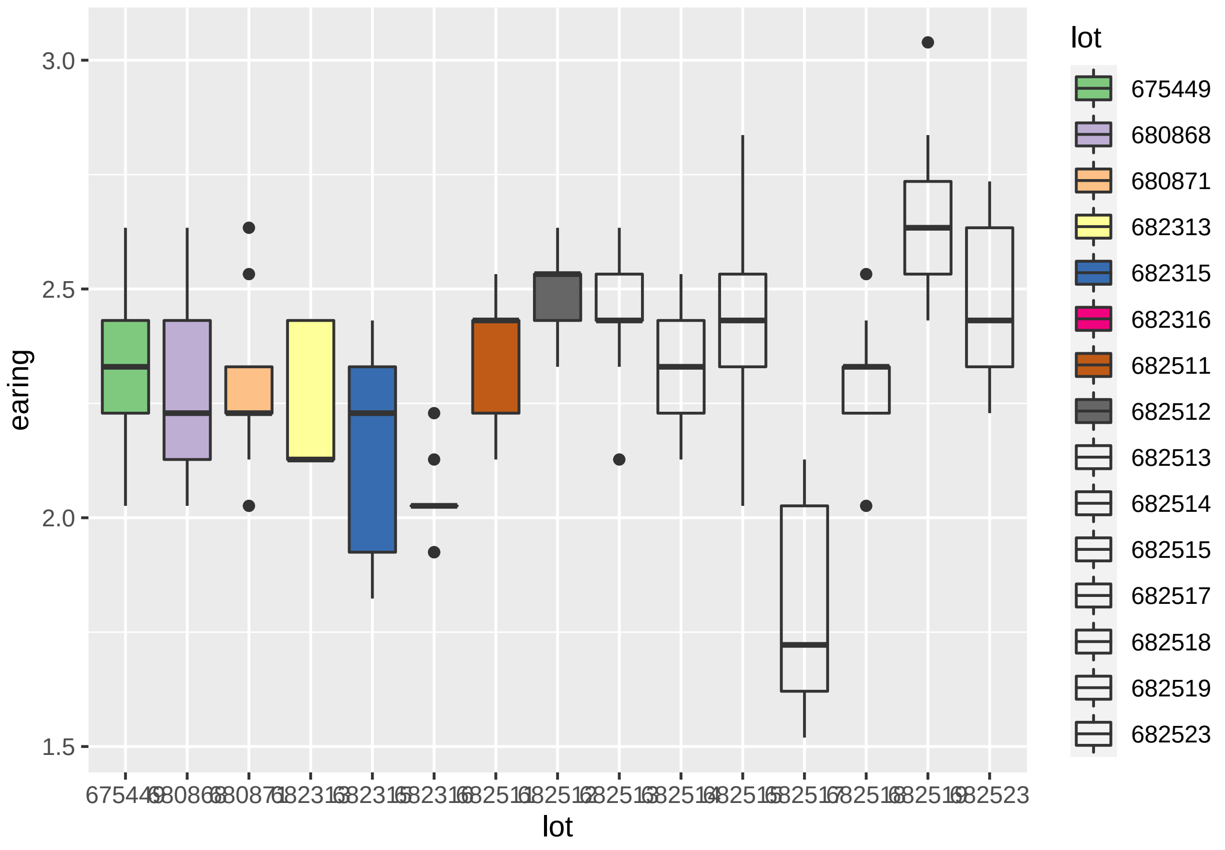


```r
# "alpha" = opacity

# scale_color/fill

# scale_color_manual() : to use custom colors
# scale_color_brewer() : to use color palettes from RColorBrewer package
# scale_color_grey() : to use grey color palettes
# Have same functions for fill
# scale_fill_brewer
# scale_fill_manual
# scale_fill_grey

# Manual; can also alter the legend through manual function
plot <- ggplot(Earing,aes(lot,earing)) +
  geom_boxplot(aes(fill = lot))+
  scale_fill_manual(
    values = c('#a6cee3','#1f78b4','#b2df8a','#33a02c','#fb9a99'
               ,'#e31a1c','#fdbf6f','#ff7f00','#cab2d6','#6a3d9a',
               '#ffff99','#b15928',"red","red","red","red","red"),
    name="test",
    labels=c("a","b")); plot
```

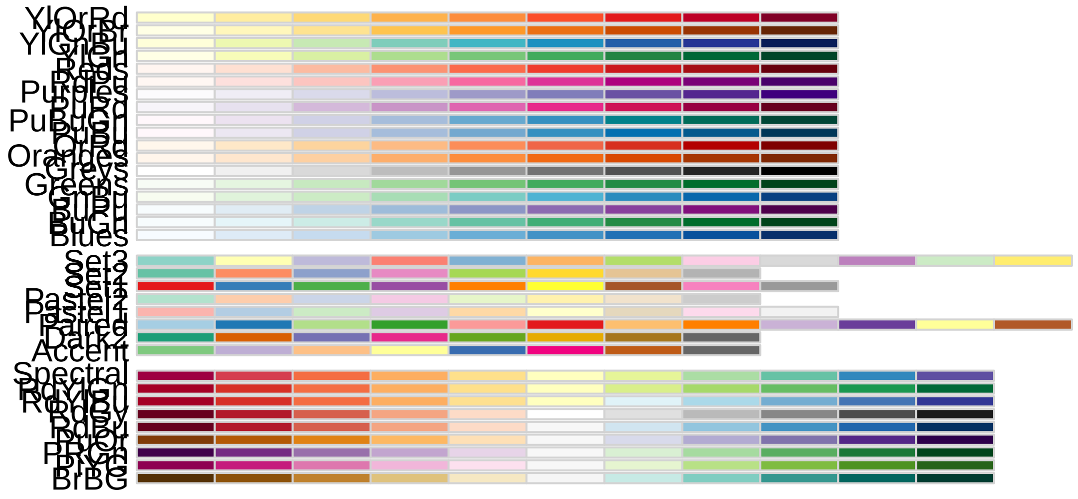


```r
# Brewer; Since there are too many groups here, there are some boxs without fill.
plot <- ggplot(Earing,aes(lot,earing)) +
  geom_boxplot(aes(fill = lot))+
  scale_fill_brewer(type = "qual",palette = "Accent");plot
```

```
## Warning in RColorBrewer::brewer.pal(n, pal): n too large, allowed maximum for palette Accent is 8
## Returning the palette you asked for with that many colors
```

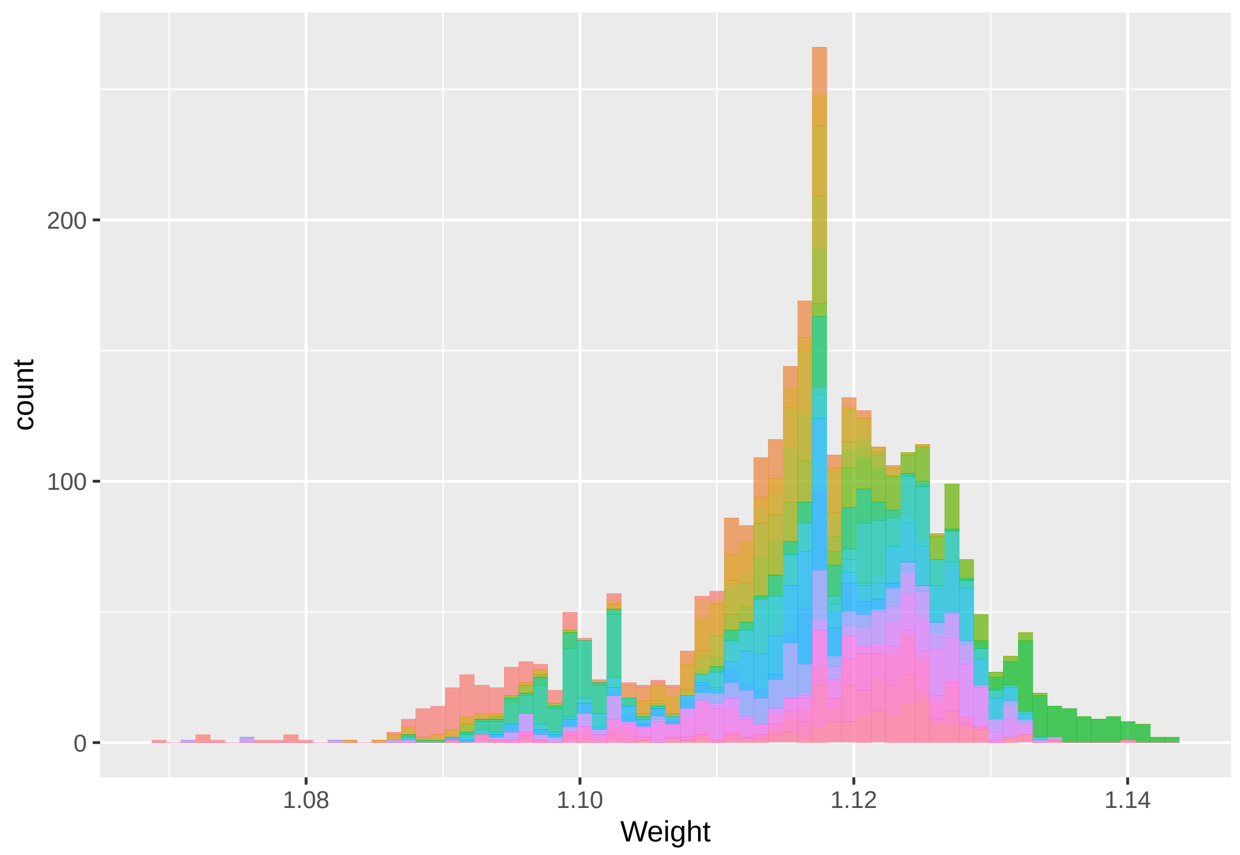


```
# Palette: Check what palette included in the package
RColorBrewer::display.brewer.all()
```

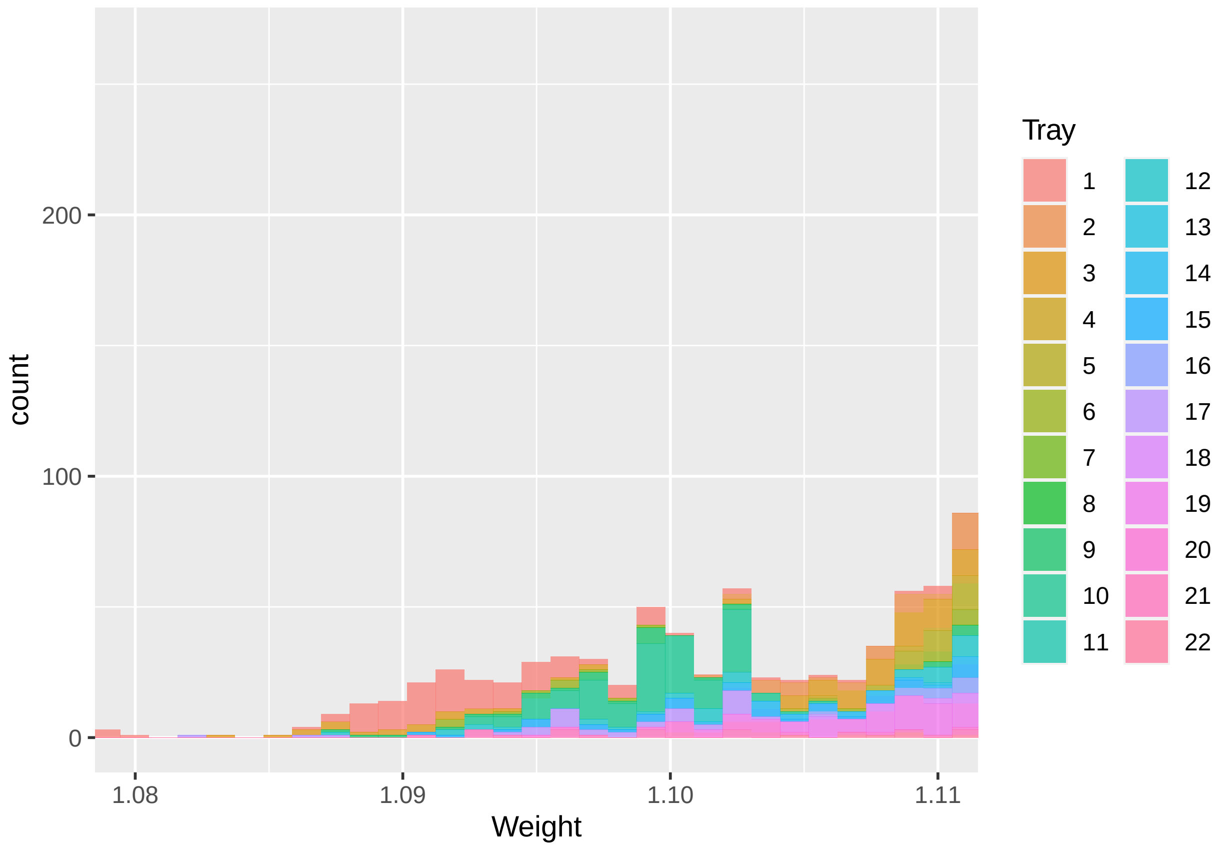


Coordinate / faceting

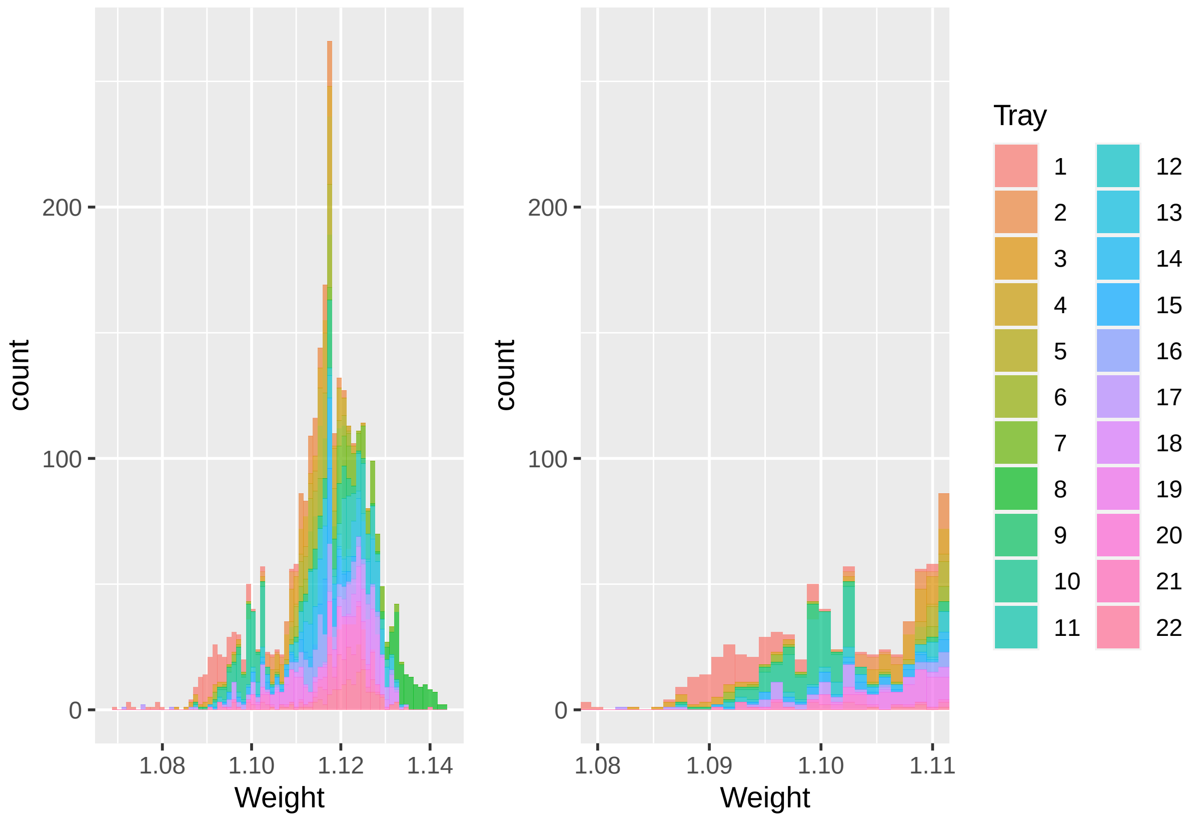
```r
# Original histogram; turn off the legend since I am going to combine 2 plots with the same legend.
hist <- ggplot(jellybeans) +
  geom_histogram(aes(Weight,fill=Tray),alpha = 0.7,bins = 70)+
  theme(legend.position = "none");hist
```



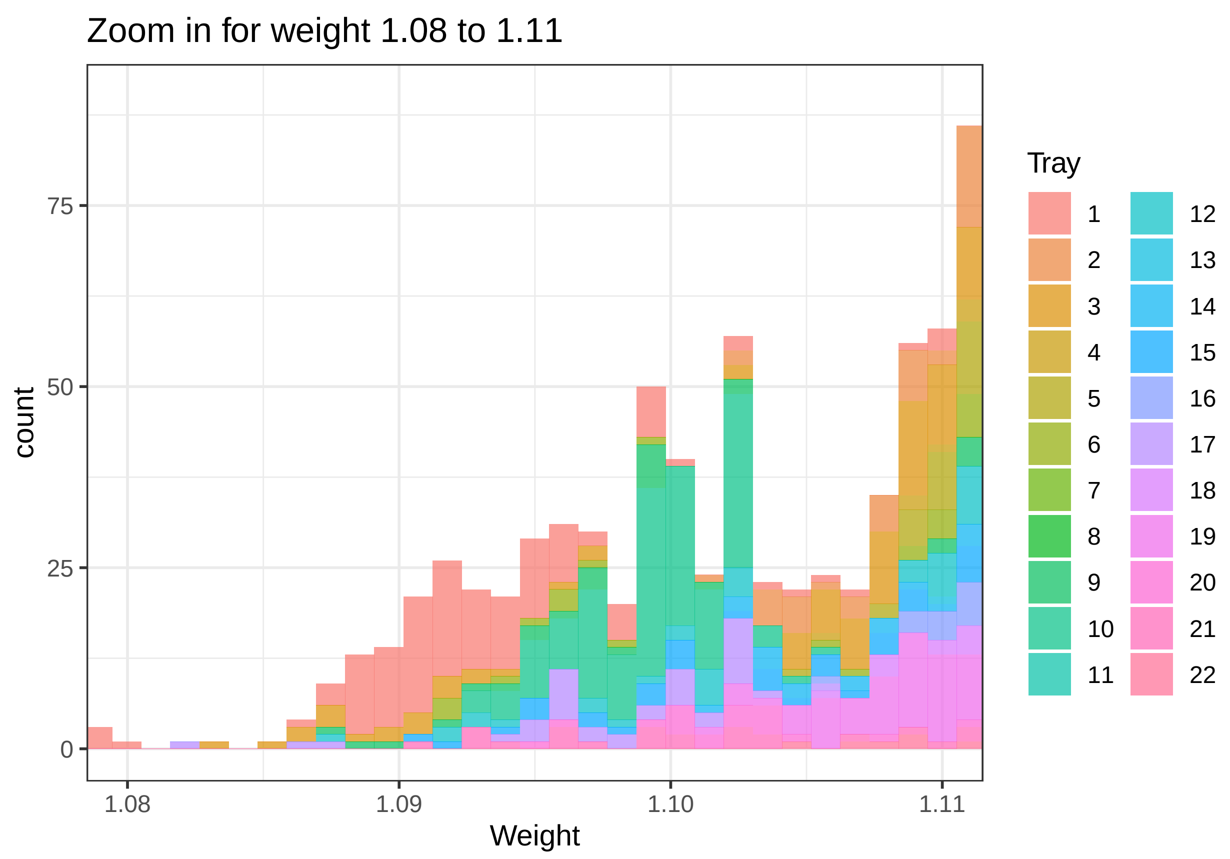
```
# adjust coordinate; use it like zooming in within certain range
hist_zoom <- ggplot(jellybeans) +
  geom_histogram(aes(Weight,fill=Tray),alpha = 0.7,bins = 70)+
  coord_cartesian(xlim=c(1.08,1.11)); hist_zoom
```



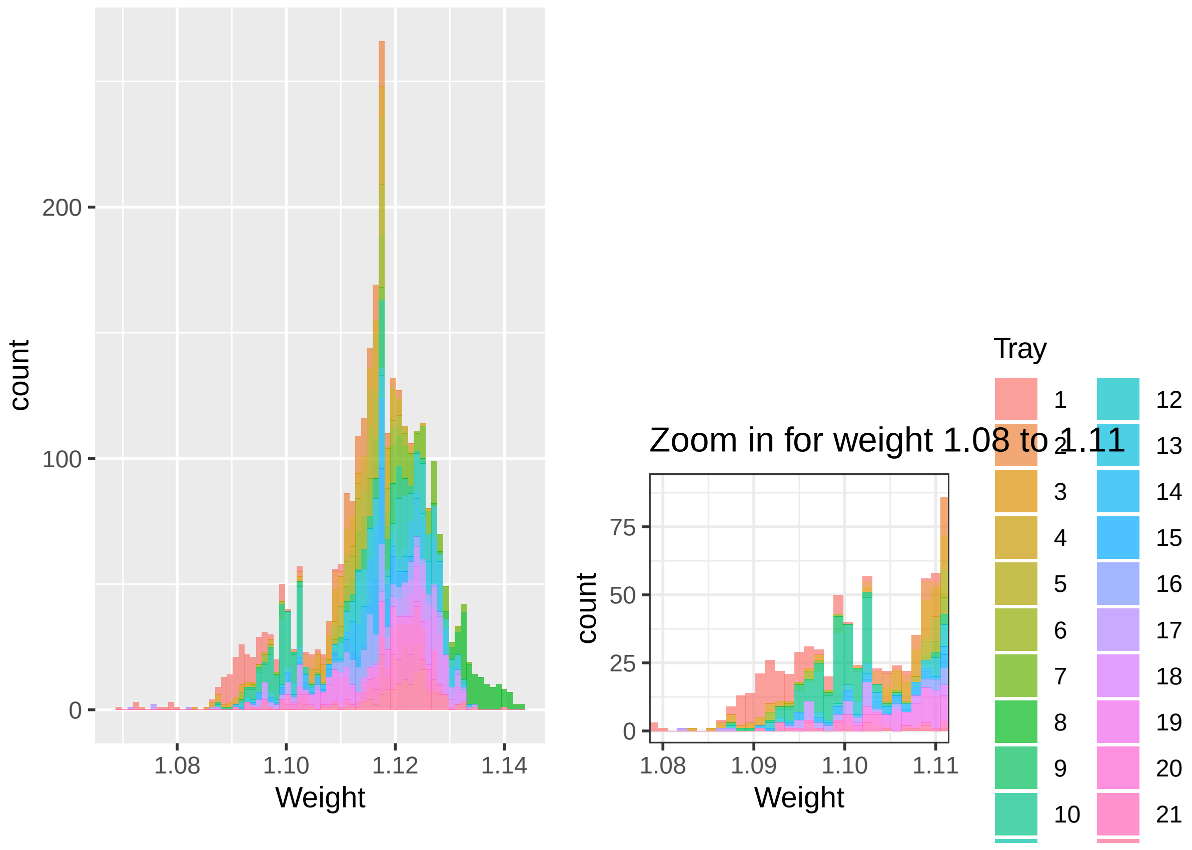
```
# Put it together
hist+hist_zoom
```



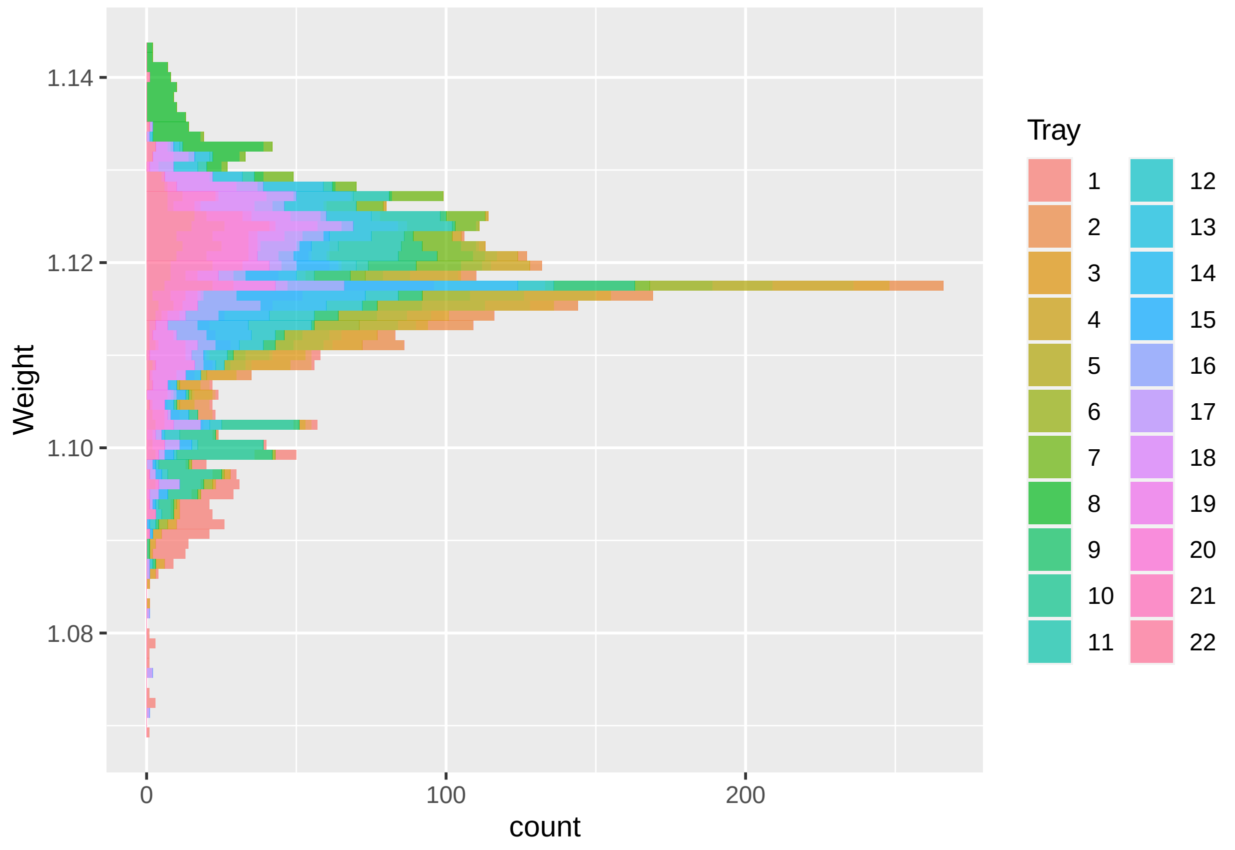
```
# Can use it with patchwork, use theme_bw for zoom plot and adjust y axis as well.
# step 1: make some adjustment for zoom plot first
# y axis adjustment / change background / add title
hist_zoom <- ggplot(jellybeans) +
  geom_histogram(aes(Weight,fill=Tray),alpha = 0.7,bins = 70)+
  coord_cartesian(xlim=c(1.08,1.11),ylim = c(0,90))+
  theme_bw()+
  labs(title = "Zoom in for weight 1.08 to 1.11"); hist_zoom
```



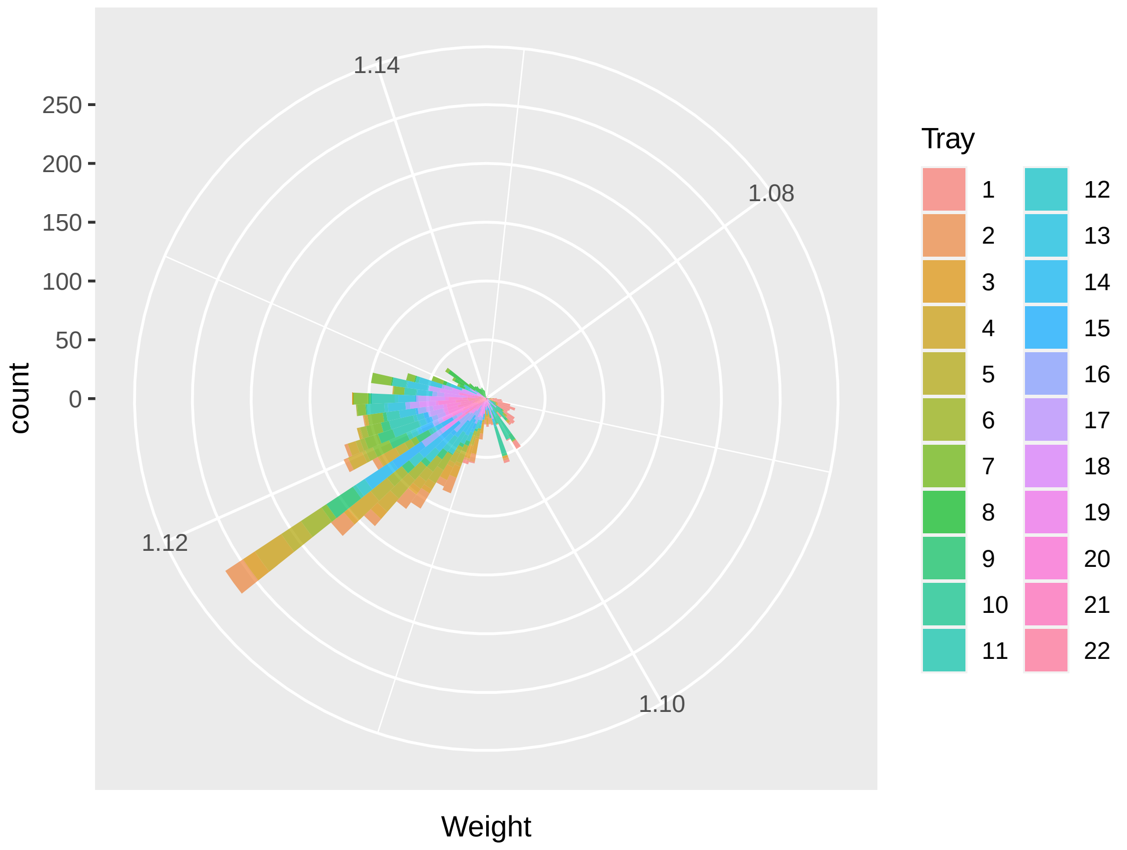
```
# Step 2 layout and composition
layout <- "
aaa##
aaa##
aaa##
aaabb
aaabb
"
# "#" means empty
hist+hist_zoom+plot_layout(design = layout)
```



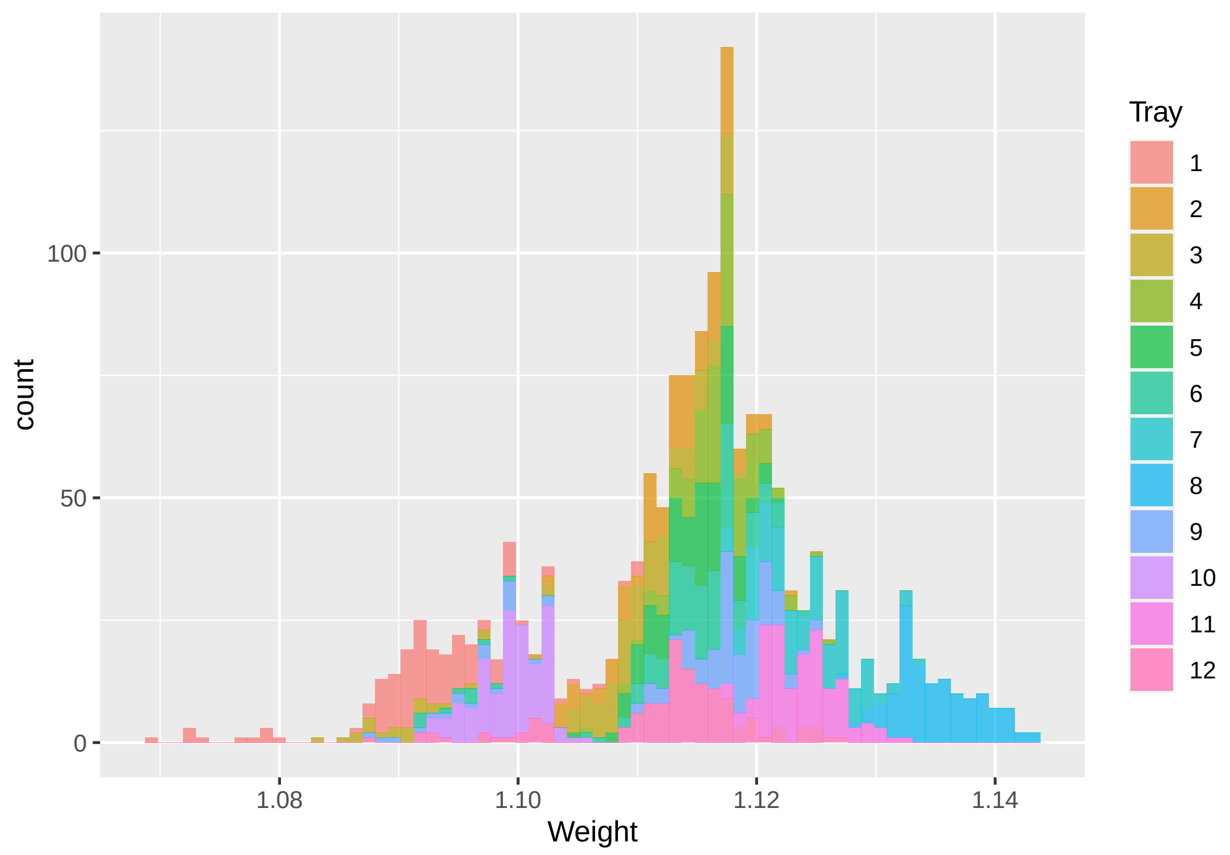
```
# Some other coordinate application
# flip
hist <- ggplot(jellybeans) +
  geom_histogram(aes(Weight,fill=Tray),alpha = 0.7,bins = 70)+
  coord_flip(); hist
```



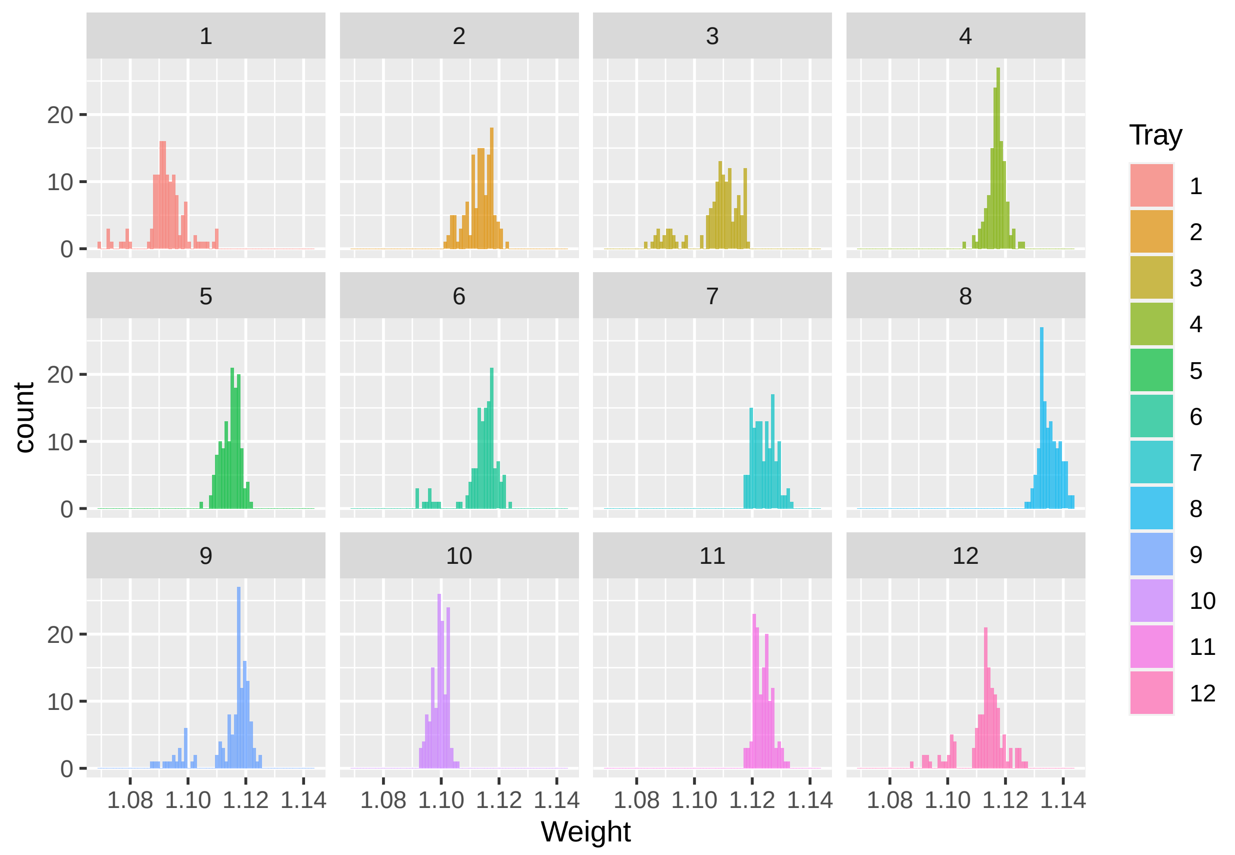
```
# polar coordinate
hist <- ggplot(jellybeans) +
  geom_histogram(aes(Weight,fill=Tray),alpha = 0.7,bins = 70)+
  coord_polar(); hist
```



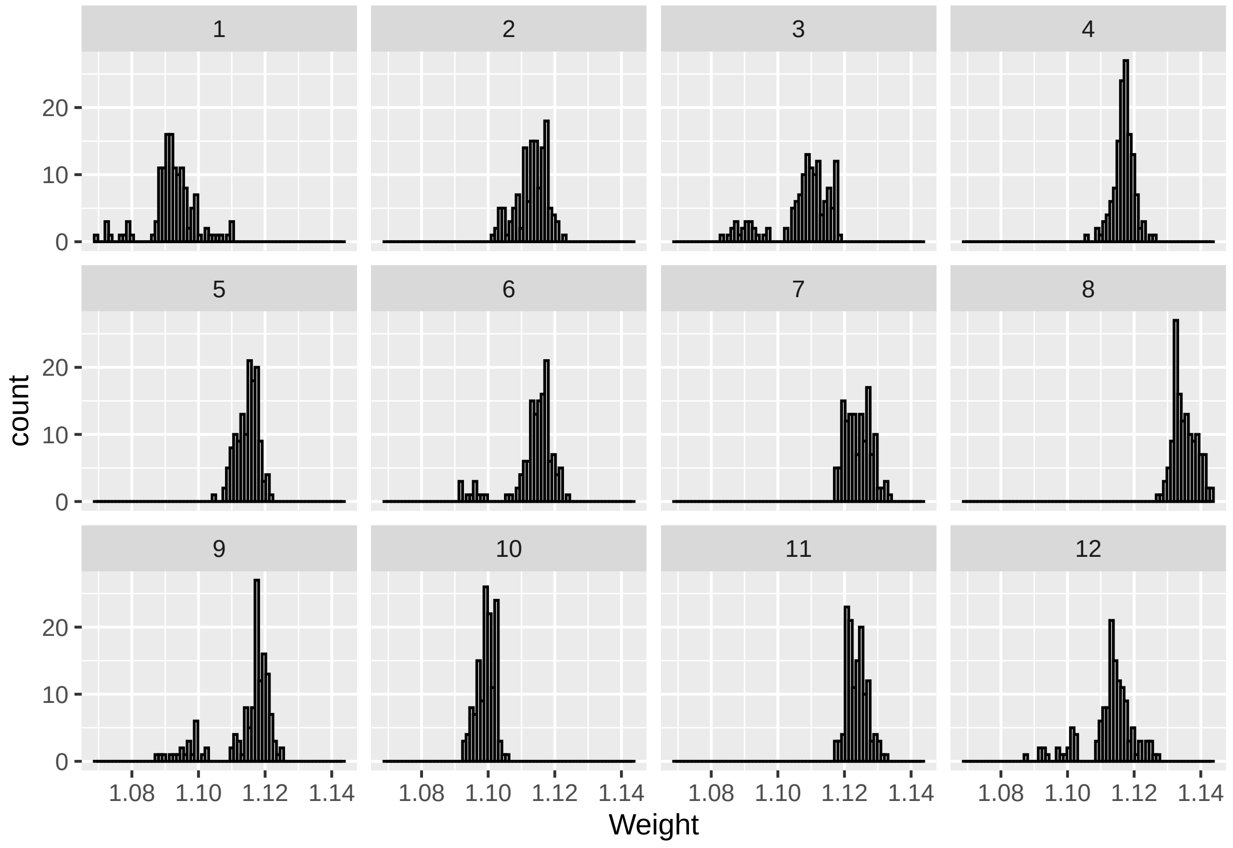
```
# Faceting; Take 12 Tray as example
jellybeans <- read.delim("C:/Users/asus/Desktop/CU boulder/Course/S22/EMEN 5042 Data Science for qualit

# jellybeans <- read.delim("~/Desktop/ggplot2/jellybeans.dat")
first12Tray <- jellybeans %>%
        filter(Tray <= 12)
first12Tray$Tray = as.factor(first12Tray$Tray)

hist <- ggplot(first12Tray) +
  geom_histogram(aes(Weight,fill=Tray),alpha = 0.7,bins = 70) ; hist
```



```
# facet_wrap
hist <- ggplot(first12Tray) +
  geom_histogram(aes(Weight,fill=Tray),alpha = 0.7,bins = 70) +
  facet_wrap(vars(Tray)); hist
```

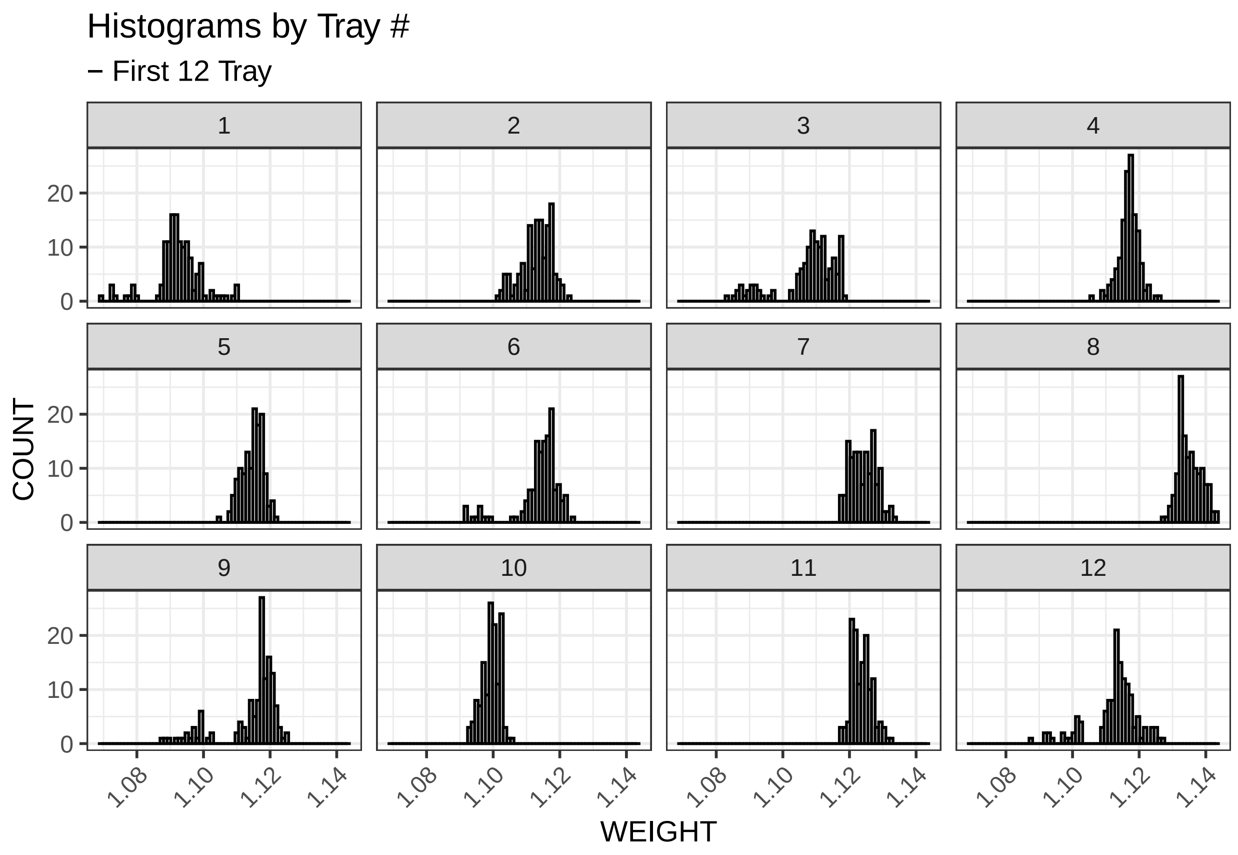


```r
# No need to use color mapping with faceting, IMO.
# Thus I use black/white to plot bars
hist <- ggplot(first12Tray) +
  geom_histogram(aes(Weight),fill = "white",color = "black",bins = 70) +
  facet_wrap(vars(Tray)); hist
```



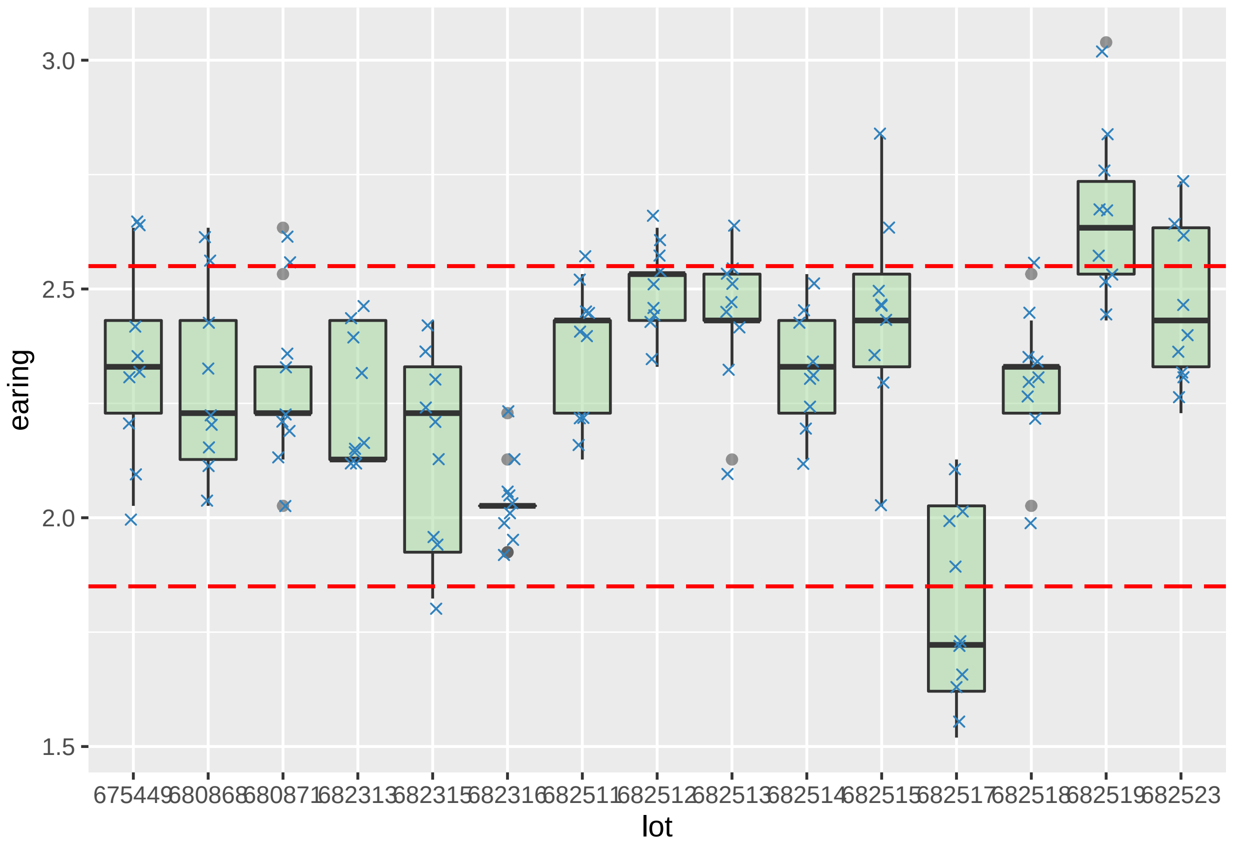
Title/Labels

```
# labs()
hist+
  labs(x = "WEIGHT",y= "COUNT",
      title = "Histograms by Tray #",
      subtitle = "- First 12 Tray",)+
  theme_bw()+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

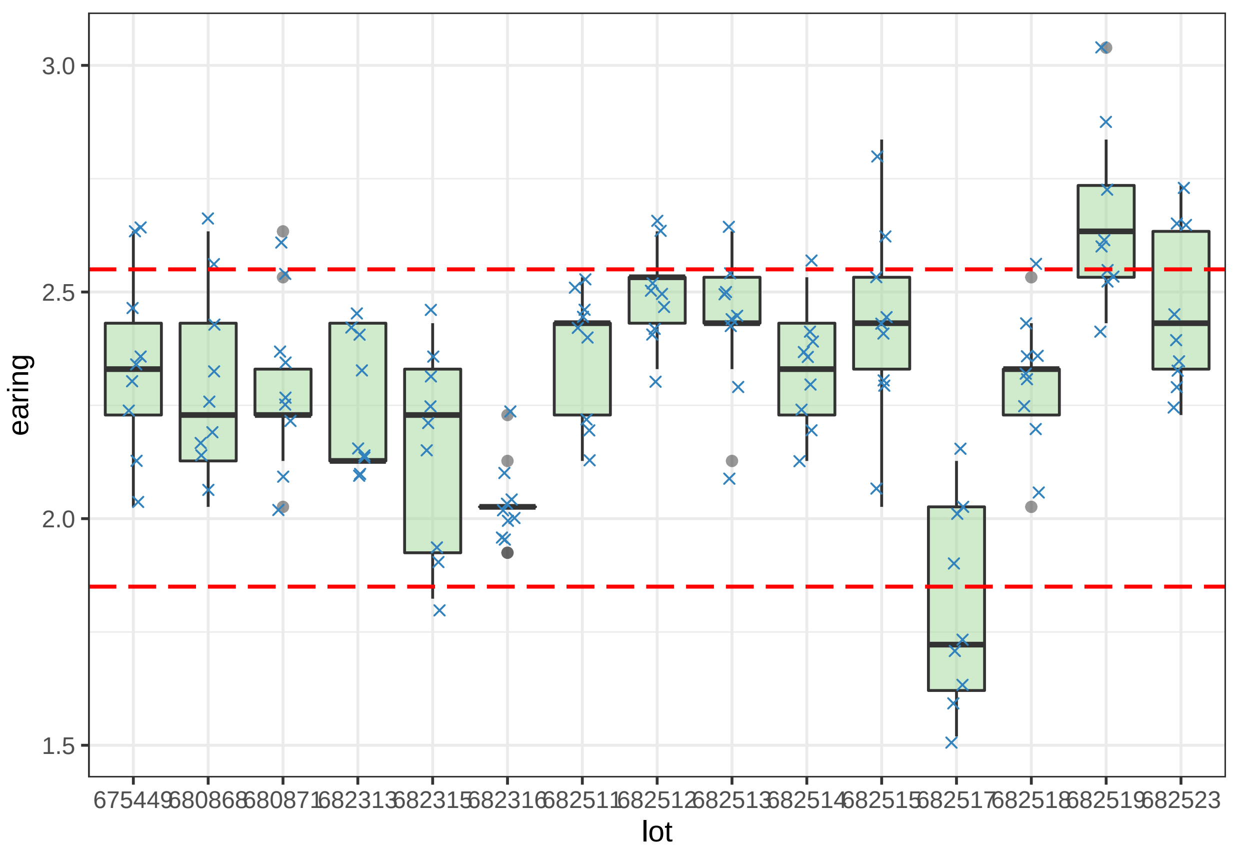


Theme

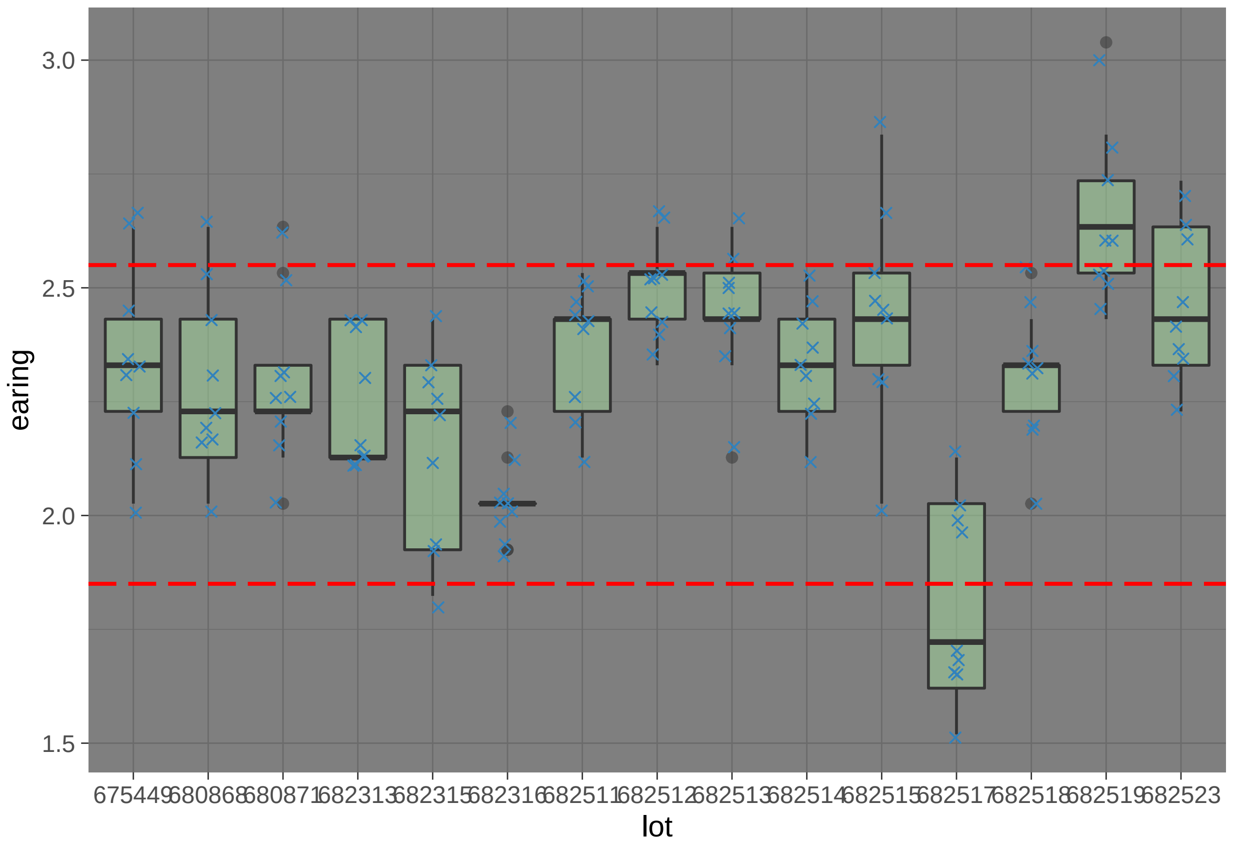
```
# Adjust background
t <- plot <- ggplot(Earing) +
  geom_boxplot(aes(lot,earing),fill = "#a1d99b",alpha = 0.5)+
  geom_jitter(aes(lot,earing),width = 0.1,shape = 4,color = "#3182bd")+
  geom_hline(aes(yintercept = 1.85),color = "red",size = 0.7,linetype ="longdash")+
  geom_hline(aes(yintercept = 2.55),color = "red",size = 0.7,linetype ="longdash");t
```


3.0
2.5
2.0
1.5
earing
675449
680868
680871
682313
682315
682316
682511
682512
682513
682514
682515
682517
682518
682519
682523
lot

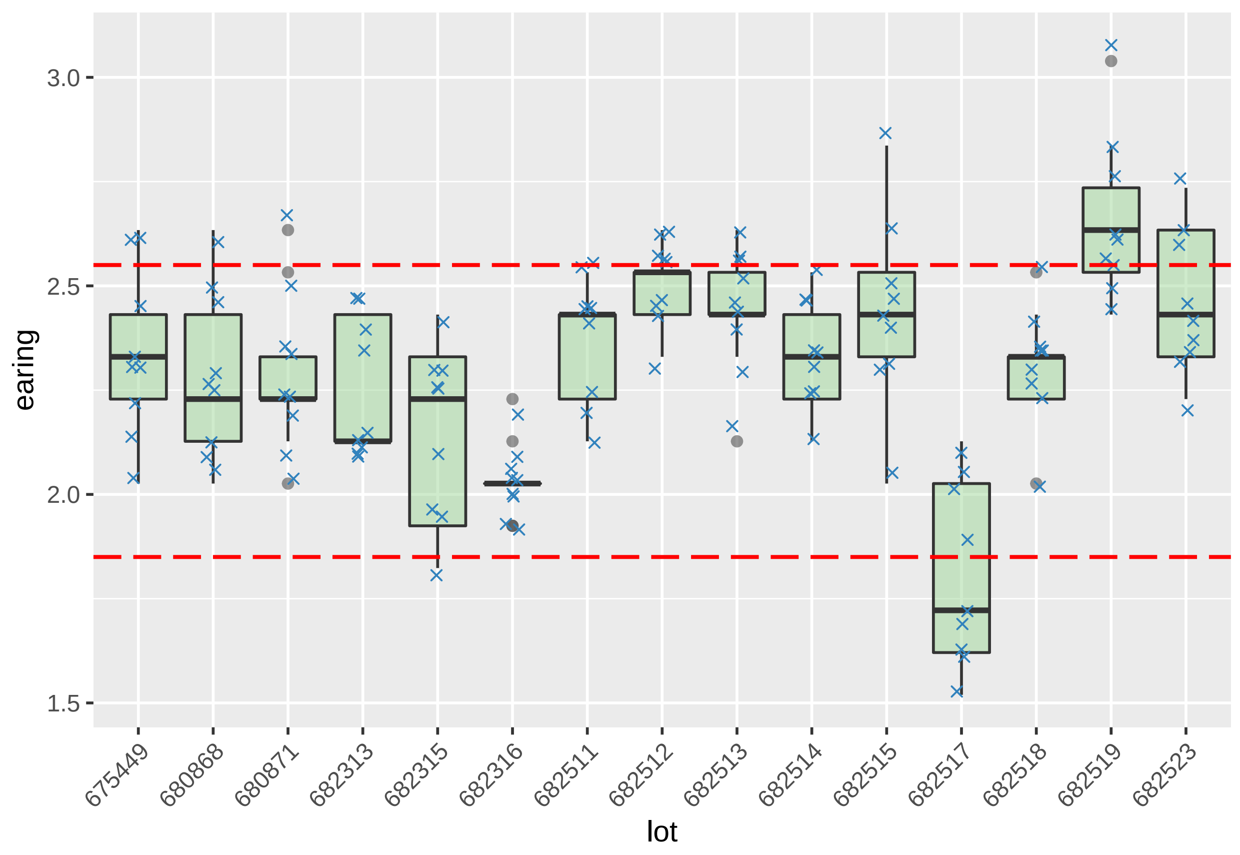
```
t + theme_bw()
```


3.0
2.5
2.0
1.5
earing
675449
680868
680871
682313
682315
682316
682511
682512
682513
682514
682515
682517
682518
682519
682523
lot

```
t + theme_dark()
```

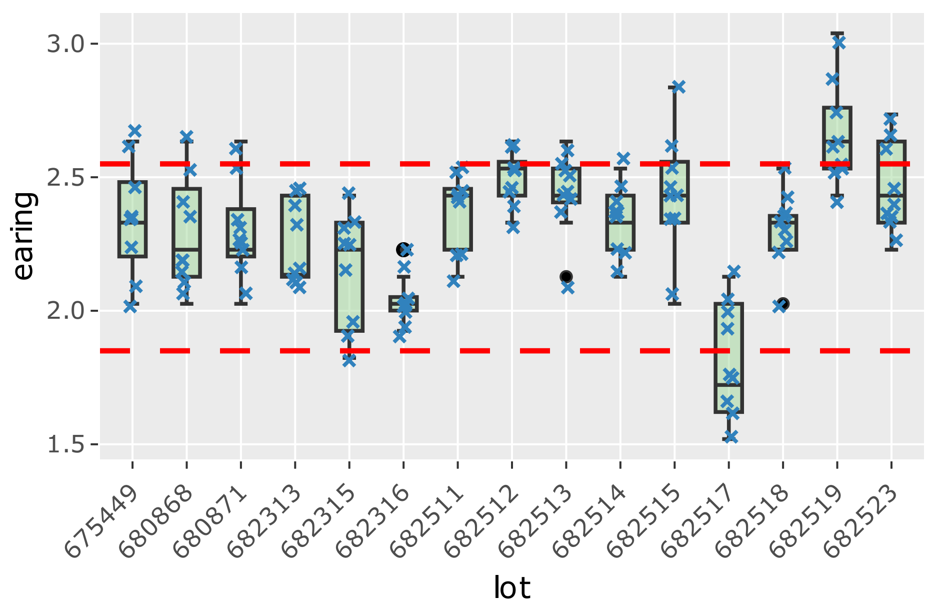


```
# rotate x-axis label
# There are many setting can be manipulate in the theme()
# try ?theme() for the document
t + theme(axis.text.x = element_text(angle = 45, hjust = 1) )
```

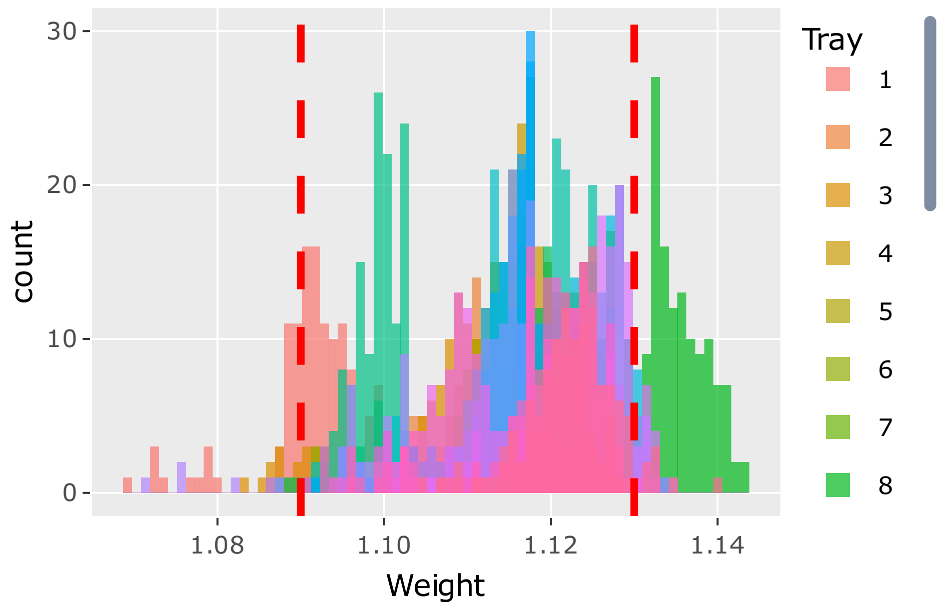


Interaction

```
# interaction
lot_boxplot <- t + theme(axis.text.x = element_text(angle = 45, hjust = 1) )
ggplotly(lot_boxplot)
```

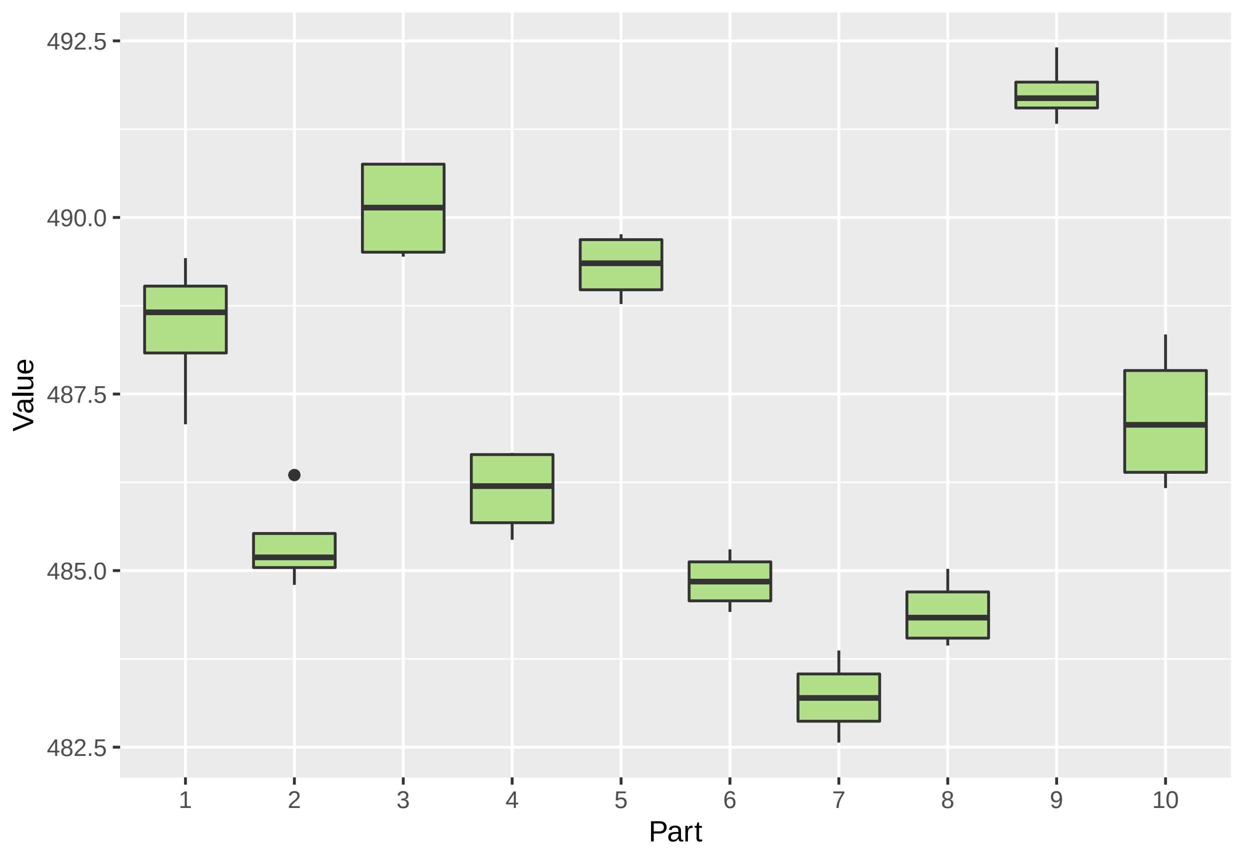


```
jellybeans$Sample <- as.factor(jellybeans$Sample)
jellybeans$Tray <- as.factor(jellybeans$Tray)


hist <- ggplot(jellybeans) +
  geom_histogram(aes(Weight,fill=Tray),alpha = 0.7, position = "identity",bins = 70)+
  geom_vline(xintercept = c(1.09,1.13),color = "red" , linetype = "longdash",size =1);

ggplotly(hist)
```

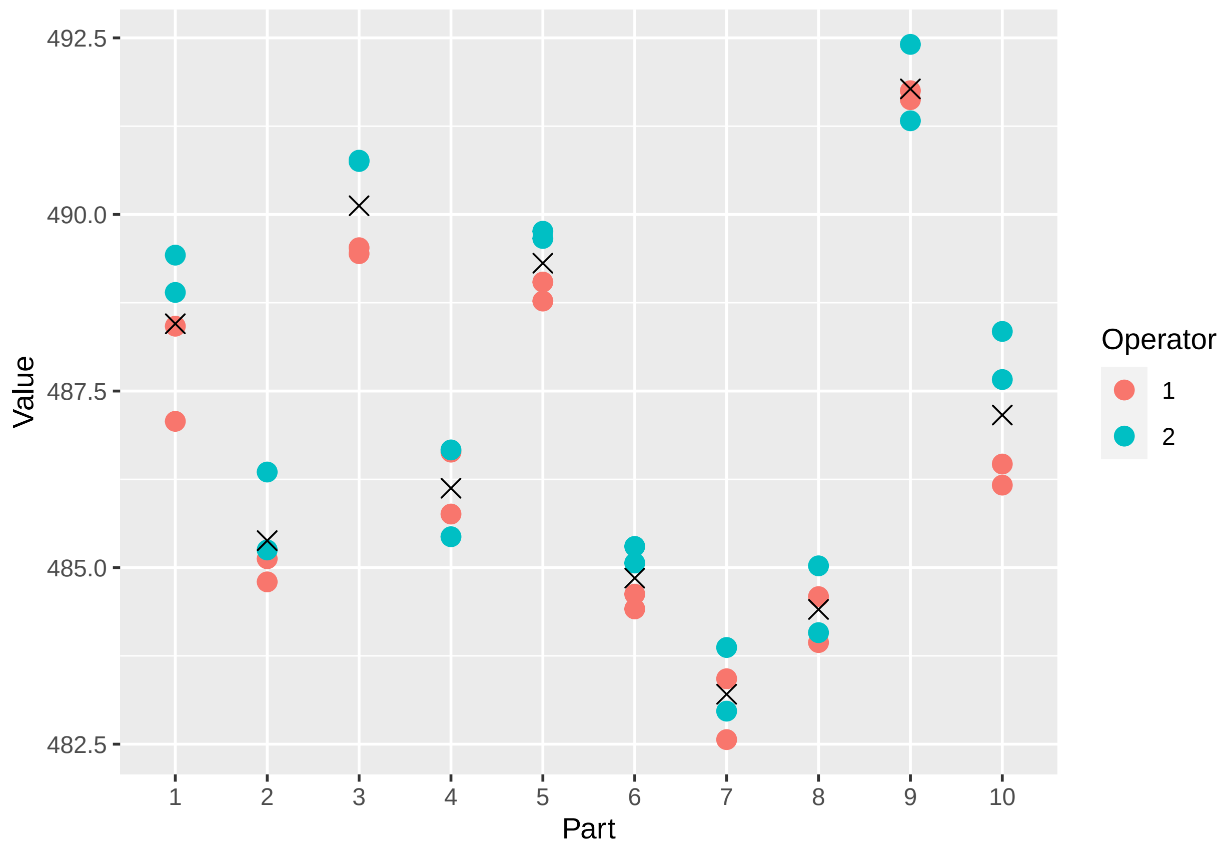


HW 10 Vis

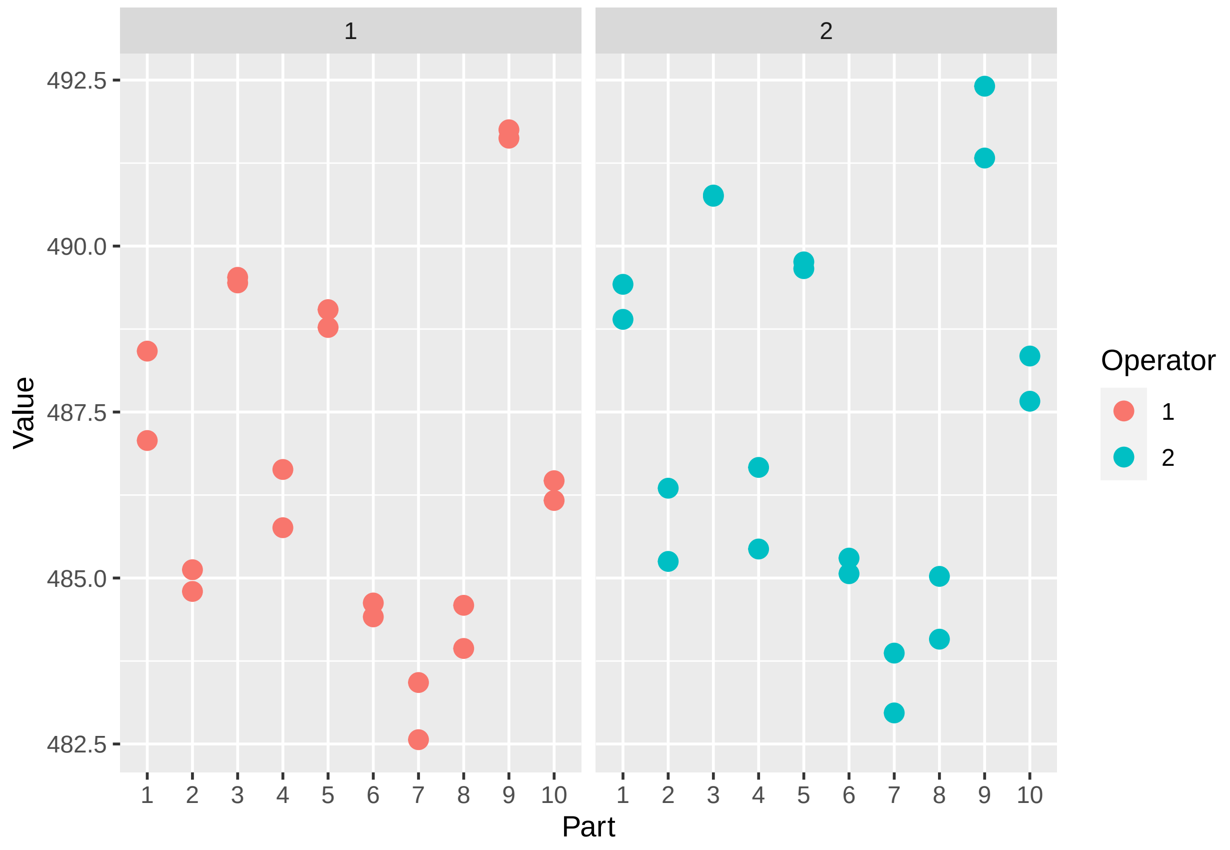
```r
height$Part <- as.factor(height$Part)
height$Operator <- as.factor(height$Operator)
height$Repetition <- as.factor(height$Repetition)

a <- ggplot(data = height,aes(Part,Value))

p1 <-  a + geom_boxplot(fill = "#b2df8a");p1
```



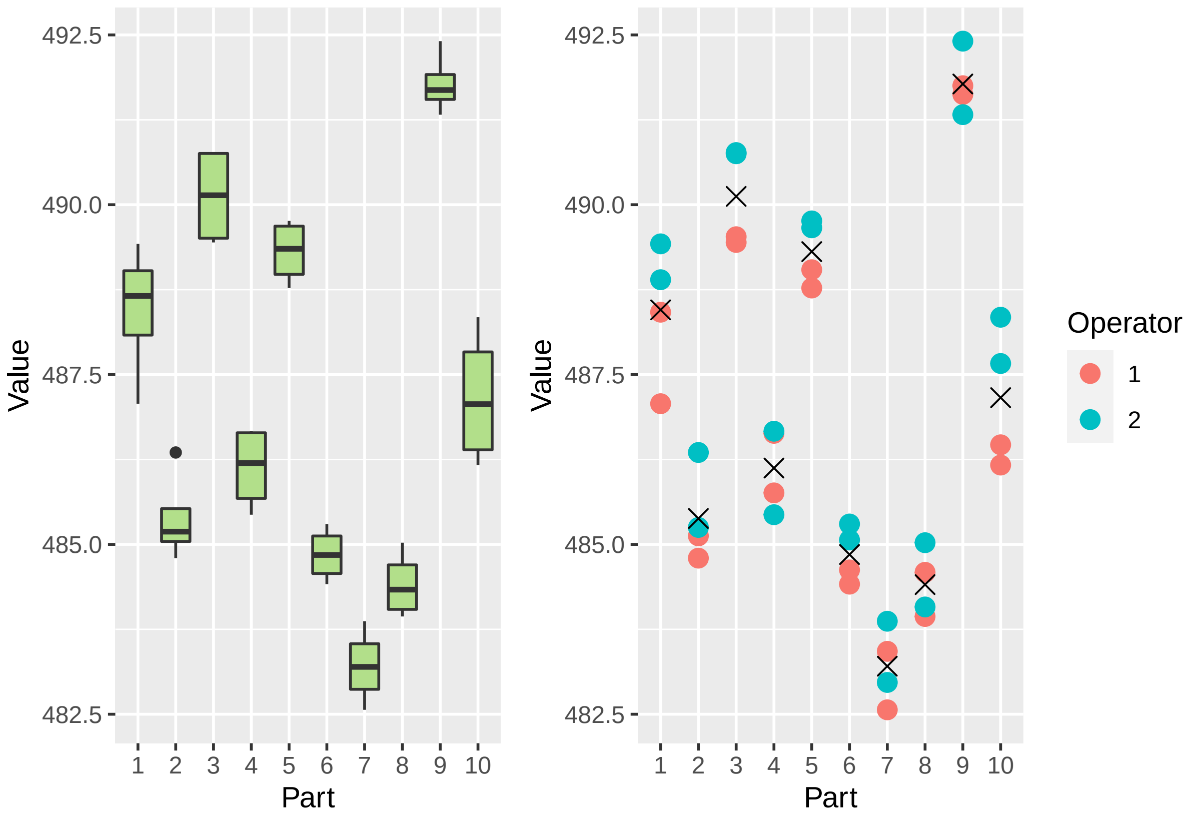
```
p2 <-  a +
  geom_point(aes(color = Operator),size = 3) +
  stat_summary(fun = "mean",geom="point",shape =4,size =3);p2
```



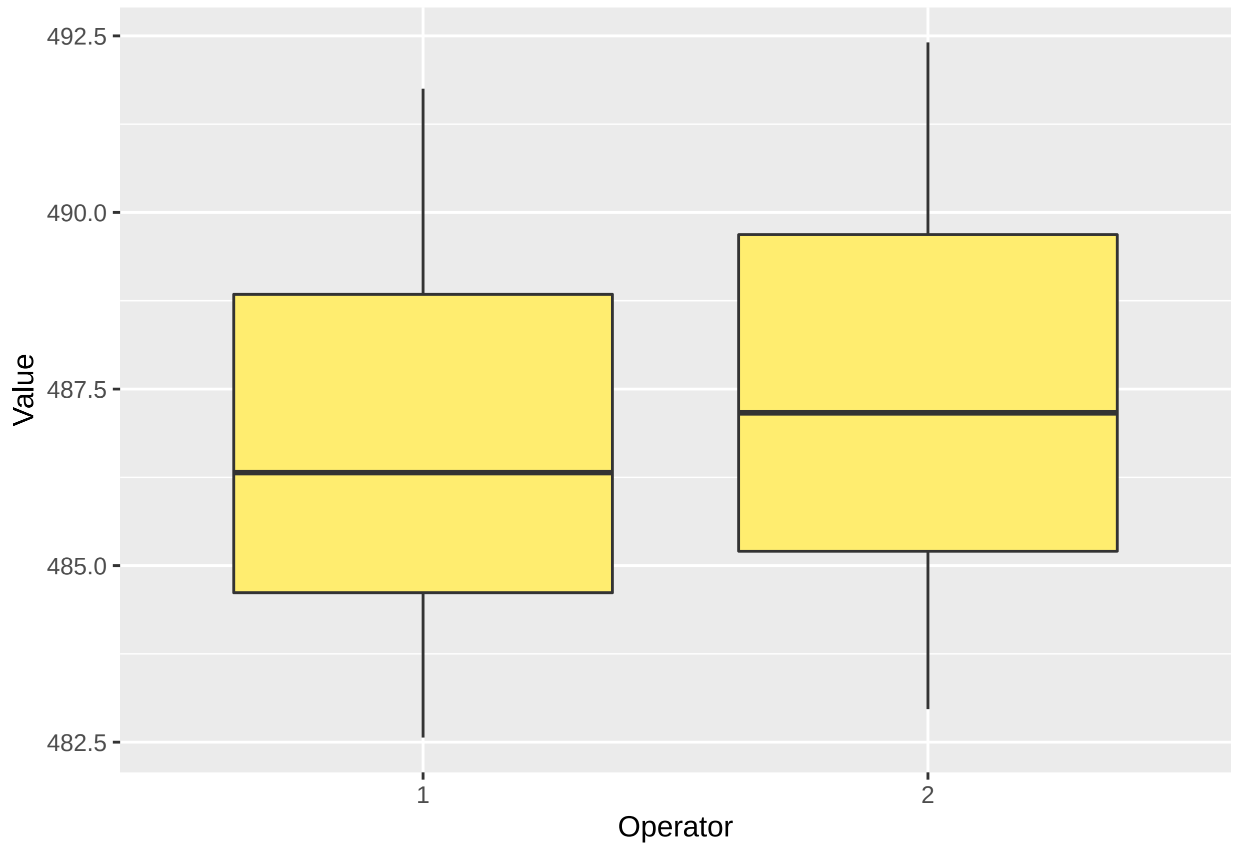
```
# Faceting
p3 <-  a +
  geom_point(aes(color = Operator),size = 3) +
  facet_grid(cols=vars(Operator));p3
```



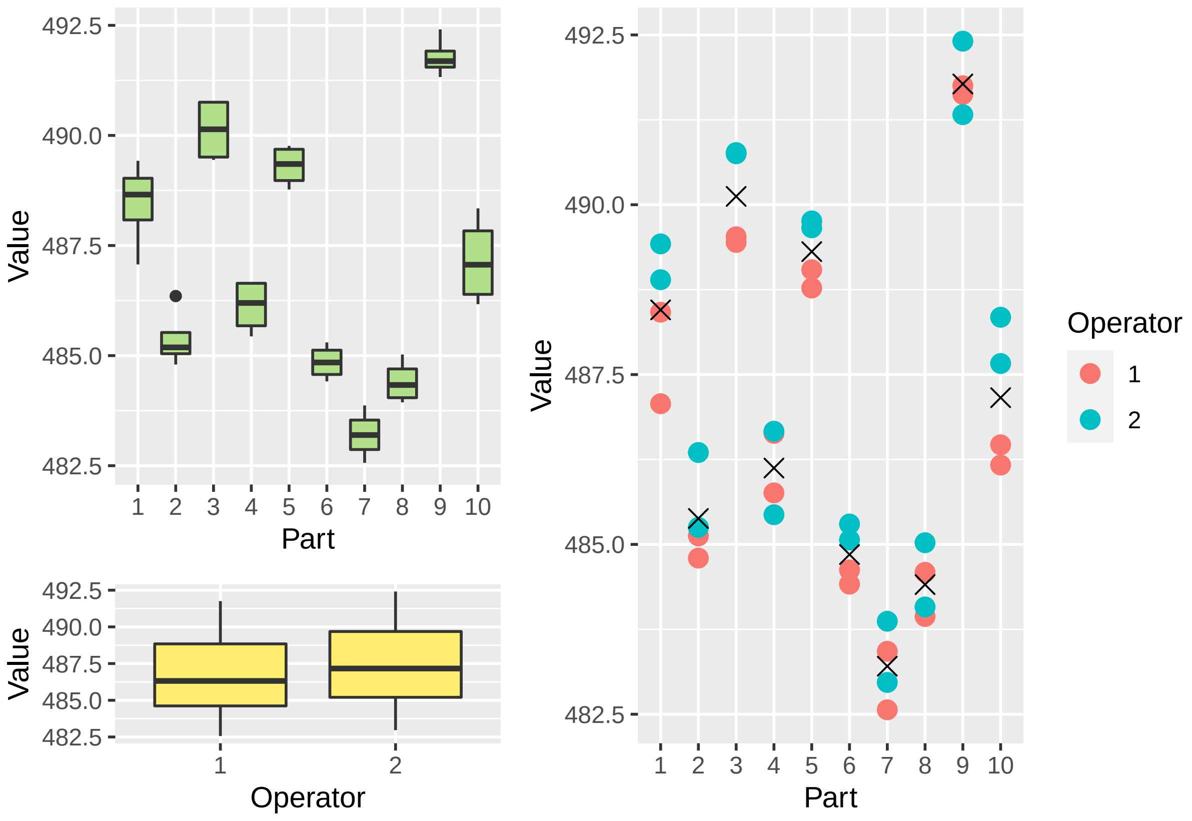
```
# Install package first
p1+p2
```



```
# define layout
layout <- "
aaaccc
aaaccc
aaaccc
bbbccc
"
p4 <- ggplot(data = height,aes(Operator,Value))+
  geom_boxplot(fill = "#ffed6f");p4
```



```
p1+p4+p2+plot_layout(design = layout)
```



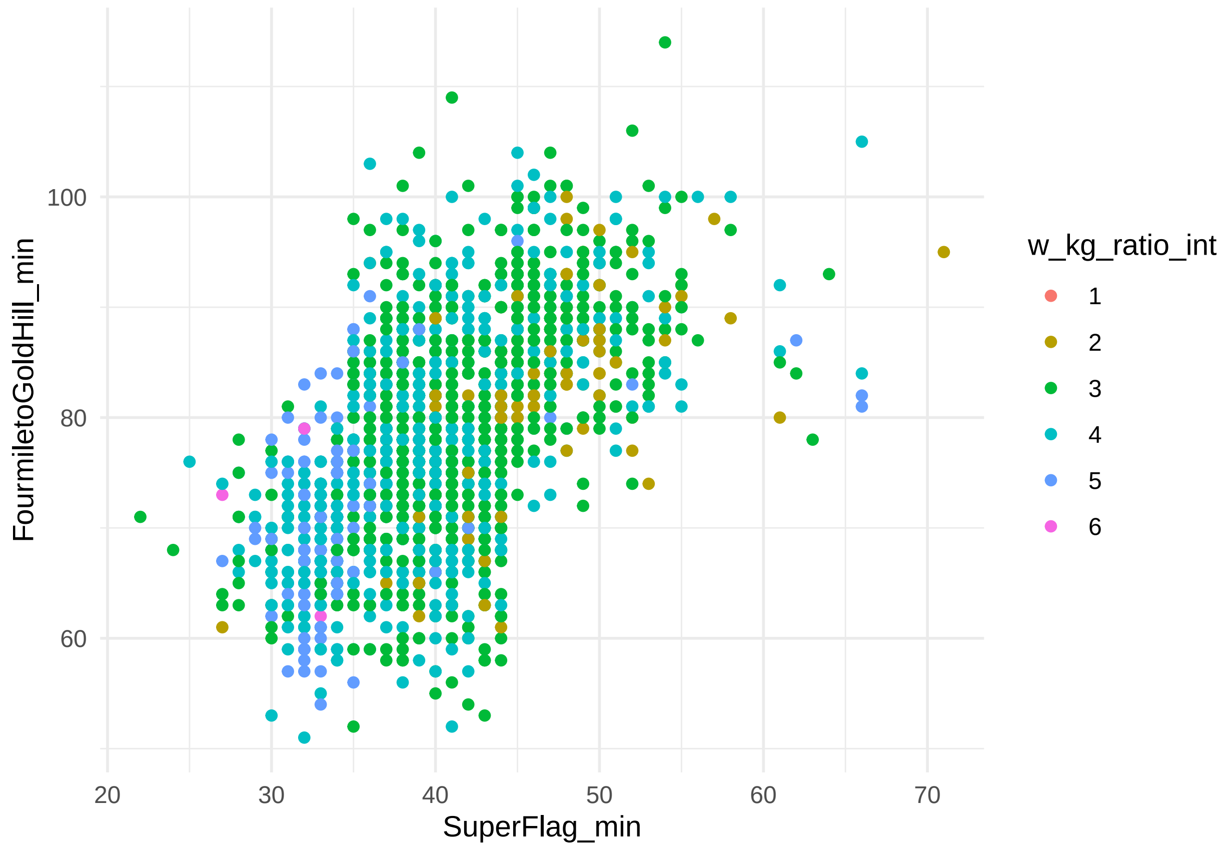
Some other visualization - Strava segments analysis.

I collect results of 3 segments from 1700 riders thru strava api and visualize the data to see if they are related. Also got the power-to-weight ratio (Higher means stronger) and used them mapping with color. I am trying to use this plot to predict the results. Like if someone can finish segment A in 20 mins, he might make it around 45 mins for segment B.

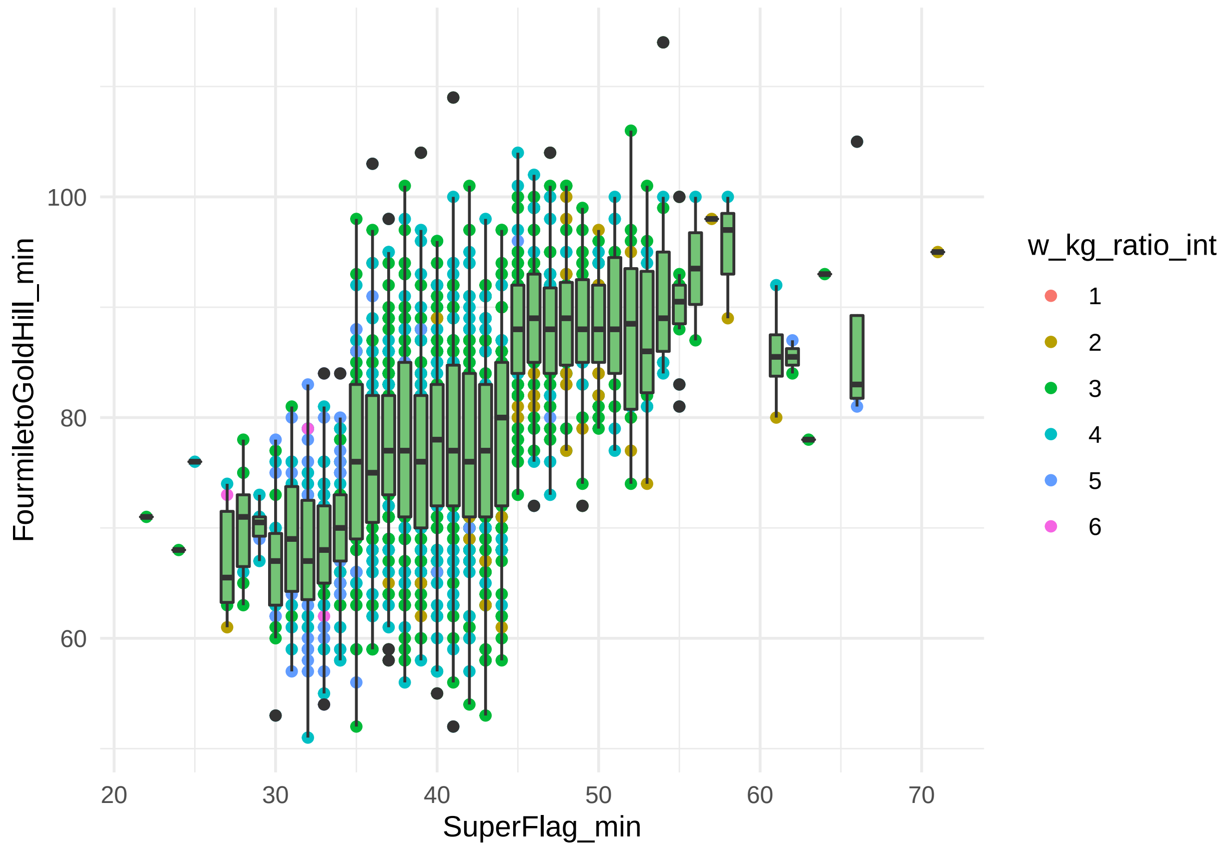
```
"  Variables:
segment$rider
segment$SunshineHillclimbChallengeCourse_sec
segment$FourmiletoGoldHill_sec
segment$SuperFlag_sec
segment$w_kg_ratio
segment$SunshineHillclimbChallengeCourse_min
segment$FourmiletoGoldHill_min
segment$SuperFlag_min
"
```

```
## [1] "  Variables:\nsegment$rider\nsegment$SunshineHillclimbChallengeCourse_sec\nsegment$FourmiletoGol
```

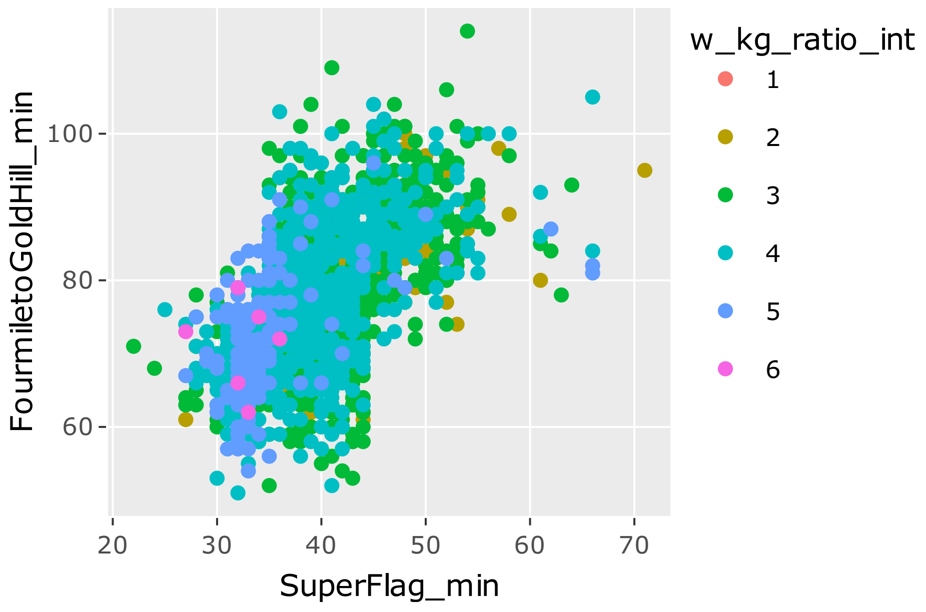
```
# point
ggplot(segment)+
  geom_point(aes(SuperFlag_min,FourmiletoGoldHill_min,color = w_kg_ratio_int))+
    theme_minimal()
```



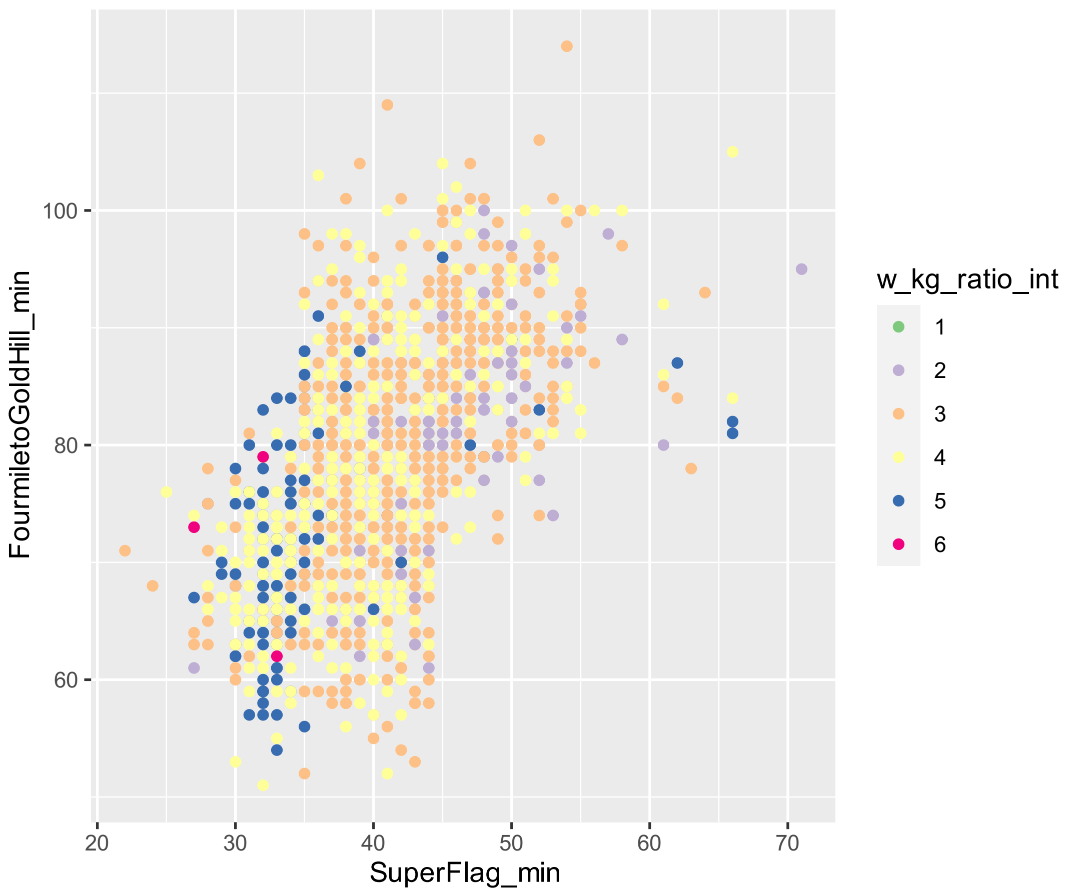
```r
# boxplot
ggplot(segment)+
  geom_point(aes(SuperFlag_min,FourmiletoGoldHill_min,color = w_kg_ratio_int))+
  geom_boxplot(aes(SuperFlag_min,FourmiletoGoldHill_min,group = SuperFlag_min),fill = "#74c476")+
  theme_minimal()
```



```
# interaction
interaction_plot <- ggplot(segment)+
  geom_point_interactive(aes(SuperFlag_min,FourmiletoGoldHill_min,color=w_kg_ratio_int,
                                  tooltip =rider,data_id=w_kg_ratio_int))+
  scale_color_brewer(type = "qual",palette = "Accent")

# Highlight by w/kg group
test_p <- ggplot(segment)+
  geom_point(aes(SuperFlag_min,FourmiletoGoldHill_min,color=w_kg_ratio_int),size = 1.5)

# filter
ggplotly(test_p)
```



```r
# interaction, hovering
girafe(ggobj=interaction_plot)
```



Reference:

- Manual:

  `https://cran.microsoft.com/snapshot/2015-01-06/web/packages/ggplot2/ggplot2.pdf`

- sorting:

  `https://www.statology.org/order-bars-ggplot2-bar-chart/#:~:text=By%20default%2C%20ggplot2%20bar%2`

- Color brewer:

  `https://colorbrewer2.org/#type=sequential&scheme=PuBuGn&n=8`

-Palette

```
> RColorBrewer::display.brewer.all()
```

- Scale color :

  ```
  https://ggplot2.tidyverse.org/reference/scale_brewer.html#palettes

  * type  = One of seq (sequential), div (diverging) or qual (qualitative)
  ```

- Interaction

  ```
  - package: plotly
  ```

- Animation

  ```
  - packages: gganimate / gifski / av
  ```