Coding club meet-up: Using ggplot2 for visualization

Lingzi Li

December 3 2019

Note of the day

There are several useful packages to check out:

UpSet: Visualizing Intersecting Sets (http://caleydo.org/tools/upset/)

Intervene: Intersection and visualization of multiple genomic region sets (https://intervene.readthedocs.io/en/latest/index.html)

ggplot2 extensions (https://www.ggplot2-exts.org/ggiraph.html)

DEGreport (https://lpantano.github.io/DEGreport/index.html)

The Anatomy of a plot

Load data

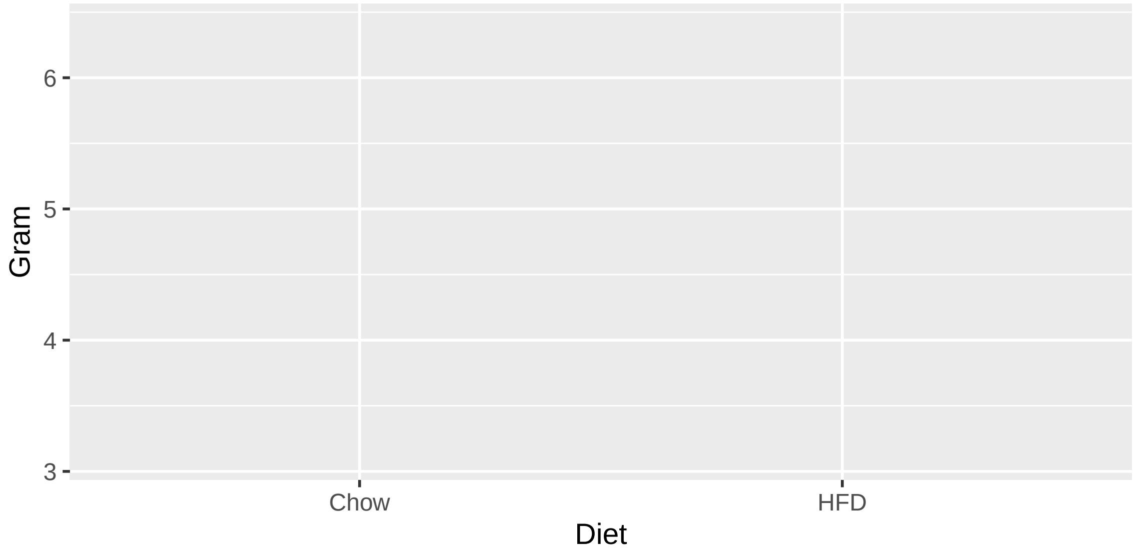
```
# Load packages
library(tidyverse)
library(readxl)
```

```
head(fatMass_Male)
```

```
## # A tibble: 6 x 7
##   Diet_duration Sex   Genotype      ID Diet  Composition  Gram
##   <chr>         <chr> <chr>      <dbl> <fct> <chr>       <dbl>
## 1 1 week        Male  GcgcreTRAP  1866 Chow  Fat           3.1
## 2 1 week        Male  TRAP        1867 Chow  Fat           3.3
## 3 1 week        Male  TRAP        1868 Chow  Fat           3.1
## 4 1 week        Male  GcgcreTRAP  1869 HFD   Fat           3.9
## 5 1 week        Male  TRAP        1872 HFD   Fat           4.4
## 6 1 week        Male  TRAP        1870 HFD   Fat           4.3
```

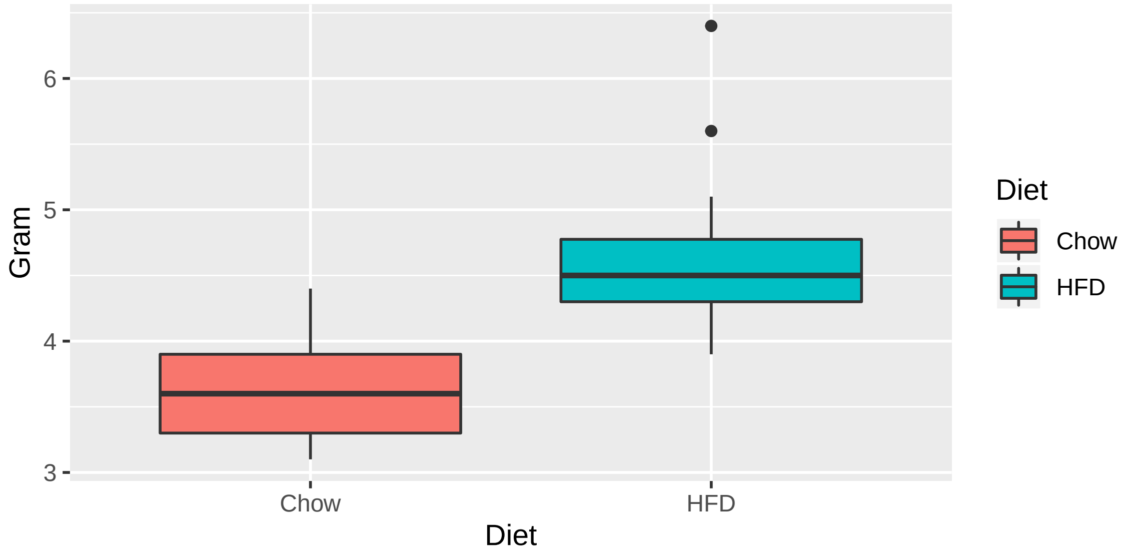
1) Plot background

```
# Plot fat mass of male
p1 <- fatMass_Male %>%
  ggplot(aes(x = Diet, y = Gram, fill = Diet))

p1
```



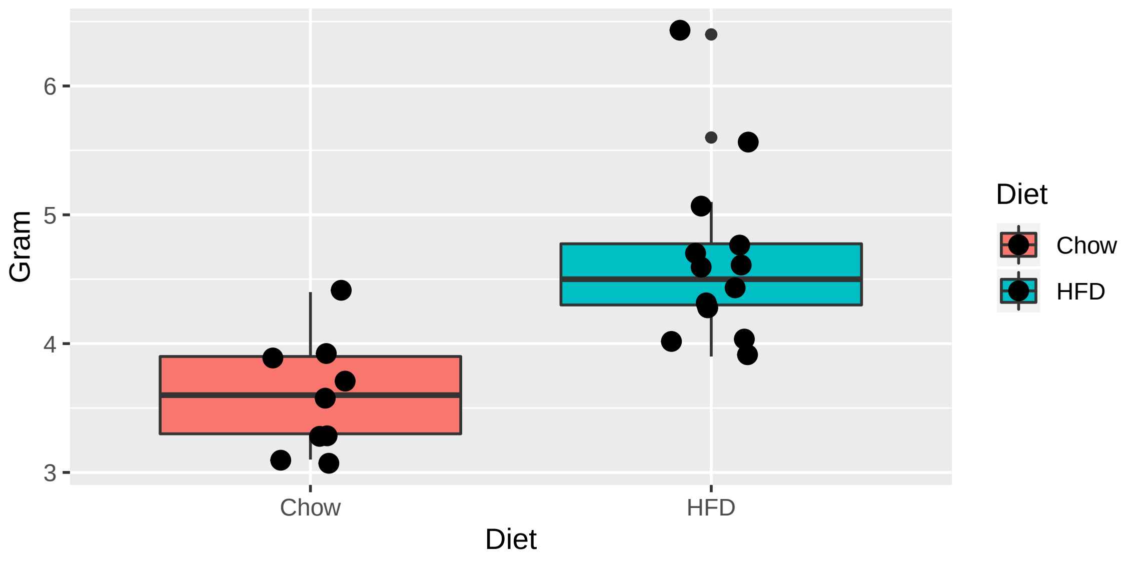
2) First layer

```
p2 <- p1 + geom_boxplot()

p2
```



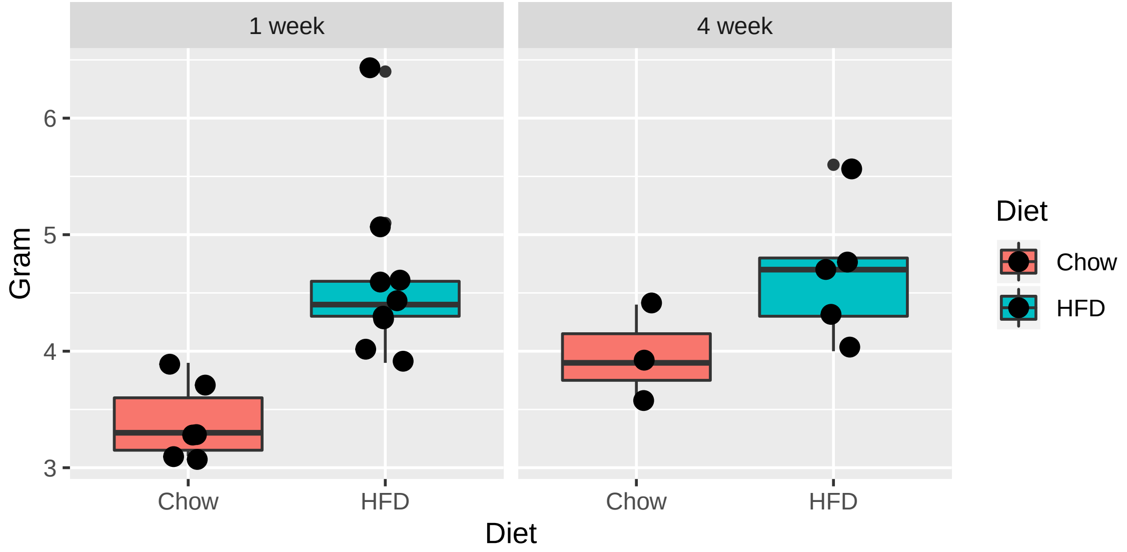
3) Another layer

```
p3 <- p2 + geom_jitter(width = .1, size = 3)

p3
```



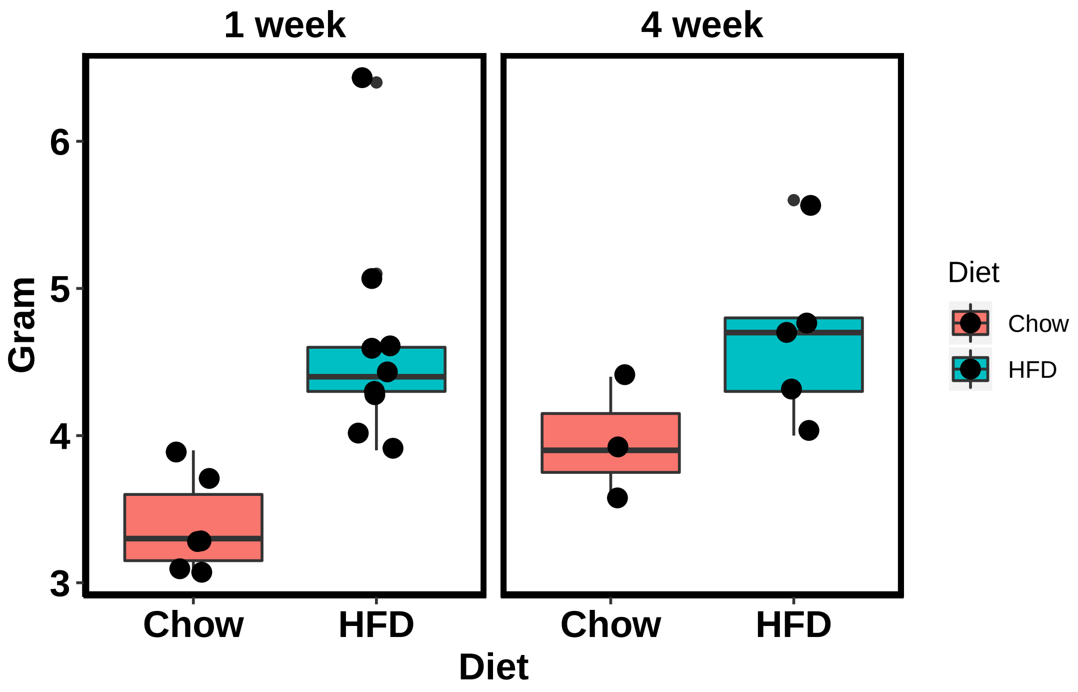
4) Lay out panels in a grid

```
p4 <- p3 + facet_grid(cols = vars(Diet_duration))

p4
```



5) Complete with a customized look

```
My_theme <- theme(
  axis.line = element_line(colour = "black"),
  axis.text.x = element_text(color = "black", size = 14, face = "bold"),
  axis.text.y = element_text(color = "black", size = 14, face = "bold"),
  axis.title.x = element_text(color = "black", size = 14, face = "bold"),
  axis.title.y = element_text(color = "black", size = 14, face = "bold"),
  strip.text.x = element_text(color = "black", size = 14, face = "bold"), # Horizontal facet labels
  strip.background = element_rect(fill = "white"), # Background of facet labels
  panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
  panel.background = element_blank(),
  panel.border = element_rect(colour = "black", fill = NA, size = 2),
  plot.margin = unit(c(0.5, 0.5, 0.5, 0.5), "cm"),
  plot.title = element_text(color = "black", size = 16, face = "bold")
)

p5 <- p4 + My_theme

p5
```



Geometric objects (geoms)

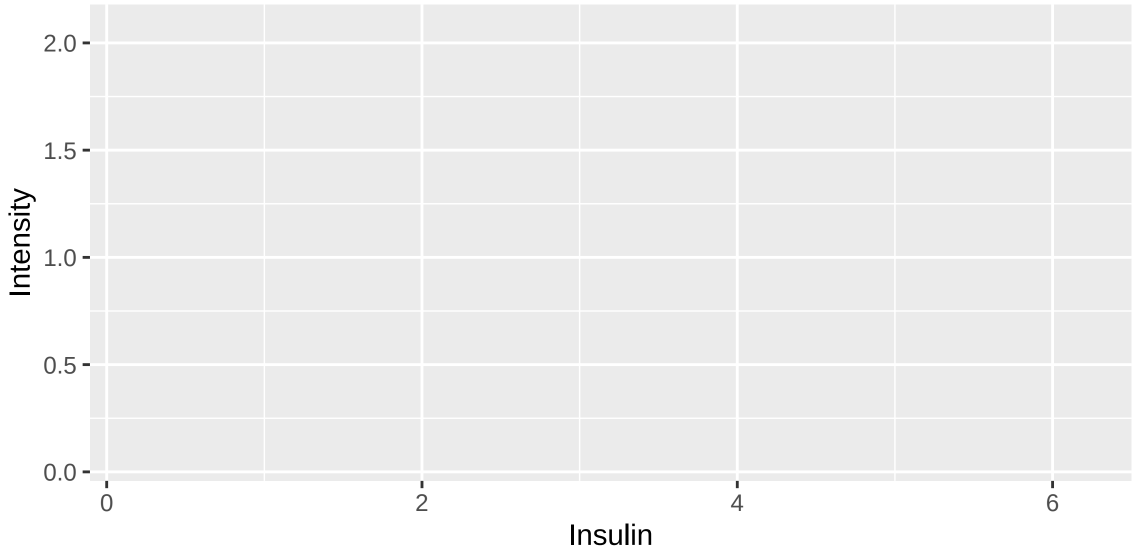
Dot plot with fitted line

Dataset: Standard curve of insulin ELISA, performed on different dates

```
##    Calibrator Insulin       Date Intensity
## 1       Cal 1   0.194 20.02.2019    0.0690
## 2       Cal 2   0.497 20.02.2019    0.0875
## 3       Cal 3   1.470 20.02.2019    0.2495
## 4       Cal 4   2.960 20.02.2019    0.5775
## 5       Cal 5   6.200 20.02.2019    1.5090
## 6       Cal 1   0.194 26.02.2019    0.0605
## 7       Cal 2   0.497 26.02.2019    0.0915
## 8       Cal 3   1.470 26.02.2019    0.2610
## 9       Cal 4   2.960 26.02.2019    0.6490
## 10      Cal 5   6.200 26.02.2019    1.7905
```

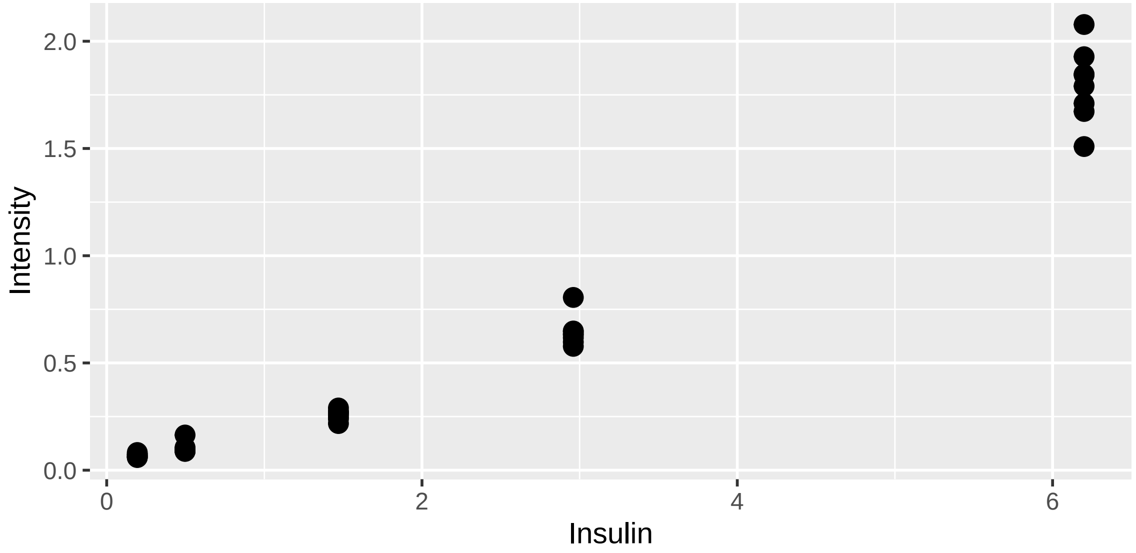
1) Plot background of standard curve

```
Ins_p1 <- Ins_Cal_v2 %>%
  ggplot(aes(x = Insulin, y = Intensity))

Ins_p1
```



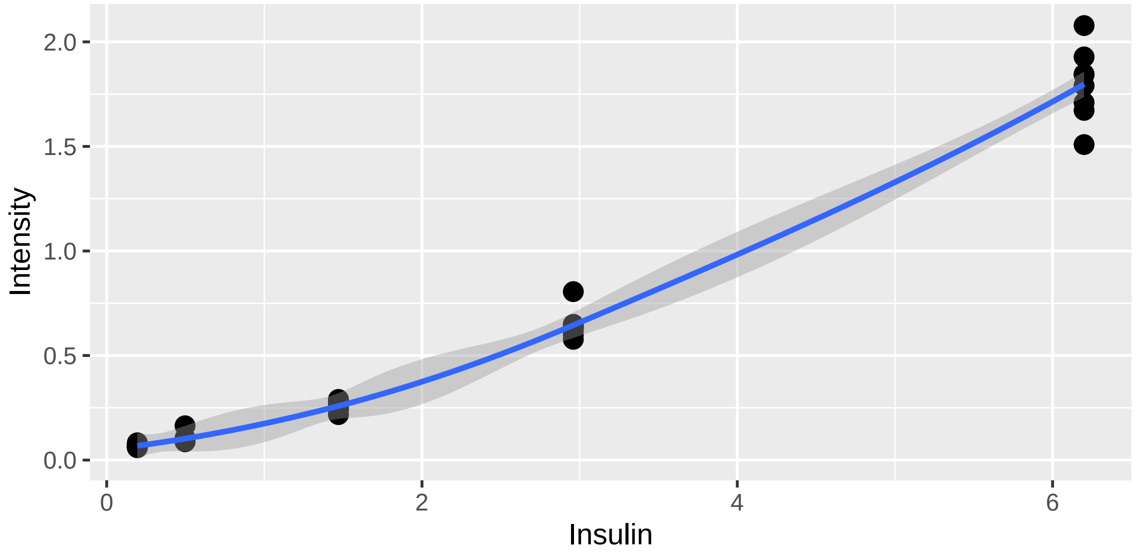
2) Add individual data points

```
Ins_p2 <- Ins_p1 + geom_point(size = 3)

Ins_p2
```



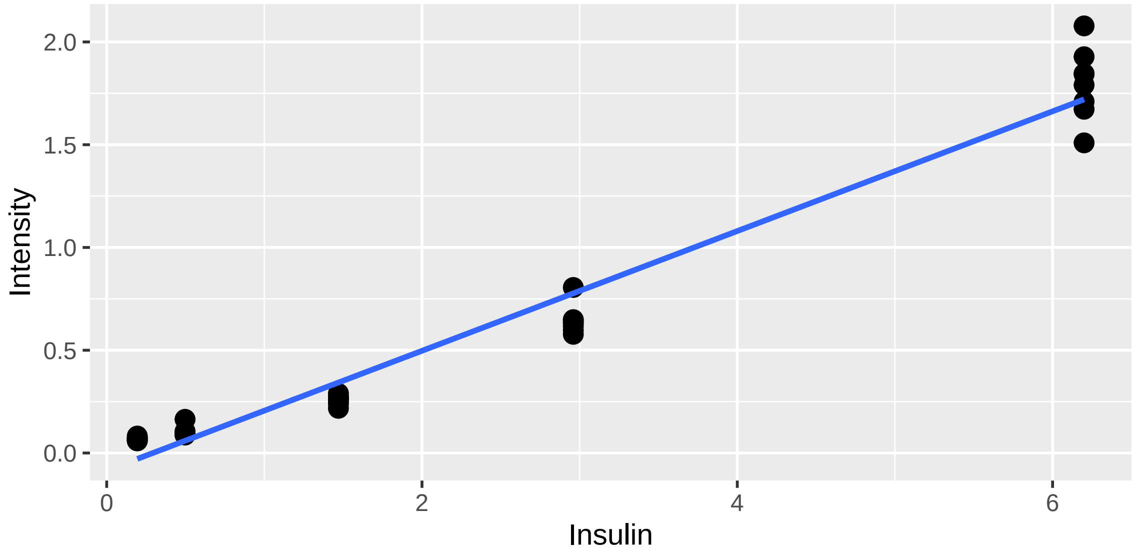
3.1) Add smoothed conditional means with standard error

```
Ins_p3 <- Ins_p2 + geom_smooth(span = 0.8, method = "loess", formula = y ~ x)

Ins_p3
```



3.2) Add a line of best fit

```
Ins_p4 <- Ins_p2 + geom_smooth(method = "lm", se = FALSE)

Ins_p4
```



```
# Linear regression
# Intensity = Intercept + (ß * Insulin)
linearMod <- lm(Intensity ~ Insulin, data = Ins_Cal_v2)
summary(linearMod)
```

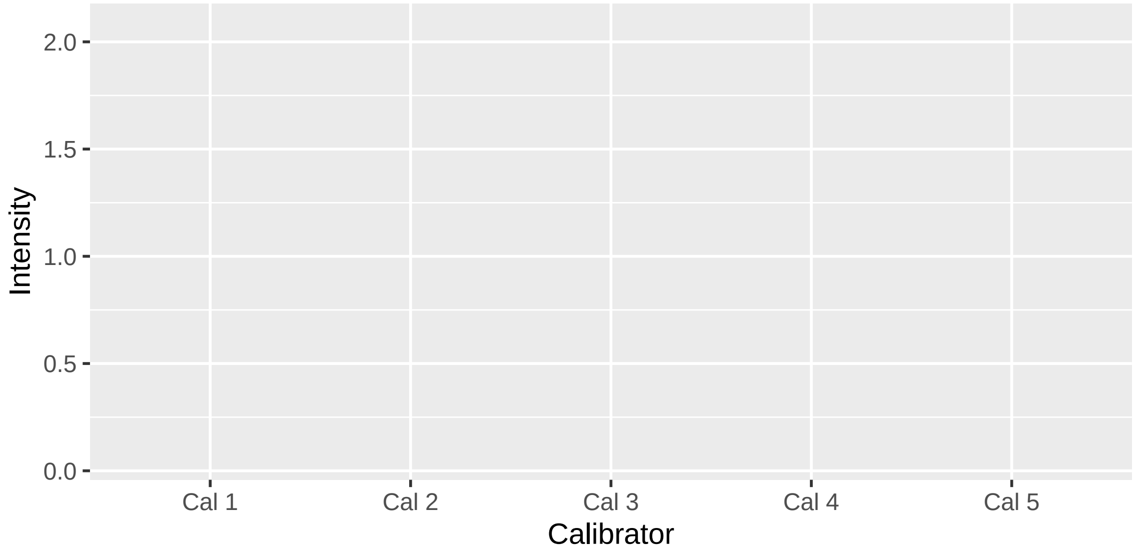
```
##
## Call:
## lm(formula = Intensity ~ Insulin, data = Ins_Cal_v2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.21166 -0.09604  0.02879  0.09088  0.35734
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.085262   0.028121  -3.032  0.00436 **
## Insulin      0.291278   0.008924  32.640  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1237 on 38 degrees of freedom
## Multiple R-squared:  0.9656, Adjusted R-squared:  0.9647
## F-statistic:  1065 on 1 and 38 DF,  p-value: < 2.2e-16
```

Bar chart

Dataset (same as above): Standard curve of insulin ELISA, performed on different dates

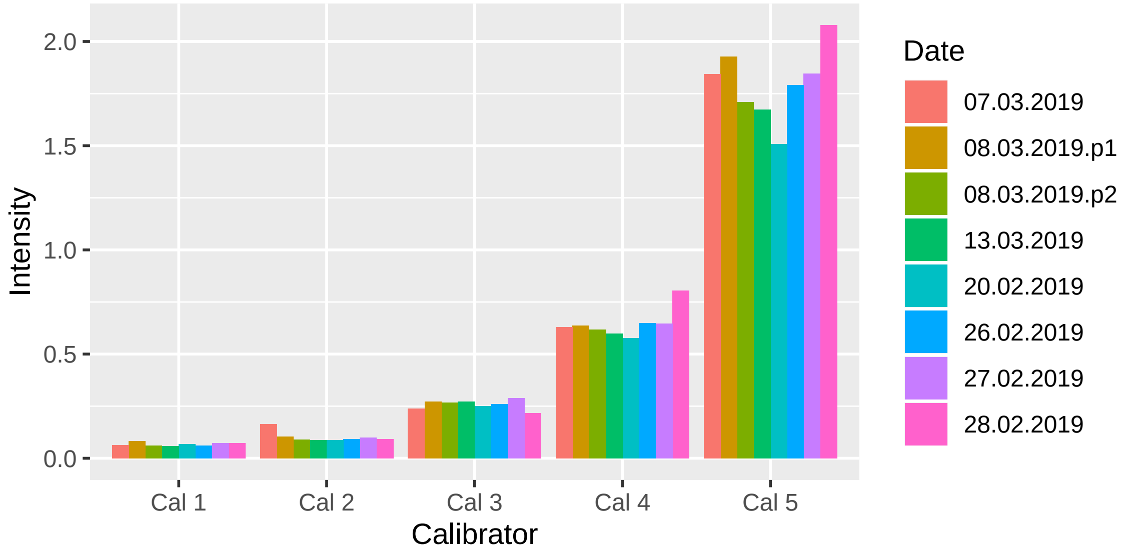
1) Plot background of bar chart

```
Ins_Bar_p1 <- Ins_Cal_v2 %>%
  ggplot(aes(x = Calibrator, y = Intensity, fill = Date))

Ins_Bar_p1
```



2) Add bars in groups

```
Ins_Bar_p2 <- Ins_Bar_p1 + geom_bar(stat = "identity", position = position_dodge())

Ins_Bar_p2
```

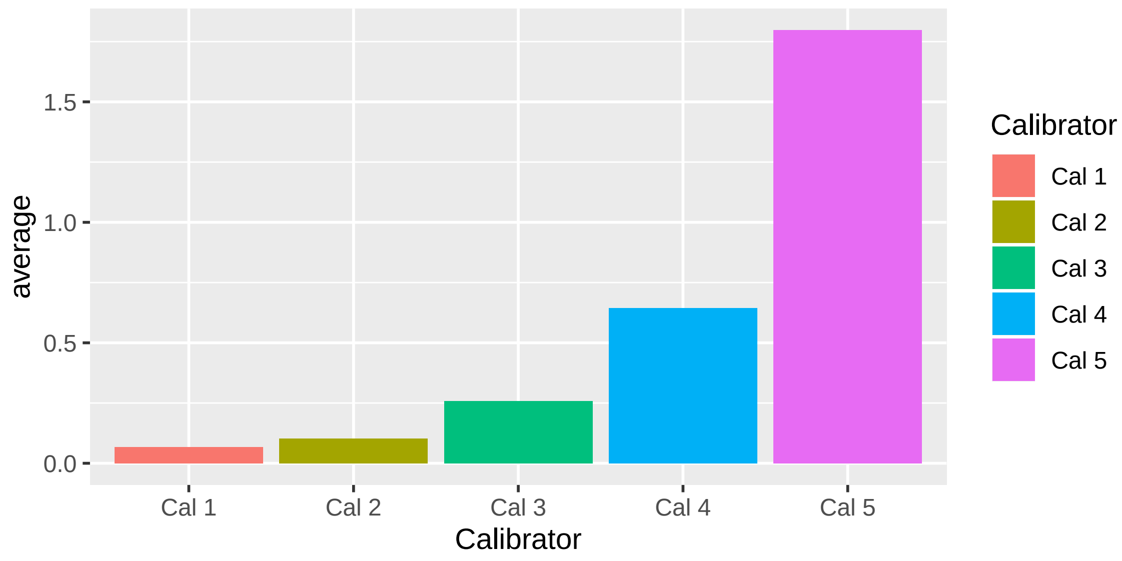


3) Add bars with means with standard error

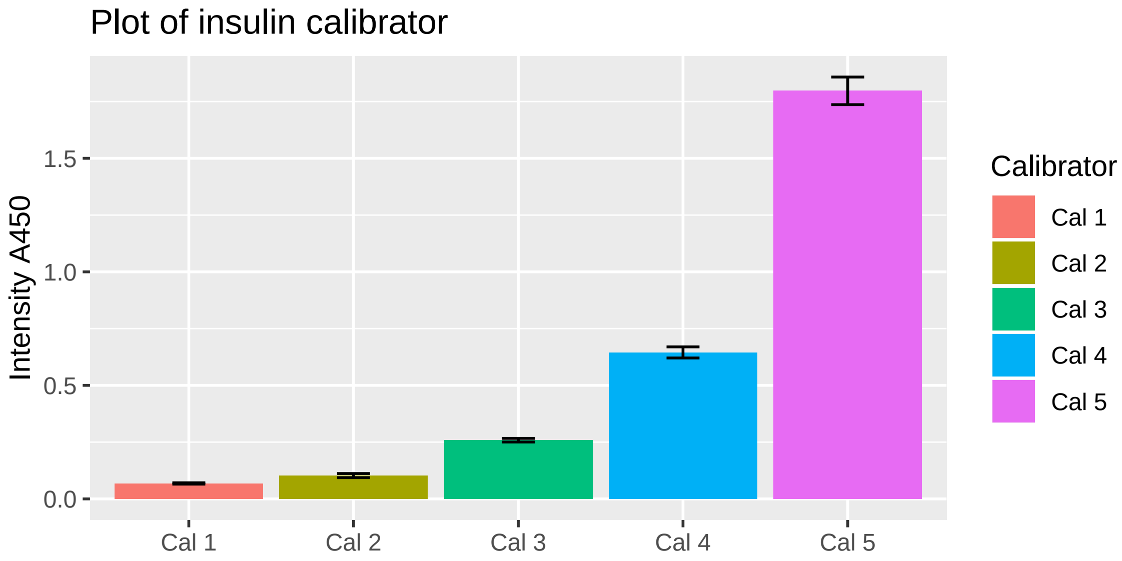
```
# Calculate mean and se
Ins_Stat <- Ins_Cal_v2 %>%
  group_by(Insulin, Calibrator) %>%
  summarise(
    average = mean(Intensity, na.rm = TRUE),
    se = sd(Intensity, na.rm = TRUE) / sqrt(length(Intensity))
  )

Ins_Stat
```

```
## # A tibble: 5 x 4
## # Groups:   Insulin [5]
##   Insulin Calibrator average      se
##     <dbl> <fct>        <dbl>   <dbl>
## 1   0.194 Cal 1        0.068 0.00291
## 2   0.497 Cal 2        0.102 0.00905
## 3   1.47  Cal 3        0.258 0.00800
## 4   2.96  Cal 4        0.645 0.0245
## 5   6.2   Cal 5        1.80  0.0608
```

```
# First plot the mean
Ins_Bar_p3 <- Ins_Stat %>%
  ggplot(aes(x = Calibrator, y = average, fill = Calibrator)) +
  geom_bar(stat = "identity")

Ins_Bar_p3
```



```
# Then add error bars
Ins_Bar_p4 <- Ins_Bar_p3 +
  geom_errorbar(aes(ymin = average - se, ymax = average + se), width = .2) +
  labs(title = "Plot of insulin calibrator", x = NULL, y = "Intensity A450")

Ins_Bar_p4
```



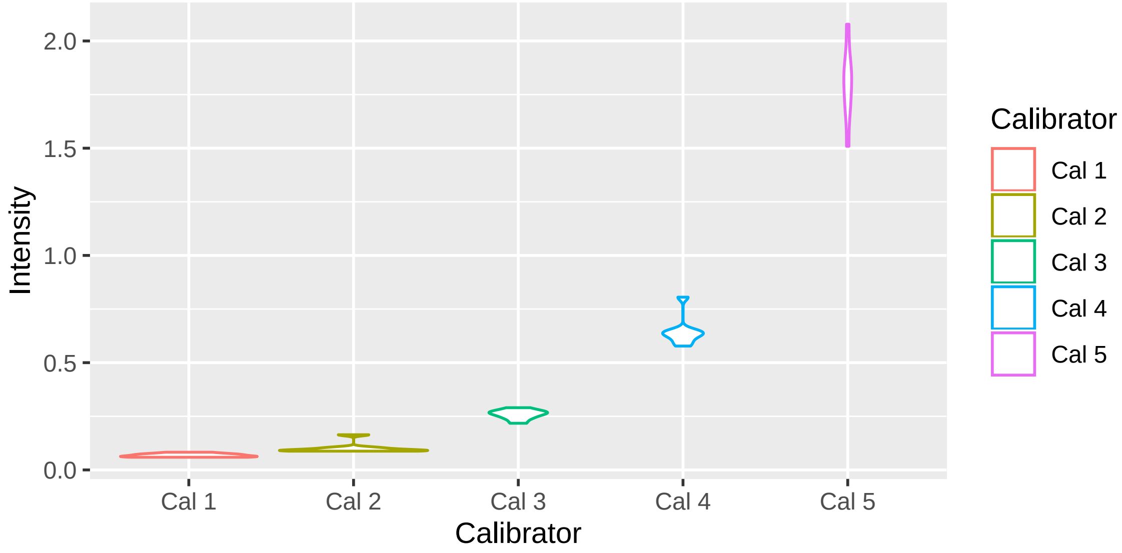
Violin plot

Dataset (same as above): Standard curve of insulin ELISA, performed on different dates

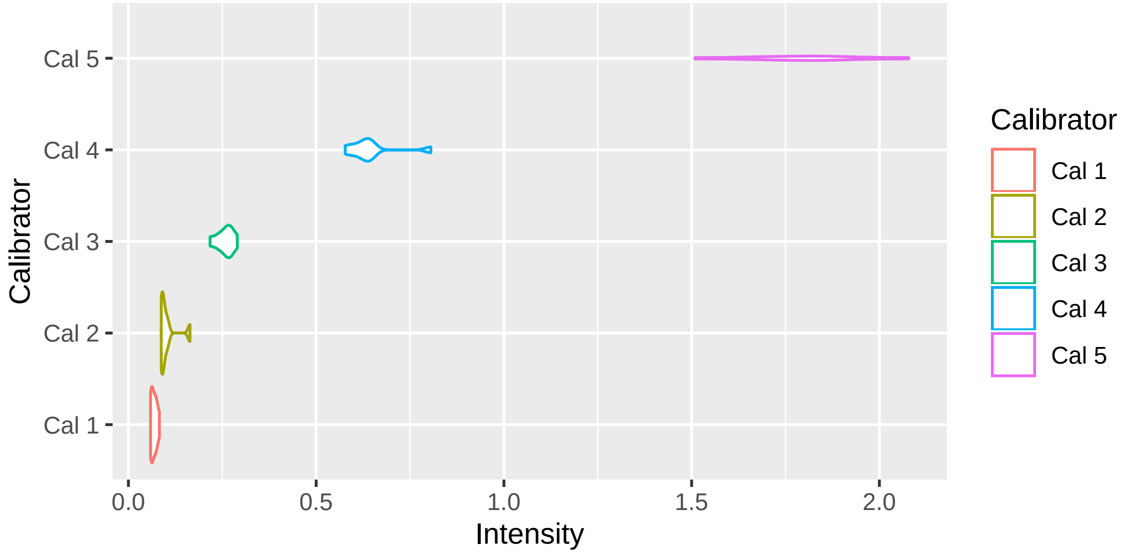
1) Basic violin plot

```
Ins_Violin_p1 <- Ins_Cal_v2 %>%
  ggplot(aes(x = Calibrator, y = Intensity, color = Calibrator)) +
  geom_violin()
```

```
Ins_Violin_p1
```

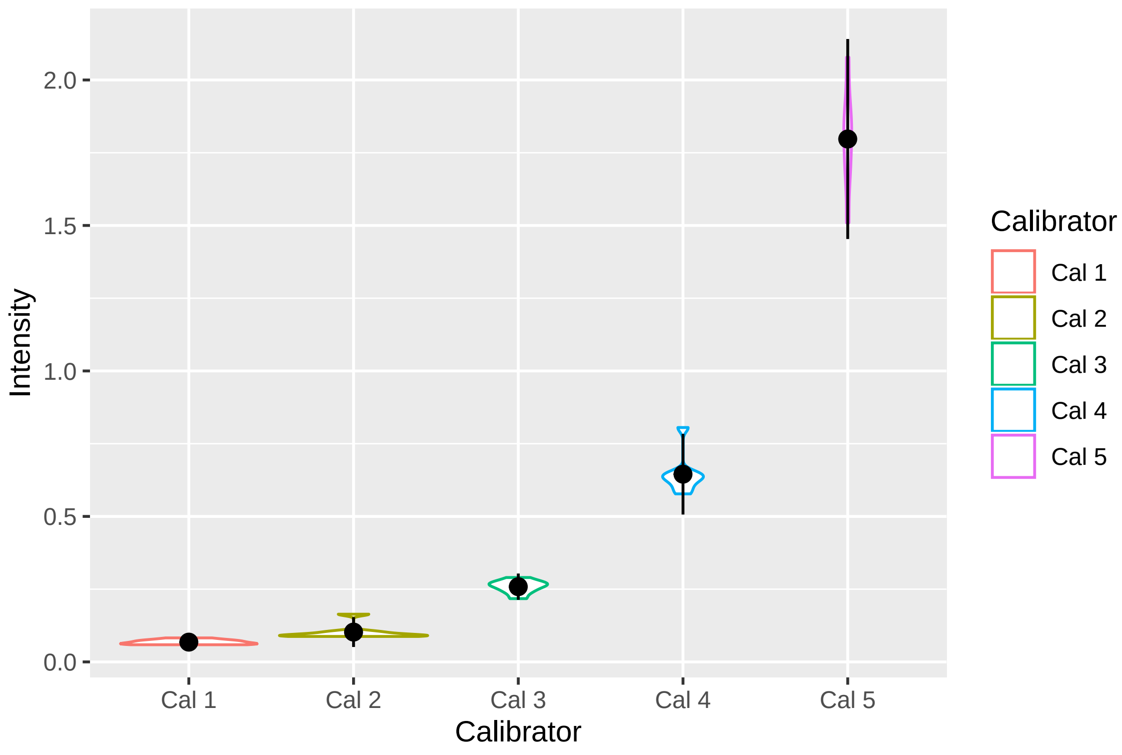


```
# Rotate the violin plot
Ins_Violin_p1 + coord_flip()
```



2) Add mean with standard deviation

```
Ins_Violin_p1 + stat_summary(fun.data = mean_sdl, geom = "pointrange", color = "black")
```



Venn diagram

Dataset for venn diagram

```
##           A    B     C
## Gene_1 TRUE TRUE  TRUE
## Gene_2 TRUE TRUE FALSE
## Gene_3 TRUE TRUE FALSE
## Gene_4 TRUE TRUE FALSE
## Gene_5 TRUE TRUE  TRUE
## Gene_6 TRUE TRUE FALSE
```

1) Use vennCounts from the package limma to compute classification counts

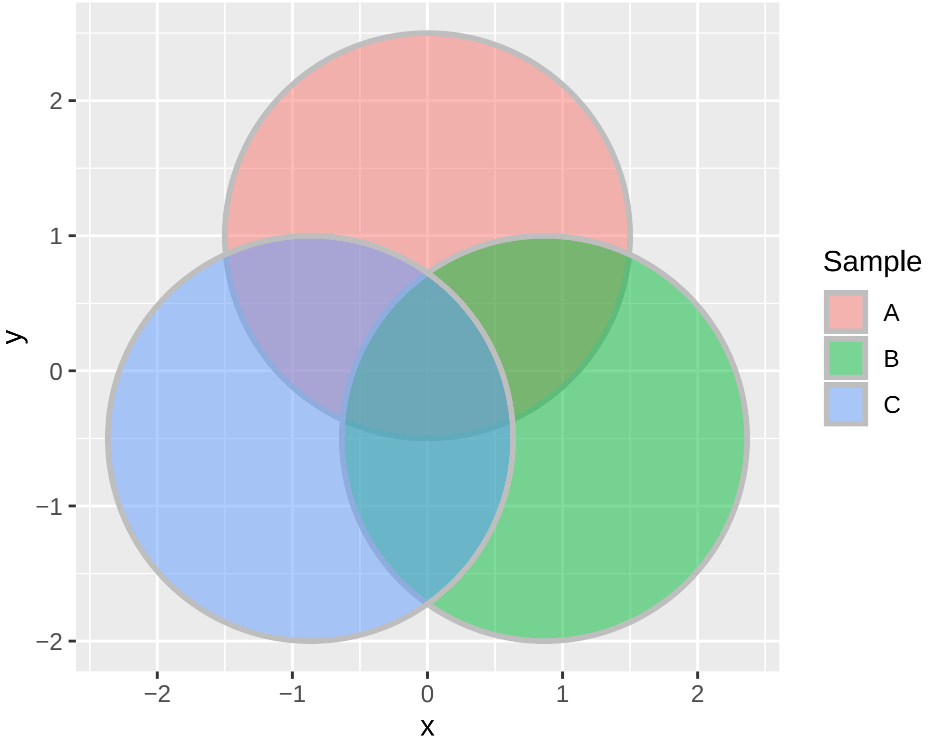
```
# Load packages
library(ggforce)
library(limma)

# Compute classification counts
counts_venn <- vennCounts(df_venn)
class(counts_venn) <- "matrix"

# Add x and y coordinates for the count annotations
counts_venn <- as.data.frame(counts_venn)[-1, ] %>%
  mutate(
    x = c(0, 1.2, 0.8, -1.2, -0.8, 0, 0),
    y = c(1.2, -0.6, 0.5, -0.6, 0.5, -1, 0)
  )
```

```
counts_venn
```

```
##   A B C Counts    x    y
## 1 0 0 1      6  0.0  1.2
## 2 0 1 0      4  1.2 -0.6
## 3 0 1 1      5  0.8  0.5
## 4 1 0 0     12 -1.2 -0.6
## 5 1 0 1     11 -0.8  0.5
## 6 1 1 0     27  0.0 -1.0
## 7 1 1 1     32  0.0  0.0
```

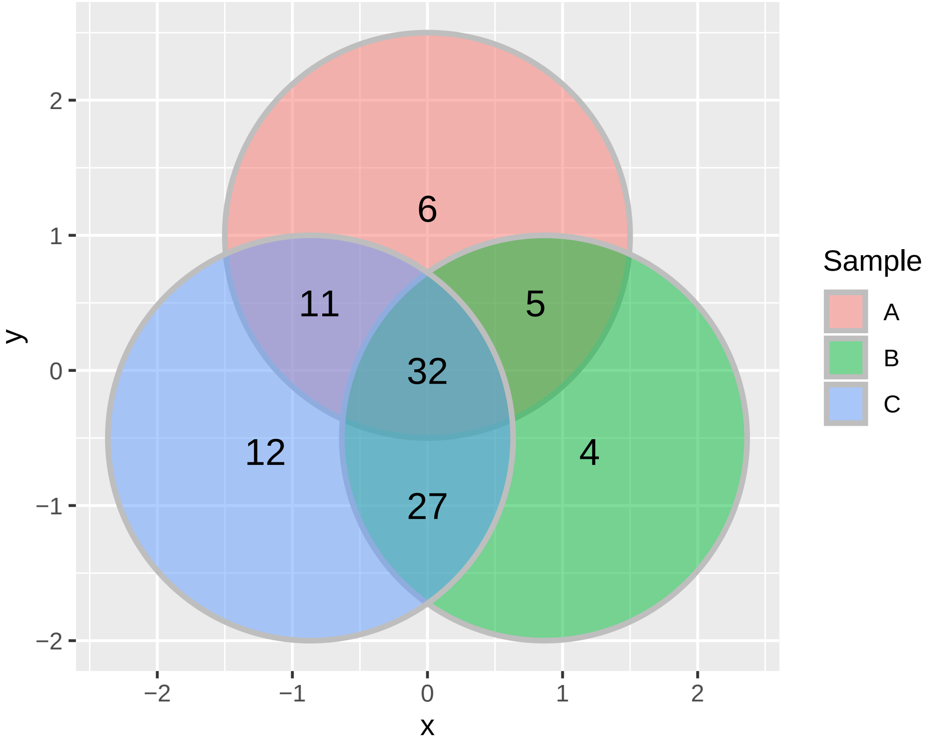
2) Define basic structure for the circles

```
# Define x and y coordinates for the circles
venn_structure <- data.frame(
  x = c(0, 0.866, -0.866),
  y = c(1, -0.5, -0.5),
  Sample = c("A", "B", "C")
)

venn_p1 <- venn_structure %>% ggplot(aes(x0 = x, y0 = y, r = 1.5, fill = Sample)) +
  geom_circle(alpha = .5, size = 1, colour = "grey") +
  coord_fixed()

venn_p1
```



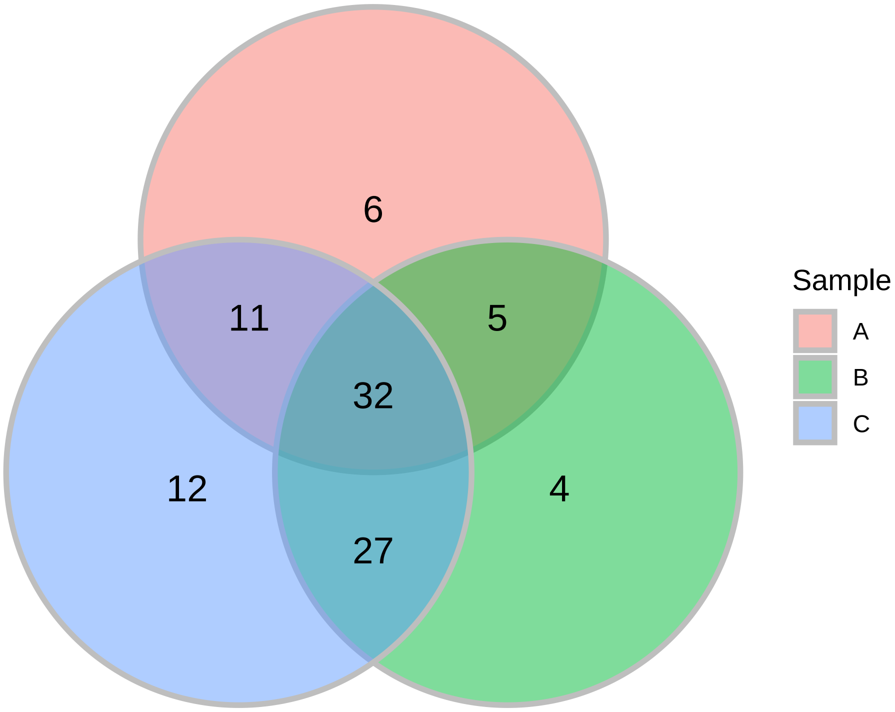
3) Venn diagram with annotation of the counts

```
venn_p2 <- venn_p1 +
  annotate("text", x = counts_venn$x, y = counts_venn$y, label = counts_venn$Counts, size = 5)

venn_p2
```



4) Finally to remove the grey background

```
venn_p2 + theme_void()
```



PCA plot

Dataset from RNA-seq tpm counts

```r
# Load kallisto counts
kallisto_count <- read.csv("~/mRNA_IP/count/kallisto_count_19112019.csv")

kallisto_df <- kallisto_count[c(1, 5, 6, 7)] %>%
  spread(target_id, tpm, fill = NA, convert = FALSE, drop = TRUE, sep = NULL)

kallisto_df[1:9, 1:6]
```

```
##                        sample   condition ENSMUST00000000001.4
## 1       Ins1creTRAP input 1       Input             2.827743
## 2       Ins1creTRAP input 2       Input             3.489938
## 3          Ins1creTRAP IP 1          IP            13.005613
## 4          Ins1creTRAP IP 2          IP             7.870618
## 5 Ins1creTRAP supernatant 1 Supernatant             3.790337
## 6 Ins1creTRAP supernatant 2 Supernatant             2.254487
## 7                TRAP input       Input             2.103856
## 8                   TRAP IP          IP             8.197811
## 9          TRAP supernatant Supernatant             1.719226
##   ENSMUST00000000003.13 ENSMUST00000000010.8 ENSMUST00000000028.13
## 1                     0                    0            0.00000000
## 2                     0                    0            0.00000000
## 3                     0                    0            1.14064983
## 4                     0                    0            0.00000000
## 5                     0                    0            0.00000000
## 6                     0                    0            0.00000000
## 7                     0                    0            0.07130594
## 8                     0                    0            0.00000000
## 9                     0                    0            0.00000000
```

```
# Calculate principal component analysis based on tpm
pca <- prcomp(kallisto_df[, -c(1, 2)])

summary(pca)[6]
```

```
## $importance
##                                 PC1         PC2         PC3         PC4
## Standard deviation     49291.05206 32069.63898 15287.04431 10993.04070
## Proportion of Variance     0.62116     0.26294     0.05975     0.03090
## Cumulative Proportion      0.62116     0.88410     0.94384     0.97474
##                               PC5        PC6        PC7        PC8
## Standard deviation     6782.76140 5623.08604 4228.81560 1814.02052
## Proportion of Variance    0.01176    0.00808    0.00457    0.00084
## Cumulative Proportion     0.98650    0.99459    0.99916    1.00000
##                                PC9
## Standard deviation     1.77849e-10
## Proportion of Variance 0.00000e+00
## Cumulative Proportion  1.00000e+00
```

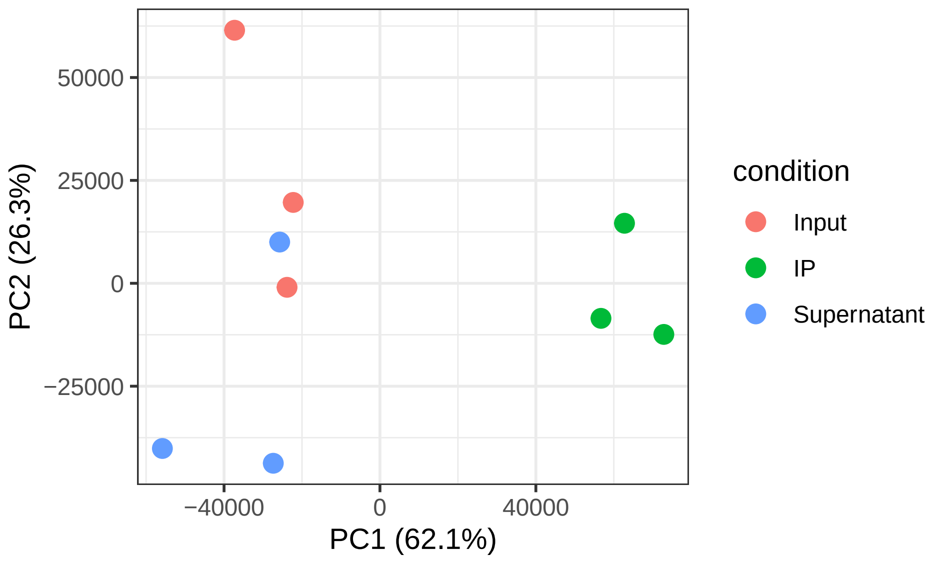
PCA plot

Option 1: By using ggplot2

```
# Create data frame for PC
df_pca <- as.data.frame(pca$x)

df_pca$condition <- kallisto_df$condition

head(df_pca)
```

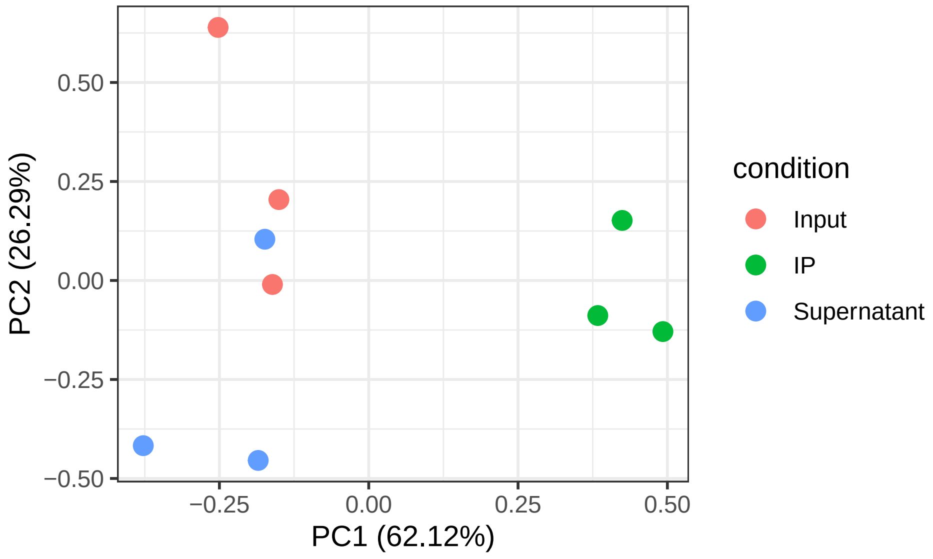
```
##          PC1        PC2        PC3         PC4         PC5        PC6
## 1 -37293.90  61478.383   1827.194   2199.2406   7091.6674  -676.2481
## 2 -22245.09  19649.611  12676.412   -513.6848 -13776.1041 -1781.4077
## 3  62742.51  14587.313 -21694.384 -15161.7450  -4035.3473  5238.8259
## 4  56707.74  -8511.307  11915.603  20109.9504   -965.5041  6760.0376
## 5 -25722.87  10023.528 -22962.362  11406.5635   2351.6401 -2047.2995
## 6 -55818.90 -40126.540  -8128.496   1312.4749  -5314.0585 -3507.3695
##          PC7         PC8           PC9   condition
## 1  3775.757 -1725.25237 -1.988686e-10       Input
## 2 -4706.968 -1114.96439  3.476534e-10       Input
## 3  1603.261   -82.63524  6.650248e-11          IP
## 4  2216.807   273.58882  5.147383e-10          IP
## 5 -5165.664  2339.50146 -1.255307e-10 Supernatant
## 6  7266.615   185.97976 -1.526460e-10 Supernatant
```

```
# Plot PCA with ggplot2
pca_p1 <- ggplot(df_pca, aes(x = PC1, y = PC2, color = condition))

pca_p1 + geom_point(size = 3) + theme_bw() + labs(x = "PC1 (62.1%)", y = "PC2 (26.3%)")
```



Option 2: By using package ggfortify

```
library(ggfortify)
pca_v2_p1 <- autoplot(pca, data = kallisto_df, colour = "condition", size = 3) +
  theme_bw()

pca_v2_p1
```



Heatmap

Dataset from RNA-seq tpm counts

```
df_tpm <- kallisto_count %>%
  filter(tpm > 0) %>%
  select(c(1, 5, 6))

head(df_tpm)
```

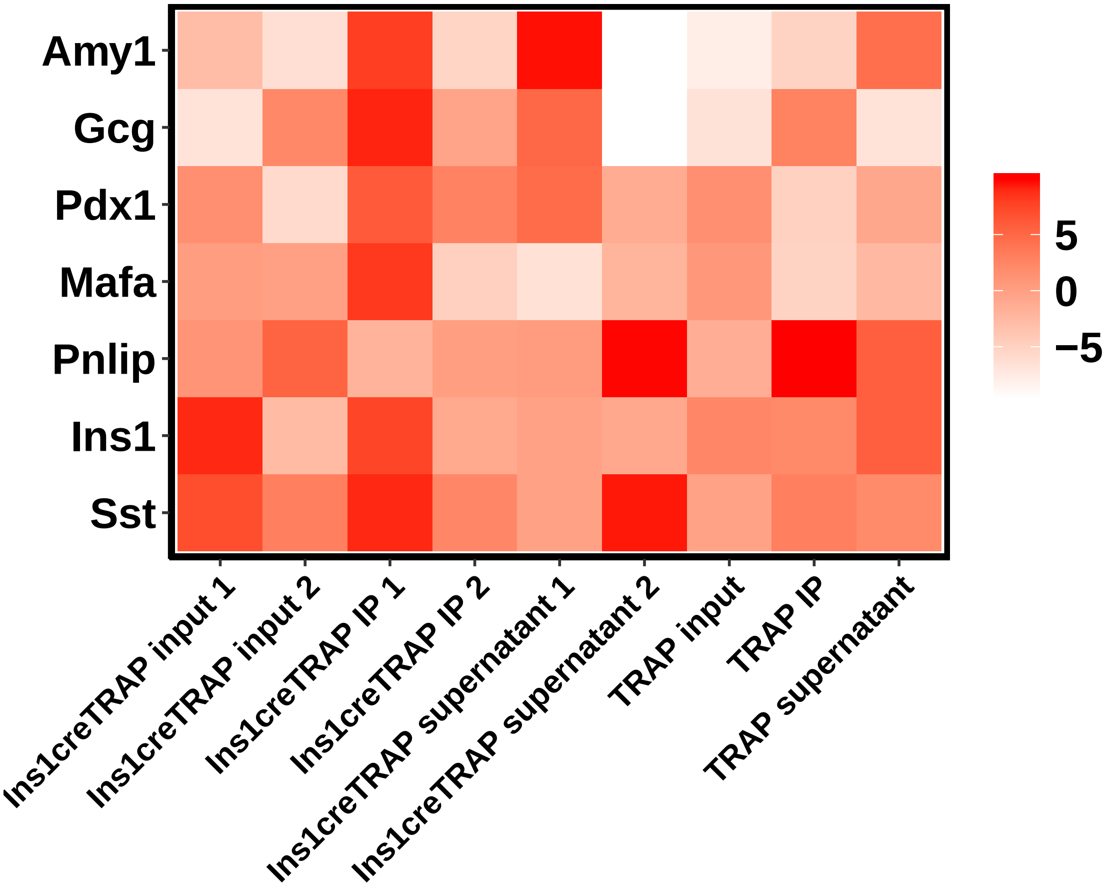
```
##              target_id       tpm                    sample
## 1 ENSMUST00000000001.4  2.827743       Ins1creTRAP input 1
## 2 ENSMUST00000000001.4  3.489938       Ins1creTRAP input 2
## 3 ENSMUST00000000001.4 13.005613          Ins1creTRAP IP 1
## 4 ENSMUST00000000001.4  7.870618          Ins1creTRAP IP 2
## 5 ENSMUST00000000001.4  3.790337 Ins1creTRAP supernatant 1
## 6 ENSMUST00000000001.4  2.254487 Ins1creTRAP supernatant 2
```

Subset a list of interested genes

```r
# Ins1: ENSMUST00000039652.5
# Mafa: ENSMUST00000062002.5
# Pdx1: ENSMUST00000085591.6
# Gcg: ENSMUST00001102733.9
# Sst: ENSMUST00000004480.4
# Amy1: ENSMUST00000106540.7
# Pnlip: ENSMUST00000057270.8

geneList <- c("ENSMUST00000039652.5", "ENSMUST00000062002.5", "ENSMUST00000085591.6", "ENSMUST0000010273

# Subset interested genes
df_subset <- subset(df_tpm, df_tpm$target_id %in% geneList)

# Calculate z scores
df_subset$z <- runif(df_subset$tpm, min = -10, max = 10)

head(df_subset)
```

```
##                 target_id       tpm                    sample          z
## 3791 ENSMUST00000004480.4 17.137394       Ins1creTRAP input 1  6.9918464
## 3792 ENSMUST00000004480.4 17.707016       Ins1creTRAP input 2  3.0645192
## 3793 ENSMUST00000004480.4 20.647121          Ins1creTRAP IP 1  9.0177151
## 3794 ENSMUST00000004480.4 17.401344          Ins1creTRAP IP 2  2.3444610
## 3795 ENSMUST00000004480.4  9.881168 Ins1creTRAP supernatant 1 -0.1431873
## 3796 ENSMUST00000004480.4  7.103728 Ins1creTRAP supernatant 2  9.5201313
```

```r
# Set the theme for heatmap
theme_heatmap <- theme(
  axis.line = element_line(colour = "black"),
  axis.text.x = element_text(color = "black", size = 12, face = "bold", angle = 45, hjust = 1),
  axis.text.y = element_text(color = "black", size = 16, face = "bold"),
  axis.title.x = element_blank(),
  axis.title.y = element_blank(),
  legend.title = element_blank(),
  legend.text = element_text(color = "black", size = 16, face = "bold"),
  legend.key = element_rect(fill = "white"), # Remove grey background of the legend
  panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
  panel.background = element_blank(),
  panel.border = element_rect(colour = "black", fill = NA, size = 2),
  plot.margin = unit(c(0.5, 0.5, 0.5, 0.5), "cm")
)

# heatmap by geom_tile
df_subset %>% ggplot(aes(x = sample, y = target_id)) +
  geom_tile(aes(fill = z)) +
  scale_fill_gradient(low = "white", high = "red") +
  theme_heatmap +
  scale_y_discrete(labels = c("ENSMUST00000039652.5" = "Ins1", "ENSMUST00000062002.5" = "Mafa", "ENSMUS
```



References

ggplot2: Elegant Graphics for Data Analysis by Hadley Wickham https://ggplot2-book.org/

ggplot2 - Essentials http://www.sthda.com/english/wiki/ggplot2-essentials