Lubrizol Exploratory Analysis

k.molloy@lancaster.ac.uk

```
library(tidyverse)
library(dplyr)
library(ggplot2)
```

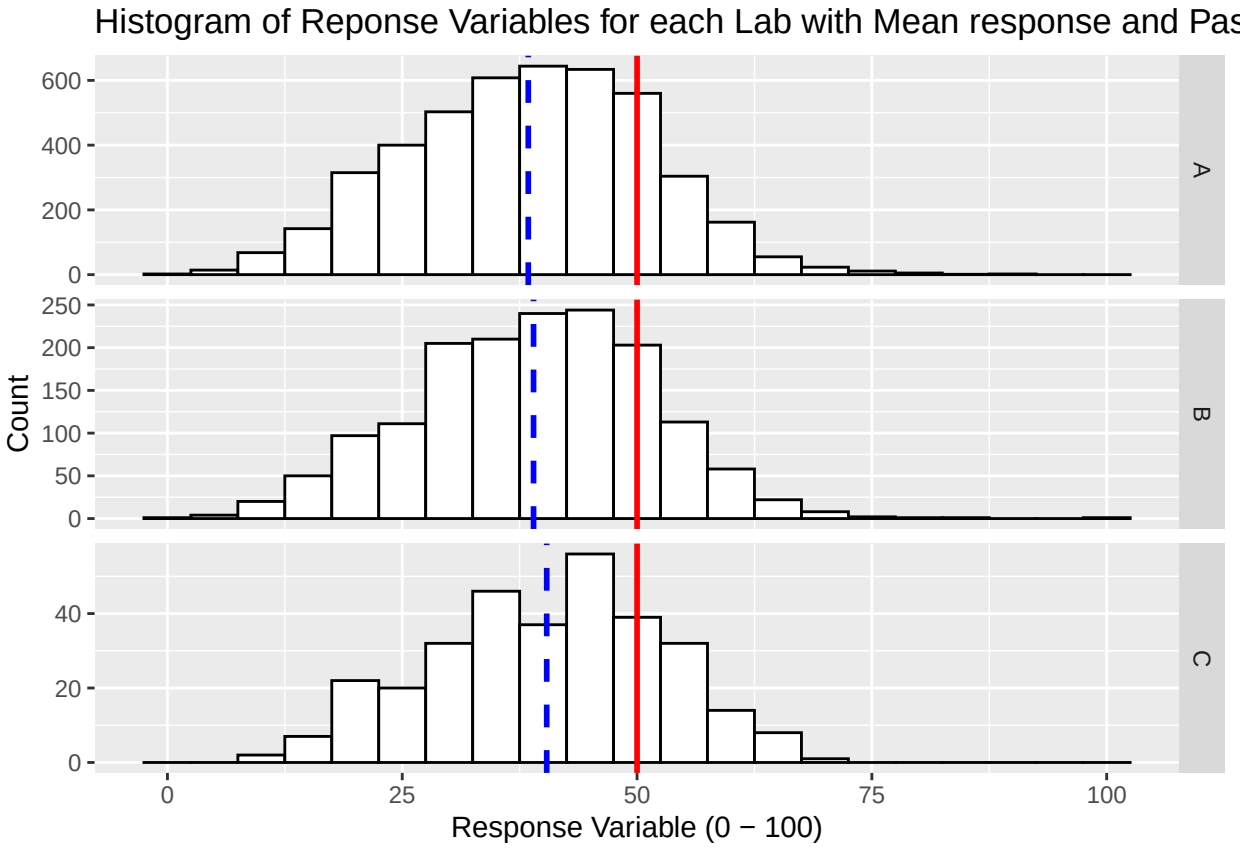
Read in data

```
# Read CSV
df = read.csv("../data/Data.csv", stringsAsFactors=FALSE)
# Factorise LAB
df$LAB = as.factor(df$LAB)
# Replace NAN's with 0's
df[is.na(df)] = 0
```

```
# Calculate Mean of each LAB
by_lab = df %>%
  group_by(LAB) %>%
  summarise(mean_val = mean(Response))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

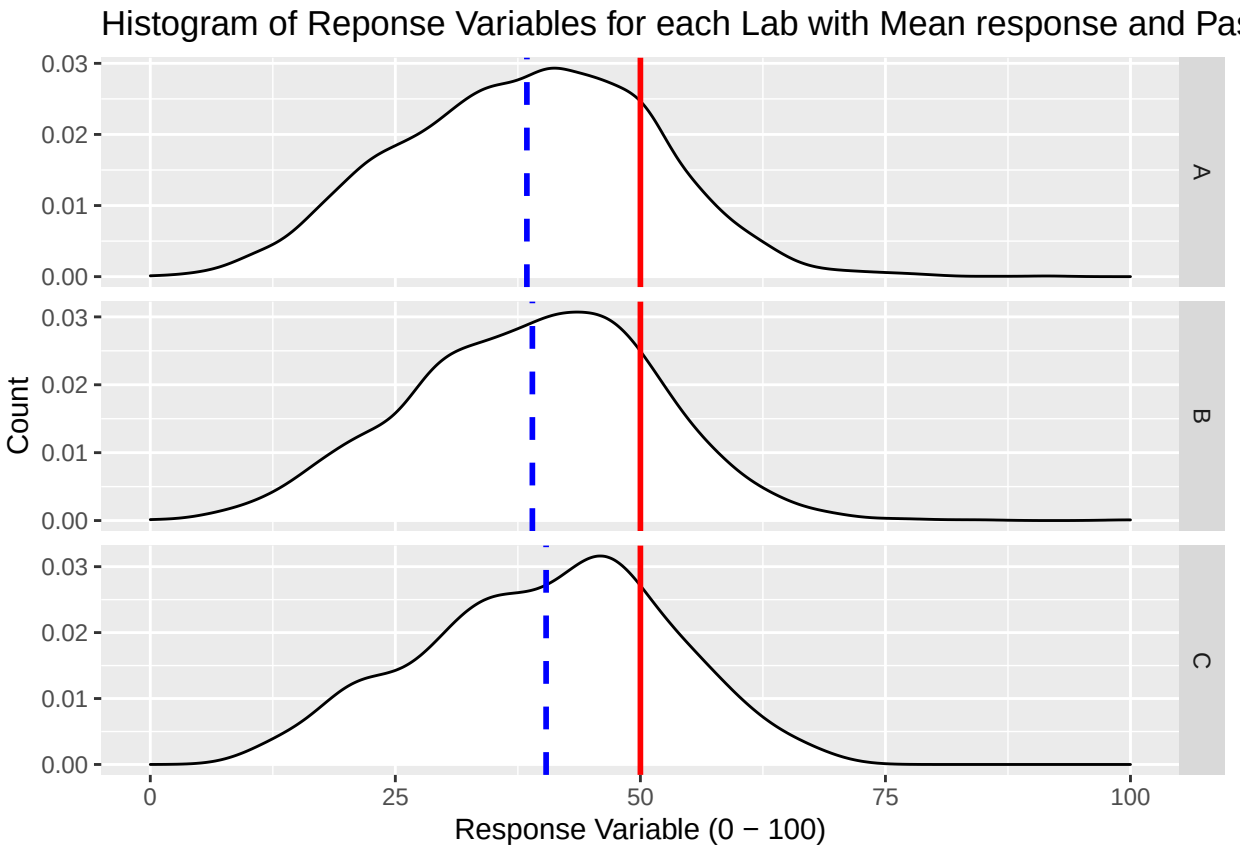
```
# Plot
p = ggplot(df, aes(x=Response)) +
  # Histogram
  geom_histogram(color="black", fill="white", binwidth = 5) +
  # Split into Facets
  facet_grid(LAB ~ ., scales="free_y") +
  # Adding Mean Reponse
  geom_vline(data = by_lab, aes(xintercept=mean_val),
             color="blue", linetype="dashed", size=1) +
  # Adding Pass Mark
  geom_vline(xintercept = 50, color="red", size=1) +
  # Labels
  ylab("Count") +
  xlab("Response Variable (0 - 100)") +
  ggtitle("Histogram of Reponse Variables for each Lab with Mean response and Pass Mark")


p
```



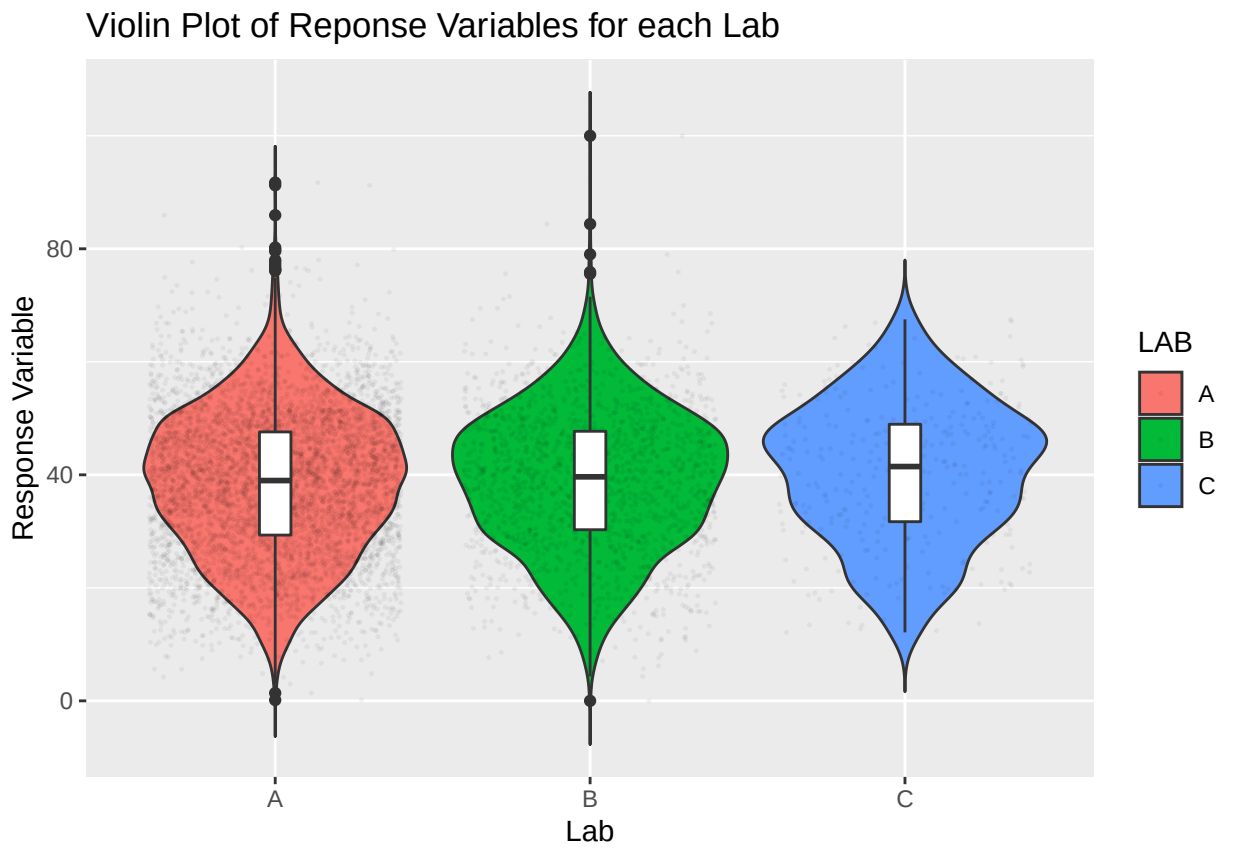

```
# Plot
p = ggplot(df, aes(x=Response)) +
  # Histogram
  geom_density(color="black", fill="white", binwidth = 5) +
  # Split into Facets
  facet_grid(LAB ~ ., scales="free_y") +
  # Adding Mean Reponse
  geom_vline(data = by_lab, aes(xintercept=mean_val),
             color="blue", linetype="dashed", size=1) +
  # Adding Pass Mark
  geom_vline(xintercept = 50, color="red", size=1) +
  # Labels
  ylab("Count") +
  xlab("Response Variable (0 - 100)") +
  ggtitle("Histogram of Reponse Variables for each Lab with Mean response and Pass Mark")
```

```
## Warning: Ignoring unknown parameters: binwidth
```

```
p
```



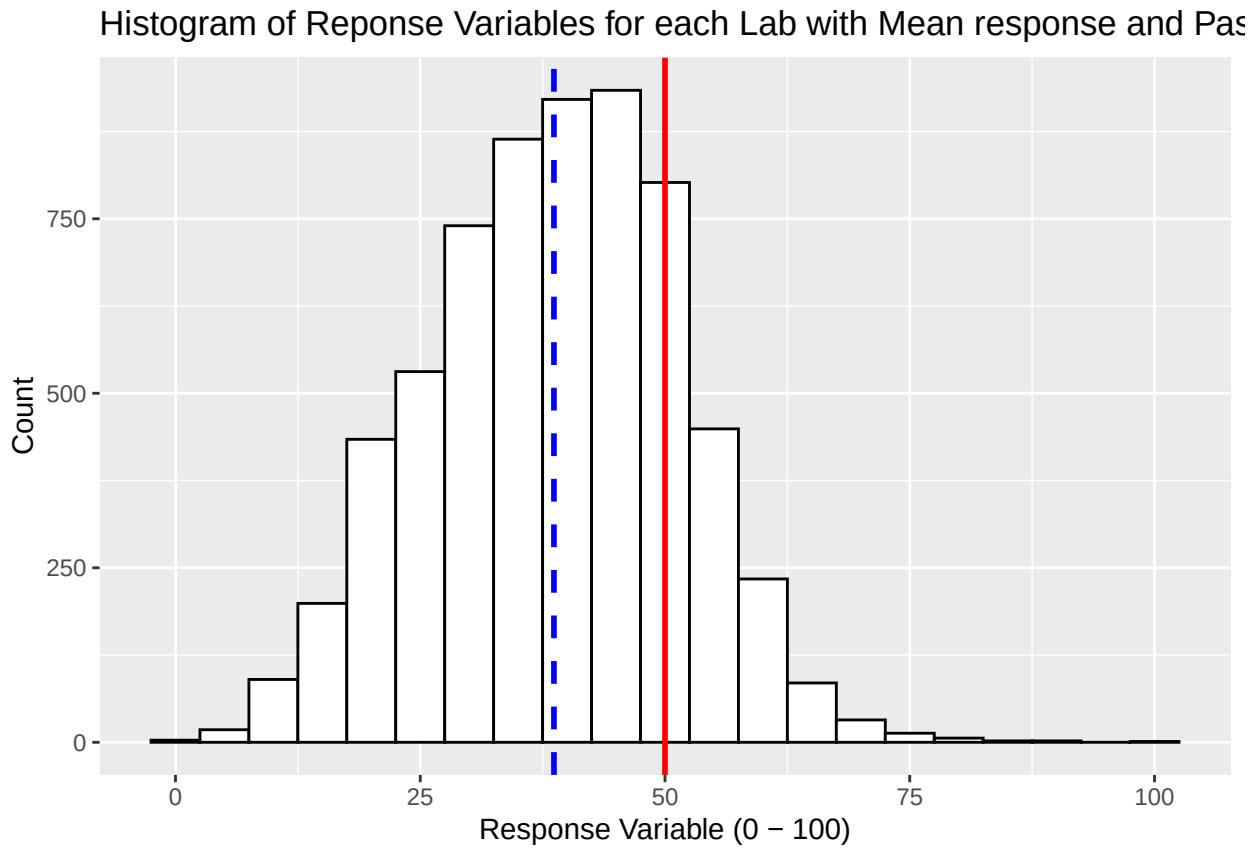

```
ggplot(df, aes(x = LAB, y = Response, fill=LAB)) +
  geom_violin(trim=FALSE) +
  scale_x_discrete(name= "Lab", labels=c("A", "B", "C")) +
  geom_jitter(color="black", size=0.2, alpha=0.05) +
  geom_boxplot(width=0.1, fill="white") +
  labs(title="Violin Plot of Reponse Variables for each Lab",x="Lab", y = "Response Variable")
```



Geyi Analysis

```
# Make copy
data_model <- df
# Drop Identifier and LAB
data_model$LAB <- NULL
# Remove Ingredients
data_model <- data_model[1:nrow(data_model),477:ncol(data_model)]
# Remove 0 majority columns, because they're not statistically significant in linear analysis
df_groups <- data_model[which(colMeans(data_model) > 0.02,)]
```

```
ggplot(df_groups, aes(x=Response)) +
  # Histogram
  geom_histogram(color="black", fill="white", binwidth = 5) +
  # Adding Mean Reponse
  geom_vline(aes(xintercept=mean(Response)),
             color="blue", linetype="dashed", size=1) +
  # Adding Pass Mark
  geom_vline(xintercept = 50, color="red", size=1) +
  # Labels
  ylab("Count") +
  xlab("Response Variable (0 - 100)") +
  ggtitle("Histogram of Reponse Variables for each Lab with Mean response and Pass Mark")
```



Rudimentary GLM

```
# Create GLM with response vs all groups
model <- glm(Response~.,data = df_groups)
# Extract p values
pvals = coef(summary(model))[,4]
```

```
# Get group names where p < 0.025
high_sig = names(pvals)[pvals < 0.025][-1]
```

Plot Regression of Highly Significant Plots

```
# Using Pivot Longer merge all groups into 2 columns using key:value
# Where Group is the key : Group1_1
# And Value is the value originally
# Also retaining the Response Variable
df_mergedgroups = df_groups %>%
  select(high_sig, Response) %>%
  pivot_longer(cols = starts_with("Group"),
               names_to = "Group",
```

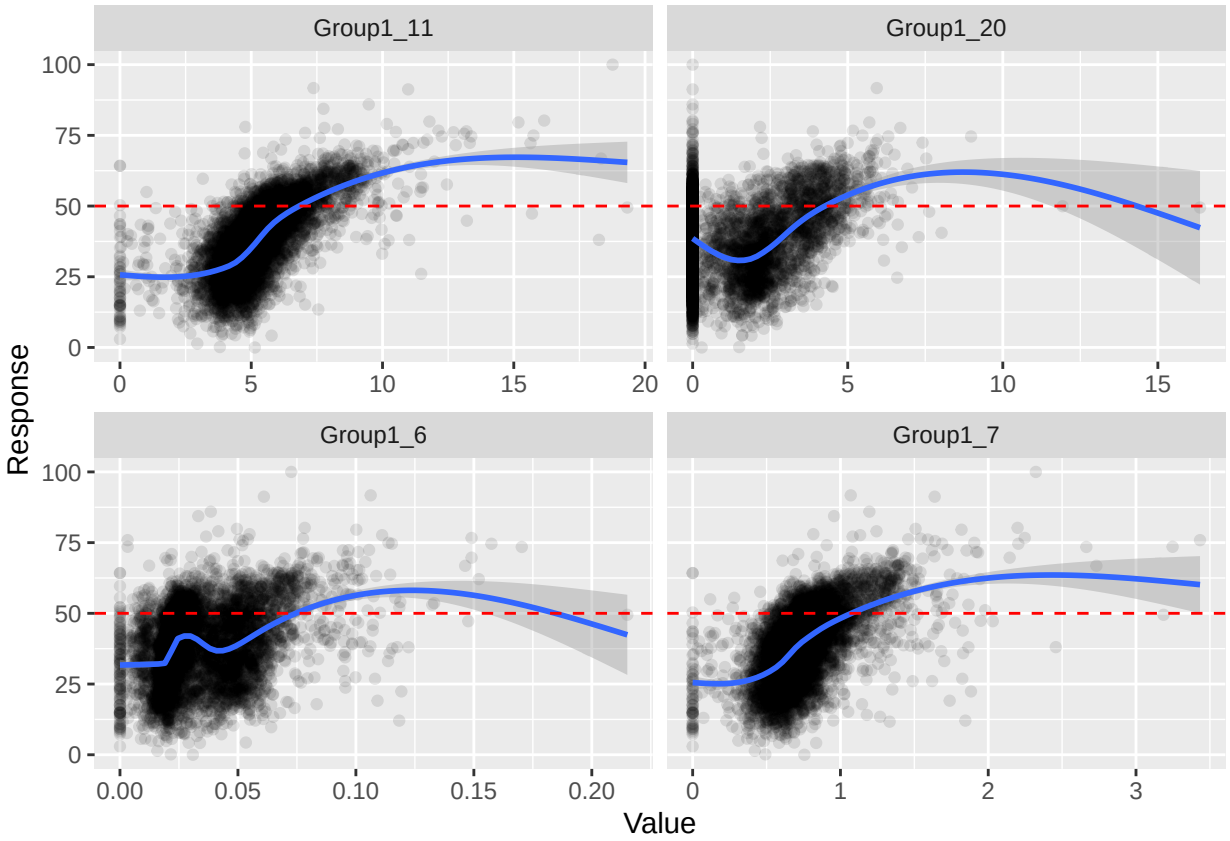
```
                values_to = "Value"
                )
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use ‘all_of(high_sig)‘ instead of ‘high_sig‘ to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

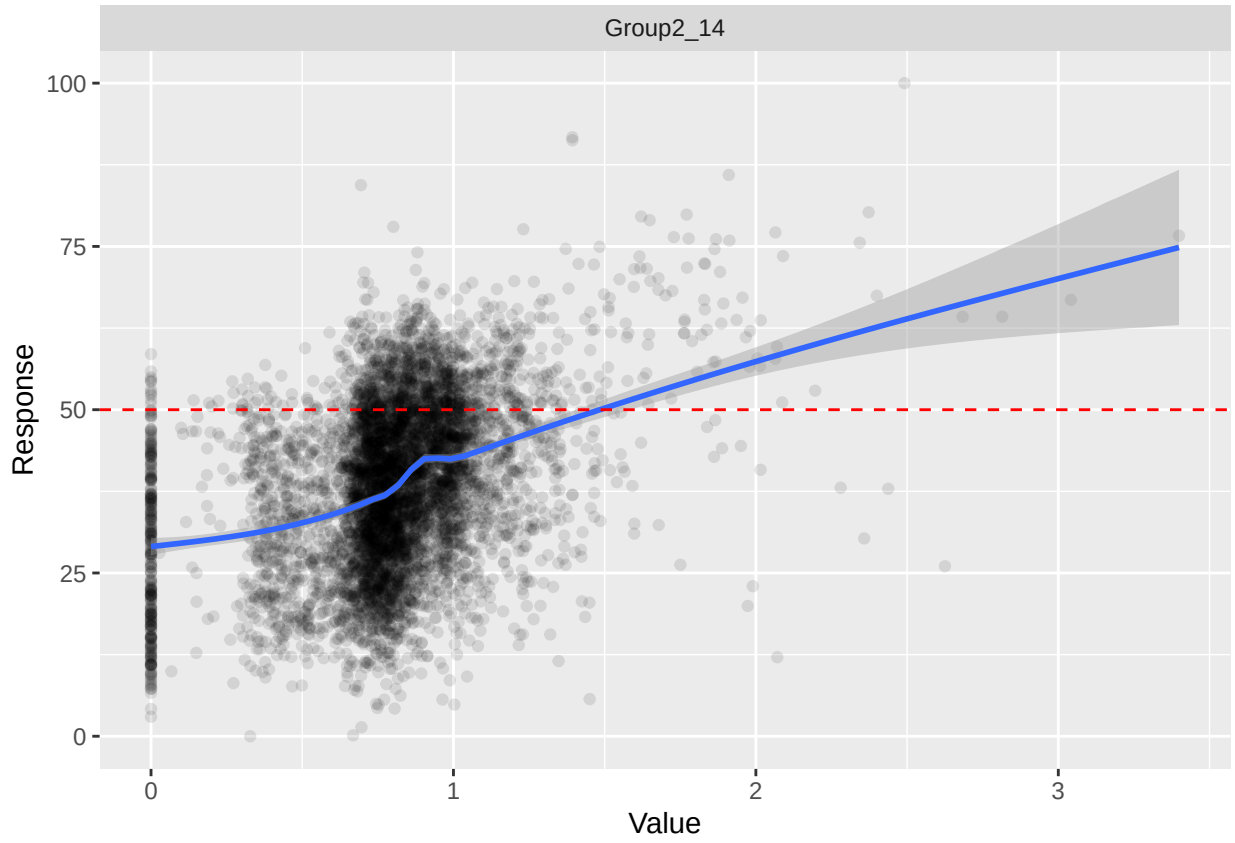
Plotting Logic

```
# Groups that have sub-groups with high confidence
groups_class = c("Group1_", "Group2_", "Group3_", "Group6_", "Group9_", "Group11_", "Group13_")

# Iterate over each group
for (group_id in groups_class){

  # Take Dataframe and create a copy
  cur_df = df_mergedgroups %>%
  # REGEX each group class (so to select only group 1, or only group 2)
  filter(str_detect(Group, group_id))


  # Create GGPLOT
  plot = ggplot(cur_df, aes(x=Value, y = Response)) +
    # Add points low alpha
    geom_point(alpha = 0.1)+
    # Perform Regression, generally gamma
    geom_smooth(method="auto", se=TRUE, fullrange=FALSE, level=0.95) +
    # Add Response Pass Rate
    geom_hline(yintercept = 50, color = "red", linetype = "dashed") +
    # Facet Wrap each sub group, allow any x axis
    facet_wrap(Group ~ ., scales="free_x")

  print(plot)

}
```

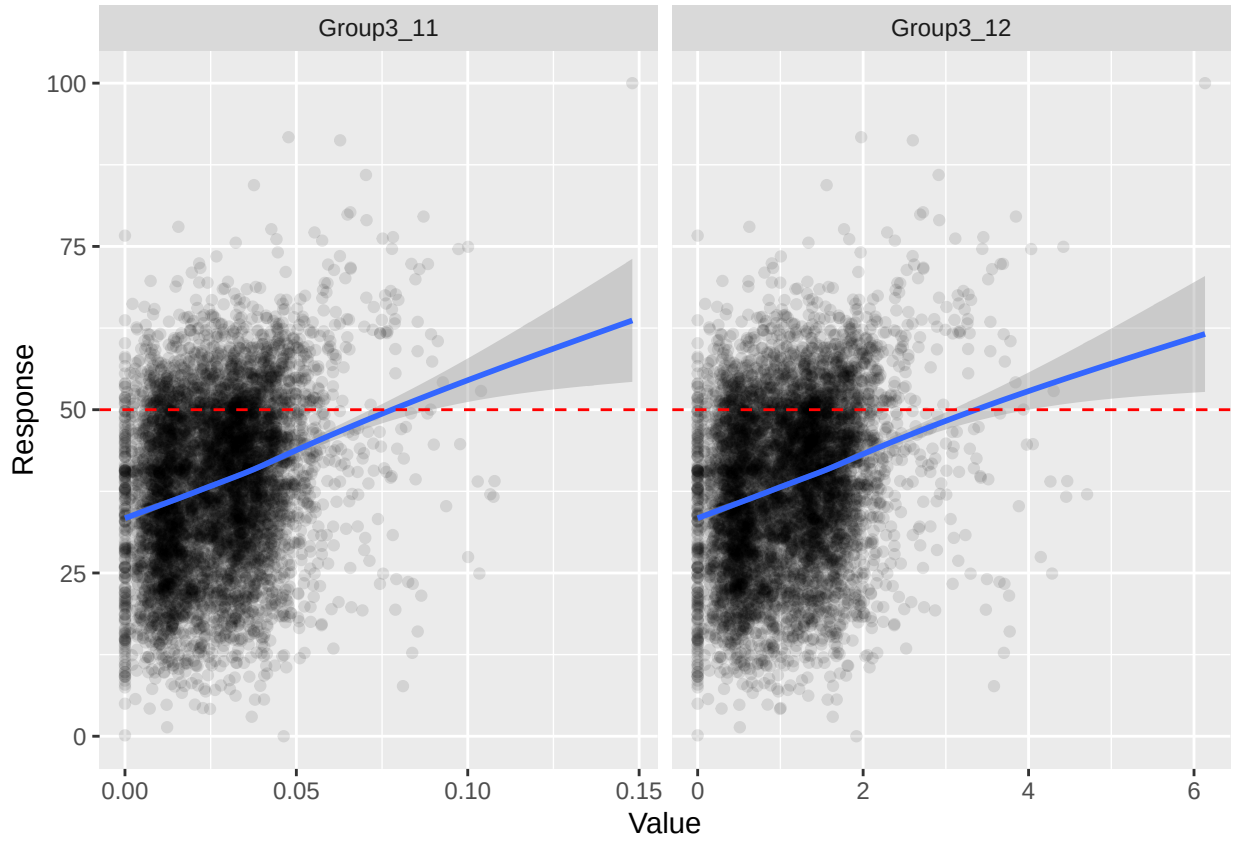
```
## ‘geom_smooth()‘ using method = ’gam’ and formula ’y ~ s(x, bs = "cs")’
```



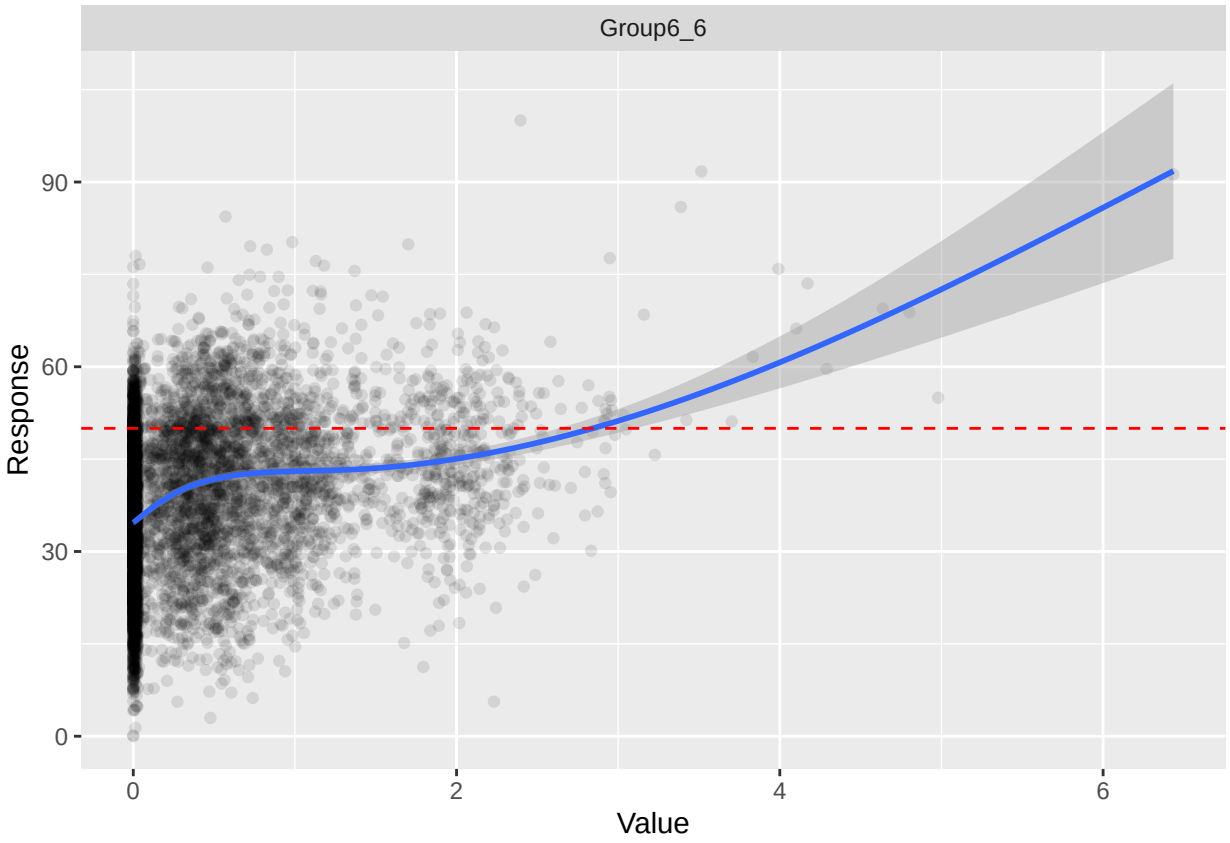

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



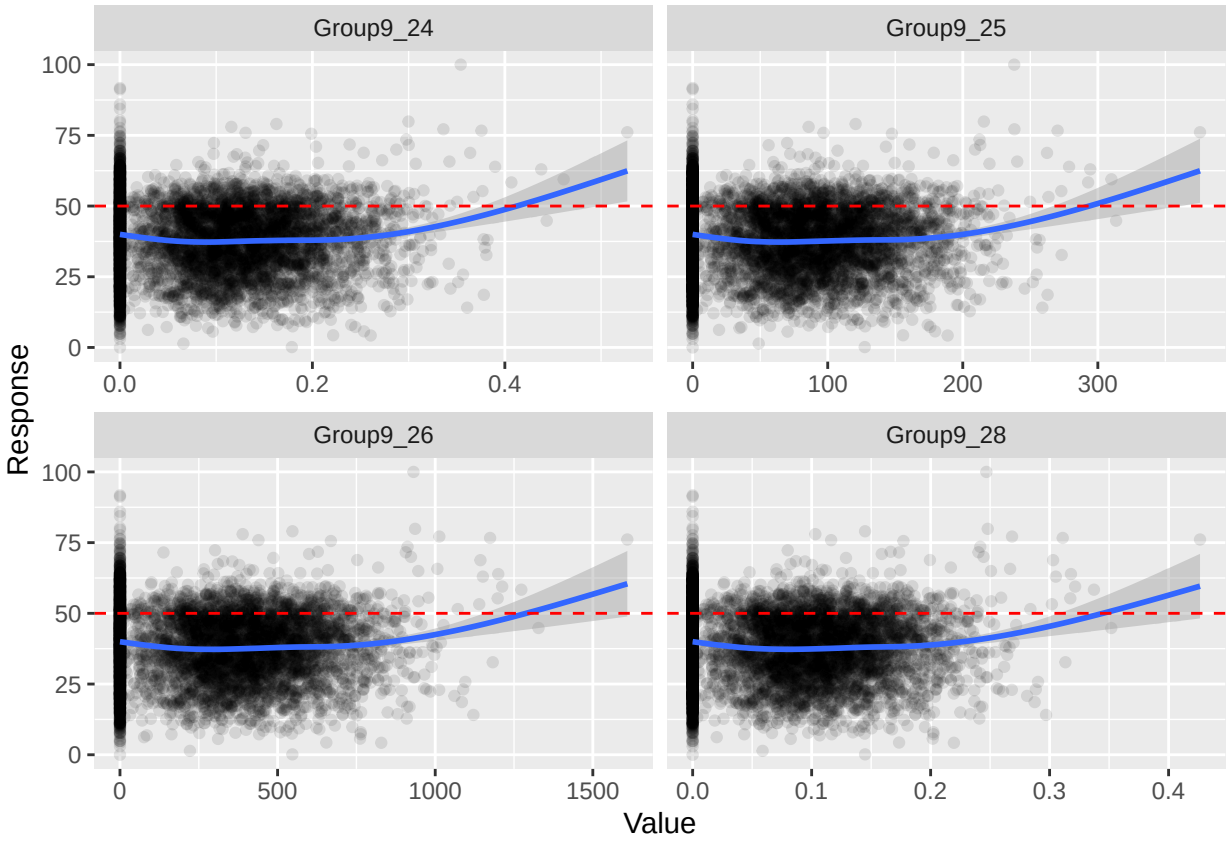

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



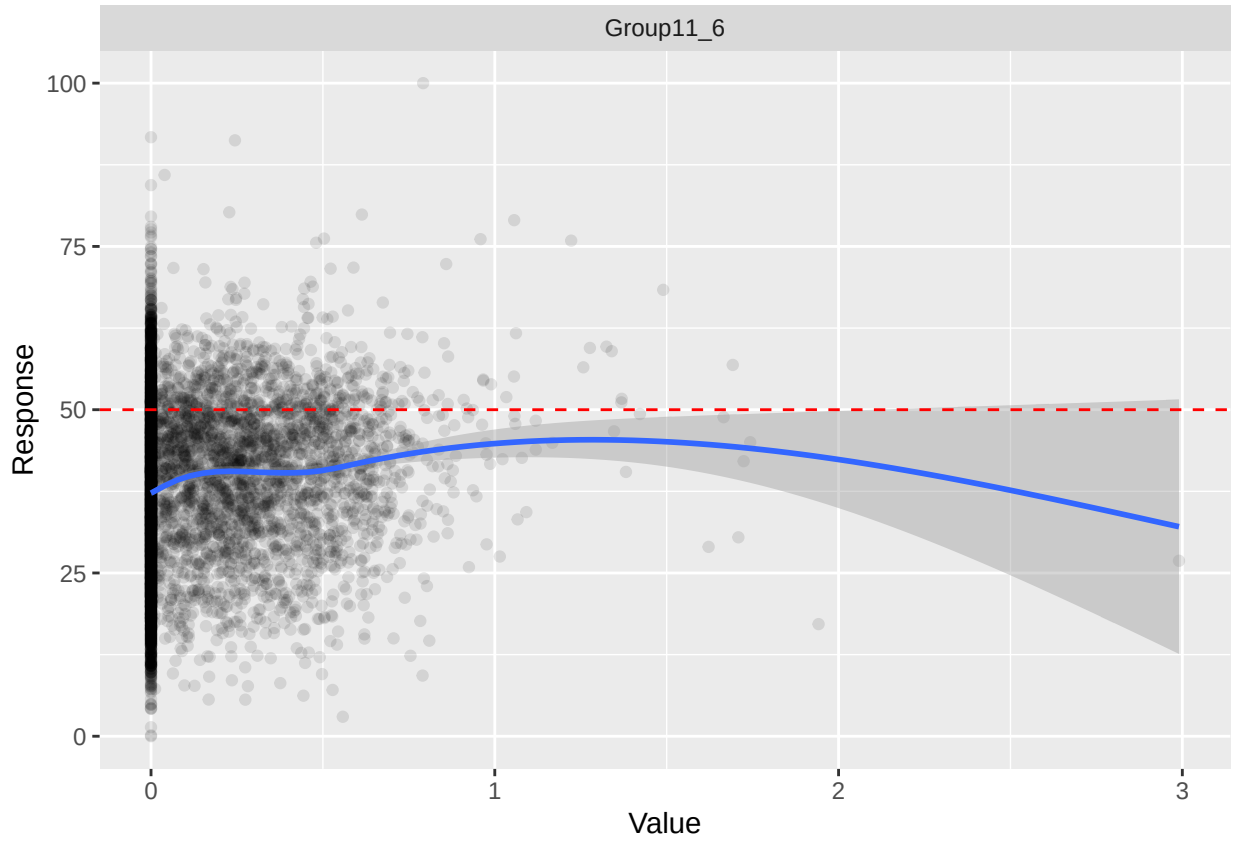

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



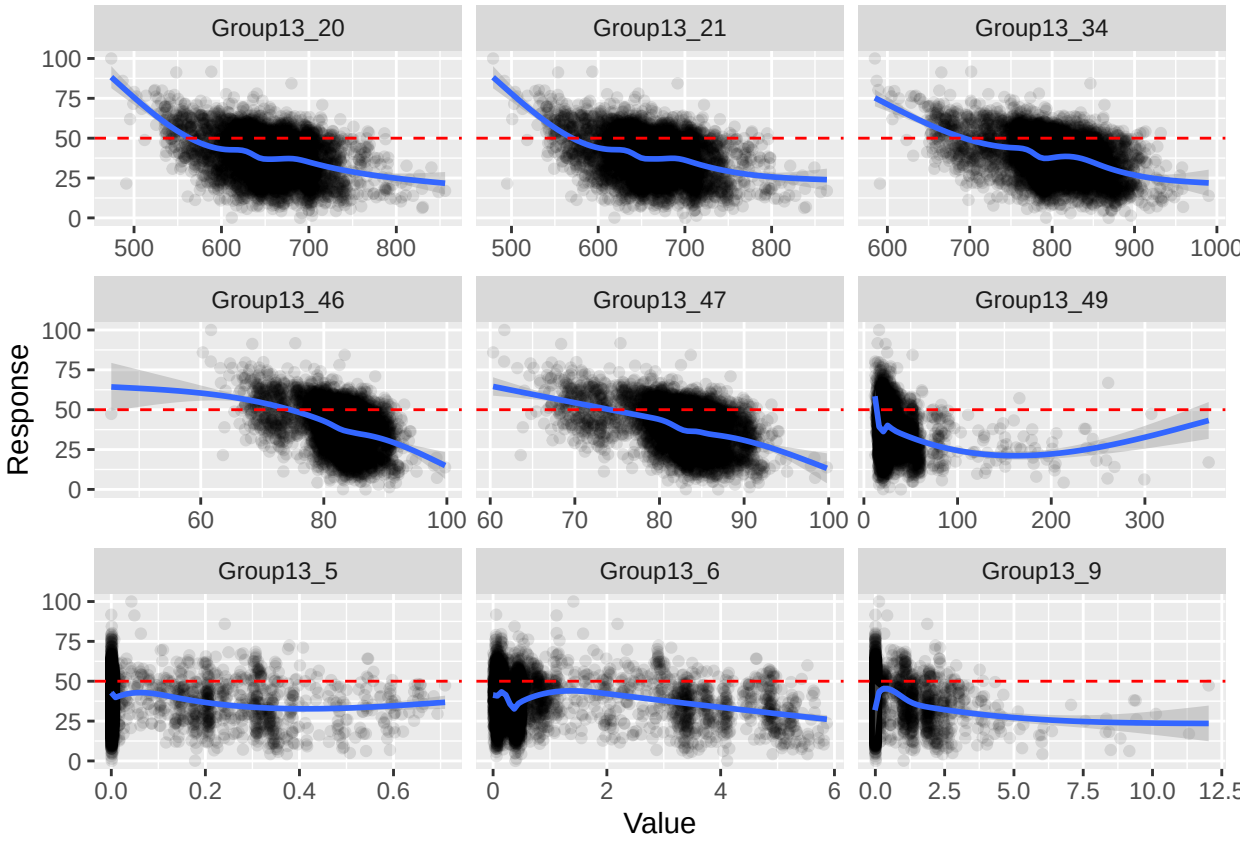

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
  # Calculate count of value=0 occurrences
  df_mergedgroups %>%
  # Filter for v < 0
  filter(Value <= 0) %>%
  count(Group) %>%
  arrange()
```

```
## # A tibble: 16 x 2
##    Group         n
##    <chr>     <int>
##  1 Group1_11    59
##  2 Group1_20  3851
##  3 Group1_6    113
##  4 Group1_7     59
##  5 Group11_6  3513
##  6 Group13_5    38
##  7 Group13_6    10
##  8 Group13_9  3930
##  9 Group2_14   285
## 10 Group3_11   186
## 11 Group3_12   186
## 12 Group6_6   1433
## 13 Group9_24  2343
## 14 Group9_25  2343
## 15 Group9_26  2343
## 16 Group9_28  2343
```