Data Analysis Code

Ryan Godin, Saman Ghazvini, William Amendola Bye, Jordyn Eovito, Vencke Gruening

2025-05-08

This code tries to reproduce a select portion of figures from Chen et al., 2023's *Deep Mutational Scanning of an Oxygen-Independent Fluorescent Protein CreiLOV for Comprehensive Profiling of Mutational and Epistatic Effects*. To run this code, please ensure that the raw data from the paper has been processed and saved in `data/processed` by running the code in `data_processing.RMD`.

The following libraries are needed for the data processing.

```
#rm(list = ls())
library(tidyr)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v purrr     1.0.4
## v forcats   1.0.0     v readr     2.1.5
## v ggplot2   3.5.1     v stringr   1.5.1
## v lubridate 1.9.4     v tibble    3.2.1
## -- 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 become errors
```

```
library(ggbeeswarm)
library(ggplot2)
library(ggpmisc)
```

```
## Loading required package: ggpp
## Registered S3 methods overwritten by 'ggpp':
##   method                  from
##   heightDetails.titleGrob ggplot2
##   widthDetails.titleGrob  ggplot2
##
## Attaching package: 'ggpp'
##
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
library(scales)
```

```
##
## Attaching package: 'scales'
##
## The following object is masked from 'package:purrr':
##
##     discard
```

```
##
## The following object is masked from 'package:readr':
##
##     col_factor
```

Figure Reproduction

Load the data

```
single_mutant_data <- read.csv("../data/processed/single_mutant_data.csv")
combinatorial_mutation_data <- read.csv("../data/processed/combinatorial_mutant_data.csv")
```

Checking the first few lines, last few lines, and dimensions of the file to see if the data loaded correctly.

```
head(single_mutant_data)
```

```
##      mutants     rep1     rep2     rep3     mean log_rep1 log_rep2 log_rep3
## 1         wt 13642.39 14855.18 15912.46 14803.35 4.134890 4.171878 4.201737
## 2 p.Arg60Asp 25759.51 26659.21 26651.36 26356.69 4.410938 4.425847 4.425719
## 3  p.Thr7Ser 25817.63 26426.82 25983.04 26075.83 4.411916 4.422045 4.414690
## 4 p.Ala29His 26195.73 26780.29 24982.15 25986.06 4.418230 4.427815 4.397630
## 5 p.Gly26Thr 25376.81 26196.46 24196.20 25256.49 4.404437 4.418243 4.383747
## 6 p.Gln47Ile 24669.23 26084.80 23552.03 24768.68 4.392156 4.416387 4.372028
##   log_mean position wt_amino_acid mutant_amino_acid
## 1 4.170360       NA          <NA>              <NA>
## 2 4.420891       60             R                 D
## 3 4.416238        7             T                 S
## 4 4.414740       29             A                 H
## 5 4.402373       26             G                 T
## 6 4.393903       47             Q                 I
```

```
tail(single_mutant_data)
```

```
##         mutants   rep1     rep2     rep3   mean log_rep1 log_rep2 log_rep3
## 2180 p.Ala11Arg 378.64 468.8357 319.7227 389.06 2.578221 2.671021 2.504773
## 2181 p.Ala11Tyr 457.16 298.7971 392.3093 382.76 2.660071 2.475376 2.593629
## 2182 p.Leu73Arg 396.47 407.6712 338.9078 381.01 2.598205 2.610310 2.530082
## 2183 p.Asn85Trp 302.48 375.7133 458.8011 379.00 2.480699 2.574857 2.661624
## 2184 p.Gly32Pro 324.20 403.8464 347.1191 358.39 2.510810 2.606216 2.540478
## 2185 p.Leu20Trp 337.22 259.4125 263.2738 286.63 2.527908 2.413991 2.420408
##      log_mean position wt_amino_acid mutant_amino_acid
## 2180 2.590022       11             A                 R
## 2181 2.582923       11             A                 Y
## 2182 2.580942       73             L                 R
## 2183 2.578638       85             N                 W
## 2184 2.554353       32             G                 P
## 2185 2.457328       20             L                 W
```

```
dim(single_mutant_data)
```

```
## [1] 2185   12
```

```
head(combinatorial_mutation_data)
```

```
##     mutants     Rep1      Rep2      Rep3     mean Rep1_log Rep2_log Rep3_log
## 1        wt 11436.15 11590.594  8621.599 10549.45 4.058280 4.064106 3.935588
## 2 p.Thr7Ser 15190.43 15177.976 10986.791 13785.07 4.181570 4.181214 4.040871
```

```
## 3 p.Arg5Asp 11742.95 14620.049 12136.214 12833.07 4.069777 4.164949 4.084083
## 4 p.Thr7His 12282.68 13610.057 12035.502 12642.75 4.089293 4.133860 4.080464
## 5 p.Leu4Asn 12075.63 10924.854  8659.384 10553.29 4.081910 4.038416 3.937487
## 6 p.Gly3Glu 11628.62  9858.208  9738.470 10408.43 4.065528 3.993798 3.988491
##   mean_log position mutation_count expected_fluorescence epistasis
## 1 4.023230       NA              0              4.023230         0
## 2 4.139409        7              1              4.139409         0
## 3 4.108331        5              1              4.108331         0
## 4 4.101841        7              1              4.101841         0
## 5 4.023388        4              1              4.023388         0
## 6 4.017385        3              1              4.017385         0
##   strong_epistasis
## 1            FALSE
## 2            FALSE
## 3            FALSE
## 4            FALSE
## 5            FALSE
## 6            FALSE
```

```
tail(combinatorial_mutation_data)
```

```
##
## 165423 p.Gly3Glu, p.Leu4Asn, p.Arg5Asp, p.Thr7His, p.Ala29Lys, p.Gly34Thr, p.Gln47Arg, p.Arg60Asn, p
## 165424 p.Gly3Glu, p.Leu4Asn, p.Arg5Asp, p.Thr7Ser, p.Ala29Lys, p.Gly34Thr, p.Gln47Cys, p.Arg60Asn, p
## 165425 p.Gly3Glu, p.Leu4Asn, p.Arg5Asp, p.Thr7His, p.Ala29Lys, p.Gly34Thr, p.Gln47Cys, p.Arg60Asn, p
## 165426 p.Gly3Glu, p.Leu4Asn, p.Arg5Asp, p.Thr7His, p.Ala29Lys, p.Gly34Thr, p.Gln47Ile, p.Arg60Asn, p
## 165427 p.Gly3Glu, p.Leu4Asn, p.Arg5Asp, p.Thr7His, p.Ala29His, p.Gly34Thr, p.Gln47Cys, p.Arg60Asn, p
## 165428 p.Gly3Glu, p.Leu4Asn, p.Arg5Asp, p.Thr7His, p.Ala29His, p.Gly34Thr, p.Gln47Val, p.Arg60Asn, p
##            Rep1     Rep2     Rep3     mean Rep1_log Rep2_log Rep3_log mean_log
## 165423 3431.368 3833.428 2841.180 3368.659 3.535467 3.583587 3.453499 3.527457
## 165424 2824.791 2871.556 1734.183 2476.844 3.450986 3.458117 3.239095 3.393899
## 165425 2209.536 1530.388 2579.866 2106.596 3.344301 3.184801 3.411597 3.323581
## 165426 1669.378 2256.697 2042.302 1989.459 3.222555 3.353473 3.310120 3.298735
## 165427 1932.942 1633.038 1767.045 1777.675 3.286219 3.212996 3.247248 3.249852
## 165428 1263.616 1140.008 1135.488 1179.704 3.101615 3.056908 3.055182 3.071773
##        position mutation_count expected_fluorescence  epistasis
## 165423      113             15              4.017385 -0.4899283
## 165424      113             15              4.017385 -0.6234868
## 165425      113             15              4.017385 -0.6938040
## 165426      113             15              4.017385 -0.7186503
## 165427      113             15              4.017385 -0.7675331
## 165428      113             15              4.017385 -0.9456123
##        strong_epistasis
## 165423            FALSE
## 165424             TRUE
## 165425             TRUE
## 165426             TRUE
## 165427             TRUE
## 165428             TRUE
```

```
dim(combinatorial_mutation_data)
```

```
## [1] 165428     14
```

Reproducing Figure 1B

We first calculate the R correlation coefficient and the regression equation so we can annotate the ggplots with them.

```
# Calculate regression data and make labels for plots
regression_12 <- lm(single_mutant_data$rep1 ~ single_mutant_data$rep2)
slope_12 <- coef(regression_12)[2]
r_12 <- cor(single_mutant_data$rep1, single_mutant_data$rep2, use = "complete.obs")
r_label_12 <- paste("italic(R) == ", format(round(r_12, 3), nsmall = 3))
eq_12 <- paste("y == ", round(slope_12, 3), " * x", sep = "")

regression_13 <- lm(single_mutant_data$rep1 ~ single_mutant_data$rep3)
slope_13 <- coef(regression_13)[2]
r_13 <- cor(single_mutant_data$rep1, single_mutant_data$rep3, use = "complete.obs")
r_label_13 <- paste("italic(R) == ", format(round(r_13, 3), nsmall = 3))
eq_13 <- paste("y == ", round(slope_13, 3), " * x", sep = "")

regression_23 <- lm(single_mutant_data$rep3 ~ single_mutant_data$rep2)
slope_23 <- coef(regression_23)[2]
r_23 <- cor(single_mutant_data$rep2, single_mutant_data$rep3, use = "complete.obs")
r_label_23 <- paste("italic(R) == ", format(round(r_23, 3), nsmall = 3))
eq_23 <- paste("y == ", round(slope_23, 3), " * x", sep = "")
```

We next generate the annotated figures and save the subplots to `figures/fig_1b`.

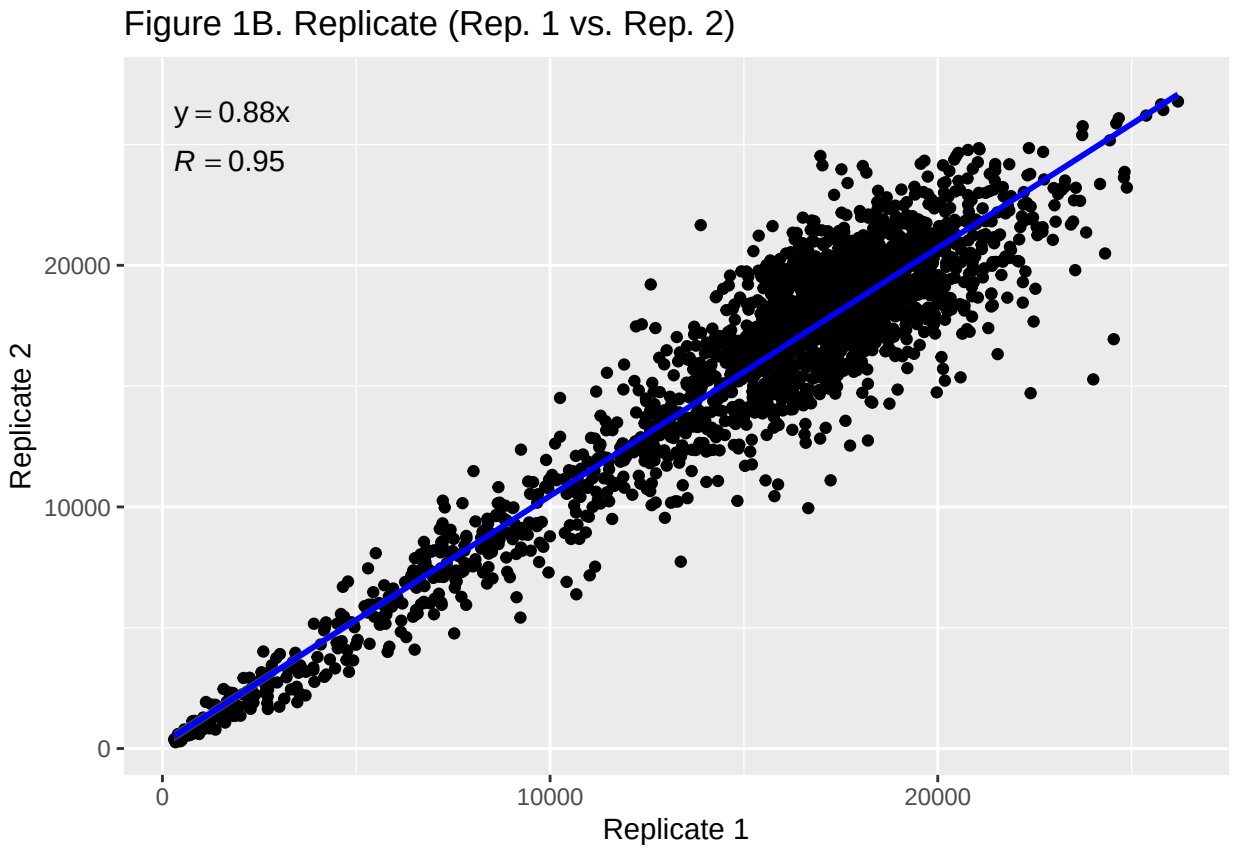
```
fig_1b_1 <- ggplot(single_mutant_data, aes(x = rep1, y = rep2)) +
  geom_point() +
  geom_smooth(method = "lm", col = "blue") +
  annotate(
    "text",
    x = min(single_mutant_data$rep1),
    y = max(single_mutant_data$rep2),
    label = eq_12,
    parse = TRUE,
    hjust = 0,
    vjust = 1,
    size = 4
  ) +
  annotate(
    "text",
    x = min(single_mutant_data$rep1),
    y = max(single_mutant_data$rep2) - 2000,
    label = r_label_12,
    parse = TRUE,
    hjust = 0,
    vjust = 1,
    size = 4
  ) +
  labs(title = "Figure 1B. Replicate (Rep. 1 vs. Rep. 2)", x = "Replicate 1", y = "Replicate 2")

fig_1b_2 <- ggplot(single_mutant_data, aes(x = rep1, y = rep3)) +
  geom_point() +
  geom_smooth(method = "lm", col = "green") +
  annotate(
    "text",
```

```
    x = min(single_mutant_data$rep1),
    y = max(single_mutant_data$rep3),
    label = eq_13,
    parse = TRUE,
    hjust = 0,
    vjust = 1,
    size = 4
  ) +
  annotate(
    "text",
    x = min(single_mutant_data$rep1),
    y = max(single_mutant_data$rep3) - 2000,
    label = r_label_13,
    parse = TRUE,
    hjust = 0,
    vjust = 1,
    size = 4
  ) +
  labs(title = "Figure 1B. Replicate (Rep. 1 vs. Rep. 3)", x = "Replicate 1", y = "Replicate 3")

fig_1b_3 <- ggplot(single_mutant_data, aes(x = rep2, y = rep3)) +
  geom_point() +
  geom_smooth(method = "lm", col = "red") +
  annotate(
    "text",
    x = min(single_mutant_data$rep2),
    y = max(single_mutant_data$rep3),
    label = eq_23,
    parse = TRUE,
    hjust = 0,
    vjust = 1,
    size = 4
  ) +
  annotate(
    "text",
    x = min(single_mutant_data$rep2),
    y = max(single_mutant_data$rep3) - 2000,
    label = r_label_23,
    parse = TRUE,
    hjust = 0,
    vjust = 1,
    size = 4
  ) +
  labs(title = "Figure 1B. Replicate (Rep. 2 vs. Rep. 3)", x = "Replicate 2", y = "Replicate 3")


# Print and save the plots and associated data.
ggsave("../figures/fig_1b/fig_1b_1.png", plot = fig_1b_1)
```

```
## Saving 6.5 x 4.5 in image
## `geom_smooth()` using formula = 'y ~ x'
```

```
print(fig_1b_1)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



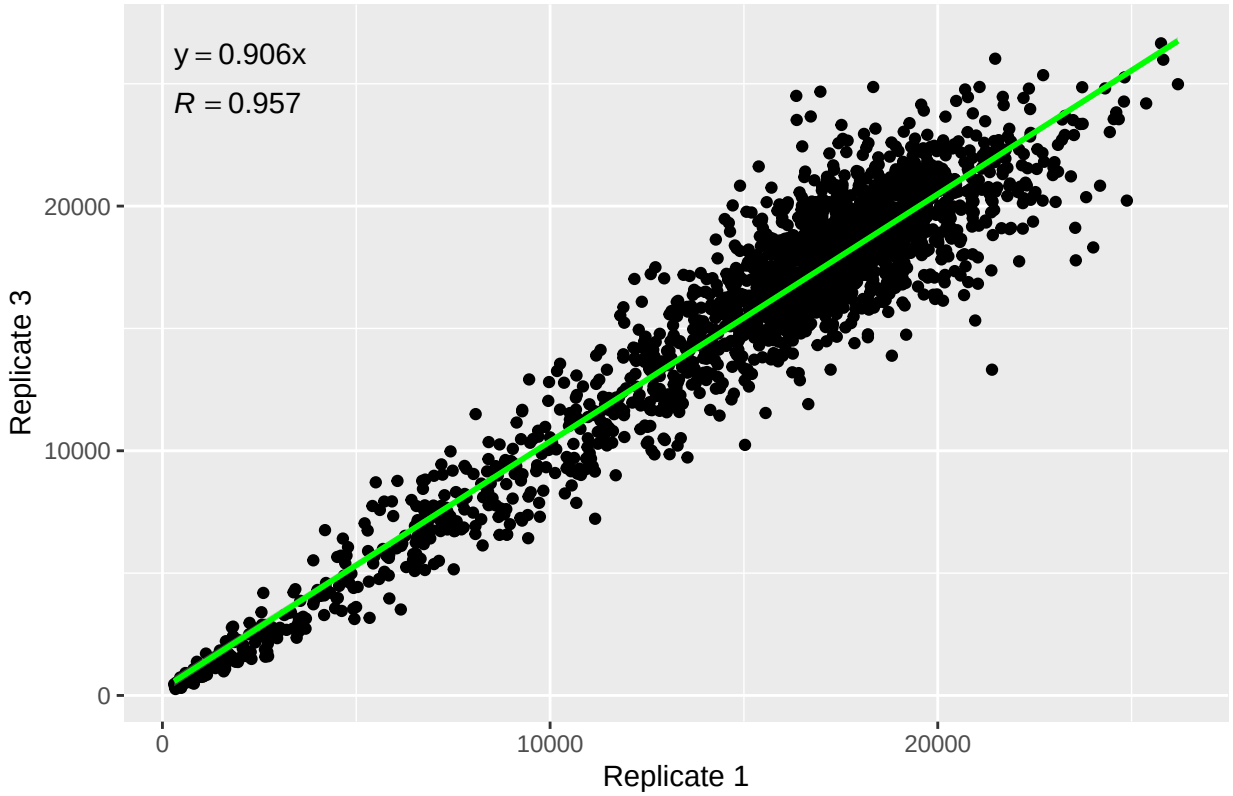

```
ggsave("../figures/fig_1b/fig_1b_2.png", plot = fig_1b_2)
```

```
## Saving 6.5 x 4.5 in image
## `geom_smooth()` using formula = 'y ~ x'
```

```
print(fig_1b_2)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

Figure 1B. Replicate (Rep. 1 vs. Rep. 3)

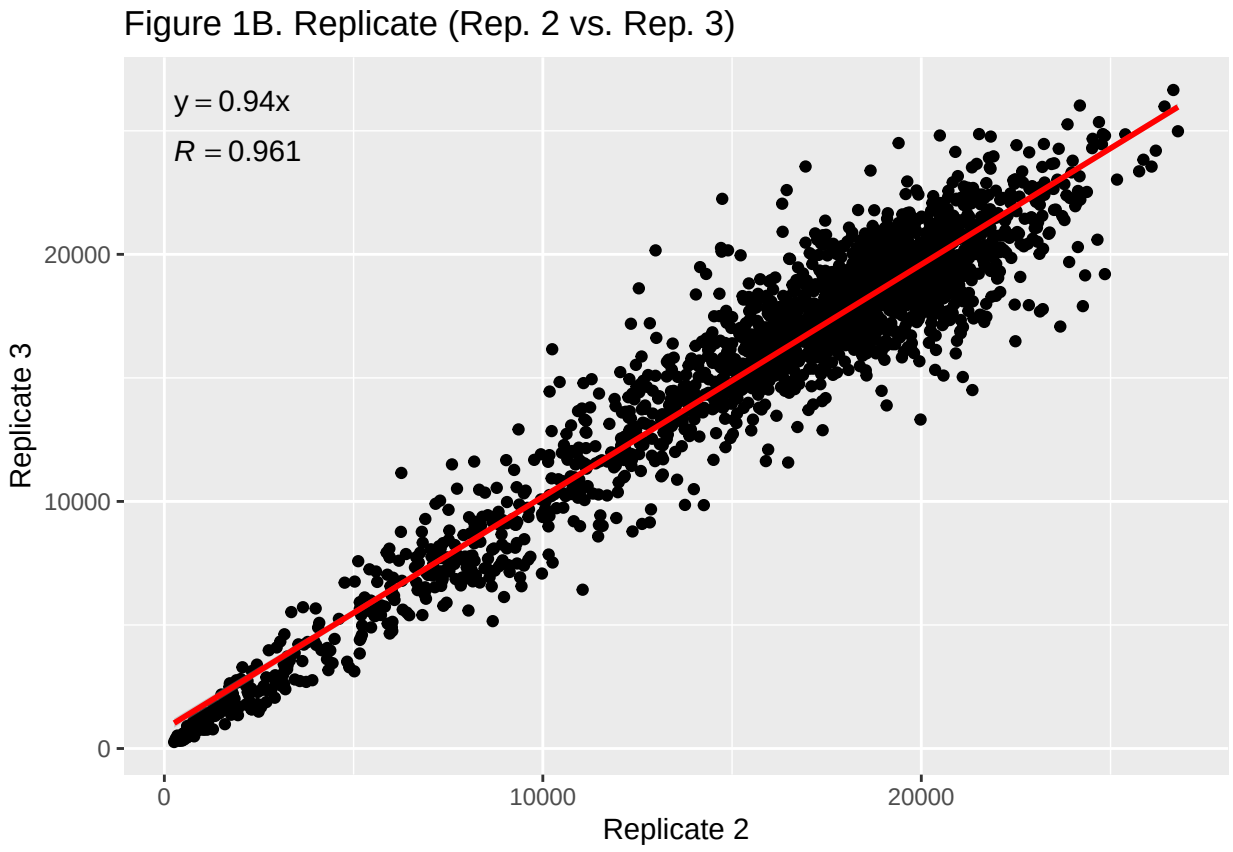



```
ggsave("../figures/fig_1b/fig_1b_3.png", plot = fig_1b_3)
```

```
## Saving 6.5 x 4.5 in image
## `geom_smooth()` using formula = 'y ~ x'
```

```
print(fig_1b_3)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Reproducing Figure 2A

We first load the target wild-type value so that it can be highlighted in the histogram.

```
target_value <- single_mutant_data[1,"log_mean"]
print(target_value)
```

```
## [1] 4.17036
```

We next generate and format the histogram.

```
## Plot and format the histogram to see if it looks like the paper.
fig_2a <- ggplot(single_mutant_data, aes(x = log_mean, y = after_stat(ndensity))) +
  geom_histogram(
    aes(fill = after_stat(xmin) <= target_value &
          after_stat(xmax) > target_value),
    bins = 47,
    color = "black"
  ) +
  scale_fill_manual(values = c("TRUE" = "red", "FALSE" = "black"),
                    guide = "none") +
  labs(x = "Fluorescence (log)", y = "Frequency", title = "Figure 2A. Replicate") +
  scale_x_continuous(
    limits = c(2, 5),
    breaks = seq(2, 5, by = 0.5),
    labels = sprintf("%.2f", seq(2, 5, by = 0.5)),
    expand = c(0, 0)
  ) +
  scale_y_continuous(expand = c(0, 0)) +
```

```
  theme_classic() +
  theme(
    axis.text = element_text(size = 14, color = "black"),
    axis.title = element_text(size = 16, color = "black"),
    axis.ticks = element_line(color = "black", size = 0.8),
    axis.ticks.length = unit(0.3, "cm"),
    axis.line = element_line(color = "black", size = 0.8)
  )
```

```
## Warning: The `size` argument of `element_line()` is deprecated as of ggplot2 3.4.0.
## i Please use the `linewidth` argument instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

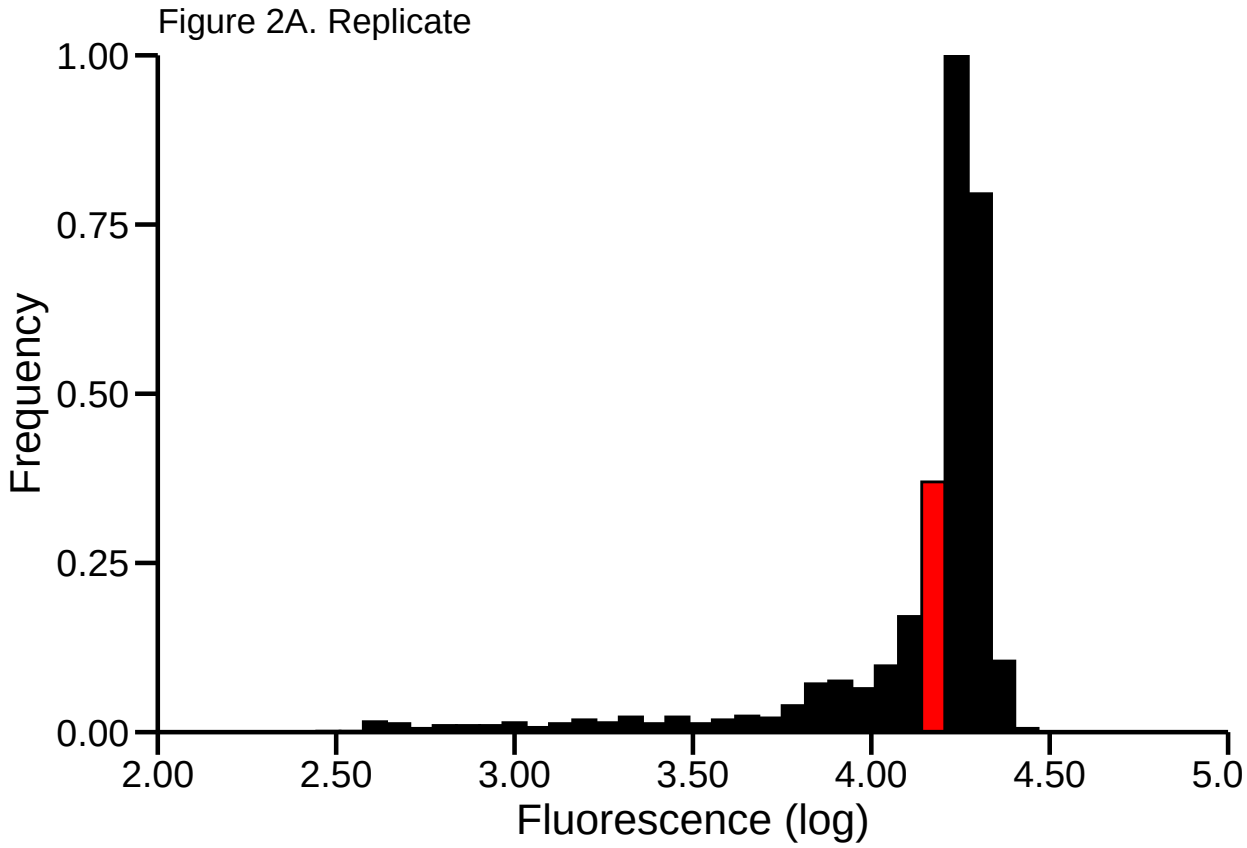
```
ggsave("../figures/fig_2a.png", plot = fig_2a)
```

```
## Saving 6.5 x 4.5 in image
```

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

```
print(fig_2a)
```

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



Reproducing Figure 2B

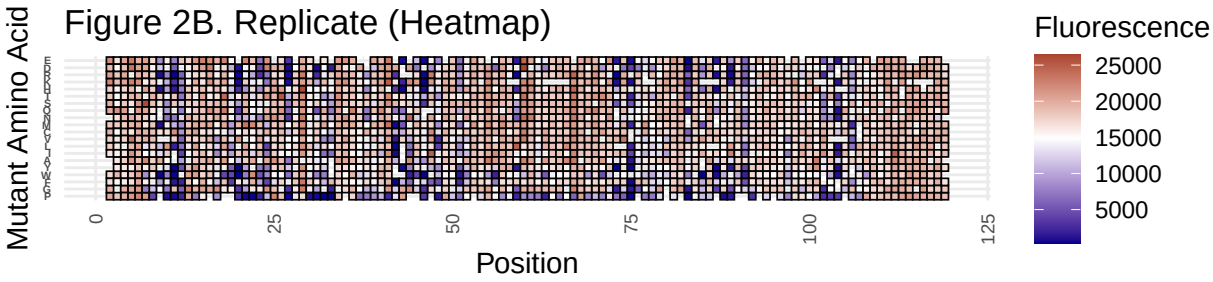
We first extract the necessary data and format it for the heatmap.

```
# Restrict positions to 1-120 and drops those with invalid amino acids.
df_parsed <- filter(
  single_mutant_data,
  position >= 1,
  position <= 120,
  !is.na(wt_amino_acid),
  !is.na(mutant_amino_acid)
)

# Custom amino acid order for Y-axis
aa_order <- c(
  'P',
  'G',
  'F',
  'W',
  'Y',
  'A',
  'I',
  'L',
  'V',
  'C',
  'M',
  'N',
  'Q',
  'S',
  'T',
  'H',
  'K',
  'R',
  'D',
  'E'
)

df_parsed <- df_parsed %>%
  mutate(mutant_amino_acid = factor(mutant_amino_acid, levels = aa_order)
)
```

We then plot the heatmap.

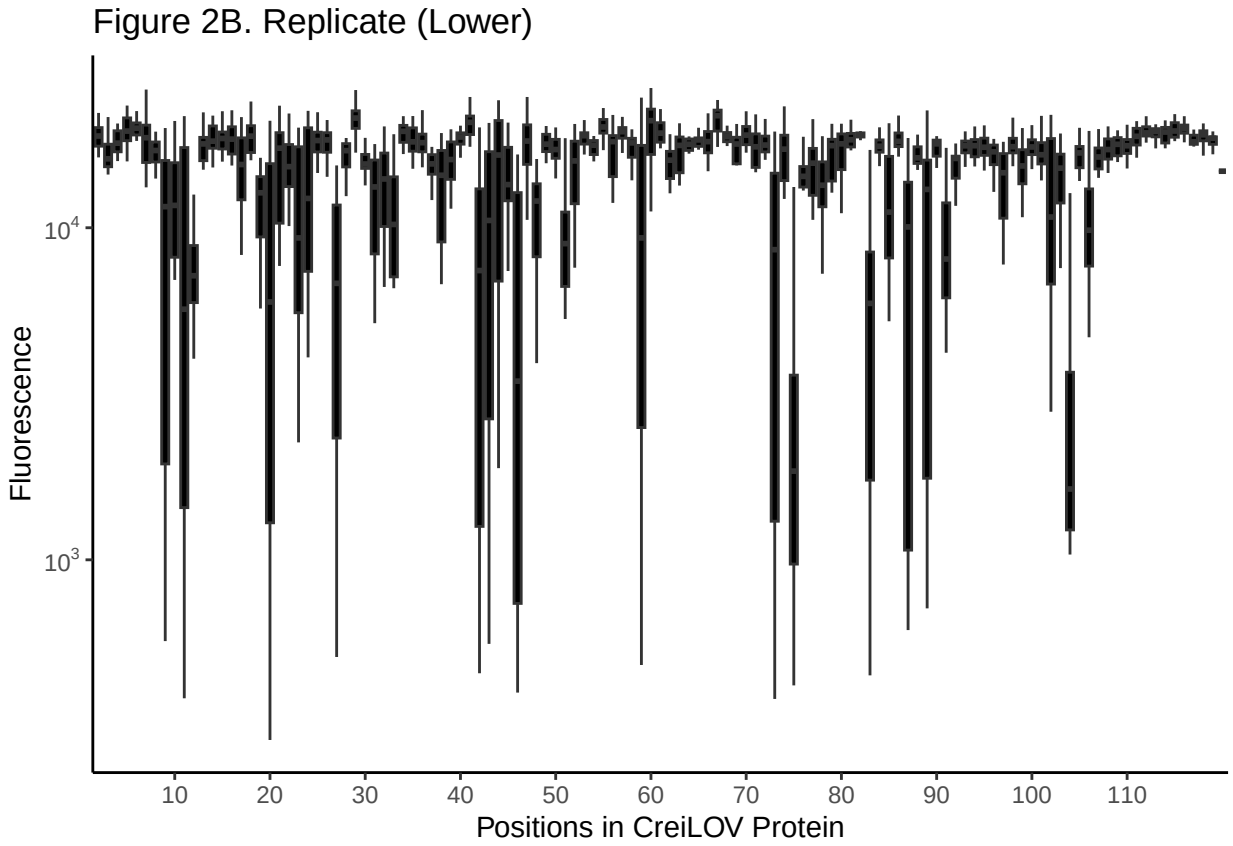
```
# Plot heatmap with all desired features
fig_2b_heatmap <-  ggplot(df_parsed, aes(x = position, y = mutant_amino_acid, fill = mean)) +
  geom_tile(color = "black", linewidth = 0.3) +  # Black borders for tiles
  scale_fill_gradient2(
    low = "darkblue",
    mid = "white",
    high = "darkred",
    midpoint = 14803,
    na.value = "gray80",
    # Lighter gray for missing values
    name = "Fluorescence"
  ) +
  theme_minimal() +
```

```
  coord_fixed() +  # Makes tiles square
  theme(
    axis.text.x = element_text(
      angle = 90,
      vjust = 0.5,
      hjust = 1,
      size = 7
    ),
    axis.text.y = element_text(size = 4, face = "bold"),
    legend.position = "right",
    legend.key.height = unit(0.5, "cm")  # Taller legend for better color gradient
  ) +
  labs(x = "Position", y = "Mutant Amino Acid", title = "Figure 2B. Replicate (Heatmap)")

ggsave("../figures/fig_2b_heatmap.png", plot = fig_2b_heatmap,width = 10, height = 6)
print(fig_2b_heatmap)
```



We next generate the figure for the second part of Figure 2B.

```
fig_2b_lower <- ggplot(single_mutant_data, mapping = aes(x = as.factor(position), y = mean)) +
  geom_boxplot(fill = "black", outlier.shape = NA) + scale_x_discrete(breaks = seq(10, 110, by = 10)) +
  scale_y_log10(
    breaks = c(1e3, 1e4),
    labels = scales::trans_format("log10", math_format(10^.x))
  ) +
  labs(x = "Positions in CreiLOV Protein", y = "Fluorescence", title = "Figure 2B. Replicate (Lower)") +
  theme_classic()

print(fig_2b_lower)
```




```
ggsave("../figures/fig_2b_lower.png", fig_2b_lower, width = 16, height = 2)
```

Reproducing Figure 3A

We first set up vector for specified positions as detailed in paper and then select the mutants they singled out and highlighted red in their figure (found in Table S2).

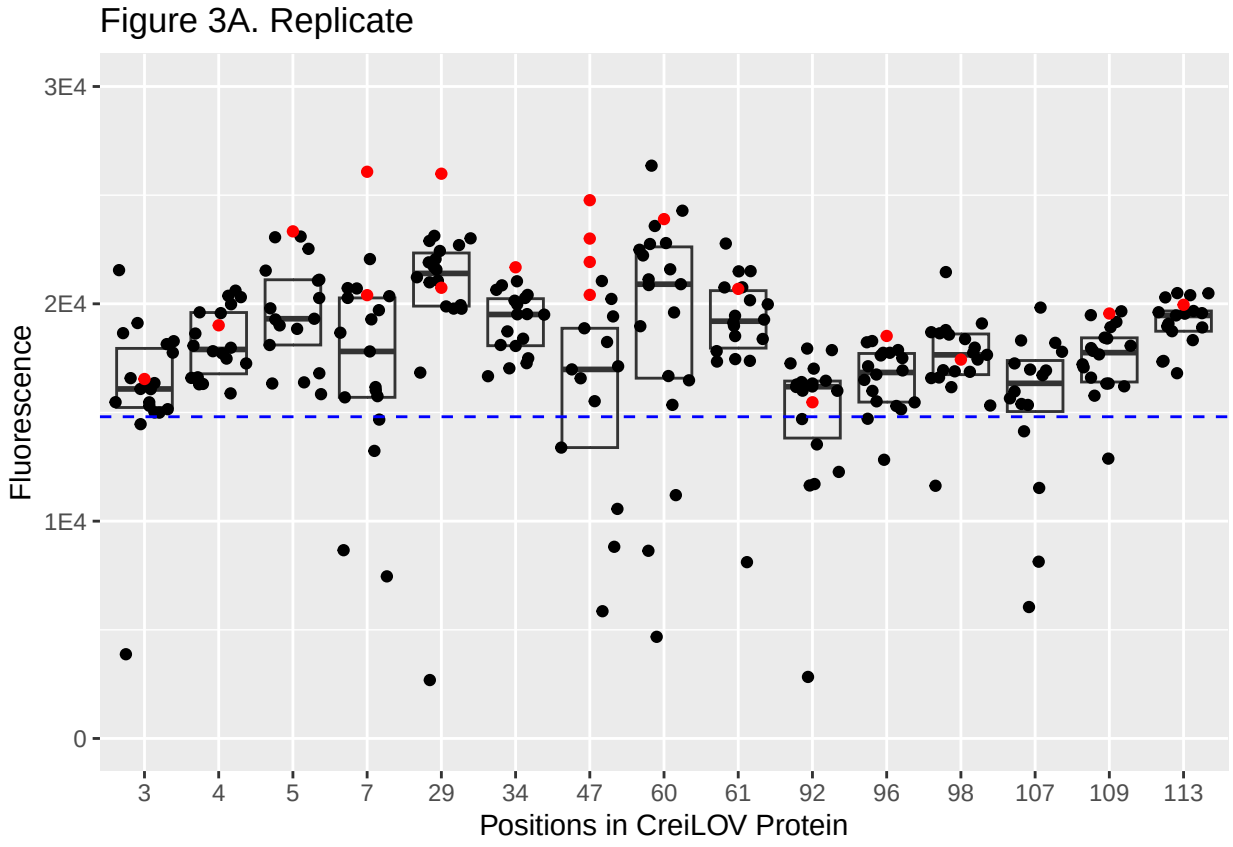
Note: Missing Val107Met - there's no Val107Met data from the s1 single mutant set)

```
table_s1 <- single_mutant_data
table_s2 <- combinatorial_mutation_data

target_positions <- c(3,4,5,7,29,34,47,60,61,92,96,98,107,109,113)
# Filters selected mutations in combinatorial mutagenesis, found in table S2
selected_mutants <- unlist(c(table_s2[2:21,1]))
# Filters table for selected positions - chosen at various distances
# from FMN chromophore, with high tolerance for amino acid substitutions
# - little detail in paper
filtered_table_s1 <- table_s1[(table_s1$position %in% target_positions), ]
filtered_table_s1_graph <- filter(
  filtered_table_s1,
  !(filtered_table_s1[[1]] %in% selected_mutants)
)
filtered_table_s2 <- table_s2[2:21, ]
selected_fluorescence <- table_s1[table_s1[[1]] %in% selected_mutants, ]
# Mean wildtype fluorescence extracted to overlay on graph as horizontal line
wt_mean <- as.numeric(c(table_s1[1,5]))
```

We now graph the data to reproduce Figure 3A. To graph fluorescence at each position, we use geom_jitter

since it shows points at each position without overlapping. Note that the Horizontal dashed line indicates the wildtype mean fluorescence value.

```
fig_3a <- ggplot(filtered_table_s1_graph, aes(x=factor(position), y=mean)) +
  geom_boxplot(alpha = 0, outlier.shape = NA, coef = 0) +
  geom_jitter() +
  geom_hline(yintercept = wt_mean, linetype = "dashed", color = "blue") +
  geom_point(selected_fluorescence,
    mapping = aes(x = factor(position), y = mean), color = "red"
  ) +
  scale_y_continuous(
    limits = c(0,3e4),
    breaks = c(0, 1e4, 2e4, 3e4),
    labels = c("0", "1E4", "2E4", "3E4")
  ) +
  labs(x = "Positions in CreiLOV Protein", y = "Fluorescence", title = "Figure 3A. Replicate")

ggsave("../figures/fig_3a.png", fig_3a, width = 7, height = 2.5)
print(fig_3a)
```



Reproducing Figure 3B

The authors combined the replicates for the fluorescence measurement when genrating Figure 3B, so we must first process the data using pivot longer before we can reproduce the figure.

```
combinatorial_mutation_data_long <- pivot_longer(
    combinatorial_mutation_data, cols = c("Rep1", "Rep2", "Rep3"),
    names_to = "replicate",
```

```
    values_to = "value"
)

head(combinatorial_mutation_data_long)
```

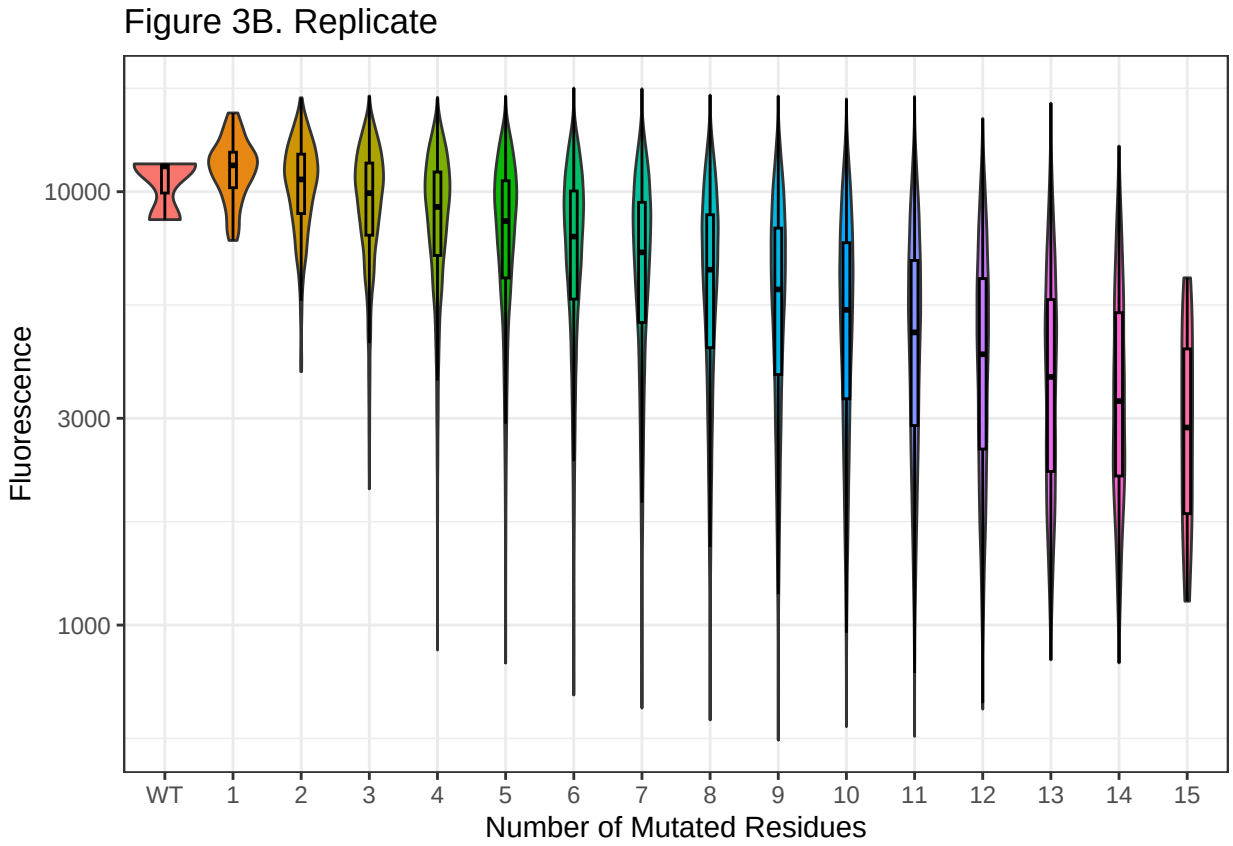
```
## # A tibble: 6 x 13
##   mutants     mean Rep1_log Rep2_log Rep3_log mean_log position mutation_count
##   <chr>      <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <int>          <int>
## 1 wt        10549.     4.06     4.06     3.94     4.02       NA              0
## 2 wt        10549.     4.06     4.06     3.94     4.02       NA              0
## 3 wt        10549.     4.06     4.06     3.94     4.02       NA              0
## 4 p.Thr7Ser 13785.     4.18     4.18     4.04     4.14        7              1
## 5 p.Thr7Ser 13785.     4.18     4.18     4.04     4.14        7              1
## 6 p.Thr7Ser 13785.     4.18     4.18     4.04     4.14        7              1
## # i 5 more variables: expected_fluorescence <dbl>, epistasis <dbl>,
## #   strong_epistasis <lgl>, replicate <chr>, value <dbl>
```

```
dim(combinatorial_mutation_data_long)
```

```
## [1] 496284     13
```

We are now ready to generate the figure.

```
combinatorial_mutation_data_long$mutation_count <- factor(
  combinatorial_mutation_data_long$mutation_count
)

fig_3b <- ggplot(
  combinatorial_mutation_data_long,
  aes(x = mutation_count, y = value, fill = mutation_count)
) +
  geom_violin() +
  geom_boxplot(width = 0.1,
               color = "black",
               outlier.shape = NA) +
  theme_bw() +
  scale_y_log10() +
  scale_x_discrete(labels = c("0" = "WT")) +
  labs(x = "Number of Mutated Residues", y = "Fluorescence", title = "Figure 3B. Replicate") +
  theme(legend.position = "none")


ggsave("../figures/fig_3b.png", plot = fig_3b, width = 10, height = 6)  # Adjust width and height here
print(fig_3b)
```



Generate Figure 3D

We first extract the data with more than one mutation since the single mutations (and the wild-type) by definition do not have any epistasis.

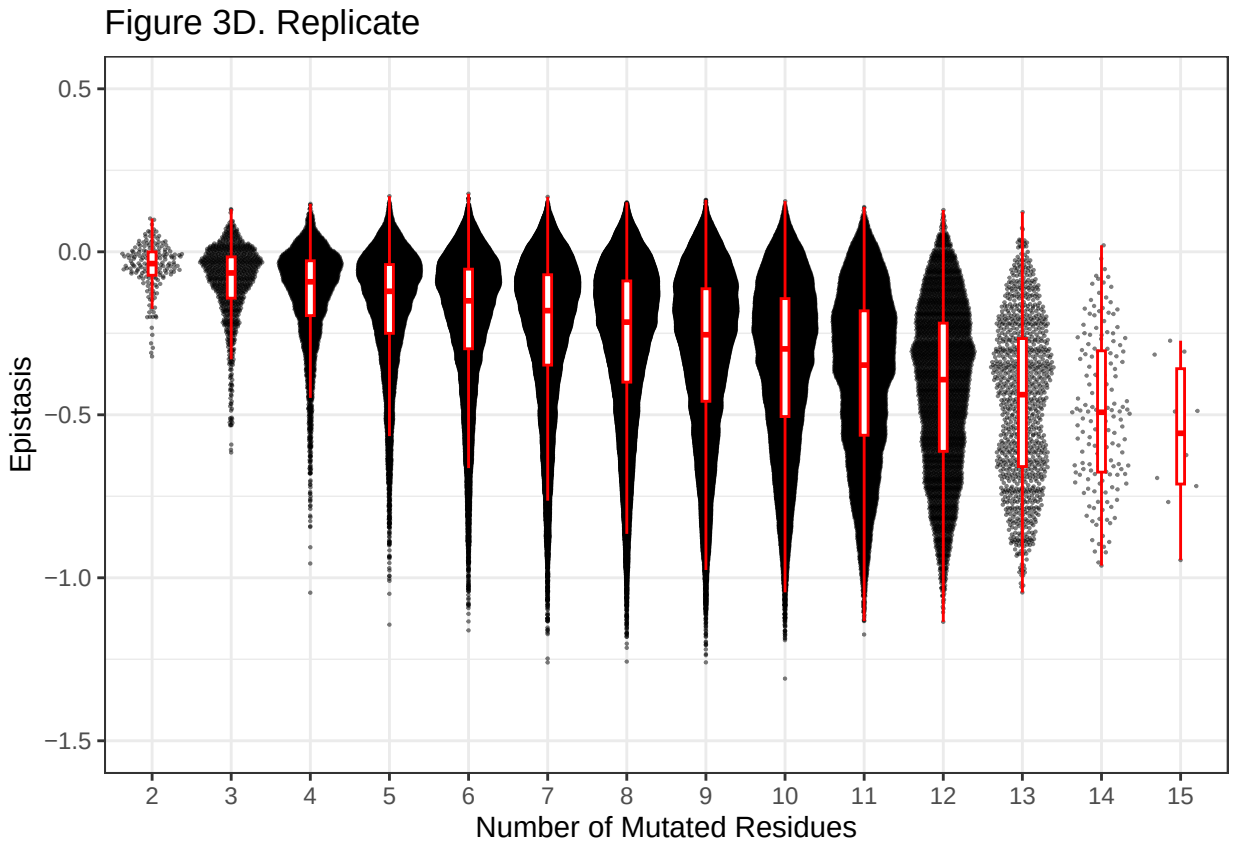
```
multiple_mutation_data <- filter(combinatorial_mutation_data, mutation_count > 1)
multiple_mutation_data$mutation_count <- factor(
    multiple_mutation_data$mutation_count
)

head(multiple_mutation_data[,c("mutants", "epistasis", "strong_epistasis")])
```

```
##                mutants   epistasis strong_epistasis
## 1 p.Arg5Asp, p.Thr7Ser  0.06326044            FALSE
## 2 p.Leu4Asn, p.Thr7Ser  0.10209432            FALSE
## 3 p.Gly3Glu, p.Thr7Ser  0.06865752            FALSE
## 4 p.Arg5Asp, p.Thr7His -0.02889073            FALSE
## 5 p.Gly3Glu, p.Arg5Asp  0.05449439            FALSE
## 6 p.Leu4Asn, p.Arg5Asp  0.03071131            FALSE
```

We then use the multiple mutation data to generate the figure.

```
fig_3d <- ggplot(multiple_mutation_data,
    aes(x = mutation_count, y = epistasis)
) +
  geom_quasirandom(width = 0.4, size = 0.1, alpha = 0.5) +
  geom_boxplot(width = 0.1, color = "red", outliers = FALSE) +
  theme_bw() +
  ylim(-1.5, 0.5) +
```

```
  labs(x = "Number of Mutated Residues", y = "Epistasis", title = "Figure 3D. Replicate") +
  theme(legend.position = "none")



ggsave("../figures/fig_3d.png", plot = fig_3d, width = 10, height = 6)  # Adjust width and height here
print(fig_3d)
```



Reproducing Figure 3E

We first need to count the proportion of mutations that are strong for a given number of mutations. We then need to reformat the data with pivot longer to generate the frequency figure. The following was used as a reference: https://sparkbyexamples.com/r-programming/group-by-count-in-r/.

Note: Requires you to run code block for Figure 3D.

```
mutation_proportion <- multiple_mutation_data %>%
  group_by(mutation_count) %>%
  summarise(
    strong_prop = sum(strong_epistasis == TRUE) / n(),
    .groups = "drop"
  )


mutation_proportion <- mutate(mutation_proportion, weak_prop = 1 - strong_prop)
mutation_proportion_long <- pivot_longer(
  mutation_proportion,
  cols=c("strong_prop", "weak_prop"),
```

```
  names_to="epistasis_type",
  values_to = "value"
)

head(mutation_proportion)
```

```
## # A tibble: 6 x 3
##   mutation_count strong_prop weak_prop
##   <fct>                <dbl>     <dbl>
## 1 2                  0           1
## 2 3                  0.00204     0.998
## 3 4                  0.0143      0.986
## 4 5                  0.0341      0.966
## 5 6                  0.0577      0.942
## 6 7                  0.0830      0.917
```

```
head(mutation_proportion_long)
```

```
## # A tibble: 6 x 3
##   mutation_count epistasis_type   value
##   <fct>          <chr>            <dbl>
## 1 2              strong_prop    0
## 2 2              weak_prop      1
## 3 3              strong_prop    0.00204
## 4 3              weak_prop      0.998
## 5 4              strong_prop    0.0143
## 6 4              weak_prop      0.986
```

```
fig_3e <- ggplot(mutation_proportion_long,
    aes(x = mutation_count, y = value, fill = epistasis_type)
  ) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.8), width = 0.7) +
  labs(x = "mutation_count", y = "value", ) +
  labs(x = "Number of Mutated Residues",
    y = "Fraction of sampled genotypes",
    title = "Figure 3E. Replicate",
    fill = "Epistasis Type"
  ) +
  scale_fill_manual(
    name = "Epistasis Type",
    values = c("strong_prop" = "#e16c5c", "weak_prop" = "#2ecddb"),
    labels = c("strong_prop" = "Strong", "weak_prop" = "Weak")
  ) +
  theme_minimal()


ggsave("../figures/fig_3e.png", plot = fig_3e, width = 10, height = 6)
print(fig_3e)
```

Figure 3E. Replicate