Goals

We have a dataset contain axon diameters of neurons from the optic nerve of control and mutant zebrafish. We'd like to know if the mean axon diameter, or the distribution of axon diameters, differs between groups. We want to implement tests that take account of within and between animal variance.

Here, we focus on the myRF data set.

Load and format data

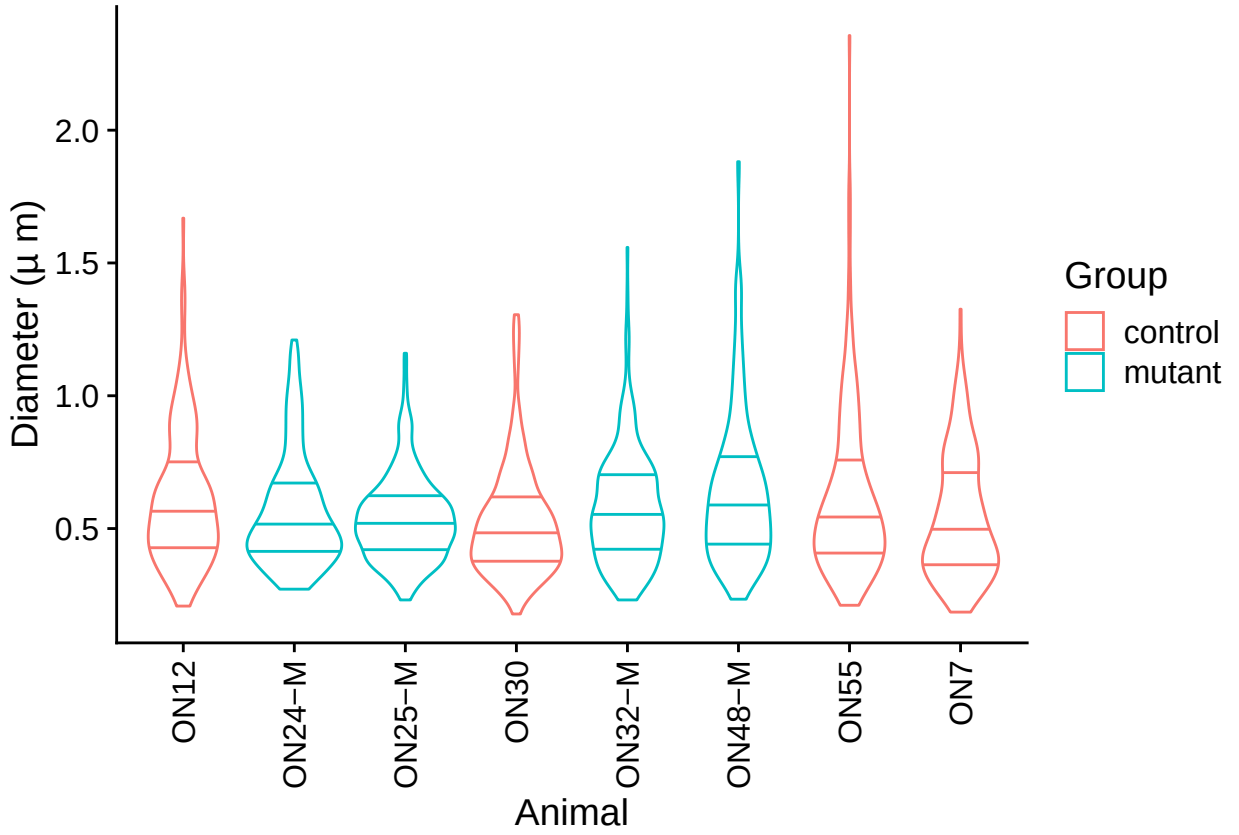
```
df <- read_excel("MyRF Axon Diameter Measurements 20240709.xlsx", range = "A3:H328") %>%
    pivot_longer(cols = 1:8) %>%
    drop_na() %>%
    mutate(group = as_factor(ifelse(name %in% c("ON7", "ON12", "ON30", "ON55"), "control", "mutant")),
           litter = as_factor(ifelse(name %in% c("ON7", "ON12"), "litter1",
                                     ifelse(name %in% c("ON24-M", "ON25-M", "ON30"), "litter2",
                                                ifelse(name %in% c("ON32-M"), "litter3",
                                                ifelse(name %in% c("ON48-M"), "litter4",
                                                ifelse(name %in% c("ON55"), "litter5", "error"))))),
           processed = as_factor(ifelse(litter %in% c("litter1", "litter2"),"processed1", "processed2"))

df$group <- as.factor(df$group)
df$name <- as.factor(df$name)
```

Plot the data

Focus here on plot of individual mice, colour coded by group.

```
(plot_by_id <- ggplot(data = df, aes(name, value)) +
     geom_violin(aes(colour = group), draw_quantiles = c(0.25, 0.5, 0.75)) +
     theme_cowplot(font_size = 14) +
     theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
    labs(x = 'Animal', y = 'Diameter (\u00B5 m)', colour = "Group"))
```




```
ggsave('Plots/violins.jpeg', plot_by_id)
```

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

Tests for differences in means

```
mm_t <- lmer(log(value) ~ group + (1 | name), data = df)
mm_t_null <- lmer(log(value) ~ (1 | name), data = df)
summary(mm_t)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: log(value) ~ group + (1 | name)
##    Data: df
##
## REML criterion at convergence: 1997.2
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.6621 -0.7026 -0.0217  0.6423  3.8142
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  name     (Intercept) 0.003713 0.06094
##  Residual             0.145934 0.38201
## Number of obs: 2160, groups:  name, 8
##
```

```
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept) -0.64693    0.03271 -19.777
## groupmutant  0.03399    0.04615   0.737
##
## Correlation of Fixed Effects:
##             (Intr)
## groupmutant -0.709
```

```
anova(mm_t, mm_t_null)
```

```
## refitting model(s) with ML (instead of REML)
```

```
## Data: df
## Models:
## mm_t_null: log(value) ~ (1 | name)
## mm_t: log(value) ~ group + (1 | name)
##           npar    AIC    BIC  logLik deviance  Chisq Df Pr(>Chisq)
## mm_t_null    3 1993.5 2010.6 -993.77   1987.5
## mm_t         4 1994.8 2017.6 -993.42   1986.8 0.6906  1      0.406
```

Test again with more complicated random effects stuctures

```
mm_t_2 <- lmer(log(value) ~ group + (1 | litter/name), data = df)
mm_t_null_2 <- lmer(log(value) ~ (1 | litter/name), data = df)
summary(mm_t_2)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: log(value) ~ group + (1 | litter/name)
##    Data: df
##
## REML criterion at convergence: 1996.8
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.6577 -0.7005 -0.0188  0.6393  3.8127
##
## Random effects:
##  Groups      Name        Variance Std.Dev.
##  name:litter (Intercept) 0.002358 0.04856
##  litter      (Intercept) 0.001445 0.03802
##  Residual                0.145938 0.38202
## Number of obs: 2160, groups:  name:litter, 8; litter, 5
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept) -0.64303    0.03472 -18.522
## groupmutant  0.04355    0.04461   0.976
##
## Correlation of Fixed Effects:
##             (Intr)
## groupmutant -0.654
```

```
anova(mm_t_2, mm_t_null_2)
```

```
## refitting model(s) with ML (instead of REML)
```

```
## Data: df
## Models:
## mm_t_null_2: log(value) ~ (1 | litter/name)
## mm_t_2: log(value) ~ group + (1 | litter/name)
##             npar    AIC    BIC  logLik deviance  Chisq Df Pr(>Chisq)
## mm_t_null_2    4 1995.5 2018.2 -993.77   1987.5
## mm_t_2         5 1996.6 2025.0 -993.31   1986.6 0.9227  1     0.3368
```

```
mm_t_3 <- lmer(log(value) ~ group + (1 | processed/name), data = df)
mm_t_null_3 <- lmer(log(value) ~ (1 | processed/name), data = df)
summary(mm_t_3)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: log(value) ~ group + (1 | processed/name)
##    Data: df
##
## REML criterion at convergence: 1996.5
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.6643 -0.7015 -0.0129  0.6368  3.8046
##
## Random effects:
##  Groups         Name        Variance Std.Dev.
##  name:processed (Intercept) 0.002588 0.05087
##  processed      (Intercept) 0.001885 0.04342
##  Residual                   0.145936 0.38202
## Number of obs: 2160, groups:  name:processed, 8; processed, 2
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept) -0.63456    0.04246 -14.947
## groupmutant  0.02162    0.04047   0.534
##
## Correlation of Fixed Effects:
##             (Intr)
## groupmutant -0.500
```

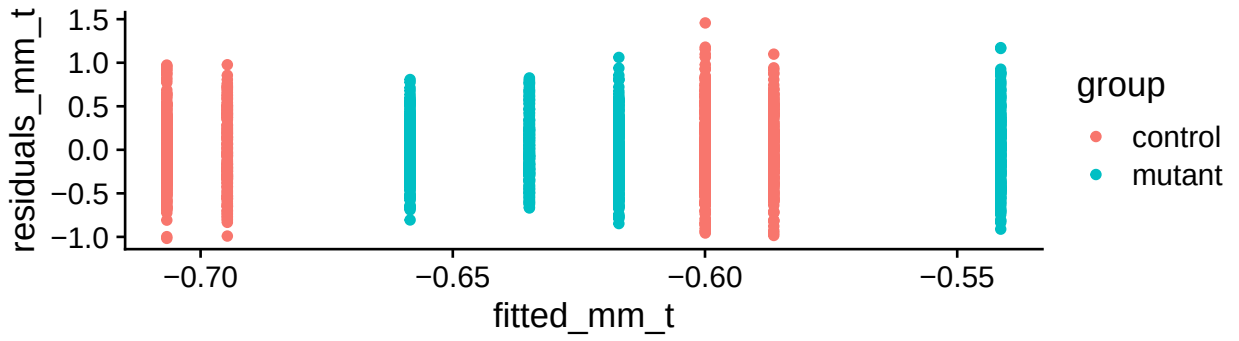
```
anova(mm_t_2, mm_t_null_3)
```

```
## refitting model(s) with ML (instead of REML)
```

```
## Data: df
## Models:
## mm_t_null_3: log(value) ~ (1 | processed/name)
## mm_t_2: log(value) ~ group + (1 | litter/name)
##             npar    AIC    BIC  logLik deviance Chisq Df Pr(>Chisq)
## mm_t_null_3    4 1995.1 2017.8 -993.54   1987.1
## mm_t_2         5 1996.6 2025.0 -993.31   1986.6 0.474  1     0.4912
```

Evaluate residuals

```
df$residuals_mm_t <- resid(mm_t)
df$fitted_mm_t <- fitted(mm_t)
(res_t_plot <- ggplot(data = df, aes(x = fitted_mm_t, y = residuals_mm_t)) +
    geom_point(aes(colour = group)) +
    theme_cowplot())
```



Compare residual distributions between groups, for the untransformed data first and then for the log-transformed data.

```
### We could use a KS test:
ks.test(residuals_mm_t ~ group, data = df)
```

```
## Warning in ks.test.default(x = DATA[[1L]], y = DATA[[2L]], ...): p-value will be
## approximate in the presence of ties
```

```
##
##  Asymptotic two-sample Kolmogorov-Smirnov test
##
## data:  residuals_mm_t by group
## D = 0.071709, p-value = 0.007785
## alternative hypothesis: two-sided
```

This suggests differences between groups in the variance, but doesn't account for within vs between subject effects in the residuals. But worry here is that we don't treat animals as indepwndent. Instead calculate SD for each animal and compare groups:

```
gr_t <- df %>% group_by(group, name) %>% summarize(avg_r = sd(residuals_mm_t))
```

```
## `summarise()` has grouped output by 'group'. You can override using the `.groups`
## argument.
```

```
t.test(avg_r ~ group, data = gr_t)
```

```
##
##  Welch Two Sample t-test
##
## data:  avg_r by group
## t = 1.9909, df = 5.4795, p-value = 0.09813
## alternative hypothesis: true difference in means between group control and group mutant is not equal
## 95 percent confidence interval:
##  -0.01468272  0.12853729
## sample estimates:
## mean in group control  mean in group mutant
```

```
##              0.4089329              0.3520056
```

Hard to interpret because group sizes are small. No firm evidence for difference in distributions but wouldn't rule it out either.

Generate histograms for all animals

```
### Make histograms for each animal
### Use log transformed data
names = unique(df$name)
### If submean = 1 then will substract means before making histograms
histfun <- function(name, df, submean = 0) {
    sub <- ifelse(submean == 1, mean(df$value[df$name==name]), 0)
    hist(log(df$value[df$name==name]-sub), seq(-2,2,0.1), plot = FALSE)
}
hists <- sapply(names, histfun, df, submean=0)

### Convert results to tibble for use with tidyverse functions
rns <- rownames(hists)
hists_tib <- as_tibble(hists) %>%
    rownames_to_column(var = "rowname") %>%
    pivot_longer(-rowname, names_to = "column", values_to = "value") %>%
    pivot_wider(names_from = rowname, values_from = value)
```

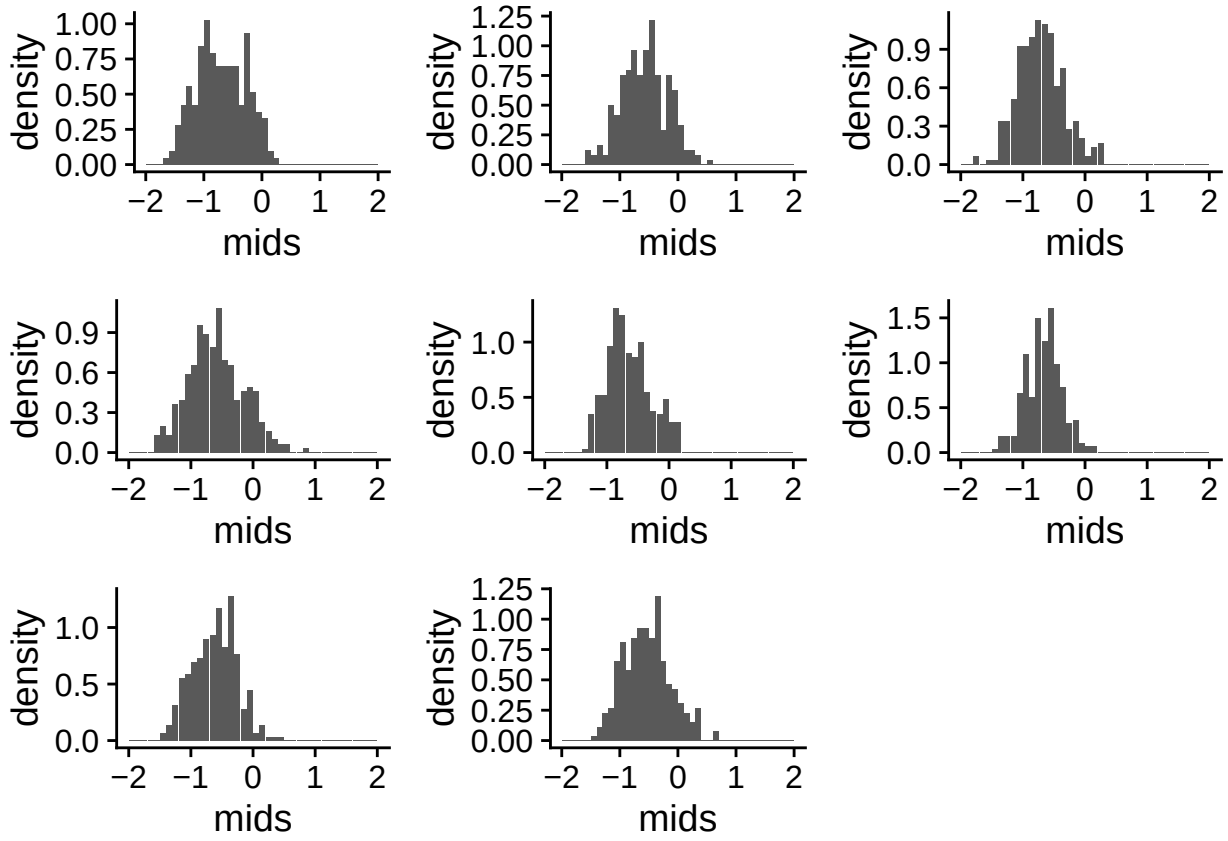
```
## Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if `.name_repair`
## is omitted as of tibble 2.0.0.
## i Using compatibility `.name_repair`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

```
colnames(hists_tib) <- c("animal", rns)


ggconvfun <- function(mids, density) {
    gghist <- cbind(mids, density)
    ggplot(gghist, aes(mids, density)) +
        geom_col() +
        theme_cowplot()
}

gghists <- map2(hists_tib$mids, hists_tib$density, ggconvfun)

plot_grid(plotlist = gghists)
```



Evaluate distributions

```
### Make histograms compatible with comparison tools
hists_as_matrix <- function(counts, mids) {matrix(c(counts/sum(counts), mids), ncol=2)}

hists_tib <- hists_tib %>%
    mutate(mat_2 = map2(density, mids, hists_as_matrix),
           mat = map2(counts, mids, hists_as_matrix))

### List all combinations of histograms
combinations <- combn(1:length(names), 2, simplify = FALSE)

### Calculate KL divergence for all combinations
kl_on_comb <- function(comb, mat) {
    P <- mat[[comb[[1]]]][,1] / sum(mat[[comb[[1]]]][,1])
    Q <- mat[[comb[[2]]]][,1] / sum(mat[[comb[[2]]]][,1])
### check order so that control is left in x
### if(df$group[[comb[[1]]]]=="mutant") {x <- rbind(Q,P)} else {x <- rbind(P,Q)}
x <- rbind(P,Q)
### KL(x)
    distance(x, "taneja") #pearson(asymmetric), divergence (symmetric), clark(symmetric), taneja (symme
}

kls <- lapply(combinations, kl_on_comb, hists_tib$mat)
```

```
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
## Metric: 'taneja' using unit: 'log'; comparing: 2 vectors.
```

```r
### Put everything together for analysis
collected <- tibble(pairs = combinations, kl  = unlist(kls))

### Add columns to identify animals
a1_func <- function(pair, names, pos) {
    names[pair[[pos]][1]]
    }
collected$pair1 <- sapply(collected$pairs, a1_func, names, 1)
collected$pair2 <- sapply(collected$pairs, a1_func, names, 2)

### Reorganise
get_group <- function(name, df) {as.character(df$group[[grep(name, df$name)[[1]]]])}
grouplist <- lapply(names, get_group, df)

group_lookup <- function(id, names, grouplist) {grouplist[[grep(id, names)]]}
compare_groups <- function(g1, g2) {g1 == g2}

collected <- collected %>%
    mutate(group1 = map_chr(pair1, group_lookup, names, grouplist),
           group2 = map_chr(pair2, group_lookup, names, grouplist),
           samegroup = map2_lgl(group1, group2, compare_groups))

### Compare KL divergece for pairs that are from the same group with pairs from different groups
wilcox.test(kl ~ samegroup, data = collected)
```

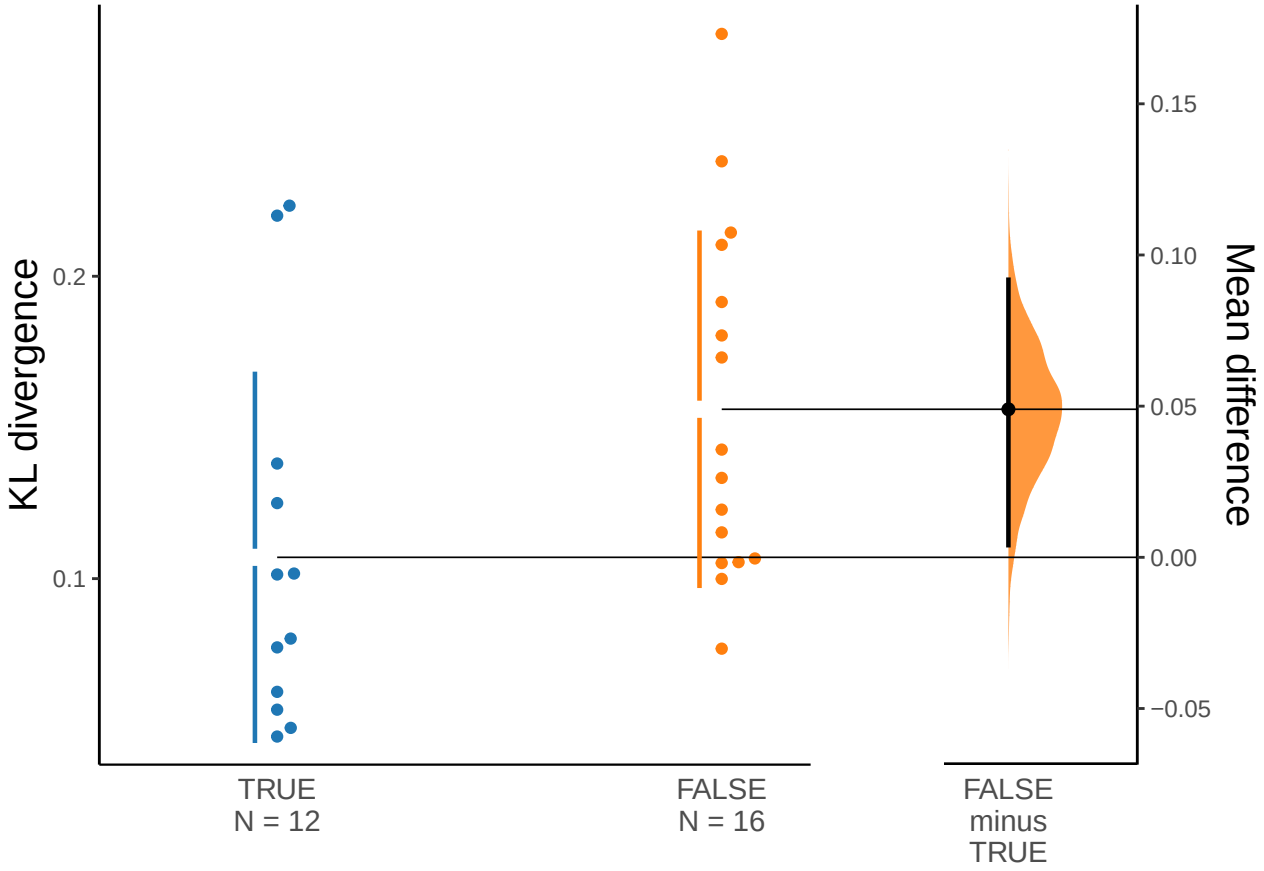
```
##
```

```
##  Wilcoxon rank sum exact test
##
## data:  kl by samegroup
## W = 143, p-value = 0.02918
## alternative hypothesis: true location shift is not equal to 0
```

Make dabest plots for the above compariosns. Use dabest_plot.

```
dabest_obj_kl.mean_diff <- load(
  data = collected,
  x = samegroup,
  y = kl,
  idx = c("TRUE", "FALSE")
) %>%
  mean_diff()

(de_kl <- dabest_plot(dabest_obj_kl.mean_diff, TRUE,
                      swarm_label = 'KL divergence'))
```



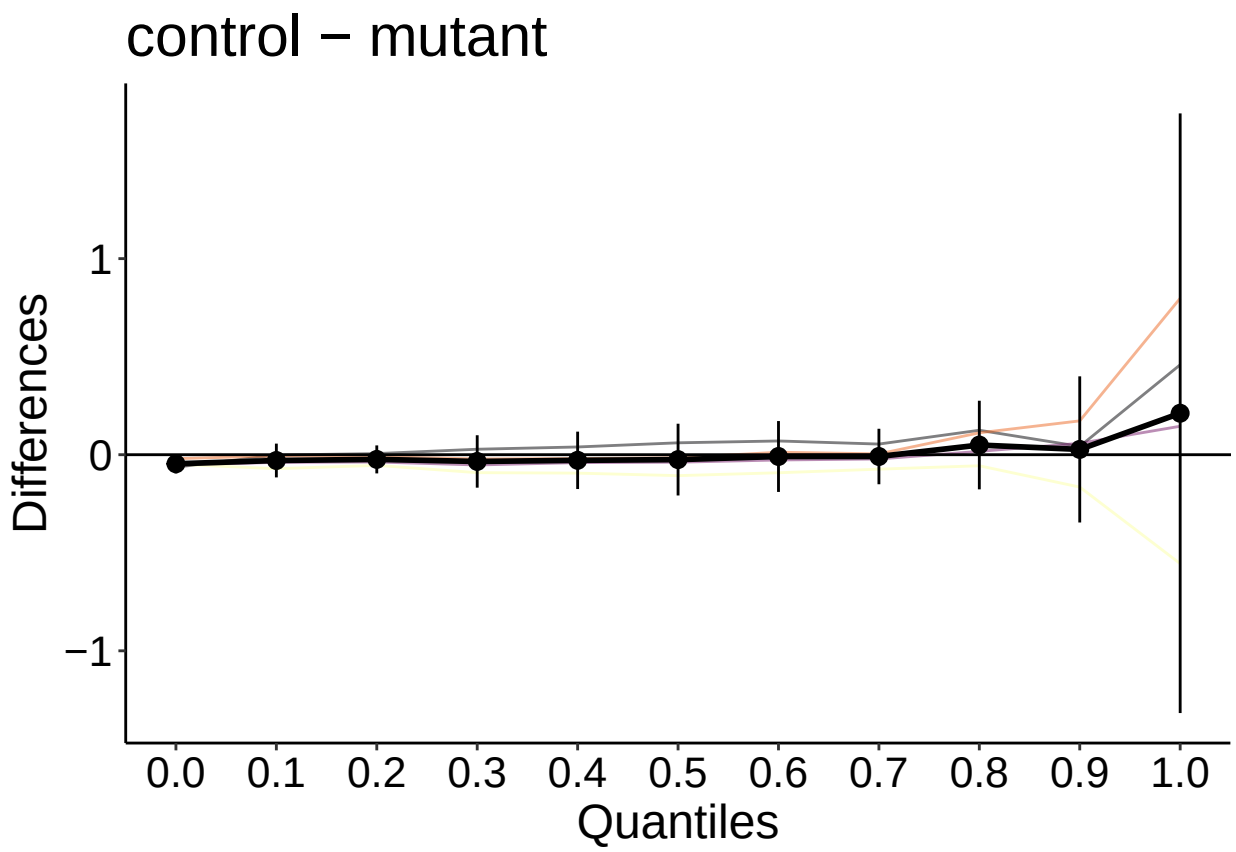

```
ggsave('Plots/KL_dabest.jpeg', de_kl)
```

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

Another approach is to compare the distributions using a hierarchical shift function. See: https://github.com/GRousselet/rogme/blob/master/docs/hsf.md

```
out <- hsf(df, log(value) ~ group + name, qseq = c(seq(0, 1, 0.1)))

plot_hsf(out)
```




```
out$adjusted_pvalues
```

```
##  [1] 0.6377587 0.8834981 0.8834981 0.8834981 0.8834981 0.8834981 0.8834981 0.8834981
##  [9] 0.8834981 0.8834981 0.8834981
```

Again, data size too small to be conclusive.