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.

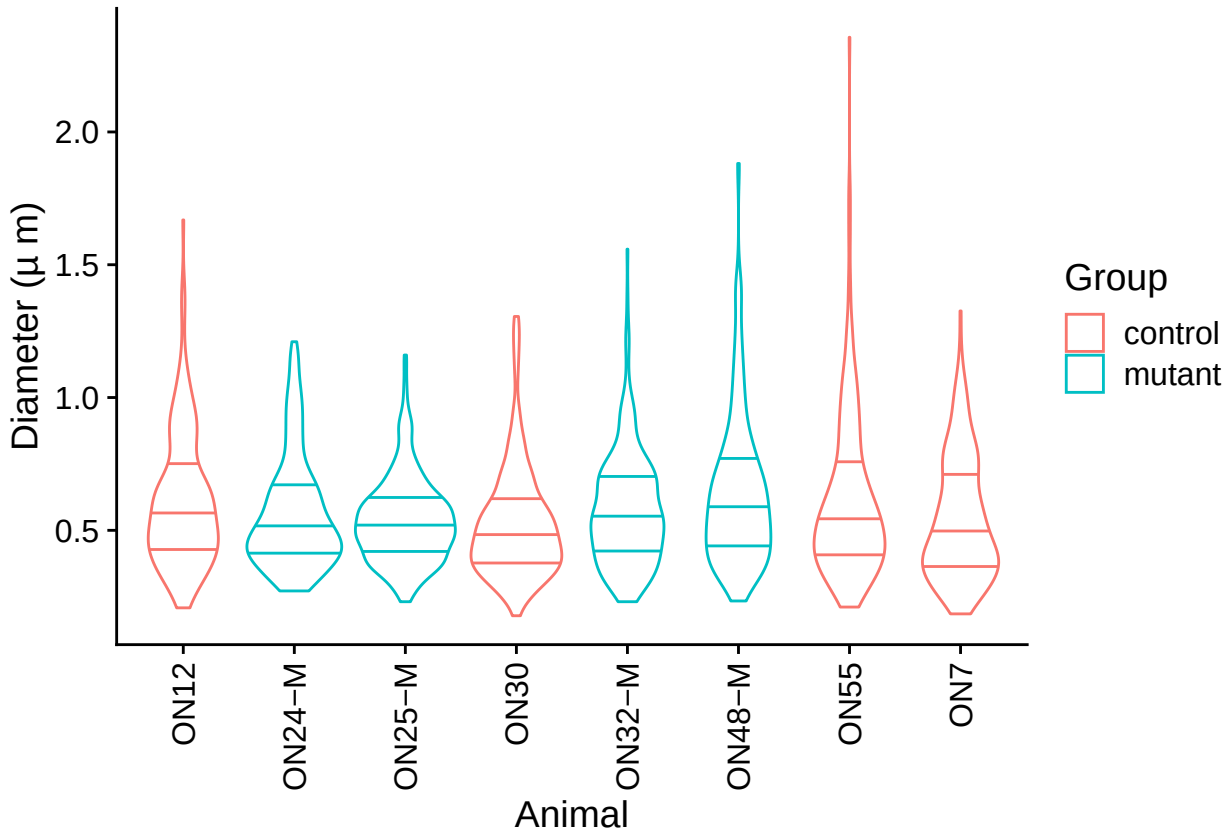
Load and format data

```
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")))
```

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)
```

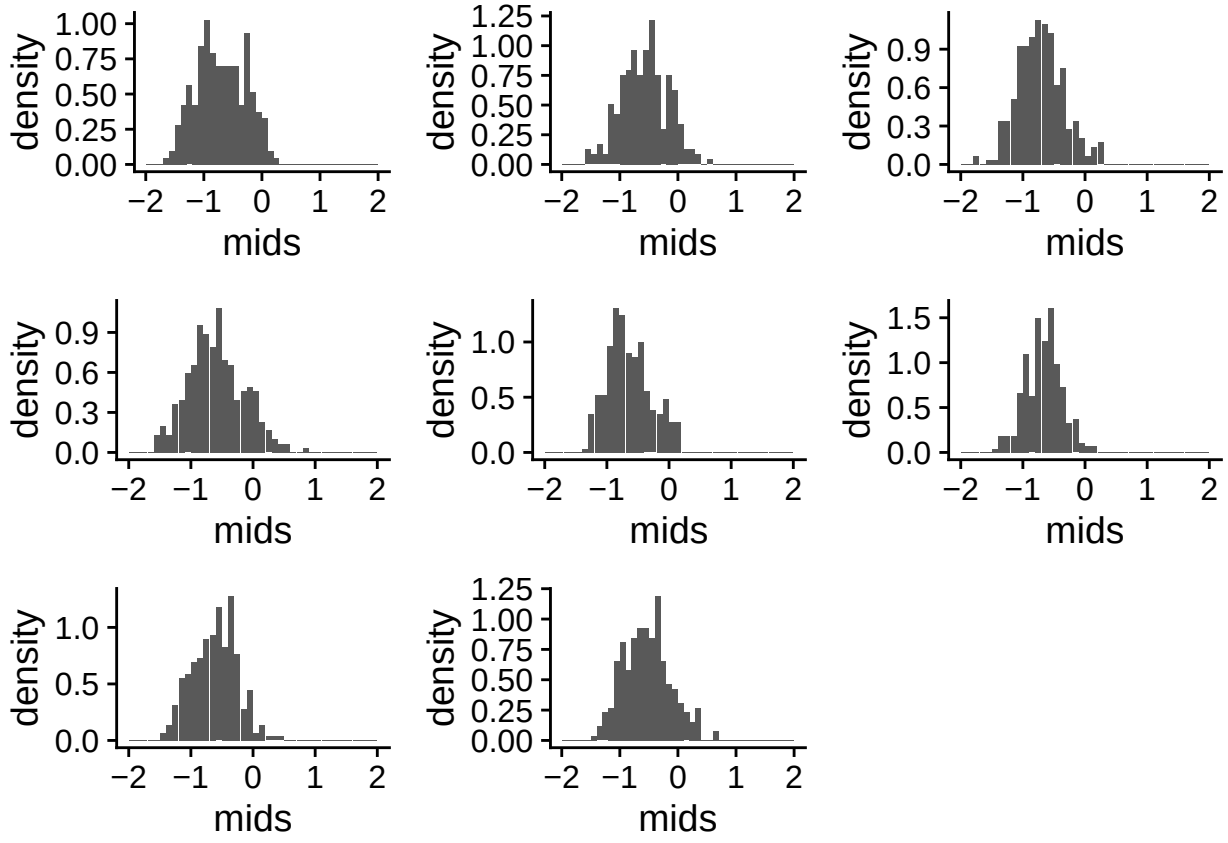
```
## refitting model(s) with ML (instead of REML)
```

```
## 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
```

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)
```

```r
    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)
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(density, mids) {matrix(c(density, mids), ncol=2)}

hists_tib <- hists_tib %>%
    mutate(mat = map2(density, 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])
    x <- rbind(P,Q)
    KL(x)
}

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

```
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
## Metric: 'kullback-leibler' using unit: 'log2'; comparing: 2 vectors.
```

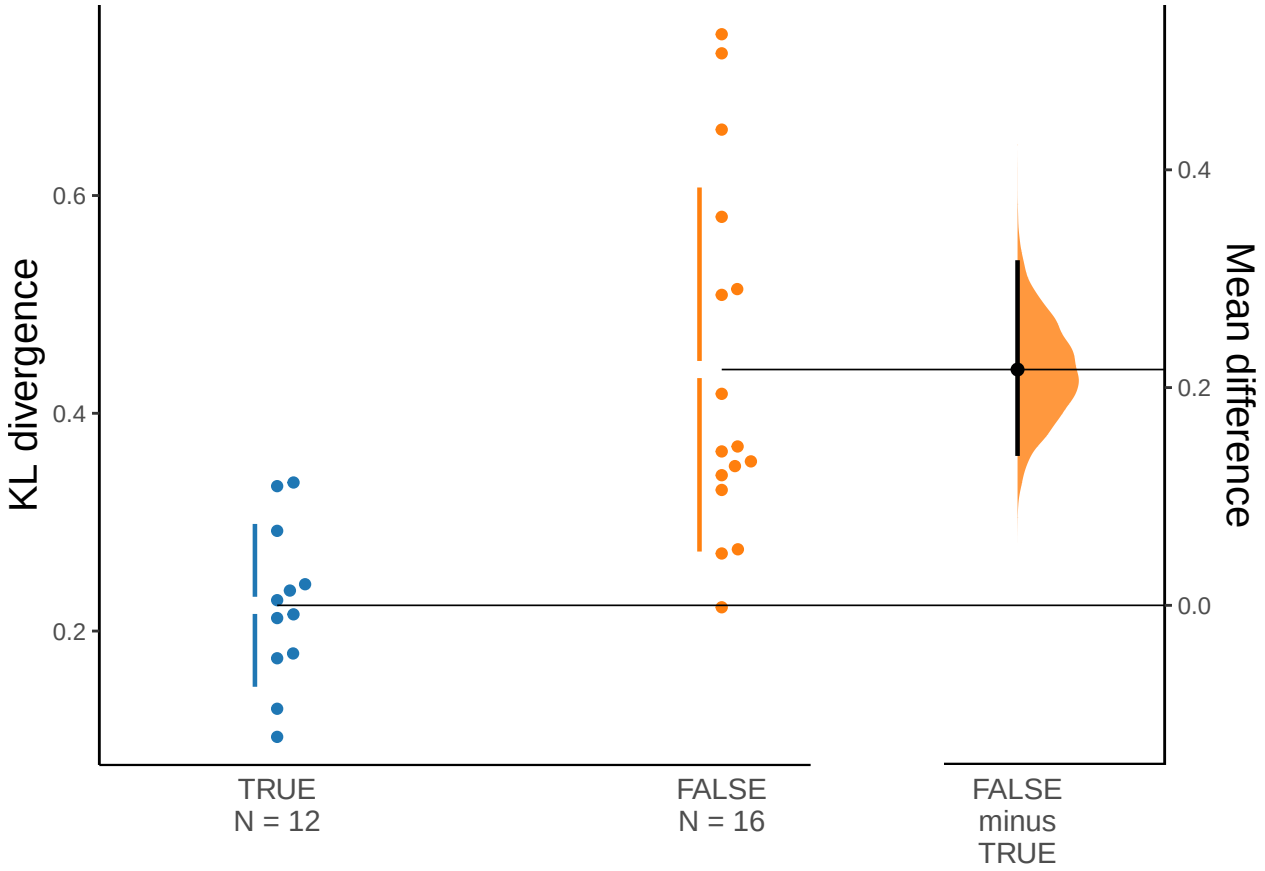
```
### Put everything together for analysis
collected <- tibble(pairs = combinations, kl  = unlist(kls))

### 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 = 178, p-value = 3.32e-05
## 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
```