Code for Mussel Freeze Thaw

Lauren Gill

16/12/2021

Contents

Libraries . . . 1
Survival . . . 2
Freeze Treatment Comparisons . . . 2
Cumulative freezing . . . 4
Basal Levels of HSP70 . . . 5
Top Band . . . 5
All bands . . . 6
Bottom Bands . . . 7
HSP70 vs position (High intertidal/Low intertidal) and recovery time (2 or 20 hours) . . . 10
Top Bands . . . 10
All bands . . . 11
Bottom bands . . . 12
Top bands . . . 12
Graphing the data for freeze time vs position . . . 13
Individual Freezes . . . 19
Ubiquitin . . . 20
Seeing if ubiquitin levels are affected by shore position or recovery time . . . 20
Plotting Ubiquitin vs recovery/position graphs . . . 23
Basal expression of Ubiquitin . . . 26
Correlation between HSP and ubiqutuin . . . 27
Testing for correlation . . . 28
Plotting correlation graphs . . . 29

Libraries

```
library(MASS)
library(plyr)
library(dplyr)
library(ggplot2)
library(ggpubr)
library(here)
library(tidyverse)
library(cowplot)
library(car)
library(multcomp)
```

```
## Warning: package 'multcomp' was built under R version 4.1.2
```

Survival

Freeze Treatment Comparisons

Reading in Data and filtering for total freeze time to be equal to 8 hours

```
survival <- read.csv(here("Data", "survival.csv"))
only.8 <- survival %>%
    filter(Total.freeze.time == 8)
```

Now running a chisq test on the data, using time frozen, acclimation time and position as variables

```
only.8$Time_Frozen_hrs <- as.factor(only.8$Time_Frozen_hrs)
only.8.glm <- glm(Survival ~ Time_Frozen_hrs * Acclimation_time_days *
    Position, family = "binomial", data = only.8)

only.8.glm.reduced <- glm(Survival ~ Time_Frozen_hrs, family = "binomial",
    data = only.8)
anova(only.8.glm, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: Survival
##
## Terms added sequentially (first to last)
##
##
##                                            Df Deviance Resid. Df Resid. Dev
## NULL                                                          71     98.420
## Time_Frozen_hrs                             2  24.0163        69     74.404
## Acclimation_time_days                       0   0.0000        69     74.404
## Position                                    1   2.0742        68     72.329
## Time_Frozen_hrs:Acclimation_time_days       0   0.0000        68     72.329
## Time_Frozen_hrs:Position                    2   9.5588        66     62.771
## Acclimation_time_days:Position              0   0.0000        66     62.771
## Time_Frozen_hrs:Acclimation_time_days:Position  0   0.0000        66     62.771
```

```
##                                                 Pr(>Chi)
## NULL
## Time_Frozen_hrs                               6.094e-06 ***
## Acclimation_time_days
## Position                                       0.149813
## Time_Frozen_hrs:Acclimation_time_days
## Time_Frozen_hrs:Position                       0.008401 **
## Acclimation_time_days:Position
## Time_Frozen_hrs:Acclimation_time_days:Position
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(glht(only.8.glm.reduced, mcp(Time_Frozen_hrs = "Tukey")))
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Tukey Contrasts
##
##
## Fit: glm(formula = Survival ~ Time_Frozen_hrs, family = "binomial",
##     data = only.8)
##
## Linear Hypotheses:
##            Estimate Std. Error z value Pr(>|z|)
## 4 - 2 == 0  -1.4351     0.7475  -1.920   0.1320
## 8 - 2 == 0  -3.2809     0.7960  -4.122   <0.001 ***
## 8 - 4 == 0  -1.8458     0.6561  -2.814   0.0135 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

Now seeing if length affects survival

```
only.8.glm.length <- glm(Survival ~ Length_mm, data = only.8)
summary(aov(only.8.glm.length))
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## Length_mm    1   0.84  0.8404   3.494  0.066 .
## Residuals   67  16.12  0.2405
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 3 observations deleted due to missingness
```

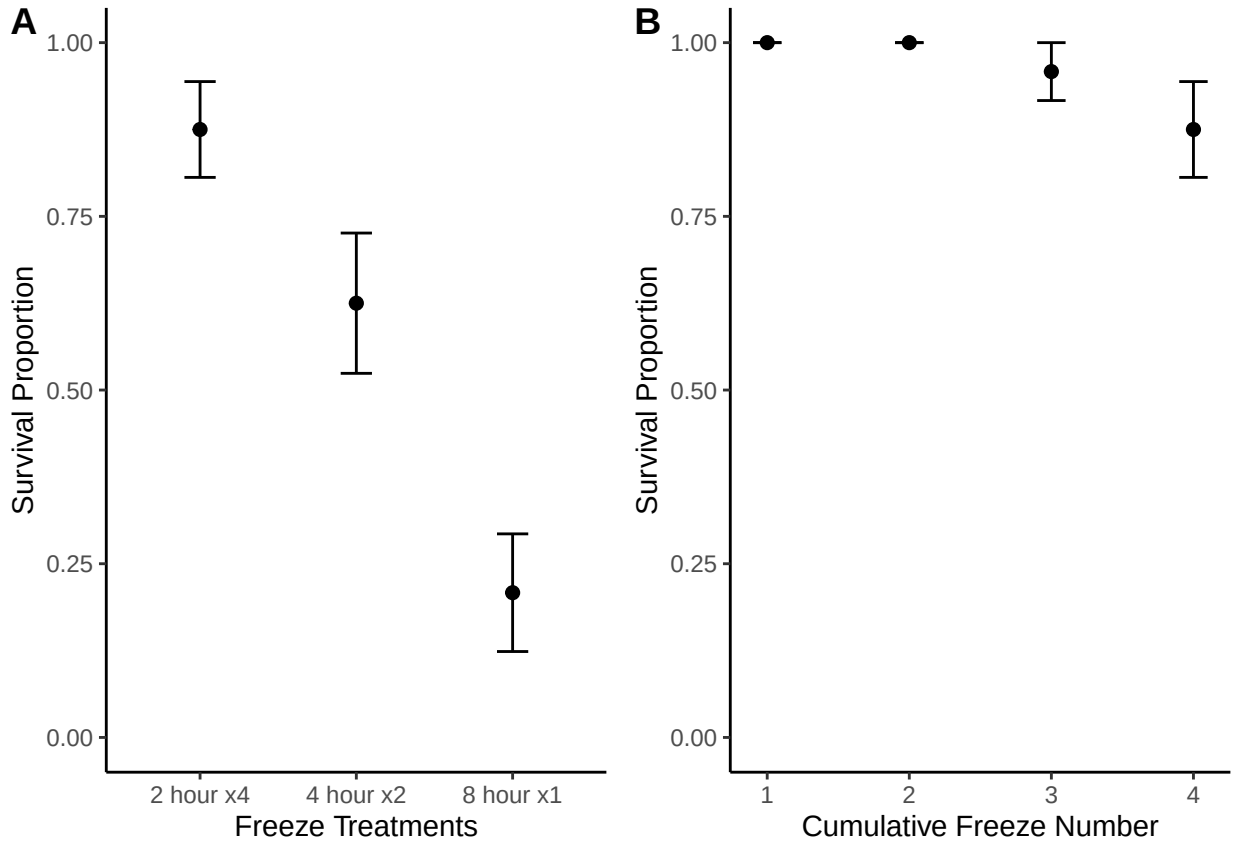
```
only.8.group <- group_by(only.8, Time_Frozen_hrs)
only.8.sum <- summarise(only.8.group, prop_surv = mean(Survival,
    na.rm = TRUE), sd = sd(Survival, na.rm = TRUE), total = n(),
    SE = sd(Survival)/sqrt(n()))

only.8.sum$Time_Frozen_hrs <- as.factor(only.8.sum$Time_Frozen_hrs)

survivalgraph <- ggplot(only.8.sum, aes(x = Time_Frozen_hrs,
```

```
    y = prop_surv)) + geom_point() + ylim(0, 1) + xlab("Freeze Treatments") +
    ylab("Survival Proportion") + geom_errorbar(aes(ymin = prop_surv -
    SE, ymax = prop_surv + SE, width = 0.2)) + theme_classic() +
    geom_point(size = 2) + scale_x_discrete(labels = c("2 hour x4",
    "4 hour x2", "8 hour x1"))
```

Cumulative freezing

```
repeat2 <- subset(survival, survival$Time_Frozen_hrs == 2)
repeat2sum <- group_by(repeat2, Culmulative_Freeze_Number)
repeat2sum1 <- summarise(repeat2sum, prop_surv = mean(Survival,
    na.rm = TRUE), sd = sd(Survival, na.rm = TRUE), total = n(),
    SE = sd(Survival)/sqrt(n()))
repeatedfreeze <- ggplot(repeat2sum1, aes(x = Culmulative_Freeze_Number,
    y = prop_surv)) + geom_point() + ylim(0, 1) + xlab("Cumulative Freeze Number") +
    ylab("Survival Proportion") + geom_errorbar(aes(ymin = prop_surv -
    SE, ymax = prop_surv + SE, width = 0.2)) + theme_classic() +
    geom_point(size = 2)
```

```
## combining cumulative and 8 hour total freezes
combinedsurival <- plot_grid(survivalgraph, repeatedfreeze, labels = c("A",
    "B"))
combinedsurival
```



```
# ggsave('survivalgraph', device = 'tiff', dpi = 300, width
# = 174, height = 100, units = 'mm')
```

Basal Levels of HSP70

Top Band

Reading in Data

```
top_bands <- read.csv(here("Data", "Top band.csv"))
```

For Basal levels we only want to use the control mussels, because these are the mussels that were not frozen, and therefore will show us how much HSP70 they have naturally, so subset the controls and then take the mean of relative density

```
controlonlytop <- subset(top_bands, top_bands$treatment == "C")
controldenstop <- controlonlytop %>%
    group_by(position, time.frozen, sam.num) %>%
    summarise(meanden = mean(rel.dens))
```

Now running a t.test on the control mussels to see if position makes a difference

```
shapiro.test(controldenstop$meanden)  #p = 0.08, its normal
```

```
##
##  Shapiro-Wilk normality test
##
## data:  controldenstop$meanden
## W = 0.92942, p-value = 0.08429
```

```
bartlett.test(meanden ~ position, data = controldenstop)  # p-value = 0.09448, Homogeneity of variances
```

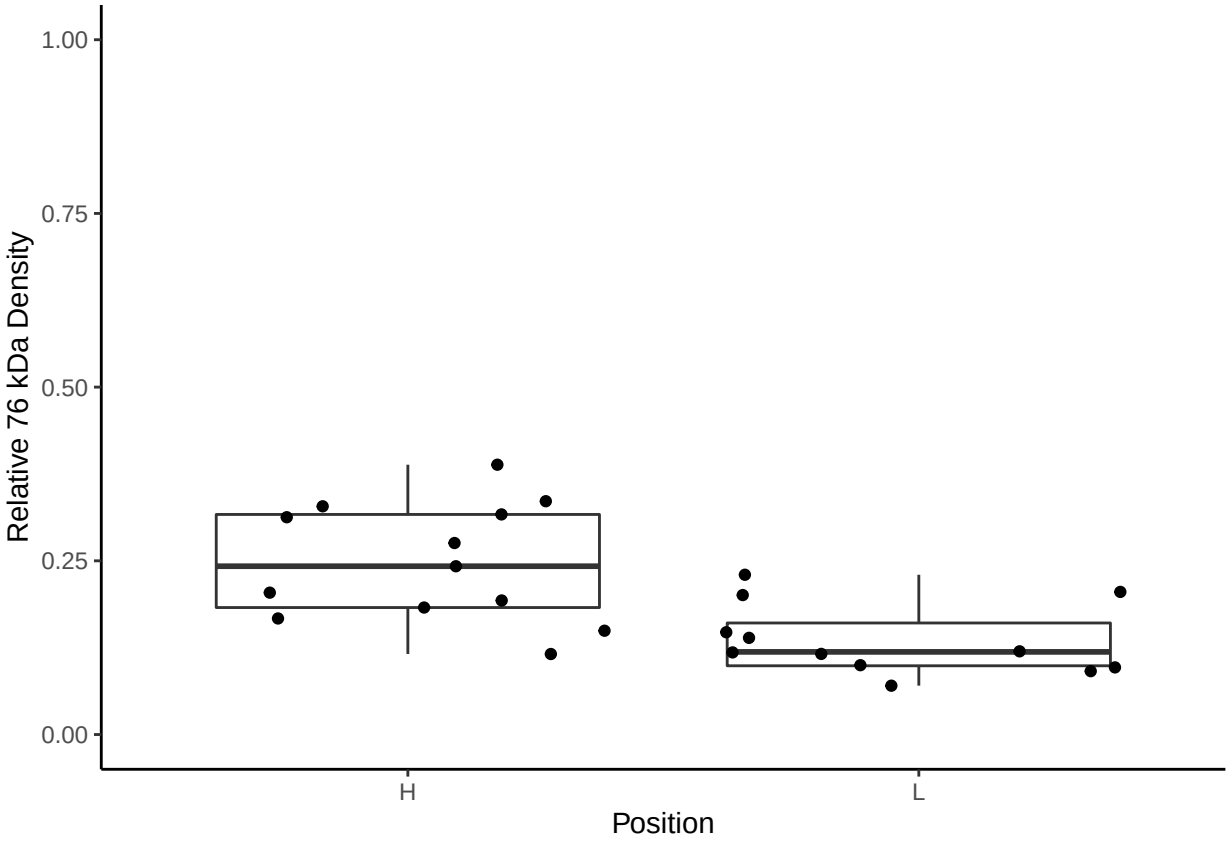
```
##
##  Bartlett test of homogeneity of variances
##
## data:  meanden by position
## Bartlett's K-squared = 2.7963, df = 1, p-value = 0.09448
```

```
t.test(meanden ~ position, var.equal = FALSE, data = controldenstop)
```

```
##
##  Welch Two Sample t-test
##
## data:  meanden by position
## t = 4.0035, df = 19.798, p-value = 0.0007095
## alternative hypothesis: true difference in means between group H and group L is not equal to 0
## 95 percent confidence interval:
##  0.05309006 0.16875285
## sample estimates:
## mean in group H mean in group L
##       0.2470952       0.1361737
```

Now making a box plot of basal levels

```
controlboxtop <- ggplot(controldenstop, aes(x = position, y = meanden)) +
    geom_boxplot() + ylim(0, 1) + xlab("Position") + ylab("Relative 76 kDa Density") +
    theme_classic() + geom_jitter()
controlboxtop
```



All bands

Below are the same steps as with the top bands

```
all_bands <- read.csv(here("Data", "All bands.csv"))
controlonlyall <- subset(all_bands, all_bands$treatment == "C")
controldensall <- controlonlyall %>%
    group_by(position, freeze.time, sam.num) %>%
    summarise(meanden = mean(rel.dens))
```

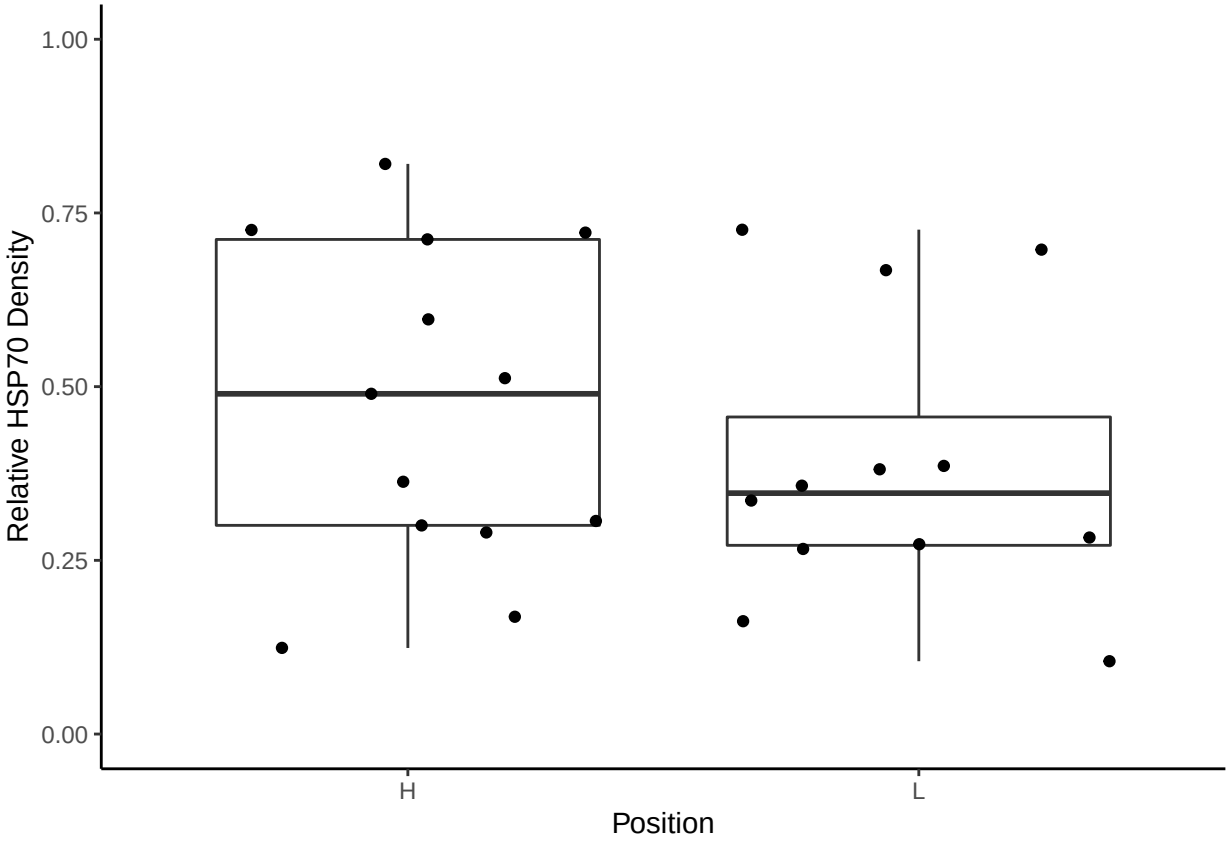
```
shapiro.test(controldensall$meanden)  #p-value = 0.04557, not normal
```

```
##
##  Shapiro-Wilk normality test
##
## data:  controldensall$meanden
## W = 0.91776, p-value = 0.04557
```

```
## Using man whitney U test
wilcox.test(meanden ~ position, data = controldensall)
```

```
##
##  Wilcoxon rank sum exact test
##
## data:  meanden by position
## W = 97, p-value = 0.3203
## alternative hypothesis: true location shift is not equal to 0
```

```
# Now making a box plot of basal levels - NO difference
controlbox <- ggplot(controldensall, aes(x = position, y = meanden)) +
    geom_boxplot() + ylim(0, 1) + xlab("Position") + ylab("Relative HSP70 Density") +
    theme_classic() + geom_jitter()
controlbox
```



Bottom Bands

```
bottom_bands <- read.csv(here("Data", "Bottom bands.csv"))

controlonlybottom <- subset(bottom_bands, bottom_bands$treatment ==
    "C")
controldensbot <- controlonlybottom %>%
```

```
    group_by(shore.position, freeze.time, sample.num) %>%
    summarise(meanden = mean(rel.dens))
```

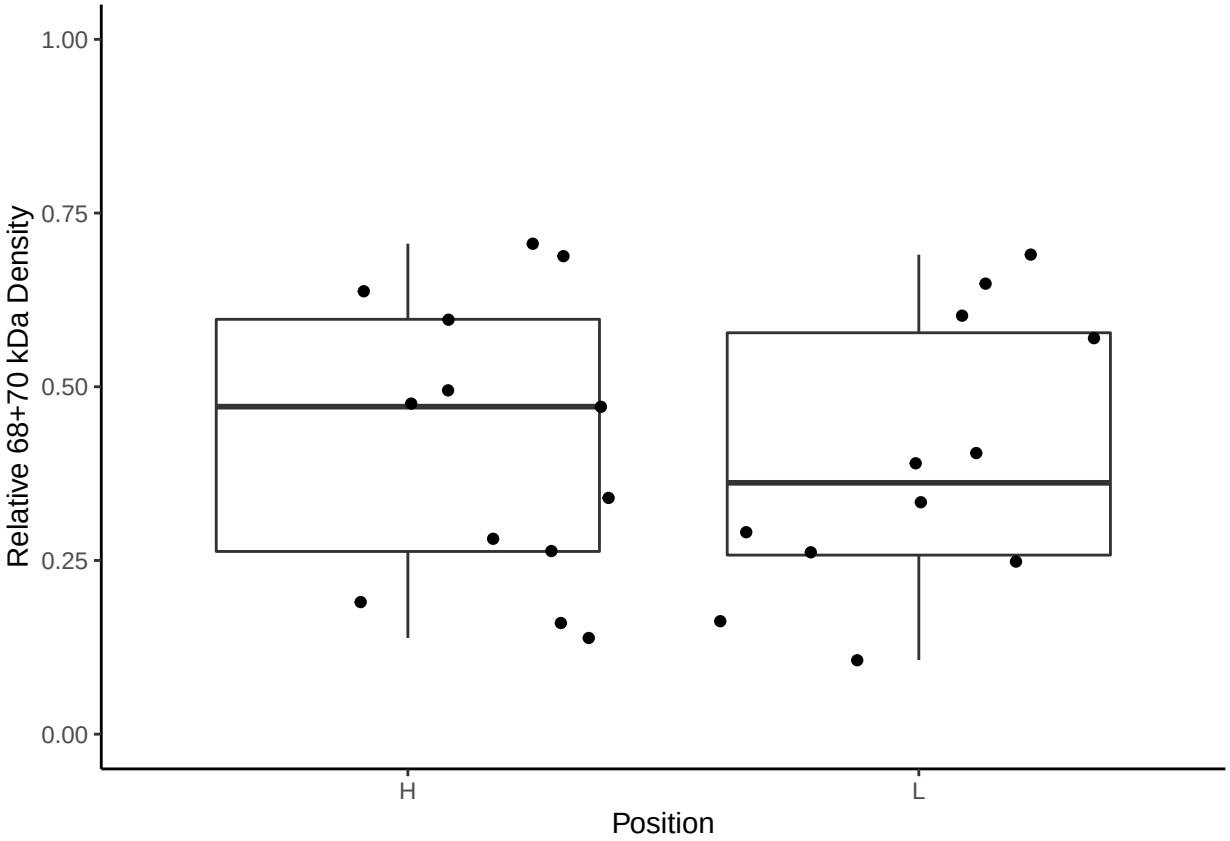
```
shapiro.test(controldensbot$meanden)  #p-value = 0.09156
```

```
##
##  Shapiro-Wilk normality test
##
## data:  controldensbot$meanden
## W = 0.93098, p-value = 0.09156
```

```
t.test(meanden ~ shore.position, var.equal = FALSE, data = controldensbot)  #p-value = 0.7409
```

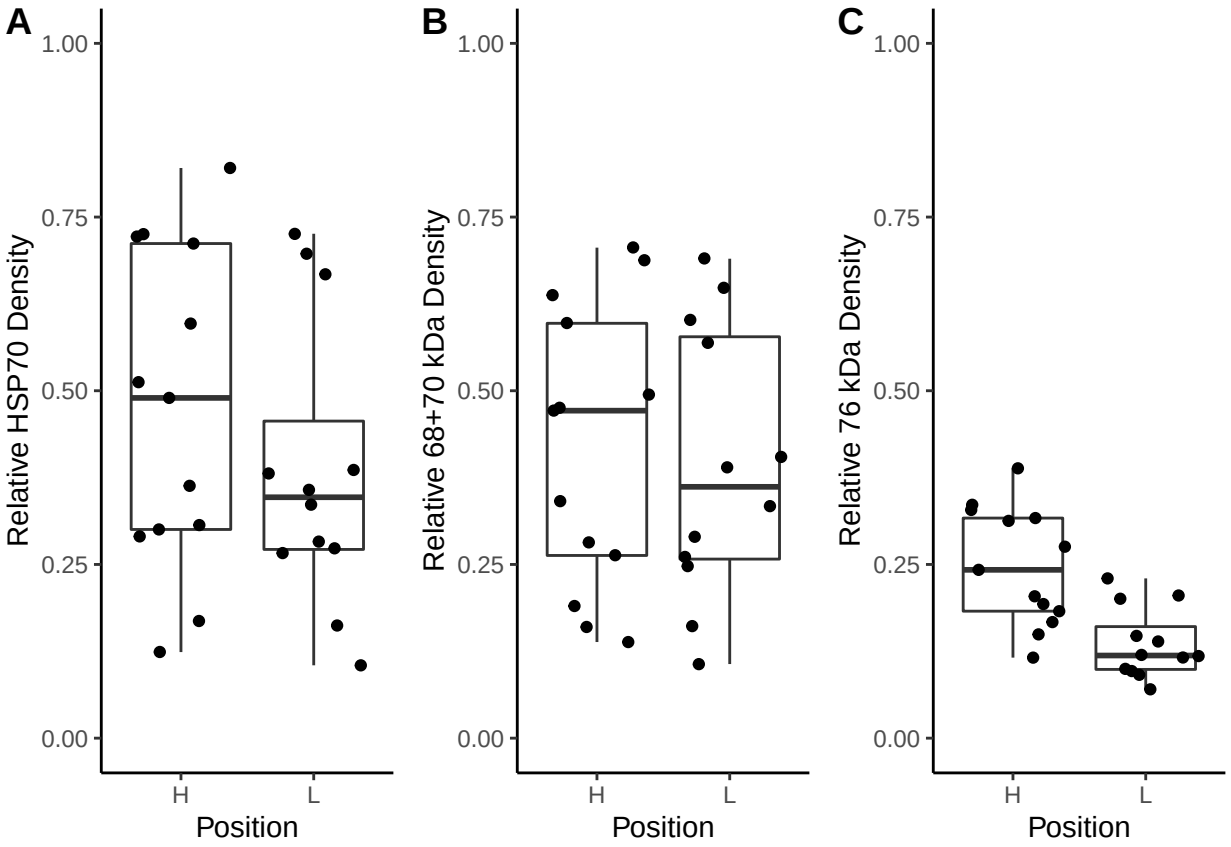
```
##
##  Welch Two Sample t-test
##
## data:  meanden by shore.position
## t = 0.33465, df = 22.959, p-value = 0.7409
## alternative hypothesis: true difference in means between group H and group L is not equal to 0
## 95 percent confidence interval:
##  -0.1378392  0.1910368
## sample estimates:
## mean in group H mean in group L
##       0.4187624       0.3921635
```

```
# Now making a box plot of basal levels - NO difference
controlboxbot <- ggplot(controldensbot, aes(x = shore.position,
    y = meanden)) + geom_boxplot() + ylim(0, 1) + xlab("Position") +
    ylab("Relative 68+70 kDa Density") + theme_classic() + geom_jitter()
controlboxbot
```



Now combining all three graphs to make one

```
plot_grid(controlbox, controlboxbot, controlboxtop, ncol = 3,
    labels = "AUTO")
```




```
ggsave("basalhsp70.tiff", device = "tiff", dpi = 300, width = 174,
    height = 100, units = "mm")
```

HSP70 vs position (High intertidal/Low intertidal) and recovery time (2 or 20 hours)

Top Bands

Make a subset of controls, and then in a seperate excel spread sheet, divide the relative densities of the top bands by the control averages. Then took the log 2 averages of those

```
controlonlytop_1 <- aggregate(controlonlytop$rel.dens, by = list(controlonlytop$position,
    controlonlytop$time.frozen, controlonlytop$recovery), FUN = mean)
log2top <- read.csv(here("Data", "meantop.csv"))
log2top.aov <- aov(log.2.fold ~ position * freeze.time * recovery,
    data = log2top)
summary(log2top.aov)
```

```
##                              Df Sum Sq Mean Sq F value  Pr(>F)
## position                      1  4.256   4.256   8.418 0.00784 **
## freeze.time                   1  0.011   0.011   0.022 0.88329
## recovery                      1  0.515   0.515   1.019 0.32282
## position:freeze.time          1  3.095   3.095   6.122 0.02081 *
## position:recovery             1  0.214   0.214   0.423 0.52144
```

```
## freeze.time:recovery           1  0.963   0.963   1.904 0.18030
## position:freeze.time:recovery  1  0.006   0.006   0.011 0.91771
## Residuals                     24 12.135   0.506
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

All bands

Group this data into just the experimentals, with mean densities

```
meandensityall <- all_bands %>%
    group_by(treatment, position, freeze.time, recovery, sam.num) %>%
    summarise(meanden = mean(rel.dens)) %>%
    filter(treatment == "E")
```

Then write this into an excel sheet, copy in control values, divide by those, and then log 2 those values. Making seperate columns for each function. Read in this new data set, mutate recovery and freeze time as a factor

```
# write.csv(meandensityall,here('Data', 'meanall1.csv'))

log2all <- read.csv(here("Data", "meanall.csv"))
log2all$recovery <- as.factor(log2all$recovery)
log2all$freeze.time <- as.factor(log2all$freeze.time)
```

Now to run an anova on this data, first checking for normality using shapirowilk, variances using bartlett test

```
shapiro.test(log2all$log.2.fold)  #p-value =  0.08344, normal
```

```
##
##  Shapiro-Wilk normality test
##
## data:  log2all$log.2.fold
## W = 0.95121, p-value = 0.08344
```

```
bartlett.test(log.2.fold ~ position, data = log2all)  #p-value = 0.4231
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  log.2.fold by position
## Bartlett's K-squared = 0.64162, df = 1, p-value = 0.4231
```

```
log2all.aov <- aov(log.2.fold ~ position * freeze.time * recovery,
    data = log2all)
summary(log2all.aov)
```

```
##                              Df Sum Sq Mean Sq F value Pr(>F)
## position                      1  0.629   0.629   0.942 0.3391
```

```
## freeze.time                    1  1.854   1.854   2.777 0.1054
## recovery                       1  2.893   2.893   4.334 0.0454 *
## position:freeze.time           1  0.980   0.980   1.468 0.2345
## position:recovery              1  0.126   0.126   0.189 0.6669
## freeze.time:recovery           1  3.443   3.443   5.157 0.0300 *
## position:freeze.time:recovery  1  0.495   0.495   0.741 0.3957
## Residuals                     32 21.363   0.668
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Bottom bands

```
log2bottom <- read.csv(here("Data", "meanbottom1.csv"))
log2bottom$recovery <- as.factor(log2bottom$recovery)
log2bottom$freeze.time <- as.factor(log2bottom$freeze.time)
shapiro.test(log2bottom$log.2.fold)  #p-value = 0.05715
```

```
##
##  Shapiro-Wilk normality test
##
## data:  log2bottom$log.2.fold
## W = 0.93586, p-value = 0.05715
```

```
bartlett.test(log.2.fold ~ position, data = log2bottom)  #p-value = 0.4688
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  log.2.fold by position
## Bartlett's K-squared = 0.52481, df = 1, p-value = 0.4688
```

```
log2bottom.aov <- aov(log.2.fold ~ position * freeze.time * recovery,
    data = log2bottom)
summary(log2bottom.aov)
```

```
##                               Df Sum Sq Mean Sq F value Pr(>F)
## position                       1  0.620  0.6204   0.872 0.3597
## freeze.time                    1  0.091  0.0913   0.128 0.7233
## recovery                       1  0.681  0.6807   0.957 0.3378
## position:freeze.time           1  2.202  2.2021   3.095 0.0913 .
## position:recovery              1  0.000  0.0001   0.000 0.9899
## freeze.time:recovery           1  1.157  1.1569   1.626 0.2144
## position:freeze.time:recovery  1  0.678  0.6785   0.954 0.3385
## Residuals                     24 17.074  0.7114
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Top bands

```
log2top <- read.csv(here("Data", "meantop.csv"))
log2top$recovery <- as.factor(log2top$recovery)
log2top$freeze.time <- as.factor(log2top$freeze.time)
logtop.aov <- aov(log.2.fold ~ position * freeze.time, data = log2top)
summary(logtop.aov)
```

```
##                      Df Sum Sq Mean Sq F value  Pr(>F)
## position              1  4.256   4.256   8.615 0.00659 **
## freeze.time           1  0.011   0.011   0.023 0.88176
## position:freeze.time  1  3.095   3.095   6.266 0.01842 *
## Residuals            28 13.832   0.494
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
TukeyHSD(logtop.aov)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = log.2.fold ~ position * freeze.time, data = log2top)
##
## $position
##          diff       lwr      upr     p adj
## L-H 0.7293835 0.2203582 1.238409 0.0065904
##
## $freeze.time
##          diff        lwr       upr     p adj
## 8-2 0.0372999 -0.4717253 0.5463252 0.8817607
##
## $`position:freeze.time`
##               diff        lwr       upr     p adj
## L:2-H:2  1.3514238  0.3919121 2.3109354 0.0033569
## H:8-H:2  0.6593402 -0.3001714 1.6188519 0.2609272
## L:8-H:2  0.7666834 -0.1928283 1.7261950 0.1531267
## H:8-L:2 -0.6920836 -1.6515952 0.2674281 0.2235163
## L:8-L:2 -0.5847404 -1.5442520 0.3747712 0.3610293
## L:8-H:8  0.1073432 -0.8521685 1.0668548 0.9898900
```

Graphing the data for freeze time vs position

All bands

```
arrangedata <- ddply(log2all, c("position", "freeze.time"), summarise,
    N = length(log.2.fold), mean = mean(log.2.fold), sd = sd(log.2.fold),
    std = sd/sqrt(N))
pd <- position_dodge(0.01)
# The errorbars overlapped, so use position_dodge to move
# them horizontally
all.bands.position <- ggplot(arrangedata, aes(x = position, y = mean,
    linetype = freeze.time, group = freeze.time)) + geom_errorbar(aes(ymin = mean -
    std, ymax = mean + std), width = 0.1, position = pd) + geom_line(size = 1.3) +
```

```
    geom_point() + ylab(bquote("" ~ Log[2] ~ "Fold Value of Relative HSP70 Density")) +
    xlab("Position") + theme(legend.position = "none") + theme_classic() +
    scale_color_manual(values = c("#E69F00", "#009E73")) + theme(text = element_text(size = 9)) +
    geom_hline(yintercept = c(0), linetype = "dashed") + scale_x_discrete(expand = expansion(mult = c(0
    0.2)), labels = c("High", "Low")) + theme(legend.position = "none") +
    ylim(-1.3, 1.2)
```

Bottom bands

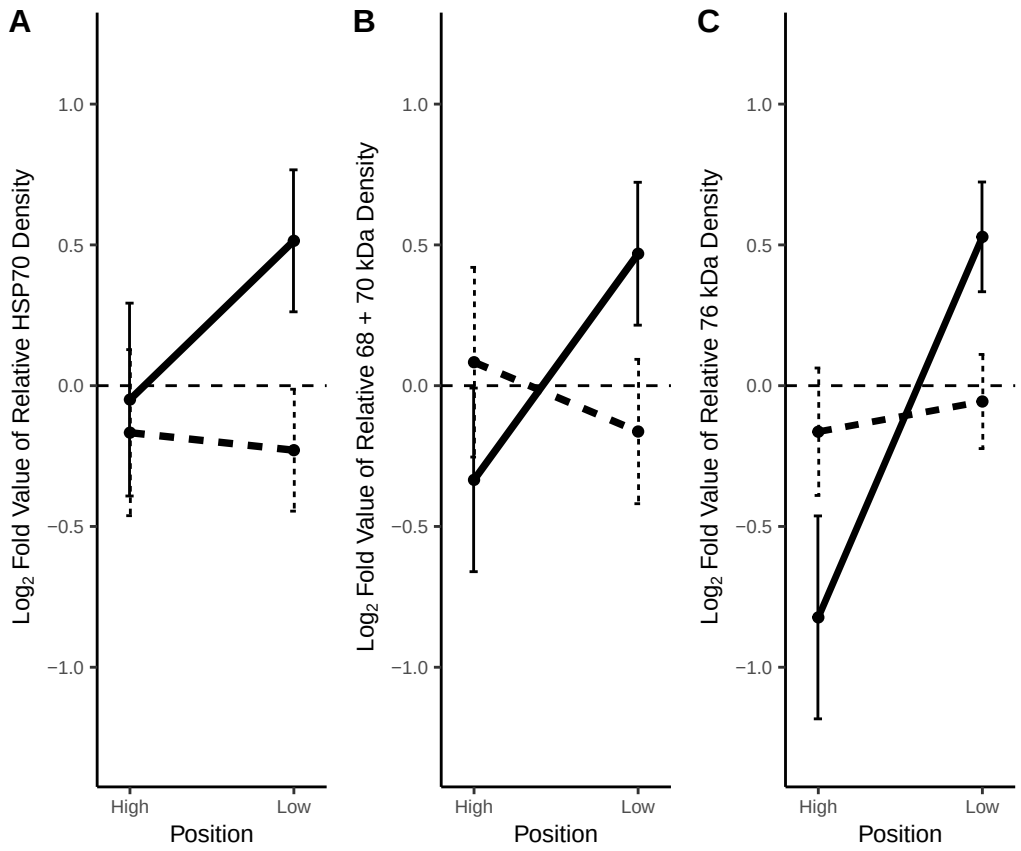
```
arrangedata.bottom <- ddply(log2bottom, c("position","freeze.time"), summarise,
                            N    = length(log.2.fold),
                            mean = mean(log.2.fold),
                            sd   = sd(log.2.fold),
                            std   = sd / sqrt(N))
pd <- position_dodge(0.01)
bottom.bands.position <- ggplot(arrangedata.bottom,
                                aes(x=position, y=mean,
                                    linetype=freeze.time,group=freeze.time)) +
  geom_errorbar(aes(ymin=mean-std, ymax=mean+std), width=.1, position=pd) +
  geom_line(size=1.3) +
  geom_point()+
  ylab(bquote(''~Log [2]~'Fold Value of Relative 68 + 70 kDa Density')) +
  xlab("Position")+
  theme_classic()+ #+scale_color_manual(values = c("#E69F00", "#009E73"))+
  theme(text = element_text(size=9))+
  geom_hline(yintercept=c(0), linetype="dashed")+
  scale_x_discrete(expand=expansion(mult=c(0.2,0.2)), labels=c("High", "Low"))+
  theme(legend.position="none")+ylim(-1.3, 1.2)
```

Top bands

```
arrangedata.top <- ddply(log2top, c("position", "freeze.time"),
    summarise, N = length(log.2.fold), mean = mean(log.2.fold),
    sd = sd(log.2.fold), std = sd/sqrt(N))
pd <- position_dodge(0.01)
top.bands.position <- ggplot(arrangedata.top, aes(x = position,
    y = mean, group = freeze.time, linetype = freeze.time)) +
    geom_errorbar(aes(ymin = mean - std, ymax = mean + std),
        width = 0.1, position = pd) + geom_line(size = 1.2) +
    geom_point() + ylab(bquote("" ~ Log[2] ~ "Fold Value of Relative 76 kDa Density")) +
    xlab("Position") + theme_classic() + theme(text = element_text(size = 9)) +
    scale_linetype_discrete(labels = c("4 x 2 h", "1 x 8 h")) +
    geom_hline(yintercept = c(0), linetype = "dashed") + scale_x_discrete(expand = expansion(mult = c(0
    0.2)), labels = c("High", "Low")) + labs(linetype = "Freeze Treatment") +
    ylim(-1.3, 1.2) + theme(legend.position = "none")
```

Now plotting all together. Not shown here, but legend was added in using get.legend function in cowplot

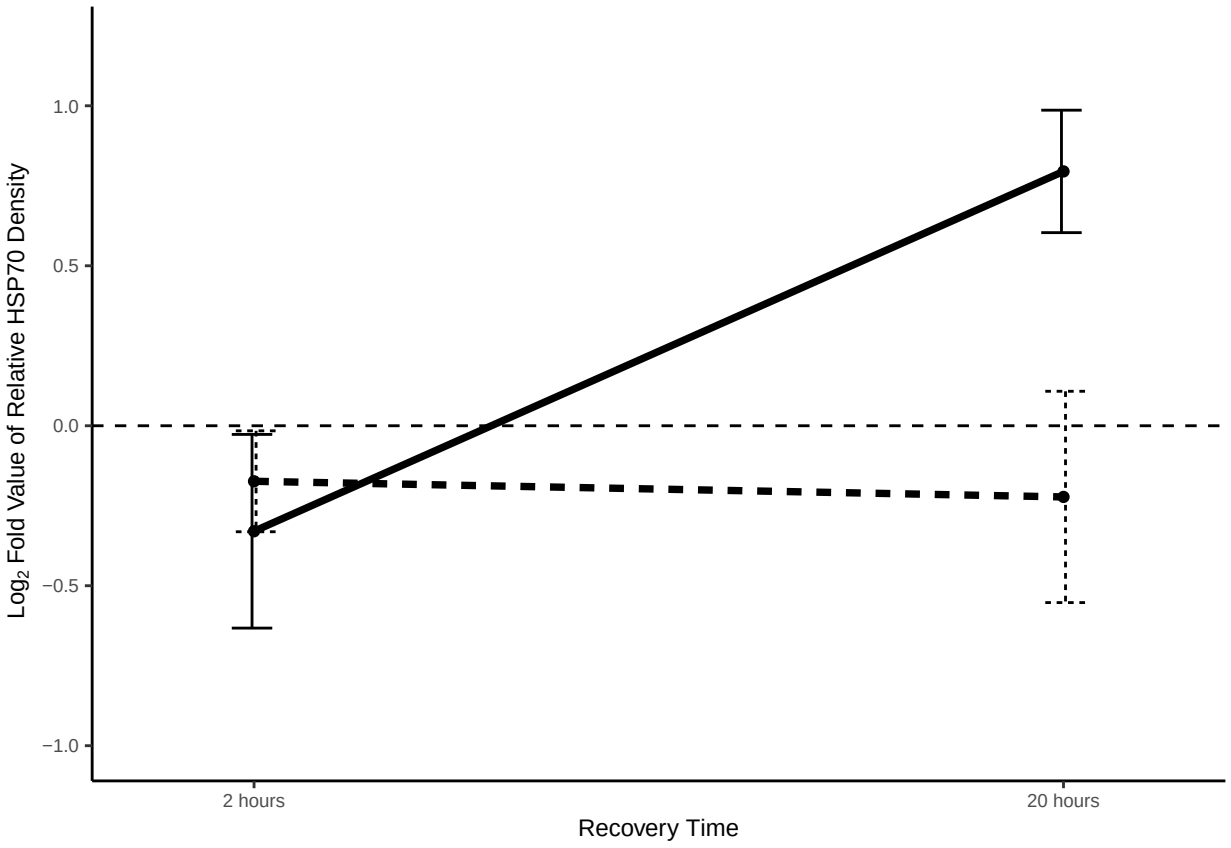
```
legend <- get_legend(top.bands.position)
fullpositiongraph <- plot_grid(all.bands.position, bottom.bands.position,
    top.bands.position, legend, labels = c("A", "B", "C"), label_size = 12,
    ncol = 4, rel_widths = c(4, 4, 4, 2.5))
fullpositiongraph
```



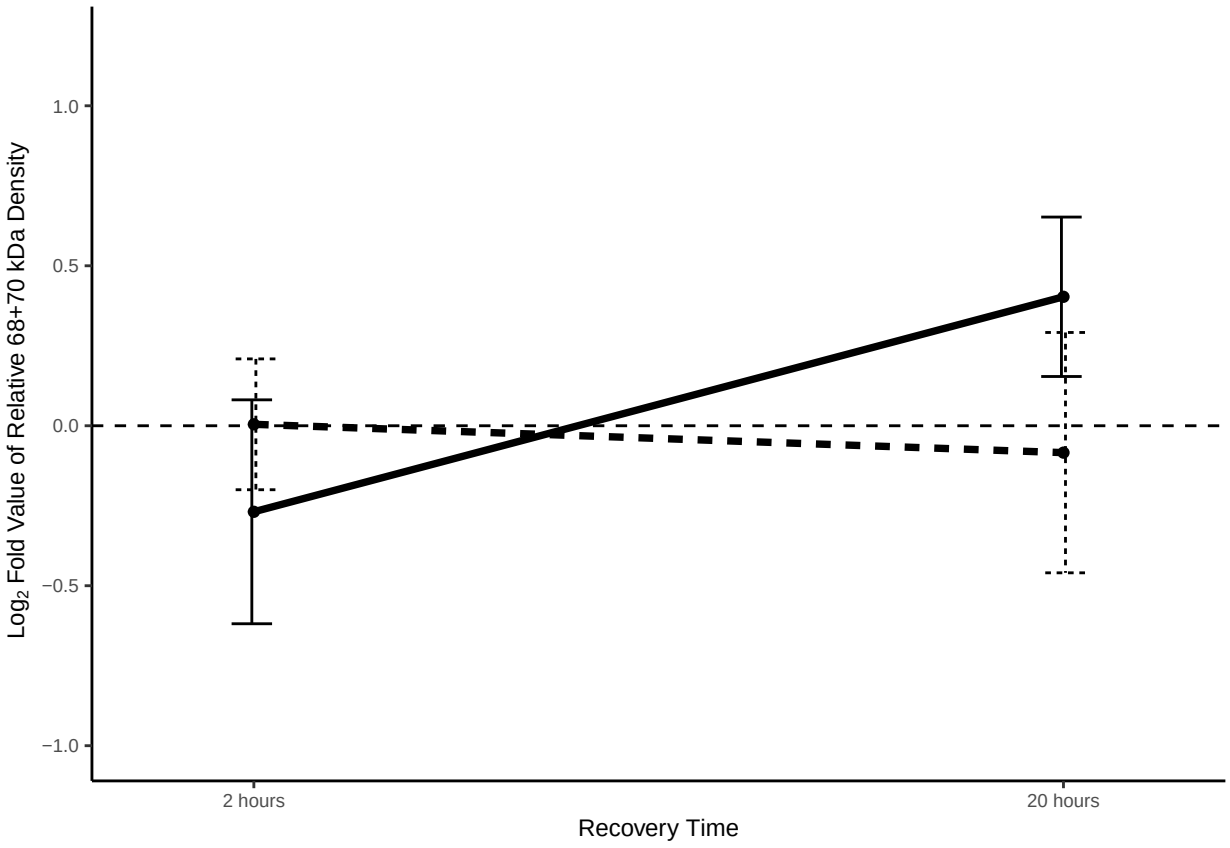

```
# ggsave('positionhsp70.tiff', device = 'tiff', dpi = 300,
# width = 174, height = 90, units = 'mm', bg = 'white')
```

This same thing was done with the recovery data, letters to denote significance were added after in "paint".

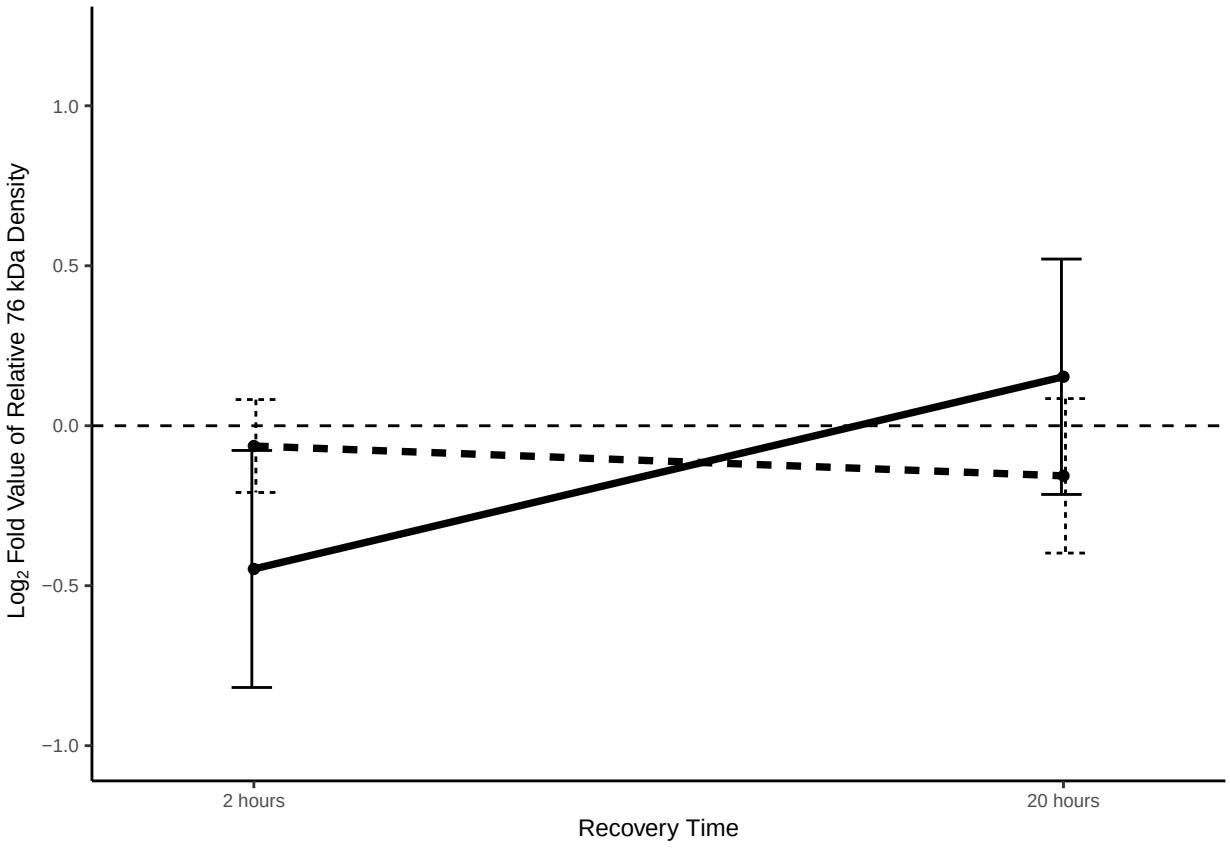
```
arrangedatar <- ddply(log2all, c("recovery","freeze.time"), summarise,
                      N    = length(log.2.fold),
                      mean = mean(log.2.fold),
                      sd   = sd(log.2.fold),
                      std   = sd / sqrt(N))
pd <- position_dodge(0.01)  #The errorbars overlapped, so use position_dodge to move them horizontally
all.bands.recovery <- ggplot(arrangedatar, aes(x=recovery, y=mean, linetype=freeze.time,group=freeze.ti
  geom_errorbar(aes(ymin=mean-std, ymax=mean+std), width=.1, position=pd) +
  geom_line(size=1.3) +
  geom_point()+
  ylab(bquote(''~Log [2]~'Fold Value of Relative HSP70 Density')) +
  xlab("Recovery Time")+ theme(legend.position = "none")+
  theme_classic() +
  theme(text = element_text(size=9))+
  scale_color_manual(values = c("#E69F00", "#009E73"), labels = c("2 x 4 h", "1 x 8 h"))+
  geom_hline(yintercept=c(0), linetype="dashed")+
  scale_x_discrete(expand=expansion(mult=c(0.2,0.2)), labels=c("2 hours", "20 hours"))+
  theme(legend.position="none")+ ylim(-1, 1.2)
all.bands.recovery
```



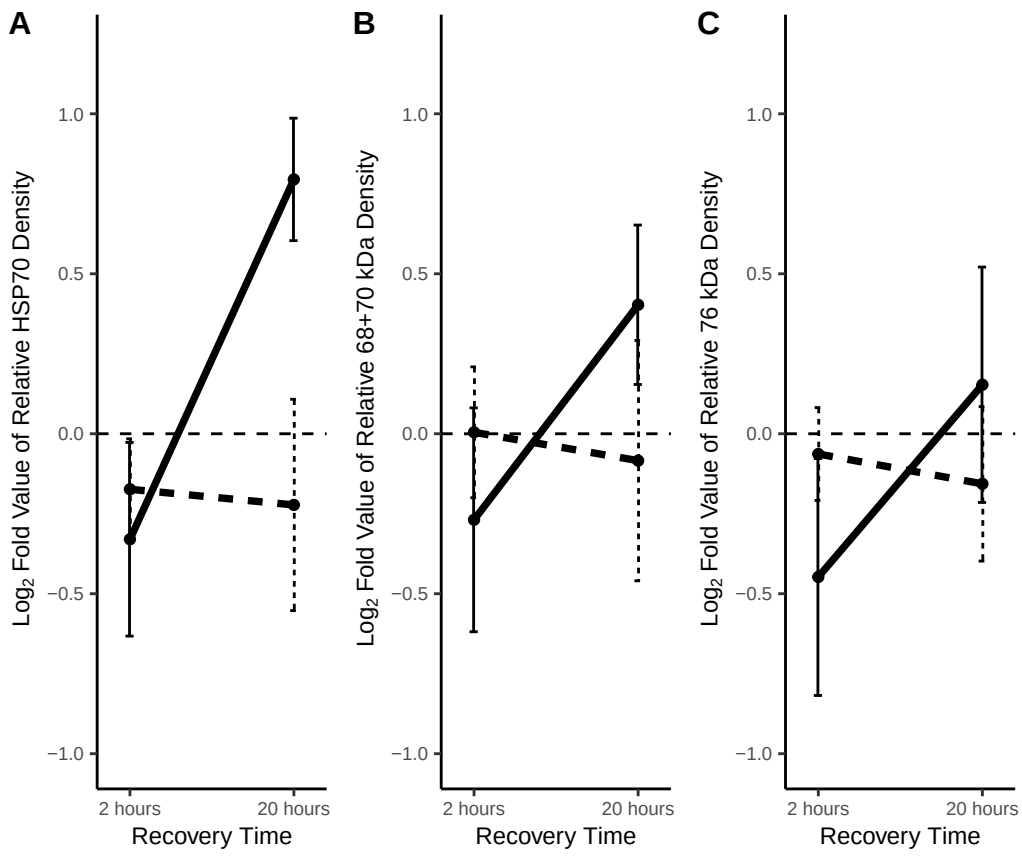

```
###Bottom bands
arrangedata.bottomr <- ddply(log2bottom, c("recovery","freeze.time"), summarise,
                             N    = length(log.2.fold),
                             mean = mean(log.2.fold),
                             sd   = sd(log.2.fold),
                             std   = sd / sqrt(N))
pd <- position_dodge(0.01)  #The errorbars overlapped, so use position_dodge to move them horizontally
bottom.bands.recovery <- ggplot(arrangedata.bottomr, aes(x=recovery, y=mean, linetype=freeze.time,group=
  geom_errorbar(aes(ymin=mean-std, ymax=mean+std), width=.1, position=pd) +
  geom_line(size=1.3) +
  geom_point()+
ylab(bquote(''~Log [2]~'Fold Value of Relative 68+70 kDa Density')) +
  xlab("Recovery Time")+
  theme_classic() +
  scale_color_manual(values = c("#E69F00", "#009E73"))+
  theme(text = element_text(size=9))+
  geom_hline(yintercept=c(0), linetype="dashed")+
  scale_x_discrete(expand=expansion(mult=c(0.2,0.2)), labels=c("2 hours", "20 hours"))+ theme(legend.pos
bottom.bands.recovery
```




```
##Top
arrangedata.topr <- ddply(log2top, c("recovery","freeze.time"), summarise,
                          N    = length(log.2.fold),
                          mean = mean(log.2.fold),
                          sd   = sd(log.2.fold),
                          std   = sd / sqrt(N))
pd <- position_dodge(0.01)  #The errorbars overlapped, so use position_dodge to move them horizontally
top.bands.recovery <- ggplot(arrangedata.topr, aes(x=recovery, y=mean, linetype=freeze.time,group=freeze
  geom_errorbar(aes(ymin=mean-std, ymax=mean+std), width=.1, position=pd) +
  geom_line(size=1.3) +
  geom_point()+
  ylab(bquote(''~Log [2]~'Fold Value of Relative 76 kDa Density')) +
  xlab("Recovery Time")+
  theme_classic() +
  scale_linetype_discrete(labels = c("4 x 2 h", "1 x 8 h"))+
  theme(text = element_text(size=9))+
  geom_hline(yintercept=c(0), linetype="dashed")+
  scale_x_discrete(expand=expansion(mult=c(0.2,0.2)), labels=c("2 hours", "20 hours"))+
  labs(linetype='Freeze Treatment')+  ylim(-1, 1.2)+theme(legend.position="none")
top.bands.recovery
```




```
legend.recovery <- get_legend(
  # create some space to the left of the legend
  top.bands.recovery + theme(legend.box.margin = margin(0, 0, 0, 12))
)

recovery<- plot_grid(all.bands.recovery, bottom.bands.recovery, top.bands.recovery, legend.recovery, lab
recovery
```




```
#ggsave("recoveryhsp70.tiff", device = "tiff", dpi = 300, width = 174, height = 90, units = "mm", bg =
```

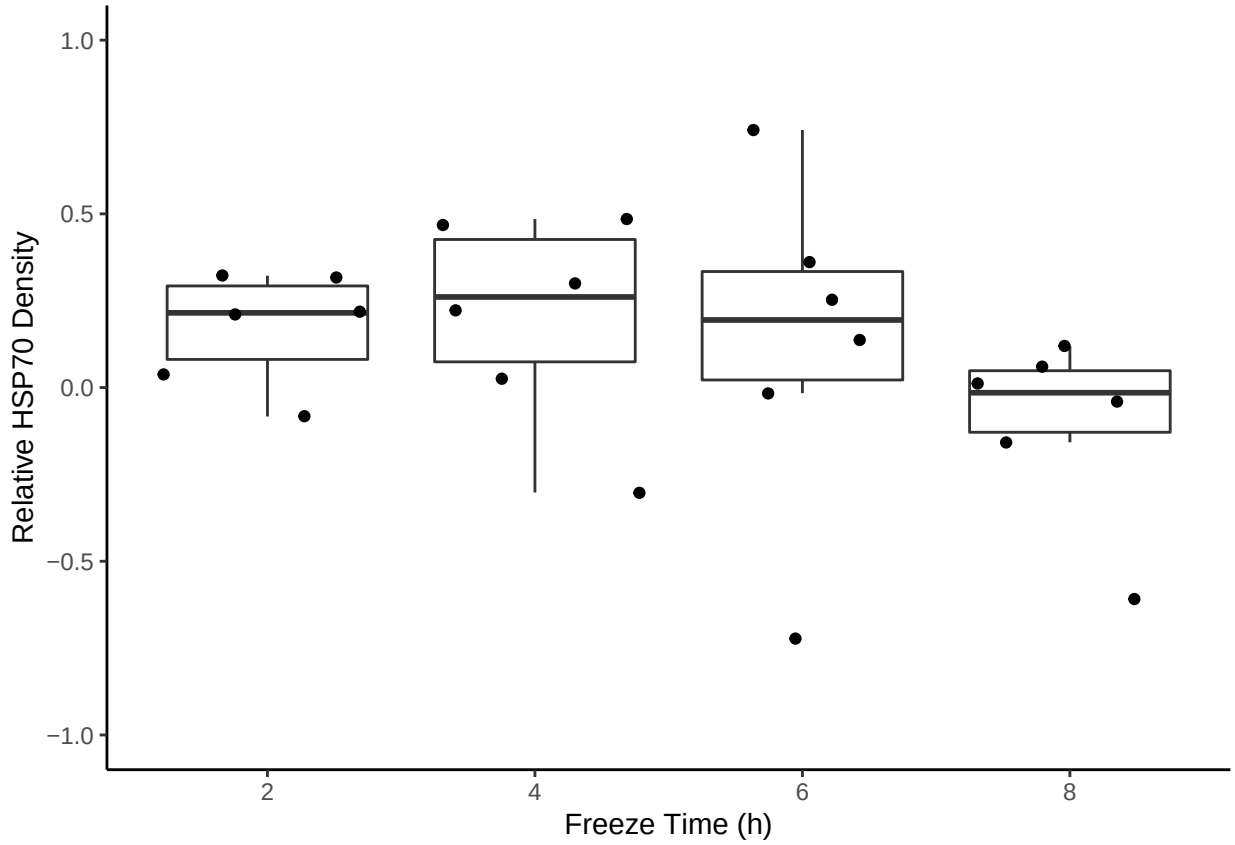
Individual Freezes

Similar methods to those used above

```
second <- read.csv(here("Data", "All bands_2.csv"))

controlsecond <- subset(second, second$treatment == "C")
controlsecond1 <- aggregate(controlsecond$dens, by = list(controlsecond$freeze.time),
    FUN = mean)
# write_xlsx(controllall1,'C:\\Users\\laure\\OneDrive\\Year
# 4\\Mussel Data\\2,4,6,8 hr experiment\\controlall.xlsx')
# write.xlsx(meandensitytop,'C:\\Users\\laure\\OneDrive\\Year
# 4\\Mussel Data\\2,4,6,8 hr experiment\\all.xlsx')

log2second <- read.csv(here("Data", "log2all_second.csv"))
log2.aov.second <- aov(log.2.fold ~ as.factor(freeze.time), data = log2second)
summary(log2.aov.second)
```

```
##                      Df Sum Sq Mean Sq F value Pr(>F)
## as.factor(freeze.time)  3 0.3397  0.1133   1.069  0.385
## Residuals              20 2.1190  0.1060
```

```
secondboxplot <- ggplot(log2second, aes(x = as.factor(freeze.time),
    y = log.2.fold)) + geom_boxplot(outlier.shape = NA) + ylim(-1,
    1) + xlab("Freeze Time (h)") + ylab("Relative HSP70 Density") +
    theme_classic() + geom_jitter()
secondboxplot
```




```
# ggsave('individualfreeze.tiff', device = 'tiff', dpi =
# 300, width = 174, height = 90, units = 'mm', bg =
# 'white')
```

Ubiquitin

Reading in data

```
ubi <- read.csv(here("Data", "Ubiquitin.csv"))
```

Seeing if ubiquitin levels are affected by shore position or recovery time

Make a subset of controls

```
ubicontrol <- subset(ubi, ubi$treatment == "C")
ubicontrol1 <- aggregate(ubicontrol$rel.dens, by = list(ubicontrol$position,
    ubicontrol$freeze.time, ubicontrol$recovery), FUN = mean)
```

Create an excel file with controls, use these to divide against experimentals

```
# write.xlsx(control1)
```

Then within excel divide each exp value by mean value for control and reupload the excel sheet, also take the log base 2 of all the relative densities

Then use this sheet from now on... first read in data

```
log2 <- read.csv(here("Data", "Ubiquitin_modified.csv"))
```

Now do an anova on the log 2 data

```
shapiro.test(log2$log.2)  #p-value = 0.2554
```

```
##
##  Shapiro-Wilk normality test
##
## data:  log2$log.2
## W = 0.96472, p-value = 0.2554
```

```
log2.aov <- aov(log.2 ~ position * as.factor(freeze.time) * as.factor(recovery),
    data = log2)
summary(log2.aov)
```

```
##                                                    Df Sum Sq Mean Sq F value
## position                                            1 0.0068 0.00680   0.583
## as.factor(freeze.time)                              1 0.0249 0.02487   2.129
## as.factor(recovery)                                 1 0.1430 0.14304  12.247
## position:as.factor(freeze.time)                     1 0.1203 0.12029  10.299
## position:as.factor(recovery)                        1 0.0234 0.02344   2.007
## as.factor(freeze.time):as.factor(recovery)          1 0.1019 0.10191   8.725
## position:as.factor(freeze.time):as.factor(recovery)  1 0.0001 0.00013   0.011
## Residuals                                          31 0.3621 0.01168
##                                                     Pr(>F)
## position                                           0.45108
## as.factor(freeze.time)                             0.15458
## as.factor(recovery)                                0.00143 **
## position:as.factor(freeze.time)                    0.00309 **
## position:as.factor(recovery)                       0.16660
## as.factor(freeze.time):as.factor(recovery)         0.00594 **
## position:as.factor(freeze.time):as.factor(recovery) 0.91608
## Residuals
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now reducing the anova down to the basics using stepAIC

```
log3.aov <- MASS::stepAIC(log2.aov)
```

```
## Start:  AIC=-166.5
## log.2 ~ position * as.factor(freeze.time) * as.factor(recovery)
##
##                                                       Df  Sum of Sq     RSS
## - position:as.factor(freeze.time):as.factor(recovery)  1 0.00013181 0.36222
## <none>                                                               0.36209
##                                                          AIC
## - position:as.factor(freeze.time):as.factor(recovery) -168.48
## <none>                                                -166.50
##
## Step:  AIC=-168.48
## log.2 ~ position + as.factor(freeze.time) + as.factor(recovery) +
##     position:as.factor(freeze.time) + position:as.factor(recovery) +
##     as.factor(freeze.time):as.factor(recovery)
##
##                                             Df Sum of Sq     RSS     AIC
## <none>                                                   0.36222 -168.48
## - position:as.factor(recovery)               1  0.026378 0.38860 -167.74
## - as.factor(freeze.time):as.factor(recovery)  1  0.101909 0.46413 -160.81
## - position:as.factor(freeze.time)             1  0.110744 0.47297 -160.08
```

```
summary(log3.aov)
```

```
##                                           Df Sum Sq Mean Sq F value  Pr(>F)
## position                                   1 0.0068 0.00680   0.601 0.44384
## as.factor(freeze.time)                     1 0.0249 0.02487   2.197 0.14805
## as.factor(recovery)                        1 0.1430 0.14304  12.637 0.00120 **
## position:as.factor(freeze.time)            1 0.1203 0.12029  10.627 0.00265 **
## position:as.factor(recovery)               1 0.0234 0.02344   2.070 0.15989
## as.factor(freeze.time):as.factor(recovery)  1 0.1019 0.10191   9.003 0.00519 **
## Residuals                                 32 0.3622 0.01132
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
TukeyHSD(log3.aov)
```

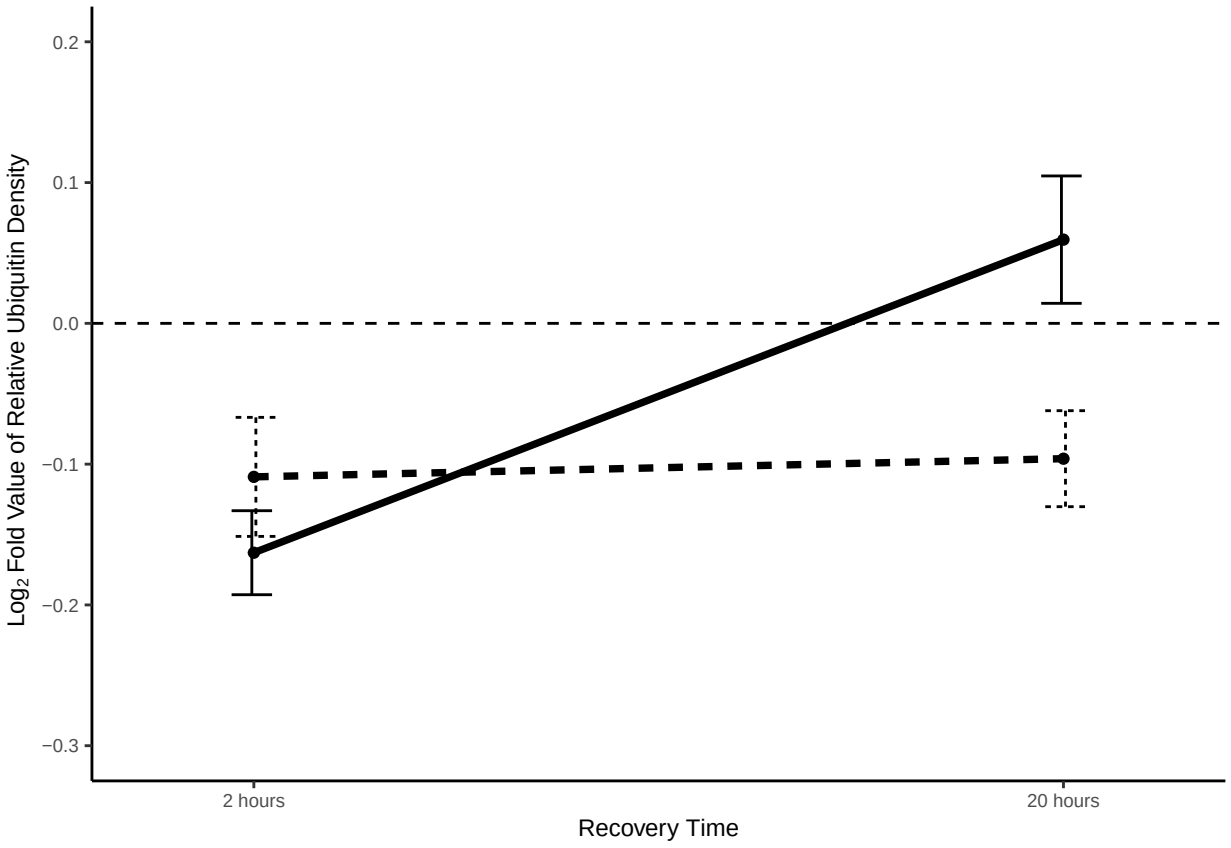
```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = log.2 ~ position + as.factor(freeze.time) + as.factor(recovery) + position:as.fac
##
## $position
##            diff        lwr        upr    p adj
## L-H -0.02642637 -0.0958536 0.04300085 0.443839
##
## $`as.factor(freeze.time)`
##            diff        lwr       upr     p adj
## 8-2 -0.05050393 -0.1199312 0.0189233 0.1481905
##
## $`as.factor(recovery)`
##           diff        lwr       upr     p adj
## 20-2 0.1210782 0.05165096 0.1905054 0.0012085
##
```

```
## $`position:as.factor(freeze.time)`
##                 diff         lwr          upr     p adj
## L:2-H:2  0.08304861 -0.04586372  0.211960936 0.3178026
## H:8-H:2  0.06361413 -0.06883069  0.196058963 0.5688261
## L:8-H:2 -0.07563533 -0.20454766  0.053277004 0.3986420
## H:8-L:2 -0.01943447 -0.15187930  0.113010357 0.9783432
## L:8-L:2 -0.15868393 -0.28759626 -0.029771601 0.0110720
## L:8-H:8 -0.13924946 -0.27169429 -0.006804631 0.0363525
##
## $`position:as.factor(recovery)`
##                   diff         lwr        upr     p adj
## L:2-H:2   -0.07741819 -0.20633052 0.05149414 0.3783019
## H:20-H:2   0.07070301 -0.06174182 0.20314784 0.4807981
## L:20-H:2   0.09154724 -0.03736509 0.22045957 0.2383966
## H:20-L:2   0.14812119  0.01567636 0.28056602 0.0236285
## L:20-L:2   0.16896543  0.04005310 0.29787776 0.0063312
## L:20-H:20  0.02084423 -0.11160059 0.15328906 0.9735342
##
## $`as.factor(freeze.time):as.factor(recovery)`
##                   diff         lwr         upr     p adj
## 8:2-2:2     0.05219258 -0.07671975  0.18110491 0.6939268
## 2:20-2:2    0.22070512  0.09179279  0.34961745 0.0003165
## 8:20-2:2    0.06835536 -0.06408947  0.20080019 0.5095686
## 2:20-8:2    0.16851254  0.03960021  0.29742487 0.0064912
## 8:20-8:2    0.01616278 -0.11628205  0.14860761 0.9872951
## 8:20-2:20 -0.15234976 -0.28479459 -0.01990493 0.0191357
```

Plotting Ubiquitin vs recovery/position graphs

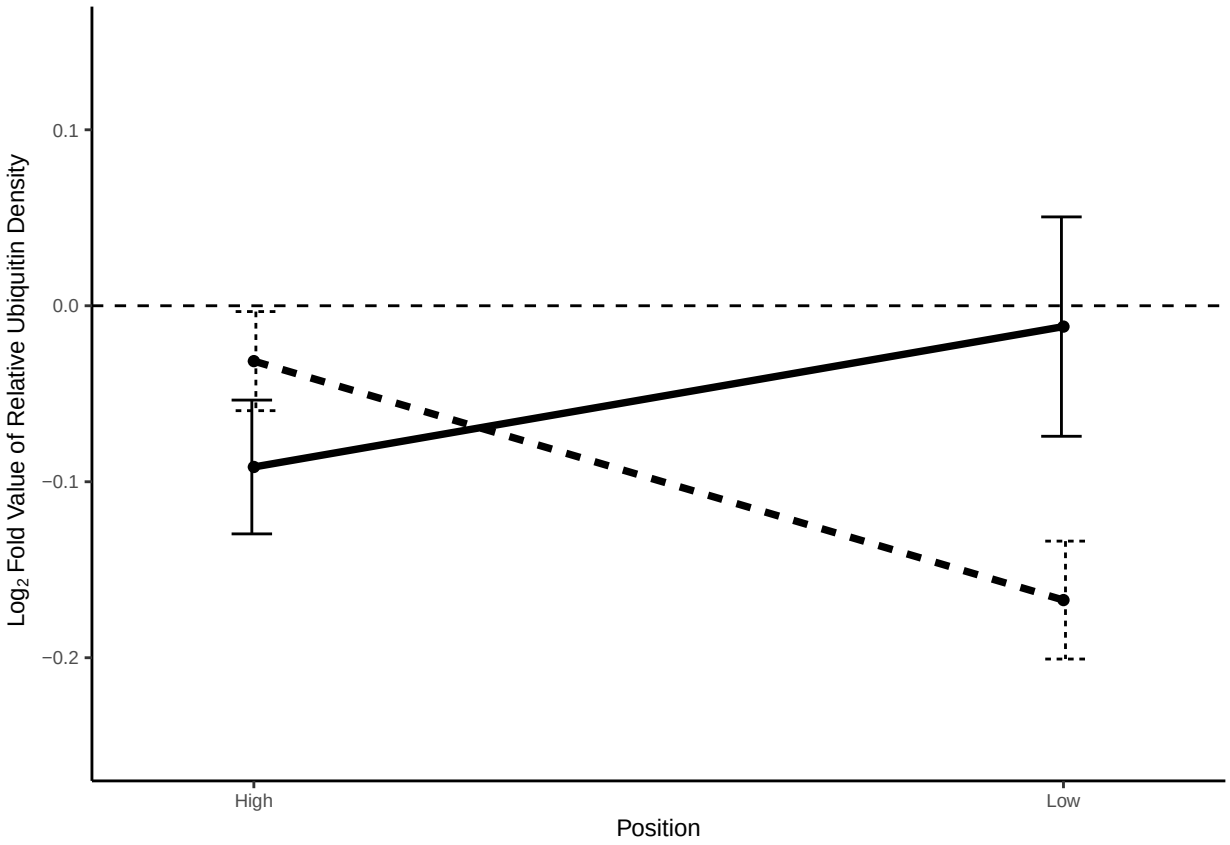
First ubiqutin expression vs recovery

```
log2$recovery <- as.factor(log2$recovery)
log2$freeze.time <- as.factor(log2$freeze.time)
arrangedata <- ddply(log2, c("recovery", "freeze.time"), summarise,
    N = length(log.2), mean = mean(log.2), sd = sd(log.2), std = sd/sqrt(N))
pd <- position_dodge(0.01)
# The errorbars overlapped, so use position_dodge to move
# them horizontally
ubiquitin_recovery <- ggplot(arrangedata, aes(x = recovery, y = mean,
    linetype = freeze.time, group = freeze.time)) + geom_errorbar(aes(ymin = mean -
    std, ymax = mean + std), width = 0.1, position = pd) + geom_line(size = 1.3) +
    geom_point() + ylab(bquote("" ~ Log[2] ~ "Fold Value of Relative Ubiquitin Density")) +
    xlab("Recovery Time") + theme_classic() + theme(text = element_text(size = 9)) +
    scale_color_manual(values = c("#E69F00", "#009E73"), labels = c("4 x 2 h",
        "1 x 8 h")) + geom_hline(yintercept = c(0), linetype = "dashed") +
    scale_x_discrete(expand = expansion(mult = c(0.2, 0.2)),
        labels = c("2 hours", "20 hours")) + labs(color = "Freeze Treatment") +
    ylim(-0.3, 0.2) + theme(legend.position = "none")
ubiquitin_recovery
```



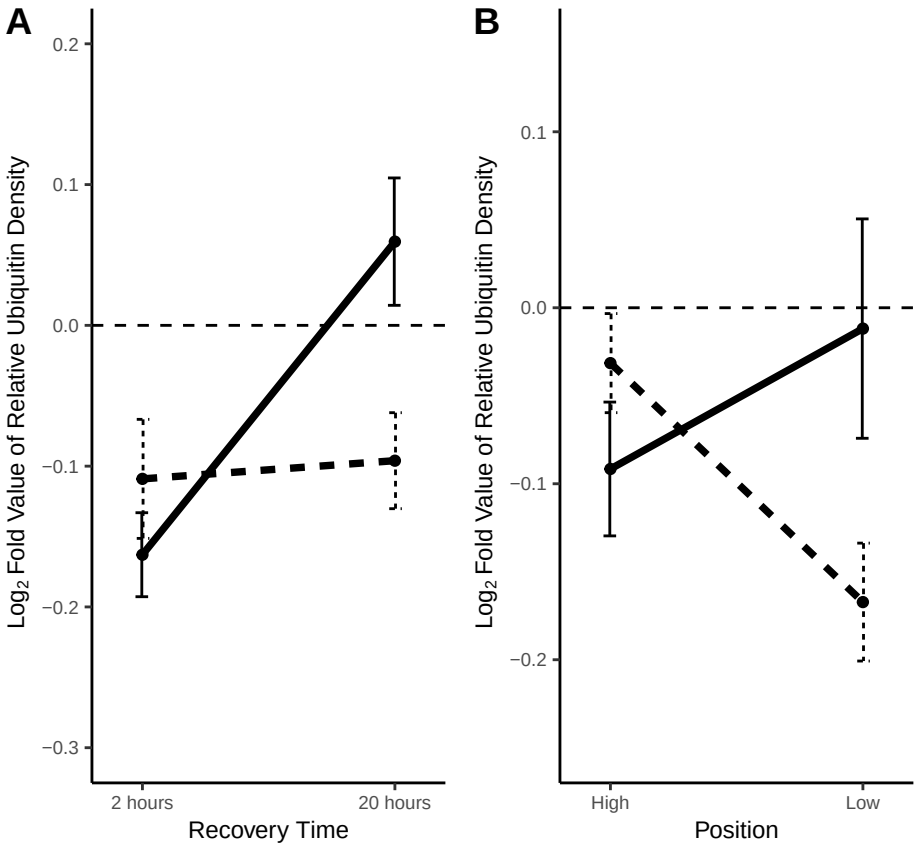
Ubiqutin vs position

```
arrangedataposition <- ddply(log2, c("position", "freeze.time"),
    summarise, N = length(log.2), mean = mean(log.2), sd = sd(log.2),
    std = sd/sqrt(N))
pd <- position_dodge(0.01)
# The errorbars overlapped, so use position_dodge to move
# them horizontally
ubiquitin_position <- ggplot(arrangedataposition, aes(x = position,
    y = mean, linetype = freeze.time, group = freeze.time)) +
    geom_errorbar(aes(ymin = mean - std, ymax = mean + std),
        width = 0.1, position = pd) + geom_line(size = 1.3) +
    geom_point() + ylab(bquote("" ~ Log[2] ~ "Fold Value of Relative Ubiquitin Density")) +
    xlab("Position") + theme_classic() + theme(text = element_text(size = 9)) +
    scale_linetype_discrete(labels = c("4 x 2 h", "1 x 8 h")) +
    geom_hline(yintercept = c(0), linetype = "dashed") + scale_x_discrete(expand = expansion(mult = c(0
    0.2)), labels = c("High", "Low")) + labs(linetype = "Freeze Treatment") +
    ylim(-0.25, 0.15) + theme(legend.position = "none")
ubiquitin_position
```



Combining the graphs

```
legend.ubiqutin <- get_legend(
  # create some space to the left of the legend
  ubiquitin_position + theme(legend.box.margin = margin(0, 0, 0, 12))
)
ubiqutin <- plot_grid(ubiquitin_recovery, ubiquitin_position,
                      legend.ubiqutin, ncol=3, rel_widths = c(3, 3, 2),
                      labels = c("A", "B"))
ubiqutin
```




```
#ggsave("ubiquitinlinegraph", device = "tiff", dpi = 300, width = 174, height = 90, units = "mm", bg =
```

Basal expression of Ubiquitin

Only use controls, these are the only ones needed for basal expression, now aggregate this subset to group it by position (H vs L intertidal)

```
highlow <- aggregate(ubicontrol$rel.dens, by = list(ubicontrol$position),
    FUN = mean)
shapiro.test(ubicontrol$rel.dens)  # p-value = 5.61e-05, not normal
```

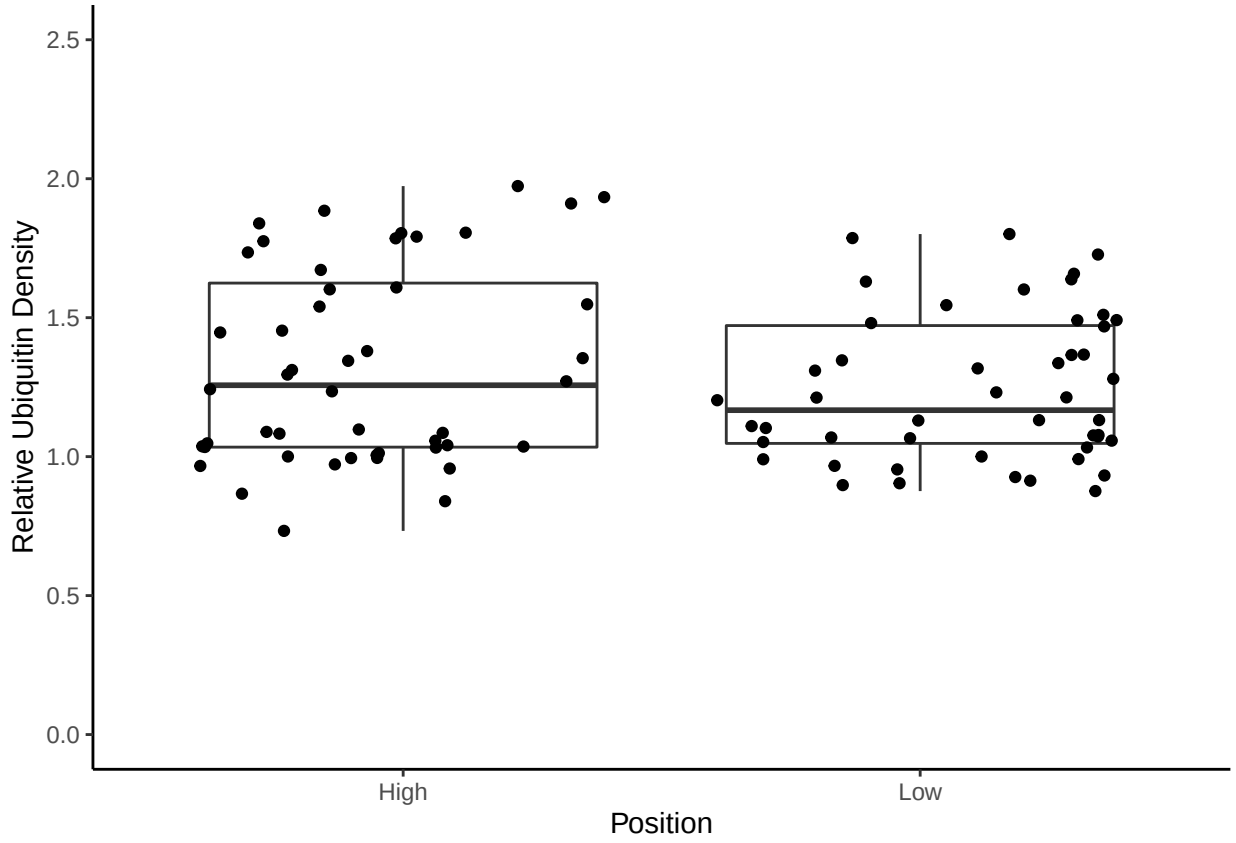
```
##
##  Shapiro-Wilk normality test
##
## data:  ubicontrol$rel.dens
## W = 0.92824, p-value = 5.61e-05
```

```
wilcox.test(rel.dens ~ position, data = ubicontrol)  #p-value = 0.3556
```

```
##
##  Wilcoxon rank sum exact test
##
## data:  rel.dens by position
## W = 1279, p-value = 0.3556
## alternative hypothesis: true location shift is not equal to 0
```

Now making a box plot of basal levels

```
ubi_basal <- ggplot(ubicontrol, aes(x = position, y = rel.dens)) +
    geom_boxplot() + ylim(0, 2.5) + xlab("Position") + ylab("Relative Ubiquitin Density") +
    theme_classic() + geom_jitter() + scale_x_discrete(labels = c("High",
    "Low"))
ubi_basal
```



Correlation between HSP and ubiqutuin

Modified the HSP and ubiquitin data to make 2 new CSVs with just the means of ubiquitin and HSP on the. One has the averages across treatment groups, the other has the value for each sample

```
compdata <- read.csv(here("Data", "Correlation btw hsp and ub.csv"))
sampledata <- read.csv(here("Data", "Sample correlation btw hsp and ub.csv"))
```

First check assumptions for ub and hsp, line is linear so it works

```
regcomp <- ggscatter(compdata, x = "ub", y = "hsp", add = "reg.line",
    conf.int = TRUE, cor.coef = TRUE, cor.method = "pearson",
    xlab = "Relative Ubiquitin Value", ylab = "Relative HSP70 Value")
shapiro.test(compdata$ub)
```

```
##
```

```
##  Shapiro-Wilk normality test
##
## data:  compdata$ub
## W = 0.9147, p-value = 0.3884
```

```
## Data = normal, p value = 0.38
shapiro.test(compdata$hsp)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  compdata$hsp
## W = 0.89586, p-value = 0.265
```

```
## Data is normal, p value = 0.28

## THen check assumptions for log.2 hsp and
## ub###############
logcomp <- ggscatter(compdata, x = "log.ub", y = "log.hsp", add = "reg.line",
    conf.int = TRUE, cor.coef = TRUE, cor.method = "pearson",
    xlab = "Log  Ubiquitin Value", ylab = "Log  HSP Value")
## Line is also linear so it works

shapiro.test(compdata$log.ub)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  compdata$log.ub
## W = 0.91513, p-value = 0.3916
```

```
## Data = normal, p value = 0.3916
shapiro.test(compdata$log.hsp)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  compdata$log.hsp
## W = 0.94058, p-value = 0.6168
```

```
## Data is normal, p value = 0.832
```

Testing for correlation

```
## un-log transformed data
hsp.ub <- cor.test(compdata$ub, compdata$hsp, method = "pearson")
hsp.ub
```

```
##
##  Pearson's product-moment correlation
##
## data:  compdata$ub and compdata$hsp
## t = 4.0252, df = 6, p-value = 0.006918
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3758218 0.9731325
## sample estimates:
##       cor
## 0.8542599
```

```
hsp.ub$p.value  #0.006917628
```

```
## [1] 0.006917628
```

```
hsp.ub$estimate  #0.8542599
```

```
##       cor
## 0.8542599
```

```
## log transformd data
log.hsp.ub <- cor.test(compdata$log.ub, compdata$log.hsp, method = "pearson")
log.hsp.ub
```

```
##
##  Pearson's product-moment correlation
##
## data:  compdata$log.ub and compdata$log.hsp
## t = 3.6738, df = 6, p-value = 0.01041
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3078295 0.9687268
## sample estimates:
##       cor
## 0.8320211
```

```
log.hsp.ub$p.value  #0.0104069
```

```
## [1] 0.0104069
```

```
log.hsp.ub$estimate  #0.8320211
```

```
##       cor
## 0.8320211
```

Plotting correlation graphs

Sample by sample graphs

```
sample <- ggscatter(sampledata, x = "ub", y = "hsp", add = "reg.line",
    conf.int = TRUE, cor.coef = TRUE, cor.method = "pearson",
    xlab = "Relative Ubiquitin Value", ylab = "Relative HSP70 Value")

fulltreatment <- ggscatter(compdata, x = "ub", y = "hsp", add = "reg.line",
    conf.int = TRUE, cor.coef = TRUE, cor.method = "pearson",
    xlab = "Relative Ubiquitin Value", ylab = "Relative HSP70 Value")
```

Combining plots

```
combinedplot <- plot_grid(sample, fulltreatment, labels = c("A",
    "B"))
```

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

```
combinedplot
```