Expression vector graphs, stats, and data wrangling

Jennifer Tran

2024-05-13

Load necessary packages for these graphs:

```
require('pacman')

p_load(dplyr, data.table, ggplot2, tidyr, RColorBrewer, stringr, ggridges, colourpicker, ggbreak)
```

Conjugative transfer and transposition efficency graphs (Fig S1)

Plasmid conjugation Load plasmid data from spreadsheet (CFUs in LB vs LB+Kanamycin) as plasmid_eff

```
##     parent rep plasmid LB_Kan    LB
## 1:   17978   1      EV  1e+07 4e+08
## 2:   17978   2      EV  7e+06 1e+08
## 3:   17978   3      EV  7e+06 4e+08
## 4: BW25113   1      EV  9e+08 2e+09
## 5: BW25113   2      EV  1e+09 3e+09
## 6: BW25113   3      EV  3e+09 2e+09
```

Calculate efficiency if not already included

```
plasmid_eff$efficiency <- plasmid_eff$LB_Kan/plasmid_eff$LB
```

Filter for *A. baumannii* ATCC 17978 and *E. coli* BW25113 empty vector data
Calculate mean efficiences for each

```
plasmid_plot <- plasmid_eff %>%
  filter(parent %in% c("17978", "BW25113"), plasmid == "EV")

mean_efficiencies <- plasmid_plot %>%
  group_by(parent) %>%
  summarise(
    mean_efficiency = mean(efficiency),
    .groups = 'drop'
  )
```

```
## # A tibble: 2 x 2
##   parent  mean_efficiency
##   <chr>             <dbl>
## 1 17978            0.0375
## 2 BW25113          0.761
```

Plot a dot plot with the mean as a bar and dashed line for limit of detection

```
ggplot() +
  geom_errorbar(data = mean_efficiencies,
                aes(x = parent, ymin = mean_efficiency, ymax = mean_efficiency),
                width = 0.2, linewidth = 1.5, color = "black") +
  geom_point(data = plasmid_plot,
             aes(x = parent, y = efficiency, color = parent),
             size = 4, alpha = 0.5) +
  geom_hline(yintercept = 0.33e-9, linetype = "dashed", color = "black") +
  scale_y_log10(
    limits = c(1e-10, 1.5),
    breaks = c(1e-10, 1e-8, 1e-6, 1e-4, 1e-2, 1),
    labels = c(expression(10^-10), expression(10^-8), expression(10^-6),
               expression(10^-4), expression(10^-2), expression(10^0))) +
  labs(x = "Strain", y = "Conjugative Efficiency",
       title = "Plasmid Transfer Efficiency") +
  theme_minimal() +
  theme(legend.position = "none")
```

Tn7 transposition Load Tn7 data from spreadsheet (CFUs in LB vs LB+apramycin)

```
##      parent parent_sJMP rep Tn7  LB_Apr    LB
##   1:  17978       12014   1 GFP 1.0e+07 1e+10
##   2:  17978       12014   2 GFP 1.0e+07 3e+10
```

```
##  3:   17978       12014   3 GFP 1.0e+07 1e+10
##  4:   19606       12015   1 GFP 0.0e+00 3e+09
##  5:   19606       12015   2 GFP 0.0e+00 1e+09
##  6:   19606       12015   3 GFP 0.0e+00 2e+09
##  7: BW25113       12048   1 GFP 1.1e+09 1e+10
##  8: BW25113       12048   2 GFP 1.0e+09 8e+09
##  9: BW25113       12048   3 GFP 2.0e+09 1e+10
## 10:   17978       12014   1  EV 1.0e+07 2e+10
## 11:   17978       12014   2  EV 8.0e+06 1e+10
## 12:   17978       12014   3  EV 9.0e+06 3e+10
## 13:   19606       12015   1  EV 1.0e+05 2e+09
## 14:   19606       12015   2  EV 1.0e+05 9e+08
## 15:   19606       12015   3  EV 1.0e+05 8e+08
## 16: BW25113       12048   1  EV 2.0e+09 1e+10
## 17: BW25113       12048   2  EV 3.0e+09 7e+09
## 18: BW25113       12048   3  EV 2.0e+09 2e+10
```

Calculate efficiency if not already included

```
Tn7_eff$efficiency <- Tn7_eff$LB_Apr/Tn7_eff$LB
```

Filter for *A. baumannii* ATCC 17978 and *E. coli* BW25113 empty vector data
Calculate the mean efficiencies for each

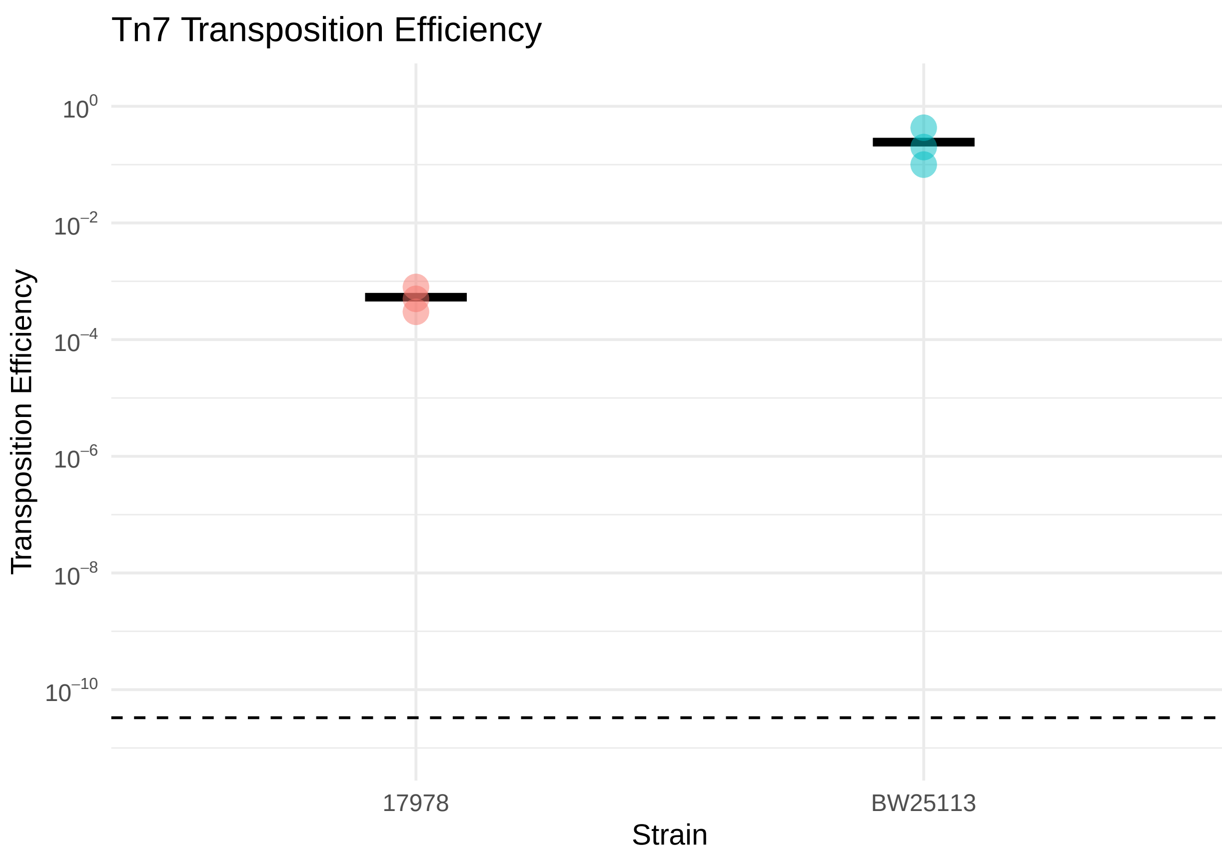
```
Tn7_plot <- Tn7_eff %>%
  filter(parent %in% c("17978", "BW25113"), Tn7 == "EV")

mean_efficiencies <- Tn7_plot %>%
  group_by(parent) %>%
  summarise(
    mean_efficiency = mean(efficiency),
    .groups = 'drop'
  )
```

```
## # A tibble: 2 x 2
##   parent  mean_efficiency
##   <chr>             <dbl>
## 1 17978          0.000533
## 2 BW25113        0.243
```

Plot a dot plot with the mean as a bar

```
ggplot() +
  geom_errorbar(data = mean_efficiencies,
                aes(x = parent, ymin = mean_efficiency, ymax = mean_efficiency),
                width = 0.2, linewidth = 1.5, color = "black") +
  geom_point(data = Tn7_plot, aes(x = parent, y = efficiency, color = parent),
             size = 4, alpha = 0.5) +
  geom_hline(yintercept = (0.33e-10), linetype = "dashed", color = "black") +
  scale_y_log10(
    limits = c(1e-11, 1.5),
    breaks = c(1e-10, 1e-8, 1e-6, 1e-4, 1e-2, 1),
```

```
    labels = c(expression(10^-10), expression(10^-8), expression(10^-6),
               expression(10^-4), expression(10^-2), expression(10^0))) +
  labs(x = "Strain", y = "Transposition Efficiency",
       title = "Tn7 Transposition Efficiency") +
  theme_minimal() +
  theme(legend.position = "none")
```



Promoter expression dot plots (Fig 2)

Load cleaned plate reader data

```
##     IPTG_conc rep  Ptrc.fluo    Ptrc.OD Pabst.fluo   Pabst.OD PabstBR.fluo
## 1           0   4   464.0000 0.19010000   357.0000 0.20250000     368.0000
## 2           0   5   526.0000 0.19689999   357.0000 0.19600000     376.0000
## 3           0   6   489.0000 0.19340000   350.0000 0.16630000     366.0000
## 4        1000   4 19671.0000 0.18780000   384.0000 0.19200000    8053.0000
## 5        1000   5 22029.0000 0.19270000   370.0000 0.18179999    6723.0000
## 6        1000   6 20529.0000 0.19320001   363.0000 0.17200001    8904.0000
## 7  background   6   330.1667 0.04271667   330.1667 0.04271667     330.1667
## 8          EV   4   368.0000 0.23729999   354.0000 0.22409999     344.0000
## 9          EV   5   362.0000 0.22820000   350.0000 0.19310000     374.0000
## 10         EV   6   352.0000 0.20850000   349.0000 0.22620000     337.0000
##    PabstBR.OD
## 1  0.19400001
```

```
## 2  0.19170000
## 3  0.19239999
## 4  0.19700000
## 5  0.18120000
## 6  0.19850001
## 7  0.04271667
## 8  0.19960000
## 9  0.24290000
## 10 0.21400000
```

Subtract background values from measurements

```r
columns_to_normalize <- c('Pabst.fluo', 'Pabst.OD', 'PabstBR.fluo',
                          'PabstBR.OD', 'Ptrc.fluo', 'Ptrc.OD')

background_vals <- IPTG_data %>% filter(IPTG_conc %like% "background") %>%
  select(all_of(columns_to_normalize))

for (col in columns_to_normalize) {
  IPTG_data[[col]] <- IPTG_data[[col]] - background_vals[[col]]
}

IPTG_data <- IPTG_data %>% filter(!(IPTG_conc %like% "background"))
```

Normalize to cell density (fluorescence/OD)

```r
fluo_columns <- grep("\\.fluo$", names(IPTG_data), value = TRUE)
od_columns <- sub("fluo", "OD", fluo_columns)

for (i in seq_along(fluo_columns)) {
  new_col_name <- paste0(sub("\\.fluo", "", fluo_columns[i]), ".Ratio")
  IPTG_data[[new_col_name]] <- IPTG_data[[fluo_columns[i]]] /
    IPTG_data[[od_columns[i]]]
}

data.norm <- IPTG_data %>%
  select(IPTG_conc, rep, ends_with(".Ratio"))
```

Subtract empty vector noise (autofluorescence)
Additionally calculate SD with propagated error

```r
#empty vector stats
ev_stats <- data.norm %>%
  filter(IPTG_conc == "EV") %>%
  summarise(across(ends_with(".Ratio"), list(mean = mean, sd = sd)))

#sample stats
mean_sd_diff <- data.norm %>%
  filter(IPTG_conc != "EV") %>%
  group_by(IPTG_conc) %>%
  summarise(across(ends_with(".Ratio"), list(mean = mean, sd = sd)))

#subtract off EV
```

```r
data_adjusted <- data.norm %>%
  filter(IPTG_conc != "EV") %>%
  rowwise() %>%
  mutate(
    Ptrc.Ratio = Ptrc.Ratio - ev_stats$Ptrc.Ratio_mean,
    Pabst.Ratio = Pabst.Ratio - ev_stats$Pabst.Ratio_mean,
    PabstBR.Ratio = PabstBR.Ratio - ev_stats$PabstBR.Ratio_mean
  )

#make data long format for ggplot downstream
data_adjusted_long <- data_adjusted %>%
  pivot_longer(
    cols = starts_with("p"),
    names_to = "promoter",
    values_to = "value"
)

mean_sd_adjusted <- data_adjusted %>%
  group_by(IPTG_conc) %>%
  summarise(across(ends_with(".Ratio"), list(mean = mean, sd = sd)))

#final stats (means and SD)
error_propagation <- mean_sd_adjusted %>%
  mutate(
    Ptrc.Ratio_PropagatedError = sqrt(mean_sd_diff$Ptrc.Ratio_sd^2 +
                                        ev_stats$Ptrc.Ratio_sd^2),
    Pabst.Ratio_PropagatedError = sqrt(mean_sd_diff$Pabst.Ratio_sd^2 +
                                         ev_stats$Pabst.Ratio_sd^2),
    PabstBR.Ratio_PropagatedError = sqrt(mean_sd_diff$PabstBR.Ratio_sd^2 +
                                           ev_stats$PabstBR.Ratio_sd^2)
  ) %>%
  select(IPTG_conc, ends_with("_mean"), ends_with("PropagatedError")) %>%
  pivot_longer(
    cols = -IPTG_conc,
    names_to = c("Sample", ".value"),
    names_pattern = "(.*)(mean|PropagatedError)$"
  ) %>%
  mutate(Sample = str_remove(Sample, "_"))

replicate_counts <- aggregate(rep ~ IPTG_conc, data = data.norm, FUN = length)
names(replicate_counts)[2] <- "Num_Replicates"

error_propagation <- merge(error_propagation, replicate_counts,
                           by = "IPTG_conc", all.x = TRUE)
```

```
##   IPTG_conc        Sample         mean PropagatedError Num_Replicates
## 1         0    Ptrc.Ratio    911.50954       184.90701              3
## 2         0   Pabst.Ratio     45.85076        18.26055              3
## 3         0 PabstBR.Ratio    150.02693        99.65630              3
## 4      1000    Ptrc.Ratio 137237.34193      6314.26419              3
## 5      1000   Pabst.Ratio    178.34965        57.17723              3
## 6      1000 PabstBR.Ratio  50303.08106      4448.91782              3
```

Prepare data for plotting

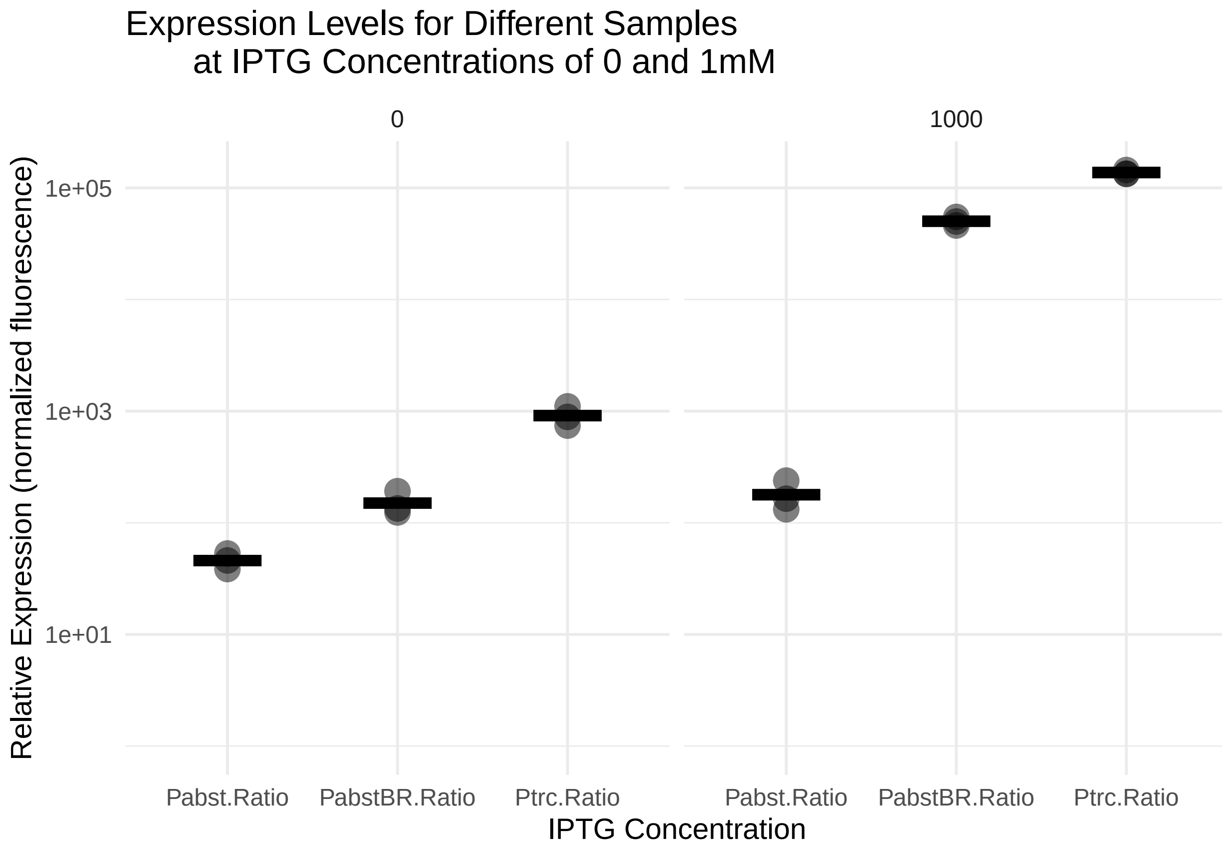
```
plot_data <- data_adjusted_long %>%
  group_by(IPTG_conc, promoter) %>%
  summarize(mean_value = mean(value))
```

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

```
ggplot() +
  geom_errorbar(data = plot_data, aes(x = as.factor(promoter),
                                      ymin = mean_value, ymax = mean_value),
                width = 0.4, size = 2, color = "black") +
  geom_point(data = data_adjusted_long,
             aes(x = as.factor(promoter), y = value), size = 4, alpha = 0.5) +
  facet_grid(. ~ IPTG_conc) +
  labs(x = "IPTG Concentration",
       y = "Relative Expression (normalized fluorescence)",
       title = "Expression Levels for Different Samples
       at IPTG Concentrations of 0 and 1mM") +
  theme_minimal() +
  theme(legend.position = "none") +
  scale_y_log10(limits = c(1, NA))
```

Plots

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Calculate statistics using propagated error as SD

```
# Separate the data by IPTG concentration
data_0 <- subset(error_propagation, IPTG_conc == 0)
data_1000 <- subset(error_propagation, IPTG_conc == 1000)

# Function to manually perform pairwise Welch's t-test
pairwise_t_test <- function(data) {
  combinations <- combn(unique(data$Sample), 2)
  results <- apply(combinations, 2, function(combo) {
    sample1 <- data[data$Sample == combo[1],]
    sample2 <- data[data$Sample == combo[2],]

    # Calculate t-statistic
    n1 <- sample1$Num_Replicates
    n2 <- sample2$Num_Replicates
    s1 <- sample1$PropagatedError
    s2 <- sample2$PropagatedError
    x1 <- sample1$mean
    x2 <- sample2$mean

    t_statistic <- (x1 - x2) / sqrt(s1^2 / n1 + s2^2 / n2)

    # Calculate degrees of freedom
    df <- ((s1^2 / n1 + s2^2 / n2)^2) /
      ((s1^2 / n1)^2 / (n1 - 1) + (s2^2 / n2)^2 / (n2 - 1))
```

```r
    # Determine the p-value
    p_value <- 2 * pt(-abs(t_statistic), df)

    return(list(samples = paste(combo, collapse = " vs "),
                t_statistic = t_statistic, p.value = p_value, df = df))
  })
  return(do.call(rbind, results))
}

# Perform the tests for both IPTG concentrations
results_0 <- pairwise_t_test(data_0)
results_1000 <- pairwise_t_test(data_1000)


# Function to perform paired t-test
paired_t_test <- function(data) {
  # Extract unique samples
  unique_samples <- unique(data$Sample)

  results <- data.frame(Sample = character(), mean_difference = numeric(),
                        t_statistic = numeric(),
                        p_value = numeric(),
                        df = numeric(), stringsAsFactors = FALSE)

  for (sample in unique_samples) {
    # Filter data for this sample at both IPTG concentrations
    sample_data <- subset(data, Sample == sample)

    if (nrow(sample_data) == 2) { # Ensure we have both IPTG = 0 and IPTG = 1000
      # Extract data for IPTG = 0 and IPTG = 1000
      d0 <- subset(sample_data, IPTG_conc == 0)
      d1000 <- subset(sample_data, IPTG_conc == 1000)

      # Calculate mean difference
      mean_diff <- d0$mean - d1000$mean

      # Calculate the combined standard error of the difference
      se_diff <- sqrt(d0$PropagatedError^2 + d1000$PropagatedError^2)

      # Calculate t-statistic
      t_statistic <- mean_diff / se_diff

      # Number of pairs is given by the number of replicates
      #(minimum from both groups to ensure matching pairs)
      n <- min(d0$Num_Replicates, d1000$Num_Replicates)

      # Degrees of freedom: n - 1
      df <- n - 1

      # Calculate p-value
      if (df > 0) {
        p_value <- 2 * pt(-abs(t_statistic), df)
      } else {
```

```r
        p_value <- NA  # Not defined for df <= 0
      }

      # Append results
      results <- rbind(results, data.frame(Sample = sample,
                                           mean_difference = mean_diff,
                                           t_statistic = t_statistic,
                                           p_value = p_value, df = df))
    }
  }

  return(results)
}

# Perform the paired t-test
paired_results <- paired_t_test(error_propagation)
```

Final stats tests with p-values Within promoter (0 vs 1 mM IPTG), paired t-test results:

```
##          Sample mean_difference t_statistic     p_value df
## 1    Ptrc.Ratio    -136325.8324  -21.580886 0.002140255  2
## 2   Pabst.Ratio       -132.4989   -2.207492 0.157974350  2
## 3 PabstBR.Ratio     -50153.0541  -11.270264 0.007781079  2
```

Welch's t-tests for promoters with no induction (0mM IPTG):

```
##      samples                          t_statistic p.value     df
## [1,] "Ptrc.Ratio vs Pabst.Ratio"    8.069497    0.01418483  2.039007
## [2,] "Ptrc.Ratio vs PabstBR.Ratio"  6.279038    0.007602443 3.071479
## [3,] "Pabst.Ratio vs PabstBR.Ratio" -1.780956   0.208899    2.134149
```

Welch's t-tests for promoters at full induction (1mM IPTG):

```
##      samples                          t_statistic p.value      df
## [1,] "Ptrc.Ratio vs Pabst.Ratio"    37.59479    0.0007060907 2.000328
## [2,] "Ptrc.Ratio vs PabstBR.Ratio"  19.49395    8.943107e-05 3.593121
## [3,] "Pabst.Ratio vs PabstBR.Ratio" -19.51293   0.002612054  2.000661
```

Titration of sfGFP expression across IPTG concentrations (Fig 3/S2)

Plasmid expression Load cleaned data from plate reader

```
##             IPTG_conc rep X19606.fluo  X19606.OD  X5075.fluo   X5075.OD
## 1                  EV   1   205.00000 0.45980001   247.00000 0.76340002
## 2                   0   1   232.00000 0.51440001   736.00000 0.66270000
## 3           0.0078125   1   244.00000 0.49470001   707.00000 0.58910000
## 4            0.015625   1   359.00000 0.53390002  1100.00000 0.56540000
## 5             0.03125   1   939.00000 0.56720001  1919.00000 0.54939997
## 6              0.0625   1  2389.00000 0.55500001  3527.00000 0.55309999
## 7               0.125   1  6302.00000 0.57510001  5183.00000 0.50599998
```

```
## 8                0.25   1  9407.00000 0.59670001 12190.00000 0.55750000
## 9                 0.5   1 20857.00000 0.59090000  7728.00000 0.51510000
## 10                  1   1 18039.00000 0.58469999  8171.00000 0.53960001
## 11                 EV   2   222.00000 0.52160001   249.00000 0.82279998
## 12                  0   2   238.00000 0.47420001   482.00000 0.75709999
## 13          0.0078125   2   297.00000 0.57139999   526.00000 0.80010003
## 14           0.015625   2   399.00000 0.54820001   726.00000 0.73180002
## 15            0.03125   2  1773.00000 0.54229999  1662.00000 0.77749997
## 16             0.0625   2  4175.00000 0.56720001  3178.00000 0.70550001
## 17              0.125   2  8631.00000 0.61369997  6807.00000 0.77120000
## 18               0.25   2 12180.00000 0.60699999  9006.00000 0.72060001
## 19                0.5   2 21647.00000 0.56770003  9972.00000 0.76929998
## 20                  1   2 23531.00000 0.59460002 12185.00000 0.71880001
## 21                 EV   3   224.00000 0.54040003   261.00000 0.75540000
## 22                  0   3   232.00000 0.49460000   367.00000 0.72790003
## 23          0.0078125   3   258.00000 0.44749999   528.00000 0.79809999
## 24           0.015625   3   477.00000 0.53950000   635.00000 0.73310000
## 25            0.03125   3  1032.00000 0.58240002  1845.00000 0.76550001
## 26             0.0625   3  2739.00000 0.60369998  3691.00000 0.72469997
## 27              0.125   3  8633.00000 0.59549999  7224.00000 0.72109997
## 28               0.25   3 11384.00000 0.58789998 10317.00000 0.71800000
## 29                0.5   3 20453.00000 0.57929999  9401.00000 0.76249999
## 30                  1   3 22185.00000 0.58350003 10578.00000 0.71730000
## 31 background_average  36    98.55556 0.04419722    98.55556 0.04419722
##    X17978.fluo  X17978.OD   Eco.fluo     Eco.OD
## 1     244.0000 0.75290000   262.0000 0.65460002
## 2     228.0000 0.62779999   760.0000 0.54390001
## 3     426.0000 0.71509999   825.0000 0.55320001
## 4    1170.0000 0.67189997  1637.0000 0.60030001
## 5    2768.0000 0.71730000  3442.0000 0.55500001
## 6    4733.0000 0.71410000  8715.0000 0.58190000
## 7    8569.0000 0.69040000  9389.0000 0.57910001
## 8   10898.0000 0.66390002  9131.0000 0.53670001
## 9   12409.0000 0.65740001 10050.0000 0.56970000
## 10  14096.0000 0.72909999 10915.0000 0.61820000
## 11    259.0000 0.76889998   308.0000 0.62900001
## 12    271.0000 0.74750000   746.0000 0.60509998
## 13    504.0000 0.74269998   947.0000 0.57499999
## 14    876.0000 0.75099999  1409.0000 0.59270000
## 15   3352.0000 0.70029998  4132.0000 0.59820002
## 16   5357.0000 0.74839997  7152.0000 0.54189998
## 17   8501.0000 0.73400003  8968.0000 0.56110001
## 18  10724.0000 0.68779999 11506.0000 0.60200000
## 19  13815.0000 0.69260001 10263.0000 0.55559999
## 20  18665.0000 0.68940002 10679.0000 0.58780003
## 21    266.0000 0.82279998   315.0000 0.65030003
## 22    276.0000 0.76099998   838.0000 0.55729997
## 23    632.0000 0.66149998   676.0000 0.67180002
## 24    915.0000 0.71829999  1248.0000 0.55690002
## 25   2073.0000 0.74589998  5388.0000 0.58289999
## 26   5197.0000 0.77399999  6527.0000 0.54589999
## 27  10039.0000 0.70980000  8920.0000 0.60450000
## 28  14142.0000 0.64600003  9936.0000 0.56639999
## 29  14477.0000 0.65960002  9386.0000 0.62290001
```

```
## 30  11415.0000 0.75120002 11683.0000 0.58560002
## 31    100.2222 0.04309167   100.2222 0.04309167
```

Subtract background values and normalize to cell density (fluorescence/OD)

```r
columns_to_normalize <- c('X19606.fluo', 'X19606.OD', 'X5075.fluo', 'X5075.OD',
                          'X17978.fluo', 'X17978.OD', 'Eco.fluo', 'Eco.OD')

background_vals <- ind_exp_data %>% filter(IPTG_conc %like% "background") %>%
  select(all_of(columns_to_normalize))

# Subtracting background averages from each column
for (col in columns_to_normalize) {
  ind_exp_data[[col]] <- ind_exp_data[[col]] - background_vals[[col]]
}

# Remove the background_average row if no longer needed
ind_exp_data <- ind_exp_data %>% filter(!(IPTG_conc %like% "background"))

# Identifying pairs of columns for division
fluo_columns <- grep("\\.fluo$", names(ind_exp_data), value = TRUE)
od_columns <- sub("fluo", "OD", fluo_columns)

# Performing division and storing results in new columns
for (i in seq_along(fluo_columns)) {
  new_col_name <- paste0(sub("\\.fluo", "", fluo_columns[i]), ".Ratio")
  ind_exp_data[[new_col_name]] <- ind_exp_data[[fluo_columns[i]]] /
    ind_exp_data[[od_columns[i]]]
}

data.norm <- ind_exp_data %>%
  select(IPTG_conc, rep, ends_with(".Ratio"))
```

Subtract autofluorescence (empty vector)
Additionally, propagate error for standard deviation

```r
# Filter data for IPTG_conc = "EV" and calculate mean and SD
ev_stats <- data.norm %>%
  filter(IPTG_conc == "EV") %>%
  summarise(across(ends_with(".Ratio"), list(mean = mean, sd = sd)))

# Calculate mean and SD of the samples
mean_sd_diff <- data.norm %>%
  group_by(IPTG_conc) %>%
  summarise(across(ends_with(".Ratio"), list(mean = mean, sd = sd))) %>%
  filter(IPTG_conc != "EV")

# Subtract mean EV from other values in each .Ratio column
data_adjusted <- data.norm %>%
  filter(IPTG_conc != "EV") %>%
  rowwise() %>%
  mutate(
    X19606.Ratio = X19606.Ratio - ev_stats$X19606.Ratio_mean,
```

```r
    X5075.Ratio = X5075.Ratio - ev_stats$X5075.Ratio_mean,
    X17978.Ratio = X17978.Ratio - ev_stats$X17978.Ratio_mean,
    Eco.Ratio = Eco.Ratio - ev_stats$Eco.Ratio_mean
  )

mean_sd_adjusted <- data_adjusted %>%
  group_by(IPTG_conc) %>%
  summarise(across(ends_with(".Ratio"), list(mean = mean, sd = sd)))

# Propagate error (assuming errors are uncorrelated)
error_propagation <- mean_sd_adjusted %>%
  mutate(
    X19606.Ratio_PropagatedError = sqrt(mean_sd_diff$X19606.Ratio_sd^2 +
                                        ev_stats$X19606.Ratio_sd^2),
    X5075.Ratio_PropagatedError = sqrt(mean_sd_diff$X5075.Ratio_sd^2 +
                                       ev_stats$X5075.Ratio_sd^2),
    X17978.Ratio_PropagatedError = sqrt(mean_sd_diff$X17978.Ratio_sd^2 +
                                        ev_stats$X17978.Ratio_sd^2),
    Eco.Ratio_PropagatedError = sqrt(mean_sd_diff$Eco.Ratio_sd^2 +
                                     ev_stats$Eco.Ratio_sd^2)
  ) %>%
  select(IPTG_conc, ends_with("_mean"), ends_with("PropagatedError")) %>%
  pivot_longer(
    cols = -IPTG_conc,
    names_to = c("Sample", ".value"),
    names_pattern = "(.*)(mean|PropagatedError)$"
  ) %>%
  mutate(Sample = str_remove(Sample, "_"))
```

```
## # A tibble: 36 x 4
##    IPTG_conc Sample        mean PropagatedError
##    <chr>     <chr>        <dbl>           <dbl>
##  1 0         X19606.Ratio  45.6            20.9
##  2 0         X5075.Ratio  444.            335.
##  3 0         X17978.Ratio  24.0            16.7
##  4 0         Eco.Ratio    976.            153.
##  5 0.0078125 X19606.Ratio 109.             37.7
##  6 0.0078125 X5075.Ratio  541.            318.
##  7 0.0078125 X17978.Ratio 429.            196.
##  8 0.0078125 Eco.Ratio    985.            355.
##  9 0.015625  X19606.Ratio 375.            120.
## 10 0.015625  X5075.Ratio  995.            625.
## # i 26 more rows
```

Plot expression with IPTG concentration on a semilog scale

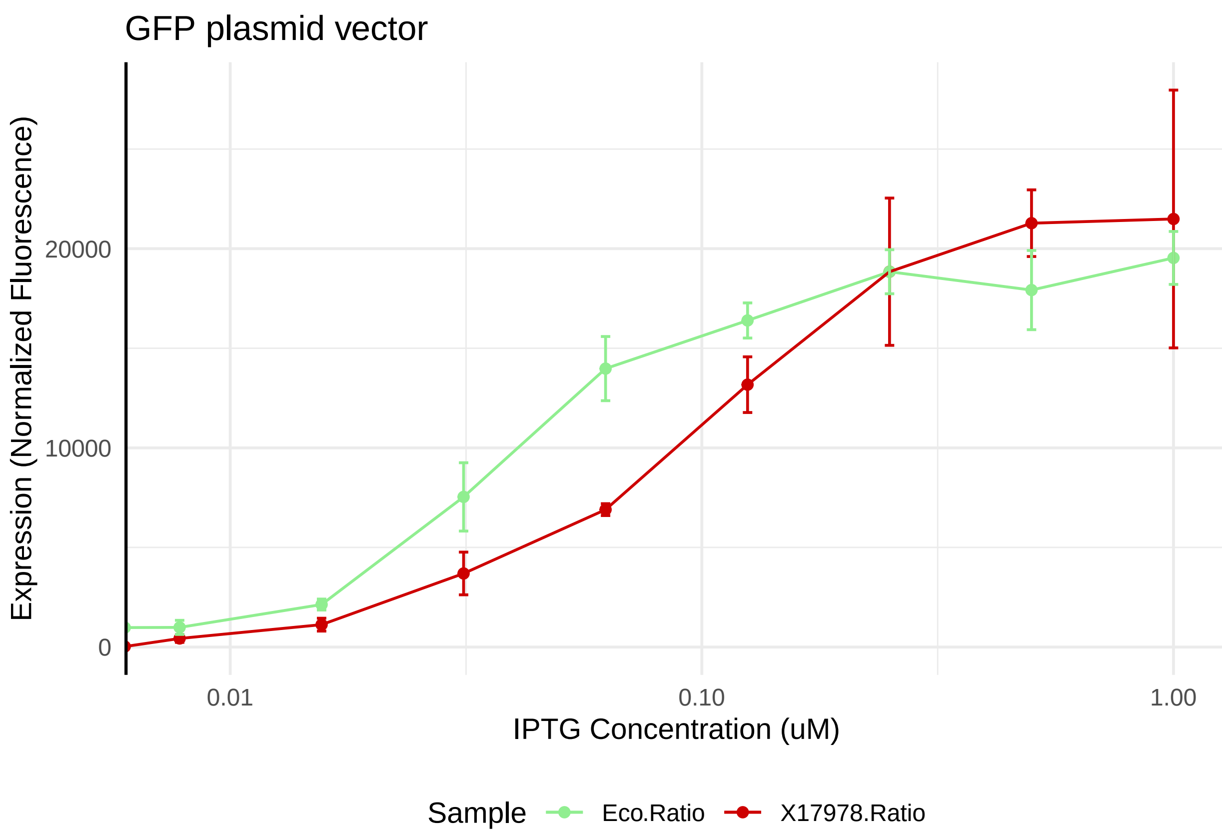
```r
# Select for strains and determine appropriate limits for the axes
mean_sd_data <- error_propagation %>%
  filter(Sample %like% "17978" | Sample %like% "Eco")
```

```r
mean_sd_data$IPTG_conc <- as.numeric(mean_sd_data$IPTG_conc)

x_limits <- range(mean_sd_data$IPTG_conc, na.rm = TRUE)
y_limits <- mean_sd_data %>%
  mutate(
    Lower = pmin(0, mean - PropagatedError),
    Upper = mean + PropagatedError
  ) %>%
  summarise(
    Min = min(Lower, na.rm = TRUE),
    Max = max(Upper, na.rm = TRUE)
  ) %>%
  unlist()
```

```r
ggplot(mean_sd_data, aes(x = IPTG_conc, y = mean, group = Sample)) +
  geom_point(aes(color = Sample)) +
  geom_line(aes(color = Sample)) +
  geom_errorbar(aes(ymin = mean - PropagatedError,
                    ymax = mean + PropagatedError, color = Sample),
                width = 0.02) +
  geom_vline(xintercept = 0, color = "black", size = 1) +
  labs(x = "IPTG Concentration (uM)",
       y = "Expression (Normalized Fluorescence)",
       title = "GFP plasmid vector") +
  theme_minimal() +
  scale_x_log10() +
  scale_y_continuous(limits = y_limits)  +
  scale_color_manual(values = c("Eco.Ratio" = "lightgreen", "X17978.Ratio" = "red3"))+
  theme(legend.position = "bottom")
```

A. baumannii ATCC 17978 and *E. coli* BW25113

```
## Warning in scale_x_log10(): log-10 transformation introduced infinite values.
## log-10 transformation introduced infinite values.
## log-10 transformation introduced infinite values.
## log-10 transformation introduced infinite values.
```



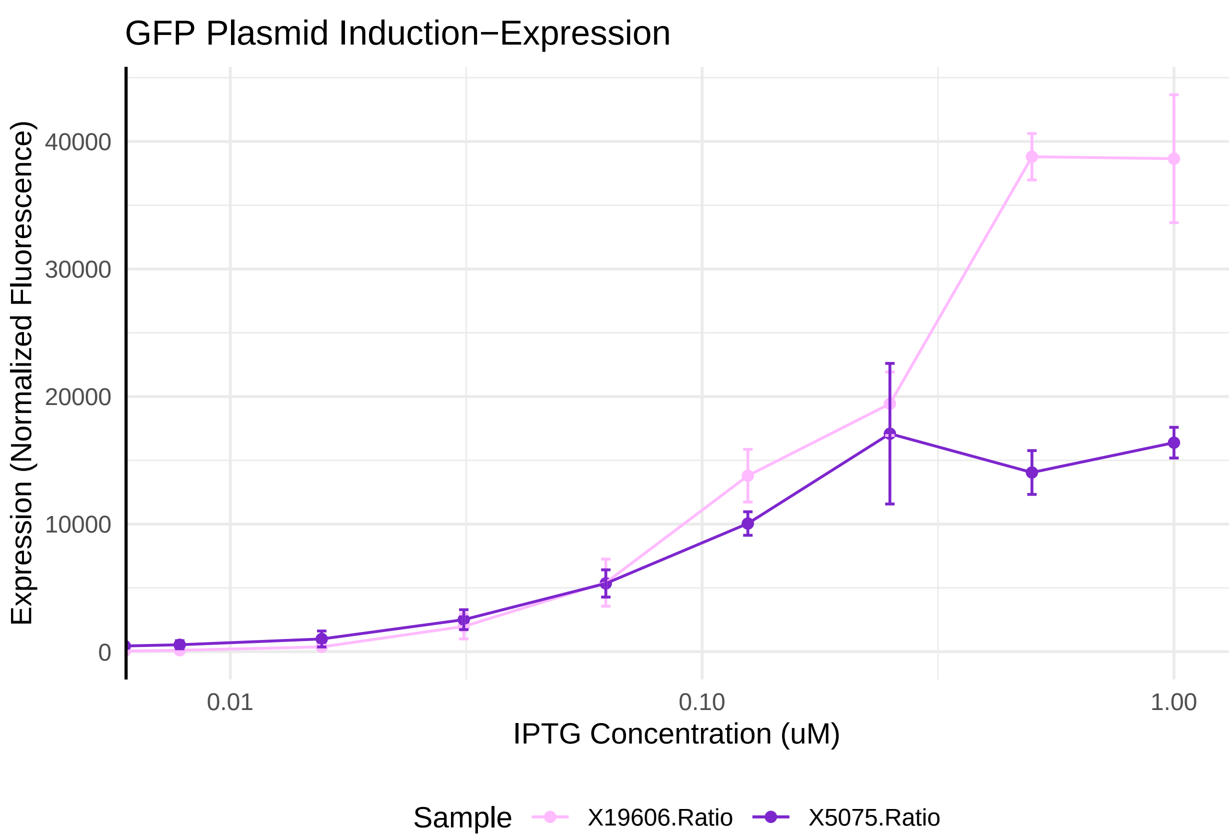

```
# Select for strains and determine appropriate limits for the axes
mean_sd_data <- error_propagation %>%
  filter(Sample %like% "5075" | Sample %like% "19606")

mean_sd_data$IPTG_conc <- as.numeric(mean_sd_data$IPTG_conc)

x_limits <- range(mean_sd_data$IPTG_conc, na.rm = TRUE)
y_limits <- mean_sd_data %>%
  mutate(
    Lower = pmin(0, mean - PropagatedError),
    Upper = mean + PropagatedError
  ) %>%
  summarise(
    Min = min(Lower, na.rm = TRUE),
    Max = max(Upper, na.rm = TRUE)
  ) %>%
  unlist()
```

```
ggplot(mean_sd_data, aes(x = IPTG_conc, y = mean, group = Sample)) +
  geom_point(aes(color = Sample)) +
  geom_line(aes(color = Sample)) +
```

```r
  geom_errorbar(aes(ymin = mean - PropagatedError,
                    ymax = mean + PropagatedError,
                    color = Sample), width = 0.02) +
  geom_vline(xintercept = 0, color = "black", size = 1) +
  labs(x = "IPTG Concentration (uM)",
       y = "Expression (Normalized Fluorescence)",
       title = "GFP Plasmid Induction-Expression") +
  theme_minimal() +
  scale_x_log10() +
  scale_y_continuous(limits = y_limits)  +
  scale_color_manual(values = c("X19606.Ratio" = "plum1", "X5075.Ratio" = "purple3"))+
  theme(legend.position = "bottom")
```

A. baumannii ATCC 19606 and AB5075

```
## Warning in scale_x_log10(): log-10 transformation introduced infinite values.
## log-10 transformation introduced infinite values.
## log-10 transformation introduced infinite values.
## log-10 transformation introduced infinite values.
```



Tn7 transposon Expression Load cleaned data from plate reader

```
##      IPTG_conc rep 19606.fluo   19606.OD 17978.fluo   17978.OD  Eco.fluo
##   1: background  36   118.5278 0.04231111   273.2778 0.03526111  118.8611
```

```
## 2:        EV  1  308.0000 0.52160001  383.0000 0.66920000  501.0000
## 3:         0  1  325.0000 0.52170003  541.0000 0.72479999  699.0000
## 4:  0.0078125  1  507.0000 0.51120001  650.0000 0.53950000  678.0000
## 5:   0.015625  1  568.0000 0.44980001  813.0000 0.53070003  747.0000
## 6:    0.03125  1  607.0000 0.42760000 1135.0000 0.54500002  884.0000
## 7:     0.0625  1  698.0000 0.47589999 1056.0000 0.49550000  998.0000
## 8:      0.125  1  657.0000 0.41720000 1142.0000 0.50989997 1008.0000
## 9:       0.25  1  521.0000 0.33649999 1038.0000 0.47020000 1154.0000
## 10:       0.5  1  702.0000 0.40390000 1109.0000 0.45800000 1086.0000
## 11:         1  1 1004.0000 0.35339999 1187.0000 0.49329999 1124.0000
## 12:        EV  2  335.0000 0.53460002  411.0000 0.74269998  629.0000
## 13:         0  2  337.0000 0.50929999  565.0000 0.76480001  747.0000
## 14: 0.0078125  2  484.0000 0.47769999  877.0000 0.56379998  659.0000
## 15:  0.015625  2  617.0000 0.48120001 1101.0000 0.59079999  773.0000
## 16:   0.03125  2  674.0000 0.43320000 1392.0000 0.55390000  832.0000
## 17:    0.0625  2  791.0000 0.46959999 1386.0000 0.54149997  960.0000
## 18:     0.125  2  797.0000 0.45370001 1503.0000 0.54159999 1087.0000
## 19:      0.25  2  750.0000 0.42410001 1515.0000 0.54549998 1071.0000
## 20:       0.5  2  889.0000 0.42320001 1393.0000 0.51410002 1149.0000
## 21:         1  2  755.0000 0.34130001 1562.0000 0.57450002 1147.0000
## 22:        EV  3  322.0000 0.52060002  405.0000 0.74049997  534.0000
## 23:         0  3  343.0000 0.48449999  439.0000 0.63120002  788.0000
## 24: 0.0078125  3  448.0000 0.47850001  588.0000 0.51010001  698.0000
## 25:  0.015625  3  593.0000 0.47900000  692.0000 0.50129998  773.0000
## 26:   0.03125  3  957.0000 0.45699999  848.0000 0.47510001  855.0000
## 27:    0.0625  3  689.0000 0.45449999  904.0000 0.48030001  964.0000
## 28:     0.125  3  790.0000 0.44839999 1124.0000 0.50070000 1138.0000
## 29:      0.25  3  778.0000 0.42320001 1292.0000 0.51410002 1089.0000
## 30:       0.5  3  496.0000 0.29470000 1007.0000 0.42860001 1160.0000
## 31:         1  3  894.0000 0.36050001 1148.0000 0.50340003 1144.0000
## 32:        EV  4  279.0000 0.48240000  326.0000 0.55989999  455.0000
## 33:         0  4  343.0000 0.50480002  469.0000 0.66180003  715.0000
## 34: 0.0078125  4  432.0000 0.44700000  819.0000 0.54830003  629.0000
## 35:  0.015625  4  534.0000 0.44679999  964.0000 0.50900000  745.0000
## 36:   0.03125  4  690.0000 0.42070001 1258.0000 0.50550002  841.0000
## 37:    0.0625  4  725.0000 0.43880001 1277.0000 0.45330000  962.0000
## 38:     0.125  4  688.0000 0.39469999 1340.0000 0.43630001  988.0000
## 39:      0.25  4  798.0000 0.43959999 1361.0000 0.42930001 1076.0000
## 40:       0.5  4  689.0000 0.33570001 1198.0000 0.39770001 1084.0000
## 41:         1  4  889.0000 0.32699999 1167.0000 0.41310000 1113.0000
## 42:        EV  5  304.0000 0.51080000  413.0000 0.76040000  561.0000
## 43:         0  5  326.0000 0.47729999  500.0000 0.66850001  760.0000
## 44: 0.0078125  5  395.0000 0.41650000  838.0000 0.59609997  683.0000
## 45:  0.015625  5  533.0000 0.48390001 1038.0000 0.59740001  751.0000
## 46:   0.03125  5  693.0000 0.47389999 1203.0000 0.51719999  886.0000
## 47:    0.0625  5  738.0000 0.46000001 1377.0000 0.55059999  925.0000
## 48:     0.125  5  713.0000 0.40450001 1481.0000 0.55400002 1012.0000
## 49:      0.25  5  778.0000 0.43709999 1282.0000 0.45190000 1046.0000
## 50:       0.5  5  670.0000 0.32730001 1371.0000 0.49210000 1180.0000
## 51:         1  5 1024.0000 0.32249999 1347.0000 0.45710000 1062.0000
## 52:        EV  6  289.0000 0.53009999  426.0000 0.74370003  489.0000
## 53:         0  6  437.0000 0.51139998  405.0000 0.56910002  735.0000
## 54: 0.0078125  6  426.0000 0.45019999  661.0000 0.50849998  646.0000
## 55:  0.015625  6  549.0000 0.48500001 1004.0000 0.53280002  725.0000
```

```
## 56:    0.03125   6   692.0000 0.43869999  1222.0000 0.48800001  859.0000
## 57:     0.0625   6   704.0000 0.45199999  1875.0000 0.54149997  952.0000
## 58:      0.125   6   867.0000 0.43959999  1820.0000 0.47819999 1060.0000
## 59:       0.25   6   730.0000 0.40509999  1887.0000 0.46380001 1107.0000
## 60:        0.5   6   612.0000 0.32820001  2024.0000 0.48780000 1054.0000
## 61:          1   6   807.0000 0.37979999  2743.0000 0.66289997 1087.0000
##      IPTG_conc rep 19606.fluo   19606.OD 17978.fluo   17978.OD  Eco.fluo
##         Eco.OD 5075.fluo    5075.OD
##  1: 0.03978889  133.3611 0.03507778
##  2: 0.69610000  280.0000 0.71530002
##  3: 0.58749998  363.0000 0.72380000
##  4: 0.57690001  398.0000 0.65130001
##  5: 0.55989999  435.0000 0.67500001
##  6: 0.55970001  505.0000 0.69660002
##  7: 0.55839998  556.0000 0.68320000
##  8: 0.52310002  692.0000 0.66729999
##  9: 0.54329997  556.0000 0.65820003
## 10: 0.54710001  616.0000 0.65930003
## 11: 0.53880000  559.0000 0.63840002
## 12: 0.64260000  294.0000 0.71980000
## 13: 0.58130002  341.0000 0.74150002
## 14: 0.61600000  430.0000 0.65219998
## 15: 0.57740003  457.0000 0.65590000
## 16: 0.61619997  486.0000 0.68349999
## 17: 0.55030000  577.0000 0.69260001
## 18: 0.56129998  792.0000 0.64029998
## 19: 0.52670002  769.0000 0.67030001
## 20: 0.54159999  756.0000 0.65399998
## 21: 0.53799999  771.0000 0.68330002
## 22: 0.61510003  294.0000 0.70389998
## 23: 0.66170001  367.0000 0.70630002
## 24: 0.61420000  397.0000 0.62220001
## 25: 0.60219997  484.0000 0.63770002
## 26: 0.56809998  497.0000 0.67760003
## 27: 0.56000000  572.0000 0.67530000
## 28: 0.57400000  593.0000 0.67610002
## 29: 0.53520000  693.0000 0.64569998
## 30: 0.55750000  661.0000 0.64270002
## 31: 0.54689997  641.0000 0.62669998
## 32: 0.61849999  295.0000 0.79710001
## 33: 0.61540002  351.0000 0.67140001
## 34: 0.66579997  371.0000 0.71969998
## 35: 0.59130001  456.0000 0.64240003
## 36: 0.54250002  509.0000 0.69889998
## 37: 0.56720001  582.0000 0.69129997
## 38: 0.52130002  700.0000 0.66270000
## 39: 0.54040003  685.0000 0.68839997
## 40: 0.53750002  681.0000 0.71340001
## 41: 0.52350003  761.0000 0.65350002
## 42: 0.62970001  300.0000 0.69950002
## 43: 0.66790003  340.0000 0.71740001
## 44: 0.57749999  367.0000 0.70050001
## 45: 0.57849997  448.0000 0.64850003
## 46: 0.57370001  654.0000 0.66119999
```

```
## 47: 0.58600003  812.0000 0.71020001
## 48: 0.59310001 1147.0000 0.68769997
## 49: 0.56919998  892.0000 0.69059998
## 50: 0.53770000  933.0000 0.69050002
## 51: 0.52429998 1189.0000 0.67970002
## 52: 0.60119999  290.0000 0.70670003
## 53: 0.58160001  333.0000 0.67100000
## 54: 0.57779998  342.0000 0.67540002
## 55: 0.60079998  425.0000 0.67989999
## 56: 0.58960003  556.0000 0.64120001
## 57: 0.52920002  503.0000 0.67949998
## 58: 0.54339999  586.0000 0.72750002
## 59: 0.53060001  591.0000 0.69610000
## 60: 0.54960000  558.0000 0.68519998
## 61: 0.53140003  654.0000 0.69749999
##          Eco.OD 5075.fluo    5075.OD
```

Subtract background values and normalize to cell density (fluorescence/OD)

```
columns_to_normalize <- c('19606.fluo', '19606.OD', '17978.fluo', '17978.OD',
                          'Eco.fluo', 'Eco.OD','5075.fluo', '5075.OD')

# Getting the background average values
background_vals <- ind_exp_Tn7data %>% filter(IPTG_conc %like% "background") %>%
  select(all_of(columns_to_normalize))

# Subtracting background averages from each column
for (col in columns_to_normalize) {
  ind_exp_Tn7data[[col]] <- ind_exp_Tn7data[[col]] - background_vals[[col]]
}

# Remove the background_average row if no longer needed
ind_exp_Tn7data <- ind_exp_Tn7data %>% filter(!(IPTG_conc %like% "background"))

# Identifying pairs of columns for division
fluo_columns <- grep("\\.fluo$", names(ind_exp_Tn7data), value = TRUE)
od_columns <- sub("fluo", "OD", fluo_columns)

# Performing division and storing results in new columns
for (i in seq_along(fluo_columns)) {
  new_col_name <- paste0(sub("\\.fluo", "", fluo_columns[i]), ".Ratio")
  ind_exp_Tn7data[[new_col_name]] <- ind_exp_Tn7data[[fluo_columns[i]]] /
    ind_exp_Tn7data[[od_columns[i]]]
}

data.norm <- ind_exp_Tn7data %>%
  select(IPTG_conc, rep, ends_with(".Ratio"))
```

Subtract autofluorescence (empty vector) Additionally, propagate error for standard deviation

```
# Filter data for IPTG_conc = "EV" and calculate mean and SD
ev_stats <- data.norm %>%
  filter(IPTG_conc == "EV") %>%
```

```
  summarise(across(ends_with(".Ratio"), list(mean = mean, sd = sd)))

# Calculate mean and SD of the sample data
mean_sd_diff <- data.norm %>%
  group_by(IPTG_conc) %>%
  summarise(across(ends_with(".Ratio"), list(mean = mean, sd = sd))) %>%
  filter(IPTG_conc != "EV")

# Subtract mean EV from other values in each .Ratio column
data_adjusted <- data.norm %>%
  filter(IPTG_conc != "EV") %>%
  rowwise() %>%
  mutate(
    `19606.Ratio` = `19606.Ratio` - ev_stats$`19606.Ratio_mean`,
    `17978.Ratio` = `17978.Ratio` - ev_stats$`17978.Ratio_mean`,
    Eco.Ratio = Eco.Ratio - ev_stats$Eco.Ratio_mean,
    `5075.Ratio` = `5075.Ratio` - ev_stats$`5075.Ratio_mean`
  )

mean_sd_adjusted <- data_adjusted %>%
  group_by(IPTG_conc) %>%
  summarise(across(ends_with(".Ratio"), list(mean = mean, sd = sd)))

# Propagate error (assuming errors are uncorrelated)
error_propagation <- mean_sd_adjusted %>%
  mutate(
    `19606.Ratio_PropagatedError` = sqrt(mean_sd_diff$`19606.Ratio_sd`^2 +
                                         ev_stats$`19606.Ratio_sd`^2),
    `17978.Ratio_PropagatedError` = sqrt(mean_sd_diff$`17978.Ratio_sd`^2 +
                                         ev_stats$`17978.Ratio_sd`^2),
    Eco.Ratio_PropagatedError = sqrt(mean_sd_diff$Eco.Ratio_sd^2 +
                                     ev_stats$Eco.Ratio_sd^2),
    `5075.Ratio_PropagatedError` = sqrt(mean_sd_diff$`5075.Ratio_sd`^2 +
                                        ev_stats$`5075.Ratio_sd`^2)) %>%
  select(IPTG_conc, ends_with("_mean"), ends_with("PropagatedError")) %>%
  pivot_longer(
    cols = -IPTG_conc,
    names_to = c("Sample", ".value"),
    names_pattern = "(.*)(mean|PropagatedError)$"
  ) %>%
  mutate(Sample = str_remove(Sample, "_"))
```

```
## # A tibble: 36 x 4
##    IPTG_conc Sample        mean PropagatedError
##    <chr>     <chr>        <dbl>           <dbl>
##  1 0         19606.Ratio  113.             94.0
##  2 0         17978.Ratio  153.             73.5
##  3 0         Eco.Ratio    391.            117.
##  4 0         5075.Ratio    91.3            26.5
##  5 0.0078125 19606.Ratio  387.             54.4
##  6 0.0078125 17978.Ratio  730.            195.
##  7 0.0078125 Eco.Ratio    282.            135.
##  8 0.0078125 5075.Ratio   166.             65.6
```

```
##  9 0.015625  19606.Ratio  649.            85.0
## 10 0.015625  17978.Ratio 1117.           248.
## # i 26 more rows
```

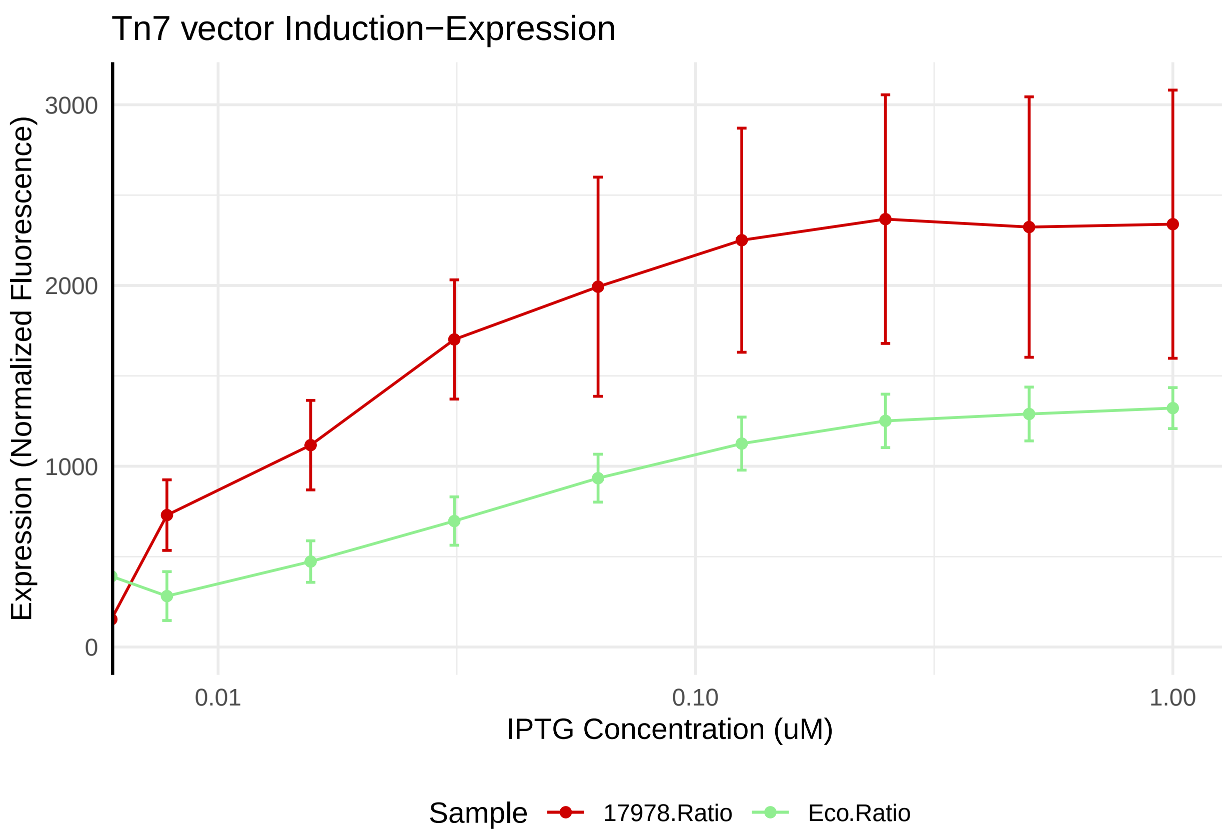
Plot expression levels across inducer (on semilog axis)

```r
# Select for strains and determine appropriate limits for the axes
mean_sd_data <- error_propagation %>%
  filter(Sample %like% "17978" | Sample %like% "Eco")

mean_sd_data$IPTG_conc <- as.numeric(mean_sd_data$IPTG_conc)

x_limits <- range(mean_sd_data$IPTG_conc, na.rm = TRUE)
y_limits <- mean_sd_data %>%
  mutate(
    Lower = pmin(0, mean - PropagatedError),
    Upper = mean + PropagatedError
  ) %>%
  summarise(
    Min = min(Lower, na.rm = TRUE),
    Max = max(Upper, na.rm = TRUE)
  ) %>%
  unlist()
```

```r
ggplot(mean_sd_data, aes(x = IPTG_conc, y = mean, group = Sample)) +
  geom_point(aes(color = Sample)) +
  geom_line(aes(color = Sample)) +
  geom_errorbar(aes(ymin = mean - PropagatedError,
                    ymax = mean + PropagatedError,
                    color = Sample), width = 0.02) +
  geom_vline(xintercept = 0, color = "black", linewidth = 1) +
  labs(x = "IPTG Concentration (uM)",
       y = "Expression (Normalized Fluorescence)",
       title = "Tn7 vector Induction-Expression") +
  theme_minimal() +
  scale_x_log10() +
  scale_y_continuous(limits = y_limits)  +
  scale_color_manual(values = c("Eco.Ratio" = "lightgreen",
                                "17978.Ratio" = "red3"))+
  theme(legend.position = "bottom")
```

A. baumannii ATCC 17978 and *E. coli* BW25113

```
## Warning in scale_x_log10(): log-10 transformation introduced infinite values.
## log-10 transformation introduced infinite values.
## log-10 transformation introduced infinite values.
## log-10 transformation introduced infinite values.
```




```r
# Select for strains and determine appropriate limits for the axes
mean_sd_data <- error_propagation %>%
  filter(Sample %like% "19606" | Sample %like% "5075")

mean_sd_data$IPTG_conc <- as.numeric(mean_sd_data$IPTG_conc)

x_limits <- range(mean_sd_data$IPTG_conc, na.rm = TRUE)
y_limits <- mean_sd_data %>%
  mutate(
    Lower = pmin(0, mean - PropagatedError),
    Upper = mean + PropagatedError
  ) %>%
  summarise(
    Min = min(Lower, na.rm = TRUE),
    Max = max(Upper, na.rm = TRUE)
  ) %>%
  unlist()
```

```r
ggplot(mean_sd_data, aes(x = IPTG_conc, y = mean, group = Sample)) +
  geom_point(aes(color = Sample)) +
  geom_line(aes(color = Sample)) +
```

```
  geom_errorbar(aes(ymin = mean - PropagatedError,
                    ymax = mean + PropagatedError,
                    color = Sample), width = 0.02) +
  geom_vline(xintercept = 0, color = "black", linewidth = 1) +
  labs(x = "IPTG Concentration (uM)",
       y = "Expression (Normalized Fluorescence)",
       title = "Tn7 vector Induction-Expression") +
  theme_minimal() +
  scale_x_log10() +
  scale_y_continuous(limits = y_limits)  +
  scale_color_manual(values = c("19606.Ratio" = "plum1",
                               "5075.Ratio" = "purple3"))+
  theme(legend.position = "bottom")
```

A. baumannii ATCC 19606 and AB5075

```
## Warning in scale_x_log10(): log-10 transformation introduced infinite values.
## log-10 transformation introduced infinite values.
## log-10 transformation introduced infinite values.
## log-10 transformation introduced infinite values.
```