

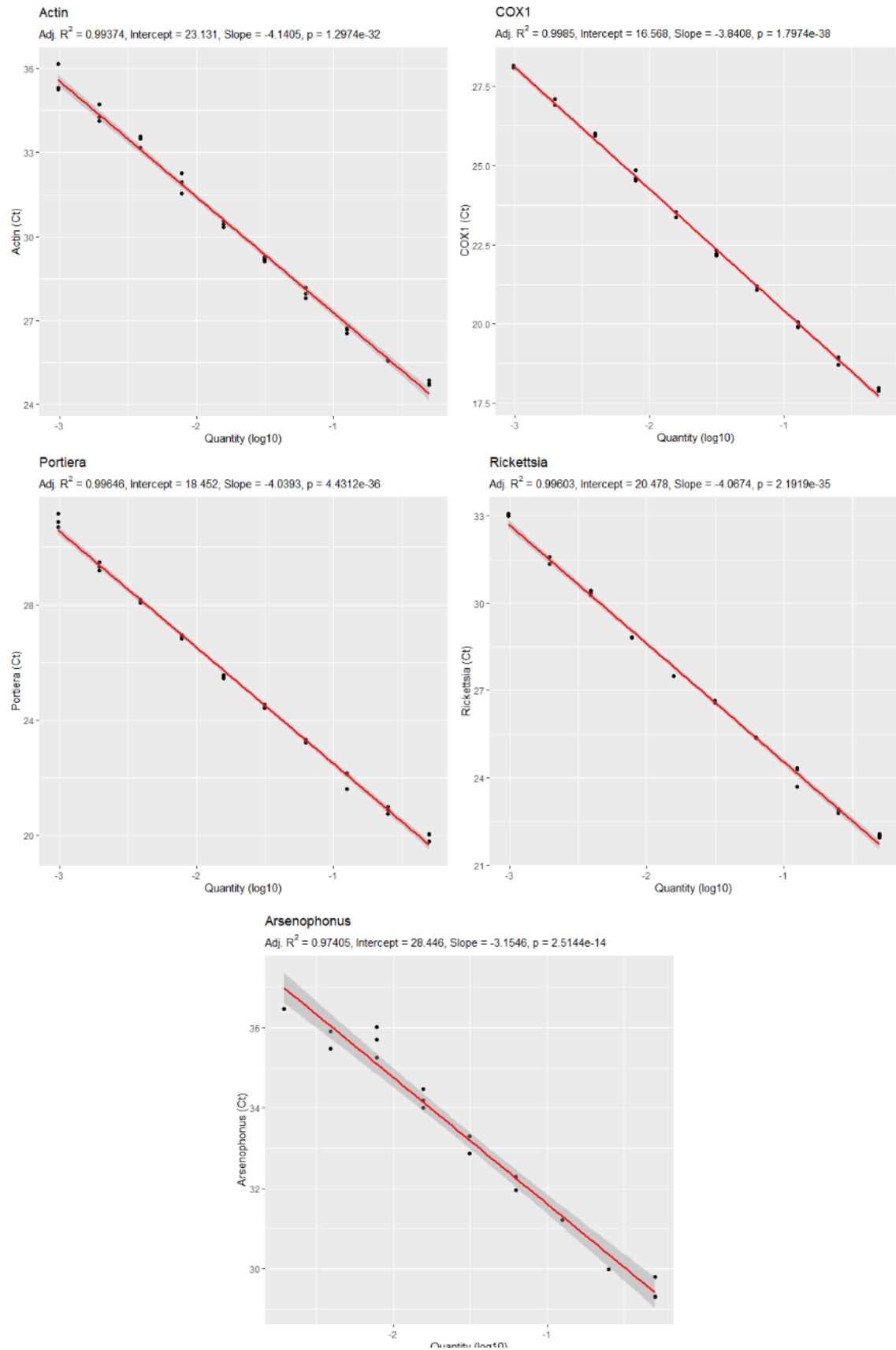


Figure S1 Fitted standard curves used for relative quantification of whitefly endosymbionts, nuclear, and mitochondrial DNA content in qPCR assays along with their corresponding slopes, intercepts, R^2 , and p-values.

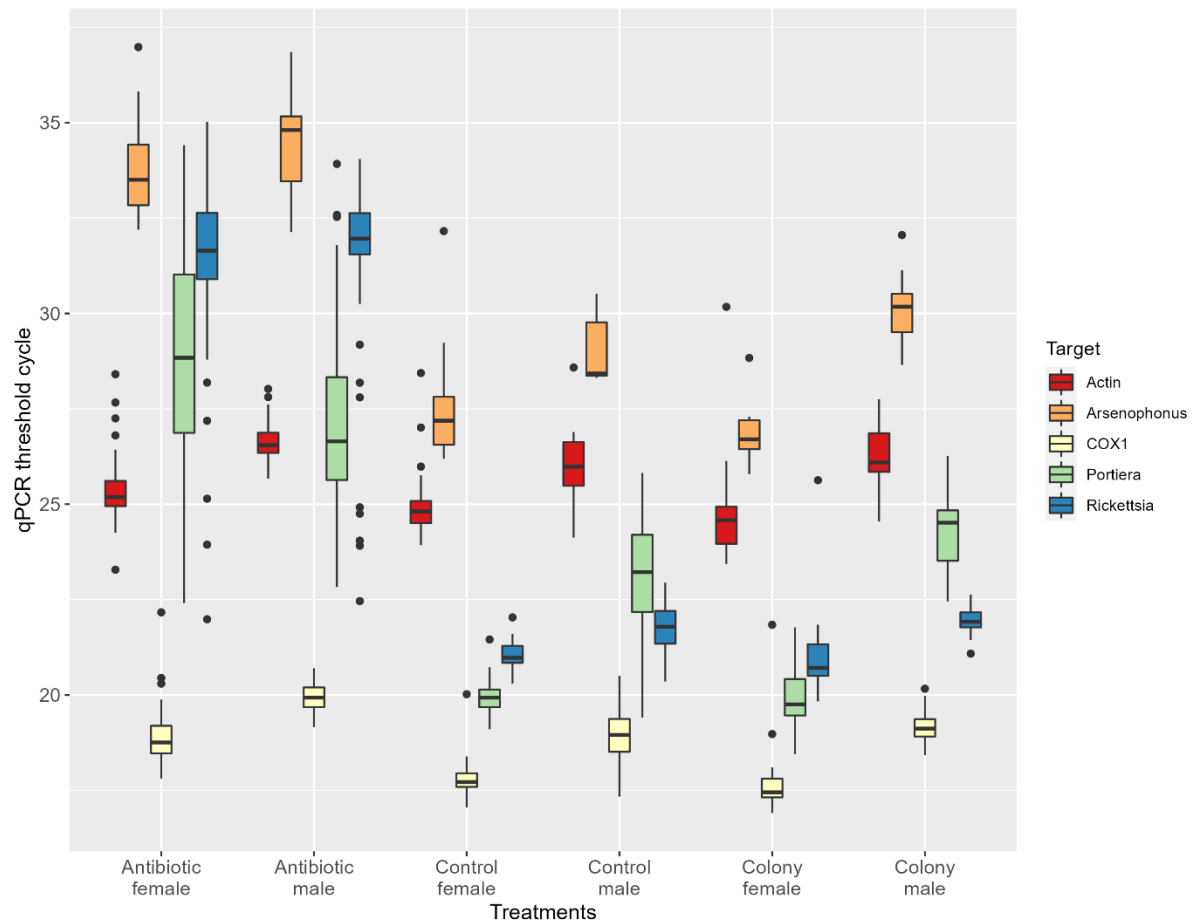


Figure S2. Raw Ct values for all samples across the treatment groups. Boxes represent the data between 25th and 75th percentile, horizontal line within the box represents median, and dots represent the outliers. A data point was considered an outlier as per the standard definition of the R package ggplot2 (when the data point (x) is either lower than $Q1 - 1.5 * IQR$ (interquartile range) or greater than $Q3 + 1.5 * IQR$). Vertical lines extending the boxes (whiskers) are drawn using standard ggplot2 function and are calculated using the following formulas: upper whisker = $\min(\max(x), Q3 + 1.5 * IQR)$, lower whisker = $\max(\min(x), Q1 - 1.5 * IQR)$. $IQR = Q3 - Q1$. Quantiles are calculated according to the default type 7 definition in R.

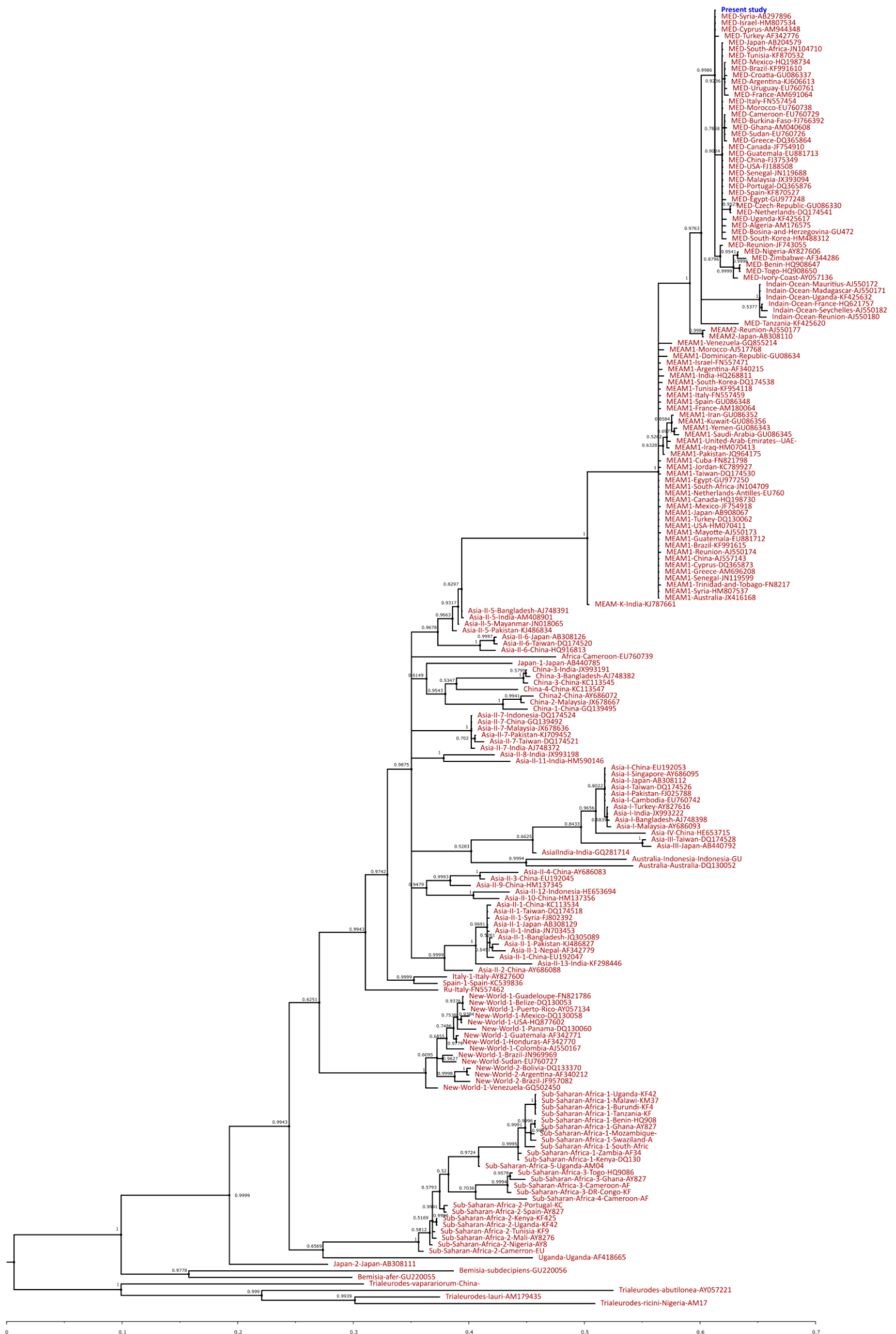


Figure S3: Phylogenetic tree of *Bemisia tabaci* reference mtCOI sequences (red) and the sequence from the present study (blue).

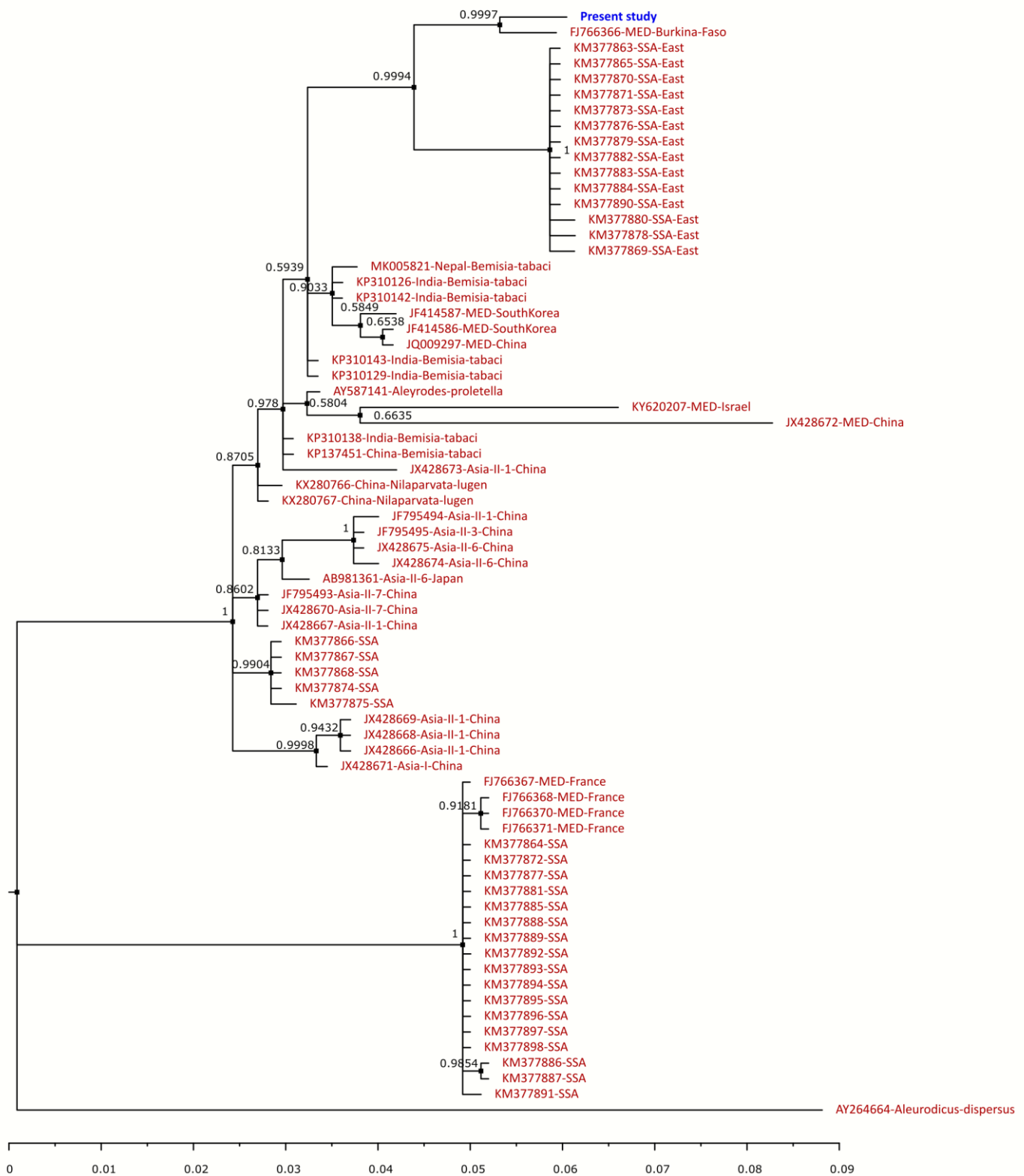


Figure S4: Phylogenetic tree of reference (red) *Arsenophonus* sequences from Kanakala and Ghanim¹¹ and top BLAST hits, and the present study (blue) based on partial 23S ribosomal RNA sequence.

Table S1: Counts of Arsenophonus positive and negative samples across treatment groups

Treatment	Positive	Negative	Total	Negative (%)
Antibiotic female	29	19	48	39.6%
Colony female	16	8	24	33.3%
Control female	17	7	24	29.2%
Antibiotic male	17	31	48	64.6%
Control male	5	11	16	68.8%
Colony male	12	12	24	50.0%

Table S2: Results of Fisher's exact test for Arsenophonus presence

Comparison	p.Fisher	p.adj.Fisher	Cramer.V
Antibiotic female : Antibiotic male	0.0241	0.3210	0.2500
Antibiotic female : Control female	0.4440	1.0000	0.1020
Antibiotic female : Control male	0.0808	0.8080	0.2530
Antibiotic female : Colony female	0.7970	1.0000	0.0609
Antibiotic female : Colony male	0.4550	1.0000	0.0992
Antibiotic male : Control female	0.0061	0.0909	0.3340
Antibiotic male : Control male	1.0000	1.0000	0.0380
Antibiotic male : Colony female	0.0229	0.3210	0.2960
Antibiotic male : Colony male	0.3090	1.0000	0.1400
Control female : Control male	0.0230	0.3210	0.3900
Control female : Colony female	1.0000	1.0000	0.0449
Control female : Colony male	0.2380	1.0000	0.2130
Control male : Colony female	0.0515	0.5660	0.3470
Control male : Colony male	0.3320	1.0000	0.1860
Colony female : Colony male	0.3800	1.0000	0.1690

```

1 ##### End of the R Script for plotting raw CT values #####
2
3 library(ggplot2)
4 library(cowplot)
5 library(tidyr)
6 library(plyr)
7 library(multcompView)
8 library(MASS)
9 library(readr)
10 library(naniar)
11
12 ## locate and read data
13 raw_data_sample <- read_csv("Data-CT_Plot.csv")
14 #View(raw_data_sample)
15
16
17 #Clean up the data
18 raw_data_sample$Group <- gsub('-', '_', raw_data_sample$Group)
19 raw_data_sample <- subset(raw_data_sample, Group != "NC" & Group != "PC_D1" & Group !=
20 "PC_D2" & Group != "PC_D3")
21 raw_data_sample <- raw_data_sample %>% replace_with_na(replace = list(Ct=40.00000))
22
23 p<-ggplot(raw_data_sample, aes(x=Group, y=Ct, fill=Target)) + scale_fill_manual(values
24 =c("#d7191c", "#fdae61", "#ffffbf", "#abdda4", "#2b83ba")) +
25   geom_boxplot(show.legend = TRUE) +
26   #ggtitle("Actin", subtitle = "Not normalized, transformed") +
27   xlab("Treatments") + ylab("qPCR threshold cycle") +
28   scale_x_discrete(labels = str_wrap(c("Antibiotic female", "Antibiotic male",
29 "Control female", "Control male", "Colony female", "Colony male"), width = 10)) +
30   theme(axis.text = element_text(size = 11)) +
31   theme(axis.title = element_text(size = 12))
32
33 p
34 ggsave(filename = "Raw-Ct.tiff", path="Paper/Graphs", width = 238, height = 185, units
35 ="mm", device='tiff', dpi=300, compression = "lzw")
36
37 ##### Beginning of the R Script for plotting raw CT values #####

```

```

1 ##### Beginning of the R Script for Actin data analysis #####
2
3 library(ggplot2)
4 library(cowplot)
5 library(tidyr)
6 library(plyr)
7 library(multcompView)
8 library(MASS)
9 library(readr)
10 library(naniar)
11 library(gridExtra)
12
13 ## locate and read data
14 raw_data_sample <- read_csv("Data_samples.csv")
15 #View(raw_data_sample)
16 raw_data_curves <- read_csv("Data_curves.csv")
17 #View(raw_data_curves)
18
19 #Clean up the data
20 raw_data_sample$Group <- gsub('-', '_', raw_data_sample$Group)
21 raw_data_sample <- subset(raw_data_sample, Group != "NC" & Group != "PC_D1" & Group !=
  "PC_D2" & Group != "PC_D3")
22
23 raw_data_sample <- raw_data_sample %>% replace_with_na(replace = list(Actin=40.00000))
24
25
26 #Function for plotting standard curves
27 ggplotRegression <- function (fit) {
28
29   require(ggplot2)
30
31   ggplot(fit$model, aes_string(x = names(fit$model)[2], y = names(fit$model)[1])) +
32     geom_point() +
33     stat_smooth(method = "lm", col = "red") +
34     ggtitle(paste("Standard curve for", names(fit$model)[1]),
35             subtitle = substitute(paste("Adj. R"^2, " = ", r,
36                                         ", Intercept = ", i,
37                                         ", Slope = ", s,
38                                         ", p = ", p),
39                                         list(r= signif(summary(fit)$adj.r.squared, 5), i=
40                                         signif(fit$coef[[1]], 5 ), s=signif(fit$coef[[2]], 5),
41                                         p=signif(summary(fit)$coef[2,4], 5))
42             ) + xlab(paste(names(fit$model)[2], "(log10)")) + ylab(paste(names(fit$
43             model)[1], "(Ct)"))
44   }
45
46 #Make standard curves, save models to "curves" and parameters to "curves_param", and
47 #plot the curves
48 curves_param <- data.frame()
49 curves <- list()
50 raw_data_curves_trans <- raw_data_curves
51 raw_data_curves_trans[, "Quantity"] = log10(raw_data_curves_trans[, "Quantity"])
52 for(i in names(raw_data_curves[, -c(2, 3)))[-1]){
53   curves[[i]] <- lm(paste(i, " ~ Quantity"), raw_data_curves_trans[, -c(2, 3)])
54   tmp <- data.frame(lapply(curves[[i]][["coefficients"]][1:2], type.convert),
55                     stringsAsFactors=FALSE)
56   tmp[, "rsquared"] = signif(summary(curves[[i]])$adj.r.squared, 5)
57   tmp[, "p"] = signif(summary(curves[[i]])$coef[2,4], 5)
58   names(tmp)[1] <- "intercept"
59   names(tmp)[2] <- "slope"
60   curves_param <- rbind(curves_param, cbind(Gene = i, tmp))
61   rm(tmp)
62   print(ggplotRegression(curves[[i]]))
63 }
64 rm(raw_data_curves_trans)
65
66 pcr_amount <- function(vec, a, b) {
67   res <- 10 ^ ((vec - a)/b)
68   return(res)
69 }

```

```

68 get_amounts <- function(df, intercept, slope) {
69
70   amounts <- mapply(function(d, a, b) pcr_amount(d, a, b),
71                     d = df, a = intercept, b = slope)
72   amounts <- as.data.frame(amounts)
73
74   return(amounts)
75 }
76
77
78 relative_quantities_perSample <- get_amounts(
79   raw_data_sample[3:7],
80   intercept = curves_param$intercept,
81   slope = curves_param$slope
82 )
83
84 not_normalized_perSample <- relative_quantities_perSample
85 not_normalized_perSample <- cbind(not_normalized_perSample, raw_data_sample[c("ID",
86   "Group")])
87
88 powerTransform <- function(y, lambda1, lambda2 = NULL, method = "boxcox") {
89
90   boxcoxTrans <- function(x, lam1, lam2 = NULL) {
91
92     # if we set lambda2 to zero, it becomes the one parameter transformation
93     lam2 <- ifelse(is.null(lam2), 0, lam2)
94
95     if (lam1 == 0L) {
96       log(y + lam2)
97     } else {
98       (((y + lam2)^lam1) - 1) / lam1
99     }
100   }
101
102   switch(method
103     , boxcox = boxcoxTrans(y, lambda1, lambda2)
104     , tukey = y^lambda1
105   )
106 }
107
108 # run the box-cox transformation to determine optimal lambda
109 bc_nn <- boxcox(not_normalized_perSample$Actin ~ not_normalized_perSample$Group)
110
111 #save and print lambda
112 (lambda_nn <- bc_nn$x[which.max(bc_nn$y)])
113
114 #BoxCox transform the data
115 not_normalized_perSample <- mutate(not_normalized_perSample, bcActin = powerTransform(
116   not_normalized_perSample$Actin, lambda_nn))
117
118 #ANOVA model using NOT normalized, transformed data
119 ANOVA_model_nn <- aov(not_normalized_perSample$bcActin~Group, data=
120   not_normalized_perSample)
121 par(mfrow = c(2, 2))
122 plot(ANOVA_model_nn)
123 summary(ANOVA_model_nn)
124 ANOVA_model_nn
125
126 #Post-hoc Tukey test
127 tHSD_nn <- TukeyHSD(ANOVA_model_nn, ordered = FALSE, conf.level = 0.95)
128 tHSD_nn
129
130 #Function to generate letter groupings for graphing
131 generate_label_df <- function(d, HSD, flev, colNo){
132   # Extract labels and factor levels from Tukey post-hoc
133   Tukey.levels <- HSD[[flev]][,4]
134   Tukey.labels <- multcompLetters(Tukey.levels)['Letters']
135   plot.labels <- names(Tukey.labels[['Letters']])
136
137   # Get highest quantile for Tukey's 5 number summary and add a bit of space to
138   # buffer between
139   # upper quantile and label placement
140   names(d)[colNo] <- "transformed"

```



```

137   boxplot.df <- ddply(d, flev, function(x) max(fivenum(x$transformed)) + 0.2)
138
139   # Create a data frame out of the factor levels and Tukey's homogenous group letters
140   plot.levels <- data.frame(plot.labels, labels = Tukey.labels[['Letters']],
141                             stringsAsFactors = FALSE)
142
143   # Merge it with the labels
144   labels.df <- merge(plot.levels, boxplot.df, by.x = 'plot.labels', by.y = flev, sort
145                     = FALSE)
146
147   return(labels.df)
148 }
149
150 not_normalized_perSample <- mutate(not_normalized_perSample, OsActin =
151 not_normalized_perSample$Actin)
152
153 library(tidyverse)
154 not_normalized_perSample <- not_normalized_perSample %>%
155   mutate(sex = ifelse(grepl("M$", Group), "Male", "Female"))
156
157 #Box plot NOT normalized data with Tukey letters, on transformed scale
158 p3<-ggplot(not_normalized_perSample, aes(x=Group, y=bcActin)) +
159   geom_boxplot(show.legend = FALSE , aes(fill=sex)) + scale_fill_manual(values=c(
160     "#bdbdbd", "#636363")) +
161   geom_text(data = generate_label_df(not_normalized_perSample, tHSD_nn, 'Group', 8),
162     aes(x = plot.labels, y = V1, label = labels)) +
163   ggtitle("Actin", subtitle = paste("Not normalized, transformed scale, Box Cox
164     lambda = ", round(lambda_nn, digits=3))) +
165   xlab("Treatments") + ylab("Relative quantity (transformed)") +
166   theme(axis.text = element_text(size = 11)) +
167   theme(axis.title = element_text(size = 12)) +
168   theme(plot.title = element_text(face="bold")) +
169   scale_x_discrete(labels = str_wrap(c("Antibiotic female", "Antibiotic male",
170     "Control female", "Control male", "Colony female", "Colony male"), width = 10))
171
172 p3
173 ggsave(filename = "Actin-NN-Ts.tiff", path="Paper/Graphs", width = 185, height = 185,
174 units="mm", device='tiff', dpi=300, compression = "lzw")
175
176 #Box plot NOT normalized data with Tukey letters, on original scale
177 p4<-ggplot(not_normalized_perSample, aes(x=Group, y=OsActin)) +
178   geom_boxplot(show.legend = FALSE , aes(fill=sex)) + scale_fill_manual(values=c(
179     "#bdbdbd", "#636363")) +
180   geom_text(data = generate_label_df(not_normalized_perSample, tHSD_nn, 'Group', 9),
181     aes(x = plot.labels, y = V1, label = labels)) +
182   ggtitle("Actin", subtitle = paste("Not normalized, original scale, Box Cox lambda =
183     ", round(lambda_nn, digits=3))) +
184   xlab("Treatments") + ylab("Relative quantity") +
185   theme(axis.text = element_text(size = 11)) +
186   theme(axis.title = element_text(size = 12)) +
187   theme(plot.title = element_text(face="bold")) +
188   scale_x_discrete(labels = str_wrap(c("Antibiotic female", "Antibiotic male",
189     "Control female", "Control male", "Colony female", "Colony male"), width = 10))
190
191 p4
192 ggsave(filename = "Actin-NN-Os.tiff", path="Paper/Graphs", width = 185, height = 185,
193 units="mm", device='tiff', dpi=300, compression = "lzw")
194
195 ##### End of the R Script for Actin data analysis #####

```

```

1 ##### Beginning of the R Script for COX1 data analysis #####
2
3 library(ggplot2)
4 library(cowplot)
5 library(tidyr)
6 library(plyr)
7 library(multcompView)
8 library(MASS)
9 library(readr)
10 library(naniar)
11 library(gridExtra)
12
13 ## locate and read data
14 raw_data_sample <- read_csv("Data_samples.csv")
15 #View(raw_data_sample)
16 raw_data_curves <- read_csv("Data_curves.csv")
17 #View(raw_data_curves)
18
19 #Clean up the data
20 raw_data_sample$Group <- gsub('-', '_', raw_data_sample$Group)
21 raw_data_sample <- subset(raw_data_sample, Group != "NC" & Group != "PC_D1" & Group !=
  "PC_D2" & Group != "PC_D3")
22
23 raw_data_sample <- raw_data_sample %>% replace_with_na(replace = list(COX1=40.00000))
24
25
26 #Function for plotting standard curves
27 ggplotRegression <- function (fit) {
28
29   require(ggplot2)
30
31   ggplot(fit$model, aes_string(x = names(fit$model)[2], y = names(fit$model)[1])) +
32     geom_point() +
33     stat_smooth(method = "lm", col = "red") +
34     ggtitle(paste("Standard curve for", names(fit$model)[1]),
35             subtitle = substitute(paste("Adj. R"^2, " = ", r,
36                                         ", Intercept = ", i,
37                                         ", Slope = ", s,
38                                         ", p = ", p),
39                                         list(r= signif(summary(fit)$adj.r.squared, 5), i=
40                                         signif(fit$coef[[1]], 5 ), s=signif(fit$coef[[2]], 5),
41                                         p=signif(summary(fit)$coef[2,4], 5))
42             ) + xlab(paste(names(fit$model)[2], "(log10)")) + ylab(paste(names(fit$
43             model)[1], "(Ct)"))
44   }
45
46 #Make standard curves, save models to "curves" and parameters to "curves_param", and
47 #plot the curves
48 curves_param <- data.frame()
49 curves <- list()
50 raw_data_curves_trans <- raw_data_curves
51 raw_data_curves_trans[, "Quantity"] = log10(raw_data_curves_trans[, "Quantity"])
52 for(i in names(raw_data_curves[, -c(2, 3)))[-1]){
53   curves[[i]] <- lm(paste(i, " ~ Quantity"), raw_data_curves_trans[, -c(2, 3)])
54   tmp <- data.frame(lapply(curves[[i]][["coefficients"]][1:2], type.convert),
55                     stringsAsFactors=FALSE)
56   tmp[, "rsquared"] = signif(summary(curves[[i]])$adj.r.squared, 5)
57   tmp[, "p"] = signif(summary(curves[[i]])$coef[2,4], 5)
58   names(tmp)[1] <- "intercept"
59   names(tmp)[2] <- "slope"
60   curves_param <- rbind(curves_param, cbind(Gene = i, tmp))
61   rm(tmp)
62   print(ggplotRegression(curves[[i]]))
63 }
64 rm(raw_data_curves_trans)
65
66 pcr_amount <- function(vec, a, b) {
67   res <- 10 ^ ((vec - a)/b)
68   return(res)
69 }

```

```

68 get_amounts <- function(df, intercept, slope) {
69
70   amounts <- mapply(function(d, a, b) pcr_amount(d, a, b),
71                     d = df, a = intercept, b = slope)
72   amounts <- as.data.frame(amounts)
73
74   return(amounts)
75 }
76
77
78 pcr_normalize <- function(vec, ref, mode) {
79   if (mode == 'subtract') {
80     res <- vec - ref
81   } else if (mode == 'divide') {
82     res <- vec / ref
83   } else {
84     stop("mode can be one of 'subtract' or 'divide'.")
85   }
86   return(res)
87 }
88
89
90 relative_quantities_perSample <- get_amounts(
91   raw_data_sample[3:7],
92   intercept = curves_param$intercept,
93   slope = curves_param$slope
94 )
95
96 normalized_perSample <- pcr_normalize(relative_quantities_perSample,
97   relative_quantities_perSample$Actin, mode = 'divide')
98 normalized_perSample <- cbind(normalized_perSample, raw_data_sample[c("ID", "Group")])
99
100 not_normalized_perSample <- relative_quantities_perSample
101 not_normalized_perSample <- cbind(not_normalized_perSample, raw_data_sample[c("ID",
102   "Group")])
103
104 #Comment next line to apply normalization
105 #normalized_perSample <- not_normalized_perSample
106
107 powerTransform <- function(y, lambda1, lambda2 = NULL, method = "boxcox") {
108
109   boxcoxTrans <- function(x, lam1, lam2 = NULL) {
110
111     # if we set lambda2 to zero, it becomes the one parameter transformation
112     lam2 <- ifelse(is.null(lam2), 0, lam2)
113
114     if (lam1 == 0L) {
115       log(y + lam2)
116     } else {
117       (((y + lam2)^lam1) - 1) / lam1
118     }
119   }
120
121   switch(method
122     , boxcox = boxcoxTrans(y, lambda1, lambda2)
123     , tukey = y^lambda1
124   )
125 }
126
127 # run the box-cox transformation to determine optimal lambda
128 bc <- boxcox(normalized_perSample$COX1 ~ normalized_perSample$Group)
129 bc_nn <- boxcox(not_normalized_perSample$COX1 ~ not_normalized_perSample$Group)
130
131 #save and print lambda
132 (lambda <- bc$x[which.max(bc$y)])
133 (lambda_nn <- bc_nn$x[which.max(bc_nn$y)])
134
135 #BoxCox transform the data
136 normalized_perSample <- mutate(normalized_perSample, bcCOX1 = powerTransform(
137   normalized_perSample$COX1, lambda))
138 not_normalized_perSample <- mutate(not_normalized_perSample, bcCOX1 = powerTransform(
139   not_normalized_perSample$COX1, lambda_nn))

```

```

137 #ANOVA model using normalized, transformed data
138 ANOVA_model_norm <- aov(normalized_perSample$bcCOX1~Group, data=normalized_perSample)
139 par(mfrow = c(2, 2))
140 plot(ANOVA_model_norm)
141 summary(ANOVA_model_norm)
142 ANOVA_model_norm
143
144 #ANOVA model using NOT normalized, transformed data
145 ANOVA_model_nn <- aov(not_normalized_perSample$bcCOX1~Group, data=
not_normalized_perSample)
146 par(mfrow = c(2, 2))
147 plot(ANOVA_model_nn)
148 summary(ANOVA_model_nn)
149 ANOVA_model_nn
150
151 #Post-hoc Tukey test
152 tHSD <- TukeyHSD(ANOVA_model_norm, ordered = FALSE, conf.level = 0.95)
153 tHSD
154 tHSD_nn <- TukeyHSD(ANOVA_model_nn, ordered = FALSE, conf.level = 0.95)
155 tHSD_nn
156
157 #Function to generate letter groupings for graphing
158 generate_label_df <- function(d, HSD, flev, colNo){
159   # Extract labels and factor levels from Tukey post-hoc
160   Tukey.levels <- HSD[[flev]][,4]
161   Tukey.labels <- multcompLetters(Tukey.levels)['Letters']
162   plot.labels <- names(Tukey.labels[['Letters']])
163
164   # Get highest quantile for Tukey's 5 number summary and add a bit of space to
buffer between
165   # upper quantile and label placement
166   names(d)[colNo] <- "transformed"
167   boxplot.df <- ddply(d, flev, function(x) max(fivenum(x$transformed)) + 0.2)
168
169   # Create a data frame out of the factor levels and Tukey's homogenous group letters
170   plot.levels <- data.frame(plot.labels, labels = Tukey.labels[['Letters']],
stringsAsFactors = FALSE)
171
172   # Merge it with the labels
173   labels.df <- merge(plot.levels, boxplot.df, by.x = 'plot.labels', by.y = flev, sort
= FALSE)
174
175   return(labels.df)
176 }
177
178 normalized_perSample <- mutate(normalized_perSample, OsCOX1 = normalized_perSample$
COX1)
179
180 not_normalized_perSample <- mutate(not_normalized_perSample, OsCOX1 =
not_normalized_perSample$COX1)
181
182 library(tidyverse)
183 not_normalized_perSample <- not_normalized_perSample %>%
184   mutate(sex = ifelse(grepl("M$", Group), "Male", "Female"))
185
186 normalized_perSample <- normalized_perSample %>%
187   mutate(sex = ifelse(grepl("M$", Group), "Male", "Female"))
188
189 #Box plot Actin normalized data with Tukey letters, on transformed scale
190 p1<-ggplot(normalized_perSample, aes(x=Group, y=bcCOX1)) +
191   geom_boxplot(show.legend = FALSE , aes(fill=sex)) + scale_fill_manual(values=c(
"#bdbdbd", "#636363")) +
192   geom_text(data = generate_label_df(normalized_perSample, tHSD, 'Group', 8), aes(x =
plot.labels, y = V1, label = labels)) +
193   ggtitle("COX1", subtitle = paste("Actin normalized, transformed scale, Box Cox
lambda = ", round(lambda, digits=3))) +
194   xlab("Treatments") + ylab("Relative quantity (transformed)") +
195   theme(axis.text = element_text(size = 11)) +
196   theme(axis.title = element_text(size = 12)) +
197   theme(plot.title = element_text(face="bold")) +
198   scale_x_discrete(labels = str_wrap(c("Antibiotic female", "Antibiotic male",
"Control female", "Control male", "Colony female", "Colony male"), width = 10))
199 p1
200 #ggsave(filename = "COX1-AN-Ts.tiff", path="Paper/original scale", width = 600,

```

```

201 height = 700, units="px", device='tiff', dpi=96)
202
203 #Box plot Actin normalized data with Tukey letters, on original scale
204 p2<-ggplot(normalized_perSample, aes(x=Group, y=OsCOX1)) +
205   geom_boxplot(show.legend = FALSE , aes(fill=sex)) + scale_fill_manual(values=c(
206     "#bdbdbd", "#636363")) +
207   geom_text(data = generate_label_df(normalized_perSample, tHSD, 'Group', 9), aes(x =
208     plot.labels, y = V1, label = labels)) +
209   ggtitle("COX1", subtitle = paste("Actin normalized, original scale, Box Cox lambda
210     = ", round(lambda, digits=3))) +
211   xlab("Treatments") + ylab("Relative quantity") +
212   theme(axis.text = element_text(size = 11)) +
213   theme(axis.title = element_text(size = 12)) +
214   theme(plot.title = element_text(face="bold")) +
215   scale_x_discrete(labels = str_wrap(c("Antibiotic female", "Antibiotic male",
216     "Control female", "Control male", "Colony female", "Colony male"), width = 10))
217 p2
218 #ggsave(filename = "COX1-AN-Os.tiff", path="Paper/original scale", width = 600,
219 height = 700, units="px", device='tiff', dpi=96)
220
221 #Box plot NOT normalized data with Tukey letters, on transformed scale
222 p3<-ggplot(not_normalized_perSample, aes(x=Group, y=bcCOX1)) +
223   geom_boxplot(show.legend = FALSE , aes(fill=sex)) + scale_fill_manual(values=c(
224     "#bdbdbd", "#636363")) +
225   geom_text(data = generate_label_df(not_normalized_perSample, tHSD_nn, 'Group', 8),
226     aes(x = plot.labels, y = V1, label = labels)) +
227   ggtitle("COX1", subtitle = paste("Not normalized, transformed scale, Box Cox lambda
228     = ", round(lambda_nn, digits=3))) +
229   xlab("Treatments") + ylab("Relative quantity (transformed)") +
230   theme(axis.text = element_text(size = 11)) +
231   theme(axis.title = element_text(size = 12)) +
232   theme(plot.title = element_text(face="bold")) +
233   scale_x_discrete(labels = str_wrap(c("Antibiotic female", "Antibiotic male",
234     "Control female", "Control male", "Colony female", "Colony male"), width = 10))
235 p3
236 #ggsave(filename = "COX1-NN-Ts.tiff", path="Paper/original scale", width = 600,
237 height = 700, units="px", device='tiff', dpi=96)
238
239 #Box plot NOT normalized data with Tukey letters, on original scale
240 p4<-ggplot(not_normalized_perSample, aes(x=Group, y=OsCOX1)) +
241   geom_boxplot(show.legend = FALSE , aes(fill=sex)) + scale_fill_manual(values=c(
242     "#bdbdbd", "#636363")) +
243   geom_text(data = generate_label_df(not_normalized_perSample, tHSD_nn, 'Group', 9),
244     aes(x = plot.labels, y = V1, label = labels)) +
245   ggtitle("COX1", subtitle = paste("Not normalized, original scale, Box Cox lambda = ",
246     round(lambda_nn, digits=3))) +
247   xlab("Treatments") + ylab("Relative quantity") +
248   theme(axis.text = element_text(size = 11)) +
249   theme(axis.title = element_text(size = 12)) +
250   theme(plot.title = element_text(face="bold")) +
251   scale_x_discrete(labels = str_wrap(c("Antibiotic female", "Antibiotic male",
252     "Control female", "Control male", "Colony female", "Colony male"), width = 10))
253 p4
254 #ggsave(filename = "COX1-NN-Os.tiff", path="Paper/original scale", width = 600,
255 height = 700, units="px", device='tiff', dpi=96)
256
257 ggsave(filename = "COX1-Ts.tiff", path="Paper/Graphs", arrangeGrob(p3, p1, nrow = 1),
258 width = 322, height = 185, units="mm", device='tiff', dpi=300, compression = "lzw")
259 ggsave(filename = "COX1-Os.tiff", path="Paper/Graphs", arrangeGrob(p4, p2, nrow = 1),
260 width = 322, height = 185, units="mm", device='tiff', dpi=300, compression = "lzw")
261
262 ##### End of the R Script for COX1 data analysis #####

```

```

1 ##### Beginning of the R Script for Portiera data analysis #####
2
3 library(ggplot2)
4 library(cowplot)
5 library(tidyr)
6 library(plyr)
7 library(multcompView)
8 library(MASS)
9 library(readr)
10 library(naniar)
11 library(gridExtra)
12
13 ## locate and read data
14 raw_data_sample <- read_csv("Data_samples.csv")
15 #View(raw_data_sample)
16 raw_data_curves <- read_csv("Data_curves.csv")
17 #View(raw_data_curves)
18
19 #Clean up the data
20 raw_data_sample$Group <- gsub('-', '_', raw_data_sample$Group)
21 raw_data_sample <- subset(raw_data_sample, Group != "NC" & Group != "PC_D1" & Group !=
  "PC_D2" & Group != "PC_D3")
22
23 raw_data_sample <- raw_data_sample %>% replace_with_na(replace = list(Portiera=
  40.00000))
24 raw_data_sample <- raw_data_sample %>% replace_with_na(replace = list(Portiera=
  40.00000))
25
26
27
28 #Function for plotting standard curves
29 ggplotRegression <- function(fit) {
30
31   require(ggplot2)
32
33   ggplot(fit$model, aes_string(x = names(fit$model)[2], y = names(fit$model)[1])) +
34     geom_point() +
35     stat_smooth(method = "lm", col = "red") +
36     ggtitle(paste("Standard curve for", names(fit$model)[1]),
37             subtitle = substitute(paste("Adj. R"^2, " = ", r,
38             ", Intercept = ", i,
39             ", Slope = ", s,
40             ", p = ", p),
41             list(r= signif(summary(fit)$adj.r.squared, 5), i=
42             signif(fit$coef[[1]], 5), s=signif(fit$coef[[2]], 5),
43             p=signif(summary(fit)$coef[2,4], 5))
44     ) + xlab(paste(names(fit$model)[2], "(log10)")) + ylab(paste(names(fit$
45     model)[1], "(Ct)"))
46   }
47
48 #Make standard curves, save models to "curves" and parameters to "curves_param", and
49 #plot the curves
50 curves_param <- data.frame()
51 curves <- list()
52 raw_data_curves_trans <- raw_data_curves
53 raw_data_curves_trans[, "Quantity"] = log10(raw_data_curves_trans[, "Quantity"])
54 for(i in names(raw_data_curves[, -c(2, 3)))[-1]){
55   curves[[i]] <- lm(paste(i, " ~ Quantity"), raw_data_curves_trans[, -c(2, 3)])
56   tmp <- data.frame(lapply(curves[[i]][["coefficients"]][1:2], type.convert),
57     stringsAsFactors=FALSE)
58   tmp[, "rsquared"] = signif(summary(curves[[i]])$adj.r.squared, 5)
59   tmp[, "p"] = signif(summary(curves[[i]])$coef[2,4], 5)
60   names(tmp)[1] <- "intercept"
61   names(tmp)[2] <- "slope"
62   curves_param <- rbind(curves_param, cbind(Gene = i, tmp))
63   rm(tmp)
64   print(ggplotRegression(curves[[i]]))
65 }
66 rm(raw_data_curves_trans)
67
68 pcr_amount <- function(vec, a, b) {

```

```

66     res <- 10 ^ ((vec - a)/b)
67     return(res)
68 }
69
70 get_amounts <- function(df, intercept, slope) {
71     amounts <- mapply(function(d, a, b) pcr_amount(d, a, b),
72                        d = df, a = intercept, b = slope)
73     amounts <- as.data.frame(amounts)
74
75     return(amounts)
76 }
77
78
79
80 pcr_normalize <- function(vec, ref, mode) {
81     if (mode == 'subtract') {
82         res <- vec - ref
83     } else if (mode == 'divide') {
84         res <- vec / ref
85     } else {
86         stop("mode can be one of 'subtract' or 'divide'.")
87     }
88     return(res)
89 }
90
91
92 relative_quantities_perSample <- get_amounts(
93     raw_data_sample[3:7],
94     intercept = curves_param$intercept,
95     slope = curves_param$slope
96 )
97
98 normalized_perSample <- pcr_normalize(relative_quantities_perSample,
99     relative_quantities_perSample$Actin, mode = 'divide')
100 normalized_perSample <- cbind(normalized_perSample, raw_data_sample[c("ID", "Group")])
101
102 not_normalized_perSample <- relative_quantities_perSample
103 not_normalized_perSample <- cbind(not_normalized_perSample, raw_data_sample[c("ID",
104 "Group")])
105
106 #Comment next line to apply normalization
107 #normalized_perSample <- not_normalized_perSample
108
109 powerTransform <- function(y, lambda1, lambda2 = NULL, method = "boxcox") {
110     boxcoxTrans <- function(x, lam1, lam2 = NULL) {
111         # if we set lambda2 to zero, it becomes the one parameter transformation
112         lam2 <- ifelse(is.null(lam2), 0, lam2)
113
114         if (lam1 == 0L) {
115             log(y + lam2)
116         } else {
117             (((y + lam2)^lam1) - 1) / lam1
118         }
119     }
120
121     switch(method
122         , boxcox = boxcoxTrans(y, lambda1, lambda2)
123         , tukey = y^lambda1
124     )
125 }
126
127 # run the box-cox transformation to determine optimal lambda
128 bc <- boxcox(normalized_perSample$Portiera ~ normalized_perSample$Group)
129 bc_nn <- boxcox(not_normalized_perSample$Portiera ~ not_normalized_perSample$Group)
130
131 #save and print lambda
132 (lambda <- bc$x[which.max(bc$y)])
133 (lambda_nn <- bc_nn$x[which.max(bc_nn$y)])
134
135 #BoxCox transform the data
136 normalized_perSample <- mutate(normalized_perSample, bcPortiera = powerTransform(

```



```

137 normalized_perSample$Portiera, lambda))
138 not_normalized_perSample <- mutate(not_normalized_perSample, bcPortiera =
139 powerTransform(not_normalized_perSample$Portiera, lambda_nn))
140 #ANOVA model using normalized, transformed data
141 ANOVA_model_norm <- aov(normalized_perSample$bcPortiera~Group, data=
142 normalized_perSample)
143 par(mfrow = c(2, 2))
144 plot(ANOVA_model_norm)
145 summary(ANOVA_model_norm)
146 ANOVA_model_norm
147 #ANOVA model using NOT normalized, transformed data
148 ANOVA_model_nn <- aov(not_normalized_perSample$bcPortiera~Group, data=
149 not_normalized_perSample)
150 par(mfrow = c(2, 2))
151 plot(ANOVA_model_nn)
152 summary(ANOVA_model_nn)
153 ANOVA_model_nn
154 #Post-hoc Tukey test
155 tHSD <- TukeyHSD(ANOVA_model_norm, ordered = FALSE, conf.level = 0.95)
156 tHSD
157 tHSD_nn <- TukeyHSD(ANOVA_model_nn, ordered = FALSE, conf.level = 0.95)
158 tHSD_nn
159 #Function to generate letter groupings for graphing
160 generate_label_df <- function(d, HSD, flev, colNo){
161   # Extract labels and factor levels from Tukey post-hoc
162   Tukey.levels <- HSD[[flev]][,4]
163   Tukey.labels <- multcompLetters(Tukey.levels)['Letters']
164   plot.labels <- names(Tukey.labels[['Letters']])
165
166   # Get highest quantile for Tukey's 5 number summary and add a bit of space to
167   # buffer between
168   # upper quantile and label placement
169   names(d)[colNo] <- "transformed"
170   boxplot.df <- ddply(d, flev, function (x) max(fivenum(x$transformed)) + 0.2)
171
172   # Create a data frame out of the factor levels and Tukey's homogenous group letters
173   plot.levels <- data.frame(plot.labels, labels = Tukey.labels[['Letters']],
174                             stringsAsFactors = FALSE)
175
176   # Merge it with the labels
177   labels.df <- merge(plot.levels, boxplot.df, by.x = 'plot.labels', by.y = flev, sort
178                       = FALSE)
179
180   return(labels.df)
181 }
182
183 normalized_perSample <- mutate(normalized_perSample, OsPortiera = normalized_perSample
184 $Portiera)
185 not_normalized_perSample <- mutate(not_normalized_perSample, OsPortiera =
186 not_normalized_perSample$Portiera)
187
188 library(tidyverse)
189 not_normalized_perSample <- not_normalized_perSample %>%
190   mutate(sex = ifelse(grepl("M$", Group), "Male", "Female"))
191
192 normalized_perSample <- normalized_perSample %>%
193   mutate(sex = ifelse(grepl("M$", Group), "Male", "Female"))
194
195 #Box plot Actin normalized data with Tukey letters, on transformed scale
196 p1<-ggplot(normalized_perSample, aes(x=Group, y=bcPortiera)) +
197   geom_boxplot(show.legend = FALSE, aes(fill=sex)) + scale_fill_manual(values=c(
198   "#bdbdbd", "#636363")) +
199   geom_text(data = generate_label_df(normalized_perSample, tHSD, 'Group', 8), aes(x =
200   plot.labels, y = V1, label = labels)) +
201   ggtitle("Portiera", subtitle = paste("Actin normalized, transformed scale, Box Cox
202   lambda = ", round(lambda, digits=3))) +
203   xlab("Treatments") + ylab("Relative quantity (transformed)") +
204   theme(axis.text = element_text(size = 11)) +
205   theme(axis.title = element_text(size = 12)) +

```



```

199   theme(plot.title = element_text(face="bold")) +
200   scale_x_discrete(labels = str_wrap(c("Antibiotic female", "Antibiotic male",
    "Control female", "Control male", "Colony female", "Colony male"), width = 10))
201 p1
202 #ggsave(filename = "Portiera-AN-Ts.tiff", path="Paper/original scale", width = 600,
height = 700, units="px", device='tiff', dpi=96)
203
204
205 #Box plot Actin normalized data with Tukey letters, on original scale
206 p2<-ggplot(normalized_perSample, aes(x=Group, y=OsPortiera)) +
207   geom_boxplot(show.legend = FALSE , aes(fill=sex)) + scale_fill_manual(values=c(
    "#bdbdbd", "#636363")) +
208   geom_text(data = generate_label_df(normalized_perSample, tHSD, 'Group', 9), aes(x =
    plot.labels, y = V1, label = labels)) +
209   ggtitle("Portiera", subtitle = paste("Actin normalized, original scale, Box Cox
    lambda = ", round(lambda, digits=3))) +
210   xlab("Treatments") + ylab("Relative quantity") +
211   theme(axis.text = element_text(size = 11)) +
212   theme(axis.title = element_text(size = 12)) +
213   theme(plot.title = element_text(face="bold")) +
214   scale_x_discrete(labels = str_wrap(c("Antibiotic female", "Antibiotic male",
    "Control female", "Control male", "Colony female", "Colony male"), width = 10))
215 p2
216 #ggsave(filename = "Portiera-AN-Os.tiff", path="Paper/original scale", width = 600,
height = 700, units="px", device='tiff', dpi=96)
217
218
219 #Box plot NOT normalized data with Tukey letters, on transformed scale
220 p3<-ggplot(not_normalized_perSample, aes(x=Group, y=bcPortiera)) +
221   geom_boxplot(show.legend = FALSE , aes(fill=sex)) + scale_fill_manual(values=c(
    "#bdbdbd", "#636363")) +
222   geom_text(data = generate_label_df(not_normalized_perSample, tHSD_nn, 'Group', 8),
    aes(x = plot.labels, y = V1, label = labels)) +
223   ggtitle("Portiera", subtitle = paste("Not normalized, transformed scale, Box Cox
    lambda = ", round(lambda_nn, digits=3))) +
224   xlab("Treatments") + ylab("Relative quantity (transformed)") +
225   theme(axis.text = element_text(size = 11)) +
226   theme(axis.title = element_text(size = 12)) +
227   theme(plot.title = element_text(face="bold")) +
228   scale_x_discrete(labels = str_wrap(c("Antibiotic female", "Antibiotic male",
    "Control female", "Control male", "Colony female", "Colony male"), width = 10))
229 p3
230 #ggsave(filename = "Portiera-NN-Ts.tiff", path="Paper/original scale", width = 600,
height = 700, units="px", device='tiff', dpi=96)
231
232
233 #Box plot NOT normalized data with Tukey letters, on original scale
234 p4<-ggplot(not_normalized_perSample, aes(x=Group, y=OsPortiera)) +
235   geom_boxplot(show.legend = FALSE , aes(fill=sex)) + scale_fill_manual(values=c(
    "#bdbdbd", "#636363")) +
236   geom_text(data = generate_label_df(not_normalized_perSample, tHSD_nn, 'Group', 9),
    aes(x = plot.labels, y = V1, label = labels)) +
237   ggtitle("Portiera", subtitle = paste("Not normalized, original scale, Box Cox
    lambda = ", round(lambda_nn, digits=3))) +
238   xlab("Treatments") + ylab("Relative quantity") +
239   theme(axis.text = element_text(size = 11)) +
240   theme(axis.title = element_text(size = 12)) +
241   theme(plot.title = element_text(face="bold")) +
242   scale_x_discrete(labels = str_wrap(c("Antibiotic female", "Antibiotic male",
    "Control female", "Control male", "Colony female", "Colony male"), width = 10))
243 p4
244 #ggsave(filename = "Portiera-NN-Os.tiff", path="Paper/original scale", width = 600,
height = 700, units="px", device='tiff', dpi=96)
245
246
247 ggsave(filename = "Portiera-Ts.tiff", path="Paper/Graphs", arrangeGrob(p3, p1,nrow = 1
), width = 322, height = 185, units="mm", device='tiff', dpi=300, compression = "lzw")
248 ggsave(filename = "Portiera-Os.tiff", path="Paper/Graphs", arrangeGrob(p4, p2,nrow = 1
), width = 322, height = 185, units="mm", device='tiff', dpi=300, compression = "lzw")
249
250 ##### End of the R Script for Portiera data analysis #####

```

```

1 ##### Beginning of the R Script for Rickettsia data analysis
2 #####
3 library(ggplot2)
4 library(cowplot)
5 library(tidyr)
6 library(plyr)
7 library(multcompView)
8 library(MASS)
9 library(readr)
10 library(naniar)
11 library(gridExtra)
12
13 ## locate and read data
14 raw_data_sample <- read_csv("Data_samples.csv")
15 #View(raw_data_sample)
16 raw_data_curves <- read_csv("Data_curves.csv")
17 #View(raw_data_curves)
18
19 #Clean up the data
20 raw_data_sample$Group <- gsub('-', '_', raw_data_sample$Group)
21 raw_data_sample <- subset(raw_data_sample, Group != "NC" & Group != "PC_D1" & Group !=
  "PC_D2" & Group != "PC_D3")
22
23 raw_data_sample <- raw_data_sample %>% replace_with_na(replace = list(Rickettsia=
  40.00000))
24
25
26 #Function for plotting standard curves
27 ggplotRegression <- function(fit) {
28
29   require(ggplot2)
30
31   ggplot(fit$model, aes_string(x = names(fit$model)[2], y = names(fit$model)[1])) +
32     geom_point() +
33     stat_smooth(method = "lm", col = "red") +
34     ggtitle(paste("Standard curve for", names(fit$model)[1]),
35             subtitle = substitute(paste("Adj. R"^2, " = ", r,
36                                         ", Intercept = ", i,
37                                         ", Slope = ", s,
38                                         ", p = ", p),
39                                         list(r= signif(summary(fit)$adj.r.squared, 5), i=
40                                         signif(fit$coef[[1]], 5), s=signif(fit$coef[[2]], 5),
41                                         p=signif(summary(fit)$coef[2,4], 5))
42             ) + xlab(paste(names(fit$model)[2], "(log10)")) + ylab(paste(names(fit$
43             model)[1], "(Ct)"))
44   }
45
46 #Make standard curves, save models to "curves" and parameters to "curves_param", and
47 #plot the curves
48 curves_param <- data.frame()
49 curves <- list()
50 raw_data_curves_trans <- raw_data_curves
51 raw_data_curves_trans[, "Quantity"] = log10(raw_data_curves_trans[, "Quantity"])
52 for(i in names(raw_data_curves[, -c(2, 3)])[-1]){
53   curves[[i]] <- lm(paste(i, " ~ Quantity"), raw_data_curves_trans[, -c(2, 3)])
54   tmp <- data.frame(lapply(curves[[i]][["coefficients"]][1:2], type.convert),
55     stringsAsFactors=FALSE)
56   tmp[, "rsquared"] = signif(summary(curves[[i]])$adj.r.squared, 5)
57   tmp[, "p"] = signif(summary(curves[[i]])$coef[2,4], 5)
58   names(tmp)[1] <- "intercept"
59   names(tmp)[2] <- "slope"
60   curves_param <- rbind(curves_param, cbind(Gene = i, tmp))
61   rm(tmp)
62   print(ggplotRegression(curves[[i]]))
63 }
64 rm(raw_data_curves_trans)
65
66 pcr_amount <- function(vec, a, b) {
67   res <- 10 ^ ((vec - a)/b)
68   return(res)
69 }

```

```

66 }
67
68 get_amounts <- function(df, intercept, slope) {
69
70   amounts <- mapply(function(d, a, b) pcr_amount(d, a, b),
71                     d = df, a = intercept, b = slope)
72   amounts <- as.data.frame(amounts)
73
74   return(amounts)
75 }
76
77
78 pcr_normalize <- function(vec, ref, mode) {
79   if (mode == 'subtract') {
80     res <- vec - ref
81   } else if (mode == 'divide') {
82     res <- vec / ref
83   } else {
84     stop("mode can be one of 'subtract' or 'divide'.")
85   }
86   return(res)
87 }
88
89
90 relative_quantities_perSample <- get_amounts(
91   raw_data_sample[3:7],
92   intercept = curves_param$intercept,
93   slope = curves_param$slope
94 )
95
96 normalized_perSample <- pcr_normalize(relative_quantities_perSample,
97   relative_quantities_perSample$Actin, mode = 'divide')
98 normalized_perSample <- cbind(normalized_perSample, raw_data_sample[c("ID", "Group")])
99
100 not_normalized_perSample <- relative_quantities_perSample
101 not_normalized_perSample <- cbind(not_normalized_perSample, raw_data_sample[c("ID",
102   "Group")])
103
104 #Comment next line to apply normalization
105 #normalized_perSample <- not_normalized_perSample
106
107 powerTransform <- function(y, lambda1, lambda2 = NULL, method = "boxcox") {
108
109   boxcoxTrans <- function(x, lam1, lam2 = NULL) {
110
111     # if we set lambda2 to zero, it becomes the one parameter transformation
112     lam2 <- ifelse(is.null(lam2), 0, lam2)
113
114     if (lam1 == 0L) {
115       log(y + lam2)
116     } else {
117       (((y + lam2)^lam1) - 1) / lam1
118     }
119   }
120
121   switch(method
122     , boxcox = boxcoxTrans(y, lambda1, lambda2)
123     , tukey = y^lambda1
124   )
125 }
126
127 # run the box-cox transformation to determine optimal lambda
128 bc <- boxcox(normalized_perSample$Rickettsia ~ normalized_perSample$Group)
129 bc_nn <- boxcox(not_normalized_perSample$Rickettsia ~ not_normalized_perSample$Group)
130
131 #save and print lambda
132 (lambda <- bc$x[which.max(bc$y)])
133 (lambda_nn <- bc_nn$x[which.max(bc_nn$y)])
134
135 #BoxCox transform the data
136 normalized_perSample <- mutate(normalized_perSample, bcRickettsia = powerTransform(
137   normalized_perSample$Rickettsia, lambda))
138 not_normalized_perSample <- mutate(not_normalized_perSample, bcRickettsia =

```

```

136 powerTransform(not_normalized_perSample$Rickettsia, lambda_nn))
137 #ANOVA model using normalized, transformed data
138 ANOVA_model_norm <- aov(normalized_perSample$bcRickettsia~Group, data=
normalized_perSample)
139 par(mfrow = c(2, 2))
140 plot(ANOVA_model_norm)
141 summary(ANOVA_model_norm)
142 ANOVA_model_norm
143
144 #ANOVA model using NOT normalized, transformed data
145 ANOVA_model_nn <- aov(not_normalized_perSample$bcRickettsia~Group, data=
not_normalized_perSample)
146 par(mfrow = c(2, 2))
147 plot(ANOVA_model_nn)
148 summary(ANOVA_model_nn)
149 ANOVA_model_nn
150
151 #Post-hoc Tukey test
152 tHSD <- TukeyHSD(ANOVA_model_norm, ordered = FALSE, conf.level = 0.95)
153 tHSD
154 tHSD_nn <- TukeyHSD(ANOVA_model_nn, ordered = FALSE, conf.level = 0.95)
155 tHSD_nn
156
157 #Function to generate letter groupings for graphing
158 generate_label_df <- function(d, HSD, flev, colNo){
159   # Extract labels and factor levels from Tukey post-hoc
160   Tukey.levels <- HSD[[flev]][,4]
161   Tukey.labels <- multcompLetters(Tukey.levels)['Letters']
162   plot.labels <- names(Tukey.labels[['Letters']])
163
164   # Get highest quantile for Tukey's 5 number summary and add a bit of space to
buffer between
165   # upper quantile and label placement
166   names(d)[colNo] <- "transformed"
167   boxplot.df <- ddply(d, flev, function (x) max(fivenum(x$transformed)) + 0.2)
168
169   # Create a data frame out of the factor levels and Tukey's homogenous group letters
170   plot.levels <- data.frame(plot.labels, labels = Tukey.labels[['Letters']],
stringsAsFactors = FALSE)
171
172   # Merge it with the labels
173   labels.df <- merge(plot.levels, boxplot.df, by.x = 'plot.labels', by.y = flev, sort
= FALSE)
174
175   return(labels.df)
176 }
177
178
179 normalized_perSample <- mutate(normalized_perSample, OsRickettsia =
normalized_perSample$Rickettsia)
180 not_normalized_perSample <- mutate(not_normalized_perSample, OsRickettsia =
not_normalized_perSample$Rickettsia)
181
182 library(tidyverse)
183 not_normalized_perSample <- not_normalized_perSample %>%
184   mutate(sex = ifelse(grepl("M$", Group), "Male", "Female"))
185
186 normalized_perSample <- normalized_perSample %>%
187   mutate(sex = ifelse(grepl("M$", Group), "Male", "Female"))
188
189 #Box plot Actin normalized data with Tukey letters, on transformed scale
190 p1<-ggplot(normalized_perSample, aes(x=Group, y=bcRickettsia)) +
191   geom_boxplot(show.legend = FALSE , aes(fill=sex)) + scale_fill_manual(values=c(
"#bdbdbd", "#636363")) +
192   geom_text(data = generate_label_df(normalized_perSample, tHSD, 'Group', 8), aes(x =
plot.labels, y = V1, label = labels)) +
193   ggtitle("Rickettsia", subtitle = paste("Actin normalized, transformed scale, Box
Cox lambda = ", round(lambda, digits=3))) +
194   xlab("Treatments") + ylab("Relative quantity (transformed)") +
195   theme(axis.text = element_text(size = 11)) +
196   theme(axis.title = element_text(size = 12)) +
197   theme(plot.title = element_text(face="bold")) +
198   scale_x_discrete(labels = str_wrap(c("Antibiotic female", "Antibiotic male",

```

```

199   "Control female", "Control male", "Colony female", "Colony male"), width = 10))
200 p1
201 #ggsave(filename = "Rickettsia-AN-Ts.tiff", path="Paper/original scale", width = 600,
202 height = 700, units="px", device='tiff', dpi=96)
203
204 #Box plot Actin normalized data with Tukey letters, on original scale
205 p2<-ggplot(normalized_perSample, aes(x=Group, y=OsRickettsia)) +
206   geom_boxplot(show.legend = FALSE, aes(fill=sex)) + scale_fill_manual(values=c(
207     "#bdbdbd", "#636363")) +
208   geom_text(data = generate_label_df(normalized_perSample, tHSD, 'Group', 9), aes(x =
209     plot.labels, y = V1, label = labels)) +
210   ggtitle("Rickettsia", subtitle = paste("Actin normalized, original scale, Box Cox
211     lambda = ", round(lambda, digits=3))) +
212   xlab("Treatments") + ylab("Relative quantity") +
213   theme(axis.text = element_text(size = 11)) +
214   theme(axis.title = element_text(size = 12)) +
215   theme(plot.title = element_text(face="bold")) +
216   scale_x_discrete(labels = str_wrap(c("Antibiotic female", "Antibiotic male",
217     "Control female", "Control male", "Colony female", "Colony male"), width = 10))
218 p2
219 #ggsave(filename = "Rickettsia-AN-Os.tiff", path="Paper/original scale", width = 600,
220 height = 700, units="px", device='tiff', dpi=96)
221
222 #Box plot NOT normalized data with Tukey letters, on transformed scale
223 p3<-ggplot(not_normalized_perSample, aes(x=Group, y=bcRickettsia)) +
224   geom_boxplot(show.legend = FALSE, aes(fill=sex)) + scale_fill_manual(values=c(
225     "#bdbdbd", "#636363")) +
226   geom_text(data = generate_label_df(not_normalized_perSample, tHSD_nn, 'Group', 8),
227     aes(x = plot.labels, y = V1, label = labels)) +
228   ggtitle("Rickettsia", subtitle = paste("Not normalized, transformed scale, Box Cox
229     lambda = ", round(lambda_nn, digits=3))) +
230   xlab("Treatments") + ylab("Relative quantity (transformed)") +
231   theme(axis.text = element_text(size = 11)) +
232   theme(axis.title = element_text(size = 12)) +
233   theme(plot.title = element_text(face="bold")) +
234   scale_x_discrete(labels = str_wrap(c("Antibiotic female", "Antibiotic male",
235     "Control female", "Control male", "Colony female", "Colony male"), width = 10))
236 p3
237 #ggsave(filename = "Rickettsia-NN-Ts.tiff", path="Paper/original scale", width = 600,
238 height = 700, units="px", device='tiff', dpi=96)
239
240 #Box plot NOT normalized data with Tukey letters, on original scale
241 p4<-ggplot(not_normalized_perSample, aes(x=Group, y=OsRickettsia)) +
242   geom_boxplot(show.legend = FALSE, aes(fill=sex)) + scale_fill_manual(values=c(
243     "#bdbdbd", "#636363")) +
244   geom_text(data = generate_label_df(not_normalized_perSample, tHSD_nn, 'Group', 9),
245     aes(x = plot.labels, y = V1, label = labels)) +
246   ggtitle("Rickettsia", subtitle = paste("Not normalized, original scale, Box Cox
247     lambda = ", round(lambda_nn, digits=3))) +
248   xlab("Treatments") + ylab("Relative quantity") +
249   theme(axis.text = element_text(size = 11)) +
250   theme(axis.title = element_text(size = 12)) +
251   theme(plot.title = element_text(face="bold")) +
252   scale_x_discrete(labels = str_wrap(c("Antibiotic female", "Antibiotic male",
253     "Control female", "Control male", "Colony female", "Colony male"), width = 10))
254 p4
255 #ggsave(filename = "Rickettsia-NN-Os.tiff", path="Paper/original scale", width = 600,
256 height = 700, units="px", device='tiff', dpi=96)
257
258 ggsave(filename = "Rickettsia-Ts.tiff", path="Paper/Graphs", arrangeGrob(p3, p1, nrow =
259   1), width = 322, height = 185, units="mm", device='tiff', dpi=300, compression =
260   "lzw")
261 ggsave(filename = "Rickettsia-Os.tiff", path="Paper/Graphs", arrangeGrob(p4, p2, nrow =
262   1), width = 322, height = 185, units="mm", device='tiff', dpi=300, compression =
263   "lzw")
264
265 ##### End of the R Script for Rickettsia data analysis #####

```

```

1 ##### Beginning of the R Script for Arsenophonus data analysis
2 #####
3 library(ggplot2)
4 library(cowplot)
5 library(tidyr)
6 library(plyr)
7 library(multcompView)
8 library(MASS)
9 library(readr)
10 library(naniar)
11 library(gridExtra)
12 library(vcd)
13
14 ## locate and read data
15 raw_data_sample <- read_csv("Data_samples.csv")
16 #View(raw_data_sample)
17 raw_data_curves <- read_csv("Data_curves.csv")
18 #View(raw_data_curves)
19
20 #Clean up the data
21 raw_data_sample$Group <- gsub('-', '_', raw_data_sample$Group)
22 raw_data_sample <- subset(raw_data_sample, Group != "NC" & Group != "PC_D1" & Group !=
  "PC_D2" & Group != "PC_D3")
23
24 raw_data_sample <- raw_data_sample %>% replace_with_na(replace = list(Arsenophonus=
  40.00000))
25
26
27 #Function for plotting standard curves
28 ggplotRegression <- function(fit) {
29
30   require(ggplot2)
31
32   ggplot(fit$model, aes_string(x = names(fit$model)[2], y = names(fit$model)[1])) +
33     geom_point() +
34     stat_smooth(method = "lm", col = "red") +
35     ggtitle(paste("Standard curve for", names(fit$model)[1]),
36             subtitle = substitute(paste("Adj. R"2, " = ", r,
37                                           ", Intercept = ", i,
38                                           ", Slope = ", s,
39                                           ", p = ", p),
40                                           list(r= signif(summary(fit)$adj.r.squared, 5), i=
41                                           signif(fit$coef[[1]], 5 ), s=signif(fit$coef[[2]], 5),
42                                           p=signif(summary(fit)$coef[2,4], 5))
43           ) + xlab(paste(names(fit$model)[2], "(log10)")) + ylab(paste(names(fit$
44 model)[1], "(Ct)"))
45
46 }
47
48 #Make standard curves, save models to "curves" and parameters to "curves_param", and
49 plot the curves
50 curves_param <- data.frame()
51 curves <- list()
52 raw_data_curves_trans <- raw_data_curves
53 raw_data_curves_trans[, "Quantity"] = log10(raw_data_curves_trans[, "Quantity"])
54 for(i in names(raw_data_curves[, -c(2, 3)])[-1]){
55   curves[[i]] <- lm(paste(i, " ~ Quantity"), raw_data_curves_trans[, -c(2, 3)])
56   tmp <- data.frame(lapply(curves[[i]][["coefficients"]][1:2], type.convert),
57 stringsAsFactors=FALSE)
58   tmp[, "rsquared"] = signif(summary(curves[[i]])$adj.r.squared, 5)
59   tmp[, "p"] = signif(summary(curves[[i]])$coef[2,4], 5)
60   names(tmp)[1] <- "intercept"
61   names(tmp)[2] <- "slope"
62   curves_param <- rbind(curves_param, cbind(Gene = i, tmp))
63   rm(tmp)
64   print(ggplotRegression(curves[[i]]))
65 }
66 rm(raw_data_curves_trans)
67
68 pcr_amount <- function(vec, a, b) {
69   res <- 10 ^ ((vec - a)/b)

```

```

66     return(res)
67 }
68
69 get_amounts <- function(df, intercept, slope) {
70
71     amounts <- mapply(function(d, a, b) pcr_amount(d, a, b),
72                        d = df, a = intercept, b = slope)
73     amounts <- as.data.frame(amounts)
74
75     return(amounts)
76 }
77
78
79 pcr_normalize <- function(vec, ref, mode) {
80     if (mode == 'subtract') {
81         res <- vec - ref
82     } else if (mode == 'divide') {
83         res <- vec / ref
84     } else {
85         stop("mode can be one of 'subtract' or 'divide'.")
86     }
87     return(res)
88 }
89
90
91 relative_quantities_perSample <- get_amounts(
92     raw_data_sample[3:7],
93     intercept = curves_param$intercept,
94     slope = curves_param$slope
95 )
96
97 normalized_perSample <- pcr_normalize(relative_quantities_perSample,
98     relative_quantities_perSample$Actin, mode = 'divide')
99 normalized_perSample <- cbind(normalized_perSample, raw_data_sample[c("ID", "Group")])
100
101 not_normalized_perSample <- relative_quantities_perSample
102 not_normalized_perSample <- cbind(not_normalized_perSample, raw_data_sample[c("ID",
103     "Group")])
104
105 #Comment next line to apply normalization
106 #normalized_perSample <- not_normalized_perSample
107
108 powerTransform <- function(y, lambda1, lambda2 = NULL, method = "boxcox") {
109
110     boxcoxTrans <- function(x, lam1, lam2 = NULL) {
111
112         # if we set lambda2 to zero, it becomes the one parameter transformation
113         lam2 <- ifelse(is.null(lam2), 0, lam2)
114
115         if (lam1 == 0L) {
116             log(y + lam2)
117         } else {
118             (((y + lam2)^lam1) - 1) / lam1
119         }
120     }
121
122     switch(method
123         , boxcox = boxcoxTrans(y, lambda1, lambda2)
124         , tukey = y^lambda1
125     )
126 }
127
128 # run the box-cox transformation to determine optimal lambda
129 bc <- boxcox(normalized_perSample$Arsenophonus ~ normalized_perSample$Group)
130 bc_nn <- boxcox(not_normalized_perSample$Arsenophonus ~ not_normalized_perSample$Group)
131
132 #save and print lambda
133 (lambda <- bc$x[which.max(bc$y)])
134 (lambda_nn <- bc_nn$x[which.max(bc_nn$y)])
135
136 #BoxCox transform the data
137 normalized_perSample <- mutate(normalized_perSample, bcArsenophonus = powerTransform(

```



```

normalized_perSample$Arsenophonus, lambda))
136 not_normalized_perSample <- mutate(not_normalized_perSample, bcArsenophonus =
powerTransform(not_normalized_perSample$Arsenophonus, lambda_nn))
137
138 #ANOVA model using normalized, transformed data
139 ANOVA_model_norm <- aov(normalized_perSample$bcArsenophonus~Group, data=
normalized_perSample)
140 par(mfrow = c(2, 2))
141 plot(ANOVA_model_norm)
142 summary(ANOVA_model_norm)
143 ANOVA_model_norm
144
145 #ANOVA model using NOT normalized, transformed data
146 ANOVA_model_nn <- aov(not_normalized_perSample$bcArsenophonus~Group, data=
not_normalized_perSample)
147 par(mfrow = c(2, 2))
148 plot(ANOVA_model_nn)
149 summary(ANOVA_model_nn)
150 ANOVA_model_nn
151
152 #Post-hoc Tukey test
153 tHSD <- TukeyHSD(ANOVA_model_norm, ordered = FALSE, conf.level = 0.95)
154 tHSD
155 tHSD_nn <- TukeyHSD(ANOVA_model_nn, ordered = FALSE, conf.level = 0.95)
156 tHSD_nn
157
158 #Function to generate letter groupings for graphing
159 generate_label_df <- function(d, HSD, flev, colNo){
160   # Extract labels and factor levels from Tukey post-hoc
161   Tukey.levels <- HSD[[flev]][,4]
162   Tukey.labels <- multcompLetters(Tukey.levels)['Letters']
163   plot.labels <- names(Tukey.labels[['Letters']])
164
165   # Get highest quantile for Tukey's 5 number summary and add a bit of space to
buffer between
166   # upper quantile and label placement
167   names(d)[colNo] <- "transformed"
168   boxplot.df <- ddply(d, flev, function (x) max(fivenum(x$transformed)) + 0.2)
169
170   # Create a data frame out of the factor levels and Tukey's homogenous group letters
171   plot.levels <- data.frame(plot.labels, labels = Tukey.labels[['Letters']],
stringsAsFactors = FALSE)
172
173   # Merge it with the labels
174   labels.df <- merge(plot.levels, boxplot.df, by.x = 'plot.labels', by.y = flev, sort
= FALSE)
175
176   return(labels.df)
177 }
178
179
180 normalized_perSample <- mutate(normalized_perSample, OsArsenophonus =
normalized_perSample$Arsenophonus)
181 not_normalized_perSample <- mutate(not_normalized_perSample, OsArsenophonus =
not_normalized_perSample$Arsenophonus)
182
183 library(tidyverse)
184 not_normalized_perSample <- not_normalized_perSample %>%
mutate(sex = ifelse(grepl("M$", Group), "Male", "Female"))
185
186
187 normalized_perSample <- normalized_perSample %>%
mutate(sex = ifelse(grepl("M$", Group), "Male", "Female"))
188
189
190 #Box plot Actin normalized data with Tukey letters, on transformed scale
191 p1<-ggplot(normalized_perSample, aes(x=Group, y=bcArsenophonus)) +
192   geom_boxplot(show.legend = FALSE, aes(fill=sex)) + scale_fill_manual(values=c(
"#bdbdbd", "#636363")) +
193   geom_text(data = generate_label_df(normalized_perSample, tHSD, 'Group', 8), aes(x =
plot.labels, y = V1, label = labels)) +
194   ggtitle("Arsenophonus", subtitle = paste("Actin normalized, transformed scale, Box
Cox lambda = ", round(lambda, digits=3))) +
195   xlab("Treatments") + ylab("Relative quantity (transformed)") +
196   theme(axis.text = element_text(size = 11)) +
197   theme(axis.title = element_text(size = 12)) +

```



```

198   theme(plot.title = element_text(face="bold")) +
199   scale_x_discrete(labels = str_wrap(c("Antibiotic female", "Antibiotic male",
    "Control female", "Control male", "Colony female", "Colony male"), width = 10))
200 p1
201 #ggsave(filename = "Arsenophonus-AN-Ts.tiff", path="Paper/original scale", width =
    600, height = 700, units="px", device='tiff', dpi=96)
202
203
204 #Box plot Actin normalized data with Tukey letters, on original scale
205 p2<-ggplot(normalized_perSample, aes(x=Group, y=OsArsenophonus)) +
206   geom_boxplot(show.legend = FALSE , aes(fill=sex)) + scale_fill_manual(values=c(
    "#bdbdbd", "#636363")) +
207   geom_text(data = generate_label_df(normalized_perSample, tHSD, 'Group', 9), aes(x =
    plot.labels, y = V1, label = labels)) +
208   ggtitle("Arsenophonus", subtitle = paste("Actin normalized, original scale, Box Cox
    lambda = ", round(lambda, digits=3))) +
209   xlab("Treatments") + ylab("Relative quantity") +
210   theme(axis.text = element_text(size = 11)) +
211   theme(axis.title = element_text(size = 12)) +
212   theme(plot.title = element_text(face="bold")) +
213   scale_x_discrete(labels = str_wrap(c("Antibiotic female", "Antibiotic male",
    "Control female", "Control male", "Colony female", "Colony male"), width = 10))
214 p2
215 #ggsave(filename = "Arsenophonus-AN-Os.tiff", path="Paper/original scale", width =
    600, height = 700, units="px", device='tiff', dpi=96)
216
217
218 #Box plot NOT normalized data with Tukey letters, on transformed scale
219 p3<-ggplot(not_normalized_perSample, aes(x=Group, y=bcArsenophonus)) +
220   geom_boxplot(show.legend = FALSE , aes(fill=sex)) + scale_fill_manual(values=c(
    "#bdbdbd", "#636363")) +
221   geom_text(data = generate_label_df(not_normalized_perSample, tHSD_nn, 'Group', 8),
    aes(x = plot.labels, y = V1, label = labels)) +
222   ggtitle("Arsenophonus", subtitle = paste("Not normalized, transformed scale, Box
    Cox lambda = ", round(lambda_nn, digits=3))) +
223   xlab("Treatments") + ylab("Relative quantity (transformed)") +
224   theme(axis.text = element_text(size = 11)) +
225   theme(axis.title = element_text(size = 12)) +
226   theme(plot.title = element_text(face="bold")) +
227   scale_x_discrete(labels = str_wrap(c("Antibiotic female", "Antibiotic male",
    "Control female", "Control male", "Colony female", "Colony male"), width = 10))
228 p3
229 #ggsave(filename = "Arsenophonus-NN-Ts.tiff", path="Paper/original scale", width =
    600, height = 700, units="px", device='tiff', dpi=96)
230
231
232 #Box plot NOT normalized data with Tukey letters, on original scale
233 p4<-ggplot(not_normalized_perSample, aes(x=Group, y=OsArsenophonus)) +
234   geom_boxplot(show.legend = FALSE , aes(fill=sex)) + scale_fill_manual(values=c(
    "#bdbdbd", "#636363")) +
235   geom_text(data = generate_label_df(not_normalized_perSample, tHSD_nn, 'Group', 9),
    aes(x = plot.labels, y = V1, label = labels)) +
236   ggtitle("Arsenophonus", subtitle = paste("Not normalized, original scale, Box Cox
    lambda = ", round(lambda_nn, digits=3))) +
237   xlab("Treatments") + ylab("Relative quantity") +
238   theme(axis.text = element_text(size = 11)) +
239   theme(axis.title = element_text(size = 12)) +
240   theme(plot.title = element_text(face="bold")) +
241   scale_x_discrete(labels = str_wrap(c("Antibiotic female", "Antibiotic male",
    "Control female", "Control male", "Colony female", "Colony male"), width = 10))
242
243 p4
244 #ggsave(filename = "Arsenophonus-NN-Os.tiff", path="Paper/original scale", width =
    600, height = 700, units="px", device='tiff', dpi=96)
245
246
247 ggsave(filename = "Arsenophonus-Ts.tiff", path="Paper/Graphs", arrangeGrob(p3, p1, nrow
    = 1), width = 322, height = 185, units="mm", device='tiff', dpi=300, compression =
    "lzw")
248 ggsave(filename = "Arsenophonus-Os.tiff", path="Paper/Graphs", arrangeGrob(p4, p2, nrow
    = 1), width = 322, height = 185, units="mm", device='tiff', dpi=300, compression =
    "lzw")
249

```

```

250 write.csv2(not_normalized_perSample, 'Not normalized per sampleA.csv')
251 write.csv2(normalized_perSample, 'Normalized per sampleA.csv')
252
253
254 no<-aggregate(Arsenophonus ~ Group, data=normalized_perSample, function(x) {sum(is.na(
x))}), na.action = NULL)
255 yes<-aggregate(cbind(count = Arsenophonus) ~ Group, data = normalized_perSample, FUN =
function(x) {NROW(x)})
256 Ars_counts <- cbind(yes, no[c("Arsenophonus")])
257 Ars_counts
258 names(Ars_counts)[1] <- "Treatment"
259 names(Ars_counts)[2] <- "Positive"
260 names(Ars_counts)[3] <- "Negative"
261 Ars_counts
262
263 #Sum up the totals, calculate percentage for easy interpretation and export
264 Ars_counts_export <- mutate(Ars_counts, Total = Ars_counts$Positive + Ars_counts$
Negative)
265 Ars_counts_export <- mutate(Ars_counts_export, PercentNegative = Ars_counts_export$
Negative / Ars_counts_export$Total)
266 write.csv2(Ars_counts_export, 'Arsenophonus-presence.csv')
267
268 matrix.please<-function(x) {
269   m<-as.matrix(x[, -1])
270   rownames(m) <- x[, 1]
271   m
272 }
273
274 Ars_counts <- matrix.please(Ars_counts)
275 mosaic(Ars_counts, shade=TRUE, legend=TRUE, labeling= labeling_border(rot_labels = c(0
, 0, 0, 0),
276
just_labels = c(
"center",
277
"center",
278
"center",
279
"right"), set_varnames = list(A = "", B = "qPCR result")))
280 library(rcompanion)
281 PT = pairwiseNominalIndependence(Ars_counts,
282   fisher = TRUE,
283   gtest = FALSE,
284   chisq = FALSE,
285   digits = 3,
286   method='holm',
287   cramer = 'TRUE',
288   workspace=2e8)
289
290 PT
291 #write.csv2(PT, 'Arsenophonus-presence-Fisher.csv')
292
293 cldList(comparison = PT$Comparison,
294   p.value = PT$p.adj.Fisher,
295   threshold = 0.05)
296
297 ##### End of the R Script for Arsenophonus data analysis #####
298

```