Fine-scale differentiation between B. anthracis and B. cereus group signatures in metagenome shotgun data

Packages and Functions

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
options(scipen=999)
library(grid)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
PATH <- '/home/rpetit/ba-paper'
FIGURE <- '/home/rpetit/ba-paper/results/figures'

read_samples <- function(gzip_file, distance) {
    s = merge(
        read.table(gzfile(gzip_file), header=TRUE),
        data.frame(sample=distance$accession, distance=distance$distance),
        by=c('sample')
    )

    s$bacillus = ifelse(
        s$is_bcg == 'True', ifelse(s$is_ba=='True', 'ba', 'bcg'), 'nonbcg'
    )

    return(s)
}

format_plot <- function(ggplot_object, xlab="Mean k-mer Coverage", ylab="Count", title=FALSE) {
    plot_title = theme(legend.title = element_blank())
    if (title != FALSE) {
        plot_title <- labs(color=title)
    }
    p <- ggplot_object +
```

```
        xlab(xlab) +
        ylab(ylab) +
        theme_bw() +
        theme(axis.text=element_text(size=12),
              axis.title=element_text(size=14,face="bold")) +
        plot_title
    return(p)
}

plot_hamming <- function(df) {
    p <- ggplot(df, aes(hamming_distance, hit)) +
        xlab("Hamming Distance") +
        ylab("Count") +
        geom_bar(stat='identity') +
        scale_x_continuous(breaks = seq(
            min(df$hamming_distance), max(df$hamming_distance), by = 1
        )) +
        geom_text(aes(label=hit), vjust = -0.5) +
        theme_bw() +
        theme(text=element_text(size=20),
              title=element_text(size=14, face="bold"),
              legend.position="none")
    return (p)
}


lm_eqn <- function(m, zero_intercept=TRUE){
    # GET EQUATION AND R-SQUARED AS STRING
    # SOURCE: http://goo.gl/K4yh
    a = ifelse(zero_intercept, 0, coef(m)[1])
    b = ifelse(zero_intercept, coef(m)[1], coef(m)[2])
    eq <- substitute(italic(y) == a + b %.% italic(x)*","~~italic(r)^2~"="~r2,
         list(a = format(a, digits = 2), b = format(b, digits = 2),
              r2 = format(summary(m)$r.squared, digits = 3)))
    as.character(as.expression(eq));
}

#' write_plot
#'
#' A wrapper for to validate given vector is multiple ids and proper type. This
#' function should not be directly used by the user.
#'
#' @param plot_object A ggplot object
#' @param name Basename for the output PDF and PNG files
#' @param height The PDF height of the output (Default: 5)
#' @param width The PDF width of the object (Default: 12)
#'
#' @export
#' @return bool TRUE is multiple ids else FALSE.
write_plot <- function(plot_object, name, height = 5, width = 12) {
    pdf(paste0(FIGURE, '/', name, ".pdf"), width=width, height=height)
    print(plot_object)
    dev_null <- dev.off()
```

```
    png(paste0(FIGURE, '/', name, ".png"), width=width*100, height=height*100)
    print(plot_object)
    dev_null <- dev.off()
}
```

Lethal Factor Gray Zone

Inputs

```
gray_ba_samples = read.table(
    gzfile(paste0(PATH, "/results/limit-of-detection-lef31/lef-ba-summary.txt.gz")),
    header=TRUE, sep="\t"
)

gray_bcg_samples = read.table(
    gzfile(paste0(PATH, "/results/limit-of-detection-lef31/lef-bcg-summary.txt.gz")),
    header=TRUE, sep="\t"
)

gray_lef_samples = read.table(
    gzfile(paste0(PATH, "/results/limit-of-detection-lef31/lef-lef-summary.txt.gz")),
    header=TRUE, sep="\t"
)

gray_summary = read.table(
    paste0(PATH, "/results/limit-of-detection-lef31/lef-subsample.txt"),
    header=TRUE, sep="\t"
)

gray_summary <- merge(
    gray_summary,
    gray_ba_samples %>%
        group_by(sample, coverage) %>%
        summarise(ba_tp=mean(tp), ba_coverage=mean(kmer_cov_mean),
                  ba_coverage_nonzero=mean(non_zero_kmer_cov_mean)),
    by=c('sample', 'coverage')
)

gray_summary <- merge(
    gray_summary,
    gray_bcg_samples %>%
        group_by(sample, coverage) %>%
        summarise(bcg_tp=mean(tp), bcg_coverage=mean(kmer_cov_mean),
                  bcg_coverage_nonzero=mean(non_zero_kmer_cov_mean)),
    by=c('sample', 'coverage')
)

gray_summary <- merge(
    gray_summary,
    gray_lef_samples %>%
```

```
        group_by(sample, coverage) %>%
        summarise(lef_tp=mean(tp), lef_coverage=mean(kmer_cov_mean),
                  lef_coverage_nonzero=mean(non_zero_kmer_cov_mean)),
    by=c('sample', 'coverage')
)

gray_area_summary <- function(df, group, status) {
    temp_df <- data.frame(table(round(df, digits = 5)))
    colnames(temp_df) <- c('coverage', 'count')
    temp_df$group <- group
    temp_df$status <- status
    temp_df$coverage <- as.numeric(as.character(temp_df$coverage))
    temp_df$cumsum <- cumsum(temp_df$count)
    temp_df$percent <- temp_df$cumsum / max(temp_df$cumsum)
    return(temp_df)
}

grid_arrange_shared_lef31 <- function(..., ncol = length(list(...)), nrow = 1, position = c("bottom", "
  plots <- list(...)
  position <- match.arg(position)
  g <- ggplotGrob(plots[[1]] + theme(legend.position = position))$grobs
  legend <- g[[which(sapply(g, function(x) x$name) == "guide-box")]]
  lheight <- sum(legend$height)
  lwidth <- sum(legend$width)
  gl <- lapply(plots, function(x) x + theme(legend.position="none"))
  gl[[1]] <-arrangeGrob(gl[[1]], top = textGrob(
        "A", x = unit(0, "npc"), y = unit(1, "npc"), just=c("left","top"),
         gp=gpar(col="black", fontsize=18, fontface="bold"))
    )
  gl[[2]] <-arrangeGrob(gl[[2]], top = textGrob(
        "B", x = unit(0, "npc"), y = unit(1, "npc"), just=c("left","top"),
         gp=gpar(col="black", fontsize=18, fontface="bold"))
    )

  gl <- c(gl, ncol = ncol, nrow = nrow)

  combined <- switch(position,
                     "bottom" = arrangeGrob(do.call(arrangeGrob, gl),
                                            legend,
                                            ncol = 1,
                                            heights = unit.c(unit(1, "npc") - lheight, lheight)),
                     "right" = arrangeGrob(do.call(arrangeGrob, gl),
                                           legend,
                                           ncol = 2,
                                           widths = unit.c(unit(1, "npc") - lwidth, lwidth)))

  grid.newpage()
  grid.draw(combined)

  # return gtable invisibly
  invisible(combined)
}
```

```
head(gray_summary)
```

```
##      sample coverage tests successes  status     ba_tp ba_coverage
## 1 ERR930296   0.0400   100        95 relaxed  5562.642  0.02347827
## 2 ERR930296   0.0700   100       100  strict  9555.610  0.04074087
## 3 ERR930297   0.0589   100        99 relaxed  8441.505  0.03589068
## 4 ERR930297   0.0900   100       100  strict 12692.440  0.05445297
## 5 ERR930298   0.0590   100        97 relaxed  8437.495  0.03590081
## 6 ERR930298   0.1000   100       100  strict 14226.180  0.06123230
##   ba_coverage_nonzero   bcg_tp bcg_coverage bcg_coverage_nonzero   lef_tp
## 1            1.010975 258.5684   0.02576120             1.013722 189.2316
## 2            1.021273 436.5200   0.04388294             1.023961 301.1000
## 3            1.018254 368.9789   0.03705247             1.023516 208.8737
## 4            1.027492 553.4500   0.05609447             1.031130 340.3700
## 5            1.019040 371.2577   0.03710551             1.017710 204.9175
## 6            1.030878 609.7900   0.06219680             1.038531 304.3400
##   lef_coverage lef_coverage_nonzero
## 1   0.07564708             1.048649
## 2   0.12483760             1.081128
## 3   0.08272228             1.030974
## 4   0.14339702             1.092726
## 5   0.08183211             1.032480
## 6   0.12476118             1.065953
```

Gray Area Group Sumamry

```
gray_area_group_summary <- rbind(
    gray_area_summary(gray_summary[gray_summary$status == 'strict',]$ba_coverage, 'ba', '100%'),
    gray_area_summary(gray_summary[gray_summary$status == 'relaxed',]$ba_coverage, 'ba', '95%'),
    gray_area_summary(gray_summary[gray_summary$status == 'strict',]$bcg_coverage, 'bcg', '100%'),
    gray_area_summary(gray_summary[gray_summary$status == 'relaxed',]$bcg_coverage, 'bcg', '95%'),
    gray_area_summary(gray_summary[gray_summary$status == 'strict',]$lef_coverage, 'lef', '100%'),
    gray_area_summary(gray_summary[gray_summary$status == 'relaxed',]$lef_coverage, 'lef', '95%')
)
```

group column - ba: Ba31 kmer coverage - bcg: BCerG kmer coverage - lef: lef31 kmer coverage

status column - **strict** lef kmer found in 100% of subsamples - **relaxed** lef kmer found in $> 95\%$ of subsamples

Ba31 k-mer Coverage & BCerG31 k-mer Coverage

```
temp_df <- gray_area_group_summary[gray_area_group_summary$group == 'ba',]
print(temp_df[temp_df$percent >= 0.95,])
```

```
##     coverage count group status cumsum   percent
## 151  0.11361     1    ba   100%    156 0.9512195
## 152  0.11532     1    ba   100%    157 0.9573171
## 153  0.11549     1    ba   100%    158 0.9634146
## 154  0.11791     1    ba   100%    159 0.9695122
## 155  0.12036     1    ba   100%    160 0.9756098
## 156  0.12039     1    ba   100%    161 0.9817073
## 157  0.12113     1    ba   100%    162 0.9878049
```

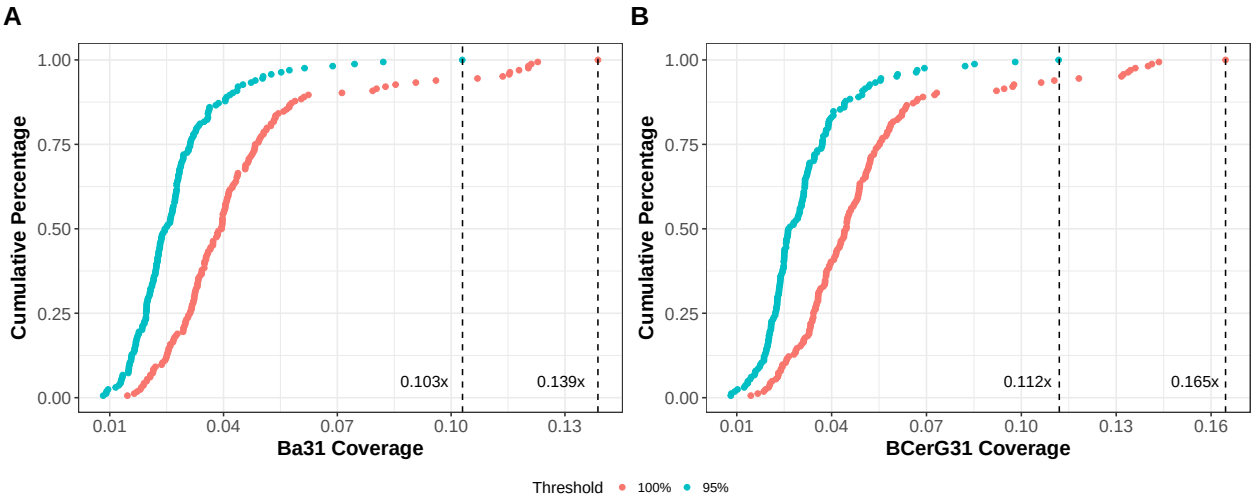
```
## 158  0.12284     1    ba   100%    163 0.9939024
## 159  0.13870     1    ba   100%    164 1.0000000
## 310  0.05044     1    ba    95%    156 0.9512195
## 311  0.05258     1    ba    95%    157 0.9573171
## 312  0.05520     1    ba    95%    158 0.9634146
## 313  0.05736     1    ba    95%    159 0.9695122
## 314  0.06137     1    ba    95%    160 0.9756098
## 315  0.06878     1    ba    95%    161 0.9817073
## 316  0.07455     1    ba    95%    162 0.9878049
## 317  0.08211     1    ba    95%    163 0.9939024
## 318  0.10294     1    ba    95%    164 1.0000000
```

```
max_95 = round(max(temp_df[temp_df$status == "95%",]$coverage), digits=3)
max_100 = round(max(temp_df[temp_df$status == "100%",]$coverage), digits=3)
g <- ggplot(temp_df, aes(coverage, percent, color=status)) + geom_point() +
    scale_x_continuous(breaks = seq(0.01, max(temp_df$coverage)+0.01, by = 0.03)) +
    geom_vline(xintercept = max_95, linetype="dashed") +
    annotate("text", x = max_95 - 0.01, y = 0.05,
             label = paste0(as.character(max_95), "x"), size=4) +
    geom_vline(xintercept = max(temp_df[temp_df$status == '100%',]$coverage), linetype="dashed") +
    annotate("text", x = max_100 - 0.01, y = 0.05,
         label = paste0(as.character(max_100), "x"), size=4)


p1 <- format_plot(g, xlab="Ba31 Coverage", ylab="Cumulative Percentage", title="Threshold")
remove(temp_df)

temp_df <- gray_area_group_summary[gray_area_group_summary$group == 'bcg',]
print(temp_df[temp_df$percent >= 0.95,])
```

```
##     coverage count group status cumsum   percent
## 469  0.13167     1   bcg   100%    156 0.9512195
## 470  0.13231     1   bcg   100%    157 0.9573171
## 471  0.13375     1   bcg   100%    158 0.9634146
## 472  0.13451     1   bcg   100%    159 0.9695122
## 473  0.13605     1   bcg   100%    160 0.9756098
## 474  0.14008     1   bcg   100%    161 0.9817073
## 475  0.14128     1   bcg   100%    162 0.9878049
## 476  0.14351     1   bcg   100%    163 0.9939024
## 477  0.16458     1   bcg   100%    164 1.0000000
## 627  0.06060     1   bcg    95%    156 0.9512195
## 628  0.06089     1   bcg    95%    157 0.9573171
## 629  0.06674     1   bcg    95%    158 0.9634146
## 630  0.06706     1   bcg    95%    159 0.9695122
## 631  0.06936     1   bcg    95%    160 0.9756098
## 632  0.08222     1   bcg    95%    161 0.9817073
## 633  0.08519     1   bcg    95%    162 0.9878049
## 634  0.09806     1   bcg    95%    163 0.9939024
## 635  0.11180     1   bcg    95%    164 1.0000000
```

```
max_95 = round(max(temp_df[temp_df$status == "95%",]$coverage), digits=3)
max_100 = round(max(temp_df[temp_df$status == "100%",]$coverage), digits=3)
g <- ggplot(temp_df, aes(coverage, percent, color=status)) + geom_point() +
    scale_x_continuous(breaks = seq(0.01, max(temp_df$coverage)+0.01, by = 0.03)) +
    geom_vline(xintercept = max_95, linetype="dashed") +
```

```
    annotate("text", x = max_95 - 0.01, y = 0.05,
             label = paste0(as.character(max_95), "x"), size=4) +
    geom_vline(xintercept = max(temp_df[temp_df$status == '100%',]$coverage), linetype="dashed") +
    annotate("text", x = max_100 - 0.01, y = 0.05,
         label = paste0(as.character(max_100), "x"), size=4)

p2 <- format_plot(g, xlab="BCerG31 Coverage", ylab="Cumulative Percentage", title="Threshold")
remove(temp_df)


pdf(paste0(FIGURE, '/figure-01-lef31-lod.pdf'), height=6, width=12, onefile=FALSE)
grid_arrange_shared_lef31(p1, p2)
dev_null <- dev.off()

png(paste0(FIGURE, '/figure-01-lef31-lod.png'), height=600, width=1200)
grid_arrange_shared_lef31(p1, p2)
dev_null <- dev.off()

grid_arrange_shared_lef31(p1, p2)
```



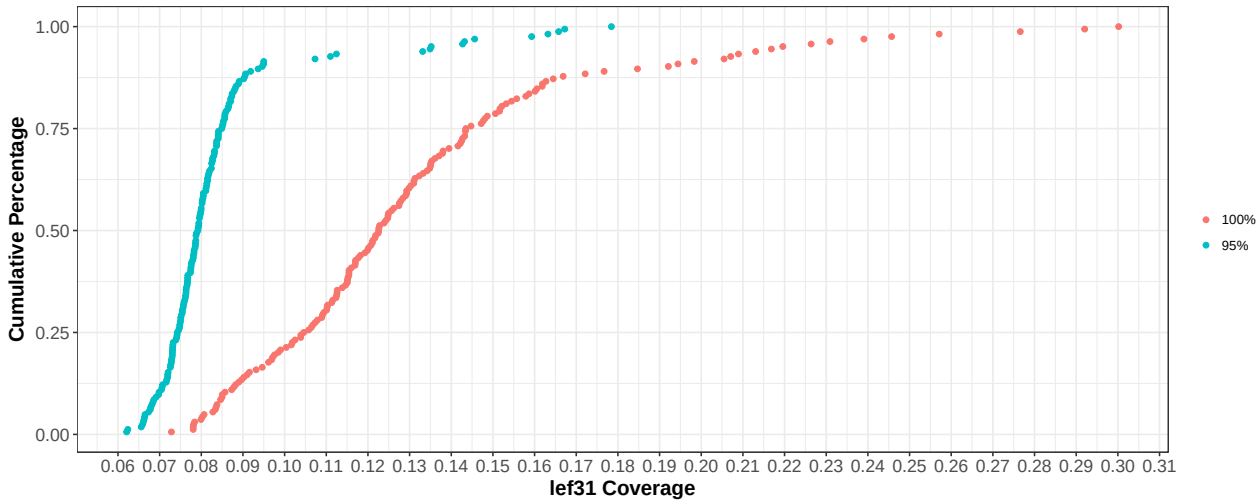
lef31 k-mer Coverage

```
temp_df <- gray_area_group_summary[gray_area_group_summary$group == 'lef',]
print(temp_df[temp_df$percent >= 0.95,])
```

```
##     coverage count group status cumsum   percent
## 788  0.21959     1   lef   100%    156 0.9512195
## 789  0.22638     1   lef   100%    157 0.9573171
## 790  0.23091     1   lef   100%    158 0.9634146
## 791  0.23906     1   lef   100%    159 0.9695122
## 792  0.24570     1   lef   100%    160 0.9756098
## 793  0.25712     1   lef   100%    161 0.9817073
## 794  0.27658     1   lef   100%    162 0.9878049
## 795  0.29204     1   lef   100%    163 0.9939024
## 796  0.30020     1   lef   100%    164 1.0000000
## 946  0.13519     1   lef    95%    156 0.9512195
```

```
## 947  0.14273     1   lef    95%    157 0.9573171
## 948  0.14318     1   lef    95%    158 0.9634146
## 949  0.14562     1   lef    95%    159 0.9695122
## 950  0.15930     1   lef    95%    160 0.9756098
## 951  0.16321     1   lef    95%    161 0.9817073
## 952  0.16578     1   lef    95%    162 0.9878049
## 953  0.16725     1   lef    95%    163 0.9939024
## 954  0.17844     1   lef    95%    164 1.0000000
```

```
g <- ggplot(temp_df, aes(coverage, percent, color=status)) + geom_point() +
    scale_x_continuous(breaks = seq(0.01, max(temp_df$coverage)+0.01, by = 0.01))

p <- format_plot(g, xlab="lef31 Coverage", ylab="Cumulative Percentage")
print(p)
```




```
remove(temp_df)
```

Simulations

Inputs

```
ba_kmer_info = read.table(paste0(PATH, "/results/ba-kmer-info.txt"),
                          header=TRUE, sep="\t")

bcg_kmer_info = read.table(paste0(PATH, "/results/bcg-kmer-info.txt"),
                           header=TRUE, sep="\t")

simulation_ba = read_samples(
    paste0(PATH, "/results/simulations/simulation-ba-summary.txt.gz"),
    ba_kmer_info
)

simulation_bcg = read_samples(
    paste0(PATH, "/results/simulations/simulation-bcg-summary.txt.gz"),
    ba_kmer_info
)
```

```
simulation_lef = read_samples(
    paste0(PATH, "/results/simulations/simulation-lef-summary.txt.gz"),
    ba_kmer_info
)

simulation_ba = merge(
    simulation_ba,
    data.frame(
        sample=simulation_bcg$sample,
        simulated_coverage=simulation_bcg$simulated_coverage,
        bcg_cov_mean=simulation_bcg$kmer_cov_mean
    ),
    by=c('sample', 'simulated_coverage')
)
```

Coverages Simulated

A total of **341** coverages were simulated.

```
print(sort(unique(simulation_ba$simulated_coverage)))
```

```
##   [1]  0.01  0.03  0.05  0.07  0.09  0.11  0.13  0.15  0.17  0.19  0.21
##  [12]  0.23  0.25  0.27  0.29  0.31  0.33  0.35  0.37  0.39  0.41  0.43
##  [23]  0.45  0.47  0.49  0.51  0.53  0.55  0.57  0.59  0.61  0.63  0.65
##  [34]  0.67  0.69  0.71  0.73  0.75  0.77  0.79  0.81  0.83  0.85  0.87
##  [45]  0.89  0.91  0.93  0.95  0.97  0.99  1.01  1.03  1.05  1.07  1.09
##  [56]  1.11  1.13  1.15  1.17  1.19  1.21  1.23  1.25  1.27  1.29  1.31
##  [67]  1.33  1.35  1.37  1.39  1.41  1.43  1.45  1.47  1.49  1.51  1.53
##  [78]  1.55  1.57  1.59  1.61  1.63  1.65  1.67  1.69  1.71  1.73  1.75
##  [89]  1.77  1.79  1.81  1.83  1.85  1.87  1.89  1.91  1.93  1.95  1.97
## [100]  1.99  2.01  2.03  2.05  2.07  2.09  2.11  2.13  2.15  2.17  2.19
## [111]  2.21  2.23  2.25  2.27  2.29  2.31  2.33  2.35  2.37  2.39  2.41
## [122]  2.43  2.45  2.47  2.49  2.51  2.53  2.55  2.57  2.59  2.61  2.63
## [133]  2.65  2.67  2.69  2.71  2.73  2.75  2.77  2.79  2.81  2.83  2.85
## [144]  2.87  2.89  2.91  2.93  2.95  2.97  2.99  3.00  3.05  3.10  3.15
## [155]  3.20  3.25  3.30  3.35  3.40  3.45  3.50  3.55  3.60  3.65  3.70
## [166]  3.75  3.80  3.85  3.90  3.95  4.00  4.05  4.10  4.15  4.20  4.25
## [177]  4.30  4.35  4.40  4.45  4.50  4.55  4.60  4.65  4.70  4.75  4.80
## [188]  4.85  4.90  4.95  5.00  5.05  5.10  5.15  5.20  5.25  5.30  5.35
## [199]  5.40  5.45  5.50  5.55  5.60  5.65  5.70  5.75  5.80  5.85  5.90
## [210]  5.95  6.00  6.05  6.10  6.15  6.20  6.25  6.30  6.35  6.40  6.45
## [221]  6.50  6.55  6.60  6.65  6.70  6.75  6.80  6.85  6.90  6.95  7.00
## [232]  7.05  7.10  7.15  7.20  7.25  7.30  7.35  7.40  7.45  7.50  7.55
## [243]  7.60  7.65  7.70  7.75  7.80  7.85  7.90  7.95  8.00  8.05  8.10
## [254]  8.15  8.20  8.25  8.30  8.35  8.40  8.45  8.50  8.55  8.60  8.65
## [265]  8.70  8.75  8.80  8.85  8.90  8.95  9.00  9.05  9.10  9.15  9.20
## [276]  9.25  9.30  9.35  9.40  9.45  9.50  9.55  9.60  9.65  9.70  9.75
## [287]  9.80  9.85  9.90  9.95 10.00 10.10 10.20 10.30 10.40 10.50 10.60
## [298] 10.70 10.80 10.90 11.00 11.10 11.20 11.30 11.40 11.50 11.60 11.70
## [309] 11.80 11.90 12.00 12.10 12.20 12.30 12.40 12.50 12.60 12.70 12.80
## [320] 12.90 13.00 13.10 13.20 13.30 13.40 13.50 13.60 13.70 13.80 13.90
## [331] 14.00 14.10 14.20 14.30 14.40 14.50 14.60 14.70 14.80 14.90 15.00
```

Samples

A total of **379** reference *Bacillus* genus genomes had reads simulated for each of the above coverages.

```
table(simulation_ba[simulation_ba$simulated_coverage == 0.01,]$bacillus)
```

```
##
##     ba    bcg nonbcg
##     48     95    236
```

Results

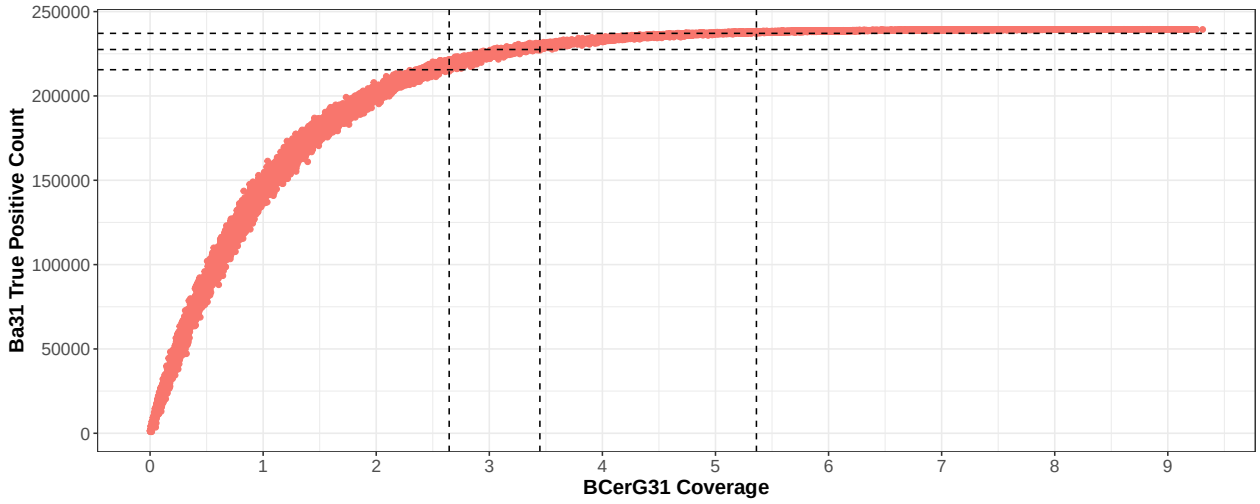
Lethal Factor

There were **0** simualtions with hits to the lethal factor kmers.

Plots

Ba31 Sensitivity

```
temp_df <- simulation_ba[simulation_ba$is_ba == 'True',]
temp_total <- max(temp_df$total_kmers)
temp_90 <- floor(temp_total*0.90)
temp_95 <- floor(temp_total*0.95)
temp_99 <- floor(temp_total*0.99)
temp_999 <- floor(temp_total*0.999)
g <- ggplot(temp_df, aes(bcg_cov_mean, tp)) +
    geom_point(aes(color=is_ba)) +
    geom_hline(yintercept=temp_90, linetype="dashed") +
    geom_hline(yintercept=temp_95, linetype="dashed") +
    geom_hline(yintercept=temp_99, linetype="dashed") +
    geom_vline(xintercept=max(temp_df[temp_df$tp <= temp_90,]$bcg_cov_mean), linetype="dashed") +
    geom_vline(xintercept=max(temp_df[temp_df$tp <= temp_95,]$bcg_cov_mean), linetype="dashed") +
    geom_vline(xintercept=max(temp_df[temp_df$tp <= temp_99,]$bcg_cov_mean), linetype="dashed") +
    scale_x_continuous(breaks = seq(0, ceiling(max(temp_df$bcg_cov_mean)), by = 1))

p <- format_plot(g, xlab="BCerG31 Coverage", ylab="Ba31 True Positive Count") +
    theme(legend.position="none")
print(p)
```




```
print(
    data.frame(
        percent=c("90%", "95%", "99%", "99.9%"),
        total=c(temp_90, temp_95, temp_99, temp_999),
        min=c(
            min(temp_df[temp_df$tp >= temp_90,]$bcg_cov_mean),
            min(temp_df[temp_df$tp >= temp_95,]$bcg_cov_mean),
            min(temp_df[temp_df$tp >= temp_99,]$bcg_cov_mean),
            min(temp_df[temp_df$tp >= temp_999,]$bcg_cov_mean)
        ),
        max=c(
            max(temp_df[temp_df$tp <= temp_90,]$bcg_cov_mean),
            max(temp_df[temp_df$tp <= temp_95,]$bcg_cov_mean),
            max(temp_df[temp_df$tp <= temp_99,]$bcg_cov_mean),
            max(temp_df[temp_df$tp <= temp_999,]$bcg_cov_mean)
        )
    )
)
```

```
##   percent  total    min    max
## 1     90% 215552 2.3684 2.6451
## 2     95% 227527 3.0363 3.4473
## 3     99% 237107 4.6638 5.3620
## 4   99.9% 239263 6.8874 8.0657
```

Ba31 Coverage and BCerG Coverage

```
ba_bcg_coverages_lm = lm(kmer_cov_mean ~ 0 + bcg_cov_mean, temp_df)
summary(ba_bcg_coverages_lm)
```

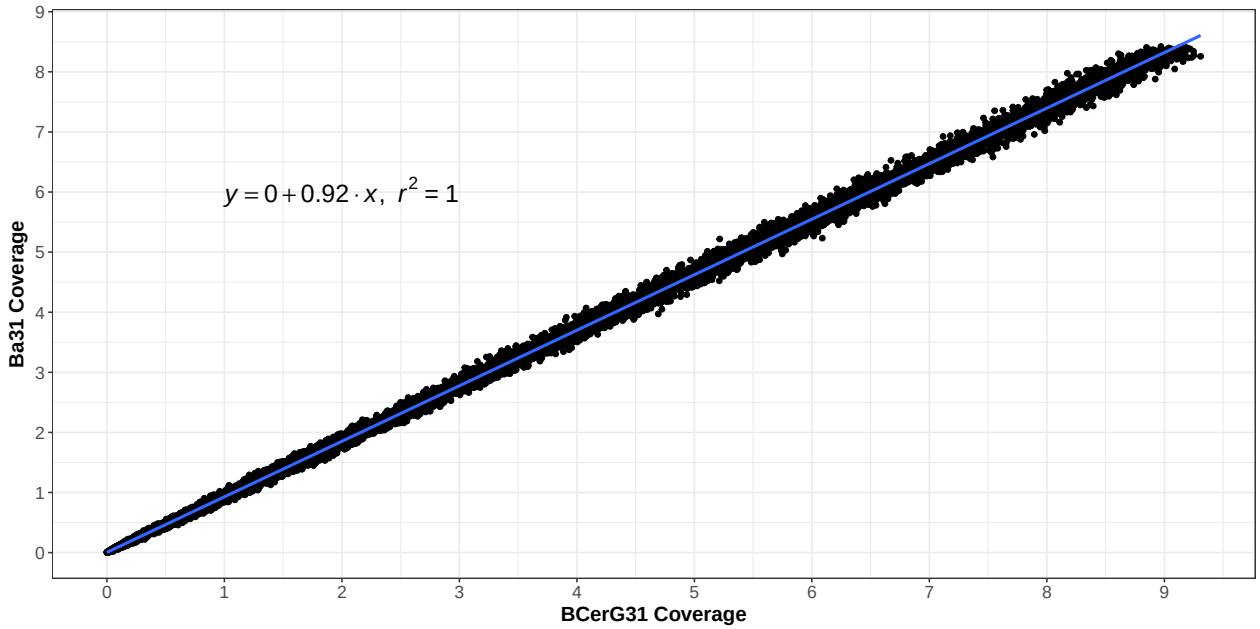
```
##
## Call:
## lm(formula = kmer_cov_mean ~ 0 + bcg_cov_mean, data = temp_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.39812 -0.03832  0.00094  0.04225  0.41704
```

```
##
## Coefficients:
##                Estimate Std. Error t value            Pr(>|t|)
## bcg_cov_mean 0.9247825  0.0001525    6065 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07775 on 16367 degrees of freedom
## Multiple R-squared:  0.9996, Adjusted R-squared:  0.9996
## F-statistic: 3.679e+07 on 1 and 16367 DF,  p-value: < 0.00000000000000022
```

```
cor.test(temp_df$bcg_cov_mean, temp_df$kmer_cov_mean, method = "pearson")
```

```
##
##  Pearson's product-moment correlation
##
## data:  temp_df$bcg_cov_mean and temp_df$kmer_cov_mean
## t = 3773.2, df = 16366, p-value < 0.00000000000000022
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9994079 0.9994431
## sample estimates:
##       cor
## 0.9994257
```

```
g <- ggplot(temp_df, aes(bcg_cov_mean, kmer_cov_mean)) +
    geom_point() +
    annotate("text", x = 2, y = 6, label = lm_eqn(ba_bcg_coverages_lm), parse = TRUE, size=6) +
    geom_smooth(method='lm')  +
    scale_y_continuous(breaks = seq(0, ceiling(max(temp_df$kmer_cov_mean)), by = 1)) +
    scale_x_continuous(breaks = seq(0, ceiling(max(temp_df$bcg_cov_mean)), by = 1))
p <- format_plot(g, xlab="BCerG31 Coverage", ylab="Ba31 Coverage")
print(p)
```



```
write_plot(p, 'supplementary-figure-03-ba31-bcerg31', height=6, width=12)
```

Ba31 Coverage and Genome Coverage

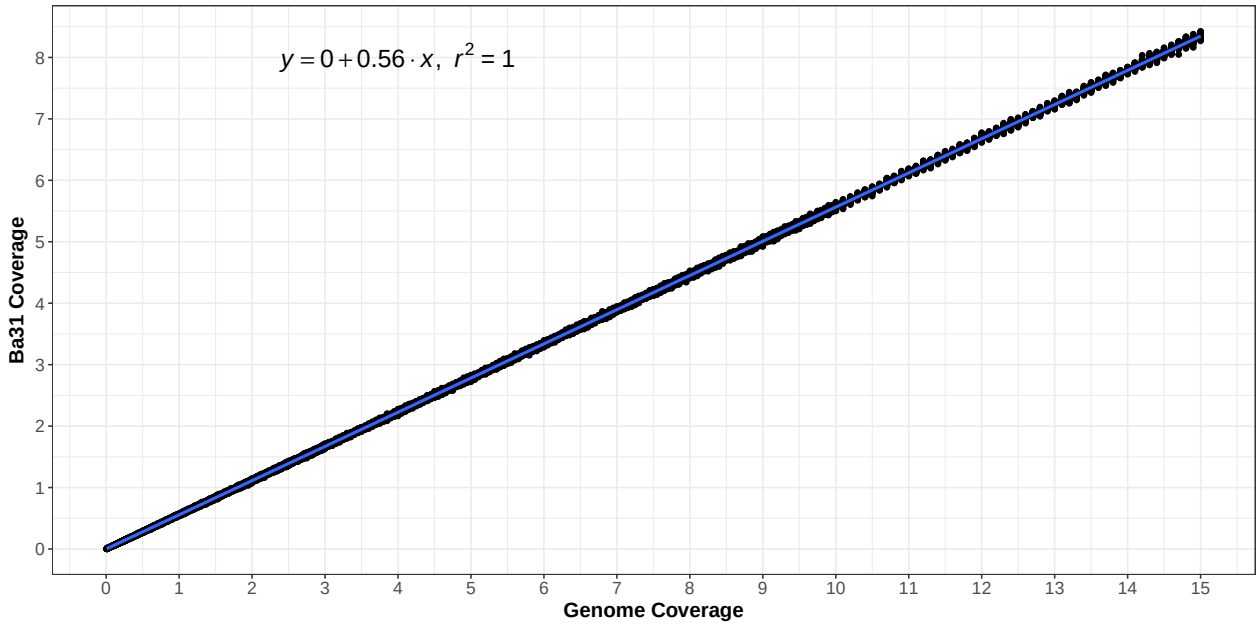
```
ba_genome_coverages_lm = lm(kmer_cov_mean ~ 0 + simulated_coverage, temp_df)
summary(ba_genome_coverages_lm)
```

```
##
## Call:
## lm(formula = kmer_cov_mean ~ 0 + simulated_coverage, data = temp_df)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.131475 -0.010506  0.000081  0.010775  0.132931
##
## Coefficients:
##                      Estimate Std. Error t value            Pr(>|t|)
## simulated_coverage 0.55641329 0.00002439   22814 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02067 on 16367 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 5.205e+08 on 1 and 16367 DF,  p-value: < 0.00000000000000022
```

```
cor.test(temp_df$kmer_cov_mean, temp_df$simulated_coverage,  method = "pearson")
```

```
##
##  Pearson's product-moment correlation
##
## data:  temp_df$kmer_cov_mean and temp_df$simulated_coverage
## t = 14192, df = 16366, p-value < 0.00000000000000022
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9999581 0.9999606
## sample estimates:
##       cor
## 0.9999594
```

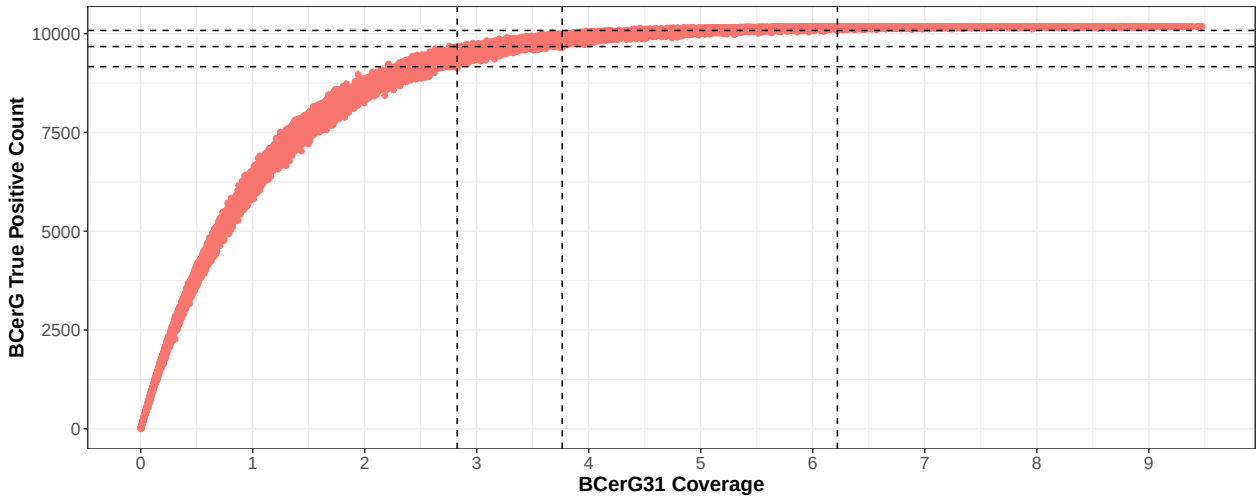
```
g <- ggplot(temp_df, aes(simulated_coverage, kmer_cov_mean)) +
    geom_point() +
    annotate("text", x = 4, y = 8, label = lm_eqn(ba_genome_coverages_lm), parse = TRUE, size=6) +
    geom_smooth(method='lm')  +
    scale_y_continuous(breaks = seq(0, ceiling(max(temp_df$kmer_cov_mean)), by = 1)) +
    scale_x_continuous(breaks = seq(0, ceiling(max(temp_df$simulated_coverage)), by = 1))
ba_coverage_plot <- format_plot(g, xlab="Genome Coverage", ylab="Ba31 Coverage")
print(ba_coverage_plot)
```




```
remove(temp_df)
remove(temp_total)
remove(temp_90)
remove(temp_95)
remove(temp_99)
remove(temp_999)
```

BCerG31 Sensitivity

```
temp_df <- simulation_bcg[simulation_bcg$is_bcg == 'True',]
temp_total <- max(temp_df$total_kmers)
temp_90 <- floor(temp_total*0.90)
temp_95 <- floor(temp_total*0.95)
temp_99 <- floor(temp_total*0.99)
temp_999 <- floor(temp_total*0.999)
g <- ggplot(temp_df, aes(kmer_cov_mean, tp)) +
    geom_point(aes(color=is_bcg)) +
    geom_hline(yintercept=temp_90, linetype="dashed") +
    geom_hline(yintercept=temp_95, linetype="dashed") +
    geom_hline(yintercept=temp_99, linetype="dashed") +
    geom_vline(xintercept=max(temp_df[temp_df$tp <= temp_90,]$kmer_cov_mean), linetype="dashed") +
    geom_vline(xintercept=max(temp_df[temp_df$tp <= temp_95,]$kmer_cov_mean), linetype="dashed") +
    geom_vline(xintercept=max(temp_df[temp_df$tp <= temp_99,]$kmer_cov_mean), linetype="dashed") +
    scale_x_continuous(breaks = seq(0, ceiling(max(temp_df$kmer_cov_mean)), by = 1))

p <- format_plot(g, xlab="BCerG31 Coverage", ylab="BCerG True Positive Count") +
    theme(legend.position="none")
print(p)
```




```
print(
    data.frame(
        percent=c("90%", "95%", "99%", "99.9%"),
        total=c(temp_90, temp_95, temp_99, temp_999),
        min=c(
            min(temp_df[temp_df$tp >= temp_90,]$kmer_cov_mean),
            min(temp_df[temp_df$tp >= temp_95,]$kmer_cov_mean),
            min(temp_df[temp_df$tp >= temp_99,]$kmer_cov_mean),
            min(temp_df[temp_df$tp >= temp_999,]$kmer_cov_mean)
        ),
        max=c(
            max(temp_df[temp_df$tp <= temp_90,]$kmer_cov_mean),
            max(temp_df[temp_df$tp <= temp_95,]$kmer_cov_mean),
            max(temp_df[temp_df$tp <= temp_99,]$kmer_cov_mean),
            max(temp_df[temp_df$tp <= temp_999,]$kmer_cov_mean)
        )
    )
)
```

```
##   percent total    min    max
## 1     90%  9164 2.2064 2.8252
## 2     95%  9673 2.7546 3.7627
## 3     99% 10081 4.1588 6.2195
## 4   99.9% 10172 5.3514 9.1855
```

BCerG31 coverage and Genome Coverage

```
bcg_genome_coverages_lm = lm(kmer_cov_mean ~ 0 + simulated_coverage, temp_df)
summary(bcg_genome_coverages_lm)
```

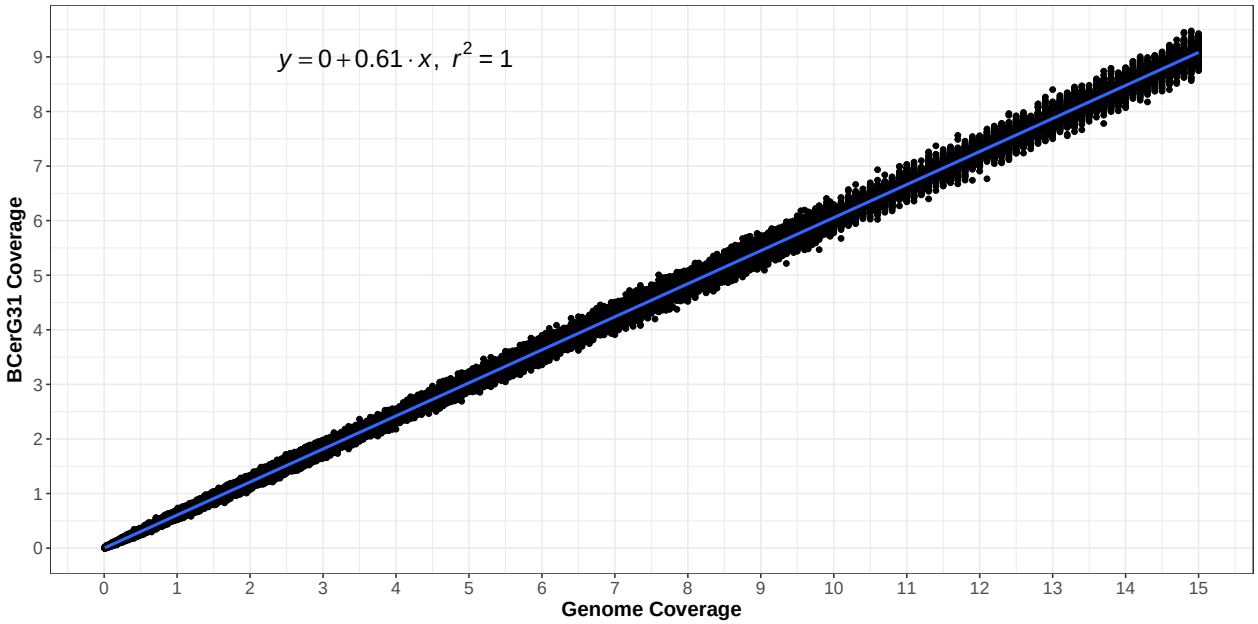
```
##
## Call:
## lm(formula = kmer_cov_mean ~ 0 + simulated_coverage, data = temp_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.55991 -0.04287 -0.00093  0.04227  0.52956
```

```
##
## Coefficients:
##                       Estimate Std. Error t value            Pr(>|t|)
## simulated_coverage 0.60547993 0.00005889   10282 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08616 on 48762 degrees of freedom
## Multiple R-squared:  0.9995, Adjusted R-squared:  0.9995
## F-statistic: 1.057e+08 on 1 and 48762 DF,  p-value: < 0.00000000000000022
```

```
cor.test(temp_df$kmer_cov_mean, temp_df$simulated_coverage,  method = "pearson")
```

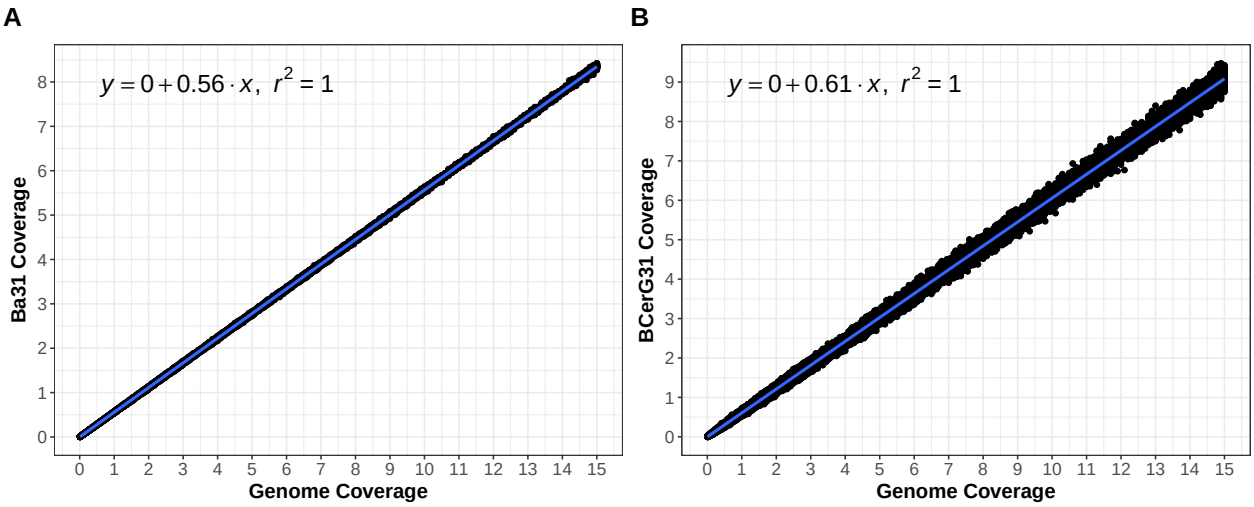
```
##
##  Pearson's product-moment correlation
##
## data:  temp_df$kmer_cov_mean and temp_df$simulated_coverage
## t = 6396.7, df = 48761, p-value < 0.00000000000000022
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9993940 0.9994152
## sample estimates:
##       cor
## 0.9994047
```

```
g <- ggplot(temp_df, aes(simulated_coverage, kmer_cov_mean)) +
    geom_point() +
    annotate("text", x = 4, y = 9, label = lm_eqn(bcg_genome_coverages_lm), parse = TRUE, size=6) +
    geom_smooth(method='lm')  +
    scale_y_continuous(breaks = seq(0, ceiling(max(temp_df$kmer_cov_mean)), by = 1)) +
    scale_x_continuous(breaks = seq(0, ceiling(max(temp_df$simulated_coverage)), by = 1))
bcg_coverage_plot <- format_plot(g, xlab="Genome Coverage", ylab="BCerG31 Coverage")
print(bcg_coverage_plot)
```



```
remove(temp_df)
remove(temp_total)
remove(temp_90)
remove(temp_95)
remove(temp_99)
remove(temp_999)
```

```
p1 <- arrangeGrob(ba_coverage_plot, top = textGrob(
    "A", x = unit(0, "npc"), y = unit(1, "npc"), just=c("left","top"),
     gp=gpar(col="black", fontsize=18, fontface="bold"))
)
p2 <- arrangeGrob(bcg_coverage_plot, top = textGrob(
    "B", x = unit(0, "npc"), y = unit(1, "npc"), just=c("left","top"),
     gp=gpar(col="black", fontsize=18, fontface="bold"))
)
plots=list()
plots[[1]] = p1
plots[[2]] = p2
grid.arrange(grobs=plots, ncol=2)
```




```
pdf(paste0(FIGURE, '/supplementary-figure-02-31-genome-coverage.pdf'),
    height=6, width=12, onefile=FALSE)
grid.arrange(grobs=plots, ncol=2)
dev_null <- dev.off()

png(paste0(FIGURE, '/supplementary-figure-02-31-genome-coverage.png'),
    height=600, width=1200)
grid.arrange(grobs=plots, ncol=2)
dev_null <- dev.off()
```

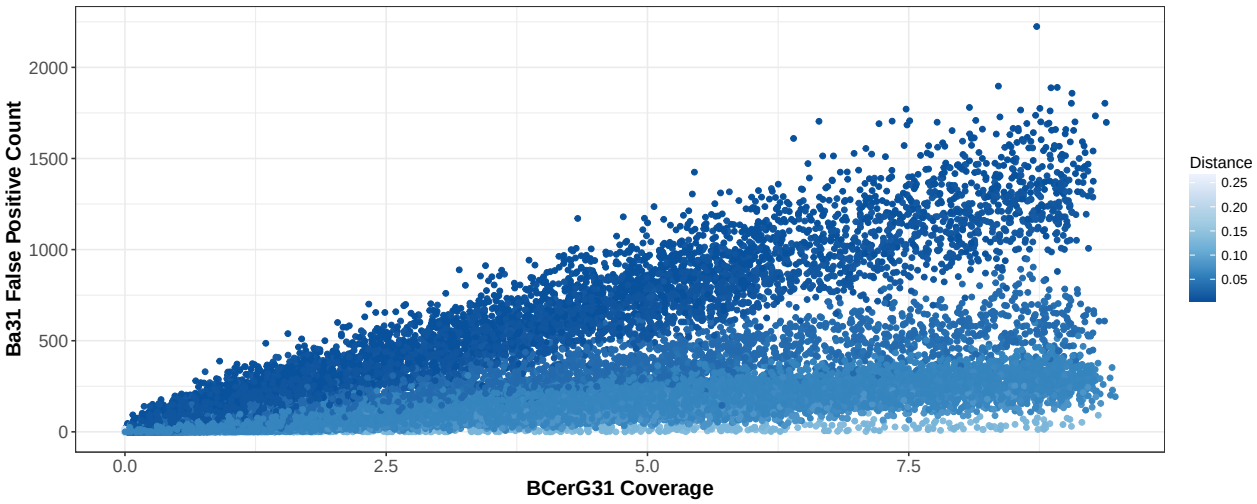
non-anthracis BCerG False Positive Model

Parse Results

```
bcg_model_set = read_samples(
    paste0(PATH, "/results/model-set/model-set-ba-summary.txt.gz"),
    ba_kmer_info
)

temp_df = read_samples(
    paste0(PATH, "/results/model-set/model-set-bcg-summary.txt.gz"),
    ba_kmer_info
)

bcg_model_set = merge(
    bcg_model_set,
    data.frame(
        sample=temp_df$sample,
        coverage=temp_df$coverage,
        replicate=temp_df$replicate,
        bcg_cov_mean=temp_df$kmer_cov_mean,
        bcg_non_zero_kmer_cov_mean=temp_df$non_zero_kmer_cov_mean
    ),
    by=c('sample', 'coverage', 'replicate')
)

bcg_model_set = merge(
    bcg_model_set,
    bcg_model_set %>%
        group_by(sample, coverage) %>%
        summarise(fp_mean=mean(fp), ba_mean=mean(kmer_cov_mean), bcg_mean=mean(bcg_cov_mean)),
    by=c('sample', 'coverage')
)
remove(temp_df)
```

BA31 False Positive Count and Distance

```
p <- ggplot(simulation_ba[simulation_ba$is_ba == 'False',],
            aes(bcg_cov_mean, fp, color=distance)) +
    geom_point()+
    xlab("BCerG31 Coverage") +
    ylab("Ba31 False Positive Count") +
    theme_bw() +
    theme(axis.text=element_text(size=12),
          axis.title=element_text(size=14,face="bold")) +
    scale_colour_distiller(palette="Blues") +
    labs(color='Distance')
print(p)
```




```
write_plot(p, 'supplementary-figure-04-distance-fp', height=6, width=12)
```

non-*B. anthracis* BCG Model Set

```
print(as.character(sort(unique(bcg_model_set$sample))))
```

```
## [1] "NC_011773"   "NZ_CP009335" "NZ_CP009596" "NZ_CP009605" "NZ_CP009720"
## [6] "NZ_CP018931" "NZ_CP018933" "NZ_CP018935"
```

Model BCerG kmer coverage and False Positive Match

```
bcg_model = lm(total_count ~ 0 + bcg_cov_mean, bcg_model_set)
anova(bcg_model)
```

```
## Analysis of Variance Table
##
## Response: total_count
##                  Df       Sum Sq      Mean Sq  F value
## bcg_cov_mean      1 719180371096 719180371096 43457509
## Residuals    311231   5150576486        16549
##                              Pr(>F)
## bcg_cov_mean < 0.00000000000000022 ***
## Residuals
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(bcg_model)
```

```
##
## Call:
## lm(formula = total_count ~ 0 + bcg_cov_mean, data = bcg_model_set)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1463.46   -53.09    -0.07    51.61  1794.24
##
```

```
## Coefficients:
##               Estimate Std. Error t value            Pr(>|t|)
## bcg_cov_mean 171.84251    0.02607    6592 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 128.6 on 311231 degrees of freedom
## Multiple R-squared:  0.9929, Adjusted R-squared:  0.9929
## F-statistic: 4.346e+07 on 1 and 311231 DF,  p-value: < 0.00000000000000022
```

Create a function to make predictions with 99% prediction intervals

```
predict_false_positive <- function(model, coverages) {
    temp_df <- cbind(
        data.frame(bcg_cov_mean=coverages),
        predict(
            model,
            newdata = data.frame(bcg_cov_mean=coverages),
            se=TRUE,
            interval='prediction',
            level=0.99
        )
    )
    return(temp_df)
}
```

Make a few predictions

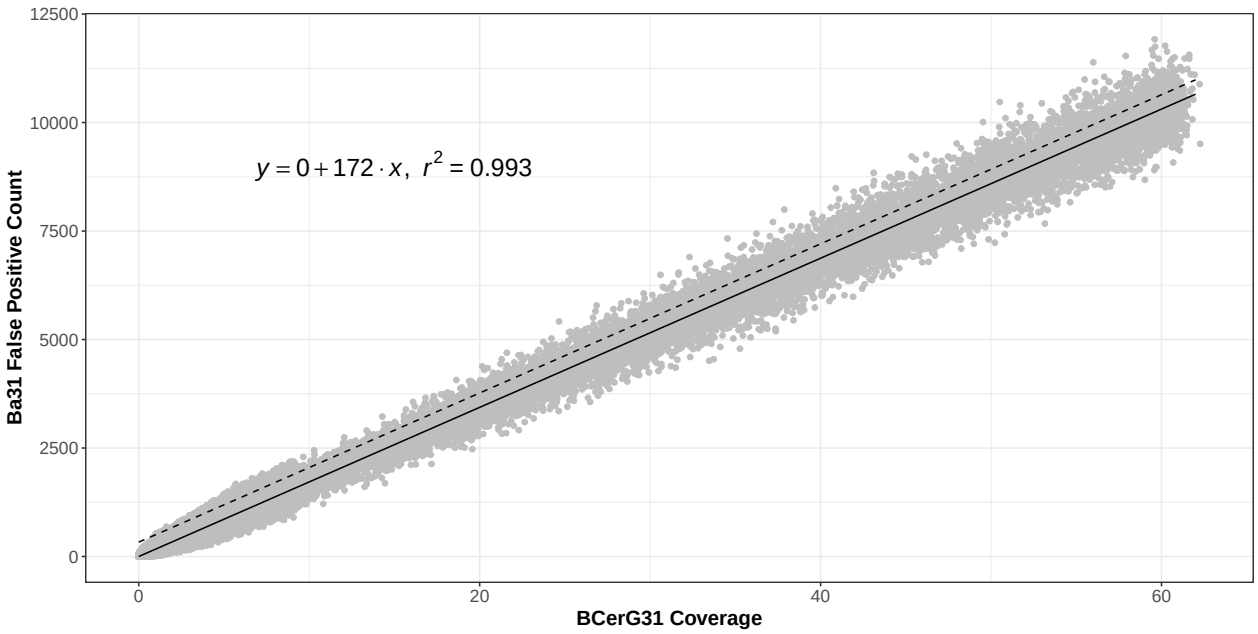
```
predict_false_positive(bcg_model, seq(0, 10, 2))
```

```
##   bcg_cov_mean   fit.fit   fit.lwr   fit.upr     se.fit     df
## 1            0    0.0000 -331.3647  331.3647 0.00000000 311231
## 2            2  343.6850   12.3203  675.0498 0.05213486 311231
## 3            4  687.3701  356.0052 1018.7349 0.10426972 311231
## 4            6 1031.0551  699.6901 1362.4200 0.15640457 311231
## 5            8 1374.7401 1043.3750 1706.1052 0.20853943 311231
## 6           10 1718.4251 1387.0598 2049.7905 0.26067429 311231
##   residual.scale
## 1       128.6431
## 2       128.6431
## 3       128.6431
## 4       128.6431
## 5       128.6431
## 6       128.6431
```

Plot: BCerG kmer Coverage and False Positive Matches

```
plot_model_fit <- function(samples, model, coverage, coverages) {
    predicted_fit <- predict_false_positive(bcg_model, coverages)
    g <- ggplot(samples[samples$bcg_cov_mean <= coverage,], aes(bcg_cov_mean, fp))+
```

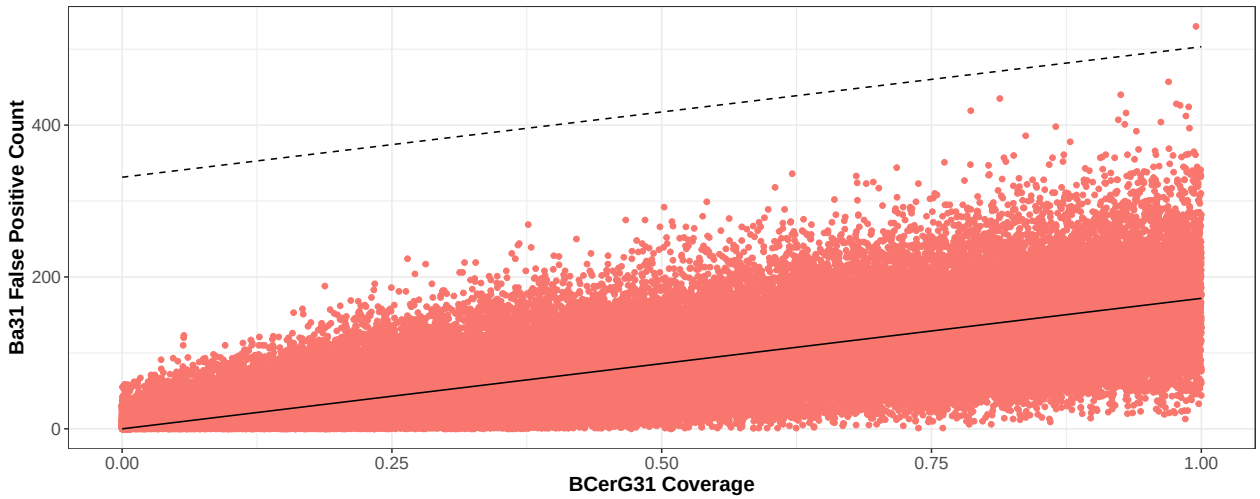
```
        geom_point(aes(color=is_ba)) +
        geom_line(data=predicted_fit, aes(x=bcg_cov_mean, y=fit.fit)) +
        geom_line(data=predicted_fit, aes(x=bcg_cov_mean, y=fit.upr), linetype="dashed")

    p <- format_plot(g, ylab="Ba31 False Positive Count", xlab="BCerG31 Coverage") +
        theme(legend.position="none")
    remove(predicted_fit)
    return(p)
}

predicted_fit <- predict_false_positive(
    bcg_model, seq(0, max(bcg_model_set$bcg_cov_mean), 1)
)
g <- ggplot(bcg_model_set, aes(bcg_cov_mean, total_count))+
    geom_point(color="gray") +
    annotate("text", x = 15, y = 9000, label = lm_eqn(bcg_model), parse = TRUE, size=6) +
    geom_line(data=predicted_fit, aes(x=bcg_cov_mean, y=fit.fit)) +
    geom_line(data=predicted_fit, aes(x=bcg_cov_mean, y=fit.upr), linetype="dashed") +
    theme(legend.position="none")

p <- format_plot(g, ylab="Ba31 False Positive Count", xlab="BCerG31 Coverage")
remove(predicted_fit)
print(p)
```




```
write_plot(p, 'figure-03-model', height=6, width=12)
```

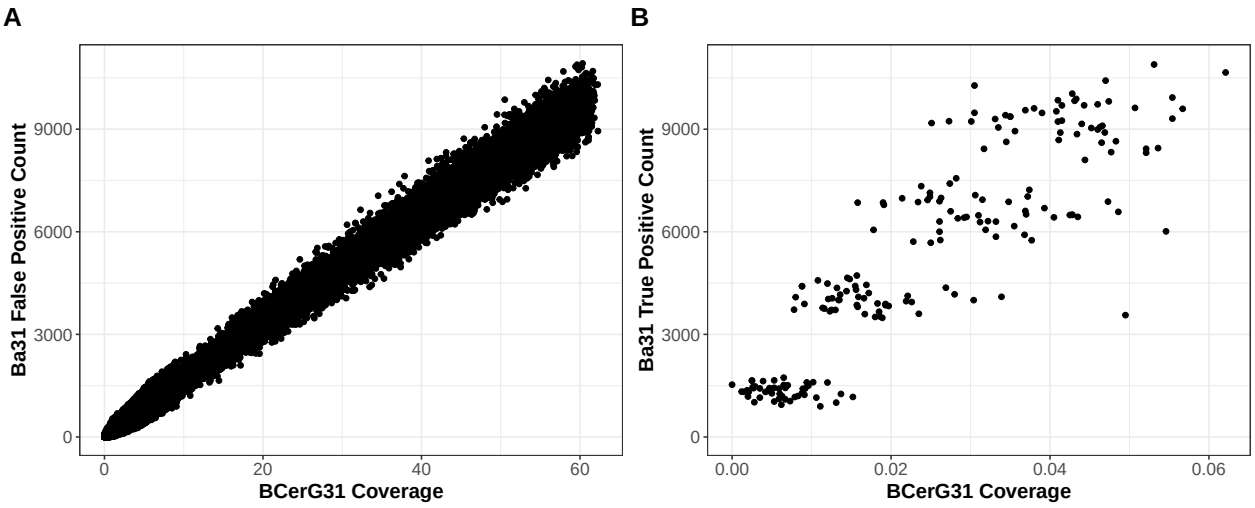
Plot the fit between 0 and 1x BCerG kmer coverage

```
p <- plot_model_fit(bcg_model_set, bcg_model, 1, seq(0, 1, 0.1))
print(p)
```



Plot: BCerG False positives and the equivalent true positive BCerG coverage using counts from simulations

```
matches = max(bcg_model_set$fp)
g <- ggplot(bcg_model_set, aes(bcg_cov_mean, fp))+
    geom_point()
p1 <- format_plot(g, ylab="Ba31 False Positive Count", xlab="BCerG31 Coverage") +
    theme(legend.position="none")

p1 <- arrangeGrob(p1, top = textGrob(
    "A", x = unit(0, "npc"), y = unit(1, "npc"), just=c("left","top"),
     gp=gpar(col="black", fontsize=18, fontface="bold"))
)

g <- ggplot(simulation_ba[simulation_ba$tp <= matches & simulation_ba$is_ba == 'True',],
            aes(bcg_cov_mean, tp)) +
    geom_point() +
    ylim(0, matches)


p2 <- format_plot(g, ylab="Ba31 True Positive Count", xlab="BCerG31 Coverage") +
    theme(legend.position="none")

p2 <- arrangeGrob(p2, top = textGrob(
    "B", x = unit(0, "npc"), y = unit(1, "npc"), just=c("left","top"),
     gp=gpar(col="black", fontsize=18, fontface="bold"))
)

grid.arrange(grobs=list(p1, p2), ncol=2)
```



B. anthracis Limit of Detection in *B. anthracis* and *B. cereus* Mixtures

Parse Results

```
kraken_report = read.table(
    paste0(PATH, "/results/limit-of-detection-ba31/kraken-report.txt"),
    header=TRUE, sep="\t"
)

subsample_ba_samples = read.table(
    gzfile(paste0(PATH, "/results/limit-of-detection-ba31/mixture-ba-summary.txt.gz")),
    header=TRUE, sep="\t"
)

subsample_bcg_samples = read.table(
    gzfile(paste0(PATH, "/results/limit-of-detection-ba31/mixture-bcg-summary.txt.gz")),
    header=TRUE, sep="\t"
)

subsample_ba_samples = merge(
    subsample_ba_samples,
    data.frame(
        replicate=subsample_bcg_samples$replicate,
        bcg_tp=subsample_bcg_samples$tp,
        bcg_coverage=subsample_bcg_samples$bcg_coverage,
        ba_coverage=subsample_bcg_samples$ba_coverage,
        bcg_cov_mean=subsample_bcg_samples$kmer_cov_mean
    ),
    by=c('replicate', 'ba_coverage', 'bcg_coverage')
)

subsample_summary <-  subsample_ba_samples %>%
    group_by(.dots=c('bcg_coverage', 'ba_coverage')) %>%
```

```
    summarise(
        ba31_hits=mean(tp + fp),
        ba31_coverage=mean(kmer_cov_mean),
        bcerg31_coverage=mean(bcg_cov_mean)
    )

subsample_summary$model_fit <- predict_false_positive(bcg_model, subsample_summary$bcerg31_coverage)$fi
subsample_summary$model_99pi <- predict_false_positive(bcg_model, subsample_summary$bcerg31_coverage)$f
subsample_summary$ba_detectable <- subsample_summary$ba31_hits > subsample_summary$model_99pi
subsample_summary$grey_zone <- subsample_summary$ba31_coverage < 0.1

subsample_summary <- merge(
    subsample_summary,
    kraken_report,
    by=c('bcg_coverage', 'ba_coverage')
)

exceeds_model <- function(mf, cov, count) {
    return(count > mf[mf$bcg_cov_mean == cov,]$fit.upr)
}

plot_ba_lod <- function(summary, bcg, ba) {
    temp_df <- summary[summary$bcg_coverage == bcg & summary$ba_coverage < ba,]

    predicted_fit <- predict_false_positive(
        bcg_model, unique(temp_df$bcerg31_coverage)
    )
    lod <- min(temp_df[
        apply(temp_df, 1, function(x) {
            exceeds_model(
              predicted_fit,
              x[["bcerg31_coverage"]],
              x[["ba31_hits"]]
            )
        })
    ,]$ba_coverage)
    temp_df$detectable <- ifelse(temp_df$ba_coverage >= lod, TRUE, FALSE)
    temp_df$ba_coverage_text <- ifelse(temp_df$ba_coverage == lod, paste0(lod, "x"), "")
    g <- ggplot(temp_df, aes(bcerg31_coverage, ba31_hits)) +
        geom_point(aes(color=detectable)) +
        # geom_text(aes(label=ba_coverage),hjust=-0.2, vjust=0) +
        geom_text(aes(label=ba_coverage_text),hjust=-0.2, vjust=0) +
        geom_line(data=predicted_fit, aes(x=bcg_cov_mean, y=fit.fit)) +
        geom_line(data=predicted_fit, aes(x=bcg_cov_mean, y=fit.upr), linetype="dashed")

    p <- format_plot(g, xlab="BCerG31 Coverage", ylab="Ba31 Count", title="Ba31 Detectable")
    remove(predicted_fit)
    remove(lod)
    return(p)
}

write.table(
    subsample_summary,
```

```
    paste0(PATH, "/results/limit-of-detection-ba31/ba31-lod-full-table.txt"),
    sep="\t",
    quote = FALSE,
    row.names = FALSE
)
```

B. anthracis Subsampled Coverages

A total of **39** *B. anthracis* subsamples were used.

```
print(sort(unique(subsample_ba_samples$ba_coverage)))
```

```
##  [1] 0.0000 0.0001 0.0002 0.0003 0.0004 0.0005 0.0006 0.0007 0.0008 0.0009
## [11] 0.0010 0.0020 0.0030 0.0040 0.0050 0.0060 0.0070 0.0080 0.0090 0.0100
## [21] 0.0200 0.0300 0.0400 0.0500 0.0600 0.0700 0.0800 0.0900 0.1000 0.1100
## [31] 0.1200 0.1300 0.1400 0.1500 0.1600 0.1700 0.1800 0.1900 0.2000
```

non-*B. anthracis* BCerG Subsample Coverages

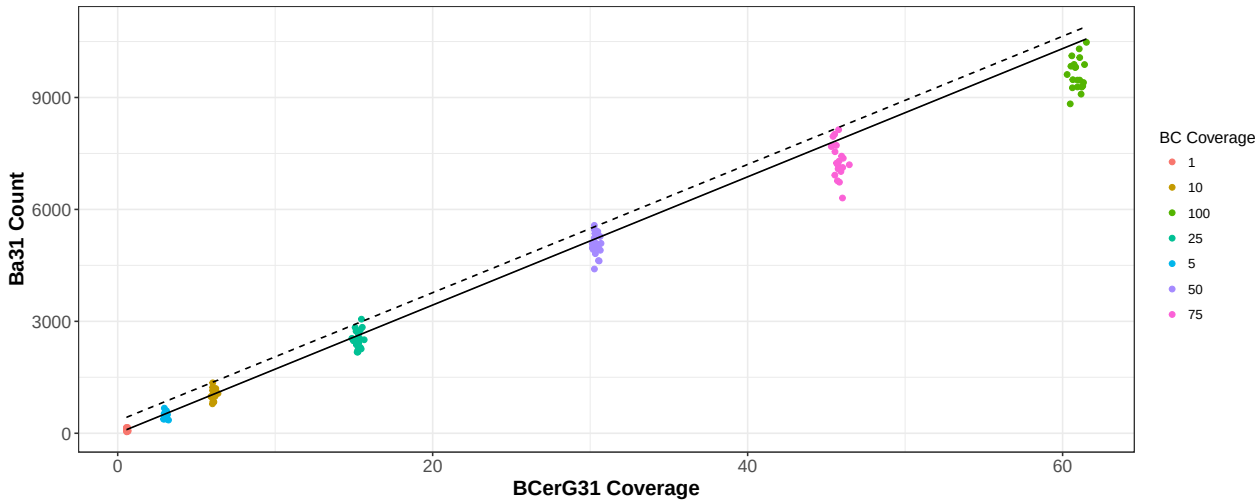
A total of **8** non-*B. anthracis* BCerG subsamples were used.

```
print(sort(unique(subsample_ba_samples$bcg_coverage)))
```

```
## [1]   0   1   5  10  25  50  75 100
```

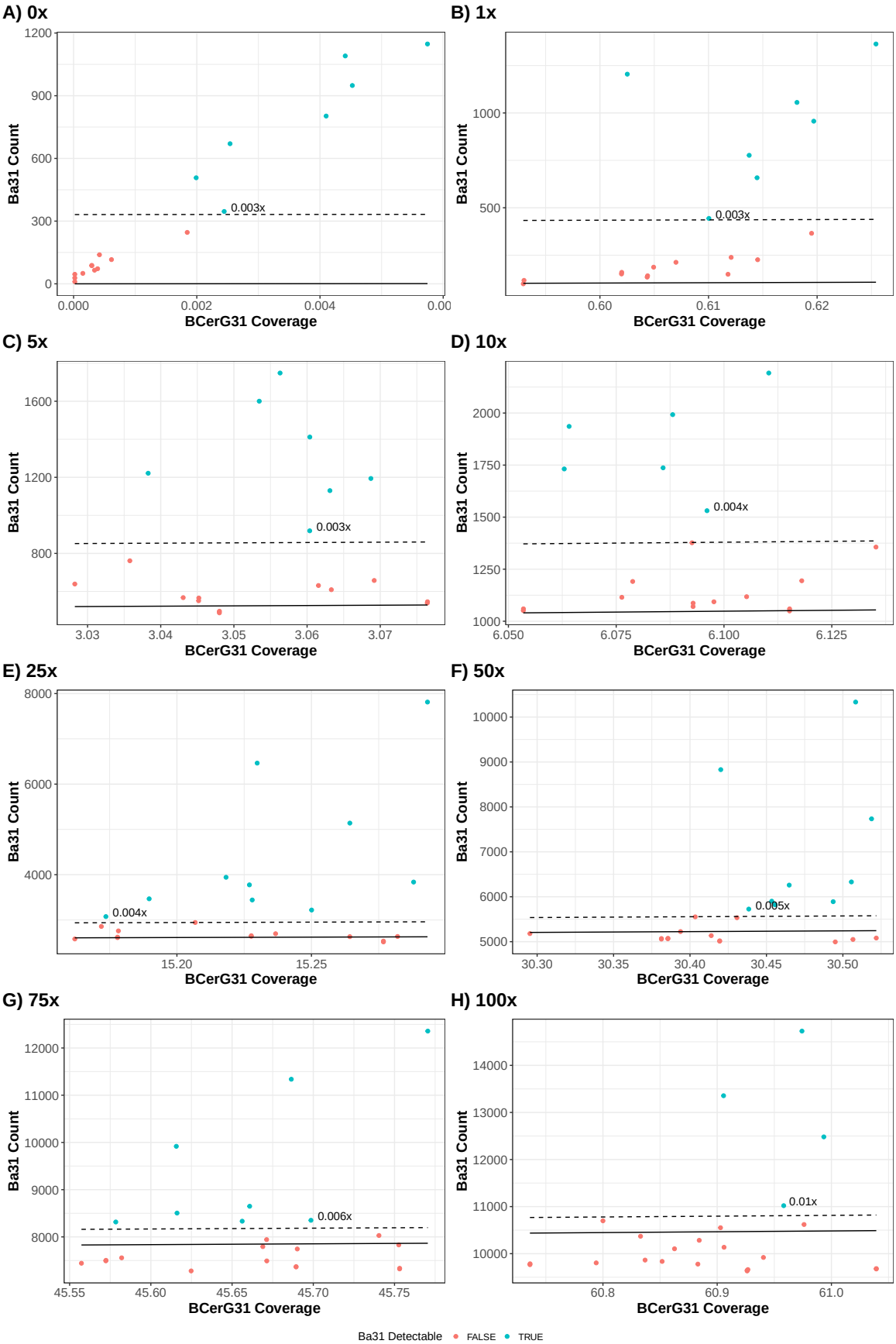
B. cereus Only

```
subsample_ba_samples$bcg_coverage_text <- as.character(subsample_ba_samples$bcg_coverage)
predicted_fit <- predict_false_positive(
    bcg_model, seq(min(subsample_ba_samples[subsample_ba_samples$ba_coverage == 0 ,]$bcg_cov_mean), max
)
g <- ggplot(subsample_ba_samples[subsample_ba_samples$ba_coverage == 0 ,], aes(bcg_cov_mean, fp)) +
    geom_point(aes(color=bcg_coverage_text)) +
    geom_line(data=predicted_fit, aes(x=bcg_cov_mean, y=fit.fit)) +
    geom_line(data=predicted_fit, aes(x=bcg_cov_mean, y=fit.upr), linetype="dashed")
p <- format_plot(g, xlab="BCerG31 Coverage", ylab="Ba31 Count", title= "BC Coverage")
print(p)
```




```
p1 <- plot_ba_lod(subsample_summary, 0, 0.01)
p2 <- plot_ba_lod(subsample_summary, 1, 0.01)
p3 <- plot_ba_lod(subsample_summary, 5, 0.01)
p4 <- plot_ba_lod(subsample_summary, 10, 0.01)
p5 <- plot_ba_lod(subsample_summary, 25, 0.05)
p6 <- plot_ba_lod(subsample_summary, 50, 0.05)
p7 <- plot_ba_lod(subsample_summary, 75, 0.05)
p8 <- plot_ba_lod(subsample_summary, 100, 0.05)

grid_arrange_shared_ba31 <- function(..., ncol = length(list(...)), nrow = 1, position = c("bottom", "r
  plots <- list(...)
  position <- match.arg(position)
  g <- ggplotGrob(plots[[1]] + theme(legend.position = position))$grobs
  legend <- g[[which(sapply(g, function(x) x$name) == "guide-box")]]
  lheight <- sum(legend$height)
  lwidth <- sum(legend$width)
  gl <- lapply(plots, function(x) x + theme(legend.position="none"))
  gl[[1]] <-arrangeGrob(gl[[1]], top = textGrob(
        "A) 0x", x = unit(0, "npc"), y = unit(1, "npc"), just=c("left","top"),
         gp=gpar(col="black", fontsize=18, fontface="bold"))
    )
  gl[[2]] <-arrangeGrob(gl[[2]], top = textGrob(
        "B) 1x", x = unit(0, "npc"), y = unit(1, "npc"), just=c("left","top"),
         gp=gpar(col="black", fontsize=18, fontface="bold"))
    )
  gl[[3]] <-arrangeGrob(gl[[3]], top = textGrob(
        "C) 5x", x = unit(0, "npc"), y = unit(1, "npc"), just=c("left","top"),
         gp=gpar(col="black", fontsize=18, fontface="bold"))
    )
  gl[[4]] <-arrangeGrob(gl[[4]], top = textGrob(
        "D) 10x", x = unit(0, "npc"), y = unit(1, "npc"), just=c("left","top"),
         gp=gpar(col="black", fontsize=18, fontface="bold"))
    )
  gl[[5]] <-arrangeGrob(gl[[5]], top = textGrob(
        "E) 25x", x = unit(0, "npc"), y = unit(1, "npc"), just=c("left","top"),
         gp=gpar(col="black", fontsize=18, fontface="bold"))
    )
  gl[[6]] <-arrangeGrob(gl[[6]], top = textGrob(
```

```
        "F) 50x", x = unit(0, "npc"), y = unit(1, "npc"), just=c("left","top"),
         gp=gpar(col="black", fontsize=18, fontface="bold"))
    )
  gl[[7]] <-arrangeGrob(gl[[7]], top = textGrob(
        "G) 75x", x = unit(0, "npc"), y = unit(1, "npc"), just=c("left","top"),
         gp=gpar(col="black", fontsize=18, fontface="bold"))
    )
  gl[[8]] <-arrangeGrob(gl[[8]], top = textGrob(
        "H) 100x", x = unit(0, "npc"), y = unit(1, "npc"), just=c("left","top"),
         gp=gpar(col="black", fontsize=18, fontface="bold"))
    )

  gl <- c(gl, ncol = ncol, nrow = nrow)

  combined <- switch(position,
                     "bottom" = arrangeGrob(do.call(arrangeGrob, gl),
                                            legend,
                                            ncol = 1,
                                            heights = unit.c(unit(1, "npc") - lheight, lheight)),
                     "right" = arrangeGrob(do.call(arrangeGrob, gl),
                                           legend,
                                           ncol = 2,
                                           widths = unit.c(unit(1, "npc") - lwidth, lwidth)))

  grid.newpage()
  grid.draw(combined)

  # return gtable invisibly
  invisible(combined)
}

pdf(paste0(FIGURE, '/supplementary-figure-05-ba31-lod.pdf'), onefile=FALSE,
    height=12, width=8)
grid_arrange_shared_ba31(p1, p2, p3, p4, p5, p6, p7, p8, ncol=2, nrow=4)
dev_null <- dev.off()

png(paste0(FIGURE, '/supplementary-figure-05-ba31-lod.png'), height=1200, width=800)
grid_arrange_shared_ba31(p1, p2, p3, p4, p5, p6, p7, p8, ncol=2, nrow=4)
dev_null <- dev.off()

grid_arrange_shared_ba31(p1, p2, p3, p4, p5, p6, p7, p8, ncol=2, nrow=4)
```


A) 0x
B) 1x
C) 5x
D) 10x
E) 25x
F) 50x
G) 75x
H) 100x
Ba31 Count
BCerG31 Coverage
0.003x
0.004x
0.005x
0.006x
0.01x
Ba31 Detectable
FALSE
TRUE

```
remove(p1)
remove(p2)
remove(p3)
remove(p4)
remove(p5)
remove(p6)
remove(p7)
remove(p8)
```

NYC Subway

Inputs

```
nyc_samples = read.table(
    gzfile(paste0(PATH, "/results/nyc/nyc-ba-summary.txt.gz")),
    header=TRUE, sep="\t"
)

temp_df = read.table(
    gzfile(paste0(PATH, "/results/nyc/nyc-bcg-summary.txt.gz")),
    header=TRUE, sep="\t"
)

nyc_samples = merge(
    nyc_samples,
    data.frame(
        run=temp_df$run,
        bcerg31_coverage=temp_df$kmer_cov_mean,
        bcerg31_hit=temp_df$hit
    ),
    by=c('run')
)

temp_df = read.table(
    gzfile(paste0(PATH, "/results/nyc/nyc-lef-summary.txt.gz")),
    header=TRUE, sep="\t"
)

nyc_samples = merge(
    nyc_samples,
    data.frame(
        run=temp_df$run,
        lef31_hit=temp_df$hit
    ),
    by=c('run')
)

remove(temp_df)
names(nyc_samples)[names(nyc_samples) == 'hit'] <- 'ba31_hits'
names(nyc_samples)[names(nyc_samples) == 'kmer_cov_mean'] <- 'ba31_coverage'
colnames(nyc_samples)
```

```
##  [1] "run"                       "is_bcg"
##  [3] "is_ba"                     "has_lethal"
##  [5] "group"                     "total_kmers"
##  [7] "ba31_hits"                 "miss"
##  [9] "kmer_cov_min"              "ba31_coverage"
## [11] "kmer_cov_median"           "kmer_cov_max"
## [13] "non_zero_kmer_cov_min"     "non_zero_kmer_cov_mean"
## [15] "non_zero_kmer_cov_median" "non_zero_kmer_cov_max"
## [17] "bcerg31_coverage"          "bcerg31_hit"
## [19] "lef31_hit"
```

Summary of NYC samples

```
nyc_summary <- nyc_samples[,c("run", "bcerg31_coverage", "ba31_hits", "lef31_hit")]
nyc_summary$model_fit <- predict_false_positive(bcg_model, nyc_summary$bcerg31_coverage)$fit.fit
nyc_summary$model_99pi <- predict_false_positive(bcg_model, nyc_summary$bcerg31_coverage)$fit.upr
nyc_summary$exceeds_99pi <- nyc_summary$ba31_hits > nyc_summary$model_99pi
nyc_summary$gray_zone <- nyc_summary$bcerg31_coverage < 0.1
head(nyc_summary)
```

```
##       run bcerg31_coverage ba31_hits lef31_hit  model_fit model_99pi
## 1 GCSS-00      0.001964058         0         0  0.3375086   331.7022
## 2 GCSS-01      0.146224099         6         0 25.1275166   356.4922
## 3 GCSS-02      0.018854954         6         0  3.2400827   334.6048
## 4 GCSS-03      0.017381911         6         0  2.9869513   334.3516
## 5 GCSS-04      0.008838260         0         0  1.5187888   332.8835
## 6 GCSS-05      0.027103997         6         0  4.6576189   336.0223
##   exceeds_99pi gray_zone
## 1        FALSE      TRUE
## 2        FALSE     FALSE
## 3        FALSE      TRUE
## 4        FALSE      TRUE
## 5        FALSE      TRUE
## 6        FALSE      TRUE
```

```
write.table(
    nyc_summary,
    paste0(PATH, "/results/nyc/nyc-summary.txt"),
    sep="\t",
    quote = FALSE,
    row.names = FALSE
)
```

NYC Samples which 0 Ba31 matches

```
nrow(nyc_summary[nyc_summary$ba31_hits == 0,])
```

```
## [1] 373
```

```
nrow(nyc_summary[nyc_summary$ba31_hits > 0 & nyc_summary$exceeds_99pi == FALSE,])
```

```
## [1] 1051
```

NYC Samples which exceed 99% Prediction interval

```
nrow(nyc_summary[nyc_summary$exceeds_99pi,])
```

```
## [1] 34
```

In the Gray Zone (lef not detectable)

```
nyc_summary[nyc_summary$exceeds_99pi & nyc_summary$gray_zone,]
```

```
##        run bcerg31_coverage ba31_hits lef31_hit model_fit model_99pi
## 689 P00738      0.002160464       425         0 0.3712595    331.736
##     exceeds_99pi gray_zone
## 689         TRUE      TRUE
```

Outside the Gray Zone (lef should have been detected)

```
nyc_summary[nyc_summary$exceeds_99pi & nyc_summary$gray_zone == FALSE,]
```

```
##         run bcerg31_coverage ba31_hits lef31_hit  model_fit model_99pi
## 107  P00139        0.7789453       793         0  133.85592   465.2206
## 163  P00200        7.9085731      2114         0 1359.02908  1690.3942
## 167  P00204        2.0751252       993         0  356.59473   687.9595
## 173  P00210        2.6157321      1884         0  449.49398   780.8587
## 176  P00213        0.2041638       423         0   35.08402   366.4487
## 180  P00217        0.2259648      1103         0   38.83037   370.1951
## 204  P00241        0.4130413       484         0   70.97806   402.3428
## 251  P00294        0.4412256       430         0   75.82131   407.1860
## 277  P00320        1.7179613       715         0  295.21879   626.5835
## 293  P00336        0.4416184       439         0   75.88881   407.2535
## 301  P00344        2.1282530       847         0  365.72434   697.0891
## 304  P00348        1.6077777      3380         0  276.28455   607.6493
## 324  P00369        0.5699696      1951         0   97.94500   429.3097
## 359  P00404        0.5555337      1299         0   95.46431   426.8290
## 403  P00450        0.2016105       927         0   34.64526   366.0100
## 451  P00498        1.6161249      1602         0  277.71897   609.0837
## 457  P00504       10.7371109      2205         0 1845.09211  2176.4576
## 486  P00535        0.2464892      1621         0   42.35733   373.7220
## 578  P00627        0.6937052      1040         0  119.20804   450.5727
## 733  P00783        0.2456054      1045         0   42.20545   373.5702
## 741  P00791        0.4188353       951         0   71.97371   403.3384
## 916  P00970        3.2344103      1206         0  555.80919   887.1740
## 923  P00977        0.3976235       426         0   68.32862   399.6933
## 927  P00981        1.3215163     20079         0  227.09267   558.4574
## 940  P00994        0.1888442       375         0   32.45145   363.8162
## 951  P01005        0.2707454       865         0   46.52556   377.8903
## 953  P01007        1.8512226      1610         0  318.11875   649.4835
## 955  P01009        0.2562113       489         0   44.02800   375.3927
## 1010 P01071        2.4528135       893         0  421.49764   752.8624
## 1055 P01116        0.1215752       731         0   20.89178   352.2565
## 1261 P01337        0.5111460      1840         0   87.83662   419.2013
## 1271 P01347        0.1404301       566         0   24.13187   355.4966
```

```
## 1304 P01380        0.4721595       981         0   81.13707   412.5018
##      exceeds_99pi gray_zone
## 107          TRUE     FALSE
## 163          TRUE     FALSE
## 167          TRUE     FALSE
## 173          TRUE     FALSE
## 176          TRUE     FALSE
## 180          TRUE     FALSE
## 204          TRUE     FALSE
## 251          TRUE     FALSE
## 277          TRUE     FALSE
## 293          TRUE     FALSE
## 301          TRUE     FALSE
## 304          TRUE     FALSE
## 324          TRUE     FALSE
## 359          TRUE     FALSE
## 403          TRUE     FALSE
## 451          TRUE     FALSE
## 457          TRUE     FALSE
## 486          TRUE     FALSE
## 578          TRUE     FALSE
## 733          TRUE     FALSE
## 741          TRUE     FALSE
## 916          TRUE     FALSE
## 923          TRUE     FALSE
## 927          TRUE     FALSE
## 940          TRUE     FALSE
## 951          TRUE     FALSE
## 953          TRUE     FALSE
## 955          TRUE     FALSE
## 1010         TRUE     FALSE
## 1055         TRUE     FALSE
## 1261         TRUE     FALSE
## 1271         TRUE     FALSE
## 1304         TRUE     FALSE
```

Session Info

```
sessionInfo()
```

```
## R version 3.4.3 (2017-11-30)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.2 LTS
##
## Matrix products: default
## BLAS: /usr/lib/libblas/libblas.so.3.6.0
## LAPACK: /usr/lib/lapack/liblapack.so.3.6.0
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
```

```
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] bindrcpp_0.2  gridExtra_2.3 ggplot2_2.2.1 dplyr_0.7.4
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.15       knitr_1.20         bindr_0.1.1
##  [4] magrittr_1.5       munsell_0.4.3      colorspace_1.3-2
##  [7] R6_2.2.2           rlang_0.1.6        plyr_1.8.4
## [10] stringr_1.2.0      tools_3.4.3        gtable_0.2.0
## [13] htmltools_0.3.6    lazyeval_0.2.1     yaml_2.1.18
## [16] assertthat_0.2.0   rprojroot_1.3-2    digest_0.6.15
## [19] tibble_1.4.2       RColorBrewer_1.1-2 glue_1.2.0
## [22] evaluate_0.10.1    rmarkdown_1.9      labeling_0.3
## [25] stringi_1.1.6      compiler_3.4.3     pillar_1.1.0
## [28] scales_0.5.0       backports_1.1.2    pkgconfig_2.0.1
```