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) * 100
    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)
}
```

```r
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

```r
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

```r
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
## 2    0.01645     1    ba   100%      2   1.219512
## 3    0.01724     1    ba   100%      3   1.829268
## 4    0.01806     1    ba   100%      4   2.439024
## 5    0.01833     1    ba   100%      5   3.048780
## 6    0.01890     1    ba   100%      6   3.658537
## 7    0.01904     1    ba   100%      7   4.268293
## 8    0.01970     1    ba   100%      8   4.878049
```

```
## 9    0.01989     1    ba   100%      9   5.487805
## 10   0.02065     1    ba   100%     10   6.097561
## 11   0.02071     1    ba   100%     11   6.707317
## 12   0.02136     1    ba   100%     12   7.317073
## 13   0.02147     1    ba   100%     13   7.926829
## 14   0.02159     1    ba   100%     14   8.536585
## 15   0.02206     1    ba   100%     15   9.146341
## 16   0.02367     1    ba   100%     16   9.756098
## 17   0.02390     1    ba   100%     17  10.365854
## 18   0.02453     1    ba   100%     18  10.975610
## 19   0.02483     1    ba   100%     19  11.585366
## 20   0.02501     1    ba   100%     20  12.195122
## 21   0.02514     1    ba   100%     21  12.804878
## 22   0.02531     1    ba   100%     22  13.414634
## 23   0.02551     1    ba   100%     23  14.024390
## 24   0.02558     1    ba   100%     24  14.634146
## 25   0.02594     1    ba   100%     25  15.243902
## 26   0.02600     1    ba   100%     26  15.853659
## 27   0.02668     1    ba   100%     27  16.463415
## 28   0.02681     1    ba   100%     28  17.073171
## 29   0.02701     1    ba   100%     29  17.682927
## 30   0.02741     1    ba   100%     30  18.292683
## 31   0.02784     1    ba   100%     31  18.902439
## 32   0.02933     1    ba   100%     32  19.512195
## 33   0.02943     1    ba   100%     33  20.121951
## 34   0.02950     1    ba   100%     34  20.731707
## 35   0.02974     1    ba   100%     35  21.341463
## 36   0.02995     1    ba   100%     36  21.951220
## 37   0.03002     1    ba   100%     37  22.560976
## 38   0.03022     1    ba   100%     38  23.170732
## 39   0.03063     1    ba   100%     39  23.780488
## 40   0.03076     1    ba   100%     40  24.390244
## 41   0.03117     1    ba   100%     41  25.000000
## 42   0.03131     1    ba   100%     42  25.609756
## 43   0.03159     1    ba   100%     43  26.219512
## 44   0.03163     1    ba   100%     44  26.829268
## 45   0.03172     1    ba   100%     45  27.439024
## 46   0.03199     1    ba   100%     46  28.048780
## 47   0.03211     1    ba   100%     47  28.658537
## 48   0.03227     1    ba   100%     48  29.268293
## 49   0.03228     1    ba   100%     49  29.878049
## 50   0.03243     1    ba   100%     50  30.487805
## 51   0.03249     1    ba   100%     51  31.097561
## 52   0.03263     1    ba   100%     52  31.707317
## 53   0.03278     1    ba   100%     53  32.317073
## 54   0.03312     1    ba   100%     54  32.926829
## 55   0.03315     1    ba   100%     55  33.536585
## 56   0.03323     1    ba   100%     56  34.146341
## 57   0.03334     1    ba   100%     57  34.756098
## 58   0.03340     1    ba   100%     58  35.365854
## 59   0.03373     1    ba   100%     59  35.975610
## 60   0.03420     1    ba   100%     60  36.585366
## 61   0.03424     1    ba   100%     61  37.195122
## 62   0.03432     1    ba   100%     62  37.804878
```

```
## 63   0.03492     1    ba   100%     63  38.414634
## 64   0.03499     2    ba   100%     65  39.634146
## 65   0.03501     1    ba   100%     66  40.243902
## 66   0.03537     1    ba   100%     67  40.853659
## 67   0.03538     1    ba   100%     68  41.463415
## 68   0.03555     1    ba   100%     69  42.073171
## 69   0.03585     1    ba   100%     70  42.682927
## 70   0.03596     1    ba   100%     71  43.292683
## 71   0.03651     1    ba   100%     72  43.902439
## 72   0.03661     1    ba   100%     73  44.512195
## 73   0.03720     1    ba   100%     74  45.121951
## 74   0.03723     1    ba   100%     75  45.731707
## 75   0.03725     1    ba   100%     76  46.341463
## 76   0.03814     2    ba   100%     78  47.560976
## 77   0.03844     1    ba   100%     79  48.170732
## 78   0.03845     1    ba   100%     80  48.780488
## 79   0.03869     1    ba   100%     81  49.390244
## 80   0.03949     1    ba   100%     82  50.000000
## 81   0.03953     1    ba   100%     83  50.609756
## 82   0.03965     1    ba   100%     84  51.219512
## 83   0.03969     1    ba   100%     85  51.829268
## 84   0.03971     1    ba   100%     86  52.439024
## 85   0.03973     1    ba   100%     87  53.048780
## 86   0.03983     2    ba   100%     89  54.268293
## 87   0.04006     1    ba   100%     90  54.878049
## 88   0.04025     1    ba   100%     91  55.487805
## 89   0.04043     1    ba   100%     92  56.097561
## 90   0.04049     1    ba   100%     93  56.707317
## 91   0.04052     1    ba   100%     94  57.317073
## 92   0.04074     1    ba   100%     95  57.926829
## 93   0.04090     1    ba   100%     96  58.536585
## 94   0.04099     1    ba   100%     97  59.146341
## 95   0.04126     1    ba   100%     98  59.756098
## 96   0.04129     1    ba   100%     99  60.365854
## 97   0.04144     1    ba   100%    100  60.975610
## 98   0.04167     1    ba   100%    101  61.585366
## 99   0.04231     1    ba   100%    102  62.195122
## 100  0.04245     1    ba   100%    103  62.804878
## 101  0.04285     1    ba   100%    104  63.414634
## 102  0.04326     1    ba   100%    105  64.024390
## 103  0.04351     2    ba   100%    107  65.243902
## 104  0.04373     1    ba   100%    108  65.853659
## 105  0.04380     1    ba   100%    109  66.463415
## 106  0.04571     2    ba   100%    111  67.682927
## 107  0.04576     1    ba   100%    112  68.292683
## 108  0.04584     1    ba   100%    113  68.902439
## 109  0.04637     1    ba   100%    114  69.512195
## 110  0.04661     1    ba   100%    115  70.121951
## 111  0.04670     1    ba   100%    116  70.731707
## 112  0.04713     1    ba   100%    117  71.341463
## 113  0.04751     1    ba   100%    118  71.951220
## 114  0.04779     1    ba   100%    119  72.560976
## 115  0.04809     1    ba   100%    120  73.170732
## 116  0.04823     1    ba   100%    121  73.780488
```

```
## 117  0.04829     1    ba   100%    122  74.390244
## 118  0.04836     1    ba   100%    123  75.000000
## 119  0.04918     1    ba   100%    124  75.609756
## 120  0.04936     1    ba   100%    125  76.219512
## 121  0.04964     1    ba   100%    126  76.829268
## 122  0.04998     1    ba   100%    127  77.439024
## 123  0.05054     1    ba   100%    128  78.048780
## 124  0.05062     1    ba   100%    129  78.658537
## 125  0.05125     1    ba   100%    130  79.268293
## 126  0.05147     1    ba   100%    131  79.878049
## 127  0.05247     1    ba   100%    132  80.487805
## 128  0.05255     1    ba   100%    133  81.097561
## 129  0.05322     1    ba   100%    134  81.707317
## 130  0.05335     1    ba   100%    135  82.317073
## 131  0.05369     1    ba   100%    136  82.926829
## 132  0.05373     1    ba   100%    137  83.536585
## 133  0.05445     1    ba   100%    138  84.146341
## 134  0.05582     1    ba   100%    139  84.756098
## 135  0.05661     1    ba   100%    140  85.365854
## 136  0.05732     1    ba   100%    141  85.975610
## 137  0.05740     1    ba   100%    142  86.585366
## 138  0.05792     1    ba   100%    143  87.195122
## 139  0.05884     1    ba   100%    144  87.804878
## 140  0.06049     1    ba   100%    145  88.414634
## 141  0.06123     1    ba   100%    146  89.024390
## 142  0.06237     1    ba   100%    147  89.634146
## 143  0.07122     1    ba   100%    148  90.243902
## 144  0.07932     1    ba   100%    149  90.853659
## 145  0.08033     1    ba   100%    150  91.463415
## 146  0.08270     1    ba   100%    151  92.073171
## 147  0.08536     1    ba   100%    152  92.682927
## 148  0.09070     1    ba   100%    153  93.292683
## 149  0.09600     1    ba   100%    154  93.902439
## 150  0.10695     1    ba   100%    155  94.512195
## 151  0.11361     1    ba   100%    156  95.121951
## 152  0.11532     1    ba   100%    157  95.731707
## 153  0.11549     1    ba   100%    158  96.341463
## 154  0.11791     1    ba   100%    159  96.951220
## 155  0.12036     1    ba   100%    160  97.560976
## 156  0.12039     1    ba   100%    161  98.170732
## 157  0.12113     1    ba   100%    162  98.780488
## 158  0.12284     1    ba   100%    163  99.390244
## 159  0.13870     1    ba   100%    164 100.000000
## 161  0.00905     1    ba    95%      2   1.219512
## 162  0.00916     1    ba    95%      3   1.829268
## 163  0.00956     1    ba    95%      4   2.439024
## 164  0.01157     1    ba    95%      5   3.048780
## 165  0.01239     1    ba    95%      6   3.658537
## 166  0.01284     1    ba    95%      7   4.268293
## 167  0.01298     1    ba    95%      8   4.878049
## 168  0.01312     1    ba    95%      9   5.487805
## 169  0.01336     1    ba    95%     10   6.097561
## 170  0.01344     1    ba    95%     11   6.707317
## 171  0.01489     1    ba    95%     12   7.317073
```

```
## 172  0.01494     1    ba    95%     13   7.926829
## 173  0.01504     1    ba    95%     14   8.536585
## 174  0.01507     1    ba    95%     15   9.146341
## 175  0.01523     1    ba    95%     16   9.756098
## 176  0.01531     1    ba    95%     17  10.365854
## 177  0.01559     1    ba    95%     18  10.975610
## 178  0.01568     1    ba    95%     19  11.585366
## 179  0.01573     1    ba    95%     20  12.195122
## 180  0.01584     1    ba    95%     21  12.804878
## 181  0.01635     1    ba    95%     22  13.414634
## 182  0.01657     1    ba    95%     23  14.024390
## 183  0.01659     1    ba    95%     24  14.634146
## 184  0.01673     1    ba    95%     25  15.243902
## 185  0.01689     1    ba    95%     26  15.853659
## 186  0.01691     1    ba    95%     27  16.463415
## 187  0.01706     1    ba    95%     28  17.073171
## 188  0.01719     1    ba    95%     29  17.682927
## 189  0.01738     1    ba    95%     30  18.292683
## 190  0.01766     1    ba    95%     31  18.902439
## 191  0.01771     1    ba    95%     32  19.512195
## 192  0.01853     1    ba    95%     33  20.121951
## 193  0.01859     1    ba    95%     34  20.731707
## 194  0.01868     1    ba    95%     35  21.341463
## 195  0.01914     1    ba    95%     36  21.951220
## 196  0.01916     1    ba    95%     37  22.560976
## 197  0.01951     1    ba    95%     38  23.170732
## 198  0.01955     1    ba    95%     39  23.780488
## 199  0.01960     2    ba    95%     41  25.000000
## 200  0.01969     1    ba    95%     42  25.609756
## 201  0.01972     1    ba    95%     43  26.219512
## 202  0.01974     1    ba    95%     44  26.829268
## 203  0.01975     1    ba    95%     45  27.439024
## 204  0.01980     1    ba    95%     46  28.048780
## 205  0.01999     1    ba    95%     47  28.658537
## 206  0.02005     1    ba    95%     48  29.268293
## 207  0.02030     1    ba    95%     49  29.878049
## 208  0.02056     1    ba    95%     50  30.487805
## 209  0.02060     1    ba    95%     51  31.097561
## 210  0.02077     1    ba    95%     52  31.707317
## 211  0.02079     1    ba    95%     53  32.317073
## 212  0.02112     1    ba    95%     54  32.926829
## 213  0.02118     1    ba    95%     55  33.536585
## 214  0.02138     1    ba    95%     56  34.146341
## 215  0.02156     1    ba    95%     57  34.756098
## 216  0.02159     1    ba    95%     58  35.365854
## 217  0.02171     1    ba    95%     59  35.975610
## 218  0.02203     1    ba    95%     60  36.585366
## 219  0.02206     1    ba    95%     61  37.195122
## 220  0.02216     1    ba    95%     62  37.804878
## 221  0.02226     1    ba    95%     63  38.414634
## 222  0.02235     1    ba    95%     64  39.024390
## 223  0.02238     1    ba    95%     65  39.634146
## 224  0.02260     1    ba    95%     66  40.243902
## 225  0.02262     1    ba    95%     67  40.853659
```

```
## 226  0.02264     1    ba    95%     68  41.463415
## 227  0.02284     1    ba    95%     69  42.073171
## 228  0.02288     1    ba    95%     70  42.682927
## 229  0.02292     1    ba    95%     71  43.292683
## 230  0.02310     1    ba    95%     72  43.902439
## 231  0.02319     1    ba    95%     73  44.512195
## 232  0.02325     1    ba    95%     74  45.121951
## 233  0.02329     1    ba    95%     75  45.731707
## 234  0.02348     1    ba    95%     76  46.341463
## 235  0.02363     1    ba    95%     77  46.951220
## 236  0.02367     1    ba    95%     78  47.560976
## 237  0.02380     1    ba    95%     79  48.170732
## 238  0.02406     1    ba    95%     80  48.780488
## 239  0.02450     1    ba    95%     81  49.390244
## 240  0.02454     1    ba    95%     82  50.000000
## 241  0.02504     1    ba    95%     83  50.609756
## 242  0.02532     1    ba    95%     84  51.219512
## 243  0.02589     1    ba    95%     85  51.829268
## 244  0.02593     1    ba    95%     86  52.439024
## 245  0.02603     1    ba    95%     87  53.048780
## 246  0.02614     1    ba    95%     88  53.658537
## 247  0.02626     1    ba    95%     89  54.268293
## 248  0.02649     1    ba    95%     90  54.878049
## 249  0.02659     1    ba    95%     91  55.487805
## 250  0.02663     1    ba    95%     92  56.097561
## 251  0.02679     1    ba    95%     93  56.707317
## 252  0.02712     2    ba    95%     95  57.926829
## 253  0.02721     1    ba    95%     96  58.536585
## 254  0.02725     1    ba    95%     97  59.146341
## 255  0.02733     1    ba    95%     98  59.756098
## 256  0.02746     1    ba    95%     99  60.365854
## 257  0.02755     1    ba    95%    100  60.975610
## 258  0.02766     1    ba    95%    101  61.585366
## 259  0.02767     2    ba    95%    103  62.804878
## 260  0.02768     1    ba    95%    104  63.414634
## 261  0.02783     1    ba    95%    105  64.024390
## 262  0.02801     1    ba    95%    106  64.634146
## 263  0.02808     1    ba    95%    107  65.243902
## 264  0.02809     1    ba    95%    108  65.853659
## 265  0.02836     1    ba    95%    109  66.463415
## 266  0.02837     1    ba    95%    110  67.073171
## 267  0.02844     1    ba    95%    111  67.682927
## 268  0.02870     1    ba    95%    112  68.292683
## 269  0.02877     1    ba    95%    113  68.902439
## 270  0.02887     1    ba    95%    114  69.512195
## 271  0.02916     1    ba    95%    115  70.121951
## 272  0.02938     1    ba    95%    116  70.731707
## 273  0.02939     1    ba    95%    117  71.341463
## 274  0.02946     1    ba    95%    118  71.951220
## 275  0.03038     1    ba    95%    119  72.560976
## 276  0.03077     1    ba    95%    120  73.170732
## 277  0.03081     1    ba    95%    121  73.780488
## 278  0.03115     1    ba    95%    122  74.390244
## 279  0.03121     1    ba    95%    123  75.000000
```

```
## 280  0.03129     1    ba    95%    124  75.609756
## 281  0.03144     1    ba    95%    125  76.219512
## 282  0.03169     1    ba    95%    126  76.829268
## 283  0.03206     1    ba    95%    127  77.439024
## 284  0.03211     1    ba    95%    128  78.048780
## 285  0.03259     1    ba    95%    129  78.658537
## 286  0.03276     1    ba    95%    130  79.268293
## 287  0.03294     1    ba    95%    131  79.878049
## 288  0.03345     1    ba    95%    132  80.487805
## 289  0.03380     1    ba    95%    133  81.097561
## 290  0.03490     1    ba    95%    134  81.707317
## 291  0.03568     1    ba    95%    135  82.317073
## 292  0.03589     1    ba    95%    136  82.926829
## 293  0.03590     2    ba    95%    138  84.146341
## 294  0.03602     1    ba    95%    139  84.756098
## 295  0.03625     1    ba    95%    140  85.365854
## 296  0.03630     1    ba    95%    141  85.975610
## 297  0.03798     1    ba    95%    142  86.585366
## 298  0.03869     1    ba    95%    143  87.195122
## 299  0.04046     1    ba    95%    144  87.804878
## 300  0.04049     1    ba    95%    145  88.414634
## 301  0.04083     1    ba    95%    146  89.024390
## 302  0.04167     1    ba    95%    147  89.634146
## 303  0.04246     1    ba    95%    148  90.243902
## 304  0.04355     1    ba    95%    149  90.853659
## 305  0.04382     2    ba    95%    151  92.073171
## 306  0.04511     1    ba    95%    152  92.682927
## 307  0.04727     1    ba    95%    153  93.292683
## 308  0.04838     1    ba    95%    154  93.902439
## 309  0.05029     1    ba    95%    155  94.512195
## 310  0.05044     1    ba    95%    156  95.121951
## 311  0.05258     1    ba    95%    157  95.731707
## 312  0.05520     1    ba    95%    158  96.341463
## 313  0.05736     1    ba    95%    159  96.951220
## 314  0.06137     1    ba    95%    160  97.560976
## 315  0.06878     1    ba    95%    161  98.170732
## 316  0.07455     1    ba    95%    162  98.780488
## 317  0.08211     1    ba    95%    163  99.390244
## 318  0.10294     1    ba    95%    164 100.000000
```

```
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
## 320  0.01665     1   bcg   100%      2   1.219512
## 321  0.01854     1   bcg   100%      3   1.829268
## 322  0.01910     1   bcg   100%      4   2.439024
## 323  0.02009     1   bcg   100%      5   3.048780
## 324  0.02041     1   bcg   100%      6   3.658537
## 325  0.02058     1   bcg   100%      7   4.268293
## 326  0.02116     1   bcg   100%      8   4.878049
## 327  0.02240     1   bcg   100%      9   5.487805
## 328  0.02298     1   bcg   100%     10   6.097561
## 329  0.02315     1   bcg   100%     11   6.707317
## 330  0.02317     1   bcg   100%     12   7.317073
## 331  0.02387     1   bcg   100%     13   7.926829
## 332  0.02399     1   bcg   100%     14   8.536585
## 333  0.02444     1   bcg   100%     15   9.146341
## 334  0.02449     1   bcg   100%     16   9.756098
## 335  0.02545     1   bcg   100%     17  10.365854
## 336  0.02569     1   bcg   100%     18  10.975610
## 337  0.02592     1   bcg   100%     19  11.585366
## 338  0.02645     1   bcg   100%     20  12.195122
## 339  0.02810     1   bcg   100%     21  12.804878
## 340  0.02861     1   bcg   100%     22  13.414634
## 341  0.02871     1   bcg   100%     23  14.024390
## 342  0.02907     1   bcg   100%     24  14.634146
## 343  0.03012     1   bcg   100%     25  15.243902
## 344  0.03058     1   bcg   100%     26  15.853659
## 345  0.03116     1   bcg   100%     27  16.463415
## 346  0.03118     1   bcg   100%     28  17.073171
## 347  0.03155     1   bcg   100%     29  17.682927
## 348  0.03244     1   bcg   100%     30  18.292683
## 349  0.03288     1   bcg   100%     31  18.902439
## 350  0.03291     1   bcg   100%     32  19.512195
## 351  0.03304     1   bcg   100%     33  20.121951
## 352  0.03326     1   bcg   100%     34  20.731707
## 353  0.03337     1   bcg   100%     35  21.341463
## 354  0.03339     1   bcg   100%     36  21.951220
## 355  0.03385     1   bcg   100%     37  22.560976
## 356  0.03394     1   bcg   100%     38  23.170732
## 357  0.03419     1   bcg   100%     39  23.780488
## 358  0.03425     2   bcg   100%     41  25.000000
## 359  0.03447     1   bcg   100%     42  25.609756
## 360  0.03494     1   bcg   100%     43  26.219512
## 361  0.03495     1   bcg   100%     44  26.829268
## 362  0.03521     1   bcg   100%     45  27.439024
## 363  0.03533     1   bcg   100%     46  28.048780
## 364  0.03570     1   bcg   100%     47  28.658537
## 365  0.03571     1   bcg   100%     48  29.268293
## 366  0.03574     1   bcg   100%     49  29.878049
## 367  0.03579     1   bcg   100%     50  30.487805
## 368  0.03588     1   bcg   100%     51  31.097561
```

```
## 369  0.03625     1   bcg   100%     52  31.707317
## 370  0.03654     1   bcg   100%     53  32.317073
## 371  0.03740     1   bcg   100%     54  32.926829
## 372  0.03783     1   bcg   100%     55  33.536585
## 373  0.03792     1   bcg   100%     56  34.146341
## 374  0.03796     1   bcg   100%     57  34.756098
## 375  0.03823     2   bcg   100%     59  35.975610
## 376  0.03829     1   bcg   100%     60  36.585366
## 377  0.03830     1   bcg   100%     61  37.195122
## 378  0.03861     1   bcg   100%     62  37.804878
## 379  0.03919     1   bcg   100%     63  38.414634
## 380  0.03926     1   bcg   100%     64  39.024390
## 381  0.03961     1   bcg   100%     65  39.634146
## 382  0.03982     1   bcg   100%     66  40.243902
## 383  0.04071     1   bcg   100%     67  40.853659
## 384  0.04084     1   bcg   100%     68  41.463415
## 385  0.04124     1   bcg   100%     69  42.073171
## 386  0.04145     1   bcg   100%     70  42.682927
## 387  0.04151     1   bcg   100%     71  43.292683
## 388  0.04185     1   bcg   100%     72  43.902439
## 389  0.04246     1   bcg   100%     73  44.512195
## 390  0.04264     1   bcg   100%     74  45.121951
## 391  0.04291     1   bcg   100%     75  45.731707
## 392  0.04295     1   bcg   100%     76  46.341463
## 393  0.04345     1   bcg   100%     77  46.951220
## 394  0.04388     1   bcg   100%     78  47.560976
## 395  0.04394     1   bcg   100%     79  48.170732
## 396  0.04403     1   bcg   100%     80  48.780488
## 397  0.04408     1   bcg   100%     81  49.390244
## 398  0.04463     1   bcg   100%     82  50.000000
## 399  0.04473     1   bcg   100%     83  50.609756
## 400  0.04482     2   bcg   100%     85  51.829268
## 401  0.04489     1   bcg   100%     86  52.439024
## 402  0.04494     1   bcg   100%     87  53.048780
## 403  0.04521     1   bcg   100%     88  53.658537
## 404  0.04545     1   bcg   100%     89  54.268293
## 405  0.04562     1   bcg   100%     90  54.878049
## 406  0.04607     1   bcg   100%     91  55.487805
## 407  0.04624     1   bcg   100%     92  56.097561
## 408  0.04688     1   bcg   100%     93  56.707317
## 409  0.04714     1   bcg   100%     94  57.317073
## 410  0.04758     1   bcg   100%     95  57.926829
## 411  0.04798     1   bcg   100%     96  58.536585
## 412  0.04829     1   bcg   100%     97  59.146341
## 413  0.04835     1   bcg   100%     98  59.756098
## 414  0.04852     1   bcg   100%     99  60.365854
## 415  0.04867     1   bcg   100%    100  60.975610
## 416  0.04871     1   bcg   100%    101  61.585366
## 417  0.04889     1   bcg   100%    102  62.195122
## 418  0.04892     1   bcg   100%    103  62.804878
## 419  0.04896     1   bcg   100%    104  63.414634
## 420  0.04996     2   bcg   100%    106  64.634146
## 421  0.05071     1   bcg   100%    107  65.243902
## 422  0.05077     1   bcg   100%    108  65.853659
```

```
## 423  0.05085     1   bcg   100%    109  66.463415
## 424  0.05116     1   bcg   100%    110  67.073171
## 425  0.05148     1   bcg   100%    111  67.682927
## 426  0.05172     1   bcg   100%    112  68.292683
## 427  0.05182     1   bcg   100%    113  68.902439
## 428  0.05208     1   bcg   100%    114  69.512195
## 429  0.05221     1   bcg   100%    115  70.121951
## 430  0.05226     1   bcg   100%    116  70.731707
## 431  0.05250     1   bcg   100%    117  71.341463
## 432  0.05331     1   bcg   100%    118  71.951220
## 433  0.05346     1   bcg   100%    119  72.560976
## 434  0.05405     2   bcg   100%    121  73.780488
## 435  0.05464     1   bcg   100%    122  74.390244
## 436  0.05518     1   bcg   100%    123  75.000000
## 437  0.05534     1   bcg   100%    124  75.609756
## 438  0.05588     1   bcg   100%    125  76.219512
## 439  0.05609     1   bcg   100%    126  76.829268
## 440  0.05710     1   bcg   100%    127  77.439024
## 441  0.05752     1   bcg   100%    128  78.048780
## 442  0.05760     1   bcg   100%    129  78.658537
## 443  0.05820     1   bcg   100%    130  79.268293
## 444  0.05829     1   bcg   100%    131  79.878049
## 445  0.05865     1   bcg   100%    132  80.487805
## 446  0.05881     1   bcg   100%    133  81.097561
## 447  0.05951     1   bcg   100%    134  81.707317
## 448  0.06074     1   bcg   100%    135  82.317073
## 449  0.06138     1   bcg   100%    136  82.926829
## 450  0.06220     1   bcg   100%    137  83.536585
## 451  0.06227     1   bcg   100%    138  84.146341
## 452  0.06239     1   bcg   100%    139  84.756098
## 453  0.06317     1   bcg   100%    140  85.365854
## 454  0.06347     1   bcg   100%    141  85.975610
## 455  0.06377     1   bcg   100%    142  86.585366
## 456  0.06577     1   bcg   100%    143  87.195122
## 457  0.06685     1   bcg   100%    144  87.804878
## 458  0.06724     1   bcg   100%    145  88.414634
## 459  0.06888     1   bcg   100%    146  89.024390
## 460  0.07258     1   bcg   100%    147  89.634146
## 461  0.07324     1   bcg   100%    148  90.243902
## 462  0.09215     1   bcg   100%    149  90.853659
## 463  0.09456     1   bcg   100%    150  91.463415
## 464  0.09727     1   bcg   100%    151  92.073171
## 465  0.09767     1   bcg   100%    152  92.682927
## 466  0.10626     1   bcg   100%    153  93.292683
## 467  0.11041     1   bcg   100%    154  93.902439
## 468  0.11820     1   bcg   100%    155  94.512195
## 469  0.13167     1   bcg   100%    156  95.121951
## 470  0.13231     1   bcg   100%    157  95.731707
## 471  0.13375     1   bcg   100%    158  96.341463
## 472  0.13451     1   bcg   100%    159  96.951220
## 473  0.13605     1   bcg   100%    160  97.560976
## 474  0.14008     1   bcg   100%    161  98.170732
## 475  0.14128     1   bcg   100%    162  98.780488
## 476  0.14351     1   bcg   100%    163  99.390244
```

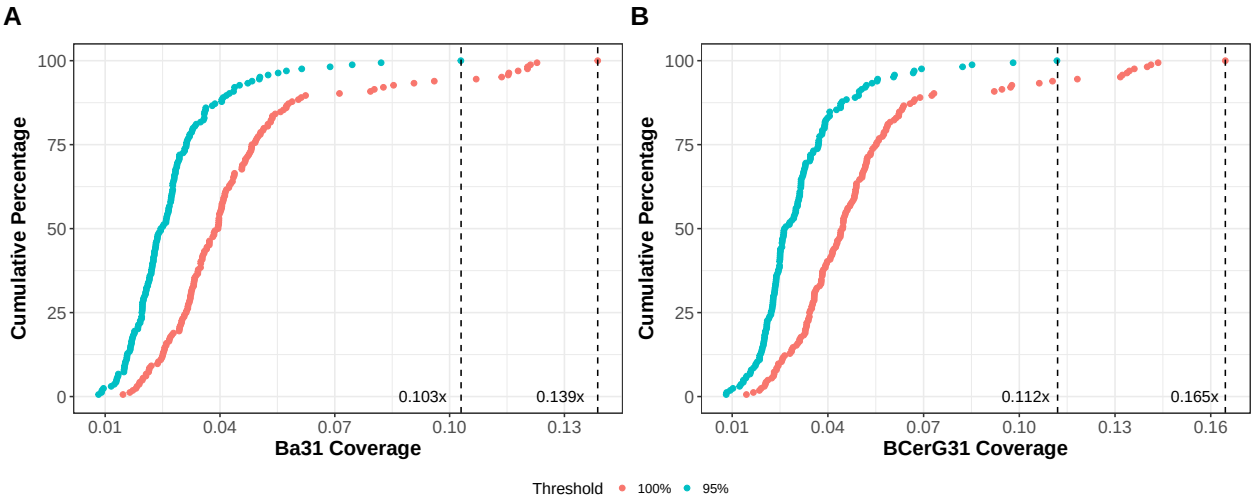
```
## 477  0.16458     1   bcg   100%    164 100.000000
## 479  0.00827     1   bcg    95%      2   1.219512
## 480  0.00938     1   bcg    95%      3   1.829268
## 481  0.01024     1   bcg    95%      4   2.439024
## 482  0.01233     1   bcg    95%      5   3.048780
## 483  0.01276     1   bcg    95%      6   3.658537
## 484  0.01349     1   bcg    95%      7   4.268293
## 485  0.01352     1   bcg    95%      8   4.878049
## 486  0.01427     1   bcg    95%      9   5.487805
## 487  0.01455     1   bcg    95%     10   6.097561
## 488  0.01560     1   bcg    95%     11   6.707317
## 489  0.01579     1   bcg    95%     12   7.317073
## 490  0.01596     1   bcg    95%     13   7.926829
## 491  0.01670     1   bcg    95%     14   8.536585
## 492  0.01726     1   bcg    95%     15   9.146341
## 493  0.01745     1   bcg    95%     16   9.756098
## 494  0.01788     1   bcg    95%     17  10.365854
## 495  0.01860     1   bcg    95%     18  10.975610
## 496  0.01863     1   bcg    95%     19  11.585366
## 497  0.01865     1   bcg    95%     20  12.195122
## 498  0.01899     1   bcg    95%     21  12.804878
## 499  0.01920     1   bcg    95%     22  13.414634
## 500  0.01938     1   bcg    95%     23  14.024390
## 501  0.01949     1   bcg    95%     24  14.634146
## 502  0.01977     1   bcg    95%     25  15.243902
## 503  0.01984     1   bcg    95%     26  15.853659
## 504  0.01998     1   bcg    95%     27  16.463415
## 505  0.02007     1   bcg    95%     28  17.073171
## 506  0.02019     1   bcg    95%     29  17.682927
## 507  0.02020     1   bcg    95%     30  18.292683
## 508  0.02045     1   bcg    95%     31  18.902439
## 509  0.02047     1   bcg    95%     32  19.512195
## 510  0.02076     1   bcg    95%     33  20.121951
## 511  0.02085     1   bcg    95%     34  20.731707
## 512  0.02086     1   bcg    95%     35  21.341463
## 513  0.02095     1   bcg    95%     36  21.951220
## 514  0.02099     1   bcg    95%     37  22.560976
## 515  0.02147     1   bcg    95%     38  23.170732
## 516  0.02171     1   bcg    95%     39  23.780488
## 517  0.02214     1   bcg    95%     40  24.390244
## 518  0.02220     1   bcg    95%     41  25.000000
## 519  0.02240     1   bcg    95%     42  25.609756
## 520  0.02254     1   bcg    95%     43  26.219512
## 521  0.02261     1   bcg    95%     44  26.829268
## 522  0.02275     1   bcg    95%     45  27.439024
## 523  0.02279     1   bcg    95%     46  28.048780
## 524  0.02283     1   bcg    95%     47  28.658537
## 525  0.02289     1   bcg    95%     48  29.268293
## 526  0.02292     1   bcg    95%     49  29.878049
## 527  0.02321     1   bcg    95%     50  30.487805
## 528  0.02330     1   bcg    95%     51  31.097561
## 529  0.02338     1   bcg    95%     52  31.707317
## 530  0.02343     1   bcg    95%     53  32.317073
## 531  0.02354     1   bcg    95%     54  32.926829
```

```
## 532  0.02364     1   bcg    95%     55  33.536585
## 533  0.02373     1   bcg    95%     56  34.146341
## 534  0.02378     1   bcg    95%     57  34.756098
## 535  0.02385     1   bcg    95%     58  35.365854
## 536  0.02387     1   bcg    95%     59  35.975610
## 537  0.02417     1   bcg    95%     60  36.585366
## 538  0.02447     1   bcg    95%     61  37.195122
## 539  0.02456     1   bcg    95%     62  37.804878
## 540  0.02482     1   bcg    95%     63  38.414634
## 541  0.02486     1   bcg    95%     64  39.024390
## 542  0.02488     2   bcg    95%     66  40.243902
## 543  0.02498     1   bcg    95%     67  40.853659
## 544  0.02500     1   bcg    95%     68  41.463415
## 545  0.02503     1   bcg    95%     69  42.073171
## 546  0.02504     1   bcg    95%     70  42.682927
## 547  0.02514     1   bcg    95%     71  43.292683
## 548  0.02515     1   bcg    95%     72  43.902439
## 549  0.02560     1   bcg    95%     73  44.512195
## 550  0.02562     2   bcg    95%     75  45.731707
## 551  0.02573     1   bcg    95%     76  46.341463
## 552  0.02576     1   bcg    95%     77  46.951220
## 553  0.02603     1   bcg    95%     78  47.560976
## 554  0.02612     1   bcg    95%     79  48.170732
## 555  0.02619     1   bcg    95%     80  48.780488
## 556  0.02621     1   bcg    95%     81  49.390244
## 557  0.02638     1   bcg    95%     82  50.000000
## 558  0.02686     1   bcg    95%     83  50.609756
## 559  0.02794     1   bcg    95%     84  51.219512
## 560  0.02803     1   bcg    95%     85  51.829268
## 561  0.02869     1   bcg    95%     86  52.439024
## 562  0.02929     1   bcg    95%     87  53.048780
## 563  0.02940     1   bcg    95%     88  53.658537
## 564  0.02965     1   bcg    95%     89  54.268293
## 565  0.02992     1   bcg    95%     90  54.878049
## 566  0.02996     1   bcg    95%     91  55.487805
## 567  0.03001     1   bcg    95%     92  56.097561
## 568  0.03048     1   bcg    95%     93  56.707317
## 569  0.03053     1   bcg    95%     94  57.317073
## 570  0.03060     1   bcg    95%     95  57.926829
## 571  0.03086     1   bcg    95%     96  58.536585
## 572  0.03094     1   bcg    95%     97  59.146341
## 573  0.03117     1   bcg    95%     98  59.756098
## 574  0.03119     1   bcg    95%     99  60.365854
## 575  0.03123     1   bcg    95%    100  60.975610
## 576  0.03151     2   bcg    95%    102  62.195122
## 577  0.03156     1   bcg    95%    103  62.804878
## 578  0.03158     1   bcg    95%    104  63.414634
## 579  0.03160     1   bcg    95%    105  64.024390
## 580  0.03161     1   bcg    95%    106  64.634146
## 581  0.03193     1   bcg    95%    107  65.243902
## 582  0.03218     1   bcg    95%    108  65.853659
## 583  0.03231     1   bcg    95%    109  66.463415
## 584  0.03247     1   bcg    95%    110  67.073171
## 585  0.03263     1   bcg    95%    111  67.682927
```

```
## 586  0.03286     1   bcg    95%    112  68.292683
## 587  0.03295     1   bcg    95%    113  68.902439
## 588  0.03307     1   bcg    95%    114  69.512195
## 589  0.03430     1   bcg    95%    115  70.121951
## 590  0.03454     1   bcg    95%    116  70.731707
## 591  0.03455     1   bcg    95%    117  71.341463
## 592  0.03458     1   bcg    95%    118  71.951220
## 593  0.03533     1   bcg    95%    119  72.560976
## 594  0.03549     1   bcg    95%    120  73.170732
## 595  0.03669     1   bcg    95%    121  73.780488
## 596  0.03692     1   bcg    95%    122  74.390244
## 597  0.03705     1   bcg    95%    123  75.000000
## 598  0.03711     1   bcg    95%    124  75.609756
## 599  0.03717     1   bcg    95%    125  76.219512
## 600  0.03741     2   bcg    95%    127  77.439024
## 601  0.03813     1   bcg    95%    128  78.048780
## 602  0.03823     1   bcg    95%    129  78.658537
## 603  0.03827     1   bcg    95%    130  79.268293
## 604  0.03891     1   bcg    95%    131  79.878049
## 605  0.03894     1   bcg    95%    132  80.487805
## 606  0.03908     1   bcg    95%    133  81.097561
## 607  0.03911     1   bcg    95%    134  81.707317
## 608  0.03936     1   bcg    95%    135  82.317073
## 609  0.03988     1   bcg    95%    136  82.926829
## 610  0.04054     1   bcg    95%    137  83.536585
## 611  0.04058     2   bcg    95%    139  84.756098
## 612  0.04269     1   bcg    95%    140  85.365854
## 613  0.04398     1   bcg    95%    141  85.975610
## 614  0.04402     1   bcg    95%    142  86.585366
## 615  0.04403     1   bcg    95%    143  87.195122
## 616  0.04448     1   bcg    95%    144  87.804878
## 617  0.04581     1   bcg    95%    145  88.414634
## 618  0.04852     1   bcg    95%    146  89.024390
## 619  0.04962     1   bcg    95%    147  89.634146
## 620  0.04982     2   bcg    95%    149  90.853659
## 621  0.05057     1   bcg    95%    150  91.463415
## 622  0.05169     1   bcg    95%    151  92.073171
## 623  0.05209     1   bcg    95%    152  92.682927
## 624  0.05384     1   bcg    95%    153  93.292683
## 625  0.05544     1   bcg    95%    154  93.902439
## 626  0.05560     1   bcg    95%    155  94.512195
## 627  0.06060     1   bcg    95%    156  95.121951
## 628  0.06089     1   bcg    95%    157  95.731707
## 629  0.06674     1   bcg    95%    158  96.341463
## 630  0.06706     1   bcg    95%    159  96.951220
## 631  0.06936     1   bcg    95%    160  97.560976
## 632  0.08222     1   bcg    95%    161  98.170732
## 633  0.08519     1   bcg    95%    162  98.780488
## 634  0.09806     1   bcg    95%    163  99.390244
## 635  0.11180     1   bcg    95%    164 100.000000
```

```
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-02-lef31-lod.pdf'), height=6, width=12, onefile=FALSE)
grid_arrange_shared_lef31(p1, p2)
dev_null <- dev.off()

png(paste0(FIGURE, '/figure-02-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
## 637  0.07806     1   lef   100%      2   1.219512
## 638  0.07807     1   lef   100%      3   1.829268
## 639  0.07815     1   lef   100%      4   2.439024
## 640  0.07840     1   lef   100%      5   3.048780
## 641  0.07992     1   lef   100%      6   3.658537
## 642  0.08024     1   lef   100%      7   4.268293
## 643  0.08067     1   lef   100%      8   4.878049
## 644  0.08278     1   lef   100%      9   5.487805
```

```
## 645  0.08337     1   lef   100%     10   6.097561
## 646  0.08356     1   lef   100%     11   6.707317
## 647  0.08378     1   lef   100%     12   7.317073
## 648  0.08465     2   lef   100%     14   8.536585
## 649  0.08499     1   lef   100%     15   9.146341
## 650  0.08504     1   lef   100%     16   9.756098
## 651  0.08570     1   lef   100%     17  10.365854
## 652  0.08730     1   lef   100%     18  10.975610
## 653  0.08780     1   lef   100%     19  11.585366
## 654  0.08825     1   lef   100%     20  12.195122
## 655  0.08899     1   lef   100%     21  12.804878
## 656  0.08968     1   lef   100%     22  13.414634
## 657  0.09023     1   lef   100%     23  14.024390
## 658  0.09100     1   lef   100%     24  14.634146
## 659  0.09155     1   lef   100%     25  15.243902
## 660  0.09316     1   lef   100%     26  15.853659
## 661  0.09464     1   lef   100%     27  16.463415
## 662  0.09614     2   lef   100%     29  17.682927
## 663  0.09687     1   lef   100%     30  18.292683
## 664  0.09713     1   lef   100%     31  18.902439
## 665  0.09758     1   lef   100%     32  19.512195
## 666  0.09848     1   lef   100%     33  20.121951
## 667  0.09903     1   lef   100%     34  20.731707
## 668  0.10039     1   lef   100%     35  21.341463
## 669  0.10162     1   lef   100%     36  21.951220
## 670  0.10186     1   lef   100%     37  22.560976
## 671  0.10252     1   lef   100%     38  23.170732
## 672  0.10387     1   lef   100%     39  23.780488
## 673  0.10392     1   lef   100%     40  24.390244
## 674  0.10460     1   lef   100%     41  25.000000
## 675  0.10577     1   lef   100%     42  25.609756
## 676  0.10639     1   lef   100%     43  26.219512
## 677  0.10676     1   lef   100%     44  26.829268
## 678  0.10733     1   lef   100%     45  27.439024
## 679  0.10789     1   lef   100%     46  28.048780
## 680  0.10891     1   lef   100%     47  28.658537
## 681  0.10908     1   lef   100%     48  29.268293
## 682  0.10941     1   lef   100%     49  29.878049
## 683  0.11007     1   lef   100%     50  30.487805
## 684  0.11021     1   lef   100%     51  31.097561
## 685  0.11034     1   lef   100%     52  31.707317
## 686  0.11130     1   lef   100%     53  32.317073
## 687  0.11156     1   lef   100%     54  32.926829
## 688  0.11234     1   lef   100%     55  33.536585
## 689  0.11250     1   lef   100%     56  34.146341
## 690  0.11260     1   lef   100%     57  34.756098
## 691  0.11263     1   lef   100%     58  35.365854
## 692  0.11385     1   lef   100%     59  35.975610
## 693  0.11462     1   lef   100%     60  36.585366
## 694  0.11504     1   lef   100%     61  37.195122
## 695  0.11508     1   lef   100%     62  37.804878
## 696  0.11521     1   lef   100%     63  38.414634
## 697  0.11542     1   lef   100%     64  39.024390
## 698  0.11545     1   lef   100%     65  39.634146
```

```
## 699  0.11547     1   lef   100%     66  40.243902
## 700  0.11603     1   lef   100%     67  40.853659
## 701  0.11691     1   lef   100%     68  41.463415
## 702  0.11697     1   lef   100%     69  42.073171
## 703  0.11704     1   lef   100%     70  42.682927
## 704  0.11770     1   lef   100%     71  43.292683
## 705  0.11820     1   lef   100%     72  43.902439
## 706  0.11923     1   lef   100%     73  44.512195
## 707  0.11988     1   lef   100%     74  45.121951
## 708  0.12016     1   lef   100%     75  45.731707
## 709  0.12067     1   lef   100%     76  46.341463
## 710  0.12105     1   lef   100%     77  46.951220
## 711  0.12117     1   lef   100%     78  47.560976
## 712  0.12168     1   lef   100%     79  48.170732
## 713  0.12185     1   lef   100%     80  48.780488
## 714  0.12249     1   lef   100%     81  49.390244
## 715  0.12259     1   lef   100%     82  50.000000
## 716  0.12263     1   lef   100%     83  50.609756
## 717  0.12281     1   lef   100%     84  51.219512
## 718  0.12387     1   lef   100%     85  51.829268
## 719  0.12428     1   lef   100%     86  52.439024
## 720  0.12476     1   lef   100%     87  53.048780
## 721  0.12484     1   lef   100%     88  53.658537
## 722  0.12497     1   lef   100%     89  54.268293
## 723  0.12569     1   lef   100%     90  54.878049
## 724  0.12623     1   lef   100%     91  55.487805
## 725  0.12743     1   lef   100%     92  56.097561
## 726  0.12757     1   lef   100%     93  56.707317
## 727  0.12794     1   lef   100%     94  57.317073
## 728  0.12841     1   lef   100%     95  57.926829
## 729  0.12903     1   lef   100%     96  58.536585
## 730  0.12931     1   lef   100%     97  59.146341
## 731  0.12932     1   lef   100%     98  59.756098
## 732  0.12984     1   lef   100%     99  60.365854
## 733  0.13029     1   lef   100%    100  60.975610
## 734  0.13100     1   lef   100%    101  61.585366
## 735  0.13106     1   lef   100%    102  62.195122
## 736  0.13131     1   lef   100%    103  62.804878
## 737  0.13235     1   lef   100%    104  63.414634
## 738  0.13327     1   lef   100%    105  64.024390
## 739  0.13418     1   lef   100%    106  64.634146
## 740  0.13483     1   lef   100%    107  65.243902
## 741  0.13502     1   lef   100%    108  65.853659
## 742  0.13509     1   lef   100%    109  66.463415
## 743  0.13534     1   lef   100%    110  67.073171
## 744  0.13610     1   lef   100%    111  67.682927
## 745  0.13706     1   lef   100%    112  68.292683
## 746  0.13789     1   lef   100%    113  68.902439
## 747  0.13804     1   lef   100%    114  69.512195
## 748  0.13946     1   lef   100%    115  70.121951
## 749  0.14160     1   lef   100%    116  70.731707
## 750  0.14217     1   lef   100%    117  71.341463
## 751  0.14248     1   lef   100%    118  71.951220
## 752  0.14270     1   lef   100%    119  72.560976
```

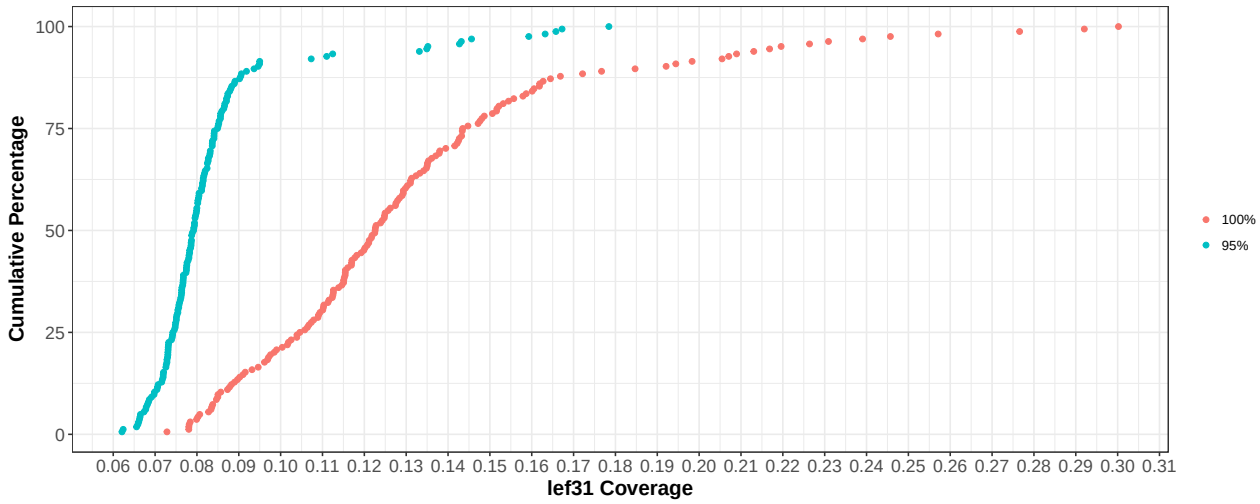
```
## 753  0.14325     1   lef   100%    120  73.170732
## 754  0.14340     2   lef   100%    122  74.390244
## 755  0.14350     1   lef   100%    123  75.000000
## 756  0.14475     1   lef   100%    124  75.609756
## 757  0.14719     1   lef   100%    125  76.219512
## 758  0.14764     1   lef   100%    126  76.829268
## 759  0.14809     1   lef   100%    127  77.439024
## 760  0.14866     1   lef   100%    128  78.048780
## 761  0.15060     1   lef   100%    129  78.658537
## 762  0.15158     1   lef   100%    130  79.268293
## 763  0.15171     1   lef   100%    131  79.878049
## 764  0.15217     1   lef   100%    132  80.487805
## 765  0.15319     1   lef   100%    133  81.097561
## 766  0.15446     1   lef   100%    134  81.707317
## 767  0.15571     1   lef   100%    135  82.317073
## 768  0.15790     1   lef   100%    136  82.926829
## 769  0.15867     1   lef   100%    137  83.536585
## 770  0.16011     1   lef   100%    138  84.146341
## 771  0.16055     1   lef   100%    139  84.756098
## 772  0.16183     1   lef   100%    140  85.365854
## 773  0.16191     1   lef   100%    141  85.975610
## 774  0.16273     1   lef   100%    142  86.585366
## 775  0.16448     1   lef   100%    143  87.195122
## 776  0.16685     1   lef   100%    144  87.804878
## 777  0.17215     1   lef   100%    145  88.414634
## 778  0.17669     1   lef   100%    146  89.024390
## 779  0.18470     1   lef   100%    147  89.634146
## 780  0.19211     1   lef   100%    148  90.243902
## 781  0.19443     1   lef   100%    149  90.853659
## 782  0.19832     1   lef   100%    150  91.463415
## 783  0.20548     1   lef   100%    151  92.073171
## 784  0.20708     1   lef   100%    152  92.682927
## 785  0.20895     1   lef   100%    153  93.292683
## 786  0.21306     1   lef   100%    154  93.902439
## 787  0.21684     1   lef   100%    155  94.512195
## 788  0.21959     1   lef   100%    156  95.121951
## 789  0.22638     1   lef   100%    157  95.731707
## 790  0.23091     1   lef   100%    158  96.341463
## 791  0.23906     1   lef   100%    159  96.951220
## 792  0.24570     1   lef   100%    160  97.560976
## 793  0.25712     1   lef   100%    161  98.170732
## 794  0.27658     1   lef   100%    162  98.780488
## 795  0.29204     1   lef   100%    163  99.390244
## 796  0.30020     1   lef   100%    164 100.000000
## 798  0.06238     1   lef    95%      2   1.219512
## 799  0.06558     1   lef    95%      3   1.829268
## 800  0.06590     1   lef    95%      4   2.439024
## 801  0.06609     1   lef    95%      5   3.048780
## 802  0.06626     1   lef    95%      6   3.658537
## 803  0.06641     1   lef    95%      7   4.268293
## 804  0.06650     1   lef    95%      8   4.878049
## 805  0.06734     1   lef    95%      9   5.487805
## 806  0.06776     1   lef    95%     10   6.097561
## 807  0.06789     1   lef    95%     11   6.707317
```

```
## 808  0.06817     1   lef    95%     12   7.317073
## 809  0.06842     1   lef    95%     13   7.926829
## 810  0.06863     1   lef    95%     14   8.536585
## 811  0.06916     1   lef    95%     15   9.146341
## 812  0.06975     1   lef    95%     16   9.756098
## 813  0.06987     1   lef    95%     17  10.365854
## 814  0.07045     1   lef    95%     18  10.975610
## 815  0.07060     1   lef    95%     19  11.585366
## 816  0.07074     1   lef    95%     20  12.195122
## 817  0.07158     1   lef    95%     21  12.804878
## 818  0.07173     1   lef    95%     22  13.414634
## 819  0.07193     1   lef    95%     23  14.024390
## 820  0.07195     1   lef    95%     24  14.634146
## 821  0.07204     1   lef    95%     25  15.243902
## 822  0.07259     2   lef    95%     27  16.463415
## 823  0.07267     1   lef    95%     28  17.073171
## 824  0.07284     1   lef    95%     29  17.682927
## 825  0.07285     1   lef    95%     30  18.292683
## 826  0.07299     1   lef    95%     31  18.902439
## 827  0.07306     1   lef    95%     32  19.512195
## 828  0.07307     1   lef    95%     33  20.121951
## 829  0.07312     1   lef    95%     34  20.731707
## 830  0.07314     1   lef    95%     35  21.341463
## 831  0.07319     1   lef    95%     36  21.951220
## 832  0.07327     1   lef    95%     37  22.560976
## 833  0.07391     1   lef    95%     38  23.170732
## 834  0.07409     1   lef    95%     39  23.780488
## 835  0.07419     1   lef    95%     40  24.390244
## 836  0.07425     1   lef    95%     41  25.000000
## 837  0.07460     1   lef    95%     42  25.609756
## 838  0.07480     1   lef    95%     43  26.219512
## 839  0.07484     1   lef    95%     44  26.829268
## 840  0.07495     1   lef    95%     45  27.439024
## 841  0.07508     1   lef    95%     46  28.048780
## 842  0.07509     1   lef    95%     47  28.658537
## 843  0.07523     1   lef    95%     48  29.268293
## 844  0.07545     1   lef    95%     49  29.878049
## 845  0.07546     1   lef    95%     50  30.487805
## 846  0.07565     1   lef    95%     51  31.097561
## 847  0.07571     1   lef    95%     52  31.707317
## 848  0.07586     1   lef    95%     53  32.317073
## 849  0.07603     1   lef    95%     54  32.926829
## 850  0.07618     1   lef    95%     55  33.536585
## 851  0.07627     1   lef    95%     56  34.146341
## 852  0.07635     1   lef    95%     57  34.756098
## 853  0.07636     1   lef    95%     58  35.365854
## 854  0.07642     1   lef    95%     59  35.975610
## 855  0.07661     1   lef    95%     60  36.585366
## 856  0.07668     1   lef    95%     61  37.195122
## 857  0.07671     1   lef    95%     62  37.804878
## 858  0.07676     1   lef    95%     63  38.414634
## 859  0.07677     1   lef    95%     64  39.024390
## 860  0.07738     1   lef    95%     65  39.634146
## 861  0.07749     1   lef    95%     66  40.243902
```

```
## 862  0.07750     1   lef    95%     67  40.853659
## 863  0.07759     1   lef    95%     68  41.463415
## 864  0.07764     1   lef    95%     69  42.073171
## 865  0.07791     1   lef    95%     70  42.682927
## 866  0.07809     1   lef    95%     71  43.292683
## 867  0.07812     1   lef    95%     72  43.902439
## 868  0.07826     1   lef    95%     73  44.512195
## 869  0.07827     1   lef    95%     74  45.121951
## 870  0.07850     1   lef    95%     75  45.731707
## 871  0.07860     1   lef    95%     76  46.341463
## 872  0.07862     1   lef    95%     77  46.951220
## 873  0.07870     1   lef    95%     78  47.560976
## 874  0.07874     2   lef    95%     80  48.780488
## 875  0.07889     1   lef    95%     81  49.390244
## 876  0.07920     1   lef    95%     82  50.000000
## 877  0.07926     1   lef    95%     83  50.609756
## 878  0.07940     1   lef    95%     84  51.219512
## 879  0.07942     1   lef    95%     85  51.829268
## 880  0.07955     2   lef    95%     87  53.048780
## 881  0.07968     1   lef    95%     88  53.658537
## 882  0.07984     1   lef    95%     89  54.268293
## 883  0.07993     1   lef    95%     90  54.878049
## 884  0.07999     1   lef    95%     91  55.487805
## 885  0.08020     2   lef    95%     93  56.707317
## 886  0.08024     1   lef    95%     94  57.317073
## 887  0.08039     1   lef    95%     95  57.926829
## 888  0.08049     1   lef    95%     96  58.536585
## 889  0.08051     1   lef    95%     97  59.146341
## 890  0.08111     1   lef    95%     98  59.756098
## 891  0.08114     1   lef    95%     99  60.365854
## 892  0.08131     1   lef    95%    100  60.975610
## 893  0.08134     1   lef    95%    101  61.585366
## 894  0.08150     1   lef    95%    102  62.195122
## 895  0.08151     1   lef    95%    103  62.804878
## 896  0.08166     1   lef    95%    104  63.414634
## 897  0.08183     1   lef    95%    105  64.024390
## 898  0.08199     1   lef    95%    106  64.634146
## 899  0.08243     1   lef    95%    107  65.243902
## 900  0.08249     2   lef    95%    109  66.463415
## 901  0.08269     1   lef    95%    110  67.073171
## 902  0.08272     1   lef    95%    111  67.682927
## 903  0.08301     1   lef    95%    112  68.292683
## 904  0.08314     1   lef    95%    113  68.902439
## 905  0.08319     1   lef    95%    114  69.512195
## 906  0.08366     2   lef    95%    116  70.731707
## 907  0.08367     1   lef    95%    117  71.341463
## 908  0.08372     1   lef    95%    118  71.951220
## 909  0.08407     1   lef    95%    119  72.560976
## 910  0.08409     1   lef    95%    120  73.170732
## 911  0.08411     1   lef    95%    121  73.780488
## 912  0.08417     1   lef    95%    122  74.390244
## 913  0.08491     1   lef    95%    123  75.000000
## 914  0.08507     1   lef    95%    124  75.609756
## 915  0.08522     1   lef    95%    125  76.219512
```

```
## 916  0.08530     1   lef    95%    126  76.829268
## 917  0.08561     1   lef    95%    127  77.439024
## 918  0.08564     1   lef    95%    128  78.048780
## 919  0.08570     1   lef    95%    129  78.658537
## 920  0.08598     1   lef    95%    130  79.268293
## 921  0.08642     1   lef    95%    131  79.878049
## 922  0.08658     1   lef    95%    132  80.487805
## 923  0.08669     1   lef    95%    133  81.097561
## 924  0.08706     1   lef    95%    134  81.707317
## 925  0.08707     1   lef    95%    135  82.317073
## 926  0.08734     1   lef    95%    136  82.926829
## 927  0.08740     1   lef    95%    137  83.536585
## 928  0.08784     1   lef    95%    138  84.146341
## 929  0.08810     1   lef    95%    139  84.756098
## 930  0.08834     1   lef    95%    140  85.365854
## 931  0.08893     1   lef    95%    141  85.975610
## 932  0.08910     1   lef    95%    142  86.585366
## 933  0.09019     1   lef    95%    143  87.195122
## 934  0.09047     1   lef    95%    144  87.804878
## 935  0.09063     1   lef    95%    145  88.414634
## 936  0.09184     1   lef    95%    146  89.024390
## 937  0.09365     1   lef    95%    147  89.634146
## 938  0.09464     1   lef    95%    148  90.243902
## 939  0.09497     1   lef    95%    149  90.853659
## 940  0.09500     1   lef    95%    150  91.463415
## 941  0.10731     1   lef    95%    151  92.073171
## 942  0.11100     1   lef    95%    152  92.682927
## 943  0.11244     1   lef    95%    153  93.292683
## 944  0.13315     1   lef    95%    154  93.902439
## 945  0.13493     1   lef    95%    155  94.512195
## 946  0.13519     1   lef    95%    156  95.121951
## 947  0.14273     1   lef    95%    157  95.731707
## 948  0.14318     1   lef    95%    158  96.341463
## 949  0.14562     1   lef    95%    159  96.951220
## 950  0.15930     1   lef    95%    160  97.560976
## 951  0.16321     1   lef    95%    161  98.170732
## 952  0.16578     1   lef    95%    162  98.780488
## 953  0.16725     1   lef    95%    163  99.390244
## 954  0.17844     1   lef    95%    164 100.000000
```

```
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

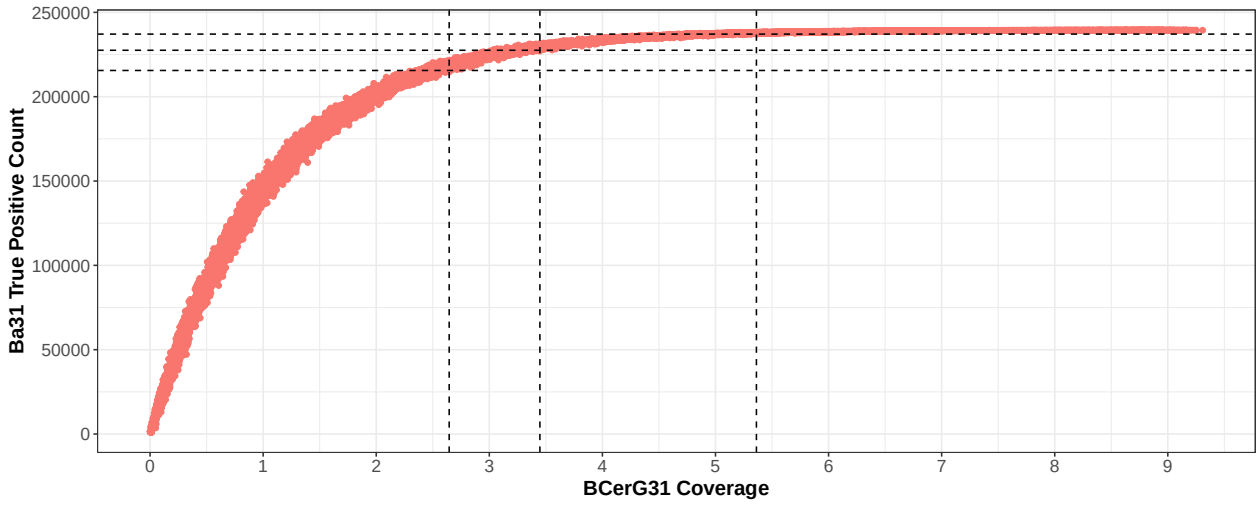
Lethal Factor

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

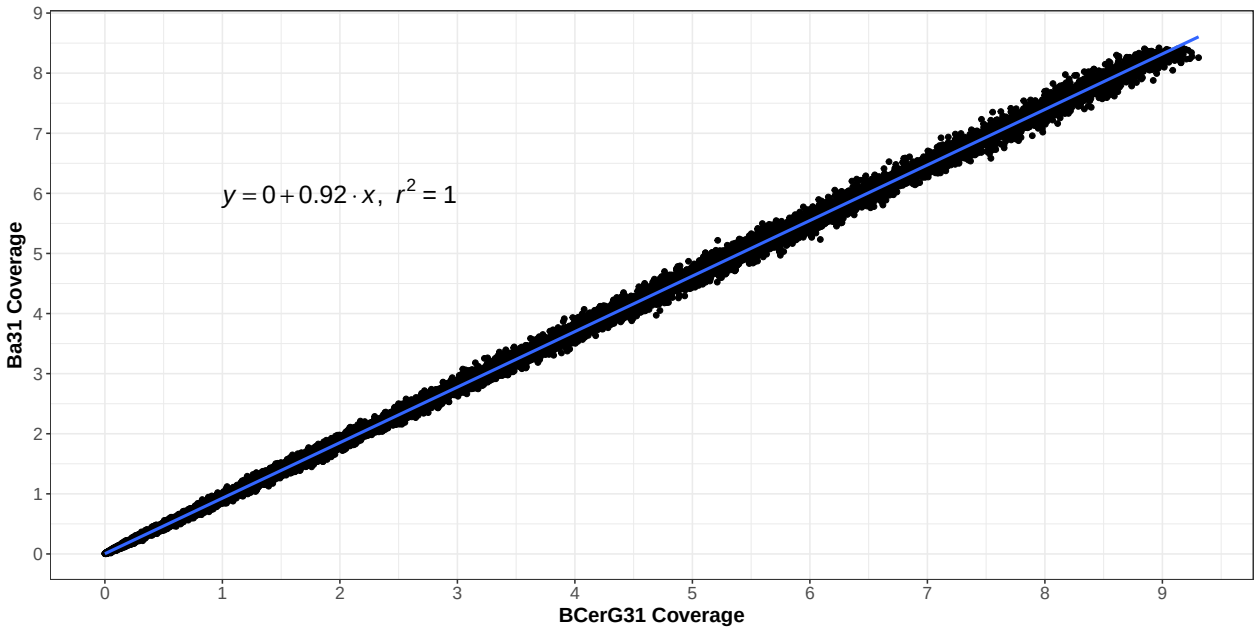
```
ba_bcg_coverages_lm = lm(kmer_cov_mean ~ 0 + bcg_cov_mean, temp_df)
summary(ba_bcg_coverages_lm)
```

```
##
## 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-02-ba31-bcerg31-anthracis', 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)
```

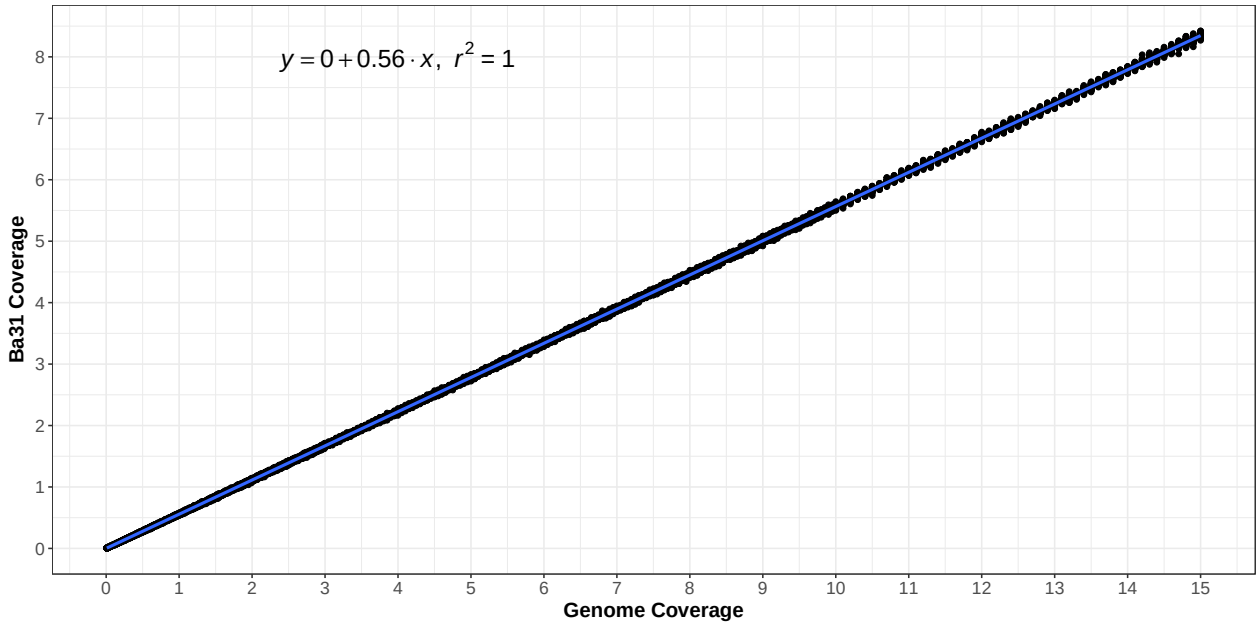
```
##
## Call:
## lm(formula = kmer_cov_mean ~ 0 + simulated_coverage, data = temp_df)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.131475 -0.010506  0.000081  0.010775  0.132931
##
## Coefficients:
##                      Estimate Std. Error t value            Pr(>|t|)
## 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)
```

Ba31 Coverage and BCerG Coverage (non-anthracis)

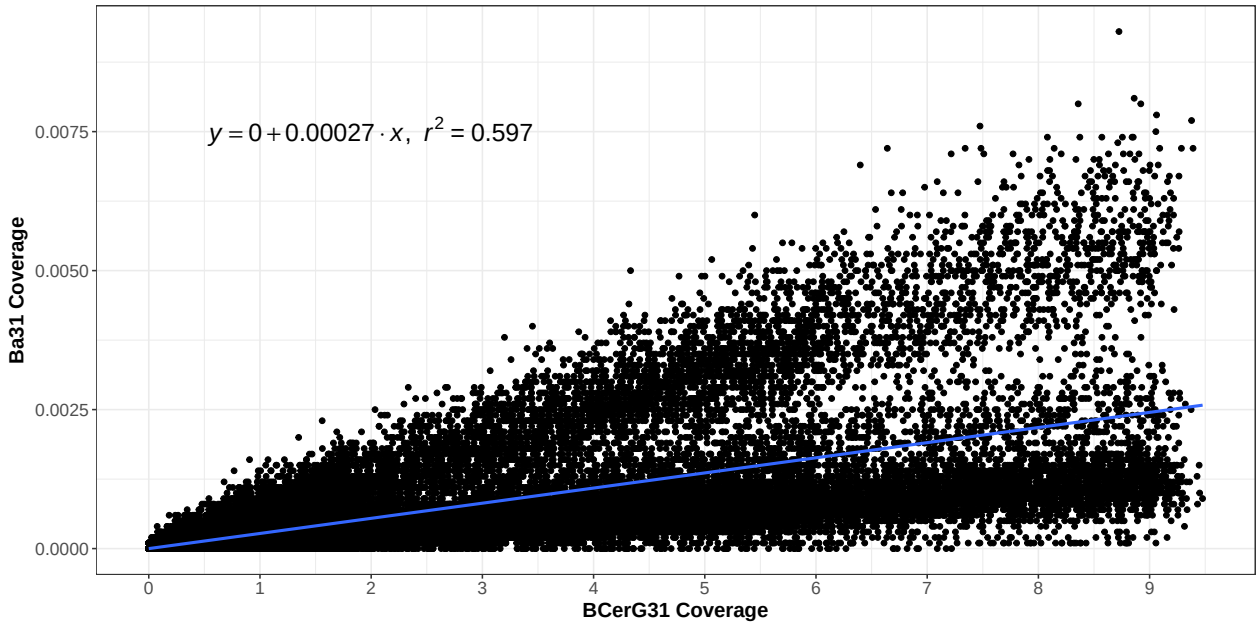
```
temp_df <- simulation_ba[simulation_ba$is_ba == 'False',]
ba_bcg_coverages_lm_non_anthracis = lm(kmer_cov_mean ~ 0 + bcg_cov_mean, temp_df)
summary(ba_bcg_coverages_lm_non_anthracis)
```

```
##
## Call:
## lm(formula = kmer_cov_mean ~ 0 + bcg_cov_mean, data = temp_df)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.002365  0.000000  0.000000  0.000000  0.006925
##
## Coefficients:
##                  Estimate   Std. Error t value            Pr(>|t|)
## bcg_cov_mean 0.0002722235 0.0000006653   409.2 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0004821 on 112870 degrees of freedom
## Multiple R-squared:  0.5973, Adjusted R-squared:  0.5973
## F-statistic: 1.674e+05 on 1 and 112870 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 = 371.5, df = 112870, p-value < 0.00000000000000022
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.7390547 0.7443041
## sample estimates:
##       cor
## 0.7416908
```

```
g <- ggplot(temp_df, aes(bcg_cov_mean, kmer_cov_mean)) +
    geom_point() +
    annotate("text", x = 2, y = 0.0075, label = lm_eqn(ba_bcg_coverages_lm_non_anthracis), parse = TRUE
    geom_smooth(method='lm')  +
    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)
```

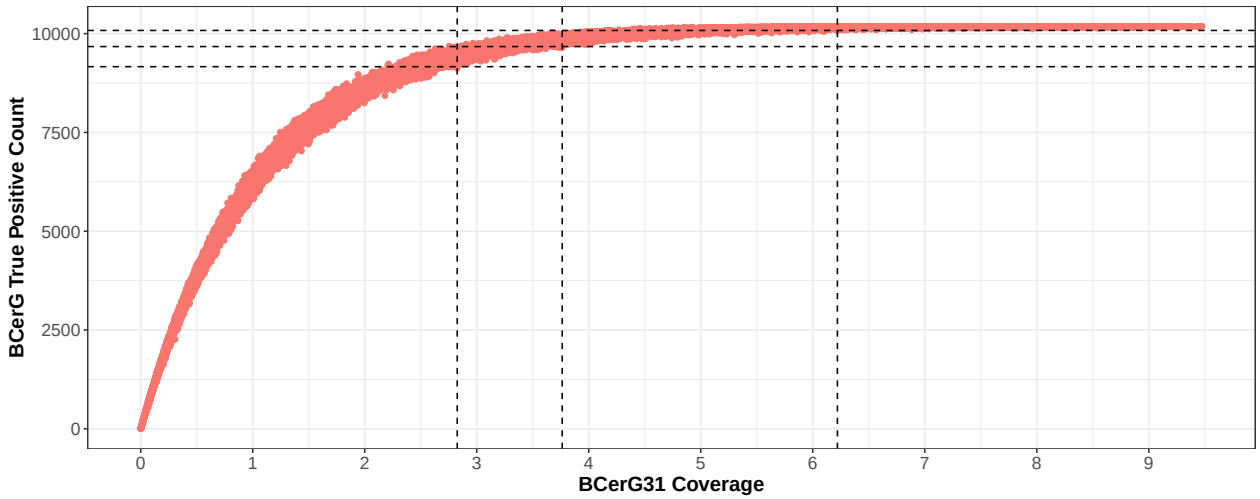



```
write_plot(p, 'supplementary-figure-03-ba31-bcerg31-nonanthracis', height=6, width=12)
```

```
remove(temp_df)
```

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)
```

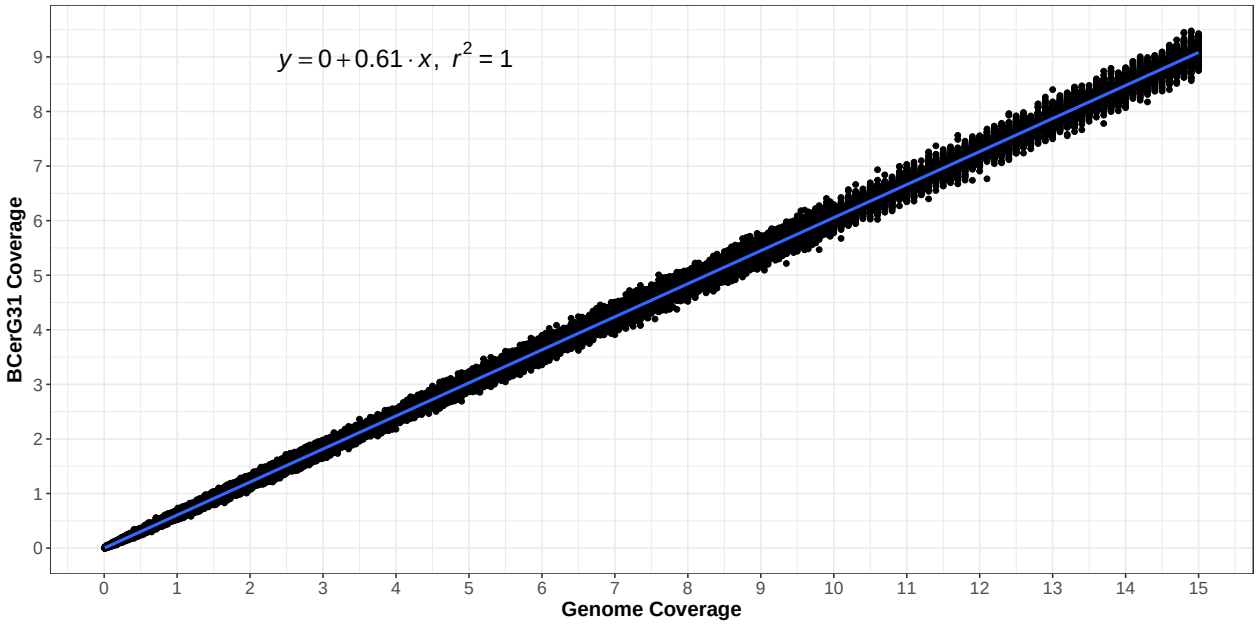
```
##
## Call:
## lm(formula = kmer_cov_mean ~ 0 + simulated_coverage, data = temp_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.55991 -0.04287 -0.00093  0.04227  0.52956
```

```
##
## 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")
```

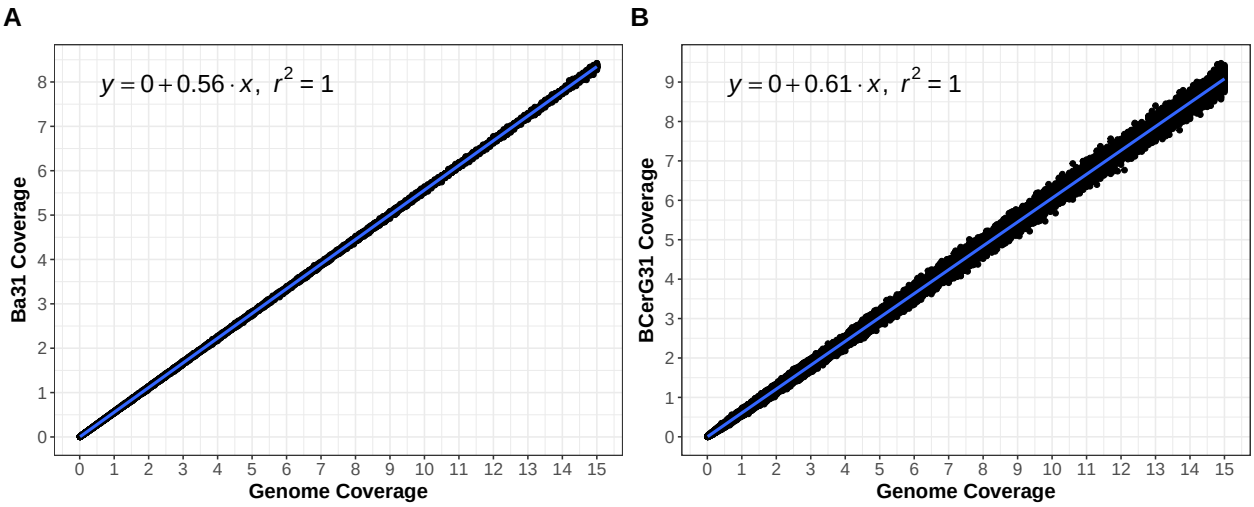
```
##
##  Pearson's product-moment correlation
##
## data:  temp_df$kmer_cov_mean and temp_df$simulated_coverage
## t = 6396.7, df = 48761, p-value < 0.00000000000000022
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9993940 0.9994152
## sample 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-01-31-genome-coverage.pdf'),
    height=6, width=12, onefile=FALSE)
grid.arrange(grobs=plots, ncol=2)
dev_null <- dev.off()

png(paste0(FIGURE, '/supplementary-figure-01-31-genome-coverage.png'),
    height=600, width=1200)
grid.arrange(grobs=plots, ncol=2)
dev_null <- dev.off()
```

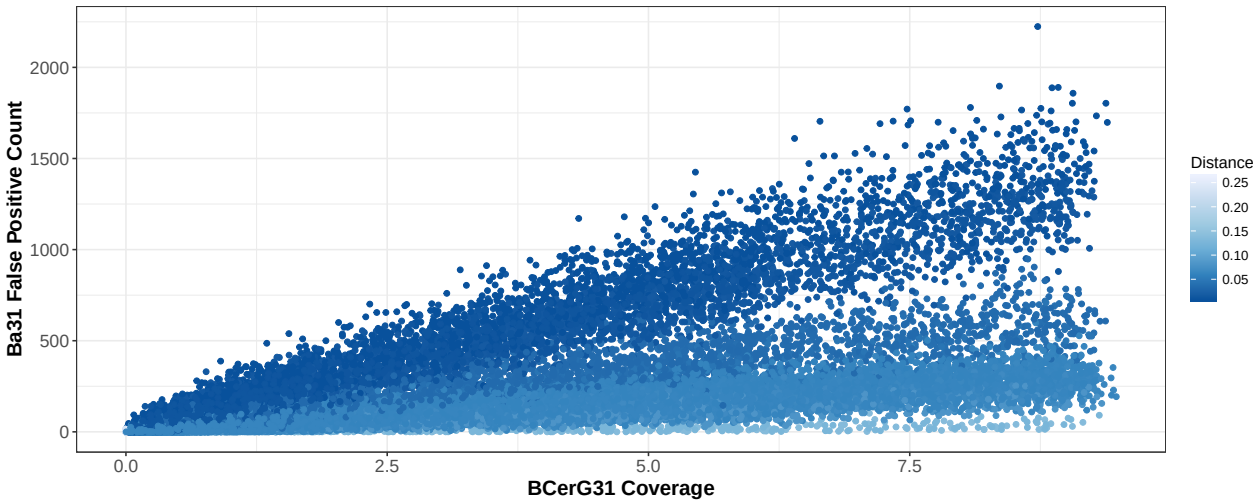
non-anthracis BCerG False Positive Model

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

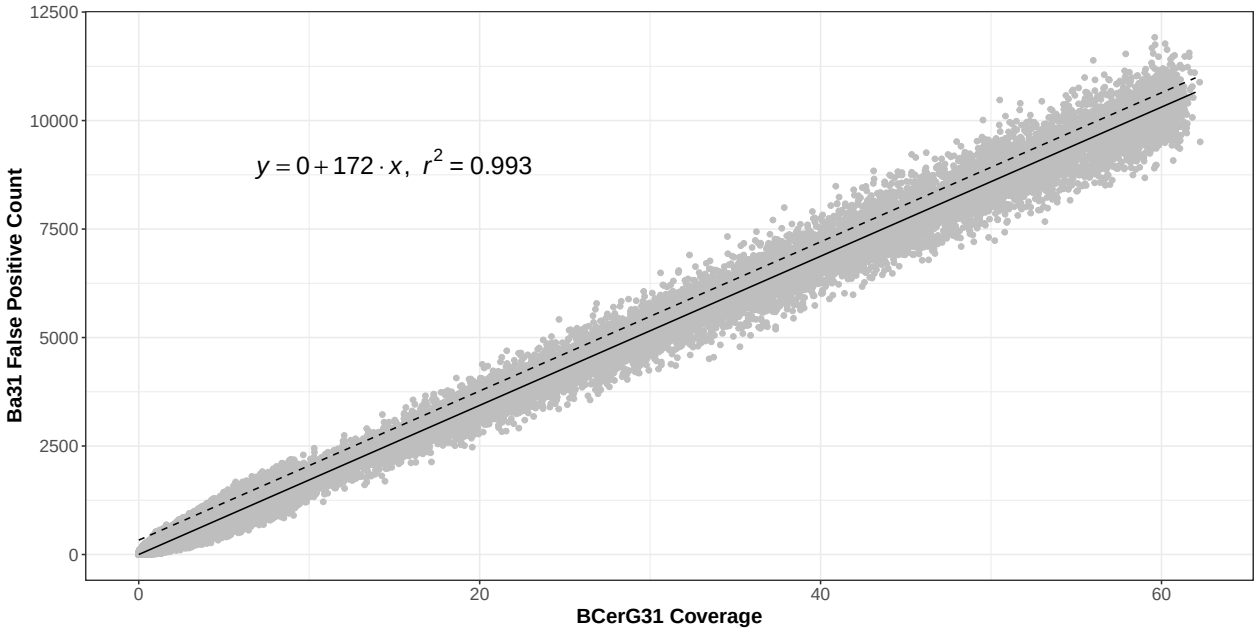
```
predict_false_positive(bcg_model, seq(0, 10, 2))
```

```
##   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))+
```

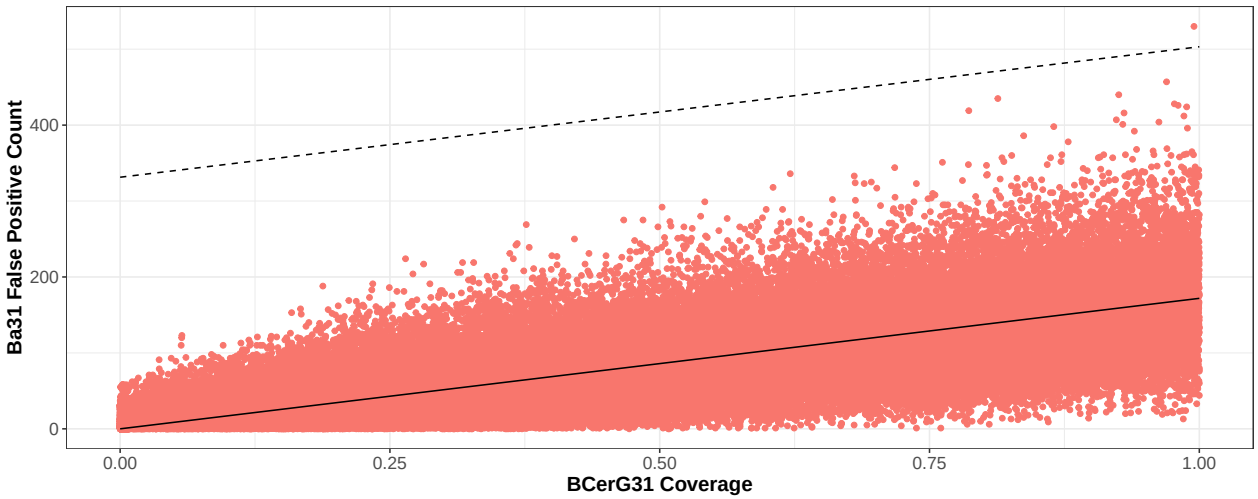
```
        geom_point(aes(color=is_ba)) +
        geom_line(data=predicted_fit, aes(x=bcg_cov_mean, y=fit.fit)) +
        geom_line(data=predicted_fit, aes(x=bcg_cov_mean, y=fit.upr), linetype="dashed")

    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)
```

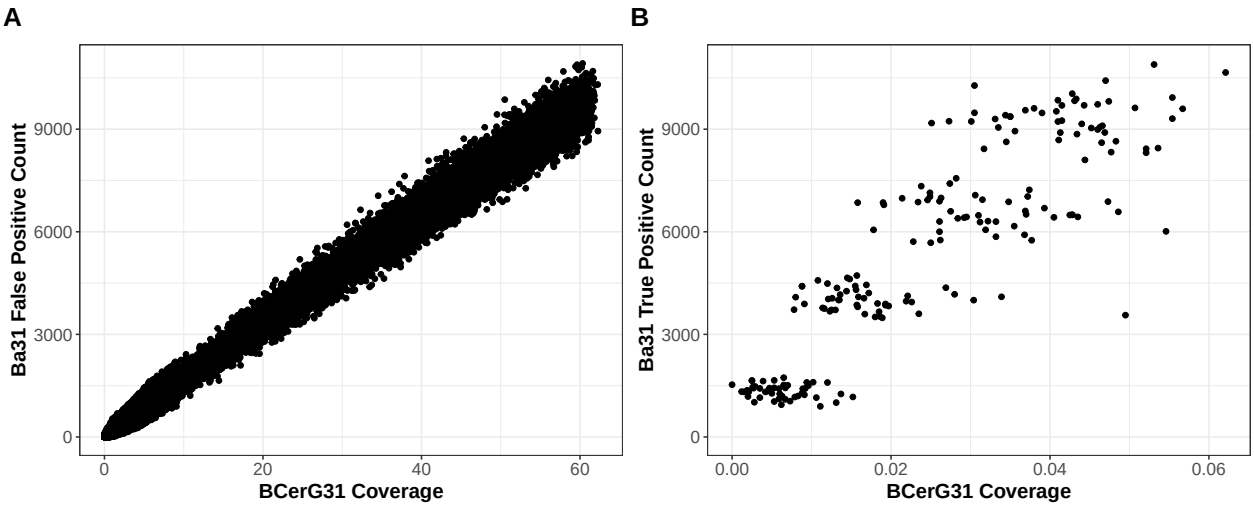
Plot the fit between 0 and 1x BCerG kmer coverage

```
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

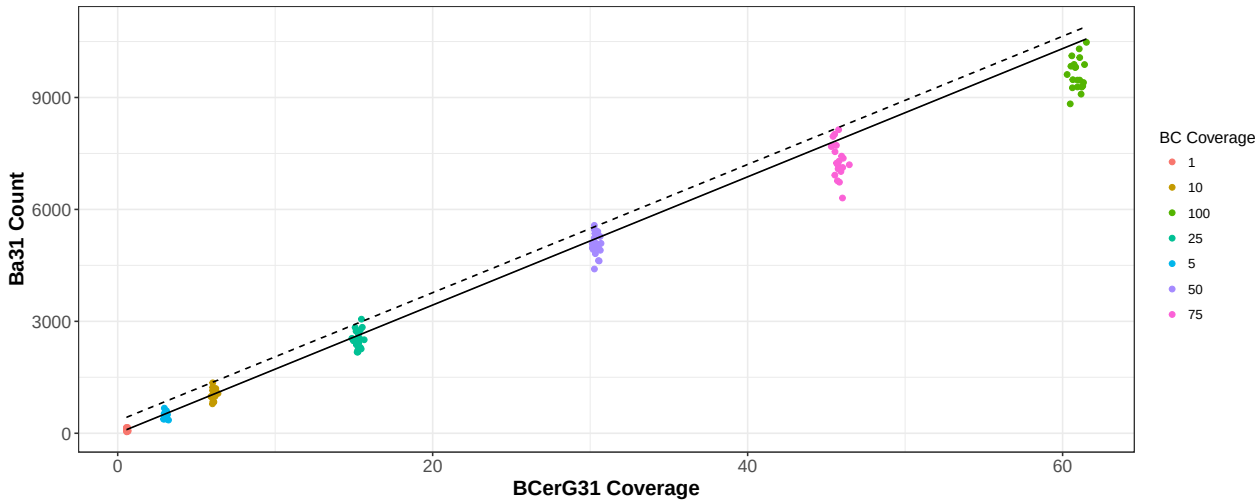
A total of **8** non-*B. anthracis* BCerG subsamples were used.

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

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

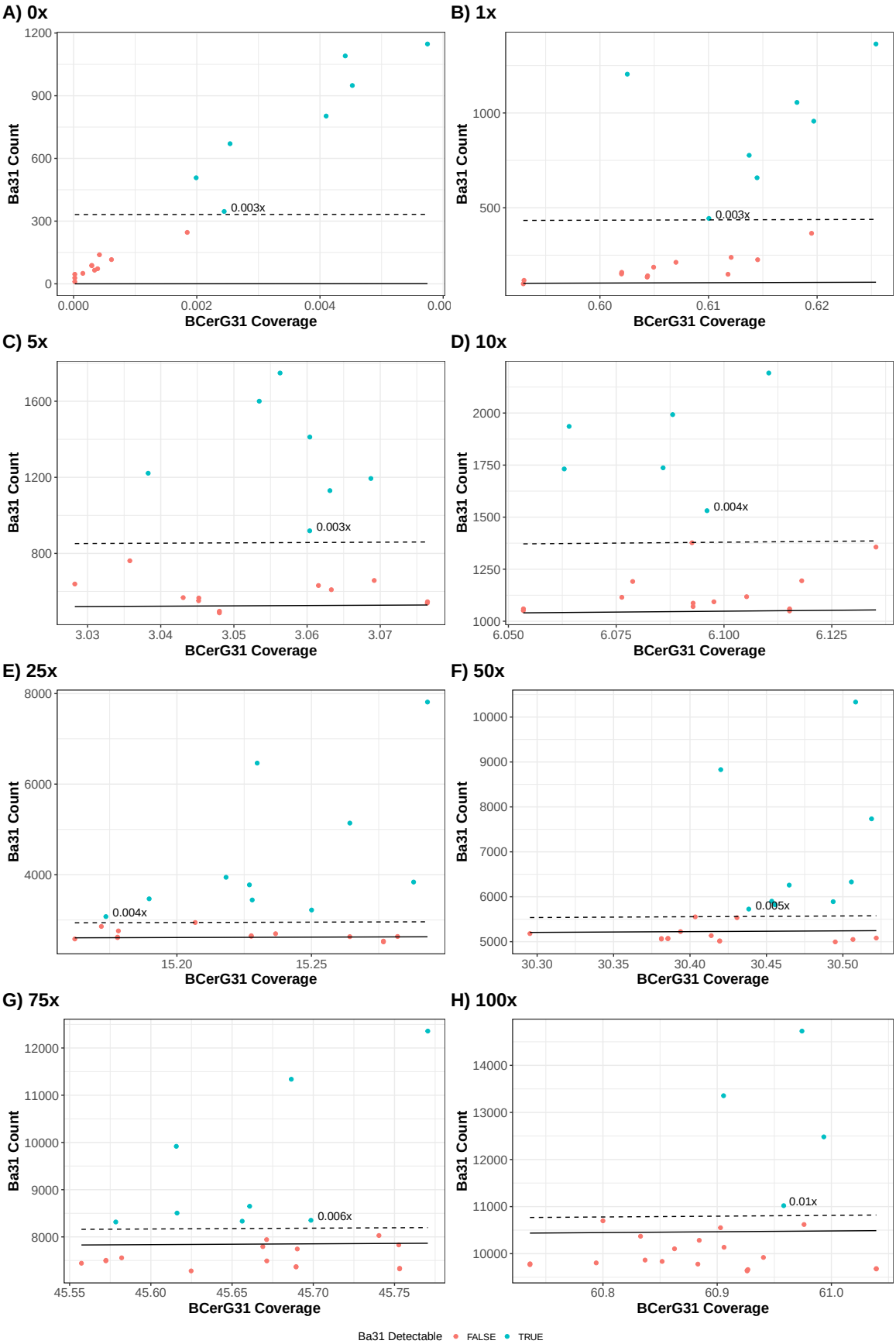
B. cereus Only

```
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
```