Univariate Report

iTIME Dev Team

15 July, 2021

Table 1: Variables selected in iTIME.Moffitt.org Univariate Summary.

Variable	Selection	Held as on Server
Include functions?	TRUE	input$printFunctions
Cell marker selected	Percent CD8 (Opal 520) Positive Cells	input$picked_marker
Clinical variable selected	race	input$picked_clinical
Threshold	1	input$choose_cont_thresh
Column heading of total cell counts	Total Cells	input$picked_total_cells
Clinical variable baseline	unknown	input$picked_modeling_reference

```r
#Function to produce Contingency Table
contingency_table <- function(summary_clinical_merge, markers = markers,
                              clin_vars = clin_vars, percent_threshold = percent_threshold){
  #Maybe provide an error for multiple columns
  cells <- summary_clinical_merge %>% select(any_of(paste(markers)))
  assign("percent_threshold", percent_threshold, envir = .GlobalEnv)

  above <- function(x, percent_threshold){ifelse(x > percent_threshold,
                                                 paste0('Greater than ', percent_threshold, '%'),
                                                 paste0('Less than ', percent_threshold, '%'))}

  table <- cells %>%
    mutate_all( ~ above(x = ., percent_threshold = as.numeric(percent_threshold))) %>%
    bind_cols(.,summary_clinical_merge %>% select(clin_vars)) %>%
    group_by(.[[paste(markers)]],.[[clin_vars]]) %>%
    summarize(n = n()) %>%
    pivot_wider(names_from = `.[[clin_vars]]`, values_from = n) %>%
    mutate_all(~replace_na(.,0))
  colnames(table)[1] = markers
  return(table)
}
```

```r
#Function use to produce the frequency table
freq_table_by_marker <-
  function(summary_clinical_merge,
           clinical = clinical,
           markers = markers) {
    cells <-
```

Table 2: Contingency table of Percent CD8 (Opal 520) Positive Cells above and below the threshold of 1 percent.

Percent CD8 (Opal 520) Positive Cells	black	unknown	white
Greater than 1%	7	139	6
Less than 1%	3	74	0

```
      summary_clinical_merge %>% select(paste(clinical),any_of(paste( markers)))

    table <-
      cells %>%
      mutate(`> 1%` = .[[paste( markers)]]>1,
             `> 2%` = .[[paste( markers)]]>2,
             `> 3%` = .[[paste( markers)]]>3,
             `> 4%` = .[[paste( markers)]]>4,
             `> 5%` = .[[paste( markers)]]>5) %>%
      group_by(.[[paste(clinical)]]) %>%
      select(`> 1%`,`> 2%`,`> 3%`,`> 4%`,`> 5%`) %>%
      summarize_all( ~ sum(.))
    colnames(table)[1] = clinical

    return(table)
}
```

Table 3: Frequency table of Percent CD8 (Opal 520) Positive Cells by race .

race	> 1%	> 2%	> 3%	> 4%	> 5%
black	7	5	4	3	3
unknown	139	76	45	27	19
white	6	6	5	4	3

```
#Function used to produce the summary table
summary_table = function(summary_clinical_merged,
                         marker,
                         clinical,
                         merged){
  data = summary_clinical_merged %>%
    select(paste(clinical), paste(merged), any_of(paste(marker)))
  colnames(data) = c("clinical","merged_var","marker")
  table =
    data %>%
    group_by(clinical) %>%
    summarise(Min = min(marker),
              Median = median(marker),
              Mean = mean(marker),
              Max = max(marker),
              SD = sd(marker),
              Subjects = length(unique(merged_var)),
              Samples = length(marker))
  colnames(table)[1] = clinical
  return(table)
}
```

Table 4: Summary table of Percent CD8 (Opal 520) Positive Cells by race showing the minimum, median, mean, max, and standard deviation as well as the number of unique subject and unique samples in the clinical and summary data frames.

race	Min	Median	Mean	Max	SD	Subjects	Samples
black	0.184065	2.111651	4.084717	15.23494	4.761006	10	10
unknown	0.000000	1.429593	2.165061	24.85137	2.772229	213	213
white	2.390378	5.954606	8.420667	21.14804	7.007699	6	6

Univariate Summary Plot

```
#Code within the server
message("Code within the server:")
cellvar <-  input$picked_marker
clinvar <- input$picked_clinical
colorscheme <- input$summaryPlotColors
data_table = summary_data_merged()

if(input$uni_transformation == "none"){
    thres = input$choose_cont_thresh
} else if(input$uni_transformation == "sqrt_transform"){
    data_table[,cellvar] = sqrt(data_table[,cellvar])
    thres = sqrt(as.numeric(input$choose_cont_thresh))
} else if(input$uni_transformation == "log2_transform"){
    data_table[,cellvar] = log2(data_table[,cellvar]+0.0001)
    thres = log2(as.numeric(input$choose_cont_thresh)+0.0001)
} else if(input$uni_transformation == "logit_transform"){
    p = (data_table[,cellvar]/100)+0.0001
    data_table[,cellvar] = log10(p/(1-p))
    tmp = (as.numeric(input$choose_cont_thresh)/100) + 0.0001
    thres = log10(tmp/(1-tmp))
}
plots = summary_plots_fn(data_table, clinvar, cellvar, colorscheme, thres)
plots[[as.integer(input$summaryPlotType)]]

#Function to produce all of the summary plots
summary_plots_fn <- function(datatable, clinvar, cellvar, colorscheme, threshold){
  box_p <- ggplot(datatable, aes(x=get(clinvar), y=get(cellvar), fill=get(clinvar))) +
    geom_boxplot() +
    xlab(str_to_title(clinvar)) + ylab(gsub("_", " ", str_to_title(cellvar))) +
    labs(fill=str_to_title(clinvar)) + theme_classic(base_size = 20) +
    viridis::scale_fill_viridis(option = colorscheme, discrete = TRUE) +
    geom_hline(yintercept = as.numeric(threshold), size = 1.25,
               linetype = "twodash", color = 'red') +
    theme(legend.position = 'none')
```

```r
  violin_p <- ggplot(datatable, aes(x=get(clinvar), y=get(cellvar), fill=get(clinvar))) +
    geom_violin() +
    xlab(str_to_title(clinvar)) + ylab(gsub("_", " ", str_to_title(cellvar))) +
    labs(fill=str_to_title(clinvar)) + theme_classic(base_size = 20) +
    viridis::scale_fill_viridis(option = colorscheme, discrete = TRUE) +
    geom_hline(yintercept = as.numeric(threshold), size = 1.25,
               linetype = "twodash", color = 'red') +
    theme(legend.position = 'none')

  hist_p <- ggplot(datatable, aes(x=get(cellvar), color=get(clinvar))) +
    geom_histogram(position='stack', fill = 'white') + facet_wrap(get(clinvar)~., nrow = 1) +
    xlab(str_to_title(gsub("_", " ", cellvar))) + ylab("Count") +
    labs(fill=str_to_title(clinvar)) + theme_classic(base_size = 20) +
    viridis::scale_color_viridis(option = colorscheme, discrete = TRUE) +
    theme(legend.position = 'none') +
    geom_vline(xintercept = as.numeric(threshold),
               size = 1.25, linetype = "twodash", color = 'red')

  if(is.character(datatable[[clinvar]])){
    scatter_p <- ggplot(datatable, aes(x=get(clinvar), y=get(cellvar), color=get(clinvar))) +
      geom_point() +
      xlab(str_to_title(clinvar)) + ylab(gsub("_", " ", str_to_title(cellvar))) +
      labs(color=str_to_title(clinvar)) + theme_classic(base_size = 20) +
      viridis::scale_color_viridis(option = colorscheme, discrete=TRUE)
  }
  else{
    scatter_p <- ggplot(datatable, aes(x=get(clinvar), y=get(cellvar), color=get(clinvar))) +
      geom_point() +
      xlab(str_to_title(clinvar)) + ylab(gsub("_", " ", str_to_title(cellvar))) +
      labs(color=str_to_title(clinvar)) + theme_classic(base_size = 20) +
      viridis::scale_color_viridis(option = colorscheme, discrete=FALSE)
  }

  summ_plots <- list(box_p, violin_p, hist_p, scatter_p)

  return(summ_plots)
}
```

Cumulative Distribution Function Plot

```r
#Code within the server
marker = input$picked_marker
data_table = summary_data_merged()
CDF_plots(summary_data_merge = data_table, markers = substr(marker, 9, nchar(marker)))

#Function used to produce the CDF plot
CDF_plots = function(summary_data_merged = summary_data_merged, markers = markers){
  sample_stats = summary_data_merged %>% select(grep('Total', colnames(.)), markers) %>%
    pivot_longer(cols = 2:ncol(.), values_to = 'Count', names_to = 'Marker') %>%
    group_by(Marker) %>%
    summarize(prob0 = mean(Count == 0, na.rm = TRUE),
```

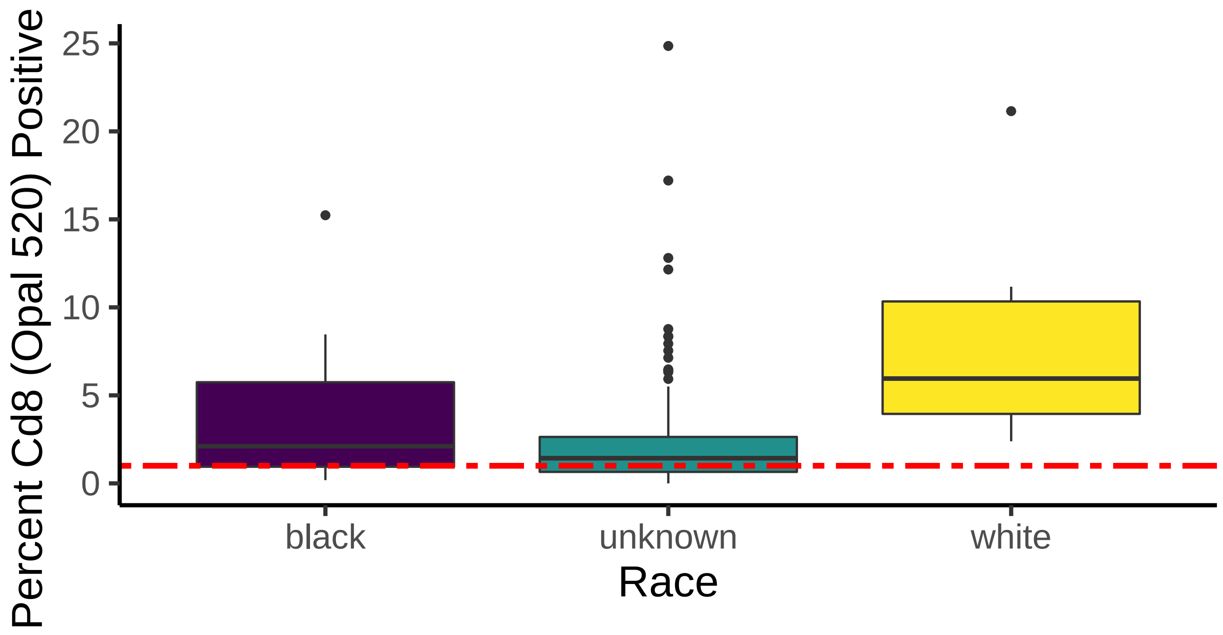

Figure 1: Summary plot of Percent CD8 (Opal 520) Positive Cells separated by race. The red line is at the threshold level in Table 1.

```
               Avg_p = mean(Count/`Total Cells`, na.rm = TRUE),
               Avg_Count = mean(Count, na.rm = TRUE),
               Avg_Total = round(mean(`Total Cells`, na.rm = TRUE)))

cdfs = summary_data_merged %>%  select(grep('Total', colnames(.)), markers) %>%
  pivot_longer(cols = 2:ncol(.), names_to = 'Marker', values_to = 'Count') %>%
  mutate(ecdf = ecdf(Count)(Count)) %>%
  mutate(Poisson =  ppois(q = Count,
                          lambda = sample_stats$Avg_Count),
         Binomial = pbinom(q = Count,
                           size = round(sample_stats$Avg_Total),
                           prob = sample_stats$Avg_p,
         ),
         `ZI Poisson` =  pzipois(q = Count,
                                 lambda = sample_stats$Avg_Count,
                                 pstr0 = sample_stats$prob0
         ),
         `ZI Binomial` = pzibinom(q = Count,
                                  size = round(sample_stats$Avg_Total),
                                  prob = sample_stats$Avg_p,
                                  pstr0 = sample_stats$prob0
         ),
         `Negative Binomial` =  pnbinom(q = Count,
                                        size = round(sample_stats$Avg_Count),
                                        prob = 1 - sample_stats$prob0
         ),
         `Beta Binomial` =  pbetabinom(q = Count,
                                       size = round(sample_stats$Avg_Total),
                                       prob = sample_stats$Avg_p,
                                                         rho = sample_stats$prob0)

    ) %>%
    pivot_longer(col = 5:ncol(.), values_to = 'CDF', names_to = 'Distribution')

  cdfs = cdfs %>%
    mutate(family = ifelse(Distribution %in% c('Poisson', 'ZI Poisson', 'Negative Binomial'),
                           'Poisson', 'Binomial'),
           Distribution = factor(Distribution))

  binomial_plot = cdfs %>%
    ggplot(aes(x = Count, y = ecdf, color = 'Empirical')) +
    geom_line(aes(color = 'Empirical'), color = 'black') +
    geom_line(aes(x = Count, y = CDF, color = Distribution, linetype = family)) + theme_bw() +
    theme(axis.text.x = element_blank(),
          axis.ticks = element_blank(),
          axis.title.x = element_blank(),
          legend.position = "right", #c(0.8, 0.3),
          axis.title.y = element_text(size = 16),
          axis.text.y = element_text(size = 16),
          strip.text = element_text(size=16),
          legend.text = element_text(size = 16),
          legend.title = element_text(size = 16)) +
    labs(color = 'Distribution') + scale_linetype_manual(values = c("solid", "longdash"))

  return(binomial_plot)
}
```

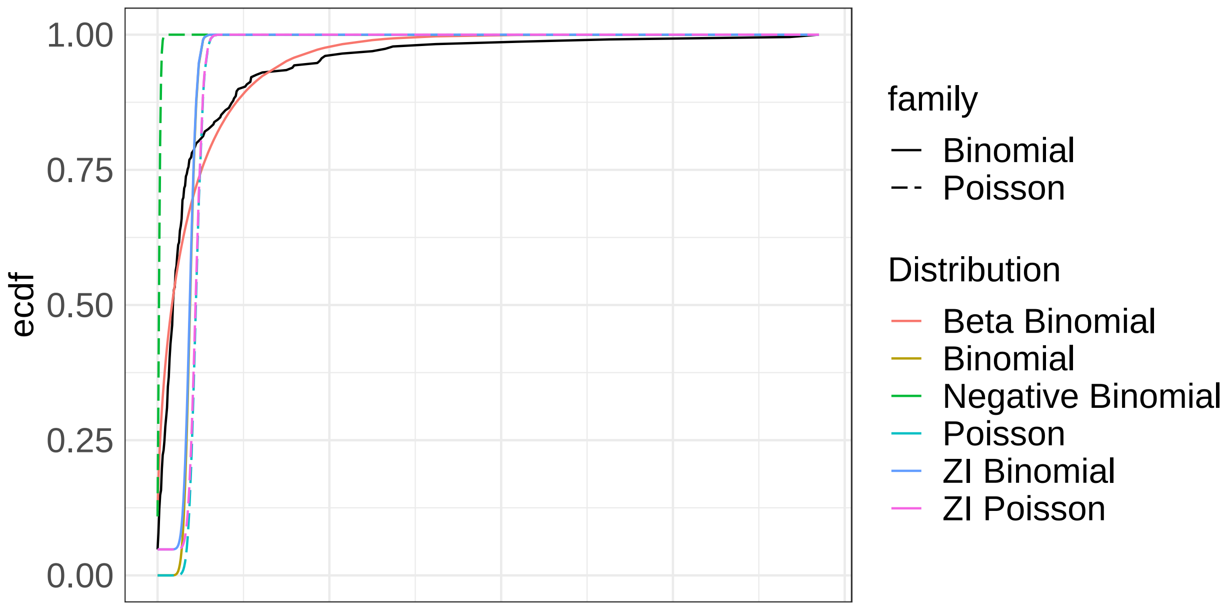

Figure 2: Cumulative distribution functions for both the binomial and Poisson families for Percent CD8 (Opal 520) Positive Cells. The default model selected in iTIME.Moffitt.org is the negative binomial model.