# Normalizing

Divya Prima Crasta-237879

2024-11-12

```
library(ggplot2)
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(tidyr)
library(gridExtra)
```

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

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

```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(knitr)
library(corrplot)
```

```
## corrplot 0.95 loaded
```

```
load('combined_assay_data.RData')
```

```r
X <- assay.data[,3:length(assay.data)]
assay.data$Toxicity <- factor(assay.data$Toxicity, levels = unique(assay.data$Toxicity))
y <- assay.data$Toxicity
```

```r
# before logarithmizing
X_normalized <- apply(X[,2:length(X)], 2, function(x) x - X$Cmax)
```

# Univariate analysis

## Summary statistics

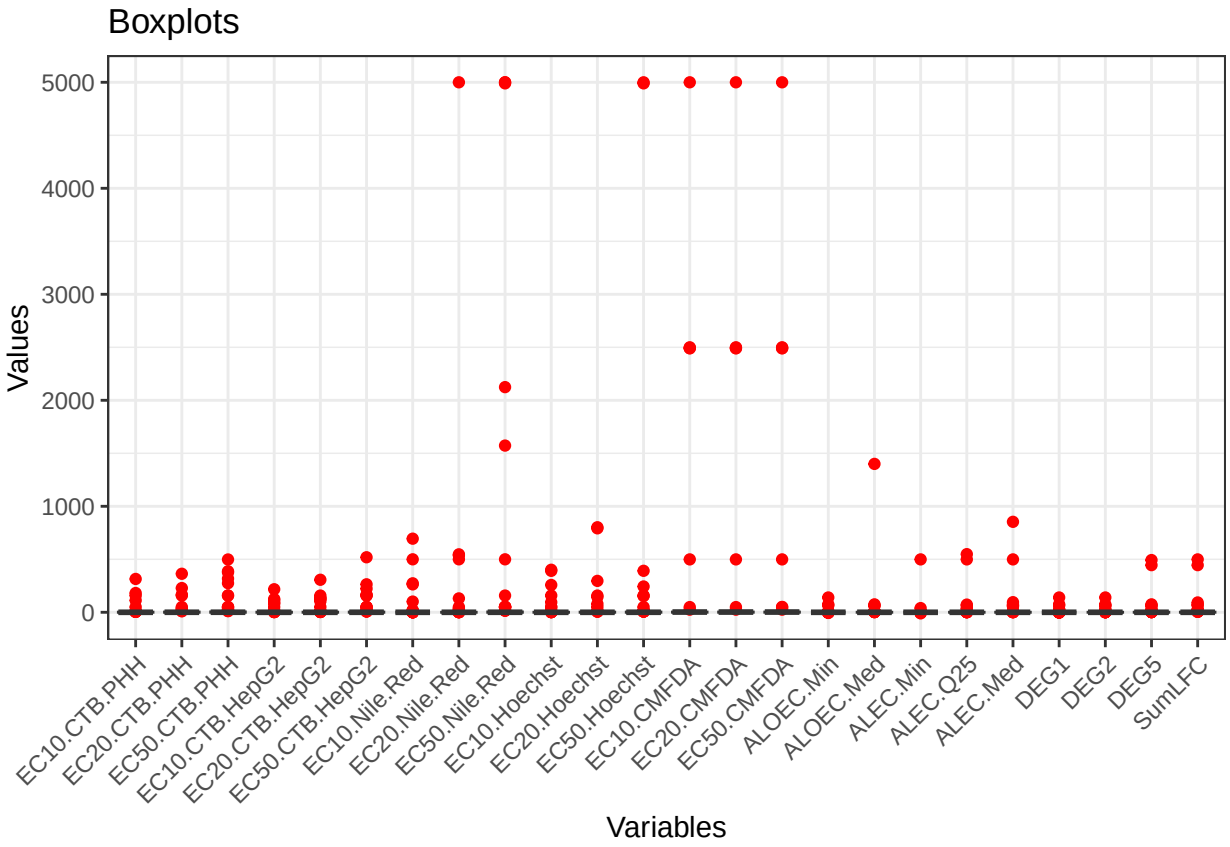
```r
# Function to calculate the required statistics for each column
summary_stats <- function(x) {
  c(
    minimum = min(x),
    Q1 = quantile(x, 0.25),
    median = median(x),
    mean = mean(x, na.rm = TRUE),
    Q3 = quantile(x, 0.75),
    maximum = max(x),
    variance = var(x),
    sd = sd(x),
    range = max(x) - min(x)
  )
}

# Apply the function to each column of the selected data
normalized_summary_df <- as.data.frame(t(apply(X_normalized, 2, summary_stats)))

# Print the result
print(normalized_summary_df)
```

```
##                    minimum        Q1.25%       median       mean     Q3.75%
## EC10.CTB.PHH   -0.007762404  3.309113e-02 2.044170e-01   9.138898 1.57484713
## EC20.CTB.PHH   -0.006875404  4.240748e-02 3.395719e-01  11.448512 2.29765775
## EC50.CTB.PHH   -0.005319404  8.436536e-02 7.082732e-01  25.204183 3.80918376
## EC10.CTB.HepG2 -0.083363000  1.899247e-02 1.040546e-01   7.595209 0.90374870
## EC20.CTB.HepG2 -0.001636305  3.239827e-02 1.858485e-01  10.899380 1.29379018
## EC50.CTB.HepG2 -0.000991298  6.538990e-02 3.375496e-01  17.871516 2.47637046
## EC10.Nile.Red  -0.649868351  5.998640e-03 3.674132e-02  19.108779 0.23612388
## EC20.Nile.Red  -0.046552279  1.443276e-02 8.578657e-02  68.964544 0.49203619
## EC50.Nile.Red  -0.023375293  6.170916e-02 4.427895e-01 198.186811 3.37338534
## EC10.Hoechst   -0.107019285  1.952632e-02 8.185485e-02  14.721234 0.73064420
## EC20.Hoechst   -0.024270697  3.682121e-02 1.180755e-01  24.113290 1.85069702
## EC50.Hoechst   -0.001494856  7.026302e-02 3.287699e-01 111.441390 2.49595471
## EC10.CMFDA     -0.015711105  4.990961e-01 2.483185e+00 132.116748 2.49981925
## EC20.CMFDA     -0.013411678  4.999040e-01 2.483185e+00 132.132939 2.49981925
## EC50.CMFDA     -0.005927245  4.999963e-01 2.492753e+00 132.739345 2.49989849
## ALOEC.Min      -6.148000000 -3.834621e-03 5.858060e-04   2.947964 0.01981741
## ALOEC.Med      -0.049844507  7.315891e-04 4.107450e-02  16.754342 0.49853001
```

```
## ALEC.Min       -9.962211999 -4.338442e-03 5.181465e-05   5.426345 0.01590561
## ALEC.Q25       -0.170062880  1.053011e-04 4.226874e-02  12.821395 0.45686087
## ALEC.Med       -0.154089591  5.116237e-03 8.045704e-02  17.010121 0.55539543
## DEG1           -3.988000000 -4.639000e-04 3.431257e-03   3.752479 0.05436765
## DEG2           -0.179622098  2.567885e-05 1.243010e-02   4.439505 0.31567660
## DEG5           -0.179622098  1.893438e-04 4.524440e-02  12.650031 0.97898936
## SumLFC         -0.179622098  6.335976e-04 8.105122e-02  13.300521 1.66126441
##                   maximum    variance        sd     range
## EC10.CTB.PHH    314.6036   1659.3283  40.73485  314.6114
## EC20.CTB.PHH    363.5386   2314.1995  48.10613  363.5455
## EC50.CTB.PHH    498.2800   7197.3277  84.83707  498.2854
## EC10.CTB.HepG2  217.5594    897.5322  29.95884  217.6427
## EC20.CTB.HepG2  306.9219   1733.6636  41.63729  306.9236
## EC50.CTB.HepG2  519.1888   4481.5361  66.94428  519.1898
## EC10.Nile.Red   694.4848   8596.1514  92.71543  695.1347
## EC20.Nile.Red  4999.8920 256391.9112 506.35157 4999.9386
## EC50.Nile.Red  4999.9942 790395.6977 889.04201 5000.0176
## EC10.Hoechst    401.1805   4020.9579  63.41102  401.2876
## EC20.Hoechst    802.3669  13699.9147 117.04663  802.3911
## EC50.Hoechst   4999.9942 494158.0284 702.96375 4999.9957
## EC10.CMFDA     4999.8920 425957.9696 652.65456 4999.9077
## EC20.CMFDA     4999.8920 425953.6678 652.65126 4999.9054
## EC50.CMFDA     4999.8920 425816.8599 652.54644 4999.8979
## ALOEC.Min       139.8920    290.2402  17.03644  146.0400
## ALOEC.Med      1399.8920  19651.5080 140.18384 1399.9418
## ALEC.Min        499.1267   2504.6202  50.04618  509.0889
## ALEC.Q25        548.6841   5508.1717  74.21706  548.8542
## ALEC.Med        853.6160   9807.2029  99.03132  853.7701
## DEG1            139.8920    300.2056  17.32644  143.8800
## DEG2            139.8920    328.5098  18.12484  140.0716
## DEG5            492.8520   4443.3733  66.65863  493.0316
## SumLFC          499.1267   4543.6748  67.40679  499.3063
```

## Boxplots

```
# Convert to long format
long_data <- as.data.frame(X_normalized) %>%
  pivot_longer(cols = everything(), names_to = "Variable", values_to = "Value")
# Ensure 'Variable' is a factor and ordered based on its appearance in the dataset
long_data$Variable <- factor(long_data$Variable, levels = colnames(X))

# Create boxplots for all numeric columns in one chart
ggplot(long_data, aes(x = Variable, y = Value)) +
  geom_boxplot(fill = "lightblue", outlier.color = "red") +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Boxplots", x = "Variables", y = "Values")
```

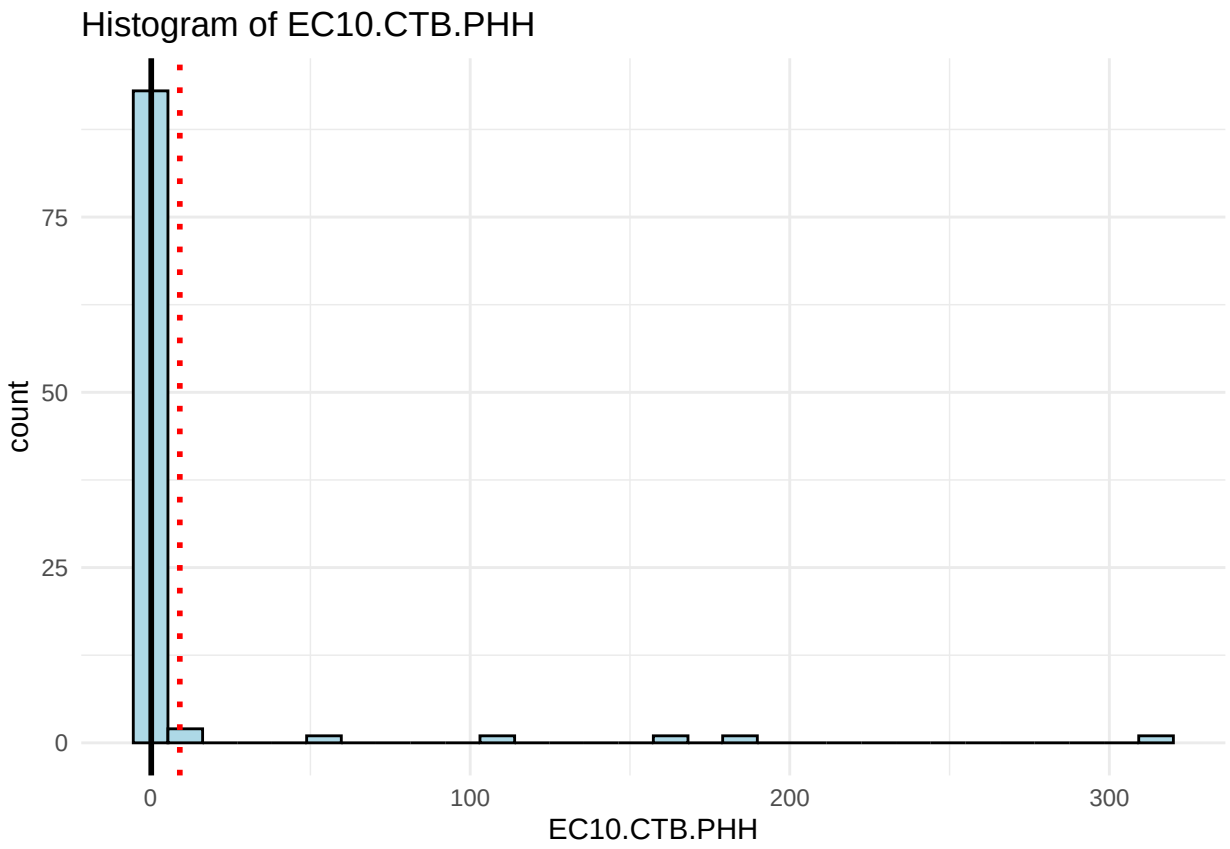



## Histograms

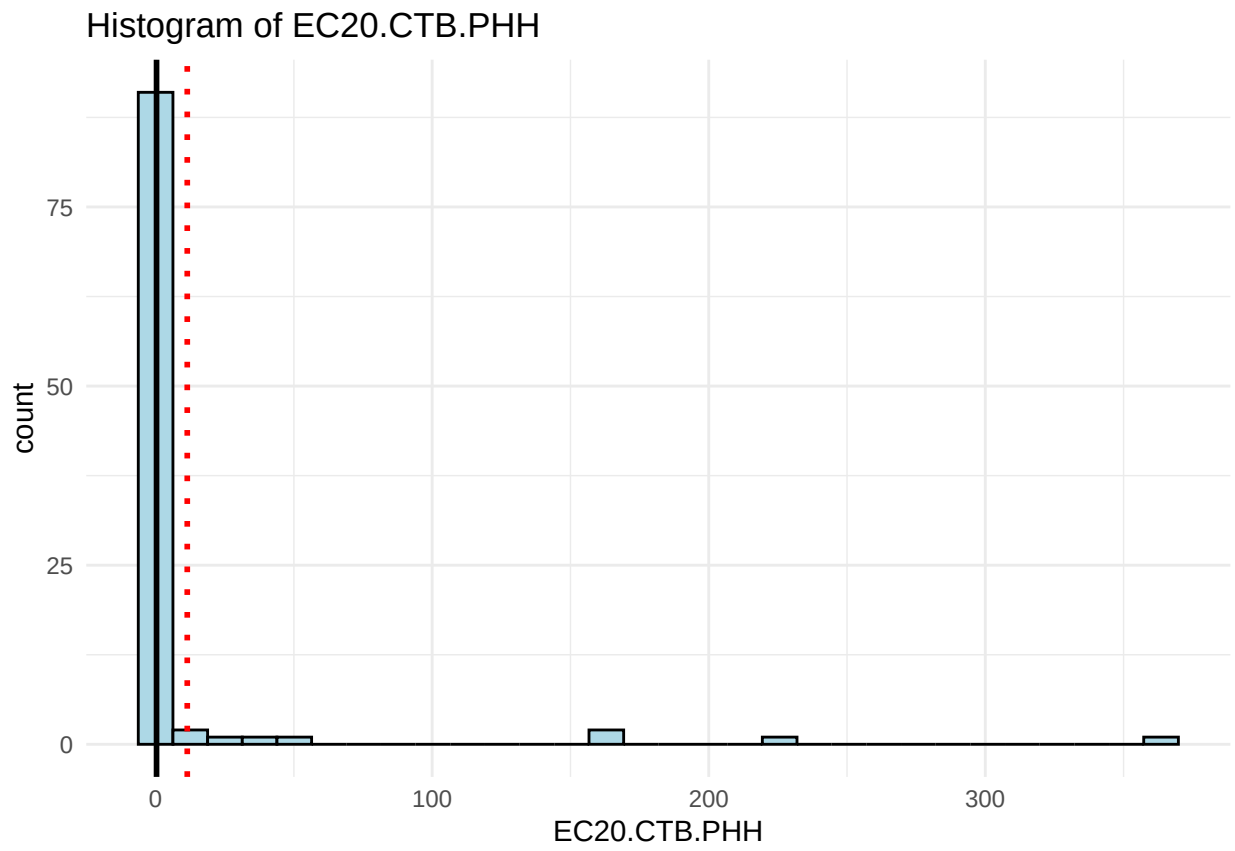
```
# Loop through each column of the dataset 'X'
for (i in 1:ncol(X_normalized)) {

  # Get the column name
  col_name <- colnames(X_normalized)[i]

  # Extract the mean and median for the current column from summary_df
  mean_val <- normalized_summary_df[col_name, "mean"]
  median_val <- normalized_summary_df[col_name, "median"]

  # Create the histogram and add vertical lines for mean and median
  gg <- ggplot(X_normalized, aes(x = get(col_name))) +
    geom_histogram( fill = "lightblue", color = "black") +
    geom_vline(aes(xintercept = mean_val), linetype = "dotted", color = "red", size = 1) + # Dotted red
    geom_vline(aes(xintercept = median_val), linetype = "solid", color = "black", size = 1) + # Solid b
    xlab(col_name) +
    theme_minimal() +
    labs(title = paste("Histogram of", col_name))

  # Print the plot
  print(gg)
}
```

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

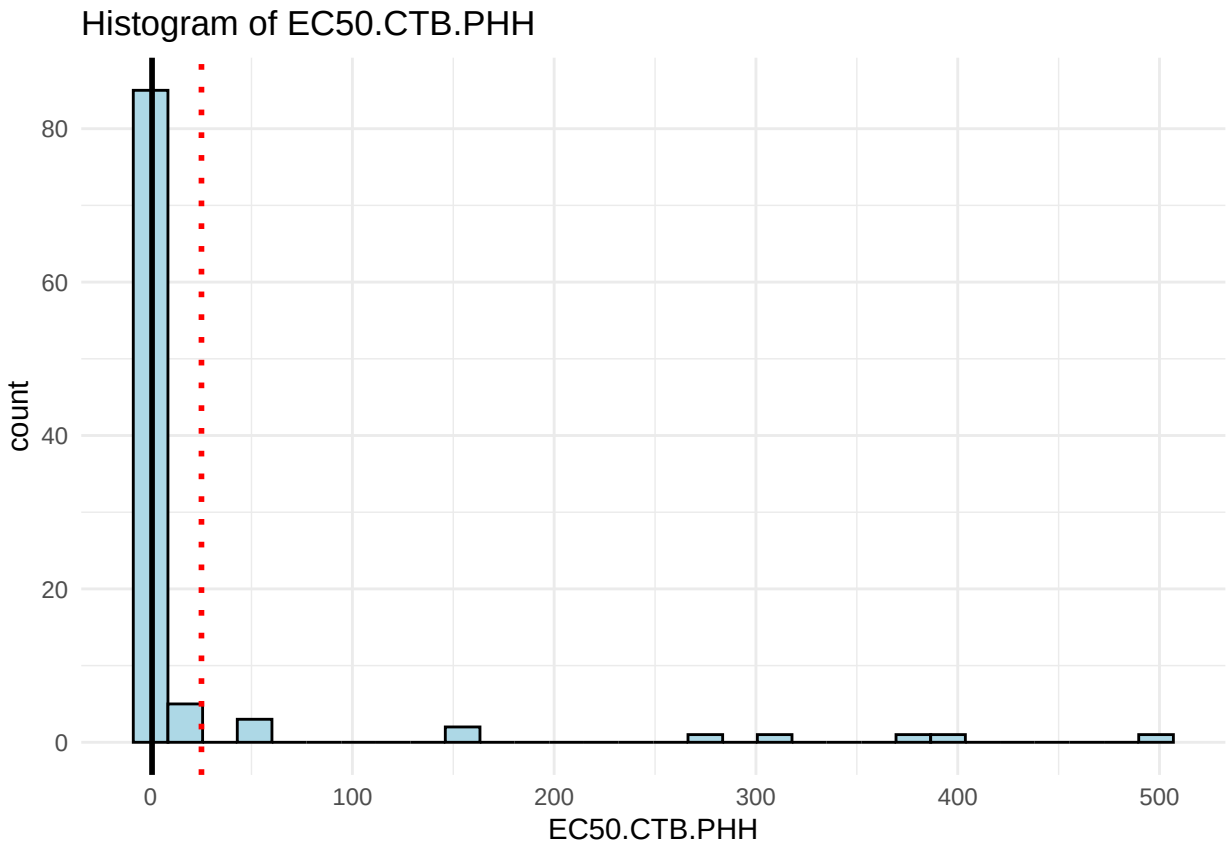
```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```



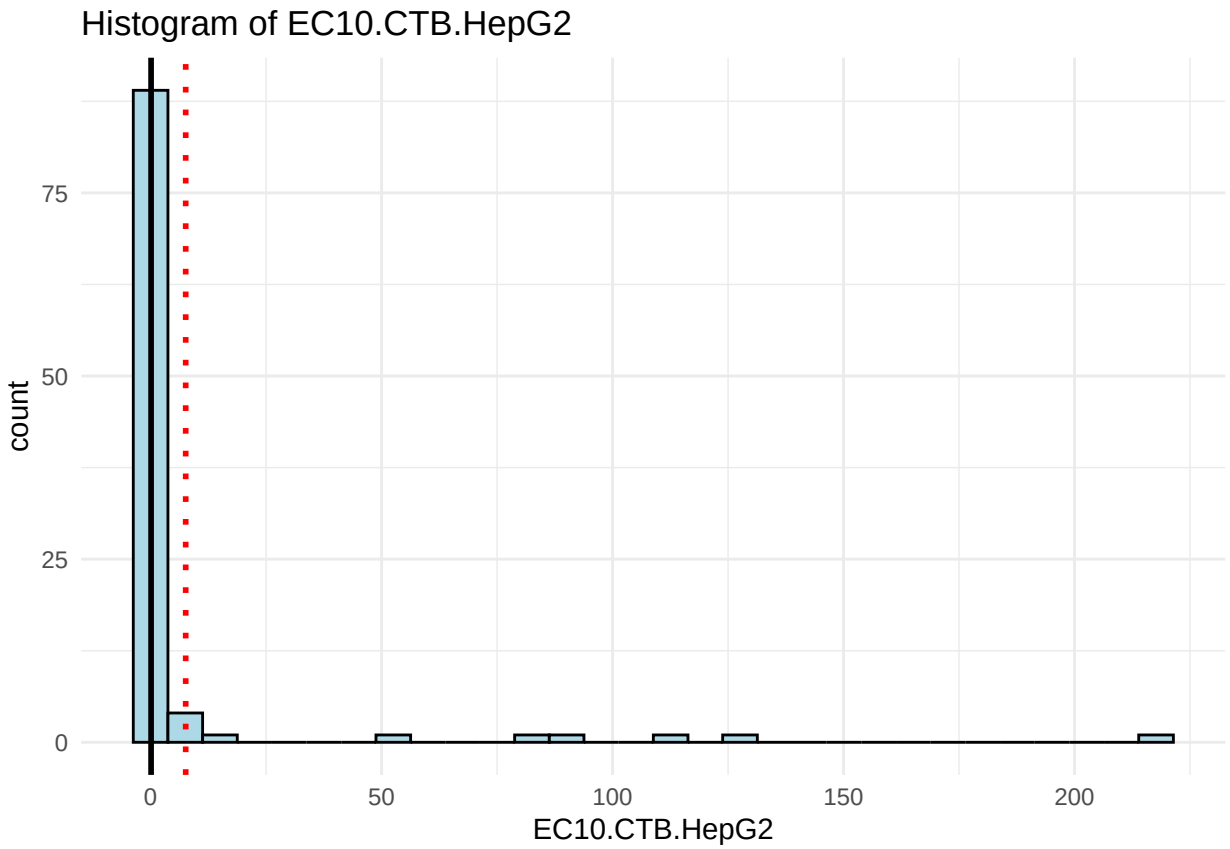

```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```



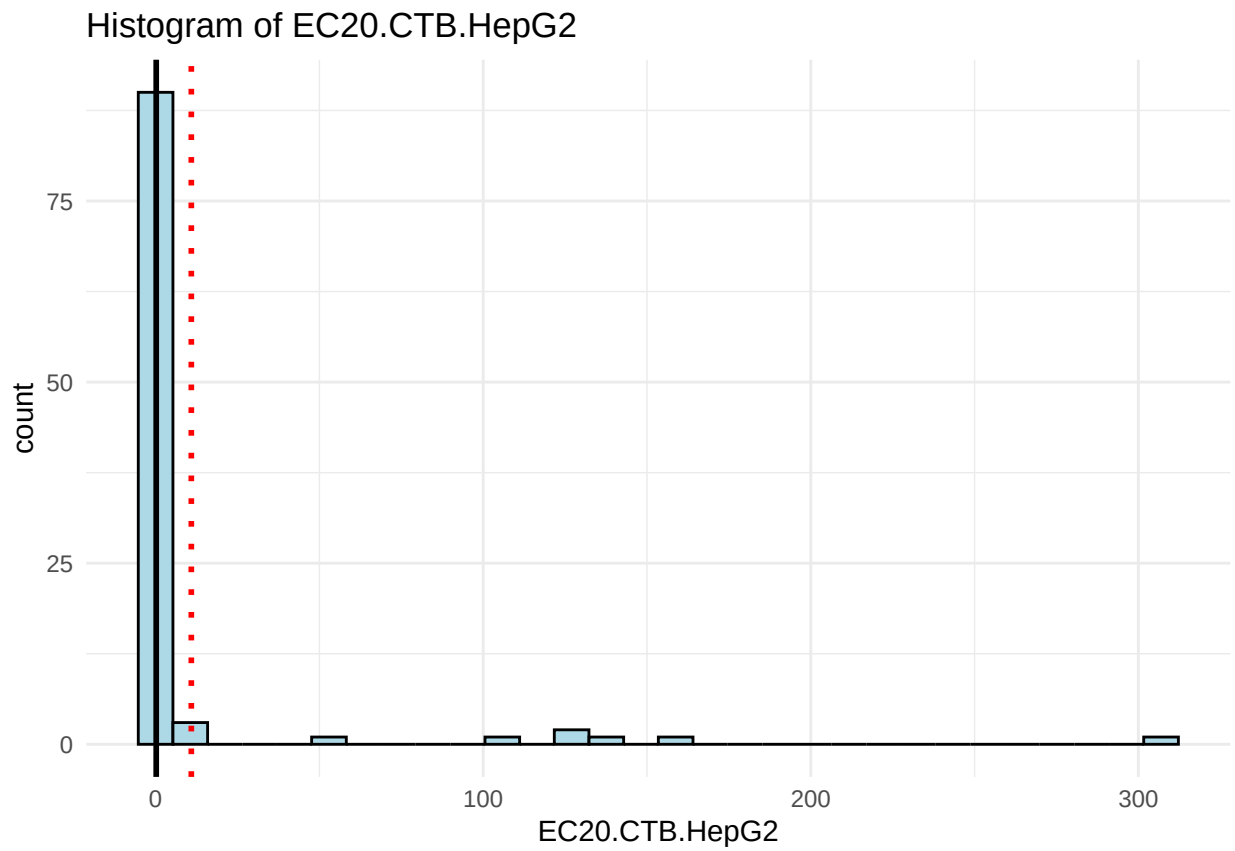

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



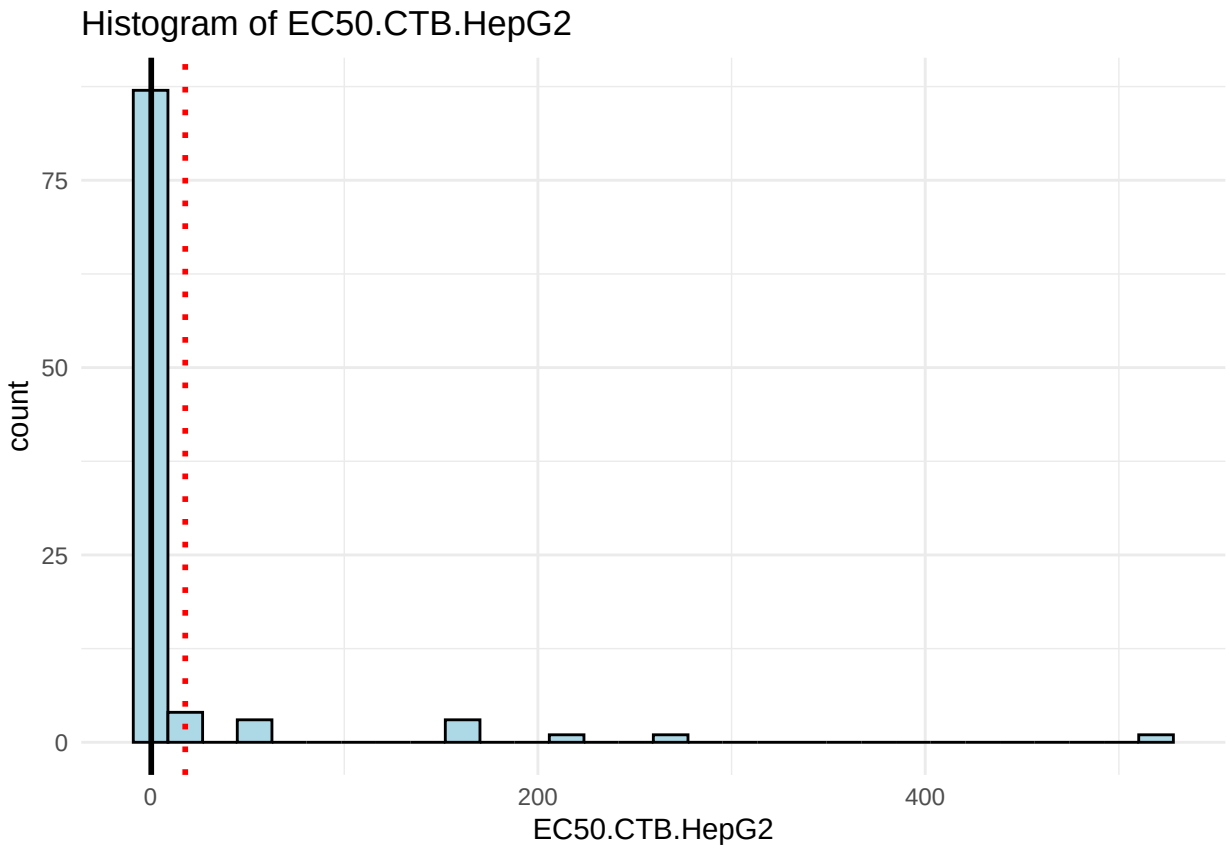

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



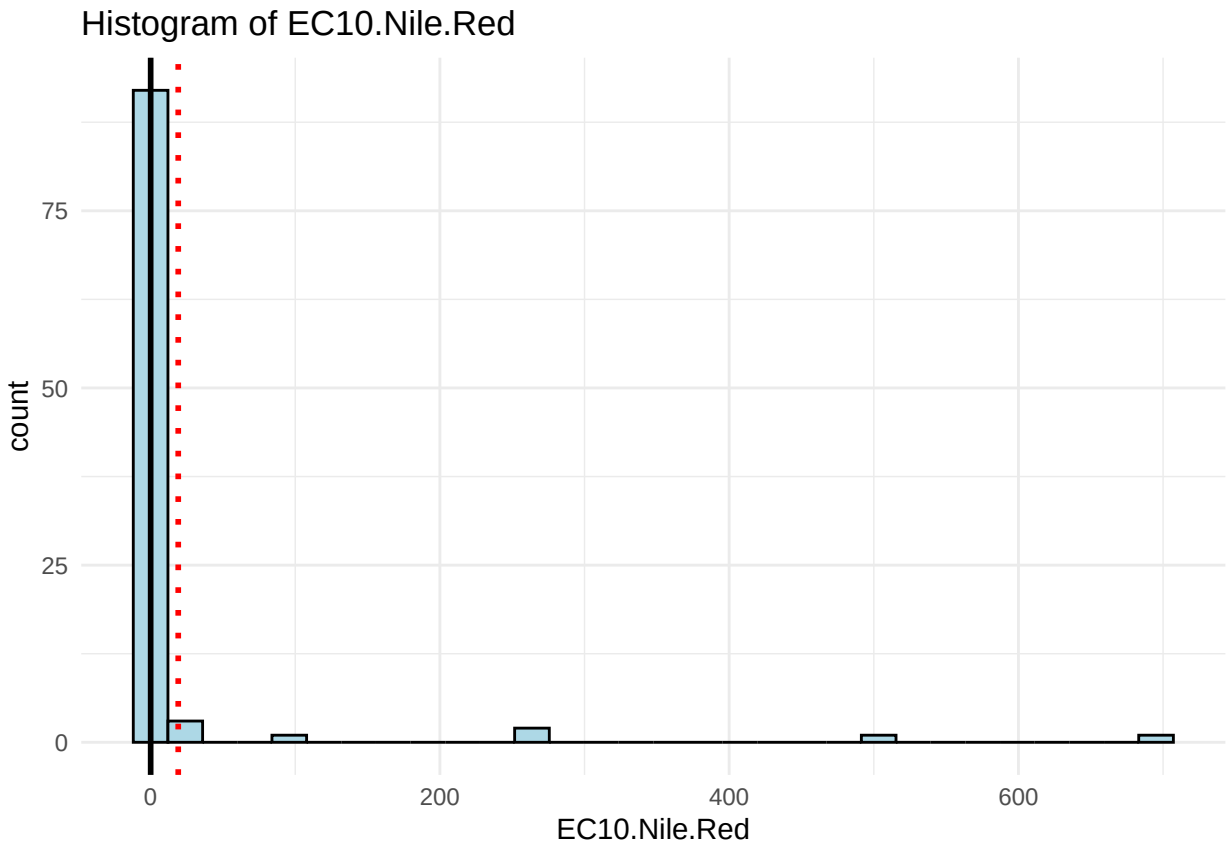

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



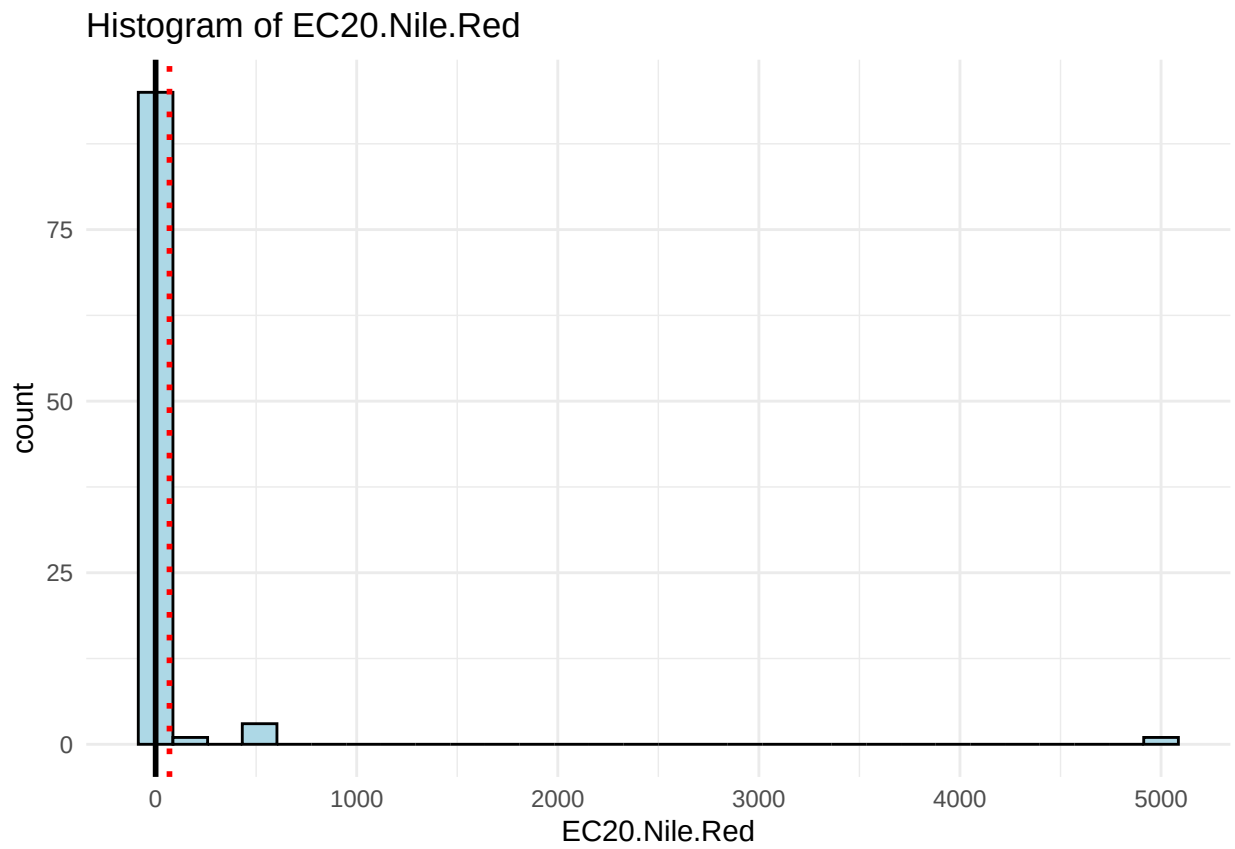

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



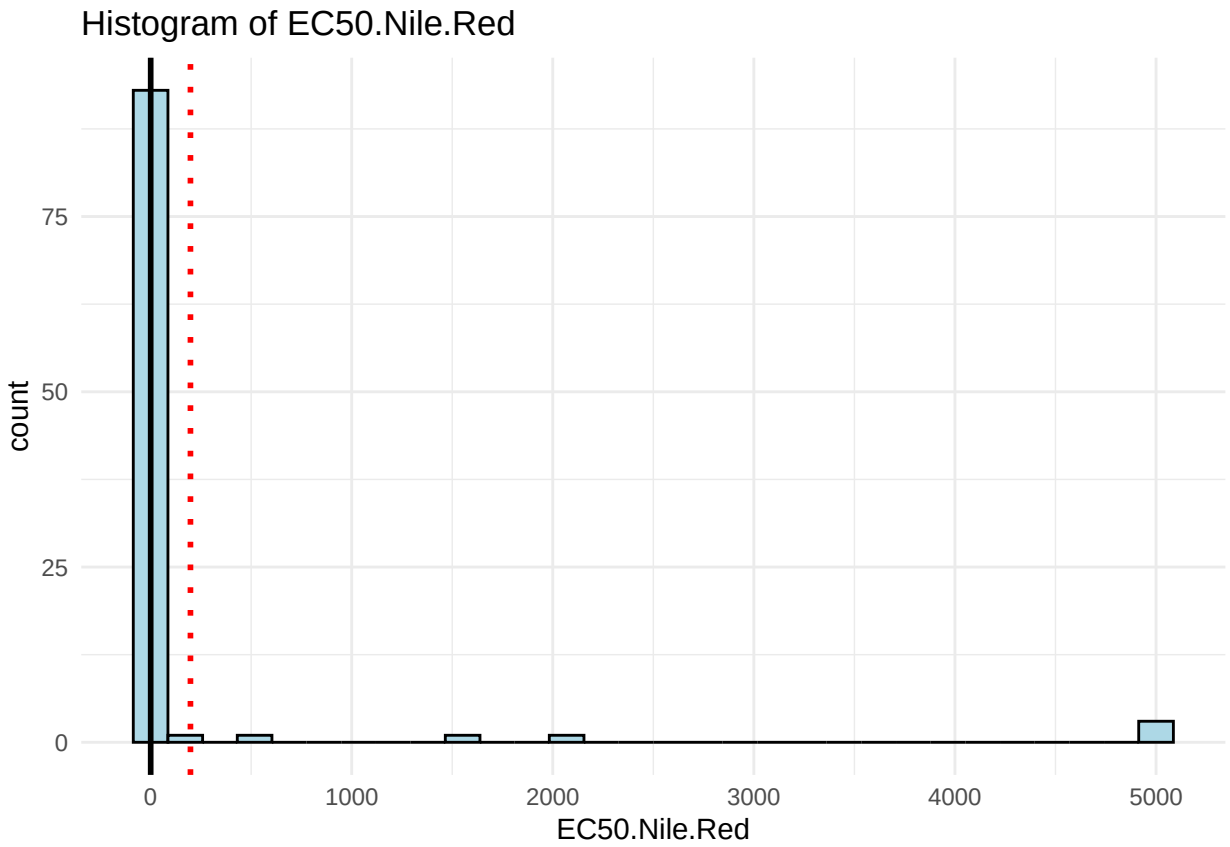

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



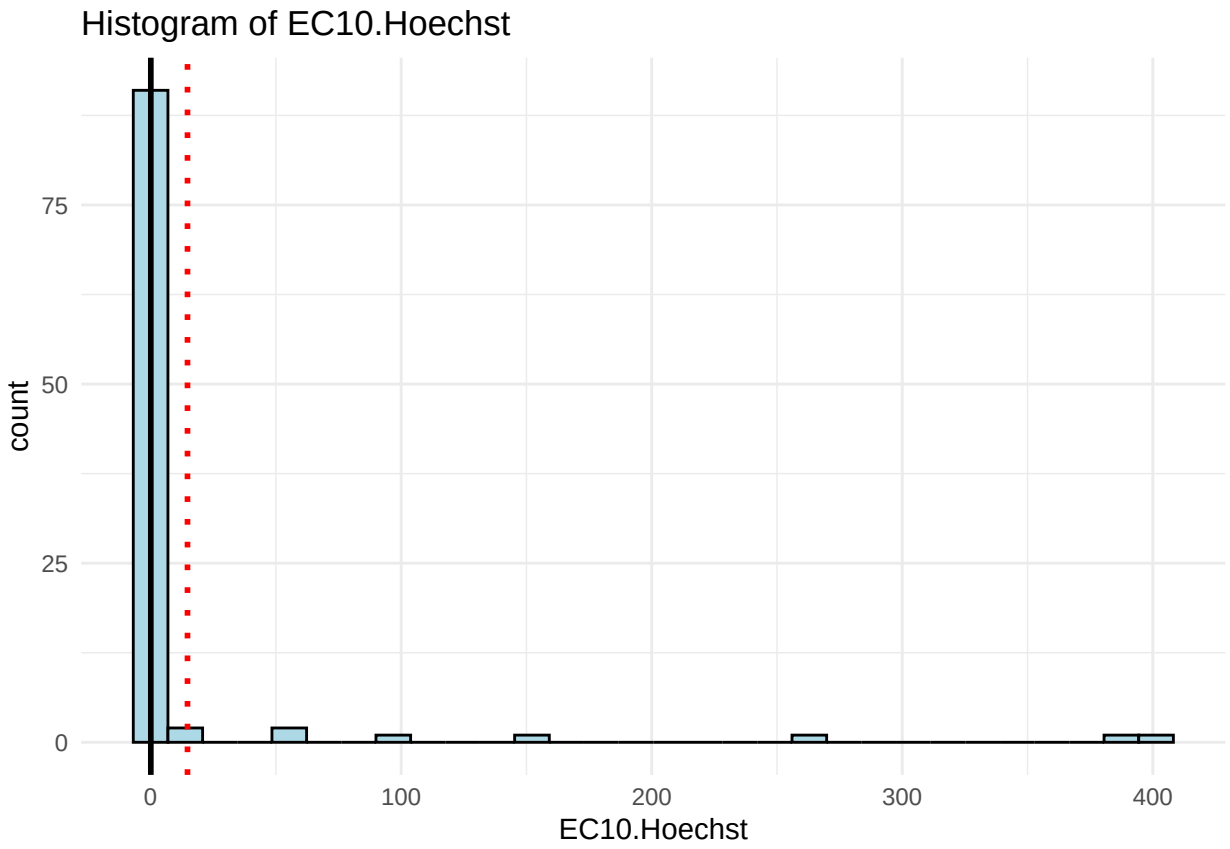

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



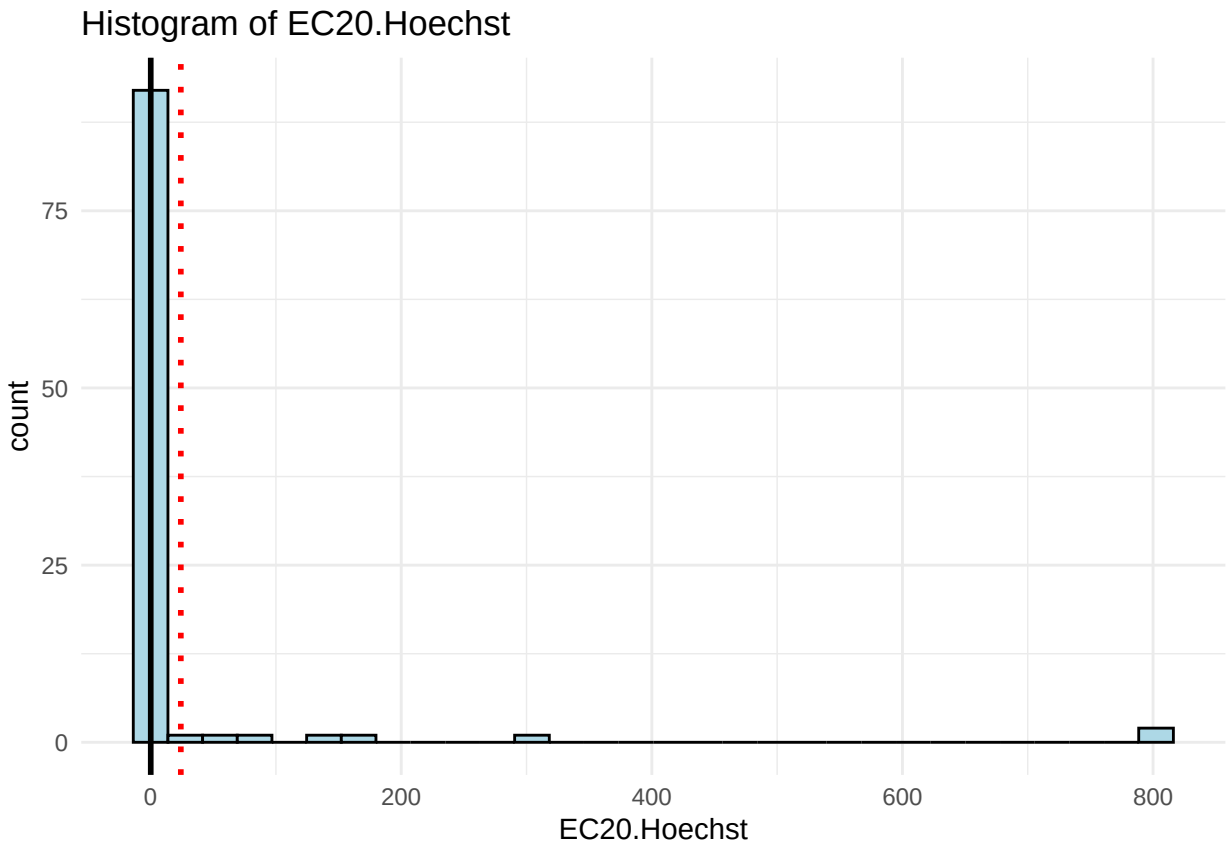

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



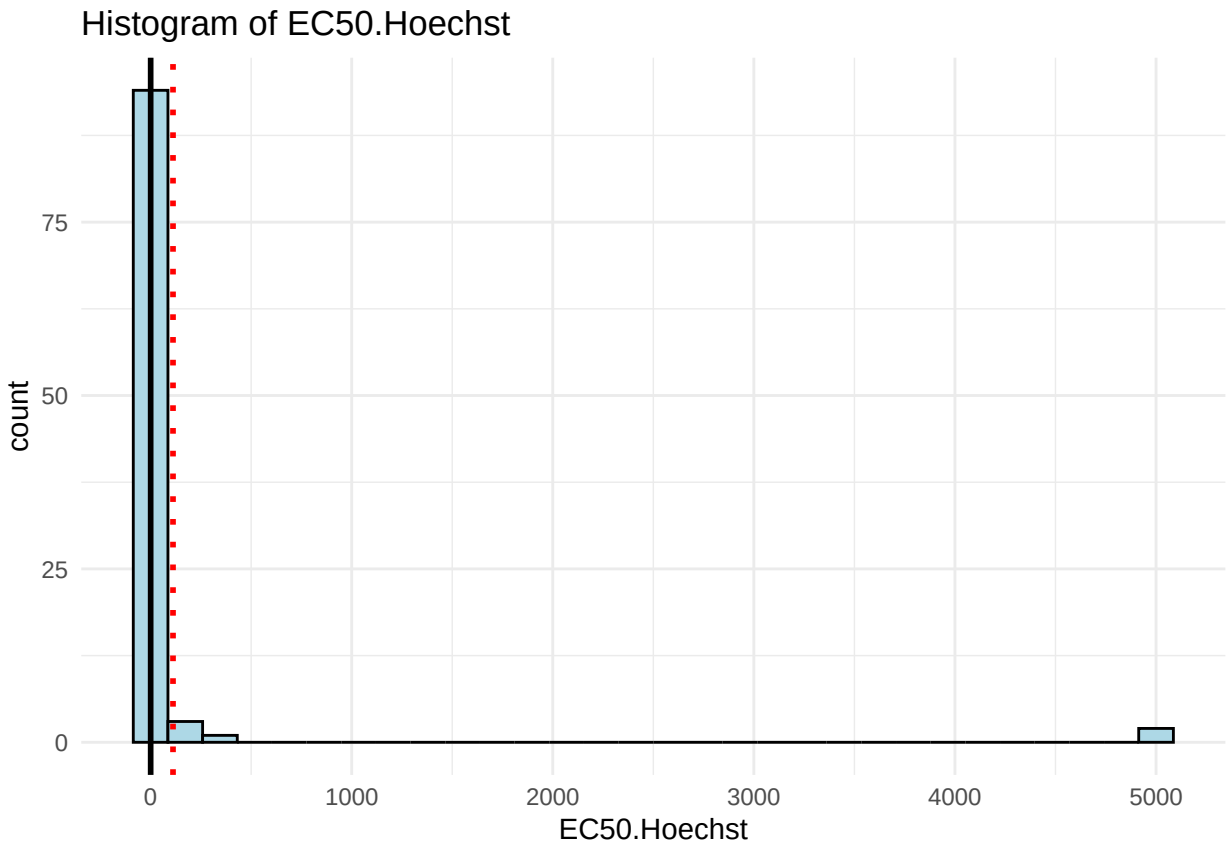

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



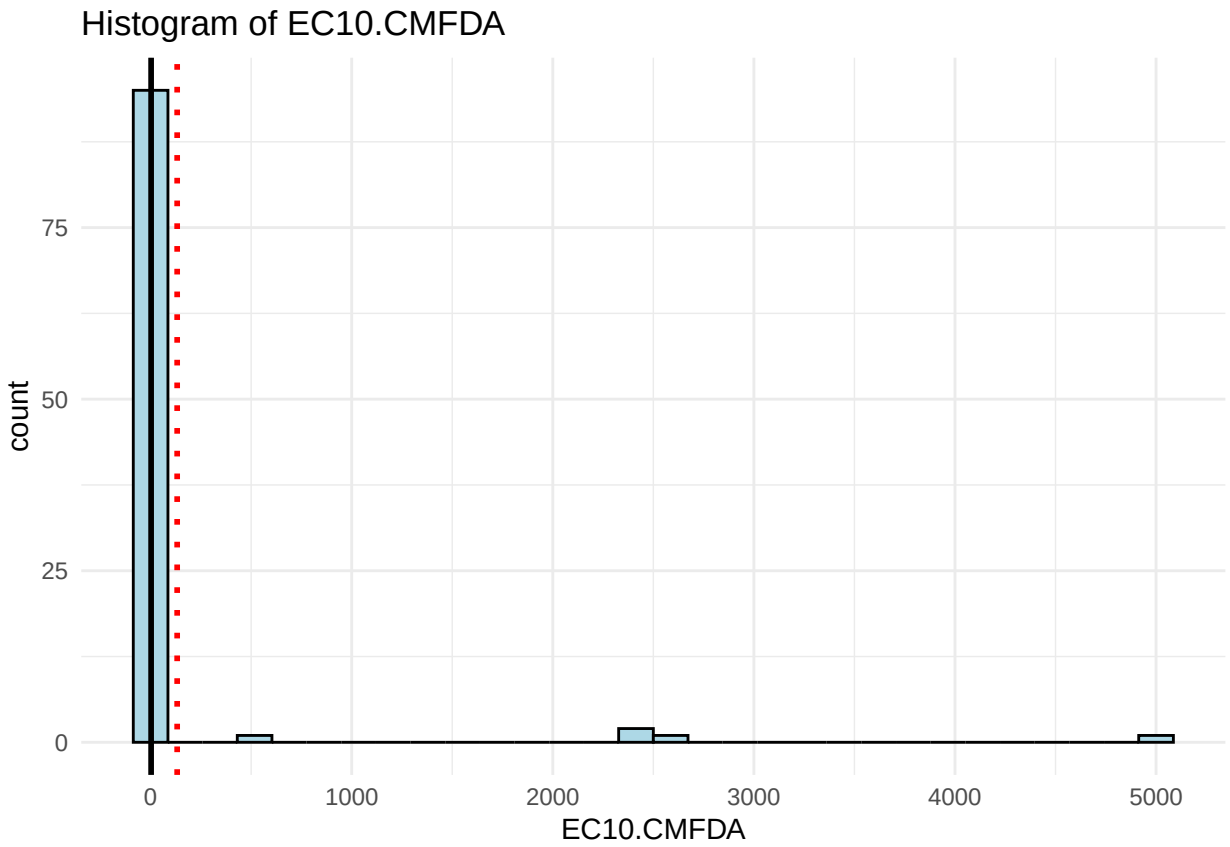

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



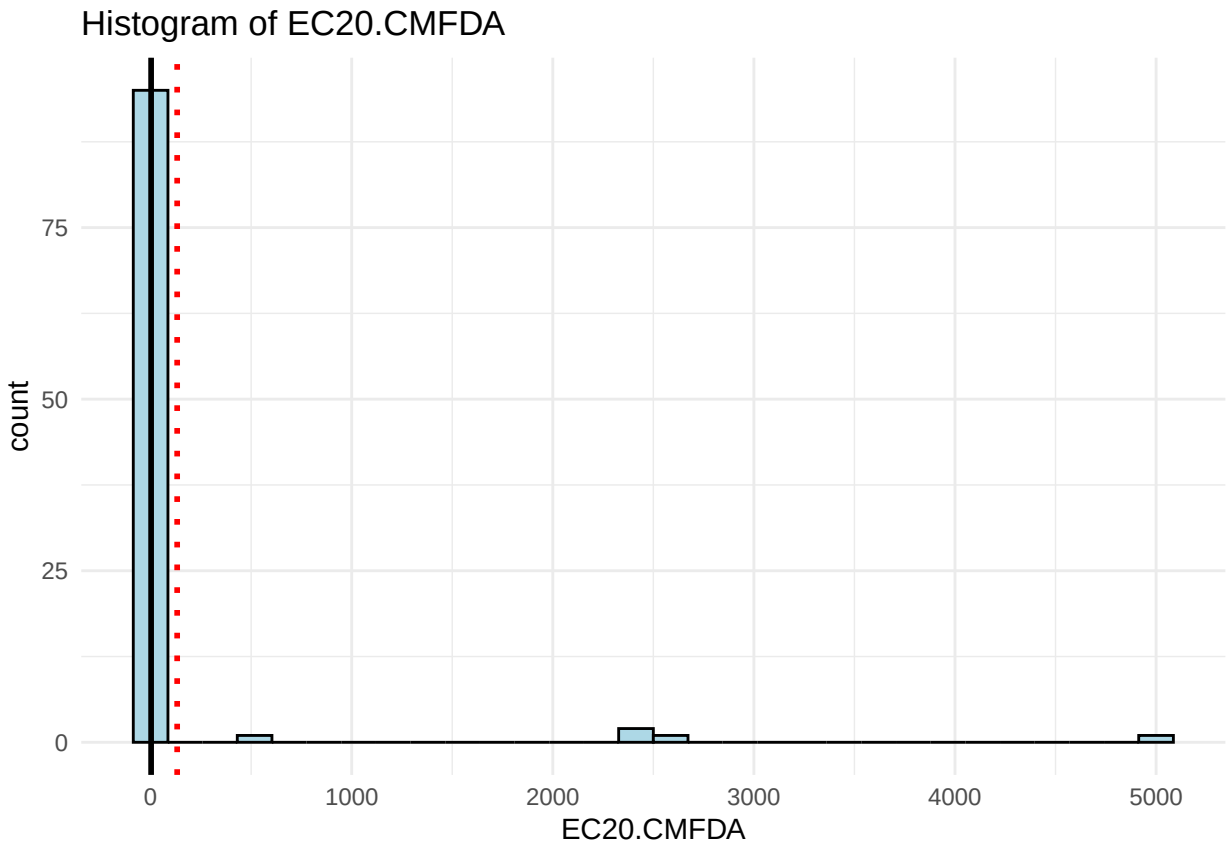

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



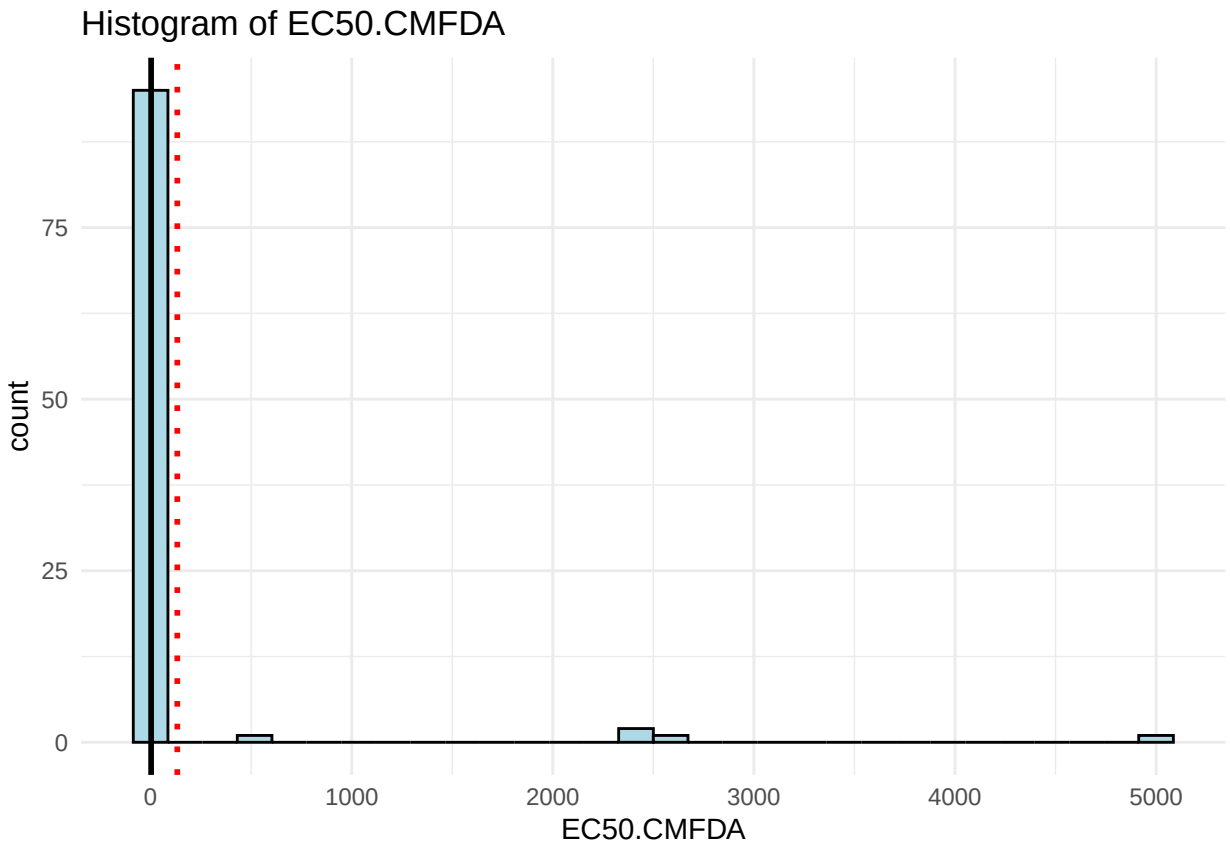

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



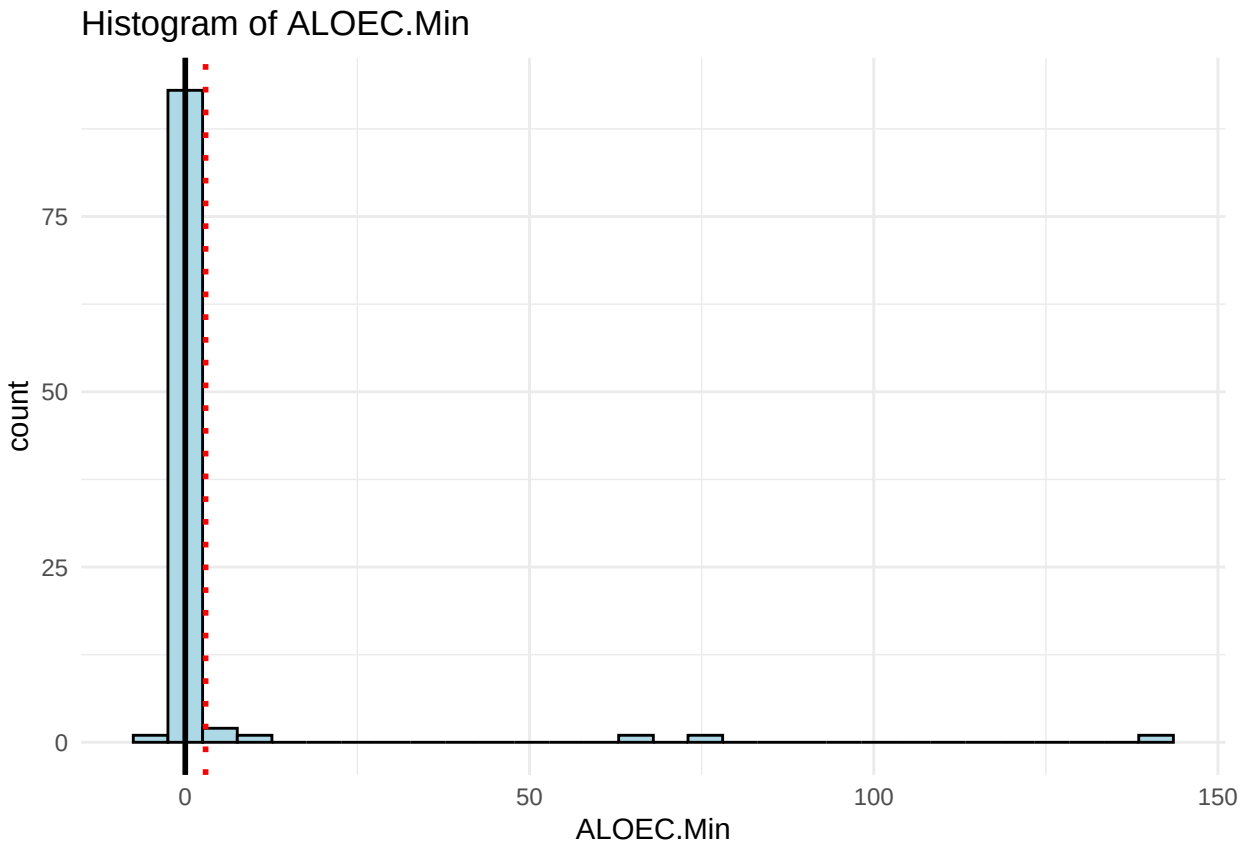

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



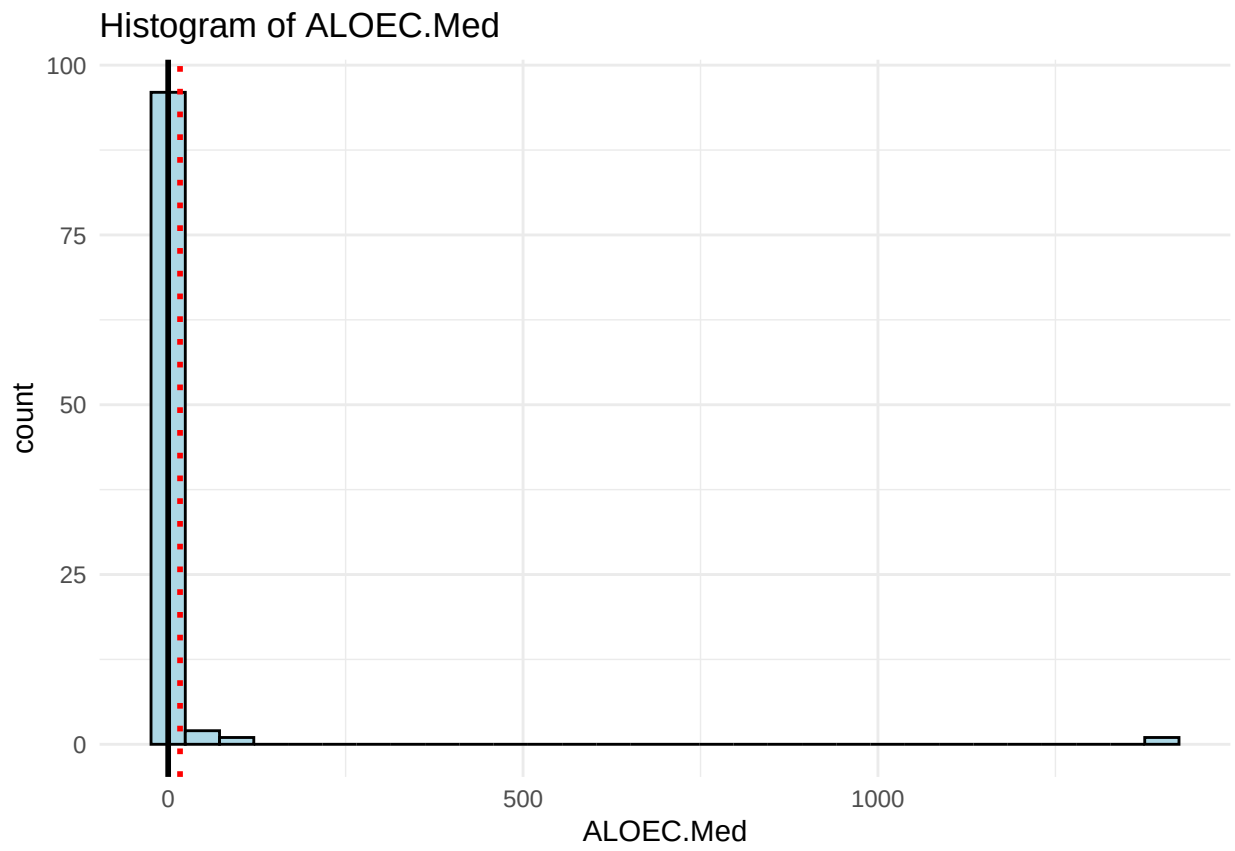

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



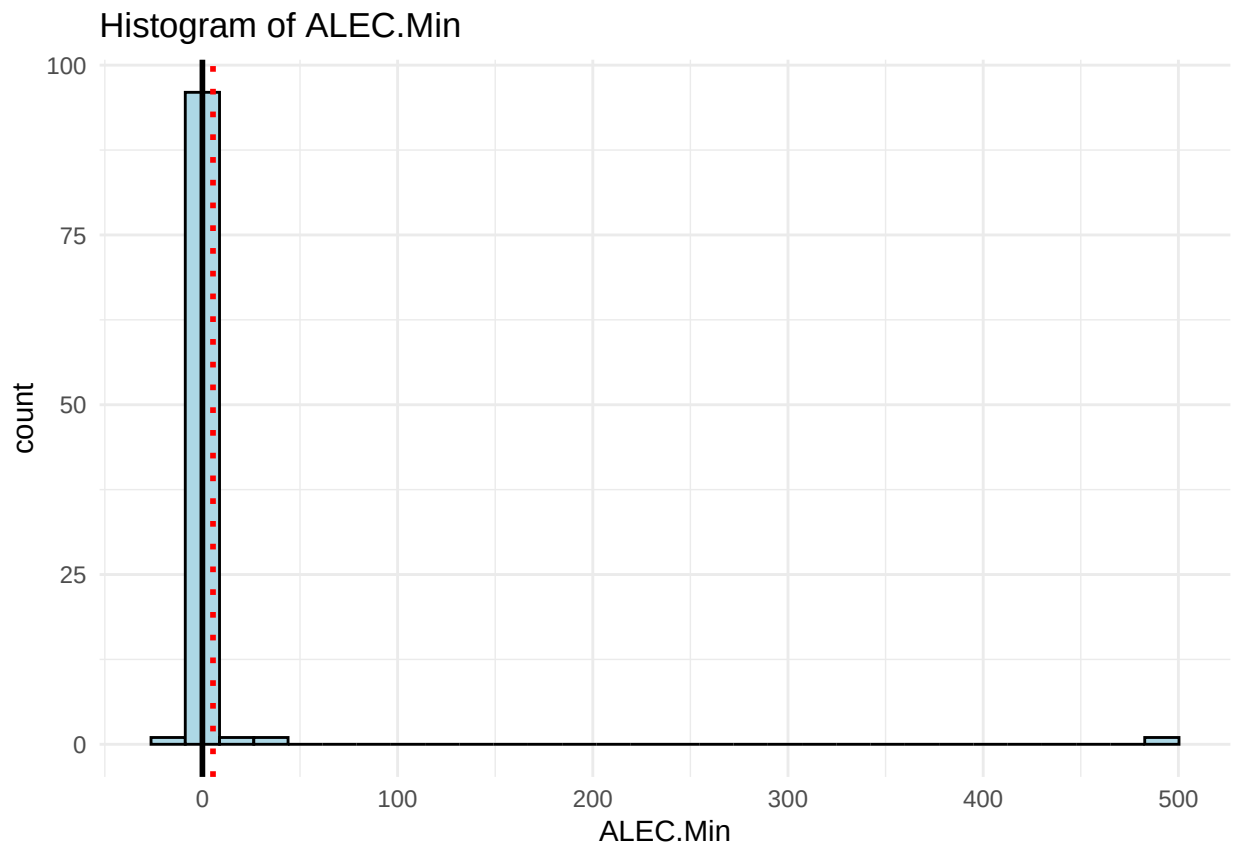

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



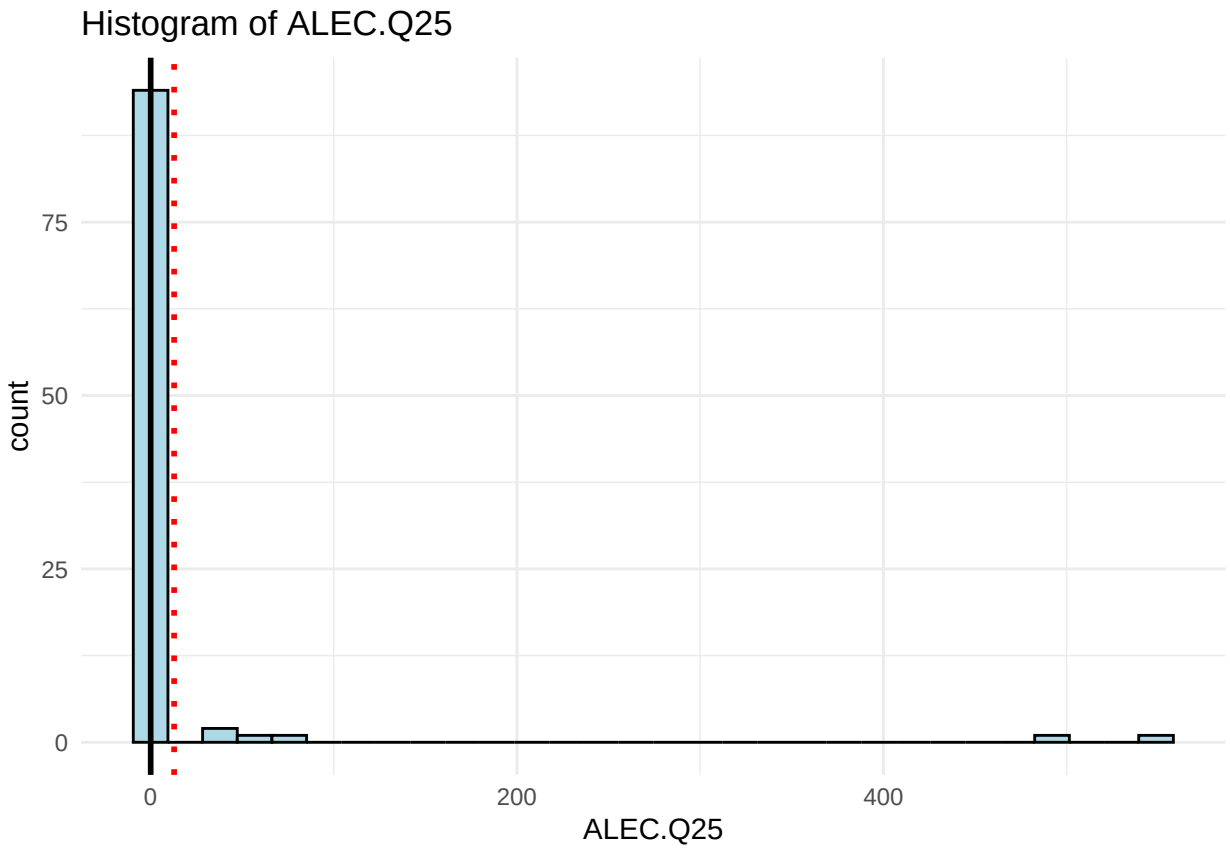

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



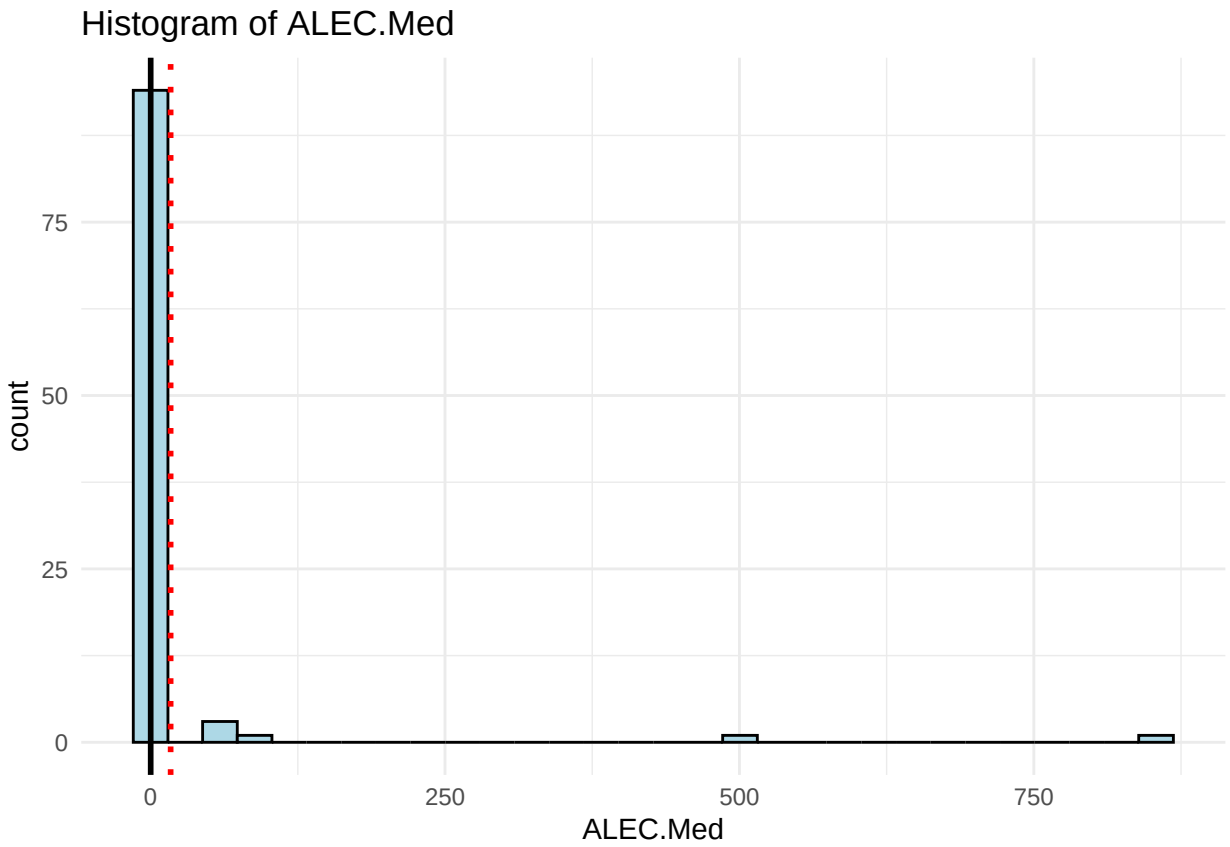

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



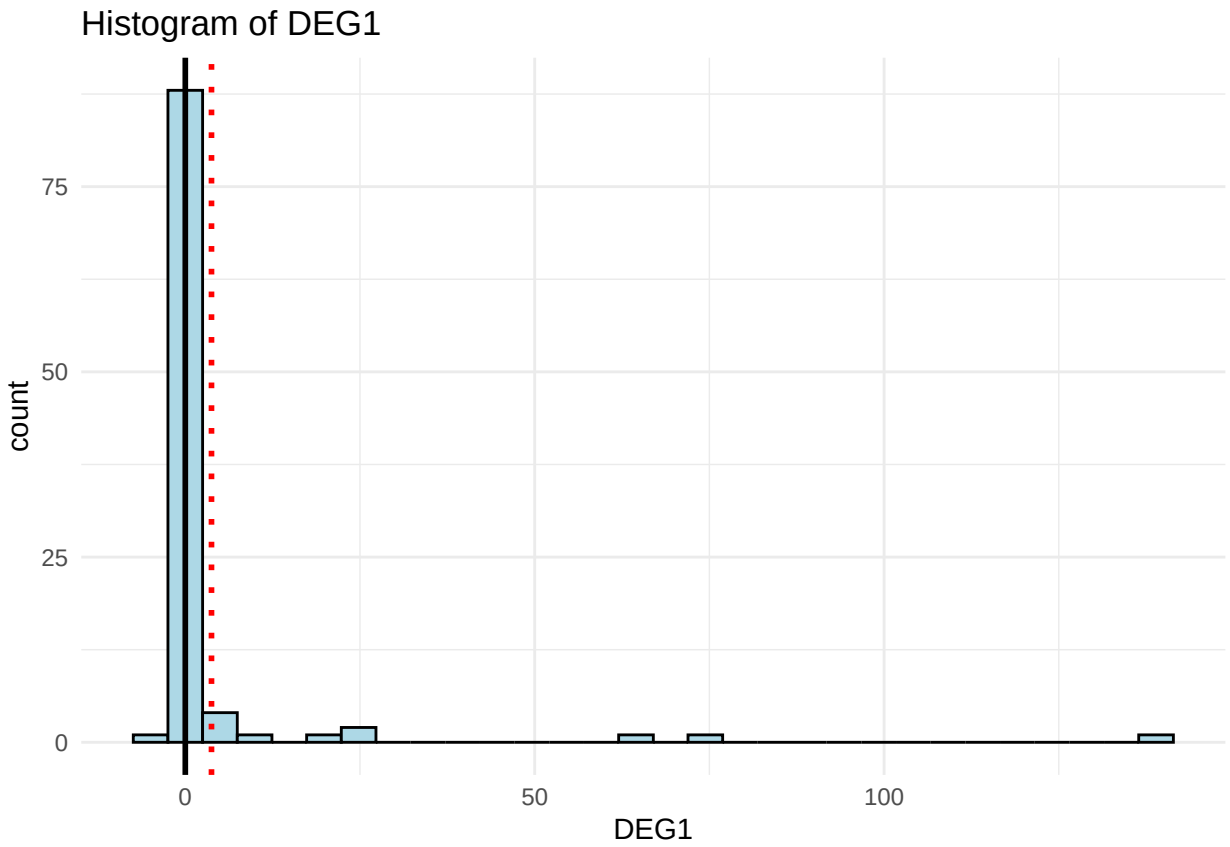

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



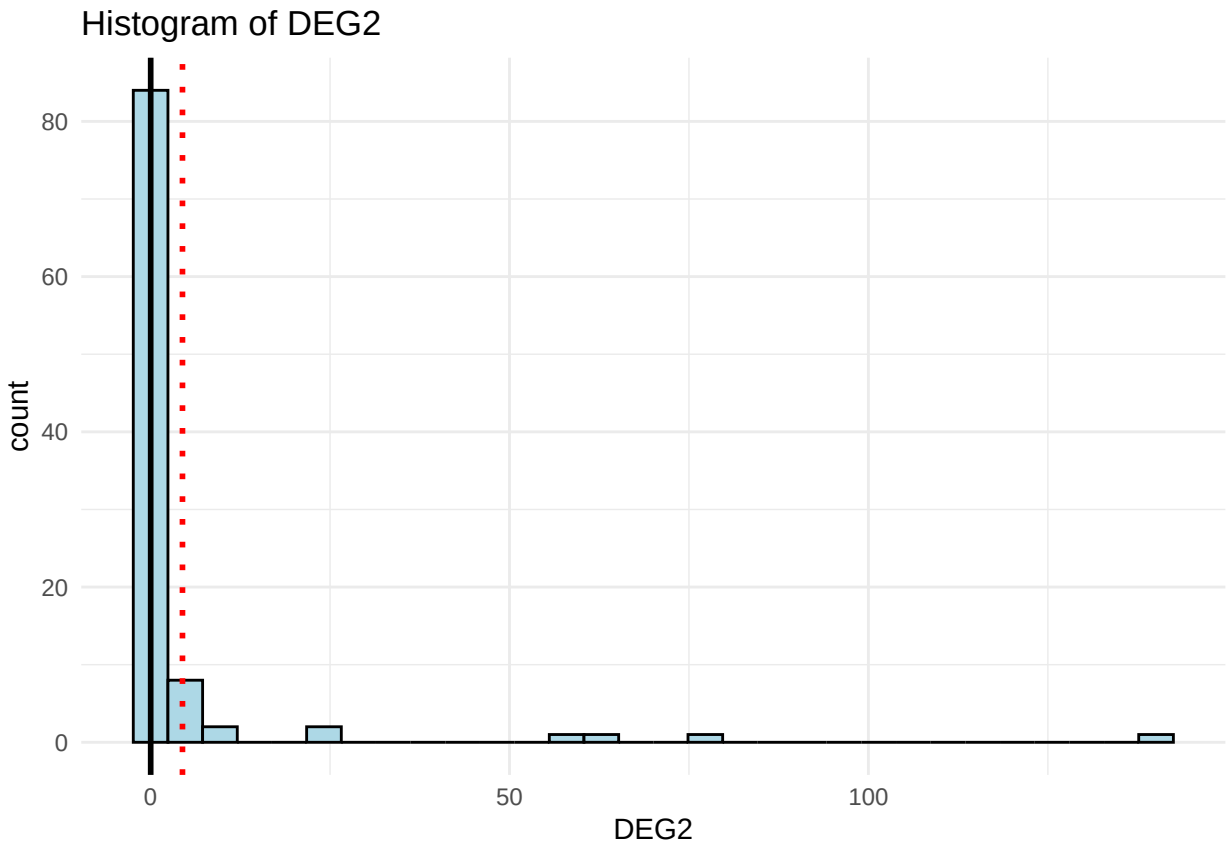

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



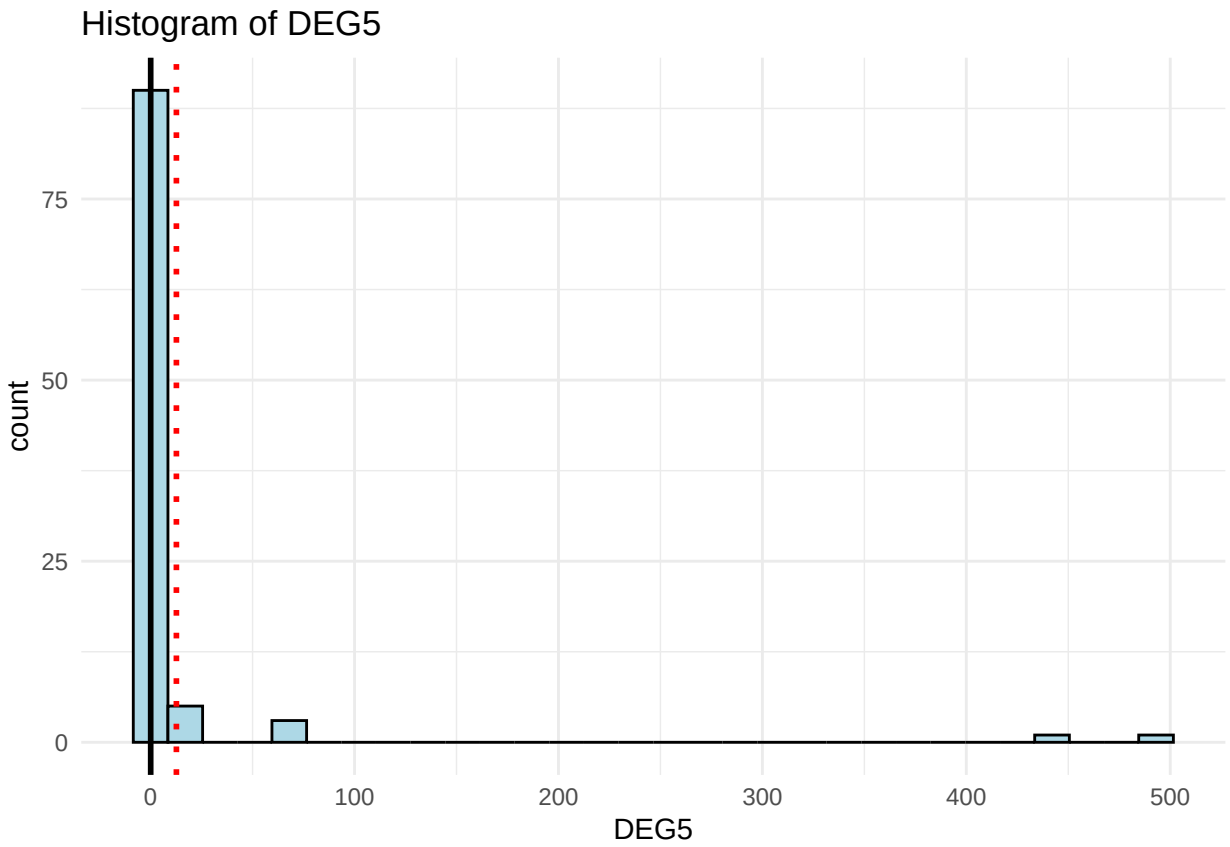

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



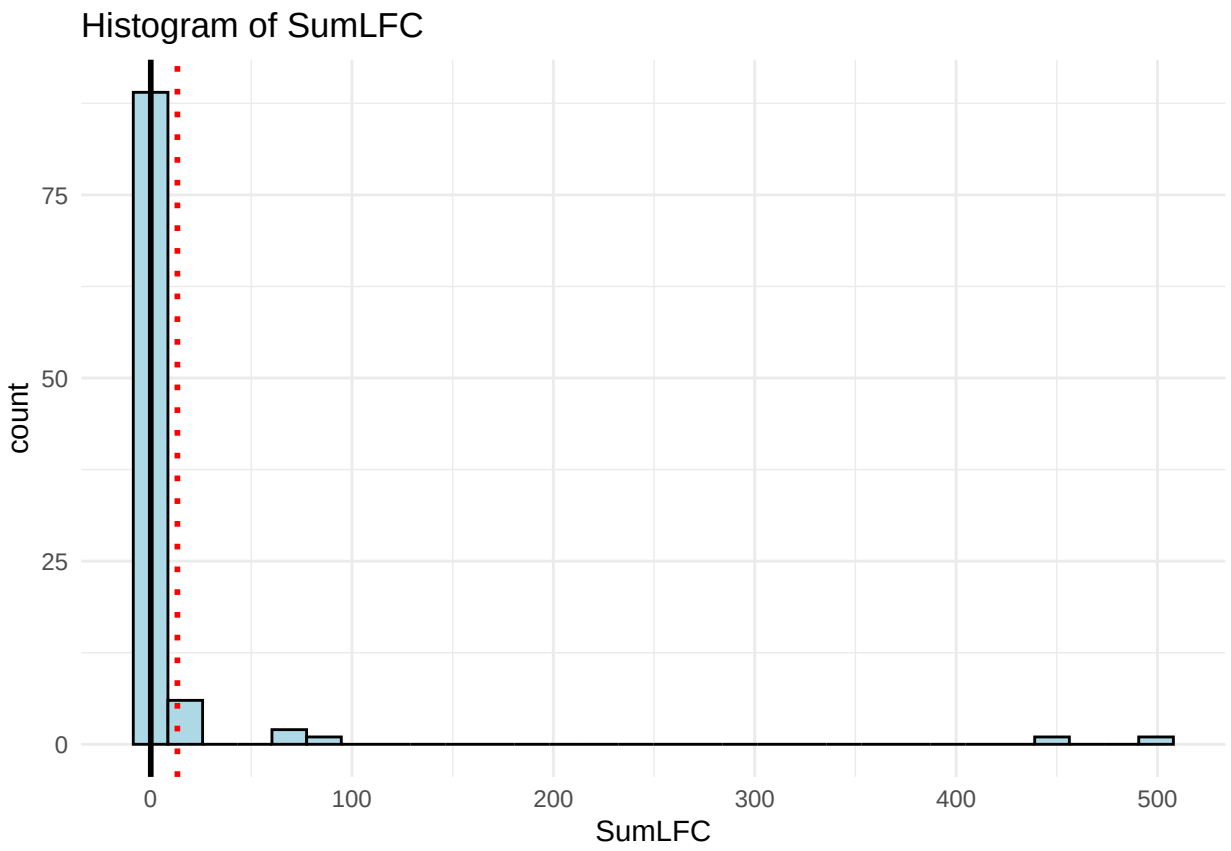

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




## Bivariate analysis

```r
cor_matrix4n <- cor(X_normalized)
# Find pairs of variables with correlation > 0.9
high_cor4n <- which(abs(cor_matrix4n) > 0.9, arr.ind = TRUE )
high_cor4n <- high_cor4n[high_cor4n[,1]!=high_cor4n[,2],]

# Get the variable names and their correlation values
high_cor_pairs4n <- data.frame(
  var1 = rownames(cor_matrix4n)[high_cor4n[, 1]],
  var2 = colnames(cor_matrix4n)[high_cor4n[, 2]],
  correlation = cor_matrix4n[high_cor4n]
)

# Remove duplicate pairs (since correlation matrix is symmetric)
high_cor_pairs4n <- high_cor_pairs4n[!duplicated(t(apply(high_cor_pairs4n, 1, sort))), ]

# Sort the pairs by correlation in descending order
high_cor_pairs4n <- high_cor_pairs4n[order(-abs(high_cor_pairs4n$correlation)), ]

# Display the sorted pairs of variables with high correlation
print(high_cor_pairs4n)
```

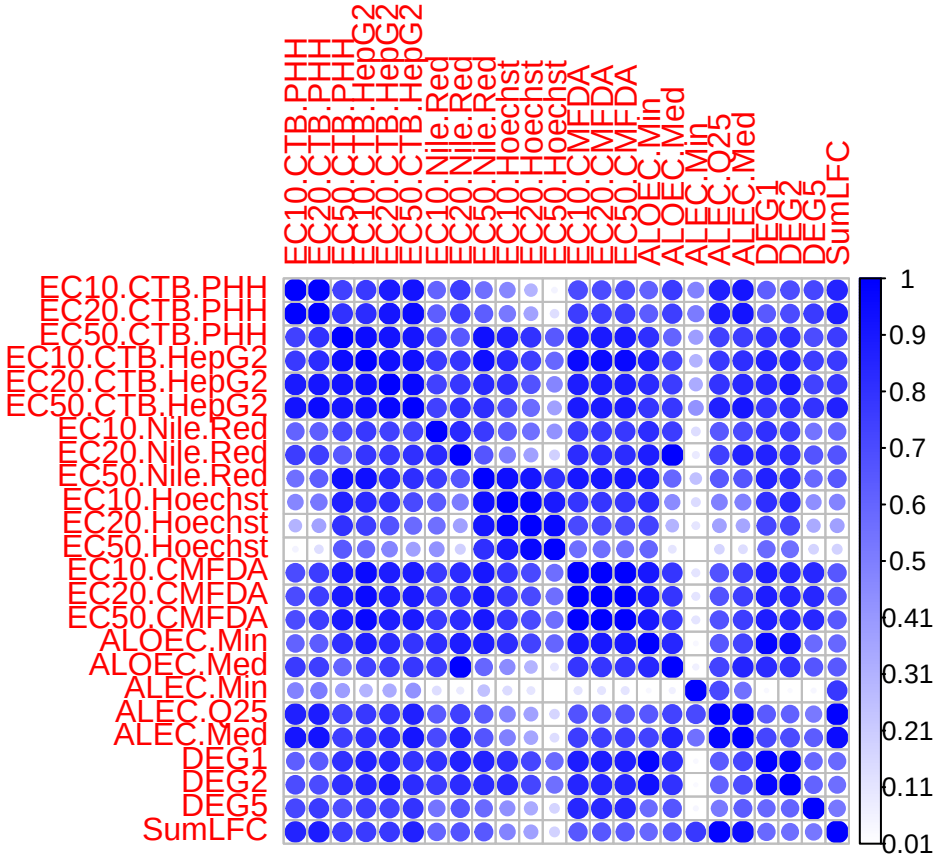
```
##              var1           var2 correlation
## 45     EC20.CMFDA     EC10.CMFDA   1.0000000
## 51     EC50.CMFDA     EC20.CMFDA   0.9999723
## 46     EC50.CMFDA     EC10.CMFDA   0.9999713
## 1    EC20.CTB.PHH   EC10.CTB.PHH   0.9918579
## 29      ALOEC.Med  EC20.Nile.Red   0.9914217
## 65         SumLFC       ALEC.Q25   0.9893673
## 38   EC20.Hoechst   EC10.Hoechst   0.9774411
## 61           DEG1      ALOEC.Min   0.9763288
## 64       ALEC.Med       ALEC.Q25   0.9763105
## 72           DEG2           DEG1   0.9718249
## 22 EC50.CTB.HepG2 EC20.CTB.HepG2   0.9697559
## 41   EC50.Hoechst   EC20.Hoechst   0.9683496
## 18     EC50.CMFDA EC10.CTB.HepG2   0.9603678
## 17     EC20.CMFDA EC10.CTB.HepG2   0.9603066
## 16     EC10.CMFDA EC10.CTB.HepG2   0.9603056
## 6  EC50.CTB.HepG2   EC20.CTB.PHH   0.9559083
## 8  EC10.CTB.HepG2   EC50.CTB.PHH   0.9482806
## 70         SumLFC       ALEC.Med   0.9477058
## 32   EC10.Hoechst  EC50.Nile.Red   0.9430167
## 14 EC50.CTB.HepG2 EC10.CTB.HepG2   0.9393316
## 15  EC50.Nile.Red EC10.CTB.HepG2   0.9366937
## 13 EC20.CTB.HepG2 EC10.CTB.HepG2   0.9365660
## 11  EC50.Nile.Red   EC50.CTB.PHH   0.9337509
## 62           DEG2      ALOEC.Min   0.9334396
## 2  EC50.CTB.HepG2   EC10.CTB.PHH   0.9248035
## 9  EC20.CTB.HepG2   EC50.CTB.PHH   0.9246008
## 7        ALEC.Med   EC20.CTB.PHH   0.9222193
## 33   EC20.Hoechst  EC50.Nile.Red   0.9154158
## 5  EC20.CTB.HepG2   EC20.CTB.PHH   0.9149542
## 3        ALEC.Med   EC10.CTB.PHH   0.9136802
## 10 EC50.CTB.HepG2   EC50.CTB.PHH   0.9130623
## 28       ALEC.Med EC50.CTB.HepG2   0.9112898
## 36     EC50.CMFDA  EC50.Nile.Red   0.9106312
## 35     EC20.CMFDA  EC50.Nile.Red   0.9102698
## 34     EC10.CMFDA  EC50.Nile.Red   0.9102636
## 52      ALOEC.Min     EC20.CMFDA   0.9032864
## 47      ALOEC.Min     EC10.CMFDA   0.9032862
## 57      ALOEC.Min     EC50.CMFDA   0.9032669
```

```
print(length(high_cor_pairs4n[,1]))
```

```
## [1] 38
```

## corrplot

```
corrplot(cor_matrix4n, is.corr=FALSE, col=colorRampPalette(c("white", "blue"))(200) )
```




## scatterplots

### Scatterplot between tartet and explanatory variables

```
# Create pairwise scatter plots colored by 'Toxicity'
#ggpairs(cbind(X_normalized, Toxicity = y, aes(fill = Toxicity, alpha = 0.7), progress = FALSE) +
 # theme_bw() +
  #labs(title = "Pairwise Scatter Plots for Predictor Variables (Colored by Toxicity)")
```

### Scatterplots between Toxicity and other variables

```
cols =colnames(X_normalized)
#cols = df_aucs$variable # in the order of sorted auc
plot_list = list()
for (i in 2:ncol(X_normalized)) {
  colname <- cols[i]
  p <- ggplot(X_normalized, aes(x = .data[[colname]], y)) +
    geom_point() +
    labs(x = colname, y = "Toxicity") +
    theme_minimal()
  #print(p)
  # Add each plot to the list
  plot_list[[i-1]] <- p
```

```
}

# Arrange the plots in a grid (choose the number of rows/columns as needed)
grid.arrange(grobs = plot_list, ncol = 2)
```