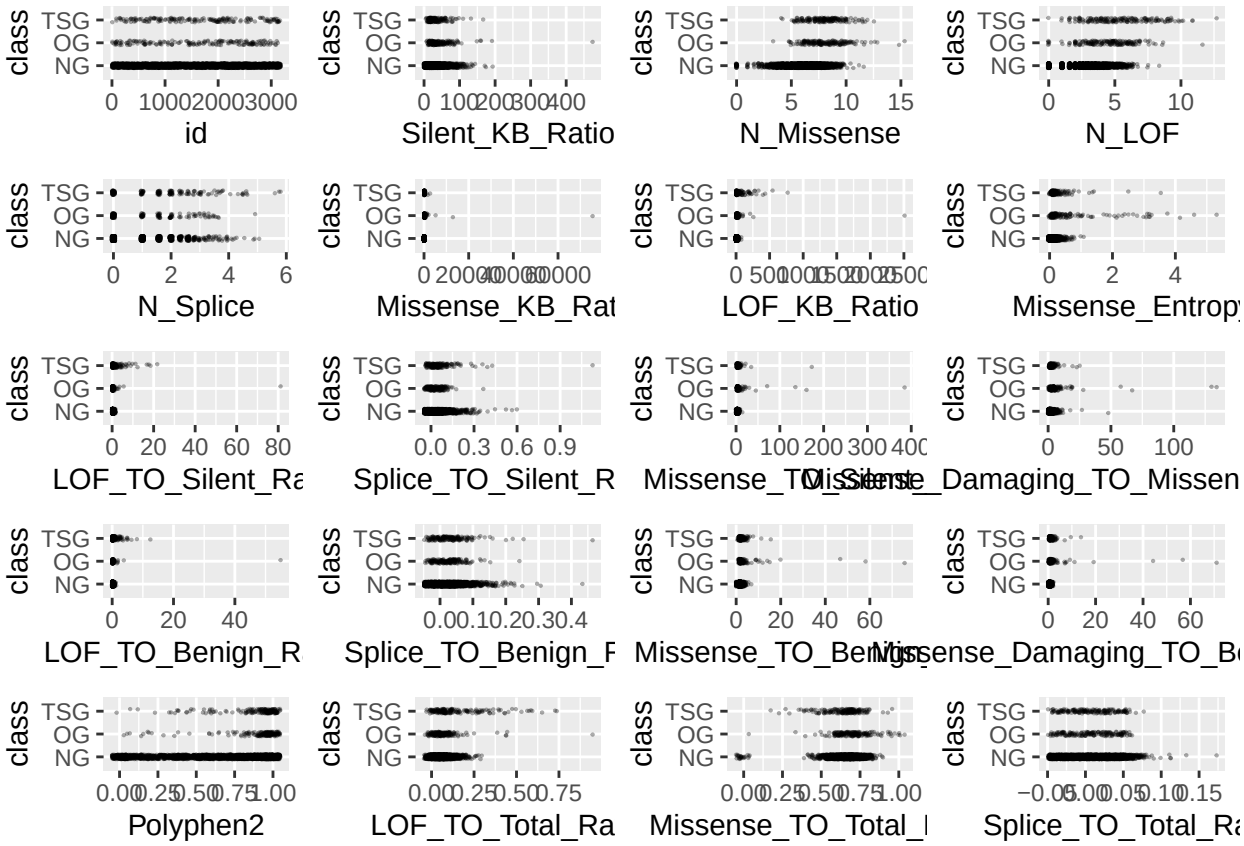
Model #5: KNN

Ethan Allavarpu (UID: 405287603)

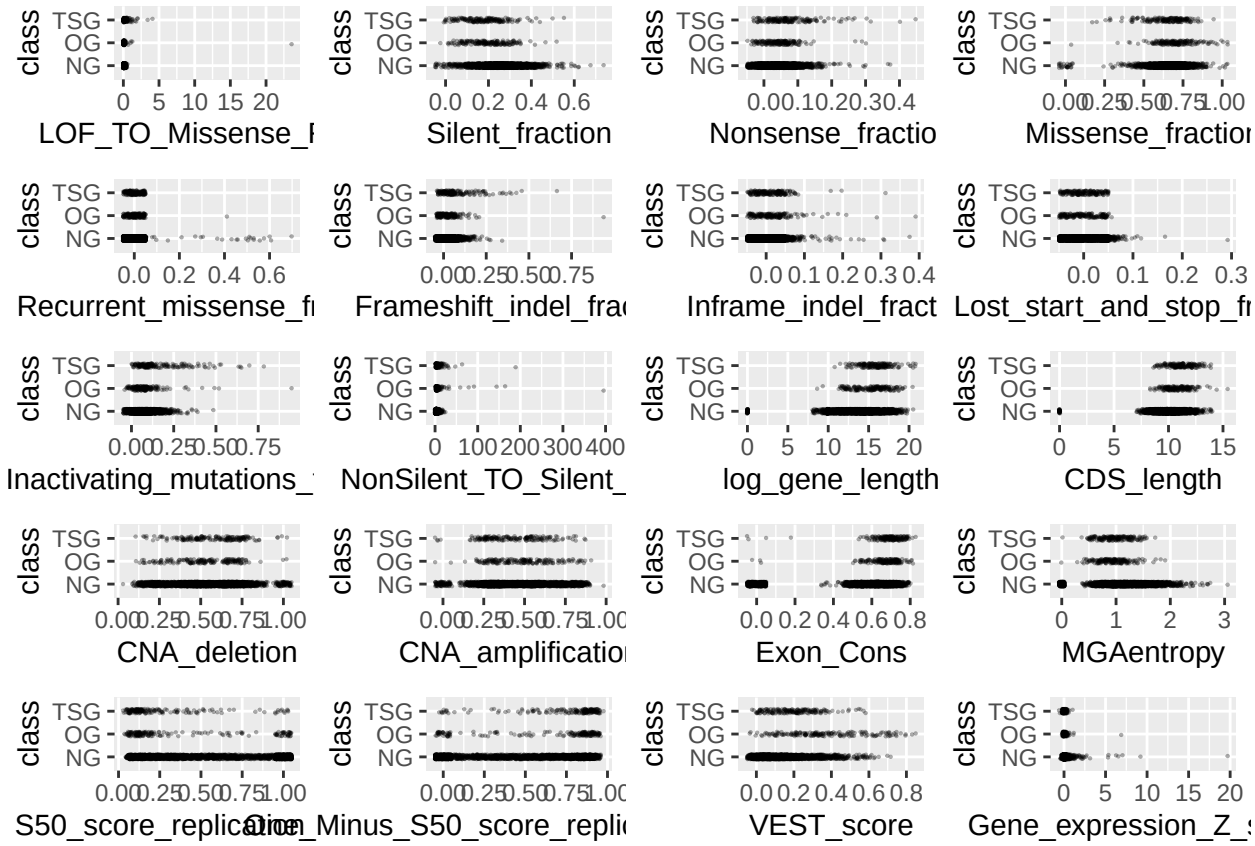
10/28/2020

Transforming and Cleaning the Data

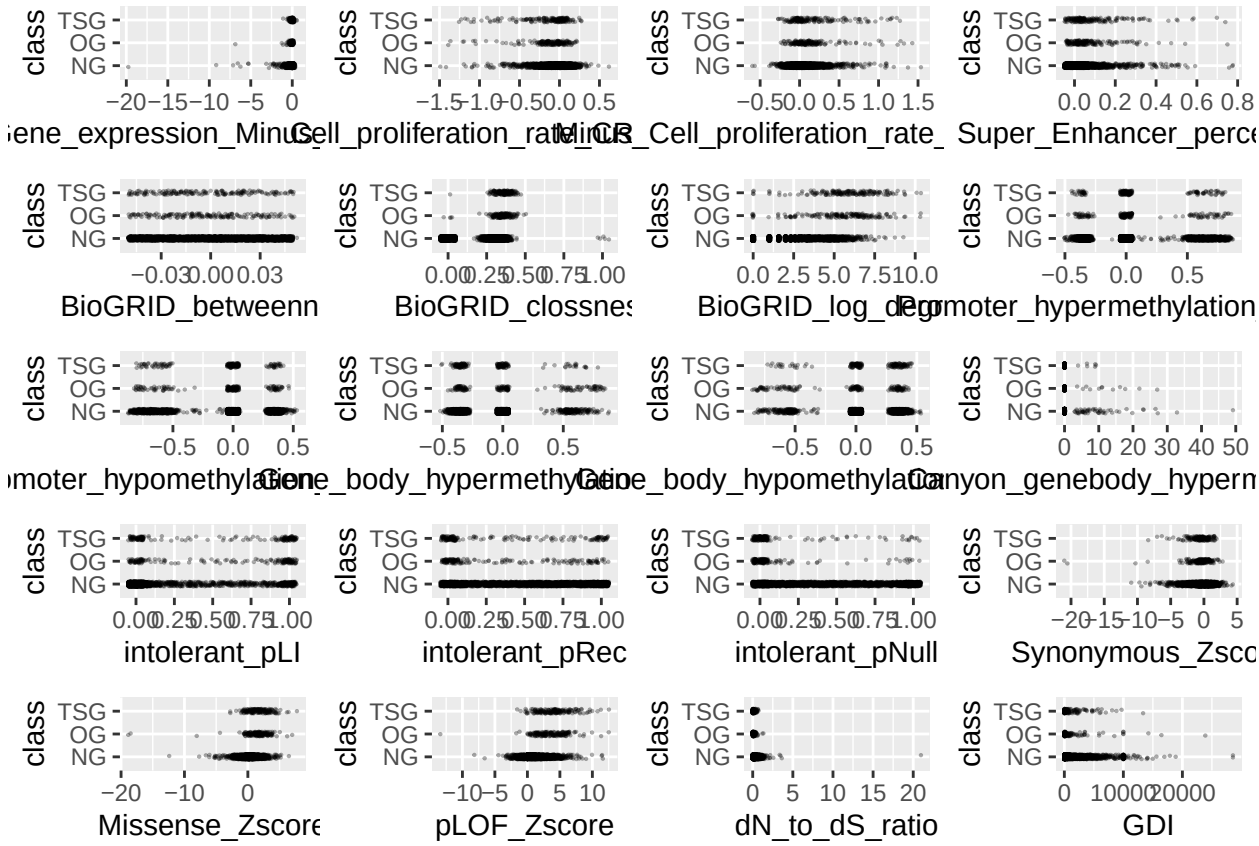
```
training <- read.csv("training.csv", stringsAsFactors = TRUE)
training$class <- factor(training$class)
levels(training$class) <- c("NG", "OG", "TSG")
outlier <- function(data) {
  low <- mean(data) - 3 * sd(data)
  high <- mean(data) + 3 * sd(data)
  which(data < low | data > high)
}
library(ggplot2)
scatter <- function(var) {
  ggplot(training, aes_string(var, "class")) +
    geom_jitter(width = 0.05, height = 0.1, size = 0.1,
                colour = rgb(0, 0, 0, alpha = 1 / 3))
}
scat_plot <- lapply(names(training)[-99], scatter)
library(gridExtra)
grid.arrange(grobs = scat_plot[1:20], ncol = 4)
```



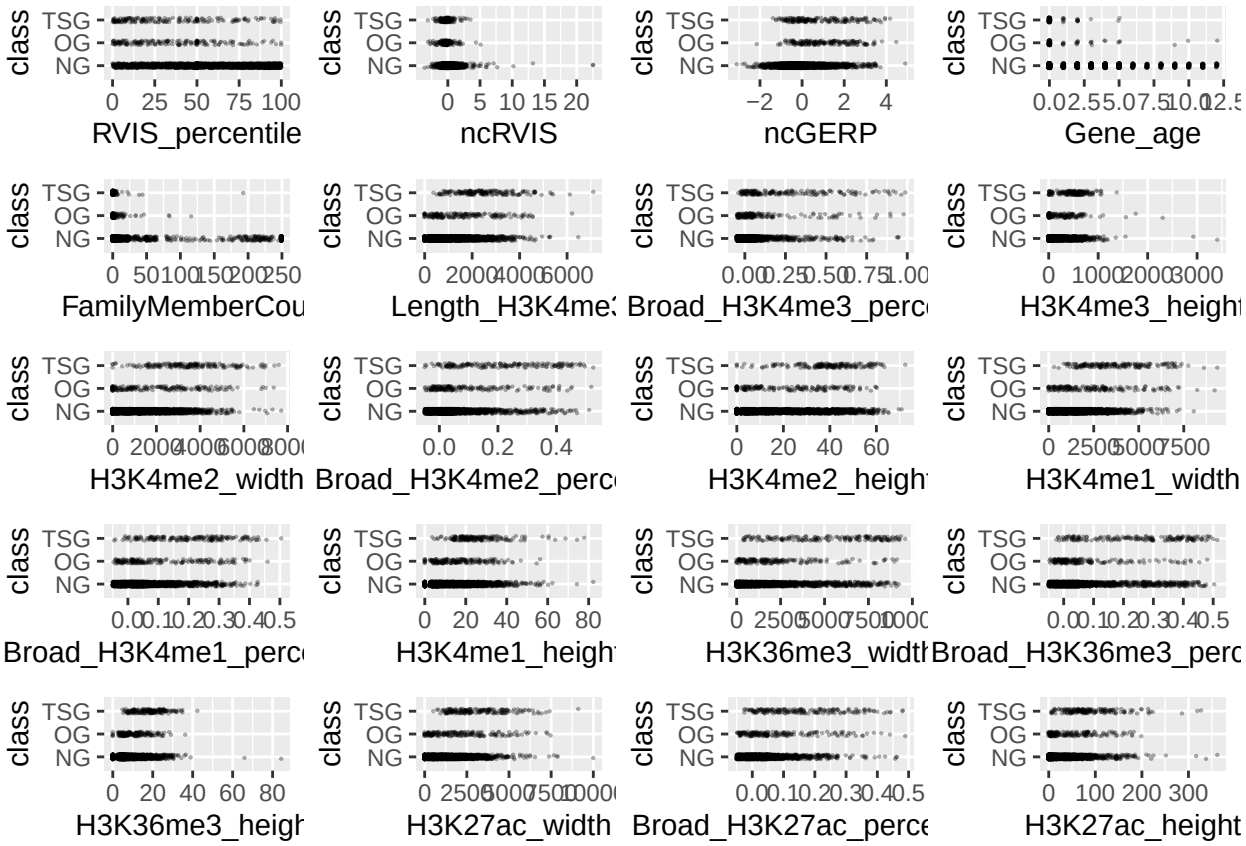

```
grid.arrange(grobs = scat_plot[21:40], ncol = 4)
```



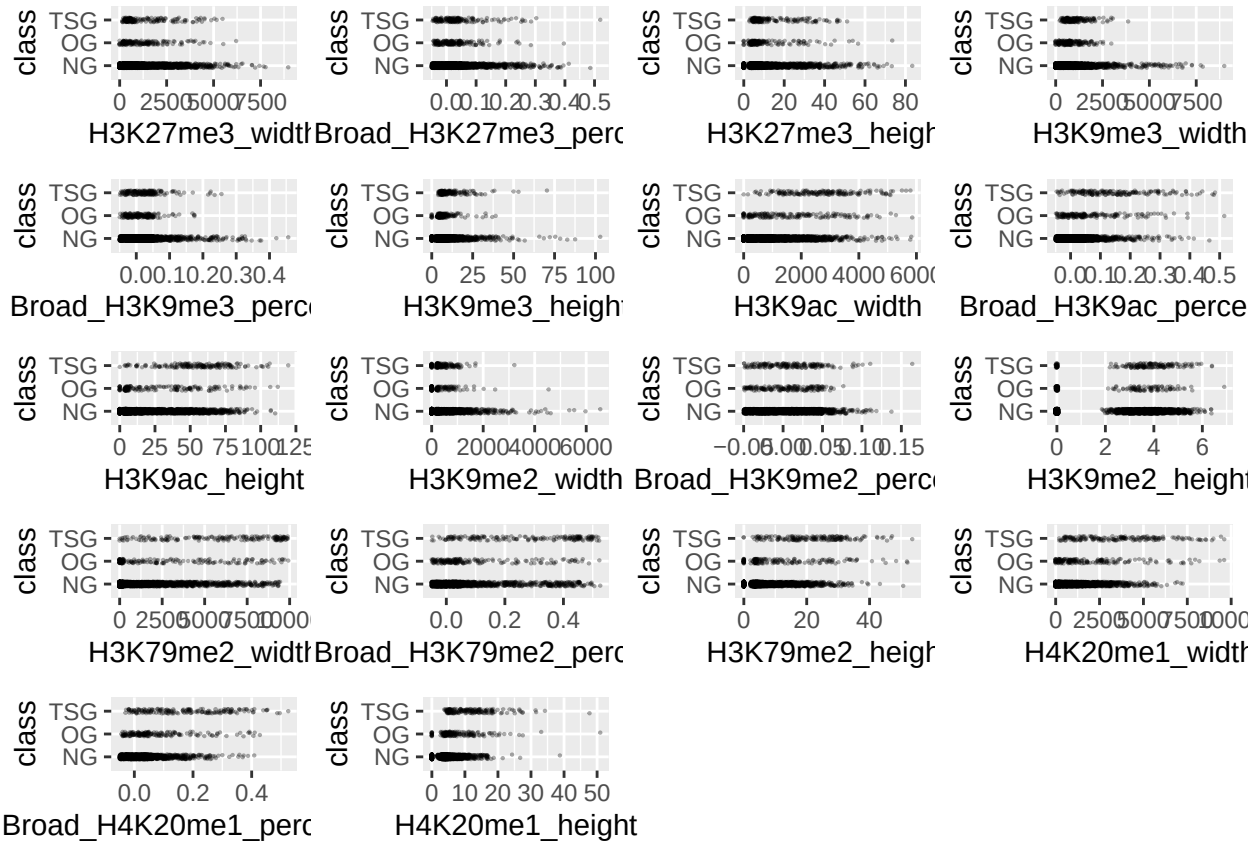

```
grid.arrange(grobs = scat_plot[41:60], ncol = 4)
```




```
grid.arrange(grobs = scat_plot[61:80], ncol = 4)
```




```
grid.arrange(grobs = scat_plot[81:98], ncol = 4)
```




```
outlier_index <- sort(table(unlist(lapply(training[,-99], outlier))), decreasing = TRUE)
outlier_index[1:100]
```

```
##
##  915 1280 2918  517 1914 2182 3052 1173 2215 3049  259  740 1749 1979 2998  417
##   24   24   24   22   22   22   20   19   19   19   18   18   18   18   18   17
##  441  806 2297  422  635 1258 1570 2278 2518 2729   80  150 2694  169  276  341
##   17   17   17   16   16   16   16   16   16   16   15   15   15   14   14   14
## 1528 1556 1726 1809 1911 1955 2071 2624 2641 3120 3142   73  277  364  751 1244
##   14   14   14   14   14   14   14   14   14   14   14   13   13   13   13   13
## 1330 2329 2787  343 1138 1171 1188 1372 1460 2031 2251 2968 2983 3166  352  634
##   13   13   13   12   12   12   12   12   12   12   12   12   12   12   11   11
##  907  923 1096 1858 2636  588 1137 1317 1463 1561 1740 1991 2487 2540 2555 2621
##   11   11   11   11   11   10   10   10   10   10   10   10   10   10   10   10
## 2815 3029   74  144  657  789  857 1267 1610 1932 2022 2093 2142 2534 2666 2721
##   10   10    9    9    9    9    9    9    9    9    9    9    9    9    9    9
## 2848 2900 3027  155
##    9    9    9    8
```

```
training <- training[-as.numeric(names(outlier_index)[1:50]),]
training <- training[-which(training$Missense_TO_Silent_Ratio > 100), ]
training <- training[-which(training$Missense_KB_Ratio > 2000), ]
training <- training[-which(training$LOF_TO_Silent_Ratio > 5), ]
training <- training[-which(training$Gene_expression_Z_score > 4), ]
training <- training[-which(training$dN_to_dS_ratio > 5),]
training <- training[-which(training$Silent_KB_Ratio > 200), ]
training <- training[-which(training$Lost_start_and_stop_fraction > 0.2),]
training <- training[-which(training$Synonymous_Zscore < -15), ]
```

```
numeric_training <- training[,-99]

n_zeroes <- rep(NA, nrow(numeric_training))

for(i in seq_len(nrow(numeric_training))){
  row_i_zeroes <- 0
  for(j in seq_len(ncol(numeric_training))){
    if(round(numeric_training[i,j], digits = 5) == 0){
      row_i_zeroes <- row_i_zeroes + 1
    }
  }
  n_zeroes[i] <- row_i_zeroes
}
n_zeroes
```

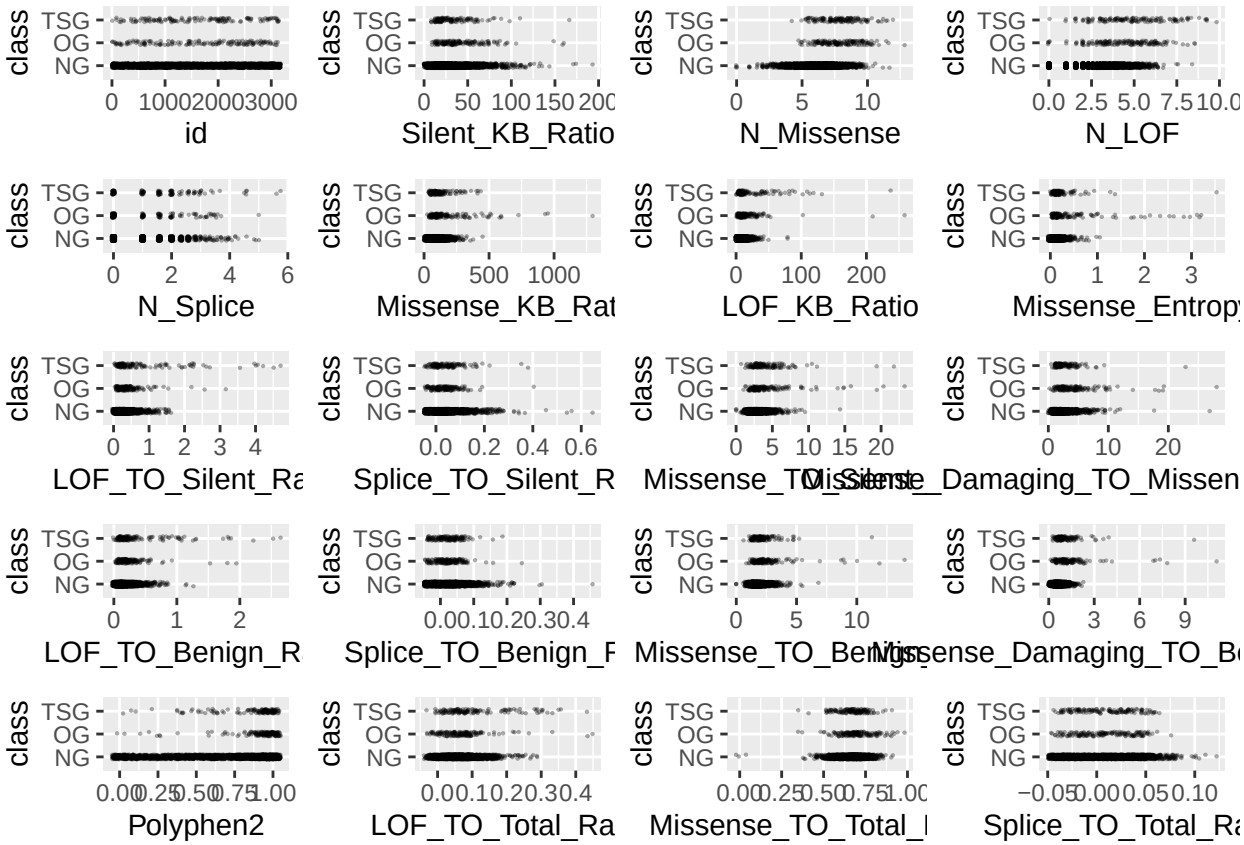
```
##    [1]  9 15 23 11 16 22 12  7 26  9 21 27 34 15 25 13  9 18 62 15  9 22 16 13
##   [25] 16 14 10 28 17 27 20 22 15 13 10 24 10 15 33 81 56 15 23 13 17 20 21 33
##   [49] 18 24 11 11 15 18 17 31 16 23 15 30 15 19  8 26 28 14 22 17 31 47 14 20
##   [73] 29 14 22 20 28 26 33 11  9 27  6  9  6 30 14 20 26 10 16 28 20 27 11 16
##   [97] 33 13 31 61 14 15 30 27 77 11 10 14 35 21 21 12  9 28  6 10 12 25 10 21
##  [121] 13 12 27 29 17 47 48 21 15 36 30 21 85 27 25 34 28 52 24 14 11 29 35 12
##  [145] 11 65 41 12 35 16 15 32 12 12 19 31  7 26 12 16 25 13 11  8 43 15 63 31
##  [169] 21 15 17 17 17  9 18 10 26 10 16 25 14 14 15 41 15 13 25 11 15 35 18 12
##  [193] 21 17 45 26 29 71 15 20 14  9 23 22 18 10 20 13 21 12 23 21 14  9 26 23
##  [217] 16 29 23 16 13 25 13  8 27 17 19 26 16 30 18 36 34  8 22 27 24 19 26 31
##  [241] 21 31 26 22  9 19 17 16 15 10 20 10 27 33 13 34 19 13 16 14 11 33 37 16
##  [265] 14 21 11 11 16 17  9 12 17 14 14 24 25 19 45 21 15 25 10 26 38 22 38 24
##  [289] 26 22 25 12 30 29 29  7 18  9 22 14 56 14 16 18 20 11 31 46 33 23 25 11
##  [313] 29  7 22 27 13 29  9 13 11  9 28 16 13 12 21 15 39 10 39 47 35 26 22 13
##  [337] 31 16 30 20 24 26 28 35 13 30 15 21 24 11 11 33  9 14 11  8 67 28 33 15
##  [361] 57 62 12 23 11 22 38 27  9 19 21  8 26 19 12 11 27 19  7 21 25 25 10 26
##  [385] 31 15 20 29 22 24 28 26  9 11 13 15 35 13 24 10 14  9 14 14 14  9 17 57
##  [409]  8 22 17 18 13 30 14 28 14 36 26 11 27 44 31 58 28 30 23 28  8 22 21 12
##  [433] 29 32 28 12 14 14 27  9 14 16 18 12  5 22  8 17  8 26 12 11 16 14 23 16
##  [457] 35  7 18 22 17 12 14 18 28 10  7 22 17 19 38 30 25 22 12 38 37  7 26 38
##  [481] 14 37 25 13 11 27 11 10 20 20 27 13 13 13 14 14 28 24 21 24 38 21 31 88
##  [505] 12 28 28 11  5 36 18  3 27 33 16 16 16 20 28 24 14  6 13 25 15 13 17 15
##  [529] 34 23 27 32 64 37 26 14 23 12 11 21 14 30 43 27 14 15  9 35 26 17 18 13
##  [553] 18 32 12 53 13 11 15 10 16 22 18 16 10 17 81 10 13 11 10  9 23  9 16 58
##  [577] 12 23 24 25 11 13 49 14 33 22 18 15 19 12 19 24 30 18 13 16 14 11 11 20
##  [601] 28 26 21 35  9  9 27 15 25 11 41 10 26 36 27 17 35 11  9 18 20 22 33 74
##  [625] 15 16 17 16  9 17 13 19 36 10 35 13 13 25 31 10 12 30 20 62 21 12 13 15
##  [649] 14 12 20 13 30 12 14 11 28 32 17 19 12 29 30 11 18 21 29 15 23 14 30 11
##  [673] 17 45 11 14 13 18 19 31  6 29  8 23 31 21 19 31 10 15 15  9 13 14 13 16
##  [697]  9  7  8 15 11 17  8 18 14 23 21 10 41 13  9 10 13 12 17 13 22 41 42  7
##  [721] 42 22 31 12 21 11 18 33 24 16 10 41 32 22 18 55 19 21 25 16 21 30 28 35
##  [745] 17 26 15 10 20 29 43 30 34 27 18 26 20 10 62 25 11 19 16 26 29 28 11 39
##  [769] 16  6  8 44 48 31  9 34 26 43  6  7 10 21 30 35 51 20  8 27 15 14 15 18
##  [793] 12 14 45 27 57 14 14 22  6 54 34 30 31 18  7 17 35 13 13 15 29 23 16 11
##  [817] 11 48 19 10 12 16 52 16 11 14 45 14 32 42 66 17 17 14  7 11 20 12 14 20
##  [841]  9 20 24 25 32 11 18  7 27 13 17 17 25 33 18 41 11 19 24 33 12 31 23 22
##  [865] 18 26 17 10 28  9 34 14 33 16 17 59 16 17 10 28  7 27  9 31 16 13 22 13
##  [889] 27 10 16 30 28 37 12 34 26 24 18  8 26 29 24 39 10  6 17 10 10 19 17  6
##  [913] 25 37 20 11  8 12 22 39 17 18 12 11  7 14 10 20 16 29 17  9 12 28 21 22
```

```
##  [937] 24 20 18 11 27 13 13 18 13 34 33 18  7 19 13 33 43 34 20 13 41 14 21 19
##  [961] 22 13 32 12  8  9 17 14 17 13 13 20 32 15 15 21 15 20 18 14 18 23 50  6
##  [985] 16 26 24  5 25 33 20 25 22 18 28 23 16  6 26 15 28 17 23  9 36 34 40 31
## [1009] 44 15 24 24 13 18 15 29 51 27  9 13  6 17 15 18 19  6 19 34 56 33 15 17
## [1033] 62 26  8 22  7 11  9 14 10 29 11 28 18 27 10 35 17 16 14 35 27 67 16 33
## [1057] 10 12 42 27 15  9 11 10 11 37 31 12 28  7 10 11 18 14 12 11 12 17 22  8
## [1081] 30 27 16  8 40 12 17  8 31 11 32 24 41 13  9 39  9  7 36 25 19 21 11  6
## [1105] 34 11 17 15 28 12 26  7  7 32 19 12 17 17 14 37  9 15 21 16 14 21 14 77
## [1129] 27 12 28 11 33 20 11 11 19 12 32 25 18 27 13 29 10 13 15 13 20 16 17 25
## [1153] 10 14 22  7 11 31 15 32 16 12 21 14 32 13 16 20 11 17 31 19 26 15 60 38
## [1177] 31 29 23 10 10 26 30 29 19 20 13 20 17 31 32 18 26 12  6 36 22  9 13 22
## [1201] 34 24 12 14 28 19 12 81 15 17 11 30 25 23 14 20 16 21 31 20 14 12 10 21
## [1225]  5 21 14 13 19 32 16 13 17 24 17 12 10  9 18 16 35 19 15 11  7 13 14 19
## [1249] 15 15 34 44 11 12  7 29 14 10 15  7 16 15 20 16 12 79 16 17 13  7  5 17
## [1273] 15 15 41 13 27  9 18 15 28 20 29 40 23 23  9 26 27 36 46 15 25 10 20 19
## [1297] 11 24 36 26  7 26 19 28  7 10  9 30 34 39 15 27 15 18 27 26 13 35 49 30
## [1321] 14  9 19 24 17 26 16 15 24 38 10 24  5 15 36 10 45 39  9 20 57 23 14 16
## [1345] 14 33 15 40 42 13 33 18 10 17 11 23 15 28 22 15 21 43 18 28 12 17 40 11
## [1369] 29 27 27 48 27 13  8 11 32 25 24 23 34  9 19 12 18 27 14 14 23 34  8 13
## [1393] 17 14 10 21 33 13 16 15 15 19 16 39 17 11 20  9 14 11 16 24 25 30 18 11
## [1417] 14 23 11 18 29 24 31 14 25 45  5 45 19 15 17 18 14  8 19 25 19 36 10 16
## [1441] 10 14 15 11 24 26 36 20 10 18 13 33 18  9 11 15 37 20 21 21 10 25 13 23
## [1465] 22 16 31 24 18 12 28 17 10 21 16 27 17 49 21 30 32 22 24 25 18 29 15 14
## [1489] 24 16  5 22 12 18 29 10 38 16  7 27 10 22 11 23  9 21 19 37 16 28 29 17
## [1513] 31  7 28 24 18 14 23 29  6 10 12 11 15 15 32 11 33 15 61  8 29 37 36 23
## [1537] 14 16 22 16  7 11 11 10 13 20 28 11 15 11 20 17 20 18 38 12 24 31 39 10
## [1561] 14 17 11 24 16 15 31 30 17 20  9 23 29 12 31 12 14 48 53 14 21 18 17 24
## [1585] 38 15 20 19 34 27 22 27 66 17 16  9 17 13 19 19 28 12 30 38 10 10 23 30
## [1609] 47  9 21 14  9 11 32 32 21 18 26 21 23 21 13 35 23 36 27 11 40 24 27 13
## [1633] 57 23 15 30  8  9  9 10  5 11 10 19 16 28 15 16 46 20 31 16 10 23 11 47
## [1657] 10 10 21 10 51 10 17 18 25 17 12 29 13 44 19 26 25 24  8 12 30 26 14 20
## [1681] 16  9 10 24 18 20 10  7 15 26 13  8 25 11 21 11 13 17 75 10 19 10 23 14
## [1705]  8  7  7 34 18 17 17 45 22 20 17  7 17 33 18 26 16  7 19 16 11 16 18 21
## [1729] 29 16 11 31 22 15 13 11 19 39 14 24 12 26 36 34 13 17 22 21  7 44 17 20
## [1753] 17 11 57 20 22 12 25 37 11 15 33 16 32 16 19 13 23 11 21 14 24 19 43 17
## [1777] 35 15 10 20 26 28 20 10 19 26 10 31 32 33 44 30 17 16 19 16 27 18 17 45
## [1801] 17 10 35 11 33 14  8 14 10 30 39 35 24 41 37 10 23 35 35 21 22 49 67 16
## [1825] 12 11 10 19 13 40 26 15 24 19 23 17 15 29 21 22 35 24 20 64 14  6 23 16
## [1849] 18 13 21 17 17 13 32 18 20 26 14 16 11 17  7 17 12 25 41  7 24 33 21 22
## [1873] 19 17 22 15  7 18 14 19 10 27 15  8 18 10 15  7 21 12 12 13 17 12  8 12
## [1897] 35 28 20 15 17 26 40 27 23  5  8 61 37 20 20  9 15 10 28 34 55 14 20 17
## [1921] 16 32  6 40 16 20 15 10 28 28 12 11  9 34 17 19 44 17  9 20 15 18 24 32
## [1945] 11 16 11 37  7 28 20 25 13 10 21 17 26 35  5 14 14 14 16 24  9 13 18 15
## [1969] 59 20 26 21 19  7 11  8  4 13 16 40 19 33 13 11 13 47 16 12 11 11 19 12
## [1993] 22  9 10 24 18 19 14 34  9 13 11  7 16 10 19 12 29 12 23 12 51 27 24 32
## [2017] 28 24 23 10 10 12 30 10 12 82 16  7 35 10 14 46 30 24 18 26 19 14 34 18
## [2041] 21 16 18 18  9 15 11 12  9  6 16 22 16 33 15 10 23  7 37 26 10 15 11 29
## [2065] 18 15 13 12 23 20 23 31 12  8 21 11 15 35 14 14 47 15 20 18 21 12 35 11
## [2089] 39 11 23 17 18 14 16 16 16 20 19 15 27 68 17 19 14 22  8 19 21 17 21 20
## [2113] 16 30 26 16 28 19 23 39 21 34 21 32 14 31 52 16 11 25 24 22 11 20 15 37
## [2137] 24 27 28 27 18 14 14 19 30 19 43 31 12 15 15 39 26 16 23  9  8 25 22 35
## [2161] 10 38 15 14 14 24 12 31 40 42 17 50 11 30 14 18 10 14 35  8 30 16 13 13
## [2185] 11 21 12 13 22 27 20  7 36 22 18 55 15 20 15 30 10 29 14 57 23 16 23 11
## [2209] 23 23 23 13 19 12 20 17  8 11 28 15 18 12 74 13 16 31 30 10 26 15 22 11
```

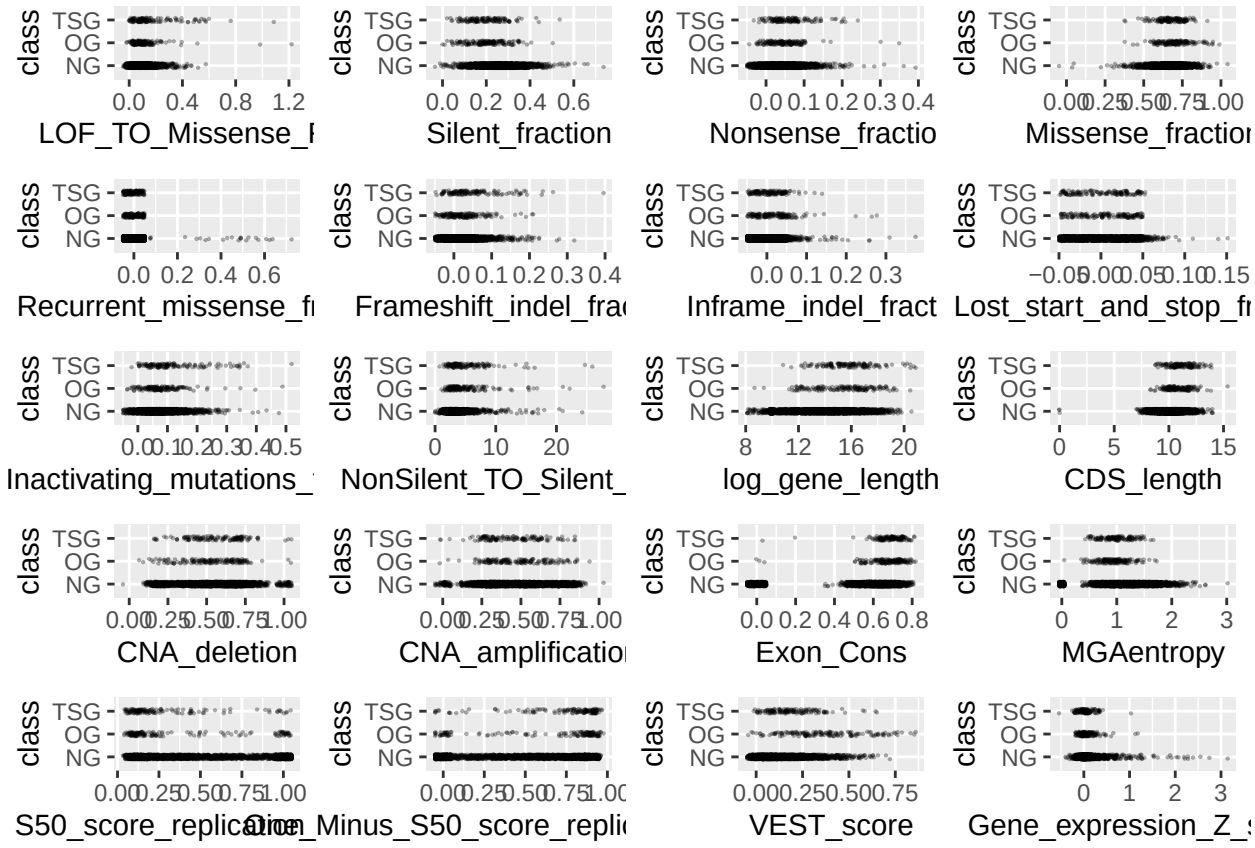
```
## [2233]  4 20 14 21 16 35 28 13 15 17 22 10  8 27 23 17 18 16 18 27 10 15 12 14
## [2257] 28 37 52 16 20 43 13 14 28 55 59 25 21 18 17 15 19 12 22 17 13 34  9 18
## [2281] 17 16 30 16  8 13 15 14 16 16 36 13 28 36 21 35 13 21 10  9 19 22 15 20
## [2305] 21 11 21 14 40 22 19 38 28 14 27 14 16 16 64 14 25 27 26 14 20 19 32 31
## [2329] 18 23 14 13  8 14 22 55 15 28 13 21 18 20 33 15  9 17 16 17 14 24 11 43
## [2353] 24 14 27  8 25 34 19 14 51 12 45 11 22 15 10 43 16 26 14 16 15 13 10 29
## [2377] 16  9 12 10 17 47 20 26 29 26 18 22 35 15 10 11 15 12 18 13 22 12 26 64
## [2401] 26 13 13  9 12 19 20 41 12 52 23 16 13 15 22 27 11  6 14  7 30 24 20 18
## [2425] 10 33 12 12 30  7 27 13 15 45 14 36 13 12 28 12 15 37 15 29 47 17 13 49
## [2449] 14 19 39 21 35 15 22 17 33 19 29 20 28 17 26 63 18 12 18 14 15 12 20 11
## [2473] 16  9 17 12 12  9  9 22 34 13 16  8 10 35 17 21 26 28 32 35 17 26 30 29
## [2497] 14 18 29  8 22 12 12 23 11 10  9 12 17 26 33 30 12 15 15 10  6 14 24 23
## [2521]  9  9 20 35  8 19 13 29 33 11 17 17 17 29 27 21 14 18 29 28 11 17 29 10
## [2545] 32 31 10 15  9 17 58 13 12 26 12 21  5 30 12  9 41 23 29 15 31 10 12  8
## [2569] 11 14 18 21 26 15 16 18 16 29 12 20 13 30 11 27 16 37 20 13  9  9 22  8
## [2593] 14 21 13 12 26 12 35 13 32  7  9 41 18 22 14 44 17 17 27 12 37 31  4 24
## [2617] 14 55 14 37 18 30 28 24 24 27 25 11 10 22 31 23 22 24  9 19 30 20 24 12
## [2641] 13 20 15 12 13 16 19 29 15 11 12 13  9 42 16 17 20 18 10  9 20 21 34 17
## [2665] 25 12 15 15 16 14 31 36 20 12 17 27 10 15 20 33 17 18 10 22 34 33 14 14
## [2689] 10  9 17  8 20 13 13 17  8 12  9 19 11 16 29 39 27 48 11 25 23 14 23 19
## [2713] 15 31 14 12 11 23 18 17 26 18 27 24 13 19 12 11 27 21 14 19 17 12 27 38
## [2737] 23 40 16 28 10 16 10 16 13 13 28  7 22 11 12 27  4 23 31 31 12  9 28 14
## [2761] 20 40 27 80 39 19 13 16 15 22 18 30 25 30 10 11 27 11 29 22 17 15 27 13
## [2785] 14 11 21 12  6 56 16 18  9 22  8 20 13 40 19 15 37 10 18 15 10 15 16 22
## [2809] 20  8 18 13 47 22 16 15 13 28 23 16 25  9 38 27 24 19 13 16 34 33 12 13
## [2833] 19 10 12 25 11 17 20 18 11 17 25 19 17 19 47  9 28 24 10 32 11 15 24 15
## [2857] 61 14  9 48 29 22 19 16 10  6 25 33 43 15 28 22 11  9  5 33 45 12 17 28
## [2881] 14 25 23 26 26 22 19 28 23 12 34 27 36 16 27 28 29 18 68 16  9 21 17 14
## [2905] 43 23 11 11 24 29 21 23 20 31 13 24 31 40 14 14  9 24 49 10 29 13 15 19
## [2929] 15 24 11 11 21 16  8 39 15 22 48 11 12 17 31 10 20 25 57 88 16 23 23 17
## [2953] 11 19 22 20 29 37 12 27 13 11 10  9  7 16 16 15  9  8 33 11 14  9 40 13
## [2977] 16 26 27 19 11 27 44 25 28 30 30 57  7 16 25 19 32 15 27 20 11 20 15 26
## [3001] 14 18 31 23 15 12 18 18 11 13 17 20 11 45 30 15 12 19  9 15 16 19 11 11
## [3025] 14 27 28 20 25 13 16 50 11 25 10 51 19 14 33  9 29 11 13  8 15 15 19  7
## [3049] 20 34 16 13 49 12 21  9  6 17 11 23 12 31 10 12 12 10 27 16 42 13 22 36
## [3073] 19 11 52 11 21 13  7 13  9 16 31 31 13 13 11 18 17 23 34 15 29 14 25 15
## [3097] 18 28 31 27 16 24 10 17 32 22 18
```

```
training <- training[n_zeroes <= 50, ]
```

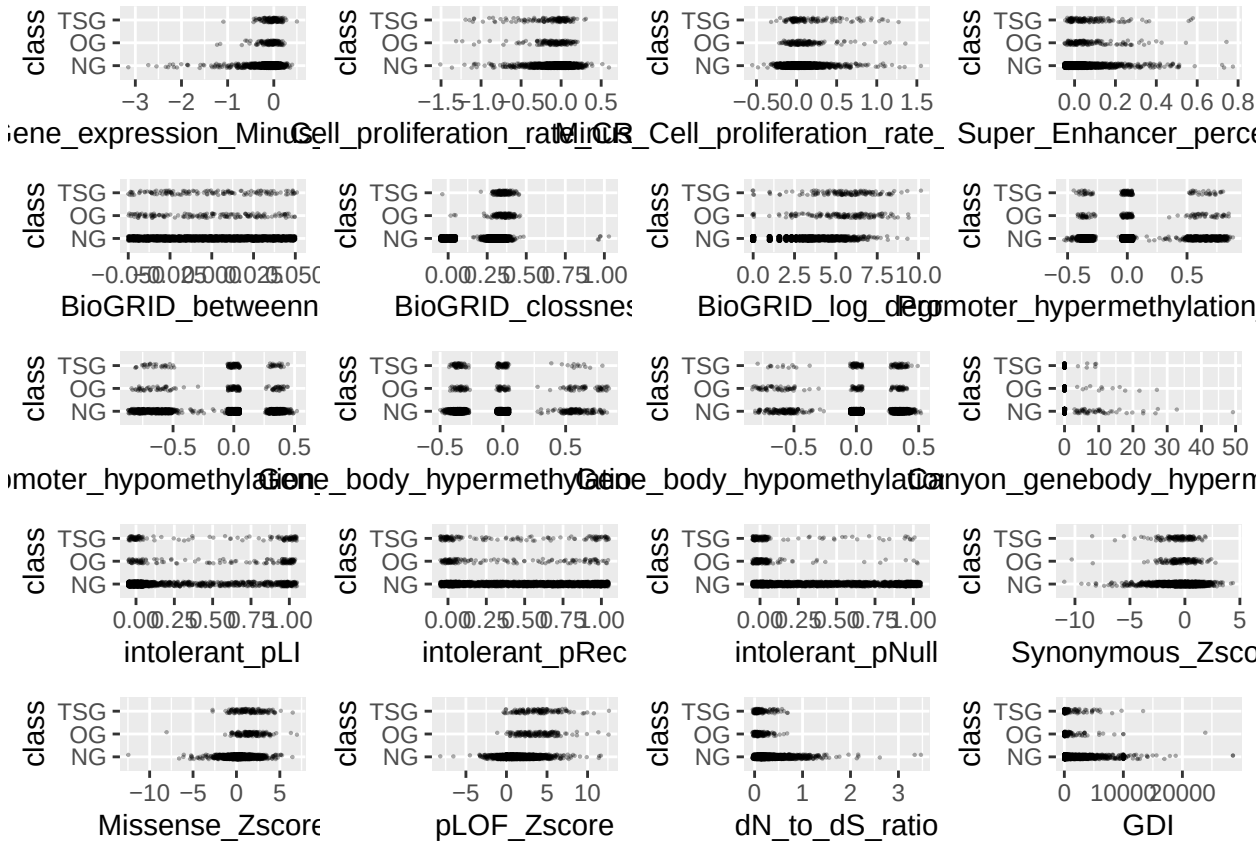
```
library(ggplot2)
scatter <- function(var) {
  ggplot(training, aes_string(var, "class")) +
    geom_jitter(width = 0.05, height = 0.1, size = 0.1,
                colour = rgb(0, 0, 0, alpha = 1 / 3))
}
scat_plot <- lapply(names(training)[-99], scatter)
library(gridExtra)
grid.arrange(grobs = scat_plot[1:20], ncol = 4)
```



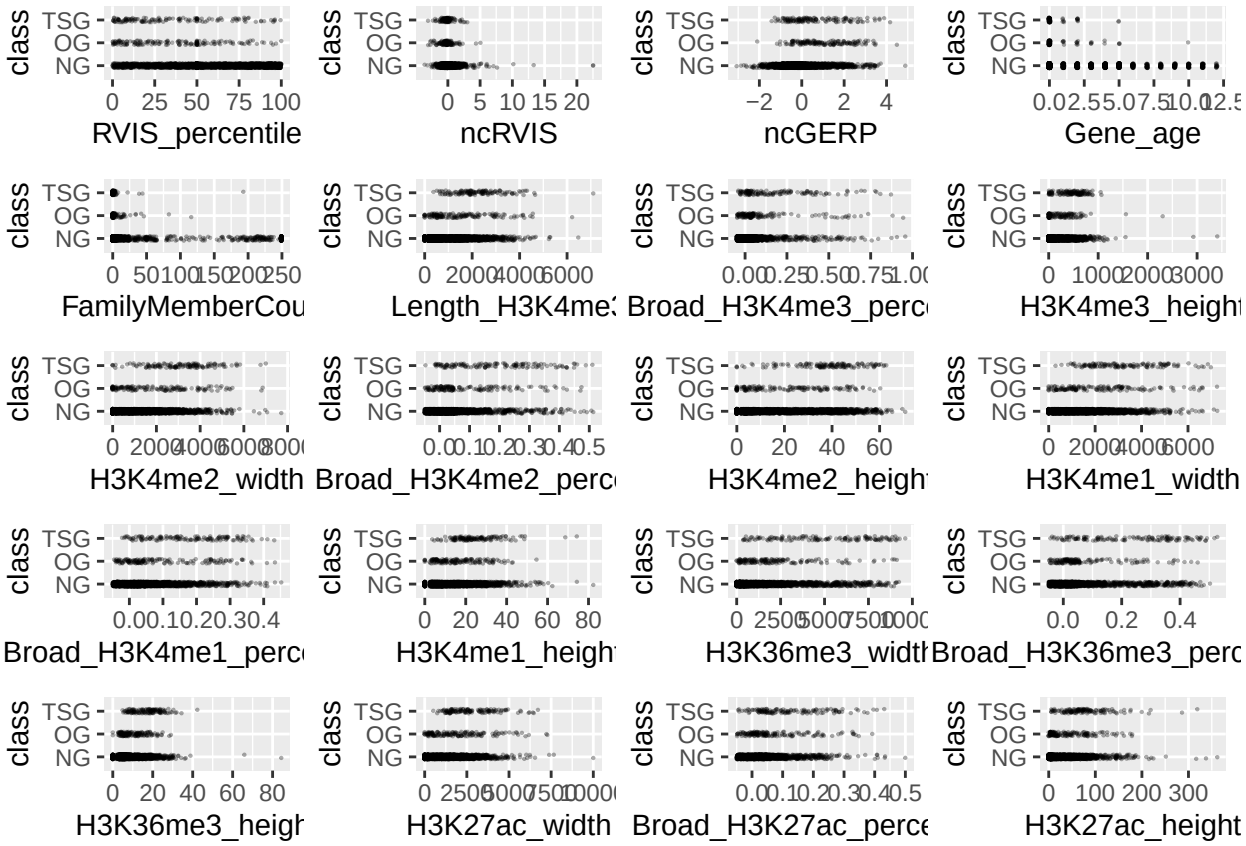

```
grid.arrange(grobs = scat_plot[21:40], ncol = 4)
```



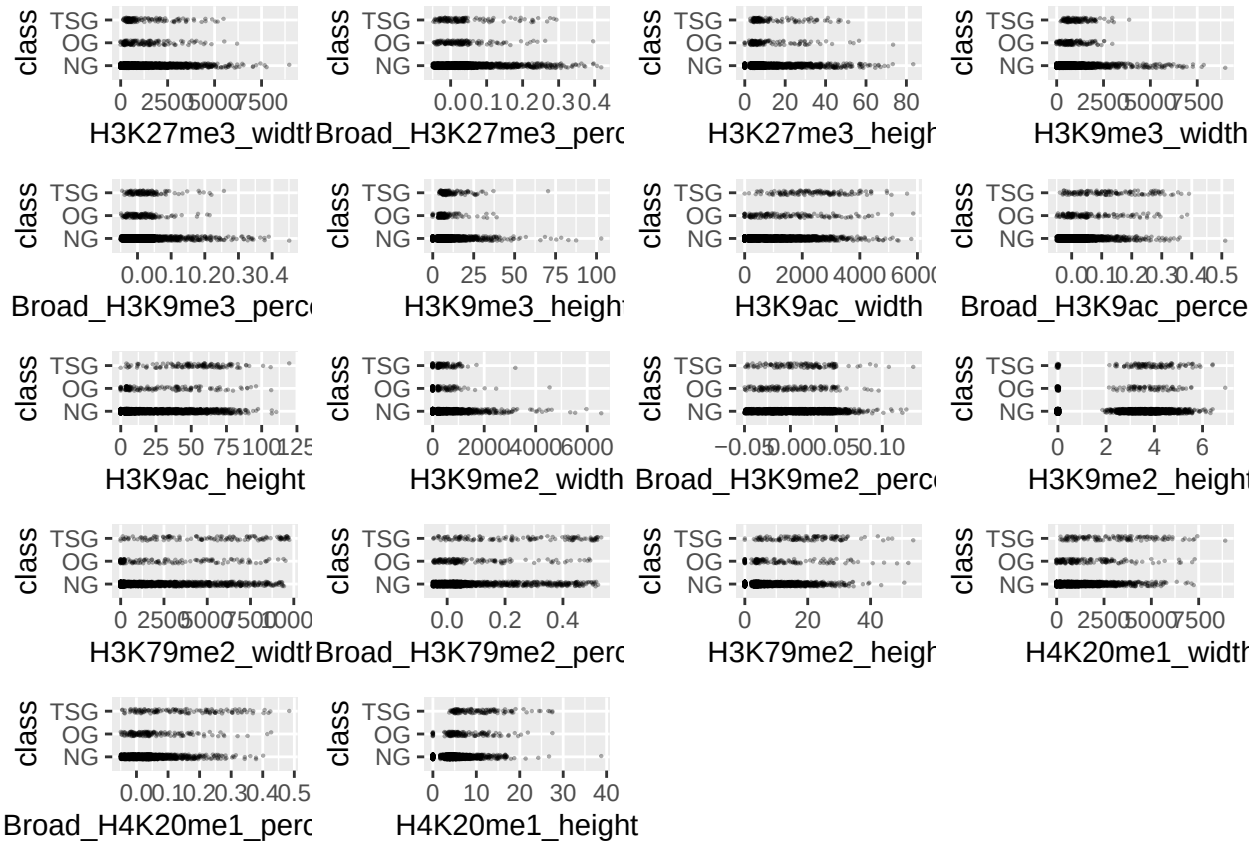

```
grid.arrange(grobs = scat_plot[41:60], ncol = 4)
```




```
grid.arrange(grobs = scat_plot[61:80], ncol = 4)
```




```
grid.arrange(grobs = scat_plot[81:98], ncol = 4)
```




```
library(dplyr)
```

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

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

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

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

```
sig <- logical(98)
names(sig) <- names(training)[-99]
k <- 1
diffs <- logical(98)
for (var in names(training)[-99]) {
  model <- aov(training[[var]] ~ factor(training$class))
  sig[k] <- summary(model)[[1]][1, 5]
  diffs[k] <- all(TukeyHSD(model)$`factor(training$class)`[, 4] < 0.05)
  k <- k + 1
}
sort(sig[diffs])
```

```
##                   Broad_H4K20me1_percentage
##                                2.563341e-146
##                                   VEST_score
##                                1.141090e-121
##                     Broad_H3K9ac_percentage
##                                7.096289e-111
##                            H3K79me2_height
##                                1.252905e-110
##                             H3K79me2_width
##                                5.618333e-110
##                           Missense_Entropy
##                                1.254512e-109
##                   Broad_H3K79me2_percentage
##                                4.654407e-107
##                              intolerant_pLI
##                                5.420482e-107
##          Missense_Damaging_TO_Benign_Ratio
##                                1.510740e-106
##                    Broad_H3K4me2_percentage
##                                1.195197e-105
##                             H4K20me1_width
##                                6.869405e-105
##                   Broad_H3K36me3_percentage
##                                8.244495e-104
##                             H3K36me3_width
##                                4.633126e-103
##                            H4K20me1_height
##                                1.332273e-101
##                         LOF_TO_Silent_Ratio
##                                4.745882e-101
##                    Broad_H3K27ac_percentage
##                                 5.173854e-97
##                    Broad_H3K4me1_percentage
##                                1.023092e-95
##                                LOF_KB_Ratio
##                                4.040183e-93
##                           Missense_KB_Ratio
##                                4.866503e-89
##                                       N_LOF
##                                1.085326e-88
##                             H3K4me1_width
##                                6.433999e-80
##                         LOF_TO_Benign_Ratio
##                                1.552250e-78
##                    Broad_H3K4me3_percentage
##                                3.171192e-78
##                             H3K36me3_height
##                                2.795995e-75
##                             H3K4me2_width
##                                9.456016e-74
##                                H3K9ac_width
##                                1.107951e-71
## Missense_Damaging_TO_Missense_Benign_Ratio
##                                2.508133e-68
```

```
##                              H3K9ac_height
##                                1.751625e-65
##                     Missense_TO_Benign_Ratio
##                                5.840566e-64
##                              H3K27ac_width
##                                2.548087e-63
##                             Length_H3K4me3
##                                1.108383e-60
##                                      ncGERP
##                                2.415790e-60
##                             H3K27ac_height
##                                2.528240e-58
##                             H3K4me3_height
##                                7.953208e-51
##                             H3K4me2_height
##                                9.844930e-49
##                          LOF_TO_Total_Ratio
##                                8.428823e-48
##                                    N_Splice
##                                4.434886e-46
##                       LOF_TO_Missense_Ratio
##                                1.945822e-42
##                             H3K4me1_height
##                                6.778589e-42
##                                  CDS_length
##                                2.137142e-34
##                    Frameshift_indel_fraction
##                                5.027810e-34
##              Inactivating_mutations_fraction
##                                1.191586e-31
##           Cell_proliferation_rate_CRISPR_KD
##                                9.738243e-23
##     Minus_Cell_proliferation_rate_CRISPR_KD
##                                9.738243e-23
##                       Inframe_indel_fraction
##                                1.917583e-12
```

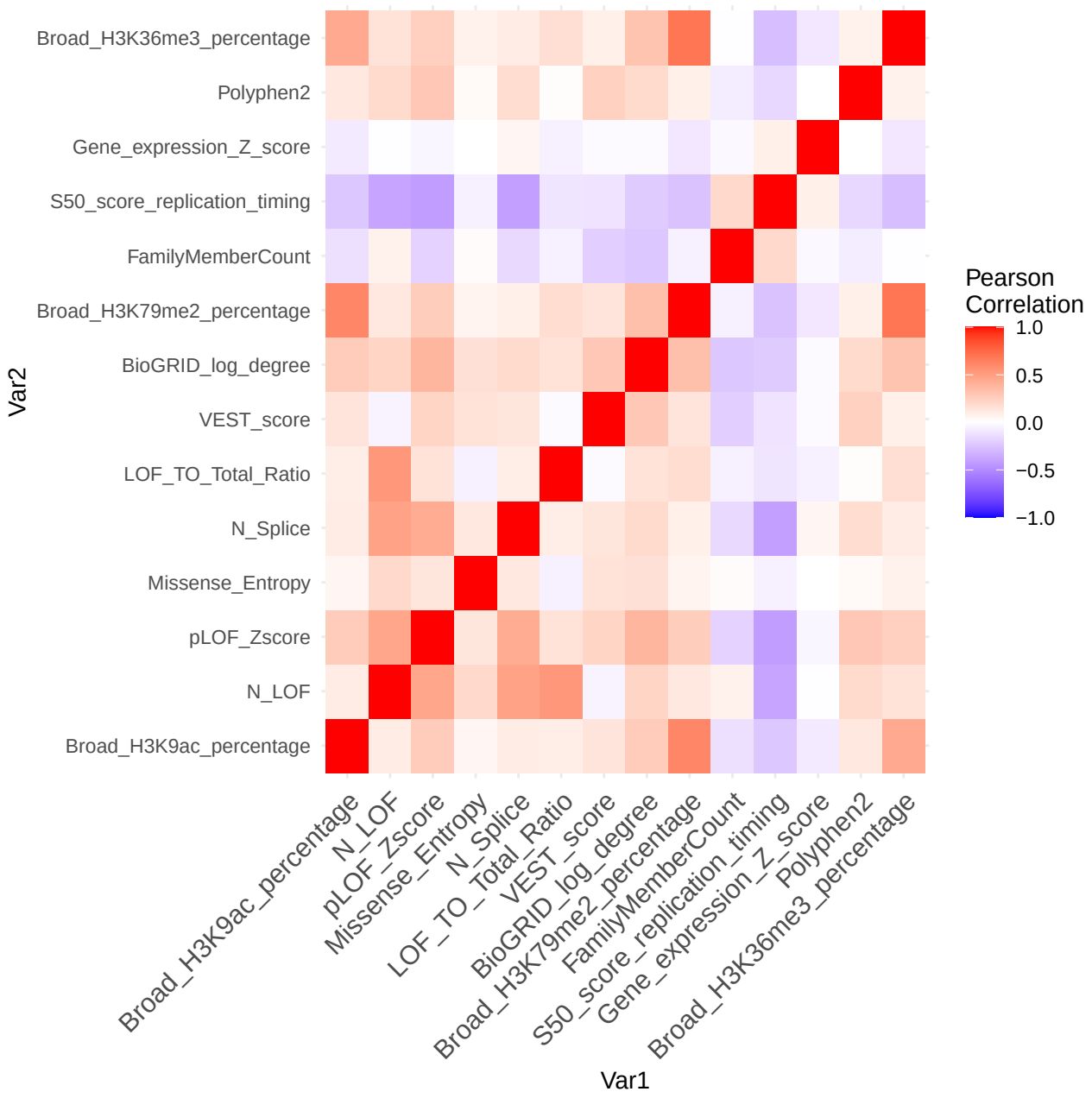
```
score <- function (conf_mat) {
  print(sum(diag(conf_mat) * c(1, 20, 20)))
  print(sum(diag(conf_mat) * c(1, 20, 20)) / sum(apply(conf_mat, 2, sum) * c(1, 20, 20)))
}

classify <- function(probs) {
  if (any(probs[2:3] > 1 / 22)) {
    subset <- probs[2:3]
    output <- which(subset == max(subset))
    if (length(output) > 1) {
        output <- sample(1:2, 1)
    }
  } else {
    output <- 0
  }
  output
}
```

```r
library(dplyr)
vars <- training %>% select(Broad_H3K9ac_percentage, N_LOF, pLOF_Zscore,
                            Missense_Entropy,
                           N_Splice, LOF_TO_Total_Ratio, VEST_score,
                           BioGRID_log_degree,
                           Broad_H3K79me2_percentage, FamilyMemberCount,
                           S50_score_replication_timing, Gene_expression_Z_score,
                           Polyphen2, Broad_H3K36me3_percentage, class)
vars$class <- factor(vars$class)
levels(vars$class) <- c("NG", "OG", "TSG")
cor_mtx = round(cor(vars[, names(vars) != "class"]), 2)
library(reshape2)
#reshape it
melted_cor_mtx <- melt(cor_mtx)

#draw the heatmap
cor_heatmap = ggplot(data = melted_cor_mtx, aes(x=Var1, y=Var2, fill=value)) + geom_tile()
cor_heatmap = cor_heatmap +
scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0, limit = c(-1,1), space =
theme_minimal() +
theme(axis.text.x = element_text(angle = 45, vjust = 1, size = 12, hjust = 1))

cor_heatmap
```




```
library(dplyr)
set.seed(12)
vars <- training %>% select(Broad_H3K9ac_percentage, N_LOF, pLOF_Zscore,
                            Missense_Entropy,
                           N_Splice, LOF_TO_Total_Ratio, VEST_score,
                           BioGRID_log_degree,
                           Broad_H3K79me2_percentage, FamilyMemberCount,
                           S50_score_replication_timing, Gene_expression_Z_score,
                           Polyphen2, Broad_H3K36me3_percentage, class)
library(caret)
```

```
## Loading required package: lattice
```

```
vars_test <- createDataPartition(vars$class, p = 0.8,
                                 list = FALSE)
vars_train <- vars[vars_test, ]
vars_test <- vars[-vars_test, ]

train_cont <- trainControl(method = "cv", number = 5, classProbs = TRUE, savePredictions = TRUE)


knn_ft <- train(class ~ ., data = vars_train, method = "knn", preProc = c("center", "scale"),
                trControl = train_cont, tuneGrid = expand.grid(k = c(1, seq(from = 5, to = 25, by = 5),
for (k in c(1, seq(from = 5, to = 25, by = 5), seq(from = 30, to = 60, by = 10))) {
  print(k)
  preds <- predict(knn_ft, newdata = vars_test, type = "prob")
  knn_mod <- table("pred"=unlist(apply(preds, 1, classify)), "obs" = vars_test$class)
  print(knn_mod)
  score(knn_mod)
}
```

```
## [1] 1
##     obs
## pred  NG  OG TSG
##    0 421   3   2
##    1  65  22   4
##    2  64   4  20
## [1] 1261
## [1] 0.7642424
## [1] 5
##     obs
## pred  NG  OG TSG
##    0 421   3   2
##    1  63  25   4
##    2  66   1  20
## [1] 1321
## [1] 0.8006061
## [1] 10
##     obs
## pred  NG  OG TSG
##    0 421   3   2
##    1  63  23   4
##    2  66   3  20
## [1] 1281
## [1] 0.7763636
## [1] 15
##     obs
## pred  NG  OG TSG
##    0 421   3   2
##    1  66  22   4
##    2  63   4  20
## [1] 1261
## [1] 0.7642424
## [1] 20
##     obs
## pred  NG  OG TSG
##    0 421   3   2
```

```
##    1  68  23   4
##    2  61   3  20
## [1] 1281
## [1] 0.7763636
## [1] 25
##     obs
## pred  NG  OG TSG
##    0 421   3   2
##    1  70  23   4
##    2  59   3  20
## [1] 1281
## [1] 0.7763636
## [1] 30
##     obs
## pred  NG  OG TSG
##    0 421   3   2
##    1  63  23   4
##    2  66   3  20
## [1] 1281
## [1] 0.7763636
## [1] 40
##     obs
## pred  NG  OG TSG
##    0 421   3   2
##    1  65  23   4
##    2  64   3  20
## [1] 1281
## [1] 0.7763636
## [1] 50
##     obs
## pred  NG  OG TSG
##    0 421   3   2
##    1  65  23   4
##    2  64   3  20
## [1] 1281
## [1] 0.7763636
## [1] 60
##     obs
## pred  NG  OG TSG
##    0 421   3   2
##    1  63  21   4
##    2  66   5  20
## [1] 1241
## [1] 0.7521212
```

```
train_cont <- trainControl(method = "cv", number = 5, classProbs = TRUE, savePredictions = TRUE)
qda_ft <- train(class ~ ., data = vars_train, method = "qda", preProc = c("center", "scale"),
                trControl = train_cont)
preds <- predict(qda_ft, newdata = vars_test, type = "prob")

qda_mod <- table("pred"=apply(preds, 1, classify), "obs" = vars_test$class)
qda_mod
```

```
##     obs
## pred  NG  OG TSG
```

```
##    0 479   4   3
##    1  37  21   6
##    2  34   4  17
```

```
score(qda_mod)
```

```
## [1] 1239
## [1] 0.7509091
```

```
train_cont <- trainControl(method = "cv", number = 5, classProbs = TRUE, savePredictions = TRUE)
lda_ft <- train(class ~ ., data = vars_train, method = "lda", preProc = c("center", "scale"),
                trControl = train_cont)
preds <- predict(lda_ft, newdata = vars_test, type = "prob")

lda_mod <- table("pred"=apply(preds, 1, classify), "obs" = vars_test$class)
lda_mod
```

```
##     obs
## pred  NG  OG TSG
##    0 509   4   4
##    1  18  22   3
##    2  23   3  19
```

```
score(lda_mod)
```

```
## [1] 1329
## [1] 0.8054545
```

```
# tests <- read.csv("test.csv")
# preds <- predict(lda_ft, newdata = tests, type = "prob")
# preds <- apply(preds, 1, classify)
# names(preds) <- tests$id
# csv_file <- data.frame("id" = tests$id,
#                        "class" = preds)
# # write.csv(csv_file, "modelpredictions4.csv", row.names = FALSE)
# model_1 <- read.csv("modelpredictions.csv")
# model_2 <- read.csv("modelpredictions2.csv")
# model_1_beta <- read.csv("modelpredictions_beta.csv")
# model_2_beta <- read.csv("modelpredictions2_beta.csv")
# model_3 <- read.csv("modelpredictions3.csv")
# table(model_1$class, model_1_beta$class)
# table("BEST MOD" = model_3$class, csv_file$class)
# mean(model_1$class == csv_file$class)
```