GLM for Landslide Occurance Prediction

Wei ZHOU

4/29/2020

```
landslide.train=read.csv("landslide_training_data.csv",header = T)
landslide.validation=read.csv("landslide_validation_data.csv",header = T)
landslide.test=read.csv("landslide_test_data.csv",header = T)
```

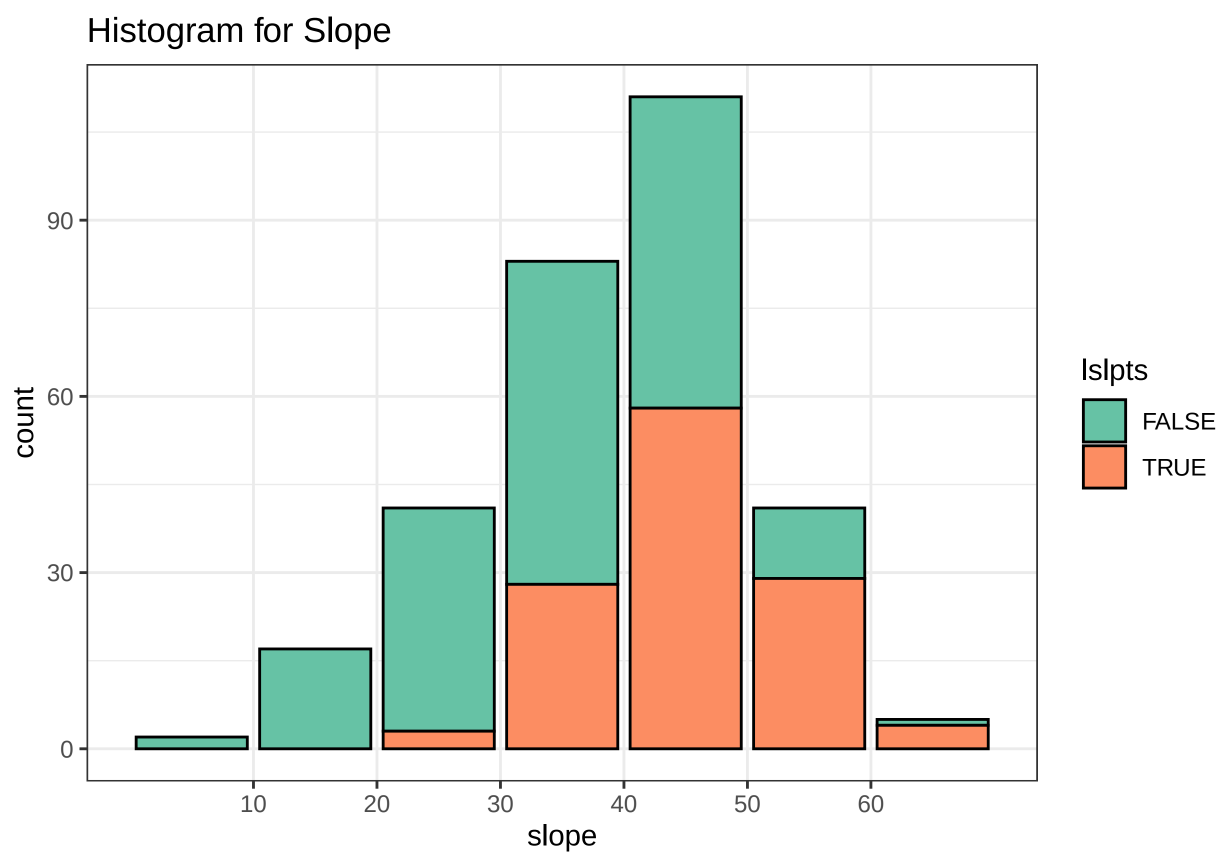
1. Data Exploration and tidying

1.a Data Exploration

- Develop histograms of the five different predictor variables in the training set
- Develop a correlation plot of the predictor variables

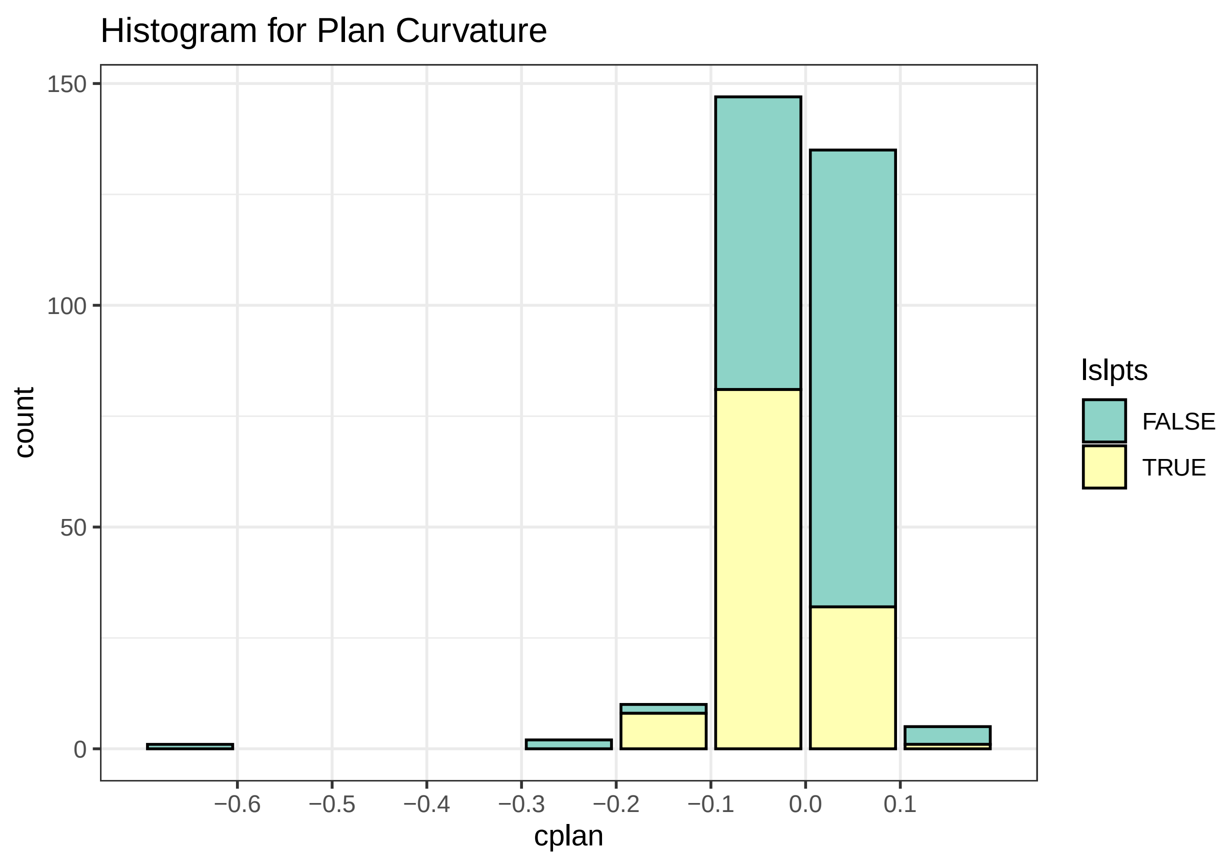
```
# slope
slope.plot <- ggplot(data=landslide.train, aes(slope,fill = lslpts)) +
  geom_bar(colour = "black", position = "stack") +
  scale_x_binned()+
  scale_fill_brewer(palette = "Set2")+
  labs(title = "Histogram for Slope")+
  theme_bw()

# cplan
cplan.plot <- ggplot(data=landslide.train, aes(cplan,fill = lslpts)) +
  geom_bar(colour = "black", position = "stack") +
  scale_x_binned()+
  scale_fill_brewer(palette = "Set3")+
  labs(title = "Histogram for Plan Curvature")+
  theme_bw()

# cprof
cprof.plot <- ggplot(data=landslide.train, aes(cprof,fill = lslpts)) +
  geom_bar(colour = "black", position = "stack") +
  scale_x_binned(breaks = c(-0.10,-0.06,-0.04,-0.02,0,0.02,0.04))+
  scale_fill_brewer(palette = "Pastel2")+
  labs(title = "Histogram for Profile Curvature")+
  theme_bw()
# elev
elev.plot <- ggplot(data=landslide.train, aes(elev,fill = lslpts)) +
  geom_bar(colour = "black") +
  scale_x_binned(breaks = c(1800,2000,2200,2400,2600,2800,3000))+
```

```r
  scale_fill_brewer(palette = "Pastel1")+
  labs(title = "Histogram for Elevation ")+
  theme_bw()

# log10_carea
log10_carea.plot <- ggplot(data=landslide.train, aes(log10_carea,fill = lslpts)) +
  geom_bar(colour = "black", position = "stack") +
  scale_x_binned()+
  scale_fill_brewer(palette = "Accent")+
  labs(title = "Histogram for Catchment Area")+
  theme_bw()


train.variable<- data.frame(landslide.train$slope ) %>%
  cbind(data.frame(landslide.train$cplan)) %>%
  cbind(data.frame(landslide.train$cprof)) %>%
  cbind(data.frame(landslide.train$elev)) %>%
  cbind(data.frame(landslide.train$log10_carea))
colnames(train.variable) <- c("slope","cplan","cprof","elev","log10_carea")

# correlation
cor_plot <- ggcorr(train.variable,label = TRUE,label_size = 3, hjust = 0.75,
                   size = 5, color = "grey40",
       low = "#3399FF", mid = "#FFFF66", high = "#CC0033", method = c("pairwise", "spearman"))+
  theme(legend.title = element_text(size = 11))

slope.plot
```

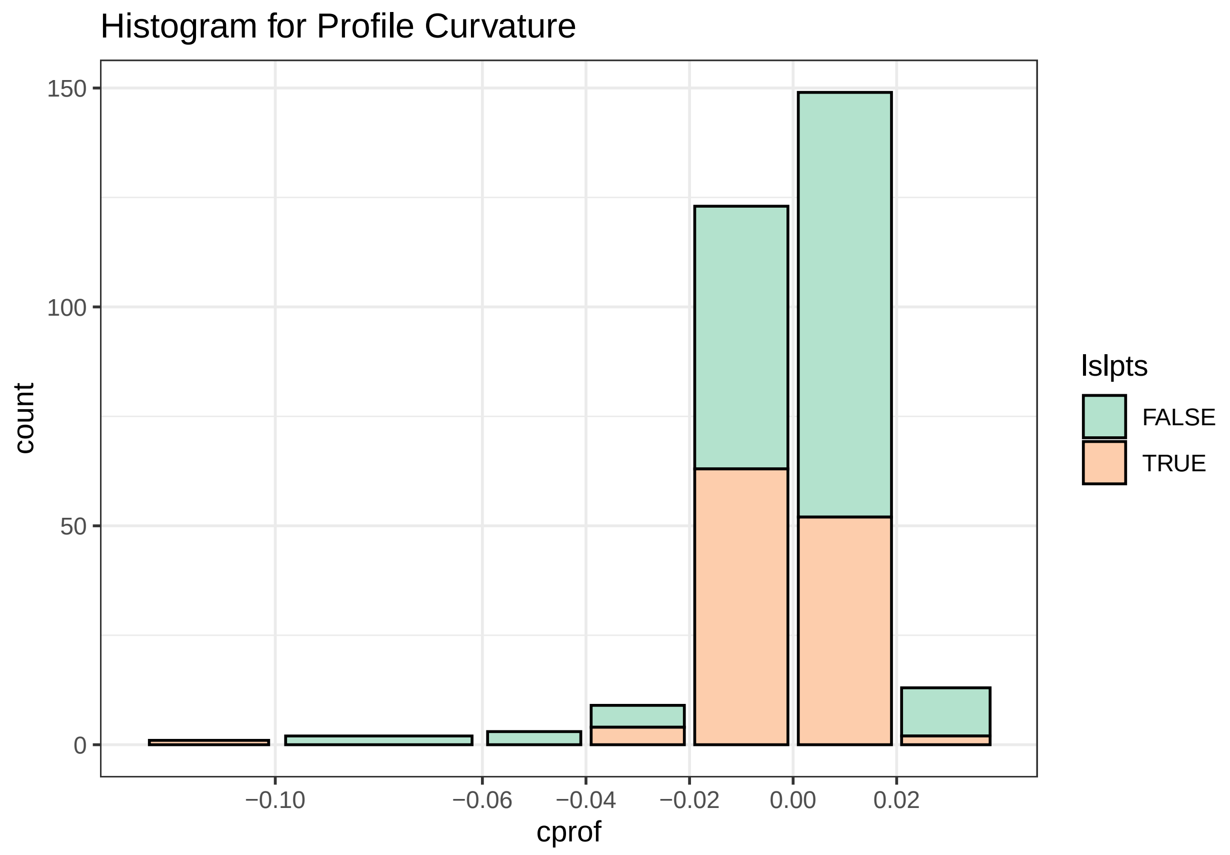


Histogram for Slope
count
slope
lslpts
FALSE
TRUE
0
30
60
90
10
20
30
40
50
60

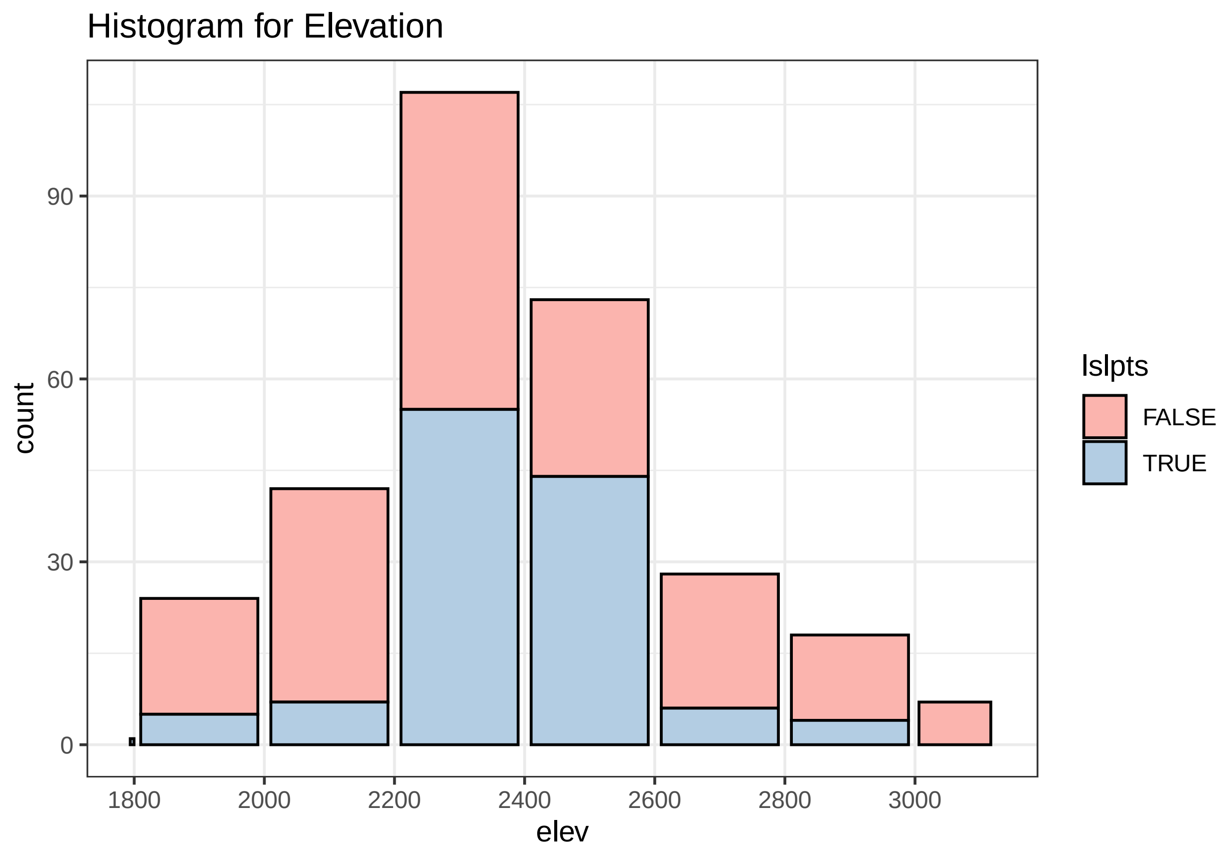
```
cplan.plot
```


Histogram for Plan Curvature
count
150
100
50
0
−0.6
−0.5
−0.4
−0.3
−0.2
−0.1
0.0
0.1
cplan
lslpts
FALSE
TRUE

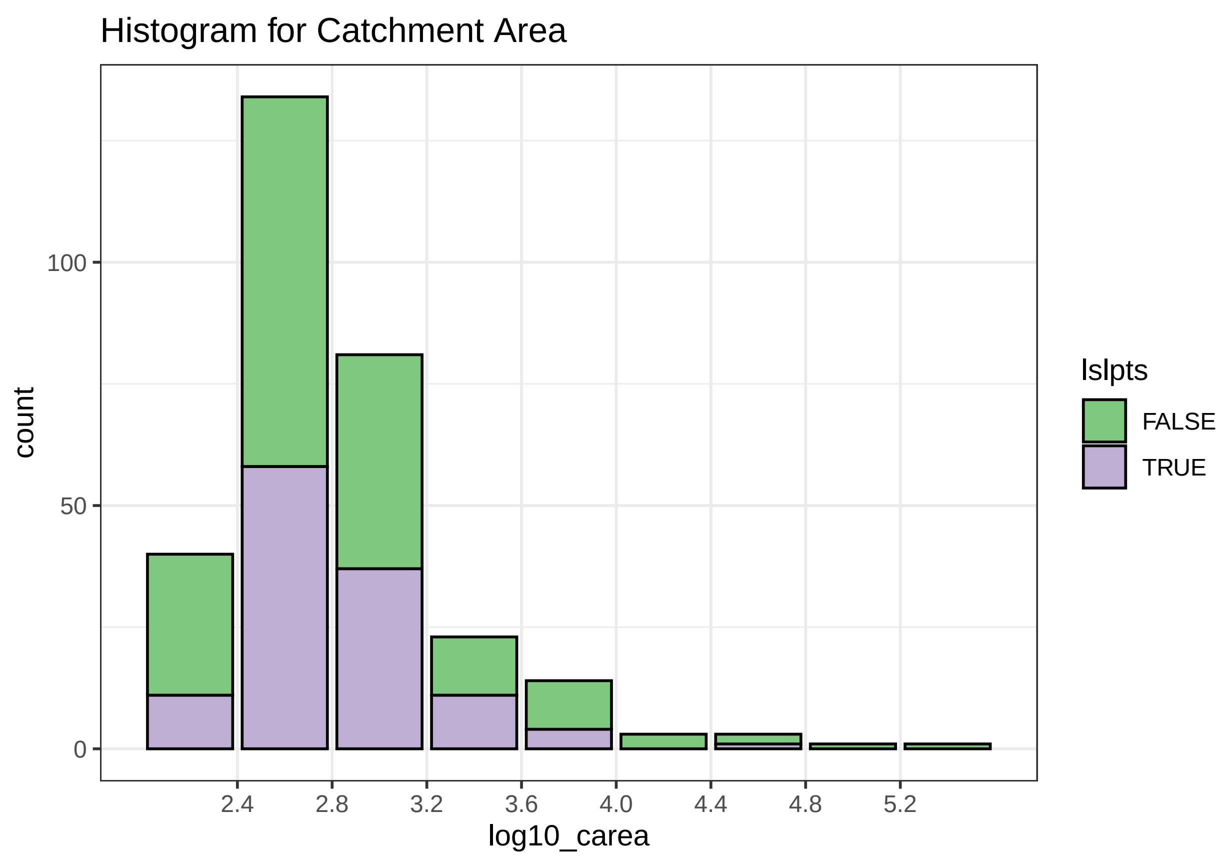
```
cprof.plot
```


Histogram for Profile Curvature
150
100
50
0
count
−0.10
−0.06
−0.04
−0.02
0.00
0.02
cprof
lslpts
FALSE
TRUE

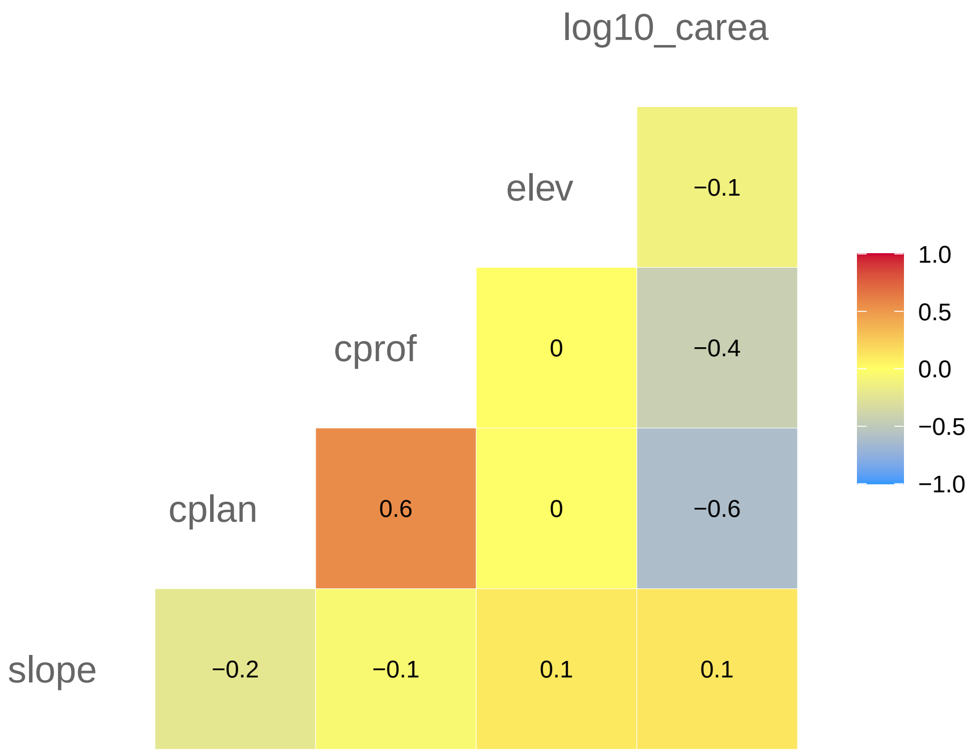
```
elev.plot
```



```
log10_carea.plot
```


Histogram for Catchment Area
count
100
50
0
2.4
2.8
3.2
3.6
4.0
4.4
4.8
5.2
log10_carea
lslpts
FALSE
TRUE

```
cor_plot
```



Comments:

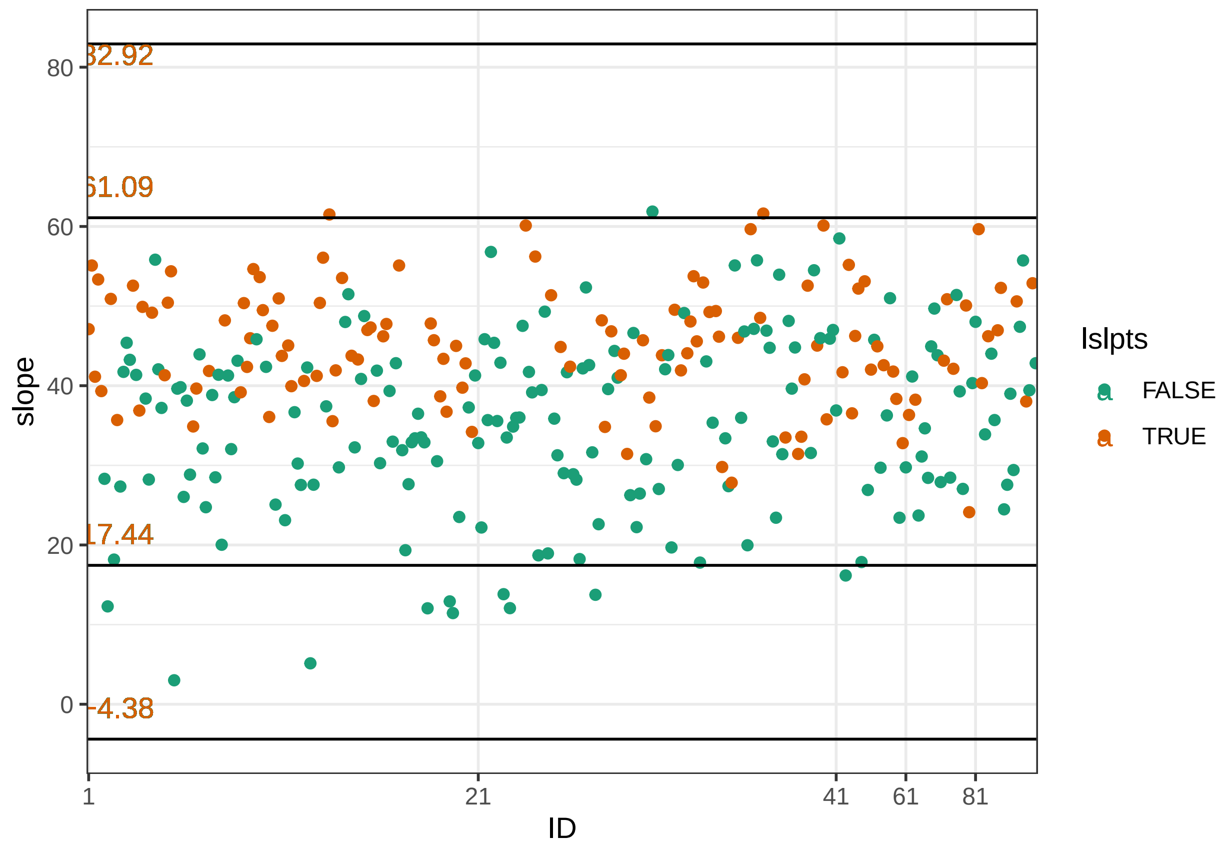
The slope degree ranges from close 0 to more than 60, but most of them lie between 20 to 50, and the data points at the far left end of the x-axis can be considered of the outliers;
The plan curvature ranges from less than -0.6 to greater than 0.1, but most of them lie between -0.1 to 0.1, and the data points at the far right end of the x-axis can be considered of the outliers;
The profile curvature ranges from less than -0.12 to greater than 0.02, but most of them lie between -0.02 to 0.02, the data points at the far right and left end of the x-axis can be considered of the outliers;
The elevation ranges less than 1800 to greater than 3100, but most of them lie between 2300 to 2500, the data points at the far right end of the x-axis can be considered of the outliers;
The catchment Area ranges from less than 24 to greater than 5.2, but most of them lie between 2.4 to 3.2, the data points at the far left end of the x-axis can be considered of the outliers;
All the parameters are normalized-like distribution.

1.b Outliers

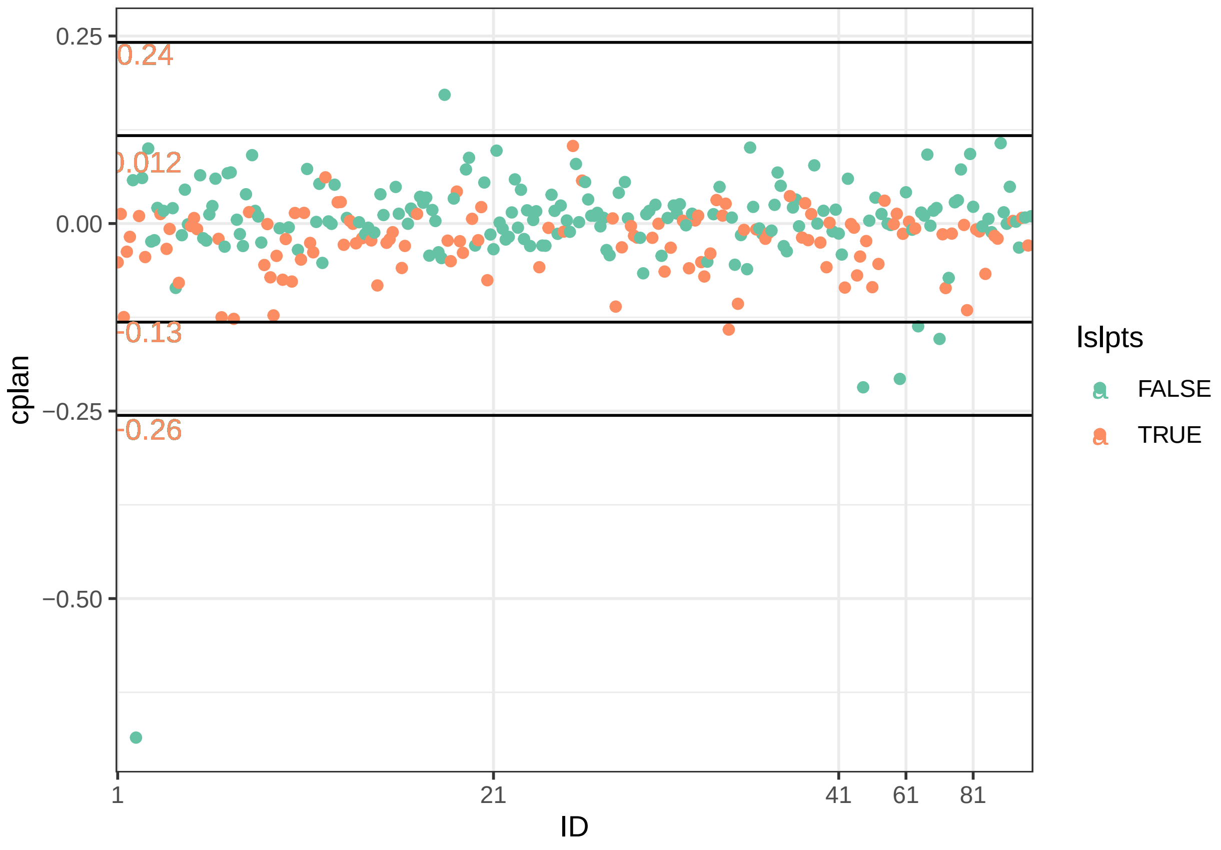
- Sometimes outliers can indicate errors in measurements or other glitch in our data-collection or processing. In the context of modelling and prediction, outliers sometimes drastically impact the fit / bias of the model. Note that loss functions related to squared errors are highly sensitive to outliers.

```
# slope
train.slope <- data.frame(landslide.train$slope,landslide.train$lslpts)
```

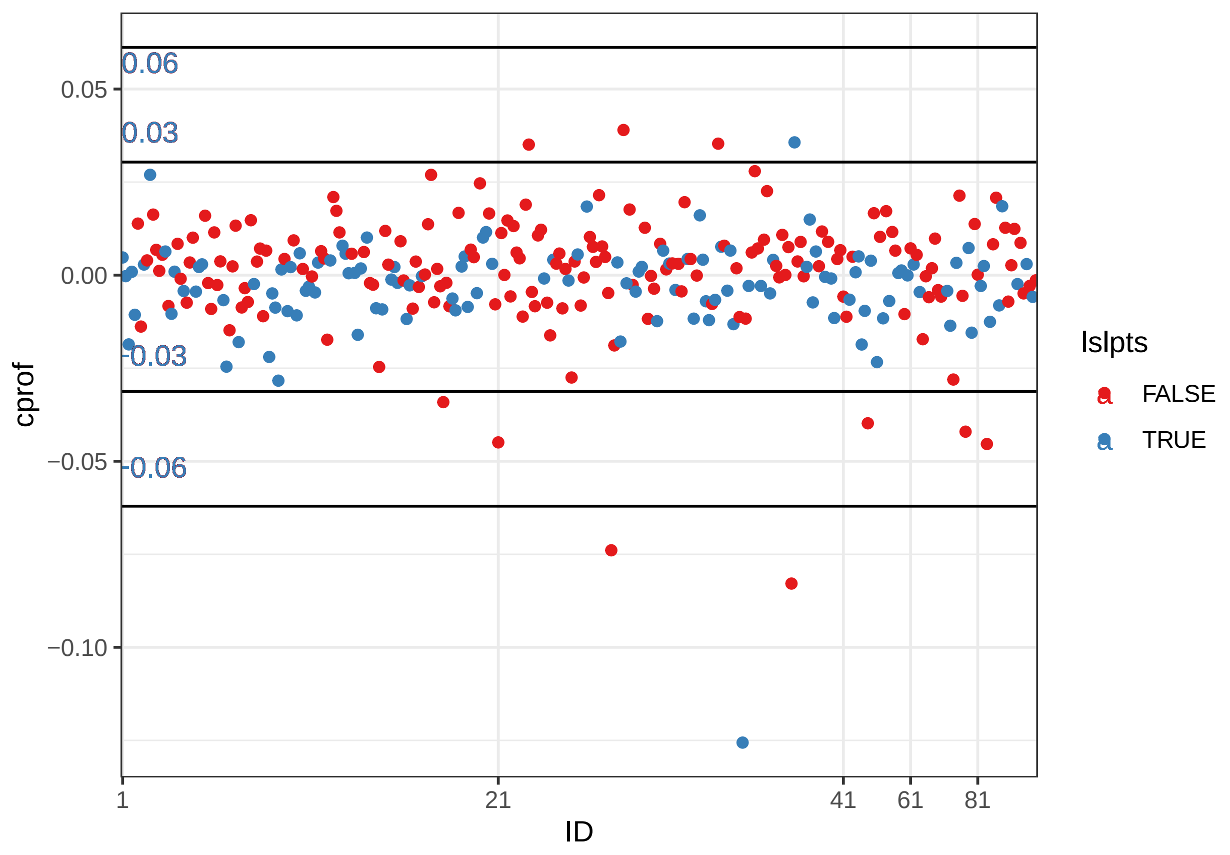
```r
train.slope <- train.slope %>% mutate(rownames(train.slope))
colnames(train.slope) <- c("slope","lslpts","ID")

train.slope_plot<-ggplot(train.slope,aes(x = ID, y = slope, color = lslpts))+
  geom_point()+
  scale_x_discrete(breaks=seq(1, 100, by = 20))+
  geom_hline(yintercept = mean(train.slope$slope)+ 2*sd(train.slope$slope))+
  geom_text(aes(10,61.09,label = 61.09, vjust = -1))+
  geom_hline(yintercept = mean(train.slope$slope)-
               2*sd(train.slope$slope))+
  geom_text(aes(10,17.44,label = 17.44, vjust = -1))+
  geom_hline(yintercept = mean(train.slope$slope)+
               4*sd(train.slope$slope))+
  geom_text(aes(10,82.92,label = 82.92, vjust = 1))+
  geom_hline(yintercept = mean(train.slope$slope)-
               4*sd(train.slope$slope))+
  geom_text(aes(10,-4.38,label = -4.38, vjust = -1))+
  theme_bw()+
  scale_color_brewer(palette = "Dark2")

# cplan
train.cplan <- data.frame(landslide.train$cplan,landslide.train$lslpts)
train.cplan <- train.cplan %>% mutate(rownames(train.cplan))
colnames(train.cplan) <- c("cplan","lslpts","ID")

train.cplan_plot<-ggplot(train.cplan,aes(x = ID, y = cplan, color = lslpts))+
  geom_point()+
  scale_x_discrete(breaks=seq(1, 100, by = 20))+
        geom_hline(yintercept = mean(train.cplan$cplan)+
               2*sd(train.cplan$cplan))+
  geom_hline(yintercept = mean(train.cplan$cplan)-
               2*sd(train.cplan$cplan))+
  geom_hline(yintercept = mean(train.cplan$cplan)+
               4*sd(train.cplan$cplan))+
  geom_hline(yintercept = mean(train.cplan$cplan)-
               4*sd(train.cplan$cplan))+
  geom_text(aes(10,0.012,label = 0.012, vjust = -2))+
  geom_text(aes(10,-0.13,label = -0.13, vjust = 1))+
  geom_text(aes(10,0.24,label = 0.24, vjust = 1))+
  geom_text(aes(10,-0.26,label = -0.26, vjust = 1))+
  theme_bw()+
  scale_color_brewer(palette = "Set2")

# cprof
train.cprof <- data.frame(landslide.train$cprof,landslide.train$lslpts)
train.cprof <- train.cprof %>% mutate(rownames(train.cprof))
colnames(train.cprof) <- c("cprof","lslpts","ID")

train.cprof_plot<-ggplot(train.cprof,aes(x = ID, y = cprof, color = lslpts))+
  geom_point()+
  scale_x_discrete(breaks=seq(1, 100, by = 20))+
      geom_hline(yintercept = mean(train.cprof$cprof)+
               2*sd(train.cprof$cprof))+
  geom_hline(yintercept = mean(train.cprof$cprof)-
             2*sd(train.cprof$cprof))+
  geom_hline(yintercept = mean(train.cprof$cprof)+
             4*sd(train.cprof$cprof))+
  geom_hline(yintercept = mean(train.cprof$cprof)-
             4*sd(train.cprof$cprof))+
  geom_text(aes(10,0.03,label = 0.03, vjust = -1))+
  geom_text(aes(10,-0.03,label = -0.03, vjust = -1))+
  geom_text(aes(10,0.06,label = 0.06, vjust = 1))+
  geom_text(aes(10,-0.06,label = -0.06, vjust = -1))+
  theme_bw()+
  scale_color_brewer(palette = "Set1")

# elev
train.elev <- data.frame(landslide.train$elev,landslide.train$lslpts)
train.elev <- train.elev %>% mutate(rownames(train.elev))
colnames(train.elev) <- c("elev","lslpts","ID")

train.elev_plot<-ggplot(train.elev,aes(x = ID, y = elev, color = lslpts))+
  geom_point()+
  scale_x_discrete(breaks=seq(1, 100, by = 20))+
    geom_hline(yintercept = mean(train.elev$elev)+
             2*sd(train.elev$elev))+
  geom_hline(yintercept = mean(train.elev$elev)-
             2*sd(train.elev$elev))+
  geom_hline(yintercept = mean(train.elev$elev)+
             4*sd(train.elev$elev))+
  geom_hline(yintercept = mean(train.elev$elev)-
             4*sd(train.elev$elev))+
  geom_text(aes(10,2914,label = 2914, vjust = -1))+
  geom_text(aes(10,1842,label = 1842, vjust = -1))+
  geom_text(aes(10,3450,label = 3450, vjust = 1))+
  geom_text(aes(10,1306,label = 1306, vjust = -1))+
  theme_bw()+
  scale_color_brewer(palette = "Accent")

# log10_carea
train.log10_carea <- data.frame(landslide.train$log10_carea,landslide.train$lslpts)
train.log10_carea <- train.log10_carea %>% mutate(rownames(train.log10_carea))
colnames(train.log10_carea) <- c("log10_carea","lslpts","ID")

train.log_plot<-ggplot(train.log10_carea,aes(x = ID, y = log10_carea, color = lslpts))+
  geom_point()+
  scale_x_discrete(breaks=seq(1, 100, by = 20))+
  geom_hline(yintercept = mean(train.log10_carea$log10_carea)+
             2*sd(train.log10_carea$log10_carea))+
  geom_hline(yintercept = mean(train.log10_carea$log10_carea)-
             2*sd(train.log10_carea$log10_carea))+
  geom_hline(yintercept = mean(train.log10_carea$log10_carea)+
             4*sd(train.log10_carea$log10_carea))+
  geom_hline(yintercept = mean(train.log10_carea$log10_carea)-
             4*sd(train.log10_carea$log10_carea))+
  geom_text(aes(10,3.79,label = 3.79, vjust = -1))+
  geom_text(aes(10,1.86,label = 1.86, vjust = -1))+
  geom_text(aes(10,4.76,label = 4.76, vjust = 1))+
  geom_text(aes(10,0.89,label = 0.89, vjust = -1))+
  theme_bw()+
  scale_color_brewer(palette = "Dark2")

train.slope_plot
```



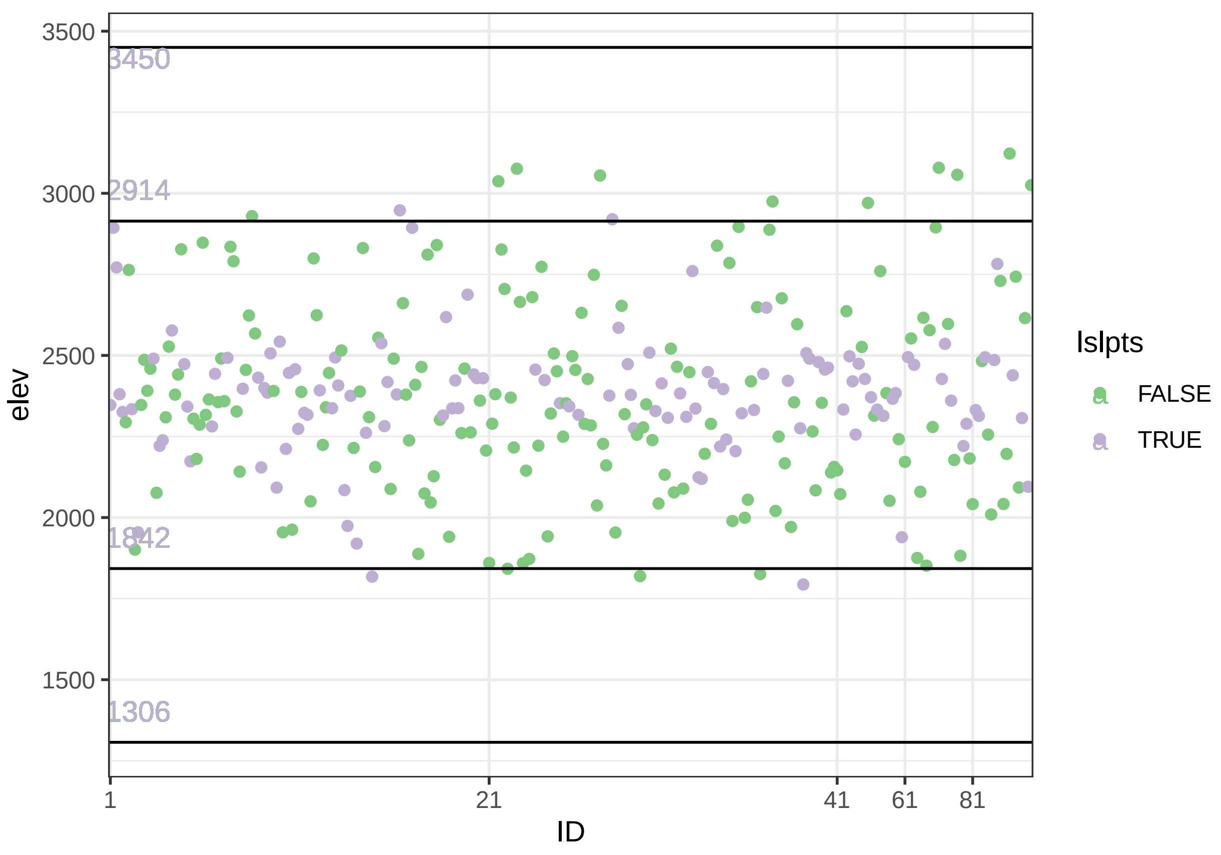
```
train.cplan_plot
```



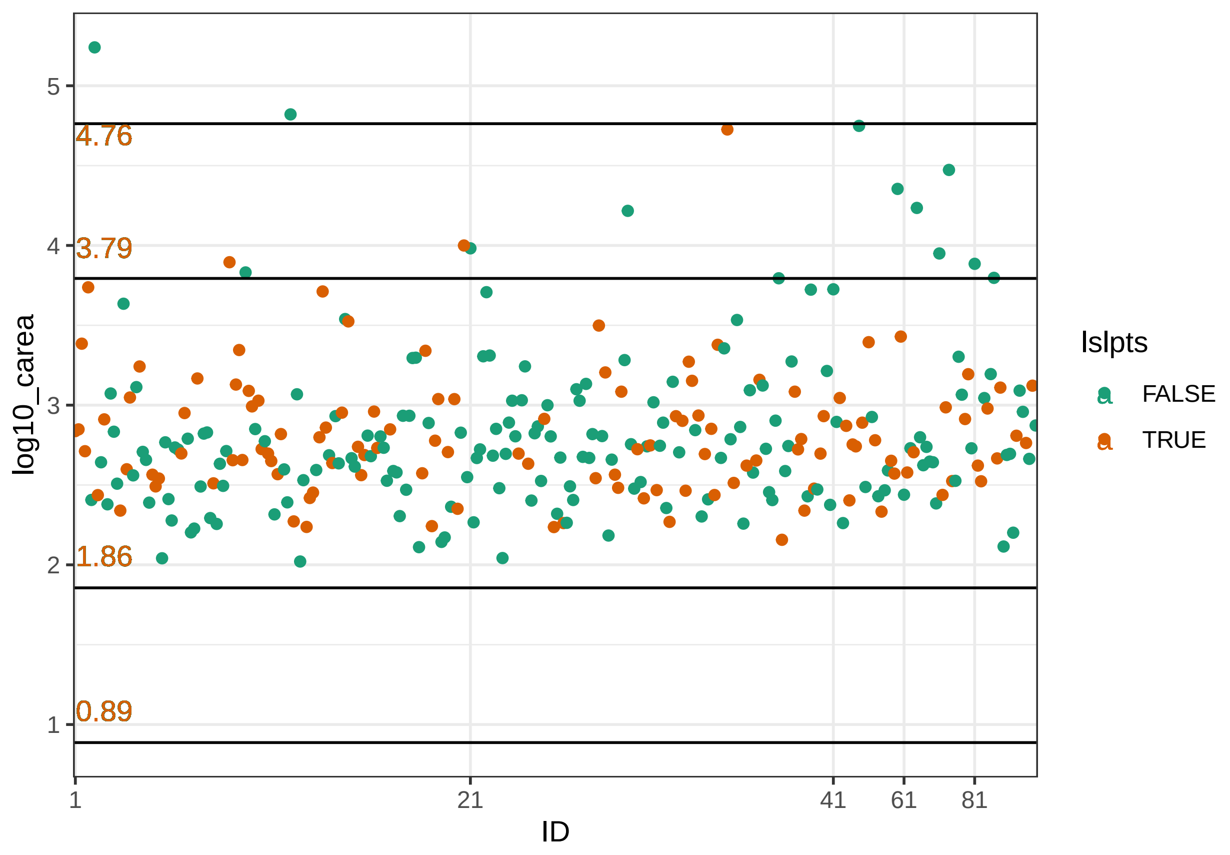
```
train.cprof_plot
```



```
train.elev_plot
```



```
train.log_plot
```



Comments:

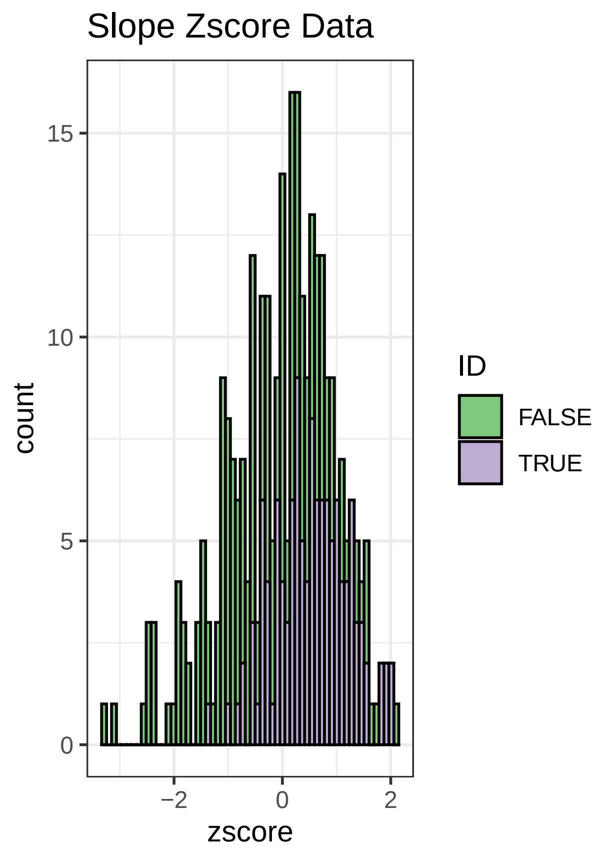
Here we firstly use two standard deviations from the mean which means that 95% values fall inside the of 2 standard deviations, and the values that fall outside are still part of the distribution, but they are unlikely or rare events – outliers.
Then we use four standard deviations from the mean which means that 99.9% values fall inside the of 4 standard deviations,and the values that fall outside are the outliers.
Inputation and capping are different approaches of treating outliers which the former one is replacing the outliers with the mean / median / mode while the later one is replacing the outliers that outside the lower limit with the value of 5th %Inter Quartile Range and those that lie above the upper limit, with the value of 95th %Inter Quartile Range.

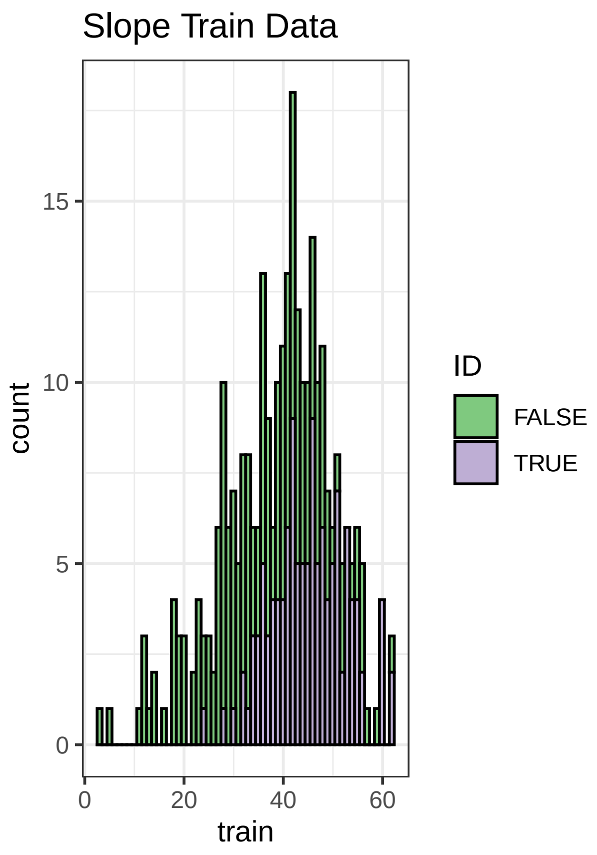
1.c. Normalization and Standardization

```
z_sd = function(x){
  z = c()
  for (i in 1:300) {
    z[i] = (x[i] - mean(x))/sd(x)
  }
  return(z)
}

z.slope <- data.frame(z_sd(landslide.train$slope))
```

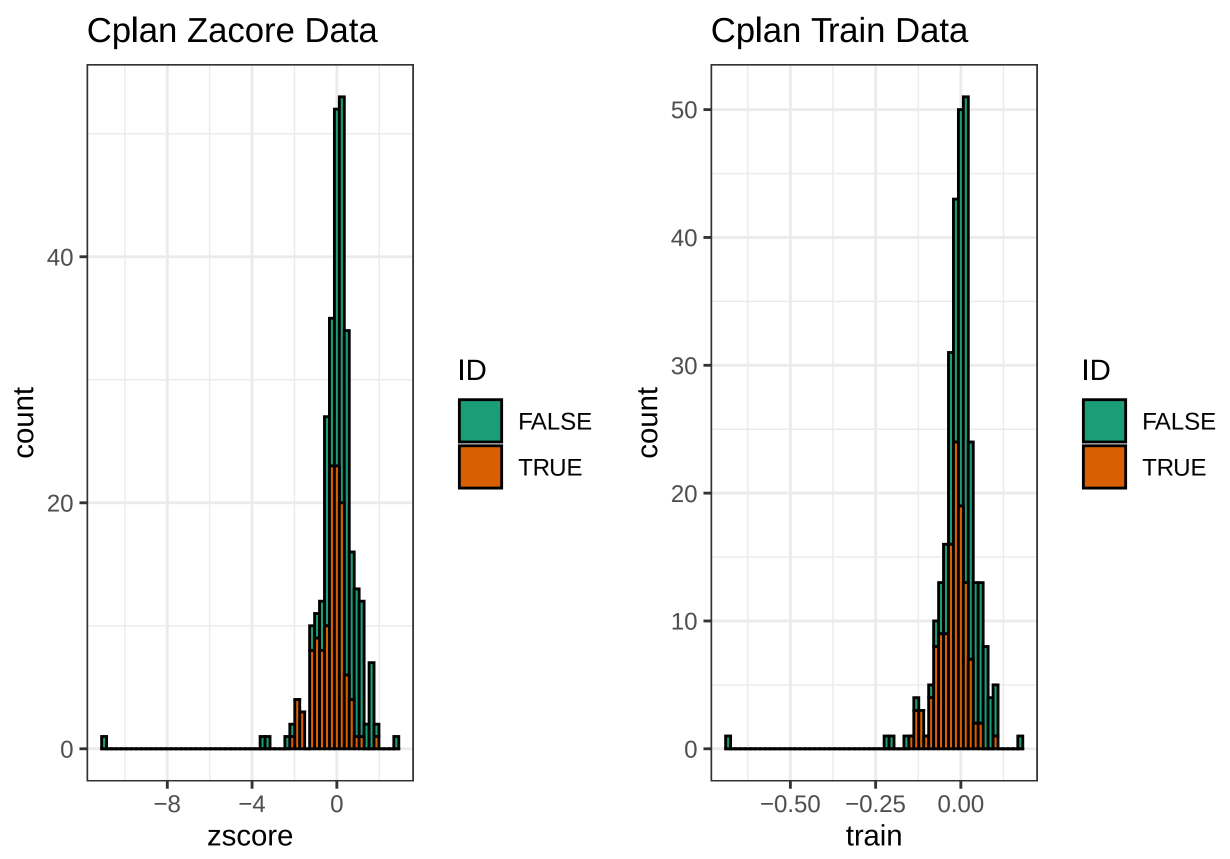
```r
z.cplan <- data.frame(z_sd(landslide.train$cplan))
z.cprof <- data.frame(z_sd(landslide.train$cprof))
z.elev <- data.frame(z_sd(landslide.train$elev))
z.log10_carea <- data.frame(z_sd(landslide.train$log10_carea))
landslide.train.standard <-z.slope %>%
  cbind(z.cplan) %>%
  cbind(z.cprof) %>%
  cbind(z.elev) %>%
  cbind(z.log10_carea)%>%
  cbind(landslide.train$lslpts)
colnames(landslide.train.standard) <-
  c("z.slope","z.cplan","z.cprof",
    "z.elev", "z.log10_carea","lslpts")
# tran.slope
train_slope <- data.frame(landslide.train$slope)

tran.slope <- train_slope %>%
  cbind(z.slope) %>%
  cbind(landslide.train$lslpts)
colnames(tran.slope) <- c("train","zscore","ID")

train.slope.plot <-
  ggplot(data=tran.slope, aes(x = train,fill = ID)) +
  geom_histogram(bins=60,colour = "black", position = "stack") +
  ggtitle("Slope Train Data")+
  scale_fill_brewer(palette = "Accent")+
  theme_bw()

zscore.slope.plot <-
  ggplot(data=tran.slope, aes(x = zscore,fill = ID)) +
  geom_histogram(bins=60,colour = "black", position = "stack") +
  ggtitle("Slope Zscore Data")+
  scale_fill_brewer(palette = "Accent")+
  theme_bw()

# tran.cplan
train_cplan <- data.frame(landslide.train$cplan)

tran.cplan <- train_cplan %>%
  cbind(z.cplan) %>%
  cbind(landslide.train$lslpts)

colnames(tran.cplan) <- c("train","zscore","ID")

train.cplan.plot <-ggplot(data=tran.cplan, aes(x = train,fill = ID)) +
  geom_histogram(bins=60,colour = "black", position = "stack") +
  ggtitle("Cplan Train Data") +
  scale_fill_brewer(palette = "Dark2")+
  theme_bw()

zscore.cplan.plot <-ggplot(data=tran.cplan, aes(x = zscore,fill = ID)) +
  geom_histogram(bins=60,colour = "black", position = "stack") +
  ggtitle("Cplan Zacore Data") +
  scale_fill_brewer(palette = "Dark2")+
  theme_bw()

# tran.cprof
train_cprof <- data.frame(landslide.train$cprof)

tran.cprof <- train_cprof %>%
  cbind(z.cprof) %>%
  cbind(landslide.train$lslpts)
colnames(tran.cprof) <- c("train","zscore","ID")

train.cprof.plot <-
  ggplot(data=tran.cprof, aes(x = train,fill = ID)) +
  geom_histogram(bins=40,colour = "black", position = "stack") +
  ggtitle("Cprof Zscore Data") +
  scale_fill_brewer(palette = "Pastel2")+
  theme_bw()

zscore.cprof.plot <-
  ggplot(data=tran.cprof, aes(x = zscore,fill = ID)) +
  geom_histogram(bins=40,colour = "black", position = "stack") +
  ggtitle("Cprof Train Data") +
  scale_fill_brewer(palette = "Pastel2")+
  theme_bw()

# tran.elev
train_elev <- data.frame(landslide.train$elev)

colnames(z.elev) <- c("zscore")
colnames(train_elev) <- c("train")

tran.elev <- train_elev %>%
  cbind(z.elev)%>%
  cbind(landslide.train$lslpts)
colnames(tran.elev) <- c("train","zscore","ID")

zscore.elev.plot <-
  ggplot(data=tran.elev, aes(x = zscore,fill = ID)) +
  geom_bar(colour = "black") +
  scale_x_binned()+
  ggtitle("Elev Zscore Data") +
  scale_fill_brewer(palette = "Set2")+
  theme_bw()

train.elev.plot <-
  ggplot(data=tran.elev, aes(x = train,fill = ID)) +
  geom_bar(colour = "black") +
  scale_x_binned()+
  ggtitle("Elev Train Data") +
  scale_fill_brewer(palette = "Set2")+
  theme_bw()

# tran.log10_carea
```

```r
train_log10_carea <- data.frame(landslide.train$log10_carea)

tran.log10_carea <- train_log10_carea %>%
  cbind(z.log10_carea) %>%
  cbind(landslide.train$lslpts)
colnames(tran.log10_carea) <- c("train","zscore","ID")

train.log10_carea.plot <-
  ggplot(data=tran.log10_carea, aes(x = train,fill = ID)) +
  geom_histogram(bins=40,colour = "black", position = "stack") +
  ggtitle("Log10_carea Train Data") +
  scale_fill_brewer(palette = "Set1")+
  theme_bw()

zscore.log10_carea.plot <-
  ggplot(data=tran.log10_carea, aes(x = train,fill = ID)) +
  geom_histogram(bins=40,colour = "black", position = "stack") +
  ggtitle("Log10_carea Zscore Data") +
  scale_fill_brewer(palette = "Set1")+
  theme_bw()


# correlation
cor_z_data <- z.slope %>%
  cbind(z.cplan) %>%
  cbind(z.cprof) %>%
  cbind(z.elev) %>%
  cbind(z.log10_carea)
colnames(cor_z_data) <-
  c("z.slope","z.cplan","z.cprof",
    "z.elev", "z.log10_carea")
cor_z_plot<- ggcorr(cor_z_data,
                    label = TRUE,label_size = 3, hjust = 0.75, size = 5, color = "grey40",
       low = "#3399FF", mid = "#FFFF66", high = "#CC0033",
       method = c("pairwise", "spearman"))+
  theme(legend.title = element_text(size = 11))


grid.arrange(zscore.slope.plot,train.slope.plot,ncol=2)
```

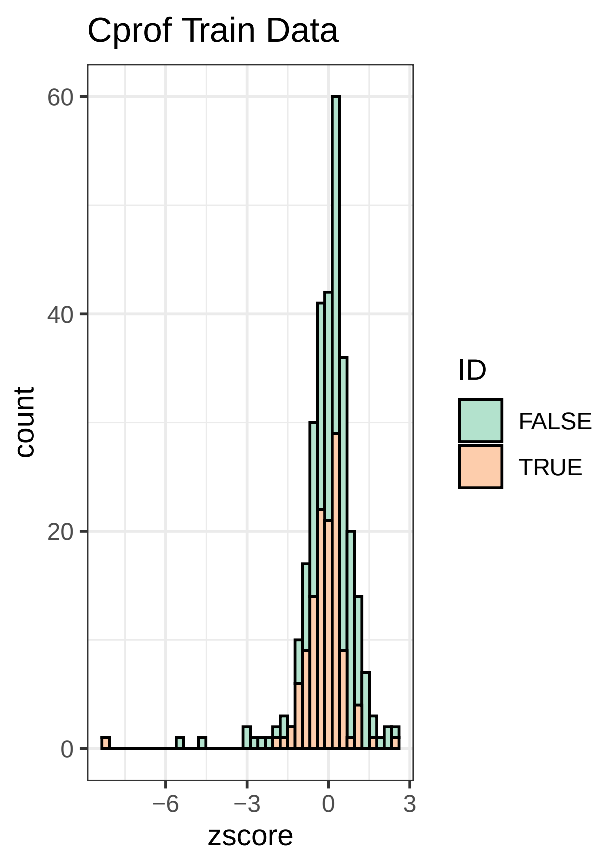


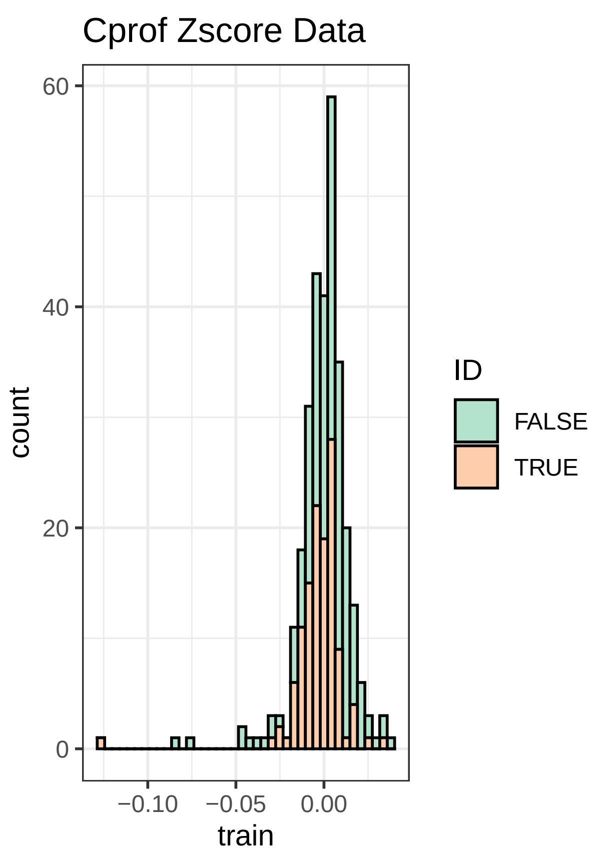


```r
grid.arrange(zscore.cplan.plot,train.cplan.plot,ncol=2)
```

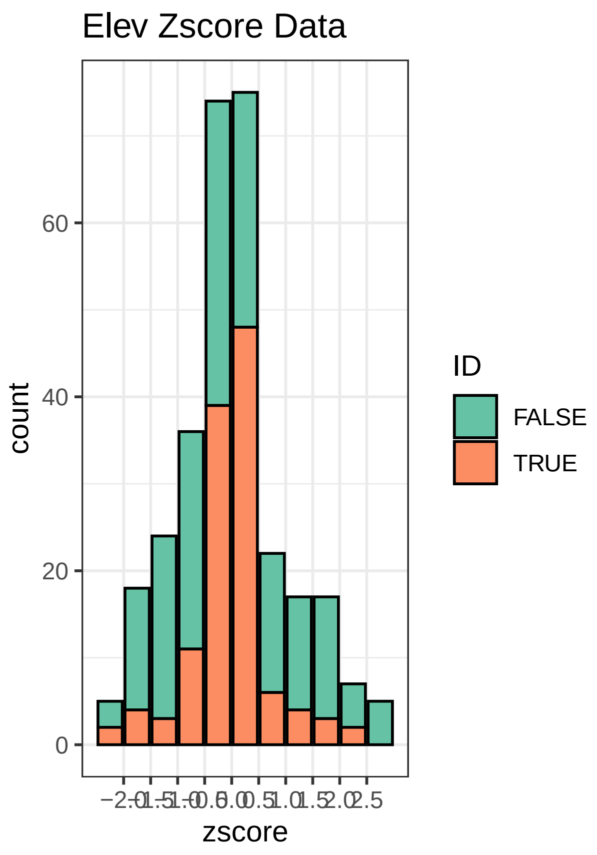


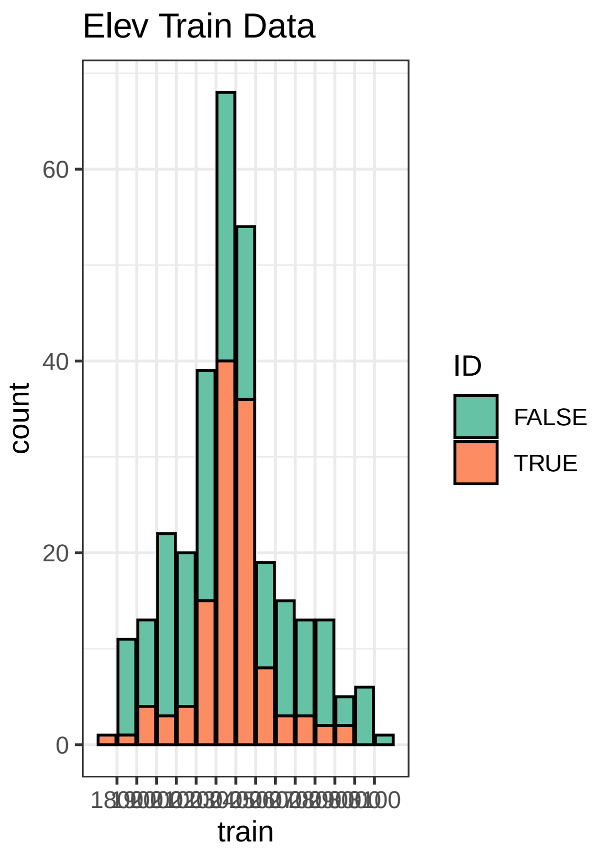
```r
grid.arrange(zscore.cprof.plot,train.cprof.plot,ncol=2)
```



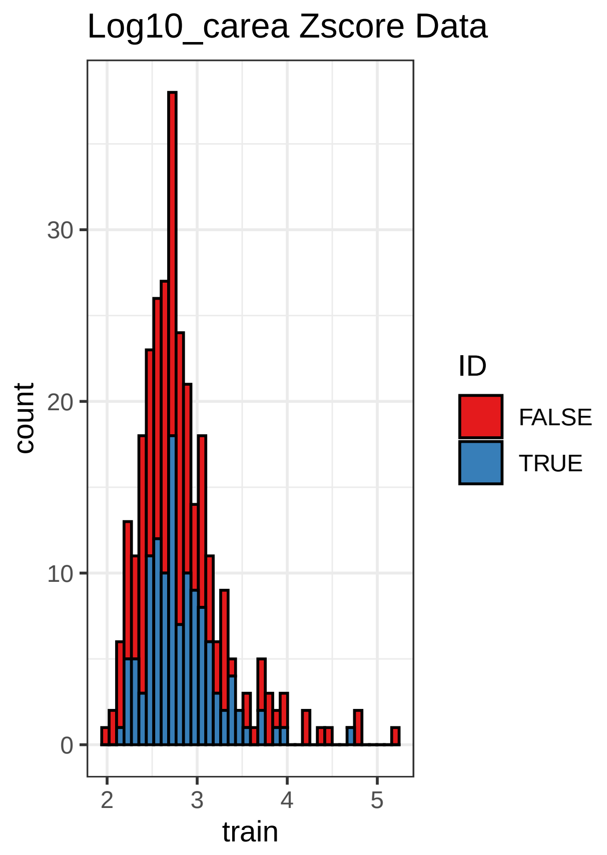


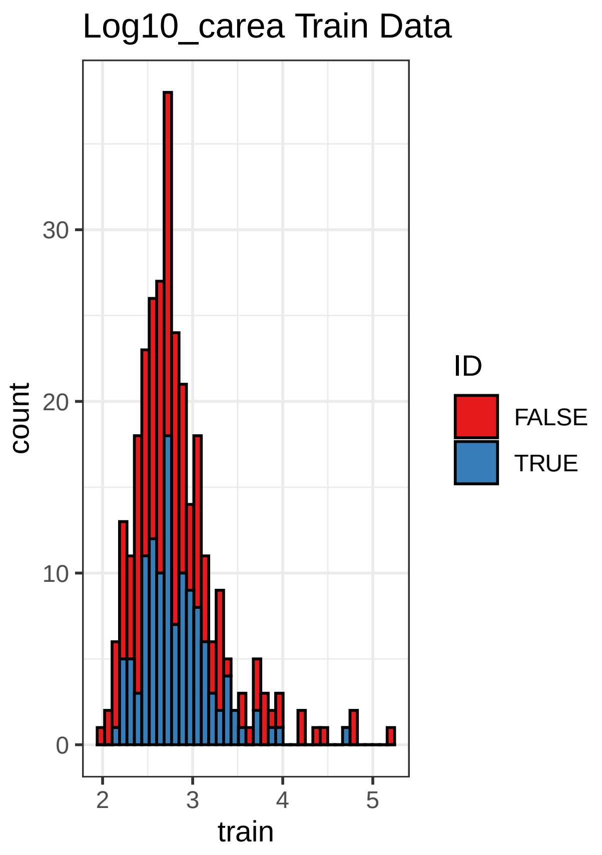
```r
grid.arrange(zscore.elev.plot,train.elev.plot,ncol=2)
```



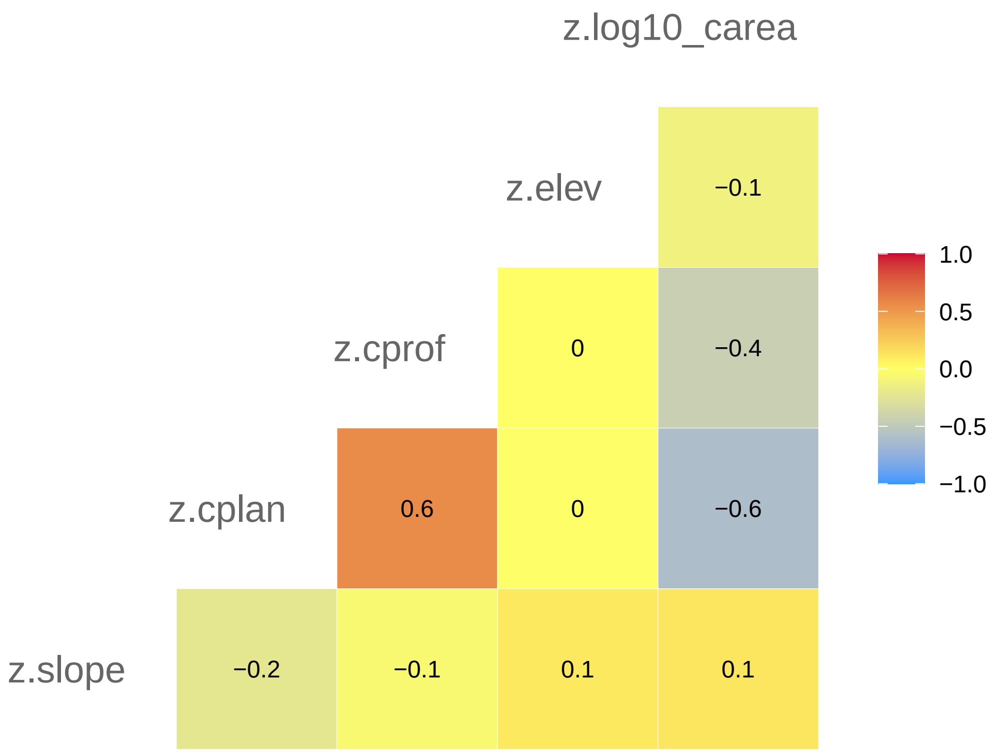


```r
grid.arrange(zscore.log10_carea.plot,train.log10_carea.plot,ncol=2)
```


Log10_carea Zscore Data
count
train
ID
FALSE
TRUE


Log10_carea Train Data
count
train
ID
FALSE
TRUE

```
cor_z_plot
```



Comments:

The data are transformed by Z score standardization which rescales and centers the data so that it has mean value of zero and standard deviation of one.The standardized data represent the distance between that raw data and the population mean in units of the standard deviation. The correlation plot here is the same as the un-normalized one since the correlation of the z-scores is the covariance of the z-scores of the z-scores, which is the covariance of the z-scores, which is just the correlation of the original scores.

2.Model Development - Logistic Regression Prediction

2.a. Original data

```
fit.model.train =glm(formula = lslpts ~ slope +
                       cplan + cprof + elev + log10_carea,
                     family = binomial('logit'), data = landslide.train)
options(scipen = 999)
summary(fit.model.train)
```

```
##
## Call:
## glm(formula = lslpts ~ slope + cplan + cprof + elev + log10_carea,
##     family = binomial("logit"), data = landslide.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6913  -0.7950  -0.3542   0.8372   2.1973
##
## Coefficients:
##                Estimate  Std. Error z value         Pr(>|z|)
## (Intercept)   1.4683397   2.0709912   0.709          0.47832
## slope         0.1206115   0.0176921   6.817 0.00000000000928 ***
## cplan       -17.3536836   4.1864977  -4.145 0.00003395832741 ***
## cprof       -30.1095279  10.6154965  -2.836          0.00456 **
## elev         -0.0007206   0.0005725  -1.259          0.20816
## log10_carea  -1.8460903   0.4716705  -3.914 0.00009080195073 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 405.37  on 299  degrees of freedom
## Residual deviance: 297.89  on 294  degrees of freedom
## AIC: 309.89
##
## Number of Fisher Scoring iterations: 5
```

```
exp(fit.model.train$coefficients)
```

```
##            (Intercept)                  slope                  cplan
## 4.34202010476675326345 1.12818657851853343388 0.00000002906638310999
##                  cprof                   elev            log10_carea
## 0.00000000000008386837 0.99927965706515098354 0.15785312757486893998
```

```
RMSE = function (error) {
  sqrt(mean(error^2)) }
RMSE(fit.model.train$residuals)
```

```
## [1] 3.556733
```

Comments:

The coefficient for slope = 0.1206115, cplan = -17.3536836, cprof = -30.109527, elev = -0.0007206, log10_carea = 1.8460903 which can be interpreted as the *expected change in log odds* **for a one-unit increase in the** *slope degree/plan curvature/profile curvature/elevation/the decadic logarithm of the catchment area(normalized).*
The odds ratio can be calculated by exponentiating these values respectively to get about *1.128, 0.000000029, 0.00000000000083, 0.999, 0.1579* **which means we expect to see**
1.For every unit change in slope degree, the odds of landslides increase by 1.128,
2.For every unit change in plan curvature, the odds of landslides increase by 0.2537602,
3.For every unit change in profile curvature, the odds of landslides increase by 0.000000029,
4.For every unit change in elevation, the odds of landslides increase by 0.999,
5.For every unit change in the decadic logarithm of the catchment area, the odds of landslides increase by 0.1579.
odds = probability of event occurence/(1-probability of event occurence)

2.b. Standardized data

```r
fit.model.train.std =glm(formula = lslpts ~ z.slope +
                       z.cplan + z.cprof + z.elev + z.log10_carea,
                     family = binomial('logit'), data = landslide.train.standard)

RMSE = function (error) {
  sqrt(mean(error^2)) }

summary(fit.model.train.std)
```

```
##
## Call:
## glm(formula = lslpts ~ z.slope + z.cplan + z.cprof + z.elev +
##     z.log10_carea, family = binomial("logit"), data = landslide.train.standard)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6913  -0.7950  -0.3542   0.8372   2.1973
##
## Coefficients:
##               Estimate Std. Error z value         Pr(>|z|)
## (Intercept)    -0.5865     0.1497  -3.919 0.00008887881111 ***
## z.slope         1.3163     0.1931   6.817 0.00000000000928 ***
## z.cplan        -1.0787     0.2602  -4.145 0.00003395832741 ***
## z.cprof        -0.4639     0.1636  -2.836          0.00456 **
## z.elev         -0.1930     0.1534  -1.259          0.20816
## z.log10_carea  -0.8943     0.2285  -3.914 0.00009080195073 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 405.37  on 299  degrees of freedom
## Residual deviance: 297.89  on 294  degrees of freedom
## AIC: 309.89
##
## Number of Fisher Scoring iterations: 5
```

```r
RMSE(fit.model.train.std$residuals)
```

```
## [1] 3.556733
```

```r
exp(fit.model.train.std$coefficients)
```

```
##   (Intercept)       z.slope       z.cplan       z.cprof        z.elev
##     0.5562463     3.7294812     0.3400519     0.6287955     0.8244431
## z.log10_carea
##     0.4088834
```

Comments:

The coefficient for z.slope = 1.3163, z.plan = -1.0787, z.cprof = -0.4639, z.elev = -0.193, z.log10__carea = -0.8943 which can be interpreted as the *expected change in log odds* **for a one-unit increase in the** *standardized slope degree/plan curvature/profile curvature/elevation/the decadic logarithm of the catchment area(normalized).*
The odds ratio can be calculated by exponentiating these values respectively to get *3.729596, 0.3400519, 0.6287955, 0.8244431, 0.4088834* **which means we expect to see**

1.For every unit change in standardized slope degree, the odds of landslides increase by about 372.95%,

2.For every unit change in standardized plan curvature, the odds of landslides increase by 25.37%,

3.For every unit change in standardized profile curvature, the odds of landslides increase by 38.60%,

4.For every unit change in standardized elevation, the odds of landslides increase by 45.18%,

5.For every unit change in the standardized decadic logarithm of the catchment area, the odds of landslides increase by 29.02%.

Compare the model fitted with origincal data and the model fitted with standardized data, the RMSE for them are the same.

3.Model Prediction

- Use the logistic model (the one fit to non-normalized data) to predict outcomes on the landslide.validation data
- Develop a confusion table for landslide occurance, using a threshold of 0.5

```r
pred.landslide=predict(fit.model.train,newdata=landslide.validation, type="response")

thr.val <- ifelse(pred.landslide >= 0.5, 1, 0)

confusion.table <- table(factor(t(thr.val),
             levels =min(landslide.validation$lslpts):max(landslide.validation$lslpts)),
      factor(as.numeric(landslide.validation$lslpts),
             levels=min(landslide.validation$lslpts):max(landslide.validation$lslpts)))

pred.landslide
```

```
##           1           2           3           4           5           6
## 0.015205399 0.642192830 0.626146973 0.847935472 0.128946606 0.190975074
##           7           8           9          10          11          12
## 0.004192568 0.132859911 0.914665894 0.470010157 0.301845507 0.821955367
##          13          14          15          16          17          18
## 0.212911296 0.097767229 0.894608530 0.094267817 0.405685660 0.771971407
##          19          20          21          22          23          24
## 0.082682016 0.461349120 0.809469486 0.479803394 0.087846559 0.831491446
##          25          26          27          28          29          30
## 0.245039974 0.825879134 0.667556821 0.327284088 0.517999876 0.014121483
##          31          32          33          34          35          36
## 0.685109275 0.022388021 0.104502309 0.008182808 0.359755950 0.076519496
##          37          38          39          40          41          42
## 0.637386455 0.053120822 0.726267236 0.673784893 0.051385448 0.296191044
##          43          44          45          46          47          48
## 0.705789069 0.519445405 0.675967340 0.337959417 0.379665044 0.890869448
##          49          50          51          52          53          54
## 0.765333394 0.457026871 0.136608703 0.587535400 0.585567188 0.767548423
##          55          56          57          58          59          60
## 0.359755950 0.727064855 0.670576397 0.220196019 0.807006303 0.742312118
##          61          62          63          64          65          66
## 0.748700399 0.359928023 0.057629760 0.083036269 0.212143107 0.847956137
##          67          68          69          70          71          72
## 0.301845507 0.132598708 0.481907707 0.725289039 0.310322716 0.442447807
##          73          74          75          76          77          78
## 0.958397958 0.799189864 0.432176386 0.048115719 0.890925017 0.246257746
##          79          80          81          82          83          84
## 0.858929467 0.398813265 0.618705078 0.521543955 0.361152568 0.089118054
##          85          86          87          88          89          90
## 0.284202556 0.802651580 0.771224586 0.237831971 0.658221510 0.020316410
##          91          92          93          94          95          96
## 0.380753338 0.874128100 0.461909958 0.846559405 0.339320176 0.834511311
##          97          98          99         100
## 0.060001336 0.561791709 0.641563173 0.707668405
```

```
confusion.table
```

```
##
##       0  1
##   0  43 11
##   1   8 38
```

```
print("Model accuracy = correct prediction/all prediction = 81%")
```

```
## [1] "Model accuracy = correct prediction/all prediction = 81%"
```

4.Model Selection

- Trial an alternative model, in this case the logistic model with fewer predictor variables has been used since sometimes the model might be overfitting the training data
- Fit the alternative new model to the training data set and predict the response on the validation data-set

```
new.model.train <- data.frame(landslide.train$slope) %>%
  cbind(landslide.train$cplan) %>%
  cbind(landslide.train$cprof) %>%
  cbind(landslide.train$lslpts)
```

```r
colnames(new.model.train) <- c("slope","cplan","cprof","lslpts")

fit.new.model.train = glm(formula = lslpts ~ slope + cplan + cprof,
                          family = binomial('logit'),
                          data = new.model.train)
summary(fit.new.model.train)
```

```
##
## Call:
## glm(formula = lslpts ~ slope + cplan + cprof, family = binomial("logit"),
##     data = new.model.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2447  -0.8825  -0.3564   0.9067   2.1837
##
## Coefficients:
##              Estimate Std. Error z value          Pr(>|z|)
## (Intercept)  -5.45577    0.74988  -7.276 0.000000000000345 ***
## slope         0.12198    0.01746   6.986 0.000000000002828 ***
## cplan        -8.59963    2.88629  -2.979           0.00289 **
## cprof       -16.91731   10.07036  -1.680           0.09297 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 405.37  on 299  degrees of freedom
## Residual deviance: 316.37  on 296  degrees of freedom
## AIC: 324.37
##
## Number of Fisher Scoring iterations: 5
```

```r
new.model.validation <- data.frame(landslide.validation$slope) %>%
  cbind(landslide.validation$cplan) %>%
  cbind(landslide.validation$cprof) %>%
  cbind(landslide.validation$lslpts)
colnames(new.model.validation) <- c("slope","cplan",
                                    "cprof","lslpts")

pred.landslide.new = predict(fit.new.model.train, newdata=new.model.validation, type="response")


thr.val.new <- ifelse(pred.landslide.new >= 0.5, 1, 0)

table(factor(t(thr.val.new),
             levels =min(new.model.validation$lslpts):
               max(new.model.validation$lslpts)),
      factor(as.numeric(new.model.validation$lslpts),
             levels=min(new.model.validation$lslpts):
               max(new.model.validation$lslpts)))
```

```
##
```

```
##      0  1
##   0 42 15
##   1  9 34
```

```
print("Model accuracy = correct prediction/all prediction = 76%")
```

```
## [1] "Model accuracy = correct prediction/all prediction = 76%"
```

Comments:

Compared to the previous logistic model using more predictor variables, whose model accuracy is 81%, model precision is 77.5%, model recall is 82.6% and model F1-score is 90.5%, the new alternative model's accuracy, precision, recall and F1-score performances are 76%, 69.39%, 79% and 88.24% respectively, which are all worse than the model from previous problem.

5.Model Evaluation

5.a. Model skill metrics

```
fit.new.model.train = glm(formula = lslpts ~ slope + cplan + cprof,
                          family = binomial('logit'),
                          data = new.model.train)
pred.landslide.new = predict(fit.new.model.train, newdata=new.model.validation, type="response")

thr.val.new <- ifelse(pred.landslide.new >= 0.5, 1, 0)

table(factor(t(thr.val.new),
             levels =min(new.model.validation$lslpts):
               max(new.model.validation$lslpts)),
      factor(as.numeric(new.model.validation$lslpts),
             levels=min(new.model.validation$lslpts):
               max(new.model.validation$lslpts)))
```

```
##
##      0  1
##   0 42 15
##   1  9 34
```

```
print("Model accuracy = correct prediction/all prediction = 76%")
```

```
## [1] "Model accuracy = correct prediction/all prediction = 76%"
```

```
print("Model precision = true positives/(true positives + false positives) = 69.39%")
```

```
## [1] "Model precision = true positives/(true positives + false positives) = 69.39%"
```

```r
print("Model recall = true positives/(true positives + false negatives) = 79%")
```

```
## [1] "Model recall = true positives/(true positives + false negatives) = 79%"
```

```r
print("Model F1-score = (1+1^2)*(precision * recall)/((1^2 * precision) + recall) = 88.24% ")
```

```
## [1] "Model F1-score = (1+1^2)*(precision * recall)/((1^2 * precision) + recall) = 88.24% "
```

5.b. Brier Score

- The Brier Score is useful for assessing model performance when predictions are probabilistic.

```r
pred.new <-data.frame(pred.landslide.new)
val.new <- data.frame(as.numeric(new.model.validation$lslpts))

B_Score = function(){
  Ts = c()
  for (i in 1:100) {
    Ts[i] = ((pred.new[i,]- val.new[i,]))^2
  }
  Bs = sum(Ts)/100
  return(Bs)
}
B_Score()
```

```
## [1] 0.1682691
```

Comments:

Although smaller scores (closer to zero) in Brier score indicate better forecasts, when model scores in the middle (e.g. 0.44, 0.69) it can be hard to interpret as "good" or "bad". Also, The Brier score becomes inadequate for very rare (or very frequent) events, because it does not sufficiently discriminate between small changes in forecast that are significant for rare events. But, apart from its limitation, the Brier score is still an efficient measure of model skills for our probability forcast model with binary outcomes.

5.c. ROC and AUC

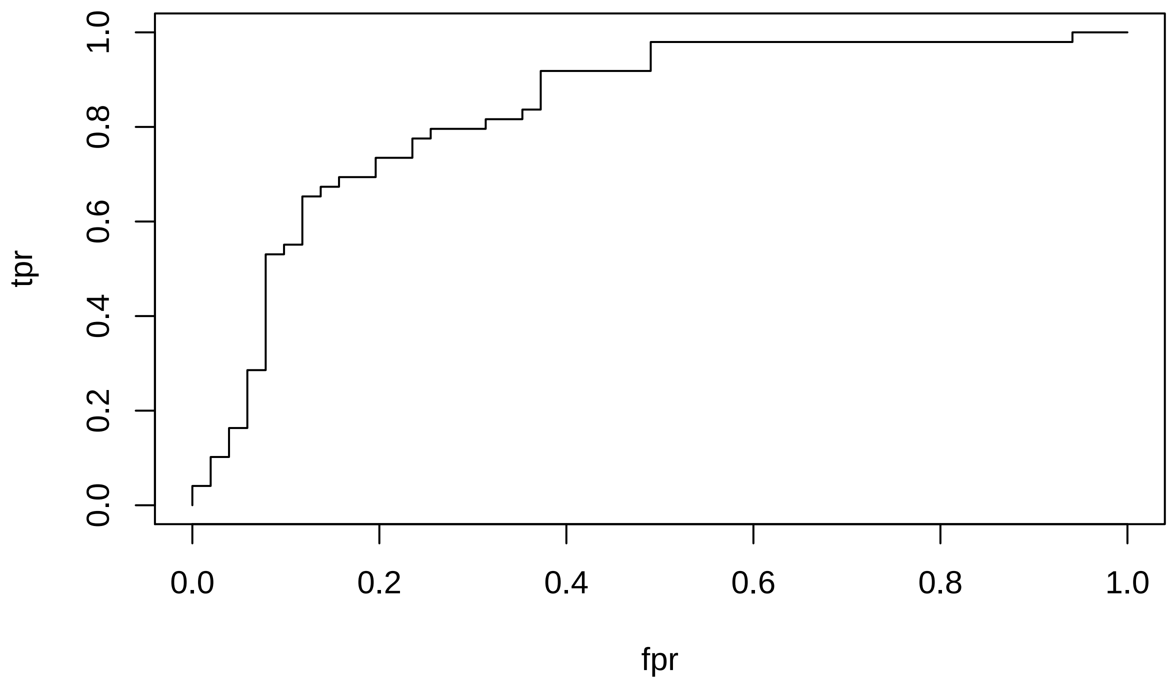
- The ROC curve (receiver operating characteristic curve) is a plot showing the performance of the classification model at all classification thresholds. It is the plot of True Positive Rate (TPR) vs False Positive Rate (FPR) for all thresholds.
- Based on the ROC curve, the Area Under the ROC Curve (AUC) can be calculated. It is a single value which provides an aggregate measure of the performance of the classification model accross all possible classification threshold.

```r
fit.new.model.train = glm(formula = lslpts ~ slope + cplan + cprof,
                          family = binomial('logit'),
                          data = new.model.train)
```

```r
pred.landslide.new = predict(fit.new.model.train,
                             newdata=new.model.validation,
                             type="response")
t = seq(0, 1, by = 0.01)

new.Roc <-roc(as.numeric(new.model.validation$lslpts),pred.landslide.new,quiet = TRUE)

coords(new.Roc, x = t, input="threshold",
       ret = c("threshold", "se", "1-sp"),
       transpose = TRUE)
```

```
##                  [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## threshold           0 0.0100000 0.0200000 0.0300000 0.0400000 0.0500000 0.0600000
## sensitivity         1 1.0000000 1.0000000 0.9795918 0.9795918 0.9795918 0.9795918
## 1-specificity       1 0.9607843 0.9411765 0.9215686 0.9019608 0.9019608 0.8627451
##                    [,8]      [,9]     [,10]     [,11]     [,12]     [,13]
## threshold     0.0700000 0.0800000 0.0900000 0.1000000 0.1100000 0.1200000
## sensitivity   0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918
## 1-specificity 0.8431373 0.8039216 0.7647059 0.7647059 0.7450980 0.6862745
##                   [,14]     [,15]     [,16]     [,17]     [,18]     [,19]
## threshold     0.1300000 0.1400000 0.1500000 0.1600000 0.1700000 0.1800000
## sensitivity   0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918
## 1-specificity 0.6470588 0.6470588 0.6470588 0.6274510 0.6274510 0.5882353
##                   [,20]     [,21]     [,22]     [,23]     [,24]     [,25]
## threshold     0.1900000 0.2000000 0.2100000 0.2200000 0.2300000 0.2400000
## sensitivity   0.9795918 0.9795918 0.9795918 0.9591837 0.9591837 0.9591837
## 1-specificity 0.5490196 0.5490196 0.5098039 0.4901961 0.4901961 0.4901961
##                   [,26]     [,27]     [,28]     [,29]     [,30]     [,31]
## threshold     0.2500000 0.2600000 0.2700000 0.2800000 0.2900000 0.3000000
## sensitivity   0.9183673 0.9183673 0.9183673 0.9183673 0.9183673 0.9183673
## 1-specificity 0.4901961 0.4509804 0.4313725 0.4313725 0.4313725 0.4313725
##                   [,32]     [,33]     [,34]     [,35]     [,36]     [,37]
## threshold     0.3100000 0.3200000 0.3300000 0.3400000 0.3500000 0.3600000
## sensitivity   0.9183673 0.9183673 0.8571429 0.8367347 0.8367347 0.8163265
## 1-specificity 0.3921569 0.3725490 0.3725490 0.3529412 0.3529412 0.3529412
##                   [,38]     [,39]     [,40]     [,41]     [,42]     [,43]
## threshold     0.3700000 0.3800000 0.3900000 0.4000000 0.4100000 0.4200000
## sensitivity   0.8163265 0.8163265 0.7959184 0.7959184 0.7959184 0.7959184
## 1-specificity 0.3333333 0.3137255 0.2745098 0.2745098 0.2745098 0.2549020
##                   [,44]     [,45]     [,46]     [,47]     [,48]     [,49]
## threshold     0.4300000 0.4400000 0.4500000 0.4600000 0.4700000 0.4800000
## sensitivity   0.7755102 0.7551020 0.7346939 0.7346939 0.7346939 0.7346939
## 1-specificity 0.2352941 0.2352941 0.2352941 0.2156863 0.2156863 0.1960784
##                   [,50]     [,51]     [,52]     [,53]     [,54]     [,55]
## threshold     0.4900000 0.5000000 0.5100000 0.5200000 0.5300000 0.5400000
## sensitivity   0.6938776 0.6938776 0.6734694 0.6734694 0.6734694 0.6530612
## 1-specificity 0.1764706 0.1764706 0.1568627 0.1568627 0.1372549 0.1372549
##                   [,56]     [,57]     [,58]     [,59]     [,60]     [,61]
## threshold     0.5500000 0.5600000 0.5700000 0.5800000 0.5900000 0.6000000
## sensitivity   0.6530612 0.6326531 0.6326531 0.6122449 0.6122449 0.6122449
## 1-specificity 0.1176471 0.1176471 0.1176471 0.1176471 0.1176471 0.1176471
##                   [,62]     [,63]     [,64]      [,65]      [,66]      [,67]
## threshold     0.6100000 0.6200000 0.6300000 0.64000000 0.65000000 0.66000000
## sensitivity   0.5918367 0.5510204 0.5510204 0.55102041 0.51020408 0.46938776
## 1-specificity 0.1176471 0.1176471 0.1176471 0.09803922 0.07843137 0.07843137
##                    [,68]      [,69]      [,70]      [,71]      [,72]      [,73]
## threshold     0.67000000 0.68000000 0.69000000 0.70000000 0.71000000 0.72000000
## sensitivity   0.40816327 0.38775510 0.34693878 0.32653061 0.32653061 0.28571429
## 1-specificity 0.07843137 0.07843137 0.07843137 0.07843137 0.07843137 0.07843137
##                    [,74]      [,75]      [,76]      [,77]      [,78]      [,79]
## threshold     0.73000000 0.74000000 0.75000000 0.76000000 0.77000000 0.78000000
## sensitivity   0.26530612 0.24489796 0.20408163 0.16326531 0.16326531 0.16326531
## 1-specificity 0.05882353 0.05882353 0.05882353 0.05882353 0.05882353 0.03921569
##                    [,80]      [,81]      [,82]      [,83]      [,84]      [,85]
## threshold     0.79000000 0.80000000 0.81000000 0.82000000 0.83000000 0.84000000
## sensitivity   0.14285714 0.10204082 0.08163265 0.06122449 0.06122449 0.04081633
## 1-specificity 0.03921569 0.03921569 0.01960784 0.01960784 0.01960784 0.01960784
##                    [,86]      [,87]      [,88]      [,89] [,90] [,91] [,92]
## threshold     0.85000000 0.86000000 0.87000000 0.88000000  0.89   0.9  0.91
## sensitivity   0.04081633 0.04081633 0.04081633 0.02040816  0.00   0.0  0.00
## 1-specificity 0.00000000 0.00000000 0.00000000 0.00000000  0.00   0.0  0.00
##               [,93] [,94] [,95] [,96] [,97] [,98] [,99] [,100] [,101]
## threshold      0.92  0.93  0.94  0.95  0.96  0.97  0.98   0.99      1
## sensitivity    0.00  0.00  0.00  0.00  0.00  0.00  0.00   0.00      0
## 1-specificity  0.00  0.00  0.00  0.00  0.00  0.00  0.00   0.00      0
```

```r
plot(tpr ~ fpr,
     coords(new.Roc, "all", ret = c("tpr", "fpr"), transpose = FALSE,),
     type="l")
```



```
auc(new.Roc)
```

```
## Area under the curve: 0.8339
```

Comments:

As a thumb rule, we have an excellent classifier if AUC is >=0.9 and a good classifier when it's >= 0.8.
The AUC of my new model is 0.83, it means there is 83% chance that the model will be able to distinguish between positives class and negative class, and it is a good classifier.