```
> data<-read.csv("~/Downloads/BSFC.csv")
> drop=c("Name")
> data = data[,!names(data) %in% drop]
> names(data)
 [1] "subsumption"                                "specific"
 [3] "recover_exception"                          "actions_abort_percentage"
 [5] "actions_continue_percentage"                "actions_empty_percentage"
 [7] "actions_log_percentage"                     "this.actions_method_percentage"
 [9] "actions_nestedtry_percentage"               "actions_return_percentage"
[11] "actions_throwcurrent_percentage"            "actions_thrownew_percentage"
[13] "actions_throwwrap_percentage"               "actions_todo_percentage"
[15] "subsumption_percentage"                     "specific_percentage"
[17] "overcatch_percentage"                       "overcatch_abort_percentage"
[19] "catch_donothing_percentage"                 "catchgeneric_percentage"
[21] "destructivewrapping_percentage"             "dummy_handler_percentage"
[23] "ignoring_interrupted_exception_percentage"  "incomplete_implementation_percentage"
[25] "log_returnnull_percentage"                  "log_throw_percentage"
[27] "multiple_line_log_percentage"               "throw_in_finally_percentage"
[29] "Number.of.Catch.Blocks"                     "Number.of.Catch.Block.LOC"
[31] "Number.of.Catch.Block.SLOC"                 "Number.of.Invoked.methods"
[33] "Number.of.Try.Blocks"                       "Number.of.Try.Block.LOC"
[35] "Number.of.Try.Block.SLOC"                   "Number.of.Throws.Generic.AP"
[37] "Number.of.Throws.Kitchen.Sink.AP"           "Number.of.Try.in.Declaration"
[39] "Number.of.Try.in.Condition"                 "Number.of.Try.in.EH"
[41] "Number.of.Try.in.Other"                     "Action.Continue"
[43] "Action.Default"                             "Action.Empty"
[45] "Action.Log"                                 "Action.Method"
[47] "Action.NestedTry"                           "Action.Return"
[49] "Code_Ownership_count"                       "lines_added"
[51] "lines_deleted"                              "total_prerel_change"
[53] "post_release_defects"                       "pre_release_defects"
[55] "AvgCyclomatic"                              "AvgCyclomaticModified"
[57] "AvgCyclomaticStrict"                        "AvgEssential"
[59] "AvgLine"                                    "AvgLineBlank"
[61] "AvgLineCode"                                "AvgLineComment"
[63] "CountDeclClass"                             "CountDeclClassMethod"
[65] "CountDeclClassVariable"                     "CountDeclExecutableUnit"
[67] "CountDeclFunction"                          "CountDeclInstanceMethod"
[69] "CountDeclInstanceVariable"                  "CountDeclMethod"
[71] "CountDeclMethodDefault"                     "CountDeclMethodPrivate"
[73] "CountDeclMethodProtected"                   "CountDeclMethodPublic"
[75] "CountLine"                                  "CountLineBlank"
[77] "CountLineCode"                              "CountLineCodeDecl"
[79] "CountLineCodeExe"                           "CountLineComment"
[81] "CountStmt"                                  "CountStmtDecl"
[83] "CountStmtExe"                               "MaxCyclomatic"
[85] "MaxCyclomaticModified"                      "MaxCyclomaticStrict"
[87] "MaxEssential"                               "MaxNesting"
[89] "RatioCommentToCode"                         "SumCyclomatic"
[91] "SumCyclomaticModified"                      "SumCyclomaticStrict"
[93] "SumEssential"
>
> drop=c("post_release_defects")
> independant=data[,!(names(data) %in% drop)]
> correlations <- cor(independant, method="spearman")
> highCorr <- findCorrelation(correlations, cutoff = .75)
> highCorr
 [1] 78 82 76 80 91 89 90 81 84 83 74 85 33 29  3 87 77 92 32 75 34 35 65 36  2 44 30 31 60 58 69 66 86
[34] 56 54  1 55 67 48 22 43 46 47 42  6 50 45  5 24
>
> low_cor_names=names(independant[, -highCorr])
> low_cor_data= independant[(names(independant) %in% low_cor_names)]
> dataforredun=low_cor_data
> redun_obj = redun (~. ,data = dataforredun ,nk =0)
> after_redun= dataforredun[,!(names(dataforredun) %in% redun_obj $Out)]
> form=as.formula(paste("post_release_defects>0~",paste(names(after_redun),collapse="+")))
```

```
> model=glm(formula=form, data=log10(data+1), family = binomial(link = "logit"))
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
>
> summary(model)

Call:
glm(formula = form, family = binomial(link = "logit"), data = log10(data +
    1))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2017  -0.1729  -0.1013  -0.0469   3.3298

Coefficients:
                                          Estimate Std. Error z value Pr(>|z|)
(Intercept)                             -5.679e+00  9.008e-01  -6.305 2.88e-10 ***
actions_abort_percentage                 1.062e+01  6.636e+00   1.600 0.109663
actions_log_percentage                  -1.419e+00  3.088e+00  -0.460 0.645868
this.actions_method_percentage           2.566e+00  1.770e+00   1.450 0.147187
actions_nestedtry_percentage             3.664e+00  2.064e+00   1.775 0.075825 .
actions_return_percentage               -1.559e+00  2.348e+00  -0.664 0.506833
actions_throwcurrent_percentage          7.766e-01  2.155e+00   0.360 0.718526
actions_thrownew_percentage              2.138e+01  1.768e+01   1.209 0.226623
actions_throwwrap_percentage             1.450e+00  1.759e+00   0.824 0.409663
actions_todo_percentage                  5.051e+00  1.148e+01   0.440 0.659848
subsumption_percentage                  -5.427e+00  1.927e+00  -2.816 0.004858 **
specific_percentage                     -2.491e+00  1.788e+00  -1.393 0.163559
overcatch_percentage                    -7.330e+00  2.577e+00  -2.845 0.004442 **
overcatch_abort_percentage              -2.887e+02  1.326e+04  -0.022 0.982626
catch_donothing_percentage              -1.177e+00  4.413e+00  -0.267 0.789674
catchgeneric_percentage                  3.318e+00  1.863e+00   1.781 0.074939 .
destructivewrapping_percentage           5.918e-01  2.382e+00   0.248 0.803769
ignoring_interrupted_exception_percentage 8.907e+00  5.686e+00   1.566 0.117252
log_returnnull_percentage               -3.436e+00  1.533e+01  -0.224 0.822697
log_throw_percentage                    -4.811e-02  4.641e+00  -0.010 0.991729
multiple_line_log_percentage             4.250e+00  1.301e+01   0.327 0.743883
throw_in_finally_percentage             -6.278e+00  2.675e+01  -0.235 0.814423
Number.of.Throws.Kitchen.Sink.AP         2.145e-01  5.477e-01   0.392 0.695416
Number.of.Try.in.Declaration            -1.493e+00  1.625e+00  -0.919 0.358290
Number.of.Try.in.Condition              -9.947e-01  7.865e-01  -1.265 0.205956
Number.of.Try.in.EH                      9.933e-01  5.941e-01   1.672 0.094559 .
Number.of.Try.in.Other                   1.417e+00  1.243e+00   1.140 0.254177
Code_Ownership_count                     1.343e+00  3.690e-01   3.639 0.000274 ***
lines_deleted                           -3.362e-01  2.146e-01  -1.567 0.117145
total_prerel_change                      1.161e+00  1.313e-01   8.837  < 2e-16 ***
pre_release_defects                      7.743e-01  1.237e+00   0.626 0.531326
AvgEssential                            -1.071e+00  1.596e+00  -0.671 0.502174
AvgLineBlank                             4.450e-01  7.334e-01   0.607 0.543989
AvgLineComment                           7.927e-02  5.623e-01   0.141 0.887903
CountDeclClass                          -3.648e-01  5.107e-01  -0.714 0.474985
CountDeclClassMethod                    -6.397e-01  2.958e-01  -2.163 0.030556 *
CountDeclClassVariable                   4.376e-01  2.913e-01   1.502 0.133042
CountDeclInstanceVariable                1.284e-01  3.038e-01   0.423 0.672526
CountDeclMethodDefault                  -3.525e-01  2.983e-01  -1.182 0.237341
CountDeclMethodPrivate                   1.355e-02  3.164e-01   0.043 0.965827
CountDeclMethodProtected                 7.381e-02  3.159e-01   0.234 0.815287
CountDeclMethodPublic                   -1.075e-01  3.286e-01  -0.327 0.743565
CountLineComment                         1.451e+00  6.648e-01   2.182 0.029090 *
RatioCommentToCode                      -8.010e+00  1.911e+00  -4.191 2.78e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1295.94  on 5207  degrees of freedom
```

```
Residual deviance:  838.11  on 5164  degrees of freedom
AIC: 926.11

Number of Fisher Scoring iterations: 15

>
>
>
> newform= post_release_defects>0~ subsumption_percentage + Code_Ownership_count + overcatch_percentage +
total_prerel_change + CountDeclClassMethod + CountLineComment + RatioCommentToCode
> newmodel=glm(formula=newform, data=log10(data+1), family = binomial(link = "logit"))
> summary(newmodel)

Call:
glm(formula = newform, family = binomial(link = "logit"), data = log10(data +
    1))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.7664  -0.1840  -0.1095  -0.0499   3.2326

Coefficients:
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)           -5.9100     0.4539 -13.021  < 2e-16 ***
subsumption_percentage  -1.8517     1.2331  -1.502   0.1332
Code_Ownership_count     1.3390     0.3335   4.014 5.96e-05 ***
overcatch_percentage    -5.5965     2.1782  -2.569   0.0102 *
total_prerel_change      1.0331     0.1063   9.717  < 2e-16 ***
CountDeclClassMethod    -0.6807     0.2515  -2.706   0.0068 **
CountLineComment         1.4348     0.3425   4.189 2.80e-05 ***
RatioCommentToCode      -8.1721     1.3728  -5.953 2.63e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1295.94  on 5207  degrees of freedom
Residual deviance:  878.28  on 5200  degrees of freedom
AIC: 894.28

Number of Fisher Scoring iterations: 9

>
>
>
> newform= post_release_defects>0~Code_Ownership_count + overcatch_percentage + total_prerel_change +
CountDeclClassMethod + CountLineComment + RatioCommentToCode
> newmodel=glm(formula=newform, data=log10(data+1), family = binomial(link = "logit"))
> summary(newmodel)

Call:
glm(formula = newform, family = binomial(link = "logit"), data = log10(data +
    1))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.7113  -0.1816  -0.1096  -0.0510   3.2155

Coefficients:
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)          -5.8749     0.4530 -12.970  < 2e-16 ***
Code_Ownership_count   1.3452     0.3325   4.046 5.21e-05 ***
overcatch_percentage  -5.7056     2.2060  -2.586  0.00970 **
total_prerel_change    1.0339     0.1064   9.715  < 2e-16 ***
CountDeclClassMethod  -0.6802     0.2516  -2.704  0.00685 **
CountLineComment       1.3375     0.3358   3.983 6.81e-05 ***
```

```
RatioCommentToCode    -7.8078     1.3418  -5.819 5.93e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1295.94  on 5207  degrees of freedom
Residual deviance:  880.69  on 5201  degrees of freedom
AIC: 894.69

Number of Fisher Scoring iterations: 9

>
>
>
> 1-880.69/1295.94
[1] 0.3204238
> anova(newmodel)
Analysis of Deviance Table

Model: binomial, link: logit

Response: post_release_defects > 0

Terms added sequentially (first to last)


                     Df Deviance Resid. Df Resid. Dev
NULL                                  5207    1295.94
Code_Ownership_count  1  210.110      5206    1085.83
overcatch_percentage  1    4.435      5205    1081.39
total_prerel_change   1  139.540      5204     941.85
CountDeclClassMethod  1    4.123      5203     937.73
CountLineComment      1    0.438      5202     937.29
RatioCommentToCode    1   56.601      5201     880.69
>
> testdata=data.frame(Code_Ownership_count =log10(mean(data$Code_Ownership_count)+1),
+ total_prerel_change=log10(mean(data$total_prerel_change)+1),
+ RatioCommentToCode =log10(mean(data$RatioCommentToCode)+1), overcatch_percentage
=log10(mean(data$overcatch_percentage)+1),
+ CountDeclClassMethod =log10(mean(data$CountDeclClassMethod)+1),
+ CountLineComment =log10(mean(data$CountLineComment)+1))
> predict(newmodel,testdata, type="response")
          1
0.006154201
>
> testdata=data.frame(Code_Ownership_count =log10(mean(data$Code_Ownership_count)*2+1),
+ total_prerel_change=log10(mean(data$total_prerel_change)+1),
+ RatioCommentToCode =log10(mean(data$RatioCommentToCode)+1), overcatch_percentage
=log10(mean(data$overcatch_percentage)+1),
+ CountDeclClassMethod =log10(mean(data$CountDeclClassMethod)+1),
+ CountLineComment =log10(mean(data$CountLineComment)+1))
> predict(newmodel,testdata, type="response")
          1
0.008612575
>
> testdata=data.frame(Code_Ownership_count =log10(mean(data$Code_Ownership_count)+1),
+ total_prerel_change=log10(mean(data$total_prerel_change)*2+1),
+ RatioCommentToCode =log10(mean(data$RatioCommentToCode)+1), overcatch_percentage
=log10(mean(data$overcatch_percentage)+1),
+ CountDeclClassMethod =log10(mean(data$CountDeclClassMethod)+1),
+ CountLineComment =log10(mean(data$CountLineComment)+1))
> predict(newmodel,testdata, type="response")
          1
0.008230919
>
```

```
> testdata=data.frame(Code_Ownership_count =log10(mean(data$Code_Ownership_count)+1),
+ total_prerel_change=log10(mean(data$total_prerel_change)+1),
+ RatioCommentToCode =log10(mean(data$RatioCommentToCode)*2+1), overcatch_percentage
=log10(mean(data$overcatch_percentage)+1),
+ CountDeclClassMethod =log10(mean(data$CountDeclClassMethod)+1),
+ CountLineComment =log10(mean(data$CountLineComment)+1))
> predict(newmodel,testdata, type="response")
          1
0.001217187
>
> testdata=data.frame(Code_Ownership_count =log10(mean(data$Code_Ownership_count)+1),
+ total_prerel_change=log10(mean(data$total_prerel_change)+1),
+ RatioCommentToCode =log10(mean(data$RatioCommentToCode)+1), overcatch_percentage
=log10(mean(data$overcatch_percentage)*2+1),
+ CountDeclClassMethod =log10(mean(data$CountDeclClassMethod)+1),
+ CountLineComment =log10(mean(data$CountLineComment)+1))
> predict(newmodel,testdata, type="response")
          1
0.005607301
>
> testdata=data.frame(Code_Ownership_count =log10(mean(data$Code_Ownership_count)+1),
+ total_prerel_change=log10(mean(data$total_prerel_change)+1),
+ RatioCommentToCode =log10(mean(data$RatioCommentToCode)+1), overcatch_percentage
=log10(mean(data$overcatch_percentage)+1),
+ CountDeclClassMethod =log10(mean(data$CountDeclClassMethod)*2+1),
+ CountLineComment =log10(mean(data$CountLineComment)+1))
>  predict(newmodel,testdata, type="response")
          1
0.005393117
> testdata=data.frame(Code_Ownership_count =log10(mean(data$Code_Ownership_count)+1),
+ total_prerel_change=log10(mean(data$total_prerel_change)+1),
+ RatioCommentToCode =log10(mean(data$RatioCommentToCode)+1), overcatch_percentage
=log10(mean(data$overcatch_percentage)+1),
+ CountDeclClassMethod =log10(mean(data$CountDeclClassMethod)+1),
+ CountLineComment =log10(mean(data$CountLineComment)*2+1))
> predict(newmodel,testdata, type="response")
          1
0.009130161
>
>
```