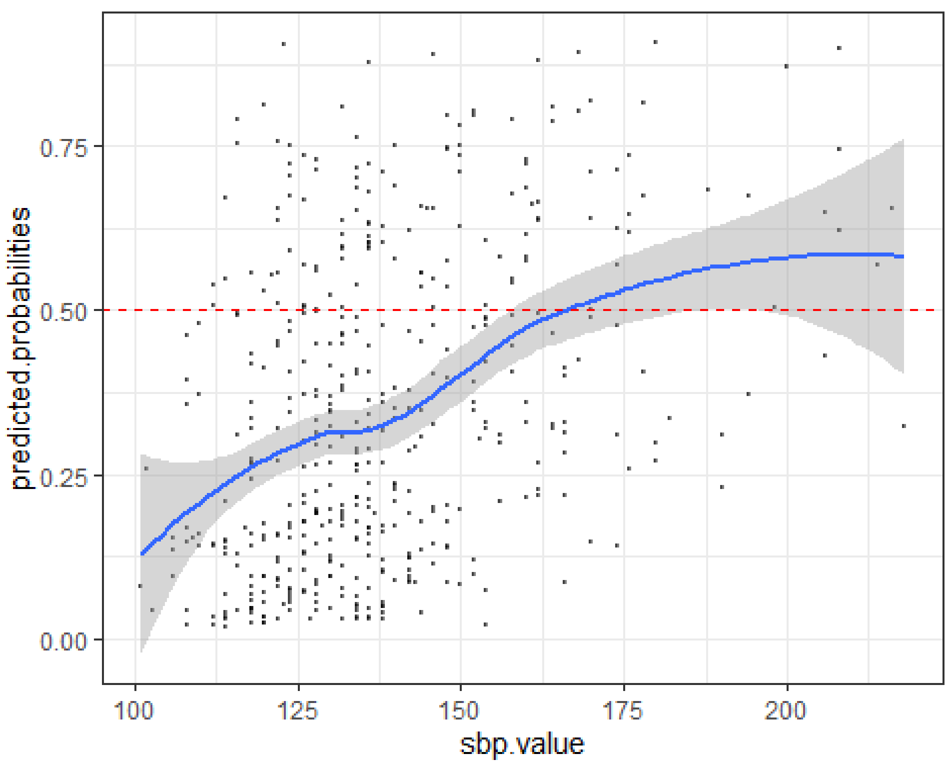
FILTER

2023-07-22

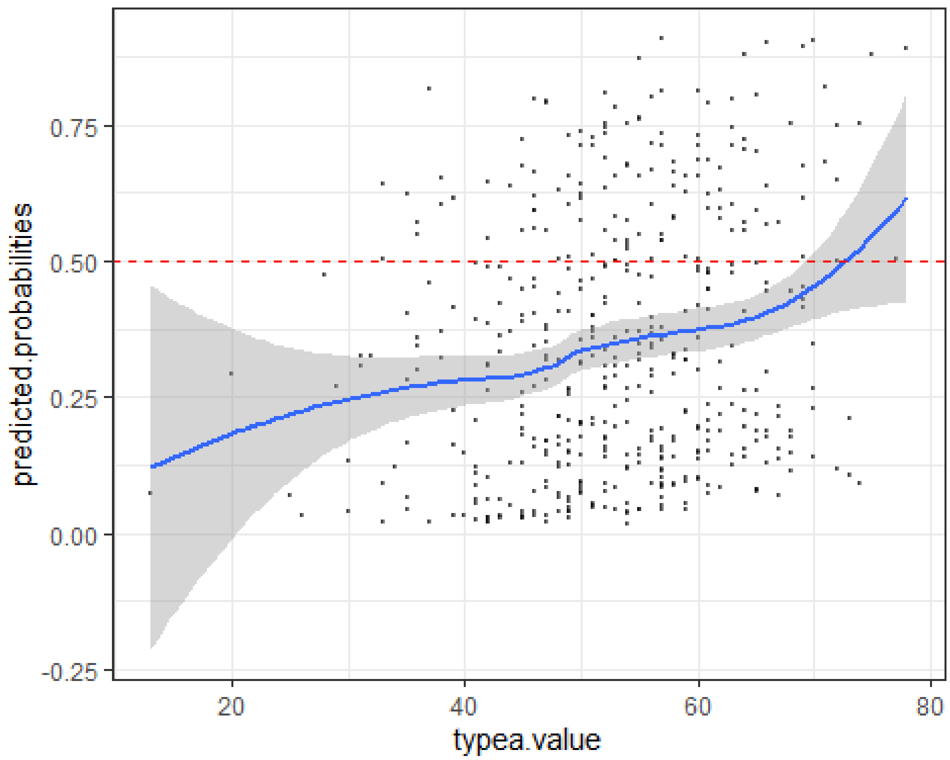
```r
Assignment <- read.csv("C:/Users/User/Downloads/chd.csv", header = TRUE)
Assignment$adiposity = NULL

library(dplyr)

## Warning: package 'dplyr' was built under R version 4.3.1

##
## 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(ggplot2)

## Warning: package 'ggplot2' was built under R version 4.3.1

library(tidyr)

## Warning: package 'tidyr' was built under R version 4.3.1

# Fit the logistic regression model
model <- glm(chd ~ ., family = binomial(link = 'logit'), data = Assignment)

# Predict probabilities
probabilities <- predict(model, type = "response")

# Predicted classes based on a threshold of 0.5
predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")

# Select only numeric predictors
mydata <- Assignment %>%
  dplyr::select_if(is.numeric)

# Add logit values to the dataframe
mydata$logit <- probabilities

# Gather the data for plotting
mydata <- mydata %>%
  gather(key = "predictors", value = "predictor.value", -logit)
```

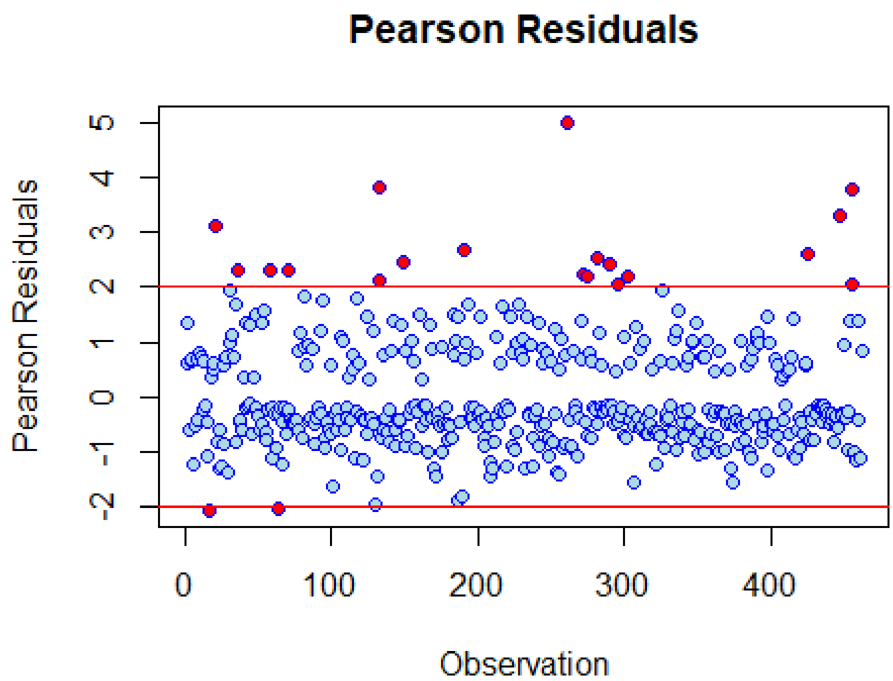
```r
#######sbp#####
# Filter only the "sbp" predictor for the plot
mydata_sbp <- mydata %>%
  filter(predictors == "sbp")

# Plot the graph for "sbp" predictor
ggplot(mydata_sbp, aes(predictor.value, logit)) +
  geom_point(size = 0.5, alpha = 0.5) +
labs(x="sbp.value",y="predicted.probabilities") +
  geom_smooth(method = "loess") +
  theme_bw() + geom_hline(yintercept = 0.5, linetype = "dashed", color =
"red")

## `geom_smooth()` using formula = 'y ~ x'
```



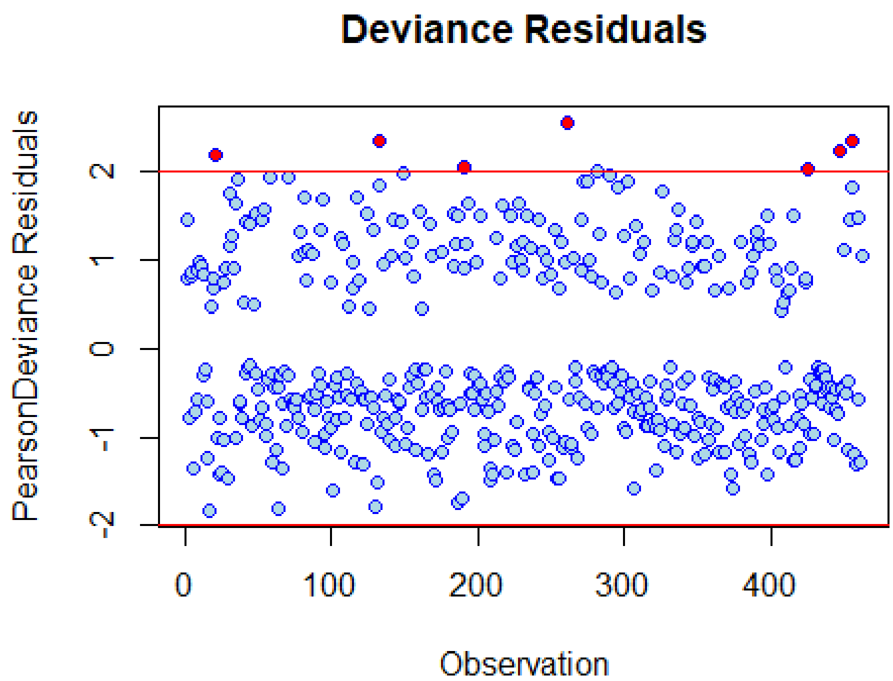

```r
mydata_typea <- mydata %>%
  filter(predictors == "typea")

# Plot the graph for "typea" predictor
ggplot(mydata_typea, aes(predictor.value, logit)) +
  geom_point(size = 0.5, alpha = 0.5) +
labs(x="typea.value",y="predicted.probabilities") +
  geom_smooth(method = "loess") +
```

```
  theme_bw() + geom_hline(yintercept = 0.5, linetype = "dashed", color =
"red")

## `geom_smooth()` using formula = 'y ~ x'
```

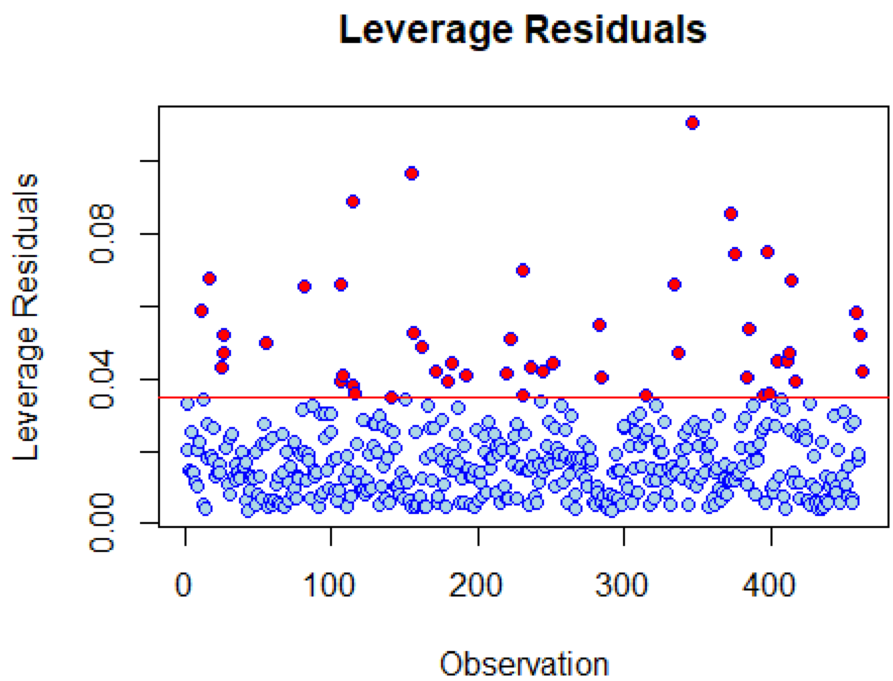




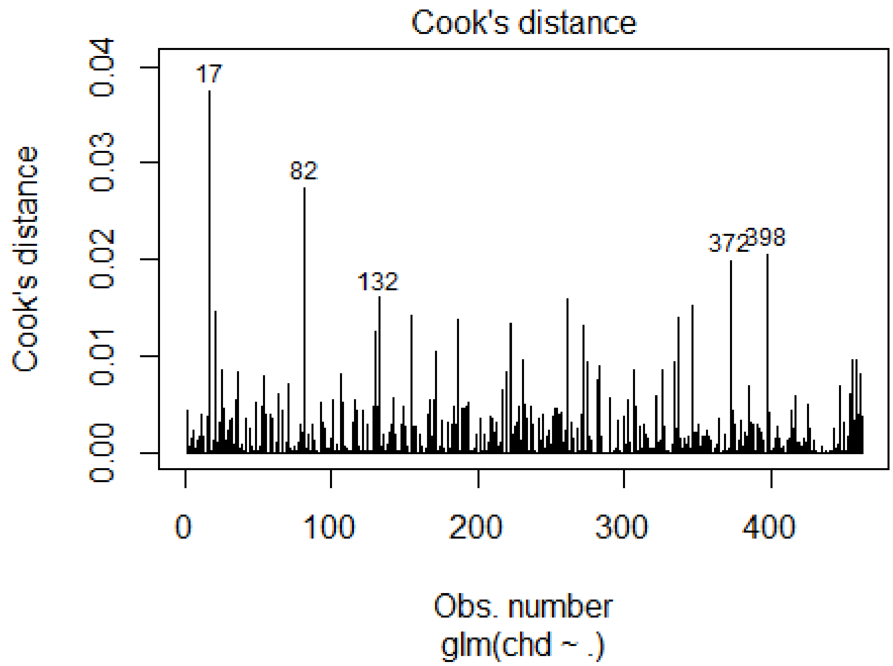

```
##  17  21  36  58  63  70 132 133 149 191 261 272 275 281 290 296 302 425
447 455
##  17  21  36  58  63  70 132 133 149 191 261 272 275 281 290 296 302 425
447 455
## 456
## 456
```



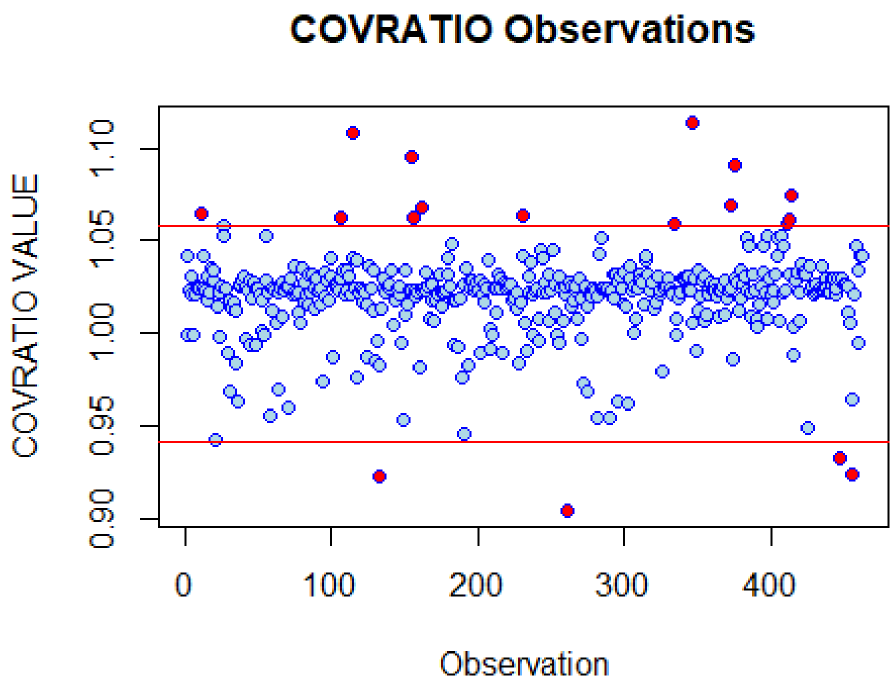

```
##   21 133 191 261 425 447 456
##   21 133 191 261 425 447 456

## [1] 0.1106625
```




```
##   11   17   25   26   27   56   82  106  107  108  114  115  116  141  155  156  162  171
180 182
##   11   17   25   26   27   56   82  106  107  108  114  115  116  141  155  156  162  171
180 182
## 192 220 222 230 231 236 244 251 283 285 315 334 337 346 372 375 383 385
395 398
## 192 220 222 230 231 236 244 251 283 285 315 334 337 346 372 375 383 385
395 398
## 399 405 411 413 414 417 458 461 462
## 399 405 411 413 414 417 458 461 462
```




```
## named integer(0)

## Warning: package 'olsrr' was built under R version 4.3.1

##
## Attaching package: 'olsrr'

## The following object is masked from 'package:datasets':
##
##     rivers
```

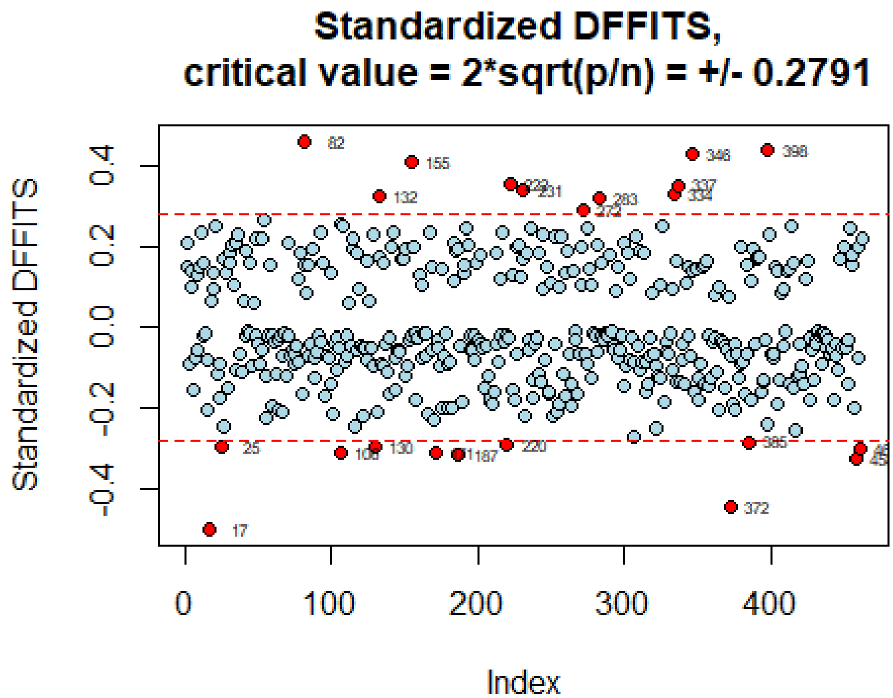



```
##   11 106 115 133 155 156 162 231 261 334 346 372 375 411 413 414 447 456
##   11 106 115 133 155 156 162 231 261 334 346 372 375 411 413 414 447 456

##            1           2           3           4           5           6
##  0.14631829  0.20668733 -0.09430808  0.09789284  0.13867155 -0.15876456
##            7           8           9          10          11          12
## -0.07459322  0.12700958 -0.05762864  0.14815796  0.23429299  0.15733995
##           13          14          15          16          17          18
## -0.02200407 -0.01532028 -0.20835283 -0.08045729 -0.50082821  0.06543484
##           19          20          21          22          23          24
##  0.09072837  0.13083809  0.24844711 -0.11869250 -0.17863937 -0.09372365
##           25          26          27          28          29          30
## -0.29615988  0.16799339 -0.24534598  0.13239633 -0.15132770  0.18176259
##           31          32          33          34          35          36
##  0.15606250  0.20179298  0.10380793  0.20928631 -0.10868563  0.22534347
##           37          38          39          40          41          42
## -0.06889027 -0.06562408 -0.11160699  0.06351756  0.18547816 -0.02266116
##           43          44          45          46          47          48
## -0.01280775  0.15666967 -0.01681539 -0.09773989  0.05613471  0.21586111
##           49          50          51          52          53          54
## -0.01959018 -0.04206899 -0.09003201 -0.05482582  0.21632957  0.26443734
##           55          56          57          58          59          60
## -0.12418954 -0.22815829 -0.02278092  0.15210530 -0.02056740 -0.19660573
##           61          62          63          64          65          66
## -0.03558132 -0.11635036 -0.20529923 -0.03268247 -0.02147978 -0.21286792
##           67          68          69          70          71          72
## -0.07388830 -0.01746062 -0.12399058  0.20597109 -0.02421277 -0.07109328
```

```
##          73          74          75          76          77          78 
## -0.05594267 -0.09412222 -0.06477749 -0.04852909 -0.08648021  0.11646651 
##          79          80          81          82          83          84 
##  0.18084066 -0.16884383  0.15422633  0.45797592  0.08330627  0.15501793 
##          85          86          87          88          89          90 
## -0.07379944 -0.04342902  0.19474179 -0.07724310 -0.12550014 -0.06006455 
##          91          92          93          94          95          96 
## -0.01991563  0.23524544 -0.03607890  0.15682773 -0.17043934 -0.17402371 
##          97          98          99         100         101         102 
## -0.07103957 -0.07756778  0.13125881 -0.14091791 -0.21420793 -0.04038978 
##         103         104         105         106         107         108 
## -0.10719813 -0.02588937 -0.05239886 -0.31376184  0.25113733  0.24693000 
##         109         110         111         112         113         114 
## -0.08720974 -0.01965693 -0.07108648  0.05691650 -0.04788762  0.19321145 
##         115         116         117         118         119         120 
##  0.21802885 -0.24768394  0.18467104 -0.05741308  0.09224339 -0.04685844 
##         121         122         123         124         125         126 
## -0.22146359 -0.05351269 -0.05175178  0.16341011 -0.08011781  0.06090677 
##         127         128         129         130         131         132 
## -0.05333361 -0.09761353  0.22647219 -0.29818731 -0.21261367  0.32296546 
##         133         134         135         136         137         138 
##  0.17121879 -0.09310966 -0.09804127  0.15667239 -0.04370002 -0.11320396 
##         139         140         141         142         143         144 
## -0.02567884 -0.16516785  0.19765361  0.23425204 -0.07841234 -0.15705589 
##         145         146         147         148         149         150 
## -0.05753827 -0.05515638 -0.06260947  0.16761805  0.16539534  0.19175210 
##         151         152         153         154         155         156 
## -0.12022390 -0.09489137 -0.03359151 -0.02013981  0.40780625  0.19614020 
##         157         158         159         160         161         162 
## -0.01603727 -0.18133937 -0.02899340  0.13015026 -0.07661265  0.10442643 
##         163         164         165         166         167         168 
## -0.01664955 -0.07409039 -0.01601235 -0.21688675  0.23189704  0.14635604 
##         169         170         171         172         173         174 
## -0.05598501 -0.23230517 -0.31378609 -0.03414631 -0.05201926 -0.09904209 
##         175         176         177         178         179         180 
## -0.20067142  0.14350265 -0.06995962 -0.01747946 -0.08952923 -0.20120886 
##         181         182         183         184         185         186 
## -0.08494139 -0.20244270  0.11287752  0.20838113  0.18663461  0.19489269 
##         187         188         189         190         191         192 
## -0.31827527 -0.04536607 -0.18767354  0.13191407  0.15211174  0.24224612 
##         193         194         195         196         197         198 
##  0.20443476 -0.05660445 -0.02025569 -0.05695236 -0.01976145 -0.08372722 
##         199         200         201         202         203         204 
##  0.15903616 -0.03360205 -0.04198138  0.17833156 -0.04941268 -0.14932703 
##         205         206         207         208         209         210 
## -0.15209924 -0.04612460 -0.12106573 -0.18577794 -0.17917280 -0.18998799 
##         211         212         213         214         215         216 
## -0.16142794  0.18514777 -0.04366483 -0.08867640 -0.02408900  0.11929268 
##         217         218         219         220         221         222 
##  0.23234204 -0.03650813 -0.02061635 -0.29095624 -0.02722128  0.35216546 
```

```
##        223        224        225        226        227        228 
## -0.15686047  0.12638075 -0.18079203 -0.06701028  0.19556406  0.20054155 
##        229        230        231        232        233        234 
##  0.12481160  0.16739221  0.33678411 -0.22252543  0.18112531 -0.03131657 
##        235        236        237        238        239        240 
## -0.05542395  0.24235549 -0.13040307 -0.17543725 -0.04249275 -0.02516715 
##        241        242        243        244        245        246 
##  0.18487057 -0.11455678 -0.13510023  0.22634904  0.09292084 -0.08175188 
##        247        248        249        250        251        252 
##  0.14802305 -0.16488418 -0.16060705  0.10672257 -0.21931603 -0.04333132 
##        253        254        255        256        257        258 
##  0.22186218 -0.21099830 -0.19211581  0.09590730  0.22151434 -0.11825577 
##        259        260        261        262        263        264 
##  0.13185567 -0.16926568  0.18991531 -0.06663164 -0.19522277 -0.14317608 
##        265        266        267        268        269        270 
##  0.13666280 -0.01378624 -0.03291085 -0.16639540 -0.06689263  0.10114297 
##        271        272        273        274        275        276 
##  0.19622472  0.28652357 -0.06764891 -0.08947190  0.24362259  0.14469661 
##        277        278        279        280        281        282 
##  0.10537604 -0.12699739 -0.01895283 -0.02158368  0.20283129 -0.06203322 
##        283        284        285        286        287        288 
##  0.31548043 -0.02229854  0.15627248 -0.03266782 -0.01727729 -0.01620961 
##        289        290        291        292        293        294 
## -0.02391930  0.17766334 -0.01214925 -0.04746638 -0.05744261  0.08681772 
##        295        296        297        298        299        300 
## -0.07501191  0.14885622 -0.05826131 -0.02202102  0.20676805 -0.14793846 
##        301        302        303        304        305        306 
## -0.09868714  0.18510266 -0.08159552  0.11664237 -0.02934577 -0.08757234 
##        307        308        309        310        311        312 
## -0.27363919  0.22078826 -0.04751729 -0.10676837  0.18602229 -0.08543235 
##        313        314        315        316        317        318 
##  0.18515694 -0.08588965 -0.16405609 -0.14261390 -0.07242913 -0.02673628 
##        319        320        321        322        323        324 
##  0.08225516 -0.06601269 -0.24919299 -0.09634494 -0.12264756  0.10423986 
##        325        326        327        328        329        330 
## -0.12715840  0.24826380 -0.18701512 -0.03804831 -0.06236462 -0.06559334 
##        331        332        333        334        335        336 
## -0.01459099 -0.10579204  0.09423402  0.32907259  0.16299580 -0.13779414 
##        337        338        339        340        341        342 
##  0.34970028 -0.13678113 -0.08058107 -0.04332717 -0.14367252  0.10910873 
##        343        344        345        346        347        348 
##  0.14780830 -0.02309397 -0.06943401  0.42888132  0.13907048 -0.16416825 
##        349        350        351        352        353        354 
##  0.14355168 -0.12809332 -0.18047406 -0.09160177  0.14956299  0.15159040 
##        355        356        357        358        359        360 
## -0.14284523  0.16157610 -0.02177354 -0.14879436 -0.12724156 -0.03641169 
##        361        362        363        364        365        366 
##  0.07684051 -0.02556336 -0.11185364 -0.20464630  0.10024191 -0.03121649 
##        367        368        369        370        371        372 
## -0.03761196 -0.04790435 -0.15029964 -0.05887418  0.07313327 -0.44589973
```

```
##         373         374         375         376         377         378 
## -0.06952687 -0.19333484 -0.20697185 -0.04112688 -0.06468023 -0.12693935 
##         379         380         381         382         383         384 
## -0.08966409  0.19973752 -0.08819937 -0.16877756  0.15482872 -0.06716779 
##         385         386         387         388         389         390 
## -0.28502247  0.11168871 -0.18362624  0.19267010 -0.04365912  0.16545479 
##         391         392         393         394         395         396 
##  0.17488316  0.17350457 -0.16319251 -0.13729453 -0.13202449 -0.03226171 
##         397         398         399         400         401         402 
## -0.23979401  0.43636208  0.22625542 -0.06675445 -0.07589716 -0.06749884 
##         403         404         405         406         407         408 
##  0.14646858  0.13996459 -0.19311637 -0.13158745  0.08185407  0.09332815 
##         409         410         411         412         413         414 
## -0.04730882 -0.01369808  0.13909391 -0.13991528  0.14482568  0.24913984 
##         415         416         417         418         419         420 
## -0.15015800  0.15622834 -0.25526394 -0.12023242 -0.11734281 -0.04405367 
##         421         422         423         424         425         426 
## -0.09645836 -0.14265774  0.11788108  0.11984371  0.16115051 -0.02559811 
##         427         428         429         430         431         432 
## -0.18033833 -0.05819372 -0.13342882 -0.03275040 -0.03946290 -0.01306577 
##         433         434         435         436         437         438 
## -0.01954249 -0.01683238 -0.09172605 -0.01585618 -0.05330722 -0.02438600 
##         439         440         441         442         443         444 
## -0.03347347 -0.06514990 -0.05896818 -0.04968165 -0.18054659 -0.07041561 
##         445         446         447         448         449         450 
## -0.04345060 -0.10440683  0.16552510 -0.05048732 -0.05359496  0.19800417 
##         451         452         453         454         455         456 
## -0.05377777 -0.03251807 -0.14308825  0.24305262  0.15260378  0.17765150 
##         457         458         459         460         461         462 
## -0.19948676 -0.32574312  0.19596308 -0.07824250 -0.30077718  0.21834598

##  17  25  82 106 130 132 155 171 187 220 222 231 272 283 334 337 346 372 
385 398 
##  17  25  82 106 130 132 155 171 187 220 222 231 272 283 334 337 346 372 
385 398 
## 458 461 
## 458 461
```




```
### NEW DATASET FILTERED

Assignment_filtered = Assignment[-
c(outlier_variables_DR,outlier_variables_PR,leverage_obs,COVRATIO_cuoffvalues
,influence_point_dffits,influence_point_dffits),]

### Calculate pseudo R-squared and p-value
### nagelkerke function also reports the McFadden, Cox and Snell, and
Nagelkerke pseudo R-squared value for the model

library(rcompanion)

## Warning: package 'rcompanion' was built under R version 4.3.1

varyingmodel <- glm(chd ~., family = binomial(link = 'logit'), data =
Assignment)
x <- varyingmodel
nagelkerke(x)

## $Models
##
## Model: "glm, chd ~ ., binomial(link = \"logit\"), Assignment"
## Null:  "glm, chd ~ 1, binomial(link = \"logit\"), Assignment"
##
## $Pseudo.R.squared.for.model.vs.null
##                              Pseudo.R.squared
## McFadden                             0.207283
```

```
## Cox and Snell (ML)                       0.234674
## Nagelkerke (Cragg and Uhler)             0.323775
##
## $Likelihood.ratio.test
##  Df.diff LogLik.diff  Chisq    p.value
##       -8     -61.782 123.56 6.0846e-23
##
## $Number.of.observations
##
## Model: 462
## Null:  462
##
## $Messages
## [1] "Note: For models fit with REML, these statistics are based on
refitting with ML"
##
## $Warnings
## [1] "None"
```

```
###### Efron's pseudo R-squared

efronRSquared(x)
```

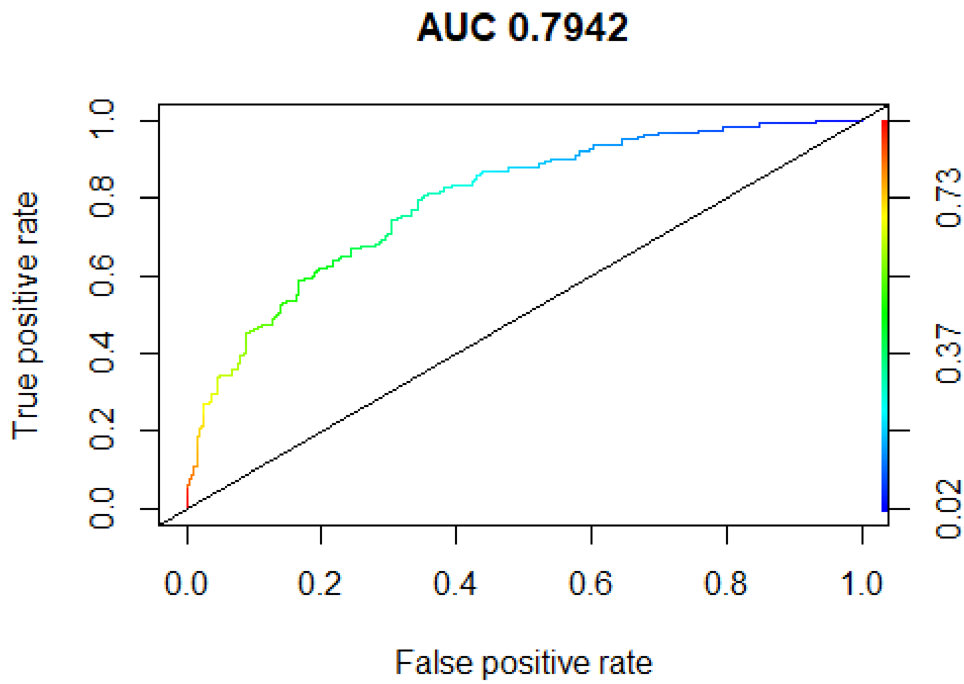
```
## EfronRSquared
##         0.245
```

```
varyingmodel <- glm(chd ~.-alcohol, family = binomial(link = 'logit'), data =
Assignment_filtered)
x <- varyingmodel
nagelkerke(x)
```

```
## $Models
##
## Model: "glm, chd ~ . - alcohol, binomial(link = \"logit\"),
Assignment_filtered"
## Null:  "glm, chd ~ 1, binomial(link = \"logit\"), Assignment_filtered"
##
## $Pseudo.R.squared.for.model.vs.null
##                              Pseudo.R.squared
## McFadden                             0.360943
## Cox and Snell (ML)                   0.351028
## Nagelkerke (Cragg and Uhler)         0.502786
##
## $Likelihood.ratio.test
##  Df.diff LogLik.diff  Chisq    p.value
##       -7     -84.527 169.05 3.9721e-33
##
## $Number.of.observations
##
## Model: 391
## Null:  391
```

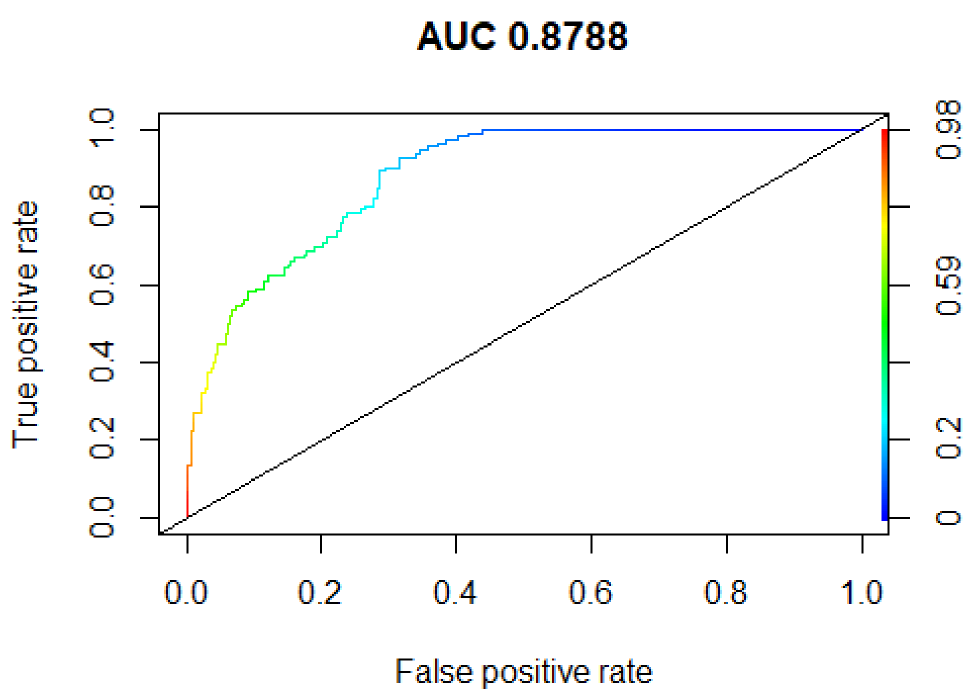
```
##
## $Messages
## [1] "Note: For models fit with REML, these statistics are based on
refitting with ML"
##
## $Warnings
## [1] "None"

###### Efron's pseudo R-squared

efronRSquared(x)

## EfronRSquared
##         0.372

#################################AUC METHOD
(***)#################################


### AUC ROC for Initial
library(prediction)

## Warning: package 'prediction' was built under R version 4.3.1

library(caret)

## Warning: package 'caret' was built under R version 4.3.1

## Loading required package: lattice

library(ROCR)

## Warning: package 'ROCR' was built under R version 4.3.1

##
## Attaching package: 'ROCR'

## The following object is masked from 'package:prediction':
##
##     prediction

model1 <- glm(chd ~. , family = binomial(link = 'logit'), data = Assignment)
pred <- predict(model1, Assignment, type = "response")
pred.rocr <- prediction(pred, Assignment$chd)



perf.rocr <- performance(pred.rocr, measure = "auc", x.measure = "cutoff")

perf.rocr@y.values[[1]] <- round(perf.rocr@y.values[[1]], digits = 4)

perf.tpr.fpr.rocr <- performance(pred.rocr,"tpr","fpr")
```

```r
plot(perf.tpr.fpr.rocr, colorize = T,main =
paste("AUC",(perf.rocr@y.values)))
abline(a=0,b=1)
```




```r
### AUC ROC for Adjusted

model2 <- glm(chd ~. , family = binomial(link = 'logit'), data =
Assignment_filtered)
pred <- predict(model2, Assignment_filtered, type = "response")
pred.rocr <- prediction(pred, Assignment_filtered$chd)
summary(model2)

##
## Call:
## glm(formula = chd ~ ., family = binomial(link = "logit"), data =
Assignment_filtered)
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.001e+01  1.966e+00  -5.091 3.56e-07 ***
## sbp          1.134e-02  8.744e-03   1.296 0.194838
## tobacco      1.406e-01  4.114e-02   3.417 0.000633 ***
## ldl          3.809e-01  9.141e-02   4.167 3.09e-05 ***
## famhist      1.396e+00  2.969e-01   4.703 2.57e-06 ***
## typea        6.208e-02  1.807e-02   3.436 0.000591 ***
```

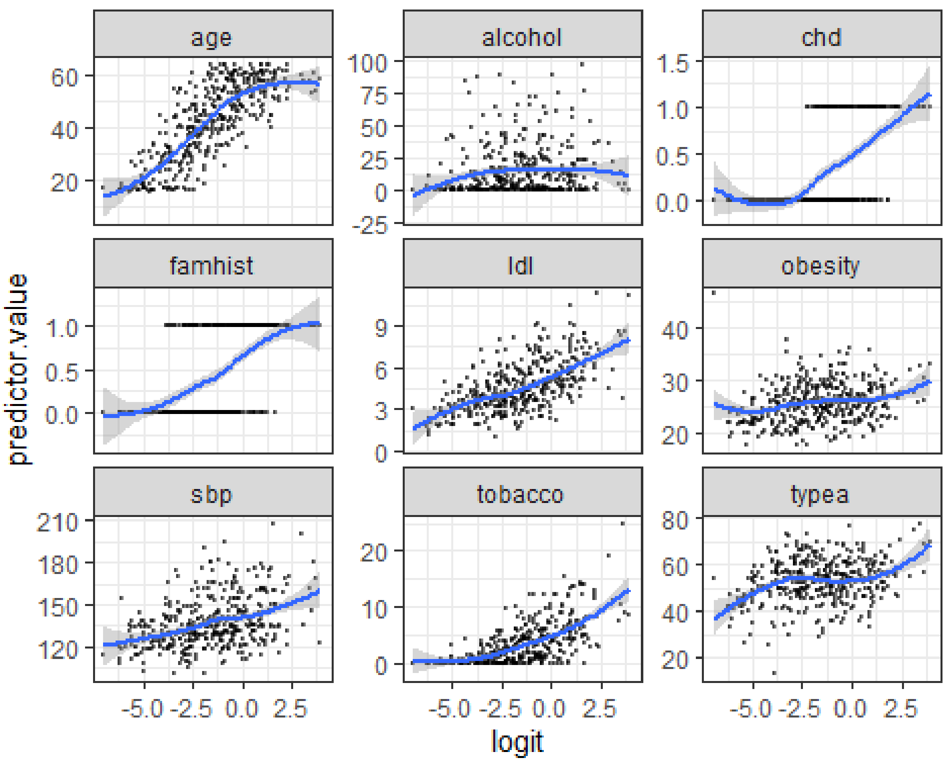
```
## obesity     -8.427e-02  4.377e-02  -1.925 0.054213 .
## alcohol     -6.428e-04  7.824e-03  -0.082 0.934528
## age          7.051e-02  1.471e-02   4.794 1.63e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 468.37  on 390  degrees of freedom
## Residual deviance: 299.31  on 382  degrees of freedom
## AIC: 317.31
##
## Number of Fisher Scoring iterations: 6

perf.rocr <- performance(pred.rocr, measure = "auc", x.measure = "cutoff")

perf.rocr@y.values[[1]] <- round(perf.rocr@y.values[[1]], digits = 4)

perf.tpr.fpr.rocr <- performance(pred.rocr,"tpr","fpr")


plot(perf.tpr.fpr.rocr, colorize = T,main =
paste("AUC",(perf.rocr@y.values)))
abline(a=0,b=1)
```




```
############TESTING WHETHER BINOMIAL LOGISTICS IS
APPROPRIATE########################
```

```r
############USING LOGISTICS REGRESSION DIAGNOSTICS################

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.3.1

## Warning: package 'readr' was built under R version 4.3.1

## Warning: package 'purrr' was built under R version 4.3.1

## Warning: package 'stringr' was built under R version 4.3.1

## Warning: package 'forcats' was built under R version 4.3.1

## Warning: package 'lubridate' was built under R version 4.3.1

## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✔ forcats   1.0.0     ✔ readr     2.1.4
## ✔ lubridate 1.9.2     ✔ stringr   1.5.0
## ✔ purrr     1.0.1     ✔ tibble    3.2.1
## ── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ✖ purrr::lift()   masks caret::lift()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors

library(broom)

## Warning: package 'broom' was built under R version 4.3.1

new_model <- glm(chd ~., family = binomial(link = 'logit'), data =
Assignment_filtered)
probabilities <- predict(new_model, type = "response")
predicted.classes <- ifelse(probabilities > 0.5,"pos","neg")


#Select only numeric predictors
mydata <- Assignment_filtered %>%
  dplyr::select_if(is.numeric)
predictors <- colnames(mydata)

#Bind the logit and tidying the data for plot
mydata <- mydata %>%
  mutate(logit = log(probabilities/(1-probabilities))) %>%
  gather(key = "predictors", value = "predictor.value", -logit)

#Graph visualization
ggplot(mydata, aes(logit, predictor.value))+
```

```
  geom_point(size = 0.5, alpha = 0.5) +
  geom_smooth(method = "loess") +
  theme_bw() +
  facet_wrap(~predictors, scales = "free_y")

## `geom_smooth()` using formula = 'y ~ x'
```




```
summary(new_model)

##
## Call:
## glm(formula = chd ~ ., family = binomial(link = "logit"), data =
Assignment_filtered)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.001e+01  1.966e+00  -5.091 3.56e-07 ***
## sbp          1.134e-02  8.744e-03   1.296 0.194838
## tobacco      1.406e-01  4.114e-02   3.417 0.000633 ***
## ldl          3.809e-01  9.141e-02   4.167 3.09e-05 ***
## famhist      1.396e+00  2.969e-01   4.703 2.57e-06 ***
## typea        6.208e-02  1.807e-02   3.436 0.000591 ***
## obesity     -8.427e-02  4.377e-02  -1.925 0.054213 .
## alcohol     -6.428e-04  7.824e-03  -0.082 0.934528
## age          7.051e-02  1.471e-02   4.794 1.63e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 468.37  on 390  degrees of freedom
## Residual deviance: 299.31  on 382  degrees of freedom
## AIC: 317.31
##
## Number of Fisher Scoring iterations: 6
```

```
stepwise_newmodel <- step(new_model, direction = "both", trace = 0, k = 2)

# Print the final selected model
summary(stepwise_newmodel)
```

```
##
## Call:
## glm(formula = chd ~ tobacco + ldl + famhist + typea + obesity +
##     age, family = binomial(link = "logit"), data = Assignment_filtered)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.66248    1.64093  -5.279 1.30e-07 ***
## tobacco      0.14075    0.03959   3.556 0.000377 ***
## ldl          0.38155    0.09013   4.233 2.30e-05 ***
## famhist      1.37933    0.29506   4.675 2.94e-06 ***
## typea        0.05897    0.01780   3.313 0.000924 ***
## obesity     -0.07732    0.04301  -1.798 0.072210 .
## age          0.07489    0.01425   5.255 1.48e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 468.37  on 390  degrees of freedom
## Residual deviance: 301.04  on 384  degrees of freedom
## AIC: 315.04
##
## Number of Fisher Scoring iterations: 6
```

```
car::vif(new_model)
```

```
##      sbp  tobacco      ldl  famhist    typea  obesity  alcohol      age
## 1.177445 1.116131 1.186976 1.029565 1.174389 1.209209 1.138752 1.280999
```

```
##################################
###### Calculate pseudo R-squared and p-value (CAN ANALYZE REMOVAL OF
RESIDUALS ****)
###### alternative test for p-value for a fitted model -> nagelkerke function
will report the p-value for a model using the LRT

###### nagelkerke function also reports the McFadden, Cox and Snell, and
```

```r
Nagelkerke pseudo R-squared value for the model

library(rcompanion)

varyingmodel <- glm(chd ~., family = binomial(link = 'logit'), data =
Assignment)
x <- varyingmodel
nagelkerke(x)

## $Models
##
## Model: "glm, chd ~ ., binomial(link = \"logit\"), Assignment"
## Null:  "glm, chd ~ 1, binomial(link = \"logit\"), Assignment"
##
## $Pseudo.R.squared.for.model.vs.null
##                              Pseudo.R.squared
## McFadden                             0.207283
## Cox and Snell (ML)                   0.234674
## Nagelkerke (Cragg and Uhler)         0.323775
##
## $Likelihood.ratio.test
##  Df.diff LogLik.diff  Chisq    p.value
##       -8     -61.782 123.56 6.0846e-23
##
## $Number.of.observations
##
## Model: 462
## Null:  462
##
## $Messages
## [1] "Note: For models fit with REML, these statistics are based on
refitting with ML"
##
## $Warnings
## [1] "None"

###### Efron's pseudo R-squared

efronRSquared(x)

## EfronRSquared
##         0.245

varyingmodel <- glm(chd ~.-alcohol, family = binomial(link = 'logit'), data =
Assignment_filtered)
x <- varyingmodel
nagelkerke(x)

## $Models
##
## Model: "glm, chd ~ . - alcohol, binomial(link = \"logit\"),
```

```
Assignment_filtered"
## Null:  "glm, chd ~ 1, binomial(link = \"logit\"), Assignment_filtered"
##
## $Pseudo.R.squared.for.model.vs.null
##                              Pseudo.R.squared
## McFadden                             0.360943
## Cox and Snell (ML)                   0.351028
## Nagelkerke (Cragg and Uhler)         0.502786
##
## $Likelihood.ratio.test
##  Df.diff LogLik.diff  Chisq    p.value
##       -7     -84.527 169.05 3.9721e-33
##
## $Number.of.observations
##
## Model: 391
## Null:  391
##
## $Messages
## [1] "Note: For models fit with REML, these statistics are based on
refitting with ML"
##
## $Warnings
## [1] "None"
```

```
###### Efron's pseudo R-squared

efronRSquared(x)
```

```
## EfronRSquared
##         0.372
```