36-669 HW2

Q1: Odds Ratio (OR) and Logistic Regression

```
# loading in the data
source("https://raw.githubusercontent.com/xuranw/469_public/master/hw2/hw2_functions.R")
famuss <- read.csv("https://raw.githubusercontent.com/xuranw/469_public/master/hw2/synthetic_famuss.csv
```

1.A

```
# AA - is the recessive variant
A <- ifelse(famuss$actn3_rs540874 == 2, 1, 0)
# reorders the table
tab <- table(A, famuss$heart_disease)[c(2,1), c(2,1)]
tab
```

```
##
## A     1   0
##   1  53  45
##   0 212 221
```

```
# Odds Ratio
(tab[1,1]/tab[1,2])/(tab[2,1]/tab[2,2])
```

```
## [1] 1.227778
```

Since the OR is greater than 1, people with genotype AA for SNP actn3_rs540874 have a higher odds of getting heart disease as opposed to people with the genotypes Aa, aa.

1.B

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-2
```

```r
library(stats)
glm_res <- stats::glm(heart_disease ~ . - norm_BMI, data = famuss, family = stats::binomial)

idx <- which(colnames(famuss) == "heart_disease")

coef_vec <- output_coefficients(glm_res, famuss, idx)
summary(glm_res)
```

```
##
## Call:
## stats::glm(formula = heart_disease ~ . - norm_BMI, family = stats::binomial,
##     data = famuss)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0942  -0.9882  -0.0016   0.9815   2.1487
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)             0.080174   0.627423   0.128  0.89832
## actn3_rs540874         -0.405830   0.552981  -0.734  0.46301
## actn3_rs1815739        -0.318006   0.595161  -0.534  0.59312
## actn3_1671064           1.202065   0.519670   2.313  0.02072 *
## adrb2_rs1042718        -0.137026   0.280682  -0.488  0.62542
## akt1_t22932c            0.133900   0.301615   0.444  0.65708
## akt1_g15129a            0.123359   0.244466   0.505  0.61384
## akt1_c10744t_c12886t    0.219607   0.617872   0.355  0.72227
## akt1_t10726c_t12868c   -0.287049   0.465894  -0.616  0.53781
## akt1_t10598a_t12740a   -0.099752   0.247804  -0.403  0.68728
## akt1_c9756a_c11898t     0.761327   1.158506   0.657  0.51108
## akt1_t8407g             0.467689   0.516984   0.905  0.36565
## akt1_a7699g            -0.014588   0.693551  -0.021  0.98322
## akt1_c6024t_c8166t      0.258051   1.219192   0.212  0.83237
## akt1_g2347t_g205t      -0.633728   0.440453  -1.439  0.15020
## akt1_g2375a_g233a       0.050232   0.292482   0.172  0.86364
## akt1_g4362c            -0.069427   0.428365  -0.162  0.87125
## akt1_g22187a           -0.058455   0.191238  -0.306  0.75986
## akt2_rs892118          -0.182085   0.194714  -0.935  0.34972
## akt2_2304186           -0.354732   0.135293  -2.622  0.00874 **
## akt2_969531             0.045336   0.168185   0.270  0.78750
## ankrd6_q122e           -0.223759   0.569976  -0.393  0.69463
## ankrd6_m485l          -13.145433 535.412243  -0.025  0.98041
## ankrd6_p636l            0.494254   0.419217   1.179  0.23840
## ankrd6_t233m            0.623379   0.274178   2.274  0.02299 *
## ankrd6_i128l            0.148630   0.384967   0.386  0.69943
## ankrd6_g197805a        -0.172361   0.207998  -0.829  0.40729
## ankrd6_a545t           -0.259629   0.532239  -0.488  0.62569
## ankrd6_k710x            0.414225   0.574491   0.721  0.47089
## bcl6_4686467           -0.134612   0.139034  -0.968  0.33295
## bcl6_3774298            0.121982   0.189174   0.645  0.51905
## bcl6_17797517          -0.035822   0.165627  -0.216  0.82877
## bcl6_1056932           -0.031219   0.177273  -0.176  0.86021
## bmp2_rs15705            0.230553   0.176903   1.303  0.19248
```

```
## c8orf68_rs6983944      0.077586   0.147975   0.524  0.60006
## carp_a8470g           -0.008492   0.134181  -0.063  0.94954
## carp_c105t             0.207022   0.159223   1.300  0.19353
## cast_rs754615          0.008474   0.152375   0.056  0.95565
## cast_rs7724759        -0.268130   0.171560  -1.563  0.11808
## cntf_g6a              -0.018692   0.197161  -0.095  0.92447
## ctsf_572846           -0.334607   0.393614  -0.850  0.39528
## ddit_rs1053227         0.028640   0.128573   0.223  0.82373
## esr1_rs1801132        -0.266698   0.167870  -1.589  0.11212
## esr1_rs1042717        -0.009527   0.276441  -0.034  0.97251
## esr1_rs2228480        -0.089727   0.181953  -0.493  0.62192
## esr1_rs2077647         0.117578   0.135941   0.865  0.38708
## fbox32_rs6690663      -0.208262   0.135444  -1.538  0.12414
## fbox32_rs3739287      -0.121456   0.245197  -0.495  0.62036
## fbox32_rs4871385      -0.469161   0.191202  -2.454  0.01414 *
## fstl1_rs9631455        0.232571   0.167176   1.391  0.16417
## gnb3_rs5443           -0.080703   0.145910  -0.553  0.58019
## igf1_pro              -0.433477   0.193386  -2.242  0.02499 *
## igf2_rs680             0.195803   0.160291   1.222  0.22188
## il15ra_3136618         0.944935   0.385747   2.450  0.01430 *
## il15ra_2228059        -0.580709   0.360917  -1.609  0.10762
## il15_1057972           0.132948   0.137908   0.964  0.33503
## il15_1589241           0.177142   0.174536   1.015  0.31014
## il15_rs2296135        -0.196793   0.193173  -1.019  0.30833
## insig2_rs7566605       0.011691   0.158454   0.074  0.94118
## irs1_g972r            -0.382828   0.317201  -1.207  0.22747
## kchj11_rs5219         -0.147218   0.153221  -0.961  0.33664
## mylk_c37885a          -0.125487   0.200064  -0.627  0.53051
## mylk_g91689t          -0.052061   0.229440  -0.227  0.82050
## myod1_rs2249104        0.238547   0.248976   0.958  0.33801
## nos3_rs1799983         0.109479   0.159090   0.688  0.49135
## p2ry2_rs1783596       -0.318362   0.135874  -2.343  0.01913 *
## pik3_rs3173908        -0.250033   0.175535  -1.424  0.15433
## ppara_1800206         -0.161186   0.295128  -0.546  0.58496
## ppar_gp12a             0.018641   0.230249   0.081  0.93548
## pparg_c1a_rs8192678   -0.237504   0.158674  -1.497  0.13444
## rs302964               0.027129   0.124807   0.217  0.82792
## tcfl72_12255372        0.439133   0.625991   0.702  0.48299
## tcfl72_7903146         0.244577   0.547976   0.446  0.65536
## tcfl72_rs12255372     -0.060135   0.606099  -0.099  0.92097
## tcfl72_rs7903146      -0.167847   0.518402  -0.324  0.74611
## tpd52l1_3778458        0.007758   0.177916   0.044  0.96522
## tpd52l1_514096         0.095078   0.135235   0.703  0.48202
## tpd52l1_4896782       -0.498774   0.416176  -1.198  0.23073
## tdp52l1_9321028       -0.207792   0.247569  -0.839  0.40128
## vdr_rs731236           0.121341   0.147104   0.825  0.40945
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 736.12  on 530  degrees of freedom
## Residual deviance: 630.21  on 451  degrees of freedom
## AIC: 790.21
```

```
##
## Number of Fisher Scoring iterations: 12
```

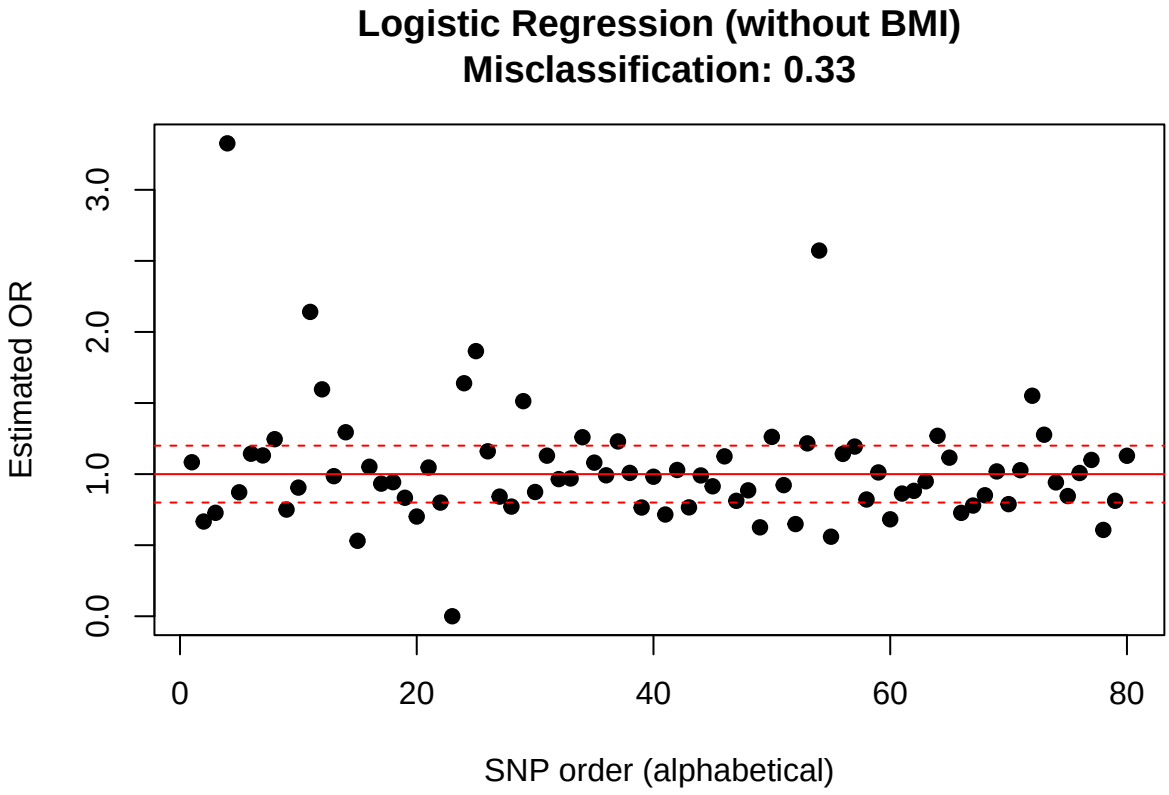
```
pred_vec <- output_predictions(glm_res, famuss, idx)

# misclassification rate
obs_response <- famuss$heart_disease
tab_mis <- table(pred_vec, obs_response)
tab_mis
```

```
##         obs_response
## pred_vec   0   1
##        0 181  90
##        1  85 175
```

```
# calculating the rate
1 - sum(diag(tab_mis))/sum(tab_mis)
```

```
## [1] 0.3295669
```

```
# graphic
coef_vec_graph <- coef_vec[-(which(coef_vec == coef_vec["(Intercept)"]))]
plot(exp(coef_vec),
     main = "Logistic Regression (without BMI)\nMisclassification: 0.33",
     xlab = "SNP order (alphabetical)", ylab = "Estimated OR",
     pch = 19)
abline(h = 1.2, col = "red", lty = 2)
abline(h = 1.0, col = "red")
abline(h = 0.8, col = "red", lty = 2)
```



1.C

```
glm_res2 <- stats::glm(heart_disease ~ ., data = famuss, family = stats::binomial)

coef_vec2 <- output_coefficients(glm_res2, famuss, idx)
summary(glm_res2)
```

```
##
## Call:
## stats::glm(formula = heart_disease ~ ., family = stats::binomial,
##     data = famuss)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.5084  -0.3105  -0.0010   0.2122   3.3187
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          2.837726   1.225273   2.316  0.02056 *
## norm_BMI             5.840162   0.659683   8.853  < 2e-16 ***
## actn3_rs540874       0.394324   0.891840   0.442  0.65838
## actn3_rs1815739     -1.246128   1.150891  -1.083  0.27892
## actn3_1671064        1.457544   1.006229   1.449  0.14747
```

```
## adrb2_rs1042718        0.264510   0.481025   0.550  0.58240
## akt1_t22932c           0.912991   0.489653   1.865  0.06224 .
## akt1_g15129a          -0.227476   0.406984  -0.559  0.57621
## akt1_c10744t_c12886t  -1.111908   1.099840  -1.011  0.31203
## akt1_t10726c_t12868c   0.404263   0.641861   0.630  0.52881
## akt1_t10598a_t12740a   0.096534   0.403304   0.239  0.81083
## akt1_c9756a_c11898t    0.479234   2.196350   0.218  0.82728
## akt1_t8407g            0.260669   1.084929   0.240  0.81013
## akt1_a7699g           -0.770936   1.247732  -0.618  0.53666
## akt1_c6024t_c8166t     1.100386   2.212909   0.497  0.61901
## akt1_g2347t_g205t     -1.068377   0.934462  -1.143  0.25291
## akt1_g2375a_g233a     -0.286731   0.523957  -0.547  0.58421
## akt1_g4362c            0.452861   0.736364   0.615  0.53856
## akt1_g22187a          -0.211445   0.347008  -0.609  0.54230
## akt2_rs892118          0.168963   0.361303   0.468  0.64004
## akt2_2304186          -0.651481   0.251055  -2.595  0.00946 **
## akt2_969531            0.073327   0.274349   0.267  0.78926
## ankrd6_q122e          -0.036508   0.946881  -0.039  0.96924
## ankrd6_m485l         -14.094755 882.745232  -0.016  0.98726
## ankrd6_p636l           0.381030   0.665126   0.573  0.56673
## ankrd6_t233m           0.762366   0.455049   1.675  0.09387 .
## ankrd6_i128l          -0.228999   0.631579  -0.363  0.71692
## ankrd6_g197805a       -0.540550   0.402335  -1.344  0.17910
## ankrd6_a545t           0.146238   1.007494   0.145  0.88459
## ankrd6_k710x           1.610624   0.934870   1.723  0.08492 .
## bcl6_4686467          -0.132375   0.255951  -0.517  0.60503
## bcl6_3774298          -0.111166   0.356745  -0.312  0.75534
## bcl6_17797517          0.152083   0.272653   0.558  0.57699
## bcl6_1056932           0.303638   0.331200   0.917  0.35926
## bmp2_rs15705           0.574826   0.340323   1.689  0.09121 .
## c8orf68_rs6983944     -0.078637   0.265578  -0.296  0.76716
## carp_a8470g           -0.117978   0.225672  -0.523  0.60112
## carp_c105t             0.150689   0.274777   0.548  0.58341
## cast_rs754615         -0.052383   0.269635  -0.194  0.84596
## cast_rs7724759        -0.290291   0.318875  -0.910  0.36263
## cntf_g6a               0.199727   0.347662   0.574  0.56564
## ctsf_572846           -0.621909   0.672978  -0.924  0.35543
## ddit_rs1053227        -0.019108   0.232531  -0.082  0.93451
## esr1_rs1801132        -0.736112   0.322343  -2.284  0.02239 *
## esr1_rs1042717        -0.745837   0.488775  -1.526  0.12703
## esr1_rs2228480        -0.302792   0.318640  -0.950  0.34198
## esr1_rs2077647         0.003913   0.244851   0.016  0.98725
## fbox32_rs6690663      -0.254277   0.241506  -1.053  0.29240
## fbox32_rs3739287      -0.737386   0.448126  -1.645  0.09987 .
## fbox32_rs4871385      -0.458534   0.333324  -1.376  0.16893
## fstl1_rs9631455       -0.057026   0.294483  -0.194  0.84645
## gnb3_rs5443           -0.210385   0.253714  -0.829  0.40698
## igf1_pro              -0.666942   0.344910  -1.934  0.05315 .
## igf2_rs680             0.331963   0.300634   1.104  0.26950
## il15ra_3136618         1.665848   0.677904   2.457  0.01400 *
## il15ra_2228059        -1.114397   0.627465  -1.776  0.07573 .
## il15_1057972           0.443599   0.257249   1.724  0.08464 .
## il15_1589241          -0.324451   0.296908  -1.093  0.27450
## il15_rs2296135        -0.086989   0.363447  -0.239  0.81084
```

```
## insig2_rs7566605      -0.039150   0.274498  -0.143  0.88659
## irs1_g972r            -0.973866   0.524369  -1.857  0.06328 .
## kchj11_rs5219         -0.058870   0.266021  -0.221  0.82486
## mylk_c37885a          -0.664834   0.340238  -1.954  0.05070 .
## mylk_g91689t           0.485969   0.428353   1.135  0.25658
## myod1_rs2249104        0.812043   0.446030   1.821  0.06867 .
## nos3_rs1799983        -0.136888   0.273443  -0.501  0.61665
## p2ry2_rs1783596       -0.011166   0.234492  -0.048  0.96202
## pik3_rs3173908        -0.896891   0.329683  -2.720  0.00652 **
## ppara_1800206         -1.247528   0.502231  -2.484  0.01299 *
## ppar_gp12a             0.098450   0.389356   0.253  0.80038
## pparg_c1a_rs8192678   -0.403895   0.285990  -1.412  0.15787
## rs302964               0.139901   0.220783   0.634  0.52630
## tcfl72_12255372        1.594845   1.575970   1.012  0.31155
## tcfl72_7903146        -1.313863   1.331251  -0.987  0.32367
## tcfl72_rs12255372     -1.362008   1.521243  -0.895  0.37061
## tcfl72_rs7903146       1.293427   1.252781   1.032  0.30186
## tpd52l1_3778458       -0.161549   0.326162  -0.495  0.62039
## tpd52l1_514096         0.271204   0.247406   1.096  0.27300
## tpd52l1_4896782       -0.542203   0.759922  -0.713  0.47554
## tdp52l1_9321028       -0.008627   0.414047  -0.021  0.98338
## vdr_rs731236           0.078389   0.254015   0.309  0.75763
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 736.12  on 530  degrees of freedom
## Residual deviance: 270.74  on 450  degrees of freedom
## AIC: 432.74
##
## Number of Fisher Scoring iterations: 13
```

```
pred_vec2 <- output_predictions(glm_res2, famuss, idx)

# misclassification rate part 2
obs_response <- famuss$heart_disease
tab_mis2 <- table(pred_vec2, obs_response)
tab_mis2
```

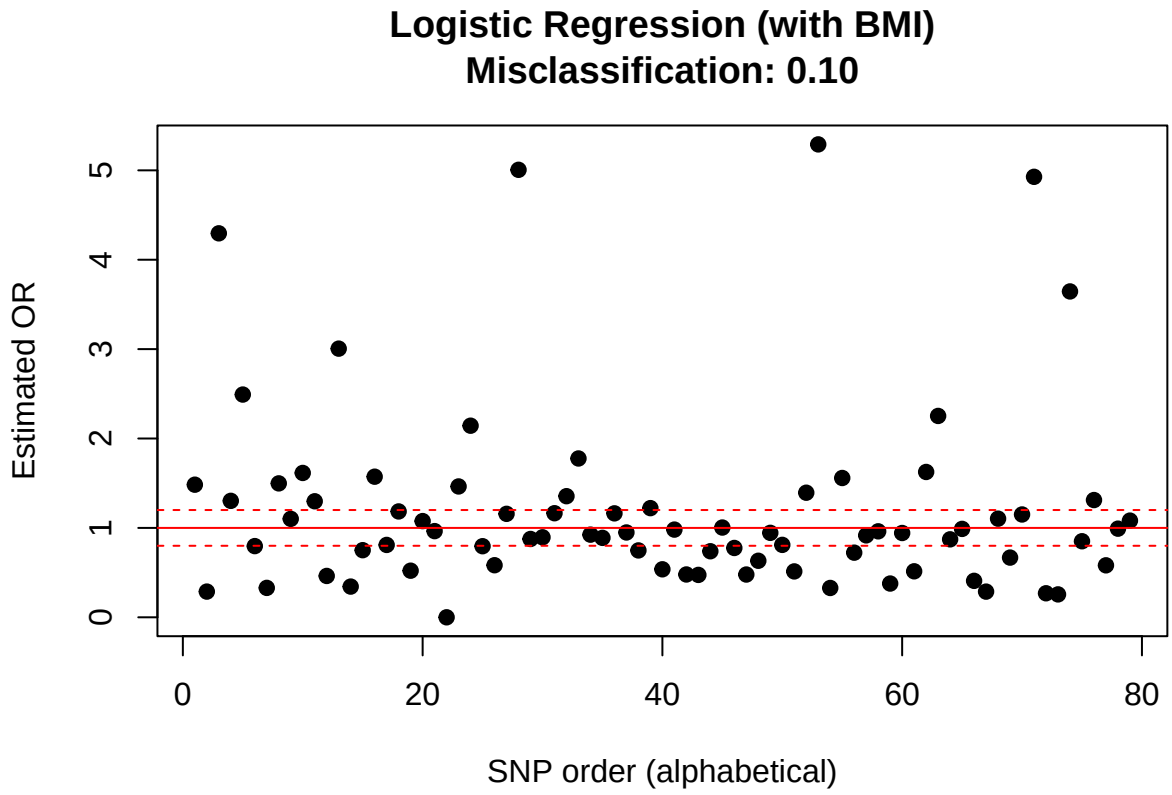
```
##          obs_response
## pred_vec2   0   1
##         0 241  29
##         1  25 236
```

```
# calculating the rate
1 - sum(diag(tab_mis2))/sum(tab_mis2)
```

```
## [1] 0.1016949
```

```
# dropping the parameters for intercept and bmi
coef_vec2_graph <- coef_vec2[-(which(coef_vec2 == coef_vec2["norm_BMI"] | coef_vec2 == coef_vec2["(Inte
```

```
plot(exp(coef_vec2_graph),
     main = "Logistic Regression (with BMI)\nMisclassification: 0.10",
     xlab = "SNP order (alphabetical)", ylab = "Estimated OR",
     pch = 19)
abline(h = 1.2, col = "red", lty = 2)
abline(h = 1.0, col = "red")
abline(h = 0.8, col = "red", lty = 2)
```




```
coef_vec2["norm_BMI"]
```

```
## norm_BMI
## 5.840162
```

After including the "norm_BMI" variable in the logistic regression equation, we see that more of the SNP coefficients have estimated ORs that lie outside of the range (0.8, 1.2). Additionally, the absolute distance of the estimated ORs of the SNPs from the range (0.8, 1.2) increases because the coefficient estimates are larger. The misclassification rate also dropped by 23%, suggesting that the model predictions have improved relative to the data. The difference between the graphs suggest that "norm_BMI" should be included in the logistic regression equation because of how the estimated ORs change once it is accounted for.

The magnitude of the coefficient for "norm_BMI" is 5.8401622.

1.D

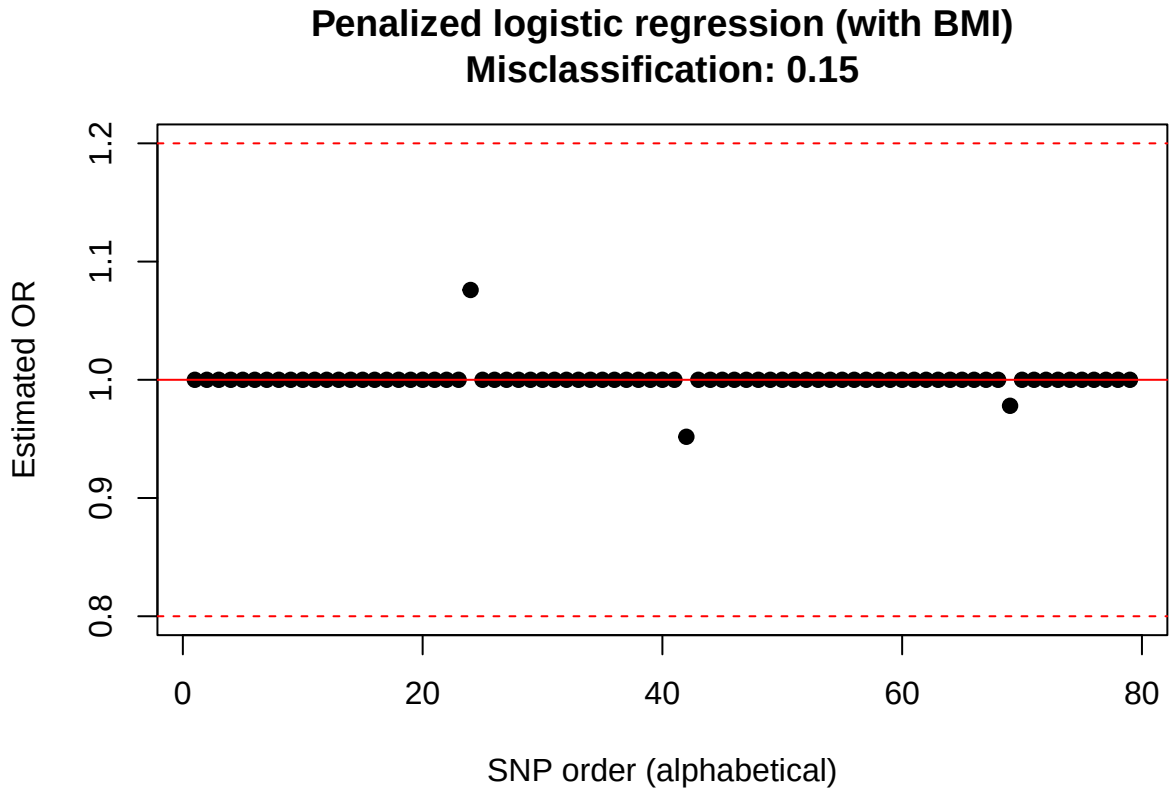
```
# cross-validation
# making the data frames for cv.regression
xs <- famuss[, -1]
y <- famuss[, "heart_disease"]

set.seed(10)
cv_glm_res <- glmnet::cv.glmnet(x = as.matrix(famuss[,-idx]), y = famuss[,idx],
                                family = "binomial", alpha = 1, intercept = T)

coef_vec3 <- output_coefficients(cv_glm_res, famuss, idx)

pred_vec3 <- output_predictions(cv_glm_res, famuss, idx)

# misclassification rate part 2
tab_mis3 <- table(pred_vec3, obs_response)
tab_mis3
```

```
##          obs_response
## pred_vec3   0   1
##         0 236  50
##         1  30 215
```

```
# calculating the rate
1 - sum(diag(tab_mis3))/sum(tab_mis3)
```

```
## [1] 0.1506591
```

```
# dropping the parameters for interceot and bmi
coef_vec3_graph <- coef_vec3[-(which(coef_vec3 == coef_vec3["norm_BMI"] | coef_vec3 == coef_vec3["(Inte
plot(exp(coef_vec3_graph),
     main = "Penalized logistic regression (with BMI)\nMisclassification: 0.15",
     xlab = "SNP order (alphabetical)", ylab = "Estimated OR",
     pch = 19, ylim = c(0.8, 1.2))
abline(h = 1.2, col = "red", lty = 2)
abline(h = 1.0, col = "red")
abline(h = 0.8, col = "red", lty = 2)
```




```
# magnitude of coefficient for norm_BMI
coef_vec3["norm_BMI"]
```

```
## norm_BMI
## 2.458888
```

Examining the graph associated with the penalized logistic regression that includes the BMI term, we see that virtually all the SNPs have estimated ORs of 1, and none of the estimated ORs are outside of the range (0.8, 1.2). This contrasts with the plot in 1.B where many of the estimated ORs lie outside this interval of values for the SNPs. Also, the misclassification rate increased slightly by approximately 5% compared to the logistic regression without a penalty (which is similar to the rate from part 1.C). The results of the Penalized logistic regression help to identify which coefficients are most important for prediction and should be included in the regression model. Since none of the estimated ORs lie outside the (0.8, 1.2), this suggests that the SNPs may not be useful for predicting heart disease.

Q2: Empirical Effects of Correlation on Lasso

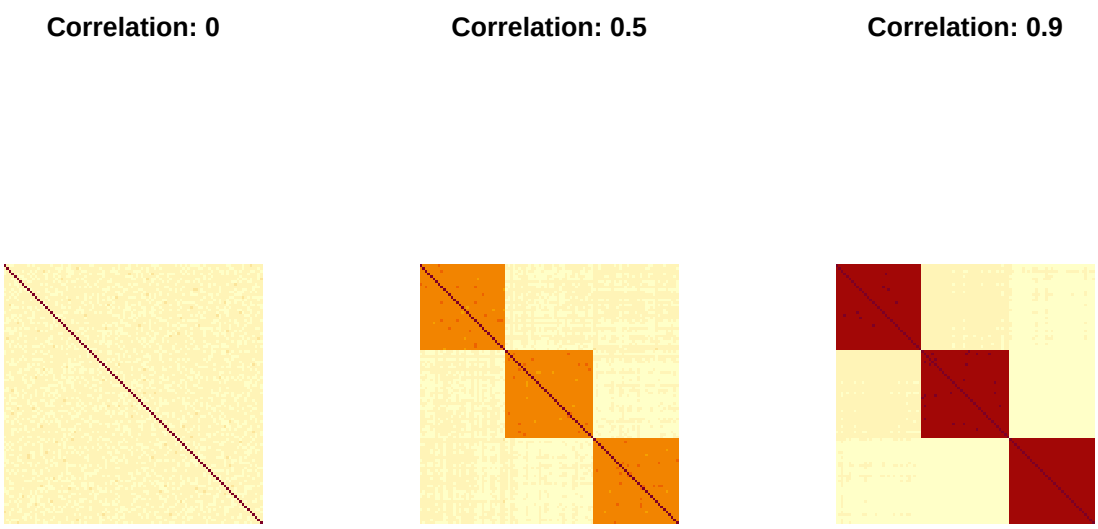
2.A

The function takes inputs for the rows (n = observations) and columns (p = predictors) of a matrix and has predefined values for "k" and "cor_within" for correlation values between the predictor values in the covariate matrix. It then creates a square matrix ("cor_mat") with p by p dimensions as well as an index vector of size p. The for loop creates k (k is 3) smaller matrices within the covariance matrix by replacing

values in the top-left, middle, and bottom-right sections with the "cor_within" value and then replaces all diagonal values with 1. It then creates a covariate matrix x from the "cor_mat" and random values the multivariate normal distribution based on the n and p inputs of the equation, along with a coefficient vector that inserts the value 5 into specific entries within the vector ("coef_truth") to represent the true responses. Finally, the function calculates values of a y vector (size p) based on matrix multiplication between x and the coefficient vector plus a random noise variable and displays x, y, and the coefficient vector in list format.

2.B

```
dat_2b_0 <- generate_data(n = 1000, p = 100, cor_within = 0)
dat_2b_5 <- generate_data(n = 1000, p = 100, cor_within = 0.5)
dat_2b_9 <- generate_data(n = 1000, p = 100, cor_within = 0.9)

# plotting the graph
par(mfrow = c(1,3))
plot_covariance(dat_2b_0$x, main = "Correlation: 0", axes = F)
plot_covariance(dat_2b_5$x, main = "Correlation: 0.5", axes = F)
plot_covariance(dat_2b_9$x, main = "Correlation: 0.9", axes = F)
```




```
par(mfrow = c(1,1))
```

As cor_within increases, the values of the data within the squares of the plot are more concentrated around the line as a result of a higher correlation value. Since the data is more concentrated for higher correlation,

the resulting box in the area of the data is a darker shade because larger data values are present. There is a lighter shade or no shade with lower correlation because there is more variation around the line and as a result, less magnitude of the data values in any given spot on the map and specifically around the correlation line. This mirrors a pairwise correlation matrix for multiple variables.

2.C

```
simulate_lasso <- function(n, cor_within) {
        # step 1
        dat <- generate_data(n = n, p = 2*n, cor_within = cor_within)
        dat_df <- as.data.frame(cbind(dat$x, dat$y))
        dat_idx <- ncol(dat_df)

        # step 2
        cvglmnet_res <- glmnet::cv.glmnet(x = as.matrix(dat_df[,-dat_idx]), y = dat_df[,dat_idx], family

        # step 3
        cvglmnet_res_coef <- output_coefficients(cvglmnet_res, dat_df, dat_idx)[-1]
        l2_error <- sqrt(sum((dat$coef_truth - cvglmnet_res_coef)^2))

        # step 4
        pred_error <- mean((dat$x %*% (dat$coef_truth - cvglmnet_res_coef))^2)
        list(l2_error = l2_error, pred_error = pred_error)
}
```

2.D

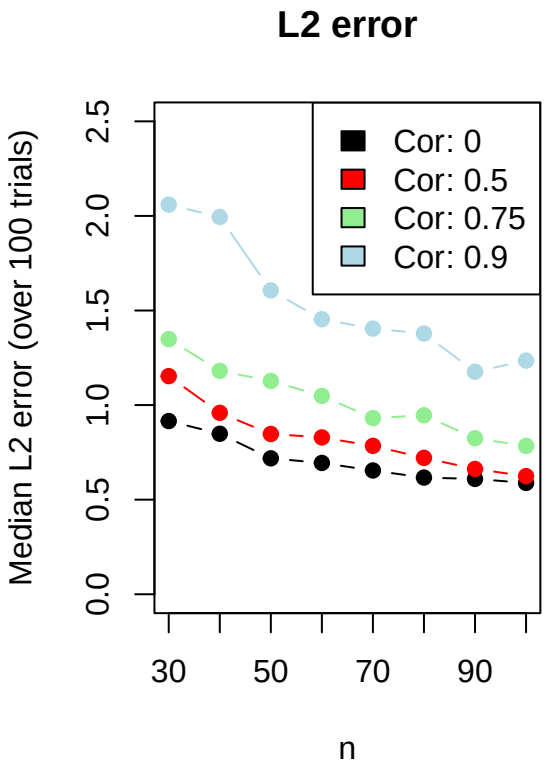
Part 1 - Simulation

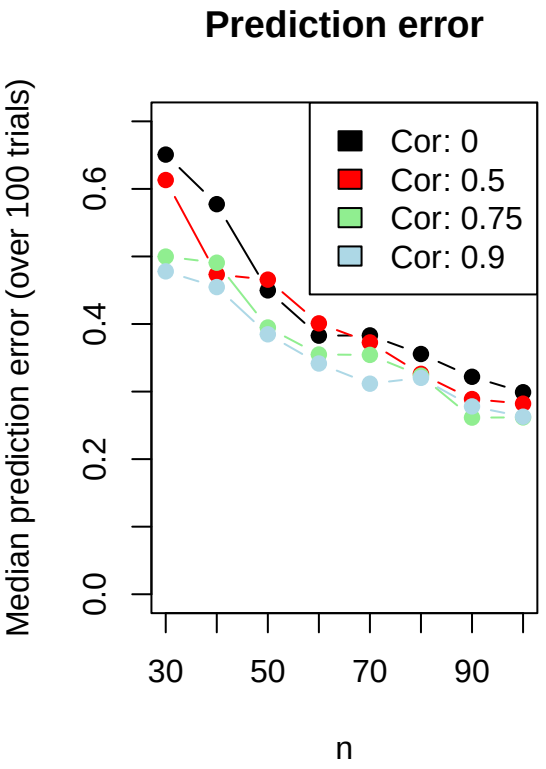
```
# part 1: simulation
# initializing the data frame to get values from for the simulation
df2d <- expand.grid(c(seq(30, 100, by = 10)), c(0, 0.5, 0.75, 0.9))
colnames(df2d) <- c("n", "cor_within")
# initializing the vectors that will collect the median values from the simulations
med_l2_vec <- rep(0, 32)
med_MSE_vec <- rep(0, 32)
# initializing vectors that will be used for the simulation
n_vec <- rep(0, 100)
n_vec2 <- rep(0, 100)

# simulation - outer loop * inner loop = 32 * 100 = 3200
for (i in seq(nrow(df2d))) { # outer loop is 32
        for (j in seq_along(n_vec)) { # inner loop is 100
                # using values in the simulate_lasso function from the data frame
                # for each simulation we store the values in an entry of the vector
                n_vec[j] <- simulate_lasso(df2d[i,1],df2d[i,2])[[1]]
                n_vec2[j] <- simulate_lasso(df2d[i,1],df2d[i,2])[[2]]
        }
        # after each iteration of the inner loop is complete, we take the median
        # of all the values from the simulation vector and store it in the entry
```

```
        # of these vectors that have the same length as the data frame
        med_l2_vec[i] <- median(n_vec)
        med_MSE_vec[i] <- median(n_vec2)
}
# adding the results of the vectors into the data frame
df2d$l2_vec <- med_l2_vec
df2d$pr_vec <- med_MSE_vec
```

Part 2 - Plotting the results of the simulation

```
# l2 vectors
n_values <- seq(30, 100, by = 10)
med_l2_cor0 <- df2d$l2_vec[df2d$cor_within == 0]
med_l2_cor05 <- df2d$l2_vec[df2d$cor_within == 0.5]
med_l2_cor075 <- df2d$l2_vec[df2d$cor_within == 0.75]
med_l2_cor09 <- df2d$l2_vec[df2d$cor_within == 0.9]
# pred vectors
med_MSE_cor0 <- df2d$pr_vec[df2d$cor_within == 0]
med_MSE_cor05 <- df2d$pr_vec[df2d$cor_within == 0.5]
med_MSE_cor075 <- df2d$pr_vec[df2d$cor_within == 0.75]
med_MSE_cor09 <- df2d$pr_vec[df2d$cor_within == 0.9]

par(mfrow = c(1,2))

# l2 plot
plot(x = n_values, y = med_l2_cor0, type = "b", pch = 19,
     xlab = "n", ylab = "Median L2 error (over 100 trials)",
     main = "L2 error", ylim = c(0, 2.5))
lines(x = n_values, y = med_l2_cor05, type = "b", col = "red", pch = 19)
lines(x = n_values, y = med_l2_cor075, type = "b", col = "lightgreen", pch = 19)
lines(x = n_values, y = med_l2_cor09, type = "b", col = "lightblue", pch = 19)
legend("topright", c("Cor: 0", "Cor: 0.5", "Cor: 0.75", "Cor: 0.9"),
       fill = c("black", "red", "lightgreen", "lightblue"))

# pred error plot
plot(x = n_values, y = med_MSE_cor0, type = "b", pch = 19,
     xlab = "n", ylab = "Median prediction error (over 100 trials)",
     main = "Prediction error", ylim = c(0, 0.7))
lines(x = n_values, y = med_MSE_cor05, type = "b", col = "red", pch = 19)
lines(x = n_values, y = med_MSE_cor075, type = "b", col = "lightgreen", pch = 19)
lines(x = n_values, y = med_MSE_cor09, type = "b", col = "lightblue", pch = 19)
legend("topright", c("Cor: 0", "Cor: 0.5", "Cor: 0.75", "Cor: 0.9"),
       fill = c("black", "red", "lightgreen", "lightblue"))
```






```
par(mfrow = c(1,1))
```

For the L2 error plot, the median L2 error tends to larger as the "cor_within" value gets closer to one for a given value of "n". However, in every case, the error decreases as the value of "n" increases. For the Prediction error plot, the median prediction error is relatively constant regardless of the value of "cor_within" for a given value of "n". In this case as well, the median prediction error decreases in all cases as "n" increases. These results illustrate the importance of examining the correlation between the predictor variables since the coefficient estimates could be noticeably wrong even if the predictions are accurate because of high correlation among the "X" variables.