Assignment 4

Henrik Olaussen

2023-10-24

Problem 1

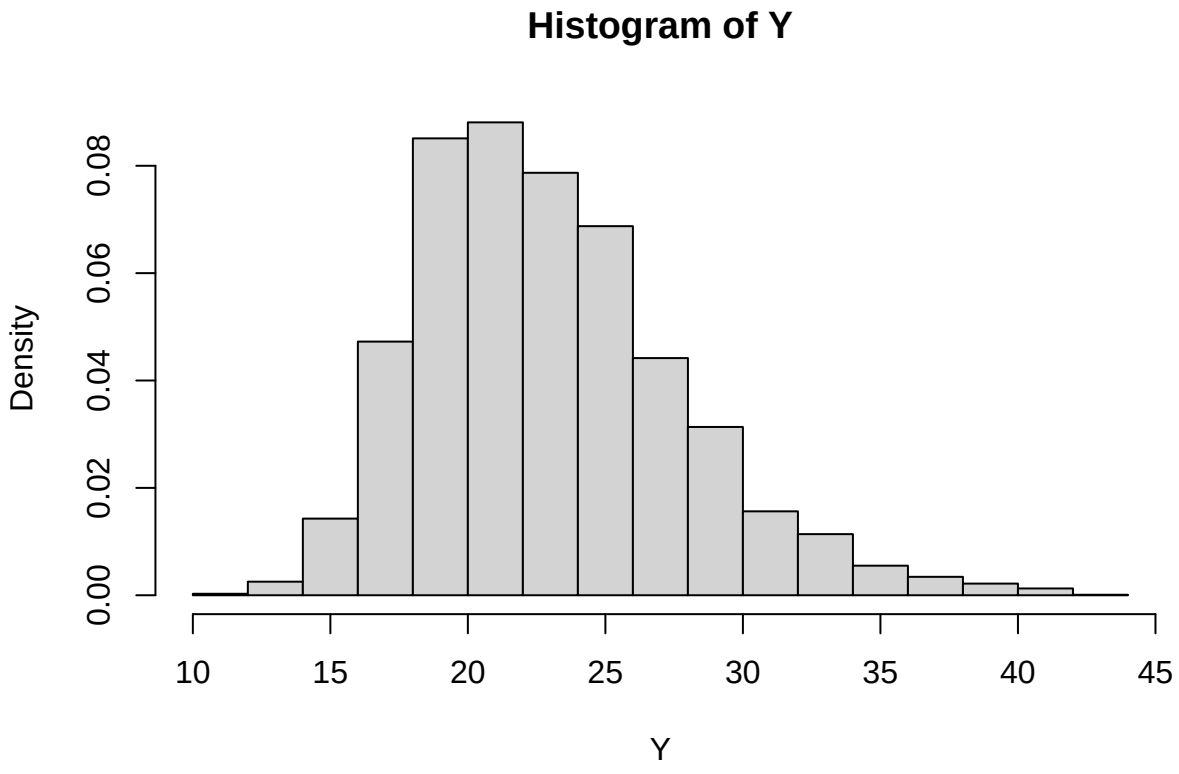
load data:

```
data1 = age_at_mar
Y = age_at_mar$age
```

a)

Here is a historgram of the data:

```
hist(Y,probability = TRUE)
```



The MLEs are given in each code chunk.

First, the Poisson model:

```
pois.model <- fitdistr(Y,'poisson')
print(pois.model)
```

```
##      lambda
##   23.44018793
##  ( 0.06508201)
```

Geometric:

```
geom.model <- fitdistr(Y,'geometric')
print(geom.model)
```

```
##        prob
##   0.0409162157
##  (0.0005386469)
```

Negative binomial:

```
negative_bin.model <- fitdistr(Y, 'negative binomial')
print(negative_bin.model)
```

```
##        size            mu
##   229.54823609    23.44071700
##  ( 23.98351114) (  0.06832585)
```

b)

In the well-specified case, we have that the distribution of the MLE $\hat{\theta}_n$ is asymptotically normal:

$$\hat{\theta}_n \overset{A}{\sim} N\left(\theta^*, \frac{1}{n}\hat{\mathbf{J}}_n(\hat{\theta}_n)\right)$$

where θ^* denotes the risk minimizer, and

$$\hat{\mathbf{J}}_n(\theta_n) = -\left\{\frac{1}{n}\sum_i^n \frac{\partial^2 f(Y_i \mid \mathbf{X}_i;\theta)}{\partial\theta\partial\theta^T}\right\}$$

Hence, the 90% CI for the MLE involves the quantiles of the normal distribution. Firstly, for the Poisson model:

```
alpha_quantile = 1.645

pois.model$estimate + c(-alpha_quantile*pois.model$sd, alpha_quantile*pois.model$sd)
```

```
##   lambda   lambda
## 23.33313 23.54725
```

Secondly, for the geometric model:

```
geom.model$estimate + c(-alpha_quantile*geom.model$sd, alpha_quantile*geom.model$sd)
```

```
##       prob       prob
## 0.04003014 0.04180229
```

Lastly, the negative binomial model:

```
#size
negative_bin.model$estimate[1] + c(-alpha_quantile*negative_bin.model$sd[1], alpha_quantile*negative_bi
```

```
##     size     size
## 190.0954 269.0011
```

```
#mu
negative_bin.model$estimate[2] + c(-alpha_quantile*negative_bin.model$sd[2], alpha_quantile*negative_bi
```

```
##       mu       mu
## 23.32832 23.55311
```

b)

AIC and BIC for the models.

Poisson model:

```
AIC(pois.model)
```

```
## [1] 32662.93
```

```
BIC(pois.model)
```

```
## [1] 32669.55
```

Geometric model:

```
AIC(geom.model)
```

```
## [1] 46216.28
```

```
BIC(geom.model)
```

```
## [1] 46222.9
```

Negative binomial:

```
AIC(negative_bin.model)
```

```
## [1] 32717.14
```

```
BIC(negative_bin.model)
```

```
## [1] 32730.37
```

We are looking for the smallest AIC/BIC. The AIC and BIC are two information criteria, both involving penalty terms to avoid overfitting. Normally, AIC tends to overfit the models more than the BIC, as the BIC penalizes overfitting more. As n approaching infinity, BIC converges in probability to the true model. This is not true for the AIC. Furthermore, the Poisson model is the model with the smallest BIC and AIC. Hence, one would choose the Poisson model.

c)

Moreover, we can fit continous models as well, supposing that age is continous.

Firstly, the exponential model:

```
exp.model <- fitdistr(Y,'exponential')
print(exp.model)
```

```
##        rate
##   0.0426617740
##  (0.0005734814)
```

Secondly, gamma model:

```
gamma.model <- fitdistr(Y,'gamma')
print(gamma.model)
```

```
##       shape         rate
##   25.91956979    1.10577503
##  ( 0.48959445) ( 0.02109002)
```

Lastly, the log-normal model:

```
log_normal.model <- fitdistr(Y,'log-normal')
print(log_normal.model)
```

```
##      meanlog        sdlog
##   3.135037411   0.195778868
##  (0.002631760) (0.001860935)
```

Furthermore, we can fit 90% CIs to each of the estimated parameters.

For the exponential model:

```
exp.model$estimate + c(-alpha_quantile*exp.model$sd, alpha_quantile*exp.model$sd)
```

```
##       rate       rate
## 0.04171840 0.04360515
```

For the gamma model:

```
#shape-parameter
gamma.model$estimate[1] + c(-alpha_quantile*gamma.model$sd[1], alpha_quantile*gamma.model$sd[1])
```

```
##    shape    shape
## 25.11419 26.72495
```

```
#rate
gamma.model$estimate[2] + c(-alpha_quantile*gamma.model$sd[2], alpha_quantile*gamma.model$sd[2])
```

```
##     rate     rate
## 1.071082 1.140468
```

For the log-normal:

```
#mean log
log_normal.model$estimate[1] + c(-alpha_quantile*log_normal.model$sd[1], alpha_quantile*log_normal.model
```

```
##  meanlog  meanlog
## 3.130708 3.139367
```

```
#standard deviation log
log_normal.model$estimate[2] + c(-alpha_quantile*log_normal.model$sd[2], alpha_quantile*log_normal.model
```

```
##     sdlog     sdlog
## 0.1927176 0.1988401
```

d)

Next, the AIC and BIC of the continous models. Firstly, the exponential model:

```
AIC(exp.model)
```

```
## [1] 45983.47
```

```
BIC(exp.model)
```

```
## [1] 45990.09
```

Gamma-model:

```
AIC(gamma.model)
```

```
## [1] 32465.44
```

```
BIC(gamma.model)
```

```
## [1] 32478.67
```

log-normal:

```
AIC(log_normal.model)
```

```
## [1] 32358.05
```

```
BIC(log_normal.model)
```

```
## [1] 32371.29
```

The log-normal model is the model with the smallest AIC and BIC. Hence, this is the best model amoung the continous models, according to AIC and BIC. Compared to the AIC and BIC of the discrete models, the log-normal has the overall smallest AIC and BIC values.

e)

Moreover, we can calculate the 90% CI's under misspecification for the best models from b) and d). Thus, we use the Poisson model and the log-normal model. Under misspecification, we make use of the sandwich covariance estimator to get hold of the standard errors. In other words, we have to find an estimate for $A^{-1}BA^{-1}$ (under correct specification we have $A^{-1} = -B$). B is given by:

$$\hat{B}_n = \frac{1}{n}\sum_{i=1}^{n} \{\partial_\theta \log f(Y_i; \therefore)\} \{\partial_\theta \log f(Y_i; \therefore)\}^\top$$

Sandwich estimator for the poisson model is calculated below:

```
library(numDeriv)
# Set up log likelihood function
logf_pois <- function(para,obs) {
  dpois(Y[obs], lambda = para,log=TRUE)
}

B_est.pois <- matrix(0,1,1)
for (ii in 1:pois.model$n) {
  Grad <- grad(logf_pois, pois.model$estimate, obs=ii)
  B_est.pois <- B_est.pois + Grad%*%t(Grad)
}

# scale Best
B_est.pois <- B_est.pois/pois.model$n

#sandwich estimator:
Ainv.pois <- pois.model$n*pois.model$vcov
sandwich.pois <-  Ainv.pois%*%B_est.pois%*%Ainv.pois
```

Thus, we have that the 90% CI for the MLE of lambda is (we have asymptotic normality also for the case where with the sandwich estimator):

```
se.pois <- sqrt(diag(sandwich.pois)/pois.model$n)
pois.model$estimate + c(-alpha_quantile * se.pois, alpha_quantile * se.pois)
```

```
##   lambda   lambda
## 23.33579 23.54458
```

Furthermore, we do the same calculation for the log-normal model:

```
logf_lognorm <- function(para,obs) {
  dlnorm(Y[obs], meanlog = para[1], sdlog = para[2],log=TRUE)
}


B_est.lognorm <- matrix(0,2,2)
for (ii in 1:log_normal.model$n) {
  Grad <- grad(logf_lognorm, log_normal.model$estimate, obs=ii)
  B_est.lognorm <- B_est.lognorm + Grad%*%t(Grad)
}

# scale Best
B_est.lognorm <- B_est.lognorm/log_normal.model$n

#sandwich estimator:
Ainv.lognorm <- log_normal.model$n*log_normal.model$vcov
sandwich.lognorm <-  Ainv.lognorm%*%B_est.lognorm%*%Ainv.lognorm
```

The 90% CI:

```
se.lognorm <- sqrt(diag(sandwich.lognorm)/log_normal.model$n)

#for the meanlog
log_normal.model$estimate[1] + c(-alpha_quantile * se.lognorm[1], alpha_quantile * se.lognorm[1])
```

```
##  meanlog  meanlog
## 3.130708 3.139367
```

```
#for the sdlog
log_normal.model$estimate[2] + c(-alpha_quantile * se.lognorm[2], alpha_quantile * se.lognorm[2])
```

```
##     sdlog     sdlog
## 0.1927606 0.1987972
```

Problem 2

```
data2 <- na.omit(gss2010)
```

a)

Firstly, the mean of the logistic model is characterized by the logistic function:

$$\pi_{\mathbf{x}} = \mu(\beta_0 + \beta^T \mathbf{x}) = \frac{\exp(\beta_0 + \beta^T \mathbf{x})}{1 + \exp(\beta_0 + \beta^T \mathbf{x})}$$

where $\beta^T = [\beta_0, \beta_{\text{hrsrelax}}, \beta_{\text{mentlhlth}}, \beta_{\text{hrs}}, \beta_{\text{degree}}]^T$

Thus, the pmf is:

$$f\left(y \mid \mathbf{x}, \underline{\beta}\right) = \left\{\frac{\exp(\beta_0 + \beta^T \mathbf{x})}{1 + \exp(\beta_0 + \beta^T \mathbf{x})}\right\}^y \left\{\frac{1}{1 + \exp(\beta_0 + \beta^T \mathbf{x})}\right\}^{1-y}$$

with the given coefficient vector β.

b)

```
logistic_glm <- glm(grass ~ ., data = data2, family = binomial(link = "logit"))

summary(logistic_glm)
```

```
##
## Call:
## glm(formula = grass ~ ., family = binomial(link = "logit"), data = data2)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.624  -1.133  -1.021   1.209   1.411
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)           0.044560   0.320067   0.139 0.889276
## hrsrelax              0.001996   0.030191   0.066 0.947295
## mntlhlth             -0.002391   0.011136  -0.215 0.830013
## hrs1                 -0.005595   0.005346  -1.047 0.295247
## degreeGRADUATE       -0.085870   0.267606  -0.321 0.748300
## degreeHIGH SCHOOL     0.108731   0.199283   0.546 0.585335
## degreeJUNIOR COLLEGE  0.290539   0.312342   0.930 0.352271
## degreeLT HIGH SCHOOL  1.006468   0.303724   3.314 0.000921 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 956.45  on 689  degrees of freedom
## Residual deviance: 940.46  on 682  degrees of freedom
## AIC: 956.46
##
## Number of Fisher Scoring iterations: 4
```

```r
unique(data2$degree)
```

```
## [1] BACHELOR       HIGH SCHOOL    JUNIOR COLLEGE GRADUATE       LT HIGH SCHOOL
## Levels: BACHELOR GRADUATE HIGH SCHOOL JUNIOR COLLEGE LT HIGH SCHOOL
```

Since degree is a categorical variable, R calculates a coefficient for each of the categories, with Bachelor as base reference category. The degreeBachelor estimate is thus given in the intercept. Furthermore, we can calculate 90% asymptotic CIs for the estimated coefficients. This can be done trough the function coefci(). Since we do not know if the model is misspecified, vcov = sandwich is used:

```r
alpha = 0.1
coefci(logistic_glm, vcov = sandwich, level = 1-alpha)
```

```
##                               5 %        95 %
## (Intercept)          -0.48810996 0.577229306
## hrsrelax             -0.04756686 0.051558269
## mntlhlth             -0.02076347 0.015981974
## hrs1                 -0.01453661 0.003346319
## degreeGRADUATE       -0.52630074 0.354560991
## degreeHIGH SCHOOL    -0.21991750 0.437378895
## degreeJUNIOR COLLEGE -0.22499701 0.806074176
## degreeLT HIGH SCHOOL  0.50789394 1.505042760
```

c)

First, we fit the model with degree as the only covariate as our null model.

```r
null_model = glm(grass ~ degree, data = data2, family = binomial(link = "logit"))

summary(null_model)
```

```
##
## Call:
## glm(formula = grass ~ degree, family = binomial(link = "logit"),
##     data = data2)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.540  -1.145  -1.066   1.210   1.293
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -0.19416    0.16140  -1.203 0.229000
## degreeGRADUATE       -0.07411    0.26702  -0.278 0.781367
## degreeHIGH SCHOOL     0.11817    0.19679   0.600 0.548184
## degreeJUNIOR COLLEGE  0.29952    0.31052   0.965 0.334758
## degreeLT HIGH SCHOOL  1.01514    0.30250   3.356 0.000791 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 956.45  on 689  degrees of freedom
## Residual deviance: 941.69  on 685  degrees of freedom
## AIC: 951.69
##
## Number of Fisher Scoring iterations: 4
```

The test can be performed using waldtest():

```
waldtest(null_model, logistic_glm, vcov = sandwich)
```

```
## Wald test
##
## Model 1: grass ~ degree
## Model 2: grass ~ hrsrelax + mntlhlth + hrs1 + degree
##   Res.Df Df      F Pr(>F)
## 1    685
## 2    682  3 0.3963 0.7557
```

Our p-value is large. As a result, there is no evidence that we can reject the null model. Hence, the model with degree as the only covariate might be statistically better than the model from b).

Furthermore, we can create a simultaneous 99% confidence set. In order to do this, we can use the bonferroni correction, as done below.

```
m = 4
alpha_star = 0.01

coefci(null_model, vcov = sandwich, level = 1-alpha_star/m)
```

```
##                          0.125 %  99.875 %
## (Intercept)          -0.6821276 0.2938156
## degreeGRADUATE       -0.8813984 0.7331824
## degreeHIGH SCHOOL    -0.4767981 0.7131383
## degreeJUNIOR COLLEGE -0.6392823 1.2383154
## degreeLT HIGH SCHOOL  0.1005822 1.9296909
```

d)

Moreover, we can look at the AIC and BIC to find the best model. Below is the AIC and BIC of the null model:

```
AIC(null_model)
```

```
## [1] 951.6892
```

```
BIC(null_model)
```

```
## [1] 974.3727
```

And the full model:

```
AIC(logistic_glm)
```

```
## [1] 956.4589
```

```
BIC(logistic_glm)
```

```
## [1] 992.7524
```

Both values are smaller for the reduced model. Hence, according to the AIC and BIC, the best fit is the reduced model. This is the same result as our previous hypothesis test.

e)

Till now, the logit mean function has been used. Now we can consider whether the logistic mean function (logit) is better than e.g. the probit link function which has the mean function given below (Φ is the cumulative Gaussian distribution):

$$\mathrm{E}\left[Y|X=x_4\right]=\Phi\left(\beta_0+\beta_{\text{degree}}x_4\right)=\int_{-\infty}^{\beta_0+\beta_{\text{degree}}x_4}\frac{1}{\sqrt{2\pi}}\exp(-\frac{1}{2}t^2)dt$$

where $\beta_{\text{degree}}x_4=\beta_{\text{degree}\text{GRADUATE}}$ if x_4 is related to the category graduate etc.

Moreover, the fitted model can be seen below:

```
probit <- glm(grass ~ degree, data = data2, family = binomial(link = "probit"))

summary(probit)
```

```
##
## Call:
## glm(formula = grass ~ degree, family = binomial(link = "probit"),
##     data = data2)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.540  -1.145  -1.066   1.210   1.293
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -0.12159    0.10094  -1.205 0.228374
## degreeGRADUATE       -0.04631    0.16680  -0.278 0.781305
## degreeHIGH SCHOOL     0.07398    0.12314   0.601 0.548017
## degreeJUNIOR COLLEGE  0.18760    0.19440   0.965 0.334530
## degreeLT HIGH SCHOOL  0.63008    0.18485   3.409 0.000653 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 956.45  on 689  degrees of freedom
## Residual deviance: 941.69  on 685  degrees of freedom
```

```
## AIC: 951.69
##
## Number of Fisher Scoring iterations: 4
```

AIC and BIC for the probit-model:

```
AIC(probit)
```

```
## [1] 951.6892
```

```
BIC(probit)
```

```
## [1] 974.3727
```

For the logit-model:

```
AIC(null_model)
```

```
## [1] 951.6892
```

```
BIC(null_model)
```

```
## [1] 974.3727
```

The AIC and BIC are equal for both models. Moreover, using the logit model as reference, the coefficient values states how likely it is for a given person to vote legal/not legal relative to a person with a bachelors degree (included in the intercept). A negative value means that a person is less likely to vote for legal than someone with a bachelor's degree, while a positive value states that the person is more likely to vote legal. Since the coefficients in a logit-model is related to the mean-function given earlier, the value of $\exp(\beta_j)$ is known as the odds-ratio. E.g. an odds ratio of $\exp(\beta_{\text{degreeHIGHSCHOOL}}) = \exp(0.11817) = 1.125$ means that a person with only a high school degree is 1.125 times more likely to vote for legal than someone with a bachelors degree.

Problem 3

a)

There is one value that is below 0. This is not possible for a gamma distribution, so this row is removed from the dataset. Moreover, the link = "identity" gives the desired mean model

$$\mu(x) = \exp(x)$$

```
data3 = Mroz

for (i in 1:length(Mroz$inc)) { #remove the negatice row
  if(Mroz$inc[i] < 0) {
    data3 = data3[-i,]
  }
}

gamma_model <- glm(inc ~., data = data3, family = Gamma(link = "identity"))
```

Moreover, we can give individual asymptotic 90% confidence intervals for each parameter. In case the model is misspecified, we use the sandwich estimator.

```
coefci(gamma_model, vcov = sandwich, level = 1-alpha)
```

```
##                    5 %       95 %
## (Intercept)  4.0112042 11.8726715
## lfpyes      -4.3133835 -1.8309767
## k5          -1.6811693  1.0805042
## k618         0.2113555  1.0372122
## age          0.1158243  0.2720895
## wcyes        1.3735943  4.6213050
## hcyes        5.8196726  8.6288270
## lwg          0.3073786  1.9449579
```

In addition, we can test the null hypothesis that Y is independent of the covariates vs the alternative that Y is dependent on the covariates.

```
waldtest(gamma_model, vcov = sandwich)
```

```
## Wald test
##
## Model 1: inc ~ lfp + k5 + k618 + age + wc + hc + lwg
## Model 2: inc ~ 1
##   Res.Df Df      F    Pr(>F)
## 1    744
## 2    751 -7 24.853 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The p-value is very low, meaning that there is evidence against the null hypothesis. Hence, the conclusion of the test is that Y is dependent on some of the covariates.

b)

Furthermore, one can use the function bestglm() to find the model with the best AIC.

```
Xy <- as.data.frame(c(data3))

bestmodel_AIC <- bestglm(Xy, family = Gamma(link = "inverse"), IC = "AIC")
```

```
## Morgan-Tatar search since family is non-gaussian.
```

```
bestmodel_AIC
```

```
## AIC
## BICq equivalent for q in (0.771263234890237, 0.964518037404803)
## Best Model:
##                  Estimate   Std. Error   t value     Pr(>|t|)
## (Intercept)  0.0793914640 0.0061525822 12.903763 1.584551e-34
```

```
## lfpyes       0.0084287544 0.0018368774  4.588632 5.234267e-06
## k618        -0.0015086651 0.0007185126 -2.099706 3.609051e-02
## age         -0.0004154885 0.0001183682 -3.510135 4.748093e-04
## wcyes       -0.0064043028 0.0022629492 -2.830069 4.779059e-03
## hcyes       -0.0171098196 0.0022040726 -7.762820 2.744601e-14
## lwg         -0.0032639743 0.0017468583 -1.868483 6.208710e-02
```

One can also use BIC to find the best model:

```
bestmodel_BIC <- bestglm(Xy, family = Gamma(link = "inverse"), IC = "BIC")
```

```
## Morgan-Tatar search since family is non-gaussian.
```

```
bestmodel_BIC
```

```
## BIC
## BICq equivalent for q in (0.104346517262481, 0.769157322523084)
## Best Model:
##                  Estimate   Std. Error   t value     Pr(>|t|)
## (Intercept)  0.0708816555 0.0051590088 13.739394 1.836119e-38
## lfpyes       0.0079750485 0.0018123718  4.400338 1.238169e-05
## age         -0.0003356838 0.0001094737 -3.066340 2.245096e-03
## wcyes       -0.0071758141 0.0021926086 -3.272729 1.114130e-03
## hcyes       -0.0174718324 0.0022201025 -7.869831 1.247525e-14
```

One can see that the optimal BIC gives a model with fewer coefficients than the optimal AIC. This isexpected as the BIC penalizes larger model more than the AIC. Furthermore, we can create an asymptotic simultanous 90% confidence sets for the two models. Like before, the bonferroni update can be used. The corrections are done below:

```
alpha_star_2 = 0.1
#AIC model
m_AIC = 6

alpha_AIC = alpha_star_2/m_AIC

#BIC model
m_BIC = 4

alpha_BIC = alpha_star_2/m_BIC
```

We get the following two simultanous 90% confidence sets:

```
#AIC
coefci(bestmodel_AIC$BestModel, level = 1-alpha_AIC)
```

```
##                   0.833 %      99.167 %
## (Intercept)  0.0646623065  0.0941206215
## lfpyes       0.0040313070  0.0128262018
## k618        -0.0032287697  0.0002114396
## age         -0.0006988597 -0.0001321173
## wcyes       -0.0118217575 -0.0009868482
## hcyes       -0.0223863247 -0.0118333144
## lwg         -0.0074459176  0.0009179691
```

```
#BIC
coefci(bestmodel_BIC$BestModel, level = 1-alpha_BIC)
```

```
##                    1.25 %       98.75 %
## (Intercept)  0.0593182391  8.244507e-02
## lfpyes       0.0039127935  1.203730e-02
## age         -0.0005810585 -9.030902e-05
## wcyes       -0.0120903331 -2.261295e-03
## hcyes       -0.0224479762 -1.249569e-02
```

Problem 4

a)

Fit the data:

```
data4 = Arrests

logistic_glm_2 <- glm(released ~., data = data4, family = binomial(link = "logit"))

summary(logistic_glm_2)
```

```
##
## Call:
## glm(formula = released ~ ., family = binomial(link = "logit"),
##     data = data4)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3909   0.3579   0.4320   0.6047   1.7067
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  9.371821  56.717803   0.165    0.869
## colourWhite  0.389109   0.085663   4.542 5.56e-06 ***
## year        -0.004218   0.028379  -0.149    0.882
## age          0.002236   0.004631   0.483    0.629
## sexMale      0.007317   0.150189   0.049    0.961
## employedYes  0.757302   0.084735   8.937  < 2e-16 ***
## citizenYes   0.576519   0.104246   5.530 3.20e-08 ***
## checks      -0.364101   0.025984 -14.013  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4776.3  on 5225  degrees of freedom
## Residual deviance: 4299.1  on 5218  degrees of freedom
## AIC: 4315.1
##
## Number of Fisher Scoring iterations: 5
```

Perform the test:

```
waldtest(logistic_glm_2, vcov = sandwich)
```

```
## Wald test
##
## Model 1: released ~ colour + year + age + sex + employed + citizen + checks
## Model 2: released ~ 1
##   Res.Df Df      F    Pr(>F)
## 1   5218
## 2   5225 -7 62.975 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The p-value is very small, hence there is strong evidence that we can reject the null hypothesis, meaning that the there is strong evidence that the model is dependent on some of the covariates.

The estimated mean is:

$$\mu(\theta^T \mathbf{x}) = \frac{\exp(\theta^T \mathbf{x})}{1 + \exp(\theta^T \mathbf{x})}$$

Where $\theta = [\beta_0, \beta_{\text{colourWhite}}, .., \beta_{\text{checks}}]$. $\mathbf{x}$ decides whether we have a person that is female/male, black/white, employed or not, etc.

b)

Finding a subset of covariates with respect to BIC:

```
X = data4[2:8]
Y = data4[1]

Xy_2 = cbind(X,Y)
best_BIC_2 <-bestglm(Xy_2, family = binomial(link = "logit"), IC = "BIC")
```

```
## Morgan-Tatar search since family is non-gaussian.
```

```
best_BIC_2
```

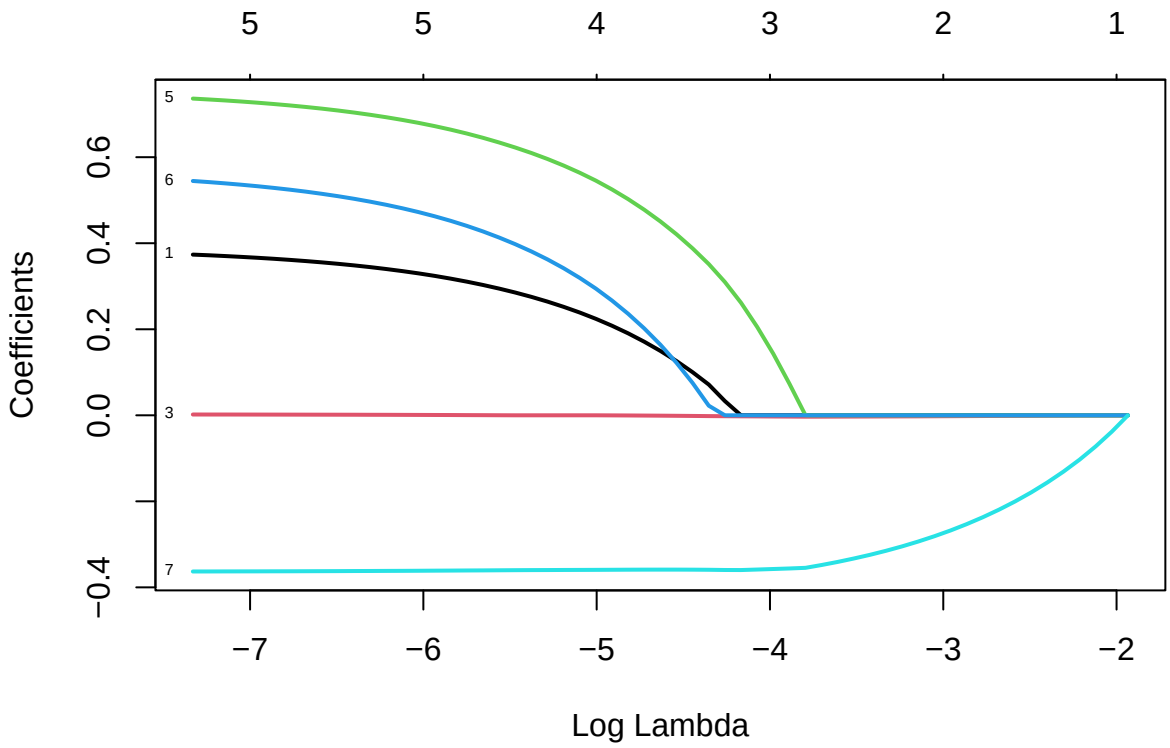
```
## BIC
## BICq equivalent for q in (0.00270773785704703, 0.984724839321298)
## Best Model:
##                Estimate Std. Error    z value     Pr(>|z|)
## (Intercept)  1.0047432 0.12740849   7.885999 3.120294e-15
## colourWhite  0.3891455 0.08521187   4.566799 4.952293e-06
## employedYes  0.7536711 0.08398239   8.974156 2.855328e-19
## citizenYes   0.5683892 0.09915602   5.732271 9.909461e-09
## checks      -0.3628292 0.02573103 -14.100840 3.752737e-45
```

The model with the optimal BIC is shown above.

c)

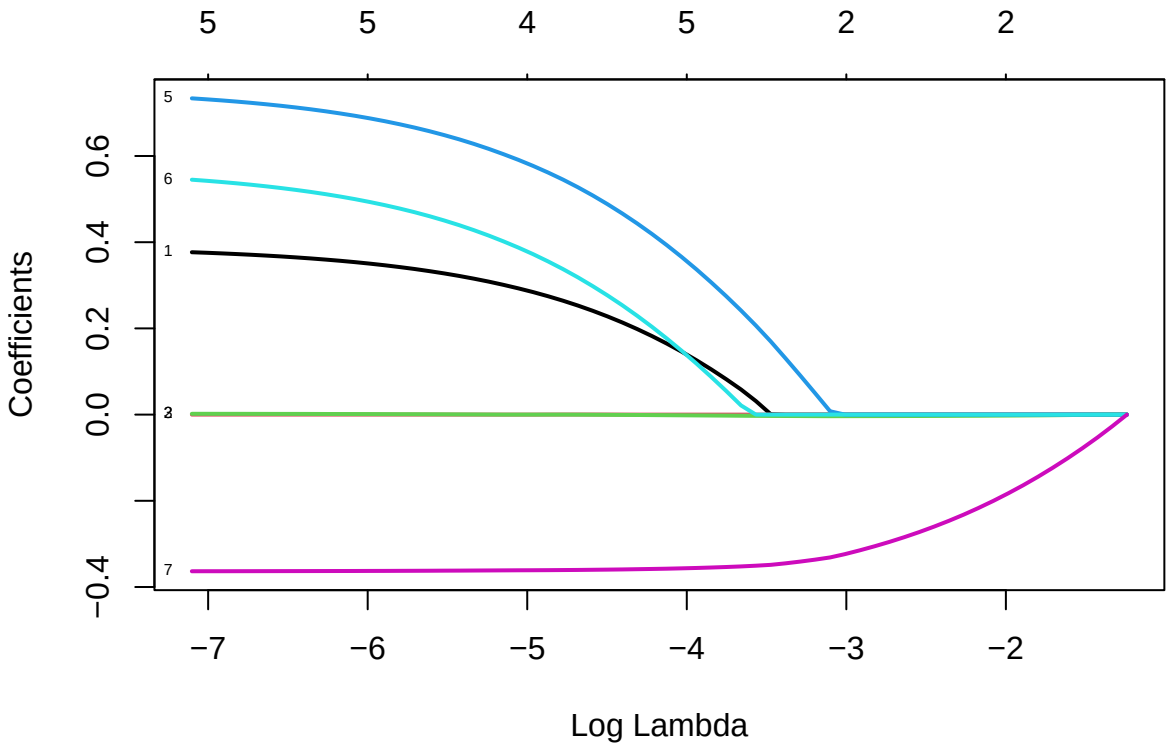
Furthermore, regularization can be applied for best model selection. Two methods are LASSO and Elastic net. Both methods makes use of a penalty on the coefficients. In both cases, the parameter lambda chooses how much we penalize the coefficients. By using the glmnet-function, we can try different values of lambda to find the best model. Firstly, we use the LASSO.

```
#model.matrix to fix the categorical variables
xx <- model.matrix(released ~., data = data4)[,-1]
yy <- data4$released
lasso <- glmnet(x = xx, y = yy, family = binomial(link = "logit"), standardize = FALSE, alpha=1)

plot(lasso,
     xvar='lambda',
     lwd=2,
     label = TRUE,
     cex = 3)
```



Secondly, Elastic net:

```
elastic_net <- glmnet(x = xx, y = yy, family = binomial(link = "logit"), standardize = FALSE, alpha=0.5)

plot(elastic_net,
     xvar='lambda',
     lwd=2,
     label = TRUE,
     cex = 3)
```



d)

Next, we want to find our optimal lambda values. This can be done through cross validation in the cv.glmnet()-function.

```
#for the lasso
lasso_cv <- cv.glmnet(x = xx, y = yy,
                      family = binomial(link = "logit"),
                      standardize = FALSE,
                      type.measure = 'deviance',
                      alpha=1,
                      nfold=5)

lasso_best <- glmnet(x = xx, y = yy, family = binomial(link = "logit"), standardize = FALSE, lambda = la

lasso_best$beta
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## colourWhite  0.368307199
## year         .
## age          0.001722859
## sexMale      .
## employedYes  0.729425327
## citizenYes   0.536028247
```

```
## checks      -0.362841339
```

```
#for the elastic net
elastic_net_cv <- cv.glmnet(x = xx, y = yy,
                      family = binomial(link = "logit"),
                      standardize = FALSE,
                      type.measure = 'deviance',
                      alpha=0.5,
                      nfold=5)

elastic_net_best <- glmnet(x = xx, y = yy, family = binomial(link = "logit"), standardize = FALSE, lambd

elastic_net_best$beta
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## colourWhite  0.356833393
## year         .
## age          0.001364442
## sexMale      .
## employedYes  0.698726890
## citizenYes   0.505913615
## checks      -0.362887036
```

In the code chunks above, cross validation has been used to find the optimal value of lambda. This is the value of lambda that minimize the MSE. One can see that the in the two resulting best models, the same covariates has been removed. However, the estimated coefficient-values are slightly different. Moreover, the mean functions are given below. Note that $\hat{\theta}$ holds the same coefficients for both models (but not the same values to each coefficient of course).

$$\mu(\hat{\theta}_{\lambda,n}^{(\text{Lasso})\text{T}}\mathbf{x}) = \frac{\exp(\hat{\theta}_{\lambda,n}^{(\text{Lasso})\text{T}}\mathbf{x})}{1+\exp(\hat{\theta}_{\lambda,n}^{(\text{Lasso})\text{T}}\mathbf{x})}$$

$$\mu(\hat{\theta}_{\lambda,n}^{(\text{E.net})\text{T}}\mathbf{x}) = \frac{\exp(\hat{\theta}_{\lambda,n}^{(\text{E.net})\text{T}}\mathbf{x})}{1+\exp(\hat{\theta}_{\lambda,n}^{(\text{E.net})\text{T}}\mathbf{x})}$$

e)

Lastly, we can draw inference about the parameters found to be non-zero through lasso and elastic-net by fitting the refitting the models and conducting t-tests. Here, the models are likely to be misspecified so we use the sandwich covariance estimator.

```
glm_refit <- glm(released ~., data = data4[,-which(lasso_best$beta==0)-1], family = binomial(link = "log

# Output misspecification adjusted tests
coeftest(glm_refit,vcov=sandwich)
```

```
##
## z test of coefficients:
##
```

```
##                Estimate Std. Error  z value  Pr(>|z|)
## (Intercept)  0.9470313  0.1854214   5.1075 3.265e-07 ***
## colourWhite  0.3899812  0.0856256   4.5545 5.251e-06 ***
## age          0.0022100  0.0049177   0.4494    0.6531
## employedYes  0.7574835  0.0855836   8.8508 < 2.2e-16 ***
## citizenYes   0.5716575  0.1010817   5.6554 1.555e-08 ***
## checks      -0.3637767  0.0263597 -13.8005 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Firstly, one can observe that the values of the coefficients in the refitted model are slightly larger than those of lasso/elastic net. This is as expected because of the penalization used in the regularization methods. Moreover, all p-values are rather small except the value of age, indicating that this covariate is not significant to the model. However, this could be because of multicollinearity, meaning that age is highly correlated with another covariates in our refitted model. Consequently, age might be significant, but due to correlation, this is not visible through the test above.