GLM

Part 1: Poisson regression

In this task we will explore how to apply generalized linear model (GLM) in R. We will use a data set where the dependent variable is number of doctor visits (docvis). We can see from the histogram of the data that a linear regression model is likely not going to fit well, being count data with a monotonically falling pmf (probability mass function).

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.1.1
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
```

```
## Warning: package 'tibble' was built under R version 4.1.1
```

```
## Warning: package 'tidyr' was built under R version 4.1.1
```

```
## Warning: package 'readr' was built under R version 4.1.1
```

```
## Warning: package 'purrr' was built under R version 4.1.1
```

```
## Warning: package 'stringr' was built under R version 4.1.1
```

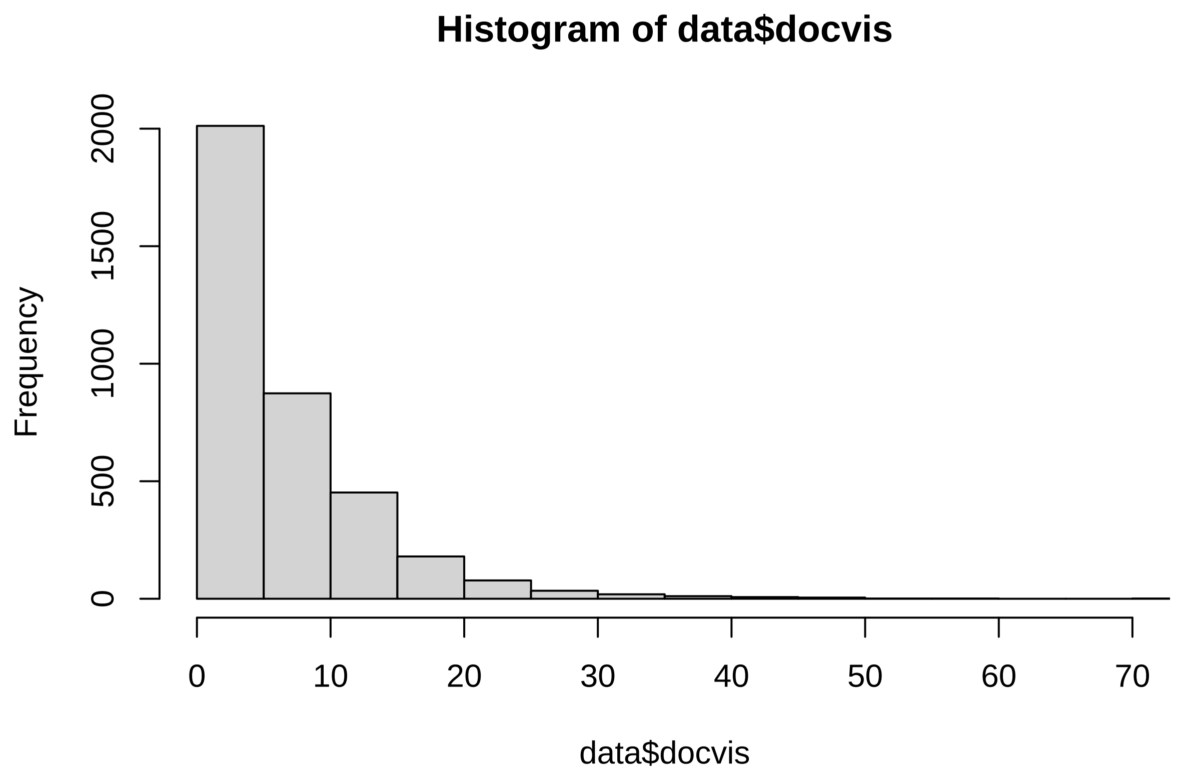
```
## Warning: package 'forcats' was built under R version 4.1.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(haven)
```

```
## Warning: package 'haven' was built under R version 4.1.1
```

```
data <- read_dta(file = "mus17data.dta")
data <- dplyr::select(data, c(age,medicaid,private,female,income,docvis,educyr,actlim,totchr))
hist(data$docvis,breaks=50,xlim=c(0,70)) # truncate for visualisation
```



task 1a)

Use the `glm` function to train a regression model with count response, $docvis_i \in \mathbb{N}_0$. Read `help(glm)` to set up `model.poisson` so that one performs Poisson regression on `docvis` including all other covariates. Which link function is assumed for Poisson regression in `glm` if one do not explicitly specify `link` argument?

```
model.poisson = glm(docvis~.,family="poisson",data=data)

task1a.linkfunction <- "The link that is assumed for poisson is log"
```

task 1b)

In the lecture *AIC* was used to show highlight imporvement of fit for GLM moldes. Now we will conduct stepwise model selection, based on *AIC*, to select covariates. To do this we will use the library `MASS` and the function `stepAIC` using backward selection. Backward selection works as follows:

- Step 1: Select the full model (the largest model one allows), and compute the *AIC* for the model.
- Step 2: Remove one covariate at the time, and compute the *AIC* for each reduced model.
- Step 3: If a rescued model has lower *AIC* than the full model go back to Step 3 with the reduced model being the new full model. Otherwise return the full model.

You can read more about both forward and backward selection in [ESL] section 3.3 p.57-61.

Use `help(stepAIC)` to perform backward selection for Poisson regression. Hint since we are using backwards direction we don't set the `scope` argument since the default options suffices here.

Additionally, find answers to the following two questions:

1. Is the model selected by backward selection guaranteed to have the smallest AIC among all Poisson regressions that we can train with the predictors at our disposal? Motivate your answer!
2. What is the difference in AIC between the selected model and the model with full covariates?" Extract the AIC values saved in `poisson.model` and `poisson.model.step` to answer this question.

```
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 4.1.1
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
model.poisson.step <- stepAIC(model.poisson,direction="backward")
```

```
## Start:  AIC=30164.98
## docvis ~ age + medicaid + private + female + income + educyr +
##     actlim + totchr
##
##            Df Deviance   AIC
## - income    1    18504 30165
## <none>           18503 30165
## - age       1    18515 30176
## - female    1    18520 30180
## - medicaid  1    18530 30191
## - private   1    18582 30242
## - actlim    1    18647 30307
## - educyr    1    18732 30392
## - totchr    1    21340 33000
##
## Step:  AIC=30164.55
## docvis ~ age + medicaid + private + female + educyr + actlim +
##     totchr
##
##            Df Deviance   AIC
## <none>           18504 30165
## - age       1    18516 30175
## - female    1    18523 30181
## - medicaid  1    18531 30190
## - private   1    18588 30246
## - actlim    1    18647 30305
## - educyr    1    18761 30419
## - totchr    1    21340 32998
```

```
task1b.does.backward.select.best.model <- "Is not the best possible model because backward selection do
task1b.AIC_diff <- model.poisson.step$aic-model.poisson$aic
```

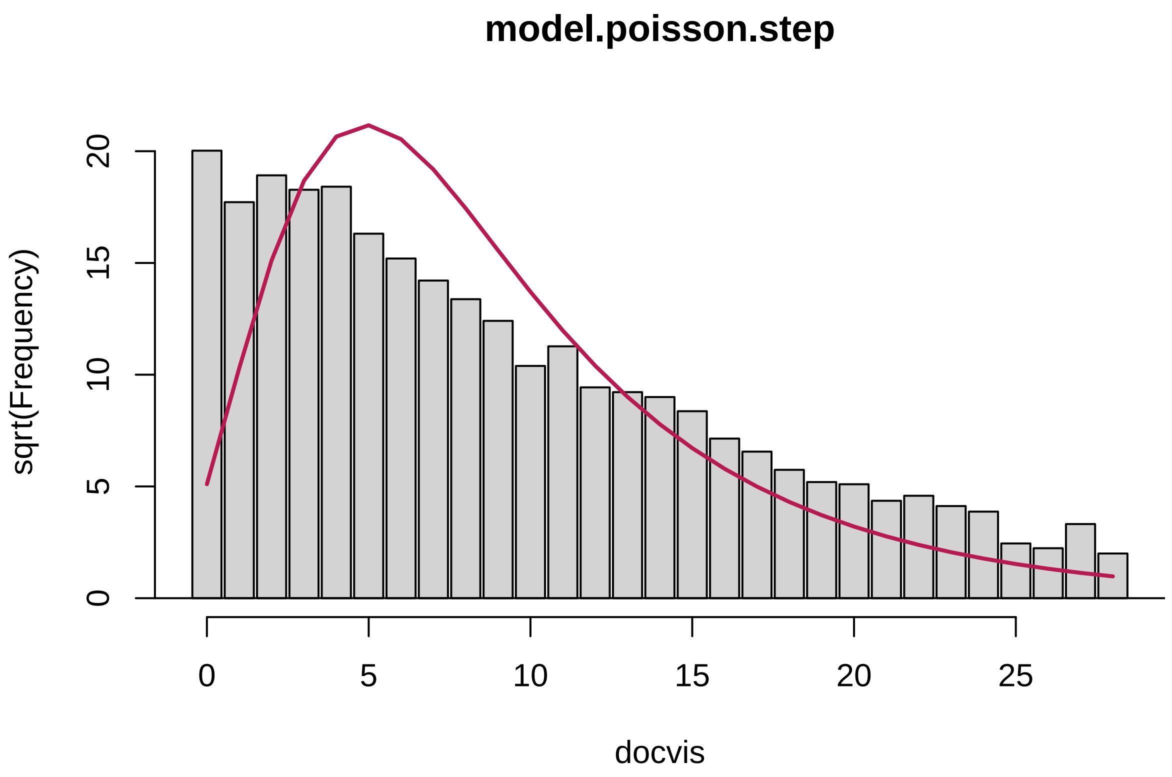
task 1 c)

We will now examine how well the selected model fits the data. In the lecture we saw in the lecture that it is crucial that the distribution of the model fits that of the data. We will now diagnose the model fit for our Poisson model. In order to do we will use what is known as rootograms. Read sections 2.1-3 in https://arxiv.org/pdf/1605.01311.pdf to understand how rootograms works. Install the package the `countreg` for using rootograms in R:

```
#install.packages("countreg", repos="http://R-Forge.R-project.org")
```

Plot the rootogram for the model selected, using which ever style you prefer. Describe the rootogram. Does the model selected in Task 1b fit the data well? Motivate your answer.

```
library(countreg)
rootogram(model.poisson.step, style ="standing")
```



```
task1c.describe.rootogram <- "The rootogram compares observed and expected values graphically. It
shows histogram-like rectangles/bars for the square-root of observations and a red curve of the square-
task1c.poisson.fit.rootogram <- "When fitting a Poisson model, we see that the curve representing
expected frequencies doesn't tracks the histogram representing observed frequencies"
```

Part 2): Negative Binomial regression

We saw in the lecture that Beta Binomial greatly improved the fit compared to the Binomal distribution due to that it allowed for overdispersion. A common extension of the Poisson model to account for overdispersion is the negative binomial distribution which have the following density:

$$p(y_i; \mu, \theta) = \frac{\Gamma(\theta + y_i)}{\Gamma(\theta) y!} \left(\frac{\mu}{\mu + \theta}\right)^y \left(\frac{\theta}{\mu + \theta}\right)^\theta$$

where $E[Y] = \mu$ and $V[Y] = \mu + \frac{1}{\theta}\mu^2$, here the parameter ϕ allows for overdispersion.

task 2a)

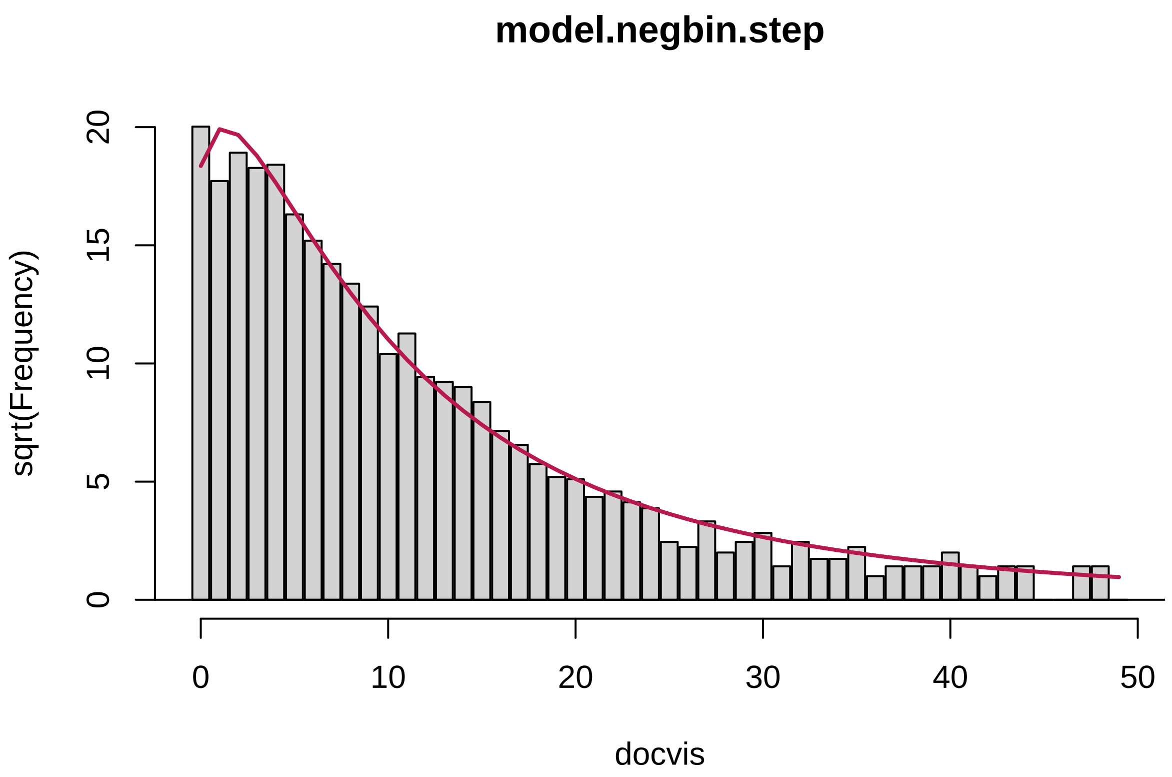
One can fit the negative binomial using `glm.nb` function. Perform the same step as done for the Poisson model but now for negative binomial, i.e. first fit the full model, do backward selection, and plot the rootogram.

Does the rootogram suggest that the negative binomial model fits the data better than the Poisson model? Motivate your answer. If you compare the final AIC for negative binomal regression and Poisson regression what does it say?

```r
model.negbin.full <- glm.nb(docvis~.,data=data)
model.negbin.step <- stepAIC(model.negbin.full,direction="backward")
```

```
## Start:  AIC=21218.2
## docvis ~ age + medicaid + private + female + income + educyr +
##     actlim + totchr
##
##            Df   AIC
## - female    1 21216
## - income    1 21217
## <none>        21218
## - age       1 21220
## - medicaid  1 21222
## - private   1 21236
## - actlim    1 21242
## - educyr    1 21256
## - totchr    1 21736
##
## Step:  AIC=21216.38
## docvis ~ age + medicaid + private + income + educyr + actlim +
##     totchr
##
##            Df   AIC
## - income    1 21216
## <none>        21216
## - age       1 21218
## - medicaid  1 21220
## - private   1 21235
## - actlim    1 21240
## - educyr    1 21255
## - totchr    1 21735
##
## Step:  AIC=21215.59
## docvis ~ age + medicaid + private + educyr + actlim + totchr
##
##            Df   AIC
## <none>        21216
## - age       1 21217
## - medicaid  1 21219
## - private   1 21236
## - actlim    1 21239
## - educyr    1 21261
## - totchr    1 21733
```

```
rootogram(model.negbin.step, style ="standing")
```



```
task2a.rootogram.model.fit.data <- "The negative binomial model fits the data better because we see less
task2a.AIC_compare <- "The AIC we obtained from the poisson was 30165,and the AIC we obtained in the neg
```

task 2b)

Negative binomial is an overdispersed version Poisson regression. This also affect the certainty of the coefficients in regression. Run `summary` one both the Poisson and Negative binomial for the selected model. What do you see for an effect in the certainty of the parameter?

```
summary(model.poisson.step)
```

```
##
## Call:
## glm(formula = docvis ~ age + medicaid + private + female + educyr +
##     actlim + totchr, family = "poisson", data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -6.0098  -1.8967  -0.6857   0.9037  24.1981
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.672186   0.080767   8.323  < 2e-16 ***
## age          0.003518   0.001016   3.464 0.000532 ***
## medicaid     0.099063   0.018985   5.218 1.81e-07 ***
## private      0.130765   0.014360   9.106  < 2e-16 ***
## female      -0.056327   0.013130  -4.290 1.79e-05 ***
## educyr       0.029800   0.001878  15.864  < 2e-16 ***
## actlim       0.174125   0.014548  11.969  < 2e-16 ***
## totchr       0.252478   0.004649  54.303  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 22873  on 3676  degrees of freedom
## Residual deviance: 18504  on 3669  degrees of freedom
## AIC: 30165
##
## Number of Fisher Scoring iterations: 5
```

```
summary(model.negbin.step)
```

```
##
## Call:
## glm.nb(formula = docvis ~ age + medicaid + private + educyr +
##     actlim + totchr, data = data, init.theta = 1.547683343, link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6245  -0.9647  -0.3079   0.3629   6.6141
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 0.484413   0.189534   2.556   0.0106 *
## age         0.004701   0.002403   1.956   0.0505 .
## medicaid    0.100550   0.045443   2.213   0.0269 *
## private     0.156184   0.033220   4.701 2.58e-06 ***
## educyr      0.029258   0.004322   6.770 1.29e-11 ***
## actlim      0.173577   0.035139   4.940 7.82e-07 ***
## totchr      0.280424   0.011751  23.864  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(1.5477) family taken to be 1)
```

```
##
##     Null deviance: 5033.9  on 3676  degrees of freedom
## Residual deviance: 4195.4  on 3670  degrees of freedom
## AIC: 21218
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  1.5477
##           Std. Err.:  0.0473
##
##  2 x log-likelihood:  -21201.5930
```

```
task2b.coeff_res <- "We can see that in the poisson model all the coeffients are significant, whereas i
```

Part 3: Zero inflated regression

Often dealing with count data the zero value is of special importance and might not be fitting the standard models. There exists two standard methods for dealing with this namely zero inflated model and hurdle model.

We here focus on the zero inflated model. Suppose $p(y; \mu, \theta)$ is the density of the negative binomial model the zero inflated density is given

$$p^{zero}(y; \theta, \mu, \pi) = \pi^{zero}\delta_0(y) + (1 - \pi^{zero})p(y; \mu, \theta)$$

here $\delta_0(y)$ takes value one at if $y = 0$ zero else, and π^{zero} is the probability of observing the zero class.

task 3a)

What is the probability of observing zero in the model above? Note we are not asking for a numerical value but a mathematical formula using π^{zero} and $p(y; \mu, \theta)$.

```
task3a.what.is.zero.prob <- "p^{zero}(0;theta,mu,pi )=pi^{zero} + (1-pi^{zero}) p(0;miu,theta)"
```

task 3b)

Have another look at the rootogram of the negative binomial distribution which you created in Task 2a. Does the corresponding negative binomial model correctly predict the number of zero counts? Motivate your answer.

```
task3b.negbin.docvis.zero <- "from the rootgram we can see that the model underpredicts the number of z
```

task 3c)

Often one only use a single π^{zero}, but one can also use a logit model for π^{zero} so that

$$\pi^{zero}(x^T\beta) = logit^{-1}(x^T\beta)$$

Now you are supposed to fit these models in R using the `zeroinfl` function in the package `countreg` function. Read the help instruction for `zeroinfl` and setup a full model using all covariates both for the negative binomial part and the zero inflation part.

```r
model.zero.negbin <- zeroinfl(docvis~.|.,dist ="negbin",data=data)
```

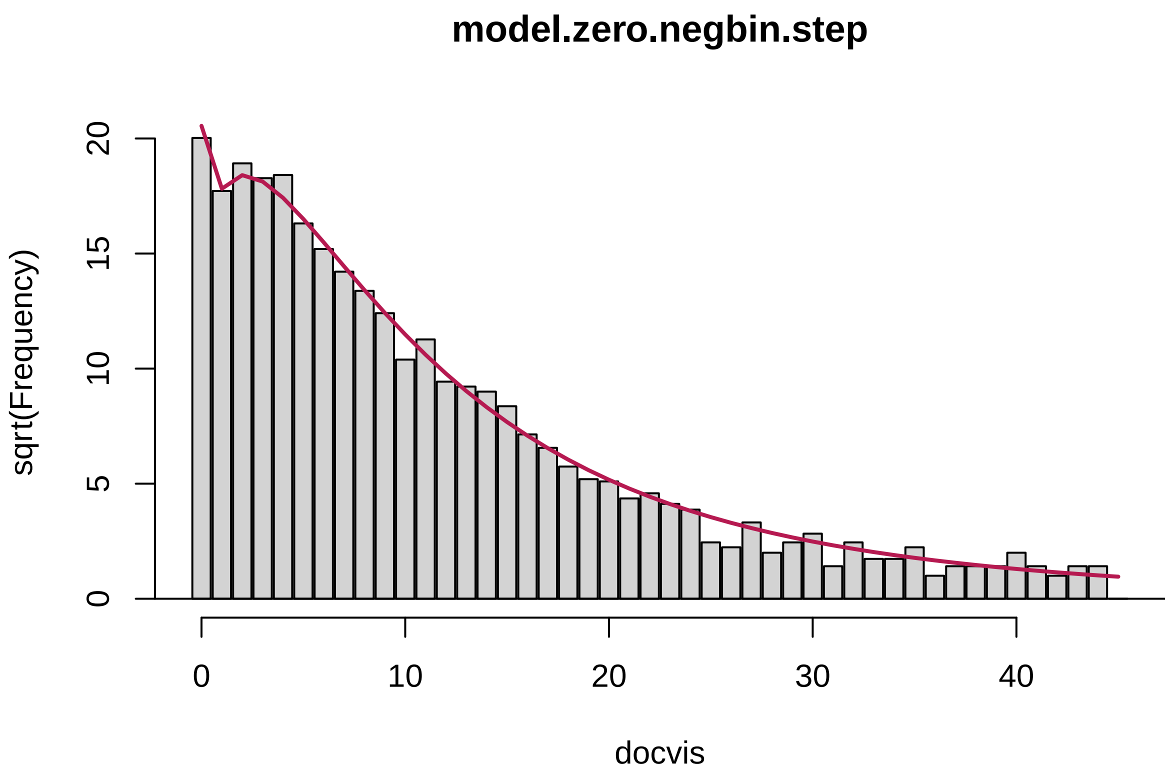
task 3d)

Now do backward selections again and create the rootogram. Does the model fit the data better? What about the AIC of the final model, which model is best according to *AIC* and what of all things starting from the full Poisson regression model gave the largest improvment in *AIC*?

```r
model.zero.negbin.step <- stepAIC(model.zero.negbin,direction="backward")
```

```
## Start:  AIC=21034
## docvis ~ age + medicaid + private + female + income + educyr +
##     actlim + totchr
##
##            Df   AIC
## - income    2 21031
## <none>        21034
## - age       2 21035
## - medicaid  2 21037
## - female    2 21043
## - actlim    2 21059
## - private   2 21059
## - educyr    2 21070
## - totchr    2 21673
##
## Step:  AIC=21031.24
## docvis ~ age + medicaid + private + female + educyr + actlim +
##     totchr
##
##            Df   AIC
## <none>        21031
## - age       2 21032
## - medicaid  2 21034
## - female    2 21041
## - actlim    2 21056
## - private   2 21058
## - educyr    2 21074
## - totchr    2 21669
```

```r
task3d.rootfit <- rootogram(model.zero.negbin.step, style ="standing")
```



```
task3d.AIC      <- "According to the rootgram, we can see that zero negative binomial fit the model bett
```

Part 4: multionmial regression and grouped lasso

We will now perform grouped lasso regularization on a multinomial regression. The data we are studying is drug consumption data. Our response variable is usage of drugs (Cocaine , Crack, Ecstasy, and Heroin) and we have three possible responses, "never used", "used more than a year ago", and "used within a year". As explanatory variables we have personality test data, demographic data, and consumption of chocolate, alcohol, and nicotine.

We start by loading the data and create the `X` matrix for the full model excluding the intercept (which will be fitted by `glmnet`). As the `y` variable should be the column `drugs.usage`.

```
drug.data <- readRDS('drug_train.RDS')
X <- model.matrix(drugs.usage ~ -1+. ,data=drug.data)
y <- drug.data$drugs.usage
```

task 4a)

Read the vingettes on how to train a multionomial model with group lasso penalty using `glmnet`. Repeat the procedure for the `drug.data`. Which variable seems to increase the probability to have used drugs within a year the most if using the lambda selected by the one standard deviation rule? Hint you can extract the coefficients using `coef`.

```r
set.seed(12345)

library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.1.1
```

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

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

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

```r
cvfit <- cv.glmnet(X,y,family="multinomial",type.multinomial = "grouped", intercept=FALSE)
coef<-coef(cvfit,s="lambda.1se")$"within a year"


task4a.which.effects.most <- "The variable that seems to increase the probabilty to have used consumpti
```

task 4b)

We will now evaluate the performance of the model on some hold out data. Use `predict` to generate predictions on the new data set, with coefficients taken from the `lambda.1se` option. There exists many different options for the argument `type` for the `predict`. Explain what different output you get for the three different types: `type='class'`, `type='link'`, and `type='response'`. What is the connection between the prediction generated in `type='class'` and `type='response'`?

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.1.1
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
drug.data.test <- readRDS('drug_test.RDS')
X.test <- model.matrix(drugs.usage ~ -1+. ,data=drug.data.test)
y.test <- drug.data.test$drugs.usage
```

```
y.pred.response <- predict(cvfit,newx =X.test,s="lambda.1se",type="response")
y.pred.class <- predict(cvfit,newx =X.test,s="lambda.1se",type="class")
y.pred.link <- predict(cvfit,newx =X.test,s="lambda.1se",type="link")


task4b.predict.response.is <- "gives the fitted probabilities"
task4b.predict.class.is    <- "produces the class label corresponding to the maximum probability"
task4b.predict.link.is     <- "gives the linear predictors"
task4b.predict.connection.response.and.class    <- "We are fitting our model with multinomial regressio

If we look at the results of y.pred.response and y.pred.class we can confirm that the response give us
```

task 4c)

Again `predict` to generate predictions on the new data set, with coefficients taken from the `lambda.1se` option. We examine the result using the confusion matrix: What is the accuracy of the model? What would be the best accuracy you could get by always just a single class all the time?

```
library(caret)
drug.data.test <- readRDS('drug_test.RDS')
X.test <- model.matrix(drugs.usage ~ -1+. ,data=drug.data.test)
y.test <- drug.data.test$drugs.usage
y.pred <- predict(cvfit,newx =X.test,s="lambda.1se",type="class")
ConfMatrix <- caret::confusionMatrix(data=factor(y.pred), reference=y.test)
```

```
## Warning in confusionMatrix.default(data = factor(y.pred), reference = y.test):
## Levels are not in the same order for reference and data. Refactoring data to
## match.
```

```
task4c.accuracy <- ConfMatrix$overall["Accuracy"]
tast4c.accuracy.single.class <- ""

#3 classes, the weighted sum to get the accuracy
```