Regression Models: Assignment 2

Daniel Alonso

January 13th, 2020

Installing libraries used

```
packages = c("dplyr","MuMIn","MASS","leaps","glmnet",
             "car","stringr","ResourceSelection",
             "boot","statmod","Epi", "Metrics",
             "caret", "ggplot2", "multcomp",
             "combinat", "pscl", "lmtest")
for (package in packages) {
    install.packages(package)
}
```

Importing libraries

```
library(dplyr)
library(MuMIn)
library(MASS)
library(leaps)
library(glmnet)
library(car)
library(stringr)
library(ResourceSelection)
library(boot)
library(statmod)
library(Epi)
library(Metrics)
library(caret)
library(ggplot2)
library(multcomp)
library(combinat)
library(pscl)
library(lmtest)
```

Exercise 1

Model and parameter interpretation

Y = Binary variable representing whether the customer will buy a car or not *income* = annual family income

Therefore we model the response as:

$\eta = \beta_0 + \beta_1 X + \epsilon$

And our values will be:

$\eta = e^{-1.98079 + X0.04342 + \epsilon}$

Where X is the annual family income.

The odds increase by $e^{\beta_1} = 1.044376$ if the predictor is increased by one unit.

95%-CI for the probability that a family with annual income of 60 thousand dollars will purchase a new car next year.

We calculate the asymptotic $(1-\alpha)\%$ confidence interval:

$\hat{\beta}_j \pm z_{\frac{\alpha}{2}} S.E.(\hat{\beta}_j)$

With our values we get:

```r
# defining a p function for prob
p = function(eta) (exp(eta)/(exp(eta)+1))

# CI-Z
z_95 <- qnorm(0.975)

# CI calculation
p(-1.98079 + z_95*0.85720 + 0.04342*60 + z_95*0.02011)
#> [1] 0.9124486
p(-1.98079 - z_95*0.85720 + 0.04342*60 - z_95*0.02011)
#> [1] 0.2506618
```

$0.2506618 \leq p_{60k} \leq 0.9124486$

Grouping into 6 levels of income, what test is used and what are the DF of the test statistic

The appropriate test for this would be the Hosmer-Lemeshow test with $G = 6$ (corresponding to 6 groups).

The DF of the test statistic for a Hosmer-Lemeshow test is $DF = G - 2$, therefore, $DF = 4$

Exercise 2

Importing and manipulating the dataset

```
cols <- c("age","lwt","race","smoke")
birthwt <- MASS::birthwt %>% dplyr::select(c("low",cols))
```

For race we should use a dummy variable per race:

```
birthwt$white <- ifelse(birthwt$race == 1, 1, 0)
birthwt$black <- ifelse(birthwt$race == 2, 1, 0)
birthwt$other <- ifelse(birthwt$race == 3, 1, 0)
cols <- c("age", "lwt", "smoke", "white", "black", "other")
birthwt <- birthwt %>% dplyr::select(c("low",cols))
```

Model fitting and selection

```
FM <- glm(low ~ ., data=birthwt, family=binomial)
staic <- stepAIC(FM, list(upper=~age*lwt*smoke*white*black*other, lower= ~1))
#> Start:  AIC=226.58
#> low ~ age + lwt + smoke + white + black + other
#>
#>
#> Step:  AIC=226.58
#> low ~ age + lwt + smoke + white + black
#>
#>               Df Deviance    AIC
#> - black        1   214.88 224.88
#> - age          1   215.01 225.01
#> <none>             214.58 226.58
#> + smoke:white  1   213.16 227.16
#> + lwt:smoke    1   213.66 227.66
#> + age:black    1   214.05 228.05
#> + lwt:black    1   214.05 228.05
#> + age:smoke    1   214.25 228.25
#> + lwt:white    1   214.45 228.45
#> + smoke:black  1   214.49 228.49
#> + age:lwt      1   214.55 228.55
#> + age:white    1   214.56 228.56
#> - lwt          1   218.86 228.86
#> - white        1   219.89 229.89
#> - smoke        1   222.66 232.66
#>
#> Step:  AIC=224.88
#> low ~ age + lwt + smoke + white
#>
#>               Df Deviance    AIC
#> - age          1   215.38 223.38
#> <none>             214.88 224.88
#> + smoke:white  1   213.67 225.67
#> + lwt:smoke    1   214.12 226.12
#> + age:smoke    1   214.47 226.47
#> + black        1   214.58 226.58
#> + other        1   214.58 226.58
#> + lwt:white    1   214.83 226.83
```

```
#> + age:lwt      1   214.83 226.83
#> - lwt          1   218.87 226.87
#> + age:white    1   214.87 226.87
#> - white        1   222.88 230.88
#> - smoke        1   223.85 231.85
#>
#> Step:  AIC=223.38
#> low ~ lwt + smoke + white
#>
#>               Df Deviance    AIC
#> <none>            215.38 223.38
#> + smoke:white  1   213.94 223.94
#> + lwt:smoke    1   214.62 224.62
#> + age          1   214.88 224.88
#> + black        1   215.01 225.01
#> + other        1   215.01 225.01
#> + lwt:white    1   215.31 225.31
#> - lwt          1   219.98 225.98
#> - white        1   224.34 230.34
#> - smoke        1   224.65 230.65
```

Using stepAIC we can see all the combinations classified by AIC. The model with the lowest AIC is the model that uses *lwt*, *smoke* and *white* and drops the *age*, *black* and *other* variables.

We can see the interactions between the variable selected and age are not particularly significant and don't seem to affect the model enough to consider them, in fact, the AIC is improved when these are not present.

We can see that in general, dropping the *age* variable yields a better result:

```
staic$anova
#> Stepwise Model Path
#> Analysis of Deviance Table
#>
#> Initial Model:
#> low ~ age + lwt + smoke + white + black + other
#>
#> Final Model:
#> low ~ lwt + smoke + white
#>
#>
#>      Step Df  Deviance Resid. Df Resid. Dev      AIC
#> 1                           183   214.5772 226.5772
#> 2 - other  0 0.0000000       183   214.5772 226.5772
#> 3 - black  1 0.2991850       184   214.8764 224.8764
#> 4   - age  1 0.5068236       185   215.3832 223.3832
```

```
options(na.action=na.fail)
MuMIn::dredge(FM)
```

Using dredge also tells us the same as stepAIC, where the best model is the one at the top (as they are ranked by AIC already).

```
anova(FM, staic, test="Chisq")$"Pr(>Chi)"[2]
#> [1] 0.6683092
```

Performing a likelihood ratio test yields a good, high p-val of 0.668 so we pick the reduced model.

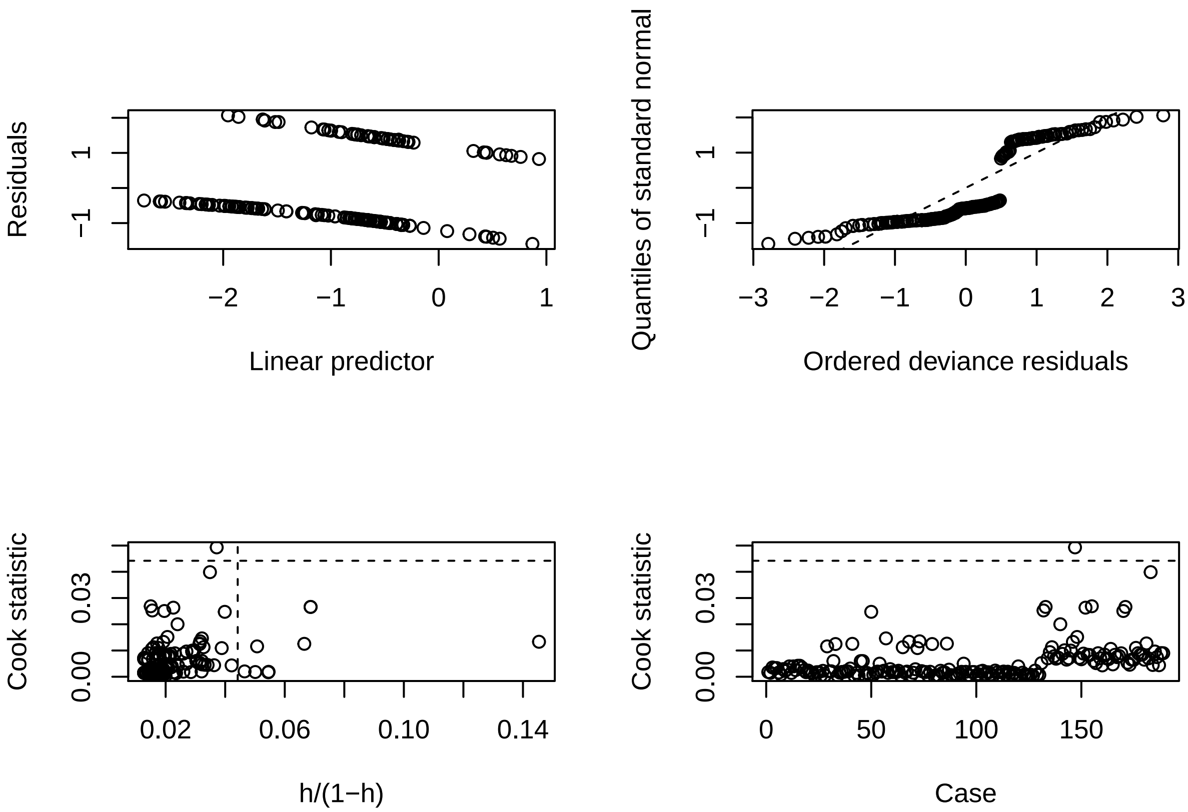
Hosmer-Lemeshow test

```
hoslem.test(birthwt$low, predict(staic, type="response"))
#>
#>  Hosmer and Lemeshow goodness of fit (GOF) test
#>
#> data:  birthwt$low, predict(staic, type = "response")
#> X-squared = 9.0869, df = 8, p-value = 0.335
```

We have a large p-value of 0.335 which indicates that our goodness of fit is most likely okay.

Residual plots and model assumptions

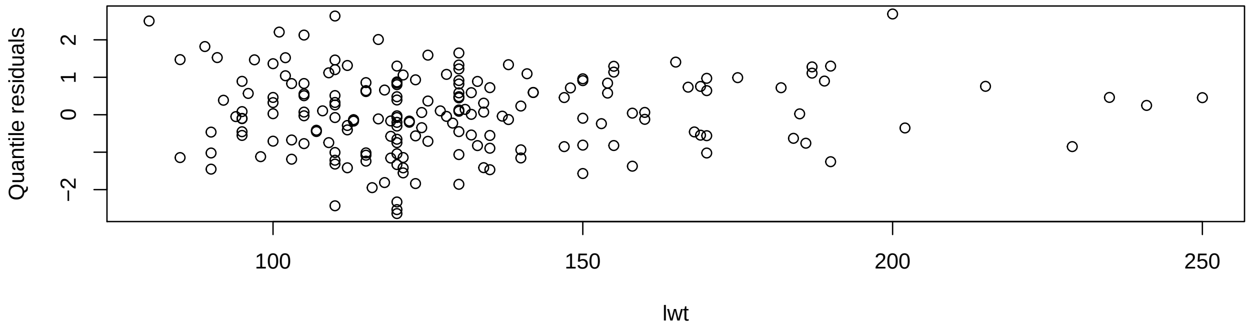
```
glm.diag.plots(staic)
```

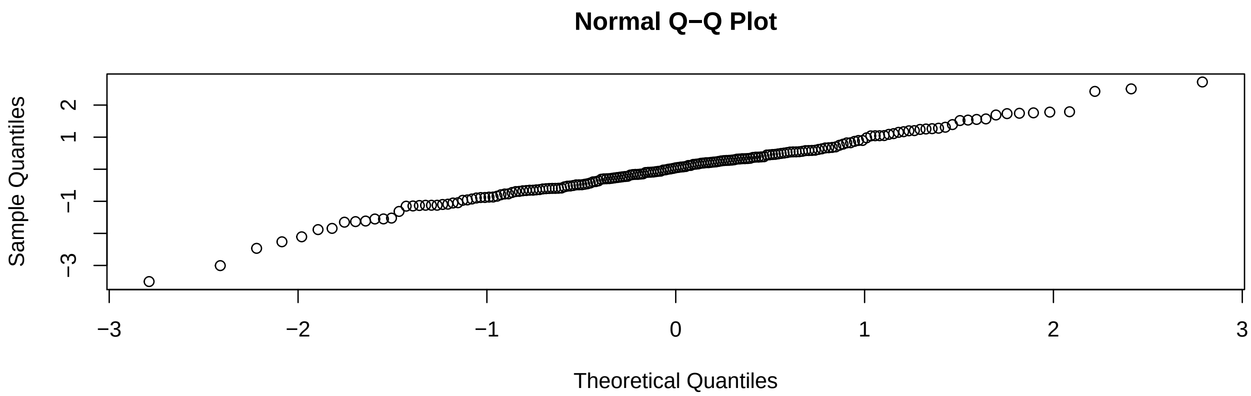


As we have 2 sets of points for the Residuals vs Linear predictor and the Quantiles of standard normal vs Ordered deviance residuals, we can't properly interpret these.

For the cook's distance we can see there is a few slightly high leverage points . However, it is not significant as other than this there's no points present in the top right quadrant of the plot.

```r
par(mfrow=c(2,1))
plot(birthwt$lwt, qres.binom(staic), xlab="lwt", ylab="Quantile residuals")
qqnorm(qres.binom(staic))
```





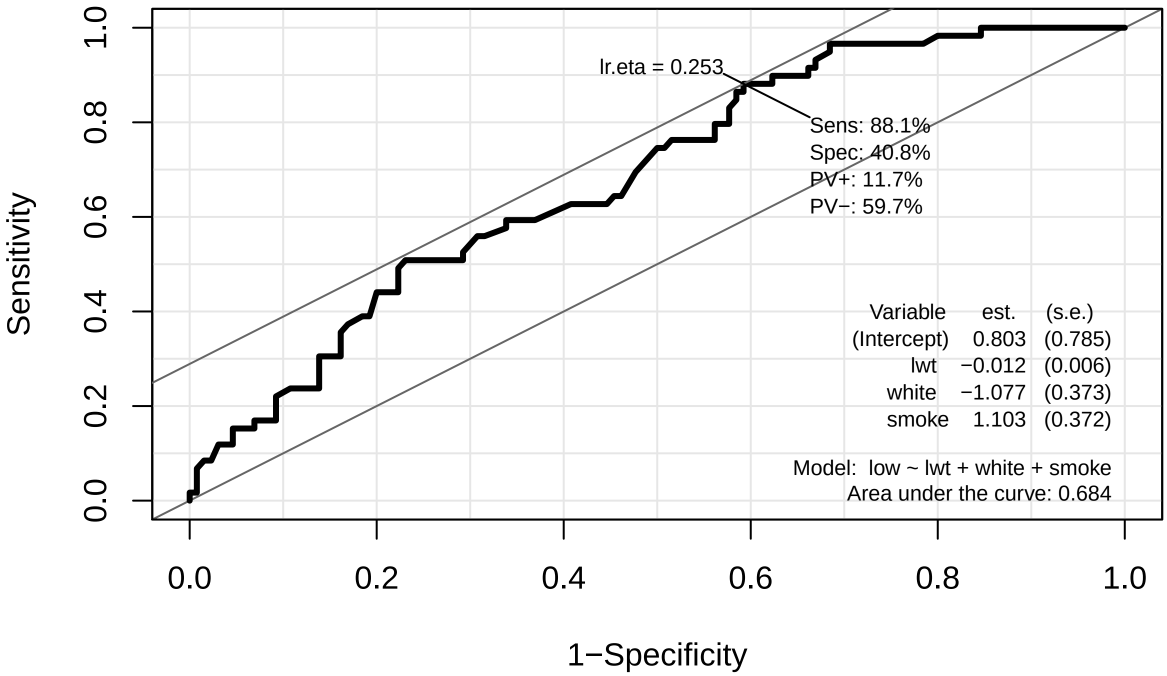
We decide not to plot the *race* or *smoke* variables given that, even though they're in the model, they're categorical variables.

For our *lwt* residual plot, everything seems to be okay, we see that. In the normal QQ plot we see that the values decently fit a normal distribution. This fits the normality assumption.

Our only continuous variable in the model (*lwt*) seems to have constant variance, therefore our model is homocedastic.

Total error rate of the model

```
Epi::ROC(form=low~lwt+white+smoke , data=birthwt, plot="ROC", lw=3, cex=1.5)
```



The model has an AUC = 0.684 which corresponds to a decent but not particularly good model, however, using this measure we can assert that the model does have predictive capability.

We can see our cutoff point is also 0.253.

Our model has a very high sensitivity, however, a very low specificity, therefore it also has a very high false negative rate. We could comfortably assert that this is the achilles heel of our model, as it still has a high accuracy for positives but a very low accuracy for negatives.

It would also be appropriate to look at the MAE and MSE for our model.

```
Metrics::mae(birthwt$low, predict(staic, type="response"))
#> [1] 0.3893964
Metrics::mse(birthwt$low, predict(staic, type="response"))
#> [1] 0.1956847
```

We can see that they're both relatively low even though our model doesn't perform amazingly.

Mothers having babies with low birth weight vs normal birth weight

First we will look at variable importance:

```
caret::varImp(staic)
#>        Overall
#> lwt   2.029237
#> smoke 2.962211
#> white 2.888260
```

We can see that the most important variable of the model is the *smoke* variable, followed by *white* and then *lwt*.

We select a subset of the original dataset which uses the model prediction and we also select a subset of the original dataset using the real classification. Both for normal birth weight babies and low birth weight babies, in order to assess which elements are characteristic of each group.

```r
# model's prediction
pred <- predict(staic, type="response")
normal_birth_weight <- birthwt[pred<0.253,]
low_birth_weight <- birthwt[pred>0.253,]

# reality
real_lbw <- birthwt %>% dplyr::filter(low == 1)
real_nbw <- birthwt %>% dplyr::filter(low == 0)
```

Smoke prevalence

```r
table(normal_birth_weight$smoke)
#>
#>  0  1
#> 53  7
```

We can see that as the model considers the variable smoke particularly important for prediction, it seems to very strongly influence its prediction of normal birth weight, therefore very effectively predicting those with normal birth weight. And as we clearly know, smoking is a high risk factor for birth issues like this. However, we can also notice that non-smokers tend to give birth to normal weight babies.

```r
table(low_birth_weight$smoke)
#>
#>  0  1
#> 62 67
```

However, when comparing its prediction of low birth weight it falls short, as not all low birth weight babies come from a mother that smokes. The model fails about 60% of the time.

In contrast to the reality:

```r
table(real_lbw$smoke)
#>
#>  0  1
#> 29 30
```

For low weight babies there's about a 50% chance that the mother is a smoker

```r
table(real_nbw$smoke)
#>
#>  0  1
#> 86 44
```

While it is significantly more probable that the mother is not a smoker when the baby has a normal birth weight. We see that the amount of non-smoker mothers represent about 66% of the normal birth weight subset.

Age

```
#> [1] "low birth weight: 22.3488372093023"
#> [1] "normal birth weight: 25.15"
```

We can see that according to the model, the median age of mothers giving birth to normal weight babies is ~25.15 years old, while the ones with low birth weight babies are ~22.35 years old.

In contrast to the reality though:

```
#> [1] "low birth weight: 22.3050847457627"
#> [1] "normal birth weight: 23.6615384615385"
```

There doesn't seem to be a significant age difference (~1 year).

Mother's weight

```r
print(stringr::str_interp('low birth weight: ${mean(low_birth_weight$lwt)}'))
#> [1] "low birth weight: 119.015503875969"
print(stringr::str_interp('normal birth weight: ${mean(normal_birth_weight$lwt)}'))
#> [1] "normal birth weight: 153.033333333333"
```

The mother's weight shows significant difference for the prediction, where normal birth weight mom's weight (on average) about 34 pounds more.

```r
print(stringr::str_interp('low birth weight: ${mean(real_lbw$lwt)}'))
#> [1] "low birth weight: 122.135593220339"
print(stringr::str_interp('normal birth weight: ${mean(real_nbw$lwt)}'))
#> [1] "normal birth weight: 133.3"
```

However, in reality, the difference is ~11 pounds on average for our dataset.

Mother's race (binary if white)

```r
table(low_birth_weight$white)
#>
#>  0  1
#> 83 46
```

The model is significantly biased towards the white race group where most low birth weight babies come from non-white mothers (about 2x more likely).

```r
table(normal_birth_weight$white)
#>
#>  0  1
#> 10 50
```

We also see that race group 1 has the highest representation among those mothers with normal birth weight babies.

In contrast to the reality:

```r
table(real_lbw$white)
#>
#>  0  1
#> 36 23
```

We can see that in the real dataset, race doesn't quite seem to play the role that the model portrays it to have in whether a baby has low birth weight or not.

```r
table(real_lbw$white)
#> 
#>  0  1 
#> 36 23
table(real_nbw$white)
#> 
#>  0  1 
#> 57 73
```

There's a clear overrepresentation of race group 1 in the normal birth weight subset.

What characteristic had the highest impact?

Following the model's result, we can definitely say that whether the mother was a smoker or not had the highest influence in its prediction, followed by the race, where there was a huge overrepresentation of group 3 in the low birth weight group.

Exercise 3

```
health <- read.table('../data/health.txt', header=TRUE)
cols <- c("g02","sex","weight")
health <- health %>% dplyr::select(g02,sex,weight)
```

```
mean(health[health$sex==1,]$weight)
#> [1] 77.89171
mean(health[health$sex==2,]$weight)
#> [1] 61.18423
```

We will make the assumption that for the *sex* column *1* = *males* and *2* = *females*, as average weight for males is (generally) higher for pretty much every country.

We will subtract 1 from the *sex* column to make it a binary variable with only 1s and 0s.

```
health$sex <- health$sex - 1
```

```
fm <- glm(g02 ~ sex+weight+sex:weight, data=health, family=binomial)
model <- glm(g02 ~ sex+weight, data=health, family=binomial)
anova(fm, model, test="Chisq")
#> Analysis of Deviance Table
#>
#> Model 1: g02 ~ sex + weight + sex:weight
#> Model 2: g02 ~ sex + weight
#>   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
#> 1      7353     7013.5
#> 2      7354     7042.3 -1  -28.858 7.79e-08 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The interaction between sex and weight is significant, therefore we will include it in the model.

Interpreting the coefficients in terms of the OR

$\eta = log(\frac{p}{1-p})$

$\eta = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k$

In terms of the odds:

$Odds = e^{2.56152+1.236831X_1-0.01171X_2-0.028984X_1X_2}$

Where X_1 represents *sex*, X_2 represents *weight* and X_1X_2 represents *sex:weight*.

For the odds ratio, we should highlight the differences between males and females.

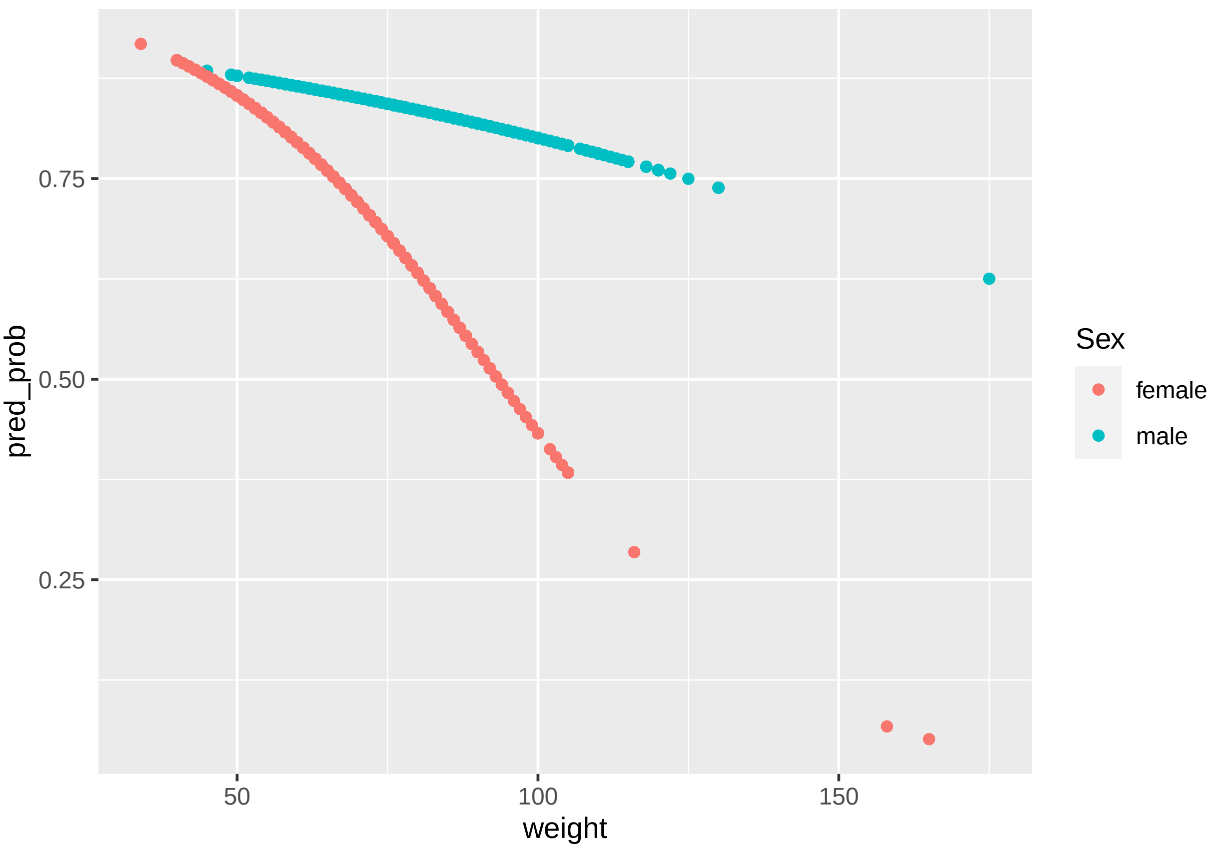
Therefore:

$O_R = \frac{e^{2.56152-0.01171X_2}}{e^{2.56152+1.236831X_1-0.01171X_2-0.028984X_1X_2}}$

Where the numerator of the fraction corresponds to the odds for **males** and the denominator corresponds to the odds for **females**.

Plotting predicted probabilities for males and females

```r
health$pred_prob <- predict(fm, type='response')
Sex <- ifelse(health$sex == 1, "female", "male")
ggplot(data=health, aes(color=Sex)) + geom_point(aes(x=weight, y=pred_prob))
```



We can see a trend here, men are significantly more likely to consider themselves healthy. The model is also telling us that weight negatively affects the probability to feel healthy in a significant way, however, much more significantly on females than males.

Both men and women seem to consider weight an important factor in their health perception. The lower usually tends to mean the better, however, it's clear that being underweight isn't a healthy trait, but some people might think otherwise.

Relative risk and odds ratio of self-perceived good health per sex for a 75kg person

We predict for both males and females by specifying the *sex=0* (for males) and *sex=1* (for females) and *weight=75* for both sexes.

```
males <- predict(fm, newdata=data.frame(sex=0,weight=75), type="response")
females <- predict(fm, newdata=data.frame(sex=1,weight=75), type="response")
```

We calculate the relative risk:

```
females/males
#>         1
#> 0.8043606
```

And the odds Ratio:

```
(females/(1-females))/(males/(1-males))
#>         1
#> 0.3918128
```

We can say that females are *0.3918128* times less likely to have self-perceived good health than males.

Estimated expected probability of self-perceived good health of females of 70kg and 110kg with CI

Females of 70kg

```
predict(fm, newdata=data.frame(sex=1,weight=70), type="response")
#>         1
#> 0.7210275
```

The expected probability of self-perceived good health for 70kg females is of ~0.721.

Females of 110kg

```
predict(fm, newdata=data.frame(sex=1,weight=110), type="response")
#>         1
#> 0.3366375
```

The expected probability of self-perceived good health for 110kg females is of ~0.336.

95%-CI for the prob. of self-perceived good health for a 70kg female

```
w1 <- predict(fm, newdata=data.frame(sex=1,weight=70), type="link", se.fit=TRUE)

p(w1$fit - qnorm(0.975)*w1$se.fit)
#>         1
#> 0.7015783
p(w1$fit + qnorm(0.975)*w1$se.fit)
#>         1
#> 0.7396794
```

The confidence interval for the probability of self-perceived good health for a 70kg female is:

$0.7015783 \leq \beta_{70kg} \leq 0.7396794$

95%-CI for the prob. of self-perceived good health for a 110kg female

```
w2 <- predict(fm, newdata=data.frame(sex=1,weight=110), type="link", se.fit=TRUE)

p(w2$fit - qnorm(0.975)*w2$se.fit)
#>         1
#> 0.2609291
p(w2$fit + qnorm(0.975)*w2$se.fit)
#>         1
#> 0.4217766
```

The confidence interval for the probability of self-perceived good health for a 70kg female is:

$0.2609291 \leq \beta_{110kg} \leq 0.4217766$

Exercise 4

Importing and manipulating the data:

We exclude *g01* as it seems to interfere with the predictions (probably because the target variable *g02* seems to be based on *g01*).

Also, during testing, year didnt seem to influence the model very much.

```
health <- read.table('../data/health.txt', header=TRUE)
cols <- names(health)[names(health) != "g01"]
health <- health %>% dplyr::select(cols)

# taking one from sex to have it as 0, 1
health$sex <- health$sex - 1
```

We create a model which includes all the variables and their interactions:

```
fm <- glm(g02~sex*weight*height*con_tab*educa*drink*age*year*imc, data=health, family=binomial)
```

The approach of this algorithm is at follows:

1 - We create a vector with the column names excluding g02

2 - We create a list called *best_models* for all the variables used in each model, and *everything* for the AIC, BIC and LRT P-value vs the full model for each model in *best_models*.

Each column of *best_models* corresponds with a list of variables used in each model, each row of *everything* has the scores for those corresponding columns of *best_models*.

3 - We loop from 2 to 9 (2-var models up to 9-var models). We add to *best_models* all the possible 2 to 9 variable unique combinations that we have available with all our predictors.

4 - we create an empty vector of AICs, BICs and LRT p-vals per amount of variables, so there will be one of these each for 2, 3, 4… etc amount of variables.

5 - We create a string with the model formulas, all of the form "g02~(var1+var2+var3)ˆ2" (for example, for 3-var models). The idea is to include all the variables in parenthesis and their 2-way interactions.

6 - We calculate the AIC, BIC and LRT p-val of this model vs de full model (which also includes 2 to 9-way interactions). The way the best model is calculated is using stepAIC for each variable combination, this way we optimize for AIC and only include the best possible model with such variable combination.

7 - We add these results to each vector (AICs, BICs and LRT p-vals) and we add them to a dataframe corresponding to its i value in the loop (i represents the amount of variables), so *everything[[2]]* contains a dataframe with all the AICs, BICs and LRT p-vals for the 2-variable models with their 2-way interactions (after optimizing for AIC).

8 - We save both the variable lists used in each model and the results for all the tested models in a CSV document. There will be 1 CSV document per amount of variables (1 for 2-var models, 1 for 3-var models, etc.)

```
# exclude target
cols <- names(health)[names(health) != "g02"]
```

```
# This takes really long to run!
best_models <- list()
everything <- list()
for (i in 2:length(cols)) {
    best_models[[i]] <- combinat::combn(cols,i)
    aics_l <- c()
    bics_l <- c()
```

```
    lrts_l <- c()
    for (k in 1:(length(best_models[[i]])/i)) {
        if (i == length(best_models[[i]])) {
            mods <- paste(best_models[[i]],collapse="+")
        } else {
            mods <- paste(best_models[[i]][,k],collapse="+")
        }
        curr_model <- stringr::str_interp('g02~(${mods})^2')
        md <- glm(curr_model, data=health, family=binomial)
        aic_optimized <- stepAIC(md)
        aics_l <- c(aics_l, AIC(aic_optimized))
        bics_l <- c(bics_l, BIC(aic_optimized))
        test <- anova(fm, aic_optimized, test="Chisq")
        lrts_l <- c(lrts_l, test$"Pr(>Chi)"[2])
    }
    mod <- 1:(length(best_models[[i]])/i)
    everything[[i]] <- data.frame(mod=mod,aics=aics_l,bics=bics_l,lrts=lrts_l)
}
```

After performing this set of calculations we find the best possible model among all the ones we have obtained.

We then sort the dataframe with the AICs, BICs, and LRT p-vals in order of priority, our most important measure will be AIC, followed by BIC and then the LRT p-val. We basically only need the LRT to be passed and a decent AIC and BIC, but the lower the better.

We will normalize the AICs, BICs and LRT p-vals using min-max scaling with values between 1 and 2 and calculate a weighted score for each measure, 4/7 of the score will correspond to AIC, and 3/7 to BIC while for the LRT p-val we will require the condition that it's below 0.05 (our chosen significance) to ensure that it does pass the test. The minimum score will correspond to the best model.

First of all, we filter and keep only models which have an LRT p-val below 0.05:

```
filtered <- list()
for (i in 2:length(everything)) {
    filtered[[i]] <- everything[[i]] %>% filter(lrts < 0.05)
}
```

We scale the AICs and BICs and calculate the scores.

```
min_max_scale <- function(vct) {return((vct - min(vct))/(max(vct) - min(vct)) + 1)}
for (i in 2:length(everything)) {
    everything[[i]]$score <- min_max_scale(everything[[i]]$aics)*(4/7) +
                             min_max_scale(everything[[i]]$bics)*(3/7)
}
```

We sort the dataframes by score, lowest goes at the top:

```
sorted <- list()
three_best <-
for (i in 2:length(everything)) {
    sorted[[i]] <- everything[[i]][order(everything[[i]]$score),]
}
```

We check the top scores for each amount of variables:

Table 1: best models

mod	aic	bic	lrt_pval	score
33	6510.211	6530.921	0.0000000	1.000000
83	6453.976	6488.493	0.0000169	1.000000
126	6411.525	6459.849	0.0016461	1.000739
70	6387.883	6443.110	0.0124628	1.000000
50	6387.883	6443.110	0.0124628	1.009738
27	6387.613	6456.647	0.0145861	1.018765
5	6373.021	6483.476	0.0561360	1.000000
1	6372.676	6503.841	0.0677725	NaN

From my perspective, the model I would pick is the model 70 with 5 variables, which has an AIC of *6387.883*, a BIC of *6443.11* and passes the LRT with a p-value of *0.0124628* vs our significance of *0.05*.

The chosen model is the following:

```r
predictors <- paste(best_models[[5]]$V70,collapse="+")
formula <- str_interp("g02~(${predictors})^2")
mod_raw <- glm(formula, data=health, family=binomial)
mod <- stepAIC(mod_raw)
```

```r
formula(mod)
#> g02 ~ sex + educa + imc + drink + age + sex:age + drink:age
#> <environment: 0x559cbfc0de80>
```

Which results from using *stepAIC* on the model:

```r
formula(mod_raw)
#> g02 ~ (sex + educa + imc + drink + age)^2
#> NULL
```

Exercise 5

Importing and manipulating the data:

```
crime <- read.table('../data/Campus_Crime.txt', header=TRUE)
crime$Type <- as.factor(crime$Type)
crime$Region <- as.factor(crime$Region)
```

We first create both a model (using the variables *Region* and *Type*) with and without the interactions:

```
fm <- glm(Property~Region+Type+Region:Type, data=crime, family=poisson, offset=log(Enrollment))
model <- glm(Property~Region+Type, data=crime, family=poisson, offset=log(Enrollment))
```

We test for the significance of the interactions between the different variables:

```
anova(fm, model, test="Chisq")
#> Analysis of Deviance Table
#>
#> Model 1: Property ~ Region + Type + Region:Type
#> Model 2: Property ~ Region + Type
#>   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
#> 1        69     3735.4
#> 2        74     4585.5 -5  -850.01 < 2.2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Given their p-val in the LRT, we can assert that the interactions are significant and we cannot drop them, basically, incidence of property crime has a significant relationship to whether it occurs in C (College) or U (University).

Our reference region is a College in the Central region, corresponding to the following model:

$\lambda_0 = e^{-4.9006}$

In contrast, this is how the model would look like if we select a University in the southwest region:

$\lambda_1 = \frac{e^{0.3155}}{e^{0.1854}e^{0.6978}e^{4.9006}}$

However, we still have to optimize the model, to achieve this, we will use *stepAIC* in order to find the model with the best AIC.

```
stepAIC(model, list(upper=~Region*Type, lower= ~1))
#> Start:  AIC=5140.47
#> Property ~ Region + Type
#>
#>                Df Deviance    AIC
#> + Region:Type  5   3735.4 4300.5
#> <none>             4585.5 5140.5
#> - Region       5   5107.1 5652.1
#> - Type         1   5505.0 6058.1
#>
#> Step:  AIC=4300.46
#> Property ~ Region + Type + Region:Type
#>
#>                Df Deviance    AIC
#> <none>             3735.4 4300.5
#> - Region:Type  5   4585.5 5140.5
#>
#> Call:  glm(formula = Property ~ Region + Type + Region:Type, family = poisson,
```

```
#>     data = crime, offset = log(Enrollment))
#>
#> Coefficients:
#>    (Intercept)        RegionMW        RegionNE        RegionSE        RegionSW
#>        -4.9006         -0.4273          1.1290          0.1818         -0.1854
#>        RegionW           TypeU  RegionMW:TypeU  RegionNE:TypeU  RegionSE:TypeU
#>        -0.1501          0.6978          0.6607         -0.9900          0.3294
#> RegionSW:TypeU   RegionW:TypeU
#>         0.3155          0.5730
#>
#> Degrees of Freedom: 80 Total (i.e. Null);  69 Residual
#> Null Deviance:       5979
#> Residual Deviance: 3735  AIC: 4300
```

We see that according to stepAIC, the best model is the one that includes all the interactions between the variables. Returning a model with an AIC of 4300.

Exercise 6

Importing the data:

```
ofp <- read.csv('../data/dt.csv')
```

Predicting number of physician office visits using poisson regression

Creating the model:

```
mod <- glm(ofp~., family=poisson, data=ofp)
```

```
exp(coef(mod))
#>      (Intercept)            hosp        numchron          gender
#>        2.7979142       1.1791542       1.1579362       0.8937583
#>           school         privins health_excellent     health_poor
#>        1.0264877       1.2234649       0.6962871       1.2818534
```

We can see in the model that according to its definition, an increase of one unit in hospital stays increases in ~1.179x the amount of visits. Private insurance increases the rate by ~1.223x, and it makes sense, as individuals might be incentivized use their private insurance for this purpose. Males tend to make less appointments (89% that of females). Self-perceived good health also has a significant impact on hospital visits, where excellent health self-perception have fewer hospital visits vs the opposite, ~0.696x for excellent health vs ~1.28x for poor health individuals.

Comparing number of zero-visit counts to the number predicted by the model

The data shows a count of 683 for zero-visits, however the model predicts that such amount is equal to 0.

There might be several reasons to explain this, but maybe the algorithm considers that if there exists a probability, no matter how small, that the person will attend to the physician, then the model will choose to at least give a positive amount to a person's visit.

Another reason might be that there might not be a big enough distinction between individuals that have zero-visits and not, therefore causing the model to still at least allocate a very low amount of physician visits to people, and so as a result, we get no zero-visits patients.

```
vct <- predict(mod, newdata=ofp, type = "response")
vct[vct == 0]
#> named numeric(0)
```

Estimating the zero-inflated Poisson regression model and predicting number of physician office visits (all explanatory variables for $log(\mu)$ part and none in the ϕ part of the model)

```
mod_zip <- zeroinfl(ofp~. | 1, data=ofp)
```

```
exp(coef(mod_zip))
#>      count_(Intercept)            count_hosp          count_numchron
#>              4.0219045             1.1724919               1.1090303
#>           count_gender          count_school           count_privins
#>              0.9368315             1.0199078               1.0896620
#> count_health_excellent     count_health_poor       zero_(Intercept)
#>              0.7267415             1.2896117               0.1754406
```

A few changes have occurred versus the previous model. Now hospital visit counts increases the visits in ~1.17249x the visits to the physician. Males have a smaller physician gaps versus the previous model (males represent 93.6% of what females represent). The significance of private insurance is lower (~1.08966x vs ~1.223x). For self-perceived good health, the difference is also significant, but perhaps less than for the rest of the variables, where poor health doesnt seem to affect the prediction much more than in the previous model and (~1.289x vs ~1.282x). For excellent health individuals, this model will predict more visits than the previous one, but not by a lot (~0.7267x vs ~0.69628x).

Using all the explanatory variables to estimate the previous ZIP model and comparing using LRT.

```
mod_zip_2 <- zeroinfl(ofp~ 1 | ., data=ofp)
```

```
exp(coef(mod_zip_2))
#>     count_(Intercept)      zero_(Intercept)             zero_hosp
#>             6.8238841             0.9805394             0.6984931
#>         zero_numchron           zero_gender           zero_school
#>             0.5756269             1.5276620             0.9423984
#>          zero_privins zero_health_excellent      zero_health_poor
#>             0.4709398             1.3271686             1.0016981
```

We can see there are significant differences with the previous model. There is a significant ratio difference between males and females (could perhaps represent men lying about hospital visits) with a ~1.527x increase. Increase chronic conditions and hospital visits both reduce the factor (could perhaps represent increase in this unit for certain people that they are significantly less likely to lie, by ~0.5756x and ~0.6985x respectively).

From the perspective of truthfulness, again, private insurance means that people will *most likely* be truthful when claiming an amount of visits to the physician. (~0.4706x likely to lie)

Self-perceived good health flips as well, with ~1.327x and ~1.002x for poor health.

Here we see the ratios between the 1st ZIP model and 2nd ZIP model's coefficients (the results shown are not correct for intercepts but yes for the rest of the variables):

```
ratios <- c()
zip1 <- exp(coef(mod_zip))
zip2 <- exp(coef(mod_zip_2))
names(zip1) <- str_replace(names(exp(coef(mod_zip))), "count", "zero")
for (i in names(zip1)) {
    zip1_l <- zip1[i]
    zip2_l <- zip2[i]
    ratios <- c(ratios, zip2_l/zip1_l)
}
ratios
#>      zero_(Intercept)             zero_hosp         zero_numchron
#>             0.2437998             0.5957339             0.5190362
#>           zero_gender           zero_school          zero_privins
#>             1.6306689             0.9240036             0.4321888
#> zero_health_excellent      zero_health_poor      zero_(Intercept)
#>             1.8261908             0.7767440             0.2437998
```

Performing a LRT:

```
lrtest(mod_zip, mod_zip_2)
#> Likelihood ratio test
#>
#> Model 1: ofp ~ . | 1
#> Model 2: ofp ~ 1 | .
#>   #Df LogLik Df  Chisq Pr(>Chisq)
#> 1   9 -16302
#> 2   9 -17288  0 1972.5  < 2.2e-16 ***
```

```
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

When performing a LRT shows that we can reject the null hypothesis and we can keep the first ZIP model, the 2nd model would be very difficult to justify, as our interpretations are based on a speculative idea (people lying) other than the actual data.

Examining how well the models estimate 0 counts

Predicting the counts using the parameter *type=response* for the first ZIP model:

```
vct <- predict(mod_zip, newdata=ofp, type = "response")
vct[vct == 0]
#> named numeric(0)
```

Doing so for the second yields the same result:

```
vct <- predict(mod_zip_2, newdata=ofp, type = "response")
vct[vct == 0]
#> named numeric(0)
```

Which hints that perhaps the right approach is to estimate such value using the probabilities obtained:

```
apply(predict(mod_zip, newdata=ofp, type = "prob")*4406,2,mean)[1]
#>       0
#> 672.297
apply(predict(mod_zip_2, newdata=ofp, type = "prob")*4406,2,mean)[1]
#>        0
#> 681.6794
```

We can see that these sort of approximate the real value of 683.