Logistic regression

Zoltan Kekecs

November 10, 2021

Logistic regression

Abstract

In this exercise we will learn how to make predictive models on binomial outcome variables. We will mainly discuss using logistic regression.

Loading packages

```
library(pscl) # for pR2
library(lmtest) # for lrtest
library(dominanceanalysis) # for dominanceAnalysis()
library(tidyverse) # for dplyr and ggplot2
```

Data management and descriptive statistics

Our dataset

We will use the **Heart Disease dataset**, a well-known dataset used to demonstrate classifications problems. The dataset contains different information about patients who were screened for the presence of heart disease.

In the code below "dec ="," " indicates that the decimal sign is "," in this online dataset (instead of "." which is the default setting in R).

```
heart_data = read.csv("https://raw.githubusercontent.com/kekecsz/SIMM61-Course-materials/main/Exercise_
```

The following variables are included in the dataset:

- age - age in years
- sex - (1 = male; 0 = female)
- cp - chest pain type (0 = asymptomatic, 1 = typical angina, 2 = atypical angina, 3 = non-anginal pain)
- trestbps - resting systolic blood pressure (in mm Hg on admission to the hospital)
- chol - serum cholesterol in mg/dl
- fbs - (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
- restecg - resting electrocardiographic results: (0 = normal; 1 = having ST-T wave abnormality; 2 = showing probable or definite left ventricular hypertrophy)
- thalach - maximum heart rate achieved during the exercise test
- exang - exercise induced angina (1 = yes; 0 = no)
- oldpeak - ST depression induced by exercise relative to rest
- slope - the slope of the peak exercise ST segment
- ca - number of major vessels (0-3) colored by flourosopy
- thal - 3 = normal; 6 = fixed defect; 7 = reversable defect
- disease_status - have disease or not (heart_disease vs. no_heart_disease)

See more information about the dataset here:

https://www.kaggle.com/ronitf/heart-disease-uci; https://archive.ics.uci.edu/ml/datasets/Heart+Disease

Data management

We start with tidying up our dataset, defining categorical variables as factors, recoding factor levels so that they are more informative, and giving new variable names that are more informative.

```
heart_data = heart_data %>%
  mutate(sex = factor(recode(sex,
                      "1" = "male",
                      "0" = "female")),
         cp = factor(recode(cp,
                            "0" = "asymptomatic",
                            "1" = "typical_angina",
                            "2" = "atypical angina",
                            "3" = "non_anginal_pain")),
         fbs = factor(recode(fbs,
                                    "1" = "true",
                                    "0" = "false")),
         disease_status = factor(disease_status)
         ) %>%
  rename(chest_pain = cp,
         sys_bloodpressure = trestbps,
         blood_sugar_over120 = fbs,
         max_HR = thalach,
         cholesterol = chol)

names(heart_data)[1] = "age"
```

Checking data

You should always check the dataset for coding errors or data that does not make sense, and explore the dataset to get a basic feel of what type of data you are dealing with. View raw data in the data viewer and display simple descriptive statistics and plots as learned in previous exercises.

```
heart_data %>%
  summary()
```

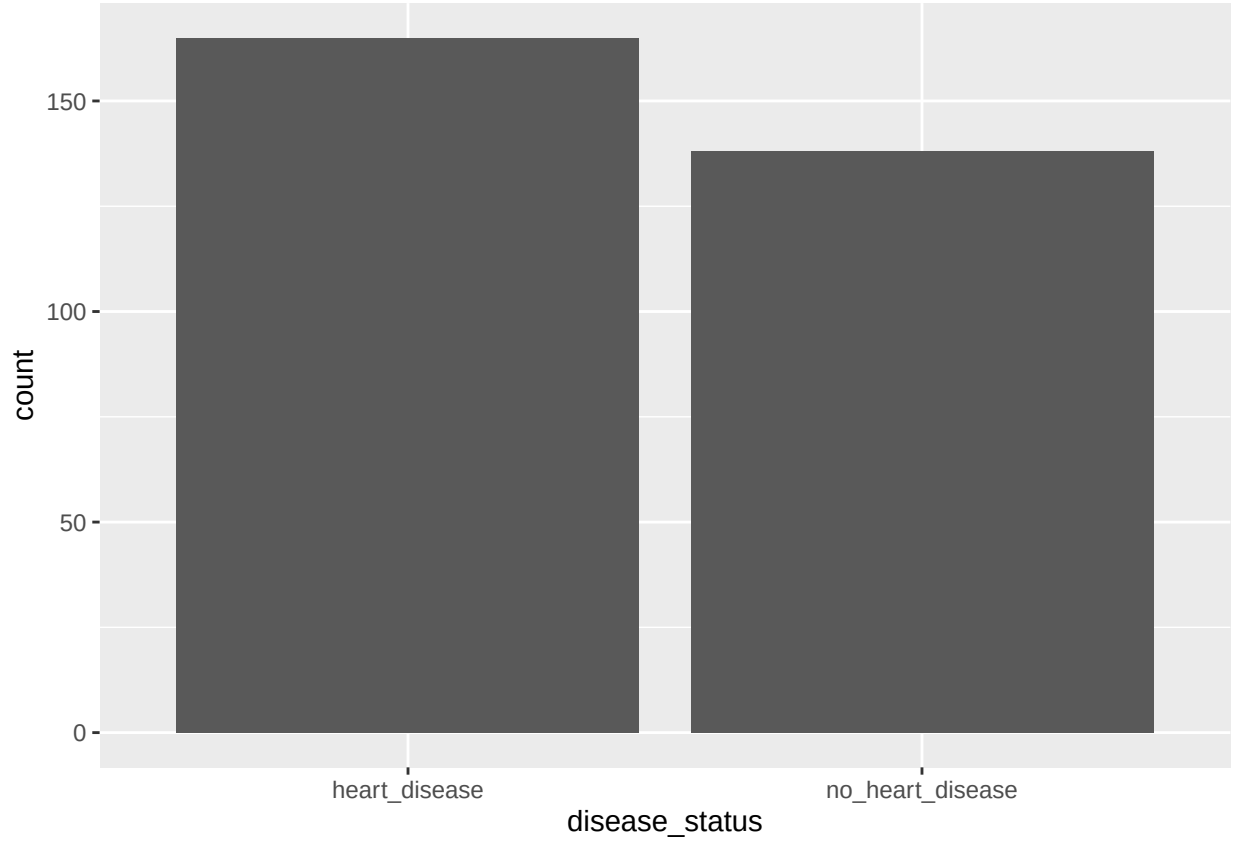
```
##       age            sex                  chest_pain  sys_bloodpressure
##  Min.   :29.00   female: 96   asymptomatic    :143   Min.   : 94.0
##  1st Qu.:47.50   male  :207   atypical angina : 87   1st Qu.:120.0
##  Median :55.00                non_anginal_pain: 23   Median :130.0
##  Mean   :54.37                typical_angina  : 50   Mean   :131.6
##  3rd Qu.:61.00                                       3rd Qu.:140.0
##  Max.   :77.00                                       Max.   :200.0
##   cholesterol    blood_sugar_over120    restecg           max_HR
##  Min.   :126.0   false:258           Min.   :0.0000   Min.   : 71.0
##  1st Qu.:211.0   true : 45           1st Qu.:0.0000   1st Qu.:133.5
##  Median :240.0                       Median :1.0000   Median :153.0
##  Mean   :246.3                       Mean   :0.5281   Mean   :149.6
##  3rd Qu.:274.5                       3rd Qu.:1.0000   3rd Qu.:166.0
##  Max.   :564.0                       Max.   :2.0000   Max.   :202.0
##      exang           oldpeak         slope             ca
##  Min.   :0.0000   Min.   :0.00   Min.   :0.000   Min.   :0.0000
```

```
##  1st Qu.:0.0000   1st Qu.:0.00   1st Qu.:1.000   1st Qu.:0.0000
##  Median :0.0000   Median :0.80   Median :1.000   Median :0.0000
##  Mean   :0.3267   Mean   :1.04   Mean   :1.399   Mean   :0.7294
##  3rd Qu.:1.0000   3rd Qu.:1.60   3rd Qu.:2.000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :6.20   Max.   :2.000   Max.   :4.0000
##       thal               disease_status
##  Min.   :0.000   heart_disease   :165
##  1st Qu.:2.000   no_heart_disease:138
##  Median :2.000
##  Mean   :2.314
##  3rd Qu.:3.000
##  Max.   :3.000
```

Research question and exploratory data analysis

Our main goal in this exercise will be to **create a model that can effectively predict disease_status**, that is, whether the person **has or does not have heart disease**. For this we will use three main predictors: maximum heart rate achieved during the exercise test (max_HR), resting systolic blood pressure (sys_bloodpressure), and the presence of chest pain (chest_pain). So lets explore these variables visually.

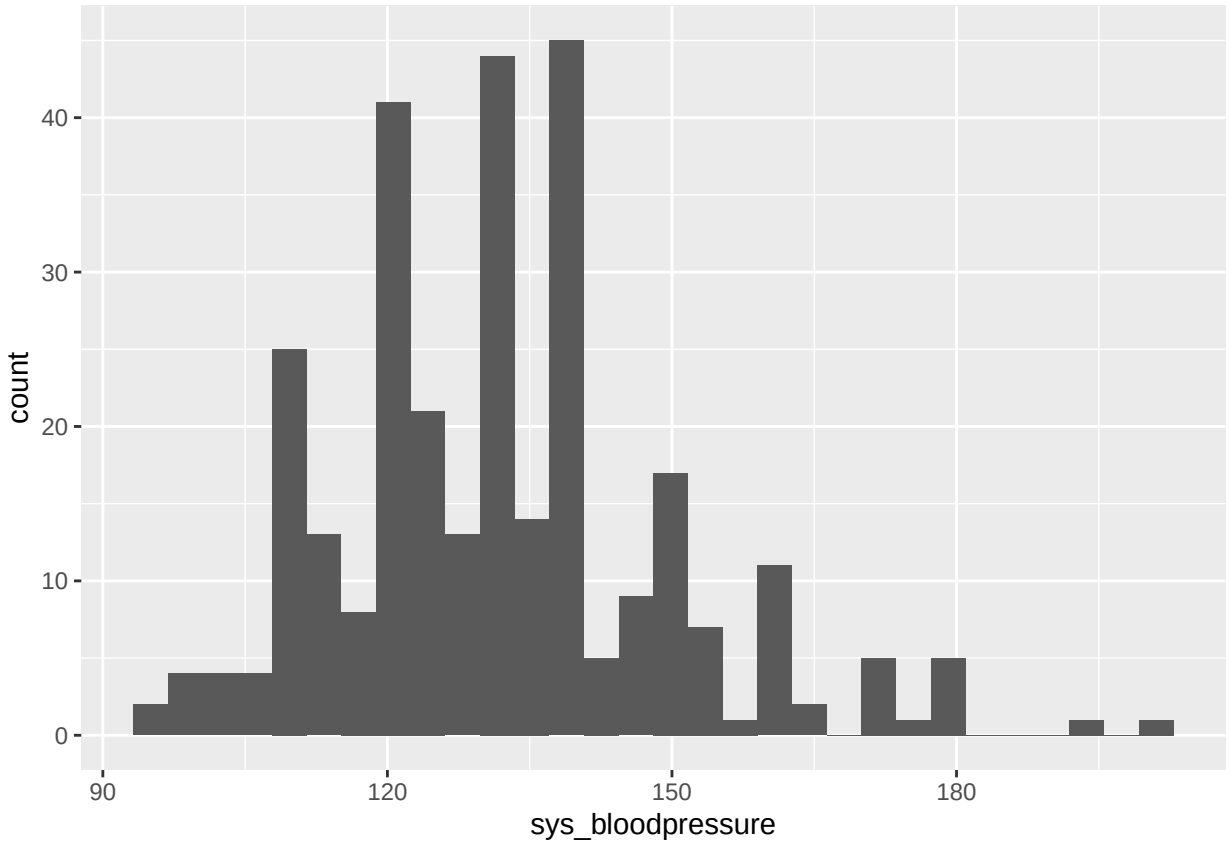
```
heart_data %>%
  ggplot() +
    aes(x = disease_status) +
    geom_bar()
```




```
heart_data %>%
  ggplot() +
```

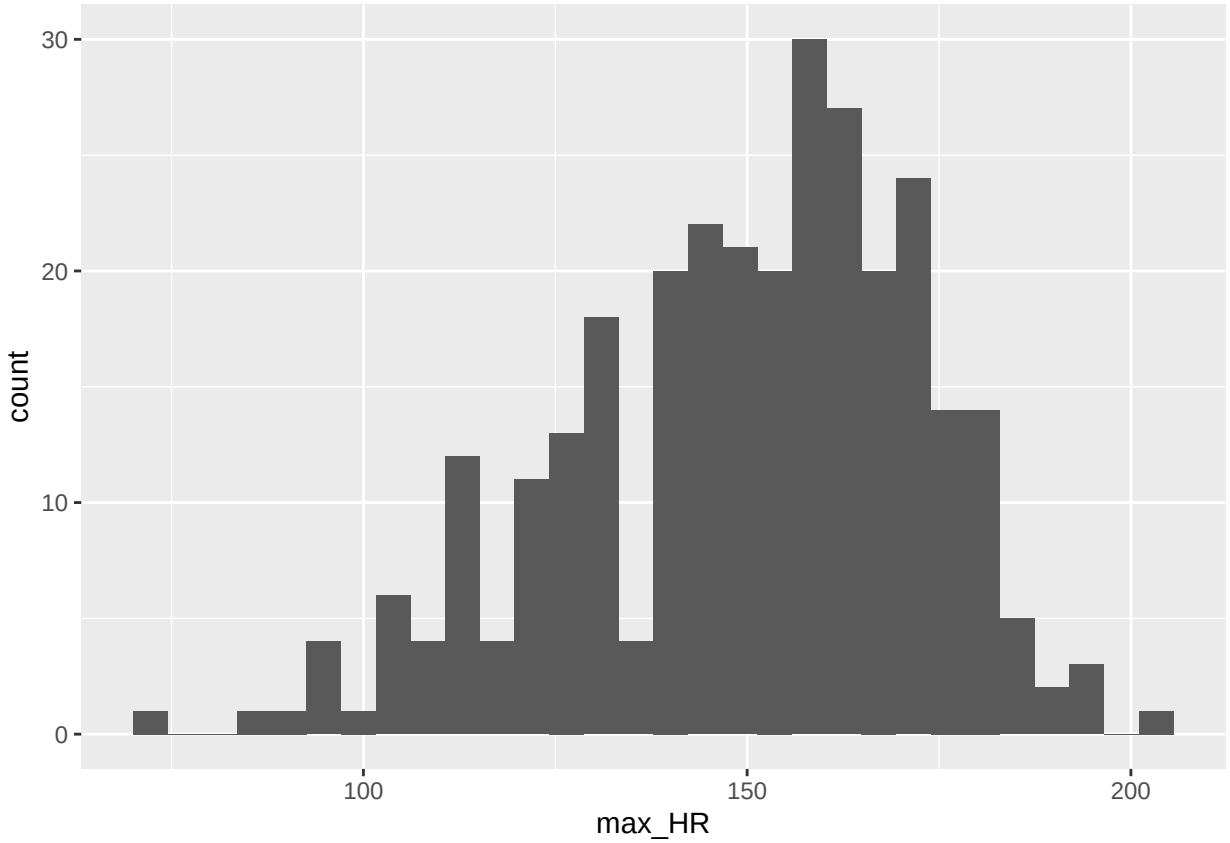
```
    aes(x = sys_bloodpressure) +
    geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

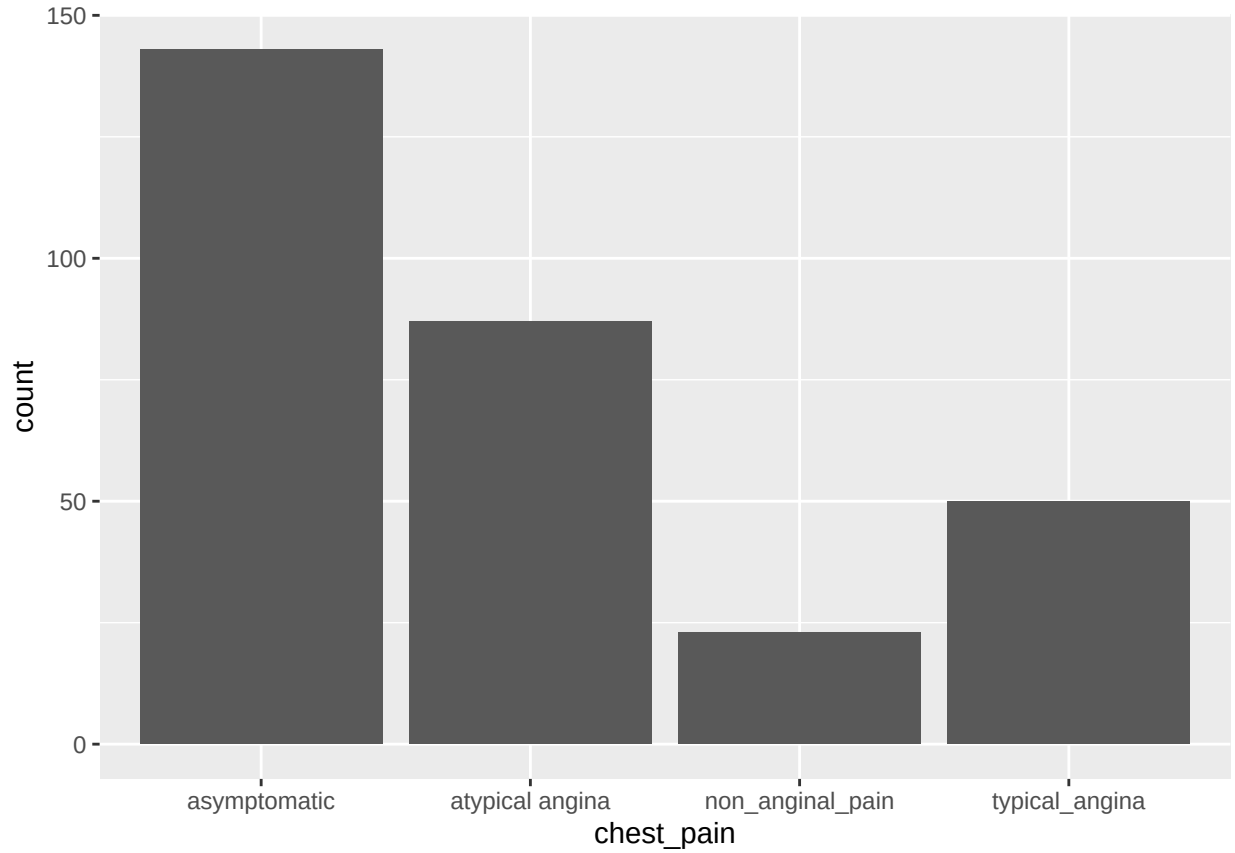



```
heart_data %>%
  ggplot() +
    aes(x = max_HR) +
    geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



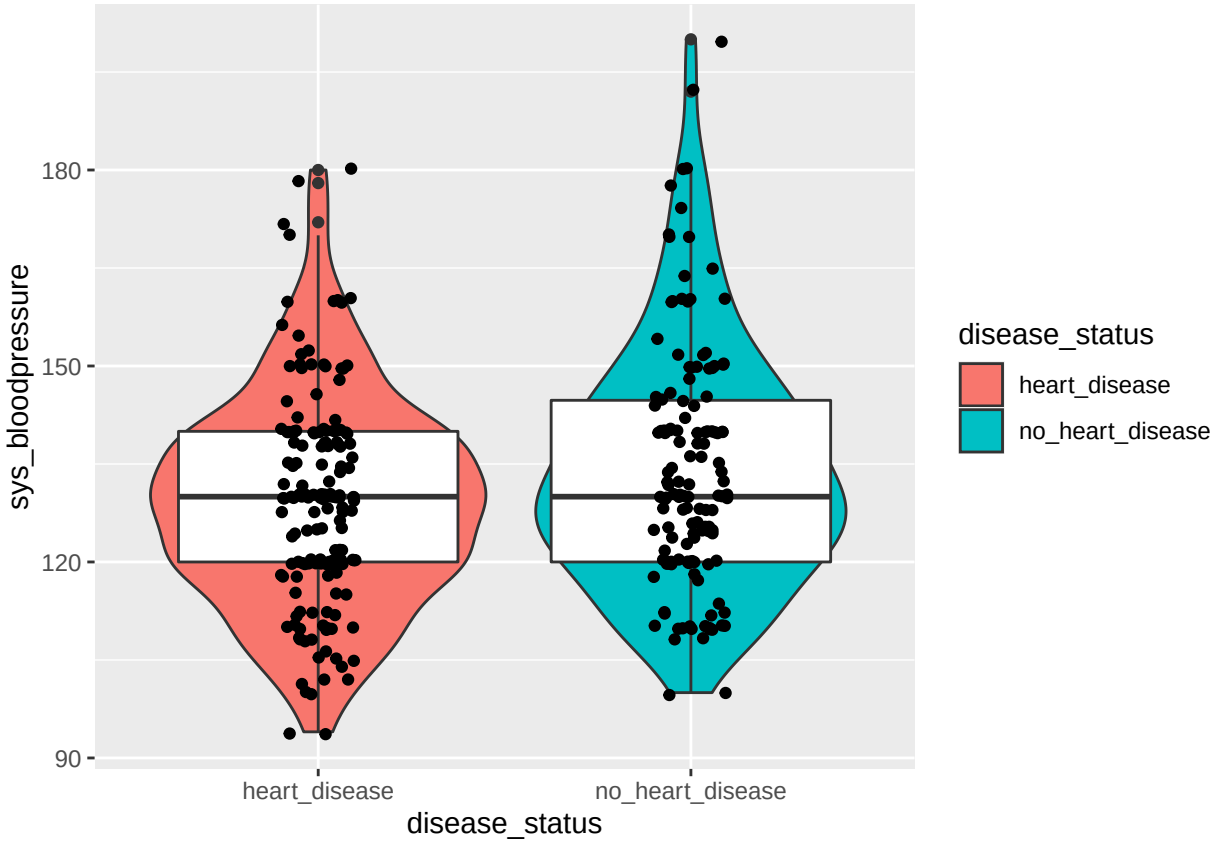

```
heart_data %>%
  ggplot() +
    aes(x = chest_pain) +
    geom_bar()
```




```
heart_data %>%
  group_by(disease_status) %>%
  summarize(mean = mean(sys_bloodpressure),
            sd = sd(sys_bloodpressure))
```

```
## # A tibble: 2 x 3
##   disease_status    mean    sd
##   <fct>            <dbl> <dbl>
## 1 heart_disease     129.  16.2
## 2 no_heart_disease  134.  18.7
```

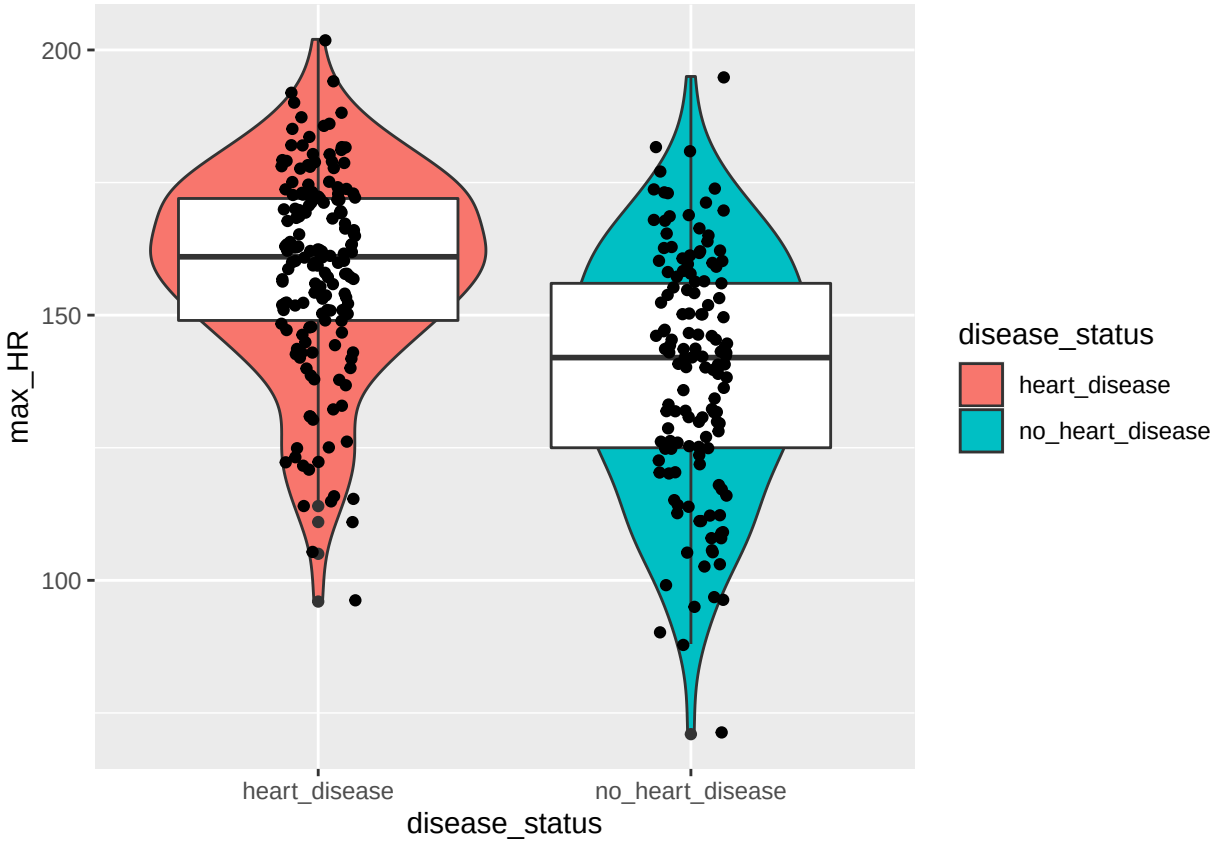
```
heart_data %>%
  ggplot() +
    aes(y = sys_bloodpressure, x = disease_status) +
    geom_violin(aes(fill = disease_status)) +
    geom_boxplot() +
    geom_jitter(width = 0.1)
```




```
heart_data %>%
  group_by(disease_status) %>%
  summarize(mean = mean(max_HR),
            sd = sd(max_HR))
```

```
## # A tibble: 2 x 3
##   disease_status    mean    sd
##   <fct>            <dbl> <dbl>
## 1 heart_disease     158.  19.2
## 2 no_heart_disease  139.  22.6
```

```
heart_data %>%
  ggplot() +
    aes(y = max_HR, x = disease_status) +
    geom_violin(aes(fill = disease_status)) +
    geom_boxplot() +
    geom_jitter(width = 0.1)
```

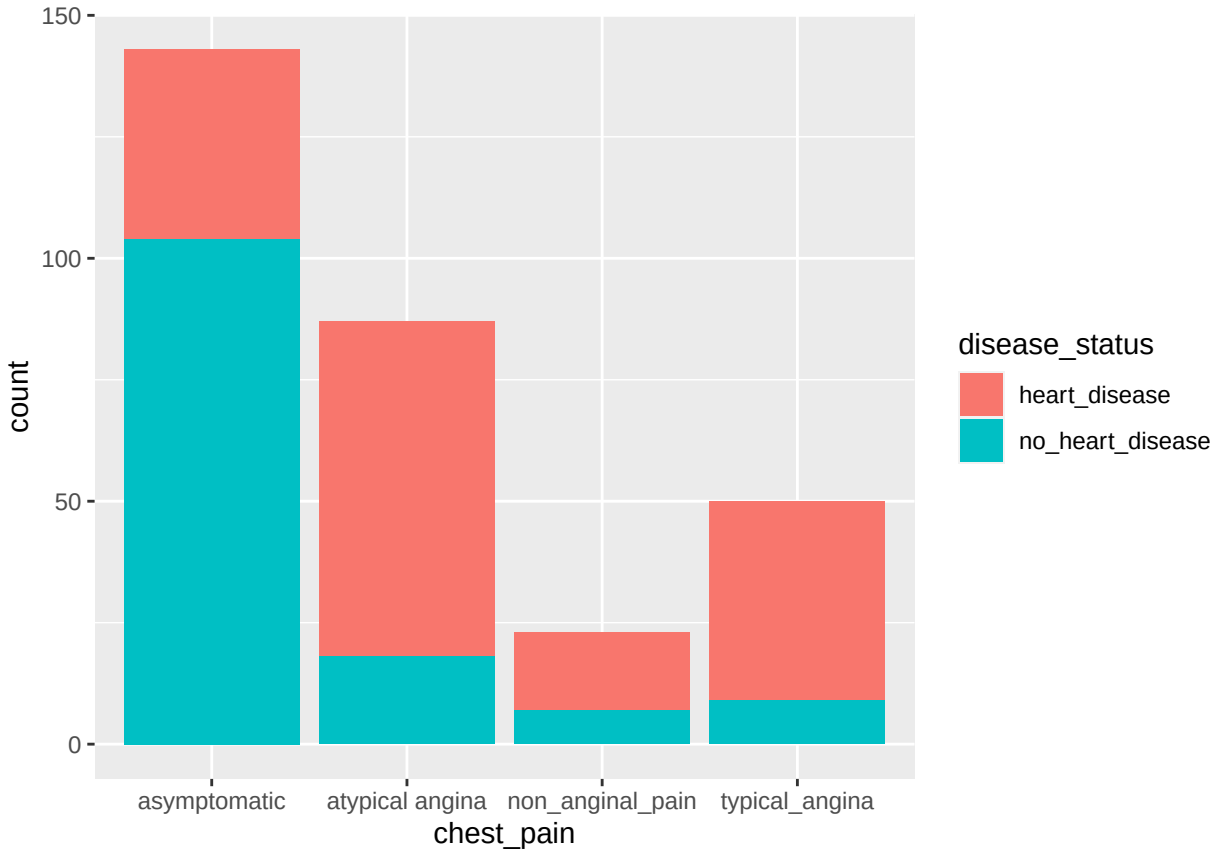



```
heart_data %>%
  group_by(disease_status, chest_pain) %>%
  summarize(n = n()) %>%
  spread(disease_status, n)
```

```
## `summarise()` has grouped output by 'disease_status'. You can override using the `.groups` argument.
```

```
## # A tibble: 4 x 3
##   chest_pain       heart_disease no_heart_disease
##   <fct>                    <int>            <int>
## 1 asymptomatic                39              104
## 2 atypical angina             69               18
## 3 non_anginal_pain            16                7
## 4 typical_angina              41                9
```

```
heart_data %>%
  ggplot() +
    aes(x = chest_pain, fill = disease_status) +
    geom_bar()
```



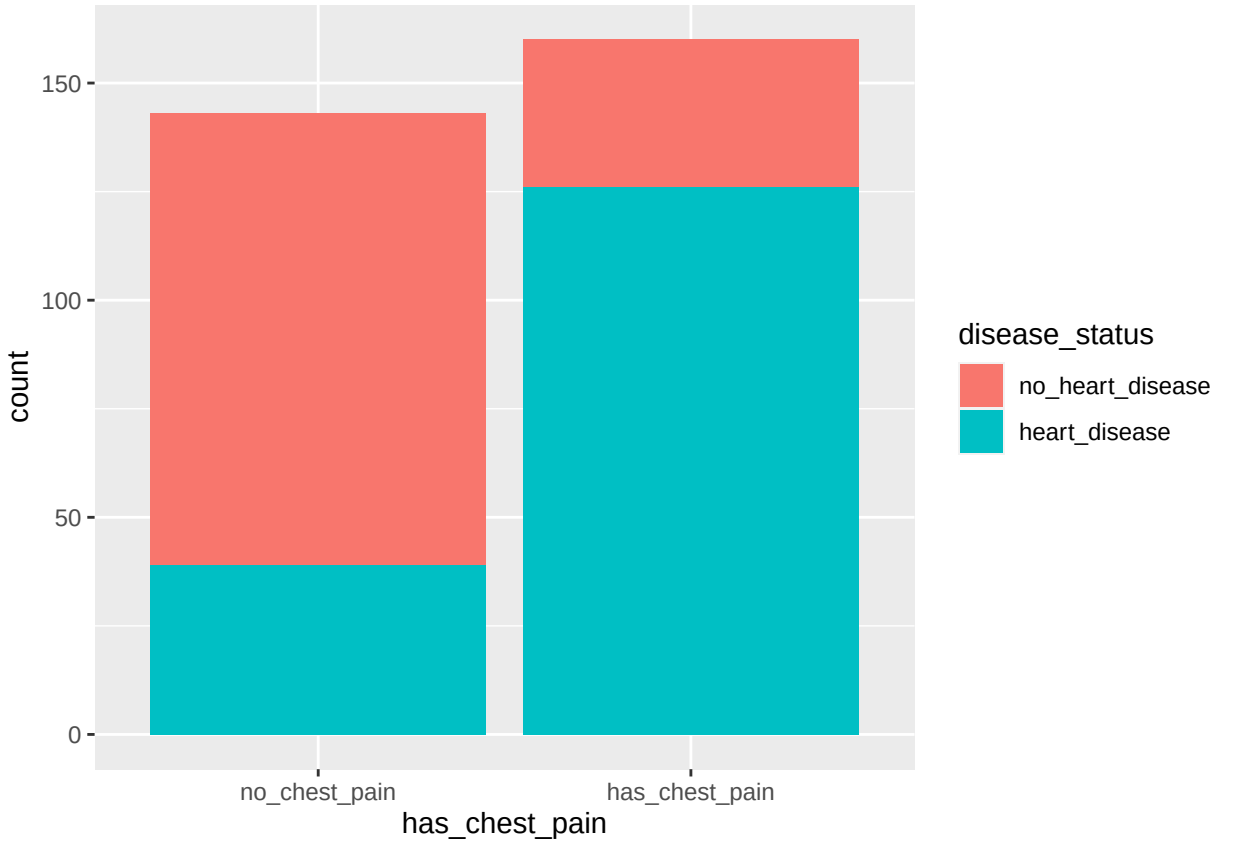
Based on this exploratory analysis on the relationship of blood pressure, heart rate, and chest pain with disease status it seems that **systolic blood pressure** has only minor relationship, while **exercise-induced heart rate** has a more substantial relationship with heart disease status.

Also, **chest pain** seems to be indicative of heart disease, but it does not seem that there would be an important distinction between the different chest pain categories, so lets **combine the three chest pain types** to form a single group against the asymptomatic group. Thus, we create a new variable called has_chest_pain with the possible levels "no_chest_pain" and "has_chest_pain".

It is also apparent from the exploration that in the factor disease_status, the **reference factor level** is "heart_disease". In a regression framework when we want to predict an event of interest, it is better to have the "no event" as the reference level, and the "event" as the non-reference level so that positive regression coefficients will correspond with a higher chance of the event of interest. Thus, we specify that "no_heart_disease" should be the reference level for disease_status. For similar reasons, we specify that "no_chest_pain" should be the reference level for the variable has_chest_pain.

```
heart_data = heart_data %>%
  mutate(has_chest_pain = factor(recode(chest_pain,
                                        "asymptomatic" = "no_chest_pain",
                                        "typical_angina" = "has_chest_pain",
                                        "atypical angina" = "has_chest_pain",
                                        "non_anginal_pain" = "has_chest_pain"),
                                        levels = c("no_chest_pain", "has_chest_pain")),
         disease_status = factor(disease_status,
                                 levels = c("no_heart_disease", "heart_disease"))
         )

heart_data %>%
```

```
  ggplot() +
    aes(x = has_chest_pain, fill = disease_status) +
    geom_bar()
```



Logistic regression analysis

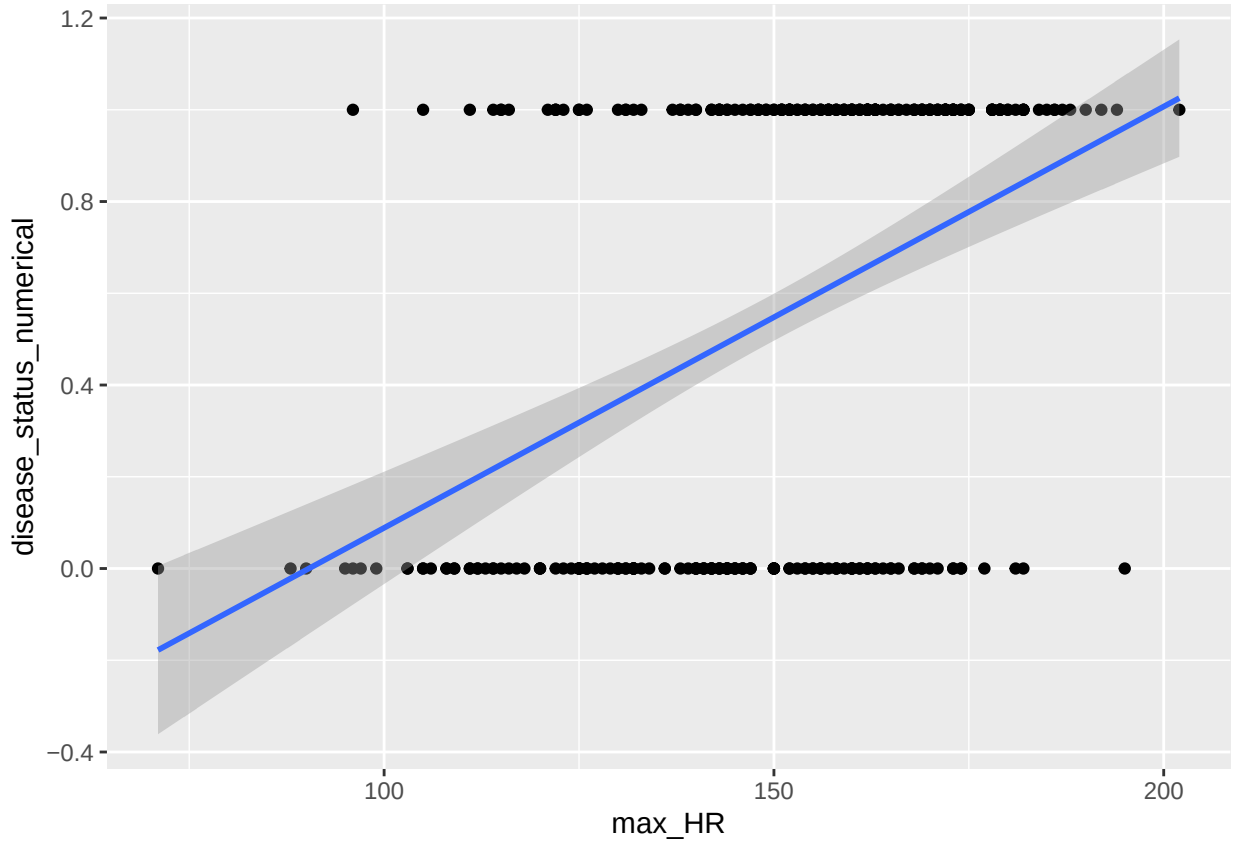
The inadequacy of linear regression for predicting categorical outcomes

Based on your experiences you might think that fitting a regression model on the data would be a good solution here, so let's start with looking at what would be the result of a regular linear regression with this data. (This just serves as a demonstration here. Don't do this at home kids!)

Let's see the relationship of a potential predictor and the predicted outcome. Here we plot the scatterplot of disease status and the maximum heart rate reached during exercise using a scatterplot to see the regression line that we would get with a simple linear regression.

```
heart_data = heart_data %>%
  mutate(disease_status_numerical = recode(disease_status,
                                           "no_heart_disease" = 0,
                                           "heart_disease" = 1))

heart_data %>%
  ggplot() +
    aes(y = disease_status_numerical, x = max_HR) +
    geom_point() +
    geom_smooth(method = "lm")
```

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




```
lm(disease_status_numerical ~ max_HR, data = heart_data)
```

```
##
## Call:
## lm(formula = disease_status_numerical ~ max_HR, data = heart_data)
##
## Coefficients:
## (Intercept)       max_HR
##   -0.829920     0.009185
```

Multiple problems are apparent from the graph and the regression equation:

1. The regression line **fits very poorly** on the data.
2. The regression equation results in **meaningless predicted values**. If we fit a regular regression model with the just created disease_status_numerical variable as an outcome and max_HR as the independent predictor variable, we get a regression coefficient of 0.009 for max_HR. This means that for every 1 points of heart rate increase, there is a 0.009 point increase in the outcome variable. If we now compute the expected outcome value for a person with 120 as the highest heart rate achieved. We would get -0.83 + 0.009*120 = 0.25. This prediction does not really make sense when the outcome can only be 0 or 1. We might think of this predicted value as the probability of having a value of 1 instead of 0 on the outcome variable. But we can also see that the model could easily return negative numbers as predictions as well, while probabilities can only take a value of 0 or 1. For example a very sporty person might only produce 90 as the highest heart rate on the exercise test that was used in this study. The predicted "probability" for this person having a heart disease would be -0.02, but there is no such thing as a negative probability for an event. So we need another approach that would return realistic predictions.

Logistic regression basic idea

The solution is to use **Generalized Linear Models (GLMs)** for prediction instead of regular Linear Models. GLMs are designed to model outcome variables that are not normally distributed. One member of this family of GLMs is Logistic regression, which is specifically designed to model binary outcomes (categorical outcome variables with only two levels).

The basic idea in GLMs is to use a link function that transforms the predicted outcome variable to a scale that would put the results of the regression on a realistic scale. The link function used by logistic regression is the logit function (which is the natural logarithm of the odds of the predicted event). So **in logistic regression** instead of predicting the actual outcome value (1 or 0), **we predict the log-odds of the outcome value**. This now can be treated in the regular linear regression framework, since log(odds) can take any value between negative infinity to infinity. So after we computed the regression equation for the log(odds) for any case, we can derive the odds or the probability of the outcome event from the regression equation as well.

The logistic regression model in R

Now let's try this in action. We will build a logistic regression model, where we predict disease status with the predictor max_HR.

The regression model is built with a similar formula as the linear regression, with the exception that we use the glm() function and we need to specify the family = binomial() to get a logistic regression.

```
mod1 = glm(disease_status_numerical ~ max_HR,
           family = binomial(), data = heart_data)

summary(mod1)
```

```
##
## Call:
## glm(formula = disease_status_numerical ~ max_HR, family = binomial(),
##     data = heart_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1383  -1.0780   0.6043   0.9200   2.1354
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.391452   0.987133  -6.475 9.50e-11 ***
## max_HR       0.043951   0.006531   6.729 1.71e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 417.64  on 302  degrees of freedom
## Residual deviance: 359.26  on 301  degrees of freedom
## AIC: 363.26
##
## Number of Fisher Scoring iterations: 4
```

Odds, odds ratio, and probability

Now that we have built a model we can look at what the model predicts. We can calculate the predicted value for each observation by using the regression equation just as before, but we need to remember that the value we predict is the **natural logarithm of odds of the event of interest** happening vs. it not happening.

This log(Odds) of the event is hard to interpret in its raw form, so we usually **convert this to Odds, or probabilities**. For this to make sense, we need to understand the meaning of Odds and probabilities and how to convert one into the other.

Odds

The odds of some event reflect the likelihood that the event will take place. It is calculated as the **ratio of the number of observations that produce the event of interest to the number of observations that do not**. The odds is usually represented as a pair of numbers. If the event/outcome of interest is surviving an airplane crash, and if the odds for survival are 1 to 4, this means that on average for every 1 person surviving, there are 4 people who do not survive. This can be also written up as the odds being 0.25 since $1/4 = 0.25$. If the odds for surviving the plane crash are 2 to 1, this means that on average for every 2 survivors there is 1 person who dies. The odds here can also be represented as 2, since $2/1 = 2$. We can compute the odds from ln(Odds) by using the exp() function: Odds = exp(ln(Odds)).

It is important to realize that the **value of odds depends on our perspective**, on what is our event of interest. In this example our event of interest is surviving, and if we switch the event of interest to not surviving (dying) in the plane crash (the other possible outcome) the odds would be the inverse of the odds of survival. So if the odds of survival is 0.25, the odds of dying is $1/0.25 = 4$. If the odds of survival is 2, the odds of dying would be $1/2 = 0.5$.

Odds ratio

Another important concept related to odds is that of odds ratio. This is an effect size measure, which can be used to assess the effect of being a member of a certain group vs. not being the member of that group on the odds of the event of interest. The odds ratio is the **ratio of the odds of the event of interest between two groups**. It is computed by dividing the odds of the event of interest in the group by the odds of the event in outside of this group. It basically represents how much higher or lower the odds (risk) for an event is in one group over another group. The easiest example is if we compare the odds of an event in two groups, so let's compute the odds ratio related to biological sex for dying in a plane accident. Let's imagine that for every 1 male who survives a plane crash there are 6 who die in the plane crash on average, making the odds of death in a plane accident 6 for males. Let's suppose that for some reason, women are less at risk of dying in a plane crash: for every 1 female who survives the crash there are 2 females who die, so the odds of death in a plane crash for females is 2. Now we can calculate the odds ration by dividing the two odds: $6/2 = 3$. So the odds ratio of males dying in a plane accident over females is 3, because the odds for males to die in a plane crash is 3 times as that of females.

Just like with odds, the odds ratio is also **dependent on our perspective**: on what is the group of interest. So if the odds ratio of males (vs. females) dying in a plane crash is 3, the odds ratio of females (vs. males) for dying in a plane crash is $1/3 = 0.3333$. The odds ratio of 0.33 meaning that the odds of die in a plane accident is 0.33 times as large of females compared to males. In other words, the odds of dying is 33% among females compared to males.

Probability

Probability is the **fraction of times you expect to see the event of interest out of many (all) observations**. If the probability is $1/4$ we expect to see that event 1 out of 4 times on average. So if the probability of surviving a plane crash is $1/4 = 0.25 = 25\%$, we expect that out of 4 people 1 will survive and 3 will die. On the other hand if the probability is $3/4 = 0.75 = 75\%$, we would expect that on average 3 will survive and 1 will die out of 4 people. We can compute the probability from log(Odds) by the following formula: p = exp(log(Odds))/ (1 + exp(log(Odds))), in other words Odds / (1 + Odds).

Interpreting the results of logistic regression

Now that we know what odds, odds ratio, and probability are and how they relate to each other, we are ready to interpret the model's results. Lets use the **regression equation to get a predicted value** for an

individual with a maximum heart rate of 182 in the exercise test. According to our model, the regression equation would return: -6.39 + 0.044 * 182 = 1.628. This is in log(odds), which can range from negative to positive infinity. If the value is negative, it means that the likelihood that the event of interest happens is smaller than the likelihood that it does not happen. So in our case, since the log(odds) is a positive number, we can say that it is more likely that the person has heart disease (this is our event of interest) than the person not having heart disease. However, other than this, it is hard to interpret the log(odds) any further without converting it to odds.

We can convert log(odds) to odds by using the exp() function. exp(1.628) = 5.09. So the odds of the person having heart disease vs. not having it is 5.09, meaning that it is about 5 times more likely that this person has heart disease than it is that the person not having it. We can also convert this to probability using the formila above: p = Odds / (1 + Odds). In our case 5.09/ (1 + 5.09) = 0.84, meaning that the probability of this person having heart disease is 84% according to our model.

As seen earlier, we dont have to calculate this by hand for every person, we can get the predicted values for each observation in our dataset by using the predict() function on the model object.

```
predict(mod1)
```

Model performance

R squared

A familiar indicator of the model's predictive power is the R^2 index. However, there is **no exact R^2 index for logistic regression**. Instead, the proportion of explained variance is estimated using different statistical procedure which are called pseudo R squared methods. There are multiple pseudo R squared indices, such as the Cox and Snell R^2, Nagelkerke R^2, and the McFadden R^2. None of these is universally accepted as a good R^2 estimate, but Cox and Snell R^2, and the Nagelkerke R^2 suffer from serious drawbacks, so if we use R^2 at all for logistic regression it is the McFadden R^2.

The issue with Cox and Snell R^2 is that it has an upper limit which is lower than 1. For those of us who are used to the good old R^2 index which can take any value between 0 and 1 and thus represents the proportion of explained variance, this might make the Cox and Snell R^2 hard to interpret. The Nagelkerke R^2 was devised to counteract this by expanding the scale of the Cox and Snell R^2 so that it is able to extend to 1. However, the correction used for this is often seen as an overcompensation and this can return unrealistically large R^2 values.

So we are left with the **McFadden R^2**. We can get this index by running the pR2() function from the pscl package.

The output of pR2() also shows us the **log likelihood** of the model under "llh". By multiplying this number with -2 it is easy to calculate the **-2 Log Likelihood (-2LL)**, which is also called **"Deviance"** in the literature. This has the same general meaning as the residual sum of squares (RSS) in regular regression. It compares the difference in probability between the predicted outcome and the actual outcome for each case and sums these differences together to provide a measure of the total error in the model. Just like RSS, and AIC, -2LL is hard to interpret outside of the context of the specific model, since its value depends on sample size, the number of parameters in the model and the goodness of fit. What is important to understand is that it shows the amount of error left after accounting for all of the variance explained by the predictors in our model, and that **the lower this number the better the model fit** is. (Conversely, the higher the log likelihood, the better the model fit).

```
pR2(mod1)
```

```
## fitting null model for pseudo-r2
```

```
##          llh      llhNull           G2     McFadden         r2ML         r2CU
## -179.6284602 -208.8190283   58.3811363    0.1397888    0.1752517    0.2342923
```

```
# -2LL, deviance
pR2(mod1)["llh"] * -2
```

```
## fitting null model for pseudo-r2
```

```
##      llh
## 359.2569
```

Prediction accuracy for categorization

In some situations, it is not enough to get a predicted odds for the individuals, we would like to predict **which category the person would belong to**. We can use some cutoff value to categorize people into groups based on the predicted log odds, odds, or probability of the event of interest/ category of interest. The most straightforward solution is to categorize the observations as having "no event of interest" if the probability of the event of the event of interest is 50% or lower for that particular observation, or as having "event of interest" if the probability of the event of interest is higher than 50% for that observation. Note, that **50% probability corresponds with an odds of 1, and a log(odds) of 0**, so in fact we can directly use the prediction of our model (which is in log(odds)) to make this categorization: log(odds) of 0 or less will be coded as "no_heart_disease", while log(odds) larger than 0 will be coded as "heart_disease", since having heart disease is the event of interest in our study. We do this in the code below, saving the predicted value of the outcome variable (disease status) into a new varialbe called "pred_mod1".

```
heart_data = heart_data %>%
  mutate(pred_mod1 = predict(mod1)) %>%
  mutate(pred_mod1 = case_when(pred_mod1 <= 0 ~ "no_heart_disease",
                               pred_mod1 > 0 ~ "heart_disease"))
```

Now we can **compare the predictions of the model with the actual** disease status **values** to see how many times did our model correctly categorize the observations. The results indicate that the disaease status (having or not having heart disease) was correctly predicted by the model in 213 out of the 303 cases (70%).

```
# coding correct guesses

heart_data = heart_data %>%
  mutate(correct_prediction = case_when(pred_mod1 == disease_status ~ "correct",
                                        pred_mod1 != disease_status ~ "incorrect"))


# correct categorization rate overall

heart_data %>%
  group_by(correct_prediction) %>%
  summarise(count = n()) %>%
  mutate(freq = count / sum(count))
```

```
## # A tibble: 2 x 3
##   correct_prediction count  freq
##   <chr>              <int> <dbl>
## 1 correct              213 0.703
## 2 incorrect             90 0.297
```

Prediction accuracy by outcome category

This 70% prediction rate might seem like a good result, but the model's performance needs to be evaluated in the **relative to** how accurate **a model would be which does not use any predictors**.

Similarly to the regular linear models, we can build a **null model** in logistic regression as well. In linear

regression, the null model used the mean of the outcome variable as the prediction. We do the same thing in logistic regression in the null model. Specifically, we use the log(odds) calculated based on the proportion of the event of interest in the sample, as the predicted log(odds) for each observation. For example in the case of the heart disease dataset we are working with now, the null model will calculate the log(odds) of having heart disease over not having heart disease based on the proportion of cases which have heart disease in the dataset. Since 165 (54%) participants in the study had heart disease and 138 (46%) did not have heart disease, the odds of having heart disease vs. not having it is 1.2 (because there are 1.2 times as many people with heart disease in the dataset as ones without heart disease). $\log(1.2) = 0.18$. This is the number that will be used in our logistic regression model as a predicted outcome.

Of course, if we use the same simple method to derive the predicted category from this log(odds) prediction as before, we will have to predict for every single person in the dataset that they have heart disease, since 0.18 is above 0. This prediction is correct for 54% of the cases, since 54% of the people in the dataset actually have heart diasease.

```
mod_null = glm(disease_status_numerical ~ 1, family = binomial(), data = heart_data)

summary(mod_null)
```

```
##
## Call:
## glm(formula = disease_status_numerical ~ 1, family = binomial(),
##     data = heart_data)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.254  -1.254   1.103   1.103   1.103
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.1787     0.1154   1.549    0.121
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 417.64  on 302  degrees of freedom
## Residual deviance: 417.64  on 302  degrees of freedom
## AIC: 419.64
##
## Number of Fisher Scoring iterations: 3
```

```
head(predict(mod_null))
```

```
##         1         2         3         4         5         6
## 0.1786918 0.1786918 0.1786918 0.1786918 0.1786918 0.1786918
```

In other words, **the prediction of the null model will always be the more frequent category** for every observation in the dataset. This means that the more frequent one of the categories/events is in the observed outcome variable, the more accurate the null model's prediction will be overall. For example, lets say we would like to predict whether a person has COVID based on some health measures using logistic regression. However, if 95% of people does not have COVID among the people we tested, even the null model without the use of any health-related predictors will be 95% accurate in predicting the outcome, since it will predict that "no COVID" for everyone, and since 95% of people don't have COVID this prediction will be correct for 95% of the individuals. In fact, the null model will correctly categorize every single case in the more frequent category, but will be wrong in every case that belong into the other category.

This means that it is not enough to report the overall prediction accuracy of the model. It is also necessary to **report the prediction accuracy by the different outcome categories**. As we have noted already,

dataset the prediction of the null model is correct for 54% of the cases in the heart disease dataset overall, and it is correct in 100% of the cases that actually have heart disease, but it is correct in 0% of the cases that actually do not have heart disease. In comparison, the model with the predictor correctly predicted that a case has heart disease in 130 out of 165 cases with heart disease (79%), and it correctly predicted that cases do not have heart disease in 83 out of 138 cases with no heart disease (60%). So, the prediction accuracy of our model with the predictors is substantially higher for predicting both possible outcomes compared to the null model.

```
# percentage of heart disease

heart_data %>%
  group_by(disease_status) %>%
  summarise(count = n())  %>%
  mutate(freq = count / sum(count))
```

```
## # A tibble: 2 x 3
##   disease_status   count  freq
##   <fct>            <int> <dbl>
## 1 no_heart_disease   138 0.455
## 2 heart_disease      165 0.545
```

```
# crosstab of disease_status and predicted values

heart_data %>%
  group_by(disease_status, pred_mod1) %>%
  summarize(n = n()) %>%
  spread(disease_status, n)
```

```
## `summarise()` has grouped output by 'disease_status'. You can override using the `.groups` argument.
```

```
## # A tibble: 2 x 3
##   pred_mod1        no_heart_disease heart_disease
##   <chr>                       <int>         <int>
## 1 heart_disease                  55           130
## 2 no_heart_disease               83            35
```

```
# correctly categorized as having heart disease

heart_data %>%
  filter(disease_status == "heart_disease") %>%
  group_by(correct_prediction) %>%
  summarise(count = n()) %>%
  mutate(freq = count / sum(count))
```

```
## # A tibble: 2 x 3
##   correct_prediction count  freq
##   <chr>              <int> <dbl>
## 1 correct              130 0.788
## 2 incorrect             35 0.212
```

```
# correctly categorized as having not having heart disease

heart_data %>%
  filter(disease_status == "no_heart_disease") %>%
  group_by(correct_prediction) %>%
  summarise(count = n()) %>%
  mutate(freq = count / sum(count))
```

```
## # A tibble: 2 x 3
##   correct_prediction count  freq
##   <chr>              <int> <dbl>
## 1 correct               83 0.601
## 2 incorrect             55 0.399
```

Fine-tuning the sensitivity of the model

It is possible to fine-tune the sensitivity of the model, if we want the model to be more sensitive to detecting the presence or the absence of the event of interest, **by changing the threshold for categorizing** a case to a certain category. As mentioned above, the threshold is usually log(odds) > 0 (which is the same as Odds > 0 and probability $> 50\%$) to categorize a case as having the event of interest, but if we want our model to be more sensitive to detecting the event of interest, we can lower this threshold. Or if we want the model to be more sensitive to categorizing cases as not having the event of interest, we can increase this threshold. This is especially important when there is a big difference in the occurrence of the event of interest and it not occurring.

In our above example our model was more accurate at categorizing case that have heart disease (79%) than those that do not have heart disease (60%). If we want to **tune the model to be better able to correctly categorize those with no heart disease**, we can increase the threshold for categorizing a case as having heart disease. In the code below we increase the threshold from log(odds) > 0 to log(odds) > 0.4.

The results indicate that with this new threshold we can correctly categorize case that do not have heart disease 72% of the times, while our correct categorization of those who have heart disease is still decent: 64%. This might be beneficial, if our priority is to minimize instances when we falsely diagnose people with heart disease. We "pay for this" in accuracy in correctly detecting heart disease and overall detection accuracy, but in some cases, this might be desirable.

```
heart_data = heart_data %>%
  mutate(pred_mod1_tuned = predict(mod1)) %>%
  mutate(pred_mod1_tuned = case_when(pred_mod1_tuned <= 0.4 ~ "no_heart_disease",
                                pred_mod1_tuned > 0.4 ~ "heart_disease"))


# coding correct guesses

heart_data = heart_data %>%
  mutate(correct_prediction_tuned = case_when(pred_mod1_tuned == disease_status ~ "correct",
                                         pred_mod1_tuned != disease_status ~ "incorrect"))


# correct categorization rate overall

heart_data %>%
  group_by(correct_prediction_tuned) %>%
  summarise(count = n()) %>%
  mutate(freq = count / sum(count))
```

```
## # A tibble: 2 x 3
##   correct_prediction_tuned count  freq
##   <chr>                    <int> <dbl>
## 1 correct                    204 0.673
## 2 incorrect                   99 0.327
```

```
# crosstab of disease_status and predicted values
```

```
heart_data %>%
  group_by(disease_status, pred_mod1_tuned) %>%
  summarize(n = n()) %>%
  spread(disease_status, n)
```

```
## `summarise()` has grouped output by 'disease_status'. You can override using the `.groups` argument.
```

```
## # A tibble: 2 x 3
##   pred_mod1_tuned  no_heart_disease heart_disease
##   <chr>                       <int>         <int>
## 1 heart_disease                  39           105
## 2 no_heart_disease               99            60
```

```
# correctly categorized as having heart disease

heart_data %>%
  filter(disease_status == "heart_disease") %>%
  group_by(correct_prediction_tuned) %>%
  summarise(count = n()) %>%
  mutate(freq = count / sum(count))
```

```
## # A tibble: 2 x 3
##   correct_prediction_tuned count  freq
##   <chr>                   <int> <dbl>
## 1 correct                   105 0.636
## 2 incorrect                  60 0.364
```

```
# correctly categorized as having not having heart disease

heart_data %>%
  filter(disease_status == "no_heart_disease") %>%
  group_by(correct_prediction_tuned) %>%
  summarise(count = n()) %>%
  mutate(freq = count / sum(count))
```

```
## # A tibble: 2 x 3
##   correct_prediction_tuned count  freq
##   <chr>                   <int> <dbl>
## 1 correct                    99 0.717
## 2 incorrect                  39 0.283
```

Is the model significantly better than the null model?

We can compare the model with the predictors and the null model with a **likelihood ratio test**, to see whether our model is significantly better than the null model in predicting the outcome. We can do this by putting both the null model object and the object of the model containing the predictors in the lrtest() function. The likelihood ratio test contrasts the LogLikelihood of the two models, and produces a Chi-squared test statistic and a p-value. If this is significant, the models are significantly different from each other in prediction accuracy. **The model with the higher LogLikelihood has the better fit**. In the example below the test has a significant result, indicating that the model with max_HR in it as a predictor is significantly better than the null model.

As with the linear regression models, we can also compare the model fit of two models using the **Akaike Information Criterion (AIC)**. As before, the model with the AIC at least 2 points lower is significantly better than the other model. If the dfference in AIC is smaller than 2 points, we do not have enough evidence to reject the null hypothesis that the two models are the same in their prediction accuracy.

```
lrtest(mod_null, mod1)
```

```
## Likelihood ratio test
##
## Model 1: disease_status_numerical ~ 1
## Model 2: disease_status_numerical ~ max_HR
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   1 -208.82
## 2   2 -179.63  1 58.381   2.16e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
AIC(mod_null, mod1)
```

```
##          df      AIC
## mod_null  1 419.6381
## mod1      2 363.2569
```

Interpreting the estimates in the model

Just like in linear regression you will find model coefficients (estimates) in the model summary, and you can extract the confidence intervals corresponding to these estimates using the confint() function.

```
summary(mod1)
```

```
##
## Call:
## glm(formula = disease_status_numerical ~ max_HR, family = binomial(),
##     data = heart_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1383  -1.0780   0.6043   0.9200   2.1354
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.391452   0.987133  -6.475 9.50e-11 ***
## max_HR       0.043951   0.006531   6.729 1.71e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 417.64  on 302  degrees of freedom
## Residual deviance: 359.26  on 301  degrees of freedom
## AIC: 363.26
##
## Number of Fisher Scoring iterations: 4
```

```
confint(mod1)
```

```
## Waiting for profiling to be done...
```

```
##                   2.5 %      97.5 %
## (Intercept) -8.41257396 -4.53271457
## max_HR       0.03164292  0.05731192
```

You can interpret the **estimates** in the same way as in linear regression, with the important distinction that in logistic regression the predicted outcome of the model is not on the original scale of the outcome variable, rather, the model predicts the **log(odds) of the event of interest** (the level that is not the reference level).

In our example, the regression coefficient corresponding to max_HR is 0.032. This means, that with every step on the scale of max_HR, the log(odds) of the event of interest increases by 0.044. We often convert this to the odds scale (this can be done with the exp() function). This way we would report that the **odds ratio** corresponding to max_HR as a predictor is exp(0.044) = 1.045. This is an effect size measure that is easier to interpret: showing that a person who has 1 point higher max_HR has 1.045 times the odds of of having heart disease.

Importantly, when you are calculating the predicted outcome based on the regression equation you should always use the estimates in their original log(odds) scale, and **only convert the result to odds (and probabilities of needed) at the end**. The odds and probability cannot be directly plugged into the regression equation.

The **intercept** is interpreted similarly to that in linear regression: this is the estimated **log(odds) of the event of interest when the value of all of the predictors is zero**.

As with the linear regression, we can use the p-values corresponding to the predictors in the model summary, or the confidence intervals in the confint() function output for making an inference about whether a particular predictor has an added predictive value in the model. In other words, whether the estimate of the particular predictor is significantly different from zero.

Relative contribution of predictors to the model

We could compare the relative contribution of the different predictors in our model using the standardized beta coefficient for regular linear regression models. In logistic regression the contribution of the different predictors is compared by comparing the model fit with and without the predictors in the model one-by-one. We could do this by building these models separately by hand and comparing them to each other, but the relative influence of the different predictors also changes as the model has more or less other predictors in them. So this is a time consuming project to do by hand. Instead, we can rely on the dominanceAnalysis() function from the dominanceanalysis package, which does all this for us.

We only had a single predictor in our example above, so we build a model with some more predictors to be able to compare their influence, which we will call mod2. Below we run this analysis on this model (note that this could take a few minutes depending on how many predictors you have in the model and your computer's computational power, because the model is refit with every possible predictor combination).

After running the dominanceAnalysis() function and saving its result in an object we can extract the useful information about the predictors' relative influence with different functions and plots. For example, we can look at the influence of adding the different predictors to our models when we have different levels of model complexity using the contributionByLevel() function, and plotting the results with the which.graph = "conditional" parameter. In this analysis the levels represent how many other predictors are there already in the model, and the fit function indicate which model fit index we would like to see the influence on. Here we use the McFadden R^2 as a model fit index of our choice (fit.functions="r2.m").

We can also look at the average influence of the predictors on the model fit by using the averageContribution() function and plotting it with the which.graph ="general" parameter.

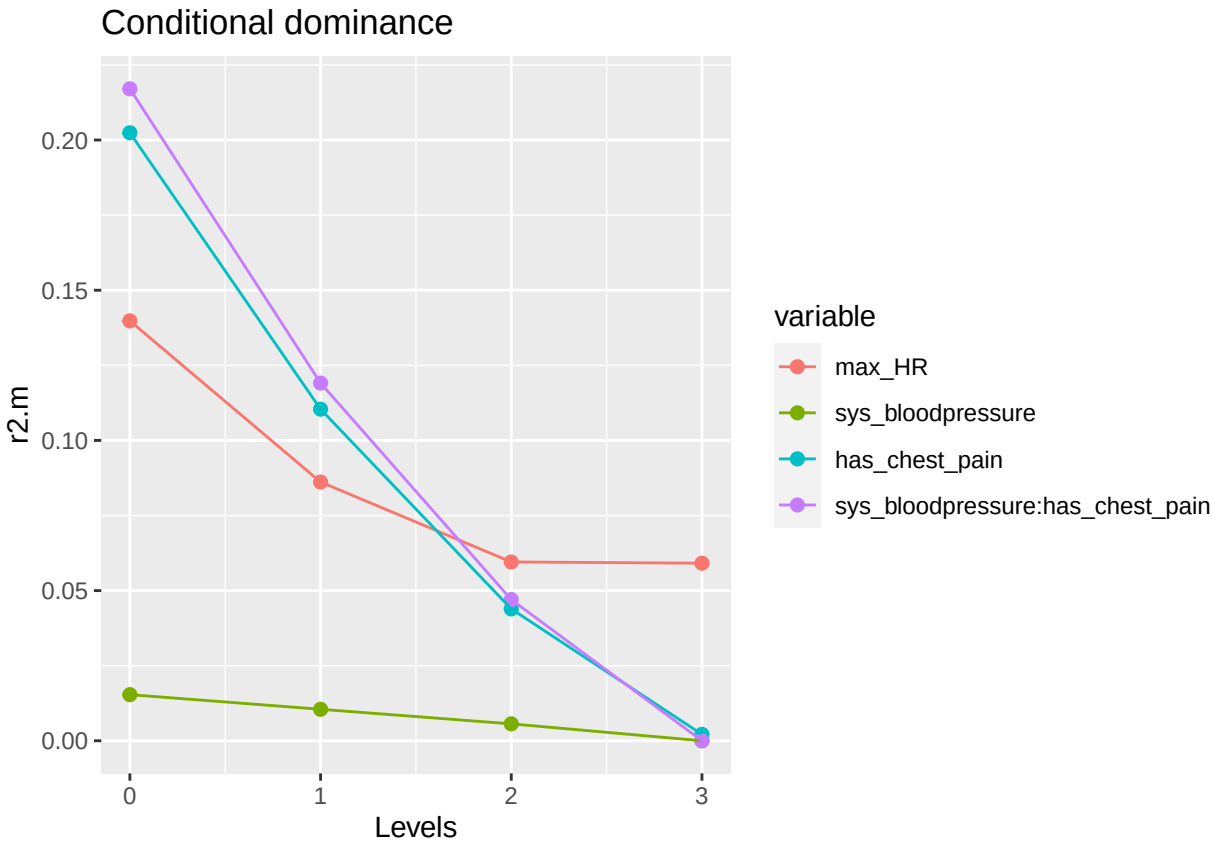
For more information on how to use and interpret dominance analysis, you can check out this guide: https://cran.r-project.org/web/packages/dominanceanalysis/vignettes/da-logistic-regression.html

```
mod2 = glm(disease_status ~ max_HR + sys_bloodpressure * has_chest_pain, family = binomial(), data = hea

dominance_mod2<-dominanceAnalysis(mod2)

contributionByLevel(dominance_mod2, fit.functions="r2.m")
```

```
##
## Contribution by level
## * Fit index:  r2.m
##  level max_HR sys_bloodpressure has_chest_pain sys_bloodpressure:has_chest_pain
##      0  0.140             0.015          0.202                            0.217
##      1  0.086             0.010          0.110                            0.119
##      2  0.060             0.006          0.044                            0.047
##      3  0.059             0.000          0.002                            0.000
```

```
plot(dominance_mod2, which.graph ="conditional",fit.function = "r2.m")
```

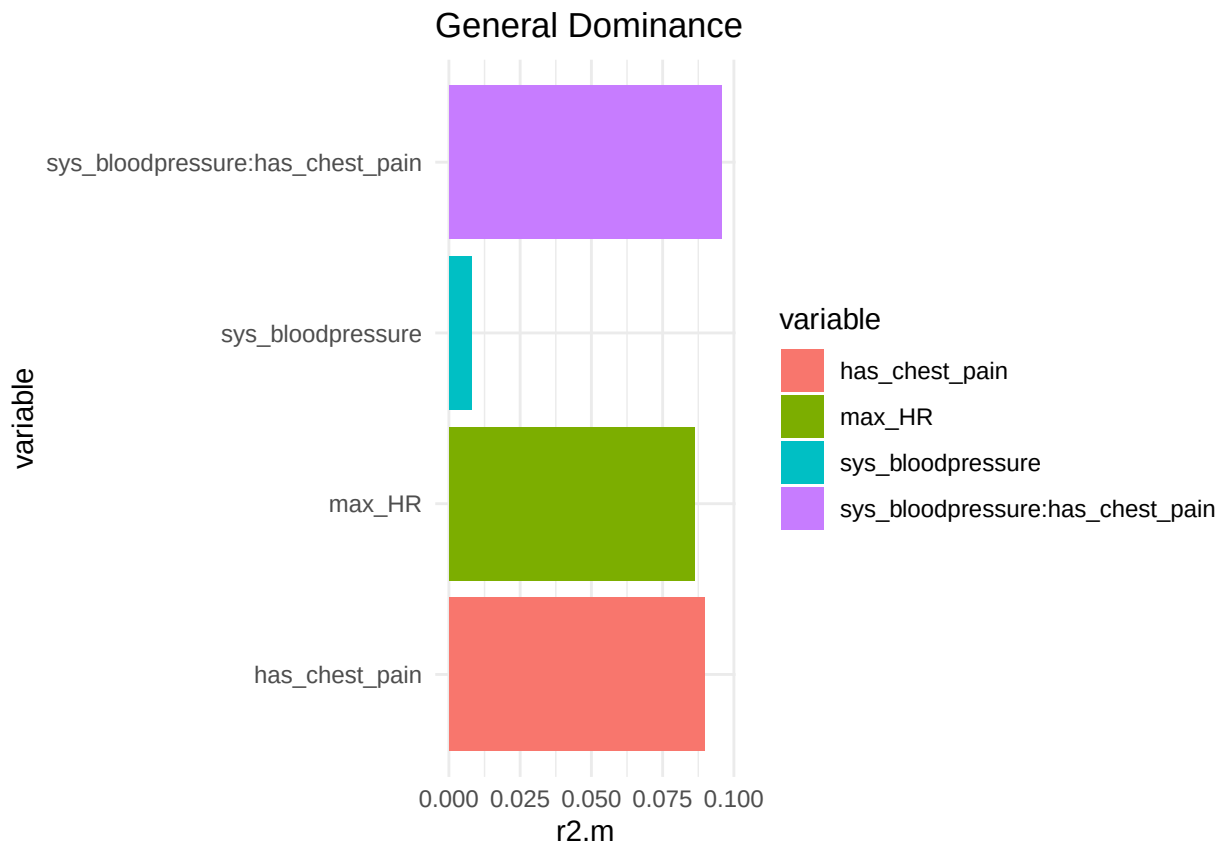
```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```




```
averageContribution(dominance_mod2,fit.functions = "r2.m")
```

```
##
## Average Contribution by predictor
##      max_HR sys_bloodpressure has_chest_pain sys_bloodpressure:has_chest_pain
## r2.m  0.086             0.008          0.09                             0.096
```

```
plot(dominance_mod2, which.graph ="general",fit.function = "r2.m") + coord_flip()
```



What to report

Now we have all the data we need for reporting our model and results.

In the **statistical analysis section** we would write:

"We have conducted a binomial logistic regression analysis where the presence of heart disease (disease_status) was the dependent variable ("no heart disease" was the reference level) and the model included one predictor: maximum heart rate achieved during exercise (max_HR)."

In the **results section** we would first report the **model fit and predictive accuracy of our model as a whole**:

"The logistic regression model with max_HR as a predictor had a significantly better model fit than the null model (Chi^2 = 58.38, df = 1, p < 0.001, AIC of the model = 363.26, -2LL of the model = 359.26, AIC of null model = 419.64, -2LL of the null model = 417.64). The model explained 14% of the variance (McFadden R^2 = 0.14). Heart disease occurred in 54.5% of the cases in our sample (165 out of 303 individuals). The final model correctly predicts the presence of heart disease in 79% of the cases, and the absence of heart disease in 60% of the cases in our sample, the overall correct prediction rate was 70%."

Second, you would usually report **the information about the regression coefficients of each predictor**, and information about their added predictive value to the model.

You can report these in a table format. The table should include the estimate and the confidence interval around the estimate, the odds ratio, the Z test statistic and the p-value for each predictor and the intercept separately. It is not common to report the relative influence of the predictors in papers, but if this is important for the report, this could be added based on the results of the dominance analysis.

_____*Practice*_____

We will continue using the Heart Disease dataset for the practice exercise. If you have not run this code already, here is some code to load the dataset and to clean/raionalize data

```
heart_data = read.csv("https://raw.githubusercontent.com/kekecsz/PSZB17-210-Data-analysis-seminar/maste

heart_data = heart_data %>%
  mutate(sex = factor(recode(sex,
                       "1" = "male",
                       "0" = "female")),
         cp = factor(recode(cp,
                            "0" = "asymptomatic",
                            "1" = "typical_angina",
                            "2" = "atypical angina",
                            "3" = "non_anginal_pain")),
         fbs = factor(recode(fbs,
                                    "1" = "true",
                                    "0" = "false")),
         disease_status = factor(disease_status)
         ) %>%
  rename(chest_pain = cp,
         sys_bloodpressure = trestbps,
         blood_sugar_over120 = fbs,
         max_HR = thalach,
         cholesterol = chol)

names(heart_data)[1] = "age"

heart_data = heart_data %>%
  mutate(has_chest_pain = factor(recode(chest_pain,
                            "asymptomatic" = "no_chest_pain",
                            "typical_angina" = "has_chest_pain",
                            "atypical angina" = "has_chest_pain",
                            "non_anginal_pain" = "has_chest_pain"),
                            levels = c("no_chest_pain", "has_chest_pain")),
         disease_status = factor(disease_status,
                                 levels = c("no_heart_disease", "heart_disease"))
         )
```

We have good reason to believe that maximum heart rate achieved during the exercise test (max_HR), resting systolic blood pressure (sys_bloodpressure), and the presence of chest pain (has_chest_pain) can help in predicting heart_disease (disease_status).

1. Build a logistic regression model where the predicted variable is presence of heart disease (disease_status) and the predictors are maximum heart rate achieved during the exercise test (max_HR), resting systolic blood pressure (sys_bloodpressure), and the presence of chest pain (has_chest_pain).
2. What is the predictive accuracy of this model? Determine how accurate the predictions are overall. Also report how accurate was the model at categorizing those cases that have heart disease and those that do not.
3. The hospital you work in would like to maximize the sensitivity for being able to detect heart disease. They would like to reach 85% accuracy at correctly detecting heart disease. Fine tune the threshold for categorizing a case as having heart disease to reach at least 85% accuracy at correctly detecting heart disease. With this adjustment, what is the accuracy for categorizing cases that actually do not have heart disease?
4. According to the logistic regression equation in your model, what is the probability(!) that a person has heart disease if this person has a max_HR = 165 and sys_bloodpressure = 190 and also has chest

pain?

5. Determine the model fit using AIC and -2LL, and the pseudo R^2 of the model.
