1. Heart disease and potential risk factors

Millions of people develop some sort of heart disease every year and heart disease is the biggest killer of both men and women in the United States and around the world. Statistical analysis has identified many risk factors associated with heart disease such as age, blood pressure, total cholesterol, diabetes, hypertension, family history of heart disease, obesity, lack of physical exercise, etc. In this notebook, we're going to run statistical tests and regression models using the Cleveland heart disease dataset to assess one particular factor -- maximum heart rate one can achieve during exercise and how it is associated with a higher likelihood of getting heart disease.

In [2]:
```
# Read datasets Cleveland_hd.csv into hd_data
hd_data <- read.csv("datasets/Cleveland_hd.csv")

# take a look at the first 5 rows of hd_data
head(hd_data, 5)
```

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	class
63	1	1	145	233	1	2	150	0	2.3	3	0	6	0
67	1	4	160	286	0	2	108	1	1.5	2	3	3	2
67	1	4	120	229	0	2	129	1	2.6	2	2	7	1
37	1	3	130	250	0	0	187	0	3.5	3	0	3	0
41	0	2	130	204	0	2	172	0	1.4	1	0	3	0

In [3]:
```
last_value <- .Last.value

# These packages need to be loaded in the first `@tests` cell.
library(testthat)
library(IRkernel.testthat)

# Then follows one or more tests of the students code.
# The @solution should pass the tests.
# The purpose of the tests is to try to catch common errors and to
# give the student a hint on how to resolve these errors.

hd_temp <- read.csv("datasets/Cleveland_hd.csv")


run_tests({

    test_that("Read in data correctly.", {
        expect_equivalent(hd_data, hd_temp,
            info = 'hd_data should contain the data in "datasets/Cleveland_hd.
csv".')
    })

    test_that("Show the top 5 rows of data correctly", {
    expect_equal(dim(last_value)[1], 5,
        info = "There should be 5 rows of data being printed out")
    })

})
```

```
<ProjectReporter>
  Inherits from: <ListReporter>
  Public:
    .context: NULL
    .end_context: function (context)
    .start_context: function (context)
    add_result: function (context, test, result)
    all_tests: environment
    cat_line: function (...)
    cat_tight: function (...)
    clone: function (deep = FALSE)
    current_expectations: environment
    current_file: some name
    current_start_time: 1.184 0.089 144.642 0.006 0.001
    dump_test: function (test)
    end_context: function (context)
    end_reporter: function ()
    end_test: function (context, test)
    get_results: function ()
    initialize: function (...)
    is_full: function ()
    out: 3
    results: environment
    rule: function (...)
    start_context: function (context)
    start_file: function (name)
    start_reporter: function ()
    start_test: function (context, test)
```

2. Converting diagnosis class into outcome variable

We noticed that the outcome variable `class` has more than two levels. According to the codebook, any non-zero values can be coded as an "event." Let's create a new variable called hd to represent a binary 1/0 outcome.

There are a few other categorical/discrete variables in the dataset. Let's also convert sex into a 'factor' for next step analysis. Otherwise, R will treat this as continuous by default.

The full data dictionary is also displayed here.

Name	Type	Description
Age	Continuous	Age Age in years
Sex	Discrete	0 = female 1 = male
Cp	Discrete	Chest pain type: 1 = typical angina, 2 = atypical angina, 3 = non-anginal pain 4 =asymptom
Trestbps	Continuous	Resting blood pressure (in mm Hg)
Chol	Continuous	Serum cholesterol in mg/dl
Fbs	Discrete	Fasting blood sugar>120 mg/dl: 1-true 0=False
Exang Continuous Maximum heart rate achieved	Discrete	Exercise induced angina: 1 = Yes 0 = No
Thalach	Continuous	Maximum heart rate achieved
Old peak ST	Continuous	Depression induced by exercise relative to rest
Slope	Discrete	The slope of the peak exercise segment : 1 = up sloping 2 = flat 3 = down sloping
Ca	Continuous	Number of major vessels colored by fluoroscopy that ranged between 0 and 3.
Thal	Discrete	3 = normal 6 = fixed defect 7= reversible defect
Class	Discrete	Diagnosis classes: 0 = No Presence 1=Least likely to have heart disease 2= >1 3= >2 4=More likely have heart disease

In [4]:
```
# load the tidyverse package
library(tidyverse)

# Use the 'mutate' function from dplyr to recode our data
hd_data %>% mutate(hd = ifelse(class > 0, 1, 0))-> hd_data

# recode sex using mutate function and save as hd_data
hd_data %>% mutate(sex = factor(sex, levels = 0:1, labels = c("Female", "Male"
)))-> hd_data
```

```
-- Attaching packages --------------------------------------- tidyverse 1.2.1
--
v ggplot2 3.0.0     v purrr   0.3.4
v tibble  3.0.3     v dplyr   1.0.2
v tidyr   1.1.2     v stringr 1.4.0
v readr   1.1.1     v forcats 0.3.0
-- Conflicts ------------------------------------------ tidyverse_conflicts()
--
x dplyr::filter()  masks stats::filter()
x purrr::is_null() masks testthat::is_null()
x dplyr::lag()     masks stats::lag()
x dplyr::matches() masks tidyr::matches(), testthat::matches()
```

In [5]:
```
# one or more tests of the students code.
# The @solution should pass the tests.
# The purpose of the tests is to try to catch common errors and to
# give the student a hint on how to resolve these errors.
last_value <- .Last.value

correct_data <- hd_temp %>% mutate(hd = ifelse(class > 0, 1, 0),sex = factor(s
ex, levels = 0:1, labels = c("Female","Male"))) -> hd_data

run_tests({

    test_that("Test that tidyverse is loaded", {
    expect_true( "package:tidyverse" %in% search(),
        info = "The tidyverse package should be loaded using library().")
    })

    test_that("hd is created correctly", {
    expect_equivalent(last_value$hd, correct_data$hd ,
        info = "hd should have value 1 if class>0.")
    })

    test_that("sex is recoded as factor correct", {
    expect_true(is.factor(last_value$sex),
        info = "Sex should be recoded into factor using factor().")
    expect_equivalent(last_value$sex, correct_data$sex ,
        info = "0=Female; 1=Male ")
    })

})
```

```
<ProjectReporter>
  Inherits from: <ListReporter>
  Public:
    .context: NULL
    .end_context: function (context)
    .start_context: function (context)
    add_result: function (context, test, result)
    all_tests: environment
    cat_line: function (...)
    cat_tight: function (...)
    clone: function (deep = FALSE)
    current_expectations: environment
    current_file: some name
    current_start_time: 2.648 0.153 149.002 0.006 0.001
    dump_test: function (test)
    end_context: function (context)
    end_reporter: function ()
    end_test: function (context, test)
    get_results: function ()
    initialize: function (...)
    is_full: function ()
    out: 3
    results: environment
    rule: function (...)
    start_context: function (context)
    start_file: function (name)
    start_reporter: function ()
    start_test: function (context, test)
```

3. Identifying important clinical variables

Now, let's use statistical tests to see which predictors are related to heart disease. We can explore the associations for each variable in the dataset. Depending on the type of the data (i.e., continuous or categorical), we use t-test or chi-squared test to calculate the p-values.

Recall, t-test is used to determine whether there is a significant difference between the means of two groups (e.g., is the mean age from group A different from the mean age from group B?). A chi-squared test for independence compares the equivalence of two proportions.

In [6]:
```
# Does sex have an effect? Sex is a binary variable in this dataset,
# so the appropriate test is chi-squared test
hd_sex <- chisq.test(hd_data$sex, hd_data$hd)

# Does age have an effect? Age is continuous, so we use t-test here
hd_age <- t.test(hd_data$age ~ hd_data$hd)

# What about thalach: maximum heart rate one can achieve during exercise?
hd_heartrate <- t.test(hd_data$thalach ~ hd_data$hd)

# Print the results to see if p<0.05.
print(hd_sex)
print(hd_age)
print(hd_heartrate)
```

```
        Pearson's Chi-squared test with Yates' continuity correction

data:  hd_data$sex and hd_data$hd
X-squared = 22.043, df = 1, p-value = 2.667e-06


        Welch Two Sample t-test

data:  hd_data$age by hd_data$hd
t = -4.0303, df = 300.93, p-value = 7.061e-05
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -6.013385 -2.067682
sample estimates:
mean in group 0 mean in group 1
       52.58537        56.62590


        Welch Two Sample t-test

data:  hd_data$thalach by hd_data$hd
t = 7.8579, df = 272.27, p-value = 9.106e-14
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 14.32900 23.90912
sample estimates:
mean in group 0 mean in group 1
        158.378         139.259
```

In [7]:
```
hd_sex_correct <- chisq.test(hd_data$sex, hd_data$hd)
hd_age_correct <- t.test(hd_data$age ~ hd_data$hd)
hd_heartrate_correct <- t.test(hd_data$thalach ~hd_data$hd)

run_tests({
    test_that("chi2 test is calculated correctly", {
    expect_equivalent(hd_sex$p, hd_sex_correct$p,
        info = "Chi-squared test should be used to test the association betwee
n hd and sex.")
    })

    test_that("t.test is calculated correctly", {
    expect_equivalent(hd_age$statistic, hd_age_correct$statistic,
        info = "t.test(y~groupvar) is the correct structure (not t.test(y,
x)).")
    })

    test_that("t.test is calculated correctly", {
    expect_equivalent(hd_heartrate$statistic, hd_heartrate_correct$statistic,
        info = "t.test(y~groupvar) should be used to test the association betw
een hd and thalach")
    })
})
```

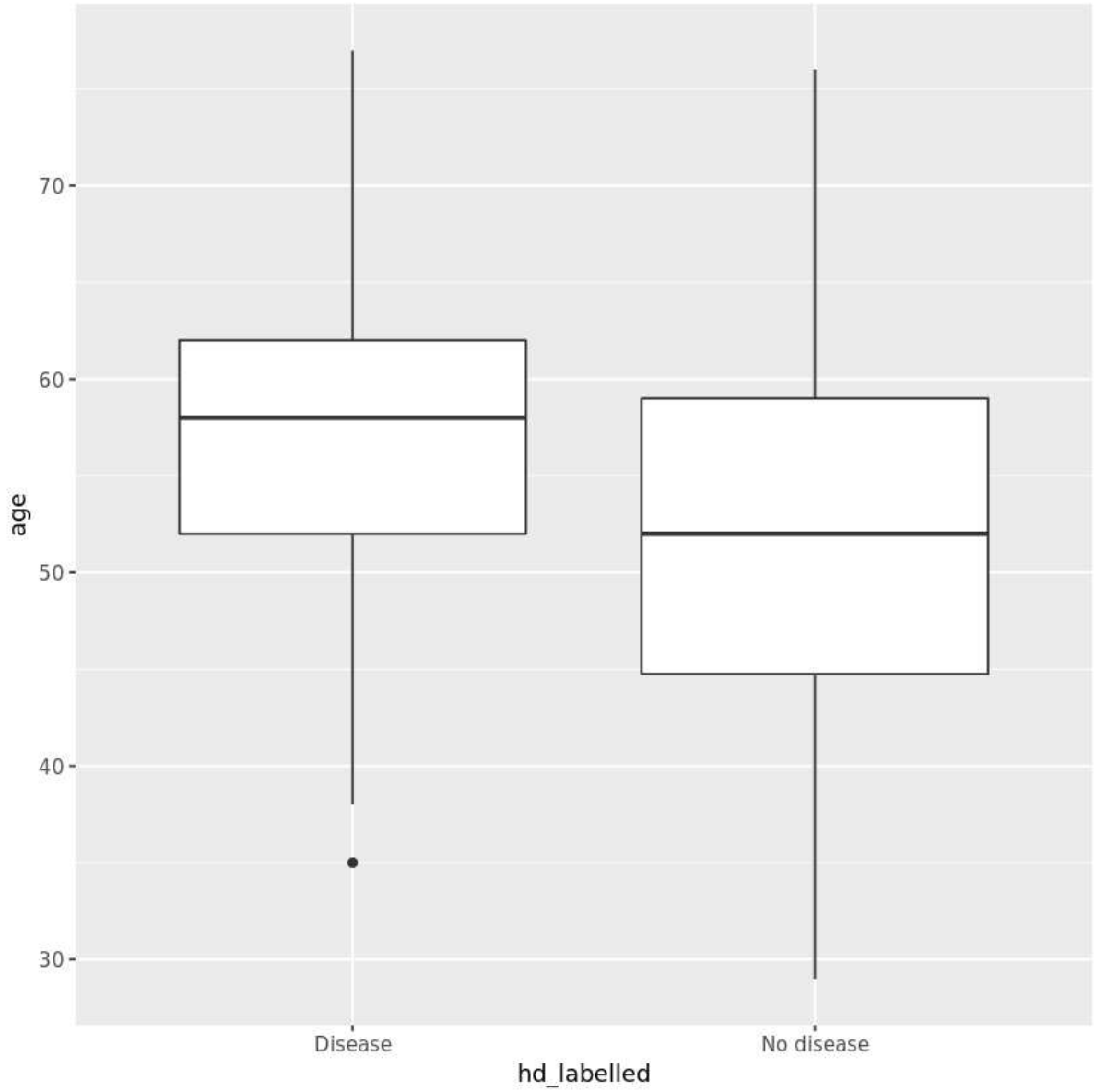
```
<ProjectReporter>
  Inherits from: <ListReporter>
  Public:
    .context: NULL
    .end_context: function (context)
    .start_context: function (context)
    add_result: function (context, test, result)
    all_tests: environment
    cat_line: function (...)
    cat_tight: function (...)
    clone: function (deep = FALSE)
    current_expectations: environment
    current_file: some name
    current_start_time: 2.743 0.157 149.115 0.006 0.001
    dump_test: function (test)
    end_context: function (context)
    end_reporter: function ()
    end_test: function (context, test)
    get_results: function ()
    initialize: function (...)
    is_full: function ()
    out: 3
    results: environment
    rule: function (...)
    start_context: function (context)
    start_file: function (name)
    start_reporter: function ()
    start_test: function (context, test)
```

4. Explore the associations graphically (i)

A good picture is worth a thousand words. In addition to p-values from statistical tests, we can plot the age, sex, and maximum heart rate distributions with respect to our outcome variable. This will give us a sense of both the direction and magnitude of the relationship.

First, let's plot age using a boxplot since it is a continuous variable.

In [8]:
```
# Recode hd to be labelled
hd_data %>% mutate(hd_labelled = ifelse(hd == 0, "No disease", "Disease")) ->
 hd_data

# age vs hd
ggplot(data = hd_data, aes(x = hd_labelled, y = age)) + geom_boxplot()
```



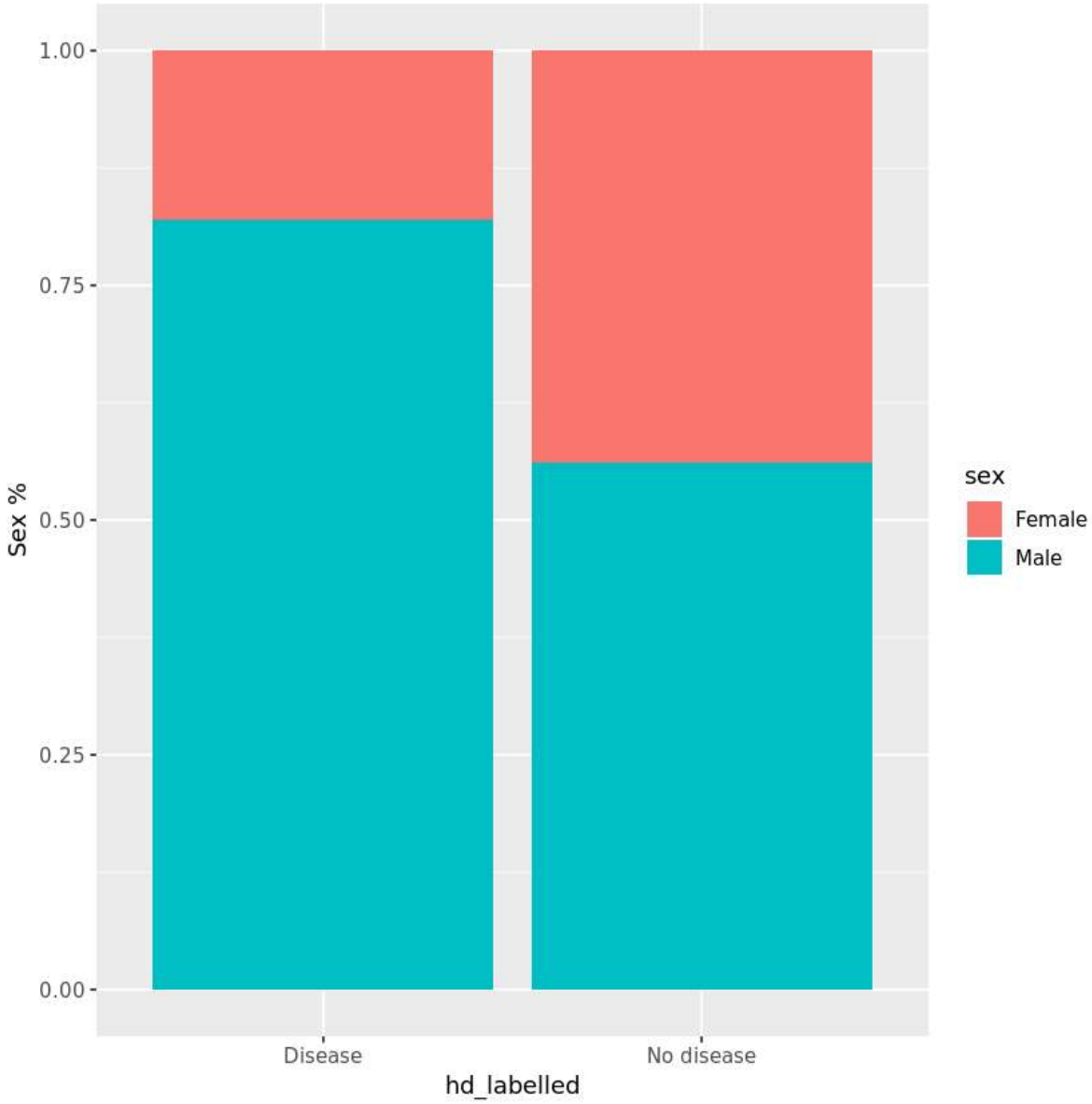
In [9]:
```
p <- last_plot()
hd_data_correct <- mutate(hd_data, hd_labelled = ifelse(hd==0, "No disease",
"Disease"))
#p_correct <- ggplot(data = hd_data_correct, aes(x = hd_labelled,y = age)) + g
eom_boxplot()

run_tests({
    test_that("correct columns are plotted", {
        mappings <- str_replace(as.character(p$mapping), "~", "")
        expect_true(all(c("hd_labelled", "age") %in% mappings),
            info = "You should plot hd_labelled on the x-axis and age on the y
-axis.")
    })
})
```

```
<ProjectReporter>
  Inherits from: <ListReporter>
  Public:
    .context: NULL
    .end_context: function (context)
    .start_context: function (context)
    add_result: function (context, test, result)
    all_tests: environment
    cat_line: function (...)
    cat_tight: function (...)
    clone: function (deep = FALSE)
    current_expectations: environment
    current_file: some name
    current_start_time: 3.974 0.177 150.821 0.006 0.001
    dump_test: function (test)
    end_context: function (context)
    end_reporter: function ()
    end_test: function (context, test)
    get_results: function ()
    initialize: function (...)
    is_full: function ()
    out: 3
    results: environment
    rule: function (...)
    start_context: function (context)
    start_file: function (name)
    start_reporter: function ()
    start_test: function (context, test)
```

5. Explore the associations graphically (ii)

Next, let's plot sex using a barplot since it is a binary variable in this dataset.

In [10]:
```
# sex vs hd
ggplot(data = hd_data, aes(x = hd_labelled, fill = sex)) + geom_bar(position =
"fill") + ylab("Sex %")
```



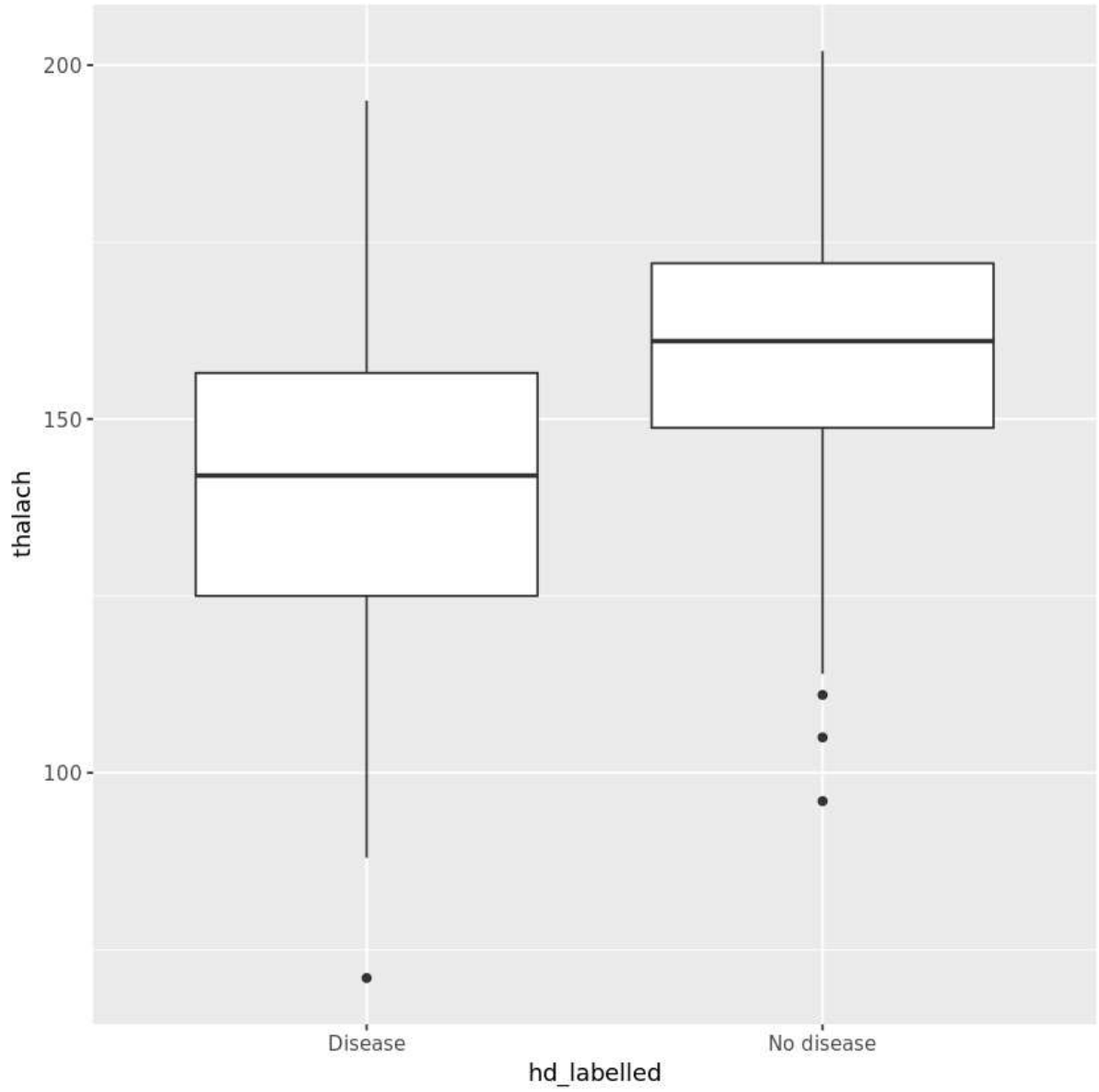
In [11]:
```
p <- last_plot()

#p_correct <- ggplot(data=hd_data, aes(x=hd_labelled, fill=sex)) + geom_bar(po
sition="fill") + ylab("Sex %")

run_tests({
    test_that("correct columns are plotted", {
        mappings <- str_replace(as.character(p$mapping), "~", "")
        expect_true(all(c("hd_labelled", "sex") %in% mappings),
            info = "You should plot hd_labelled on the x-axis and color fill t
he bar by sex.")
    })

    test_that("Y-axis is labelled correctly", {
    expect_equivalent(p$labels$y,"Sex %",
            info = "You should plot hd_labelled on the x-axis, color fill the
 bar by sex and label the y-axis as Sex %")
    })

    test_that("Position is fill", {
    expect_true(p$layers[[1]]$position$fill,
            info = "The position parameter should be 'fill'.")
    })
})
```

```
<ProjectReporter>
  Inherits from: <ListReporter>
  Public:
    .context: NULL
    .end_context: function (context)
    .start_context: function (context)
    add_result: function (context, test, result)
    all_tests: environment
    cat_line: function (...)
    cat_tight: function (...)
    clone: function (deep = FALSE)
    current_expectations: environment
    current_file: some name
    current_start_time: 5.393 0.184 152.448 0.006 0.001
    dump_test: function (test)
    end_context: function (context)
    end_reporter: function ()
    end_test: function (context, test)
    get_results: function ()
    initialize: function (...)
    is_full: function ()
    out: 3
    results: environment
    rule: function (...)
    start_context: function (context)
    start_file: function (name)
    start_reporter: function ()
    start_test: function (context, test)
```

6. Explore the associations graphically (iii)

And finally, let's plot thalach using a boxplot since it is a continuous variable.

In [12]:
```
# max heart rate vs hd
ggplot(data = hd_data, aes(x = hd_labelled, y = thalach)) + geom_boxplot()
```



In [13]:
```
p <- last_plot()
# p_correct <- ggplot(data=hd_data,aes(x=hd_labelled,y=thalach))+geom_boxplot
()

run_tests({
    test_that("correct columns are plotted", {
        mappings <- str_replace(as.character(p$mapping), "~", "")
        expect_true(all(c("hd_labelled", "thalach") %in% mappings),
            info = "You should plot hd_labelled on the x-axis and thalach on t
he y-axis.")
    })
})
```

```
<ProjectReporter>
  Inherits from: <ListReporter>
  Public:
    .context: NULL
    .end_context: function (context)
    .start_context: function (context)
    add_result: function (context, test, result)
    all_tests: environment
    cat_line: function (...)
    cat_tight: function (...)
    clone: function (deep = FALSE)
    current_expectations: environment
    current_file: some name
    current_start_time: 5.798 0.188 152.856 0.006 0.001
    dump_test: function (test)
    end_context: function (context)
    end_reporter: function ()
    end_test: function (context, test)
    get_results: function ()
    initialize: function (...)
    is_full: function ()
    out: 3
    results: environment
    rule: function (...)
    start_context: function (context)
    start_file: function (name)
    start_reporter: function ()
    start_test: function (context, test)
```

7. Putting all three variables in one model

The plots and the statistical tests both confirmed that all the three variables are highly significantly associated with our outcome (p<0.001 for all tests).

In general, we want to use multiple logistic regression when we have one binary outcome variable and two or more predicting variables. The binary variable is the dependent (Y) variable; we are studying the effect that the independent (X) variables have on the probability of obtaining a particular value of the dependent variable. For example, we might want to know the effect that maximum heart rate, age, and sex have on the probability that a person will have a heart disease in the next year. The model will also tell us what the remaining effect of maximum heart rate is after we control or adjust for the effects of the other two effectors.

The `glm()` command is designed to perform generalized linear models (regressions) on binary outcome data, count data, probability data, proportion data, and many other data types. In our case, the outcome is binary following a binomial distribution.

In [14]:
```
# use glm function from base R and specify the family argument as binomial
model <- glm(data = hd_data, hd ~ age + sex + thalach, family = "binomial" )

# extract the model summary
summary(model)
```

```
Call:
glm(formula = hd ~ age + sex + thalach, family = "binomial",
    data = hd_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2250  -0.8486  -0.4570   0.9043   2.1156

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  3.111610   1.607466   1.936   0.0529 .
age          0.031886   0.016440   1.940   0.0524 .
sexMale      1.491902   0.307193   4.857 1.19e-06 ***
thalach     -0.040541   0.007073  -5.732 9.93e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 417.98  on 302  degrees of freedom
Residual deviance: 332.85  on 299  degrees of freedom
AIC: 340.85

Number of Fisher Scoring iterations: 4
```

In [15]:
```
lastvalue <-  .Last.value
model_correct <- glm(data = hd_data, hd ~ age + sex + thalach, family = "binom
ial" )

run_tests({
    test_that("the model is a glm object", {
    expect_is(model, "glm",
        info = "The model should be a glm object.")
    })

    test_that("the model family is binomial", {
    expect_equivalent(model$family$family, "binomial",
        info = "The model family should be binomial.")
    })


    test_that("the model summary is printed correctly", {
    expect_equivalent(lastvalue, summary(model_correct),
        info = "The model summary should be printed using the summary() functi
on.")
    })
})
```

```
<ProjectReporter>
  Inherits from: <ListReporter>
  Public:
    .context: NULL
    .end_context: function (context)
    .start_context: function (context)
    add_result: function (context, test, result)
    all_tests: environment
    cat_line: function (...)
    cat_tight: function (...)
    clone: function (deep = FALSE)
    current_expectations: environment
    current_file: some name
    current_start_time: 5.861 0.192 153.138 0.006 0.001
    dump_test: function (test)
    end_context: function (context)
    end_reporter: function ()
    end_test: function (context, test)
    get_results: function ()
    initialize: function (...)
    is_full: function ()
    out: 3
    results: environment
    rule: function (...)
    start_context: function (context)
    start_file: function (name)
    start_reporter: function ()
    start_test: function (context, test)
```

8. Extracting useful information from the model output

It's common practice in medical research to report Odds Ratio (OR) to quantify how strongly the presence or absence of property A is associated with the presence or absence of the outcome. When the OR is greater than 1, we say A is positively associated with outcome B (increases the Odds of having B). Otherwise, we say A is negatively associated with B (decreases the Odds of having B).

The raw glm coefficient table (the 'estimate' column in the printed output) in R represents the log(Odds Ratios) of the outcome. Therefore, we need to convert the values to the original OR scale and calculate the corresponding 95% Confidence Interval (CI) of the estimated Odds Ratios when reporting results from a logistic regression.

In [16]:
```
# load the broom package
library(broom)

# tidy up the coefficient table
tidy_m <- tidy(model)
tidy_m

# calculate OR
tidy_m$OR <- exp(tidy_m$estimate)

# calculate 95% CI and save as lower CI and upper CI
tidy_m$lower_CI <- exp(tidy_m$estimate - 1.96 * tidy_m$std.error)
tidy_m$upper_CI <- exp(tidy_m$estimate + 1.96 * tidy_m$std.error)

# display the updated coefficient table
tidy_m
```

term	estimate	std.error	statistic	p.value
(Intercept)	3.11161046	1.607466382	1.935724	5.290157e-02
age	0.03188572	0.016439824	1.939541	5.243548e-02
sexMale	1.49190218	0.307192627	4.856569	1.194372e-06
thalach	-0.04054143	0.007072952	-5.731897	9.931367e-09

term	estimate	std.error	statistic	p.value	OR	lower_CI	upper
(Intercept)	3.11161046	1.607466382	1.935724	5.290157e-02	22.4571817	0.9617280	524.3
age	0.03188572	0.016439824	1.939541	5.243548e-02	1.0323995	0.9996637	1.066
sexMale	1.49190218	0.307192627	4.856569	1.194372e-06	4.4455437	2.4346539	8.117
thalach	-0.04054143	0.007072952	-5.731897	9.931367e-09	0.9602694	0.9470490	0.973

In [17]:
```
tidy_m_correct <- tidy(model)
tidy_m_correct$OR <- exp(tidy_m_correct$estimate)
tidy_m_correct$upper_CI <- exp(tidy_m_correct$estimate + 1.96 * tidy_m_correct
$std.error)

run_tests({
test_that("Test that broom is loaded", {
    expect_true( "package:broom" %in% search(),
        info = "The broom package should be loaded using library().")
    })

    test_that("OR is calculated correctly.", {
        expect_equivalent(tidy_m$OR, tidy_m_correct$OR,
            info = 'OR=exp(model_table$estimate).')
    })

    test_that("upper CI is calculated correctly.", {
        expect_equivalent(tidy_m$upper_CI, tidy_m_correct$upper_CI,
            info = 'upper_CI=exp(model_table$estimate+1.96*model_table$std.err
or).')
    })
})
```

```
<ProjectReporter>
  Inherits from: <ListReporter>
  Public:
    .context: NULL
    .end_context: function (context)
    .start_context: function (context)
    add_result: function (context, test, result)
    all_tests: environment
    cat_line: function (...)
    cat_tight: function (...)
    clone: function (deep = FALSE)
    current_expectations: environment
    current_file: some name
    current_start_time: 6.051 0.192 153.377 0.006 0.001
    dump_test: function (test)
    end_context: function (context)
    end_reporter: function ()
    end_test: function (context, test)
    get_results: function ()
    initialize: function (...)
    is_full: function ()
    out: 3
    results: environment
    rule: function (...)
    start_context: function (context)
    start_file: function (name)
    start_reporter: function ()
    start_test: function (context, test)
```

9. Predicted probabilities from our model

So far, we have built a logistic regression model and examined the model coefficients/ORs. We may wonder how can we use this model we developed to predict a person's likelihood of having heart disease given his/her age, sex, and maximum heart rate. Furthermore, we'd like to translate the predicted probability into a decision rule for clinical use by defining a cutoff value on the probability scale. In practice, when an individual comes in for a health check-up, the doctor would like to know the predicted probability of heart disease, for specific values of the predictors: a 45-year-old female with a max heart rate of 150. To do that, we create a data frame called newdata, in which we include the desired values for our prediction.

In [18]:
```
# get the predicted probability in our dataset using the predict() function
# We include the argument type=”response” in order to get our prediction.
pred_prob <- predict(model, hd_data, type="response")

# create a decision rule using probability 0.5 as cutoff and save the predicte
d decision into the main data frame
hd_data$pred_hd <- ifelse(pred_prob >= 0.5, 1, 0)

# create a newdata data frame to save a new case information
newdata <- data.frame(age=45, sex="Female", thalach=150)

# predict probability for this new case and print out the predicted value
p_new <- predict(model, newdata, type="response")
p_new
```

1: 0.177300249223782

In [19]:
```
pred_prob_correct <- predict(model, hd_data, type="response")
hd_data$pred_hd_correct <- ifelse(pred_prob_correct >= 0.5, 1, 0)
p_new_correct <- predict(model, newdata, type="response")

run_tests({
test_that("Test that pred_prob is calculated correctly", {
    expect_equivalent(pred_prob, pred_prob_correct,
        info = "The predict() function should be used on the model object.")
    })

    test_that("Predicted hd status is calculated correctly.", {
        expect_equivalent(hd_data$pred_hd, hd_data$pred_hd_correct,
            info = 'pred_prob>0.5 should be coded as 1, and 0 otherwise.')
    })

    test_that("The new person's HD probability is calculated correctly.", {
        expect_equivalent(p_new, p_new_correct,
            info = 'Apply the predict() function on the newdata.')
    })
})
```

```
<ProjectReporter>
  Inherits from: <ListReporter>
  Public:
    .context: NULL
    .end_context: function (context)
    .start_context: function (context)
    add_result: function (context, test, result)
    all_tests: environment
    cat_line: function (...)
    cat_tight: function (...)
    clone: function (deep = FALSE)
    current_expectations: environment
    current_file: some name
    current_start_time: 6.189 0.192 153.526 0.006 0.001
    dump_test: function (test)
    end_context: function (context)
    end_reporter: function ()
    end_test: function (context, test)
    get_results: function ()
    initialize: function (...)
    is_full: function ()
    out: 3
    results: environment
    rule: function (...)
    start_context: function (context)
    start_file: function (name)
    start_reporter: function ()
    start_test: function (context, test)
```

10. Model performance metrics

Are the predictions accurate? How well does the model fit our data? We are going to use some common metrics to evaluate the model performance. The most straightforward one is Accuracy, which is the proportion of the total number of predictions that were correct. On the other hand, we can calculate the classification error rate using 1-accuracy. However, accuracy can be misleading when the response is rare (i.e., imbalanced response). Another popular metric, Area Under the ROC curve (AUC), has the advantage that it's independent of the change in the proportion of responders. AUC ranges from 0 to 1. The closer it gets to 1 the better the model performance. Lastly, a confusion matrix is an N X N matrix, where N is the level of outcome. For the problem at hand, we have N=2, and hence we get a 2 X 2 matrix. It cross-tabulates the predicted outcome levels against the true outcome levels.

After these metrics are calculated, we'll see (from the logistic regression OR table) that older age, being male and having a lower max heart rate are all risk factors for heart disease. We can also apply our model to predict the probability of having heart disease. For a 45 years old female who has a max heart rate of 150, our model generated a heart disease probability of 0.177 indicating low risk of heart disease. Although our model has an overall accuracy of 0.71, there are cases that were misclassified as shown in the confusion matrix. One way to improve our current model is to include other relevant predictors from the dataset into our model, but that's a task for another day!

In [20]:
```
# load Metrics package
library(Metrics)

# calculate auc, accuracy, clasification error
auc <- auc(hd_data$hd, hd_data$pred_hd)
accuracy <- accuracy(hd_data$hd, hd_data$pred_hd)
classification_error <- ce(hd_data$hd, hd_data$pred_hd)

# print out the metrics on to screen
print(paste("AUC=", auc))
print(paste("Accuracy=", accuracy))
print(paste("Classification Error=", classification_error))

# confusion matrix
table(hd_data$hd, hd_data$pred_hd, dnn=c("True Status", "Predicted Status")) # confusion matrix
```

```
[1] "AUC= 0.706483593612915"
[1] "Accuracy= 0.70957095709571"
[1] "Classification Error= 0.29042904290429"

           Predicted Status
True Status   0   1
          0 122  42
          1  46  93
```

In [21]:
```
lastvalue <- .Last.value #table output
auc_correct <- auc(hd_data$hd, hd_data$pred_hd)
accuracy_correct <- accuracy(hd_data$hd, hd_data$pred_hd)
classification_error_correct <- ce(hd_data$hd, hd_data$pred_hd)
confusionmatrix <- table(hd_data$hd, hd_data$pred_hd, dnn=c("True Status","Pre
dicted Status"))

run_tests({
    test_that("AUC is calculated correctly", {
    expect_equal(auc,auc_correct,
        info = "The auc() function should be used.")
    })

    test_that("Accuracy is calculated correctly.", {
        expect_equal(accuracy, accuracy_correct,
            info = 'The accuracy() function should be used.')
    })

    test_that("Classification error is calculated correctly", {
    expect_equal(classification_error,classification_error_correct,
        info = "The ce() function should be used.")
    })

    test_that("Confusion matrix is calculated correctly.", {
        expect_equivalent(lastvalue, confusionmatrix,
            info = 'The first argument should be the true status and the secon
d should be your predicted status.')
    })
})
```

```
<ProjectReporter>
  Inherits from: <ListReporter>
  Public:
    .context: NULL
    .end_context: function (context)
    .start_context: function (context)
    add_result: function (context, test, result)
    all_tests: environment
    cat_line: function (...)
    cat_tight: function (...)
    clone: function (deep = FALSE)
    current_expectations: environment
    current_file: some name
    current_start_time: 6.299 0.196 153.672 0.006 0.001
    dump_test: function (test)
    end_context: function (context)
    end_reporter: function ()
    end_test: function (context, test)
    get_results: function ()
    initialize: function (...)
    is_full: function ()
    out: 3
    results: environment
    rule: function (...)
    start_context: function (context)
    start_file: function (name)
    start_reporter: function ()
    start_test: function (context, test)
```