Heart Data Analyses

Andrew Wang

9/13/2020

An overview of heart disease

Heart disease is used to describe a range of disease that impact the heart. For example, diseases like coronary artery disease, arrhythmias, and congenital heart defects all fall under heart disease. While symptoms vary based on the specific disease, some general symptoms include shortness of breath, fatigue, and chest pain. While causes also vary, most of the main causes circulate around lifestyle choices. Things like smoking, having an unhealthy diet, drug abuse, excess caffine, and even stress can increase you chance of developing a heart disease.

How does the heart work?

The heart is an organ at the center of your circulatory system, the body system which carries nutrients throughout the body. Blood carries important nutrients to all parts of the body, which is necessary for muscles and organs to function properly. The heart is divided into two parts, the left side and the right side. The right side of the heart receives oxygen-poor (lacking oxygen) blood from your veins and moves it into your lungs. In the lungs, this blood drops off Carbon Dioxide (CO_2), and picks up Oxygen (O_2). Then, this blood moves into the left side of the heart, where it is pumped through arteries to distibute nutrients to the rest of your body.

Blood flow through the heart

There are four separate chambers of the heart, two on each side. Each side has both a ventricle and an atrium. First, oxygen-poor blood goes into the right atrium of the heart. The blood is then pumped into right ventricle through the tricuspid valve. The valves are important as they prevent the backflow of blood into previous sections of the heart, similar to an assembly line at a factory (in which products only move forward). The blood in the right ventricle is then moved into the lungs through the pulmonary valve, and it drops off the CO_2 and picks up O_2 (as mentioned earlier). After this, the blood moves into the left atrium and is pumped into the left atrium through the mitral valve. Finally, the left ventricle pumps the blood through the aortic valve back into the body.

Focus: Angina Pectoris

Angina Pectoris, also known as ischemic chest pain, is chest pain caused by reduced blood flow to the heart. It is the medical term for chest pain from coronary heart disease (a heart disease caused by buildup of plaque in arteries). Because of this reduced blood flow, the heart muscles are not getting as much blood as they need. Although it is relatively common, it can be hard to distinguish from other kinds of chest pain, like heartburn or indigestion. While is it not as serious as many of the other heart diseases, its an important sign of increased risk for heart attack or stroke. Risk factors are similar to those of many other heart diseases, including smoking, high blood pressure, and obesity.

Symptoms and treatment

Symptoms of Angina Pectoris include tightness, heaviness, or pressure in the chest. However, pain can also be experienced in the neck, arms, or shoulders. The severity of Angina Pectoris can largely vary, and treatment will revolve around around its severity. Treatment can range from lifestyle changes to angioplasty, a surgery done to widen arteries for increased blood flow, along with everything in-between. For example, medications like Nitrates, Aspirin, or other clot-preventing drugs can be used to widen your arteries.

Types of Angina

There are multiple types of Angina a person may experience, which is also largely dependent on the severity of the condition. Below, we summarize the two main types of Angina, stable and unstable.

Stable Angina

Stable Angina is the most common form of Angina, and occurs when you exert yourself. This kind of Angina will go away after rest, usually lasting less than 5 minutes. Stable Angina will occur when your heart works harder than normal, as your need to pump more blood through your body, but your circulatory system is not used to pumping the increased quantity of blood, leading to chest pain. For example, you may get stable Angina during or after intense exercise, as your heart needs to work harder. Stable Angina is usually predictable.

Unstable Angina

Unstable Angina is a less common and more deadly form of Angina. Unstable Angina usually occurs during rest, and should be treated as an emergency (due to its instability and unpredictability). While the causes for unstable Angina are the same as those for stable Angina, unstable Angina is more deadly because of its occurrence during rest and its severity. It is important to note that stable Angina can become unstable, if it lasts for an unnaturally long time or worsens over time.

Data cleanup and setup

In this section, we are going over various functions and methods for cleaning data in R, which will allow us to better perform EDA (exploratory data analysis) on it. First, we need to clean up and set up the R environment with the working directory, and load up another R script for reference later.

```
# clean up and setup
rm(list=ls()) # clean up any old stuff in R
setwd("C:/Users/hyper/OneDrive/Desktop/Desktop Folders/Programming/R/Assignments/Week 3") # go to this folder
#load up myfunctions.R
source("C:/Users/hyper/OneDrive/Desktop/Desktop Folders/Programming/R/myfunctions.R")
```

Note that your directory structure will be different than mine, so don't directly copy the code here.

Load the library

Next, we want to load the library we'll be using to clean and prepare the data. The library name is `dplyr`. However, you will need to install the library first with the command `install.packages("dplyr")`. After this, you can import the library with `library(dplyr)`.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Importing the csv file

After installing the library, it is time to import the csv file. First, make sure your csv file is in the same location as your specified working directory (defined above). This will make sure R knows where to go to find the csv file. After this, we run a few inital viewing functions to look at basic attributes of the dataframe, such as names and structure (str).

```
#input csv file
heartData <- read.csv("dirtyheart.csv")
View(heartData)
names(heartData)
```

```
##  [1] "age"      "sex"      "cp"       "trestbps" "chol"     "fbs"
##  [7] "restecg"  "thalach"  "exang"    "oldpeak"  "slope"    "ca"
## [13] "thal"     "target"
```

```
str(heartData)
```

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age     : int  63 37 41 NA 57 57 56 44 52 57 ...
##  $ sex     : int  1 1 0 1 0 1 0 1 1 1 ...
##  $ cp      : int  1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps: int  145 130 130 120 120 140 NA 120 172 150 ...
##  $ chol    : int  233 250 204 236 354 192 294 263 199 168 ...
##  $ fbs     : int  1 0 0 0 0 0 0 0 1 0 ...
##  $ restecg : int  0 1 0 1 1 1 0 1 1 1 ...
##  $ thalach : int  150 187 172 178 163 148 153 173 162 174 ...
##  $ exang   : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ oldpeak : num  2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
##  $ slope   : int  0 0 2 2 2 1 1 2 2 2 ...
##  $ ca      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ thal    : int  6 3 7 3 3 3 3 3 7 7 ...
##  $ target  : int  1 1 1 1 1 1 1 1 1 1 ...
```

Renaming the columns

To start off with, we want to rename the columns headers to words that actually represent what the column stands for, and give insight into the purpose of the values in the column. We do that below by using the `rename` function of `dplyr`. After this, we print out the names again to make sure the names have changed.

```
heartData <- rename(heartData , gender = sex, chestPain = cp, RBP = trestbps, Cholesterol = chol, FBS = fbs, restECG = reste
cg, maxHR = thalach, exAng = exang, depressionST = oldpeak, thal. = thal, fluoroMV = ca, heartAttack = target)
names(heartData)
```

```
##  [1] "age"          "gender"       "chestPain"    "RBP"          "Cholesterol"
##  [6] "FBS"          "restECG"      "maxHR"        "exAng"        "depressionST"
## [11] "slope"        "fluoroMV"     "thal."        "heartAttack"
```

Fixing missing data values

One of the most, if not the most, important steps in data cleaning and preparation is the make sure you fix the missing values. While this can be done through deletion, it will remove lots of data and is generally not considered good practice. Instead, what we can do is make the missing data points the average of all other nonmissing data points. By doing this, we can keep all data rows (observations), while also keep any analysis we do mostly consistent. Below is a huge block of code which essentially goes through each column to fix the missing data. Each line essentially follows the same format, which is:

`heartData$columnName <- ifelse(is.na(heartData$columnName) | heartData$columnName == 0, round(mean(heartData$columnName, na.rm = TRUE,0),he`

What is this of code does is it goes through each value of the specified column (given by `columnName`), checking if it is empty or equal to 0. After this if it is empty it will be replaced with a rounded mean value (as most of th existing data is as an integer, we want to keep the format consistent). However, there are a few exceptions to this rule. Firstly, columns in which the numbers actually stand for words (gender, heartAttack, etc) we dont' want the `heartData$columnName == 0`, as having 0 values is needed for accurate data transcription. After all of the data editing, we check to see if any empty values remain.

```
#fix missing data
# you don't want the or == 0 for the bool columns where zero stands for something
heartData$age <- ifelse(is.na(heartData$age) | heartData$age == 0 ,round(mean(heartData$age,na.rm=TRUE),0),heartData$age)
heartData$gender <- ifelse(is.na(heartData$gender),round(mean(heartData$gender,na.rm=TRUE),0),heartData$gender)
heartData$chestPain <- ifelse(is.na(heartData$chestPain) | heartData$chestPain == 0 ,round(mean(heartData$chestPain,na.rm=TR
UE),0),heartData$chestPain)
heartData$RBP <- ifelse(is.na(heartData$RBP) | heartData$RBP == 0 ,round(mean(heartData$RBP,na.rm=TRUE),0),heartData$RBP)
heartData$Cholesterol <- ifelse(is.na(heartData$Cholesterol) | heartData$Cholesterol == 0 ,round(mean(heartData$Cholesterol,
na.rm=TRUE),0),heartData$Cholesterol)
heartData$FBS <- ifelse(is.na(heartData$FBS),round(mean(heartData$FBS,na.rm=TRUE),0),heartData$FBS)
heartData$restECG <- ifelse(is.na(heartData$restECG),round(mean(heartData$restECG,na.rm=TRUE),0),heartData$restECG)
heartData$maxHR <- ifelse(is.na(heartData$maxHR) | heartData$maxHR == 0 ,round(mean(heartData$maxHR,na.rm=TRUE),0),heartData
$maxHR)
heartData$exAng <- ifelse(is.na(heartData$exAng),round(mean(heartData$exAng,na.rm=TRUE),0),heartData$exAng)
heartData$depressionST <- ifelse(is.na(heartData$depressionST) | heartData$depressionST == 0 ,round(mean(heartData$depressio
nST,na.rm=TRUE),1),heartData$depressionST)
heartData$slope <- ifelse(is.na(heartData$slope) | heartData$slope == 0 ,round(mean(heartData$slope,na.rm=TRUE),0),heartData
$slope)
heartData$fluoroMV <- ifelse(is.na(heartData$fluoroMV) | heartData$fluoroMV == 0 ,round(mean(heartData$fluoroMV,na.rm=TRUE),
0),heartData$fluoroMV)
heartData$thal. <- ifelse(is.na(heartData$thal.),3,heartData$thal.)
heartData$heartAttack <- ifelse(is.na(heartData$heartAttack),mean(heartData$heartAttack,na.rm=TRUE),heartData$heartAttack)
print(sum(is.na(heartData)))
```

```
## [1] 0
```

Renaming column values to words

As mentioned in the previous step, we have certain columns in which the values stand for words. To convert these numbers to words, we use the `factor()` function to transcribe from numbers to words. We want to do this to make the data more readable, its easier to look through words with a direct meaning than numbers with an implied meaning. The agruements presented to the `factor()` function are as follows: the name of the column being changed, the current existing values in the column, and the new intended values for the column.

```
#updates numeric values with actual their word representations
heartData$heartAttack = factor(heartData$heartAttack, levels = c(0,1), labels = c("No","Yes"))
heartData$gender = factor(heartData$gender, levels = c(0,1), labels = c("Female","Male"))
heartData$chestPain = factor(heartData$chestPain, levels = c(1,2,3,4), labels = c("Typical","Atypical","Non-Anginal","Asympt
omatic"))
heartData$restECG = factor(heartData$restECG, levels = c(0,1,2), labels = c("Normal","Abnormal","LVH"))
heartData$exAng = factor(heartData$exAng, levels = c(0,1), labels = c("No","Yes"))
heartData$slope = factor(heartData$slope, levels = c(1,2,3), labels = c("Up","Flat","Down"))
heartData$FBS = factor(heartData$FBS, levels = c(0,1), labels = c("False","True"))
heartData$thal. = factor(heartData$thal., levels = c(3,6,7), labels = c("Normal","Fixed","Reversible"))
```

Arrange the data

We next want to sort the data by age, where we can use the `arrange` function within the `dplyr` package. The following code snippet is rather simple, as it just sorts the existing dataframe by age.

```
#arrange dataset by age
heartData <- arrange(heartData, age)
```

Numeric data subset

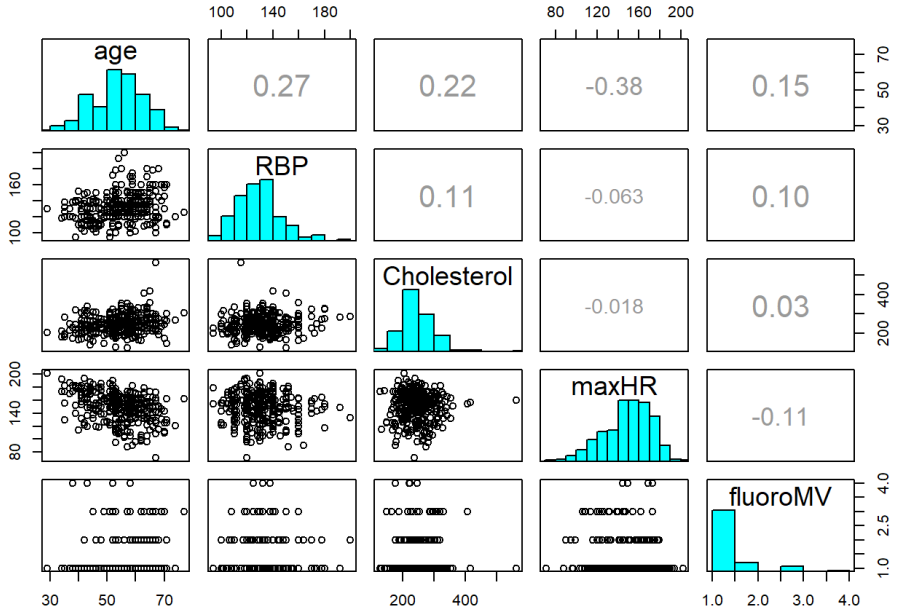
Now that we are done cleaning our data, we can start creating subsets for analysis. Below, I create three subsets for the numeric data. The first of which is normal numeric data, and the other two are specified by gender. We ues the `filter` function to sort the data by gender for the `numericsMale` and `numericsFemale` subsets. However, we have to keep gender in the subset so we can filter based on the gender. We delete this column afterwards.

```
#select numeric data, left out a few values because the margins were otherwise too large
numerics <- select(heartData, age, gender, RBP, Cholesterol, maxHR, fluoroMV)
# make subsets based on filtering gender
numericsMale <- filter(numerics, gender == "Male")
numericsFemale <- filter(numerics, gender == "Female")
#we then nullify gender because we needed it for the filter variable
numerics$gender <- NULL
numericsMale$gender <- NULL
numericsFemale$gender <- NULL
```

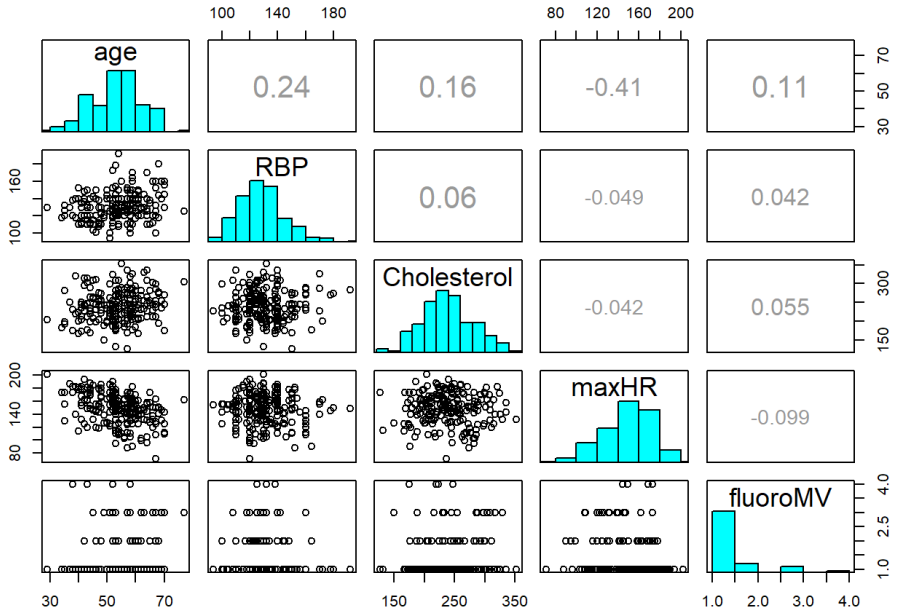
Pair plot of numeric data

After making subsets for numeric data, we want to create a pair plot with them so we can look at the correlation between various numeric factors, along with an added gender component.

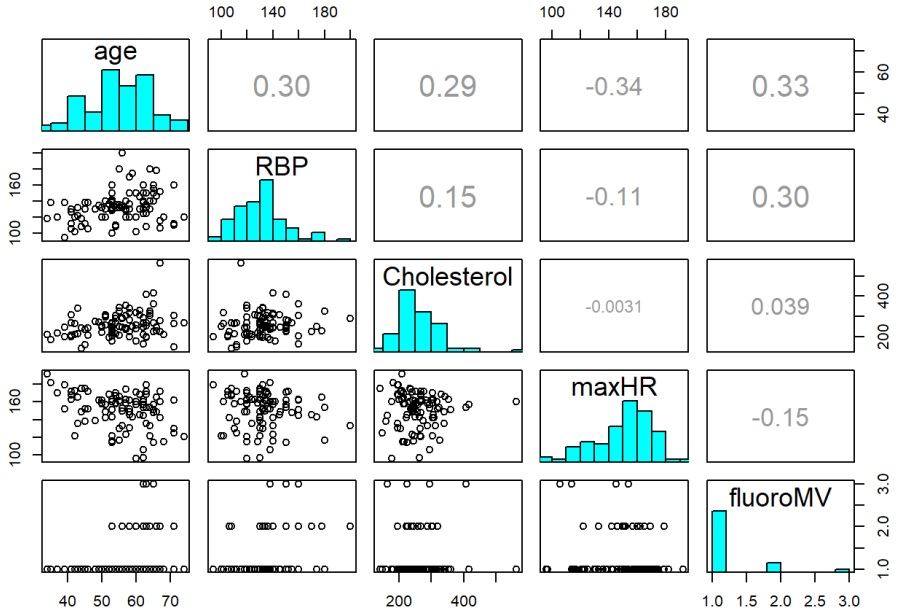
```
pairs(numerics, upper.panel = panel.cor, diag.panel = panel.hist)
```




```
pairs(numericsMale, upper.panel = panel.cor, diag.panel = panel.hist)
```




```
pairs(numericsFemale, upper.panel = panel.cor, diag.panel = panel.hist)
```



From these plots we can see that the data is the data for numerics is not very correlated on any dimension. However, filtering the male and female data, we can see that the female data is considerably more correlated than the male data in nearly all values.

Group_by for a chest pain summary

Here, we want to make a new subset to hold some mean values. We want to use something called a pipe, which is the symbol `%>%`. The pipe symbol is basically a 'siphon' for data, in which data is passed through the pipe to various steps. In the code below, we make three subsets which are divided by the type of chest pain. First, we refer to the `heartData` dataframe, and use the `group_by` function to sort it by the types of chestPain. Finally, we use the `summarize()` function to set this newly grouped data to something. Our new subsetsfind the mean values for resting blood pressure, cholesteral, and max heart rate regarding various types of chest pain. Furthermore, we again seperate them by gender.

```
#group_by + summerize
chestPainSummary <- heartData %>% group_by(chestPain) %>% summarize(
  meanRBP = mean(RBP),
  meanCholesteral = mean(Cholesterol),
  meanMaxHR = mean(maxHR),
)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
chestPainSummaryMale <- filter(heartData, gender == "Male") %>% group_by(chestPain) %>% summarize(
  meanRBP = mean(RBP),
  meanCholesteral = mean(Cholesterol),
  meanMaxHR = mean(maxHR),
)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
chestPainSummaryFemale <- filter(heartData, gender == "Female") %>% group_by(chestPain) %>% summarize(
  meanRBP = mean(RBP),
  meanCholesteral = mean(Cholesterol),
  meanMaxHR = mean(maxHR),
)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

Output the subsets

Here, we simply print out the subsets we just made with the `head()` command. The `head()` command prints out the first six rows of a dataframe, and since none of our subsets are greater than 6 rows we can use it here.

```
head(chestPainSummary)
```

```
## # A tibble: 4 x 4
##   chestPain    meanRBP meanCholesteral meanMaxHR
##   <fct>          <dbl>           <dbl>     <dbl>
## 1 Typical         130.            255.      148.
## 2 Atypical        129.            244.      149.
## 3 Non-Anginal     132.            246.      151.
## 4 Asymptomatic    133.            244.      149.
```

```
head(chestPainSummaryMale)
```

```
## # A tibble: 4 x 4
##   chestPain    meanRBP meanCholesteral meanMaxHR
##   <fct>          <dbl>           <dbl>     <dbl>
## 1 Typical         133.            246.      146
## 2 Atypical        129.            240.      148.
## 3 Non-Anginal     132.            234.      149.
## 4 Asymptomatic    131.            241.      150.
```

```
head(chestPainSummaryFemale)
```

```
## # A tibble: 4 x 4
##   chestPain    meanRBP meanCholesteral meanMaxHR
##   <fct>          <dbl>           <dbl>     <dbl>
## 1 Typical         124.            275       152.
## 2 Atypical        128.            251.      151.
## 3 Non-Anginal     132.            271.      155.
## 4 Asymptomatic    138.            251.      148.
```

A subset for numeric deviation

In the following code, we use the `mutate()` command to create a new data subset off of the existing `numerics` dataframe. The `mutate()` command allows you to add new columns to the end o the dataset, and here we use it to add columns calcuating the difference of each value from the mean of the column. The `mutate()` function is very versitle and allows you to do a variety of things around adding new columns. After making the `numericDev` subset, we need to delete the columns with the original data, as we only want the columns with deviation values. Finally, we print out the first six rows with the `head()` command.

```
#mutate
numericDev <- mutate(numerics,
                     ageDev = round(mean(age)-age,2),
                     RBPDev = round(mean(RBP)-RBP,2),
                     cholesterolDev = round(mean(Cholesterol)-Cholesterol,2),
                     maxHRDev = round(mean(maxHR)-maxHR,2),
                     fluoroMVDev = round(mean(fluoroMV)-fluoroMV,2))
numericDev[1:5] <- NULL
head(numericDev)
```

```
##   ageDev RBPDev cholesterolDev maxHRDev fluoroMVDev
## 1  25.13    1.7           41.2   -52.58         0.3
## 2  20.13   13.7           63.2   -24.58         0.3
## 3  20.13   13.7           35.2   -42.58         0.3
## 4  19.13   -6.3           62.2   -32.58         0.3
## 5  19.13   -0.3           53.2   -24.58         0.3
## 6  19.13   11.7           47.2    19.42         0.3
```