Lab: EDA

36-600

Fall 2022

In today's lab, you were perform exploratory data analysis on a dataset related to heart disease and the cost billed to insurance.

Data

Your first job is to retrieve the dataset, `heart_disease.csv`, from the course `Canvas` site. You will find the dataset in the `DATA` directory in the `Files` hierarchy.

Examine the downloaded data file. Think about how you would input these data (hint: do any strings represent factor variables? do you need to specify column types? etc.). Then…

Questions

Question 1

Input the data into `R`, and into a data frame named `df`.

```
df <- read.csv("heart_disease.csv",stringsAsFactors=TRUE)
```

Question 2

Summarize the data, via a base-`R` function mentioned in today's notes. Scan the output to see if there are missing data or if anything appears weird.

```
summary(df)
```

```
##       Cost              Age           Gender    Interventions        Drugs
ERVisit
##  Min.   :    0.0   Min.   :24.00   Female:608   Min.   : 0.000   Min.   :0.0000   Mi
n.   : 0.000
##  1st Qu.:  161.1   1st Qu.:55.00   Male  :180   1st Qu.: 1.000   1st Qu.:0.0000   1st
Qu.: 2.000
##  Median :  507.2   Median :60.00                Median : 3.000   Median :0.0000   Med
ian : 3.000
##  Mean   : 2800.0   Mean   :58.72                Mean   : 4.707   Mean   :0.3388   Mea
n   : 3.425
##  3rd Qu.: 1905.5   3rd Qu.:64.00                3rd Qu.: 6.000   3rd Qu.:0.0000   3rd
Qu.: 5.000
##  Max.   :52664.9   Max.   :70.00                Max.   :47.000   Max.   :2.0000   Ma
x.   :20.000
##  Complications     Comorbidities       Duration            id
##  Min.   :0.00000   Min.   : 0.000   Min.   :  0.00   Min.   :  1.0
##  1st Qu.:0.00000   1st Qu.: 0.000   1st Qu.: 41.75   1st Qu.:197.8
##  Median :0.00000   Median : 1.000   Median :165.50   Median :394.5
##  Mean   :0.05457   Mean   : 3.767   Mean   :164.03   Mean   :394.5
##  3rd Qu.:0.00000   3rd Qu.: 5.000   3rd Qu.:281.00   3rd Qu.:591.2
##  Max.   :1.00000   Max.   :60.000   Max.   :372.00   Max.   :788.0
```

Question 3

One thing you might have noticed in Question 2 is that `Drugs` apparently can only take on the values 0, 1, and 2, and that `Complications` is either 0 or 1. This hints that these are actually factor variables, and not numeric. For purposes of visualization and analysis, it can be helpful to forcibly transform these variables from being of `numeric` type to being of `factor` type. You would do that as follows:

```
df$Drugs <- factor(df$Drugs)
```

Convert both variables, and re-display the summary.

```
df$Drugs <- factor(df$Drugs)
df$Complications <- factor(df$Complications)
summary(df)
```

```
##       Cost              Age           Gender    Interventions    Drugs      ERVisit
Complications
##  Min.   :    0.0   Min.   :24.00   Female:608   Min.   : 0.000   0:610   Min.   : 0.0
00   0:745
##  1st Qu.:  161.1   1st Qu.:55.00   Male  :180   1st Qu.: 1.000   1: 89   1st Qu.: 2.0
00   1: 43
##  Median :  507.2   Median :60.00                Median : 3.000   2: 89   Median : 3.0
00
##  Mean   : 2800.0   Mean   :58.72                Mean   : 4.707           Mean   : 3.4
25
##  3rd Qu.: 1905.5   3rd Qu.:64.00                3rd Qu.: 6.000           3rd Qu.: 5.0
00
##  Max.   :52664.9   Max.   :70.00                Max.   :47.000           Max.   :20.0
00
##  Comorbidities       Duration            id
##  Min.   : 0.000   Min.   :  0.00   Min.   :  1.0
##  1st Qu.: 0.000   1st Qu.: 41.75   1st Qu.:197.8
##  Median : 1.000   Median :165.50   Median :394.5
##  Mean   : 3.767   Mean   :164.03   Mean   :394.5
##  3rd Qu.: 5.000   3rd Qu.:281.00   3rd Qu.:591.2
##  Max.   :60.000   Max.   :372.00   Max.   :788.0
```

Question 4

Look at your summary output again. Are there any obviously non-informative columns? If so, remove them here. For instance, use `dplyr` functions to remove the offending column(s), and save the output to `df`. Note: to remove a single column, you can name it and put a minus sign in front. Then show the names of the columns of `df` so you can convince yourself that the offending column(s) are gone.

```
suppressMessages(library(tidyverse))

df %>% select(.,-id) -> df
names(df)
```

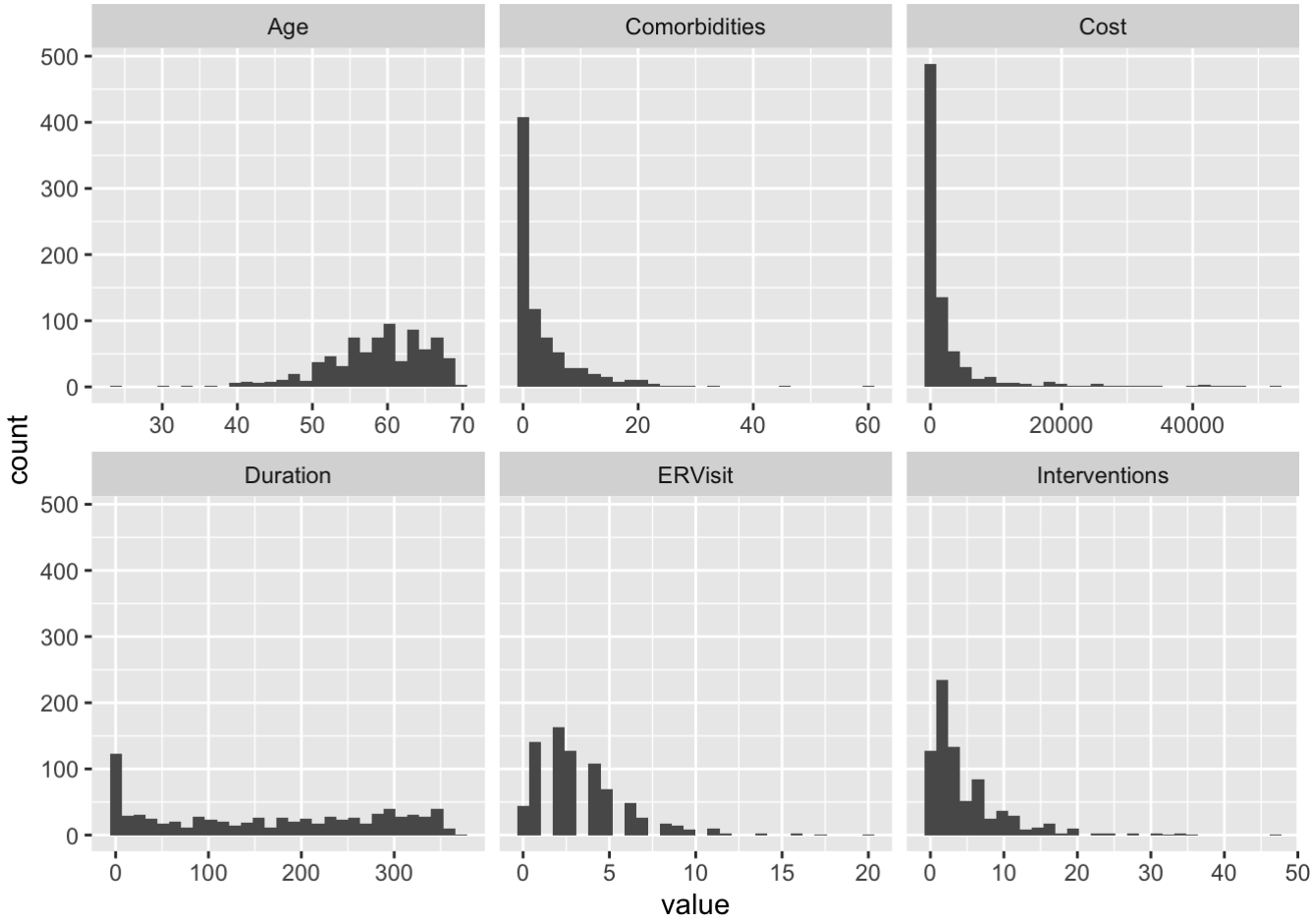
```
## [1] "Cost"          "Age"           "Gender"        "Interventions" "Drugs"
"ERVisit"       "Complications"
## [8] "Comorbidities" "Duration"
```

Question 5

Create a faceted histogram for all the variables that are truly quantitative, meaning leave `Gender`, `Drugs`, and `Complications` out. Go back to previous labs and look for how we used the `gather()` function.

```
df %>% select(.,-Gender,-Drugs,-Complications) %>% gather(.) -> df.quant
ggplot(data=df.quant,mapping=aes(x=value)) + geom_histogram() + facet_wrap(~key,scales=
'free_x')
```

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



Question 6

Look at `Cost` : it is right skew. Make a histogram of the base-10 logarithm of `Cost` , i.e., do

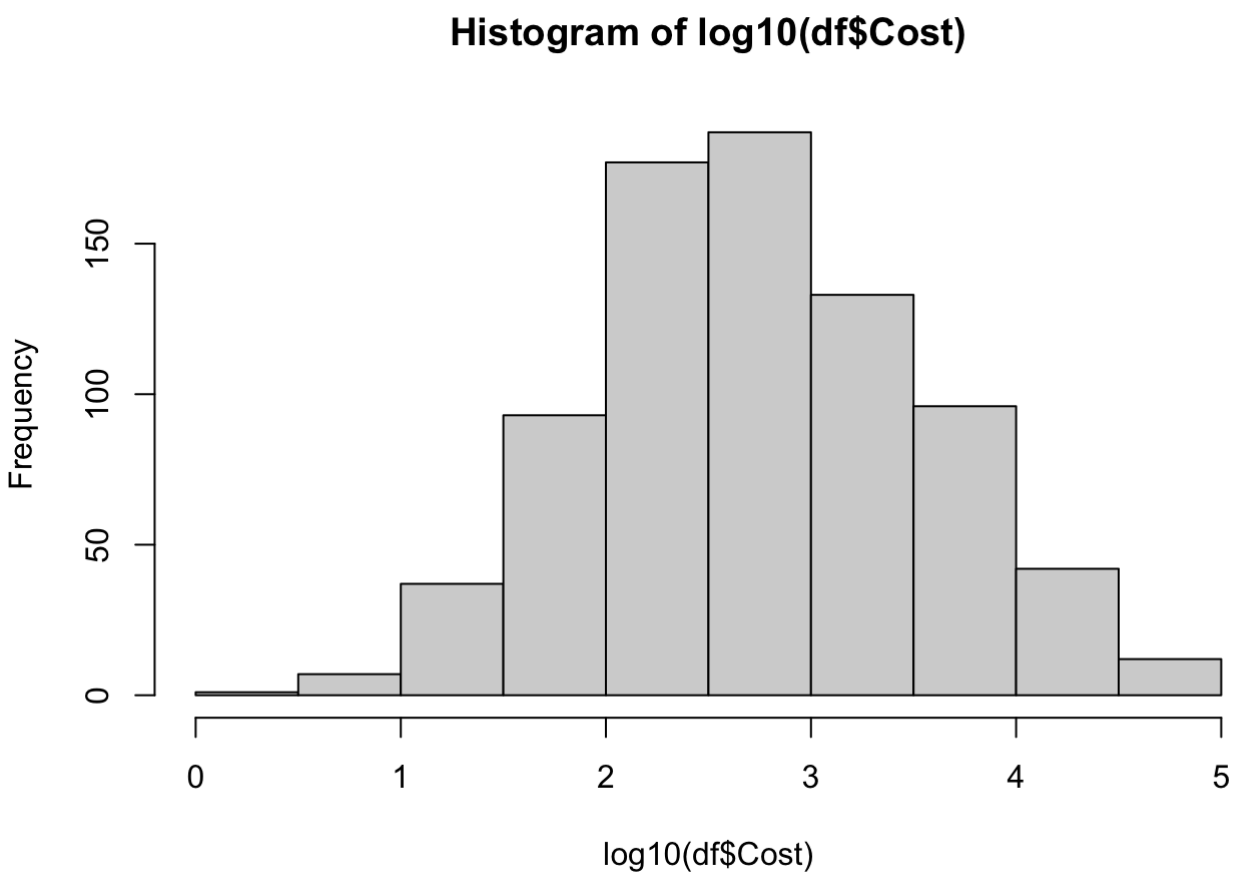
```
hist(log10(df$Cost))   # quick'n'dirty, no ggplot needed here!
```

Does this look more symmetric? If yes, replace the `Cost` column, i.e., do

```
df %>% filter(.,Cost>0) -> df
df$Cost = log10(df$Cost)
```

Note that we will not transform the other right-skew variables that have minimum values of zero.

```
hist(log10(df$Cost))
```




```
df %>% filter(.,Cost>0) -> df
df$Cost = log10(df$Cost)
```

Question 7

Create base- `R` tables and `ggplot` -style bar charts for `Gender` , `Drugs` , and `Complications` . (To be clear, issue separate function calls for each variable!)

```
table(df$Gender)
```

```
##
## Female   Male
##    608    177
```

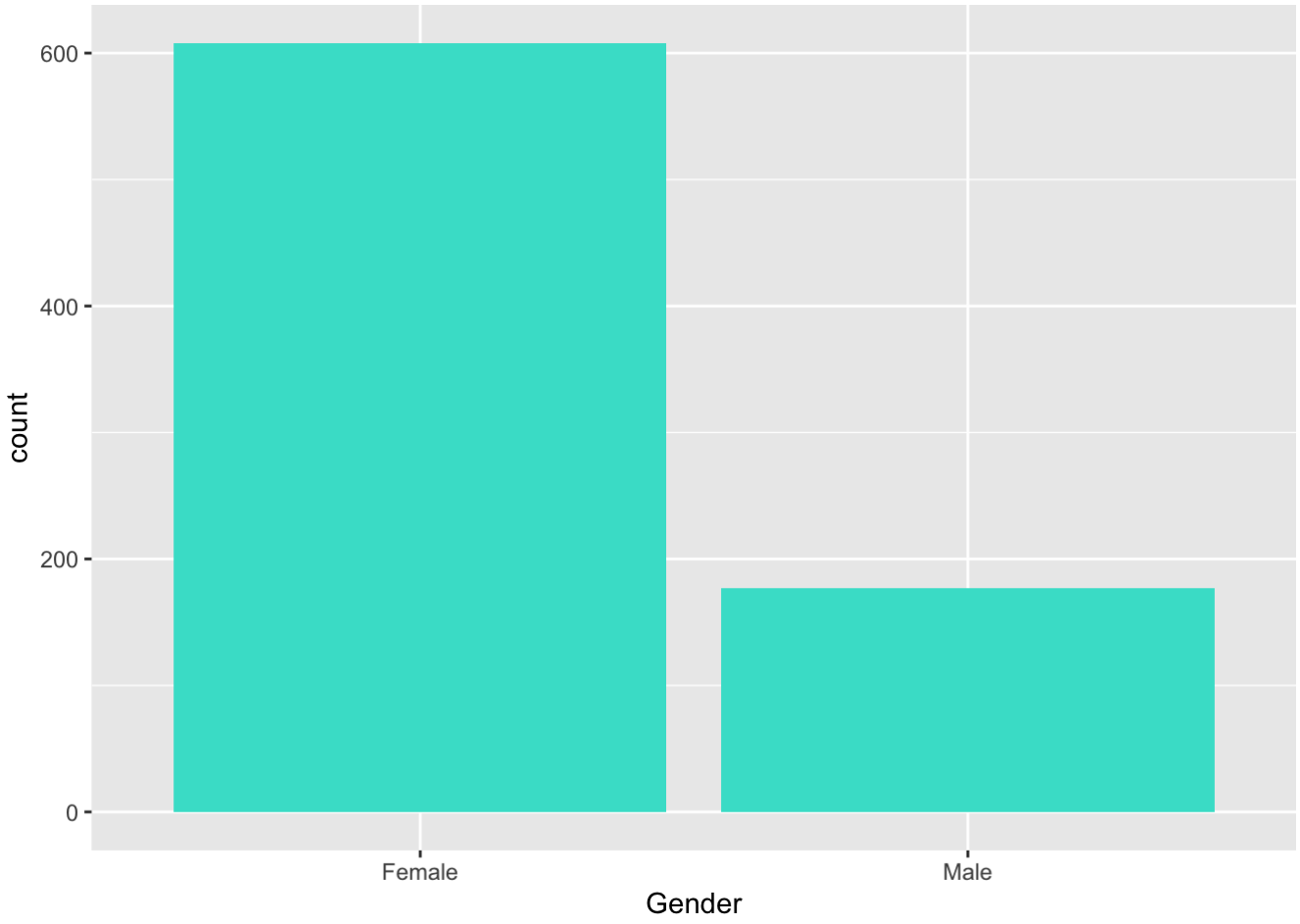
```
table(df$Drugs)
```

```
##
##   0   1   2
## 608  88  89
```

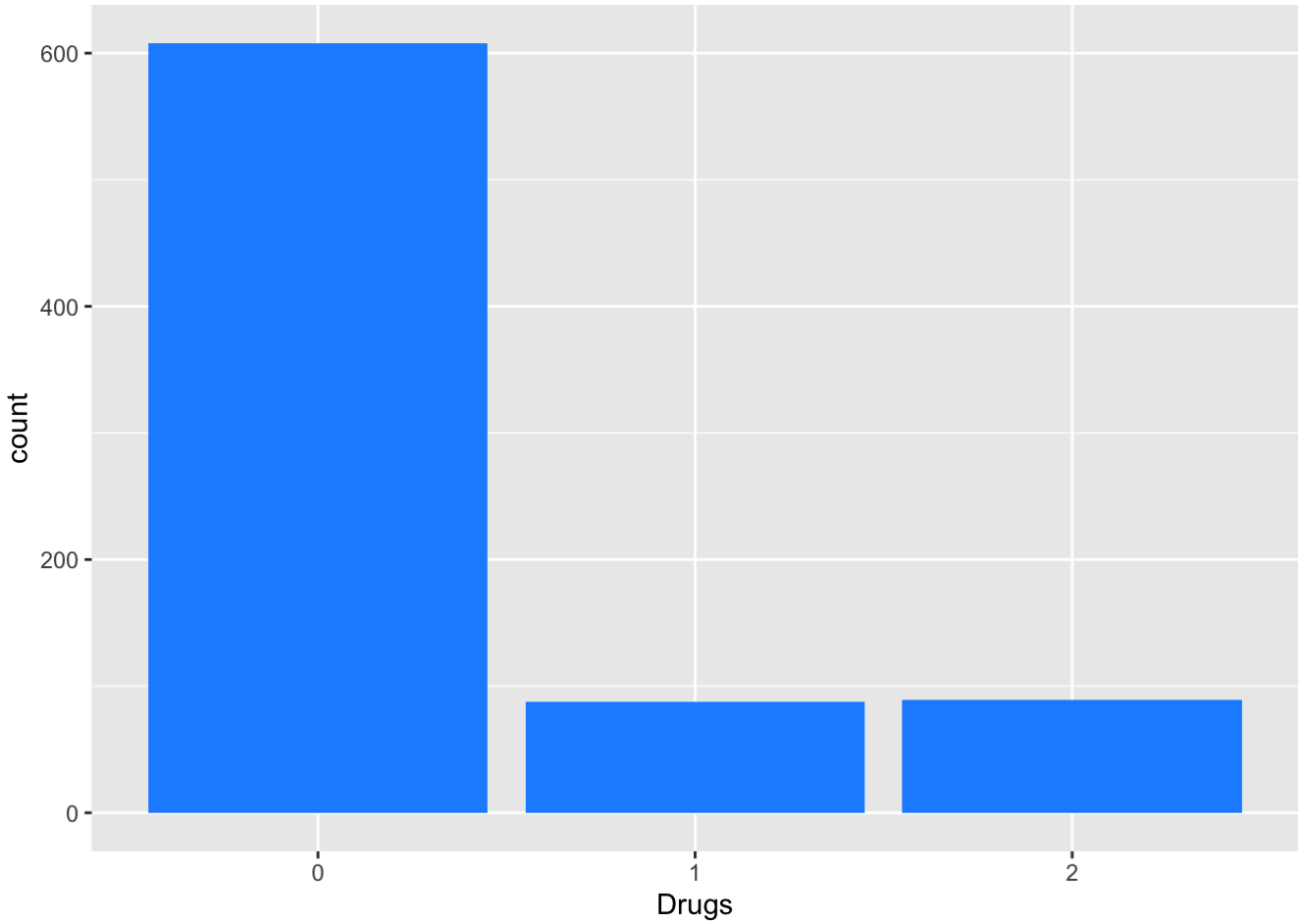
```
table(df$Complications)
```

```
##
##   0   1
## 742  43
```

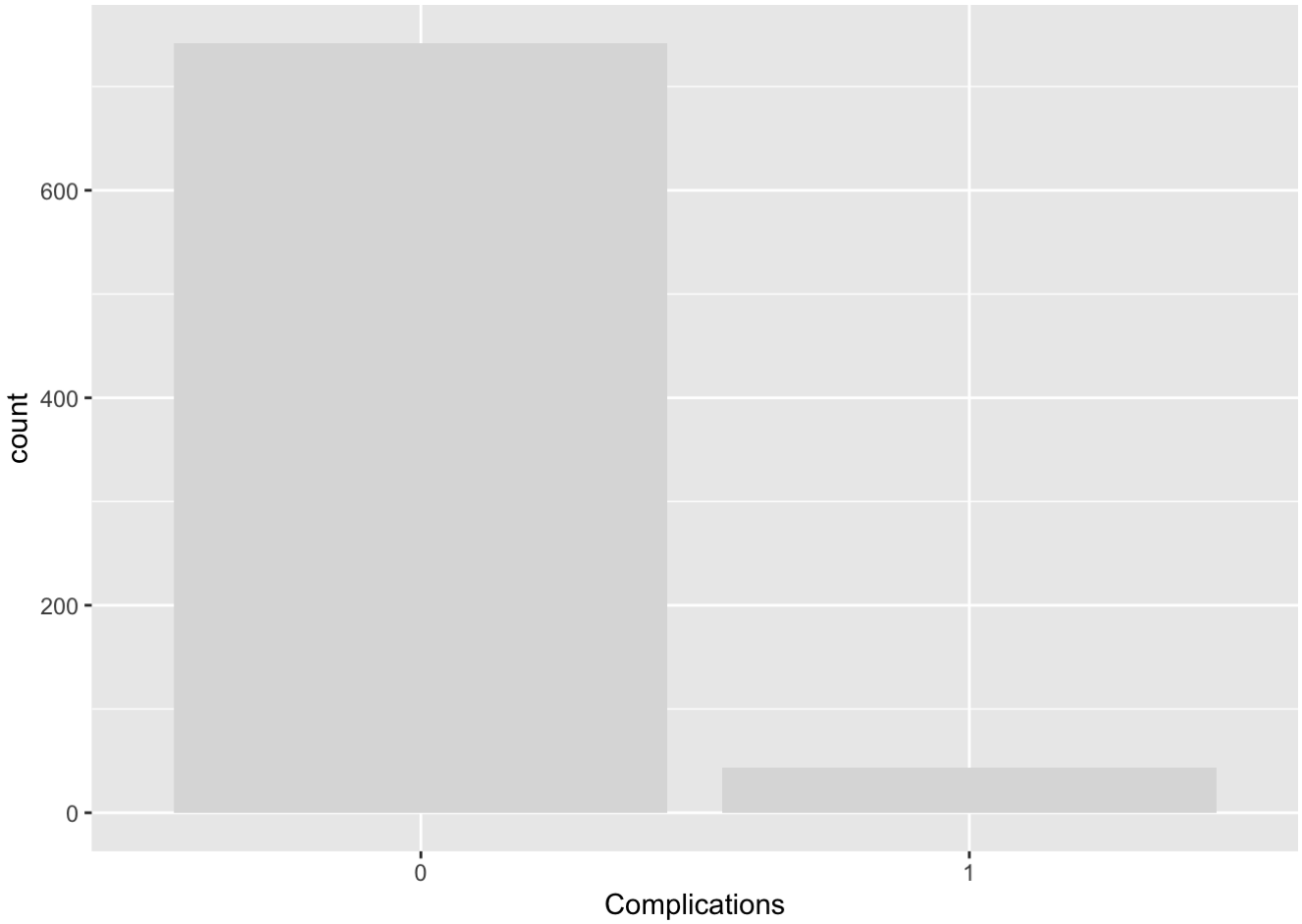
```
ggplot(data=df,mapping=aes(x=Gender)) + geom_bar(fill="turquoise")
```




```
ggplot(data=df,mapping=aes(x=Drugs)) + geom_bar(fill="dodgerblue")
```




```
ggplot(data=df,mapping=aes(x=Complications)) + geom_bar(fill="gainsboro")
```



Question 8

Let's visualize `Drugs` and `Complications` at the same time. One way to do this is via a two-way table: simply pass both variable names to `table()` and see what happens. Such visualization can also be done in `ggplot` but it is considerably more complicated a task than we want to tackle here.

```
table(df$Drugs,df$Complications)
```

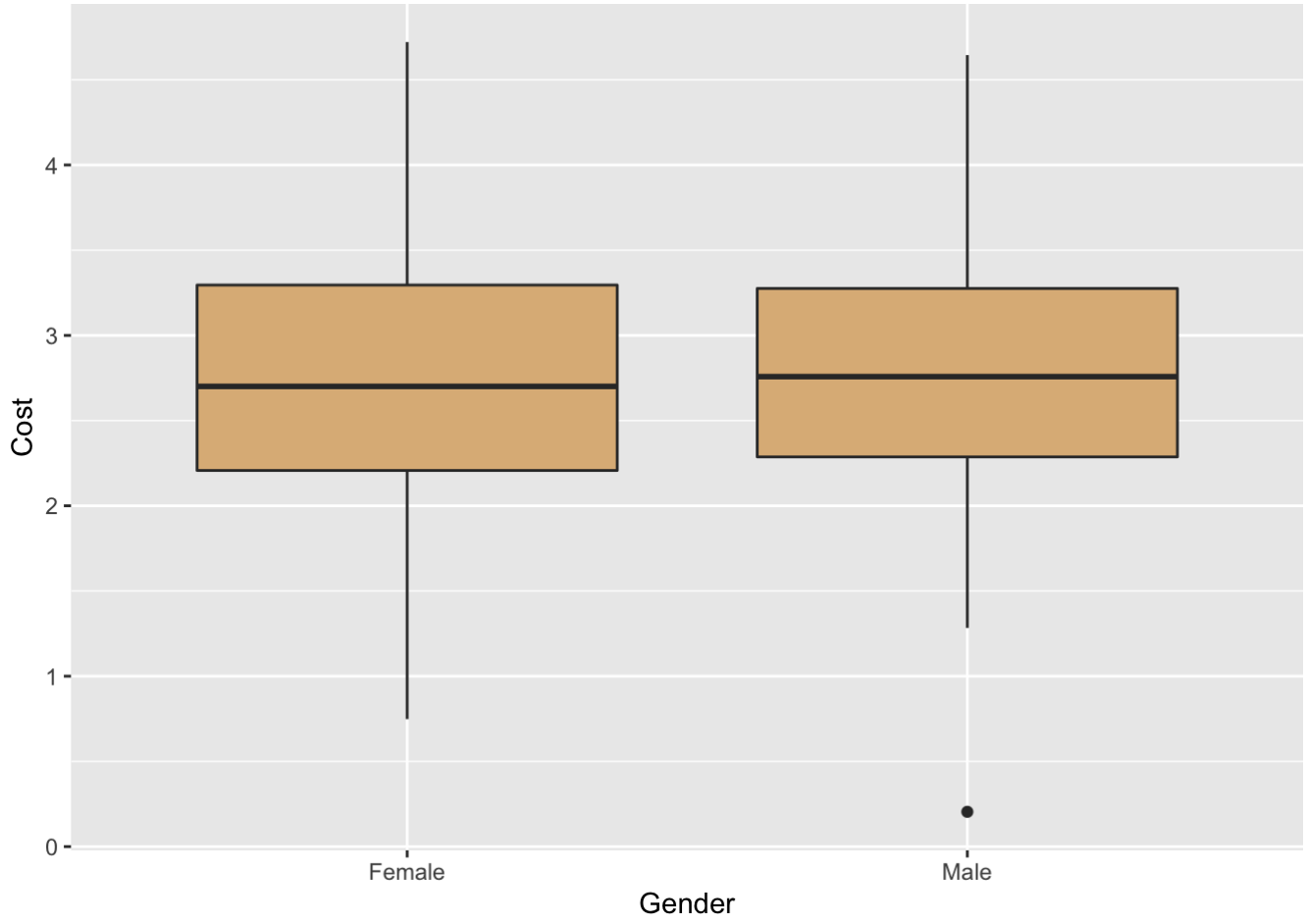
```
##
##       0   1
##   0 580  28
##   1  81   7
##   2  81   8
```

Question 9

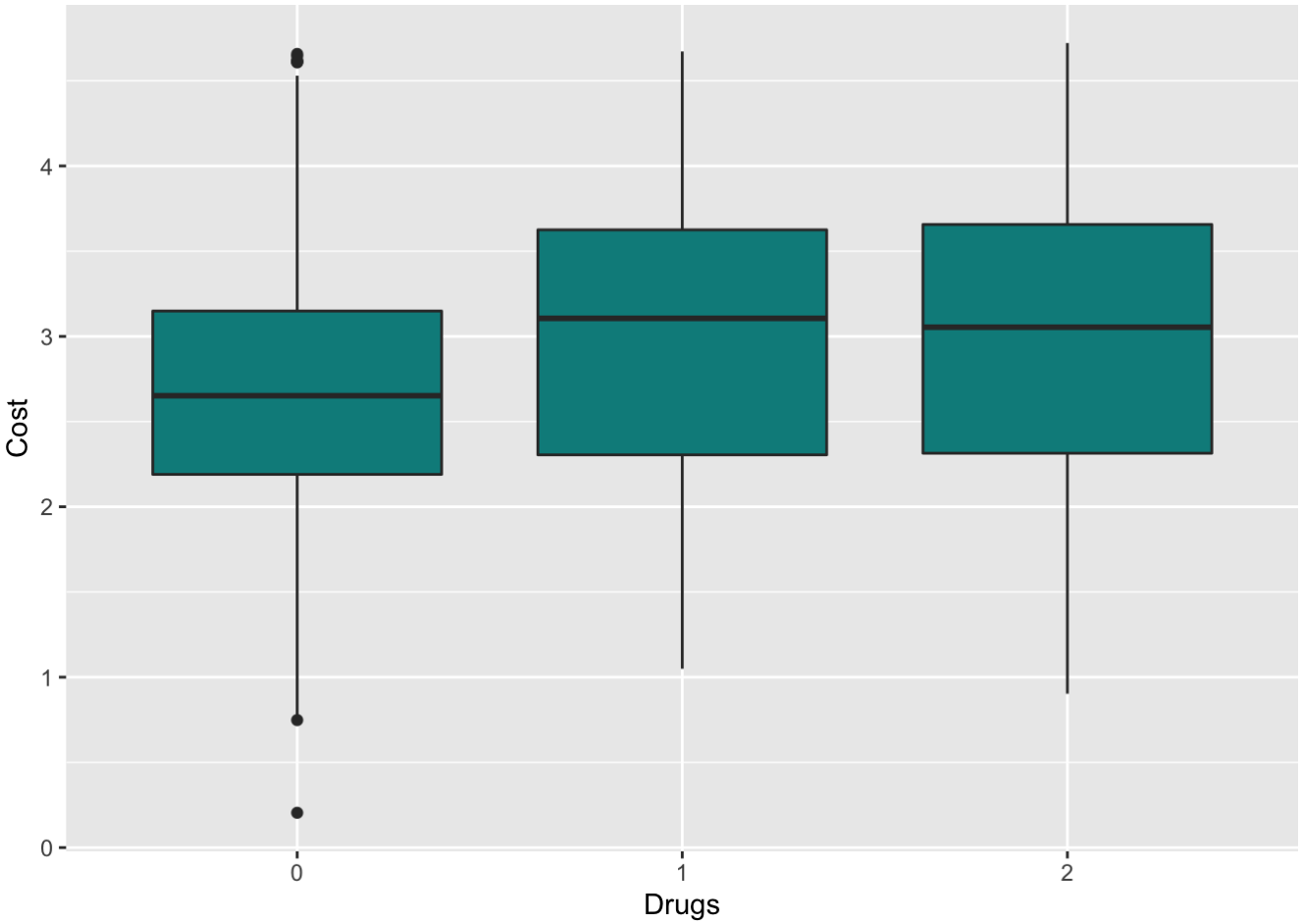
Let's assume that `Cost` is our response variable: ultimately we want to learn regression models that predict `Cost` given the values of the remaining (predictor) variables. (We'll actually carry this out later!) What we might want to do now is see how `Cost` varies as a function of other variables.

First job: create side-by-side boxplots for `Cost` vs. `Gender`, `Cost` vs. `Drugs`, and `Cost` vs. `Complications`. Just make the plots; you need not write down any conclusions you reach. Simply file them away for when we return to this dataset in a future lab.

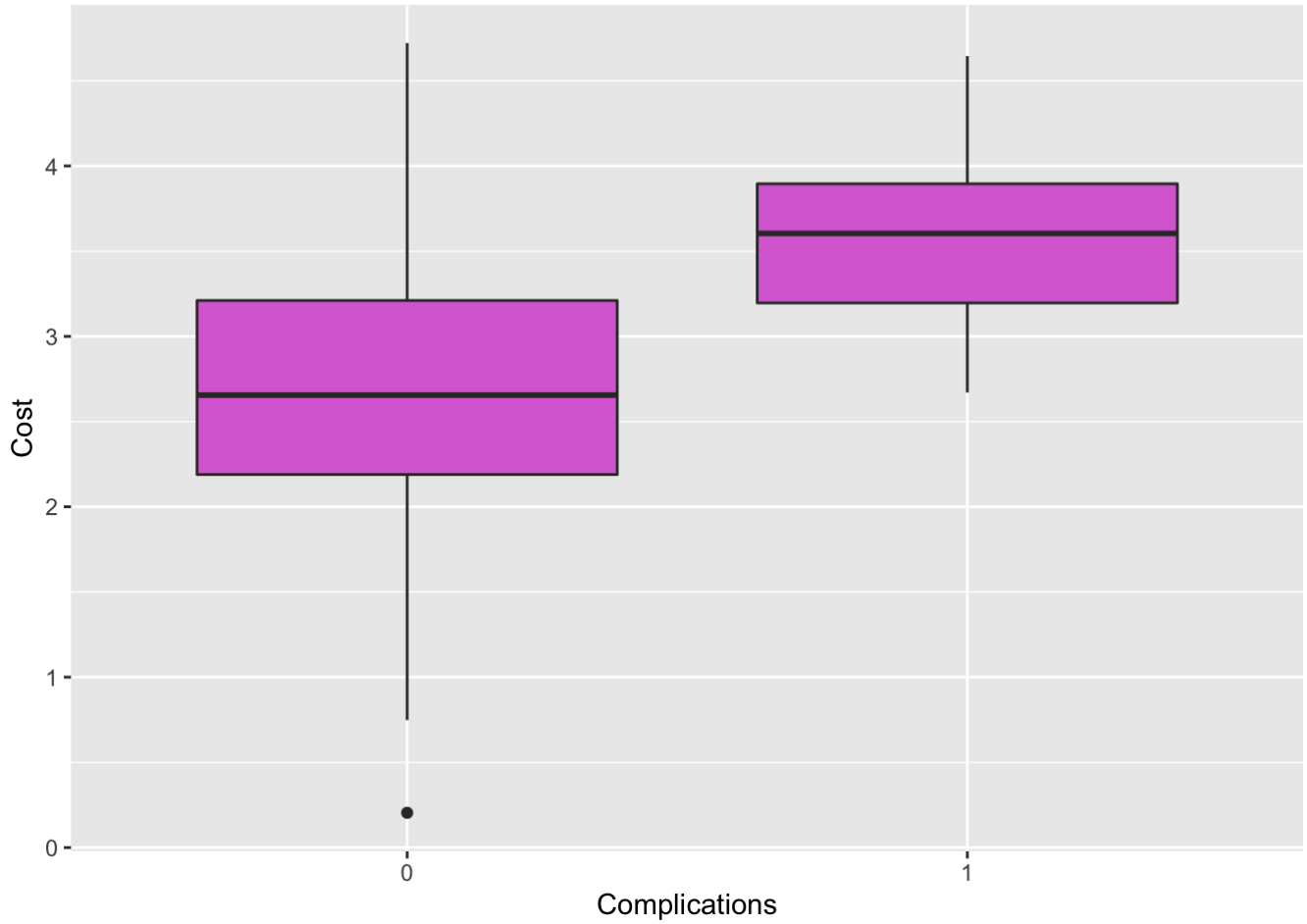
```
ggplot(data=df,mapping=aes(x=Gender,y=Cost)) + geom_boxplot(fill="burlywood")
```




```
ggplot(data=df,mapping=aes(x=Drugs,y=Cost)) + geom_boxplot(fill="darkcyan")
```



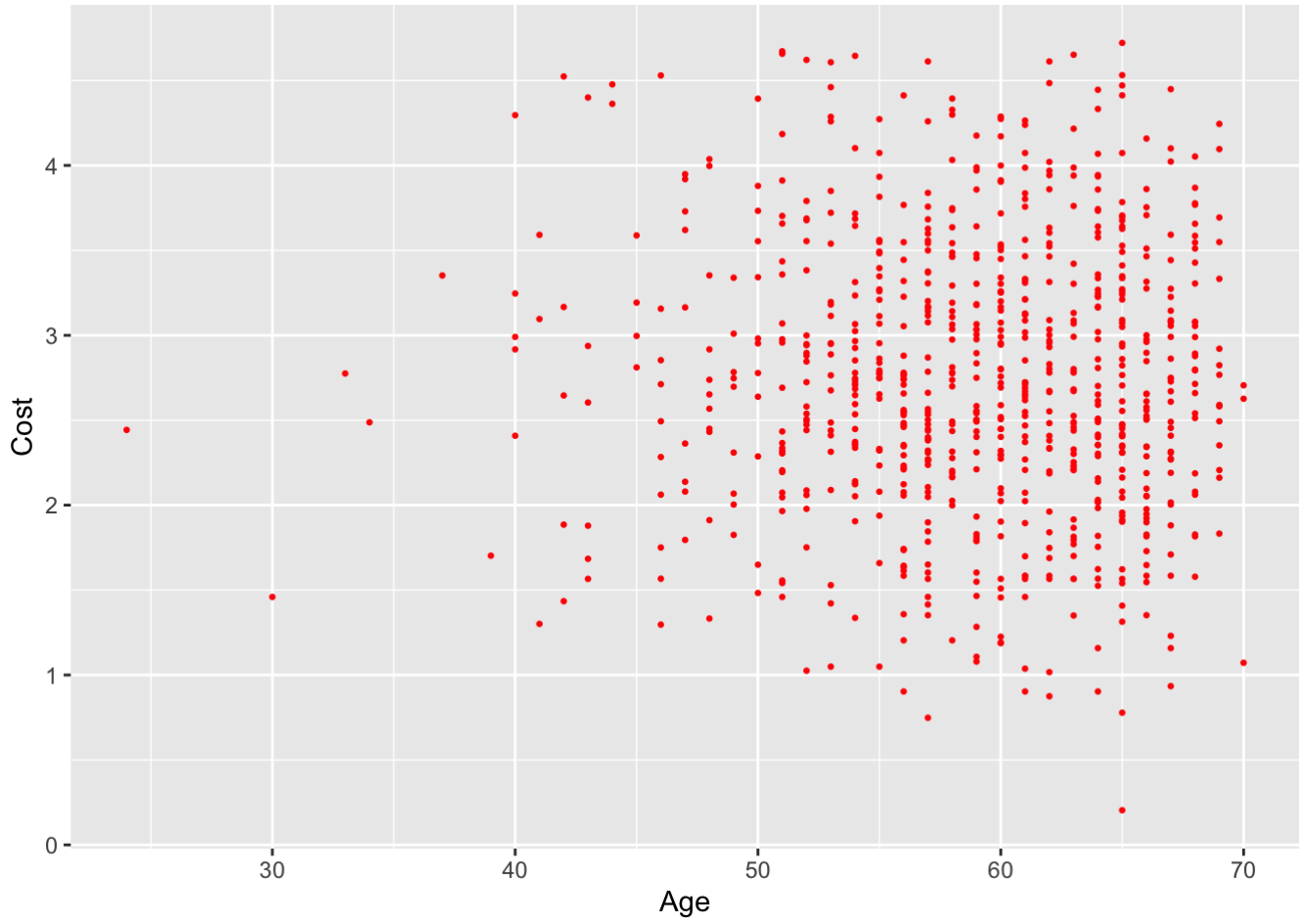

```
ggplot(data=df,mapping=aes(x=Complications,y=Cost)) + geom_boxplot(fill="orchid")
```



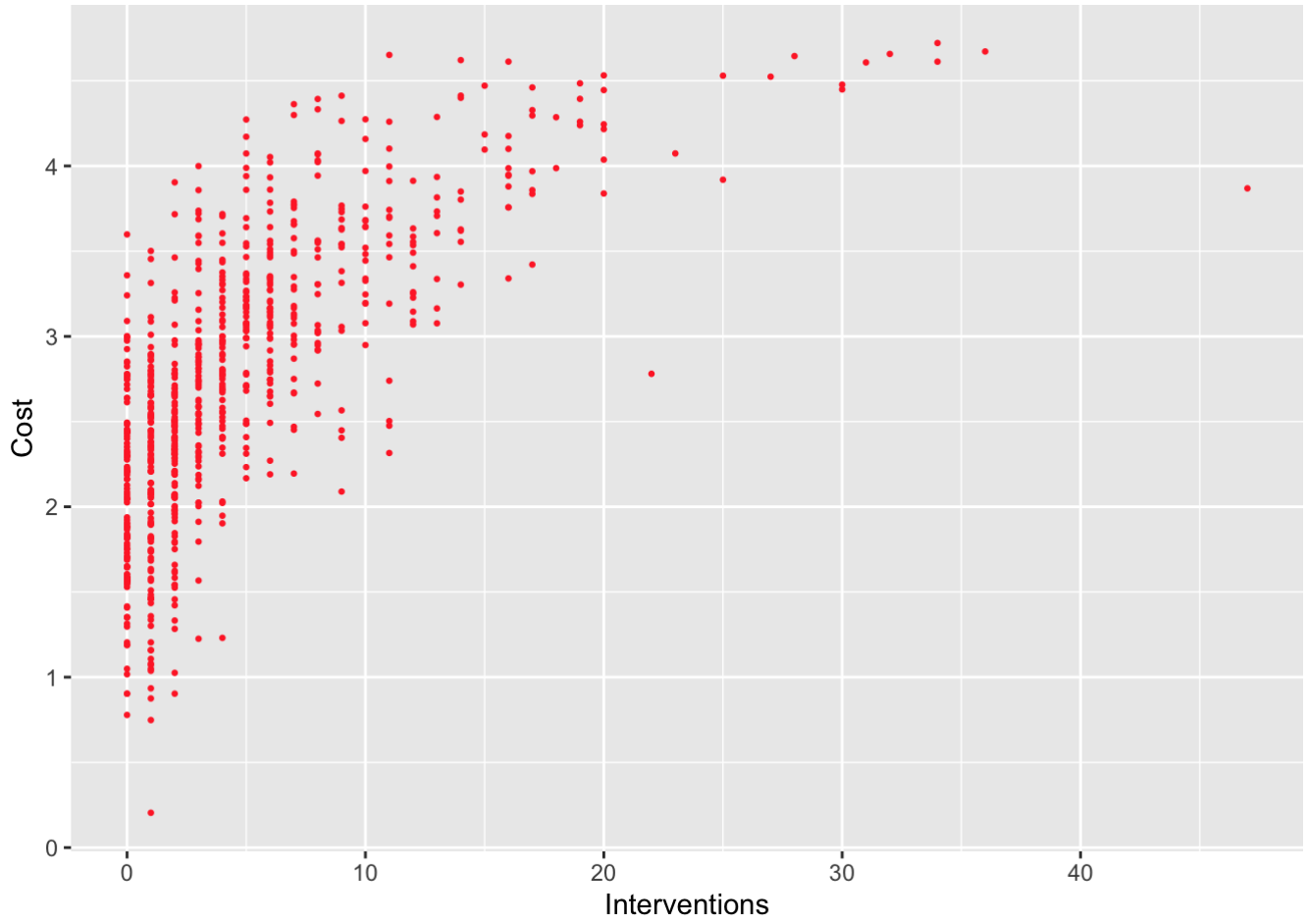
Question 10

Your next job: show scatter plots of `Cost` (y-axis) versus all the remaining predictor variables. Again, try to visually infer associations…will we eventually be able to learn a model that predicts `Cost` ? (And again, there is no need to write anything down.)

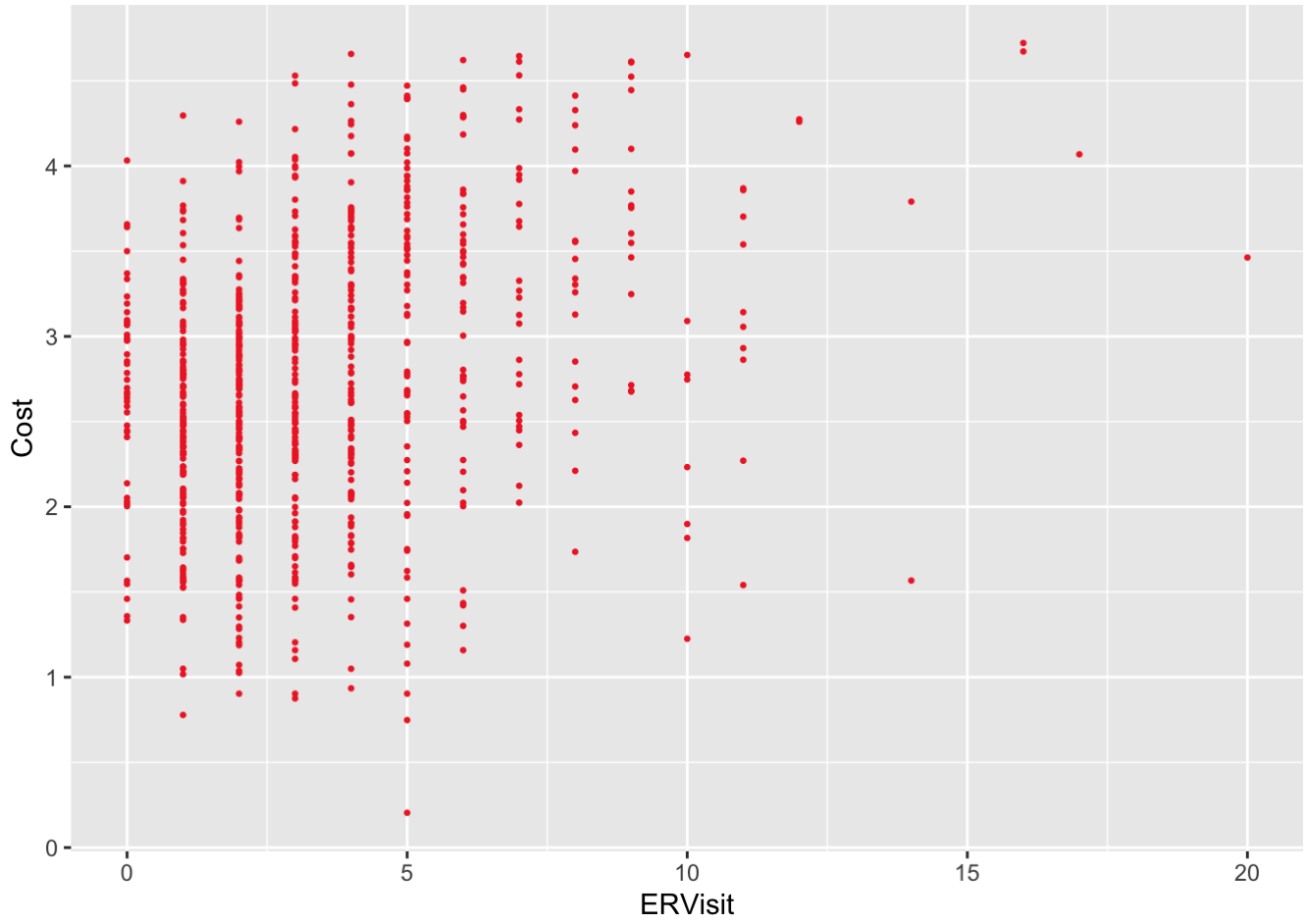
```
ggplot(data=df,mapping=aes(x=Age,y=Cost)) + geom_point(size=0.5,color="red")
```



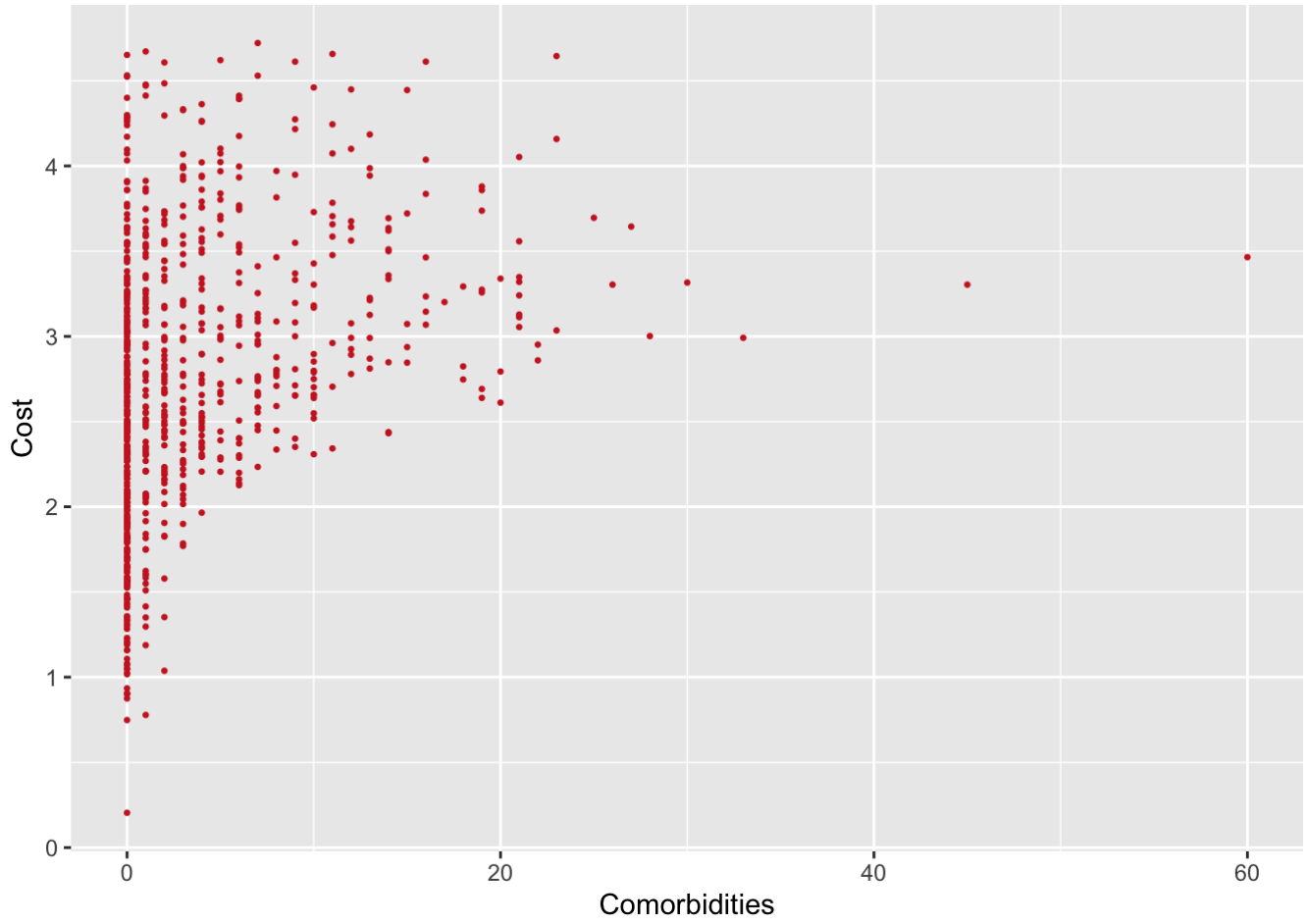

```
ggplot(data=df,mapping=aes(x=Interventions,y=Cost)) + geom_point(size=0.5,color="firebri
ck1")
```



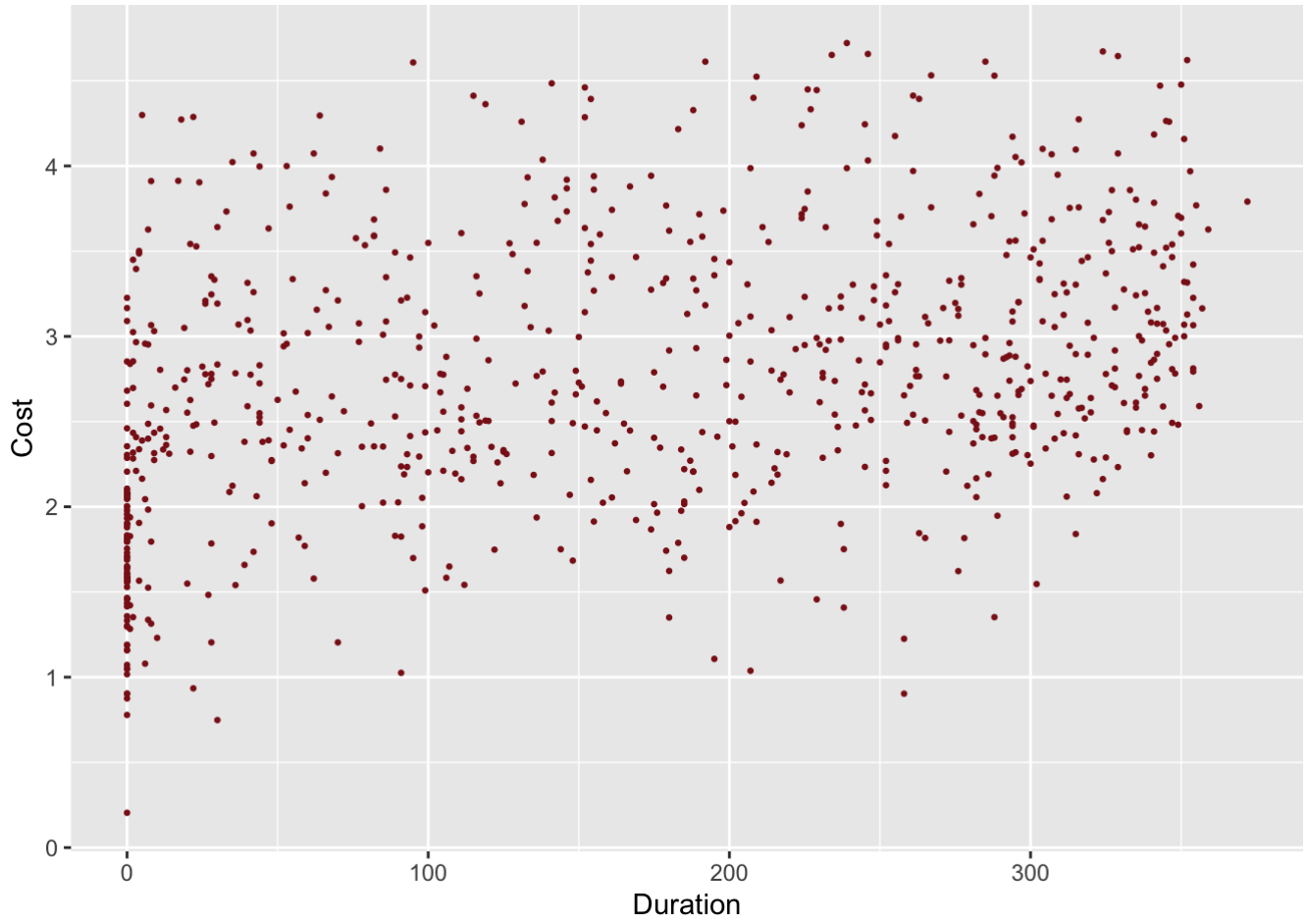

```
ggplot(data=df,mapping=aes(x=ERVisit,y=Cost)) + geom_point(size=0.5,color="firebrick2")
```




```
ggplot(data=df,mapping=aes(x=Comorbidities,y=Cost)) + geom_point(size=0.5,color="firebri
ck3")
```




```
ggplot(data=df,mapping=aes(x=Duration,y=Cost)) + geom_point(size=0.5,color="firebrick4")
```

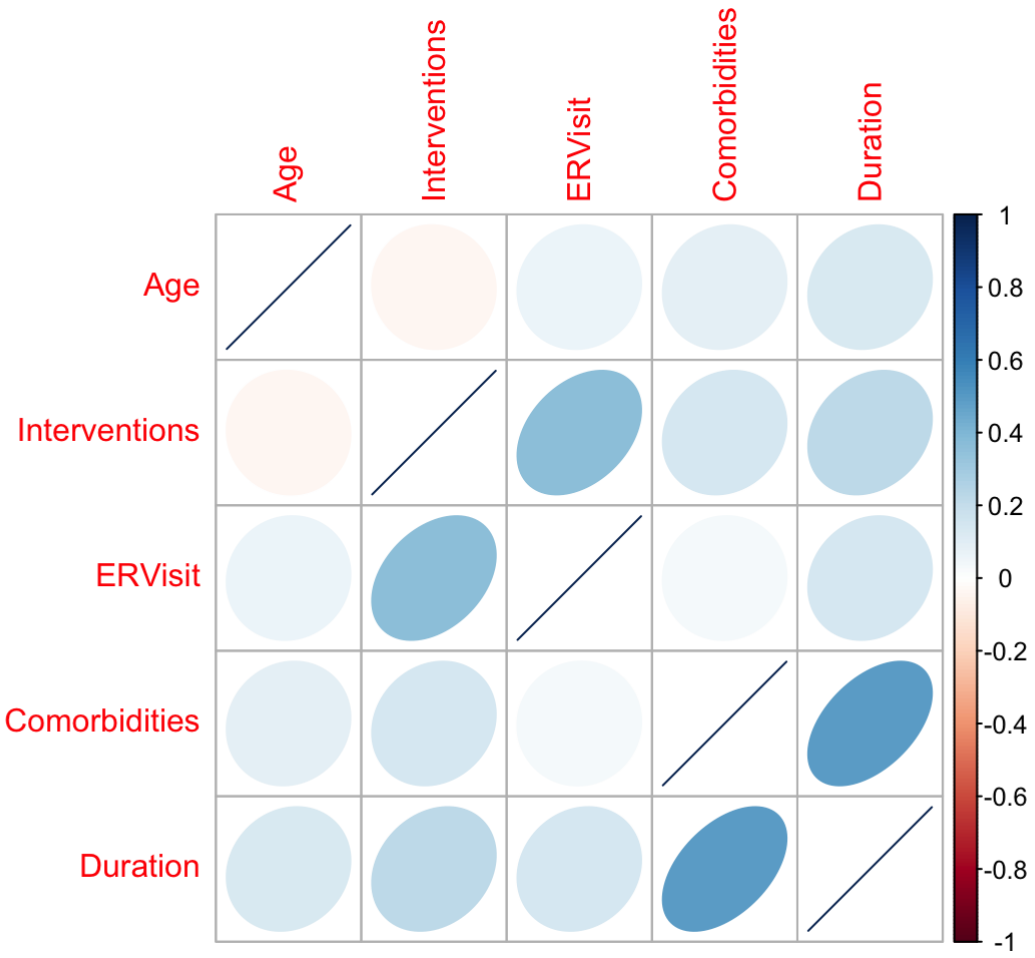


Question 11

And your next job: visually determine the level of correlation (i.e., level of linear dependence) between all the predictor variables. (Hint: `corrplot` .) Include all the variables, both quantitative and categorical. In a sense, this plot replaces the need to generate all pairwise scatter plots (of which there would be, I believe, 36 for eight predictor variables). Why might apparent associations between variables be bad, if you see any? We'll talk about this at length in a later lecture, but in short it would be evidence of *multicollinearity*, which can affect your ability to interpret any models that you learn (particularly linear regression models).

Before you start, there's a wrinkle here: `cor()` does not accept factor variables. So, remove them.

```
library(corrplot)
df %>% select(.,-Cost,-Gender,-Drugs,-Complications) %>% cor(.) %>% corrplot(.,method="e
llipse")
```



Question 12

Your last job: create a `ggpairs()` plot for all the predictor variables. (Filter out `Cost` ! Note that here, there is no need to convert the factor variables to numeric type.) Note that in the output pane, there are three buttons to the upper right: a filled square, two carets, and an x. Click on the filled square to create a new window with your plot, which you can then resize to make larger and easier to see. Note that just about all the information you could ever want is on this plot, but it lends itself to a certain amount of cognitive overload, to put it lightly.

```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
df %>% select(.,-Cost) %>% ggpairs(.,progress=FALSE)
```

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

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