Using R Control Structures

Karen Mazidi

Exploring R Control Structures with the PimaIndiansDiabetes2 data

First we load the mlbench package and then the diabetes data set. The data has 768 observations of 9 variables, which we can see with the structure str() function. The data was collected from Pima Indian women in the late 1990s.

```
library(mlbench)
data(PimaIndiansDiabetes2)
df <- PimaIndiansDiabetes2   # df points to the data frame
str(df)
```

```
## 'data.frame':    768 obs. of  9 variables:
##  $ pregnant: num  6 1 8 1 0 5 3 10 2 8 ...
##  $ glucose : num  148 85 183 89 137 116 78 115 197 125 ...
##  $ pressure: num  72 66 64 66 40 74 50 NA 70 96 ...
##  $ triceps : num  35 29 NA 23 35 NA 32 NA 45 NA ...
##  $ insulin : num  NA NA NA 94 168 NA 88 NA 543 NA ...
##  $ mass    : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 NA ...
##  $ pedigree: num  0.627 0.351 0.672 0.167 2.288 ...
##  $ age     : num  50 31 32 21 33 30 26 29 53 54 ...
##  $ diabetes: Factor w/ 2 levels "neg","pos": 2 1 2 1 2 1 2 1 2 2 ...
```

Look for NAs

The sapply() function applies a function to elements of a list. In this case the elements of the list are columns in our data frame. The function is an anonymous function (we didn't name it), and it just sums, ignoring NAs.

```
sapply(PimaIndiansDiabetes2, function(x) sum(is.na(x)))
```

```
## pregnant  glucose pressure  triceps  insulin     mass pedigree      age
##        0        5       35      227      374       11        0        0
## diabetes
##        0
```

Write a function

As an example of how to write a function, we write a function named fill_NA that takes two arguments and returns a vector. In R the return() statement is often not needed since R will return the last thing evaluated. In this function, the mean_med variable is a switch to choose whether to fill NAs with the mean or the median.

After the function is defined, we can call it with different columns. Then we use the complete.cases() function to get rid of the remaining rows that have NAs.

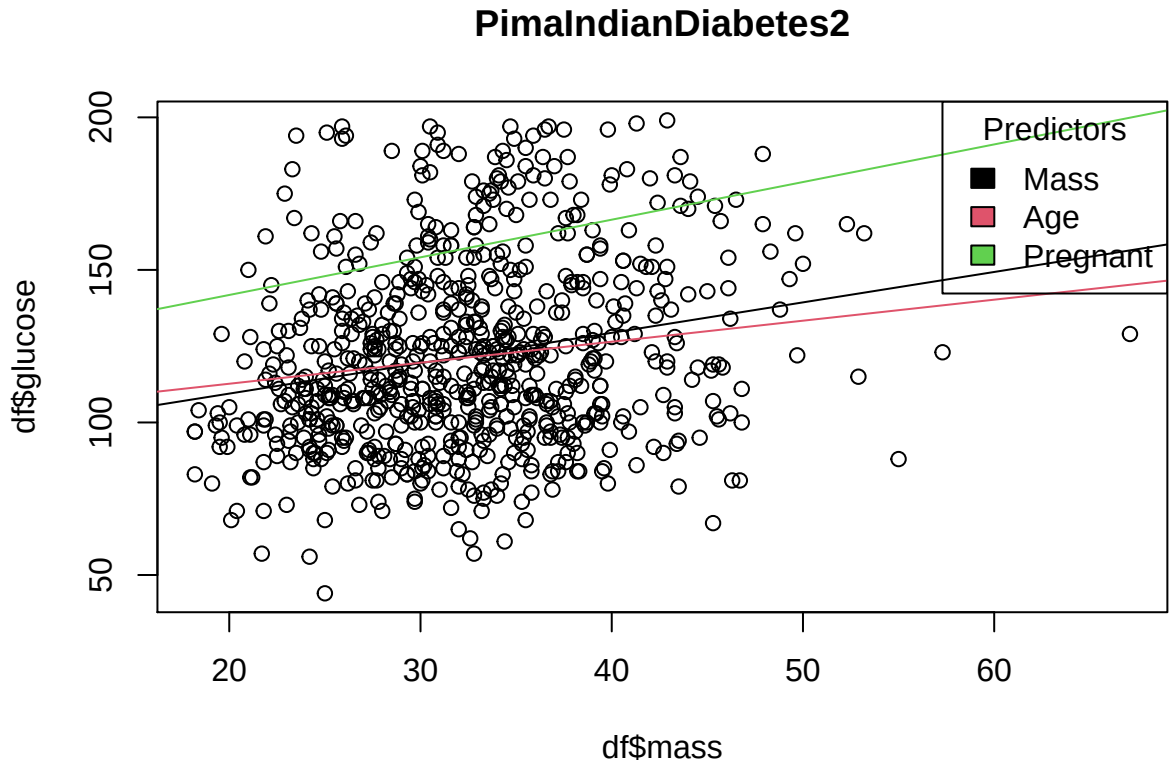
```r
fill_NA <- function(mean_med, v){
  # fill missing values with either 1=mean or 2=median
  if (mean_med == 1){
    m <- mean(v, na.rm=TRUE)
  } else {
    m <- median(v, na.rm=TRUE)
  }
  v[is.na(v)] <- m
  v
}

# make a new data set with NA's filled
df <- PimaIndiansDiabetes2[]
df$triceps <- fill_NA(1, df$triceps)
df$insulin <- fill_NA(1, df$insulin)
df <- df[complete.cases(df),]
```

Plots and for loops

The following plots some data from the data set and 3 ablines which are 3 linear regression lines created by lm(). Each regression line is a different color and we added a legend.

```r
cols <- c(6,8,1)
plot(df$mass, df$glucose, main="PimaIndianDiabetes2")
for (i in 1:3){
    model <- lm(glucose~df[,cols[i]], data=df)[1]
    abline(model, col=i)
}
legend("topright", title="Predictors", c("Mass", "Age", "Pregnant"), fill=c(1,2,3))
```



Using ifelse()

We use ifelse() to set a variable=1 if insulin is over 155 and 0 otherwise. Then we convert it to a factor, and place it in a new variable.

Then we plot.

```
df$large <- factor(ifelse(df$insulin>155,1,0))
plot(df$mass, df$glucose, pch=21, bg=c("blue","red")[unclass(df$large)])
```