Activity 6 - R Programming - If condition and Nested loop

Gioconda Prada

…

---

1. Let's write `if` and `else` statement and `nested for loop`.

a. We have a vector of area codes and we want to recode them as either in New York or outside New York. Show a frequency table displaying how many are in NY and outside NY.

```
area.codes <-
c(865,865,423,615,615,206,626,514,308,308,514,931,425,212,917,585,607,718,347
,929) #data
ny.codes <-
c(212,332,646,917,315,516,518,585,607,631,716,718,845,914,680,838,934,347,929
) #area codes in NY

in.newyork <- c()
for (i in 1:length(area.codes)) {
  if( area.codes[i] %in% ny.codes ) { in.newyork[i] <- "yes" }
  else { in.newyork[i] <- "no" }
}
in.newyork <- factor(in.newyork)
table(in.newyork)
## in.newyork
##  no yes
##  13   7
```

b. In the CUSTLOYALTY data in regclass, the lifetime values and total transactions of customers are recorded along with some of their characteristics. Let's find the average total transactions for each of the 6 income groups for married, then do the same again for single. Show a matrix that reports the average total transactions for each income group and marital status.

```
library(regclass)
## Loading required package: bestglm
## Loading required package: leaps
## Loading required package: VGAM
## Loading required package: stats4
## Loading required package: splines
## Loading required package: rpart
## Loading required package: randomForest
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
## Important regclass change from 1.3:
## All functions that had a . in the name now have an _
## all.correlations -> all_correlations, cor.demo -> cor_demo, etc.
data(CUSTLOYALTY)
levels(CUSTLOYALTY$Married)
## [1] "Married" "Single"
levels(CUSTLOYALTY$Income)
## [1] "f0t30"    "f30t45"   "f45t60"   "f60t75"   "f75t90"   "f90toINF"

VALUE <- matrix(0, nrow = nlevels(CUSTLOYALTY$Married), ncol =
nlevels(CUSTLOYALTY$Income))
rownames(VALUE) <- levels(CUSTLOYALTY$Married)
colnames(VALUE) <- levels(CUSTLOYALTY$Income)

for (marital_status in 1:length(levels(CUSTLOYALTY$Married))) {
  for (incomegroup in 1:length(levels(CUSTLOYALTY$Income))) {
    SUB <- subset(CUSTLOYALTY, Married ==
levels(CUSTLOYALTY$Married)[marital_status] & Income ==
levels(CUSTLOYALTY$Income)[incomegroup])
    VALUE[marital_status, incomegroup] <- mean(SUB$TotTransactions)
  }
}
VALUE
##            f0t30   f30t45   f45t60  f60t75   f75t90 f90toINF
## Married 2.909091 5.241379 5.600000 8.52381 7.125000 9.250000
## Single  3.611111 5.044118 6.313433 6.80000 7.402778 7.955556
```

Activity 7

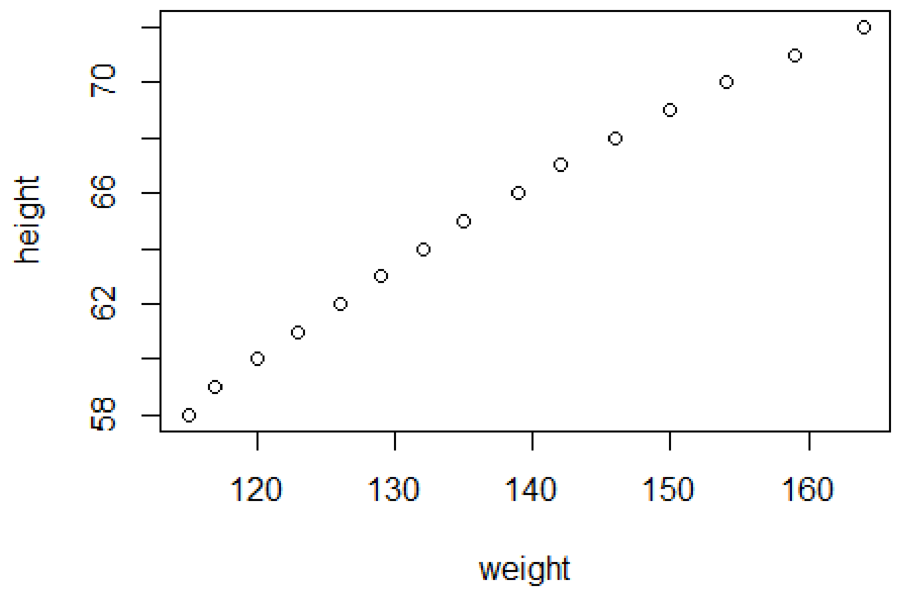
1. We shall generate several figures by using generic `plot` function.
a. Let's load `women` data which is saved in `dataset` R package. And check what variables are included in this dataset by using `names` Weightfunction.

```
data(women, package="datasets")
names(women)
## [1] "height" "weight"
```

---

b. Let's generate a basic plot by displaying x=height, y=weight.

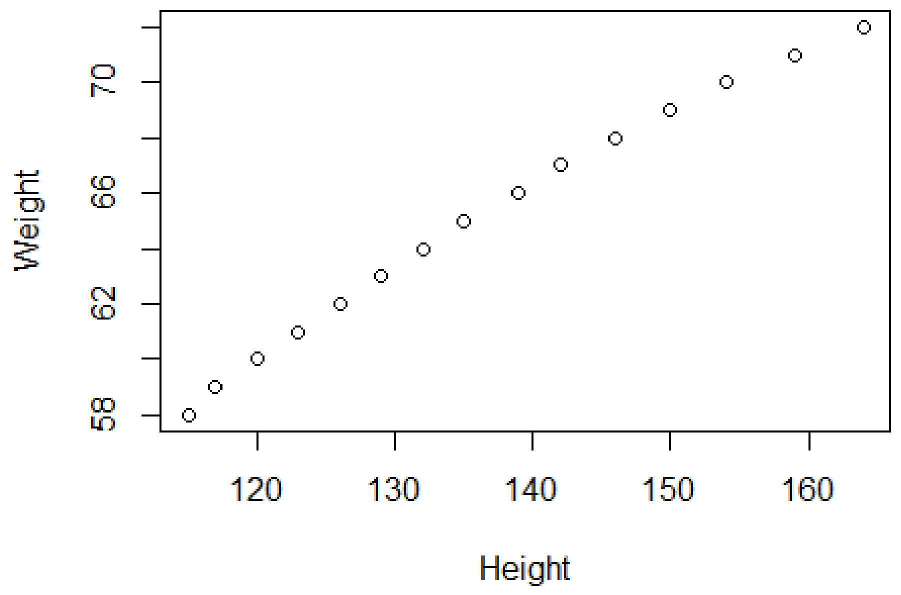
```
plot(height~weight, data=women)
```



---

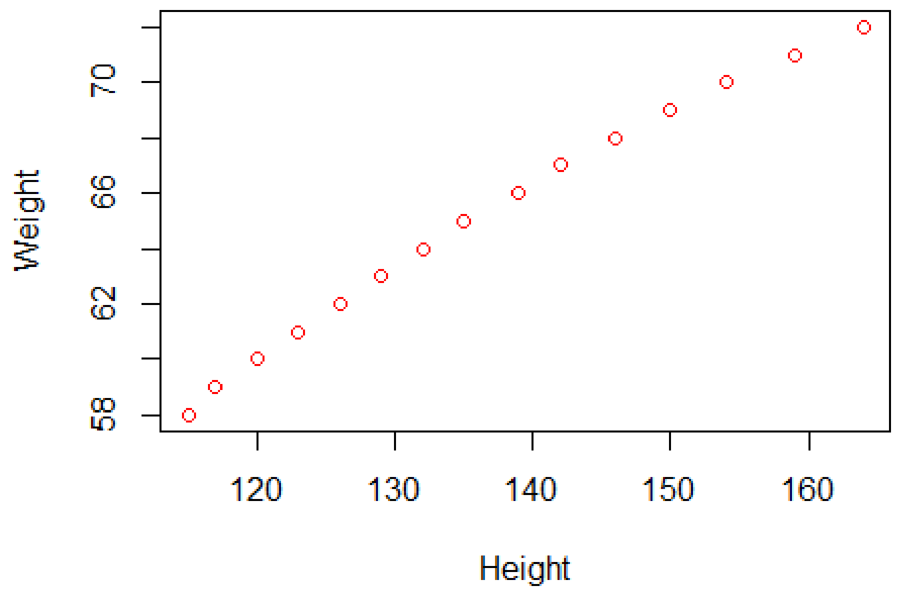
c. Let's add labels for x and y, e.g., x-label = Height, y-label = Weight.

```
plot(height~weight, data=women, xlab="Height", ylab="Weight")
```



d. Let’s change the color of the points to be `red`.

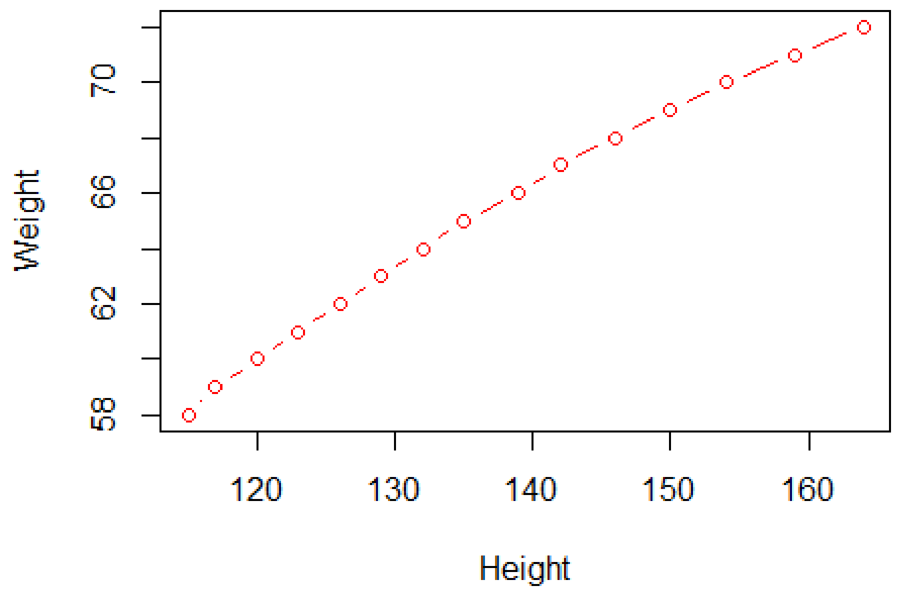
```
plot(height~weight, data=women, xlab="Height", ylab="Weight", col="red")
```



---

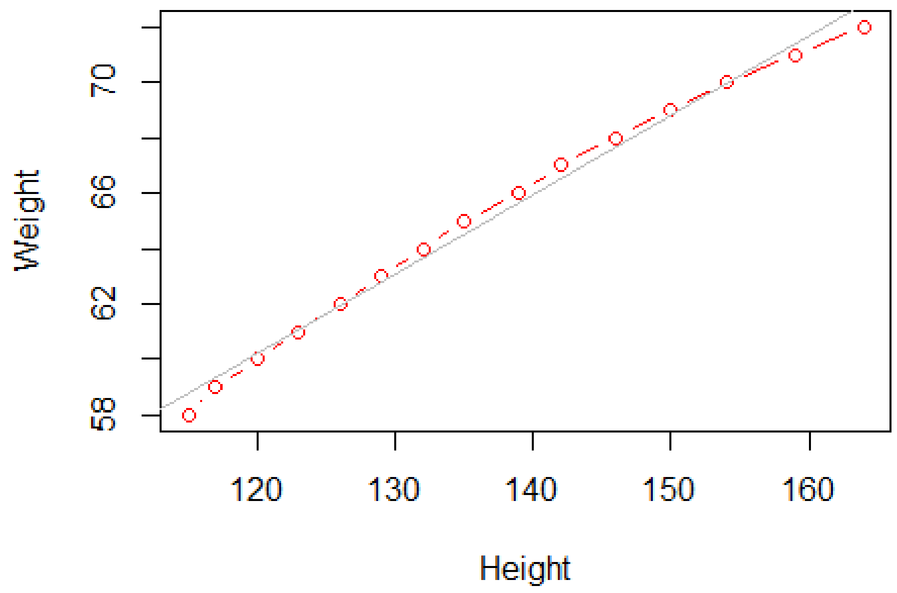
e. Let’s generate a different type of a plot which displays both the points and lines. Hint: turn on `type` argument in a generic `plot()` function.

```
plot(height~weight, data=women, xlab="Height", ylab="Weight", col="red",
type='b')
```



f. Let's add a regression line to the previous plot and let this regression line has gray color.

```
plot(height~weight, data=women, xlab="Height", ylab="Weight", col="red",
type='b')
abline(lm(height~weight, data=women), col="gray")
```

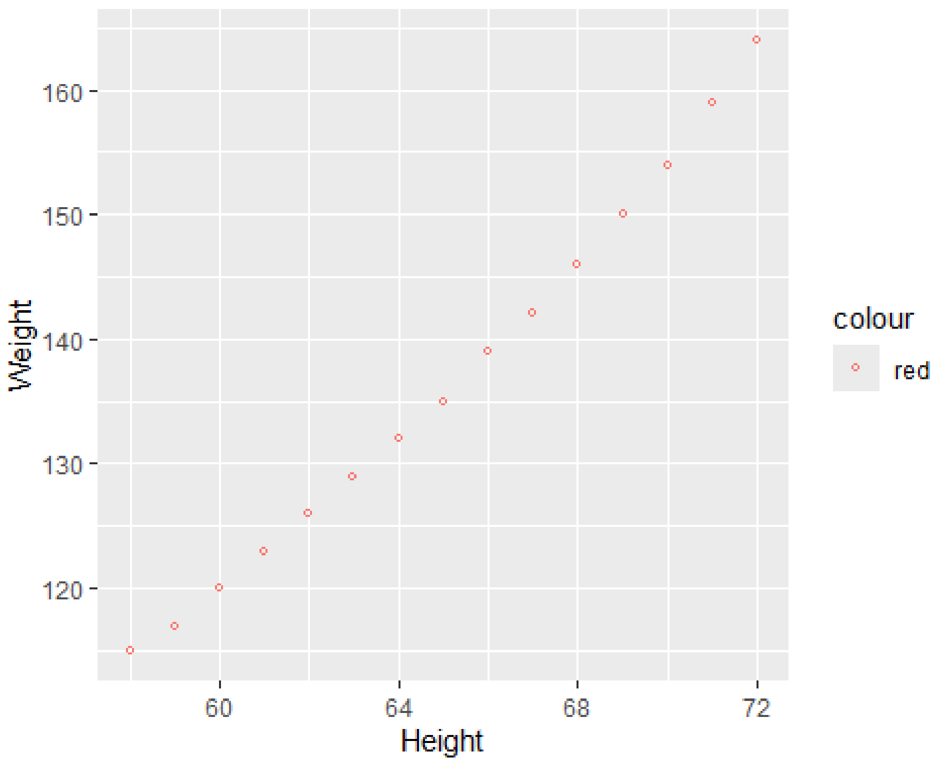


2. We will use ggplot2 R package to generate similar plots. Load ggplot2 R package.

```
#install.packages("ggplot2")
library(ggplot2)
## Warning: package 'ggplot2' was built under R version 4.3.3
##
## Attaching package: 'ggplot2'
## The following object is masked from 'package:randomForest':
##
##     margin
```

a. Use `ggplot()` function to generate a (point) plot with x=Height, y=Weight. Add labels for x and y like the graph in Question 1 (c). And let the points be `red`. Also, let `pch=1` and `size=1` for the points. Hint: Use `ggplot()`, `geom_point()`, `ylab`, and `xlab` (See Visualization slides)

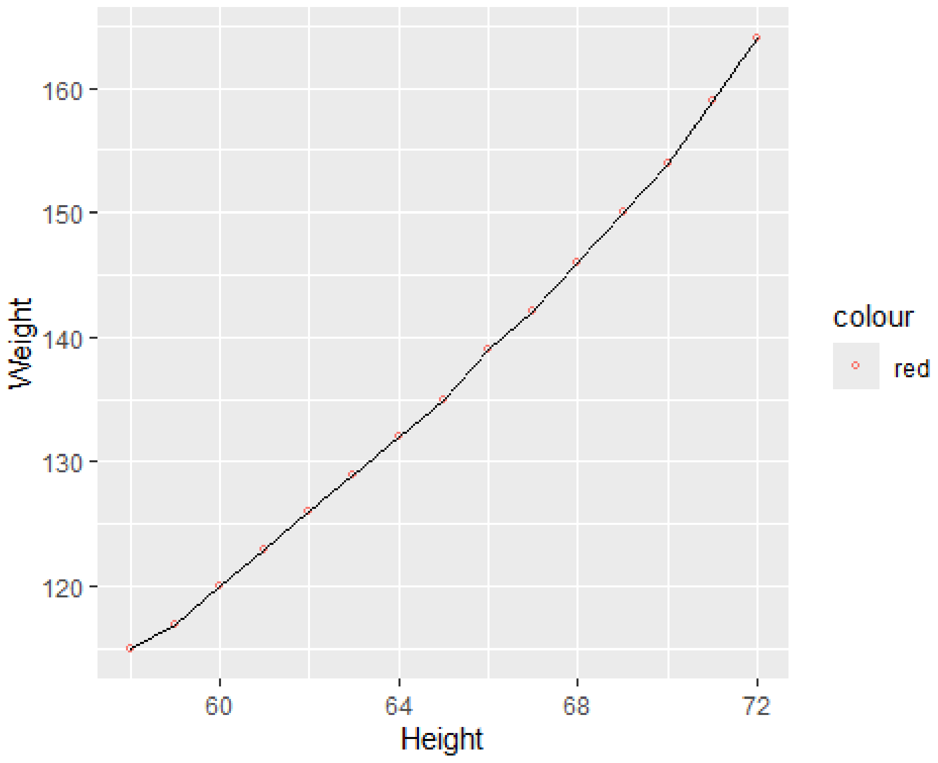
```
ggplot(women, aes(x = height, y = weight, color='red'))+
geom_point(size = 1, pch = 1)+
  labs(y="Weight", x="Height")
```



---

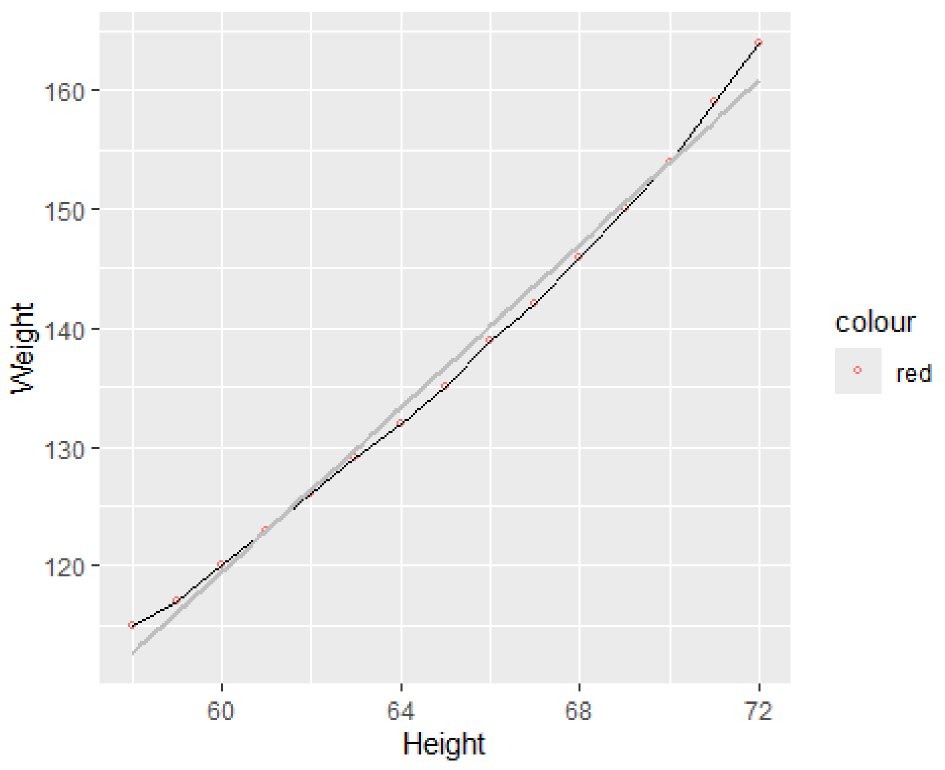
b. Let's add lines to the previous graph and connect the points. Let the lines be black color. Hint: Use `geom_line()` (See Visualization slides)

```
ggplot(women, aes(x = height, y = weight, color='red'))+
geom_point(size = 1, pch = 1)+
  labs(y="Weight", x="Height")+
  geom_line(color="black")
```



c. Let’s add a regression line (without the confidence intervals) to the previous graph and let this regression line has gray color. Hint: Use `geom_smooth()` (See Visualization slides)

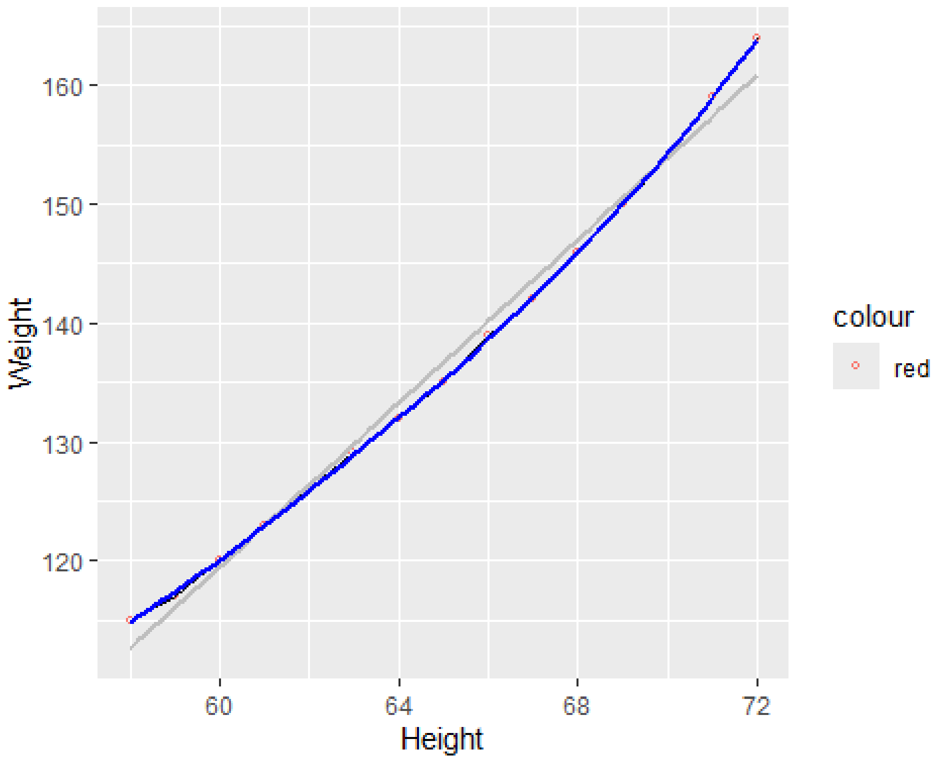
```
ggplot(women, aes(x = height, y = weight, color='red'))+
geom_point(size = 1, pch = 1)+
  labs(y="Weight", x="Height")+
  geom_line(color="black")+
  geom_smooth(method="lm", color="gray",se=FALSE)
## `geom_smooth()` using formula = 'y ~ x'
```



---

d. Let's add a smooth line using `loess` (without the confidence intervals) to the previous graph and let this regression line has blue color. Hint: Use `geom_smooth()` (See Visualization slides)

```
ggplot(women, aes(x = height, y = weight, color='red'))+
geom_point(size = 1, pch = 1)+
  labs(y="Weight", x="Height")+
  geom_line(color="black")+
  geom_smooth(method="lm", color="gray",se=FALSE)+
  geom_smooth(method="loess", color="blue", se=FALSE)
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```


Weight
160
150
140
130
120
60
64
68
72
Height
colour
red