R Notebook

This is an [R Markdown](http://rmarkdown.rstudio.com) Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Ctrl+Alt+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Ctrl+Shift+K* to preview the HTML file).

The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.

### Dependency

library(ggplot2)  
library(ggpubr)

## Creating the dataframe

x = c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131, 153, 177, 148, 189, 138, 146, 199, 167, 153, 130)  
y = c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48, 65, 84, 59, 93, 49, 55, 79, 75, 66, 49)  
df = data.frame("X" = x, "Y" = y)  
df

## X Y  
## 1 151 63  
## 2 174 81  
## 3 138 56  
## 4 186 91  
## 5 128 47  
## 6 136 57  
## 7 179 76  
## 8 163 72  
## 9 152 62  
## 10 131 48  
## 11 153 65  
## 12 177 84  
## 13 148 59  
## 14 189 93  
## 15 138 49  
## 16 146 55  
## 17 199 79  
## 18 167 75  
## 19 153 66  
## 20 130 49

## five point summary

summary(df)

## X Y   
## Min. :128.0 Min. :47.00   
## 1st Qu.:138.0 1st Qu.:55.75   
## Median :152.5 Median :64.00   
## Mean :156.9 Mean :66.35   
## 3rd Qu.:174.8 3rd Qu.:76.75   
## Max. :199.0 Max. :93.00

## Creating a linear regression model using the dataframe

fit<- lm(df$Y~df$X)  
summary(fit)

##   
## Call:  
## lm(formula = df$Y ~ df$X)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -14.1573 -1.7267 0.7701 2.6045 6.2102   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -33.55669 8.25032 -4.067 0.000723 \*\*\*  
## df$X 0.63675 0.05213 12.215 3.79e-10 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 4.846 on 18 degrees of freedom  
## Multiple R-squared: 0.8924, Adjusted R-squared: 0.8864   
## F-statistic: 149.2 on 1 and 18 DF, p-value: 3.788e-10

# 1

## Formula for standard deviation is square root of [ (1/(N-1))\*(xi - mu)^2 ]

### Manual calculation

df$X\_xbar = df$X - mean(x)   
df$Y\_ybar = df$Y - mean(y)  
df$X\_xbar\_times\_Y\_Ybar = df$X\_xbar \* df$Y\_ybar  
df$X\_xbar\_sq = (df$X - mean(x))^2  
df$Y\_ybar\_sq = (df$Y - mean(y))^2  
df

## X Y X\_xbar Y\_ybar X\_xbar\_times\_Y\_Ybar X\_xbar\_sq Y\_ybar\_sq  
## 1 151 63 -5.9 -3.35 19.765 34.81 11.2225  
## 2 174 81 17.1 14.65 250.515 292.41 214.6225  
## 3 138 56 -18.9 -10.35 195.615 357.21 107.1225  
## 4 186 91 29.1 24.65 717.315 846.81 607.6225  
## 5 128 47 -28.9 -19.35 559.215 835.21 374.4225  
## 6 136 57 -20.9 -9.35 195.415 436.81 87.4225  
## 7 179 76 22.1 9.65 213.265 488.41 93.1225  
## 8 163 72 6.1 5.65 34.465 37.21 31.9225  
## 9 152 62 -4.9 -4.35 21.315 24.01 18.9225  
## 10 131 48 -25.9 -18.35 475.265 670.81 336.7225  
## 11 153 65 -3.9 -1.35 5.265 15.21 1.8225  
## 12 177 84 20.1 17.65 354.765 404.01 311.5225  
## 13 148 59 -8.9 -7.35 65.415 79.21 54.0225  
## 14 189 93 32.1 26.65 855.465 1030.41 710.2225  
## 15 138 49 -18.9 -17.35 327.915 357.21 301.0225  
## 16 146 55 -10.9 -11.35 123.715 118.81 128.8225  
## 17 199 79 42.1 12.65 532.565 1772.41 160.0225  
## 18 167 75 10.1 8.65 87.365 102.01 74.8225  
## 19 153 66 -3.9 -0.35 1.365 15.21 0.1225  
## 20 130 49 -26.9 -17.35 466.715 723.61 301.0225

# So standard deviation for X  
S\_x = sqrt(sum(df$X\_xbar\_sq)/(length(x)-1))  
S\_x

## [1] 21.32678

# So standard deviation for Y  
S\_y = sqrt(sum(df$Y\_ybar\_sq)/(length(y)-1))  
S\_y

## [1] 14.37569

### Using built-in R functions for standard deviation

print(paste("Sx =",sd(x)))

## [1] "Sx = 21.3267807919378"

print(paste("Sy =",sd(y)))

## [1] "Sy = 14.3756922030137"

# 2

## We know that r = sum((X - xbar)*(Y - ybar)) /√(sum((X - xbar)^2)*sum((Y - ybar)^2)

# let slope = r  
r = sum(df$X\_xbar\_times\_Y\_Ybar)/sqrt(sum(df$X\_xbar\_sq)\*sum(df$Y\_ybar\_sq))  
r

## [1] 0.944644

### Using built-in R functions for correlation coefficient

cor(x, y)

## [1] 0.944644

## We know slope, b = r \* (Sy/Sx)

b = r \* (S\_y/S\_x)  
b

## [1] 0.6367539

### Using built-in R functions for slope

# slope  
fit$coefficients[2]

## df$X   
## 0.6367539

# 3

## We know that intercept, 𝒂= ybar- b \* xbar

a = mean(y)-b\*mean(x)  
a

## [1] -33.55669

### Using built-in R functions for intercept

# intercept  
fit$coefficients[1]

## (Intercept)   
## -33.55669

# 4

## So the linear equation will be y = a + bx

print(paste0("y =",round(a,2),"+",round(b,2),"x"))

## [1] "y =-33.56+0.64x"

# 5

## We know that the value of R-squared is the square of correlation coefficient (r)

r^2

## [1] 0.8923523

### Using built-in R functions for R-squared using linear regression

summary(fit)$r.squared

## [1] 0.8923523

## Scatter plot

#create plot with regression line, regression equation, and R-squared  
ggplot(data=df, aes(x=X, y=Y)) +  
 geom\_smooth(method="lm") +  
 geom\_point() +  
 stat\_regline\_equation(label.x=130, label.y=90) +  
 stat\_cor(aes(label=..rr.label..), label.x=130, label.y=80) + labs( x="X", y="Y")

## `geom\_smooth()` using formula 'y ~ x'

