Linear Statistical Models

Week-2: Graded Assignment

Subjective Assignment: (Manual-grading) **Max. Marks: 30**

NOTE: All the plots should be properly labelled.

1. (a) Generate 1000 discrete random numbers within the range [1,5], and store it in a vector named 'x'. [1 Mark]

Solution:

We can use the following codes to generate 1000 discrete random numbers within the range [1,5].

```
> library(purrr)
> x=rdunif(1000,5,1)
> head(x)
[1] 2 5 1 4 3 4
```

1000 discrete random numbers are stored in vector x and only 6 random numbers have been shown in the output with the help of head() command.

(b) Generate 1000 random numbers from the normal distribution with mean μ and variance σ^2, and store it in a new vector 'y'. [1 Mark]

Note: *You can choose any value of* μ *and* σ.

Solution:

We can generate 1000 random numbers from normal distribution with mean 1 and variance 1 by using the following codes:

```
> y=rnorm(1000,mean=1,sd=1)
> head(y)
[1]  2.3404923  0.9350446  2.0926104  1.8483328 -0.3780834 -0.1084671
>
```

1000 random numbers generated from normal distribution are stored in vector named y and head() command is used to show first 6 values of the output.

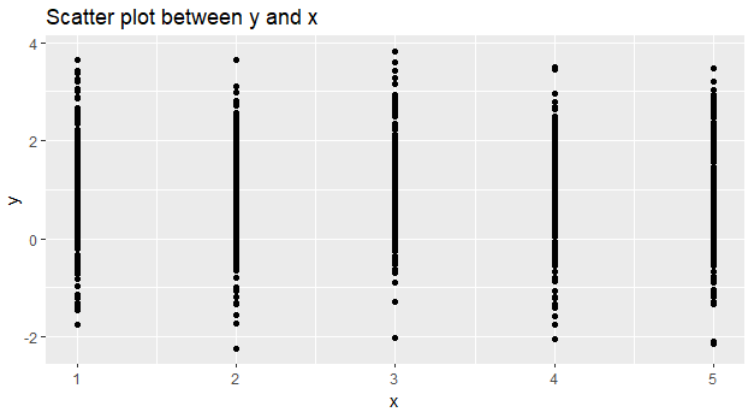
(c) Using 'ggplot' in R, plot the scatter plot between the variables 'y' and 'x'. [1 Mark]

Solution:

The scatter plot between variables 'y' and 'x' by using 'ggplot' in R can be done as follows:

```
> data=data.frame(x,y)
> head(data)
  x          y
1 2  2.3404923
2 5  0.9350446
3 1  2.0926104
4 4  1.8483328
5 3 -0.3780834
6 4 -0.1084671
> library(ggplot2)
> plot=ggplot(data=data)+geom_point(mapping=aes(x=x,y=y))+ggtitle("Scatter
plot between y and x")
> plot
```

Output of scatter plot between the variables 'y' and 'x' as follows:



(d) Based on the visualization of the above plotted scatter plot, categorize the pairs (x, y) into 5 appropriate categories, in the above plotted scatter plot (by representing them in different colors). Comment on the visualization of categories in scatter plot. [3 Marks]

Solution:

It can be easily observe that pairs are already categorize into 5 categories according to the values of x.

Now, we can represent the different categories in different colors by using the following code:

```
data <- data.frame(x,y)
identify_rows <- ((x<1.5) & (x>1.25))|((x <2.5) &
+(x >2.25))|((x<3.25) & (x>3))|((x<4)&(x>3.75))|((x<5)&(x>4.75))
heights = data[identify_rows, ]
ggplot(data = heights)+
  ggtitle('Relation␣on␣categorization')+
  geom_point(mapping=aes(x, y,colour = x))
```

```
scale_colour_viridis_d()
```

2. Explore the site: Click to access.
   Pick any section of your choice which you find interesting and elaborate on it. Also, try it out on any publicly available datasets. [8 Marks]
   Note : Explore it on your own.

3. Use the in-built data set cars in R-software of the library datasets to answer the following questions.

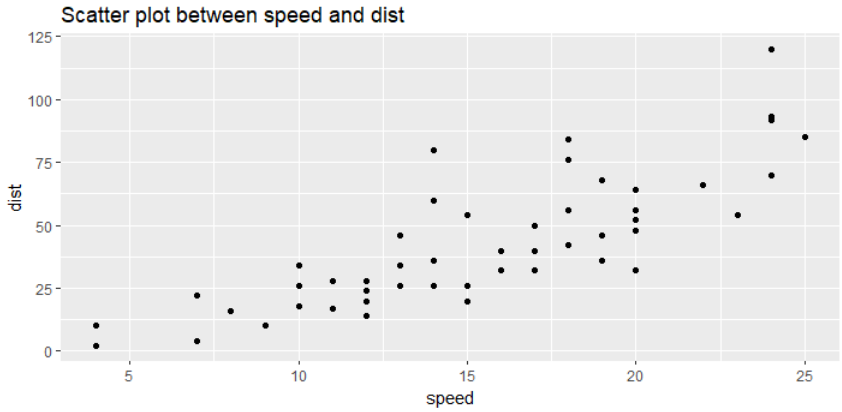
   (a) Give a brief description of the dataset and read the data set as data frame in R. [2 Marks]
   Solution:
   Data set cars is an in-built data set in R - software which consists the speed of cars and the distance taken to stop. This data were recorded in the 1920s.
   The data set is in the format of a data frame with 50 observations and 2 columns.
   We can read the data set as data frame by using following code:

```
> car_data=data.frame(cars)
> head(car_data)
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10
```

   head() has been used to display the first 6 rows of the data set.

   (b) Using 'ggplot' in R, plot the scatter plot between the variables 'dist' and 'speed'. On visualizing the scatter-plot, comment on the relationship between 'dist' and 'speed'. [2 Marks]
   Solution:
   Scatter plot between the variables 'dist' and 'speed' can be plotted in 'ggplot' by using the following code:

```
> Car_plot = ggplot(data = car_data) + geom_point(mapping = aes
(x=speed, y= dist))+ggtitle("Scatter plot between speed and dis
t")
> Car_plot
```



From the above scatter plot, we can clearly observe that as the speed of car increases so does the stopping distance, i.e., there is a linear relationship between the variables 'speed' and 'dist'. Also, it is clear that there are few outliers in the data set.

(c) Create and plot 200 random linear models for the 'cars' data set ('dist' on 'speed'). Also, find the mean squared error for each of the fitted models. [3 Marks]

Solution:

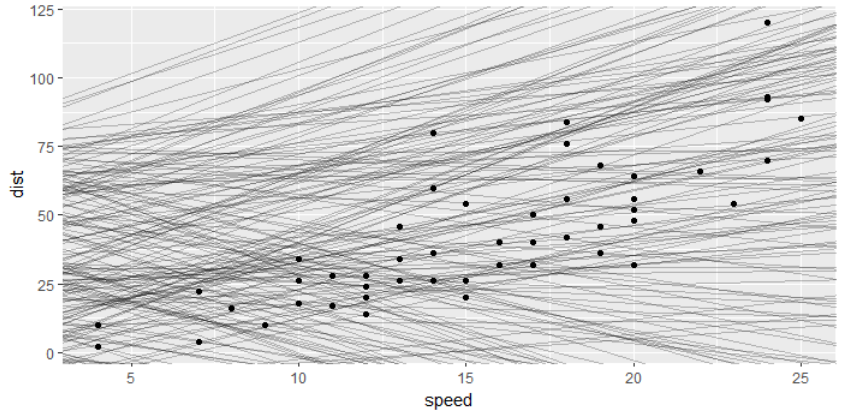
We can create 200 random linear models for the 'cars' data set by creating two random vectors $a1$ and $a2$ for intercept and slope of the model respectively.

```
> a1=runif(200, -20, 80)
> a2=runif(200, -5, 5)
```

Now, 200 random linear models for the 'cars' data set can be generated and plotted by using the codes as follows:

```
> random_models=data.frame(a1,a2)
> models_plot=ggplot(data=car_data,aes(x=speed,y=dist))+geom_abli
ne(aes(intercept=a1, slope = a2), data = random_models, alpha =
1/4) + geom_point()
> models_plot
```

To find the mean squared error for each of the fitted models, we can use the following codes:




```
> models=function(a, data){
+ a[1] + data$speed * a[2]
+ }
> mse = function(mod, data){
+ diff = data$dist - models (mod, data)
+ sqrt(mean(diff^2))
+ }
> mse_models=numeric(length=length(a1))
> for ( i in 1:length(a1)){
+ model = models(c(a1[i],a2[i]), car_data)
+ diff=car_data$dist - model
+ mse_models[i] = sqrt(mean(diff^2))
+ }
> mse_random_models = data.frame(a1,a2,mse_models)
> head(mse_random_models)
          a1         a2 mse_models
1  -5.347813  4.277275   23.19610
2  53.985142  1.278348   36.90607
3 -12.261781 -4.860758  138.81996
4  17.114460 -3.925218   96.79307
5  72.214858 -2.419841   37.37890
6  34.178133 -0.564191   32.96963
```

(d) From the above fitted 200 models, select the best one and write down its equation. Also, mention the reason why you think the selected model is best among the fitted 200 models. [3 Marks]

Solution:

To find the best model, we need to find the coefficients of the model that has least squared error which can be done by using the following codes:

```
> best_model = optim(c(0,0),mse,data= car_data)
> best_model$par
[1] -17.586668   3.932742
> mse(c(best_model$par[1],best_model$par[2]),car_data)
[1] 15.06886
```

Equation of the best model is: dist = -17.586668 + 3.932742 $\times$ speed.

(e) Use the inbuilt function lm() in R, to find the equation of best fitted linear model ('dist' on 'speed') for the 'cars' dataset. [1 Mark]

Solution:

Execution of lm() function for 'cars' data set is as follows:

```
> linear_model=lm(dist~speed, data = car_data)
> linear_model

Call:
lm(formula = dist ~ speed, data = car_data)

Coefficients:
(Intercept)        speed
    -17.579        3.932
```

Best model equation by using lm() function is: dist = -17.579 + 3.932 × speed, which is same as best model obtained in part (d).

(f) Using 'ggplot' in R, plot the linear models obtained in part (*d*) and (*e*) in the same scatter plot (plotted in part (*b*)). [2 Marks]

Solution:

```
ggplot(data=car_data, aes(x=speed, y=dist))+
  geom_point()+
  geom_abline(intercept = best_model$par[1], slope = best_model$par[2],
  geom_smooth(method = "lm",se=FALSE, color='blue')
```

Plots can be obtained by executing the above command.

Note: You can modify the above commands to obtain a better version of the plots.

(g) For each of the fitted models (part (d) and (e)), plot the 'residuals' vs 'speed' using 'ggplot' in R. Comment on the obtained plots. [3 Marks]

Solution:

Computing and plotting the residuals of the fitted model in part (*d*):

```
y_pred<- best_model$par[1] + car_data$speed*best_model$par[2]
residual<- rep(0, length(y_pred))
for (i in 1:length(y_pred)){
  residual[i]=y_pred[i]-car_data$dist[i]
}
residual_1 = data.frame(residual, car_data$speed)
ggplot(data= residual_1, aes(x=residual, y=car_data$speed))+
  geom_point()
```

Computing and plotting the residuals of the fitted model in part (*e*):

```
y_pred<- y_pred<- linear_model$coefficients[1] +
car_data$speed*linear_model$coefficients[2]
residual<- rep(0, length(y_pred))
```

```
for (i in 1:length(y_pred)){
  residual[i]=y_pred[i]-car_data$dist[i]
}
residual_1 = data.frame(residual, car_data$speed)
ggplot(data= residual_1, aes(x=residual, y=car_data$speed))+
  geom_point()
```