Shopify Technical Challenge Submission (link)

Ahnaf Ryan

16/09/2021

R Markdown

Question 1

```
shopify<-read.csv("shopify.csv") ##imported data, note: name has been changed for convinience
#head(shopify) ## for checking the first 6 rows of the dataframe
mean(shopify$order_amount) ## cross checking the given AOV
```

```
## [1] 3145.128
```

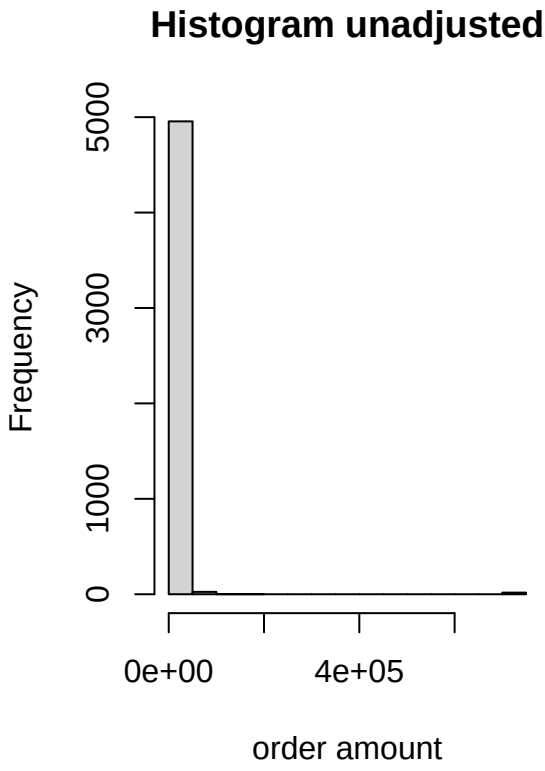
```
summary(shopify$order_amount) ## summary statistics to find out that
```

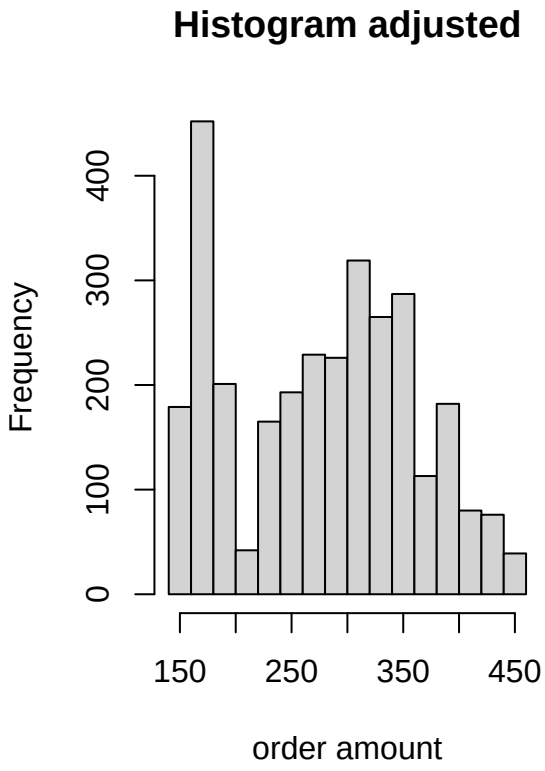
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      90     163     284    3145     390  704000
```

```
par(mfrow=c(1,2))
hist(shopify$order_amount, main = "Histogram unadjusted",xlab = "order amount")
correct_amount<-shopify$order_amount[which(shopify$order_amount>=quantile(shopify$order_amount,0.2) &  s
length(correct_amount)/length(shopify$order_amount)
```

```
## [1] 0.6096
```

```
hist(correct_amount, main = "Histogram adjusted",xlab = "order amount")
```






```
mean(correct_amount)
```

```
## [1] 278.5308
```

a. The error with the analysis was not considering the distribution of the order amount data which led to the false believe that the mean is 3145. The distribution is extremely right skewed with shops that had order amounts over few thousand dollars. Hence, if we consider an average of 3145.13, it will be false over-estimate of the AOV.

b. In order to tackle the problem, 80% trimmed mean is considered so that a better estimate of AOV is provided. In particular, some of the high order amounts, considered as outliers are removed. We could have provided the median metric however with trimmed mean analysis, we can perform trimmed t-tests if we have any hypothesis in next steps.

c. The AOV found in our analysis is $278.5 which is a better representation as we can see from our adjusted histogram below.

Question 2

a. 54

The idea was to group the order numbers by shipper name with condition that shippername is speedy express. In order to do that, joining the shipper table and orders was required.

SELECT count(a.OrderID) as count_order,b.shipperName FROM Orders a inner JOIN sHIPPERS b ON a.SHIPPERID=b.SHIPPERID where b.shipperName like 'Speedy Express' group by b.shippername ;

b. lastname is Peacock with 40 orders

The idea was to do two same select statements where the latter would act like a max condition under the having function

SELECT count(a.OrderID) as countorder, b.lastName FROM Orders a INNER JOIN employees b ON a.employeeid=b.employeeid group by b.lastname having count(a.OrderID)=(SELECT max(count_max) FROM (SELECT count(c.OrderID) as count_max, d.lastName FROM Orders c INNER JOIN employees d ON c.employeeid=d.employeeid group by d.lastname))

c. Product Name is Gorgonzola Telino with 5 products

The idea for b and c is similar

SELECT a.productname,count(a.productid) as product_count FROM (((Products a inner join orderdetails b on a.productid=b.productid) inner join orders c on c.orderid=b.orderid) inner join customers d on c.customerid=d.customerid) where d.country='Germany' group by a.productname having count(a.productid)=(select max(count_max) from (SELECT a.productname,count(a.productid) as count_max FROM (((Products a inner join orderdetails b on a.productid=b.productid) inner join orders c on c.orderid=b.orderid) inner join customers d on c.customerid=d.customerid) where d.country='Germany' group by a.productname))