Answers 5 - Interactive plots

Osama Mahmoud

Getting started

Load the `movies` dataset from the `BristolVis` R package. The data can be called and viewed using:

```
data(bmov, package = "BristolVis")
head(bmov)
```

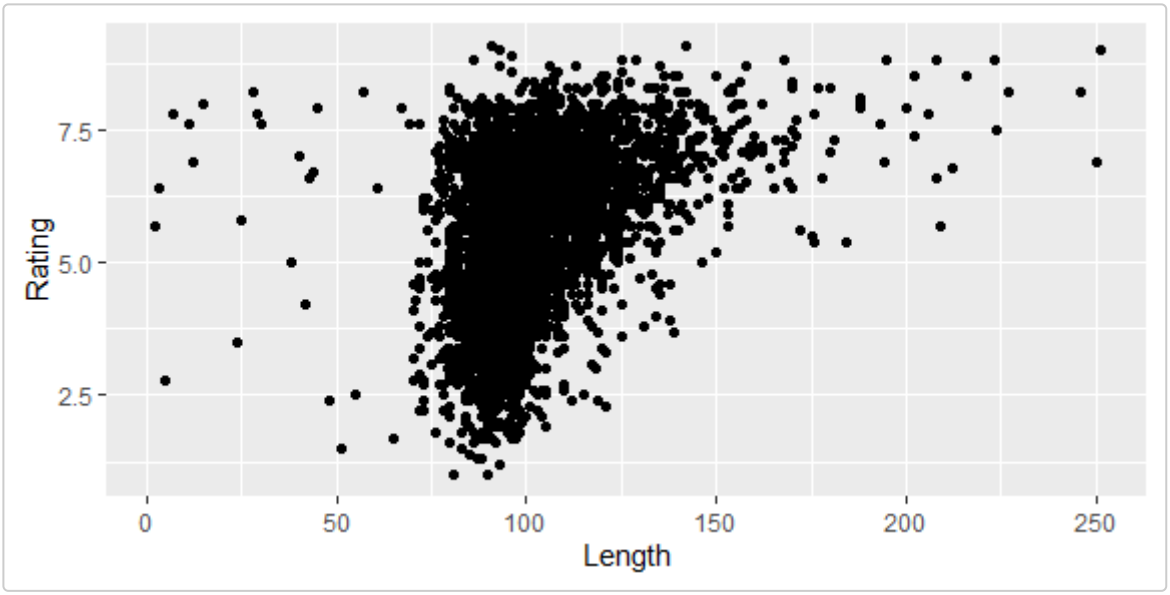
```
##                     Title Year Length   Budget Rating Votes   r1  r2  r3
## 1     A.k.a. Cassius Clay 1970     85       -1    5.7    43  4.5 0.0 4.5
## 2                     AKA 2002    123       -1    6.0   335 24.5 4.5 4.5
## 3 AVP: Alien Vs. Predator 2004    102 45000000    5.4 14651  4.5 4.5 4.5
## 4                 Abandon 2002     99 25000000    4.7  2364  4.5 4.5 4.5
## 5               Abendland 1999    146       -1    5.0    46 14.5 4.5 4.5
## 6              Aberration 1997     93       -1    4.8   149 14.5 4.5 4.5
##     r4   r5   r6   r7   r8  r9  r10  mpaa Action Animation Comedy Drama
## 1 14.5  4.5 24.5 14.5 14.5 4.5 14.5    PG      0         0      0     0
## 2  4.5  4.5  4.5 14.5 14.5 4.5 14.5     R      0         0      0     1
## 3  4.5 14.5 14.5 14.5  4.5 4.5  4.5 PG-13      1         0      0     0
## 4 14.5 14.5 14.5 14.5  4.5 4.5  4.5 PG-13      0         0      0     1
## 5  4.5  4.5  4.5 14.5 14.5 4.5 24.5     R      0         0      0     0
## 6 14.5 14.5 14.5 14.5  4.5 4.5  4.5     R      0         0      0     0
##   Documentary Romance Short
## 1           1       0     0
## 2           0       0     0
## 3           0       0     0
## 4           0       0     0
## 5           0       0     0
## 6           0       0     0
```

Scatter plots (15 minutes)

Let's start with some simple scatter plots using the `bmov` data:

1. Plot length Vs. rating using the advanced graphics package (`ggplot2`)

```
require(ggplot2)
(G = ggplot(bmov, aes(Length, Rating)) + geom_point())
```



2. Use the `cut` function to generate a categorical form of the variable Year with sensible cutpoints.

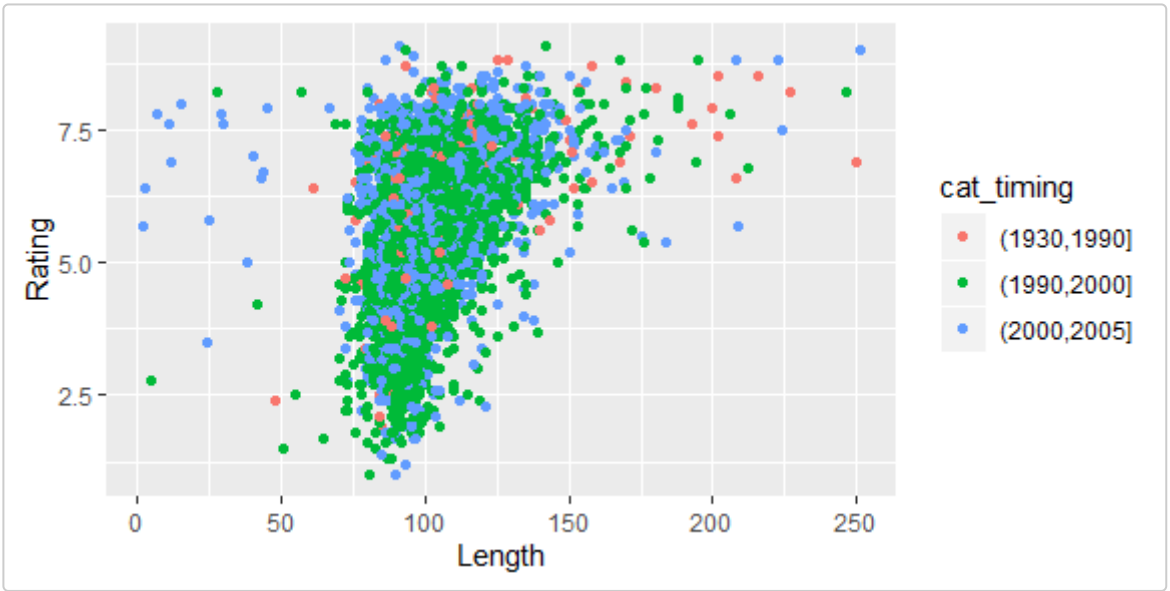
```
summary(bmov$Year)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1934    1997    1999    1998    2002    2005
```

```
bmov$cat_timing = cut(bmov$Year, breaks = c(1930, 1990, 2000, 2005))
```

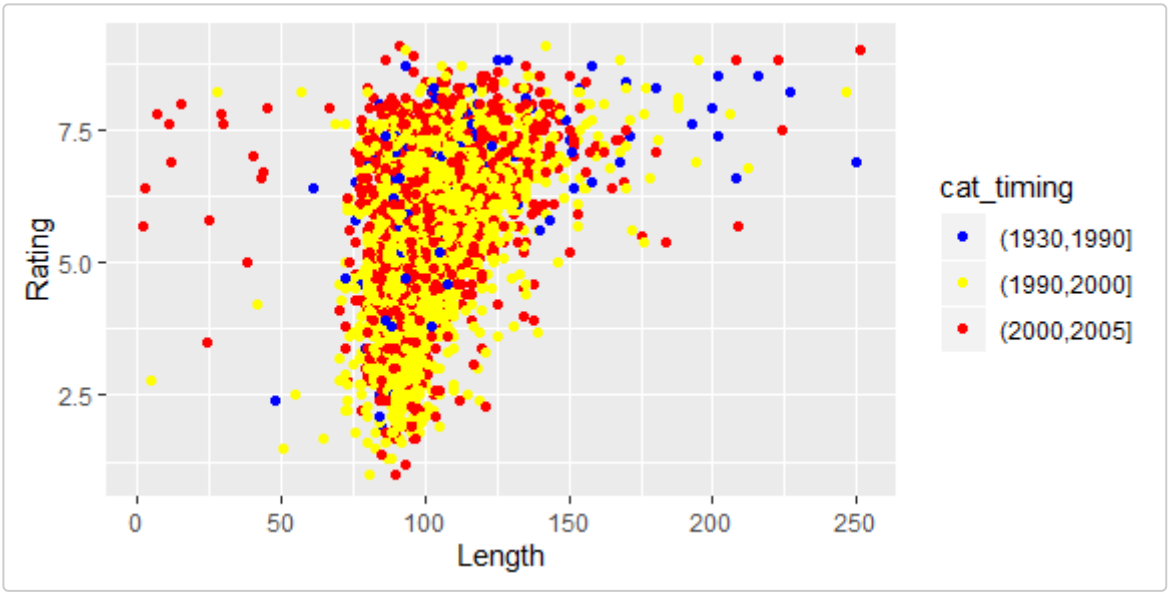
3. Plot `length` Vs. `rating` such that points are coloured using categories of your generated timing

```
(G = ggplot(bmov, aes(Length, Rating, color = cat_timing)) + geom_point())
```



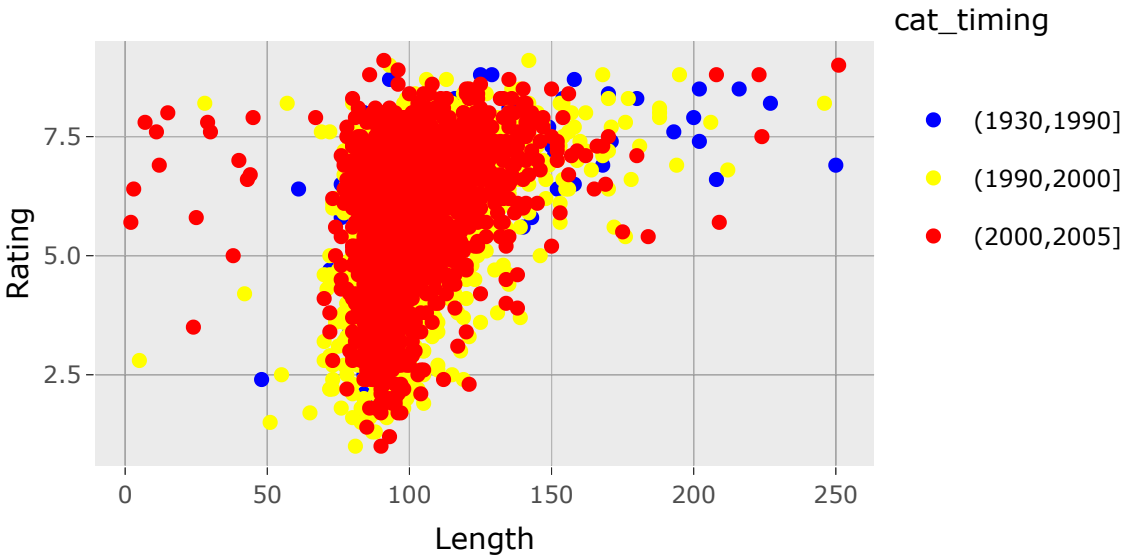
4. The default colors of the previous plot are terrible! use your own color selections to generate a better plot.

```
(G = G + scale_color_manual(values = c("blue", "yellow", "red")))
```



5. Generate an interactive plot of the plot in (4) using the plotly package and name it Fig_scatter.

```
require(plotly)
(Fig_scatter = ggplotly(G))
```



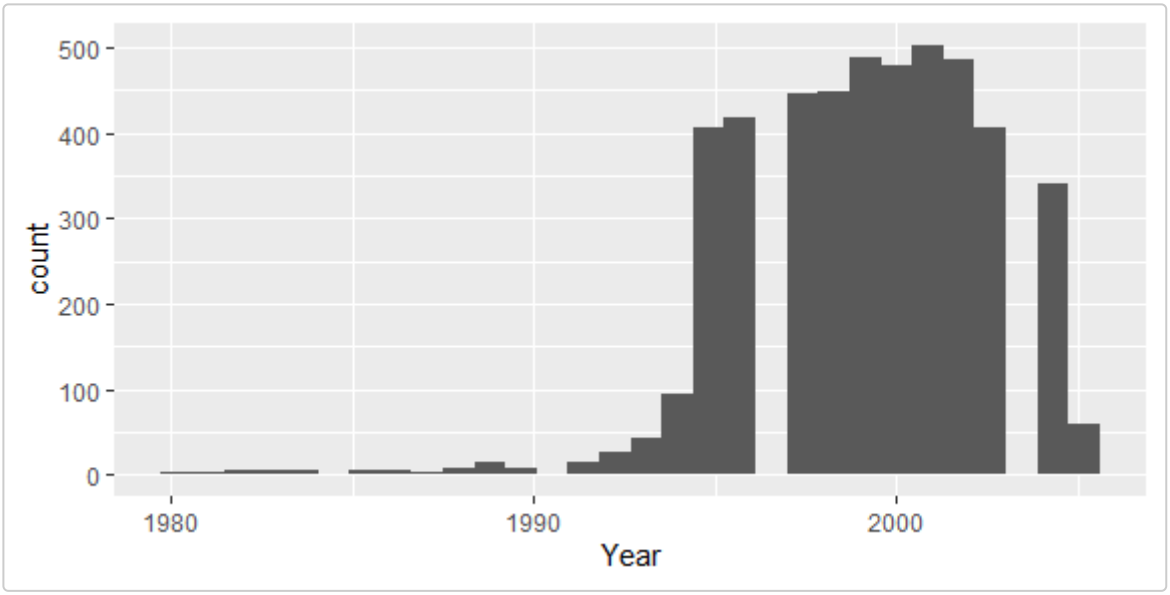
6. Try to zoom in using your mouse by box selection to explore further detailed information.
7. Save your interactive plot as an html file.

```
htmlwidgets::saveWidget(Fig_scatter, "Fig_scatter.html")
```

Histograms (10 minutes)

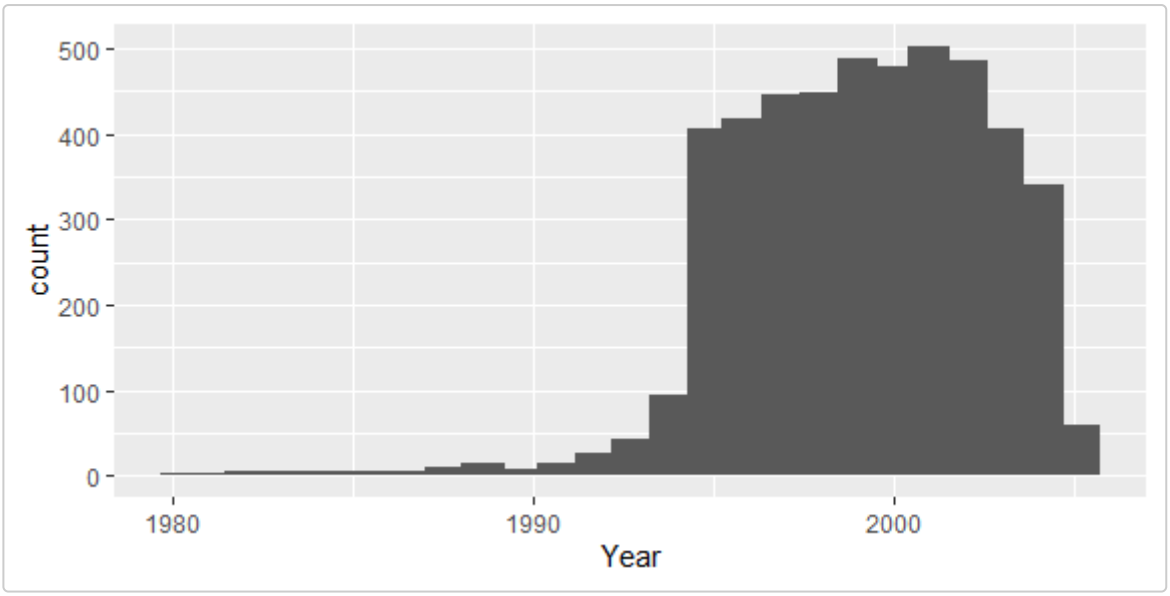
1. Use the ggplot2 to plot a histogram of the movie years restricted to data after 1980.

```
(G = ggplot(bmov[bmov$Year>=1980,], aes(Year)) + geom_histogram())
```



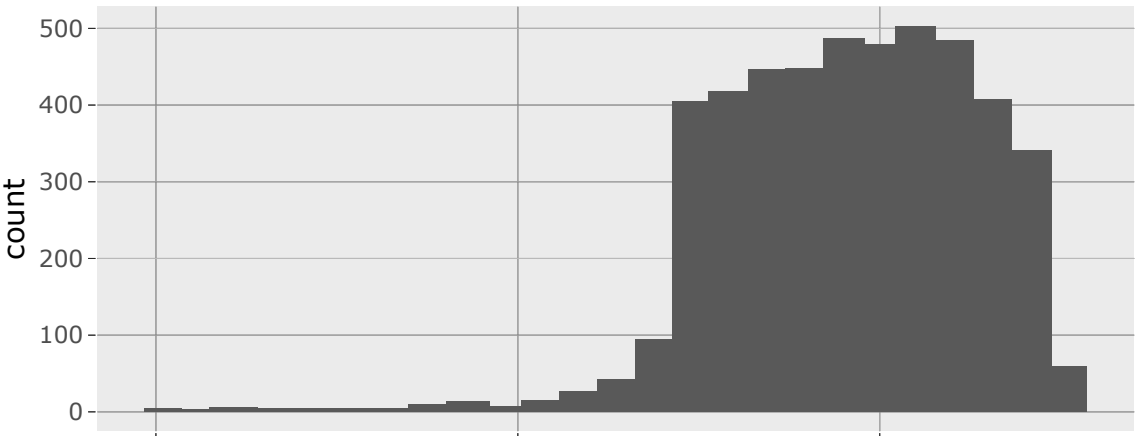
2. Produce the same plot as in (1), but set the number of bins to 25.

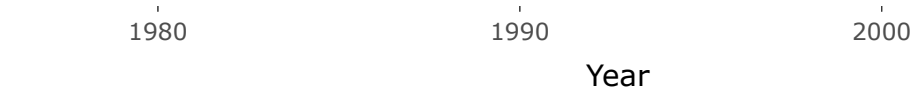
```
(G = G + geom_histogram(bins = 25))
```



3. Generate an interactive plot of the plot in (2) using the plotly package and name it `Fig_hist`.

```
(Fig_hist = ggplotly(G))
```





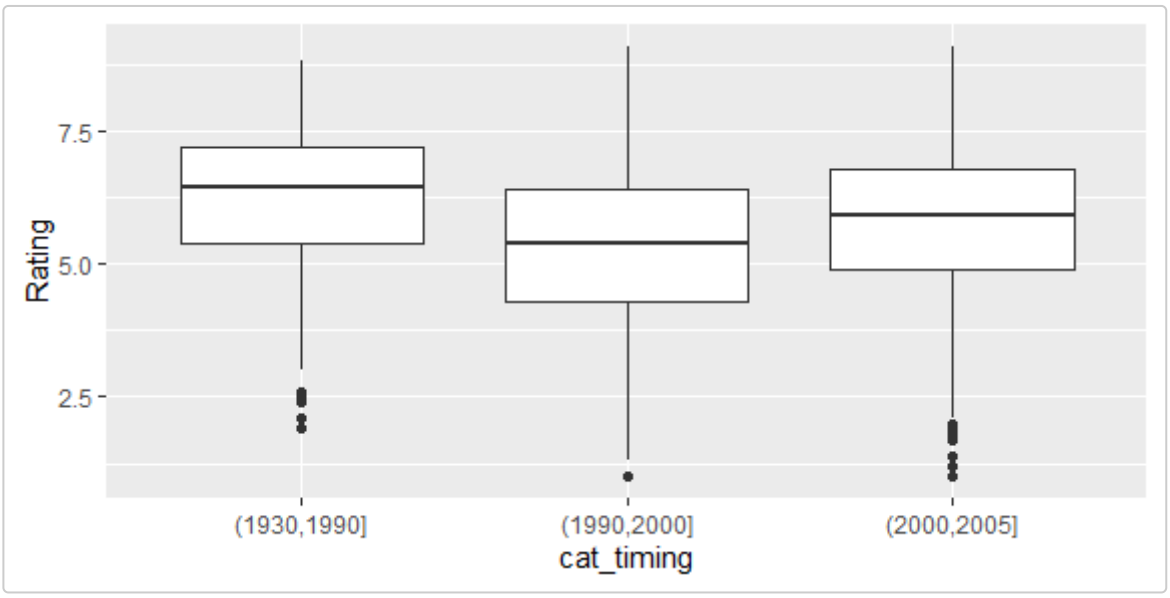
4. Try to zoom in using your mouse by box selection to explore further detailed information and reset the plot (double-click).
5. Save your interactive plot as an html file.

```
htmlwidgets::saveWidget(Fig_hist, "Fig_hist.html")
```

Boxplots (10 minutes)

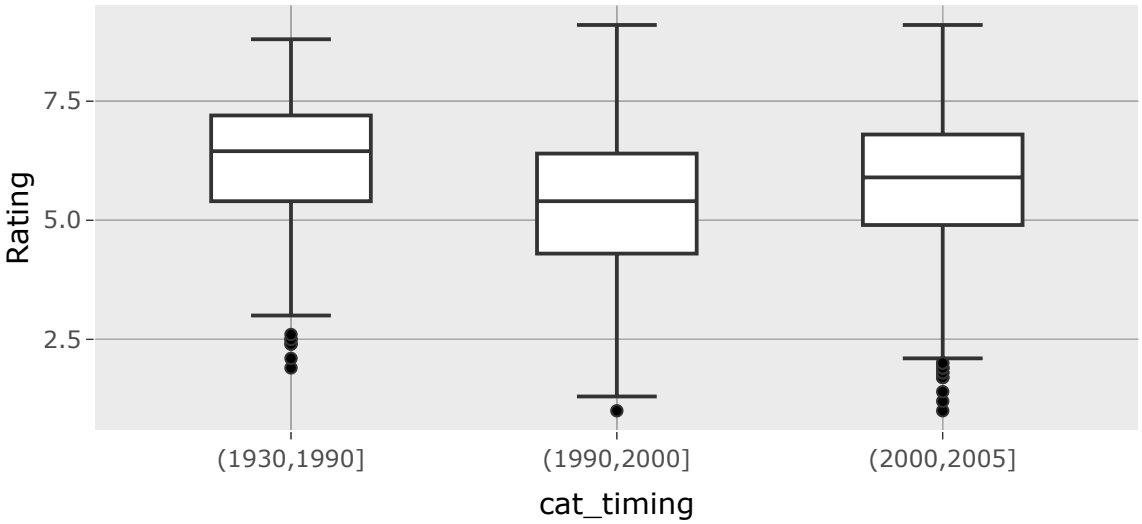
1. Generate a boxplot for the ratings data by generated categories of production timing using ggplot2.

```
(G = ggplot(bmov, aes(x=cat_timing, y =Rating)) + geom_boxplot(aes(group = cat_timing)))
```



2. Try generating a similar interactive boxplot.

```
(Fig_box = ggplotly(G))
```



3. save the interactive plot to an html file.

```
htmlwidgets::saveWidget(Fig_box, "Fig_box.html")
```

Corrlation matrix (15 minutes)

1. Use the built-in `iris` data to compute a correlation matrix and correlation p-values for the continous (first four) variables.

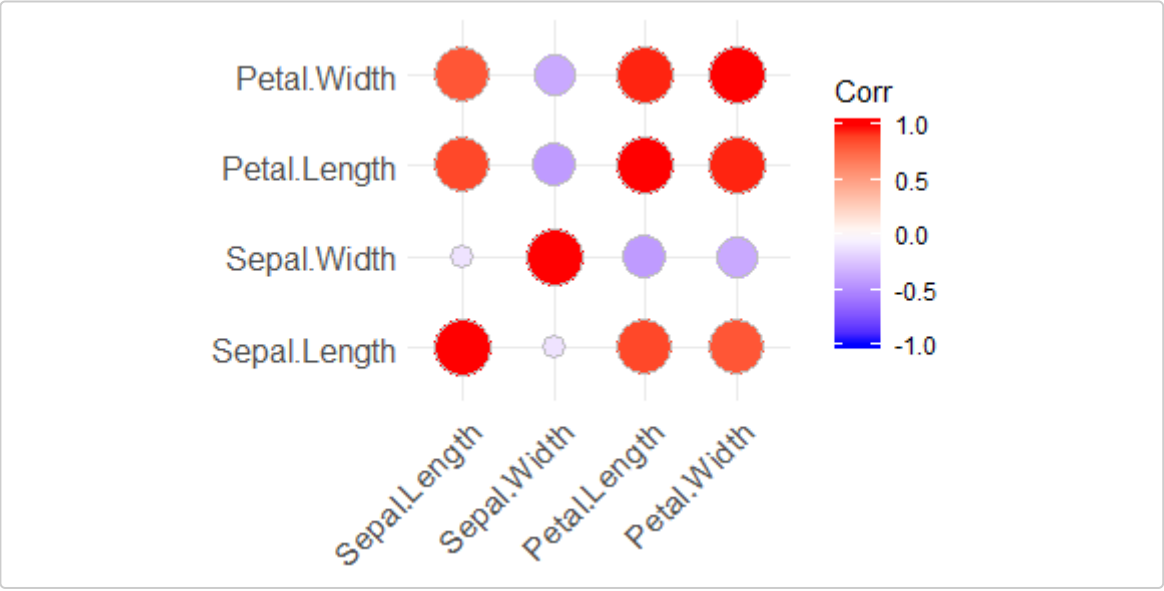
```
data(iris)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
data_cont = iris[,1:4]
Corr = cor(data_cont)
require(ggcorrplot)
corr.p = cor_pmat(data_cont)
```

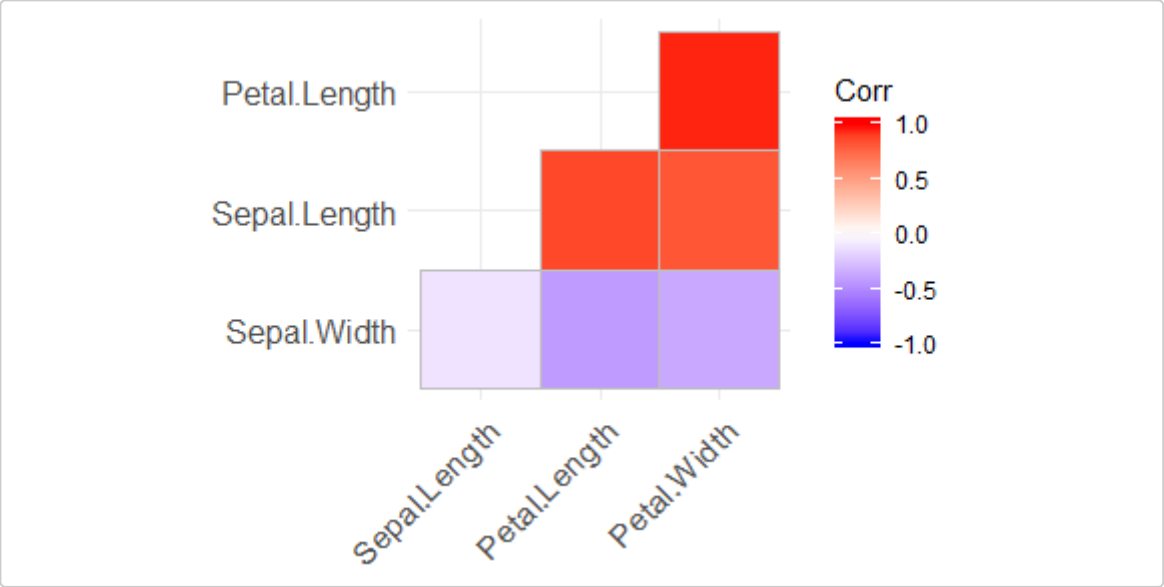
2. Visualize the correlation matrix using the method = “circle”.

```
(G = ggcorrplot(Corr, method = "circle"))
```



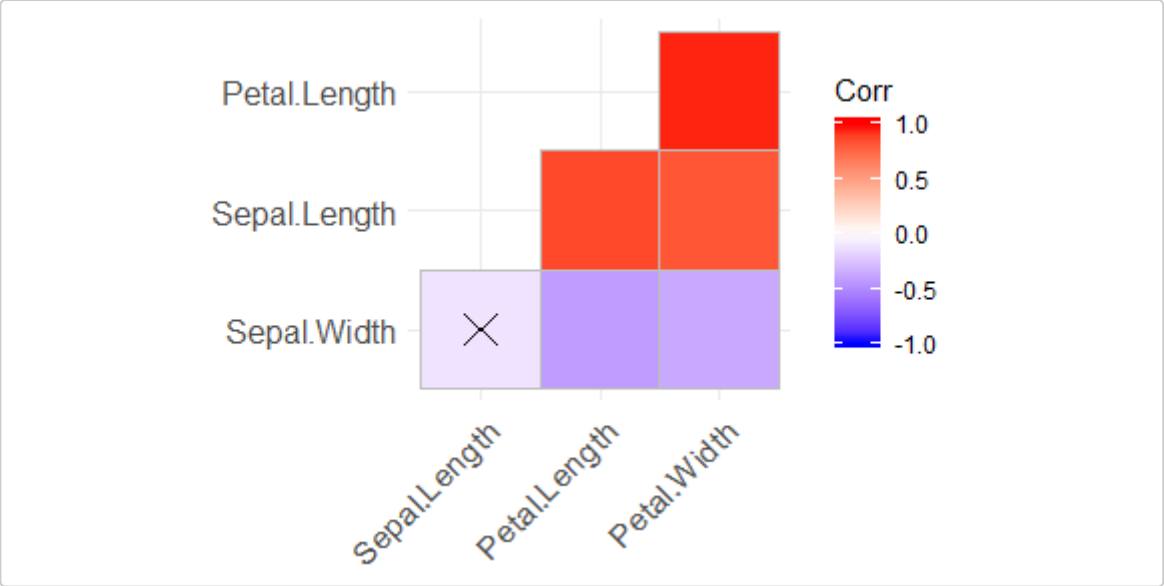
3. Show the lower triangle using hierarchical clustering and square method rather than circle.

```
(G = ggcorrplot(Corr, hc.order = TRUE, type = "lower"))
```



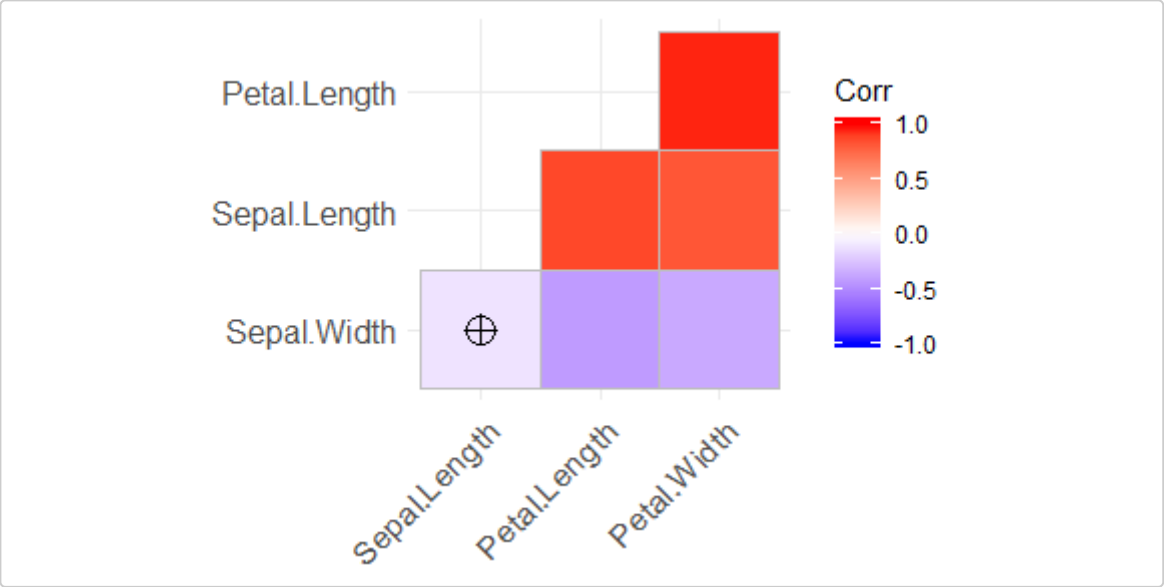
4. Use correlation significance level 0.01 to highlight the non-significant coefficient.

```
(G = ggcorrplot(Corr, hc.order = TRUE, type = "lower", p.mat = corr.p, sig.level = 0.01))
```



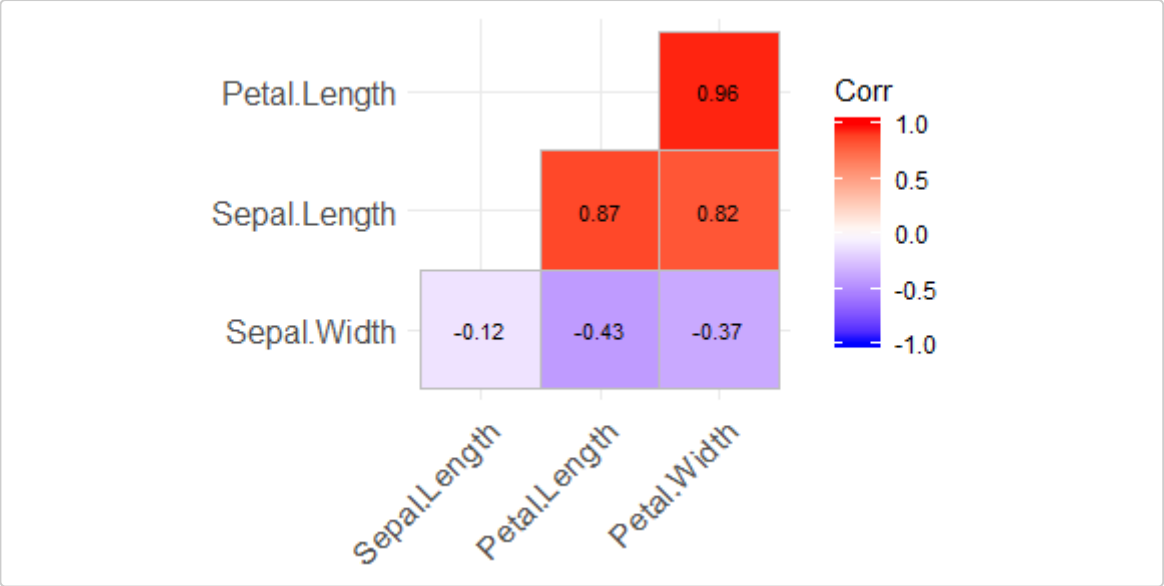
5. Use different shape - rather than the cross - to highlight the non-significant coefficient (use help of the function ggcorrplot by typing: `?ggcorrplot` to find out).

```
(G = ggcorrplot(Corr, hc.order = TRUE, type = "lower", p.mat = corr.p, sig.level = 0.01, pch =
10))
```



6. Add coeficient values on the plot in (3)

```
(G1 = ggcorrplot(Corr, hc.order = TRUE, type = "lower", lab = TRUE, lab_size = 3))
```



7. Produce interactive plot of the plot in (6).

```
Fig_cor = ggplotly(G1)
```

8. Save the interactive plot in as html.

```
htmlwidgets::saveWidget(Fig_cor, "Fig_cor.html")
```