Assignment

Zarrin Sovha Khan

```
#readxl is used in R to import data from Excel files efficiently
library(readxl)
Yield_Performance <- read_excel("Yield Performance.xlsx")
View(Yield_Performance)
#This function is use for displaying data

Yield_Performance$GY
```

```
##  [1] 4.300000 5.680000 5.550000 2.910000 3.100000 3.160000 2.690000
1.700000
##  [9] 3.570000 4.800000 4.370000 4.980000 4.390000 4.050000 3.830000
4.110000
## [17] 4.050000 3.940000 4.760000 4.240000 3.640000 3.310000 4.400000
3.731552
## [25] 4.160000
```

```
#This refers to accessing the GY column from a data frame named
Yield_Performance.

Yield_Performance[5,]
```

```
## # A tibble: 1 × 9
##   SL    Env      Gen      GY    GD    PH    TN    PN    GP
##   <chr> <chr>    <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 S5    Barishal G5      3.1   127   108     9     9    79
```

```
#This is used to access the fifth row of the Yield_Performance data frame

Yield_Performance[5,4]
```

```
## # A tibble: 1 × 1
##      GY
##   <dbl>
## 1   3.1
```

```
#This is used to access the value at the 5th row and 4th column of the
Yield_Performance data frame.

Yield_Performance[,4]
```

```
## # A tibble: 25 × 1
##       GY
##    <dbl>
##  1  4.3
##  2  5.68
##  3  5.55
##  4  2.91
```

```
##  5  3.1
##  6  3.16
##  7  2.69
##  8  1.7
##  9  3.57
## 10  4.8
## # ℹ 15 more rows

#This is used to access all the values in the 4th column of the
Yield_Performance
```

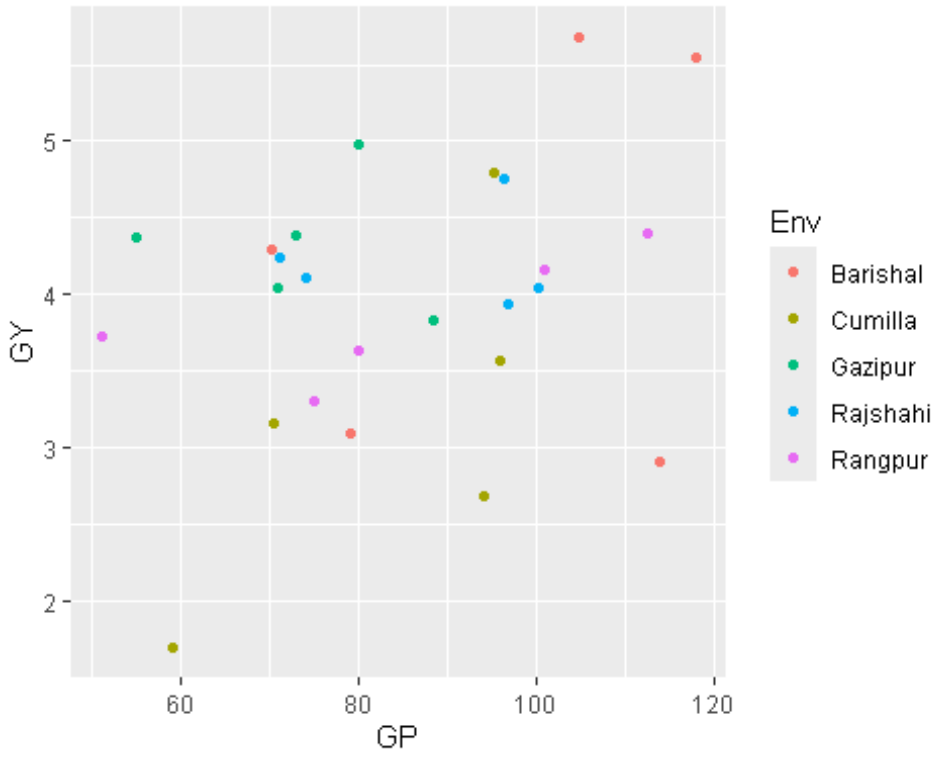
Title

Sub-title

Sub-sub-title

Body

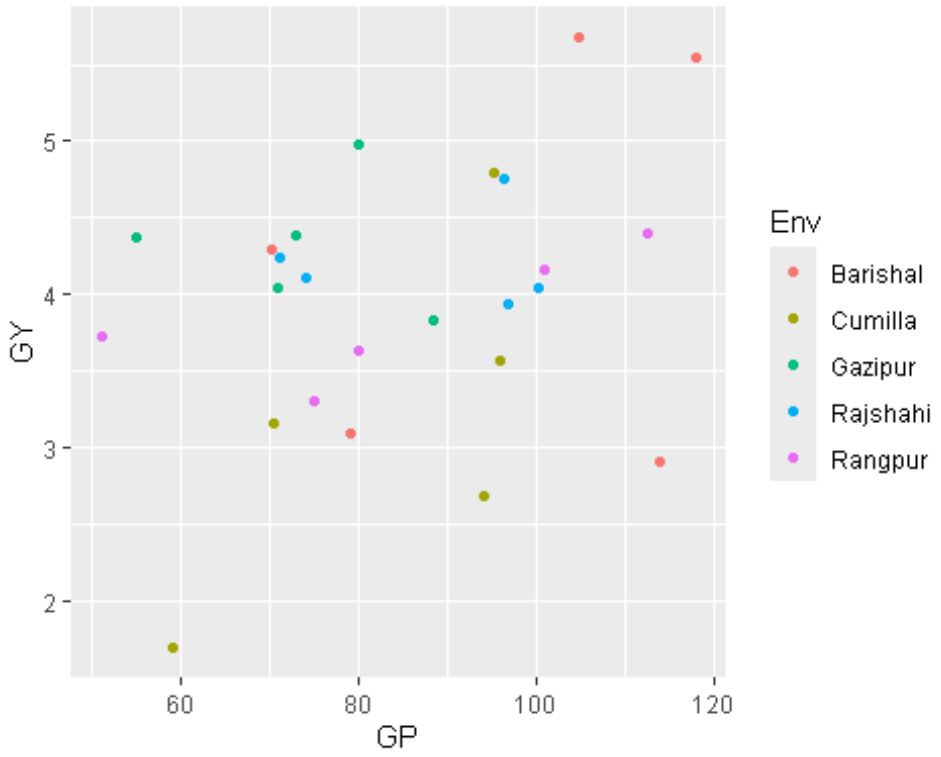
```
library(ggplot2)
#This is used to load the ggplot2 package, which is one of the most popular
and powerful tools for creating advanced and customizable data
visualizations. By calling this command, we are making the functions and
features of the ggplot2 package.this package based on the "Grammar of
Graphics," which allows you to create complex and aesthetically pleasing
plots by building them layer by layer.

ggplot(Yield_Performance,aes(x=GP,y=GY, colour = Env))+
  geom_point()
```



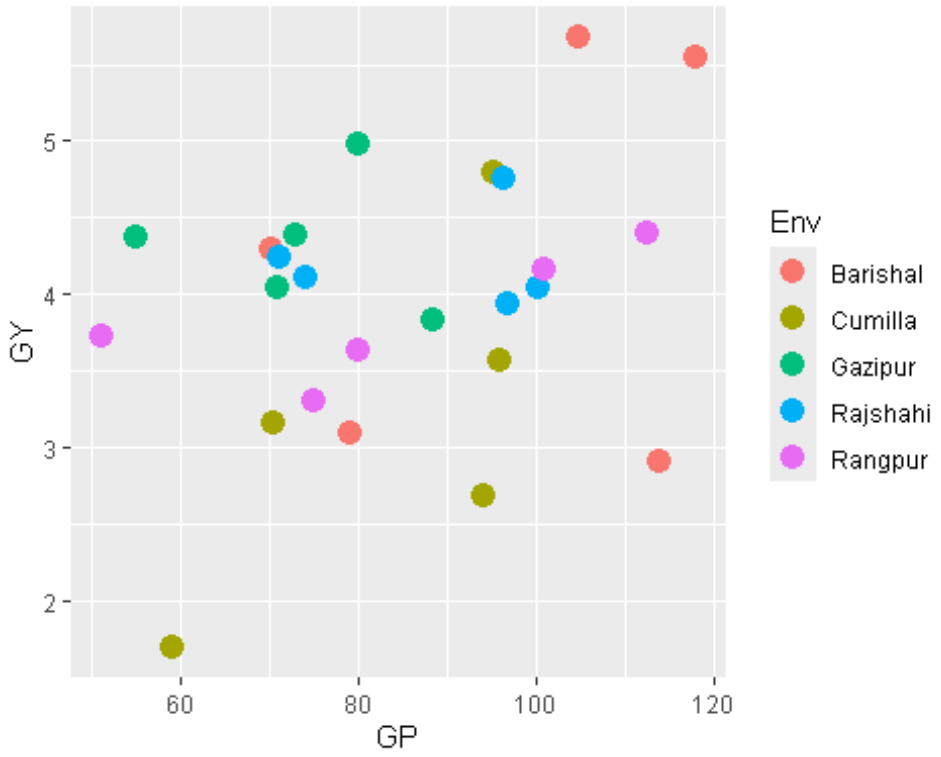

```
#This code generates a scatter plot with GP on the x-axis, GY on the y-axis,
and different colors for points based on the Env variable.

Yield_Performance[4,3]

## # A tibble: 1 × 1
##   Gen
##   <chr>
## 1 G4

#Here Yield_Performance[4,3] accesses the value at the intersection of the
4th row and 3rd column of the Yield_Performance data frame.
```

Scatter plot

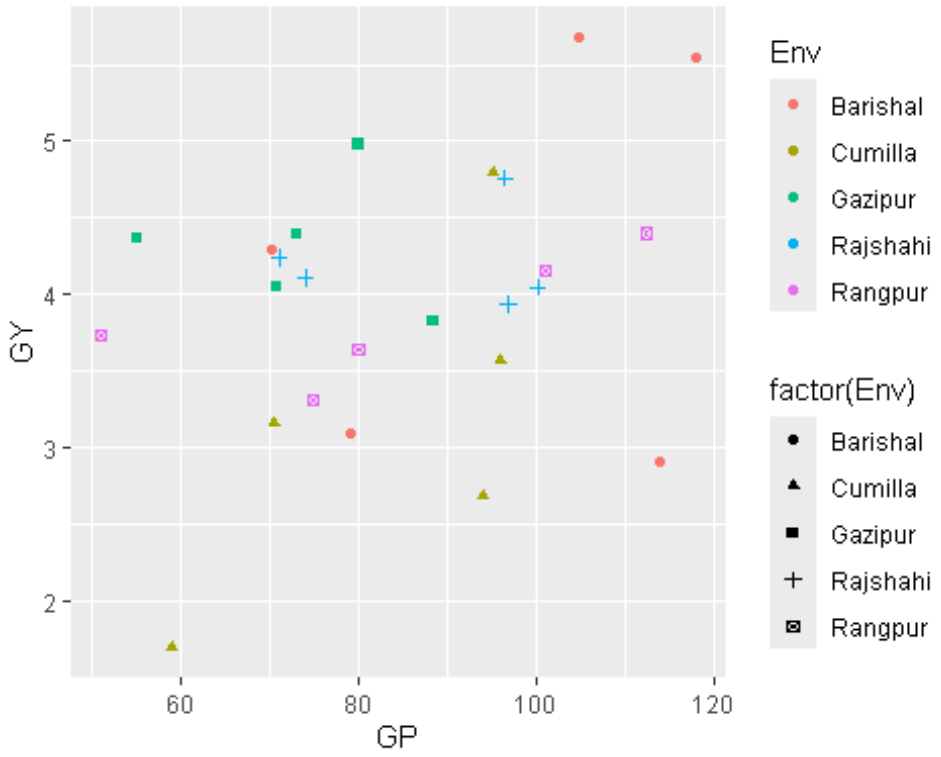
```
ggplot(Yield_Performance,aes(x=GP,y=GY,colour = Env))+geom_point()
```



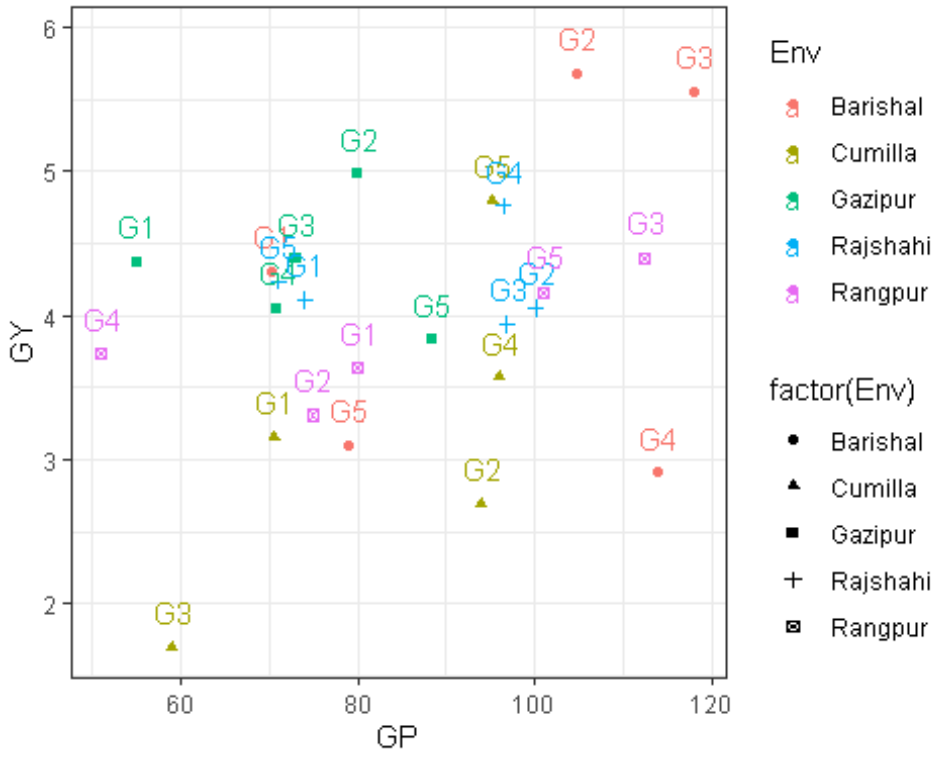

```
#This code generates a scatter plot with GP on the x-axis, GY on the y-axis,
and colors the points based on the Env variable, allowing for differentiation
between different environments

Yield_Performance[20,4]

## # A tibble: 1 × 1
##      GY
##   <dbl>
## 1  4.24

ggplot(Yield_Performance,aes(x=GP,y=GY,colour = Env))+geom_point(size=4)
```



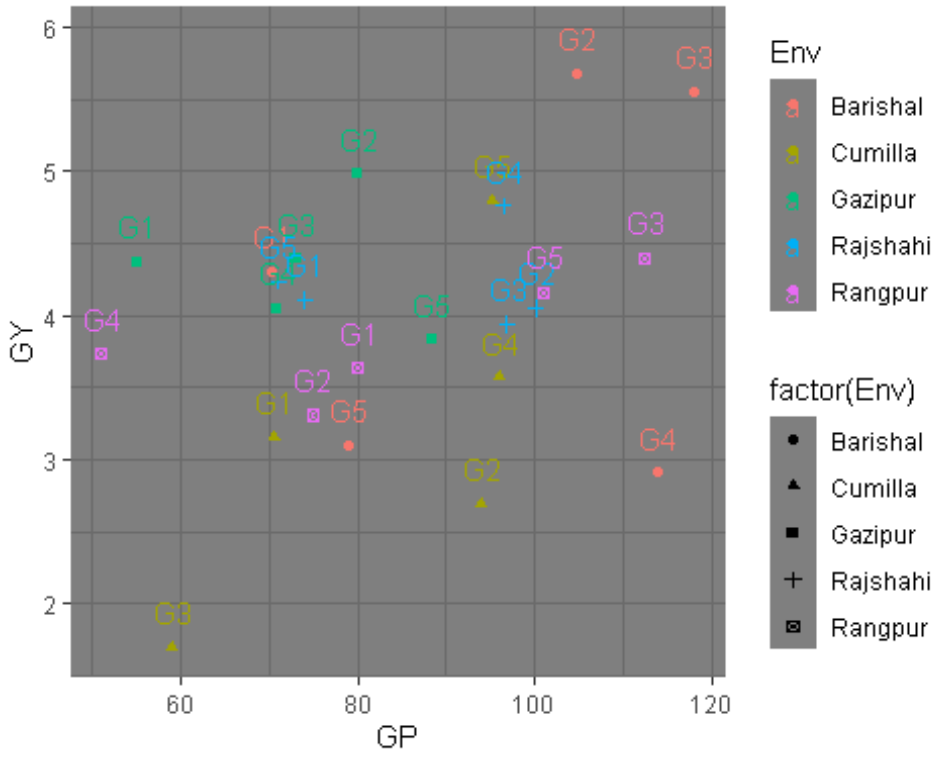

```
#This The code generates a scatter plot with GP on the x-axis, GY on the y-axis, and colors the points based on the Env variable. The size=4 argument makes the points larger for better visibility.

ggplot(Yield_Performance,aes(x=GP,y=GY,colour = Env))+geom_point(aes(shape =factor(Env)))
```



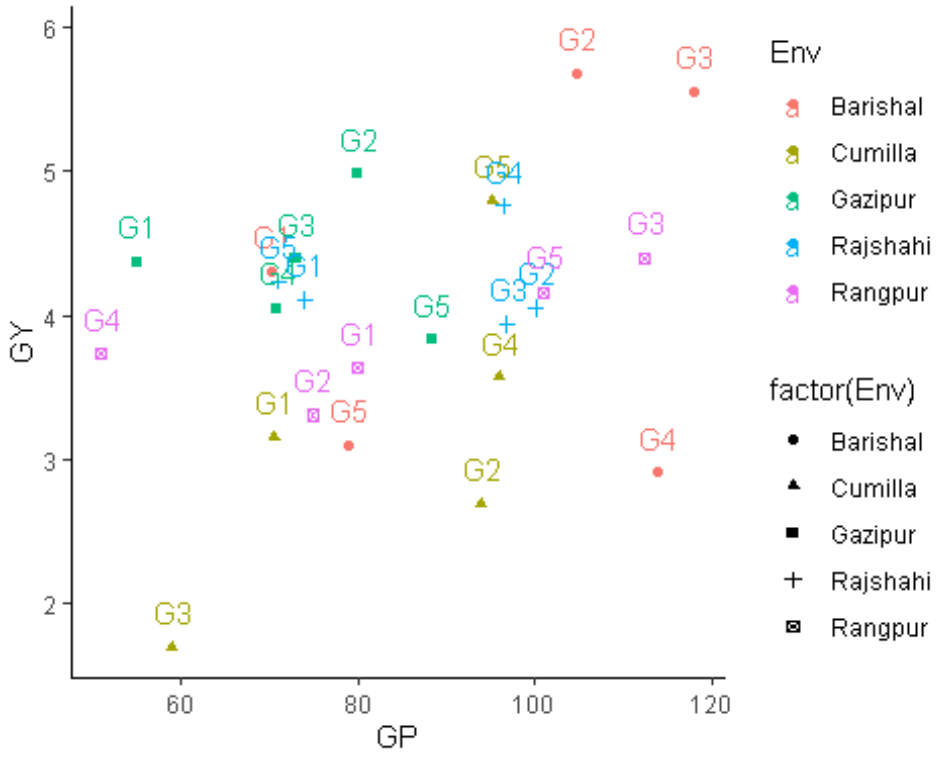

```
#This code generates a scatter plot with GP on the x-axis, GY on the y-axis,
and points colored and shaped according to the Env variable. This enhances
the visualization by allowing differentiation based on both color and shape.

ggplot(Yield_Performance,aes(x=GP,y=GY,colour = Env))+geom_point(aes(shape
=factor(Env)))+
  geom_text(label=Yield_Performance$Gen, nudge_x=.25,nudge_y =.25
)+theme_bw()
```




```
#This code generates a scatter plot with GP on the x-axis, GY on the y-axis, and points colored and shaped by the Env variable. It also labels each point with the Gen column values and uses a clean theme for better visibility

ggplot(Yield_Performance,aes(x=GP,y=GY,colour = Env))+geom_point(aes(shape =factor(Env)))+
  geom_text(label=Yield_Performance$Gen, nudge_x=.25,nudge_y =.25 )+
  theme_dark()
```



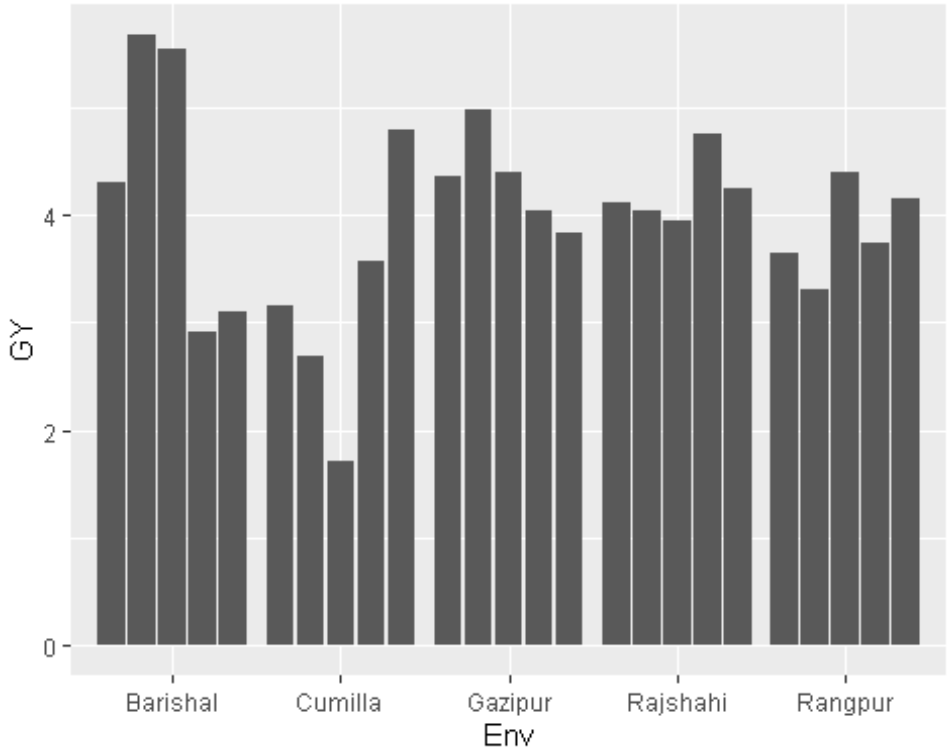

```
#This code creates a scatter plot with GP on the x-axis, GY on the y-axis,
and points colored and shaped by the Env variable. It labels each point with
the Gen column values and uses a dark theme for a visually impactful
appearance.

ggplot(Yield_Performance,aes(x=GP,y=GY,colour = Env))+geom_point(aes(shape
=factor(Env)))+
  geom_text(label=Yield_Performance$Gen, nudge_x=.25,nudge_y =.25 )+
  theme_classic()
```



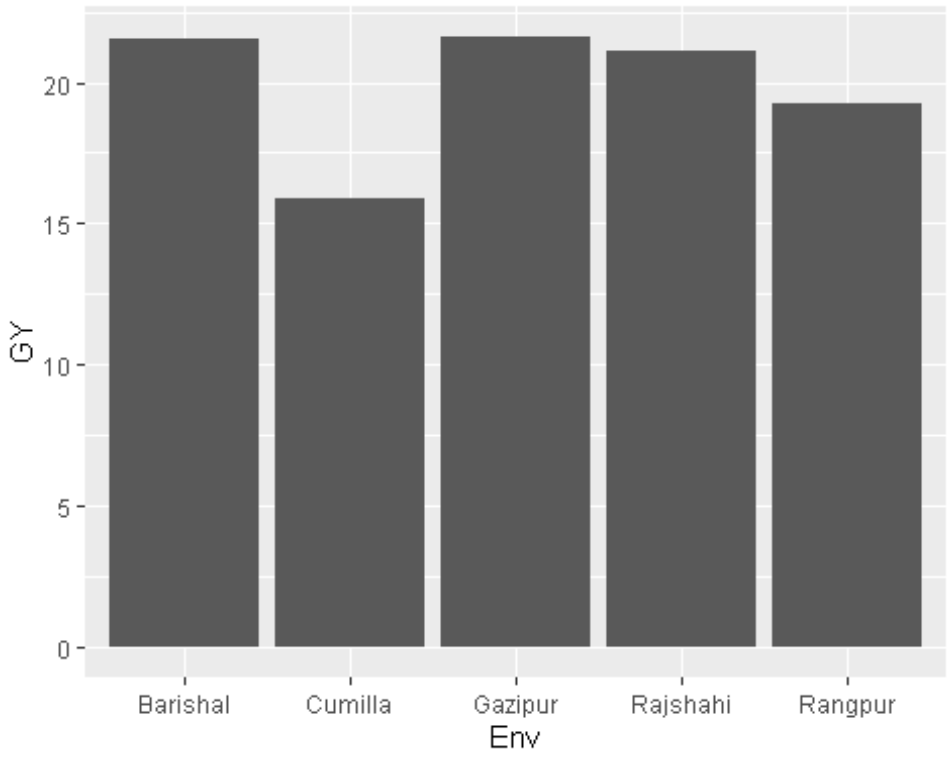

```
#This code generates a scatter plot with GP on the x-axis, GY on the y-axis, and points colored and shaped by the Env variable. It labels each point with the Gen column values and uses a classic theme for a clean and professional look.
```

Bar chart

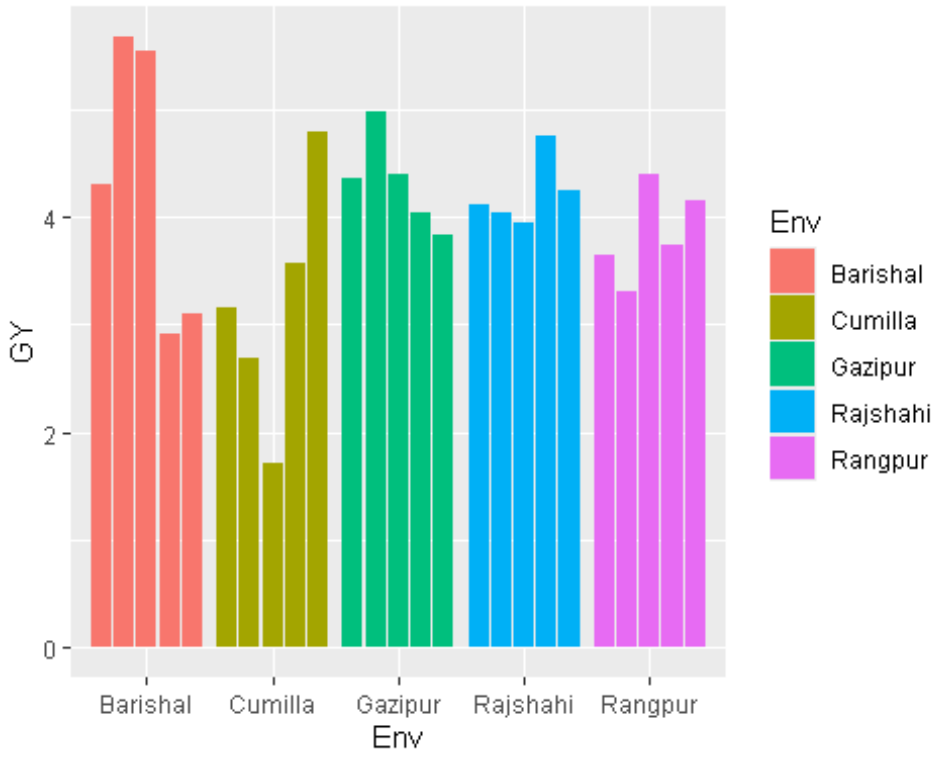
```
ggplot(Yield_Performance,aes(x=Env,y=GY))+geom_bar(stat ="identity",position =position_dodge2())
```



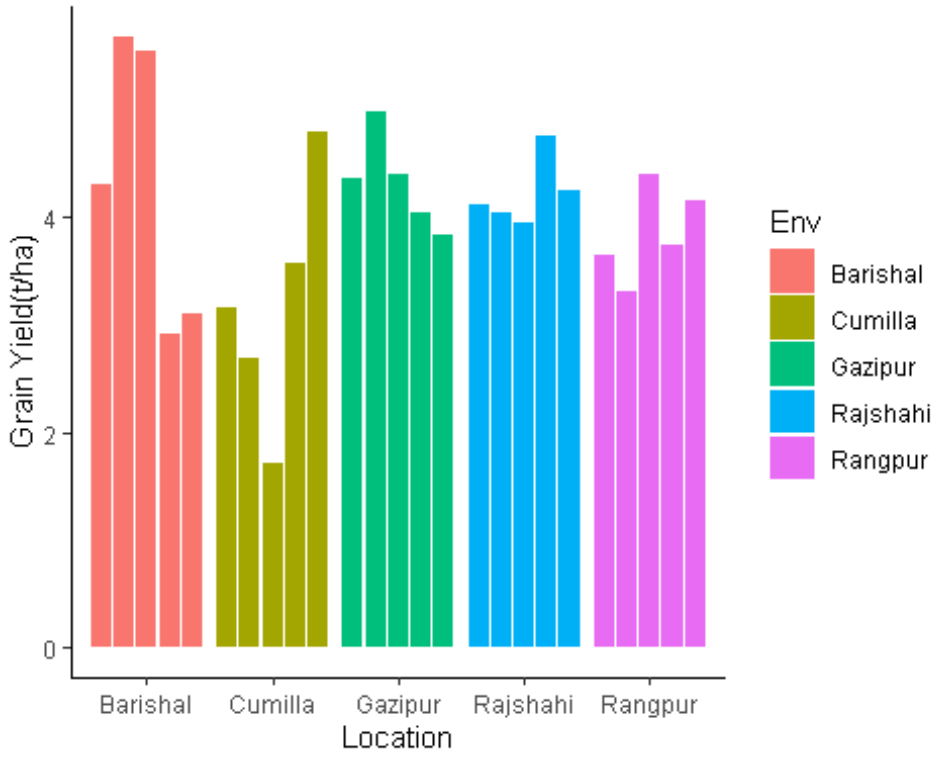

```
#This code generates a bar plot with Env on the x-axis and GY on the y-axis. Bars are placed side-by-side to facilitate comparison of GY values across different environments.

ggplot(Yield_Performance,aes(x=Env,y=GY))+geom_bar(stat ="identity")
```



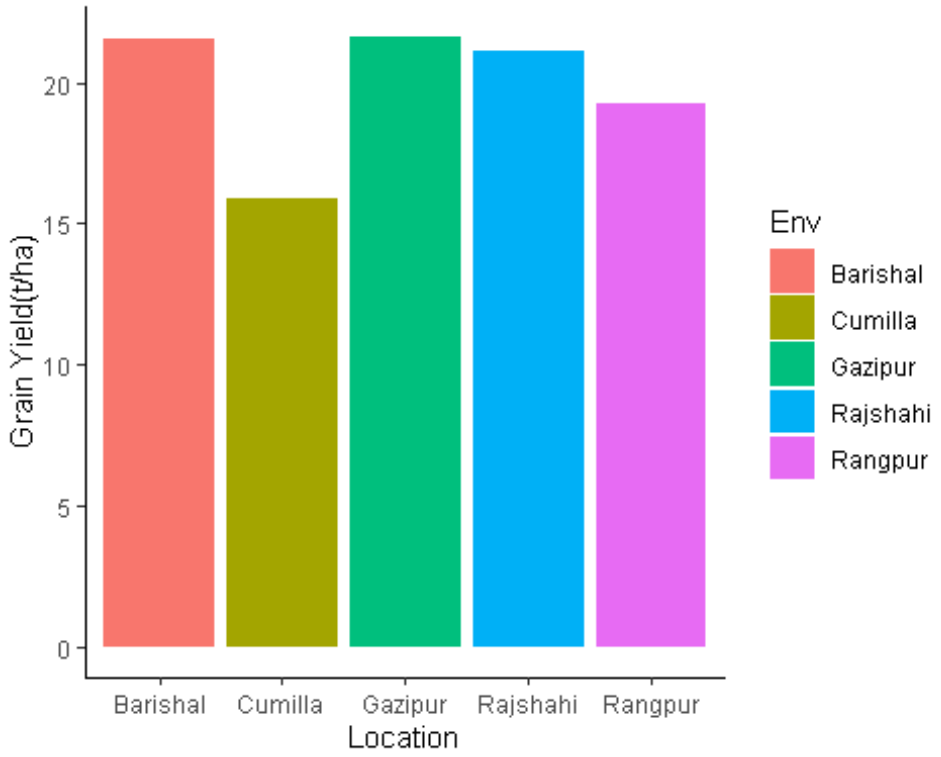

```
#This code generates a bar plot with Env on the x-axis and GY on the y-axis,
showing the GY values for each environment as bar heights.

ggplot(Yield_Performance,aes(x=Env,y=GY,fill = Env))+geom_bar(stat
="identity",position =position_dodge2())
```



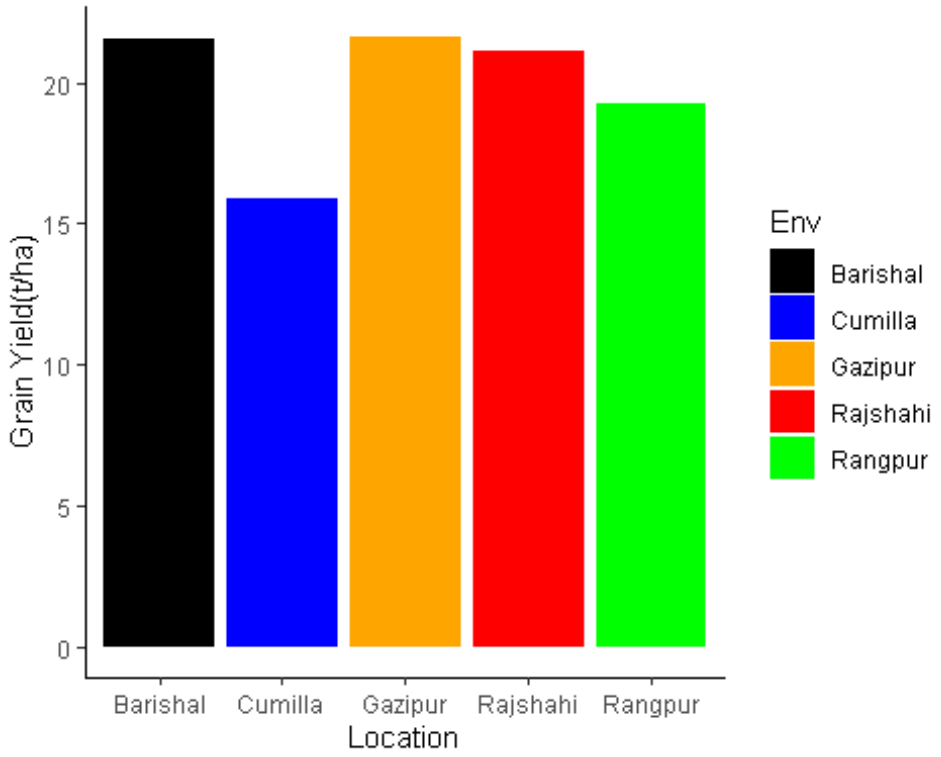

```
#This code generates a bar plot with Env on the x-axis and GY on the y-axis, showing the GY values for each environment as bar heights. Bars are filled with different colors based on the Env variable and are placed side-by-side for easy comparison.

ggplot(Yield_Performance,aes(x=Env,y=GY,fill = Env))+geom_bar(stat
="identity",position =position_dodge2())+
  xlab("Location")+ ylab("Grain Yield(t/ha)")+theme_classic()
```




```
#This code generates a bar plot with Env on the x-axis, GY on the y-axis, and
bars filled by the Env variable. The bars are placed side-by-side for easy
comparison, and the axes are labeled "Location" and "Grain Yield(t/ha)." The
plot uses a classic theme for a clean and professional look.

ggplot(Yield_Performance,aes(x=Env,y=GY,fill = Env))+geom_bar(stat
="identity")+
  xlab("Location")+ ylab("Grain Yield(t/ha)")+theme_classic()
```




```
#This code generates a bar plot with Env on the x-axis, GY on the y-axis, and bars filled by the Env variable. The axes are labeled "Location" and "Grain Yield(t/ha)," and the plot uses a classic theme for a clean and professional appearance

ggplot(Yield_Performance,aes(x=Env,y=GY,fill =
Env))+scale_fill_manual(values=c("black","blue","orange","red","green"))+
  geom_bar(stat ="identity")+
  xlab("Location")+ ylab("Grain Yield(t/ha)")+theme_classic()
```




```
#This code generates a bar plot with Env on the x-axis, GY on the y-axis, and bars filled with manually specified colors. The axes are labeled "Location" and "Grain Yield(t/ha)," and the plot uses a classic theme for a clean and professional look.

library(metan)

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2

## |=============================================================|

## | Multi-Environment Trial Analysis (metan) v1.19.0            |

## | Author: Tiago Olivoto                                       |

## | Type 'citation('metan')' to know how to cite metan          |

## | Type 'vignette('metan_start')' for a short tutorial         |

## | Visit 'https://bit.ly/metanpkg' for a complete tutorial     |

## |=============================================================|

# The metan package in R is a valuable tool for conducting meta-analyses and analyzing multi-environment trial data. It facilitates both fixed and random-effects meta-analyses, meta-regression, and provides robust visualization capabilities such as forest plots. Additionally, metan offers methods for
```

assessing stability and performance across different environments, making it a comprehensive package for researchers working with multi-environment trials. For example, using metan, one can easily perform a meta-analysis on study data by specifying effect sizes and standard errors, and visualize the results in a clear, interpretable format.

```
inspect(Yield_Performance[4:9])
```

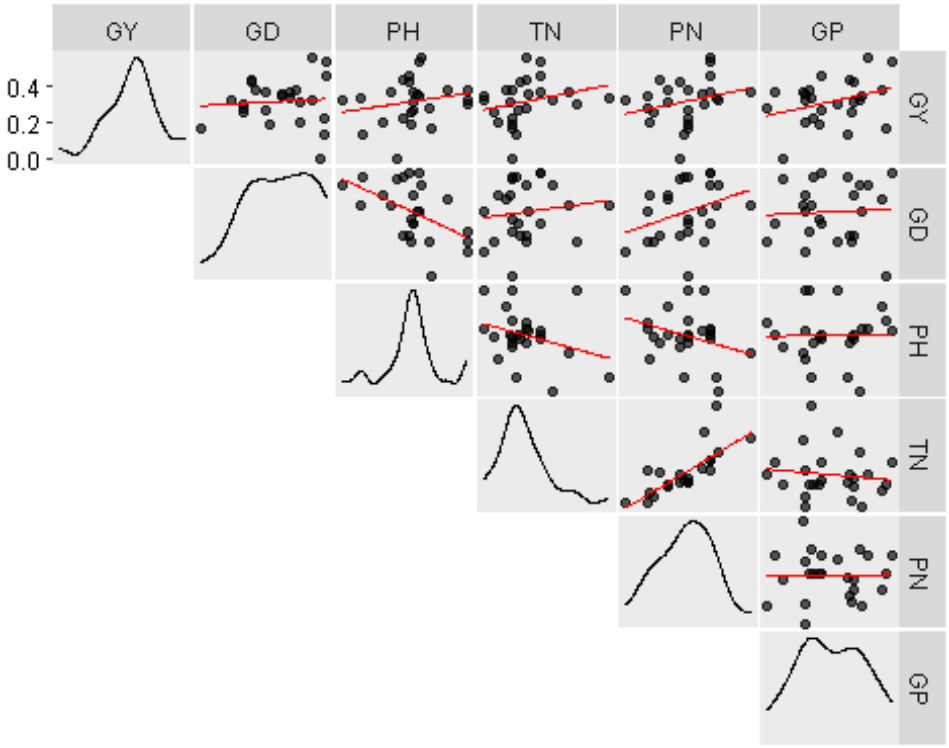
```
## # A tibble: 6 × 10
##   Variable Class   Missing Levels Valid_n   Min Median    Max Outlier Text
##   <chr>    <chr>   <chr>   <chr>    <int> <dbl>  <dbl>  <dbl>   <dbl>
<lgl>
## 1 GY       numeric No      -           25   1.7   4.05   5.68       2 NA
## 2 GD       numeric No      -           25 109   130    142          0 NA
## 3 PH       numeric No      -           25  80.6 108.   131          6 NA
## 4 TN       numeric No      -           25   7     9.4   16          1 NA
## 5 PN       numeric No      -           25   6     9     12          0 NA
## 6 GP       numeric No      -           25  51    80    118          0 NA

## Warning: Considering the levels of factors, .data should have 1 rows, but
it
## has 25. Use 'as_factor()' for coercing a variable to a factor.

## Warning: Expected three or more factor variables. The data has only 0.

## Warning: Possible outliers in variable(s) GY, PH, TN. Use
'find_outliers()' for
## more details.
```

 # This Yield_Performance[4:9] is used to access rows 4 to 9 of the Yield_Performance data frame.The inspect function is part of a specific package to inspect or view this subset of data.

```
inspect(Yield_Performance[4:9],plot =T )
```

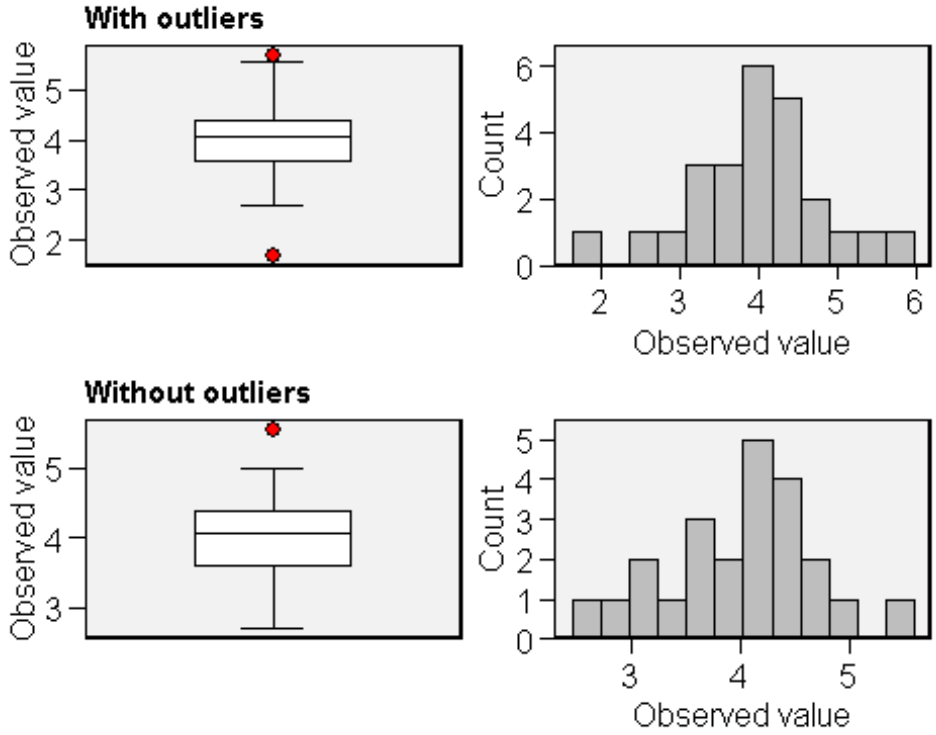
```
## # A tibble: 6 × 10
##   Variable Class   Missing Levels Valid_n   Min Median    Max Outlier Text
##   <chr>    <chr>   <chr>   <chr>    <int> <dbl>  <dbl>  <dbl>   <dbl>
<lgl>
## 1 GY       numeric No      -           25   1.7   4.05   5.68       2 NA
## 2 GD       numeric No      -           25 109   130    142          0 NA
## 3 PH       numeric No      -           25  80.6 108.   131          6 NA
## 4 TN       numeric No      -           25   7     9.4   16          1 NA
## 5 PN       numeric No      -           25   6     9     12          0 NA
## 6 GP       numeric No      -           25  51    80    118          0 NA

## Warning: Considering the levels of factors, .data should have 1 rows, but
it
## has 25. Use 'as_factor()' for coercing a variable to a factor.

## Warning: Expected three or more factor variables. The data has only 0.
```

```
## Warning: Possible outliers in variable(s) GY, PH, TN. Use
'find_outliers()' for
## more details.
```




```
#This command inspect(Yield_Performance[4:9], plot = T) suggests the use of
an inspect function, likely from a specific package, to examine the subset of
the Yield_Performance data frame containing rows 4 to 9. The plot = T
argument indicates that the function should generate a plot based on this
subset of data.

find_outliers(Yield_Performance$GY)

## Trait: .data
## Number of possible outliers: 2
## Line(s): 2 8
## Proportion: 8.7%
## Mean of the outliers: 3.69
## Maximum of the outliers: 5.68  | Line 2
## Minimum of the outliers: 1.7  | Line 8
## With outliers:    mean = 3.977 | CV = 22.157%
## Without outliers: mean = 4.002 | CV = 17.292%

#This command is use to  find_outliers(Yield_Performance$GY) function
suggests the intention to identify outliers in the GY column of the
Yield_Performance data frame.

find_outliers(Yield_Performance$GY,plots =T)
```

```
## Trait: .data
## Number of possible outliers: 2
## Line(s): 2 8
## Proportion: 8.7%
## Mean of the outliers: 3.69
## Maximum of the outliers: 5.68  | Line 2
## Minimum of the outliers: 1.7  | Line 8
## With outliers:    mean = 3.977 | CV = 22.157%
## Without outliers: mean = 4.002 | CV = 17.292%
```




```
#This command find_outliers(Yield_Performance$GY, plots = T) suggests the use of a function to identify outliers in the GY column of the Yield_Performance data frame and create visual plots to display them.
```