Exploratory Data Analysis with R

Displaying Data with ggplot()

Xuemao Zhang
East Stroudsburg University

September 23, 2022

Outline

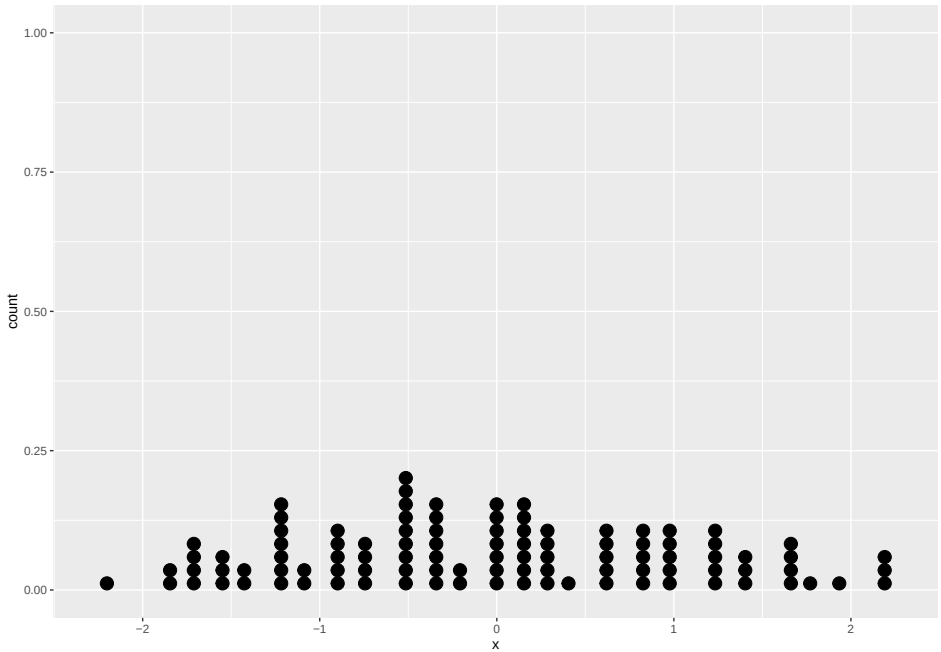
- Univariate Plots
  - Scatter plots
  - Histograms
  - KDE plots
  - Cumulative Frequencies
  - Box Plots
  - Violin Plots
  - Bar Chart
  - Pie Charts
- Bivariate and Multivariate Plots
  - Bivariate Scatter Plots
  - 3D scatter plots

Univariate scatter plots

- We just plot each individual data point. It is geom_dotplot() in ggplot2.

```
library(ggplot2)
x=rnorm(100);
# we must make the data a data frame to use ggplot2
x=as.data.frame(x);
ggplot(x, aes(x=x)) +
  geom_dotplot(dotsize=0.5);
```

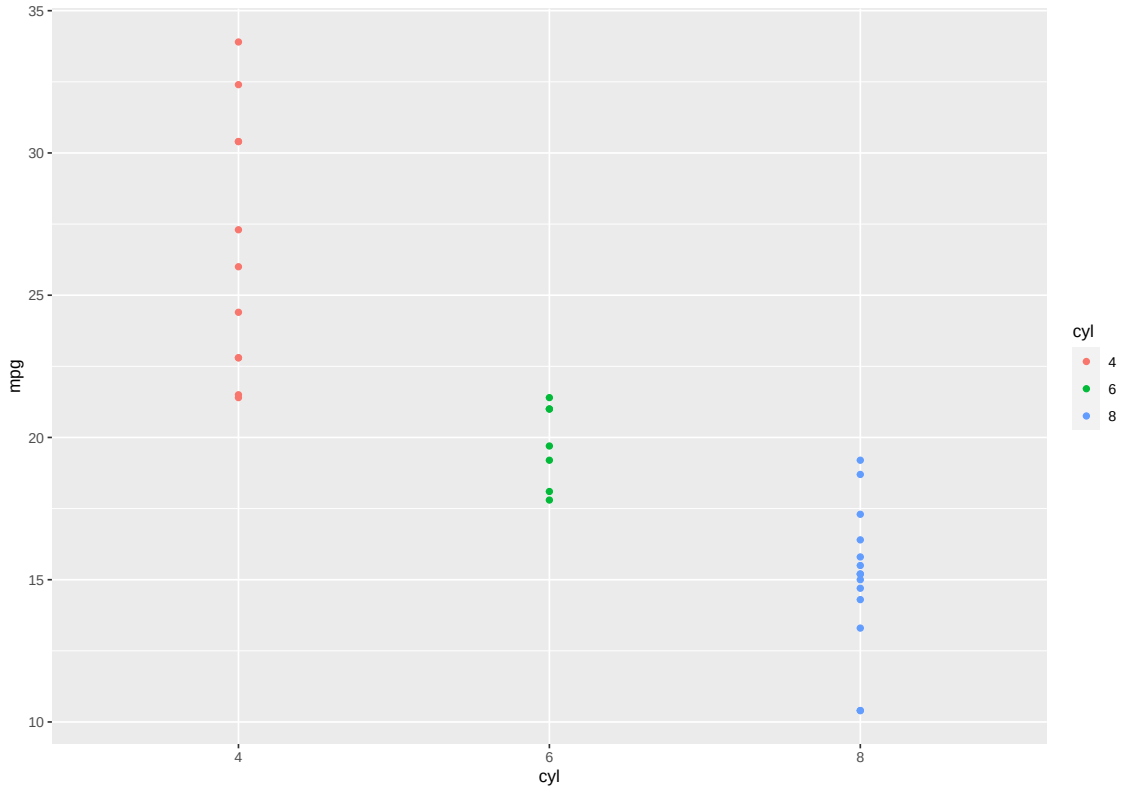
```
## Bin width defaults to 1/30 of the range of the data. Pick better
```



Univariate scatter plots

- Plot points by groups

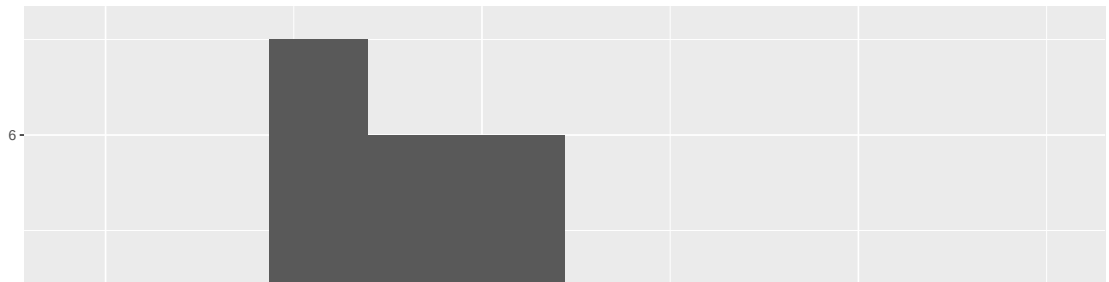
```
library(ggplot2);
mtcars$cyl=as.factor(mtcars$cyl);
ggplot(data=mtcars, aes(x=cyl,y=mpg, color=cyl)) +
  geom_point();
```



Histograms

- Histogram provide a first good view of the distribution of your data.
  - A Mode of a histogram is a hump or high-frequency bin:
    - ★ Unimodal: one mode
    - ★ Bimodal: two modes
    - ★ Multimodal: 3 or more modes
  - Symmetry
  - Gap and outliers
  - Skewness (for unimodal data)
    - ★ A histogram is **skewed right** if the longer tail is on the right side of the mode.
    - ★ A histogram is **skewed left** if the longer tail is on the left side of the mode.

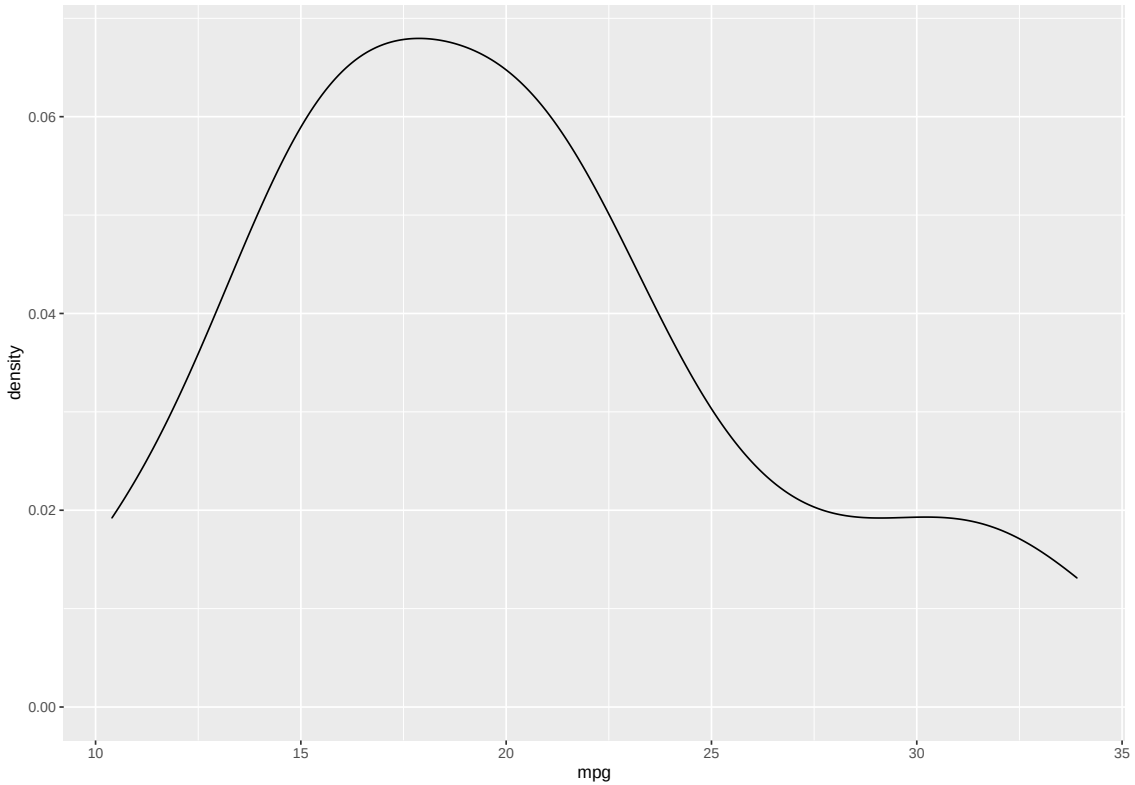
```
library(ggplot2);
ggplot(data=mtcars, aes(x=mpg)) +
  geom_histogram(bins=10);
```



Kernel-Density-Estimation (KDE) plots

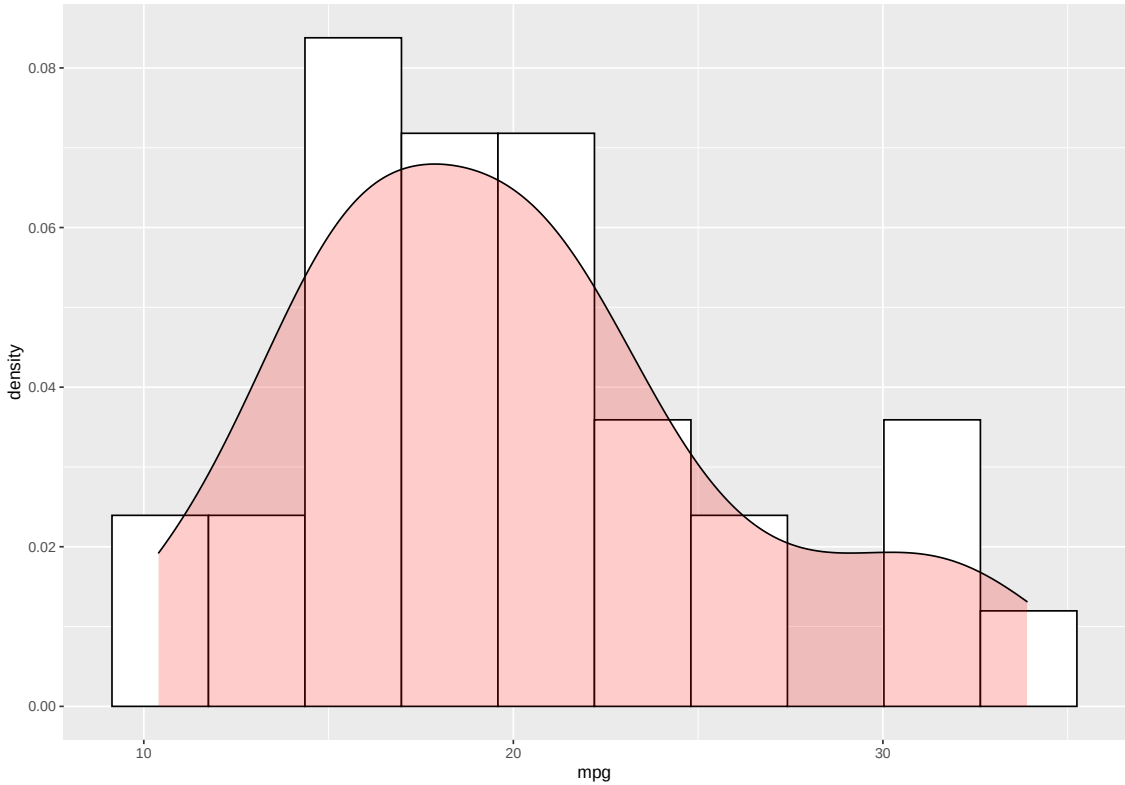
- The technique of Kernel-Density-Estimation (KDE) can be used to estimate the smooth **probabillity density curve** of a data set.

```
ggplot(data=mtcars, aes(x=mpg, y = ..density..)) +
  geom_density();
```



Kernel-Density-Estimation (KDE) plots

```
ggplot(data=mtcars, aes(x=mpg)) +
   geom_histogram(aes(y= ..density..),bins=10,colour="black",
                  fill="white")+
  #Histogram with density instead of count on y-axis
  geom_density(alpha=.2, fill="red");
```

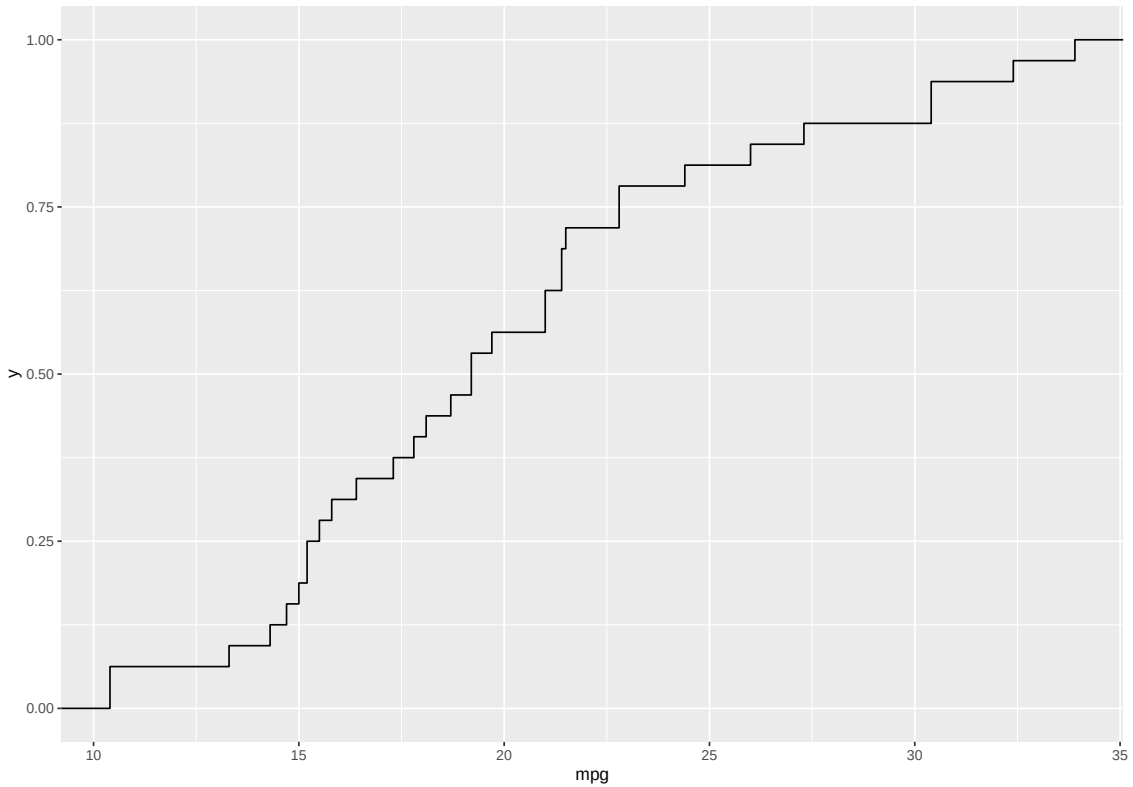



```
  #Overlay with transparent density plot;
```

Cumulative Frequencies

- A cumulative frequency curve indicates the number (or percent) of data with less than a given value.
- `stat_ecdf` in ggplot2

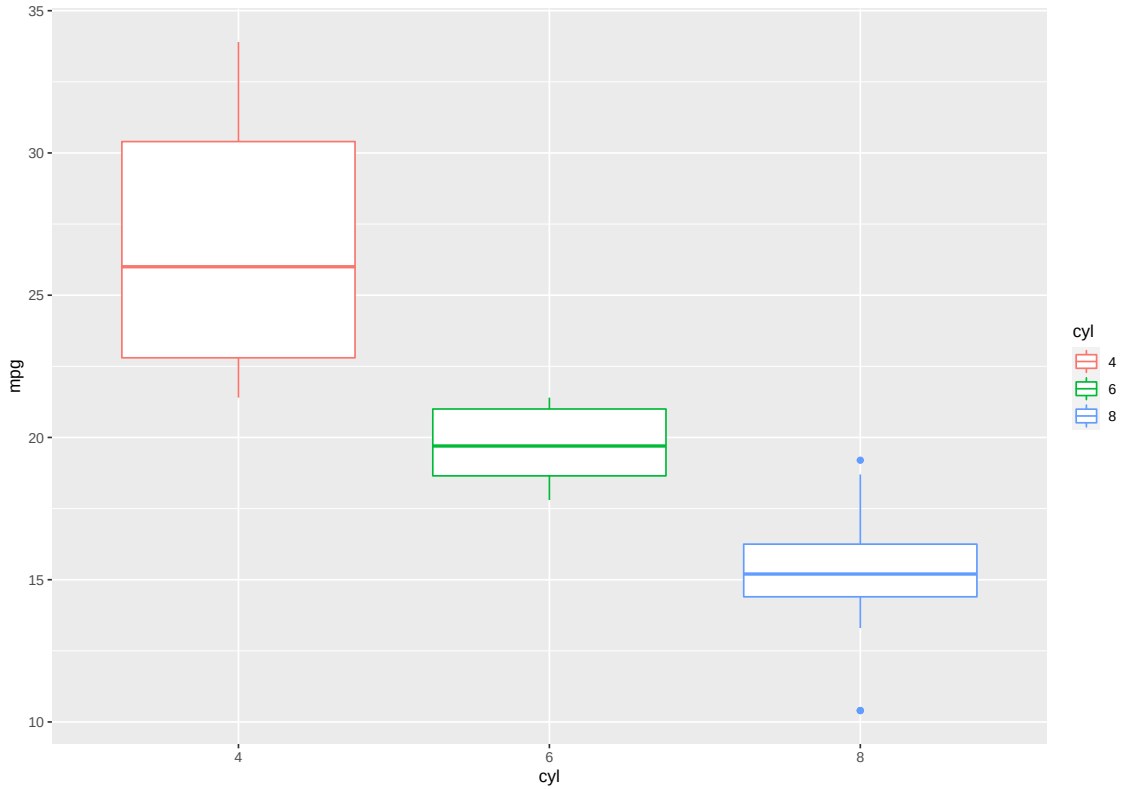
```
ggplot(mtcars, aes(x=mpg)) +
  stat_ecdf(geom = "step"); #
```



Box Plots

- Box plots are frequently used in scitific publications to indicate data values in two or more groups.

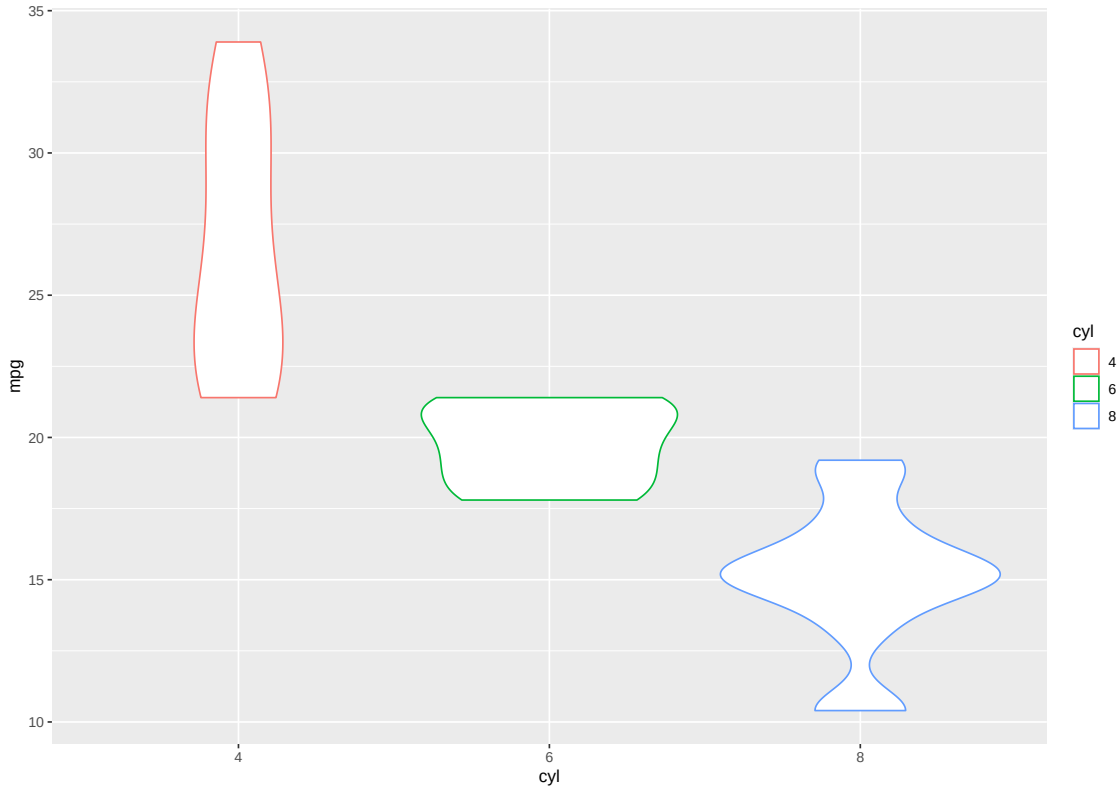
```
mtcars$cyl=as.factor(mtcars$cyl);
ggplot(data=mtcars, aes(x=cyl,y=mpg, color=cyl)) +
  geom_boxplot();
```



Violin Plots

- Boxplot can be combined with KDE-plots to produce the so-called violin plots, where the vertical axis is the same as for the box-plot, but in addition a KDE-plot is shown symmetically along the horizontal direction.

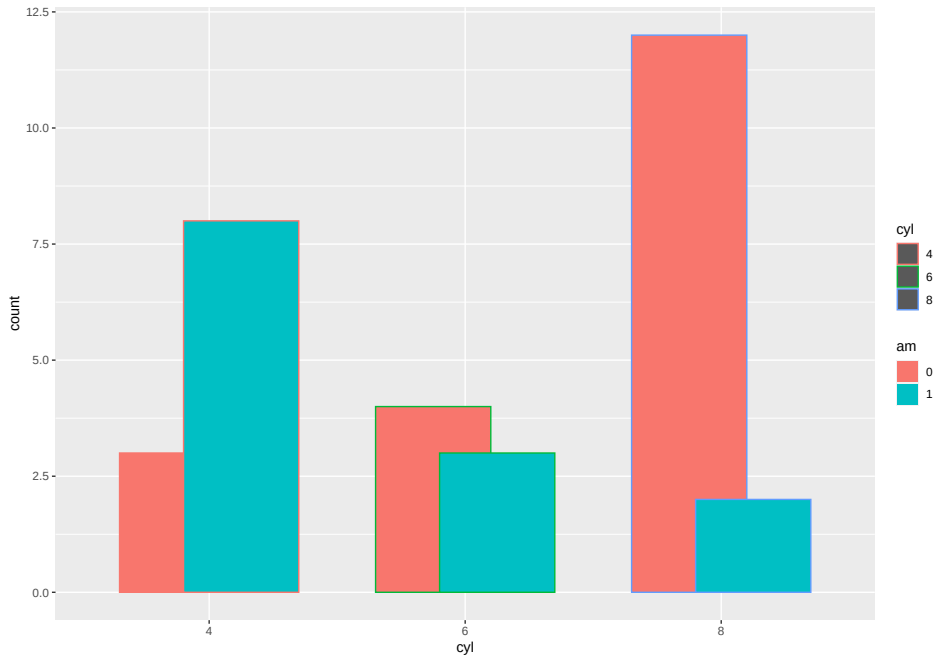
```
ggplot(data=mtcars, aes(x=cyl,y=mpg, color=cyl)) +
  geom_violin();
```



Categorical data: Bar Charts and Pie Charts

- Sometimes, we want to have grouped bar charts

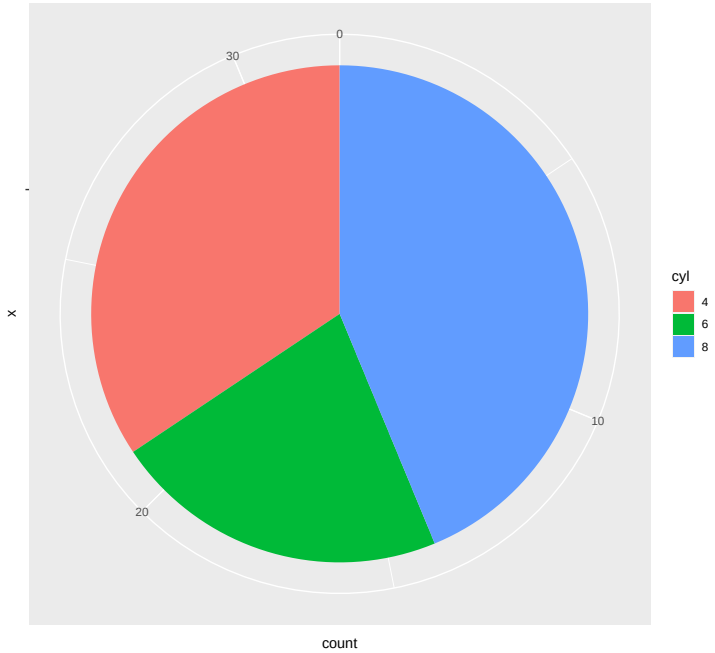
```
mtcars$cyl=as.factor(mtcars$cyl);
mtcars$am=as.factor(mtcars$am);
ggplot(data=mtcars, aes(x=cyl, color=cyl)) +
  geom_bar(aes(fill=am),position=position_dodge(width=0.5));
```




```
#Dodging preserves the vertical position of an geom
#while adjusting the horizontal position
```

Categorical data: Bar Charts and Pie Charts

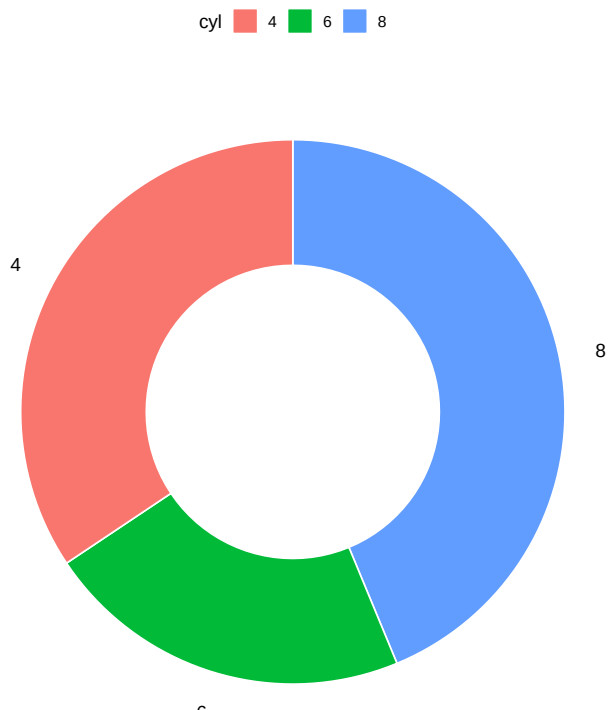
```
ggplot(data=mtcars, aes(x="", fill=cyl)) +
  geom_bar()+
  coord_polar("y"); #Create a pie chart
```



Categorical data: Bar Charts and Pie Charts

- To get a donut plot, it's better to use an extended package ggpubr. For more information, see http://rpkgs.datanovia.com/ggpubr/reference/ggdonutchart.html

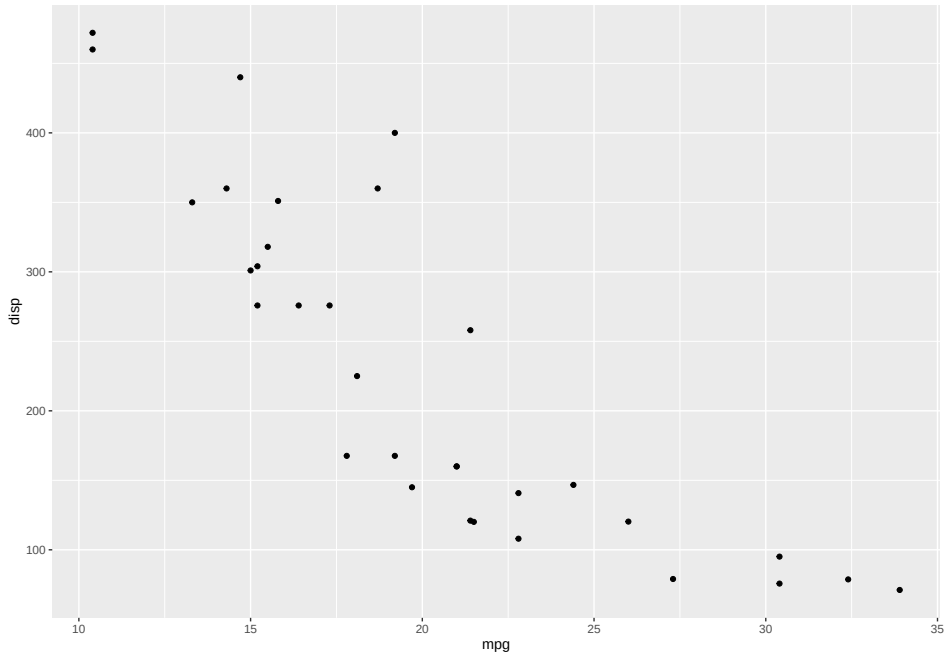
```
library(ggpubr)
data=data.frame(table(mtcars$cyl))
names(data)=c("cyl","count")
ggdonutchart(data, "count", label = "cyl",
             fill="cyl",color = "white")
```



Bivariate Scatter Plots

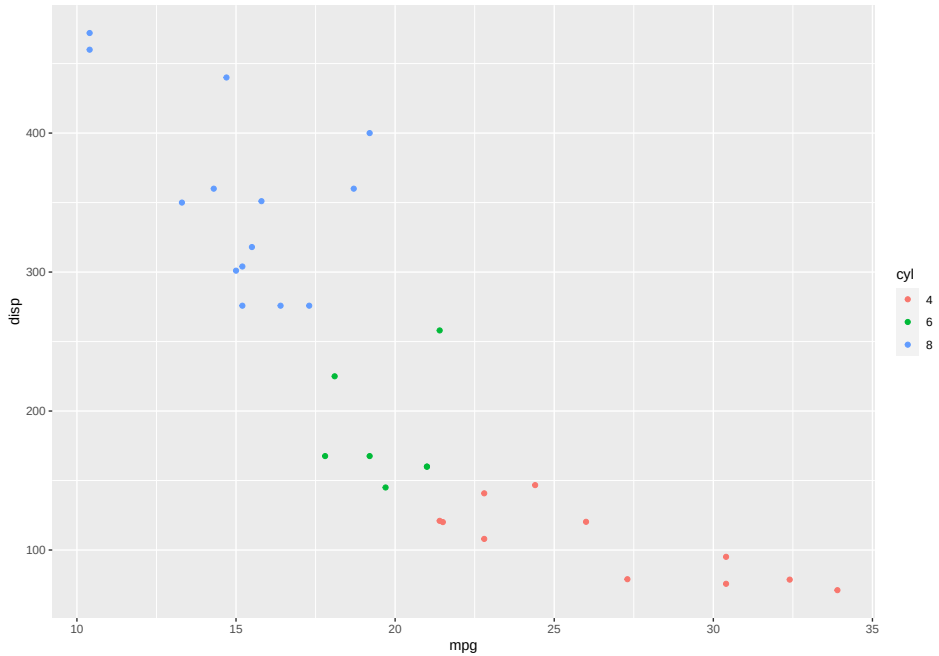
- Bivariate scatter plots tell us the relationship between two numerical variables

```
ggplot(data=mtcars, aes(x=mpg, y=disp)) +
  geom_point();
```



Bivariate Scatter Plots

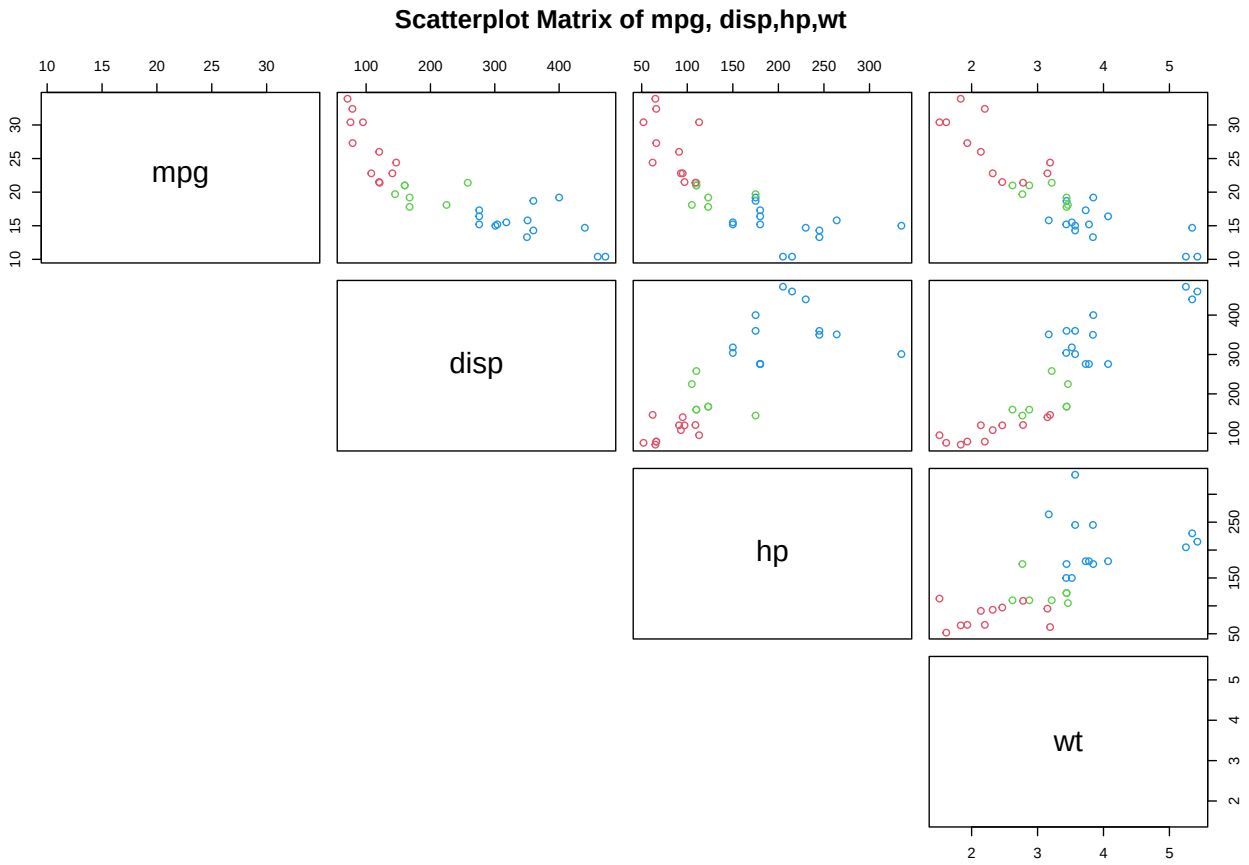
```
mtcars$cyl=factor(mtcars$cyl);
ggplot(data=mtcars, aes(x=mpg, y=disp,color=cyl)) +
  geom_point();
```



Bivariate Scatter Plots

- pairs() function creates beautiful scatter plot matrix.

```
pairs(~mpg+disp+hp+wt,data=mtcars, col=c(2,3,4)[mtcars$cyl],
main="Scatterplot Matrix of mpg, disp,hp,wt ",lower.panel = NULL);
```



Bivariate Scatter Plots

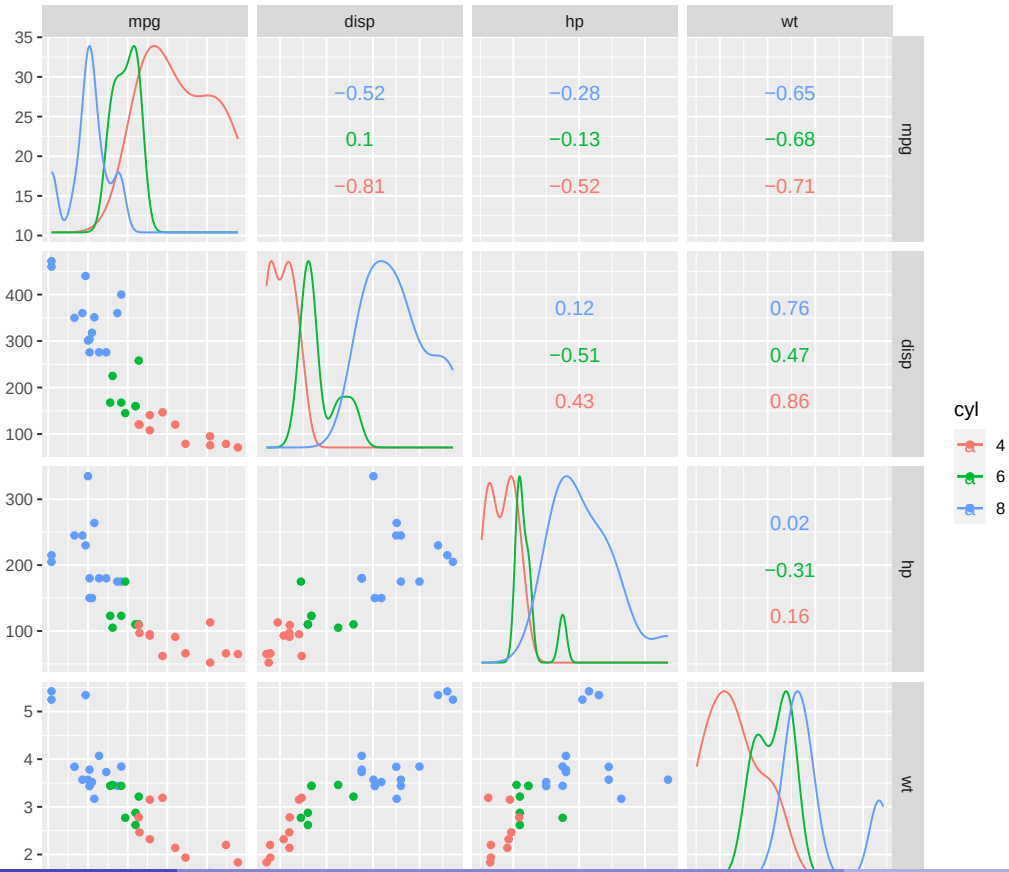
- GGally R package extends 'ggplot2' by adding several functions to reduce the complexity of combining geometric objects with transformed data. For example, pairwise plot matrix. The package is maintained by Barret Schloerke.
- The function `ggscatmat` and `ggpairs()` are used to produce a matrix of scatter plots for visualizing the correlation between variables.

Bivariate Scatter Plots

- The function ggscatmat strictly take numeric variables.

```
library(GGally);
library(dplyr)
mtcars %>% select(mpg,disp,hp,wt,cyl) %>% ggscatmat(color="cyl");
```

```
## Warning in ggscatmat(., color = "cyl"): Factor variables are omit
```



Bivariate Scatter Plots

```
mtcars %>% select(mpg,disp,hp,wt,cyl) %>%
  ggpairs(aes(color=cyl,alpha = 0.4));
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`
```

Bivariate Scatter Plots

```
mtcars %>% select(mpg,disp,hp,wt,cyl) %>%
  ggpairs(aes(color=cyl,alpha = 0.4),
    lower = list(combo=wrap("facethist",binwidth=10)));
```

3D scatter plots

- The rgl package is used to produce interactive 3-D plots.
- The package `rgl`(https://cran.r-project.org/web/packages/rgl/rgl.pdf) is maintained by Duncan Murdoch.
- Output may be on screen using OpenGL, or to various standard 3D file formats including WebGL, PLY, OBJ, STL as well as 2D image formats, including PNG, Postscript, SVG, PGF.

3D scatter plots

- The function `plot3d(x,y,z)` can be used to draw a 3D scatter plot. It is similar to the classic `plot` function, but works in 3 dimensions. If you call `plot3d` again, it will overwrite the current plot.

```
library(rgl);
x = iris$Sepal.Length;
y = iris$Petal.Length;
z = iris$Sepal.Width;
plot3d(x,y,z,type="s", size=1, col=as.numeric(iris$Species));
# "s" is for spheres
```

Exporting images: Export images as png or pdf

- `rgl` has functions to save snapshots or other recordings of a scene, without any 3D information being saved
- The function `rgl.snapshot()` is used to save the screenshot as png file:

```
plot3d(x,y,z,type="s", size=1, col=as.numeric(iris$Species));
rgl.snapshot(filename = "plot1.png")
```

- The function `rgl.postscript()` is used to save the screenshot to a file in ps, eps, tex, pdf, svg or pgf format:

```
plot3d(x,y,z,type="s", size=1, col=as.numeric(iris$Species));
rgl.postscript("plot1.pdf", fmt="pdf");
```

Export the plot into an interactive HTML file

```
writeWebGL(dir="webGL",filename=file.path(dir, "index.html"));
```

```
plot3d(x,y,z,type="s", size=1, col=as.numeric(iris$Species));
writeWebGL( filename= "plot1.html");
```

Export the plot into an interactive PLY file

- `writePLY()` is used to export a plot into a PLY file commonly used in 3D printing.

```
plot3d(x,y,z,type="s", size=1, col=as.numeric(iris$Species));
writePLY("plot1.ply");
```

License