ggplot2 in R

Neba Nfonsang

University of Denver

“The simple graph has brought more information to the data analyst’s mind than any other device.” — John Tukey

ggplot2

- ggplot2 is a system for creating graphics where you provide the data and the aesthetic mapping to initialize the ggplot object, then add layers such as geom_point(), geom_histogram(), etc.

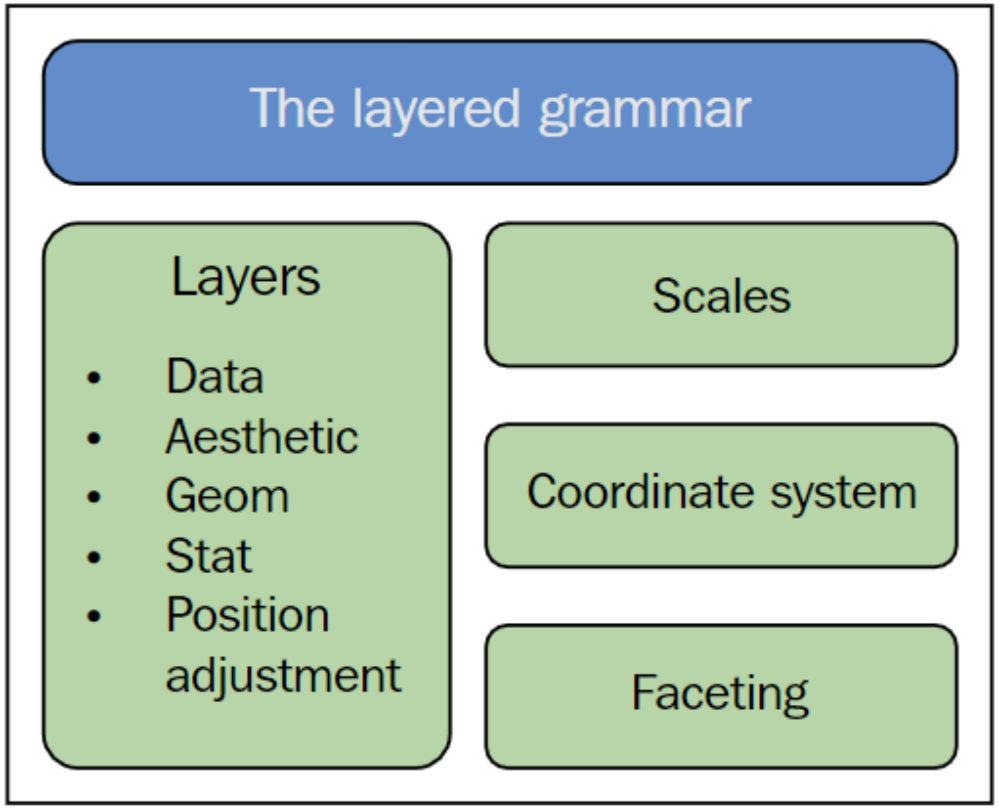
- The ggplot2 can be installed through whole tidyverse package or just the ggplot2 package:

```
install.packages("tidyverse")

install.packages("ggplot2")

library(ggplot2)# load ggplot2
```

https://ggplot2.tidyverse.org/index.html

Components of the Grammar of Graphics in ggplot2

- ggplot is built on the idea that, providing smaller set of functions can be used to define different components of a graph and can be combined to create a larger variety of plots.
- The grammar of graphics identifies and separates each step of the graphic process.



The ggplot() Function

- The **ggplot()** function initializes the ggplot object. It takes the data and aesthetics intended to be common throughout all subsequent layers unless specifically overridden.

```
ggplot(data, aes())
```

- The **data** specified is the default dataset to use in the plot. if not specified, must be supplied in each layer added to the plot.
- The **aes()** function takes a list of properties of the objects, to be used for the plot. If not specified, must be supplied in each layer added to the plot

https://ggplot2.tidyverse.org/reference/ggplot.html

The ggplot() Function

- There are three common ways to create the ggplot() object.

```
# default data and aesthetics are provided
ggplot(data, aes(x, y, other aesthetics)
# aesthetics need to be supplied to layers
added
ggplot(data)
# data and aesthetics need to supplied to
other layers added
ggplot()
```

Aesthetic Mapping

- Each aesthetic attribute can mapped to a variable to represent a cluster of data points or set to a constant value.
- Aesthetic in ggplot refers to something you can see, and the aes() function is used to specify:
  - Horizontal and vertical position of points (x and y)
  - Size of points (size)
  - Shape of points (shape)
  - “outside” color of points (color)
  - “inside” color of points (fill)
  - linetype

ggplot2 with the Iris Dataset

```
require(tidyverse, quiet=T)
```

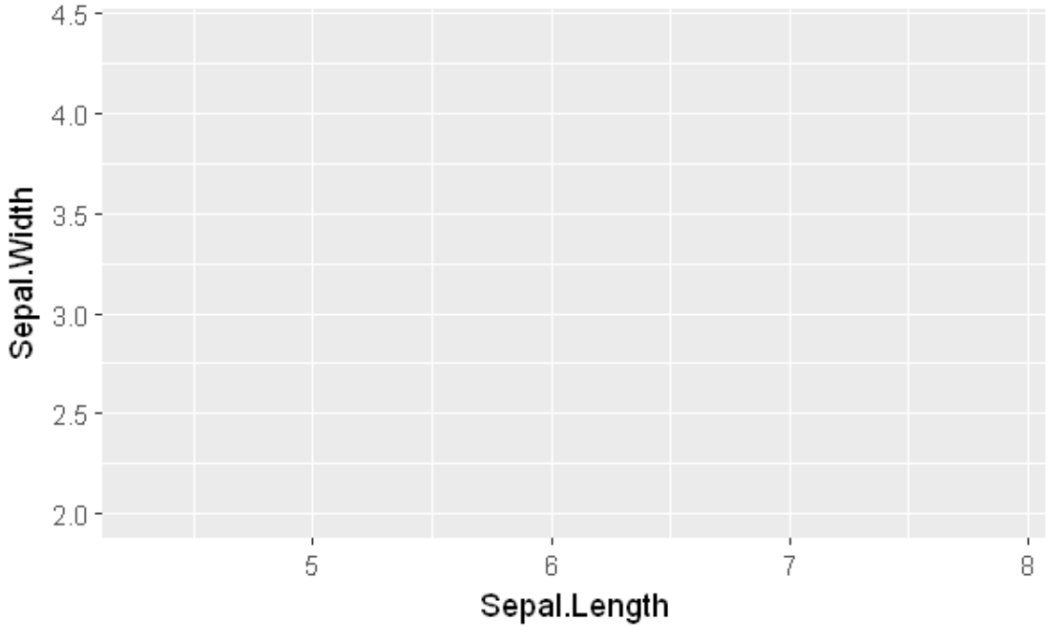
```
# view the iris data in base r
head(iris)
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

ggplot2 with the Iris Dataset

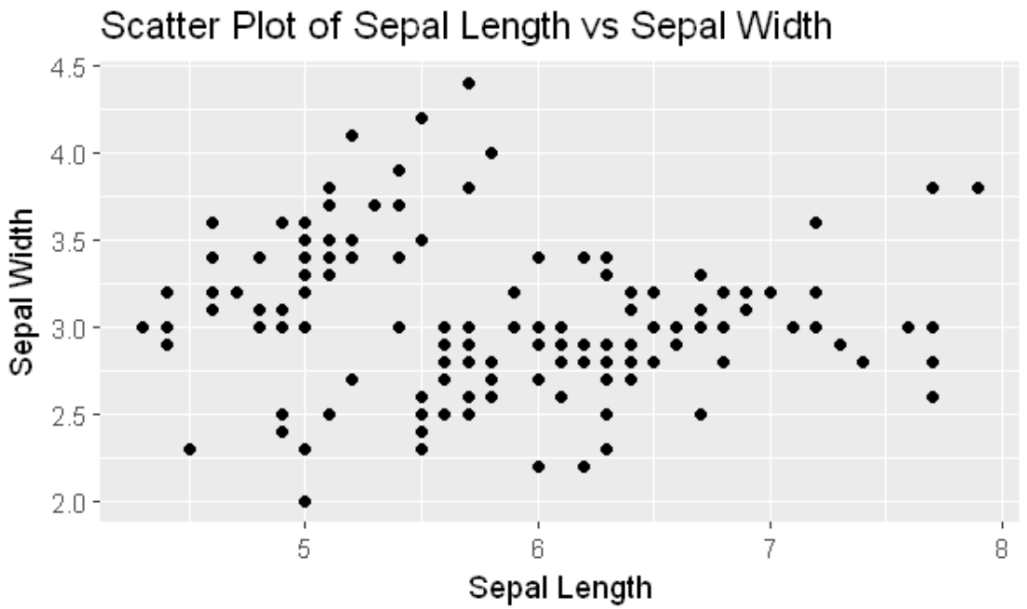
ggplot() creates an empty ggplot object. Plot layers need to be added.

```
# set plot size
options(repr.plot.width=5, repr.plot.height=3)

# initialize the plot object with the iris data
# and supply the x and y axis values
g <- ggplot(data=iris,
            aes(x=Sepal.Length,
                y=Sepal.Width))
g
```



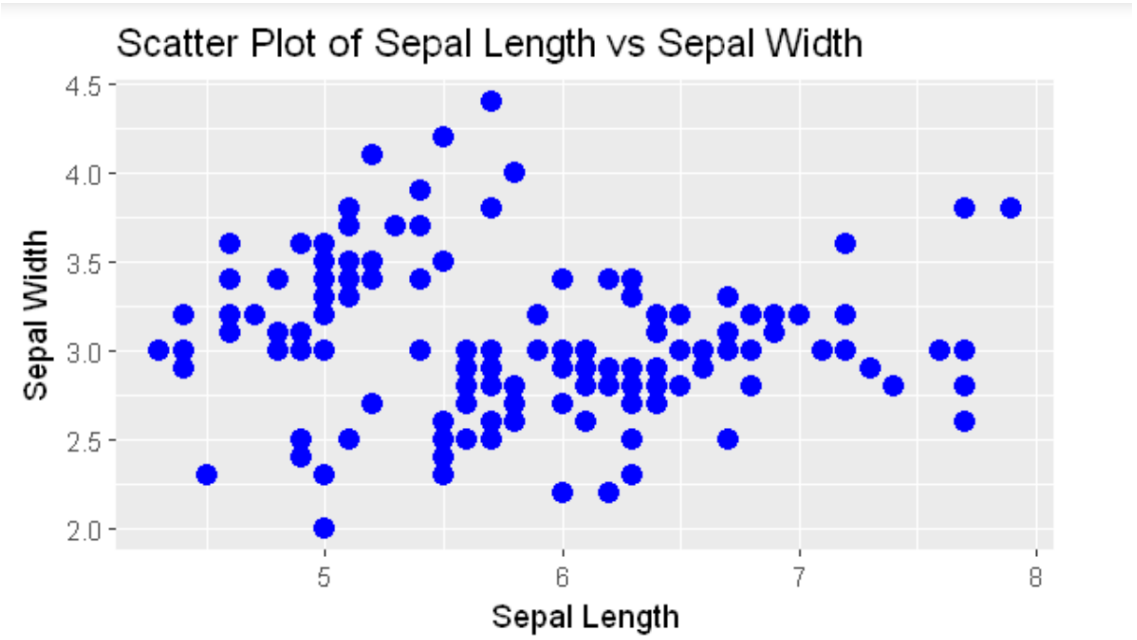
Add a geom_point () Layer

```
g <- ggplot(data=iris,
            aes(x=Sepal.Length,
                y=Sepal.Width))

# add tittle and axes labels using the labs() function
g <- g + labs(title="Scatter Plot of Sepal Length vs Sepal Width",
              x="Sepal Length", y="Sepal Width")

# add a geom_point() layer
g <- g + geom_point()
g
```



Add a geom_point() Layer with Aesthetics

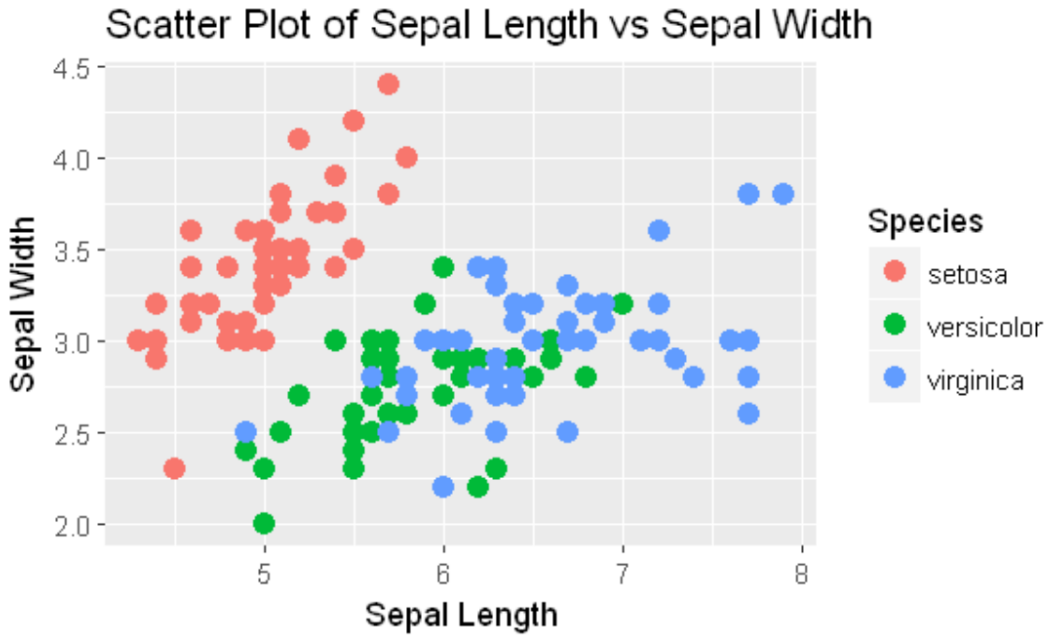
```
g <- ggplot(data=iris,
            aes(x=Sepal.Length,
                y=Sepal.Width))

# add tittle and axes labels using the labs() function
g <- g + labs(title="Scatter Plot of Sepal Length vs Sepal Width",
              x="Sepal Length", y="Sepal Width")

# add a geom_point() layer, specifying color and size of points
g <- g + geom_point(color="blue", size=3)
g
```



Add a geom_point() Layer with Aesthetics

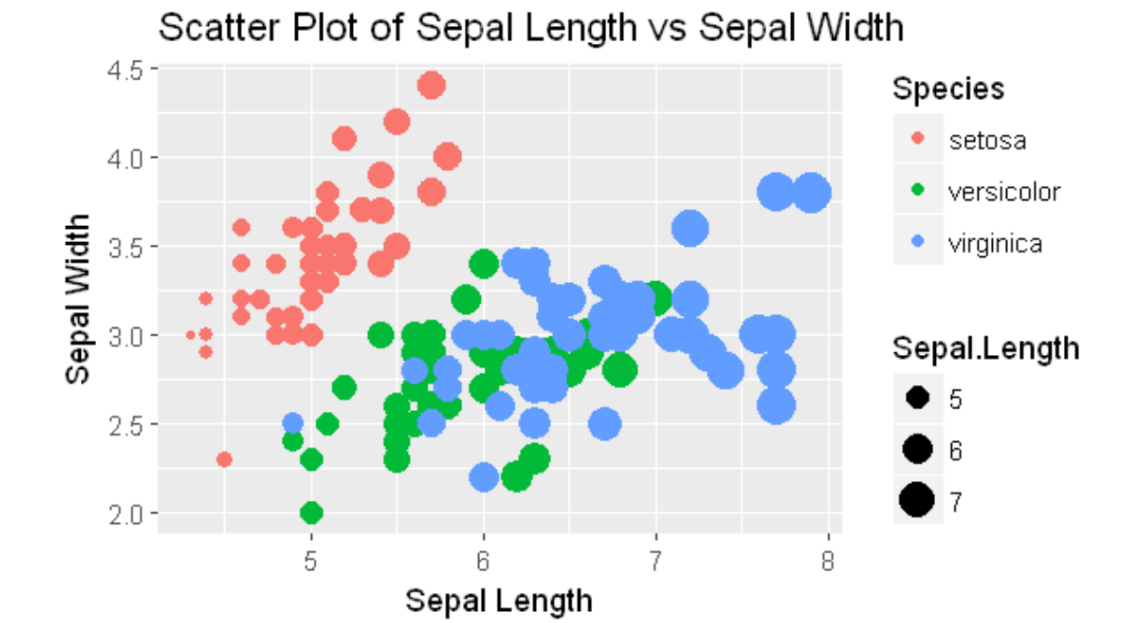
Specify the colors to vary by the Species variable on the data frame

```
g <- ggplot(data=iris,
            aes(x=Sepal.Length,
                y=Sepal.Width))

# add tittle and axes labels using the labs() function
g <- g + labs(title="Scatter Plot of Sepal Length vs Sepal Width",
              x="Sepal Length", y="Sepal Width")

# add a geom_point() layer, let the color vary by Species
# Species is a column name on the data frame
g <- g + geom_point(aes(color=Species), size=3)
g
```



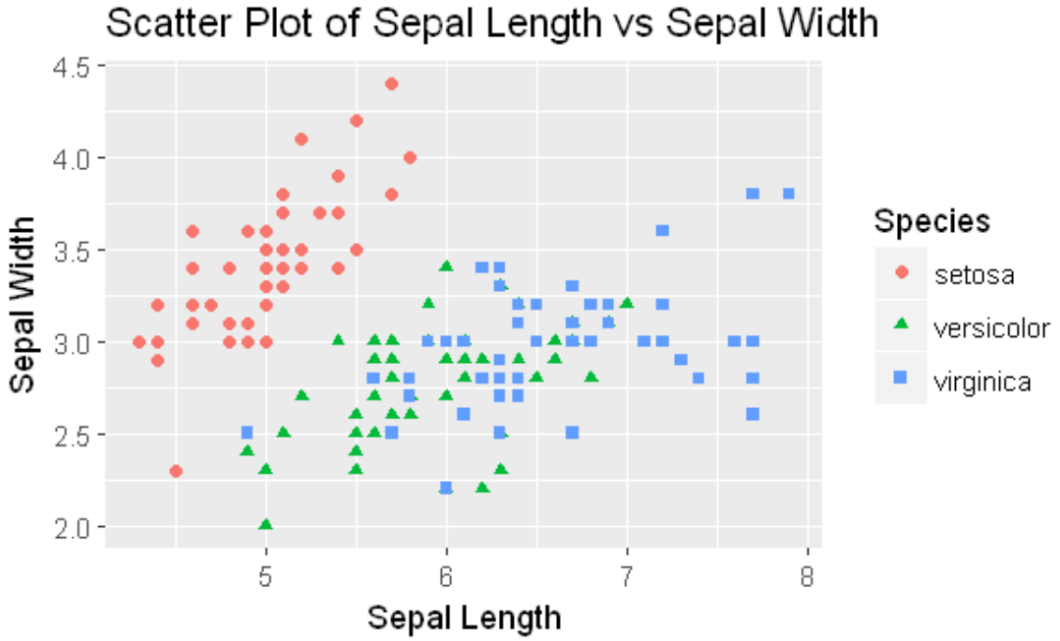
Add a geom_point() Layer with Aesthetics

```
g <- ggplot(data=iris,
            aes(x=Sepal.Length,
                y=Sepal.Width))

# add tittle and axes labels using the labs() function
g <- g + labs(title="Scatter Plot of Sepal Length vs Sepal Width",
              x="Sepal Length", y="Sepal Width")

# add a geom_point() layer, let the color vary by Species
# and let the size vary by Sepal Length
g <- g + geom_point(aes(color=Species, size=Sepal.Length))
g
```



Add a geom_point() Layer with Aesthetics

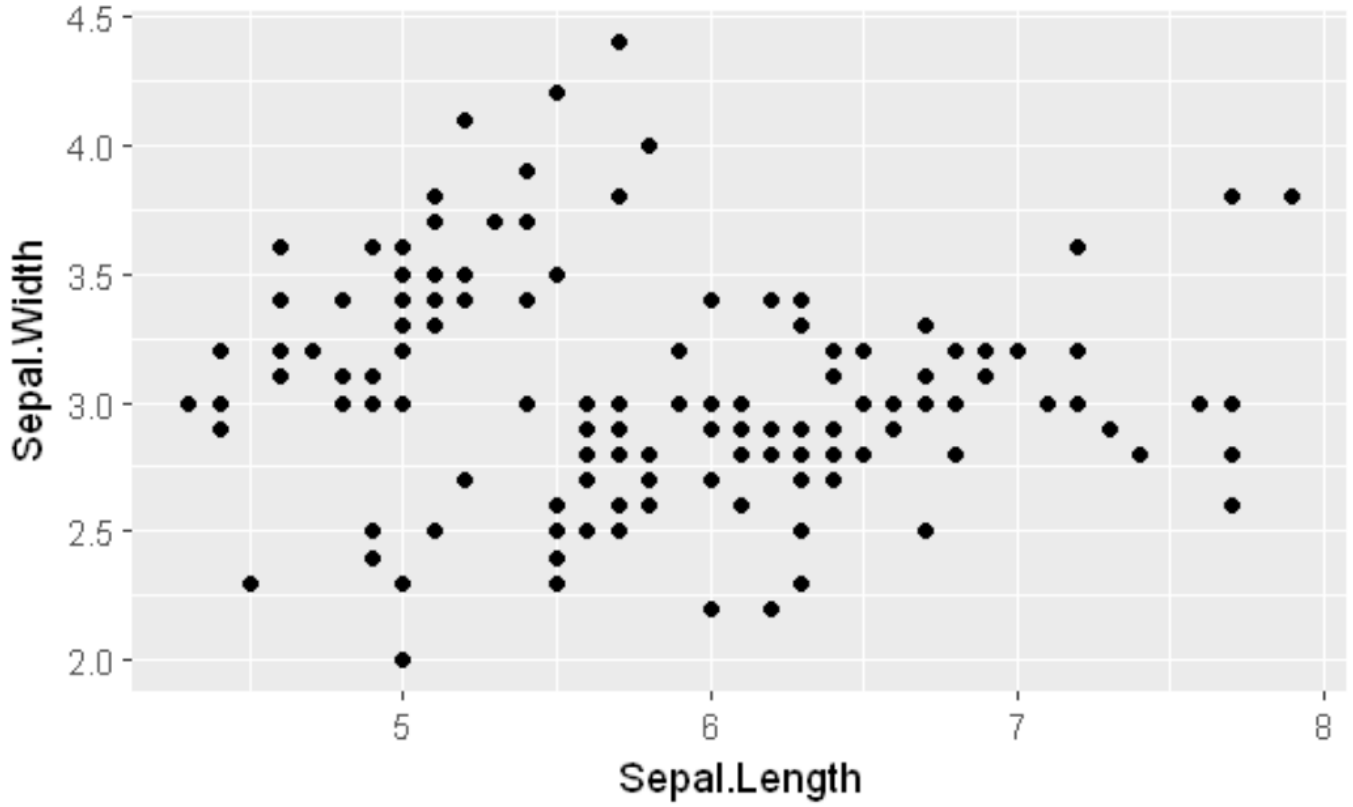
```
g <- ggplot(data=iris,
            aes(x=Sepal.Length,
                y=Sepal.Width))

# add tittle and axes labels using the labs() function
g <- g + labs(title="Scatter Plot of Sepal Length vs Sepal Width",
              x="Sepal Length", y="Sepal Width")

# add a geom_point() layer, let the color and shape vary by Species
g <- g + geom_point(aes(color=Species, shape=Species))
g
```



 # To view the plot details contained in the ggplot object, call summary on the ggplot object: summary(g)

Initialize a ggplot Object with no Data or Aesthetic

```
# initial the ggplot object without any data or aesthestic
ggplot() + geom_point(data=iris, aes(x=Sepal.Length, y=Sepal.Width))
```



Geometric Objects

- Geometric objects such as points, bars and lines are used to represent data inside the ggplot object.
- Geometric layers could be applied to the same dataset or different datasets.
- Examples of geom_functions() with required aesthetics.

```
# To plot points
geom_points(aes(x, y))
# To plot a histogram
geom_histogram(aes(x, y))
# To plot a bar chart
geom_bar(aes(x, y))
# To plot column chart
geom_col(aes(x, y))
# To plot a boxplot
geom_boxplot(aes(x))
# To plot a line
geom_line(aes(x, y))
# To plot a kernel density
geom_density(aes(x, y))
```

Geometric Objects

```
# To plot a reference line: diagonal,
horizontal or vertical
geom_abline(aes(intercept, slope))
# To plot a reference line: horizontal
geom_vline(aes(xintercept))
# To plot a reference line: vertical
geom_hline(aes(yintercept))
# To plot the pattern in the data
geom_smooth(aes(x, y))
# To add labels for each row in the data
geom_text(aes(label, x, y))
```

Geometric Objects

- Generally, optional arguments inside the aes() function of the geom_function include:
  - alpha (adjust color transparency)
  - color
  - fill
  - linetype (not for geom_points())
  - size
- The geom_function() has a parameter, **data,** which is NULL by default.
- If the data in the geom_function() is NULL, the default, the data is inherited from the plot data as specified in the call to ggplot() function.

Geometric Objects

- To find the name of a geometric object, you can check the ggplot2 website.
- You could also use the apropos() function to find names of functions if you know only part of the name.
- Example: **`apropose(geom)`**

- To separate multiple lines of code, use the plus sign at the end of each layer:

```
ggplot(data, aes(x, y) +

labs(title="scatter plot") +

geom_point(aes(color=z), size=3)
```

Geometric Objects: Checking Names

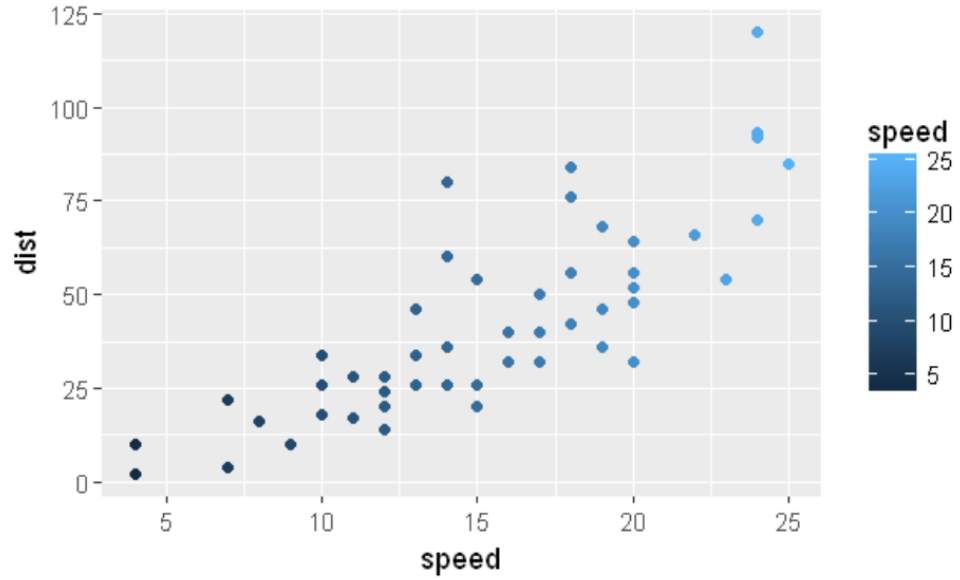
```
apropos("geom")
```

'dgeom' 'Geom' 'geom_abline' 'geom_area' 'geom_bar' 'geom_bin2d' 'geom_blank' 'geom_boxplot' 'geom_col' 'geom_contour' 'geom_count' 'geom_crossbar' 'geom_curve' 'geom_density' 'geom_density_2d' 'geom_density2d' 'geom_dotplot' 'geom_errorbar' 'geom_errorbarh' 'geom_freqpoly' 'geom_hex' 'geom_histogram' 'geom_hline' 'geom_jitter' 'geom_label' 'geom_line' 'geom_linerange' 'geom_map' 'geom_path' 'geom_point' 'geom_pointrange' 'geom_polygon' 'geom_qq' 'geom_quantile' 'geom_raster' 'geom_rect' 'geom_ribbon' 'geom_rug' 'geom_segment' 'geom_smooth' 'geom_spoke' 'geom_step' 'geom_text' 'geom_tile' 'geom_violin' 'geom_vline' 'GeomAbline' 'GeomAnnotationMap' 'GeomArea' 'GeomBar' 'GeomBlank' 'GeomBoxplot' 'GeomCol' 'GeomContour' 'GeomCrossbar' 'GeomCurve' 'GeomCustomAnn' 'GeomDensity' 'GeomDensity2d' 'GeomDotplot' 'GeomErrorbar' 'GeomErrorbarh' 'GeomHex' 'GeomHline' 'GeomLabel' 'GeomLine' 'GeomLinerange' 'GeomLogticks' 'GeomMap' 'GeomPath' 'GeomPoint' 'GeomPointrange' 'GeomPolygon' 'GeomQuantile' 'GeomRaster' 'GeomRasterAnn' 'GeomRect' 'GeomRibbon' 'GeomRug' 'GeomSegment' 'GeomSmooth' 'GeomSpoke' 'GeomStep' 'GeomText' 'GeomTile' 'GeomViolin' 'GeomVline' 'pgeom' 'qgeom' 'rgeom' 'update_geom_defaults'

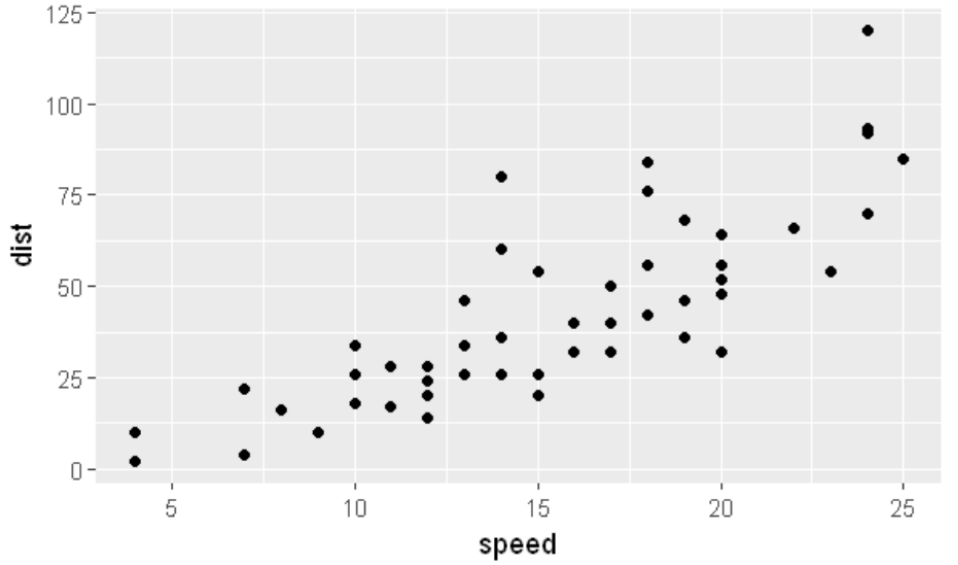
Geometric Objects

- You can overwrite default aesthetic attributes in the ggplot object by setting different values or NULL in a layer.

```
ggplot(data=cars, aes(x=speed, y=dist, color=speed)) +
geom_point(aes())
```



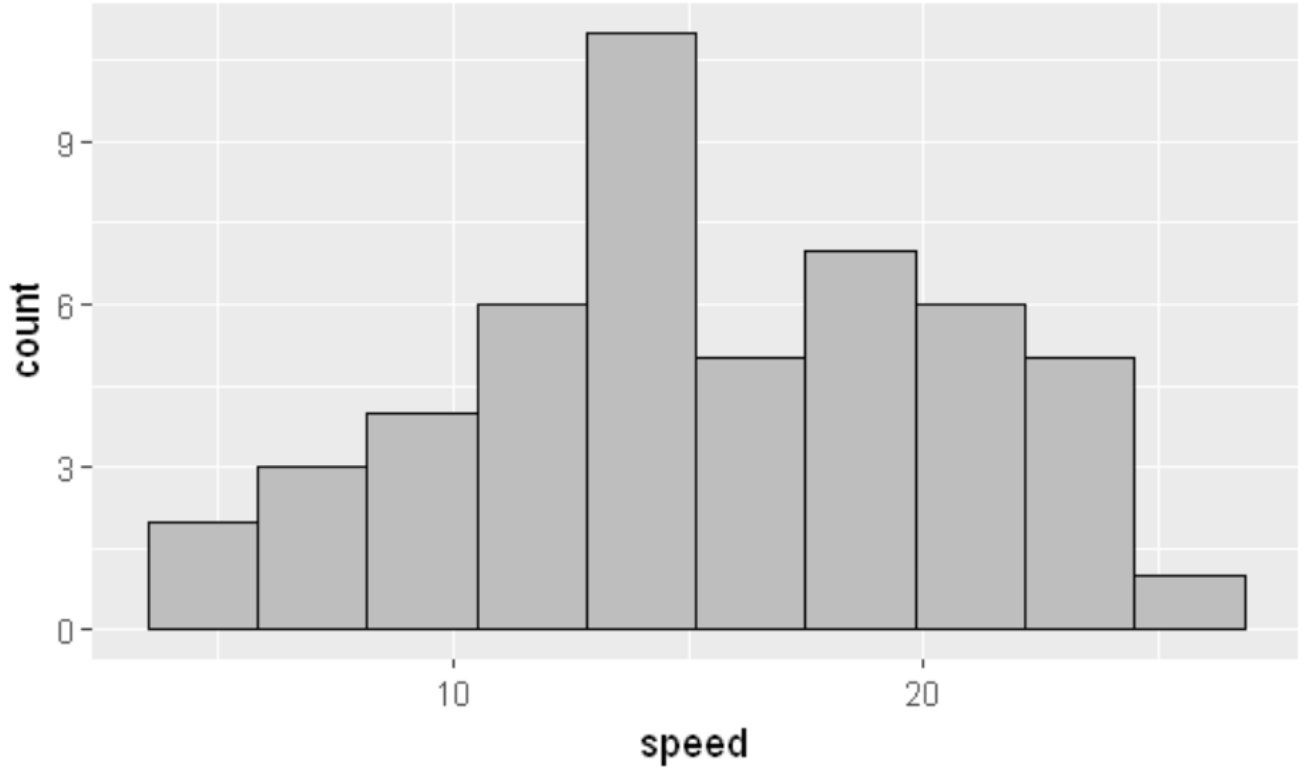

```
# override the color
ggplot(data=cars, aes(x=speed, y=dist, color=speed)) +
geom_point(aes(color=NULL))
```



Geometric Objects

Histogram

```
ggplot(data=cars, aes(x=speed)) +
geom_histogram(bins=10, fill="gray", color="black")
```



Geometric Objects

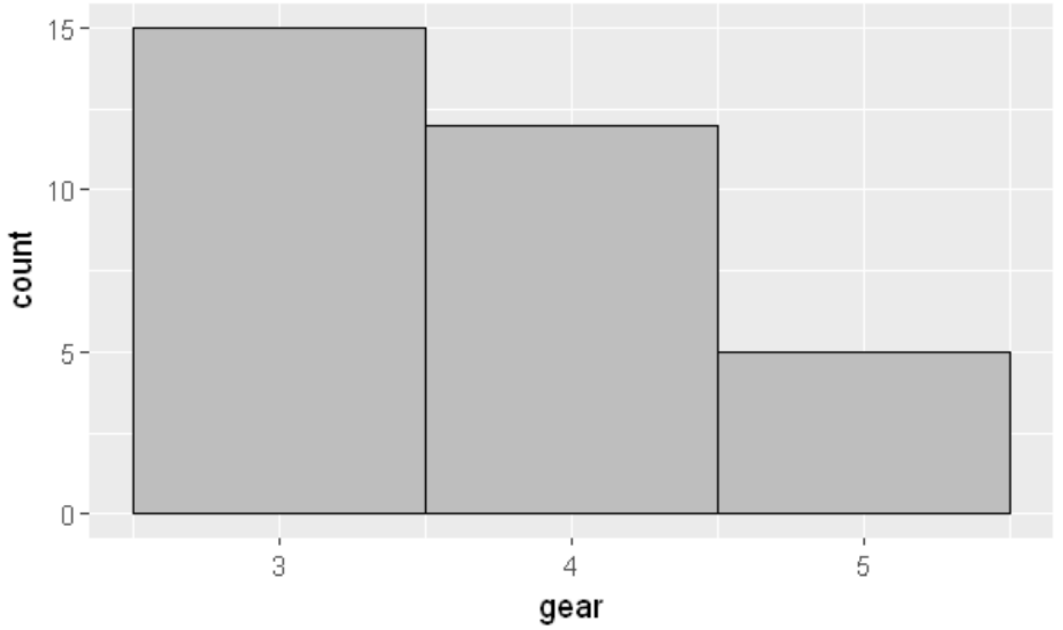
Barchart

Plots the counts of the values categorical variable

```
: head(mtcars)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

```
ggplot(data=mtcars, aes(x=gear)) +
geom_bar(width=1, fill="gray", color="black")
```

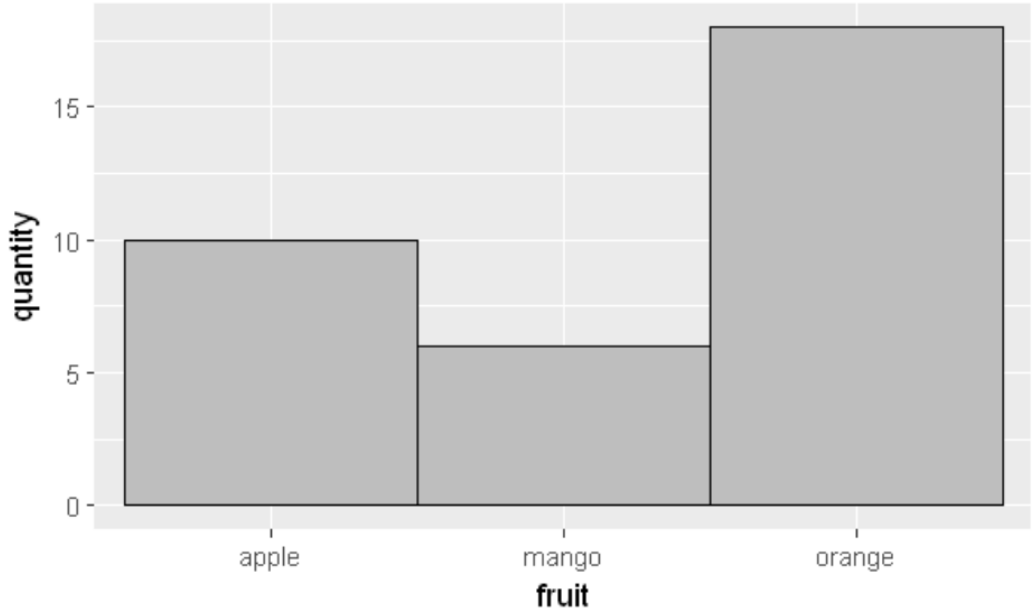


Geometric Objects

Column plot

Similar to a bar plot but plots the values of a categorical variable against the values of another variable.

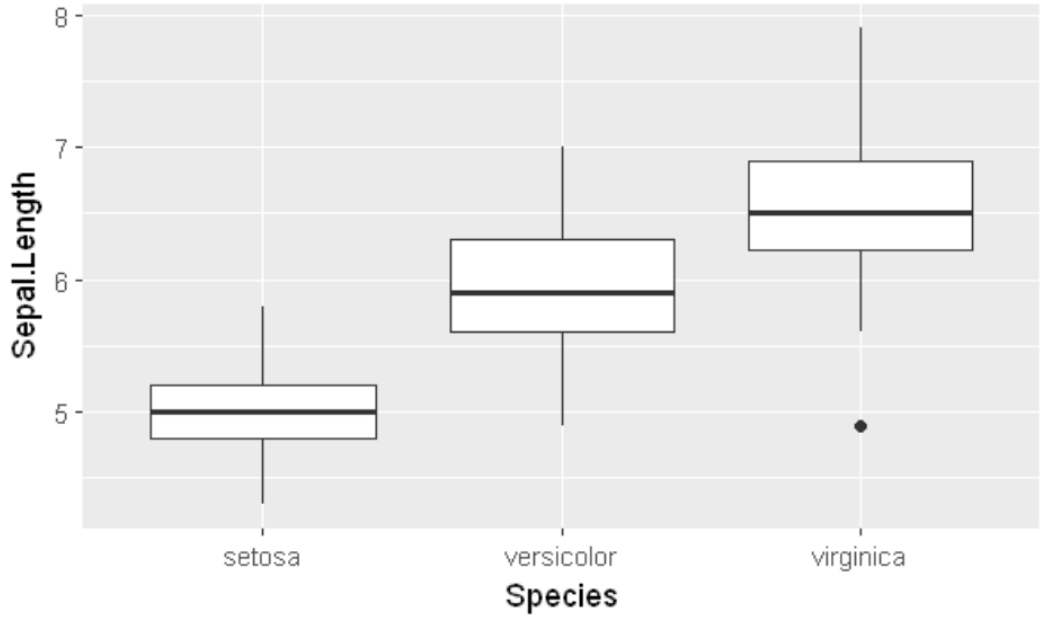
```
# creat fruit data
fruit_data <- data.frame(fruit=c("apple", "orange", "mango"),
                         quantity=c(10, 18, 6))

# plot a column plot
ggplot(data=fruit_data, aes(x=fruit, y=quantity)) +
geom_col(width=1, fill="gray", col="black")
```



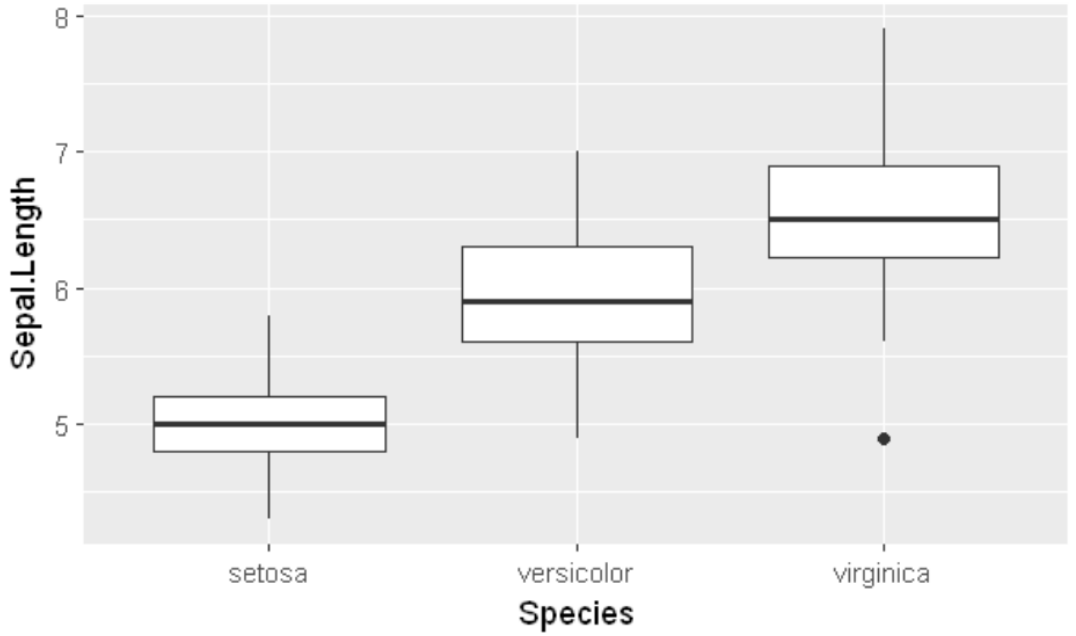
Geometric Objects

Boxplot

```
ggplot(data=iris, aes(x=Species, y=Sepal.Length)) +
geom_boxplot()
```



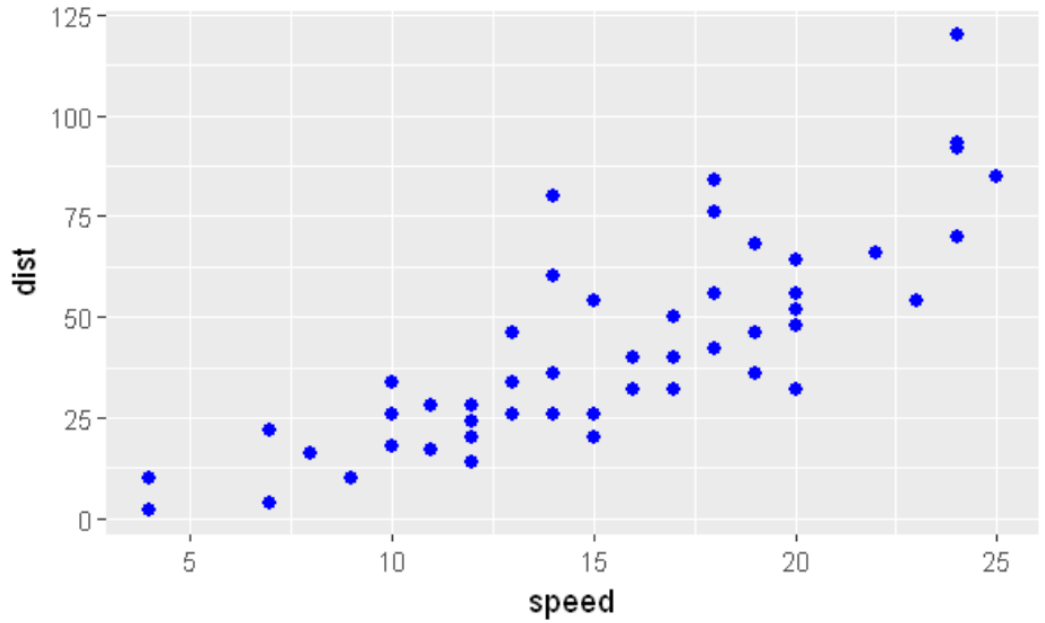

```
# use a stat transformation
ggplot(data=iris, aes(x=Species, y=Sepal.Length)) +
stat_boxplot()
```



Geometric Objects

Scatter plot

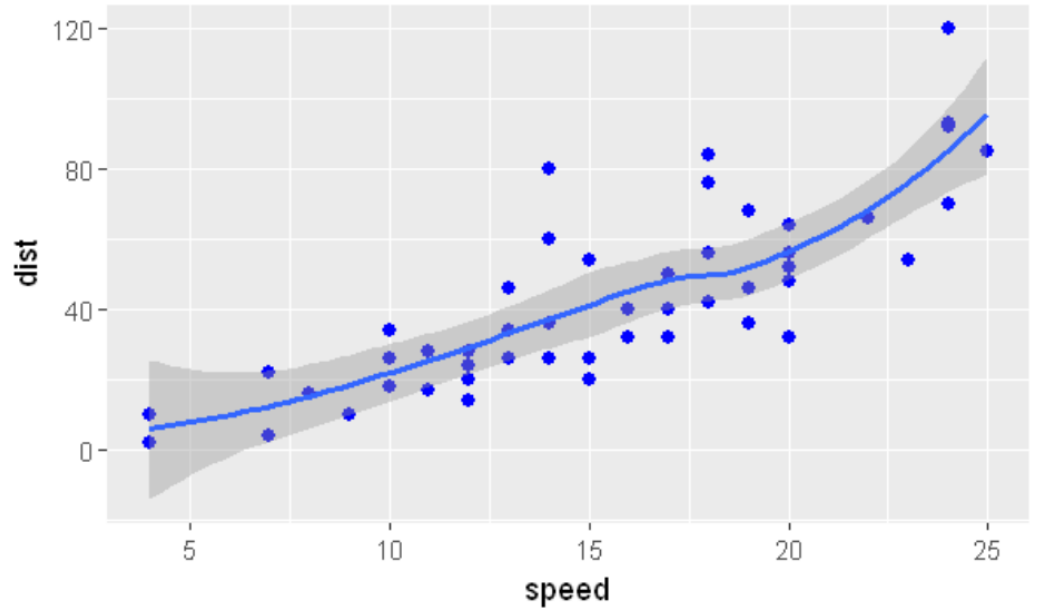
```
ggplot(data=cars, aes(x=speed, y=dist)) +
geom_point(size=2, color="blue")
```



method="lm" could be used to fit a straight line

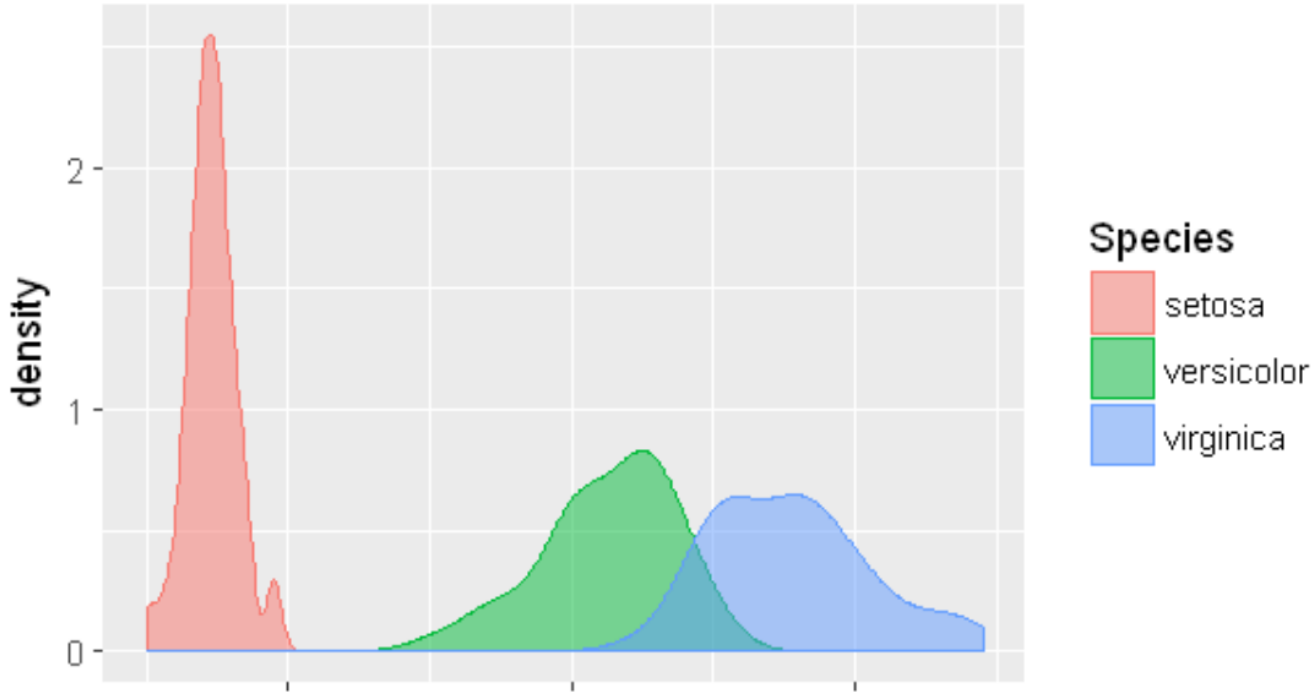
```
ggplot(data=cars, aes(x=speed, y=dist)) +
geom_point(size=2, color="blue") +
geom_smooth()
```

```
`geom_smooth()` using method = 'loess'
```



Geometric Object: geom_density()

```
ggplot(data=iris, aes(x=Petal.Length,color=Species,fill=Species)) +
geom_density(alpha=0.5)
```

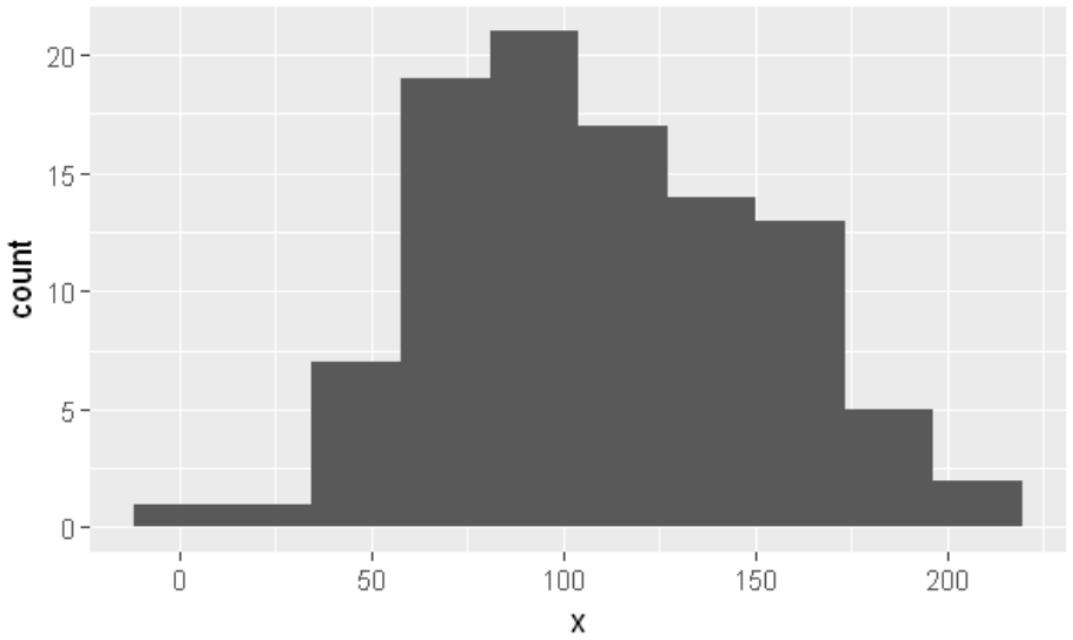


To break multiple lines of code, add the plus sign towards the end

Stat

- Stat or statistical transformation is a statistical manipulation applied to the data, usually to summarize the data.
- For example, the stat_bin() function summarizes the data in bins to create a histogram.

```
# stat_bin() summarizes the data in bins
# to create a histogram
ggplot(data=dat, aes(x=x)) +
stat_bin(bins=10)
```



Stat

- The stat_x() function is the general form of the stat transformation function where x represents the statistical transformation.
- Note that each geometry function comes with a default statistical transformation.

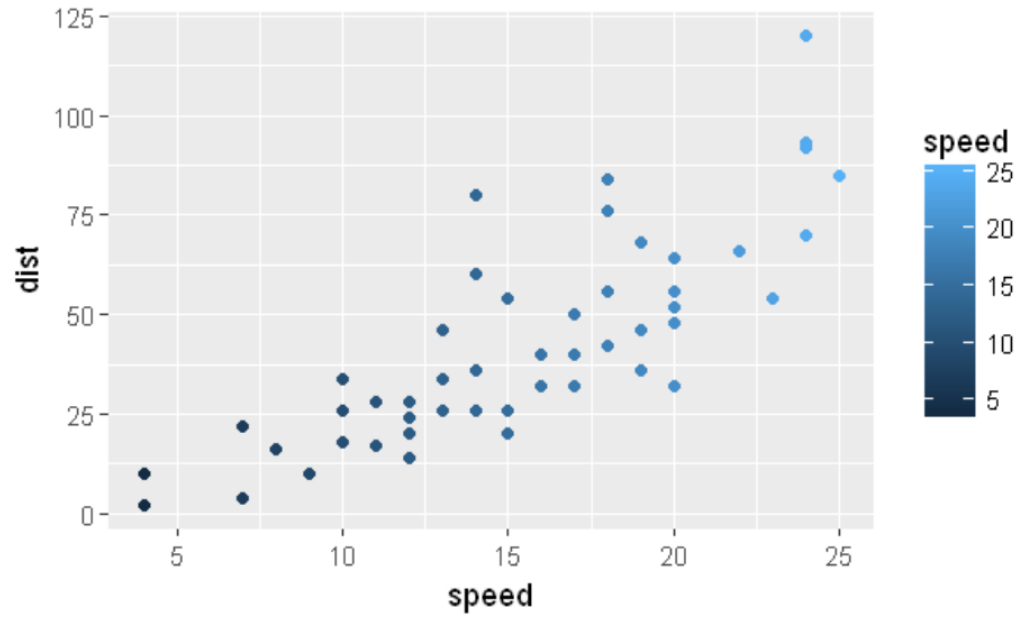
geom() function	Default stat
geom_abline()	abline
geom_bar()	bin
geom_density()	density
geom_histogram()	bin
geom_hline()	hline
geom_point()	identity
geom_vline()	vline

Stat

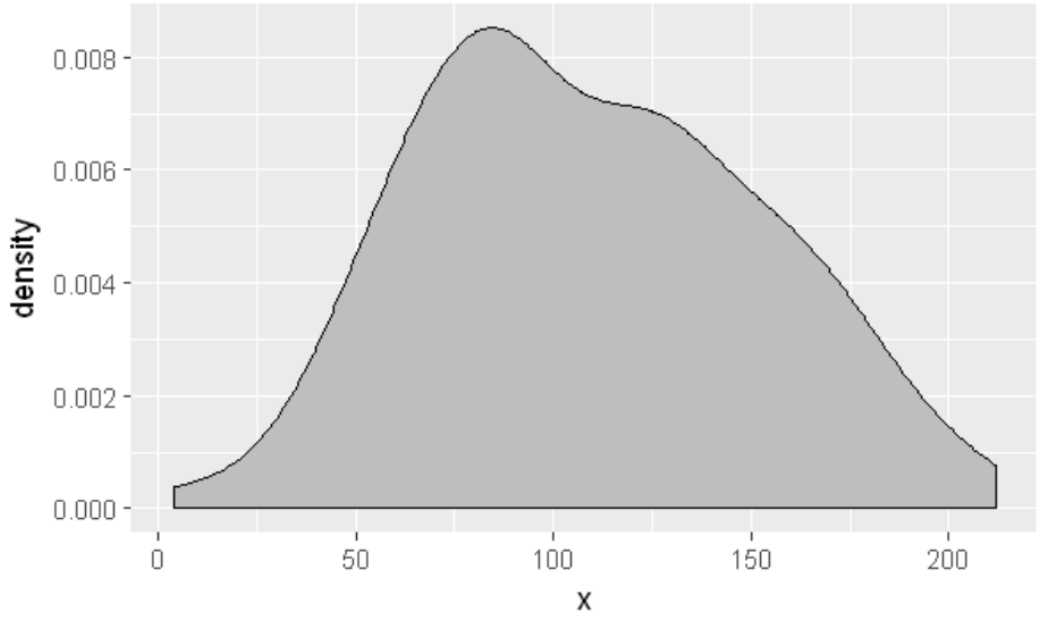
Stat function	Description
Stat_bin()	Splits data into bins for histograms
stat_density()	Calculates the kernel density estimate for a density plot
Stat_function()	Superimposes a function to a plot
Stat_identify()	Plots data without any statistical transformation
Stat_smooth()	Add a smoother line
Etc.	

Stat: Examples

```
# the stat_identity () function
# plots without any transformation
ggplot(data=cars, aes(x=speed, y=dist, color=speed)) +
stat_identity()
```

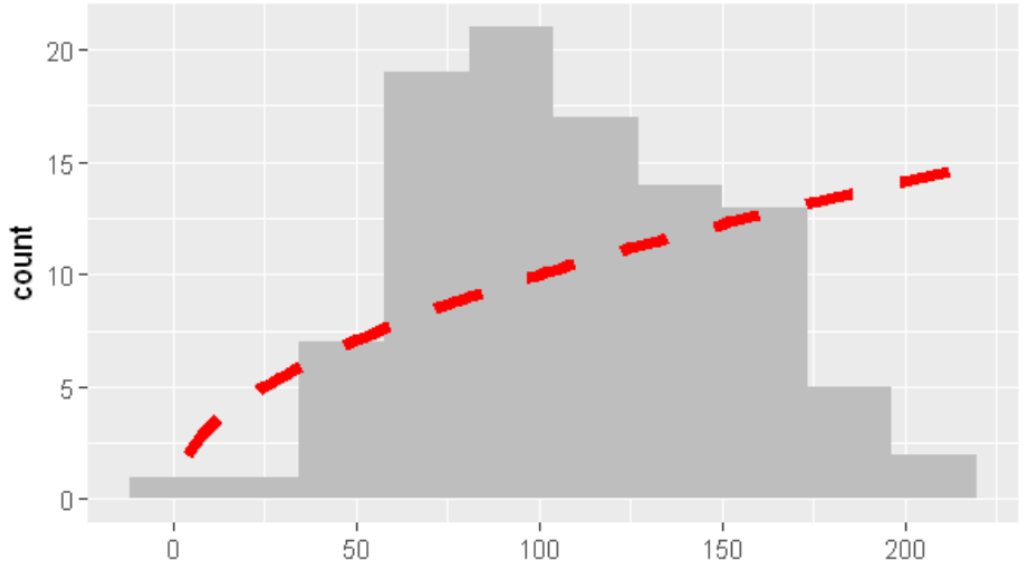



```
# stat_density creates a kernel density
ggplot(data=dat, aes(x=x)) +
stat_density(fill="gray", col="black")
```

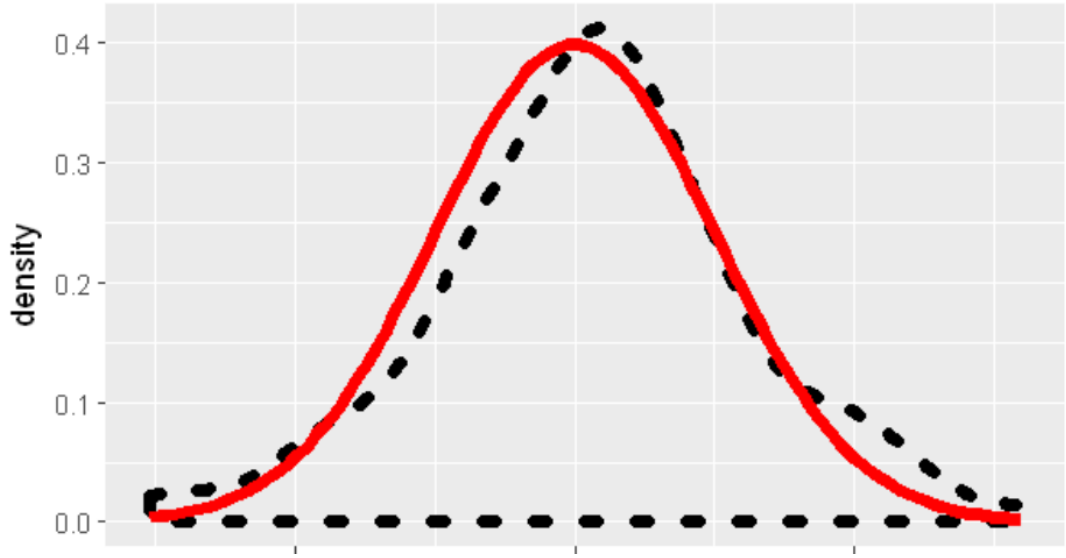


Stat

```r
# define a square root function
func <- function(x){x^0.5}

# plot the function using stat_function
ggplot(data=dat, aes(x=x)) +
stat_bin(bins=10, fill="gray") +
stat_function(fun=func, color="red", size=2, linetype="dashed")
```



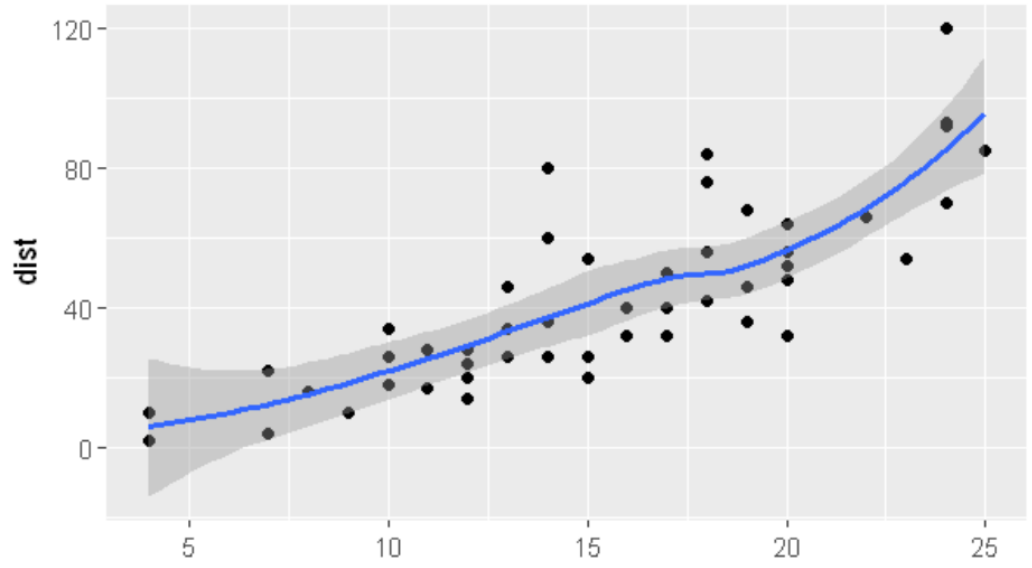

```r
# superimpose a function on top of an existing plot
# using the stat_function()
set.seed(2020)
values <- data.frame(x=rnorm(100))
ggplot(data=values, aes(x=x)) +
geom_density(color="black", linetype="dotted", size=2) +
stat_function(fun=dnorm, color="red", size=2)
```



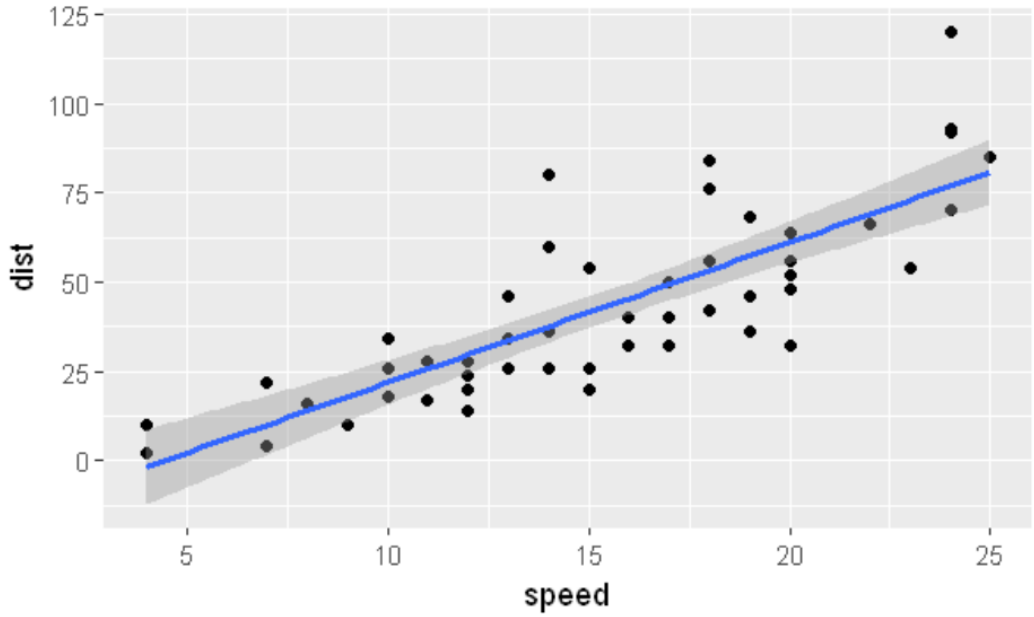
Stat

```
# find the pattern in the data
# use the default method = "loess" inside the stat_smooth() function
ggplot(data=cars, aes(x=speed, y=dist)) +
geom_point() +
stat_smooth()
```

```
`geom_smooth()` using method = 'loess'
```



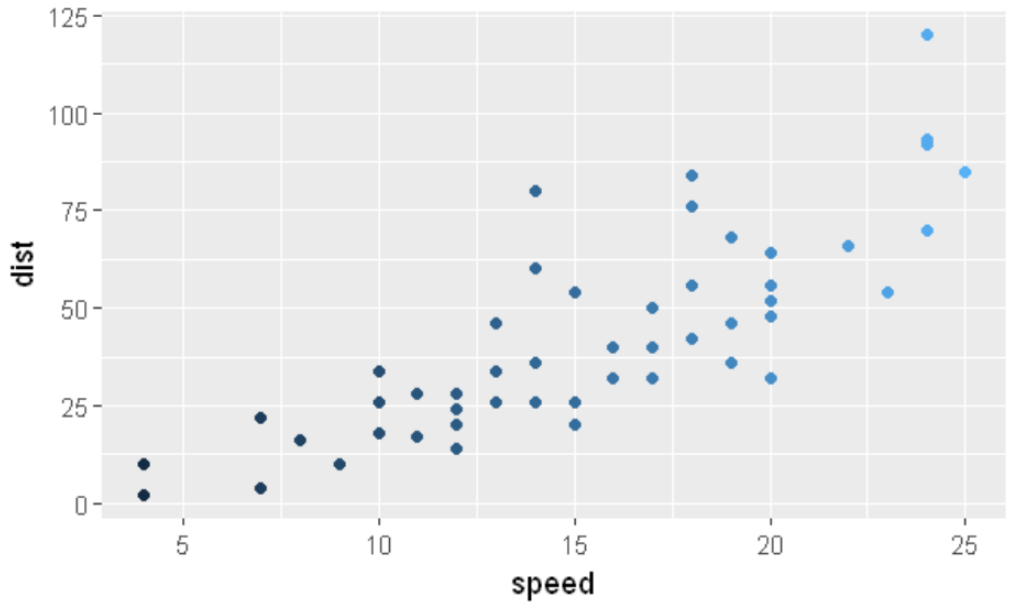

```
# find the pattern in the data
# use the method = "lm" inside the stat_smooth() function
ggplot(data=cars, aes(x=speed, y=dist)) +
geom_point() +
stat_smooth(method="lm")
```



Themes

- Themes are useful to customize non-data components of the plot.
- For example, the position of the legend can be change to the “top”, “bottom”, “left” or “right”. The legend can be overridden by setting legend position to “none”

```
# override the legend using the theme layer
ggplot(data=cars, aes(x=speed, y=dist, color=speed)) +
geom_point(aes()) +
theme(legend.position="none")
```

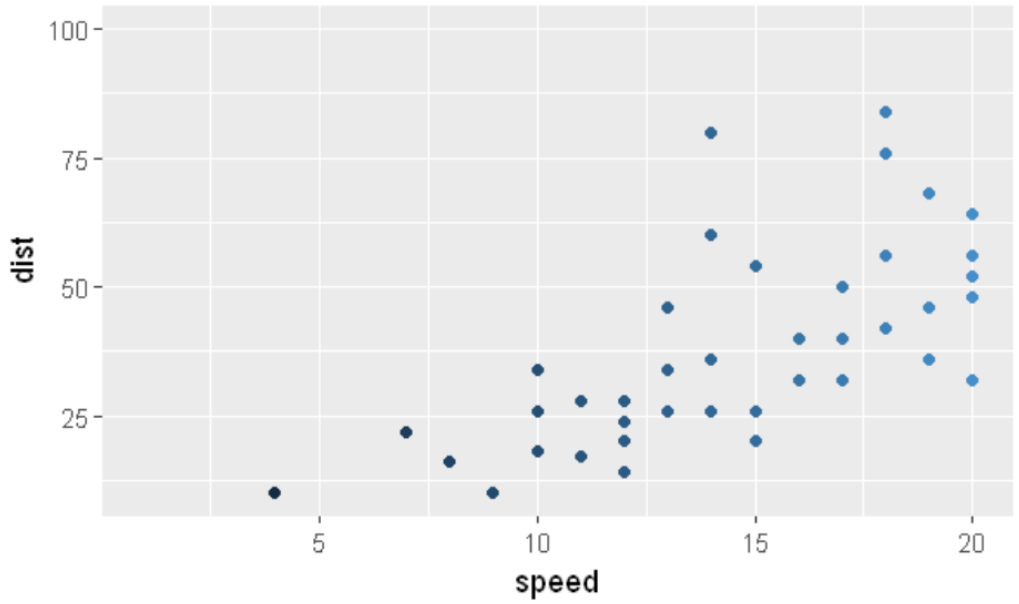


https://ggplot2.tidyverse.org/reference/theme.html

The Coordinate System

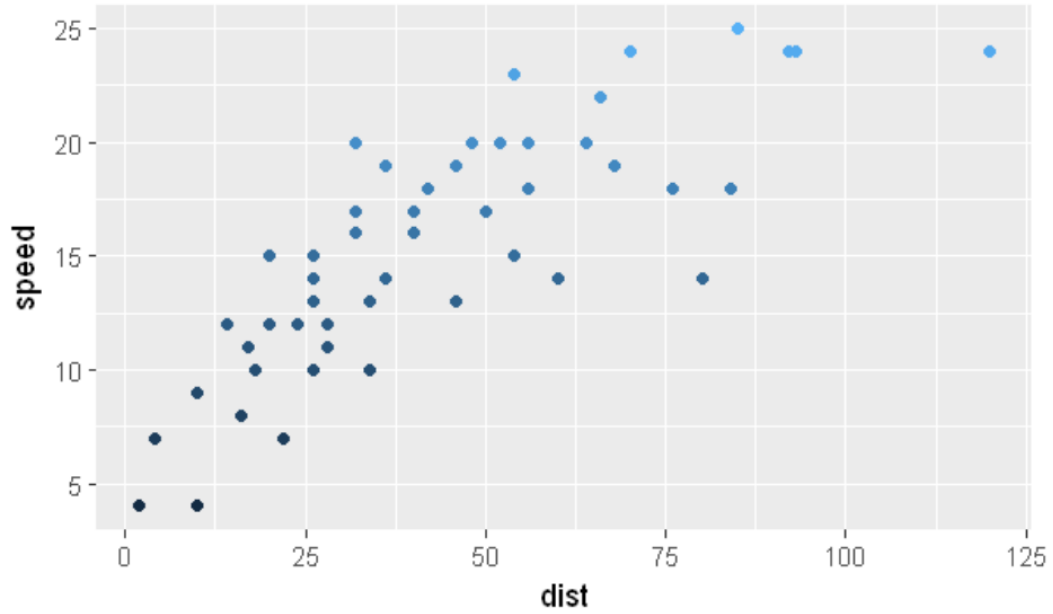
```
# The cartesian coordinate with x and y limits set as vectors
coord_cartesian(xlim, ylim)
# Flips the x and y axes of the cartesian coordinate.
coord_flp()
# The polar coordinate system
Coord_polar(thetha,start, direction)
# Define the ration between y and x axes
Coord_fixed(ratio)
```

```
# specify the limits of the coordinate system
ggplot(data=cars, aes(x=speed, y=dist, color=speed)) +
geom_point(aes()) +
theme(legend.position="none") +
coord_cartesian(xlim=1:20, ylim=10:100)
```

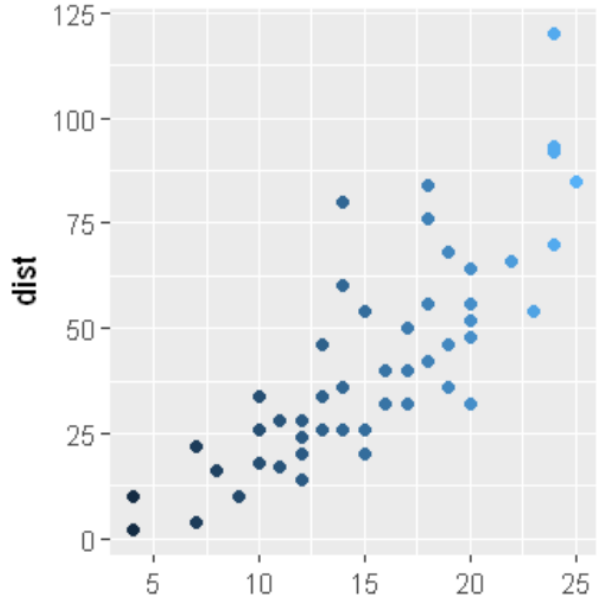


The Coordinate System

```
# flip the axes of the coordinate system
ggplot(data=cars, aes(x=speed, y=dist, color=speed)) +
geom_point(aes()) +
theme(legend.position="none") +
coord_flip()
```




```
# flip the axes of the coordinate system
ggplot(data=cars, aes(x=speed, y=dist, color=speed)) +
geom_point(aes()) +
theme(legend.position="none") +
coord_fixed(ratio=.2)
```



Faceting

- This is the mechanism of automatically laying out multiple plots on one page.
- The faceting option splits the data into subsets and each subset is plotted separately.
- However, the separate plots are formatted in an overall plot page with a header at the top.
- Two main ways to perform faceting in ggplot2 are:
  - Grid faceting
  - Wrap faceting

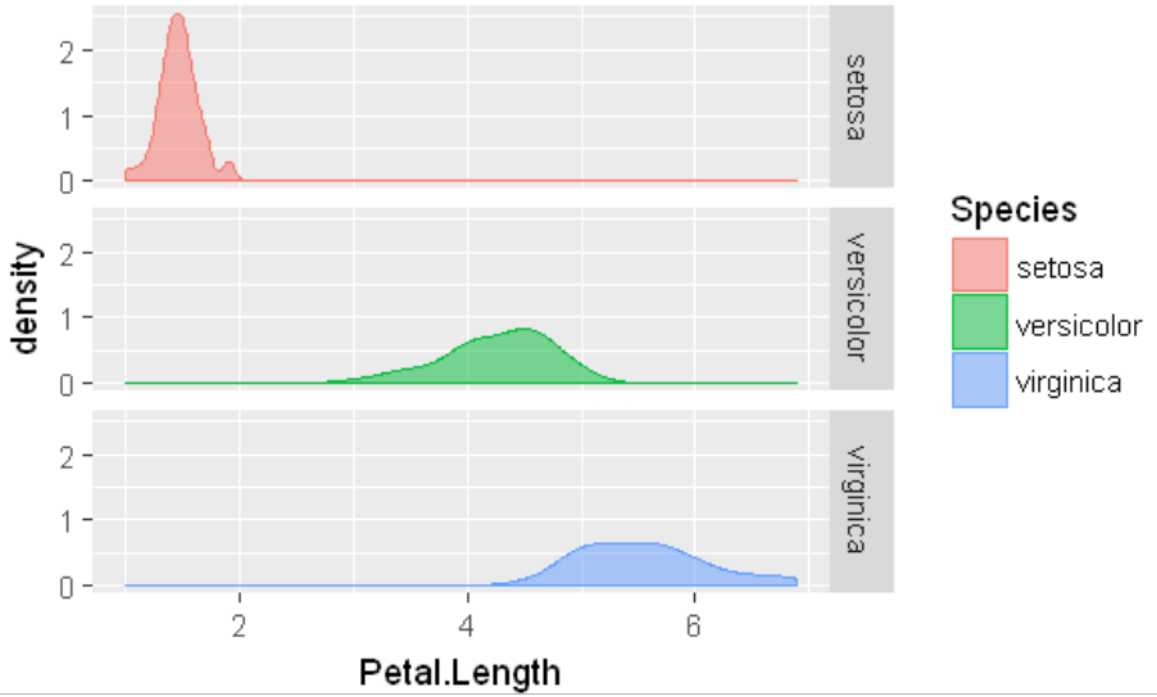
Faceting: grid faceting

- Grid faceting is probably used most of the time.
- Grid faceting splits data into subgroups relative to two or more variables.
- The split data is the used to construct subplot for a different combination of variables.
- At least, two variables are provided to the face_grid() function.
- Generally, facet_grid(x~y) will split the plot on x into rows and will split on y into columns.
- It is important that both x and y are categorical or discrete.

Faceting: grid faceting

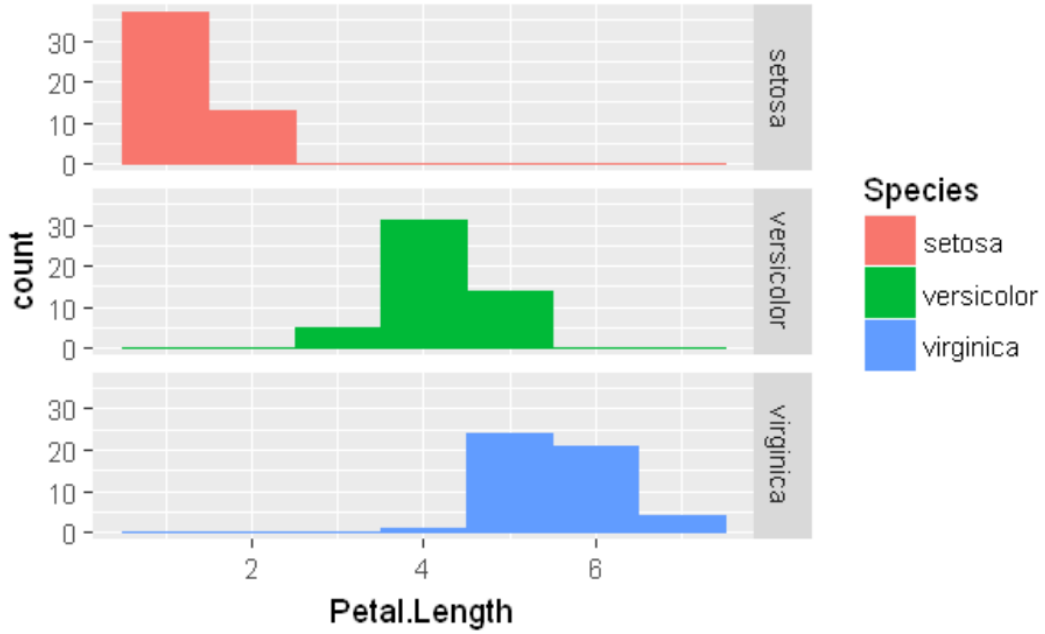
- face_grid(x~.) means you want to split the plot you have created by x.
- Here, we split the density plot of petal length by Species.

```
ggplot(data=iris, aes(x=Petal.Length,color=Species,fill=Species)) +
geom_density(alpha=0.5)  +
facet_grid(Species~.)
```

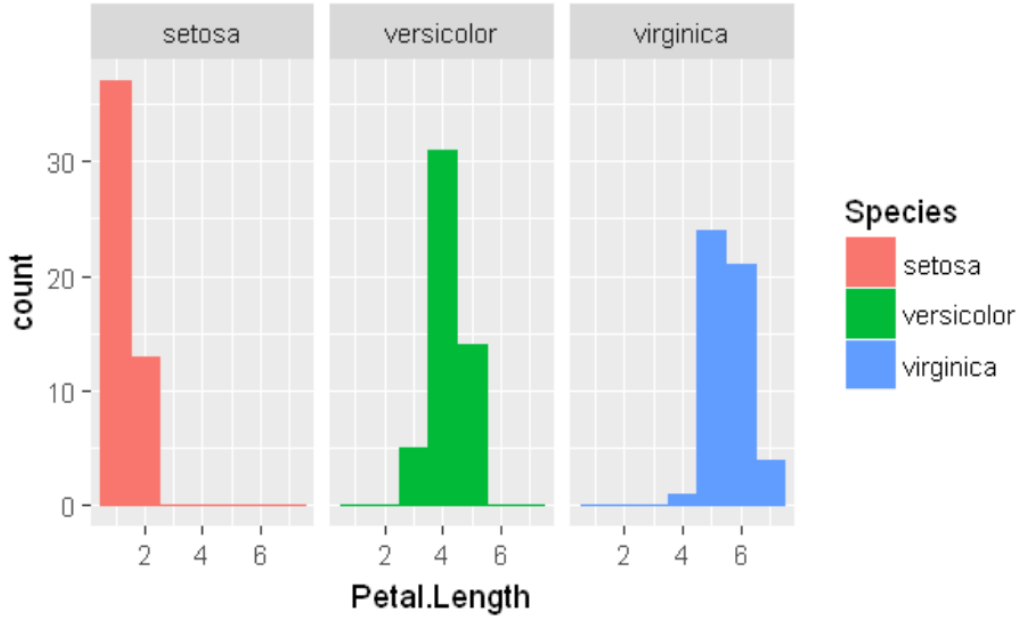


Faceting: grid faceting

```
# split the plot on Species into rows
ggplot(data=iris, aes(x=Petal.Length,color=Species,fill=Species)) +
geom_histogram(binwidth=1)  +
facet_grid(Species~.)
```




```
# split the plot on Species into columns
ggplot(data=iris, aes(x=Petal.Length,color=Species,fill=Species)) +
geom_histogram(binwidth=1)  +
facet_grid(.~Species)
```

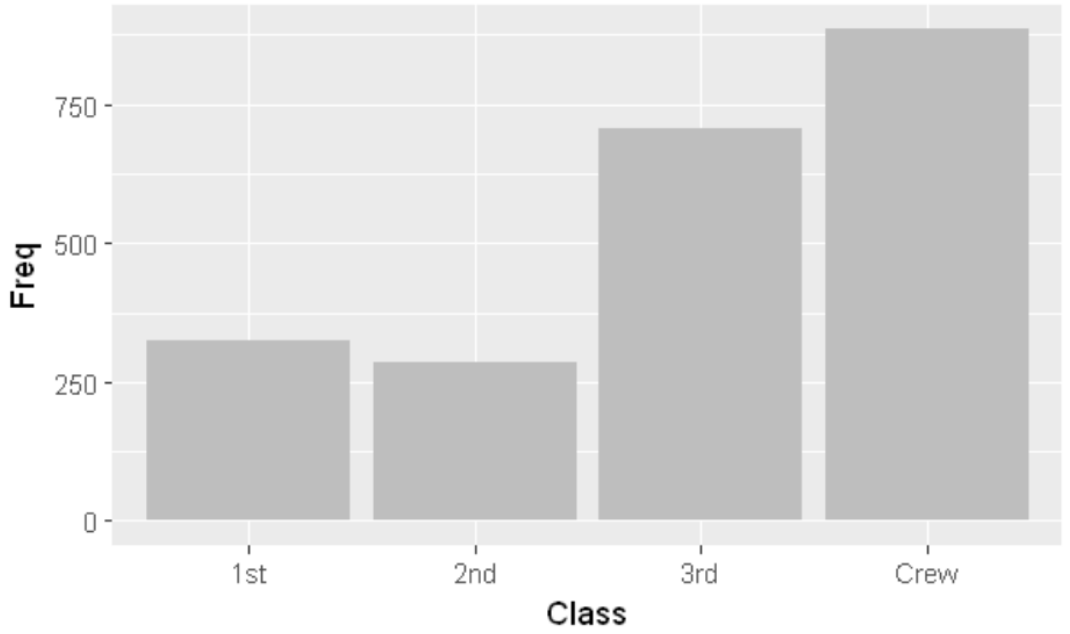


Faceting: grid faceting

```
titanic <- data.frame(Titanic)
head(titanic)
```

Class	Sex	Age	Survived	Freq
1st	Male	Child	No	0
2nd	Male	Child	No	0
3rd	Male	Child	No	35
Crew	Male	Child	No	0
1st	Female	Child	No	0
2nd	Female	Child	No	0

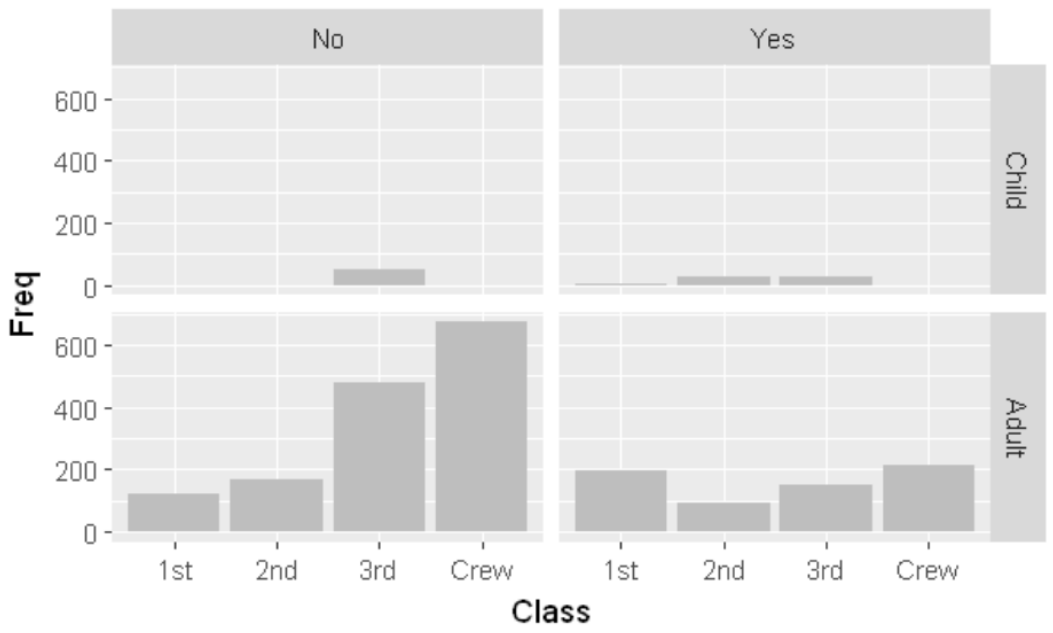
```
ggplot(data=titanic, aes(x=Class)) +
geom_col(width=0.9, aes(y=Freq), fill="gray")
```



Faceting: grid faceting

- The plot is split on two variables. This can help us see the interaction between two variables on separate plots.

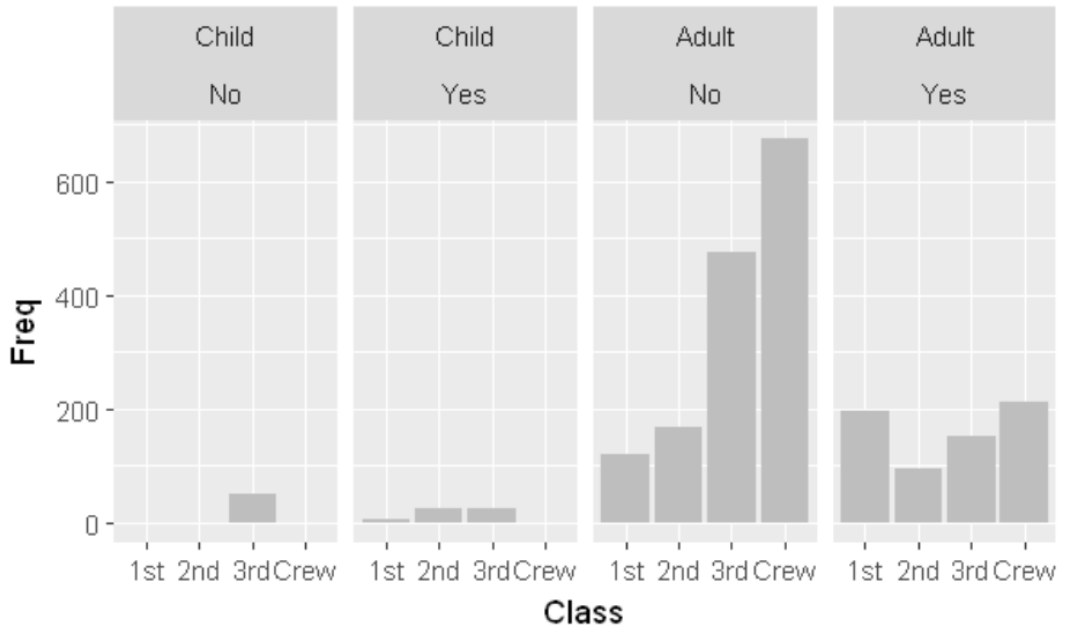
```
# plot facet grid for two variables
ggplot(data=titanic, aes(x=Class)) +
geom_col(width=0.9, aes(y=Freq), fill="gray") +
facet_grid(Age~Survived)
```



Faceting: grid faceting

- We can also split the data on two variables into columns only (or even into rows only).
- The dot (.) before the tilde (~) implies we will not split the plot into rows.

```
# plot facet grid for two variables
ggplot(data=titanic, aes(x=Class)) +
geom_col(width=0.9, aes(y=Freq), fill="gray") +
facet_grid(.~Age + Survived)
```



Faceting: wrap faceting

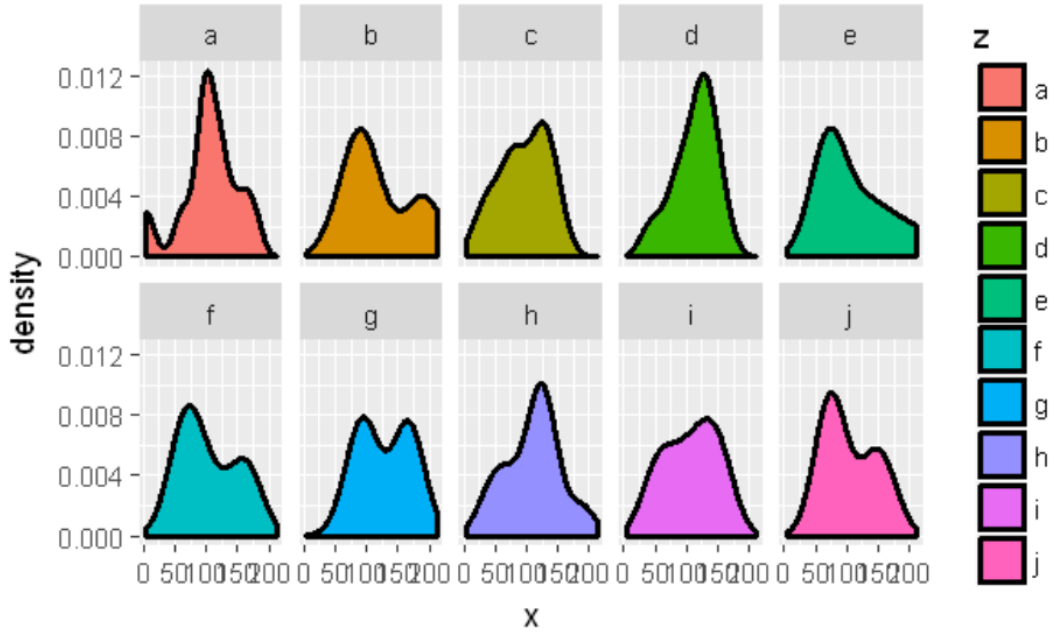
- Wrap faceting produces a single ribbon of plots that are spread along one or more rows.
- The facet_wrap() function is used.
- The arguments are provided only after the tilde (~) sign.
- So we could use:
  - facet_wrap(~x, nrow) for a split on a single variable.
  - facet_wrap(~x+y+z, nrow) to split on multiple variables x, y, and z.
- Note that, no dot(.) is included or precedes the tilde as it was the case in grid faceting.

Faceting: wrap faceting

```
dat <- data.frame(x=rnorm(100, mean=100, sd=50),
                  z=rep(letters[1:10], 10))
head(dat)
```

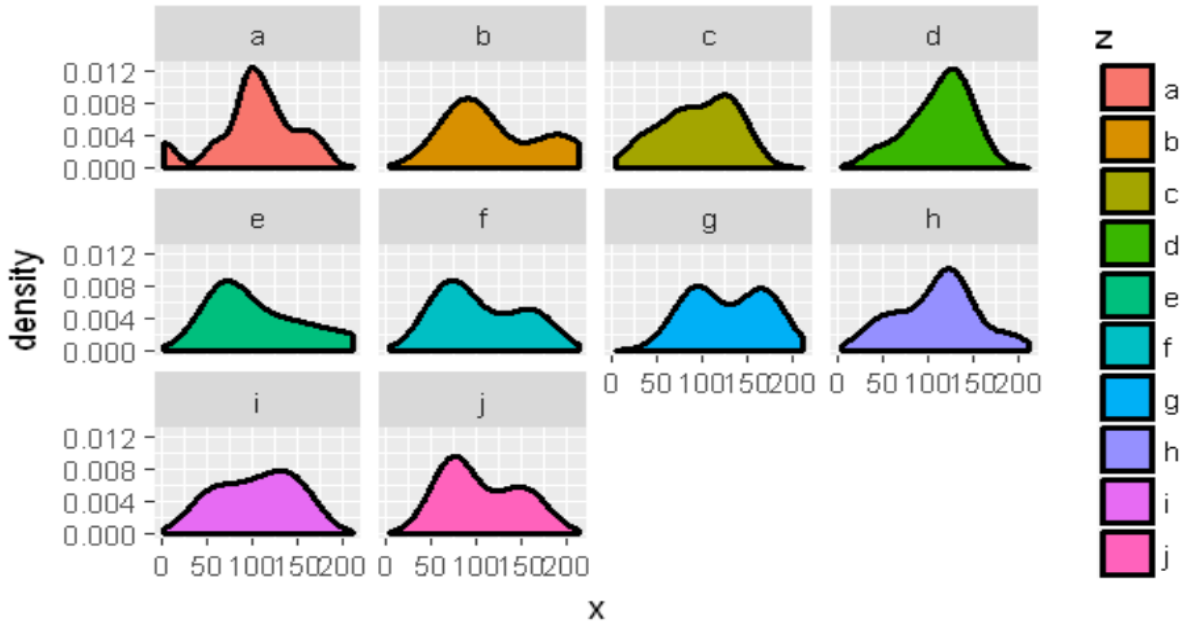
x	z
3.89374	a
58.08420	b
75.44894	c
130.63757	d
60.70485	e
82.41253	f

```
# arrange plots into two rows
ggplot(data=dat, aes(x=x, fill=z)) +
geom_density(size=1) +
facet_wrap(~z, nrow=2)
```



Faceting: wrap faceting

```
# arrange plots into three rows
ggplot(data=dat, aes(x=x, fill=z)) +
geom_density(size=1) +
facet_wrap(~z, nrow=3)
```



Tidyverse

- Other members of the tidyverse are as follows:

```
library(tidyverse)
#> ── Attaching packages ──────────────────
#> ✓ ggplot2 3.2.1     ✓ purrr   0.3.3
#> ✓ tibble  2.1.3     ✓ dplyr   0.8.3
#> ✓ tidyr   1.0.0     ✓ stringr 1.4.0
#> ✓ readr   1.3.1     ✓ forcats 0.4.0
```

- package::function() can be used when calling a function in R instead of just function().
- package::function() is used when you want to be explicit about the package to which the function belongs. For example, ggplot2::ggplot(). This is helpful for solving conflicts in the name space.

Quick Datasets in R

Here are a few datasets in base R that you can quick use for experimenting, plotting and data analysis.

```
library(datasets) # to load datasets
data() # to view the various datasets
# load the specific data
```

- ✓ `data(iris) # the iris data: predict specie from flower measurements`
- ✓ `data(BostonHousing) # the Boston housing data: predict house prices`
- ✓ `data(Glass) # predict glass type from chemical properties`
- ✓ `data(Titanic)`

References

- https://ggplot2.tidyverse.org/reference/index.html#section-layer-geoms
- http://statseducation.com/Introduction-to-R/modules/graphics/ggplot2/