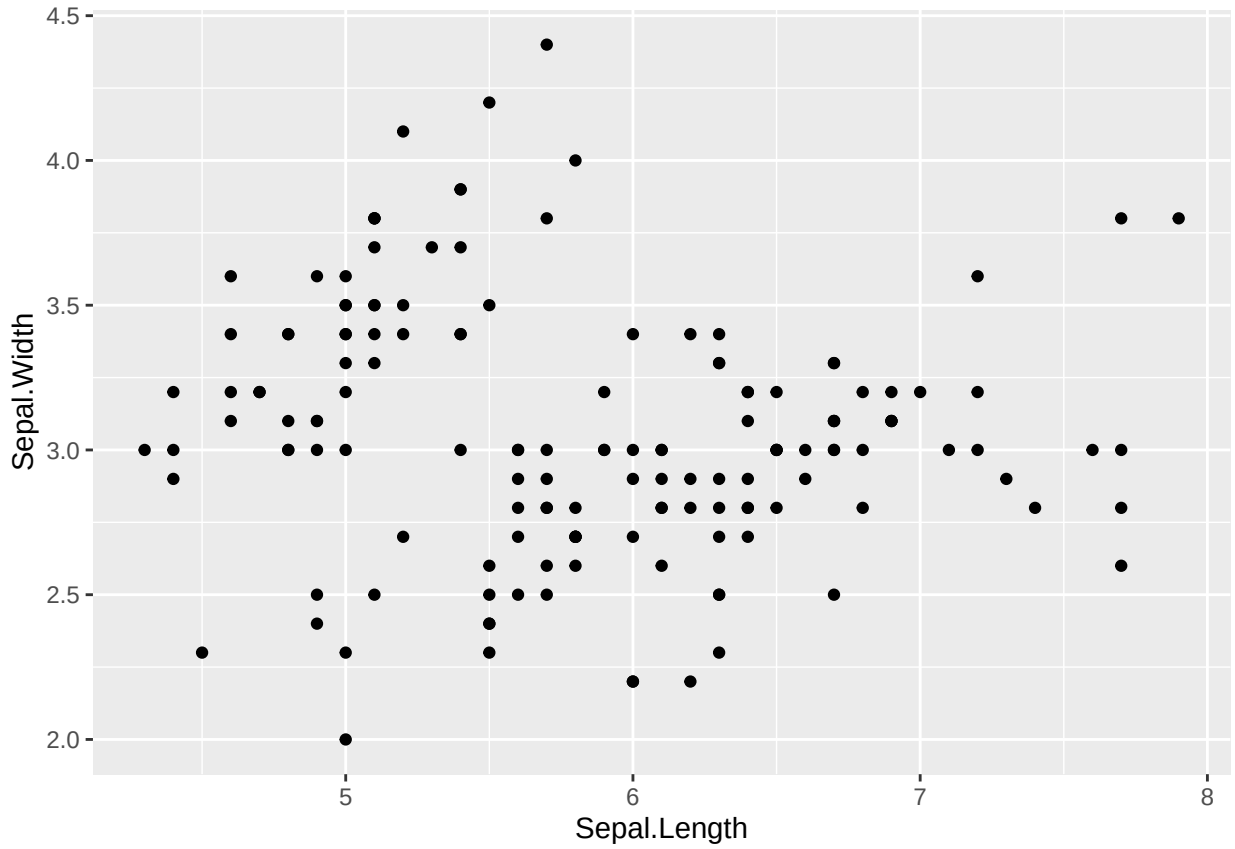
Datacamp_Data Visualization with ggplot2 (Part 1)___qplot and wrap-up

dizhen

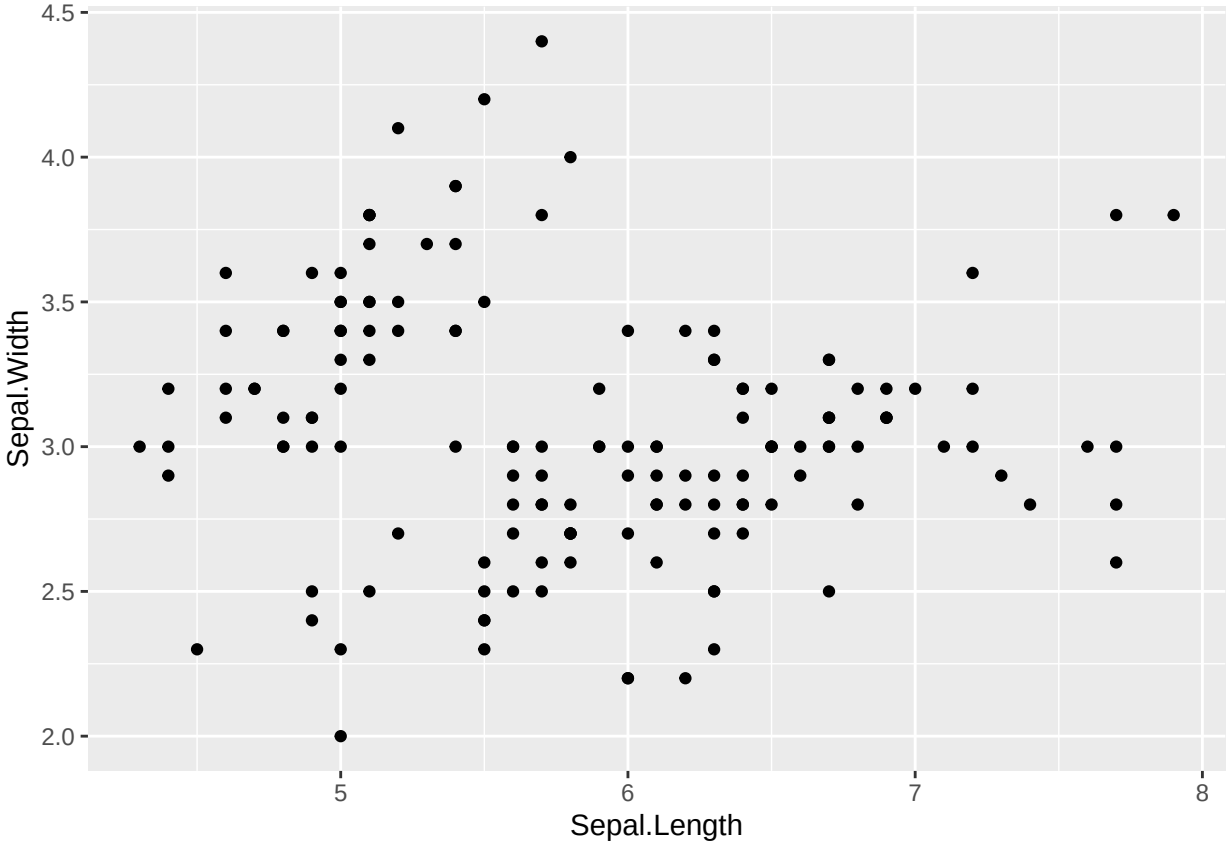
2020/4/9

qplot

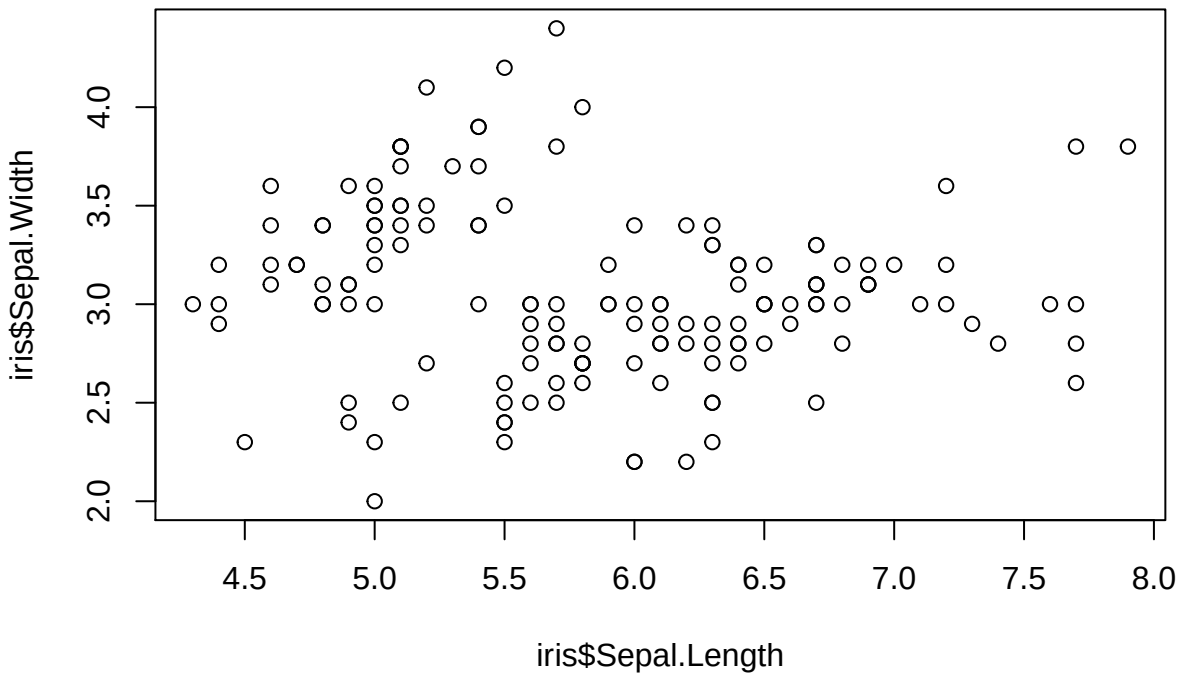
```
library("ggplot2")

# ggplot
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) +
 geom_point()
```



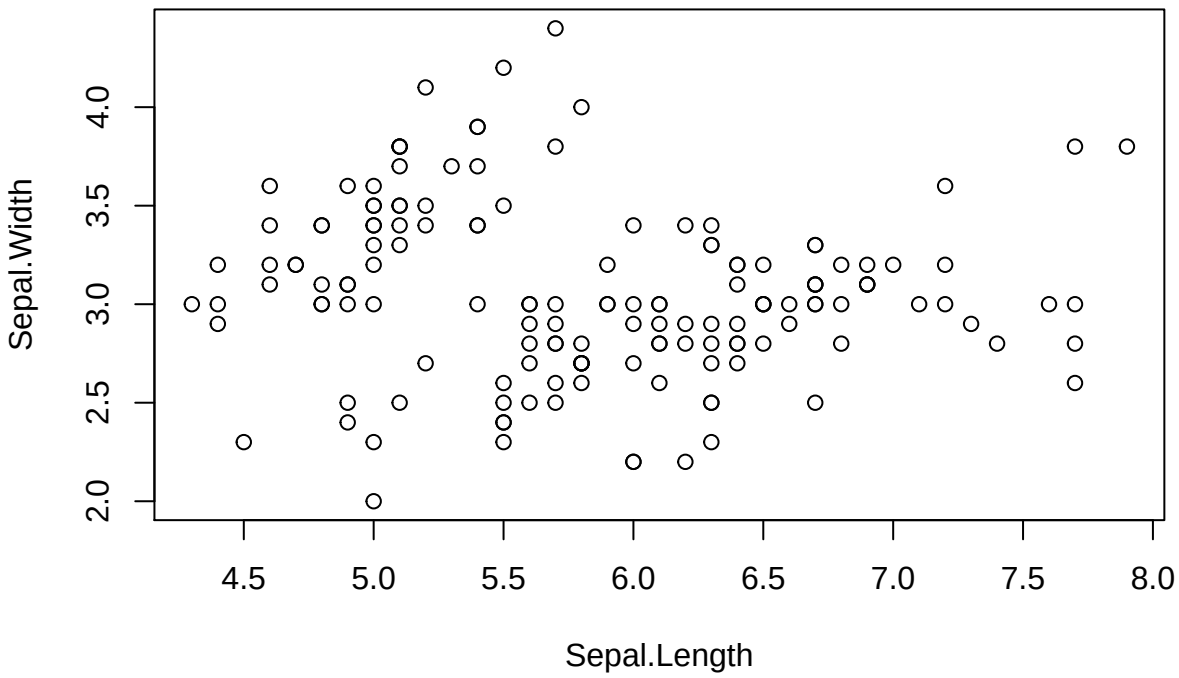

```
# qplot
qplot(Sepal.Length, Sepal.Width, data = iris)
```



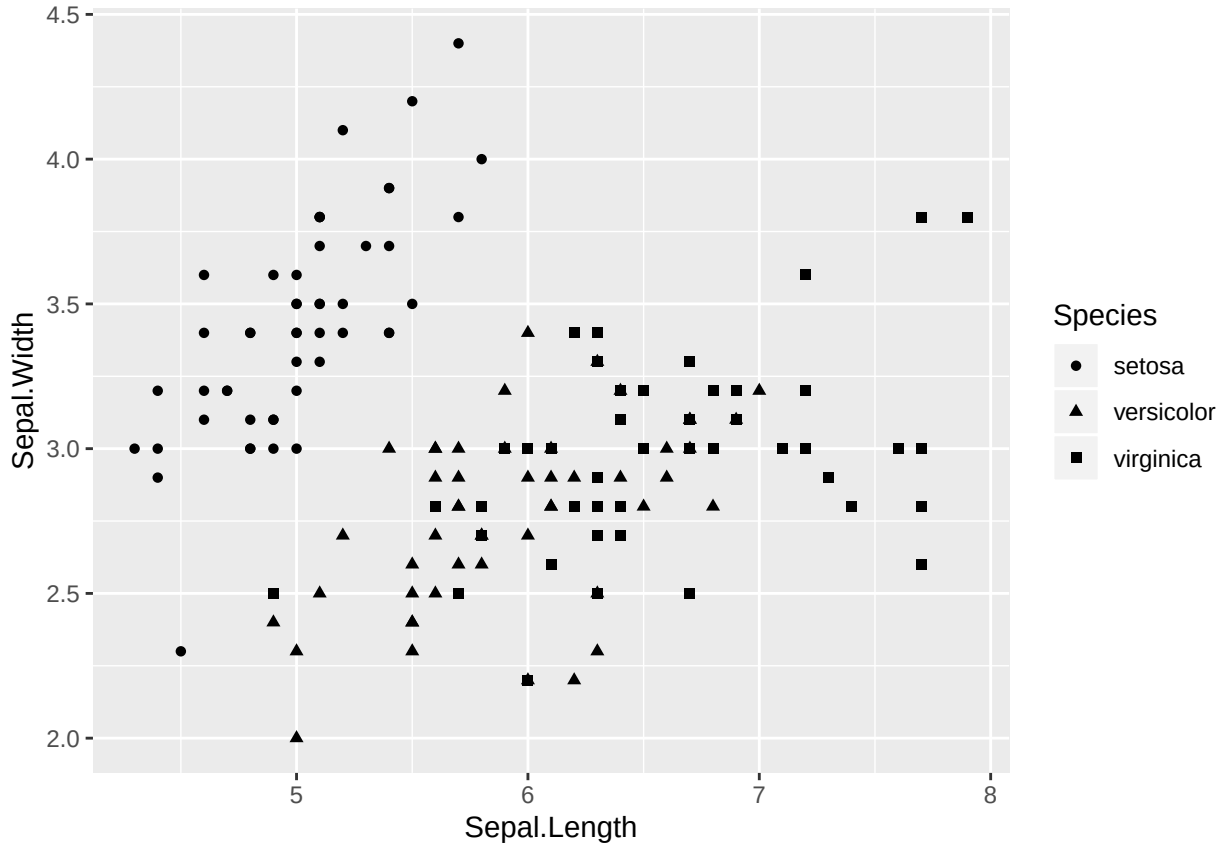

```
# base
plot(iris$Sepal.Length, iris$Sepal.Width)
```




```
plot(Sepal.Width ~ Sepal.Length, data = iris)
```

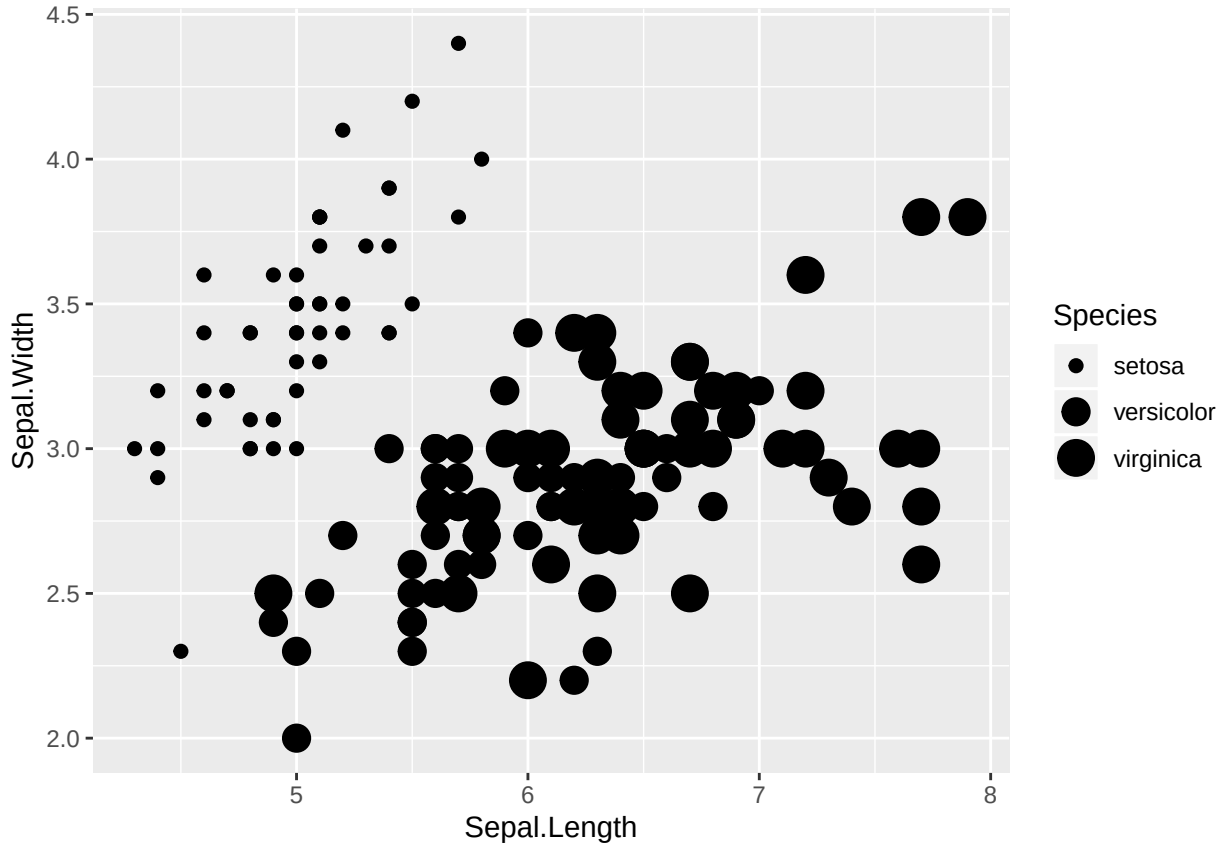



```
# shape = Species
qplot(Sepal.Length, Sepal.Width, data = iris, shape = Species)
```

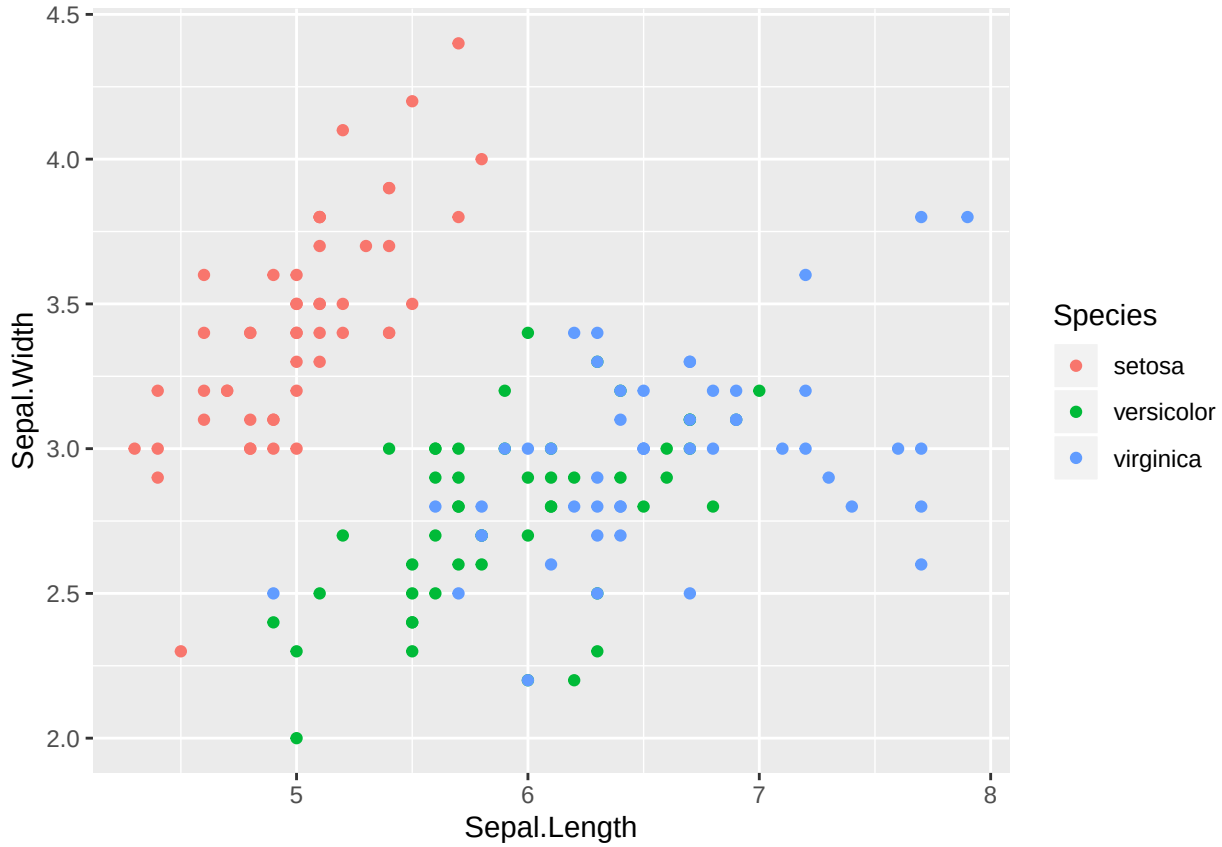



```
# size = Species
qplot(Sepal.Length, Sepal.Width, data = iris, size = Species)
```

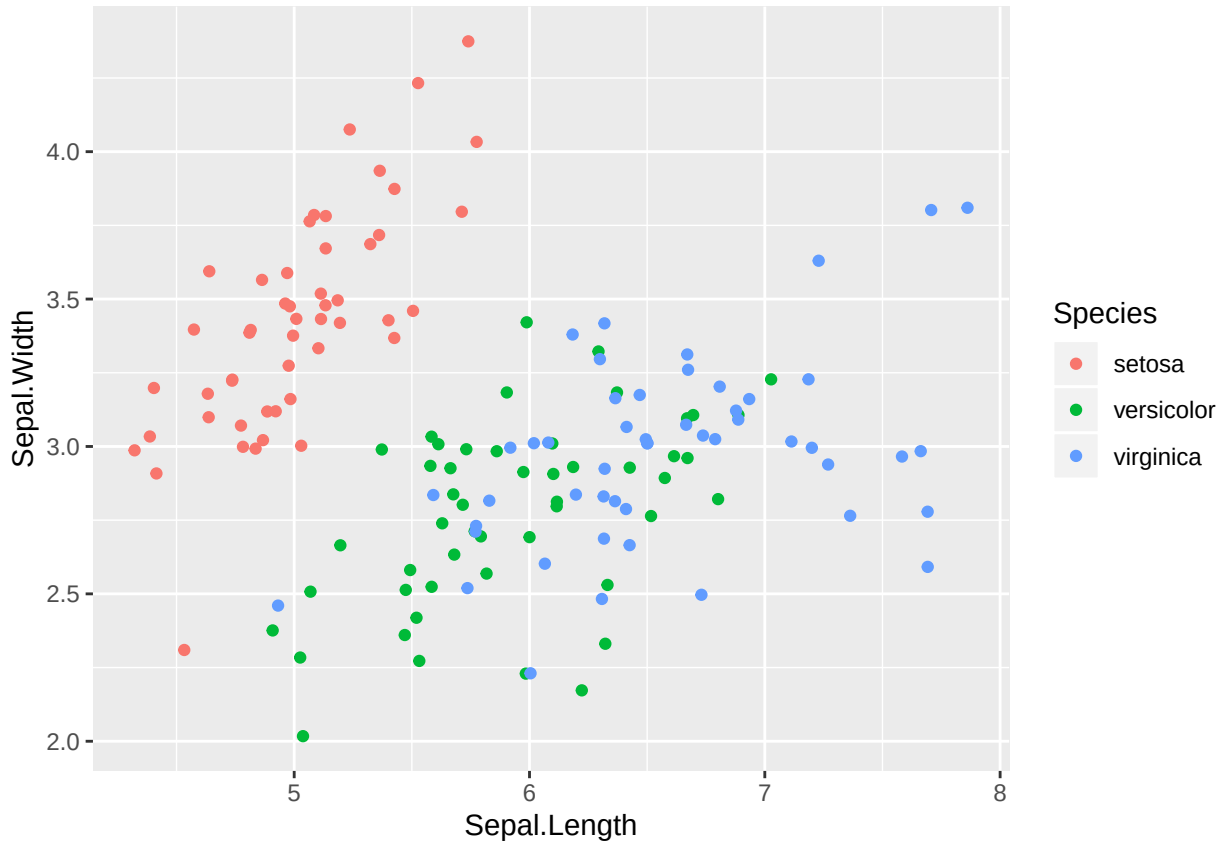
```
## Warning: Using size for a discrete variable is not advised.
```




```
# col = Species
qplot(Sepal.Length, Sepal.Width, data = iris, col = Species)
```

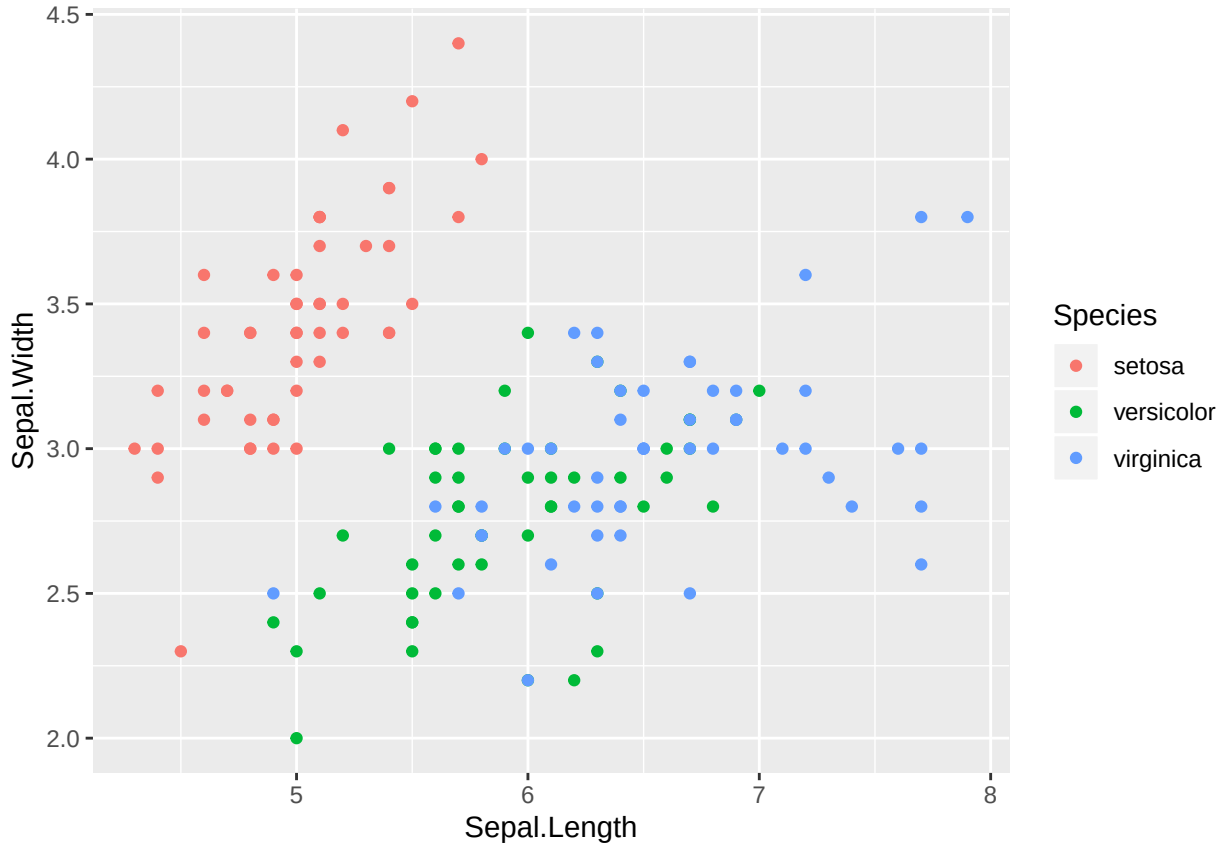



```
# geom argument
qplot(Sepal.Length, Sepal.Width, data = iris, col = Species,
  geom = "jitter")
```



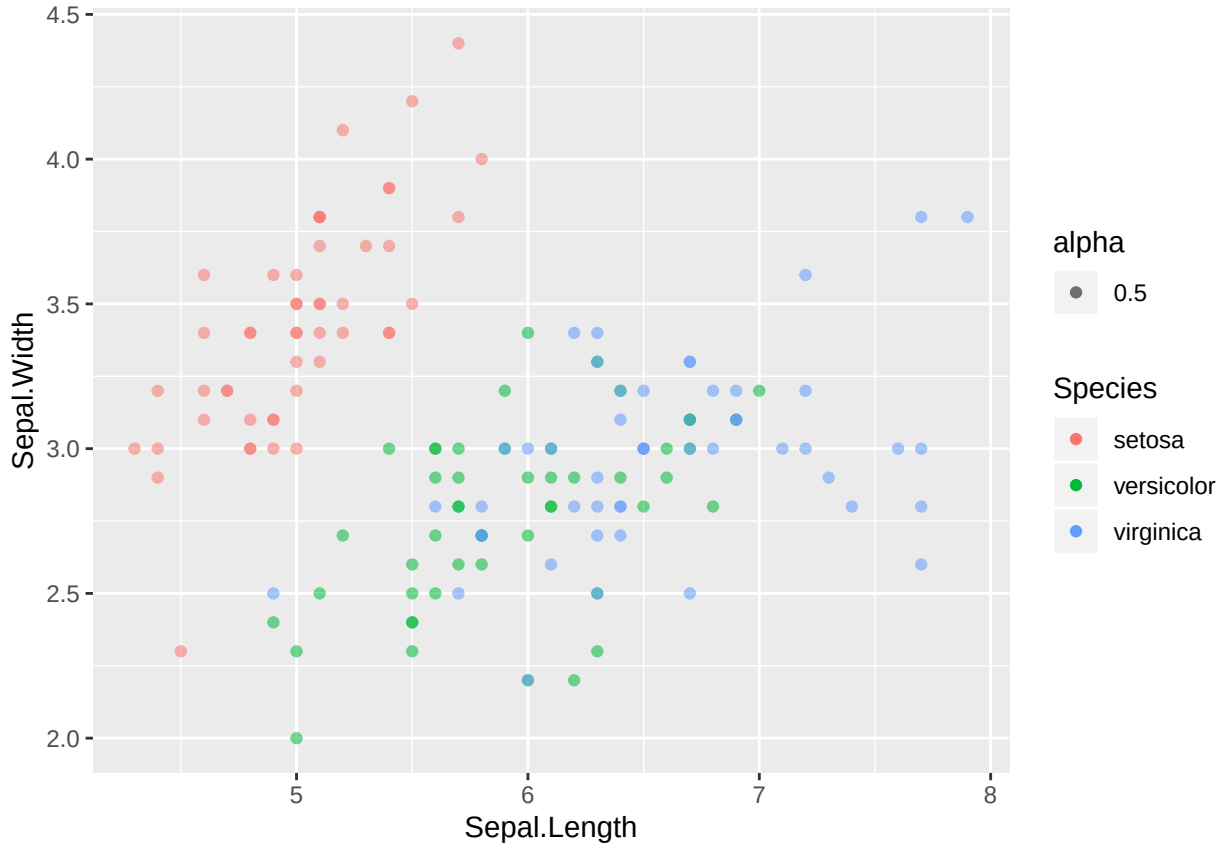

```
# position argument
qplot(Sepal.Length, Sepal.Width, data = iris, col = Species,
  position = "jitter")
```

```
## Warning: `position` is deprecated
```



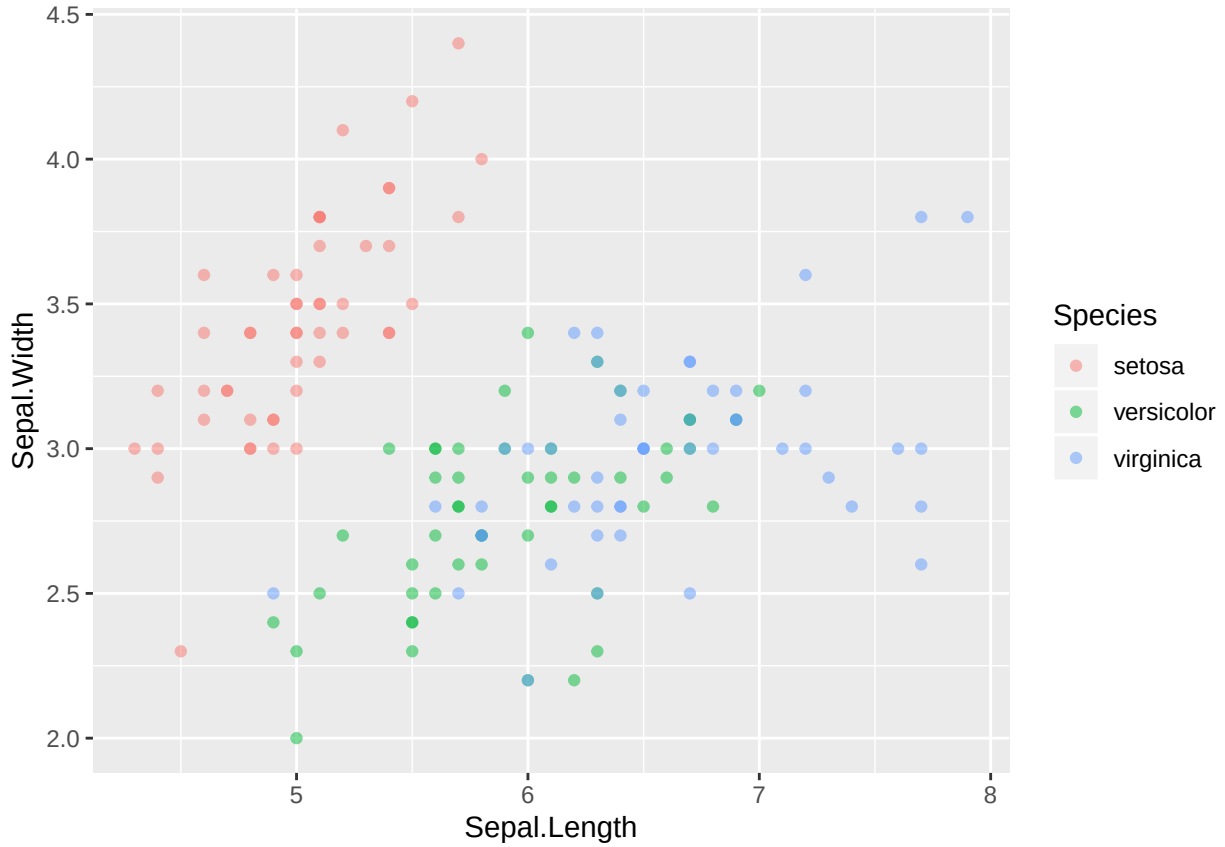

```
# alpha
qplot(Sepal.Length, Sepal.Width, data = iris, col = Species,
  position = "jitter", alpha = 0.5)
```

```
## Warning: `position` is deprecated
```




```
qplot(Sepal.Length, Sepal.Width, data = iris, col = Species,
  position = "jitter", alpha = I(0.5))
```

```
## Warning: `position` is deprecated
```



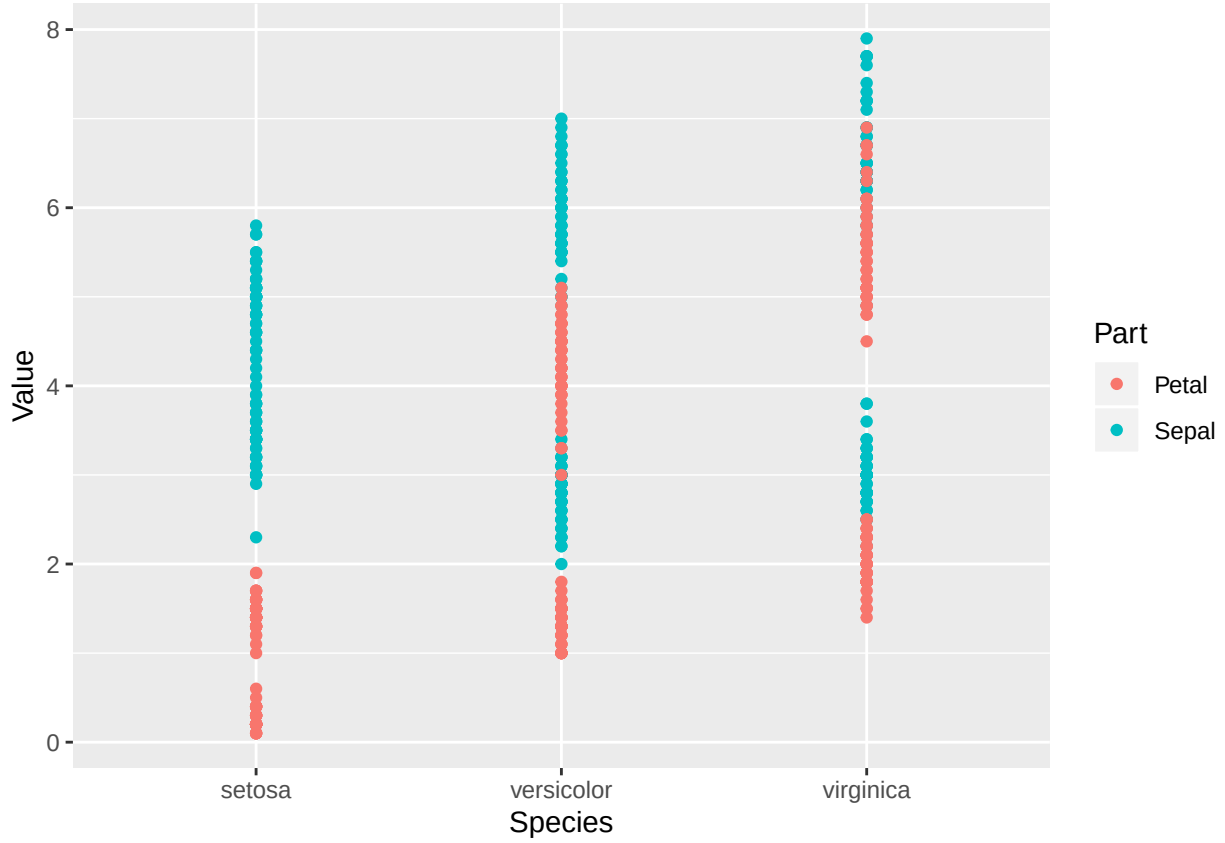

```
library(tidyr)
library(tidyverse)
```

```
## -- Attaching packages -------- tidyverse 1.2.1 --

## v tibble  2.1.3     v dplyr   0.8.3
## v readr   1.3.1     v stringr 1.4.0
## v purrr   0.3.2     v forcats 0.4.0

## -- Conflicts ----------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
iris.tidy <- iris %>%
  gather(key, Value, -Species) %>%
  separate(key, c("Part", "Measure"), "\\.")
str(iris)
```

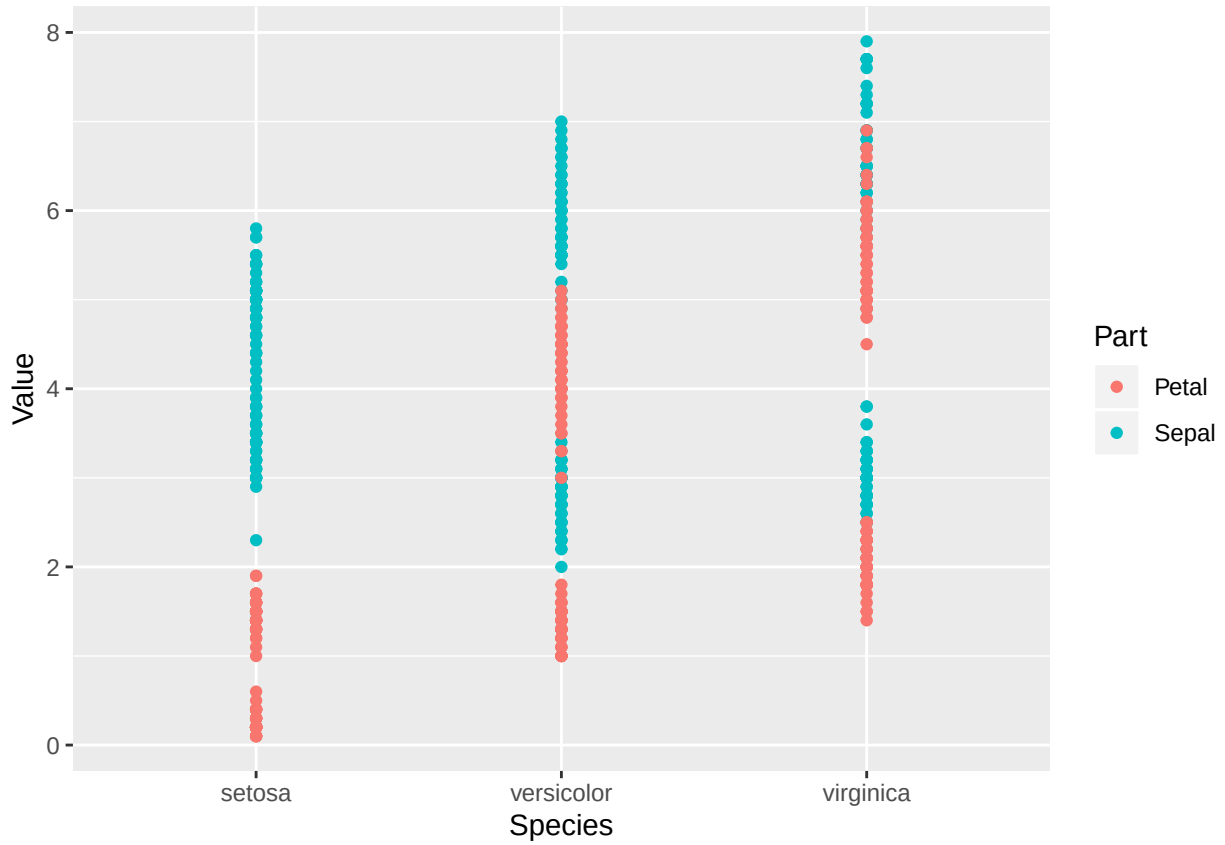
```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
# continuous vs categorical
qplot(Species, Value, data = iris.tidy, col = Part)
```



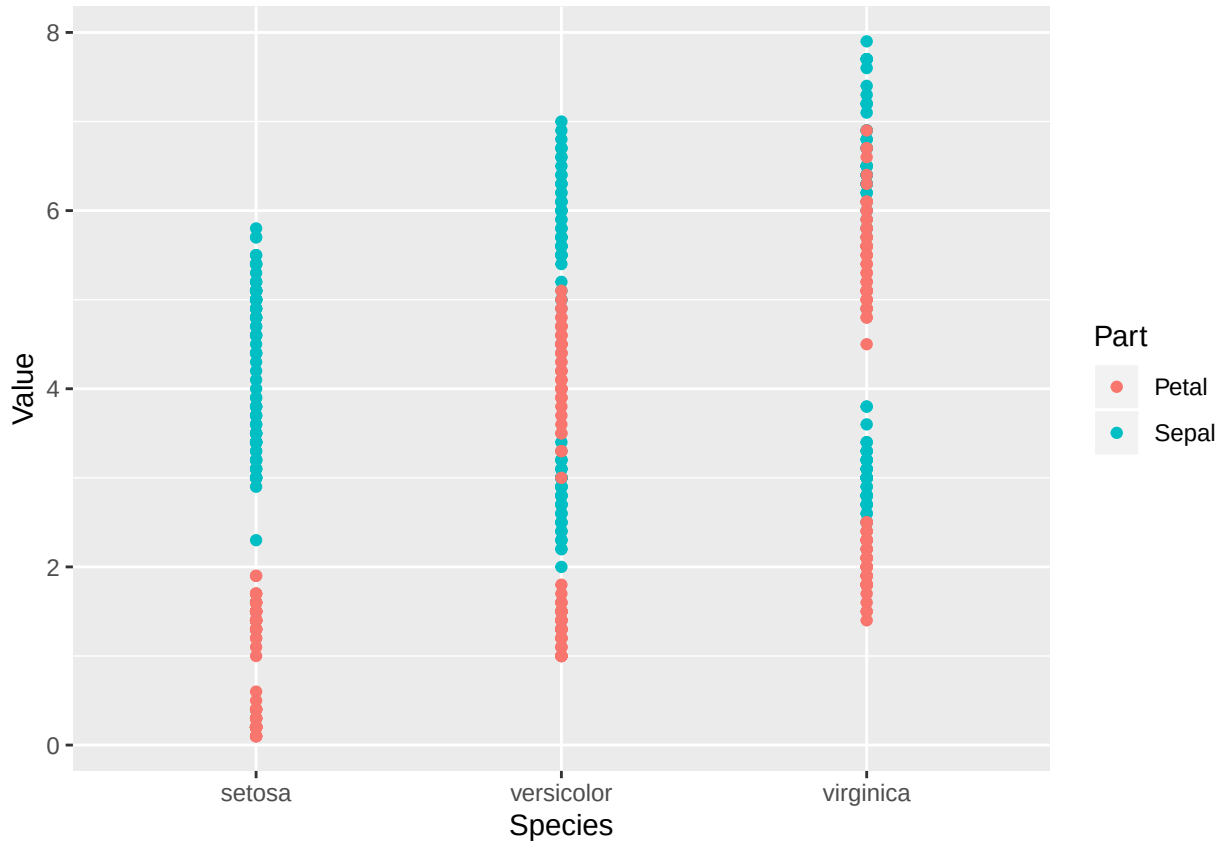

```
# position = "jitter"
qplot(Species, Value, data = iris.tidy, col = Part,
  position = "jitter")
```

```
## Warning: `position` is deprecated
```



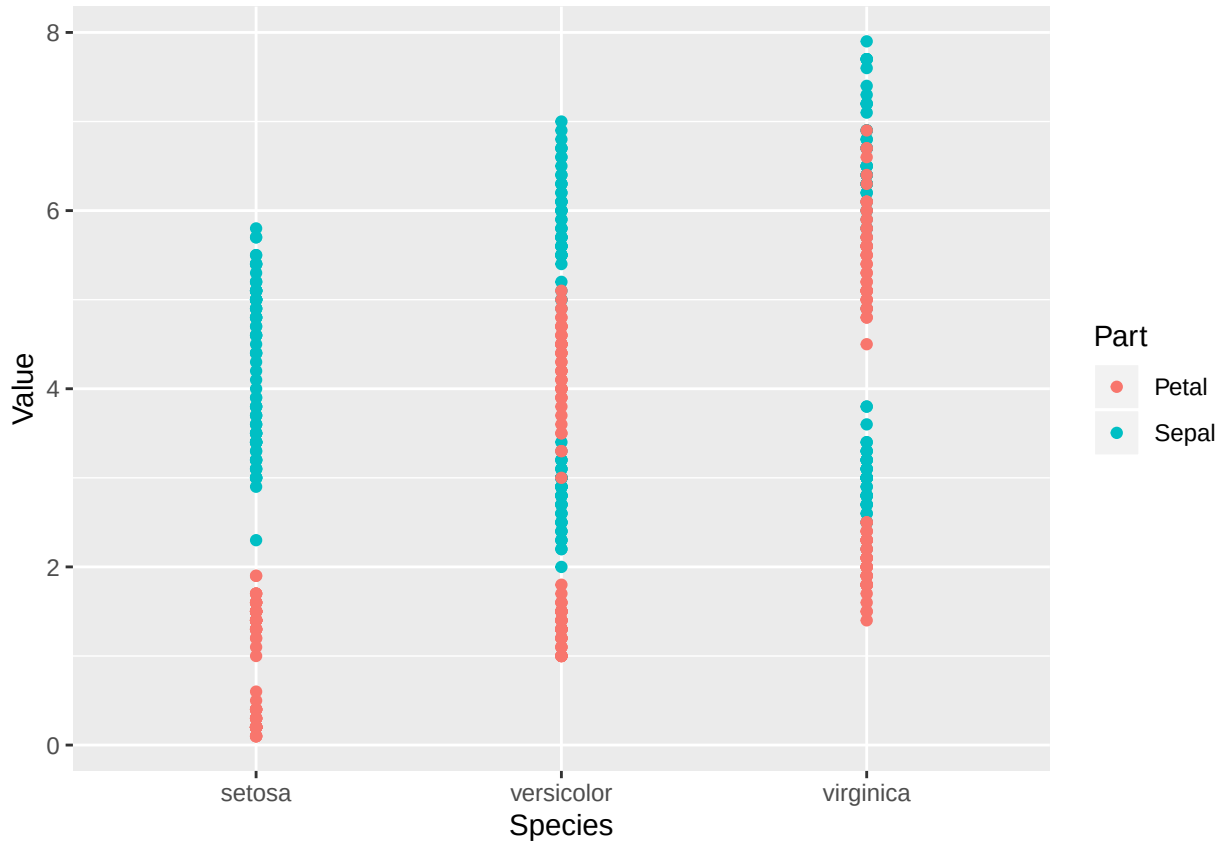

```
# jitter manually
posn.j <- position_jitter(0.1)
qplot(Species, Value, data = iris.tidy, col = Part,
  position = posn.j)
```

```
## Warning: `position` is deprecated
```

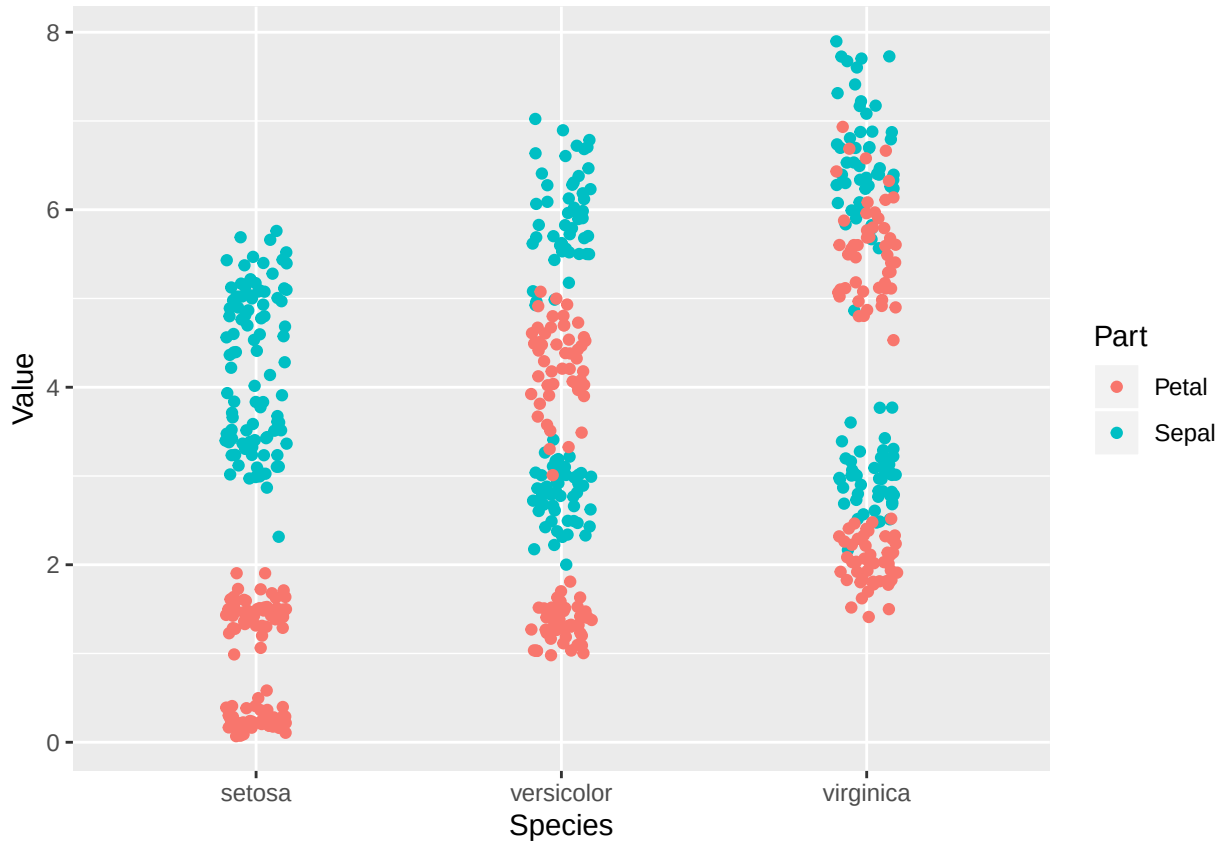



```
# comparison
posn.j <- position_jitter(0.1)
qplot(Species, Value, data = iris.tidy, col = Part,
  position = posn.j) # Fine for easy plot
```

```
## Warning: `position` is deprecated
```

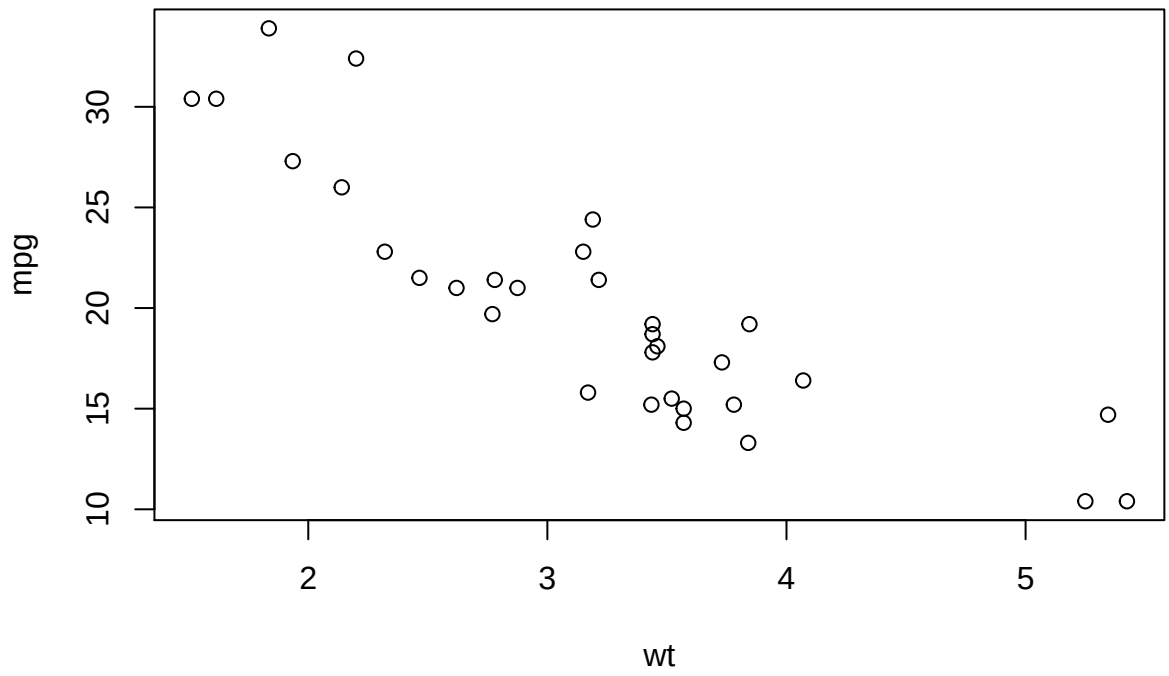



```
ggplot(iris.tidy, aes(x = Species, y = Value, col = Part)) +
  geom_point(position = posn.j) # very flexible
```

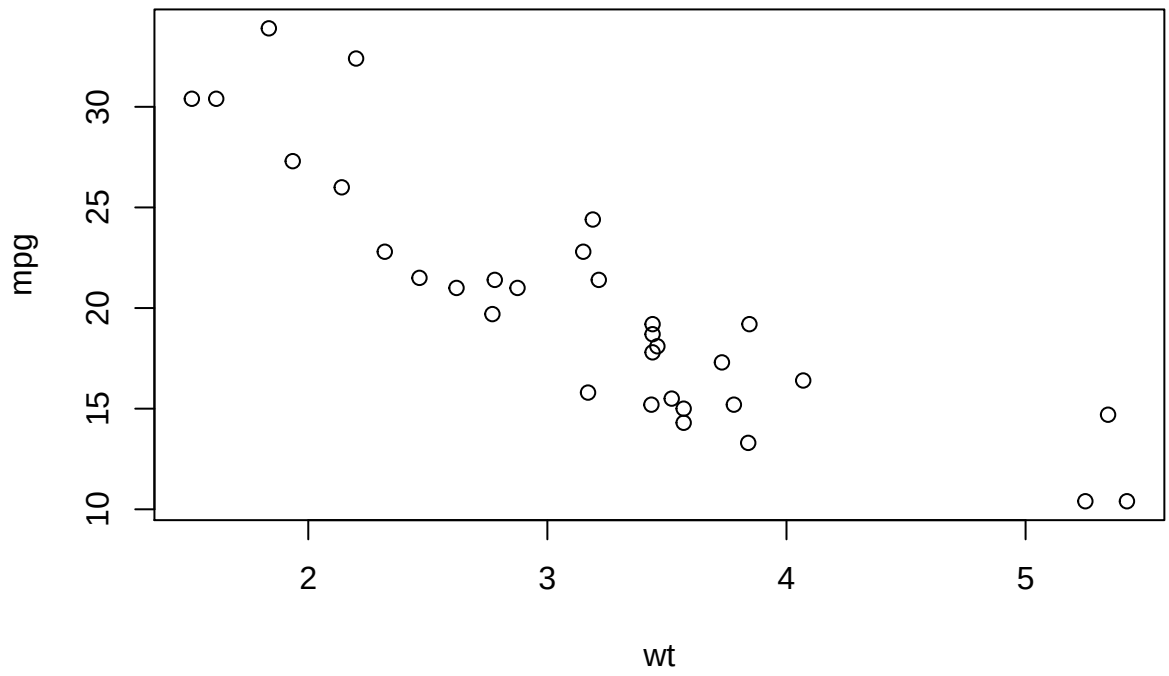


Practice

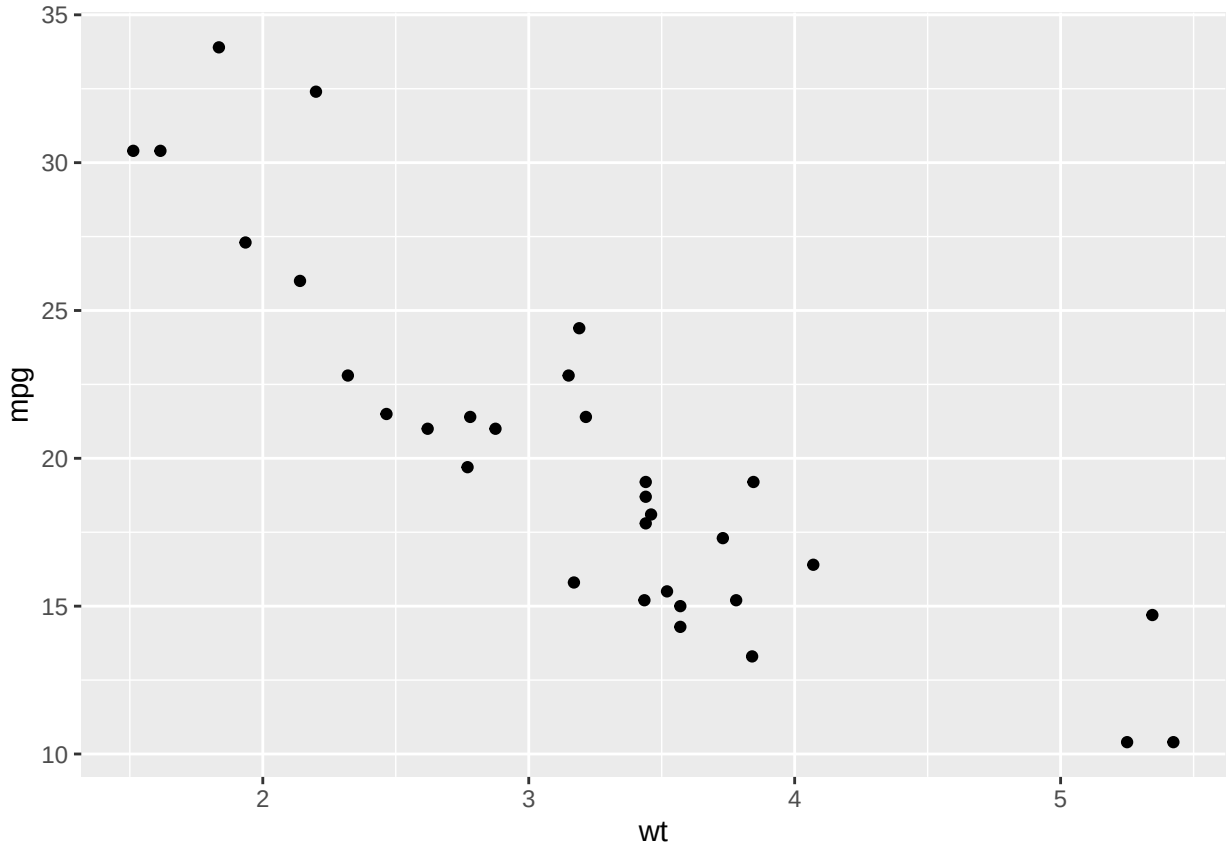
```
# The old way (shown)
plot(mpg ~ wt, data = mtcars) # formula notation
```



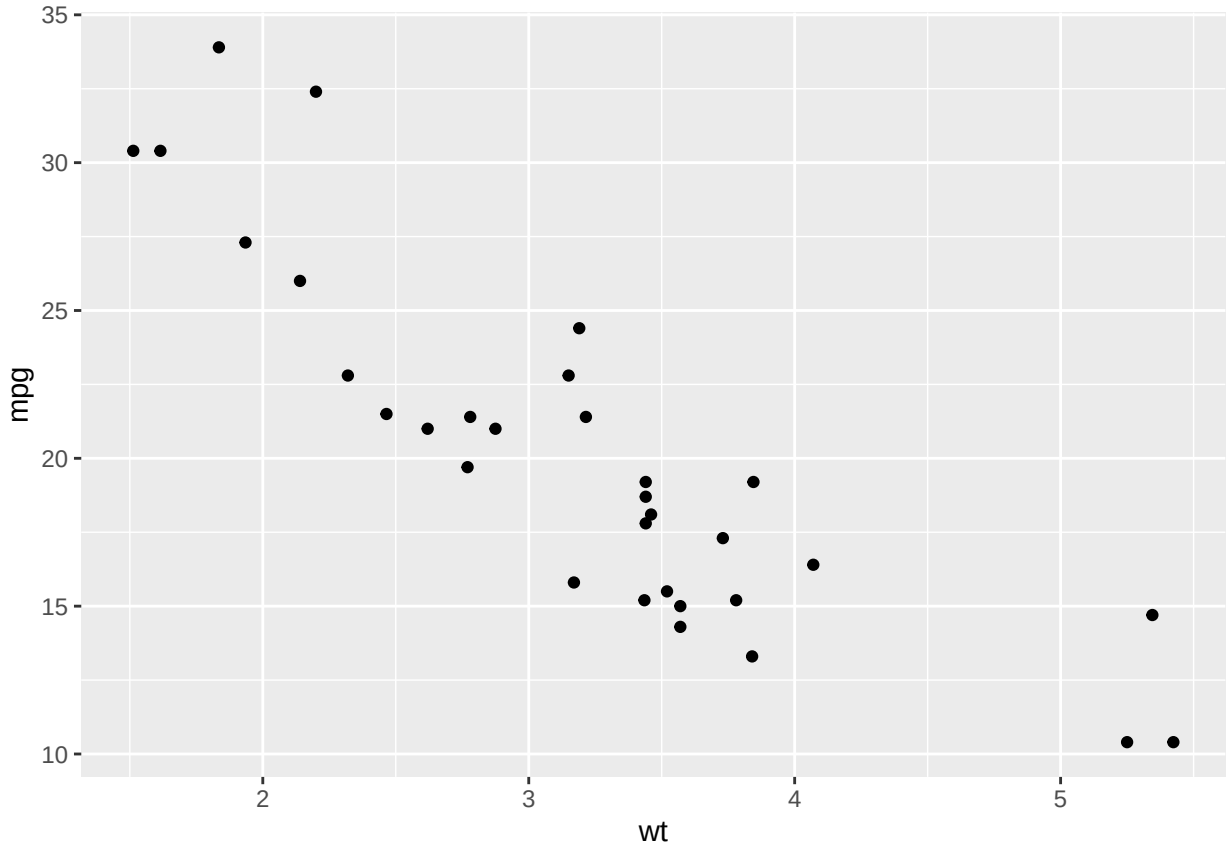

```
with(mtcars, plot(wt, mpg)) # x, y notation
```



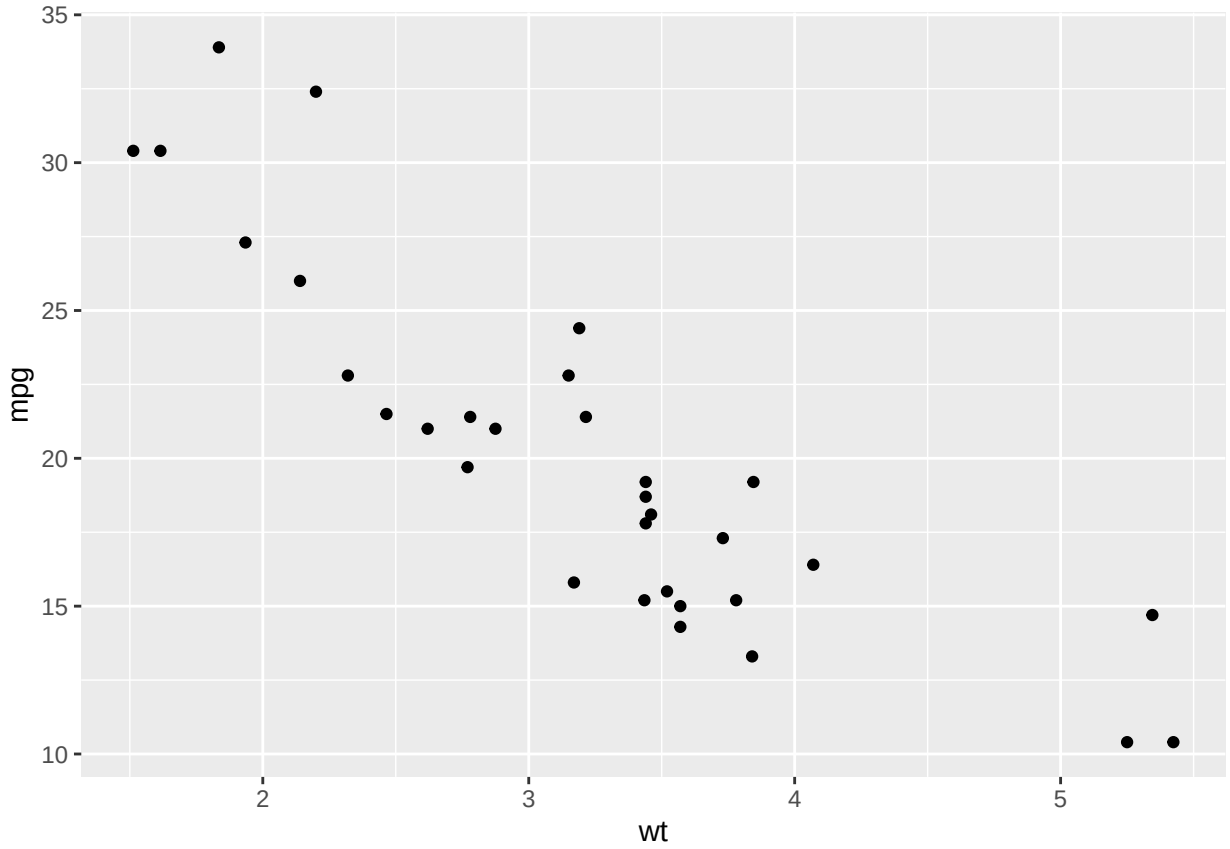

```
# Using ggplot:
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point()
```




```
# Using qplot:
qplot(wt, mpg, data = mtcars)
```

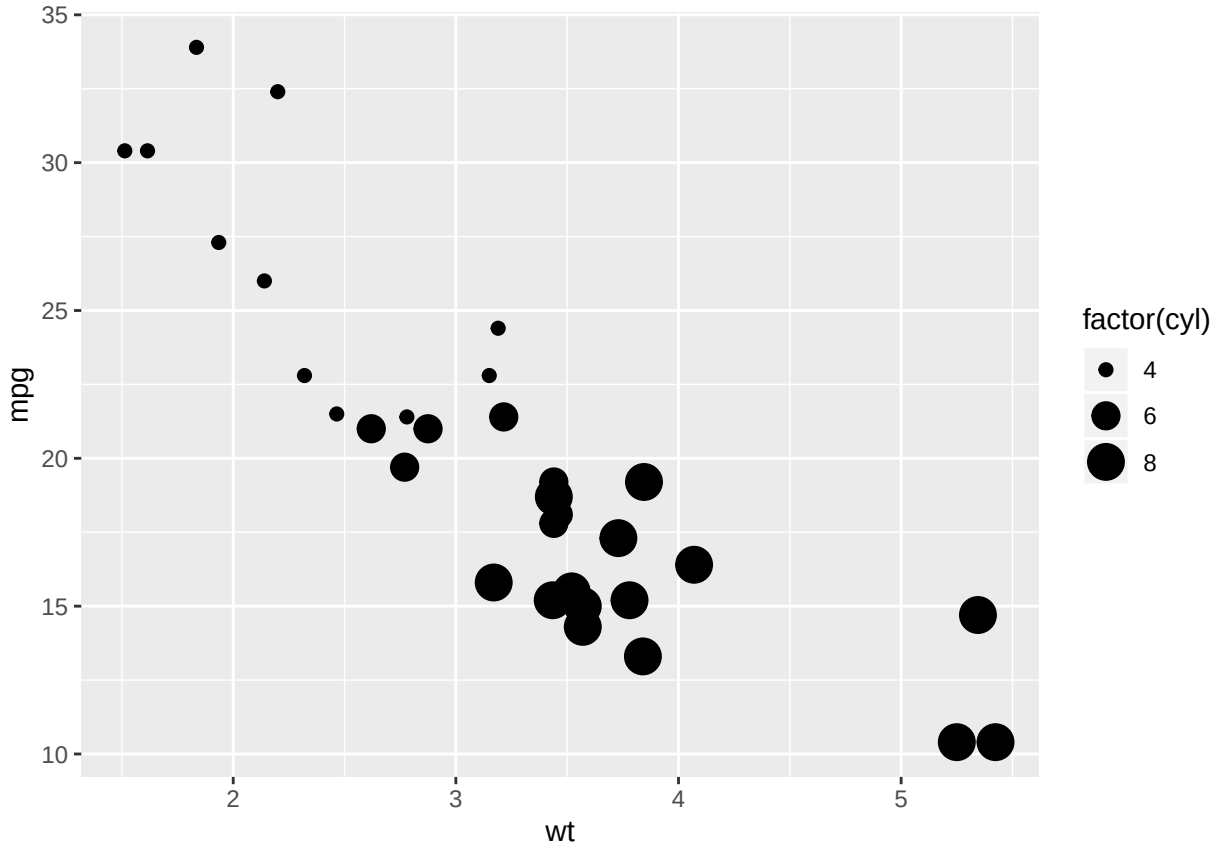



```
# basic qplot scatter plot:
qplot(wt, mpg, data = mtcars)
```



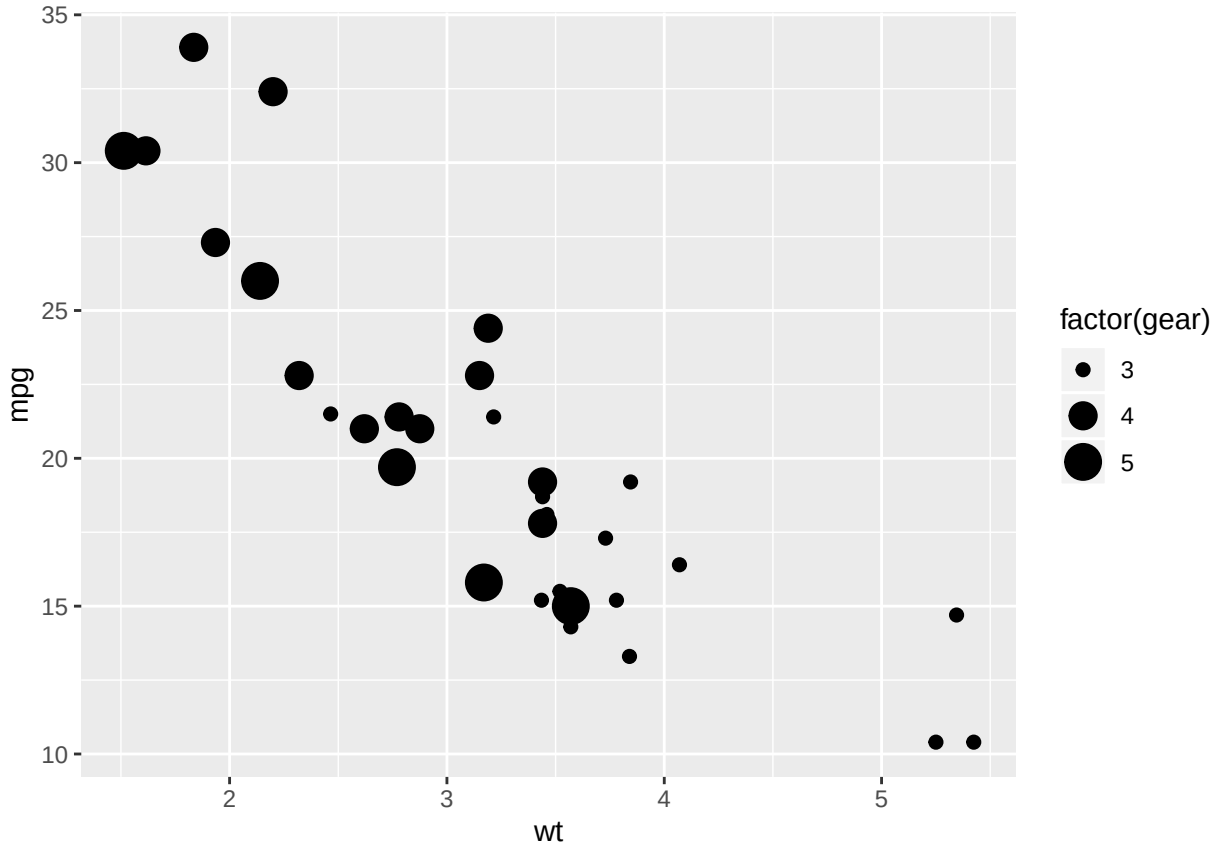

```
# Categorical variable mapped onto size:
# cyl
qplot(wt, mpg, data = mtcars, size = factor(cyl))
```

```
## Warning: Using size for a discrete variable is not advised.
```

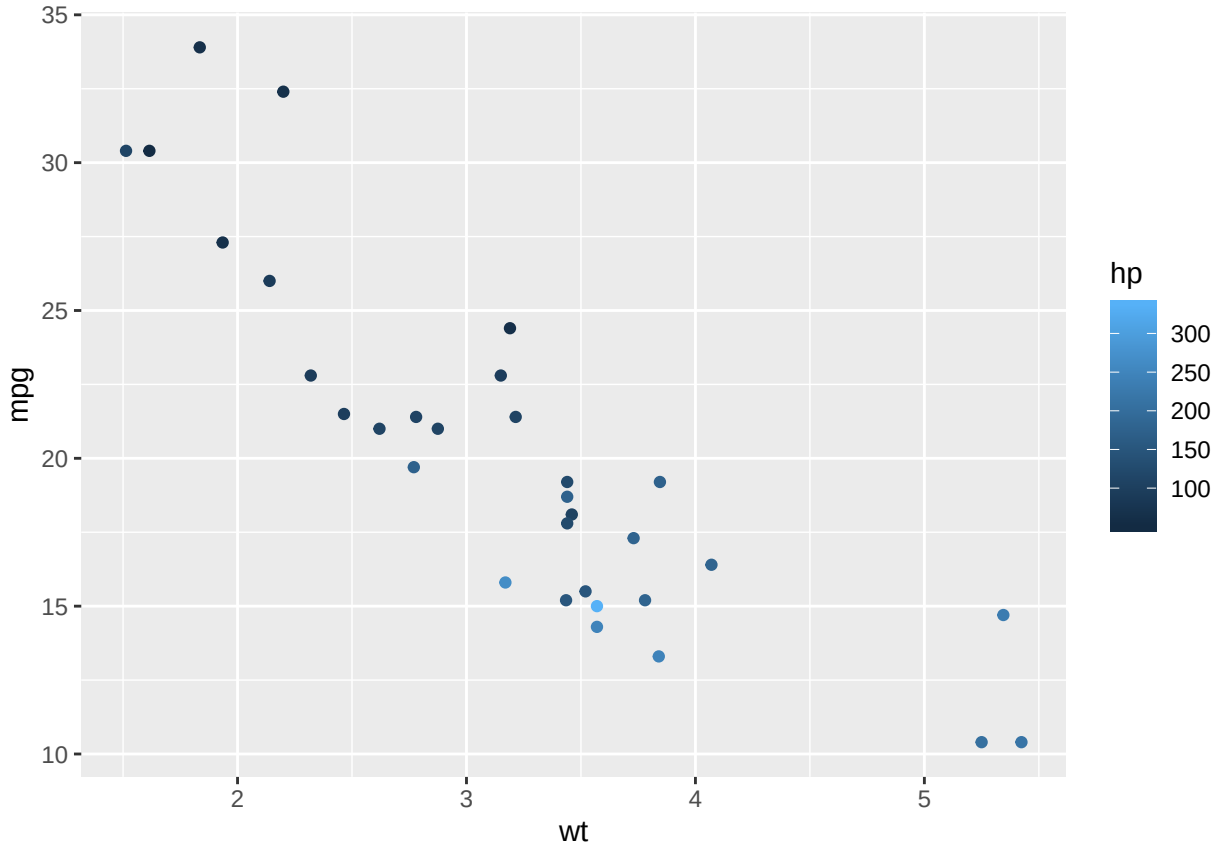



```
# gear
qplot(wt, mpg, data = mtcars, size = factor(gear))
```

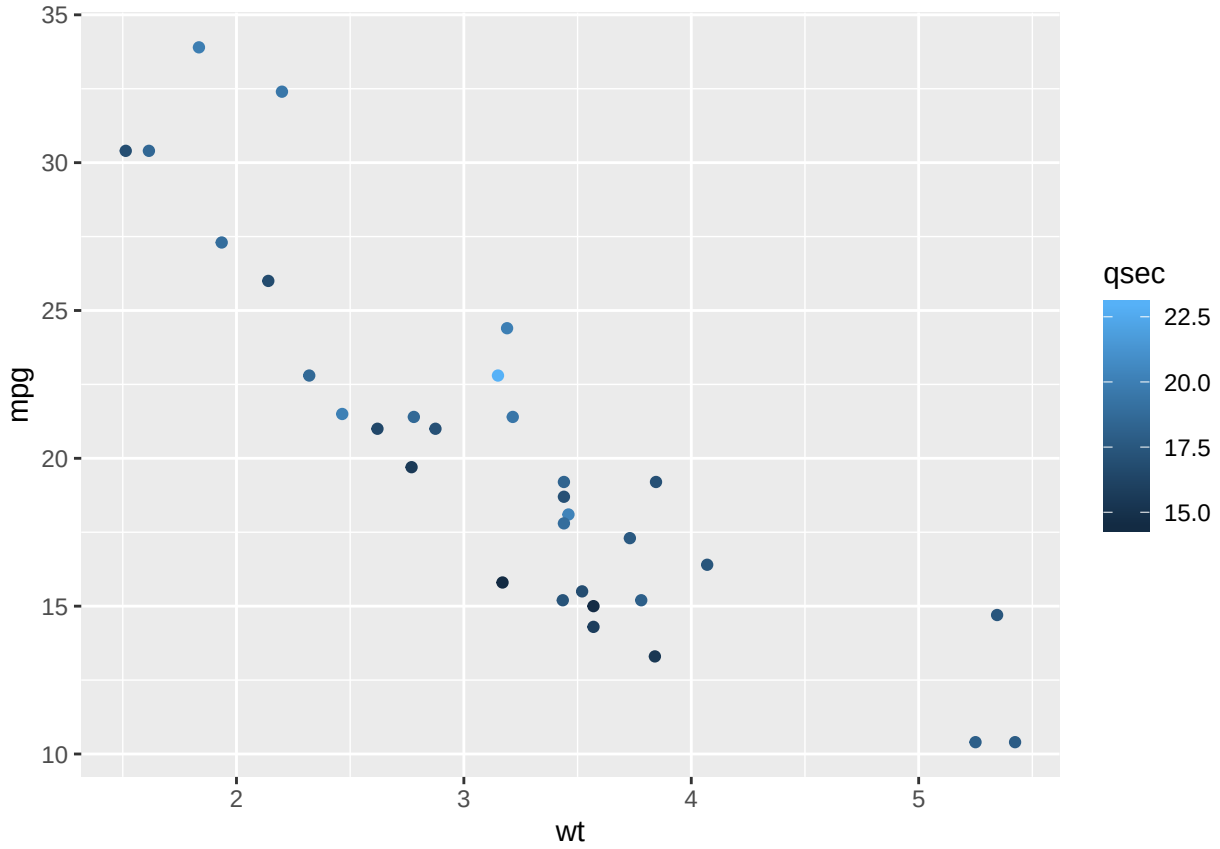
```
## Warning: Using size for a discrete variable is not advised.
```



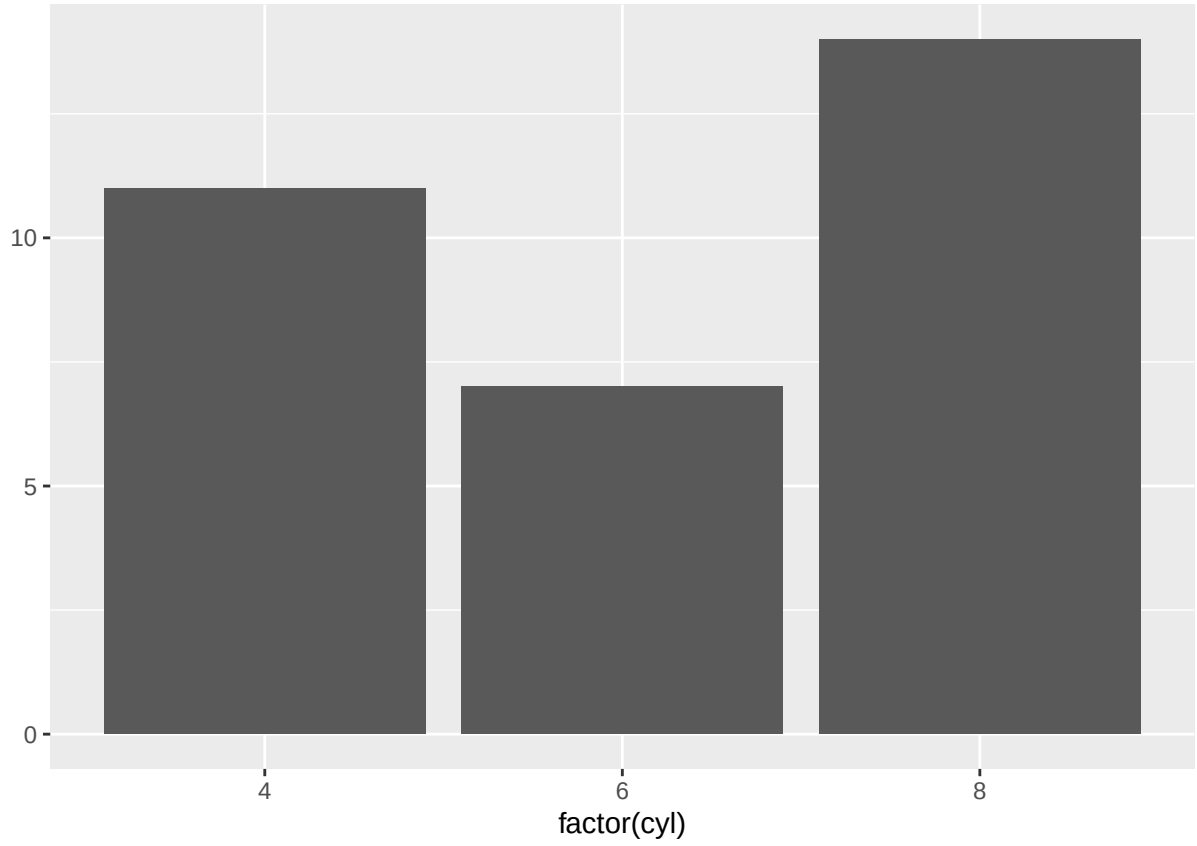

```
# Continuous variable mapped onto col:
# hp
qplot(wt, mpg, data = mtcars, col = hp)
```



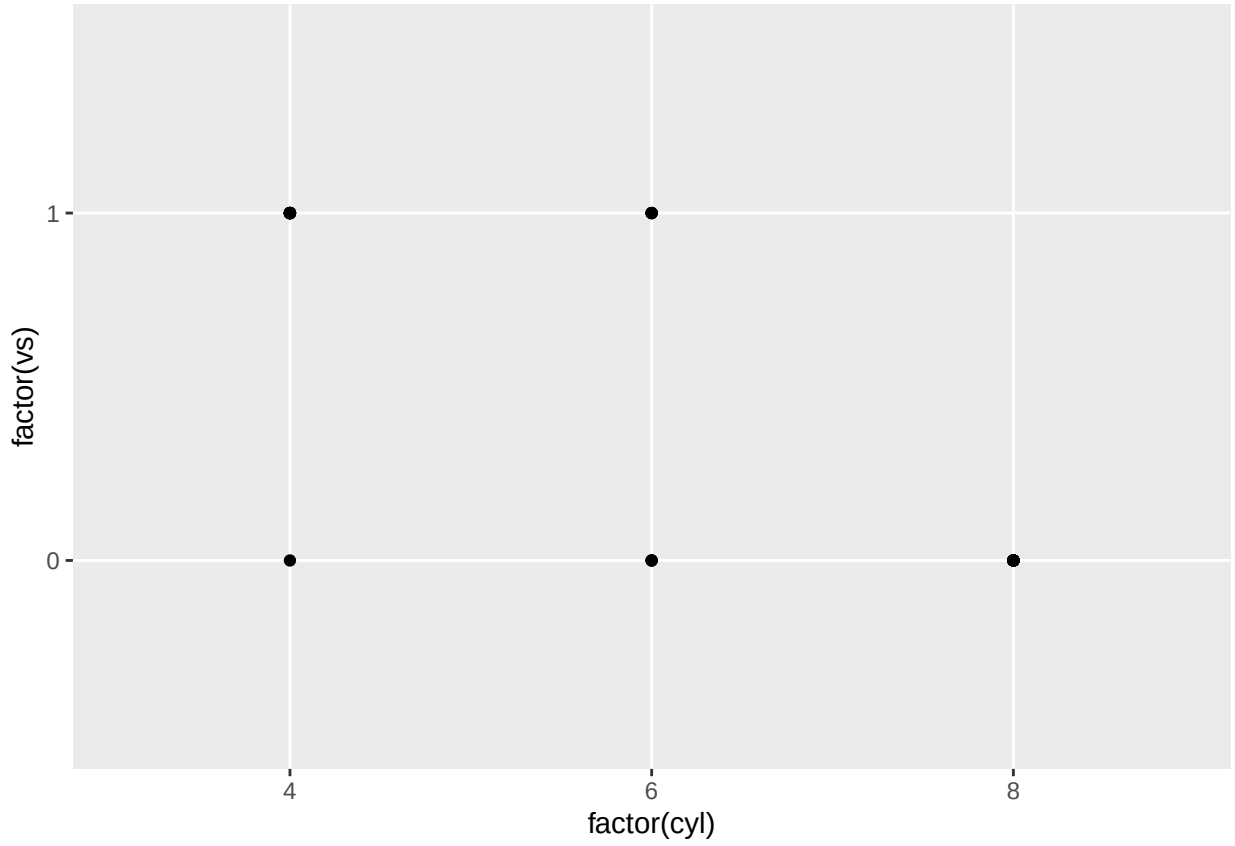

```
# qsec
qplot(wt, mpg, data = mtcars, col = qsec)
```



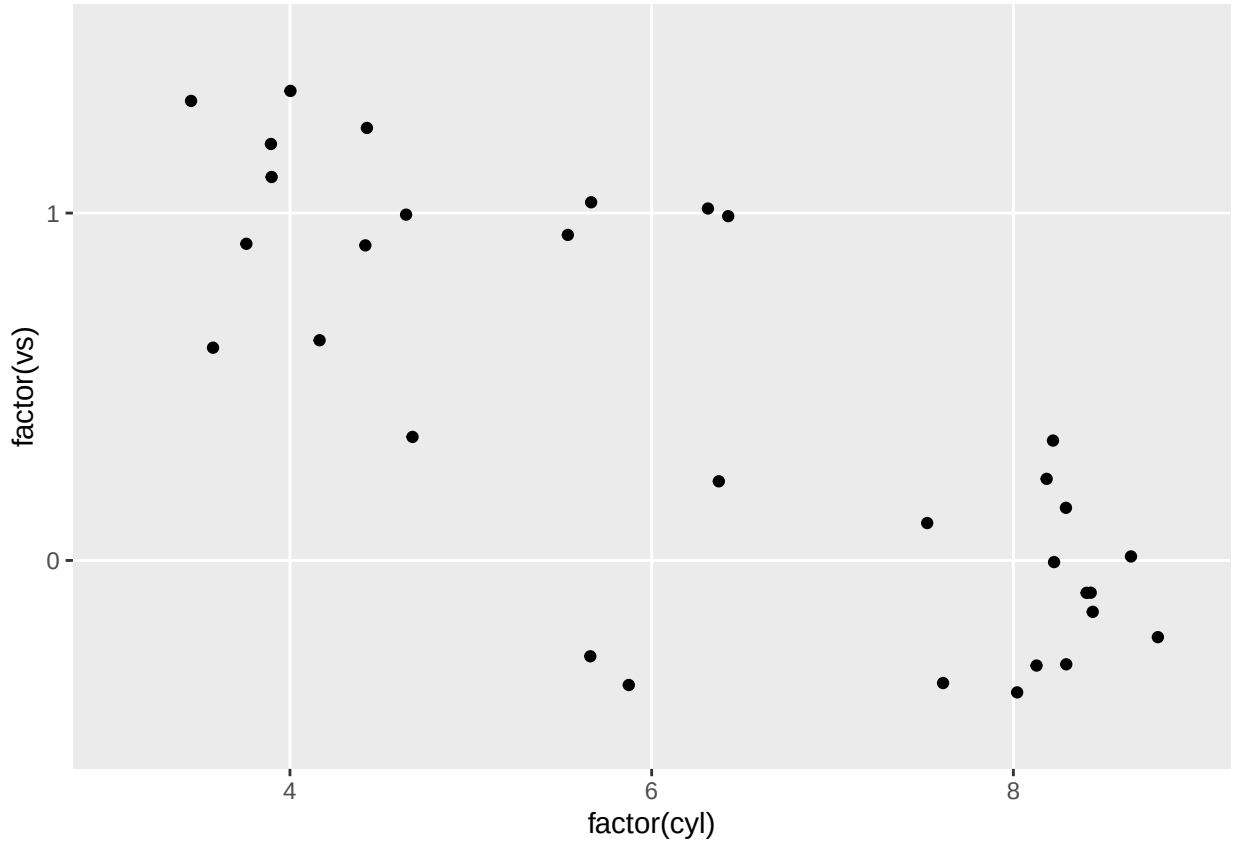

```
# qplot() with x only
qplot(x = factor(cyl), data = mtcars)
```




```r
# qplot() with x and y
qplot(x = factor(cyl), y = factor(vs),data = mtcars)
```




```
# qplot() with geom set to jitter manually
qplot(x = factor(cyl), y = factor(vs),data = mtcars,geom ="jitter")
```



Some naming conventions:

- Scatter plots:

Continuous x, continuous y.

- Dot plots:

Categorical x, continuous y.

You use geom_point() for both plot types. Jittering position is set in the geom_point() layer.

However, to make a "true" dot plot, you can use geom_dotplot(). The difference is that unlike geom_point(), geom_dotplot() uses a binning statistic. Binning means to cut up a continuous variable (the y in this case) into discrete "bins". You already saw binning with geom_histogram() (see this exercise for a refresher).

One thing to notice is that geom_dotplot() uses a different plotting symbol to geom_point(). For these symbols, the color aesthetic changes the color of its border, and the fill aesthetic changes the color of its interior.

```
# cyl and am are factors, wt is numeric
class(mtcars$cyl)
```

```
## [1] "numeric"
```

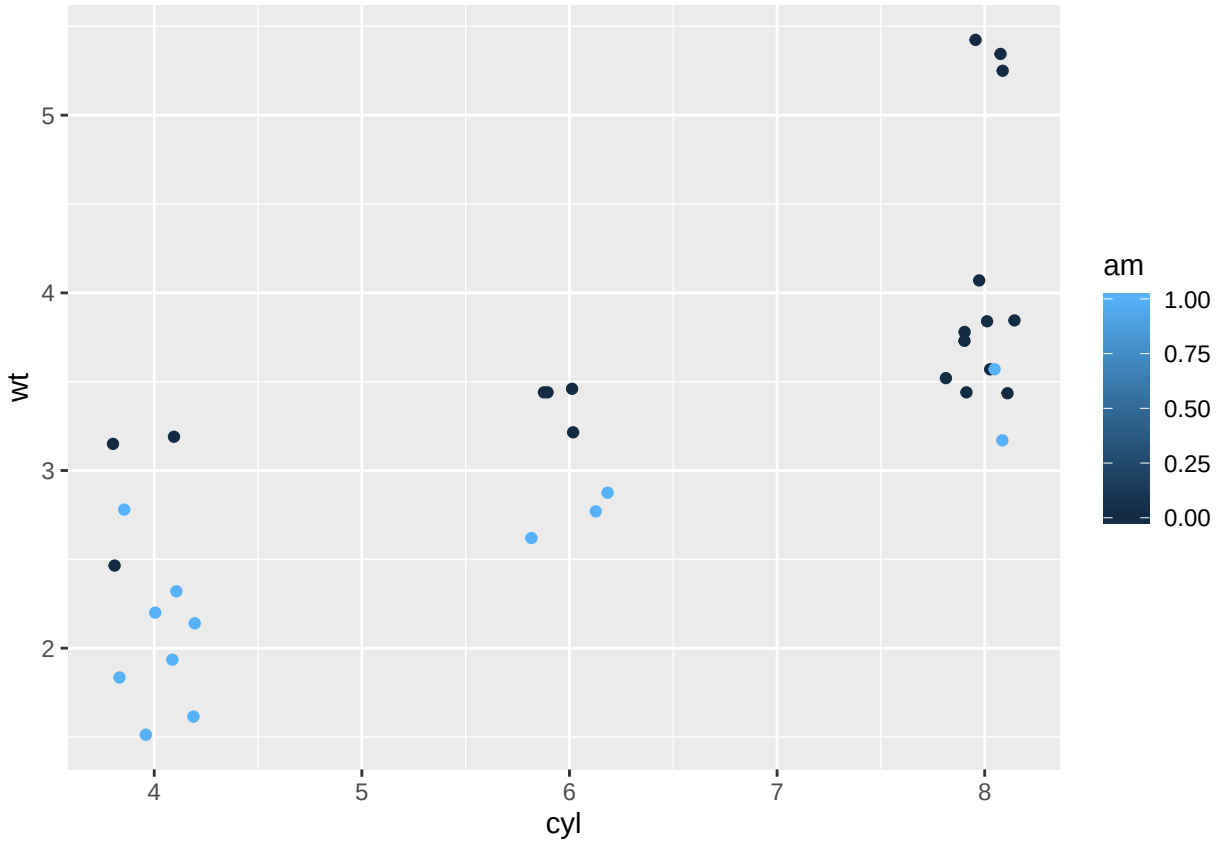
```r
class(mtcars$am)
```

```
## [1] "numeric"
```

```r
class(mtcars$wt)
```

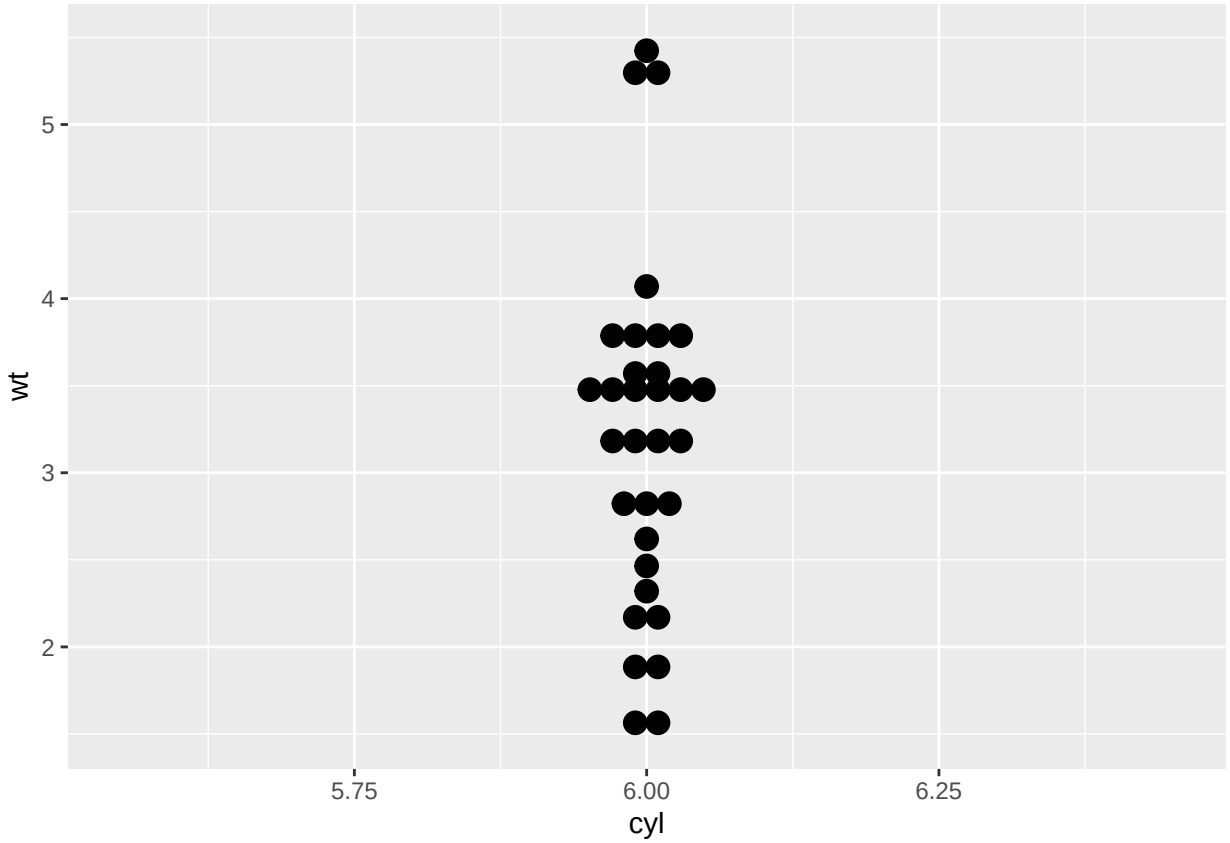
```
## [1] "numeric"
```

```r
# "Basic" dot plot, with geom_point():
ggplot(mtcars, aes(cyl, wt, col = am)) +
  geom_point(position = position_jitter(0.2, 0))
```



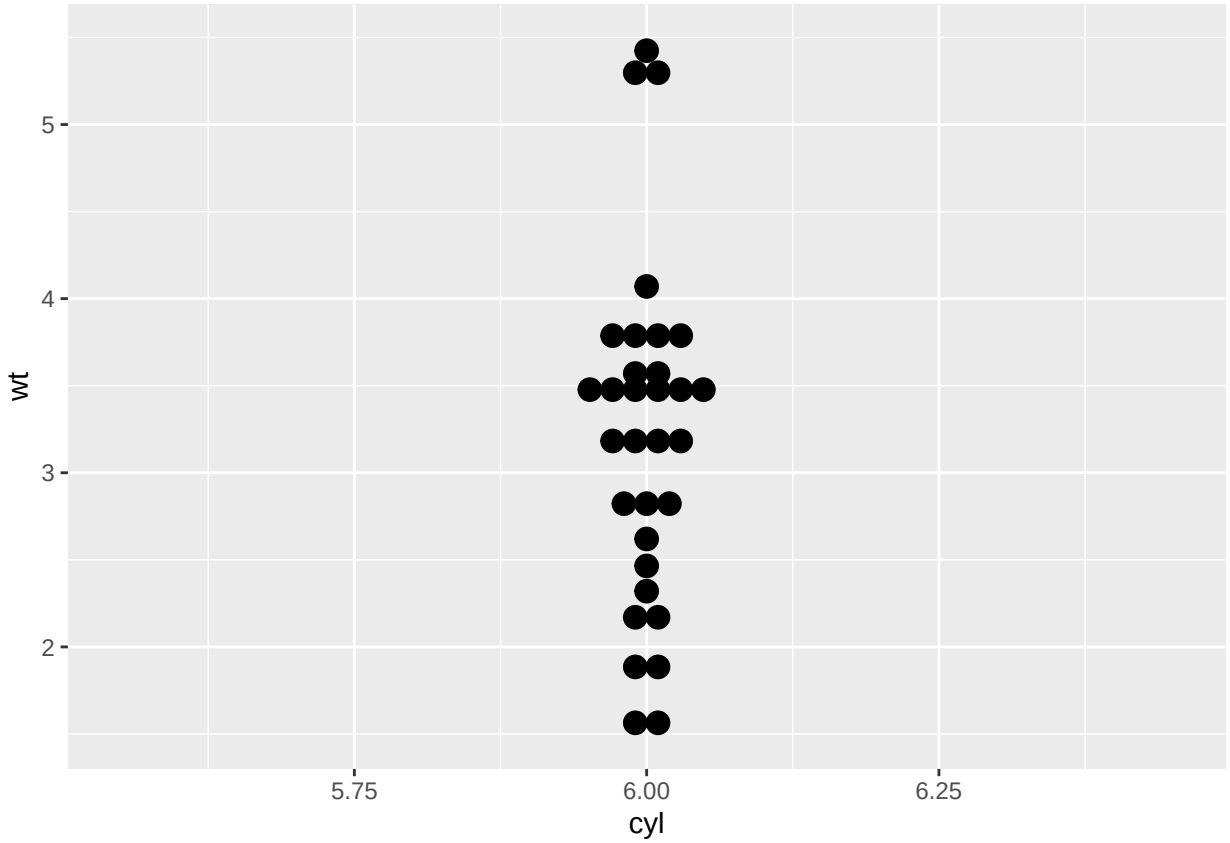

```r
# 1 - "True" dot plot, with geom_dotplot():
ggplot(mtcars, aes(cyl, wt, fill = am)) +
  geom_dotplot(binaxis = "y", stackdir = "center")
```

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




```
# 2 - qplot with geom "dotplot", binaxis = "y" and stackdir = "center"
qplot(
  x = cyl, y = wt,
  data = mtcars,
  fill = am,
  geom = "dotplot",
  binaxis = "y",
  stackdir = "center"
)
```

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



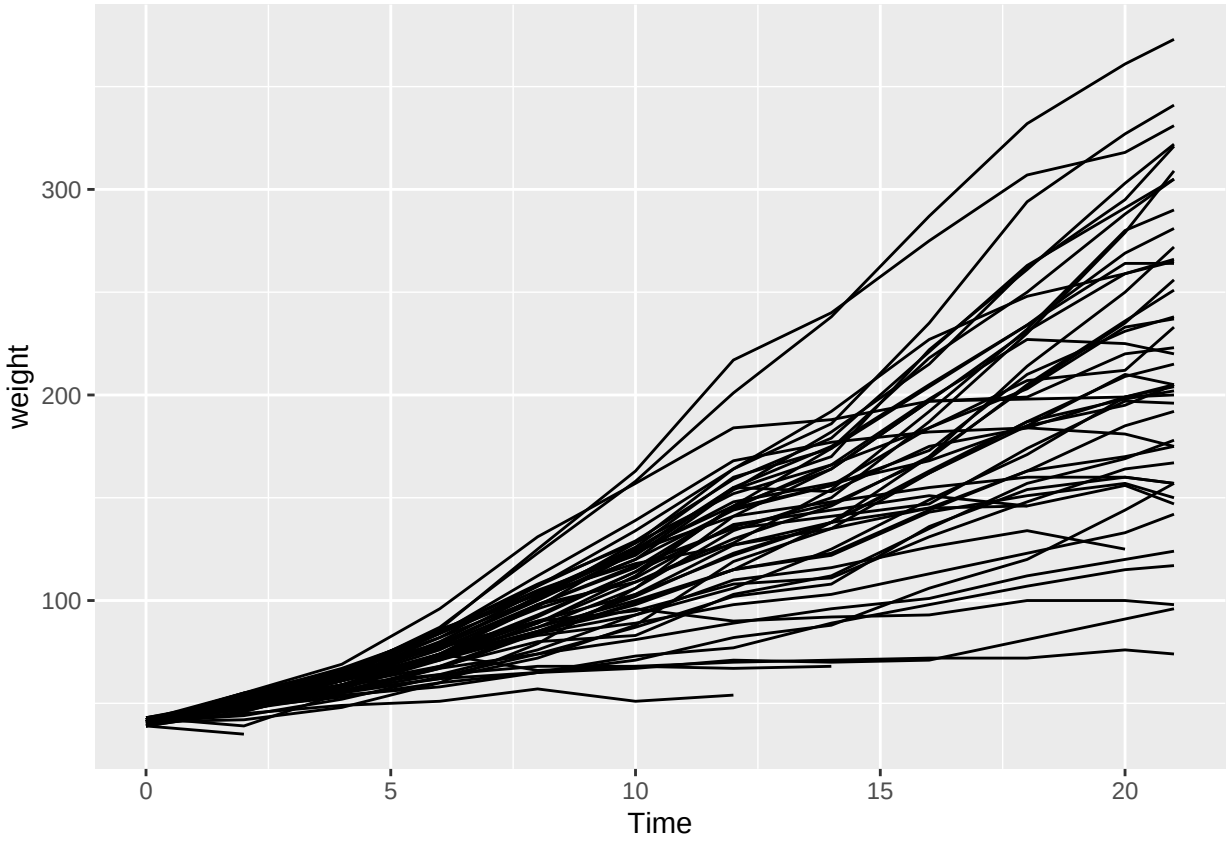
Chicken weight

The ChickWeight dataset is a data frame which represents the progression of weight of several chicks. The little chicklings are each given a specific diet. There are four types of diet and the farmer wants to know which one fattens the chicks the fastest.

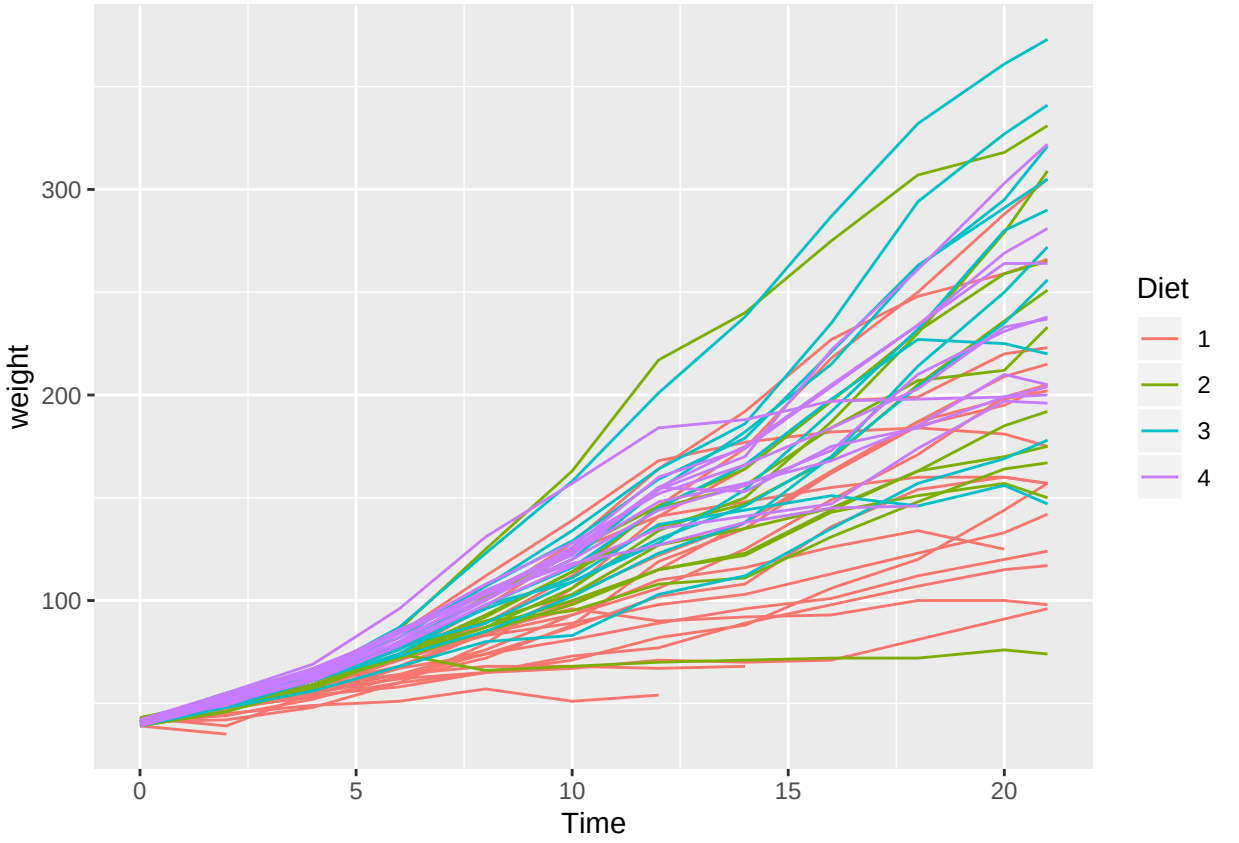
```
# ChickWeight is available in your workspace
# 1 - Check out the head of ChickWeight
head(ChickWeight)
```

```
## Grouped Data: weight ~ Time | Chick
##   weight Time Chick Diet
## 1     42    0     1    1
## 2     51    2     1    1
## 3     59    4     1    1
## 4     64    6     1    1
## 5     76    8     1    1
## 6     93   10     1    1
```

```
# 2 - Basic line plot
ggplot(ChickWeight, aes(x = Time, y = weight)) +
  geom_line(aes(group = Chick))
```

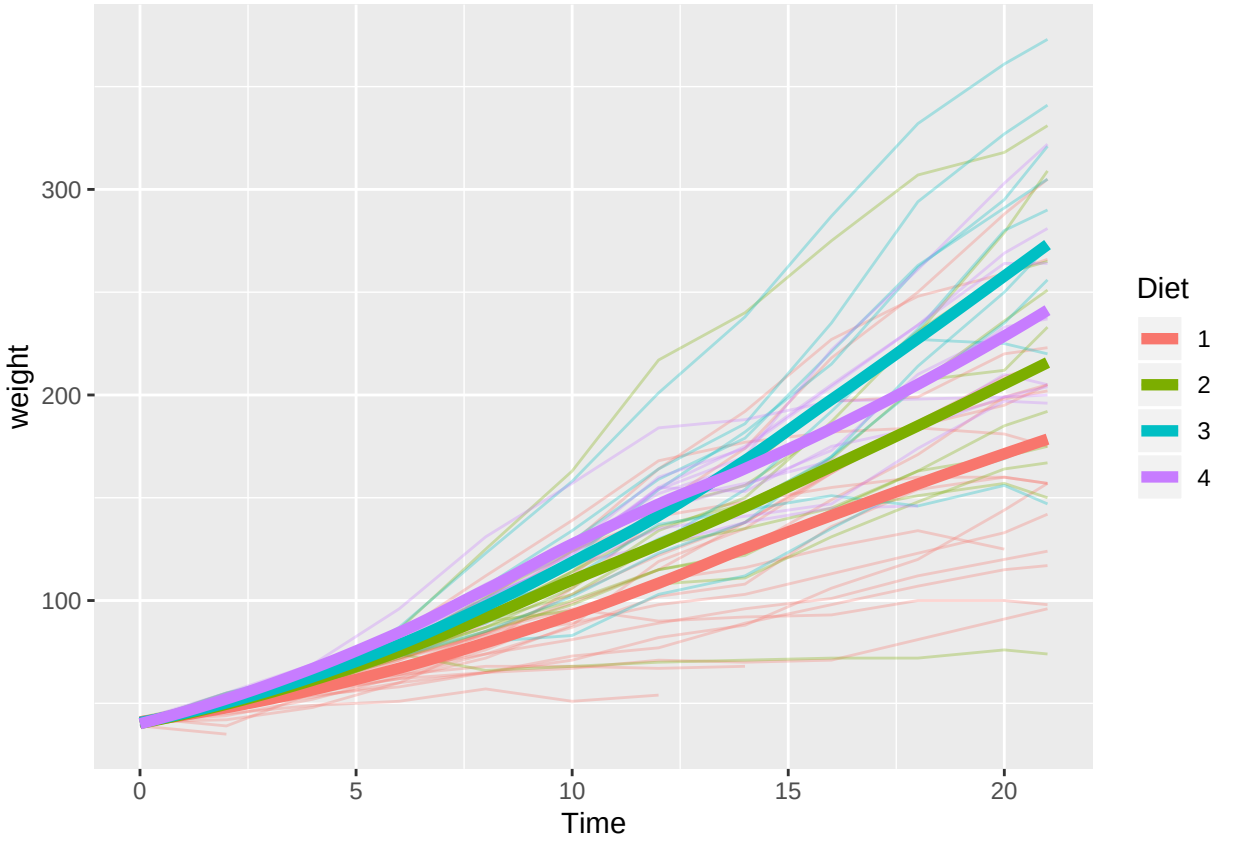



```
# 3 - Take plot 2, map Diet onto col.
ggplot(ChickWeight, aes(x = Time, y = weight, col = Diet)) +
  geom_line(aes(group = Chick))
```




```
# 4 - Take plot 3, add geom_smooth()
ggplot(ChickWeight, aes(x = Time, y = weight, col = Diet)) +
  geom_line(aes(group = Chick), alpha = 0.3) + geom_smooth(lwd = 2, se = FALSE)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Titanic

You've watched the movie Titanic by James Cameron (1997) again and after a good portion of sobbing you decide to investigate whether you'd have a chance of surviving this disaster.

```
library(readr)
titanic <- read_csv("data/titanic.csv")
```

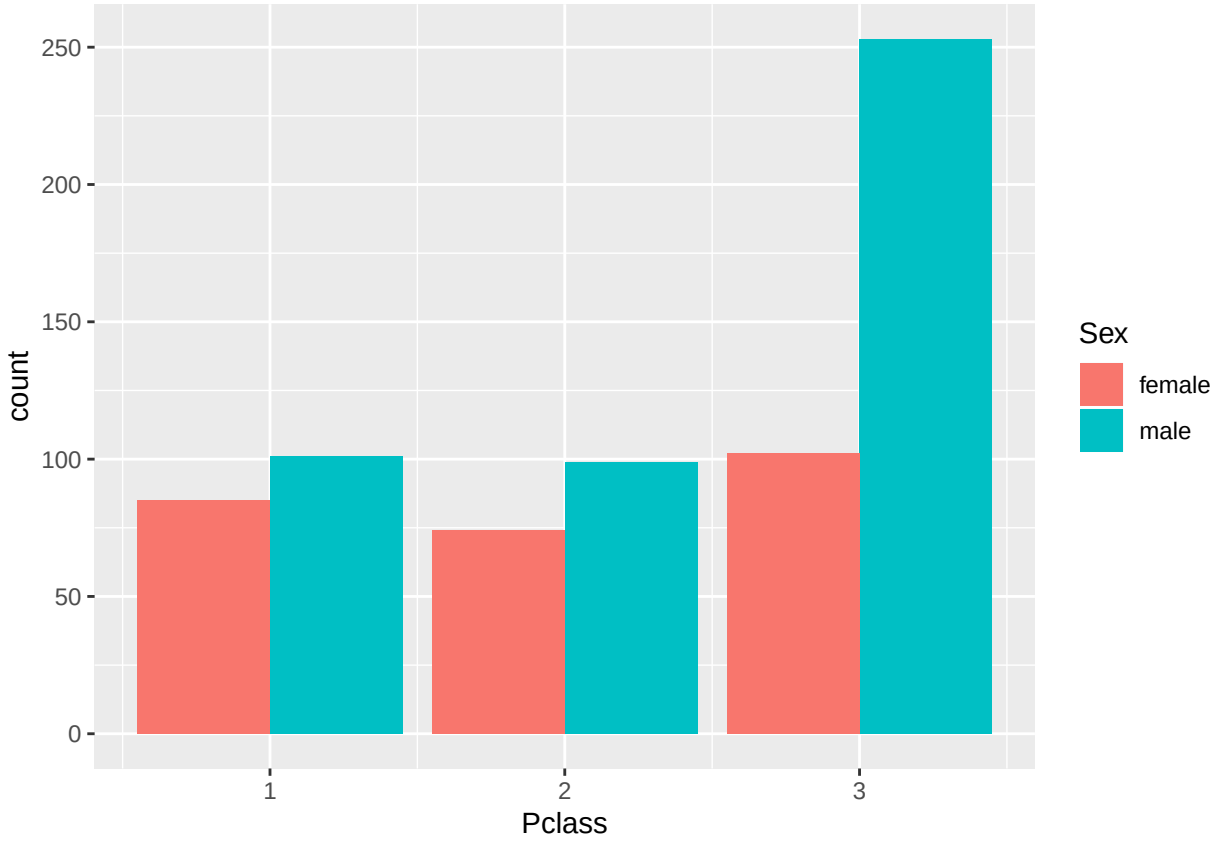
```
## Parsed with column specification:
## cols(
##   PassengerId = col_double(),
##   Survived = col_double(),
##   Pclass = col_double(),
##   Name = col_character(),
##   Sex = col_character(),
##   Age = col_double(),
##   SibSp = col_double(),
##   Parch = col_double(),
##   Ticket = col_character(),
##   Fare = col_double(),
##   Cabin = col_character(),
##   Embarked = col_character()
## )
```

```
titanic <- titanic[,c("Survived","Pclass","Sex","Age")]
titanic <- na.omit(titanic)
```

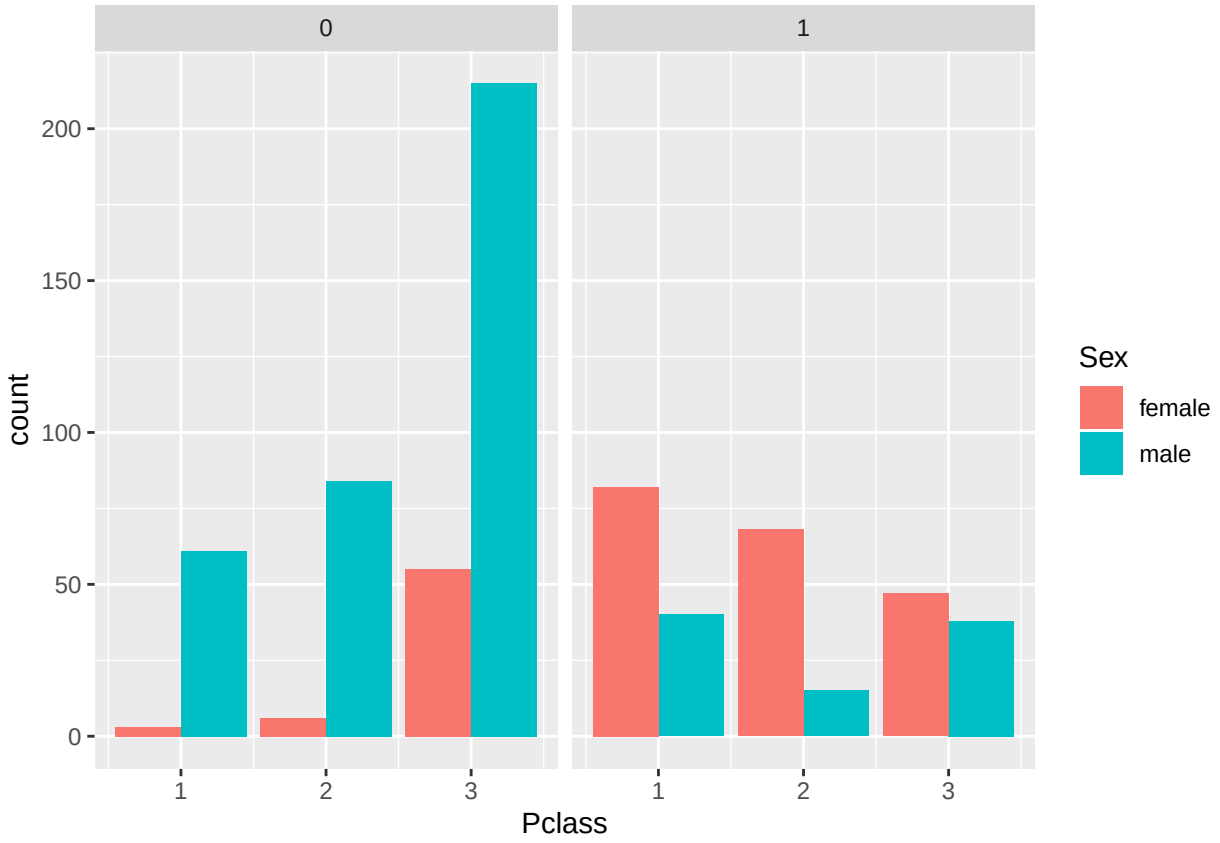
```r
titanic$Survived <- as.integer(titanic$Survived)
titanic$Pclass <- as.integer(titanic$Pclass)
titanic$Sex <- as.factor(titanic$Sex)


# 1 - Check the structure of titanic
str(titanic)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    714 obs. of  4 variables:
##  $ Survived: int  0 1 1 1 0 0 0 1 1 1 ...
##  $ Pclass  : int  3 1 3 1 3 1 3 3 2 3 ...
##  $ Sex     : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 1 1 1 ...
##  $ Age     : num  22 38 26 35 35 54 2 27 14 4 ...
##  - attr(*, "na.action")= 'omit' Named int  6 18 20 27 29 30 32 33 37 43 ...
##   ..- attr(*, "names")= chr  "6" "18" "20" "27" ...
```

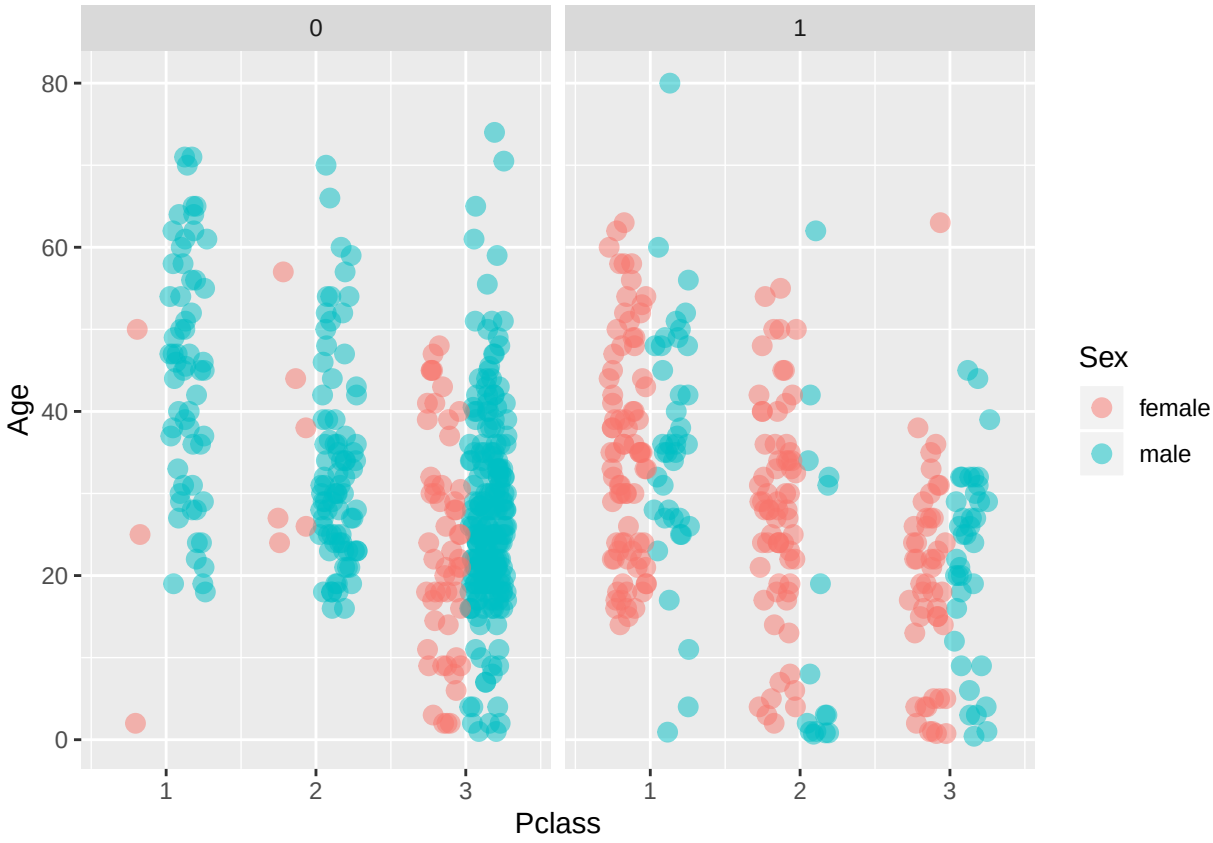
```r
# 2 - Use ggplot() for the first instruction
ggplot(titanic, aes(x = Pclass, fill = Sex)) +
  geom_bar(position = "dodge")
```




```r
# 3 - Plot 2, add facet_grid() layer
ggplot(titanic, aes(x = Pclass, fill = Sex)) +
  geom_bar(position = "dodge") + facet_grid(.~Survived)
```




```
# 4 - Define an object for position jitterdodge, to use below
posn.jd <- position_jitterdodge(0.5, 0, 0.6)

# 5 - Plot 3, but use the position object from instruction 4
ggplot(titanic, aes(x = Pclass, y = Age, col = Sex)) +
  geom_point(position = posn.jd, size = 3, alpha = 0.5) + facet_grid(.~Survived)
```


0
1
80
60
40
20
0
Age
1
2
3
Pclass
Sex
female
male