GGPLOT AND VISUALIZATIONS

Tobias Niedermaier

GGPLOT2

WHAT IS GGPLOT2?

- **ggplot2** is a powerful data visualization package in R.
- It implements the *Grammar of Graphics*, allowing users to build complex plots from simple components.
- Part of the `tidyverse`.
- Install `ggplot2`

```
1  install.packages("ggplot2")
```

- load library in your script

```
1  library(ggplot2)
```

GENERAL SYNTAX

```
ggplot(data = ..., mapping = aes(...)) +
  geom_... +
  ... +
  ...
```

- `data`: the function expects a data frame
- `mapping(...)` Aesthetic mappings: Defines the variables that are mapped to certain visual properties with a function `aes(...)`
- `geom_...`: Geometric objects defining the type of plot

A FIRST EXAMPLE: SCATTERPLOT

- We use `iris` data set as an working example

```
1  library(ggplot2)
2  data("iris", package = "datasets")
3  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
```

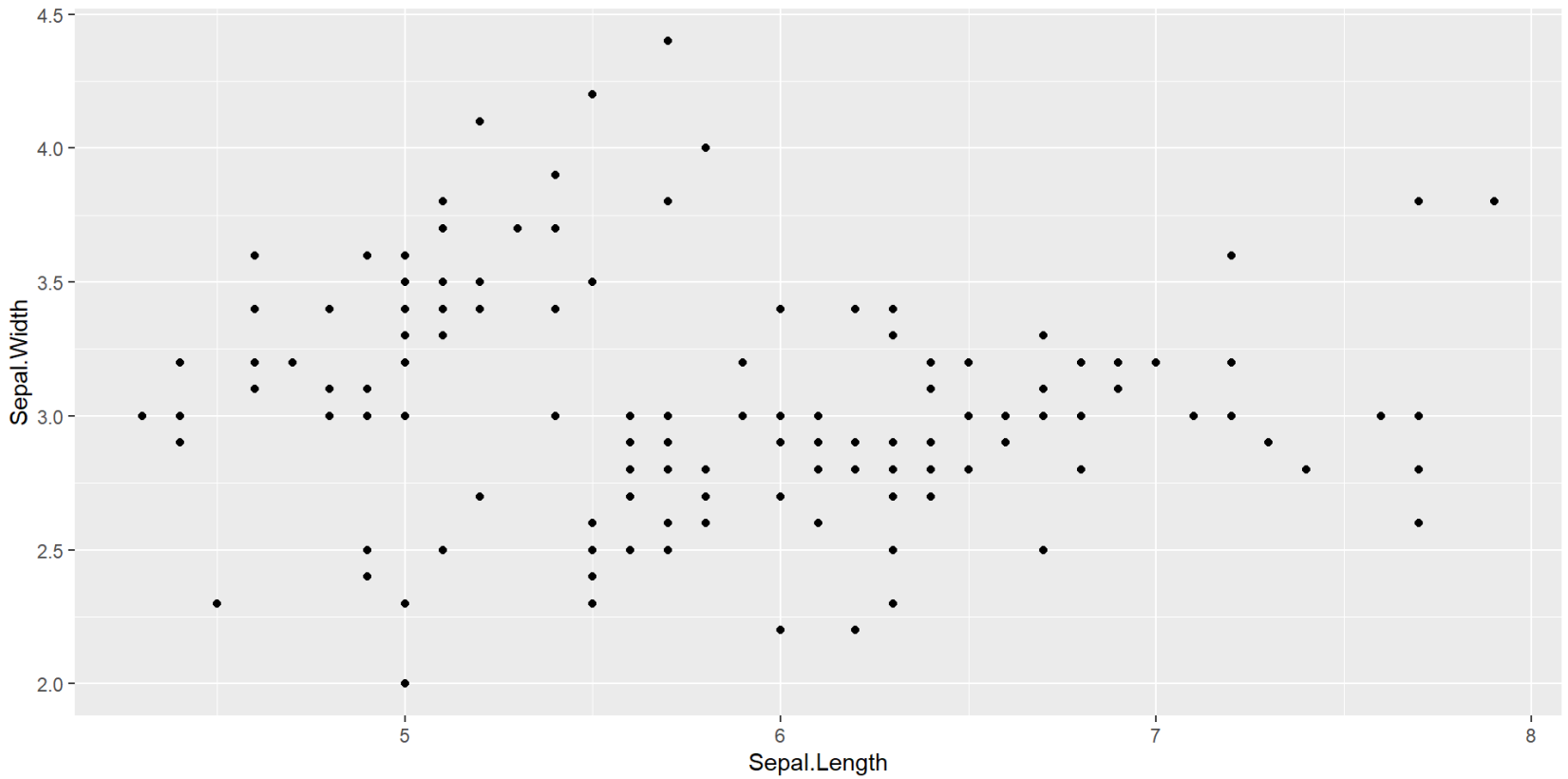
- A scatterplot can be specified using `geom_point()`

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

- Note that the variable names are *not* in quotation marks. Call them as they are actual objects.

A FIRST EXAMPLE: SCATTERPLOT

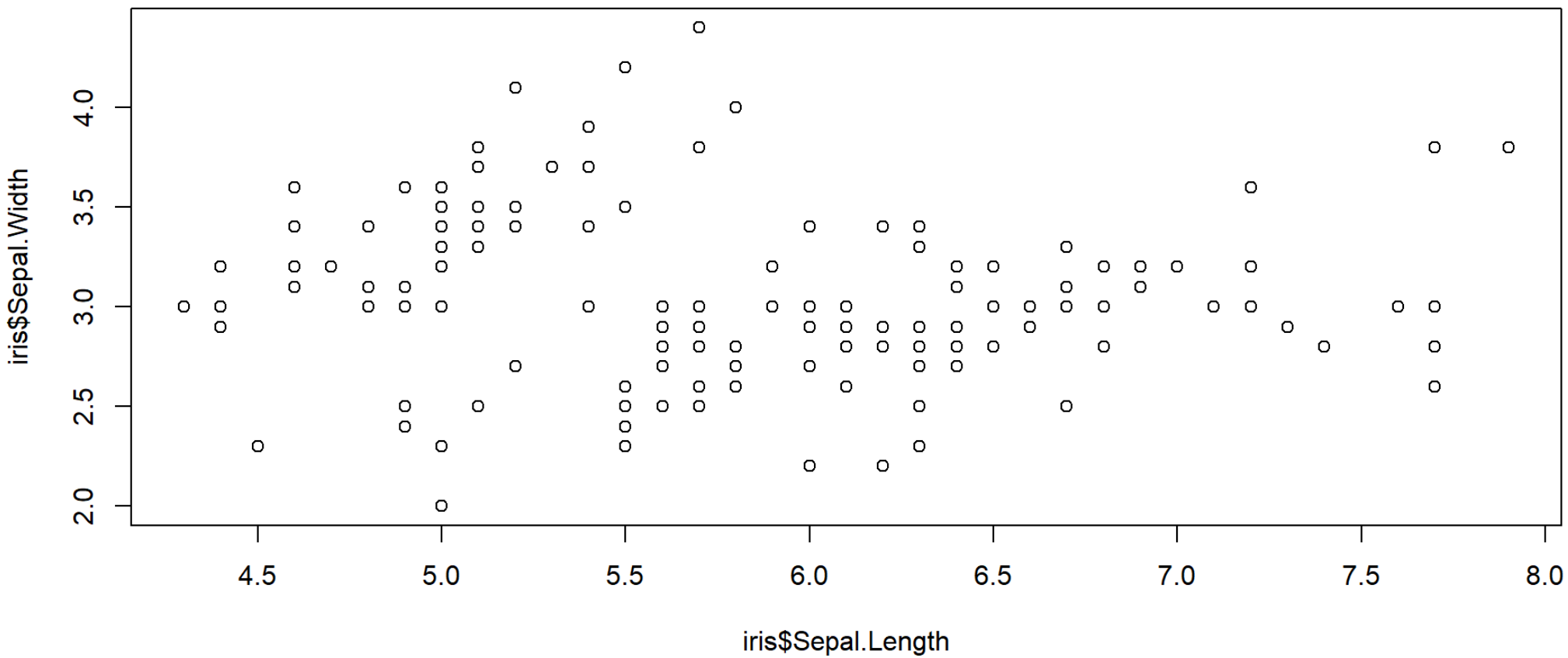
```
1  ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width))+
2    geom_point()
```



- Note that the variable names are *not* in quotation marks. Call them as they are actual objects.

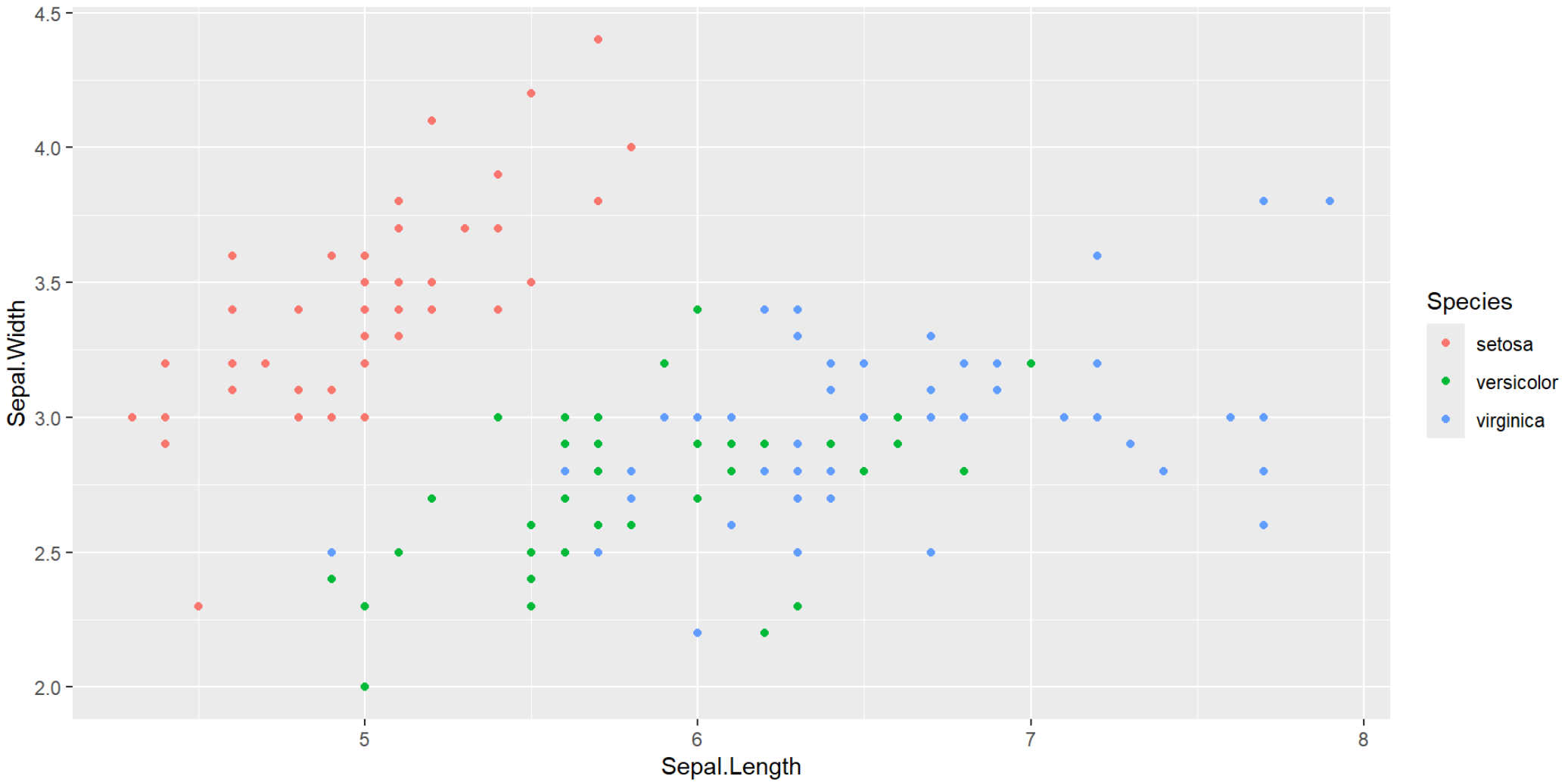
THE SAME (MORE OR LESS) IN BASE R GRAPHICS

```
1  plot(iris$Sepal.Length, iris$Sepal.Width) #add , pch=20 for filled points
```



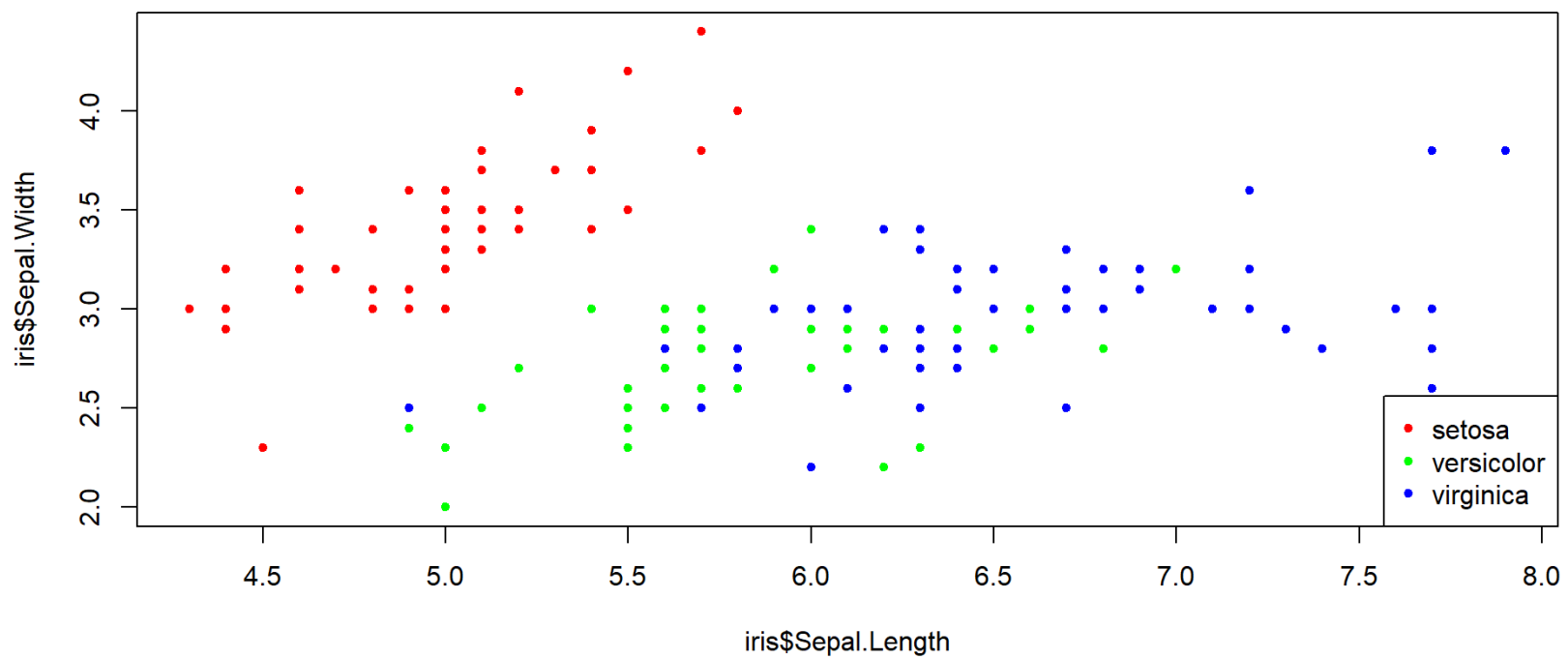
ADDING ANOTHER AESTHETIC MAPPING

```
1  ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species))+
2    geom_point()
```



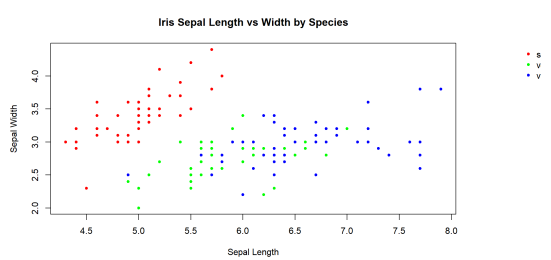
THE SAME (MORE OR LESS) IN BASE R GRAPHICS

```
1  plot(iris$Sepal.Length, iris$Sepal.Width,
2      col=ifelse(iris$Species=="setosa", "red",
3      ifelse(iris$Species=="versicolor", "green", "blue")),
4      pch=20)
5  legend("bottomright", c("setosa","versicolor","virginica"),
6      pch=20,
7    col=c("red","green","blue"))
```



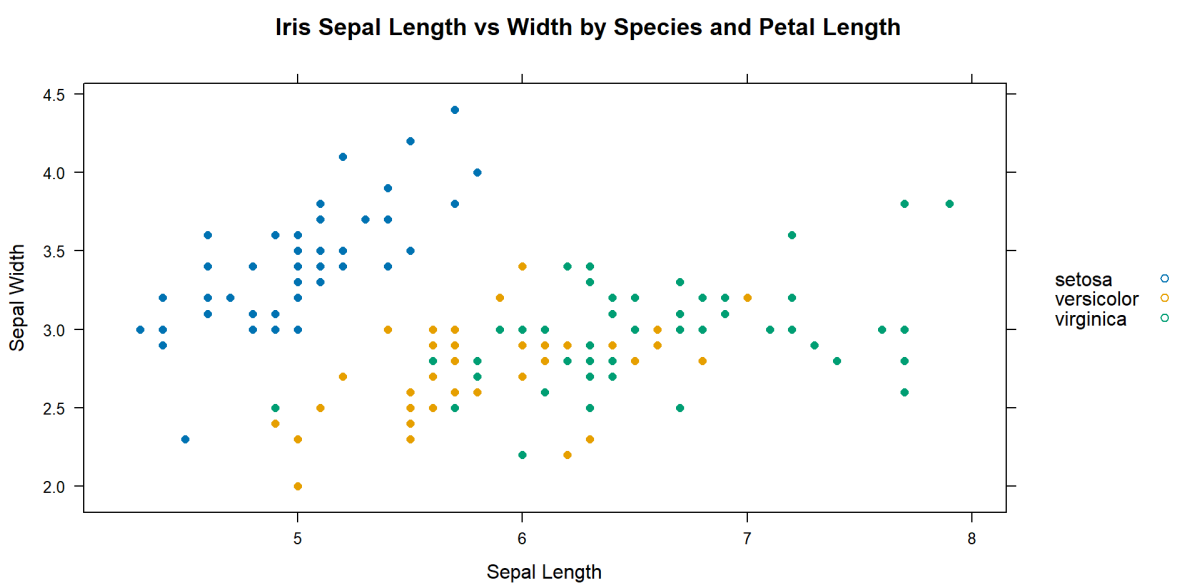
TRY THIS IF YOU WANT TO MAKE IT EVEN MORE SIMILAR TO GGPLOT

```r
# allow extra space on right; allow drawing outside the plot region
par(mar = c(5, 4, 4, 8), xpd = TRUE)

plot(iris$Sepal.Length,
     iris$Sepal.Width,
     col = ifelse(iris$Species == "setosa", "red",
                  ifelse(iris$Species == "versicolor", "green", "blue")),
     pch = 20,
     xlab = "Sepal Length",
     ylab = "Sepal Width",
     main = "Iris Sepal Length vs Width by Species")

# place legend outside on the right
legend("topright",
       inset = c(-0.3, 0),   # shift 0.3 widths to the right
       legend = c("setosa", "versicolor", "virginica"),
       pch = 20,
       col = c("red", "green", "blue"),
       bty = "n")
```



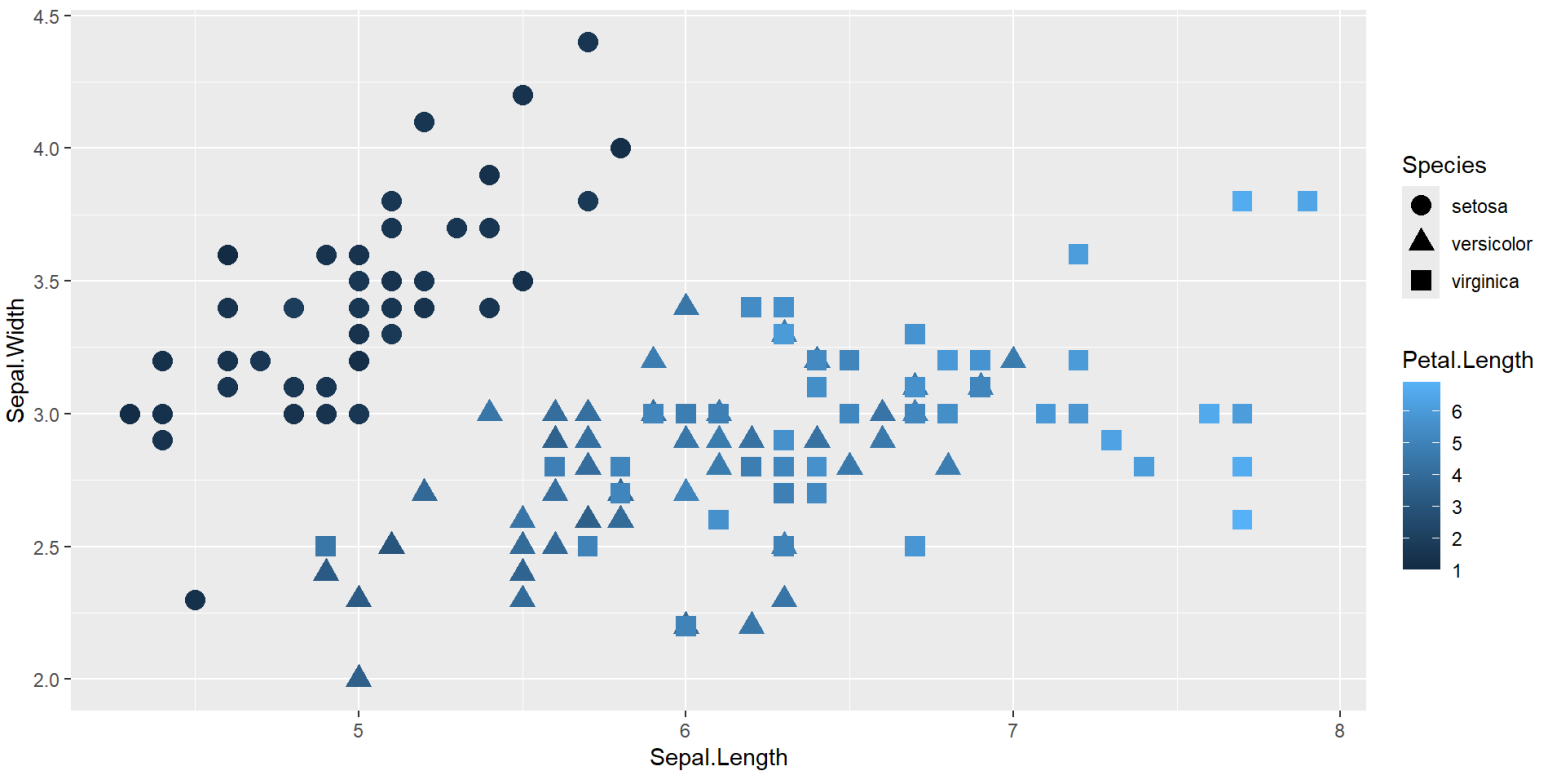
LATTICE ALTERNATIVE

```r
library(lattice)

xyplot(Sepal.Width ~ Sepal.Length,
       data = iris,
       groups = Species,                  # color by Species
       auto.key = list(space = "right"), # add legend
       pch = 16,                         # solid points
       xlab = "Sepal Length",
       ylab = "Sepal Width",
       main = "Iris Sepal Length vs Width by Species and Petal Length",
 # par.settings = list(superpose.symbol = list(pch = 16, cex = 1.1)) #optional: point shape/size
       )
```



ADDING COLOR FOR CONTINUOUS DATA AND SHAPE

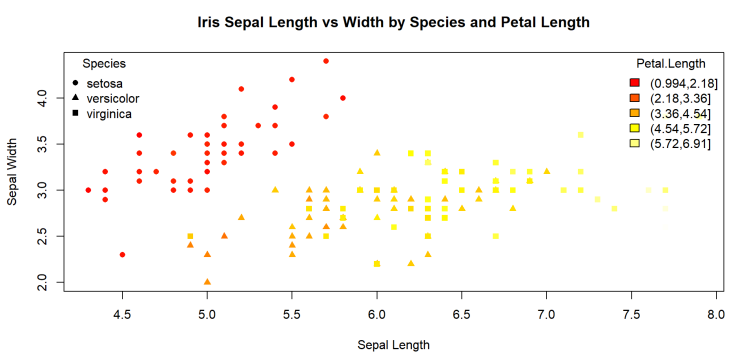
```
1  ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width,
2                          color = Petal.Length, shape = Species))+
3    geom_point(size = 4)
```



- We added a `size` argument to the geom_point-function to make the points larger

BASE R ALTERNATIVE (ADMITTEDLY MORE COMPLICATED THAN GGPLOT OR LATTICE)

```r
pal  <- heat.colors(5)
bins <- cut(iris$Petal.Length, breaks = 5)          # bin Petal.Length
pch_map <- c(setosa = 16, versicolor = 17, virginica = 15)
plot(iris$Sepal.Length, iris$Sepal.Width,
     col = heat.colors(100)[as.numeric(cut(iris$Petal.Length, breaks = 100))],
     pch = ifelse(iris$Species == "setosa", 16,
                  ifelse(iris$Species == "versicolor", 17, 15)),
     xlab = "Sepal Length",
     ylab = "Sepal Width",
     main = "Iris Sepal Length vs Width by Species and Petal Length")
legend("topleft", title = "Species",
       legend = names(pch_map), pch = pch_map, bty = "n")
legend("topright", title = "Petal.Length",
       fill = pal, legend = levels(bins), bty = "n")
```



IRIS (CODE)

```
pal  <- heat.colors(5)
bins <- cut(iris$Petal.Length, breaks = 5)
pch_map <- c(setosa = 16, versicolor = 17, virginica = 15)
plot(iris$Sepal.Length, iris$Sepal.Width,
     col = heat.colors(100)[as.numeric(cut(iris$Petal.Length, breaks = 100))],
     pch = ifelse(iris$Species == "setosa", 16,
                  ifelse(iris$Species == "versicolor", 17, 15)),
     xlab = "Sepal Length",
     ylab = "Sepal Width",
     main = "Iris Sepal Length vs Width by Species and Petal Length")
legend("topleft", title = "Species",
       legend = names(pch_map), pch = pch_map, bty = "n")
legend("topright", title = "Petal.Length",
       fill = pal, legend = levels(bins), bty = "n")
```

IRIS (PLOT)


Iris Sepal Length vs Width by Species and Petal Length
Species
setosa
versicolor
virginica
Petal.Length
(0.994,2.18]
(2.18,3.36]
(3.36,4.54]
(4.54,5.72]
(5.72,6.91]
Sepal Width
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Sepal Length

EXERCISES 3 TASKS 1

LINES

We define a simple function

$$f(x) = sin(x) + cos(x \cdot 0.5)$$

```
1  foo <- function(x) sin(x) + cos(x*0.5)
2  x <- seq(0, 20, len = 100)
3  y <- foo(x)
```

- We now deliberately ignore the `data` argument in ggplot and just define x and y.

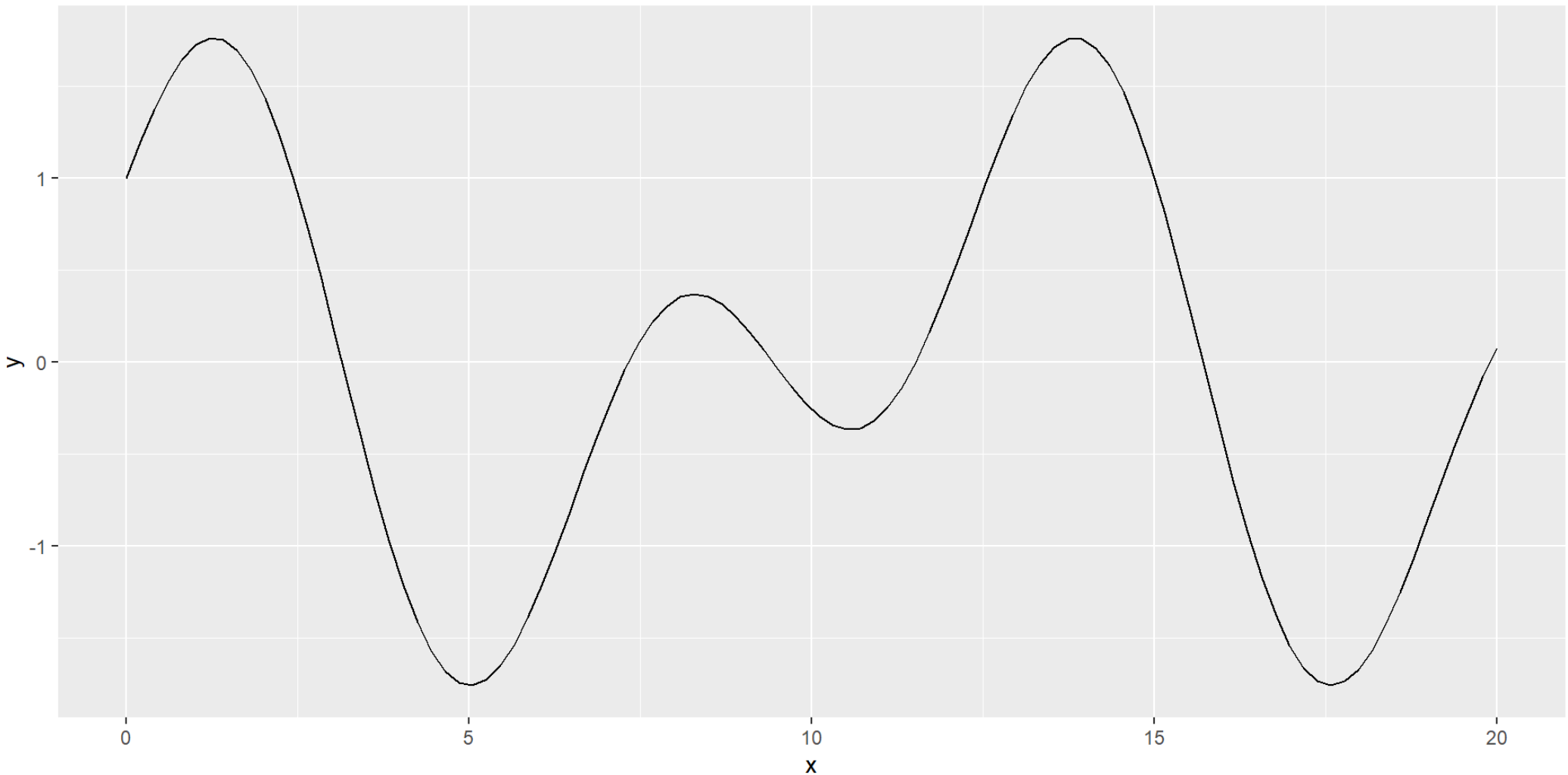
```
1  ggplot(mapping = aes(x = x, y = y))+
2    geom_line()
```

- Same in Base R:

```
1  plot(x, y, type="l")
```

LINES

```
1  ggplot(mapping = aes(x = x, y = y))+
2    geom_line()
```



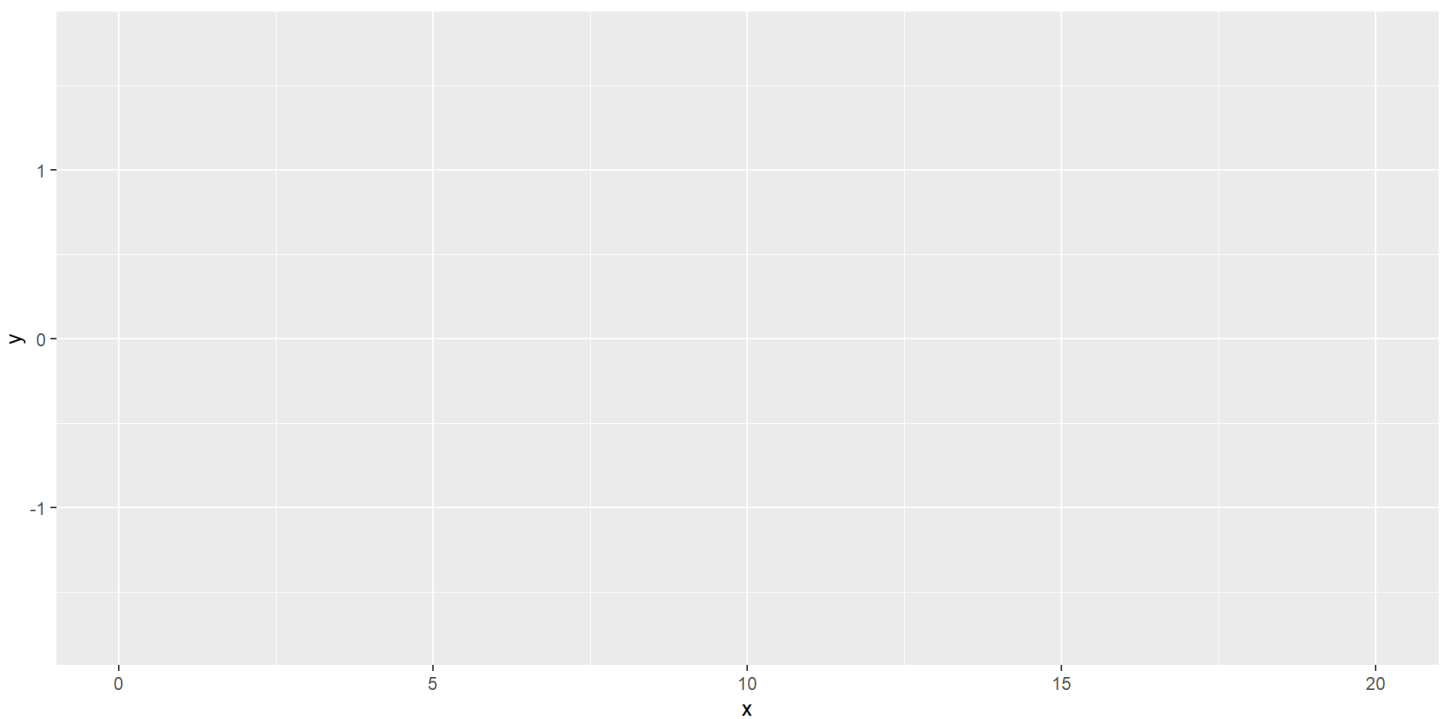
GGPLOT OBJECTS

- We can assign the ggplot as an object…

```
1  g <- ggplot(mapping = aes(x = x, y = y))
```

- and look at it:

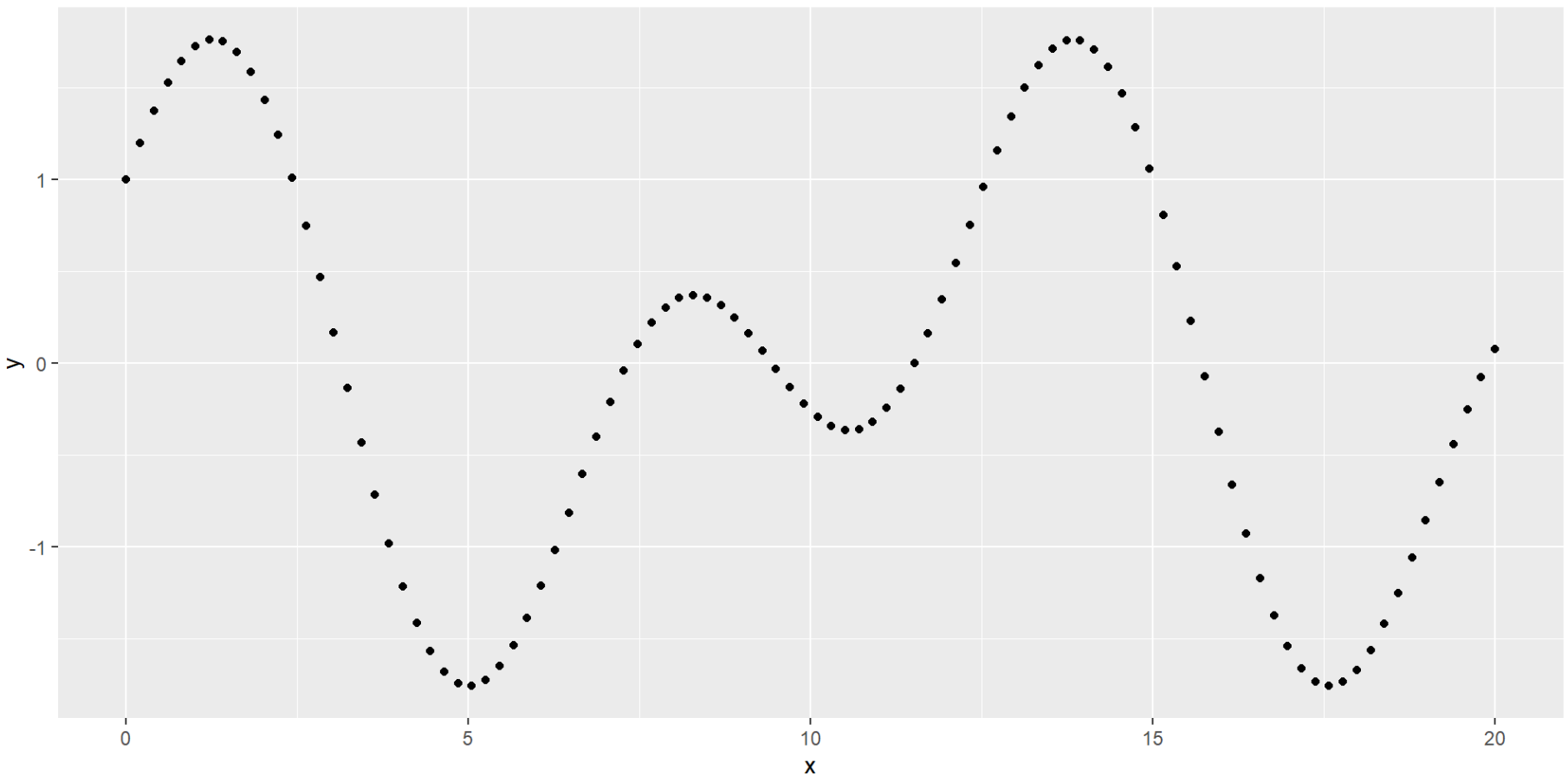
```
1  g
```



GGPLOT OBJECTS

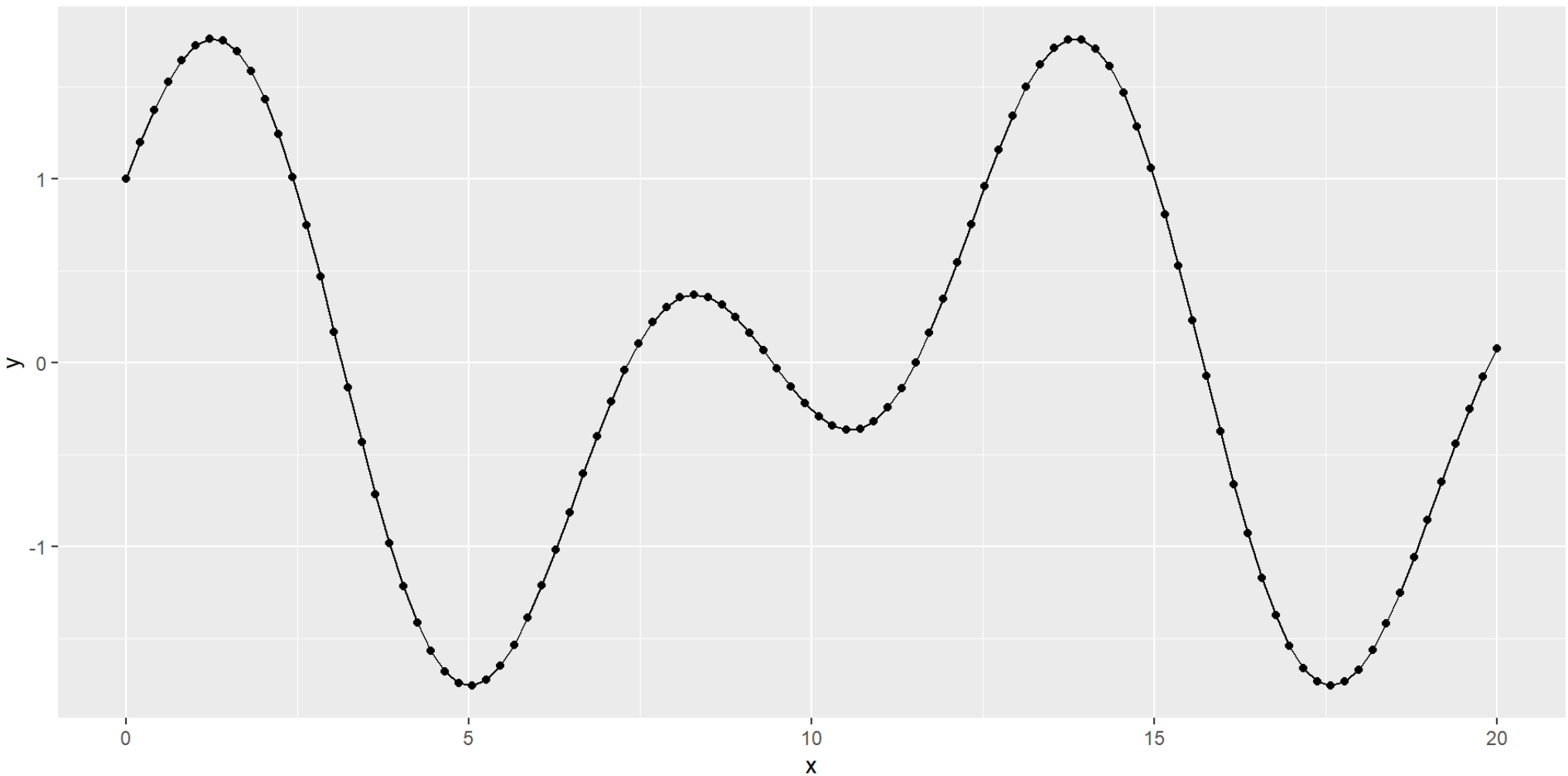
- adding layers later to an object:

```
1  g +
2    geom_point()
```



ADD MULTIPLE LAYERS

```
1 g +
2   geom_point()+
3   geom_line()
```



SUBPLOTS

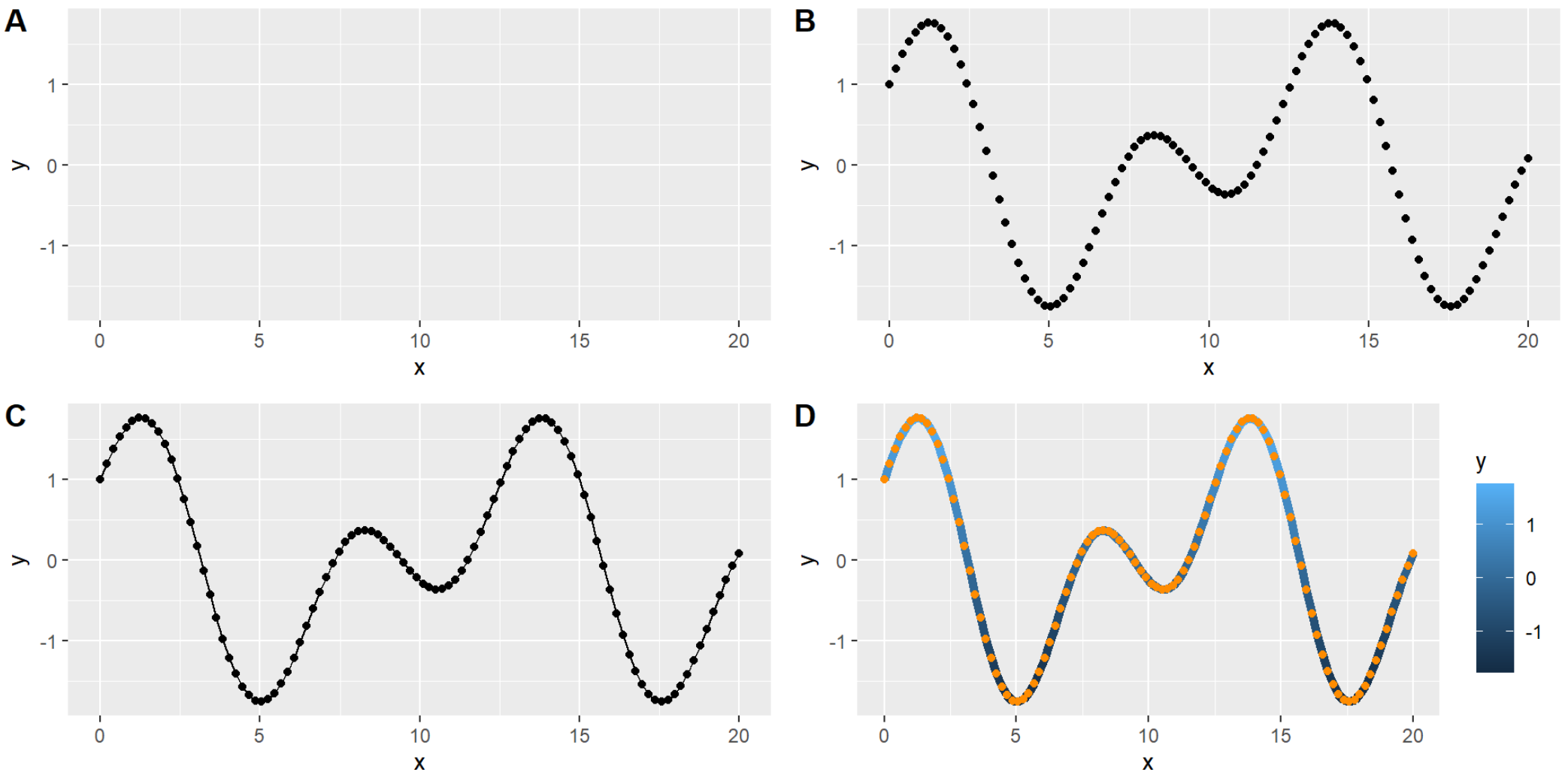
Multiple plots can be designed using external packages. Here, we use `cowplot`.

```
 1  library(cowplot)
 2
 3  # assign two objects
 4  g_point <- g +
 5    geom_point()
 6
 7  g_point_line <- g +
 8    geom_point()+
 9    geom_line()
10
11  g_point_line_color <- g +
12    geom_line(aes(color = y), linewidth=2)+
13    geom_point(color = "darkorange")
14
15  plot_grid(g, g_point, g_point_line, g_point_line_color,
16            nrow = 2, ncol = 2,
17            labels="AUTO")
```

Note that we have different color arguments:

- In line 12 *inside* `aes(...)` with a variable name
- In line 13 *outside* of `aes(...)`
- Control line width accordingly using `linewidth` (here: outside `aes(...)`)

SUBPLOTS



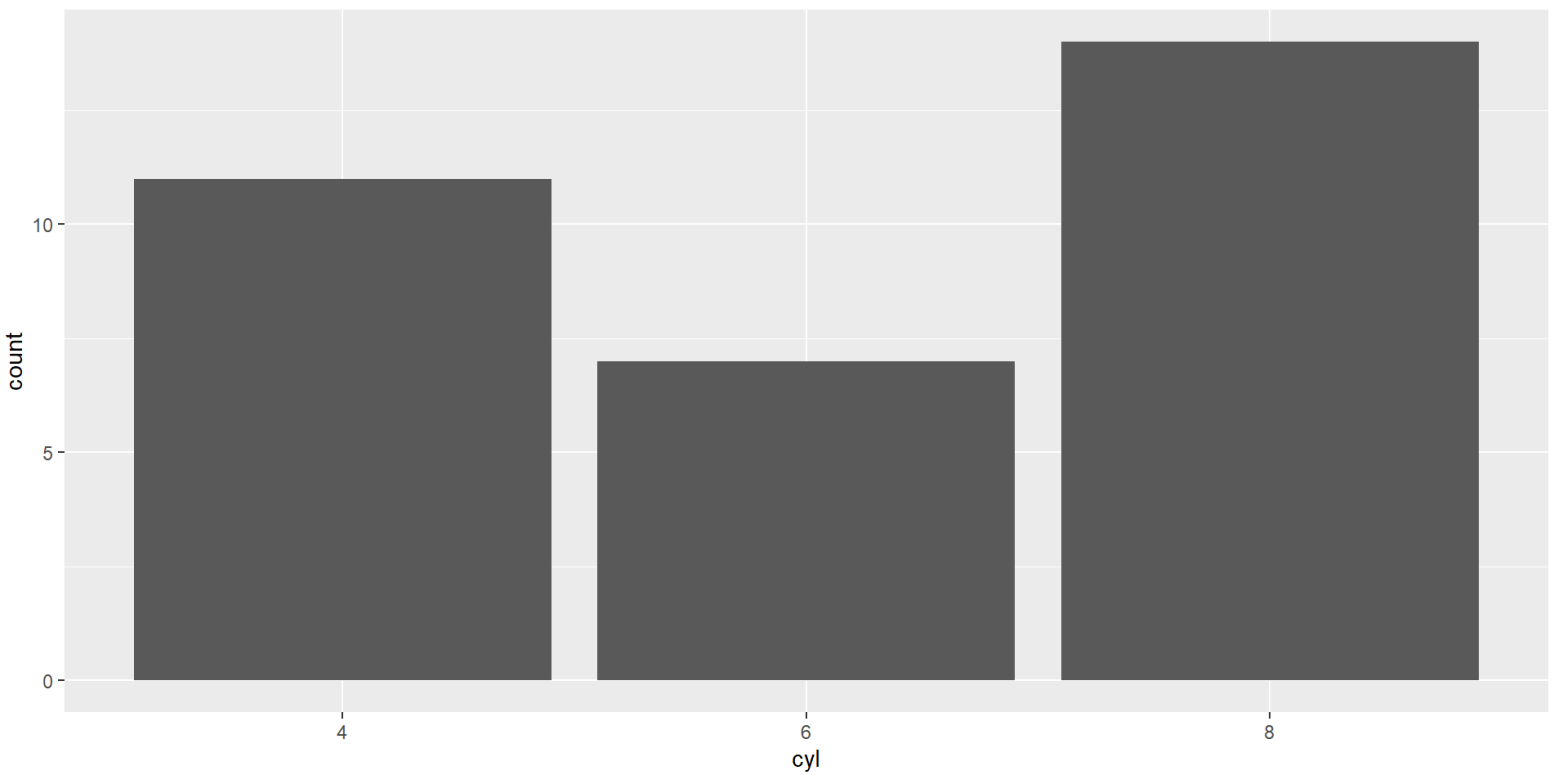
EXERCISES 3 TASKS 2

OTHER TYPES OF PLOT

BARPLOT

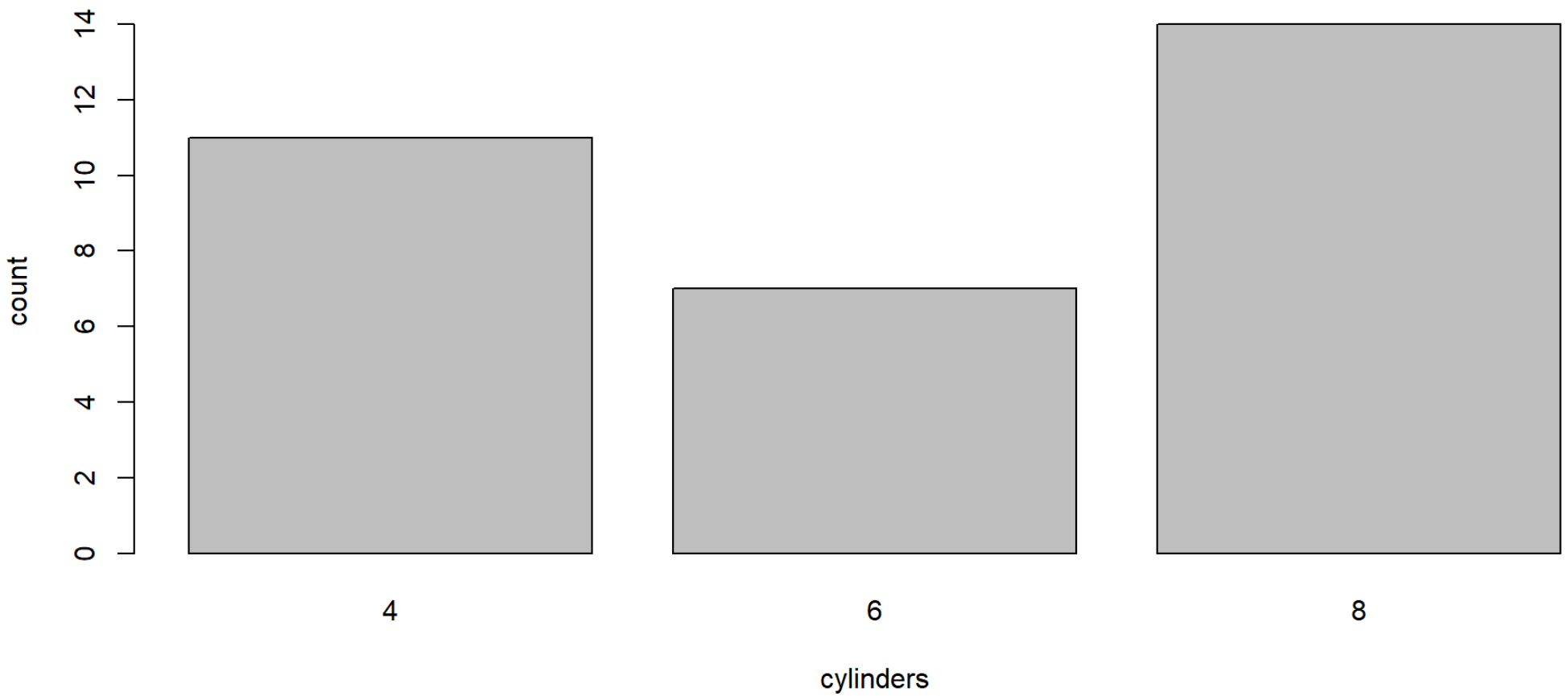
The syntax stays the same for a type of plots. - A barplot only requires aesthetics for x. - We use the `mtcars` data set as an example

```
1  data <- mtcars
2  data$cyl <- as.factor(data$cyl)
3  ggplot(data, aes(cyl))+
4    geom_bar()
```



THE SAME (MORE OR LESS) IN BASE R GRAPHICS

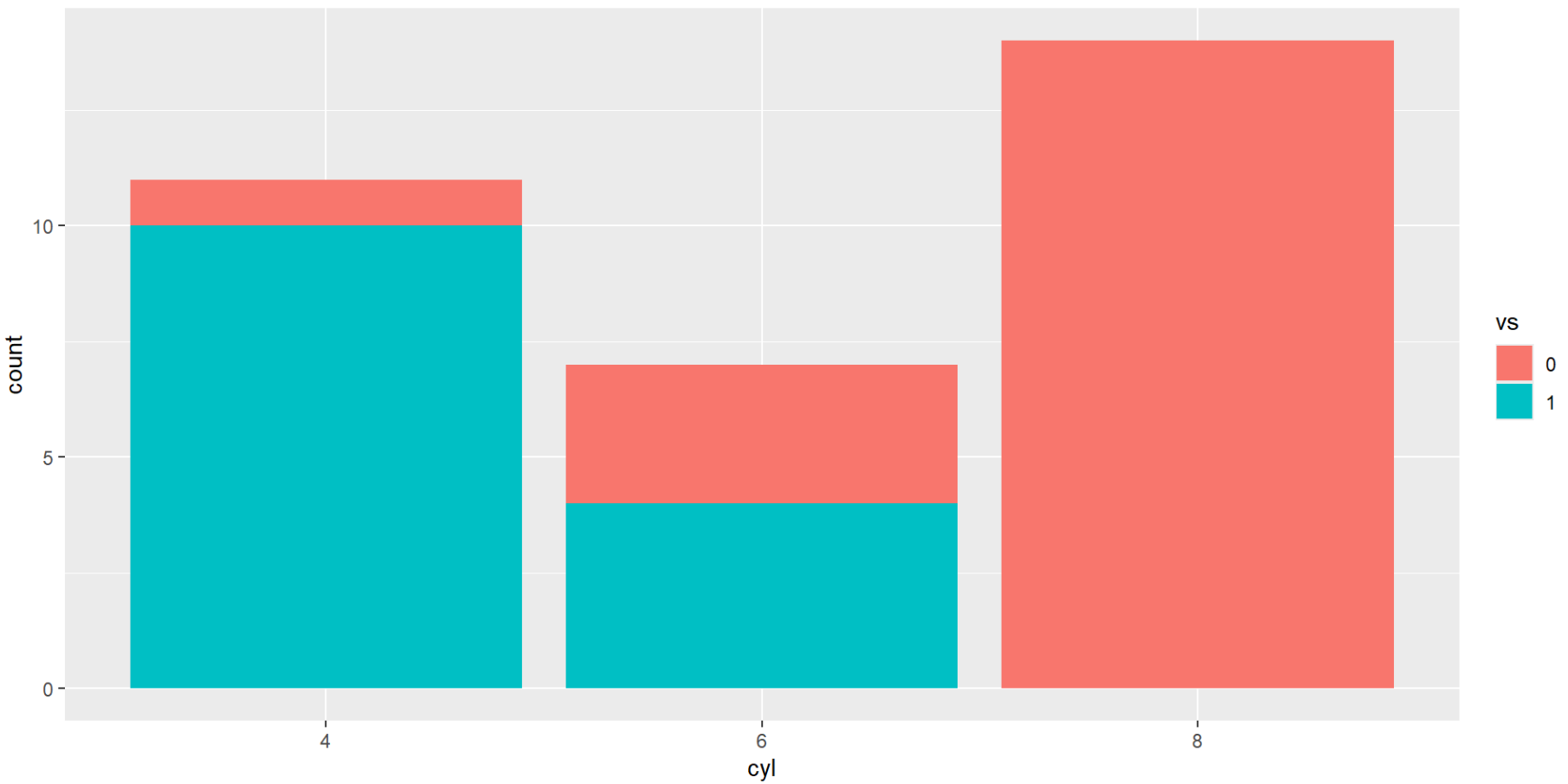
```
1  barplot(table(mtcars$cyl), xlab="cylinders", ylab="count")
```



ADD COLOR

Use `fill` instead of color here.

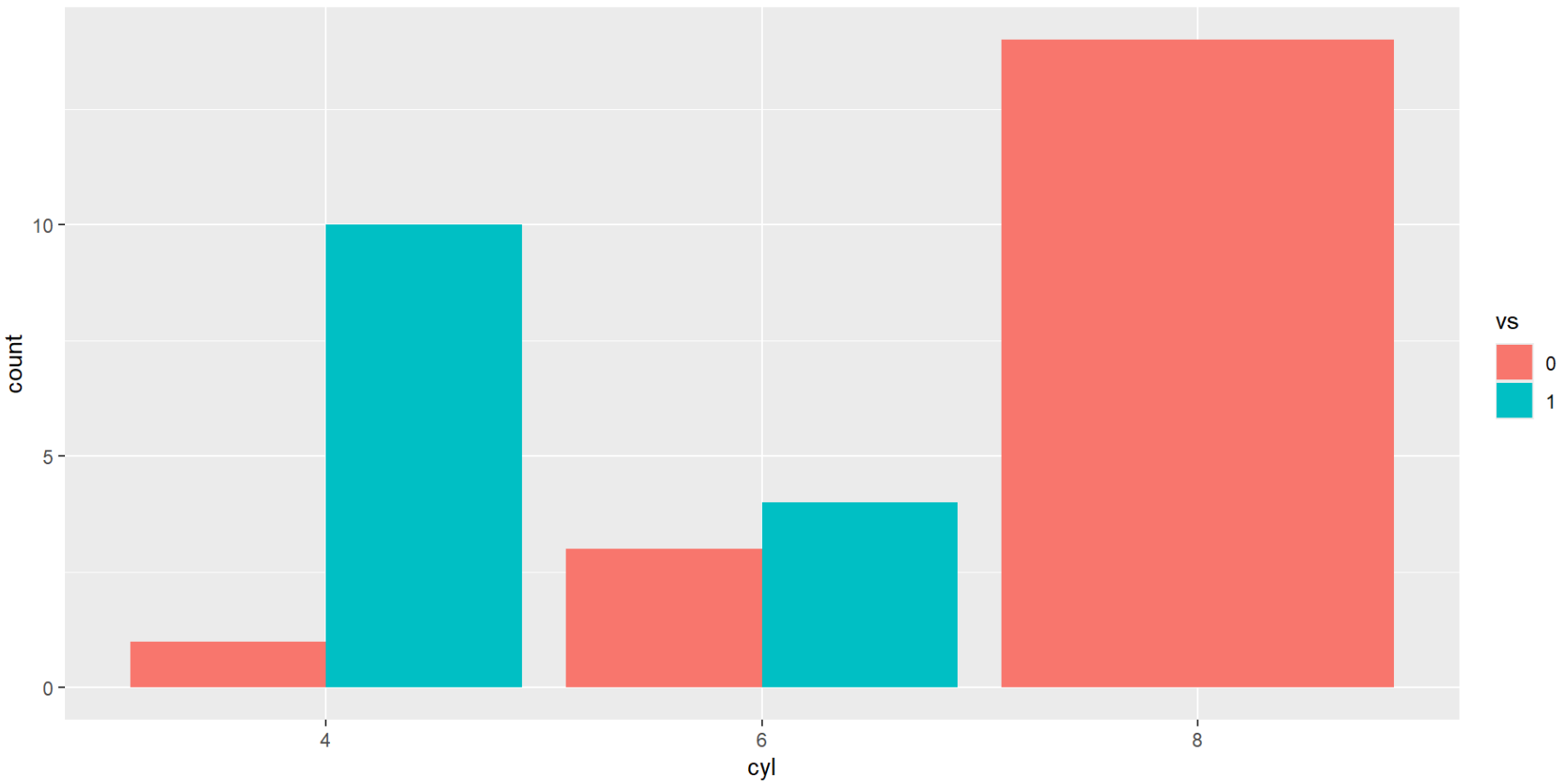
```
1  data$vs <- as.factor(data$vs)
2  ggplot(data, aes(cyl, fill = vs))+
3    geom_bar()
```



ADD COLOR

Side by side:

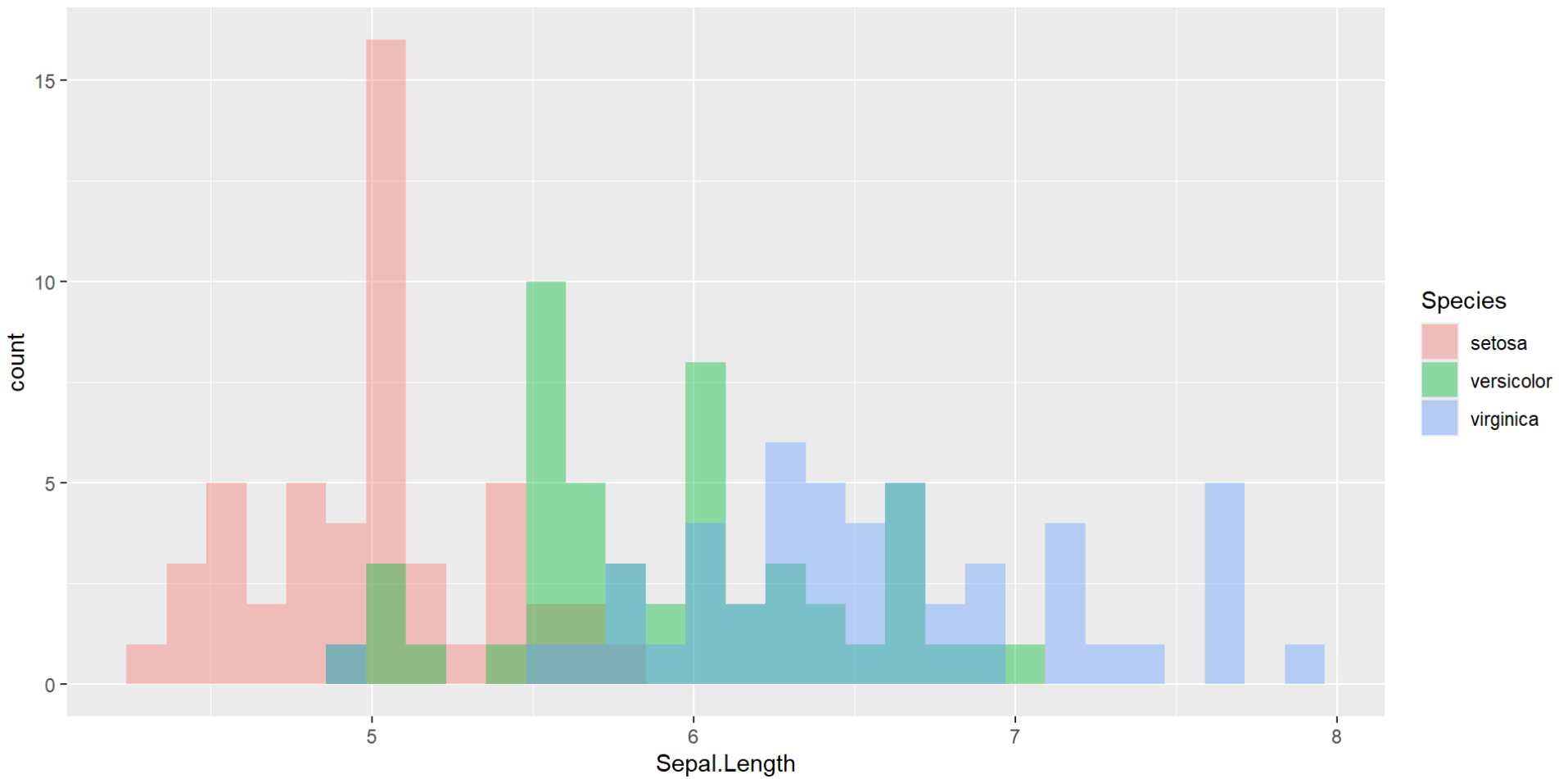
```
1  data$vs <- as.factor(data$vs)
2  ggplot(data, aes(cyl, fill = vs))+
3    geom_bar(position = "dodge")
```



HISTOGRAM

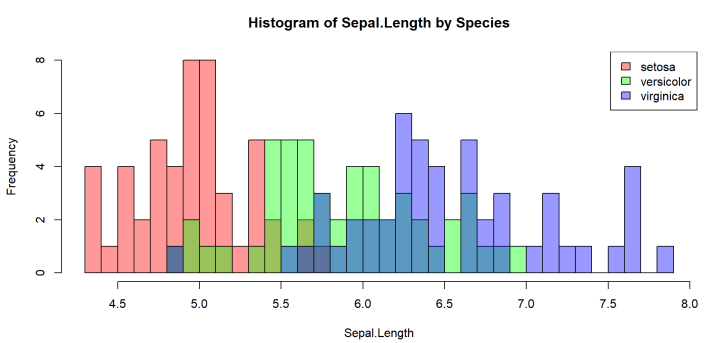
Here, we use `iris` again. - `position = "identity"` to overplot histograms

```
1  ggplot(iris, aes(Sepal.Length, fill = Species))+
2    geom_histogram(bins = 30, alpha = 0.4, position = "identity") # alpha for transparency
```



BASE R ALTERNATIVE

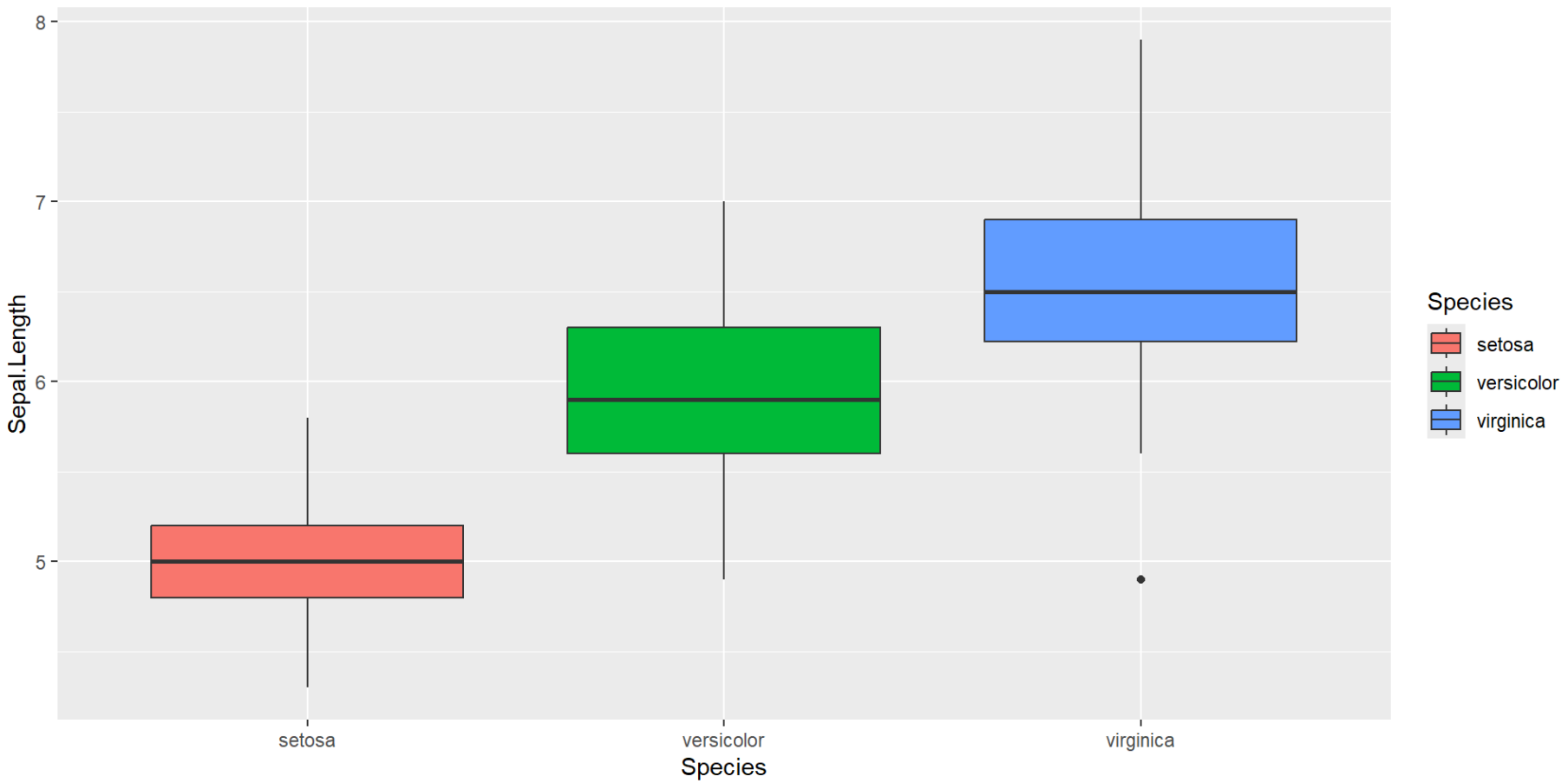
```r
x <- iris$Sepal.Length
brks <- hist(x, breaks = 30, plot = FALSE)$breaks  # common breaks

hist(x[iris$Species == "setosa"],
     col  = rgb(1,0,0,0.4),
     xlab = "Sepal.Length",
     main = "Histogram of Sepal.Length by Species",
     breaks = brks)

hist(x[iris$Species == "versicolor"],
     col = rgb(0,1,0,0.4), add = TRUE, breaks = brks)
hist(x[iris$Species == "virginica"],
     col = rgb(0,0,1,0.4), add = TRUE, breaks = brks)

legend("topright",
       legend = c("setosa","versicolor","virginica"),
       fill   = c(rgb(1,0,0,0.4), rgb(0,1,0,0.4), rgb(0,0,1,0.4)))
```



BOXPLOT

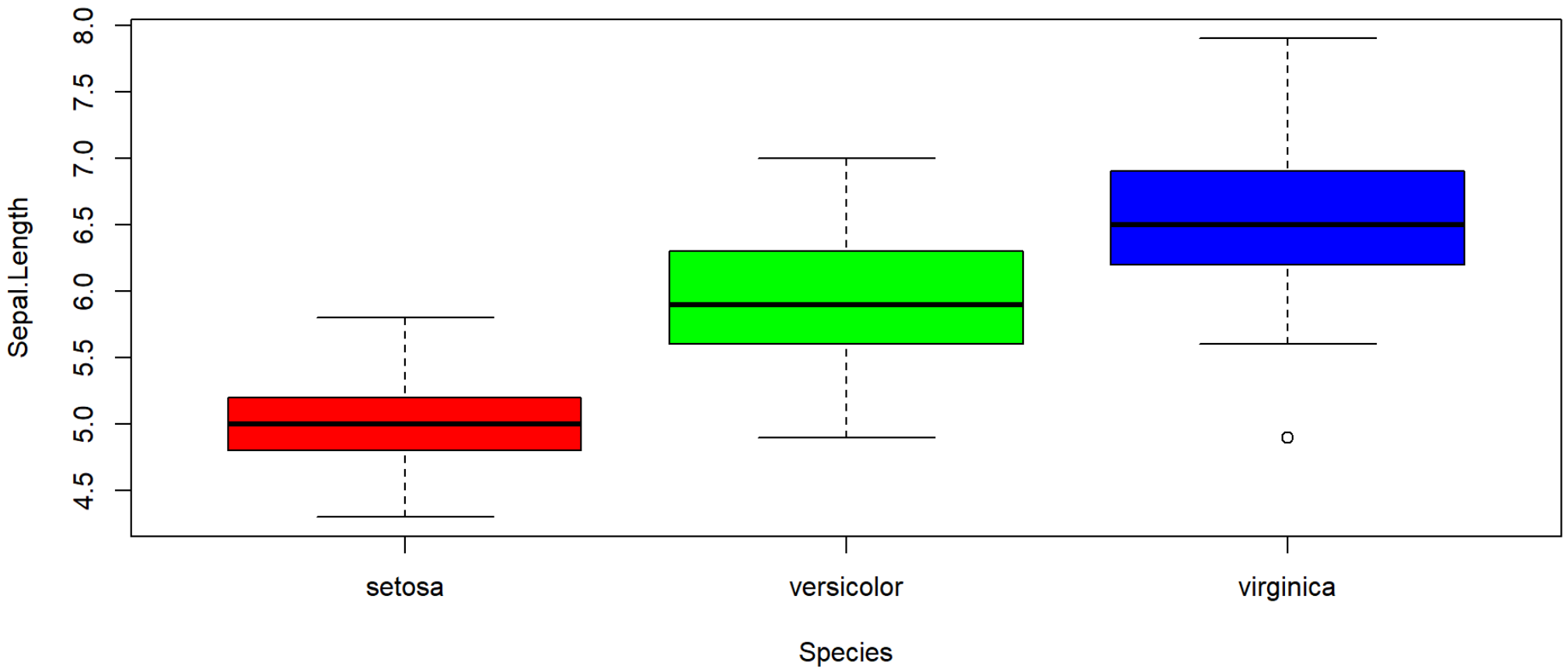
- Note that we have `Species` on the x-axis *and* as fill color

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



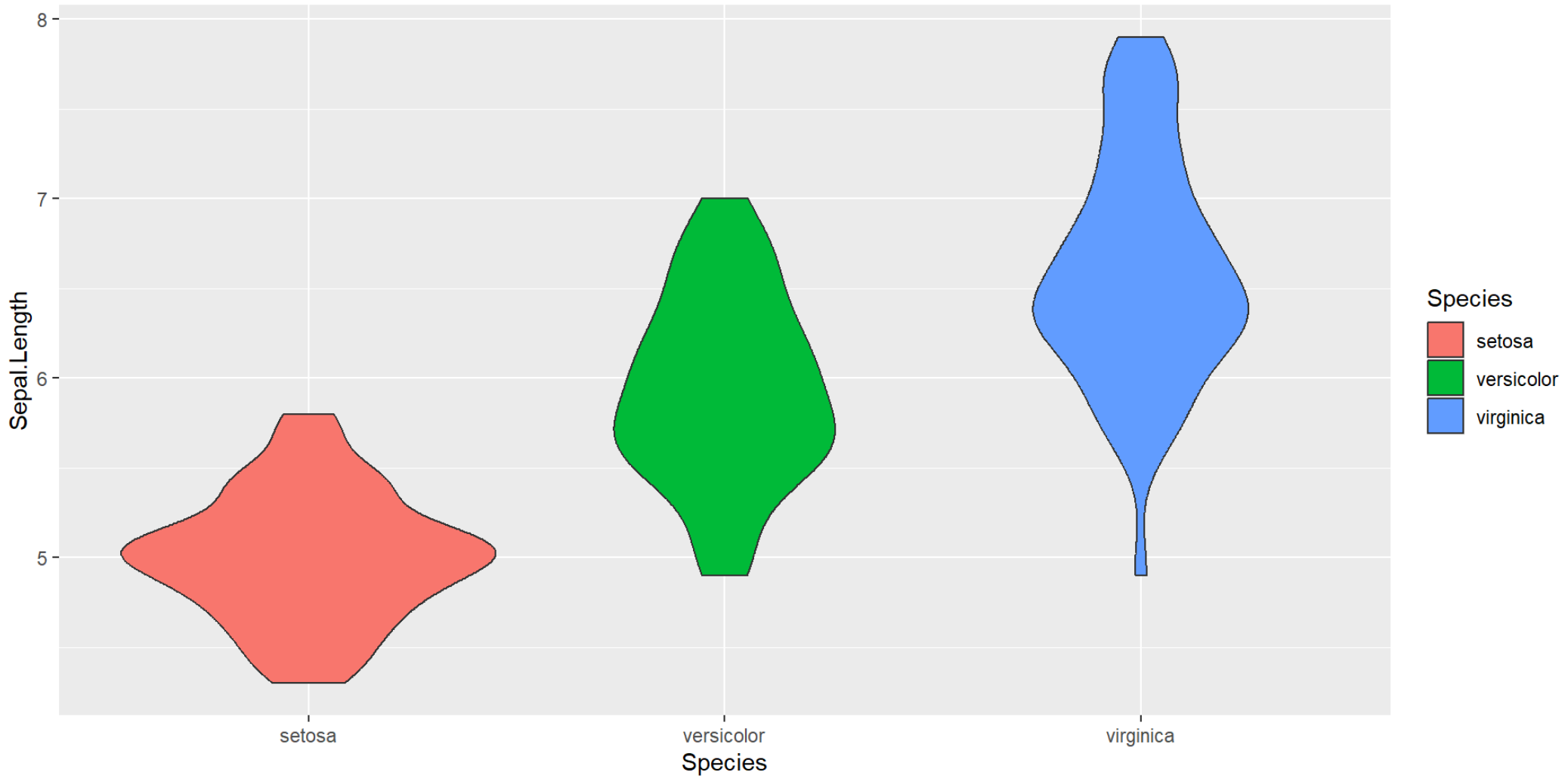
A SIMILAR BOXPLOT IN BASE R GRAPHICS

```
1  boxplot(Sepal.Length ~ Species, data=iris, col=c("red","green","blue"))
```



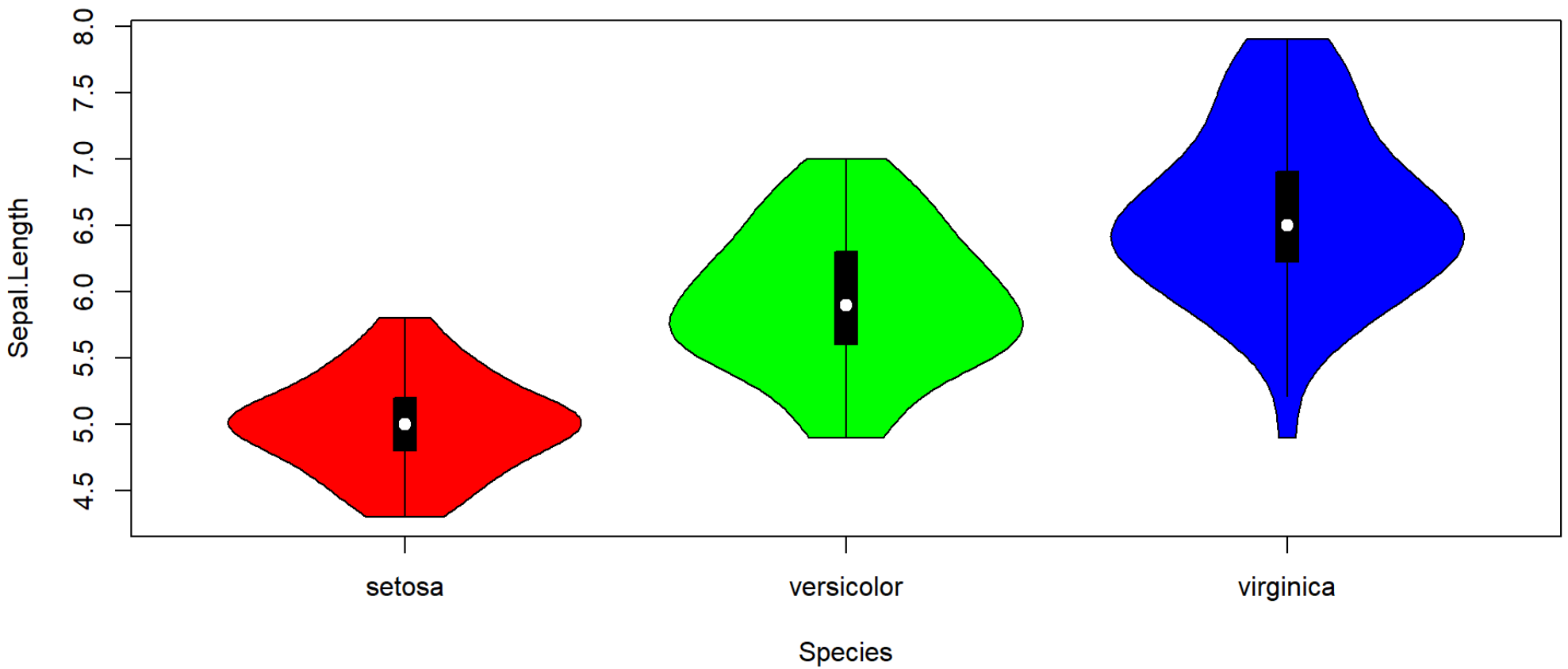
VIOLIN

```
1  ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species))+
2    geom_violin()
```



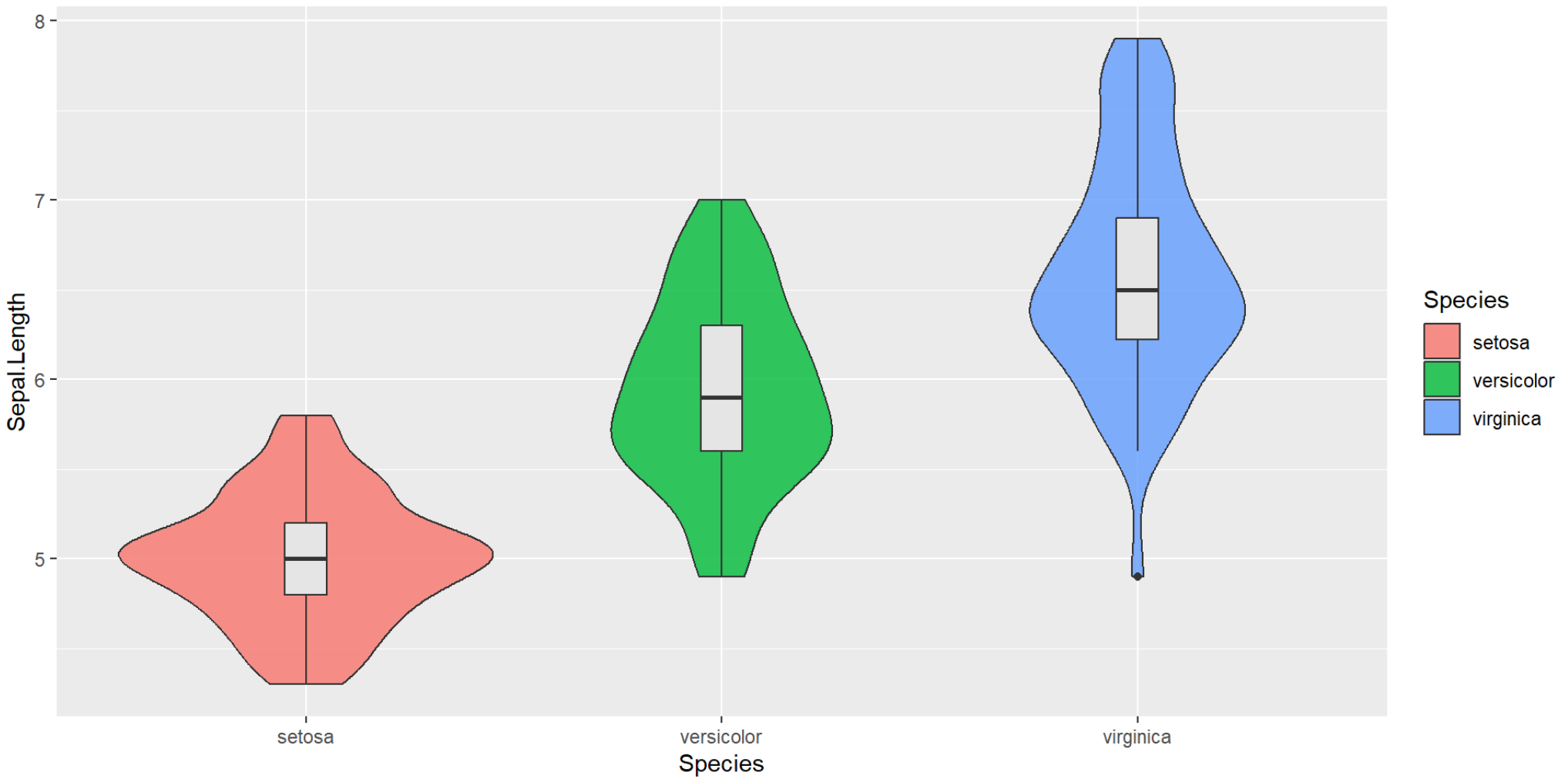
OR IF YOU PREFER A SYNTAX CLOSER TO BASE R GRAPHICS

```
1  #install.packages("vioplot")
2  library(vioplot)
3  vioplot(Sepal.Length ~ Species, data=iris, col=c("red","green","blue"))
```



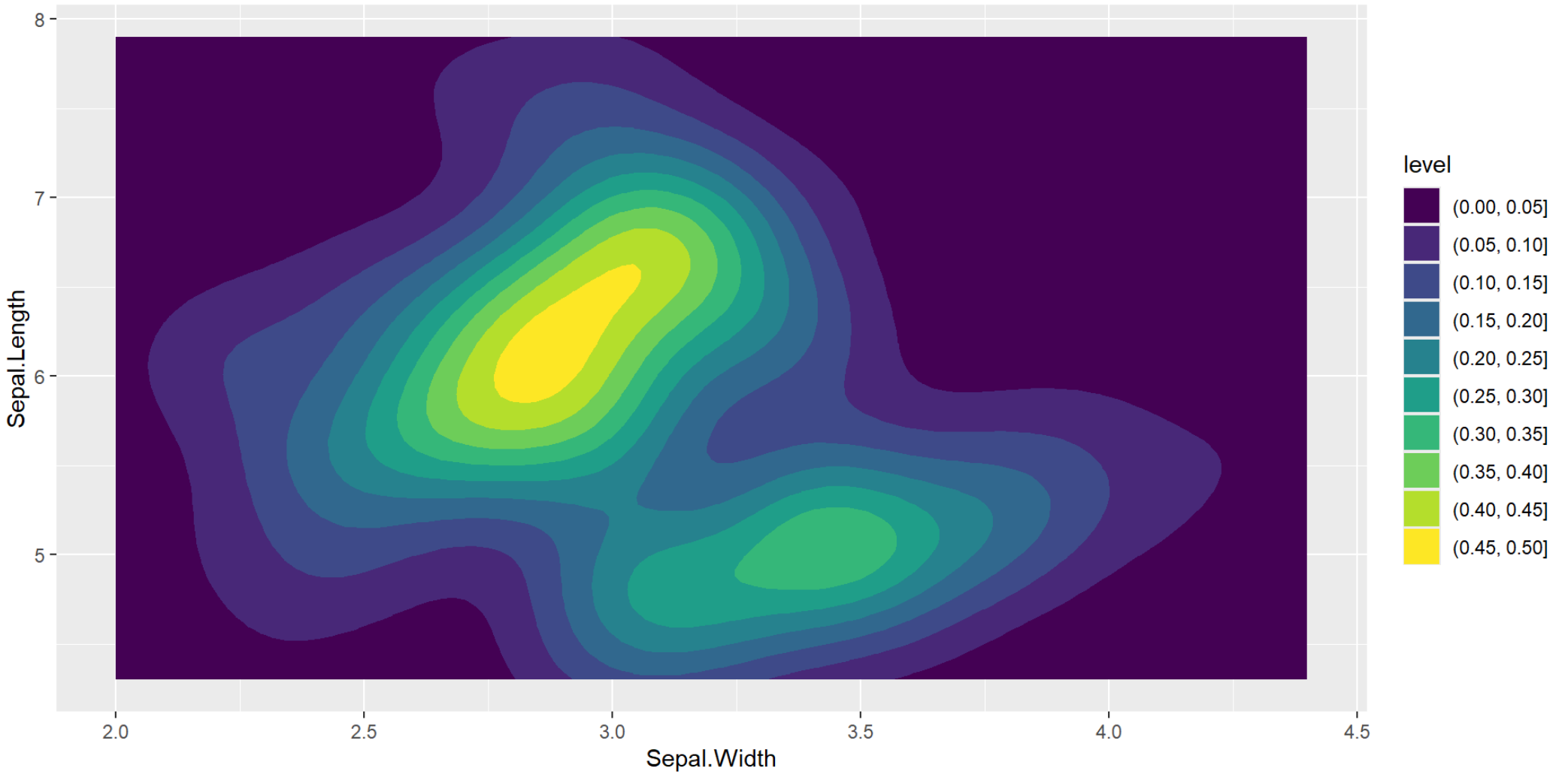
COMBINATION

```
1  ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species))+
2    geom_violin(alpha = 0.8)+
3    geom_boxplot(width=0.1, fill="grey90")
```



2-DIM DENSITY

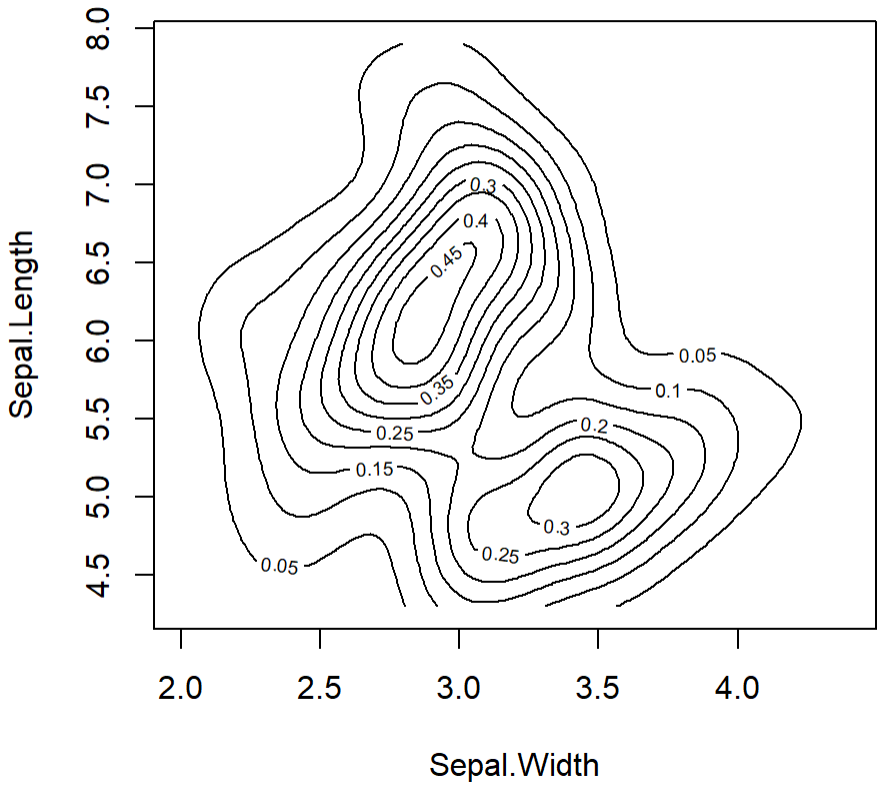
```
1  ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length))+
2    geom_density2d_filled()
```

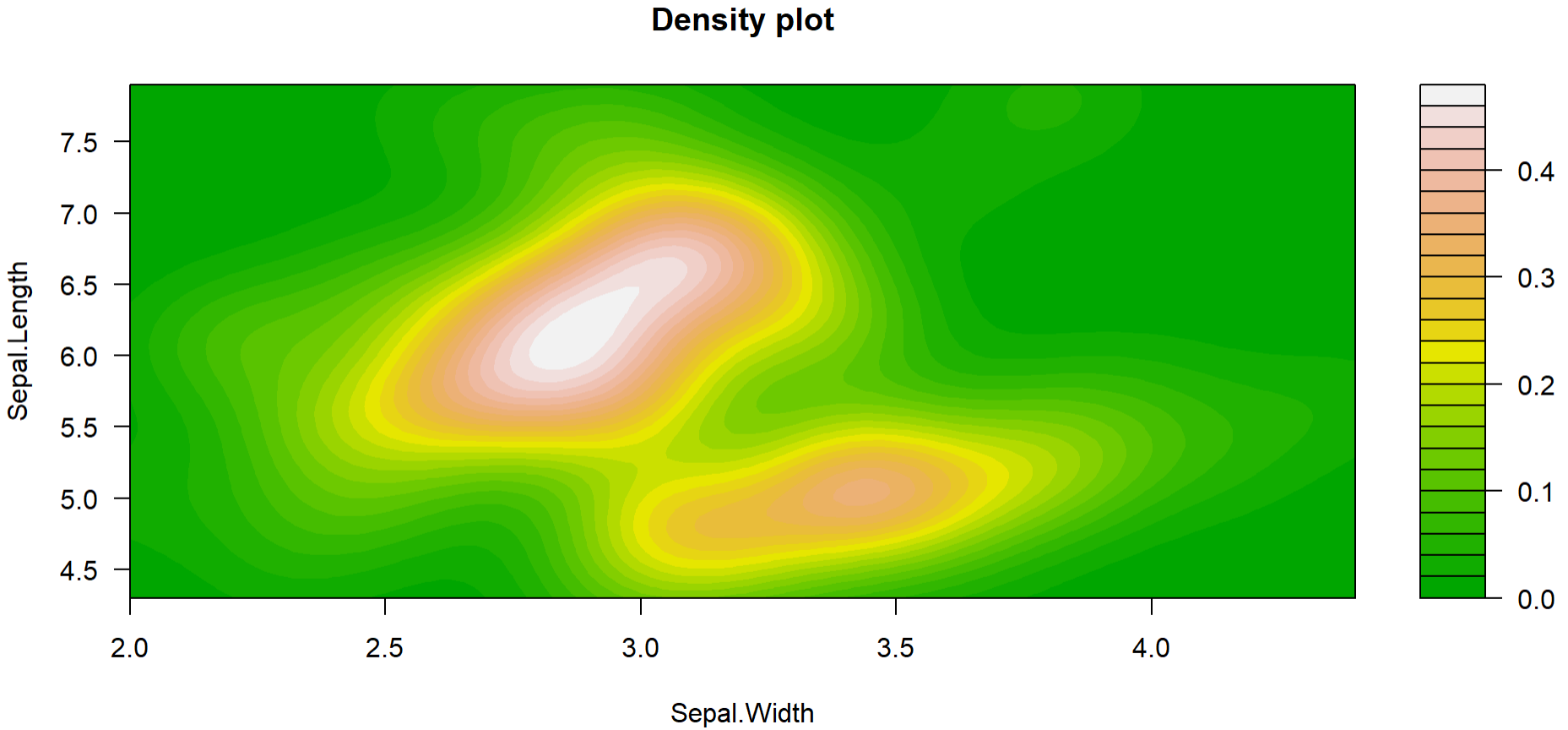


BASE R ALTERNATIVE

Again, the same can be done in base R graphics, but this time it is really more complicated:

```r
x <- iris$Sepal.Width
y <- iris$Sepal.Length

 library(MASS)
 dens <- kde2d(x, y, n = 100)
 x <- iris$Sepal.Width
 y <- iris$Sepal.Length
 library(MASS)
 dens <- kde2d(x, y, n = 100)  # n = Rastergröße

layout(matrix(1:2, nrow = 1))
  contour(dens, xlab = "Sepal.Width", ylab = "Sepal.Length")

  #Or prettier:
 filled.contour(
   dens,
   color.palette = terrain.colors,
   xlab = "Sepal.Width",
   ylab = "Sepal.Length",
   main = "Density plot"
 )
```


Sepal.Length
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
2.0
2.5
3.0
3.5
4.0
Sepal.Width
0.3
0.4
0.45
0.05
0.1
0.35
0.2
0.25
0.15
0.3
0.25
0.05


Density plot
7.5
7.0
6.5
6.0
5.5
5.0
4.5
Sepal.Length
2.0
2.5
3.0
3.5
4.0
Sepal.Width
0.4
0.3
0.2
0.1
0.0

EXERCISES 3 TASKS 3