Data Visualization, continued

September 10, 2021

Contents

Boxplots and violin plots

Boxplots and violin plots are good for comparing numerical distributions for multiple categories. Specifically, boxplots (or box and whisker plots) allow you to compare the **median**, and quartiles of the data sets, while violin plots depict the density of data points.

boxplot() in base R

?boxplot -> formula and x -> is iris wide or long? -> it is mixed

Base R has a built-in function `boxplot()`. Pay attention to the ~ notation in the argument passed to the parameter `formula` **if** you are using `formula`.

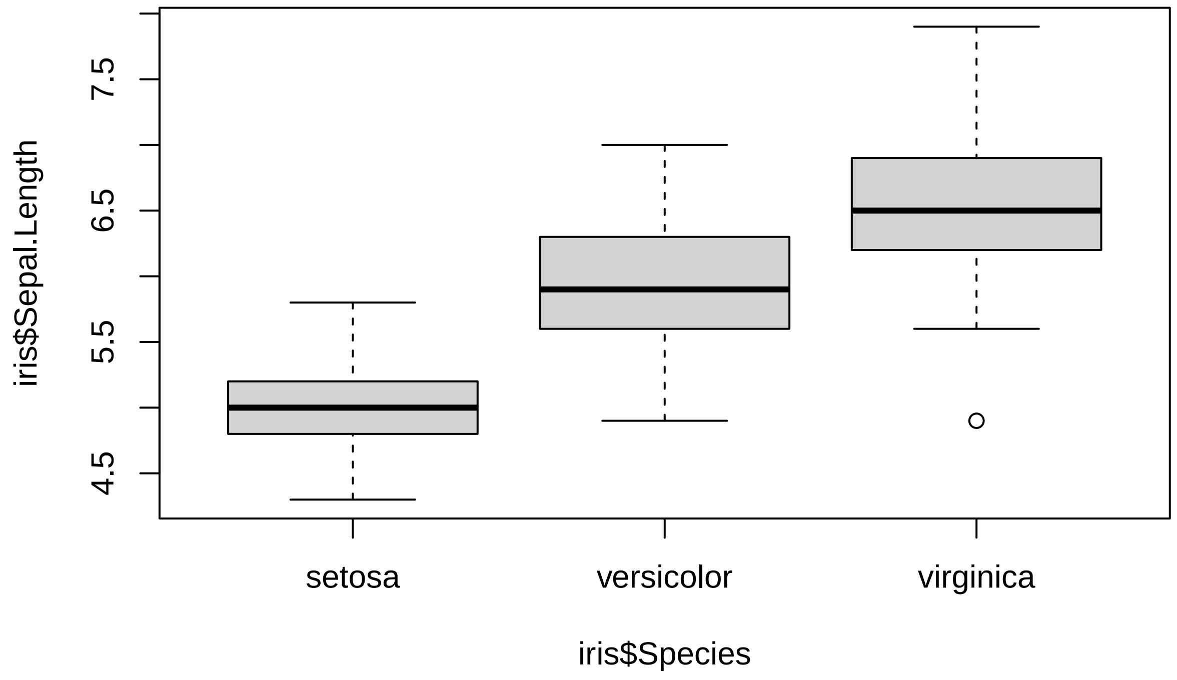
```
# remind yourself what the iris data look like
head(iris)
```

```
##   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
```

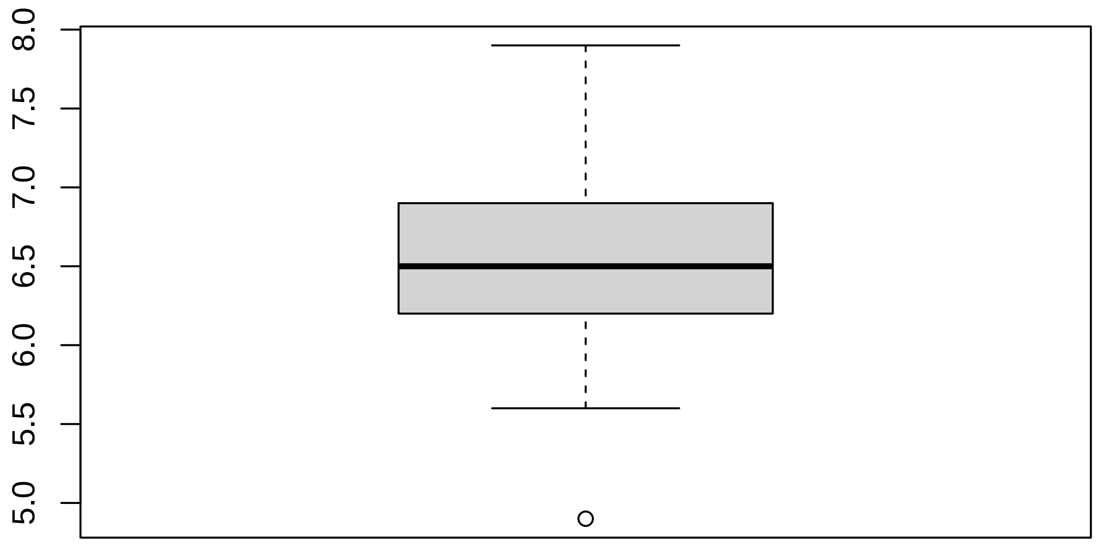
```
tail(iris)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 145          6.7         3.3          5.7         2.5 virginica
## 146          6.7         3.0          5.2         2.3 virginica
## 147          6.3         2.5          5.0         1.9 virginica
## 148          6.5         3.0          5.2         2.0 virginica
## 149          6.2         3.4          5.4         2.3 virginica
## 150          5.9         3.0          5.1         1.8 virginica
```

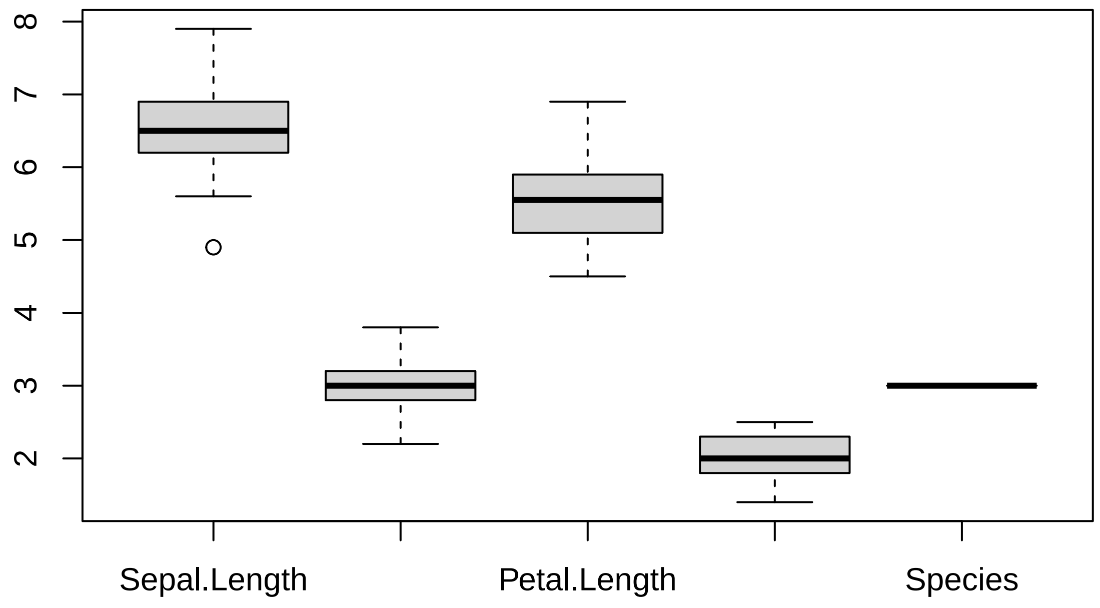
```
# create a boxplot by providing an argument to the "formula" parameter
boxplot(formula = iris$Sepal.Length ~ iris$Species)
```



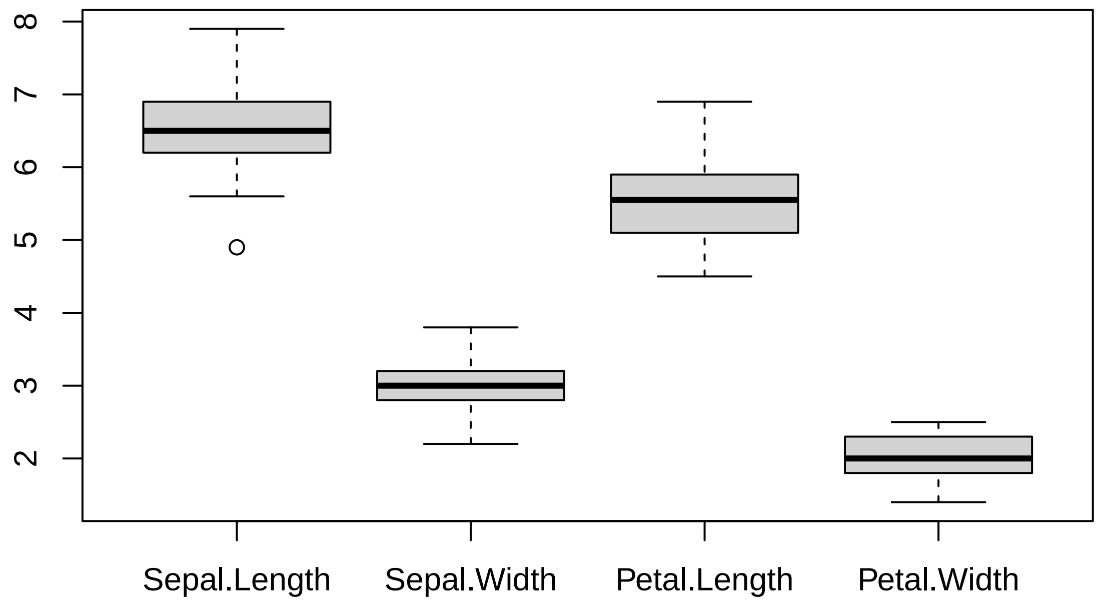
```
# you can also create boxplots by providing a single vector or a matrix with WIDE data to the "x" param

# first, subset iris to only keep the virginica data
virginica_rows = which(iris$Species=="virginica")
iris_virginica = iris[virginica_rows,]

# x = vector
boxplot(x = iris_virginica$Sepal.Length)
```



```
# x = matrix
boxplot(x = iris_virginica)
```



```r
# x = matrix, but do not plot the Species column
boxplot(x = iris_virginica[,1:4])
```

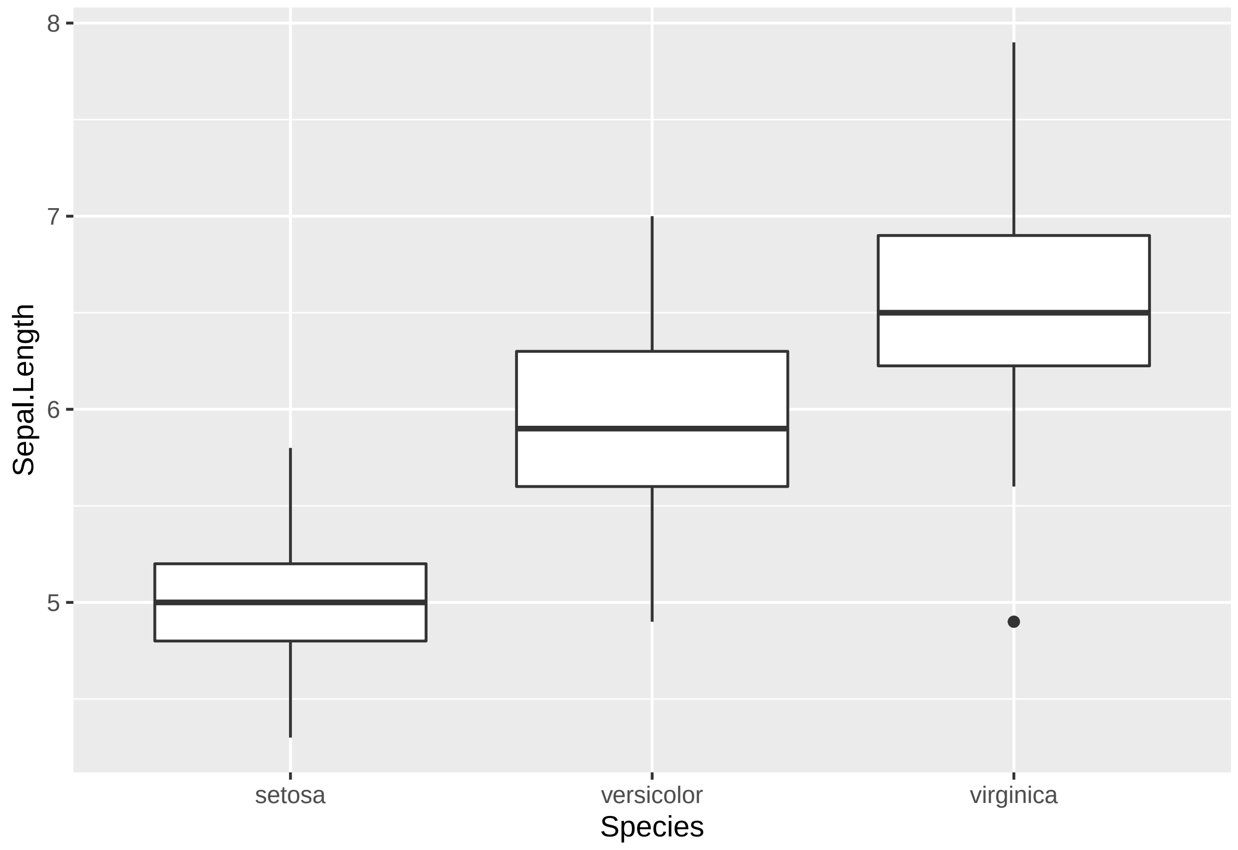


geom_boxplot() and geom_violin() in ggplot2

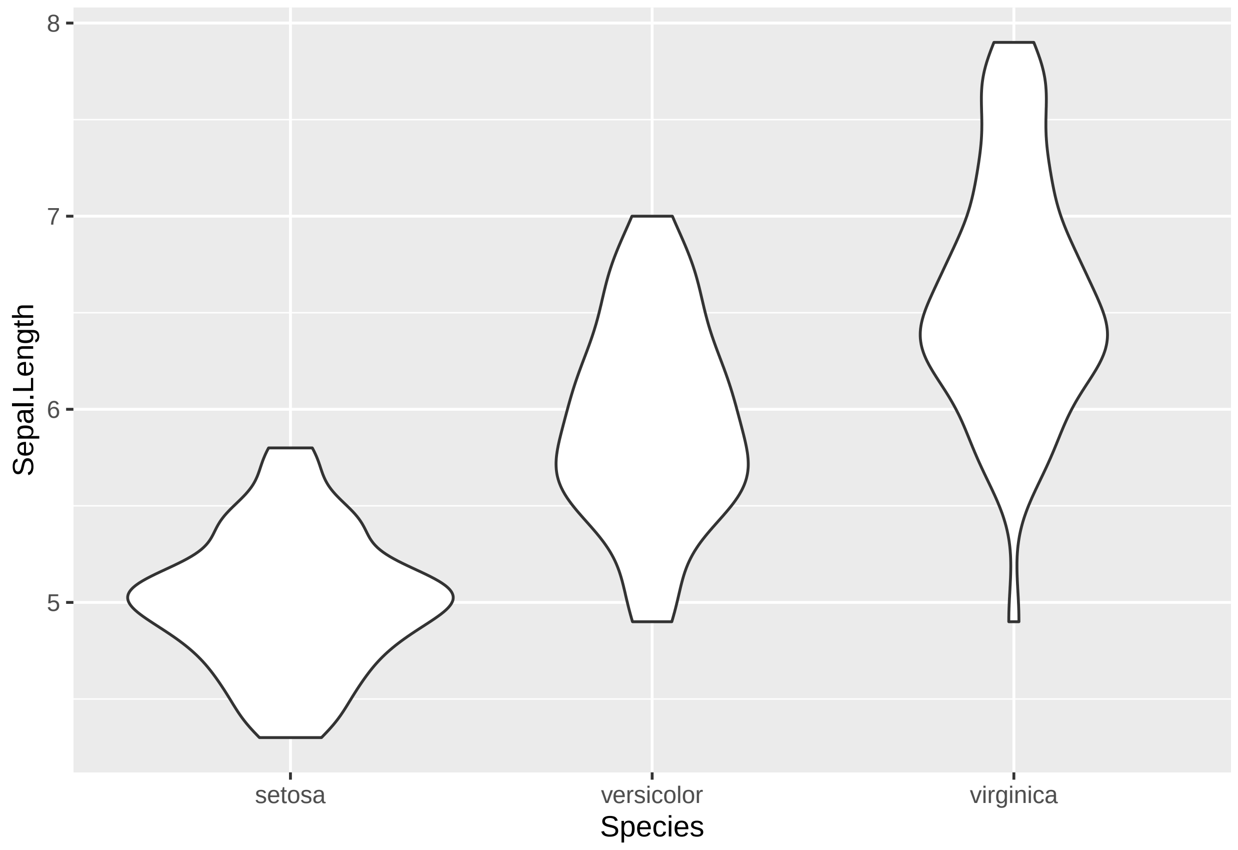
Below we make boxplots and violinplots for the same data.

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

then do +geom_point()
looks neat but deceptive! do dim(iris) -> expect 50 points!
then do +geom_jitter(width=0.1,height=0)

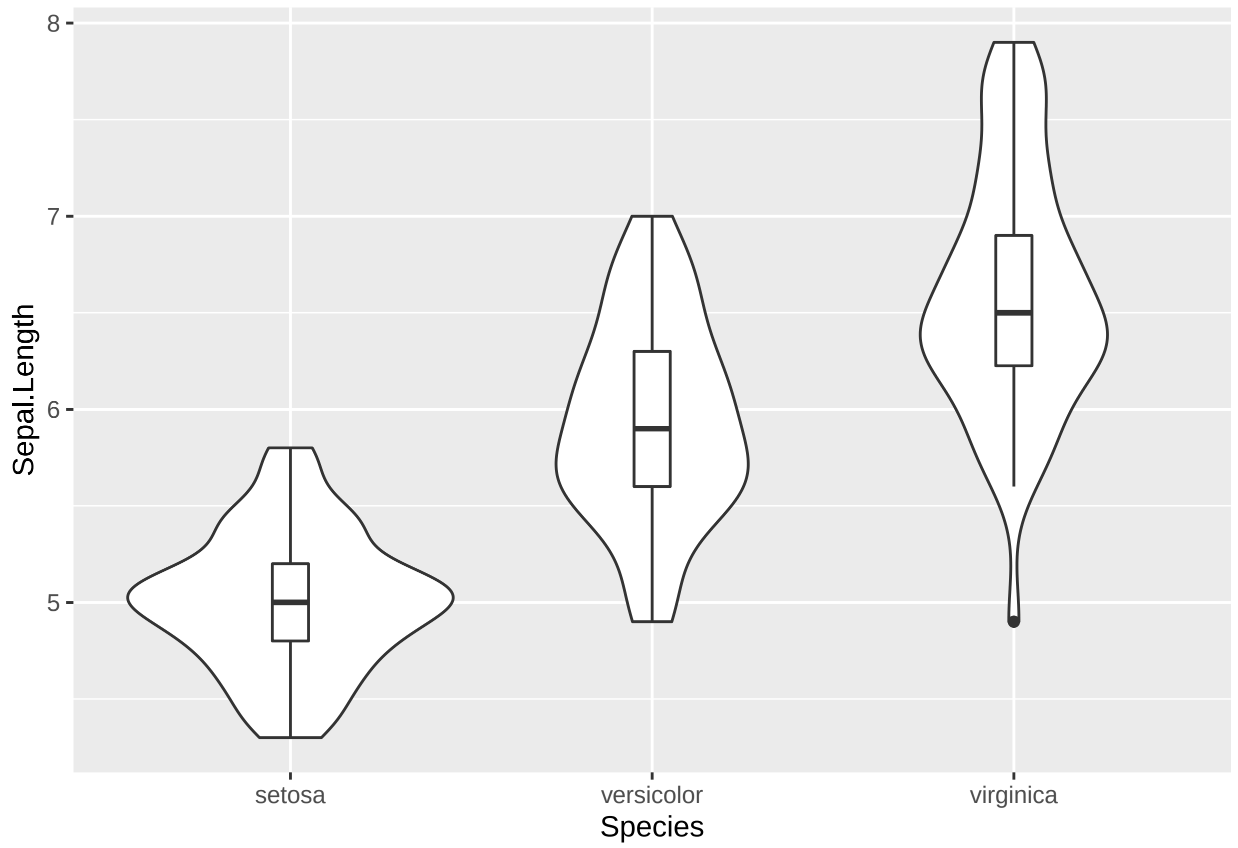


```
# violin plot
ggplot(data=iris,
       mapping=aes(x=Species,
                   y=Sepal.Length)) +     color=Species
  geom_violin()
```



do boxplot+violin with no arguments, then violin+boxplot, then set width, then add color to violin

```
# a combination of both
ggplot(data=iris,
       mapping=aes(x=Species,
                   y=Sepal.Length)) +
  geom_violin() + add color here
  geom_boxplot(width=0.1)
```



discuss when to use boxplot and when to use violin plot

Faceting

Sometimes we want to show numerical data separated by category, or split them according to multiple categories. In other cases we just want to look at several plots side-by-side. For this, we can arrange plot in multiple **panels**.

Matrix Filled by ROW

mfrow() and mfcol() in base R

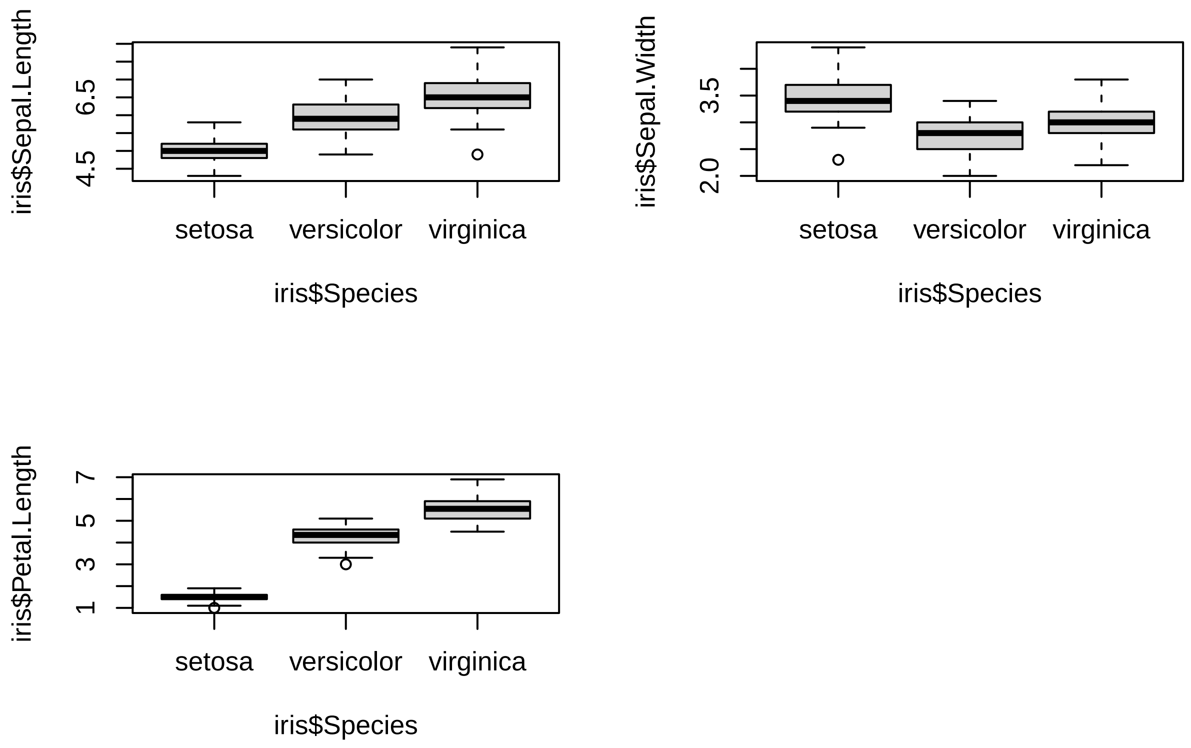
- mfrow – number of plots to be drawn in a window (also mfcol)

The `mfrow()` argument in the `par()` function allows you to define the number of figure panels you want in the device, and how they are to be arranged in terms of rows and columns.

Note that repeated calls to plotting functions will fill up the device according to the number of panels specified by `mfrow()`. If `plot()` is called more times than the number of plots specified, then a new window is opened with the same `mfrow()` options.

```
# display three plots in a 2x2 grid
par(mfrow=c(2,2))

boxplot(iris$Sepal.Length ~ iris$Species)
boxplot(iris$Sepal.Width ~ iris$Species)
boxplot(iris$Petal.Length ~ iris$Species)
```

```r
# reset display to one graph only
par(mfrow=c(1,1))
```



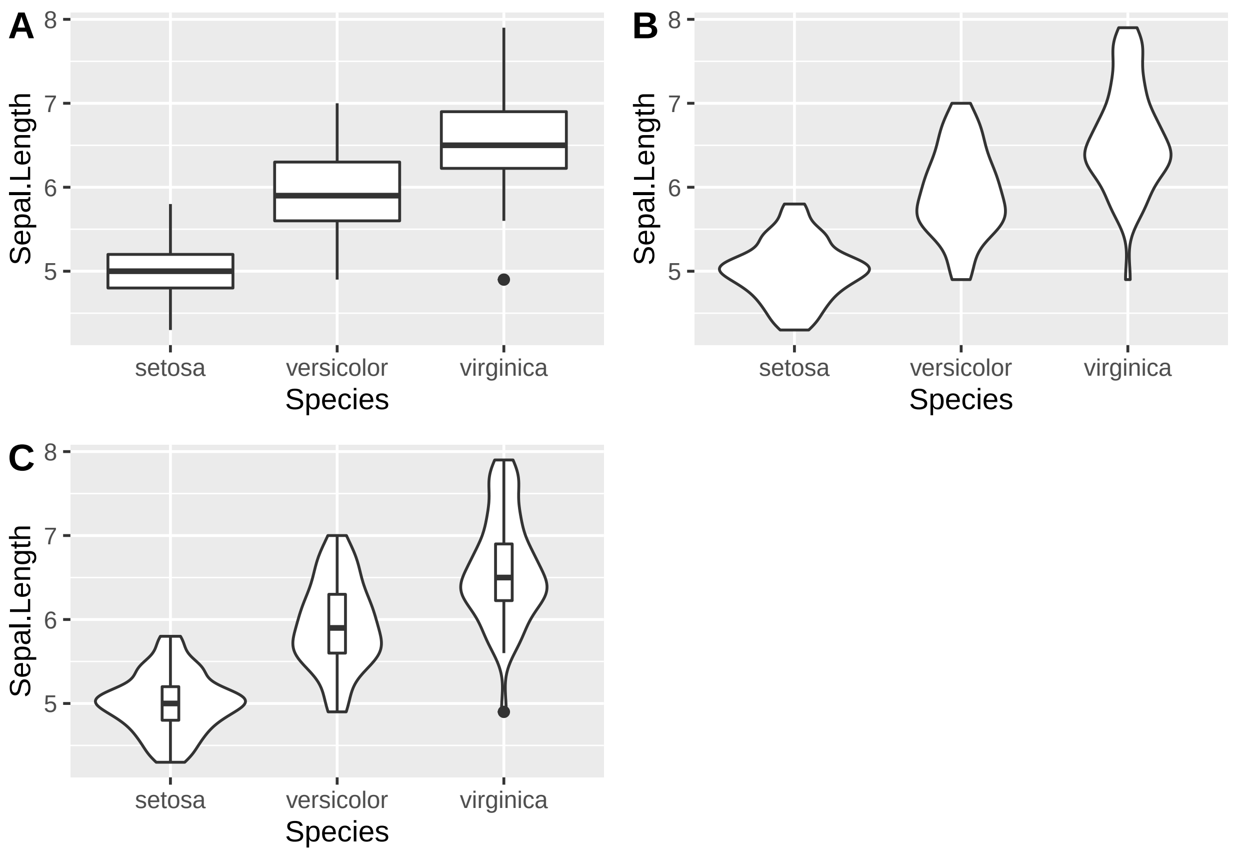
If instead of using R Markdown, you are running the code above in the console, you need to be careful in those situations when not every position is filled (e.g. 3 plots on a 2x2 grid leave one position vacant). To see an example of how things may go wrong, copy the code below and run it in the console:

```
# first, copy and run this
par(mfrow=c(2,2))
boxplot(iris$Sepal.Length ~ iris$Species)
boxplot(iris$Sepal.Width ~ iris$Species)
boxplot(iris$Petal.Length ~ iris$Species)

# then copy and run this
boxplot(iris$Sepal.Length ~ iris$Species)
boxplot(iris$Sepal.Width ~ iris$Species)
boxplot(iris$Petal.Length ~ iris$Species)
```

ggpubr::ggarrange()

To manually combine several ggplot2 plots in a figure, we will use the function `ggarrange()` from the package called **ggpubr** (stands for "GGplot2-based PUBlication Ready plots").

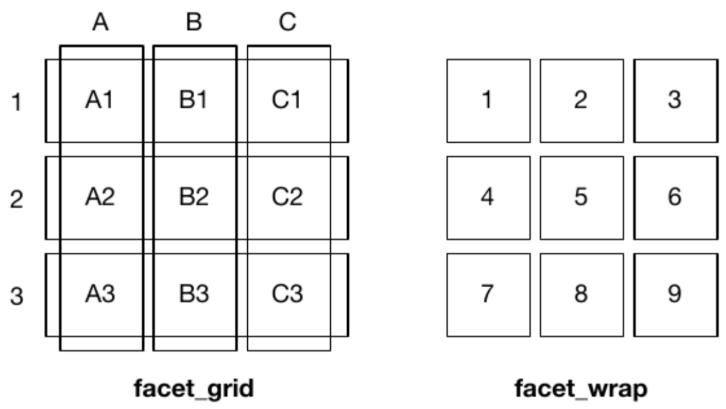
```r
library(ggpubr)

# boxplot
box = ggplot(data=iris,
       mapping=aes(x=Species,
                   y=Sepal.Length)) +
  geom_boxplot()

# violin plot
violin = ggplot(data=iris,
       mapping=aes(x=Species,
                   y=Sepal.Length)) +
  geom_violin()

# box + violin plot
combined = ggplot(data=iris,
       mapping=aes(x=Species,
                   y=Sepal.Length)) +
  geom_violin() +
  geom_boxplot(width=0.1)

# arrange in a 2x2 grid and label panels as A, B, C
ggarrange(box, violin, combined,
          ncol = 2,
          nrow = 2,
          labels = c("A","B","C"))
```



facet_wrap() and facet_grid() in ggplot2

- Facet plots, also called "lattice" or "trellis" plots, are a powerful tool for exploratory data analysis: you can rapidly compare patterns in different parts of the data and see whether they are the same or different. There are three types of faceting:
- facet_null() - a single plot (default)
- facet_wrap() - "wraps" a 1D ribbon of panels into 2D (useful if you have a large number of categories)
- facet_grid() - produces a 2D grid of panels defined by different variables, which form the rows and columns

This figure from the book illustrates the differences between wrapping and making a grid:



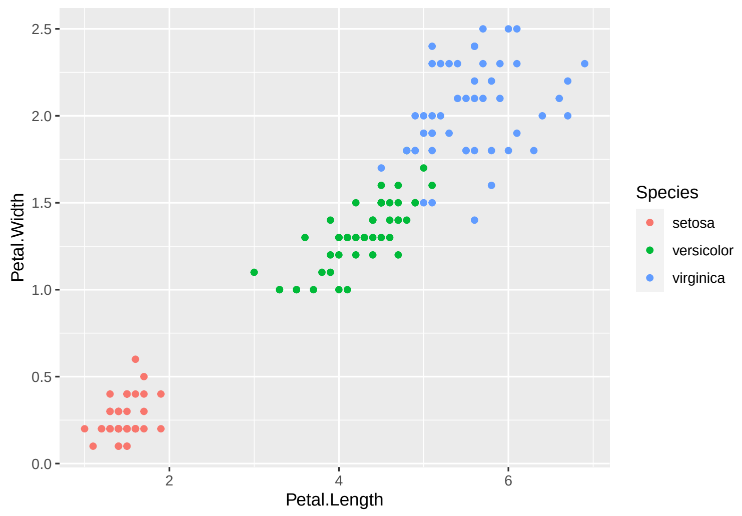
Both `facet_wrap()` and `facet_grid()` can be specified with one or two variables defined by a **formula** (specified with a tilde symbol, `~`). The difference is in the specific **syntax**.

The output will be similar for one variable, but for two variables the axis labels will differ, and and `facet_grid()` will produce a more sensible representation of the data.

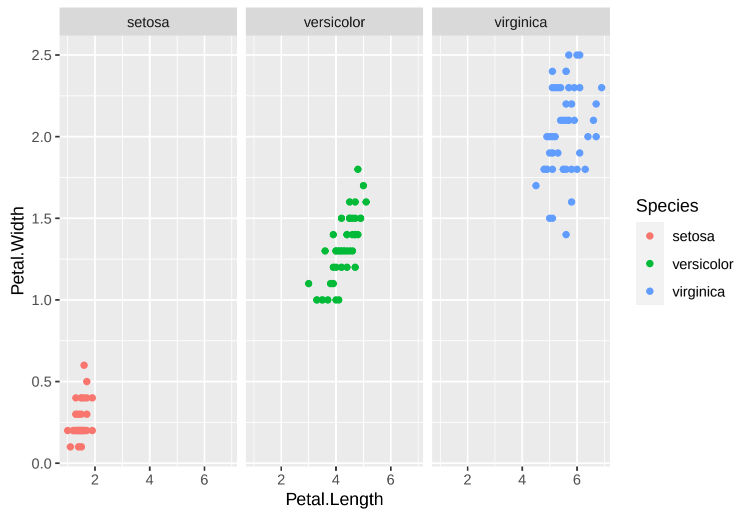
- Syntax for `facet_wrap()`:
  - `(~ a)` - spreads the values of ***a*** across panels facilitates comparisons of ***y*** position, because the vertical scales are aligned.
  - `(~ a + b)` - spreads the combinations of values for both ***a***and ***b***

For the **iris** dataset:

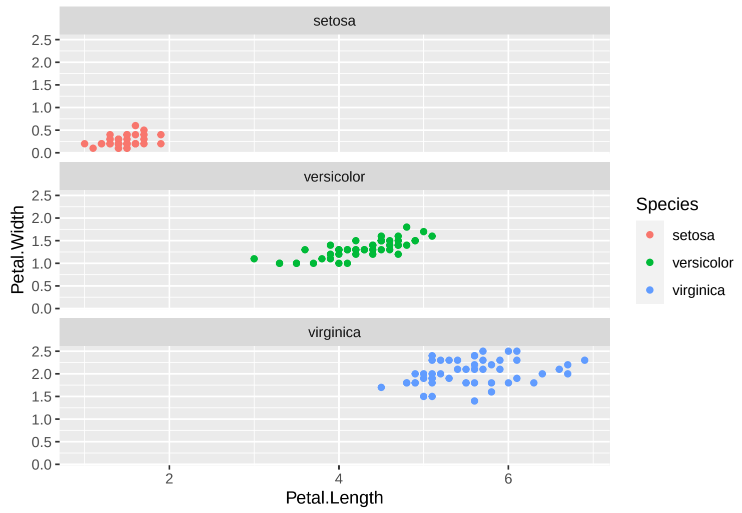
```
base_plot = ggplot(data=iris,
                   mapping=aes(x=Petal.Length,
                               y=Petal.Width,
                               color=Species)) +
  geom_point()

base_plot # display
```



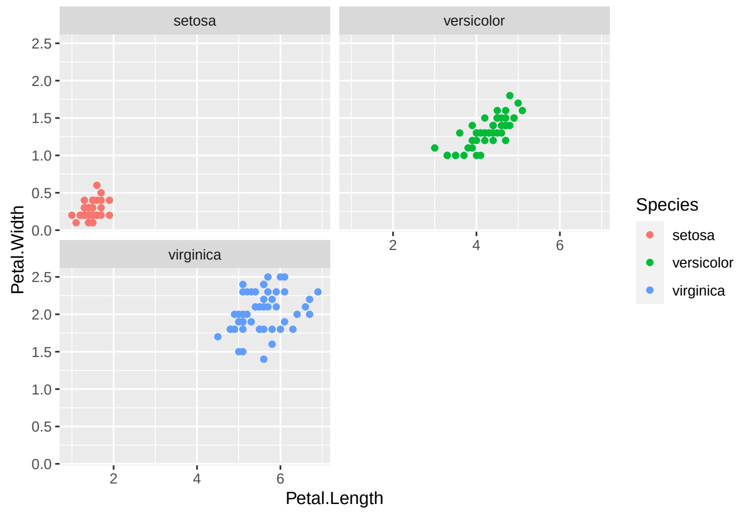
```r
# wrap by Species
base_plot + facet_wrap(~Species)
```



```r
# the same but arrange vertically
base_plot + facet_wrap(~Species,
                       ncol=1)
```



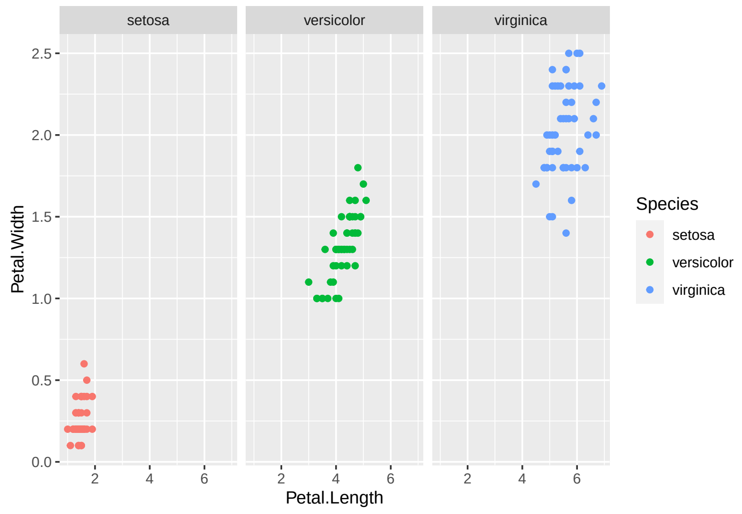
```r
# the same but arrang in a 2x2 grid
base_plot + facet_wrap(~Species,
                       ncol=2,
                       nrow=2)
```



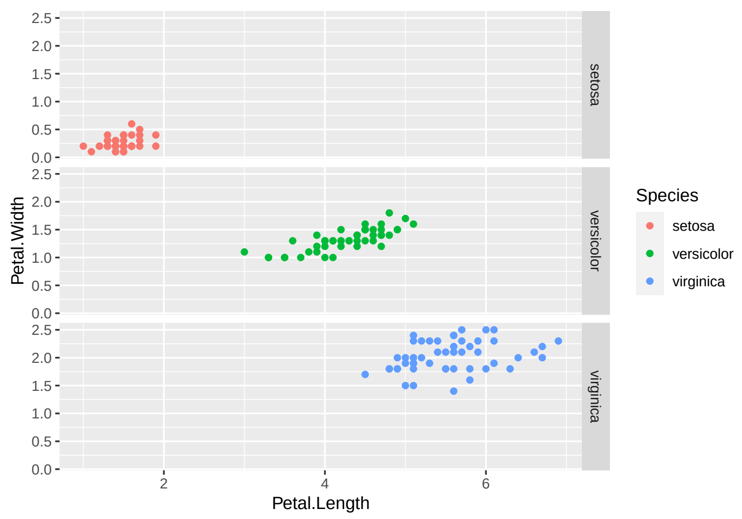
- Syntax for `facet_grid()`:
  - `(. ~ a)` - spreads the values of ***a*** across the columns. This direction facilitates comparisons of ***y*** position, because the vertical scales are aligned.
  - `(b ~ .)` - spreads the values of ***b*** down the rows. This direction facilitates comparison of ***x*** position because the horizontal scales are aligned. This makes it particularly useful for comparing distributions.
  - `(a ~ b)` - spreads ***a*** across columns and ***b*** down rows. You'll usually want to put the variable with the greatest number of levels in the columns, to take advantage of the aspect ratio of your screen.

We can easily split our plots by **Species** to produce the same output as `facet_wrap()`:

```
# by columns
base_plot + facet_grid(. ~ Species)
```



```
# by rows
base_plot + facet_grid(Species ~ .)
```

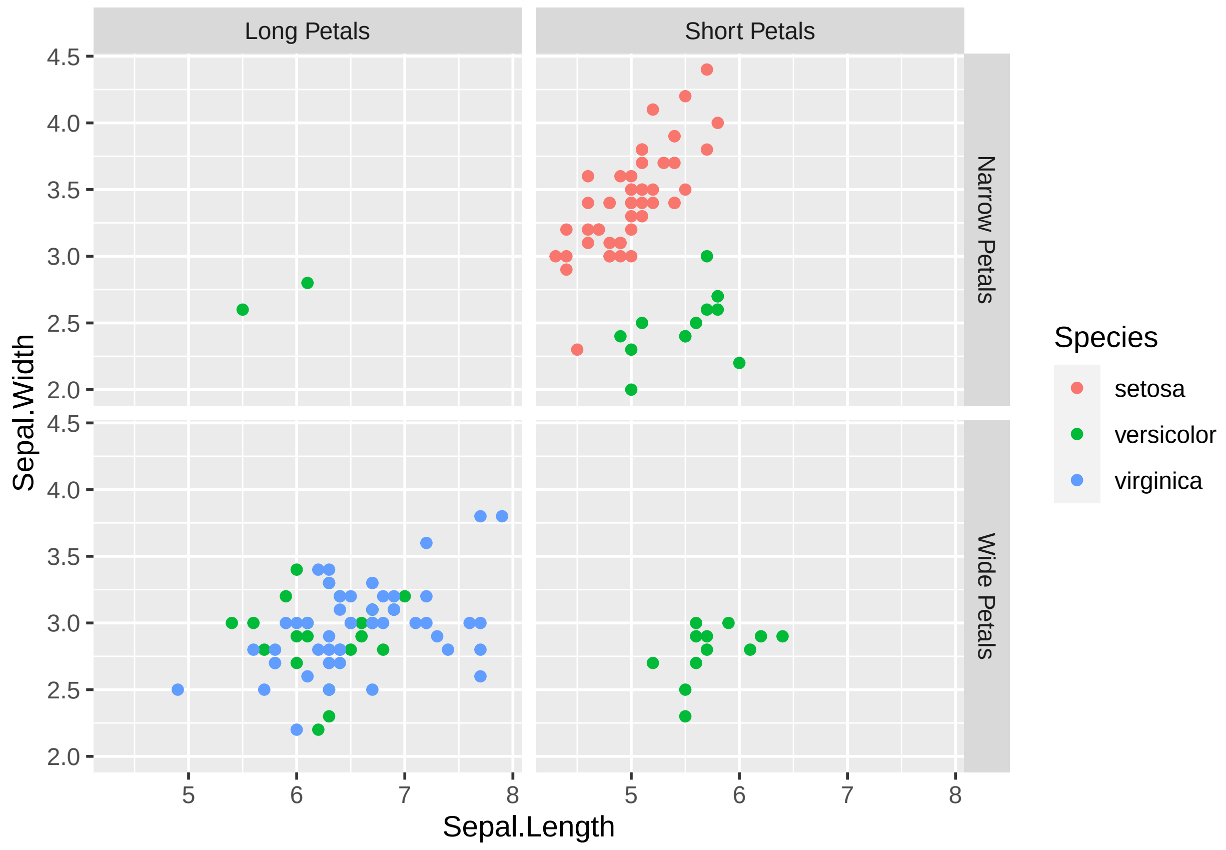


However, since the **iris** dataset contains only **one categorical variable**, we would have to do something fancier to get good mileage out of `facet_grid()` for this example.

We can do this by *splitting some of the quantitative data into categories* (this is kind of a kluge)[1]. Below we plot Sepal Length against Sepal Width (quantitative variables) split out by Petal type (Long vs. Short, Narrow vs. Wide):

```
iris_categories = iris # make a copy

# add a new column called Petal.Width.Categories
# where all petals narrower than 1.3 cm are "Narrow Petals" and the rest are "Wide Petals"
iris_categories$Petal.Width.Categories = factor(ifelse(iris_categories$Petal.Width<1.3,
                                                  "Narrow Petals",
                                                  "Wide Petals"))

# add a new column called Petal.Length.Categories
# where all petals shorter than 4.35 cm are "Short Petals" and the rest are "Long Petals"
iris_categories$Petal.Length.Categories = factor(ifelse(iris_categories$Petal.Length<4.35,
                                                        "Short Petals",
                                                        "Long Petals"))

# plot and wrap
ggplot(data=iris_categories,
       mapping=aes(x=Sepal.Length,
                   y=Sepal.Width,
                   color=Species)) +
  geom_point() +
  facet_grid(Petal.Width.Categories ~ Petal.Length.Categories)
```

[1]https://stackoverflow.com/questions/40350230/variable-hline-in-ggplot-with-facet



You should consult the ggplot2 book chapter for information on customization, such as controlling axis **scales**, or cutting continuous variables into bins in order to facet them.