Module 5 – Boxplots

CSCI E-5a: Programming in R

Let's clear the global computing environment:

```r
rm( list = ls() )
```

Now we'll load in the objects for the module:

```r
load( "Module 5 R Objects.Rdata" )

ls()
```

```
## [1] "example.1.data" "example.2.data"
```

Module Preview

Hello! And welcome to Module 5: Boxplots.

In this module, we will learn a new visualization method called the *boxplot* (not to be confused with the barplot!).

- In Section 1, we'll define boxplots, and see how to construct one.
- In Section 2, we'll learn how to modify the basic plot.

Once you've completed this module, you'll be able to:

- Explain the structure of a boxplot.
- Construct a basic boxplot.
- Modify the basic display of the boxplot.

There is one new built-in R function in this module:

- `boxplot()`

All right! Let's get started by learning how to visualize a set of numeric values by using a boxplot.

Section 1: Boxplot Fundamentals

Main Idea: *We can visualize a set of numeric values by using a barplot*

In this section, we'll define boxplots, and see how to construct one.

So far, we've seen two methods for visualizing the values in a one-dimensional numerical range (i.e. a vector):

- We can use a stripchart to display each individual value.
- We can use a histogram to display the distribution of values across a set of subintervals.

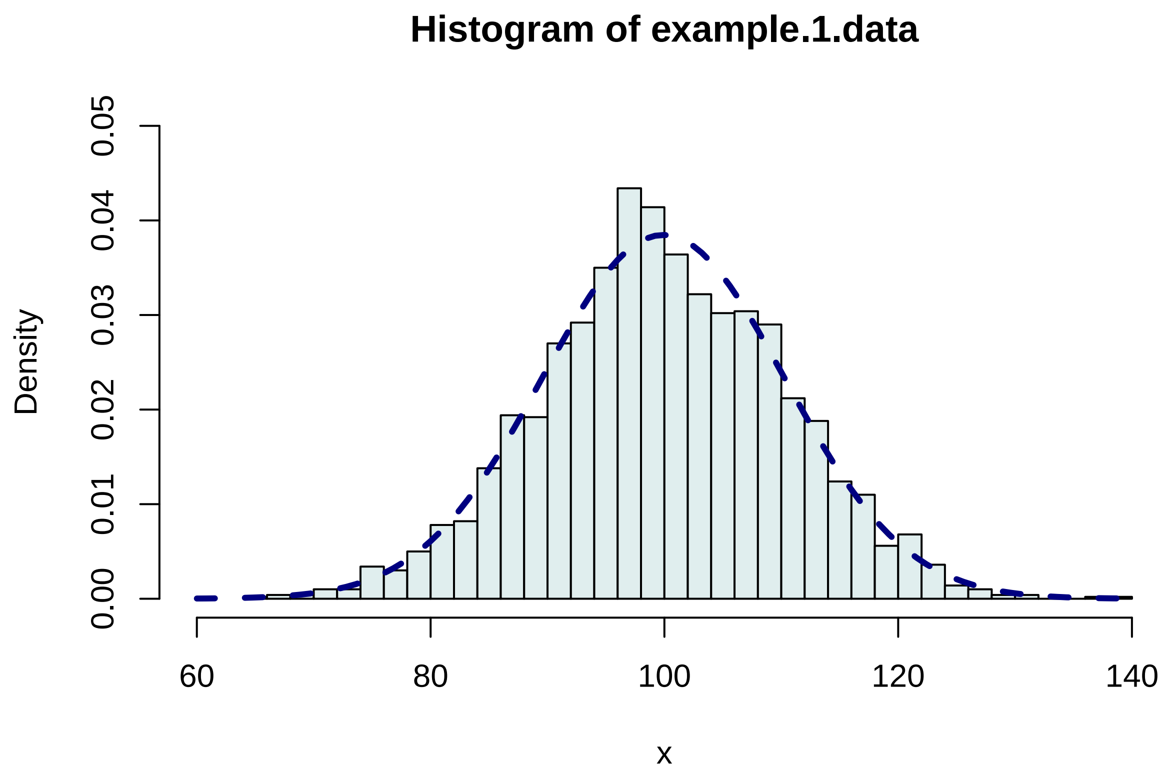
The stripchart is great because it provides a high degree of granularity: we can literally see each individual data value.

Unfortunately, for large datasets this doesn't work so well, because there are so many individual points that the graph loses any resolution.

For instance, the R object `example.1.data` consists of 2,500 values from a normal distribution.

Let's make a histogram of the values in the vector `example.1.data`, and superimpose the best-fitting normal density curve:

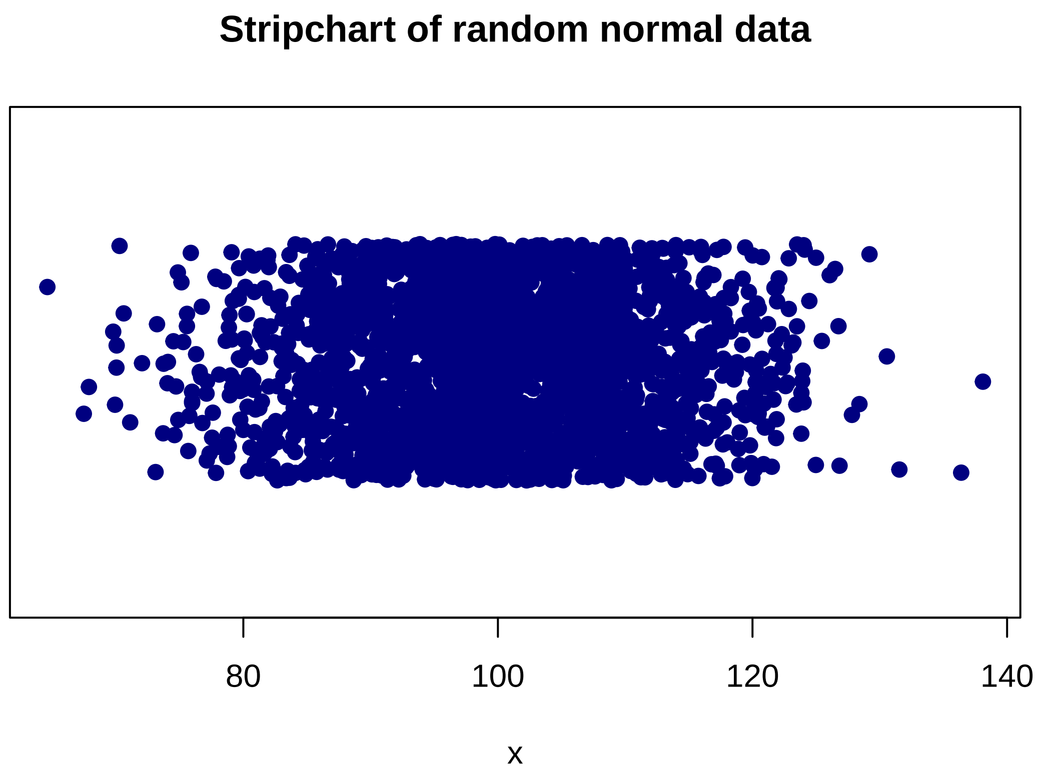
```r
hist(
  x = example.1.data,
  xlim = c(60, 140),
  ylim = c(0, 0.05),
  main = "Histogram of example.1.data",
  xlab = "x",
  ylab = "Density",
  prob = TRUE,
  breaks = 50,
  col = "azure2"
)

sample.mean <- mean( example.1.data )

sample.sd <- sd( example.1.data )

curve(
  dnorm(x, mean = sample.mean, sd = sample.sd ),
  lty = "dashed",
  lwd = 3,
  col = "navy",
  add = TRUE
)
```



So this data is normally distributed, with the characteristic symmetrical “mountain” shape.

Now let’s form a stripchart of this data:

```r
stripchart(
    x = example.1.data,
    ylim = c(0, 2),
    main = "Stripchart of random normal data",
    xlab = "x",
    ylab = "",
    method = "jitter",
    jitter = 0.5,
    pch = 19,
    cex = 1,
    col = "navy"
)
```



Notice that with the stripchart we are starting to lose resolution in the densely packed regions, because there are so many points.

Stripcharts work very well with hundreds of data points, but as you can see once we have a few thousand data points then this approach starts to break down.

Thus, stripcharts don't really scale well for very large data sets.

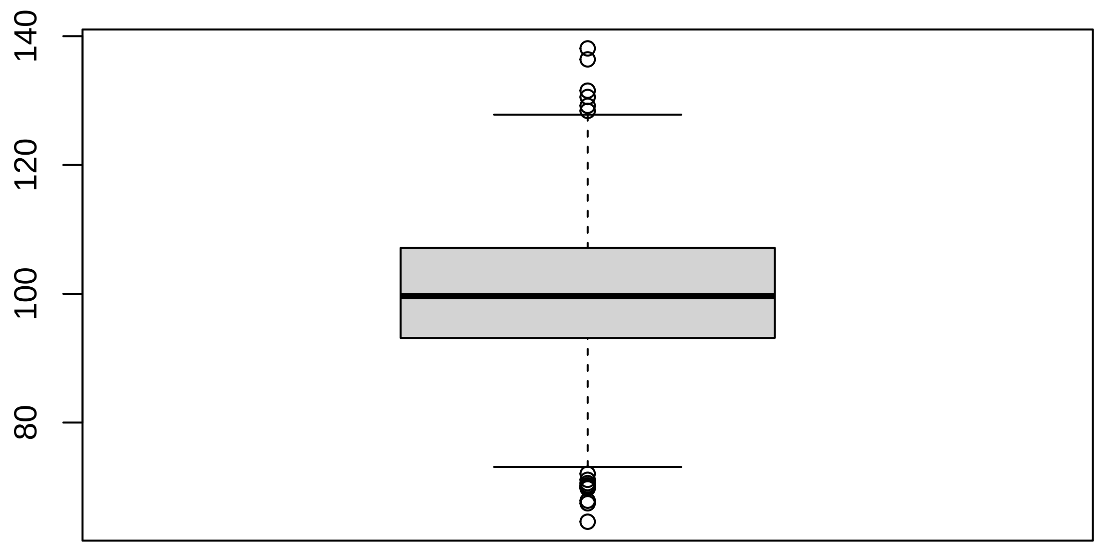
On the other hand, sometimes we don't need to visualize *every* value in a vector.

Instead, we want a more abstract representation of the range of values.

Then a *boxplot*, also known as a "box-and-whiskers plot", can be very useful.

We can draw a basic boxplot by calling the `boxplot()` function with our data:

```
boxplot(
    x = example.1.data
)
```



The structure of the boxplot is a little complicated:

- The lower edge of the box is the 25th percentile, which is the point at which 25% of the data are below.
- The thick central band in the middle of the box is the median, so that 50% of the observations in this dataset are below this line.
- The top of the box is the 75th percentile, so that 75% of the values in the vector are below this line.
- The difference between the 75th and 25th percentiles is called the "inter-quartile range", and provides a measure of the central range of the data.
- The whiskers then extend out 1.5 times the inter-quartile range or to the most extreme point, whichever is less.
- Finally, any points that are beyond 1.5 times the inter-quartile range are then graphed individually.

Thus, a boxplot enables us to quickly see the central location of the data (that's the box), the more dispersed ranges of the data (that's the whiskers), and finally any outliers (those are the points that lie outside the whiskers and are graphed explicitly.)

Since barplots do not display every value in the vector they scale nicely for larger data sets.

Boxplots are a popular way of visualizing a one-dimensional range of numeric values.

In the year 2021, all the cool kids are into boxplots, and they are very fashionable.

The advantage of boxplots is that they are an ingenious and sophisticated way to display a lot of high-level information about the range of numeric values, so they present a more abstract and high-level representation of the information.

The disadvantage of boxplots is that they are an ingenious and sophisticated way to display a lot of high-level information about the range of numeric values, so it can be difficult for normal people to understand what's going on.

You might have noticed that it was complicated to explain what the components of the boxplot are, and if you don't use this on a regular basis you'll quickly forget the details.

We now have 3 different methods for visualizing a range of numeric values:

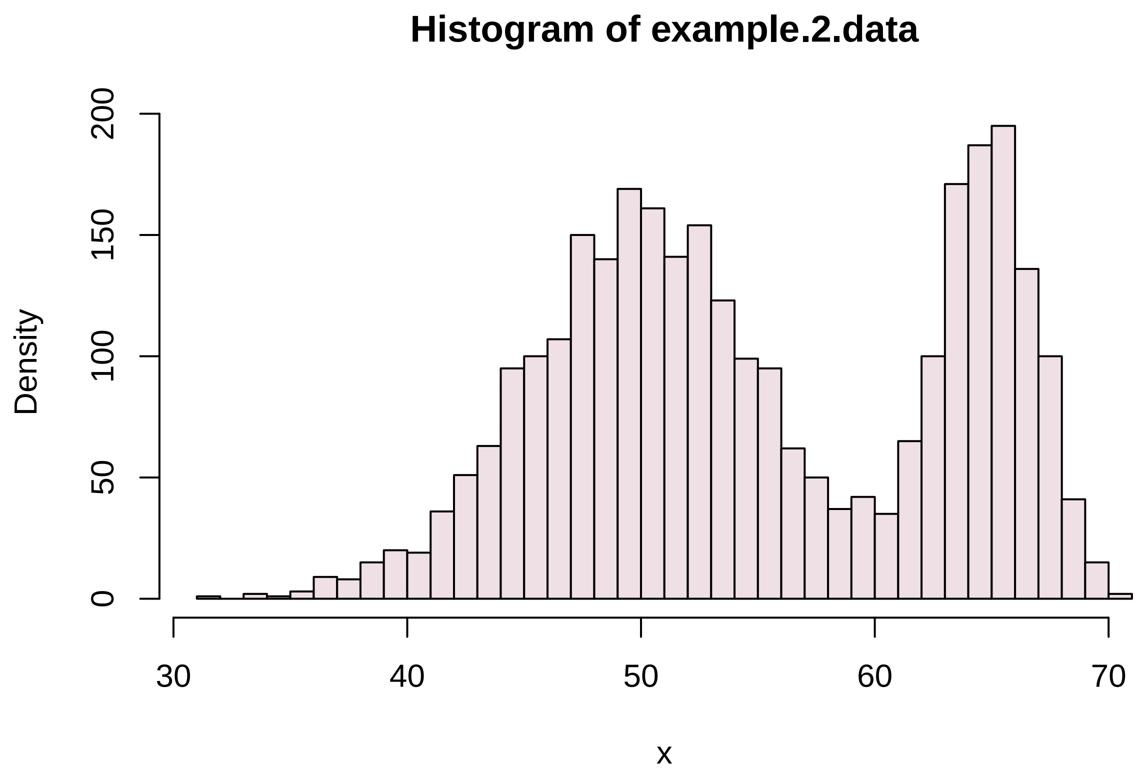
- The stripchart is the most granular, and displays every point of the range, but this doesn't work very well once have more than a few thousand points.
- The histogram can represent the general shape of the distribution without needing to show every point, and scales well for even very large datasets.
- The boxplot is the most abstract representation, and provides a high-level description of the data.

Boxplots work well with normally distributed data, and in general with data that has a more-or-less regular distribution.

However, if your data has irregular features, then the boxplot won't do a good job of displaying these features.

For instance, let's construct a histogram for the data in `example.2.data`:

```
hist(
  x = example.2.data,
  main = "Histogram of example.2.data",
  xlab = "x",
  ylab = "Density",
  col = "lavenderblush2",
  breaks = 50
)
```

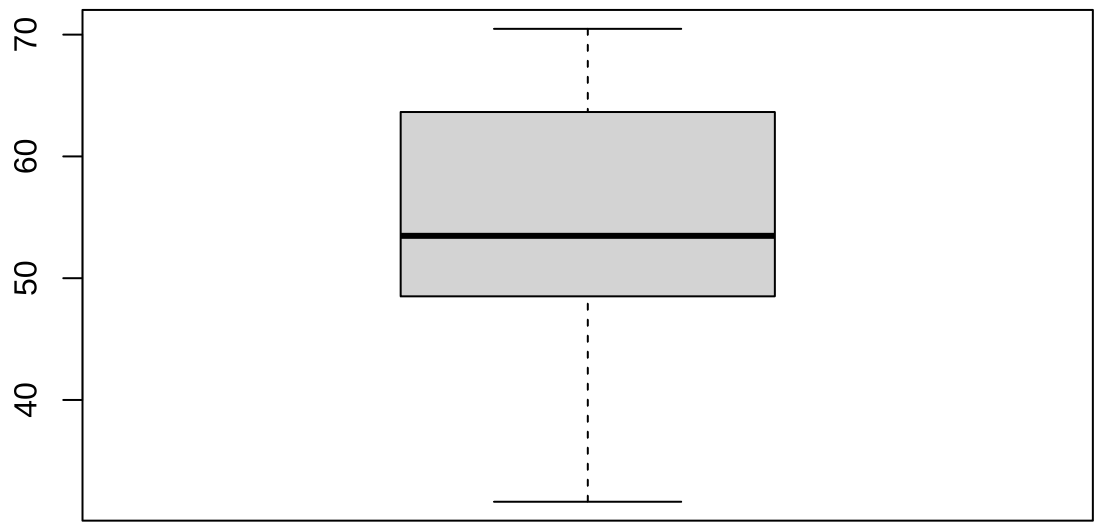


This data is *bimodal*, which means that it has two hump shapes.

Bimodal data can arise in practice when you are sampling from a mixture of two different populations.

If we construct a boxplot of this data, it won't be able to show this feature of the distribution:

```
boxplot(
  x = example.2.data
)
```



Thus, the abstract nature of the boxplot representation is both an advantage and a disadvantage:

- The advantage is that it enables us to quickly get a sense of the range of the values in the data by removing a lot of detail.
- The disadvantage is that we lose a lot of valuable information about the distribution by removing that detail.

Personally, I'm not a huge fan of boxplots, and I prefer stripcharts and histograms.

I like stripcharts because they enable me to visualize every single value in the data.

I like histograms because they allow me to visualize the shape of the distribution without seeing every individual point.

As we've seen, stripcharts don't scale well, precisely because of their granularity, but histograms scale extremely well, so between the two approaches you'll usually be able to get a good sense of your data.

For me, boxplots are too abstract, and while they do a good job of displaying the overall range of values they leave too much out.

But this is a personal thing, and as I've observed many people are enthusiastic about boxplots.

So that's what a boxplot is, and how to construct a basic boxplot.

Now let's see how to modify this display.

Section 2: Modifying the boxplot display

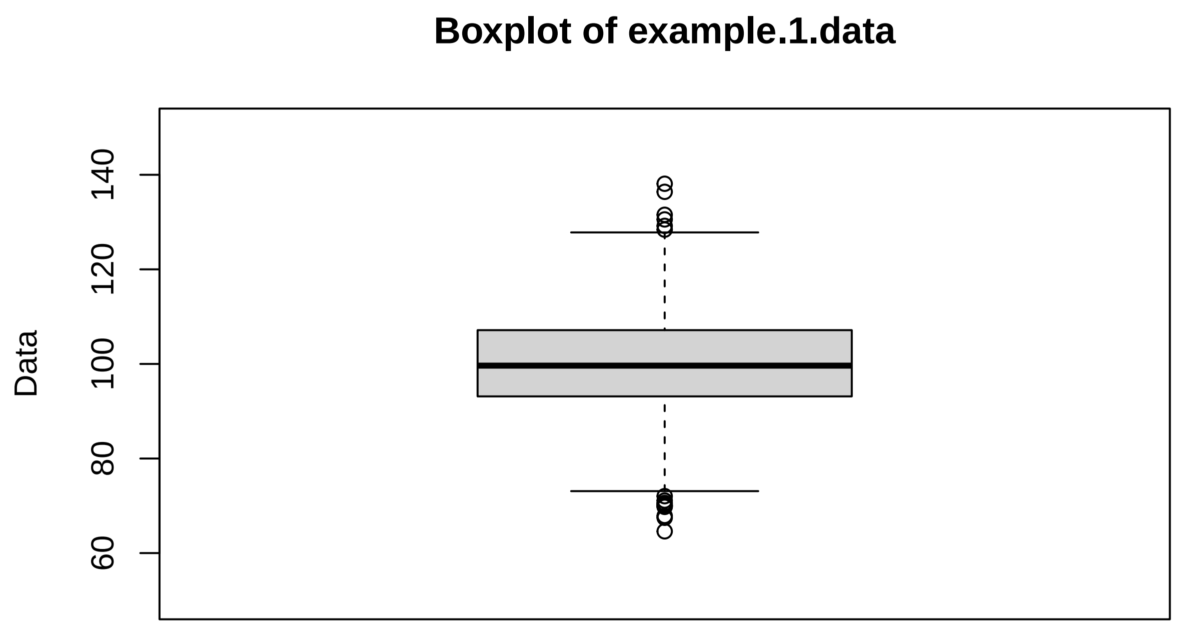
Main Idea: *We can modify the display of a boxplot*

In this section, we'll learn how to modify the basic boxplot.

You might have noticed that we didn't need a pre-existing plotting region to draw a boxplot, and in fact the function can create its own plotting region.

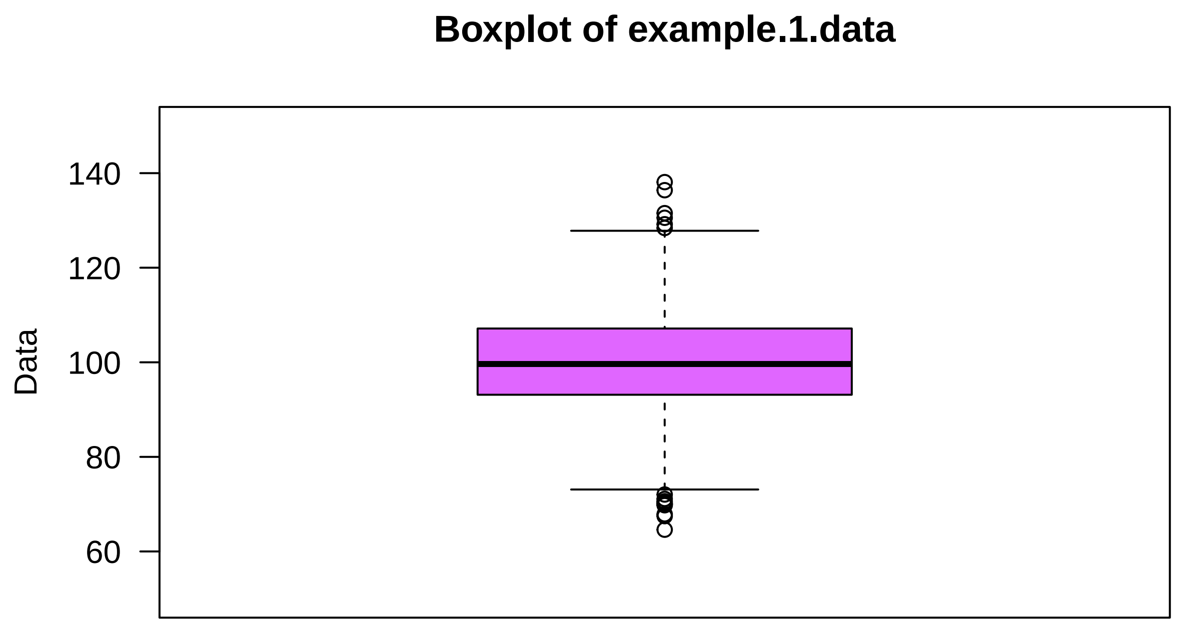
Thus, we can use all the usual plotting options:

```
boxplot(
  x = example.1.data,
  ylim = c(50, 150),
  main = "Boxplot of example.1.data",
  ylab = "Data"
)
```



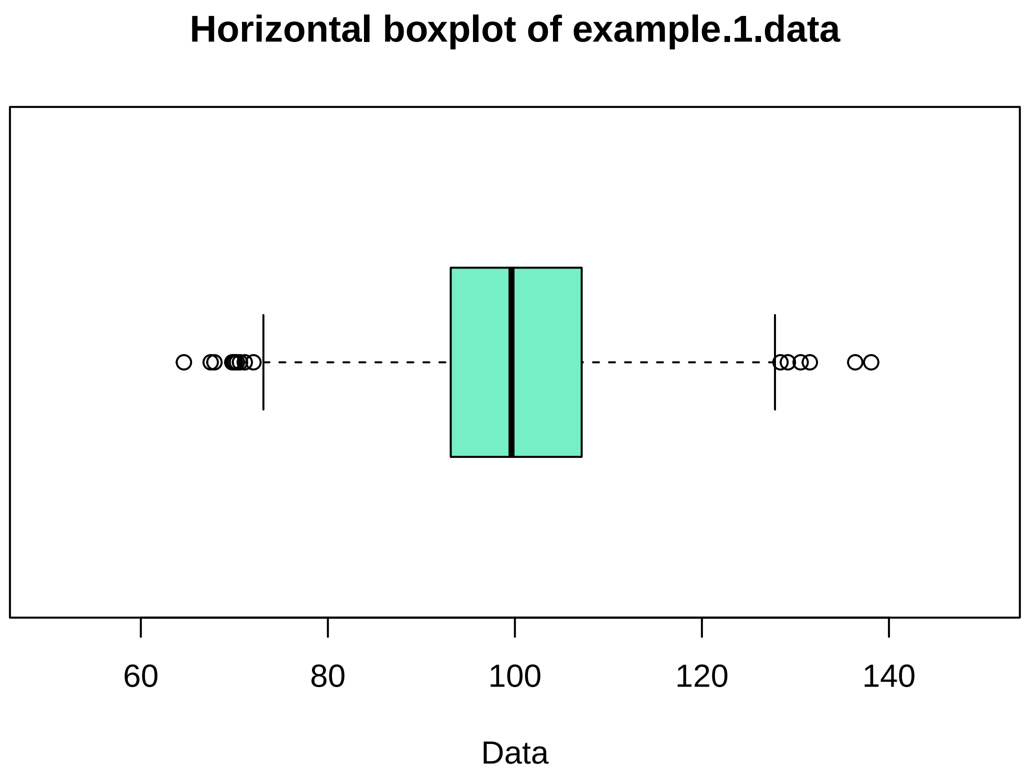
We can change the color of the box using the `col` option:

```
boxplot(
  x = example.1.data,
  ylim = c(50, 150),
  main = "Boxplot of example.1.data",
  ylab = "Data",
  las = 1,
  col = "mediumorchid1"
)
```



We can display the boxplot in a horizontal orientation by setting the `horizontal` option to TRUE (remember to then label the correct axis!):

```
boxplot(
    x = example.1.data,
    ylim = c(50, 150),
    main = "Horizontal boxplot of example.1.data",
    xlab = "Data",
    horizontal = TRUE,
    col = "aquamarine2"
)
```



In Lecture 9, we'll see how to create *stratified* boxplots, which will enable us to simulataneously compare multiple distributions.

So that's how to modify the display of a boxplot.

Now let's review what we've learned in this module.

Exercise 1: The `rivers` dataset

Construct a horizontal boxplot for the values in the `rivers` dataset. Be sure to include a main title and any appropriate axis titles, and also to explicitly specify the color of the box.

Solution

Module Review

In this module, we learned a new visualization method called the *boxplot* (not to be confused with the barplot!).

- In Section 1, we defined boxplots, and saw how to construct one.
- In Section 2, we learned how to modify the basic plot.

Now that you've completed this module, you should be able to:

- Explain the structure of a boxplot.
- Construct a basic boxplot.
- Modify the basic display of the boxplot.

There was one new built-in R function in this module:

- `boxplot()`

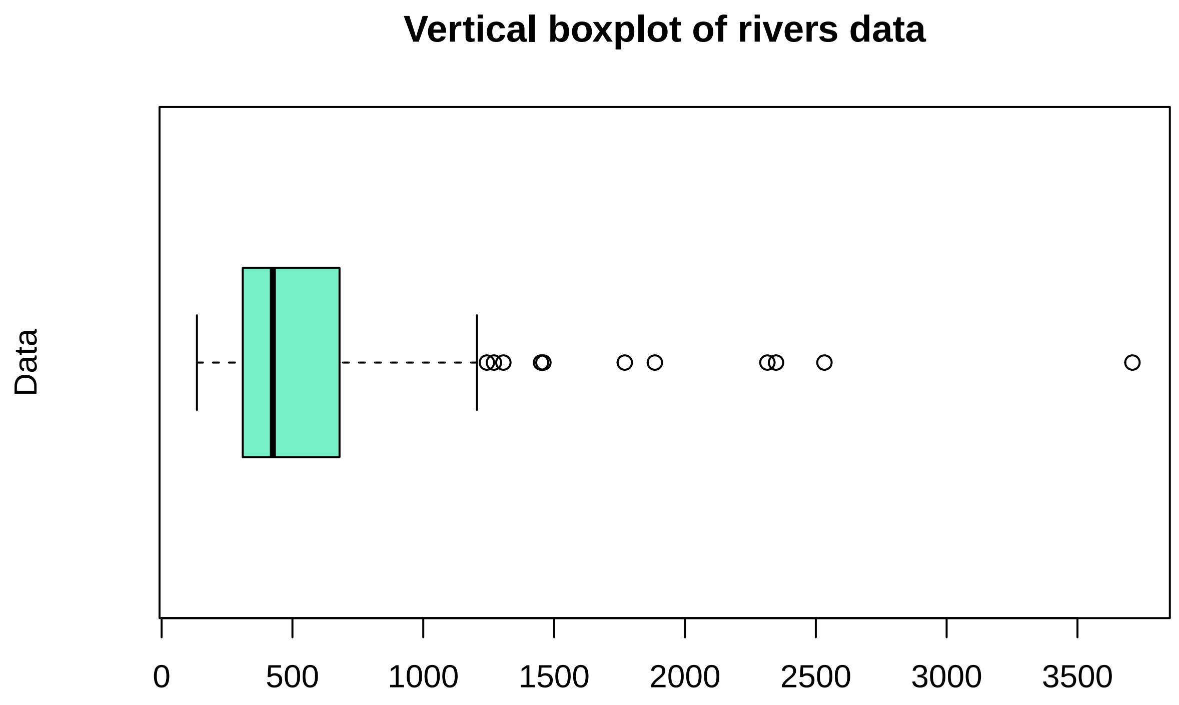
Solution to the Exercise

Exercise 1: The `rivers` dataset

Construct a horizontal boxplot for the values in the `rivers` dataset. Be sure to include a main title and any appropriate axis titles, and also to explicitly specify the color of the box.

Solution

```
boxplot(
    x = rivers,
    main = "Vertical boxplot of rivers data",
    ylab = "Data",
    horizontal = TRUE,
    las = 1,
    col = "aquamarine2"
)
```


Vertical boxplot of rivers data
Data
0
500
1000
1500
2000
2500
3000
3500