Week 4 Module 5: Moving Averages

CSCI E-5a: Programming in R

Let's clear the global computing environment:

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

Module Overview and Learning Objectives

Hello! And welcome to Module 4: Moving Averages.

In this module, we'll implement an algorithm for smoothing stock prices.

- In Section 1, we'll review how to create a line chart for stock prices.
- In Section 2, we'll work through a method called a "moving average" for smoothing stock prices.
- In Section 3, we'll see how to avoid creating a lot of dedicated variables by pre-allocating the smoothed values vector.
- In Section 4, I'll make a few observations about this method.

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

- Construct a line chart for stock price data.
- Explain how the moving average method works.
- Pre-allocate the smoothed values vector.
- Discuss some of the technical issues with this algorithm.

There are no new built-in R functions in this module.

All right! Let's start by reviewing line charts.

Section 1: Line Chart Review

> **Main Idea:** *Let's review line charts*

In this section, we'll review how to create a line chart for stock prices.

The closing prices for WiDgT stock on five consecutive trading days are:

Trading Day	Closing Price
1	41.00
2	43.75
3	43.00
4	44.00
5	42.50

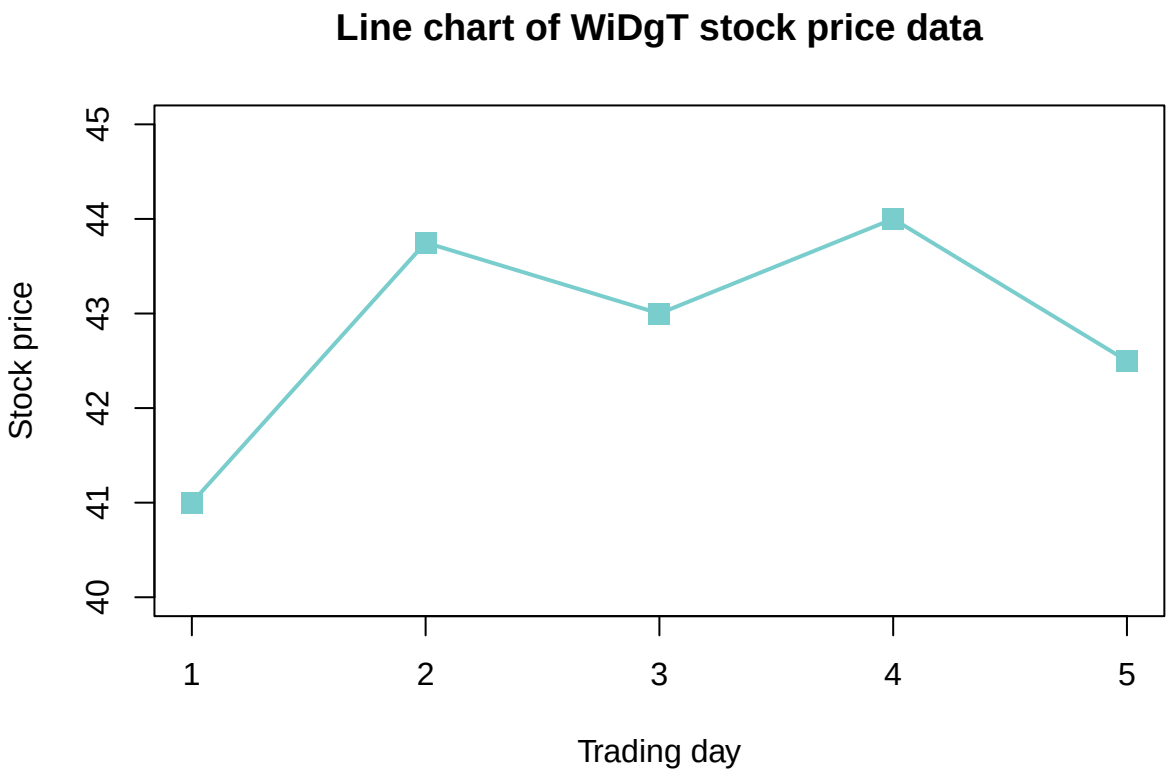
Let's store the data in these columns in vectors:

```
# Example 1: Storing the data in vectors

trading.day.index.vector <- 1:5

stock.price.vector <-
  c( 41.00, 43.75, 43.00, 44.00, 42.50)
```

Now we can draw a line chart of this data:

- First, create an empty plot with no data; remember to explicitly specify the range of the x- and y-axes, the main title, and the titles for the axes.
- Next, use the `lines()` function to draw a line graph of the daily stock price of WiDgT.
- Finally, use the `points()` function to draw points for the daily stock prices.

```
# Example 2: Drawing a line chart for the stock prices

plot(
  x = NULL,
  xlim = c(1, 5),
  ylim = c(40, 45),
  main = "Line chart of WiDgT stock price data",
  xlab = "Trading day",
  ylab = "Stock price"
)

lines(
  x = trading.day.index.vector,
  y = stock.price.vector,
  lty = "solid",
  lwd = 2,
  col = "darkslategray3"
)

points(
  x = trading.day.index.vector,
  y = stock.price.vector,
  pch = 15,
  cex = 1.5,
  col = "darkslategray3"
)
```



So that's how to draw a line chart for stock data.

Now let's explore the "moving average" method.

Exercise 5.1: Line chart of stock data

The closing prices for Sugar Bomz stock on five consecutive trading days are:

Trading Day	Closing Price
1	62.50
2	59.50
3	62.25
4	63.75
5	64.00

Store these values in vectors, and then create a line chart of this stock data.

It's fine to re-use the vector of trading days from lecture, but you should use a different name for the vector of stock prices such as `sbz.stock.price.data`.

Solution

Section 2: Smoothing with a moving average

Main Idea: *We can smooth data using a moving average*

In this section, we'll work through a method called a "moving average" for smoothing stock prices.

This graph is very jagged, and it would be nice to smooth it out.

We will use a simple strategy known as a *moving average*.

I'm going to call the observed stock prices the "raw" data, as opposed to the smoothed values created by the moving average smoothing process.

The basic strategy here is to "smooth" the graph at each time point from 2 to 4 by taking the average of the raw stock price on the day before, the raw stock price on the day, and the raw stock price on the day after.

To calculate the smoothed value at time $t = 2$, we take the average of the raw stock prices at times $t = 1$, 2, and 3.

```
# Example 3: Calculating the smoothed value for day 2

day.2.smoothed.stock.price <-
  (stock.price.vector[ 1 ] +
     stock.price.vector[ 2 ] +
     stock.price.vector[ 3 ]) /
  3

cat(
  "Smoothed value of stock price on second day:",
  formatC(
    x = day.2.smoothed.stock.price,
    format = "f",
    digits = 2
  )
)
```

```
## Smoothed value of stock price on second day: 42.58
```

To calculate the moving average for the third day, we take the average of the raw stock prices from the second day, the third day, and the fourth day.

```
# Example 4: Calculating the smoothed value for day 3

day.3.smoothed.stock.price <-
  (stock.price.vector[ 2 ] +
     stock.price.vector[ 3 ] +
     stock.price.vector[ 4 ]) /
  3

cat(
  "Smoothed value of stock price on third day:",
  formatC(
    x = day.3.smoothed.stock.price,
    format = "f",
    digits = 2
  )
)
```

```
## Smoothed value of stock price on third day: 43.58
```

To calculate the moving average for the fourth day, we take the average of the raw stock price from the third day, the fourth day, and the fifth day.

```
# Example 5: Calculating the smoothed value for day 4

day.4.smoothed.stock.price <-
  (stock.price.vector[ 3 ] +
     stock.price.vector[ 4 ] +
     stock.price.vector[ 5 ]) /
  3

cat(
  "Smoothed value of stock price on fourth day:",
  formatC(
    x = day.4.smoothed.stock.price,
    format = "f",
    digits = 2
  )
)
```

```
## Smoothed value of stock price on fourth day: 43.17
```

Notice that we are always calculating an average, but the values for this calculation change depending on which day we are working with.

That's why this method is called a "moving average".

So far, we've smoothed the values for days 2 through 4.

But what about the first day?

We can't use our moving average algorithm, because there's no previous day.

For CSCI E-5a, we will simply use the value of the raw stock price on day 1 for the smoothed value:

```
# Example 6: Calculating the smoothed value for day 1

day.1.smoothed.stock.price <-
  stock.price.vector[ 1 ]
```

Similarly, for the last day, we can't use the moving average method, because there is no data for the following day.

Again, for day 5, we'll just use the raw stock price as the smoothed value:

```
# Example 7: Calculating the smoothed value for day 5

day.5.smoothed.stock.price <-
  stock.price.vector[ 5 ]
```

Thus, for the end points, we'll just use the raw stock prices for our smoothed values.

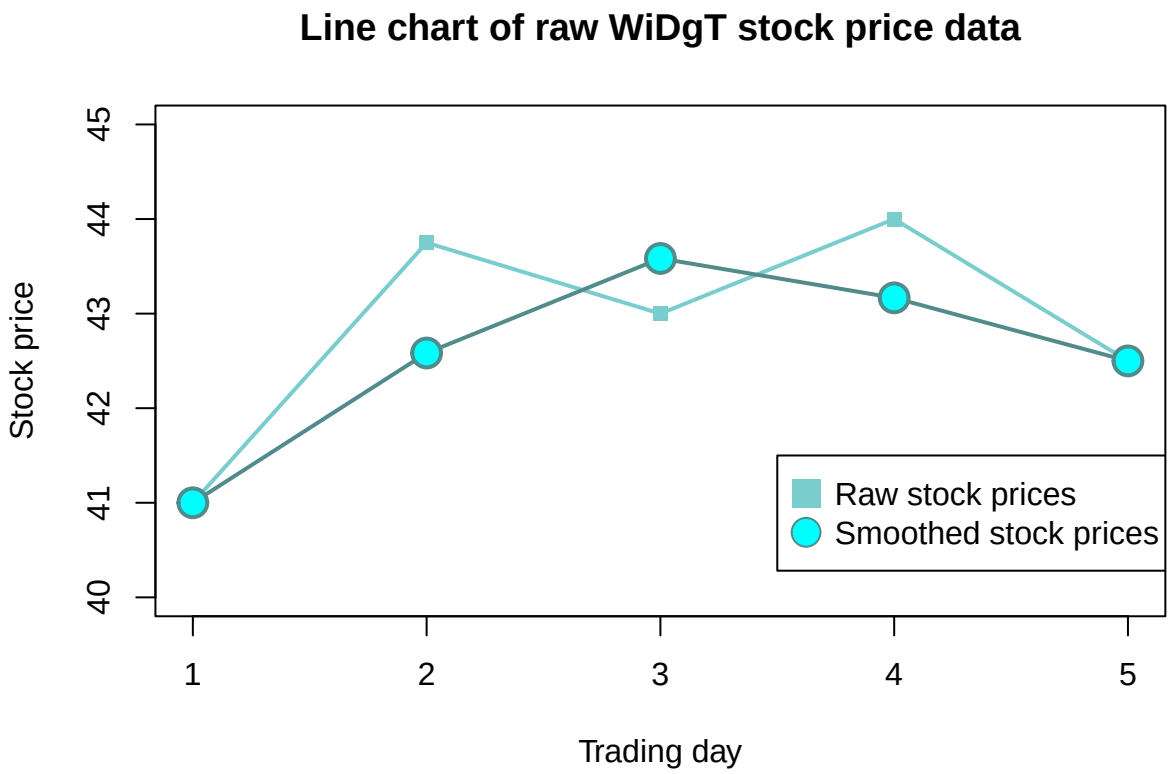
Let's store these smoothed values in a vector:

```
# Example 8: Storing the smoothed values in a vector

smoothed.stock.price.vector <-
    c(
        day.1.smoothed.stock.price,
        day.2.smoothed.stock.price,
        day.3.smoothed.stock.price,
        day.4.smoothed.stock.price,
        day.5.smoothed.stock.price
    )
```

Now we're going to redraw the graph that we did previously, but we'll add in the smoothed values.

```
# Example 9: Line chart with raw and smoothed values

plot(
  x = NULL,
  xlim = c(1, 5),
  ylim = c(40, 45),
  main = "Line chart of raw WiDgT stock price data",
  xlab = "Trading day",
  ylab = "Stock price"
)

lines(
  x = trading.day.index.vector,
  y = stock.price.vector,
  lty = "solid",
  lwd = 2,
  col = "darkslategray3"
)

points(
  x = trading.day.index.vector,
  y = stock.price.vector,
  pch = 15,
  lwd = 2,
  col = "darkslategray3"
)

lines(
  x = trading.day.index.vector,
  y = smoothed.stock.price.vector,
  lty = "solid",
  lwd = 2,
  col = "darkslategray4"
)

points(
  x = trading.day.index.vector,
  y = smoothed.stock.price.vector,
  pch = 21,
  cex = 2,
  lwd = 2,
```

```
  bg = "cyan",
  col = "darkslategray4"
)

legend(
  x = 3.5,
  y = 41.5,
  legend = c( "Raw stock prices", "Smoothed stock prices" ),
  pch = c( 15, 21),
  col = c( "darkslategray3", "darkslategray4"),
  pt.bg = "cyan",
  pt.cex = 2
)
```



You can see that our moving average smoother did indeed generate a graph that looks less jagged (i.e. "smoother") that the original raw stock price.

So that's how to smooth data using a moving average.

Now let's see how to make this code more general by pre-allocating a storage vector.

Exercise 5.2: Line chart of smoothed stock data

The closing prices for Sugar Bomz stock on five consecutive trading days are:

Trading Day	Closing Price
1	62.50
2	59.50
3	62.25
4	63.75
5	64.00

Calculate the smoothed values for these stock prices using the moving average approach. Then copy your graph from Exercise 5.1 and superimpose the line graph for the smoothed values.

Solution

Section 3: Pre-allocating the smoothed values vector

Main Idea: *We can pre-allocate storage*

In this section, we'll see how to avoid creating a lot of dedicated variables by pre-allocating the smoothed values vector.

In our calculation, we had to create separate dedicated variables for each trading day to hold the smoothed value for that day.

What would happen if we had 50 points?

It would be a lot of work to have to create 50 separate variables to hold the smoothed values.

There is a better approach.

Instead of creating lots of dedicated variables for each trading day and then assembling a vector using the `c()` function, we can first create a vector for the smoothed values and then populate it as we perform our calculations.

When we create a vector, we are allocating storage space for the values, so I'm going to call this approach "pre-allocating the smoothed values vector".

To create a numeric vector with 5 elements, we can use the `numeric()` function, which takes a single input argument which specifies the number of elements:

```
# Example 10: Pre-allocating the smoothed values vector

smoothed.stock.price.vector <- numeric( 5 )
```

This will create a numeric vector with 5 elements, and as it turns out all the values are 0:

```
# Example 11: Directly displaying the smoothed values vector

smoothed.stock.price.vector
```

```
## [1] 0 0 0 0 0
```

It actually doesn't matter what the initial values are, because we're going to destructively modify all of them, so any method we have for creating a vector with 5 elements will be sufficient.

Can you think of another method for creating a vector of length 5 where all the elements of the vector have the value 0?

The first element of the smoothed values is just the raw stock price for the first day:

```
# Example 12: Assigning the smoothed value for day 1

smoothed.stock.price.vector[ 1 ] <-
  stock.price.vector[ 1 ]
```

The second element of the smoothed values is the moving average value for day 2:

```
# Example 13: Assigning the smoothed value for day 2

smoothed.stock.price.vector[ 2 ] <-
  (stock.price.vector[1] +
     stock.price.vector[2] +
     stock.price.vector[3]) /
  3
```

The third element of the smoothed values is the moving average value for day 3:

```
# Example 14: Assigning the smoothed value for day 3

smoothed.stock.price.vector[ 3 ] <-
  (stock.price.vector[2] +
     stock.price.vector[3] +
     stock.price.vector[4]) /
  3
```

The fourth element of the smoothed values is the moving average value for day 4:

```
# Example 15: Assigning the smoothed value for day 4

smoothed.stock.price.vector[ 4 ] <-
  (stock.price.vector[3] +
     stock.price.vector[4] +
     stock.price.vector[5]) /
  3
```

Finally, the fifth element of the smoothed values is just the raw stock price for day 5:

```
# Example 16: Assigning the smoothed value for day 5

smoothed.stock.price.vector[ 5 ] <-
  stock.price.vector[ 5 ]
```

Notice that we didn't have to create dedicated variables for each trading day, and only later assemble them into the vector of smoothed values.

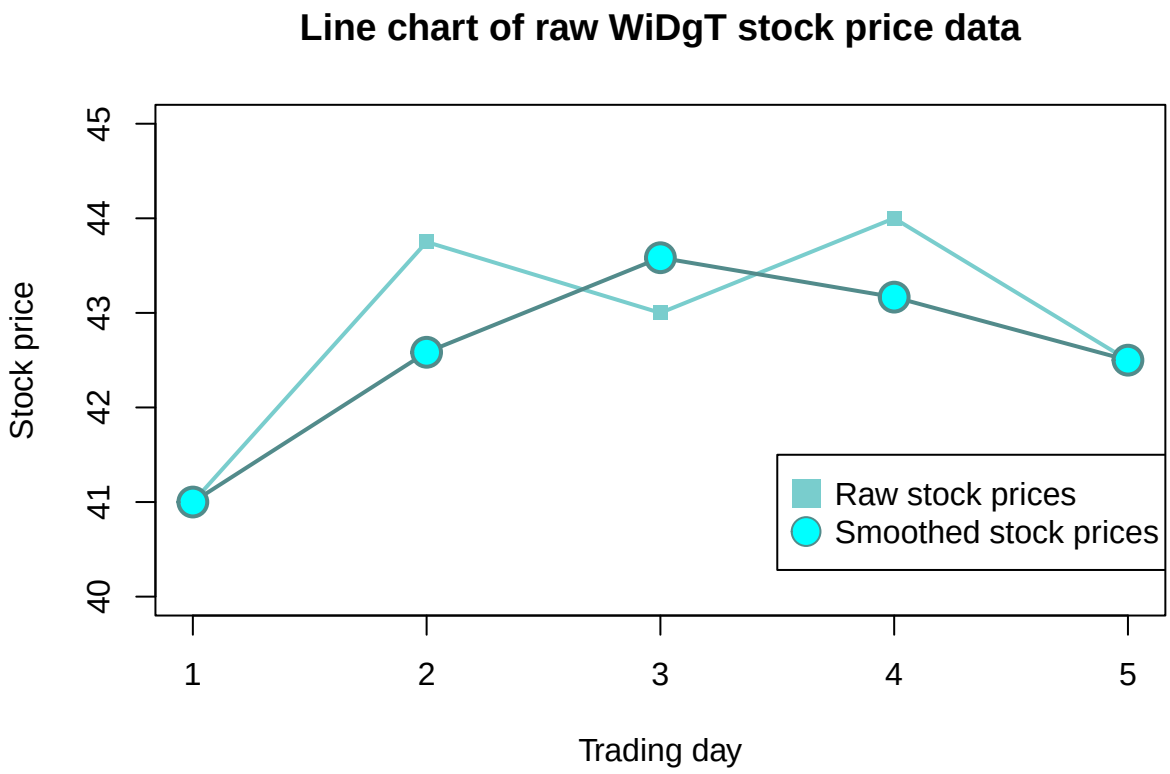
Instead, by creating the vector first, and then using positive integer indexing, we avoided having to create any special dedicated variables at all.

Of course, in the end, we have the same set of smoothed values, because the actual algorithm that we are using to calculate the smoothed values is the same.

Let's redraw the graph using this vector:

```r
# Example 17: Line chart with raw and smoothed values

plot(
  x = NULL,
  xlim = c(1, 5),
  ylim = c(40, 45),
  main = "Line chart of raw WiDgT stock price data",
  xlab = "Trading day",
  ylab = "Stock price"
)

lines(
  x = trading.day.index.vector,
  y = stock.price.vector,
  lty = "solid",
  lwd = 2,
  col = "darkslategray3"
)

points(
  x = trading.day.index.vector,
  y = stock.price.vector,
  pch = 15,
  lwd = 2,
  col = "darkslategray3"
)

lines(
  x = trading.day.index.vector,
  y = smoothed.stock.price.vector,
  lty = "solid",
  lwd = 2,
  col = "darkslategray4"
)

points(
  x = trading.day.index.vector,
  y = smoothed.stock.price.vector,
  pch = 21,
  cex = 2,
  lwd = 2,
  bg = "cyan",
  col = "darkslategray4"
)

legend(
  x = 3.5,
  y = 41.5,
  legend = c( "Raw stock prices", "Smoothed stock prices" ),
  pch = c( 15, 21),
  col = c( "darkslategray3", "darkslategray4"),
  pt.bg = "cyan",
  pt.cex = 2
```

```
)
```



Of course this looks the same as before.

So that's how to pre-allocate storage for the moving average.

Now let's reflect on some features of this procedure.

Exercise 5.3: Line chart of smoothed stock data

The closing prices for Sugar Bomz stock on five consecutive trading days are:

Trading Day	Closing Price
1	62.50
2	59.50
3	62.25
4	63.75
5	64.00

Pre-allocate storage for the smoothed values vector. Then calculate the smoothed values for these stock prices using the moving average approach and store these directly in the smoothed values vector without creating any intermediate variables. Then copy your graph from Exercise 5.1 and superimpose the line graph for the smoothed values.

Solution

Section 4: A few observations

Main Idea: *Let's reflect about this moving average method*

In this section, I'll make a few observations about this method.

Here are a few observations about this moving average calculation.

Because of the nature of the moving average method, we can't use it for the first time point, because there is no previous time point for the average.

In CSCI S-5a, for day 1 we will always use the raw stock price for the smoothed value.

Similarly, we can't use the algorithm for the last time point, so if there are n time points in our data then the smoothed value for day n will just be the raw stock price for that day.

In our example, we only had a few trading days to work with, and I listed all the values in a little table, so it was easy to realize that there were only 5 trading days.

In the real world, you won't necessarily know how many trading days there are, and your code will have to determine this number.

For trading days 2 through *n-1* we use the moving average method to calculate the smoothed value.

This procedure is highly repetitive, and we perform a similar calculation for days 2, 3, and 4.

Since this process is so repetitive, it's a natural candidate for automation by using a `for` loop.

Big hint: I think you'll find that this `for` loop should iterate over indices.

So this will be our general strategy:

- First, determine how many trading days there are.
- Next, pre-allocate a vector for the smoothed values.
- For the first element of the smoothed values vector, assign the raw stock price on day 1.
- For the last element of the smoothed values vector, assign the raw stock price on day n.
- Finally, for trading days 2 through *n-1*, construct a `for` loop that will calculate the moving average for each trading day, and then assign that to the corresponding element of the smoothed values vector.

Once you've created the smoothed values vector, it's straightforward to graph it:

- Create an empty plot with no data.
- Use the `lines()` and `points()` functions to plot the raw stock price values.
- Use the `lines()` and `points()` functions to plot the smoothed stock price values.

So those are some things to think about.

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

Module Review

In this module, we implemented an algorithm for smoothing stock prices.

- In Section 1, we reviewed how to create a line chart for stock prices.
- In Section 2, we worked through a method called a "moving average" for smoothing stock prices.
- In Section 3, we saw how to avoid creating a lot of dedicated variables by pre-allocating the smoothed values vector.
- In Section 4, I made a few observations about this method.

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

- Construct a line chart for stock price data.
- Explain how the moving average method works.
- Pre-allocate the smoothed values vector.
- Discuss some of the technical issues with this algorithm.

There were no new built-in R functions in this module.

All right! That's it for Module 4: Moving Averages.

Now let's move on to Module 5: Histograms.

Solutions to the Exercises

Exercise 5.1: Line chart of stock data

The closing prices for Sugar Bomz stock on five consecutive trading days are:

Trading Day	Closing Price
1	62.50
2	59.50
3	62.25
4	63.75
5	64.00

Store these values in vectors, and then create a line chart of this stock data.

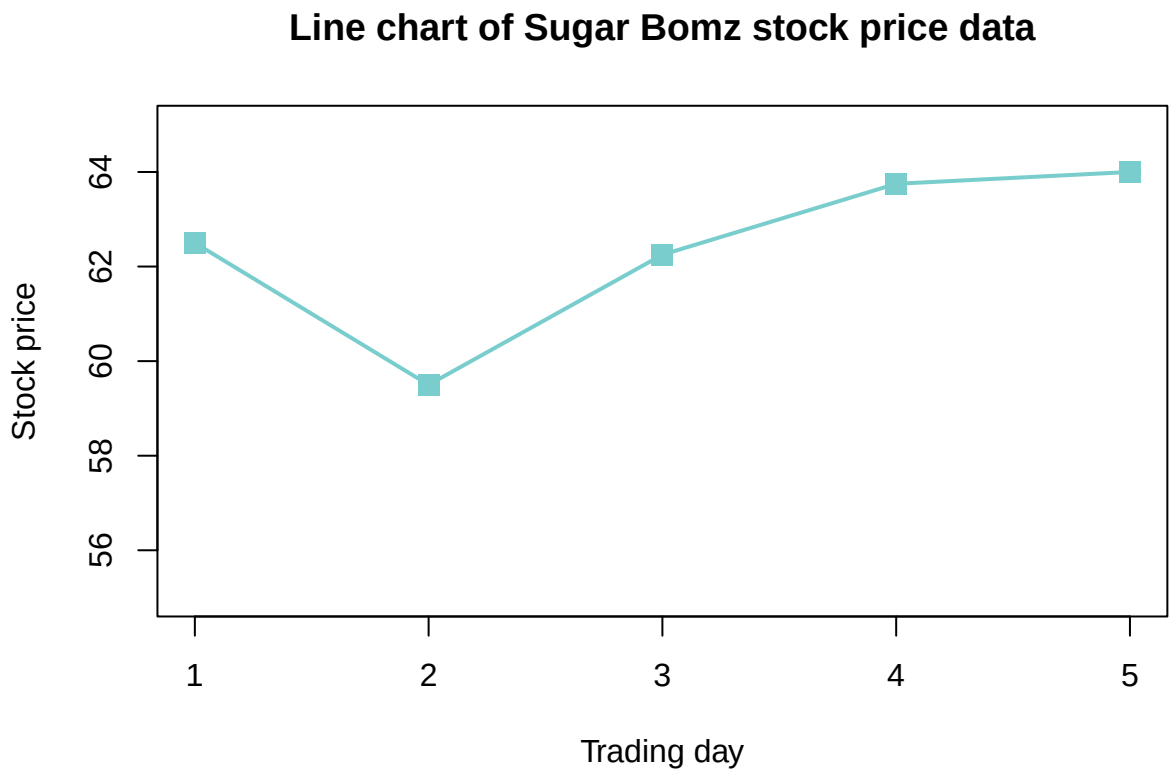
Solution

Let's store the data in these columns in vectors:

```
sbz.stock.price.vector <-
  c( 62.50, 59.50, 62.25, 63.75, 64.00)
```

Now we can draw the line chart of Sugar Bomz stock prices:

```
plot(
  x = NULL,
  xlim = c(1, 5),
  ylim = c(55, 65),
  main = "Line chart of Sugar Bomz stock price data",
  xlab = "Trading day",
  ylab = "Stock price"
)

lines(
  x = trading.day.index.vector,
  y = sbz.stock.price.vector,
  lty = "solid",
  lwd = 2,
  col = "darkslategray3"
)

points(
  x = trading.day.index.vector,
  y = sbz.stock.price.vector,
  pch = 15,
  cex = 1.5,
  col = "darkslategray3"
)
```



Exercise 5.2: Line chart of smoothed stock data

The closing prices for Sugar Bomz stock on five consecutive trading days are:

Trading Day	Closing Price
1	62.50
2	59.50
3	62.25
4	63.75
5	64.00

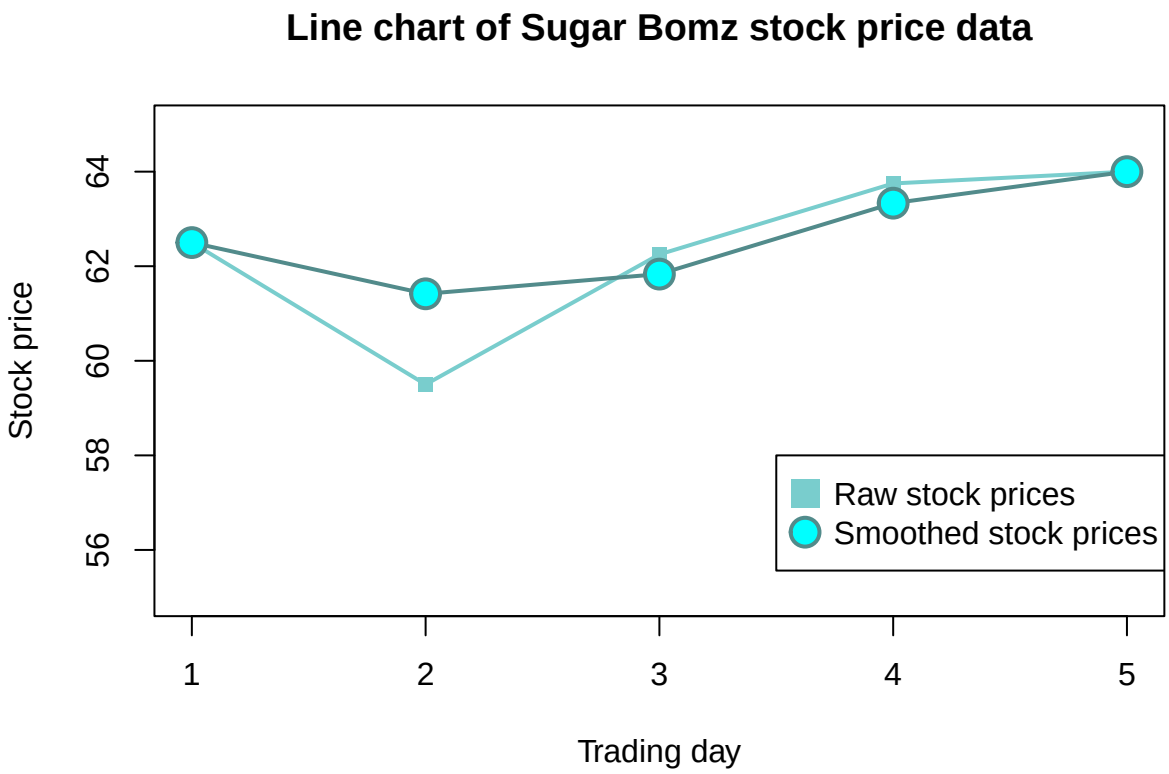
Calculate the smoothed values for these stock prices using the moving average approach. Then copy your graph from Exercise 5.1 and superimpose the line graph for the smoothed values.

Solution

```
sbz.day.1.smoothed.stock.price <-
  sbz.stock.price.vector[ 1 ]

sbz.day.2.smoothed.stock.price <-
  (sbz.stock.price.vector[ 1 ] +
     sbz.stock.price.vector[ 2 ] +
     sbz.stock.price.vector[ 3 ]) /
  3

sbz.day.3.smoothed.stock.price <-
  (sbz.stock.price.vector[ 2 ] +
     sbz.stock.price.vector[ 3 ] +
     sbz.stock.price.vector[ 4 ]) /
  3

sbz.day.4.smoothed.stock.price <-
  (sbz.stock.price.vector[ 3 ] +
     sbz.stock.price.vector[ 4 ] +
     sbz.stock.price.vector[ 5 ]) /
  3

sbz.day.5.smoothed.stock.price <-
  sbz.stock.price.vector[ 5 ]

sbz.smoothed.stock.price.vector <-
    c(
        sbz.day.1.smoothed.stock.price,
        sbz.day.2.smoothed.stock.price,
        sbz.day.3.smoothed.stock.price,
        sbz.day.4.smoothed.stock.price,
        sbz.day.5.smoothed.stock.price
    )
```

Now we can create the graph:

```
plot(
  x = NULL,
  xlim = c(1, 5),
  ylim = c(55, 65),
  main = "Line chart of Sugar Bomz stock price data",
  xlab = "Trading day",
  ylab = "Stock price"
)

lines(
  x = trading.day.index.vector,
  y = sbz.stock.price.vector,
  lty = "solid",
  lwd = 2,
  col = "darkslategray3"
)

points(
  x = trading.day.index.vector,
  y = sbz.stock.price.vector,
  pch = 15,
  lwd = 2,
  col = "darkslategray3"
)

lines(
  x = trading.day.index.vector,
  y = sbz.smoothed.stock.price.vector,
  lty = "solid",
  lwd = 2,
  col = "darkslategray4"
)

points(
  x = trading.day.index.vector,
  y = sbz.smoothed.stock.price.vector,
  pch = 21,
  cex = 2,
  lwd = 2,
  bg = "cyan",
  col = "darkslategray4"
)

legend(
  x = 3.5,
  y = 58,
  legend = c( "Raw stock prices", "Smoothed stock prices" ),
  pch = c( 15, 21),
  col = c( "darkslategray3", "darkslategray4"),
  pt.bg = "cyan",
  pt.lwd = 2,
  pt.cex = 2
)
```



Exercise 5.3: Line chart of smoothed stock data

The closing prices for Sugar Bomz stock on five consecutive trading days are:

Trading Day	Closing Price
1	62.50
2	59.50
3	62.25
4	63.75
5	64.00

Pre-allocate storage for the smoothed values vector. Then calculate the smoothed values for these stock prices using the moving average approach and store these directly in the smoothed values vector without creating any intermediate variables. Then copy your graph from Exercise 5.1 and superimpose the line graph for the smoothed values.

Solution

```
sbz.smoothed.stock.price.vector <-
  numeric( 5 )

sbz.smoothed.stock.price.vector[ 1 ] <-
  sbz.stock.price.vector[ 1 ]
```

```
sbz.smoothed.stock.price.vector[ 2 ] <-
  (sbz.stock.price.vector[ 1 ] +
     sbz.stock.price.vector[ 2 ] +
     sbz.stock.price.vector[ 3 ] ) /
  3

sbz.smoothed.stock.price.vector[ 3 ] <-
  (sbz.stock.price.vector[ 2 ] +
     sbz.stock.price.vector[ 3 ] +
     sbz.stock.price.vector[ 4 ] ) /
  3

sbz.smoothed.stock.price.vector[ 4 ] <-
  (sbz.stock.price.vector[ 3 ] +
     sbz.stock.price.vector[ 4 ] +
     sbz.stock.price.vector[ 5 ] ) /
  3


sbz.smoothed.stock.price.vector[ 5 ] <-
  sbz.stock.price.vector[ 5 ]
```

```
plot(
  x = NULL,
  xlim = c(1, 5),
  ylim = c(55, 65),
  main = "Line chart of Sugar Bomz stock price data",
  xlab = "Trading day",
  ylab = "Stock price"
)

lines(
  x = trading.day.index.vector,
  y = sbz.stock.price.vector,
  lty = "solid",
  lwd = 2,
  col = "darkslategray3"
)

points(
  x = trading.day.index.vector,
  y = sbz.stock.price.vector,
  pch = 15,
  lwd = 2,
  col = "darkslategray3"
)

lines(
  x = trading.day.index.vector,
  y = sbz.smoothed.stock.price.vector,
  lty = "solid",
  lwd = 2,
  col = "darkslategray4"
```

```
)

points(
  x = trading.day.index.vector,
  y = sbz.smoothed.stock.price.vector,
  pch = 21,
  cex = 2,
  lwd = 2,
  bg = "cyan",
  col = "darkslategray4"
)

legend(
  x = 3.5,
  y = 58,
  legend = c( "Raw stock prices", "Smoothed stock prices" ),
  pch = c( 15, 21),
  col = c( "darkslategray3", "darkslategray4"),
  pt.bg = "cyan",
  pt.lwd = 2,
  pt.cex = 2
)
```