Time series in html Canvas

In this tutorial, we will create an html document and use it to animate a time series.

html

html (Hypertext Markup Language) is used to make web pages. No software is needed to create an html document except for a text editor, but a browser is needed if you want to view an html document.

To create an html document, open up a text file (for example in Notepad) and save it as "example.html". Type the following lines into the text file.

```
<!doctype html>
<html>

</html>
```

The things like `<html>` are called **tags**. They determine the nature of whatever appears in between them. They appear in pairs; each tag, such as `<html>` has a corresponding closing tag, in this case `</html>`. When the browser encounters an html document, it uses the tags to decide how to display different parts of the document. Adding the tags is called **marking up** the document.

Head and body

Each html document should include a header section, enclosed in <head> tags, and a body section, enclosed in <body> tags. These must go in between the `<html>` tags. The body is the main part of the document. Add head and body sections to example.html, so that it now looks like this.

```
<!doctype html>
<html>

<head></head>

<body></body>

</html>
```

Now put a title in the header section, enclosed in `title` tags.

```
<!doctype html>
<html>

<head>
<title>A Time Series Example</title>
</head>

<body></body>

</html>
```

Save the document and then open it in a browser. You will see a blank page, but the tab should say "A Time Series Example". This is the title of your page. Now add some text to the body section. We will make a graph of the price of a stock, United Marshmallow. Add this into the page and it will look like this.

```
<!doctype html>
<html>

<head>
<title>A Time Series Example</title>
</head>

<body>
United Marshmallow
</body>

</html>
```

That title doesn't look very impressive, so make it a heading by enclosing it in <h1> tags.

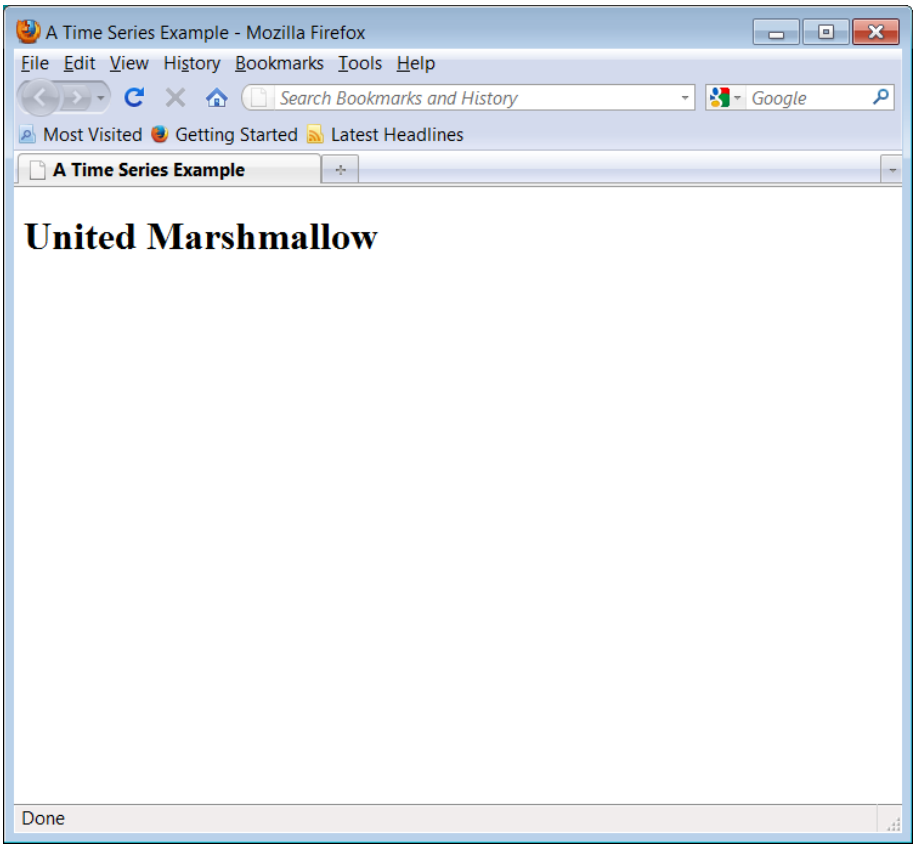
```
<!doctype html>
<html>

<head>
<title>A Time Series Example</title>
</head>

<body>
<h1>United Marshmallow</h1>
</body>

</html>
```

This is what the file should look like if you open it up in your browser by double-clicking.



divs

Now let's make a stock ticker. Inside the <body> tags, below the <h1> line, insert the following line.

```
<div id="price">20</div>
```

The tags don't change the appearance of the text in between them, but they are very useful for dividing up the document. In order to identify the div later on, it has been given an id, in this case "price".

Scripts and JavaScript

At the moment, the price of United Marshmallow is constant, which is not very realistic. Let us make the price change over time. We can do this using a **script**. How can we include a script in our document? We can use <script> tags!

You can put the script anywhere in between the <html> tags, but we will put it after the <body> section. It's common to put the script in a separate file, unless it is very short. In this example, we will keep everything in one file. Add the following lines between the </body> and </html> tags.

```
<script type="text/javascript">
</script>
```

Our script will go in between the <script> tags. It is written in a programming language called **JavaScript**. This language will run automatically when we open the webpage, unless scripts are blocked by our browser.

JavaScript is superficially similar to R, and I am a superficial programmer. JavaScript is actually nicer than R in many ways. Although it has many notorious "gotchas" (unexpected or illogical behaviours) it does not have as many as R does.

United Marshmallow is not a real company, so we will need to make up its stock price. We will assume that at each moment the stock price either increases by a factor of 1.1 or decreases by a factor of 0.9. We will use a variable price to represent the price.

In JavaScript, variables are declared with the var keyword. Insert these lines in between the script tags. Note that every command in a JavaScript program has to end in a semicolon. There is no harm in including spaces in between lines if it makes the code easier to read.

```
var price = 50;
var flip;

while (0==0){
  flip = Math.random();
  if (flip > 0.5){
    price = price*1.1;
    }
  else{
    price = price*0.9;
  }
document.getElementById("price").innerHTML = price;
}
```

In the code, we declare a variable `price` and another variable `flip`. We give `price` the value 20, but we don't have to give a value if we don't want to. Then we start a while loop which will run forever because the condition `0==0` is always true.

Inside the loop, we assign a random number between 0 and 1 to the variable `flip` using the `Math.random()` command (this is identical to `runif(1)` in R). We could have put the variable declaration and the assignment into a single line by writing

```
var flip = Math.random();
```

Then, the if ... else statement changes the price as desired. Finally, we have the complicated-looking line:

```
document.getElementById("price").innerHTML = price;
```

This command tells JavaScript to get the object named "price", which in our case is a `<div>` which starts off with the value 20 written in it. This object has an attribute called `innerHTML`, which stands for whatever is inside it. We change whatever is inside it to whatever value is stored in the variable `price`. The effect of the whole line is to write the current price in the "price" section of the document.

example.html should now look like this:

```
<!doctype html>
<html>

<head>
<title>A Time Series Example</title>
</head>

<body>
<h1>United Marshmallow</h1>
<div id="price">20</div>
</body>

<script type="text/javascript">
var price = 50;
var flip;

while (0==0){
  flip = Math.random();
  if (flip > 0.5){
    price = price*1.1;
    }
  else{
    price = price*0.9;
  }
document.getElementById("price").innerHTML = price;
}
</script>

</html>
```

If you try to run this in Firefox, it will freeze up and crash. This is because your script is updating too quickly. To make it work, we need to add a **timer** which updates it at regular intervals.

Timers

Adding a timer is easy in JavaScript. The timer needs to call a **function**, so we replace the while loop with a function, like this:

```
function updatePrice(){
  flip = Math.random();
  if (flip > 0.5){
    price = price*1.1;
    }
  else{
    price = price*0.9;
  }
  document.getElementById("price").innerHTML = price;
}
```

and then we just need to add the line

```
var timer = setInterval(updatePrice, 10);
```

and our script will call the `updatePrice()` function every ten milliseconds.

The script should now run and you should see the price being rapidly updated. The price has many decimal digits, so we can round it off to the nearest whole number using `Math.round()` like this

```
document.getElementById("price").innerHTML = Math.round(price);
```

Here is the whole file so far.

```
<!doctype html>
<html>

<head>
<title>A Time Series Example</title>
</head>

<body>
<h1>United Marshmallow</h1>
<div id="price">20</div>
</body>

<script type="text/javascript">
var price = 50;
var flip;

function updatePrice(){
  flip = Math.random();
  if (flip > 0.5){
    price = price*1.1;
    }
  else{
    price = price*0.9;
  }
  document.getElementById("price").innerHTML = Math.round(price);
}

var timer= setInterval(updatePrice, 10);
</script>
```

```
</html>
```

Canvas

Now we will make a graph of the stock price. This is done using the <canvas> element. This is a feature of HTML5 and it will only work in more modern browsers. Add a canvas by typing the following into the <body> section below the .

```
<canvas id="myCanvas" width="600" height="300" ></canvas>
```

If you run the file now, it will look exactly the same as before. But now it includes a 300x500 pixel canvas on which we can draw. Let us verify this by filling the canvas with a black rectangle.

Add these lines at the start of the script.

```
var context = document.getElementById("myCanvas").getContext("2d");
context.fillStyle = "black";
context.fillRect(0, 0, 600, 300);
```

The context is an object which allows us to draw on the canvas and we have to fetch it in order to draw anything. We then set the `fillStyle` to "black" and call the `fillRect()` function (aka "method") which is part of the context object. You should now see a black rectangle when you open up the file in a browser.

Drawing lines on the canvas

Now we are ready to animate the time series. Add this to the script to declare that we will be drawing white lines.

```
context.strokeStyle = "white";
```

Now we will add some code to the `updatePrice()` function. At the t'th time step, we want to draw a line from (t, current price) to (t+1, new price). Add two more variables

```
var t=0;
var oldPrice = price;
```

and then in the body of the `updatePrice()` function, add the following commands.

```
context.beginPath();
context.moveTo(t, oldPrice);
context.lineTo(t+1, price);
context.stroke();
context.closePath();

t = t+1;
oldPrice = price;
```

You can see that Canvas graphics are a bit lower-level than R graphics. First, we tell it to begin a path. Then we tell it where to move. Then where to draw a line. Then we use the `stroke` method to actually draw a line. Then we tell it to stop drawing the path.

We also have to increment `t`, or it will not move along the x-axis. Similarly, we need to set `oldPrice` to the current price before we update the price when we next call the function.

You should now see a nice-looking graph of the United Marshmallow stock price. It will be different each time you re-load the page.

There is one thing wrong here; the stock seems to go up when the price goes down! This is because computer graphics tend to have (0,0) in the top left-hand corner with the x- and y-axes going like this:

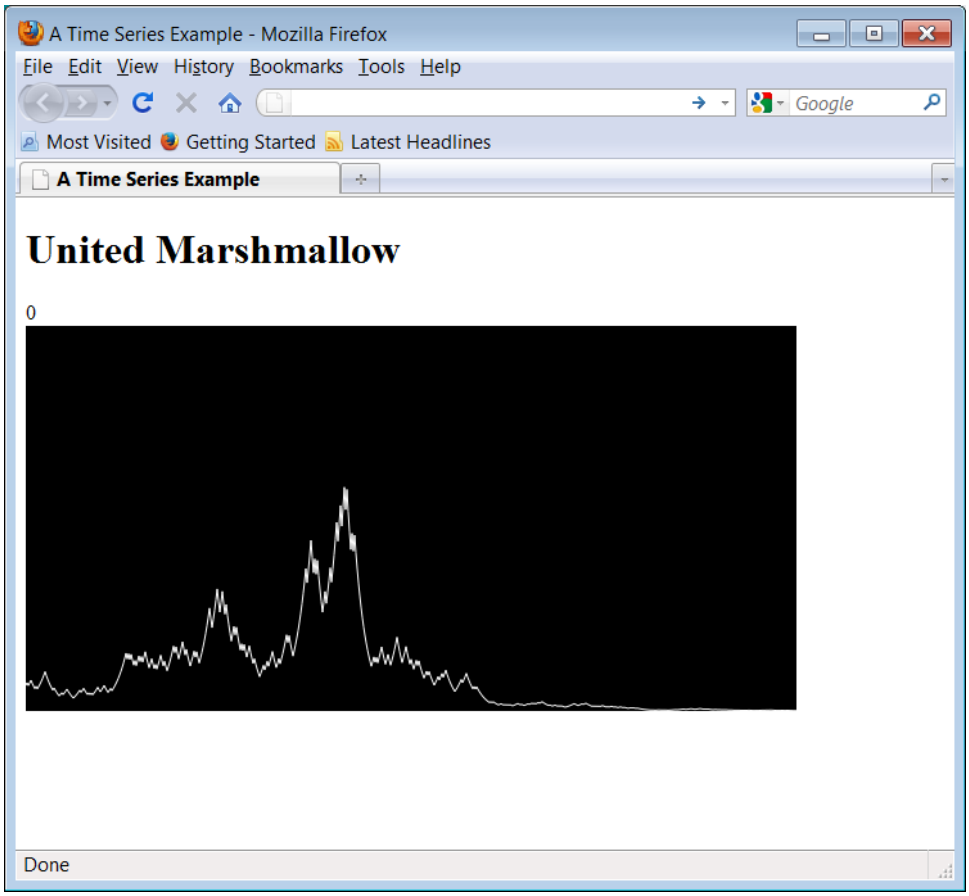
Languages like R and MatLab are unusual in this respect because they are designed for scientists. We can easily fix this by just changing the y coordinate of any points on our canvas to myCanvas.height – y. Replace the lines

```
context.moveTo(t, oldPrice);
context.lineTo(t+1, price);
```

by

```
context.moveTo(t, 300 - oldPrice);
context.lineTo(t+1, 300 - price);
```

and now the graph will be the right way up (note: rather than hard-coding the value 300, better practice is to use `myCanvas = document.getElementById("myCanvas"); height = myCanvas.height;` and then using the value `height`. Then if you change height, everything will still work.)



Further information

There are many other things you can do with JavaScript and Canvas. Here is a list of references.

- StackOverflow http://stackoverflow.com/ is a good reference for programming questions.
- You can right-click on any webpage and choose View Source in order to see the underlying html code.
- W3Schools.com has many html and JavsScript tutorials. It tends to be viewed with disdain by professional programmers, but I find it quite useful.
- An excellent free JavaScript textbook by Martijn Haverbeke is available at http://eloquentjavascript.net/ It assumes no prior knowledge of programming and can be read on-line.
- For Canvas, I found the book *HTML5 Canvas* by Fulton and Fulton very useful. It is published by O'Reilly.
- There are also many free Canvas tutorials on the internet.

More sophisticated JavaScript involves the use of libraries (collections of code that play a similar role to packages in R.)

- jQuery is a very popular library which contains useful functions.
- D3.js is a JavaScript library for data-driven documents. I haven't used it but it is extremely popular. It uses SVG, which is another way of putting graphics in web pages and is different to Canvas.

The finished example.html

```
<!doctype html>
<html>

<head>
<title>A Time Series Example</title>
</head>

<body>
<h1>United Marshmallow</h1>
<div id="price">20</div>
<canvas id="myCanvas" height="300" width="600"></canvas>
</body>

<script type="text/javascript">
var price = 50;
var flip;
var t=0;
var oldPrice = price;

var context = document.getElementById("myCanvas").getContext("2d");
context.fillStyle = "black";
context.fillRect(0, 0, 600, 300);
context.strokeStyle = "white";

function updatePrice(){
  flip = Math.random();
  if (flip > 0.5){
    price = price*1.1;
    }
  else{
    price = price*0.9;
  }
  document.getElementById("price").innerHTML = Math.round(price);

  context.beginPath();
  context.moveTo(t, 300 - oldPrice);
  context.lineTo(t+1, 300 - price);
  context.stroke();
  context.closePath();

  t = t+1;
  oldPrice = price;
}

var timer= setInterval(updatePrice, 10);
</script>

</html>
```