Filtering by Action

Code

My goal is to look at filtering out the actions so that in each condition it would be easier to see where actions such as play, pause, and rewind were happening.

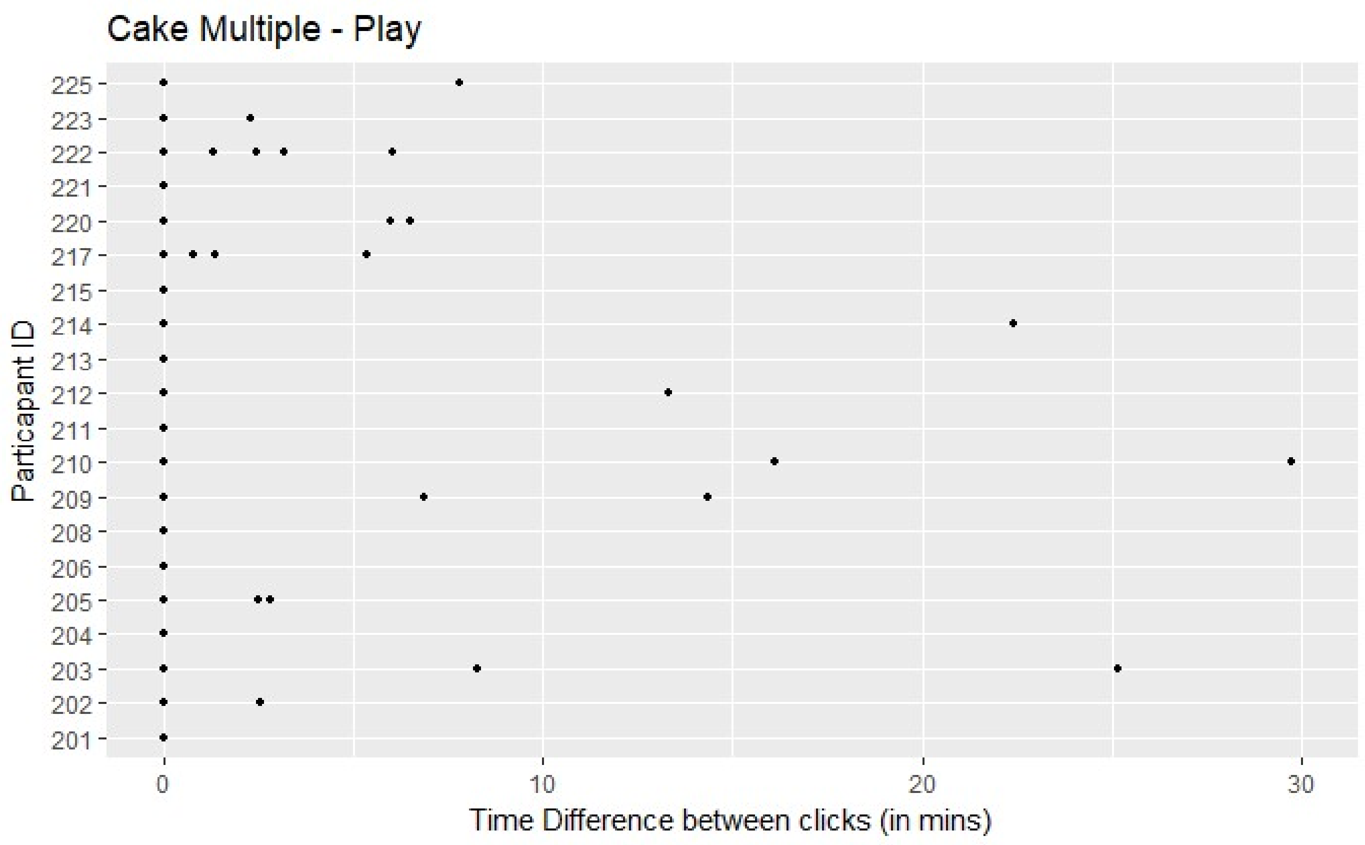
Filtering Individual Actions

I took Cake Multiple and looked at the opening 30 mins.

`r` r

Hide

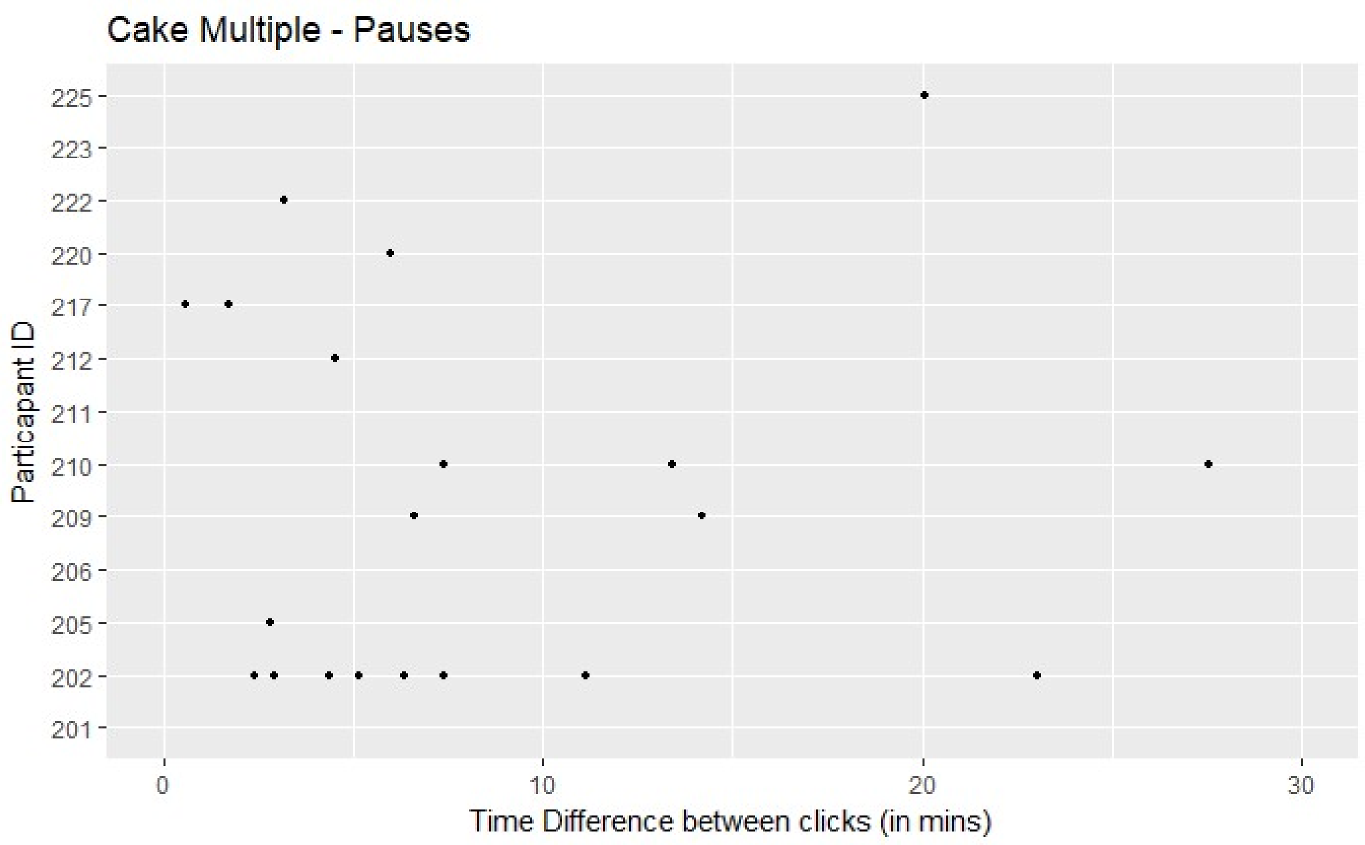
```
plot(play)
```



`r` r

Hide

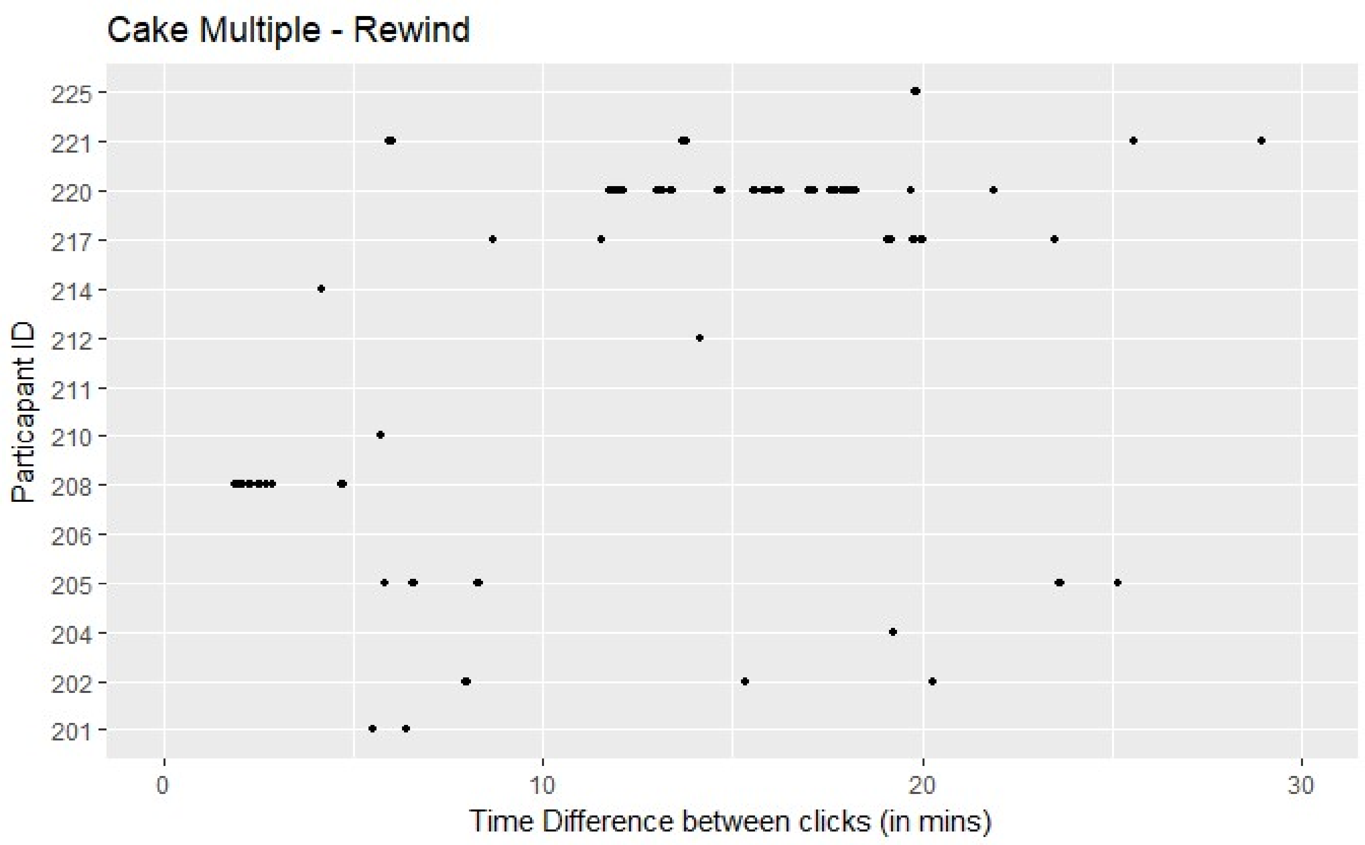
```
plot(pause)
```



`r` r

Hide

```
plot(rewind)
```



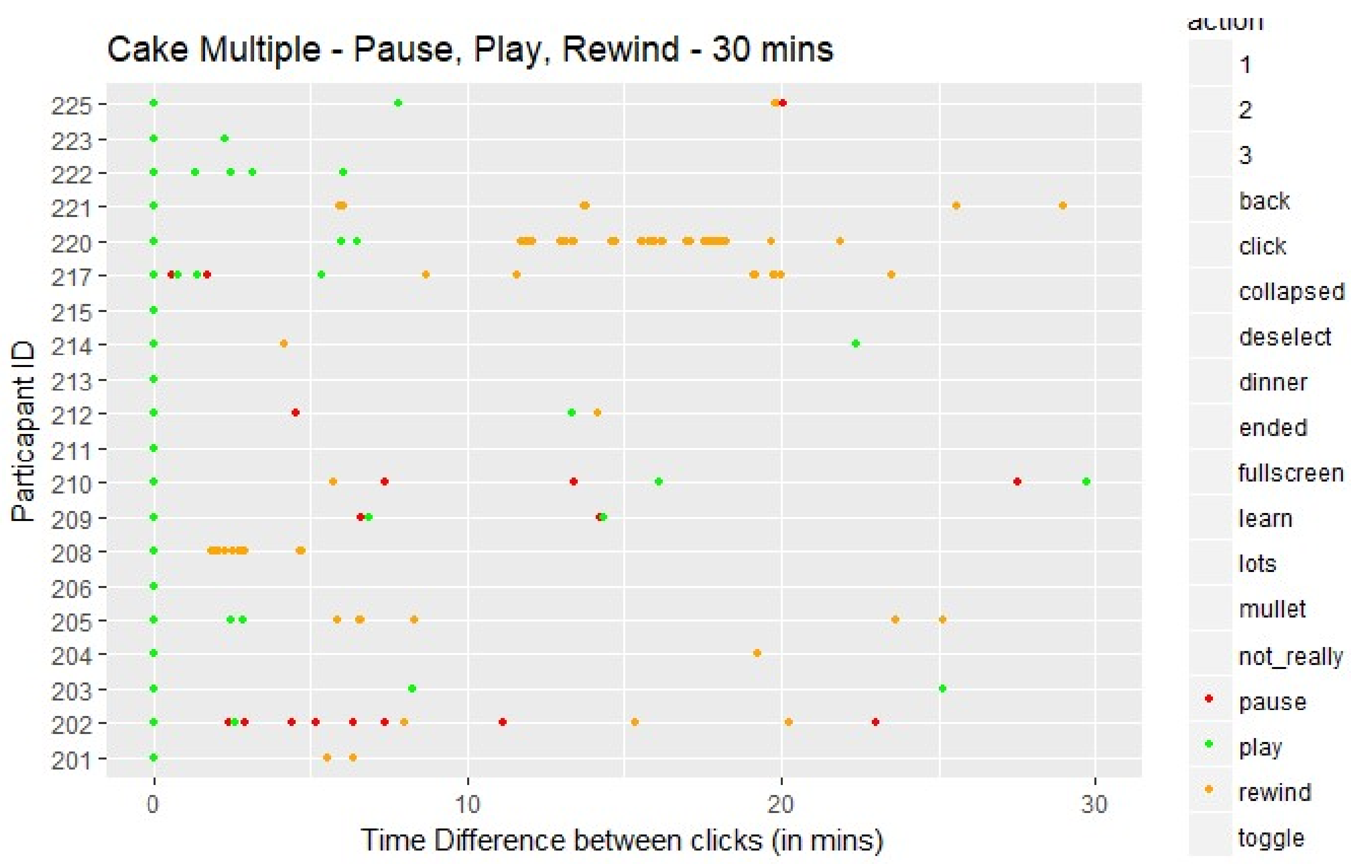
Filtering Individual Actions With Colour

It would be more interesting having them all on one graph and giving each a colour, so that it's possible to have all three action son one plot.

`r` r

Hide

```
plot(combined_colour)
```



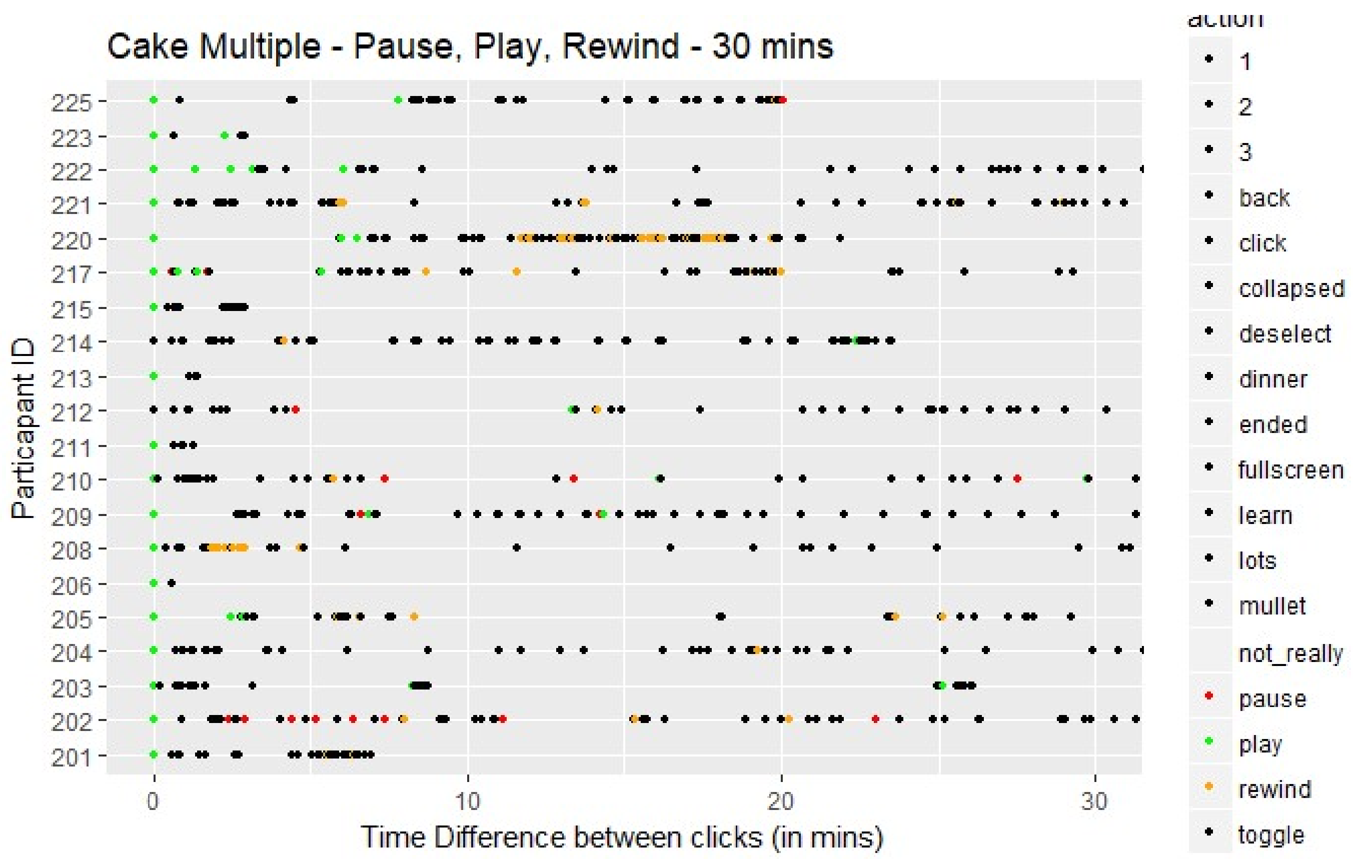
The colours are hard to read, and if you look at the code in filter_action.R the code is messy too. I have assigned colours manually to play, plause and rewind.

To show a greater context I wanted to add all the other clicks but have them black.

`r` r

Hide

```
plot(combined_colour_black)
```

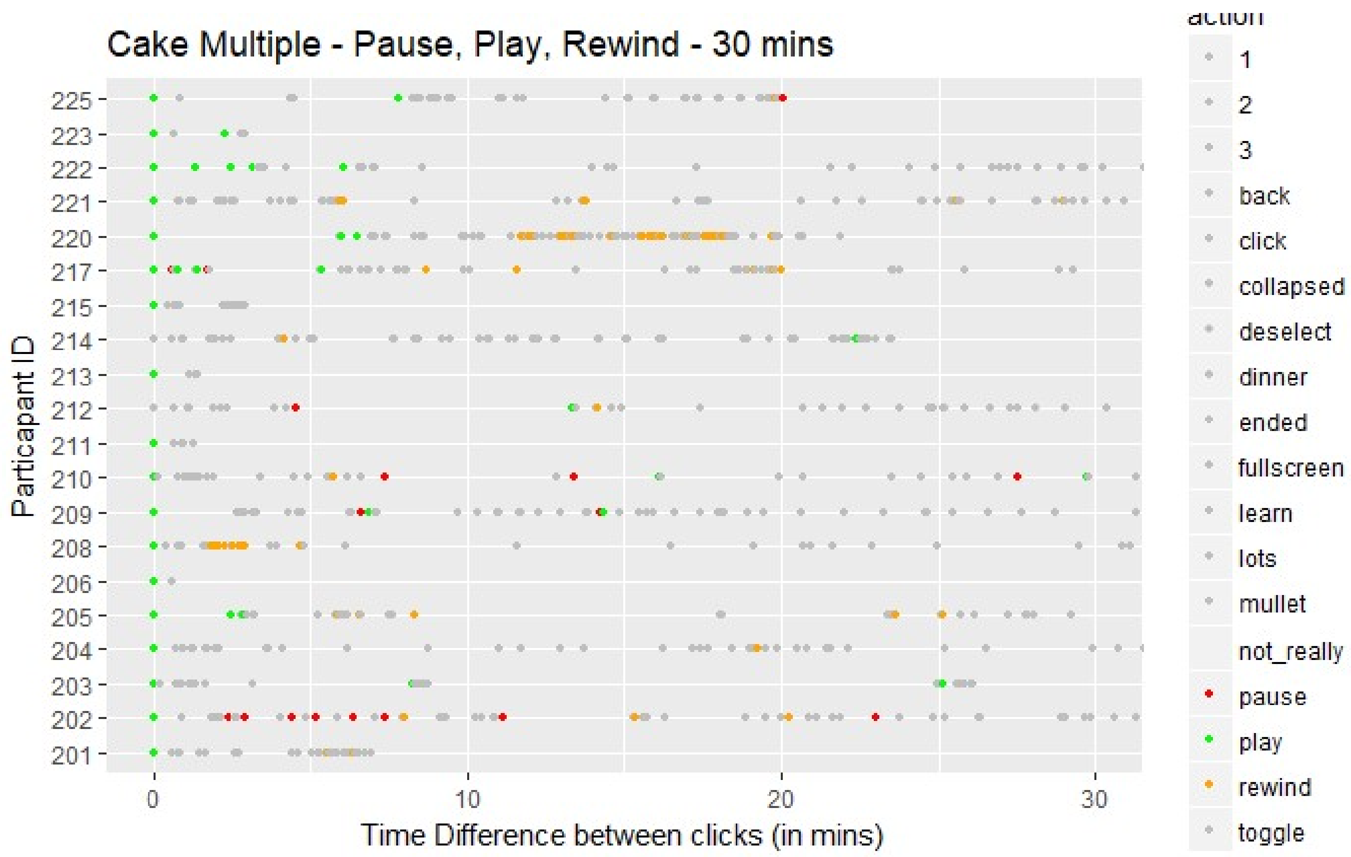


It looks messy and difficult to use. A different colour scheme need sto be employed to make it easier to read.

`r` r

Hide

```
plot(combined_colour_grey)
```

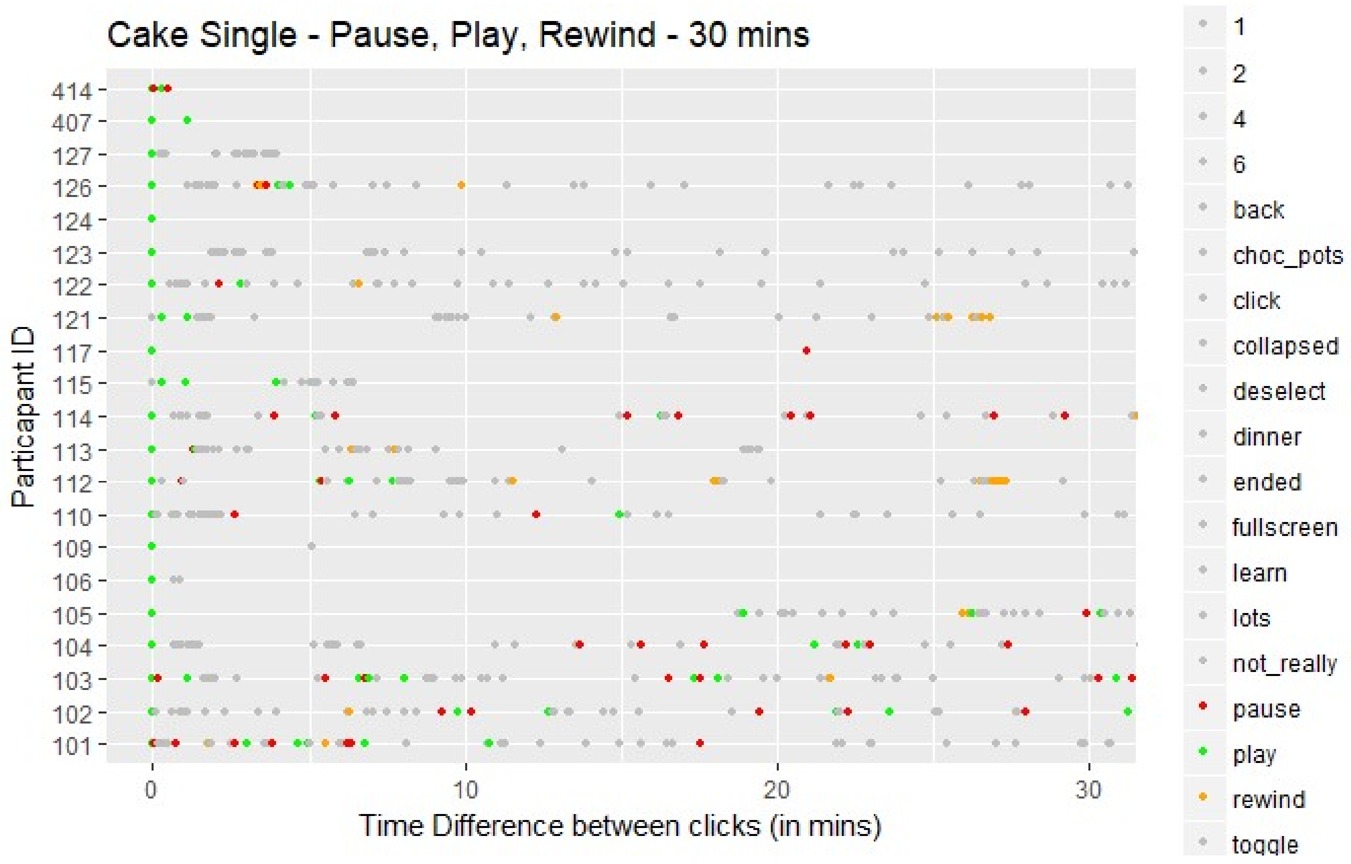


The grey makes it easier to read and easier on the eyes. This is perhaps a good place to start for how to think of doing a interactive visualisation, where each action can be given an individual colour.

`r` r

Hide

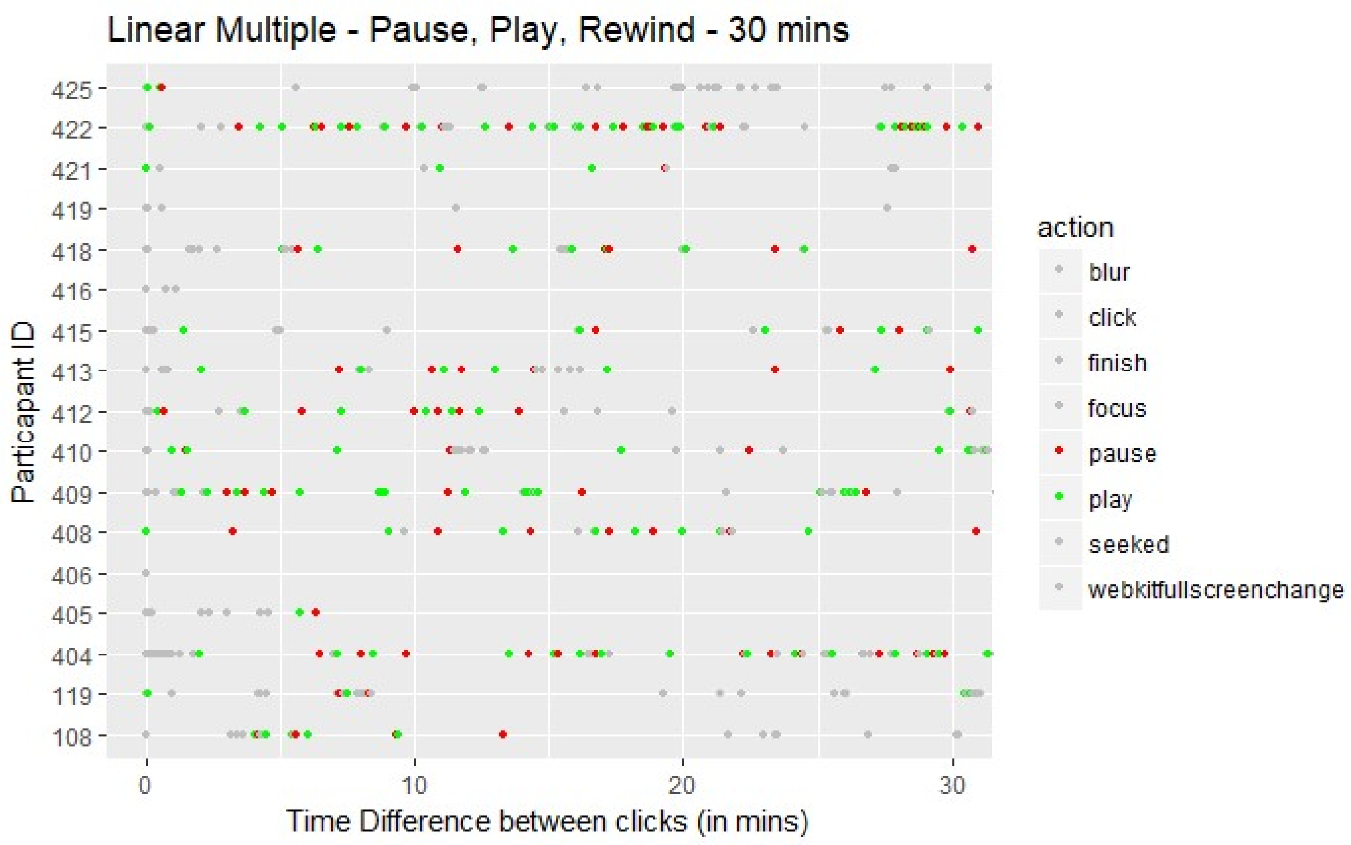
```
plot(combined_colour_grey_two)
```



r r

Hide

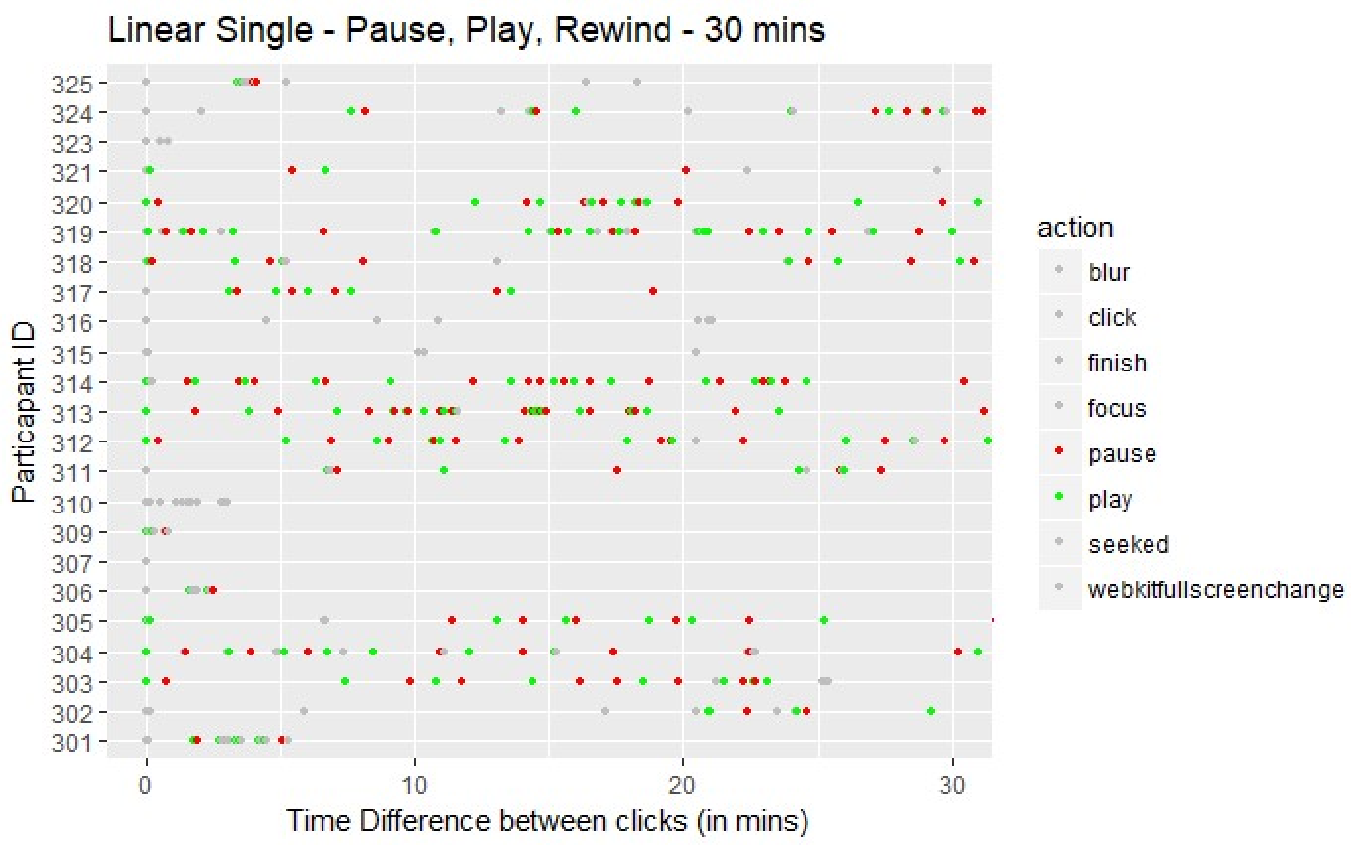
```
plot(combined_colour_grey_three)
```



`r` r

Hide

```
plot(combined_colour_grey_four)
```



Streamling the Plot Process

After consulting David Mawdsley for some help streamlininig the code he suggested some changes such as using piping for plotting the graphs as well as a function that can plot the information with the parameters that we give it. David also mentioned that it will make it easier to create interactive plots using a package called Shiny.

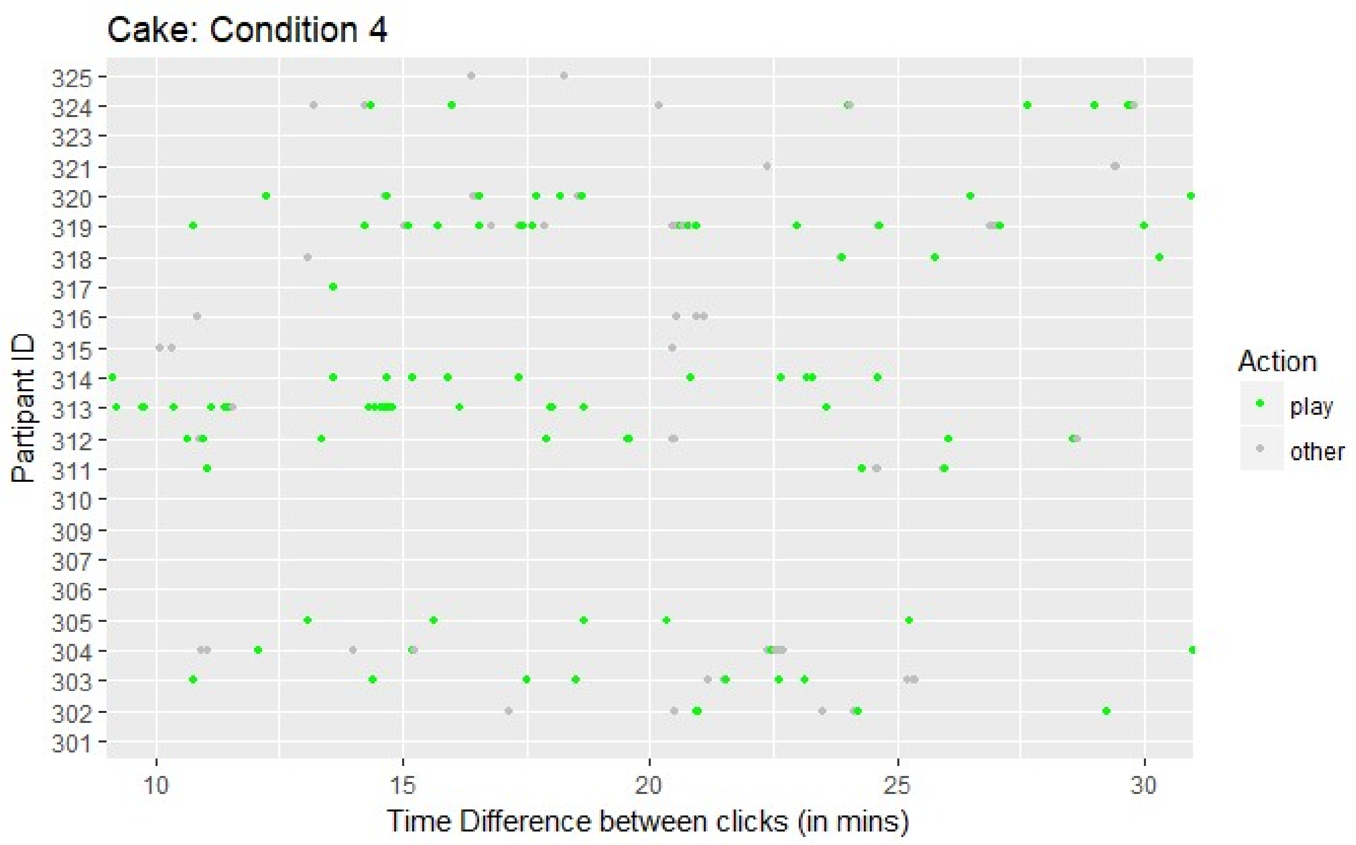
NOTE: The orginal function was written in “src/filter_action”. It has since been seperated and stored in “src/functions/” to help organisation, reproducability etc

I modified the function so it can accept a start and end time in minutes so it can look at any length of time. I set the parameters to look at the the actions between 10 and 30 mins

`r` r

Hide

```
plot(example)
```

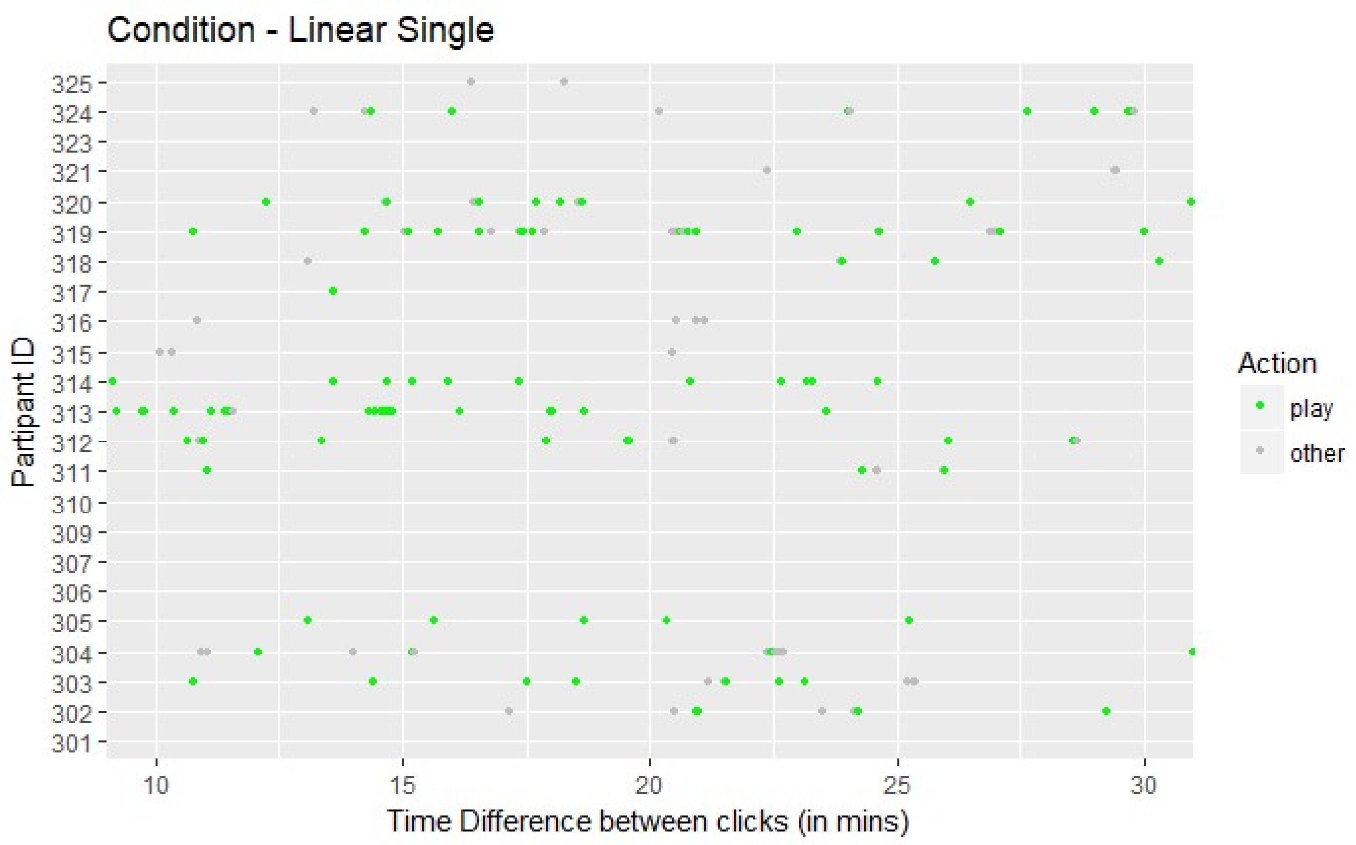


I modified the function to make the Title more descriptive. Instead of CAKE Condition 4, it says Condition - Linear Single

`r` r

Hide

```
plot(example_two)
```



With this function it will take out some of the legwork involved in searching for individual actions ands plotting them against all other types of actions.

I rigged the function again so that the mutatiion of action_simplified is inside the function, rather than outside. I have updated the seperate function script as well.

Currently the type of actions shown are hard coded into the function so they match a certain colour. This allows for consistency, but means we are limited in the number of actions by number of distinct colours. Ideally they'd be automatically matched but with the sheer number of actions it means the colours will look similar, making it hgarder to pick out individual actions.