In-Class Exercise 7: InteractiveVis with R

Dr. Kam Tin Seong
Assoc. Professor of Information Systems
School of Computing and Information Systems,
Singapore Management University

2020-2-15 (updated: 2021-07-04)

Interactive with R - A tile of Two Packages

- ggiraph
- plotlyr

Getting Started

Write a code chunk to check, install and launch **ggiraph**, **plotly**, **DT** and **tidyverse** packages of R

The solution:

```
packages = c('ggiraph', 'plotly',
             'DT', 'patchwork',
             'tidyverse')
for (p in packages){
  if(!require(p, character.only = T)){
    install.packages(p)
  }
  library(p,character.only = T)
}
```

Importing Data

Using *read_csv()* (https://readr.tidyverse.org/reference/read_delim.html) of **readr** package, import *gps.csv* into R.

The solution:

```
exam_data <- read_csv("data/Exam_data.csv")
glimpse(exam_data)
```

```
## Rows: 322
## Columns: 7
## $ ID      <chr> "Student321", "Student30
5", "Student289", "Student227", "Stude~
## $ CLASS   <chr> "3I", "3I", "3H", "3F", "3
I", "3I", "3I", "3I", "3H", "3~
## $ GENDER  <chr> "Male", "Female", "Male",
 "Male", "Male", "Female", "Male", "M~
## $ RACE    <chr> "Malay", "Malay", "Chines
e", "Chinese", "Malay", "Malay", "Chi~
## $ ENGLISH <dbl> 21, 24, 26, 27, 27, 31, 3
1, 31, 33, 34, 34, 36, 36, 36, 37, 38~
## $ MATHS   <dbl> 9, 22, 16, 77, 11, 16, 21,
18, 19, 49, 39, 35, 23, 36, 49, 30,~
## $ SCIENCE <dbl> 15, 16, 16, 31, 25, 16, 2
5, 27, 15, 37, 42, 22, 32, 36, 35, 45~
```

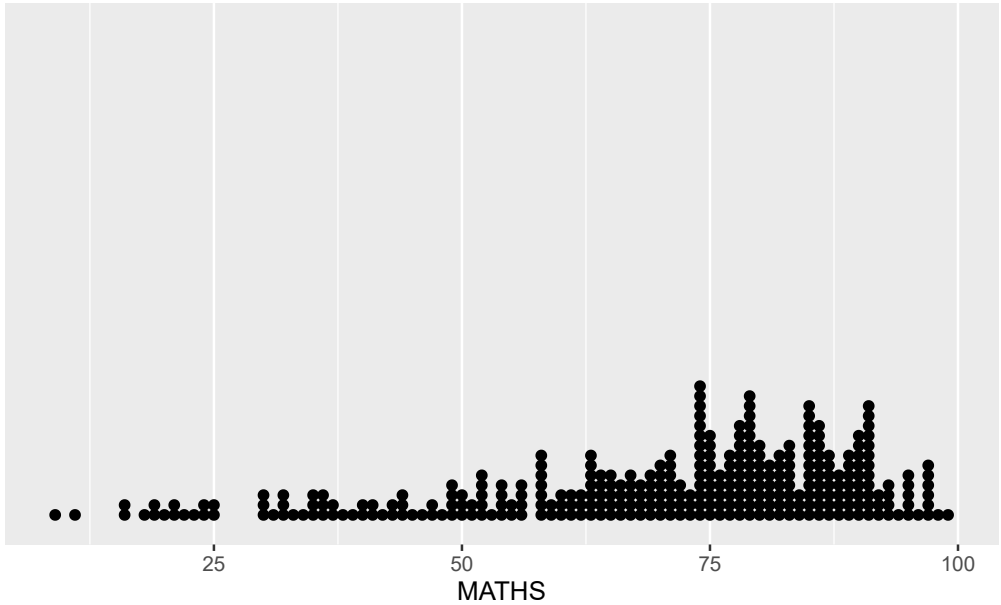
Interactive Data Visualisation - ggiraph methods

- An htmlwidget and a ggplot2 extension. It allows ggplot graphics to be interactive.
- Interactive is made with ggplot geometries that can understand three arguments:
  - **Tooltip**: a column of data-sets that contain tooltips to be displayed when the mouse is over elements.
  - **Onclick**: a column of data-sets that contain a JavaScript function to be executed when elements are clicked.
  - **Data_id**: a column of data-sets that contain an id to be associated with elements.
- If it used within a shiny application, elements associated with an id (data_id) can be selected and manipulated on client and server sides.

Reference: ggiraph (https://davidgohel.github.io/ggiraph/index.html) package

Tooltip effect with *tooltip* aesthetic

```
p <- ggplot(data=exam_data,
       aes(x = MATHS)) +
  geom_dotplot_interactive(
    aes(tooltip = ID),
    stackgroups = TRUE,
    binwidth = 1,
    method = "histodot") +
  scale_y_continuous(NULL,
                     breaks = NULL)
girafe(
  ggobj = p,
  width_svg = 6,
  height_svg = 6*0.618
)
```



Interactivity: hovering displays student's ID

Comparing ggplot2 and ggiraph codes

The original ggplot2 code chunk.

```
ggplot(data=exam_data,
       aes(x = MATHS)) +
  geom_dotplot(binwidth=2.5,
               dotsize = 0.5) +
  scale_y_continuous(NULL,
                     breaks = NULL)
```

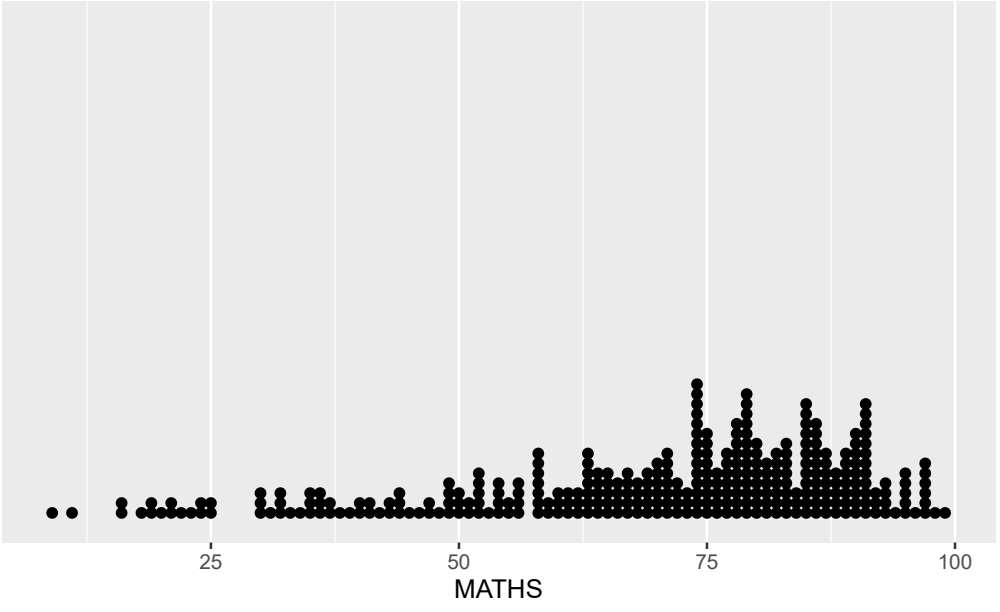
The ggiraph code chunk.

```
p <- ggplot(data=exam_data,
       aes(x = MATHS)) +
  geom_dotplot_interactive(
    aes(tooltip = ID),
    stackgroups = TRUE,
    binwidth = 1,
    method = "histodot") +
  scale_y_continuous(NULL,
                     breaks = NULL)
girafe(
  ggobj = p,
  width_svg = 6,
  height_svg = 6*0.618
)
```

A complete list of geometries supported by ggiraph and their corresponding command syntax can be found here (https://davidgohel.github.io/ggiraph/reference/index.html).

Hover effect with *data_id* aesthetic

```
p <- ggplot(data=exam_data,
       aes(x = MATHS)) +
  geom_dotplot_interactive(aes(data_id = CL
ASS),
    stackgroups = TRUE,
    binwidth = 1,
    method = "histodot") +
  scale_y_continuous(NULL,
                     breaks = NULL)
girafe(
  ggobj = p,
  width_svg = 6,
  height_svg = 6*0.618
)
```



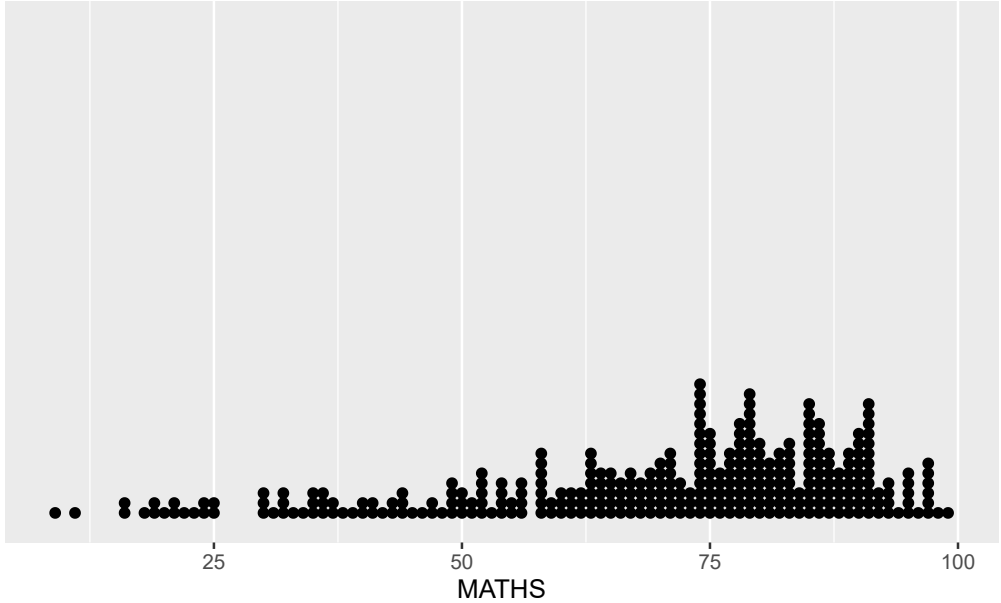
Interactivity: Elements associated with a *data_id* (i.e CLASS) will be highlighted upon mouse over.

Note that the default value of the hover css is *hover_css = "fill:orange;"* .

Styling hover effect

In the code chunk below, css codes are used to change the highlighting effect.

```
p <- ggplot(data=exam_data,
       aes(x = MATHS)) +
  geom_dotplot_interactive(
    aes(data_id = CLASS),
    stackgroups = TRUE,
    binwidth = 1,
    method = "histodot") +
  scale_y_continuous(NULL,
                     breaks = NULL)
girafe(
  ggobj = p,
  width_svg = 6,
  height_svg = 6*0.618,
  options = list(
    opts_hover(css = "fill: #202020;"),
    opts_hover_inv(css = "opacity:0.2;")
  )
)
```

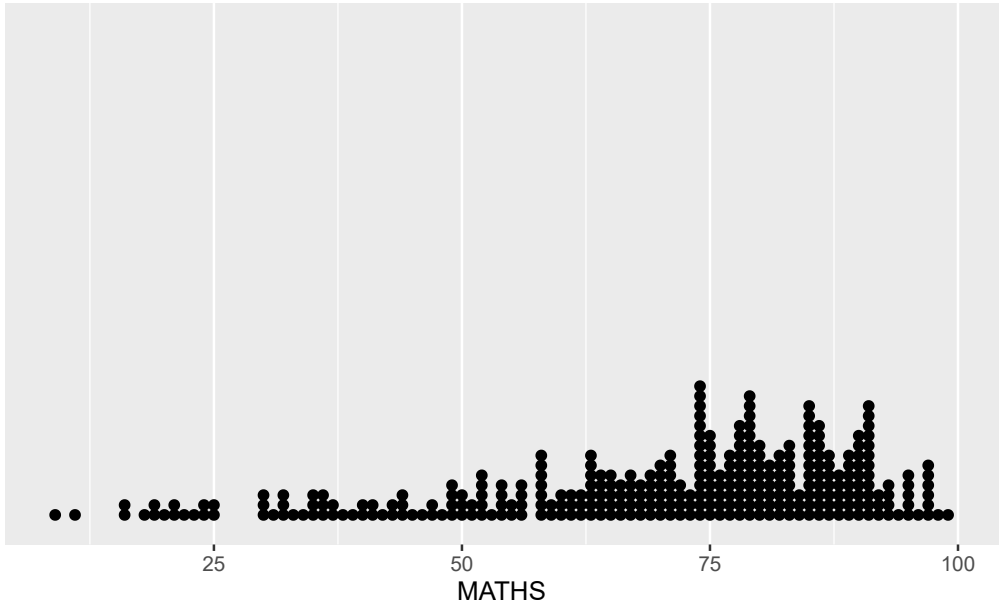


Interactivity: Elements associated with a *data_id* (i.e CLASS) will be highlighted upon mouse over.

Click effect with onclick

```
exam_data$onclick <- sprintf("window.open
(\"%s%s\")",
"https://www.moe.gov.sg/schoolfinder?journe
y=Primary%20school", as.character(exam_data
$ID) )

p <- ggplot(data=exam_data,
       aes(x = MATHS)) +
  geom_dotplot_interactive(
    aes(onclick = onclick),
    stackgroups = TRUE,
    binwidth = 1,
    method = "histodot") +
  scale_y_continuous(NULL,
                     breaks = NULL)
girafe(
  ggobj = p,
  width_svg = 6,
  height_svg = 6*0.618)
```

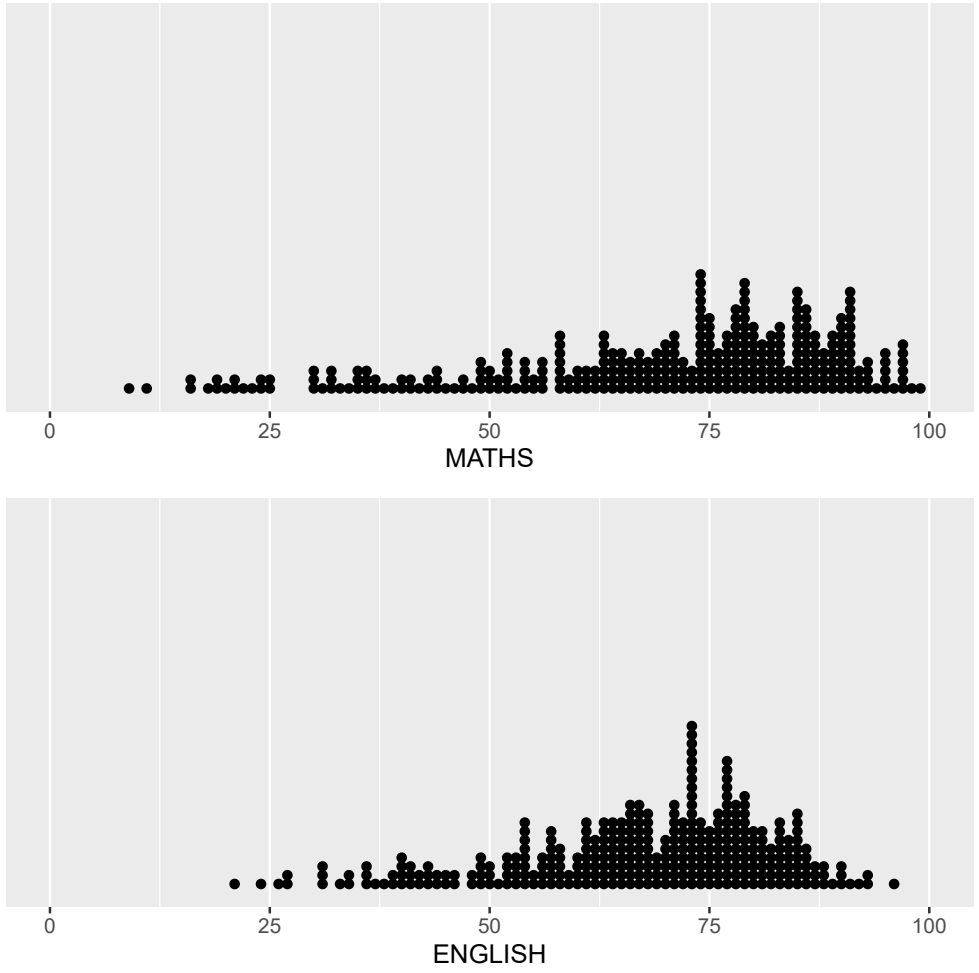


Interactivity: Web document link with a data object will be displayed on the web browser upon mouse click.

Coordinated Multiple Views with ggiraph

Coordinated multiple views methods has been implemented in the data visualisation on the right.

- when a data point of one of the dotplot is selected, the corresponding data point ID on the second data visualisation will be highlighted too.



Coordinated Multiple Views with ggiraph

In order to build a coordinated multiple views, the following programming strategy will be used:

1. Appropriate interactive functions of **ggiraph** will be used to create the multiple views.
2. *patchwork* function of patchwork (https://patchwork.data-imaginist.com/) package will be used inside girafe function to create the interactive coordinated multiple views.

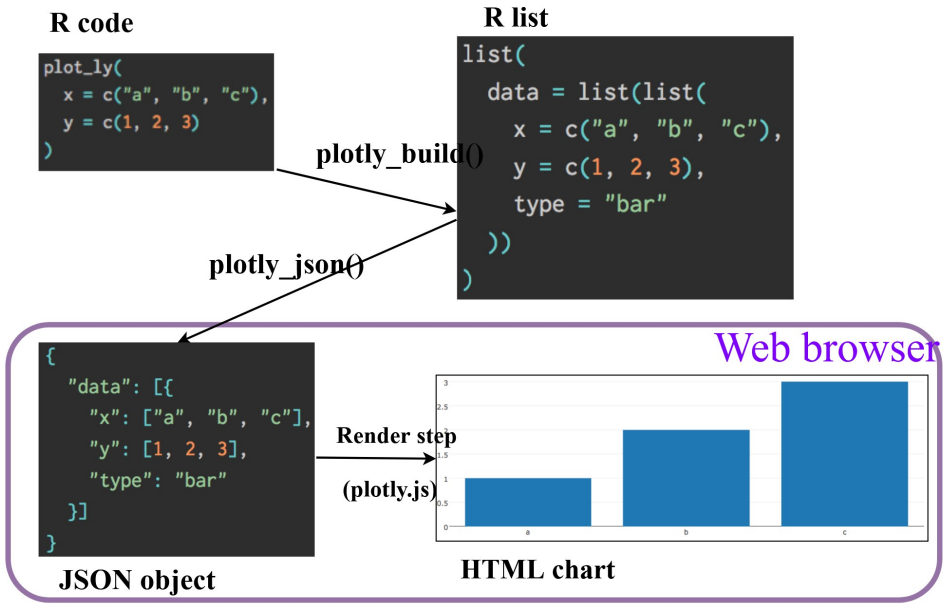
```
p1 <- ggplot(data=exam_data,
       aes(x = MATHS)) +
  geom_dotplot_interactive(
    aes(data_id = ID),
    stackgroups = TRUE,
    binwidth = 1,
    method = "histodot") +
  coord_cartesian(xlim=c(0,100)) +
  scale_y_continuous(NULL,
                     breaks = NULL)
```

```
p2 <- ggplot(data=exam_data,
       aes(x = ENGLISH)) +
  geom_dotplot_interactive(
    aes(data_id = ID),
    stackgroups = TRUE,
    binwidth = 1,
    method = "histodot") +
  coord_cartesian(xlim=c(0,100)) +
  scale_y_continuous(NULL,
                     breaks = NULL)

girafe(code = print(p1 / p2),
       width_svg = 6,
       height_svg = 6,
       options = list(
         opts_hover(css = "fill: #202020;"
),
         opts_hover_inv(css = "opacity:0.
2;")
         )
       )
```

Interactive Data Visualisation - plotly methods!

- Plotly's R graphing library create interactive web graphics from **ggplot2** graphs and/or a custom interface to the (MIT-licensed) JavaScript library **plotly.js** (https://plotly.com/javascript/) inspired by the grammar of graphics.
- Different from other plotly platform, plot.R is free and open source.



There are two ways to create interactive graph by using plotly, they are:

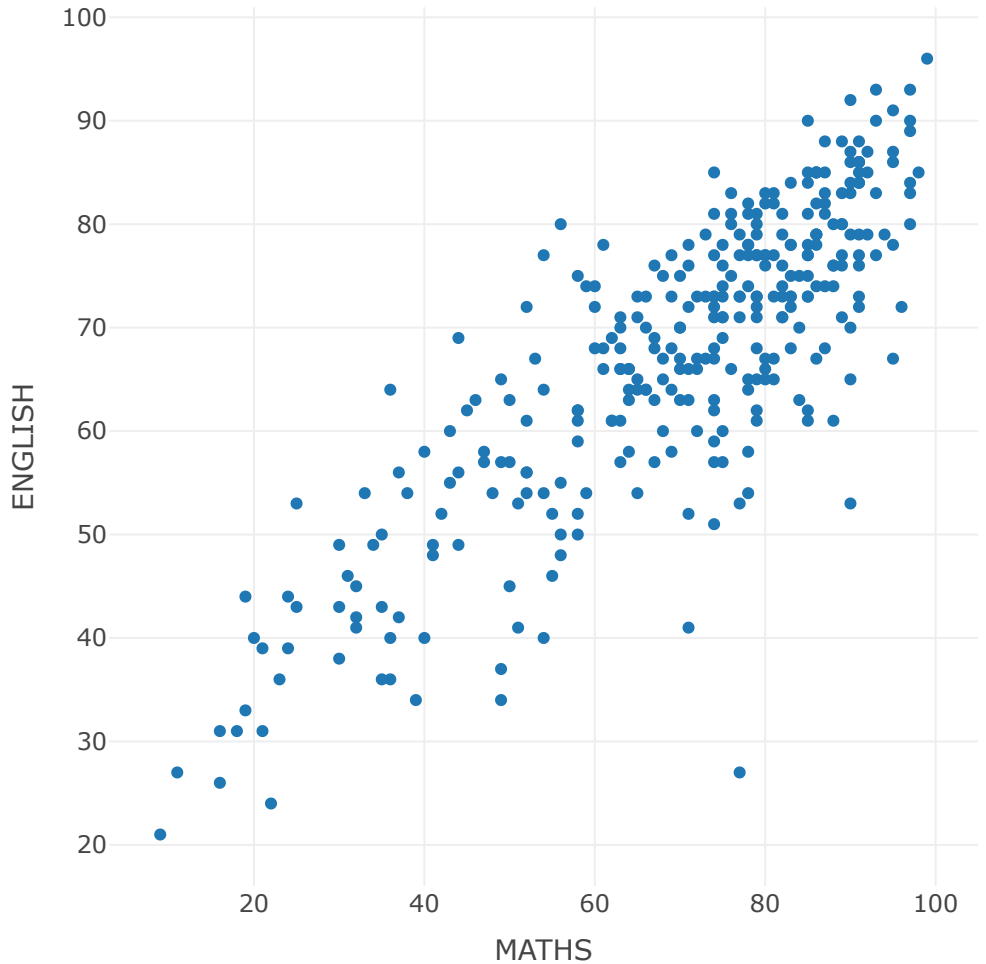
- by using *plot_ly()*, and
- by using *ggplotly()*

Creating an interactive scatter plot: plot_ly() method

The code chunk below plots an interactive scatter plot by using *plot_ly()*.

```
plot_ly(data = exam_data,
             x = ~MATHS,
             y = ~ENGLISH)
```

The output:



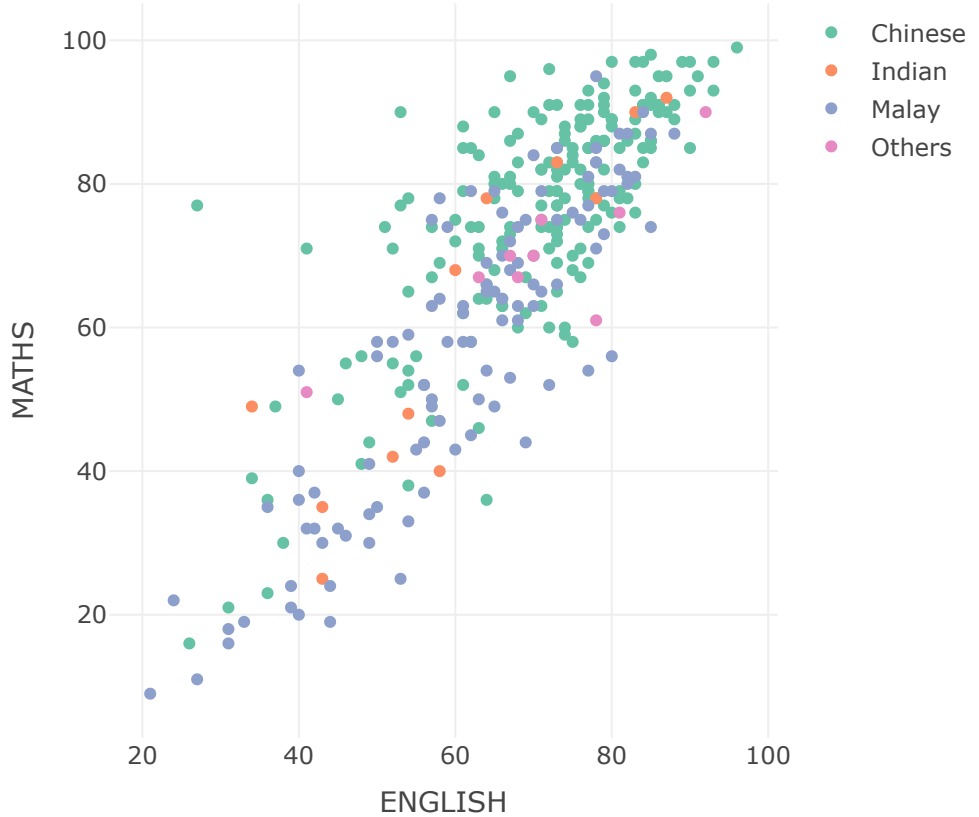
Working with visual variable: plot_ly() method

In the code chunk below, *color* argument is mapped to a qualitative visual variable (i.e. RACE).

```
plot_ly(data = exam_data,
        x = ~ENGLISH,
        y = ~MATHS,
        color = ~RACE)
```

Interactive:

- Click on the colour symbol at the legend.



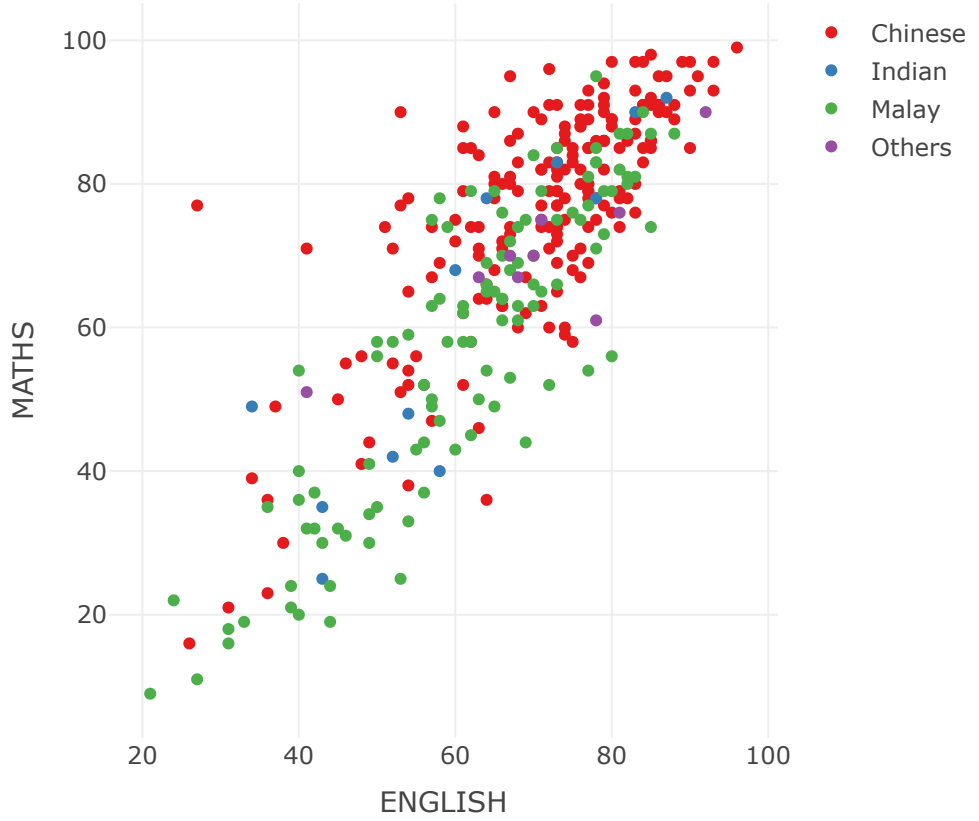
Changing colour pallete: plot_ly() method

In the code chunk below, *colors* argument is used to change the default colour palette to ColorBrewel (https://www.r-graph-gallery.com/38-rcolorbrewers-palettes.html) colour palette.

```
plot_ly(data = exam_data,
        x = ~ENGLISH,
        y = ~MATHS,
        color = ~RACE,
        colors = "Set1")
```

Interactive:

- Click on the colour symbol at the legend.



Customising colour scheme: plot_ly() method

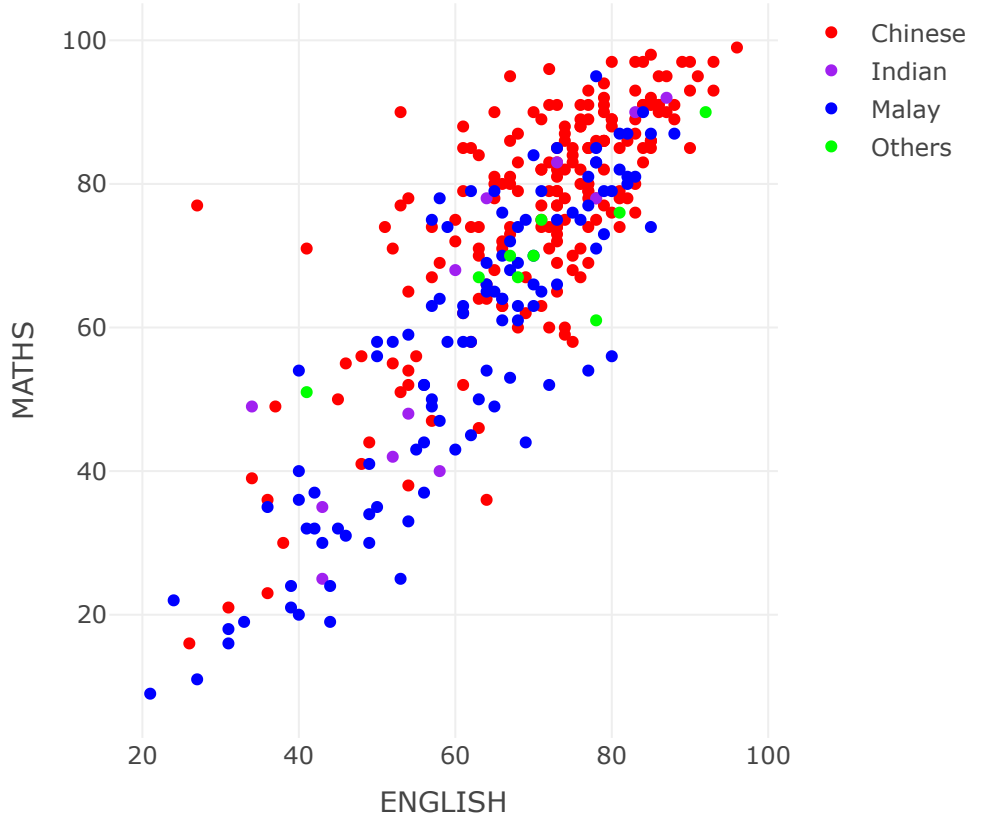
In the code chunk below, a customised colour scheme is created. Then, *colors* argument is used to change the default colour palette to the customised colour scheme.

```
pal <- c("red", "purple", "blue", "green")

plot_ly(data = exam_data,
        x = ~ENGLISH,
        y = ~MATHS,
        color = ~RACE,
        colors = pal)
```

Interactive:

- Click on the colour symbol at the legend.



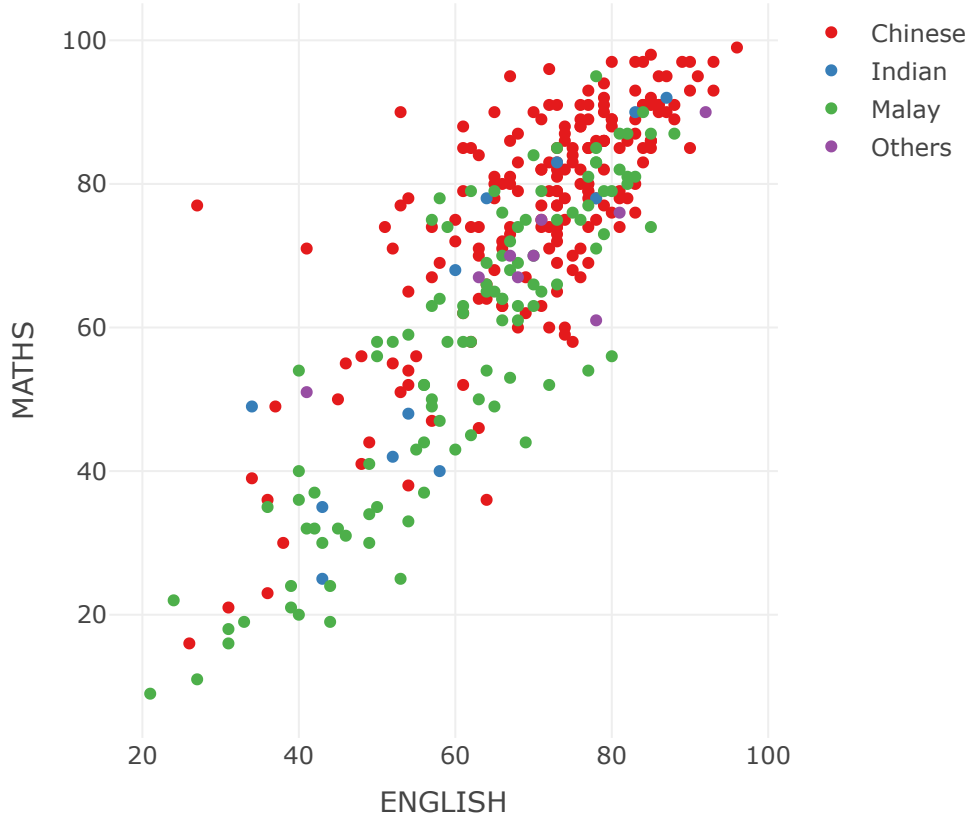
Customising tooltip: plot_ly() method

In the code chunk below, *text* argument is used to change the default tooltip.

```
plot_ly(data = exam_data,
        x = ~ENGLISH,
        y = ~MATHS,
        text = ~paste("Student ID:", ID,
                      "<br>Class:", CLASS),
        color = ~RACE,
        colors = "Set1")
```

Interactive:

- Click on the colour symbol at the legend.



Working with layout: plot_ly() method

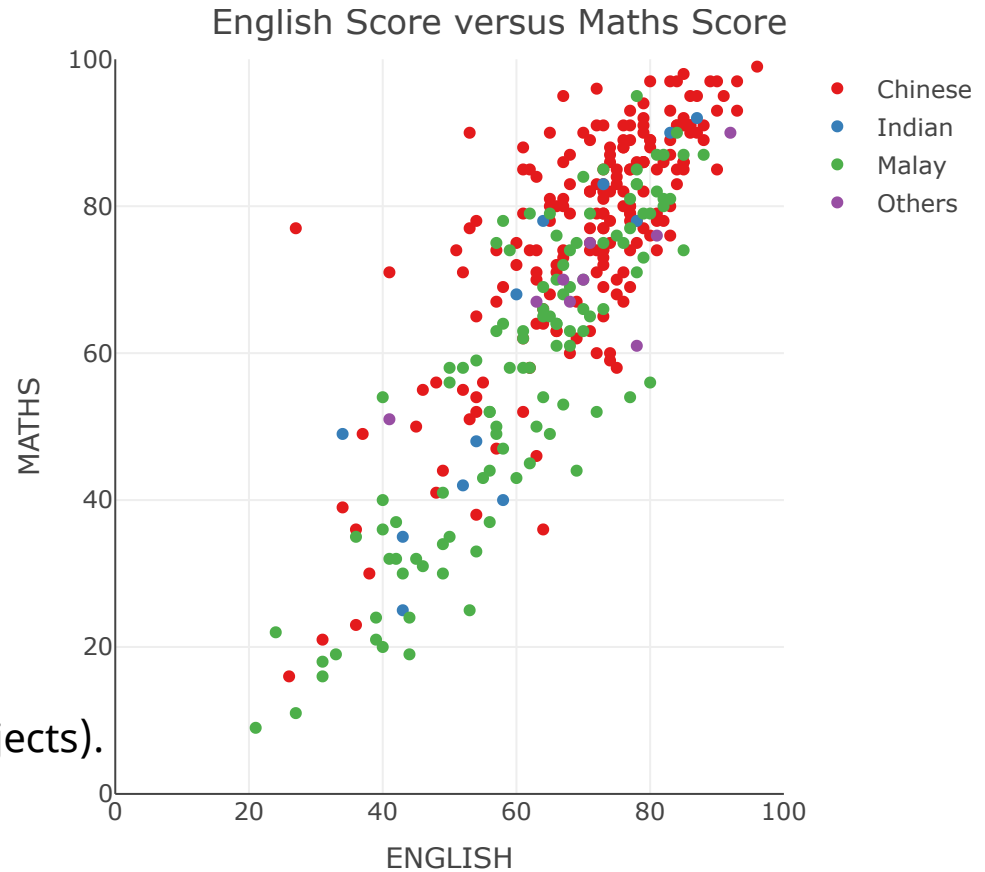
In the code chunk below, *layout* argument is used to change the default tooltip.

```
plot_ly(data = exam_data,
        x = ~ENGLISH,
        y = ~MATHS,
        text = ~paste("Student ID:", ID,
                      "<br>Class:", CLASS),
        color = ~RACE,
        colors = "Set1") %>%
  layout(title = 'English Score versus Math
s Score ',
         xaxis = list(range = c(0, 100)),
         yaxis = list(range = c(0, 100)))
```

To learn more about layout, visit this link (https://plotly.com/r/reference/#Layout_and_layout_style_objects).

Interactive:

- Click on the colour symbol at the legend.

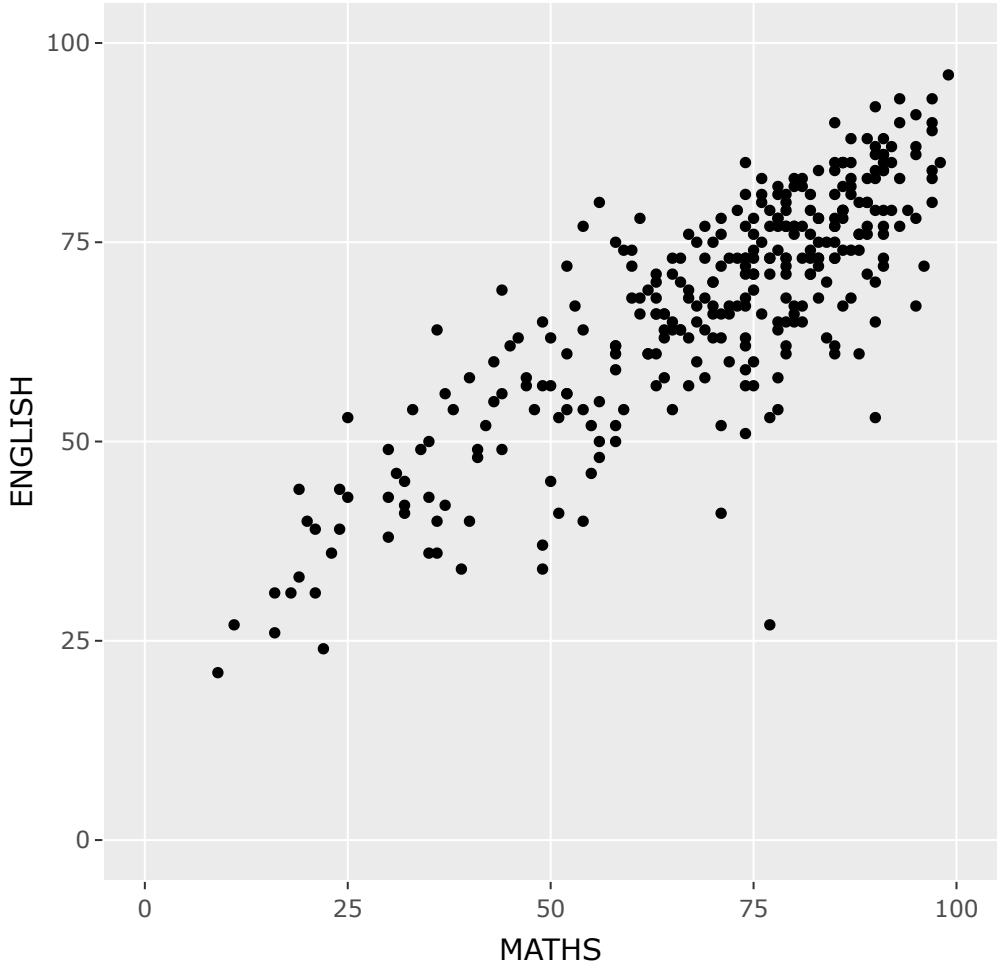


Creating an interactive scatter plot: ggplotly() method

The code chunk below plots an interactive scatter plot by using *ggplotly()*.

```
p <- ggplot(data=exam_data,
            aes(x = MATHS,
                y = ENGLISH)) +
  geom_point(dotsize = 1) +
  coord_cartesian(xlim=c(0,100),
                  ylim=c(0,100))
ggplotly(p)
```

Notice that the only extra line you need to include in the code chunk is *ggplotly()*.



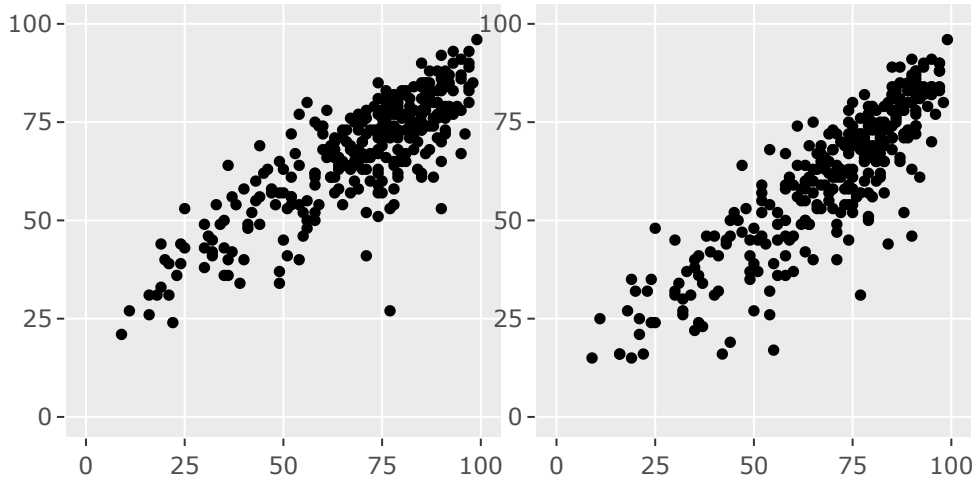
Coordinated Multiple Views with plotly

Code chunk below plots two scatterplots and places them next to each other side-by-side by using *subplot()* (https://plotly.com/r/subplots/) of **plotly** package.

```
p1 <- ggplot(data=exam_data,
               aes(x = MATHS,
                   y = ENGLISH)) +
  geom_point(size=1) +
  coord_cartesian(xlim=c(0,100),
                  ylim=c(0,100))

p2 <- ggplot(data=exam_data,
             aes(x = MATHS,
                 y = SCIENCE)) +
  geom_point(size=1) +
  coord_cartesian(xlim=c(0,100),
                  ylim=c(0,100))
subplot(ggplotly(p1),
        ggplotly(p2))
```

The side-by-side scatterplots.



Notice that these two scatter plots are not linked.

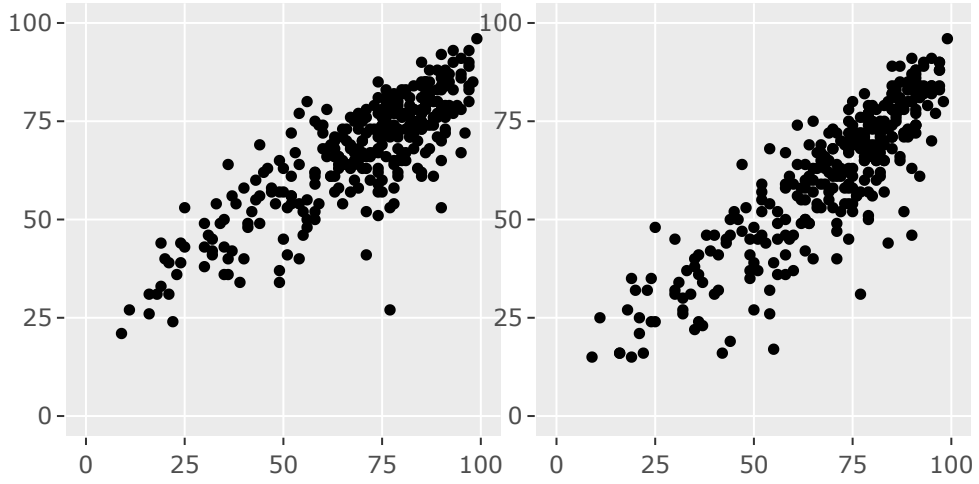
Coordinated Multiple Views with plotly

To create a coordinated scatterplots, *highlight_key()* of **plotly** package is used.

```
d <- highlight_key(exam_data)
p1 <- ggplot(data=d,
             aes(x = MATHS,
                 y = ENGLISH)) +
  geom_point(size=1) +
  coord_cartesian(xlim=c(0,100),
                  ylim=c(0,100))

p2 <- ggplot(data=d,
             aes(x = MATHS,
                 y = SCIENCE)) +
  geom_point(size=1) +
  coord_cartesian(xlim=c(0,100),
                  ylim=c(0,100))
subplot(ggplotly(p1),
        ggplotly(p2))
```

Click on a data point of one of the scatterplot and see how the corresponding point on the other scatterplot is selected.



Thing to learn from the code chunk:

- *highlight_key()* simply creates an object of class crosstalk::SharedData.
- Visit this link (https://rstudio.github.io/crosstalk/) to learn more about crosstalk,

Interactive Data Table: DT package

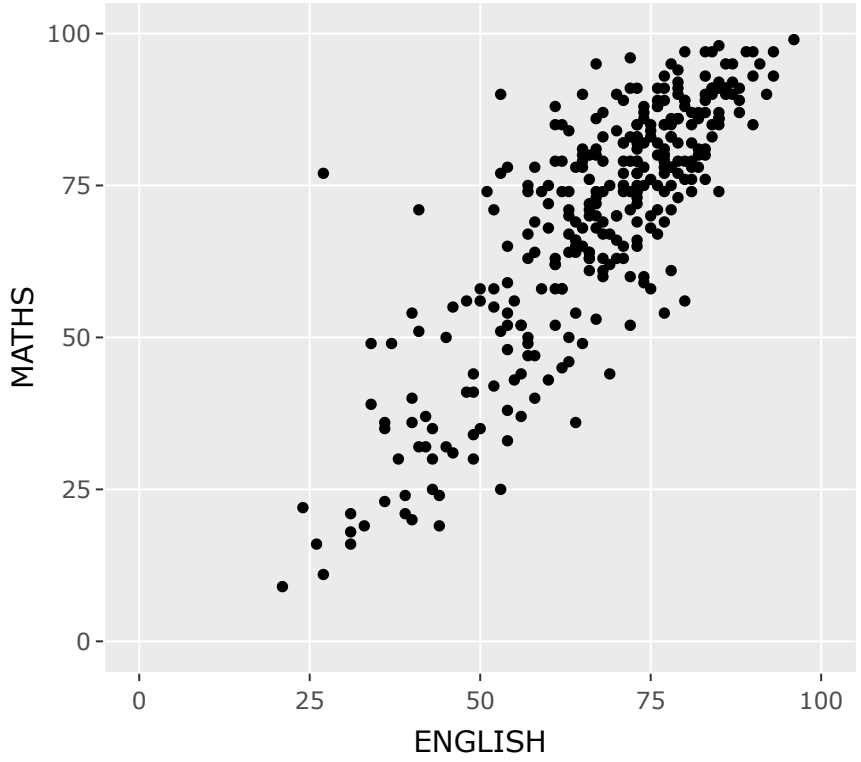
- A wrapper of the JavaScript Library DataTables (https://datatables.net/)
- Data objects in R can be rendered as HTML tables using the JavaScript library 'DataTables' (typically via R Markdown or Shiny).

```
DT::datatable(exam_data)
```

Show 10 **entries** **Search:**

	ID	CLASS	GENDER	RACE	ENGLISH	MATHS	SCIENCE	onclick
1	Student321	3I	Male	Malay	21	9	15	window.open("https://www.moe journey=Primary%20schoolStude
2	Student305	3I	Female	Malay	24	22	16	window.open("https://www.moe journey=Primary%20schoolStude
3	Student289	3H	Male	Chinese	26	16	16	window.open("https://www.moe journey=Primary%20schoolStude
4	Student227	3F	Male	Chinese	27	77	31	window.open("https://www.moe journey=Primary%20schoolStude

Linked brushing: crosstalk method



Show 10 entries Search:

	ID	CLASS	GENDER	RACE	ENGLISH	MATHS
1	Student321	3I	Male	Malay	21	
2	Student305	3I	Female	Malay	24	2
3	Student289	3H	Male	Chinese	26	1
4	Student227	3F	Male	Chinese	27	7
5	Student318	3I	Male	Malay	27	1
6	Student306	3I	Female	Malay	31	1
7	Student313	3I	Male	Chinese	31	2

Linked brushing: crosstalk method

Code chunk below is used to implement the coordinated brushing shown on Slide 17.

```
d <- highlight_key(exam_data)
p <- ggplot(d,
            aes(ENGLISH,
                MATHS)) +
  geom_point(size=1) +
  coord_cartesian(xlim=c(0,100),
                  ylim=c(0,100))

gg <- highlight(ggplotly(p),
                "plotly_selected")

crosstalk::bscols(gg,
                  DT::datatable(d),
                  widths = 5)
```

Things to learn from the code chunk:

- *highlight()* is a function of **plotly** package. It sets a variety of options for brushing (i.e., highlighting) multiple plots. These options are primarily designed for linking multiple plotly graphs, and may not behave as expected when linking plotly to another htmlwidget package via crosstalk. In some cases, other htmlwidgets will respect these options, such as persistent selection in leaflet.
- *bscols()* is a helper function of **crosstalk** package. It makes it easy to put HTML elements side by side. It can be called directly from the console but is especially designed to work in an R Markdown document. **Warning:** This will bring in all of Bootstrap!.

Reference

ggiraph

This link (https://davidgohel.github.io/ggiraph/index.html) provides online version of the reference guide and several useful articles. Use this link (https://cran.r-project.org/web/packages/ggiraph/ggiraph.pdf) to download the pdf version of the reference guide.

- How to Plot With Ggiraph (https://www.r-bloggers.com/2018/04/how-to-plot-with-ggiraph/)
- Interactive map of France with ggiraph (http://rstudio-pubs-static.s3.amazonaws.com/152833_56a4917734204de7b37881d164cf8051.html)
- Custom interactive sunbursts with ggplot in R (https://www.pipinghotdata.com/posts/2021-06-01-custom-interactive-sunbursts-with-ggplot-in-r/)
- This link (https://github.com/d-qn/2016_08_02_rioOlympicsAthletes) provides code example on how ggiraph is used to interactive graphs for Swiss Olympians - the solo specialists (https://www.swissinfo.ch/eng/rio-2016-_swiss-olympians---the-solo-specialists-/42349156?utm_content=bufferd148b&utm_medium=social&utm_source=twitter.com&utm_campaign=buffer).

plotly for R

- Getting Started with Plotly in R (https://plotly.com/r/getting-started/)
    - A collection of plotly R graphs are available via this link (https://plotly.com/r/).
- Carson Sievert (2020) **Interactive web-based data visualization with R, plotly, and shiny**, Chapman and Hall/CRC is the best resource to learn plotly for R. The online version is available via this link (https://plotly-r.com/)