Interactive Visualization with R - Interactive Plots - 1

One should look for what is and not what he thinks should be. (Albert Einstein)

Interactive plots: topic introduction

In this part of the course, we will cover the following concepts:

- Create correlation plots, column plots, box plots
- Save and view interactive plots by using `htmlwidgets` library

Warm-up trivia

- Before moving forward, let’s review the series used in highcharter
- Below are 2 columns:
  - Column A shows the highcharter series type
  - Column B is the plot type
- Pair the correct series type and plot type, and share your answers in chat

Column A	Column B
Highcharter series type	Plot type
1 - `column`	a - density
2 - `area`	b - bar plot
3 - `bar`	c - horizontal bar plot

Review trivia answers

- `column` is for creating a bar plot (1-b)
- `area` is for a density plot (2-a)
- `bar` is for creating a horizontal bar plot (3-c)

Module completion checklist

Objective	Complete
Construct and save a boxplot and a column plot with hchart	
Visualize a correlation plot with hchart	

Creating plots with highcharter

- We will work with the healthcare stroke dataset to create interactive visualizations
- We will start with a boxplot to visualize the distribution of bmi based on the smoking status
- We will then make a multiple-column plot to help compare different summary statistics by variable
- Finally, we will create a correlation plot to assess the strength of the relationships between variables

Directory settings

- In order to maximize the efficiency of your workflow, use the `box` package and encode your directory structure into `variables`
- Let the `main_dir` be the variable corresponding to your materials folder

```r
# Set `main_dir` to the location of your materials folder.

path = box::file()
main_dir = dirname(dirname(path))
```

Directory settings (cont'd)

- We will store all datasets in the `data` directory inside the materials folder in your environment; hence we will save their path to a `data_dir` variable
- We will save all the plots in the `plots` directory corresponding to `plot_dir` variable

- To append one string to another, use `paste0` command and pass the strings you would like to paste together

```
# Make `data_dir` from the `main_dir` and
# remainder of the path to data directory.
data_dir = paste0(main_dir, "/data")
# Make `plots_dir` from the `main_dir` and
# remainder of the path to plots directory.
plot_dir = paste0(main_dir, "/plots")
```

Introducing HDS data set

- We will explore a dataset called `healthcare-dataset-stroke-data`

- This dataset contains information about age, gender, hypertension, bmi, and other parameters to know the chances of getting a stroke

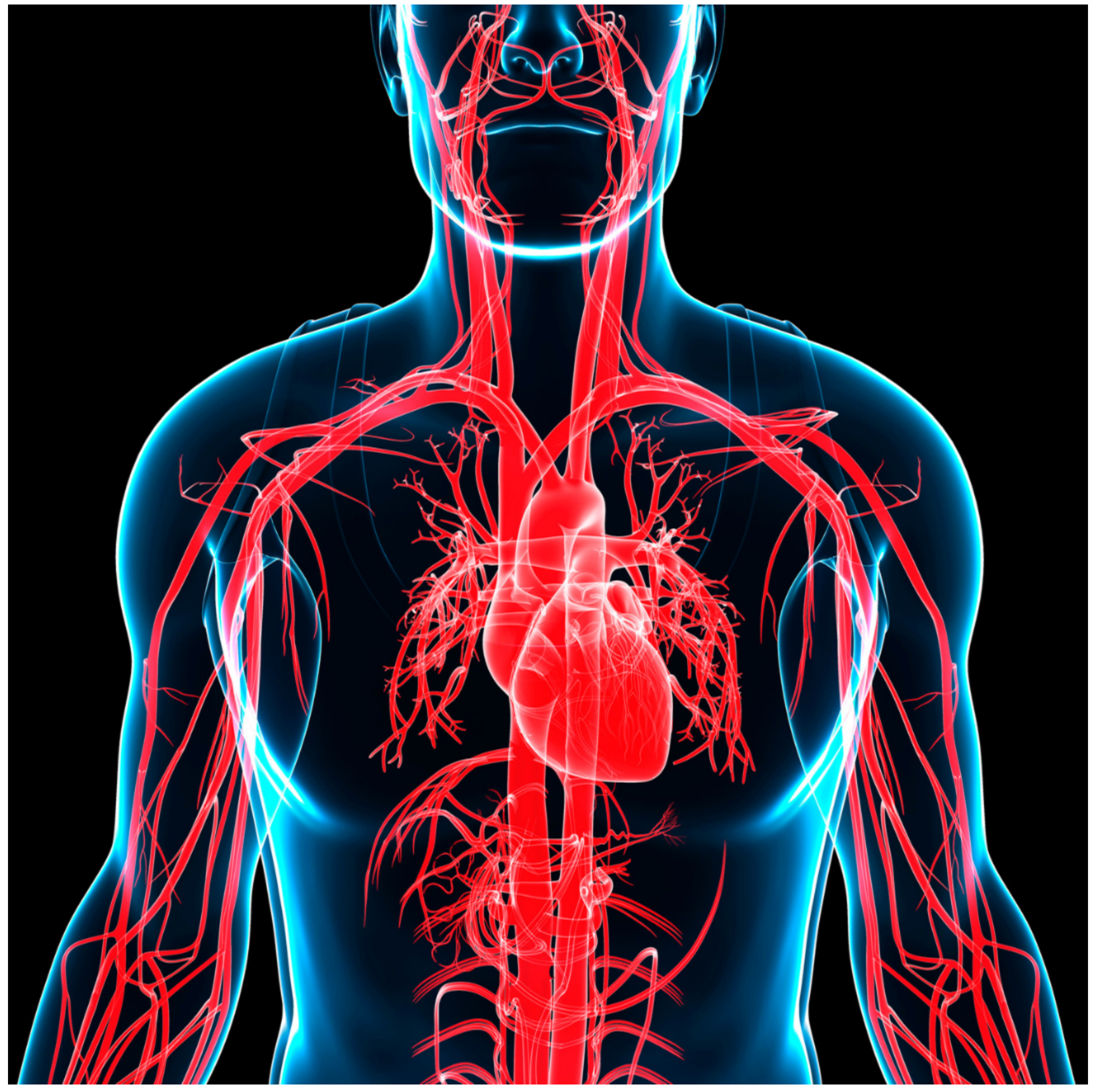
- The goal is to understand how different variables in the dataset affect the chances of a person suffering from a stroke

- The dataset has 12 characteristics (columns), of which:
  - **10 columns** relate to the **quality and characteristics** of the life of different people
  - The **stroke column** represents whether the people had a stroke or not

Load HDS dataset

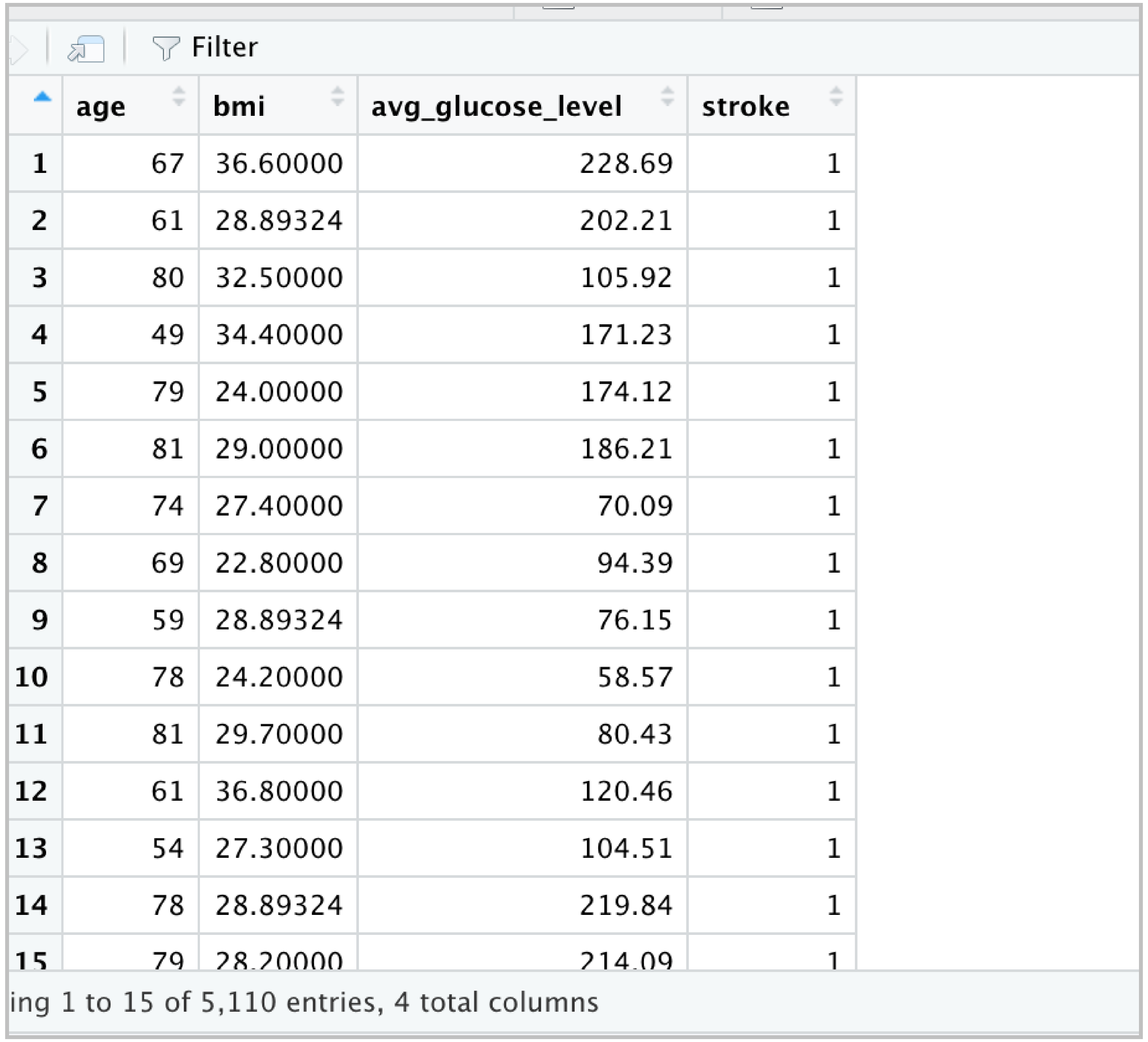
- Let’s load the HDS dataset from our `data_dir` into R’s environment and subset it

```
# Read CSV file called "healthcare-dataset-
stroke-data.csv"
HDS = read.csv(file =
file.path(data_dir,"/healthcare-dataset-stroke-
data.csv"), #<- provide file path
                header = TRUE,                #<- if
file has header set to TRUE
                stringsAsFactors = FALSE) #<- read
strings as characters, not as factors
```



Variables in the data

- In this module, we will explore a subset of this data set, which includes the following variables:
  - age
  - bmi
  - average_glucose_level and
  - stroke



	age	bmi	avg_glucose_level	stroke
1	67	36.60000	228.69	1
2	61	28.89324	202.21	1
3	80	32.50000	105.92	1
4	49	34.40000	171.23	1
5	79	24.00000	174.12	1
6	81	29.00000	186.21	1
7	74	27.40000	70.09	1
8	69	22.80000	94.39	1
9	59	28.89324	76.15	1
10	78	24.20000	58.57	1
11	81	29.70000	80.43	1
12	61	36.80000	120.46	1
13	54	27.30000	104.51	1
14	78	28.89324	219.84	1
15	79	28.20000	214.09	1

ing 1 to 15 of 5,110 entries, 4 total columns

Prepare data

- But before sub-setting the data, let’s handle the missing data in the dataset
- Then convert `bmi` into a numeric column followed by imputing the missing values with the mean

```
HDS$bmi <- as.numeric(as.character(HDS$bmi)) ##converting bmi column to numeric
# NA imputation
# we can use is.na() function to know about NA values
HDS$bmi[is.na(HDS$bmi)]<-mean(HDS$bmi,na.rm=TRUE) # Replacing na values of bmi column with it's
mean bmi
```

Subset data

- Now, we will subset our data to include the columns we will use to build the box plot

```
# Let's make a vector of column indices we would like to save.
column_ids= select(HDS, age, bmi, smoking_status,work_type)
# Let's save the subset into a new variable.
HDS_subset = column_ids
```

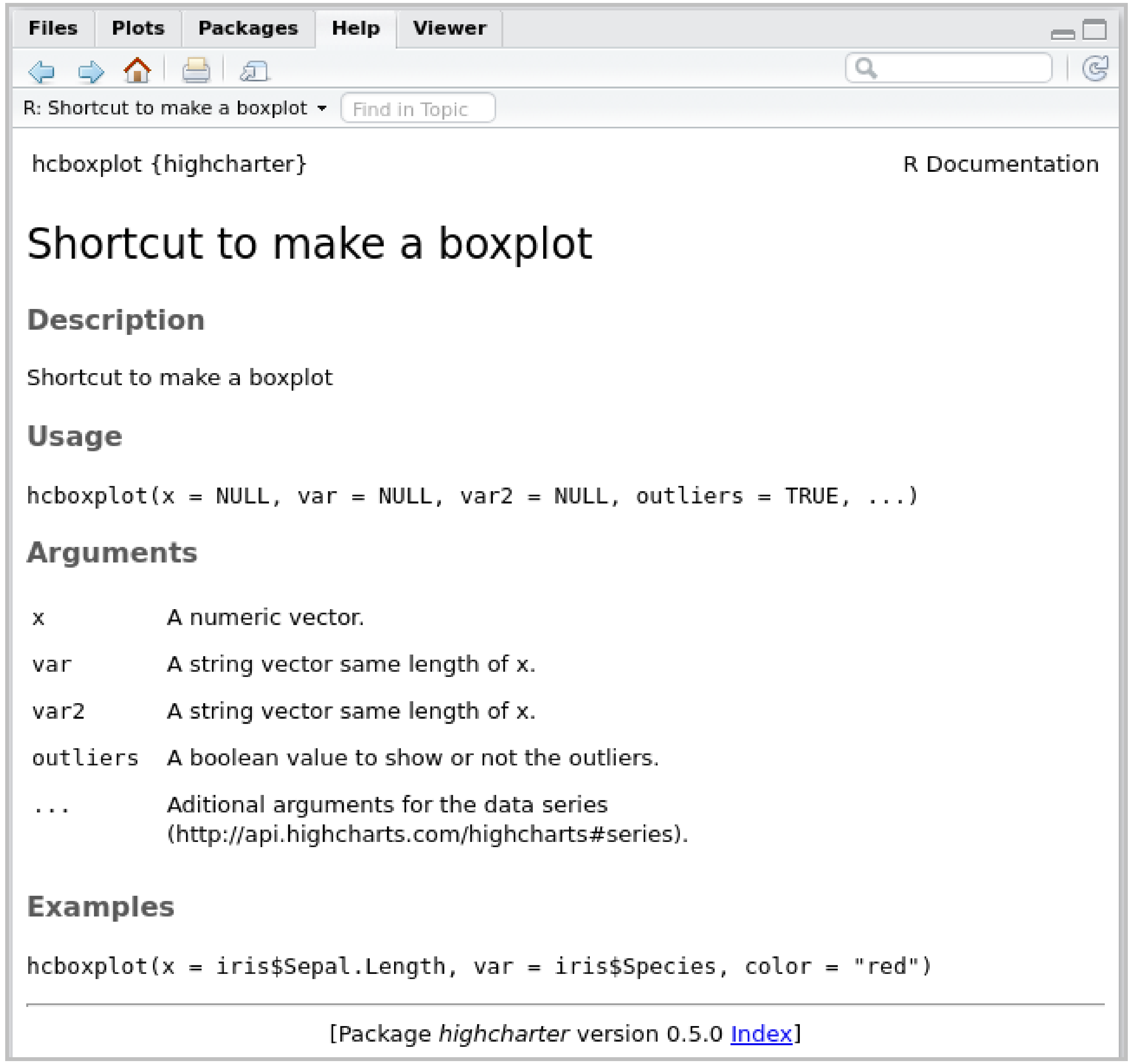
Highcharter boxplot: `hcboxplot()`

- `hcboxplot()` allows us to create an interactive boxplot

```r
library(highcharter)
library(tidyverse)
```

```r
?hcboxplot

hcboxplot(x = Numeric_data_vector,
          var = Categorical_data_vector,
          ...)
```

- It needs two arguments:
  - `x` requires the numeric data to be plotted along the x-axis (`hcboxplots` in highcharter are horizontal by default)
  - `var` requires categorical data to be plotted along the y-axis



hcboxplot {highcharter} R Documentation

Shortcut to make a boxplot

Description

Shortcut to make a boxplot

Usage

```r
hcboxplot(x = NULL, var = NULL, var2 = NULL, outliers = TRUE, ...)
```

Arguments

`x`	A numeric vector.
`var`	A string vector same length of x.
`var2`	A string vector same length of x.
`outliers`	A boolean value to show or not the outliers.
`...`	Aditional arguments for the data series (http://api.highcharts.com/highcharts#series).

Examples

```r
hcboxplot(x = iris$Sepal.Length, var = iris$Species, color = "red")
```

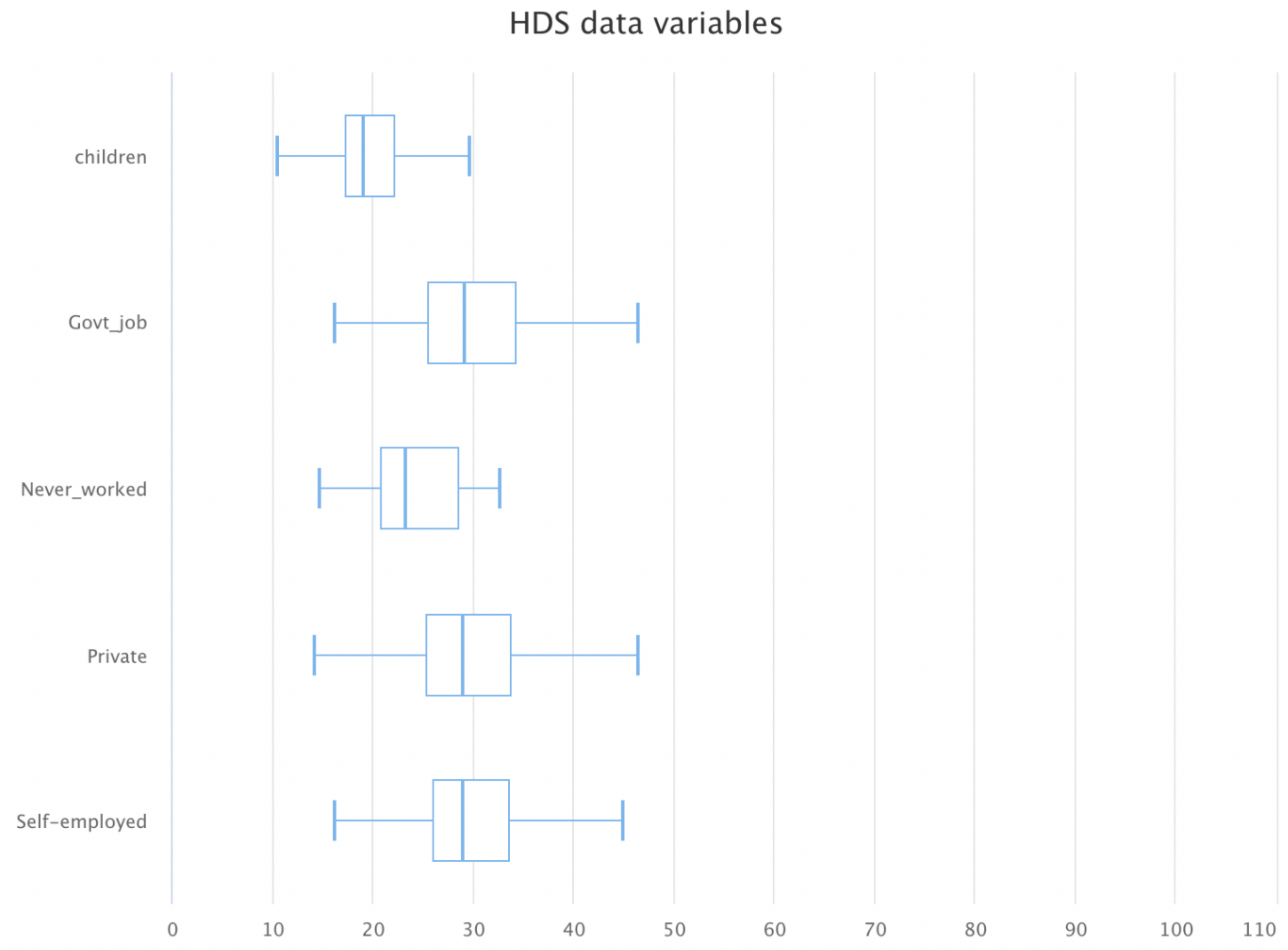
[Package *highcharter* version 0.5.0 Index]

Highcharter boxplot: hcboxplot (cont’d)

- We can use the subset we prepared for univariate analysis to create a boxplot
- How easy is it to interpret the resulting visualization?

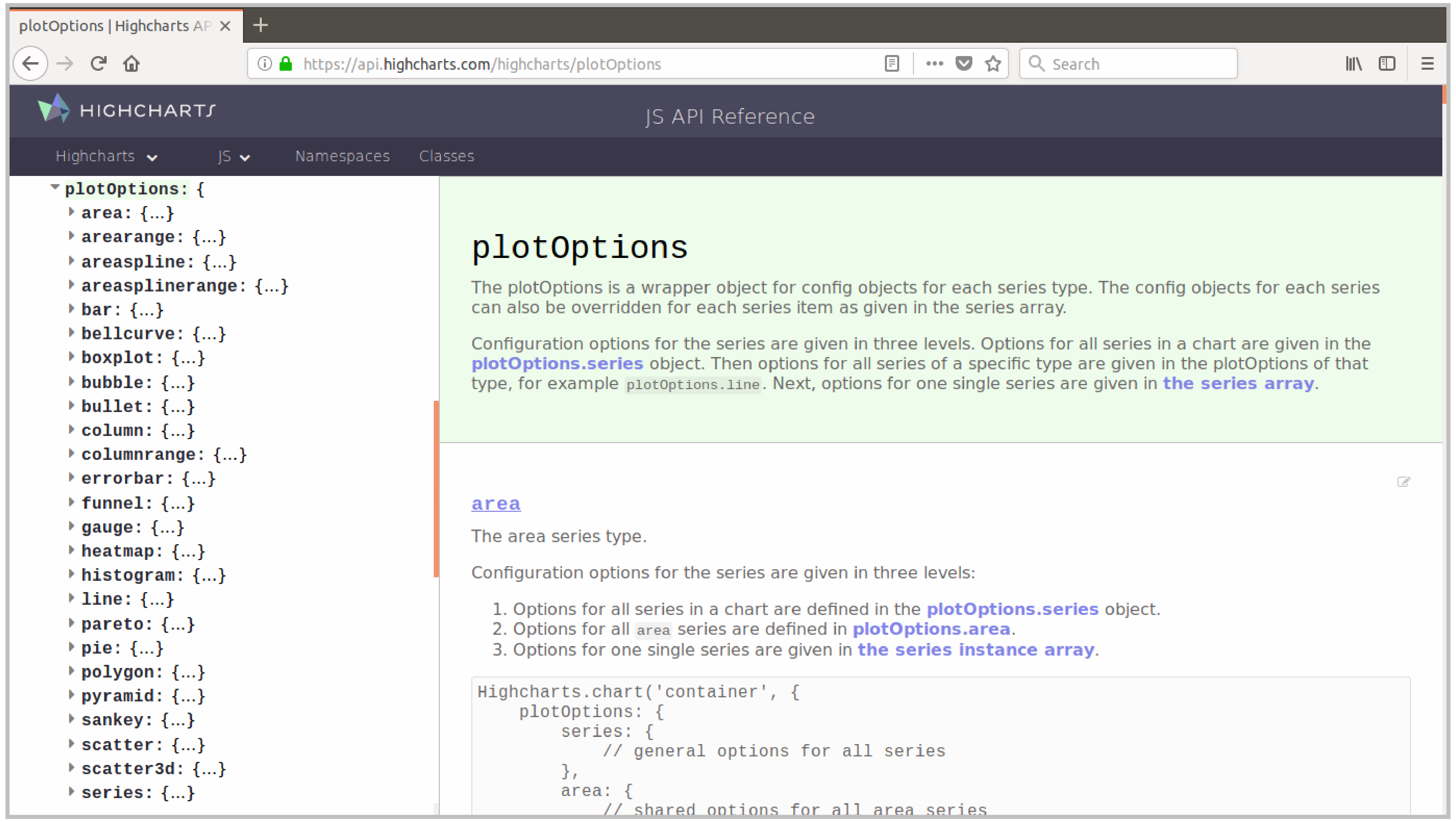
```
# Construct an interactive boxplot.
boxplot_interactive =
  hcboxplot(x = HDS_subset$bmi,
            var = HDS_subset$work_type) %>%
  hc_title(text = "HDS data variables")
```

```
boxplot_interactive
```



Highcharter Plotting Options

- To control individual layer/series options for various plot types, we use the `hc_plotOptions()` function
- It can be **piped (`%>%`)** to the original chart to **enhance** our base plot
- Each series type in the chart can be given a unique set of options
- For a complete breakdown, refer to ***Highcharts API documentation***



plotOptions | Highcharts AP ×
https://api.highcharts.com/highcharts/plotOptions
Search
HIGHCHARTS
JS API Reference
Highcharts
JS
Namespaces
Classes

```
plotOptions: {
  area: {…}
  arearange: {…}
  areaspline: {…}
  areasplinerange: {…}
  bar: {…}
  bellcurve: {…}
  boxplot: {…}
  bubble: {…}
  bullet: {…}
  column: {…}
  columnrange: {…}
  errorbar: {…}
  funnel: {…}
  gauge: {…}
  heatmap: {…}
  histogram: {…}
  line: {…}
  pareto: {…}
  pie: {…}
  polygon: {…}
  pyramid: {…}
  sankey: {…}
  scatter: {…}
  scatter3d: {…}
  series: {…}
```

plotOptions

The plotOptions is a wrapper object for config objects for each series type. The config objects for each series can also be overridden for each series item as given in the series array.

Configuration options for the series are given in three levels. Options for all series in a chart are given in the **plotOptions.series** object. Then options for all series of a specific type are given in the plotOptions of that type, for example `plotOptions.line`. Next, options for one single series are given in **the series array**.

area

The area series type.

Configuration options for the series are given in three levels:

1. Options for all series in a chart are defined in the **plotOptions.series** object.
2. Options for all `area` series are defined in **plotOptions.area**.
3. Options for one single series are given in **the series instance array**.

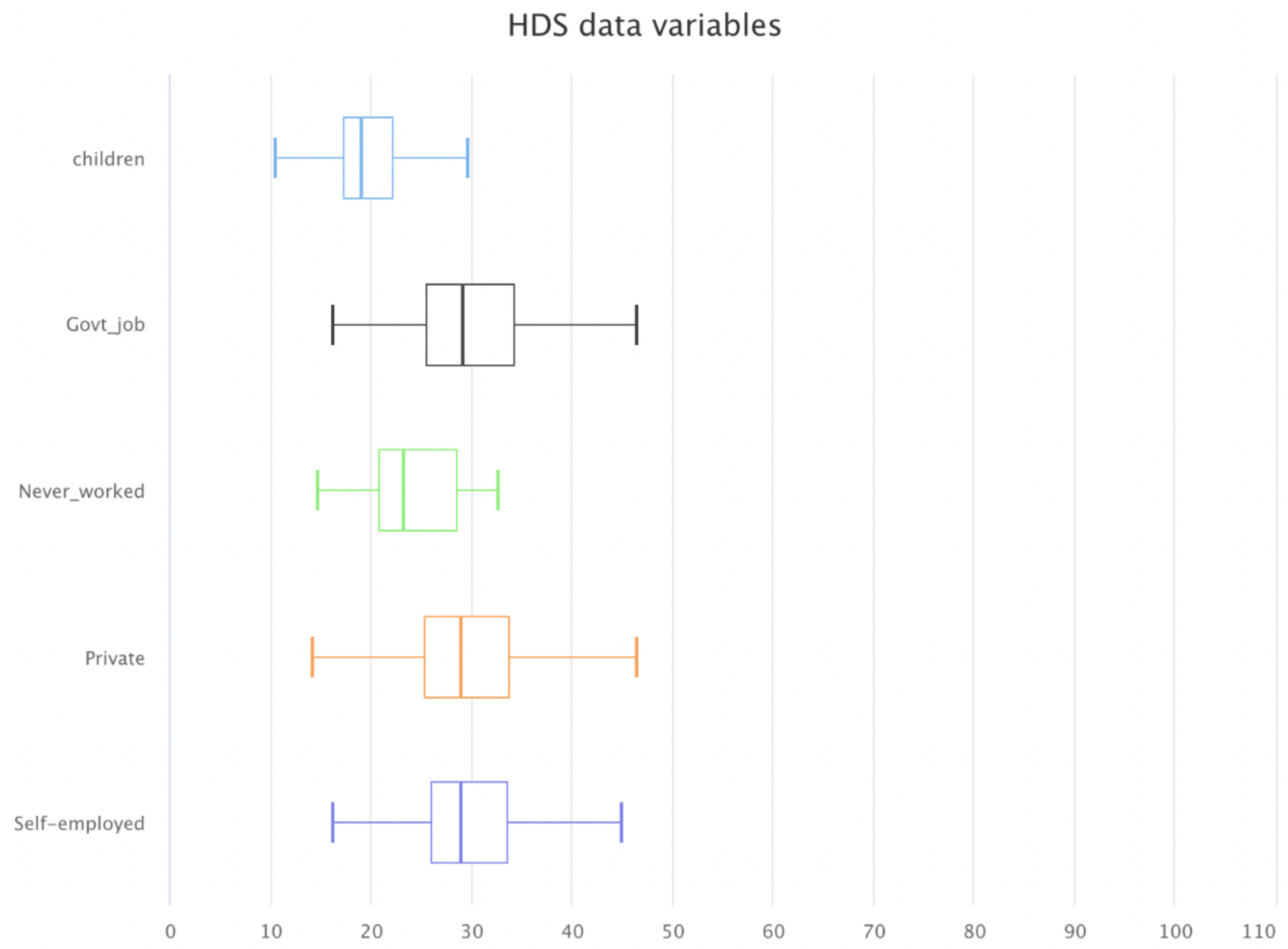
```
Highcharts.chart('container', {
    plotOptions: {
        series: {
            // general options for all series
        },
        area: {
            // shared options for all area series
```

Customize hcboxplot with `hc_plotOptions`

- Now we will use the `hc_plotOptions()` function to customize the color of the boxplot

```
# Enhance original boxplot with color options.
boxplot_interactive = boxplot_interactive %>%
  hc_plotOptions(   #<- plot options
    boxplot = list(      #<- for boxplot
      colorByPoint = TRUE)) #<- color each box
```

```
boxplot_interactive
```



Column plot: prepare data

- Let’s create a new subset of our data with the columns needed to create a column plot

```
column_ids= select(HDS, age,bmi,avg_glucose_level)
# Let's save the subset into a new variable.
HDS_subset_col = column_ids
```

Column plot: prepare data (contd)

- Let's create an interactive multiple-column plot to compare summary by variable
- We first need to get the summary statistics and save them as a separate data frame

```
# Create data summary.
HDS_summary = summary(HDS_subset_col)

# Save it as a data frame.
HDS_summary = as.data.frame(HDS_summary)

# Inspect the data.
head(HDS_summary)
```

```
  Var1     Var2               Freq
1           age Min.   : 0.08
2           age 1st Qu.:25.00
3           age Median :45.00
4           age Mean   :43.23
5           age 3rd Qu.:61.00
6           age Max.   :82.00
```

- The data frame contains a variable with no values, so let's remove this column
- We can then rename the other columns for ease of readability

```
# Remove an empty variable.
HDS_summary$Var1 = NULL

# Rename remaining columns.
colnames(HDS_summary) = c("Variable",
                          "Summary")
# Inspect updated data.
head(HDS_summary)
```

```
  Variable         Summary
1      age Min.   : 0.08
2      age 1st Qu.:25.00
3      age Median :45.00
4      age Mean   :43.23
5      age 3rd Qu.:61.00
6      age Max.   :82.00
```

Column plot: prepare data (contd)

- Let’s separate the values from the statistics using tidyr’s `separate()` function

```
# Separate `Summary` column into 2 columns.
HDS_summary = HDS_summary %>%              #<- set original data
  separate(Summary,                        #<- separate `Summary` variable
           into = c("Statistic", "Value"), #<- into 2 columns: `Statistic`, `Value`
           sep = ":",                      #<- set separating character
           convert = TRUE)                 #<- where applicable convert data (to numeric)


# Inspect the first few entries in data.
head(HDS_summary)
```

```
  Variable Statistic Value
1      age   Min.       0.08
2      age   1st Qu. 25.00
3      age   Median  45.00
4      age   Mean    43.23
5      age   3rd Qu. 61.00
6      age   Max.    82.00
```

```
# Inspect total number of rows in data including NAs.
nrow(HDS_summary)
```

```
[1] 18
```

Column plot: create plot

- Then, clean the data by removing NA observations and build the chart

```
# Inspect `Value` column for `NAs`.
which(is.na(HDS_summary$Value) == TRUE)
```

```
integer(0)
```

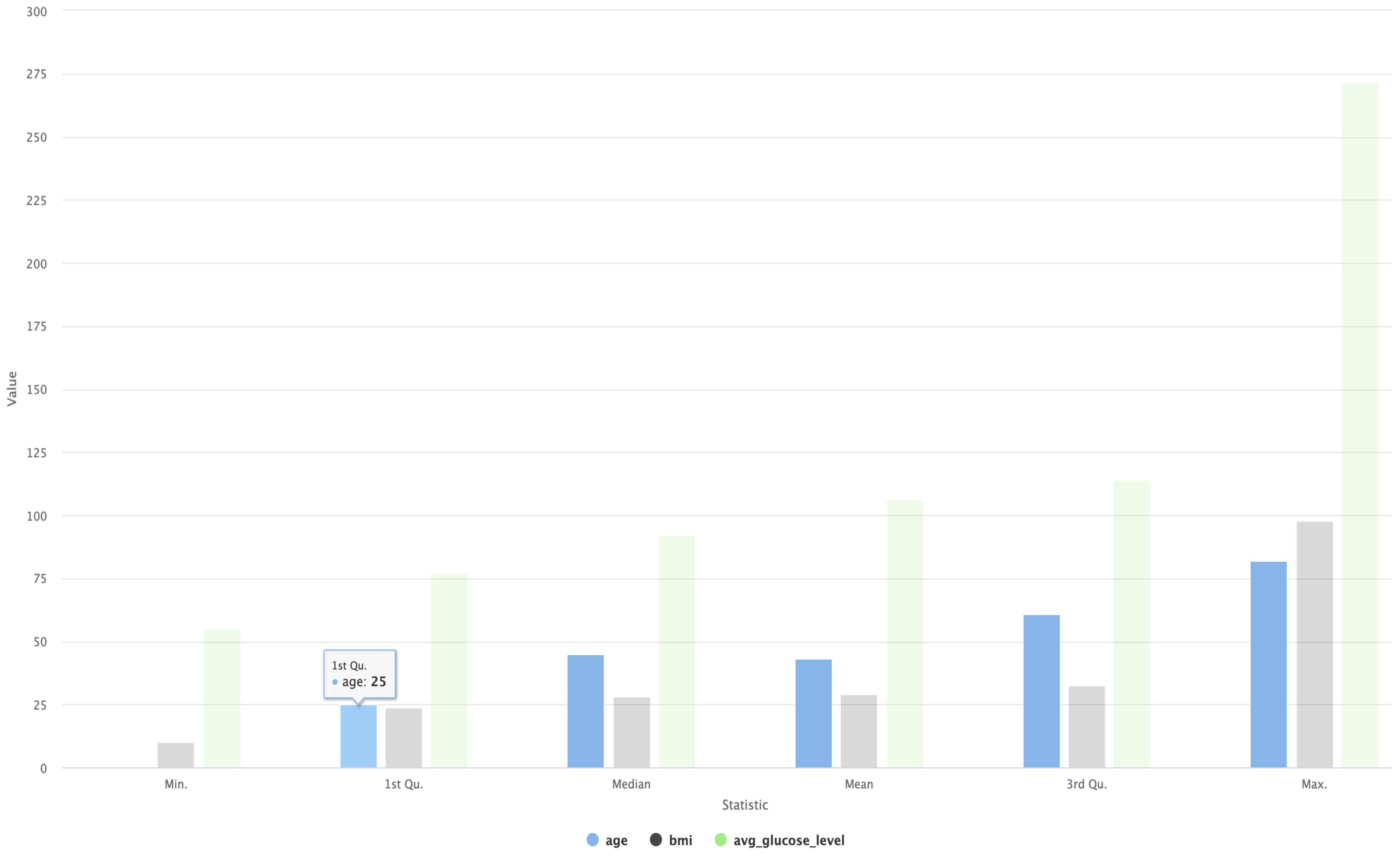
```
HDS_summary = subset(HDS_summary, !is.na(Value))

nrow(HDS_summary)
```

```
[1] 18
```

```
# Construct the summary chart.
HDS_summary_interactive =
  hchart(HDS_summary,                #<- set data
         "column",                   #<- set type (`column` in highcharts)
         hcaes(x = Statistic,        #<- arrange `Statistics` across x-axis
               y = Value,            #<- map `Value` of each `Statistic` to y-axis
               group = Variable))    #<- group columns by `Variable`
```

Column plot: display plot

```
HDS_summary_interactive
```



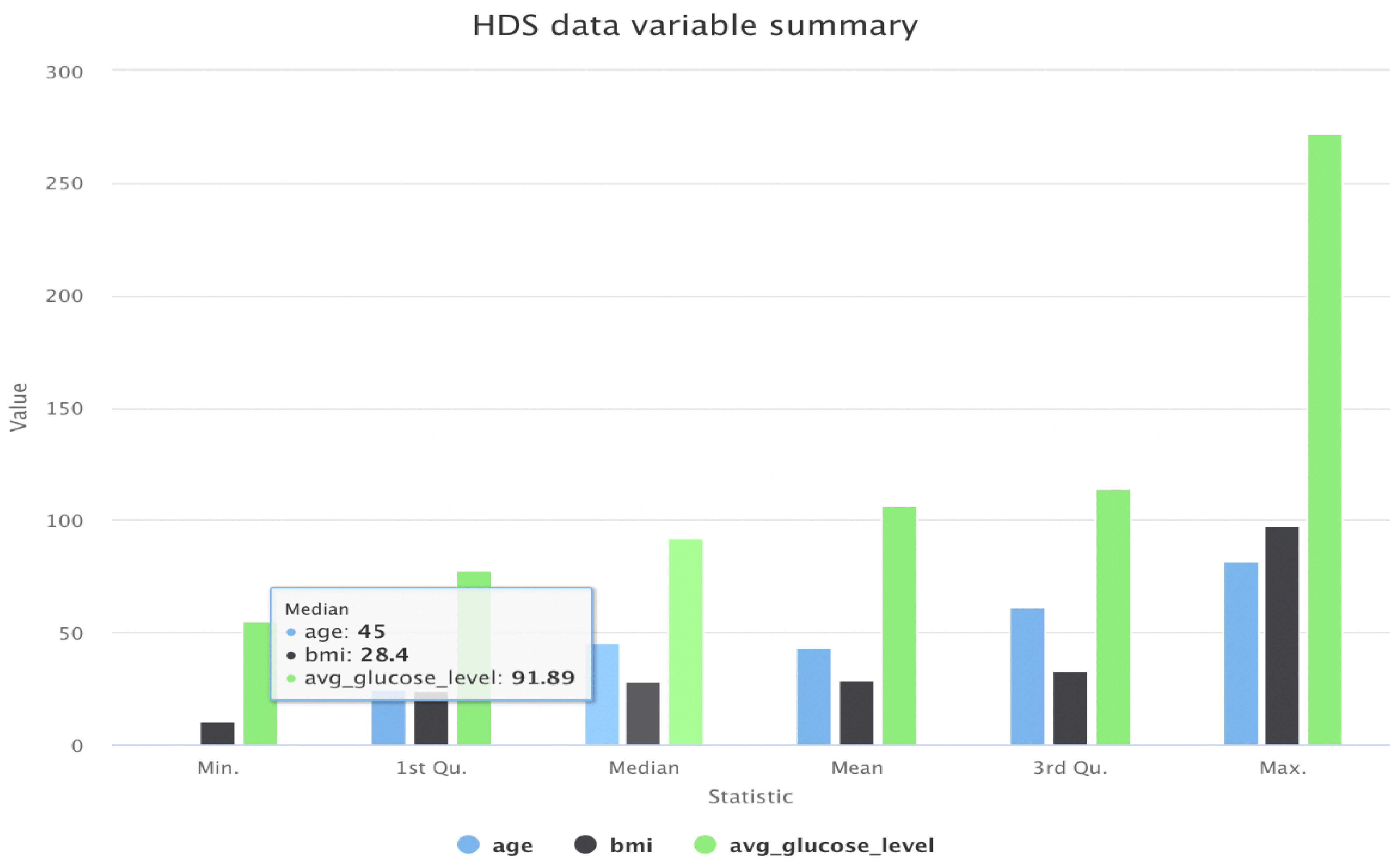
Column plot: customize tooltip

- While comparing each variable’s summary statistics, it would be convenient for the `tooltip` to contain information about the `group` of variables i.e `age, bmi,` and `avg_glucose_level` rather than than the individual variables
- We can control different `tooltip` options of the chart using the `hc_tooltip()` option
- The `shared` option is often used to share a tooltip between members of a `group`

```r
# Adjust tooltip options by piping `hc_tooltip` to base plot.
HDS_summary_interactive = HDS_summary_interactive %>%
  hc_tooltip(shared = TRUE)  %>%               #<- `shared` needs to be set to `TRUE`
  hc_title(text = "HDS data variable summary") #<- add title to your plot
```

Column plot: customize tooltip (cont'd)

```
HDS_summary_interactive
```



Module completion checklist

Objective	Complete
Construct and save a boxplot and a column plot with hchart	✔
Visualize a correlation plot with hchart	

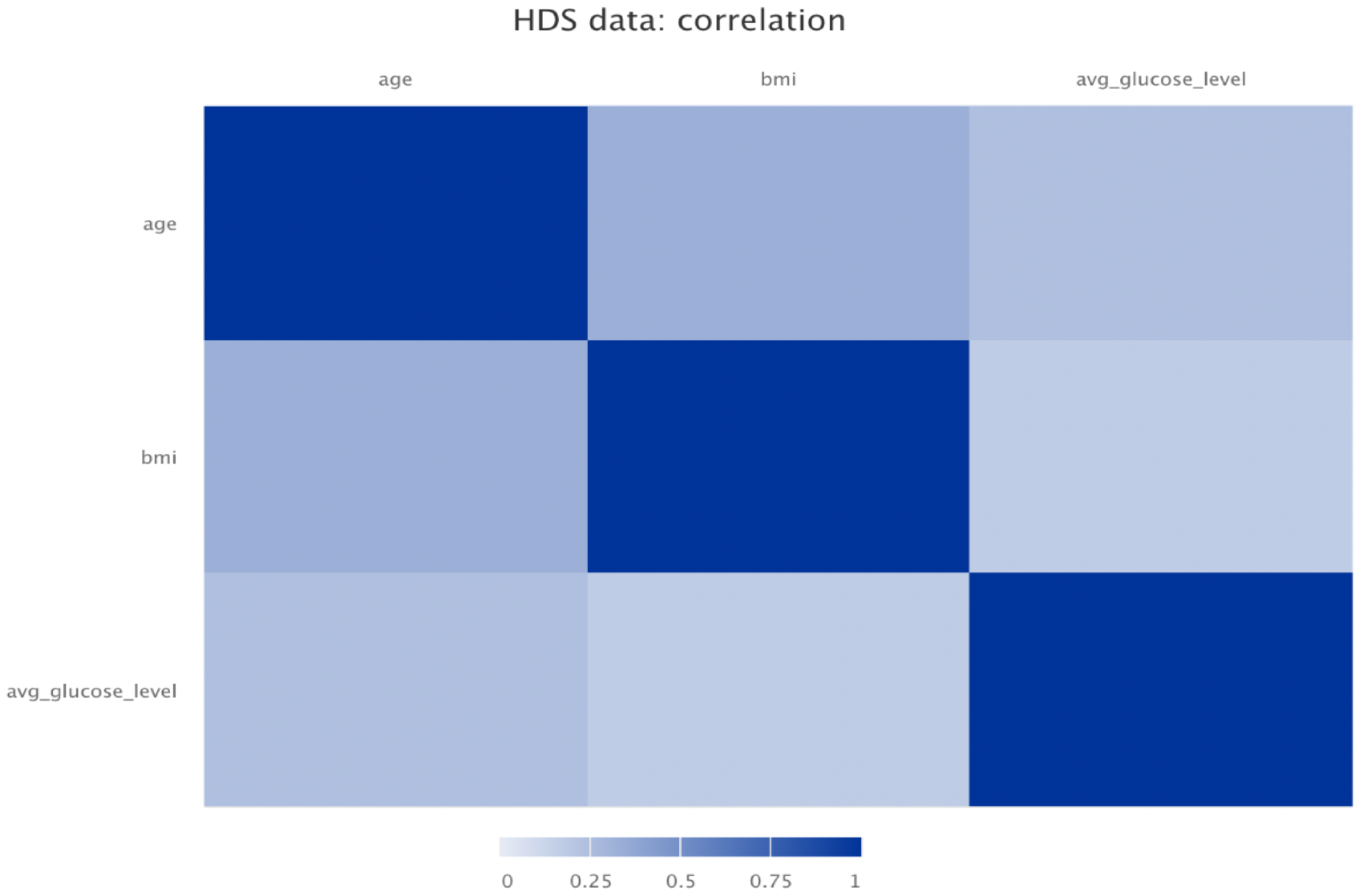
hchart: correlation plot

- Certain charts require less data processing within the highcharter package
- For instance, if we pass `hchart()` a **correlation matrix**, it will recognize the data type and create a **correlation plot** in response
- No other arguments are necessary to create a basic plot

```
# Compute a correlation matrix for the first
# 4 variables in our data.
cor_matrix = cor(HDS_subset_col[, 1:3])

# Construct a correlation plot by
# simply giving the plotting function
# a correlation matrix.
correlation_interactive = hchart(cor_matrix) %>%
  # Add title to the plot.
  hc_title(text = "HDS data: correlation")
```

hchart: correlation plot (cont'd)

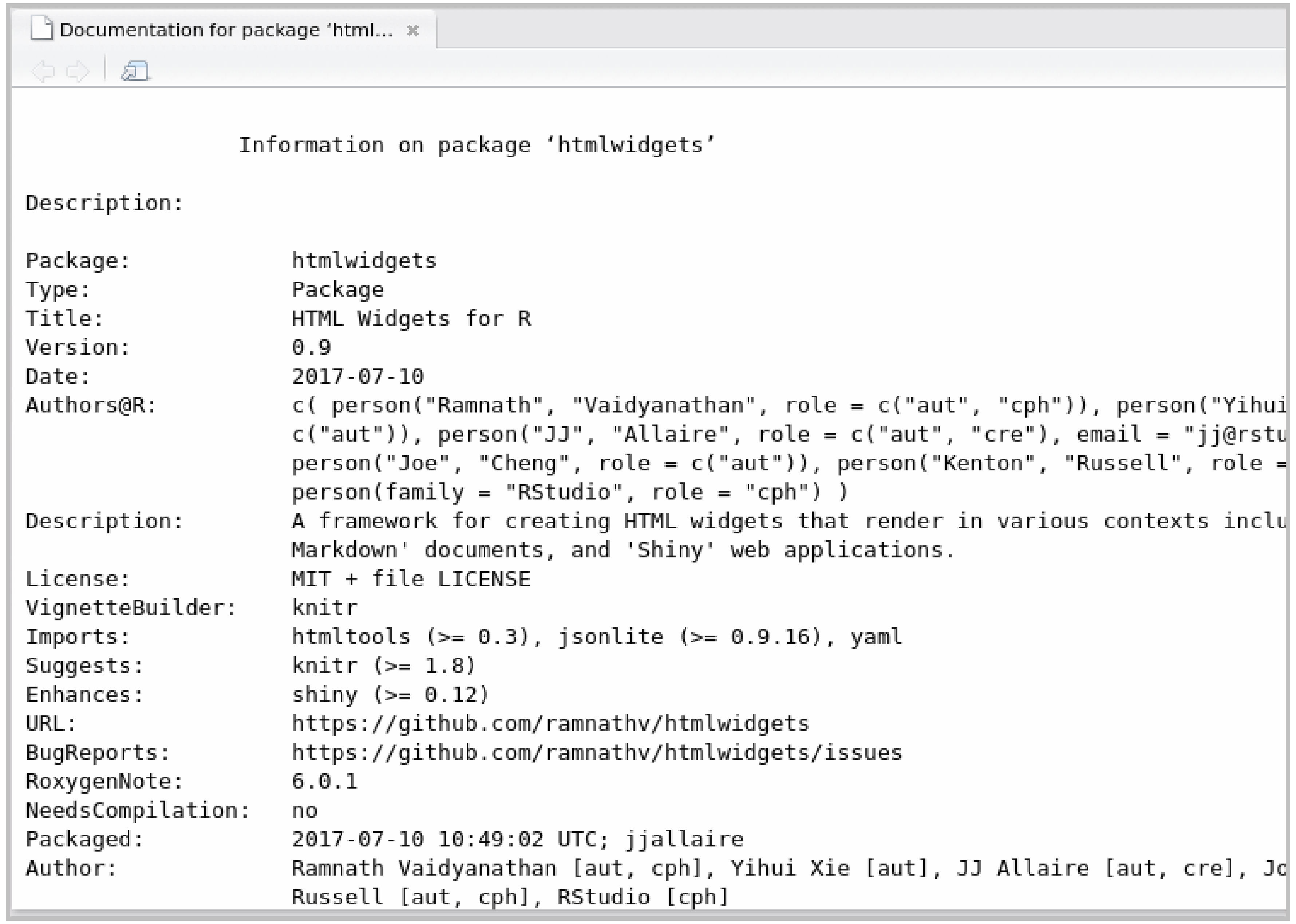
```
correlation_interactive
```



Save interactive plots: htmlwidgets

- The htmlwidgets package allows us to use JavaScript visualization libraries in R console
- We can embed widgets in R markdown and Shiny web applications
- ***Find out more about `htmlwidgets`***

```r
# Install `htmlwidgets` package.
install.packages("htmlwidgets")

# Load the library.
library(htmlwidgets)

# View documentation.
library(help = "htmlwidgets")
```



```
                Information on package 'htmlwidgets'

Description:

Package:            htmlwidgets
Type:               Package
Title:              HTML Widgets for R
Version:            0.9
Date:               2017-07-10
Authors@R:          c( person("Ramnath", "Vaidyanathan", role = c("aut", "cph")), person("Yihui
                    c("aut")), person("JJ", "Allaire", role = c("aut", "cre"), email = "jj@rstu
                    person("Joe", "Cheng", role = c("aut")), person("Kenton", "Russell", role =
                    person(family = "RStudio", role = "cph") )
Description:        A framework for creating HTML widgets that render in various contexts inclu
                    Markdown' documents, and 'Shiny' web applications.
License:            MIT + file LICENSE
VignetteBuilder:    knitr
Imports:            htmltools (>= 0.3), jsonlite (>= 0.9.16), yaml
Suggests:           knitr (>= 1.8)
Enhances:           shiny (>= 0.12)
URL:                https://github.com/ramnathv/htmlwidgets
BugReports:         https://github.com/ramnathv/htmlwidgets/issues
RoxygenNote:        6.0.1
NeedsCompilation:   no
Packaged:           2017-07-10 10:49:02 UTC; jjallaire
Author:             Ramnath Vaidyanathan [aut, cph], Yihui Xie [aut], JJ Allaire [aut, cre], Jo
                    Russell [aut, cph], RStudio [cph]
```

Save interactive plots: htmlwidgets (cont'd)

- We can save widgets to a dedicated plot directory

```
# Set working directory to where you save plots.
setwd(plot_dir)

# Save desired interactive plot to an HTML file.
saveWidget(scatter_interactive,        #<- plot object to save
           "interactive_scatterplot.html", #<- name of file to where the plot is to be saved
           selfcontained = TRUE)              #<- set `selfcontained` to TRUE, so that
                                              #   all necessary files and scripts are embedded
                                              #   into the HTML file itself
```

Knowledge check

Exercise

You are now ready to try tasks 1-9 in the Exercise for this topic

Module completion checklist

Objective	Complete
Construct and save a boxplot and a column plot with hchart	✔
Visualize a correlation plot with hchart	✔

Interactive plots: topic summary

In this part of the course, we have covered:

- Creating correlation plots, column plots, box plots
- Saving and viewing interactive plots by using `htmlwidgets` library

Congratulations on completing this module!