Module3_2_Radar_Charts_Reviewed

May 17, 2021

RADAR CHARTS

0.1 Table of Contents

Introduction

Radar charts are a way to visualize and aggregate multivariate data in one plot. They are circular plots with "spokes" representing an axis for each variable in the plot. Lines are drawn through these axes to demonstrate the differences between the variable values of the subjects plotted.

Usage scenarios

Radar Charts are particularly useful when dealing with multivariate data. In particular, they are especially good at showing differences between a handful of different entities with regards to a number of variables. A usual application of radar plots is, for example, demonstrating strengths or weaknesses of particular subjects, such as in sports. Another example is differentiating between the business approaches of different companies in a few categories such as marketing, investments, etc.

R Implementation

First let's download the following libraries:

0.1.1 ggplot2

`ggplot2` is our main plotting library. It is a specialized library made to create visually pleasing data visualizations. There's no need to install ggplot2 because it already exists on your Jupyter environment.

0.1.2 ggradar

A ggplot2 extension that allows us to create radar graphs with a simple syntax. The last version of this extension available on CRAN is not compatible with our R version. To circumvent this, let's download from the GitHub repository of the developer, Ricardo Bion .

[1]:
```
devtools::install_github("ricardo-bion/ggradar",
                         dependencies=TRUE)
```

```
Downloading GitHub repo ricardo-bion/ggradar@master
from URL https://api.github.com/repos/ricardo-bion/ggradar/zipball/master
Installing ggradar
Installing extrafont
Installing extrafontdb
'/usr/lib/R/bin/R' --no-site-file --no-environ --no-save --no-restore --quiet  \
  CMD INSTALL '/tmp/Rtmp0ww9AD/devtools6113e839318/extrafontdb'  \
  --library='/resources/common/R/Library' --install-tests

Installing Rttf2pt1
'/usr/lib/R/bin/R' --no-site-file --no-environ --no-save --no-restore --quiet  \
  CMD INSTALL '/tmp/Rtmp0ww9AD/devtools6117474f89/Rttf2pt1'  \
  --library='/resources/common/R/Library' --install-tests

'/usr/lib/R/bin/R' --no-site-file --no-environ --no-save --no-restore --quiet  \
  CMD INSTALL '/tmp/Rtmp0ww9AD/devtools61158220264/extrafont'  \
  --library='/resources/common/R/Library' --install-tests

Skipping install of 'extrafontdb' from a cran remote, the SHA1 (1.0) has not
changed since last install.
  Use `force = TRUE` to force installation
Installing tidyr
Installing Rcpp
'/usr/lib/R/bin/R' --no-site-file --no-environ --no-save --no-restore --quiet  \
  CMD INSTALL '/tmp/Rtmp0ww9AD/devtools6114bf709e7/Rcpp'  \
  --library='/resources/common/R/Library' --install-tests

'/usr/lib/R/bin/R' --no-site-file --no-environ --no-save --no-restore --quiet  \
  CMD INSTALL '/tmp/Rtmp0ww9AD/devtools6116b1cf903/tidyr'  \
  --library='/resources/common/R/Library' --install-tests

'/usr/lib/R/bin/R' --no-site-file --no-environ --no-save --no-restore --quiet  \
  CMD INSTALL  \
  '/tmp/Rtmp0ww9AD/devtools6116c078962/ricardo-bion-ggradar-559eeb7'  \
  --library='/resources/common/R/Library' --install-tests
```

0.1.3 dplyr

`dplyr` is responsible for pipe `%%` operation in R.

A pipe basically takes the output from one function and feeds it to the next function. An in-depth explanation of pipes and dplyr can be seen here.

[2]:
```
install.packages("dplyr")
```

```
Installing package into '/resources/common/R/Library'
(as 'lib' is unspecified)
```

0.1.4 Scales

`scales` provides methods for automatically determining labels for axes and legends.

[3]:
```
install.packages("scales")
```

```
Installing package into '/resources/common/R/Library'
(as 'lib' is unspecified)
```

Now let's load our libraries:

[8]:
```
library(ggplot2)
library(ggradar)
library(dplyr)
library(scales)
```

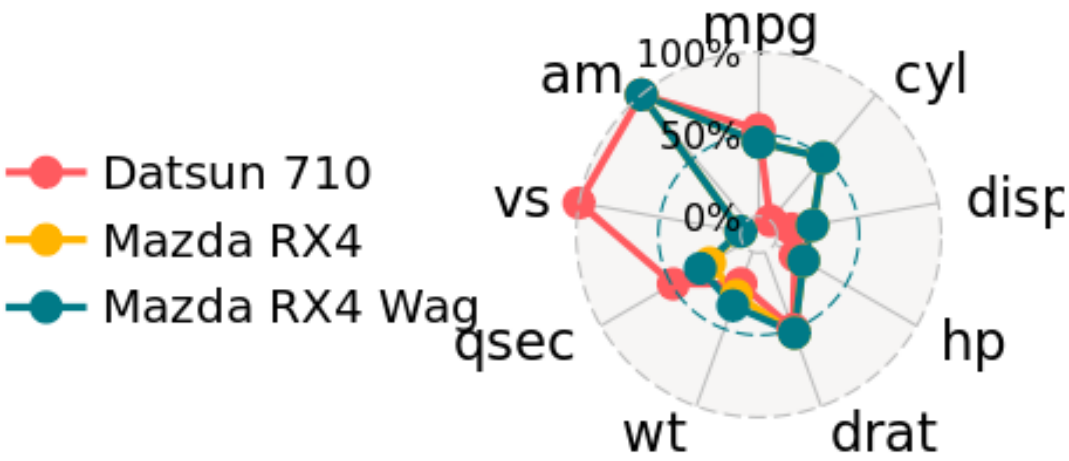
Now let's create our graph using the `mtcars` dataset.

[9]:
```
#Select our dataset
mtcars %>%
  #atribute rownames to a variable
  add_rownames( var = "group" ) %>%
  #assign each variable -- car names -- to their related variables
  mutate_each(funs(rescale), -group) %>%
  #select which  data to plot
  head(3) %>% select(1:10) -> mtcars_radar
```

Now let's plot our graph!

[10]:
```
#this code will generate lots of warnings, so let's supress them
options(warn=-1)
ggradar(mtcars_radar)
```

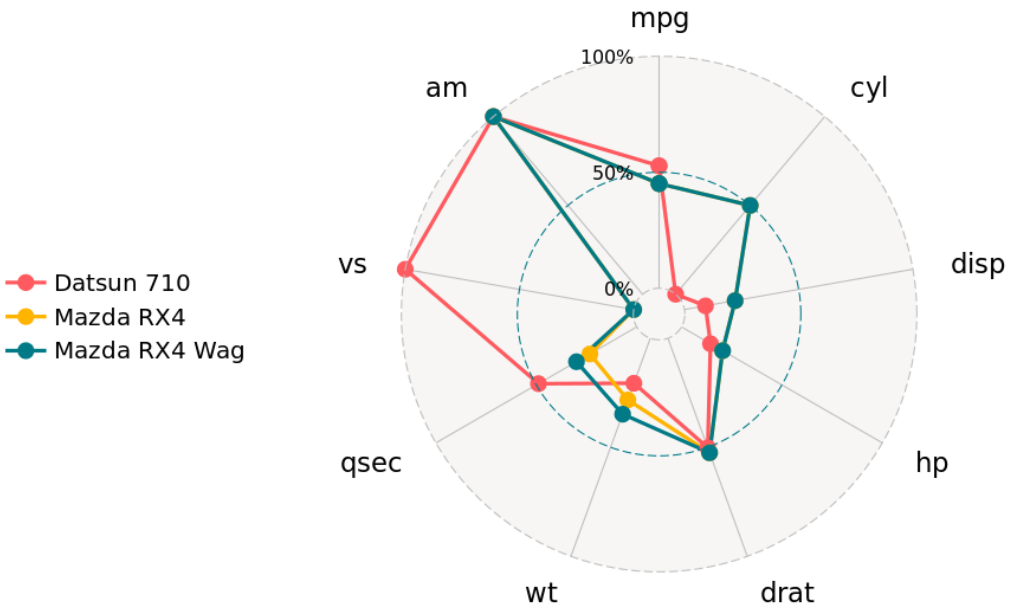
[10]:



The default size of the plot is too small. To enlarge our plot here on Jupyter, we need to use `IRkernel` from the library `devtools`

[11]:
```
IRkernel::set_plot_options(width=950, height=600, units='px')
ggradar(mtcars_radar)
```

[11]:



0.1.5 About the Author:

Hi! It's Francisco Magioli, the author of this notebook. I hope you found R easy to learn! There's lots more to learn about R but you're well on your way. Feel free to connect with me if you have any questions.