DATA VISUALISATION IN PERL

A REPORT ON PACKAGE

SUBJECT: OPERATING SYSTEMS

DEPARTMENT OF APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCES, PSG COLLEGE OF TECHNOLOGY,COIMBATORE – 641004

SUBMITTED BY

JESWANTH MURTHY V - 18PT14

SHUKI VARSUN J - 18PT35

TABLE OF CONTENTS

DATA VISUALISATION IN PERL

Abstract

The package aims at understanding the basics of perl script and use its various CPAN modules to achieve data visualisation. Precisely, we can use this package to understand the handling of datasets and using them efficiently to attain data visualisation via graphical analysis in the perl script.

1.1 Introduction

The objective of this package is to learn the basic functionalities of perl script and use them to analyse data sets graphically. In this package we deeply analyse the FIFA World cup 2018 graphically based on the stats provided by the datasets. We provide a menu-driven user interface for overall analysis and country wise analysis which would provide the user with graphs based on his/her input. Our task was to create appropriate functions to analyse the datasets in various ways through various stats and show the analysis as images i.e. graphs.

1.2 Description

We created the following subroutines to attain data visualisation,

- display_teams () : To print all the countries that participated in the FIFA World cup 2018 on the console so as to provide the choice for the user to input.
- squad () : To print all the players in the squad of the particular input country in the FIFA World cup 2018.

- goal_bar () : To manipulate the CSV file, obtain required data to create a bar chart that provides the visualisation of goal scorers and the number of goals they scored for a particular country, store it in the local drive as jpeg or png file and display it.
- age_pie () : To manipulate the CSV file, obtain required data to create a pie chart that provides the visualisation of age composition of the players in the squad of a particular country, store it in the local drive as jpeg or png file and display it.
- shots () : To manipulate the CSV file, obtain required data to create a pie chart that provides the visualisation of the shot accuracy and conversion rate as goals in home matches and away matches of a country in the FIFA World cup 2018, store it in the local drive as jpeg or png file and display it.
- gd_line () : To manipulate the CSV file, obtain required data to create a line graph that provides the stats of goal difference (Goals scored – Goals conceded) of a country, as the tournament progresses, store it in the local drive as jpeg or png file and display it.
- goal_all_point () : To manipulate the CSV file, obtain required data to create a point graph that provides the visualisation of total goals scored, goals scored in home matches and goals scored in away matches of all the countries in the FIFA World cup 2018, store it in the local drive as jpeg or png file and display it.
- position_pie () : To manipulate the CSV file, obtain required data to create a pie chart that provides the visualisation of the compostion of all goals scored by the players of all the countries in the FIFA World cup 2018 based on their playing position (i.e. attack or midfield or defense), store it in the local drive as jpeg or png file and display it.
- attendance_line() : To manipulate the CSV file, obtain required data to create a line graph that provides the visualisation of total stadium attendance in all the matches of FIFA World cup 2018, store it in the local drive as jpeg or png file and display it.

1.3 System calls used

Some system calls that were used include:

1. system() - to call various operating system commands from within the program

2. open() - to open the required files

3. close() - to close files that were opened

1.4 Tools and Technologies

1. ActivePerl-5.28.1.0000-MSWin32-x64-865dc3eb Application
2. Notepad++
3. Required CPAN modules were added

1.5 Workflow

For data visualistaion in perl certain modules need to be installed using the “cpan” command in the command prompt. To open a csv file “Text::CSV_XS” module is used, which helps in reading the data from the csv file as whole at once and store in a variable. This helps in reading the data stored in the variable according to the user’s need, like line by line or word by word, etc. To plot graphs or charts the “GD::Graph” module is installed. This module helps the user to use many sub-modules like “GD::Graph::bars”, “GD::Graph::points”, “GD::Graph::lines”,” GD::Graph::pie” and “GD::Graph::colour” to plot bar graph, scatterplot, line graph, pie chart and specify colours to the attributes of the graph respectively. The system calls “open()”, “close()” and “system()” are used to open a file, close a file and open the file specified in the system command in windows in their respective application which is used to display the graph plotted respectively.

The csv file is read fully and stored in a variable. Then we read the variable line by line and store it in another variable. Then this variable is broken into array of strings with the delimiter “,” using the “field” method in class “Text::csv”. Then from this array we extract the required string using the index and store it in another array using which we plot the graph. This way we plot all the graphs and we manipulate the data using the array. After plotting the graphs it is stored as an “jpeg” or “png” file in the current working directory. Using the system() function we open the file to display the image for the user to see.

All graphs are plotted in the same manner with small changes in the attributes,

Example: #initialisation of an array

```
@data = (
["1st","2nd","3rd","4th","5th","6th","7th", "8th", "9th"],
[    1,    2,    5,    6,    3,  1.5,    1,     3,     4],
[ sort { $a <=> $b } (1, 2, 5, 6, 3, 1.5, 1, 3, 4) ]
);
my $graph = GD::Graph::<type of graph>->new(400, 300);
#initialising values for the attributes of the graph
$graph->set(
x_label           => 'X Label',
y_label           => 'Y label',
title             => 'Some simple graph',
y_max_value       => 8,
y_tick_number     => 8,
y_label_skip      => 2
) or die $graph->error;
my $gd = $graph->plot(\@data) or die $graph->error;
```

Note: the plot() function should always have the address of the array.

For storing the plotted graph as an image,

Example : my $file = <filename>;

```
open(my $out, '>', $file) or die "Cannot open '$file' for write: $!";
binmode $out;
print $out $graph->gd->jpeg;
close $out;
```

1.6 Result and Discussion

On observing the purposes of data visualisation, we can understand and get a wholesome idea of implementing the data visualisation using perl along with the usage of various data manipulation. Extensive use of modules like GD::Graph::bars, GD::Graph::bars , GD::Graph::colour , GD::Graph::data , GD::Graph::pie, GD::Graph::lines, GD::Graph::points and others like Text::CSV_XS, Switch along with the regular ones can be observed.

1.7 Conclusion

We have successfully completed this assignment and have gained a deep insight into the usage of data visualisation and perl. We have also understood and demonstrated the various ways of representing the data and manipulating the data to arrive at a conclusion or result using various graphs and charts and presenting them in a user friendly way. Therefore, data visualisation using perl has been successfully demonstrated.

1.8 Bibliography

1.81 Books

- Perl Graphics Programming by Shawn Wallace

1.82 Websites

- https://www.geeksforgeeks.org/
- https://perlmaven.com/how-to-read-a-csv-file-using-perl
- https://www.tutorialspoint.com/perl/index.htm
- https://footystats.org/
- https://metacpan.org/pod/GD::Graph
- http://perl.mines-albi.fr/perl5.6.1/site_perl/5.6.1/GD/Graph.html
- https://www.perlmonks.org/?node_id=946099