Department of Computer Engineering

Government Polytechnic for Girls, Surat

JULY–2023 Vol–23

TechTrends

E-Newsletter

Vision:

To empower girls of diploma computer engineering to excel in IT Industries and serve the society.

Mission:

- To strive for academic excellence and professional competence among students and staff.
- To encourage innovative ideas among students to enhance their entrepreneurship skills.
- To provide high tech educational resources and supportive infrastructure.

Follow us on

gpgdceenewsletter@gmail.com

SASS (Syntactically Awesome Stylesheet)

INTRODUCTION

Sass stands for **S**yntactically **A**wesome **S**tylesheet. Sass is a CSS pre-processor. A CSS pre-processor is a scripting language that extends CSS by allowing developers to write code in one language and then compile it into CSS. Sass lets you use features that do not exist in CSS, like variables, nested rules, mixins, imports, inheritance, built-in functions, and other stuff. The aim of sass is to make the coding process simpler and more efficient. Sass is compatible with all versions of CSS. It reduces repetition of CSS and therefore saves time. It was designed by Hampton Catlin and developed by Natalie Weizenbaum in 2006, moreover it is free to use.

Sass is a CSS pre-processor. A CSS pre-processor is a scripting language that extends CSS by allowing developers to write code in one language and then compile it into CSS.

Mr. Harsh M. Shah

**Lecturer,
Department of
Computer Engineering**

How it works?

A browser does not understand Sass code. Therefore, you will need a Sass pre-processor to convert Sass code into standard CSS. This process is called transpiling. So, you need to give a transpiler (some kind of program) some Sass code and then get some CSS code back. It uses the .scss file extension and is fully compliant with CSS syntax. Transpiling is a term for taking a source code written in one language and transform/translate it into another language.

Sass Variables

Variables are a way to store information that you can re-use later on. With Sass, you can store information in variables, like: strings, numbers, colors, Booleans, lists, nulls, etc. Sass uses the $ symbol, followed by a name, to declare variables:

```
$myFont: Helvetica, sans-serif;
$myColor: blue;
$myFontSize: 14px;
$myWidth: 720px;
body {
  font-family: $myFont;
  font-size: $myFontSize;
  color: $myColor;
}
```

```
#header {
  width: $myWidth;
}
```

So, when the Sass file is transpiled, it takes the variables (myFont, myColor, etc.) and outputs normal CSS with the variable values placed in the CSS, like this:

```
body {
  font-family: Helvetica, sans-serif;
  font-size: 14px;
  color: blue;
}
#header {
  width: 720px;
}
```

Nesting

It provides an excellent method for reducing the amount of code you need to write. The idea is to nest your CSS selectors in such a way as to mimic your HTML hierarchy.

The following shows a basic navigation style that uses nesting:

```
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }
  li {
    display: inline-block;
  }
  a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
  }
}
```

Notice that in Sass, the ul, li, and a selectors are nested inside the nav selector.

While in CSS, the rules are defined one by one (not nested):

```
nav ul {
  margin: 0;
  padding: 0;
  list-style: none;


}
nav li {
  display: inline-block;
}
nav a {
  display: block;
  padding: 6px 12px;
  text-decoration: none;
}
```

Importing Files

Just like CSS, Sass also supports the @import directive. The @import directive allows you to include the content of one file in another. The CSS @import directive has a major drawback due to performance issues; it creates an extra HTTP request each time you call it. However, the Sass @import directive includes the file in the CSS; so no extra HTTP call is required at runtime! You do not need to specify a file extension, Sass automatically assumes that you mean a .sass or .scss file. You can also import CSS files. The @import directive imports the file and any variables or mixins defined in the imported file can then be used in the main file.

You can import as many files as you need in the main file:

```
@import "variables";
@import "colors";
@import "reset";
```

Mixin

The @mixin directive lets you create CSS code that is to be reused throughout the website. The @include directive is created to let you use (include) the mixin. For example,

```
@mixin important-text {
  color: red;
  font-size: 25px;
  font-weight: bold;
  border: 1px solid blue;
}
```

```
.danger {
  @include important-text;
  background-color: green;
}
```

The Sass transpiler will convert the above to normal CSS:

```
.danger {
  color: red;
  font-size: 25px;
  font-weight: bold;
  border: 1px solid blue;
  background-color: green;
}
```

@extend Directive

The @extend directive lets you share a set of CSS properties from one selector to another. The @extend directive is useful if you have almost identically styled elements that only differ in some small details.

The following Sass example first creates a basic style for buttons (this style will be used for most buttons). Then, we create one style for a "Report" button and one style for a "Submit" button. Both "Report" and "Submit" button inherit all the CSS properties from the .button-basic class, through the @extend directive. In addition, they have their own colors defined:

```
.button-basic  {
  border: none;
  padding: 15px 30px;
  text-align: center;
  font-size: 16px;
  cursor: pointer;
}

.button-report  {
  @extend .button-basic;
  background-color: red;
}

.button-submit  {
  @extend .button-basic;
  background-color: green;
  color: white;
}
```

After compilation, the CSS will look like this:

```
.button-basic, .button-report, .button-submit {
 border: none;
 padding: 15px 30px;
 text-align: center;
 font-size: 16px;
 cursor: pointer;
}

.button-report  {
 background-color: red;
}

.button-submit  {
 background-color: green;
 color: white;
}
```

Operators

Having the ability to perform calculations in your CSS allows you to do more, like convert pixel values into percentages. You'll have access to standard maths functions like addition, subtraction, multiplication and division. Of course, these functions can be combined to create complex calculations. In addition, Sass includes a few built-in functions to help manipulate numbers. Functions like percentage(), floor() and round() to name a few.

References

1. https://sass-lang.com
2. https://www.w3schools.com/sass/sass_intro.php

R Programming

INTRODUCTION

R is a software environment which is used to analyze statistical information and graphical representation. R allows us to do modular programming using functions.

"R is an interpreted computer programming language which was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand." It is also a software environment used to analyze statistical information, graphical representation, reporting, and data modeling.

R not only allows us to do branching and looping but also allows doing modular programming using functions. R allows integration with the procedures written in the C, C++, .Net, Python, and FORTRAN languages to improve efficiency.

Ms. Arti N. Chauhan
Lecturer,
Department of Computer Engineering

R is the implementation of the S programming language, which is combined with lexical scoping semantics.

R not only allows us to do branching and looping but also allows doing modular programming using functions. R allows integration with the procedures written in the C, C++, .Net, Python, and FORTRAN languages to improve efficiency.

In the present era, R is one of the most important tool which is used by researchers, data analyst, statisticians, and marketers for retrieving, cleaning, analyzing, visualizing, and presenting data.

History of R Programming

The history of R goes back about 20-30 years ago. R was developed by Ross lhaka and Robert Gentleman in the University of Auckland, New Zealand, and the R Development Core Team currently develops it. This programming language name is taken from the name of both the developers. The first project was considered in 1992. The initial version was released in 1995, and in 2000, a stable beta version was released.

Features of R programming

R is a domain-specific programming language which aims to do data analysis. It has some unique features which make it very powerful. The most important arguably being the notation of vectors. These vectors allow us to perform a complex operation on a set of values in a single command. There are the following features of R programming:

1. It is a simple and effective programming language which has been well developed.
2. It is data analysis software.
3. It is a well-designed, easy, and effective language which has the concepts of user-defined, looping, Conditional, and various I/O facilities.
4. It has a consistent and incorporated set of tools which are used for data analysis.
5. For different types of calculation on arrays, lists and vectors, R contains a suite of operators.
6. It provides effective data handling and storage facility.

7. It is open-source, powerful, and highly extensible software.
8. It provides highly extensible graphical techniques.
9. It allows us to perform multiple calculations using vectors.
10. R is an interpreted language.

Why use R Programming?

The important task in data science is the way we deal with the data: clean, feature engineering, feature selection, and import. It should be our primary focus. Data scientist job is to understand the data, manipulate it, and expose the best approach. For machine learning, the best algorithms can be implemented with R. Keras and TensorFlow allow us to create high-end machine learning techniques. R has a package to perform Xgboost. Xgboost is one of the best algorithms for Kaggle competition.

R communicate with the other languages and possibly calls Python, Java, C++. The big data world is also accessible to R. We can connect R with different databases like Spark or Hadoop.

In brief, R is a great tool to investigate and explore the data. The elaborate analysis such as clustering, correlation, and data reduction are done with R.

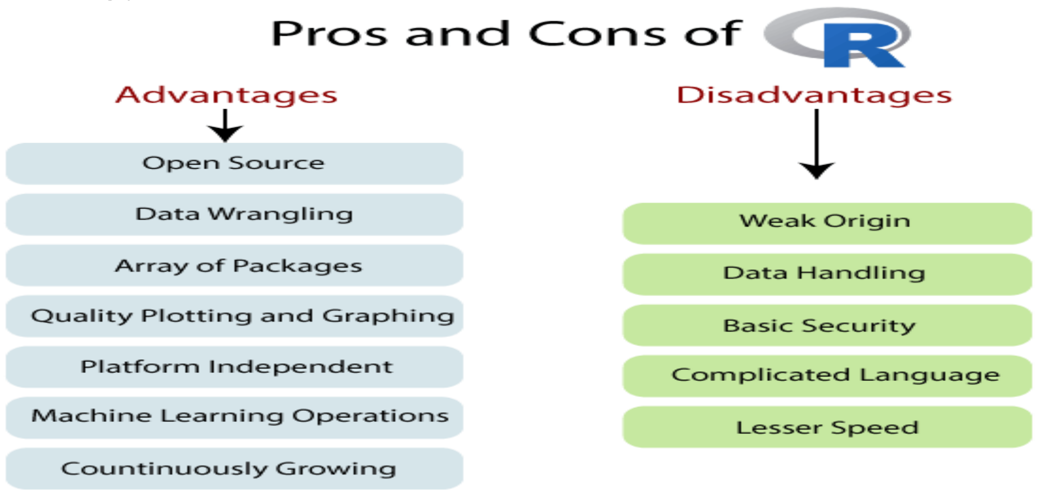
RStudio IDE

RStudio is an integrated development environment which allows us to interact with R more readily. RStudio is similar to the standard RGui, but it is considered more user-friendly. This IDE has various drop-down menus, Windows with multiple tabs, and so many customization processes. The first time when we open RStudio, we will see three Windows. The fourth Window will be hidden by default. We can open this hidden Window by clicking the File drop-down menu, then New File and then R Script.

R Advantages and Disadvantages

R is the most popular programming language for statistical modelling and analysis. Like other programming languages, R also has some advantages and disadvantages. It is a continuously evolving language which means that many cons will slowly fade away with future updates to R.

There are the following pros and cons of R



List of R packages

R is the language of data science which includes a vast repository of packages. These packages appeal to different regions which use R for their data purposes. CRAN has 10,000 packages, making it an ocean of superlative statistical work. There are lots of packages in R, but we will discuss the important one.

There are some mostly used and popular packages which are as follows:

tidyr
plotly
ggplot2
tidytext
ggraph
stringr
dplyr
reshape2
tidyquant
dichromat
list of R Packages
dygraphs
digest
leaflet
MASS
ggmap
caret
glue
e1071
shiny
sentimentr

1) tidyr

The word tidyr comes from the word tidy, which means clear. So the **tidyr** package is used to make the data' tidy'. This package works well with dplyr. This package is an evolution of the reshape2 package.

2) ggplot2

R allows us to create graphics declaratively. R provides the **ggplot** package for this purpose. This package is famous for its elegant and quality graphs which sets it apart from other visualization packages.

3) ggraph

R provides an extension of ggplot known as **ggraph**. The limitation of **ggplot** is the dependency on tabular data is taken away in ggraph.

4) dplyr

R allows us to perform data wrangling and data analysis. R provides the **dplyr** library for this purpose. This library facilitates several functions for the data frame in R.

5) tidyquant

The tidyquant is a financial package which is used for carrying out quantitative financial analysis. This package adds to the **tidyverse** universe as a financial package which is used for importing, analyzing and visualizing the data.

6) dygraphs

The dygraphs package provides an interface to the main JavaScript library which we can use for charting. This package is essentially used for plotting time-series data in R.

7) leaflet

For creating interactive visualization, R provides the **leaflet** package. This package is an open-source JavaScript library. The world's popular websites like the New York Times, Github and Flicker, etc. are using leaflet. The leaflet package makes it easier to interact with these sites.

8) ggmap

For delineating spatial visualization, the **ggmap** package is used. It is a mapping package which consists of various tools for geolocating and routing.

9) glue

R provides the **glue** package to perform the operations of data wrangling. This package is used for evaluating R expressions which are present within the string.

10) shiny

R allows us to develop interactive and aesthetically pleasing web apps by providing a **shiny** package. This package provides various extensions with HTML widgets, CSS, and JavaScript.

11) plotly

The plotly package provides online interactive and quality graphs. This package extends upon the JavaScript library **-plotly.js**.

12) tidytext

The **tidytext** package provides various functions of text mining for word processing and carrying out analysis through ggplot, dplyr, and other miscellaneous tools.

13) stringr

The stringr package provides simplicity and consistency to use wrappers for the '**stringi**' package. The stringi package facilitates common string operations.

14) reshape2

This package facilitates flexible reorganization and aggregation of data using melt () and decast () functions.

15) dichromat

The R dichromat package is used to remove Red-Green or Blue-Green contrasts from the colors.

16) digest

The digest package is used for the creation of cryptographic hash objects of R functions.

17) MASS

The **MASS** package provides a large number of statistical functions. It provides datasets that are in conjunction with the book "Modern Applied Statistics with S."

18) caret

R allows us to perform classification and regression tasks by providing the caret package. **CaretEnsemble** is a feature of caret which is used for the combination of different models.

19) e1071

The **e1071** library provides useful functions which are essential for data analysis like Naive Bayes, Fourier Transforms, SVMs, Clustering, and other miscellaneous functions.

20) sentimentr

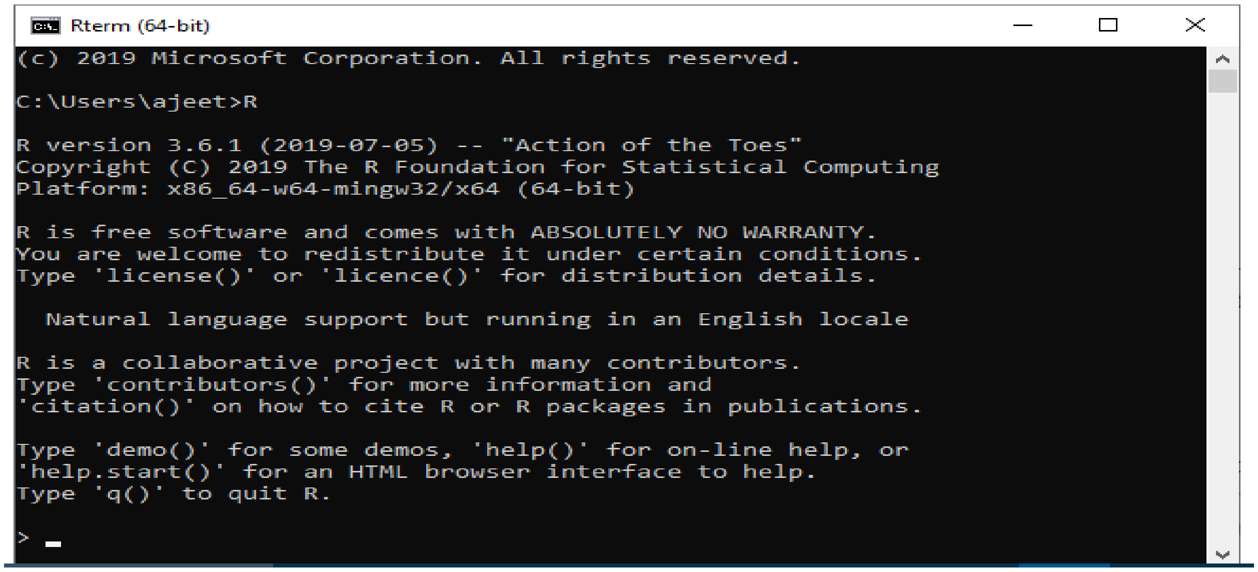
The sentiment package provides functions for carrying out sentiment analysis. It is used to calculate text polarity at the sentence level and to perform aggregation by rows or grouping variables.

Syntax of R Programming

R Programming is a very popular programming language which is broadly used in data analysis. The way in which we define its code is quite simple. The "Hello World!" is the basic program for all the languages. We can write our code either in command prompt, or we can use an R script file.

R Command Prompt

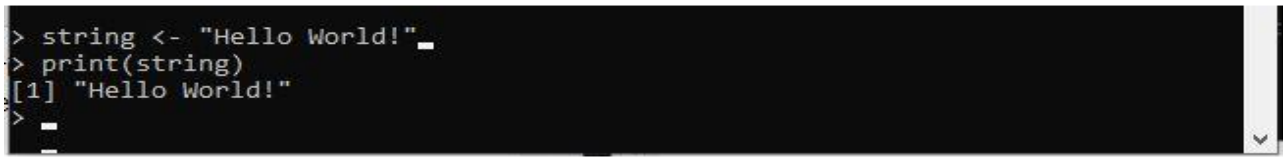
It is required that we have already installed the R environment set up in our system to work on the R command prompt. After the installation of R environment setup, we can easily start R command prompt by typing R in our Windows command prompt. When we press enter after typing R, it will launch interpreter, and we will get a prompt on which we can code our program.



```
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\ajeet>R

R version 3.6.1 (2019-07-05) -- "Action of the Toes"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 
```

"Hello, World!" Program

The code of "Hello World!" in R programming can be written as:



```
> string <- "Hello World!"
> print(string)
[1] "Hello World!"
> 
```

In the above code, the first statement defines a string variable string, where we assign a string "Hello World!". The next statement print() is used to print the value which is stored in the variable string.

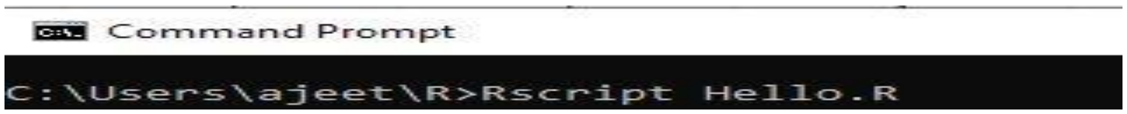
R Script File

The R script file is another way on which we can write our programs, and then we execute those scripts at our command prompt with the help of R interpreter known as **Rscript**. We make a text file and write the following code. We will save this file with .R extension as:

Hello.R

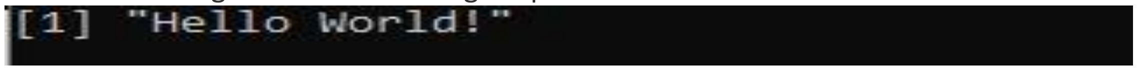
```
1.  string <-"Hello World!"
2.  print(string)
```

To execute this file in Windows and other operating systems, the process will remain the same as mentioned below.



```
Command Prompt
C:\Users\ajeet\R>Rscript Hello.R
```

When we press enter it will give us the following output:



```
[1] "Hello World!"
```

Creating Variables

Variables are containers for storing data values.R does not have a command for declaring a variable. A variable is created the moment you first assign a value to it. To assign a value to a variable, use the <- sign. To output (or print) the variable value, just type the variable name:

Example:

```
name<- "John"
age<- 40
name   #output "John"
age    # output 40
```

Print / Output Variables

Compared to many other programming languages, you do not have to use a function to print/output variables in R. You can just type the name of the variable:

Example:

```
name <- "John Doe"
name # auto-print the value of the name variable
```

Basic Data Types

Basic data types in R can be divided into the following types:

- numeric - (10.5, 55, 787)
- integer - (1L, 55L, 100L, where the letter "L" declares this as an integer)
- complex - (9 + 3i, where "i" is the imaginary part)
- character (a.k.a. string) - ("k", "R is exciting", "FALSE", "11.5")
- logical (a.k.a. boolean) - (TRUE or FALSE)

We can use the class() function to check the data type of a variable.

Statistics Introduction

Statistics is the science of analysing, reviewing and conclude data.
Some basic statistical numbers include:

- Mean, median and mode
- Minimum and maximum value
- Percentiles
- Variance and Standard Deviation
- Covariance and Correlation
- Probability distributions

The R language was developed by two statisticians. It has much built-in functionality, in addition to libraries for the exact purpose of statistical analysis.

References

1. https://www.javatpoint.com/r-tutorial
2. https://www.w3schools.com/r/default.asp