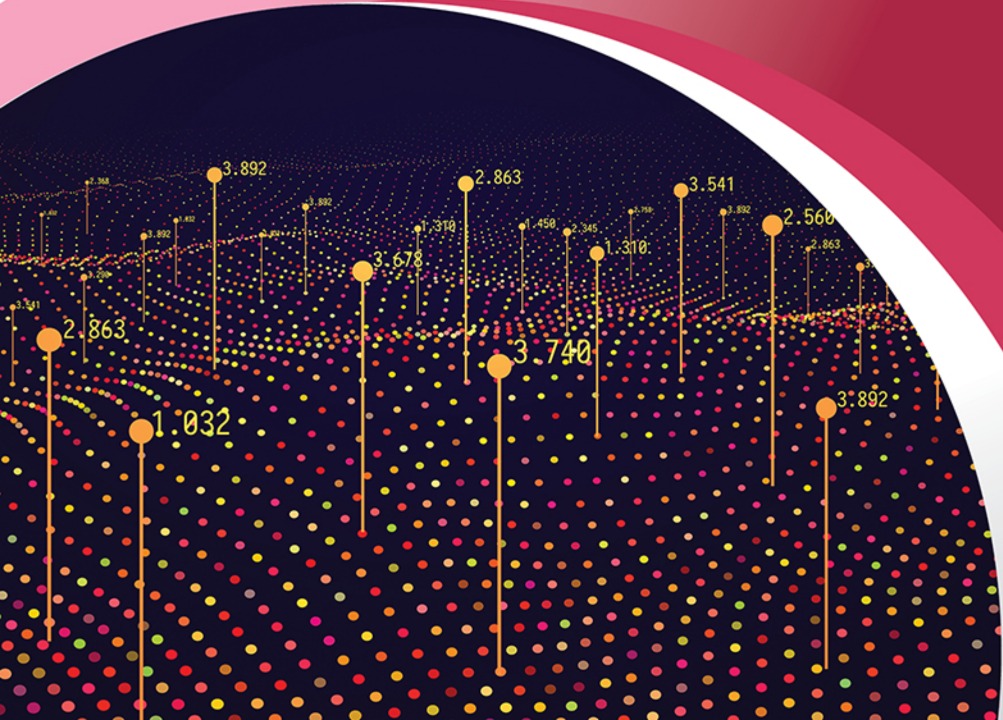
WHAT EVERY ENGINEER SHOULD KNOW

CRC Press
Taylor & Francis Group

What Every Engineer Should Know About Data-Driven Analytics

Satish Mahadevan Srinivasan
and Phillip A. Laplante



What Every Engineer Should Know About Data-Driven Analytics

What Every Engineer Should Know About Data-Driven Analytics provides a comprehensive introduction to the theoretical concepts and approaches of machine learning that are used in predictive data analytics. By introducing the theory and providing practical applications, this text can be understood by students of every engineering discipline. It offers a detailed and focused treatment of the important machine learning approaches and concepts that can be exploited to build models to enable decision making in different domains.

- Utilizes practical examples from different disciplines and sectors within engineering and other related technical areas to demonstrate how to go from data, to insight, and to decision making.
- Introduces various approaches to building models that exploit different algorithms.
- Discusses predictive models that can be built through machine learning and used to mine patterns from large datasets.
- Explores the augmentation of technical and mathematical materials with explanatory worked examples.
- Includes a glossary, self-assessments, and worked-out practice exercises.

Written to be accessible to non-experts in the subject, this comprehensive introductory text is suitable for students, professionals, and researchers in engineering and data science.

What Every Engineer Should Know About Data-Driven Analytics

Satish Mahadevan Srinivasan
Phillip A. Laplante

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

Cover image: Shutterstock

First edition published 2023
by CRC Press
6000 Broken Sound Parkway NW, Suite 300, Boca Raton, FL 33487-2742

and by CRC Press
4 Park Square, Milton Park, Abingdon, Oxon, OX14 4RN

CRC Press is an imprint of Taylor & Francis Group, LLC

ISBN: 978-1-032-23543-1 (hbk)
ISBN: 978-1-032-23540-0 (pbk)
ISBN: 978-1-003-27817-7 (ebk)

DOI: 10.1201/9781003278177

Typeset in Times
by SPi Technologies India Pvt Ltd (Straive)

Dedication

Each author would like to thank his respective family members, parents, grandparents, great-grandparents, and so on down the line. Without these ancestors the authors and this book would never exist.

This book is dedicated to the memory of our dear colleague and gentle friend Partha Mukherjee who sadly passed away before this book was completed.

Contents

Preface

INTRODUCTION

This book provides a comprehensive introduction to the machine learning theoretical concepts and approaches that are used in predictive data analytics through practical applications (case studies and examples). Using machine learning we can build predictive models that can be used to mine patterns from large datasets. Such models can also be tailored to reason why it sees a particular pattern in the dataset.

Mining large datasets from different domains (healthcare, financial, sports, manufacturing, social media, advertisement, etc.) needs a different type of mindset and skillset for predictive model building. This textbook will offer a detailed and focused treatment of the important machine learning approaches and concepts that can be exploited to build models to enable decision making in different domains. Whenever required, technical and mathematical materials will be augmented with explanatory worked examples to illustrate their importance in the given context.

Through case studies, this book demonstrates how to go from data, to insight, to decision making. In each of the case studies, we have taken a unique approach to building models that exploit different algorithms. In addition to that, the case studies also highlight the techniques used for validating and evaluating predictive models. The book, informed by the author's many years of teaching machine learning, and working on predictive data analytics projects, is suitable for use by graduates, professionals, and researchers in the area of data science.

AUDIENCE

This book is intended for professional data engineers, software engineers, systems engineers, and senior and graduate students of analytics and artificial intelligence. Much of the material is derived from the graduate-level "Data Analytics" course taught at Penn State's Great Valley School of Graduate and Professional Studies and online through its World Campus, where the authors work. The typical student in that course has five years of work experience in any of a variety of technical or business roles and an undergraduate degree in engineering, science, or business. Typical readers of this book will have one of the following or similar job titles:

Data analyst
Data scientist
IT analyst
Software engineer
Systems engineer
Sales engineer
Systems analyst
[XYZ] engineer (where "XYZ" is an adjective for most engineering disciplines, such as "electrical," "computer," or "mechanical")

Project manager
Business analyst
Technical architect
Lead architect
Product owner

Many others can benefit from this text including the users of complex systems and other stakeholders.

COURSE ADOPTION

This text is suitable for use in the following courses as a primary reference: predictive analytics, machine learning, data-driven decision making, and data science.

It can also be used as a secondary reference, typically in courses such as data mining, statistics, natural language processing, and artificial intelligence.

ERRORS

The authors have tried to uphold the highest standards for accuracy in terms of fact and quality of presentation. Despite these best efforts and those of the reviewers and publisher, there are still likely a few errors to be found. Therefore, if you believe that you have found an error—whether it is a referencing issue, factual error, or typographical error—please contact the authors at sus64@psu.edu or pal11@psu.edu.

Acknowledgments

There are many who helped in the development and writing of this book, directly or indirectly. In particular, we would like to express our gratitude to our Senior Publisher and Sponsor at T&F, Allison Shatkin, for her support and constant encouragement along the way. We would also like to thank the following students for assisting with some of the code examples, figures, and tables.

Shahed Mahbub, Raghava Rao Sunkanapally, Nikhitha Kunduru, Junjun Tao, Shichu Chen, Deeksha Joshi, Haruka George, Dinesh Kumar Chowdhary and Manish Ranjan.

We would also like to thank the following colleagues for their encouragement and ideas, and for reviewing portions of the draft manuscript.

Raghvinder S. Sangwan, Youakim Badr, Partha Mukherjee

There are surely others who have provided us with additional inspiration along the way and the omission of their names is inadvertent, but we would like to thank them all collectively.

About the Authors

Satish Mahadevan Srinivasan (Satish Srinivasan) is an Associate Professor of Information Science and a member of the graduate faculty at the Pennsylvania State University. His research, teaching, and consulting focus on predictive analytics particularly with respect to text preprocessing, building data pipelines, and artificial intelligence systems.

Satish Srinivasan received his B.E. in Information Technology from Bharathidasan University, India, and his M.S. in Industrial Engineering and Management from the Indian Institute of Technology Kharagpur, India. He earned his Ph.D. in Information Technology from the University of Nebraska at Omaha. Prior to joining Penn State Great Valley, he worked as a postdoctoral research associate at the University of Nebraska Medical Center, Omaha. Dr. Srinivasan teaches courses related to database design, data mining, data collection and cleaning, computer, network and web securities, and business process management. He has authored or edited 2 book chapters and more than 50 papers, articles, reviews, and editorials.

Phillip A. Laplante is Professor of Software and Systems Engineering and a member of the graduate faculty at the Pennsylvania State University. His research, teaching, and consulting focus on software quality, particularly with respect to artificial intelligence and critical systems.

Prior to his academic career, Dr. Laplante spent nearly a decade as a software engineer and project manager working on avionics (including the Space Shuttle), CAD, and software test systems. He has authored or edited 39 books and more than 300 papers, articles, reviews, and editorials.

Dr. Laplante received his B.S., M.Eng., and Ph.D. in computer science, electrical engineering, and computer science, respectively, from the Stevens Institute of Technology and an M.B.A. from the University of Colorado at Colorado Springs. He is a Licensed Professional Engineer in Pennsylvania and is a Certified Software Development Professional. He is a fellow of the IEEE and SPIE and a member of numerous professional societies and program committees.

1 Data Collection and Cleaning

In the 21st century, data are everywhere. Across different application areas, data are being collected at an unprecedented rate. Decisions in past were purely made based on guesswork, expert opinions, or by using constructed models; but these days decisions are made solely based on the data available. Large amounts of data or so-called "Big Data" has the potential to revolutionize different aspects of modern society. Application areas of Big Data include scientific research, financial services, retail manufacturing, biological and physical sciences, healthcare, transportation, environmental modeling, energy saving, homeland security, social network analysis, and much more [1].

So how should an engineer or data scientist approach the analysis of all this data? Here is the general approach. After recording the data in the repositories, the next step is to curate and analyze the data. The discussions in this book will focus on analyzing the data using the tools and techniques in the domain of machine learning, data mining, and predictive analytics. The potential uses of Big Data are exciting. For example, in the education sector, the collected data related to the academic performance of every student can be used as a guide for delivering future instructions. In the healthcare sector, Information Technology (IT) and data analytics can reduce the cost of healthcare while improving its quality and outcomes by making preventive care more personalized and affordable. In the United States alone, the advent of Big Data technology can result in IT savings for the healthcare sector, which is estimated to be close to 300 billion dollars [1, 2].

While the potential benefits are significant, there remain many technical challenges to be addressed for realizing the potential of Big Data. Challenges exist along several different dimensions, namely: *Volume*, *Variety*, *Velocity*, *Veracity*, and *Value* [1–3].

The dimension of *volume* represents the amount of data. The collected data can be either structured (numeric, relational model) or unstructured (non-numeric, text type, video, audio) which is represented by the dimension of *variety*. The *velocity* dimension represents the rate at which the data arrive over time (every second, every minute, hourly, daily, etc.) and also accounts for the time within which the data have to be acted upon. The dimension of *veracity* is all about the validity and the correctness of data, i.e., how accurate and usable are the data? Finally, how valuable are the data captured by the *value* dimension? [1–3]

A data analysis pipeline to deal with the different dimensions of Big Data is shown in Figure 1.1. The data analysis pipeline includes multiple phases, namely the *data acquisition/recording phase*, the *extraction/cleaning/annotation phase*, the *integration/aggregation/representation phase*, the *analysis/modeling phase*, and the

DOI: 10.1201/9781003278177-1

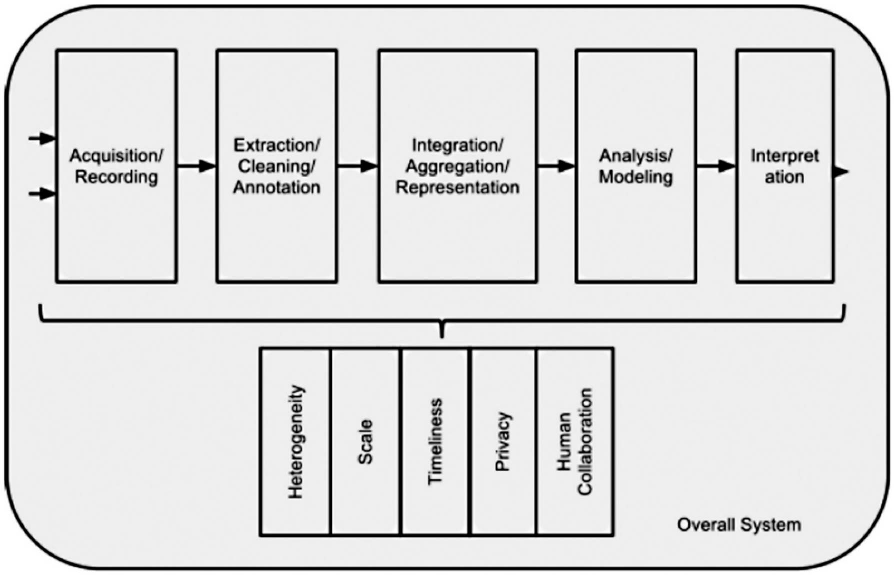


FIGURE 1.1 The Big Data analysis pipeline. (Source: Figure adapted from Challenges and Opportunities with Big Data, A community white paper developed by leading researchers across the United States, https://cra.org/ccc/wp-content/uploads/sites/2/2015/05/bigdata whitepaper.pdf.)

interpretation phase. Each of these phases is crucial and has its own set of challenges that needs to be addressed [1].

In the *data acquisition phase*, the challenge is to generate the right metadata to describe what data needs to be recorded and measured. *Data provenance*[1] is also an issue in this phase. Any data not originating from its source would have gone through several stages of transformation or edition. Therefore, an error in the data processing can render subsequent analysis useless. Thus, in the data analysis pipeline, it is important to carry both the provenance of data and metadata together [1].

The *information extraction and cleaning phase* is responsible for converting the collected data in a format that is ready for analysis [1].

The *data integration, aggregation, and representation phase* is devoted to hiding the heterogeneity of the data and making it available in the required format for analysis and modeling [1].

The *query processing, data modeling, and analysis phase* is devoted to building tools and techniques for the effective large-scale analysis of data in a completely automated manner [1].

Finally, the *interpretation phase* provides the means for the decision makers to interpret the results of the analysis and make Big Data more actionable [1].

Now let us focus our discussion on the challenges associated with the data analysis pipeline. The challenges involved in the analysis pipeline can be classified as *heterogeneity*, *scale*, *timelines*, *privacy*, and *human collaboration* [1].

The *heterogeneity and incompleteness* are challenges because the data representing the same entity in different sources lack a consistent format, are erroneous, and incomplete. Again, managing large and rapidly increasing volumes of data challenges the limitation of the tools, techniques, and the algorithms that process them. Often analyzing the large datasets takes a longer time, which is a challenge because with time, the value of the data diminishes for a decision maker [1].

Privacy in Big Data is a major concern as still there is no established protocol that allows the sharing of private data while limiting the disclosure and ensuring sufficient data utility [1].

Finally, the advances in computing analysis have still left a gap in identifying many patterns that are only detectable by humans. This is a challenge to the automation of the data analysis pipeline as *human intervention and collaboration* are of paramount importance in the successful realization of the potentiality of the data [1].

DATA-COLLECTION STRATEGIES

Big Data collection is the methodical approach to collecting and analyzing massive amounts of information from a variety of sources. Earlier it was indicated that Big Data collection entails collecting structured, semi-structured, and unstructured data generated by sources such as people, computers, and sensors. The value of the data does not depend on its quantity but on its quality. Structured data are highly organized and exist in a predefined format. On the other hand, there is no predefined format for unstructured data. Therefore, it exists in the format in which it was generated. Semi-structured data on the other hand is a mix of structured and unstructured data. For example, data related to GPS[2] coordinates is an example of structured data. Data collected from social media sites is a good example of unstructured data. Data like email addresses and their contents are good examples of semi-structured data. Data can also be classified as quantitative and qualitative. Quantitative data have numerical forms such as statistics and percentages, while qualitative data are more descriptive in nature, like sex, religion, etc. The typical sources of data include [1–3]:

- Operational systems producing transactional data,
- IoT endpoint device,
- Social media data from users and customers,
- Location data and other vitals from the smartphone devices,
- and more.

To provide access to accurate and consistent data, integration of data from several sources into a *data warehouse*[3] is very vital. Data warehouses require and provide extensive support for data cleaning. Since data warehouses need to load and continuously refresh huge amounts of data from a variety of sources, the probability of data being *dirty*[4] or inconsistent is very high. In order to provide effective and timely responses to queries, data cleaning in data warehouses is important and is supported by a so-called *ETL* process. The *ETL* process comprises of three phases namely the *extraction phase*, the *transformation phase*, and the *loading phase* [1, 3].

During the *extraction phase* data from several sources are collected. The source systems files (data) can be in multiple file formats including flat files with delimiters (CSV format), XML, non-relational database structures including IMS (Information Management Systems), data structures such as VSAM (Virtual Storage Access Method) or ISAM (Indexed Sequential Access Method) or data fetched through screen-scraping or web spidering. A source for data could also be a legacy system in which the files are in the arcane format. In this step, the schema (metadata) from different sources is extracted and translated. In addition to the schemas, instances are also extracted from the data source and are moved/stored into intermediate data sources before loading them into the data staging area [1, 3].

In the *transformation stage*, multiple data manipulation steps are performed such as moving, splitting, translating, merging, sorting, and pivoting data, all in accordance with the data quality rules. In this phase, the translated schema is matched and integrated to create a unique schema for the data warehouse. In addition to the schemas, the instances are also matched and integrated into the data staging area. A series of rules or functions is applied on the data extracted from the data sources which includes selective loading of the columns, translation of the coded values, encoding of free-form values, deriving calculated values, sorting, joining data from multiple sources, aggregation, transposing, etc. A large number of tools of varying functionalities are available to support these tasks, but often a significant portion of the cleaning and transformation work has to be done manually or by a low-level program that is difficult to write and maintain [1, 3].

Finally, in the *loading stage*, the translated and integrated schema from the transformation is implemented on the target (data warehouse) system and the instances from the data staging area are filtered, aggregated, and loaded into the data warehouse. Depending on the organization requirements, the process of loading the extracted data into the data warehouse is frequently done on a daily, weekly, or monthly basis [1, 3].

DATA PREPROCESSING STRATEGIES

Data preprocessing is performed to transform the raw data into a useful and efficient format. Major tasks involved in the data preprocessing stage are data cleaning, data integration, data reduction, data transformation, and data discretization. Data collection results in accumulating noisy, missing, and inconsistent data which need to be corrected for downstream analysis. Data preprocessing avoids any potential problems with accuracy, completeness, consistency, timeliness, believability, and interpretability [1–5].

Data Cleaning: Data can have many irrelevant and missing parts. Therefore, it is important to perform *data cleaning* to deal with missing and noisy data. Missing data can be handled by either ignoring the entire tuple or by imputing the missing values. On the other hand, noisy data can be handled by using the binning method, regression, or clustering [1, 3–5].

Data Integration: In this step tasks such as entity identification, removal of redundant data, and data deduplication are performed. Data fused from different sources and single entities from two different sources can have attributes that are referred to different naming conventions or might be referring to the same characteristic. Redundant attributes can be detected by correlation and covariance analysis. Duplicated data can be identified by recognizing the repeated tuples or by performing descriptive statistics [1, 3–5].

Data Reduction: Data reduction is the process to obtain a reduced representation of the dataset that is much smaller in volume but yet produces the same analytical results. A database or a data warehouse may store terabytes of data and analyzing this amount of data can take a very long time. Therefore, it is important to perform data reduction. Data reduction strategies include dimensionality reduction (attribute subset selection, attribute creation, wavelet transformation, principal components analysis (PCA), etc.), numerosity reduction (regression and log-linear models, histograms, clustering, sampling, data cube aggregation, etc.), data compression (string compression, audio/video compression, etc.), etc [1, 3–5].

Data Transformation: Data transformation is the process of converting data from one type to another. The strategy here is to map the values of a given attribute to a new set. The motivation behind the data transformation is to remove the skewness in the data to achieve symmetric distribution. Transforming data makes it easier to visualize, and to improve the data interpretability. Data transformations involve different operations such as smoothing to remove noise from the data, attribute/feature construction, normalization i.e., to scale data to fall within a smaller and specified range, discretization where raw values are replaced by intervals or conceptual labels, min-max normalization, z-score normalization, normalization by decimal scaling, concept hierarchy generation, etc [1, 3–5].

Data Discretization: Data discretization is a process of converting a large number of data values into smaller ones. Data discretization is performed in order to make data evaluation, visualization, and management easier [1, 3–5].

A brief introduction to the basics of R and Python programming is provided here, which will be very helpful for the readers to navigate through the other chapters in this book [2].

PROGRAMMING WITH R

Data Types in R

The basic data types in R are character, numeric, integer, and logical. The assignment operator (←) can be used to assign any data to a variable.

```
For example
x <- 10
y <- 'This is a sample string'
```

```
# Let us now print the value of x & y
x
[1] 10
y
[1] "This is a sample string"
```

Data Structures in R

The commonly used data structures in R include vectors, matrices, dataframe, lists, and factors.

Vectors—A vector is a sequence of data elements of the same basic type. A vector can be defined as

```
a <- c(1,2,5.3,6,-2,4) # numeric vector
b <- c("one", "two", "three") # character vector
c <- c(TRUE, TRUE, TRUE, FALSE, TRUE, FALSE) #logical vector
```

Matrices—A matrix is a collection of data elements arranged in a two-dimensional rectangular layout. For example, the matrix A of size 2 × 3 can be created as:

```
A = matrix(
c(2, 4, 3, 1, 5, 7), # the data elements
nrow=2,              # number of rows
ncol=3,              # number of columns
byrow = TRUE)        # fill matrix by rows
```

The matrix A can then be viewed by just entering the matrix name

```
A
    [,1] [,2] [,3]
[1,]  2    4    3
[2,]  1    5    7
```

The elements of the matrix can be directly accessed by specifying the corresponding row and column number as shown

```
A[2, 3]      # element at 2nd row, 3rd column
[1] 7
```

Dataframe—Compared to matrices, in a dataframe, the different columns can have different modes (numeric, character, factor, etc.). A dataframe in R can be created as

```
d <- c(1,2,3,4)
e <- c("green", "blue", "green", NA)
f <- c(TRUE,TRUE,TRUE,FALSE)
```

```
mydata <- data.frame(d,e,f) # create a data frame called
  mydata
names(mydata) <- c("ID","Color","Passed") # variable names
```

Now let us display the created dataframe *mydata*

```
mydata
  ID Color Passed
1  1  green  TRUE
2  2  blue   TRUE
3  3  green  TRUE
4  4  <NA>   FALSE
```

Lists—A list is defined as an ordered collection of objects. The objects can be of different types and possibly unrelated. For example,

```
# Example of a list with 4 components -
# a string, a numeric vector, a matrix, and a scaler
w <- list(name="Sam", mynumbers=xyz, mymatrix=abc, age=5.3)
# Example of a list containing two lists
v <- c(list1,list2) # where list1 and list2 are lists
```

Factors—Conceptually speaking factors are variables in R which take on a limited number of different values. These variables are often referred to as categorical variables.

```
# Define the variable gender with 20 "male" entries and 30
  "female" entries
gender <- c(rep("male",20), rep("female", 30))
gender <- factor(gender)
# The gender variable stores gender as 20 1s and 30 2s and
  associates 1 to male and 2 to female

# R now treats gender as a nominal variable
summary(gender)
female   male
  30      20
```

An ordered factor is used to represent an **ordinal variable**.

```
# A variable rating can be coded as "large", "medium", "small'
rating <- c("large", "medium", "small")
rating <- ordered(rating)
# recodes the variable rating to 1,2,3 and associates
# 1=large, 2=medium, 3=small internally
# R now treats rating as ordinal
```

Package Installation in R

To install a package in R, use the following command:

```
install.packages('chron') # "chron" is the name of the package
```

Once the package is installed, the command to load the installed package is library(). For example,

```
 library(chron)
```

Reading and Writing Data in R

R supports various packages that allow developers to read data from various file formats and load them into objects or write data to various file formats. Here, we will discuss about how to read data from the CSV file and write data into the CSV file.

To read data from a CSV file and assign it to a dataframe "df," use the following command:

```
df <- read.csv("<file-name>", header=TRUE, sep=",")
```

Note here that

df : Name of the dataframe
<file-name>: Name of the source file with complete path to its location. If path is not specified, the file will be read from the working directory.
Header: Setting this parameter to TRUE indicates that the source file has the names of the columns to be read.
Sep: this parameter is used to indicate the delimiter in the source file, traditionally the delimiter for a csv is a comma (,)

To write the contents from the dataframe to a CSV file, use the following command:

```
write.csv(df, file = "<file-name>",row.names=FALSE)
```

Note:

df: the dataframe that you want to write to the CSV file.
<file-name>: target file name with complete path to its location. If path is not specified, the file will be written to the working directory.
row.names: this parameter will indicate whether you want to write the row.names (the index number of each row) to the target file.

Some commonly used functions that might come in handy in R programming are:

- length(object) # number of elements or components
- str(object) # structure of an object
- class(object) # class or type of an object
- names(object) # names
- c(object,object,…) # Combine objects into a vector
- cbind(object, object, …) # Combine objects as columns
- rbind(object, object, …) # Combine objects as rows
- rm(object) # delete an object
- colnames(object) # retrieve the column names of a matrix like object (matrix, dataframe, etc.)

Now consider some general programming examples.

Using the FOR Loop in R

The for loop in R can be implemented with the traditional syntax

```
# Creating a vector and assigning values to it
new <- c(2010,2011,2012,2013,2014,2015)
# for loop to display the values in the vector
for (year in new){  print(paste("The year is", year))}
```

Using the WHILE Loop in R

We will be using the same vector "new" that was created for the previous example.

```
# creating an increment counter
i <- 2010
# while loop to display the values in the vector
while (i < 2016) {print(i); i = i+1}
```

Using the IF-ELSE Statement in R

We will try to determine if the years mentioned in the vector "new" is a leap-year or not. For this we first need to define a function which checks the same.

```
# Function definition
is.leapyear=function(year){
   return(((year %% 4 == 0) & (year %% 100 != 0)) | (year %%
      400 == 0))
}
# Using the for loop and the if-else statements to determine
  the result
# the paste0 function is used to print different types of data
  together, both text and variable value
```