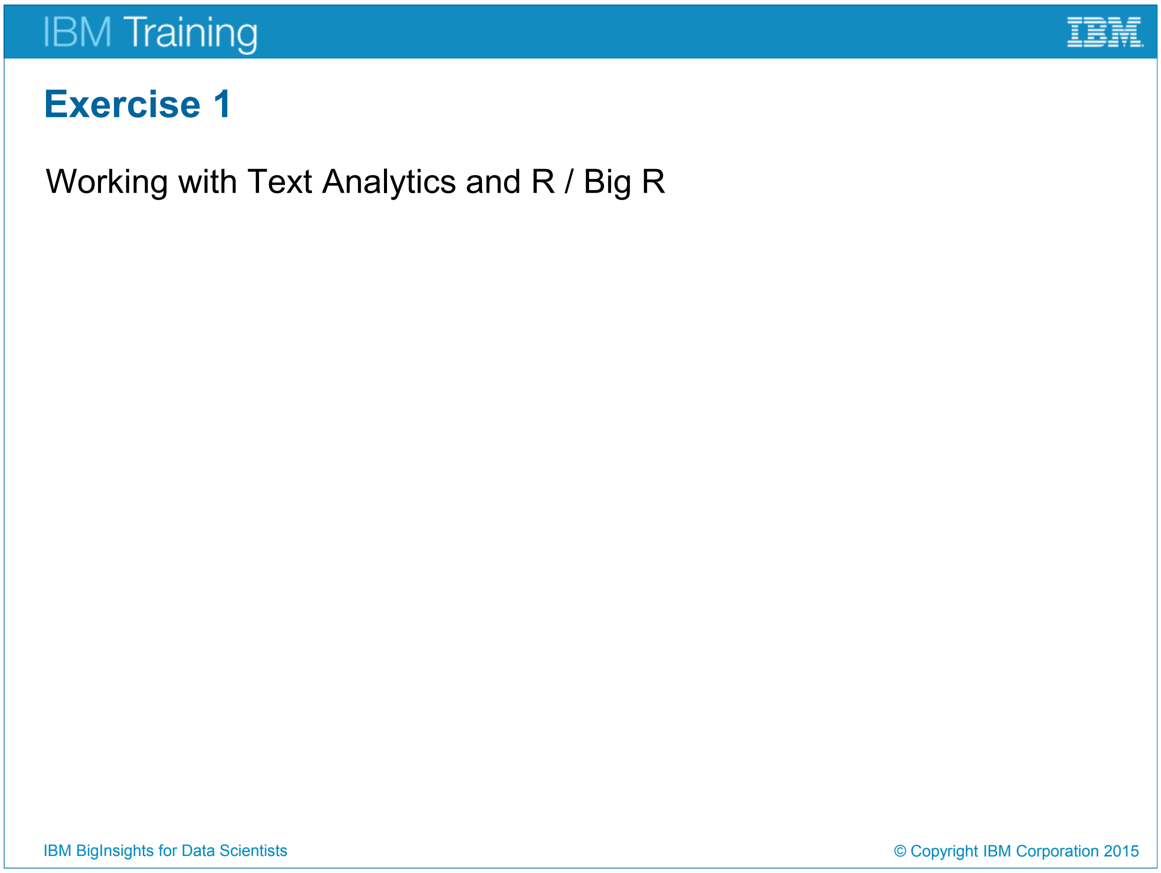


IBM Training

Exercise 1

Working with Text Analytics and R / Big R

IBM BigInsights for Data Scientists

© Copyright IBM Corporation 2015

Exercise 1: Working with Text Analytics and R / Big R

Exercise 1: Working with Text Analytics and R / Big R

Purpose:
You will create a new text analytics web tooling project and load documents to scan for certain keywords. You will start up the R console and run basic commands on it. You will also load the BigR libraries and run basic BigR operations.

Estimated time:	**1 hour**
User/Password:	**biadmin/biadmin**
	root/dalvm3
Services Password:	**ibm2blue**

Important: Before doing this exercise, ensure that your access and services are configured and running. Check that:

- /etc/hosts displays your environment's IP address
- in the Ambari console, ensure that all BigInsights services are running

If you are unsure of the steps, please refer to Unit 1, Exercise 1 to ensure that your environment is ready to proceed. You should review the steps in Task 1 (Configure your image) and Task 2 (Start the BigInsights components).

Task 1. Launching the text analytics Web Tooling module.

IBM BigInsights provides a Web Tooling module that makes text analytics easy. In this task, you will see how to use the Web Tooling module to create a project, and load some documents to start the analysis.

1. To open a new terminal, right-click the desktop, and then click **Open in Terminal**.

   You will review the set of files that you will be using for this exercise.

2. Navigate to **/home/biadmin/labfiles/ta/WatsonData/Data/**, and then type `ls` to see the files.

   These are sample blog files by IBM.

3. Launch **Firefox**, and then if necessary, navigate to the **Ambari** login page, **http://ibmclass.localdomain:8080**.
4. Log in to the **Ambari** console as **admin/admin**, and ensure that all of the components have started.
5. Click the **Knox** component, click **Service Actions**, and then click **Start Demo LDAP**.

6. Click **OK**, and then when the **Start Demo LDAP** process is complete, click **OK** again.

   Leave the Ambari tab open in Firefox for Task 2 of this exercise.

7. To launch the BigInsights home page, open a new browser tab and type:

   https://ibmclass.localdomain:8443/gateway/default/BigInsightsWeb/index.html

   There is a bookmark saved on the toolbar. The id and password is guest / guest-password, but that is also saved for you in the lab envrionment.

   Note: You may need to wait for a minute before the two links display (BigSheets and Text Analytics).

8. Click the **Text Analytics** link to open up the Web Tooling module.

   You are going to create a project and load in some documents to do a text extraction for the Watson keyword.

9. On the **Projects** tab, click **New** ➕ to create a new project.

10. Beside **Name**, type **Watson**, and then click **Create**.

    You will load in a set of documents.

11. In the **Documents** pane, click **Add Documents** ➕.

12. Click **Browse**, and then navigate to the **biadmin/labfiles/ta/WatsonData/Data/** directory.

13. Shift+click the first and the last document to select all of the documents, click **Open**, and then click **Add**.

    Notice the documents contain XML markup. You can easily remove all the tags at once.

14. Click the **Remove Tags** button in the upper right corner.

    The goal is to find blogs about the Watson computer. The extractor should look for the word Watson.

15. Click the **New Literal** ABC button at the top of your Watson project pane.

    A textbox appears in the project pane.

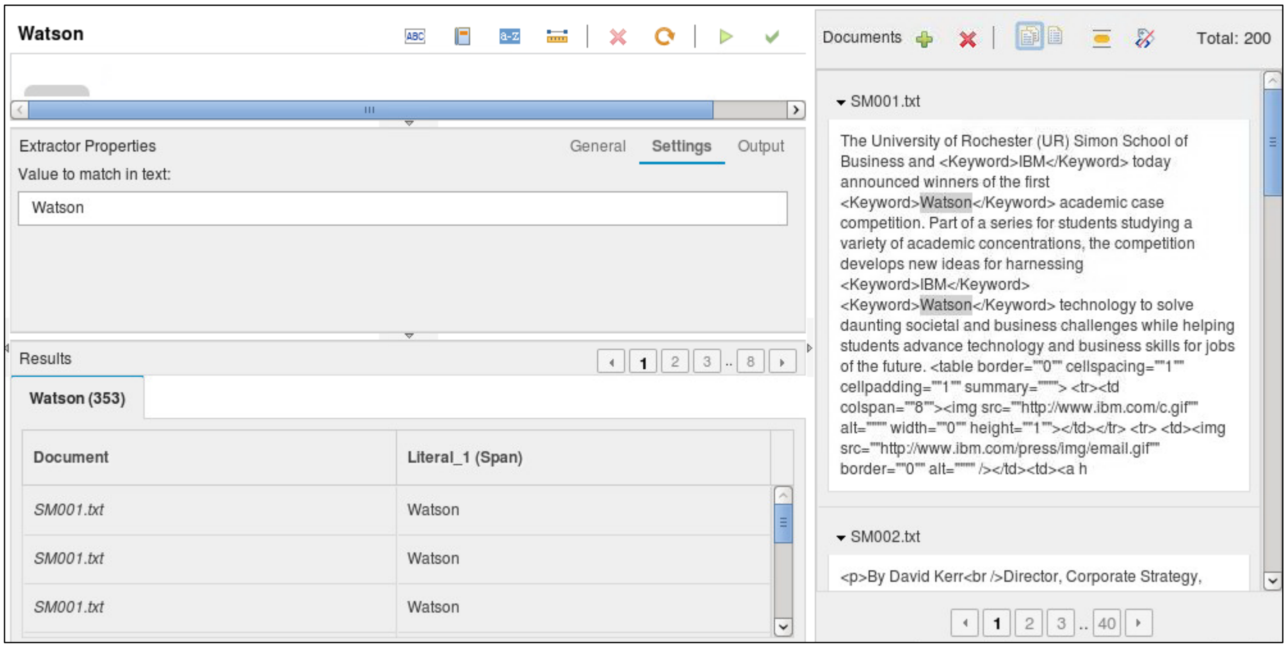
16. Type **Watson** in the textbox, and then press **Enter** to confirm.

17. Select the **Watson** literal, and then click the **Run Selected** to run the extractor.

    Hint: You may need to scroll the project pane further to the right to see the rest of the icons.

    It will take a few seconds to run. All of the occurrences of the word Watson will be highlighted in the Documents pane (on the right). You can see details of each match in the Results pane (on the bottom).

    The results appear as follows:



Watson

Extractor Properties — General | Settings | Output

Value to match in text: Watson

Results — Watson (353)

Document	Literal_1 (Span)
SM001.txt	Watson
SM001.txt	Watson
SM001.txt	Watson

Documents — Total: 200

SM001.txt

The University of Rochester (UR) Simon School of Business and <Keyword>IBM</Keyword> today announced winners of the first <Keyword>Watson</Keyword> academic case competition. Part of a series for students studying a variety of academic concentrations, the competition develops new ideas for harnessing <Keyword>IBM</Keyword> <Keyword>Watson</Keyword> technology to solve daunting societal and business challenges while helping students advance technology and business skills for jobs of the future. <table border=""0"" cellspacing=""1"" cellpadding=""1"" summary=""""> <tr><td colspan=""8""><img src=""http://www.ibm.com/c.gif"" alt="""" width=""0"" height=""1""></td></tr> <tr> <td><img src=""http://www.ibm.com/press/img/email.gif"" border=""0"" alt="""" /></td><td><a h

SM002.txt

<p>By David Kerr<br />Director, Corporate Strategy,

    You will put some context around the word Watson, creating a dictionary of terms frequently associated with the Watson computer.

18. Click **New Dictionary** in the middle pane, and then type **PositiveClues** in the text box that appears in the canvas.

    Now, you will want to peruse your documents for words. For example, Watson is commonly associated with IBM. You will want to add this to your PositiveClues dictionary.

19. Click on the **PositiveClues** extractor, and select **Settings** (this is located below your canvas).

20. Click **Add Term**, and then type **IBM**.

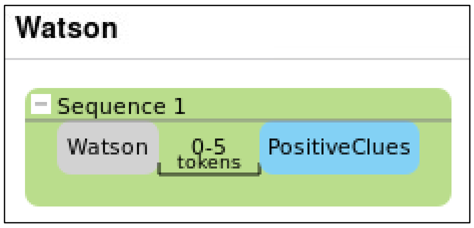
    Hint: You can quickly add in words by typing and hitting enter.

21. Add in **computer**, **computing**, **solutions**, and **technology** as positive clues for the context around the Watson.
22. To run the **PositiveClues** extractor, select it in the canvas, and then click **Run Selected**.

    The words will be highlighted. Review the results pane for details of the search.

    You are going to combine the Watson extractor with the PositiveClues extractor with a proximity rule so that when the clues appear within five words, you know that the Watson keyword is of the correct context.
23. Click **New Proximity Rule** , and then type **0-5**.
24. Drag the **proximity rule** to the Watson extractor and release it when you see a blue bar, indicating that they are joined together.
25. Drag and drop the **PositiveClues** to the right side of the rule as well.

    The results appear as follows:



    A new sequence is now created with all words of Watsons within five tokens of one of your dictionary terms.
26. Click the **Sequence 1**, and then in the **Extractor Properties** pane, click **Output**.
27. Rename **Sequence 1** to **WatsonSpan**.
28. Rename **Literal_1** to **Watson**.

    Do not rename PositiveClues.
29. Select the extractor, and then click **Run Selected**.
30. Click a few of the rows in the Results pane. You can see the words of Watson that comes with the dictionary words.

    This only extracts the word Watson with the clues that comes after. If you want to have the same context in front of the Watson word as well, what would you need to do?

    Similarly, you may have false positives, such as a person's name of Watson. You would not want to include that in your search. What would you need to do there?

    The last two are left as open exercises for you to try on your own. This exercise serves as an overview and just barely scratches the surface of text analytics with WebTooling.

Task 2. Running R commands and operations.

In Tasks 2 and 3, you will use R and Big R commands via the R Console. An additional component that you can install separately as an IDE for development of R / Big R projects, is RStudio. Open source R have been around, and is great for statistical analysis on small(er) datasets. With big data, Big R is needed to parallelize the operations across the Hadoop clusters in order to improve the accuracy and the model that is generated from the analysis. The goal of this section is to introduce you to Big R through the R Console.

1. Click the **Ambari** tab in Firefox.
2. Make sure that **R** and **Big R** has been started in the **Ambari** console; if not, start it now.
3. To open a new terminal, right-click the desktop, and then click **Open in Terminal**.
4. To start the R Console, type `R`.

   Once the console starts up, you should immediately see the R version, as well as the welcome message. You are now in the interactive mode.

   R comes with a base set of functionality, but it can be easily extended by installing R packages. To install R packages the `install.packages()` function is used, but you should not require any new packages during this lab.

   Note: The packages could either be made available within the Comprehensive R Archive Network (CRAN) or from a standard compressed file. For example, if you wanted to connect to a relational database then you could install the RJDBC package using the command `install.packages("RJDBC")`.

   A package is a collection or group of R objects. These functions may contain functions, data structures, links to other libraries, and documentation.
5. To list the install packages in R, as the ">" prompt, type `library()`.
6. To exit the *library()* command, type `q`.
7. Notice that the *bigr* is an installed package.

   To know more a bit about the packages, use the *help()* command for details on the contents of the packages, functions, and datasets. This *help()* function provides access to the embedded documentation and is very useful to use while learning about the libraries.
8. To learn more about the bigr package, type:

   ```
   help (package='bigr')
   ```

   The help should indicate the version of Big R, as well other information pertaining to the package.

9. To quit using help, type **`q`**.

   Packages must be loaded into memory before you can use them.

10. To see which packages are currently loaded into memory, type **`search()`**.

    This returns an ordered list of packages that are currently in memory and available for our R scripts. Notice that the base package is the final package in the search path.

11. If you want to use the bigr package you would first need to load it into memory using the require() or library() function; to do this, type **`require(bigr)`**.

    You can use the help() command to get help on functions as well.

12. To fetch the help for the bigr.frame function, type **`help(bigr.frame)`**.
13. Quit the help function.
14. Another method of obtaining help is by appending a question mark (?) in front of the object. To get help for R's data.frame, type **`?data.frame`**.
15. Type **`q`** to quit.

    If you don't remember the exact name of the function you can use the double question mark (??) to search across all of the packages for a match

16. To look for possible histogram functions, type **`??histogram`**.
17. Type **`q`** to quit.

    Listing R objects (data structures / functions).

18. To list the current objects within your environment, type **`ls()`**.

    You will not see many objects if you run the ls() function now, as you have not created anything yet.

    When R is started, the current directory is considered the working area or working directory. Whenever you load or save files from within R, the location is relative to the current R working directory.

19. To verify the working directory, type **`getwd()`**.
20. To set the working directory to /home/biadmin/labfiles/bigr/labs-r, type:

    ```
    setwd('/home/biadmin/labfiles/bigr/labs-r')
    ```

21. To examine the new working directory, type **`dir()`**.

    You should see a list of files with various extensions. You can pass regular expressions into this command to only list the files that have the word "plot" in the name.

22. Type: **`dir(pattern="*plot*")`**.

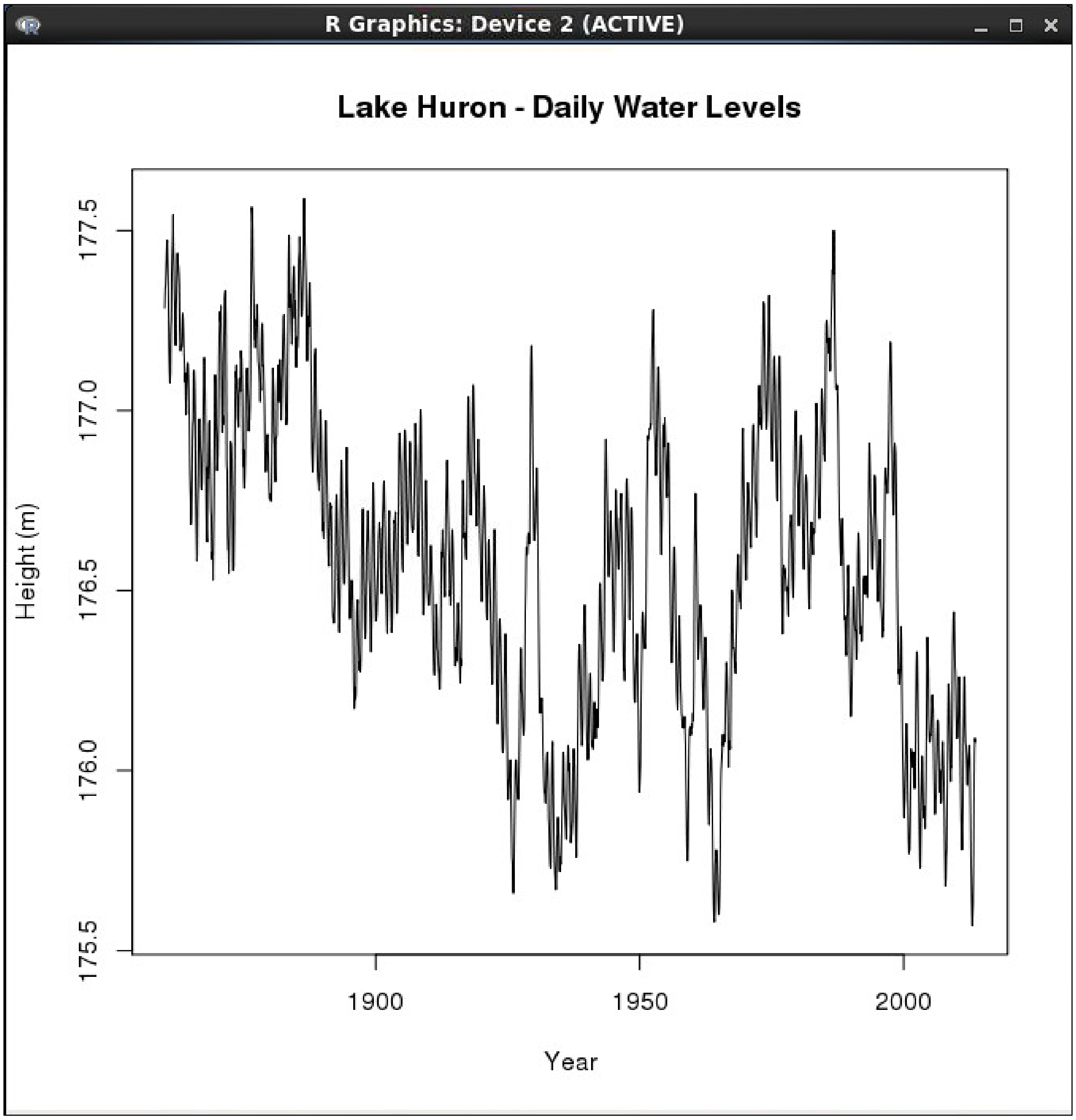
    R programs or scripts are simple text files and they can be executed from inside the R Console. They can also be executed outside of the console.

23. To will run the script "ex1_huron.R" from within the console using the source() function, type:

    `source("ex1_huron.R")`

    The ex1_huron.R script will generate output to the R console and it will create a graph of the water level of Lake Huron over many years.

    The results appear as follows:



24. Close the graphics window and return to the R Console.

    The other scripts are provided just for reference if you wish to try them within this course. It will not be covered in this course.

25. To end an R Console session, the q() or quit() function is used; do this now.

    You will be prompted to save the workspace image [y/n/c].

26. At this time you do not want to save any changes, so type **`n`**.

Task 3. Running Big R commands and operations.

This task will get you up and running with Big R using the interactive R Console. RStudio can also be used if you download and install it yourself; that is beyond the scope of this course. Remember that Big R is designed to work with big data, like those stored on the HDFS.

1. Start the R Console, or resume from the previous section.
2. To set the working directory to /home/biadmin/labfiles/bigr/labs-bigr, type:

    ```
    setwd("/home/biadmin/labfiles/bigr/labs-bigr")
    ```

3. To clear the workspace, type **`rm(list = ls())`**.
4. To load the Big R package into your R session, type **`library(bigr)`**.
5. To connect to BigInsights, type the following:

    ```
    bigr.connect("ibmclass.localdomain", "bigr", "ibm2blue")
    ```

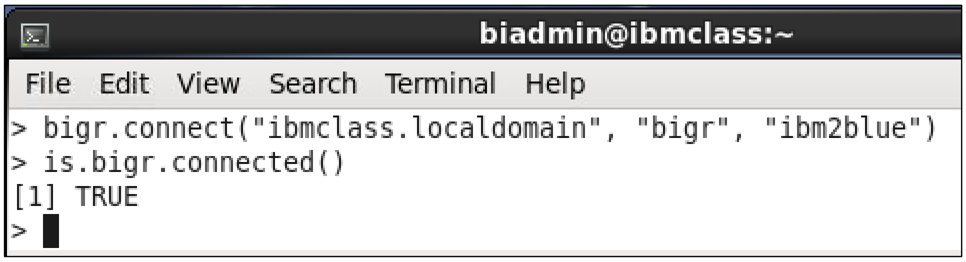
    where you supply the host, the user id, and the password.

    In the lab environment, LDAP has been turned off, so the user id and the password are ignored. Do not do this for the production environment.

    If you are unable to connect, make sure the Big R service is started. Try a restart if it still does not work.

6. To verify that the connection was successful, type **`is.bigr.connected()`**.

    The results appear as follows:



```
> bigr.connect("ibmclass.localdomain", "bigr", "ibm2blue")
> is.bigr.connected()
[1] TRUE
>
```

Once connected, you will be able to browse the HDFS file system and examine datasets that have already been loaded onto the cluster.

7. To list the files under root, type `bigr.listfs()`:
8. To list the files under /user/biadmin, type `bigr.listfs("/user/biadmin/").`

   You will see some of the files that you uploaded to the hdfs from previous exercises that you have completed.
9. To upload the *airline_lab.csv* file onto the hfds under */user/biadmin/*, open a new terminal and type:

   ```
   hdfs dfs -put /home/biadmin/labfiles/bigr/labs-bigr/airline_lab.csv
   /user/biadmin/
   ```
10. Exit the terminal and return to the R Console.

    You want to connect to a big data set and do some exploration. The airline_lab.csv file is a comma-delimited file (type = "DEL"). You will create a bigr.frame over the dataset. A bigr.frame is an R object that mimics R's own data.frame. However, unlike R, a bigr.frame does not load that data in memory, as that would be impractical. The data stays in HDFS. However, you will still be able to explore this data using the Big R API.
11. Type:

    ```
    air <- bigr.frame(dataSource="DEL", header=T,
    dataPath="/user/biadmin/airline_lab.csv")
    ```
12. This creates an object air that is of bigr.frame; to check out the air object, type `class(air).`
13. Examine the structure of the dataset.

    Note that the output looks very similar to R's data.frames. The dataset has 29 variables (for example, columns). The first few values of each column are also shown.
14. To examine the columns and see what they may possibly represent, type `str(air).`

    Notice that the column types are all "character" (abbreviated as "chr"). Unless specified otherwise, Big R automatically assumes all data to be strings. However, only columns Year (1), Month (2), UniqueCarrier (9), TailNum (11), Origin (17), Dest (18), CancellationCode (23) are strings, while the rest are numbers. You will assign the correct column types.
15. To build a vector that holds the column types for all columns,type:

    ```
    ct <- ifelse(1:29 %in% c(1,2,9,11,17,18,23), "character", "integer")
    ```

    ```
    print (ct)
    ```

16. To assign the column types, type `coltypes(air) <- ct`.

    This data originally comes from US Department of Transportation (http://www.rita.dot.gov), and it provides us information on every US flight over the past couple of decades. The original data has approximately 125+ million rows. For this lab, you are only using a small sample; you will examine the dimensions of the dataset.

17. To get the number of rows (flights) in the data, type `nrow(air).`

    Warning: Some of these commands may take a while to execute.

18. To get the number of columns (attributes) recorded for each flight, type: `ncol(air).`

    You will summarize some key columns to gain further understanding of this data. You will see that the years range from 1987-2008.

19. Type:

    ```
    summary(air[, c("Year", "Month", "UniqueCarrier")])
    ```

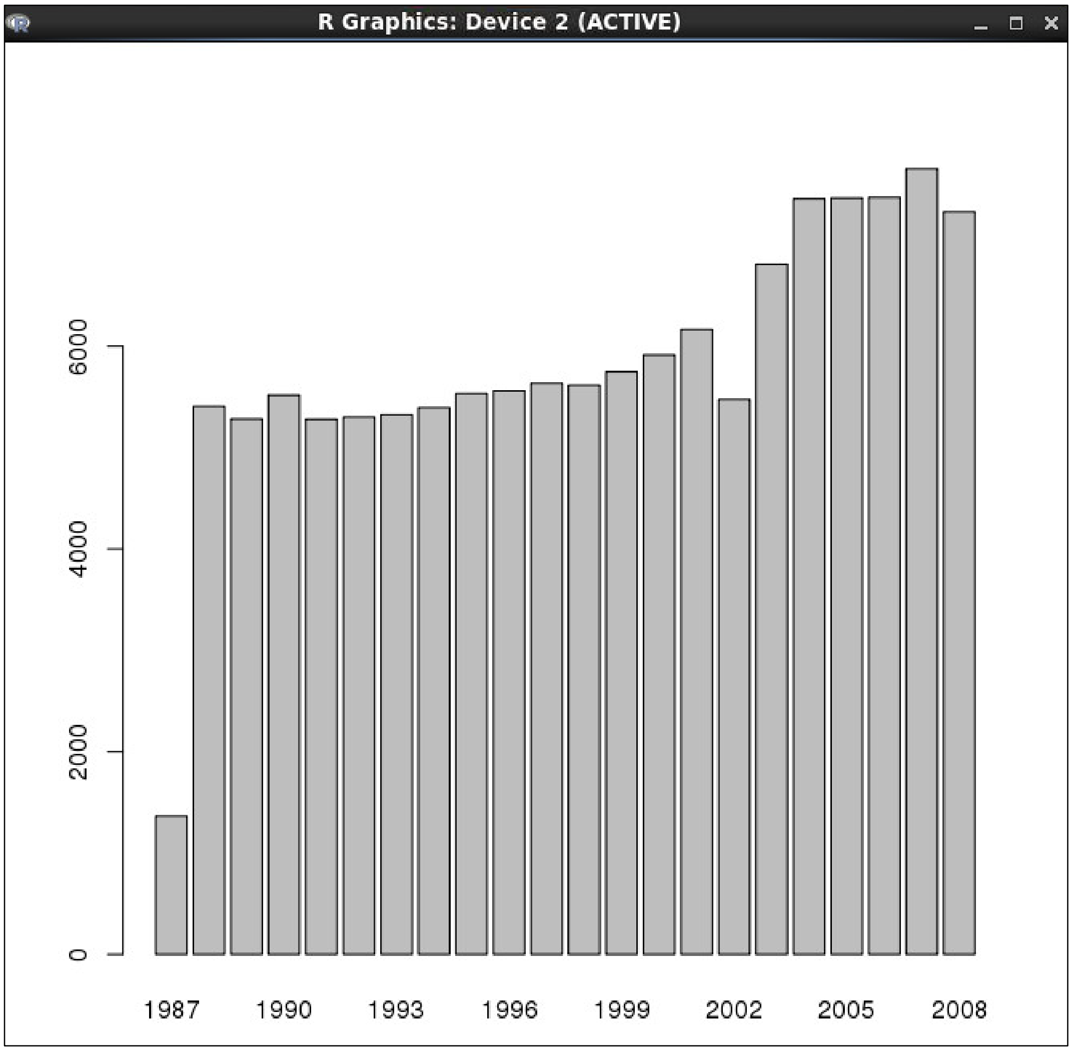
    Summarizing columns (vectors) one by one will give you additional information. In some cases, you will also visualize the information. The following statement returns the distribution of flights by year. Again, you have 22 years worth of data. What you will see is a vector that has the "year" for the name, and the flight count for the values.

20. Type:

    ```
    summary(air$Year)
    ```

21. To visualize the data using some of R's visualization capabilities to see the same data distribution graphically, type **`barplot(summary(air$Year)).`**

    The results appear as follows:



22. Review and then close the graph.

    Similarly, you can also examine the distribution of flights by airline (UniqueCarrier). We have 29 airlines in this dataset, including United (UA), Delta (DL), and many others.

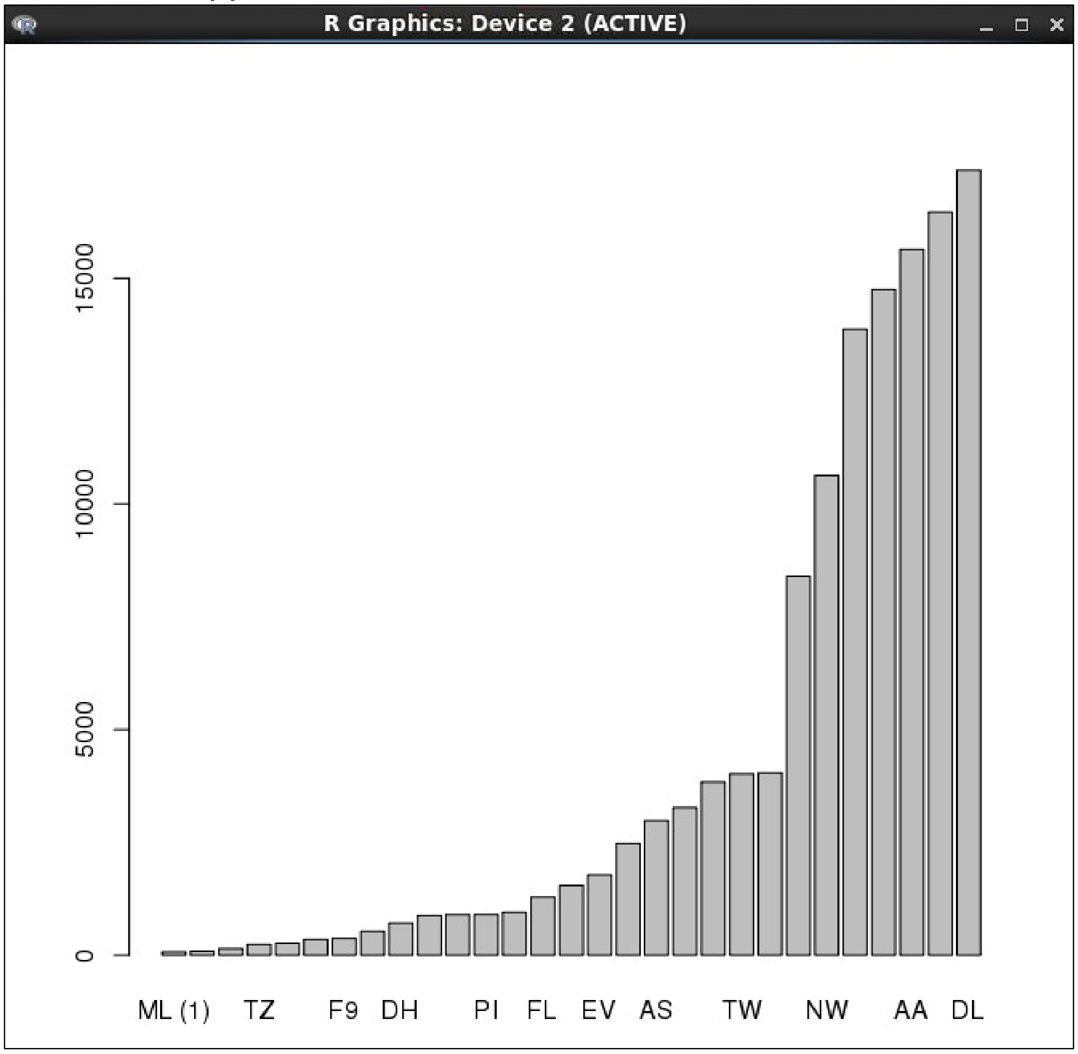
23. Type:

```
summary(air$UniqueCarrier)
```

24. To visualize the data, type:

    ```
    barplot(sort(summary(air$UniqueCarrier)))
    ```

    The results appear as follows:



25. Close all open windows.

    This concludes this exercise. You should now be able to get started with R and Big R exercises using IBM BigInsights. You also had an overview into the Web Tooling module for Text Analytics. These value-adds and more, are available with the IBM BigInsights Data Scientist module.

Results:
You have created a new Text Analytics Web Tooling project and loaded documents to scan for certain keywords. You started up the R console and ran basic commands on it. You also loaded the BigR libraries and ran basic BigR operations.