Visual Analysis of Historical Lessons Learned during Exercises for the United States Air Force Europe (USAFE)

Samantha O'Rourke
skorourke@unomaha.edu

Follow this and additional works at: https://digitalcommons.unomaha.edu/university_honors_program

Part of the Analysis Commons, Data Science Commons, and the Programming Languages and Compilers Commons

Recommended Citation

This Dissertation/Thesis is brought to you for free and open access by the University Honors Program at DigitalCommons@UNO. It has been accepted for inclusion in Theses/Capstones/Creative Projects by an authorized administrator of DigitalCommons@UNO. For more information, please contact unodigitalcommons@unomaha.edu.
Visual Analysis of Historical Lessons Learned during Exercises for the United States Air Force Europe (USAFE)

Samantha O’Rourke

Department of Mathematics, University of Nebraska at Omaha

University Honors Program Thesis/Capstone/Creative Project

Project Advisor: Fabio Vitor

May 2021
Abstract

Within the United States Air Force, there are repeated patterns of differences observed during exercises. After an exercise is completed, forms are filled out detailing observations, successes, and recommendations seen throughout the exercise. At the most, no two reports are identical and must be analyzed by personnel and then categorized based on common themes observed. Developing a computer application will greatly reduce the time and resources used to analyze each After Action Report. This application can visually represent these observations and optimize the effectiveness of these exercises. The visualization is done through graphs displaying the frequency of observations and recommendations. Thus, the Air Force will be able to see what the common observations and recommendations are visually. This will impact the Air Force and all of the Department of Defense by saving them both time and money if this application is implemented during future exercises.
## Table of Contents

1. Introduction ................................................. 4
2. Project Description ........................................ 5
3. Development ................................................ 6
   3.1. Data Mining .......................................... 6
   3.2. Data Visualization ................................. 18
4. Results and Analysis ..................................... 19
5. Conclusion .................................................. 22
6. Future Work ................................................. 23
7. Appendix ..................................................... 24
1. **Introduction**

Within the United States Air Force Europe (USAFE), there persists an issue with After Action Reports being analyzed one at a time manually. After Action Reports are compiled using data provided by the United States Air Force. The information pertains to observations and recommendations that are seen following the completion of exercises. Files in PDF and Microsoft Word format included After Action Reports from exercises conducted by USAFE over the previous years. The unclassified data is provided by USAFE. This project’s design creates a filtering computer application that cycles through the After Action Report and finds observations and recommendations. Therefore, this project began as an application to determine the most common observations throughout all exercises performed by the Air Force.

The data has been mined to find specific keywords and remove any information that is not needed. After filtering the data, it was exported to a Microsoft Excel format to view the findings more efficiently. The data was then filtered again to find the most common issues that are observed during exercises. Finally, the data was converted to a graph. Thus, it is easy to identify the top problems encountered during exercises visually. The process of finding and extracting patterns from large datasets and converting them into a comprehensible form is done so that USAFE can use them for future exercises. The graphical representation of the data was extracted from data mining using visual guides like graphs, charts, and maps. It helped visualize analyzing massive amounts of information.

Following the introduction, the report is organized as follows: Section 2 presents the development of this project, including some background information, the methodology used to create the model, data manipulation, and model implementation; Section 3 presents the results
obtained by using the dataset along with a discussion of the results; Section 4 concludes the report and provides topics for future work.

2. **Project Description**

Every year, the United States Air Force conducts exercises of their forces on each of their bases and Jointly with other organizations. Exercises can be anything from a meeting to a computer simulated real-world event to a live-flight training event where they practice how they would execute their mission. Every unit of the Air Force runs exercises that are specific to their specific expertise. Throughout these exercises, observations are recorded about events that happened during the exercise. Each of these observations is recorded by a participant in the exercise and recommendations are made in order for future exercises to run smoother and not have to deal with the same issues. All the observations and recommendations from one exercise are compiled into one document called an After Action Report. There exists a format that members are supposed to follow to compile their After Action Report; however, personnel writing the reports rarely follow the format because either the employee is not aware of the formatting or does not possess the time to format their documents correctly. Most documents in the database are vastly different, thus, employees have to spend time reading through the reports and manually record all the observations and recommendations while ignoring all the other statements in the report. This is a problem because it wastes time and resources that could be spent doing other projects. Therefore, an application is required and has been implemented via this study to solve this issue. The application is explained as follows.
3. Development

The software was developed using R®, a language and an environment for statistical computing and graphics. It is a GNU project which was developed at Bell Laboratories. Preliminary algorithms were run on a Quad-Core Intel Core i5 at 2 GHz with 16 GB of RAM.

3.1. Data Mining

When using this application, the user is prompted to input keywords multiple times to mine within the application itself through the document for the specified keywords. The application goes through the file to find the positions of each keyword. Using the position of the keywords, lists are created to save paragraphs of text to be then exported to a Microsoft Excel Spreadsheet. The spreadsheet helps in data visualization later. Data mining is separated into the following categories: data acquisition; preparing documents, data cleaning, and prompting user input; altering the text to be universal; finding all positions; creating a list of observations and recommendations; and exporting data frames to Microsoft Excel and data testing.

3.1.1. Data Acquisition

To begin the coding process, the user is first prompted to input how many documents they want to run through. The application has a for loop that runs however many times the user inputs. After the for loop, the document’s title that they wish to have analyzed by the program must be entered. The input must include the file extension (.pdf or .docx). This input is then saved into the variable var, as seen in Figure 1. The last four characters in the string of var are saved into var_end.
A true/false statement is created to check if the document name entered into var was a PDF or a Microsoft Word. Hence, making sure that the correct function is called to read in the file’s format. For both statements, once the file is read and saved, all words are set to lowercase and unlist text from a list into a vector. All spaces are removed, and each word is designated as a character separate from other words. All this is saved as all_stat_lines, as viewed in Figure 2.

3.1.2. Preparing Documents, Data Cleaning, and Prompting User Input

Once the document has been prepared to be examined, the user is again prompted to provide inputs. First, they must state how many keywords from the document they wish the
program to look for; then they input those keywords, including punctuation; finally, they input
what word they wish the application to stop at.

```r
num_keywords <- readline(prompt = "Enter number of keywords: ")
num_keywords <- as.integer(num_keywords)

keywords <- list()
for (i in 1:num_keywords)
{
    key <- readline(prompt = "Enter keyword (all lowercase and including punctuation): ")
    keywords <- c(keywords, key)
}
end_keyword <- readline(prompt = "Enter ending keyword (such as conclusion or final): ")
```

Figure 3: Section of the code that prompts the user to input the number of keywords and also the keywords themselves and stores the inputs.

The first prompt is done by reading in an integer, with a line after the prompt to ensure the input is an integer and not a string or character. Then an empty list is created to store the values of the keywords to be imputed. A `for` loop is made to ask the user the number of times they imputed for the previous prompt. For example, if the user stated they wanted the application to look for four keywords, it would prompt them four times to put in those keywords individually. Each keyword is stored at the end of the previously created empty list. Finally, the user is prompted once more to enter the words they wish to end the program. Since most documents do not end in the same manner, the user could input the end word of ‘conclusion’ or ‘signed’ or the name of the person signing the document at the end. All of the previous statements can be viewed in Figure 3.

### 3.1.3. Altering Text to be Universal

After having the user input which keywords they want the program to scan for, the next step is to alter the text to be in a universal format. Two `for` loops with the variables of `i` and `j` were created to run through the entirety of the document by position. An `if` statement was designed to
make sure that there would be no out-of-bounds errors when combing through the document. Next, a series of if statements were created to check if the character at position $i,j$ of `all_stat_lines` equaled the keyword that the user inputted first. If this is true, then the character at that position would be changed to $(o)$. This represents observations that were found in the document. Else if the character at position $i,j$ were the same as the second keyword, then the word at the position of $i$ and $j$ would be changed to $(r)$ to represent recommendations. Finally, the program checks if the user inputted more than two keywords; if they did, then from the third keyword to however many keywords the user inputted are converted to $(s)$. This was done because the document that this program was initially built for had $(o)$, $(r)$, and $(s)$ to stand for observations, recommendations, and successes, respectively. Thus, when the program was altered to work for any document, these initial keywords were kept in later code to not run into issues with accounting for however many keywords the user inputted. All of this is seen in Figure 4.

3.1.4. Finding all Positions

Since the text has been formatted for universal use, it is time to find the positions of each observation and recommendation so that it could eventually be exported to Microsoft Excel and visualized. To achieve this, variables are to be created to be used in loops.

As can be observed in Figure 5, four empty lists were created to hold the positions of the row and columns of observations and recommendations, respectively. Since the document was held in a long list, there would be the row number and then the column number of where the keyword was held. For example, the list of `obs_list_row` would contain (30, 44, 59, 62, 69), and usually `obs_list_column` would be at the same position (2, 2, 2, 2, 2). Two integers were also created to increase when an observation or recommendation was found. They were used as the element number for the list. So, if an observation were found, then the position would be placed at
obs number of the list, and obs would be increased by one, so the next time, it would be in the next element of the list. The way these variables were implemented can be seen in Figure 6.

```r
for (i in 1:length(all_stat_lines))
{
  for (j in 1:length(all_stat_lines[[i]]))
  {
    if(length(all_stat_lines[[i]]) > 0)
    {
      if(all_stat_lines[[i]][j] == keywords[1])
      {
        all_stat_lines[[i]][j] <- "(o)"
      }
    }
    else if(all_stat_lines[[i]][j] == keywords[2])
    {
      all_stat_lines[[i]][j] <- "(r)"
    }
    if (length(keywords) > 2)
    {
      for(x in 3:length(keywords))
      {
        if(all_stat_lines[[i]][j] == keywords[x])
        {
          all_stat_lines[[i]][j] <- "(s)"
        }
      }
    }
  }
}
```

Figure 4: Section of the code that converts the first two keywords to (o) and (r), respectively, and the remainder keywords into (s).

```r
obst <- 1
obs_list_row <- list()
obst_list_column <- list()
rec <- 1
rec_list_row <- list()
rec_list_column <- list()
```

Figure 5: Creations of variables to be used later.
for (i in 1:length(all_stat_lines))
{
    for (j in 1:length(all_stat_lines[[i]]))
    {
        if(length(all_stat_lines[[i]]) > 0)
        {
            if (all_stat_lines[[i]][j] == "(o)")
            {
                position_i_obs = i
                position_j_obs = j

                obs_list_row[[obs]] <- position_i_obs
                obs_list_column[[obs]] <- position_j_obs
                obs <- obs + 1
            }
            if (all_stat_lines[[i]][j] == "(r)")
            {
                position_i_rec = i
                position_j_rec = j

                rec_list_row[[rec]] <- position_i_rec
                rec_list_column[[rec]] <- position_j_rec
                rec <- rec + 1
            }
        }
    }
}

Figure 6: Double for loops that iterate over the document to find the positions of the keywords that were inputted.

Two for loops were created to iterate over the length of all_stat_lines with i and j increasing every time the loop starts over. Next, an if statement is designed to ensure that the length was greater than zero, so no out-of-bounds errors were thrown. Another two if statements are made to find if the character or word of all_stat_lines at the position of row i and column j, then the variables position_i_obs and position_j_obs were set equal to i and j, respectively. Then the position number is appended to the empty lists of obs_list_row and obs_list_column at element number one since obs equals one initially. Finally, obs is incremented by one so that the next time the if statement is true, i and j will be saved at element two in obs_list_row and obs_list_column.
The next if statement is a repeat of the previous if statement, but now find the position for recommendations.

3.1.5. Creating a List of Observations and Recommendations

After finding all of the positions where the keywords were located, it was time to save all the words in the sentence or sentences following the keyword. An example of the sentences can be seen in Figure 17 through Figure 19.

```r
k <- 1
flag1 <- 0
flag2 <- 1
s <- 1
observations <- list()
count <- 1
obs_position <- list()
count2 <- 0
```

Figure 7: Creations of variables to be used later.

Before the program can begin with the lists, eight variables must be created, as seen in Figure 7. Six integers were created to be used as counters either through the length of the document or in empty lists. Two empty lists, `observations`, and `obs_position`, were created to store all the characters of observations and store the position number of the last character in the observations paragraphs.
while (s <= length(all_stat_lines))
{
    for (r in 1:length(all_stat_lines[[s]]))
    {
        if(length(all_stat_lines[[s]]) > 0)
        {
            if ((s == obs_list_row[[k]]) | (flag1 == 1))
            {
                if ((r == obs_list_column[[k]]) & (flag2 == 1))
                {
                    flag1 <- 1
                    flag2 <- 0
                }
                else if (flag2 == 0)
                {
                    if ((all_stat_lines[[s]][r] == "(o)") | (all_stat_lines[[s]][r] == "(r)") | (all_stat_lines[[s]][r] == "(s)"))
                    {
                        obs_position <- c(obs_position, count2)
                        flag2 <- 1
                        flag1 <- 0
                        s <- s - 1
                        if (k < length(obs_list_column))
                        {
                            k <- k + 1
                        }
                    }
                    else
                    {
                        observations[count] <- all_stat_lines[[s]][r]
                        count2 <- count2 + 1
                    }
                }
            }
            count <- count + 1
        }
        s <- s + 1
    }
}

Figure 8: While loop to iterate over the document to save the observations into an empty list.

Each of the previously stated variables can be seen implemented in the `while` loop in Figure 8. While $s$, which is initially one, is less than or equal to the length of `all_stat_lines`, do the following. **For** the variable $r$ in the length of `all_stat_lines`, **if** the length is greater than zero, **if** the
number in \textit{obs\_list\_row} at element $k$ is equal to the number of $s$ or \textit{flag\_1} is equal to one, and \textbf{if} the number in \textit{obs\_list\_column} at element $k$ is equal to the number of $r$ and \textit{flag\_2} is equal to one, then \textit{flag\_1} is set equal to one and \textit{flag\_2} is set to zero. This accounts for the positions in which the keyword is at. Thus the program starts the list of words for that observation. By setting \textit{flag\_2} equal to zero, the program enters an \textbf{else} statement where it checks if \textit{all\_stat\_lines} is equal to (o), (r), or (s). If this returns true, then \textit{obs\_position} is updated with the ending position of that observation.

Next, \textit{flag\_1} is set to zero, and \textit{flag\_2} is set to one. The variable $S$ is decreased by one so that the program will go back in the text to make sure that if two observations are right after each other, the second observation is not overlooked. As long as $k$ is less than the length of \textit{obs\_list\_column}, $k$ is increased by one. Now that the beginning and end of the observation have been accounted for, everything in between is added to the list of \textit{observations} at an element of \textit{count} (initially one). \textit{Count\_2}, \textit{count}, and $s$ are increased by one to find the following observation.

The same is done for recommendations; however, a few changes are made. The names of variables are altered, so there is no variable not starting at zero or one. Also, the statement where it checks if \textit{all\_stat\_lines} is equal to (o), (r), or (s) is changed to account for \textit{end\_keyword} as well. Otherwise, the last recommendation is not included in the Excel spreadsheet. This can be observed in Figure 9.
1 <- 1
pole1 <- 0
pole2 <- 1
i <- 1
recommendations <- list()
x <- 1
rec_position <- list()
x2 <- 0

while (i <= length(all_stat_lines))
{
    for (j in 1:length(all_stat_lines[[i]]))
    {
        if(length(all_stat_lines[[i]]) > 0)
        {
            if ((i == rec_list_row[[l]]) | (pole1 == 1))
            {
                if ((j == rec_list_column[[l]]) & (pole2 == 1))
                {
                    pole1 <- 1
                    pole2 <- 0
                }
                else if (pole2 == 0)
                {
                    if ((all_stat_lines[[i]][j] == "(o)"") | (all_stat_lines[[i]][j] == "(r)"") | (all_stat_lines[[i]][j] == "(s)"") | (all_stat_lines[[i]][j] == end_keyword))
                    {
                        rec_position <- c(rec_position, x2)
                        pole2 <- 1
                        pole1 <- 0
                        i <- i - 1
                        if (l < length(rec_list_column))
                        {
                            l <- l + 1
                        }
                    }
                    else
                    {
                        recommendations[x] <- all_stat_lines[[i]][j]
                        x2 <- x2 + 1
                    }
                }
            }
            else
            {
                recommendations[x] <- all_stat_lines[[i]][j]
                x2 <- x2 + 1
            }
        }
        x <- x + 1
    }
    i <- i + 1
}

Figure 9: While loop to iterate over the document to save the recommendations into an empty list.
3.1.6. Exporting Dataframe to Microsoft Excel and Data Testing

The final step of the data mining is exporting the list of observations and recommendations to a Microsoft Excel spreadsheet. First, the list *observations* is row bound to another variable. That variable is converted into a data frame. By converting it to a data frame, it is easier to view the findings, and R®️ allows data frames to be exported to Microsoft Excel and not lists. The data frame is transposed so that each character is in one row but a different column. Using *obs_position* from the previous section, a **for** loop is created to combine individual observations. Instead of each word being in a different column, they are now in one column. A matrix is created, and the column names are labeled as Observations. The same is done for recommendations. This can be viewed in Figure 11.

```r
list_observations <- rbind(observations)
df_observation <- do.call("rbind", lapply(list_observations, as.data.frame))
df2_observation <- as.data.frame(t(df_observation))
df_all_obs_final <- list()
initial_position <- 1
for (i in 1:length(obs_position))
{
  df_observation_final <- unite(df2_observation, "Observation",
    initial_position:obs_position[[i]], sep = " ")
  df_all_obs_final[[i]] <- subset(df_observation_final, select=Observation)
  initial_position <- obs_position[[i]] + 1
}

data1 <- data.frame(matrix(unlist(df_all_obs_final),
nrow=length(df_all_obs_final)))
colnames(data1) <- "Observations"
```

Figure 10: Section of code that converted the list of observations into a dataframe.
list_recommendations <- rbind(recommendations)

df_recommendations <- do.call("rbind", lapply(list_recommendations, as.data.frame))
df2_recommendations <- as.data.frame(t(df_recommendations))

df_all_rec_final <- list()
initial_position <- 1
for (i in 1:length(rec_position))
{
  df_recommendations_final <- unite(df2_recommendations, "Recommendation",
                                 initial_position:rec_position[[i]], sep = " ",
  df_all_rec_final[i] <- subset(df_recommendations_final,
                                 select="Recommendation")
  initial_position <- rec_position[[i]] + 1
}

data2 <- data.frame(matrix(unlist(df_all_rec_final), nrow=length(df_all_rec_final)))
colnames(data2) <- "Recommendations"

Figure 11: Section of code that converted the list of recommendations into a dataframe.

Subsequently, since both data frames of observations and recommendations have been compiled, they are combined into a list with two elements. The first being observations and the second being recommendations. Then the user is prompted one final time to enter what name they would like the Microsoft Excel spreadsheet to be called. They must put .xlsx at the end of the title. Then all is exported to Microsoft Excel, and the code is shown in Figure 12. The final result can be seen in the Results section.

list_of_datasets <- list("Observations" = data1, "Recommendations" = data2)

Excel <- readline(prompt = "Enter Excel Spreadsheet Name (followed by .xlsx): ")

write.xlsx(list_of_datasets, Excel)

Figure 12: Exporting the data frame to a Microsoft Excel Spreadsheet.
3.2. Data Visualization

Once all data mining is completed, the visualization of repeat observations and recommendations is shown. There is a small amount of visualization now, but there will be more in the future. Since the given documents were unclassified, much of the visualization is subject to only unclassified information. Classified documents might vary much to the format of the PDFs and Microsoft Word documents are given at present. The visualization forthwith is of the frequency of words found in observations and recommendations for one document. This document was chosen because it had the best format, and the solution could be seen. The final graph can be viewed in the next section and the coding in Figure 13.

```r
# Code for creating the visualization

# Load necessary libraries
library(tm)
library(e1071)
library(ggplot2)
library(gridExtra)
library(scales)

# Read data from observations
data1 <- read.table("observations.txt", header = TRUE, sep = "\t")

# Create corpus
corpus = Corpus(VectorSource(data1))
corpus = tm_map(corpus, removePunctuation)
corpus = tm_map(corpus, removeWords, c("cloth", stopwords("english")))
corpus = tm_map(corpus, stemDocument)

doc_mat <- TermDocumentMatrix(corpus)
m <- as.matrix(doc_mat)
v <- sort(rowSums(m), decreasing = TRUE)
d_Rcran <- data.frame(word = names(v), freq = v)
head(d_Rcran, 5)

# Create bar plot
barplot(d_Rcran[1:25,]$freq,
    las = 2,
    names.arg = d_Rcran[1:25,]$word,
    col = brewer.pal(n = 10, name = "GnBu"),
    main = "Most Frequent Words in Observations",
    ylab = "Word Count")

# Figure 13: Code to create a visualization of frequency of observations.
```

The vector of the data from observations is saved into a variable `corpus`. Then the punctuation and common words are removed from `corpus`. Therefore, the frequency will skip over
words such as “the,” “a,” and “and” so that the issues that were observed are graphed. Then a bar graph is created. The same is done for recommendations, but the graph is a different color. The coding of this is seen in Figure 14.

```r
corpus = Corpus(VectorSource(data2))
corpus = tm_map(corpus, removePunctuation)
corpus = tm_map(corpus, removeWords, stopwords("english"))
corpus = tm_map(corpus, stemDocument)

doc_mat <- TermDocumentMatrix(corpus)

m <- as.matrix(doc_mat)

v <- sort(rowSums(m), decreasing = TRUE)

d_Rcran <- data.frame(word = names(v), freq = v)

head(d_Rcran, 5)

barplot(d_Rcran[1:25,]$freq,
   las = 2,
   names.arg = d_Rcran[1:25,]$word,
   #col="#69b3a2",
   col = brewer.pal(n = 10, name = "PuBu"),
   main = "Most Frequent Words in Recommendations",
   ylab = "Word Count")
```

Figure 14: Code to create a visualization of frequency of recommendations.

After the application was complete, the results were analyzed to see if an improvement could be made. The results from the data mining and data visualization were viewed and compared to the original After Action Report. Continuing onto the Results and Analysis section, one can see what was found after the application is run.

4. Results and Analysis

The results from exporting the data frame to Microsoft Excel are shown in Figure 15. However, observations and recommendations are on separate sheets. These results are from the
PDF that can be seen in Figures 17 through 19. In the PDFs, observations were started with (O), recommendations started with (R), and another keyword was (S), which stood for successes. There were five observations and four recommendations found in the After Action Report. The issue that will be corrected in the future is that the following number for the next keyword is included at the end of the previous keyword. For now, they are included so that the program will work universally. If it were to remove the last character, it might remove important characters from other documents.

The output from the data visualization can be seen in Figure 16.

<table>
<thead>
<tr>
<th>Observations</th>
<th>Recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>rt19 provided excellent training for the 94th aw and the other participating nations. Participants planned and executed combined air operations (como) as part of an integrated war scenario with four partner nations. rt19 presented a number of unique challenges for air and ground participants. aircrews, mission planners, and intel integrated with coalition partners to develop, plan, and execute realistic scenario mission objectives. real world ops and scheduling conflicts based on the time of year limited the amount of countries who were able to participate. prolonged dry weather also limited any cas or fire training for fighter aircraft. 2.</td>
<td>when working on a limited planning timeline, it is essential that ops and intel be on the same page in regards to the flying schedule. ops and intel where in separate areas during the planning conference. neither had a common framework to operate with during the coordination process with host nation entities. establishing early expectations for utilization before obligating assets is necessary to avoid overcommitting. 5.</td>
</tr>
<tr>
<td>due to the timing of rt19 this year, the prt only hosted one planning conference. it was very big picture with minimal details finalized at the conclusion of the conference. much of the communication and exercise planning was via email. the added language barrier combined with a truncated planning cycle left various mission details unconfirmed. 4.</td>
<td>in future editions of rt19 ensure the jia includes provision not just for lavatory equipment but who will be responsible for its operation. if us operation is required, ensure qualified personnel are part of package. 9.</td>
</tr>
<tr>
<td>based on rt18 recommendations, the 94aw did not participate in the rt19 meal plan. rt19 participants brought their own lunches and was not an issue. all participants had restaurants and grocery stores within walking distance of hotel. 7.</td>
<td>the customs process has been an issue now in both rt18 and rt19. the common theme seems to be a lack of embassy involvement early in the process to avoid delays in customs processing. the dao office and gso in lisbon were vital to resolving the customs vat tax issues. 2 11.</td>
</tr>
<tr>
<td>c-130 lavatory servicing cart was available in accordance with the joint implementation arrangement (jia). however, no qualified us personnel were present to operate the equipment. host nation was able and generous enough to assist but was not obligated to help. 8.</td>
<td>while intel support was more than adequate, two intelligence personnel would be optimal to conduct cas and unit support training. this will provide intel members the opportunity to participate in the entire daily intel mission planning cycle to include ato breakout, map plotting, create tactical picture from intel updates, provide situation briefs, crew brief/debrief and miscrps. 3.</td>
</tr>
<tr>
<td>customs in lisbon made it difficult to release a part required to fix a malfunctioning aircraft system. the 94th air and prt logistics made numerous phone calls and filled out multiple forms to get the part released and shipped to ab13 via fedex but where unsuccessful. eventually hrs personnel had to drive to lisbon and with the help of a contact in the us embassy was able to get the part released. 10.</td>
<td></td>
</tr>
</tbody>
</table>

Figure 15: Microsoft Excel Spreadsheet of observations and recommendations found in an After Action Report.
From observations, participation, plan, and part were the most frequent words found throughout all observations. The frequency skipped over words such as “the,” “a,” and “and” so that the issues that were observed are not taking into account common phrases that are of no use. By viewing Figure 16, most issues found in the After Action Report dealt with participation and planning. Since the exercise involved the R19 group, one can ignore the frequency of r19 and move onto mission. The main three issues that R19 had were participation, planning, and missions.

In recommendations, the most common words were process, custom, and operation. Therefore, the main recommendations for future exercises similar to this one deal with processing, customs, and operations. To further understand the main observations and recommendations, the user will need to consult the Microsoft Excel Spreadsheet and search these keywords.
5. Conclusion

An issue that persists in USAFE is the non-uniform After Action Reports that are produced following the multiple exercises hosted in the European Theater. After Action Reports are compiled to include observations and recommendations that are seen during exercises. Their purpose is to inform the USAFE leadership of things that need their attention and provide recommendations on how to resolve the issue that was observed. This project was designed to filter through After Action Reports and to output the observations and recommendations found throughout each one. To do this, a computer application was created in R® code, a powerful programming language for data mining and data visualization, in order to determine what the most common observations were seen throughout all exercises performed by the Air Force. The data was mined to find specific keywords and remove any information that was not needed. After the data was filtered, it was exported to an Excel format to view the findings more efficiently. The graphical representation of the data was extracted from the data mining using visual guides like

Figure 16: Visualization of frequency of observations found in an After Action Report.
graphs, charts, and maps. It helps to visualize and analyze massive amounts of information. The results of this application were tested and were presented to the USAFE sponsor and the product met all the needs of the Command for the unclassified version.

6. Future Work

The results of this project are being presented to USAFE leadership on May 20, 2021. Since this project was created only around unclassified documents, the program will be expanded upon by USAFE personnel to be implemented with classified documents. A position within USAFE has already been created for a cadet from the US Air Force Academy to continue with the application this summer. Therefore, this application will be given to the cadet so that they may expand it for use with classified documents. An alteration to the application can be made to allow for more than one document to be read by it at a time. Since more than one document will be read, a correlation can be observed to compare the common observations and recommendations across exercises. Further expansion of the visualization capabilities will show how the observations and recommendations found can be correlated with other observations and recommendations from other After Action Reports. The visualizations will be in a bar chart form to be easily visualized by USAFE, or can be expanded into other types of visualization products.
Appendix

Figure 17: Photo of After Action Report, page 1.
Intel integrated with coalition partners to develop, plan, and execute realistic scenario mission objectives. Real world ops and scheduling conflicts based on the time of year limited the amount of countries who were able to participate. Prolonged dry weather also limited any CAS or live fire training for fighter aircraft.

2. (S) Facilities and equipment provided by the Portuguese Air Force (PRT) were suitable for operations. A marked improvement from RT18. There was plenty of space for both maintenance and operations. Operations had two rooms, one for the MPC and another for DETCO support, SARM, and Comm. MX received ample hanger space with a separate room for MXS. AFE and LRS was also set up in the same workspace as MXS.

3. (O) Due to the timing of RT19 this year, the PRT only hosted one planning conference. It was very big picture with minimal details finalized at the conclusion of the conference. Much of the communication and exercise planning was via email. The added language barrier combined with a truncated planning cycle left various mission details unconfirmed.

4. (R) When working on a limited planning timeline, it is essential that Ops and MX be on the same page in regards to the flying schedule. Ops and MX where in separate areas during the planning conference. Neither had a common framework to operate with during the coordination process with Host Nation entities. Establishing early expectations for utilization before obligating assets is necessary to avoid overcommitting.

5. (S) 1 CBRS provided Ops two VHF and two UHF air-to-ground radios to communicate with aircraft as well as four LMRs for MX. Additionally they set up three laptops with full NIPR access and three DSN capable telephones. This set up was invaluable to communications with outside supports agencies and OPCON POCs.

6. (O) Based on RT18 recommendations, the 94AW did not participate in the RT19 meal plan. RT19 participants brought their own lunches and was not an issue. All participants had restaurants and grocery stores within walking distance of hotel.

7. (O) C-130 Lavatory Servicing Cart was available in accordance with the Joint Implementation Arrangement (JIA). However, no qualified US personnel were present to operate the equipment. Host nation was able and generous enough to assist but was not obligated to help.

8. (R) In future editions of RT19 ensure the JIA includes provision not just for lavatory equipment but who will be responsible for its operation. If USAF operation is required, ensure qualified personnel are part of MX package.

9. (O) Customs in Lisbon made it difficult to release a part required to fix a malfunctioning aircraft system. The 94th LRS and PRT Logistics made numerous phone calls and filled out multiple forms to get the part released and shipped to AB11 via Fed Ex but where unsuccessful. Eventually LRS personnel had to drive to Lisbon and with the help of a contact in the US embassy was able to get the part released.

10. (R) The customs process has been an issue now in both RT18 and RT19. The common theme seems to be a lack of embassy involvement early in the process to avoid delays in customs processing. The DAO office and GSO in Lisbon were vital to resolving the customs VAT tax issues.
11. (S) PRT MOCC and Aerospace Ground Equipment support was outstanding. They responded to coordinating transportation of personnel and equipment without hesitation. PRT logistical support was outstanding and essential to MX success.

12. (S) Portuguese intelligence seamlessly and collaboratively integrated 94th OSS Intel into their daily intelligence processes. Exercise provided realistic training for 94th OSS Intel.

13. (R) While Intel support was more than adequate, two intelligence personnel would be optimal to conduct AOC and unit support training. This will provide Intel members the opportunity to participate in the entire daily Intel mission planning cycle to include ATO breakout, map plotting, create tactical picture from Intel updates, provide situation briefs, crew brief/debrief and MISREPs.

3. CONCLUSION: Although the size and scope of RT19 was smaller than previous years, the exercise served as an excellent opportunity for integrated planning and execution of a collaborative NATO mission. A significant amount of the 94th AW participants had little to no experience in mission planning and executing combined air operations with NATO partners. The lack of fighter involvement during the first week, while not ideal, allowed the mission objectives to focus more on tactical air transport. This allowed some of our lead qualified officers the opportunity to serve as mission commanders when normally a CAF pilot would occupy that role. Even with a limited number of participants and an abbreviated exercise planning cycle, RT19 was a value added exercise for the 94th AW. Overall this experience improved our over readiness and enhanced our ability to operate in an integrated environment. We succeeded in supporting RT19 requirements and look forward to new opportunities for supporting future USAFE exercises.
Discussion: G-7 exercise staff worked non-stop but does not have the capacity to plan an exercise of this scale with such a small staff. While the G-7 always got back to me with good answers, response time was slow because there were simply too many issues for three people.

Recommendation: Look at the U.S. Army Europe (USAREUR) exercise staff as a model for proper staff size.

3.33. Observation: (USARAF DCSENG) Initial Planning Conference lacked continuity from Concept Development

Discussion: Planner arrived at the IPC without information about the concept of the exercise. More planning and staff education required before attending the IPC. USARAF staff simply told the participating units to figure out what they wanted to do at the Field Training Exercise (FTX).

Recommendation: USARAF should provide moderator for each planning cell to keep discussion on track and inform participants of the ground rules, intent and training objectives.

3.34. Observation: (USARAF DCSENG) Staff support planning and collaboration

Discussion: Not all staff sections had Training Objective (TO) representation. The low collaboration was symptomatic of a poor exercise design. Some collaboration was made during and between official events but was often reactionary vs. collaborating to ensure exercise execution ran smoothly. Field Training Exercise (FTX) and CPX planners duties and responsibilities should focus on the exercise at planning events but often were distracted by real world logistics.

Recommendation: Create separate JELC events/locations for exercise design/planning and execution logistics. Create a CPX and a FTX planning/coordination team supervised by one overarching exercises director.

3.35. Observation: (USARAF DCSENG) Exercise design was not inclusive of all warfighting functions (WFF)

Discussion: The CPX focused on primarily on G2/J2 Intelligence functions. The Combined Arms Live Fire Exercise (CALFEX) was the focus for the FTX planners. The FTX and CPX were separate and not linked for the most part but some Intelligence and Protection focused events were integrated to the point that it was a detriment to the other warfighting functions. Exercise design was unwilling to shift course to meet additional TOs. The CALFEX created risk to Service Members (SMs), was high cost, and did not create training effects for units in the FTX.

Figure 20: Photo of another After Action Report.