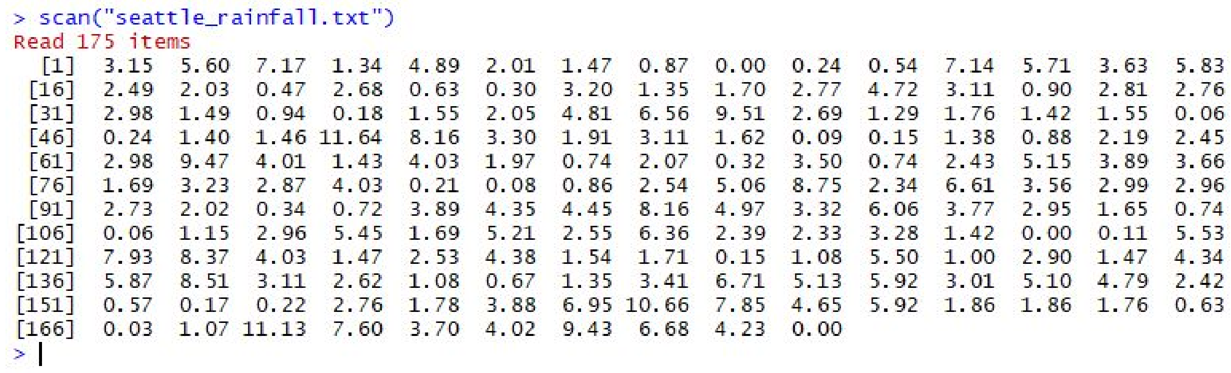
Alexander Rasho

1. Download the linked text file, `seattle_rainfall.txt,` which is the amount of monthly rainfall in a location in Seattle between November 2002 and May 2017[1]. The goal is to convert this to a R `ts` object with the appropriate frequency and then decompose it into trend, seasonal, and random components. Write R code to do the following tasks:

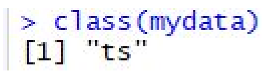
a) Load the file into a R variable. Call `scan('seattle_rainfall.txt')` [show code]

> scan("seattle_rainfall.txt")



```
> scan("seattle_rainfall.txt")
Read 175 items
  [1]  3.15  5.60  7.17  1.34  4.89  2.01  1.47  0.87  0.00  0.24  0.54  7.14  5.71  3.63  5.83
 [16]  2.49  2.03  0.47  2.68  0.63  0.30  3.20  1.35  1.70  2.77  4.72  3.11  0.90  2.81  2.76
 [31]  2.98  1.49  0.94  0.18  1.55  2.05  4.81  6.56  9.51  2.69  1.29  1.76  1.42  1.55  0.06
 [46]  0.24  1.40  1.46 11.64  8.16  3.30  1.91  3.11  1.62  0.09  0.15  1.38  0.88  2.19  2.45
 [61]  2.98  9.47  4.01  1.43  4.03  1.97  0.74  2.07  0.32  3.50  0.74  2.43  5.15  3.89  3.66
 [76]  1.69  3.23  2.87  4.03  0.21  0.08  0.86  2.54  5.06  8.75  2.34  6.61  3.56  2.99  2.96
 [91]  2.73  2.02  0.34  0.72  3.89  4.35  4.45  8.16  4.97  3.32  6.06  3.77  2.95  1.65  0.74
[106]  0.06  1.15  2.96  5.45  1.69  5.21  2.55  6.36  2.39  2.33  3.28  1.42  0.00  0.11  5.53
[121]  7.93  8.37  4.03  1.47  2.53  4.38  1.54  1.71  0.15  1.08  5.50  1.00  2.90  1.47  4.34
[136]  5.87  8.51  3.11  2.62  1.08  0.67  1.35  3.41  6.71  5.13  5.92  3.01  5.10  4.79  2.42
[151]  0.57  0.17  0.22  2.76  1.78  3.88  6.95 10.66  7.85  4.65  5.92  1.86  1.86  1.76  0.63
[166]  0.03  1.07 11.13  7.60  3.70  4.02  9.43  6.68  4.23  0.00
> |
```

> mydata = read.table("seattle_rainfall.txt")
>mydata <- as.ts(mydata)

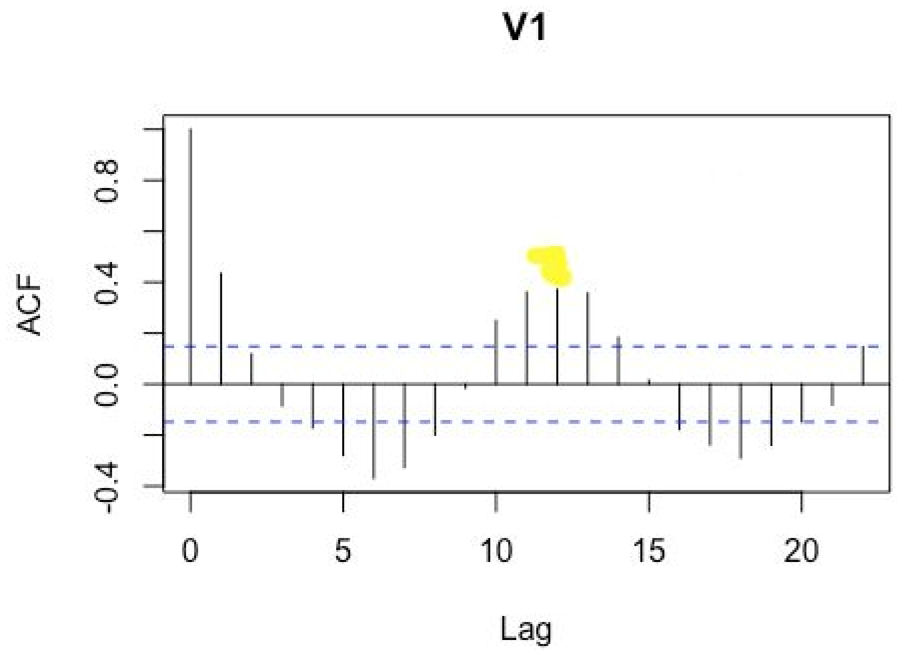


```
> class(mydata)
[1] "ts"
```

b) Find the frequency of the seasonal component by plotting the autocorrelation (plot the output of the `acf` function and pick the peak away from x=0) [show code and plot]

>acf(mydatats)

[1] https://catalog.data.gov/dataset/observed-monthly-rain-gauge-accumulations



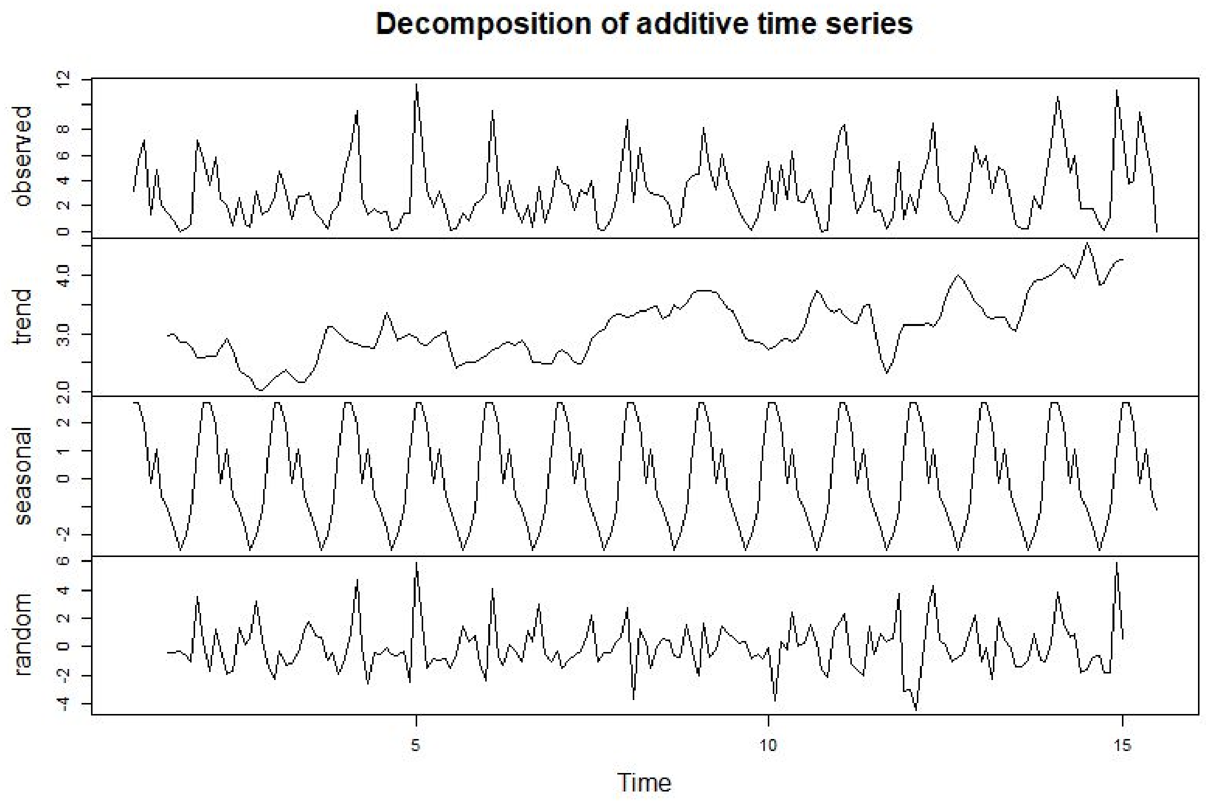
Peak point is at 12 equivalent to 12 months

c) Convert to a `ts` object (call the constructor `ts` with frequency parameter) [show code]

   xts<-ts(mydatats,frequency=12)

d) Decompose the ts dataset (call `decompose` function). Plot the output showing the trend, seasonal, random). [show code and plot]

```
> plot(decompose(xts))
```



2. **Compute the Dynamic Time Warping distance between the two time series:**
   a. **X = [1, 2, 5, 6]**
   b. **Y = [1, 2, 6]**

   Let the cost function be $cost(x_i, y_i) = (x_i - y_i)^2$ **.**

a) **Show the cost matrix. This is partially complete below**.

1	2	6
0	1	25
1	0	16
16	9	1
25	16	0

b) **Show the DTW matrix. This is partially complete below.**

0	1	26
1	0	16

17	9	1
42	25	1

c) **The DTW distance between the two time-series is _____1______.**

3**. Consider the following three short documents:**

Document 1: ***hello world***

Document 2: ***cout hello world***

Document 3: ***cout hello hello***

The following questions are meant to be done by hand though you can also use R. Note that some of the R functions normalize/don't normalize.

a) **Show the Document-Term matrix weighted by Term-frequency (Tf). Do not normalize (i.e., use the word counts).**

	Document 1	**Document 2**	**Document 3:**
cout:	0	1	1
hello:	1	1	2
World:	1	1	0

b) **Show the Document-Term matrix weighted by Term-frequency (Tf). Normalize by the number of words in each document**.

	Document 1	**Document 2**	**Document 3:**
cout:	0	.33	.33
hello:	.5	.33	.66
World:	.5	.33	0

c) What is the inverse document frequency (Idf) of each word?

cout:	log(3/2) = .176
hello:	log(3/3) = 0
World:	log**(**3/2) = .176

d) Show the Document-Term matrix weighted by Tf-Idf for this dataset.

	Document 1	**Document 2**	**Document 3:**
cout:	0	.058	.058
hello:	0	0	0
World:	.088	.058	0

e) Which word has the most "information" based on Tf-idf? Which is the least informative?

World is most informative and **hello** is least informative.

4. Download the linked CSV file that contains tweets relating to airline flights[2]. Each tweet also has an associated "sentiment" - whether the expressed opinion in the tweet is positive, negative, or neutral. The goal is to determine the sentiment of the first tweet by only comparing the first tweet to the remaining tweets. The sentiment of the most similar tweet will then be the predicted sentiment of the first tweet.
The goal is to get familiar with fundamental text processing steps; so do not install and run any sentiment analysis library or function for this question.

Write R code to do the following tasks (please refer to the sample text processing R code posted on Titanium):

a) **Load the given CSV file and convert it to a VCorpus data type.**

```
> data <- read.csv("airline_tweets_sentiment2.csv")
> corp <- VCorpus(VectorSource(data))
```

b) **Print the full text of the first tweet.**

```
> corp[[4]]$content[[1]]
[1] "@united yes. We waited in line for almost an hour to do so. Some passengers just
left not wanting to wait past 1am."
```

c) **Transform all text to remove numbers, remove punctuation, and remove stopwords**

```
> corpt <- tm_map(corp,removeNumbers)
> corpt <- tm_map(corpt,removePunctuation)
> corpt <- tm_map(corpt,removeWords, stopwords("english"))
```

d) **Convert the document to a Document-Term matrix with Tf-Idf weighting. How many words are present?**

```
> dtm_tfidf <- DocumentTermMatrix(corpt, control = list(weighting = weightTfIdf))
> data <- read.csv("airline_tweets_sentiment2(1).csv")
> corp <- VCorpus(VectorSource(data$tweet))
> corpt <- tm_map(corp,removeNumbers)
> corpt <- tm_map(corpt,removePunctuation)
> corpt <- tm_map(corpt,removeWords, stopwords("english"))
> dtm_tfidf <- DocumentTermMatrix(corpt, control = list(weighting = weightTfIdf))
> inspect(dtm_tfidf)
<<DocumentTermMatrix (documents: 14640, terms: 14579)>>
```

[2] Modified from https://www.kaggle.com/crowdflower/twitter-airline-sentiment

There are 14579 terms.

e) **Remove all words from the Document-Term matrix with sparsity >= 99%. How many words are now present?**

```
> dtm_tfidf <- removeSparseTerms(dtm_tfidf, .99)
> inspect(dtm_tfidf)
<<DocumentTermMatrix (documents: 14640, terms: 169)>>
```

There are 169 terms now present.

f) **Compute the cosine similarity between the first tweet and the remaining tweets. It is easiest to write R code from scratch instead of installing a library just for this:**

   a) **Define a function that takes two vectors as input and computes their cosine similarity:** $cosine(A,B) = \frac{sum(A_i B_i)}{\sqrt{sum(A_i^2) \times sum(B_i^2)}}$

```
> cosineSim <- function(A, B) {
sum1 <- sum(A*B)
sum2 <- sum(A*A)*sum(B,B)
sum3 <- sum1/sqrt(sum2)
return(sum3)
}
```

   b) **Convert the Document-Term sparse matrix to a regular matrix, like so:**
      i) mymatrix <- as.matrix(mydtm)

```
> matrix <- as.matrix(dtm_tfidf)
```

   c) **Call the cosine function with mymatrix[1,:] and every other row. Find the index which gives the maximum value.**

```
for (i in 1:length(matrix[1,])) { cosVec[i] <- cosineSim(matrix[1,],matrix[i,])  }
> print(which.max(cosVec))
[1] 10144
```

Index 10144 has the highest value.

g) **Print the full text of the tweet with the maximum similarity**

```
> corp[[10144]]$content
```

a) "@AmericanAir Err 2 hour wait time for the Exec. Plat. line?"

h) Print the corresponding sentiment of the above tweet. This is the predicted sentiment of the first tweet. Does this match the given sentiment (from row 1)?

```
> data <- read.csv("airline_tweets_sentiment2(1).csv")
> corp <- VCorpus(VectorSource(data))
> corp[[2]]$content[[10144]]
[1] "negative"
> corp[[2]]$content[[1]]
[1] "negative"
```

Yes it matches.

i) Is this bag-of-words approach in general a good way to predict sentiment in tweets? Why or why not? Answer in 2-3 sentences.

Yes, it matches the predicted sentiment and carrys similarities within the tweets. This seems like a decently good approach in predicting sentiments.