Summer Research Final Report

Zhirui Li

7/23/2022

Explanatory Data Analysis

```
## [1] 51738    50
```

We are going to use the Listeria Monocytogenes dataset from the National Center for Biotechnology Information to predict Listeriosis outbreak. The dataset we are working with has 51738 observations and 50 columns (variables).

```
## [1] 0.004774054
```

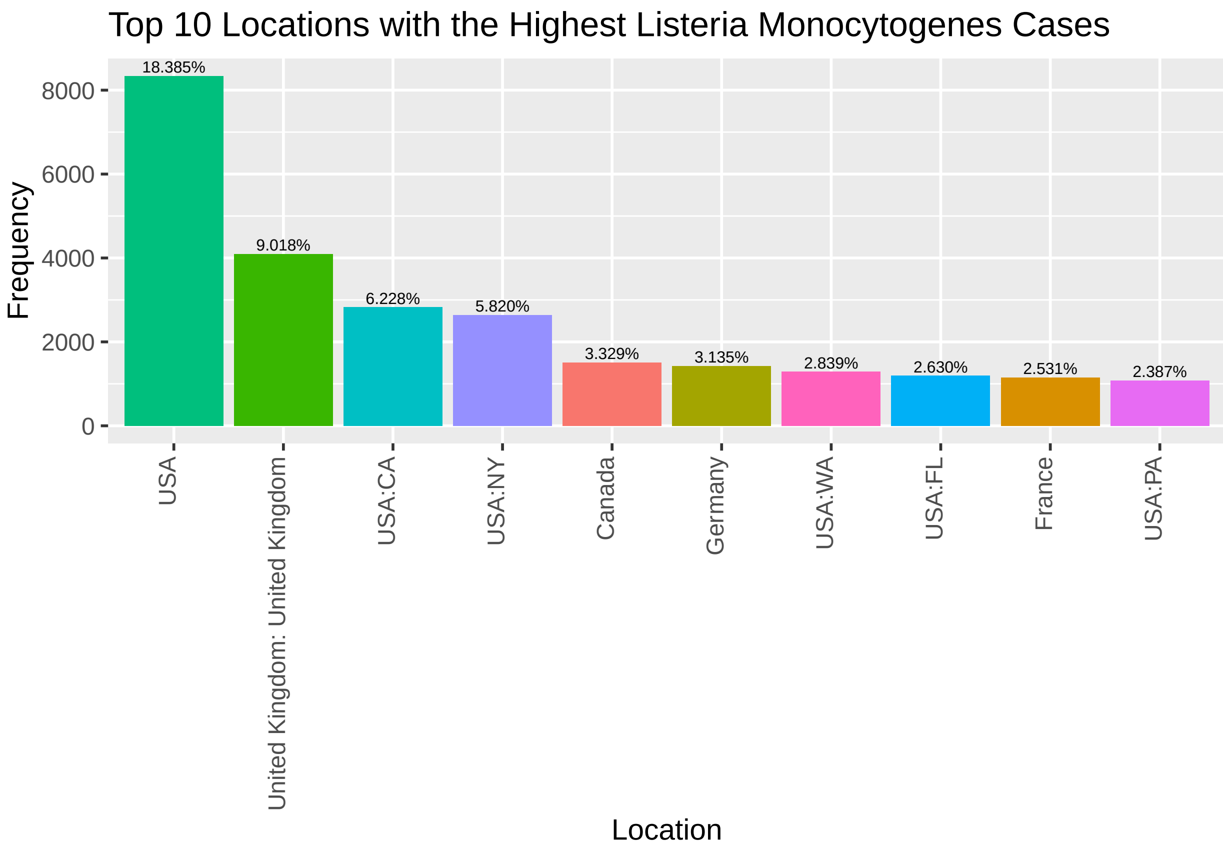
```
##
##     0     1
## 51491   247
```

The 'Outbreak' column in the dataset indicates that if there is an actual outbreak for the Listeriosis infection. Approximately 99.5% of its information are missing so that for all the missing values, we encoded it as 0 (indicating that there is no Listeria outbreak). If there is a data entry for the 'Outbreak' column, we encoded it as 1 (indicating that there is a Listeria outbreak). We can see that there are a total of 247 Listeria outbreaks.

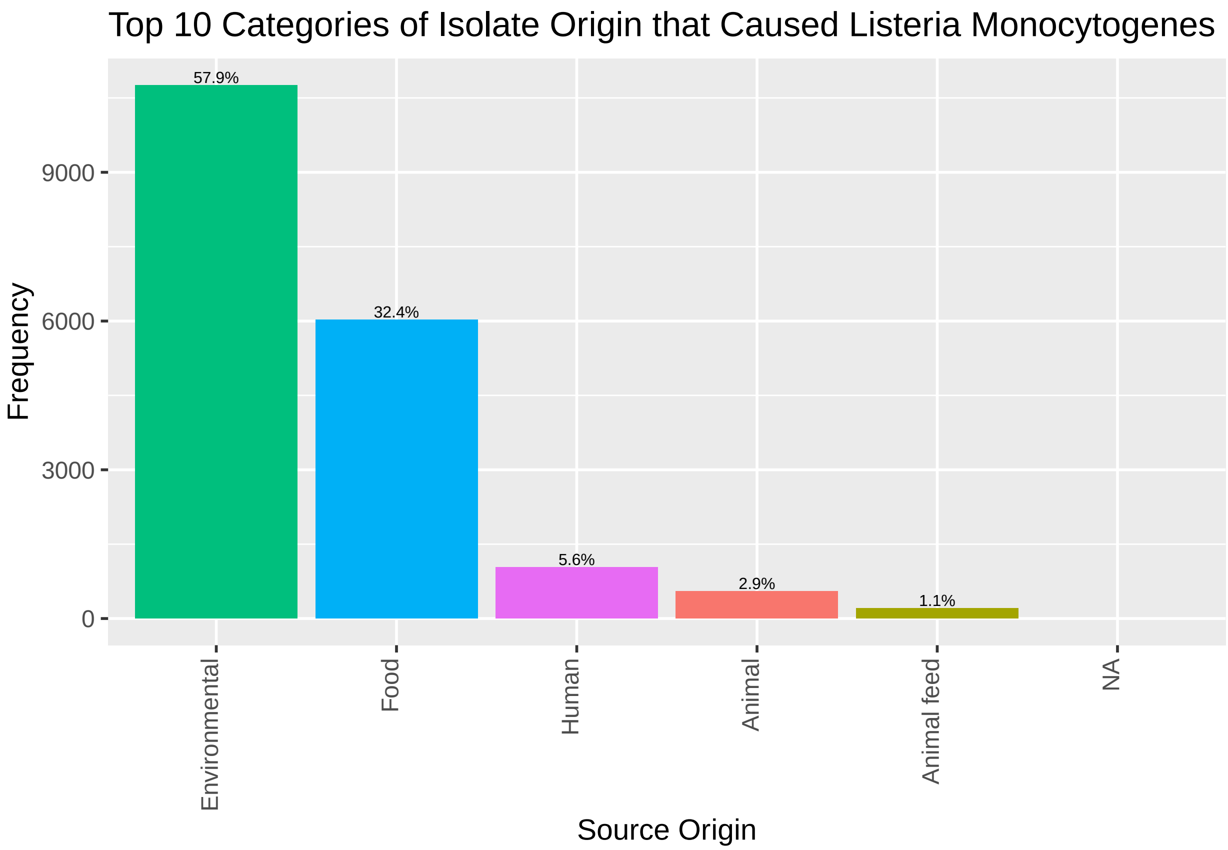
```
##                Library_layout                        Method
##                    0.88265878                    0.92315126
##                 WGS_accession                    WGS_prefix
##                    0.92693958                    0.92693958
##                  Host_disease                         TaxID
##                    0.13011713                    1.00000000
## PFGE_secondary_enzyme_pattern   PFGE_primary_enzyme_pattern
##                    0.05883490                    0.06250725
##                         Level                  Species_TaxID
##                    0.93362712                    1.00000000
##                      Outbreak              SRA_release_date
##                    1.00000000                    0.88265878
##                    SRA_Center                           Run
##                    0.88265878                    0.88265878
##                      Platform             AMR_genotypes_core
##                    0.88265878                    1.00000000
##                       Contigs                           N50
##                    1.00000000                    1.00000000
##                        Length                    BioProject
##                    1.00000000                    1.00000000
##               Collection_date               Stress_genotypes
##                    0.84181839                    0.34038811
##                       Lat/Lon                  Collected_by
##                    0.13224323                    0.76711508
```

```
##   PD_Ref_Gene_Catalog_version   AMRFinderPlus_analysis_type
##                    1.00000000                    1.00000000
##          AMRFinderPlus_version               Scientific_name
##                    1.00000000                    1.00000000
##            Virulence_genotypes                 AST_phenotypes
##                    0.00000000                    0.00000000
##                    K-mer_group                Organism_group
##                    1.00000000                    1.00000000
##                           Host                   Source_type
##                    0.22341413                    0.35888515
##                 IFSAC_category                        Strain
##                    0.36298272                    0.82486760
##            Isolate_identifiers                       Serovar
##                    0.99984537                    0.19069156
##                        Isolate                   Create_date
##                    1.00000000                    1.00000000
##                       Location              Isolation_source
##                    0.87678302                    0.77848003
##                 Isolation_type                   SNP_cluster
##                    0.88447563                    0.82788279
##                       Min-same                      Min-diff
##                    0.72407128                    0.50191349
##                      BioSample                      Assembly
##                    1.00000000                    0.93362712
##                  AMR_genotypes                Computed_types
##                    1.00000000                    0.00000000
```

Above table shows the missing pattern for each variable in the dataset. Since we are going to use times series models to forecast Listeriosis infection cases, we don't need to conduct a complete case analysis because missing values in each variable will not affect time series model's performance.



Above graph depicts top 10 location with the highest Listeriosis cases. We can see that USA has the highest cases and accounts for almost 19% of all the occurences.



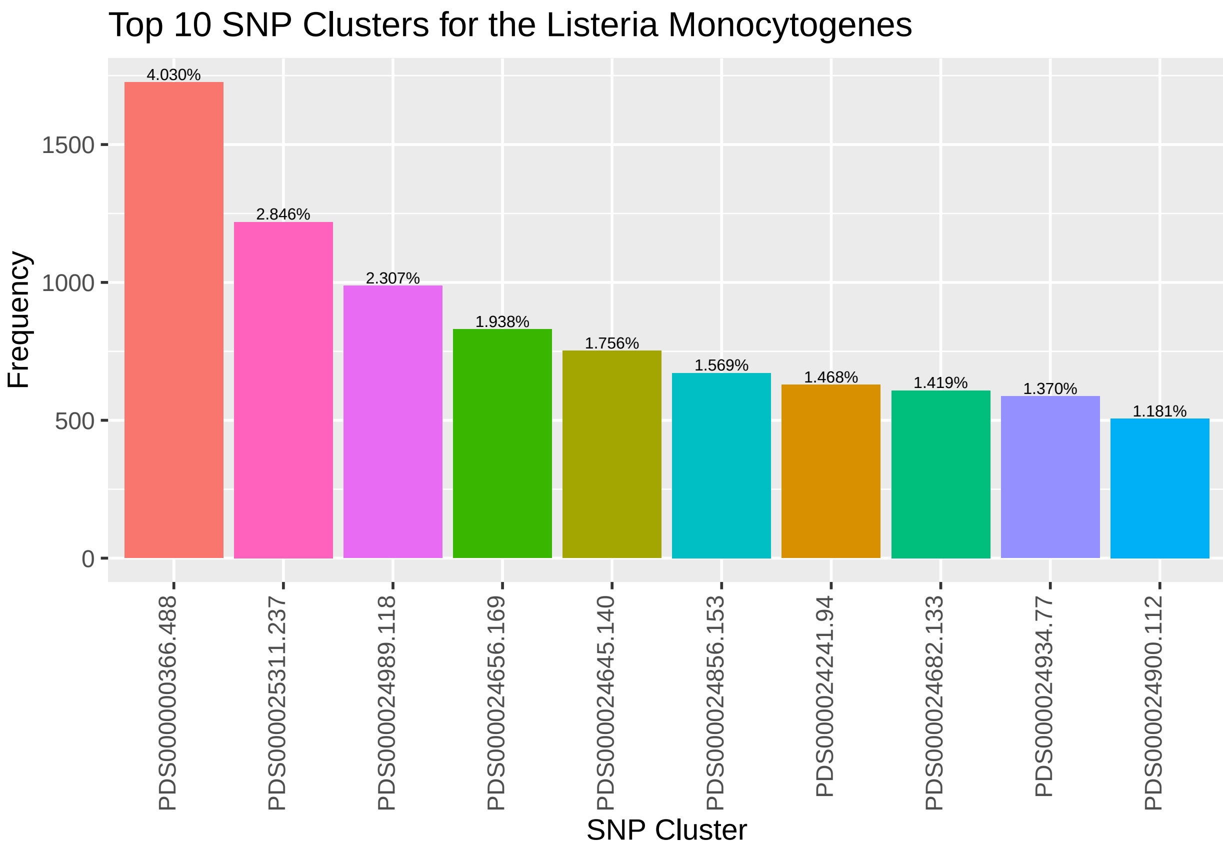
Above graph depicts the general categories of isolate origin. We can see that most of the Listeria isolates have an environmental origin (about 58% of all the data); 33% of all the Listeria isolates are coming from food sources.

```
## [1] 4378
```

In the whole dataset, we have 4378 different SNP clusters for the Listeria Monocytogenes.

```
## [1] 0.198842
```

The top 10 SNP clusters for the Listeria Monocytogenes captured almost 20% of the whole dataset.



Above graph depicts top 10 SNP clusters for Listeria Monocytogenes cases. We can see that cluster PDS000000366.488 is the biggest one and accounts for almost 4% of all the cases.

Summary Statistics for each SNP Cluster

```
## [1] 8517   50
```

Next, we are going to compare and contrast the top 10 SNP clusters. I filtered out all the observations that are not in the top 10 SNP clusters for Listeria Monocytogenes. Right now, we have 8517 observations left.

Then, for each cluster in the top 10 SNP clusters, I selected columns such as 'Location', 'Source_type', 'Min-same', and 'Min-diff' to form the summary table (those are the most relevant information in the dataset). 'Location' represents the geographical origin of the sample. 'Source_type' represents the general category of isolate origin. 'Min-same' represents the minimum SNP distance from this isolate to one of the same isolation type. 'Min-diff' represents the minimum SNP distance from this isolate to one of a different isolation type.

```
## [1] "this is the summary table for SNP cluster PDS000000366.488"
##     Location           Source_type     Min-same         Min-diff
##  USA:NY :226   Animal        :  1   Min.   : 0.000   Min.   : 0.00
##  Canada :183   Animal feed   : 10   1st Qu.: 1.000   1st Qu.:11.00
##  USA    :180   Environmental:796   Median : 2.000   Median :17.00
##  USA:CA :158   Food          :255   Mean   : 4.666   Mean   :17.49
##  USA:FL :136   Human         : 12   3rd Qu.: 6.000   3rd Qu.:24.00
##  (Other):808   NA's          :652   Max.   :45.000   Max.   :56.00
##  NA's   : 35                        NA's   :68       NA's   :68
##            SNP_cluster
##  PDS000000366.488:1726
##  PDS000000204.5  :   0
```

```
##  PDS000000211.14 :   0
##  PDS000000212.17 :   0
##  PDS000000217.19 :   0
##  PDS000000218.24 :   0
##  (Other)         :   0
## [1] "this is the summary table for SNP cluster PDS000025311.237"
##                           Location          Source_type     Min-same
##  United Kingdom: United Kingdom:200   Animal       :  0   Min.   : 0.000
##  Germany                       :193   Animal feed  :  1   1st Qu.: 0.000
##  Denmark                       : 51   Environmental: 52   Median : 2.000
##  Italy                         : 35   Food         :192   Mean   : 6.198
##  USA                           : 35   Human        :  7   3rd Qu.: 8.000
##  (Other)                       :398   NA's         :967   Max.   :38.000
##  NA's                          :307                       NA's   :143
##     Min-diff              SNP_cluster
##  Min.   : 0.00   PDS000025311.237:1219
##  1st Qu.: 5.00   PDS000000204.5  :   0
##  Median :13.00   PDS000000211.14 :   0
##  Mean   :14.03   PDS000000212.17 :   0
##  3rd Qu.:23.00   PDS000000217.19 :   0
##  Max.   :40.00   PDS000000218.24 :   0
##  NA's   :143     (Other)         :   0
## [1] "this is the summary table for SNP cluster PDS000024989.118"
##     Location          Source_type     Min-same        Min-diff
##  USA:TX :142   Animal       :  0   Min.   : 0.000   Min.   : 0.00
##  USA    :118   Animal feed  :  1   1st Qu.: 0.000   1st Qu.: 9.00
##  USA:CA :105   Environmental:540   Median : 1.000   Median :21.00
##  USA:WI : 68   Food         :143   Mean   : 4.334   Mean   :19.32
##  USA:NY : 58   Human        : 13   3rd Qu.: 5.000   3rd Qu.:29.00
##  (Other):454   NA's         :291   Max.   :41.000   Max.   :49.00
##  NA's   : 43                       NA's   :31       NA's   :31
##            SNP_cluster
##  PDS000024989.118:988
##  PDS000000204.5  :  0
##  PDS000000211.14 :  0
##  PDS000000212.17 :  0
##  PDS000000217.19 :  0
##  PDS000000218.24 :  0
##  (Other)         :  0
## [1] "this is the summary table for SNP cluster PDS000024656.169"
##                           Location          Source_type     Min-same
##  United Kingdom: United Kingdom:458   Animal       :  0   Min.   : 0.000
##  France                        : 37   Animal feed  :  0   1st Qu.: 0.000
##  Australia                     : 21   Environmental: 19   Median : 1.000
##  Canada                        : 20   Food         : 52   Mean   : 4.317
##  Ireland                       : 14   Human        :  1   3rd Qu.: 4.000
##  (Other)                       :134   NA's         :758   Max.   :40.000
##  NA's                          :146                       NA's   :113
##     Min-diff              SNP_cluster
##  Min.   : 0.00   PDS000024656.169:830
##  1st Qu.: 6.00   PDS000000204.5  :  0
##  Median :17.00   PDS000000211.14 :  0
##  Mean   :17.94   PDS000000212.17 :  0
##  3rd Qu.:28.00   PDS000000217.19 :  0
```

```
## Max.   :45.00   PDS000000218.24 :  0
## NA's   :113     (Other)         :  0
## [1] "this is the summary table for SNP cluster PDS000024645.140"
##                             Location          Source_type    Min-same
## United Kingdom: United Kingdom:216   Animal       :  0   Min.   : 0.000
## USA:RI                        : 68   Animal feed  :  0   1st Qu.: 0.000
## France                        : 33   Environmental:105   Median : 2.000
## Italy                         : 33   Food         : 70   Mean   : 6.188
## Denmark                       : 27   Human        :  1   3rd Qu.: 9.000
## (Other)                       :151   NA's         :576   Max.   :38.000
## NA's                          :224                       NA's   :177
##    Min-diff             SNP_cluster
## Min.   : 1    PDS000024645.140:752
## 1st Qu.: 9    PDS000000204.5  :  0
## Median :18    PDS000000211.14 :  0
## Mean   :19    PDS000000212.17 :  0
## 3rd Qu.:30    PDS000000217.19 :  0
## Max.   :47    PDS000000218.24 :  0
## NA's   :177   (Other)         :  0
## [1] "this is the summary table for SNP cluster PDS000024856.153"
##                             Location          Source_type    Min-same
## USA                           :106   Animal       :  0   Min.   : 0.000
## China:Sichuang                : 59   Animal feed  :  0   1st Qu.: 1.000
## USA:NY                        : 51   Environmental: 79   Median : 3.000
## United Kingdom: United Kingdom: 29   Food         : 46   Mean   : 7.027
## China: Beijing                : 27   Human        :  4   3rd Qu.:11.250
## (Other)                       :365   NA's         :543   Max.   :36.000
## NA's                          : 35                       NA's   :12
##    Min-diff               SNP_cluster
## Min.   : 0.00   PDS000024856.153:672
## 1st Qu.:12.00   PDS000000204.5  :  0
## Median :18.00   PDS000000211.14 :  0
## Mean   :18.04   PDS000000212.17 :  0
## 3rd Qu.:24.00   PDS000000217.19 :  0
## Max.   :58.00   PDS000000218.24 :  0
## NA's   :12      (Other)         :  0
## [1] "this is the summary table for SNP cluster PDS000024241.94"
##                             Location          Source_type    Min-same
## Italy                         :148   Animal       :  4   Min.   : 0.000
## Norway                        :112   Animal feed  :  1   1st Qu.: 0.000
## Germany                       : 39   Environmental: 33   Median : 2.000
## United Kingdom: United Kingdom: 17   Food         :157   Mean   : 5.296
## USA:VT                        : 13   Human        : 15   3rd Qu.: 7.000
## (Other)                       :140   NA's         :419   Max.   :45.000
## NA's                          :160                       NA's   :88
##    Min-diff              SNP_cluster
## Min.   : 0.00   PDS000024241.94:629
## 1st Qu.:14.00   PDS000000204.5 :  0
## Median :23.00   PDS000000211.14:  0
## Mean   :21.64   PDS000000212.17:  0
## 3rd Qu.:28.00   PDS000000217.19:  0
## Max.   :45.00   PDS000000218.24:  0
## NA's   :88      (Other)        :  0
## [1] "this is the summary table for SNP cluster PDS000024682.133"
```

```
##                                Location          Source_type     Min-same
##  Germany                          :196   Animal       :  0   Min.   : 0.000
##  United Kingdom: United Kingdom  :134   Animal feed  :  0   1st Qu.: 0.000
##  South Africa                    : 57   Environmental:  5   Median : 1.000
##  Netherlands                     : 41   Food         : 19   Mean   : 4.866
##  United Kingdom: North of England: 19   Human        :  3   3rd Qu.: 5.000
##  (Other)                         :102   NA's         :581   Max.   :40.000
##  NA's                            : 59                       NA's   :27
##     Min-diff              SNP_cluster
##  Min.   : 0.00   PDS000024682.133:608
##  1st Qu.: 6.00   PDS000000204.5  :  0
##  Median :17.00   PDS000000211.14 :  0
##  Mean   :15.07   PDS000000212.17 :  0
##  3rd Qu.:20.00   PDS000000217.19 :  0
##  Max.   :39.00   PDS000000218.24 :  0
##  NA's   :27      (Other)         :  0
## [1] "this is the summary table for SNP cluster PDS000024934.77"
##     Location          Source_type     Min-same         Min-diff
##  USA:CA :299   Animal       :  1   Min.   : 0.000   Min.   : 0.000
##  USA    :153   Animal feed  :  0   1st Qu.: 0.000   1st Qu.: 2.000
##  USA:MI : 49   Environmental:308   Median : 2.000   Median : 8.000
##  USA:OH : 13   Food         : 75   Mean   : 4.811   Mean   : 9.063
##  Canada : 12   Human        :  9   3rd Qu.: 7.000   3rd Qu.:12.000
##  (Other): 53   NA's         :194   Max.   :33.000   Max.   :41.000
##  NA's   :  8                       NA's   :1        NA's   :1
##           SNP_cluster
##  PDS000024934.77:587
##  PDS000000204.5 :  0
##  PDS000000211.14:  0
##  PDS000000212.17:  0
##  PDS000000217.19:  0
##  PDS000000218.24:  0
##  (Other)        :  0
## [1] "this is the summary table for SNP cluster PDS000024900.112"
##                               Location          Source_type     Min-same
##  United Kingdom: United Kingdom:119   Animal       :  0   Min.   : 0.000
##  Canada                        : 55   Animal feed  :  0   1st Qu.: 0.000
##  Australia                     : 39   Environmental: 50   Median : 2.000
##  France                        : 27   Food         : 44   Mean   : 5.463
##  South Africa                  : 14   Human        :  0   3rd Qu.: 7.750
##  (Other)                       :143   NA's         :412   Max.   :42.000
##  NA's                          :109                       NA's   :96
##     Min-diff              SNP_cluster
##  Min.   : 0.00   PDS000024900.112:506
##  1st Qu.:16.00   PDS000000204.5  :  0
##  Median :23.00   PDS000000211.14 :  0
##  Mean   :23.19   PDS000000212.17 :  0
##  3rd Qu.:30.00   PDS000000217.19 :  0
##  Max.   :60.00   PDS000000218.24 :  0
##  NA's   :96      (Other)         :  0
```

For SNP cluster PDS000000366.488, most of the cases took place in New York, Canada, California and Florida. The origin for all the Listeria cases is mainly environmental. The mean for 'Min-same' is 4.66 and the mean for 'Min-diff' is 17.5.

For SNP cluster PDS000025311.237, most of the cases took place in United Kingdom and Germany. The origin for all the Listeria cases is mainly from food source. The mean for ‘Min-same’ is 6.2 and the mean for ‘Min-diff’ is 14.

For SNP cluster PDS000024989.118, most of the cases took place in Texas and California. The origin for all the Listeria cases is mainly environmental. The mean for ‘Min-same’ is 4.3 and the mean for ‘Min-diff’ is 19.

For SNP cluster PDS000024656.169, most of the cases took place in United Kingdom. The origin for all the Listeria cases is mainly from food source. The mean for ‘Min-same’ is 4.3 and the mean for ‘Min-diff’ is 18.

For SNP cluster PDS000024645.140, most of the cases took place in United Kingdom and Rhode Island. The origin for all the Listeria cases is mainly environmental. The mean for ‘Min-same’ is 6.2 and the mean for ‘Min-diff’ is 19.

For SNP cluster PDS000024856.153, most of the cases took place in USA and Sichuan(China). The origin for all the Listeria cases is mainly environmental. The mean for ‘Min-same’ is 7 and the mean for ‘Min-diff’ is 18.

For SNP cluster PDS000024241.94, most of the cases took place in Italy and Norway. The origin for all the Listeria cases is mainly from food source. The mean for ‘Min-same’ is 5.3 and the mean for ‘Min-diff’ is 21.6.

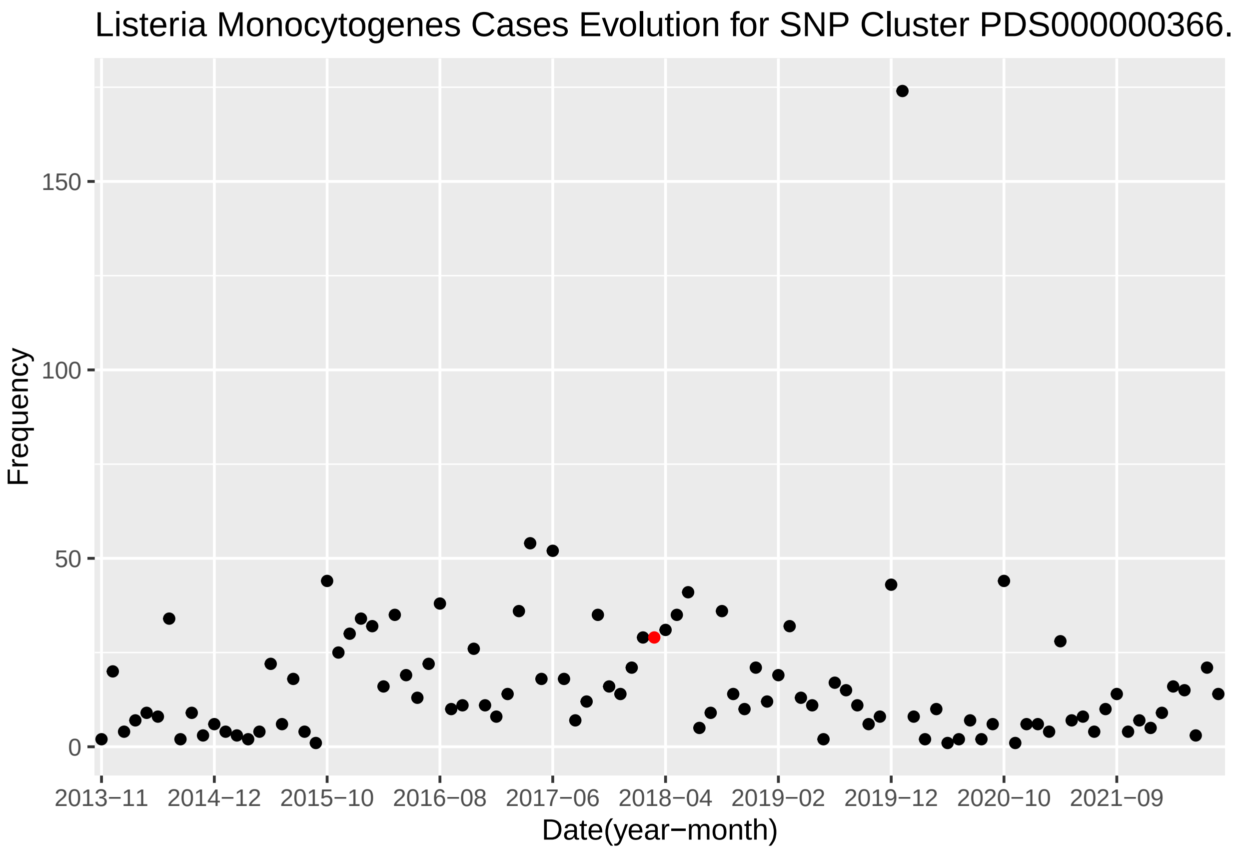
For SNP cluster PDS000024682.133, most of the cases took place in Germany, United Kingdom, and South Africa. The origin for all the Listeria cases is mainly from food source. The mean for ‘Min-same’ is 4.87 and the mean for ‘Min-diff’ is 15.1.

For SNP cluster PDS000024934.77, most of the cases took place in California. The origin for all the Listeria cases is mainly environmental. The mean for ‘Min-same’ is 4.8 and the mean for ‘Min-diff’ is 9.1.

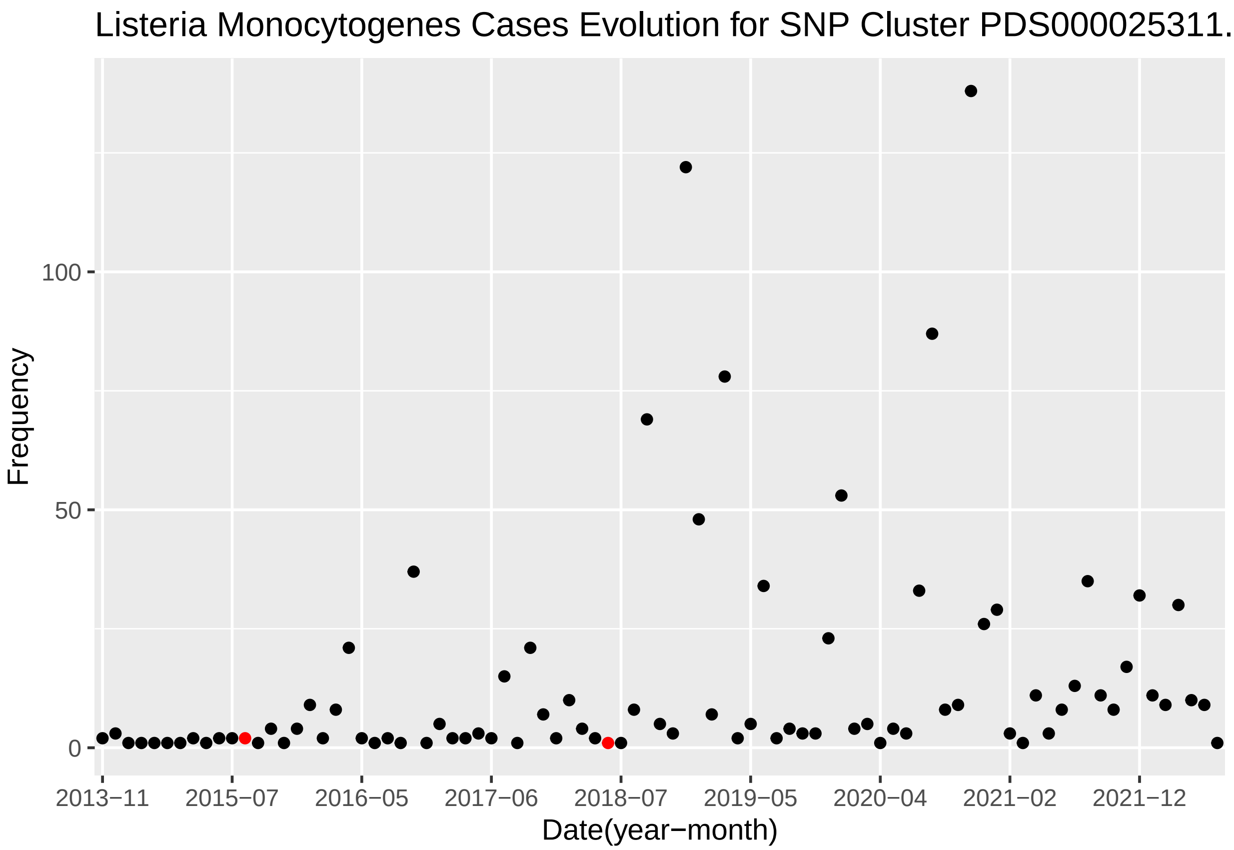
For SNP cluster PDS000024900.112, most of the cases took place in United Kingdom and Canada. The origin for all the Listeria cases is mainly environmental. The mean for ‘Min-same’ is 5.5 and the mean for ‘Min-diff’ is 23.2.

SNP Cluster Visualization by Month and by Week

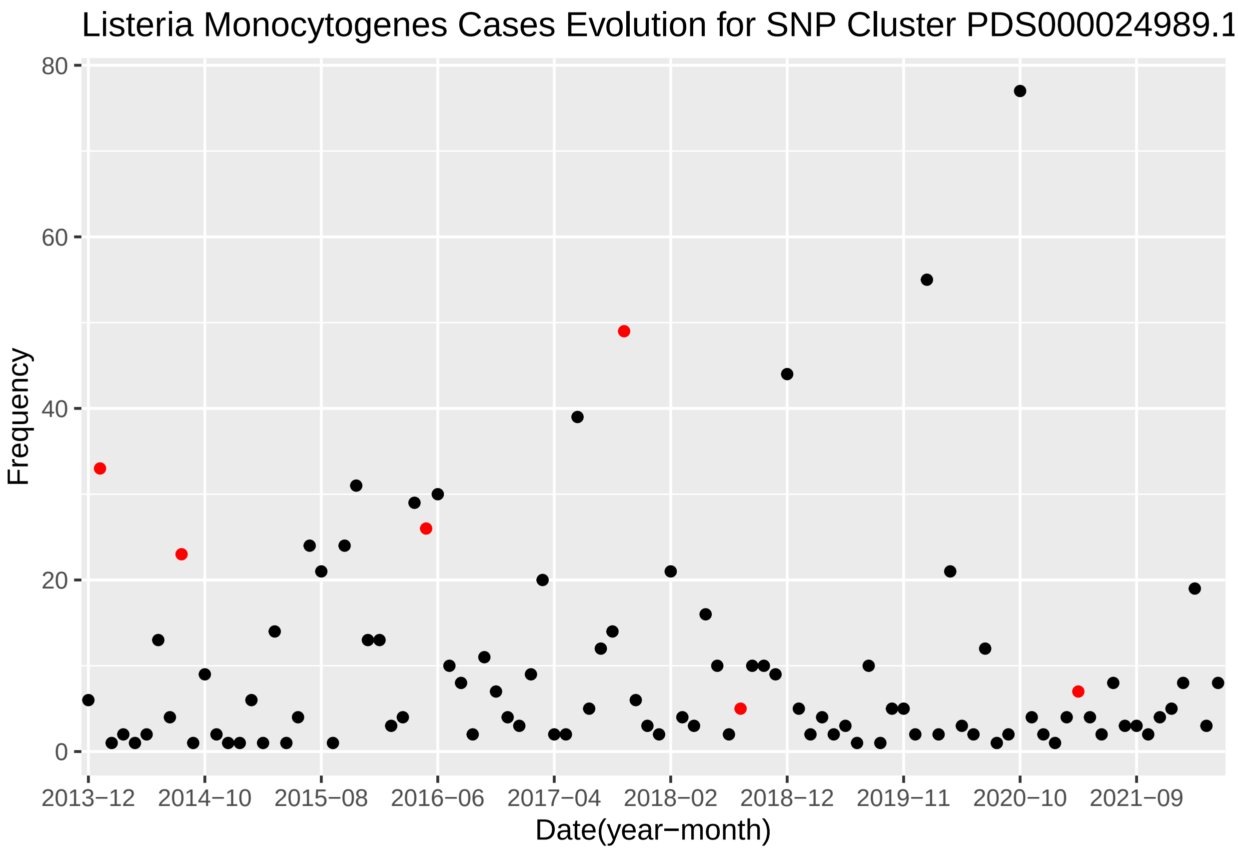
After analyzing the summary statistics for each SNP cluster, I am going to visualize the evolution of Listeria cases within each SNP cluster with month as an interval unit.



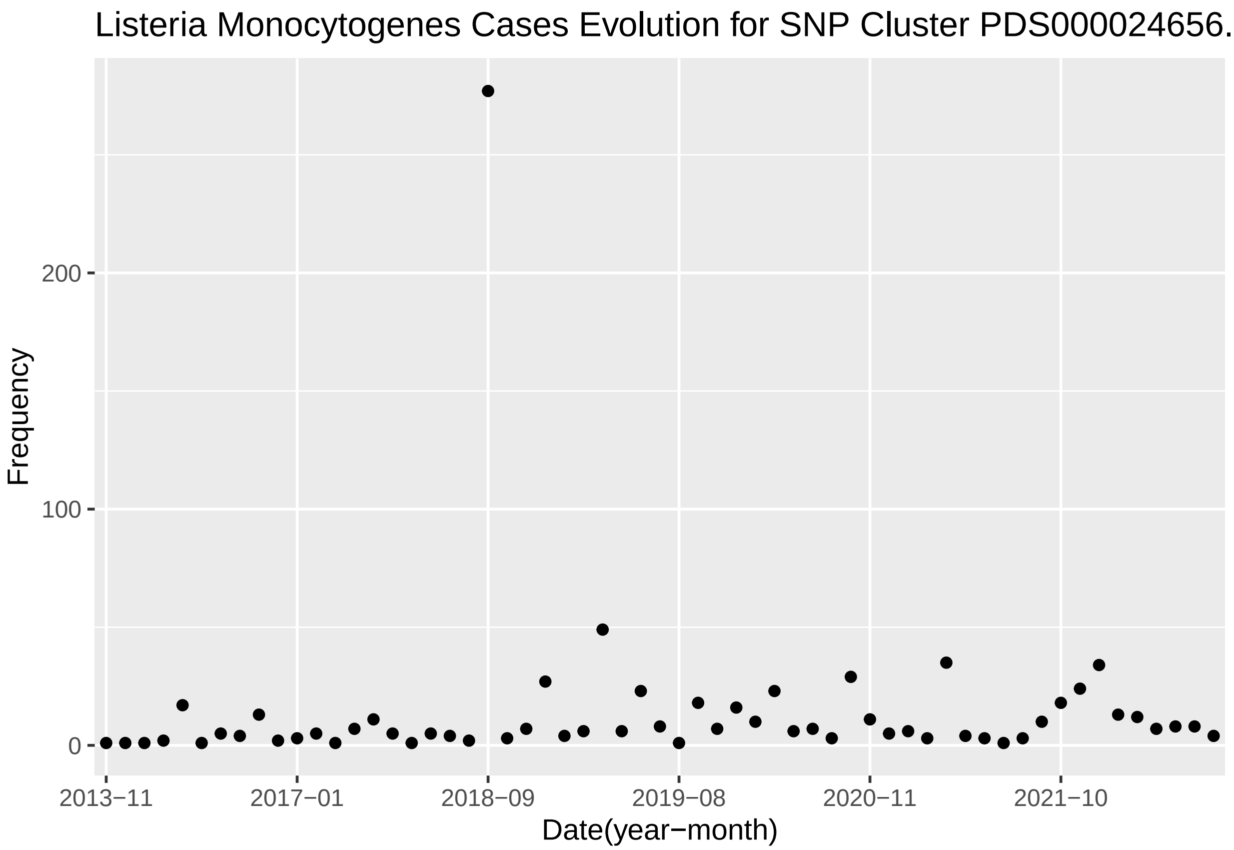

```
## [1] "SNP Cluster PDS000000366.488 has the highest cases of listeria monocytogenes at 2020-01"
```



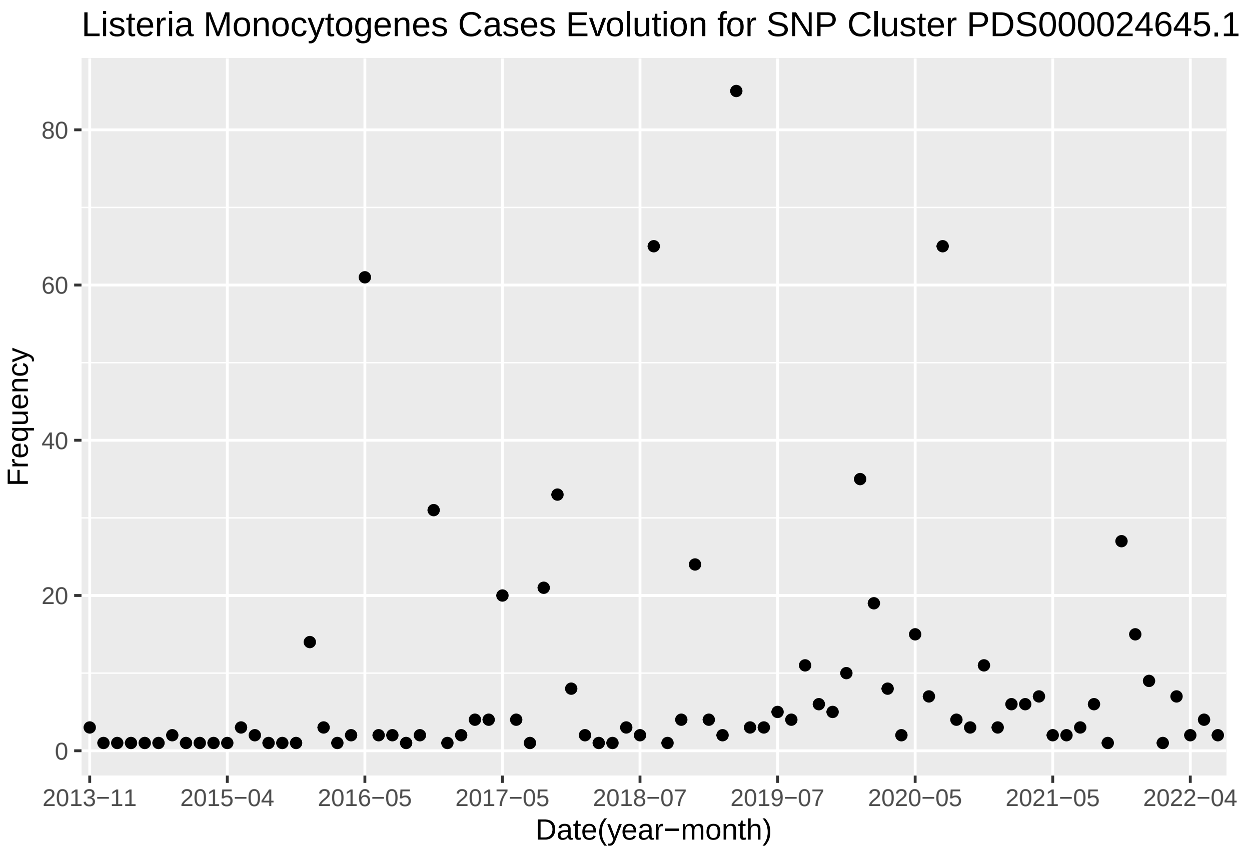

```
## [1] "SNP Cluster PDS000025311.237 has the highest cases of listeria monocytogenes at 2020-11"
```



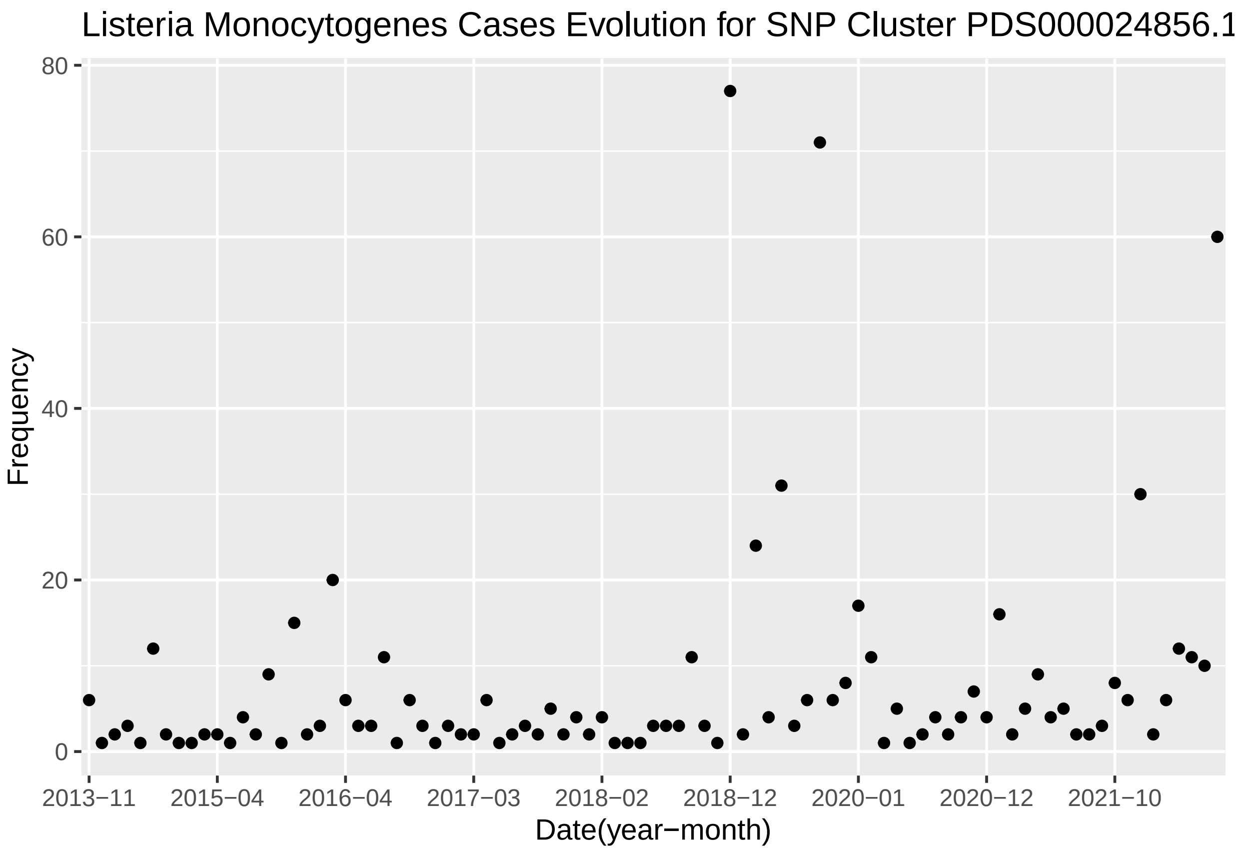

```
## [1] "SNP Cluster PDS000024989.118 has the highest cases of listeria monocytogenes at 2020-10"
```



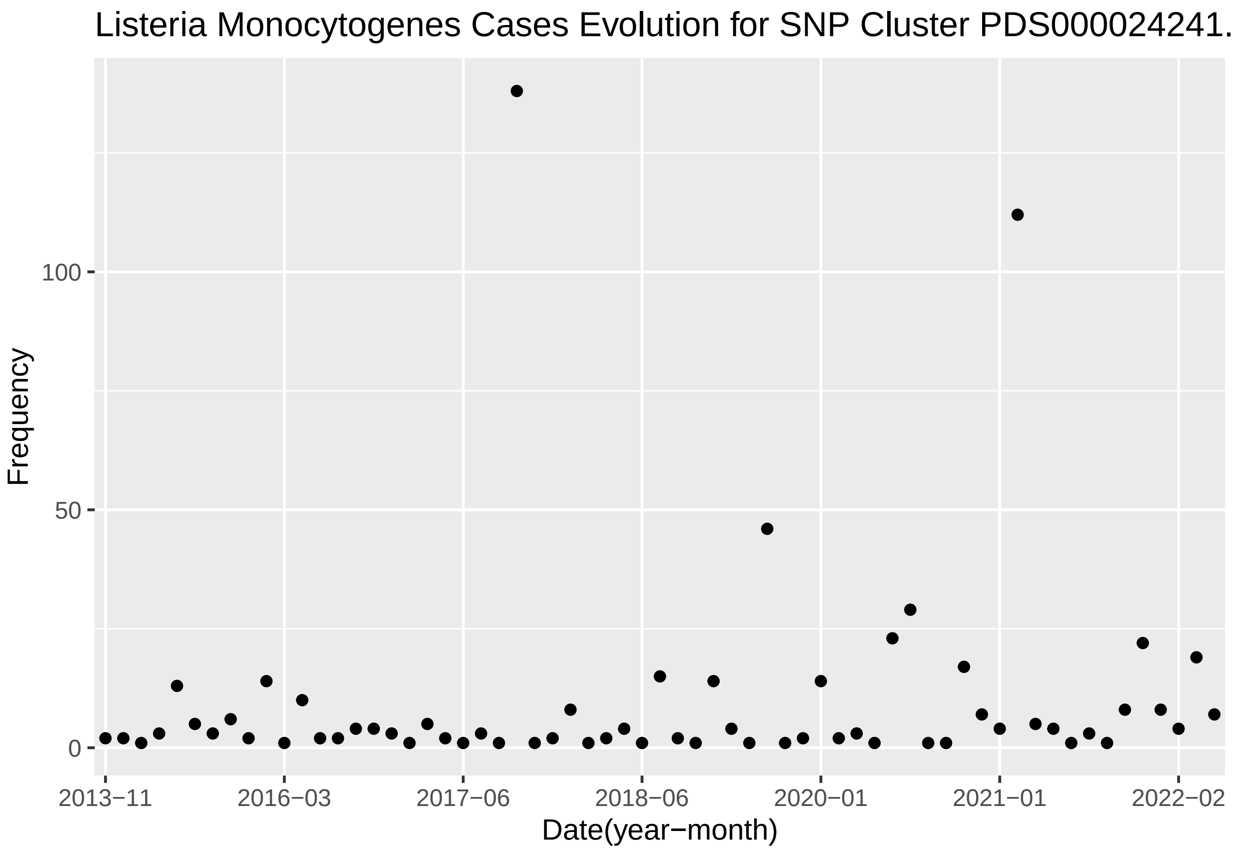

```
## [1] "SNP Cluster PDS000024656.169 has the highest cases of listeria monocytogenes at 2018-09"
```



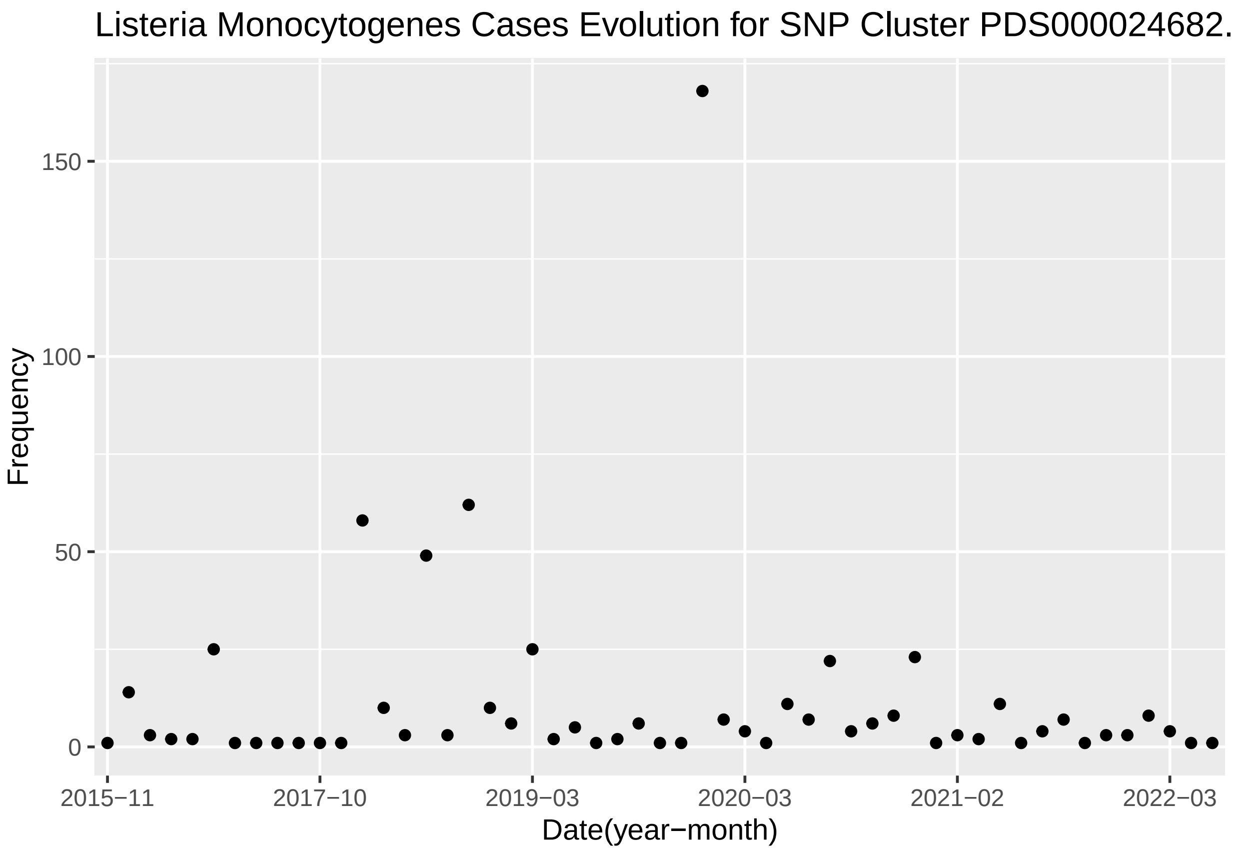

```
## [1] "SNP Cluster PDS000024645.140 has the highest cases of listeria monocytogenes at 2019-03"
```



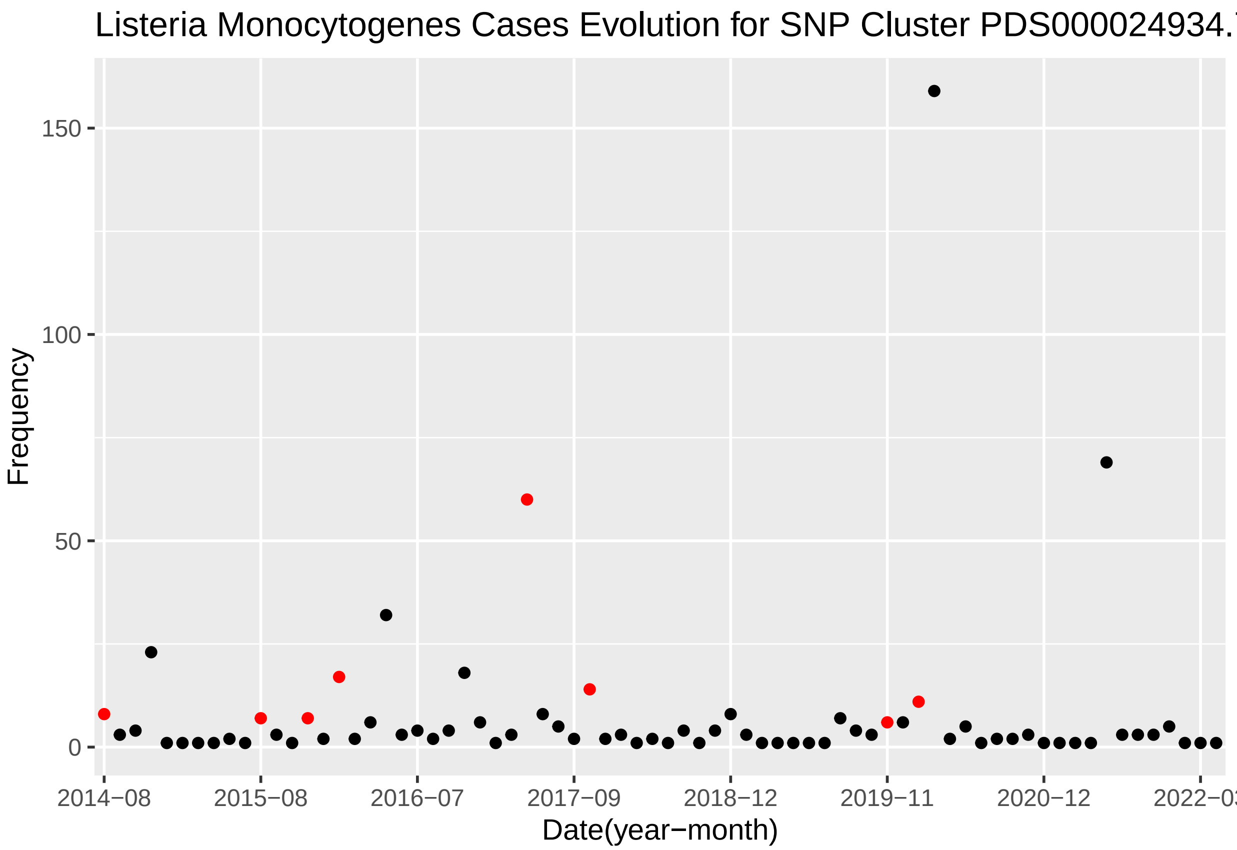

```
## [1] "SNP Cluster PDS000024856.153 has the highest cases of listeria monocytogenes at 2018-12"
```



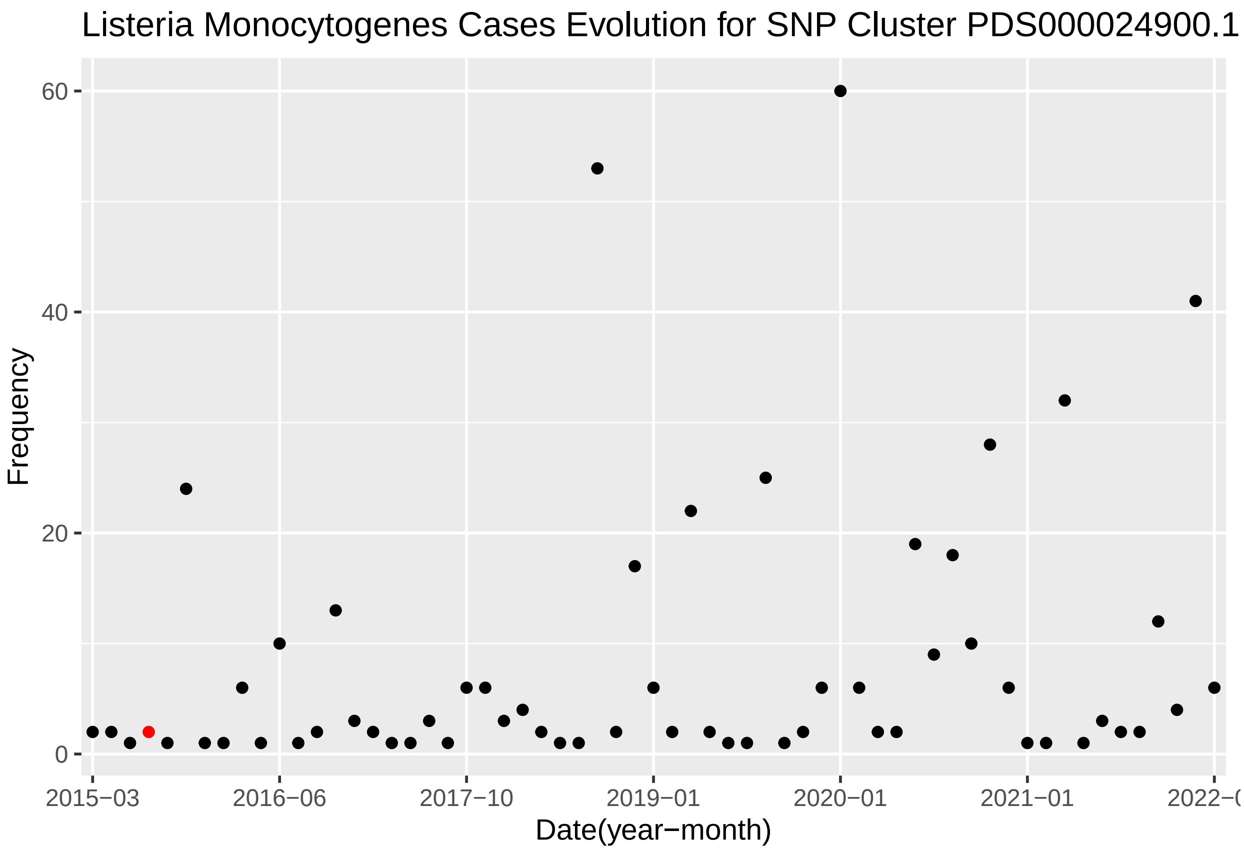

```
## [1] "SNP Cluster PDS000024241.94 has the highest cases of listeria monocytogenes at 2017-10"
```




```
## [1] "SNP Cluster PDS000024682.133 has the highest cases of listeria monocytogenes at 2020-01"
```




```
## [1] "SNP Cluster PDS000024934.77 has the highest cases of listeria monocytogenes at 2020-02"
```



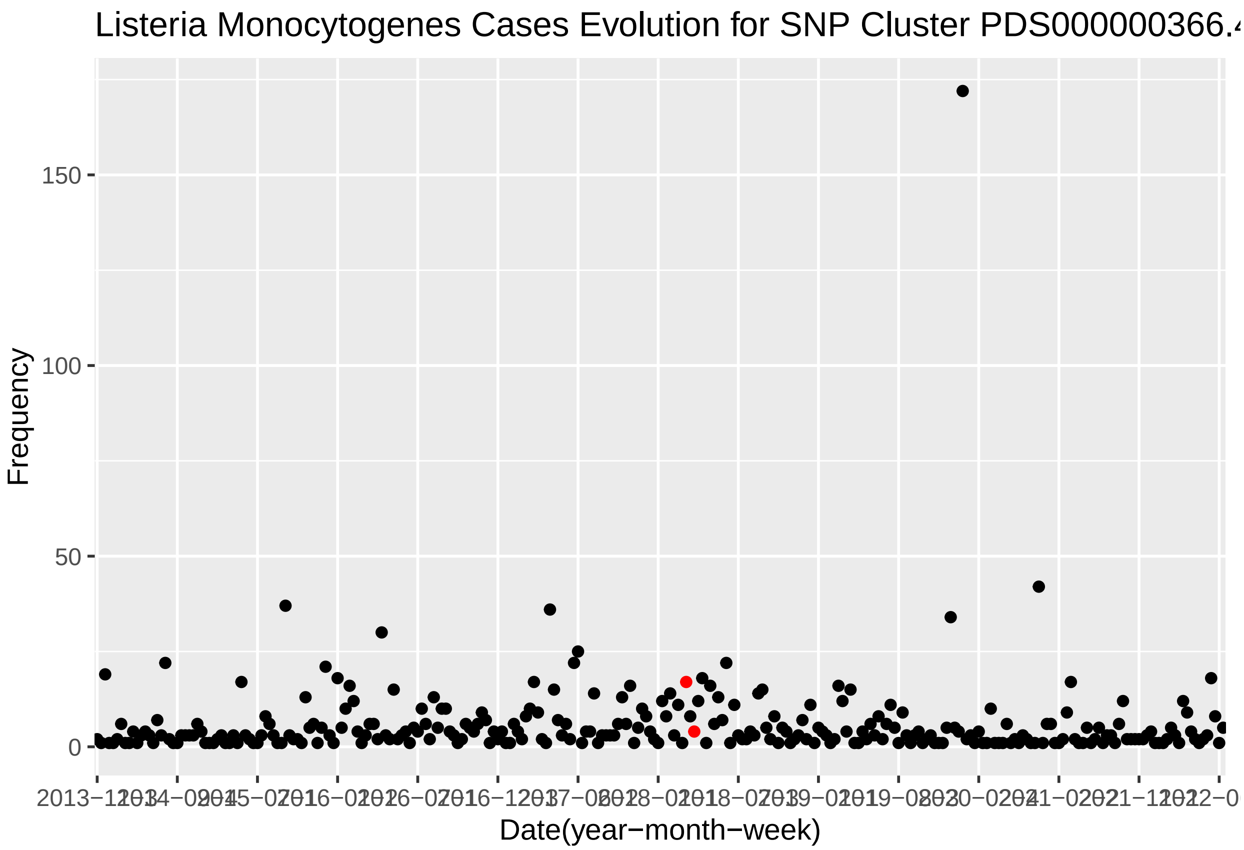

```
## [1] "SNP Cluster PDS000024900.112 has the highest cases of listeria monocytogenes at 2020-01"
```

All the dots in each graph represents the total case of Listeriosis happened in that month. The x-axis represents the year-month information and y-axis represents the total Listeria cases occurred in that month.

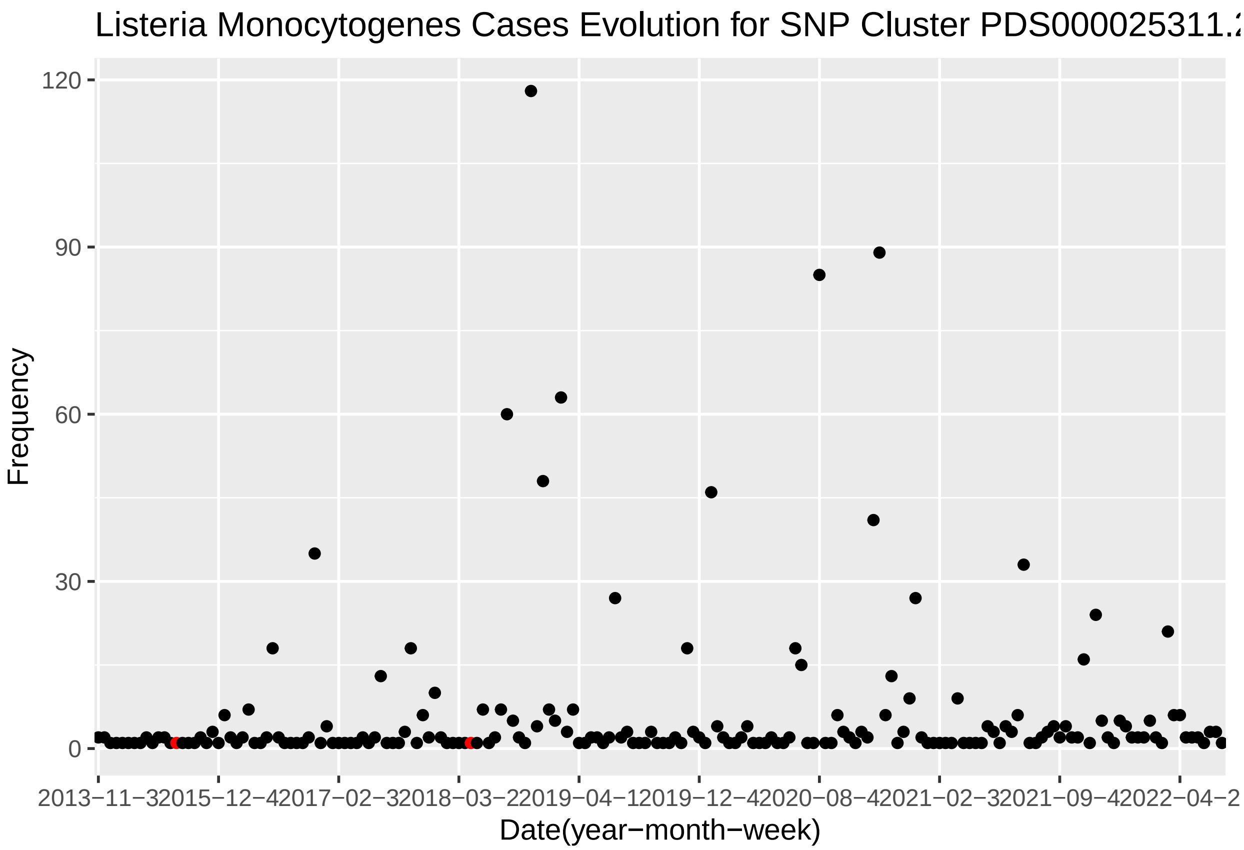
All the red dots in each graph indicates one more information that there is an actual outbreak happened being recorded by the National Center for Biotechnology Information. Surprisingly, it doesn't match my expectations that a month with the highest Listeria cases should indicate an outbreak.

For clusters PDS000000366.488(1), PDS000024682.133(8), and PDS000024900.112(10), they all had highest cases of Listeriosis at 2020-01. Clusters PDS000024656.169(4) and PDS000032941.132(11) had highest cases of Listeriosis at 2018-09. All the other clusters had the highest cases of the disease at a separate date(year-month).

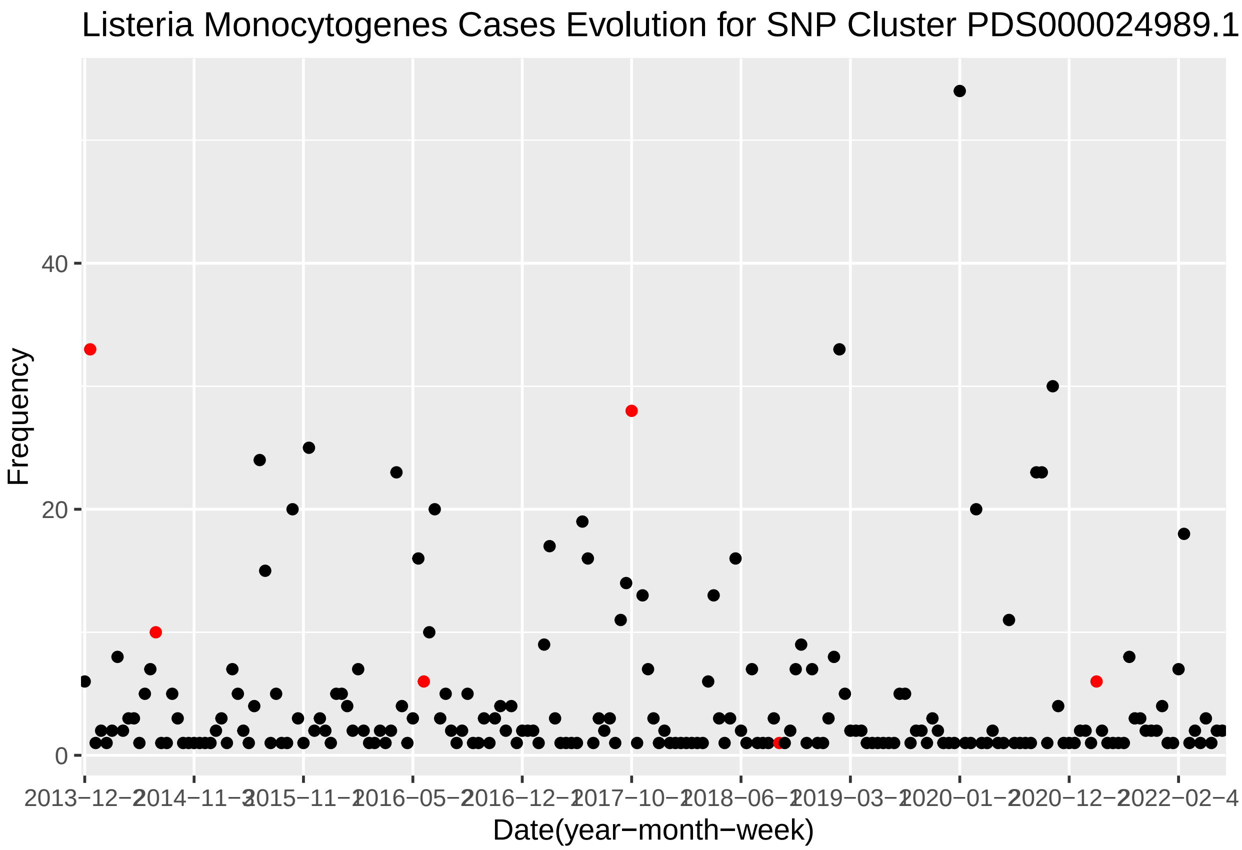
Next, I am going to visualize the evolution of cases within each SNP cluster for Listeria Monocytogenes with week as an interval unit. For each month, I encoded date 1 to date 7 as the first week; date 8 to date 14 as the second week; date 15 to date 21 as the third week; and the rest of the day within each month as the fourth week.



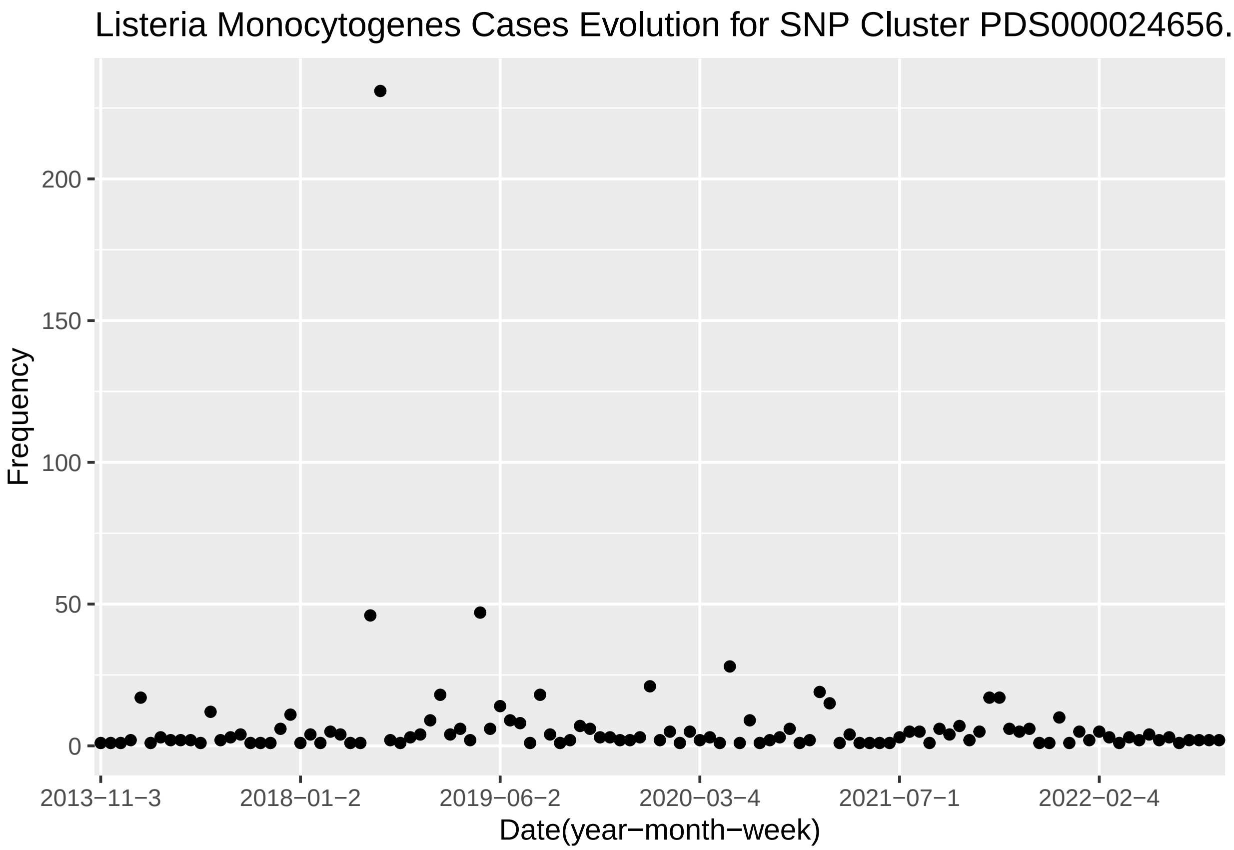

```
## [1] "SNP Cluster PDS000000366.488 has the highest cases of listeria monocytogenes at 2020-01-2"
```



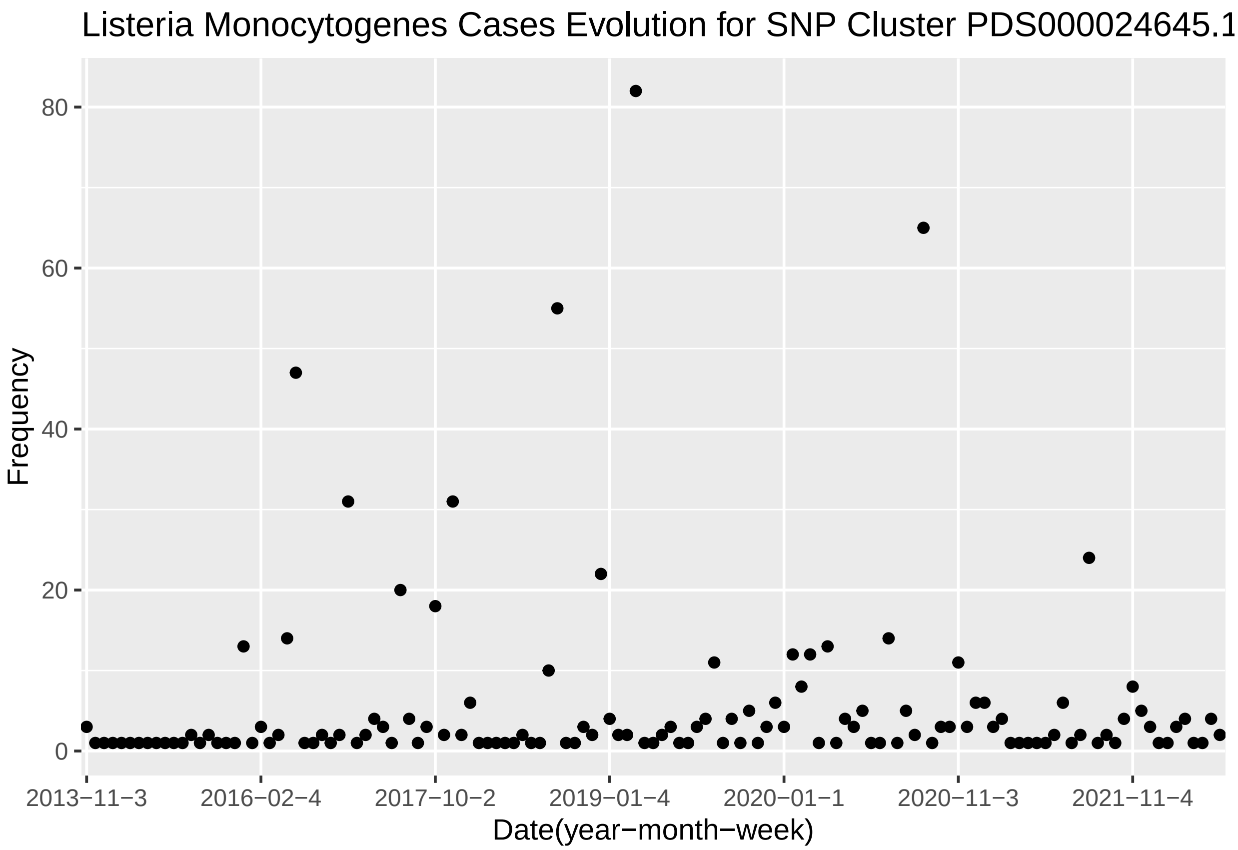

```
## [1] "SNP Cluster PDS000025311.237 has the highest cases of listeria monocytogenes at 2018-12-3"
```



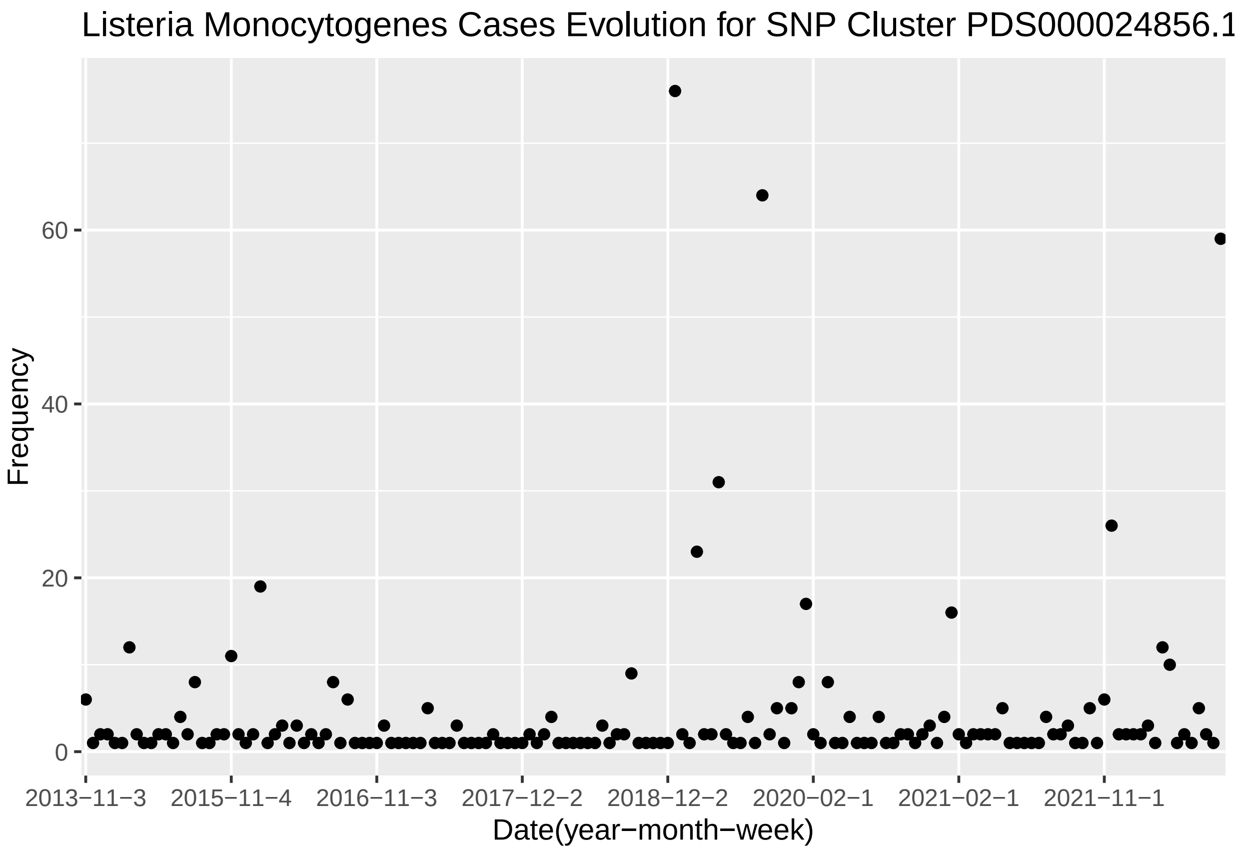

```
## [1] "SNP Cluster PDS000024989.118 has the highest cases of listeria monocytogenes at 2020-01-2"
```



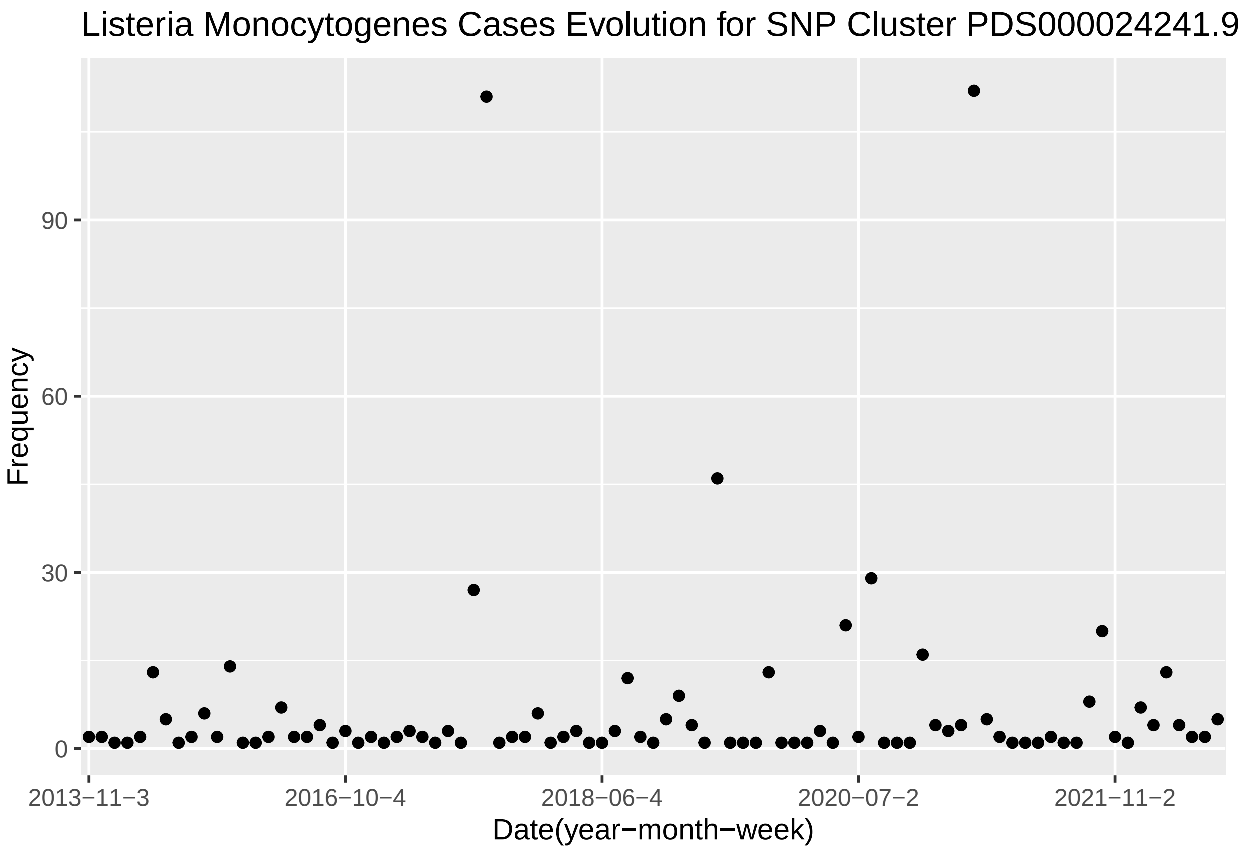

```
## [1] "SNP Cluster PDS000024656.169 has the highest cases of listeria monocytogenes at 2018-09-3"
```



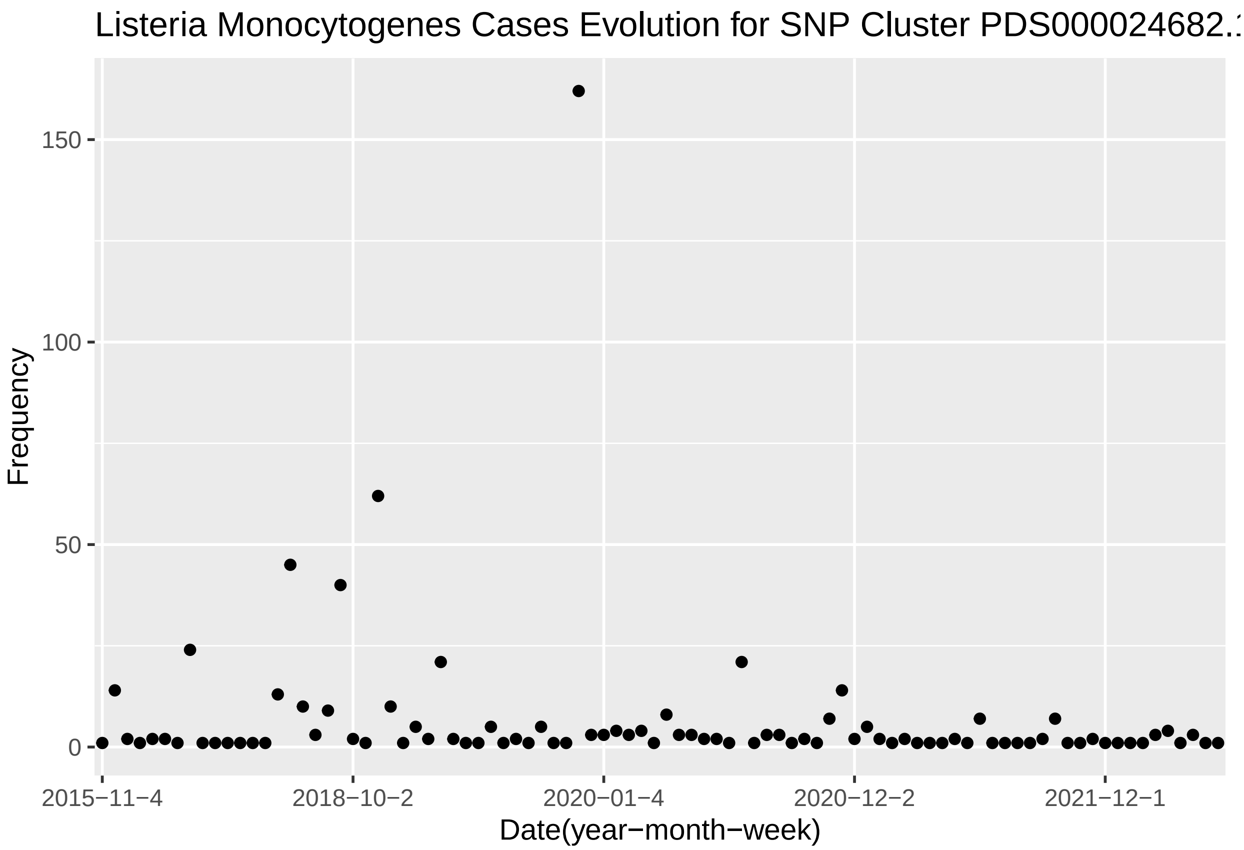

```
## [1] "SNP Cluster PDS000024645.140 has the highest cases of listeria monocytogenes at 2019-03-2"
```



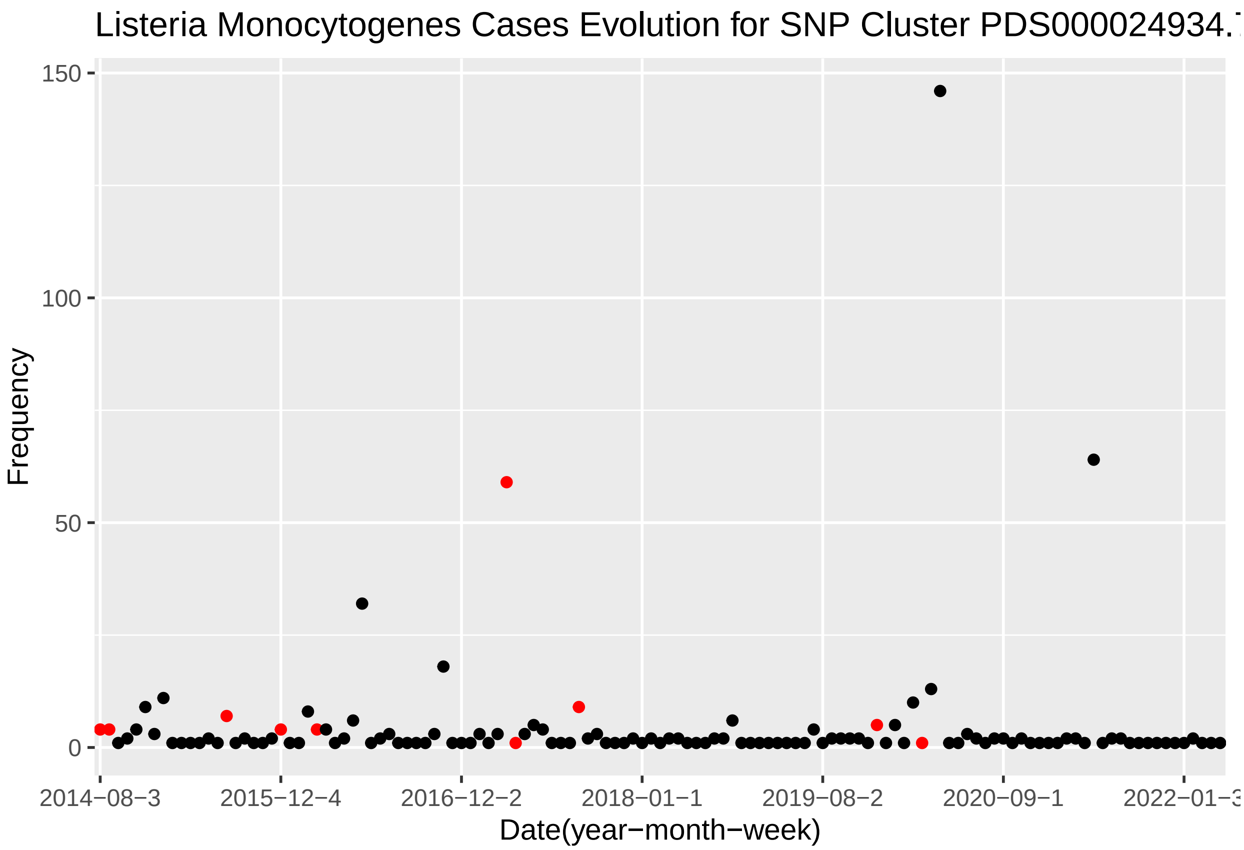

```
## [1] "SNP Cluster PDS000024856.153 has the highest cases of listeria monocytogenes at 2018-12-3"
```



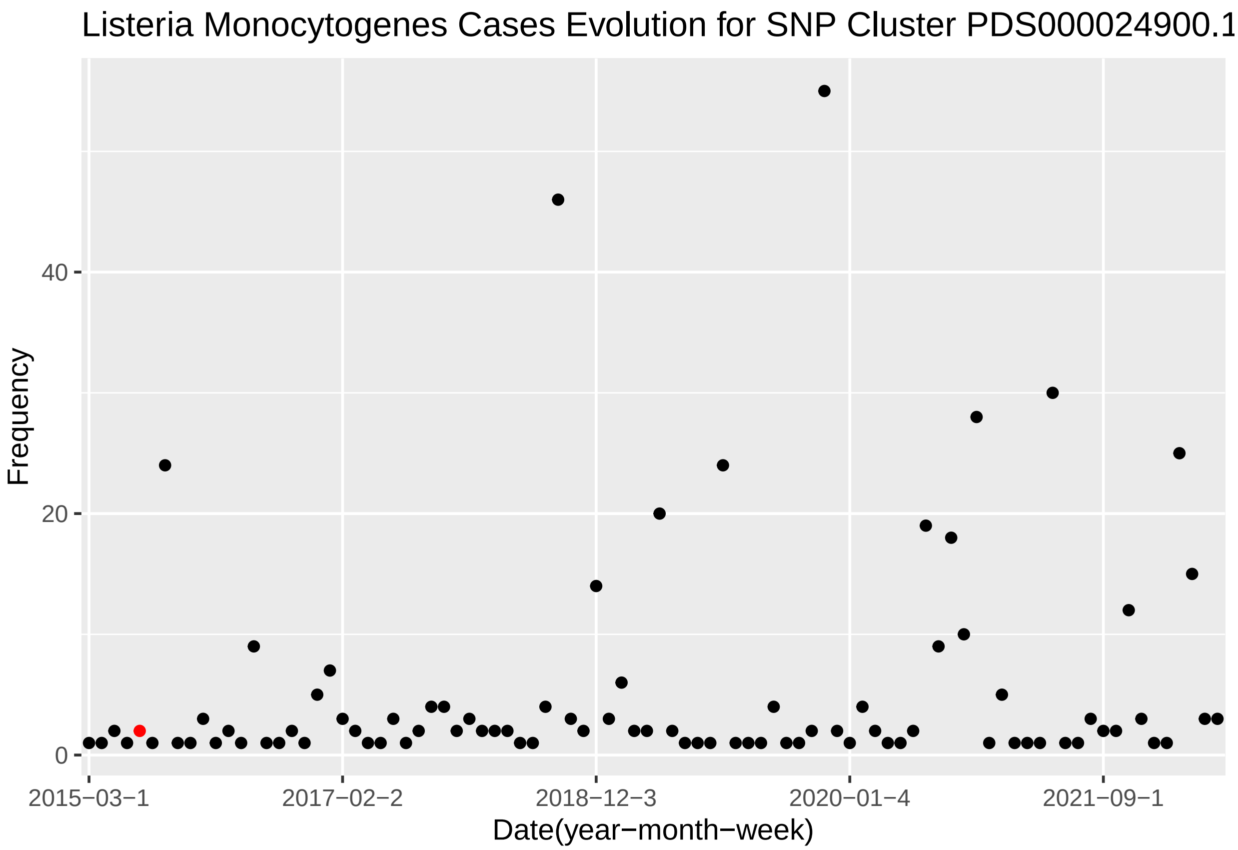

```
## [1] "SNP Cluster PDS000024241.94 has the highest cases of listeria monocytogenes at 2021-02-3"
```




```
## [1] "SNP Cluster PDS000024682.133 has the highest cases of listeria monocytogenes at 2020-01-2"
```




```
## [1] "SNP Cluster PDS000024934.77 has the highest cases of listeria monocytogenes at 2020-02-4"
```




```
## [1] "SNP Cluster PDS000024900.112 has the highest cases of listeria monocytogenes at 2020-01-2"
```

All the dots in each graph represents the total case of Listeriosis happened in that week. The x-axis represents the year-month-week information and y-axis represents the total Listeria cases occurred in that week.

All the red dots in each graph indicates one more information that there is an actual outbreak happened being recorded by the National Center for Biotechnology Information. It doesn't match my expectations that a week with the highest Listeria cases should indicate an outbreak.

Time Series Analysis for the First SNP Cluster

Using Monthly Dataset

We are going to perform time series analysis on the first SNP cluster (PDS000000366.488) first. Since we are using time series models, we don't necessarily need to use other covariates in the dataset, we will going to predict the instances of Listeriosis within a period based on the past values (autoregressive) and the average change in a data series over time (moving average).

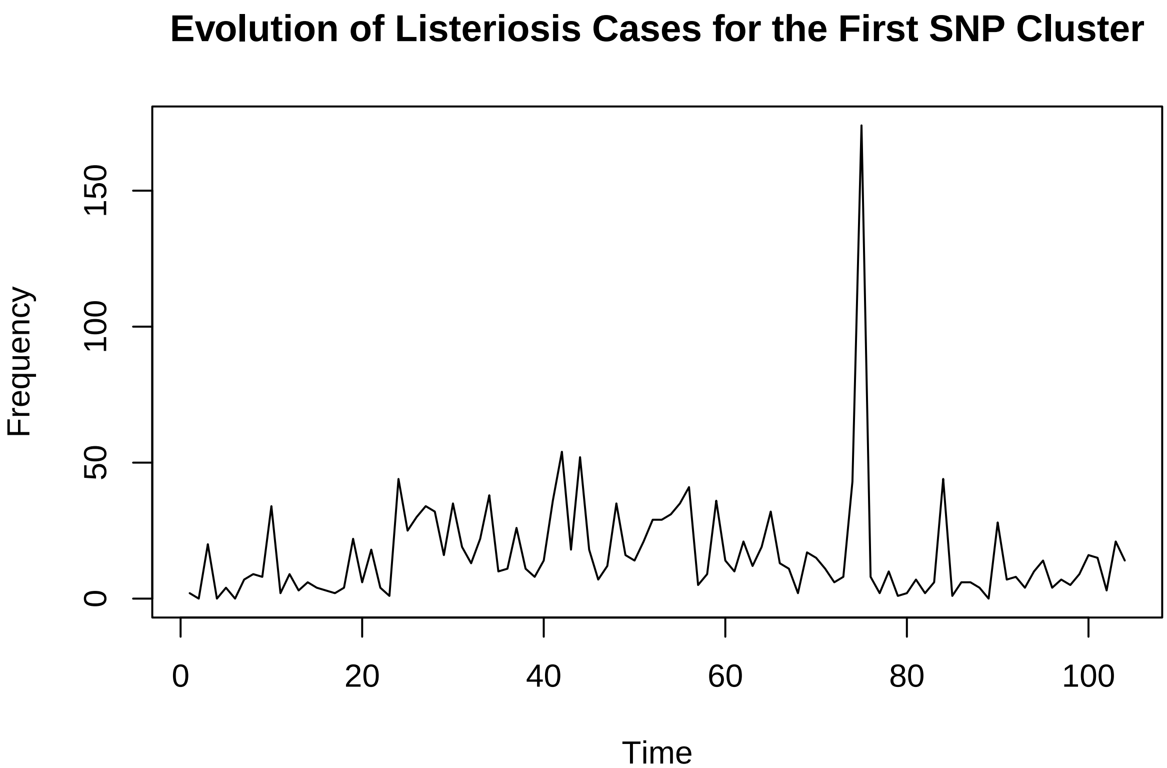
```
##       Date Frequency
## 1  2013-11         2
## 2  2013-12         0
## 3  2014-01        20
## 4  2014-02         0
## 5  2014-03         4
## 6  2014-04         0
## 7  2014-05         7
## 8  2014-06         9
## 9  2014-07         8
```

```
## 10 2014-08        34
```

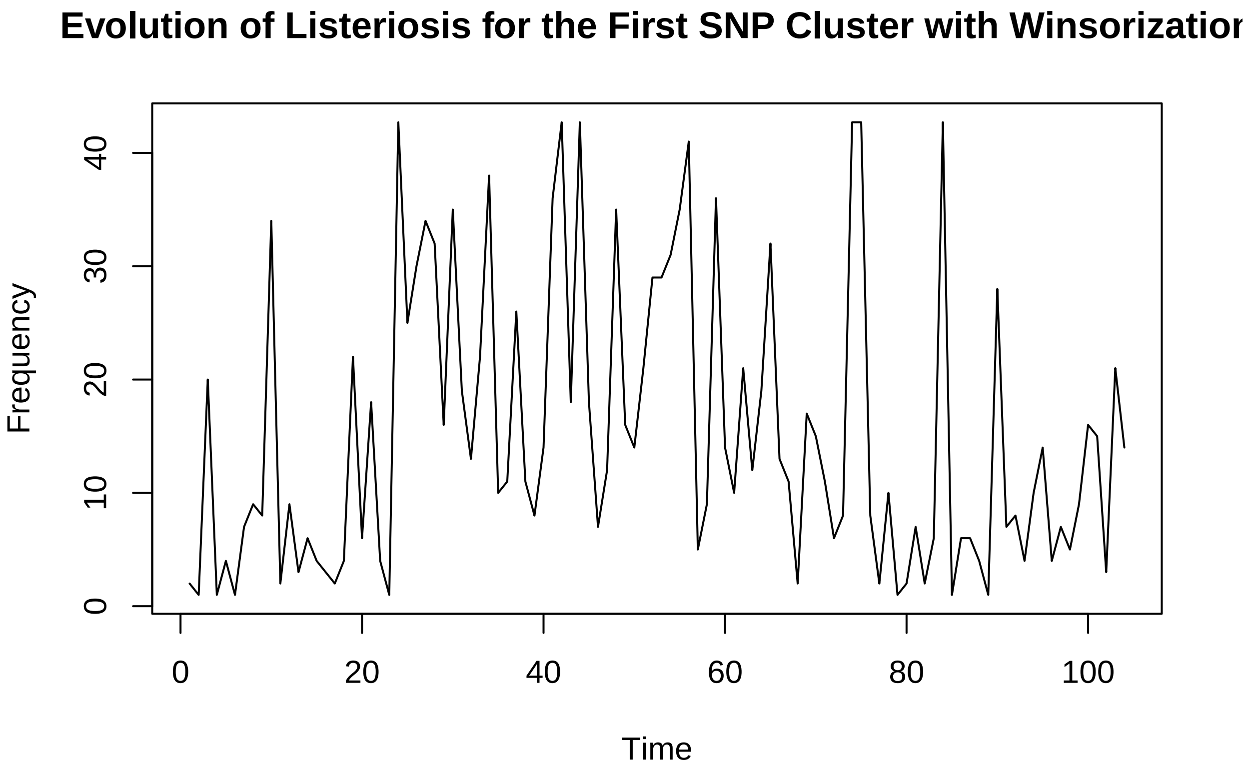
Above table (only 10 rows are showed) records the number of cases of Listeriosis happened in each month from 2013-11 to 2022-06. Months like 2013-12, 2014-02, 2014-04, and 2021-03 without any instances have a 0 value for frequency.

```
## Time Series:
## Start = 1
## End = 104
## Frequency = 1
##   [1]   2   0  20   0   4   0   7   9   8  34   2   9   3   6   4   3   2   4
##  [19]  22   6  18   4   1  44  25  30  34  32  16  35  19  13  22  38  10  11
##  [37]  26  11   8  14  36  54  18  52  18   7  12  35  16  14  21  29  29  31
##  [55]  35  41   5   9  36  14  10  21  12  19  32  13  11   2  17  15  11   6
##  [73]   8  43 174   8   2  10   1   2   7   2   6  44   1   6   6   4   0  28
##  [91]   7   8   4  10  14   4   7   5   9  16  15   3  21  14
```

Then, we convert frequency data in above table to an univariate time series.



Above plot visualizes the evolution of Listeriosis cases for the first SNP cluster(PDS000000366.488). We can clearly see an outlier around time point 75, so that we are going to perform winsorization to alleviate outlier's effect on our time series models.



Above plot visualizes the same trend but with winsorization performed on the time series.

```
##    February March April May June July August September October November
## 1         0     0     0   0    0    0      0         0       0        1
## 2         0     0     0   0    0    0      0         0       0        0
## 3         0     0     0   0    0    0      0         0       0        0
## 4         1     0     0   0    0    0      0         0       0        0
## 5         0     1     0   0    0    0      0         0       0        0
## 6         0     0     1   0    0    0      0         0       0        0
## 7         0     0     0   1    0    0      0         0       0        0
## 8         0     0     0   0    1    0      0         0       0        0
## 9         0     0     0   0    0    1      0         0       0        0
## 10        0     0     0   0    0    0      1         0       0        0
##    December
## 1         0
## 2         1
## 3         0
## 4         0
## 5         0
## 6         0
## 7         0
## 8         0
## 9         0
## 10        0
```

Above matrix (only 10 rows are showed) indicates each observation is from which month. If an observation has 0s for all the columns, then it is from January. This matrix represents all the external regressors (seasonality) we are going to input to the time series models.

ARIMA:

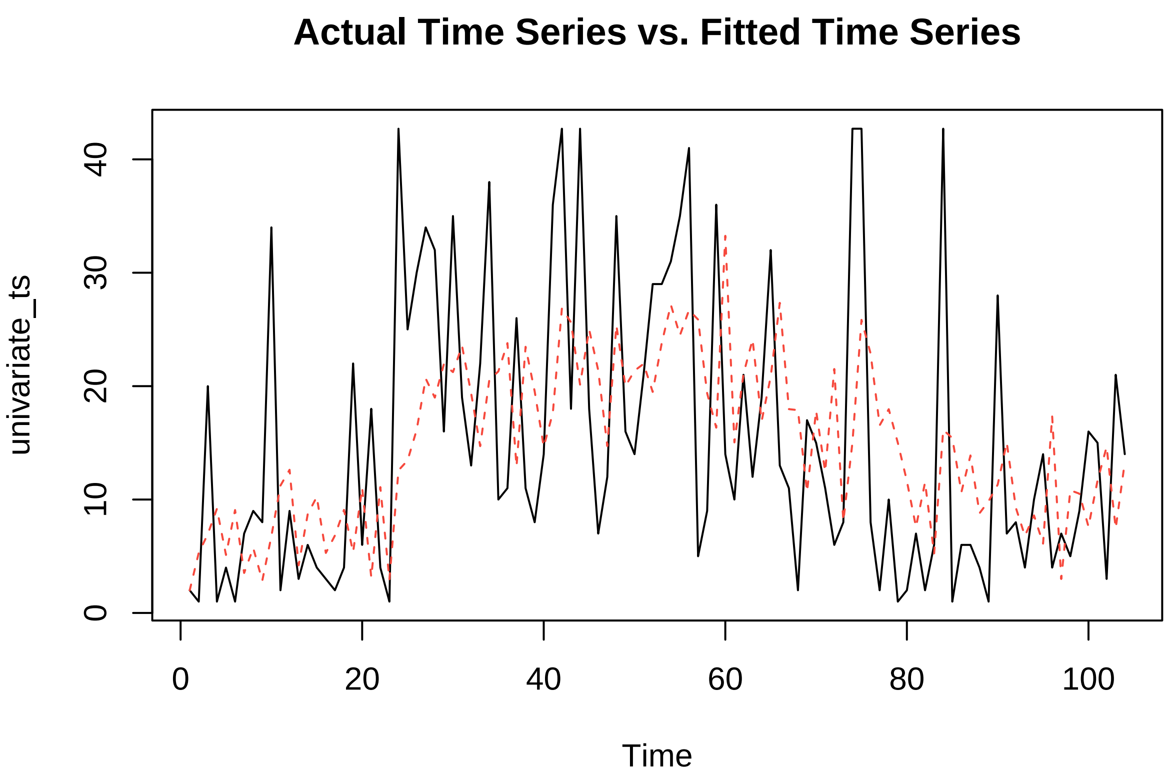
We were going to use ARIMA model first to model the evolution of cases for Listeriosis. The ARIMA model consists of three parts: AR, I, and MA. The "AR" stands for autoregression, where we predict something based on past values of that same thing. The "I" stands for integrated, and it represents the differencing in time series. The "MA" stands for moving average, where the next observation is the mean of every past observation. In order to account for seasonality, we supplied monthly information for each observation in the xreg matrix. We performed nested cross validation to choose the best hyperparameters (p, d, and q) for the ARIMA model. We used exhaustive grid search algorithm to perform nested cross validation, where we created total 216 models between different combinations of p, d, and q. The possible values for p, d, and q are 0, 1, 2, 3, 4, and 5 (total 216 models being evaluated).

```
## [1] 22.21181
```

The best model with the lowest root-mean-square deviation (RMSD) is the model with p = 0, d = 1, and q = 2. The RMSD for this ARIMA model is 22.21 and it is calculated based on the test set. After we finished training the model, for all the observations in the test set, each time we made one-step forward prediction and compared it with the value in the test set. Then, we updated our model with this new test set observation(without re-estimating the coefficients) and made the next one-step forward prediction.

```
##        ma1        ma2   February      March      April        May       June
## -0.6889653 -0.1722320 -3.4419724 -2.2451469  1.1401353 -1.8485463 -2.2397674
##       July     August  September    October   November   December
## -5.7064539 -3.2508873 -6.6548814  2.3614807 -6.4797639 -2.2027550
```

Above values are the coefficients for the best ARIMA model chosen by nested cross validation using monthly dataset for the first SNP cluster (PDS000000366.488).



Above graph shows the evolution for the actual time series (the black line) and the fitted time series (the red line). The x-axis represent different time points, where each tick value represent a month. The y-axis represents total cases for that particular month.

GARIMA:

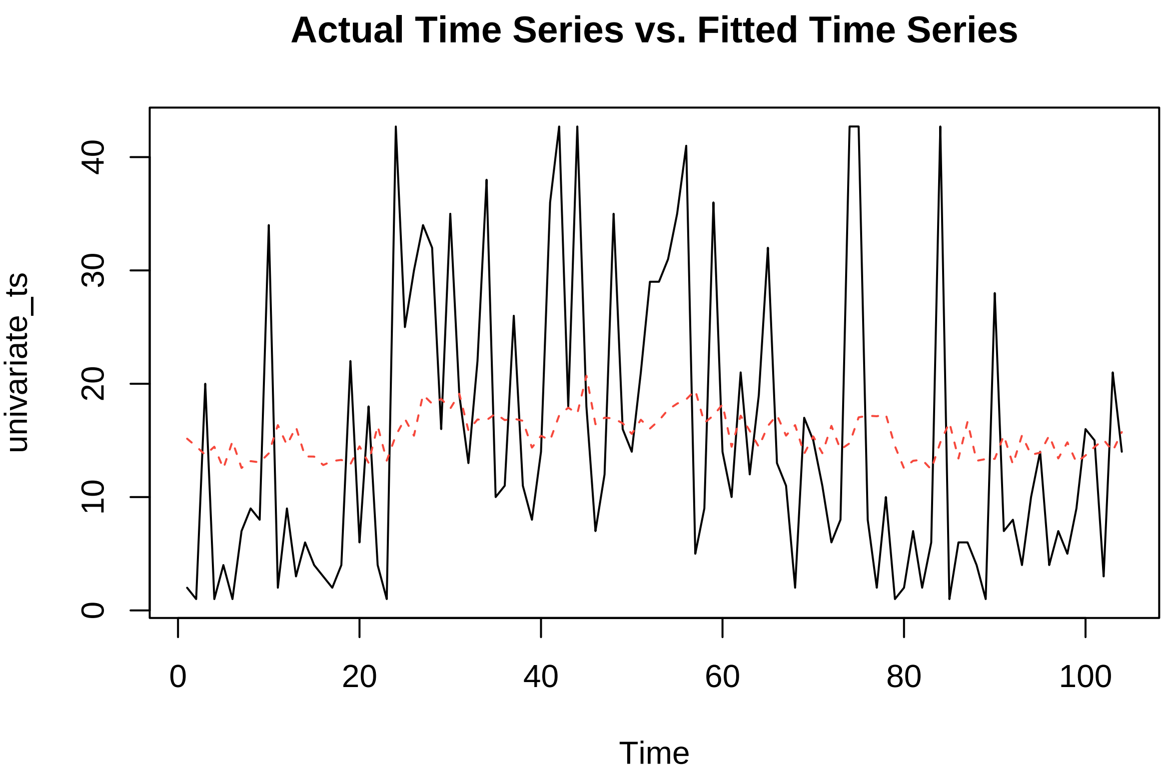
Next, we are going to use generalized ARIMA(GARIMA) model to model the evolution of cases for Listeriosis. GARIMA doesn't have the assumption that all the observations need to come from the normal distribution, so that it is more flexible. Here, we specify that all the observations are coming from the negative binomial distribution. Xreg matrix and nested cross validation procedure are the same compared to the ARIMA model. We use xreg matrix to account for seasonality and nested cross validation to select best values for hyperparameters. The best GARIMA model is chosen using the lowest RMSD, which is calculated the same way described earlier in the ARIMA section (one step forward prediction is made and compared to the test set then add to the original model without re-estimating the coefficients).

```
## [1] 20.50862
```

The GARIMA model with the lowest RMSD has p = 5 and q = 2 (The possible values for p and q are 0, 1, 2, 3, 4, and 5; total 36 models being evaluated). This is our final GARIMA model using the monthly dataset for the first SNP cluster.

```
##  (Intercept)       beta_1       beta_2       beta_3       beta_4       beta_5
## 1.162981e+01 1.008412e-01 1.534862e-07 8.960506e-02 1.064954e-02 5.701031e-11
##      alpha_1      alpha_2     February        March        April          May
## 2.676415e-11 3.145270e-02 1.272345e-04 2.519806e-01 6.244795e-01 1.407905e-01
##         June         July       August    September      October     November
## 8.998533e-05 6.265522e-05 3.695883e-01 1.148869e-04 1.515120e+00 3.480620e-06
##     December
## 6.512570e-01
```

Above are the coefficients for the final GARIMA model.



Above graph shows the evolution for the actual time series (the black line) and the fitted time series (output from the final GARIMA model). The x-axis represent different time points, where each tick value represent a month. The y-axis represents total cases for that particular month.

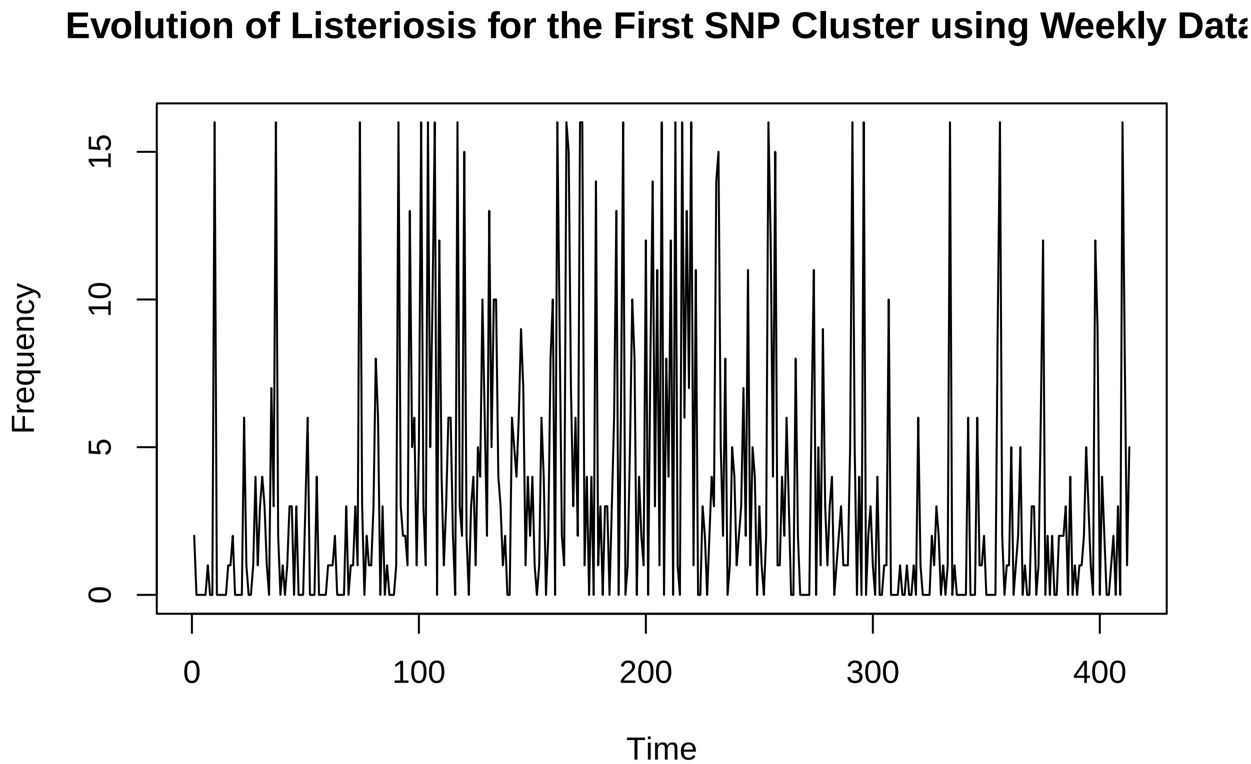
Using Weekly Dataset

```
##    Date(YMW) Frequency
## 1  2013-11-3         2
## 2  2013-11-4         0
## 3  2013-12-1         0
## 4  2013-12-2         0
## 5  2013-12-3         0
## 6  2013-12-4         0
## 7  2014-01-1         1
## 8  2014-01-2         0
## 9  2014-01-3         0
## 10 2014-01-4        19
```

After analyzing the first cluster using monthly data, we are going to use weekly data to compared the results. For each month, we coded date 1 to date 7 as the first week; date 8 to date 14 as the second week; date 15 to date 21 as the third week; and the rest of the days within each month as the fourth week. Above table shows the first 10 observations after we converted monthly data to weekly data. Each row of the column 'Frequency' now represents the total cases of Listeriosis occurred in that week.

```
## Time Series:
## Start = 1
## End = 413
## Frequency = 1
##   [1]  2  0  0  0  0  0  1  0  0 16  0  0  0  0  0  1  1  2  0  0  0  0  6  1  0
##  [26]  0  1  4  1  3  4  3  1  0  7  3 16  2  0  1  0  1  3  3  0  3  0  0  0  3
##  [51]  6  0  0  0  4  0  0  0  0  1  1  1  2  0  0  0  0  3  0  1  1  3  1 16  3
##  [76]  0  2  1  1  3  8  6  0  3  0  1  0  0  0  1 16  3  2  2  1 13  5  6  1  5
## [101] 16  3  1 16  5 10 16  0 12  4  1  3  6  6  2  0 16  3  2 15  2  0  3  4  1
## [126]  5  4 10  6  2 13  5 10 10  4  3  1  2  0  0  6  5  4  6  9  7  1  4  2  4
## [151]  1  0  1  6  4  0  2  8 10  0 16  9  2  1 16 15  7  3  6  2 16 16  1  4  0
## [176]  4  0 14  1  3  0  3  3  0  3  6 13  0  6 16  0  1  5 10  8  0  4  2  1 12
## [201]  0  8 14  3 11  1 16  0  8  4 12  0 16  1  0 16  6 13  7 16  1 11  0  0  3
## [226]  2  0  2  4  3 14 15  5  2  8  0  1  5  4  1  2  3  7  2 11  1  5  4  0  3
## [251]  1  0  2 16 12  4 15  1  1  4  2  6  3  0  0  8  2  0  0  0  0  0  6 11  0
## [276]  5  1  9  3  1  3  4  0  1  2  3  1  1  1  5 16  5  0  4  0 16  0  2  3  1
## [301]  0  4  0  0  1  1 10  0  0  0  0  1  0  0  1  0  0  1  0  6  1  0  0  0  0
## [326]  2  1  3  2  0  1  0  1 16  0  1  0  0  0  0  0  6  0  0  0  6  1  1  2  0
## [351]  0  0  0  0  9 16  2  0  1  1  5  0  1  2  5  0  1  0  0  3  3  0  1  6 12
## [376]  0  2  0  2  0  0  2  2  2  3  0  4  0  1  0  1  1  2  5  3  1  0 12  9  0
## [401]  4  2  0  0  1  2  0  3  0 16  8  1  5
```

Then, we converted weekly data of Listeriosis cases to an univariate time series. We also performed winsorization to alleviate the effect of outliers.



Above plot visualizes the evolution of Listeriosis cases for the first SNP cluster(PDS000000366.488) using weekly data (after winsorization).

```
##    February March April May June July August September October November
## 1         0     0     0   0    0    0      0         0       0        1
## 2         0     0     0   0    0    0      0         0       0        1
## 3         0     0     0   0    0    0      0         0       0        0
## 4         0     0     0   0    0    0      0         0       0        0
## 5         0     0     0   0    0    0      0         0       0        0
## 6         0     0     0   0    0    0      0         0       0        0
## 7         0     0     0   0    0    0      0         0       0        0
## 8         0     0     0   0    0    0      0         0       0        0
## 9         0     0     0   0    0    0      0         0       0        0
## 10        0     0     0   0    0    0      0         0       0        0
##    December
## 1         0
## 2         0
## 3         1
## 4         1
## 5         1
## 6         1
## 7         0
## 8         0
## 9         0
## 10        0
```

Above matrix (only ten rows are showed) indicates each observation is from which month. If an observation has 0s for all the columns, then it is from January. This matrix represents all the external regressors (seasonality) we are going to input to the time series models.

ARIMA

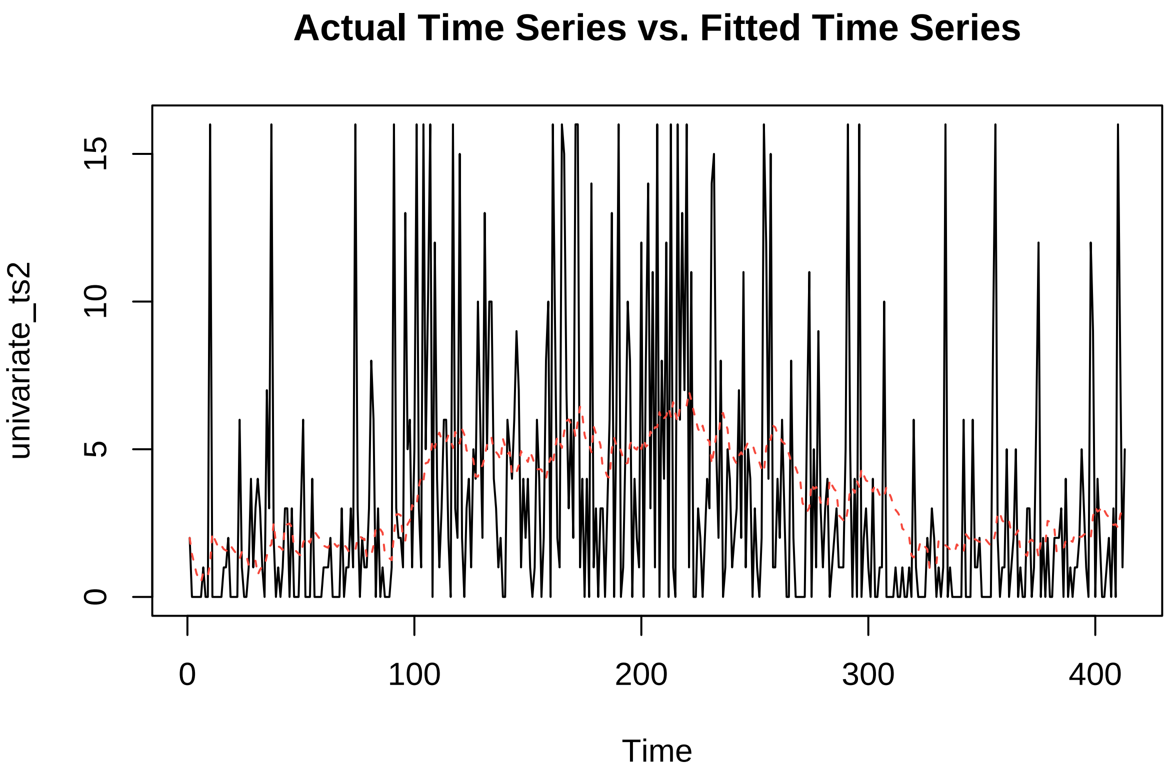
In the ARIMA model building part using weekly dataset for the first SNP cluster, we followed the same logic described earlier. We still used nest-cross validation with grid search, xreg matrix and RMSD to choose the model with the best performance.

```
## [1] 9.950878
```

The model with the lowest RMSD(9.969139) has hyperparameters: p = 0, d = 1, and q = 1 (The possible values for p, d, and q are 0, 1, 2, 3, 4, and 5; total 216 models being evaluated).

```
##          ma1     February        March        April          May         June
## -0.950321305 -0.071673928  0.167756193  0.352520401  0.253517761 -0.007618597
##         July       August    September      October     November     December
## -0.581441533 -0.164512460 -0.831151424  0.038402485 -0.729740595 -0.375054940
```

Above table shows the coefficients for the best-performing ARIMA model.



Above graph shows the evolution for the actual time series (the black line) and the fitted time series (output from the final ARIMA model). The x-axis represent different time points, where each tick value represent a week. The y-axis represents total cases for that particular week.

GARIMA

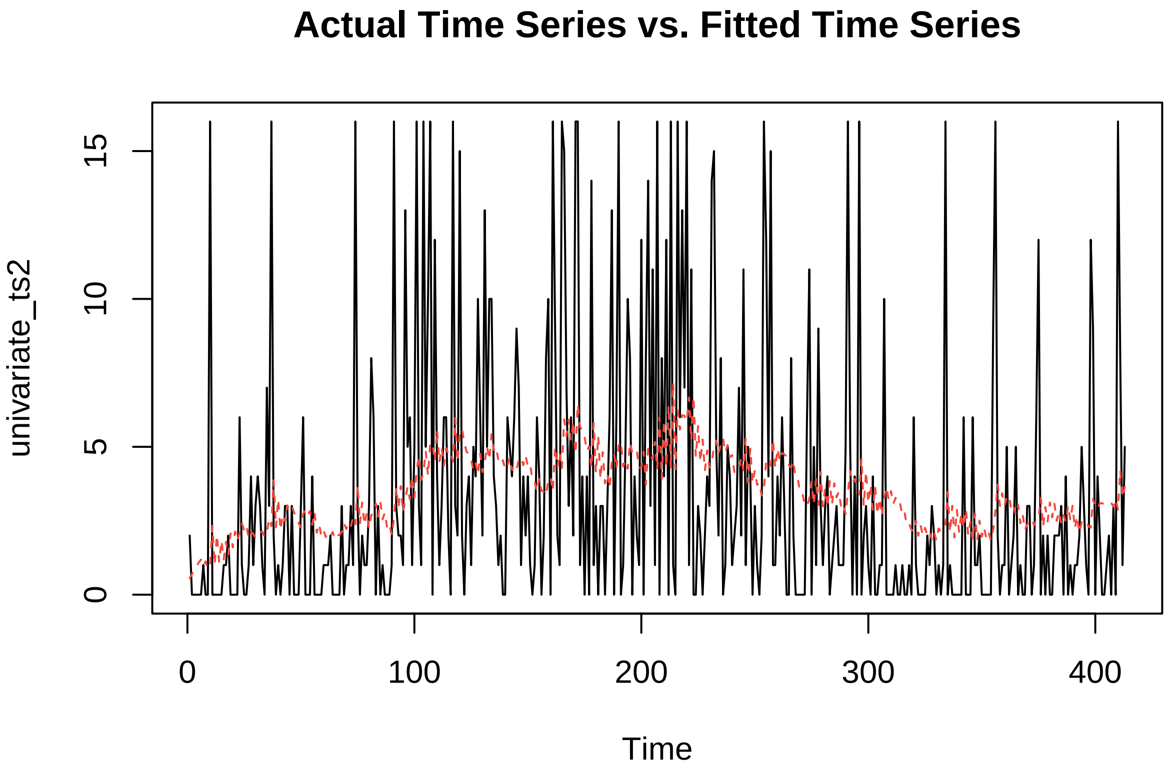
In the GARIMA model building part using weekly dataset for the first SNP cluster, we followed the same logic described earlier. We still used nest-cross validation with grid search, xreg matrix and RMSD to choose the model with the best performance.

```
## [1] 9.862185
```

The model with the lowest RMSD(9.848444) has hyperparameters: p = 1 and q = 3 (The possible values for p and q are 0, 1, 2, 3, 4, and 5; total 36 models being evaluated).

```
##   (Intercept)        beta_1       alpha_1       alpha_2       alpha_3      February 
## 2.745788e-02 8.957010e-02 3.071009e-02 7.797420e-01 5.084527e-02 1.004654e-01 
##        March        April          May         June         July       August 
## 1.447163e-01 3.748638e-01 2.590803e-01 2.507346e-13 3.764117e-02 2.209470e-01 
##    September      October     November     December 
## 9.976098e-07 4.094288e-01 2.677224e-04 3.369421e-01
```

Above table shows the coefficients for the best-performing GARIMA model.



Above graph shows the evolution for the actual time series (the black line) and the fitted time series (output from the final GARIMA model). The x-axis represent different time points, where each tick value represent a week. The y-axis represents total cases for that particular week.

Time Series Analysis Using the Full Dataset

Using Monthly Version Dataset

```
##       Date Frequency
## 1  2013-11       275
## 2  2013-12        83
## 3  2014-01       213
## 4  2014-02        66
## 5  2014-03       158
## 6  2014-04        83
## 7  2014-05       106
## 8  2014-06       196
## 9  2014-07       333
## 10 2014-08       237
```

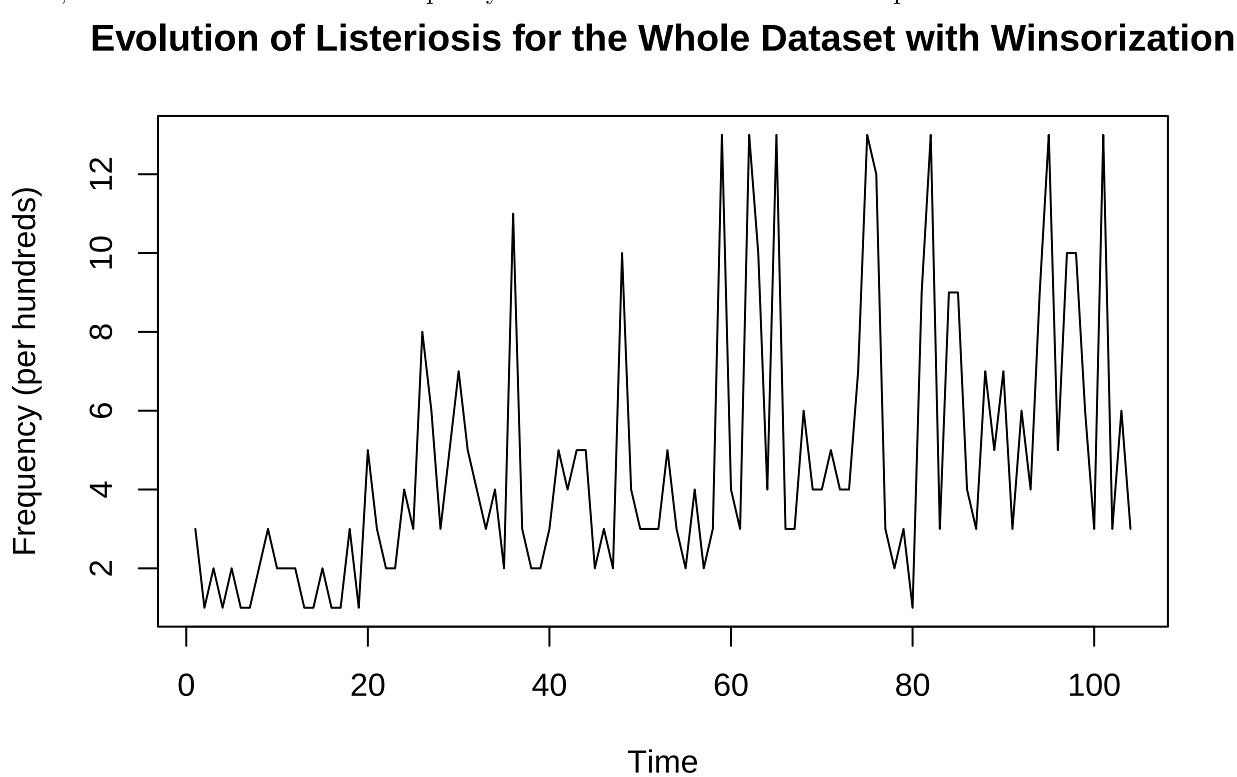
After analyzing the first SNP cluster, we are going to build time series models on the monthly version of the full dataset to try to model evolution of Listeriosis cases over time. Above table shows the first 10 rows of the monthly version dataset (contains all the SNP clusters, not just the first one). Column 'Frequency' indicates total cases of Listeriosis occurred in that month.

```
##       Date Frequency Month
## 1  2013-11         3    11
## 2  2013-12         1    12
## 3  2014-01         2    01
## 4  2014-02         1    02
## 5  2014-03         2    03
## 6  2014-04         1    04
## 7  2014-05         1    05
## 8  2014-06         2    06
## 9  2014-07         3    07
## 10 2014-08         2    08
```

For the column 'Frequency', we converted the unit to hundreds. For example, for date 2013-11, the frequency is 3, which means that there were approximately a total of 300 cases of Listeriosis happened in that month. The reason we converted the unit to hundreds is to rescale the data so that models are able to converge.

```
## Time Series:
## Start = 1
## End = 104
## Frequency = 1
##   [1]  3  1  2  1  2  1  1  2  3  2  2  2  1  1  2  1  1  3  1  5  3  2  2  4  3
##  [26]  8  6  3  5  7  5  4  3  4  2 11  3  2  2  3  5  4  5  5  2  3  2 10  4  3
##  [51]  3  3  5  3  2  4  2  3 13  4  3 13 10  4 13  3  3  6  4  4  5  4  4  7 13
##  [76] 12  3  2  3  1  9 13  3  9  9  4  3  7  5  7  3  6  4  9 13  5 10 10  6  3
## [101] 13  3  6  3
```

Then, we converted the column 'Frequency' to an univariate time series and performed winsorization.



Above plot visualizes the evolution of Listeriosis cases for the monthly version full dataset after winsorization (including all SNP clusters).

```
##    February March April May June July August September October November
## 1         0     0     0   0    0    0      0         0       0        1
## 2         0     0     0   0    0    0      0         0       0        0
## 3         0     0     0   0    0    0      0         0       0        0
## 4         1     0     0   0    0    0      0         0       0        0
## 5         0     1     0   0    0    0      0         0       0        0
## 6         0     0     1   0    0    0      0         0       0        0
## 7         0     0     0   1    0    0      0         0       0        0
## 8         0     0     0   0    1    0      0         0       0        0
## 9         0     0     0   0    0    1      0         0       0        0
## 10        0     0     0   0    0    0      1         0       0        0
##    December
## 1         0
## 2         1
## 3         0
## 4         0
## 5         0
## 6         0
## 7         0
## 8         0
## 9         0
## 10        0
```

Above xreg matrix (only 10 rows are showed) indicates each observation is from which month. If an observation has 0s for all the columns, then it is from January. This matrix represents all the external regressors (seasonality) we are going to input to the time series models.

ARIMA

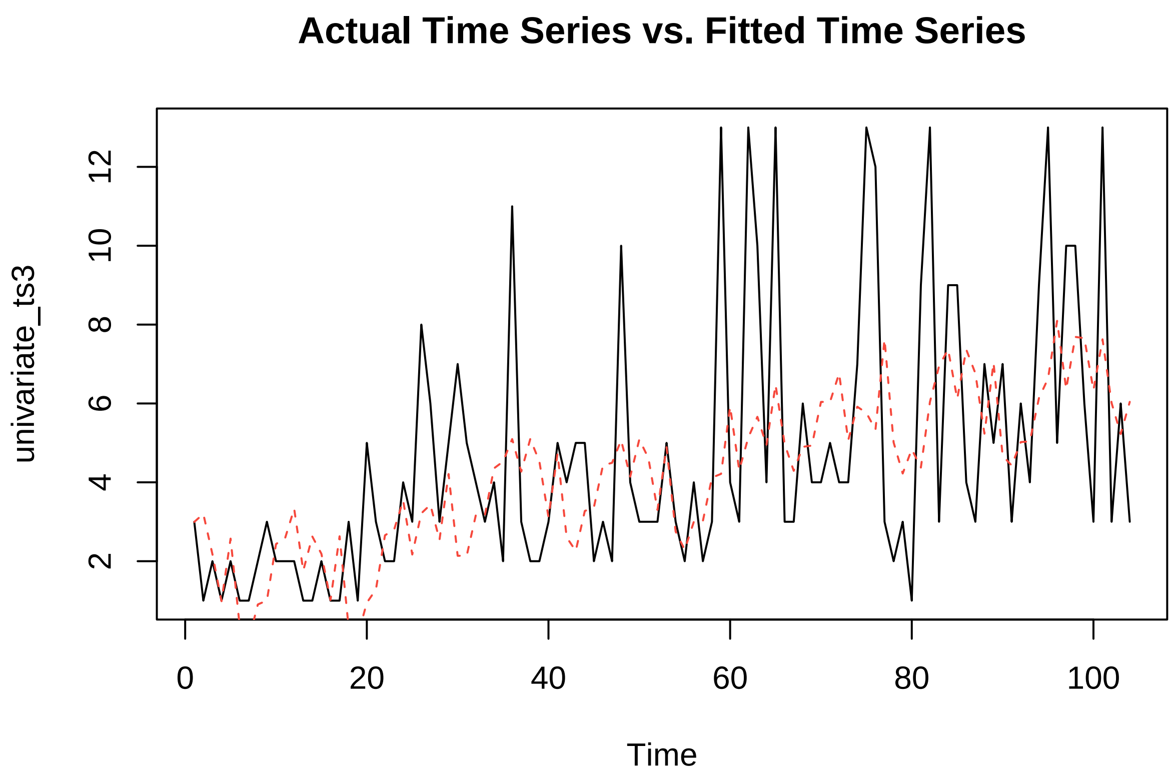
The first model we are trying to build using the monthly version full dataset is the ARIMA model. The procedure is the same. We use nested cross validation with grid search, xreg matrix and RMSD to choose the best-performing ARIMA model.

```
## [1] 4.657219
```

The ARIMA model with the lowest RMSD has a value of 4.657219. The p, d, and q for that model are 0, 1, and 1 respectively (The possible values for p, d, and q are 0, 1, 2, 3, 4, and 5; total 216 models being evaluated).

```
##        ma1   February      March      April        May       June       July
## -0.8996291 -1.1520582  0.4721033 -1.6820977 -2.1666978 -1.4286490 -1.5076591
##     August  September    October   November   December
## -0.3066243 -0.1000976  0.7243348 -0.6941584  0.2644663
```

Above table shows the coefficients for the best-performing ARIMA model.



Above graph shows the evolution for the actual time series (the black line) and the fitted time series (output from the final ARIMA model). The x-axis represent different time points, where each tick value represents a month. The y-axis represents total cases for that particular month and the unit is in hundreds.

GARIMA

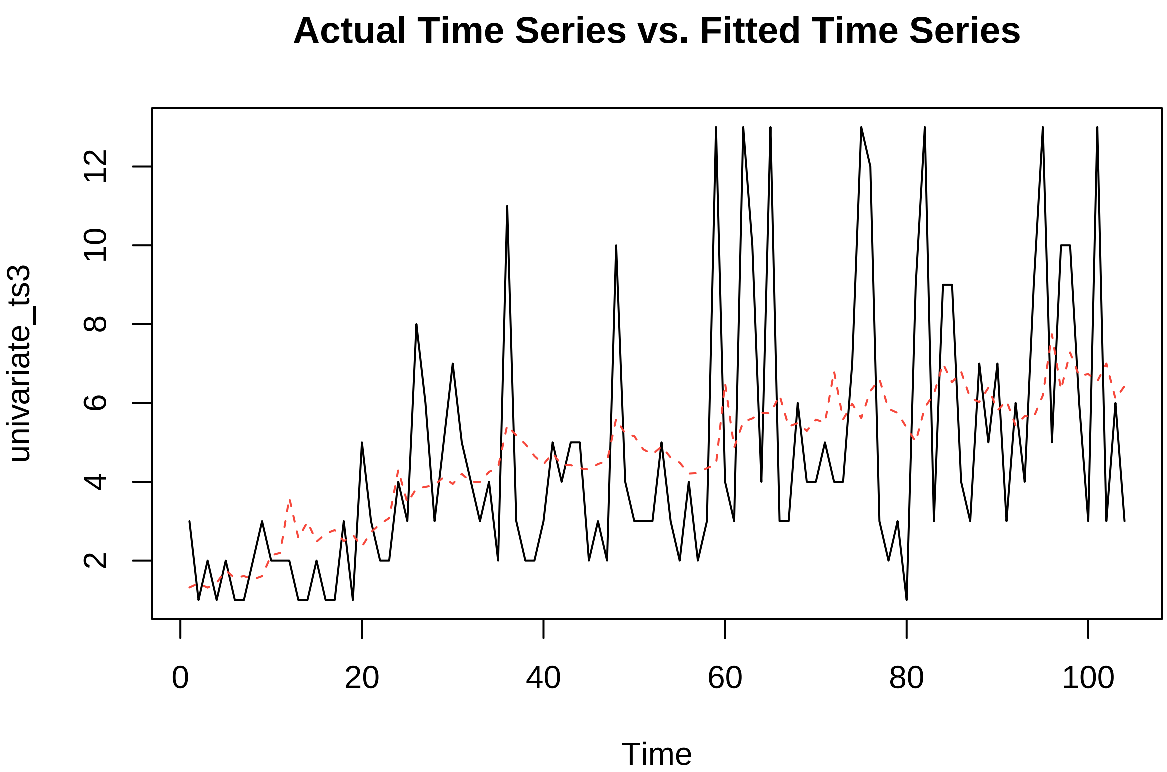
Then, we try to use the monthly version of the full dataset to build GARIMA models. The procedure is the same. We use nested cross validation with grid search, xreg matrix and RMSD to choose the best-performing GARIMA model.

```
## [1] 4.637046
```

The GARIMA model with the lowest RMSD has a value of 4.637046. The p and q for that model are 1 and 2 respectively (The possible values for p and q are 0, 1, 2, 3, 4, and 5; total 36 models being evaluated).

```
##  (Intercept)       beta_1      alpha_1      alpha_2     February        March
## 5.678213e-02 9.015660e-02 2.617928e-01 6.024720e-01 3.607860e-10 4.316484e-01
##        April          May         June         July       August    September
## 7.947635e-13 1.441102e-05 1.475997e-02 5.341853e-04 4.683964e-01 4.337228e-01
##      October     November     December
## 1.495199e+00 7.167494e-02 1.384464e-04
```

Above table shows the coefficients for the best-performing GARIMA model.



Above graph shows the evolution for the actual time series (the black line) and the fitted time series (output from the final GARIMA model). The x-axis represent different time points, where each tick value represents a month. The y-axis represents total cases for that particular month and the unit is in hundreds.

Using Weekly Version Dataset

```
##    Date(YMW) Frequency
## 1  2013-11-3       275
## 2  2013-11-4         0
## 3  2013-12-1         0
## 4  2013-12-2        52
## 5  2013-12-3        27
## 6  2013-12-4         4
## 7  2014-01-1       136
## 8  2014-01-2         8
## 9  2014-01-3        29
## 10 2014-01-4        40
```

Next, we are going to build ARIMA and GARIMA models again on the weekly version of the full dataset to try to model evolution of Listeriosis cases over time. Above table shows the first 10 rows of the weekly version dataset (contains all the SNP clusters, not just the first one). Column 'Date(YMW)' contains date information in year-month-week format. Column 'Frequency' indicates total cases of Listeriosis occurred in that week.

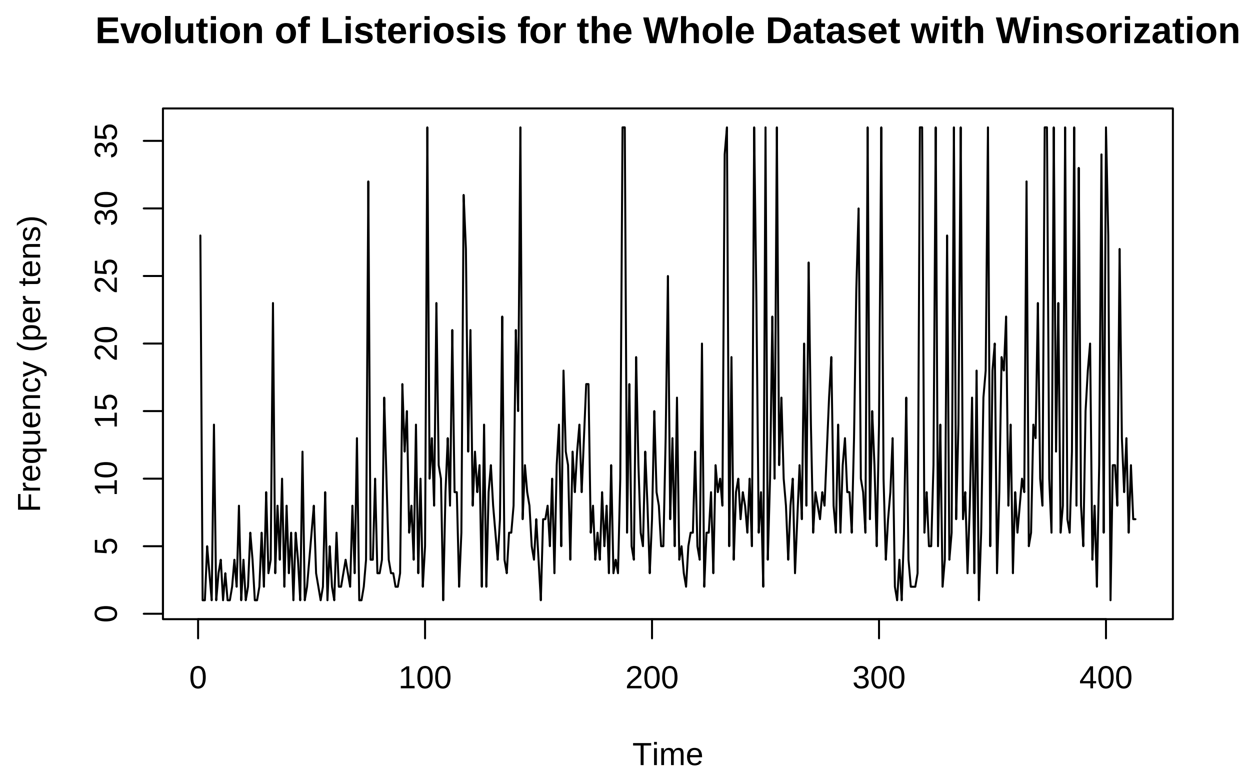
```
##    Date(YMW) Frequency Month
## 1  2013-11-3        28    11
## 2  2013-11-4         0    11
## 3  2013-12-1         0    12
## 4  2013-12-2         5    12
## 5  2013-12-3         3    12
```

```
## 6  2013-12-4         0    12
## 7  2014-01-1        14    01
## 8  2014-01-2         1    01
## 9  2014-01-3         3    01
## 10 2014-01-4         4    01
```

For the column ‘Frequency’, we converted the unit to tens. For example, for date 2013-11-3, the frequency is 28, which means that there were approximately a total of 280 cases of Listeriosis happened in that week. The reason we converted the unit to tens is to rescale the data so that models are able to converge.

```
## Time Series:
## Start = 1
## End = 413
## Frequency = 1
##   [1] 28  1  1  5  3  1 14  1  3  4  1  3  1  1  2  4  2  8  1  4  1  2  6  4  1
##  [26]  1  2  6  2  9  3  4 23  3  8  4 10  2  8  3  6  1  6  4  1 12  1  2  4  6
##  [51]  8  3  2  1  2  9  1  5  2  1  6  2  2  3  4  3  2  8  3 13  1  1  2  4 32
##  [76]  4  4 10  3  3  4 16 10  4  3  3  2  2  3 17 12 15  6  8  4 14  3 10  2  5
## [101] 36 10 13  8 23 11 10  1  9 13  8 21  9  9  2  6 31 27 12 21  8 12  9 11  2
## [126] 14  2  9 11  8  6  4  7 22  4  3  6  6  8 21 15 36  7 11  9  8  5  4  7  4
## [151]  1  7  7  8  5 10  3 11 14  5 18 12 11  4 12  9 12 14  9 13 17 17  6  8  4
## [176]  6  4  9  5  8  3 11  3  4  3 10 36 36  6 17  5  4 19 11  6  5 12  8  3  7
## [201] 15  9  8  5  5 13 25  7 13  5 16  4  5  3  2  5  6  6 12  5  4 20  2  6  6
## [226]  9  3 11  9 10  8 34 36  5 19  4  9 10  7  9  8  6 10  5 36 23  6  9  2 36
## [251]  4 10 22 10 36 11 16 10  8  4  8 10  3  7 11  7 20  8 26 14  6  9  8  7  9
## [276]  8 12 16 19  8  6 14  6 11 13  9  9  6 13 24 30 10  9  6 36  7 15 11  5 14
## [301] 36 10  4  7  9 13  2  1  4  1  6 16  4  2  2  2  3 36 36  6  9  5  5 11 36
## [326]  5 14  2  4 28  4  6 36  7 14 36  7  9  3  8 16  3 18  1  6 16 18 36  5 18
## [351] 20  3  9 19 18 22  8 14  3  9  6  8 10  9 32  5  6 14 13 23 10  8 36 36 10
## [376]  6 36 12 23  6  8 36  7  6 12 36  8 33  8  5 15 18 20  4  8  2 11 34  6 36
## [401] 28  1 11 11  8 27 13  9 13  6 11  7  7
```

Then, we converted the column ‘Frequency’ to an univariate time series and performed winsorization.



Above plot visualizes the evolution of Listeriosis cases for the weekly version full dataset after winsorization (including all SNP clusters).

```
##    February March April May June July August September October November
## 1         0     0     0   0    0    0      0         0       0        1
## 2         0     0     0   0    0    0      0         0       0        1
## 3         0     0     0   0    0    0      0         0       0        0
## 4         0     0     0   0    0    0      0         0       0        0
## 5         0     0     0   0    0    0      0         0       0        0
## 6         0     0     0   0    0    0      0         0       0        0
## 7         0     0     0   0    0    0      0         0       0        0
## 8         0     0     0   0    0    0      0         0       0        0
## 9         0     0     0   0    0    0      0         0       0        0
## 10        0     0     0   0    0    0      0         0       0        0
##    December
## 1         0
## 2         0
## 3         1
## 4         1
## 5         1
## 6         1
## 7         0
## 8         0
## 9         0
## 10        0
```

Above xreg matrix (only 10 rows are showed) indicates each observation is from which month. If an observation has 0s for all the columns, then it is from January. This matrix represents all the external regressors (seasonality) we are going to input to the time series models.

ARIMA

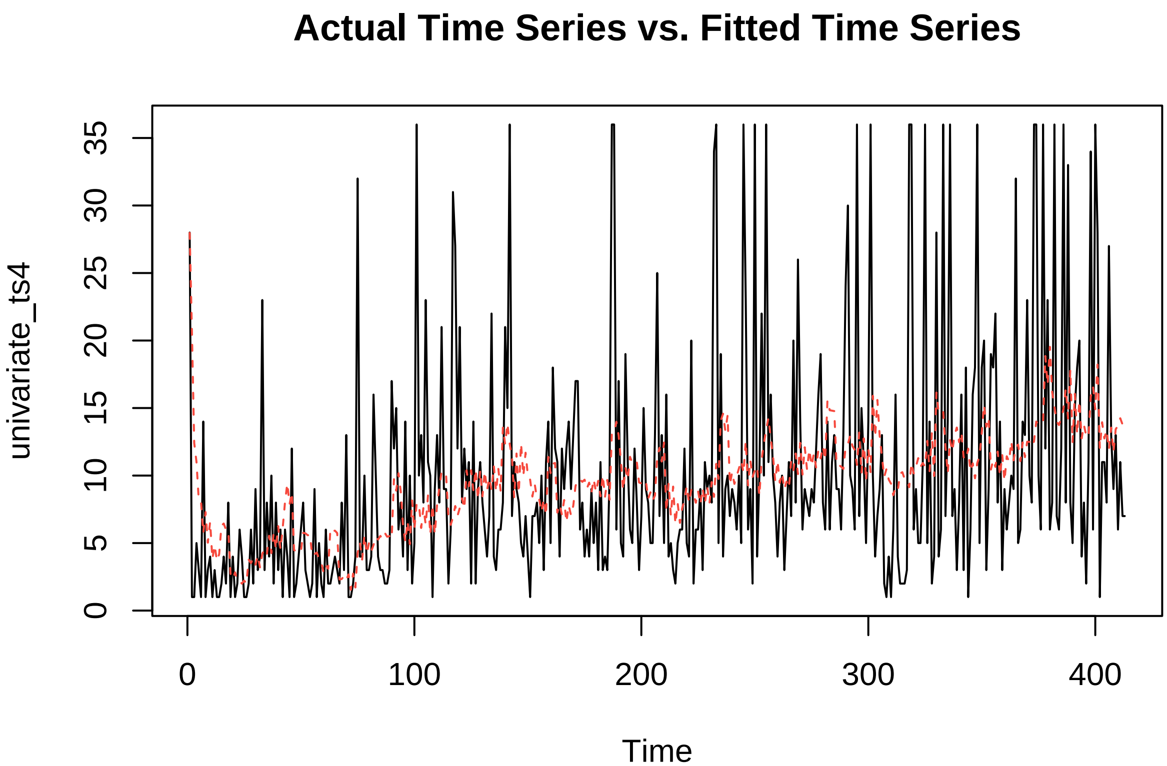
The first model we are trying to build using the weekly version of the full dataset is the ARIMA model. The procedure is the same. We use nested cross validation with grid search, xreg matrix and RMSD to choose the best-performing ARIMA model.

```
## [1] 21.99127
```

The ARIMA model with the lowest RMSD has a value of 21.99127. The p, d, and q for that model are 1, 1, and 3 respectively (The possible values for p, d, and q are 0, 1, 2, 3, 4, and 5; total 216 models being evaluated).

```
##          ar1          ma1          ma2          ma3     February        March
## -0.794514176 -0.147225812 -0.818039350  0.026954137 -0.930178438  2.001942054
##        April          May         June         July       August    September
## -1.385502829 -1.992229050 -0.396707824 -0.566206934 -0.214696591  0.156886329
##      October     November     December
##  3.767727034 -0.009010746  1.250290961
```

Above table shows the coefficients for the best-performing ARIMA model.



Above graph shows the evolution for the actual time series (the black line) and the fitted time series (output from the final ARIMA model). The x-axis represent different time points, where each tick value represents a week. The y-axis represents total cases for that particular week and the unit is in tens.

GARIMA

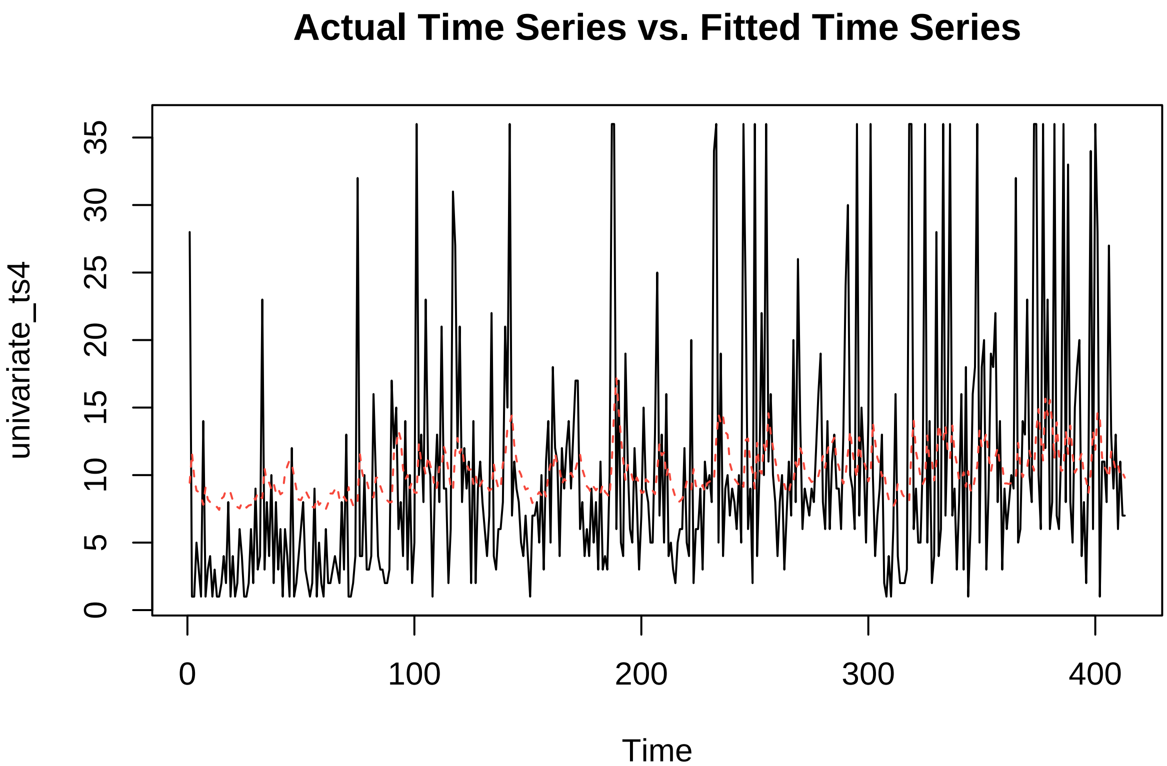
Then, we try to use the weekly version of the full dataset to build GARIMA models. The procedure is the same. We used nested cross validation with grid search, xreg matrix and RMSD to choose the best-performing GARIMA model.

```
## [1] 22.22489
```

The GARIMA model with the lowest RMSD has a value of 22.22489. The p and q for that model are 2 and 2 respectively (The possible values for p and q are 0, 1, 2, 3, 4, and 5; total 36 models being evaluated).

```
##  (Intercept)       beta_1       beta_2      alpha_1      alpha_2     February
## 4.267615e+00 1.196106e-01 5.122062e-02 1.173124e-01 2.594137e-01 8.498315e-03
##        March        April          May         June         July       August
## 9.529713e-01 1.154661e-03 8.246723e-12 3.585418e-01 1.458474e-01 2.331862e-01
##    September      October     November     December
## 2.302740e-01 2.174992e+00 2.679105e-03 2.291249e-01
```

Above table shows the coefficients for the best-performing GARIMA model.



Above graph shows the evolution for the actual time series (the black line) and the fitted time series (output from the final GARIMA model). The x-axis represent different time points, where each tick value represents a week. The y-axis represents total cases for that particular week and the unit is in tens.

Summary

```
##                    AIC       BIC      RMSD
## ARIMA(0,1,2)  822.7414  859.6276 11.338199
## GARIMA(5,2)        Inf       Inf 11.866818
## ARIMA(0,1,1) 2410.2531 2462.5264  4.351531
## GARIMA(1,3)  1931.7938 2000.1924  4.339950
## Arima(0,1,1)  540.7913  575.0427  2.907148
## GARIMA(1,2)   501.8040  544.1143  2.890416
## ARIMA(1,1,3) 2951.0196 3015.3560  8.323166
## GARIMA(2,2)  2733.2572 2801.6558  8.628902
```

Above table summarizes all the best-performing models chosen by the nested cross validation with grid search. ARIMA(0,1,2) and GARIMA(5,2) are built using the monthly version of the first SNP cluster dataset. ARIMA(0,1,1) and GARIMA(1,3) are built using the weekly version of the first SNP cluster dataset.

Arima(0,1,1) and GARIMA(1,2) are built using the monthly version of the full dataset (contains all the SNP clusters). Unit of the count data (frequency) is converted to hundreds. ARIMA(1,1,3) and GARIMA(2,2) are built using the weekly version of the full dataset (contains all the SNP clusters). Unit of the count data (frequency) is converted to tens.

Code Appendix:

```
knitr::opts_chunk$set(echo = TRUE)
library(naniar)
library(readr)
library(dplyr)
library(ggplot2)
library(gsarima)
library(forecast)
library(caret)
library(zoo)
library(astsa)
library(DescTools)
library(tscount)
setwd("~/Desktop")
isolates <- read_csv("isolates.csv")
dim(isolates)
sum(complete.cases(isolates$Outbreak))/nrow(isolates)
isolates$Outbreak = ifelse(is.na(isolates$Outbreak), 0, 1)
table(isolates$Outbreak)
apply(isolates, 2,  function(x) sum(complete.cases(x))/nrow(isolates))
isolates <- isolates %>%
    mutate(across(.cols=c(Location, Source_type, SNP_cluster), .fns = as.factor))
count_location = as.data.frame(table(isolates$Location))
colnames(count_location)[colnames(count_location) == "Var1"] <- "Location"
colnames(count_location)[colnames(count_location) == "Freq"] <- "Frequency"
count_location =count_location[order(-count_location$Frequency),]
# order by descending
# order() returns indices
count_location_10 = count_location[1:10,]
location_percentage = numeric(10)
for (i in 1:10){
  location_percentage[i] = count_location$Frequency[i]/sum(count_location$Frequency)
}
count_location_10['location_percentage'] <- location_percentage
ggplot(data = count_location_10, aes(x = reorder(Location, -Frequency),
                                 y = Frequency,
                                 label = scales::percent(location_percentage),
                                 fill = Location)) +
  geom_bar(stat = 'identity') +
  ggtitle('Top 10 Locations with the Highest Listeria Monocytogenes Cases') +
  geom_text(vjust = -0.3,
            size = 2) +
  labs(x = 'Location', y = 'Frequency') +
  theme(axis.text.x = element_text(angle=90, hjust=1, vjust=0.1)) +
  theme(legend.position="none")
count_source = as.data.frame(table(isolates$Source_type))
colnames(count_source)[colnames(count_source) == "Var1"] <- "Source"
```

```
colnames(count_source)[colnames(count_source) == "Freq"] <- "Frequency"
count_source =count_source[order(-count_source$Frequency),]

count_source_10 = count_source[1:10,]
source_percentage = numeric(10)
for (i in 1:10){
  source_percentage[i] = count_source$Frequency[i]/sum(count_source$Frequency)
}
count_source_10['source_percentage'] <- source_percentage
ggplot(data = count_source_10, aes(x = reorder(Source, -Frequency),
                                   y = Frequency,
                                   label = scales::percent(source_percentage),
                                   fill = Source)) +
  geom_bar(stat = 'identity') +
  ggtitle('Top 10 Categories of Isolate Origin that Caused Listeria Monocytogenes') +
  geom_text(vjust = -0.2,
            size = 2) +
  labs(x = 'Source Origin', y = 'Frequency') +
  theme(axis.text.x = element_text(angle=90, hjust=1, vjust=0.1)) +
  theme(legend.position="none")
unique_cluster = unique(isolates$SNP_cluster)
length(unique_cluster)
count_SNP = as.data.frame(table(isolates$SNP_cluster))
colnames(count_SNP)[colnames(count_SNP) == "Var1"] <- "SNP_cluster"
colnames(count_SNP)[colnames(count_SNP) == "Freq"] <- "Frequency"
count_SNP =count_SNP[order(-count_SNP$Frequency),]

count_SNP_10 = count_SNP[1:10,]
SNP_percentage = numeric(10)
for (i in 1:10){
  SNP_percentage[i] = count_SNP$Frequency[i]/sum(count_SNP$Frequency)
}
count_SNP_10['SNP_percentage'] <- SNP_percentage
count_SNP_10 = count_SNP[1:10,]
SNP_percentage = numeric(10)
for (i in 1:10){
  SNP_percentage[i] = count_SNP$Frequency[i]/sum(count_SNP$Frequency)
}
count_SNP_10['SNP_percentage'] <- SNP_percentage
sum(count_SNP_10$SNP_percentage)
ggplot(data = count_SNP_10, aes(x = reorder(SNP_cluster, -Frequency),
                                   y = Frequency,
                                   label = scales::percent(SNP_percentage),
                                   fill = SNP_cluster)) +
  geom_bar(stat = 'identity') +
  ggtitle('Top 10 SNP Clusters for the Listeria Monocytogenes') +
  geom_text(vjust = -0.2,
            size = 2) +
  labs(x = 'SNP Cluster', y = 'Frequency') +
  theme(axis.text.x = element_text(angle=90, hjust=1, vjust=0.1)) +
  theme(legend.position="none")
count_SNP = as.data.frame(table(isolates$SNP_cluster))
colnames(count_SNP)[colnames(count_SNP) == "Var1"] <- "SNP_cluster"
```

```r
colnames(count_SNP)[colnames(count_SNP) == "Freq"] <- "Frequency"
count_SNP =count_SNP[order(-count_SNP$Frequency),]

count_SNP_10 = count_SNP[1:10,]
SNP_percentage = numeric(10)
for (i in 1:10){
  SNP_percentage[i] = (count_SNP$Frequency[i]/sum(count_SNP$Frequency))*100
}
count_SNP_10['SNP_percentage'] <- SNP_percentage
SNP_vector1 = vector()

for (i in 1:11){
  name = sprintf("%s", count_SNP_10[i,1])
  SNP_vector1 = append(SNP_vector1, name)
}
filtered_isolates = isolates %>%
  filter((SNP_cluster %in% SNP_vector1))
dim(filtered_isolates)
partial_dataset = filtered_isolates %>%
  select(c(Location, Source_type, `Min-same`, `Min-diff`, SNP_cluster))
for (i in 1:10){
  new_cluster =  partial_dataset %>%
    filter((SNP_cluster == count_SNP_10[i,1]))
  print(sprintf("this is the summary table for SNP cluster %s", SNP_vector1[i]))
  print(summary(new_cluster))
}
count_SNP_20 = count_SNP[1:20,]
SNP_percentage = numeric(20)
for (i in 1:20){
  SNP_percentage[i] = (count_SNP$Frequency[i]/sum(count_SNP$Frequency))*100
}
count_SNP_20['SNP_percentage'] <- SNP_percentage
for (i in 1:10){
  new_cluster = isolates %>%
    filter((SNP_cluster == count_SNP_20[i,1]))

  new_cluster$Create_date_YM = format(as.Date(new_cluster$Create_date), "%Y-%m")

  count_date = as.data.frame(table(new_cluster$Create_date_YM))
  colnames(count_date)[colnames(count_date) == "Var1"] <- "Date"
  colnames(count_date)[colnames(count_date) == "Freq"] <- "Frequency"

  count_date$red = 0
  for (j in 1:dim(new_cluster)[1]){
    if(new_cluster$Outbreak[j] == 1){
      number = which(count_date$Date == noquote(new_cluster$Create_date_YM[j]))
      count_date$red[number] = 1
      }
    }


  cluster_name = count_SNP_20[i,1]
```

```
  print(ggplot(data = count_date) +
          geom_point (mapping = aes (x=Date, y=Frequency), color=ifelse(count_date$red == 1, "red", "bla
          scale_x_discrete(breaks = count_date$Date[seq(1, length(count_date$Date), by = 10)]) +
          ggtitle(paste("Listeria Monocytogenes Cases Evolution for SNP Cluster", cluster_name)) +
          labs(x = 'Date(year-month)', y = 'Frequency'))


  count_date = count_date[order(-count_date$Frequency),] # order returns indexes
  print(sprintf("SNP Cluster %s has the highest cases of listeria monocytogenes at %s", cluster_name, co
}
for (i in 1:10){
  new_cluster = isolates %>%
    filter((SNP_cluster == count_SNP_20[i,1]))
  new_cluster$Create_date = format(as.Date(new_cluster$Create_date), "%Y-%m-%d")
  new_cluster$Create_date_YM = format(as.Date(new_cluster$Create_date), "%Y-%m")

  for (j in 1:dim(new_cluster[1])){
  date = as.numeric(format(as.Date(new_cluster$Create_date[j]), "%d"))
  new_cluster$week[j] = if(date >= 1 && date <= 7){
    1
  } else if(date >= 8 && date <= 14){
    2
  } else if(date >= 15 && date <= 21){
    3
  } else {
    4
  }
  new_cluster$Create_date_YMW[j] = sprintf("%s-%s", new_cluster$Create_date_YM[j], new_cluster$week[j])
  }

  count_date = as.data.frame(table(new_cluster$Create_date_YMW))
  colnames(count_date)[colnames(count_date) == "Var1"] <- "Date"
  colnames(count_date)[colnames(count_date) == "Freq"] <- "Frequency"

  count_date$red = 0
  for (j in 1:dim(new_cluster)[1]){
    if(new_cluster$Outbreak[j] == 1){
      number = which(count_date$Date == noquote(new_cluster$Create_date_YMW[j]))
      count_date$red[number] = 1
      }
    }

  # print(table(count_date$red))

  cluster_name = count_SNP_20[i,1]

  print(ggplot(data = count_date) +
          geom_point (mapping = aes (x=Date, y=Frequency), color=ifelse(count_date$red == 1, "red", "bla
          scale_x_discrete(breaks = count_date$Date[seq(1, length(count_date$Date), by = 20)]) +
          ggtitle(paste("Listeria Monocytogenes Cases Evolution for SNP Cluster", cluster_name)) +
          labs(x = 'Date(year-month-week)', y = 'Frequency'))
```

```
  count_date = count_date[order(-count_date$Frequency),]  # order returns indexes
  print(sprintf("SNP Cluster %s has the highest cases of listeria monocytogenes at %s", cluster_name, co
}
cluster_1 = isolates %>%
  filter((SNP_cluster == count_SNP_20[1,1]))
cluster_1$Create_date_YM = format(as.Date(cluster_1$Create_date), "%Y-%m")
datatouse = as.data.frame(table(cluster_1$Create_date_YM))
colnames(datatouse)[colnames(datatouse) == "Var1"] <- "Date"
colnames(datatouse)[colnames(datatouse) == "Freq"] <- "Frequency"
datatouse$Date = as.character(datatouse$Date)
datatouse[nrow(datatouse)+1,] = c("2013-12", 0)
datatouse[nrow(datatouse)+1,] = c("2014-02", 0)
datatouse[nrow(datatouse)+1,] = c("2014-04", 0)
datatouse[nrow(datatouse)+1,] = c("2021-03", 0)
datatouse = datatouse[order(datatouse$Date),]
rownames(datatouse) <- NULL
datatouse$Frequency = as.numeric(datatouse$Frequency)
datatouse[1:10,]
univariate_ts = as.ts(datatouse$Frequency)
univariate_ts
ts.plot(univariate_ts, main = 'Evolution of Listeriosis Cases for the First SNP Cluster', ylab = 'Freque
univariate_ts = Winsorize(univariate_ts)
ts.plot(univariate_ts, main = 'Evolution of Listeriosis for the First SNP Cluster with Winsorization', y
create_xreg <- function(ds){
  xreg = model.matrix(~as.factor(ds$Month))
  xreg = xreg[,-1]
  colnames(xreg) = c("February", "March", "April", "May", "June", "July", "August", "September", "Octobe
  return(xreg)
}
datatouse$Month = format(as.Date(paste(datatouse$Date,"-01",sep="")), "%m")
xreg_touse = create_xreg(datatouse)
xreg_touse[1:10,]
df <- data.frame(matrix(0, nrow=216, ncol=4))
colnames(df) = c("p","d","q","RMSD")
p = c(0,1,2,3,4,5)
d = c(0,1,2,3,4,5)
q = c(0,1,2,3,4,5)


count1 = 1

for(a in p){
    for (b in d){
      for (c in q){
                df[count1,3] = c
                df[count1,2] = b
                df[count1,1] = a
                count1 = count1 + 1
      }
    }
}
for (i in 1:4){
  start = 1
```

```
  end = i * round(nrow(datatouse)/5)
  end2 = (i+1) * round(nrow(datatouse)/5)
  if (end2 > nrow(datatouse)){
    end2 = dim(datatouse)[1]
  }
  datatouse_full_train = datatouse[start:end,]
  datatouse_full_test = datatouse[(end+1):end2,]

  datatouse_full_train_f = datatouse_full_train$Frequency
  datatouse_full_test_t = datatouse_full_test$Frequency

  univariate_ts_train = as.ts(datatouse_full_train_f)


  xreg_test = create_xreg(datatouse_full_test)


  for (j in 1:nrow(df)){
    p = df[j,1]
    d = df[j,2]
    q = df[j,3]
    mse_list = c()

    xreg_train = create_xreg(datatouse_full_train)
    new_model = Arima(univariate_ts_train, order = c(p,d,q), method = "CSS", xreg = xreg_train)


    datatouse_full_train_f = datatouse_full_train$Frequency

    for (k in 1:length(datatouse_full_test_t)){
      predict_model = predict(new_model, n.ahead=1, newxreg = t(xreg_test[k,]))
      mse_list[k] = (predict_model$pred - datatouse_full_test_t[k])^2
      datatouse_full_train_f = append(datatouse_full_train_f, datatouse_full_test_t[k])
      new_data_ts = as.ts(datatouse_full_train_f)
      xreg_train = rbind(xreg_train, xreg_test[k,])
      new_model = Arima(new_data_ts, model=new_model, xreg = xreg_train)
    }

    mse_value = mean(mse_list)
    mse_value = sqrt(mse_value)  # RMSD
    df[j,4] = df[j,4] + mse_value
  }
}

df = df %>%
  mutate(RMSD = RMSD/4)
min(df$RMSD)
final_arima = Arima(univariate_ts, order = c(0,1,2), xreg = xreg_touse)
final_arima$coef
ts.plot(univariate_ts)
final_arima_fit = univariate_ts - residuals(final_arima)
points(final_arima_fit, type = "l", col=2, lty=2)
title(main = "Actual Time Series vs. Fitted Time Series")
```

```
create_gsarima <- function(ts,p,q,x){
  if (p==0 & q==0){
    new_gsarima = tsglm(ts, model = list(), distr = "nbinom", xreg = x)
  }
  else if (p==0){
    new_gsarima = tsglm(ts, model = list(past_mean = 1:q), distr = "nbinom", xreg = x)
  }
  else if (q==0){
    new_gsarima = tsglm(ts, model = list(past_obs = 1:p), distr = "nbinom", xreg = x)
  } else {
    new_gsarima = tsglm(ts, model = list(past_obs = 1:p, past_mean = 1:q), distr = "nbinom", xreg = x)
  }
  return(new_gsarima)
}
df2 <- data.frame(matrix(0, nrow=36, ncol=3))
colnames(df2) = c("p","q","RMSD")
pp = c(0,1,2,3,4,5)
qq = c(0,1,2,3,4,5)

count = 1

for (a in pp){
  for (b in qq){
    df2[count,1] = a
    df2[count,2] = b
    count = count + 1
  }
}
for (i in 1:4){
  start = 1
  end = i * round(nrow(datatouse)/5)
  end2 = (i+1) * round(nrow(datatouse)/5)
  if (end2 > nrow(datatouse)){
    end2 = dim(datatouse)[1]
  }
  datatouse_full_train = datatouse[start:end,]
  datatouse_full_test = datatouse[(end+1):end2,]

  univariate_ts_train = as.ts(datatouse_full_train$Frequency)

  xreg_train = create_xreg(datatouse_full_train)
  xreg_test = create_xreg(datatouse_full_test)


  for (i in 1:nrow(df2)){
    p = df2[i,1]
    q = df2[i,2]
    gsarima = create_gsarima(univariate_ts_train, p, q, xreg_train)
    predict_model = predict(gsarima, n.ahead = nrow(datatouse_full_test), newobs = datatouse_full_test$
    mse_value = mean((predict_model$pred - datatouse_full_test$Frequency)^2)
    mse_value = sqrt(mse_value)
    df2[i,3] = df2[i,3] + mse_value
  }
```

```
}

df2 = df2 %>%
  mutate(RMSD = RMSD/4)
min(df2$RMSD)
final_garima = create_gsarima(univariate_ts, 5, 2, xreg_touse)
final_garima$coefficients
ts.plot(univariate_ts)
final_garima_fit = univariate_ts - residuals(final_garima)
points(final_garima_fit, type = "l", col=2, lty=2)
title(main = "Actual Time Series vs. Fitted Time Series")
cluster_1$Create_date = format(as.Date(cluster_1$Create_date), "%Y-%m-%d")
cluster_1$Create_date_YM = format(as.Date(cluster_1$Create_date), "%Y-%m")
for (i in 1:dim(cluster_1[1])){
  date = as.numeric(format(as.Date(cluster_1$Create_date[i]), "%d"))
  cluster_1$week[i] = if(date >= 1 && date <= 7){
    1
  } else if(date >= 8 && date <= 14){
    2
  } else if(date >= 15 && date <= 21){
    3
  } else {
    4
  }
  cluster_1$Create_date_YMW[i] = sprintf("%s-%s", cluster_1$Create_date_YM[i], cluster_1$week[i])
}
datatouse2 = as.data.frame(table(cluster_1$Create_date_YMW))
colnames(datatouse2)[colnames(datatouse2) == "Var1"] <- "Date(YMW)"
colnames(datatouse2)[colnames(datatouse2) == "Freq"] <- "Frequency"
datatouse2$`Date(YMW)` = as.character(datatouse2$Date)
datatouse2[nrow(datatouse2)+1,] = c("2013-11-4", 0)
datatouse2[nrow(datatouse2)+1,] = c("2013-12-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2013-12-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2013-12-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2013-12-4", 0)
datatouse2[nrow(datatouse2)+1,] = c("2014-01-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2014-01-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2014-02-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2014-02-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2014-02-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2014-02-4", 0)
datatouse2[nrow(datatouse2)+1,] = c("2014-03-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2014-04-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2014-04-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2014-04-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2014-04-4", 0)
datatouse2[nrow(datatouse2)+1,] = c("2014-05-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2014-05-4", 0)
datatouse2[nrow(datatouse2)+1,] = c("2014-07-4", 0)
datatouse2[nrow(datatouse2)+1,] = c("2014-09-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2014-09-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2014-10-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2014-11-1", 0)
```

```
datatouse2[nrow(datatouse2)+1,] = c("2014-11-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2014-11-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2014-12-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2014-12-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2014-12-4", 0)
datatouse2[nrow(datatouse2)+1,] = c("2015-01-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2015-01-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2015-01-4", 0)
datatouse2[nrow(datatouse2)+1,] = c("2015-02-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2015-03-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2015-03-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2015-03-4", 0)
datatouse2[nrow(datatouse2)+1,] = c("2015-04-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2015-04-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2015-06-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2015-08-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2015-08-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2015-09-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2015-09-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2015-09-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2016-02-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2016-04-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2016-05-4", 0)
datatouse2[nrow(datatouse2)+1,] = c("2016-10-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2016-10-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2017-01-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2017-02-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2017-03-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2017-07-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2017-07-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2017-08-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2017-09-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2017-10-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2017-11-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2017-12-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2018-01-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2018-03-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2018-04-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2018-05-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2018-07-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2018-07-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2018-08-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2018-10-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2019-01-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2019-02-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2019-05-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2019-05-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2019-06-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2019-06-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2019-06-4", 0)
datatouse2[nrow(datatouse2)+1,] = c("2019-07-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2019-07-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2019-08-1", 0)
```

```
datatouse2[nrow(datatouse2)+1,] = c("2019-10-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2019-12-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-01-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-01-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-02-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-03-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-03-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-04-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-04-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-04-4", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-05-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-05-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-05-4", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-06-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-06-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-07-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-07-4", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-08-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-08-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-08-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-09-4", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-10-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-11-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-11-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-11-4", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-12-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-12-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2020-12-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2021-01-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2021-01-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2021-01-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2021-02-4", 0)
datatouse2[nrow(datatouse2)+1,] = c("2021-03-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2021-03-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2021-03-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2021-03-4", 0)
datatouse2[nrow(datatouse2)+1,] = c("2021-04-4", 0)
datatouse2[nrow(datatouse2)+1,] = c("2021-05-4", 0)
datatouse2[nrow(datatouse2)+1,] = c("2021-06-4", 0)
datatouse2[nrow(datatouse2)+1,] = c("2021-07-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2021-07-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2021-08-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2021-09-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2021-09-4", 0)
datatouse2[nrow(datatouse2)+1,] = c("2021-10-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2021-10-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2021-11-4", 0)
datatouse2[nrow(datatouse2)+1,] = c("2021-12-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2021-12-4", 0)
datatouse2[nrow(datatouse2)+1,] = c("2022-02-3", 0)
datatouse2[nrow(datatouse2)+1,] = c("2022-03-2", 0)
datatouse2[nrow(datatouse2)+1,] = c("2022-04-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2022-04-2", 0)
```

```
datatouse2[nrow(datatouse2)+1,] = c("2022-05-1", 0)
datatouse2[nrow(datatouse2)+1,] = c("2022-05-3", 0)


datatouse2 = datatouse2[order(datatouse2$Date),]
rownames(datatouse2) <- NULL
datatouse2$Frequency = as.numeric(datatouse2$Frequency)
datatouse2[1:10,]
univariate_ts2 = as.ts(datatouse2$Frequency)
univariate_ts2 = Winsorize(univariate_ts2)
univariate_ts2
ts.plot(univariate_ts2, main = 'Evolution of Listeriosis for the First SNP Cluster using Weekly Data', y
datatouse2$Month = format(as.Date(datatouse2$`Date(YMW)`), "%m")
xreg_touse2 = create_xreg(datatouse2)
xreg_touse2[1:10,]
df3 <- data.frame(matrix(0, nrow=216, ncol=4))
colnames(df3) = c("p","d","q","RMSD")
p = c(0,1,2,3,4,5)
d = c(0,1,2,3,4,5)
q = c(0,1,2,3,4,5)


count1 = 1

for(a in p){
    for (b in d){
      for (c in q){
                df3[count1,3] = c
                df3[count1,2] = b
                df3[count1,1] = a
                count1 = count1 + 1
      }
    }
}
for (i in 1:4){
  start = 1
  end = i * round(nrow(datatouse2)/5)
  end2 = (i+1) * round(nrow(datatouse2)/5)
  if (end2 > nrow(datatouse2)){
    end2 = dim(datatouse2)[1]
  }
  datatouse_full_train = datatouse2[start:end,]
  datatouse_full_test = datatouse2[(end+1):end2,]

  datatouse_full_train_f = datatouse_full_train$Frequency
  datatouse_full_test_t = datatouse_full_test$Frequency

  univariate_ts_train = as.ts(datatouse_full_train_f)


  xreg_test = create_xreg(datatouse_full_test)
```

```
  for (j in 1:nrow(df3)){
    p = df3[j,1]
    d = df3[j,2]
    q = df3[j,3]
    mse_list = c()

    xreg_train = create_xreg(datatouse_full_train)
    new_model = Arima(univariate_ts_train, order = c(p,d,q), method = "CSS", xreg = xreg_train)


    datatouse_full_train_f = datatouse_full_train$Frequency

    for (k in 1:length(datatouse_full_test_t)){
      predict_model = predict(new_model, n.ahead=1, newxreg = t(xreg_test[k,]))
      mse_list[k] = (predict_model$pred - datatouse_full_test_t[k])^2
      datatouse_full_train_f = append(datatouse_full_train_f, datatouse_full_test_t[k])
      new_data_ts = as.ts(datatouse_full_train_f)
      xreg_train = rbind(xreg_train, xreg_test[k,])
      new_model = Arima(new_data_ts, model=new_model, xreg = xreg_train)
    }

    mse_value = mean(mse_list)
    mse_value = sqrt(mse_value)  # RMSD
    df3[j,4] = df3[j,4] + mse_value
  }
}

df3 = df3 %>%
  mutate(RMSD = RMSD/4)
min(df3$RMSD)
final_arima_weekly = Arima(univariate_ts2, order = c(0,1,1), xreg = xreg_touse2)
final_arima_weekly$coef
ts.plot(univariate_ts2)
final_arima_fit2 = univariate_ts2 - residuals(final_arima_weekly)
points(final_arima_fit2, type = "l", col=2, lty=2)
title(main = "Actual Time Series vs. Fitted Time Series")
df4 <- data.frame(matrix(0, nrow=36, ncol=3))
colnames(df4) = c("p","q","RMSD")
pp = c(0,1,2,3,4,5)
qq = c(0,1,2,3,4,5)

count = 1

for (a in pp){
  for (b in qq){
    df4[count,1] = a
    df4[count,2] = b
    count = count + 1
  }
}
for (i in 1:4){
  start = 1
  end = i * round(nrow(datatouse2)/5)
```

```
  end2 = (i+1) * round(nrow(datatouse2)/5)
  if (end2 > nrow(datatouse2)){
    end2 = dim(datatouse2)[1]
  }
  datatouse_full_train = datatouse2[start:end,]
  datatouse_full_test = datatouse2[(end+1):end2,]

  univariate_ts_train = as.ts(datatouse_full_train$Frequency)

  xreg_train = create_xreg(datatouse_full_train)
  xreg_test = create_xreg(datatouse_full_test)


  for (i in 1:nrow(df4)){
    p = df4[i,1]
    q = df4[i,2]
    gsarima = create_gsarima(univariate_ts_train, p, q, xreg_train)
    predict_model = predict(gsarima, n.ahead = nrow(datatouse_full_test), newobs = datatouse_full_test$
    mse_value = mean((predict_model$pred - datatouse_full_test$Frequency)^2)
    mse_value = sqrt(mse_value)
    df4[i,3] = df4[i,3] + mse_value
  }
}

df4 = df4 %>%
  mutate(RMSD = RMSD/4)
min(df4$RMSD)
final_garima_weekly = create_gsarima(univariate_ts2, 1, 3, xreg_touse2)
final_garima_weekly$coefficients
ts.plot(univariate_ts2)
final_garima_fit2 = univariate_ts2 - residuals(final_garima_weekly)
points(final_garima_fit2, type = "l", col=2, lty=2)
title(main = "Actual Time Series vs. Fitted Time Series")
isolates$Create_date_YM = format(as.Date(isolates$Create_date), "%Y-%m")
datatouse_full = as.data.frame(table(isolates$Create_date_YM))
colnames(datatouse_full)[colnames(datatouse_full) == "Var1"] <- "Date"
colnames(datatouse_full)[colnames(datatouse_full) == "Freq"] <- "Frequency"
datatouse_full$Date = as.character(datatouse_full$Date)
datatouse_full = datatouse_full[15:nrow(datatouse_full),]
rownames(datatouse_full) = NULL
datatouse_full[1:10,]

datatouse_full$Frequency = datatouse_full$Frequency / 100
datatouse_full$Frequency = round(datatouse_full$Frequency)
datatouse_full$Month = format(as.Date(paste(datatouse_full$Date,"-01",sep="")), "%m")
datatouse_full[1:10,]
univariate_ts3 = as.ts(datatouse_full$Frequency)
univariate_ts3 = Winsorize(univariate_ts3)
univariate_ts3
ts.plot(univariate_ts3, main = 'Evolution of Listeriosis for the Whole Dataset with Winsorization', ylal
datatouse_full$Month = format(as.Date(paste(datatouse_full$Date,"-01",sep="")), "%m")
xreg_touse_full = create_xreg(datatouse_full)
xreg_touse_full[1:10,]
```

```
df5 <- data.frame(matrix(0, nrow=216, ncol=4))
colnames(df5) = c("p","d","q","RMSD")
p = c(0,1,2,3,4,5)
d = c(0,1,2,3,4,5)
q = c(0,1,2,3,4,5)


count1 = 1

for(a in p){
    for (b in d){
      for (c in q){
                df5[count1,3] = c
                df5[count1,2] = b
                df5[count1,1] = a
                count1 = count1 + 1
      }
    }
}
for (i in 1:4){
  start = 1
  end = i * round(nrow(datatouse_full)/5)
  end2 = (i+1) * round(nrow(datatouse_full)/5)
  if (end2 > nrow(datatouse_full)){
    end2 = dim(datatouse_full)[1]
  }
  datatouse_full_train = datatouse_full[start:end,]
  datatouse_full_test = datatouse_full[(end+1):end2,]

  datatouse_full_train_f = datatouse_full_train$Frequency
  datatouse_full_test_t = datatouse_full_test$Frequency

  univariate_ts_train = as.ts(datatouse_full_train_f)


  xreg_test = create_xreg(datatouse_full_test)


  for (j in 1:nrow(df5)){
    p = df5[j,1]
    d = df5[j,2]
    q = df5[j,3]
    mse_list = c()

    xreg_train = create_xreg(datatouse_full_train)
    new_model = Arima(univariate_ts_train, order = c(p,d,q), method = "CSS", xreg = xreg_train)


    datatouse_full_train_f = datatouse_full_train$Frequency

    for (k in 1:length(datatouse_full_test_t)){
      predict_model = predict(new_model, n.ahead=1, newxreg = t(xreg_test[k,]))
      mse_list[k] = (predict_model$pred - datatouse_full_test_t[k])^2
```

```
      datatouse_full_train_f = append(datatouse_full_train_f, datatouse_full_test_t[k])
      new_data_ts = as.ts(datatouse_full_train_f)
      xreg_train = rbind(xreg_train, xreg_test[k,])
      new_model = Arima(new_data_ts, model=new_model, xreg = xreg_train)
    }

    mse_value = mean(mse_list)
    mse_value = sqrt(mse_value)  # RMSD
    df5[j,4] = df5[j,4] + mse_value
  }
}

df5 = df5 %>%
  mutate(RMSD = RMSD/4)
min(df5$RMSD)
final_arima_full = Arima(univariate_ts3, order = c(0,1,1), xreg = xreg_touse_full)
final_arima_full$coef
ts.plot(univariate_ts3)
final_arima_fit3 = univariate_ts3 - residuals(final_arima_full)
points(final_arima_fit3, type = "l", col=2, lty=2)
title(main = "Actual Time Series vs. Fitted Time Series")
df6 <- data.frame(matrix(0, nrow=36, ncol=3))
colnames(df6) = c("p","q","RMSD")
pp = c(0,1,2,3,4,5)
qq = c(0,1,2,3,4,5)

count = 1

for (a in pp){
  for (b in qq){
    df6[count,1] = a
    df6[count,2] = b
    count = count + 1
  }
}
for (i in 1:4){
  start = 1
  end = i * round(nrow(datatouse_full)/5)
  end2 = (i+1) * round(nrow(datatouse_full)/5)
  if (end2 > nrow(datatouse_full)){
    end2 = dim(datatouse_full)[1]
  }
  datatouse_full_train = datatouse_full[start:end,]
  datatouse_full_test = datatouse_full[(end+1):end2,]

  univariate_ts_train = as.ts(datatouse_full_train$Frequency)

  xreg_train = create_xreg(datatouse_full_train)
  xreg_test = create_xreg(datatouse_full_test)


  for (i in 1:nrow(df6)){
    p = df6[i,1]
```

```
    q = df6[i,2]
    gsarima = create_gsarima(univariate_ts_train, p, q, xreg_train)
    predict_model = predict(gsarima, n.ahead = nrow(datatouse_full_test), newobs = datatouse_full_test$
    mse_value = mean((predict_model$pred - datatouse_full_test$Frequency)^2)
    mse_value = sqrt(mse_value)
    df6[i,3] = df6[i,3] + mse_value
  }
}

df6 = df6 %>%
  mutate(RMSD = RMSD/4)
min(df6$RMSD)
final_garima_full = create_gsarima(univariate_ts3, 1, 2, xreg_touse_full)
final_garima_full$coefficients
ts.plot(univariate_ts3)
final_garima_fit3 = univariate_ts3 - residuals(final_garima_full)
points(final_garima_fit3, type = "l", col=2, lty=2)
title(main = "Actual Time Series vs. Fitted Time Series")
isolates$Create_date = format(as.Date(isolates$Create_date), "%Y-%m-%d")
isolates$Create_date_YM = format(as.Date(isolates$Create_date), "%Y-%m")
for (i in 1:dim(isolates[1])){
  date = as.numeric(format(as.Date(isolates$Create_date[i]), "%d"))
  isolates$week[i] = if(date >= 1 && date <= 7){
    1
  } else if(date >= 8 && date <= 14){
    2
  } else if(date >= 15 && date <= 21){
    3
  } else {
    4
  }
  isolates$Create_date_YMW[i] = sprintf("%s-%s", isolates$Create_date_YM[i], isolates$week[i])
}
datatouse2_full = as.data.frame(table(isolates$Create_date_YMW))
colnames(datatouse2_full)[colnames(datatouse2_full) == "Var1"] <- "Date(YMW)"
colnames(datatouse2_full)[colnames(datatouse2_full) == "Freq"] <- "Frequency"
datatouse2_full$`Date(YMW)` = as.character(datatouse2_full$Date)
datatouse2_full = datatouse2_full[17:nrow(datatouse2_full),]
datatouse2_full[nrow(datatouse2_full)+1,] = c("2013-11-4", 0)
datatouse2_full[nrow(datatouse2_full)+1,] = c("2013-12-1", 0)
datatouse2_full = datatouse2_full[order(datatouse2_full$`Date(YMW)`),]
rownames(datatouse2_full) <- NULL
datatouse2_full[1:10,]
datatouse2_full$Frequency = as.numeric(datatouse2_full$Frequency)
datatouse2_full$Month = format(as.Date(datatouse2_full$`Date(YMW)`), "%m")
datatouse2_full$Frequency = datatouse2_full$Frequency / 10
datatouse2_full$Frequency = round(datatouse2_full$Frequency)
datatouse2_full[1:10,]
univariate_ts4 = as.ts(datatouse2_full$Frequency)
univariate_ts4 = Winsorize(univariate_ts4)
univariate_ts4
ts.plot(univariate_ts4, main = 'Evolution of Listeriosis for the Whole Dataset with Winsorization', yla
xreg_touse_full2 = create_xreg(datatouse2_full)
```

```
xreg_touse_full2[1:10,]
df7 <- data.frame(matrix(0, nrow=216, ncol=4))
colnames(df7) = c("p","d","q","RMSD")
p = c(0,1,2,3,4,5)
d = c(0,1,2,3,4,5)
q = c(0,1,2,3,4,5)


count1 = 1

for(a in p){
    for (b in d){
      for (c in q){
                df7[count1,3] = c
                df7[count1,2] = b
                df7[count1,1] = a
                count1 = count1 + 1
      }
    }
}
for (i in 1:4){
  start = 1
  end = i * round(nrow(datatouse2_full)/5)
  end2 = (i+1) * round(nrow(datatouse2_full)/5)
  if (end2 > nrow(datatouse2_full)){
    end2 = dim(datatouse2_full)[1]
  }
  datatouse_full_train = datatouse2_full[start:end,]
  datatouse_full_test = datatouse2_full[(end+1):end2,]

  datatouse_full_train_f = datatouse_full_train$Frequency
  datatouse_full_test_t = datatouse_full_test$Frequency

  univariate_ts_train = as.ts(datatouse_full_train_f)


  xreg_test = create_xreg(datatouse_full_test)


  for (j in 1:nrow(df7)){
    p = df7[j,1]
    d = df7[j,2]
    q = df7[j,3]
    mse_list = c()

    xreg_train = create_xreg(datatouse_full_train)
    new_model = Arima(univariate_ts_train, order = c(p,d,q), method = "CSS", xreg = xreg_train)


    datatouse_full_train_f = datatouse_full_train$Frequency

    for (k in 1:length(datatouse_full_test_t)){
      predict_model = predict(new_model, n.ahead=1, newxreg = t(xreg_test[k,]))
```

```
      mse_list[k] = (predict_model$pred - datatouse_full_test_t[k])^2
      datatouse_full_train_f = append(datatouse_full_train_f, datatouse_full_test_t[k])
      new_data_ts = as.ts(datatouse_full_train_f)
      xreg_train = rbind(xreg_train, xreg_test[k,])
      new_model = Arima(new_data_ts, model=new_model, xreg = xreg_train)
    }

    mse_value = mean(mse_list)
    mse_value = sqrt(mse_value)  # RMSD
    df7[j,4] = df7[j,4] + mse_value
  }
}

df7 = df7 %>%
  mutate(RMSD = RMSD/4)
min(df7$RMSD)
final_arima_full_weekly = Arima(univariate_ts4, order = c(1,1,3), xreg = xreg_touse_full2)
final_arima_full_weekly$coef
ts.plot(univariate_ts4)
final_arima_full_weekly_fit = univariate_ts4 - residuals(final_arima_full_weekly)
points(final_arima_full_weekly_fit, type = "l", col=2, lty=2)
title(main = "Actual Time Series vs. Fitted Time Series")
df8 <- data.frame(matrix(0, nrow=36, ncol=3))
colnames(df8) = c("p","q","RMSD")
pp = c(0,1,2,3,4,5)
qq = c(0,1,2,3,4,5)

count = 1

for (a in pp){
  for (b in qq){
    df8[count,1] = a
    df8[count,2] = b
    count = count + 1
  }
}
for (i in 1:4){
  start = 1
  end = i * round(nrow(datatouse2_full)/5)
  end2 = (i+1) * round(nrow(datatouse2_full)/5)
  if (end2 > nrow(datatouse2_full)){
    end2 = dim(datatouse2_full)[1]
  }
  datatouse_full_train = datatouse2_full[start:end,]
  datatouse_full_test = datatouse2_full[(end+1):end2,]

  univariate_ts_train = as.ts(datatouse_full_train$Frequency)

  xreg_train = create_xreg(datatouse_full_train)
  xreg_test = create_xreg(datatouse_full_test)


  for (i in 1:nrow(df8)){
```

```
    p = df8[i,1]
    q = df8[i,2]
    gsarima = create_gsarima(univariate_ts_train, p, q, xreg_train)
    predict_model = predict(gsarima, n.ahead = nrow(datatouse_full_test), newobs = datatouse_full_test$F
    mse_value = mean((predict_model$pred - datatouse_full_test$Frequency)^2)
    mse_value = sqrt(mse_value)
    df8[i,3] = df8[i,3] + mse_value
  }
}

df8 = df8 %>%
  mutate(RMSD = RMSD/4)
min(df8$RMSD)
final_garima_full_weekly = create_gsarima(univariate_ts4, 2, 2, xreg_touse_full2)
final_garima_full_weekly$coefficients
ts.plot(univariate_ts4)
final_garima_full_weekly_fit = univariate_ts4 - residuals(final_garima_full_weekly)
points(final_garima_full_weekly_fit, type = "l", col=2, lty=2)
title(main = "Actual Time Series vs. Fitted Time Series")
AIC_column = c(AIC(final_arima), AIC(final_garima), AIC(final_arima_weekly), AIC(final_garima_weekly), A

BIC_column = c(BIC(final_arima), BIC(final_garima), BIC(final_arima_weekly), BIC(final_garima_weekly), B

RMSD_column = c(sqrt(mean((final_arima$residuals)^2)), sqrt(mean((final_garima$residuals)^2)), sqrt(mean
final_df = data.frame(AIC_column, BIC_column, RMSD_column)
colnames(final_df) = c('AIC', 'BIC', 'RMSD')
rownames(final_df) = c('ARIMA(0,1,2)', 'GARIMA(5,2)', 'ARIMA(0,1,1)', 'GARIMA(1,3)', 'Arima(0,1,1)', 'G

final_df
df <- data.frame(matrix(0, nrow=125, ncol=4))
colnames(df) = c("p","d","q","MSE_valid")
p = c(0,1,2,3,4)
d = c(0,1,2,3,4)
q = c(0,1,2,3,4)


count1 = 1

for(a in p){
    for (b in d){
      for (c in q){
              df[count1,3] = c
              df[count1,2] = b
              df[count1,1] = a
              count1 = count1 + 1
      }
    }
}
for (i in 1:4){
  start = 1
  end = i * round(nrow(datatouse)/5)
  end2 = (i+1) * round(nrow(datatouse)/5)
  if (end2 > nrow(datatouse)){
```

```
  end2 = dim(datatouse)[1]
}
datatouse_full_train = datatouse[start:end,]
datatouse_full_test = datatouse[(end+1):end2,]

datatouse_full_train_f = datatouse_full_train$Frequency
datatouse_full_test_t = datatouse_full_test$Frequency

datatouse_full_train2 = datatouse_full_train[1:floor(0.7*length(datatouse_full_train_f)),]
datatouse_full_valid = datatouse_full_train[(floor(0.7*length(datatouse_full_train_f))+1):length(data


univariate_ts_train = as.ts(datatouse_full_train2$Frequency)



xreg_test = create_xreg(datatouse_full_test)
xreg_full = create_xreg(datatouse_full_train)

xreg_valid = xreg_full[(floor(0.7*length(datatouse_full_train_f))+1):length(datatouse_full_train_f),]


datatouse_full_valid_series = datatouse_full_valid$Frequency

for (j in 1:nrow(df)){
  p = df[j,1]
  d = df[j,2]
  q = df[j,3]
  mse_list = c()

  xreg_train = xreg_full[1:floor(0.7*length(datatouse_full_train_f)),]
  new_model = Arima(univariate_ts_train, order = c(p,d,q), method = "CSS", xreg = xreg_train)


  datatouse_full_train2_f = datatouse_full_train2$Frequency

  for (k in 1:nrow(datatouse_full_valid)){
    predict_model = predict(new_model, n.ahead=1, newxreg = t(xreg_valid[k,]))
    mse_list[k] = (predict_model$pred - datatouse_full_valid_series[k])^2
    datatouse_full_train2_f = append(datatouse_full_train2_f, datatouse_full_valid_series[k])
    new_data_ts = as.ts(datatouse_full_train2_f)
    xreg_train = rbind(xreg_train, xreg_valid[k,])
    new_model = Arima(new_data_ts, model=new_model, xreg = xreg_train)
  }

  mse_value_valid = mean(mse_list)
  mse_value_valid = sqrt(mse_value_valid)  # RMSD
  df[j,4] = df[j,4] + mse_value_valid
}

best_index = which(df$MSE_valid == min(df$MSE_valid))
print(best_index)
```

```
  df$MSE_valid = 0

  new_p = df[best_index,1]
  new_d = df[best_index,2]
  new_q = df[best_index,3]


  xreg_full = create_xreg(datatouse_full_train)
  final_arima_model = Arima(datatouse_full_train_f, order = c(new_p, new_d, new_q), method = "CSS", xreg

  mse_list2 = c()


  for (l in 1:nrow(datatouse_full_test)){
      predict_model = predict(final_arima_model, n.ahead=1, newxreg = t(xreg_test[l,]))
      mse_list2[l] = (predict_model$pred - datatouse_full_test_t[l])^2
      datatouse_full_train_f = append(datatouse_full_train_f, datatouse_full_test_t[l])
      new_data_ts = as.ts(datatouse_full_train_f)
      xreg_full = rbind(xreg_full, xreg_test[l,])
      final_arima_model = Arima(new_data_ts, model=final_arima_model, xreg = xreg_full)
    }

  mse_value_test = mean(mse_list2)
  mse_value_test = sqrt(mse_value_test)  # RMSD
  print(mse_value_test)
}
```