File Uploading

For now our system is only supporting the .csv (Comma Separated Value) files. It will read the values from the uploaded csv file and store it as a data frame.

```
> datf ← read.delim(x, sep = ",")
```

Finding Grouping Mark and Decimal Mark

For finding the delimiters there are several steps. We start searching from the very first row of the dataset then traverse forward according to the rules. First we will take the value of the first row as string in a variable called `num`.

```
>num ← as.character(datf[rowIndex,1])
```

After that we will look for the delimiters in this number with their index value. There are four different types of delimiter that are used globally in number formats. They are: comma(,); dot(.); space( ) and single quote (‘). According to their position in the number they have different meanings. So at the very beginning we will find the delimiters and their index value.

```
> commaInd ← max(which(strsplit(num, "")[[1]] == ","))
> dotInd ← max(which(strsplit(num, "")[[1]] == "."))
> spaceInd ← max(which(strsplit(num, "")[[1]] == " "))
> quoteInd ← max(which(strsplit(num, "")[[1]] == "'"))
```

`commaInd` will hold the maximum index number for comma delimiter in this number. `dotInd` will hold the maximum index number for dot delimiter in this number. `spaceInd` will hold the maximum index number for space delimiter in this number. `quoteInd` will hold the maximum index number

for single quote delimiter in this number. If any of the four delimiters is absent inside the number, the return value will be minus infinity (-Inf).

After getting the index of every delimiter we keep them in an array according to the sequence. We will also take an array of the delimiters according to the sequence.

```
> indexArray ← c(commaInd,dotInd,spaceInd,quoteInd)
> delimArray ← c(",",".","  ","'")
```

Then we will look for how many types of delimiter are present in the number. If any of the indexes of the `indexArray` has other values except `-Inf`, we can say that the delimiter is present in the number. Then we will take the sum.

```
> delimCount ← sum( !(indexArray %in% "-Inf"))
```

If the `delimCount` has the value 1

It means there is only one type of delimiter present inside the number. If the number has only one type of delimiter we will follow the rules mentioned below.

*If the delimiter is either comma/dot

Step 1: Check for the type of the specific delimiter

As the number has only one delimiter inside it, at first we will find which delimiter is it.

```
> delimIndex ← which(!( indexArray %in% "-Inf"))
```

So `delimIndex` is the index number from the `indexArray` which has a non `-Inf` value. The index number refers to the type of delimiter.

Step 2: Check for the number of occurrence of this specific delimiter

- **If there is only one delimiter inside of the number:**
  - If there are 3 digits after the comma for the 1st instance of the dataset,then we will initially consider the delimiter as a grouping mark and check other rows for more information. for example: 123,333. Here the delimiter is comma

```
if((str_length(num) - indexArray[1]) == 3){
    if(decimalMark != ","){
       groupingMark ← ","
     }
    next
   }
```

  - If any of the instances has any number of digits except 3 after delimiter, then the delimiter will be used as a decimal mark. For example: 12,2231;1234,22. Here the delimiter is comma.

```
if (sum(strsplit(num, "")[[1]] == delim) == 1){
  if((str_length(num) - indexArray[1]) != 3){
        decimalMark ← delim
      }
    }
```

- **If there is more than one delimiter inside of the number:**

In this case the delimiter will be considered as a grouping mark for the conversion. For example: 123,223,231. Here the delimiter is comma.

```
if (sum(strsplit(num, "")[[1]] == delim) == 1){
     ..........
}
else{
          groupingMark ← delim
  }
```

*If the delimiter is either space/single quote

The delimiter space and single quote is only used as a grouping mark in the number format globally. So if we found any of these delimiters we will use it as a grouping mark.

```
else if(delimIndex == 3){
          groupingMark ← " "
        }
 else if(delimIndex == 4){
          groupingMark ← "'"
        }
```

If the `delimCount` has the value 2

It means there are two types of delimiter present inside the number. If the number has two types of delimiter we will follow the rules mentioned below.

If there are two type of delimiter found in a single number we can easily say that one of them is a decimal mark and the other one is the grouping mark. In this type of case we will search for both of the delimiters. The delimiter

situated in the right most index will be considered as a decimal mark and the other one will be a grouping mark.

For example : 123,456,789.23 ; In this number there are two types of delimiter. Comma and point. The index of the commas are 4,8 respectively and the index of the point is 12. So the point is the right most delimiter. So in this example point is considered as a decimal mark and the comma will be a grouping mark.

```
if(indexArray[delimIndex[1]] > indexArray[delimIndex[2]]){
          groupingMark ← delimArray[delimIndex[2]]
          decimalMark ← delimArray[delimIndex[1]]
 }
 else{
          groupingMark ← delimArray[delimIndex[1]]
          decimalMark ← delimArray[delimIndex[2]]
 }
```

Final Conversion

After getting both the groupingMark and the decimalMark we will stop traversing the rows. Otherwise the traversing will continue until we can find both marks.

```
for (rowIndex in 1:nrow(datf)) {
      if(groupingMark != decimalMark){
        if(groupingMark != "X" && decimalMark != "X"){
          break;
        }
```

After breaking up the loop or finishing the loop we will parse the numbers of the dataset according to the information that we have so far by traversing the rows

- If we can not find any of the marks then we will only parse the number without any grouping or decimal mark.

```
if(groupingMark == decimalMark){
          if(groupingMark == "X"){
                datf[,1] = parse_number(datf[,1])
 }
```

- If we find same delimiter as grouping and decimal mark, we can consider it as decimal mark.

```
 else{
    datf[,1] = parse_number(datf[,1], locale =   locale
(decimal_mark = decimalMark))
   }
```

- If any of the marks is not present inside of the numbers of the dataset, we will parse the numbers according to the mark that we have.

```
 if(groupingMark == "X"){
     datf[,1] = parse_number(datf[,1], locale = locale
(decimal_mark = decimalMark))
   }
 else if (decimalMark == "X"){
     datf[,1] = parse_number(datf[,1], locale = locale
(grouping_mark = groupingMark))
   }
```

- If we have both of the marks it is very easy to parse them according to the locale with the grouping and decimal marks.

```
 datf[,1] = parse_number(datf[,1], locale = locale
(grouping_mark = groupingMark, decimal_mark = decimalMark))
```