Introduction to R for Data Science

Week 3

Value Matching

`%in%` is used to match values inside of a vector (list).

```
# Get the top 20 airports according to the most number of flights
> top20airports <- names(sort(table(myDataFile$Origin))[1:20], decreasing = TRUE)

> top20airports

[1] "ATL" "ORD" "DFW" "DEN" "LAX" "PHX" "IAH" "LAS" "DTW" "SFO" "SLC" "EWR" "MCO" "MSP" "CLT" "LGA" "JFK" "BOS" "SEA" "BWI"

# Using %in%
# How many of the flights originated from one of the top 20 airports?
> sum(myDataFile$Origin %in% top20airports)

[1] 3597054
```

Using Table to Select Data

The `table` command can be helpful to select data only for specific values.

```
# Select only one destination
> table(myDataFile$Dest)['DCA']

DCA
86671

# Select multiple destinations, here c('DCA', 'IAD') will concat these two destinations.
# Also known as indexing.
> table(myDataFile$Dest)[c('DCA', 'IAD')]

DCA   IAD
86671 76022
```

Use Indices in Table

```
# Top 20 airports
> top20airports <- names(sort(table(myDataFile$Origin))[1:20], decreasing = TRUE)

# Here we use the names of the top 20 airports and grab their destination count
# by providing indices from `top20airports` to the table command
> table(myDataFile$Dest)[top20airports]

ATL   ORD   DFW   DEN   LAX   PHX   IAH   LAS   DTW   SFO   SLC
414521 350452 281401 241470 215685 199416 185160 172871 162000 140579 139077

EWR   MCO   MSP   CLT   LGA   JFK   BOS   SEA   BWI
138491 130859 130320 126030 119117 118802 117944 109075 104068
```

Question

What does these pieces of code do?

```
> tapply(myDataFile$DepDelay <= 0 & myDataFile$Origin == 'IND', myDataFile$Origin, sum, na.rm = TRUE)['IND']
```

```
> table(myDataFile$Origin)['IND']
```

Try dividing these two operations and evaluate the result.

```
> tapply(myDataFile$DepDelay <= 0 & myDataFile$Origin == 'IND', myDataFile$Origin, sum, na.rm = TRUE)['IND'] /
  table(myDataFile$Origin)['IND']
```

Leaving a Specification Blank

Calculating the number of flights from IND over the 12 months.

```
# These two commands do the same thing
# Leave a blank specification after the comma to get all the months
> tapply(myDataFile$Origin, list(myDataFile$Origin, myDataFile$Month), length)['IND', 1:12]

> tapply(myDataFile$Origin, list(myDataFile$Origin, myDataFile$Month), length)['IND', ]

   1    2    3    4    5    6    7    8    9   10   11   12
3580 3414 3764 3644 3768 3852 3986 3700 3300 3418 3126 3198
```

Class and Dimension

To find the type of the result we can use the `class` command.

```
> class(tapply(myDataFile$Origin, list(myDataFile$Origin, myDataFile$Month), length)[c('IND', 'ATL'), ])

[1] "matrix"
```

To find the dimension of the matrix we can use the `dim` command.

```
dim(apply(myDataFile$Origin, list(myDataFile$Origin, myDataFile$Month), length)[c('IND', 'ATL'), ])

[1]  2 12
```

Adding a New Column

Adding a new column to the dataset.

```
# Splitting time into 4 parts
> v <- ceiling(myDataFile$DepTime/600)

# Creating a new variable with NA values as long as the data file
> partsofday <- rep(NA, times=dim(myDataFile)[1])

# Splitting parts of the day into names
> partsofday[v == 1] <- 'early morning'
> partsofday[v == 2] <- 'late morning'
> partsofday[v == 3] <- 'early evening'
> partsofday[v == 4] <- 'late evening'

# Adding partsofday as a new column TimeofDay to the dataset
> myDataFile$TimeofDay <- partsofday
```

End of Week 3