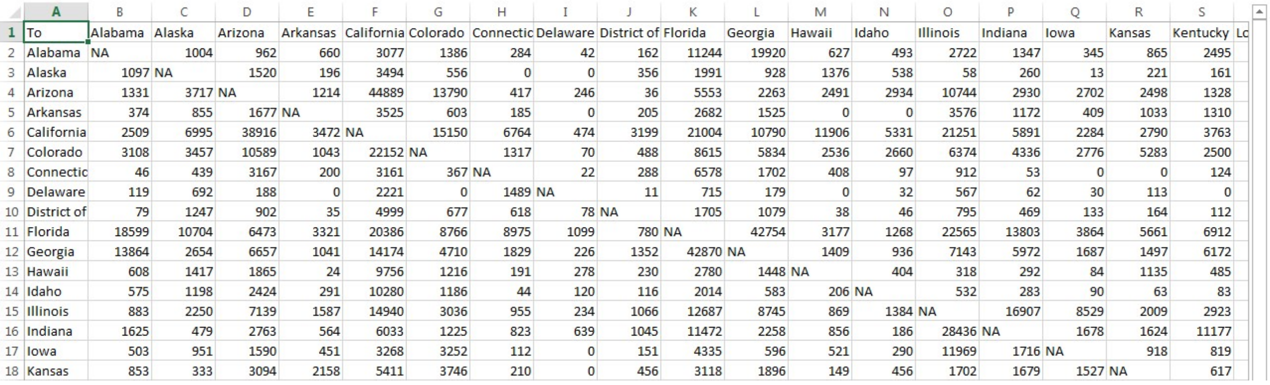
Hand in ALL questions on 6th March, 2017

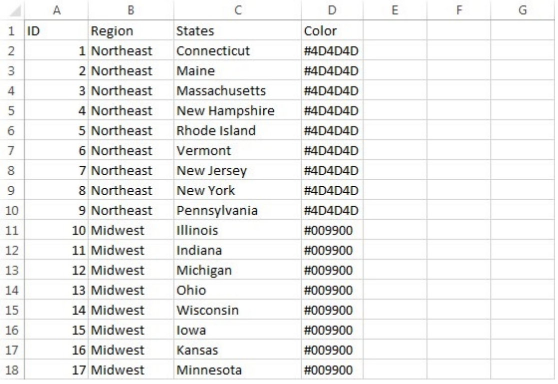
1. In this question, we are required to replicate the interactive version of chord diagram on page 54 of Chapter 2. To construct this chord diagram, the following data files are used: (1) `migration2012.csv` which stores the migration figures among the states in the United States (2) `state_chord.csv` which stores the configuration of the states in the chord diagram. The screenshots of both files are given below:

`migration2012.csv`



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	
1	To	Alabama	Alaska	Arizona	Arkansas	California	Colorado	Connectic	Delaware	District of	Florida	Georgia	Hawaii	Idaho	Illinois	Indiana	Iowa	Kansas	Kentucky	Lc
2	Alabama	NA	1004	962	660	3077	1386	284	42	162	11244	19920	627	493	2722	1347	345	865	2495	
3	Alaska	1097	NA	1520	196	3494	556	0	0	356	1991	928	1376	538	58	260	13	221	161	
4	Arizona	1331	3717	NA	1214	44889	13790	417	246	36	5553	2263	2491	2934	10744	2930	2702	2498	1328	
5	Arkansas	374	855	1677	NA	3525	603	185	0	205	2682	1525	0	0	3576	1172	409	1033	1310	
6	California	2509	6995	38916	3472	NA	15150	6764	474	3199	21004	10790	11906	5331	21251	5891	2284	2790	3763	
7	Colorado	3108	3457	10589	1043	22152	NA	1317	70	488	8615	5834	2536	2660	6374	4336	2776	5283	2500	
8	Connectic	46	439	3167	200	3161	367	NA	22	288	6578	1702	408	97	912	53	0	0	124	
9	Delaware	119	692	188	0	2221	0	1489	NA	11	715	179	0	32	567	62	30	113	0	
10	District of	79	1247	902	35	4999	677	618	78	NA	1705	1079	38	46	795	469	133	164	112	
11	Florida	18599	10704	6473	3321	20386	8766	8975	1099	780	NA	42754	3177	1268	22565	13803	3864	5661	6912	
12	Georgia	13864	2654	6657	1041	14174	4710	1829	226	1352	42870	NA	1409	936	7143	5972	1687	1497	6172	
13	Hawaii	608	1417	1865	24	9756	1216	191	278	230	2780	1448	NA	404	318	292	84	1135	485	
14	Idaho	575	1198	2424	291	10280	1186	44	120	116	2014	583	206	NA	532	283	90	63	83	
15	Illinois	883	2250	7139	1587	14940	3036	955	234	1066	12687	8745	869	1384	NA	16907	8529	2009	2923	
16	Indiana	1625	479	2763	564	6033	1225	823	639	1045	11472	2258	856	186	28436	NA	1678	1624	11177	
17	Iowa	503	951	1590	451	3268	3252	112	0	151	4335	596	521	290	11969	1716	NA	918	819	
18	Kansas	853	333	3094	2158	5411	3746	210	0	456	3118	1896	149	456	1702	1679	1527	NA	617	

`state_chord.csv`



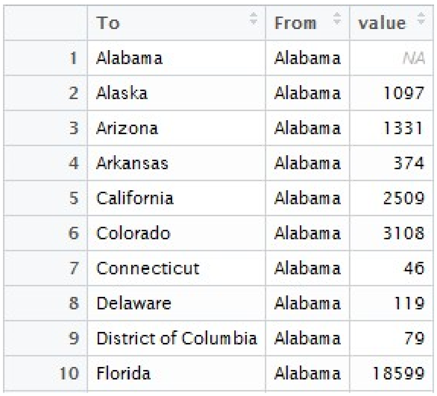
	A	B	C	D	E	F	G
1	ID	Region	States	Color			
2	1	Northeast	Connecticut	#4D4D4D			
3	2	Northeast	Maine	#4D4D4D			
4	3	Northeast	Massachusetts	#4D4D4D			
5	4	Northeast	New Hampshire	#4D4D4D			
6	5	Northeast	Rhode Island	#4D4D4D			
7	6	Northeast	Vermont	#4D4D4D			
8	7	Northeast	New Jersey	#4D4D4D			
9	8	Northeast	New York	#4D4D4D			
10	9	Northeast	Pennsylvania	#4D4D4D			
11	10	Midwest	Illinois	#009900			
12	11	Midwest	Indiana	#009900			
13	12	Midwest	Michigan	#009900			
14	13	Midwest	Ohio	#009900			
15	14	Midwest	Wisconsin	#009900			
16	15	Midwest	Iowa	#009900			
17	16	Midwest	Kansas	#009900			
18	17	Midwest	Minnesota	#009900			

Refer to the `migration2012.csv`, we observed that 1004 people migrated from Alaska to Alabama, for example. Similarly, 1331 people migrated from Alabama to Arizona. To construct the interactive chord diagram, the following procedures are required:

(a) Read the data values of data files `migration2012.csv` and `state_chord.csv` into data frames `dat` and `states` respectively.

(b) Convert the data frame `dat` into a matrix in which the migration figures of states are arranged in the order of regions in the United States.

(i) Since the migration figures in `dat` are arranged in the format of matrix, you are required to rearrange the data values into the following format by the function `melt()` in the package `reshape`.

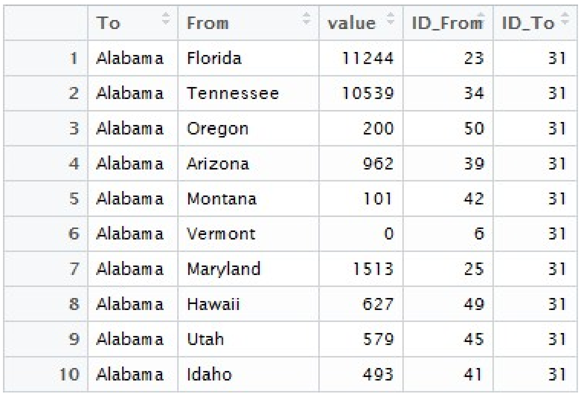


	To	From	value
1	Alabama	Alabama	NA
2	Alaska	Alabama	1097
3	Arizona	Alabama	1331
4	Arkansas	Alabama	374
5	California	Alabama	2509
6	Colorado	Alabama	3108
7	Connecticut	Alabama	46
8	Delaware	Alabama	119
9	District of Columbia	Alabama	79
10	Florida	Alabama	18599

Note that the variable names of columns are required to change as well.

(ii) Based on the data frames in (i), the states name in the columns `From` and `To` may contain the punctuation `:`. Then, the function `gsub()` can be considered to replace all colons by blanks.

(iii) In the data frame `states`, the states are categorized into four regions in the United States. Then, based on the data frame in (ii), construct two new columns `ID_From` and `ID_To` which are indeed the values of `ID` in the data frame `states`.

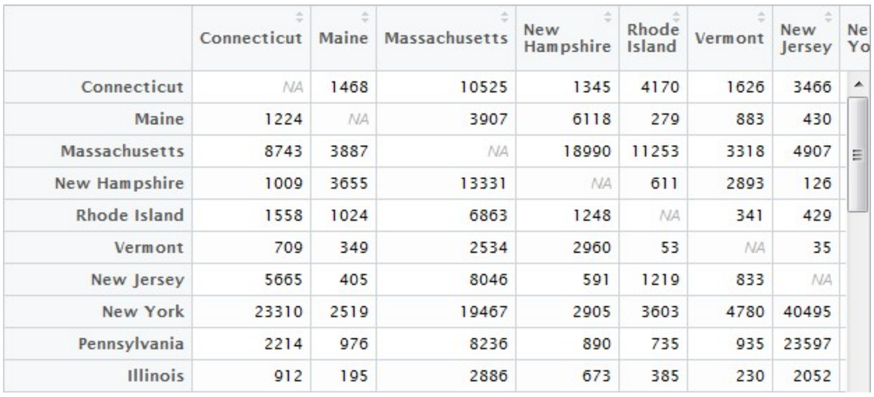


	To	From	value	ID_From	ID_To
1	Alabama	Florida	11244	23	31
2	Alabama	Tennessee	10539	34	31
3	Alabama	Oregon	200	50	31
4	Alabama	Arizona	962	39	31
5	Alabama	Montana	101	42	31
6	Alabama	Vermont	0	6	31
7	Alabama	Maryland	1513	25	31
8	Alabama	Hawaii	627	49	31
9	Alabama	Utah	579	45	31
10	Alabama	Idaho	493	41	31

Hint: You may use the `merge()` function and `colnames()` function to construct the above data frame.

(iv) Using the package `dplyr`, sort the observations of the data frame in (iii) in the ascending order of `ID_From` and then `ID_To`. [Hint: Use `arrange()` function.]

(v) Convert the data frame in (iv) into a matrix and the column names and row names are changed to the state names.

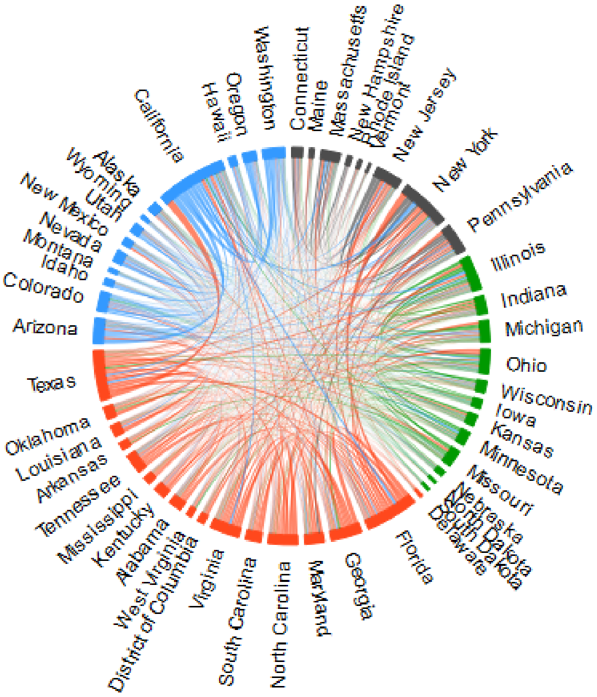


	Connecticut	Maine	Massachusetts	New Hampshire	Rhode Island	Vermont	New Jersey	Ne Yo
Connecticut	NA	1468	10525	1345	4170	1626	3466	
Maine	1224	NA	3907	6118	279	883	430	
Massachusetts	8743	3887	NA	18990	11253	3318	4907	
New Hampshire	1009	3655	13331	NA	611	2893	126	
Rhode Island	1558	1024	6863	1248	NA	341	429	
Vermont	709	349	2534	2960	53	NA	35	
New Jersey	5665	405	8046	591	1219	833	NA	
New York	23310	2519	19467	2905	3603	4780	40495	
Pennsylvania	2214	976	8236	890	735	935	23597	
Illinois	912	195	2886	673	385	230	2052	

(c) Display the interactive chord diagram using the function `chorddiag()` in the package `chorddiag` which can be installed by the following command:

```
devtools::install_github('mattflor/chorddiag')
```

The usage of the function `chorddiag()` can be found in the documentation. If the visualization is generated appropriately, the following output will be produced.



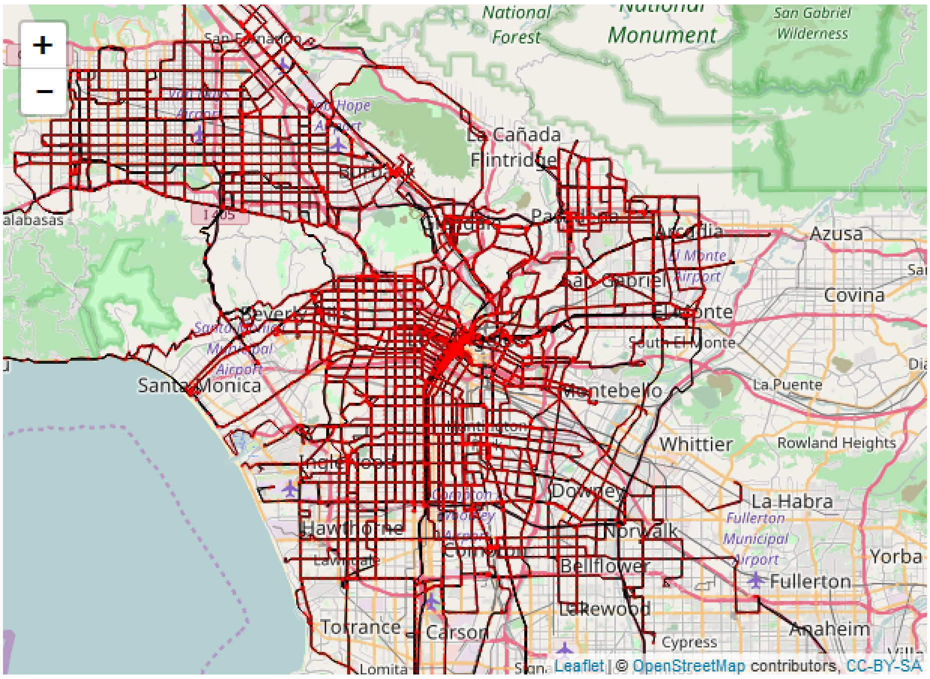
2. In Example Class 3, the bus map of New York City was constructed. In that process, the shapefiles of bus stops and bus lines are re-projected to the same coordinate reference system (CRS) WGS84. The `SpatialLineDataFrame` of all bus lines was reconstructed and stored in `SLDF`. Then, the bus lines and bus stops are overlaid on the map of New York City using the `leaflet()` function of the package `leaflet`.

   In this question, you are required to repeat this exercise to map of Los Angeles. However, the bus lines in Los Angeles are far more complicated. Information of bus stops and bus routes can be found in the following website:

   ```
   http://developer.metro.net/introduction/gis-data/download-gis-data/
   ```

   The GIS data of bus stops is stored in the file `BusStops1216.zip` which contains two shapefiles: one is for "Lines Serving Stops" which shows what Lines serve each stop and the other is for "Stops On Lines" which shows the stops on a given Line; it is sorted by Line, Direction and Stop Number. Then, we can choose the "Stops On Lines" shapefile for our purpose. On the other hand, the GIS data of bus lines are stored in the files `LimExp1216.zip`, `RapidBRT1216.zip`, `ComCir1216.zip`, `LocalCBD1216.zip`, `Individuals1216.zip` and `LocalNonCBD1216.zip`. All of them can be downloaded from the above link.

   Basically, the techniques of generating the bus map of Los Angeles is similar to those of New York City. One major difference is the concatenation of all GIS data of bus lines into one `SpatialLineDataFrame`. Then, the map of Los Angeles is overlaid with the GIS data of bus stops and bus routes. Therefore, the map centered at Los Angeles is also required during the process. For the simplicity, all bus lines in Los Angeles are represented by the same color and they are shown as follows:



3. In Example Class 2, the package `highcharter` was considered to demonstrate the time series plot of exchange rates USD/JPY and AUD/JPY. Since both measurement scales of both exchange rates are rather similar, the same vertical axis can be used. Now, you are required to produce a time series plot of exchange rate AUD/JPY and GBP/USD using `highcharter` package. Here are the requirement of this chart:

   (a) Two individual vertical axes are used for plotting these two times series.

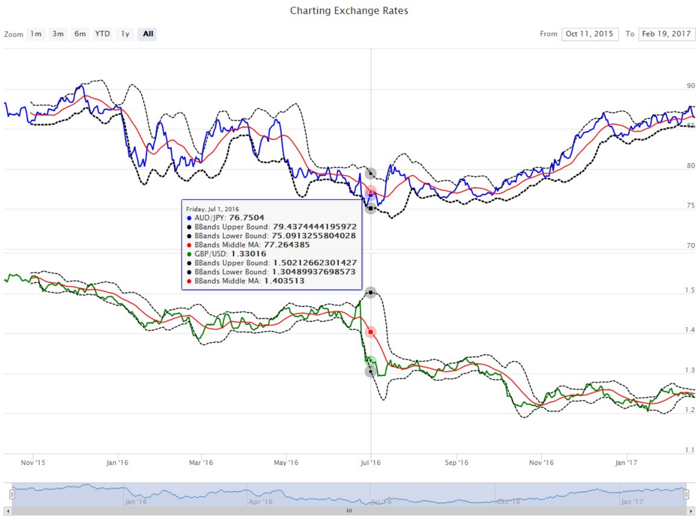
   (b) Labels of both vertical axes appear on the same side.

   (c) Each vertical axis shares the same proportion of height.

   (d) The lines of AUD/JPY and GBP/USD are indicated by blue line and green line respectively.

   (e) The Bollinger's band will be used as the technical indicator of both exchange rates. The moving average in the middle will be red in color while the lower bound and upper bound will be black in color and dashed.

   (f) Each line in the plot should be labeled.



4. The flights data frame in the package nycflights13 contains 336,776 observations related to On-time data for all flights that departed NYC in 2013. The columns in this data frame are described below:

Variables	Description
year, month, day	Date of departure
dep_time, arr_time	Actual departure and arrival times, local time zone
sched_dep_time, sched_arr_time	Scheduled departure and arrival times, local time zone
dep_delay, arr_delay	Departure and arrival delays, in minutes. Negative times represent early departures/arrivals
hour, minute	Time of scheduled departure broken into hour and minutes
carrier	Two letter carrier abbreviation, e.g. CX = Cathay Pacific
tailnum	Plane tail number
flight	Flight number
origin, dest	Origin and destination
air_time	Amount of time spent in the air, in minutes
distance	Distance between airports, in miles
time_hour	Scheduled date and hour of the flight as a POSIXct date

In this question, the packages ffbase, ffbase2, biglm and pROC are required. Since the package ffbase2 is not available in CRAN, you may install this package by the following command:

```
devtools::install_github(’edwindj/ffbase2’)
```

From the package ffbase2, the internal functions in dplyr can be used.

(a) Transform the data frame flights into the data frame of type ffdf and store the result to flightff. [Hint: Use the expression nycflights13::flights to access data frame flights.]

(b) Construct the following variables in the data frame fflightff:

(i) Delay = 1 when dep_delay > 0 or = 0 otherwise

(ii) DepHour = hour

(iii) Car = 1 if Carrier = ’DL’, ’US’, ’DH’ or ’UA’ or = 0 otherwise

(iv) Night = 1 when (hour $\geq$ 18 or hour $\leq$ 6) or = 0 otherwise

(v) Weekend = 1 when the departure date is Saturday, or = 0 otherwise

(c) Construct the data frame logitff of type ffdf to include the variables constructed in (b). Note that the records with missing Delay are excluded in the data frame.

(d) Construct the training and test data frames trainset and testset respectively by the following commands:

```
indx <- ff(1:nrow(logitff))
p <- 0.7
trainIndx <- ff(indx[1:trunc(length(indx)*p)])
trainset <- logitff[trainIndx,]
testIndx <- ff(indx[(trunc(length(indx)*p)+1):length(indx)])
testset <- logitff[testIndx,]
```

(e) Fit the binary logistic regression model with data frame trainset using the bigglm() function where the response is Delay and the predictors are DepHour, Car, Night and Weekend.

(f) Compute the results of predicted respone for both trainset and testset. The results are stored in the train_pred and test_pred respectively.

(g) Construct the confusion matrices for both training and test data sets separately using table() function. Note that the cutoff probability 0.5 is specified to identify predicted positive outcomes.

(h) Construct the ROC curve for the prediction results of test data using the roc() function and plot the ROC curve using the plot() function.

5. In this question, the NYC motor vehicle accidents during 2013-2016 will be be considered and the observations are stored in the data file NYPD_Motor_Vehicle_Collisions.csv. You are required to use the package sparklyr to finish the following tasks:

   (a) Transform the observations in the data file into a Spark DataFrame and store the result to nypd_tbl.

   (b) Using functions in the package dplyr, remove those records whose LATITUDE or LONGITUDE is missing or their values are zero or BOROUGH is missing. Then, sort the records in the descending order of UNIQUE_KEY.

   (c) After the cleaning process in (b), partition the data frame into training and test data by the sdf_partition() function. The resulting data sets are stored in train and test respectively.

   (d) Consider the training data in (c) only, apply the decision tree model with BOROUGH as the response and LATITUDE and LONGITUDE as the predictors. [Hint: Use ml_decision_tree() function with parameters max.bin=200L, max.depth=10L and seed=123L.]