Data Mining the US Department of Transportation Statistics on Aviation

Stephen Dimig

January 31, 2016

Introduction

The goal of this paper is to analyze the transportation dataset from the US Bureau of Transportation Statistics (BTS) that is hosted as an Amazon EBS volume snapshot and answer a set of interesting questions about the data. The dataset contains data and statistics from the US Department of Transportation on Aviation in CSV format. The dataset we are using does not extend beyond 2008, it contains flight data such as departure and arrival delays, flight times, etc. The set of questions that will be answered fall into three groups as outlined below.

All code and full results can be found at https://github.com/stephendimig/cc-capstone .

Group 1 Questions

1. Rank the top 10 most popular airports by numbers of flights to/from the airport.
2. Rank the top 10 airlines by on-time arrival performance.
3. Rank the days of the week by on-time arrival performance.

Group 2 Questions

1. For each airport X, rank the top-10 carriers in decreasing order of on-time departure performance from X. See Task 1 Queries for specific queries.
2. For each airport X, rank the top-10 airports in decreasing order of on-time departure performance from X. See Task 1 Queries for specific queries.
3. For each source-destination pair X-Y, rank the top-10 carriers in decreasing order of on-time arrival performance at Y from X. See Task 1 Queries for specific queries.

Group 3 Questions

1. Does the popularity distribution of airports follow a Zipf distribution? If not, what distribution does it follow?
2. Tom wants to travel from airport X to airport Z. However, Tom also wants to stop at airport Y for some sightseeing on the way. More concretely, Tom has the following requirements (see Task 1 Queries for specific queries):

- The second leg of the journey (flight Y-Z) must depart two days after the first leg (flight X-Y). For example, if X-Y departs January 5, 2008, Y-Z must depart January 7, 2008.
- Tom wants his flights scheduled to depart airport X before 12:00 PM local time and to depart airport Y after 12:00 PM local time.
- Tom wants to arrive at each destination with as little delay as possible (Clarification 1/24/16: assume you know the actual delay of each flight).

Methods and Data

System Installation and Setup

All work for this paper was performed on Amazon Web Services using a virtual machine instance running HortonWorks Sandbox 2.1. An EBS volume was created from a pre-existing snapshot containing the BTS transportation data statistics and attached to the virtual machine. In addition to this basic setup, the Apache Cassandra NoSQL database and the R Programming Language were also installed.

Attribute	Value	Description
Inst. Type	C3.xlarge	
AMI ID	ami-36d95d5e	hortonworks 2.1 - sandbox
vCPUs	4	
Memory	7.5 GB	
Inst. Storage	128 GB	Increased the storage size
EBS Vol. ID	snap-23a9cf5e	BTS transportation data
R	3.2	R programming language
Cassandra	2-1.2.10-1	NoSQL Database

MapReduce is fantastic at parallelizing work done on large data sets, but due to it's nature it can be difficult to use for some smaller tasks. Rather than struggling to make MapReduce perform every task required here, several languages were used together to perform the task.

Language	Description
Java	Used for map reduce programs to solve problems in Group 1
Pig	A language that generates map reduce from an SQL-like syntax
R	Used for post processing data filtered by MapReduce
Python	Used to filter and process data

cql	SQL-like query language for Cassandra

R is a programming language and software environment for statistical computing. It is exceptional at dealing with tabular data like what was found in this set of problems, but does not scale and is performs poorly on large datasets. R was used to process data where the majority of the heavy lifting was already done using MapReduce (either with Java or Pig). The following R packages were used in analyzing this data.

Package	Description
devtools	Requuired to install rhdfs
rhdfs	Provides basic connectivity to HDFS
dplyr	Used for cleaning data
zipfR	Used for zipf distributions
fitdistrplus	Used to find a distribution to it data

Cleaning the Data

The data was cleaned by reading it off the attached EBS data volume, processing it with R to filter out only the required fields, generating a temporary file, and then moving the file to HDFS.

The main R code that cleans the data looks like this.

```
# Unzip and read each file from the EBS volume
df <- read.csv(unz(zipfile, csvfile), stringsAsFactors=FALSE)

# Explicitly convert the date.
df$FlightDate <- as.Date(df$FlightDate)

# Select only certain rows required for the capstone.
my_df <- select(df, FlightDate, FlightNum, Origin, Dest, UniqueCarrier,
Carrier, ArrTime, ArrDelay, ArrDelayMinutes, DepTime, DepDelay,
DepDelayMinutes, DayOfWeek)
```

```r
# Write cleaned file, put it in HDFS, and remove local copy.
write.csv(my_df, file=txtfile, quote=FALSE, col.names=FALSE)
```

Group 1 Problems

The Group 1 Problems were solved using straight MapReduce with Java. For smaller problems this works well. A Java program is written using the Hadoop MapReduce framework and compiler. The jar file is then executed within Hadoop and the output is stored in HDFS.

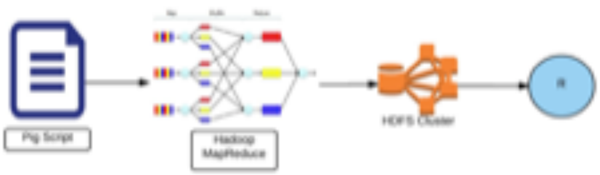
Group 2 Problems

The Group 2 Problems were the most complex as far as integration goes. I could not get the Cassandra/Pig interface to work so instead, I wrote a python language filter for each problem that took the output from the Pig script and created all of the cql commands that were required to load that data into Cassandra. The cql file was then run through cqlsh.

The pyhon scripts basically apply a regular expression to a line in the output file and then generate a corresponding cql statement.

Group 3 Problems

The Group 3 problems required more analysis with no database interaction. This set of problems was solved with R directly reading the output of the Pig script from HDFS.



Pig provides a higher level SQL-like syntax that is translated into MapReduce code. The Pig scripts perform the more computationally expensive work in this process.

Results

All code and full results can be found at https://github.com/stephendimig/cc-capstone .

Group 1 Questions

Rank the top 10 most popular airports by numbers of flights to/from the airport.

Airport	Description	Flights
ORD	Chicago O'Hare International	12449354
ATL	Hartsfield Jackson Atlanta International	11540422
DFW	Dallas Fort Worth International	10799303
LAX	Los Angeles International	7723596
PHX	Phoenix Sky Harbor International Airport	6585534
DEN	Denver International	6273787
DTW	Detroit Metropolitan Wayne County	5636622
IAH	George Bush Intercontinental Houston	5480734
MSP	Minneapolis-St Paul International	5199213
SFO	San Francisco International	5171023

Rank the top 10 airlines by on-time arrival performance.

Carrier	Description	Avg Delay
HA	Hawaiian Airlines, Inc.	3.9542668
AQ	9 Air Co Ltd	4.9505897
PS	Ukraine International Airlines	5.627902
ML	Air Mediterranee	8.518365
WN	Southwest Airlines Co.	9.025299
F9	Frontier Airlines, Inc.	9.871182
PA	M/S Airblue (PVT) Ltd	10.189628
US	Piedmont Airlines, Inc	10.285916
NW	Northwest Airlines, Inc.	10.332496
EA	Operador Aereo Andalus S.A	10.360811

Rank the days of the week by on-time arrival performance.

Day	Avg Delay
FRI	9.265108
MON	10.237862
SUN	10.864509
SAT	11.019846
TUE	11.180128
WED	12.689463
THU	13.256688

Group 2 Questions

For each airport X, rank the top-10 carriers in decreasing order of on-time departure performance from X. See Task 1 Queries for specific queries.

See Appendix A.1.

For each airport X, rank the top-10 airports in decreasing order of on-time departure performance from X. See Task 1 Queries for specific queries.

See Appendix A.2.

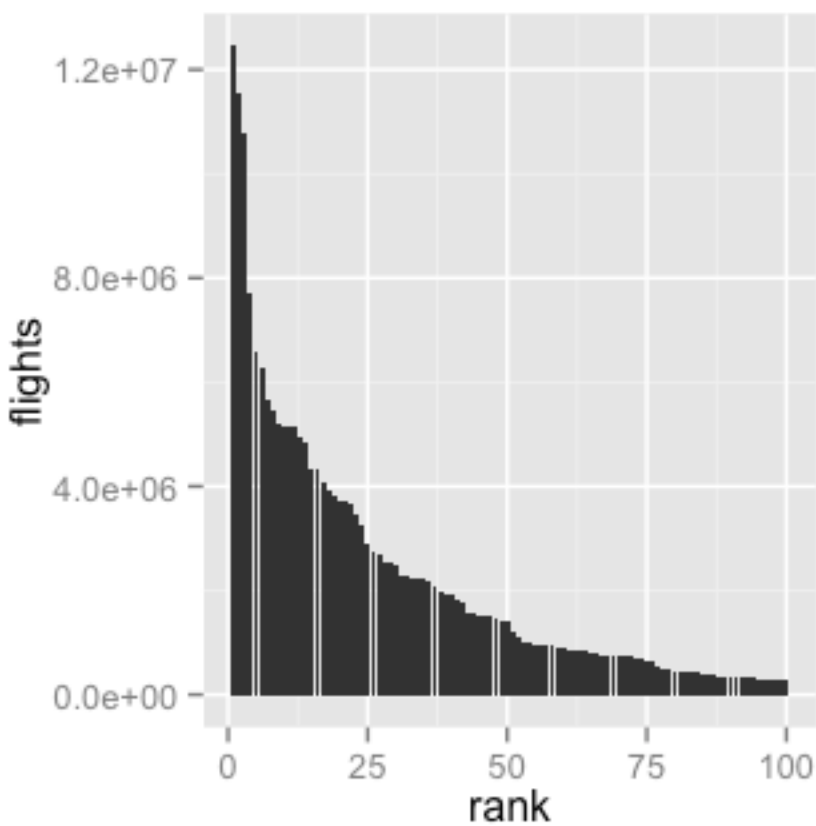
For each source-destination pair X-Y, rank the top-10 carriers in decreasing order of on-time arrival performance at Y from X. See Task 1 Queries for specific queries.

See Appendix A.3.

Group 3 Questions

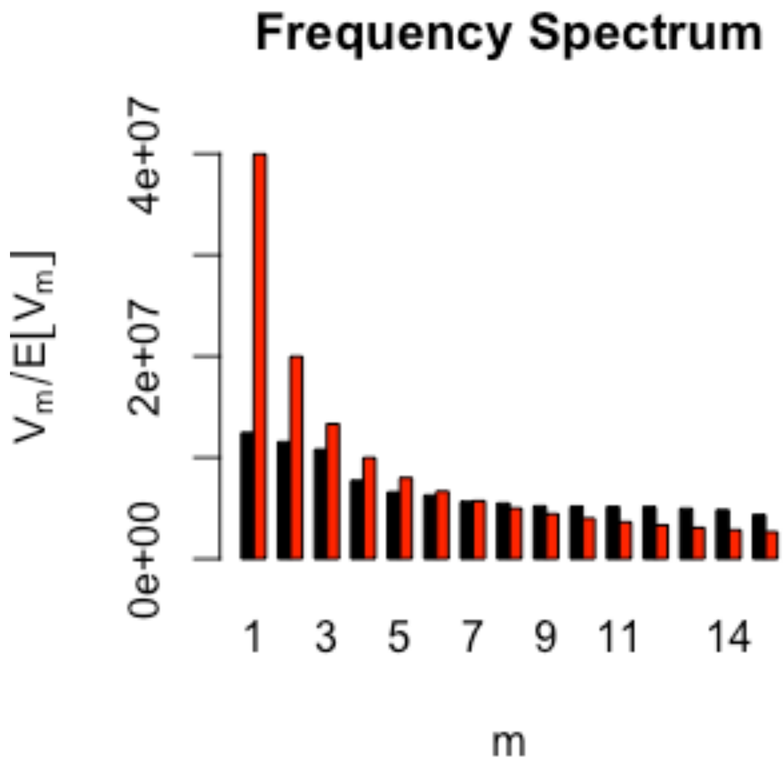
Does the popularity distribution of airports follow a Zipf distribution? If not, what distribution does it follow?

Zipf distributions are used in linguistics. Zipf's law states that given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table. As applied to airports in our problem, this means that the highest ranked airport should have roughly double the nunmber of flights as the second rated. The second rated should have double the third and so on. Our data when the number of flights looks very much like a zipf distribution. There is enough doubt about that bulge in the middle though (a typical zipf has an almost 90 degree elbow) to warrant some analysis.

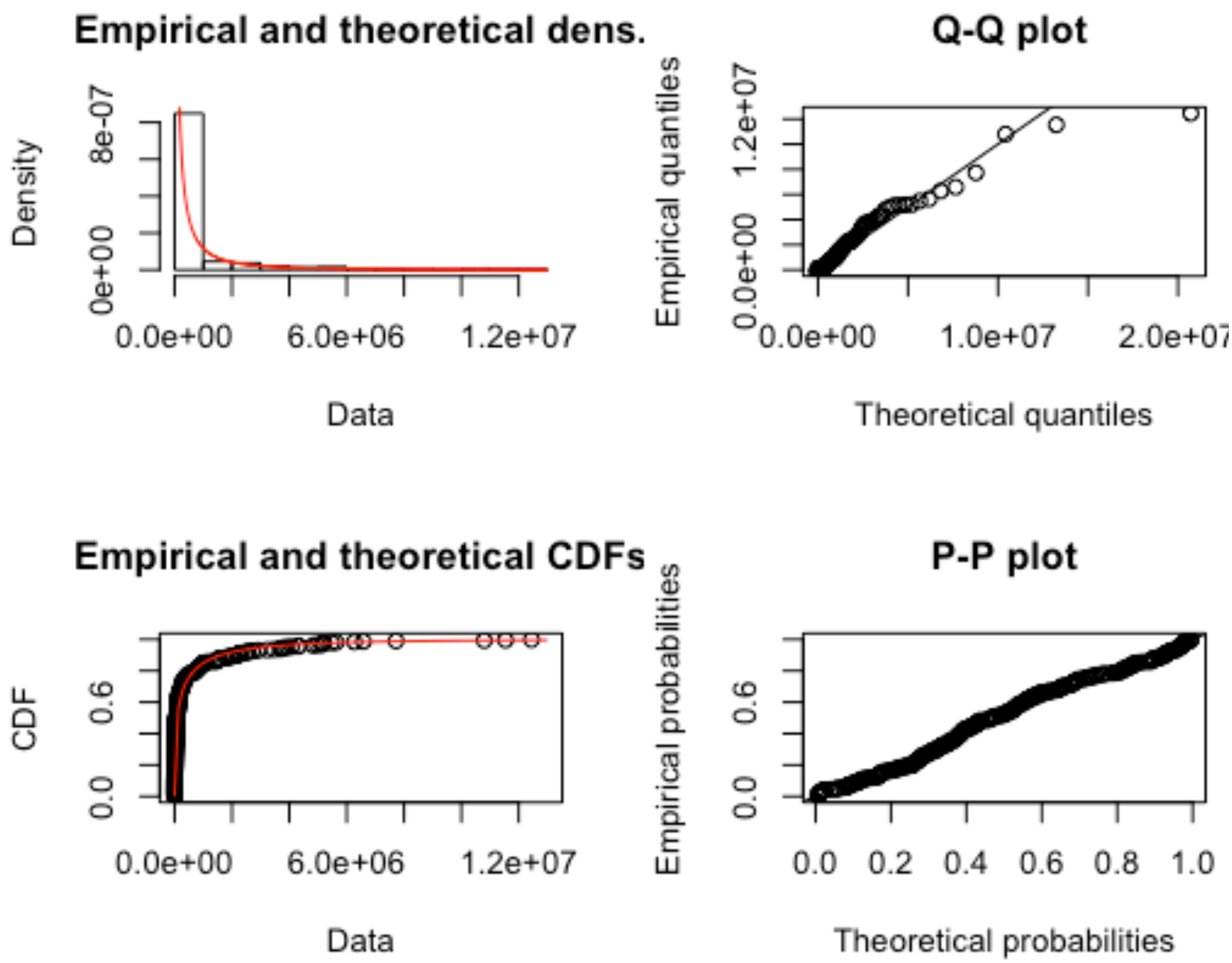


The zipfR R package allows you to compare your data against what a theoretical zipf distribution would look like if it had the same kind of bounds. When you run our data against the theoretical zipf, you see the problem that the most popular airports are not quie popular enough for a zipf.

```
## Warning in estimate.model.lnre.zm(model, spc = spc, param.names =
## missing.param, : estimated parameter values may be incorrect (code
3)
```



So what distribution does our data follow? The fitdistrplus R packages allows you to run various diagnostics against your data to determine which distribution it follows. It is a kind of trial and error approach, but the tools are nice enough that you can find a distribution. In our case, the data seem to fit a Weibull distribution almost perfectly.



Tom's Unusual Flight

See Appendix A.3.

Discussion

I like the results in the data but I think I might have not cleaned it properly. For example, I believe that flight cancellations should be removed rater than replacing the delay values with zeroes which skews the data for carriers with a smaller number of flights. I struggled at the beginning of this project due to some technical difficulties with the ami image I was using. I figured that out though and had a lot of fun. I was wanting to do something similar to this in the Data Science specialization from Johns Hopkins since R is so slow with large data sets. This proves to me you can extract the majority of the data using Hadoop and do the final analysis in R in a powerful way.

Appendix

A.1 For each airport X, rank the top-10 carriers in decreasing order of on-time departure performance from X. See Task 1 Queries for specific queries.

origin	unique_carrier	dep_delay_avg
CMI	DH	9.6494
CMI	EV	9.6927
CMI	MQ	11.754
CMI	OH	5.3643
CMI	PI	4.5229
CMI	TW	4.1582
CMI	US	2.8827

origin	unique_carrier	dep_delay_avg
BWI	AA	7.6571
BWI	CO	7.1413
BWI	DL	8.8151
BWI	EA	9.172
BWI	F9	4.9161
BWI	NW	8.3094
BWI	PA (1)	5.9429
BWI	TW	9.0849
BWI	US	8.5142
BWI	YV	7.676

origin	unique_carrier	dep_delay_avg
MIA	9E	0.5
MIA	EV	5.6696
MIA	ML (1)	7.632
MIA	NW	6.9902
MIA	PA (1)	4.8435
MIA	PI	9.0639
MIA	TZ	6.823
MIA	UA	8.2735
MIA	US	7.4273
MIA	XE	6.1034

```
 origin | unique_carrier | dep_delay_avg
--------+----------------+---------------
    LAX |             AA |        8.4199
    LAX |             F9 |        8.3621
    LAX |             FL |        8.0823
    LAX |         ML (1) |        7.1013
    LAX |             MQ |        5.0697
    LAX |             NW |        7.2525
    LAX |             OO |        6.0953
    LAX |             PS |        4.9739
    LAX |             TZ |        7.4569
    LAX |             US |        7.8037


 origin | unique_carrier | dep_delay_avg
--------+----------------+---------------
    IAH |             AA |        7.2697
    IAH |             DL |        8.2771
    IAH |             HP |        8.0406
    IAH |             NW |        6.1598
    IAH |             OO |        7.9431
    IAH |         PA (1) |        5.7343
    IAH |             PI |        4.6433
    IAH |             TW |        7.4534
    IAH |             US |        7.0557
    IAH |             WN |        6.2322
```

A.2 For each airport X, rank the top-10 airports in decreasing order of on-time departure performance from X. See Task 1 Queries for specific queries.

```
 origin | dest | dep_delay_avg
--------+------+---------------
    CMI |  ABI |             0
    CMI |  ATL |        9.6927
    CMI |  CVG |        6.3794
    CMI |  DAY |        3.6273
    CMI |  DFW |        9.5562
    CMI |  ORD |        11.943
    CMI |  PIA |        4.6324
    CMI |  PIT |        2.1701
    CMI |  STL |        4.0183


 origin | dest | dep_delay_avg
--------+------+---------------
    BWI |  CHO |        4.8261
```

```
    BWI |  DAB |         3.8378
    BWI |  GSP |         5.4311
    BWI |  IAD |         3.0871
    BWI |  MDT |         4.9014
    BWI |  MLB |         2.3842
    BWI |  OAJ |           5.32
    BWI |  SAV |              0
    BWI |  SRQ |         4.2689
    BWI |  UCA |         4.9399


 origin | dest | dep_delay_avg
--------+------+---------------
    MIA |  BUF |              1
    MIA |  GNV |          6.008
    MIA |  HOU |         3.6411
    MIA |  ISP |         4.4566
    MIA |  MCI |         5.3605
    MIA |  PSE |         4.9469
    MIA |  SAN |         2.5137
    MIA |  SHV |              0
    MIA |  SLC |         4.0702
    MIA |  TLH |         5.4429


 origin | dest | dep_delay_avg
--------+------+---------------
    LAX |  BZN |              1
    LAX |  DRO |              0
    LAX |  IDA |              0
    LAX |  LAX |              0
    LAX |  MAF |              0
    LAX |  PIH |              0
    LAX |  PMD |              3
    LAX |  RSW |              0
    LAX |  SDF |              0
    LAX |  VIS |         2.4805


 origin | dest | dep_delay_avg
--------+------+---------------
    IAH |  AGS |         2.8315
    IAH |  EFD |         3.9199
    IAH |  HOU |         2.3019
    IAH |  MDW |         5.9158
    IAH |  MLI |              0
    IAH |  MSN |              0
    IAH |  MTJ |          5.635
    IAH |  PIH |              4
```

origin	dest	dep_delay_avg
IAH	RNO	5.5072
IAH	VCT	5.3176

origin	dest	dep_delay_avg
SFO	BNA	3.0649
SFO	FAR	0
SFO	LGA	1.2121
SFO	MEM	5.4396
SFO	MSO	0.58333
SFO	OAK	2.5486
SFO	PIE	2.7283
SFO	PIH	0
SFO	SCK	4
SFO	SDF	0

A.3 For each source-destination pair X-Y, rank the top-10 carriers in decreasing order of on-time arrival performance at Y from X. See Task 1 Queries for specific queries.

origin	dest	unique_carrier	arrival_delay_avg
CMI	ORD	MQ	15.739

origin	dest	unique_carrier	arrival_delay_avg
IND	CMH	AA	8.25
IND	CMH	CO	4.3942
IND	CMH	DL	12.63
IND	CMH	EA	13.065
IND	CMH	HP	7.9906
IND	CMH	NW	7.6015
IND	CMH	US	7.8386

origin	dest	unique_carrier	arrival_delay_avg
DFW	IAH	AA	12.148
DFW	IAH	CO	10.001
DFW	IAH	DL	10.204
DFW	IAH	EV	10.692
DFW	IAH	MQ	12.976
DFW	IAH	OO	9.7365
DFW	IAH	PA (1)	9.3333
DFW	IAH	UA	8.8994
DFW	IAH	XE	12.893

```
 origin | dest | unique_carrier | arrival_delay_avg
--------+------+----------------+-------------------
    LAX |  SFO |             AA |            12.465
    LAX |  SFO |             CO |            14.002
    LAX |  SFO |             DL |            13.484
    LAX |  SFO |             EV |            13.399
    LAX |  SFO |             F9 |            6.9653
    LAX |  SFO |             MQ |            10.933
    LAX |  SFO |             NW |             12.79
    LAX |  SFO |             PS |            5.8304
    LAX |  SFO |             TZ |            6.2381
    LAX |  SFO |             US |            10.822


 origin | dest | unique_carrier | arrival_delay_avg
--------+------+----------------+-------------------
    JFK |  LAX |             AA |            15.045
    JFK |  LAX |             DL |            16.631
    JFK |  LAX |             HP |            14.865
    JFK |  LAX |         PA (1) |            17.094
    JFK |  LAX |             TW |            18.288
    JFK |  LAX |             UA |            11.469


 origin | dest | unique_carrier | arrival_delay_avg
--------+------+----------------+-------------------
    ATL |  PHX |             DL |            13.867
    ATL |  PHX |             EA |            14.009
    ATL |  PHX |             FL |             12.61
    ATL |  PHX |             HP |            13.367
    ATL |  PHX |             US |            12.687
```

A.4 Tom's Unusual Flight

```
Moved: 'hdfs://sandbox.hortonworks.com:8020/user/root/output' to trash
at: hdfs://sandbox.hortonworks.com:8020/user/root/.Trash/Current
[1] "CMI -> ORD Flights"
[1] "==================="
     flightno origin dest carrier       date dep_time delay
3206     4278    CMI  ORD      MQ 2008-04-03      706     0
3236     4373    CMI  ORD      MQ 2008-04-03      908     0
3265     4374    CMI  ORD      MQ 2008-04-03      557     0
3290     4401    CMI  ORD      MQ 2008-04-03      808     0
[1] ""
[1] "ORD -> LAX Flights"
[1] "==================="
     flightno origin dest carrier       date dep_time delay
```

```
3031       121     ORD   LAX       UA 2008-04-05     1219     0
3375       607     ORD   LAX       AA 2008-04-05     1948     0
3403       889     ORD   LAX       AA 2008-04-05     1815     0
3435      1345     ORD   LAX       AA 2008-04-05     1404     0
3463      1407     ORD   LAX       AA 2008-04-05     1213     0
3369       557     ORD   LAX       AA 2008-04-05     1641     6
3094       129     ORD   LAX       UA 2008-04-05     2102    12
3153       943     ORD   LAX       UA 2008-04-05     1506    12
3123       941     ORD   LAX       UA 2008-04-05     1712    19
3064       127     ORD   LAX       UA 2008-04-05     1847    20
3023       111     ORD   LAX       UA 2008-04-05     1208    38
[1] ""
Moved: 'hdfs://sandbox.hortonworks.com:8020/user/root/output' to trash
at: hdfs://sandbox.hortonworks.com:8020/user/root/.Trash/Current
[1] "JAX -> DFW Flights"
[1] "==================="
     flightno origin dest carrier       date dep_time delay
1545       845     JAX   DFW       AA 2008-09-09      722     1
[1] ""
[1] "DFW -> CRP Flights"
[1] "==================="
     flightno origin dest carrier       date dep_time delay
1493      3627     DFW   CRP       MQ 2008-09-11     1648     0
1521      3701     DFW   CRP       MQ 2008-09-11     1310     8
1438      3419     DFW   CRP       MQ 2008-09-11     1504     9
[1] ""
Moved: 'hdfs://sandbox.hortonworks.com:8020/user/root/output' to trash
at: hdfs://sandbox.hortonworks.com:8020/user/root/.Trash/Current
[1] "No flights found matching criteria X=SLC; Y=BFL; Z=LAX; DATE=2008-
01-04"
Moved: 'hdfs://sandbox.hortonworks.com:8020/user/root/output' to trash
at: hdfs://sandbox.hortonworks.com:8020/user/root/.Trash/Current
[1] "No flights found matching criteria X=LAX; Y=SFO; Z=PHX; DATE=2008-
12-07"
Moved: 'hdfs://sandbox.hortonworks.com:8020/user/root/output' to trash
at: hdfs://sandbox.hortonworks.com:8020/user/root/.Trash/Current
[1] "DFW -> ORD Flights"
[1] "==================="
     flightno origin dest carrier       date dep_time delay
5155      6441     DFW   ORD       OO 2008-10-06      920     0
5232      1104     DFW   ORD       UA 2008-10-06      655     0
5289      2268     DFW   ORD       AA 2008-10-06      920     0
5320      2320     DFW   ORD       AA 2008-10-06      556     0
5418      2328     DFW   ORD       AA 2008-10-06      812     0
5542      2336     DFW   ORD       AA 2008-10-06     1003     0
5604      2340     DFW   ORD       AA 2008-10-06     1047     0
5665      2344     DFW   ORD       AA 2008-10-06     1148     0
5356      2324     DFW   ORD       AA 2008-10-06      703     6
[1] ""
[1] "ORD -> DFW Flights"
```

```
[1] "==================="
     flightno origin dest carrier       date dep_time delay
5175      357    ORD  DFW      UA 2008-10-08     1658     0
5204      725    ORD  DFW      UA 2008-10-08     2016     0
5260       47    ORD  DFW      AA 2008-10-08     1919     0
5389     2325    ORD  DFW      AA 2008-10-08     1240     0
5451     2329    ORD  DFW      AA 2008-10-08     1332     0
5636     2341    ORD  DFW      AA 2008-10-08     1650     0
5692     2345    ORD  DFW      AA 2008-10-08     1754     0
5748     2357    ORD  DFW      AA 2008-10-08     1945     0
5776     2361    ORD  DFW      AA 2008-10-08     2100     0
5482     2331    ORD  DFW      AA 2008-10-08     1429     2
5138     5949    ORD  DFW      OO 2008-10-08     1529    11
5513     2333    ORD  DFW      AA 2008-10-08     1520    17
5721     2349    ORD  DFW      AA 2008-10-08     2024    94
5575     2337    ORD  DFW      AA 2008-10-08     1909   184
[1] ""
Moved: 'hdfs://sandbox.hortonworks.com:8020/user/root/output' to trash
at: hdfs://sandbox.hortonworks.com:8020/user/root/.Trash/Current
[1] "LAX -> ORD Flights"
[1] "==================="
     flightno origin dest carrier       date dep_time delay
1898      944    LAX  ORD      UA 2008-01-01      700     1
1831      110    LAX  ORD      UA 2008-01-01     1005     9
1957       88    LAX  ORD      AA 2008-01-01      853    11
1985      764    LAX  ORD      AA 2008-01-01      558    11
1802      106    LAX  ORD      UA 2008-01-01      856    12
2070     2276    LAX  ORD      AA 2008-01-01      631    12
2032     1372    LAX  ORD      AA 2008-01-01     1106    70
2055     1740    LAX  ORD      AA 2008-01-01      217   161
[1] ""
[1] "ORD -> JFK Flights"
[1] "==================="
     flightno origin dest carrier       date dep_time delay
2135      918    ORD  JFK      B6 2008-01-03     1853     0
1743     5366    ORD  JFK      OH 2008-01-03     1736     2
2133      908    ORD  JFK      B6 2008-01-03     1208     5
2134      916    ORD  JFK      B6 2008-01-03     1603    10
2103     2352    ORD  JFK      AA 2008-01-03     1708    18
1927     4138    ORD  JFK      MQ 2008-01-03     1425    28
1744     5466    ORD  JFK      OH 2008-01-03     1335   145
[1] ""
```