On the analysis of HL7 messages to track patient location and demographic trends

James Tsai (jjtsai@mail.smu.edu), Student and Wid Sogata (wsogata@mail.smu.edu), Student

***Abstract*—Currently there is no accurate methodology to track general demographic information of patients accessing the services of the medical center. For instance, the distances patients are traveling in order to access the services is not generally known. The most reliable source of truth can be derived from the HL7 messages since all patients that arrive at the medical must first be electronically registered and admitted. The goal is to parse this information from the HL7 messages and derive useful information and population trends.**

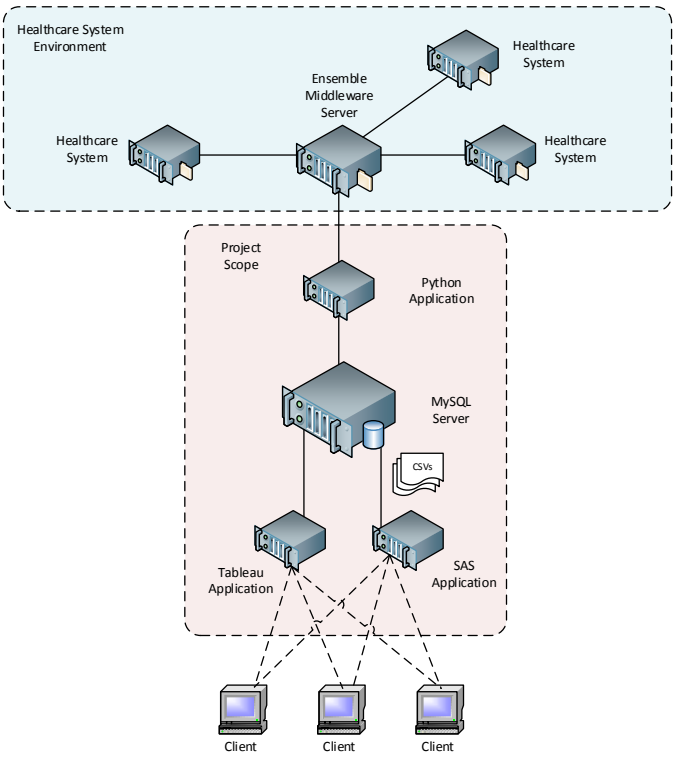
I. Introduction

THE HL7 messaging standard is used for transfer of clinical and administrative data between software applications used by various healthcare providers. Medical centers utilize this standard in order to exchange information between disparate systems that otherwise would not communicate with each other. For the purposes of this project, we are interested in the registration of a patient. The message type representing this event is the A01 message. Since the integration team archives all HL7 messages in a file-based format, it contains all the registration and admission of every patient that arrives at the medical center for any service. It is a matter of parsing and aggregating patient information contained in these message types in order to construct a baseline where we can perform further analysis.

II. Conceptual Design

In Figure 1, the data flow and components for our project is shown. The box labeled "Project Scope" contains the components that are created.

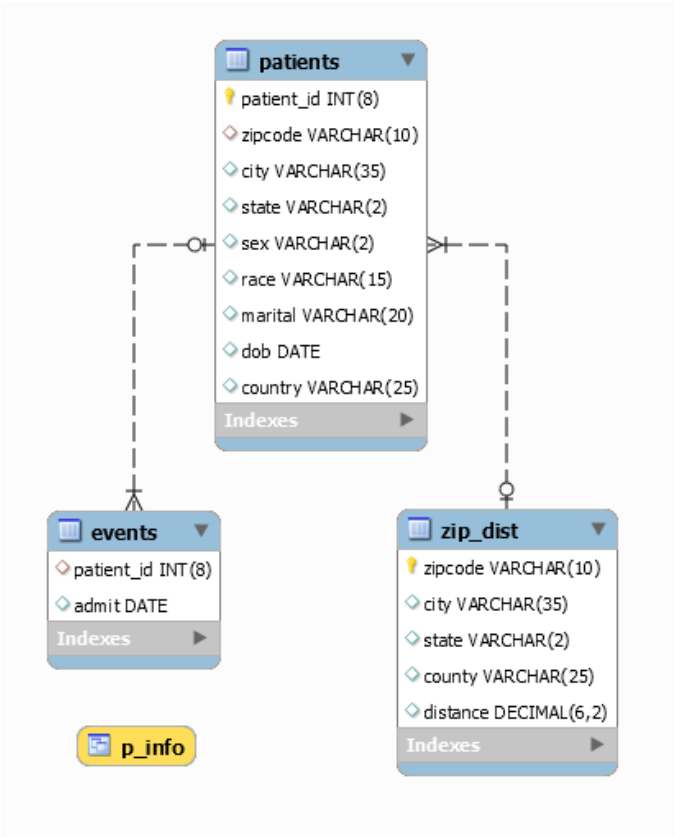
To achieve our goal, we propose a 3-tier architecture. Since we are dealing with patient information, we first implement a Python script to parse only the A01 messages and grab only the country code, zip code, date-of-birth, and gender from the PID segment, without propagating any of the patient identifying information such as Name, Address, SSN, or Medical Record Number. We will also grab the date of the event from the MSH segment. We then calculate the distance of the patient based on the latitude/longitude of their zip code and the latitude/longitude of the medical center's zip code. The database of latitude/longitude information for zip codes is downloaded from http://opengeocode.org/. The information we derived will then be inserted into a MySQL database on a daily basis for the previous days of HL7 messages. The trigger for this will be a daily cron job using a scheduler such as the crontab function in the UNIX operating system. The final component is to display the visualization of the data through the application Tableau and statistic results through SAS.



III. System Components

Main system components for this project:
Oracle Linux 7 Host Server – Oracle Corp.
Python ver. 2.7.10 – Python Software Foundation
My SQL ver. 5.6.27 – Oracle Corporation
Tableau Desktop ver. 9.0 – Tableau Software
SAS 9.4 – SAS Institute Inc.

IV. DATABASE DIAGRAM



V. TECHNICAL WORK

In this project we collect sample of HL7 messages data in the form of several flat files. The data from these files have exactly the same structure and format as the ones generated by Ensemble server.

Python Script and HL7 Data Directories:

```
HL7 Data directory:
/data/smu/hl7

Python Script directory:
/script/smu/hl7
```

Job Scheduler (cron):

```
Scheduler (CRON) - daily data processing at
midnight:
0 0 * * * python /script/smu/hl7/hl7parse.py
```

Python Script:

```
# MSDS7730
# hl7parse.py to process HL7 data and enter the results
# into the database hl7db
# James Tsai, Wid Sogata
# 12/10/15

import mysql.connector as sqlconn
import string, hl7, random, glob, os
from datetime import datetime
from datetime import date
from math import sin
from math import cos
from math import acos
from math import radians
from math import degrees

# start time
t1 = datetime.now()

# MySQL connection
print "Please enter database connection information."
h_name = raw_input ("Host Name: ")
u_name = raw_input ("User Name: ")
os.system ("stty -echo")
pword = raw_input ("Enter Password: ")
os.system ("stty echo")
db = sqlconn.connect(host = h_name, user = u_name,\
                     passwd = pword)

# Create database hl7db and its objects
dbc = db.cursor()
dbc.execute("CREATE DATABASE IF NOT EXISTS hl7db")
dbc.execute("USE hl7db")
dbc.execute("SET foreign_key_checks=0")
dbc.execute("DROP TABLE IF EXISTS events")
dbc.execute("DROP TABLE IF EXISTS patients")
dbc.execute("DROP TABLE IF EXISTS zip_dist")
dbc.execute("DROP VIEW IF EXISTS p_info")
dbc.execute("SET foreign_key_checks=1")

# Create table zip_dist to store zipcode information and distance
# from hospital
dbc.execute("CREATE TABLE zip_dist (\
                zipcode varchar(10) primary key,\
                city varchar(35),\
                state varchar(2),\
                county varchar(25),\
                distance numeric(6,2))")

# Create table patients to store patients information
dbc.execute("CREATE TABLE patients (\
                patient_id int(8) primary key,\
                zipcode varchar(10), \
                city varchar(35),\
                state varchar(2),\
                sex varchar(2),\
                race varchar(15),\
                marital varchar(20),\
                dob date,\
                country varchar(25),\
                FOREIGN KEY fk_zip (zipcode)\
                REFERENCES zip_dist (zipcode))")

# Create table events to store events related to patients
dbc.execute("CREATE TABLE events (\
                patient_id int(8),\
                admit date,\
                FOREIGN KEY fk_p_id (patient_id)\
                REFERENCES patients(patient_id)\
                ON DELETE CASCADE ON UPDATE CASCADE)")

# Create view p_info to dynamically store information needed for
# reporting and visualization
dbc.execute("CREATE VIEW p_info AS \
                SELECT distinct p.patient_id, \
                TIMESTAMPDIFF(YEAR,p.dob,CURDATE()) as age,\
                    p.sex, e.admit, z.distance\
                FROM patients p, events e, zip_dist z\
                WHERE p.zipcode = z.zipcode\
                AND p.patient_id = e.patient_id")

# Create indexes to improve performance
dbc.execute("CREATE INDEX p_pid_idx ON patients(patient_id)")
dbc.execute("CREATE INDEX p_zip_idx ON patients(zipcode)")
dbc.execute("CREATE INDEX e_pid_idx ON events(patient_id)")
dbc.execute("CREATE INDEX z_zip_idx ON zip_dist(zipcode)")
dbc.execute("commit")

# Function calculate_age to calculate patients age based on dob
def calculate_age(born):
    today = date.today()
    return today.year - born.year - \
            ((today.month, today.day) < (born.month, born.day))

# Function calcDist to calculate distance based on zipcodes
# latitude and longitude
def calcDist(lat_A, long_A, lat_B, long_B):
    distance = (sin(radians(lat_A)) *
                sin(radians(lat_B)) +
                cos(radians(lat_A)) *
                cos(radians(lat_B)) *
                cos(radians(long_A - long_B)))
    distance = (degrees(acos(distance))) * 69.09
    return distance

# Loop process to parse and read zips.csf file that contains
# geo location of zipcodes. It then stores the results to table
zip_dist.
f = open ('zips.csv')
for line in f:
    try:
        values = line.split(",")
        zip = values[0]
```

```
        state = values[4]
        lat = values[1]
        long = values[2]
        city = values[3]
        county = values[5]
        dist = calcDist(34.073759, -118.37376, float(lat),
float(long))
        dbc.execute ("INSERT INTO zip_dist \
                        (zipcode,city,state,county,distance)\
                      VALUES
(%s,%s,%s,%s,%s)",[zip,city,state,county,dist])
    except:
        pass
        print "duplicate found on zipcode ",zip
f.close()
dbc.execute("commit")

# Loop process to parse and read HL7 data files. It then stores
# the results to table patients and events.
now = datetime.now()
inserts = 0
count = 0
p_id = 0
for fname in glob.glob('/home/oracle/smu/PROJ/*.dat'):
    f = open (fname)
    for line in f:
        count = count+1
        try:
            h = hl7.parse(line)
            #patient demographicsj
            p_id += 1
            print p_id
            p_zip = str(h[2][11][0][4])
            print p_zip
            p_country = str(h[2][11][0][5])
            p_city =  str(h[2][11][0][2])
            p_state = str(h[2][11][0][3])
            p_sex =   str(h[2][8])
            p_death = str(h[2][30])
            p_race =  str(h[2][10])
            p_marital = str(h[2][16])
            p_birth_pl =   str(h[2][23])
            dob = str(h[2][7])
            dob = string.replace(dob, '-', '')
            admit = str(h[1][2])
            p_admit = datetime.strptime(admit[0:8], '%Y%m%d')
            p_dob = datetime.strptime(dob[0:8], '%Y%m%d')
            if p_dob > now:
               p_dob = now
            if len(p_city) > 35:
                p_city = p_city[:34]
            if len(p_state) > 20:
               p_state = p_state[:20]
            if len(p_zip) > 5:
               print p_zip
               p_zip = p_zip[:5]
            if p_zip == "":
               pzip = '0'
            if p_country != 'USA':
               print p_country
               pzip = '0'
            if len(p_race) > 15:
               p_race = 'Other'
            if len(p_marital) == 0:
               p_marital = 'U'
            else:
               p_marital = p_marital[0]
            dbc.execute ("INSERT INTO patients (patient_id,zipcode,\
                            city,state,country,sex,race,marital,dob)\
                          VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s)",\

[p_id,p_zip,p_city,p_state,p_country,p_sex,\
                           p_race,p_marital,p_dob])
            dbc.execute ("INSERT INTO events (patient_id, admit)\
                    VALUES (%s,%s)",[p_id,p_admit])
            inserts += 1
        except Exception, e:
            print 'could not parse message:',count
            print '%s' % e
            pass
    f.close()
    os.rename(fname, fname[:-3] + 'old')
    dbc.execute("commit")
    print "Processed file:",fname
    print "Total HL7:",count
    print "Inserted:",inserts

# print elapsed time
t2 = datetime.now()
delta = t2-t1
print "Total elapsed time:",delta.total_seconds(),"seconds."#
```

The following is actual steps process of this project:

- A job scheduler is created in Linux host server that will run python script daily to process HL7 data.
- Python script parse and extract message data using regular expression from flat files into Python objects. HL7 Messages have a limited number of levels. The top level is the message which is delimited by a carriage return. Each message contains a number of segments. Each of the segments are comprised of fields. The content of a field is either a primitive data type (such as a string) or a composite data type comprised of one or more components. Components are in turn comprised of sub-components. Segments/fields that are included in this proposal for analysis are:
    - MSH-7 (Date/Time)
    - MSH-9.2 (Message Type)
    - PID-7.1 (Date of Birth)
    - PID-8 (Gender)
    - PID-11.5 (Zip Code)
    - PID-11.6 (Country Code)
- Using MySQL, objects database are created and they relationships. The parsing result then is stored and indexes created to improve query performance.
- Tableau Desktop software connects to MySQL database and pulls parsing result from the database. We then generate visual representation of the data according to the subjects that we are interested in.
- To run statistical analysis on the data, we utilize SAS as the software choice. Due to licensing issue direct connection from SAS to MySQL cannot be configured. As a work around , CSV format files are dumped from tables and pulled from MySQL. These files then consumed by SAS for analytical processing according to the scripts.

VI. Sample Data

HL7 message sample data:

MSH|^~\&|TEST|TEST|IEPRD|TEST|20151002104146|7196|
ADT^A01|185992044|T|2.3|||AL|||||
EVN|A01|20151002104146||ADT_EVENT|7196^CSHS^ED^
REGISTRARAAA^^^^^CSMC^^^^^MAH|20151002104141|
PID|1||E2528374^^^^ EPI
~200710904^^^GSMRN^GSMRN||CUPCAKE^REDVELVE
T^^^^^D~CUPCAKE^REDVELVET^^^^^L|SMITH^^^|1955
0909|F||White|25 WESTST^^PASADENA^CA^91108^USA^
P^^LOS ANGELES|LOS ANGELES|(333)4445555^P^PH^^^
333^4445555||EN^ENGLISH|MARRIED|NR|7734563|000-00
-0001|||NON-HISPANIC|^^NYC^^^|||||||N||
PV1|1|ED|ER^ IE3
7^01^CAAC^D^^MAH^^^^|ER|||||EMD|||||||49103634192|SE
LF|||||||||||||||||||AD|^^^CAAC^^^^^^^||20151002104000|||||||||PV
2|||||||||||||||||||N|||||||||||||Car||||||||||
AL1|1||^^|||||||||||||||
DG1|1||^ouch|||GT1|1|1229779|CUPCAKE^REDVELVET^^||
25 WEST

ST^^PASADENA^CA^91108^USA^^^LOSANGELES|(333)444-5555^^^^^333^4445555||19550909|F|P/F|SLF|000-00-0001|||||^^^^^USA|||Full||||||||||||||||||||||||||||
UB2|||||||ZCL|||||||||||||N|||

VII. Schedule

First Prototype, October 30, 2015
Final Prototype, December 1, 2015
Submission, December 10, 2015

VIII. Analysis

The system prototype will use one year's worth of HL7 data, representing one year's worth of patient activity at the medical center. The amount of information collected in one year will be substantial, and specific trends and descriptive statistics will most likely be significant. The number of A01 messages that will be analyzed across one year is approximately 150,000.

Statistical analysis to be performed:

- Patient distance from medical center
- Patient demographics (gender, age, zip code)

IX. Real Data Source

The HL7 messages we acquired had the following characteristics. Despite a few problems, the messages were very clean in general.

- Patients admitted between December 2014 and November 2015
- 807 messages could not be parsed my Python hl7 module due to errors in the message formatting issues
- 12,967 messages missing zip codes
- 66 messages missing country code (3-digit ISO code)

X. Data Visualization Distance/Age

After running the Python process into MySQL database, a worldwide view of all patient admissions was created using Tableau (Figure 1). It is immediately apparent that 99.6% of all patient admissions came from within the USA. Honing into the data, we see that within USA, only 8 states have greater than 100 patient admissions, and they are CA, OR, AZ, CO, TX, IL, FL, and NY. In fact, the clear winner is CA at 137,265 patient admissions (Figure 2). We next focus in on CA, and generate a graph ranking the top 30 zip codes and the number of records they contain, in additional to the average distance from the medical center (Figure 3). We visually confirm the rankings by creating a location view of the top 30 zip codes (Figure 4).
Next we create a visualization of the patient age/gender. We highlight a few interesting data points, first showing that there is a spike of patients between zero and one year old, and also a unusual number of female patients between age 25 and 40.

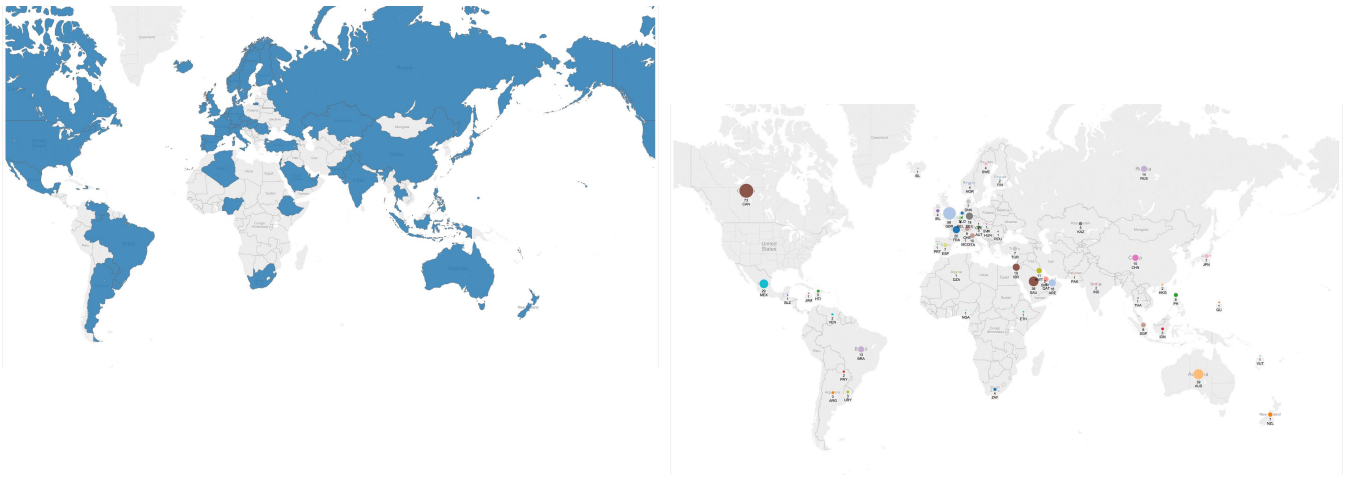


Figure 1. Patient Admissions Count, World View.

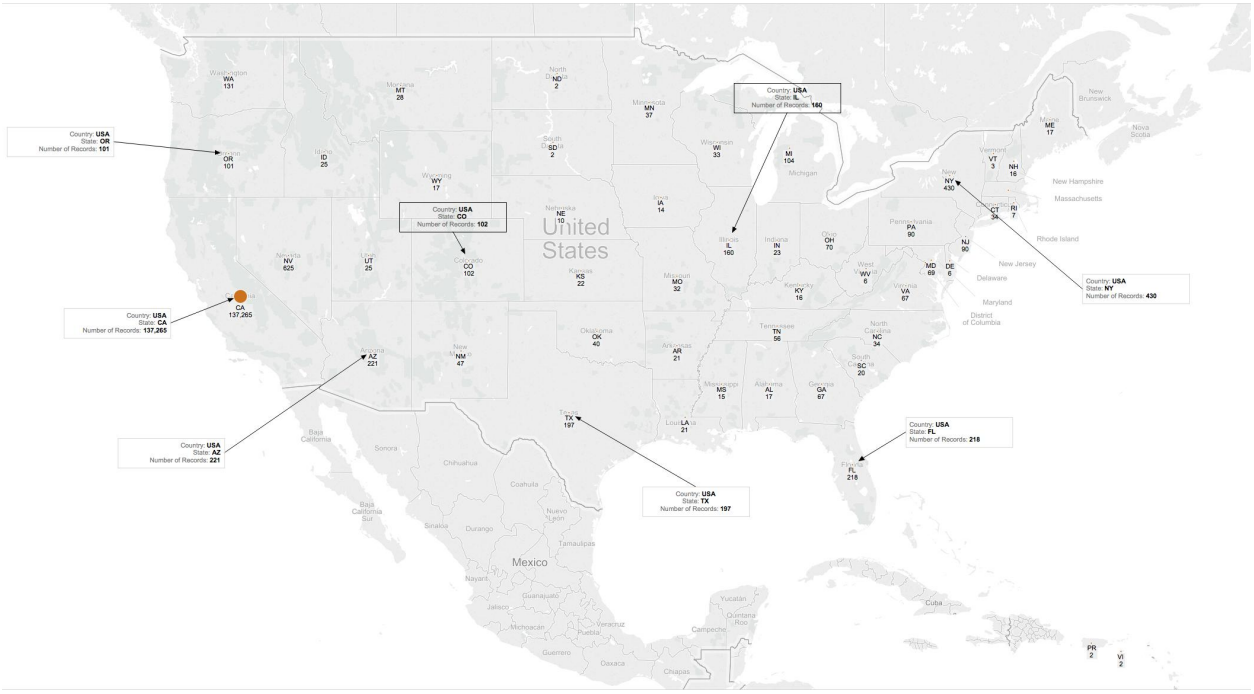


Figure 2. Patients Admissions in Lower 48 States

Rank of Count of Number of Records	State	Zipcode	Distance
1	CA	90046	1.85495
2	CA	90048	0
3	CA	90036	1.42165
4	CA	90035	1.60062
5	CA	90019	2.6015
6	CA	90210	2.12368
7	CA	90069	1.26529
8	CA	90016	3.27653
9	CA	90034	3.29236
10	CA	90024	3.66004
11	CA	90004	3.79609
12	CA	90212	1.84119
13	CA	90008	4.89612
14	CA	90028	3.16687
15	CA	90018	4.42651
16	CA	90043	6.38204
17	CA	90211	0.740993
18	CA	90025	4.54971
19	CA	90064	4.13812
20	CA	90068	3.91722
21	CA	90049	5.82659
22	CA	90038	2.8079
23	CA	91604	4.98336
24	CA	90027	5.52594
25	CA	90230	5.52333
26	CA	90066	5.93704
27	CA	90047	8.7632
28	CA	91423	6.45231
29	CA	90044	9.53017
30	CA	90020	3.76141

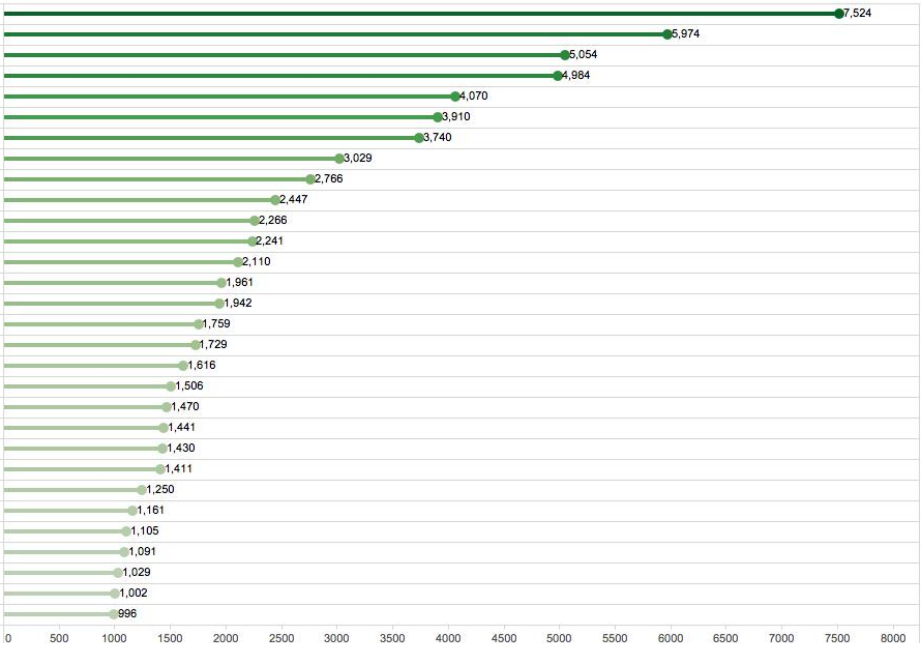


Figure 3. 30 Top Ranked Zips By Patient Admissions

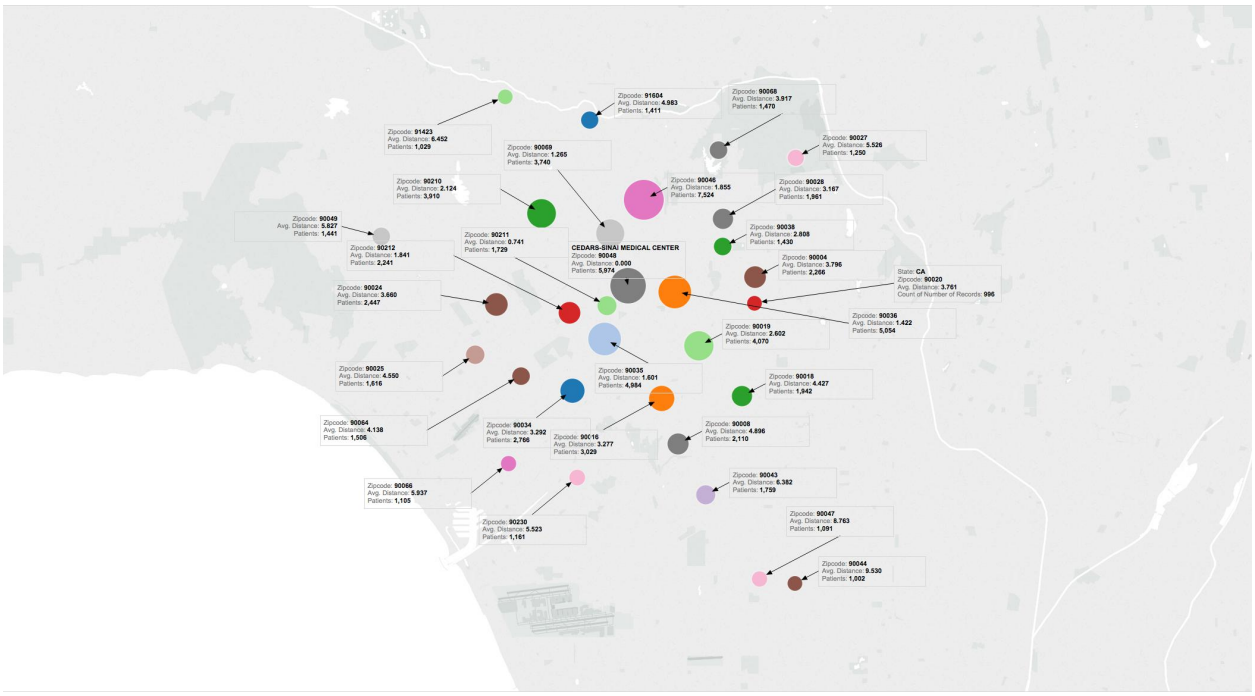


Figure 4. 30 Top Ranked Zips By Patient Admissions

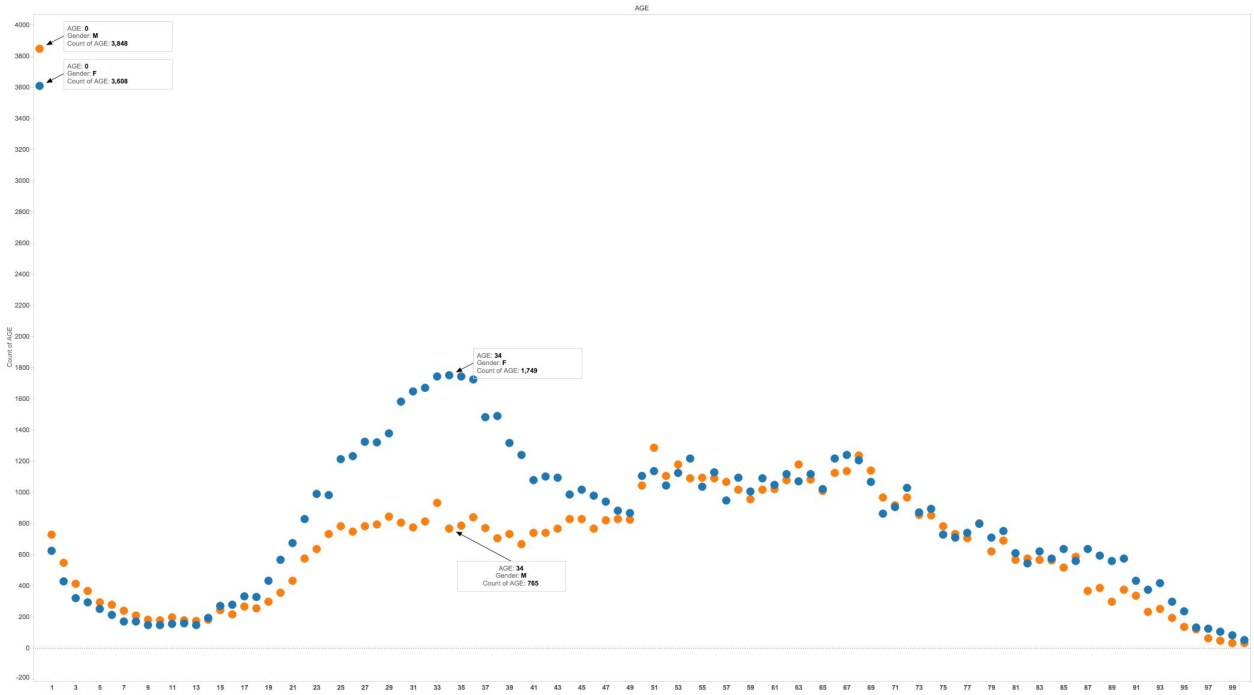
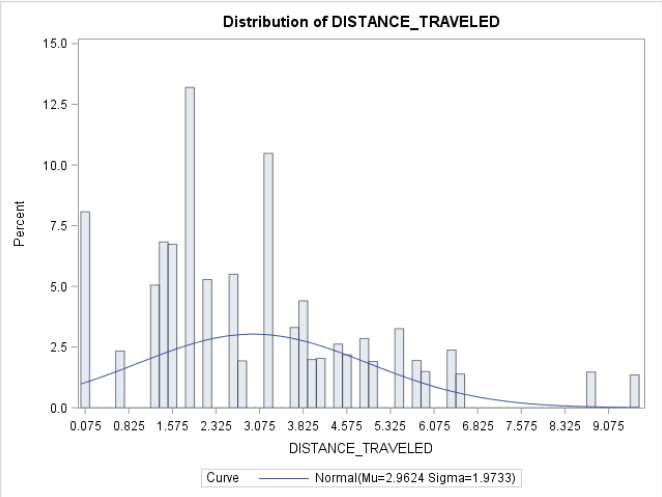


Figure 5. Patient Age/Gender By Count

XI. STATISTICAL CONCLUSION UTILIZING SAS

Distance:

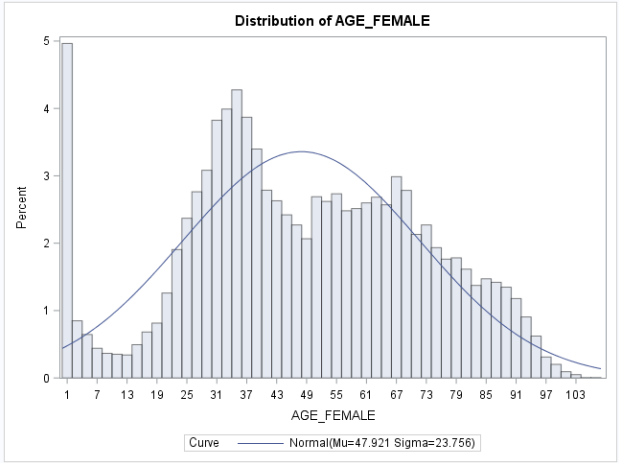


Basic Statistical Measures			
Location		Variability	
Mean	2.962372	Std Deviation	1.97326
Median	2.601500	Variance	3.89375
Mode	1.854950	Range	9.53017
		Interquartile Range	2.31660

Figure 6. Distance Traveled Distribution by Top 30 Zip Codes

- The Top 30 Zip Codes by Patient Admits fall within a 10-mile radius of the Medical Center.
- The Top 30 Zip Codes by Patient Admits accounts for 52.3% of all Patient Admits from 11/2014 to 12/2015.
- For the Top 30 Zip Codes, a typical patient travels within an average radius of 2.96 miles to the Medical Center, with a standard deviation of 1.97 miles.

Patient Age/Female:

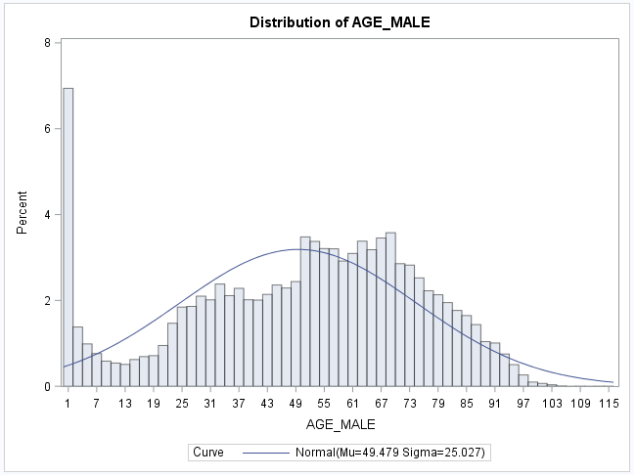


Basic Statistical Measures			
Location		Variability	
Mean	47.92088	Std Deviation	23.75568
Median	47.00000	Variance	564.33212
Mode	0.00000	Range	107.00000
		Interquartile Range	35.00000

Figure 7. Female Age Patients Distribution by Percentage

- The population of female ages does not follow a normal curve exactly.
- There is a large spike of female patients between 0 and 1 year old.
- There is a disproportionate number of female patients between ages 25 to 40.
- The average female patient is 47.9 years old with a standard deviation of 23.76 years.

Patient Age/Male:



Basic Statistical Measures			
Location		Variability	
Mean	49.47865	Std Deviation	25.02726
Median	53.00000	Variance	626.36374
Mode	0.00000	Range	115.00000
		Interquartile Range	36.00000

Figure 8. Male Age Patients Distribution by Percentage

- The population of male ages does not follow a normal curve exactly.
- There is a large spike of male patients between 0 and 1 year old.
- The average male patient is 49.5 years old with a standard deviation of 25.03 years.

SAS Code:

```
DATA MALE;
INFILE 'E:\Uploads\male_fixed.csv' FIRSTOBS=1;
INPUT AGE_MALE;
RUN;

PROC UNIVARIATE DATA=MALE;
    QQPLOT/NORMAL(MU=EST SIGMA=EST);
    HISTOGRAM/NORMAL(MU=EST SIGMA=EST);
RUN;

DATA FEMALE;
INFILE 'E:\Uploads\female_fixed.csv' FIRSTOBS=1;
INPUT AGE_FEMALE;
RUN;

PROC UNIVARIATE DATA=FEMALE;
    QQPLOT/NORMAL(MU=EST SIGMA=EST);
    HISTOGRAM/NORMAL(MU=EST SIGMA=EST);
RUN;
```

XII. Conclusion

Insight into patient demographics trends can potentially improve hospital services. One possible example of this is the ability for the medical center to make wiser choices when making decisions on expanding the medical center services, and focusing on the population trends and choosing locations with the highest patient densities.

References

[1] Health Level Seven Implementation Support Guide for HL7 Standard Version 2.3, Health Level Seven, 1998, pp 4.9–4.10.
[2] Timothy Boronczyk, *Jump Start MySQL*, Sitepoint, 2015, ch. 1-6.
[3] Mark Lutz, *Learning Python 5th Edition*, O'Reilly Media, 2013.
[4] Albert Lukaszewski, PhD, MySQL for Python, Packt Publishing, ch.1-4.
[5] Allan C. Elliot, *SAS Essentials*, Jossey-Bass, ch 1-8
[6] Ben Jones, Communicating Data With Tableau, O'Reilly Media, 2014 ch. 1-6.