PharmaSUG China 2023 - Paper BR-170

Application of real-world data in clinical trial site selection and improving patient recruitment

Jinzhi Zhou, Kaiping Yang, Jie Liu, Xiaojuan Zhang, BeiGene (Beijing) Co., Ltd.

ABSTRACT

Clinical trial site selection and patient recruitment efficacy are important aspects of clinical trials and key factors affecting the success of clinical trials. Real-world data (RWD) collects a large amount of medical information, which can objectively represent distribution of potential eligible patients and provide valuable reference to clinical trial site establishment and patient recruitment.

The purpose of this paper is to investigate how RWD can help clinical trial site selection and patient recruitment. First, we screened RWD from available databases that meet study objectives, including patient demographics, diagnosis, prior lines of therapy, as well as physician professional background, demographics etc., depending on study design. Then a model will be established to identify the most suitable regions, hospitals, and physicians where there are more potentially eligible patients are around after cleaning, integrating, and analyzing these data.

A hotspot map which can reflect the scale of potential patients will also be created. The latitude and longitude of targeted patients and sites obtained from their zip code (for patients), or address (for sites) will be used to calculate the distance between site and patients. After linking potential eligible patients with sites with considering the distance, we can get the number of potential patients around each target site within certain distance (for example: 50 miles). The scale of potential eligible patients will be visualized in a geographic map using hotspot to provide an intuitive proof of site selection.

This practice demonstrates the value and potential use of real-world data in application of site selection and improving patient recruitment. Utilizing real-world data could be an effectively way to locate the right site and patients, which will reduce costs and risks.

INTRODUCTION

Real world data (RWD) refers to data related to patient health status and/or health care services collected in real-world settings. Sources of RWD include electronic health records, medical insurance claims, disease registries, patient surveys, etc. [1] [2]. RWD is different from data from randomized controlled trials, it is observational data that can reflect more heterogeneity and diversity [1] [2]. The evidence obtained from RWD analysis is called real world evidence (RWE), which can provide clinical evidence of the potential benefits or risks of using medical products [3]. RWE plays an increasingly important role in clinical research and health care decision making, and some countries' regulatory agencies also use RWE to support drug evaluation and monitoring [2] [3].

R Markdown is a file format and a tool that allows you to create dynamic documents with R code and text. R Markdown files have the extension .Rmd and can be edited in any text editor or in RStudio, an integrated development environment for R. R Markdown files can be converted into various output formats, such as HTML, PDF, Word, PowerPoint, and more. R Markdown is useful for producing reproducible reports, presentations, dashboards, websites, books, and blogs that combine data analysis, visualization, and narrative. R Markdown can not only execute R code, but also execute code in other languages, such as Python, Bash, SQL, etc.

SQL, which stands for Structured Query Language, is a computer language designed for managing data stored in relational databases. SQL allows users to perform various operations on data, such as querying, manipulating, defining, and controlling access. SQL is a powerful and versatile language that can handle complex and large-scale data analysis tasks.

In this paper, we will explore how the RWD help with the clinical trial site selection and improving patient recruitment and data analysis will be done in R Markdown by R and SQL.

SCREENED RWD FROM DATABASES

First, we will connect to databases and use SQL query RWD that meet study objectives in R Markdown.

Below is an example R Markdown script that enable to connect database and query, analyze data from the two tables: patient, physician.

The R script in R Markdown below can be used to connect the database:

````
---

title: "An example of connecting to databases and using SQL language to operate databases in Markdown"

output: html_document

---

```{r setup, include=FALSE}

# Load the required packages

library(RMySQL)

library(knitr)

# Create a MySQL connection object

con <- dbConnect(RMySQL::MySQL(),

        user = "root", # username

        password = "123456", # password

        dbname = "test", # database name

        host = "localhost") # host name

```
````

and then we can use SQL statements to query and analyze data from the database and display the results in tables. To execute SQL code in R Markdown, we need to use SQL code chunks, which have the following format:

````
```{sql, connection, ...}

# SQL code

```
````

Where `connection` is an R object that represents the connection to the database, we can use different R packages to create this object. `...` represents other options, such as output format, cache, etc. We can write any valid SQL statements in the SQL code chunks, they will be sent to the database and executed, and then return the results.

Here is an example SQL code chunks that enable to query, analyze data from the two tables: patient, physician.

````
```{sql, connection=con}

-- Query all data from the patient table

SELECT * FROM patient;
````
````

```

```{sql, connection=con}

-- Query the total number of patients for each physician

SELECT c.name, SUM(o.number) AS total_number

FROM physician c JOIN patient o ON c.id = o.id

GROUP BY c.id;

```

When data analysis is done, we can close the connection to the database to release resources by use the R script below:

```{r}

dbDisconnect(con)

```

By the steps above we can get patient demographics, diagnosis, prior lines of therapy, as well as physician professional background, demographics etc., and then we will put this information into a model to identify the most suitable regions, hospitals, and physicians where there are more potentially eligible patients are around.

To identify the most suitable hospitals, we need to identify the number of patients that apart from the hospitals at certain distance (like 50 miles). First, we can get the latitude and longitude by zip code or address to calculate the distance between patient and site.

CALCULATE DISTANNCE BETWEEN SITE AND PATIENT

FINDING LATITUDE AND LONGITUDE THROUGH ZIP CODE BY USING *USZIPCODE* PACKAGE

We can use the ***uszipcode*** module in Python to find the latitude and longitude of patients by using zip code. This module provides a ***SearchEngine*** class that can query a database of zip codes and their associated information, such as city, state, population, area, etc.

To use the ***uszipcode*** module, you can install it first by running `pip install uszipcode` in terminal. Then, you can import it in Python script and create a SearchEngine object. Below is an example:

```
from uszipcode import SearchEngine

engine = SearchEngine()
```

To find the latitude and longitude of a zip code, you can use the **by_zipcode** method of the ***SearchEngine*** object. This method takes a zip code as an argument and returns a zipcode object that contains various attributes, including **lat** and **lng**. For example:

```
zipcode = engine.by_zipcode(60007)

print(zipcode.lat, zipcode.lng)
```

This will print the latitude and longitude of the zip code 60007, which is in Elk Grove Village, Illinois. The output will like this:
```

```
42.003918 -87.970346
```

You can also use other methods of the ***SearchEngine*** object to find zip codes base on different information, such as city, state, coordinates, prefix, population, etc. In this way, we can also get the latitude and longitude by address. When we get the latitude and longitude by address and want to check back, we can use ***geopy.geocoders*** to find the address by latitude and longitude.

FINDING ADDRESSES BY LATITUDE AND LONGITUDE USING *GEOPY.GEOCODERS*

geopy.geocoders is a Python module that provides a unified interface to various geocoding services. Geocoding is the process of converting addresses or place names into geographic coordinates (latitude and longitude) or vice versa.

To use ***geopy.geocoders***, you need to install it first, then import a class of a geocoding service, such as ***Nominatim***. Then, you can create an instance of the geocoding service and call its reverse method, passing in a coordinate or a Point object. It will return a Location object, containing attributes such as address, coordinate and raw response. For example:

```
from geopy.geocoders import Nominatim

geolocator = Nominatim(user_agent="my_app")

location = geolocator.reverse("40.7127281, -74.0060152")

print(location.address)

# New York City Hall, 260, Broadway, Civic Center, Manhattan Community Board 1, Manhattan, N
ew York County, New York, 10007, United States
```

If you have a pandas data frame that contains multiple coordinates, you can use the apply method to perform reverse geocoding on each row and store the results in a new column. However, you need to be aware that if the data frame has many rows, it may generate a lot of geocoding requests, which may be throttled or timed out by the service. To avoid this problem, you can use the ***geopy.extra.rate_limiter.RateLimiter*** class to automatically add delays and retry mechanisms. For example:

```
from geopy.geocoders import Nominatim

from geopy.extra.rate_limiter import RateLimiter

import pandas as pd

df = pd.DataFrame({"Lat": [53.352280, 53.352345, 53.352604], "Long": [-6.263668, -6.263758, -6.2
64143]})

geolocator = Nominatim(user_agent="my_app")

reverse = RateLimiter(geolocator.reverse, min_delay_seconds=1)

df["address"] = df.apply(lambda row: reverse((row["Lat"], row["Long"])).address, axis=1)

print(df)

#       Lat     Long    address

# 0  53.352280 -6.263668  O'Connell Street Upper, North City, Dublin 1,...

# 1  53.352345 -6.263758  O'Connell Street Upper, North City, Dublin 1,...

# 2  53.352604 -6.264143  O'Connell Street Upper, North City, Dublin 1,...
```

By the method above we can get the latitude and longitude for both patients and sites then we can calculate the distance between them.

USING *MPU.HAVERSINE_DISTANCE* TO CALCULATE THE DISTANCE BETWEEN TWO POINTS.

The ***mpu.haversine_distance*** is a Python function that calculates the distance between two points on the Earth's surface, using the haversine formula. The haversine formula is a mathematical equation that approximates the great-circle distance, which is the shortest distance along the surface of a sphere. The function takes four arguments: the latitude and longitude of the first point, and the latitude and longitude of the second point. The function returns the distance in kilometers by default, but can also return the distance in other units, such as miles, meters, or nautical miles. The function is part of the ***mpu*** package, which is a collection of utilities for working with geospatial data in Python.

The Python script below can be used to calculate the distance between two points:

```
import mpu

point1 = (48.8567, 2.3508) # Paris

point2 = (40.7033962, -74.2351462) # New York

#Calculate the distance between them in kilometers

distance = mpu.haversine_distance(point1, point2)

print(distance)

5837.009384225264

#Calculate the distance between them in miles

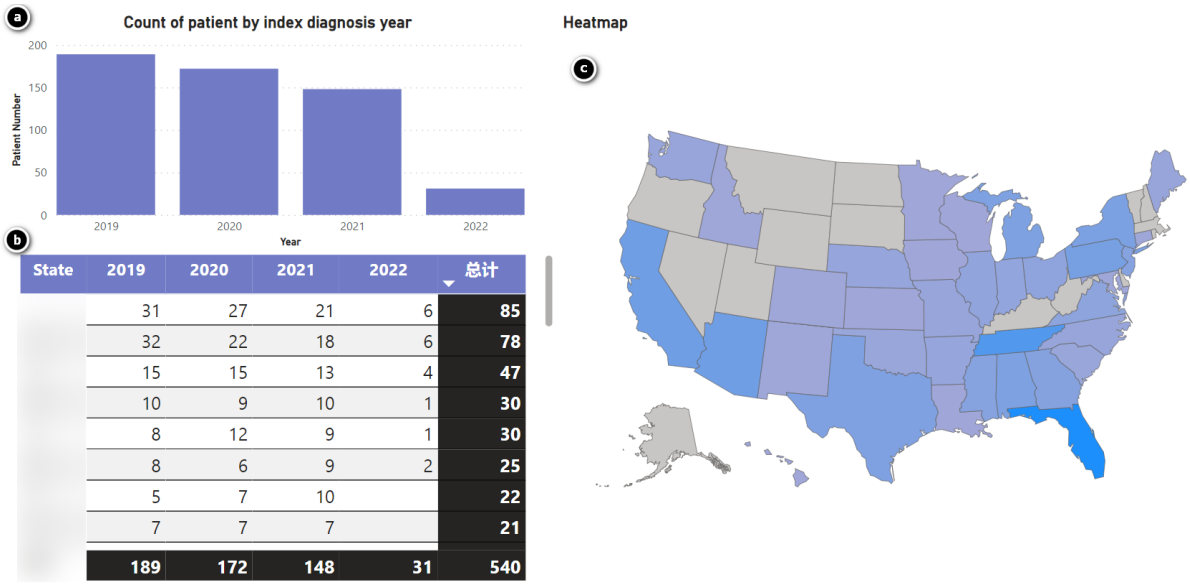
distance = mpu.haversine_distance(point1, point2, "mi")

print(distance)

3626.666509948269
```

DATA MODELING AND VISULAZATION

After cleaning, integrating, and analyzing RWD, we can identify the new diagnosis patient number per year (figure 1) which can support decision making at some scale. Once we get the distance between targeted patients and sites, we can get the number of potential patients around each target site within certain distance (for example: 50 miles) (figure2) which can help with identify the most suitable regions, hospitals where there are more potentially eligible patients are around. The patient number per physician (Figure 3) also can provide a reference for patient recruitment.



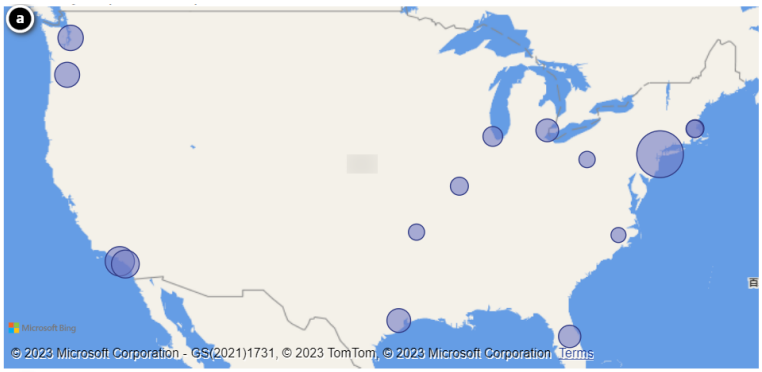
State	2019	2020	2021	2022	总计
	31	27	21	6	85
	32	22	18	6	78
	15	15	13	4	47
	10	9	10	1	30
	8	12	9	1	30
	8	6	9	2	25
	5	7	10		22
	7	7	7		21
	189	172	148	31	540

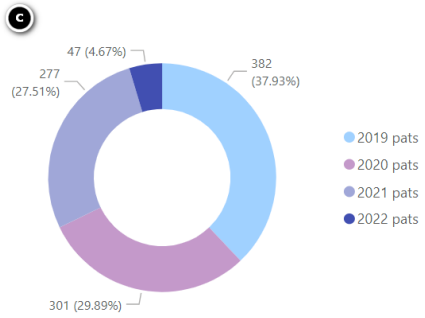
Figure1.New diagnosis patient number per year and patient distribution

a) First diagnosis #patient per year.

b) First diagnosis #patient per year per state.

c) Heatmap for #patient per state





b

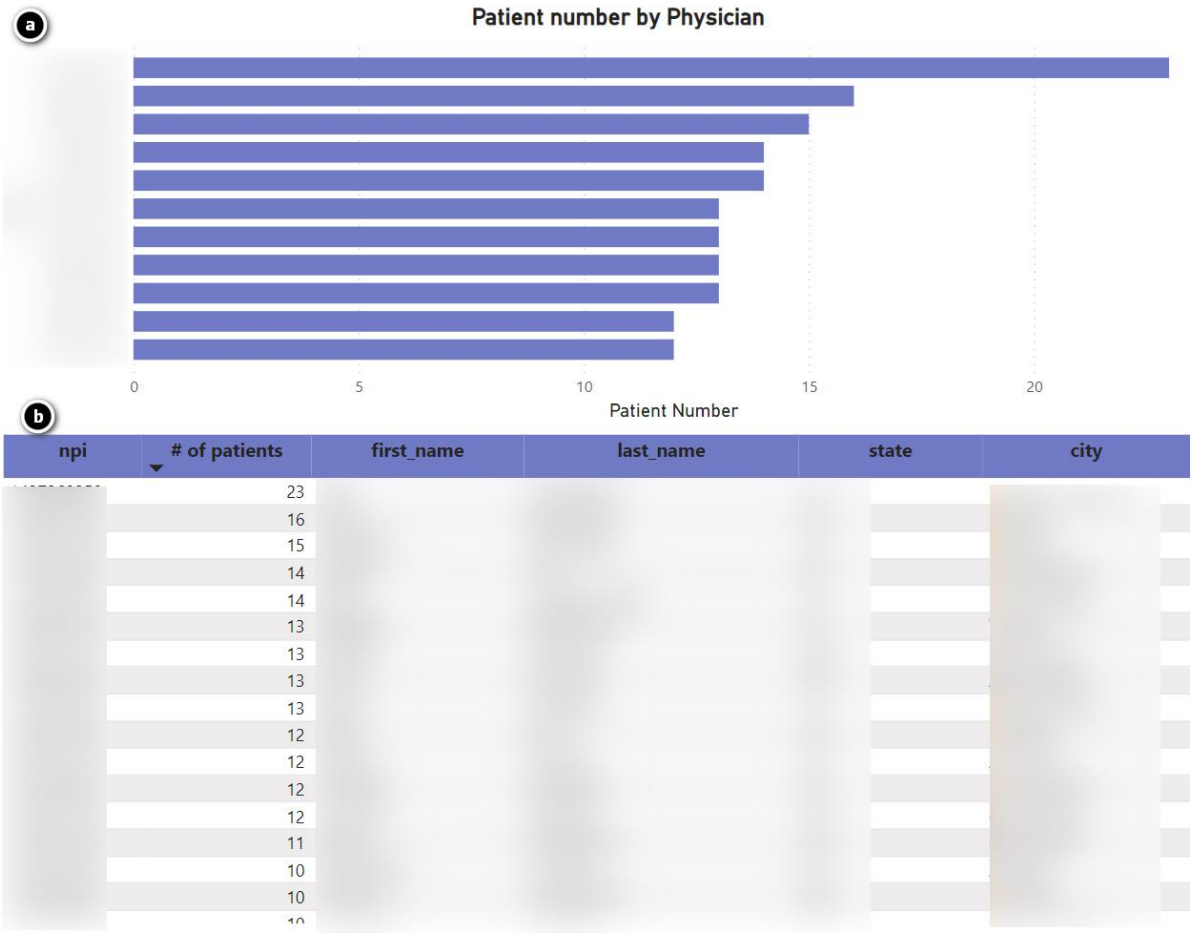
LocationName	pontential pats	2019 pats	2020 pats	2021 pats	2022 pats
	116	28	46	31	11
	105	25	38	32	10
	86	30	32	15	9
	248	97	66	79	6
	83	31	18	28	6

Figure2.Site list and potential patient number around the site

a) Map for potential site. Bubble in the map stands for the location of site and size of the bubble stands for potential #patient.

b) Table for site information. site name, potential #patient will list in the table.

c) potential #patient for site: patient within xx miles from site.



npi	# of patients	first_name	last_name	state	city
	23				
	16				
	15				
	14				
	14				
	13				
	13				
	13				
	13				
	12				
	12				
	12				
	12				
	11				
	10				
	10				

Figure3.Physican list and potential patient number

a) Bar chart for #patient per physician, order by #patient.
b) Table for physician information.

CONCLUSION

Based on the examples in this paper, we could find the value and potential use of real-world data in application of site selection and improving patient recruitment. Utilizing real-world data could be an effectively way to locate the right site and patients, which will reduce costs and risks.

REFERENCES

1. Cowley, Andrea. "What is real world data?". CRC Australia. Clinical Research Corporation. Retrieved 8 May 2018.
2. https://www.cdisc.org/standards/real-world-data
3. https://www.fda.gov/science-research/science-and-research-special-topics/real-world-evidence

ACKNOWLEDGMENTS

Thanks to our managers (Leo Li, Cindy Song and Tony Guo) for their great support on the conference and other colleagues for their comments on this shell script.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Jinzhi Zhou
BeiGene (Beijing) Co., Ltd.
jinzhi.zhou@beigene.com

Jie Liu
BeiGene (Beijing) Co., Ltd.
jie1.liu@beigene.com

Kaiping Yang
BeiGene (Beijing) Co., Ltd.
kaiping.yang@beigene.com

Xiaojuan Zhang
BeiGene (Beijing) Co., Ltd.
xiaojuan.zhang@beigene.com