Integrating MySQL With R for Data Science Research

An Analysis of Made-up but Realistic Insurance Data

Nicholas V. Nikolov
Dept. of Mathematics
Illinois State University
Normal, IL U.S.A.
Ndnikol@ilstu.edu

ABSTRACT

The integration between R and SQL is an important aspect of statistical analyses. Luckily, R offers several packages which can be used to send queries directly from the R IDE so that there is no necessity to access SQL programming software. This package is known as RMySQL which allows integration with MySQL and only requires that the data scientist provide valid login credentials to the database, the database name, the host, and the port.

This package allows the user to type SQL queries, using SQL syntax, in order to extract necessary data. Demonstrated in this project is the ability to perform all queries that are possible in traditional SQL software. A mock database was created in MySQL subject to traditional integrity constraints, then the database was queried through R and a data frame with the important variables was created. These variables were then used as any other variable to create a linear model. The results demonstrate the ease of accessing an SQL database from an external programming source.

1 Traditional Data Science Process-flow

Data Science is the study of abstract numerical or categorical data for the purpose of deriving or discovering coherent and useable conclusions or models. Traditionally, this type of field was simple in that it required the collection of small samples (perhaps a few dozen or a few hundred) which could be used to train a model and make conclusions. These conclusions can determine the difference between profitable corporate decisions or catastrophic failures leading to exorbitant activity in the accounts payable.

Because of the importance of accurate conclusions, it becomes imperative that adequately large and updated datasets become available. The former can theoretically, though with limitations, be achieved by an Excel spreadsheet, but the latter requires more ingenuity. Namely, a database upon which to store and update data.

The way large companies in the modern era operate is through certain automated ways of data collection which may lead to constant updates of a dynamic statistical model. For example, through statistical research, Allstate may have determined that in insuring homeowners, certain variables hold key importance such as the structural integrity of a roof. This variable, among others, is confirmed through the use of aerial drone photography. Now, there are two ways to evaluate image data like this. Either an expert evaluates the quality of the roof based on the images collected or a trained model (these days this is likely a convolutional neural network due to the successful performance) evaluates the images and determines if the patterns indicate some fault. This variable is then used to determine the risk category of the applicant or insured which, in turn, determines the cost to insure. But where does information about an insured client go?

This is where the database comes in. Every new client will have a slew of data associated with them. In our Allstate example the focus would be on the various statuses of the roof, local weather, local crime, etc. Lastly, claim amount is the most important response variable because it allows a statistician to model what variables lead to high claim amounts. For example, if a home is in an area prone to forest fires, they might be more prone to make claims or higher claims. When a claim is made it is immediately updated in the database which allows data scientists to update their models. Data scientists typically create dynamic models that update based on changes in the database which may lead to changes in what variables are deemed important.

One possible occurrence is the emergence of a new variable that can cause an effect on the response variable being studied by the data scientists. Let's consider vehicles now. The introduction of autonomous vehicles may raise the question of whether drivers of such vehicles are more or less prone to accidents. Five years ago there was likely no information on whether or not a vehicle had autonomous driving capabilities but as autonomous vehicles are introduced, omitting this variable may lead to biases in the model coefficients and a subsequent decrease in predictive accuracy.

1.1 Integrating R and MySQL

Several packages in R allow data scientists to seamlessly integrate R and SQL for the purpose of data analytics. The library used for this instance was RMySQL which allows an analyst to access the database through queries made within R. The user needs only to specify the database user, password, name, table, host, and port.

For the sake of this study a toy database was made in a simplified but similar way to how State Farm likely collects data in their Drive Safe and Save program. The official Drive Safe and Save program offered by State Farm utilizes a GPS tracker to monitor various driving qualities such as speed, acceleration, cornering, braking, and even the time of day being driven. These variables are then used to determine if the individual is a safe driver or not. The focus of the database was on the individual drivers, the vehicles driven, and the Drive Safe and Save statistics. The toy database was additionally simplified to only specify averages. For example, instead of an individual having thousands of cornering observations associated with him or her, there is only one cornering average. In theory this is not ideal since it leads to a sacrifice of a lot of information by only providing the average, especially since we cannot develop research such as cornering patterns as a function of time of day or cornering locations.

As mentioned above, the database focused on the three important entities pertaining to driving quality. The driver entity which included a driver ID, first, middle, and last name, age, and years spent driving (the last two being potential variables for a model study. Car data contained the VIN number, make of the car, the model, the year of the car, and two foreign keys referencing the driver entity and the drive safe and save entity which I'll discuss next. The Drive Safe and Save relation is the relation that includes quality of driving statistics. This contains the vehicle VIN number to reference back to the car, the monthly mileage driven, average acceleration, average braking, average cornering, average speed, and the average time of day driven. The acceleration, braking, and cornering are measured in G's where 1G = 9.82 m/s^2 (gravity on Earth's surface).

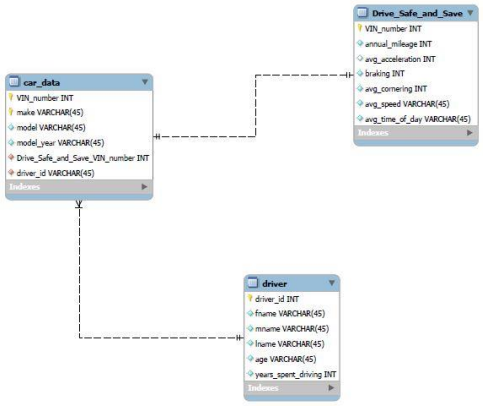

Figure 1: Simple relational schema showing the three main tables of the study.

The database is established in such a way that it is necessary to extract data from multiple relations. This is because we, the data scientist, will need to research how various factors affect claims. However, the Drive Safe and Save statistics are in one relation but claims data is connected to the individual driver. Although not considered for this project, additional data on the vehicle driven may provide additional insight. For example, representing the class of a vehicle as a sports car, luxury sedan, SUV, etc.

When the individual relations are read in we can perform all further actions within R. At this point it becomes straight forward to proceed since by now we have already read in our dataset which is simply convertible to the necessary format. In this case it was necessary to convert the read-in data to a data frame format. This is where predictive models can be used to generate predictions on probability of accident or expected number of claims. For example, we can define the dataset as those that made claims and those that did not, fit a binary model, and predict the probability of an individual making a claim. Then this projected probability can be written back out to the database and when anyone extracts data on a certain client they will be able to determine the probability of claim.

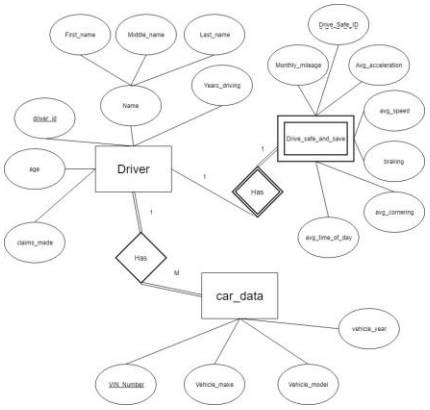

Figure 2: Enhanced entity relation diagram that demonstrates the database in greater detail.

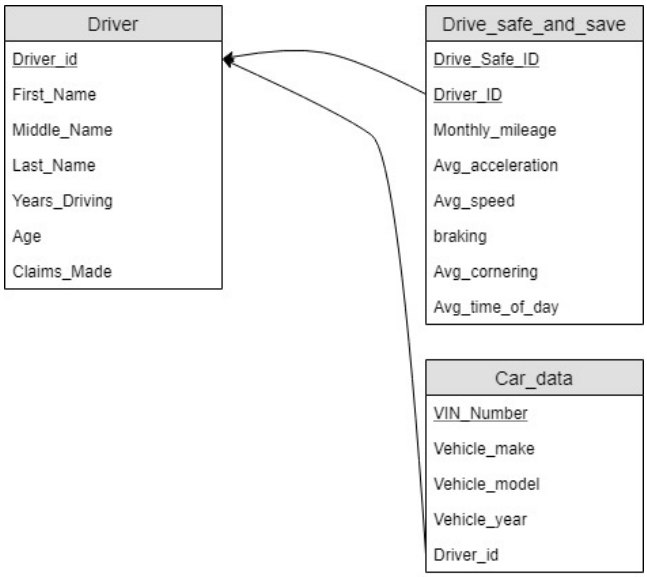

Figure 3: EER to relational mapping of the imaginary State Farm dataset which only excludes the predicted claims column that was added later.

1.2 Results of Application

The results were successful though rather than the binary model mentioned above the model was designed to output a projected number of claims made by a new customer. The new customer was a dangerous driver by the name of Ricky Bobby who, as you may know, was the main character of Talladega Nights. Ricky Bobby really "wants to go fast" and this might be concerning to the insurance company. However, we do not know how poorly he drives. For that reason, we have collected data, with his consent, on his driving habits.

Ricky Bobby's most recent averages in this month demonstrated that he experienced 3.4G's during acceleration (this is immensely fast) and 6.6G's during deceleration meaning he is slamming his brakes. Keep in mind these values are exaggerated. 6.6G's would likely be the result of a direct car crash and the survivability would be called into question. Given that these are average values it's impossible to truly experience 6.6G's unless his first stop was a crash and never drove again for the measured time period.

The model used was a simple regression model which was trained on the data, excluding Ricky Bobby, which included actual claims made. Using this model, and the attributes measured for Ricky Bobby, a prediction was made on the number of claims he was expected to make. Because of his obscene driving habits, Ricky Bobby was expected to make 5 claims in a given period. That's a lot of crashing! Coincidentally, in Talladega Nights he crashed his car at least twice and the film was only 2 hours long.

After the predicted claims was acquired, it was fed into a separate column in the database, under driver, called expected_claims. This updated value can now be seen in the database and the information can be used to determine the Ricky Bobby's premiums.

```
rickyBobbyPred = predict(lmModel,driveSafeData[3,c(3,4,5,6)])


dbSendQuery(database,"update driver set estimatedClaims=4.9 where driver_id=18")
fetch(dbSendQuery(database,"select * from driver"),n=-1)
```

Figure 4: Presenting the code that can be utilized to update the column of estimated claims.

12	Steve	Johnson	Steve	22	13	0	NULL
13	Mike	Mike	Michaels	23	1	4	NULL
14	Luke	Mike	Steve	19	1	7	NULL
15	Mike	Steve	John	55	23	1	NULL
16	Mike	Teve	Ohn	52	25	0	NULL
17	Talladega	Teve	Knight	35	15	9	NULL
18	Ricky	Shakinb...	Bobby	34	29	0	5

Figure 5: Demonstrating the input of estimated claims which is rounded up to 5 for the new customer. This input appears in MySQL's log when selecting all customers in the driver table.

This input is only singular and there is a potential for extension beyond manually predicting claims. It is possible to have automatically processed predictions which can be used for online estimates of premiums on an insurance website. For example, when determining an estimate, the user may input their age, years spent driving, number of accidents, etc. and the model would automatically read into the software that utilizes the predictive model and in turn creates a claims quantity or claims amount prediction.

Although humorously presented with a famous film character, this form of integration between R and SQL is paramount to modern data analytics. Every modern data scientist or statistician is expected to know how to integrate these tools and it is important that they know how to perform queries in their respective programming IDE as opposed to needing to access the SQL database management software.

1.3 Conclusion and Future Research

Successfully integrating R and SQL is integral to any data scientist. With the constantly growing pools of data, databases will become more and more prevalent and being able to work with them will become a necessity.

What we have seen today is the ability to read data in from the database, utilize the data to make predictions, and write the predictions back into the database for the corresponding individual.

What would be useful to study is the automation of this process. The way this system was designed only had the prediction stored as a variable in R and then the data scientist would see it and manually write it into the query. This is not feasible for large datasets which need thousands of predictions at any given time.

Further, it is possible to create dynamically-trained model that retrain at some given interval or with new observations, so they can reflect changes to the customers. For example, if more car manufacturers follow Tesla, Inc.'s lead and develop self-driving cars, it is possible that there will be a shift in aggressive driving. Our model was designed with aggressive driving in mind. In a world without steering wheels (one of Elon Musk's bold ideas) this model would be inaccurate and would unfairly punish drivers.

Further automation of the above process may also be detrimental with modern cyber threats. More research from statisticians is appearing that focuses on adversarial statistical modeling that focuses on defending the ability of a model to adequately predict when observing slightly modified images.

For example, the MNIST dataset is a famous image-classification problem which now serves as a sort of "Hello World" data project for analysts entering the field of image classification. A convolutional neural network trained to predict based on the images it sees has been shown to suffer from slight alterations at the locations where the convolutional neurons are trained to "see." This sight is based on a square that may be as small as 2x2 or 3x3. What cyber attackers can do is change the color of a few of these squares (only a small count of pixels) to match the background. The result ends up being that the neurons no longer fire because they aren't activated by the expected color. In this way, we can start to misclassify 8's into 6's.

Adversarial machine learning utilizes game theory research to attempt to model the competition between attacker and defender. The goal is to protect dense clusters of data from minor differences. This is straight-forward to grasp in the realm of cluster analysis, one of the most famous unsupervised learning techniques. The defender might be able to specify a classification region that is much smaller than would otherwise be specified. Outside of this region is where the slightly altered datapoints may lay. The cost is inevitably a loss of predictive accuracy but also an added layer of defense in the event of a cyber attack.

This may further be a concern with active image classification for autonomous vehicles which can be fooled by a sticker on an important road sign which can lead to an inability to adequately classify the sign. This is especially concerning if it is something like a stop sign. This was even demonstrated with a model capable of detecting and outline humans but could not when a small, printer sized, paper with various shapes was held to the camera.

With dynamic models, the erroneous data can be fed back to the database and automatically re-extracted. This would then lead to an incorrectly retrained model.

The last concern is explicit cyber-attacks. If a cyber attack occurs during an automatic data-extract and the dynamic model is retrained, then the model may fail to adequately predict what it was designed to predict. The automatic retraining of a model would be subsequently erroneous, and the cyber attacker could take advantage of the changed design to financially exploit the company.

Ultimately, determining the type and frequency of access into a database is variable and dependent on the application. Some organizations may find that it is necessary to automate a predictive modeling process while others find that there are severe risks associated with such a model. Whatever direction is pursued, it is necessary to approach the problem with a conceptually thoughtful mindset. Convenience is an exceptional goal, but not at severe cost.

REFERENCES

[1] Ooms, James, DebRoy, Wickham, Horner (2019). RMySQL: Database
[2] Haung, Joseph, Nelson, Rubinstein, Tygar (2011). Adversarial Machine Learning
[3] Boyd, Keromytis (2004). Preventing SQL Injection Attacks