Henry Nguyen

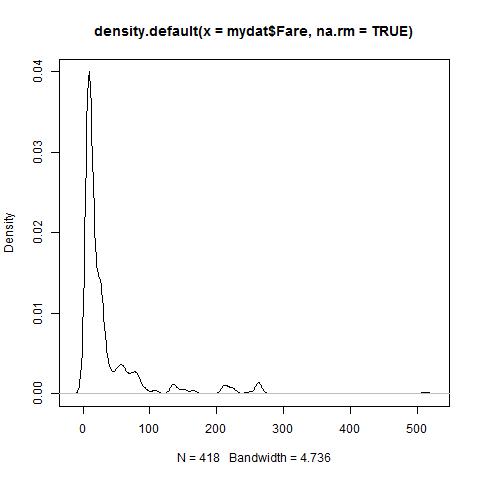
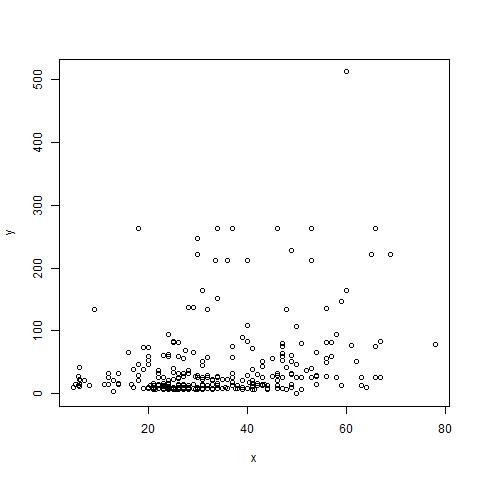
COMP 331

April 8 2017

Project

**Missing Data Imputation**

* We first use the read.csv() function to load in the test.csv data set (link in next page)
* We use linear regression using the lm() function to model the relationship between the dependent variable “Fare”, and the rest are the independent variables: “Pclass”, “Age”, “SibSp”, “Parch”; we also assign the function to a variable named “linmod”.
  + Sex and Embarked are character types and are not included.
* After looking at the data set, we find that “Fare” has a missing record, so we use the is.na() function and assign it to a variable named “missing”.
* The is.na() function returns a true value on one of the record (index 153), so we predict the value using the predict() function.
* We pass the arguments, “linmod” and “missing” into the predict() function and assign it to a variable named “predictFare”.
* The predict() function returns 9.160766 on the index 153 for the missing value.
* We define the impute function, and pass the arguments “mydat$Fare”, which is the input values of the variable “Fare” from the test data set, and “predictFare”, which is the predict function we defined.
* We then assign the impute function, with its arguments, into a variable named “newFare”.
* We then assign “newFare” into the input data set “mydat[“Fare”]”.
* Our linear regression model has the coefficients:
  + Intercept = 115.8285; Pclass = -39.2210; Age = 0.3713; SibSp = 5.7929; Parch = 16.6186
* Where we assign the independent variables to “x” and “Fare” to a variable named “y”.
* We then produce a scatterplot and a kernel density estimate graph:



**Data Merging**

* We merge the test data set and train data set
* Since the two are similar, in which the test set does not provide ground truth for each passenger, whereas the training set already has the outcome/ground truth for each passenger.
* We use the merge function using the two data sets and sort by “Name”, and assign it to “m1”.
* To make the table smaller and easier to read, we assign null values to the following columns: PassengerId, Sex, Age, SibSp, Parch, Ticket, Cabin, and Embarked.
* We then use the unique() function with the argument “m1”, to avoid duplicates, and assign it to “uniqueName”
* With the order by name, we can see x (test) and y (train) for Pclass and Fare, in which there isn’t any difference between the two data sets.

**Set-Based Metrics: The Jaccard Metric**

* Using the same two data sets, we can use it find the similarity and difference between samples.
* We first define the jaccard function:
  + Assign function() that takes two arguments: the first string, and the second string.
  + We use the paste() function to add in # symbols at the start and end of the string and remove the space between names. We do this for both strings and assign them to “s1” and “s2”.
  + Create vectors for the strings by assigning c() to “s1vec” and “s2vec”
  + Count their characters using the nchar() function that takes in a string argument, and assign it to “s1length” and “s2length”.
  + Put it through a for loop that iterates from index 2 to the length of the string -1, then takes the substring and add it to the vector for both of the strings.
  + Calculate the intersect and union score with the intersect() function and the union() function, where each takes two arguments, “s1vec” and “s2vec”.
  + Take the length of the intersect and union using the length() function, where the length of the intersect divided by length of the union is assigned to a variable named “Score”, then we return “Score” and finish defining the function.
* We then call the function by using jaccard(string 1, string 2), where string 1 and string 2 are the strings we want to compare.
* For example, jaccard(test$Name[1], train$Name[1]) returns the bi-grams of the strings and the Jaccard measure, which in this case is 0.1714286.

# References

Kaggle. (2014). *Titanic data*. Retrieved from Titanic: Machine Learning from Disaster: https://www.kaggle.com/c/titanic/data