CS 251: Statistical Computing: R (Outlab)

- Due: 11:55 AM 24/09
- Please write (only if true) the honor code. If you used any source (person or thing) explicitly state it. You can find the honor code on the web page.

Overview

In the inlab you worked with some of the basics of R. Here we turn to some "real world" issues that crop up.

A. Broken-line Regression

Note: No package is required for this. Do not use any extra package.

Linear Regression works nicely if the data was nearly linear to begin with. That was the case with `data_lin.csv`, and you got a pretty good linear fit. But linear regression doesn't fare so well when used on non-linear data. In the inlab we tried to model a polynomial to fit our data.

Another technique is to approximate the data with a set of lines instead of just one. For this exercise, we'll try to use a two line fit, also known as Broken-line Regression. A simple idea that might come to you is to sort the data on x co-ordinate, bucket them into a left half and right half and run linear regression independently. What could go wrong?

Here's a formulation. Let's say that C is that point where we need the line to change its direction. For the points in the left bin we need to fit in a linear regression, and then add some additional terms which is again linear in those x in the right bin:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 (x_i - C)^+ + \varepsilon_i \qquad i = 1, \ldots, n,$$

$(x_i - C)^+$ is a derived variable which takes the value of 0 for values of x_i less than C and the values $x_i - C$ for values of x_i greater than or equal to C. You can work it out for a small case, and you'll see that you end up generating two lines.

But the question is that what it the change-point C? That in itself is another problem studied widely. The point of split can be anywhere in the x space, but one way to find it is by iterating over each of the x_is. Write a script `taskA.R` to achieve the following

0. Read from `data_bi_lin.csv`, and run a simple linear fit on it. Print the error and plot the line
1. Now use the above formula and run a broken line regression on this data. Choose the value of C from $x_i s$ such that the error is minimized. Print the new error and the value of C, and also plot the broken line on the same plot
2. Your final output being printed should look something like

```
error lm = 34
error broken = 34
C = 1.3
```

Copy the output into `outA.txt`, save the plot as `taskA.png` and sumbit `taskA.R, outA.txt, taskA.png`.

B: Outlier Removal

When you made the plots in the inlab or in taskA, you might have observed some outliers that affects your fit significantly. They have been manually introduced here, but in real life data, they occur naturally due to instrumentation error, anomalous events and human errors. Such outliers disrupt our model, sometimes significantly. So in order to identify these outliers, we will use the **Local Outlier Factor** algorithm.
This algorithm uses some of the neighbors of each point, calculates the density, and computes a dissimilarity metric. The higher this metric is, the higher are the chances that the point is very different from its neighbors, making it an outlier. You can read more about this algorithm here.

We are going to code this algorithm. Instead, here's a hint for you: There is a function in some R library which computes the dissimilarity metric for you.

Write a script `taskB.R` to achieve the following

0. Load data from `data_non_lin.csv`
1. Learn how to include libraries in you script and use their functions. Include the `Rlof` library to your script.
2. Calculate the LOF metric for each of the points. Use 5 neighbors for calculation of the metric. Have a look at the vector. You'll see that the outliers are clearly separated out.
3. Decide a good threshold for the cutoff. Keep in mind, that if you keep it too low, you might discard good points, and if you keep it too high, it might not work for some other data with different outliers (This is a parameter tuning problem in some sense. What you can expect from us is that we'll use "similar" data to test your program on.)
4. Once you have the outliers, remove them from you matrix, and redo the broken line regression problem (outlab) and the polynomial regression (inlab) on the new data
5. Plot a single graph, with the outliers shown in red circles instead of black (just for display, they're not used in the calculation). Show both the broken line and the polynomial curve.
6. You script should print the outliers(x,y), and the errors of the new fits.
7. Your final output being printed should look something like

```
error brokenline = 34
error polynomial = 34
outliers
1 2
3 4
2 5
```

Copy the output into `outB.txt`, save the plot as `taskB.png` and sumbit `taskB.R, outB.txt and taskB.png`.

C. Hypothesis Testing

A hypothesis test is a statistical test that is used to determine whether there is enough evidence in a sample of data to infer that a certain condition is true for the entire population. A hypothesis test examines two opposing hypotheses about a population: the null hypothesis and the alternative hypothesis.

- **Null Hypothesis (H_0)** : The null hypothesis states that a population parameter is equal to a value. The null hypothesis is often an initial claim that researchers specify using previous research or knowledge.

- **Alternative Hypothesis (H_1)** : The alternative hypothesis states that the population parameter is different than the value of the population parameter in the null hypothesis. The alternative hypothesis is what you might believe to be true or hope to prove true.

Under some statistical assumptions about the population, decide an appropriate test statistic, whose distribution is known. Decide a significance level (α) – a probability threshold below which the null hypothesis will be rejected. How is this related to the confidence intervals?

So the test kind of boils down to computing the value of the statistic from the observations, calculating the p-value and then rejecting the null hypothesis if it is less than the significance level. But the tough question is that which statistic to use in a given situation?

t-Test

This test is useful if you know that the population follows normal distribution, but you have no idea about the mean or the standard deviation. Your friend tells you that the mean is μ. You need to test his claim. What is the null hypothesis here? You take a random sample from the population and perform a standard test - the Student's t-test. You can read more about it here.
R has many datasets available. One of them is `PlantGrowth`. In order to do the experiment you sample randomly some plants and that is available in the column `weight`.

0. Run the t-test and report whether you'll accept the hypothesis that mean weight of the plants is 4.76 under the confidence level : 99, 95 and 90%. Report a "Yes" or "No" for each of them. Also provide the respective confidence intervals.

chi-squared test

The Chi-square test is intended to test how likely it is that an observed distribution is due to chance. It is also called a "goodness of fit" statistic, because it measures how well the observed distribution of data fits with the distribution that is expected if the variables are independent. A Chi-square test is designed to analyze categorical data. That means that the data has been counted and divided into categories.

Professor HareRam Bumblefuss takes a random sample of students enrolled in CS 989. He finds the following: there are 25 freshman in the sample, 32 sophomores, 15 thirdies and 20 fourthies. Test the null hypothesis that freshman, sophomores, thirdies and fourthies are equally represented among students signed up for CS 989.

0. Test Prof. Bumblefuss's hypothesis using the chi squared test and report whether you'll accept the value under confidence levels : 99, 95, 90%. Report a "Yes" or "No" for each of them.

Professor Iconoclast argues that Bumblefuss is wrong, and that the number of freshman and sophomores enrolled is each twice the number of juniors and the number of seniors.

1. Test Iconoclast's hypothesis from these same data as above and report a "Yes" or "No"for each confidence value.

You may use the R console to do this task. For each sub task report the answer along with the commands used in `outC.txt` and submit `outC.txt`.

D. Statistical Analysis in Ranking

In life, data isn't tagged with its distribution, and problems aren't tagged with their solution. We have to try out a bunch of things, do a bit of analysis before we can settle on something that "seems" correct.

We have with us a dataset of mutual fund companies with their performance (100 is the best) and we're gonna rank the companies. (The short scoop on mutual fund companies: Stocks are considered

better investment than, say, fixed deposits since they beat inflation. However stocks go up and down, and it's hard to predict which one will win. To spread the risk, a mutual fund company enters the scene. It buys stocks on behalf of its customers, but it buys lots of stocks in lots of companies because it has lots of customers each with a different risk mindset. The company makes some profit in between by taking a portion of the customer's profit.)

Our goal is to rank the Mutual Fund companies, and to output the top X performers. This is very important since new investors will go to the best company (aka IIT Bombay Mutual Fund Inc), so how to do the ranking?

You can download the dataset from the usual place. Each row is a company. Each column contains performance. One cell can contain more than one score, or no score at all. The columns are in sets of 4, one each for passive Index funds, or Common Market (CM), Hot Fund (HF) (this is the flagship fund for the company), Bonds (PG) and Emerging Market (EM). Each set is for six months. Data starts in 2006 June, and there should be 20 sets of data.

Before ranking, you might need to clean up the data and make it more "usable". This is a very common nuisance since the supplier of data just puts out the data in whatever is convenient to him.

Write a script `taskD.R` to achieve different types of ranking:

1. Load the data in R. You'll get a data frame with a whole bunch of "NA"s, that correspond to missing data. For cells containing multiple values, use each of them independently.

2. We want to standardize each column. For this, you have to subtract the mean from each entry, and divide with the standard deviation. It the data originated from a normal distribution, what we have done has made it a standard normal sample.

3. For easy readability of final rankings, add 6 to the score and multiply by 10. (Why?). This is the standardized score for the company in the column of interest.

4. For the first algorithm, we want to score each candidate based on the average rating over all the years that they've been in business. Don't use the NA values in the count for averaging. Report the row number of the top X=15 performers with their scores in descending order.

5. This doesn't seem fair to the companies, who might have had a bad year or two, and a bunch of good years, or to the companies who have improved over the years. So we want to consider the best-n normalized scores for each prof, and rank over the sum of these values. Count the number of inversions from the previous ranking to this one. An inversion is when one company is above the other company in one ranking, and below in the second. Choose n=15, X=15. Also choose n=10, and X=15.

6. We're binning the funds based on 4 categories. But there are cases, in which there is intermixing of stocks, like when common stocks get merged with Hot stocks. So the performance numbers are also mixed up. Instead of binning, what if we standardize 4 columns at once by considering all the data at once. Do that, and find the ranking using the best-n algorithm. Report the number of inversions from the previous ranking. Choose n=15, X=15. Also choose n=10, and X=15.

7. We have been standardizing the data and using it, but we never tested if that is a sane assumption. If the data is not from a "near-normal" distribution, the normalized values don't give us any useful information. We're gonna run a test on the data, and check if the data is normal or not. One of the standard tests for this is the Shapiro-Wilk Test. You can read up on that here. Run this test on every column, and report what fraction is not normally distributed. Also report this number if you run this on groups of 4 columns, as in the last algorithm.

8. (Extra Credit): If the Shapiro-Wilk test fails in any case, repeat steps above by using a power transformation $y = f(x)$ with a suitable value of λ (Report it) in case the normal distribution test does not pass. If it does pass, do nothing. If it does not pass, and if the power transform

does not work (i.e. it does not pass the normal distribution test), do it again, and keep doing it till it does pass. If it doesn't pass at all, give up and use the original data. Report statistics for all these different cases.

$$y = \begin{cases} \frac{x^\lambda - 1}{\lambda}, & \lambda \neq 0 \\ \log(x) & \lambda = 0 \end{cases}$$

Choose n and X as before, and as before report the number of inversions.

Important note: Summarize your findings by writing a Beamer document(**report.pdf**) with a nice table which summarizes your finding and executive summary. Also, output a `rankings.txt` file from your R script, with the three sets of rankings in three columns.

Submission Guidelines

(Stay tuned for clarifications on Piazza)

1. When you submit, please document individual percentages such as Student 1: 80%, Student 2:100%, Student 3:10% in the readme.txt. In this example, the second student will get full marks (10/10) and the first student will receive 8/10.

2. Do include a readme.txt (telling us whatever you want to tell me). (The reflection essay is not required for inlab, but is required for outlab.) Do include group members (name, roll number), group number, honor code, citations etc.

3. The folder and its compressed version should both be named `lab07_groupXY_outlab` for example folder should be named `lab07_group07_outlab` and the related `tar.gz` should be named `lab07_group07_outlab.tar.gz`

4. Your submission folder should look something like this:

```
lab07_groupXY_outlab
├── taskA.R
├── taskA.png
├── outA.txt
├── taskB.R
├── taskB.png
├── outB.txt
├── outC.txt
├── taskD.R
├── report.pdf
├── rankings.txt
└── readme.txt
```

How We will Grade You [70 Marks]

Task A and task B will be testes on a different dataset as in inlab

1. Task A [**10 Marks**]
   - `taskA.png` : 2 Marks
   - `outA.txt` : 2 Marks
   - `Hidden dataset` : 6 Marks
2. Task B [**5 Marks**]

- `taskB.png` : 1 Marks
- `outB.txt` : 1 Marks
- `Hidden dataset` : 3 Marks

3. Task C [**5 Marks**]
    - t-Test : 2 Marks
    - chi-squared test : 1 + 2 Marks
4. Task D [**50 Marks + 10 Marks**]
    - Ranking 1 : 10 Marks
    - Ranking 2 : 10 Marks
    - Ranking 3 : 10 Marks
    - `report.pdf` : 20 Marks
    - (**Extra Credits**) All Required Statistics : 10 Marks