Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: 12 Dec 2017
Manuscript Category

CSE:549 Computational Biology

Uncertainty Principle

Aditya Srivastava , Aman Agarwal and Mahima Parashar

Department of Computer Science, Stony Brook University, New York, 11790, USA

Abstract

In an attempt to provide some measure of confidence in the estimates of transcript abundance by Salmon, bootstrapping tends to underestimate the uncertainty. This paper aims to determine the specific properties of the transcript which leads to this failure and define a quality score which is indicative of the level of confidence in the transcript-level estimates. Regression models are used to classify a given transcript as good or faulty and adjust the counts so that they lie in the maximum likelihood of true counts. After training the models, the number of faulty transcripts reduced from 47.27% to 35.1% (for poly$_{mo}$).

Keywords: $Salmon, Bootstrapping, Regression, RNA,$

1 Introduction

Gene expression is the process by which information from a gene is used in the synthesis of a functional gene product. These products are often proteins, but in non-protein coding genes such as transfer RNA (tRNA) or small nuclear RNA (snRNA) genes, the product is a functional RNA.

Measuring gene expression is an important part of many life sciences, as the ability to quantify the level at which a particular gene is expressed within a cell, tissue or organism can provide a lot of valuable information. For example, measuring gene expression can:

- Identify viral infection of a cell (viral protein expression).
- Determine an individual's susceptibility to cancer (oncogene expression).
- Find if a bacterium is resistant to penicillin (beta-lactamase expression).

Salmon, a lightweight method for measuring gene expression (quantifying transcript abundance from RNA sequence reads). Salmon combines a new dual-phase parallel inference algorithm and feature-rich bias models with an ultra-fast read mapping procedure. Accurate and efficient quantification of transcript abundance from RNA–seq data is an especially pressing problem owing to the wide range of technical biases that affect the RNA sequence fragmentation, amplification, and sequencing process, the exponentially increasing number of experiments, and the adoption of expression data for medical diagnosis.

Salmon determines expression levels by solving a maximum likelihood problem. A result of this formulation is that one often gets accurate estimates of the transcript abundances, but has no notion of confidence in these predictions. That is, predictions can be highly specific and highly accurate, or highly specific and highly inaccurate – this depends on the *shape* of the likelihood function, and how optimization proceeds. (In statistics, a likelihood function (often simply the likelihood) is a function of the parameters of a statistical model given data. Likelihood functions play a key role in statistical inference, especially methods of estimating a parameter from a set of statistics. In informal contexts, "likelihood" is often used as a synonym for "probability." In statistics, a distinction is made, depending on the roles of outcomes vs. parameters. Probability is used before data are available to describe plausibility of a future outcome, given a value for the parameter. Likelihood is used after data are available to describe plausibility of a parameter value.) It is very valuable to provide some measure of confidence in the estimates that are inferred. There are a number of ways to estimate such confidence. One way is *bootstrapping*. This approach treats the observed sample data as the population, samples from the original data a number of times, and reruns the maximum likelihood estimator independently on all of these samples. By looking at the distribution of these different runs, one can form an empirical confidence interval, that provides a notion of the uncertainty of the maximum likelihood point estimate. However, a close inspection on simulated data demonstrates that these empirical intervals fail to capture the uncertainty adequately (i.e., they tend to underestimate the uncertainty).

The goal here is to determine what are the specific properties, if any, of the transcripts that tend to fall outside the predicted uncertainty interval, using sequence similarity or transcript quantification estimates or other variables output by Salmon during the bootstrapping phase. Further, we come up with a quality score based on these properties. This score will reflect the level of confidence we have in the transcript-level estimates.

2 Approach

Given a set of features, true counts and the bootstrapping outputs corresponding to each transcript, the first step is to classify a transcript as good or faulty. A good transcript is one in which the deviation of bootstrap output's mean is with 95 percentile of it's true count whereas a transcript is faulty if it's bootstrap mean does not lie in 95 percentile of its true count. Once this classification is done, the features of the failing transcripts are analyzed so as to see what are the specific properties of a transcript which leads to its failure. Now, because the goal is to correct the bootstrap counts of each failing transcript, the mean of the failing transcripts should be shifted. The result of this shift would lead the counts of failing transcripts to fall within 95 percentile of true counts. Thus, the value of this shift is predicted by the training a regression model and the transcripts which are faulty are shifted by this predicted value. Finally, the accuracy is calculated which represents the percentage of the good transcripts.

3 Method

Phase-1: In the first phase of the project, for given samples, poly_ro and poly_mo, the transcripts are segregated in two classes: good and faulty. The following method is performed to separate the transcripts:

The given file $quant\_bootstrap.csv$ is the output file of the bootstrap process. Each column of the file represents a transcript and each subsequent row represents the result of a bootstrap run. So, the mean and standard deviation is calculated for each transcript from this file. Next the standard deviation is compared with its true count value from the values present in $poly\_truth.csv$. This gave the deviation of true count from mean value for each transcript. Finally, the transcripts are segregated into two classes: âĽ˜goodâĽ™ which had deviation between 2 and -2 and âĽ˜faultyâĽ™ otherwise.

Once this classification is done, the feature set of the failing transcripts are took into consideration and the features of those transcripts are studied due to which this failure occurs which is performed as follows:

1. First step in this direction is to perform feature reduction. Correlation is calculated among the four properties which are Length, Effective Length, NumReads and TPM and found that NumReads and TPM are highly correlated and so are Effective Length and Length. Thus, now the 4 features reduces to 2 features because of the high correlation.
2. Logistic regression model is used to predict which property highly contributes to a transcript being good or faulty. The features fed in this system are effective length and TPM.
3. The results obtained from regression model were that TPM contributes more to a transcript being good or faulty.
4. To compare good and faulty transcripts, we plotted scatter graph between their TPM and deviation. Next, we tried to look for any anomalous pattern among faulty transcripts.
5. Similarly, step 4 is repeated for effective length.

Thus in this phase, classification of a transcript is performed and the features of the failing transcripts are analyzed.

Phase 2: As a transcript is classified as good or faulty, the next step is to make the faulty transcripts good by shifting there mean by some value so that the mean value of the transcript now lies in the 95 percentile of their true counts. The steps involved in this phase are as described below:

1. Till now only two features were fed into the regression model, and the features from equivalence classes were not yet considered. Now a feature from equivalence class is extracted, which is as follows:
   eq_param(of each transcript) = average of the number of times a transcript appears in an equivalence class.

   Thus, eq_param(transcript) = $\frac{\sum size(equivalenceclassitappearsin)}{numberoftimesitappears}$

   This feature will represent the extent to which a given transcript is ambiguous. As seen in phase 1, the more deviated a transcript's mean is from it's true count, the more ambiguous the transcript is. Thus, the feature derived above will give higher value to the more ambiguous transcripts and a smaller number to a less ambiguous transcript.

   One binary feature named faulty is also added in the set of features which is 0 for a faulty transcript and 1 for a good transcript.
2. Now the data is divided into training data and cross validation data and the ratio in which they are divided is 7:3.
3. The training data is fed into 3 regression models i.e. Lasso Regression, Ridge Regression and Linear Regression, and the target value is kept as the difference between the mean and true counts of each transcripts.
4. When the training is done, in the testing phase, the test data is given to the model and the predicted value is set as the adjusting factor.
5. For each faulty transcript, this adjusting factor is subtracted from each bootstrap counts.

4 Implementation

This project is implemented in python on jupyter. The libraries used are pandas, scikit-learn, numpy and mathplotlib. These libraries have helped implementing the various training models used in the project and other machine learning tools. The flow of the project is as described below:

4.1 Correlation

We started with checking the correlation among the features which would give us a better insight about how each character affects other character and how dependent they are. We used panda's library 'corr' to find the correlation, in which we passed 'Effective Length', 'Length', 'NumReads', 'TPM', 'count', 'deviation', 'mean' and 'standard deviation' as our feature set. We then used seaborn to print the heatmap which is more human readable. From the heatmap we could infer high correlation between 'Effective Length' and 'Length' and, 'NumReads', 'TPM', 'count' and 'mean' had high correlation. (Here 'count' is my truth-value and 'mean' and 'standard deviation' are the values calculated from bootstrap file).

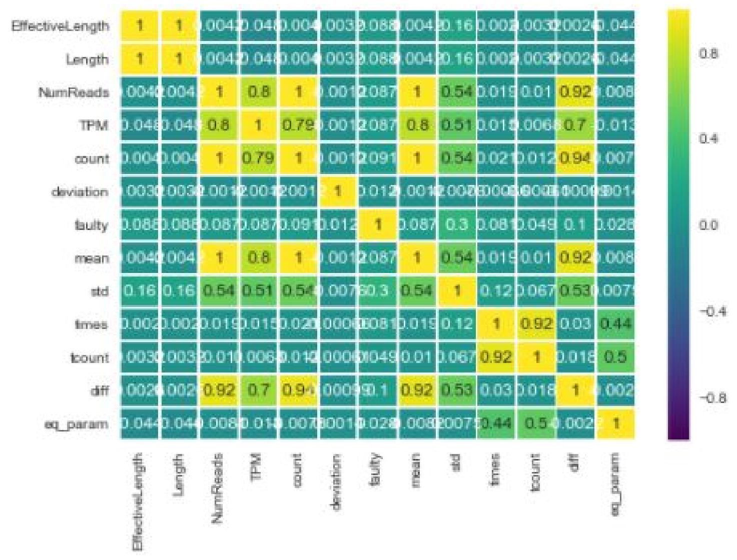


Fig1: Correlation heat map for poly$_{m}o$

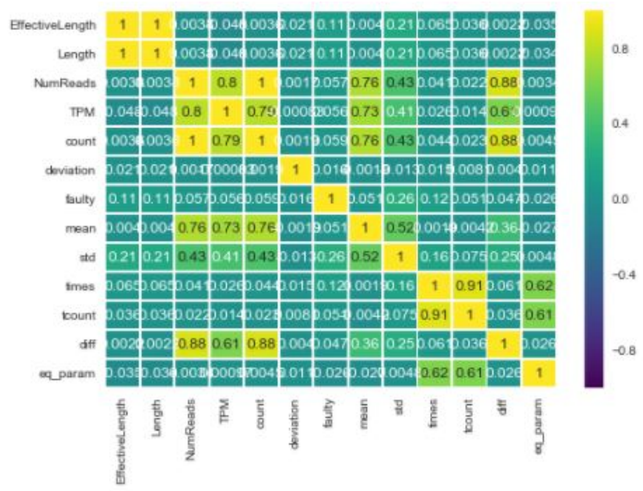


Fig1: Correlation heat map for poly$_ro$

4.2 Classification - Logistic Regression

We used logistics regression to classify the input into good and faulty transcripts. For training the model, the target variable is set to 1 for faulty transcript and 0 for good transcript. Other parameter fed into the regression are 'Effective Length', 'TPM', 'mean', 'standard deviation'. We were able to correctly predict 70% of the faulty transcript as faulty.

For poly_mo

Table 1. Statistics

Model	Accuracy	Mean Square Error
Logistic Regression	0.76	0.23

For poly_ro we observered even better accuracy:

Table 2. Statistics

Model	Accuracy	Mean Square Error
Logistic Regression	0.786	0.213

4.3 Analysis of features

Once the classification is done, the features of failing transcripts are analyzed. This analysis gave us an intuition as to what features are leading to the failure of a transcript. This was also the midway results that were achieved. While analysis the following observations were made:

For Poly_mo

1. TPM
   - 13084 out of 14176 (92.3%) good transcripts had a TPM below 10.
   - 6421 out of 12713 (50%) faulty transcripts had a TPM above 10.
   - Thus we can conclude that for TPM > 10, 85.46% transcripts are faulty
2. Effective Length
   - We saw 13121 out of 14176 (92.55%) good transcripts had EffectiveLength less than 5000.
   - And, 11307 out of 12713 (88.9%) faulty transcripts had EffectiveLength less than 5000
   - Hence, there was no clear range where either of them had a majority

Fig. 1: Scatter plot between good transcript and TPM

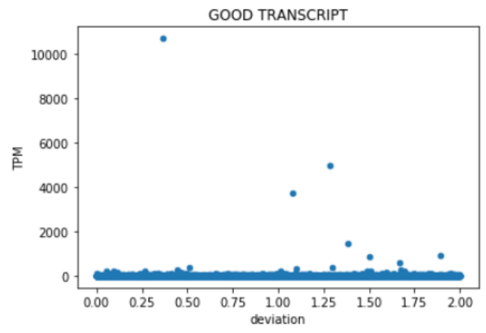


Fig. 2: Scatter plot between good transcript and Effective Length

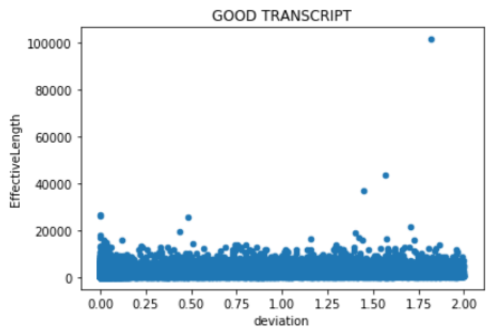


Fig. 3: Scatter plot between faulty transcript and TPM

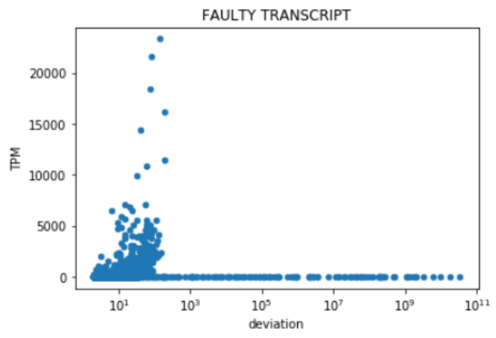
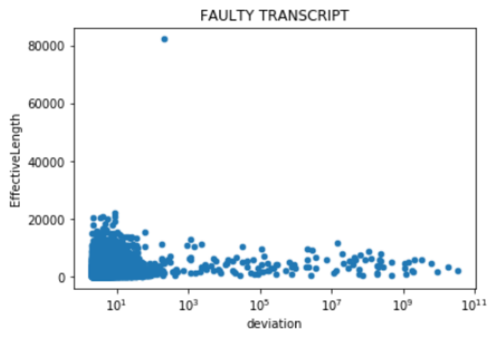


Fig. 4: Scatter plot between faulty transcript and Effective Length



For Poly_ro

1. TPM
   - 7917 out of 9097 (87%) good transcripts had a TPM below 10.
   - And, 11140 out of 17792 (62.61%) faulty transcripts had a TPM below 10.
   - Thus we can conclude that for TPM > 10, 85.01% transcripts are faulty.
2. Effective Length
   - 1. We saw 8540 out of 9097 (93.8%) good transcripts had EffectiveLength below 5000.
   - 1. And, 15986 out of 17792 (89%) faulty transcripts had EffectiveLength less than 5000.
   - 1. Hence, we can conclude that for EffectiveLength > 5000, 76.4% are faulty.

4.4 Cross-Validation

Now, the second phase of the project begins. The data is divided into two sets using cross-validation: training data and cross-validation data in the ratio 7:3. The training data is used to train the model whereas the evaluation data is used to test our prediction model.
Cross-validation data is fetched using scikit-learn library. Ideally, cross-validation data is used to tune the parameters but in our project this data is used to test our model.

4.5 Regression

In order to train the model against these feature set, we analyzed three regression models: Linear regression, Ridge Regression and Lasso Regression to check which gives us a good mean-square error and choose that regression model. We are training the model to predict the difference between the true count and mean of the transcript feeding the model a feature set of Effective Length, TPM, truth count, mean, standard deviation and a feature extracted from equivalence classes (which represents the average number of transcripts a transcript is multimapped with).

Following are the mean squared error value we got on running cross-validation on all the regressions models 5 times and on taking the average of them. The values were very close so we decided to go with linear regression.

Table 3. Statistics- Poly_mo

Model	Mean Squared Error
Linear Regression	1943949.109
Ridge Regression	1943949.128
Lasso Regression	1943957.532

Table 4. Statistics- Poly_ro

Model	Mean Squared Error
Linear Regression	1288892.49
Ridge Regression	1288892.49
Lasso Regression	1288902.73

4.6 Updating Counts

The predicted values obtained form above step are subtracted from bootstrap counts of each transcript so that the mean of the counts are shifted. These new updated values are now written in another file corresponding to each transcript.

5 Results

1. For poly_mo we saw, 12713 faulty transcripts out of which we were able to correct 3295 (25.91% of the faulty) transcripts. In total out of 26889 transcript, of which 47.27% were faulty, we could reduce that to 35.1%.

2. For poly_ro we saw, 17774 faulty transcripts out of which we were able to correct 5613 (31.5% of the faulty) transcripts. In total out of 26889 transcript, of which 66.16% were faulty, we could reduce that to 54.7%.

6 Conclusion

After shifting the mean of bootstrap values, the key observation is that most of the corrected transcripts are shifted ahead, i.e. bootstrap output was underestimating the counts. Thus, shifting their means ahead will bring them within the likelihood of true counts.

7 How to run the code

1. Github Link: https://github.com/agarwal1510/Uncertainity-Principle
2. Files above 100MB are not pushed in the repository (because of size restrictions, so please copy/move quant_bootstrap.tsv into the running directory).
3. In the poly_mo folder, first run the uncertainity_principle_poly_mo.ipynb file because python notebook creates 3 csvs namely good_prob.csv, faulty_prob.csv and combined_result.csv. (This is our first phase of our approach)
4. Next run final_nb_polymo.ipynb file, which contains all the regression models and does all the computation and finally creates a updated quant_boot.csv (which contains corrected bootstrap values)
5. For poly_ro just run uncertainity_principle_poly_ro.ipynb which outputs the corrected bootstrap values.
6. eq_class.py is used to parse eq_classes.txt to fetch eq_param feature for our model

References

[1]Patro, R., Duggal, G., Love, M., Irizarry, R. and Kingsford, C. (2017). Salmon provides fast and bias-aware quantification of transcript expression. Nature Methods, 14(4), pp.417-419.
[2]https://www.analyticsvidhya.com/blog/2017/06/a-comprehensive-guide-for-linear-ridge-and-lasso-regression/
[3]http://scikit-learn.org/stable/modules/cross_validation.html
[4]http://scikit-learn.org/stable/modules/generated/sklearn.metrics.matthews_corrcoef.html
[5]https://en.wikipedia.org/wiki/Gene_expression
[6]http://learn.genetics.utah.edu/content/science/expression/
[7]http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html