Proj. 3

Uncertainty Quantification

Team members:

Zhengwei Wei, Chong Ou,

Yunqing Yang, Haozhi Qu

Project Description

Salmon is a tool for measuring gene expression, and gives estimates of transcript abundances. Using 'bootstrapping' technique, we can measure the confidence in these estimates. However, this approach tends to underestimate the uncertainty with more transcripts falling out of the interval than expected.

Our goal is to **filter out these failed transcripts, find common properties between them**, and come up with a quality score based on those properties which measures our confidence in the estimates.

Our Approaches

1. Parse the data files (poly_truth.tsv, quant_bootstrap.tsv, quant.sf);
2. Find range of confidence interval;
3. Filter out the failed transcripts;
4. Group data by true/failed transcripts, analyze common properties.

Implementation Details

Parse quant_bootstraps.tsv

This file gives us the bootstrap data (200 rounds of sample taking).
Note: we are using `DataFrame` from the `pandas` package to easier do data analysis.

```
quant_bootstraps = tsv.TsvReader(open(root_path +
                                      "quant_bootstraps.tsv"))
quant_boot = [line for line in quant_bootstraps]
```

```
df_quant_boot = pd.DataFrame.from_records(quant_boot[1:],
                                          columns=quant_boot[0])
```

Parse poly_truth.tsv

This file gives us the true count of each transcript.

```
# Poly_truth.tsv: true counts for each transcript
poly_truth = open(root_path + "poly_truth.tsv")
lines = poly_truth.readlines()
poly_truth.close()

poly_truth = [['transcript_id','count']]
poly_truth.extend(line[:-1].split('\t') for line in lines)
df_poly_truth = pd.DataFrame.from_records(poly_truth[1:],
                                          columns=poly_truth[0])
```

```
df_poly_truth['transcript_id']=
                df_poly_truth['transcript_id'].astype(str)
df_poly_truth['count']=df_poly_truth['count'].astype(int)
```

Parse quant.sf

This file gives us some attributes of the transcripts.

```
quant_file = open(root_path + "quant.sf")
lines = quant_file.readlines()
quant_file.close()
quant = [line[:-1].split('\t') for line in lines]
```

```
df_quant = pd.DataFrame.from_records(quant[1:],
                                     columns=quant[0])
df_quant.Name = df_quant.Name.astype(str)
df_quant.Length = df_quant.Length.astype(int)
df_quant.EffectiveLength = df_quant.EffectiveLength.astype
                                                   (float)
df_quant.TPM = df_quant.TPM.astype(float)
df_quant.NumReads = df_quant.NumReads.astype(float)
```

We find and retrieve the intersecting transcript ids of poly_truth and quant_bootstraps, and sort each id's data by ascending order. There are transcripts in quant_bootstraps that don't show up in poly_truth, we'll deal with them later.

```
set_qb_id = set(df_quant_boot.columns)
set_pt_id = set(df_poly_truth.transcript_id)
intersect_ids = set_qb_id & set_pt_id

sort_qb = []
use_id = []
for id in intersect_ids:
    listed = list(df_quant_boot[id])
    listed.sort()
    use_id.append(id)
    sort_qb.append(listed)
sort_qb = list(map(list,zip(*sort_qb)))
```

Find confidence interval

Since we have already sorted each transcript id's data, we can find an empirical confidence interval of 95% by locating the numbers at index `(total_length) * 2.5%` and `(total_length) * 97.5%`, which would be the lower and upper bound.

```
df_poly_truth = df_poly_truth.set_index(['transcript_id'])

sum = len(sort_qb)
percent2dot5 = df_qb_sorted.loc[int(sum*0.025)-1]
percent97dot5 = df_qb_sorted.loc[int(sum*0.975)-1]
```

Find the failed transcripts

Compare the counts given by poly_truth with the lower and upper bound we found earlier. If not in range we treat it as a failed transcript, else true.

```
true_id = []
false_id = []
for id in use_id:
    down = float(percent2dot5[id])
    up = float(percent97dot5[id])
    true_count = float(df_poly_truth.loc[id])
    if down < true_count < up:
        true_id.append(id)
    else:
        false_id.append(id)
```

We go back to deal with the 'diff' transcript ids we ignored earlier. The counts of these diff transcript ids are zero, and we assume that these are true transcripts.

```
true_id.extend(list(set_qb_id.difference(set_pt_id)))
all_id = true_id[:]
all_id.extend(false_id)
```

We add a label of 1 representing true transcripts and 0 representing failed transcripts for easy grouping later on.

```
label = [1 if i < len(true_id) else 0 for i in
                        range(len(true_id) + len(false_id))]
labeled_id = [all_id,label]
labeled = list(map(list,zip(*labeled_id)))
```

Common Properties of Failed Transcripts

We group the data by true and failed transcripts, and first observe the mean, std, max and min.

Observing the mean, the average TPM and NumReads of failed transcripts is a lot bigger than the true ones.

label	Length	EffectiveLength	TPM	NumReads
0	2445.007350	2245.894343	52.556885	1784.370671
1	1905.694197	1706.993431	0.841197	20.235059

label	Length	EffectiveLength	TPM	NumReads
0	2320.422083	2121.332135	41.485945	1378.079501
1	1901.681056	1703.008206	0.154861	6.681882

With the std, we find that failed transcripts tend to have a significantly larger variance of TPM and NumReads.

label	Length	EffectiveLength	TPM	NumReads
0	2380.728560	2380.669848	448.368497	15740.352178
1	2055.905694	2055.526264	45.873037	235.234059

label	Length	EffectiveLength	TPM	NumReads
0	2302.710811	2302.636007	396.200319	13629.239858
1	2059.808430	2059.407366	9.489371	152.179457

Min values don't seem to have much to offer other than maybe a slightly bigger length.

label	Name	Length	EffectiveLength	TPM	NumReads
0	ENST00000000233	158	10.987	0.0	0.0
1	ENST00000000412	21	9.784	0.0	0.0

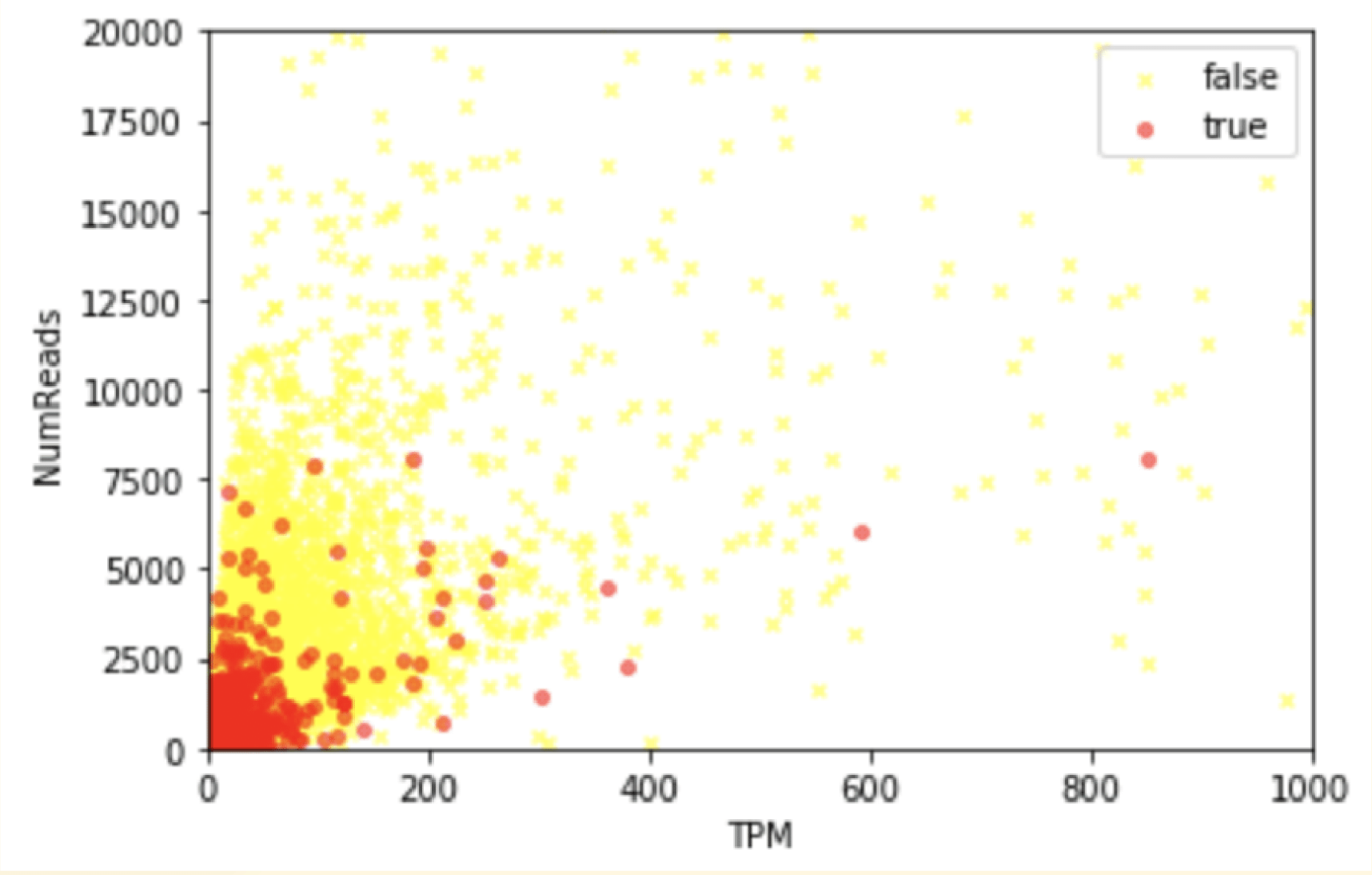
label	Name	Length	EffectiveLength	TPM	NumReads
0	ENST00000000233	82	10.861	0.0	0.0
1	ENST00000000412	21	9.784	0.0	0.0

With max values, the TPM and NumReads attributes seem interesting, which given the info from std, tells us that failed transcripts have a significantly wider range and variance of TPM and NumReads.

label	Name	Length	EffectiveLength	TPM	NumReads
0	ENST00000610278	101518	101318.991	23356.420222	1.109005e+06
1	ENST00000610279	109224	109024.991	10710.459004	3.769085e+04

label	Name	Length	EffectiveLength	TPM	NumReads
0	ENST00000610279	101518	101318.991	23356.420222	1.109005e+06
1	ENST00000610276	109224	109024.991	2435.717783	3.110509e+04

This looks more clear when we plot the distribution:



Classification Models

We used `sklearn` package for this.

First we gave **linear regression** a shot, and we ended up with:

mse= 0.24674520923824544
accuracy= 0.5450268817204301

The results were not ideal. Looks like it doesn't seem to be linear, so we tried another classification model.

With **support vector regression**, we achieved:

mse= 0.0616298873525875 8
accuracy= 0.889258861439312 5

88.9% was a pretty high accuracy level, but we also managed to reach:

mse= 0.0616298873525875 8
accuracy= 0.966733870967741 9

96.7% accuracy after we dropped the '0' count value transcripts from the 'true' group.