1 Introduction

1.1 The data set

The google data set is our main focus. In fact our experiment is based on a small subset that contains usage information of 100,000 tasks.

While the size of this subset seems adequate for our experimenting purpose, however, this data set is highly imbalanced — only 232(0.2%) tasks failed eventually. This will usually cause negative impacts on learning algorithms. A sampling technique called undersampling was used to overcome this problem. Undersampling is the process of removing samples of the majority, non-failed tasks in this case, from the training set to force a reasonable data ratio. For example, if we have a training set containing 200 failed tasks and a data ratio of 2:1(non-failed tasks : failed tasks) is required, we use undersampling to reduce the amount of non-failed tasks to 400.

1.2 Prediction technique

The prediction technique we used is a combination of Sliding Window and CART(Classification and Regression Tree). This could simply be broken down into two steps:

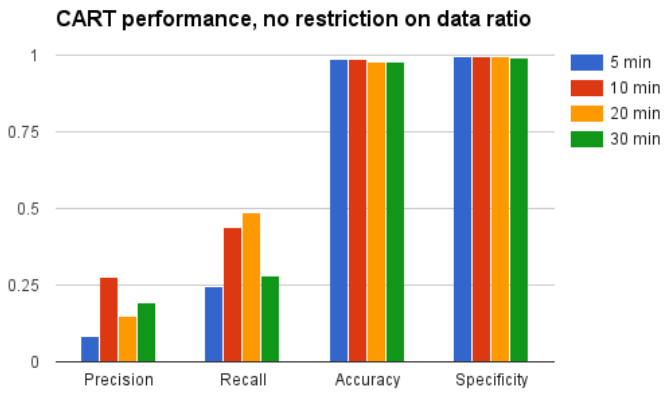
Step 1: Use Sliding Window to convert our complicated learning problem into a general supervised learning problem, which could be later input into any kind of supervised learning algorithms.

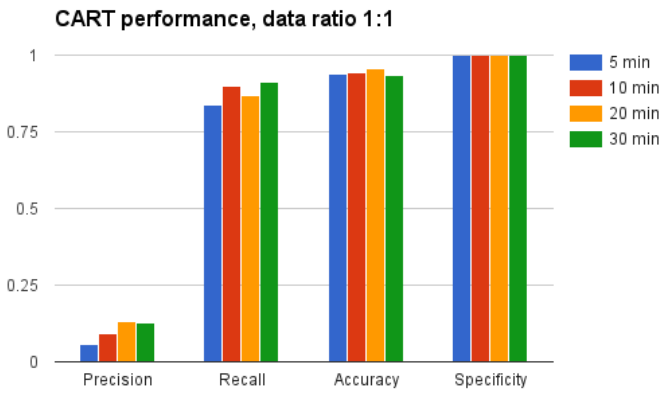
In our settings, each window contains usage information of a single task in a period of its life time. Windows may or may not overlap(overlapping of time intervals) depending on their sizes. The size of a window is just the length of time it covers. Since 5 minutes is the smallest unit we can get from this data set and most tasks are short in time, we decided to construct windows in four different sizes: 5min, 10min, 20min and 30min. 5min windows does not overlap each other, two consecutive 10min windows overlaps by length of 5min, similarly two consecutive 20min windows overlaps by 15min. The label of each window is 0 or 1 where 1 means the task failed eventually and 0 otherwise.

Step 2: Feed the window-label pairs into the CART algorithm and wait for results.

2 Bar charts

Balancing data ratio improves recalls significantly.

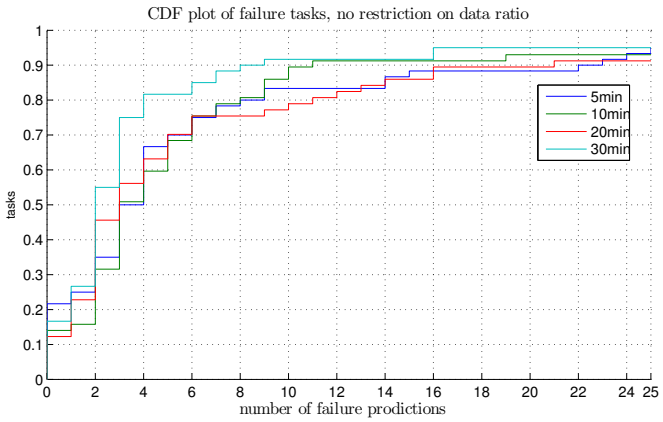


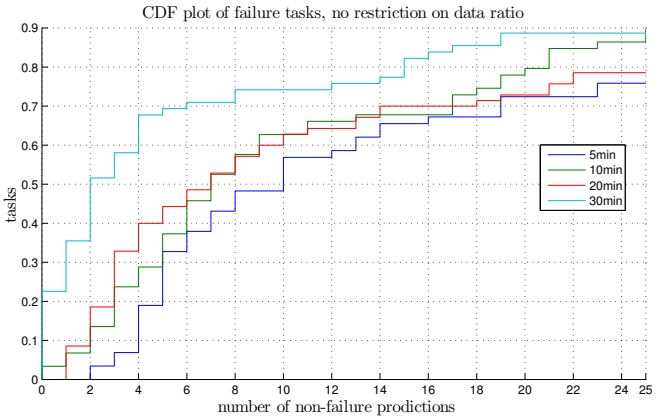


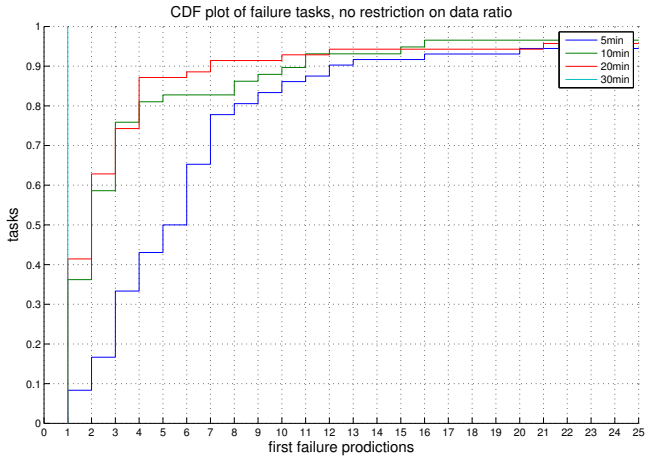
3 Failed tasks

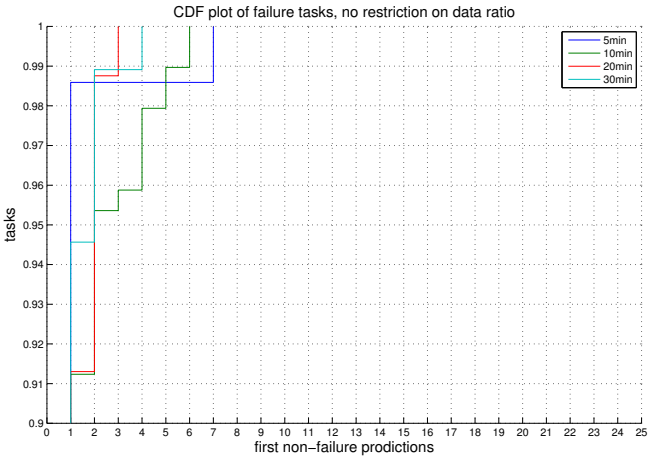
Predictions on failed tasks.

3.1 CDF's for all failed tasks, lines broken down by window size

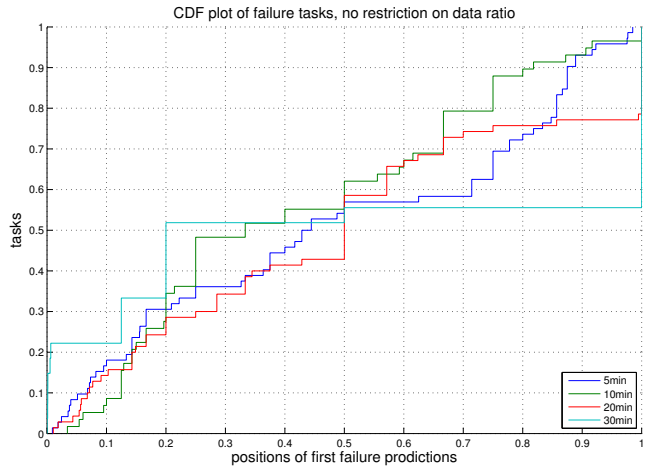


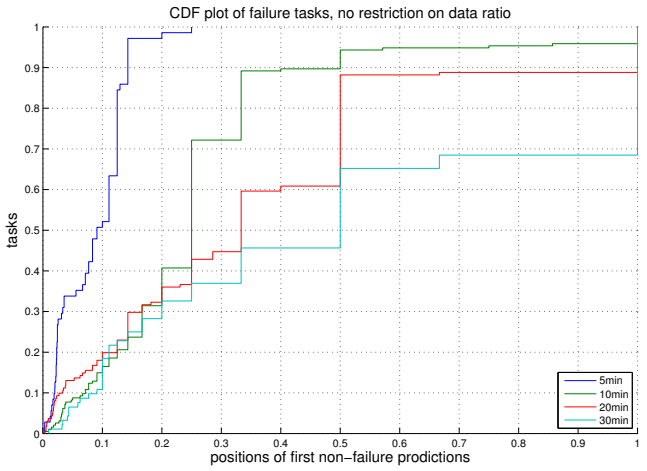




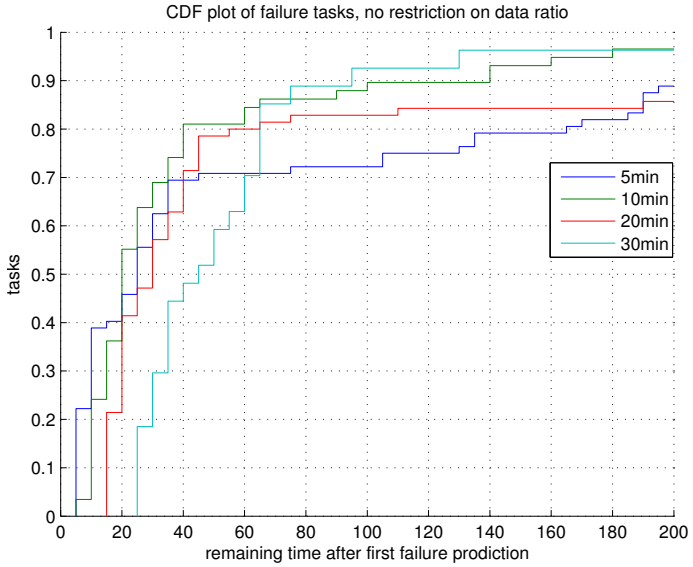


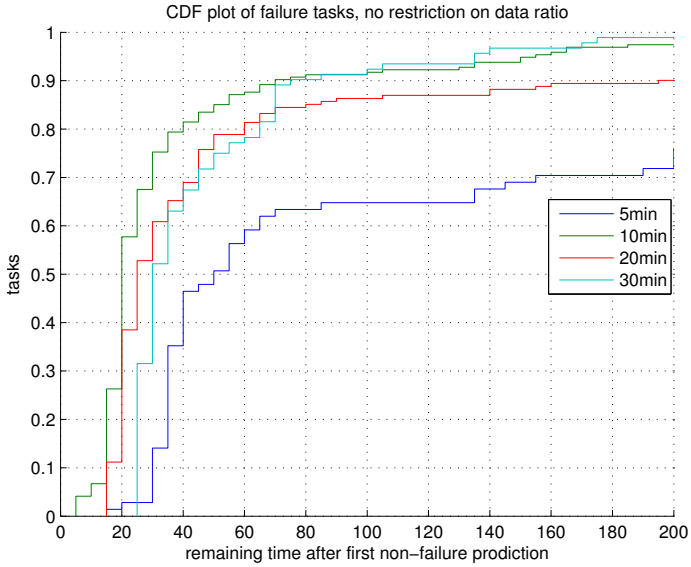
⇒More non-failure predictions than failure predictions, but no much difference.





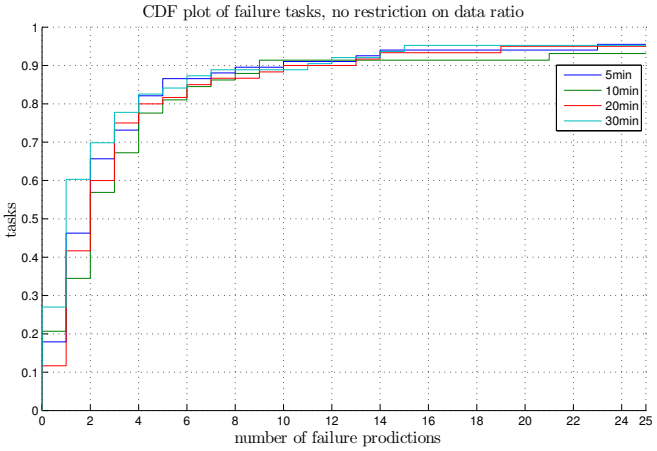
⇒First non-failure predictions happens earlier than first failure predictions. Position = first/total

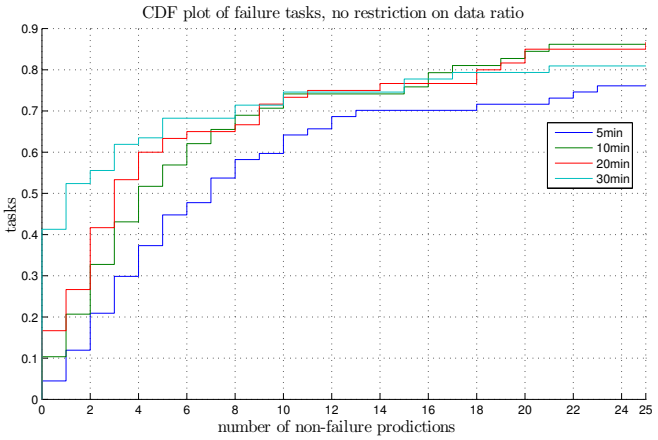




3.2 [2-consequtive]CDF's for all failed tasks, lines broken down by window size

Same plots but only 2 consecutive failure predictions are recognized as one failure prediction.



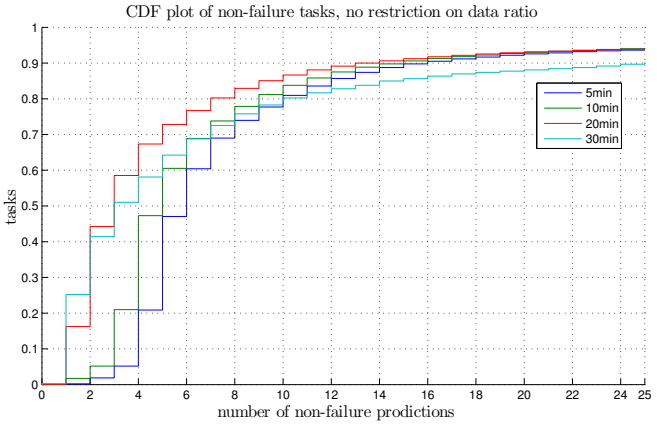


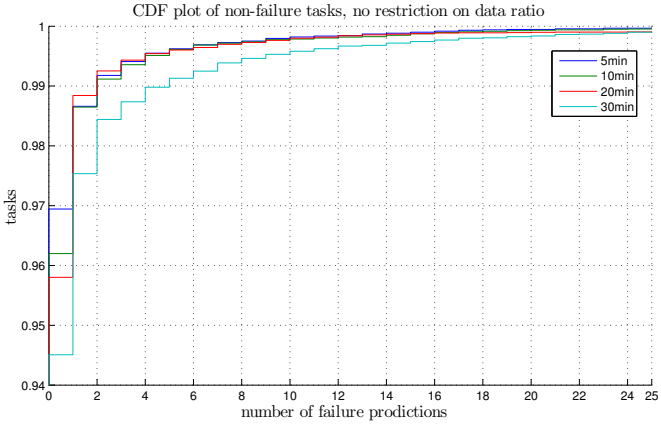
⇒ **Pretty much identical to previous plots**

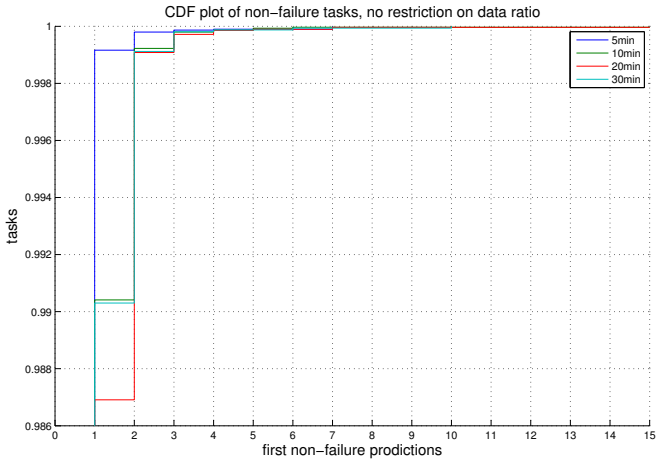
4 Non-failed tasks

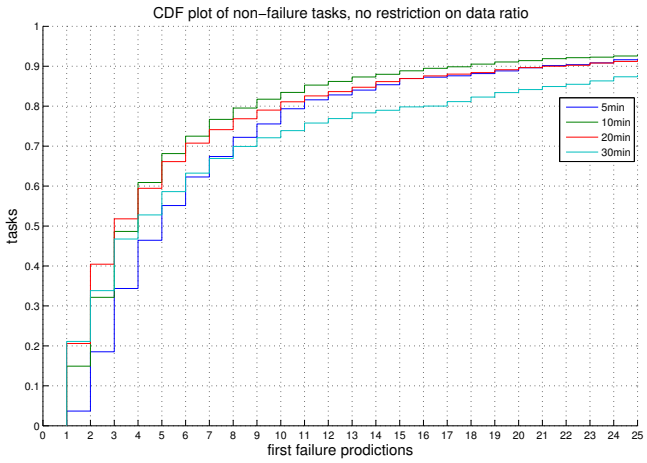
Predictions on non-failed tasks.

4.1 CDF's for all non-failed tasks, lines broken down by window size

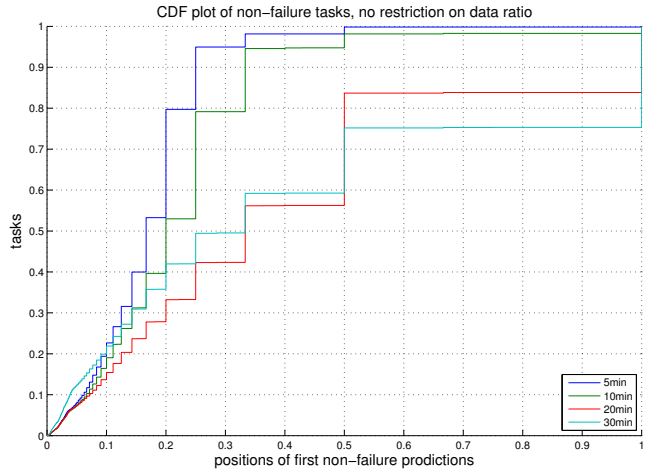


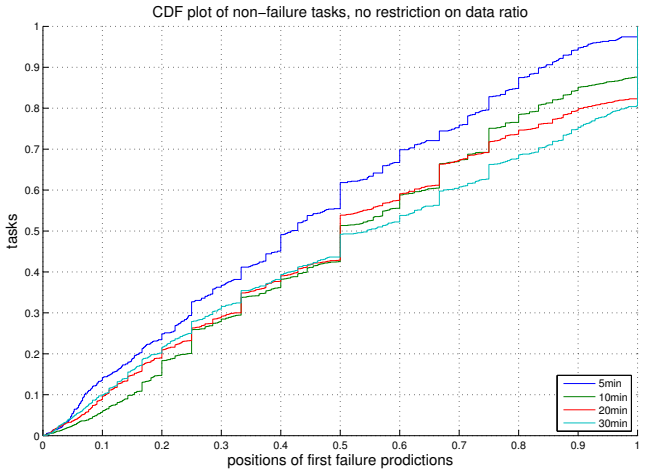




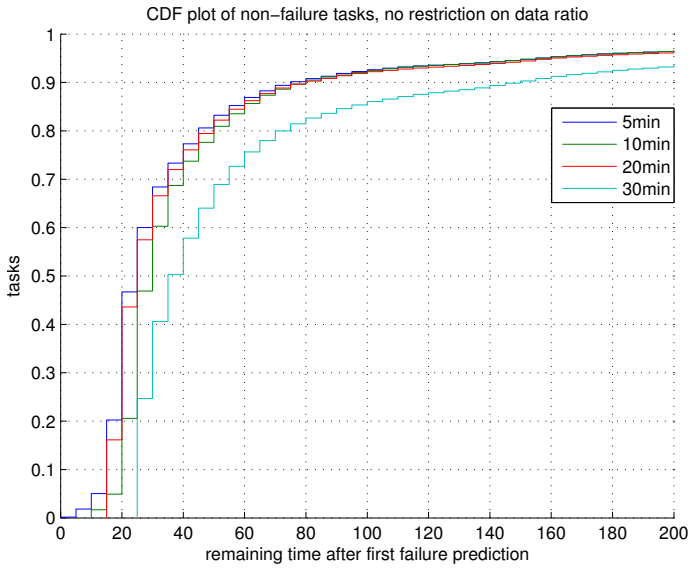


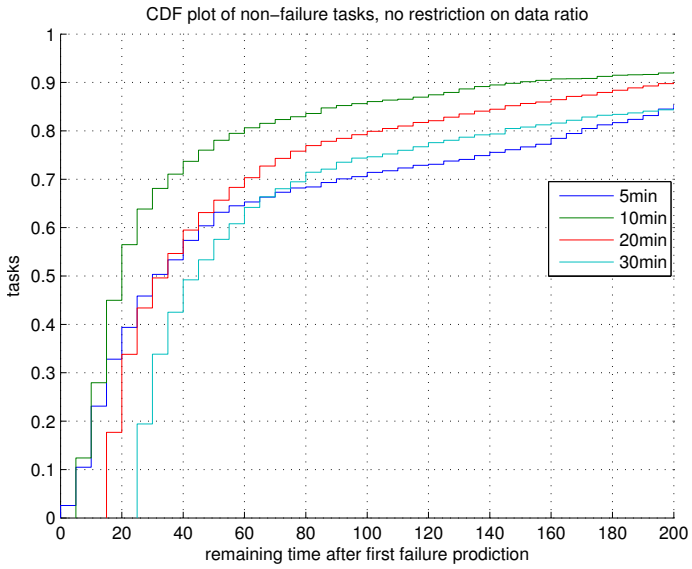
⇒ **Non-failure predictions dominates.**





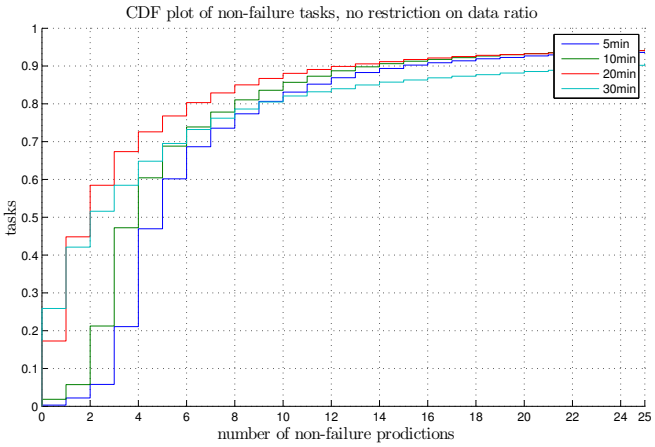
⇒ **First non-failure happens earlier than first failure predictions.**

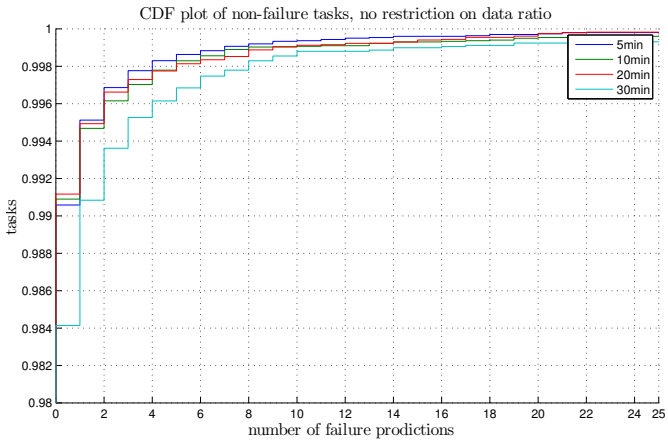




4.2 [2-consequtive]CDF's for all non-failed tasks, lines broken down by window size

Same plots but only 2 consecutive failure predictions are recognized as one failure prediction.





⇒ **Pretty much identical to previous plots**