Universidade Federal da Paraíba (UFPB)
Curso: Sistemas de Informação
Disciplina: Avaliação de Desempenho de Sistemas
Professor: Marcus Carvalho

Prática 3 – Intervalo de confiança

Nesta atividade de laboratório é apresentado um problema de intervalo de confiança, envolvendo a análise de desempenho de três algoritmos de ordenação de arrays de números inteiros. Você deve executar o programa fornecido e avaliar o desempenho do mesmo, a partir das métricas gravadas pelo programa. Para responder às questões, você deve escrever um relatório com as suas análises.

Deve ser enviado pelo Google Classroom o relatório, de preferência em um arquivo do Google Docs contendo as respostas das questões com as análises, além de um arquivo *ZIP* com os dados coletados na medição dos programas (arquivos *.txt* gerados). É **fortemente recomendado** o uso de gráficos nos relatórios, para a exibição dos dados coletados e para ajudar na sua análise. Tabelas também devem ser usadas para exibir dados quando necessário.

No início do relatório, descreva a configuração da máquina na qual você está executando cada análise de desempenho, com informações sobre a CPU, Memória, Cache, Disco, etc.

Problema

O programa consiste na implementação de três algoritmos de ordenação para arrays de números inteiros: *quick sort*, *merge sort* e *counting sort*. A idéia desta atividade é comparar o desempenho dos três algoritmos para diferentes situações de valores de entrada para ordenação. Para executar o programa, deve-se passar três parâmetros de entrada. A linha de execução esperada é:

```
java –cp bin MedidorDeOrdenacao <ordenador(QUICK, MERGE, COUNTING)> <tamanho-da-entrada> <valor-maximo>
```

Os parâmetros de entrada esperados são os seguintes:

- **Ordenador**: um dos três valores (QUICK, MERGE, ou COUNTING) que indica qual o algoritmo de ordenação será usado na medição.
- **Tamanho da entrada (T)**: tamanho do array de inteiros a ser ordenado.
- **Valor máximo (MAX)**: o valor máximo possível dos números inteiros que irão compor o array de entrada. Os valores do array serão gerados aleatoriamente, sendo gerados uma quantidade **T** de valores inteiros no intervalo [0, MAX].

Especificações:

Processador: Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz 2.30 GHz
RAM: 12GB
Disco: 1T

Questões

1. Queremos comparar o desempenho de cada algoritmo de ordenação, medindo o tempo para ordenar arrays de inteiros de certos tamanhos e com diferentes faixas de possíveis valores inteiros.
    a. Realize 30 medições para cada algoritmo (quick, merge, counting) e com base nos dados obtidos calcule, para cada um, quantas vezes cada um deve ser executado (tamanho da amostra) para que se tenha um intervalo de confiança com uma margem de erro menor ou igual a 2%, para um nível de confiança de 95%. Use como parâmetros de entrada um tamanho de entrada de 8000000 (8 milhões) e o valor máximo da entrada 800000 (800 mil). Exemplo:

    java -cp bin MedidorDeOrdenacao quick 8000000 800000

    java -cp bin MedidorDeOrdenacao merge 8000000 800000

    java -cp bin MedidorDeOrdenacao counting 8000000 800000

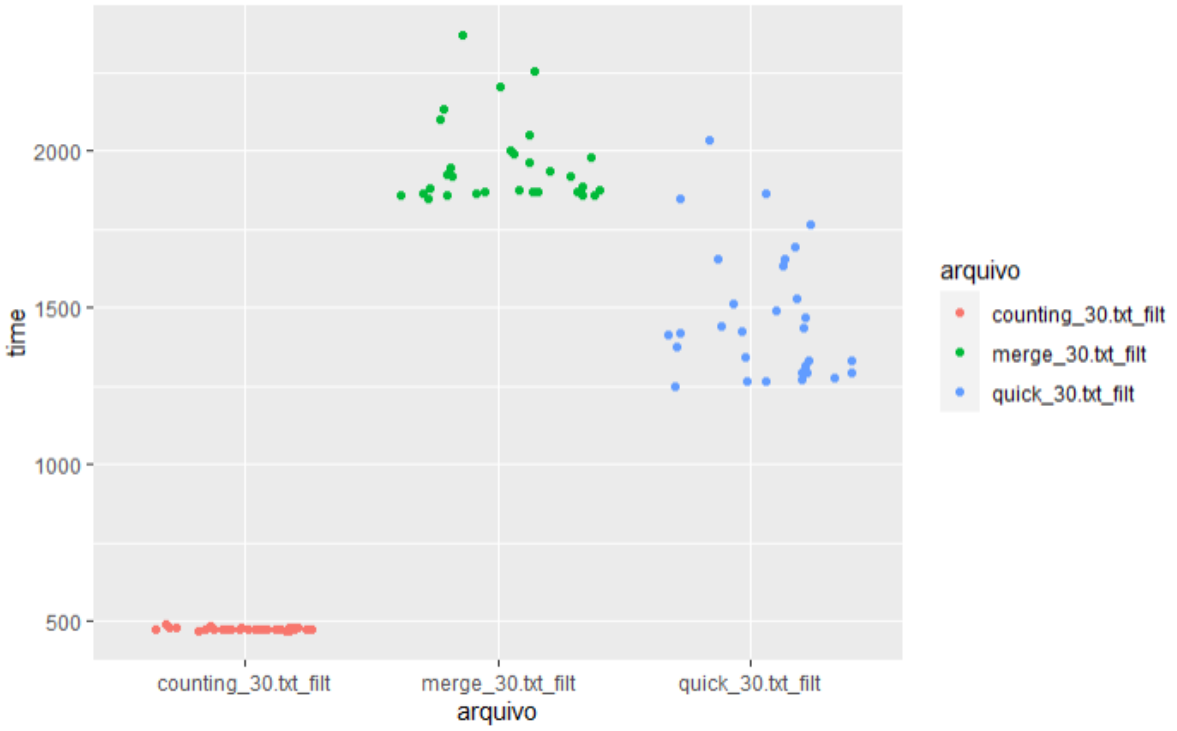
    Média:

    Quick: 1472,833

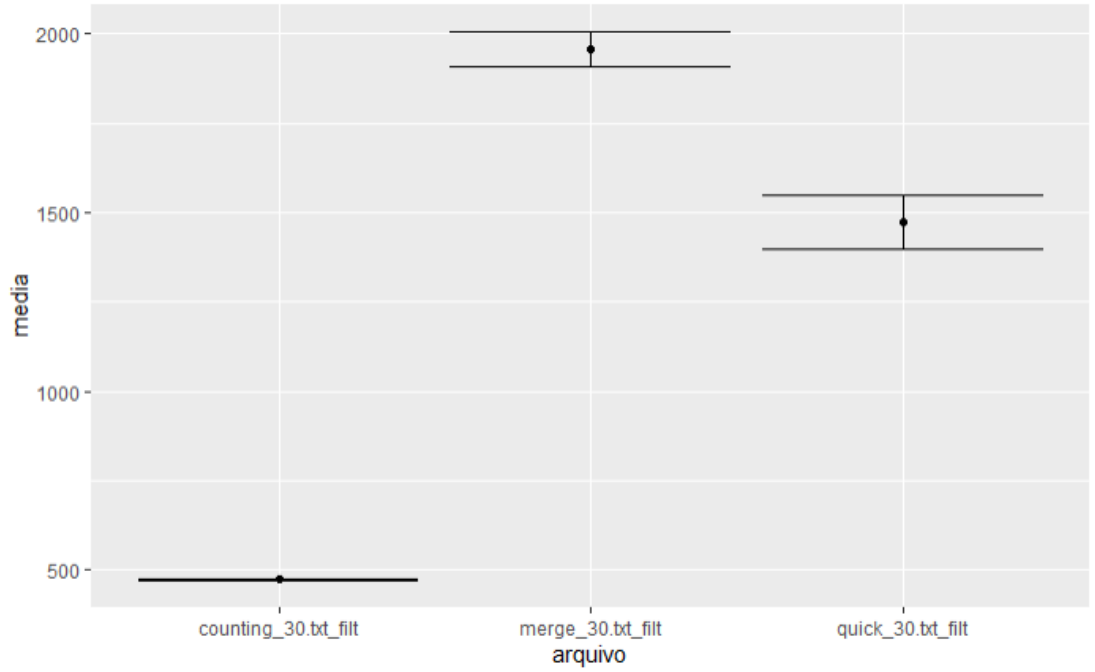
    Merge: 1957,800

    Counting: 474,200

Sistema	Média	Desvio padrão	Tamanho da amostra	Intervalo - erro	Intervalo - limite inferior	Intervalo - limite superior	Margem de erro	Tamanho da amostra para margem 2%
Quick	1,472.833	208.758	30.000	74.702	1,398.132	1,547.535	5.07%	192.9438103
Merge	1,957.800	133.025	30.000	47.601	1,910.199	2,005.401	2.43%	44.33851406
Counting	474.200	4.902	30.000	1.754	472.446	475.954	0.37%	1.02621698



Intervalo de confiança:



Com o número de amostra igual a 30 não estávamos atingindo uma margem de erro menor que 2%, o Quick está com uma margem de erro de 5,07% e o Merge de 2,43%. Para alcançarmos um margem de erro menor ou igual a 2% vamos precisar de uma amostra igual a:

Quick: 193

Merge: 45

Counting: 30

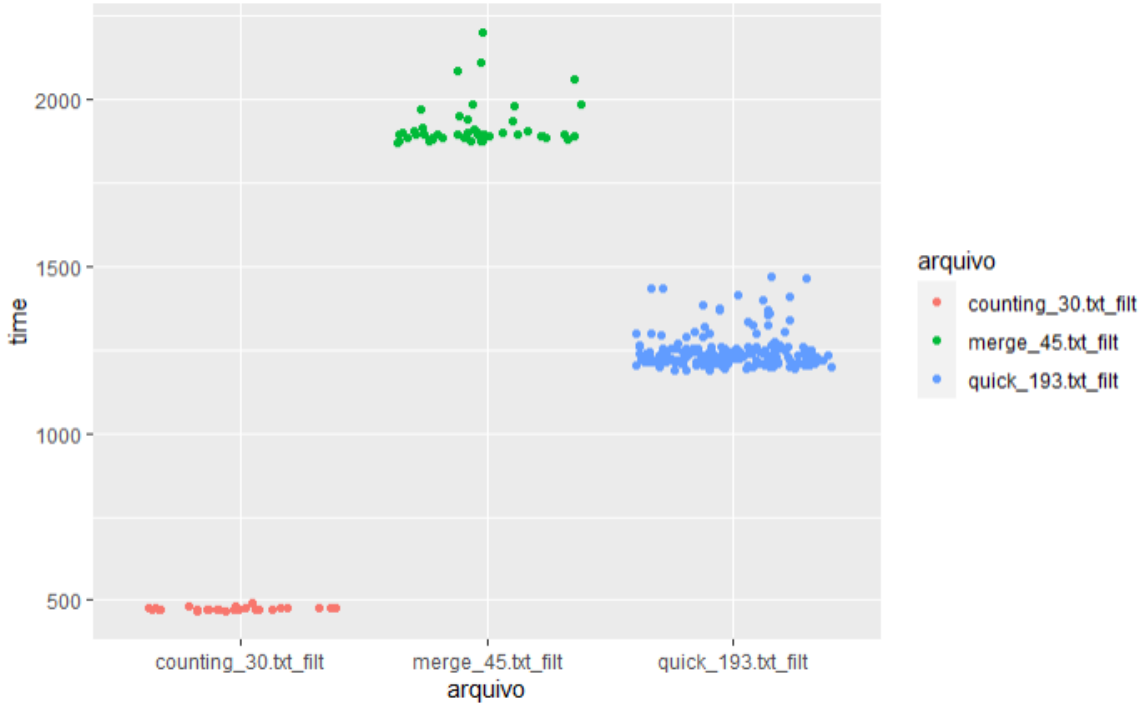
Com esses valores o quick alcança uma margem de erro de 0.59%, o merge 1.05 e o counting 0.37%.

b. Use o maior valor dos três tamanhos de amostra (n) obtidos na questão anterior, execute as ordenações n vezes para cada algoritmo, com a mesma entrada anterior. Qual a média e o intervalo de confiança, com nível de confiança de 95%, do tempo de ordenação obtidos para cada algoritmo?

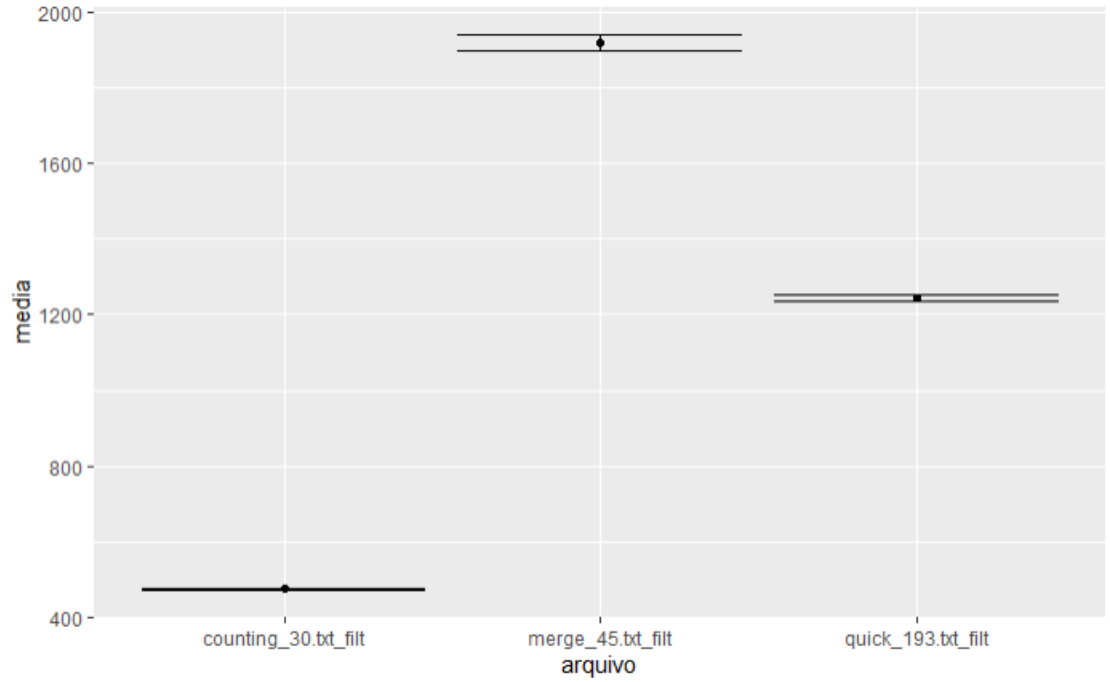
Almentando o número da amostra para obter uma margem de erro igual ou menor a 2% tivemos algumas mudanças nos valores, veremos a seguir:

Média:
Quick: 1244,301
Merge: 1920,933
Counting: 474,200

Sistema	Média	Desvio padrão	Tamanho da amostra	Intervalo - erro	Intervalo - limite inferior	Intervalo - limite superior	Margem de erro	Tamanho da amostra para margem 2%
Quick	1,244.301	51.761	193.000	7.303	1,236.998	1,251.603	0.59%	16.61907568
Merge	1,920.933	69.305	45.000	20.249	1,900.684	1,941.183	1.05%	12.50138806
Counting	474.200	4.902	30.000	1.754	472.446	475.954	0.37%	1.02621698



E alcançamos uma margem de erro menor que 2%:



No algoritmo Quick precisamos rodar 193 vezes para alcançarmos uma margem de erro menor que 2%, já no merge, precisamos executá-lo 45 vezes, e o counting já se encontrava com uma margem de erro menor que 2% com a sua amostra de início, que é igual a 30.

c. Com base no intervalo de confiança da resposta anterior, podemos afirmar **estatisticamente** que desses três algoritmos existem uns que são melhores que os

outros em relação ao tempo de ordenação? Qual seria o melhor e o pior algoritmo para esta entrada? Justifique.

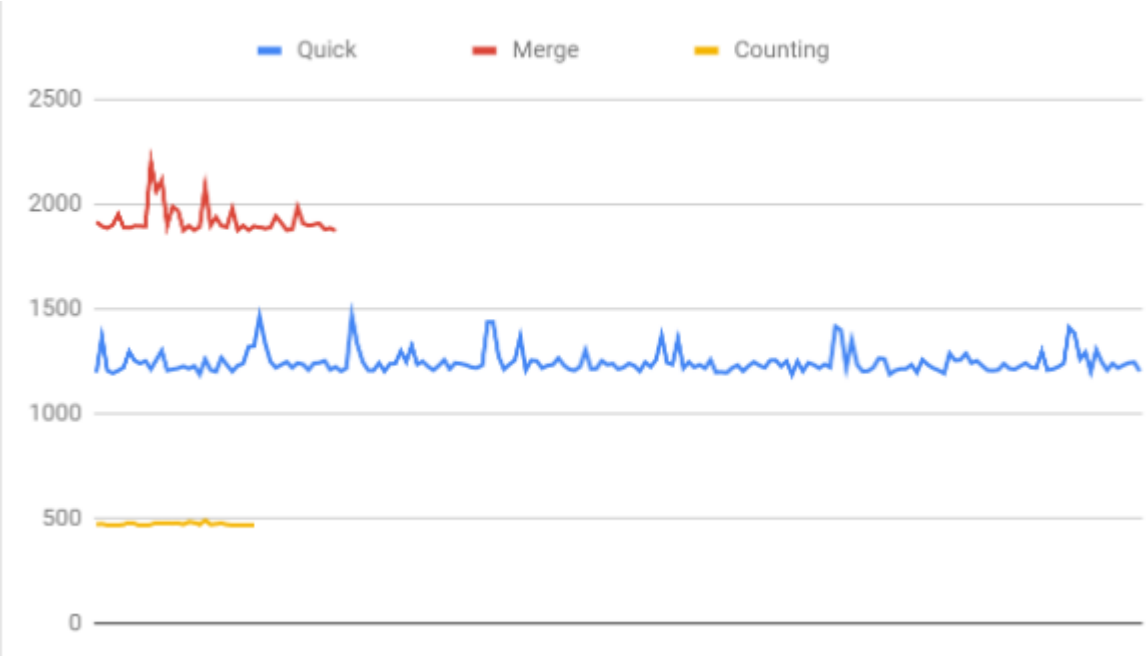
Como podemos observar no gráfico de intervalo de confiança da letra b, o algoritmo mais eficiente sem dúvidas é o counting, que teve uma média 474,200 ms para cada execução, com uma margem de erro de 0,59% podendo alcançar um limite inferior de 472,446 e um limite superior de 475,954.

Em seguida vem o algoritmo Quick, que tem uma média de 1244,301 ms para cada execução, podendo alcançar um limite superior de 1251,603 e um limite inferior de 1236,998 que é 2,62 vezes mais que o Counting.

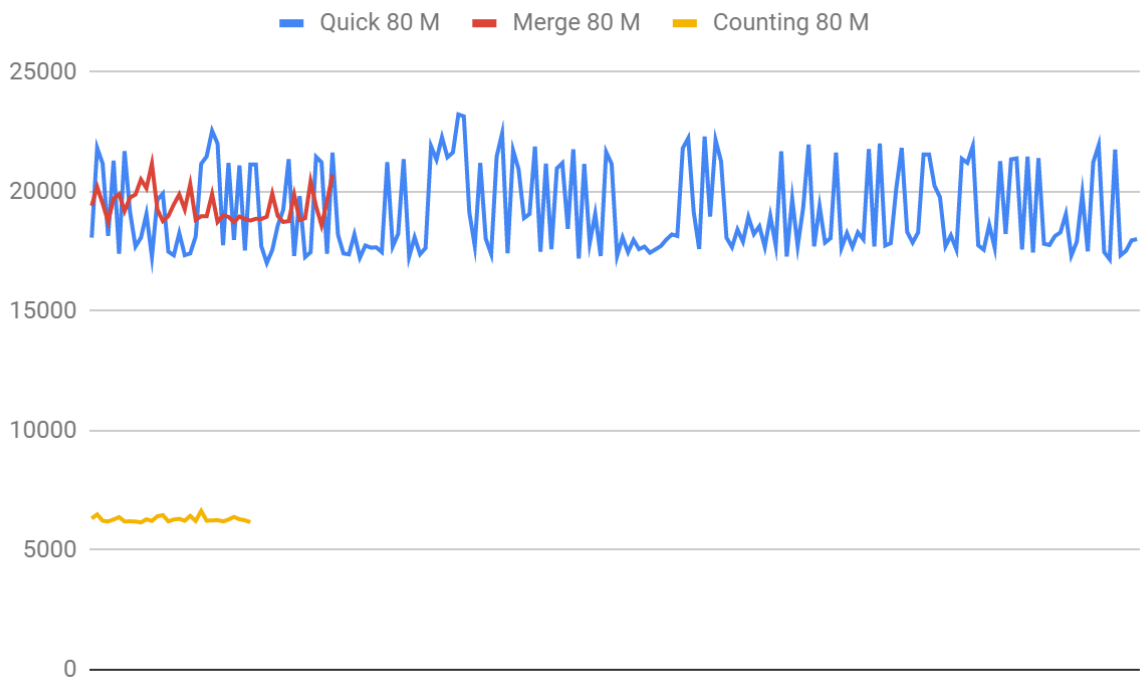
E por último vem o Merge, que tem em média 1920,933 ms para cada execução, com limite superior de 1941,183 e limite inferior de 1900,684, sendo 4 vezes mais demorada que o Counting e 1,54 vezes mais demorado que o quick.

2. Queremos analisar agora como o tempo de ordenação de cada algoritmo varia ao mudarmos a entrada.
    a. Verifique como o tempo de ordenação aumenta ao aumentarmos o tamanho da entrada, para cada algoritmo. Mantenha fixo o valor máximo em 800000 (800 mil) e varie o tamanho da entrada em dois valores: 8000000 (8 milhões) e 80000000 (80 milhões). Analise o padrão de aumento do tempo de ordenação de cada algoritmo e indique qual o melhor e o pior em cada cenário, usando a análise de intervalos de confiança. O tamanho da amostra deve ser suficiente para ter uma margem de erro menor ou igual a 2%.

Gráfico com tamanho de entrada de 8 Milhões

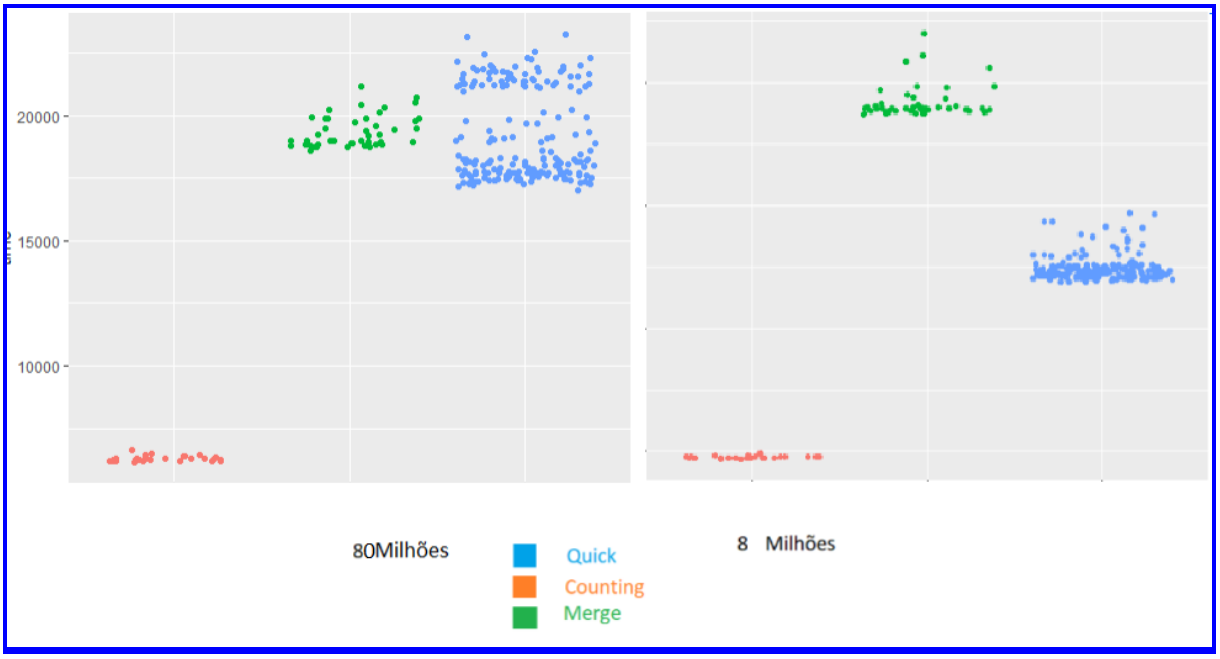


Quando rodei os algoritmos com a entrada de 8 milhões, obtive um resultado no algoritmo Quick muito fora do normal, um valor igual a 734097 ms sendo que a média é por volta de 22000 ms então apaguei ele para ter uma visualização melhor nos gráficos.

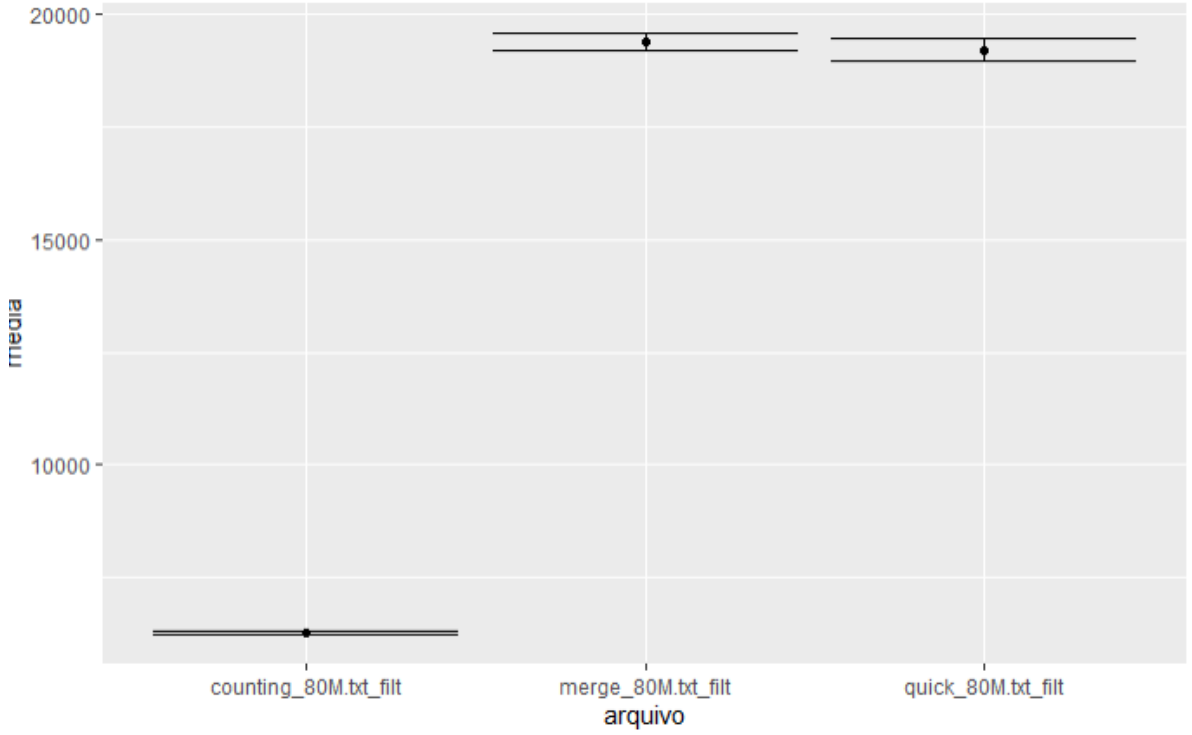


Observando os dois gráficos acima, podemos ver que, com o aumento da taxa de entrada, no primeiro gráfico com a entrada igual de 8 milhões o counting teve uma média de tempo melhor que a de todos os outros algoritmos, e o Merge a pior média de tempo. Quando observamos o gráfico 2, que tem como taxa de entrada 80 milhões, o counting continua sendo o melhor, mas é difícil visualizar quem é o que está com a média de tempo maior. Então vamos observar o intervalo de confiança.

	Taxa de entrada	8 Milhões	80 Milhões
Quick	**Média**	1244.301	19,192.693
Merge	**Média**	1920.933	19,381.978
Counting	**Média**	**474.200**	**6,293.167**

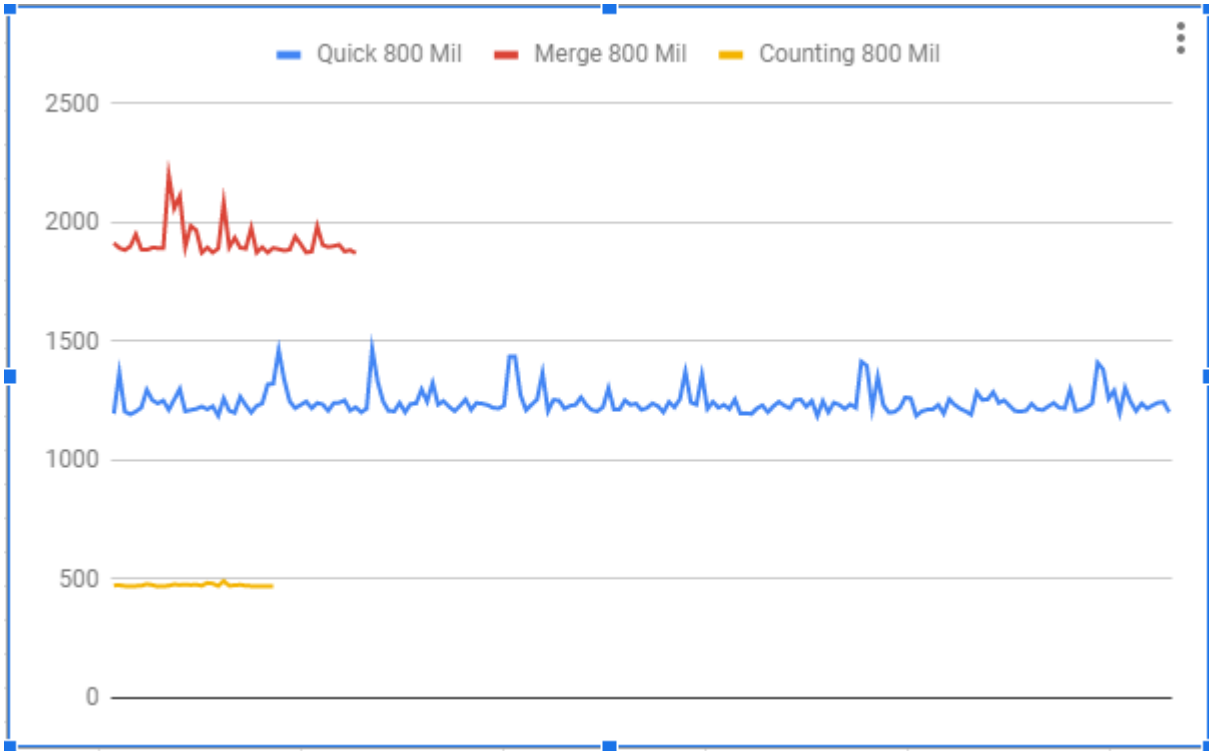


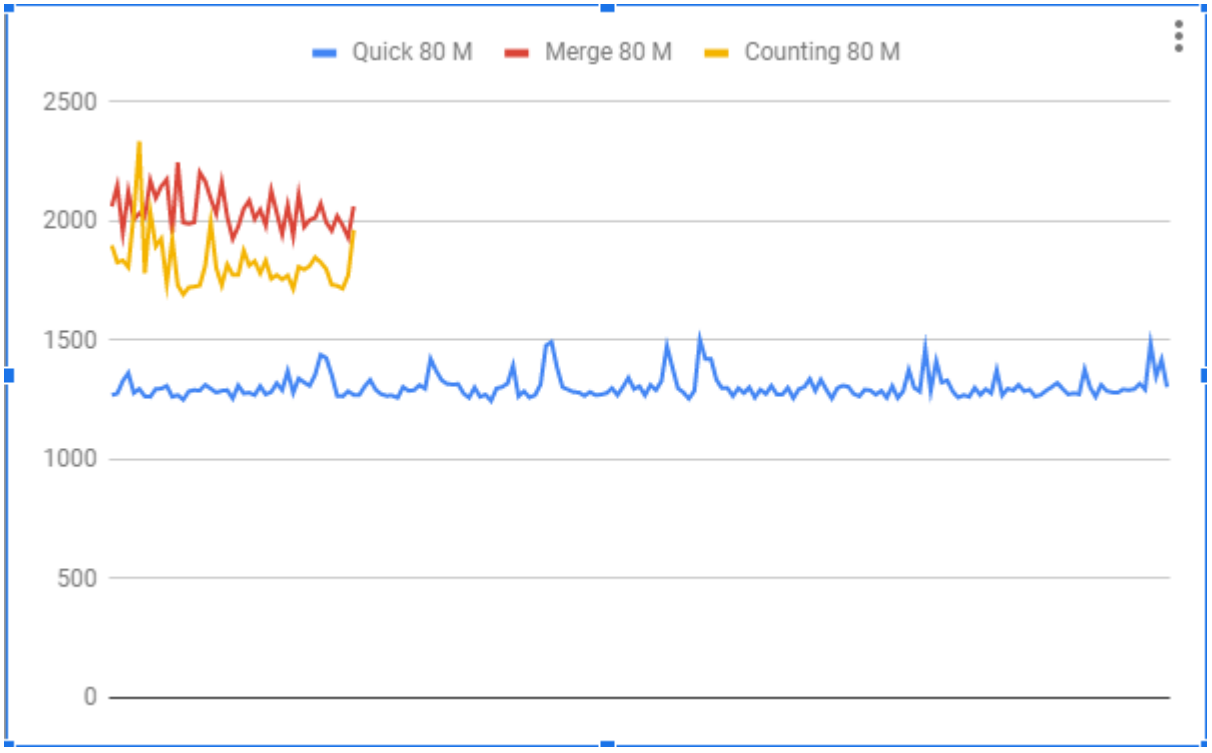
Mas só pela média não podemos concluir que com o aumento da taxa de entrada o algoritmo quick é o melhor, então vamos analisar o intervalo de confiança:



Observando o gráfico acima podemos concluir que o counting é melhor que os outros algoritmos, mas não dá pra concluir quem é melhor entre o merge e o quick pois temos um empate técnico. Nessa avaliação foi realizado o teste aumentando o tamanho da população para tentar sair do empate técnico, mas não foi possível alcançar tal resultado.

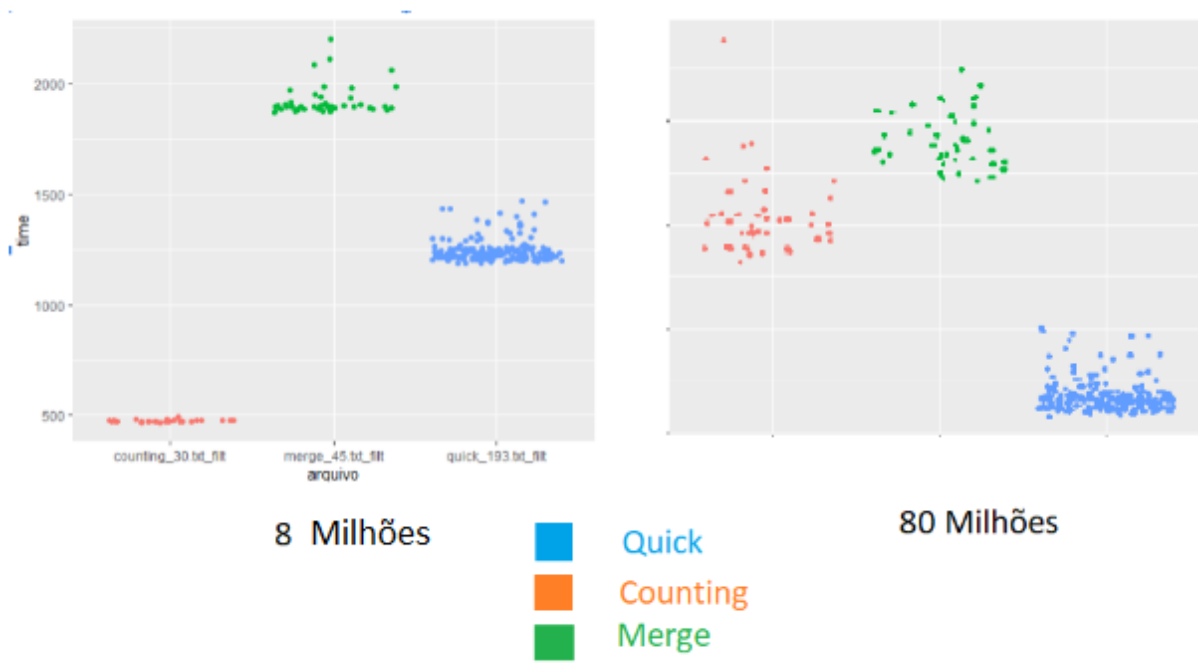
b. Verifique como o tempo de ordenação aumenta ao aumentarmos a faixa de valores de entrada possíveis (ou seja, aumentando valor máximo da entrada). Mantenha o tamanho da entrada fixo em 8000000 (8 milhões) e varie o valor máximo de entrada em dois valores: 800000 (800 mil) e 80000000 (80 milhões). Analise o padrão de aumento do tempo de ordenação de cada algoritmo e indique qual o melhor e o pior em cada cenário, usando a análise de intervalos de confiança. O tamanho da amostra deve ser suficiente para ter uma margem de erro menor ou igual a 2%.



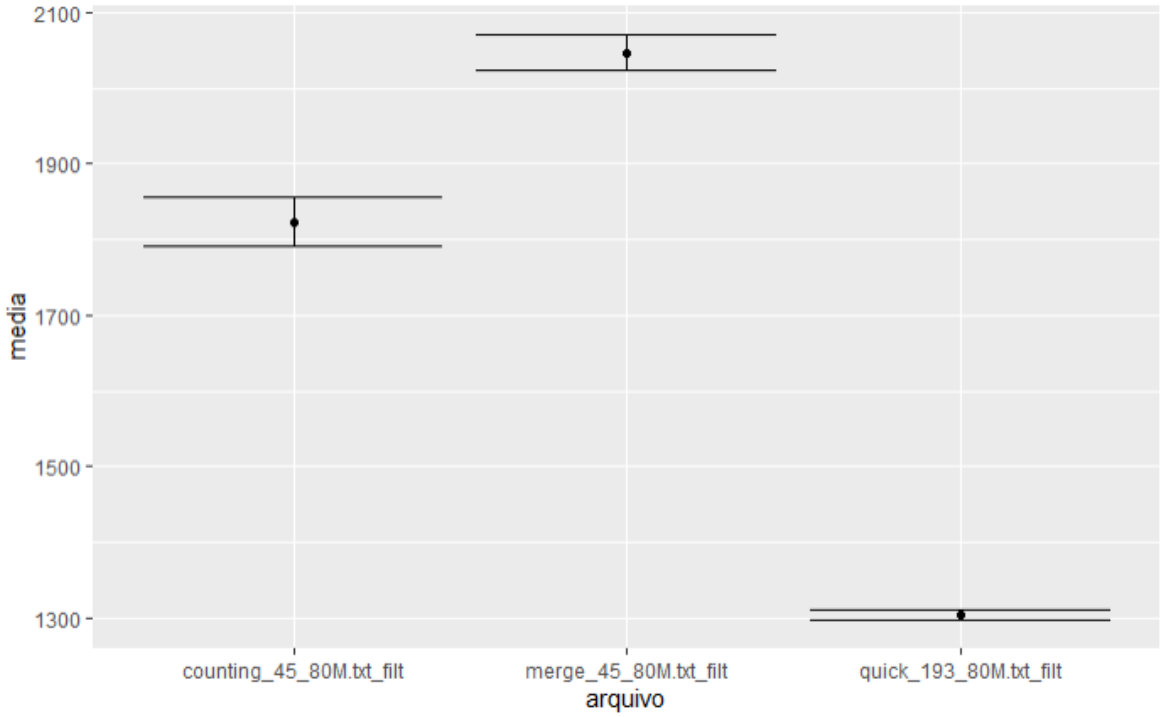


Observando os dois gráficos acima, podemos ver que, com o aumento do valor máximo de entrada, o algoritmo Quick acaba ficando mais rápido que os outros, no primeiro gráfico ele é mais lento do que o counting, mas quando se aumenta a taxa de entrada o quick leva vantagem, podemos ver isso no segundo gráfico, analise as informações:

	Taxa de entrada	8 Milhões	80 Milhões
Quick	**Média**	1244.301	**1305.301**
Merge	**Média**	1920.933	2046.778
Counting	**Média**	**474.200**	1,823.622



Mas só pela média não podemos concluir que com o aumento do valor máximo de entrada o algoritmo quick é o melhor, então vamos analisar o intervalo de confiança:



Pelo intervalo de confiança podemos ver, que o quick, quando aumenta o valor máximo de entrada passa a ser melhor que os outros.

3. Com base nas questões anteriores e em outras análises que você pode fazer, quais algoritmos são os mais adequados para diferentes situações de tamanho da entrada e do valor máximo? Justifique com base nas análises de intervalos de confiança.

Com base nos dados, e nos gráficos, podemos afirmar que o algoritmo counting é melhor do que os outros algoritmos na maioria dos casos, menos quando se tem um valor fixo de entrada muito alto, nesse caso o quick é superior. Analisando os outros casos podemos ver que o algoritmo quick é melhor que o merge em quase todos os casos, menos quando se tem uma taxa de entrada muito alta, que é quando se tem um empate técnico entre os dois.

Enfim, podemos concluir que para os casos com um valor fixo de entrada muito alto é melhor utilizarmos o quick, já nos outros casos é melhor utilizar o counting.