SEA 2018 report

Contents

1 Experimental setup **1**
1.1 Time measurements . . . 1

2 WL tests **1**
2.1 Input data . . . 1
2.2 Double rank . . . 2
2.3 Lazy . . . 3
2.4 Rank and fail . . . 4
2.5 Double rank, Lazy, and Rank and fail versus Single rank . . . 5
2.6 Maxrep . . . 6

3 Optimizations on parent operations **10**
3.1 Input data . . . 10

4 Genome tests **12**

1 Experimental setup

1.1 Time measurements

We run a program k times with and without the optimization and recorded the sum construction time of the MS and RUNS vectors. The plots report (median, with quartile ranges) of the speedup of each optimized time $t_i^{\texttt{opt}}$ relative to the average non-optimized time in the construction time of the MS vector. In other words

$$d^{(i)} = \frac{\bar{t}_{\texttt{non_opt}}}{t^{(i)}_{\texttt{opt}}}$$

with $\bar{t}_{\texttt{non_opt}} = 1/n \sum t^{(i)}_{\texttt{non_opt}}$, and $i = 1, \ldots, k$.

The boxplots report the raw times.

2 WL tests

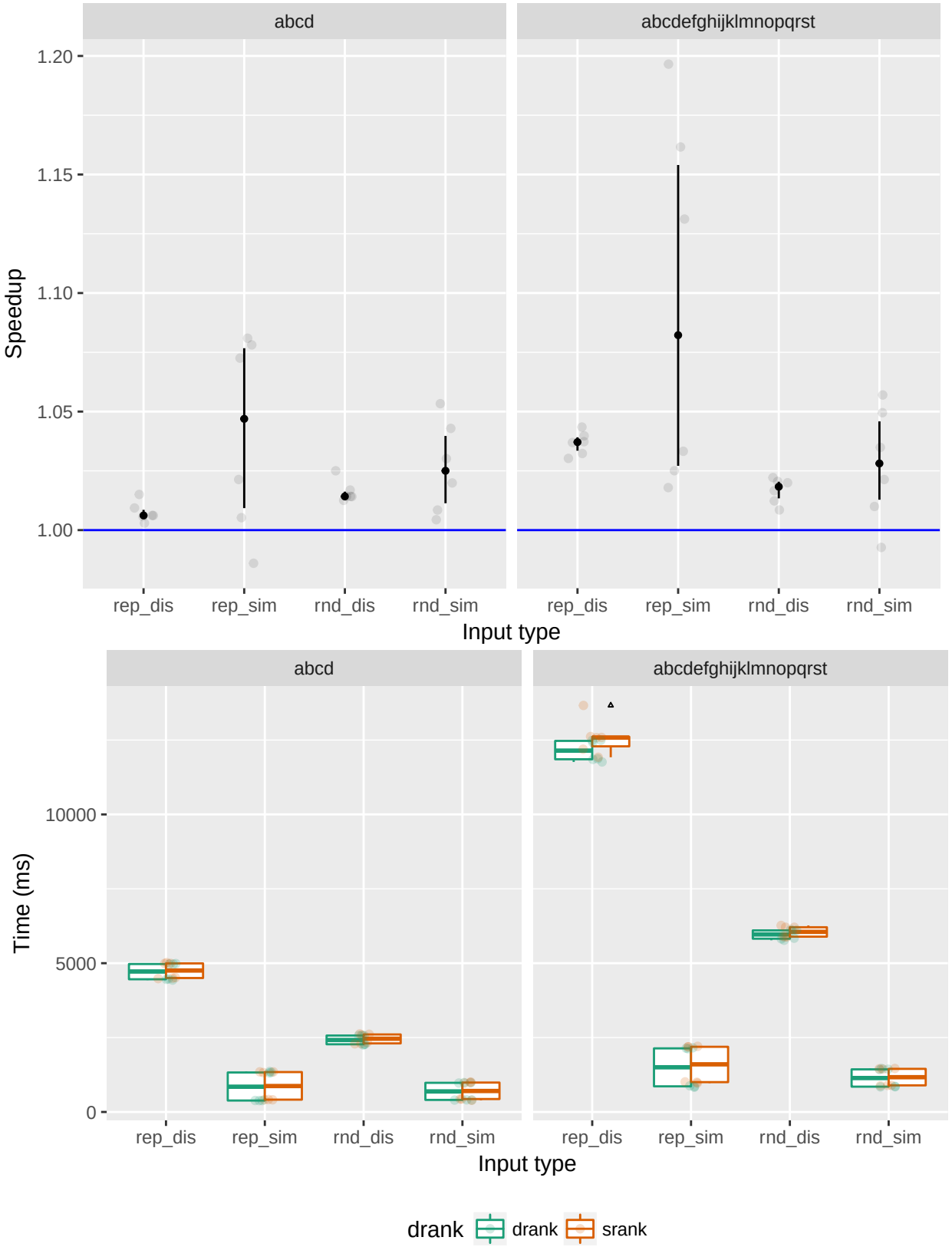
2.1 Input data

We perform tests on the Weiner Link optimizations on 4 types of input.

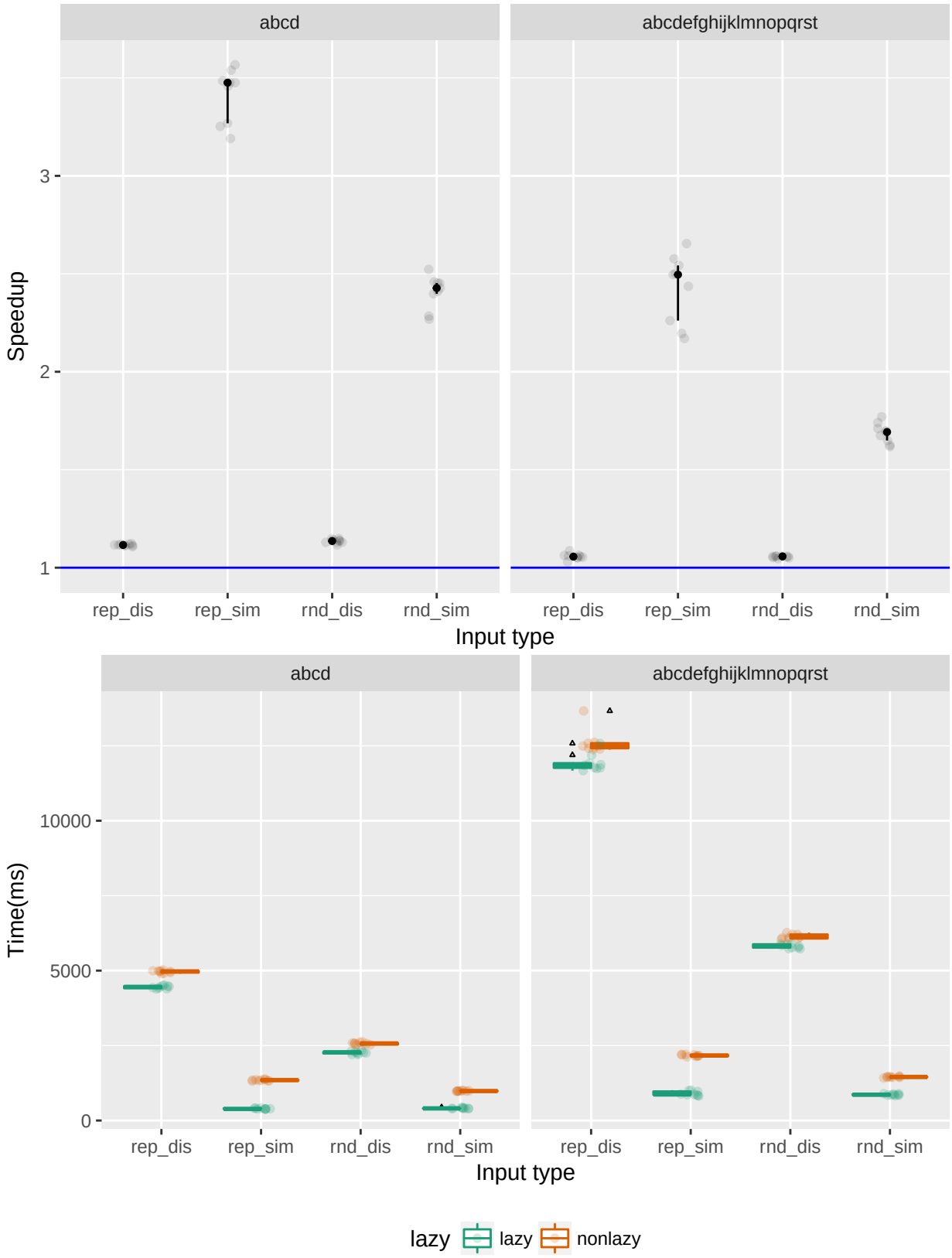
- Index string with repeats, query string random (code: `rep_dis`)
- Index string with repeats, query string similar to index (code: `rep_sim`)
- Index string random, query string random (code: `rnd_dis`)
- Index string random, query string similar to index (code: `rnd_sim`)

Further, we generate all of the above input data for two alphabet sizes: $\Sigma_1| = 4$ and $\Sigma_2| = 20$. For all input types, the index string is of length 100MB and the query 500KB.

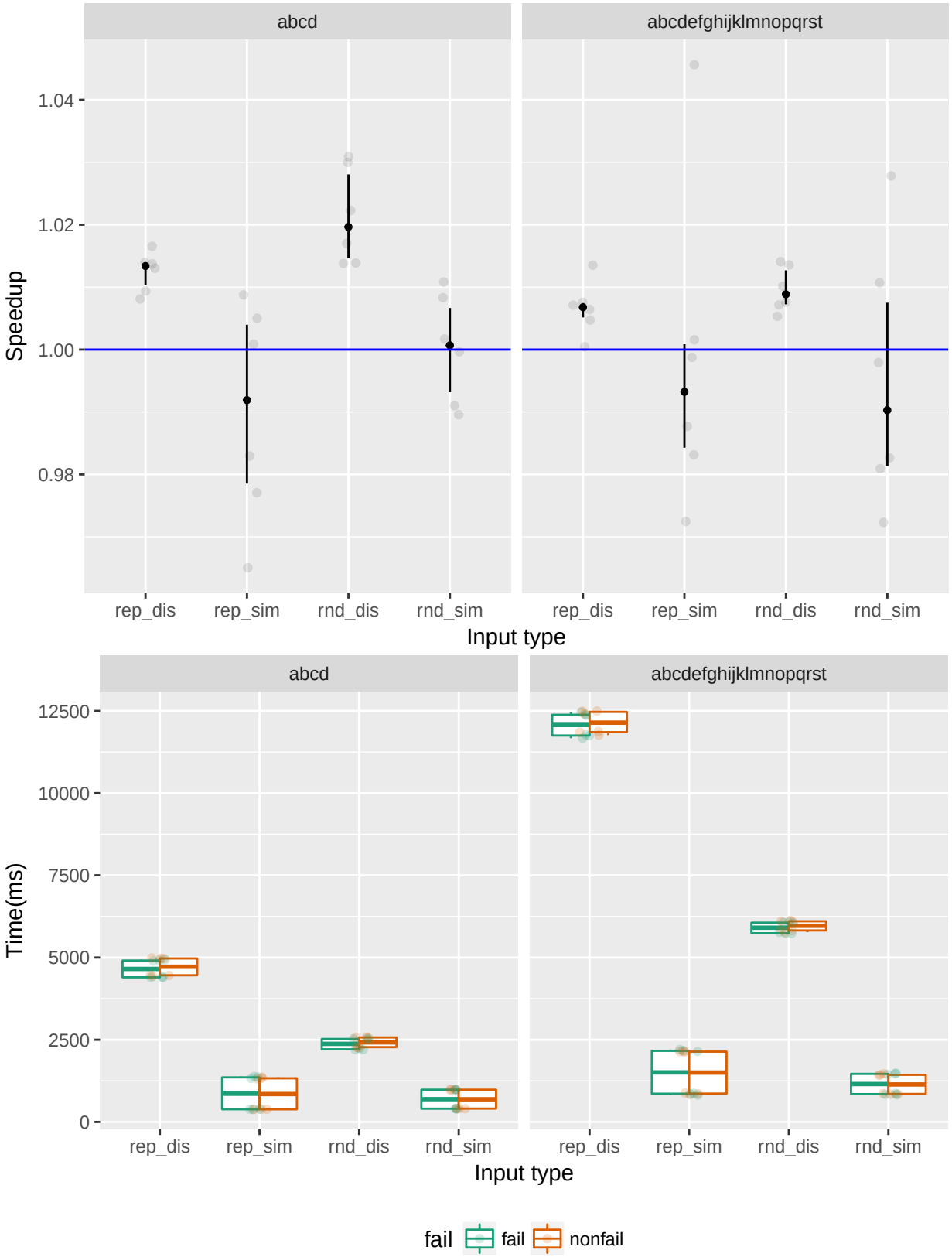
2.2 Double rank



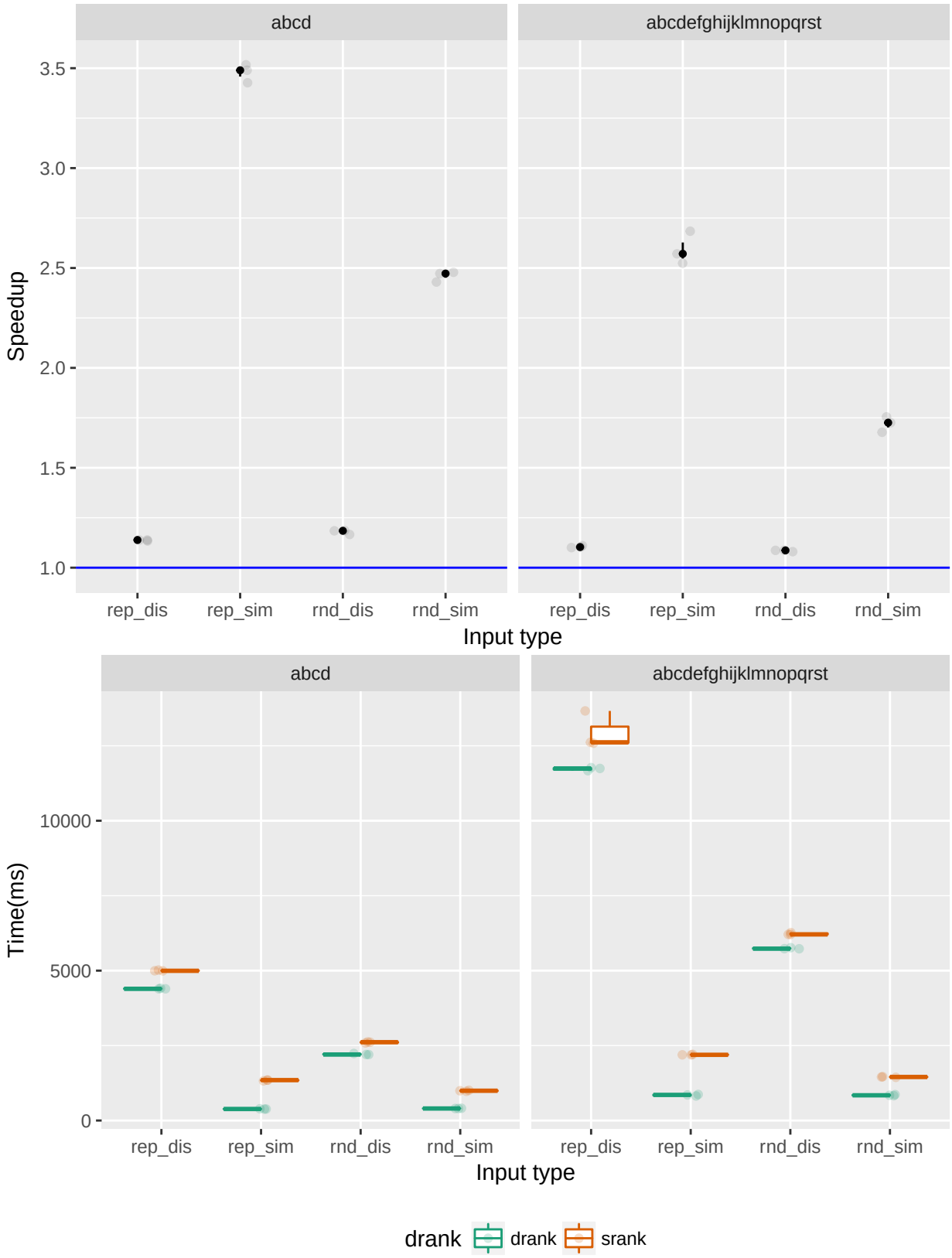
2.3 Lazy


abcd
abcdefghijklmnopqrst
Speedup
3
2
1
rep_dis
rep_sim
rnd_dis
rnd_sim
Input type
Time(ms)
10000
5000
0
lazy
lazy
nonlazy

2.4 Rank and fail

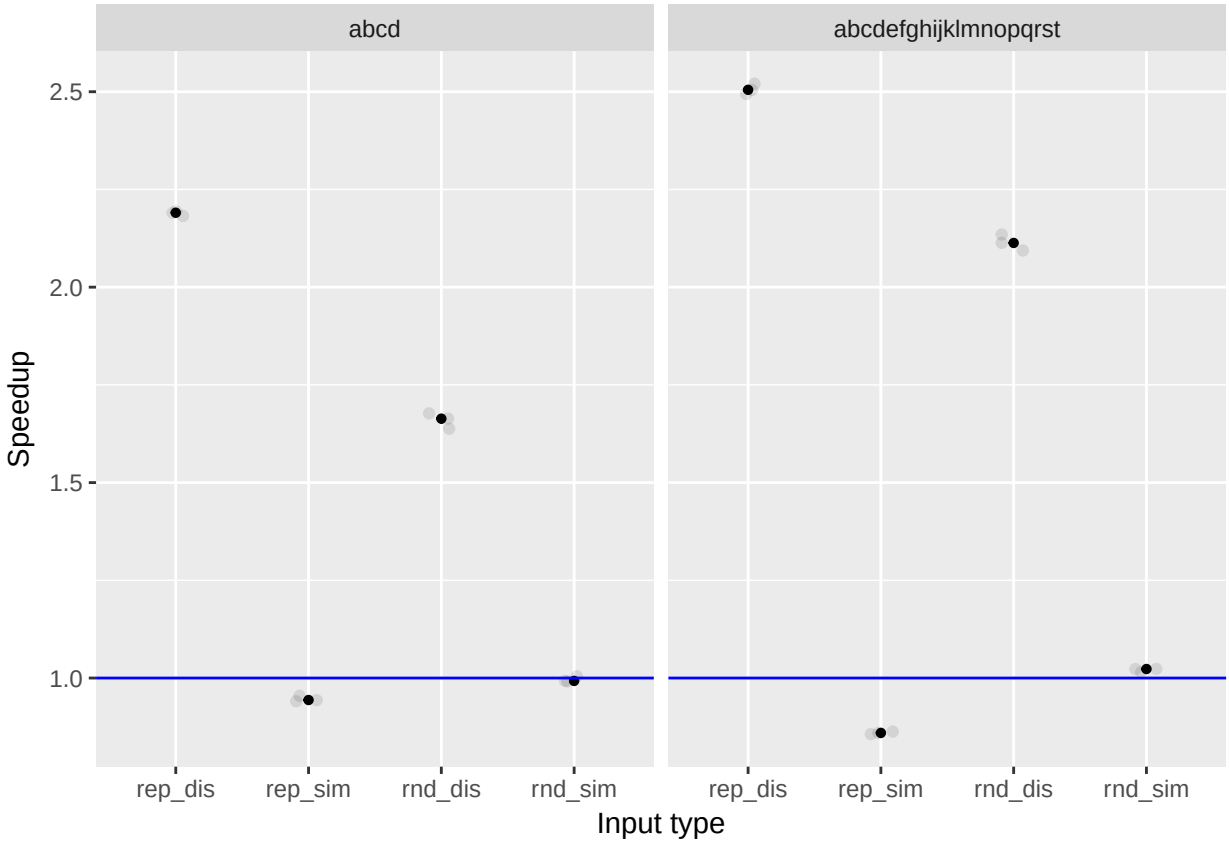


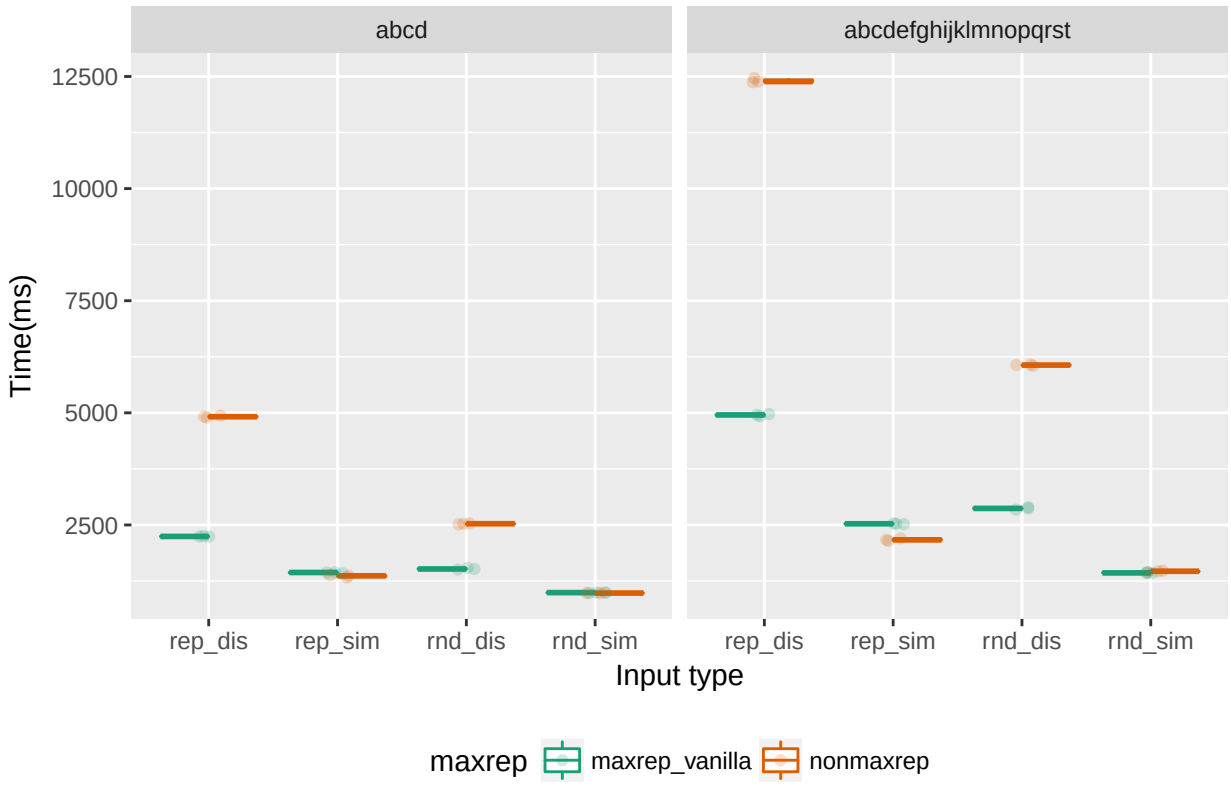
2.5 Double rank, Lazy, and Rank and fail versus Single rank



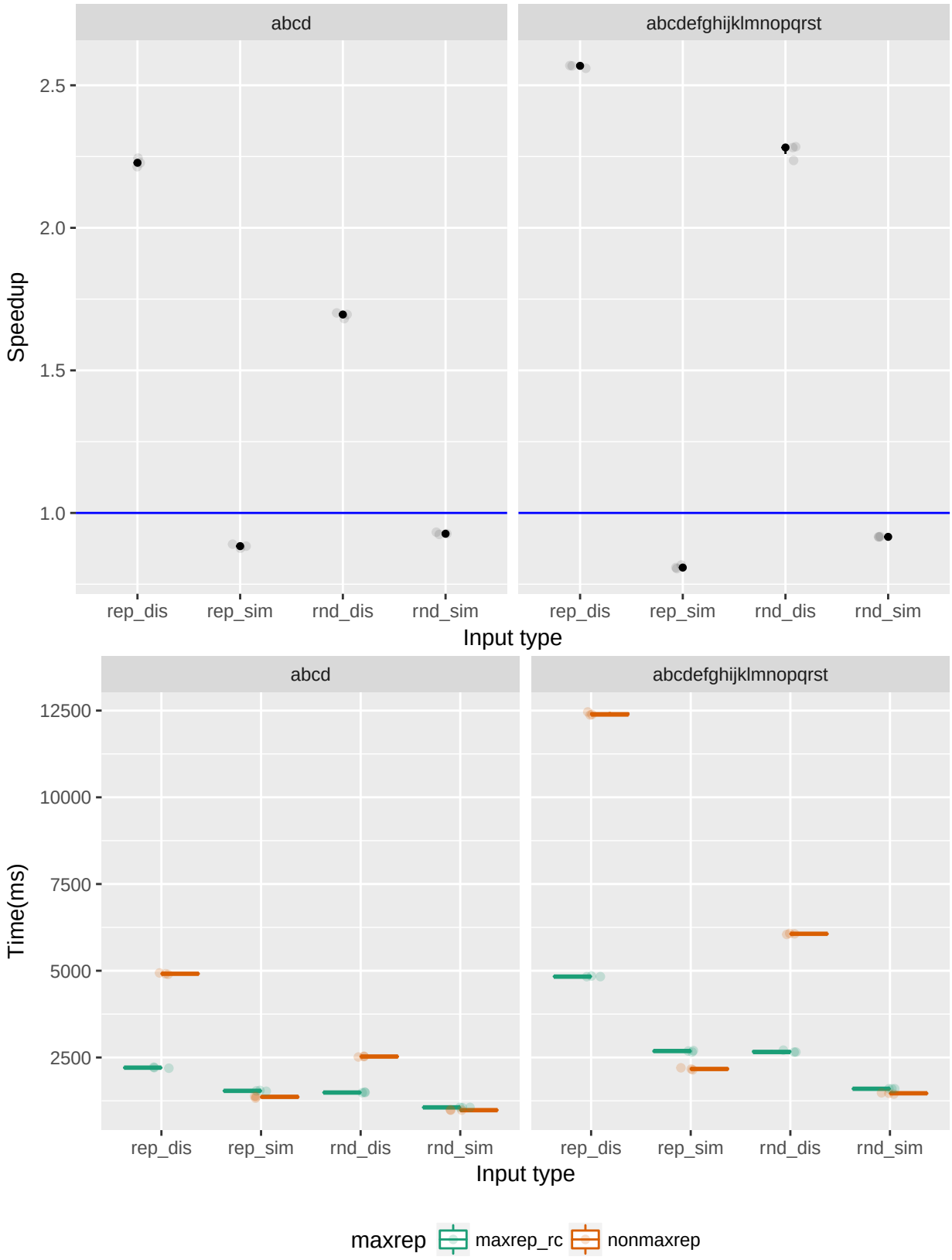
2.6 Maxrep

2.6.1 Vanilla Maxrep vs. non-maxrep

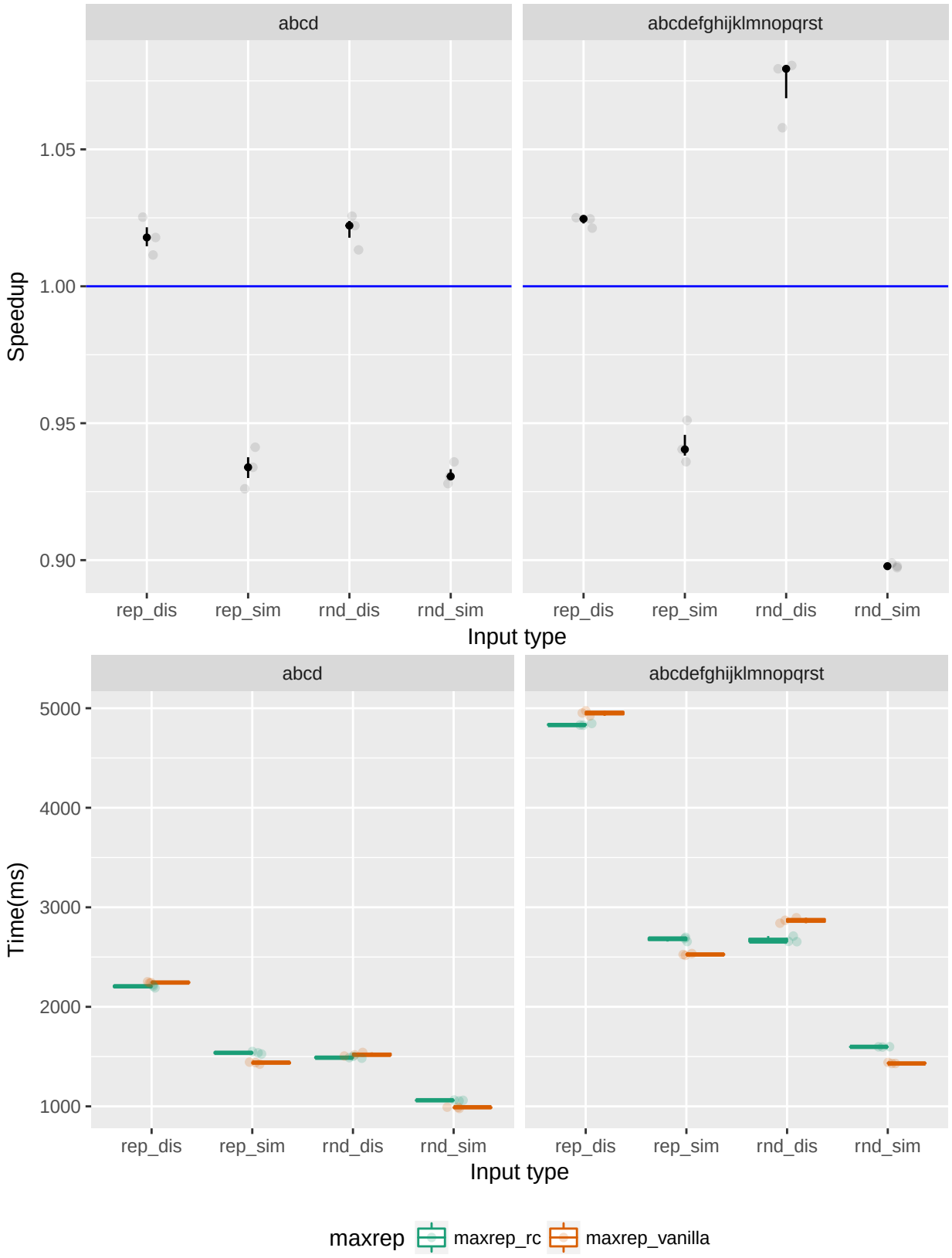



abcd
abcdefghijklmnopqrst
12500
10000
7500
5000
2500
Time(ms)
rep_dis
rep_sim
rnd_dis
rnd_sim
Input type
maxrep
maxrep_vanilla
nonmaxrep

2.6.2 Rank and check maxrep versus non-maxrep



2.6.3 Rank and check maxrep versus Vanilla maxrep



3 Optimizations on parent operations

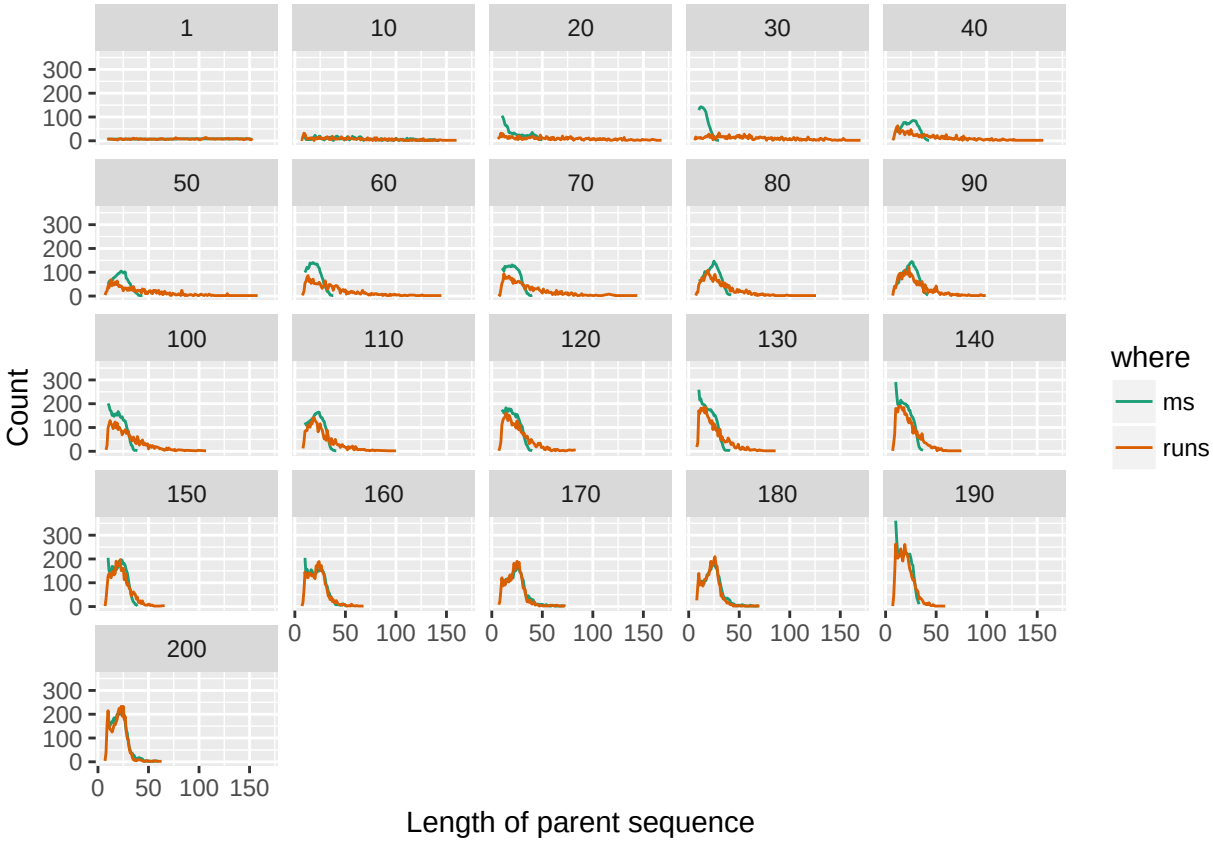
3.1 Input data

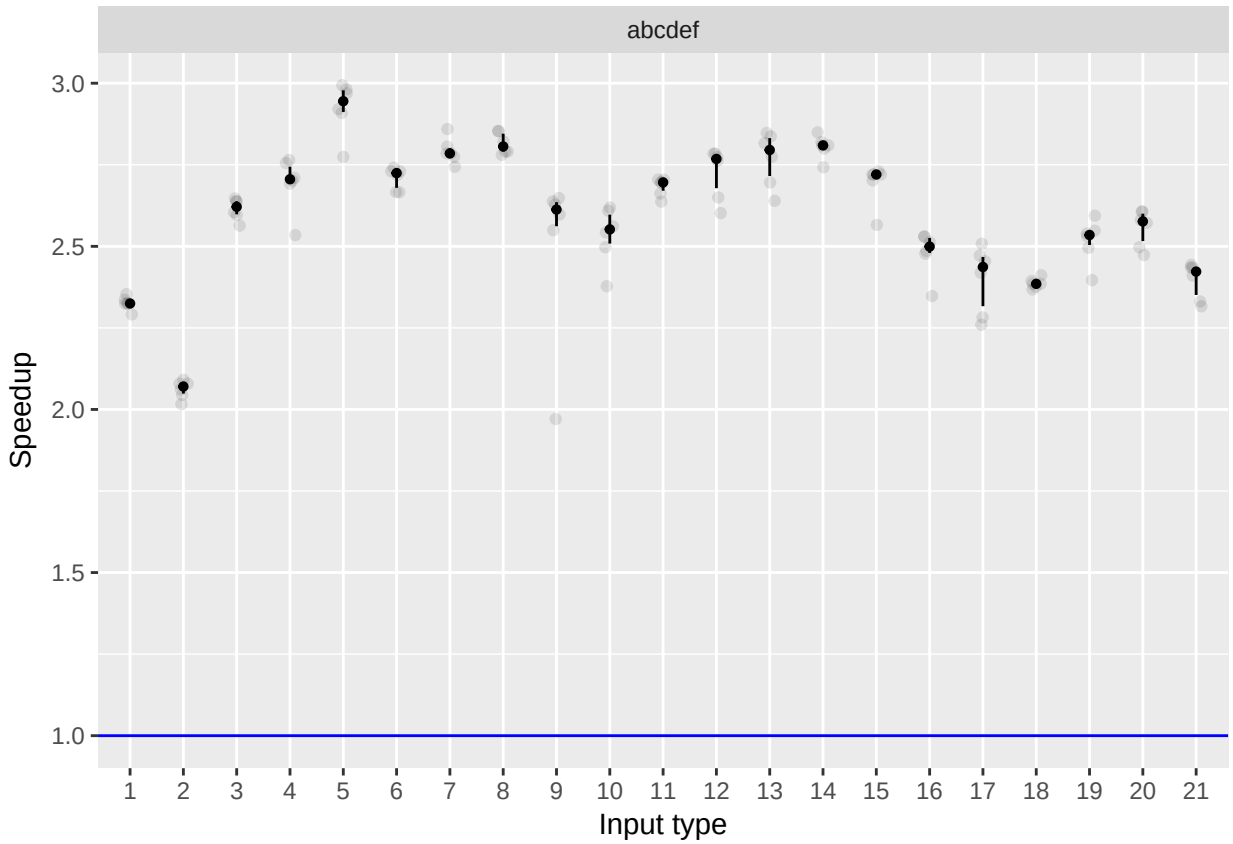
We generate the index input string with repetitions as follows. We generate a random seed block b of length 200. Next, we generate blocks of the same length b_k by introducing k mutations on b. The index string of length 10MB is $b \circ b_k^{(1)} \ldots \circ b_k^{(4999)}$.

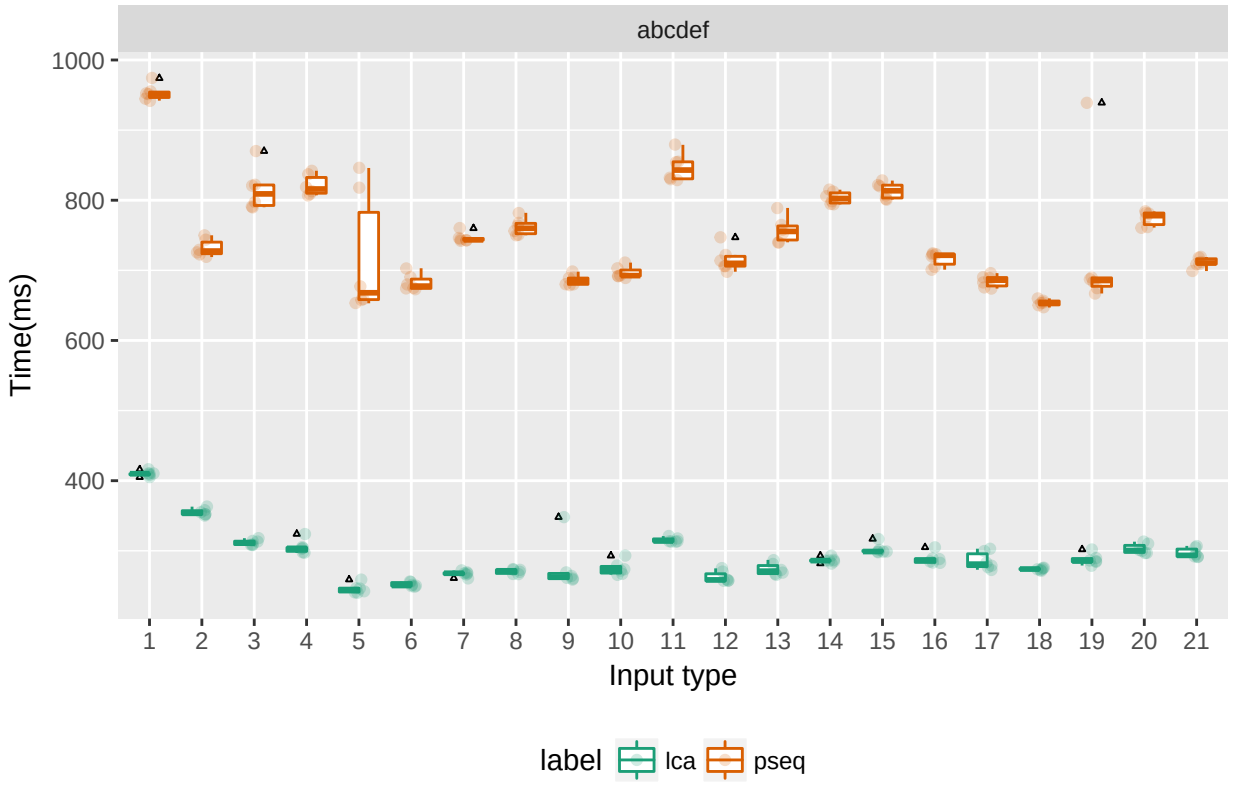
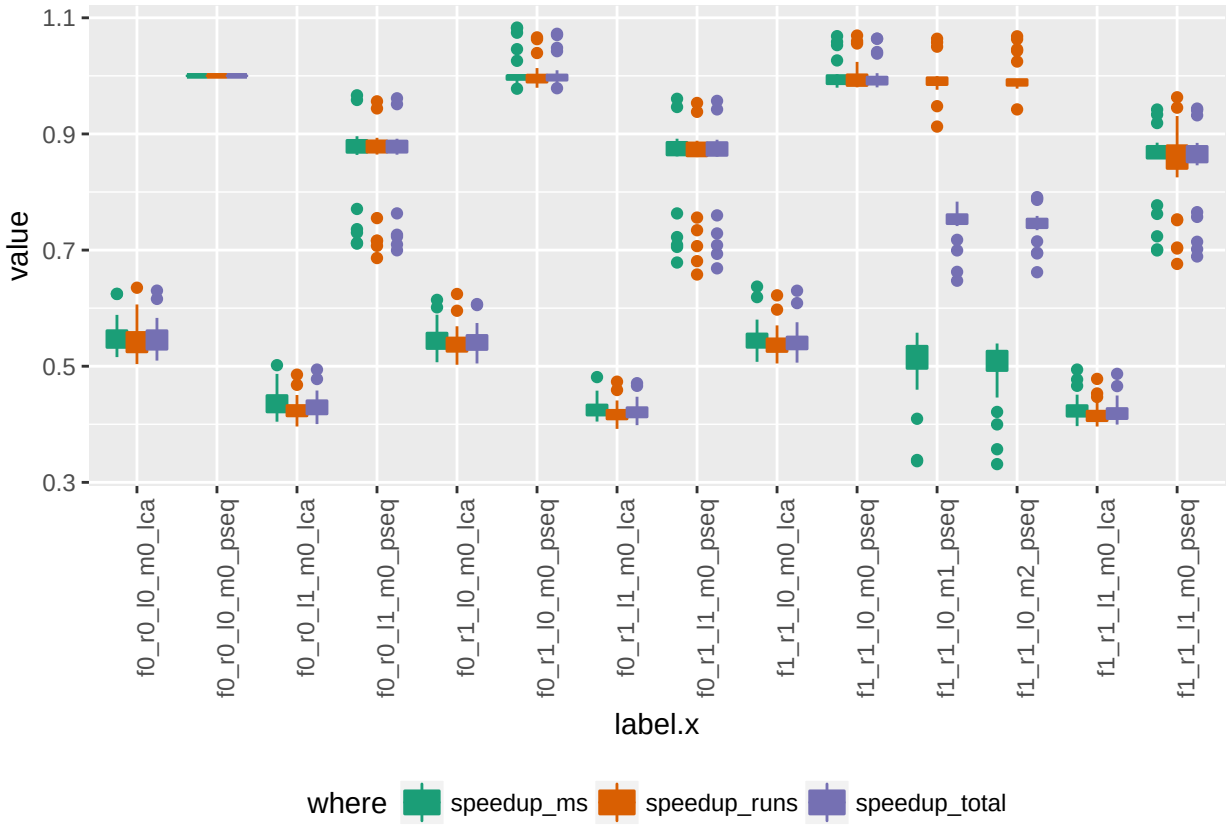
The query string is obtained as a concatenation of labels from nodes of the suffix tree of s. We select nodes with node depth of at least 10 and string length at most 170 for a total string length of 103KB. We separate the labels with a sentinel character that does not appear in s.

Furthermore, we perform experiments for various choices of $1 \geq k \geq |b|$.

The plot below shows a histogram of the length of consecutive parent operations. This quantity is important since the speedup of this optimization is proportional to the length of sequence of parent operations. Importantly, the optimization might not even be beneficial if the length of the sequence of parent operations is less than 3.




abcdef
Speedup
3.0
2.5
2.0
1.5
1.0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
Input type


abcdef
Time(ms)
1000
800
600
400
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
Input type
label
lca
pseq

4 Genome tests