Input properties

O. Denas

9/29/2017

Input properties

Experiments are performed on the following inputs:

Table 1: Input datasets

slen	tlen	alp	rep_mut	sim_mut	num_blocks	block_len	s_mutation_rate	t_mutation_rate
100'000	10'000	5	100	10000	1'000	100	1,00	1,0000
100'000	10'000	5	100	1000	1'000	100	1,00	0,1000
100'000	10'000	5	100	100	1'000	100	1,00	0,0100
100'000	10'000	5	100	1	1'000	100	1,00	0,0001
100'000	10'000	5	10	10000	1'000	100	0,10	1,0000
100'000	10'000	5	10	1000	1'000	100	0,10	0,1000
100'000	10'000	5	10	100	1'000	100	0,10	0,0100
100'000	10'000	5	10	1	1'000	100	0,10	0,0001
100'000	10'000	5	1	10000	1'000	100	0,01	1,0000
100'000	10'000	5	1	1000	1'000	100	0,01	0,1000
100'000	10'000	5	1	100	1'000	100	0,01	0,0100
100'000	10'000	5	1	1	1'000	100	0,01	0,0001
100'000	10'000	20	100	10000	1'000	100	1,00	1,0000
100'000	10'000	20	100	1000	1'000	100	1,00	0,1000
100'000	10'000	20	100	100	1'000	100	1,00	0,0100
100'000	10'000	20	100	1	1'000	100	1,00	0,0001
100'000	10'000	20	10	10000	1'000	100	0,10	1,0000
100'000	10'000	20	10	1000	1'000	100	0,10	0,1000
100'000	10'000	20	10	100	1'000	100	0,10	0,0100
100'000	10'000	20	10	1	1'000	100	0,10	0,0001
100'000	10'000	20	1	10000	1'000	100	0,01	1,0000
100'000	10'000	20	1	1000	1'000	100	0,01	0,1000
100'000	10'000	20	1	100	1'000	100	0,01	0,0100
100'000	10'000	20	1	1	1'000	100	0,01	0,0001

The index (aka S) is generated by the concatenation of $(s_1, \ldots, s_k)$ where $\sum_i |s_i| = |S|$. The query (aka T) is generated by mutating $S[1..|T|]$.

Description of columns:

- slen/tlen: length of the index/query string. I.e., $|S|$ and $|T|$.
- alp: the alphabet length
- rep_mut: number of mutations introduced to block s_1 to obtain blocks s_i with $1 < i \leq k$.
- sim_mut: number of mutations introduced to $S[1..|T|]$ in order to obtain T – I.e., T differs from S by `sim_mut` characters.
- num_blocks: number of blocks to obtain S (k above).
- block_len: length of s_i for $1 \leq i \leq n$
- s_mutation_rate: `sim_mut` / `block_len`
- t_mutation_rate: `sim_mut` / `tlen`

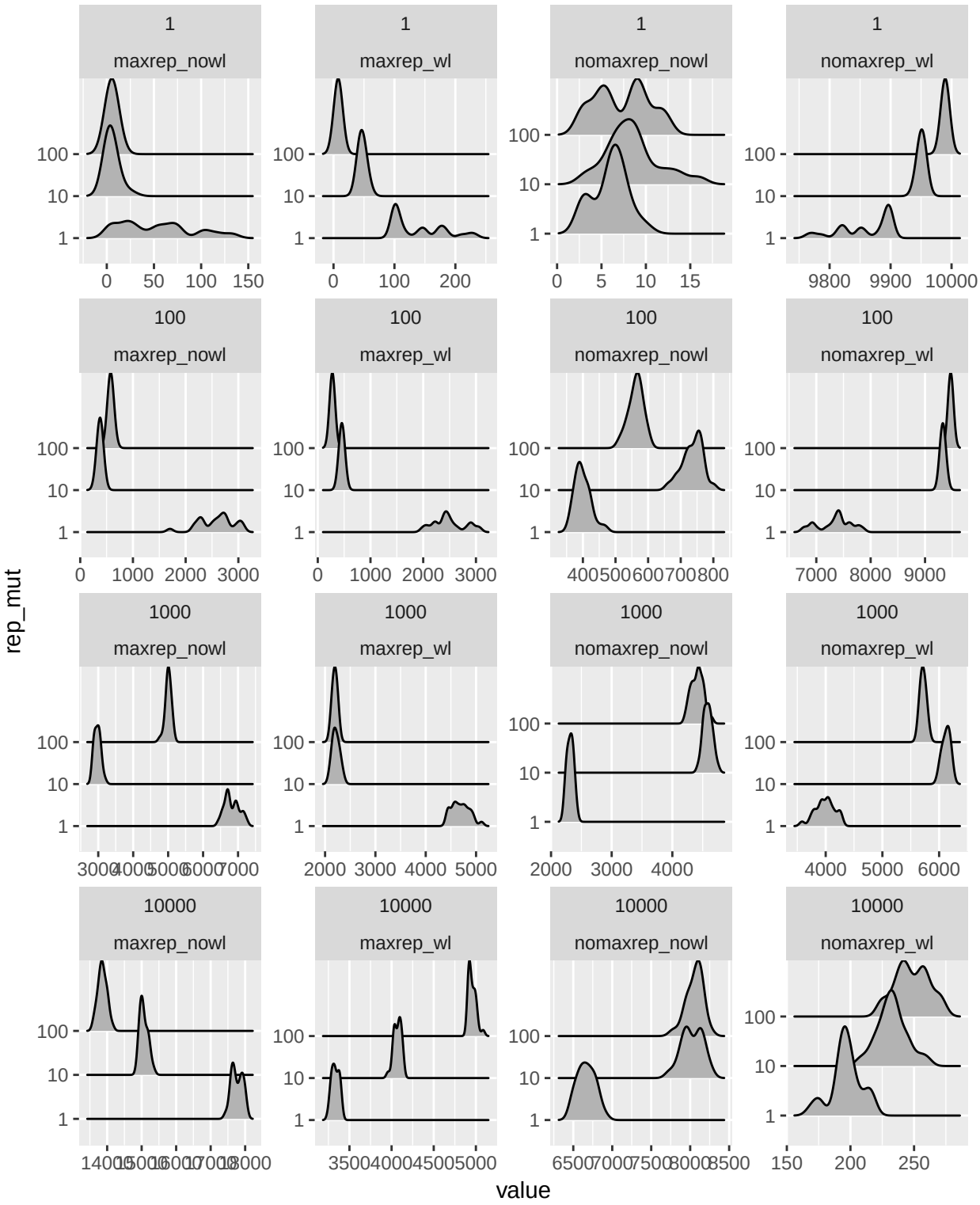
Notice that the row with `rep_mut=rep_sim=1` has `S` that is a almost a perfect repeat of blocks, and `T` almost a perfect copy of it.

On the other hand, `rep_mut=block_len` and `rep_sim=` $|T|$ has `S` pretty much random (since we are mutating all chars of a block) and `T` having mismatches everywhere.

Node properties

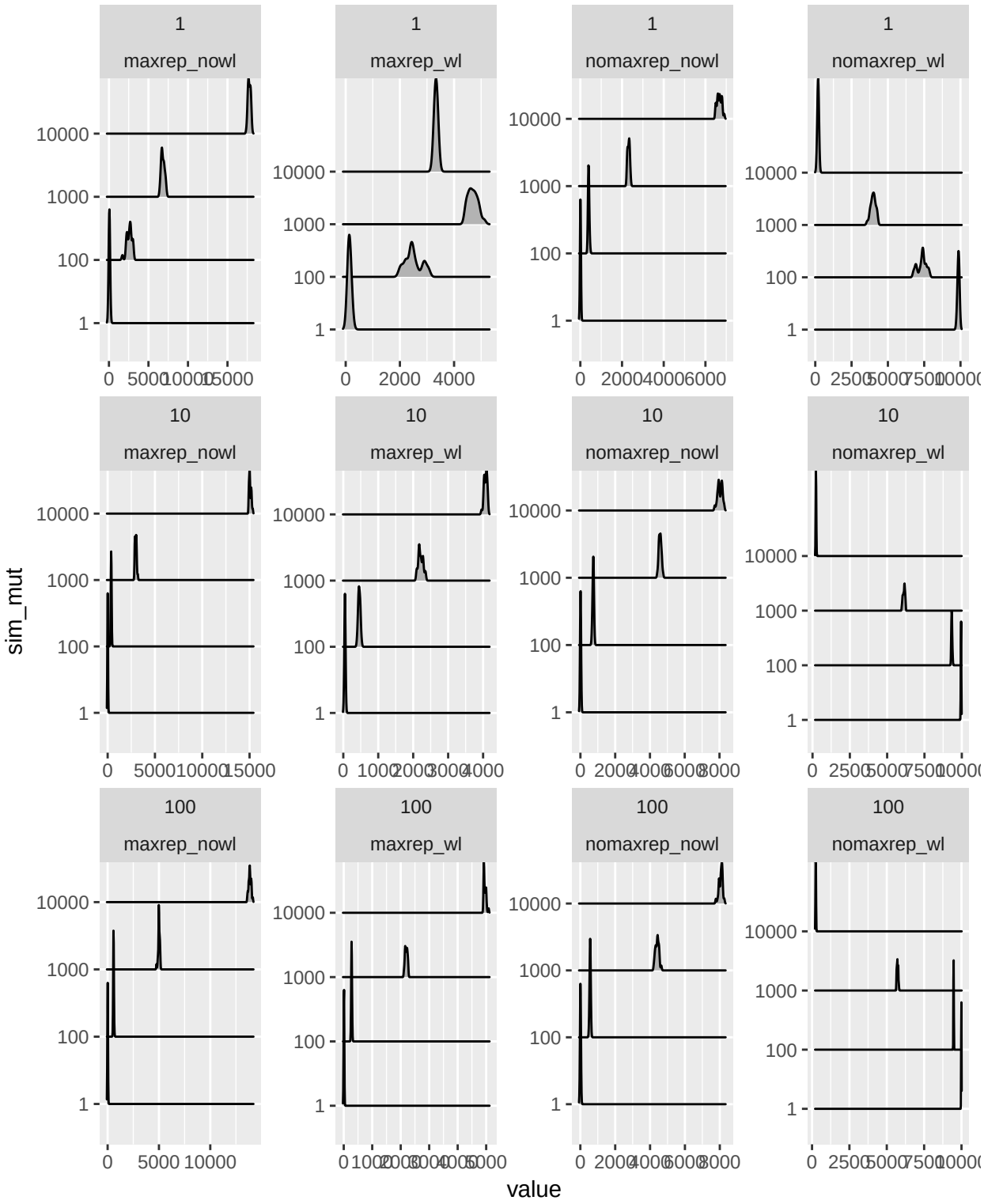
Before making a `wl(v, c)` call register whether `v` is a maximal repeat, and whether the Weiner link exists or not.

```
## Picking joint bandwidth of 7.23
## Picking joint bandwidth of 7.44
## Picking joint bandwidth of 0.94
## Picking joint bandwidth of 7.57
## Picking joint bandwidth of 60.9
## Picking joint bandwidth of 53.5
## Picking joint bandwidth of 11.4
## Picking joint bandwidth of 52.2
## Picking joint bandwidth of 57.3
## Picking joint bandwidth of 46.3
## Picking joint bandwidth of 36
## Picking joint bandwidth of 44.8
## Picking joint bandwidth of 63.7
## Picking joint bandwidth of 21.1
## Picking joint bandwidth of 56.1
## Picking joint bandwidth of 4.35
```



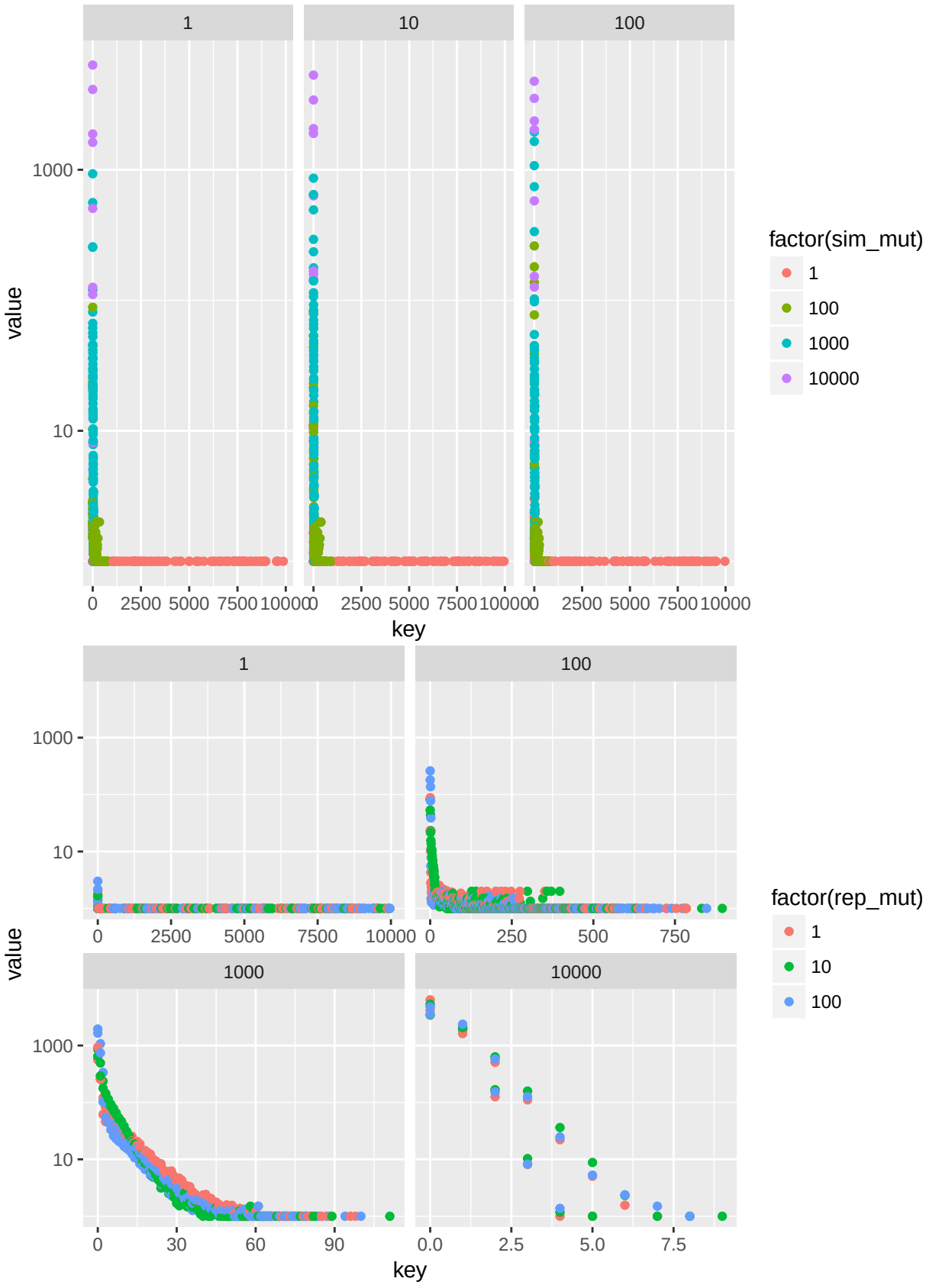

```
## Picking joint bandwidth of 88.9
## Picking joint bandwidth of 67.5
## Picking joint bandwidth of 22.6
## Picking joint bandwidth of 61.3
```

```
## Picking joint bandwidth of 27.9
## Picking joint bandwidth of 17.5
## Picking joint bandwidth of 28.7
## Picking joint bandwidth of 12.7
## Picking joint bandwidth of 25.1
## Picking joint bandwidth of 11.2
## Picking joint bandwidth of 27
## Picking joint bandwidth of 7.74
```



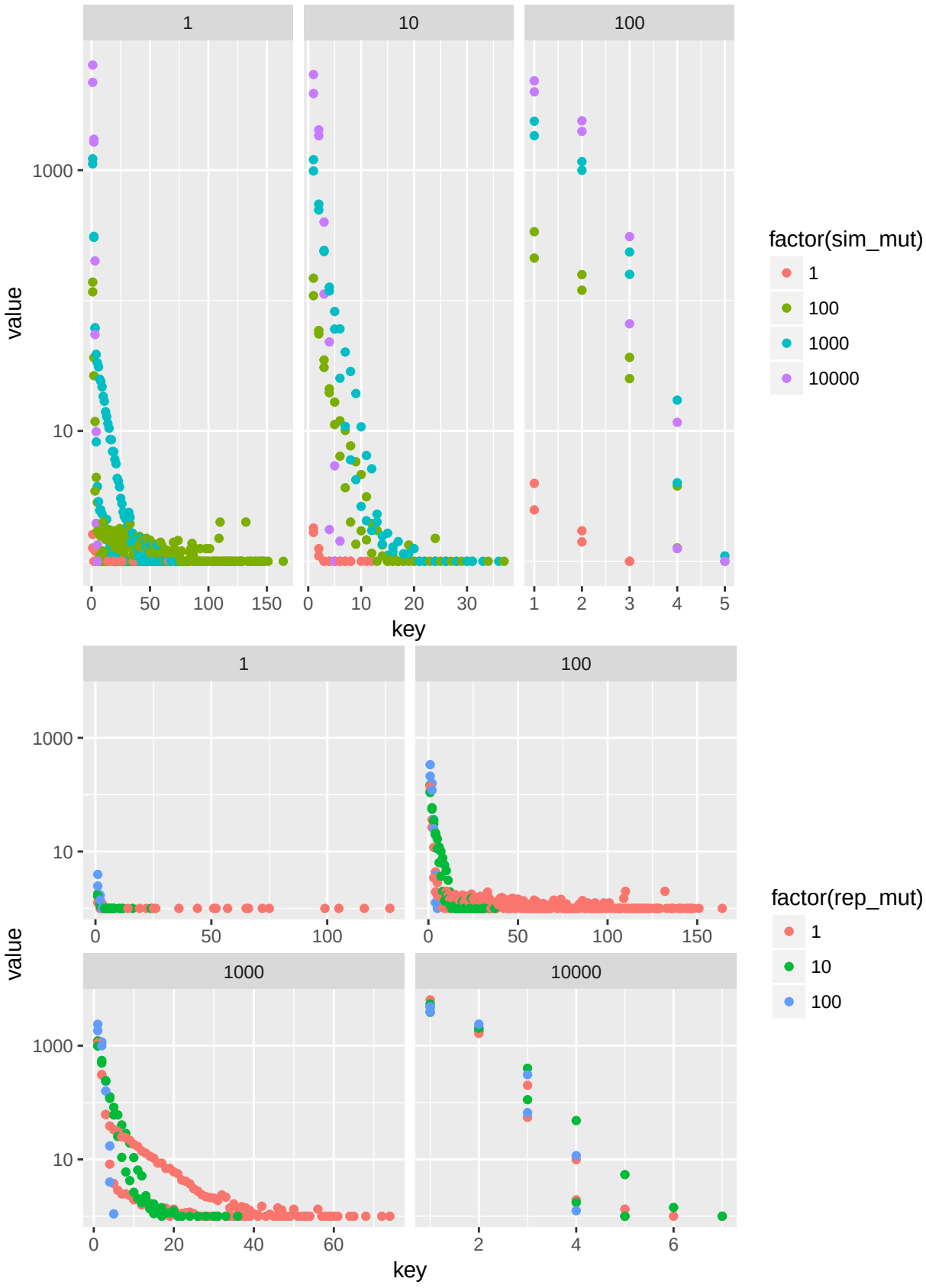
Consecutive wl() calls

During the MS vector construction, count the number of consecutive wl() calls due to matches between reversed indexed string and the query. In other words count the k-length iterations of the while cycle.


1
10
100
1000
10
value
0
2500
5000
7500
10000
key
factor(sim_mut)
1
100
1000
10000
1
100
1000
10000
250
500
750
30
60
90
0.0
2.5
5.0
7.5
factor(rep_mut)
1
10
100

Consecutive parent() calls

During the MS vector construction, count the number of consecutive `parent()` calls after a failed `wl()` call.


1
10
100
1000
10
value
0
50
100
150
0
10
20
30
1
2
3
4
5
key
factor(sim_mut)
1
100
1000
10000
1
100
1000
10000
1000
10
value
0
50
100
0
50
100
150
0
20
40
60
2
4
6
key
factor(rep_mut)
1
10
100