Run `haplinSlide`

With `winlength = 1`

Sliding Haplin run, single SNPs

Load data:

```
pres.data <- genDataLoad(filename = "data_preprocessed",
 dir.in = "data")
```

SLIDING HAPLIN RUN, SINGLE SNPS

`haplinSlide` **run:**

- Single SNP analyses
- Simply performs a sequence of Haplin analyses
- Mostly the same arguments as `haplin`
- Can choose number of cpu's to use in parallel
- Returns a list. Each element is either

  `TABLE.OUTPUT = F`: A result of running `haplin(.....)`

  `TABLE.OUTPUT = T`: A result of running `haptable(haplin(.....))`

```
result <- haplinSlide(data = pres.data, markers = 1:10,
 use.missing = T, table.output = F)
```

INSPECT RESULTS FROM haplinSlide, WHEN table.output = F

Names of markers used:

```
names(result)
```

```
[1] "rs1"  "rs3"  "rs5"  "rs6"  "rs7"  "rs8"  "rs9"
[8] "rs10" "rs11" "rs12"
```

Each element can be summarized as usual:

```
summary(result[["rs5"]])
plot(result[["rs5"]])
haptable(result[["rs5"]])
```

INSPECT RESULTS FROM haplinSlide, WHEN table.output = F

Sequence of plots, one for each result:

```
pdf(file = "haplin_plots.pdf") # Open a pdf file
lapply(result, plot) # Apply plot to every element
dev.off() # Close pdf file
```

More summaries than you want(?):

```
sink(file = "haplin_summaries.txt") # Open a txt file
lapply(result, summary) # Apply full summary to every element
sink() # Close txt file
```

SLIDING HAPLIN RUN, SINGLE SNPS

`haplinSlide` **run:**

```
result <- haplinSlide(data = pres.data, markers = 1:10,
use.missing = T, table.output = T)
```

INSPECT RESULTS FROM haplinSlide, WHEN table.output = T

Names of markers used:

```
names(result)
```

```
 [1] "rs1"  "rs3"  "rs5"  "rs6"  "rs7"  "rs8"  "rs9"
 [8] "rs10" "rs11" "rs12"
```

Each element is a haptable:

```
result[["rs5"]]
```

```
  marker alleles    counts    HWE.pv Original After.rem.NA After.re
1    rs5     a/T 1196/1978 0.3014432      559           559
2   <NA>    <NA>      <NA>        NA      559           559

...etc.
```

INSPECT RESULTS FROM haplinSlide, WHEN table.output = T

Haptables can be plotted:

```
plot(result[["rs5"]])
```

But standard summaries are not available (nor needed, really).

Recommendation: When running long scans, use:

- `winlength = 1` (default, one SNP at a time)
- `table.output = T` (default, saves memory)
- `response = "mult"`
  (not default, but increases speed and protects better against HWE deviations)

Run `haplinSlide`

With `winlength > 1`

SLIDING WINDOW APPROACH

gene	snp
ABCA1	rs2230806
ABCA1	rs2230808
ABCA1	rs2472384
ABCA1	rs2487054
ABCA1	rs2740479
ABCA1	rs3858075
ABCA1	rs4149272
ABCA1	rs4149313

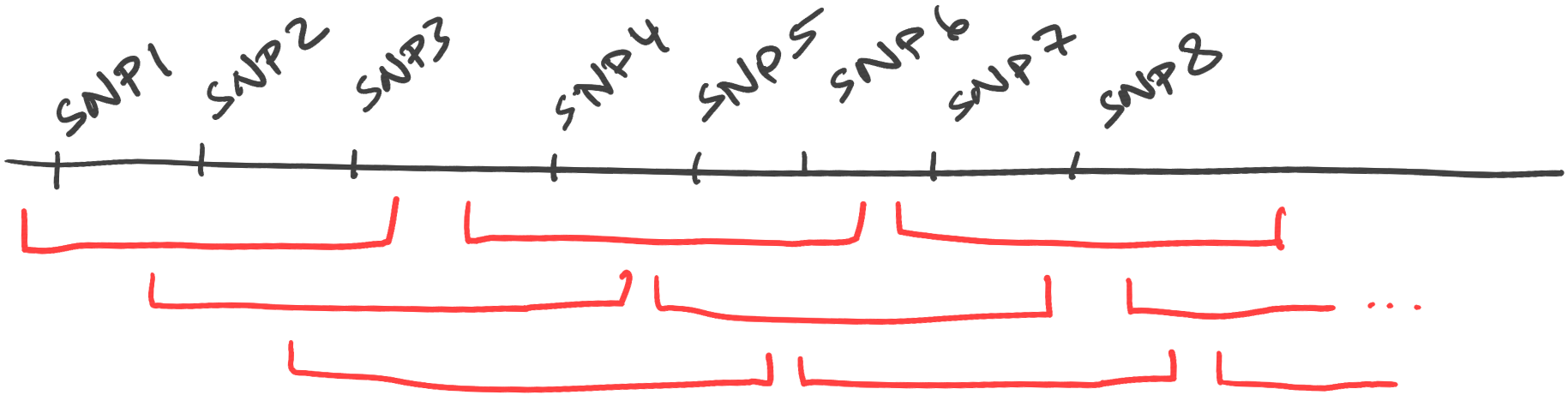
Problem:

- More than 4 SNPs demand heavy computations
- Long haplotypes “break up” (don’t really exist in population)

Solution:

- Sliding window approach
- “Bracketing” of mutations

SLIDING WINDOW APPROACH



- Sliding window, haplotypes of length, say, 2, 3, or 4.

Syntax:

```
result <- haplinSlide(data = pres.data, markers = 1:10,
  use.missing = T, table.output = T, response = "mult",
  reference = "ref.cat", winlength = 2)
```

SLIDING WINDOW APPROACH

```
length(result)
```

```
[1] 9
```

```
names(result)
```

```
[1] "rs1-rs3"   "rs3-rs5"   "rs5-rs6"   "rs6-rs7"   "rs7-rs8"   "rs
[7] "rs9-rs10"  "rs10-rs11" "rs11-rs12"
```

SLIDING WINDOW APPROACH

```
result[[3]]
```

```
  marker alleles    counts    HWE.pv Original After.rem.NA
1    rs5     a/T 1196/1978 0.3014432      559          559
2    rs6     c/G  276/2906 0.3472571      559          559
3   <NA>    <NA>      <NA>        NA      559          559

  After.rem.Mend.inc. After.rem.unused.haplos pv.overall
1                 559                     551  0.0658392
2                 559                     551  0.0658392
3                 559                     551  0.0658392

  haplos  haplofreq haplofreq.lower haplofreq.upper
1    T-c 0.09936227      0.08257788       0.1195111
2    a-G 0.37384676      0.34428285       0.4040745
3    T-G 0.52625245      0.49551534       0.5570534

  reference   RR.est.  RR.lower  RR.upper RR.p.value
1         - 0.6858970 0.4961210 0.9416996     0.0218
2         - 0.9564963 0.7952565 1.1494720     0.6312
3       ref 1.0000000 1.0000000 1.0000000     1.0000

  RRdd.est. RRdd.lower RRdd.upper RRdd.p.value
1 0.4704547  0.2461360  0.8867981       0.0218
2 0.9148851  0.6324328  1.3212859       0.6312
3 1.0000000  1.0000000  1.0000000       1.0000
```

CORRECTING FOR MULTIPLE TESTING WITHIN A REGION

- `haplinSlide` provides a series of tests
- Each test has an overall p-value
- P-values are correlated because of LD and overlapping windows
- The p-values can be merged into a single joint p-value for a region, for instance within a gene
- Use the function `suest`, which adjusts for dependence
- Better than Bonferroni and Šidák
- See `multiple_testing.pdf`

Run `haplinSlide`

With `cpus > 1`

haplinSlide IN PARALLEL

- "Trivial" to run computations in parallel.
- Argument `cpus` invokes `parallel` package when > 1.

```
result <- haplinSlide(data = pres.data, markers = 1:100,
  use.missing = T, table.output = T, response = "mult",
  reference = "ref.cat", winlength = 1, cpus = 4)
```

```
names(result)
```

```
 [1] "rs1"   "rs3"   "rs5"   "rs6"   "rs7"   "rs8"
 [7] "rs9"   "rs10"  "rs11"  "rs12"  "rs13"  "rs16"
[13] "rs17"  "rs18"  "rs19"  "rs20"  "rs21"  "rs22"
[19] "rs23"  "rs26"  "rs27"  "rs28"  "rs29"  "rs30"

...
```

haplinSlide IN PARALLEL, REPORTING PROGRESS

- Minor issue: Under Windows there is no report until finished.
- Fix: Use `slaveOutfile` argument to specify a report filename.
- And refresh file along the way.

```
result <- haplinSlide(data = pres.data, markers = 1:100,
  use.missing = T, table.output = T, response = "mult",
  reference = "ref.cat", winlength = 1, cpus = 4,
  slaveOutfile = "temp.txt")
```

haplinSlide IN PARALLEL, REPORTING PROGRESS

```
starting worker pid=10216 on localhost:11415 at 14:08:11.984
starting worker pid=2848 on localhost:11415 at 14:08:12.228
starting worker pid=460 on localhost:11415 at 14:08:12.441
starting worker pid=18164 on localhost:11415 at 14:08:12.656

--- Running haplinSlide using 4 cpu's ---
Running Haplin on Window 'rs1' (1/100)...: Running Haplin on Windo
Running Haplin on Window 'rs33' (27/100)...: Running Haplin on Wind
Running Haplin on Window 'rs103' (79/100)...: done
Running Haplin on Window 'rs35' (29/100)...: Running Haplin on Wind
Running Haplin on Window 'rs104' (80/100)...: done
Running Haplin on Window 'rs69' (55/100)...: Running Haplin on Wind
Running Haplin on Window 'rs105' (81/100)...: Running Haplin on Win
Running Haplin on Window 'rs107' (82/100)...:
.....
 --- haplinSlide has completed ---
```

Full `haptable`

COLLECT RESULTS FROM haplinSlide IN A FULL haptable

```
result <- haplinSlide(data = pres.data,
 markers = 1:100, use.missing = T, table.output = T,
 response = "mult", reference = "ref.cat",
 winlength = 1, cpus = 4)
```

Each element is a haptable:

```
result[["rs5"]]
```

```
  marker alleles    counts    HWE.pv Original After.rem.NA
1    rs5     a/T 1196/1978 0.3014432      559          559
2   <NA>    <NA>      <NA>        NA      559          559
  After.rem.Mend.inc. After.rem.unused.haplos pv.overall
1                 559                     559  0.9396003
2                 559                     559  0.9396003
  haplos haplofreq haplofreq.lower haplofreq.upper
1      a 0.3796496       0.3500153       0.4100448
2      T 0.6203504       0.5899552       0.6499847

... etc.
```

COLLECT RESULTS FROM haplinSlide IN A FULL haptable

More useful(?) with a stacked table:

```
result1 <- haptable(result)
head(result1)
```

```
  window row.no marker alleles    counts    HWE.pv Original
1    rs1      1    rs1     c/G  146/3040 0.7075356      559
2    rs1      2   <NA>    <NA>      <NA>        NA      559
3    rs3      1    rs3     A/t  2460/706 0.3620846      559
4    rs3      2   <NA>    <NA>      <NA>        NA      559
5    rs5      1    rs5     a/T 1196/1978 0.3014432      559
6    rs5      2   <NA>    <NA>      <NA>        NA      559
  After.rem.NA After.rem.Mend.inc. After.rem.unused.haplos
1          559                 559                    559
2          559                 559                    559
3          559                 559                    559
4          559                 559                    559
5          559                 559                    559
6          559                 559                    559
```

COLLECT RESULTS FROM haplinSlide IN A FULL haptable

A bit neater with a stacked table, one row per SNP:

```
result1 <- toDataFrame(result, reduce = T)
head(result1)
```

	element	marker	alleles	counts	HWE.pv	Original	After.rem.	NA
1	rs1	rs1	c/G	146/3040	0.7075356	559	559	
3	rs3	rs3	A/t	2460/706	0.3620846	559	559	
5	rs5	rs5	a/T	1196/1978	0.3014432	559	559	
7	rs6	rs6	c/G	276/2906	0.3472571	559	559	
9	rs7	rs7	c/G	1274/1898	0.3554434	559	559	
11	rs8	rs8	A/g	2330/836	0.2787112	559	559	

	After.rem.Mend.inc.	After.rem.unused.haplos	pv.overall	haplos
1	559	559	0.47335197	c
3	559	559	0.67199029	t
5	559	559	0.93960032	a
7	559	559	0.04892800	c
9	559	559	0.03298692	c
11	559	559	0.20616500	g

haptable VERSUS toDataFrame

haptable:

- Applies to results from haplin
- Applies to results from haplinSlide, whether table.output = T or table.output = F
- When applied to results from haplinSlide, will always produce stacked table with all rows included.

toDataFrame:

- Applies only to results from haplinSlide, when table.output = T
- Produces stacked table
- But option reduce = T will remove redundant rows when winlength = 1

(reduce = T will at some point be incorporated in haptable)