Volkan Lab Faire-Seq

Charlie Soeder

8/1/2019

Contents

1 Introduction

words words

2 Materials, Methods, Data, Software

generic overview words

2.1 Reference Genomes

The dm6 reference genome was used for read alignment:

Size and Consolidation of Reference Genomes

Drosophila Melanogaster	
measure	dm6
number bases	$138M$
number contigs	8

2.2 Reference Annotations

Reference annotations were used to locate features within the genome for comparison:

In addition to the full annotations, subsets containing prespecified genes of interest will also be used.

Here are those subsets and their sizes:

Predefined Subsets of Gene Annotation

measure	histoneMod	ionotropic	mating	nervSysDev	synapseSig
annotated count	8	246	3	90	1
avg size	$5.9K$	$15.2K$	$1.7K$	$19.8K$	$27.1K$
percent annotation size	0.0%	3.7%	0.0%	1.7%	0.0%
percent genome size	0.0%	2.7%	0.0%	1.3%	0.0%
percent of annotations	0.0%	1.4%	0.0%	0.5%	0.0%
total count	8	246	3	93	1
total size	$46.9K$	$3.7M$	$5.0K$	$1.8M$	$27.1K$

TODO: mention number of distinct locii

2.2.1 Ionotropic

A list of ionotropic receptors supplied by Corbin via Flybase & George et al 2019 (email 28 May 2019). This contained 335 entries, some with mutiple genes, some not unique. Once merged & uniqued : 246 Annotation symbols (CGxxxxx) gonverted to FlyBase gene games (FBgnxxxx) using flybase ID converter (http://flybase.org/convert/id)

239 converted cleanly; 5 had duplicate conversions and were corrected by hand:

```
CG11430 is FBgn0041585, not FBgn0050323
CG43368 is FBgn0263111, not FBgn0041188
CG8885 is FBgn0262467, not FBgn0081377
CG9090 is FBgn0034497, not FBgn0082745
CG9126 is FBgn0045073, not FBgn0053180
```

Two were corrected to be consistent with the dm6_genes annotation:

```
CG9907 (para), is listed as FBgn0264255 not FBgn0285944
CG42345 (straw) is listed as FBgn0259247 (laccase2)
```

2.2.2 Derived from GO terms

Nervous System Development:

```
nrd, FBgn0002967, no annotated gene model
l(2)23Ab, FBgn0014978, same
aloof, FBgn0020609, same
Imp, FBgn0285926, is FBgn0262735
```

Mating:

```
Only three, but all good
```

synapse signalling

```
1 gene
```

Histone modification, DNA trans factor act, synapse org

```
MT
```

2.3 Sequenced Reads

FAIRE-Seq reads were sequenced for four experimental treatments. Each had one "input" (in which the DNA was fragmented without crosslinking to chromatin) and three replicates in which the DNA was crosslinked and then fragmented.

rep	sample count
47b1-7	
1	1
2	1
3	1
input	1
Fru7	
1	1
2	1
3	1
input	1
GH7	
1	1
2	1
3	1
input	1
SH7	
1	1
2	1
3	1
input	1

To accomodate varying data quality, analyses were done on three nested subsets of the data, each including an input and at least one replicate for each experimental treatment. Groups A, B, and C correspond to 1, 2, and 3:

```
ok, do the alignment and MACS2 analysis like you did before.  Do three version:

1) Only unmarked
2) include the orange
3) all date
```

(Corbin, 20 May 2019 email)

```
data_sets.df %>% select(c(experimental, rep, subgroups)) %>% mutate(dummy="1") %>% filter(subgroups!='a
```

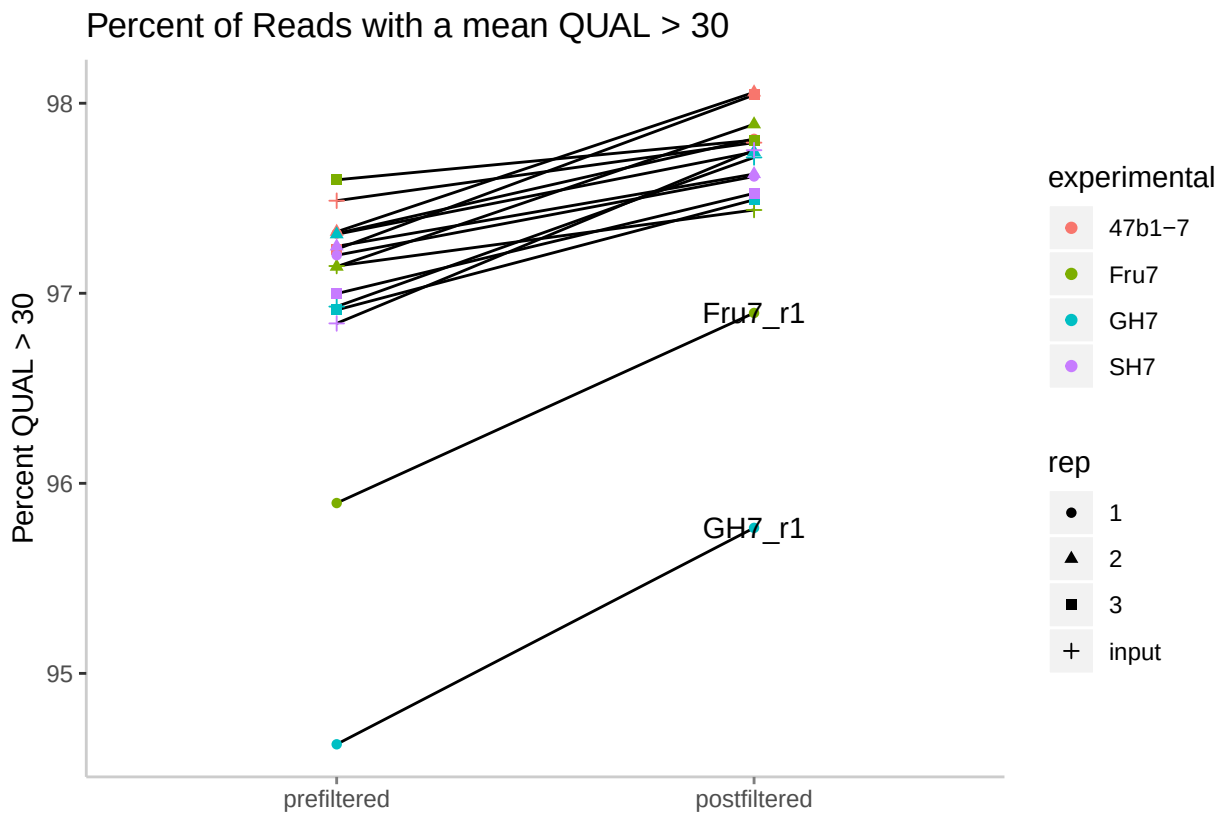
experimental	rep	A	B	C
47b1-7	1	o	o	o
47b1-7	2	o	o	o
47b1-7	3	x	x	o
47b1-7	input	o	o	o
Fru7	1	o	o	o
Fru7	2	x	o	o
Fru7	3	o	o	o
Fru7	input	o	o	o
GH7	1	x	x	o
GH7	2	o	o	o
GH7	3	o	o	o
GH7	input	o	o	o
SH7	1	x	o	o
SH7	2	o	o	o
SH7	3	o	o	o
SH7	input	o	o	o

2.3.1 Pre-Processing

These reads were preprocessed with FASTP (S. Chen et al. 2018) for quality control and analytics. Starting FASTQ files contained a total of $247M$ reads; after QC, this dropped to $247M$.

Read Retention Rate during Preprocessing

	minimum	average	maximum
prefiltered	$272K$	$15M$	$44M$
postfiltered	$271K$	$15M$	$44M$
percent retention	99	100	100

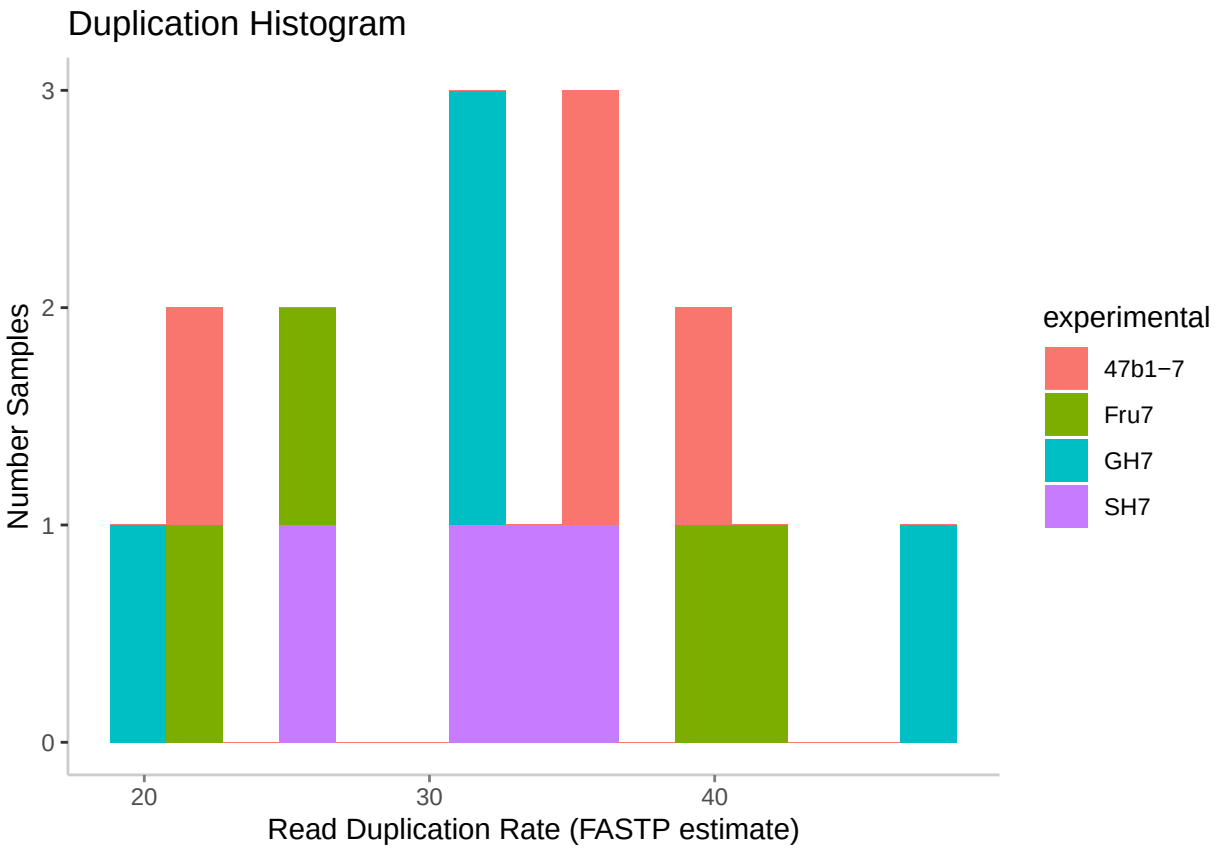


Duplicate reads were also detected; however, duplicate reads are less concerning in FAIRE-seq given the relatively smaller genome they are sampled from.

https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/8%20Duplicate%20Sequences.html http://seqanswers.com/forums/showthread.php?t=40440

Percentage Duplication
FASTP estimate

minimum	average	median	maximum
19.0	32.2	32.6	46.7

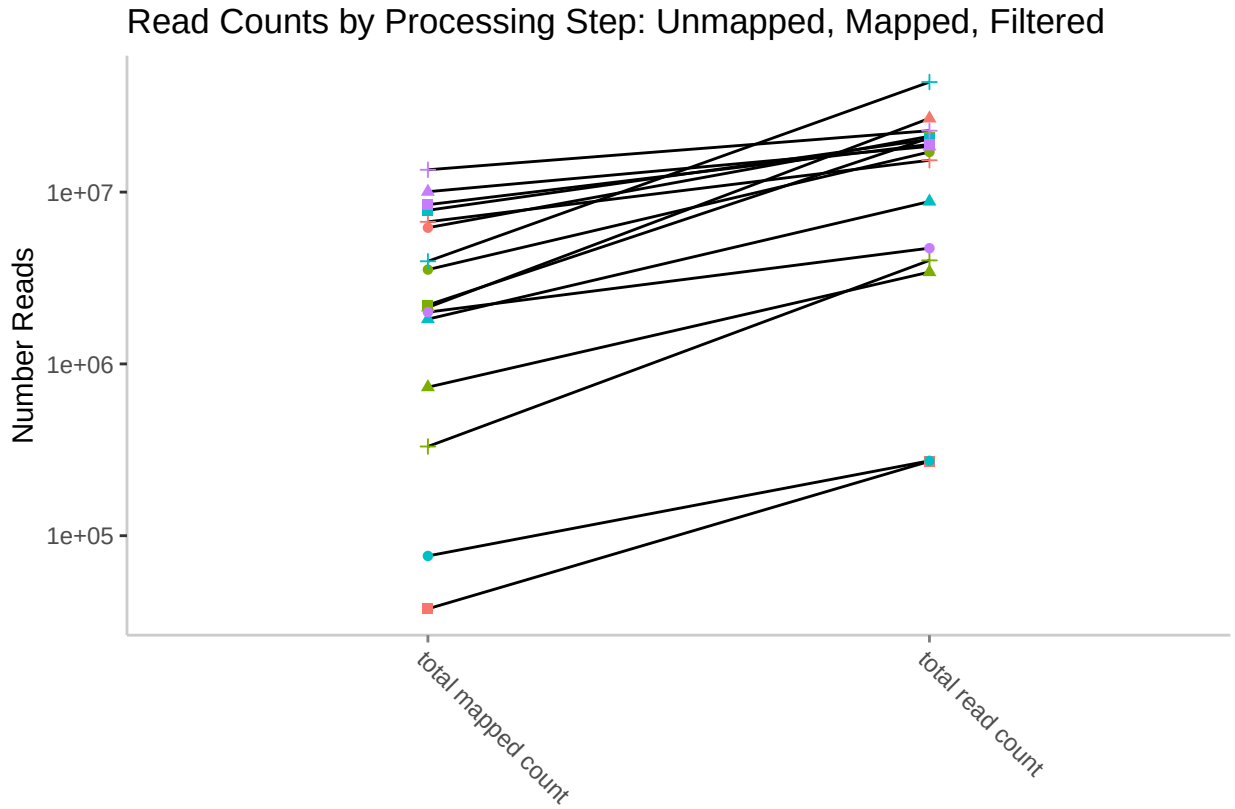


2.4 Mapped Reads

Reads were first mapped to the reference genome using the BWA SAMPE/SE algorithm. Currently, multimapping reads are assigned randomly and the alignments are used unfiltered.

2.4.1 Read & Alignment Quality

The mapping rate per sample can be calculated from the number of mapped reads compared to the total number of sequenced reads:



Read Counts During Alignment & Filtration

measure	minimum	average	median	maximum
percent mapping	7.94	28.31	24.66	59.21
total mapped count	37.5K	4.3M	2.9M	13.5M
total read count	271.1K	15.4M	17.8M	43.7M

2.5 Peak Calling

MACS was used to QC pilot data, but it wasn't designed for use on FAIRE-Seq. MACS2 was extended for use with FAIRE (https://epigeneticsandchromatin.biomedcentral.com/articles/10.1186/1756-8935-7-33) but has its own difficulties, such as python2/3 incompatibilities, and only running on all samples under inhomogeneous settings. Here, Fseq (Boyle et al. 2008) is used to infer peaks from mapped reads. It does this using a kernel density estimation to find intervals with significantly (4 sigma by default) more mapped reads than expected from the average regional coverage. Additionally, a signal strength is calculated, as the highest kernel density in the interval, scaled by interval size.

from the 16 samples, 542k total peaks were called:

Called Peak Count
by contig and sample

	by contig								
sample	chr2L	chr2R	chr3L	chr3R	chr4	chrM	chrX	chrY	total
47b1_7_in	6.0K	10.6K	10.7K	11.6K	587.0	2.0	2.8K	3.2K	45.6K

47b1_7_r1	6.4K	11.8K	10.9K	11.8K	420.0	1.0	3.4K	3.2K	47.9K
47b1_7_r2	4.1K	8.3K	7.8K	8.2K	411.0	4.0	3.9K	3.7K	36.4K
47b1_7_r3	2.8K	3.3K	3.5K	4.1K	154.0	1.0	2.0K	400.0	16.2K
Fru7_in	7.5K	9.7K	10.2K	11.9K	338.0	8.0	5.1K	2.5K	47.2K
Fru7_r1	3.0K	7.4K	6.6K	6.7K	359.0	5.0	3.2K	3.8K	31.0K
Fru7_r2	3.7K	7.0K	6.7K	6.9K	345.0	4.0	3.5K	3.2K	31.5K
Fru7_r3	4.6K	8.7K	8.3K	8.7K	347.0	5.0	3.5K	3.8K	38.0K
GH7_in	5.1K	9.2K	9.0K	9.7K	526.0	2.0	3.2K	4.0K	40.8K
GH7_r1	2.8K	3.5K	3.7K	4.0K	174.0	5.0	1.9K	947.0	17.0K
GH7_r2	3.4K	7.5K	6.9K	7.3K	346.0	3.0	3.4K	3.8K	32.7K
GH7_r3	4.7K	9.1K	8.0K	9.0K	296.0	2.0	3.2K	3.2K	37.5K
SH7_in	3.7K	7.3K	7.1K	7.3K	345.0	4.0	2.5K	3.4K	31.8K
SH7_r1	2.8K	6.8K	6.3K	6.4K	349.0	2.0	3.3K	3.8K	29.7K
SH7_r2	2.7K	6.2K	5.7K	6.0K	283.0	2.0	2.5K	3.2K	26.5K
SH7_r3	3.5K	7.7K	6.7K	7.4K	313.0	8.0	3.0K	3.4K	32.1K

We can also check the peak-calling efficiency:

Peak Calling Efficiency

peaks called per read sequenced/mapped

experimental	rep	total peak count	peaks per thousand sequenced reads	peaks per thousand mapped reads
47b1-7	input	45.6K	3.0	6.8
47b1-7	1	47.9K	2.3	7.7
47b1-7	2	36.4K	1.4	17.1
47b1-7	3	16.2K	59.8	432.1
Fru7	input	47.2K	11.8	142.7
Fru7	1	31.0K	1.8	8.8
Fru7	2	31.5K	9.2	43.0
Fru7	3	38.0K	1.8	17.2
GH7	input	40.8K	0.9	10.3
GH7	1	17.0K	62.4	223.5
GH7	2	32.7K	3.7	17.9
GH7	3	37.5K	1.9	4.8
SH7	input	31.8K	1.4	2.4
SH7	1	29.7K	6.3	14.8
SH7	2	26.5K	1.4	2.6
SH7	3	32.1K	1.7	3.8

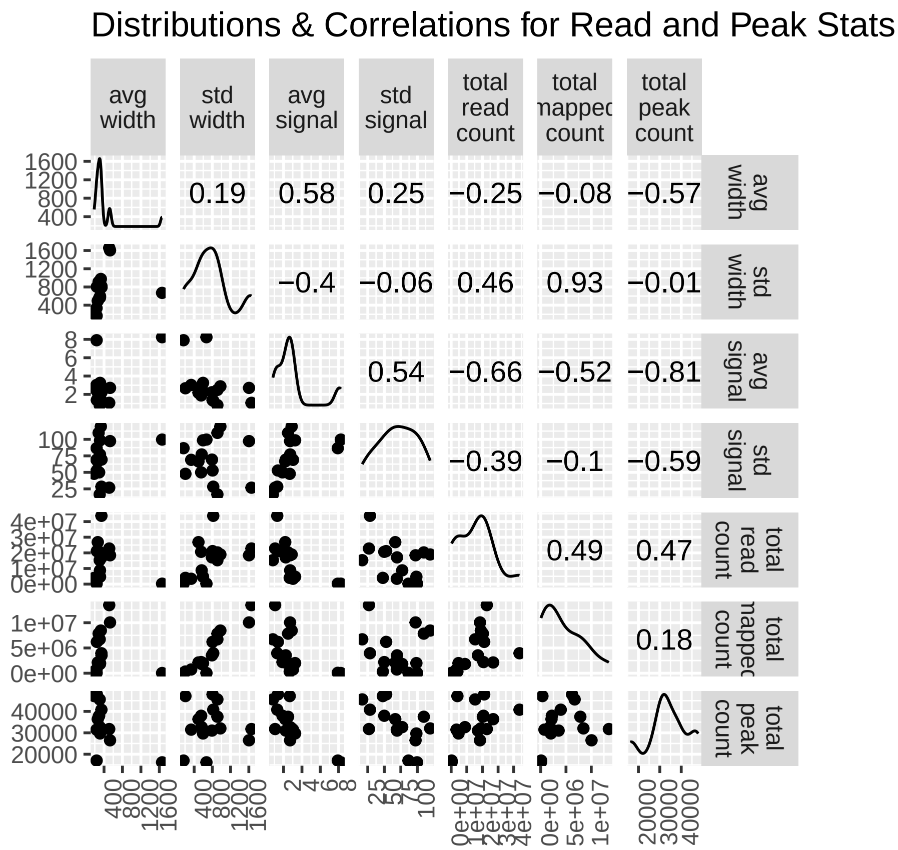
2.5.1 Raw Peaks

peaks had an average width of 401 base pairs. Peaks also had intensity values measuring signal enhancement over the genomic background; these averaged at 2.92 Both of these values were highly variable within and between samples:

Peak Size & Signal Strength

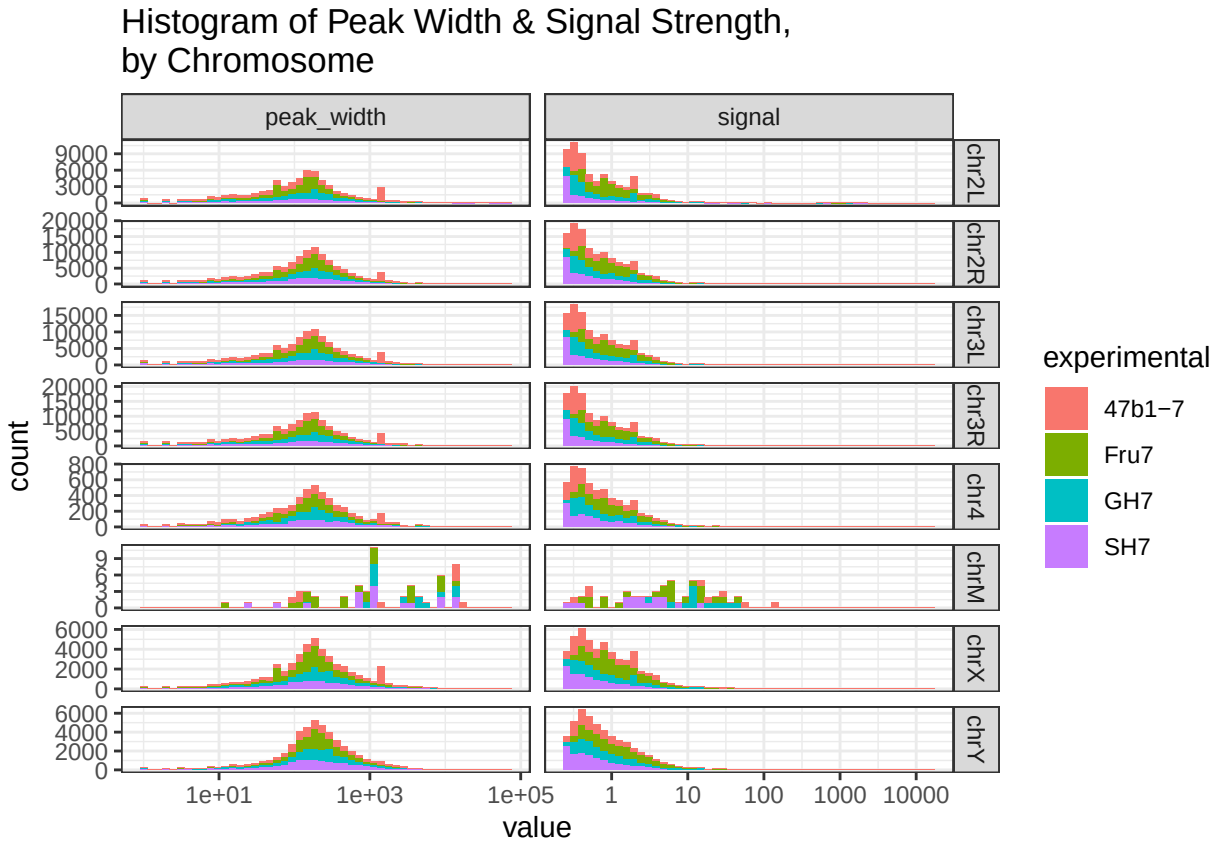
		width		signal	
experimental	rep	avg	std	avg	std
47b1-7	input	306.0	910.4	0.8	16.6
47b1-7	1	245.0	804.4	1.4	52.8
47b1-7	2	265.2	495.4	2.2	66.6

47b1-7	3	1,659.8	669.9	8.2	99.6
Fru7	input	186.4	208.5	2.7	47.8
Fru7	1	346.2	791.6	2.2	69.3
Fru7	2	238.0	335.5	3.0	69.0
Fru7	3	294.8	554.0	1.9	50.0
GH7	input	343.9	819.2	1.3	28.3
GH7	1	239.5	167.0	7.9	86.6
GH7	2	312.5	565.2	2.7	77.1
GH7	3	283.2	911.4	2.5	109.8
SH7	input	512.9	1,658.0	1.1	26.8
SH7	1	314.4	595.3	3.3	98.6
SH7	2	530.9	1,604.6	2.7	97.3
SH7	3	331.0	975.9	2.9	119.4



!! of the $542k$ peaks, $5.76k$ had a width of zero !!

```
There shouldn't be any peaks that consist of zero bases - if you're looking at the narrow peak file, the
(emails from Terry Furey, 6 Aug 2019)
```



The consistent spike in peak width in the 47b1-7 experiment, slightly above 1kb, comes from replicate 3. This sample has an order of magnitude more peaks large than 1kb, than the other 15.

Number of Peaks Larger than 1 kb
by experiment and replicate

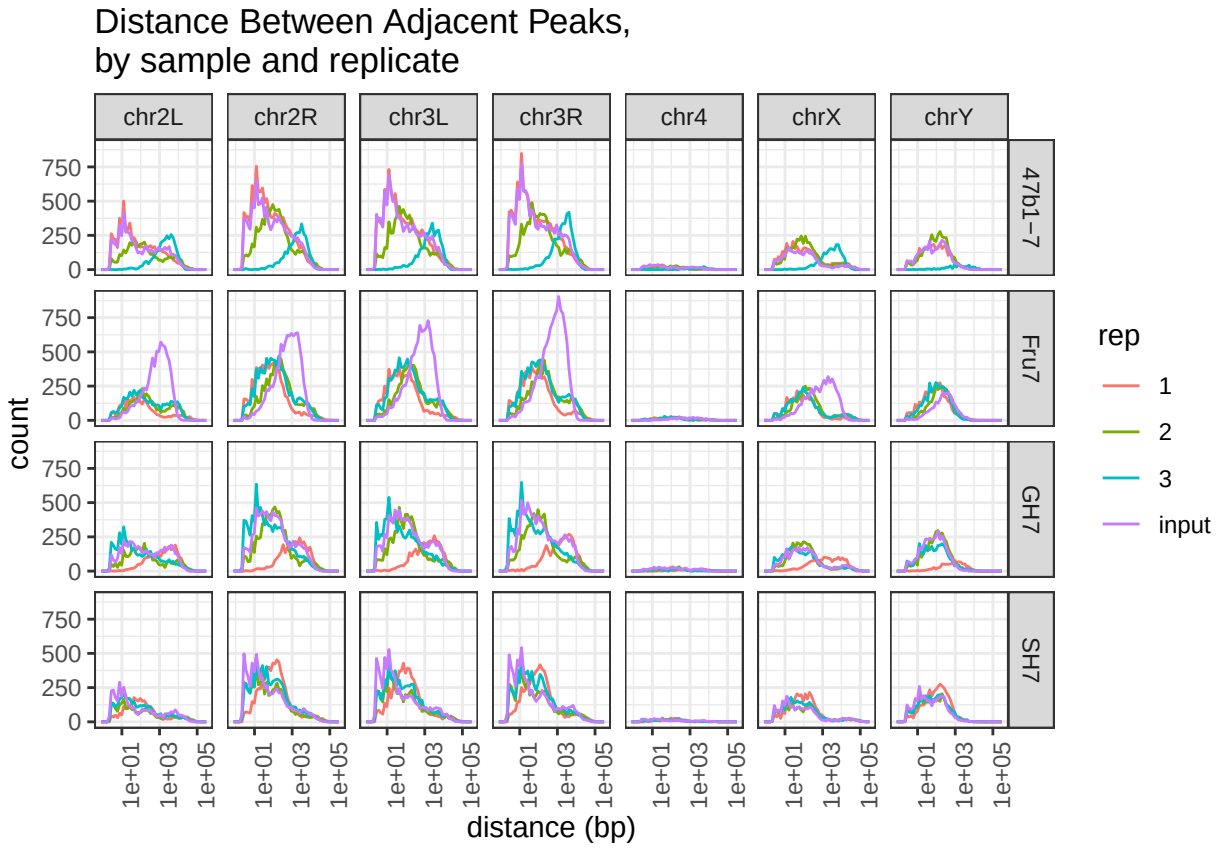
experimental	1	2	3	input
47b1-7	2016	1456	16211*	2641
Fru7	2082	566	1955	353
GH7	80	1657	1846	2824
SH7	1541	2969	2022	3435

*

47b1-7 replicate #3 is only included in the "C" analysis group (Section 2.6).

check intrasample, inter-peak distances: look for potentially non-distinct peaks.

The distance between adjacent peaks within a sample was measured.



Some peaks were very close to other peaks; as many as 20% of peaks were within 10bp of another one, and as many as 65% were within 100 bp:

Percentage of Peaks Very Close to Other Peaks
within a sample

experimental treatment	rep	< 10 bp	< 100 bp
47b1-7			
	input	19.5%	58.1%
	1	19.8%	60.5%
	2	8.1%	48.6%
	3	0.3%	3.4%
Fru7			
	input	1.5%	15.9%
	1	12.2%	57.8%
	2	4.1%	34.4%
	3	8.1%	47.7%
GH7			
	input	11.4%	51.6%
	1	0.6%	7.2%
	2	7.6%	48.7%
	3	20.2%	60.5%
SH7			
	input	28.5%	65.9%

1	7.6%	50.3%
2	21.6%	61.0%
3	18.7%	60.4%

2.5.2 Collapsed Peaks (within input/output)

Although the inputs consisted of a single replicate, each experimental output had three replicates. These were collapsed into a single set of peaks each for every input and output. The collapsed peak region was defined as the union of all peaks being considered. Calculating a combined signal value for multiple peaks is an open problem; several approaches are tested here:

```
* Average: a flat average of the component peaks' signal values
* Rescale: in which the component peaks' maximum kernel densities are averaged, and then scaled by r
* Weighted: like Rescale, but the average max kernel density is weighted by the component peak widt
* Pessimistic: like weighted, but the result is further scaled by the fraction of replicates suppor
* Fine: (in development)
```

All strategies but Average will give NA for singleton zero-width peaks.

```
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
```

```
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
```

```
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
```

```
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
```

```
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
```

```
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
```

```
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
```

```
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
```

```
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
```

```
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
## Parsed with column specification:
## cols(
```

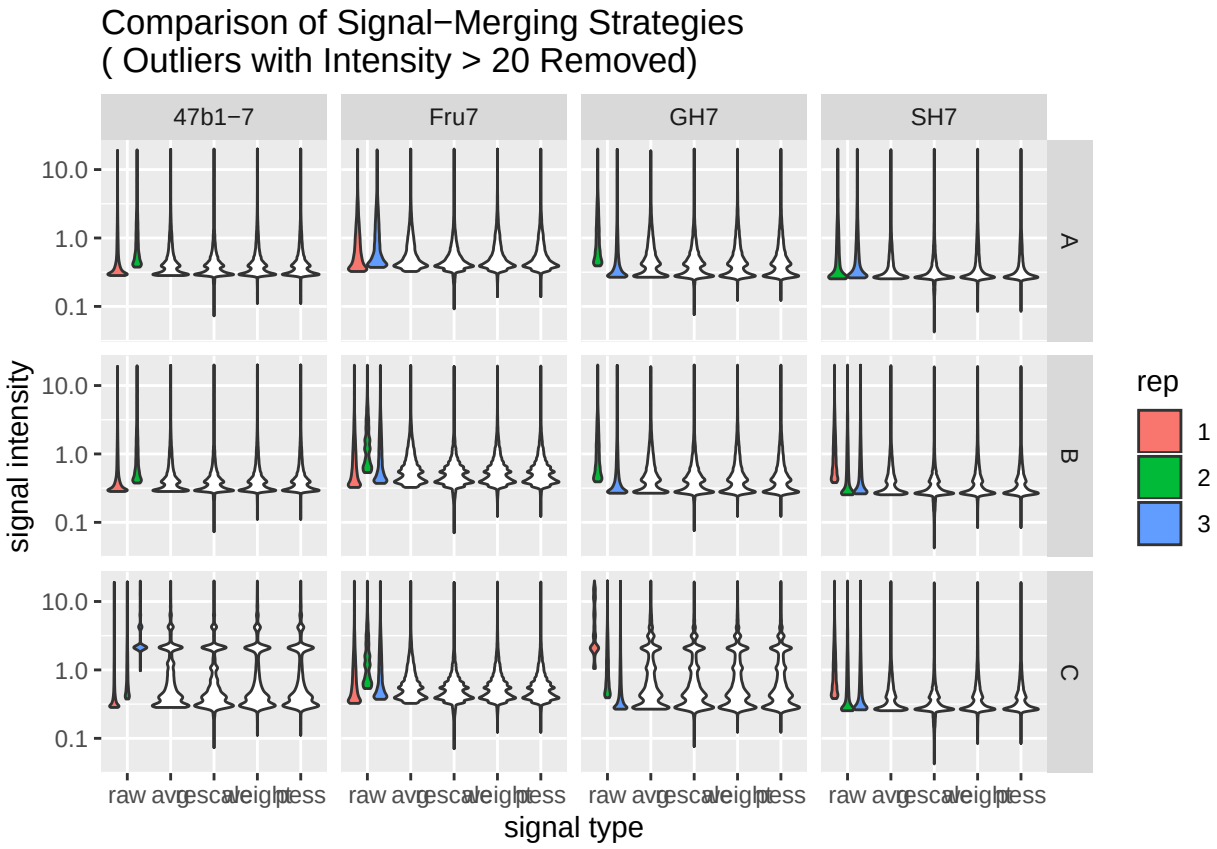
```
##   X1 = col_character(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double()
## )
```

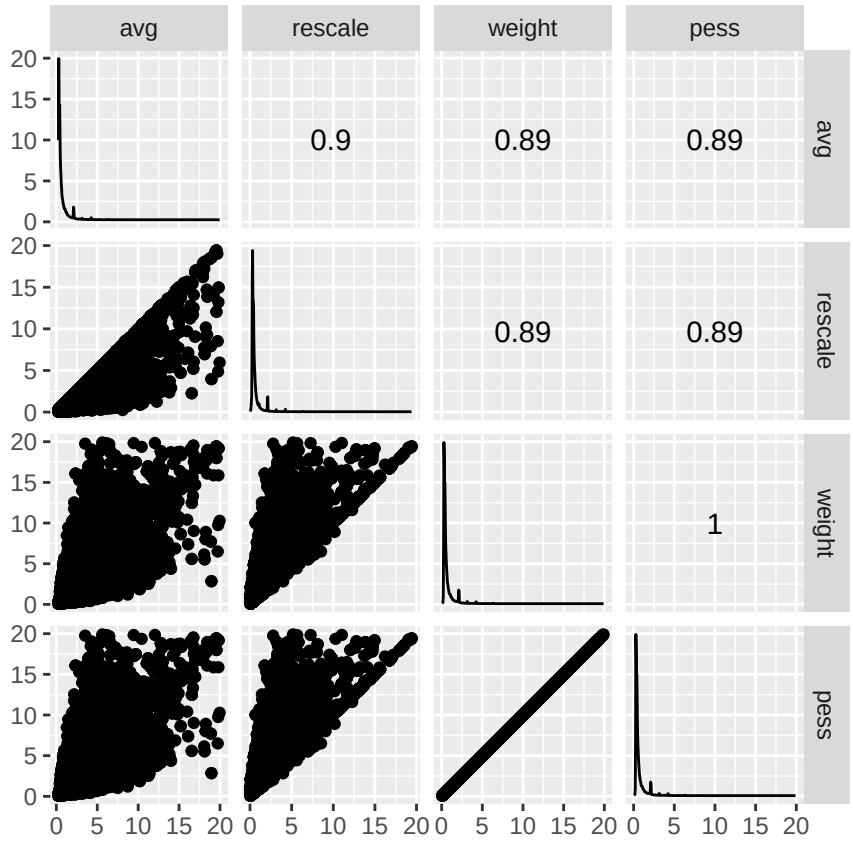
Since the inputs consist of one replicate each, their collapsed peaks are identical to their raw peaks (except for those with a width of zero, whose collapsed signal strengths are NA except for the flat average). In the case of the outputs, with 1-3 replicates depending on experimental treatment and

When the small number (809 m percent of raw peaks, 158 m percent of collapsed peaks) of high-signal outliers (raw or collapsed signal > 20) were excluded, the distributions of the raw/collapsed signals were as follows

```
## Warning: Transformation introduced infinite values in continuous y-axis
```

```
## Warning: Removed 36 rows containing non-finite values (stat_ydensity).
```





ggscattercorr + naniar::geom_miss_point ?

2.5.3 Merged Peaks (proximity within input/output)

Once the peaks have been collapsed within a treatment, the issue of nearby peaks not being distinct remains. It is trivial to merge nearby peaks into a single feature, but the issue of how to combine signal strengths remains.

3 Bibliography

```
##
## To cite package 'topGO' in publications use:
##
##   Adrian Alexa and Jorg Rahnenfuhrer (2018). topGO: Enrichment
##   Analysis for Gene Ontology. R package version 2.34.0.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {topGO: Enrichment Analysis for Gene Ontology},
##     author = {Adrian Alexa and Jorg Rahnenfuhrer},
##     year = {2018},
##     note = {R package version 2.34.0},
##   }
```

```
##
## ATTENTION: This citation information has been auto-generated from
## the package DESCRIPTION file and may need manual editing, see
## 'help("citation")'.


##
## To cite ggplot2 in publications, please use:
##
##   H. Wickham. ggplot2: Elegant Graphics for Data Analysis.
##   Springer-Verlag New York, 2016.
##
## A BibTeX entry for LaTeX users is
##
##   @Book{,
##     author = {Hadley Wickham},
##     title = {ggplot2: Elegant Graphics for Data Analysis},
##     publisher = {Springer-Verlag New York},
##     year = {2016},
##     isbn = {978-3-319-24277-4},
##     url = {https://ggplot2.tidyverse.org},
##   }


##
## To cite package 'GGally' in publications use:
##
##   Barret Schloerke, Jason Crowley, Di Cook, Francois Briatte,
##   Moritz Marbach, Edwin Thoen, Amos Elberg and Joseph Larmarange
##   (2018). GGally: Extension to 'ggplot2'. R package version 1.4.0.
##   https://CRAN.R-project.org/package=GGally
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {GGally: Extension to 'ggplot2'},
##     author = {Barret Schloerke and Jason Crowley and Di Cook and Francois Briatte and Moritz Marbach
##     year = {2018},
##     note = {R package version 1.4.0},
##     url = {https://CRAN.R-project.org/package=GGally},
##   }


##
## To cite package 'ggnewscale' in publications use:
##
##   Elio Campitelli (2019). ggnewscale: Multiple Fill and Color
##   Scales in 'ggplot2'. R package version 0.3.0.
##   https://CRAN.R-project.org/package=ggnewscale
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {ggnewscale: Multiple Fill and Color Scales in 'ggplot2'},
##     author = {Elio Campitelli},
##     year = {2019},
```

```
##     note = {R package version 0.3.0},
##     url = {https://CRAN.R-project.org/package=ggnewscale},
##   }
```

Boyle, Alan P, Justin Guinney, Gregory E Crawford, and Terrence S Furey. 2008. "F-Seq : a feature density estimator for high-throughput sequence tags" 24 (21): 2537–8. doi:10.1093/bioinformatics/btn480.

Chen, Shifu, Yanqing Zhou, Yaru Chen, and Jia Gu. 2018. "Fastp: An ultra-fast all-in-one FASTQ preprocessor." *Bioinformatics* 34 (17): i884–i890. doi:10.1093/bioinformatics/bty560.