Pre-infusion versus Post Infusion Integration Site Clumps in WAS Patients

November 25, 2014

Contents

1 Introduction

This report investigates the clustering or *clumping* of integration site distributions in cell samples from gene-corrected WAS patients. The study in-

cluded only six patients, so definitive conclusions about safety await future studies with greater power.

Adverse events have taken place in gene therapy trials with early stage gammaretroviral vectors that involved insertional activation of cancer-related genes. Leukemias developed after reinfusion of gene-corrected cells, and analysis of the transformed cells showed that vector DNA had integrated near the transcription start sites of gene such as LMO2, CCND2, and MECOM/EVI1. These observations and others support the idea that insertional activation of these genes contributed *hits* toward transformation.

In addition, analysis of populations of integration sites in samples from these trials showed that many integration sites could be found near the transcription start sites of LMO2, CCND2, MECOM/EVI1 and other genes of concern even in non-transformed cells. These findings supported the idea that after transplantation, cells with integration sites in these loci proliferated or survived preferentially. Supporting this idea, repeating one of these trials with a vector with reduced potential for insertional activation resulted in no detection of such clumps ([Hacein-Bey-Abina et al., 2014]). A previous trial to treat WAS using an early-version gammaretroviral vector with high potential for insertional activation resulted in detection of integration site clumps associated with LMO2, CCND2, MECOM/EVI1, and other genes of concern ([Braun et al., 2014]). Integration sites from many of the patients were found in each of these clumps.

Thus, as a surrogate for safety in the WAS lentivirus trial reported here, we investigated clumping of integration sites in patient samples using scan statistics ([Berry et al., 2014]). In one approach, we compared distributions of integration sites in each subject pre-transplantation and post-transplantation. We sought to determine whether preferential outgrowth was detectable associated with integration near LMO2, CCND2, MECOM/EVI1, or other genes of concern.

Comparing over clumps seen in all patients, clumps were found near cancer-related genes no more frequently than expected by chance. A handful of patient-specific clumps could be detected, and some of these were near cancer-related genes (broadly defined), but none were near LMO2, CCND2, MECOM/EVI1 or other genes implicated in adverse events in human gene therapy.

In another approach, we compared integration site clumps from the WAS-lentiviral vector trial reported here to integration site clumps from the Braun et al. trial using a WAS gammaretroviral vector, in which more than half of the subjects experienced adverse events. A total of 280 clumps were detected at the cut value studied, of which 230 were specific to the WAS-

gammaretroviral vector trial. Among these were clumps at LMO2, CCND2, and MECOM/EVI1 — genes that were involved in adverse events in the WAS-gammaretroviral vector study. (That a large number of clumps was discovered is unsurprising given that different methods of recovering integration sites were used and this can generate differences in recovery rates that generate clumps as was shown in [Berry et al., 2014], for example.) Thus we infer that the pattern of integration site clumping seen in the WAS-lentiviral vector trial is consistent with improved safety.

A technical account of the statistical analysis is presented below.

2 The Data

This section covers details of data processing used specifically for this report. These data were provided by Nirav Malani of the Bushman Lab at UPenn to Charles Berry who performed the processing for this document. This report uses the data to look for differences between the pre-infusion and post-infusion distributions of integration sites.

The `R` code used appears in the appendix. We start by loading the `GRanges` object prepared by Nirav Malani and adding `hg18` metadata. As a prelim, the `GenomicRanges` [Lawrence et al., 2013] and `geneRxCluster` [Berry et al., 2014] `R` packages [R Core Team, 2013] are loaded. Then the actual `GRanges` object that gives the integration sites is loaded. Also, information about the chromosomes in the `hg18` freeze is added.

```
## [1] "data/was.gr.RData"
## [1] "was.gr"
```

The integration sites that are unique to the patient and infusion are identified, the `start` of sites on the `+` strand are moved to match the site of attack with the lower position number, and then all sites are ordered by position on each chromosome.

The numbers of pre and post infusion sites per patient are reported here — starting by checking the potential mismappings by comparing the `start` of adjacent sites according to strand.

And now here is the table of the successive differences. An excess of `++` and `--` compared to `-+` and `+-` suggests mismapping for small differences in the successive postions (`start`). As seen here a difference of '1' shows such an effect, therefore these will be deleted before proceeding.

```
##
##         ++    +-    -+    --
##   0      0    21    16     0
##   1    266     2    21    44
##   2      2     0     4     0
##   3      1     2     2     2
##   4      0     3     1     2
##   5      1     0     3     0
##   6     29     1     1    36
##   7      9     0     1    13
##   8      8     0     1     5
##   9      7     3     1     4
##   10 20263 20124 20097 19854
```

3 Dereplication

Dereplication rids extraneous positions whose mismapping may result in counting a single integration site as multiple sites. The code is a bit tortuous. In the end it finds a logical vector of sites to be dropped as misplaced duplicates. It uses the clonecount to make this decision as to which of several competing positions to use with a random tiebreaker if necessary.

And here is the table of patients by pre/post-infusion.

```
##
##        post   pre
##   Pt1  1633  5250
##   Pt2  7341 17356
##   Pt3  1452  1026
##   Pt4 11854  2481
##   Pt5  2256   176
##   Pt6   822 29221
```

This table guides some choices in the analysis. Since there are several patients with many fewer sites in either the pre or post phase, filtering the marginal counts for single patients could be troublesome. Also, the imbalances could contribute to artifactual findings in the form of Simpson's paradox if data from all patients are combined. However, single patient analyses have many fewer sites and might yield low power comapred to when all

sites are used. As a consequence, both single patient analysis and combined patient analysis will be performed.

4 Patient Specific Clustering of Sites

By tabling the successive patient identities of adjacent sites, one gets a quick indication of whether there is patient specific clustering of pre-infusion versus post-infusion integration sites. Here is the table (rows refer to the site at the lesser position, columns to the greater).

```
##
##        Pt1  Pt2 Pt3  Pt4 Pt5   Pt6
##   Pt1  664 2040 237 1170 201  2567
##   Pt2 2031 7891 768 4325 734  8936
##   Pt3  232  761  85  462  68   868
##   Pt4 1247 4305 441 2968 510  4858
##   Pt5  223  776  74  491 104   764
##   Pt6 2484 8914 869 4911 813 12038
```

Here are the fitted values under the assumption of no patient clustering:

```
##
##          Pt1    Pt2   Pt3    Pt4   Pt5     Pt6
##   Pt1  585.6 2101.0 210.5 1219.3 206.8  2555.8
##   Pt2 2101.4 7539.3 755.5 4375.4 742.1  9171.3
##   Pt3  210.8  756.2  75.8  438.9  74.4   919.9
##   Pt4 1219.8 4376.3 438.6 2539.8 430.8  5323.7
##   Pt5  207.0  742.8  74.4  431.1  73.1   903.6
##   Pt6 2556.3 9171.4 919.1 5322.6 902.8 11156.8
```

And here are the differences of observed and expected values:

```
##
##         Pt1    Pt2   Pt3    Pt4   Pt5    Pt6
##   Pt1  78.4  -61.0  26.5  -49.3  -5.8   11.2
##   Pt2 -70.4  351.7  12.5  -50.4  -8.1 -235.3
##   Pt3  21.2    4.8   9.2   23.1  -6.4  -51.9
##   Pt4  27.2  -71.3   2.4  428.2  79.2 -465.7
##   Pt5  16.0   33.2  -0.4   59.9  30.9 -139.6
##   Pt6 -72.3 -257.4 -50.1 -411.6 -89.8  881.2
```

Notice that the diagonal is positive — more observed than expected in the same patient — and the off diagonal elements are mostly negative. And for completeness the X^2 test:

```
##
##  Pearson's Chi-squared test
##
## data:  pp.tab
## X-squared = 348.7741, df = 25, p-value < 2.2e-16
```

Given this, it seems best to account for patient effects in any assessment of clumping of sites in which the sites from multiple patients are studied. A permutation tests in which the pre-post identities of sites for each patient are permuted en bloc.

5 Clumps in Individual Patients

For each patient, the `gRxCluster` function is applied with no filtering for low density of sites. The default criterion for the target number of false discoveries is adjusted to 5/6 (versus the default of 5) so that when the defaults are added over all six patients, they sum to 5.

Here is the summary of the discoveries for each patient. As can be seen all the clumps discovered for 5 of the six patients are at an FDR of 0.10 or less.

```
##     Clusters_Discovered  FDR
## Pt1                   9 0.06
## Pt2                  52 0.02
## Pt3                   6 0.11
## Pt4                   8 0.05
## Pt5                   0 0.34
## Pt6                   4 0.05
```

Clumps could be idiosyncratically associated with just one patient or seen in multiple patients. Here the overlapping clumps are joined together.

The next table shows how many patients have clumps that overlap those of other patients. The '1' means only one clump shares the region (with itself), '2' means two clumps share a region, et cetera.

```
##
##  1  2  3  5
## 39  8  4  1
```

There are quite a few clumps that seem to be idiosyncratic, none shared by all patients, and only one clump shared by 5 of the 6 patients. However, maybe a clump discovered in one patient exists in another, but the signal is not strong enough to warrent 'discovery'. The following code sets the stage for exploring correlation of clump signals between patients.

First, the count of clumps favoring pre-infusion sites is compared to the count favoring post-infusion sites.

```
##    POST
## PRE  0  1  2  3  5
##   0  0 22  6  4  1
##   1 17  0  0  0  0
##   2  2  0  0  0  0
```

It looks like when clumps are discovered in different patients at the same genomic location they tend to agree as to whether pre or post infusion sites are favored. Here are the clumps that or discovered in two or more patients.

Here a code of `PRE` or `POST` identifies discovered clumps while `pre` or `post` indicates which time is favored for the sites in a region discovered in other patients but not the patient listed in the column heading. As can be seen, even when no clump is discovered in a patient, the direction favored usually agrees with the direction in patients in which clumps were discovered.

```
##     Pt1  Pt2  Pt3  Pt4  Pt5  Pt6
## 4  POST POST post post post post
## 7  POST POST POST POST post POST
## 12 POST POST  pre POST post post
## 20 POST POST POST post  pre post
## 21 post POST POST post post post
## 24 POST POST POST post post post
## 25 post POST post post post POST
## 26 post POST post post  pre POST
## 34 post post POST POST post post
## 39  pre  PRE post  PRE  pre  pre
## 40  pre  PRE  pre  PRE  pre  pre
## 42 POST POST  pre POST post post
```

```
## 51 post POST POST post post post
```

```
##    chromo     start       end    width
## 4    chr1 152697830 155022997  2325168
## 7   chr11  64721170  71336379  6615210
## 12  chr12  48101903  54594114  6492212
## 20  chr17  32393122  45086180 12693059
## 21  chr17  70859851  74367028  3507178
## 24  chr19   7681725  18287070 10605346
## 25  chr19  43133759  51508964  8375206
## 26  chr19  53620448  54764090  1143643
## 34   chr3  49090753  52690505  3599753
## 39   chr4  99533135 110213671 10680537
## 40   chr5  89268705  98267507  8998803
## 42   chr6  32269912  36508667  4238756
## 51   chr9 127473502 131821313  4347812
```

6 All Patients

In this section, sites for all patients are combined and subjected to discovery by `gRxCluster`. A set of permutations is developed. The permutations are used to find the false discovery rate.

Here are the sites discovered at an FDR of less than 0.15. The column labelled 'value1' counts the number of pre-infusion sites, while that labelled 'value2' counts the post-infusion sites. The 'FDR' column is the false discovery rate estimated by permuting the pre-post condition labels in each patient as a block.

	chromo	start	end	width	value1	value2	FDR
17	chr1	153908358	154451388	543031	43	80	0.12
26	chr11	64806544	65110410	303867	41	108	0.07
56	chr14	105563367	106348468	785102	46	88	0.12
81	chr17	37409347	38585509	1176163	86	142	0.12
90	chr17	71474109	75578416	4104308	233	332	0.12
97	chr19	2245314	3147284	901971	51	83	0.12
100	chr19	10280360	11286268	1005909	70	113	0.12
104	chr19	14111371	15238855	1127485	42	88	0.12
106	chr19	17292023	18242345	950323	26	95	0.09
113	chr19	53582121	54767807	1185687	82	164	0.03

Table 1: Clumps Discovered Using All Patients.

Here is the number of such sites that overlap with the clumps discovered in two or more individual patients or not.

```
##
##     Over Not Over
##        8        2
```

Here is the number of regions with clumps discovered in 2 or more patients that overlie (or not) a clump discovered when all sites are used.

```
##
##     Over Not Over
##        6        7
```

7 Gene Annotations

Now look at how the clumps discovered relate to gene annotations. First, some gene annotations are read in for the bodies and edges of Lymphoma associated genes (`LymBody`, `LymEdge`) and for the bodies and edges of a list of Cancer associated genes (`AllBody`, `AllEdge`).

7.1 Individual Patients

A table is prepared showing the number of clumps that overlie one or more genes according to each of four annotations. Here the clumps discovered in each patient that also occur in at least one other patient are studied. The table below shows the number of clumps that overlie one or more genes.

```
##     LymBody LymEdge AllBody AllEdge
## Pt1       2       2       6       6
## Pt2       1       2      18      18
## Pt3       2       2       6       6
## Pt4       0       0       6       6
## Pt5       0       0       0       0
## Pt6       0       0       4       4
```

So, most patients have clumps that are shared with others and that overlie one or more genes. To see if this is remarkable, 50 permutations of the pre-post infusion indicator are performed. For each patient, the clumps are discovered in the same manner as before, and the clumps it shares with

others are screened to see if any genes lie within the region of the clump. A χ^2 goodness-of-fit tests is performed, taking the mean number of clumps overlying genes as the expected value.

First the permutations are performed and the jointly discovered clumps are noted. Now the results are rendered as tables like that above for each permutation. Now the tables are used to find expected values and χ^2 statistics. The p-values are subjected to Holm adjustment for multiple comparisons.

```
##      Var2
## Var1  LymBody LymEdge AllBody AllEdge
##   Pt1 0.64    0.64    <0.01   <0.01
##   Pt2 1       0.64    <0.01   <0.01
##   Pt3 0.64    0.64    <0.01   <0.01
##   Pt4 1       1       <0.01   <0.01
##   Pt5 1       1       1       1
##   Pt6 1       1       <0.01   <0.01
```

As can be seen, most patients have more clumps overlying `All` cancer genes or bodies than expected. To determine the degree to which these clumps favor the pre or post infusion sample and whether the number of such sites is enough to raise concern, some further tables were created. For each patient the clumps seen in at least one other patient were studied. The logarithm of the odds ratio (`log.OR`) for the number of sites in the clump (versus elsewhere) comparing the post to the pre infusion samples was computed. Also the expected numbers of sites given the width of the clump and the overall numbers of sites pre and post infusion were computed and the ratios of observed to expected (`pre.OE`, `post.OE`) computed. It should be noted that it is expected that the observed/expected ratios will typically be greater than 1.0, since retroviral vectors have fairly strong preferences for sites of integration, i.e most sites are in regions of higher than average density. Here are the tables:

chromo	start	end	log.OR	pre.OE	post.OE	LymBody	LymEdge	AllBody	AllEdge
chr1	152697830	154439441	1.56	3.39	16.36	0	0	5	10
chr11	64721170	66729654	1.12	10.30	31.21	0	0	11	22
chr12	48101903	54594114	1.44	1.82	7.61	0	0	12	25
chr17	32393122	45086180	1.29	2.37	8.38	0	0	37	72
chr19	7681725	18287070	1.26	3.57	12.18	2	4	31	62
chr6	32270883	36508667	1.29	3.35	12.10	1	2	11	23

Table 2: Pt1

chromo	start	end	log.OR	pre.OE	post.OE	LymBody	LymEdge	AllBody	AllEdge
chr1	153908358	155022997	1.44	3.21	13.65	0	0	4	8
chr11	64967955	65153744	2.27	4.81	50.04	0	0	3	9
chr11	66667440	67615444	1.57	5.09	24.52	0	0	4	7
chr12	48381634	49336955	1.79	1.87	11.50	0	0	1	1
chr12	52279780	54443377	1.32	1.65	6.25	0	0	4	7
chr17	32571350	39895318	1.27	2.12	7.50	0	0	25	50
chr17	43486382	45037513	1.77	1.27	7.63	0	0	6	11
chr17	71306851	73581972	1.26	5.18	18.20	0	0	4	9
chr19	10697020	12448582	1.73	1.94	11.10	0	0	3	6
chr19	13079099	15086486	1.32	2.76	10.31	0	1	3	8
chr19	17114448	18073086	1.81	2.05	12.78	0	0	4	8
chr19	43133759	51508964	1.26	1.56	5.45	0	0	23	46
chr19	53620448	54764090	1.64	4.06	21.06	0	0	3	5
chr4	99533135	108052803	-2.19	1.53	0.15	1	1	5	9
chr5	89268705	94895250	-3.56	1.30	0.00	0	0	1	2
chr6	32269912	32885356	1.47	4.36	19.23	0	0	1	6
chr6	33497458	34724059	1.61	1.75	8.96	0	0	2	4
chr9	127655338	130377171	1.43	1.77	7.45	0	0	2	2

Table 3: Pt2

chromo	start	end	log.OR	pre.OE	post.OE	LymBody	LymEdge	AllBody	AllEdge
chr11	66286871	71336379	2.23	0.60	8.04	1	2	11	22
chr17	35218090	41595694	1.95	0.95	8.04	0	0	22	43
chr17	70859851	74367028	1.53	2.59	13.40	0	0	9	16
chr19	11096147	15314600	2.12	0.72	8.61	1	2	12	24
chr3	49324552	52690505	1.72	1.80	12.06	0	0	21	40
chr9	127473502	131821313	3.28	0.00	8.84	0	0	6	11

Table 4: Pt3

chromo	start	end	log.OR	pre.OE	post.OE	LymBody	LymEdge	AllBody	AllEdge
chr11	64900799	65197021	3.08	0.00	45.05	0	0	6	13
chr12	49767843	53709174	3.08	0.00	3.39	0	0	9	18
chr3	49090753	49982312	3.17	0.00	16.44	0	0	8	16
chr4	106050211	110213671	-2.31	3.30	0.31	0	0	2	4
chr5	93302482	98267507	-1.99	2.77	0.37	0	0	1	2
chr6	34411534	36467219	3.33	0.00	8.40	0	0	4	8

Table 5: Pt4

chromo	start	end	log.OR	pre.OE	post.OE	LymBody	LymEdge	AllBody	AllEdge
chr11	64963436	65194518	3.04	6.43	130.65	0	0	5	12
chr11	65678181	66166500	2.67	3.91	54.10	0	0	2	4
chr19	46741596	47245026	2.65	4.43	59.97	0	0	2	5
chr19	53970263	54759514	2.35	5.25	52.60	0	0	2	4

Table 6: Pt6

7.2 All Patients

Now the results for clumps derived from all patients combined are considered. Here the clumps that achieved an FDR$<$0.15 are examined and the number of genes that they overlie is noted. Again counts of genes overlapped by each clump, the log odds ratio, and the ratios of observed to expected sites are presented.

	LymBody	LymEdge	AllBody	AllEdge	log.OR	pre.OE	post.OE
17	0	0	1	2	1.40	4.43	18.02
26	0	0	4	8	1.75	7.54	43.48
56	0	0	0	0	1.43	3.27	13.71
81	0	0	6	11	1.29	4.09	14.77
90	0	0	10	20	1.15	3.17	9.90
97	0	0	3	6	1.27	3.16	11.26
100	0	0	4	10	1.26	3.89	13.74
104	0	0	4	7	1.52	2.08	9.55
106	0	1	4	9	2.07	1.53	12.23
113	0	0	3	6	1.48	3.86	16.92

Table 7: Gene Overlaps for Clumps Discovered Using All Patients

To get an idea if these should be given weight, the p-values determined by permutation are given here. There is one p-value for each gene annotation class. These are subjected to the Holm adjustment for multiple comparisons.

```
## LymBody LymEdge AllBody AllEdge
##   1.000   0.250   0.125   0.125
```

8 Comparison to WAS γ-Retroviral Trial

The authors of a recent paper on gene therapy for Wiskott-Aldrich Syndrome using a γ-retroviral vector ([Braun et al., 2014]) have graciously shared their data, thus allowing a direct comparison of the distribution of integration sites from these two trials. The Braun et al. data were lifted over to the `hg18` freeze to enforce comparability with our data.

Before comparing our post-infusion data to their data, we performed curation analogous to that performed on the present data to eliminate duplicate copies of the same integration site in a single patient. This is necessary, because the same site can appear in different samples from the same patient, and because variations in the reported sequence for different reads on he same integration site can result in the position being mapped in multiple adjacent or nearby positions. The following table demonstrates the situation before such curation. The table examines the distances between adjacent, unique positions of integration sites (allowing duplicates if they arise from differnt patients). It also determines the orientation of the integration site — reporting '+ +' if both sites in an adjacent pair lie on the '+' strand, '+ -' if the first lies on the '+' strand and the second lies on the '-' strand, and so on. The rows report the distances between adjacent pairs (with '20' referring to twenty or more), while the columns report the orientations.

```
##
##       + +   + -   - +   - -
##   0     0   350   420     0
##   1  1642   217   195  1713
##   2   930   158   161   956
##   3  4298   434   282  4254
##   4  1141   220   196  1157
##   5   114   122   159   137
##   6   210   164   178   194
##   7   201   165   162   207
##   8   188   145   126   167
##   9   176   142   143   154
##   10  193   169   153   190
##   11  162   149   138   154
##   12  142   112   140   171
##   13  156   142   117   152
##   14  150   129   106   127
##   15  156   137   104   139
##   16  148   126   130   141
##   17  129   132   117   105
##   18  135   125   117   135
##   19  127   126   141   136
##   20 37273 35431 35607 36837
```

As is evident, there are many more pairs with both sites lying in the same orientation at a distance of 1 to 5 than pairs with the sites in opposite orientations. There are no pairs of sites in the same orientation at the same position, because at most one position per patient in each orientation was used by us and because the data we received had evidently been filtered to remove instances in which the same position andd orientation appeared in more than one patient.

The data were filtered by performing hierarchical clustering on the positions in each patient and orientation, cutting the tree at a height of 5 (joining positions differing by no more than 5 as a cluster), determining the dominant position in each cluster according to read count, and then filtering out all but the one dominant position. After filtering, the sites were combined with the current post-infusion data (WAS-lenti). Here are the integration site numbers:

```
## trial
##     Braun WAS-lenti
##    156488     25358
```

The resulting dataset was compared to current post-infusion data (WAS-lenti) using the function `gRxCluster` using between 75 and 120 sites as the window width and filtering out windows spanning more than the tenth percentile of width for the number of sites they contain. Patient-wise permutation was used to establish False Discovery Rates for the discovered clumps.

A total of 280 clumps were discovered at an estimated FDR of 0.098. Of these 230 clumps were enriched for sites from the Braun trial, while 50 clumps were enriched for post-infusion sites from the current trial(WAS-lenti). The following table shows them all. In comparing the numbers of integration sites, recall that the ratio of sites is $\frac{\texttt{Braun sites}}{\texttt{Was-lenti sites}} = \frac{156488}{25358} = 6.17$

The FDR shown in the rightmost column is for the clump in that row and all clumps discovered at the same (or lesser) target for false discovery as the clump of that row.

Note that the clump at `chr11:33,560,205-33,935,491` overlaps LMO2, the clump at `chr12:3,848,551-4,270,056` overlaps CCND2, and the clump at `chr3:170,336,316-171,166,914` overlaps MECOM/EVI1. The rows for these clumps are highlighted in the table.

	chromo	start	end	width	Braun	WAS-Lenti	FDR
1	chr1	2731220	3274223	543004	400	2	0.05
2	chr1	8398200	8676506	278307	81	43	0.05
3	chr1	9047419	9076498	29080	96	1	0.06
4	chr1	16023117	16150731	127615	53	31	0.05
5	chr1	20995515	21353314	357800	116	77	0.05
6	chr1	26733475	26986899	253425	78	31	0.08
7	chr1	27552886	27777034	224149	65	35	0.05
8	chr1	28540944	28640264	99321	35	28	0.05
9	chr1	65051397	65471523	420127	148	0	0.05
10	chr1	92610151	92706447	96297	92	0	0.05
11	chr1	101327734	101549911	222178	112	0	0.05
12	chr1	108222206	108508897	286692	113	3	0.08
13	chr1	120336975	120364049	27075	168	0	0.05
14	chr1	143970125	143997243	27119	155	2	0.05
15	chr1	148721739	148864250	142512	98	0	0.05
16	chr1	196792190	196931384	139195	99	2	0.08
17	chr1	198386339	198594639	208301	126	2	0.06
18	chr1	205969263	206178826	209564	133	0	0.05
19	chr1	228348187	228493867	145681	83	0	0.06
20	chr1	232675869	233225687	549819	276	3	0.05
21	chr1	237633541	237821037	187497	92	1	0.06
22	chr2	8284001	8738746	454746	220	0	0.05
23	chr2	16283870	16535509	251640	196	4	0.05
24	chr2	37870285	38211261	340977	146	0	0.05
25	chr2	42973286	43060492	87207	135	2	0.06

26	chr2	45915040	46046848	131809	84	1	0.08
27	chr2	58651594	58704402	52809	80	1	0.09
28	chr2	58788098	58991567	203470	140	0	0.05
29	chr2	62381081	62589100	208020	101	2	0.07
30	chr2	66502728	66764946	262219	249	4	0.05
31	chr2	68801388	68835242	33855	91	0	0.05
32	chr2	69974031	70272277	298247	143	2	0.06
33	chr2	85387597	85779094	391498	141	1	0.05
34	chr2	108616156	108702146	85991	80	0	0.06
35	chr2	113098468	113315538	217071	155	1	0.05
36	chr2	207680190	207814138	133949	82	0	0.06
37	chr2	218859785	218980250	120466	76	1	0.10
38	chr2	238187995	238280476	92482	87	0	0.06
39	chr3	49689772	50085718	395947	57	85	0.02
40	chr3	69210892	69517768	306877	159	2	0.05
41	chr3	71515960	71589088	73129	98	0	0.05
42	chr3	72211136	72729788	518653	184	0	0.05
43	chr3	87892380	88199919	307540	121	3	0.07
44	chr3	113560395	113600134	39740	88	0	0.05
45	chr3	128960295	129023811	63517	86	1	0.07
46	chr3	153284746	153471144	186399	129	1	0.05
47	chr3	161135712	161432692	296981	136	1	0.05
48	chr3	170336316	171166914	830599	1980	14	0.05
49	chr3	186942362	186973656	31295	127	1	0.05
50	chr3	196081484	196387889	306406	129	0	0.05
51	chr3	197702394	197922897	220504	114	0	0.05
52	chr4	5079279	5103783	24505	81	0	0.06
53	chr4	15364657	15673597	308941	139	0	0.05
54	chr4	39866405	40101424	235020	116	0	0.05
55	chr4	41013073	41155501	142429	83	0	0.06
56	chr4	55201663	55574188	372526	190	3	0.05
57	chr4	75157719	75455306	297588	116	3	0.07
58	chr4	116294501	116471157	176657	114	2	0.06
59	chr4	140943953	141384697	440745	163	2	0.05
60	chr4	152136121	152240866	104746	85	0	0.06
61	chr4	156707576	156902598	195023	120	0	0.05
62	chr5	296693	460331	163639	112	3	0.08
63	chr5	10446123	10724035	277913	117	0	0.05
64	chr5	32198041	32267509	69469	83	0	0.06
65	chr5	32648528	33013153	364626	113	3	0.08
66	chr5	43020226	43208584	188359	108	0	0.05
67	chr5	76026817	76190845	164029	70	0	0.08
68	chr5	134707883	134804390	96508	77	0	0.06
69	chr5	138543185	139104589	561405	219	7	0.06
70	chr5	169625918	169752329	126412	77	1	0.10
71	chr5	171467398	171573422	106025	83	0	0.06
72	chr6	2580294	2810508	230215	212	5	0.05
73	chr6	6597248	6772594	175347	147	0	0.05
74	chr6	11421650	11520766	99117	76	0	0.06
75	chr6	25134112	25193485	59374	77	0	0.06
76	chr6	32375938	32825030	449093	88	67	0.05
77	chr6	34347667	34476637	128971	58	41	0.05
78	chr6	35105106	35128138	23033	79	1	0.09
79	chr6	37243773	37269816	26044	102	0	0.05
80	chr6	42082805	42125908	43104	136	0	0.05
81	chr6	42642975	42897127	254153	61	47	0.05

82	chr6	45452739	45585985	133247	82	0	0.06
83	chr6	74210706	74547565	336860	128	1	0.05
84	chr6	90941298	90998330	57033	98	0	0.05
85	chr6	109717618	109751861	34244	85	0	0.06
86	chr6	124154879	124276662	121784	82	1	0.08
87	chr6	133061694	133189508	127815	97	2	0.08
88	chr6	149394398	149650737	256340	120	0	0.05
89	chr7	2613632	2746883	133252	129	0	0.05
90	chr7	5423874	5642298	218425	172	1	0.05
91	chr7	23341896	23497248	155353	87	1	0.07
92	chr7	26538585	26711648	173064	204	3	0.05
93	chr7	37348430	37692560	344131	160	4	0.06
94	chr7	105458469	105507454	48986	78	0	0.06
95	chr7	130305498	130398127	92630	139	0	0.05
96	chr7	134481259	134559864	78606	81	0	0.06
97	chr7	141819154	141899849	80696	83	0	0.06
98	chr7	142092497	142112370	19874	75	0	0.06
99	chr8	6361229	6592264	231036	111	0	0.05
100	chr8	27237837	27318190	80354	81	0	0.06
101	chr8	30366956	30446134	79179	119	0	0.05
102	chr8	56846906	56997802	150897	92	0	0.05
103	chr8	66998311	67085539	87229	77	0	0.06
104	chr8	69027414	69145603	118190	78	1	0.09
105	chr8	104101433	104287341	185909	92	0	0.05
106	chr8	108369748	108431231	61484	111	2	0.06
107	chr8	121032380	121165353	132974	84	0	0.06
108	chr8	134065090	134225436	160347	89	0	0.05
109	chr8	134537864	134650889	113026	82	1	0.08
110	chr9	458618	828646	370029	133	1	0.05
111	chr9	5551546	5852443	300898	134	1	0.05
112	chr9	6670668	6967835	297168	122	0	0.05
113	chr9	20109681	20399259	289579	174	0	0.05
114	chr9	70698776	70995973	297198	115	3	0.07
115	chr9	74742485	75048504	306020	218	2	0.05
116	chr9	78310631	78497284	186654	126	1	0.05
117	chr9	99732539	99954328	221790	133	0	0.05
118	chr9	123022559	123360913	338355	138	4	0.09
119	chr9	124828409	124839924	11516	78	0	0.06
120	chr9	127550640	127724364	173725	56	34	0.05
121	chr9	131670790	131847699	176910	78	35	0.05
122	chr9	138366121	138951939	585819	123	118	0.02
123	chr10	3781144	4015766	234623	108	0	0.05
124	chr10	6128499	6278199	149701	91	0	0.05
125	chr10	11214888	11416853	201966	133	0	0.05
126	chr10	17209470	17594652	385183	202	3	0.05
127	chr10	30547757	31092782	545026	184	1	0.05
128	chr10	49324424	49455809	131386	83	1	0.08
129	chr10	63172954	63229635	56682	98	0	0.05
130	chr10	73027645	73193482	165838	101	0	0.05
131	chr10	80477859	80620273	142415	89	0	0.05
132	chr10	101078511	101371629	293119	124	0	0.05
133	chr10	105409231	105666097	256867	107	0	0.05
134	chr10	116495238	116573157	77920	75	0	0.06
135	chr10	130720416	130823699	103284	75	0	0.06
136	chr10	133992131	134225439	233309	114	1	0.05
137	chr11	3754061	3858256	104196	100	1	0.06

138	chr11	9568885	9700448	131564	149	1	0.05
139	chr11	10561241	10728778	167538	156	4	0.06
140	chr11	33560205	33935491	375287	571	1	0.05
141	chr11	62076623	62385308	308686	81	53	0.05
142	chr11	64518271	65827066	1308796	273	420	0.00
143	chr11	66581788	66866076	284289	37	153	0.00
144	chr11	71242981	71492970	249990	69	44	0.05
145	chr11	72234484	72547013	312530	55	76	0.02
146	chr11	74715586	74788727	73142	77	0	0.06
147	chr11	109216250	109497229	280980	119	1	0.05
148	chr11	117597557	117618933	21377	87	0	0.06
149	chr11	120915260	120966572	51313	76	1	0.10
150	chr11	127916781	128221466	304686	250	0	0.05
151	chr12	544545	706633	162089	155	3	0.06
152	chr12	3848551	4270056	421506	418	8	0.05
153	chr12	6418296	6884530	466235	116	69	0.05
154	chr12	9985879	10102378	116500	95	1	0.06
155	chr12	11589421	11942520	353100	166	1	0.05
156	chr12	13069321	13300481	231161	94	0	0.05
157	chr12	14993462	15269969	276508	111	0	0.05
158	chr12	24837304	25099744	262441	124	4	0.09
159	chr12	29145472	29331974	186503	165	3	0.05
160	chr12	45133307	45341737	208431	117	1	0.05
161	chr12	52052119	52298435	246317	74	36	0.05
162	chr12	64502031	64645449	143419	359	5	0.05
163	chr12	66996873	67150042	153170	93	2	0.09
164	chr12	74128111	74164536	36426	104	0	0.05
165	chr12	92649727	92766004	116278	86	1	0.07
166	chr12	94963890	95140897	177008	95	1	0.06
167	chr12	100645942	100903472	257531	115	0	0.05
168	chr12	103394136	103691601	297466	124	0	0.05
169	chr12	120499972	120634115	134144	99	0	0.05
170	chr12	123904661	124106420	201760	98	0	0.05
171	chr13	20449445	20611824	162380	79	0	0.06
172	chr13	29778779	29949330	170552	92	0	0.05
173	chr13	31502219	31590540	88322	80	1	0.09
174	chr13	40025402	40152282	126881	127	0	0.05
175	chr13	40452692	40532051	79360	77	0	0.06
176	chr13	45525068	45683628	158561	119	0	0.05
177	chr13	48991748	49171369	179622	95	1	0.06
178	chr13	90585437	90739178	153742	135	0	0.05
179	chr14	21947684	22001746	54063	106	1	0.06
180	chr14	33454342	33616744	162403	92	0	0.05
181	chr14	34753230	35024577	271348	137	0	0.05
182	chr14	49502366	49653755	151390	131	0	0.05
183	chr14	60822193	61025811	203619	96	0	0.05
184	chr14	76450737	76635311	184575	203	1	0.05
185	chr14	77107563	77143880	36318	79	0	0.06
186	chr14	99586628	99773824	187197	174	1	0.05
187	chr14	105356310	105679229	322920	103	60	0.05
188	chr14	106148196	106348468	200273	68	49	0.05
189	chr15	68523342	68655021	131680	92	0	0.05
190	chr15	72773891	72951961	178071	117	4	0.09
191	chr15	76111041	76211769	100729	92	0	0.05
192	chr15	79102990	79402956	299967	118	3	0.07
193	chr15	91152348	91228804	76457	77	0	0.06

194	chr15	99443489	99654404	210916	93	1	0.06
195	chr16	1384456	1829733	445278	53	117	0.01
196	chr16	10739176	11028229	289054	138	0	0.05
197	chr16	12342815	12462554	119740	83	1	0.08
198	chr16	16061121	16093280	32160	92	1	0.06
199	chr16	17332019	17420258	88240	112	0	0.05
200	chr16	23768875	23896416	127542	183	3	0.05
201	chr16	29316017	29581055	265039	52	43	0.05
202	chr16	30176279	30674491	498213	112	61	0.06
203	chr16	56191198	56288143	96946	141	3	0.07
204	chr16	66666493	66894714	228222	58	34	0.05
205	chr16	68113091	68343647	230557	54	43	0.05
206	chr16	73584413	73692620	108208	165	2	0.05
207	chr16	80196926	80424728	227803	108	1	0.05
208	chr16	87179373	87561812	382440	88	50	0.05
209	chr17	1926159	1951190	25032	108	0	0.05
210	chr17	1952831	2353427	400597	119	57	0.05
211	chr17	4067312	4377248	309937	64	69	0.02
212	chr17	7099869	7856255	756387	159	136	0.02
213	chr17	22683322	22932684	249363	104	1	0.06
214	chr17	37589410	37702619	113210	65	41	0.05
215	chr17	37731943	37905887	173945	50	58	0.03
216	chr17	43858061	44051339	193279	132	1	0.05
217	chr17	50470703	50780202	309500	228	6	0.05
218	chr17	52111408	52222592	111185	75	0	0.06
219	chr17	52717487	52955124	237638	140	0	0.05
220	chr17	55082999	55260590	177592	67	33	0.05
221	chr17	59407909	59506983	99075	95	0	0.05
222	chr17	60317523	60437485	119963	82	1	0.08
223	chr17	70646727	71129777	483051	72	83	0.02
224	chr17	71178211	71853758	675548	120	152	0.01
225	chr17	72871030	73088516	217487	127	4	0.09
226	chr17	73472627	74356442	883816	139	153	0.00
227	chr17	76159169	76405301	246133	83	63	0.05
228	chr17	77016569	77296851	280283	49	72	0.02
229	chr18	2961048	3054506	93459	86	1	0.07
230	chr18	3575561	3615435	39875	90	0	0.05
231	chr18	5442895	5536566	93672	76	1	0.10
232	chr18	9044864	9110051	65188	104	0	0.05
233	chr18	13342898	13570797	227900	137	0	0.05
234	chr18	27851730	27918868	67139	79	0	0.06
235	chr18	40279277	40953730	674454	256	4	0.05
236	chr18	44680972	44838152	157181	92	0	0.05
237	chr18	50580944	50764404	183461	90	1	0.07
238	chr18	58686922	59186066	499145	288	1	0.05
239	chr18	66435320	66545486	110167	86	1	0.07
240	chr18	72815864	73114300	298437	143	0	0.05
241	chr19	1780633	2276073	495441	116	70	0.04
242	chr19	3085066	3391087	306022	79	42	0.05
243	chr19	6680222	6924264	244043	75	47	0.05
244	chr19	11023555	11294169	270615	82	34	0.06
245	chr19	13819168	14116005	296838	71	35	0.06
246	chr19	17999630	18290883	291254	87	34	0.05
247	chr19	46748910	46968544	219635	64	36	0.05
248	chr19	49915982	50051883	135902	80	0	0.06
249	chr19	53255850	53451134	195285	54	39	0.05

250	chr19	54403103	54816529	413427	42	116	0.01
251	chr20	2989441	3027697	38257	97	0	0.05
252	chr20	3940678	4136761	196084	107	0	0.05
253	chr20	4702765	5056651	353887	141	0	0.05
254	chr20	9046782	9131818	85037	139	0	0.05
255	chr20	10193114	10603639	410526	258	1	0.05
256	chr20	29485582	29790241	304660	139	0	0.05
257	chr20	30495975	30737922	241948	154	0	0.05
258	chr20	45410863	45454525	43663	132	0	0.05
259	chr20	49538925	49753392	214468	107	1	0.05
260	chr20	51504428	51814692	310265	267	2	0.05
261	chr20	54400321	54477456	77136	105	0	0.05
262	chr20	57962815	58142080	179266	94	1	0.06
263	chr21	14678853	14979736	300884	121	4	0.09
264	chr21	15413268	15809085	395818	336	5	0.05
265	chr21	16470691	16538073	67383	102	2	0.07
266	chr21	25765945	26031880	265936	132	0	0.05
267	chr21	35178167	35399650	221484	102	2	0.07
268	chr21	37551482	37665869	114388	75	0	0.06
269	chr21	38512931	39486285	973355	415	9	0.05
270	chr21	42299833	42797769	497937	256	7	0.05
271	chr21	44266853	44447893	181041	65	27	0.09
272	chr21	45118862	45185070	66209	114	1	0.05
273	chr22	26272999	26572395	299397	187	1	0.05
274	chr22	27521880	27546796	24917	151	0	0.05
275	chr22	34816457	35060989	244533	124	0	0.05
276	chr22	35992535	36307153	314619	133	3	0.07
277	chr22	49153607	49458277	304671	56	64	0.03
278	chrX	1392807	1736242	343436	117	0	0.05
279	chrX	12809147	13123394	314248	117	2	0.06
280	chrY	21810424	21979675	169252	83	0	0.06

Table 8: Clumps Discovered Comparing Post-Infusion to WAS γ-Retroviral trial of Braun et al.

9 Software Used

This report was prepared with `emacs org-mode`, `R`, and `R` packages. Here is some detail about the computing environment.

- R version 3.1.2 (2014-10-31), `x86_64-apple-darwin10.8.0`
- Locale: `C`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils
- Other packages: BiocGenerics 0.12.0, GenomeInfoDb 1.2.0, GenomicRanges 1.18.0, IRanges 2.0.0, S4Vectors 0.4.0, cellTypeImputation 0.1-7, geneRxCluster 1.2.0, knitr 1.7, xtable 1.7-4

- Loaded via a namespace (and not attached): XVector 0.6.0, codetools 0.2-9, digest 0.6.4, evaluate 0.5.5, formatR 1.0, highr 0.4, stringr 0.6.2, tools 3.1.2

References

[Berry et al., 2014] Berry, C. C., Ocwieja, K. E., Malani, N., and Bushman, F. D. (2014). Comparing DNA site clusters with Scan Statistics. *Bioinformatics*.

[Braun et al., 2014] Braun, C. J., Boztug, K., Paruzynski, A., Witzel, M., Schwarzer, A., Rothe, M., Modlich, U., Beier, R., Göhring, G., Steinemann, D., et al. (2014). Gene therapy for wiskott-aldrich syndrome—long-term efficacy and genotoxicity. *Science translational medicine*, 6(227):227ra33–227ra33.

[Hacein-Bey-Abina et al., 2014] Hacein-Bey-Abina, S., Pai, S.-Y., Gaspar, H. B., Armant, M., Berry, C. C., Blanche, S., Bleesing, J., Blondeau, J., de Boer, H., Buckland, K. F., et al. (2014). A modified γ-retrovirus vector for x-linked severe combined immunodeficiency. *New England Journal of Medicine*, 371(15):1407–1417.

[Lawrence et al., 2013] Lawrence, M., Huber, W., Pagès, H., Aboyoun, P., Carlson, M., Gentleman, R., Morgan, M. T., and Carey, V. J. (2013). Software for computing and annotating genomic ranges. *PLoS Computational Biology*, 9(8):e1003118.

[R Core Team, 2013] R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

10 Appendix - R code

```
### Chunk: unnamed-chunk-1

suppressPackageStartupMessages(
    require(GenomicRanges))
suppressPackageStartupMessages(
    require(geneRxCluster))
suppressPackageStartupMessages(
    require(xtable))
opts_chunk$set(tidy=FALSE,echo=FALSE)

### Chunk: load-was
```

```
print(   # print object name
    load(
        print(  # print file name
            Sys.glob("data/was.gr*"))))
sqlev <- seqlevels(was.gr)
seqinfo(was.gr) <-
    seqinfo(BSgenome.Hsapiens.UCSC.hg18:::Hsapiens)[sqlev,]

patient.codes <- c( "pFR03"="Pt4", "pFR01"="Pt2", "pWAS_UK04"="Pt6",
"pWAS_UK02"="Pt1", "pFR04"="Pt5", "pFR02"="Pt3")

was.gr$patient <- patient.codes[was.gr$patient]

### Chunk: move-start

siteID <- paste(seqnames(was.gr),
                start(was.gr),
                strand(was.gr))
pt.pp.unique <-
    !duplicated(cbind(siteID,
                      was.gr$patient,
                      was.gr$infusion))
was.unique <- was.gr[pt.pp.unique,]
## move start to lower site of attack:
was.unique <-
    shift(was.unique,
          as.vector(0L+(strand(was.unique)=="+")*5L))

## order by seqnames, start, and use runif
## to shuffle multiple hits

was.unique.indx <-
    order(as.vector(seqnames(was.unique)),
          as.vector(start(was.unique)),
          runif(length(was.unique)))
was.unique <- was.unique[ was.unique.indx ,]

### Chunk: adjacency-check

start.list <-
    split(start( was.unique ),
          paste0( was.unique$patient,
                  was.unique$infusion,
                  seqnames(was.unique)))
```

```
diff.list <-
    lapply(start.list,diff)


strand.list <-
    split(strand( was.unique ),
          paste0( was.unique$patient,
                  was.unique$infusion,
                  seqnames(was.unique)))

strand.pair <-
    lapply(strand.list,
           function(x) paste0( head(x,-1),
                               tail(x,-1)))

### Chunk: table-diff-by-strand

table( pmin(10, unlist(diff.list )),
       unlist( strand.pair ))

### Chunk: cull-1-pp-or-mm

  was.split <- split(was.unique,was.unique$patient)

## make a GRangesList of adjacent sites on same strand
  contig.same.strand <-
      function(x,strnd)
  {
      this.strand <- strand(x)==strnd
      y <- coverage(x, weight=as.integer(this.strand))
      y.2 <-
          endoapply(y,
                    function(z) {
                        runValue(z)[runLength(z)==1] <- 0
                        z})
      y.2.gr <- as(y.2,"GRanges")
      hits <- findOverlaps(x[this.strand,],
                           y.2.gr[ y.2.gr$score>0,])
      split(x[ this.strand,][queryHits(hits),],
            subjectHits(hits))
  }


  was.contigs <-
```

```
      lapply(was.split,
             function(x) {
                 plus <- contig.same.strand(x,strnd="+")
                 minus <- contig.same.strand(x,strnd="-")
                 c(unname(plus),unname(minus))
             })

## retain those to be deleted
  contig.flag.dels <-
      function(x) {
          cc <- x$clonecount +
              runif(length(x$clonecount))
          bad <- (cc < max(cc)) &
              all(x$infusion==x$infusion[1])
          res <- x[bad,]
          res
      }

  bad.contigs <-
      lapply(was.contigs,
             function(y) {
                 if (length(y))
                     endoapply( y,contig.flag.dels)
                 else
                     GRanges()
             })

## mark the elements to remove

  was.unique.ok <-
      mapply(
          function(x,y) !(x %over% y),
          was.split,
          bad.contigs)


  was.no.derep <- unsplit(was.unique.ok,was.unique$patient)

  was.derep <- was.unique[was.no.derep,]

### Chunk: inf-per-pt

table(was.derep$patient,was.derep$infusion)
```

```
### Chunk: pt-pt-tab

 pp.tab <-
    table(head(was.derep$patient,-1),
          tail(was.derep$patient,-1),
          head(as.vector(seqnames(was.derep)),-1) ==
          tail(as.vector(seqnames(was.derep)),-1))[,,"TRUE"]
pp.tab

### Chunk: test-pp-tab

pp.fit <- loglin( pp.tab, 1:2, fit=TRUE, print = FALSE )$fit
round( pp.fit , 1)

### Chunk: unnamed-chunk-2

round( pp.tab-pp.fit , 1)

### Chunk: chisq

chisq.test( pp.tab )

### Chunk: to-each-his-own

each.own <-
    sapply(sort(unique(was.derep$patient)),
           function(x) {
               gr <- was.derep[was.derep$patient==x,]
               res <-
                   gRxCluster(seqnames(gr),start(gr),
                              gr$infusion=="post",
                              kvals=10:50,nperm=100,
                              cutpt.filter.expr=5e6,
                              cutpt.tail.expr=
                              critVal.target(k,n,
                                             target=5/6,
                                             posdiff=x))
               seqinfo(res) <-
                   intersect(seqinfo(res),
                             seqinfo(was.derep))
               res
           },simplify=FALSE)

### Chunk: unnamed-chunk-3
```

```
summaryTab <-
    function(x) unlist(gRxSummary(x)[1:2])
  round(t( sapply( each.own, summaryTab )),2)

### Chunk: unify

all.clumps <- each.own[[1]]
for (i in 2:length(each.own))
    if (length(each.own[[i]]))
        all.clumps <-
    suppressWarnings(union( all.clumps,
                           each.own[[i]]))

each.over <-
    sapply(each.own,function(x) all.clumps%over%x)

### Chunk: unnamed-chunk-4

table(rowSums(each.over))

### Chunk: unnamed-chunk-5

  clump.tab <-
      function(x) {
          xtabs(~was.derep$infusion +
                was.derep%over%x +
                was.derep$patient)}

  all.odds.tab <-
      lapply(seq(along=all.clumps),
             function(x) clump.tab(all.clumps[x,]))

## positive favors post-in-clump
  all.odds <-
      sapply(all.odds.tab,
             function(x) apply(x,3,
                               function(y) {
                                   y <- y + 0.5
                                   sum(log(y)*c(-1,1,1,-1))
                               }))


  clump.favor <-
      c("pre","","post")[2+sign(t(all.odds))]
```

```
  clump.favor[ each.over ] <-
      toupper(clump.favor[ each.over ])
  dim(clump.favor) <- dim(each.over)
  dimnames(clump.favor) <- dimnames(each.over)

  mcols(all.clumps) <- as(clump.favor,"DataFrame")

### Chunk: unnamed-chunk-6

table(PRE=rowSums(clump.favor=="PRE"),
POST=rowSums(clump.favor=="POST"))

### Chunk: show-two-pts-grp

print(as.data.frame(
    mcols(all.clumps))[rowSums(each.over)>1,],
      quote=FALSE)

### Chunk: show-two-pts-loc

all.df <-
    as.data.frame(all.clumps)[rowSums(each.over)>1,1:4]
colnames(all.df)[1] <- "chromo"
all.df

### Chunk: all2gether

all.pt <-
  gRxCluster(seqnames(was.derep),
             start(was.derep),
             was.derep$infusion=="post",
             kvals=10:50)

### Chunk: permutation-over-pt

pp.perms <-
    as.matrix(
        expand.grid(
            rep( list(c("pre","post")), 6)))
colnames(pp.perms) <-
    sort(unique(was.derep$patient))

### Chunk: perm-clumps-by-pt
```

```
perm.call <-
    metadata(all.pt)["call"]


perm.res <- list()
## by symmetry only need first half
for (x in 1:32) {
    perm.grp <-
        was.derep$infusion ==
            pp.perms[x,was.derep$patient]
    perm.res[[x]] <-
        update(perm.call, group=perm.grp)}

### Chunk: fdr-per-pt

perm.targets <-
    unlist(sapply(perm.res,
                  function(x) x$target.min))

tm.obs <-
    sort(unique(all.pt$target.min))

perm.vs.all <-
    sapply(tm.obs,
           function(x) sum(perm.targets<=x))

fdr <-
    perm.vs.all/ 32 /
    (1+1:(length(unique(all.pt$target.min))))

fdr <- rev(cummin(rev(fdr)))

fdr.per.clump <-
    fdr[match(all.pt$target.min, tm.obs)]

### Chunk: all-lt-pt-2

gtab <-
    cbind(
        as.data.frame(
            all.pt[ fdr.per.clump <0.15,])[,c(1,2,3,4,6,7)],
        FDR=round(fdr.per.clump[fdr.per.clump<0.15],3))
gtab$value1 <- as.integer(gtab$value1)
gtab$value2 <- as.integer(gtab$value2)
```

```
colnames(gtab)[1] <- "chromo"
tab <- xtable(gtab)
caption(tab) <- "Clumps Discovered Using All Patients."
print(tab,
      digits=3,
      table.placement="H",
      size="scriptsize"
      )

### Chunk: tab-all-pt-over-singles

table(factor(all.pt[fdr.per.clump<0.15,] %over%
      all.clumps[ rowSums(each.over)>1,],
             c(TRUE,FALSE), c("Over","Not Over")))

### Chunk: tab-single-over-all

table(factor(all.clumps[ rowSums(each.over)>1,] %over%
    all.pt[fdr.per.clump<0.15,],
             c(TRUE,FALSE), c("Over","Not Over")))

### Chunk: read-genes

onco.files <-
    Sys.glob("../data/Oct2013/*onco*RData")
onco.objs <-
    lapply(onco.files,load,env=globalenv())
names(onco.objs) <- sub(".gr","",onco.objs)
onco.labels <- c("LymBody","LymEdge",
                 "AllBody","AllEdge")
names(onco.objs) <- onco.labels

### Chunk: each-over

all.2plus <- all.clumps[rowSums(each.over)>1,]
onco.res.each <-
    suppressWarnings(
        sapply(onco.objs,function(y) {
             yg <- (get(y))
             sapply(each.own,function(x) {
                 x <- x[x%over%all.2plus,]
                 sum(x%over%yg)})}))

onco.res.each
```

```
### Chunk: each-own-perm

eo.call <- metadata(each.own[[1]])$call
eo.call$group <- call("sample",eo.call$group)
eo.call$nperm <- 0L

eo.perms <-
    suppressWarnings(
        replicate(50,{
            eo.perm <-
                sapply(
                    sort(unique(was.derep$patient)),
                    function(x) {
                        gr <- was.derep[was.derep$patient==x,]
                        res <- eval(eo.call)
                        seqinfo(res) <-
                            intersect(seqinfo(res),
                                      seqinfo(was.derep))
                        res
                    },simplify=FALSE)
            eo.perm <- Filter(length,eo.perm)
            if (length(eo.perm)){
                all.clumps.perm <- eo.perm[[1]]
                for (i in seq(along=eo.perm)[-1])
                    if (length(eo.perm[[i]]))
                        all.clumps.perm <-
                            union( all.clumps.perm,
                                  eo.perm[[i]])
                each.over.perm <-
                    sapply(eo.perm,
                           function(x) all.clumps.perm%over%x)
                list(each=eo.perm,clumps=all.clumps.perm,
                     over=each.over.perm)}}))

### Chunk: unnamed-chunk-7

perm.over.genes <-
  lapply(eo.perms,
    function(xo) {
      if (!is.null(xo) &&
        any(gt1 <-
          rowSums(as.matrix(xo$over))>1)){
        all.over <- xo$clump[gt1,]
        suppressWarnings(
```

```
          sapply(onco.objs,function(y) {
            yg <- (get(y))
            sapply(xo$each,
                 function(z) {
                   sum(z[z%over%all.over,]%over%yg)})}))
      } else {
        matrix(c(0,0,0,0),nrow=1)}})

### Chunk: adft

perm.onco.df <-
    cbind(as.data.frame.table(onco.res.each),rep=0)
perm.onco.df$Freq <- 0
for (itab in seq(along=perm.over.genes))
    perm.onco.df <-
    rbind( perm.onco.df,
          if (length(dimnames(perm.over.genes[[itab]])))
          cbind(as.data.frame.table(perm.over.genes[[itab]]),
                rep=itab)
          else
          cbind(as.data.frame.table(onco.res.each*0),rep=itab))
po.tab <- xtabs(Freq~.,perm.onco.df)
exp.onco.res <- apply(po.tab[,,-1],1:2,mean)

onco.res.pvals <- exp.onco.res
onco.res.pvals[] <-
    round(
        p.adjust(1 -
                 pchisq((onco.res.each-exp.onco.res)^2/
                        (pmax(1,exp.onco.res)),1),"holm"),
        2)
mode(onco.res.pvals) <- "character"
onco.res.pvals[onco.res.pvals=="0"] <- "<0.01"
print(onco.res.pvals,quote=FALSE)

### Chunk: patients-clumps-onco

clump.number <-
    function(x,y)
{
    hit <- findOverlaps(x,y,select="first")
    hit[is.na(hit)] <- 0
    hit}
```

```
## positive favors post-in-clump
log.odds.ratio <-
    function(x)
{
    z <- colSums(x)+1
    y <- log(x+0.5)
    yc <- log(sweep(x,2,z,function(a,b) b-a))
    y[,1]-y[,2]-yc[,1]+yc[,2]
}

## return GRanges with odds

pt.odds <-
    function(x,y,z=NULL)
{
    if (!is.null(z))
        y <- y[y %within% z,]
    tab <-
        table(clump.number(x,y),x$infusion)
    mcols(y)$log.OR <- log.odds.ratio(tab)[-1]
    y
}

## apply it

pt.log.odds <-
    mapply(pt.odds,
           split(was.derep,was.derep$patient),
           each.own,SIMPLIFY=FALSE)

### Chunk: pt-each-over-genes

density.of.sites <-
    function(x,pt,y){
        expect <-
            table(y$infusion[y$patient==pt]) %o%
                as.numeric(width(x))/
                    sum(as.numeric(seqlengths(y)))
        oe.pre <- x$value1/expect['pre',]
        oe.post <- x$value2/expect['post',]
        x$pre.OE <- oe.pre
        x$post.OE <- oe.post
        x}
```

```
pt.log.odds <-
    mapply(density.of.sites,
           pt.log.odds,
           names(pt.log.odds),
           MoreArgs=list(y=was.derep))

pt.smry <-
    lapply(pt.log.odds,
           function(x) {
               for (i.onco in names(onco.objs)){
                   onc.gr <- get(onco.objs[[i.onco]])
                   mcols(x)[,i.onco] <-
                       suppressWarnings(
                           countOverlaps(x, onc.gr))
               }
               x})

pt.smry.df <-
    lapply(pt.smry,
           function(x){
               y <- x[ x %within%
                      all.clumps[ rowSums(each.over)>1,],]
               res <-
                   as.data.frame(y)[,c(1,2,3,10:16)]
               colnames(res)[1] <- "chromo"
               res
           })

### Chunk: pt-eog-show

require(xtable)
for (i in names(pt.smry.df))
    if (nrow(pt.smry.df[[i]]))
        {
            tab <- xtable(pt.smry.df[[i]])
            caption(tab) <- sub("_","\\\\_",i)
            print(tab,
                  include.rownames=FALSE,
                  digits=3,
                  table.placement="H",
                  size="tiny"
                  )
        }
```

```
### Chunk: all-over-onco

tm.15 <-
    max(all.pt$target.min[fdr.per.clump<0.15])

onco.olaps <-
    suppressWarnings(
        lapply(onco.objs,
               function(x) {
                   onco.hit <-
                       countOverlaps(
                           all.pt[all.pt$target.min<=tm.15,],
                           get(x))
               }))

df <- as.data.frame(onco.olaps)
rownames(df) <- which(all.pt$target.min<=tm.15)


all.pt.odds <-
    pt.odds(was.derep,all.pt[rownames(df),])

seqinfo(all.pt) <- seqinfo(was.derep)

all.pt.expect <-
    table(was.derep$infusion) %o%
    as.numeric(width(all.pt.odds)) /
    sum(as.numeric(seqlengths(all.pt)))

df$log.OR <- all.pt.odds$log.OR
df$pre.OE <- all.pt.odds$value1/all.pt.expect['pre',]
df$post.OE<- all.pt.odds$value2/all.pt.expect['post',]

tab <- xtable(df)
caption(tab) <- "Gene Overlaps for Clumps Discovered Using All Patients"
print(tab,
      digits=3,
      table.placement="H",
      size="tiny"
      )

### Chunk: perm-onc

onco.res.ref <-
```

```
    suppressWarnings(
        sapply(onco.objs,function(y) {
            yg <- GIntervalTree(get(y))
            sapply(perm.res,function(x) {
                x <- x[x$target.min<tm.15,]
                sum(x%over%yg)})}))

onco.p.vals <-
    colMeans(onco.res.ref>=
             rep(onco.res.ref[1,],each=nrow(onco.res.ref)))
round(p.adjust(onco.p.vals,method="holm"),3)

### Chunk: unnamed-chunk-8

## Previously the code in these comments was run to produce
## a GRanges object called "braun.gr"

## The file "data/Braun.csv" is the CSV version of the EXCEL
## spreadsheet containing the data supplied by Braun et al.

## NCThg18 <- read.csv("data/Braun.csv")
## braun.df <- NCThg18[,1:13]
## save(braun.df,file="data/braun.df.RData")

## require(GenomicRanges)
## braun.gr <-
##     GRanges(seqnames=braun.df$Chromosome,
##             IRanges(start=braun.df$Integration.Locus,width=1),
##             strand=braun.df$Sequence.Orientation)
## mcols(braun.gr) <- as(braun.df,"DataFrame")
## braun.gr <- sortSeqlevels(braun.gr)
## braun.gr <- sort(braun.gr,ignore.strand=TRUE)
## save(braun.gr,file="data/braun.gr.RData")

require(GenomicRanges)
load("data/braun.gr.RData")

 braun.unique <- unlist(unique(split(braun.gr,braun.gr$Patient)))
 braun.unique <- sortSeqlevels(braun.unique)
 braun.unique <- sort(braun.unique,ignore.strand=TRUE)

 ### split by patient - add read.counts annotation

 braun.counted <-
```

```
     mendoapply(function(x,y)
         {z <- countOverlaps(x,y)
          mcols(x)$count <- z
          x},
       splitAsList(braun.unique,braun.unique$Patient),
       splitAsList(braun.gr,braun.gr$Patient))

 ### - look for adjacent sites in each patient
 ### - number adjacent sites in order 1, 2, 3 ... by position

.abundUtes <-  if (!("abundUtils" %in% search())){
    abundEnv <- attach(NULL,name="abundUtils")
    sys.source("../../abundance/src/utils.R",abundEnv)
}

bus.adj2 <-
    endoapply(braun.counted,
              function(x) {
                  ss <- as.vector(paste(seqnames(x),strand(x)))
                  st.ss <- split(as.vector(start(x)),ss)
                  cid.ss <- sapply(st.ss,cutStart,ht=5)
                  cid <- paste(unsplit(cid.ss,ss),ss)
                  mcols(x)$cluster <- cid
                  mcols(x)$rel.start <-
                      ave(start(x),cid,FUN=rank)
                  mcols(x)$cl.length <-
                      as.numeric(ave(cid,cid,FUN=length))
                  mcols(x)$cl.count <-
                      ave(as.vector(x$count),cid,FUN=sum)
                  x})

 br.adj.gr <- unsplit(bus.adj2,braun.unique$Patient)

 part1 <- with(as.data.frame(braun.gr,row.names=NULL),
               paste(Patient,seqnames,strand,start))
 part2 <- with(as.data.frame(br.adj.gr,row.names=NULL),
               paste(Patient,seqnames,strand,start))
 index2 <- match(part1,part2)

 new.cols <- setdiff(colnames(mcols(br.adj.gr)),
                     colnames(mcols(braun.gr)))

 mcols(braun.gr) <-
     cbind(mcols(braun.gr), mcols(br.adj.gr)[index2,new.cols])
```

```
### Chunk: unnamed-chunk-9

table(pmin(20,pmax(-1,diff(start(braun.unique)))),
      paste(head(as.character(strand(braun.unique)),-1),
            tail(as.character(strand(braun.unique)),-1)))[-1,]

### Chunk: do-f-b-b

max.element <-
    function(x){
        if (length(x)==1){
            TRUE
        } else {
            x <- x+runif(length(x));
            x==max(x)}
    }

braun.best <-
    with(as.data.frame(braun.gr),
                  ave(Sequence.Count,paste(Patient,cluster),
                      FUN=max.element))

braun.cl.start <- start(braun.gr)[braun.best==1]
pt.cluster <- paste(braun.gr$Patient,braun.gr$cluster)



mcols(braun.gr)$best.start <-
    braun.cl.start[match(pt.cluster,
                         pt.cluster[braun.best==1])]

braun.derep <- braun.gr[braun.best==1,]

### Chunk: make-comparisons clumps

braun.sites <- braun.derep
postInfuse <- was.derep[was.derep$infusion=="post",]
mcols(braun.sites) <- NULL
mcols(braun.sites)$trial <- "Braun"
mcols(braun.sites)$patient <- paste("Br",braun.derep$Patient)

pi.patient <- mcols(postInfuse)$patient
mcols(postInfuse) <- NULL
mcols(postInfuse)$trial <- "WAS-lenti"
```

```
mcols(postInfuse)$patient <- pi.patient

br.vs.loc <- c(braun.sites,postInfuse)
br.vs.loc <- sortSeqlevels(br.vs.loc)
br.vs.loc <- sort(br.vs.loc,ignore.strand=TRUE)

bvl.cluster <-
    gRxCluster(seqnames(br.vs.loc),start(br.vs.loc),
               br.vs.loc$trial=="WAS-lenti",
               kvals=c(75L:120L),
               cutpt.filter.expr =
                   as.double(apply(x,2,quantile,
                                   na.rm=TRUE,probs=0.1)))

n.perms <- 100
pt.all <- unique(br.vs.loc$patient)
n.pt.loc <-
    length(
        unique(br.vs.loc$patient[br.vs.loc$trial=="WAS-lenti"]))
pt.perms <-
    lapply(1:n.perms,
           function(x) {
               br.vs.loc$patient %in% sample(pt.all,n.pt.loc)
           })

perm.call <-
    metadata(bvl.cluster)["call"]
require(parallel)

bvl.perm <-
    mclapply(pt.perms,function(x) update(perm.call, group=x),
             mc.cores=3L)

bvl.perm.targets <-
    unlist(sapply(bvl.perm,
                  function(x) x$target.min))

tm.obs <-
    sort(unique(bvl.cluster$target.min))

perm.vs.all <-
    sapply(tm.obs,
           function(x) sum(bvl.perm.targets<=x))
r.discover <- table(factor(bvl.cluster$target.min,tm.obs))
```

```
bvl.fdr <-
    perm.vs.all/ n.perms /
        (1+cumsum(r.discover))

bvl.fdr <- rev(cummin(rev(bvl.fdr)))

bvl.fdr.per.clump <-
    bvl.fdr[match(bvl.cluster$target.min, tm.obs)]

### Chunk: unnamed-chunk-10

ntab <- xtabs(~trial,as.data.frame(br.vs.loc))
ntab

### Chunk: unnamed-chunk-11

lenti.favor <-
    sign(ntab[1]*bvl.cluster$value2>ntab[2]*bvl.cluster$value1)
lentab <- table(lenti.favor)

### Chunk: braun-vs-local-list

bgtab <-
    cbind(
        as.data.frame(
            bvl.cluster)[,c(1,2,3,4,6,7)],
        FDR= round(bvl.fdr.per.clump,4))
bgtab$value1 <- as.integer(bgtab$value1)
bgtab$value2 <- as.integer(bgtab$value2)
colnames(bgtab)[c(1,5,6)] <- c( "chromo", "Braun", "WAS-Lenti")
tab <- xtable(bgtab)
caption(tab) <-
    paste("Clumps Discovered Comparing Post-Infusion to ",
          "WAS $\\gamma$-Retroviral trial of Braun et al.")

bad.genes.gr <-
    GRanges(seqnames=c("chr11","chr12","chr3"),
            IRanges(start =
                        c(33560205, 3848551, 170336316),
                    end =
                        c(33935491, 4270056, 171166914)
                    ),strand=rep("*",3),
            gene=c("LMO2","CCND2","MECOM/EVI1"))
```

```
addrow <-
    list( pos=as.list( which( bvl.cluster %over% bad.genes.gr) - 1),
                command=rep("\\rowcolor{yellow} ",3))
print(tab,
      digits=4,
      add.to.row=addrow,
      tabular.environment="longtable",
      floating=FALSE,
      size="scriptsize"
      )

### Chunk: sessioninf

toLatex(sessionInfo())

### Chunk: print-all-code

codelines <-
    knitr:::knit_code$get()

cat("\\begin{knitrout}
\\definecolor{shadecolor}{rgb}{0.969, 0.969, 0.969}
\\color{fgcolor}
\\begin{kframe}
\\begin{alltt}\n")

## highr NAMESPACE is guaranteed by knitr

for (i in names(codelines)){
    cat("",
        highr:::hi_latex(paste0("### Chunk: ",i)),
        "",
        highr:::hi_latex(codelines[[i]]),
        sep="\n")
}

### show utils not echo-ed in any chunk
cat(highr:::hi_latex(
    "\n### Abundance Utils code read by sys.source:\n"),
    highr:::hi_latex(
        readLines("../../abundance/src/utils.R")),
    sep="\n")

cat("\n\\end{alltt}\n\\end{kframe}\n\\end{knitrout}\n")
```

```
### Abundance Utils code read by sys.source:
##' add together tables
##'
##' numeric names of table elements are used to combine
##' the tables --- summing counts of like labelled
##' cells and appending novel cells
##' @title reduce.tabs
##' @param x a table of counts with
##' \code{all(!is.na(as.numeric(names(x))))}
##' @param y a table like \code{x}
##' @return  table
##' @author Charles Berry
reduce.tabs <-
    function(x,y){
        inx <- intersect(names(x),names(y))
        onlyy <- setdiff(names(y),names(x))
        x[inx] <-x[inx] + y[inx]
        if (length(onlyy)){
            x[as.numeric(onlyy)] <- y[onlyy]
            names(x) <- 1:length(x)
            x[is.na(x)] <- 0
        }
        x}

##' dist class from lower.tri vector
##'
##' find dimensions, fill matrix, and coerce
##' @title as.DistVector
##' @param x
##' @return dist object
##' @author Charles Berry
as.distVector <- function(x)
    {
        lx <- length(x)
        ly <- (1+sqrt(1+8*lx))/2
        res <- array(0,c(ly,ly))
        res[lower.tri(res)] <- x
        ## not needed with as.dist wrapper
        ##    res <- res+t(res)
        as.dist(res)}

euc1 <- function(x) sqrt(rowSums(x^2))
euc2 <-
    function(x) pmin(12,sqrt((x^2) %*% c(1,4,64)))
```

```
minlen <-
    function(x) do.call(pmin,as.data.frame(x))

##' ##' weighted Manhattan distance
##'
##' \code{ID + length + 8*rep}
##' @title manhat
##' @param x matrix of distances
##' @return distance
##' @author Charles Berry
manhat <- function(x) x%*%c(1,1,8)


##' hierarchical clustering of distance
##'
##' combine various distnce metrics and do hclust
##' @title hcID
##' @param x a matrix of distances
##' @param ht height at which to cut
##' @param distfun how to combine distances to scalar
##' @return hclust object with cut component
##' categorizing the leaves after cutting
##' @author Charles Berry
hcID <-
    function(x,ht=5,distfun=manhat)
{
    dists <- as.distVector(distfun(x))
    hc <- hclust(dists)
    hc$cut <- cutree(hc,h=ht)
    hc
    }

##' Figure the cluster IDs for length and ID separately
##'
##'
##' @title hcEach
##' @param x result of distFuns
##' @param ht
##' @return
##' @author Charles Berry
hcEach <-
    function(x,ht=3){
        hclID <-
            hclust(as.distVector(x[,"ID"]))
```

```
        hclLength <-
            hclust(as.distVector(x[,"length"]))
        cutID <- cutree(hclID,h=ht)
        cutLength <- cutree(hclLength,h=ht)
        cbind(ID=cutID,length=cutLength)}

## hcEach(g3.dists[[1]])

##' inspect distance subsets
##'
##'
##' @title splitDist
##' @param xforest
##' @param xsplit
##' @param n
##' @param m
##' @return
##' @author Charles Berry
splitDist <-
    function(xforest,xsplit,n,m)
    {
     distsmry <-
         sapply(xforest,
                function(x) c(n=length(x$cut),
                              m=length(unique(x$cut))))
     dsFlag <- distsmry["n",] %in% n & distsmry["m",] %in% m
     index <- colnames(distsmry)[ dsFlag ]
     index.length <- sapply(xsplit[index],length)
     counts <- as.vector(unlist(xsplit[index])$count)
     cuts <- unlist(lapply(xforest[index],"[[","cut"),
                    use.names=FALSE)
     tab <-
         xtabs(counts~.,
               list(site=rep(index,index.length),count=cuts))
     df <-
         cbind(
             as.data.frame(
                 unlist(xsplit[index]),
                 row.names=NULL)[,-(3:5)],
             cut=cuts)
     list(table=tab,data.frame=df)
    }

##' summarize the distances by tabling them
```

```
##'
##'
##' @title distSmry
##' @param xdists list of distance matrices
##' @param IDFun  function to get distance factor
##' @param lenFun function to get distance factor
##' @param repFun function to get distance factor
##' @return list of lists of tables
##' @author Charles Berry
distSmry <-
    function(xdists,
             IDFun=function(x) factor(pmin(12,x),0:12),
             lenFun=function(x) factor(pmin(10,x),0:10),
             repFun=function(x) factor(x,0:4))
    lapply(xdists, function(x)
           list( ID=table(IDFun(x[,"ID"])),
                length=table(lenFun(x[,"length"])),
                 rep=table(repFun(x[,"rep"]))))

##' sum tables of distances
##'
##' use this for Reduce
##' @title distsmry.redFun
##' @param x
##' @param y
##' @return
##' @author Charles Berry
distsmry.redFun <-
    function(x,y) list(
        ID=x$ID+y$ID,
        length=x$length+y$length,
        rep=x$rep+y$rep)

##' widths of clumps of values
##'
##' Note that \code{clump} is the number of the clump
##' of contiguous sites. \code{clump==0} is assigned to
##' vacant positions
##' @title bleed.table
##' @param x integer vector
##' @return frequency table for clump widths
##' @author Charles Berry
bleed.table <-
    function(x){
```

```
        rv <- runValue(coverage(IRanges(start=x,width=1)))
        clump <- cumsum(rv==0)*(rv!=0)
        table(table(clump)[-1])}

distFuns <-
    function(x) cbind(length=dist(as.vector(x$length)),
                      ID=stringDist(x$FASTA),
                      rep=dist(as.vector(x$rep)))

##' do \code{cutree(hclust(x),h)} smartly when x is
##' long, but \code{unique(x)} is short
##'
##' In this case, i.e. x is a vector with no dim
##' attribute
##' \code{cutree(hclust(dist(x)),h)[!duplicated(x)]}
##' equals
##' \code{cutree(hclust(dist(x[!duplicated(x)])),h)}. The
##' function tries be more efficient than the brute
##' force clustering by working on a smaller \code{x}
##'
##' @title cutStart
##' @param x - numeric vector
##' @param ht h cutpoint for cutree
##' @return see \code{cutree}
##' @author Charles Berry
cutStart <-
    function(x,ht=5)
    {
        cutBig <-
            function(x,height) {
                x.uniq <- unique(x)
                len.uniq <- length(x.uniq)
                if (len.uniq==1)
                    return(rep(1L,length(x)))
                x.ind <- match(x,x.uniq)
                cutree(hclust(dist(x.uniq)),h=height)[x.ind]
            }
        lencat <- findInterval(length(x),c(1,2,100))
        switch(lencat,
               ## singleton
               1L,
               ## small tuple
               cutree(hclust(dist(x)),h=ht),
               ## larger tuple
```

```
                cutBig(x,ht)
                )
    }

##' smarter \code{cutree(hclust(stringDist(x)),h=5)}
##'
##' See the details for \code{cutStart}
##' @title cutID
##' @param x FASTA
##' @param ht h cutpoint for cutree
##' @return cluster IDs
##' @author Charles Berry
cutID <-
    function(x,ht=5)
    {
        cutBig <-
            function(x,height) {
                x.uniq <- unique(x)
                if (length(x.uniq)==1) return(1L)
                x.ind <- match(x,x.uniq)
                cutree(hclust(stringDist(x.uniq)),h=height)[x.ind]
            }
        lencat <- findInterval(length(x),c(1,2,100))
        switch(lencat,
               ## singleton
               1L,
               ## small tuple
               cutree(hclust(stringDist(x)),h=ht),
               ## larger tuple
               cutBig(x,ht)
               )
    }

##' first word of each element
##'
##' the leading nonblank characters are returned
##' @title first.word
##' @param x character vector
##' @return charcter vector
##' @author Charles Berry
first.word <-
    function(x) sub("[ ].*","",x)
```