

Tidy Analysis of Genomic Data

Michael Love

2022-05-11

Data organization depends on purpose

Table 1

	Genotype A			Genotype B		
	Rep 1	Rep 2	Rep 3	Rep 1	Rep 2	Rep 3
Drug 1	0.084	0.853	0.096	0.067	0.367	0.392
Drug 2	0.696	0.998	0.182	0.085	0.698	0.791
Drug 3	0.409	0.093	0.495	0.003	0.768	0.689
	Key:	Potential outlier				

“Tidy data” is organized for programming

One row per observation, one column per variable

```
head(dat)
```

```
## # A tibble: 6 x 5
```

```
##   drug  genotype    rep outlier  value
##   <fct> <chr>      <dbl> <lgl>    <dbl>
## 1 1      a          1 FALSE    0.954
## 2 1      a          2 FALSE    0.561
## 3 1      a          3 FALSE    0.367
## 4 2      a          1 FALSE    0.979
## 5 2      a          2 FALSE    0.0311
## 6 2      a          3 FALSE    0.950
```

The pipe

```
command | command | command > output.txt
```

“Pipes rank alongside the hierarchical file system and regular expressions as one of the most powerful yet elegant features of Unix-like operating systems.”

<http://www.linfo.org/pipe.html>

In R we use '%>%' instead of '|' to chain operations.

Verb-based operations

In the R package *dplyr*:

- ▶ `mutate()` adds new variables that are functions of existing variables.
- ▶ `select()` picks variables based on their names.
- ▶ `filter()` picks cases based on their values.
- ▶ `slice()` picks cases based on their position.
- ▶ `summarize()` reduces multiple values down to a single summary.
- ▶ `arrange()` changes the ordering of the rows.

<https://dplyr.tidyverse.org/>

Summarize after grouping

A useful paradigm is to *group* data and then *summarize*:

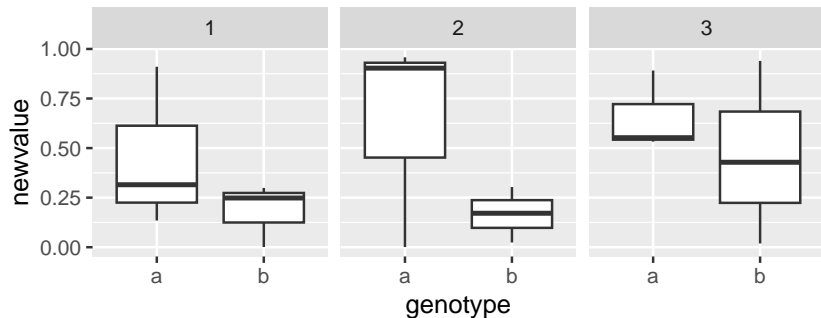
```
dat %>%  
  group_by(drug, genotype) %>%  
  filter(!outlier) %>%  
  summarize(mu_hat = mean(value))
```

Summarized output

```
## # A tibble: 6 x 3
## # Groups:   drug [3]
##   drug  genotype mu_est
##   <fct> <chr>      <dbl>
## 1 1      a          0.628
## 2 1      b          0.265
## 3 2      a          0.653
## 4 2      b          0.373
## 5 3      a          0.737
## 6 3      b          0.587
```

Piping into plots

```
dat %>%  
  mutate(newvalue = value^2) %>%  
  ggplot(aes(genotype, newvalue)) +  
  geom_boxplot() +  
  facet_wrap(~drug)
```



Lots of genomic range data is already tidy

chr1	100122271	100122495	Peak_75319	65	.	4.24709	6.53
chr1	100148962	100149149	Peak_47035	78	.	5.42118	7.87
chr1	10035625	10035783	Peak_83599	60	.	4.24908	6.01
chr1	10113652	10114012	Peak_22696	102	.	5.88792	10.2
chr1	10165234	10165473	Peak_61426	70	.	4.89948	7.04
chr1	10166426	10166654	Peak_52303	75	.	4.05875	7.56
chr1	10166709	10167142	Peak_101485	56	.	4.29447	5.62
chr1	10228978	10229286	Peak_56552	73	.	4.40606	7.37
chr1	10233774	10233984	Peak_54437	74	.	4.78393	7.43
chr1	10257595	10257832	Peak_144324	43	.	3.23111	4.35
chr1	10300983	10301435	Peak_55477	74	.	4.26907	7.41
chr1	10485619	10485897	Peak_128866	48	.	3.79116	4.85
chr1	10486926	10487197	Peak_64148	68	.	4.92835	6.83
chr1	105184501	105185026	Peak_98454	56	.	4.04794	5.69
chr1	105199317	105199602	Peak_117608	49	.	3.59369	4.96
chr1	105310436	105310779	Peak_23716	100	.	5.55389	10.0
chr1	105312808	105313002	Peak_104599	54	.	3.38229	5.46
chr1	105367824	105367998	Peak_12375	123	.	7.39252	12.3

Practical considerations

- ▶ Many comp students are already familiar with dplyr/ggplot2
- ▶ Can help to reduce hard-to-read variable names, e.g.:

```
dat3 <- dat2[dat2$signal > x]
```

- ▶ It's not necessarily *less code*, but aims for readable code

Why consider “tidy analysis” paradigm for genomics?

- ▶ Encourages exploratory analysis, vs. “all-in-one” functions
- ▶ Encourages efficiency: less calls to underlying C code
- ▶ Generalizes from simple to complex cases

Bringing range data into R

```
library(plyranges)
pks <- read_narrowpeaks("ENCFF231UNV.bed.gz")
```

Bringing range data into R

pks

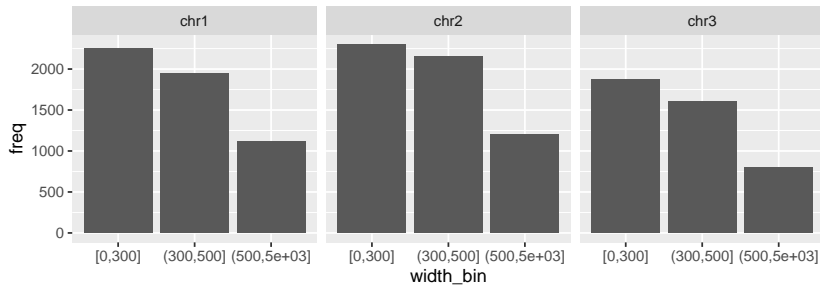
```
## GRanges object with 74284 ranges and 6 metadata columns:
```

```
##          seqnames          ranges strand |          name          score signalValue
##          <Rle>            <IRanges> <Rle> | <character> <numeric> <numeric>
##      [1]   chr1 100122272-100122495    * | Peak_75319          65      4.24709
##      [2]   chr1 100148963-100149149    * | Peak_47035          78      5.42118
##      [3]   chr1 10035626-10035783     * | Peak_83599          60      4.24908
##      [4]   chr1 10113653-10114012     * | Peak_22696         102      5.88792
##      [5]   chr1 10165235-10165473     * | Peak_61426          70      4.89948
##      ...     ...                     ...   ...     ...         ...         ...
## [74280]   chrX 99530000-99530373      * | Peak_24840         100      6.40685
## [74281]   chrX 99530681-99531004      * | Peak_84432          60      3.66625
## [74282]   chrX 99550895-99551287      * | Peak_56421          74      5.12695
## [74283]   chrX 99567509-99567986      * | Peak_91747          58      4.43250
## [74284]   chrY 1116052-1116527       * | Peak_30698          92      5.51992
##          pValue      qValue      peak
##          <numeric> <numeric> <integer>
##      [1]    6.53821    4.00852      126
##      [2]    7.87250    5.08650       90
##      [3]    6.01848    3.55308      134
##      [4]   10.26247    7.07929      138
##      [5]    7.04738    4.40822      107
##      ...     ...         ...         ...
## [74280]   10.04640    6.89683      242
## [74281]    6.01494    3.54969      231
## [74282]    7.40639    4.71603      272
## [74283]    5.84328    3.41019      399
## [74284]    9.29023    6.26238      118
## -----
## seqinfo: 22 sequences (1 circular) from mm10 genome
```

Example operations with plyranges

```
pks %>%  
  filter(seqnames %in% paste0("chr",1:3),  
         qValue > 2) %>%  
  mutate(width_bin = cut(width,  
                          breaks=c(0,300,500,5000),  
                          include.lowest=TRUE)) %>%  
  group_by(width_bin, seqnames) %>%  
  summarize(freq = n()) %>%  
  as_tibble() %>%  
  ggplot(aes(width_bin, freq)) +  
  geom_col() +  
  facet_wrap(~seqnames)
```

Example operations with plyranges



Making use of range information

Grouping by genomic overlaps (with maxgap and/or minoverlap)

Suppose a query set of ranges, gr:

```
gr %>%  
  group_by_overlaps(pks) %>%  
  summarize(mu=mean(score), sd=sd(score), n=n())
```

DataFrame with 3 rows and 4 columns

##	query	mu	sd	n
##	<integer>	<numeric>	<numeric>	<integer>
## 1	1	101.4521	42.9574	73
## 2	2	78.2500	27.9616	36
## 3	3	95.6818	49.8114	22

Picking out a representative

```
gr %>%  
  group_by_overlaps(pks) %>%  
  slice(which.max(score)) %>%  
  ungroup() %>%  
  select(score)
```

```
## GRanges object with 3 ranges and 1 metadata column:  
##           seqnames           ranges strand |           score  
##           <Rle>             <IRanges>  <Rle> | <numeric>  
##    [1]      chr1 510000001-520000000      * |           283  
##    [2]      chr1 520000001-530000000      * |           177  
##    [3]      chr1 530000001-540000000      * |           265  
##    -----  
##    seqinfo: 22 sequences (1 circular) from mm10 genome
```

Keeping the ranges of the representative

```
pks %>%  
  select(score) %>%  
  join_overlap_inner(gr) %>%  
  group_by(query_id) %>%  
  slice(which.max(score))
```

```
## GRanges object with 3 ranges and 2 metadata columns:
```

```
## Groups: query_id [3]
```

	seqnames	ranges	strand	score	query_id
	<Rle>	<IRanges>	<Rle>	<numeric>	<integer>
## [1]	chr1	51507255-51507557	*	283	1
## [2]	chr1	52253831-52254329	*	177	2
## [3]	chr1	53757564-53757891	*	265	3

```
## -----
```

```
## seqinfo: 22 sequences (1 circular) from mm10 genome
```

More complex cases

- ▶ The most common cases are computing summaries, overlaps
- ▶ More complex computations are possible, e.g.:
 - ▶ For peaks near genes, compute correlation of cell-type-specific accessibility and expression (Wancen Mu)
 - ▶ For regulatory variants falling in ATAC peaks, visualize their distribution stratified by SNP and peak categories (Jon Rosen)
 - ▶ For looped and unlooped enhancer-promoter pairs, compare average ATAC and RNA time series, while controlling for genomic distance and contact frequency (Eric Davis)

Going further: extracting info from fitted models

```
library(broom)
gr %>%
  group_by_overlaps(pks) %>%
  as_tibble() %>%
  select(query, score, qValue) %>%
  nest(data = -query) %>%
  mutate(fit = map(data, ~lm(score ~ qValue, data=.)),
         fitted = map(fit, ~.x$fitted)) %>%
  unnest(c(data, fitted))
```

Going further: extracting info from fitted models

```
## # A tibble: 131 x 5
##   query score qValue fit      fitted
##   <int> <dbl>  <dbl> <list> <dbl>
## 1     1     92   6.25 <lm>    91.9
## 2     1    135   9.85 <lm>   134.
## 3     1     68   4.22 <lm>    67.9
## 4     1     75   4.84 <lm>    75.2
## 5     1     43   2.23 <lm>    44.4
## 6     1     68   4.22 <lm>    67.9
## 7     1     98   6.77 <lm>   98.0
## 8     1    100   6.90 <lm>   99.5
## 9     1     36   1.70 <lm>    38.1
## 10    1     68   4.22 <lm>    67.9
## # ... with 121 more rows
```

Some pointers

- ▶ TSS: `anchor_5p() %>% mutate(width=1)`
- ▶ Flanking sequence: `flank_left` or `flank_upstream`
- ▶ Flatten/break up ranges: `reduce_ranges`, `disjoin_ranges`
- ▶ Concatenating ranges: `bind_ranges`
- ▶ Overlaps are handled often with “joins”: `join_overlap_*`, `join_nearest`, `join_nearest_downstream`, etc.

More tutorials online

- ▶ *plyranges* vignettes (on Bioc and GitHub)
- ▶ Enrichment of peaks and genes: “Fluent Genomics” workflow
- ▶ *nullranges* vignettes (on Bioc and GitHub), which provides block bootstrap and matching functionality that pairs easily with *plyranges*
- ▶ Other examples, incl. bootstrap: “Tidy Ranges Tutorial”
- ▶ BioC2022: Wancen Mu & Eric Davis *nullranges* workshop

Summary: tidy analysis for genomic data



- ▶ Encourages exploratory analysis, vs. “all-in-one” functions
- ▶ Encourages efficiency: less calls to underlying C code
- ▶ Generalizes from simple to complex cases

Reading

- ▶ Lee, S., Cook, D. & Lawrence, M. plyranges: a grammar of genomic data transformation. Genome Biology 20, 4 (2019).
<https://doi.org/10.1186/s13059-018-1597-8>
- ▶ Lee S, Lawrence M and Love MI. Fluent genomics with plyranges and tximeta. F1000Research 2020, 9:109
<https://doi.org/10.12688/f1000research.22259.1>
- ▶ plyranges vignettes <https://sa-lee.github.io/plyranges>
- ▶ Tidy Ranges Tutorial
<https://nullranges.github.io/tidy-ranges-tutorial>
- ▶ bootRanges, matchRanges, see:
<https://nullranges.github.io/nullranges>

Tidy analysis for matrix data:

- ▶ Mangiola, S., Molania, R., Dong, R. et al. tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology 22, 42 (2021). <https://doi.org/10.1186/s13059-020-02233-7>
- ▶ tidySummarizedExperiment, see:
<https://stemangiola.github.io/tidySummarizedExperiment>