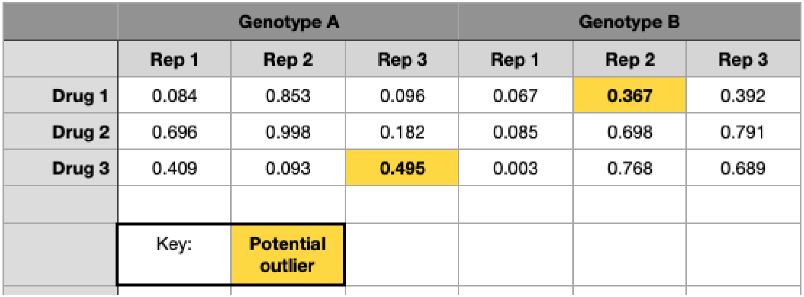
Tidy Analysis of Genomic Data

Michael Love
Dept of Genetics &
Dept of Biostatistics
UNC-Chapel Hill

UVA ~ October 2023

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

```
## # A tibble: 6 x 5
##   drug  genotype   rep outlier value
##   <fct> <chr>    <dbl> <lgl>   <dbl>
## 1 1     a            1 FALSE   0.267
## 2 1     a            2 FALSE   0.524
## 3 1     a            3 FALSE   0.974
## 4 2     a            1 FALSE   0.786
## 5 2     a            2 FALSE   0.283
## 6 2     a            3 FALSE   0.527
```

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 %>% or `|>` 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.
- `group_by()` perform any operation by group.

https://dplyr.tidyverse.org/

Summarize after grouping

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

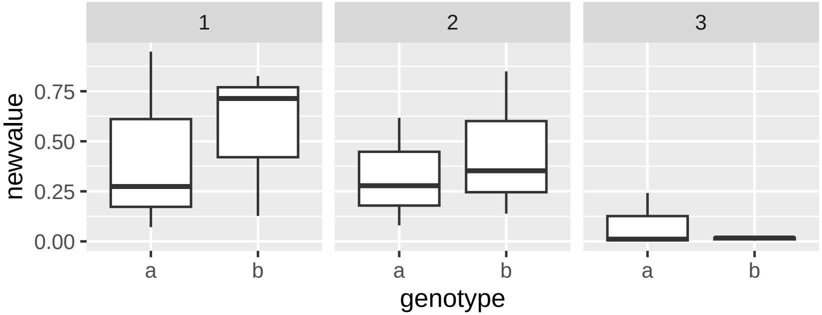
```r
dat %>%
  filter(!outlier) %>%
  group_by(drug, genotype) %>%
  summarize(mu_hat = mean(value))
```

Summarized output

```
## # A tibble: 6 x 3
## # Groups:   drug [3]
##   drug  genotype mu_est
##   <fct> <chr>     <dbl>
## 1 1     a         0.588
## 2 1     b         0.877
## 3 2     a         0.532
## 4 2     b         0.629
## 5 3     a         0.252
## 6 3     b         0.110
```

Piping directly into plots facilitates data exploration

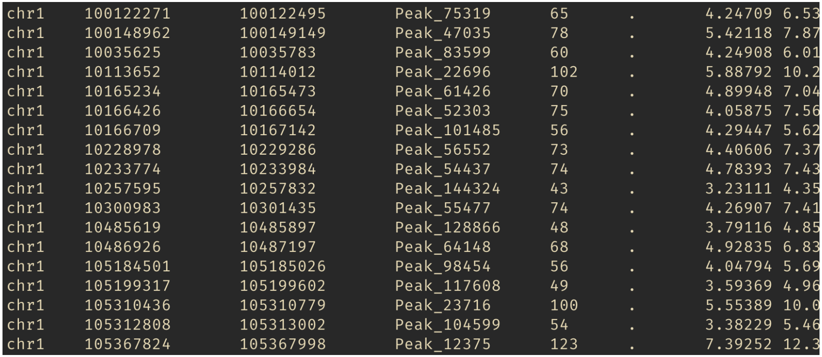
```r
dat %>%
  mutate(newvalue = value^2) %>%
  ggplot(aes(genotype, newvalue)) +
  geom_boxplot() +
  facet_wrap(~drug)
```



Summary I

- I teach both base R and “tidy”
- Both are wrappers, choose based on 1) efficiency 2) flow
- I use the former for writing software, latter for scripting
- Students know dplyr/ggplot2 already
- Next:
  - tidy for genomic ranges
  - tidy for matrix data (scRNA-seq)

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
```

Great packages in Bioconductor to work with ranges

- LOLA - facilitates testing overlaps, fast, useful databases
- COCOA - explore sample variation along genome
- regioneR - permutation testing
- ChIPpeakAnno - facilitates downstream analysis

Going to talk now about data exploration

Exploring data with tidy syntax

Helps avoid intermediate variables, and tucks away control code

```r
dat3 <- dat2[dat2$signal > 5]

# vs.

dat %>%
  filter(signal > 5)
```

This is *plyranges* from Stuart Lee, Michael Lawrence and Di Cook

Bringing range data into R

ENCODE mouse embryonic fibroblast, H3K4me1:

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

or equivalently:

```
pks <- read.csv("file.csv") %>%
  rename(seqnames = chr) %>%
  as_granges()
```

Another common paradigm, separating single column

```
pks <- read.delim("file.tsv") %>%
  tidyr::separate_wider_delim(
    location,
    delim=":|-", # e.g. chr1:123-456
    into=c("seqnames","start","end")
  ) %>%
  as_granges()
```

Ranges are rows, metadata are columns

```r
pks %>%
  slice(1:3) %>% # first 3 ranges
  select(signalValue) # just one metadata column
```

```
## GRanges object with 3 ranges and 1 metadata column:
##       seqnames              ranges strand | signalValue
##          <Rle>           <IRanges>  <Rle> |   <numeric>
##   [1]     chr1 100122272-100122495      * |     4.24709
##   [2]     chr1 100148963-100149149      * |     5.42118
##   [3]     chr1   10035626-10035783      * |     4.24908
##   -------
##   seqinfo: 22 sequences (1 circular) from mm10 genome
```

Example use of *plyranges*

- Suppose query ranges, `tiles` (e.g. ~1 Mb)
- Find all overlaps between `pks` and `tiles`
- Perform computation on the overlaps
- Many other choices in Bioc for enrichment (e.g. LOLA)

Example use of *plyranges*

Created with `tile_ranges` (see also `tileGenome`):

```
tiles
```

```
## GRanges object with 3 ranges and 1 metadata column:
##       seqnames            ranges strand |   tile_id
##          <Rle>         <IRanges>  <Rle> | <integer>
##   [1]     chr1 51000001-52000000      * |         1
##   [2]     chr1 52000001-53000000      * |         2
##   [3]     chr1 53000001-54000000      * |         3
##   -------
##   seqinfo: 22 sequences (1 circular) from mm10 genome
```

Consider genomic overlaps as a `join`

- We are joining two sources of information by match
- How would you then pick top scoring peak (`pks`) per `tile`?
- What verbs would be involved?

Consider overlaps as a `join`

```r
pks %>%
  select(score) %>% # just `score` column
  join_overlap_inner(tiles) %>% # overlap -> add cols from tiles
  group_by(tile_id) %>% # group matches by which tile
  slice(which.max(score)) # take the top scoring peak
```

```
## GRanges object with 3 ranges and 2 metadata columns:
## Groups: tile_id [3]
##       seqnames            ranges strand |     score   tile_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
```

Counting overlaps

- Use "." to specify self within a command
- Add number of overlaps to each entry in `tiles`:
- Can specify `maxgap` and/or `minoverlap`

```
tiles %>%
  mutate(n_overlaps = count_overlaps(., pks))
```

```
## GRanges object with 3 ranges and 2 metadata columns:
##       seqnames            ranges strand |   tile_id n_overlaps
##          <Rle>         <IRanges>  <Rle> | <integer>  <integer>
##   [1]     chr1 51000001-52000000      * |         1         73
##   [2]     chr1 52000001-53000000      * |         2         36
##   [3]     chr1 53000001-54000000      * |         3         22
##   -------
##   seqinfo: 22 sequences (1 circular) from mm10 genome
```

More complex cases

- For peaks near genes, compute correlation of cell-type-specific accessibility and expression (Wancen Mu) $\rightarrow$ similar to COCOA
- For regulatory variants falling in open chromatin peaks, visualize their distribution stratified by SNP and peak categories (Jon Rosen)
- For looped and un-looped enhancer-promoter pairs, compare average ATAC and RNA time series, while controlling for genomic distance and contact frequency (Eric Davis)

Nest → map → unnest

```r
library(purrr)
pks %>%
  join_overlap_inner(tiles) %>%
  as_tibble() %>%
  select(tile_id, score, qValue) %>%
  nest(data = -tile_id) %>%
  mutate(fit = map(data, ~lm(score ~ qValue, data=.)),
         fitted = map(fit, ~.x$fitted)) %>%
  unnest(c(data, fitted))
# see also broom::glance and broom::augment
```

Nest → map → unnest

```
## # A tibble: 131 x 5
##    tile_id 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
## # i 121 more rows
```

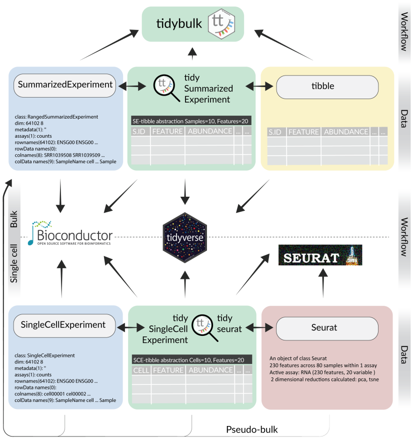
More *plyranges*-based tutorials online

- *plyranges* vignettes (on Bioc and GitHub)
- Enrichment of peaks and genes: “Fluent Genomics” workflow
- Null regions: *nullranges* vignettes (on Bioc and GitHub)
- Other examples, incl. bootstrap: “Tidy Ranges Tutorial”
- `#tidiness_in_bioc` and `#nullranges` Slack channels

Summary: tidy analysis for genomic range data

nullranges development sponsored by CZI EOSS

Tidy analysis of matrix data


tidybulk
Workflow
SummarizedExperiment
tidy Summarized Experiment
tibble
Data
SE-tibble abstraction Samples=10, Features=20
S.ID | FEATURE | ABUNDANCE | ... | ...
S.ID | FEATURE | ABUNDANCE | ... | ...
Bulk
Single cell
Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS
tidyverse
SEURAT
Workflow
SingleCellExperiment
tidy SingleCell Experiment
tidy seurat
Seurat
Data
SCE-tibble abstraction Cells=10, Features=20
CELL | FEATURE | ABUNDANCE | ... | ...
Pseudo-bulk

```
class: RangedSummarizedExperiment
dim: 64102 8
metadata(1): ''
assays(1): counts
rownames(64102): ENSG00 ENSG00 ...
rowData names(0):
colnames(8): SRR1039508 SRR1039509 ...
colData names(9): SampleName cell ... Sample
```

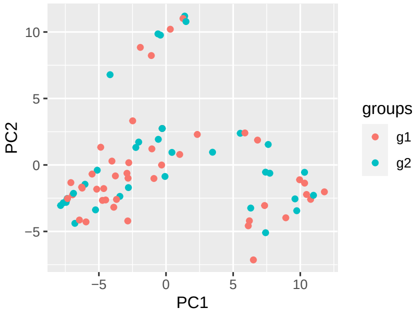
```
class: SingleCellExperiment
dim: 64102 8
metadata(1): ''
assays(1): counts
rownames(64102): ENSG00 ENSG00 ...
rowData names(0):
colnames(8): cell00001 cel00002 ...
colData names(9): SampleName cell ... Sample
```

```
An object of class Seurat
230 features across 80 samples within 1 assay
Active assay: RNA (230 features, 20 variable )
 2 dimensional reductions calculated: pca, tsne
```

tidy-* from Stefano Mangiola (WEHI) *et al.*

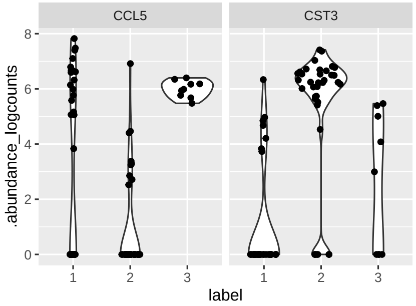
Example use of tidySingleCellExperiment

```
sce %>%
  scater::runPCA(ncomp=2, subset_row=var_genes) %>%
  ggplot(aes(PC1, PC2, color=groups)) +
  geom_point()
```



Example use of tidySingleCellExperiment

```
sce %>%
  join_features(c("CCL5","CST3")) %>%
  ggplot(aes(label, .abundance_logcounts)) +
  geom_violin() +
  geom_sina() +
  facet_wrap(~.feature)
```



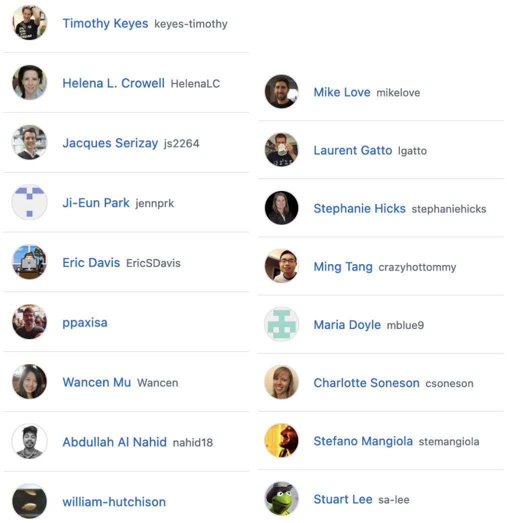
More complex cases

- Join extra cell-level data
- Perform nested analyses per cell population
- Create a custom expression signature from subset of genes
- Find genes near ChIP-seq peaks, convert to pseudobulk, plot

See our Bioc2023 workshop and tidyseurat / tidySCE

Altogether, “tidyomics”

https://github.com/tidyomics



Timothy Keyes keyes-timothy

Helena L. Crowell HelenaLC

Jacques Serizay js2264

Ji-Eun Park jennprk

Eric Davis EricSDavis

ppaxisa

Wancen Mu Wancen

Abdullah Al Nahid nahid18

william-hutchison

Mike Love mikelove

Laurent Gatto lgatto

Stephanie Hicks stephaniehicks

Ming Tang crazyhottommy

Maria Doyle mblue9

Charlotte Soneson csoneson

Stefano Mangiola stemangiola

Stuart Lee sa-lee

Reading

- Hutchison, WJ, Keyes, TJ, *et al.* The tidyomics ecosystem: Enhancing omic data analyses *bioRxiv* (2023) 10.1101/2023.09.10.557072
- Lee, S, Cook, D, Lawrence, M. plyranges: a grammar of genomic data transformation. *Genome Biology* (2019) 10.1186/s13059-018-1597-8
- Lee S, Lawrence M, Love MI. Fluent genomics with plyranges and tximeta. *F1000Research* (2020) 10.12688/f1000research.22259.1

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* (2021) 10.1186/s13059-020-02233-7
- tidySE, tidySCE, tidyseurat stemangiola.github.io/tidytranscriptomics

Extra slides

plyranges pointers

- TSS: `anchor_5p() %>% mutate(width=1)`
- Overlaps can specify `*_directed` or `*_within`
- Flatten/break up ranges: `reduce_ranges`, `disjoin_ranges`
- Concatenating ranges: `bind_ranges` with `.id` argument
- Overlaps are handled often with "joins": `join_overlap_*`, `join_nearest`, `join_nearest_downstream`, etc.
- Also `add_neareast_distance`
- Load *plyranges* last to avoid name masking with *AnnotationDbi* and *dplyr*