tidyomics: Enhancing Omic Data Analyses

Table of contents

What is `tidyomics`?

The `tidyverse` and Bioconductor ecosystems are transforming R-based data science and biological data analysis. **`tidyomics` bridges the gap between these ecosystems, enabling analysts to leverage the power of tidy data principles in omic analyses.**

This integration fosters cross-disciplinary collaborations, reduces barriers to entry for new users and enhances code readability, reproducibility and transparency. The tidy standard applied to biological software creates an extensible development ecosystem where independent researchers can interface with new software.

Ultimately, the `tidyomics` ecosystem, consisting of new and publicly available R packages, has the potential to greatly accelerate scientific discovery. The mission of this collaborative, worldwide project has been described in more detail in Nature Methods (2024):

> *Hutchison, William J., Timothy J. Keyes, Helena L. Crowell, Jacques Serizay, Charlotte Soneson, Eric S. Davis, Noriaki Sato, et al. 2024. "The tidyomics ecosystem: enhancing omic data analyses." Nat. Methods 21 (July): 1166–70. (https://doi.org/10.1038/s41592-024-02299-2).*

Core values

Our Code of Conduct is available here.

The `tidyomics` organization is open to new members and contributions; it is an effort of many developers in the Bioconductor community and beyond.

- See our tidyomics open challenges project to see what we are currently working on;
- Issues tagged with good first issue are those that developers think would be good for a new developer to start working on;
- Read over our Guidelines for contributing;
- As with new users, for new developers please consider joining our Slack Channel, #tidiness_in_bioc. Most of the tidyomics developers are active there and we are happy to talk through updates, PRs, or give guidance on your development of a new package in this space.

References

Preamble

`tidyomics` is an open project to develop and integrate software and documentation to enable a tidy data analysis framework for omics data objects. `tidyomics` enables the use of familiar `tidyverse` verbs (`select`, `filter`, `mutate`, etc.) to manipulate rich data objects in the Bioconductor ecosystem. Importantly, the data objects are not modified, but `tidyomics` provides a tidy interface to work on the native objects, leveraging existing Bioconductor classes and algorithms.

A key innovation in Bioconductor is the use of object-oriented programming and specific data structures.

As described in @Gentleman2004Sep,

> an `exprSet` is a data structure that binds together array-based expression measurements with covariate and administrative data for a collection of [experiments]… [its] design facilitates a three-tier architecture for providing analysis tools for new microarray platforms: low-level data are bridged to high-level analysis manipulations via the `exprSet` structure.

In Bioconductor, rich, structured data about experiments is maintained throughout analyses by passing data objects from one method to another. E.g. `estimateDispersions` adds dispersion information to the `rowData` slot of a `DESeqDataSet` which is a sub-class of a `SummarizedExperiment`, therefore inheriting the structure and methods of that class. The structure of the data is preserved after running the function (like many Biodonductor methods, it is an endomorphic function).

The goal of `tidyomics` is to preserve the object-oriented programming style and stucture of Bioconductor data objects, while allowing users to manipulate these data objects with expressive commands, familiar to `tidyverse` users.

`tidyomics` aims to allow users to flexibly explore and plot biological datasets, by combining simple functions with human-readable names in a modular fashion to perform complex operations, including grouping and summarization tasks. Operations should still be performed with comparable efficiency to the underlying base R/Bioconductor code. asdcasdc

References

Part I

Fundamental omics data

1 Genomic intervals data

1.1 Importing GRanges from files

Recommended time: 10 min

The `BiocIO::import()` generic function lets one import `bed` files (or alike) into `GRanges` object in R.

```
library(GenomicRanges)

library(rtracklayer)

bedf <- system.file('extdata', 'S288C-borders.bed', package = 'Bioc2024tidyWorkshop', mustWo

import(bedf)
```

The tidy way of importing genomic ranges in R, however, is to read files as a `data.frame` (or `tibble`) and *coerce* it as a `GRanges` once it's imported in memory.

```
library(tidyverse)

tib <- read_tsv(bedf, col_names = FALSE)

tib

library(plyranges)

gr <- as_granges(tib, seqnames = X1, start = X2, end = X3)

gr
```

Note how refering to column names is done using tidy evaluation.

1.2 Manipulating `GRanges` with tidy verbs

Recommended time: 10 min

With `plyranges`, a number of tidy operations are readily available, e.g.:

- `mutate()`
- `select()`
- `filter()`
- `group_by()` + `summarize()`
- …

Just like with `tidyverse` functions, these operations work well with the native `|>` pipe in R.

```
gr |>
    mutate(score = runif(n())) |>
    filter(score > 0.2) |>
    mutate(round_score = round(score, digits = 1)) |>
    group_by(round_score) |>
    summarize(mean = mean(score))
```

But really, what `plyranges` excels at is providing direct access to “protected” variables, i.e. `seqnames`, `start`, `end`, `width`, `strand`, …

```
gr |>
    mutate(
        seqnames = factor('XVI', levels(seqnames)),
        width = 1,
        strand = rep(c('-', '+'), n()/2)
    )
```

Finally, a number of operations for genomic arithmetics are provided by `plyranges`:

```
gr |>
    anchor_center() |>
    stretch(extend = -1000) |>
    shift_upstream(250) |>
    flank_upstream(100)
```

Resources

- “Tidy Ranges Tutorial” by Michael Love
- A Bioc2024 workshop on plyranges and others

Session info

Click to expand

References

2 Genomic interactions data

2.1 What are `GInteractions`?

2.1.1 Creating a `GInteractions` object from scratch

```r
library(InteractionSet)

gr1 <- GRanges("I:10-50")

gr2 <- GRanges("I:100-110")

GInteractions(anchor1 = gr1, anchor2 = gr2)
```

```r
GInteractions(anchor1 = c(1, 2, 3), anchor2 = c(1, 4, 5), regions = gr)
```

2.1.2 Importing genomic interaction data from files

```
bedpef <- system.file('extdata', 'S288C-loops.bedpe', package = 'Bioc2024tidyWorkshop', mustW

tib <- read_tsv(bedpef, col_names = FALSE)

tib

library(plyinteractions)

gi <- tib |>
    as_ginteractions(
        seqnames1 = X1, start1 = X2, end1 = X3,
        seqnames2 = X4, start2 = X5, end2 = X6
    )

gi
```

2.2 Manipulating `GInteractions` the tidy way

2.2.1 Moving anchors around

```
gi |>
    mutate(
        seqnames1 = factor('XVI', levels(seqnames1)),
        strand1 = '+',
        start2 = end1,
        width2 = width1 + 100,
        score = runif(length(gi)),
        is_cis = ifelse(seqnames1 == seqnames2, TRUE, FALSE)
    )
```

2.2.2 Filtering interactions

```r
gi |> filter(seqnames1 == 'I')

gi |> filter(seqnames2 == 'I')

gi |>
    mutate(score = runif(length(gi))) |>
    filter(seqnames2 == 'I', score > 0.2)
```

2.2.3 Overlapping anchors

```r
centros <- system.file('extdata', 'col', package = 'Bioc2024tidyWorkshop', mustWork = TRUE)
    read_tsv() |>
    as_granges(seqnames = seqID) |>
    anchor_center() |>
    stretch(20000)

gi |>
    join_overlap_left(centros) |>
    filter(!is.na(patternName))
```

```r
gi |>
    pin_anchors1() |>
    join_overlap_left(centros) |>
    filter(!is.na(patternName))

gi |>
    pin_anchors2() |>
    join_overlap_left(centros) |>
    filter(!is.na(patternName))
```

2.3 Real-world use case: computing a P(s)

2.3.1 Importing data from pairs file

```
pairsf <- system.file('extdata', 'mESCs.pairs.gz', package = 'Bioc2024tidyWorkshop', mustWor

pairs <- read_tsv(pairsf, col_names = FALSE, comment = "#") |>
    set_names(c(
        "ID", "seqnames1", "start1", "seqnames2", "start2", "strand1", "strand2"
    )) |>
    as_ginteractions(end1 = start1, end2 = start2, keep.extra.columns = TRUE)
```

2.3.2 Counting interactions by strands

```
df <- pairs |>
    add_pairdist() |>
    filter(pairdist < 2000) |>
    group_by(strand1, strand2, pairdist) |>
    count()

ggplot(df, aes(x = pairdist, y = n, col = interaction(strand1, strand2))) +
    geom_smooth() +
    scale_y_log10()
```

2.3.3 Plot P(s)

```
x <- 1.1^(1:200-1)
lmc <- coef(lm(c(1,1161443398)~c(x[1], x[200])))
bins_breaks <- unique(round(lmc[2]*x + lmc[1]))
bins_widths <- lead(bins_breaks) - bins_breaks

# Bin distances
df <- pairs |>
```

```r
    add_pairdist(colname = 's') |>
    mutate(
        binned_s = bins_breaks[as.numeric(cut(s, bins_breaks))],
        bin_width = bins_widths[as.numeric(cut(s, bins_breaks))]
    ) |>
    group_by(binned_s, bin_width) |>
    count(name = "n") |>
    as_tibble() |>
    mutate(Ps = n / sum(n) / bin_width)

ggplot(df, aes(x = binned_s, y = Ps)) +
    geom_line() +
    scale_y_log10() +
    scale_x_log10() +
    annotation_logticks() +
    labs(x = "Genomic distance", y = "Contact frequency") +
    theme_bw()
```

Resources

Session info

Click to expand

References

3 Summarized experiment data

Resources

Session info

Click to expand

References

Part II

Complex omics data

4 Transcriptomic data

Resources

- tidybulk vignette
- Fluent genomics workflow

Session info

Click to expand

References

5 Epigenomic data

5.1 Introduction to `tidyCoverage`

5.1.1 `CoverageExperiment` and `AggregatedCoverage` class

```r
library(tidyCoverage)

data(ce)

data(ac)

ce

ac
```

5.1.2 Creating a `CoverageExperiment` object from tracks and features

```r
tracks <- BigWigFileList(c(
    mnase = system.file("extdata", "MNase.bw", package = "tidyCoverage"),
    cohesin = system.file("extdata", "Scc1.bw", package = "tidyCoverage")
))
features <- GRangesList(
    TSSs = system.file("extdata", "TSSs.bed", package = "tidyCoverage") |> import() |> sampl
    TTSs = system.file("extdata", "TTSs.bed", package = "Bioc2024tidyWorkshop") |> import()
)
```

```r
ce2 <- CoverageExperiment(
    tracks = tracks,
    features = features,
    width = 2000,
    ignore.strand = FALSE
)

ce2
```

```r
colData(ce2)

rowData(ce2)

assay(ce2, 'coverage')

class(assay(ce2, 'coverage')['TSSs', 'mnase'])

class(assay(ce2, 'coverage')[['TSSs', 'mnase']])

dim(assay(ce2, 'coverage')[['TSSs', 'mnase']])
```

5.1.3 Tidy coverage? That's right!

```r
library(tidySummarizedExperiment)

ce2

ce2 |> filter(features == 'TSSs')

ce2 |> slice(2)

ce2 |> select(features, n)
```

5.1.4 `expand()` or `aggregate()`

```r
tib <- expand(ce2)

tib
```

```r
ac2 <- aggregate(ce2)

ac2
```

5.1.5 Visualizing aggregated coverage

-
-

```r
CoverageExperiment(tracks, GRanges("II:1-100000"), window = 100) |>
    expand() |>
    ggplot() +
    geom_coverage() +
    facet_grid(track ~ features, scales = "free") +
    labs(x = 'chrV', y = 'Signal coverage')

ggplot(ac2) +
    geom_aggrcoverage() +
    facet_grid(track ~ features, scales = "free") +
    labs(x = 'Distance from genomic features', y = 'Signal coverage')
```

5.2 Real-world use case: studying epigenomic landscape of reulatory elements

5.2.1 Fetch coverage data from ENCODE

```
library(AnnotationHub)
ah <- AnnotationHub()
ids <- c('AH32207', 'AH35187')
names(ids) <- c('DNAse', 'H3K4me3')
bws <- lapply(ids, function(.x) ah[[.x]] |> resource()) |> BigWigFileList()
names(bws) <- names(ids)
```

5.2.2 Plotting coverage data over several loci

```
ce3 <- CoverageExperiment(
    bws,
    list(
        ccno = GRanges("chr5:55220001-55235000"),
        mcidas = GRanges("chr5:55159001-55174000")
    ),
    window = 50
)
expand(ce3) |>
    mutate(coverage = scales::oob_squish(coverage, c(0, 10))) |>
    ggplot() +
    geom_coverage(aes(fill = track), unit = 'Mb') +
    facet_grid(track~features, scales = 'free')
```

5.2.3 Import DNase peaks from ENCODE

```
features <- list(DNase = ah[['AH30077']] |> filter(zScore > 100) |> sample(1000))
```

5.2.4 Generating coverage aggregates and heatmaps over DNAse peak

```r
ce4 <- CoverageExperiment(bws, features, width = 2000, window = 10)
```

```r
aggregate(ce4) |>
    ggplot(aes(x = coord, y = mean)) +
    geom_aggrcoverage(aes(col = track)) +
    facet_wrap(~track) +
    labs(x = 'Distance from DNAse peak', y = 'Signal')
```

Resources

Session info

Click to expand

References

6 Single-cell data

Resources

- tidySingleCellExperiment vignette
- A Bioc2023 workshop on tidySingleCellExperiment

Session info

Click to expand

References

7 Spatial single-cell data

Resources

Session info

Click to expand

References

8 Flow cytometry data

Resources

Session info

Click to expand

References

9 Mass cytometry data

Resources

Session info

Click to expand

References

A Talks and workshops

A.1 Workflows

- Tidy single-cell analyses
- Investigating chromatin composition and architecture
- Tidy ranges tutorial
- Tidy intro talk
- T1D GWAS SNPs and CD4+ peaks
- Fluent genomics workflow

A.2 Talks

- Tidy enrichment analysis with plyranges and nullranges
- Tidy analysis of genomic data

our repository

A.3 Related projects

- biobroom

B List of packages included in the `tidyomics` framework

B.1 Core packages

tidySummarizedExperiment	Vignette	GitHub
tidySingleCellExperiment	Vignette	GitHub
tidySeurat	Vignette	GitHub
tidySpatialExperiment	Vignette	GitHub
tidytof		GitHub
plyranges	Vignette	GitHub
plyinteractions	Vignette	GitHub
tidybulk	Vignette	GitHub

B.2 Helper packages

nullranges	Vignette	GitHub
easylift	Vignette	GitHub
tidygate	Vignette	GitHub

C tidyomics contributors