BioC 2016 Developer Day

Core team updates

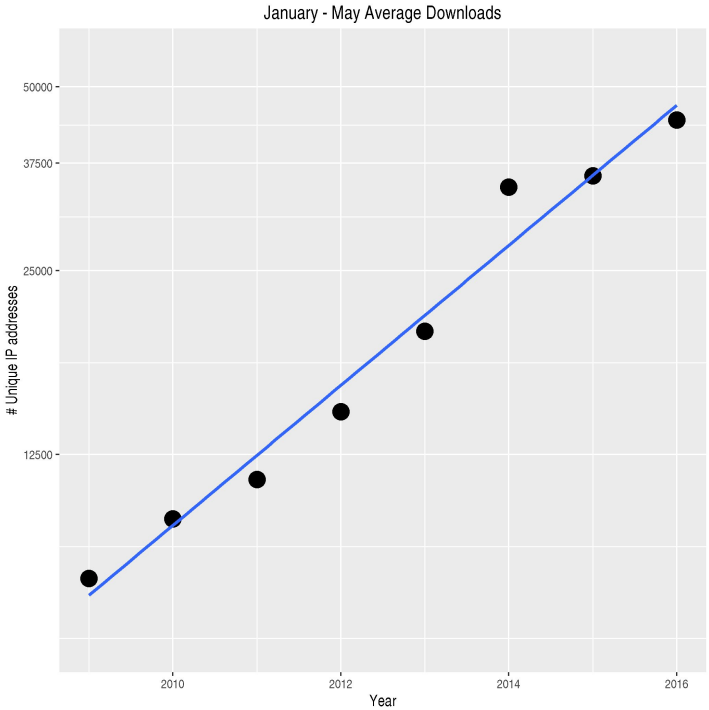
Welcome and Project Update

Thanks!

- Pam Jarrett, Ellen Sanders Noonan, Ellen Van Stone
- Susan Holmes, Sean Davis
- Speakers and Workshop presenters
- Bioc developers!

The project since last year

- 2 Releases with 187 new packages
- Lots of activity on the support site
- Steadily growing user base
- Move to Roswell Park



Activities and opportunities

Core team activities

- *GenomicRanges* infrastructure
- *AnntotationHub* and *ExperimentHub*
- *BiocParallel / GenomicFiles*
- Progress on *MultiAssayExperiment*
- On-disk / lazy evaluation of large data
- Public new package submissions
- User and developer support

Keeping up with the burgeoning R community

- Package development best practices
- Approaches to version control and testing

Increasingly cloud-based computing

- Efficient access to cloud-based resources
- Participation in cloud-based bioinformatics initiatives
- Computation in the cloud

Career opportunities!

- Senior Programmer / Analyst -- creative web / system administration / development -- https://goo.gl/2s26pp

Acknowledgements

Core team (**current** & recent): **Valerie** Obenchain, **Hervé** Pagès, **Dan** Tenenbaum, **Lori** Shepherd, **Marcel** Ramos, Jim Hester, Jim Java, Brian Long, Sonali Arora, Nate Hayden, Paul Shannon, Marc Carlson

Technical advisory board: Vincent Carey, Wolfgang Huber, Robert Gentleman, Rafael Irizzary, Levi Waldron, Michael Lawrence, Sean Davis, Aedin Culhane

Scientific advisory board: Simon Tavare (CRUK), Paul Flicek (EMBL/EBI), Simon Urbanek (AT&T), Vincent Carey (Brigham & Women's), Wolfgang Huber (EBI), Rafael Irizzary (Dana Farber), Robert Gentleman (23andMe)

Research reported in this presentation was supported by the National Human Genome Research Institute and the National Cancer Institute of the National Institutes of Health under award numbers U41HG004059 and U24CA180996. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

Lori Shepherd

GenomicFiles / VcfStack / RangedVcfStack

disjoin() in IRanges / GenomicRanges

GenomicFiles

VcfStack / RangedVcfStack

VcfStack / RangedVcfStack

The VcfStack class is a vector of related VCF files, for instance each file representing a separate chromosome. The class helps manage these files as a group.

The RangedVcfStack class extends VcfStack by associating genomic ranges of interest to the collection of VCF files.

VcfStack / RangedVcfStack

```
VcfStack(files=NULL, seqinfo=NULL, colData=NULL)
```

files: A character vector of files paths pointing to VCF files. The character vector must be named, with names correspond to seqnames in each VCF file.

seqinfo: A Seqinfo object describing the levels genome and circularity of each sequence.

colData: An optional DataFrame describing each sample in the VcfStack. When present, row names must correspond to sample names in the VCF file.

```
RangedVcfStack(vs=NULL, rowRanges=NULL)
```

vs: A VcfStack object.

rowRanges: An optional GRanges object associating the genomic ranges of interest to the collection of VCF files. The seqnames of rowRanges are a subset of seqnames(vs). If missing, a default is created from the seqinfo object of the provided VcfStack

VcfStack / RangedVcfStack Accessors

- dim(x)
- dimnames(x)
- rownames(x)
- colnames(x)

As well as your typical getters and setters for object attributes:

- files(x)
- seqinfo(x)
- colData(x)
- rowRanges(x)

VcfStack / RangedVcfStack Methods

assay(x, i, …)

Get matrix of genotype calls from VCF files

readVcfStack(x, i, j=colnames(x))

Get content of VCF files in the VcfStack

show(x)

Display abbreviated information about VcfStack / RangedVcfStack

i: indicated which files to read
is a GRanges object, character() vector of seqnames, numeric() vector, logical() vector, or can be missing. For a RangedVcfStack object, assay and readVcfStack will use the associated rowRanges object for i.

j: indicates which samples to read
can be missing or a character() vector of sample names

VcfStack / RangedVcfStack Subsetting

x[i, j]

Get elements from ranges i and samples j as a VcfStack or RangedVcfStack object

x: is a VcfStack or RangedVcfStack object

i: indicated which files to subset

can be missing, a character() vector of seqnames, numeric() vector of indexes, or logical() vector. When x is a VcfStack instance, i can also be a GRanges object; seqnames(i) is then used to subset the files in the VcfStack.

j: indicated which samples to subset.

can be missing, a character() vector of sample names, a numeric() vector, or logical() vector.

IRanges / GenomicRanges

disjoin()

IRanges / GenomicRanges

disjoin(x, with.revmap=FALSE)

- Ranges
- RangesList
- CompressedIRangesList

disjoin(x, with.revmap=FALSE, ignore.strand=FALSE)

- GenomicRanges
- GRangesList

with.revmap
TRUE or FALSE. Should the mapping from output to input ranges be stored in the returned object? If yes, then it is stored as metadata column revmap of type IntegerList

GenomicRanges ‘GRanges’ Example

```
> gr <- GRanges(Rle(c("chr1", "chr3"), c(2, 2)),
                IRanges(c(8,6,8,6),c(11,15,11,15), names=c("k","l","m","n")),
                Rle(strand(c("-", "-","+","*"))),
                score=11:14, GC=c(.2,.3,.3,.1))
```

> gr

```
GRanges object with 4 ranges and 2 metadata columns:
    seqnames    ranges strand |     score        GC
       <Rle> <IRanges>  <Rle> | <integer> <numeric>
  k     chr1   [8, 11]      - |        11       0.2
  l     chr1   [6, 15]      - |        12       0.3
  m     chr3   [8, 11]      + |        13       0.3
  n     chr3   [6, 15]      * |        14       0.1
  -------
  seqinfo: 2 sequences from an unspecified genome; no
seqlengths
```

> dgr <- disjoin(gr, with.revmap=TRUE)
> dgr

```
GRanges object with 5 ranges and 1 metadata column:
      seqnames    ranges strand |        revmap
         <Rle> <IRanges>  <Rle> | <IntegerList>
  [1]     chr1  [ 6,  7]      - |             2
  [2]     chr1  [ 8, 11]      - |           1,2
  [3]     chr1  [12, 15]      - |             2
  [4]     chr3  [ 8, 11]      + |             3
  [5]     chr3  [ 6, 15]      * |             4
  -------
  seqinfo: 2 sequences from an unspecified genome; no seqlengths
```

To Get Original Metadata Values:

```
> revmap <- mcols(dgr)$revmap
> score <- extractList(mcols(gr)$score, revmap)
> GC <-  extractList(mcols(gr)$GC, revmap)
> mcols(dgr)$score <- score
> mcols(dgr)$GC <- GC
> dgr
```

```
GRanges object with 5 ranges and 3 metadata columns:
      seqnames    ranges strand |        revmap         score             GC
         <Rle> <IRanges>  <Rle> | <IntegerList> <IntegerList> <NumericList>
  [1]     chr1  [ 6,  7]      - |             2            12           0.3
  [2]     chr1  [ 8, 11]      - |           1,2         11,12       0.2,0.3
  [3]     chr1  [12, 15]      - |             2            12           0.3
  [4]     chr3  [ 8, 11]      + |             3            13           0.3
  [5]     chr3  [ 6, 15]      * |             4            14           0.1
  -------
  seqinfo: 2 sequences from an unspecified genome; no seqlengths
```

GenomicRanges ‘GRangesList’ Example

```
gr <- GRanges(Rle(c("chr1", "chr3"), c(2, 2)),
              IRanges(c(8,6,8,6),c(11,15,11,15), names=c("k","l","m","n")),
              Rle(strand(c("-", "-","+","*"))),
              score=11:14, GC=c(.2,.3,.3,.1))
grl <- GRangesList(gr, gr)
```

> grl

```
GRangesList object of length 2:
[[1]]
GRanges object with 4 ranges and 2 metadata columns:
    seqnames    ranges strand |     score        GC
       <Rle> <IRanges>  <Rle> | <integer> <numeric>
  k     chr1   [8, 11]      - |        11       0.2
  l     chr1   [6, 15]      - |        12       0.3
  m     chr3   [8, 11]      + |        13       0.3
  n     chr3   [6, 15]      * |        14       0.1

[[2]]
GRanges object with 4 ranges and 2 metadata columns:
    seqnames  ranges strand | score  GC
  k     chr1 [8, 11]      - |    11 0.2
  l     chr1 [6, 15]      - |    12 0.3
  m     chr3 [8, 11]      + |    13 0.3
  n     chr3 [6, 15]      * |    14 0.1

-------
seqinfo: 2 sequences from an unspecified genome; no seqlengths
```

> disjoin(grl, with.revmap=TRUE)

```
GRangesList object of length 2:
[[1]]
GRanges object with 5 ranges and 1 metadata column:
      seqnames    ranges strand |        revmap
         <Rle> <IRanges>  <Rle> | <IntegerList>
  [1]     chr1  [ 6,  7]      - |             2
  [2]     chr1  [ 8, 11]      - |           1,2
  [3]     chr1  [12, 15]      - |             2
  [4]     chr3  [ 8, 11]      + |             3
  [5]     chr3  [ 6, 15]      * |             4

[[2]]
GRanges object with 5 ranges and 1 metadata column:
      seqnames   ranges strand | revmap
  [1]     chr1 [ 6,  7]      - |      2
  [2]     chr1 [ 8, 11]      - |    1,2
  [3]     chr1 [12, 15]      - |      2
  [4]     chr3 [ 8, 11]      + |      3
  [5]     chr3 [ 6, 15]      * |      4

-------
seqinfo: 2 sequences from an unspecified genome; no seqlengths
```

Valerie Obenchain

ExperimentHub

ExperimentHub

Resource to house curated data from experiments, publications or courses

Similar interface as AnnotationHub except ...

- Parent package documentation
- List resources by package
- Interface with the data through the package or ExperimentHub
- All data stored in AWS S3; no web downloads

ExperimentHub: parent package documentation

```
> library(ExperimentHub)

> eh = ExperimentHub()
snapshotDate(): 2016-06-08

> eset = eh[[100]]
see ?curatedMetagenomicData and browseVignettes('curatedMetagenomicData') for documentation
downloading from ‘https://experimenthub.bioconductor.org/fetch/100’
retrieving 1 resource
  |======================================================================| 100%

> ?curatedMetagenomicData
```

ExperimentHub: list resources by package

```
> head(package(eh), 3)
        EH1                 EH2                         EH3
  "GSE62944" "curatedMetagenomicData" "curatedMetagenomicData"


> table(package(eh))
  curatedMetagenomicData              GSE62944
                     162                     1
```

ExperimentHub: interface with data via package

```
> eh["EH100"]
ExperimentHub with 1 record
# snapshotDate(): 2016-06-08
# package(): curatedMetagenomicData
# $dataprovider: Human Microbiome Project Consortium
# $species: Homo sapiens
# $title: hmp.r_retroauricular_crease.marker_ab.eset.rda
...

> ?hmp.r_retroauricular_crease.marker_ab.eset  ## package man page
> hmp.r_retroauricular_crease.marker_ab.eset()  ## loads the data
> hmp.r_retroauricular_crease.marker_ab.eset(metadata = TRUE)  ## loads the metadata
```

ExperimentHubData

Information on adding resources to ExperimentHub is found in the ExperimentHubData vignette.

Marcel Ramos

MultiAssayExperiment

MultiAssayExperiment

A package to manage multiple assays on sets of samples or specimens

- A container class for handling overlapping sets of samples
- User-friendly operations (subsetting)
- Mapping scheme for relating samples to participants or experiment results to specimen data
- Set up for common genomic computations across diverse assays
- On-disk representation of data (moving to lazy eval with `HDF5Array`)

Hierarchy of information:

Study
- Experiment
  - Biological Unit

Datasets will soon be accessible via **ExperimentHub**

MultiAssayExperiment: Structure Overview

- **MultiAssayExperiment** class
  - **Elist** class and slot - ***workhorse container***
    - Any class that has a `[` bracket method, `colnames`, `rownames` and `dim`.
      - *RangedRaggedAssay*
      - *SummarizedExperiment*, *RangedSummarizedExperiment*
      - *ExpressionSet*
      - *matrix*
  - **pData** (of class *DataFrame*) - ***specimen description***
    - Each row is a patient or specimen
    - Includes demographics and/or other specimen-wide variables
  - **sampleMap** (of class *DataFrame*) - ***mapping scheme***
    - Maps sample identifiers to participants/specimen in a table
  - **metadata** (*ANY* class)
    - Include additional study level information

MultiAssayExperiment: Quick Example

> library(MultiAssayExperiment)

> example(“MultiAssayExperiment”)

> myMultiAssayExperiment

```
A "MultiAssayExperiment" object of 3 listed
 experiments with user-defined names and respective classes.
 Containing an "Elist" class object of length 3:
 [1] Affy: "ExpressionSet" - 2 rows, 4 columns
 [2] Methyl450k: "matrix" - 2 rows, 5 columns
 [3] CNVgistic: "RangedRaggedAssay" - 5 rows, 3 columns
To access slots use:
 Elist() - to obtain the "Elist" of experiment instances
 pData() - for the primary/phenotype "DataFrame"
 sampleMap() - for the sample availability "DataFrame"
 metadata() - for the metadata object of "ANY" class
See also: subsetByAssay(), subsetByRow(), subsetByColumn()
```

MultiAssayExperiment: Thorough Example

An in-depth example on how to build your own **MultiAssayExperiment** can be found in the package vignette

Hervé Pagès

Recent developments:

- GPos class
- HDF5Array, DelayedArray

What’s next?

GPos

A very light GRanges-like container for storing a set of *positions* along the genome.

Particularly memory-efficient when the object contains long runs of adjacent positions.

Can be put on a SummarizedExperiment object (as rowRanges).

```
> gpos
GPos object with 12162995 positions and 0 metadata columns:
             seqnames       pos strand
                <Rle> <integer>  <Rle>
         [1]     chrI         1      *
         [2]     chrI         2      *
         [3]     chrI         3      *
         ...      ...       ...    …
  [12162993]  2micron      6316      *
  [12162994]  2micron      6317      *
  [12162995]  2micron      6318      *
  -------
  seqinfo: 18 sequences (2 circular) from sacCer2 genome
```

All the single positions along the Yeast genome are represented.

```
> object.size(gpos)
14000 bytes
```

GPos

Metadata columns need to be light too.

Good candidates:

- ➔ Rle (e.g. coverage)
- ➔ DNAString
- ➔ sparse object (e.g. Matrix)
- ➔ on-disk object (e.g. HDF5Array)
- ➔ ?

Current limitation: length of a GPos object cannot exceed 2^31 (2 billions).

See ?GPos in the GenomicRanges package for more information.

```
> gpos
GPos object with 12162995 positions and 2 metadata columns:
             seqnames       pos strand |    cov          dna
                <Rle> <integer>  <Rle> | <Rle> <DNAString>
         [1]     chrI         1      * |     0           C
         [2]     chrI         2      * |     0           C
         [3]     chrI         3      * |     0           A
         [4]     chrI         4      * |     0           C
         [5]     chrI         5      * |     0           A
         ...      ...       ...    ... .   ...         …
  [12162991]  2micron      6314      * |     0           A
  [12162992]  2micron      6315      * |     0           A
  [12162993]  2micron      6316      * |     0           C
  [12162994]  2micron      6317      * |     0           G
  [12162995]  2micron      6318      * |     0           A
  -------
  seqinfo: 18 sequences (2 circular) from sacCer2 genome
```

HDF5Array / DelayedArray

Convenient access and manipulation of HDF5 datasets.

Can be used inside a SummarizedExperiment object (assay data).

```
A dataset with coverage for 6 samples along Human chr 16:

> cov0 <- HDF5Array(tally_file, "/ExampleStudy/16/Coverages")
> cov0
HDF5Array object of 6 x 2 x 90354753 integers:
, , 1
     [,1] [,2]
[1,]    0    0
[2,]    0    0
 ...    .    .
[5,]    0    0
[6,]    0    0
…
, , 90354753
     [,1] [,2]
[1,]    0    0
[2,]    0    0
 ...    .    .
[5,]    0    0
[6,]    0    0
```

HDF5Array / DelayedArray

Support delayed operations.

Result is a DelayedArray object.

`as.array()` would ***realize it in memory***. Don't do that!

Instead ***realize it on disk*** (if you really need to) with `writeHDF5Dataset()`.

```
Compute unstranded coverage:


> pcov <- drop(cov0[ , 1, ])  # delayed
> mcov <- drop(cov0[ , 2, ])  # delayed
> cov <- pcov + mcov          # delayed
> cov
DelayedMatrix object of 6 x 90354753 integers:
            [,1]        [,2]        [,3]           . [,90354751]
[1,]           0           0           0           .            0
[2,]           0           0           0           .            0
[3,]           0           0           0           .            0
[4,]           0           0           0           .            0
[5,]           0           0           0           .            0
[6,]           0           0           0           .            0
     [,90354752] [,90354753]
[1,]           0           0
[2,]           0           0
[3,]           0           0
[4,]           0           0
[5,]           0           0
[6,]           0           0
```

HDF5Array / DelayedArray

Block-processing:

- Operations that cannot be delayed (e.g. `rowSums()` or matrix multiplication) process the DelayedArray object block-by-block, one block at a time.
- Each block is *realized* (i.e. all delayed operations are executed) and the current operation (e.g. `rowSums`) applied to the result.

See `?DelayedArray` in the HDF5Array package for more information.

```
> sum_cov <- rowSums(cov)  # block-processing
> sum_cov
[1] 39807797 45246576 18405376 36487401 17218497 36681571

> gc()
          used  (Mb) gc trigger  (Mb) max used  (Mb)
Ncells 2947878 157.5    4703850 251.3  4703850 251.3
Vcells 3765245  28.8   67472700 514.8 58464312 446.1

Loading the full dataset at once in memory would use 4 Gb
of RAM!
```

What’s next?

- **HDF5Array:**
  - Support more operations on DelayedArray objects
  - Vignette
  - Integration of HDF5Array to some common workflows (e.g. `summarizeOverlaps`)

- Support **long Vector derivatives** (e.g. long Rle, long DataFrame, long GRanges, long Hits, long DNAString, long DNAStringSet, etc). Will require important changes to the internals of several core packages (S4Vectors, IRanges, GenomicRanges, Biostrings, and more...)

- **On-disk GRanges objects.** Indexed for fast extraction of elements that overlap a set of regions of interest (i.e. fast `subsetByOverlaps`). Analog to `scanBam` “which” feature. An immediate use case for this is to speed up `snpsByOverlaps`.

- Support **easy creation of standalone BSgenome objects** (from 2bit, FASTA, and maybe other sources).

- Maybe other "**genomic Views**" objects (in addition to BSgenomeViews).

- Build system: **incremental builds**.