Conducting Genomic Symphonies with Bioconductor

Michael Lawrence

December 4, 2017

Outline

Genentech

Genomic workflows

R

Bioconductor

Usability

HelloRanges

plyranges

Outline

Genentech

Genomic workflows

R

Bioconductor

Usability

HelloRanges

plyranges

Fast facts about Genentech and Roche

- Genentech
  - Founded in 1976
  - Headquartered in South San Francisco
  - ~14,000 employees
  - Became a member of the Roche Group in March 2009
  - Headquarters for all Roche pharmaceutical operations in the U.S.
- Roche Group
  - Founded in 1896
  - Headquartered in Basel, Switzerland
  - ~88,500 employees worldwide, active in 150 countries
  - World's largest biotech company
  - Top five globally in pharmaceuticals
  - Number one globally in *in vitro* diagnostics

Statistics from 2016

Genentech
A Member of the Roche Grout

gRED’s emphasis on scientific research

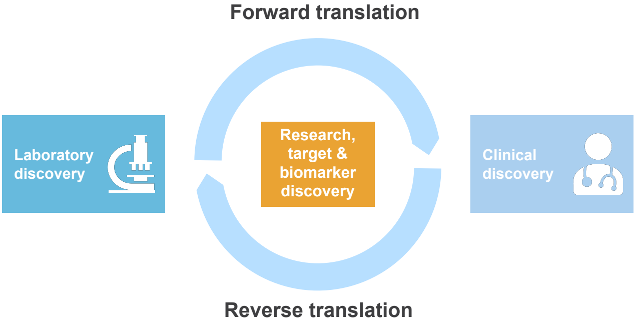
2,100	**gRED employees**
1,200	researchers and scientists
785,000	square feet dedicated to research; the largest in the world

3,303	**peer-reviewed publications in the last ten years**
22	*Nature*, *Science* and *Cell* publications in 2014
#1	employer according to *Science* for 8 of 13 past years; always in the top 3

Statistics from 2015

Genentech
A Member of the Roche Group

A growing scientific advantage: the ability to combine rich forward and reverse translation



Forward translation

Laboratory discovery

Research, target & biomarker discovery

Clinical discovery

Reverse translation

- The best information about human disease, including response to drug, is in the context of actual human patients.
- Beyond randomization, clinical data are always associative. Nailing down cause and effect—in order to fully justify new therapeutic strategies—requires controlled experiments.

Outline

Genentech

Genomic workflows

R

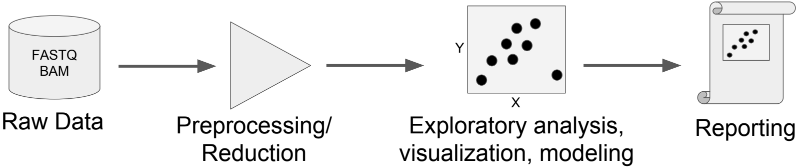
Bioconductor

Usability

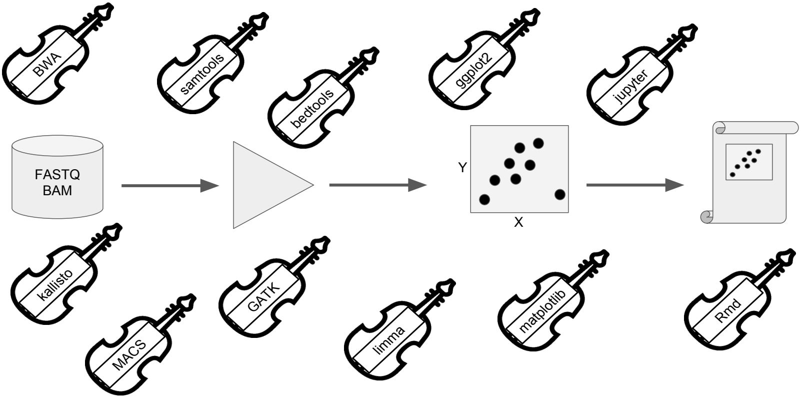
HelloRanges

plyranges

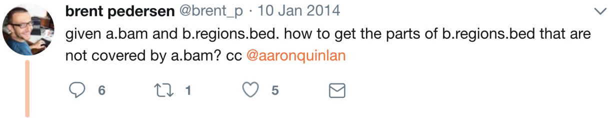
Genomic workflows are symphonies of different tools



Genomic workflows are symphonies of different tools



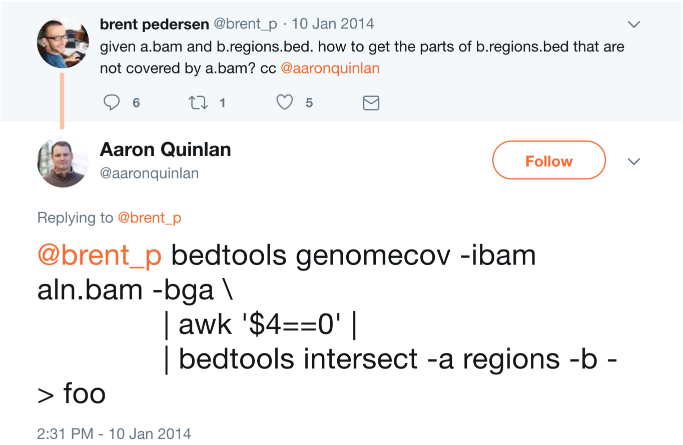
Tweet-size example from bedtools tutorial



brent pedersen @brent_p · 10 Jan 2014

given a.bam and b.regions.bed. how to get the parts of b.regions.bed that are not covered by a.bam? cc @aaronquinlan

Tweet-size example from bedtools tutorial



brent pedersen @brent_p · 10 Jan 2014

given a.bam and b.regions.bed. how to get the parts of b.regions.bed that are not covered by a.bam? cc @aaronquinlan

6 1 5

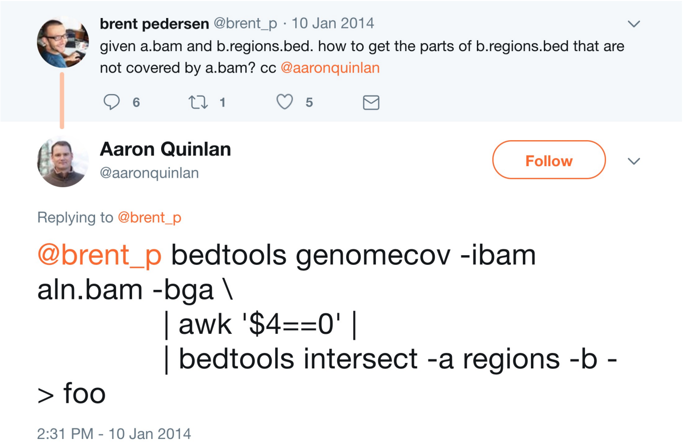
Aaron Quinlan
@aaronquinlan

Replying to @brent_p

```
@brent_p bedtools genomecov -ibam
aln.bam -bga \
              | awk '$4==0' |
              | bedtools intersect -a regions -b -
> foo
```

2:31 PM - 10 Jan 2014

Tweet-size example from bedtools tutorial



brent pedersen @brent_p · 10 Jan 2014

given a.bam and b.regions.bed. how to get the parts of b.regions.bed that are not covered by a.bam? cc @aaronquinlan

6 1 5

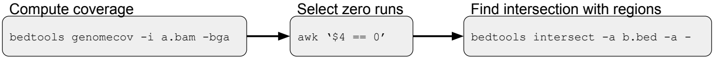
Aaron Quinlan
@aaronquinlan

Follow

Replying to @brent_p

```
@brent_p bedtools genomecov -ibam
aln.bam -bga \
          | awk '$4==0' |
          | bedtools intersect -a regions -b -
> foo
```

2:31 PM - 10 Jan 2014



Compute coverage

`bedtools genomecov -i a.bam -bga`

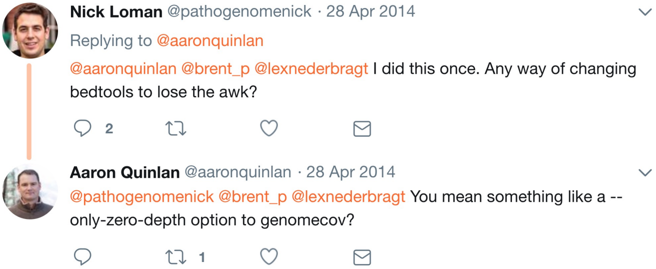
Select zero runs

`awk '$4 == 0'`

Find intersection with regions

`bedtools intersect -a b.bed -a -`

Tweet-size example from bedtools tutorial



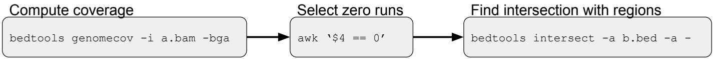
Nick Loman @pathogenomenick · 28 Apr 2014

Replying to @aaronquinlan

@aaronquinlan @brent_p @lexnederbragt I did this once. Any way of changing bedtools to lose the awk?

Aaron Quinlan @aaronquinlan · 28 Apr 2014

@pathogenomenick @brent_p @lexnederbragt You mean something like a --only-zero-depth option to genomecov?



Compute coverage

```
bedtools genomecov -i a.bam -bga
```

Select zero runs

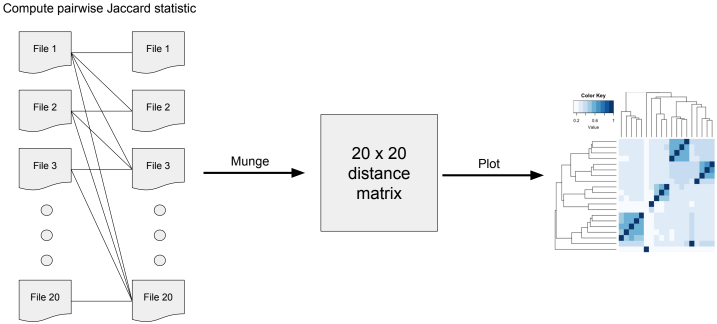
```
awk '$4 == 0'
```

Find intersection with regions

```
bedtools intersect -a b.bed -a -
```

Typical real-world example from bedtools tutorial

Compute the pairwise similarity between samples of DNAse hypersensitivity regions, according to the bedtools Jaccard statistic.



bedtools solution

Languages used

Side-effects

bedtools solution

Languages used

- shell
- GNU parallel
- awk

Compute pairwise distances in parallel

```
parallel ”bedtools jaccard -a {1} -b {2} \
      | awk ’NR>1’ \
      | cut -f 3 \
      > {1}.{2}.jaccard” \
      ::: `ls *.merge.bed`
      ::: `ls *.merge.bed`
```

Side-effects

- 400 .jaccard

bedtools solution

Languages used

- shell
- GNU parallel
- awk
- sed
- perl

Side-effects

- 400 `.jaccard`
- `pairwise.txt`

Combine jaccard files

```
find . \
   | grep jaccard \
   | xargs grep "" \
   | sed -e s"/\.\///" \
   | perl -pi -e "s/.bed./.bed\t/" \
   | perl -pi -e "s/.jaccard:/\t/" \
   > pairwise.txt
```

bedtools solution

Languages used

- shell
- GNU parallel
- awk
- sed
- perl
- python

Side-effects

- 400 `.jaccard`
- `pairwise.txt`
- `pairwise.mat`

Reshape into matrix

```
awk ’NF==3’ pairwise.txt \
| awk ’$1 ~ /^f/ && $2 ~ /^f/’ \
| python make-matrix.py \
> pairwise.mat
```

bedtools solution

Languages used

- shell
- GNU parallel
- awk
- sed
- perl
- python
- R

Side-effects

- 400 .jaccard
- pairwise.txt
- pairwise.mat

Plot the matrix

R

```
library(gplots)
library(RColorBrewer)
jaccard_df <-
    read.table('pairwise.dnase.mat')
jaccard_matrix <-
    as.matrix(jaccard_df[,-1])
heatmap.2(jaccard_matrix,
          col = brewer.pal(9, "Blues"),
          margins = c(14, 14),
          density.info = "none",
          lhei = c(2, 8),
          trace = "none")
```

Typical obstacles in implementing genomic data analyses

- Tools are difficult to build, install and run
- Limitations require mixing languages and semi-compatible, inconsistently documented toolsets
- Interoperability depends on inefficient, complex file formats
- Analyst has to directly manipulate and manage files, instead of focusing on the analysis
- Reproducibility is hard

Outline

Genentech

Genomic workflows

R

Bioconductor

Usability

HelloRanges

plyranges

R is a platform and language for statistical computing

- Core principes according to John Chambers in "Extending R":
  - Everything is an object
  - Everything that happens is a function call
  - Interfaces to other software are core to R
- Addendum: every published extension is a package
  - Primary mechanism for distributing statistical computing research

R packages are easy to install

- CRAN, Bioconductor distribute vetted packages
  - Tested as a cohort
  - Standardized through `R CMD check`
- Package installation usually just works
  - `install.packages("gplots")`

R has consistent, function-level documentation

Standalone programs provide documentation in different ways:

- `man bedtools`?
- `bedtools intersect --help`?
- Google?

Every R package provides a man page of each function:

```
?brewer.pal
```

```
ColorBrewer palettes

Description:

     Creates nice looking color palettes especially for thematic maps

Usage:

     brewer.pal(n, name)

Arguments:

       n: Number of different colors in the palette, minimum 3, maximum
          depending on palette

    name: A palette name from the lists below
```

R enables reproducibility

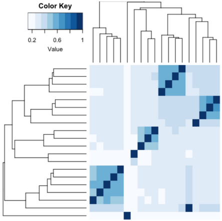
- Dependencies trackable through versioned packages
- Packages like switchr and packrat make it easy to record and restore sets of package versions
- `sessionInfo()`

```
R Under development (unstable) (2017-08-02 r73018)
Platform: x86_64-apple-darwin15.6.0 (64-bit)
Running under: OS X El Capitan 10.11.6

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] gplots_3.0.1       RColorBrewer_1.1-2

loaded via a namespace (and not attached):
[1] compiler_3.5.0     tools_3.5.0        KernSmooth_2.23-15 gdata_2.18.0
[5] caTools_1.17.1     bitops_1.0-6       gtools_3.5.0
```

R unifies workflows through object-oriented interfaces

An object affords interoperability and abstraction:

```
library(gplots)
library(RColorBrewer)
jaccard_df <- utils::read.table(’pairwise.mat’)
jaccard_matrix <- as.matrix(jaccard_df[,-1])
heatmap.2(jaccard_matrix, col = brewer.pal(9, "Blues"))
```



R is improving

Pushing object orientation down to the C level

R 3.5 will add:

- Object-oriented mechanism for custom implementations of R vectors

  Compact representations Run-length encodings, `1:10` sequences

  External storage Spark, databases, HDF5, Arrow, etc

- Notions of sortedness and any missingness to the vector API
- Heuristics that construct compact vectors when it makes sense

Luke Tierney, Gabe Becker, Tomas Kalibera

Outline

Genentech

Genomic workflows

R

Bioconductor

Usability

HelloRanges

plyranges

Bioconductor

A unified platform for the analysis and comprehension of high-throughput genomic data.

- Started 2002
- Led by Martin Morgan
- Core infrastructure maintained by about 8 people, based in Roswell Park CRC in Buffalo, NY
- 1476 software packages that form a unified platform
- Well-used and respected.
  - 53k unique IP downloads / month.
  - 21,700 PubMedCentral citations.
- Embraces the R principles of object, function, interface and package

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

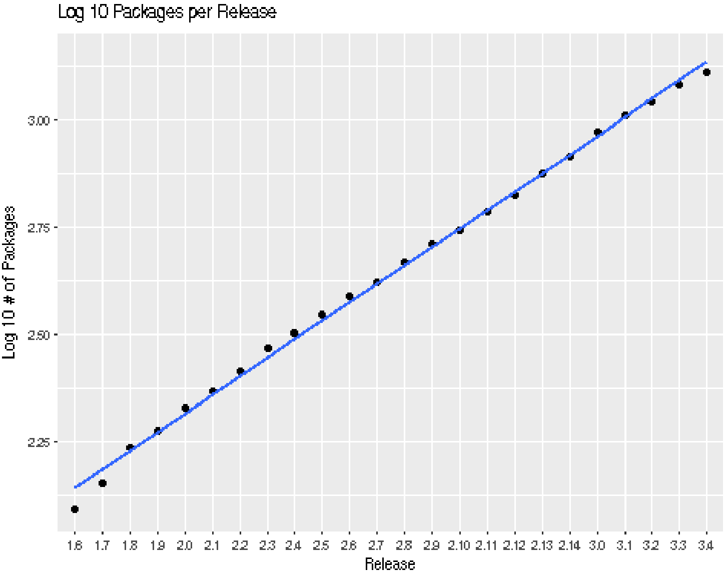
ROSWELL PARK CANCER INSTITUTE

NIH National Human Genome Research Institute

NIH NATIONAL CANCER INSTITUTE

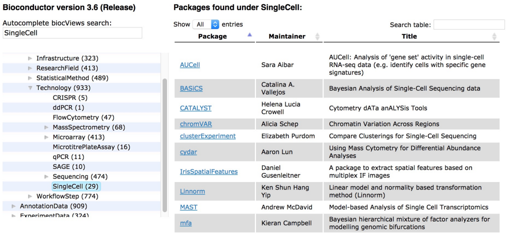
SOUND

Bioconductor is growing



Bioconductor qualities

- Discoverable
- Installable
- Reliable
- Documented
- Supported
- Integrated
- Scalable
- State of the art
- Community-driven



Bioconductor version 3.6 (Release)

Autocomplete biocViews search: SingleCell

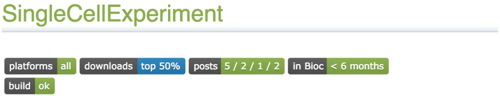
- Infrastructure (323)
- ResearchField (413)
- StatisticalMethod (489)
- Technology (933)
  - CRISPR (5)
  - ddPCR (1)
  - FlowCytometry (47)
  - MassSpectrometry (68)
  - Microarray (413)
  - MicrotitrePlateAssay (16)
  - qPCR (11)
  - SAGE (10)
  - Sequencing (474)
  - SingleCell (29)
- WorkflowStep (774)
- AnnotationData (909)
- ExperimentData (324)

Packages found under SingleCell:

Show All entries

Search table:

Package	Maintainer	Title
AUCell	Sara Aibar	AUCell: Analysis of 'gene set' activity in single-cell RNA-seq data (e.g. identify cells with specific gene signatures)
BASiCS	Catalina A. Vallejos	Bayesian Analysis of Single-Cell Sequencing data
CATALYST	Helena Lucia Crowell	Cytometry dATa anALYSis Tools
chromVAR	Alicia Schep	Chromatin Variation Across Regions
clusterExperiment	Elizabeth Purdom	Compare Clusterings for Single-Cell Sequencing
cydar	Aaron Lun	Using Mass Cytometry for Differential Abundance Analyses
IrisSpatialFeatures	Daniel Gusenleitner	A package to extract spatial features based on multiplex IF images
Linnorm	Ken Shun Hang Yip	Linear model and normality based transformation method (Linnorm)
MAST	Andrew McDavid	Model-based Analysis of Single Cell Transcriptomics
mfa	Kieran Campbell	Bayesian hierarchical mixture of factor analyzers for modelling genomic bifurcations



SingleCellExperiment

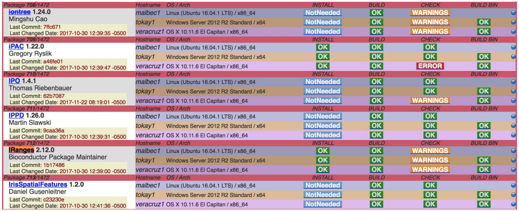
Bioconductor qualities

- Discoverable
- Installable
- Reliable
- Documented
- Supported
- Integrated
- Scalable
- State of the art
- Community-driven

```
source("https://bioconductor.org/biocLite.R")
biocLite()
biocLite("Gviz")
```

Bioconductor qualities

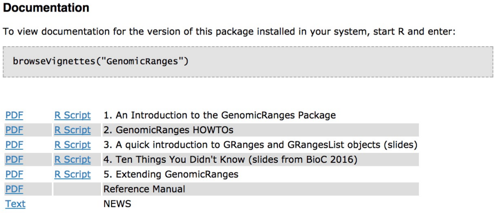
- Discoverable
- Installable
- Reliable
- Documented
- Supported
- Integrated
- Scalable
- State of the art
- Community-driven



Package 708/1472	Hostname	OS / Arch	INSTALL	BUILD	CHECK	BUILD BIN	
~~iontree~~ 1.24.0 Mingshu Cao Last Commit: 7ffc671 Last Changed Date: 2017-10-30 12:39:35 -0500	malbec1	Linux (Ubuntu 16.04.1 LTS) / x86_64	NotNeeded	OK	WARNINGS		●
	tokay1	Windows Server 2012 R2 Standard / x64	NotNeeded	OK	WARNINGS	OK	●
	veracruz1	OS X 10.11.6 El Capitan / x86_64	NotNeeded	OK	WARNINGS	OK	●
Package 709/1472	Hostname	OS / Arch	INSTALL	BUILD	CHECK	BUILD BIN	
iPAC 1.22.0 Gregory Ryslik Last Commit: a46fe01 Last Changed Date: 2017-10-30 12:39:47 -0500	malbec1	Linux (Ubuntu 16.04.1 LTS) / x86_64	OK	OK	OK		●
	tokay1	Windows Server 2012 R2 Standard / x64	OK	OK	OK	OK	●
	veracruz1	OS X 10.11.6 El Capitan / x86_64	OK	OK	ERROR	OK	
Package 710/1472	Hostname	OS / Arch	INSTALL	BUILD	CHECK	BUILD BIN	
IPO 1.4.1 Thomas Riebenbauer Last Commit: 62b7087 Last Changed Date: 2017-11-22 08:19:01 -0500	malbec1	Linux (Ubuntu 16.04.1 LTS) / x86_64	NotNeeded	OK	OK		●
	tokay1	Windows Server 2012 R2 Standard / x64	NotNeeded	OK	OK	OK	●
	veracruz1	OS X 10.11.6 El Capitan / x86_64	NotNeeded	OK	WARNINGS	OK	●
Package 711/1472	Hostname	OS / Arch	INSTALL	BUILD	CHECK	BUILD BIN	
IPPD 1.26.0 Martin Slawski Last Commit: 9caa36a Last Changed Date: 2017-10-30 12:39:31 -0500	malbec1	Linux (Ubuntu 16.04.1 LTS) / x86_64	NotNeeded	OK	OK		●
	tokay1	Windows Server 2012 R2 Standard / x64	NotNeeded	OK	OK	OK	●
	veracruz1	OS X 10.11.6 El Capitan / x86_64	NotNeeded	OK	OK	OK	●
Package 712/1472	Hostname	OS / Arch	INSTALL	BUILD	CHECK	BUILD BIN	
IRanges 2.12.0 Bioconductor Package Maintainer Last Commit: 1b17486 Last Changed Date: 2017-10-30 12:39:00 -0500	malbec1	Linux (Ubuntu 16.04.1 LTS) / x86_64	OK	OK	WARNINGS		●
	tokay1	Windows Server 2012 R2 Standard / x64	OK	OK	WARNINGS	OK	●
	veracruz1	OS X 10.11.6 El Capitan / x86_64	OK	OK	WARNINGS	OK	●
Package 713/1472	Hostname	OS / Arch	INSTALL	BUILD	CHECK	BUILD BIN	
IrisSpatialFeatures 1.2.0 Daniel Gusenleitner Last Commit: c23230e Last Changed Date: 2017-10-30 12:41:36 -0500	malbec1	Linux (Ubuntu 16.04.1 LTS) / x86_64	NotNeeded	OK	OK		●
	tokay1	Windows Server 2012 R2 Standard / x64	NotNeeded	OK	OK	OK	●
	veracruz1	OS X 10.11.6 El Capitan / x86_64	NotNeeded	OK	OK	OK	●

Bioconductor qualities

- Discoverable
- Installable
- Reliable
- Documented
- Supported
- Integrated
- Scalable
- State of the art
- Community-driven



Documentation

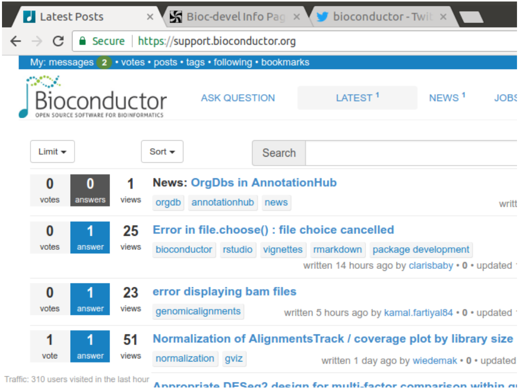
To view documentation for the version of this package installed in your system, start R and enter:

```
browseVignettes("GenomicRanges")
```

PDF	R Script	1. An Introduction to the GenomicRanges Package
PDF	R Script	2. GenomicRanges HOWTOs
PDF	R Script	3. A quick introduction to GRanges and GRangesList objects (slides)
PDF	R Script	4. Ten Things You Didn't Know (slides from BioC 2016)
PDF	R Script	5. Extending GenomicRanges
PDF		Reference Manual
Text		NEWS

Bioconductor qualities

- Discoverable
- Installable
- Reliable
- Documented
- Supported
- Integrated
- Scalable
- State of the art
- Community-driven


Latest Posts
Bioc-devel Info Pag
bioconductor - Twit
Secure
https://support.bioconductor.org
My: messages 2 • votes • posts • tags • following • bookmarks
Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS
ASK QUESTION
LATEST 1
NEWS 1
JOBS
Limit
Sort
Search

0 votes 0 answers 1 views **News: OrgDbs in AnnotationHub**
orgdb annotationhub news
writt

0 votes 1 answer 25 views **Error in file.choose() : file choice cancelled**
bioconductor rstudio vignettes rmarkdown package development
written 14 hours ago by clarisbaby • **0** • updated

0 votes 1 answer 23 views **error displaying bam files**
genomicalignments
written 5 hours ago by kamal.fartiyal84 • **0** • updated

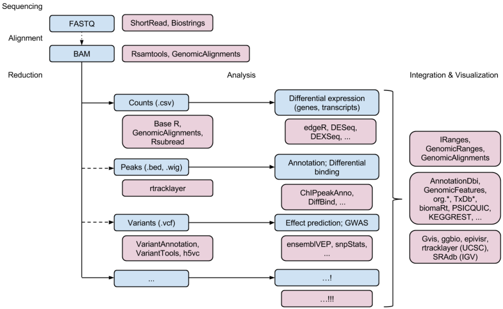
1 vote 1 answer 51 views **Normalization of AlignmentsTrack / coverage plot by library size**
normalization gviz
written 1 day ago by wiedemak • **0** • updated

Traffic: 310 users visited in the last hour

Appropriate DESeq2 design for multi-factor comparison within gr

Bioconductor qualities

- Discoverable
- Installable
- Reliable
- Documented
- Supported
- Integrated
- Scalable
- State of the art
- Community-driven



Sequencing

FASTQ

ShortRead, Biostrings

Alignment

BAM

Rsamtools, GenomicAlignments

Reduction

Analysis

Integration & Visualization

Counts (.csv)

Base R, GenomicAlignments, Rsubread

Differential expression (genes, transcripts)

edgeR, DESeq, DEXSeq, ...

Peaks (.bed, .wig)

rtracklayer

Annotation; Differential binding

ChIPpeakAnno, DiffBind, ...

Variants (.vcf)

VariantAnnotation, VariantTools, h5vc

Effect prediction; GWAS

ensemblVEP, snpStats, ...

...

...!

...!!!

IRanges, GenomicRanges, GenomicAlignments

AnnotationDbi, GenomicFeatures, org.*, TxDb*, biomaRt, PSICQUIC, KEGGREST, ...

Gvis, ggbio, epivisr, rtracklayer (UCSC), SRAdb (IGV)

Bioconductor qualities

- Discoverable
- Installable
- Reliable
- Documented
- Supported
- Integrated
- Scalable
- State of the art
- Community-driven

```r
se <- TENxBrainData()
se
```

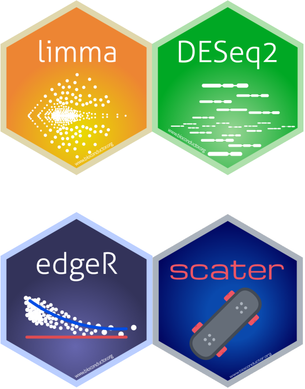
```
## class: SingleCellExperiment
## dim: 27998 1306127
## metadata(0):
## assays(1): counts
## rownames: NULL
## rowData names(2): Ensembl Symbol
## colnames(1306127): AAACCTGAGATAGGAG-1 AAACCTGAGCGGCTTC-1 ...
##   TTTGTCAGTTAAAGTG-133 TTTGTCATCTGAAAGA-133
## colData names(4): Barcode Sequence Library Mouse
## reducedDimNames(0):
## spikeNames(0):
```

```r
libSize <- colSums(assay(se)[, 1:1000])
range(libSize)
```

```
## [1]  1453 34233
```

Bioconductor qualities

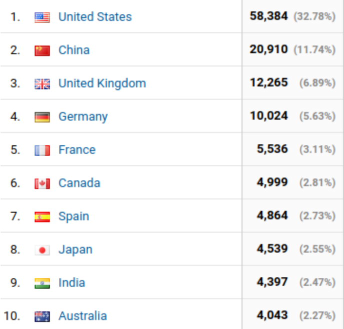
- Discoverable
- Installable
- Reliable
- Documented
- Supported
- Integrated
- Scalable
- State of the art
- Community-driven



Bioconductor qualities

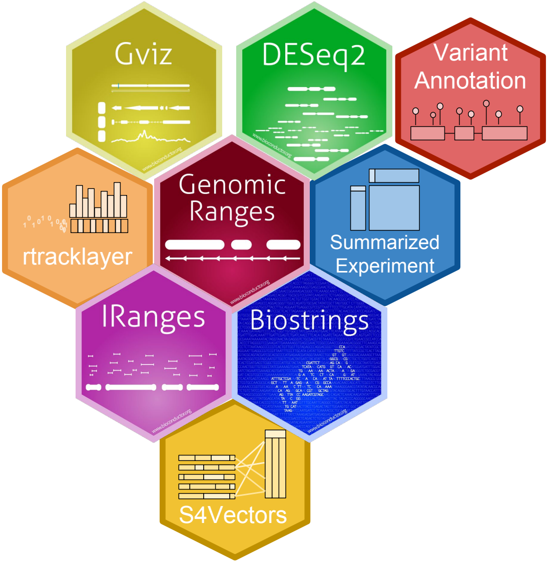
- Discoverable
- Installable
- Reliable
- Documented
- Supported
- Integrated
- Scalable
- State of the art
- Community-driven

- 1064 unique package maintainers
- Web users by country:



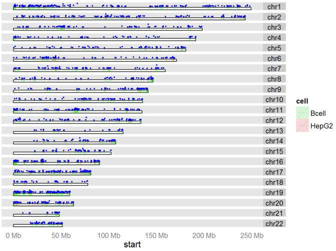
1.	United States	58,384	(32.78%)
2.	China	20,910	(11.74%)
3.	United Kingdom	12,265	(6.89%)
4.	Germany	10,024	(5.63%)
5.	France	5,536	(3.11%)
6.	Canada	4,999	(2.81%)
7.	Spain	4,864	(2.73%)
8.	Japan	4,539	(2.55%)
9.	India	4,397	(2.47%)
10.	Australia	4,043	(2.27%)

Bioconductor is built on shared infrastructure

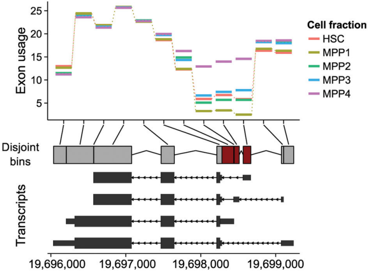


Central data structures of Bioconductor

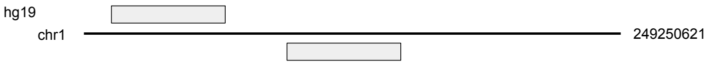
Data on genomic ranges



Summarized data



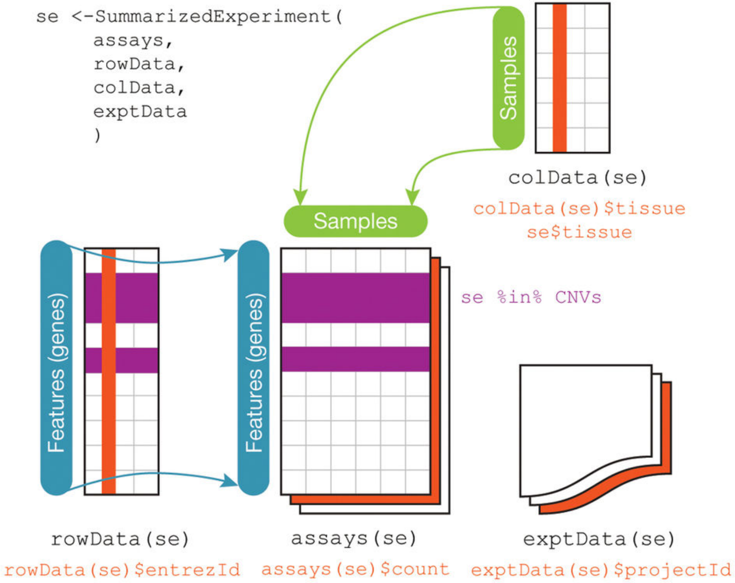
GRanges: data on genomic ranges



seqnames	start	end	strand	...
chr1	1	10	+	
chr1	15	24	-	

- Plus, sequence information (lengths, genome, etc)

SummarizedExperiment: the central data model



```
se <-SummarizedExperiment(
     assays,
     rowData,
     colData,
     exptData
     )
```

```
colData(se)
colData(se)$tissue
se$tissue
```

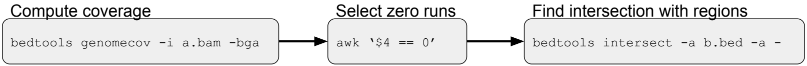
```
se %in% CNVs
```

```
rowData(se)
rowData(se)$entrezId
```

```
assays(se)
assays(se)$count
```

```
exptData(se)
exptData(se)$projectId
```

Bioconducting the tweeted workflow



Compute coverage

```
bedtools genomecov -i a.bam -bga
```

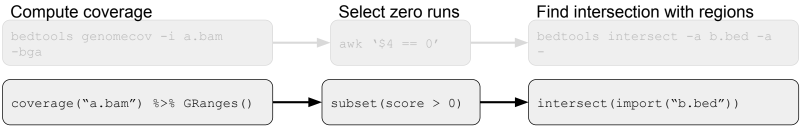
Select zero runs

```
awk '$4 == 0'
```

Find intersection with regions

```
bedtools intersect -a b.bed -a -
```

Bioconducting the tweeted workflow



Compute coverage

```
bedtools genomecov -i a.bam
-bga
```

```
coverage(“a.bam”) %>% GRanges()
```

Select zero runs

```
awk ‘$4 == 0’
```

```
subset(score > 0)
```

Find intersection with regions

```
bedtools intersect -a b.bed -a
-
```

```
intersect(import(“b.bed”))
```

Bioconducting the pairwise Jaccard workflow

Define a function for the Jaccard statistic

```
jaccard <- function(x, y) {
    gr_x <- import(x)
    gr_y <- import(y)
    intersects <- intersect(gr_x, gr_y, ignore.strand=TRUE)
    unions <- union(gr_x, gr_y, ignore.strand=TRUE)
    sum(width(intersects)) / sum(width(unions))
}
```

Bioconducting the pairwise Jaccard workflow

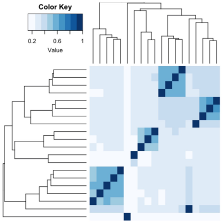
Compute the statistics in parallel

```r
files <- Sys.glob("*.merge.bed")
jaccard_matrix <- outer(files, files,
      function(a, b) mcmapply(jaccard, a, b))
```

Bioconducting the pairwise Jaccard workflow

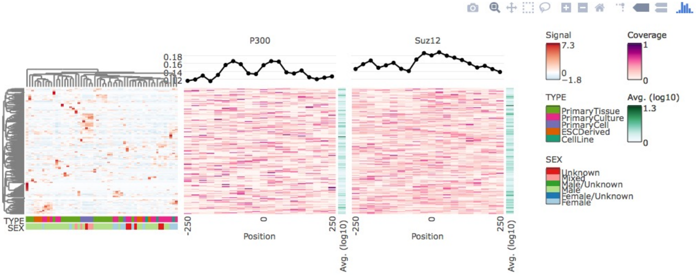
Make the plot

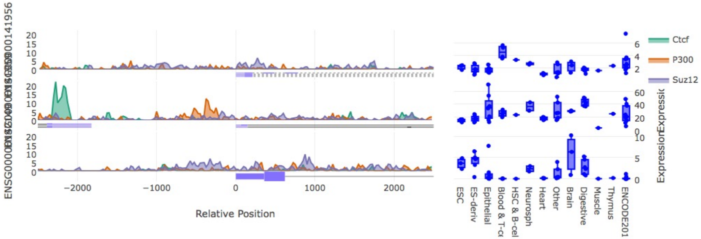
```
library(gplots)
library(RColorBrewer)
heatmap.2(jaccard_matrix, col = brewer.pal(9, "Blues"))
```



GenomicWidgets: interactive genomic plots for Shiny/RMD

by Alicia Schep, Sarah Kummerfeld at Genentech





Outline

Genentech

Genomic workflows

R

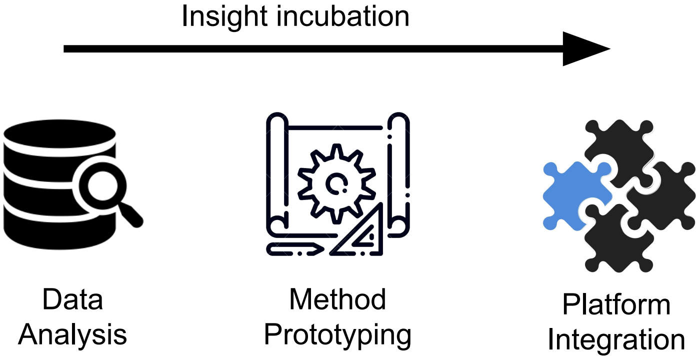
Bioconductor

Usability

HelloRanges

plyranges

The Ranges infrastructure is an incubator



Insight incubation

Data Analysis

Method Prototyping

Platform Integration

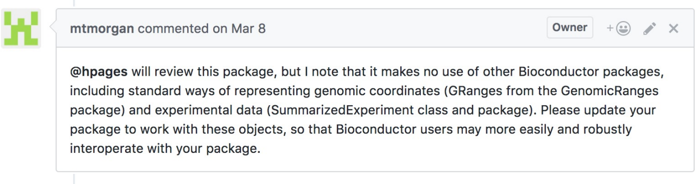
- Should be accessible to the average Bioconductor user

Is the transition happening?

- From a typical package submission:

```
Imports: checkmate, dplyr, ggplot2, tidyr
```

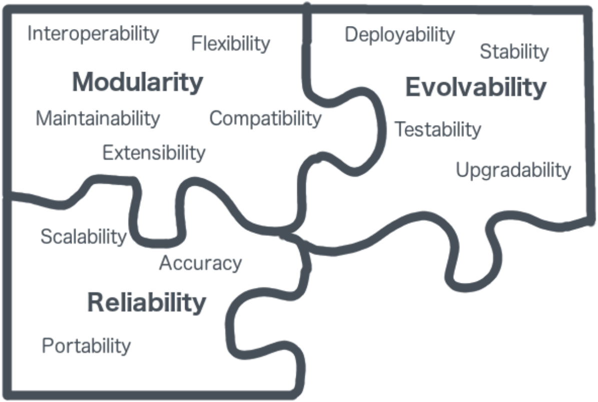
- A typical initial response:



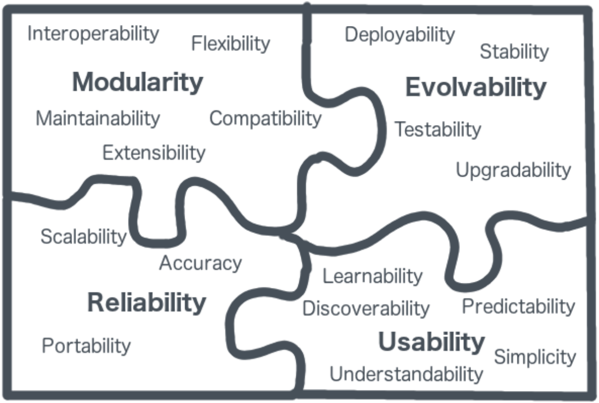
mtmorgan commented on Mar 8

@hpages will review this package, but I note that it makes no use of other Bioconductor packages, including standard ways of representing genomic coordinates (GRanges from the GenomicRanges package) and experimental data (SummarizedExperiment class and package). Please update your package to work with these objects, so that Bioconductor users may more easily and robustly interoperate with your package.

Aspects of software quality: the ilities



Aspects of software quality: the ilities



Cognitive Dimensions of Notations

- Thomas Green and Marian Petre (1996) proposed 14 dimensions of usability in the context of visual programming
- Many are interrelated and in balance with each other
- Guide for evaluating usability and as a framework for discussing interface design trade-offs

Green's cognitive dimensions

- Abstraction gradient
- Closeness of mapping
- Consistency
- Diffuseness
- Error-proneness
- Hard mental operations
- Hidden dependencies
- Provisionality
- Premature commitment
- Progressive evaluation
- Role-expressiveness
- Secondary notation
- Viscosity (robustness)
- Visibility

Abstraction

Procedural abstraction

A compound operation that enables the user tell the computer what to do without telling it how to do it.

Data abstraction

> *"A methodology that enables us to isolate how a compound data object is used from the details of how it is constructed from more primitive data objects"*

Structure and Interpretation of Computer Programs (1979)

In the absence of abstraction

- We often start with a BED file:

  ```
  bash-3.2$  ls *.bed
  ```

  ```
  my.bed
  ```

- And we turn to R to analyze the data

  ```
  df <- read.table("my.bed", sep="\t")
  colnames(df) <- c("chrom", "start", "end")
  ```

  ```
    chrom     start       end
  1  chr7 127471196 127472363
  2  chr7 127472363 127473530
  3  chr7 127473530 127474697
  4  chr9 127474697 127475864
  5  chr9 127475864 127477031
  ```

But file formats differ in important ways

Now for a GFF file:

```r
df <- read.table("my.bed", sep="\t")
colnames(df) <- c("chr", "start", "end")
```

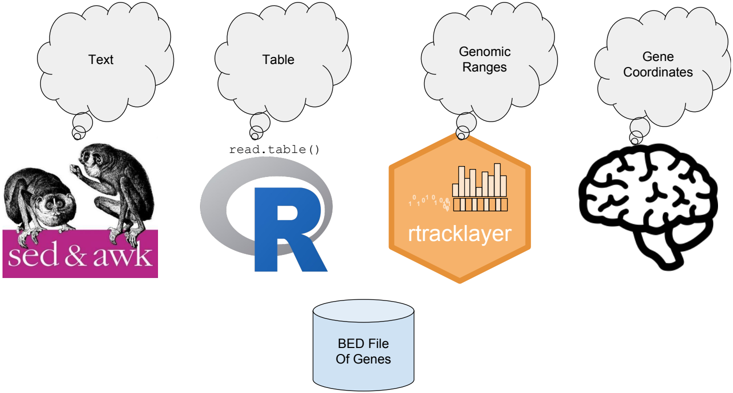
GFF

```
   chr     start       end
1 chr7 127471197 127472363
2 chr7 127472364 127473530
3 chr7 127473531 127474697
4 chr9 127474698 127475864
5 chr9 127475865 127477031
```

BED

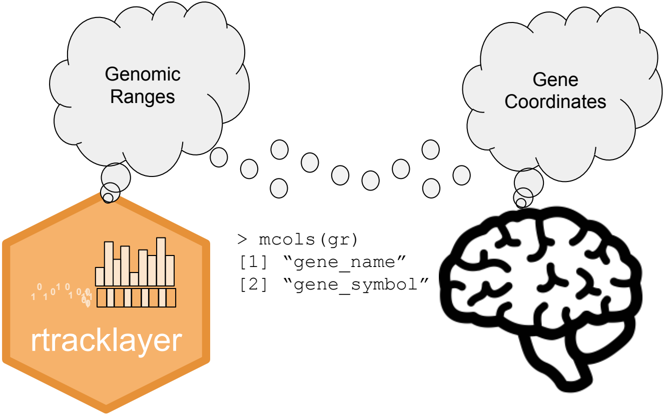
```
  chrom     start       end
1  chr7 127471196 127472363
2  chr7 127472363 127473530
3  chr7 127473530 127474697
4  chr9 127474697 127475864
5  chr9 127475864 127477031
```

Abstraction lets us focus on the important



- Abstraction is semantic enrichment
  - Enables the user to think of data in terms of the problem domain
  - Hides implementation details
  - Unifies frameworks

Semantic slack with adjectives



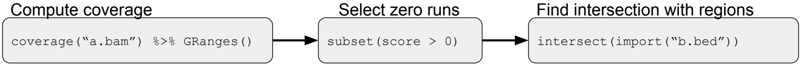
```
> mcols(gr)
[1] “gene_name”
[2] “gene_symbol”
```

- Science defies rigidity: we define flexible objects that combine strongly typed fields with arbitrary user-level metadata

Diffuseness (vs expressiveness)

- Relates to the information density of the code and how well it communicates the *intent* of the programmer
- Enable the user to convey more meaning with less code
- Terseness for its own sake makes code obscure, difficult to unpack
- For genomic data, we want the user to express computations in terms of the biology

Our workflow could be more expressive



Compute coverage

```
coverage("a.bam") %>% GRanges()
```

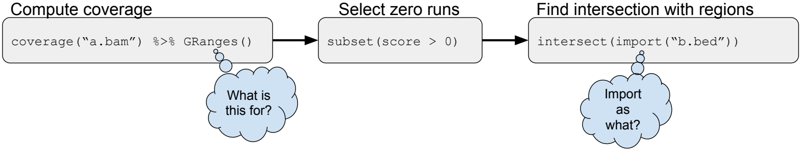
Select zero runs

```
subset(score > 0)
```

Find intersection with regions

```
intersect(import("b.bed"))
```

Our workflow could be more expressive



```
coverage("a.bam") %>% GRanges()
```

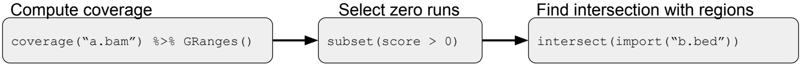
```
subset(score > 0)
```

```
intersect(import("b.bed"))
```

Hard mental operations

How hard the user has to think about things other than the motivating task

Bioconductor is intrinsically complex



Compute coverage

```
coverage("a.bam") %>% GRanges()
```

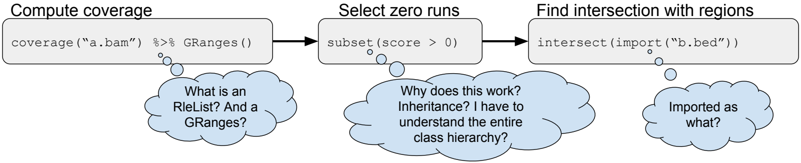
Select zero runs

```
subset(score > 0)
```

Find intersection with regions

```
intersect(import("b.bed"))
```

Bioconductor is intrinsically complex



Compute coverage

```
coverage("a.bam") %>% GRanges()
```

What is an RleList? And a GRanges?

Select zero runs

```
subset(score > 0)
```

Why does this work? Inheritance? I have to understand the entire class hierarchy?

Find intersection with regions

```
intersect(import("b.bed"))
```

Imported as what?

Language complexity

- Bioconductor has large, complex APIs

```
library(VariantAnnotation)
length(methods(class="GRanges"))
```

```
[1] 278
```

- Bioconductor has large, complex class hierarchies

```
pkgs <- package_dependencies("rtracklayer",
                             installed.packages())[[1L]]
pkgs <- setdiff(pkgs, c("methods", "XML", "RCurl"))
cl <- unlist(lapply(pkgs,
             function(p) getClasses(getNamespace(p))))
length(cl)
```

```
[1] 243
```

- In total, 2239 methods on 422 generics

What needs to improve?

- Education?
- Documentation?
- The software?
- All of the above?

Outline

Genentech

Genomic workflows

R

Bioconductor

Usability

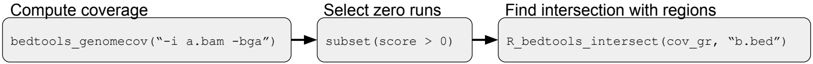
HelloRanges

plyranges

HelloRanges: an onramp to Bioconductor

- bedtools has a low barrier to entry but lacks the supporting ecosystem to cleanly handle realistic workflows
- We want to teach new users how to perform bedtools-style operations within R/Bioconductor
- HelloRanges compiles R code from bedtools invocations, so the student can learn by:
  - studying the output,
  - integrating it into the workflow,
  - and potentially customizing it
- Output prompts the user to fill in details like the genome build
- Supports all bedtools operations and arguments
- Research goal: comparative analysis of bedtools and Bioconductor

HelloRanges exposes the complexity of Bioconductor



Compute coverage

```
bedtools_genomecov(“-i a.bam -bga”)
```

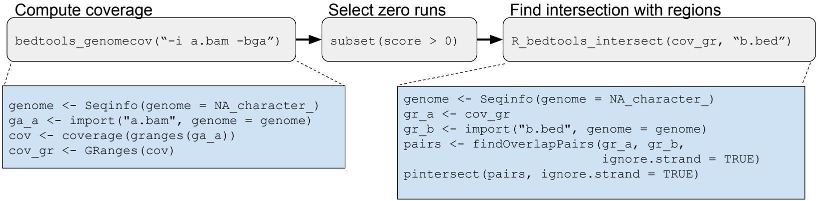
Select zero runs

```
subset(score > 0)
```

Find intersection with regions

```
R_bedtools_intersect(cov_gr, “b.bed”)
```

HelloRanges exposes the complexity of Bioconductor



Compute coverage

```
bedtools_genomecov("-i a.bam -bga")
```

Select zero runs

```
subset(score > 0)
```

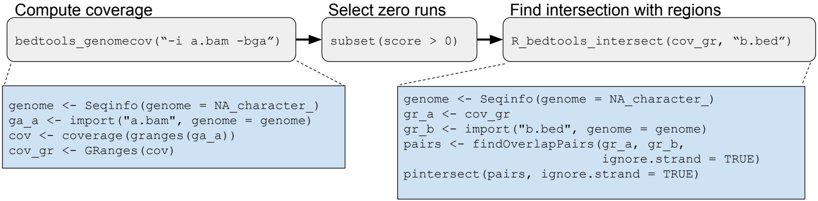
Find intersection with regions

```
R_bedtools_intersect(cov_gr, "b.bed")
```

```
genome <- Seqinfo(genome = NA_character_)
ga_a <- import("a.bam", genome = genome)
cov <- coverage(granges(ga_a))
cov_gr <- GRanges(cov)
```

```
genome <- Seqinfo(genome = NA_character_)
gr_a <- cov_gr
gr_b <- import("b.bed", genome = genome)
pairs <- findOverlapPairs(gr_a, gr_b,
                          ignore.strand = TRUE)
pintersect(pairs, ignore.strand = TRUE)
```

HelloRanges exposes the complexity of Bioconductor



```
bedtools_genomecov("-i a.bam -bga")
```

```
subset(score > 0)
```

```
R_bedtools_intersect(cov_gr, "b.bed")
```

```
genome <- Seqinfo(genome = NA_character_)
ga_a <- import("a.bam", genome = genome)
cov <- coverage(granges(ga_a))
cov_gr <- GRanges(cov)
```

```
genome <- Seqinfo(genome = NA_character_)
gr_a <- cov_gr
gr_b <- import("b.bed", genome = genome)
pairs <- findOverlapPairs(gr_a, gr_b,
                          ignore.strand = TRUE)
pintersect(pairs, ignore.strand = TRUE)
```

Data structures required:

- *Seqinfo*
- *GAlignments*
- *GRanges*
- *RleList*
- *Pairs*

Lesson learned

- Better onramps only help to a point
- Simplifying the software would make everything easier
- The `bedtools` approach of "everything is a BED file" motivates the axiom:

Everything is a GRanges (or SummarizedExperiment)

Consolidating to a small number of data structures enables:

- comprehension,
- endomorphism, and thus
- fluency and chainability

Outline

Genentech

Genomic workflows

R

Bioconductor

Usability

HelloRanges

plyranges

Simplify, but keep the semantics

It can scarcely be denied that the supreme goal of all theory is to make the irreducible basic elements as simple and as few as possible without having to surrender the adequate representation of a single datum of experience.
– Albert Einstein

Everything Should Be Made as Simple as Possible, But Not Simpler
– Apocryphal Einstein quote, paraphasing above

Taking cues from the dplyr package

- dplyr is a API for tabular data manipulation
- Inspired by relational algebra, SQL
- Unified about a single, data model: the tibble
- Operations are:
  - Cohesive (do a single thing)
  - Endomorphic (return the same type as their input)
  - Verb-oriented in syntax
- Fluency emerges from chaining of verbs

```
genes %>%
    group_by(seqnames) %>%
    summarize(count_per_chr=n())
```

Goal

Extend dplyr to genomics, a more complex problem domain, to achieve the accessibility of bedtools

plyranges

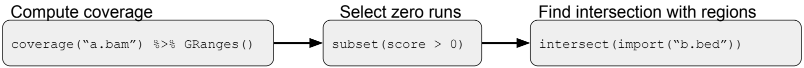
https://github.com/sa-lee/plyranges

- A dplyr-based API for computing on genomic ranges
- Extending the relational algebra with genomic notions
- Large set of visible verbs acting only on the core data structures:

  GRanges represents annotated genomic ranges

  SummarizedExperiment coordinates experimental assay data with sample and feature annotations
- Collaboration with Stuart Lee and Di Cook @ Monash

plyranges is simple and expressive



Compute coverage

```
coverage("a.bam") %>% GRanges()
```

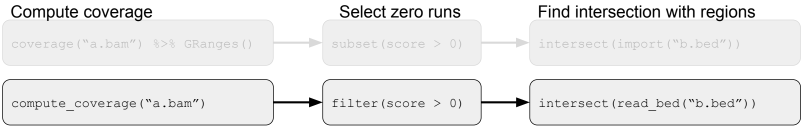
Select zero runs

```
subset(score > 0)
```

Find intersection with regions

```
intersect(import("b.bed"))
```

plyranges is simple and expressive



Compute coverage

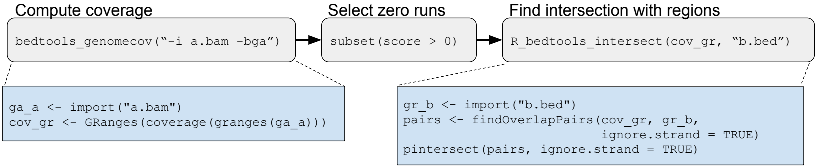
Select zero runs

Find intersection with regions

```
coverage(“a.bam”) %>% GRanges()
subset(score > 0)
intersect(import(“b.bed”))
```

```
compute_coverage(“a.bam”)
filter(score > 0)
intersect(read_bed(“b.bed”))
```

plyranges is simple and expressive



Compute coverage

```
bedtools_genomecov(“-i a.bam -bga”)
```

```
ga_a <- import("a.bam")
cov_gr <- GRanges(coverage(granges(ga_a)))
```

Select zero runs

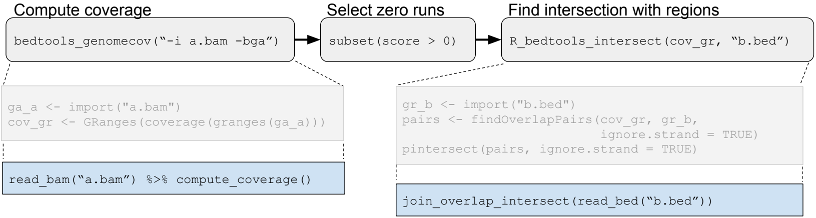
```
subset(score > 0)
```

Find intersection with regions

```
R_bedtools_intersect(cov_gr, “b.bed”)
```

```
gr_b <- import("b.bed")
pairs <- findOverlapPairs(cov_gr, gr_b,
                          ignore.strand = TRUE)
pintersect(pairs, ignore.strand = TRUE)
```

plyranges is simple and expressive



Compute coverage

```
bedtools_genomecov("-i a.bam -bga")
```

Select zero runs

```
subset(score > 0)
```

Find intersection with regions

```
R_bedtools_intersect(cov_gr, "b.bed")
```

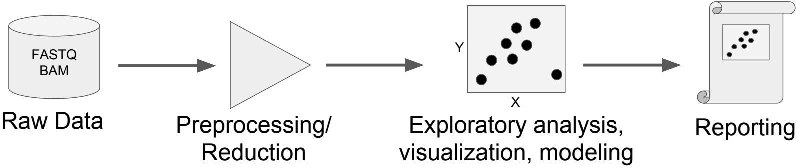
```
ga_a <- import("a.bam")
cov_gr <- GRanges(coverage(granges(ga_a)))
```

```
read_bam("a.bam") %>% compute_coverage()
```

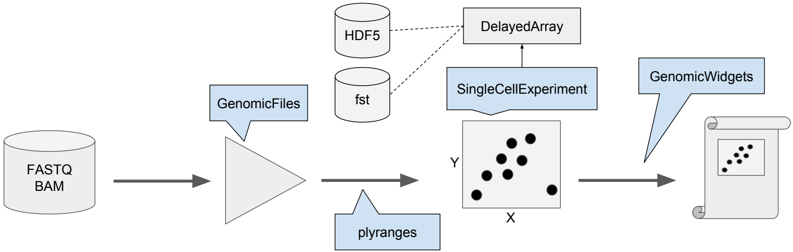
```
gr_b <- import("b.bed")
pairs <- findOverlapPairs(cov_gr, gr_b,
                          ignore.strand = TRUE)
pintersect(pairs, ignore.strand = TRUE)
```

```
join_overlap_intersect(read_bed("b.bed"))
```

The ever evolving Bioconductor



The ever evolving Bioconductor


HDF5
DelayedArray
fst
SingleCellExperiment
GenomicWidgets
GenomicFiles
FASTQ
BAM
Y
X
plyranges