Scientific software in-the-large

Michael Lawrence (Genentech Research)

September 7, 2018

Outline

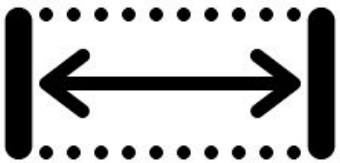
Introduction

Bioconductor as a software distillery

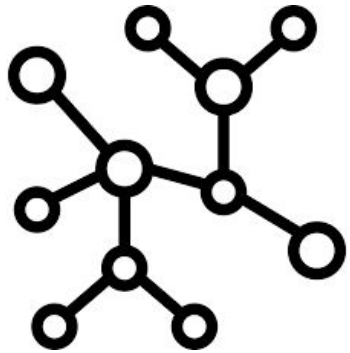
The plyranges package as a catalyst of Bioconductor

Scalability through deferred evaluation and the hailr package

Challenges in genomics software development



Breadth
Many data types
Many questions

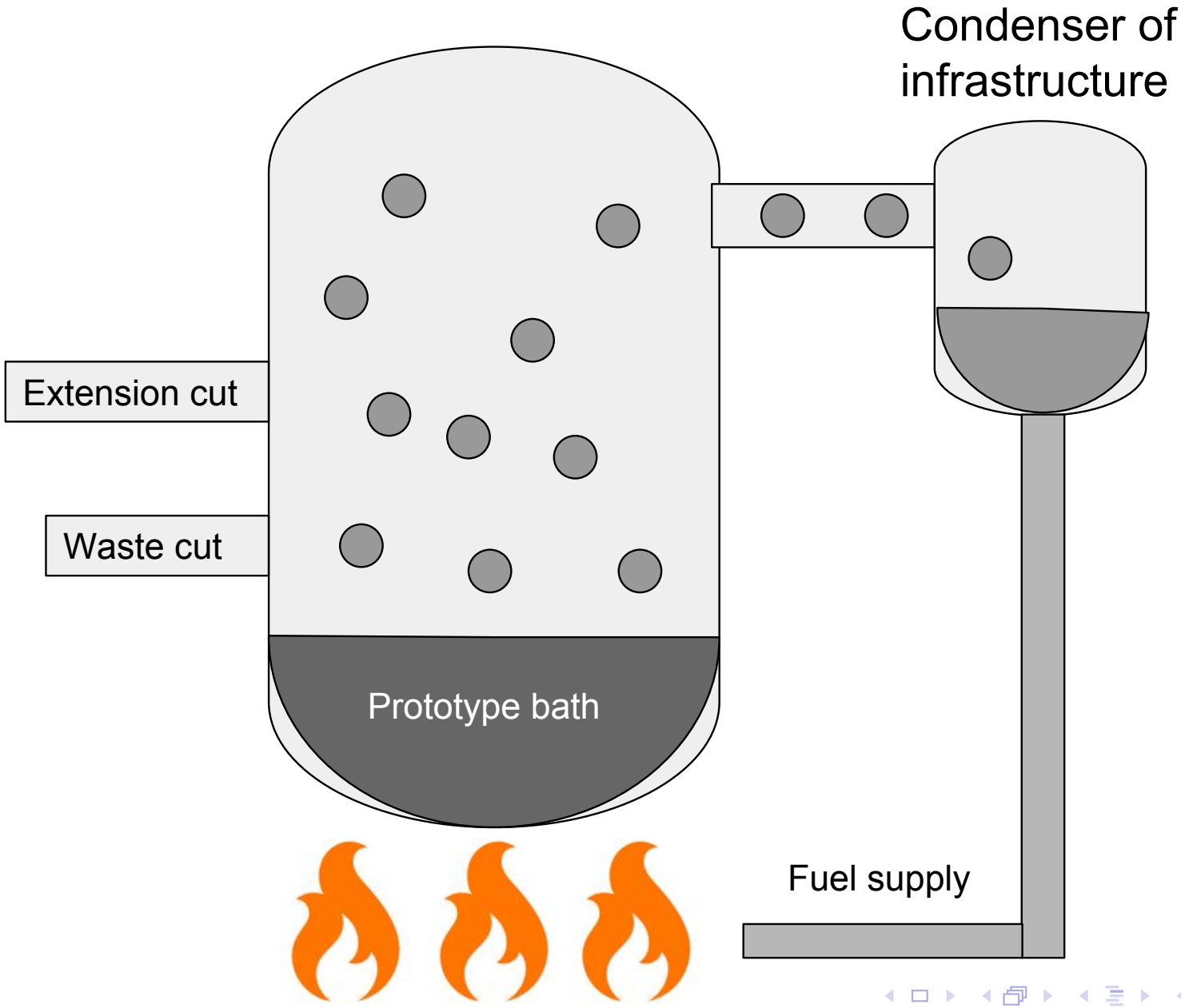


Complexity
Algorithms
Scalability

Evolution
New questions
New technologies

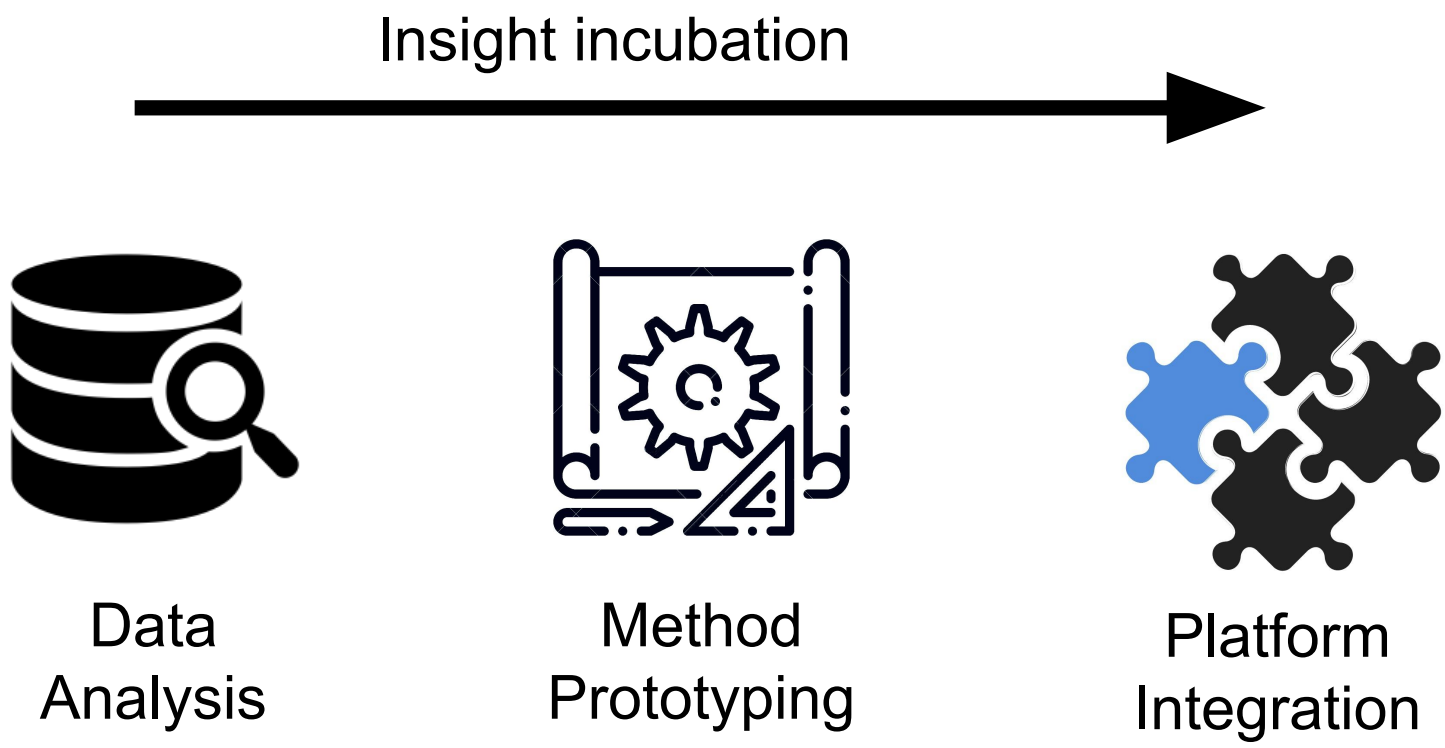
Distilling scientific software

Bottom-up innovation, top-down consolidation

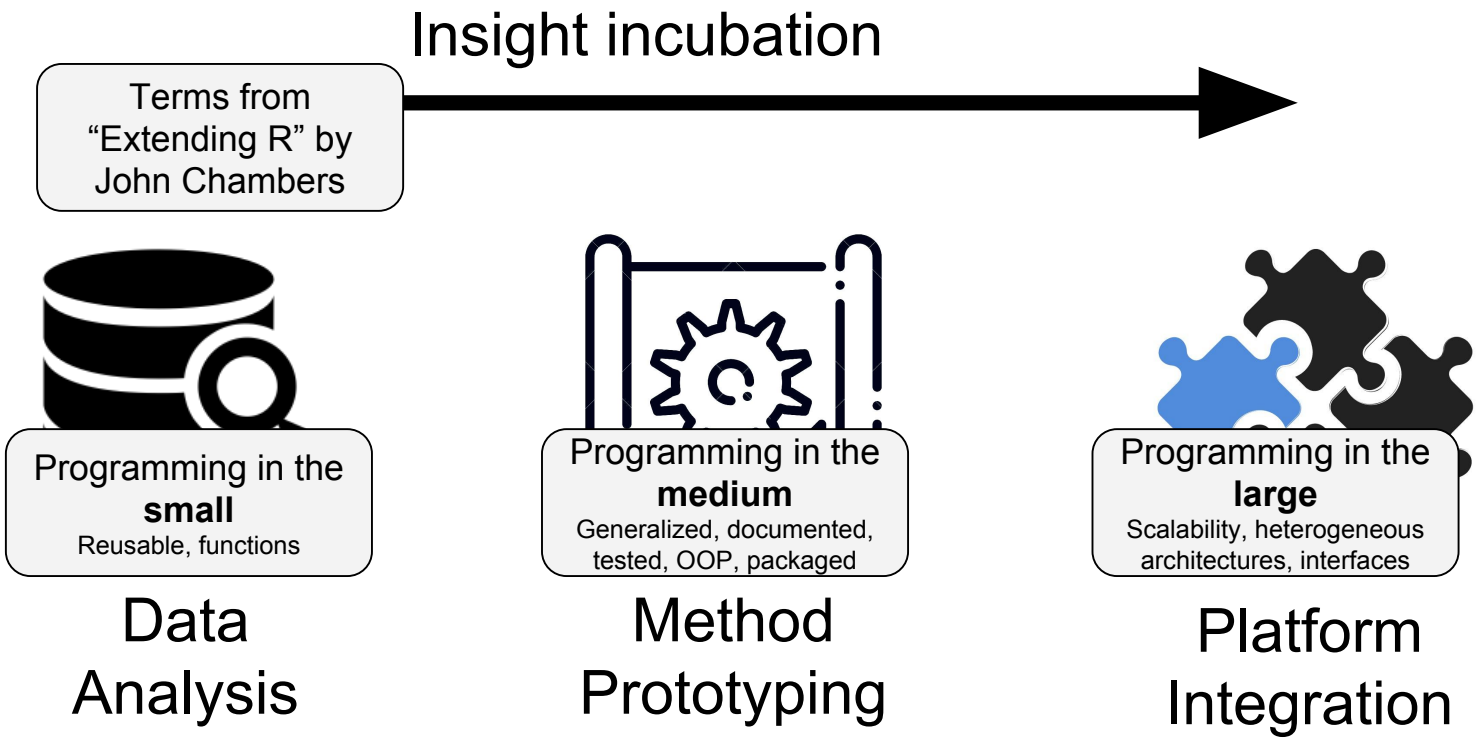


Roles are fluid and context-dependent

Enabling insight incubation



Programming at different scales



Challenges to *scientific* programming in-the-large

Integration	of independently developed modules into a platform on top of shared infrastructure
Translation	of analyses and prototypes to software, based on transitable interfaces
Scalability	through object-oriented abstractions

Outline

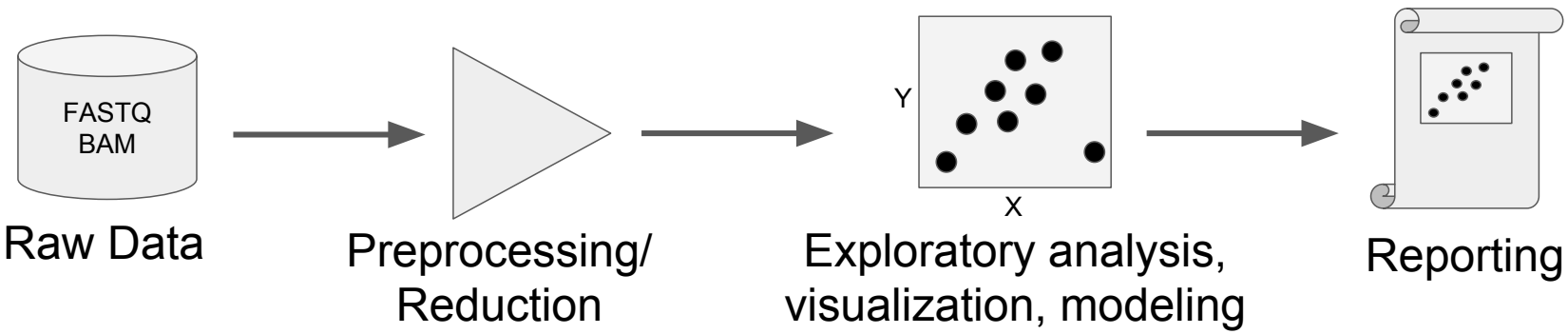
Introduction

Bioconductor as a software distillery

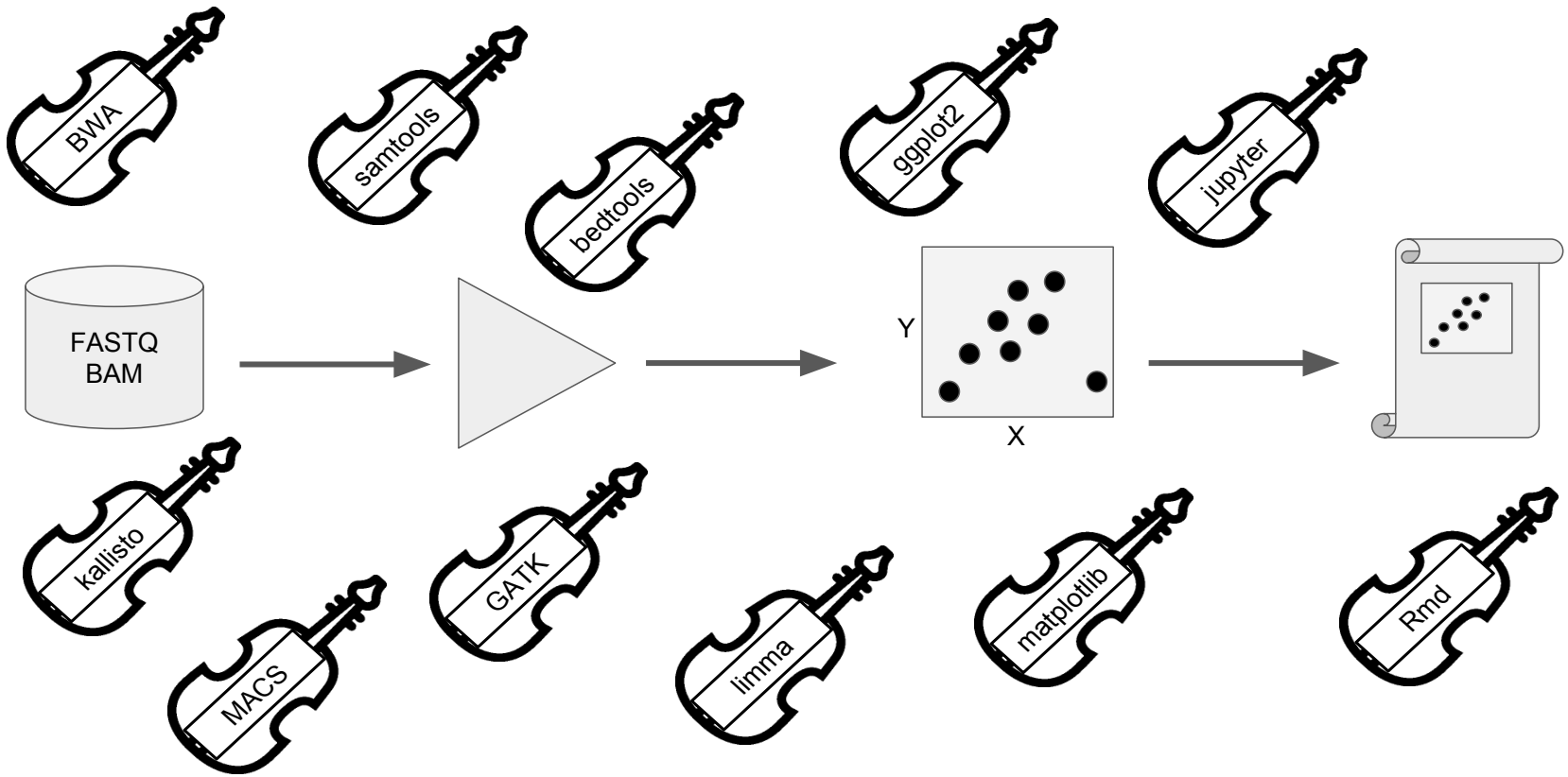
The plyranges package as a catalyst of Bioconductor

Scalability through deferred evaluation and the hailr package

Genomics workflows rely on a multitude of tools



Genomics workflows rely on a multitude of tools



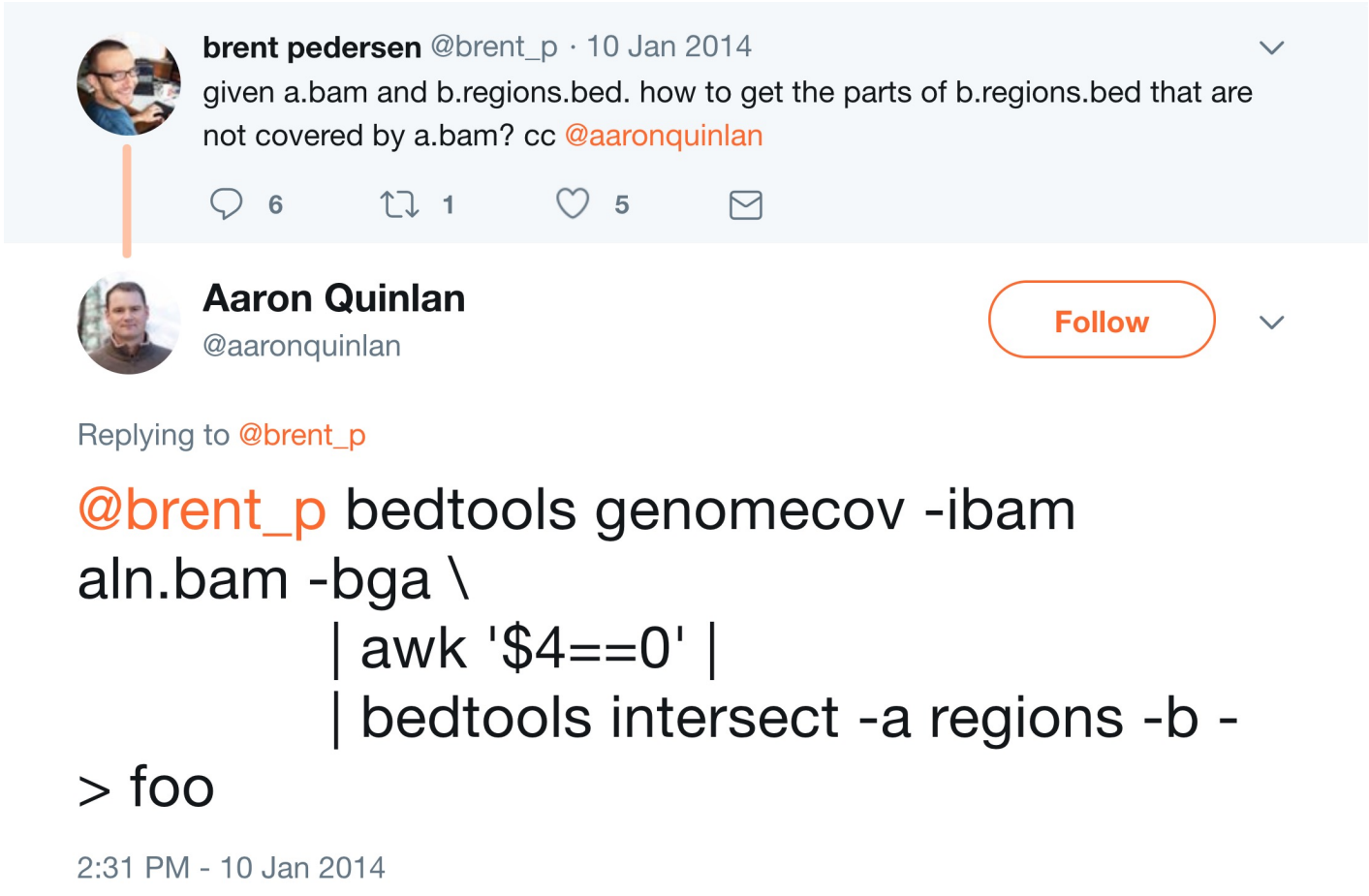
Tweet-size example from bedtools tutorial

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

6 1 5

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

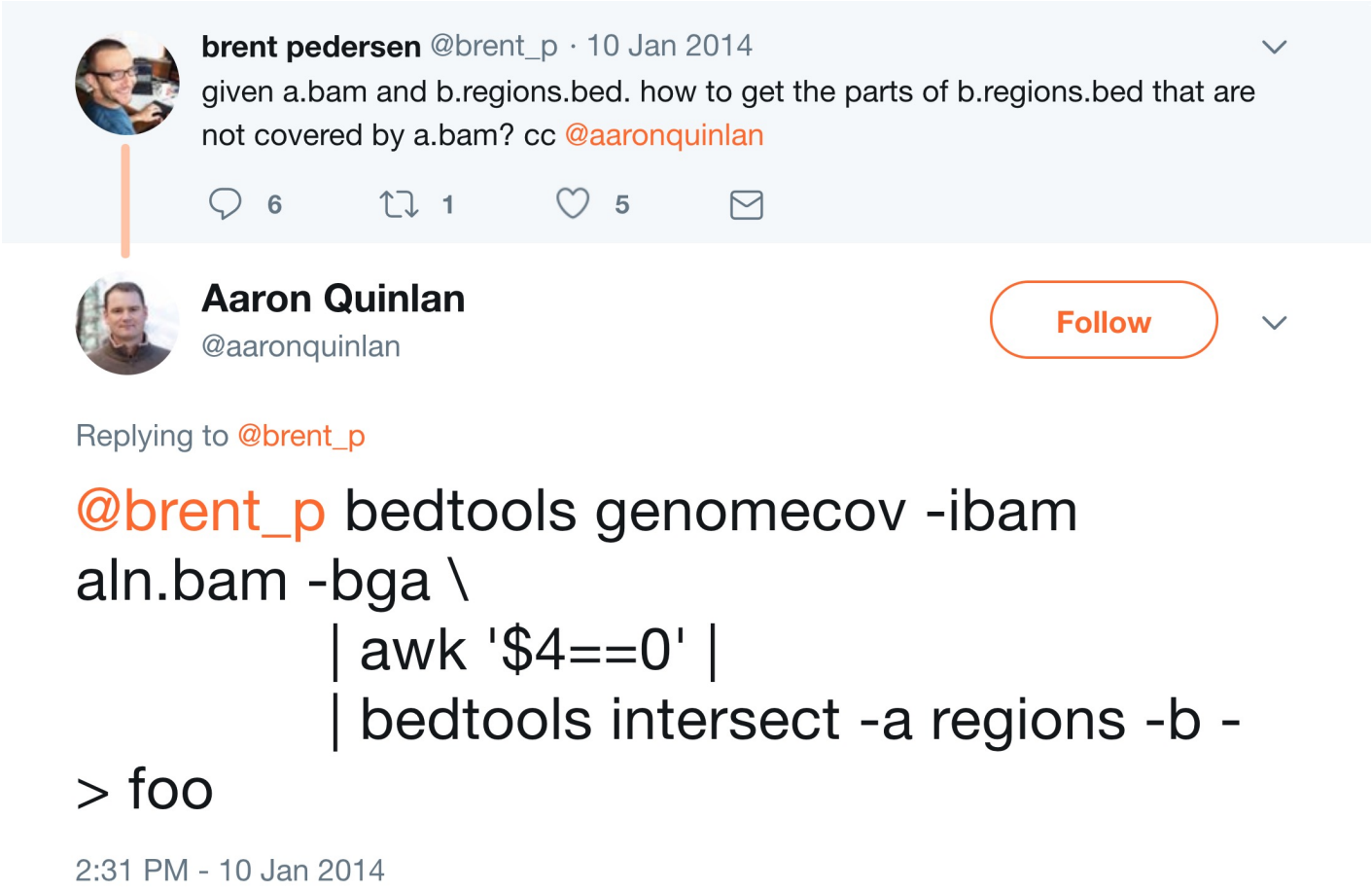
Follow

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

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

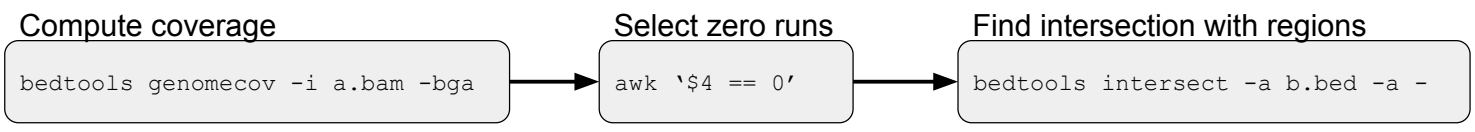
Aaron Quinlan
@aaronquinlan

Follow

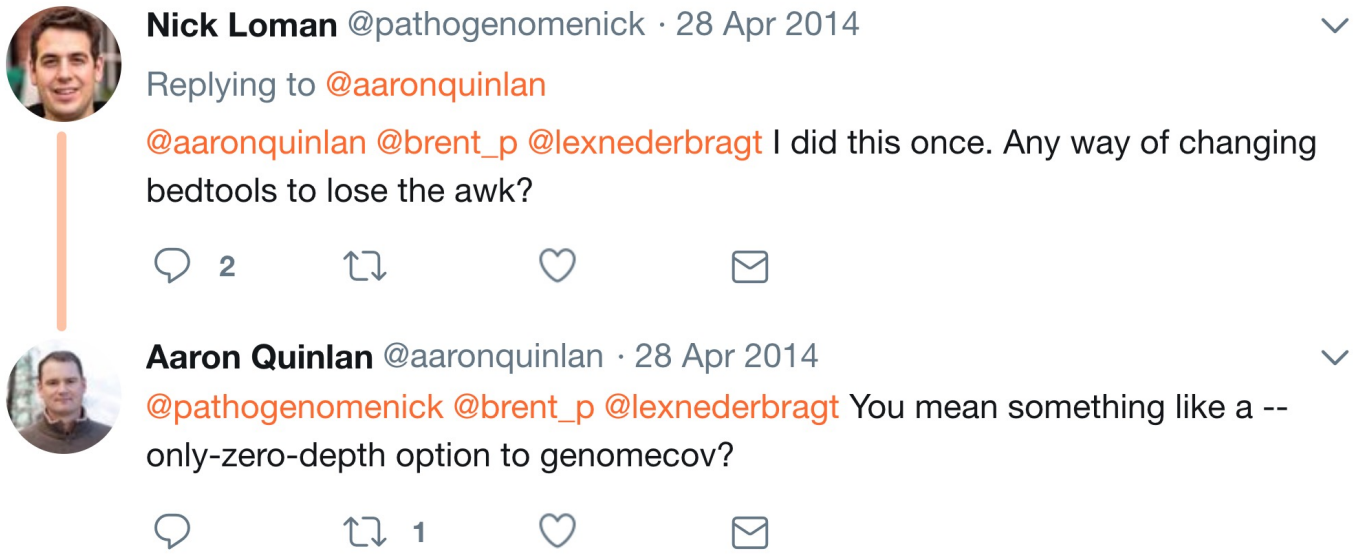
Replying to @brent_p

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

2:31 PM - 10 Jan 2014



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?

2

Aaron Quinlan @aaronquinlan · 28 Apr 2014

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

1

Compute coverage

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

→

Select zero runs

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

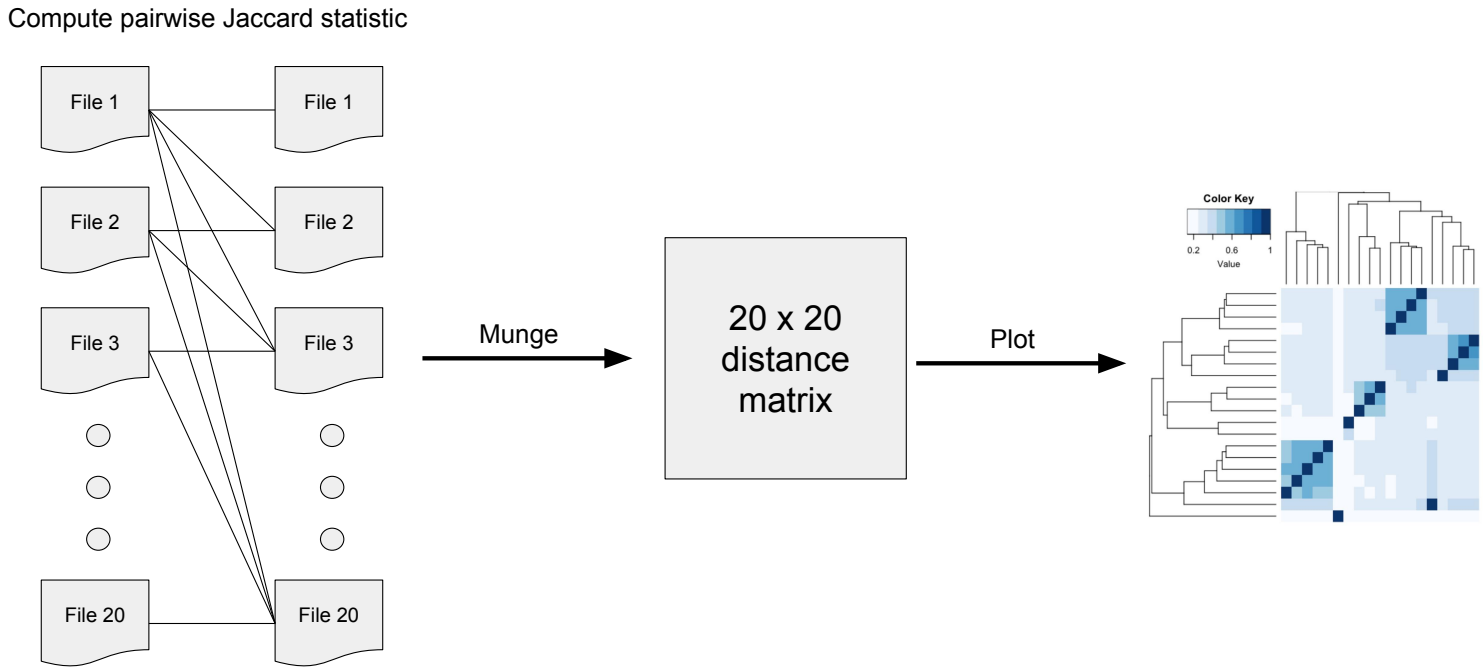
→

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

Side-effects

- 400 `.jaccard`

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`
```

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

Reshape into matrix

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

Side-effects

- 400 `.jaccard`
- `pairwise.txt`
- `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")
```

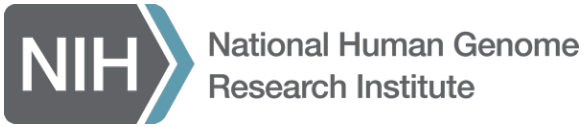
Typical obstacles in implementing genomic data analyses

- 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

Bioconductor

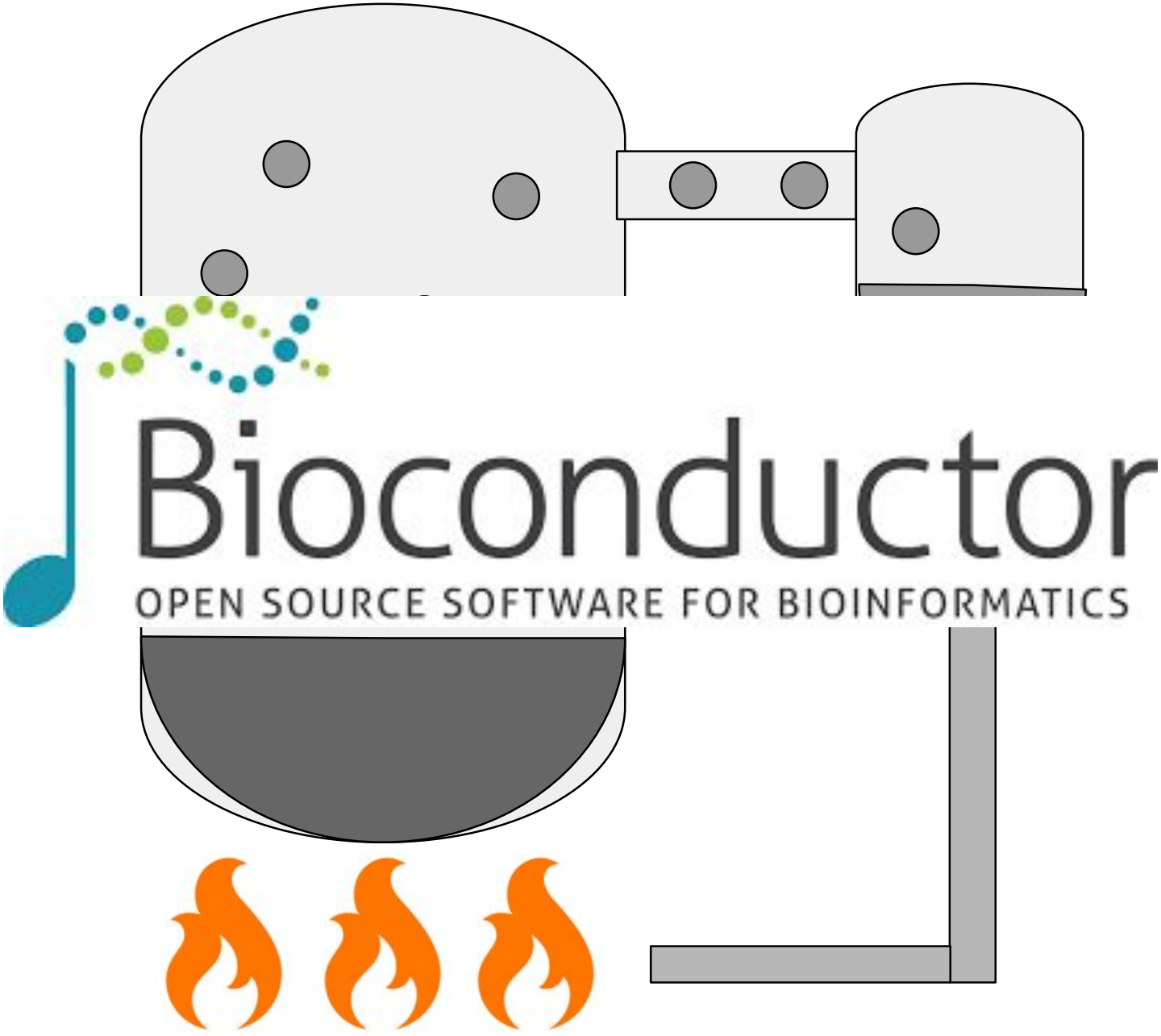
A unified platform for the analysis and comprehension of high-throughput genomic data.

- Started 2002
- Led by Martin Morgan
- Core infrastructure maintained by about 8 people, based in Roswell Park CRC in Buffalo, NY
- >1500 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



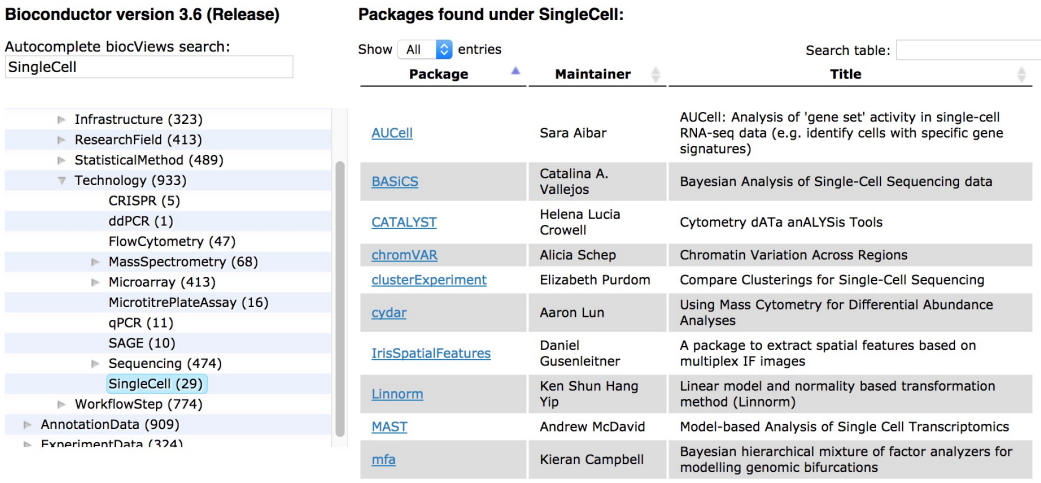
SOUND

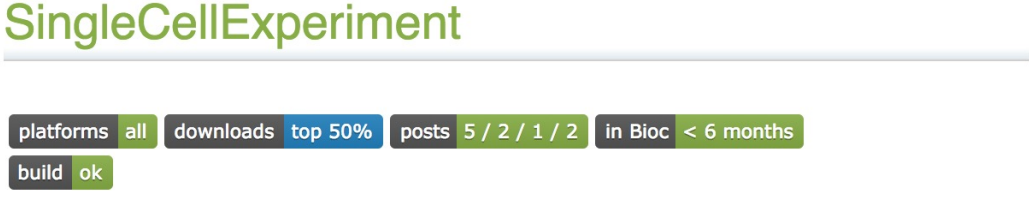
Bioconductor distills the cacophony to a symphony



Bioconductor qualities

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





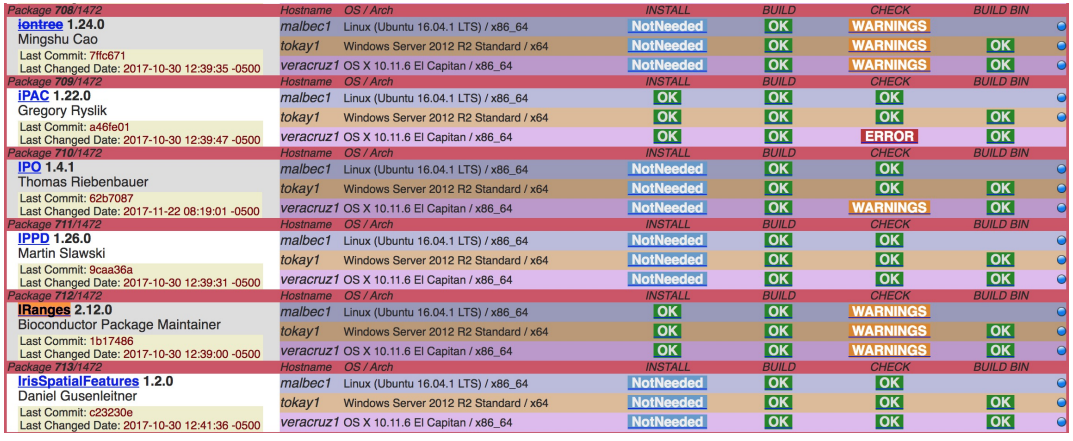
Bioconductor qualities

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

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

Bioconductor qualities

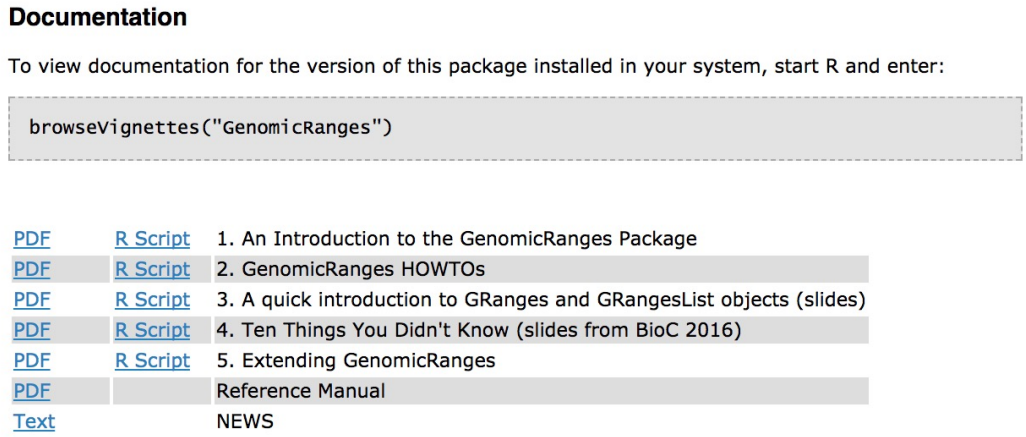
- Discoverable
- Installable
- Reliable
- Documented
- Supported
- Integrated
- Scalable
- State of the art
- Community-driven



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

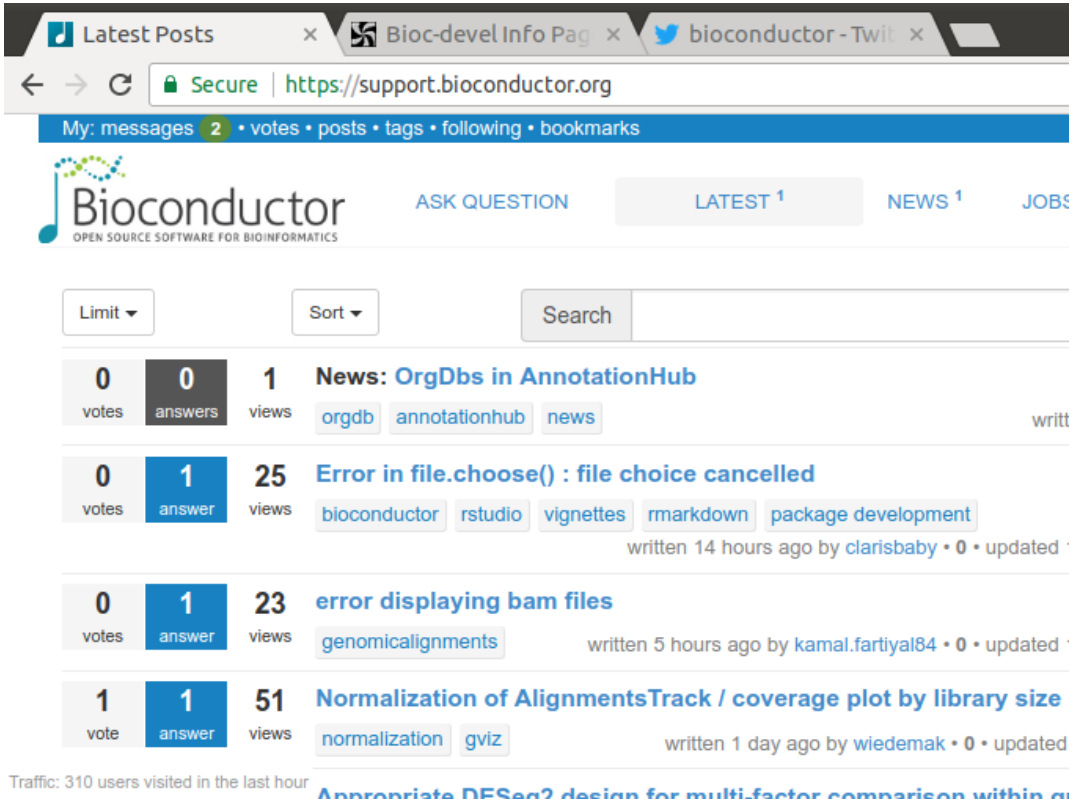
To view documentation for the version of this package installed in your system, start R and enter:

```
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

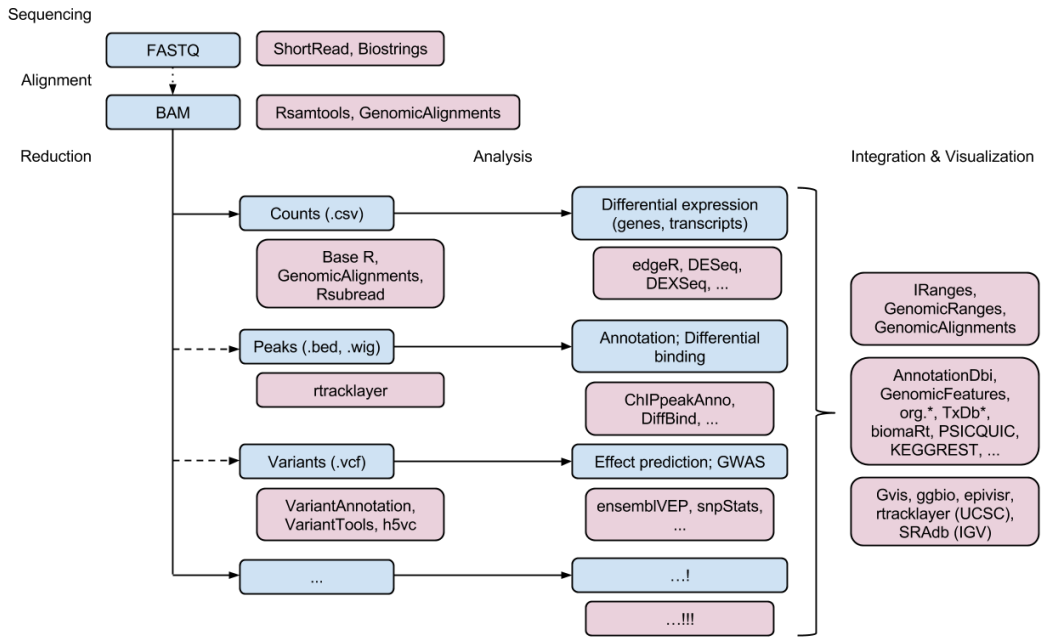
Bioconductor qualities

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



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

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

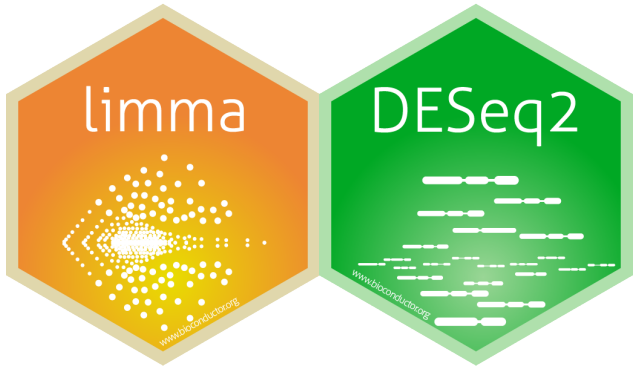
```
## class: SingleCellExperiment
## dim: 27998 1306127
## metadata(0):
## assays(1): counts
## rownames: NULL
## rowData names(2): Ensembl Symbol
## colnames(1306127): AAACCTGAGATAGGAG-1 AAACCTGAGCGGCTTC-1 ...
##   TTTGTCAGTTAAAGTG-133 TTTGTCATCTGAAAGA-133
## colData names(4): Barcode Sequence Library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

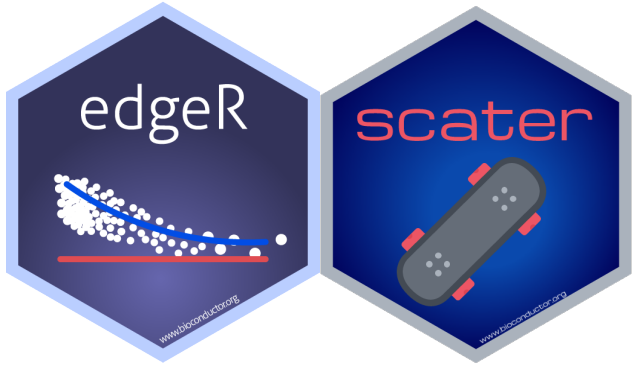
```
libSize <- colSums(assay(se)[, 1:1000])
range(libSize)
```

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

Bioconductor qualities

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





Bioconductor qualities

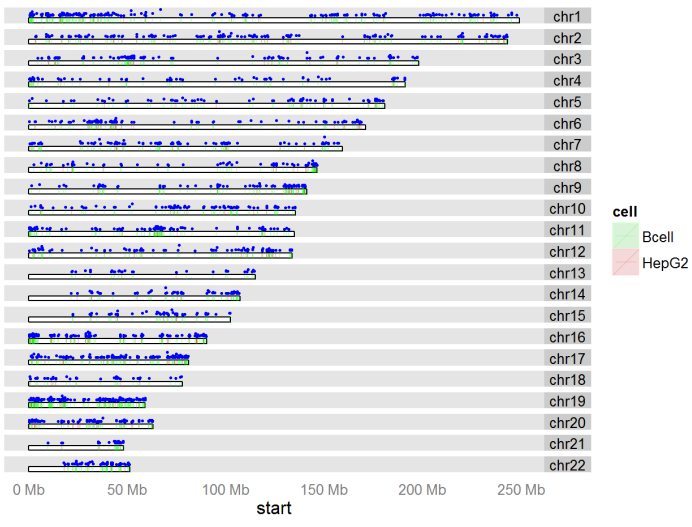
- Discoverable
- Installable
- Reliable
- Documented
- Supported
- Integrated
- Scalable
- State of the art
- Community-driven

- 1064 unique package maintainers
- Web users by country:

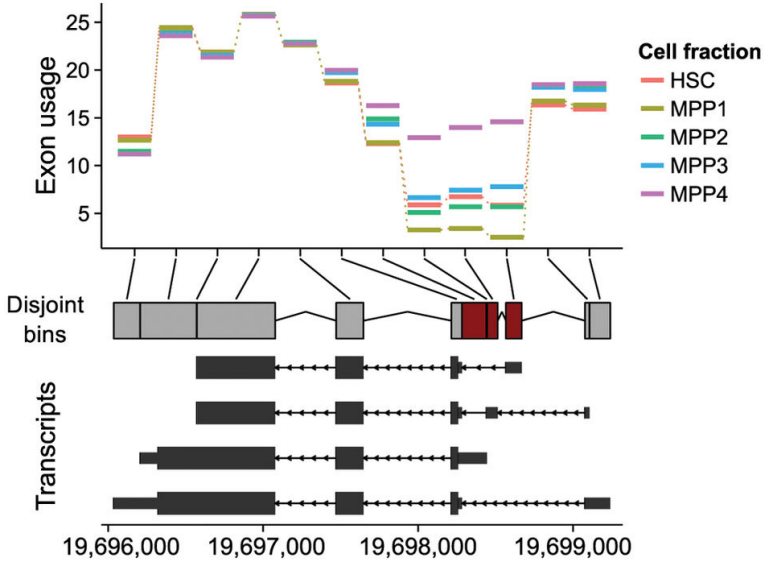
1.	United States	**58,384** (32.78%)
2.	China	**20,910** (11.74%)
3.	United Kingdom	**12,265** (6.89%)
4.	Germany	**10,024** (5.63%)
5.	France	**5,536** (3.11%)
6.	Canada	**4,999** (2.81%)
7.	Spain	**4,864** (2.73%)
8.	Japan	**4,539** (2.55%)
9.	India	**4,397** (2.47%)
10.	Australia	**4,043** (2.27%)

Central data structures of Bioconductor

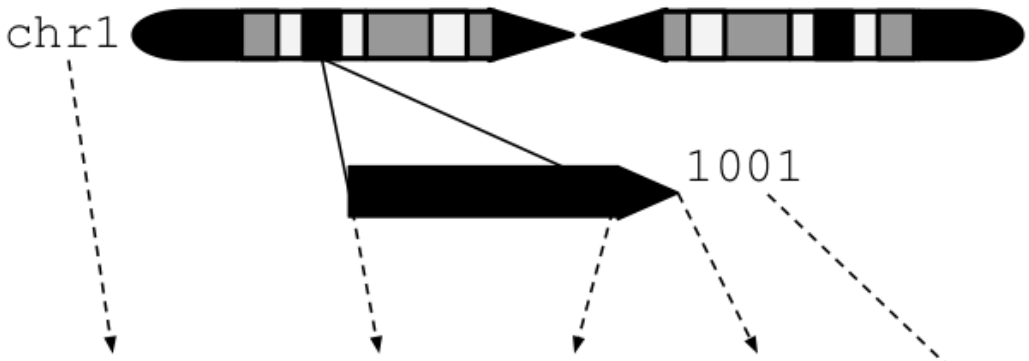
Data on genomic ranges



Summarized data

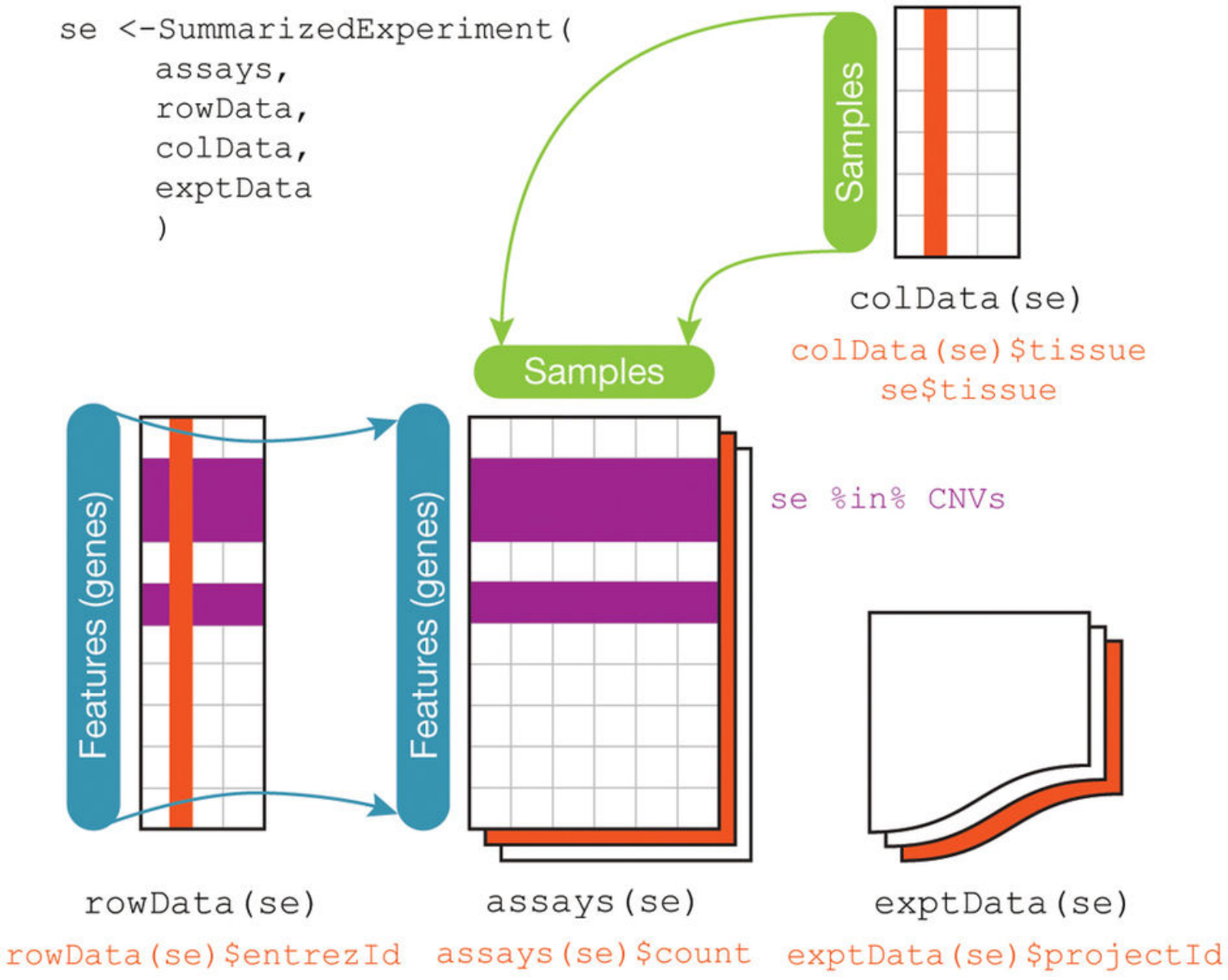


GRanges: data on genomic ranges

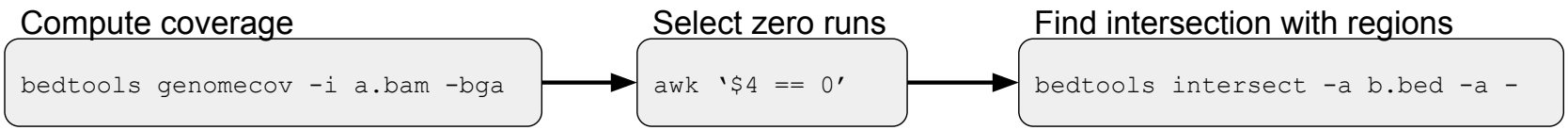


seqname	start	end	strand	gene_id	score
chr1	102012	120303	+	1001	10
chr1	520211	526211	-	2151	25
...	...	...	...	...	...

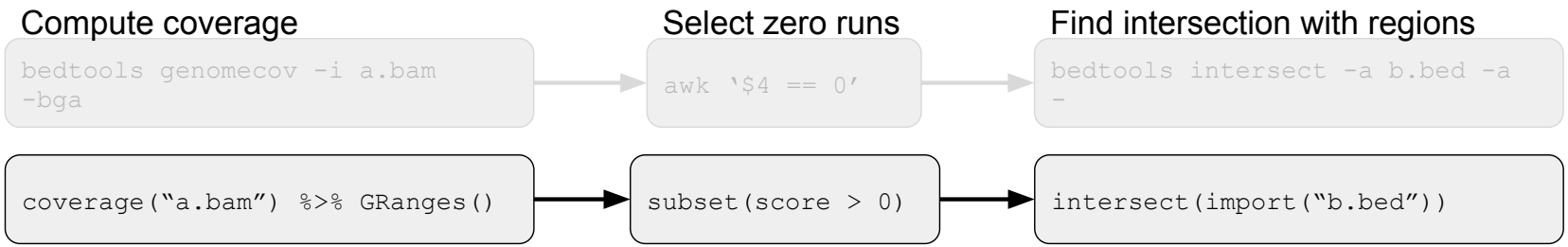
SummarizedExperiment: the central data model



Bioconducting the tweeted workflow



Bioconducting the tweeted workflow



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

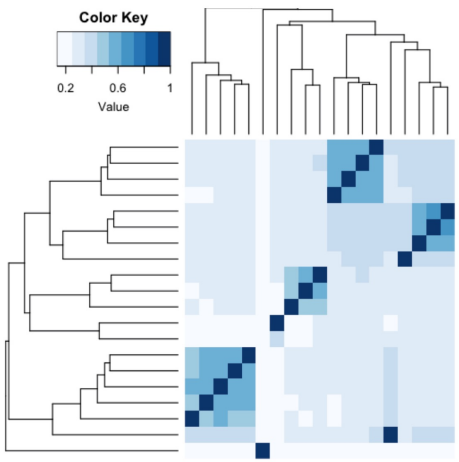
Compute the statistics in parallel

```
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"))
```



Outline

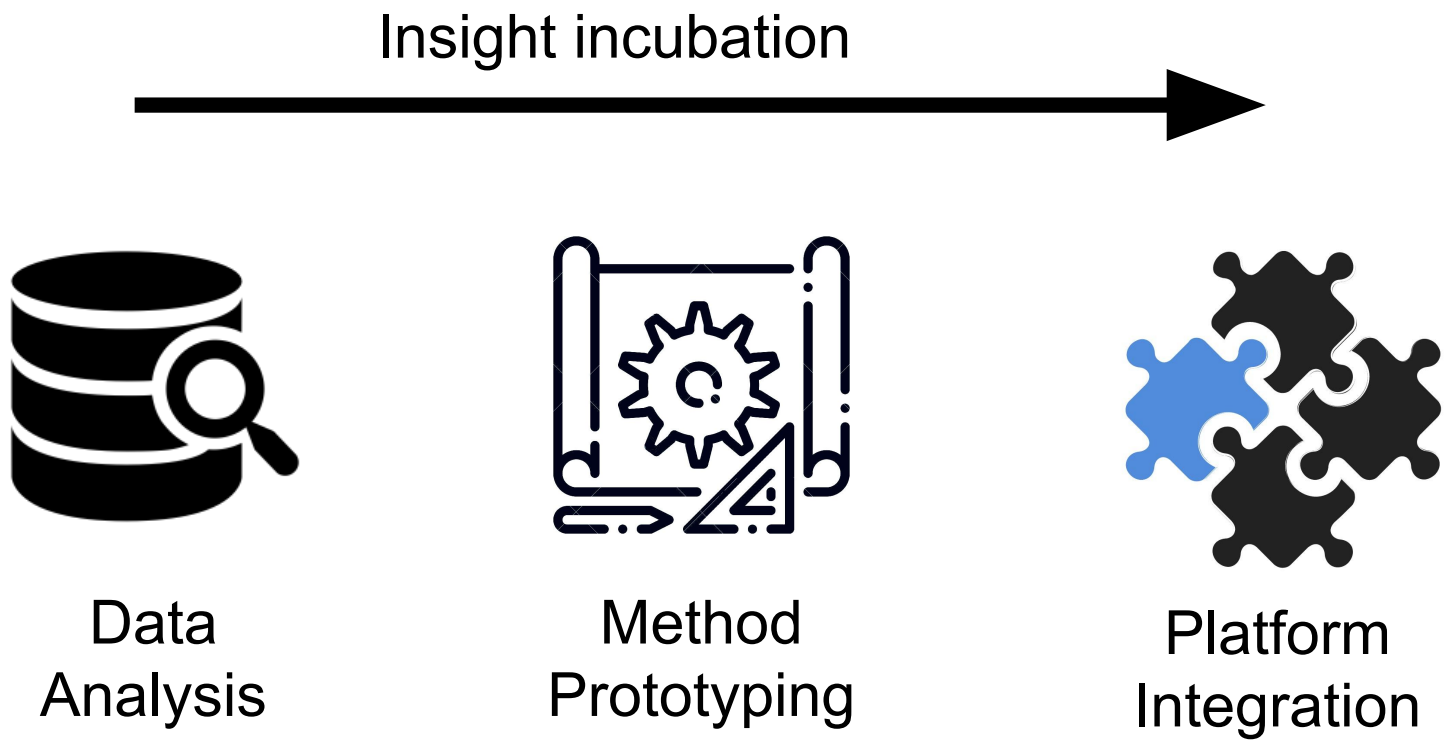
Introduction

Bioconductor as a software distillery

The plyranges package as a catalyst of Bioconductor

Scalability through deferred evaluation and the hailr package

The Ranges infrastructure is an incubator



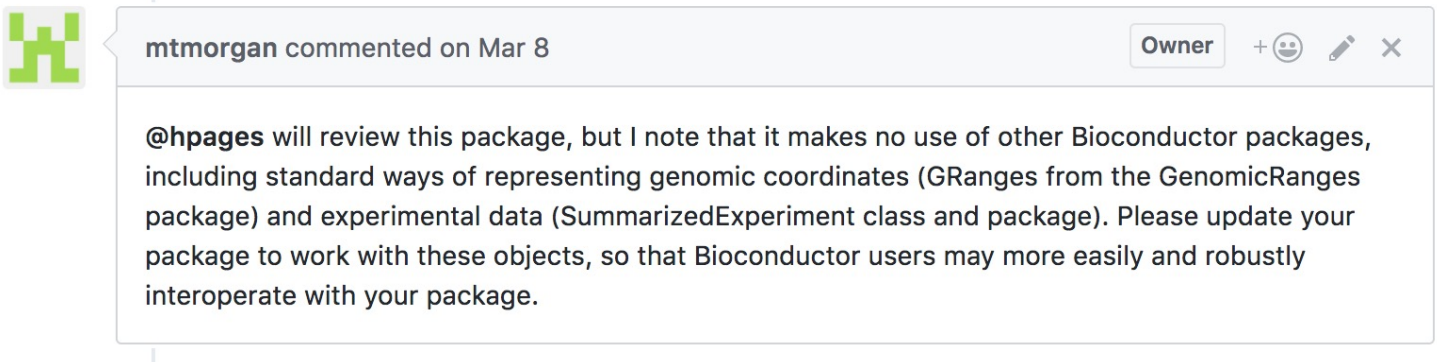
- Should be accessible to the average Bioconductor user

Is the transition happening?

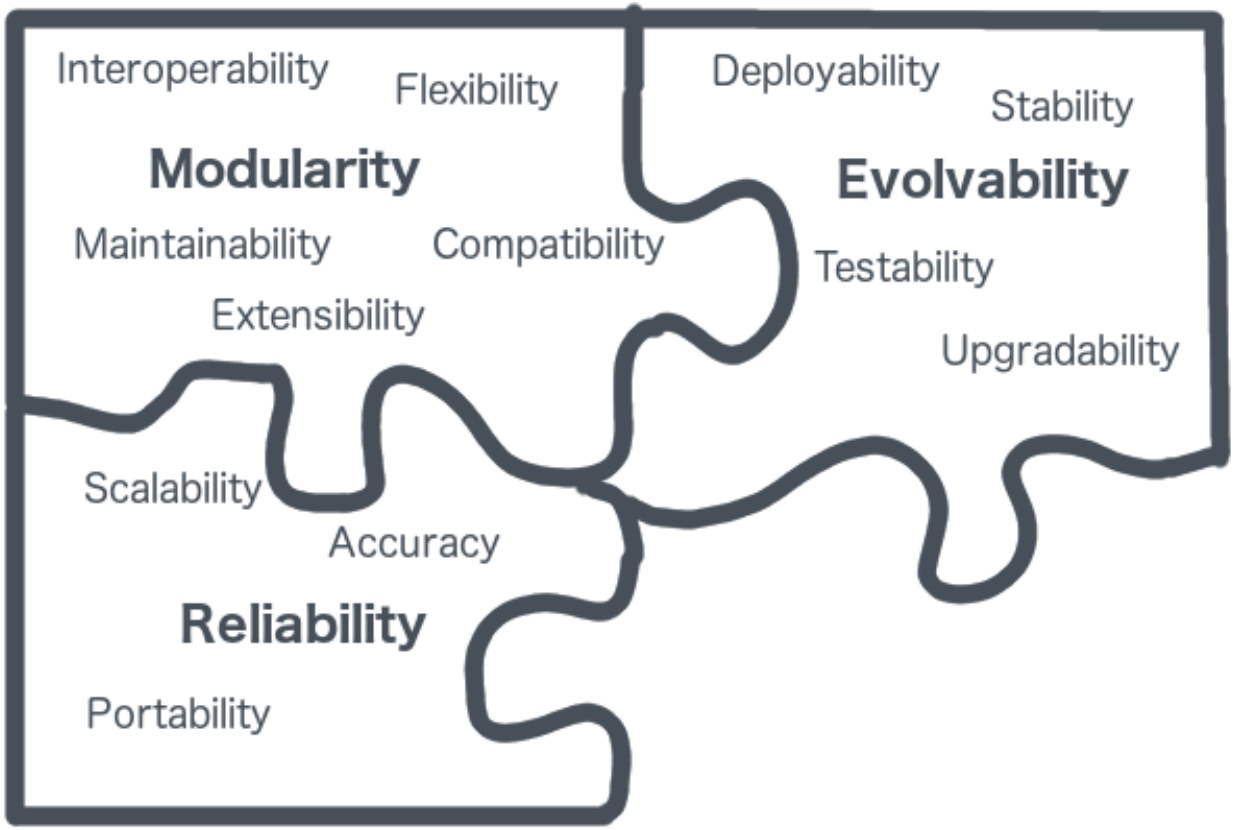
- From a typical package submission:

  `Imports: checkmate, dplyr, ggplot2, tidyr`

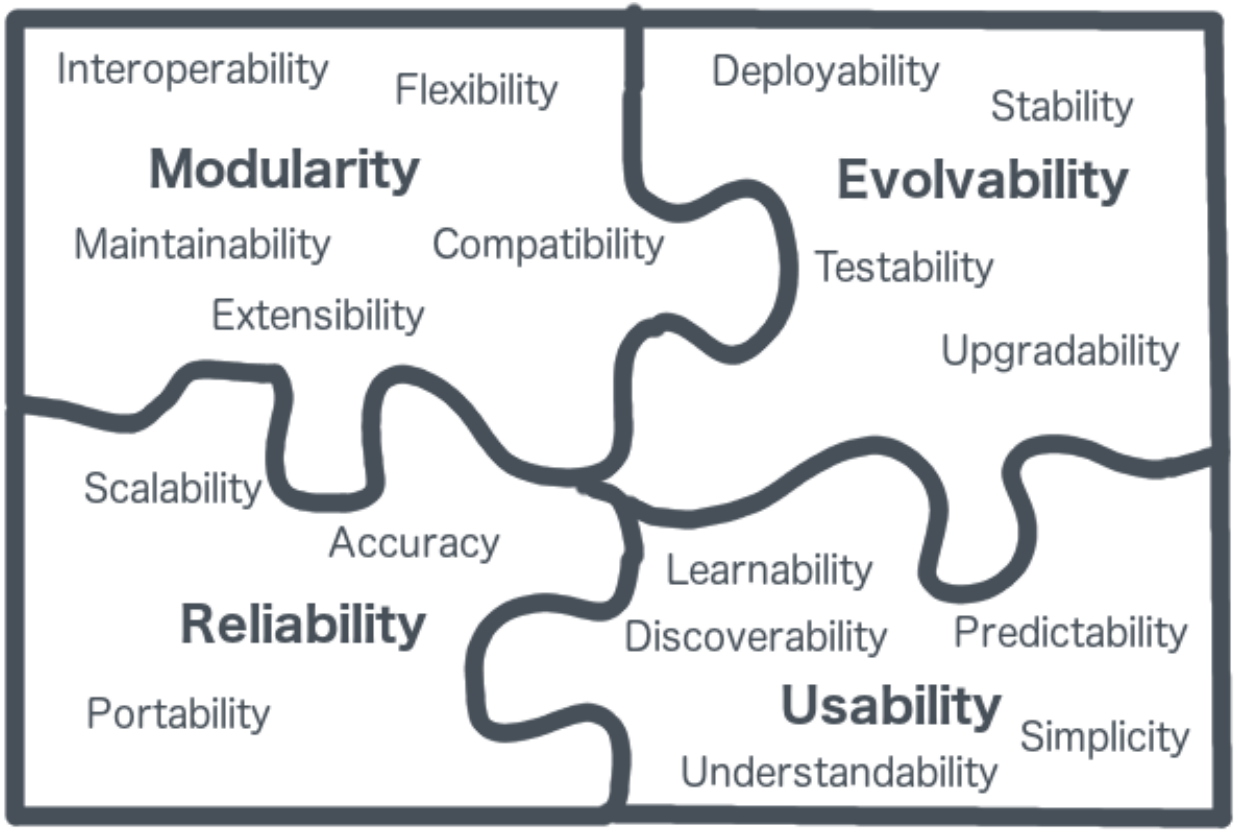
- A typical initial response:



Aspects of software quality: the ilities



Aspects of software quality: the ilities



Bioconductor is complex

```
pkgs_to_get_started <-
  c("S4Vectors", "IRanges", "GenomicRanges")
pkg_classes <- function(.)
  methods::getClasses(asNamespace(.))

n_classes <- sum(lengths(lapply(pkgs_to_get_started,
                                pkg_classes)))
n_classes
```

```
143
```

```
n_methods <- length(methods(class = "Ranges"))
n_methods
```

```
28
```

Taking cues from the dplyr package

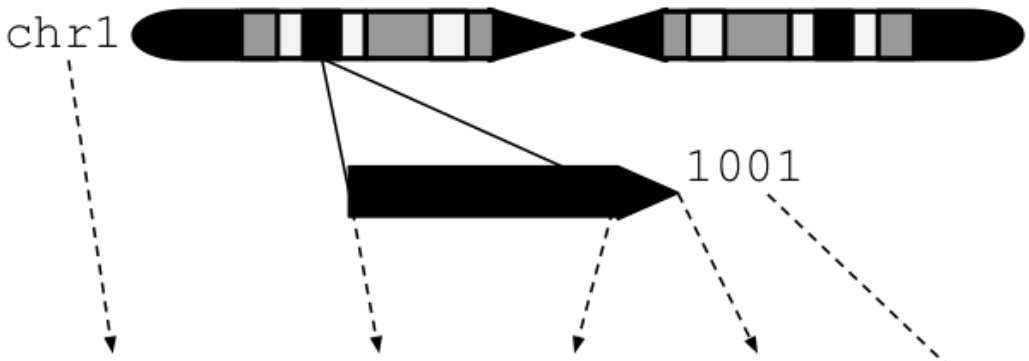
- dplyr is a API for tabular data manipulation
- Inspired by relational algebra, SQL
- Unified about a single, data model: the tibble
- Operations are:
  - Cohesive (do a single thing)
  - Endomorphic (return the same type as their input)
  - Verb-oriented in syntax
- Fluency emerges from chaining of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

GRanges are tidy!



seqname	start	end	strand	gene_id	score
chr1	102012	120303	+	1001	10
chr1	520211	526211	-	2151	25
...	...	...	...	...	...

plyranges

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

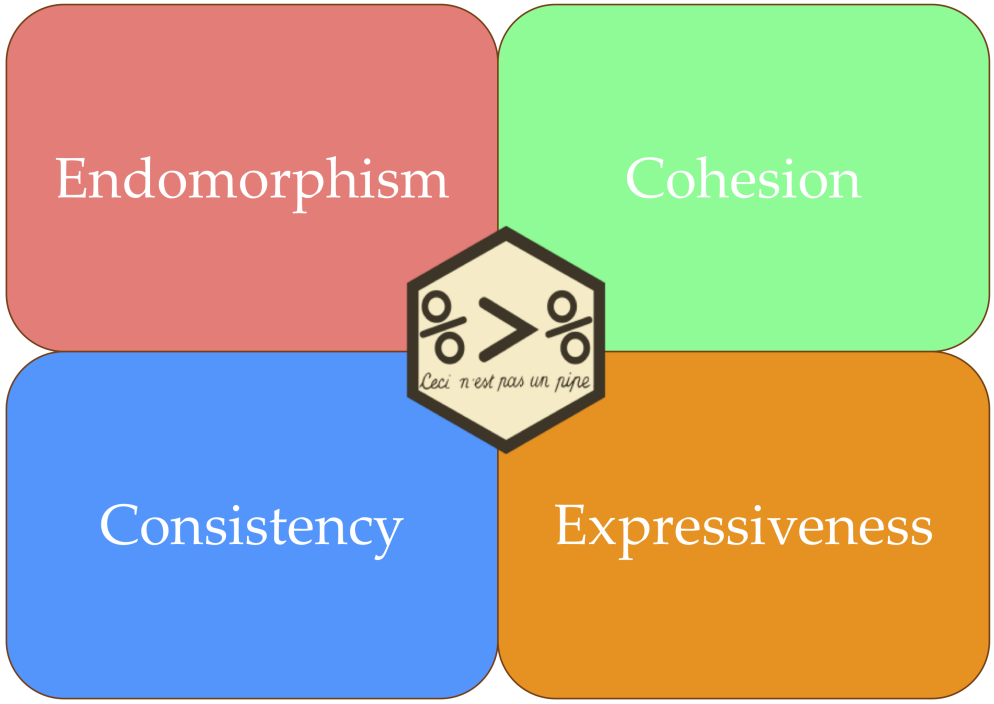
- A dplyr-based API for computing on genomic ranges
- Extending the relational algebra with genomic notions
- Large set of visible verbs acting only on the core data structures:

  GRanges represents annotated genomic ranges

  SummarizedExperiment coordinates experimental assay data with sample and feature annotations

- Collaboration with Stuart Lee and Di Cook @ Monash

Designing a grammar



Genomic semantics on common operations

Arithmetic	mutating/shifting/re-sizing/flanking/coverage
Restriction	filtering by metadata or ranges
Aggregation	summarizing over groups/overlaps/unions
Merging	combining ranges based on overlaps/nearest neighbors

Verbs are explicit about genomic features and their intentions

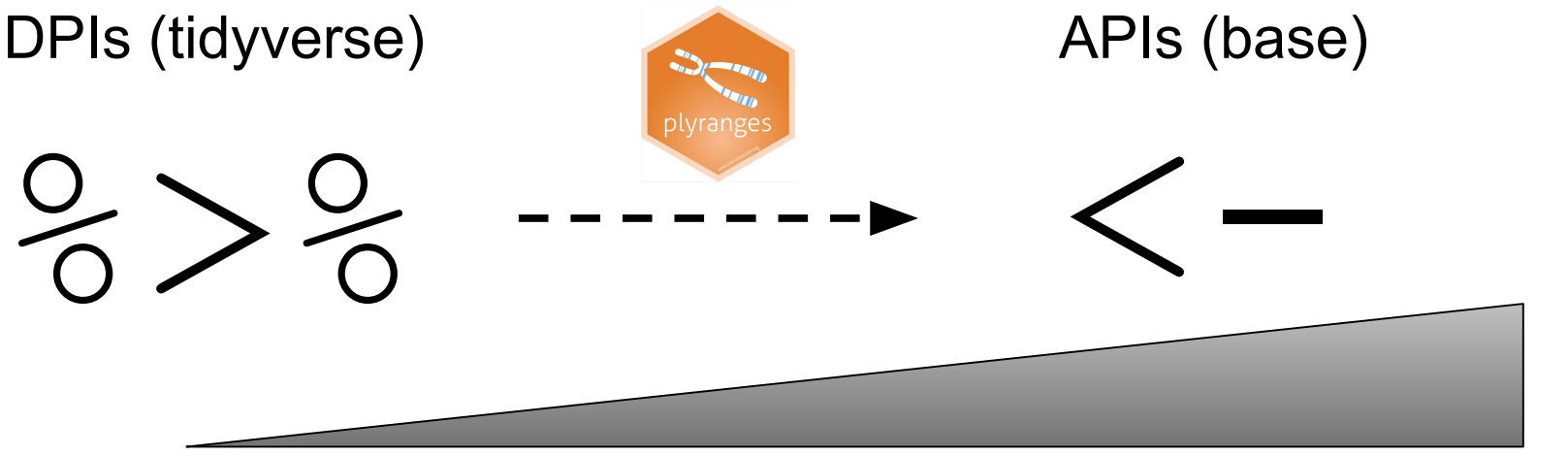
```
exons %>%
  flank_downstream(2L)
exons %>%
  anchor_3p() %>%
  mutate(width = 2*width)
exons %>%
  shift_upstream(10L)
```

Merging ranges through overlap joins

- Reimagine overlap/nearest neighbour operations as table joins
- Carry over metadata
- Flatten API via function calls

```
join_overlap_inner(a, b)
join_overlap_inner_within(a, b)
join_overlap_inner_directed(a, b)
join_overlap_intersect(a, b)
join_overlap_left(a, b)
```

Formal data structures enable interface fluidity



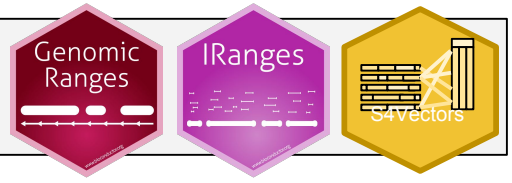
Programming in the **small**
Reusable, functions

Programming in the **medium**
Generalized, documented, tested, OOP, packaged

Programming in the **large**
Scalability, heterogeneous architectures, interfaces

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

Interoperable data structures



Outline

Introduction

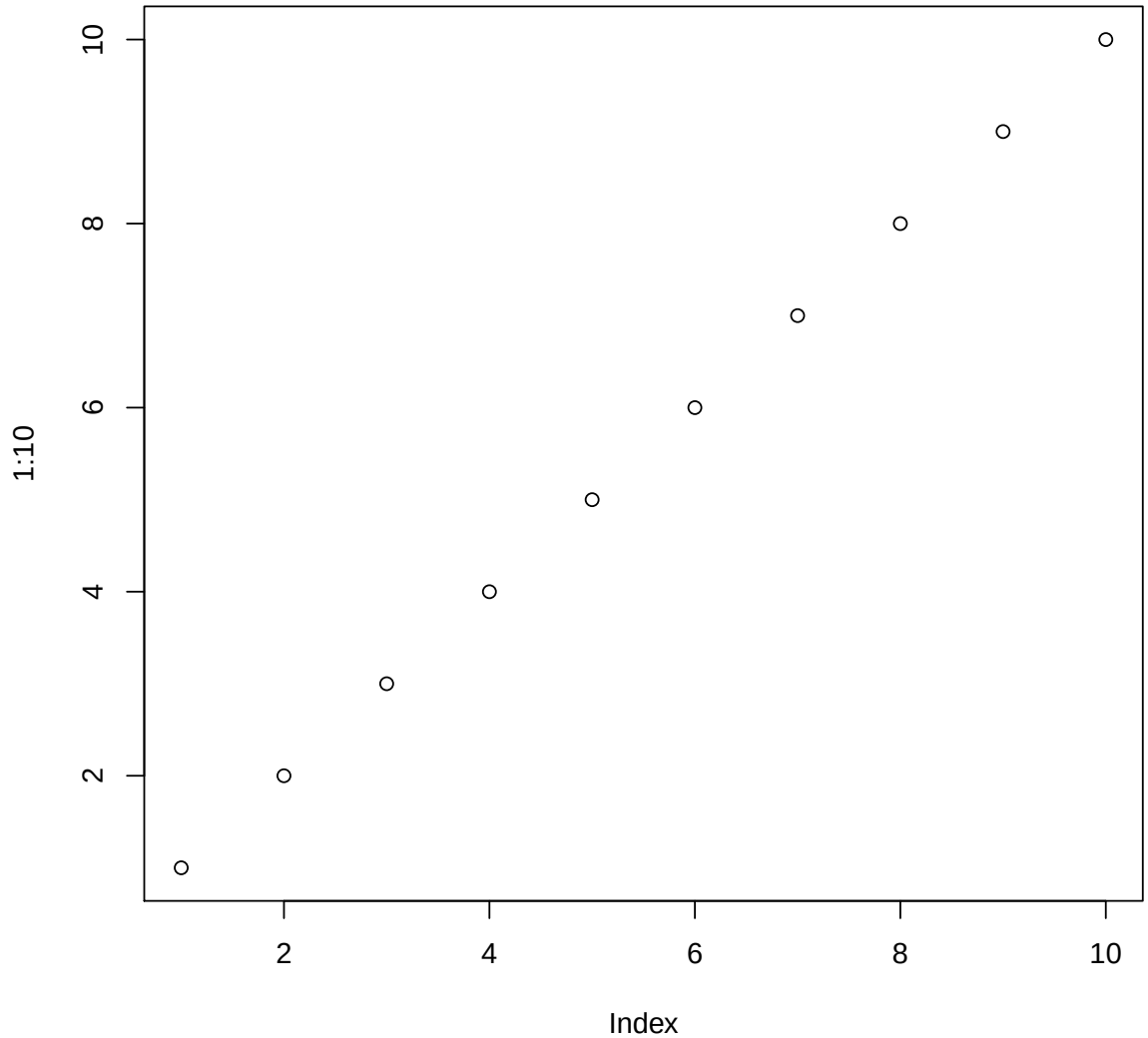
Bioconductor as a software distillery

The plyranges package as a catalyst of Bioconductor

Scalability through deferred evaluation and the hailr package

R's magical axis labels

```
plot(1:10)
```



Dispelling the magic

```
fun <- function(arg) substitute(arg)
fun(1:10)
```

```
1:10
```

Lazy evaluation

- Delay the evaluation of an expression until its value becomes necessary

```
fun <- function(arg) {
    z <- arg
    substitute(z)
}
fun(1:10)
```

```
[1]  1  2  3  4  5  6  7  8  9 10
```

Deferred data structures

Strategic laziness, eager evaluation

For some promise "`x`":

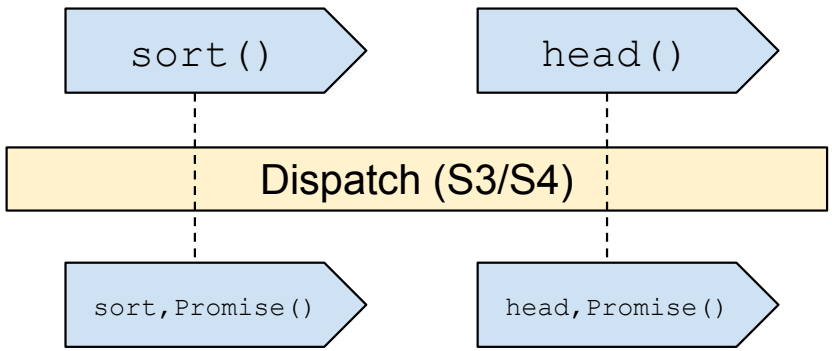
```
> head(sort(x))
```

Deferred data structures

Strategic laziness, eager evaluation

For some promise "`x`":

```
> head(sort(x))
```

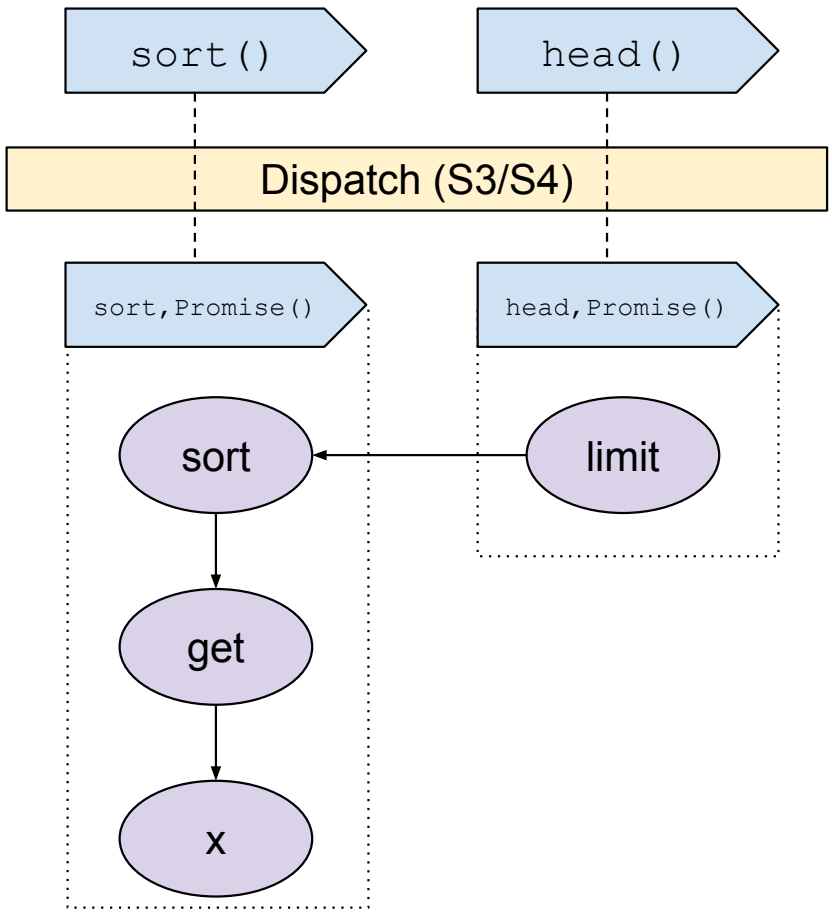


Deferred data structures

Strategic laziness, eager evaluation

For some promise "`x`":

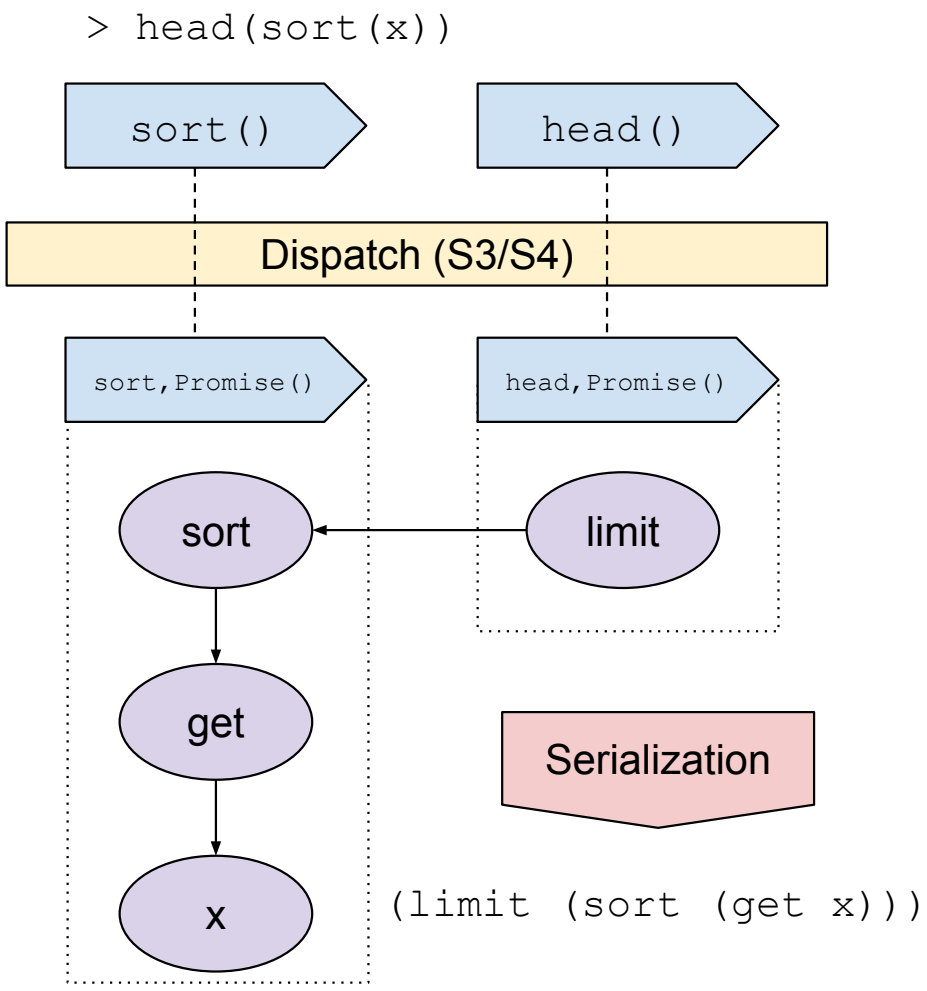
```
> head(sort(x))
```



Deferred data structures

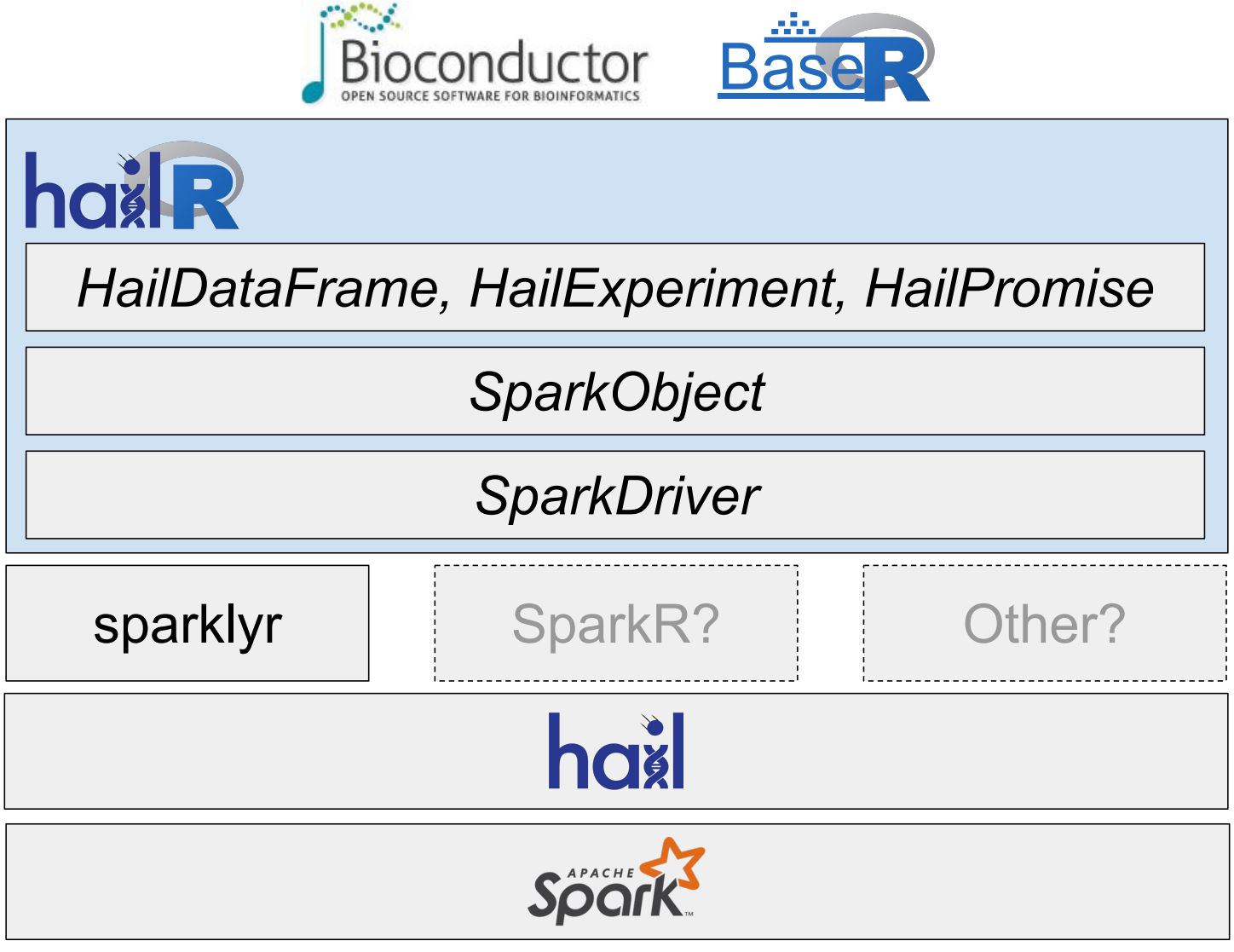
Strategic laziness, eager evaluation

For some promise "`x`":



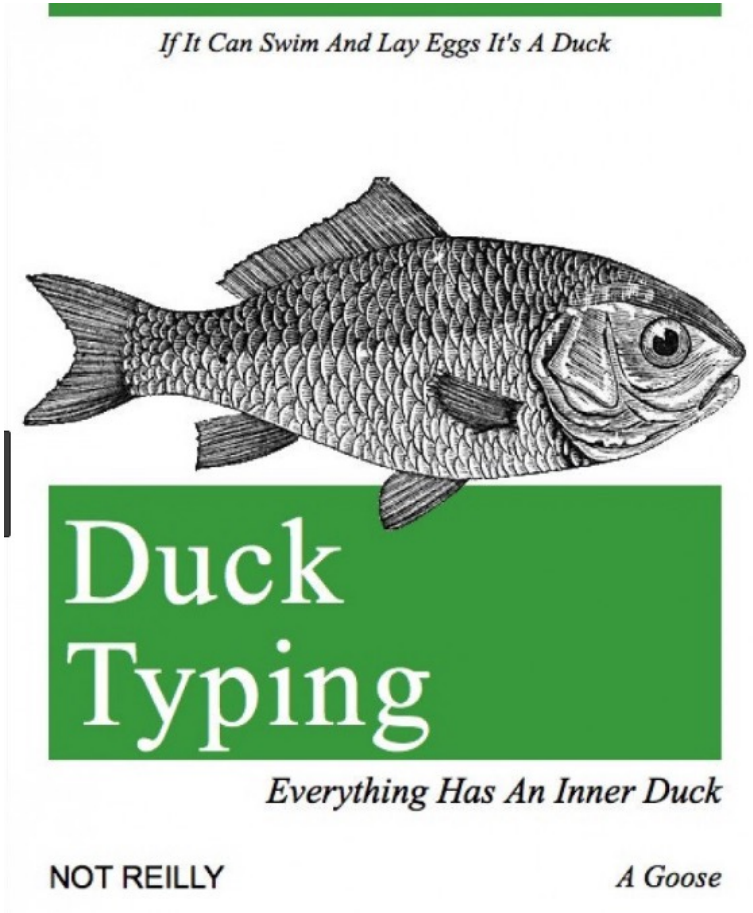
- A platform for distributed genomics on Apache Spark
- Initially aimed at genetics but becoming more general
- Defines MatrixTable, an analog of SummarizedExperiment
  - Stored with efficient parquet-based storage format (VDS)
  - Represented outside of Java heap (Java Unsafe) for performance and interoperability
- Defines its own byte code targeted by Python and now R
  - Filtering, transformation, aggregation, joins of matrix data and tabular metadata
  - Implemented in C++ where beneficial via Java Unsafe

The hailr package

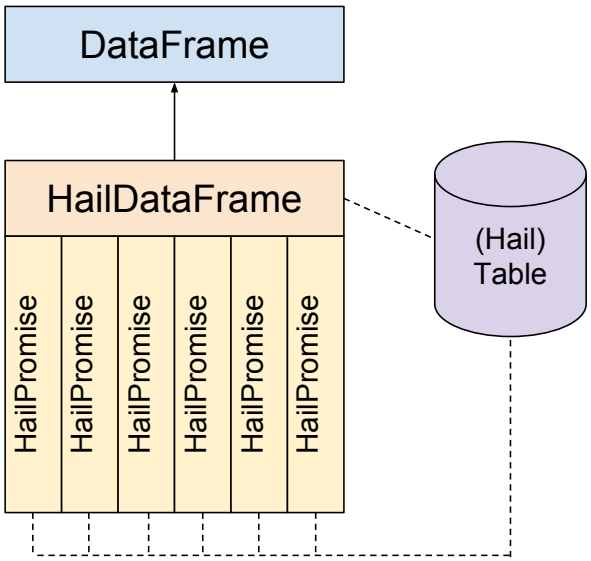


Bioconductor containers are generic

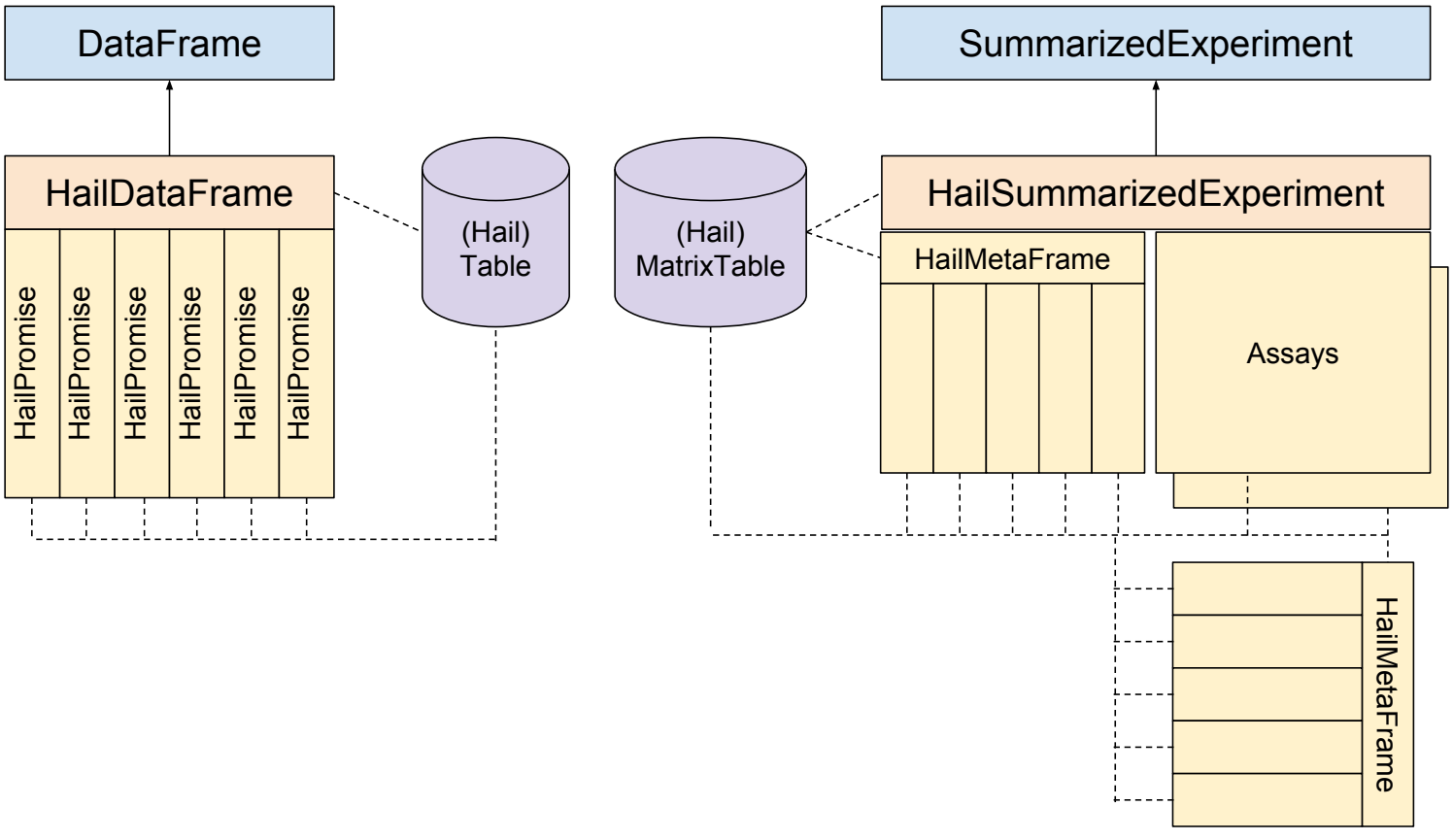
- Bioconductor containers assume elements implement key functions from the base API
  - *DataFrame* allows anything "vector-like" to be a column
  - *SummarizedExperiment* allows anything "matrix-like" to hold assay values
- Since our promises implement the base API, they just work
- But we still want to map DataFrame operations to Hail Table operations



Hierarchical extension of Bioconductor



Hierarchical extension of Bioconductor



Load data into Hail

Directly from a text file:

```
library(hailr)
data_dir <- system.file("extdata", package="hailr")
tsv1 <- file.path(data_dir, "kt_example1.tsv")
df <- readHailDataFrameFromText(tsv1, header=TRUE)
```

Copying from an R data.frame:

```
df <- copy(read.table(tsv1, header=TRUE), hail())
```

Get it back out

```
df
```

```
HailDataFrame with 4 rows and 8 columns
      ID             HT             SEX              X              Z
  <Int32Promise> <Int32Promise> <StringPromise> <Int32Promise> <Int32Promise>
1              1             65               M              5              4
2              2             72               M              6              3
3              3             70               F              7              3
4              4             60               F              8              2
      C1             C2             C3
  <Int32Promise> <Int32Promise> <Int32Promise>
1              2             50              5
2              2             61              1
3             10             81             -5
4             11             90            -10
```

```
df$ID
```

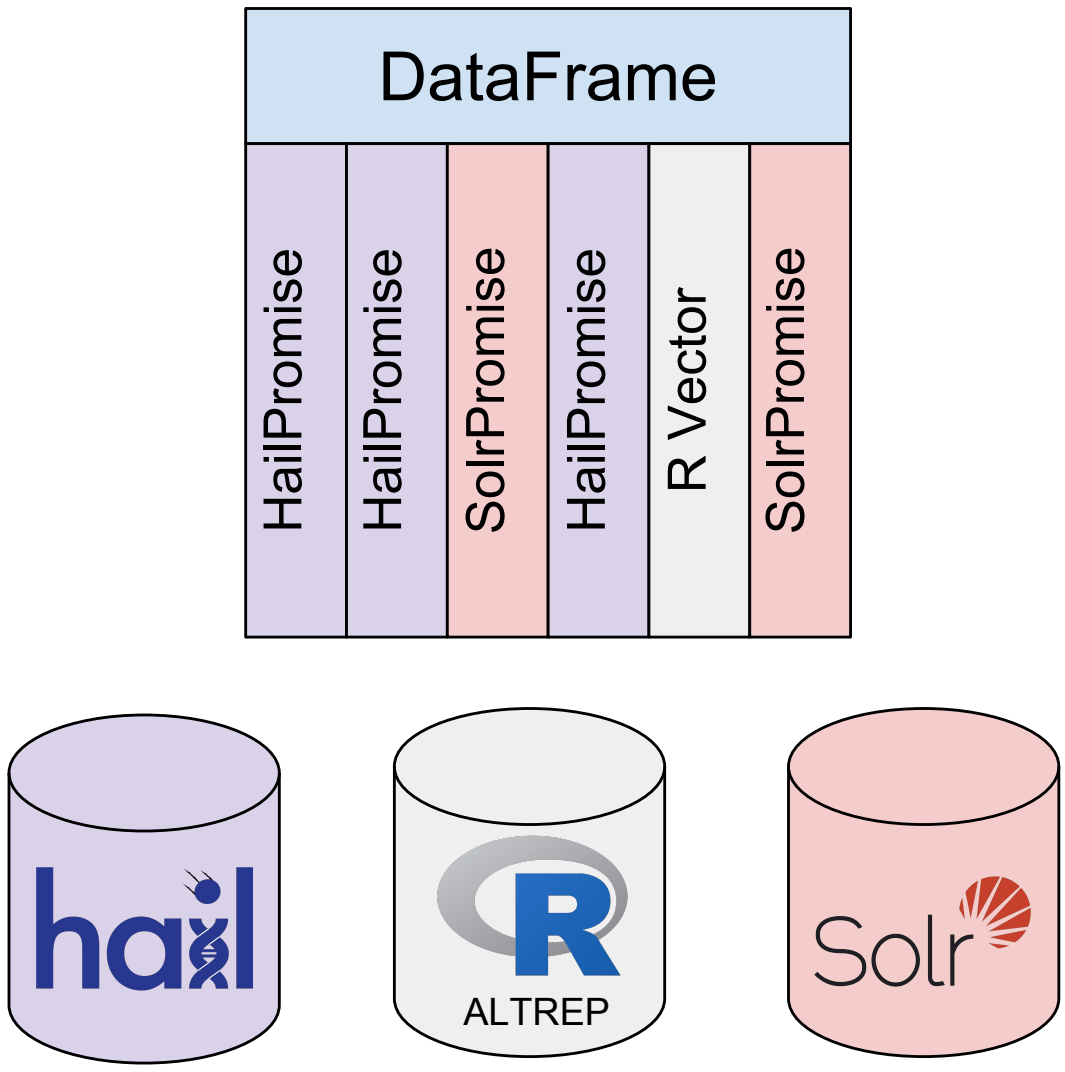
```
[1] 1 2 3 4
```

A glimpse into the compiler

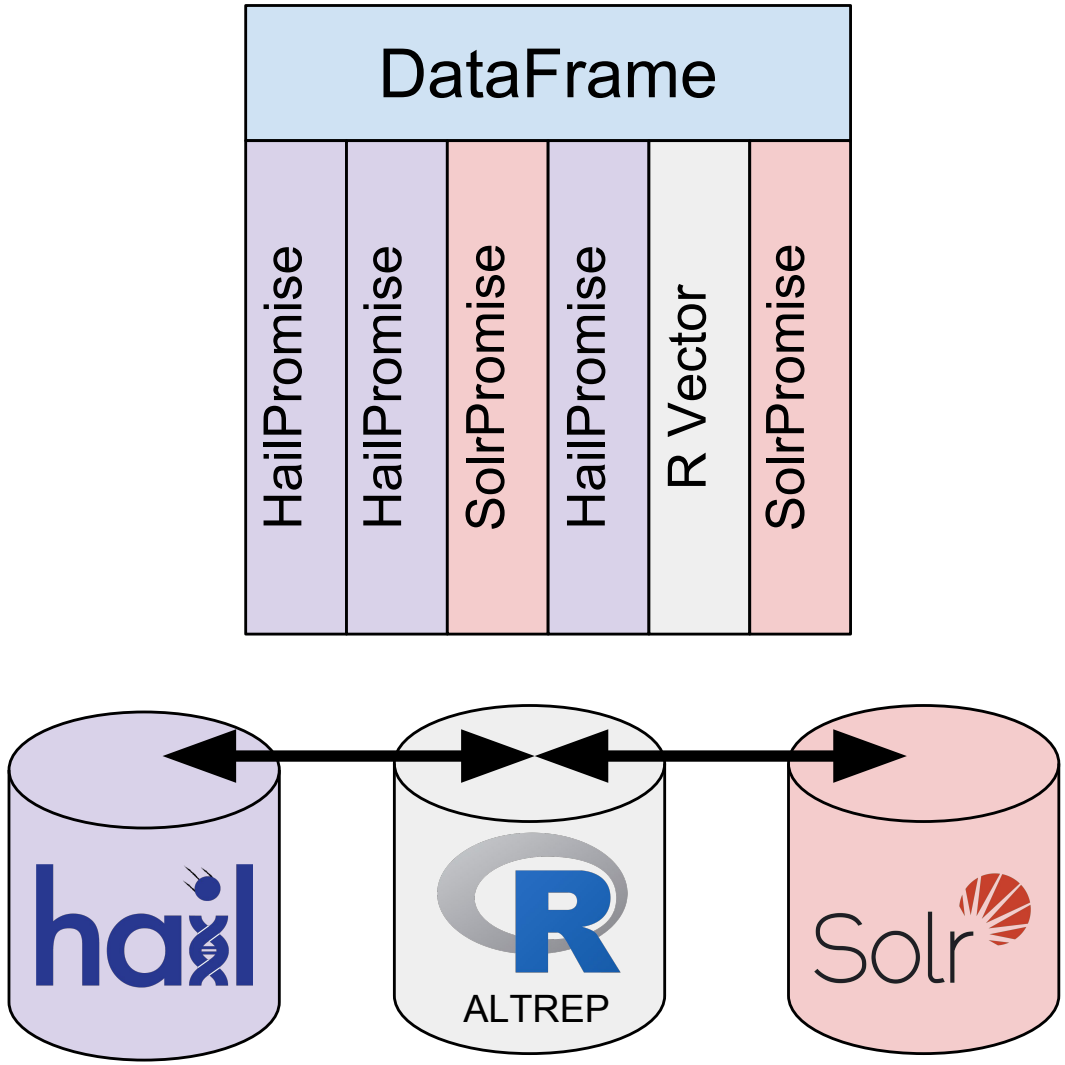
```
as.character(df$ID@expr)
```

```
[1] "(GetField ID (Ref row))"
```

Abstractions enable mixed evaluation



Abstractions enable mixed evaluation

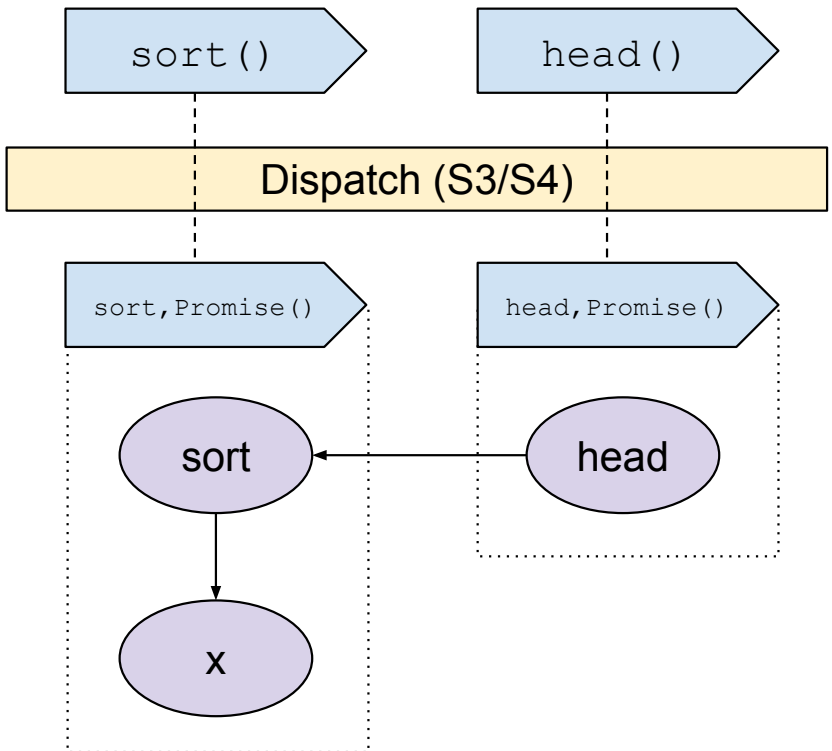


Looking forward: generalized, integrated compute

Intermediate algebra, optimization with backend-informed cost model

For some promise "x":

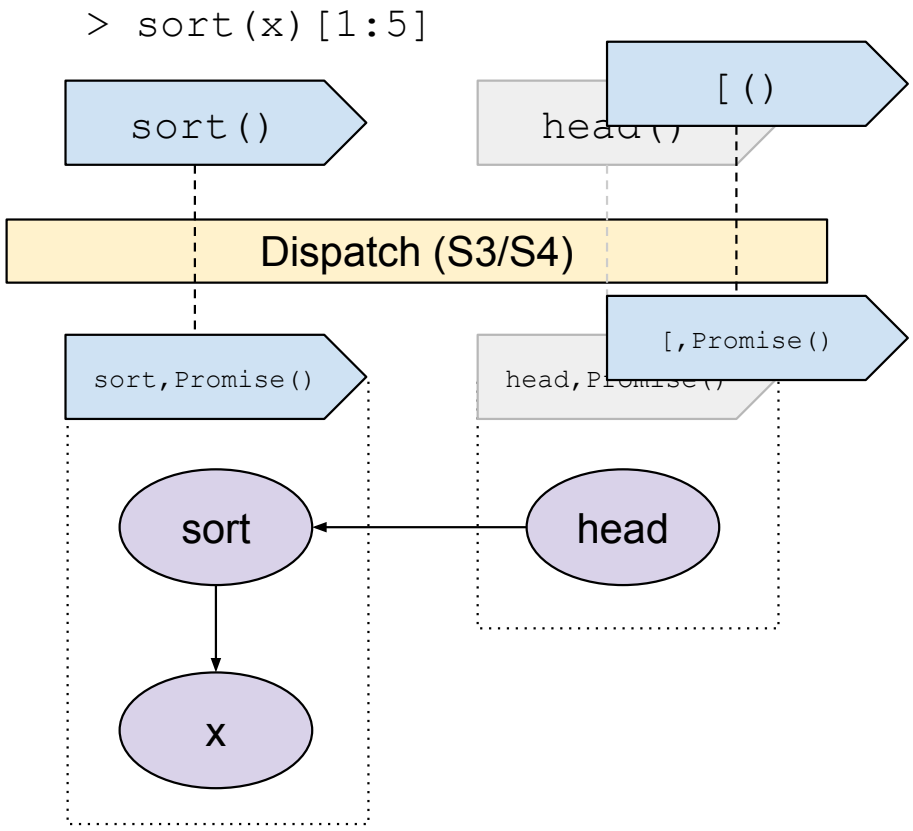
```
> head(sort(x))
```



Looking forward: generalized, integrated compute

Intermediate algebra, optimization with backend-informed cost model

For some promise "x":

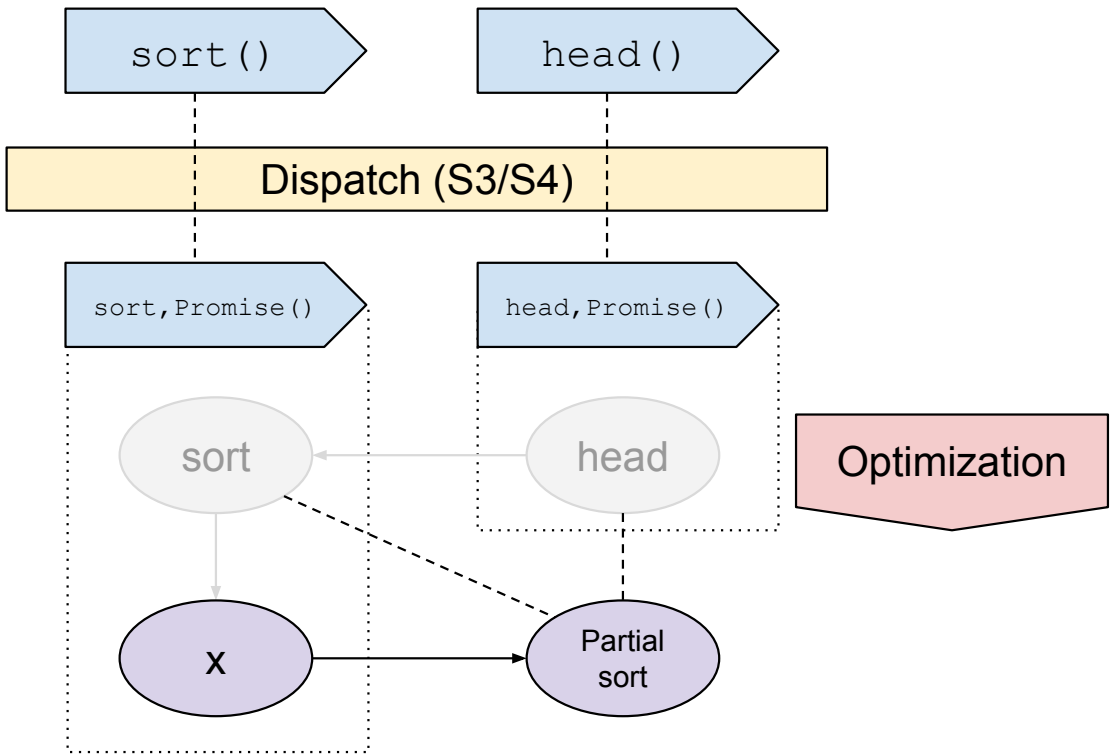


Looking forward: generalized, integrated compute

Intermediate algebra, optimization with backend-informed cost model

For some promise "x":

```
> head(sort(x))
```

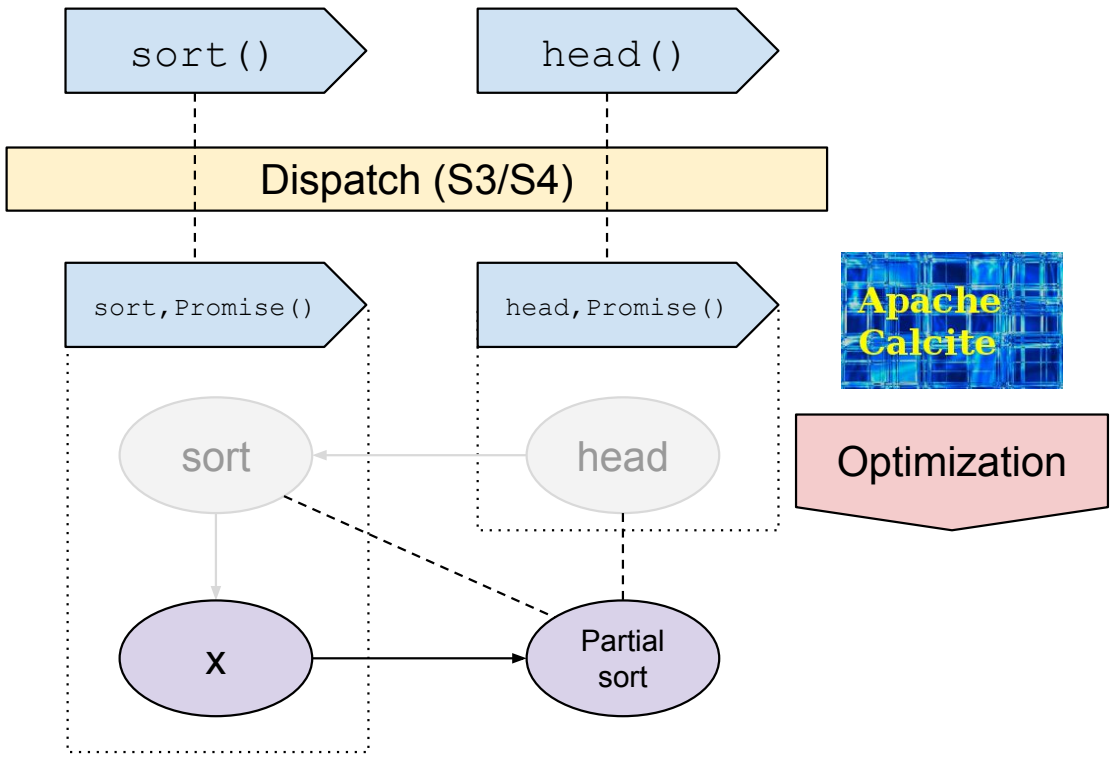


Looking forward: generalized, integrated compute

Intermediate algebra, optimization with backend-informed cost model

For some promise "x":

```
> head(sort(x))
```

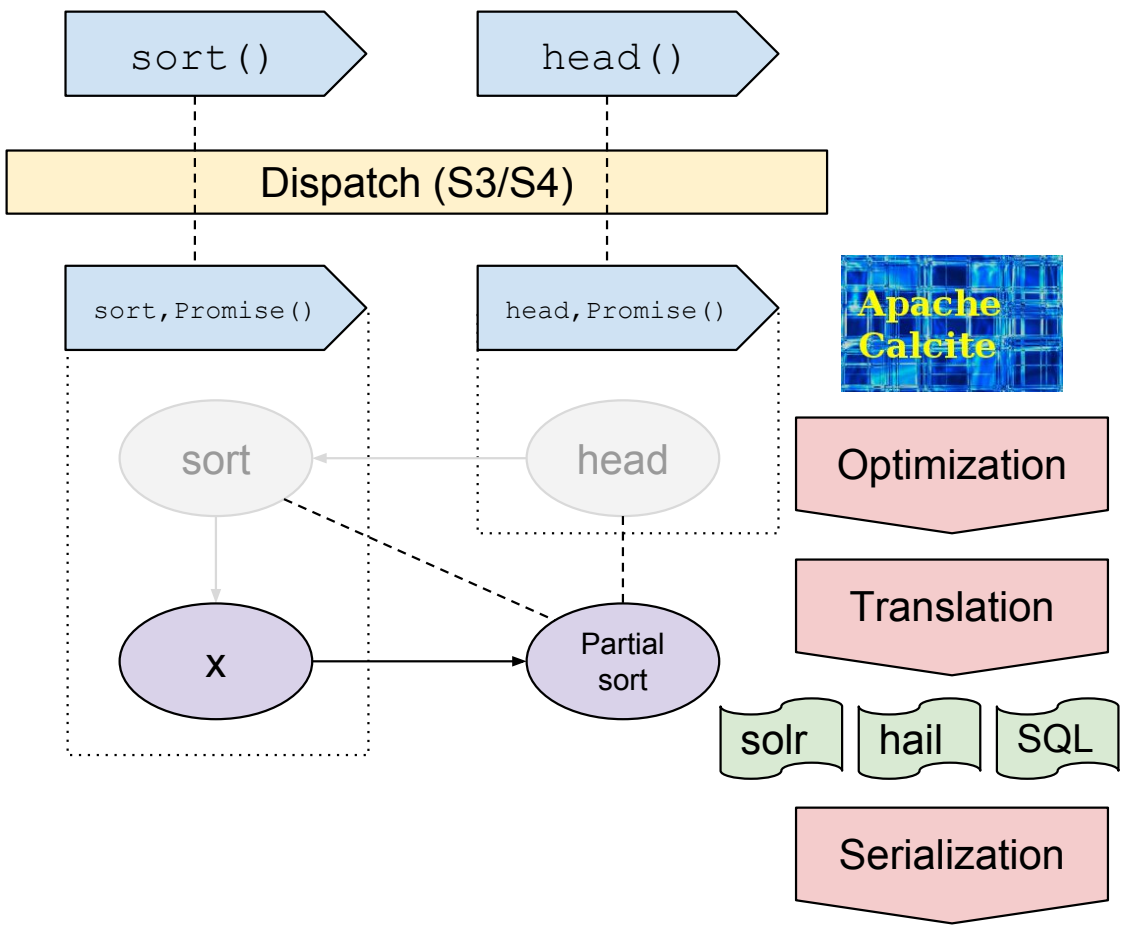


Looking forward: generalized, integrated compute

Intermediate algebra, optimization with backend-informed cost model

For some promise "x":

```
> head(sort(x))
```



Related developments

DelayedArray Bioconductor framework for operating on large, out-of-core arrays

- Pluggable backends for different storage modes
- Defers operations
- Processes chunkwise

ALTREP Generalization of internal R vector implementation

- Compact representations
- Out-of-core storage
- Extensible by packages