Exploring the Ranges Infrastructure

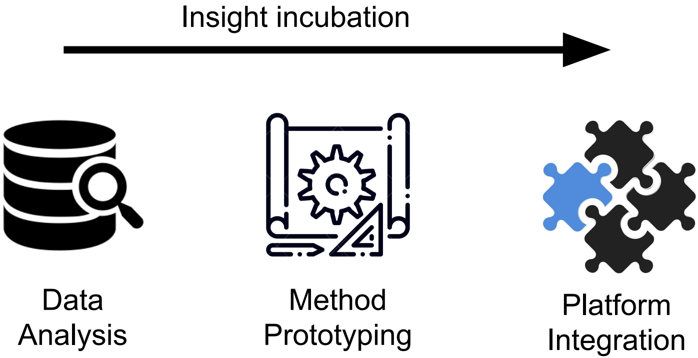
Michael Lawrence

July 27, 2017

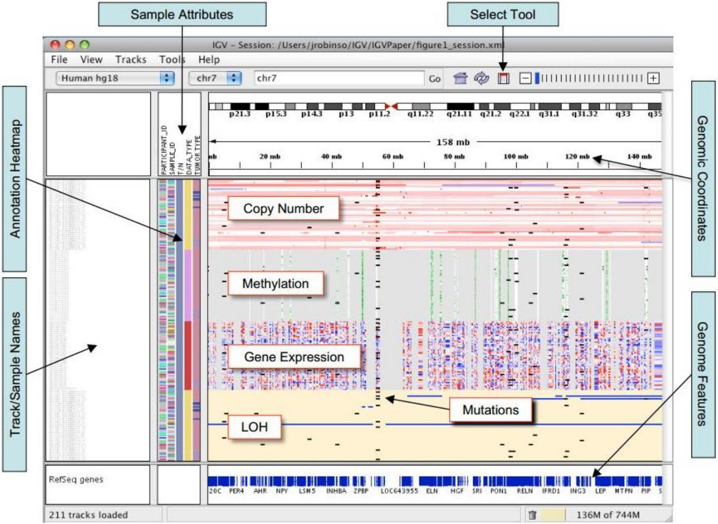
Outline

Outline

The Ranges infrastructure: what is it good for?

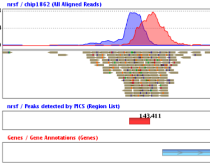


Integrative data analysis

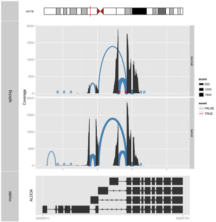


Developing and prototyping methods

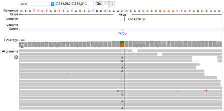
Peak calling



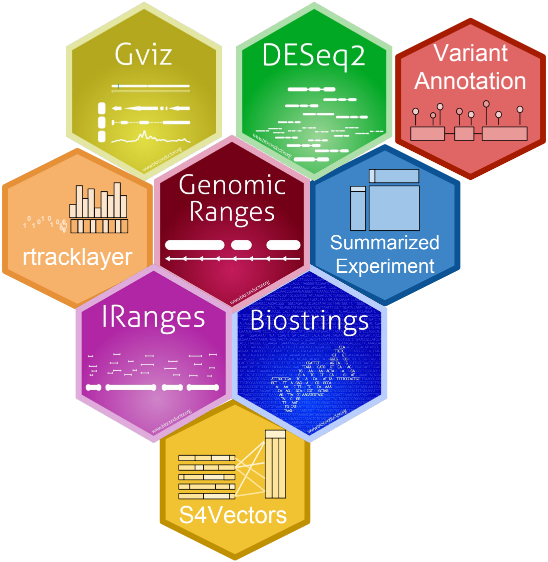
Isoform expression



Variant calling



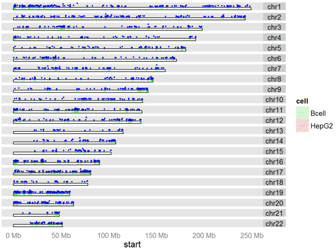
Software integration



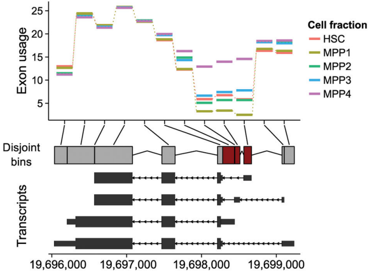
Outline

Data types

Data on genomic ranges



Summarized data



Reality

- In practice, we have 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
```

Reality bites

Now for a GFF file:

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

GFF

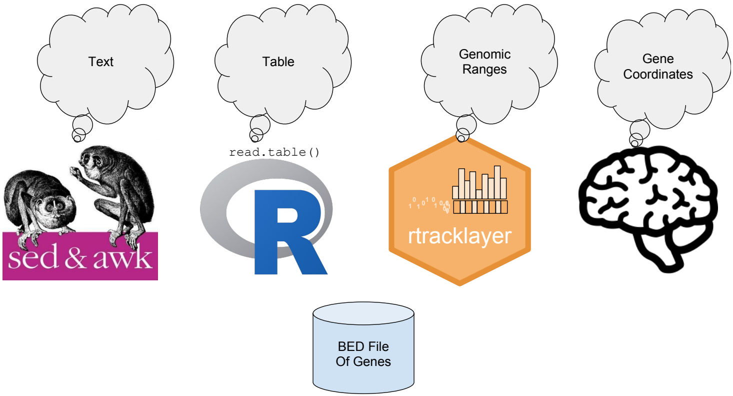
	chr	start	end
1	chr7	127471197	127472363
2	chr7	127472364	127473530
3	chr7	127473531	127474697
4	chr9	127474698	127475864
5	chr9	127475865	127477031

BED

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

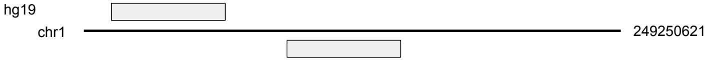
From reality to ideality

The abstraction gradient



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

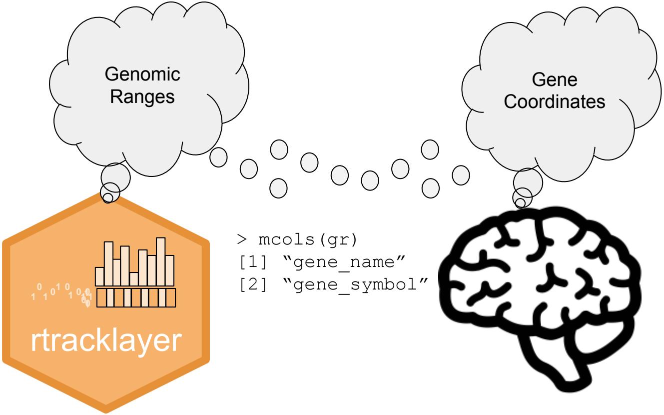
GRanges: data on genomic ranges



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

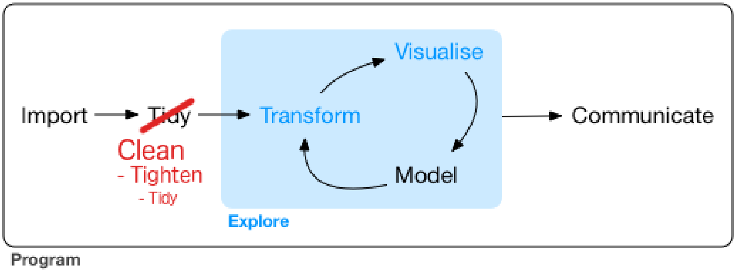
- Plus, sequence information (lengths, genome, etc)

Semantic slack



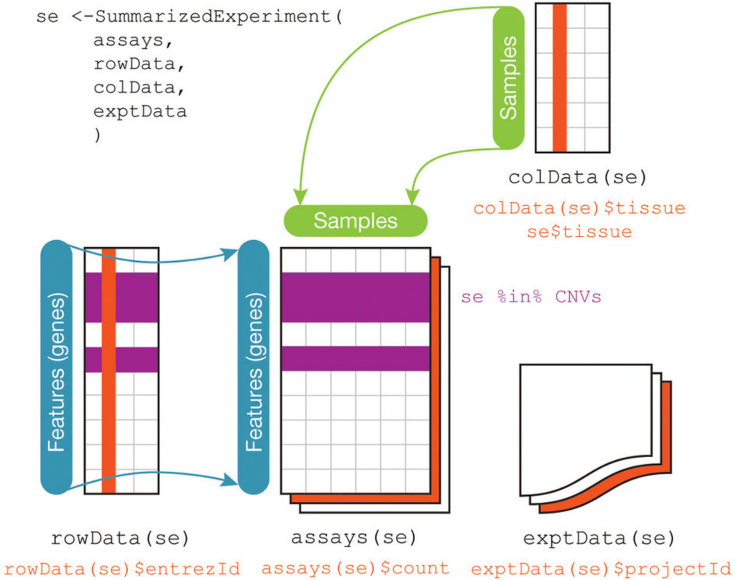
- Science defies rigidity: we define flexible objects that combine strongly typed fields with arbitrary user-level metadata

Abstraction is the responsibility of the user



- Only the user knows the true semantics of the data
- Explicitly declaring semantics:
  - Helps the software do the right thing
  - Helps the user be more *expressive*

SummarizedExperiment: the central data model



Outline

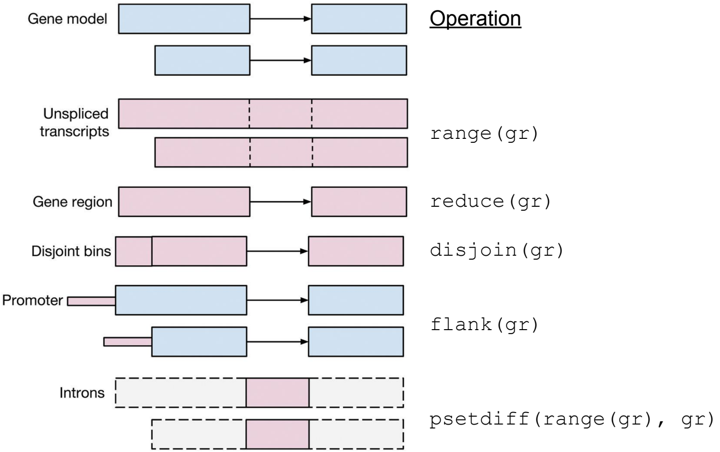
The Ranges API

- Semantically rich data enables:
  - Semantically rich vocabularies and grammars
  - Semantically aware behavior (DWIM)
- The range algebra expresses typical range-oriented operations
- Base R API is extended to have range-oriented behaviors

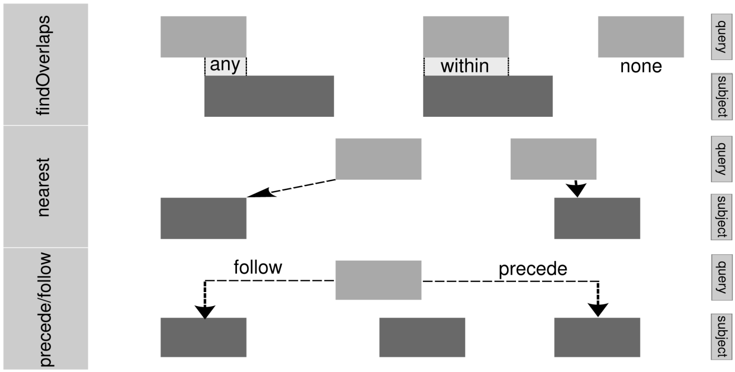
The Ranges API: Examples

Type	Range operations	Range extensions
Filter	subsetByOverlaps()	[()
Transform	shift(), resize()	*() to zoom
Aggregation	coverage(), reduce()	intersect(), union()
Comparison	findOverlaps(), nearest()	match(), sort()

Range algebra



Overlap detection

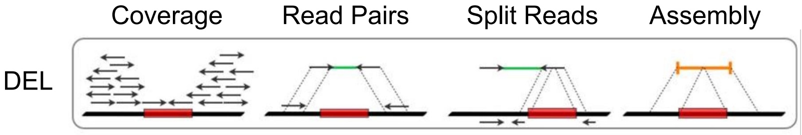


Outline

Structural variants are important for disease

- SVs are rarer than SNVs
  - SNVs: ~ 4,000,000 per genome
  - SVs: 5,000 - 10,000 per genome
- However, SVs are much larger (typically > 1kb) and cover more genomic space than SNVs.
- The effect size of SV associations with disease is larger than those of SNVs.
  - SVs account for 13% of GTEx eQTLs
  - SVs are 26 - 54 X more likely to modulate expression than SNVs (or indels)

Detection of deletions from WGS data



Motivation

Problem

- Often need to evaluate a tool before adding it to our workflow
- "lumpy" is a popular SV caller

Goal

Evaluate the performance of lumpy

Data

- Simulated a FASTQ containing known deletions using varsim
- Aligned the reads with BWA
- Ran lumpy on the alignments

Overview

1. Import the lumpy calls and truth set
2. Tidy the data
3. Match the calls to the truth
4. Compute error rates
5. Diagnose errors

Data import

Read from VCF:

```
library(RangesTutorial2017)
calls <- readVcf(system.file("extdata", "lumpy.vcf.gz",
                             package="RangesTutorial2017"))
truth <- readVcf(system.file("extdata", "truth.vcf.bgz",
                             package="RangesTutorial2017"))
```

Select for deletions:

```
truth <- subset(truth, SVTYPE=="DEL")
calls <- subset(calls, SVTYPE=="DEL")
```

Data cleaning

Make the seqlevels compatible:

```
seqlevelsStyle(calls) <- "NCBI"
truth <- keepStandardChromosomes(truth,
                                 pruning.mode="coarse")
```

Tighten

Move from the constrained VCF representation to a range-oriented model (*VRanges*) with a tighter cognitive link to the problem:

```
calls <- as(calls, "VRanges")
truth <- as(truth, "VRanges")
```

More cleaning

Homogenize the ALT field:

```
ref(truth) <- "."
```

Remove the flagged calls with poor read support:

```
calls <- calls[called(calls)]
```

Comparison

- How to decide whether a call represents a true event?
- Ranges should at least overlap:

```
hits <- findOverlaps(truth, calls)
```

- But more filtering is needed.

Comparing breakpoints

Compute the deviation in the breakpoints:

```
hits <- as(hits, "List")
call_rl <- extractList(ranges(calls), hits)
dev <- abs(start(truth) - start(call_rl)) +
    abs(end(truth) - end(call_rl))
```

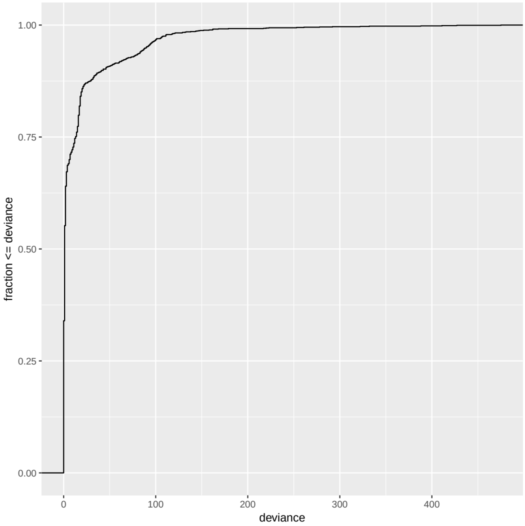
Select and store the call with the least deviance, per true deletion:

```
dev_ord <- order(dev)
keep <- phead(dev_ord, 1L)
truth$deviance <- drop(dev[keep])
truth$call <- drop(hits[keep])
```

Choosing a deviance cutoff

```
library(ggplot2)
rdf <- as.data.frame(truth)
ggplot(aes(x=deviance),
       data=subset(rdf, deviance <= 500)) +
    stat_ecdf() + ylab("fraction <= deviance")
```

Choosing a deviance cutoff



Applying the deviance filter

```r
truth$called <-
    with(truth, !is.na(deviance) & deviance <= 300)
```

Sensitivity

```r
mean(truth$called)
```

```
[1] 0.8214107
```

Specificity

Determine which calls were true:

```r
calls$fp <- TRUE
calls$fp[subset(truth, called)$call] <- FALSE
```

Compute FDR:

```r
mean(calls$fp)
```

```
[1] 0.1009852
```

Explaining the FDR

- Suspect that calls may be error-prone in regions where the population varies
- Load alt regions from a BED file:

```
file <- system.file("extdata",
                    "altRegions.GRCh38.bed.gz",
                    package="RangesTutorial2017")
altRegions <- import(file)
seqlevelsStyle(altRegions) <- "NCBI"
altRegions <-
    keepStandardChromosomes(altRegions,
                            pruning.mode="coarse")
```

FDR and variable "alt" regions

- Compute the association between FP status and overlap of an alt region:

```
calls$inAlt <- calls %over% altRegions
xtabs(~ inAlt + fp, calls)
```

```
       fp
inAlt    FALSE TRUE
  FALSE   1402  112
  TRUE      58   52
```