

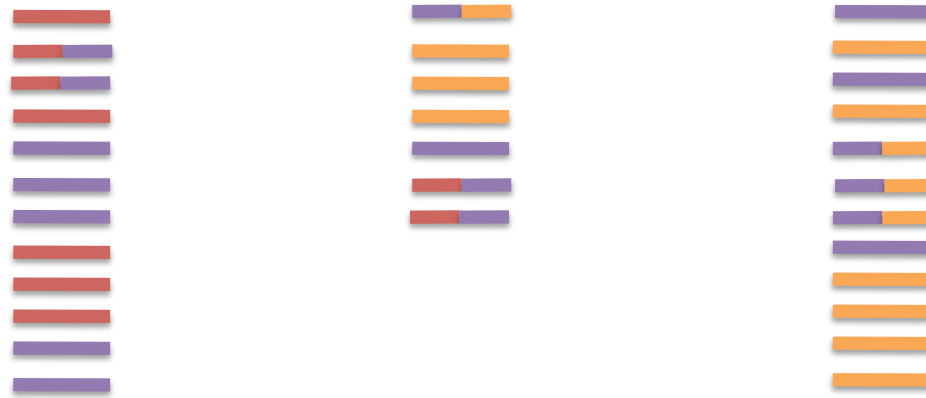
Ballgown

**flexible RNA-seq differential
expression analysis**

Alyssa Frazee
Johns Hopkins Biostatistics
@acfrazee

RNA-seq data

Reads
(50-100
bases)



Transcripts
(RNA)

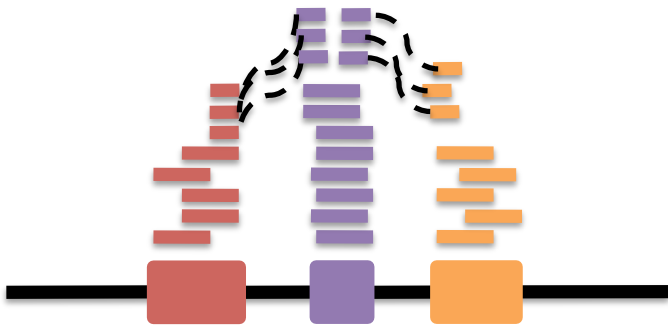


Genome
(DNA)



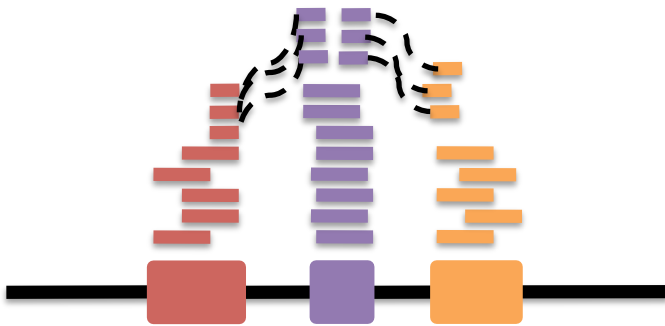
RNA-seq data analysis

align



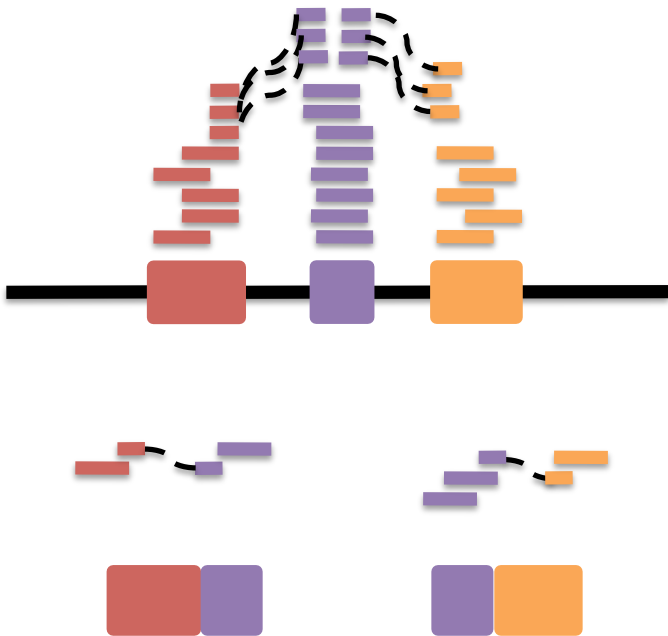
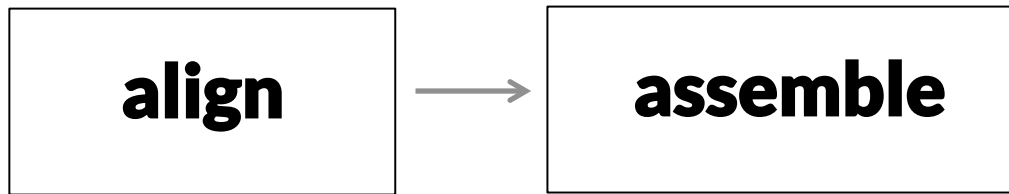
RNA-seq data analysis

align

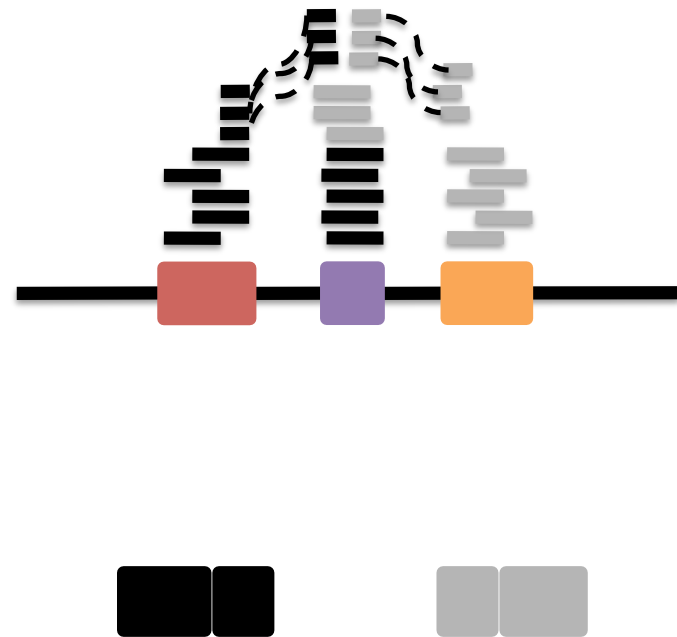
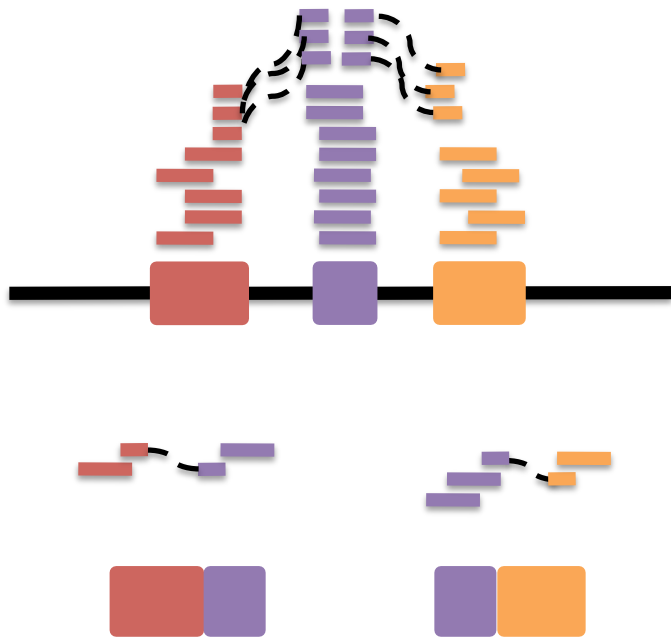
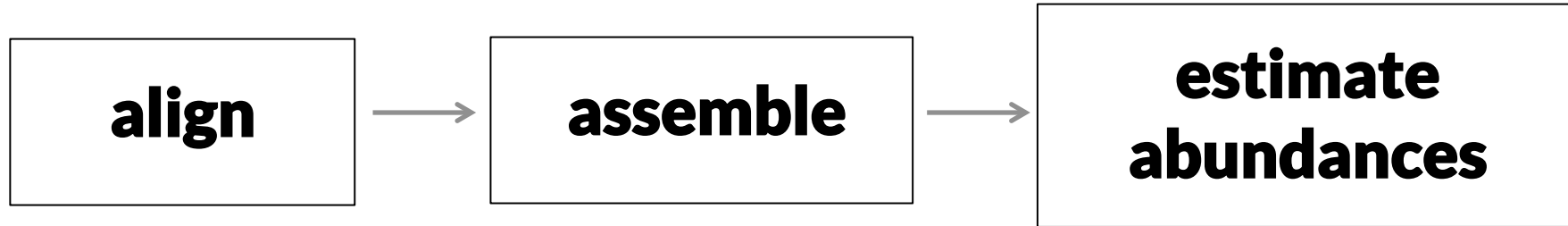


(or not: *de novo* assembly)

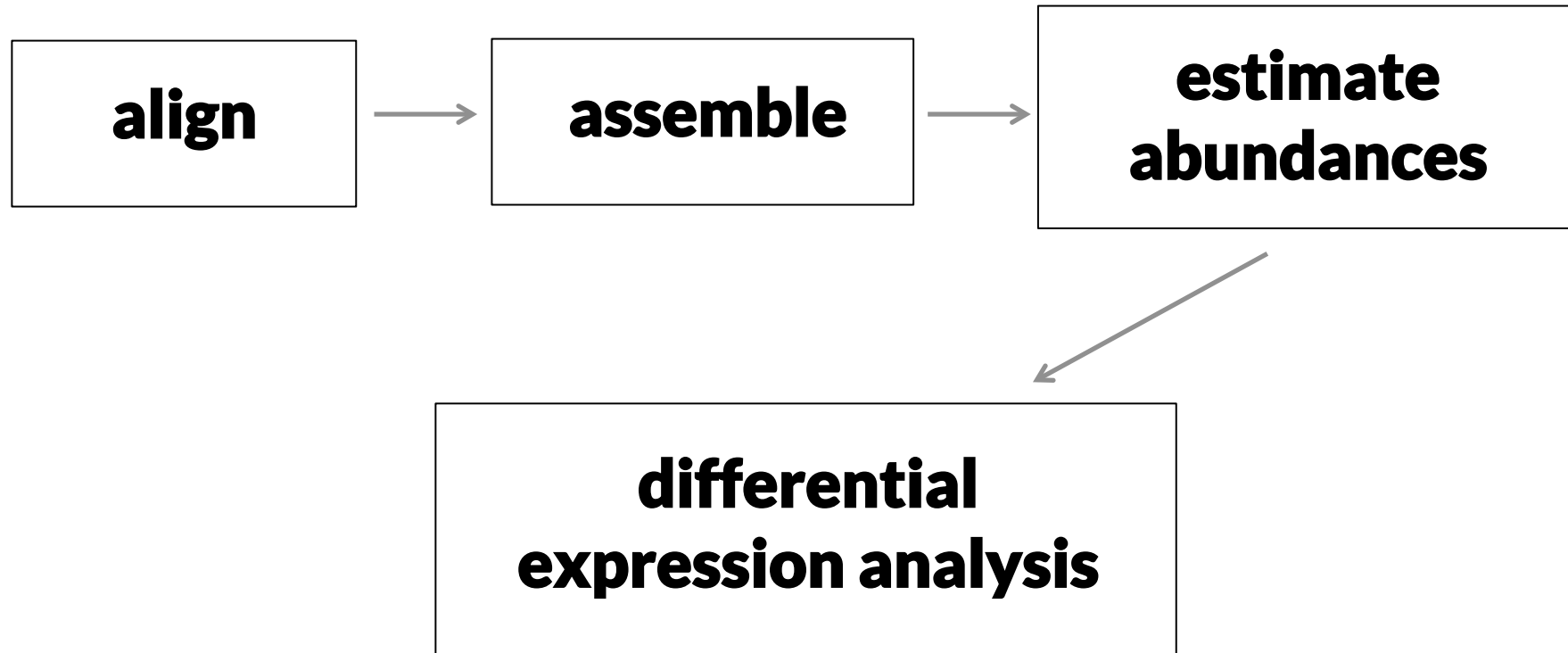
RNA-seq data analysis



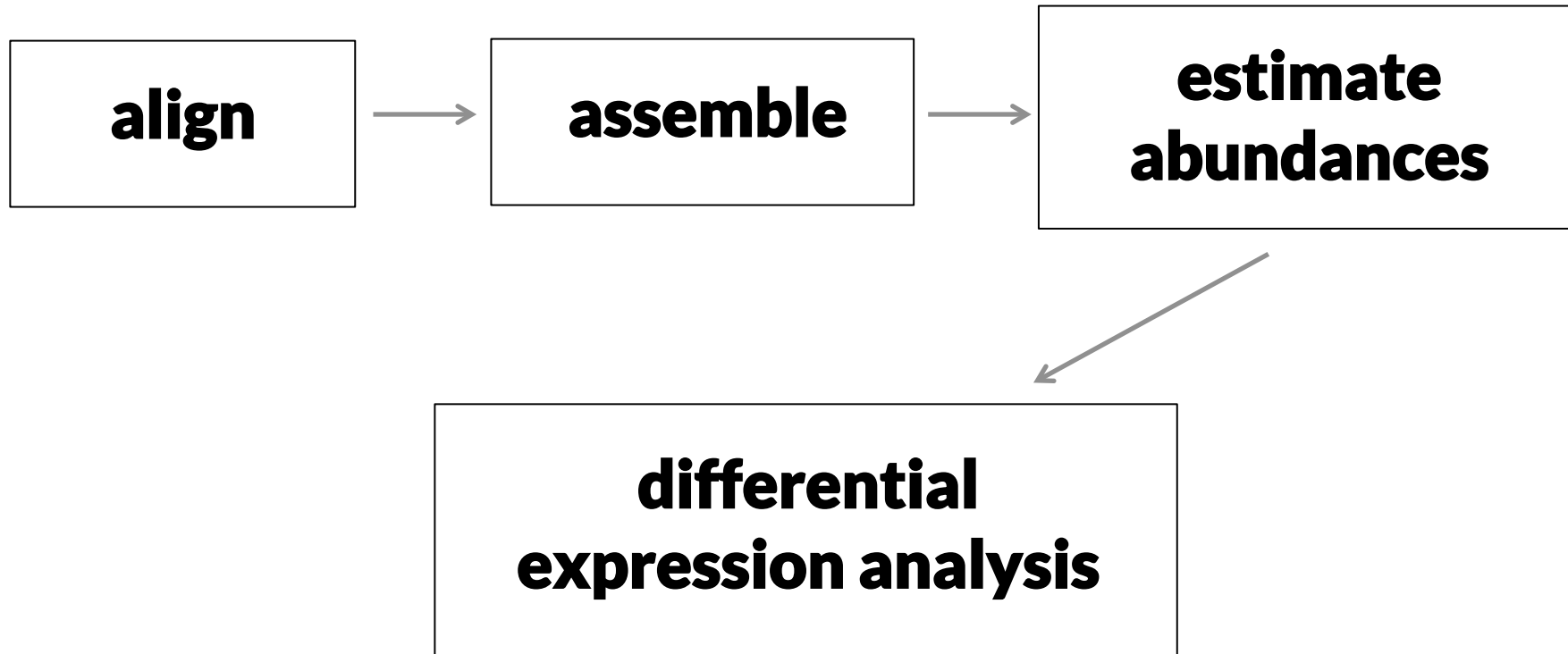
RNA-seq data analysis



RNA-seq data analysis

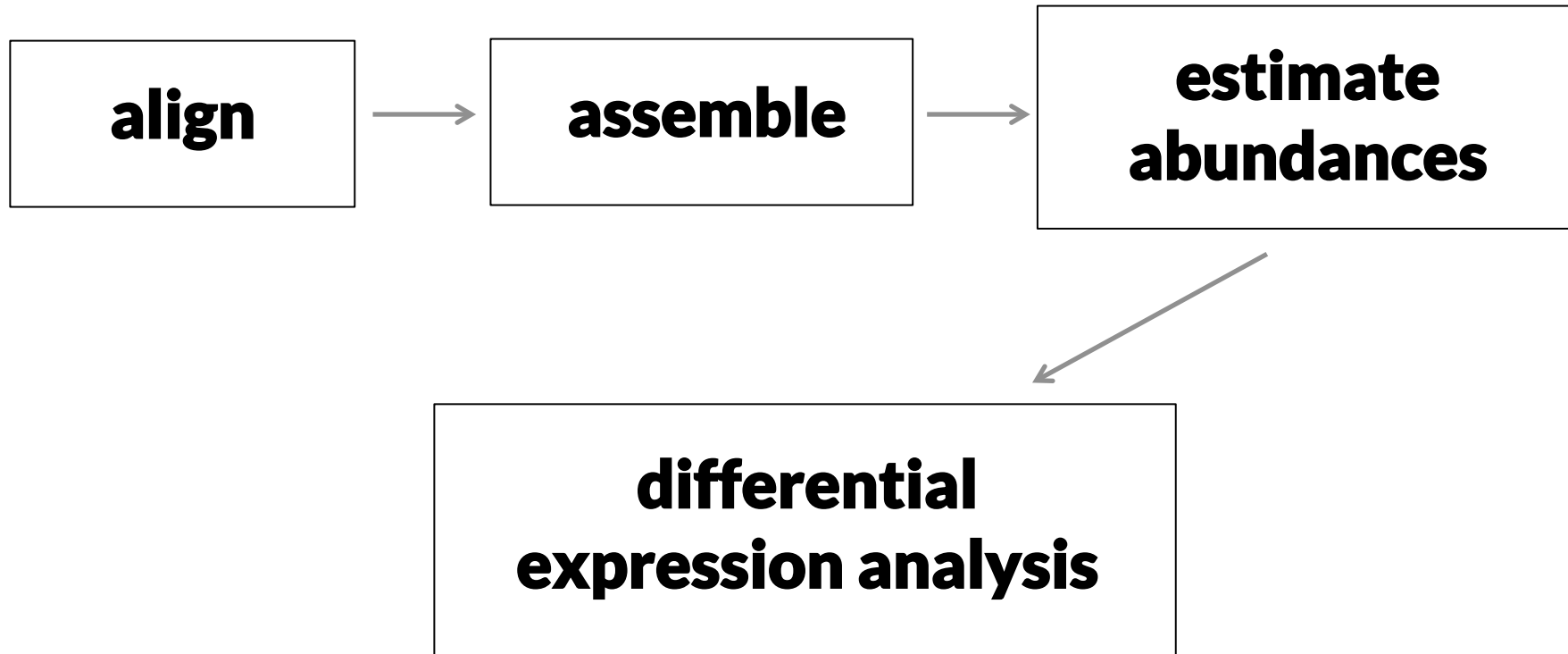


RNA-seq data analysis



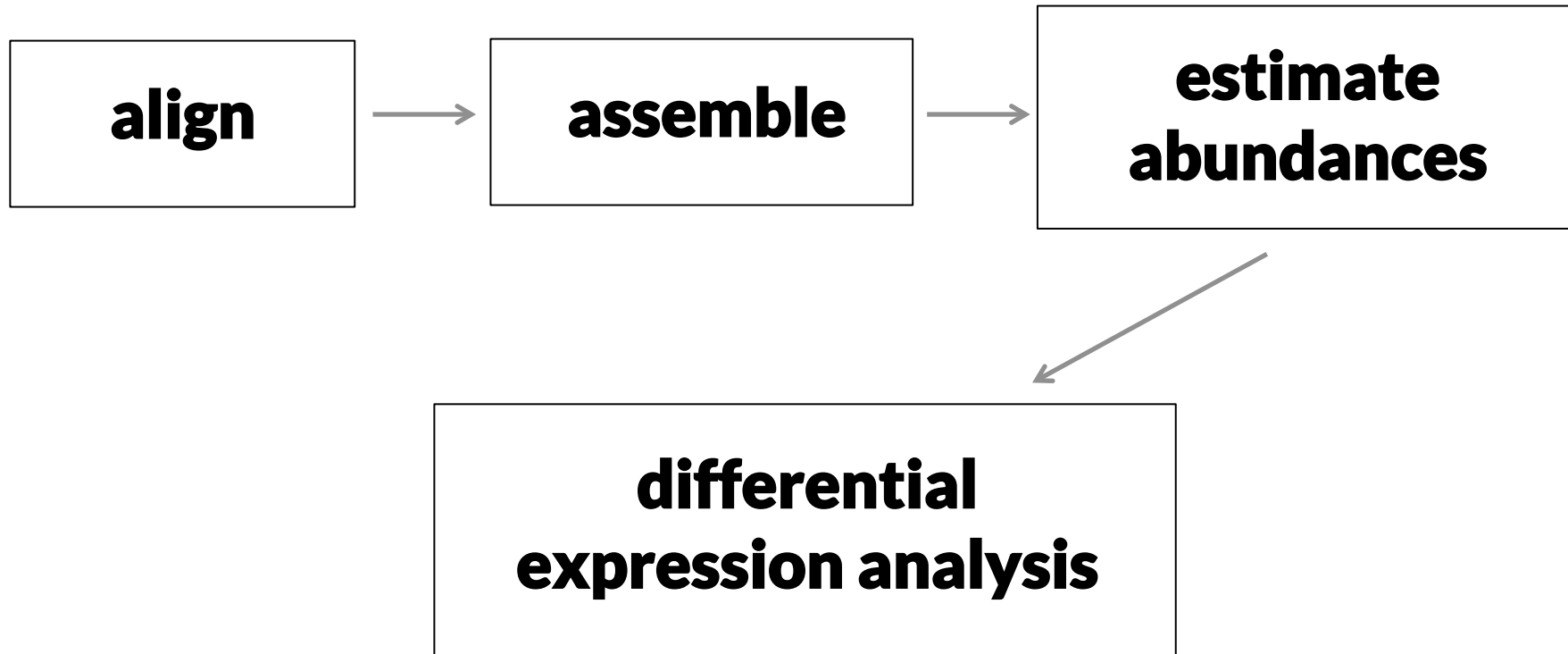
Is transcript X expressed differently in population A
vs. population B?

RNA-seq data analysis



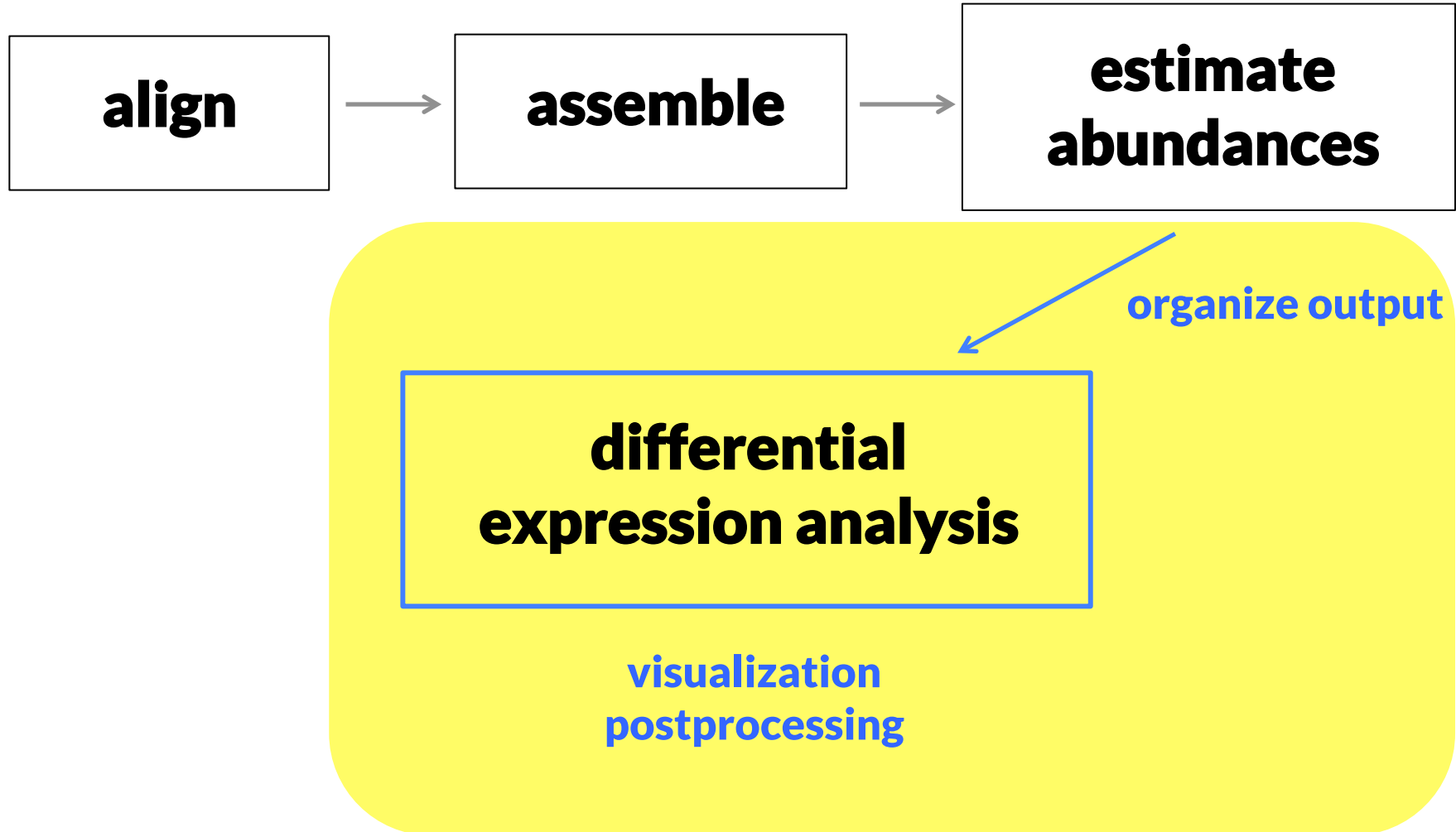
Is transcript X expressed differently in population A vs. population B vs. population C?

RNA-seq data analysis



Is transcript X expressed constantly over the course of time in the study?

Ballgown



organize output: Tablemaker

```
top_dir/  
  sample01/  
    e2t.ctab  
    e_data.ctab  
    i2t.ctab  
    i_data.ctab  
    t_data.ctab  
  sample02/  
    e2t.ctab  
    e_data.ctab  
    i2t.ctab  
    i_data.ctab  
    t_data.ctab  
  sample03/  
    e2t.ctab  
    e_data.ctab  
    i2t.ctab  
    i_data.ctab  
    t_data.ctab
```

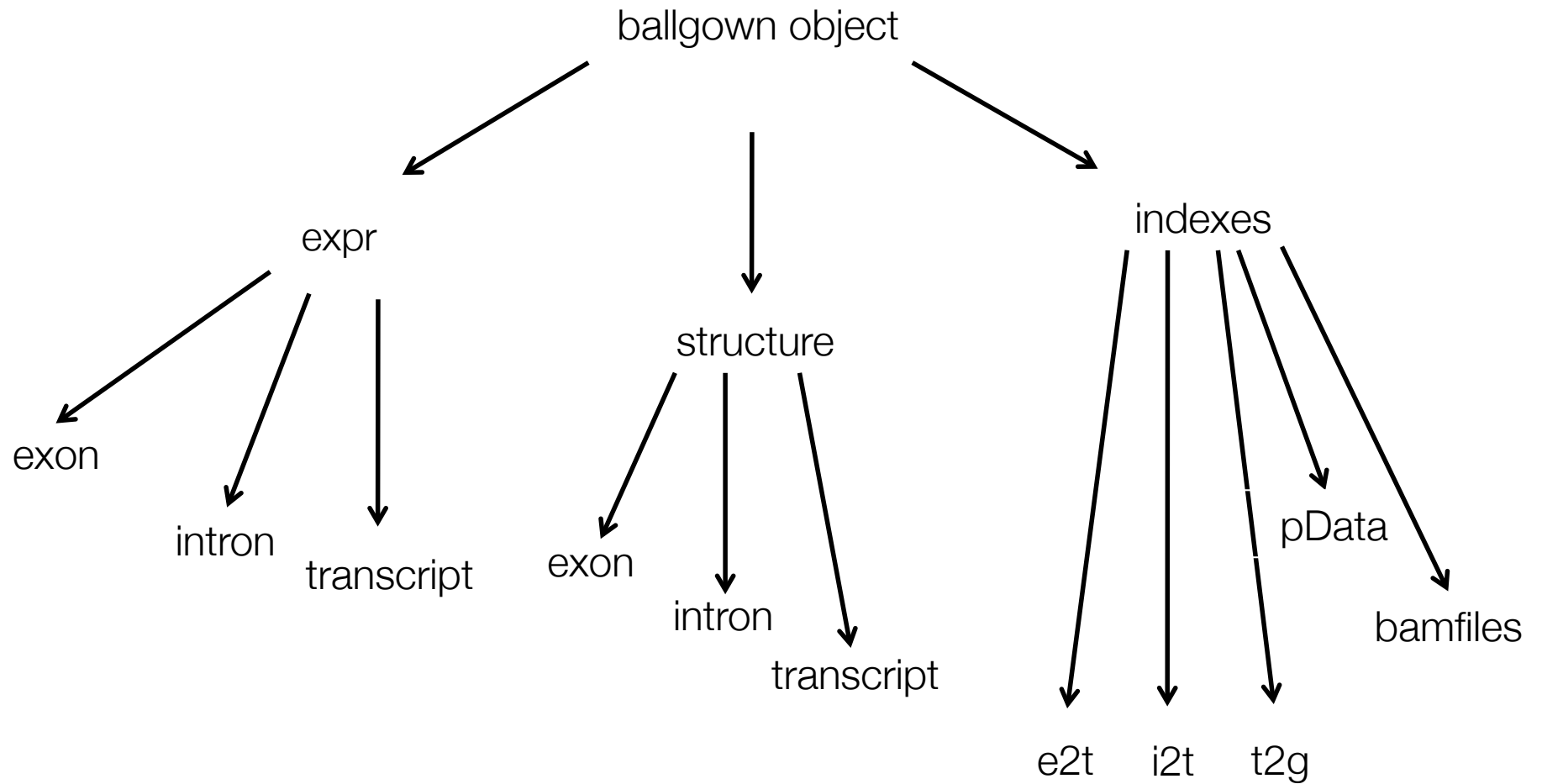
S4 class for transcript assemblies

```
library(ballgown)

my_assembly = ballgown(dataDir='top_dir', samplePattern='sample')
my_assembly
## ballgown instance with 124892 transcripts and 3 samples

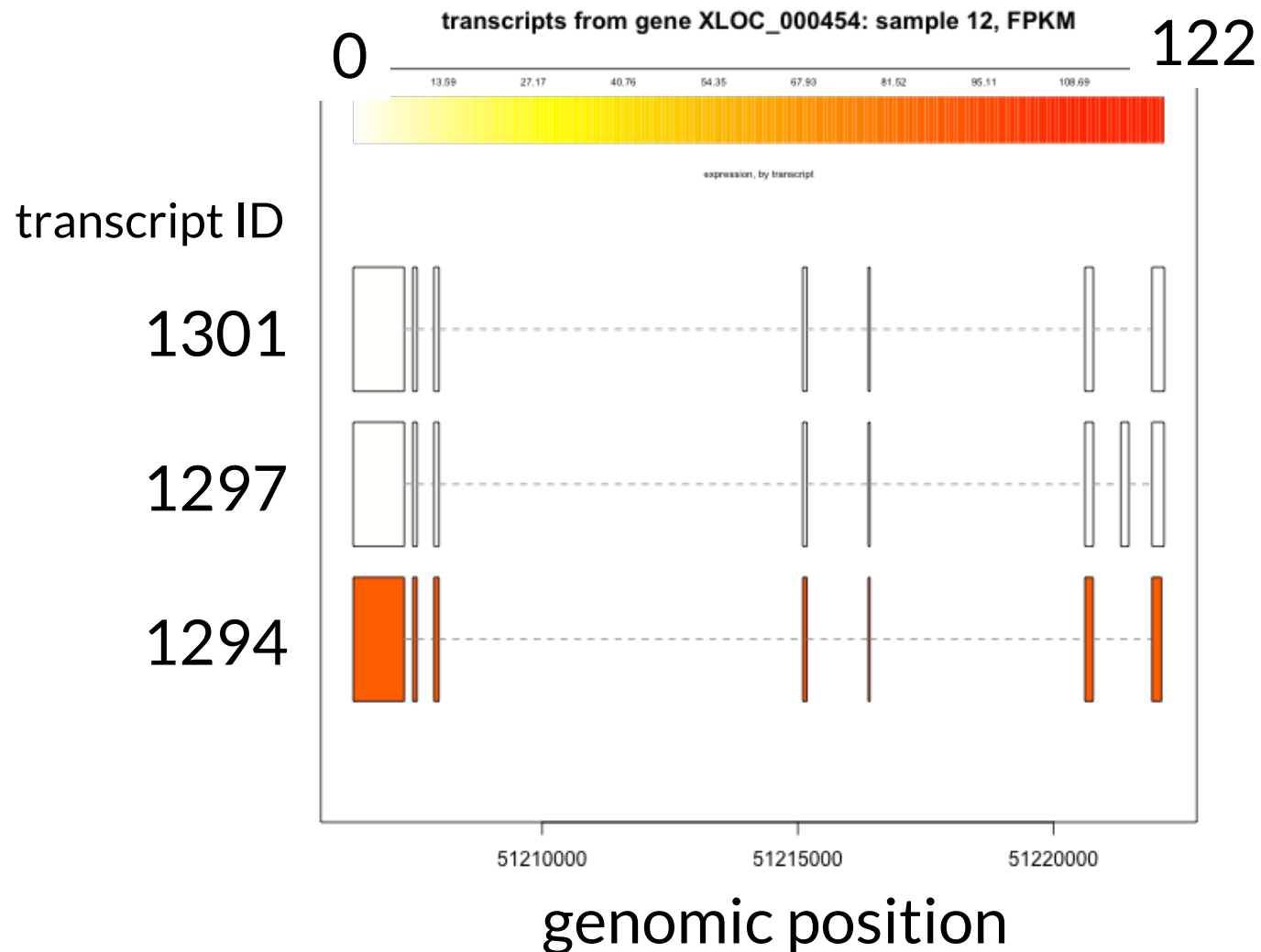
my_rsem_assembly = ballgownrsem(dir='top_dir', samples=c('sample01', 'sample02', 'sample03'),
  gtf='hg19_genes.gtf')
my_rsem_assembly
## ballgown instance with 45914 transcripts and 3 samples
```

S4 class for transcript assemblies



convenient visualization

```
plotTranscripts(gene='XLOC_000454', gown=my_assembly,  
samples='sample12', meas='FPKM', colorby='transcript',  
main='transcripts from gene XLOC_000454: sample 12, FPKM')
```



fast, flexible statistical analysis

```
stat_results = statstest(my_assembly, feature='transcript',  
                        meas='FPKM', covariate='group')
```

```
head(stat_results)
```

```
## feature      id      pval      qval  
## transcript  10      0.01381576  0.105212332  
## transcript  25      0.26773622  0.791149753  
## transcript  35      0.01085070  0.089518254  
## transcript  41      0.47108019  0.902537475  
## transcript  45      0.08402948  0.489348136  
## transcript  67      0.27317385  0.79114975
```

drop-in replacement for Cuffdiff

fast, flexible statistical analysis

```
stat_results = statstest(my_assembly, feature='transcript',  
                        meas='FPKM', covariate='group')
```

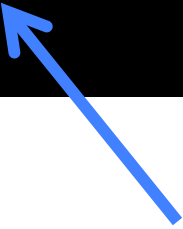
```
head(stat_results)
```

```
## feature      id      pval      qval  
## transcript  10      0.01381576  0.105212332  
## transcript  25      0.26773622  0.791149753  
## transcript  35      0.01085070  0.089518254  
## transcript  41      0.47108019  0.902537475  
## transcript  45      0.08402948  0.489348136  
## transcript  67      0.27317385  0.79114975
```

drop-in replacement for Cuffdiff
bonus: handles more experimental designs!

timecourse

```
stat_results = stattest(my_assembly, feature='transcript',  
                        meas='FPKM', covariate='group')  
  
timecourse_results = stattest(my_assembly,  
                              feature='transcript', meas='FPKM', covariate='time',  
                              timecourse=TRUE)
```



models expression over
time as smooth curve;
compares to model where
expression is assumed
constant over time

custom models

```
stat_results = statstest(my_assembly, feature='transcript',  
                        meas='FPKM', covariate='group')
```

```
timecourse_results = statstest(my_assembly,  
                               feature='transcript', meas='FPKM', covariate='time',  
                               timecourse=TRUE)
```

```
mod = model.matrix(~ time + library_size)  
mod0 = model.matrix(~ library_size)  
my_timecourse_results = statstest(my_assembly,  
                                  feature='transcript', meas='FPKM', mod=mod, mod0=mod0)
```

easy integration with other packages

```
stat_results = statstest(my_assembly, feature='transcript',  
                        meas='FPKM', covariate='group')
```

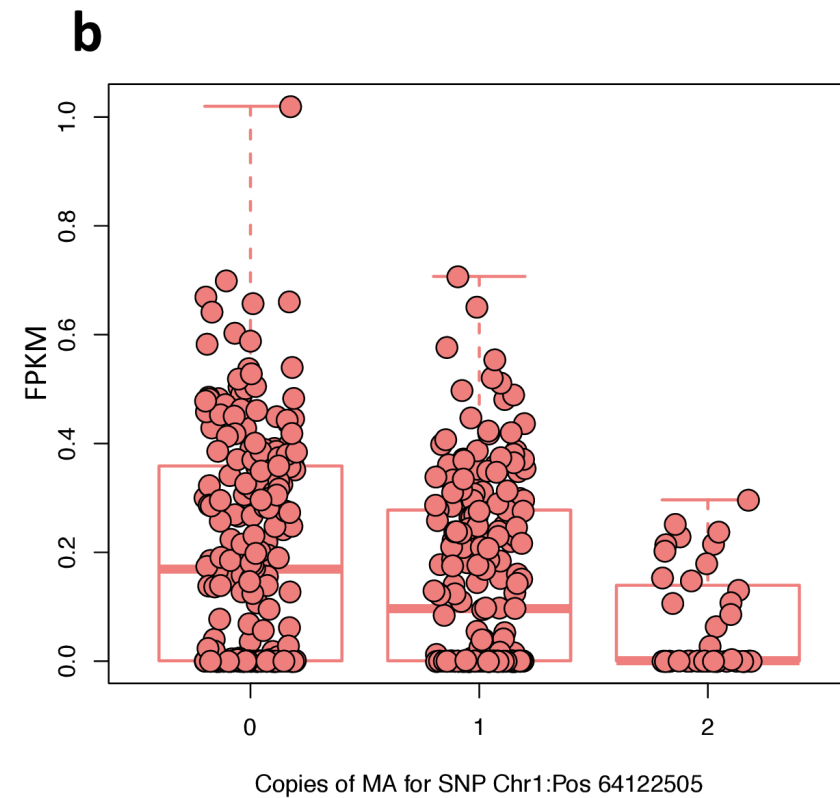
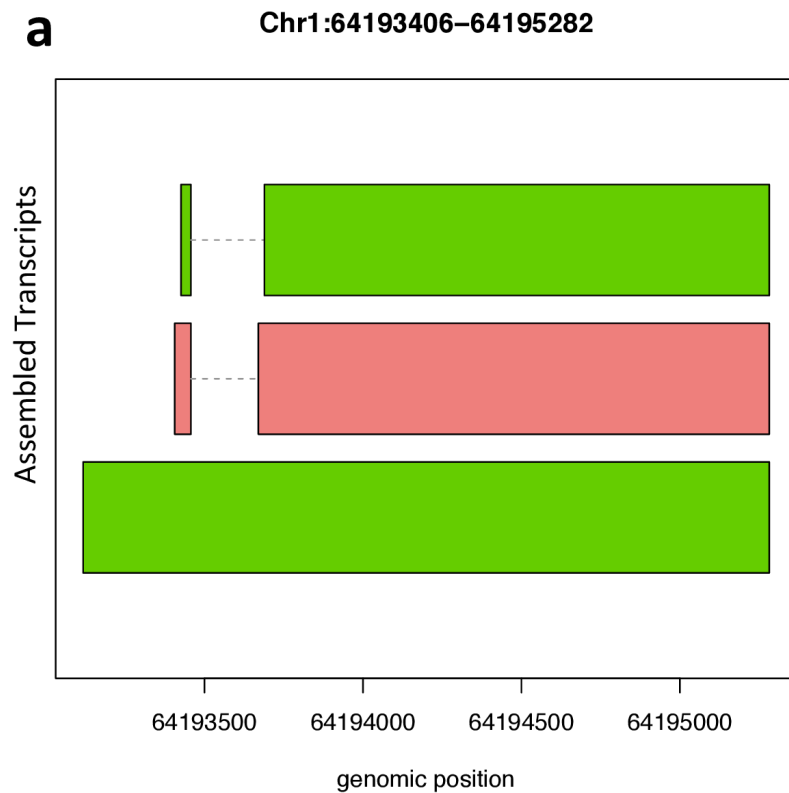
```
timecourse_results = statstest(my_assembly,  
                               feature='transcript', meas='FPKM', covariate='time',  
                               timecourse=TRUE)
```

```
mod = model.matrix(~ time + library_size)  
mod0 = model.matrix(~ library_size)  
my_timecourse_results = statstest(my_assembly,  
                                  feature='transcript', meas='FPKM', mod=mod, mod0=mod0)
```

```
library(limma)  
expression_mat = log2(texpr(my_assembly, 'FPKM') + 1)  
fit = lmFit(y, mod)  
fit = eBayes(fit, trend=TRUE)
```

wrangling large datasets

reanalysis of GEUVADIS data: eQTL



wrangling large datasets

reanalysis / processing of GEUVADIS data

Ballgown objects: figshare

TopHat read alignments: ArrayExpress (E-GEUV-6)

analysis code: GitHub,
https://github.com/alyssafrazee/ballgown_code

thanks!

Collaborators: Jeff Leek (advisor), Geo Pertea, Andrew Jaffe, Ben Langmead, Steven Salzberg

References:

manuscript: <http://biorxiv.org/content/early/2014/03/30/003665>

limma: “Linear models for Microarray Data” by Gordon K Smyth, in Bioinformatics and Computational Biology Solutions using R and Bioconductor, Springer 2005

Cufflinks: PMID 20436464 (Trapnell et al 2010)

RSEM: PMID 21816040 (Li & Dewey 2011)

Software: Ballgown is under Bioconductor review; currently available and fully documented at <https://github.com/alyssafrazee/ballgown>

differential expression model

for each transcript, compare the fits of the following models using an F-test. Null hypothesis is that the fits of model (a) and model (b) are equally good; alternative is that (a) fits better.

$$(a) \text{ expression}_i = \alpha + \beta_0 \text{group}_i + \sum_{p=1}^P \gamma_p \text{confounder}_{ip} + \text{noise}_{ip}$$

$$(b) \text{ expression}_i = \alpha^* + \sum_{p=1}^P \gamma_p^* \text{confounder}_{ip} + \text{noise}_{ip}^*$$