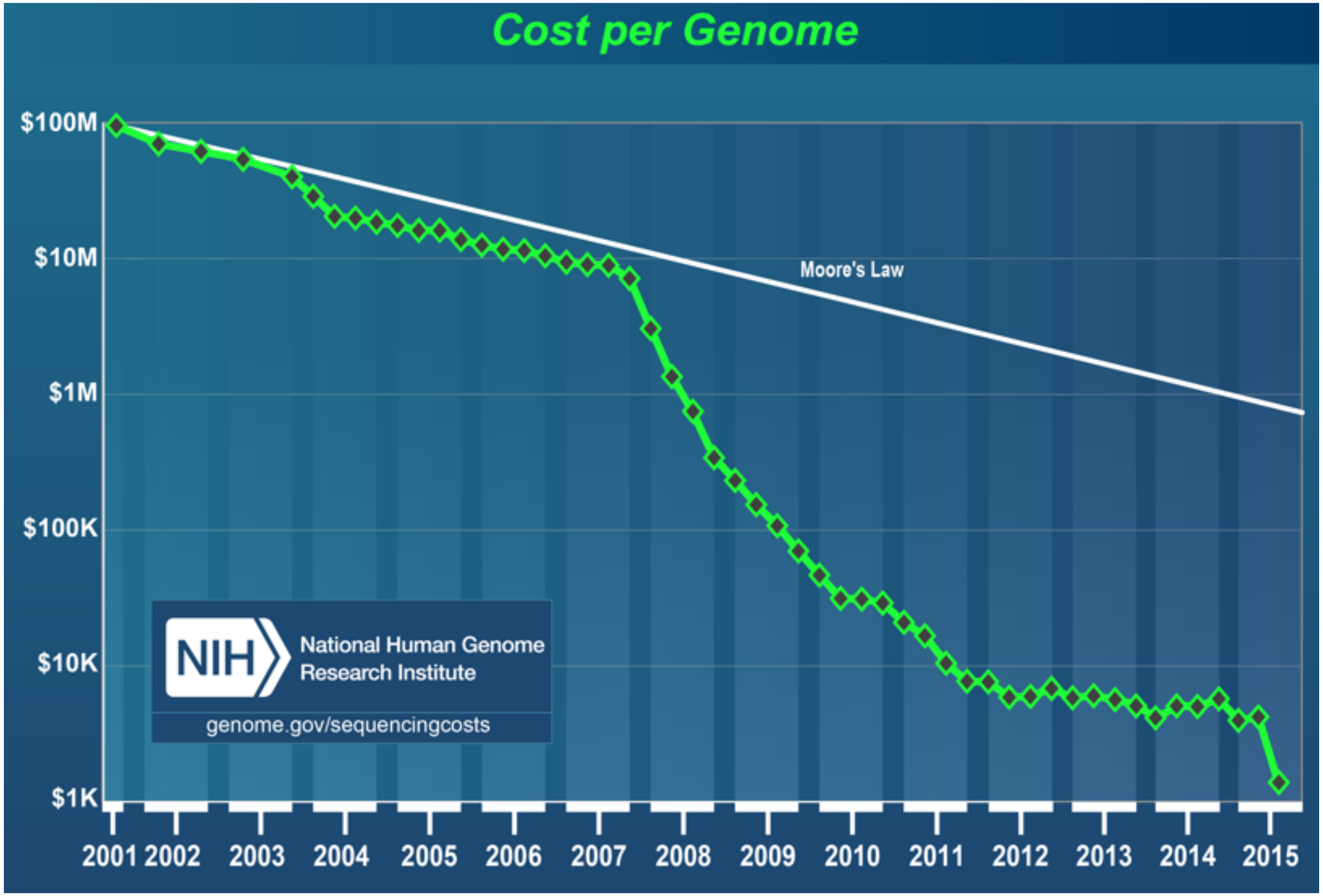
MakeMyTranscriptome

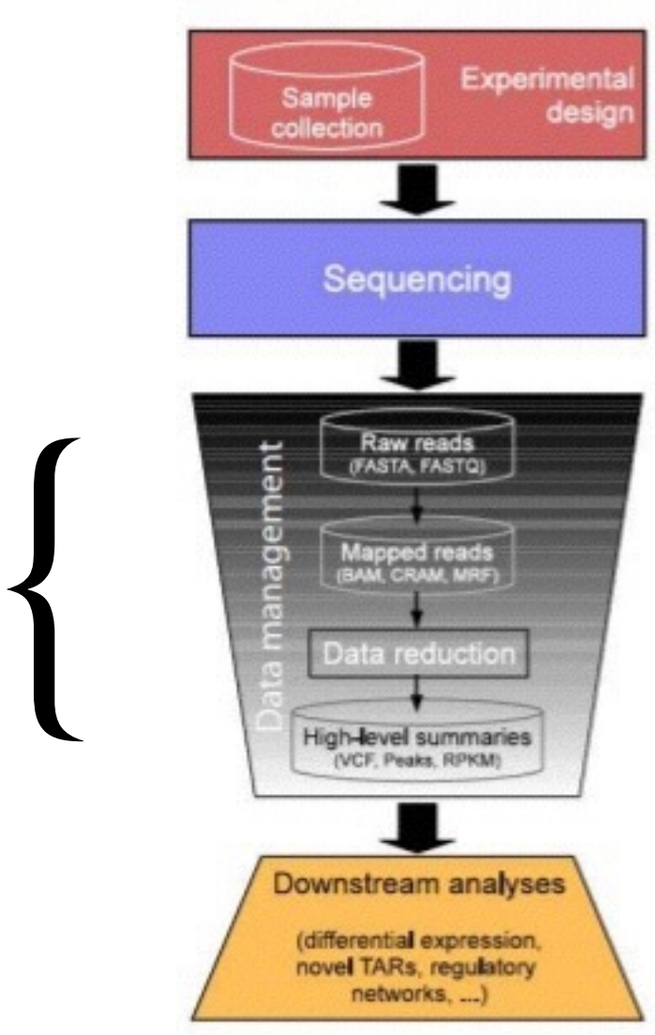
an automated pipeline for *de novo* transcriptomics

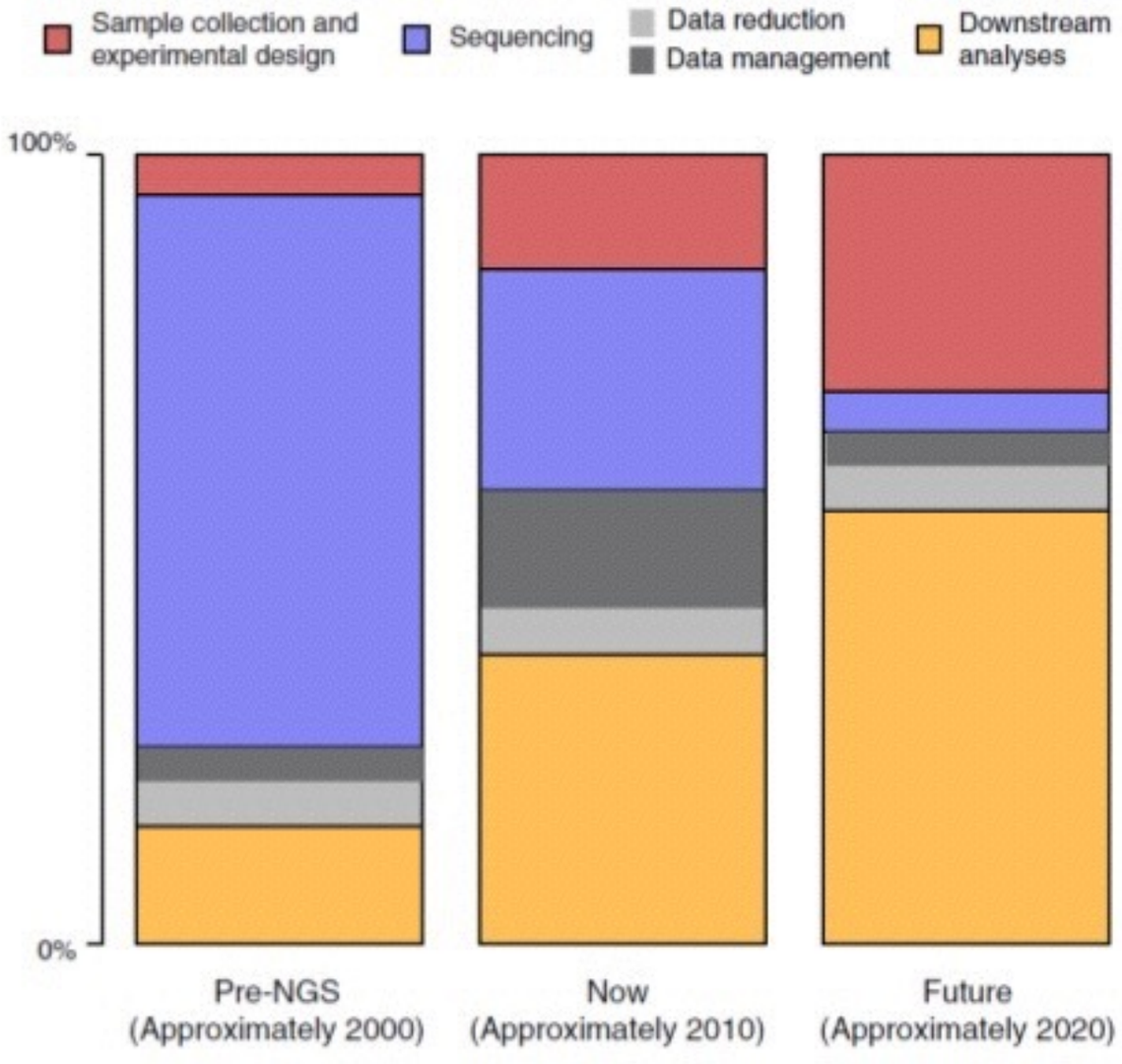
SIO Bioinformatics Users Group
N Tessa Pierce
2/8/2016

Cost of sequencing



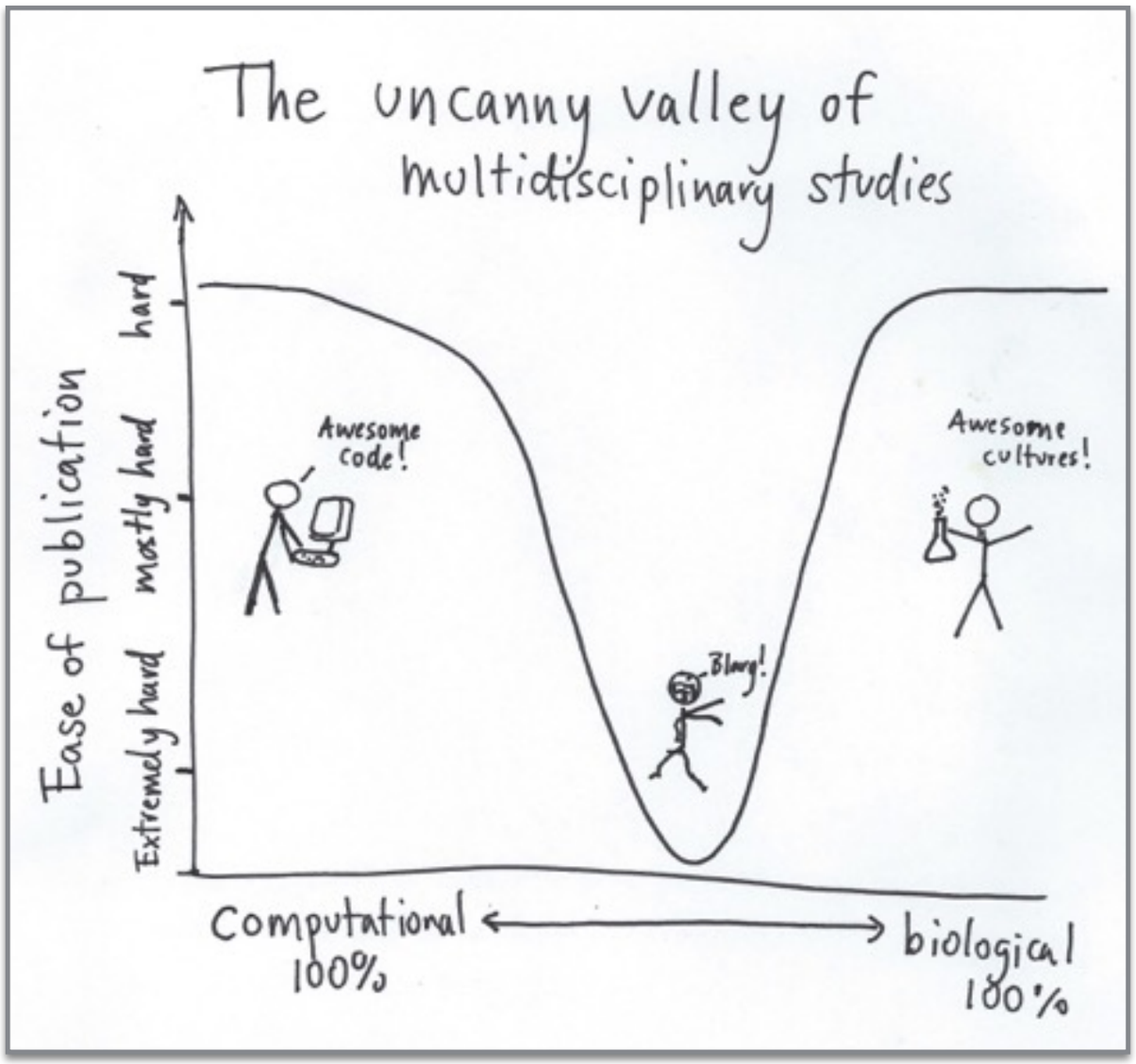
REAL Cost of sequencing…





Sooner et al, 2011. The real cost of sequencing: higher than you think!

Solutions?

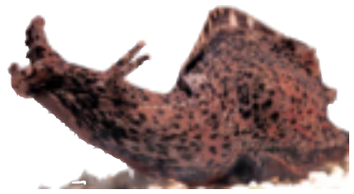


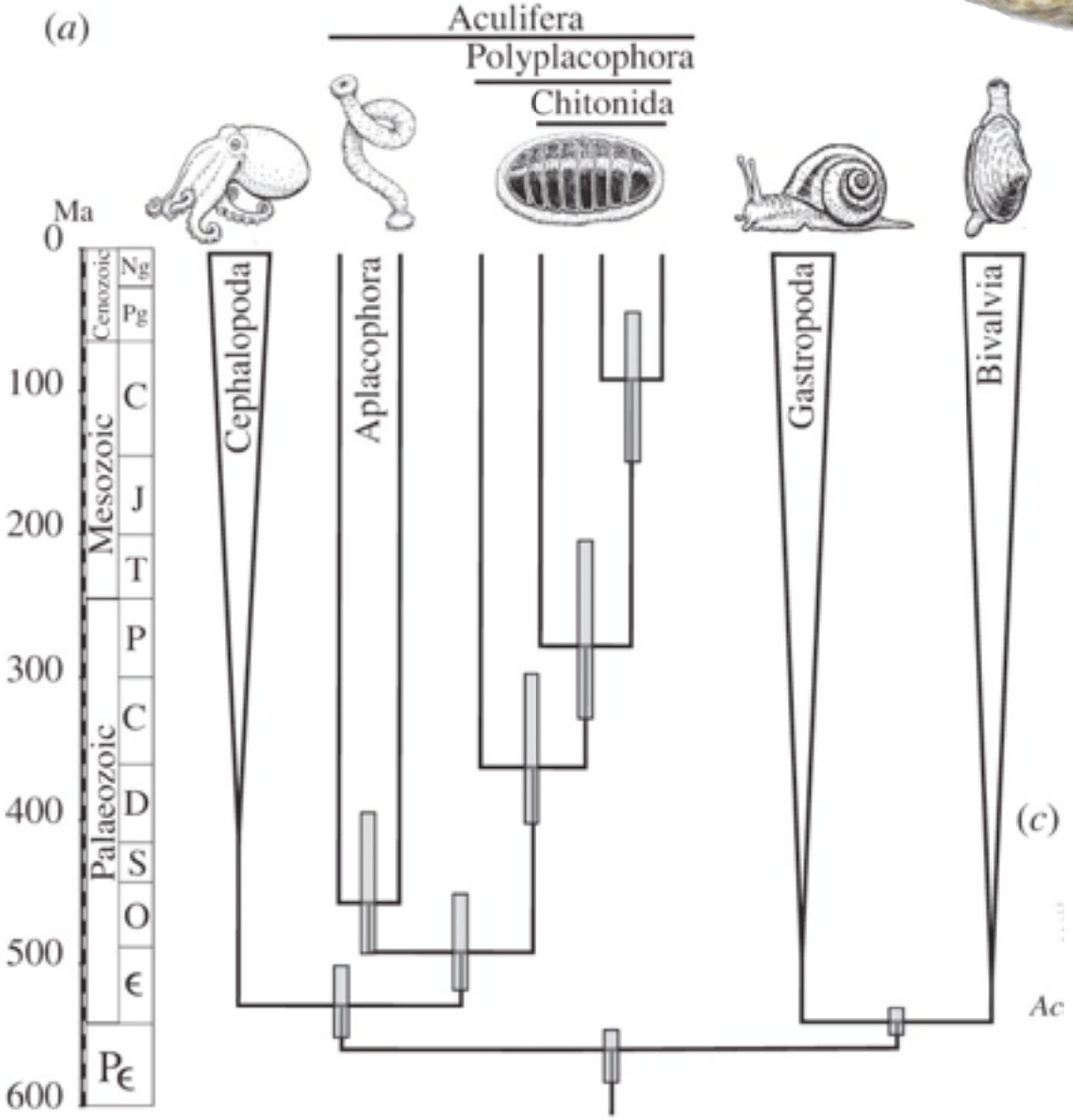
Training + programming

SIMPLER software

Goal: zero-entry analysis

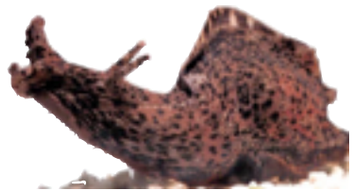
Initial goal: develop transcriptome resource

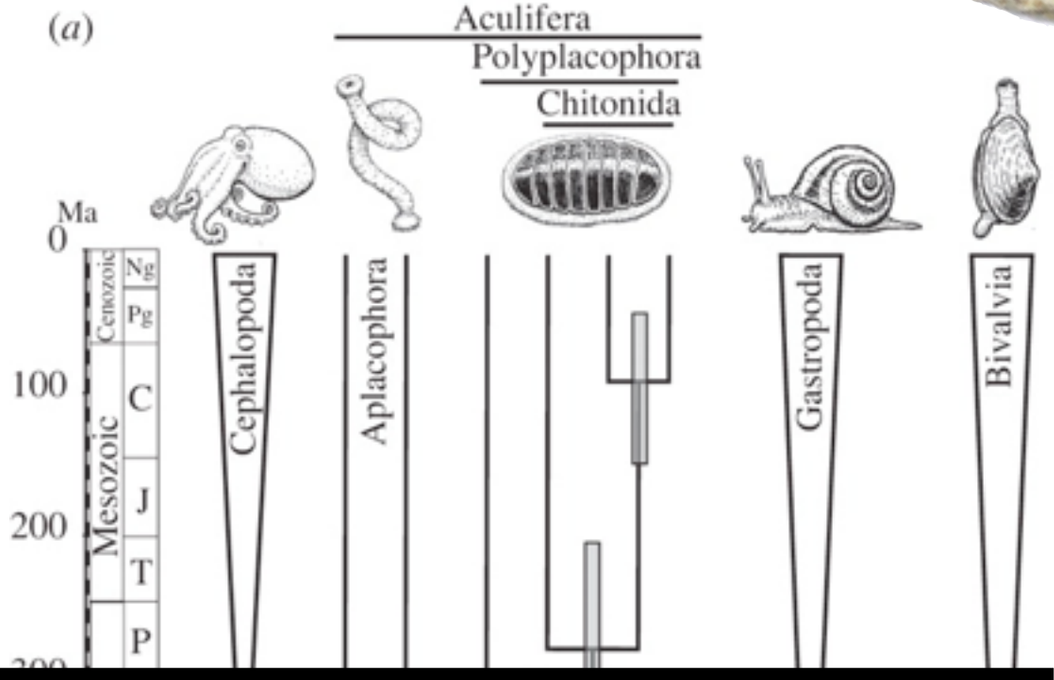




Vinther et al. 2012

Initial goal: develop transcriptome resource

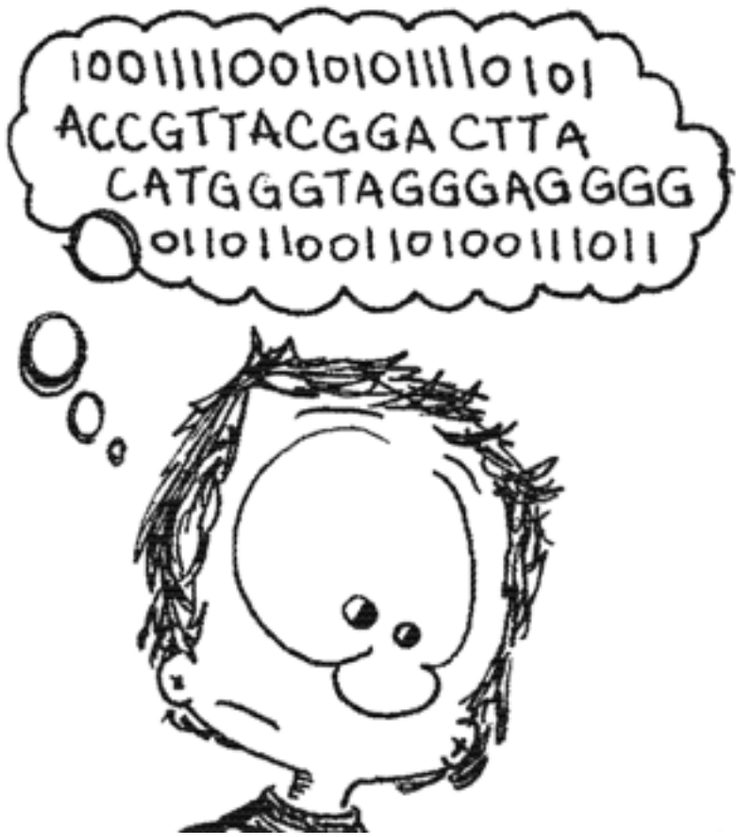


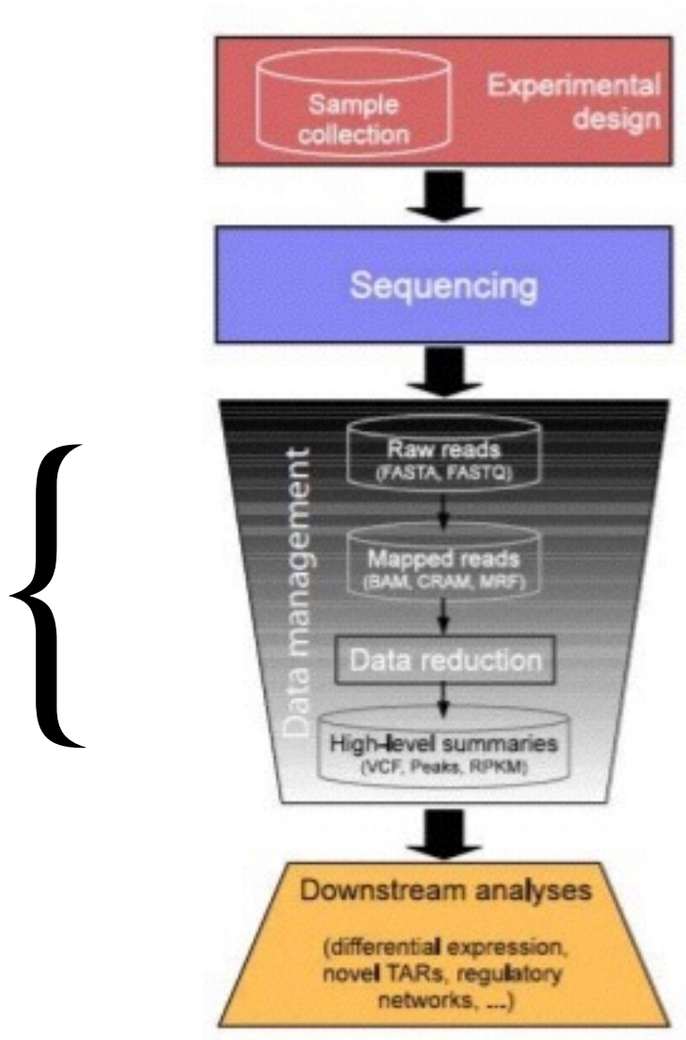


Approach: *de novo* resource generation from *D. opalescens* adult

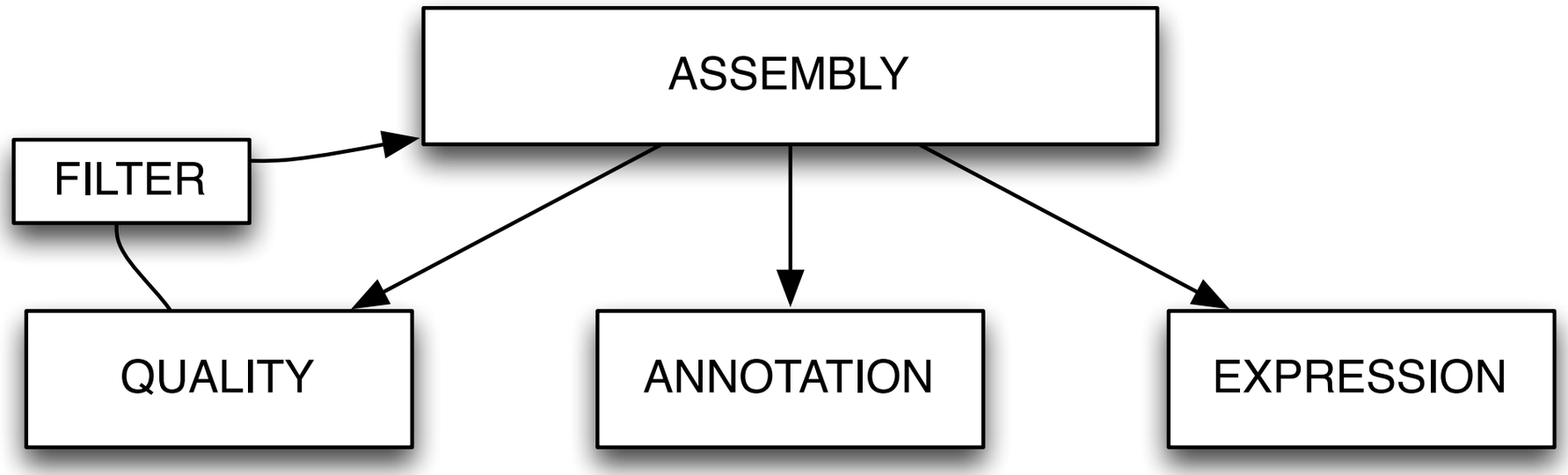
Vinther et al. 2012

Larger goal: develop software to create transcriptome resources





MakeMyTranscriptome

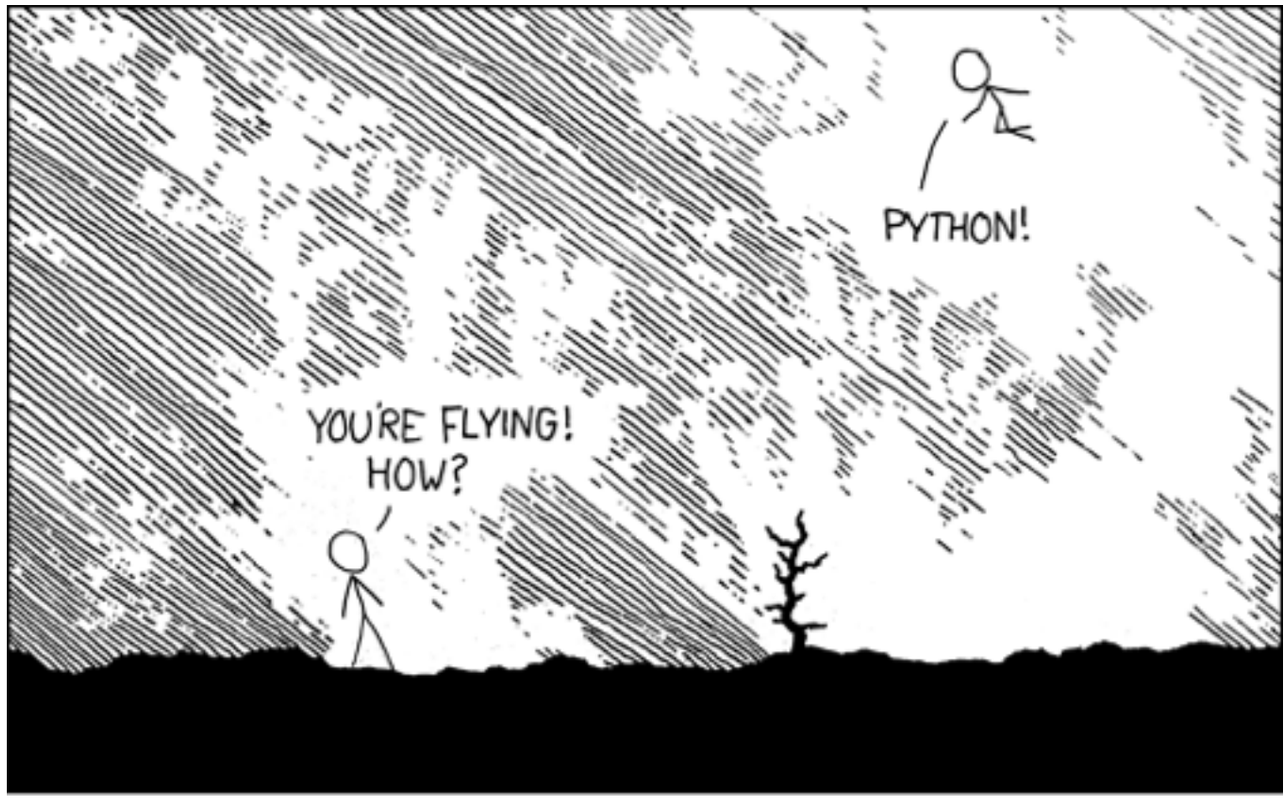


AUTOMATED via python “Supervisor”

Easily interchangeable components

Supervisor framework

- Lightweight python automation tool
- Is a python object; Simple to use
- Order tasks,only re-run with new info. etc
- CPU cap



xkcd

Supervisor tasks

Tasks for supervisor contain:

- **command** = actual command to be run
- **dependencies** = tasks that must be executed prior to this task
- **targets** = files that this task creates upon successful execution

optional: name, location for log files, # processors to use, email for updates

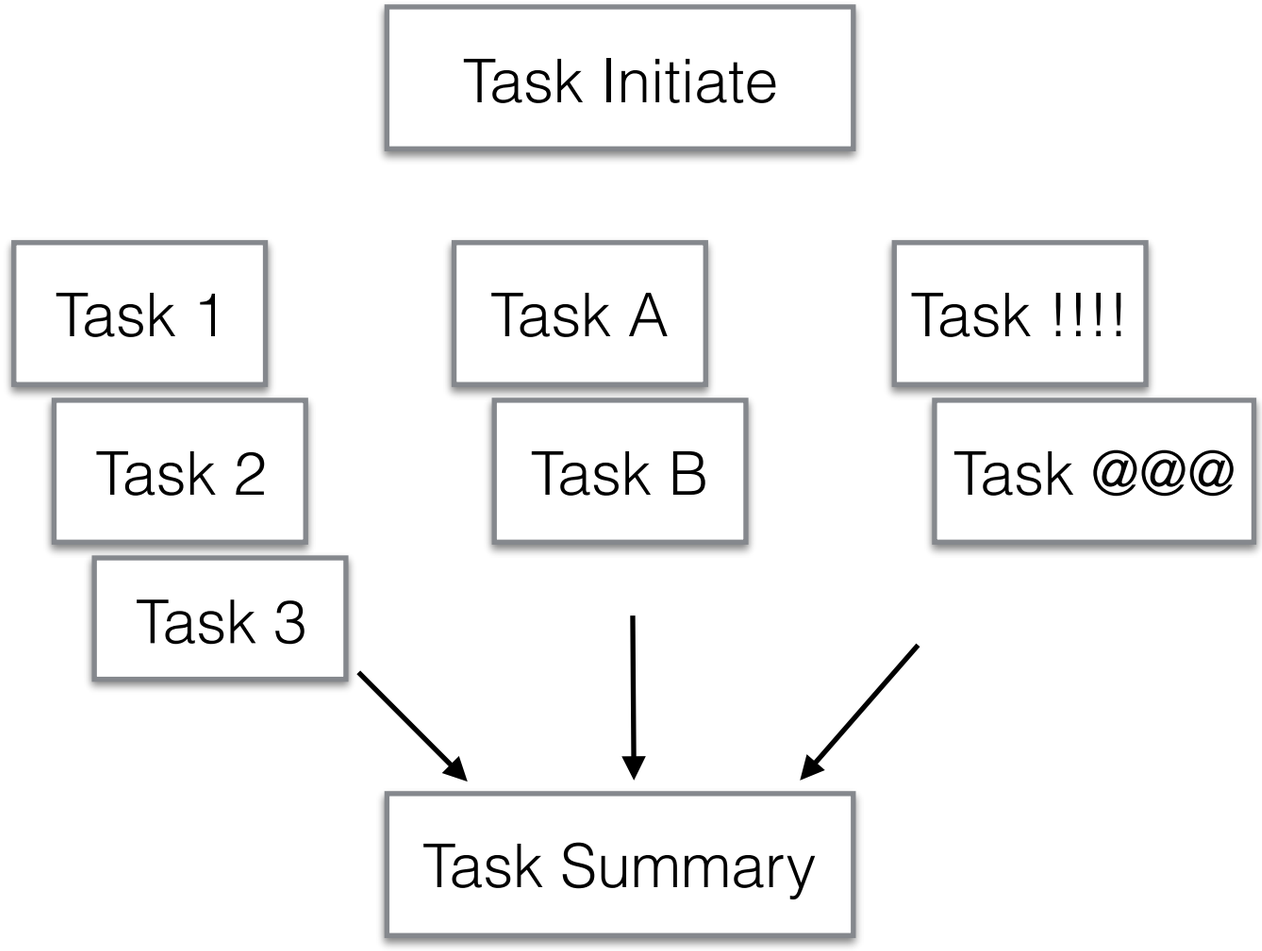
```
def assembly_stats_task(out_dir,assembly,tasks):

    trgs = ['{0!s}/assembly_stats.json'.format(out_dir)]
    cmd = 'python {0!s}/assembly_stats.py {1!s} > {2!s}'.format(fg.PATH_SCRIPTS,assembly,trgs[0])
    name = 'assembly_stats'
    out,err = fg.GEN_LOGS(name)
    return Task(command=cmd,dependencies=tasks,targets=trgs,name=name,stdout=out,stderr=err)
```

Supervisor tasks

Tasks for supervisor contain:

- **command** = actual command to be run
- **dependencies** = tasks that must be executed prior to this task
- **targets** = files that this task creates upon successful execution



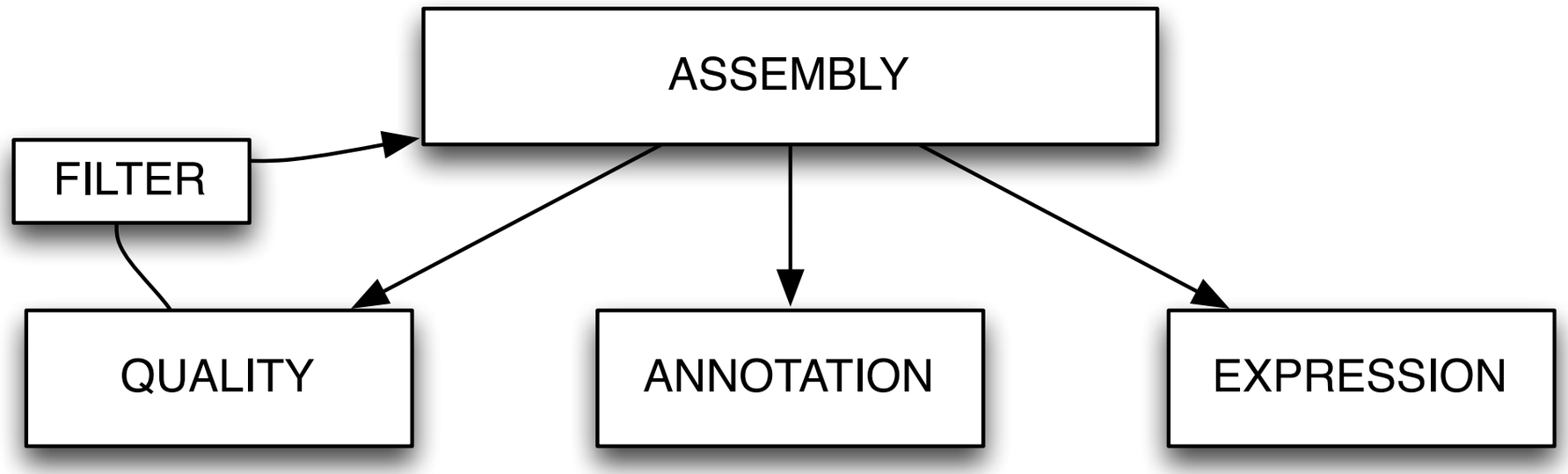
Automate with supervisor

Generate a supervisor:

- Define tasks; add them to a list
- create a Supervisor object containing these tasks
- **supervisor.run()**

```
def gen_quality_supervisor(transrate_fq1, transrate_fq2, transrate_unp, dependency_set, busco_refs, cpu=12, (
    tasks = []
    for busco_ref in busco_refs:
        tasks.append(fq.busco_task(assembly_path, assembly_name, out_dir, busco_ref, int(cpu/2), []))
    return Supervisor(tasks=tasks,dependencies=dependency_set)
```

MakeMyTranscriptome



Helper modules:

- setup
- databases
- tools
- compare assemblies

MakeMyTranscriptome

Quick Start Guide:

MMT is a convenience tool that runs a number of bioinformatics tools in an automated fashion. In order to use it, you'll have to download these other tools, and put these into your PATH. If you're working on a Linux machine, MMT can download the tools for you; otherwise, it will assess the tools you have available and print recommended installation instructions.

First, download MMT via git (below) or click the "Download ZIP" button.

```
git clone https://github.com/bluegenes/MakeMyTranscriptome.git
```

cd into the MakeMyTranscriptome directory.

```
cd MakeMyTranscriptome
```

In order to run MMT, the general structure is:

```
mmt [TOOL_SELECTOR] [ARGUMENTS]
```

	A	B	C	D
1	name	read1	read2	condition
2	msA11eye	msA11eye_1.fastq	msA11eye_2.fastq	control
3	msA3_A8eye	msA8eye_1.fastq	msA8eye_2.fastq	control
4	msB11_B23eye	msB11_B23eye_1.fastq	msB11_B23eye_2.fastq	lowpH
5	msB20eye	msB20eye_1.fastq	msB20eye_2.fastq	lowpH
6	msC13eye	msC13eye_1.fastq	msC13eye_2.fastq	lowpHhighT
7	msC9_C19eye	msC9_C19eye_1.fastq	msC9_C19eye_2.fastq	lowpHhighT

For example, to run a full test of the pipeline on sample data:

```
mmt full -test
```

Assembly

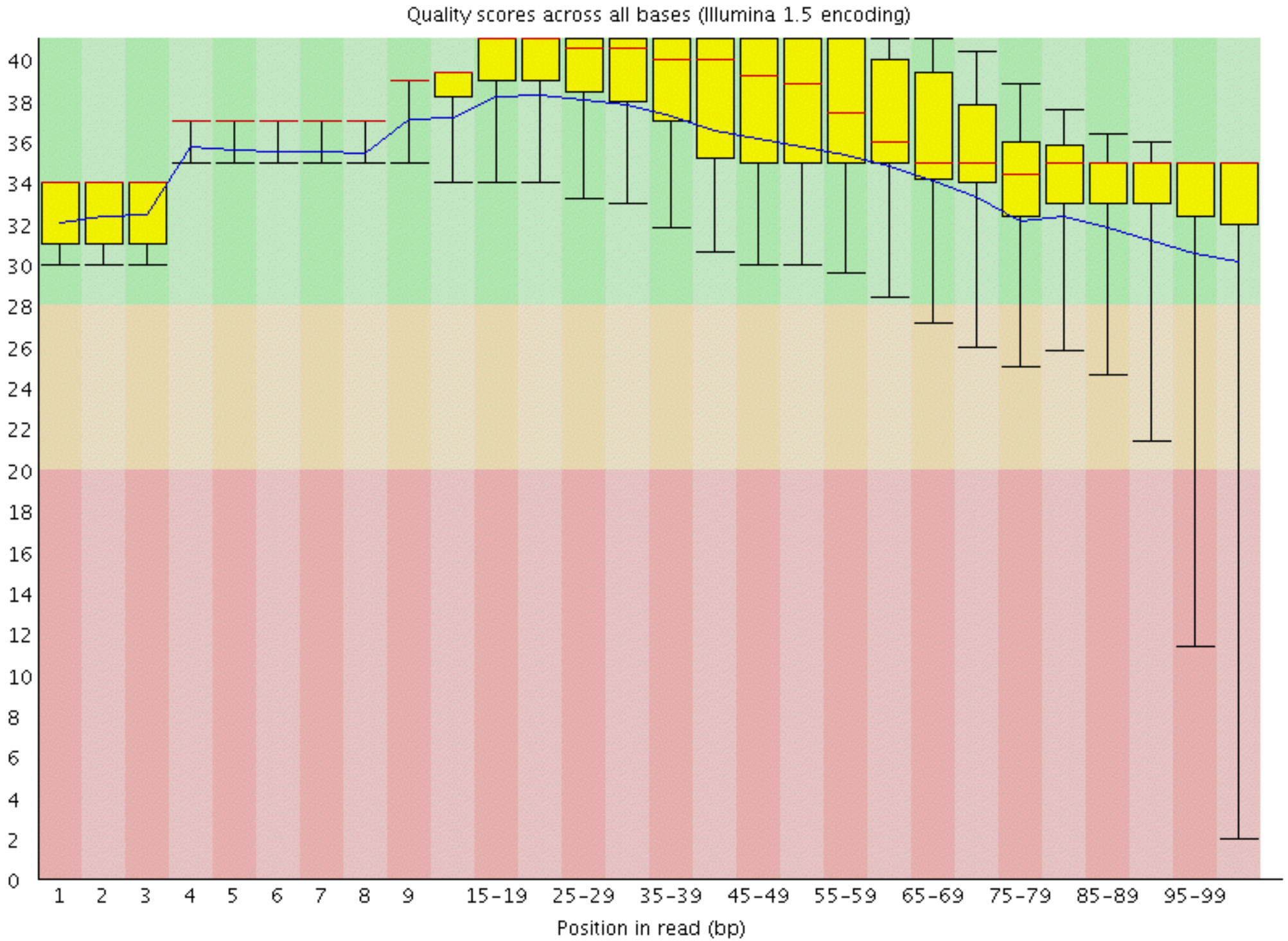
Workflow:

- Quality control
  - Remove adapters
  - Read trimming
  - Digital normalization
- Assembly
  - Trinity
  - rnaSPAdes

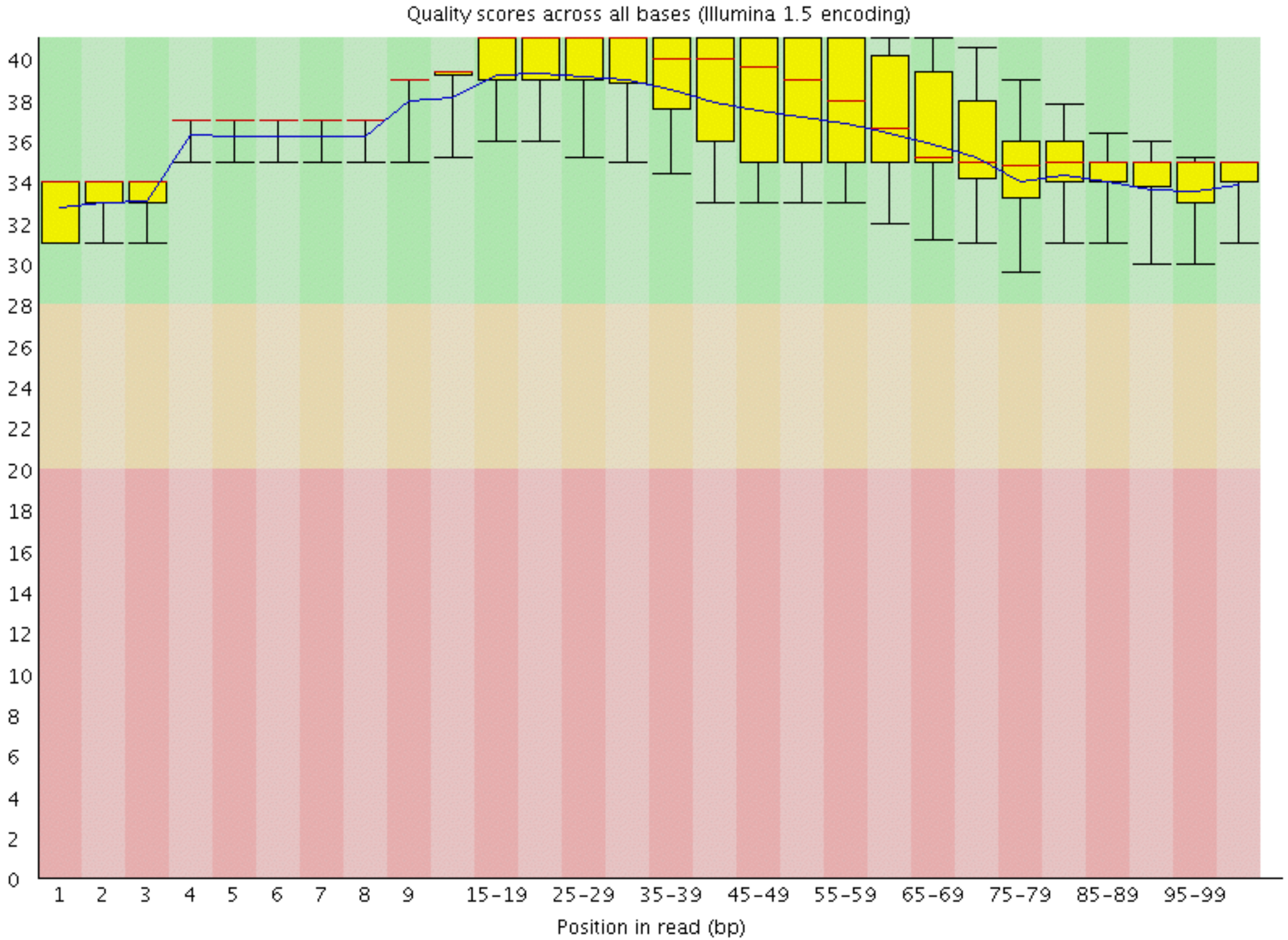
INPUT:

- FastQ reads
- CSV (optional)

Quality Control: FastQC



Quality Control: FastQC



Reference Generation Method:

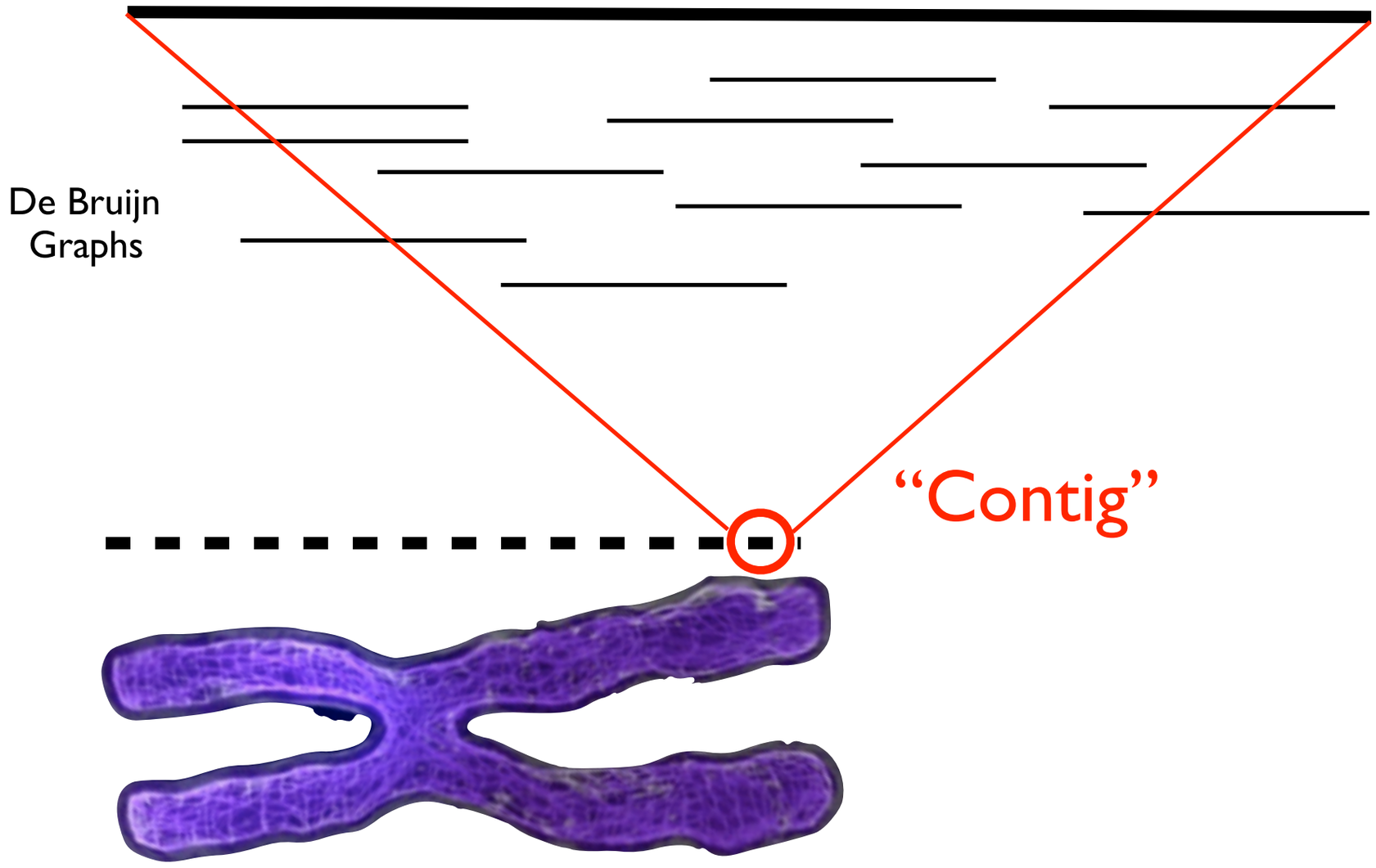
Genome-Wide Gene Expression: RNA-Seq

Raw Data:

"transcripts"
"reads"

ATCACAGTGGGCACTCCATTCGTACAGTACA

Create *De Novo* Reference Assembly:



each contig should represent an expressed gene

Reference Generation Method:

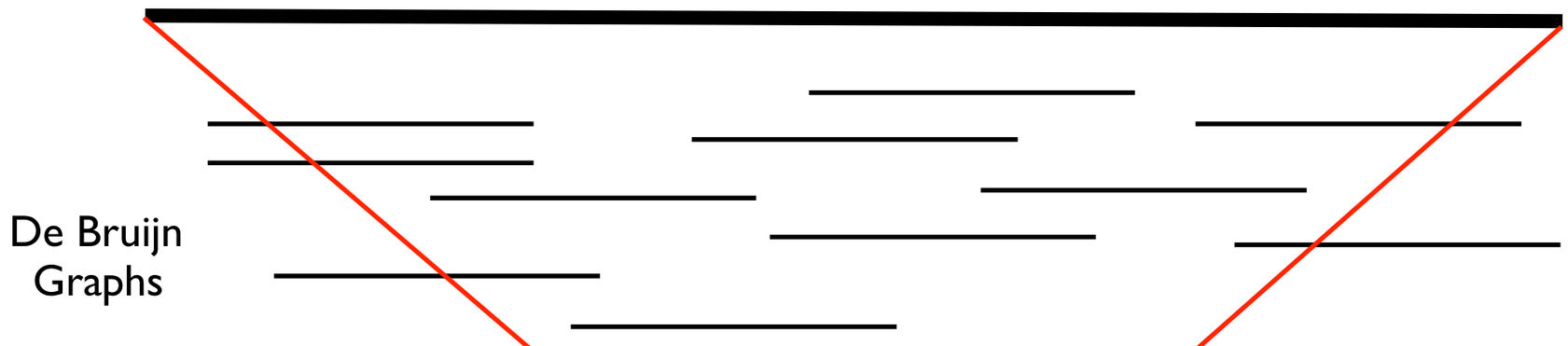
Genome-Wide Gene Expression: RNA-Seq

Raw Data:

"transcripts"
"reads"

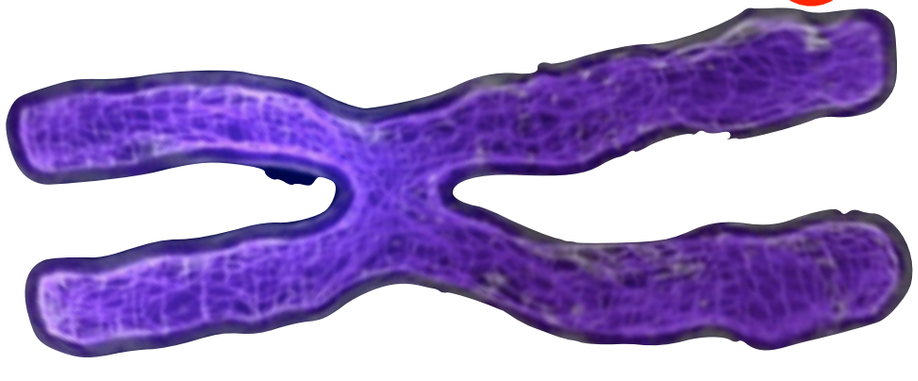
ATCACAGTGGGCACTCCATTCGTACAGTACA

Create *De Novo* Reference Assembly:



Differences in Assembly Algorithms Affect Reconstruction of Genes

"Contig"



Annotation

Tools

TransDecoder

DIAMOND

HMMER

NCBI BLAST

Databases

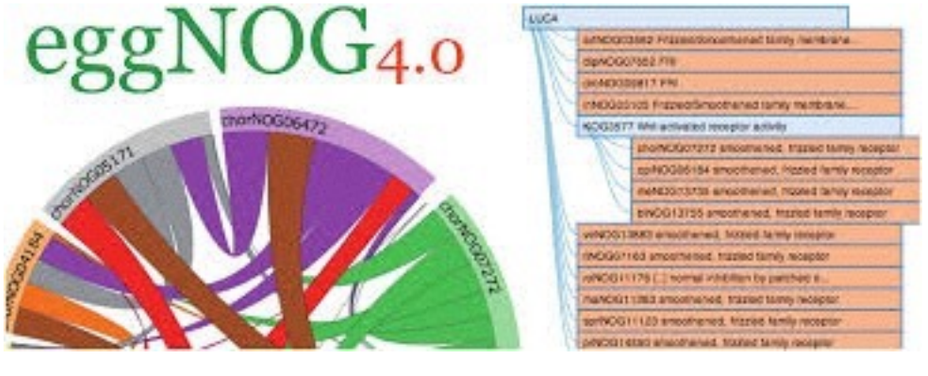
UniProt

Pfam

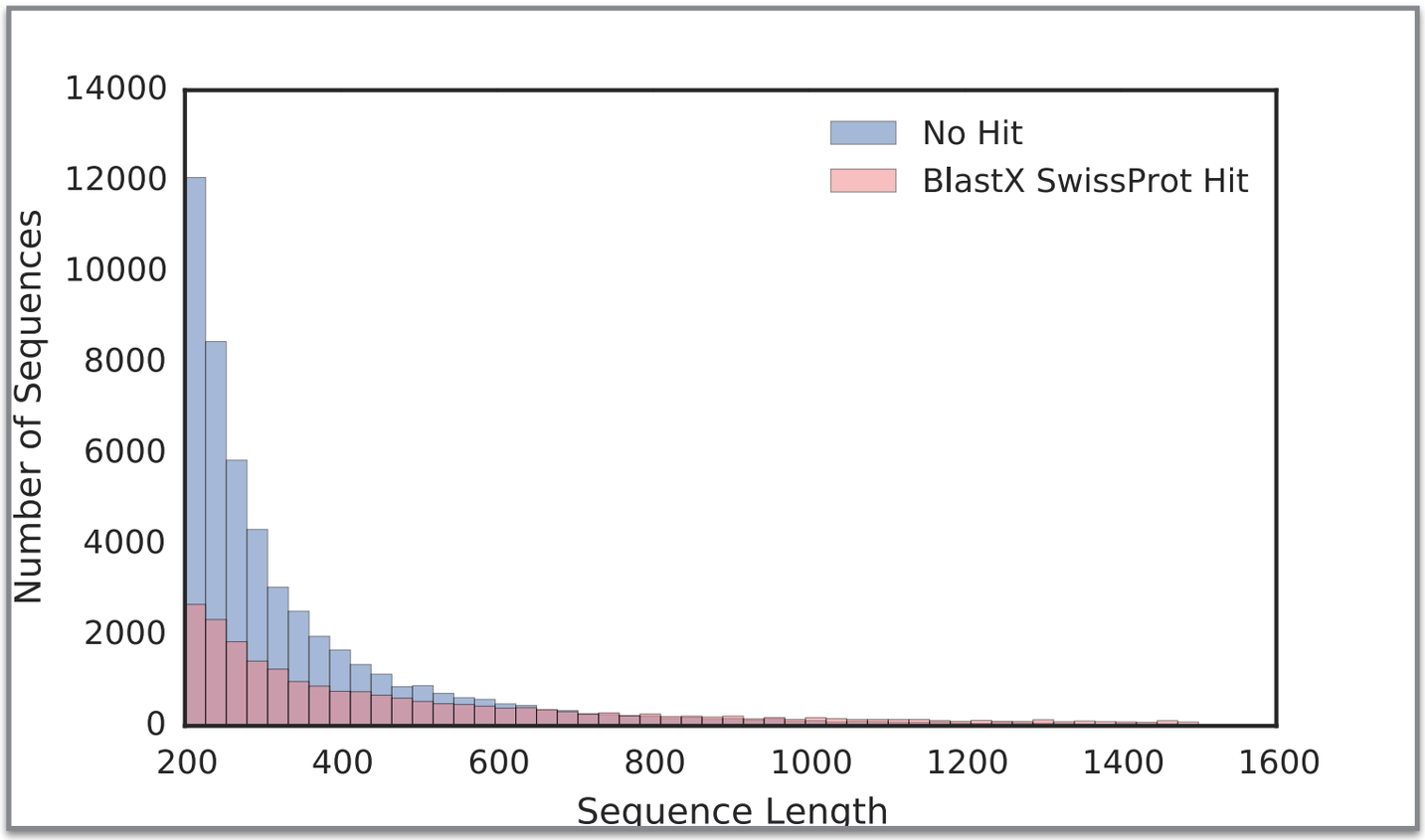
NR (GenBank)

GO the Gene Ontology

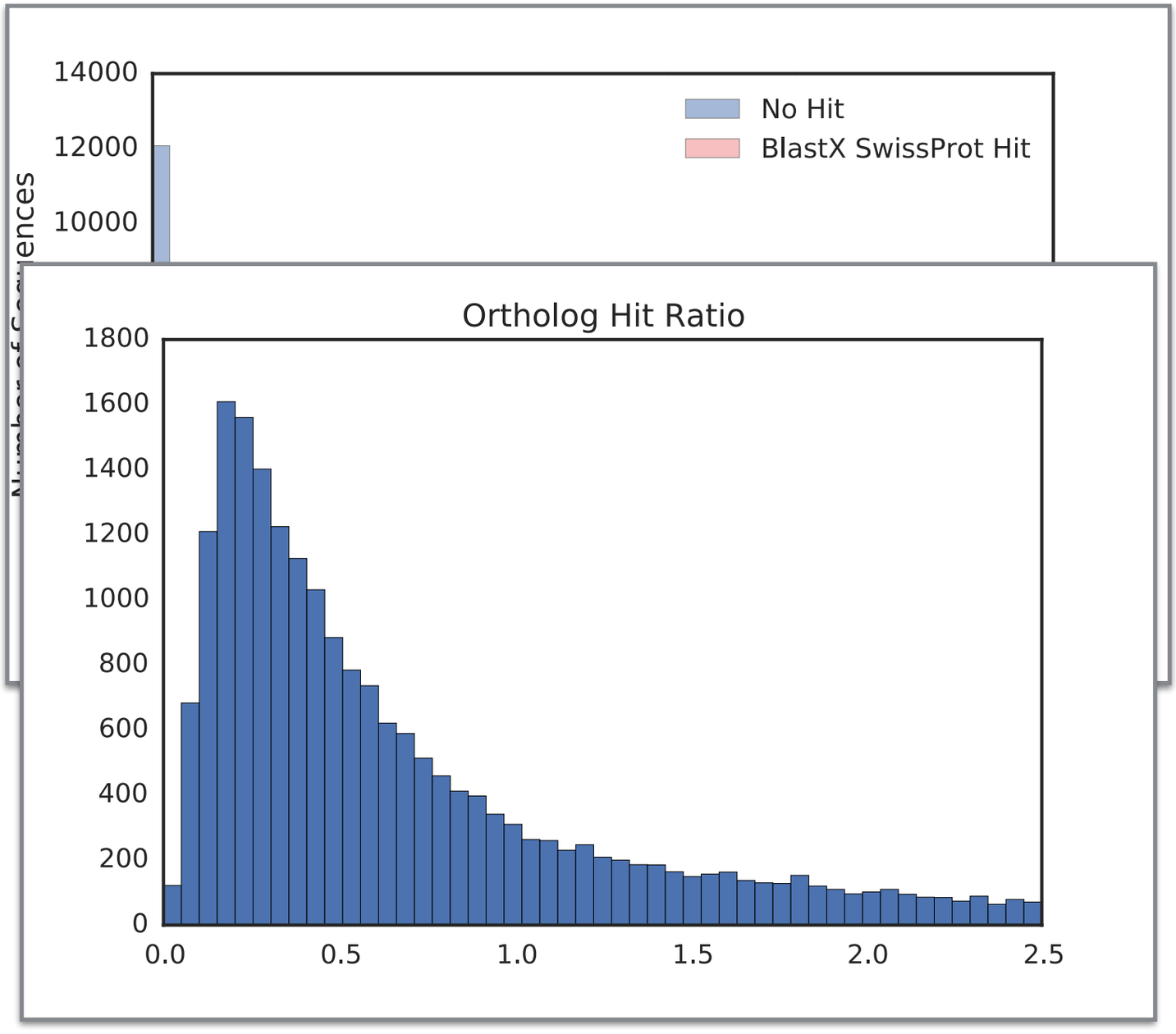
KEGG Kyoto Encyclopedia of Genes and Genomes



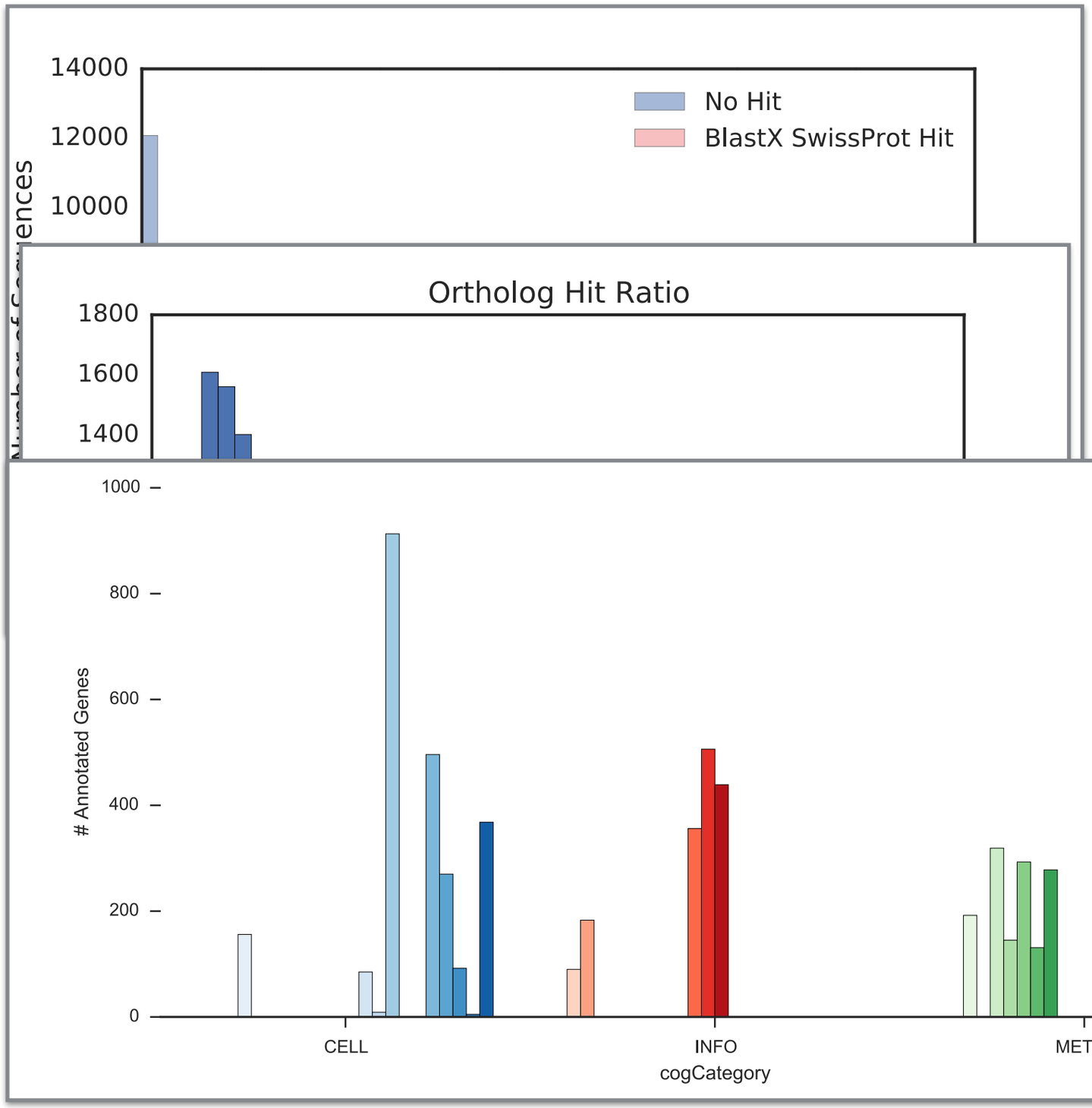
Annotation Summary



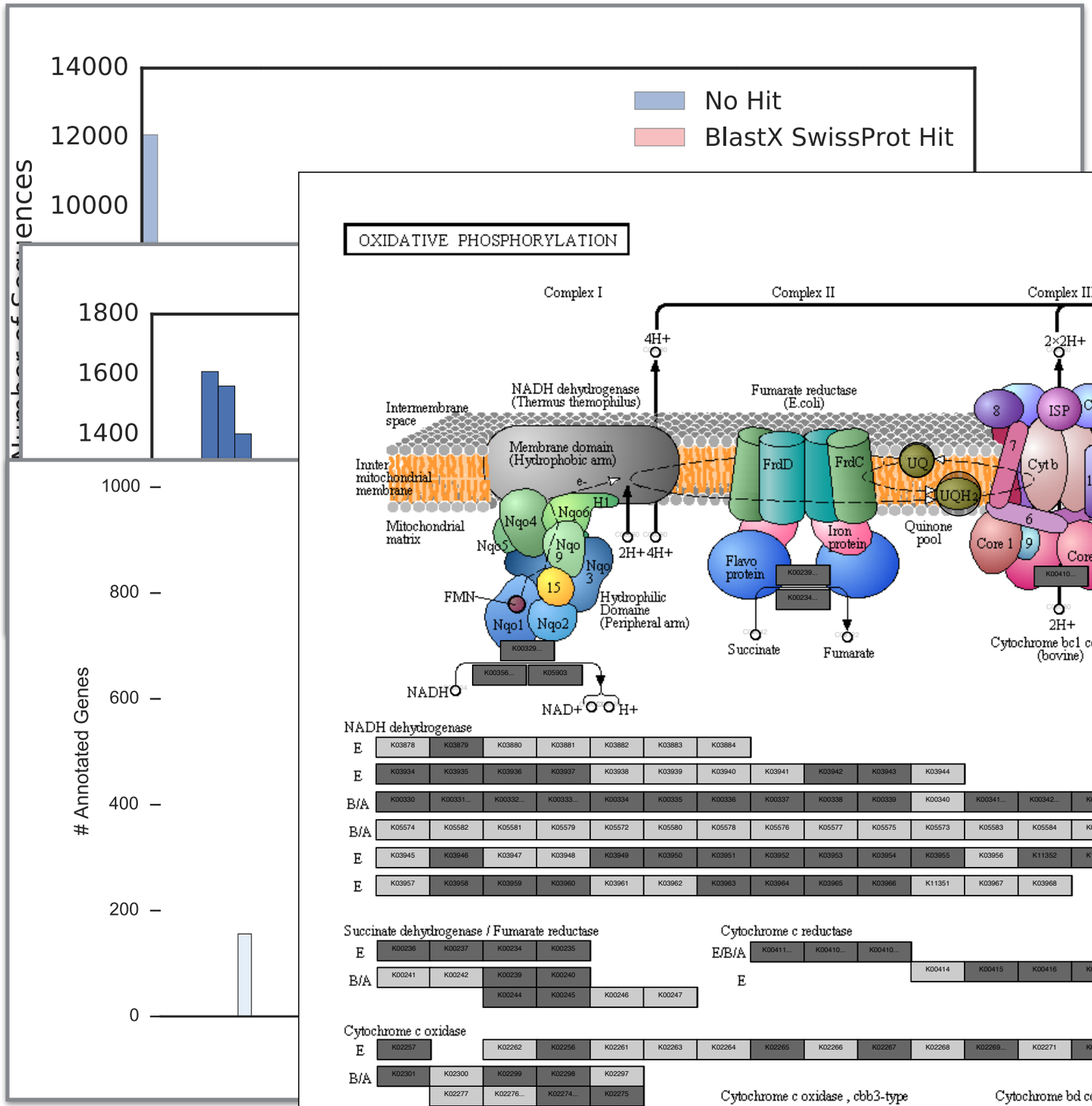
Annotation Summary



Annotation Summary



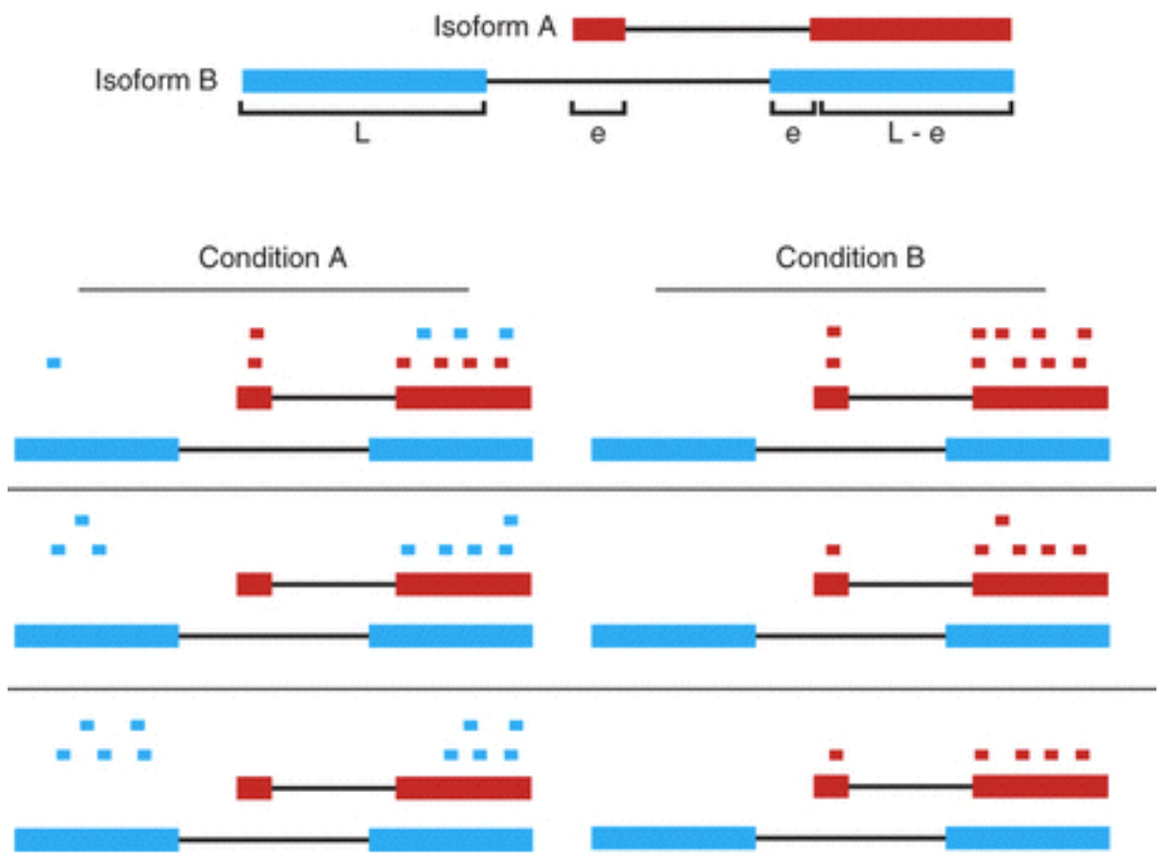
Annotation Summary



Expression

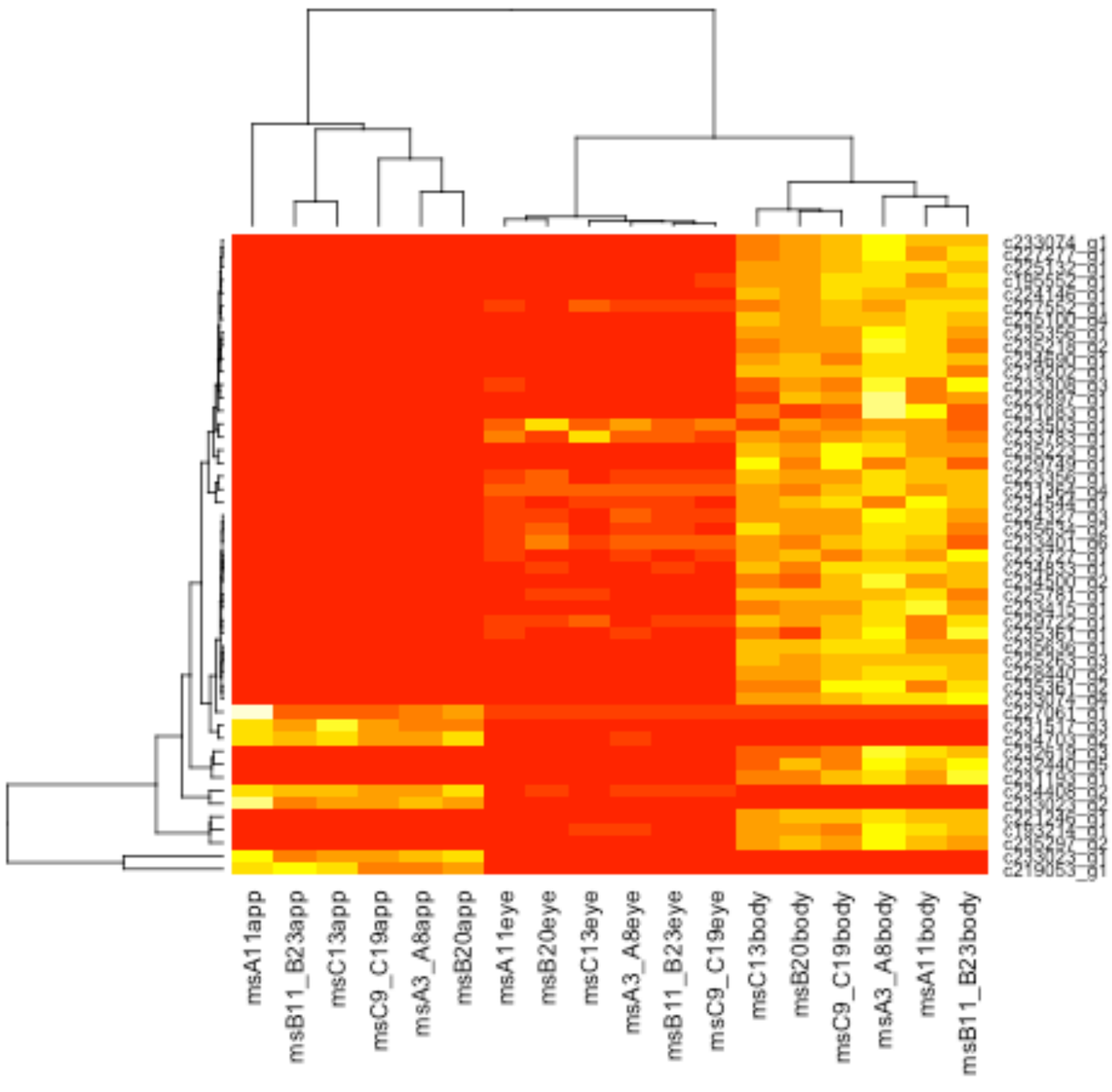
Workflow:

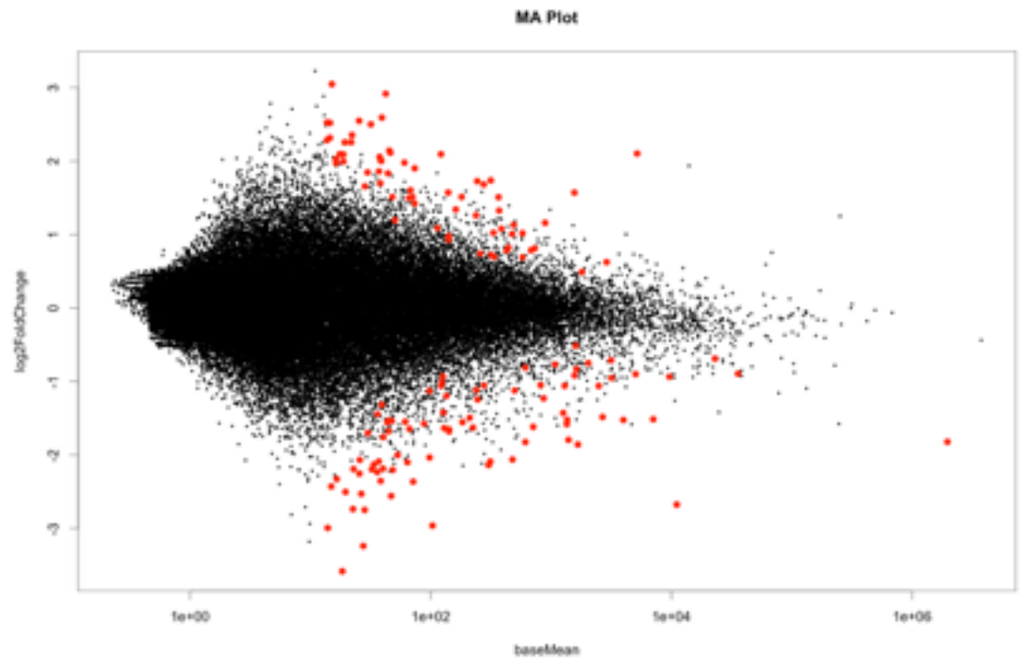
- Mapping & Quantification
  - Bowtie2 — > eXpress
  - Bowtie2 — BEDtools intersect
  - **Salmon**
- Differential Expression
  - DESeq2
  - Sleuth (partial)

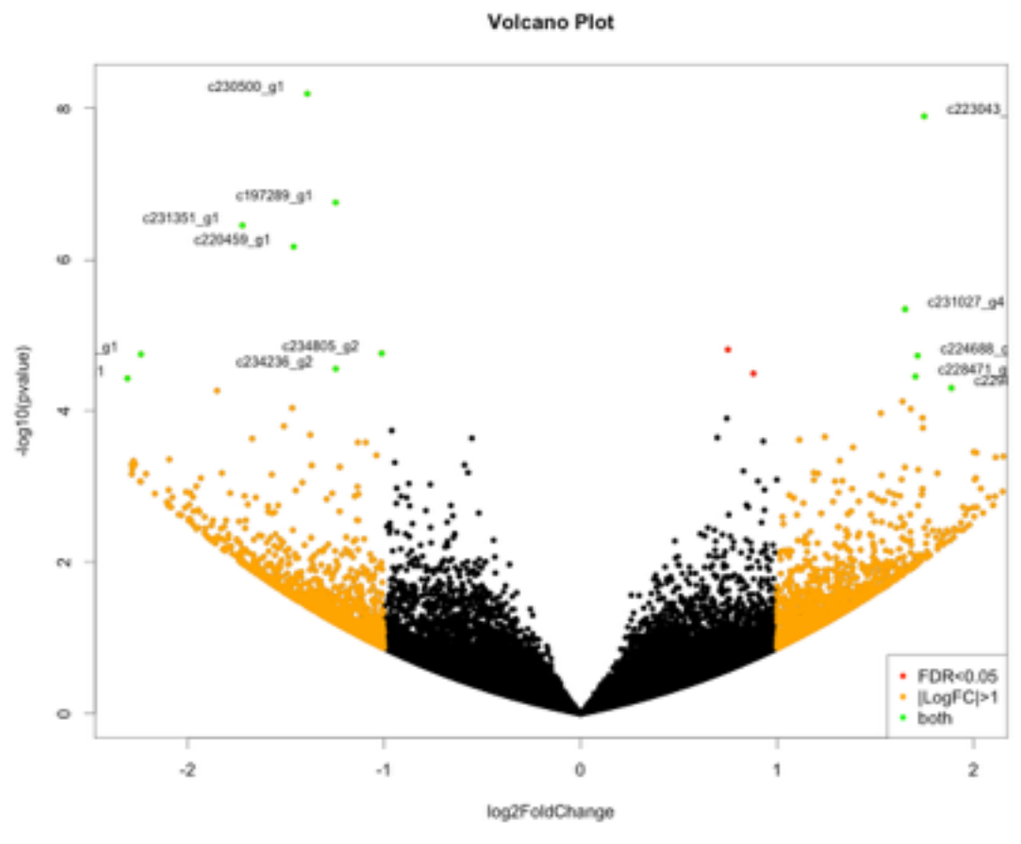


Trapnell et al, 2013. Differential Analysis of Gene Regulation at Transcript Level with RNA-Seq

Expression Output







Optimizing RNA-Seq

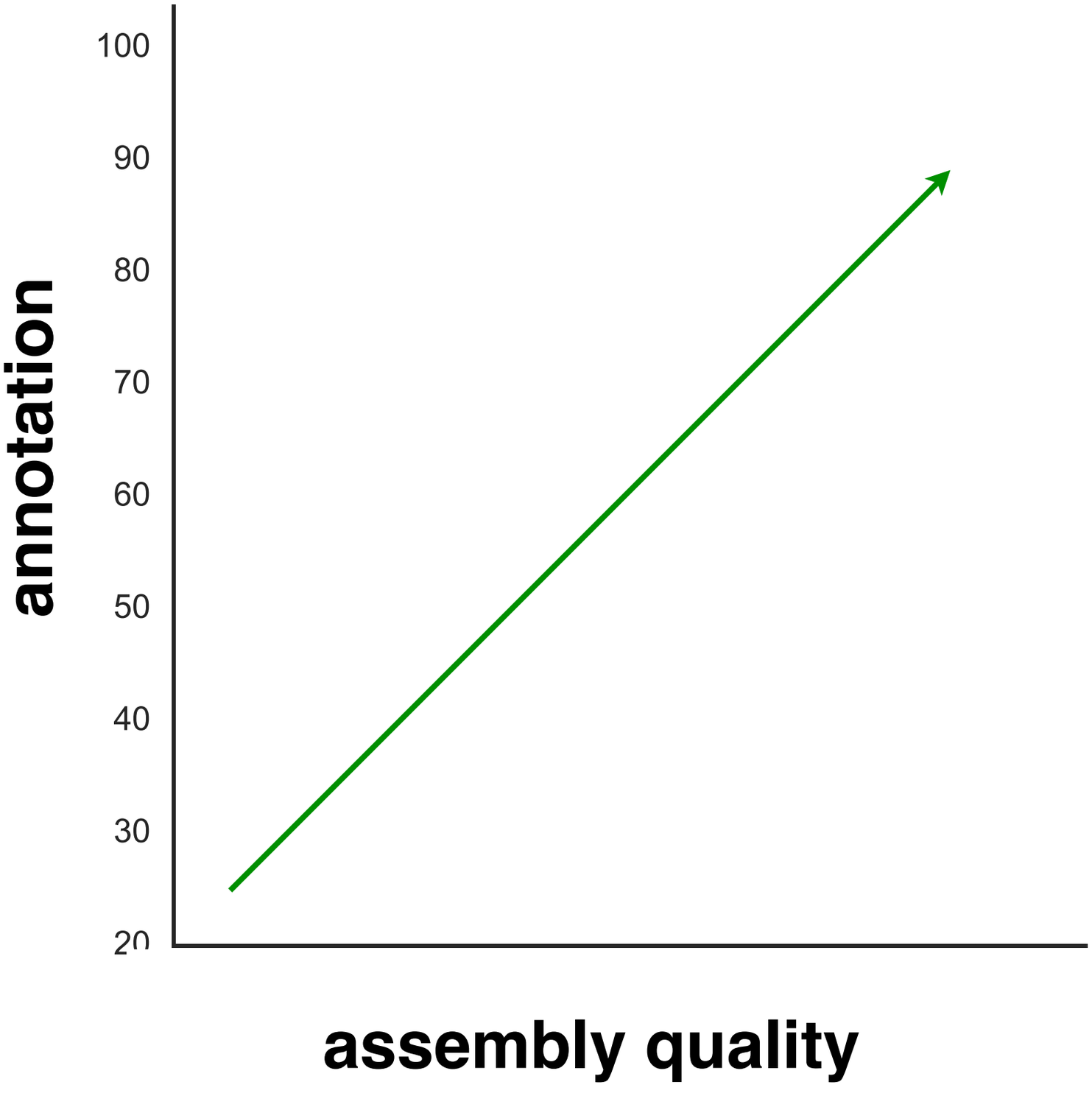
How deep do we need to sequence?

Do we really need to spend $ on long, PE reads?

But first…

a note on Assembly Metrics…

How to Assess Results?



Transcriptome Evaluation Methods

Length-Based:

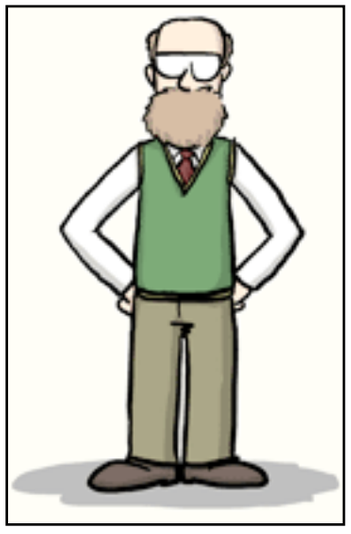
50%

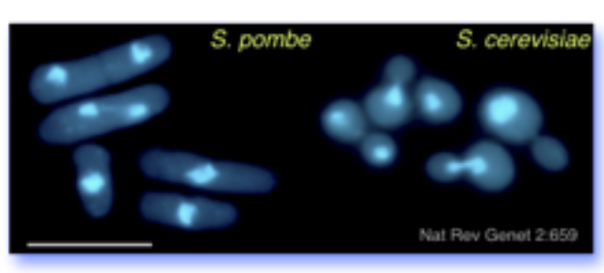
N50

50%

Annotation-Based:

Core Eukaryotic Genes

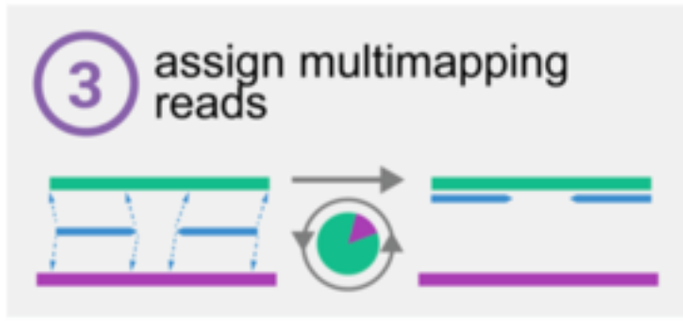




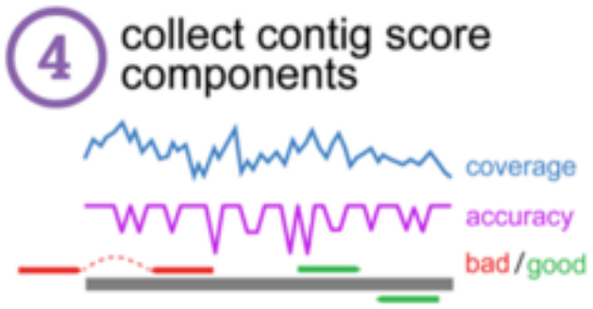
Evaluate *De Novo* Assembly Algorithms on the same input data to obtain an optimal assembly pipeline

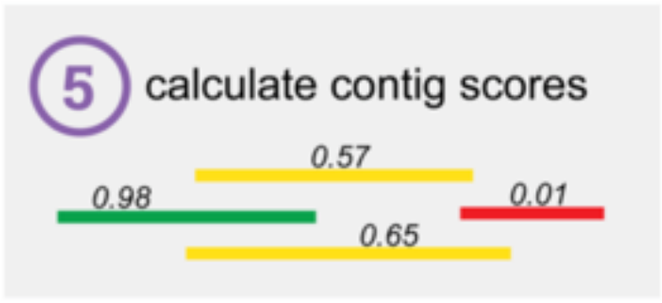
Assembly Quality

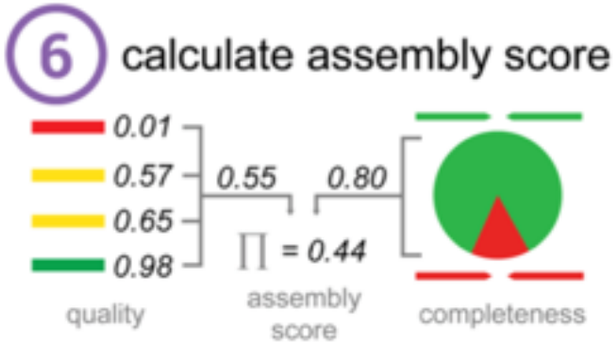
transrate
understand your transcriptome assembly



BUSCO







Arthropods: Vertebrates: Fungi: Bacteria:

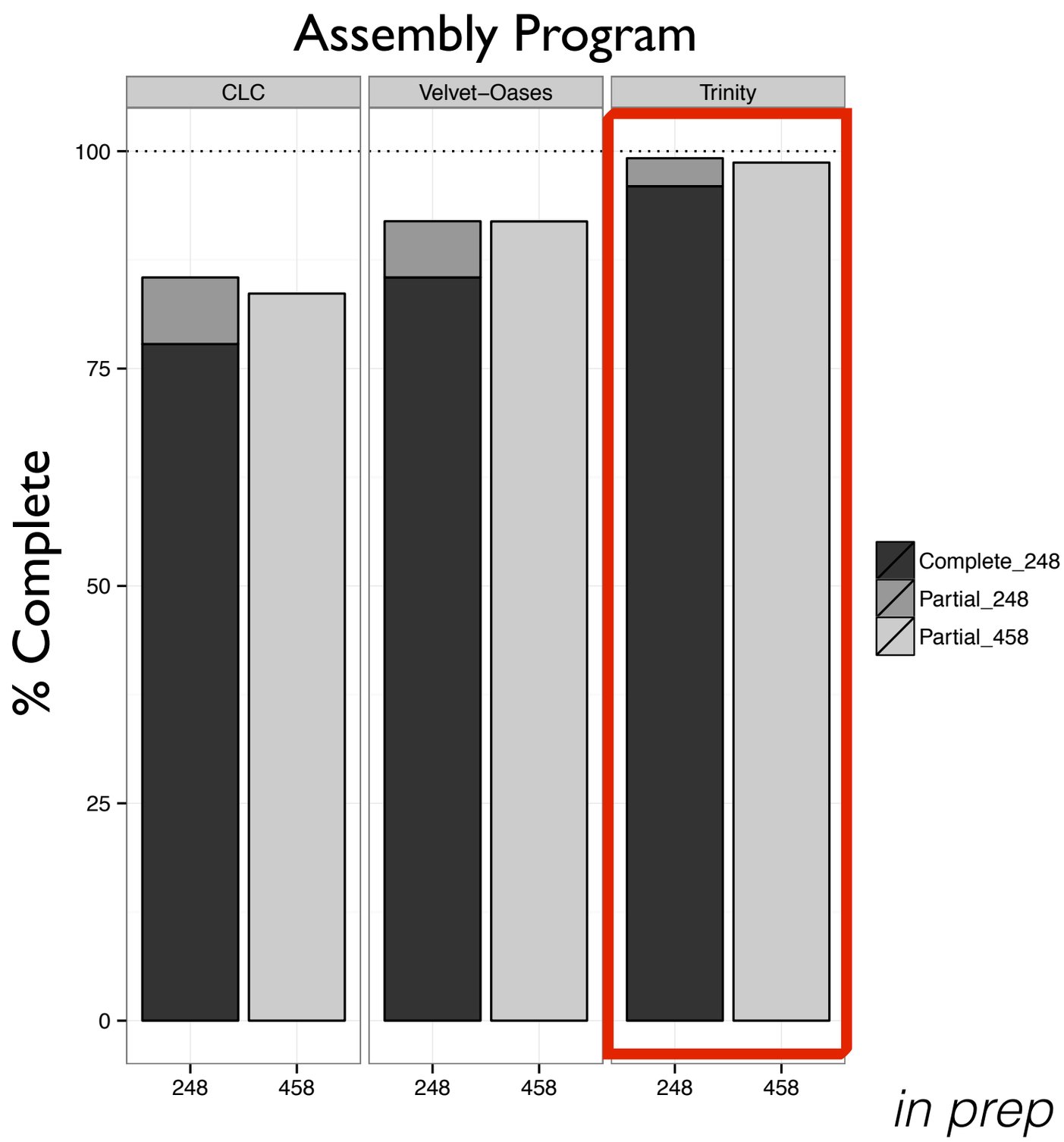
Metazoans: & & Eukaryotes: & & &

Plants: Early access available upon request.

Trinity Assembler Maximized % Complete

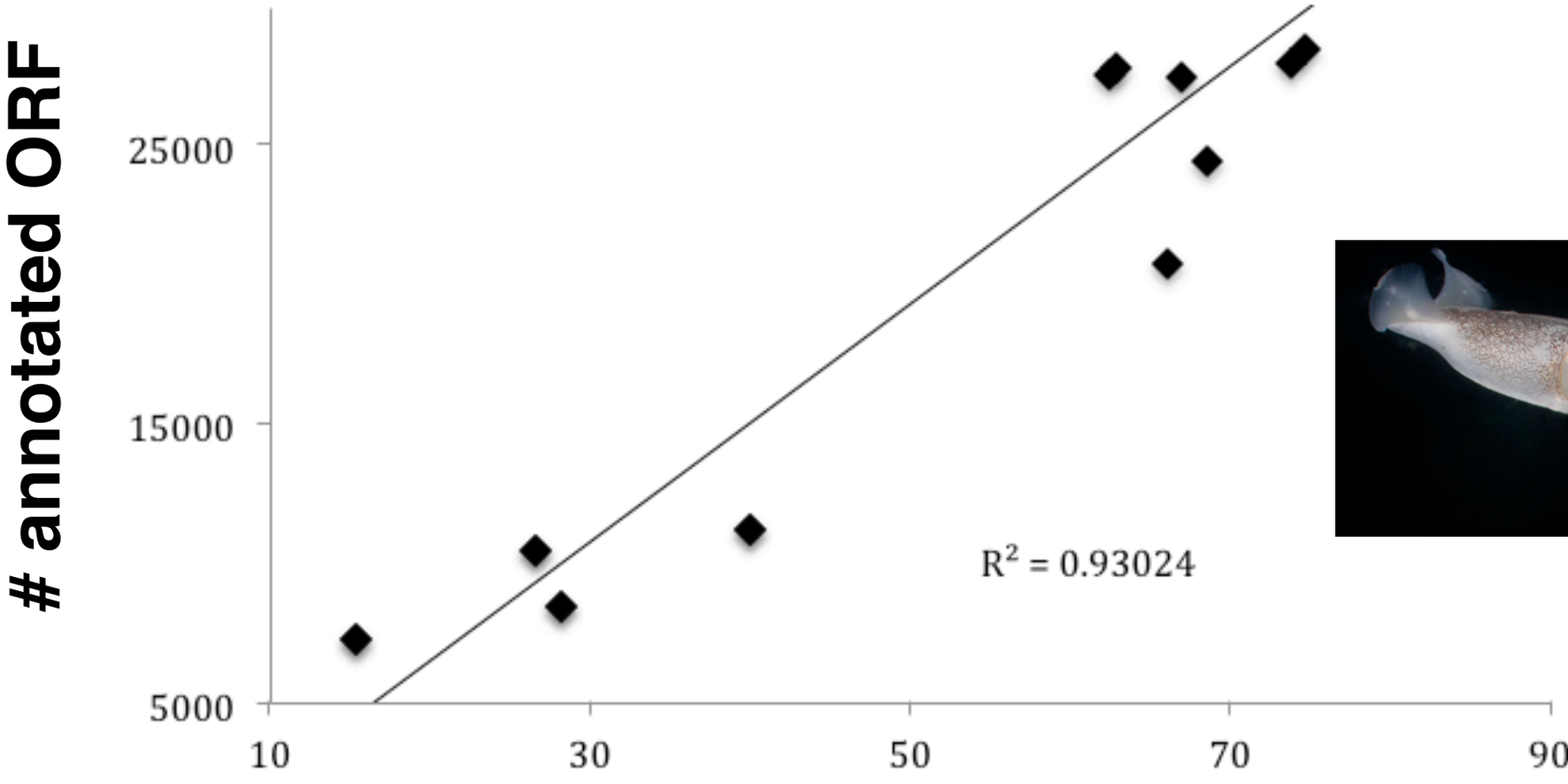
Assembly Algorithms:

- CLC
- Velvet-Oases
- Trinity



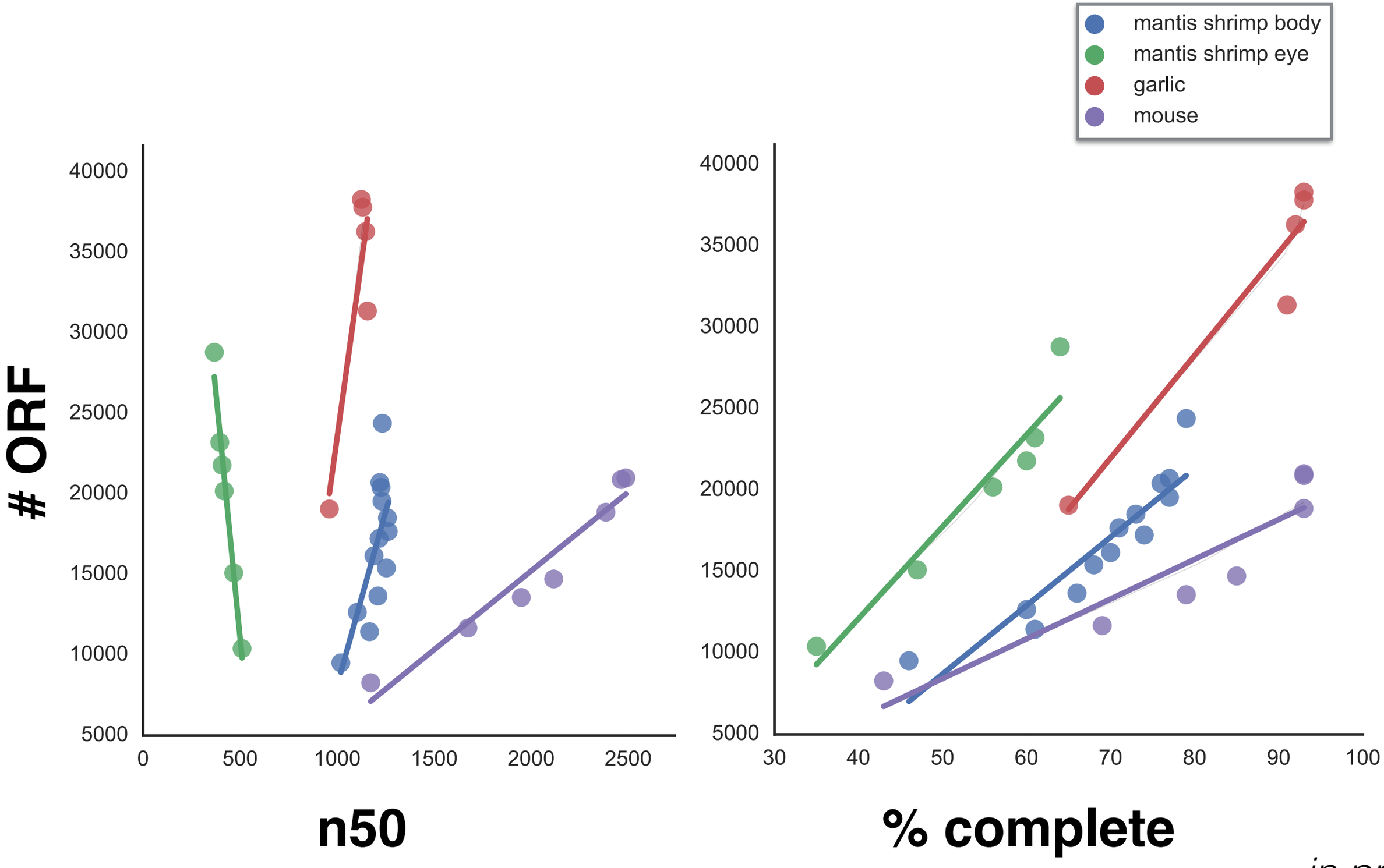
in prep

Annotation-based metrics



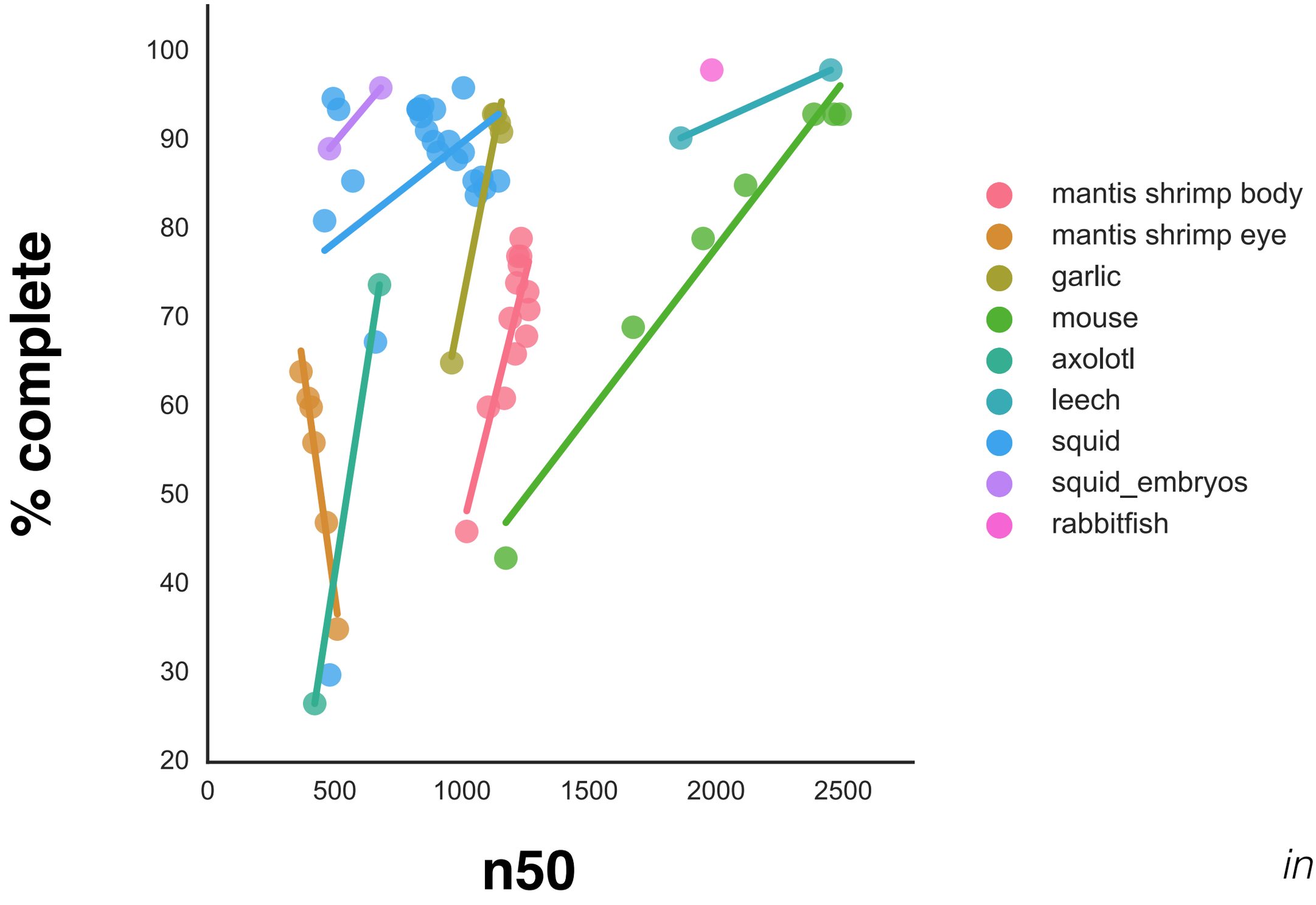
in prep

Annotation Metics > n50



in prep

n50 does not correlate with annotation

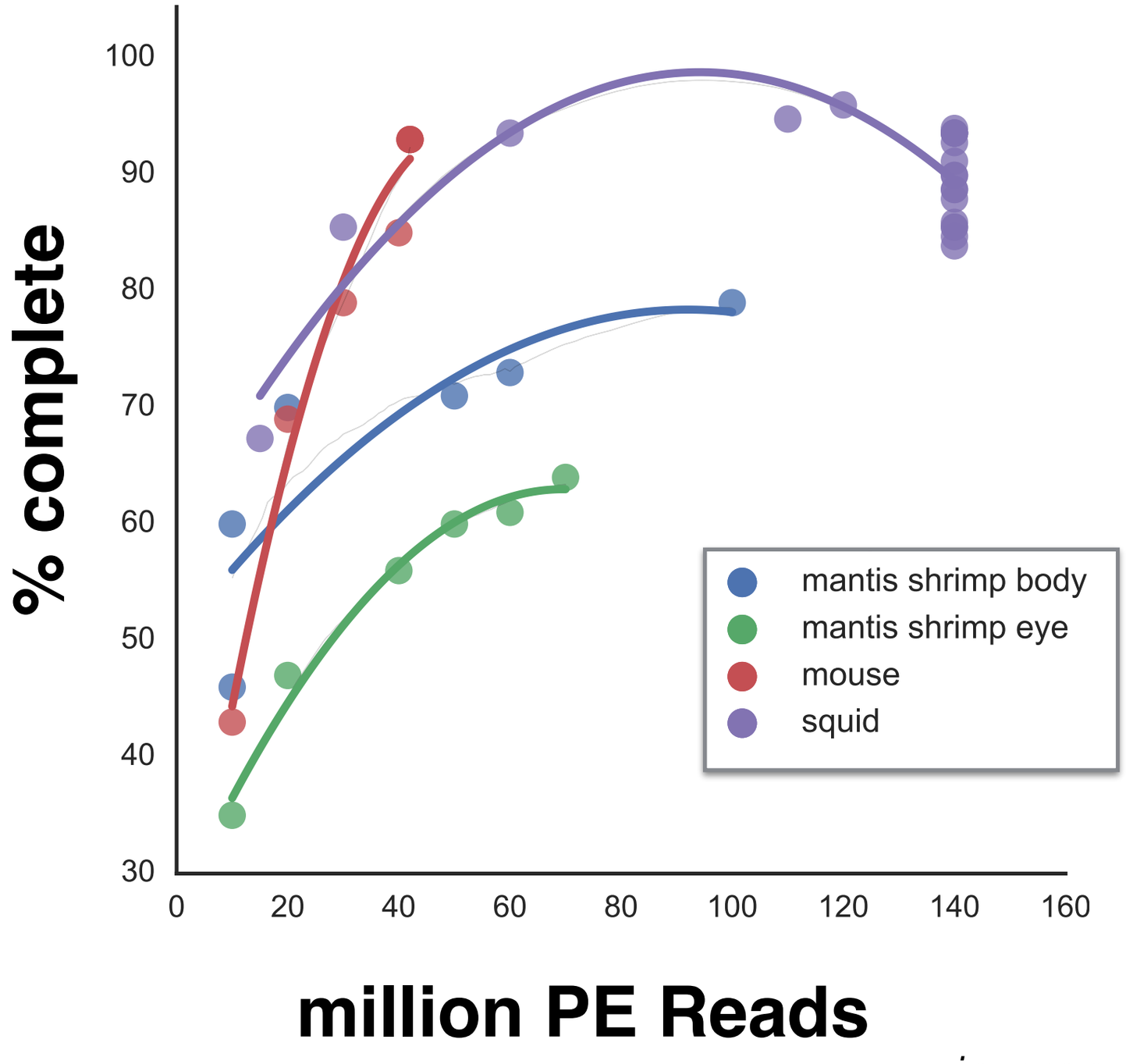


in prep

How much to sequence?

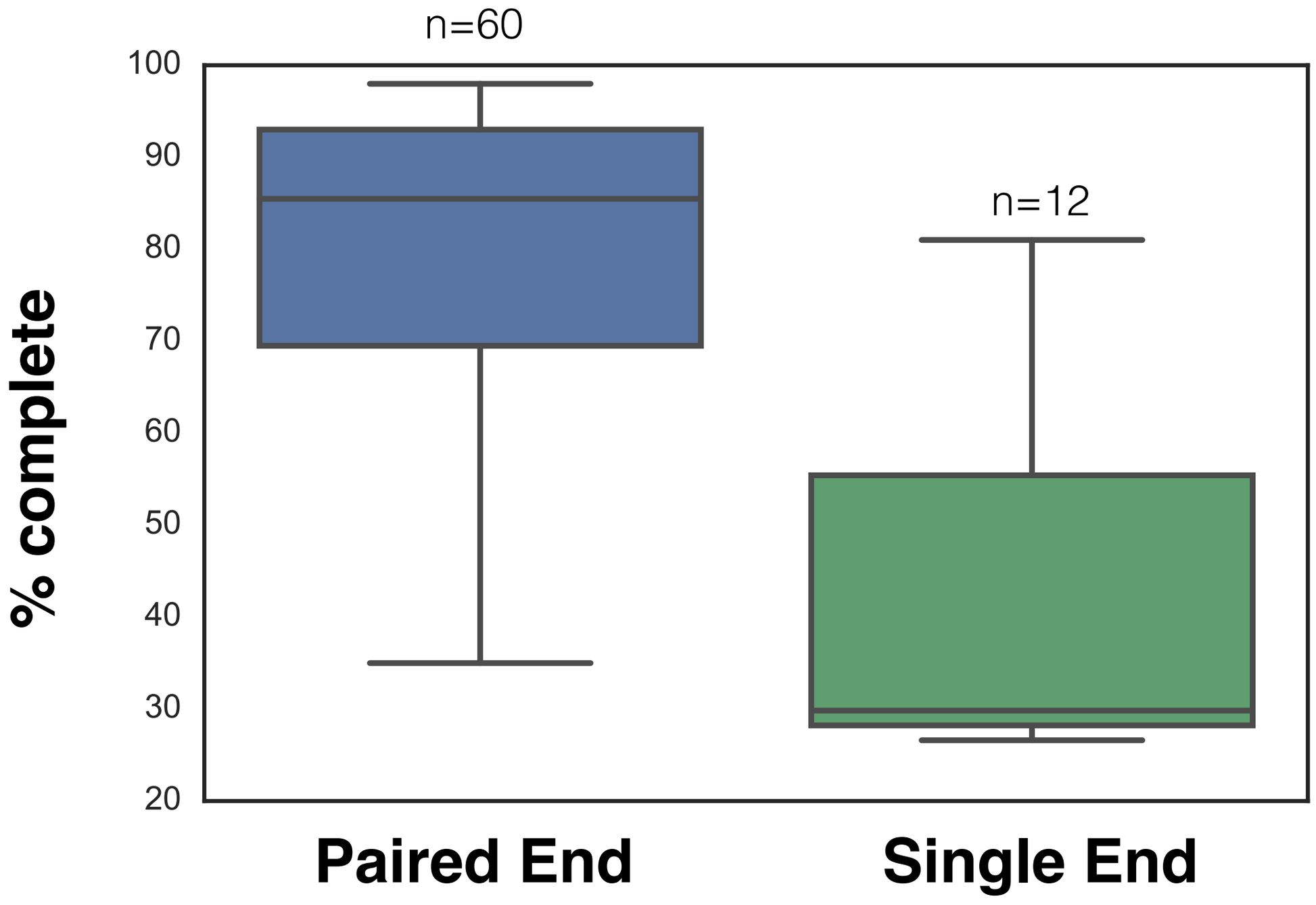
Recommendation:

- ~50-70 million *cleaned* reads
- Digitally normalize large datasets



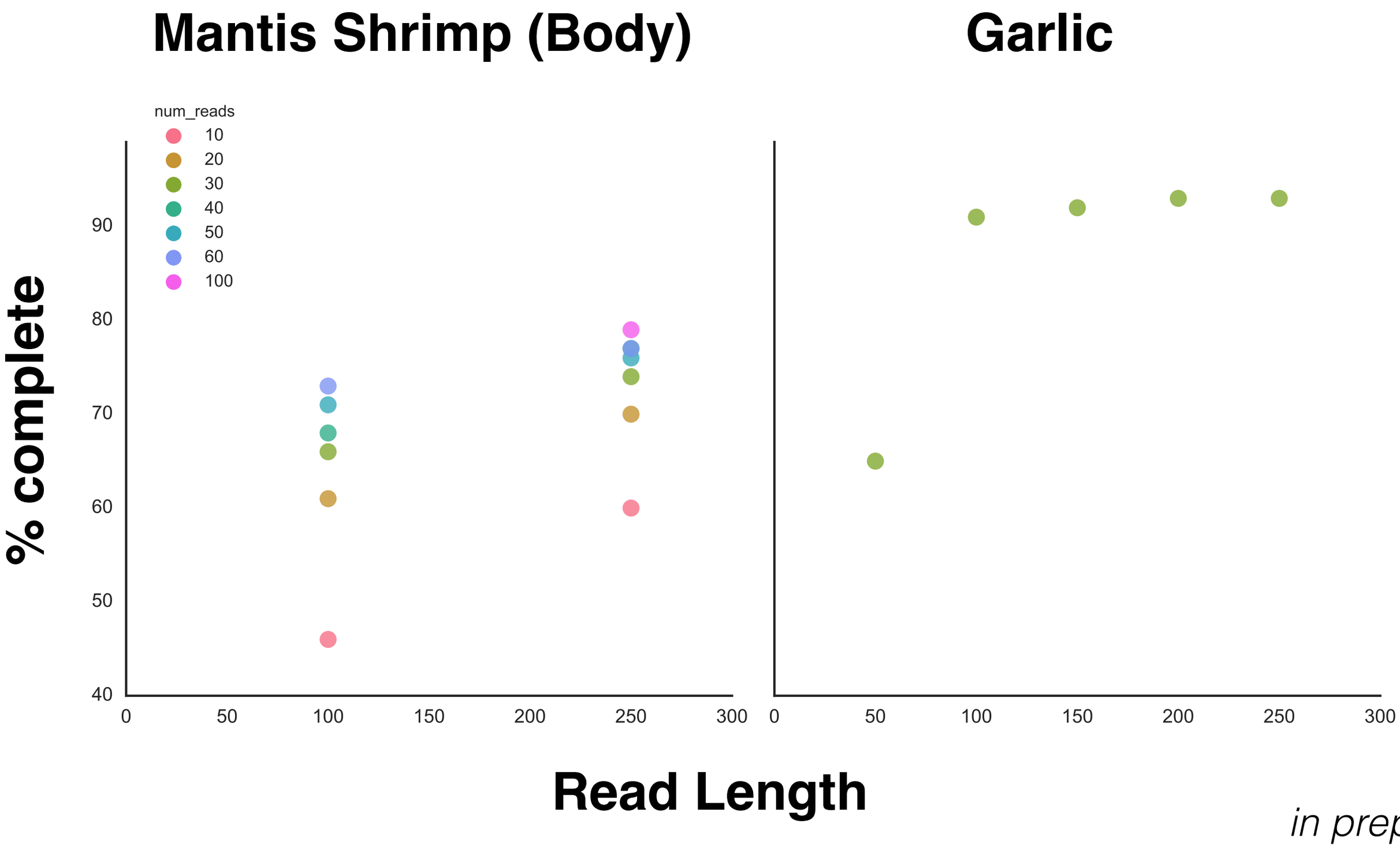
in prep

SE vs PE?



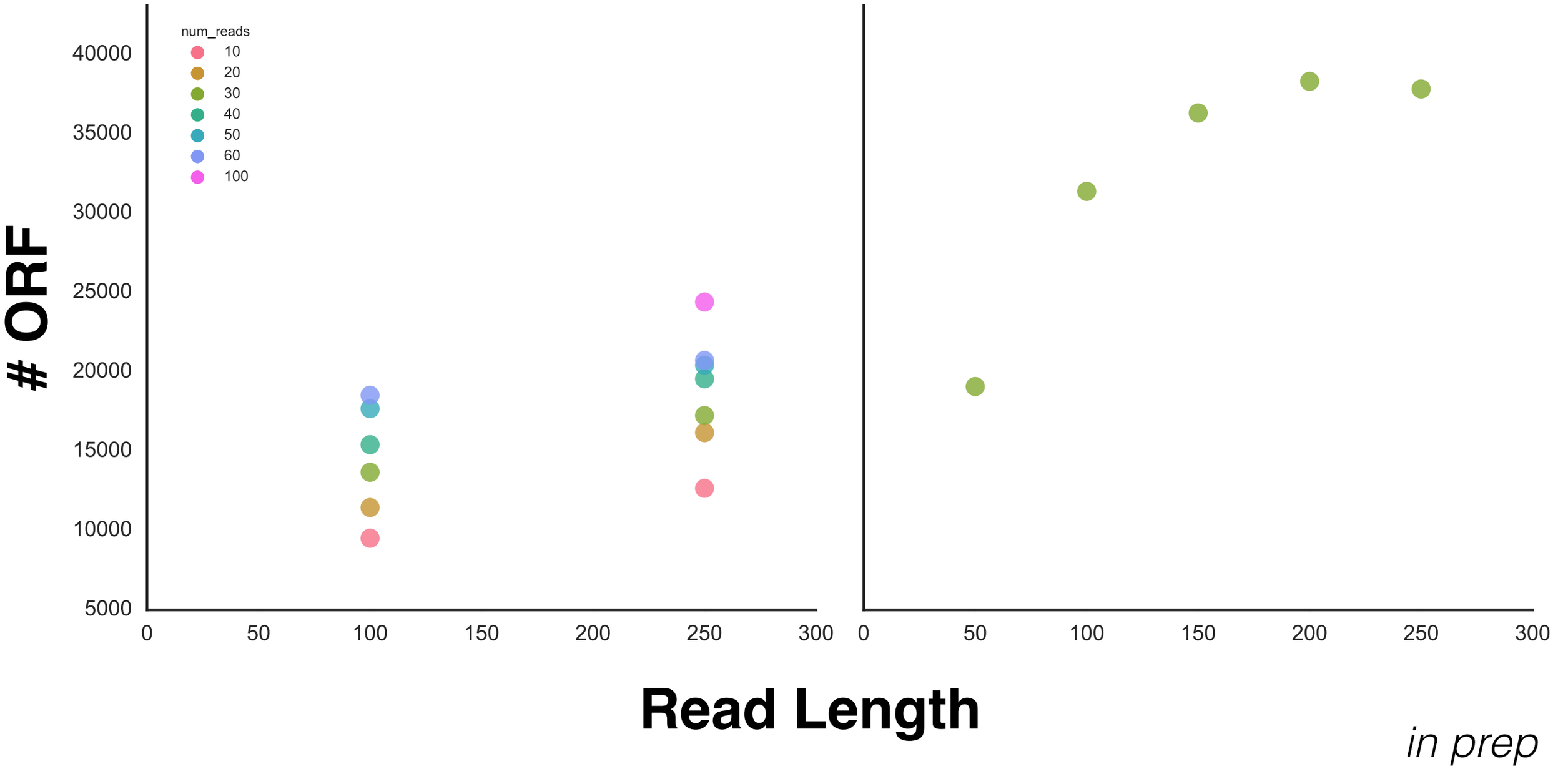
in prep

Read length to sequence?



in prep

Read length to sequence?



in prep

why use MMT?

- Ease of use: 20-30 programs with one command
- Ease of substitution (once familiar with python)
- Good starting parameters
- Good Logging

```
'github.com/bluegenes/MakeMyTranscriptome.
```