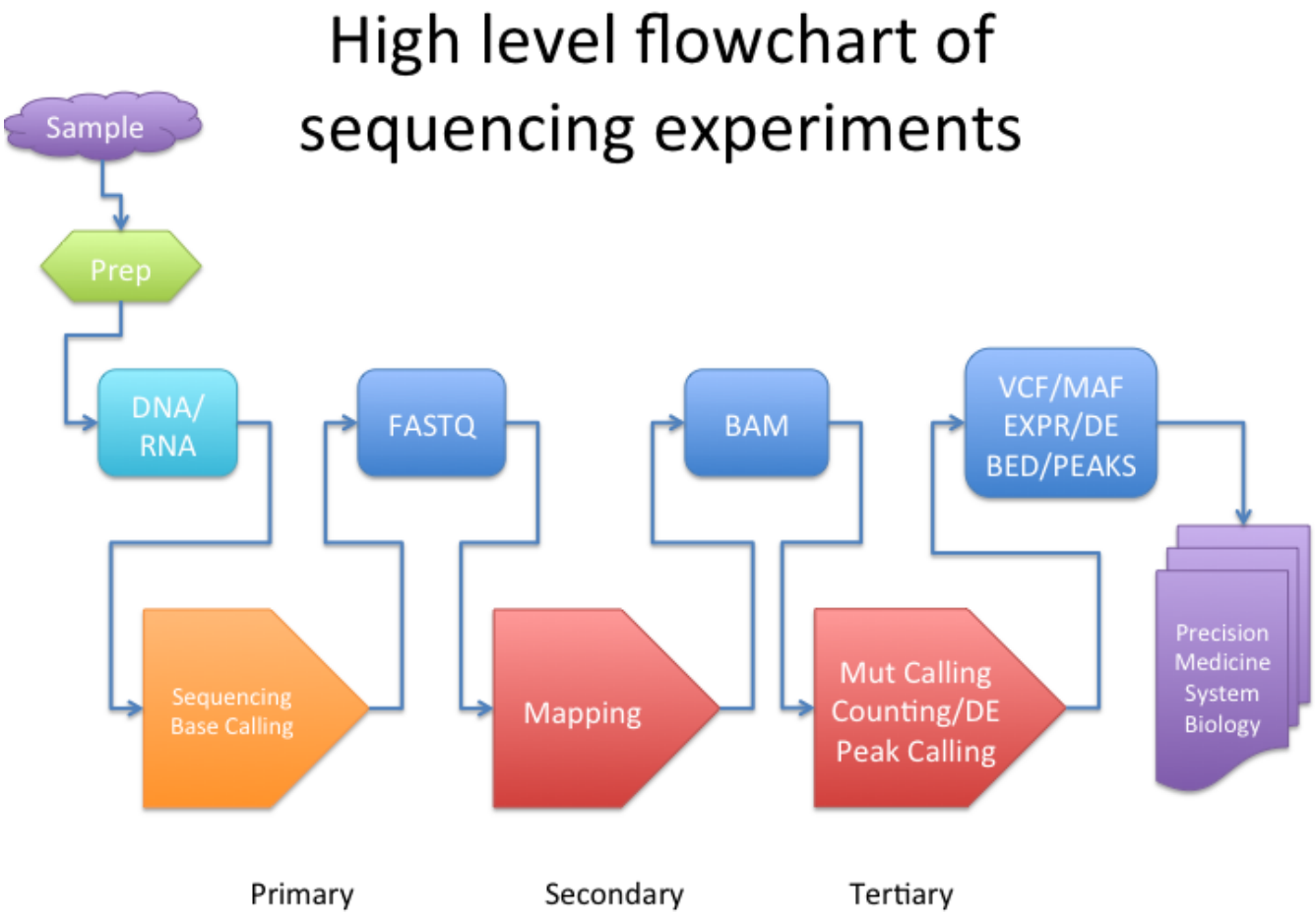
Module 1: Technology

Overview

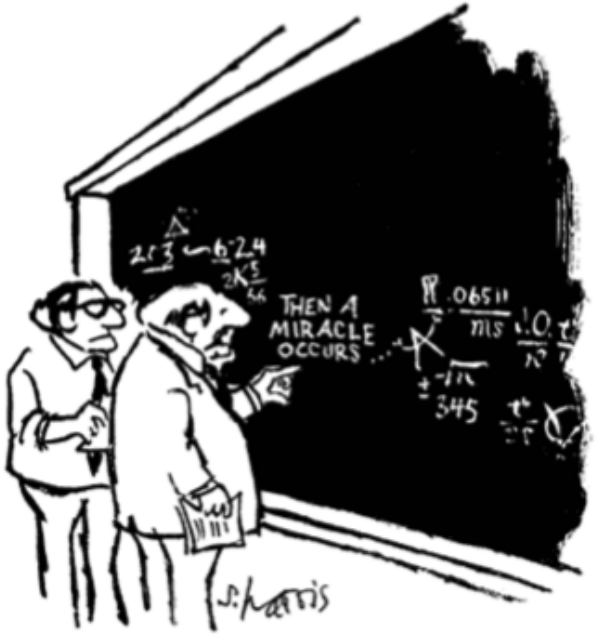

High level flowchart of sequencing experiments
Sample
Prep
DNA/ RNA
FASTQ
BAM
VCF/MAF EXPR/DE BED/PEAKS
Sequencing Base Calling
Mapping
Mut Calling Counting/DE Peak Calling
Precision Medicine System Biology
Primary
Secondary
Tertiary

Computer Stuff

- All the stuff in the middle involves heavy use of computers
  - No way to avoid it
- But to many that stuff in the middle is impenetrable
  - And often computer/math/physics types are not all that helpful

Obtuse and Obfuscated

Complicated Computer Stuff



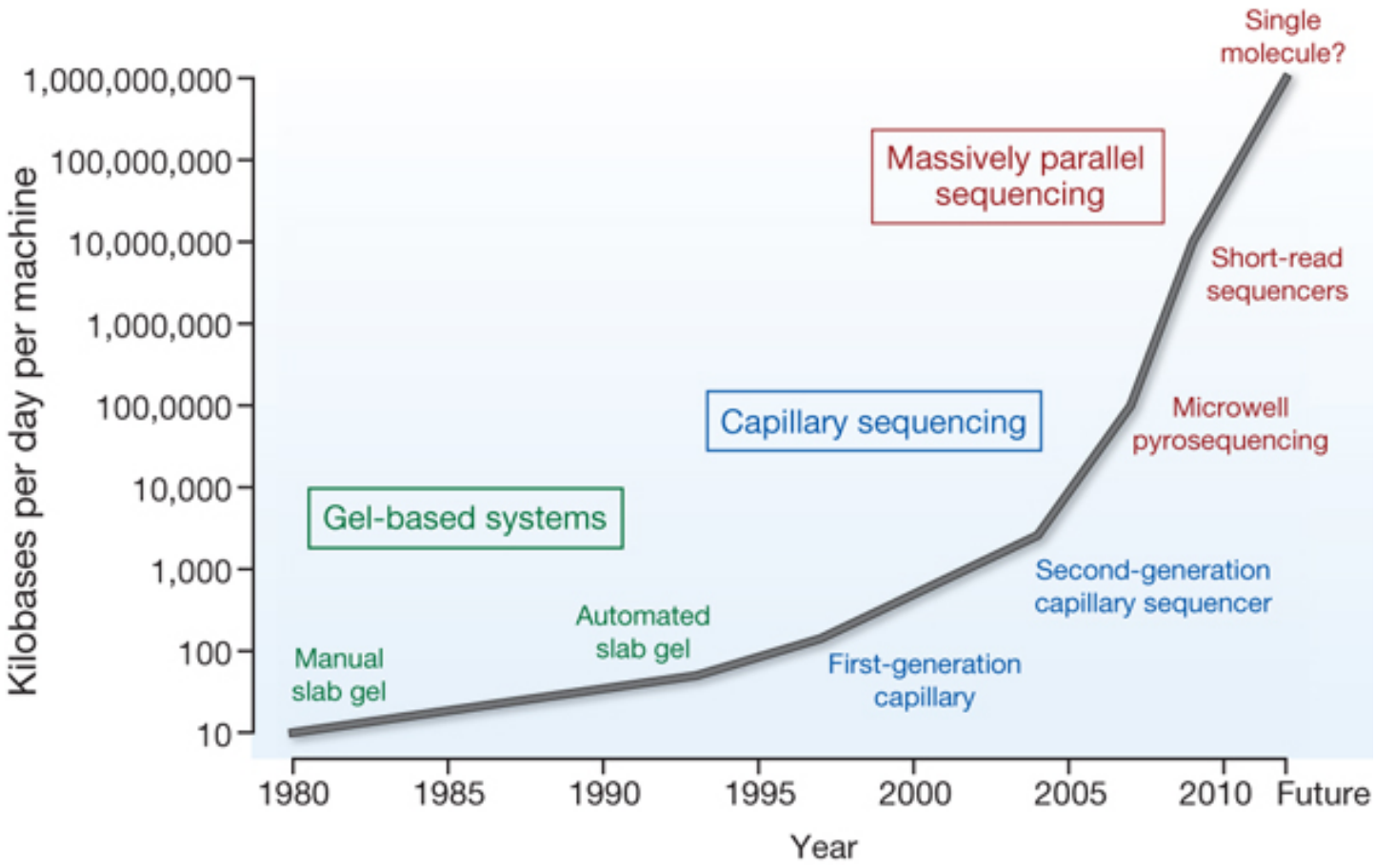
"I think you should be more explicit here in step two."

Here to help

- Or at least try

Introduction to sequencing technologies

Manditory Growth Slide

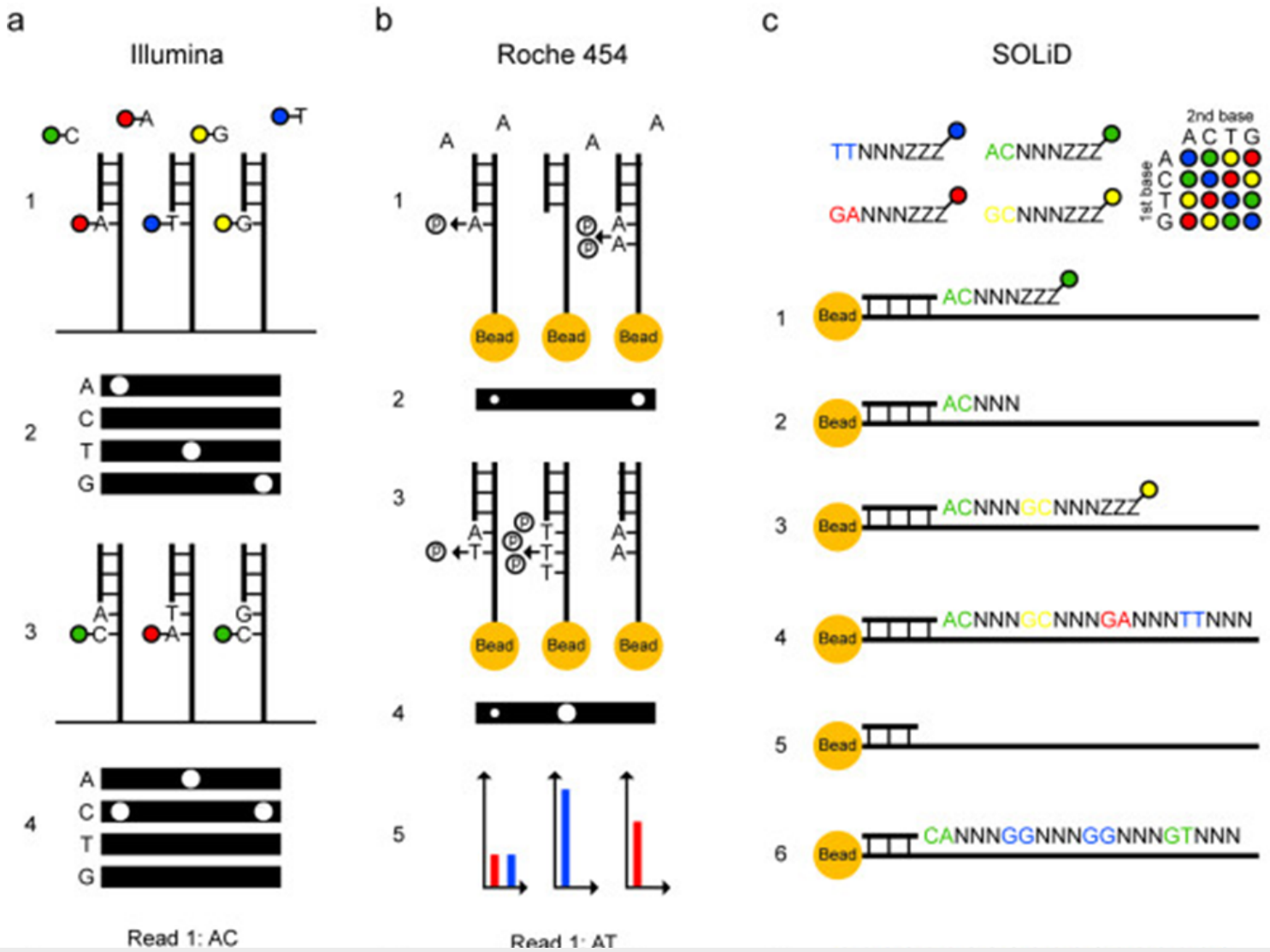


MR Stratton *et al. Nature* **458**, 719-724 (2009)

Multiple technologies

- Illumina
- SOLiD
- 454 (successor Ion Torrent)
- PacBio

Technology comparison



Illumina: Sequencing by synthesis

Cyclic reversible termination

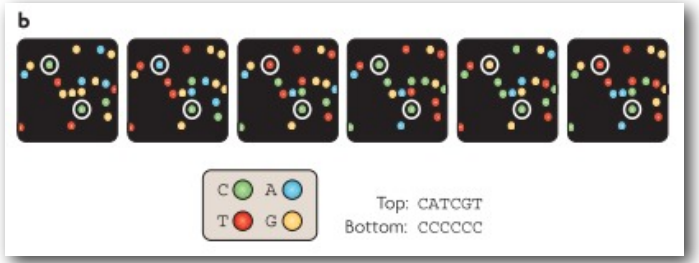
DNA synthesis is terminated after adding single nucleotide

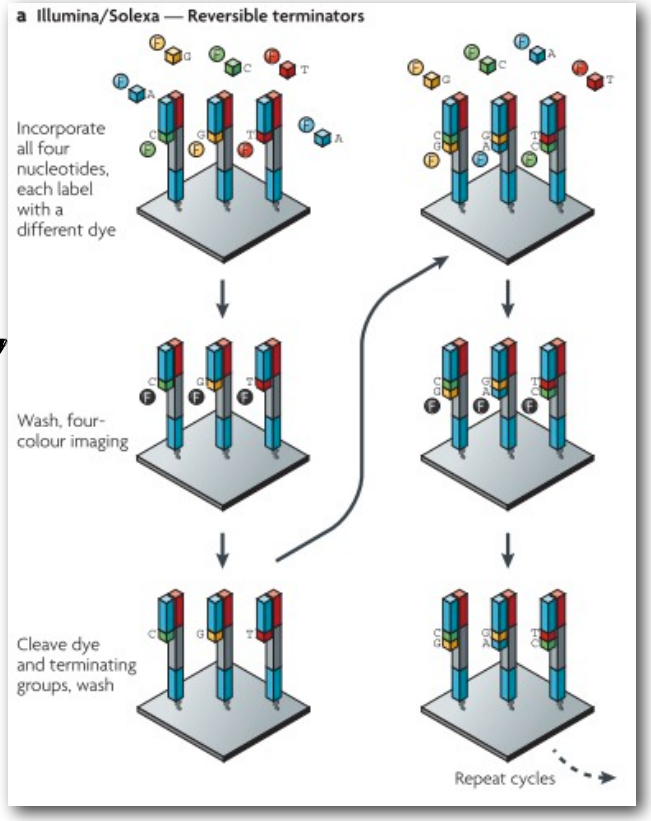
start/stop/start/stop/start/stop/...

Illumina: 4-colour

sequencing result

sequencing steps





Metzker et al, 2010

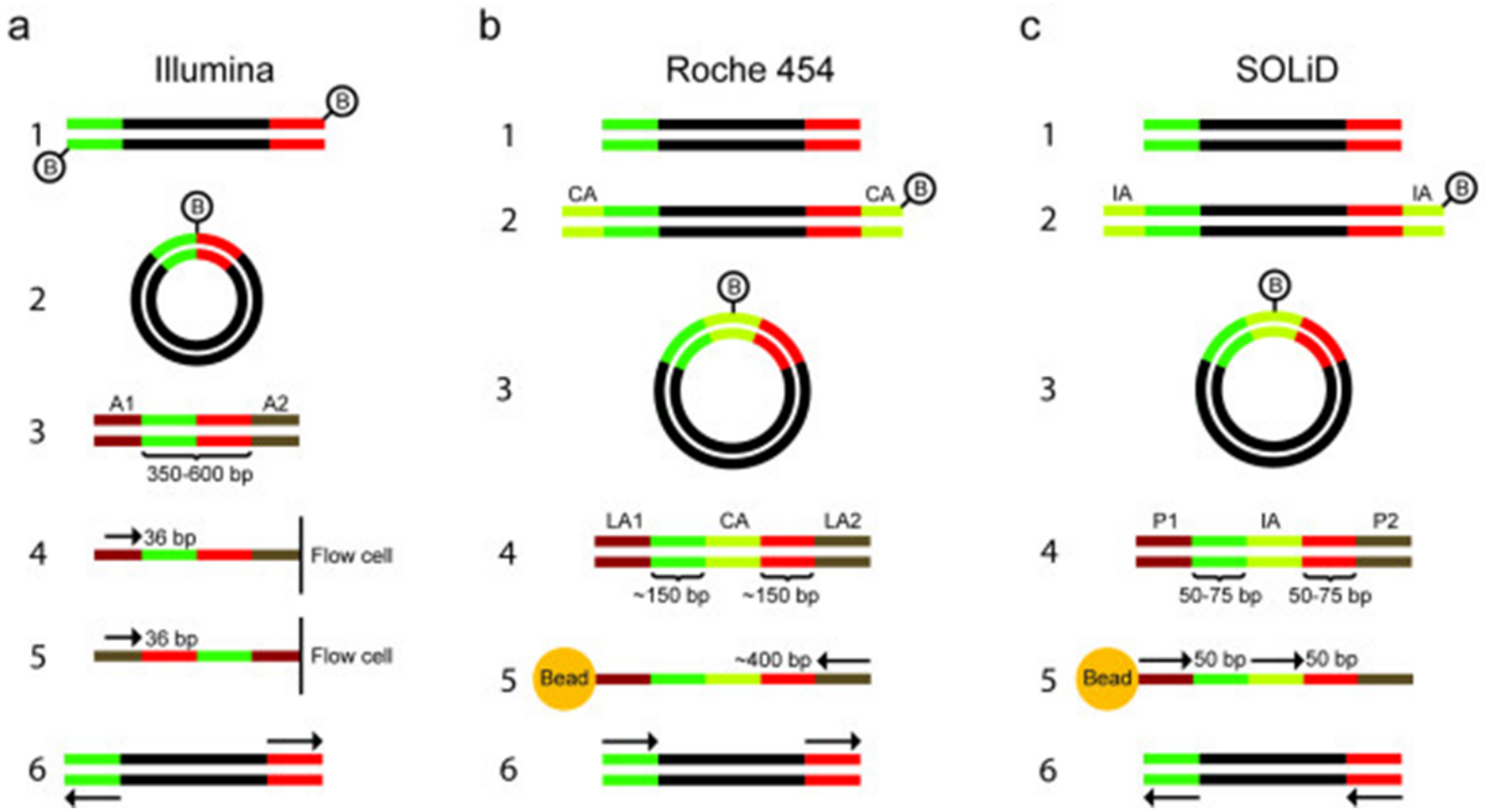
Accuracy

- Sanger > SOLiD > Illumina >> 454/IonTorrent >> PacBio
  - 454/IonTorrent problem with homopolymers
  - However with the exception of Sanger read length goes up as you move to the right. Less accuracy but longer reads
  - And Sanger has problem with low frequency events

pyrosequencing homopolymer problem

- Affects 454 and IonTorrent
- Because it reads multiple runs of the same base in one cycle there is a signal to noise issue;
  - Need to discriminate $(N-1)/N$
    - threshold is like $(1/N)$
  - This gets very hard as N gets large
  - Practical limit 5-8 mers
  - But when 2mers are issues

Paired End Sequencing



Paired End Sequencing, II

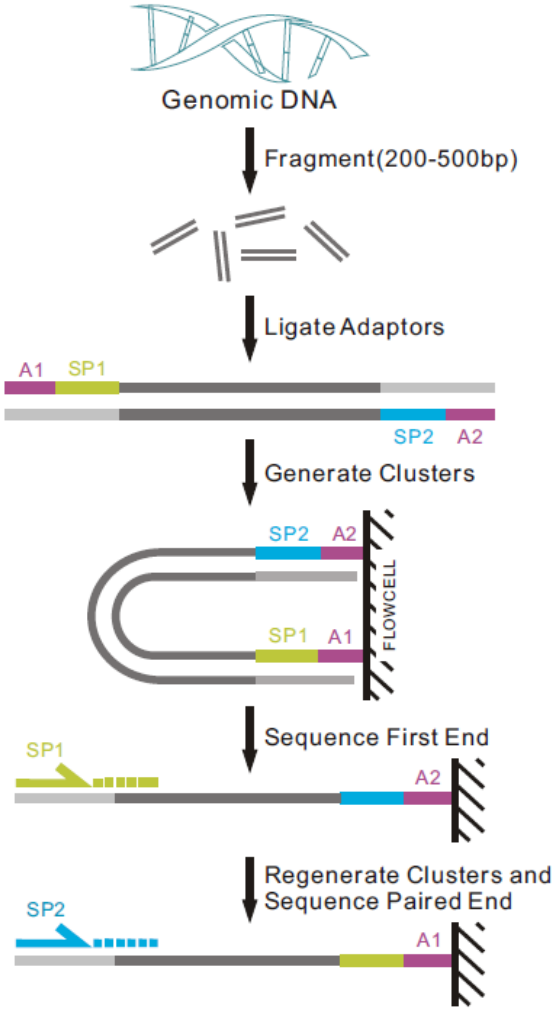


Figure 1-2-1 Pipeline of paired-end sequencing (www.illumina.com)

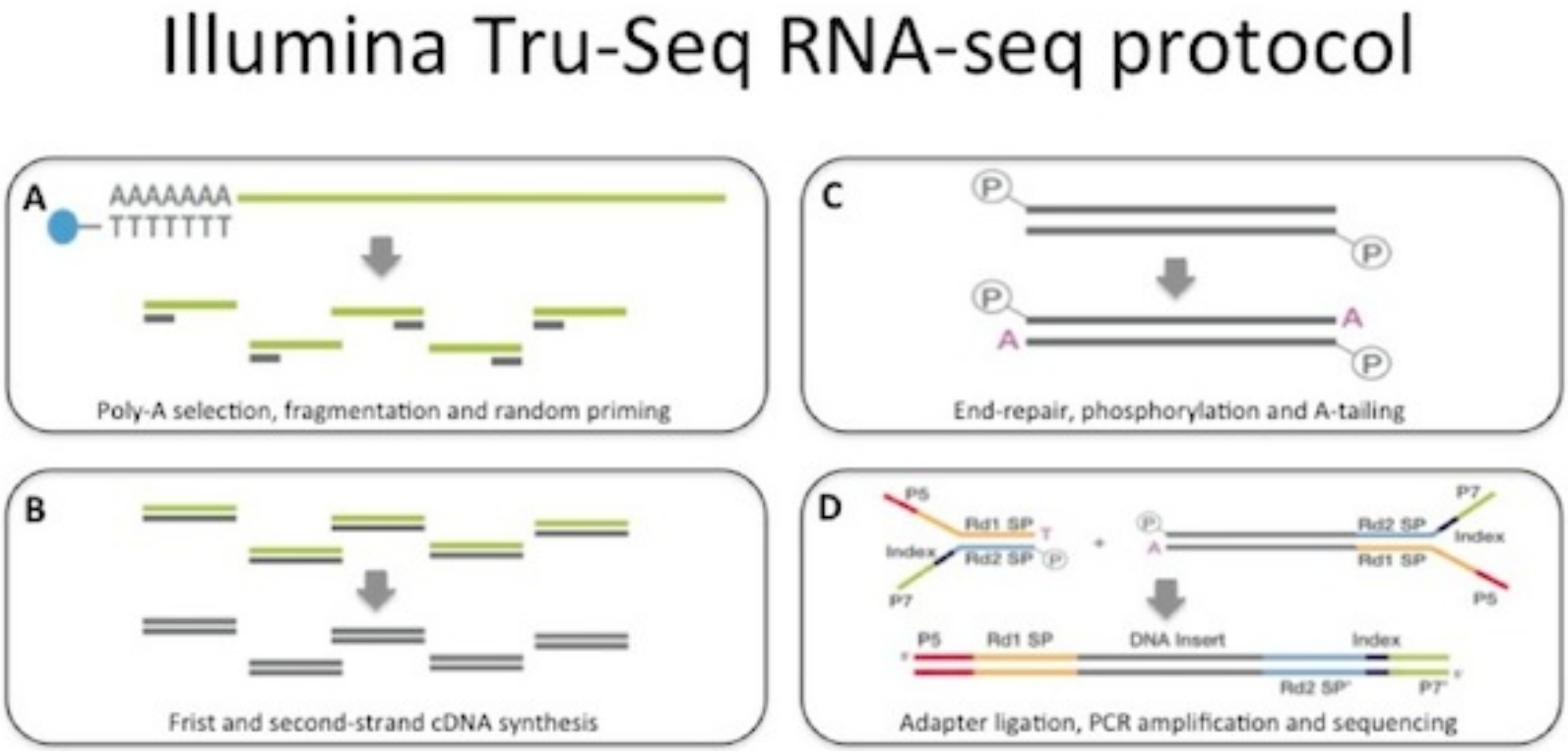
Applications of NGS

- Resequencing: mutation/variant detection
  - Targeted assay (Whole exome, IMPACT)
    - Go over in great deatil tomorrow
- RNAseq
- ChIPseq
- others not discussed
  - Whole Genome Sequencing (WGS)
  - BiSulfite
  - XXXseq

RNAseq library types (for ChIPSeq guys)

- From a bioinformatics view you need to know (you really do)
  - Poly-A unstranded (Illumina TruSeq Poly-A Selection)
    - Unstranded
  - SMARTer Amplification
    - Strand Forward, FIRST_READ_TRANSCRIPTION_STRAND
  - KAPA mRNA Stranded
    - Strand Reverse, SECOND_READ_TRANSCRIPTION_STRAND
  - Ribo-minus (Illumina TruSeq RiboDeplete)
    - Strand Reverse, SECOND_READ_TRANSCRIPTION_STRAND

Illumina True Seq RNAseq



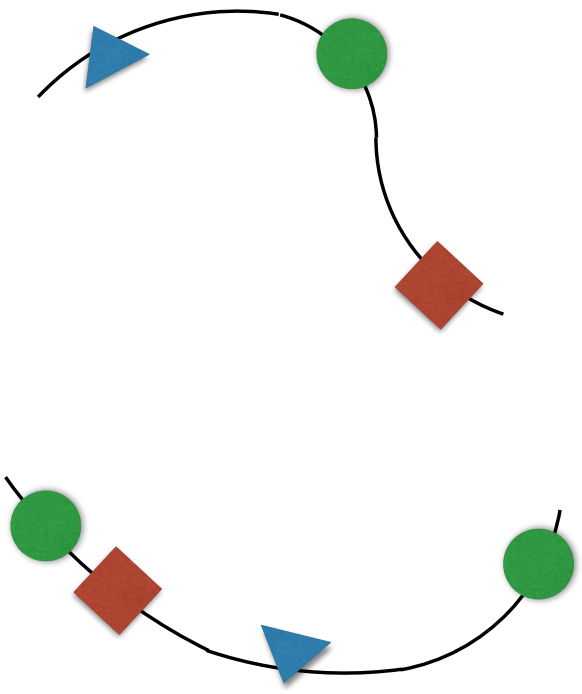
Library prep begins from 100ng-1ug of Total RNA which is poly-A selected (A) with magnetic beads. Double-stranded cDNA (B) is phosphorylated and A-tailed (C) ready for adapter ligation. The library is PCR amplified (D) ready for clustering and sequencing.

Two different ChIP libraries

- From a bioformatics view you know
  - Focal Binding ChIP: le protein binding is strongly localized
    - Transcription Factors
  - Diffuse Binding ChIP: binding is weak-localized
    - Histone (chromotin) or Methyl binding factors
- MACS calls there model and non-model cases

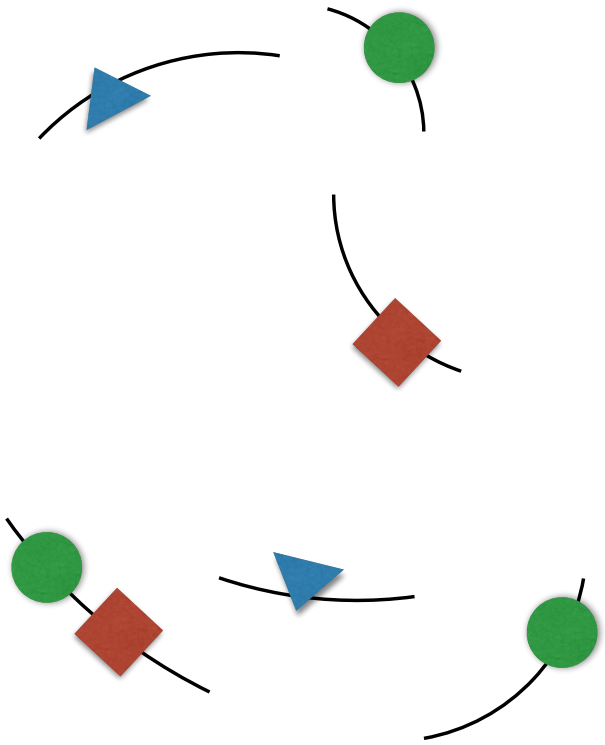
ChIPseq library prep (for RNAseq guys)

- Cross-link DNA and proteins
- Isolate DNA & fragmentation
- Chromatin Immunoprecipitation
- Reverse cross-links and purify DNA
- Add adapters & sequence



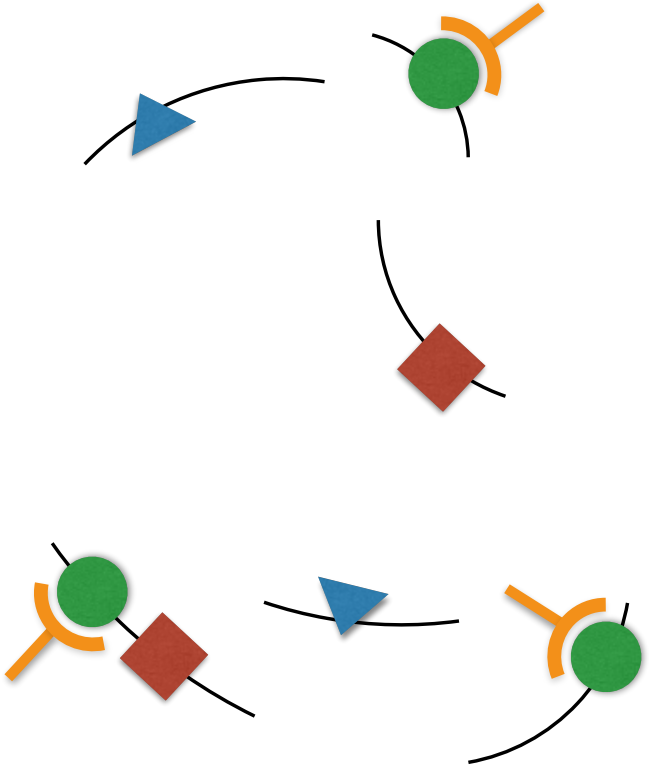
ChIPseq library prep

- Cross-link DNA and proteins
- Isolate DNA & fragmentation
- Chromatin Immunoprecipitation
- Reverse cross-links and purify DNA
- Add adapters & sequence



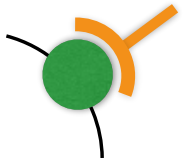
ChIPseq library prep

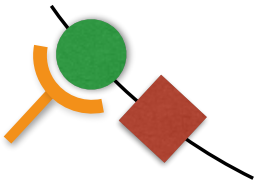
- Cross-link DNA and proteins
- Isolate DNA & fragmentation
- Chromatin Immunoprecipitation
- Reverse cross-links and purify DNA
- Add adapters & sequence

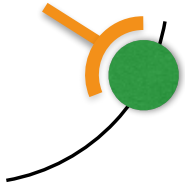


ChIPseq library prep

- Cross-link DNA and proteins
- Isolate DNA & fragmentation
- Chromatin Immunoprecipitation
- Reverse cross-links and purify DNA
- Add adapters & sequence

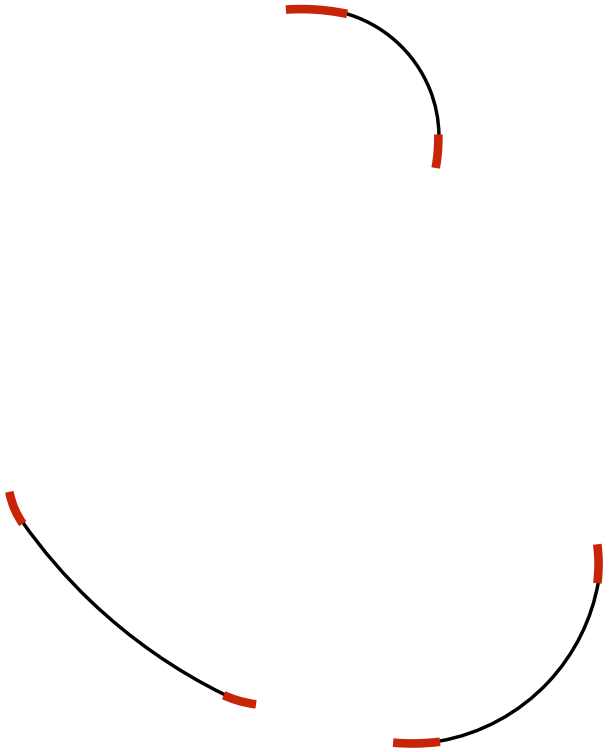






ChIPseq library prep

- Cross-link DNA and proteins
- Isolate DNA & fragmentation
- Chromatin Immunoprecipitation
- Reverse cross-links and purify DNA
- Add adapters & sequence



Next-gen Sequencing data file formats:

1. Sequence
   - FASTA/FASTQ
2. Alignment/Mapping
   - BAM
3. Variants
   - VCF/MAF

Sequence data:

Several format in use

- FASTA/FASTQ
- SRA (Short read archive)
- ABI (sanger)
- Lots of proprietary formats
- DOC (word)
- uBAM (unmapped BAM)

Main format: FASTA/FASTQ

Original format: FASTA

- For both xNA (nucleotides) and AA (proteins)
- Basic structure:

```
>gi|31563518|ref|NP_852610.1| microtubule-associated
MKMRFFSSPCGKAAVDPADRCKEVQQIRDQHPSKIPVIIERYKGEKQ
LPVLDKTKFLVPDHVNMSELVKIIRRRLQLNPTQAFFLLVNQHSMVS
VSTPIADIYEQEKDEDGFLYMVYASQETFGFIRENE
```

FASTA, cont.

- Can encode multiple sequences

```
>SEQUENCE_1
MTEITAAMVKELRESTGAGMMDCKNALSETNGDFDKAVQLLREKGL
LVSVKVSDDFTIAAMRPSYLSYEDLDMTFVENEYKALVAELEKENE
IPQFASRKQLSDAILKEAEEKIKEELKAQGKPEKIWDNIIPGKMNS
MGQFYVMDDKKTVEQVIAEKEKEFGGKIKIVEFICFEVGEGLEKKT
>SEQUENCE_2
SATVSEINSETDFVAKNDQFIALTKDTTAHIQSNSLQSVEELHSST
ATIGENLVVRRFATLKAGANGVVNGYIHTNGRVGVVIAAACDSAEV
```

Extension to store quality of reads: FASTQ

- Change delimiter and add an additional line of quality information

```
@SEQ_ID
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTT
+
!''*((((***+))%%%++)(%%%%).1***-+*''))**55CCF>>
```

- the 4th line encodes the Quality value (Q) for each base

Q value / PHRED scale

- The q value is defined to be

$$Q = -10 \log_{10}(P_{err})$$

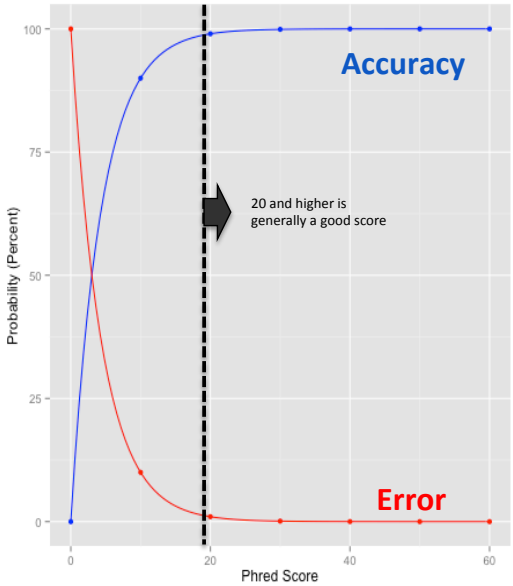
where P_{err} is the probability the base is *incorrect*

Q	Perr	Nerr
10	0.1	1 in 10
20	0.01	1 in 100
30	0.001	1 in 1,000
40	0.0001	1 in 10,000

Q value graph

Phred scaling makes it easier to handle probability scores

- *Phred value* = −10 * *log*10(ε)
- Examples:
  - 90% confidence (10% error rate) = Q10
  - 99% confidence (1% error rate) = Q20
  - 99.9% confidence (.1% error rate) = Q30
- SAM encoding adds 33 to the value
  (because ASCII 33 is the first visible character)



Q encoding

- The Q value has over time been encoded in different ways

```
 SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS.....................................................
 ..........................XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX......................
 ..............................IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII......................
 ................................JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ......................
 LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL.....................................................
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
 |                          |    |        |                              |                     |
33                         59   64       73                            104                   126
 0........................26...31.......40
                          -5....0........9.............................40
                                0........9.............................40
                                   3.....9.............................40
 0.2......................26...31........41

S - Sanger        Phred+33,  raw reads typically (0, 40)
X - Solexa        Solexa+64, raw reads typically (-5, 40)
I - Illumina 1.3+ Phred+64,  raw reads typically (0, 40)
J - Illumina 1.5+ Phred+64,  raw reads typically (3, 40)
    with 0=unused, 1=unused, 2=Read Segment Quality Control Indicator (bold)
    (Note: See discussion above).
L - Illumina 1.8+ Phred+33,  raw reads typically (0, 41)
```

Q encoding

Illumina 1.8+ $Phred + 33$

$ASCII \rightarrow Q$

$$Q \leftarrow ord(C) - 33$$

where `ord` is the ascii value for a character

$Q \rightarrow ASCII$

$$C \leftarrow chr(Q + 33)$$

where `chr` converts an integer to ascii

Quality Control (Manipulating FASTA files)

- FastQC toolkit: (http://www.bioinformatics.babraham.ac.uk/projects/fastqc/)
- Show Samples

Unix Scripting Crash Course

History

- First shell, Ken Thompson, *1971* (44yr)
- First (?) UNIX shell, Bourne Shell [SH], Stephan Bourne, 1977
- C-Shell [TCSH], Bill Joy, 1978
- People (especially scientist) have been using some from of a shell to talk to computers for longer that most of people in this room were alive.
- Probably will still be using it after we are gone.

Google has (nearly) all the answers

- You can pretty much ask Google simple computing questions as get answer almost all the time
- Should really get in the habit of using it

Where to start

- Assume you have basic knowledge of:
  - Basic commands
  - Files and directories

Important intermediate commands

`history`

- list and rerun commands
- `!!` usually replaced by up-arrow (^p)
- history editting replaced with cut-and-paste

`man`

- make sure to go over `man`
  - `man -k` == `apropos`

Important intermediate commands

`fgrep`, `egrep`, `grep`

- `fgrep` == fast grep (`grep -F`)
- `egrep` == extended grep (`grep -E`)
  - regular expression crash course
  - like wild cards but different syntax

From man page:

> *Direct invocation as either `egrep` or `fgrep` is deprecated, but is provided to allow historical applications that rely on them to run unmodified.*

i.e., for old people

Shell (Bash) Scripting

- It is possible (and often very usefull) to write programs (usually called scripts) using the `Bash` shell.
  - Advantage: Syntax very similar to what you do on the command line
  - Disadvantage: Syntax is often confusing and sometimes bizzare.
- Can be a very powerfull tool for writing *pipelines*:
  - Pipeline := Typically a series of commands (programs) run in sequence to transform one file/data type to another.
- **IMPORTANT** as was said yesterday; if you want to have *reproducible research*:
  - Everything needs to be a *script*
  - Does not have to be a bash script but some script

Simple variant pipeline in bash pseudo-code

```bash
#!/bin/bash
# Map a FASTQ file to the genome post-process BAM file and then call variants

INPUT_FASTQ=$1
GENOME=$2
ODIR=$3

TDIR=$ODIR/tmp
mkdir -p $TDIR

bwa mem $GENOME $INPUT_FASTQ >$TDIR/bwa.sam
picard SortSam I=$TDIR/bwa.sam O=$TDIR/sort.bam SO=coordinate
picard MarkDuplicates I=$TDIR/sort.bam O=$TDIR/md.bam M=$ODIR/markDups.txt
mutect $TDIR/md.bam $ODIR/mutect.vcf
```

Run this pipeline with

- If those lines were saved in a file: `variantPipeline.sh` in your current directory you could run it with:

```
$ ./variantPipeline.sh sample1.fastq.gz /genome/human_b37.fa /res/sample1
```

- And you could create another script to process all the FASTQ files in the current directory:

```
#!/bin/bash
GENOME=/genome/human_b37.fa
for fastq in *fastq.gz; do
    ./variantPipeline.sh sample1.fastq.gz $GENOME /res/${fastq/.fastq.gz/}
done
```

Alternatives

- If this code looks horrifing or ugly or . . . , there are many, many alternatives for writing pipelines
- Can use nearly any programming language that has `system` system call
  - PERL: `cmd`
  - Python: lots of choices: subprocess lib best?
  - C/C++: system()
  - R: system
- and lots of modules/libraries/packages that will wrap `system` more nicely
- Other shells besides bash:
  - tcsh
  - zsh
  - korn

Alternatives: MAKE-like (implicit)

- make/Makefiles and derivatives
  - Some people love this others find it evil
  - Scons (python)
  - Rake (ruby)
  - SnakeMake: (python) somewhat popular in bioinformatics
  - Nextflow
  - BigDataScript

Alternatives: Workflow systems

- Tons of these; some popular bioinformatics ones
  - bpipe (java/groovy)
  - Ruffus (python)
  - Galaxy (www gui)
  - Taverna (gui)
  - Common Workflow Language (CWL):
    - Arvados

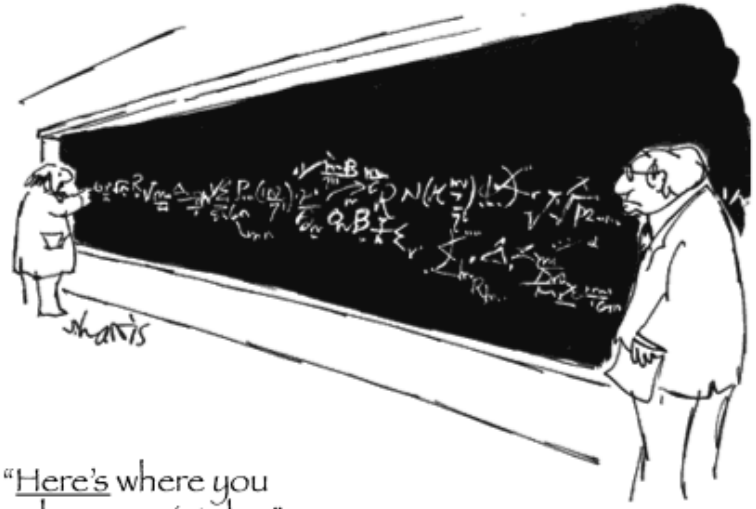
Bottom line:

- `Bash` is probably the most awefull in terms of syntax, modern programming ideas (lack of)
- However it is the most light-weight (install nearly everywhere and ready to use), closest to what how we usually work
- Will use `bash` here; strongly encourage you to look at others
- Reference for comparisons and alternative viewpoints:

  J. Leipzig, A review of bioinformatic pipeline frameworks, Briefings in Bioinformatics, 2016, 1-7

Parting thought

Beware



"Here's where you made your mistake."