UNIVERSITÄT LEIPZIG

ECSEQ BIOINFORMATICS

de.NBI/de.STAIR Training Course: A primer for RNA-Seq processing, interpreting and visualization

Konstantin Riege
Steve Hoffmann

de.NBI/de.STAIR Training Course: A primer for RNA-Seq processing, interpreting and visualization

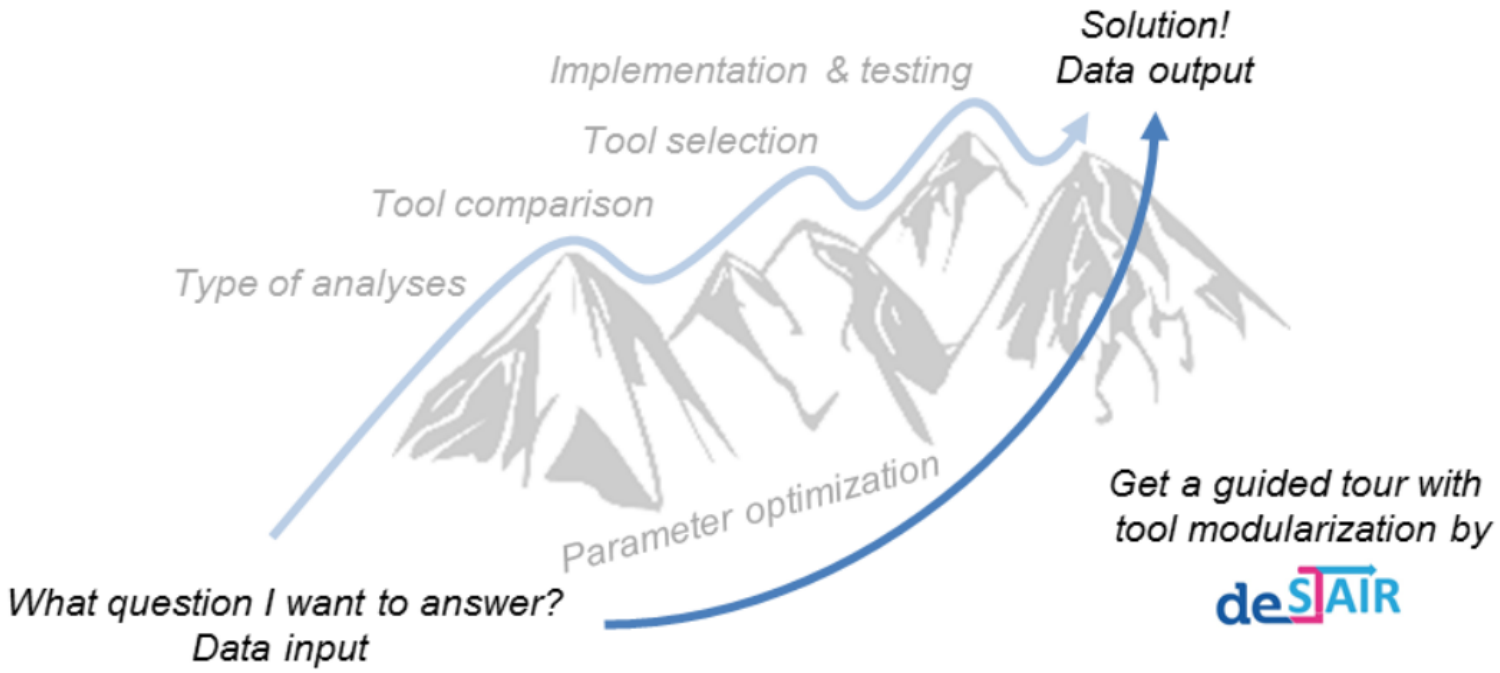
- Introduction into Sequencing techniques
- Introduction into Unix/Linux Command Line
- Introduction into Galaxy
- What is behind Galaxy – Conda, Docker, Galaxy Setup

- RNA-Seq data processing using Galaxy
- Visualization Methods using Galaxy

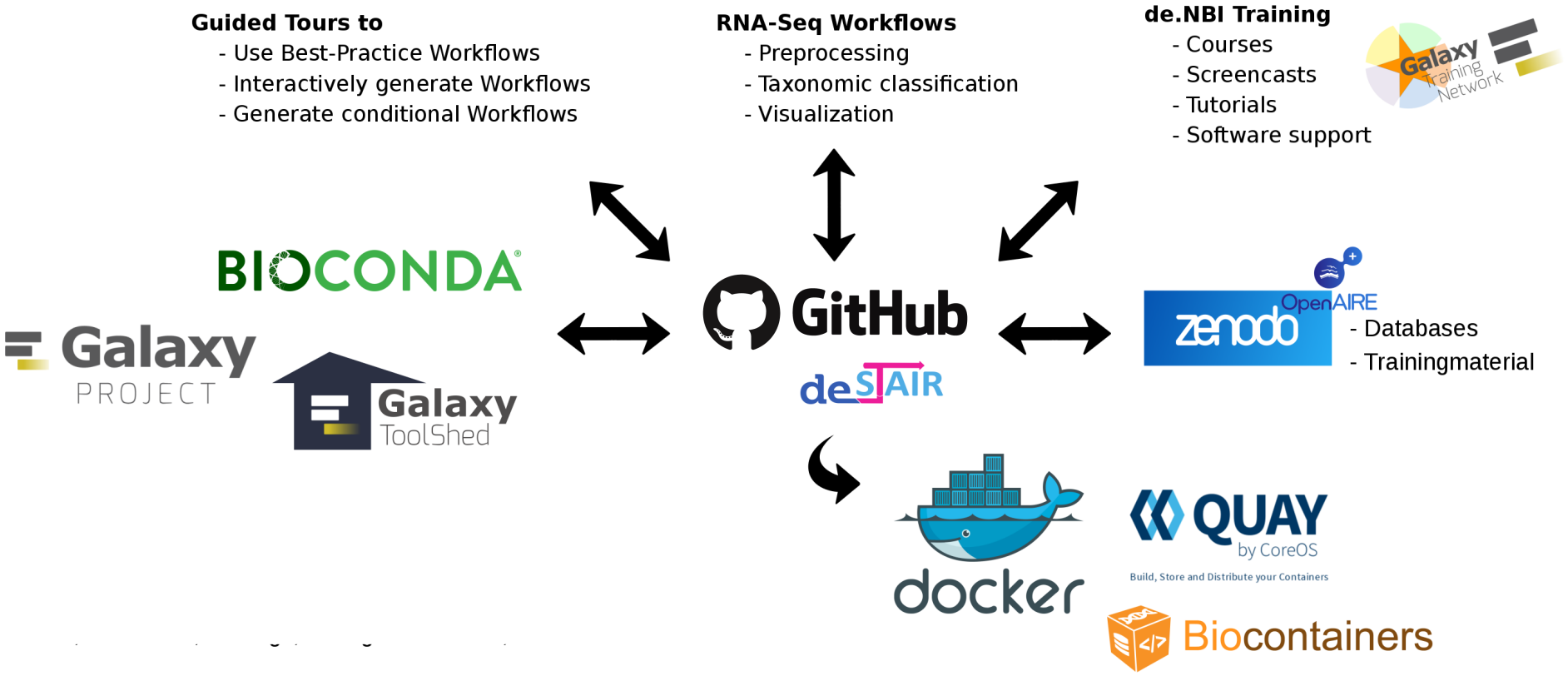
- Metatranscriptomic analysis using Galaxy
- Prediction of ncRNAs

- Poster session
- Lunch, Dinner, Drinks

Bioinformatics Services for Structured Analysis and Integration of RNA-Seq experiments (de.STAIR)

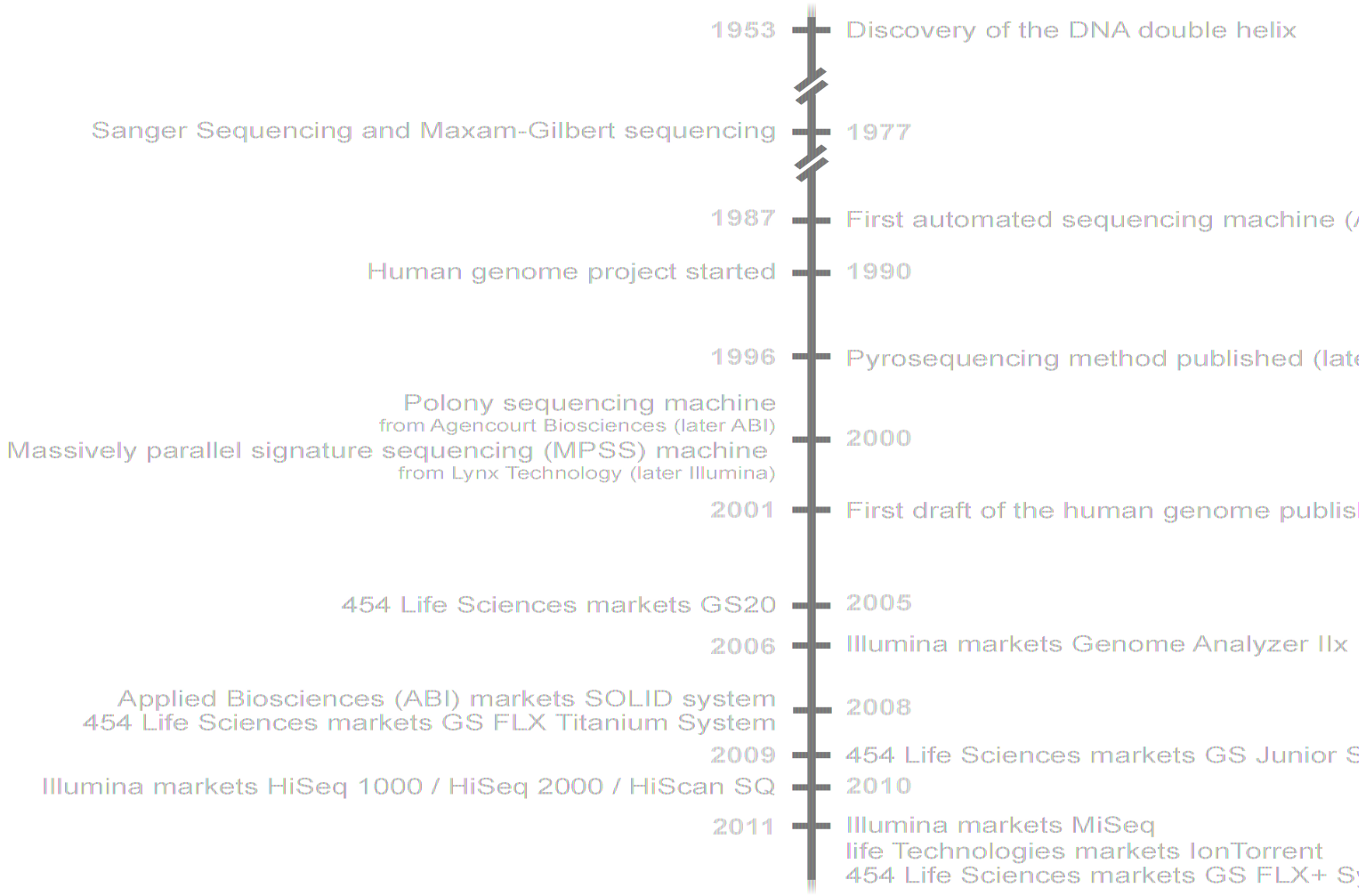


Bioinformatics Services for Structured Analysis and Integration of RNA-Seq experiments (de.STAIR)



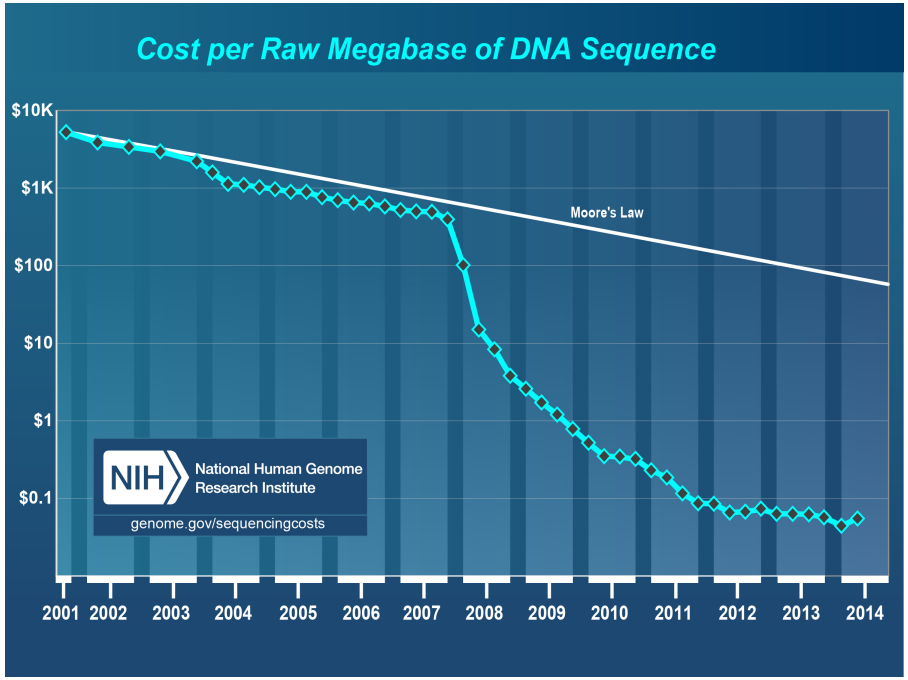
Sequencing Techniques

History of DNA sequencing

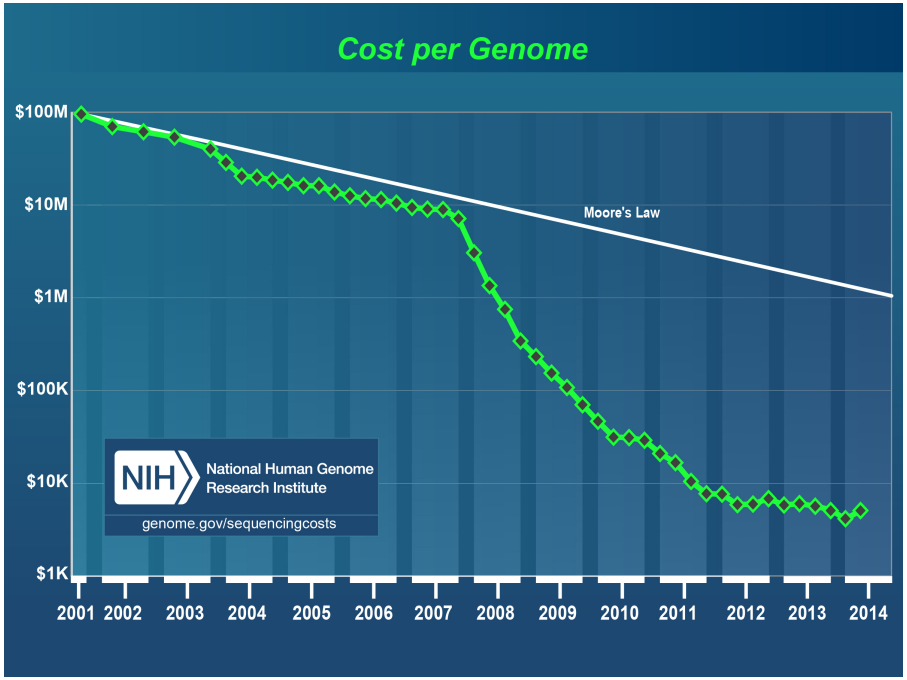


History of DNA sequencing

Cost per Raw Megabase of DNA sequence



Cost per Genome



Illumina Sequencing Platforms

miSeq

NextSeq 500

HiSeq 2500

HiSeq X Ten

ABI 3500 Series Genetic Analyzer

Machine cost: $130k

Run time: 30 min - 3h

Run cost per Mb: ~$1,800

Read Length: up to 850 bp

 # single reads: 8 - 24

Throughput: 138 Kb/day

Error Rate: <0.1%

Illumina HiSeq 2500

Machine cost: $740k

Run time: 6 days

Run cost per Gb: $29

Read Length: 2x125 bp

 # single reads: 4B

Throughput: 1 Tb/run

Error Rate: <0.1%

Illumina HiSeq X Ten

Machine cost: $1M*

Run time: 3 days

Run cost per Gb: $7

Read Length: 2x150 bp

 # single reads: 6B

Throughput: 1.8 Tb/run

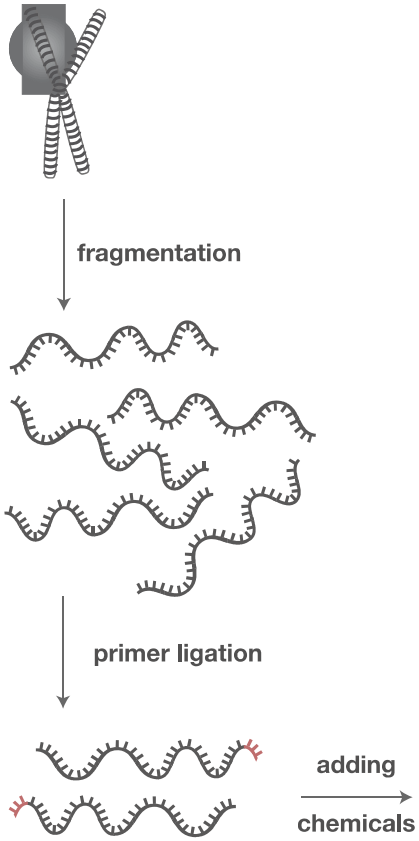
Error Rate: <0.1%

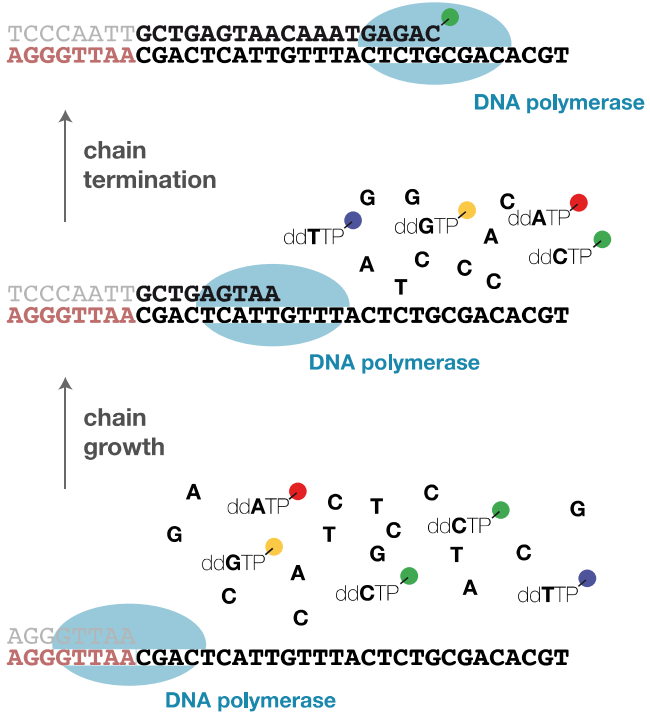
* Minimum purchase of 10 machines

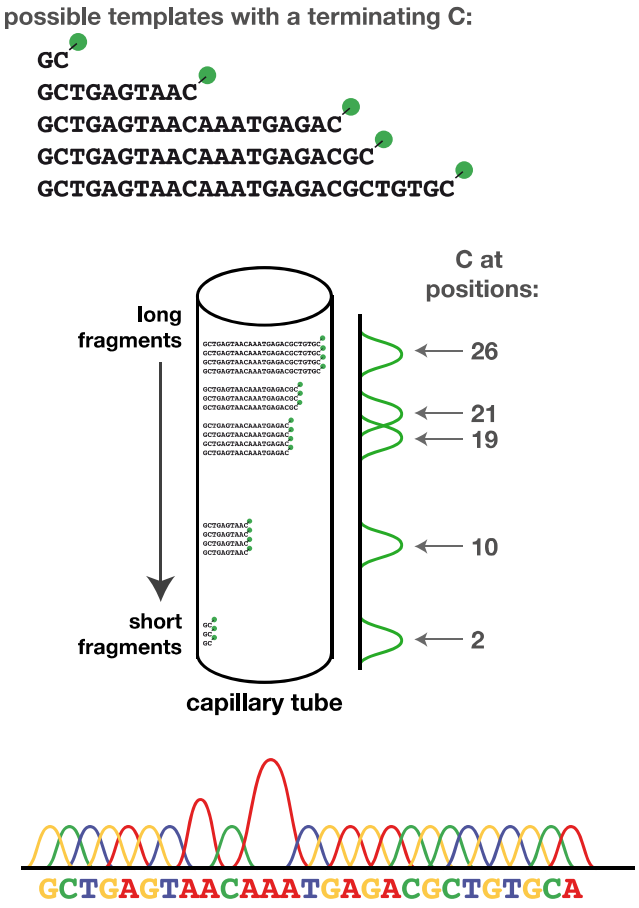
Illumina: Overview

	Run time	Read length	Throughput		Cost	
	(hrs)	(bp)	# reads	bases/run	machine	per Gb
miSeq	65	2 x 300	25M	15Gb	$125k	$93
NextSeq 500	29	2 x 150	400M	129Gb	$250k	$33
HiSeq 2500	144	2 x 125	4B	1Tb	$740k	$29
HiSeq X Ten	72	2 x 150	6B	1.8Tb	$1M	$7

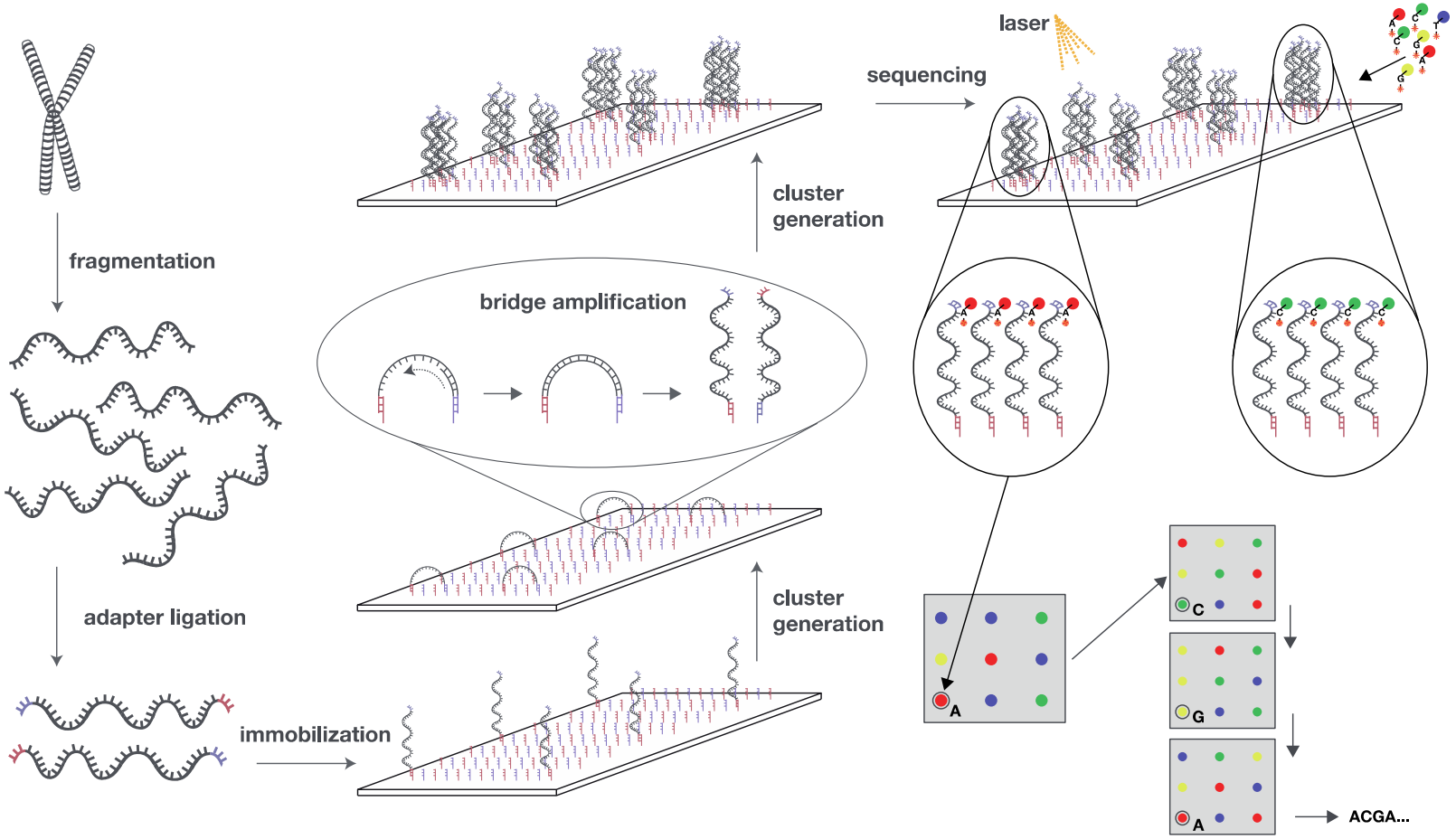
Sequencing workflow



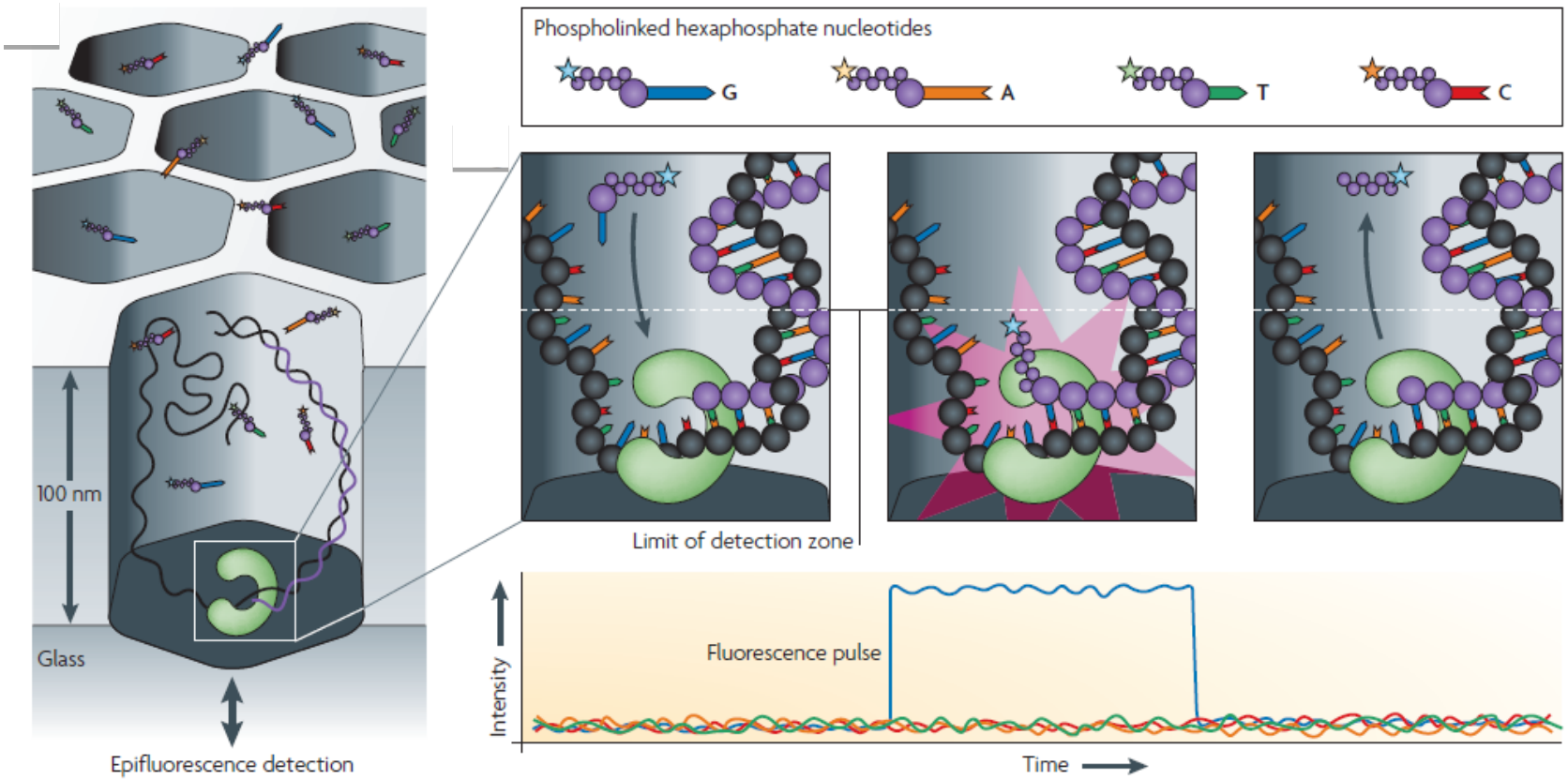




Sequencing workflow



Pacific Biosciences



Pacific Biosciences

Machine cost: $700k

Run time: 180min

Run price: $400

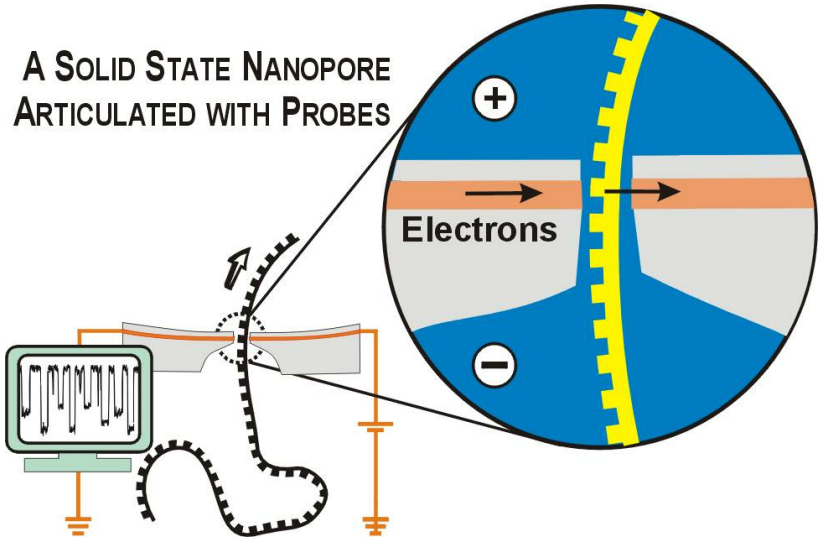
Mean Read Length: 5.5kb

 # single reads: 50k

Throughput: up to 275 MB/run

Error Rate: ~15%

Oxford Nanopore

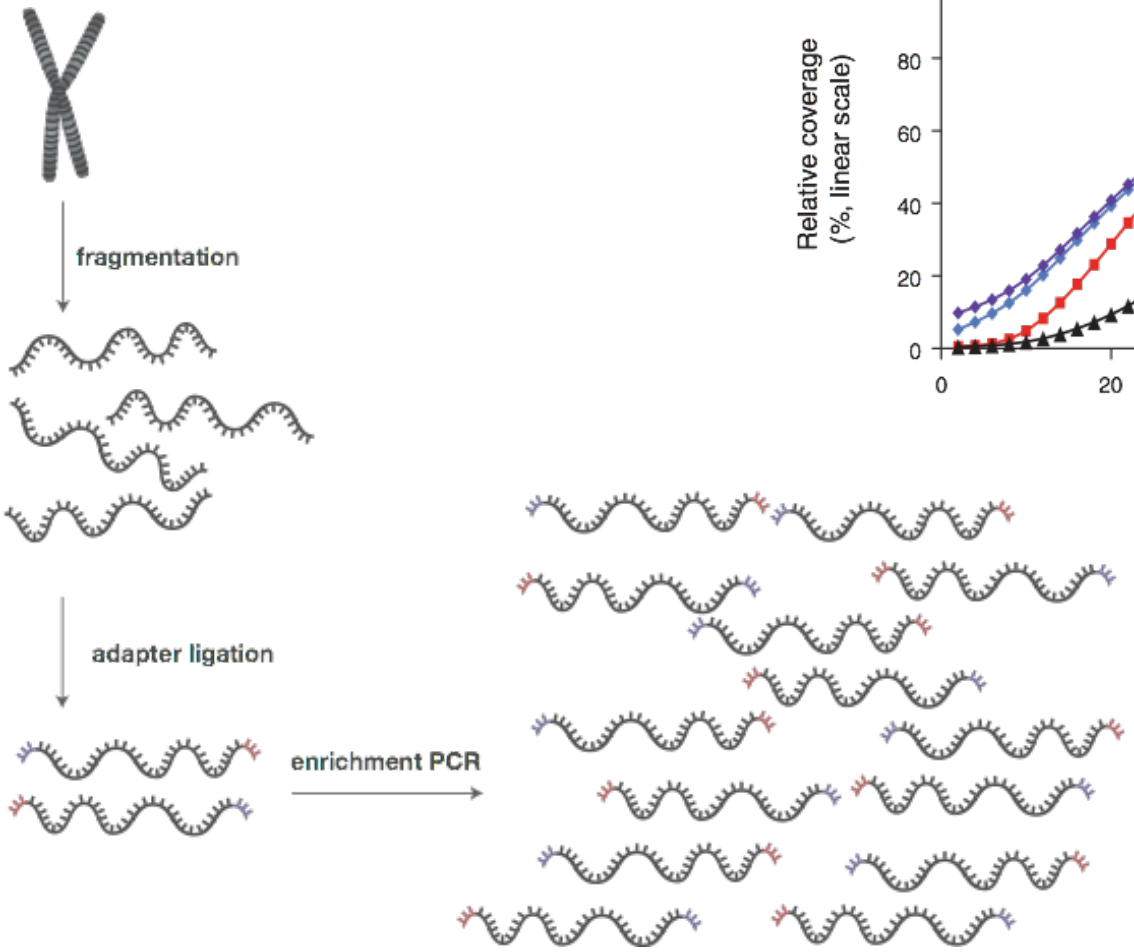


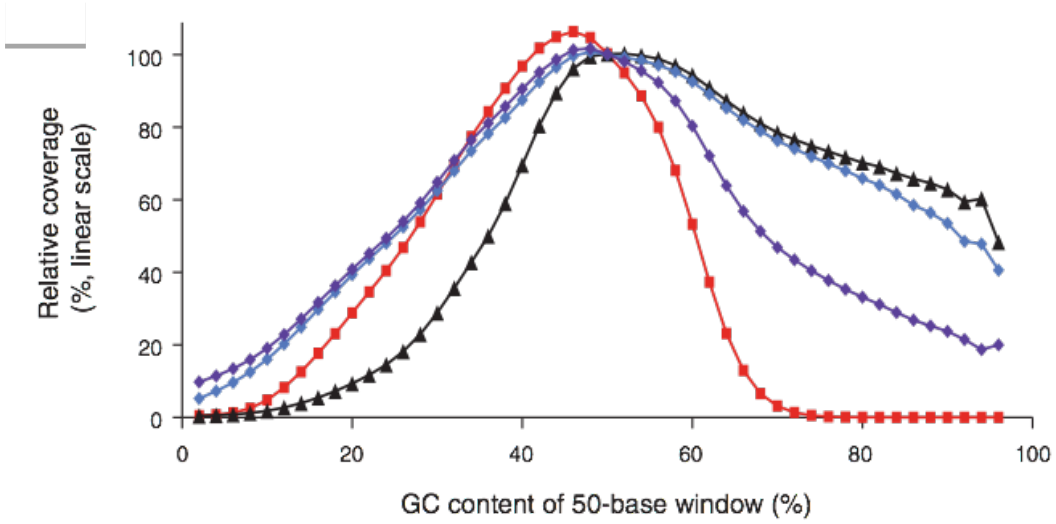
Possible sequencing errors

Possible sequencing errors

Platform	Substitutions	GC bias	AT bias	InDels
Illumina	+++	+	+	+
454	+	+	+	+++
IonTorrent	+++	+	+	+++
PacBio	+++			+++

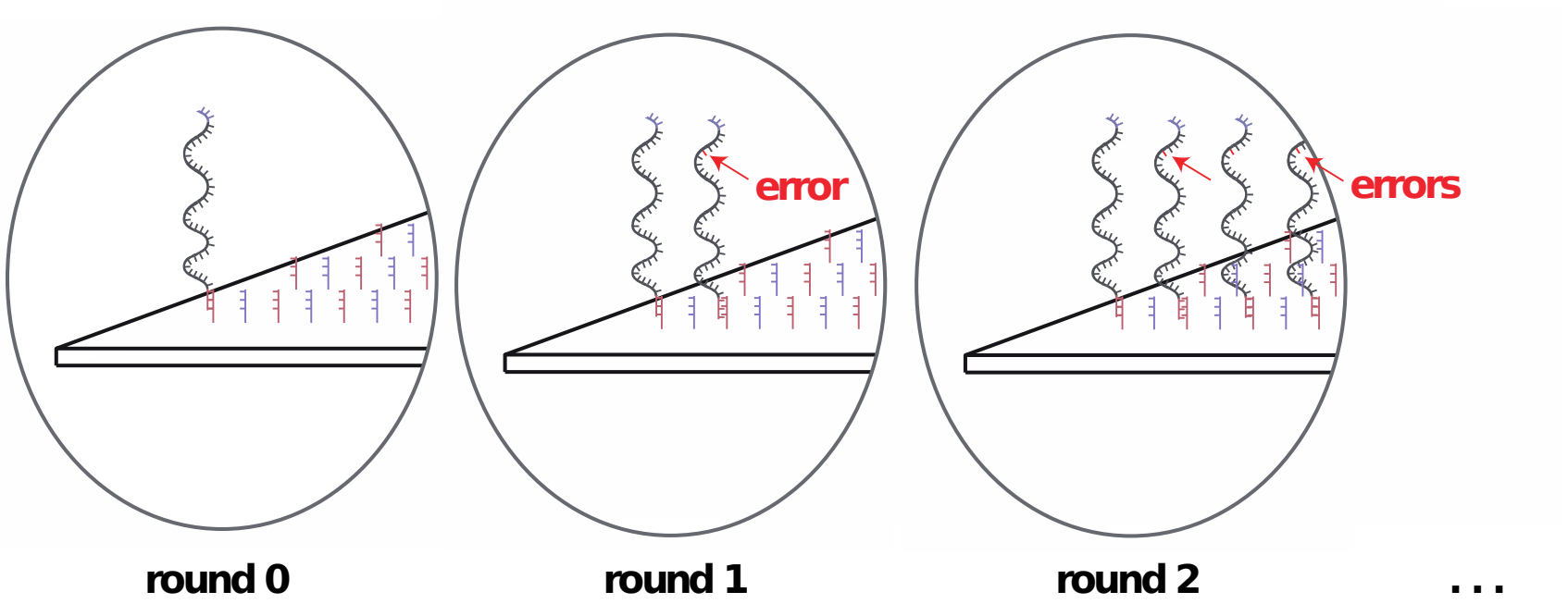
G/C bias





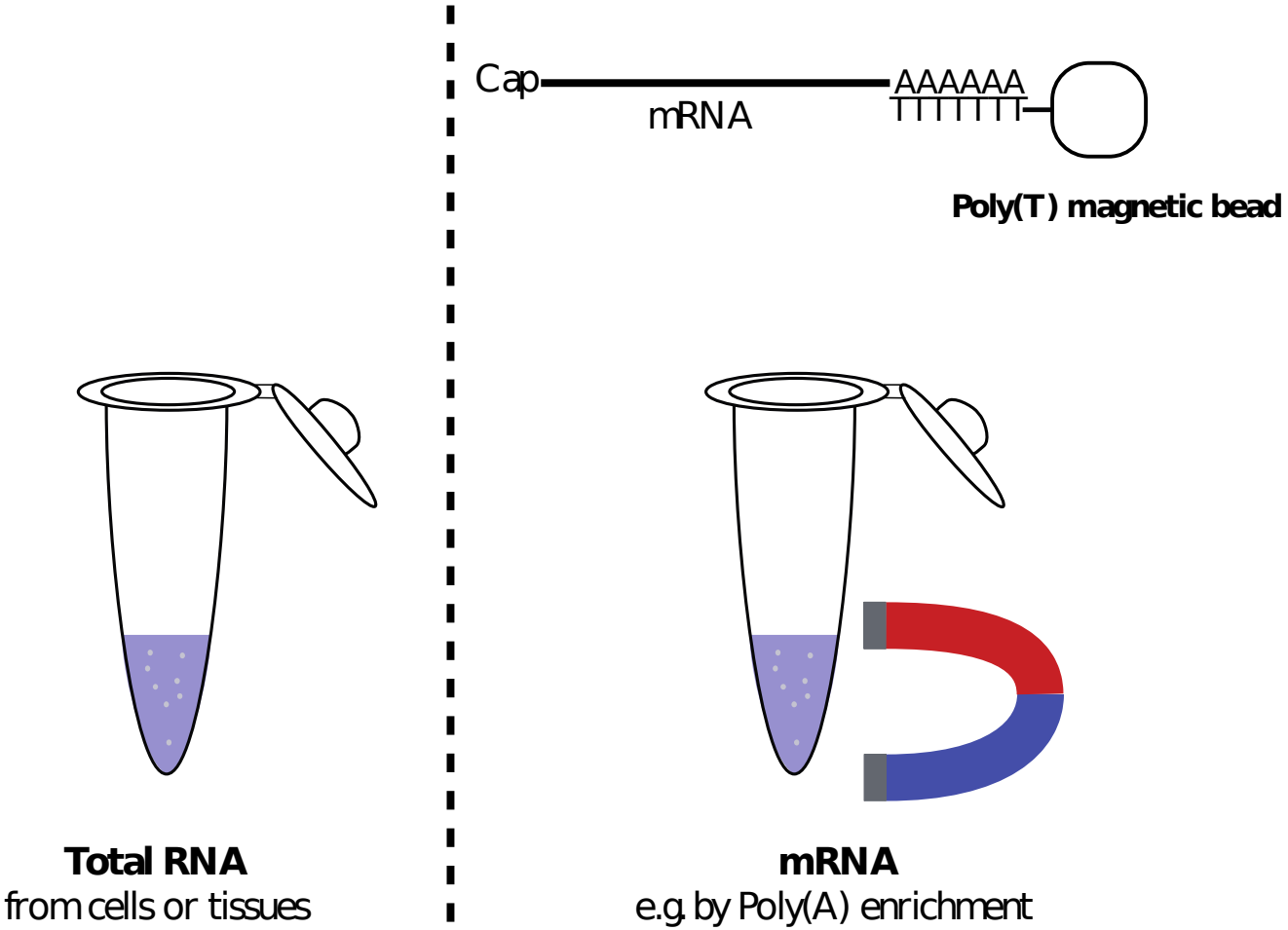
Aird et al., Genome Biology (2011)

Amplification

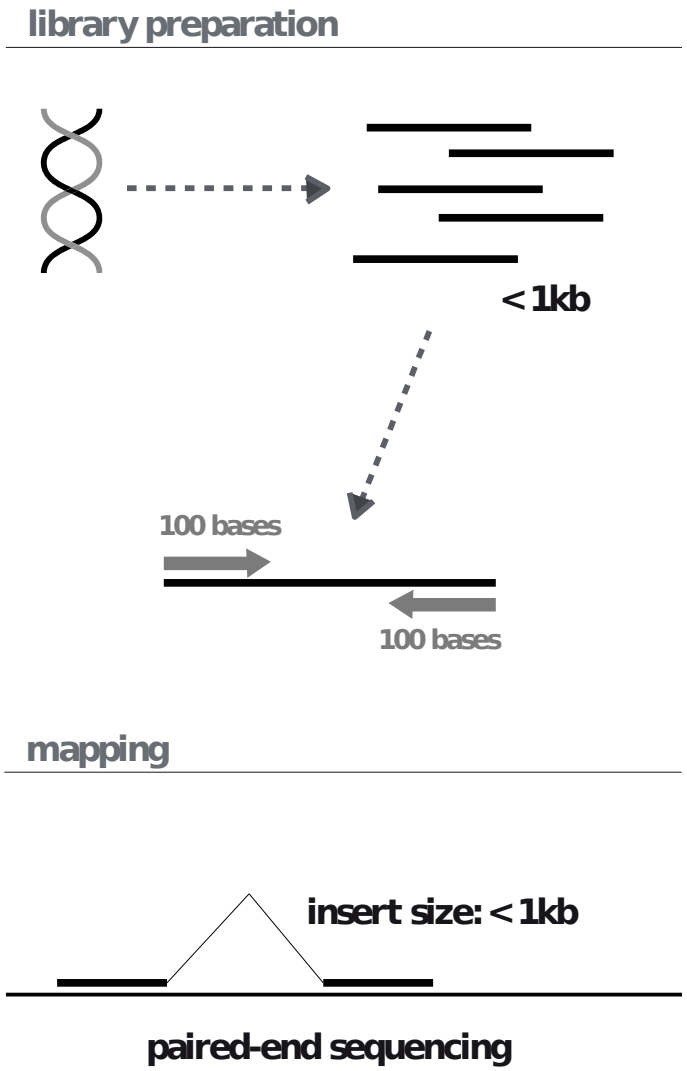


Library Preparation

RNA-seq library preparation

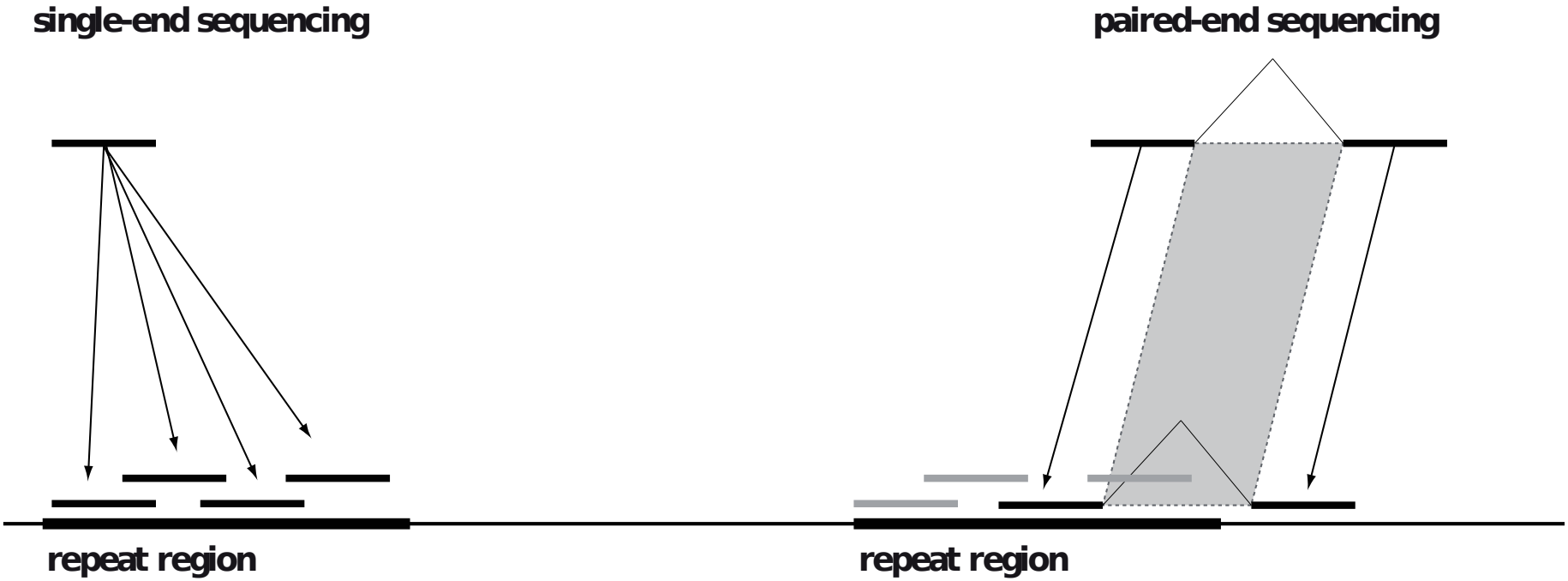


Paired-End Sequencing



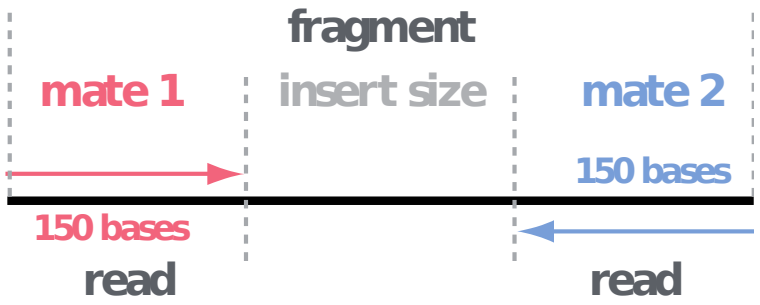
Paired-End Sequencing

mapping



Basic Notations

Basic Notations



fragment
molecule to be sequenced

read
One sequenced part of a biological fragment (mate1 and/or mate2)

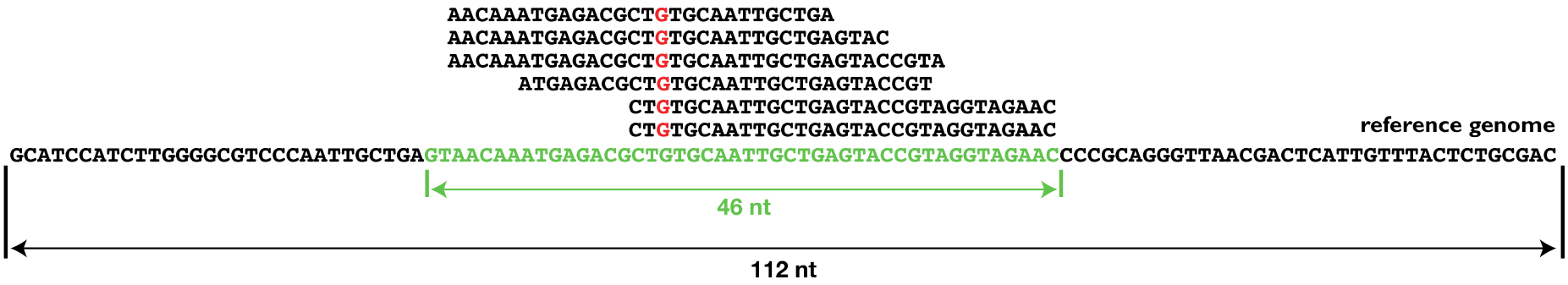
mate 1
sequence of the 5' end of paired-end sequencing

mate 2
sequence of the 3' end of paired-end sequencing

sequencing depth
The total number of all the sequences, reads or bases represented in a single sequencing experiment

coverage
whole genome: (# of sequenced bases) / (size of genome)
one locus: (# of bases mapping to the locus) / (size of locus)
one position: (# of reads overlapping with one position)

Coverage



whole genome: (# of sequenced bases) / (size of genome)

188 / 112 = 1.68 fold

one locus: (# of bases mapping to the locus) / (size of locus)

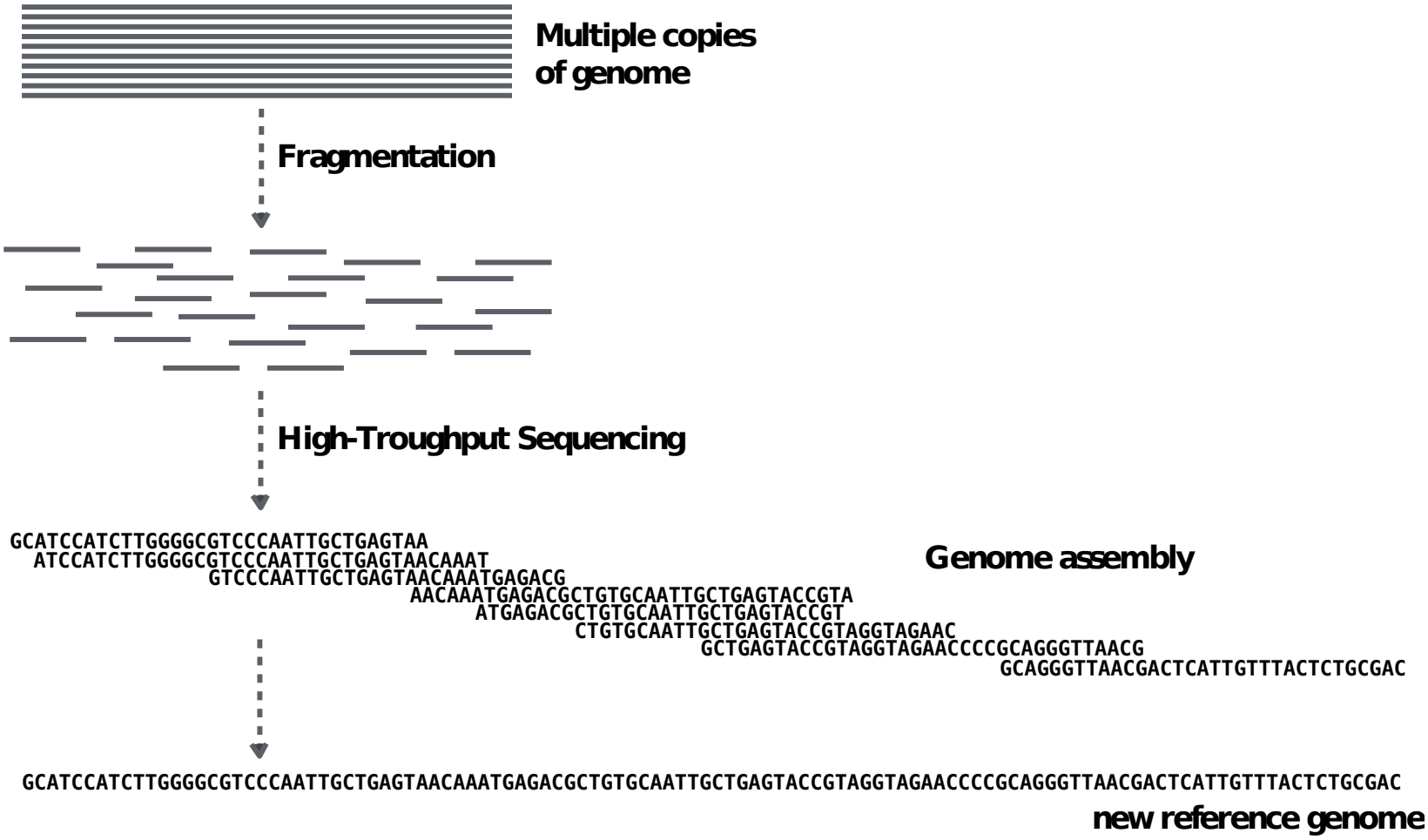
188 / 46 = 4.09 fold

one position: (# of reads overlapping with one position)

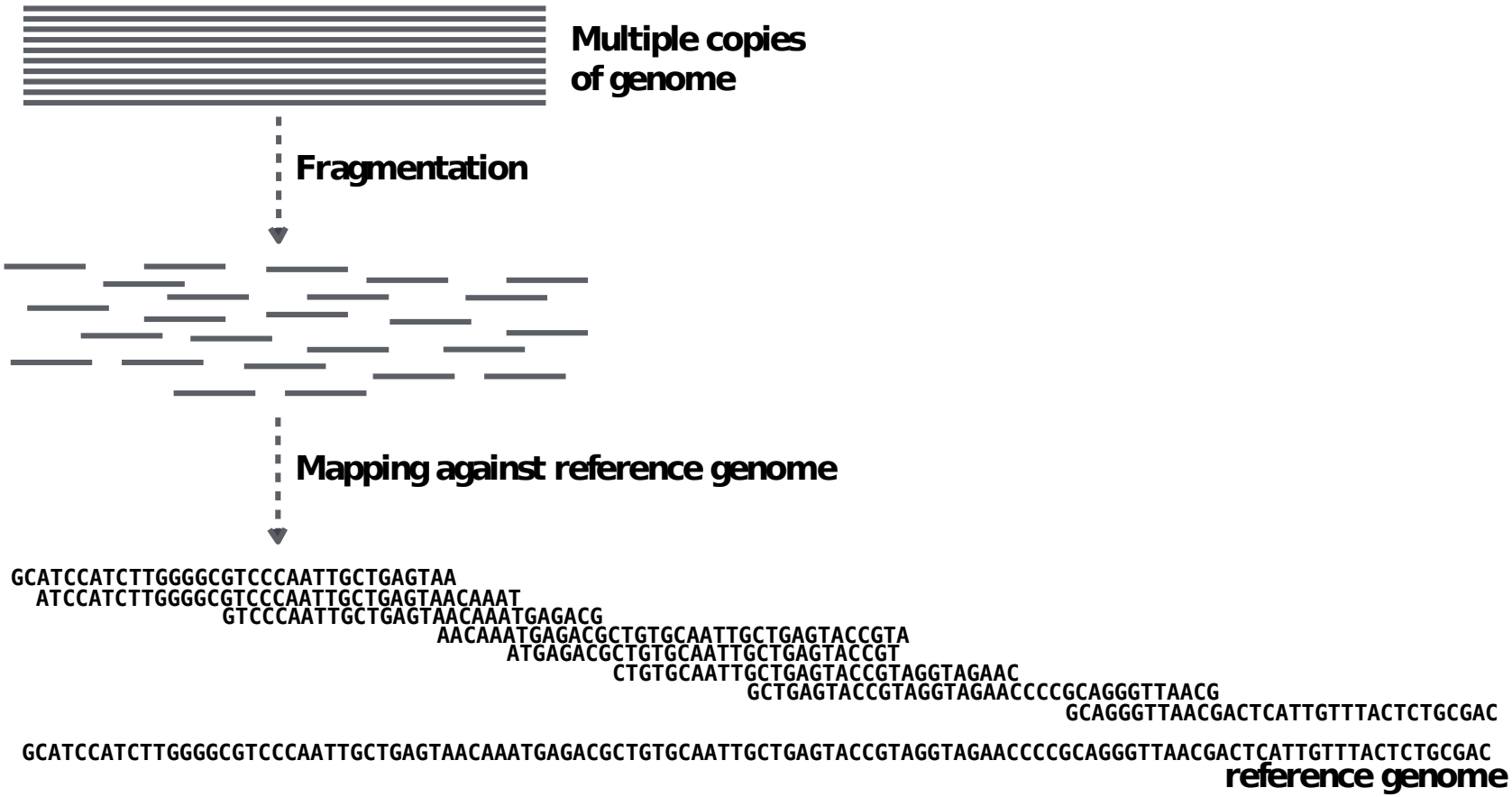
6 fold

Within the scope of sequencing

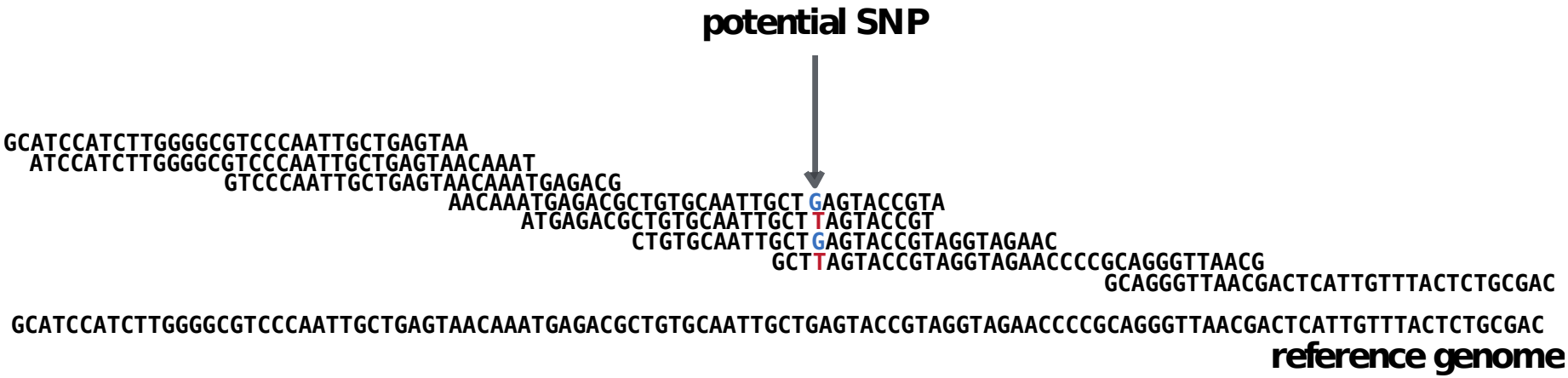
De-novo genome assembly



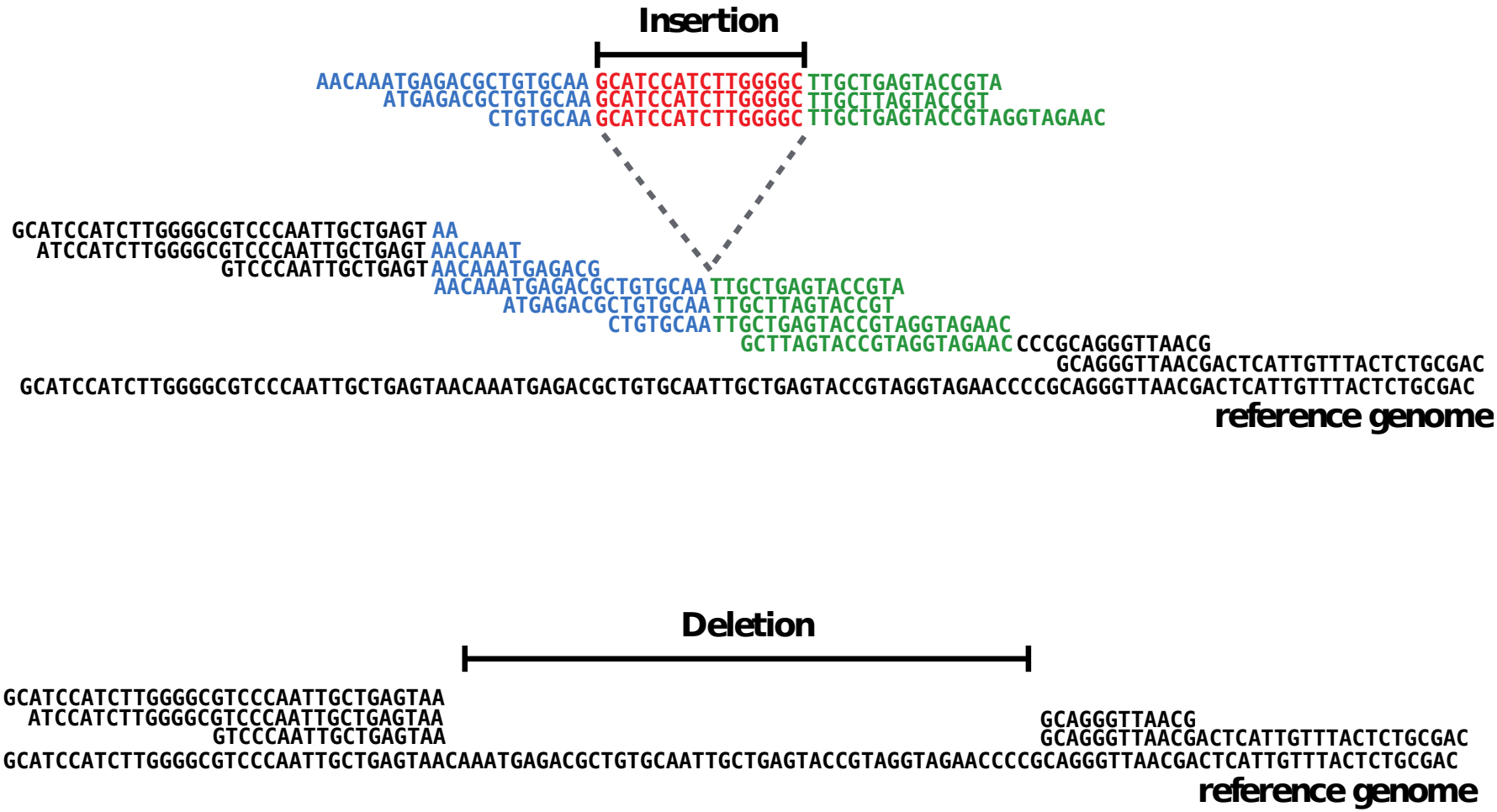
Genome re-sequencing



SNP discovery

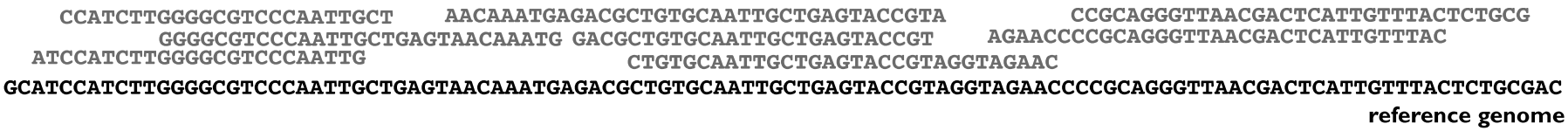


Genome rearrangements

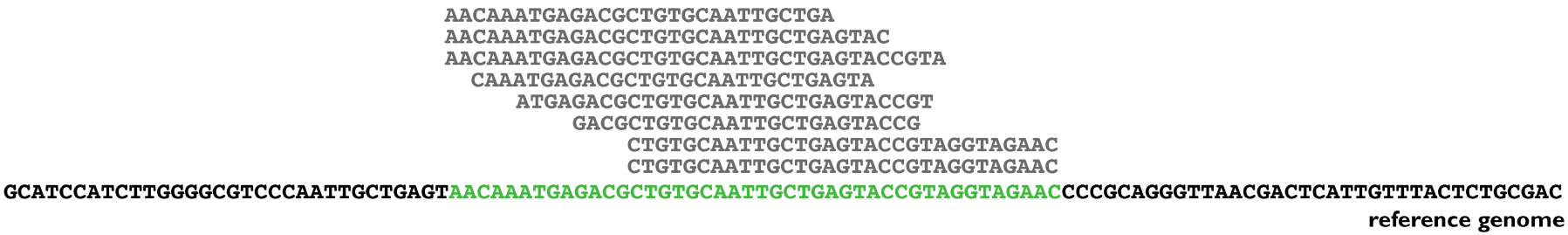


Exome sequencing / Targeted Sequencing

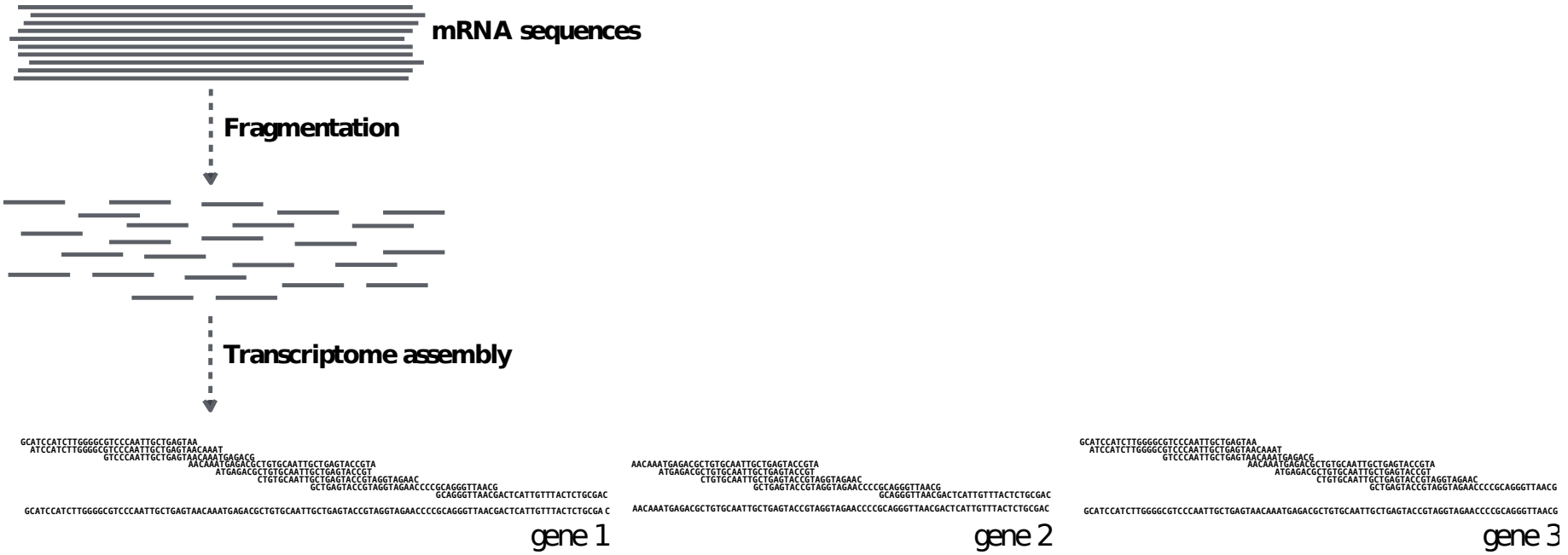
Whole Genome Sequencing



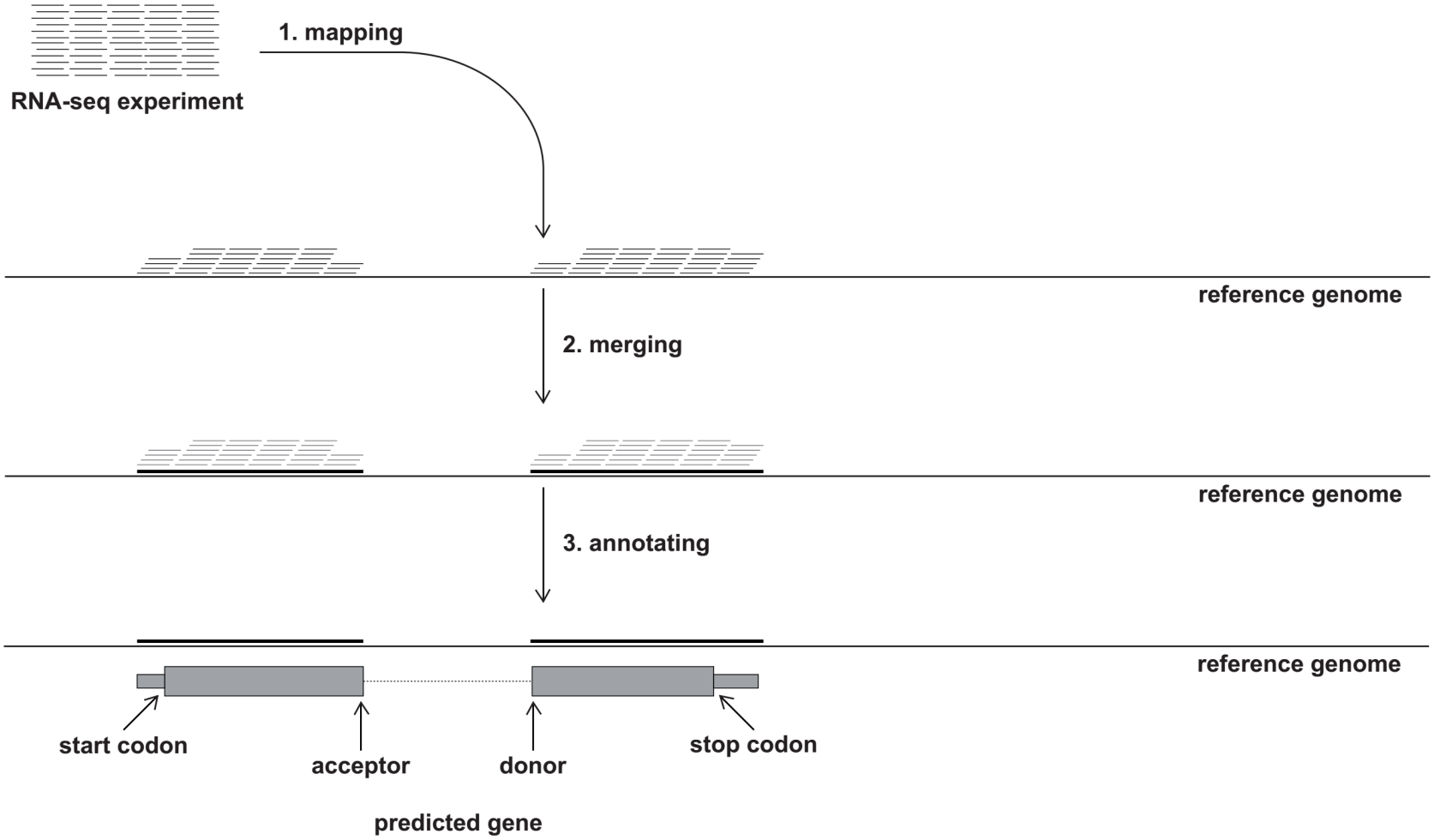
Targeted Sequencing



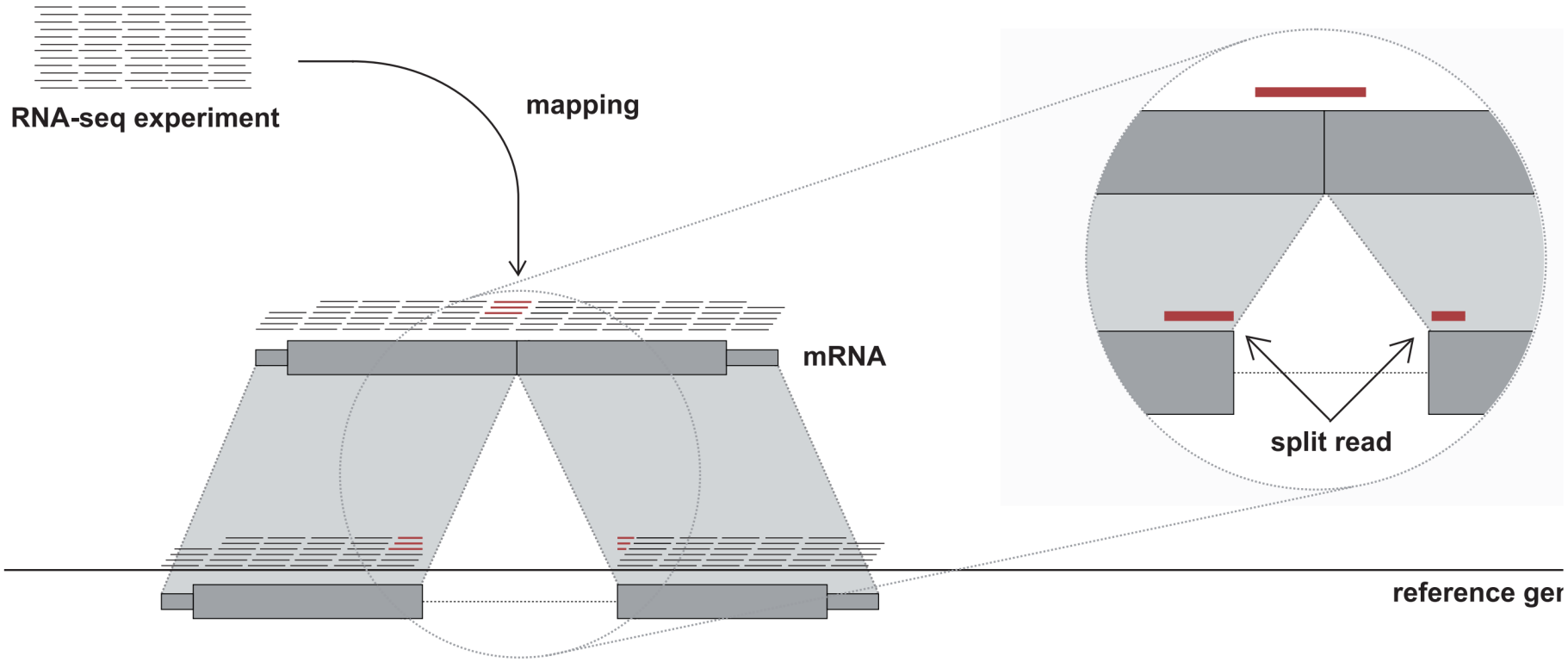
De-novo transcriptome assembly



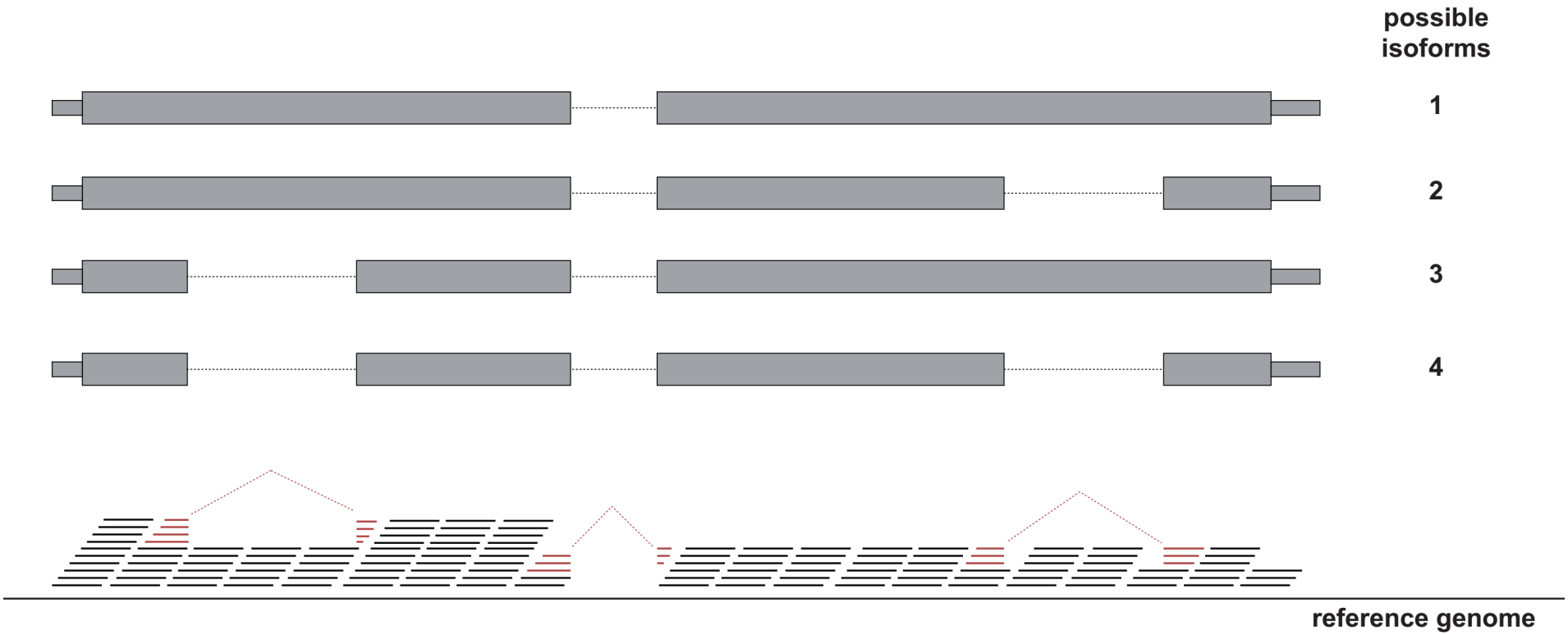
De-novo gene prediction



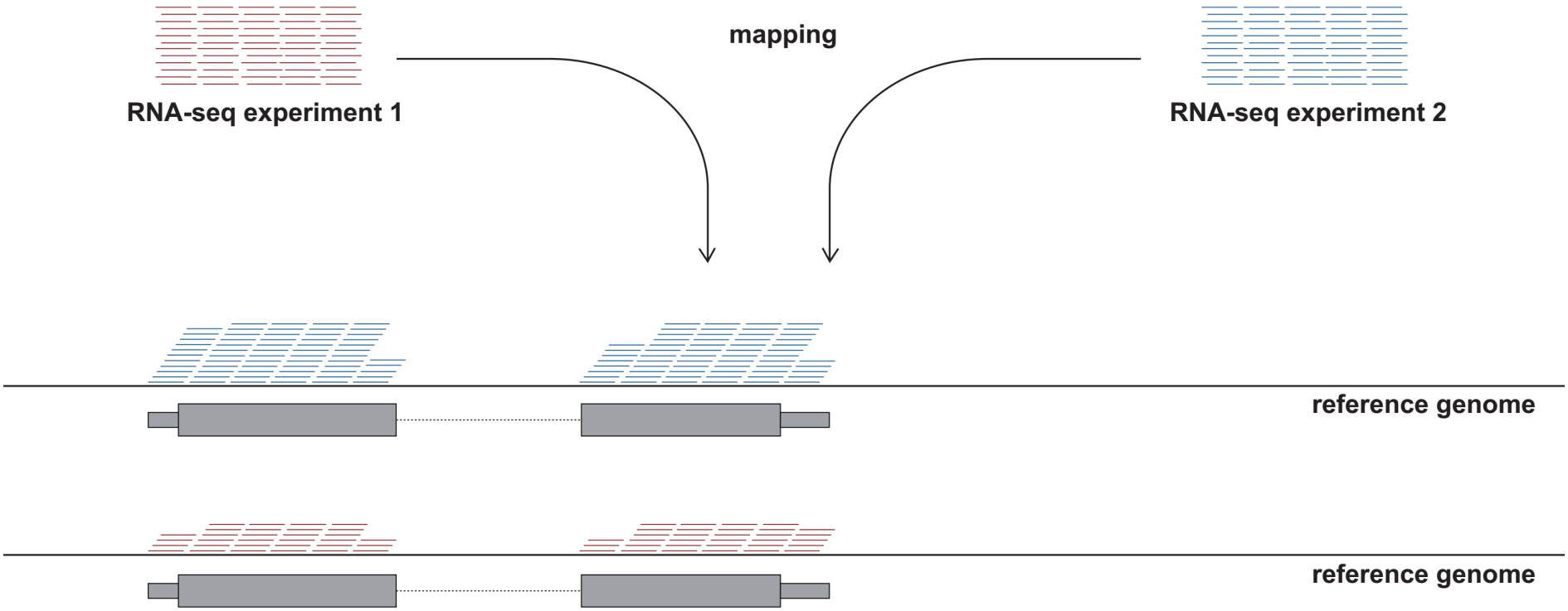
Splice site detection



Isoform differentiation

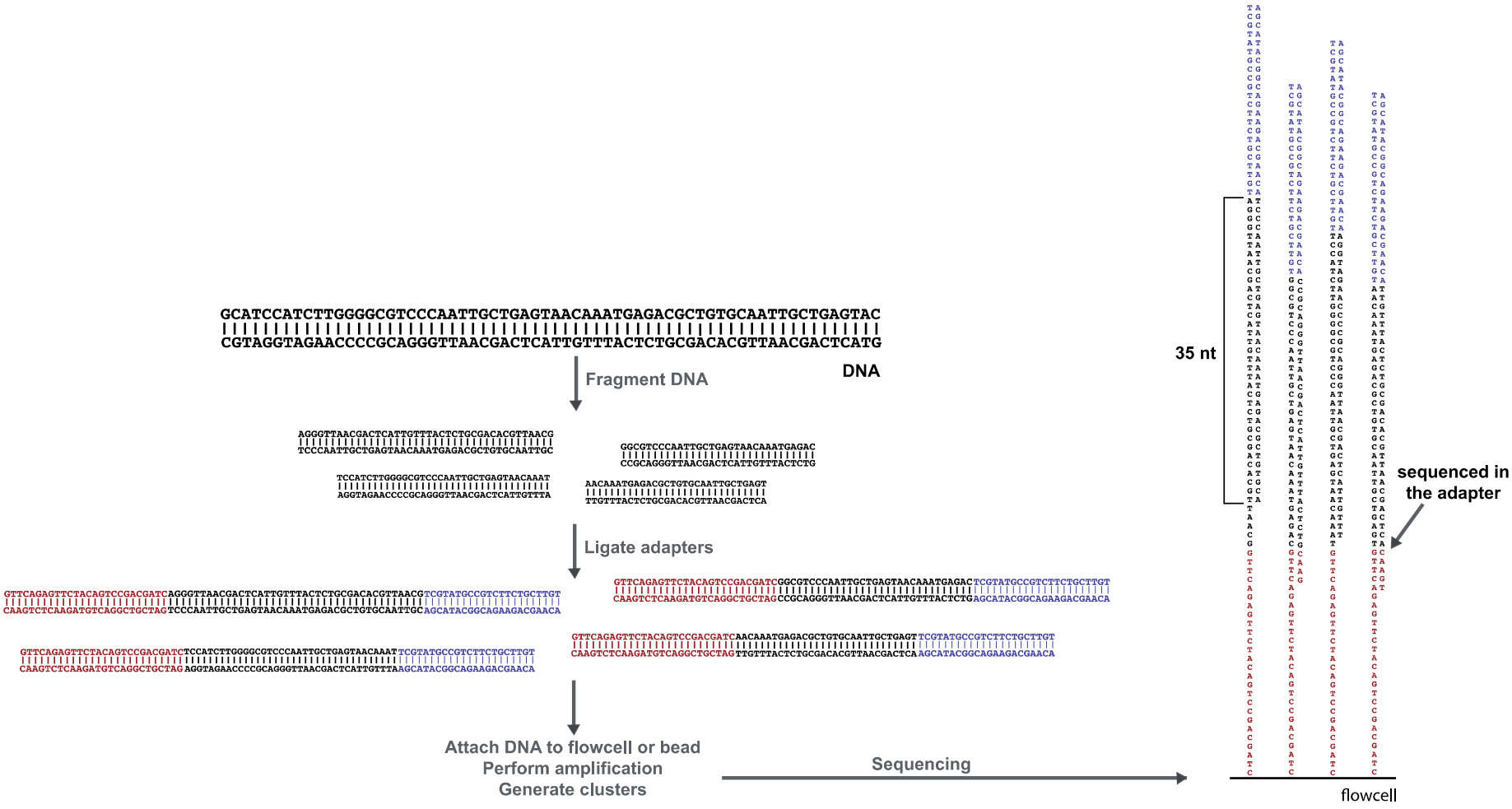


Differential gene expression



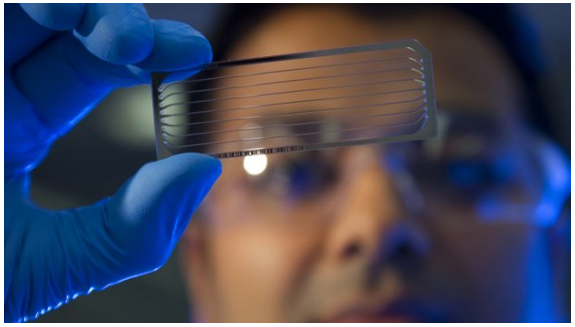
Base-calling to RAW FastQ

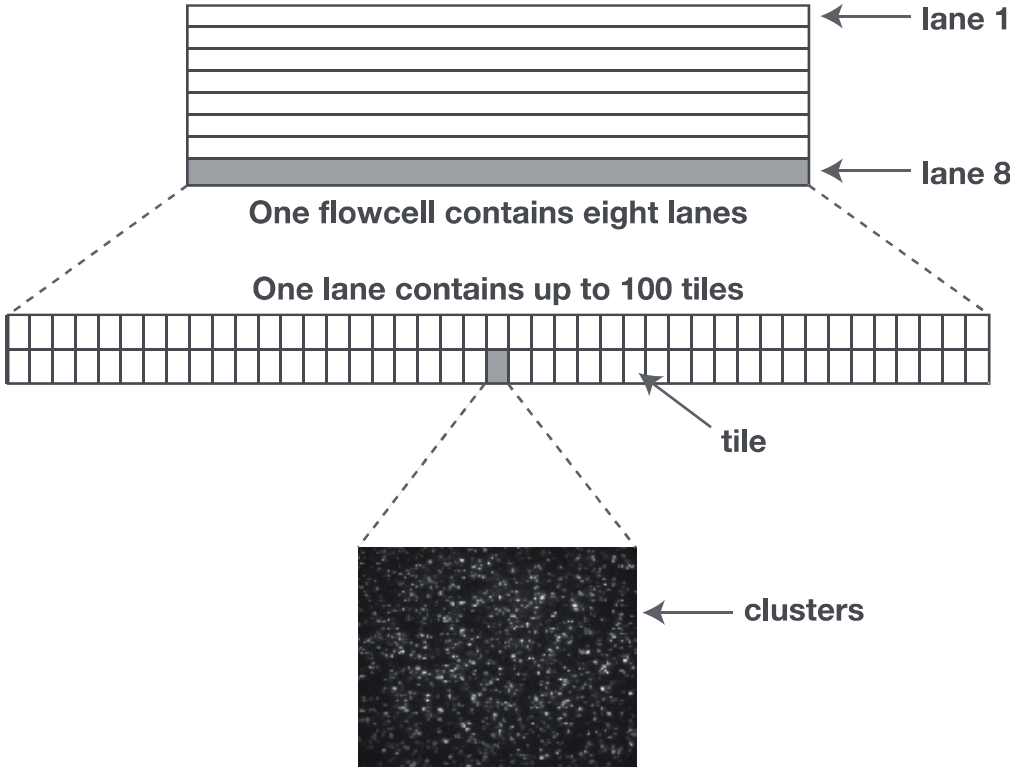
Base-calling



Base-calling

- Bases are called from normalized, k-mean clustered intensities of emitted fluorescence
- Cluster density influences the base-calling quality





FASTQ format (1st line)

```
$ zcat SRR359063_1.fastq.gz | head
@SRR359063.1 D042KACXX:3:1101:2690:2160 length=101
NCATCGTCCGGTATGTAGAACAGGGGAACCGGACGTTTTCCAAGGCGTAGCCATGTTAGACAAGGCGCAGATATA
GGTGATGCTGATGCAGAAAAACGATT
+SRR359063.1 D042KACXX:3:1101:2690:2160 length=101
#4=DBDDDHFHFFHIGHIIIJJJJJJJJJJJBHDAGHJGGGHIJHFFFFDDEDCCDCCCCDDDDDBDBD>CDEE
>C@CDDDDDDCACAACCDCDBDBB<1
@SRR359063.2 D042KACXX:3:1101:5202:2193 length=101
CTCTGGTACAGAACACGTGGATTATAAGAGTTGCCGCTTCGCACAGAAGTCGGAGTTCTCTCACCACTTTTGAGC
TCTTCCTCGGCTTCTTCTTCCTCTTT
```

SRR359063.1	run ID
D042KACXX	flowcell ID
3	flowcell lane
1101	tile number within the flowcell lane
2690	'x'-coordinate of the cluster within the tile
2160	'y'-coordinate of the cluster within the tile

FASTQ format (2nd line)

```
$ zcat SRR359063_1.fastq.gz | head
@SRR359063.1 D042KACXX:3:1101:2690:2160 length=101
NCATCGTCCGGTATGTAGAACAGGGGAACCGGACGTTTTCCAAGGCGTAGCCATGTTAGACAAGGCGCAGATATA
GGTGATGCTGATGCAGAAAAACGATT
+SRR359063.1 D042KACXX:3:1101:2690:2160 length=101
#4=DBDDDHFHFFHIGHIIIJJJJJJJJJJJJBHDAGHJGGGHIJHFFFFDDEDCCDCCCCDDDDDBDBD>CDEE
>C@CDDDDDDCACAACCDCDBDBB<1
@SRR359063.2 D042KACXX:3:1101:5202:2193 length=101
CTCTGGTACAGAACACGTGGATTATAAGAGTTGCCGCTTCGCACAGAAGTCGGAGTTCTCTCACCACTTTTGAGC
TCTTCCTCGGCTTCTTCTTCCTCTTT
```

Info
The raw sequence letters.

FASTQ format (3rd line)

```
$ zcat SRR359063_1.fastq.gz | head
@SRR359063.1 D042KACXX:3:1101:2690:2160 length=101
NCATCGTCCGGTATGTAGAACAGGGGAACCGGACGTTTTCCAAGGCGTAGCCATGTTAGACAAGGCGCAGATATA
GGTGATGCTGATGCAGAAAAACGATT
+SRR359063.1 D042KACXX:3:1101:2690:2160 length=101
#4=DBDDDHFHFFHIGHIIIJJJJJJJJJJJJBHDAGHJGGGHIJHFFFFDDEDCCDCCCCDDDDDBDBD>CDEE
>C@CDDDDDDCACAACCDCDBDBB<1
@SRR359063.2 D042KACXX:3:1101:5202:2193 length=101
CTCTGGTACAGAACACGTGGATTATAAGAGTTGCCGCTTCGCACAGAAGTCGGAGTTCTCTCACCACTTTTGAGC
TCTTCCTCGGCTTCTTCTTCCTCTTT
```

Info

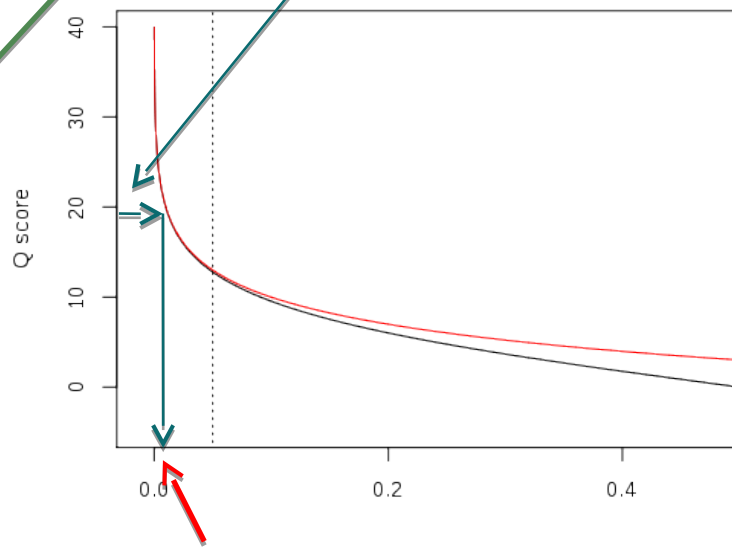
Begins with a '+' character and is optionally followed by the same sequence identifier (and any description) again.

FASTQ format (4th line)

```
#4=DBDDDHFHFFHIGHIIIJJJJJJJJJJJJJBHDAGHJGGGHIJHFFFFDDEDCCDCCCCDDDDDBDBD>CDEE
>C@CDDDDDDCACAACCDCDBDBB<1
```

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	De
0	0	000	NUL (null)	32	20	040		Space	6
1	1	001	SOH (start of heading)	33	21	041	!	!	6
2	2	002	STX (start of text)	34	22	042	"	"	6
3	3	003	ETX (end of text)	35	23	043	#	#	6
4	4	004	EOT (end of transmission)	36	24	044	$	$	6
5	5	005	ENQ (enquiry)	37	25	045	%	%	6
6	6	006	ACK (acknowledge)	38	26	046	&	&	7
7	7	007	BEL (bell)	39	27	047	'	'	7
8	8	010	BS (backspace)	40	28	050	(	(	7
9	9	011	TAB (horizontal tab)	41	29	051	)	)	7
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	7
11	B	013	VT (vertical tab)	43	2B	053	+	+	7
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	7
13	D	015	CR (carriage return)	45	2D	055	-	-	7
14	E	016	SO (shift out)	46	2E	056	.	.	7
15	F	017	SI (shift in)	47	2F	057	/	/	7
16	10	020	DLE (data link escape)	48	30	060	0	0	8
17	11	021	DC1 (device control 1)	49	31	061	1	1	8
18	12	022	DC2 (device control 2)	50	32	062	2	2	8
19	13	023	DC3 (device control 3)	51	33	063	3	3	8
20	14	024	DC4 (device control 4)	52	34	064	4	4	8
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	8
22	16	026	SYN (synchronous idle)	54	36	066	6	6	8
23	17	027	ETB (end of trans. block)	55	37	067	7	7	8
24	18	030	CAN (cancel)	56	38	070	8	8	8

52 – 33 = 19 = Phred-(33)-Score



the probability that the corresponding base call is incorrect

FASTQ format - Quality values

- Error rates are inferred by reference sequencing of Phi X bacteriophage during calibration
- Illumina sequencer perform a mapping to Phi X known reference genome
- Mismatches of mapped reads lead to the calculation of base-calling error probabilities

Phred Q-Score	Error probability	Accuracy
10	1 in 10	90%
20	1 in 100	99%
30	1 in 1000	99.9%
40	1 in 10000	99.99%

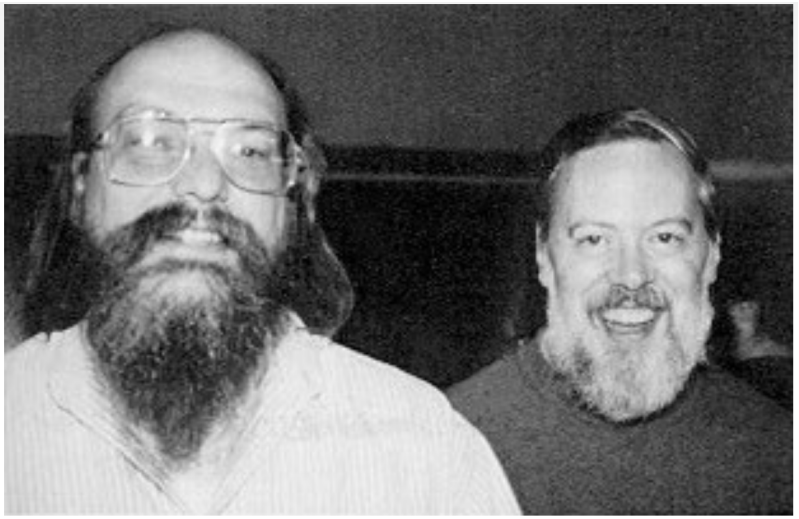
Unix and GNU/Linux

What is an operating system?

- The set of all the programs needed to use a machine
  - A software collection to coordinate all hardware components (kernel)
  - A software to draw windows (graphical interface)
  - Software to edit texts, browse Internet, install other software, etc..

Unix

- To solve the problem of computer compatibility, the Bell labs 1969 officially released Unix - developed by Ken Thompson (l) and Dennis Ritchie (r)
- The concepts of its kernel can be found in todays Linux distributions and Android based phones



Innovations introduced by Unix

- A novel approach to organize files and scripts
- Monolithic kernel including all hardware drivers
- Fully backward compatibility
- Free license for universities and companies
- Open source

- Good and efficient approach to data analysis
  → Unix was rapidly adopted worldwide, and became the standard operating system, especially in universities

GNU

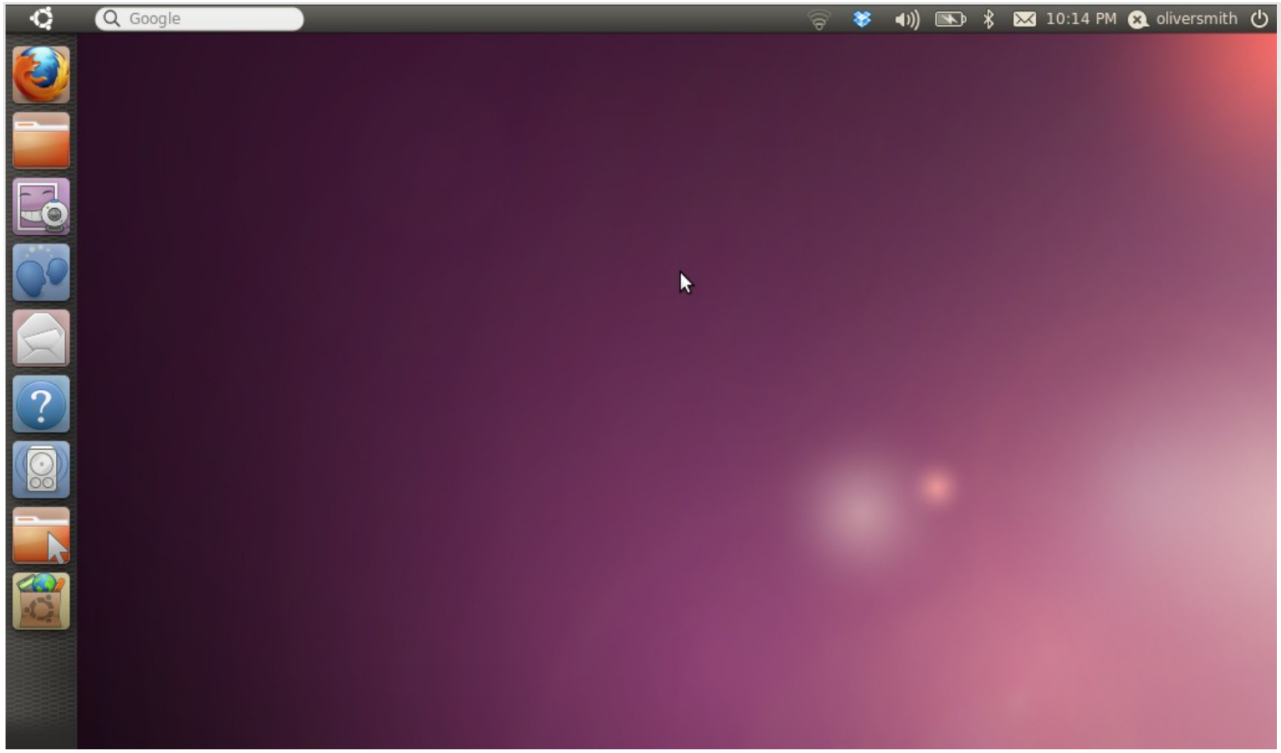
- GNU is the name of an operating system, inspired to Unix and distributed as free software
- It was never fully completed, but most of its tools are still used
- Most of the commands we will see today are GNU

Linux

- The original GNU project was never fully completed
- In 1991, it was merged with another project called Linux
- Linux provided the last components that were missing in GNU
- When it appeared in 1991, GNU/Linux finally provided a free Unix-like operating system
- Thanks to the adoption of GNU/Linux servers, Internet grew considerably in 1991, 1992

Linux - Ubuntu

- One of the most popular distributions
- Good for novices (no much extra configuration)



Linux - Fedora

- The community project of commercial Red Hat Linux (famous Linux for enterprise environments)
- Popular in many universities

Linux - Debian

- One of the most distributed and oldest Linux OS (1993)
- Basis of many other Linux distributions

Linux - BioLinux

- Distribution for bioinformaticians
- Ships with bioinformatics tools (Blast, BioPerl …)

MacOS

- MacOS system is also inspired to Unix
- In principle, you can do bioinformatics and follow this course on a MacOS system
- Installing bioinformatics specific software may be a bit more difficult (see homebrew package manager)

Differences of Linux distributions

- The default software included when you install the system
- The default configuration
  - Desktop
  - Windows
  - Package manager for software repositories
- Some technical details such as the libraries used to compile the software
- Some Linux distributions include only free software, others are less strict
- However, all the software available for Linux is installable in any other distribution

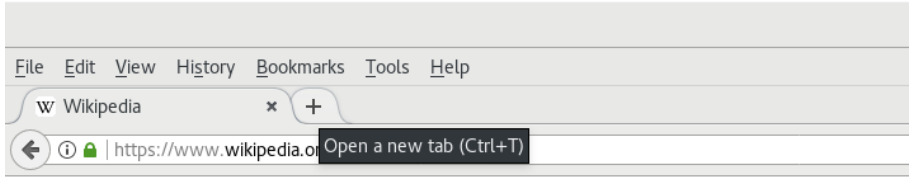
Howto install software in Linux

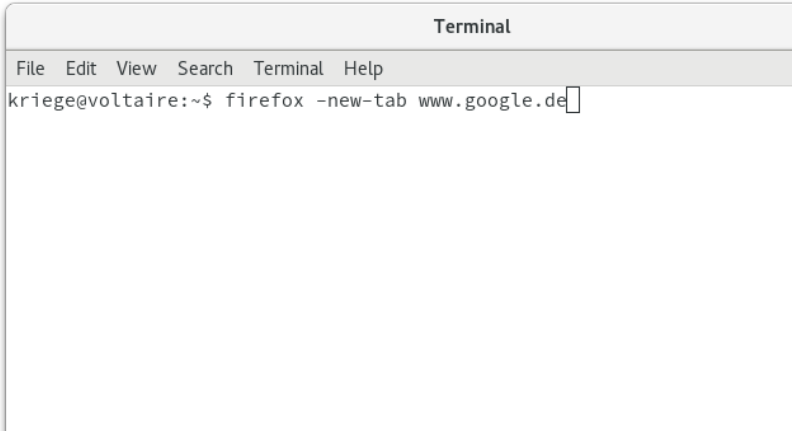
- Setup.exe → NO!
- Use repositories of compatible software collections
- “”App-Stores””
  - DEP → Debian Packages
  - RPM → Red Hat Package Manager
  - Conda → Python based package manager
- Compile source code

Command Line Interface

Terminal emulator

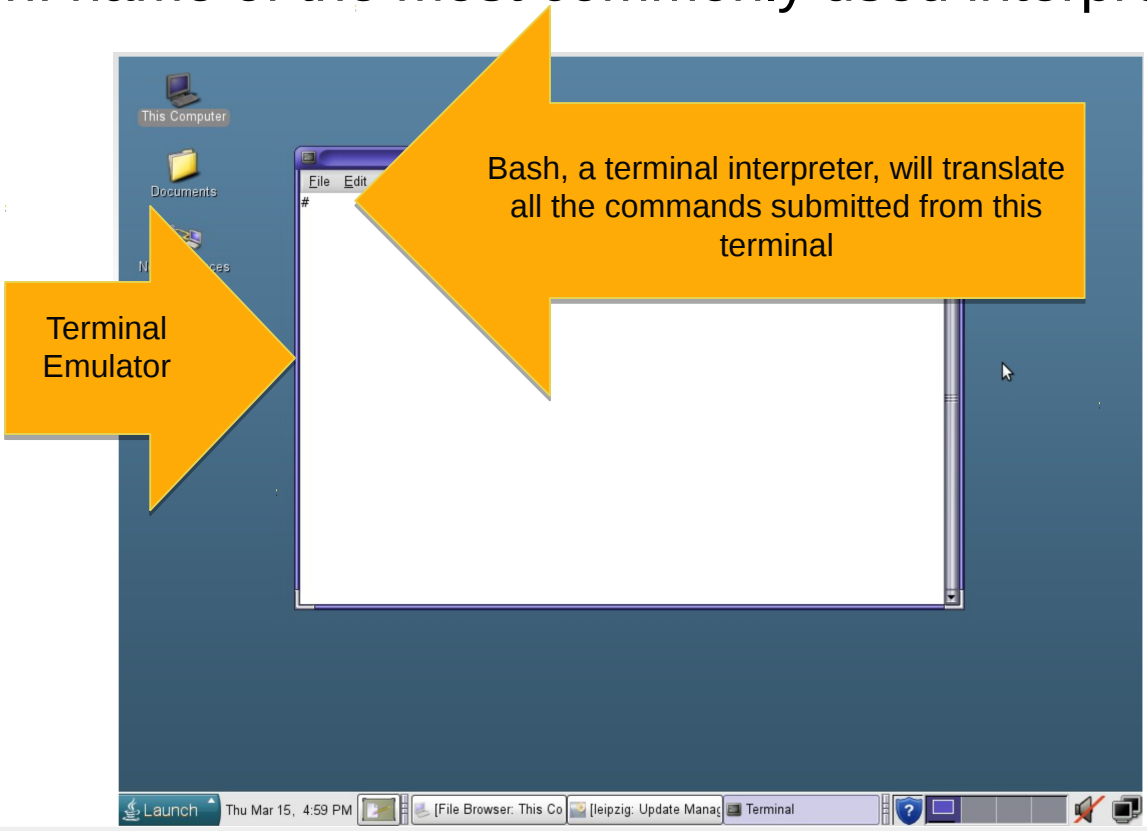
- The terminal emulator is a software that starts the command line interface (CLI)
- CLI allows to execute commands by typing
- Instead of clicking an icon in a menu, we call a software by writing its name





Terminal emulator

- Interpreter: the software that translate commands to the computer
- Bash: name of the most commonly used interpreter



Why do we need the terminal?

- Handling many and big text files
- Efficient and fast controlling of the OS
- Automated control of software and their parameters
- Combining commands

The Unix Philosophy

*“Write programs that do one thing and do it well.
Write programs to work together. Write programs to handle text streams, because that is a universal interface.”*

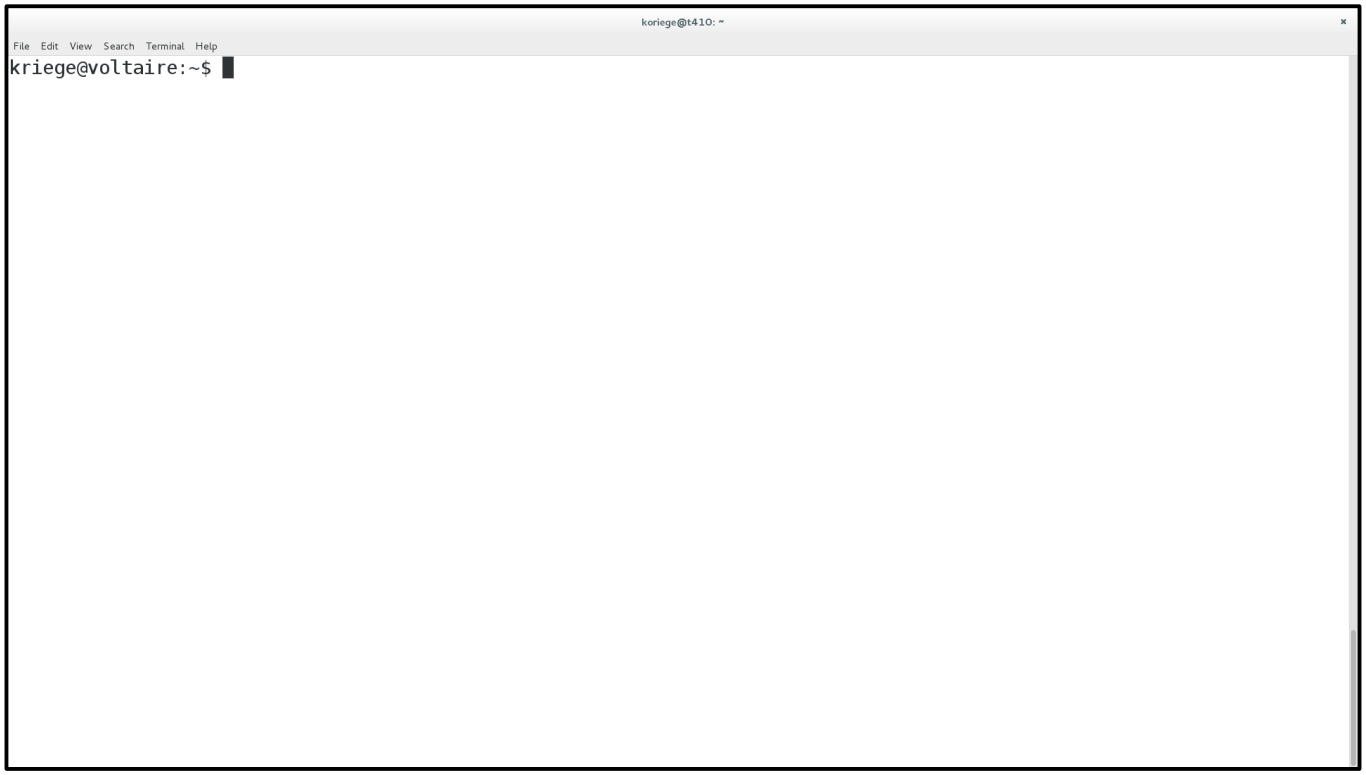
(Doug McIlroy, Unix pipes inventor)

The Unix Philosophy

- The Unix Philosophy can be summarized as
  - Store everything on human readable text files
  - Write small programs that carry out single task
  - Chain small programs to perform more complex tasks

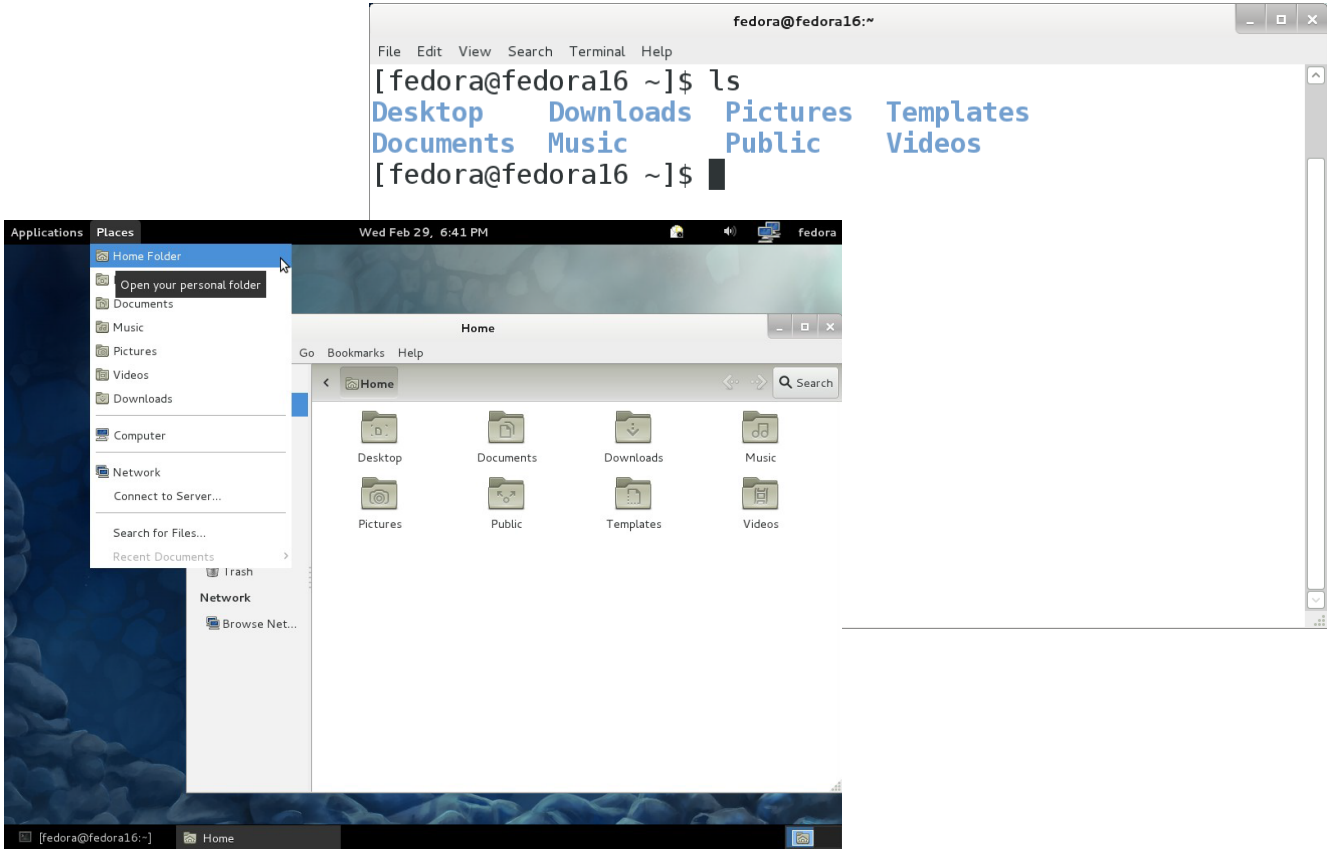
Are you there?

- https://github.com/destairdenbi/trainings
- Open your terminal



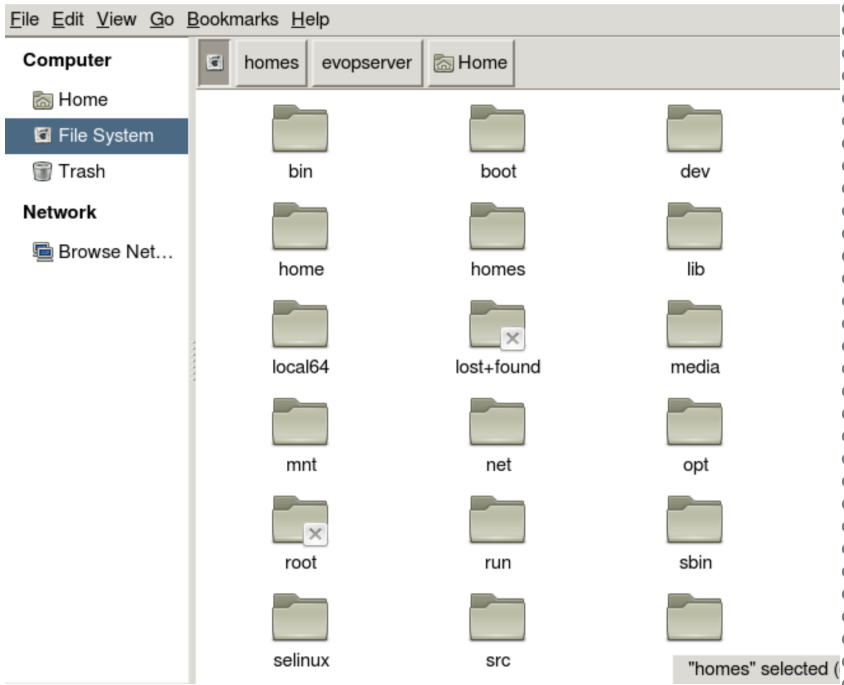
ls – list files and directories

- If you want to see which files are being shown by ls, you can type nautilus or dolphin instead to open the file manager
  1. ls
  2. nautilus
  3. dolphin



cd - change directory

- Let's start navigating the file system!
- The command “cd” allows you to move to another folder
- Enter the root folder “/” (Root space)
  1. cd /
  2. ls




```
kriege@voltaire:/$ ls -l
total 136
drwxr-xr-x   2 root   root   4096 Sep 19 08:42 bin
drwxr-xr-x   3 root   root   4096 Sep 20 08:44 boot
drwxr-xr-x   3 root   root   4096 Sep 19 09:51 data
drwxr-xr-x   4 root   root   4096 Sep 27 13:55 data2
drwxr-xr-x  21 root   root   4220 Sep 27 13:53 dev
drwxr-xr-x 140 root   root  12288 Sep 29 11:35 etc
drwxr-xr-x  42 root   root      0 Sep 29 10:49 gen
drwxr-xr-x   2 root   root   4096 Sep 29 10:49 gsc
drwxr-xr-x  37 root   root      0 Sep 29 10:49 home
drwxr-xr-x  12 root   root   4096 Sep 19 08:55 lib
drwxr-xr-x   7 root   root  12288 Sep 19 09:49 lib64
drwx------   2 root   root  16384 Sep 19 07:49 lost+found
drwxr-xr-x  12 root   root      0 Sep 29 10:49 misc
drwxr-xr-x   2 root   root   4096 May 10 01:26 mnt
drwxr-xr-x  26 root   root   4096 Sep 19 10:02 oldroot
drwxr-xr-x   4 root   root   4096 Sep 19 10:13 opt
dr-xr-xr-x 607 root   root      0 Sep 26 10:17 proc
drwx------  14 root   root   4096 Oct  2 13:27 root
drwxr-xr-x  32 root   root   1160 Oct  2 13:06 run
drwxr-xr-x   2 root   root  12288 Sep 19 09:04 sbin
drwxr-xr-x   9 root   root      0 Sep 29 10:49 scratch
drwxr-xr-x   2 root   root   4096 May 10 01:26 selinux
drwxr-xr-x   5 root   root   4096 Sep 19 07:57 srv
drwxr-xr-x   4 root   root   4096 Sep 22 10:45 ssd
drwxr-xr-x   2 kriege users  4096 Sep 22 10:46 ssd2
drwxr-xr-x   2 root   root   4096 Sep 19 09:47 suse423
dr-xr-xr-x  13 root   root      0 Oct  2 11:27 sys
drwxrwxrwt  16 root   root  12288 Oct  2 15:50 tmp
drwxr-xr-x   3 root   root   4096 Sep 19 08:46 u
drwxr-xr-x  14 root   root   4096 Sep 19 09:14 usr
drwxr-xr-x  13 root   root      0 Sep 29 10:49 usr2
drwxr-xr-x  12 root   root   4096 Sep 19 09:51 var
kriege@voltaire:/$
```

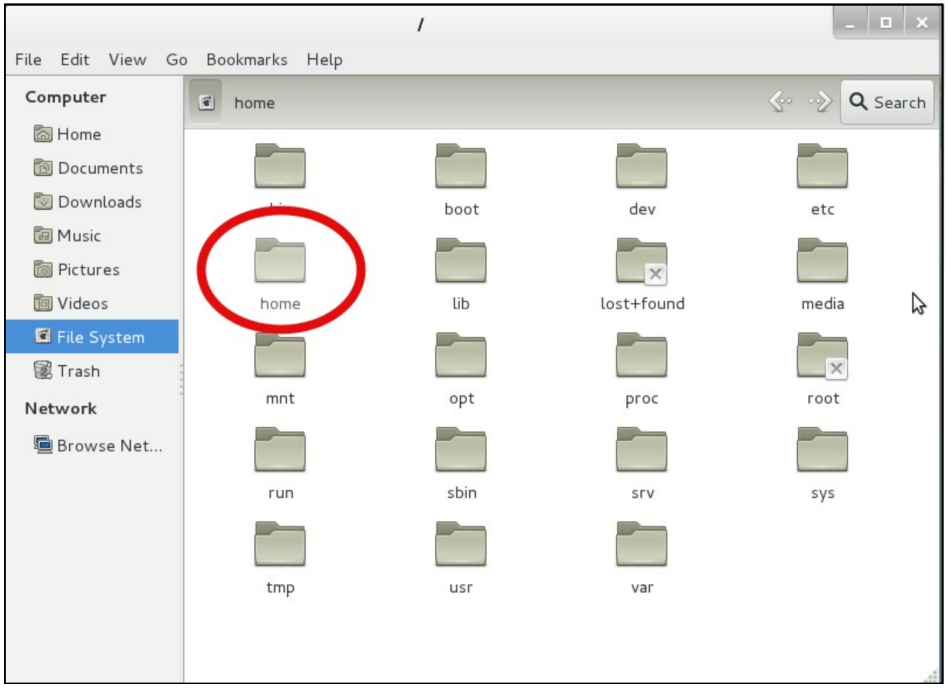
Root space

- Modifications need administration privileges
- /bin, /usr, /local, /sys → these folders contain the software installed
- /etc, /var → systemwide configuration files
- /home → contains users' private files
- Ignore rest for now

Home space - User space

- The /home folder contains all the users' folder
- Only the administrator (root) has the right to manipulate other users's directories
- If you open it, you will see folders for each user

1. cd home
2. ls
3. cd <USER>



Anatomy of a command in the terminal

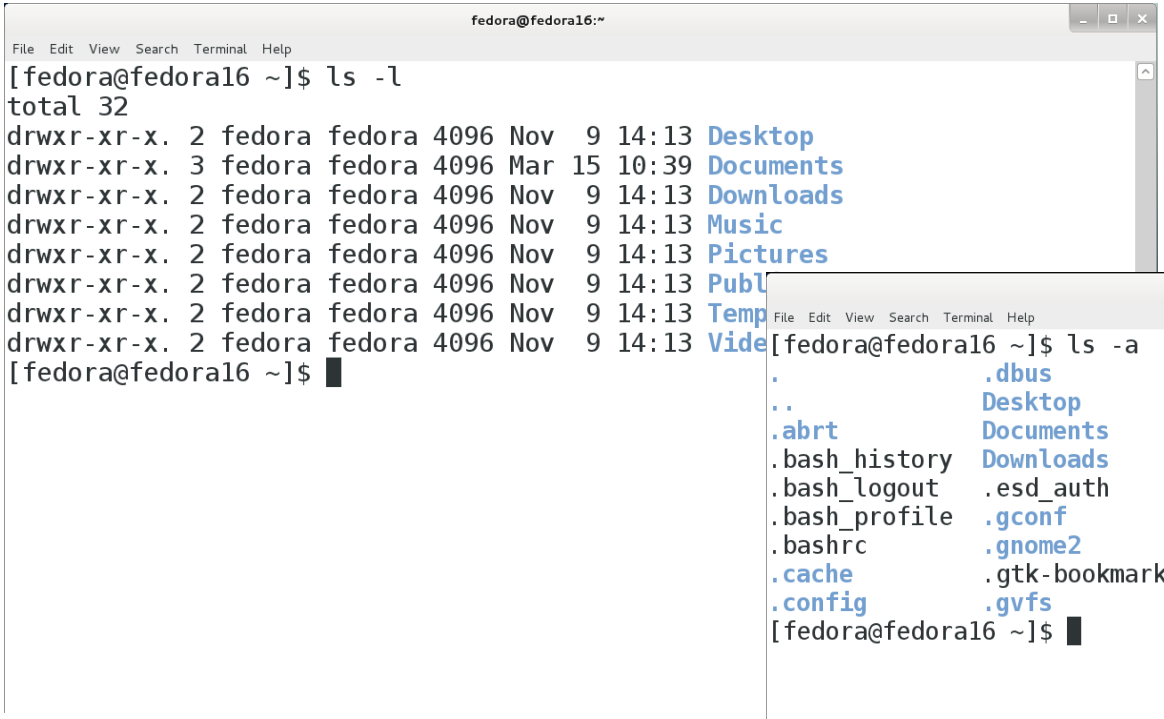
- Each command call is usually composed by three parts
  - The command itself
  - Parameters (optionals)
  - Arguments

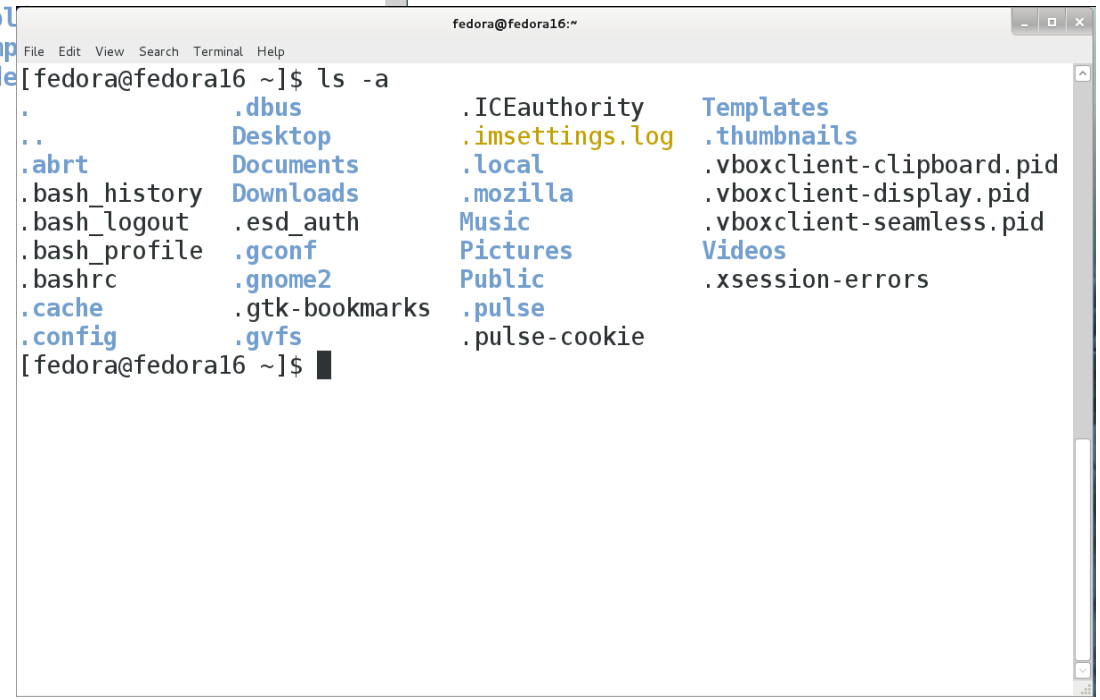
  ls -l /home/<USER>/

ls - Parameters

- Parameters are optional items that can be used to customize the behavior of a command
- For example
  1. ls -l → shows the list of files in a long format
  2. ls -a → shows hidden files → they start with a “.”
  3. ls -t → list files sorted by modification time
  4. ls -l -a -t → adjust ls by multiple parameters
- Some parameters have a long and a short syntax
  - ls -a
  - ls --all

ls - Parameters





How to get the documentation of a command?

- Three methods
  - --help parameter
  - man command
  - Info command
- The simplest way to get the documentation of a command is by using the “--help” parameter
- Some Unix programs also accept the short syntax “-h”

1. ls --help
2. man ls

How to get the documentation of a command?

```
LS(1)                          User Commands                          LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuvSUX nor --sort.

       Mandatory arguments to long options are  mandatory  for  short  options
       too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
              with -l, print the author of each file

       -b, --escape
              print C-style escapes for nongraphic characters

       --block-size=SIZE
              use SIZE-byte blocks.  See SIZE format below

       -B, --ignore-backups
              do not list implied entries ending with ~

       -c     with -lt: sort by, and show, ctime (time of last modification of
              file status information) with -l: show ctime and  sort  by  name
              otherwise: sort by ctime

       -C     list entries by columns

       --color[=WHEN]
              colorize  the  output.   WHEN  defaults  to  `always'  or can be
              `never' or `auto'.  More info below
 Manual page ls(1) line 1
```

Name of command

Synopsis (how to use it)
Options in square brackets are optional

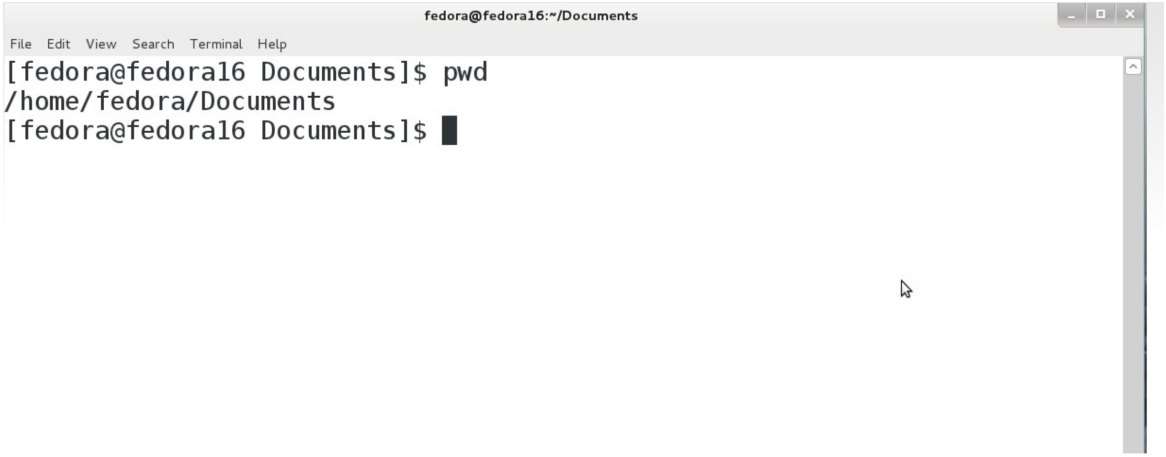
Parameters & arguments

ls --color=auto -a /home/<USER>

Where am I?

- If you don't know which folder are you in, you can use the command pwd to get the current working directory
- Typing “cd” without arguments will always bring you back to your home directory

1. cwd
2. cd
3. pwd



Handling paths

- Files and directories can be reached by absolute paths beginning with root “/”
  1. cd /home/<USER>/Documents
- cd “..” lets you return to the relative parent folder of the current working directory
  1. cd .. → returns to /home/<USER>
  2. cd Documents
  3. Cd ../../.. → returns to /

Bash completion

- You can use the <TAB> key on the keyboard to complete commands and arguments
- Thanks to tab completion, you can save a lot of typing
- Example:
  1. cd
  2. cd Doc<tab> → completes to “cd Documents”

Other commands (1)

1. mkdir mydir → create directory
2. ls
3. mv mydir yourdir → move a file or directory
4. ls
5. cp -r yourdir mydir → copy a file or directory
6. ls
7. rm -r yourdir → delete a file or directory (irretrievable)
8. ls

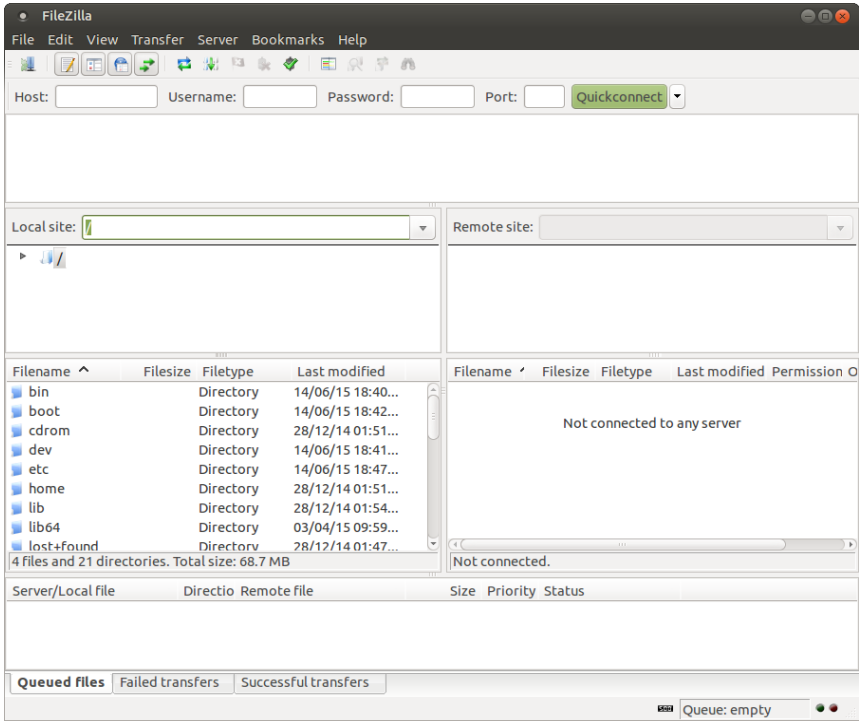
Other commands (2)

1. echo “hello” → print a message
2. echo $PWD → value of variable $PWD → in-house variable which stores the current path (pwd)
3. history → show the history of the commands typed

- gunzip → Extract *.gz compressed files
- unzip → Extract *.zip compressed files

Other commands (3)

- wget → download large files from the internet
  - Can continue interrupted downloads like FileZilla using parameter “--continue” or short “-c”
- md5sum → check digital fingerprint of a file to for example verify a downloaded file



Get hands on real data

1. wget <URL>/Homo_sapiens.GRCh37.87.gtf.gz
2. ctr+c
3. wget -c <URL>/Homo_sapiens.GRCh37.87.gtf.gz
4. gunzip Homo_sapiens.GRCh37.87.gtf.gz
5. mv Homo_sapiens.GRCh37.87.gtf hg19.gtf

6. wget <URL>/GCA_000009045.1_ASM904v1_genomic.fna.gz
7. md5sum GCA_000009045.1_ASM904v1_genomic.fna.gz
   → should be 60b4445468b67e1c2cf7ab2b1d2ede32 according to <URL>/md5checksums.txt

8. gunzip GCA_000009045.1_ASM904v1_genomic.fna.gz
9. mv GCA_000009045.1_ASM904v1_genomic.fna b_subtilis.fasta

Grep

- The command grep allows to search for patterns in a file
- Basic usage:
  - grep “searchword” filename
- Most command line tools are case sensitive
- Note that grep is case sensitive. “SeARchWord” is different than “searchword”
- Parameter “-i” ignores the case
- Example:
  1. grep “IL2” hg19.gtf
  2. grep -i “il2ra” hg19.gtf

Redirection

- It may be useful to save the results of a command to a file
- To do this, you can use the “>” operator
- To append results to a file use to “>>” operator
- Example:

  1. grep “IL2” hg19.gtf > hg19_il2.gtf
  2. grep “IL10” hg19.gtf >> hg19_il2.gtf

Cut

- Extract a column from a file
- Basic cut usage:
  - cut –f <column> <filename>
- The exercise file has a lot of columns
- Let's use cut to extract only the fourth column of the file

  1.head hg19_il2.gtf

  2.cut –f 4 hg19_il2.gtf

Sort

- The sort command can be used to sort a file by a column
- Basic sort usage:
  - sort -k <column> <filename>
- Let's sort the sample file by chromosome position
  1.sort –k 2 hg19_il2.gtf

Each tool is specialized on a task

- Did you notice that each of the tools we have seen today is specialized on one single task?
  - grep searches for a word in a file
  - cut extracts columns
  - sort orders a file
- Almost all Unix command line tools have one task
  - diff
  - find
  - echo
  - head
  - sleep
  - ...

Each tool is specialized on a task

- The Unix approach to programming is to write small programs specialized on single tasks, and organize them together
- To chain commands together, we will use the “pipe” operator “|”
- The pipe operator redirects the output of a command into another one

Chaining commands

1. grep –i “il2ra” hg19_il2.gtf | cut -f 4
2. grep –i “il2ra” hg19_il2.gtf | cut -f 4 | sort
3. grep –i “il2ra” hg19_il2.gtf | cut -f 4 | sort | wc

- wc stands for “word count”
- It counts the number of words, lines and characters
- Example:
  1. wc hg19.gtf → 2612766 102091714 1188326231
- Number of lines / number of words / number of characters

Advanced commands

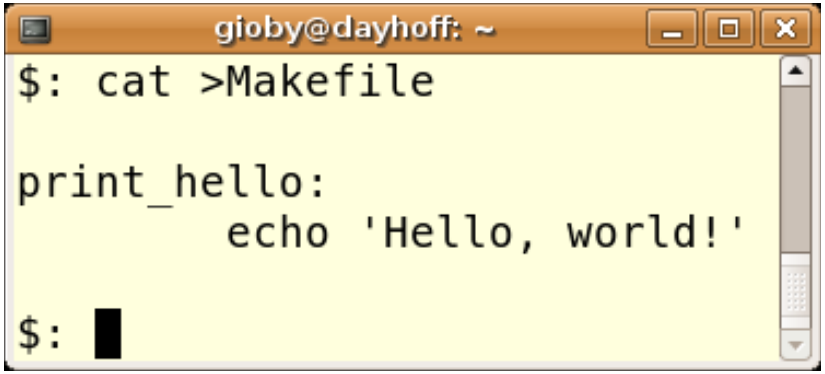
- tr→ replace a character in a text file
  1. cat b_subtilis.fasta | tr “ACGT” “acgt” > b_subtilis.tolower.fasta
- paste → merge lines of files
  1. paste b_subtilis.fasta b_subtilis.tolower.fasta
- join → join files by a common column
  1. tail hg19.gtf | cut -f 3 | sort | uniq -c | column -t > 1.txt
  2. tail hg19.gtf | cut -f 3,4 | sort -k 1 > 2.txt
  3. join -1 2 -2 1 1.txt 2.txt → joines file 1 by column 2 and file 2 by column 1

Conclusions on Unix Command line interface

- CLI enables fast manipulation of text files
- The Unix approach is more flexible and adaptable to other situations
- You don't need to use the Unix approach always, but you should be aware of it

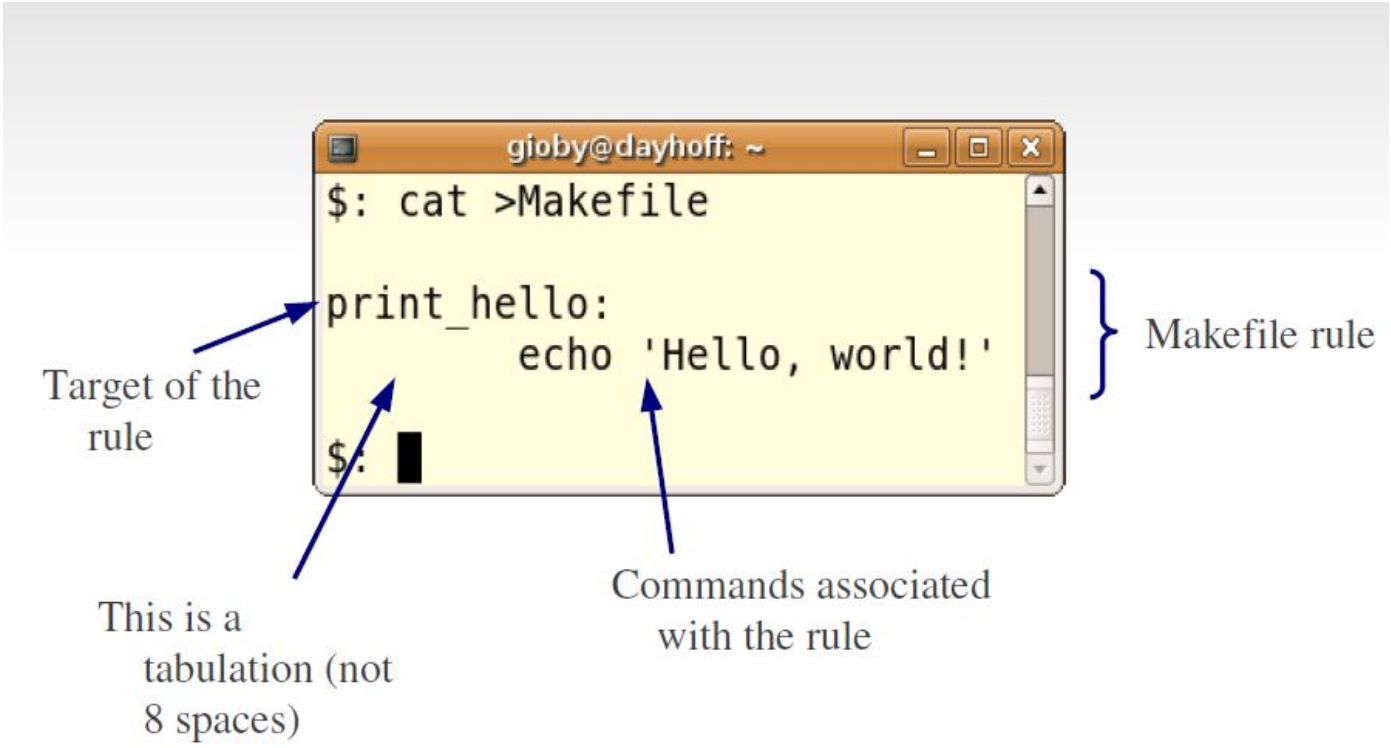
make - A build system

- make is a tool used by programmers to define how software should be compiled and installed
- It is also used as a way to store a set of commands to recall them later
- The simplest Makefile contains just the name of a task and the associated commands



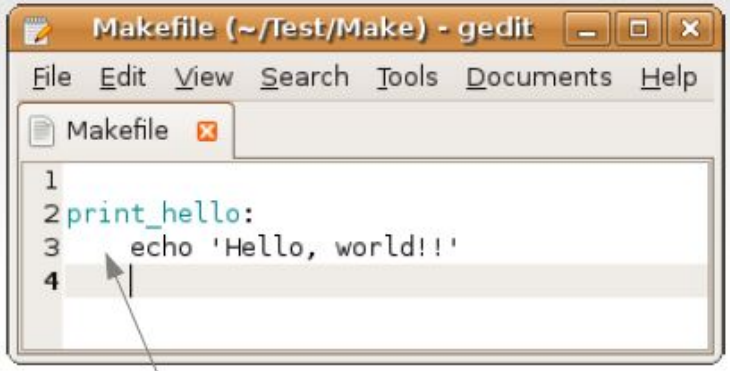
make - A build system

- print_hello is a Makefile “rule”
- Under a rule commands are stored, needed to show the message “Hello, world!”



make - A build system

- Create a file in your computer and save it as 'Makefile'.
- Write these instructions in it:




```
print_hello:
    echo 'Hello, world!!'
```

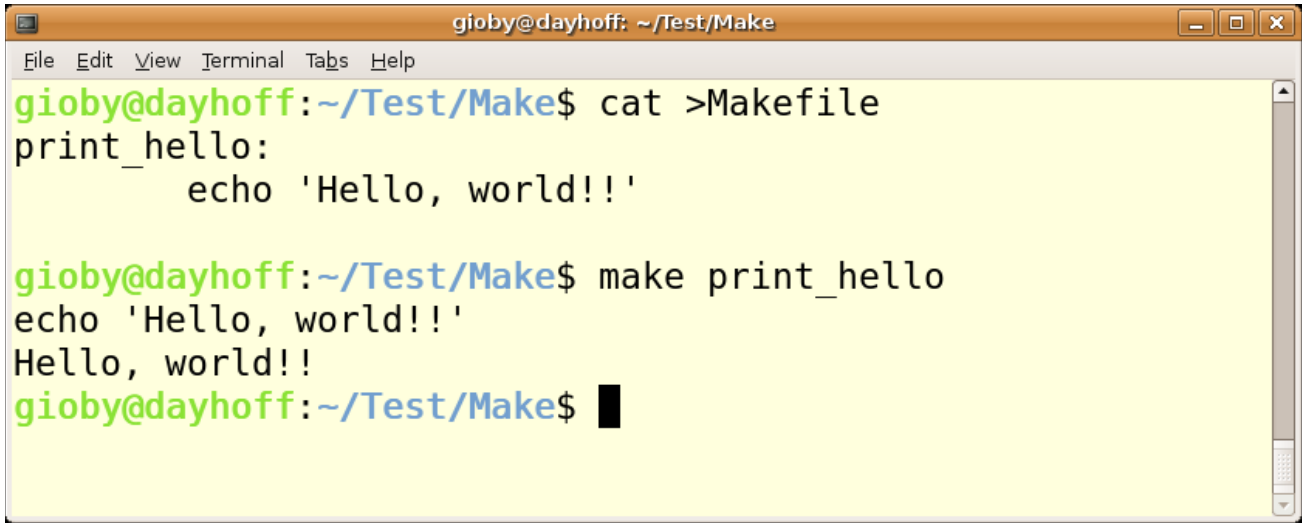
This is a tabulation (<Tab> key)

- Then, open a terminal and type:

```
make -f Makefile print_hello
```

make - A build system

- When invoked, the program make looks for a file in the current directory called “Makefile”
- When we type 'make print_hello', it executes any procedure (target) called “print_hello” in the Makefile



make - A build system

Makefile (~/Test/Make) - VIM

File Edit View Terminal Tabs Help

```
print_hello:
    echo "hello world"
    perl -e 'print "hello world\n";'

connect:
    # open an ipython shell initializing a connection to a database
    ipython -i -c 'from schema.connection import *'

get_kegg_pathways:
    # execute a script with download a list of genes from KEGG
    python get_pathways_from_kegg.py --method=soap --pathway=eco00020 --output=eco00020.genes

convert_genbank2fasta_all:
    perl convert_format.pl --inputformat=genbank --outputformat=fasta --inputfile=seq1.fasta
--outfile=seq1.genbank

exonerate_all:
    # use exonerate (an alignment tool) to align all the EST sequences to a target gene
    exonerate -q EST*.fasta -t targetgenome.fasta --model est2gene --geneticcode 3 --exhausti
ve y

convert_to_embl:
    # use seqret from emboss to convert a sequence to the embl format
    seqret myfile.seq embl::myfile2.seq
                                                                    14,112-115    Tut
```

Bash scripts

- A script is a file containing a series of instructions and variables to be interpreted as a whole
- The interpreter is optionally defined in the first line (shebang)
  1. gedit grep.sh → see “history | grep il2ra”
  2. bash grep.sh

```
#! /usr/bin/bash

echo "Start to grep"
sleep 2

variable="/home/$USER/hg19_il2.gtf"
grep -i “il2ra” $variable | cut -f 4 | sort -k 2 | wc
```

Setup Docker and Galaxy

Bioinformatics pipeline design

- Until now we used only programs shipped with the Linux Distribution
- Third party software needs to be installed first via
  - Linux Package Manager (administrator)
  - Custom Package Manager (user)
  - Compile (user, needs compiler and dependencies)
    → very tricky and error prone

Bioinformatics pipeline design

- Example fast extract subsequences from FASTA
- samtools faidx b_subtilis.fasta AL009126.3:1-10

```
kriege@voltaire:~$ head b_subtilis.fasta
>AL009126.3 Bacillus subtilis subsp. subtilis str. 168 complete genome
ATCTTTTTCGGCTTTTTTTAGTATCCACAGAGGTTATCGACAACATTTTCACATTACCAACCCCTGTGGACAAGGTTTTT
TCAACAGGTTGTCCGCTTTGTGGATAAGATTGTGACAACCATTGCAAGCTCTCGTTTATTTTGGTATTATATTTGTGTTT
TAACTCTTGATTACTAATCCTACCTTTCCTCTTTATCCACAAAGTGTGGATAAGTTGTGGATTGATTTCACACAGCTTGT
GTAGAAGGTTGTCCACAAGTTGTGAAATTTGTCGAAAAGCTATTTATCTACTATATTATATGTTTTCAACATTTAATGTG
TACGAATGGTAAGCGCCATTTGCTCTTTTTTTGTGTTCTATAACAGAGAAAGACGCCATTTTCTAAGAAAAGGAGGGACG
TGCCGGAAGATGGAAAATATATTAGACCTGTGGAACCAAGCCCTTGCTCAAATCGAAAAAAAGTTGAGCAAACCGAGTTT
TGAGACTTGGATGAAGTCAACCAAAGCCCACTCACTGCAAGGCGATACATTAACAATCACGGCTCCCAATGAATTTGCCA
GAGACTGGCTGGAGTCCAGATACTTGCATCTGATTGCAGATACTATATATGAATTAACCGGGGAAGAATTGAGCATTAAG
TTTGTCATTCCTCAAAATCAAGATGTTGAGGACTTTATGCCGAAACCGCAAGTCAAAAAAGCGGTCAAAGAAGATACATC
kriege@voltaire:~$ samtools faidx b_subtilis.fasta AL009126.3:1-10
>AL009126.3:1-10
ATCTTTTTCG
kriege@voltaire:~$
```

Bioinformatics pipeline design

- wget <samtools-URL>
- unzip samtools.zip
- Make -f Makefile

```
File  Edit  View  Search  Terminal  Help
kriege@voltaire:~$ cd samtools-1.6/
kriege@voltaire:~/samtools-1.6$ make -f Makefile
config.mk:33: ../htslib/htslib.mk: No such file or directory
config.mk:34: ../htslib/htslib_static.mk: No such file or directory
make: *** No rule to make target `../htslib/htslib_static.mk'.  Stop.
kriege@voltaire:~/samtools-1.6$
```

Bioinformatics pipeline design

- How to create and distribute a Pipeline/Software which is based on third party programs?
  - Design comprehensive setup routines for different operating systems
  - Use Anaconda/Conda
  - Use Docker and become an administrator
  - Use a fully setupped Galaxy Server
  - ...

Anaconda - Conda

- Conda Is a free community developed package manager
- Easily install many software and dependencies by simple commands as user
- Software is retrieved from the online repository “Anaconda cloud”
- Bioinformatics tools can be found in the bioconda channel

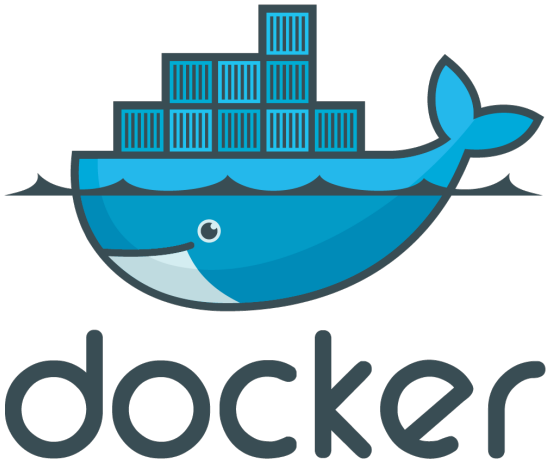
Anaconda - Conda

1. cd
2. wget <url>/Miniconda2-latest-Linux-x86_64.sh
3. bash Miniconda2-latest-Linux-x86_64.sh -b -p conda
4. source conda/bin/activate → load files and folders
5. conda install -c bioconda samtools
6. samtools faidx b_subtilis.fasta AL009126.3:1-10

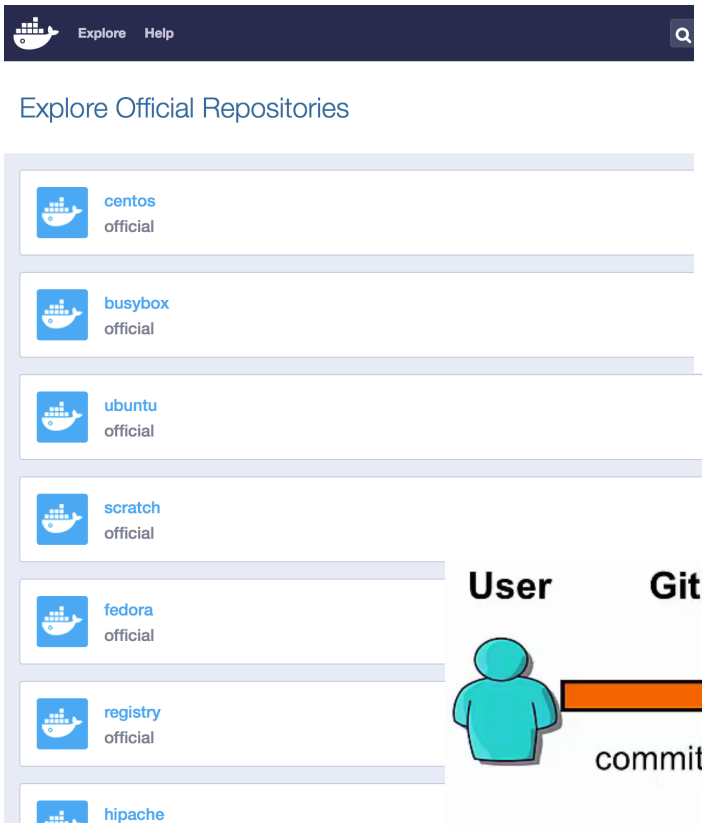
- conda install -c bioconda fastqc trimmomatic tophat htseq

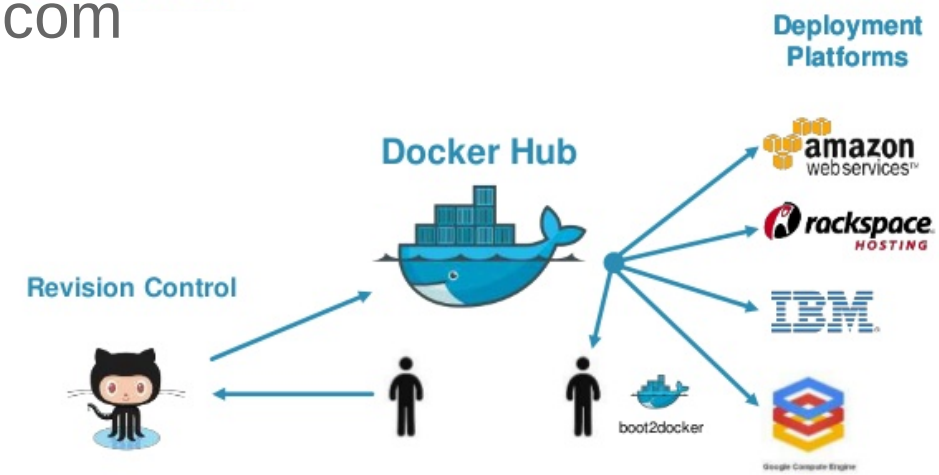
Docker

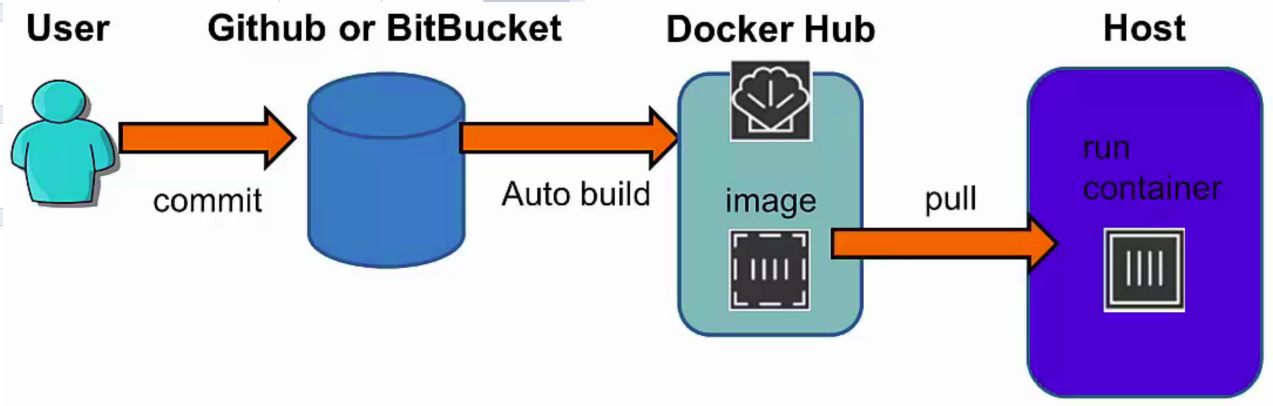
- Is a virtualization technique to setup an uniform environment in which necessary software can be installed
- The whole environment can be shipped as an “image”
- Such Docker “containers” can be executed on any operating system (Mac, Win, Linux, Unix)
  → Include own applications and run them everywhere



Docker - www.hub.docker.com

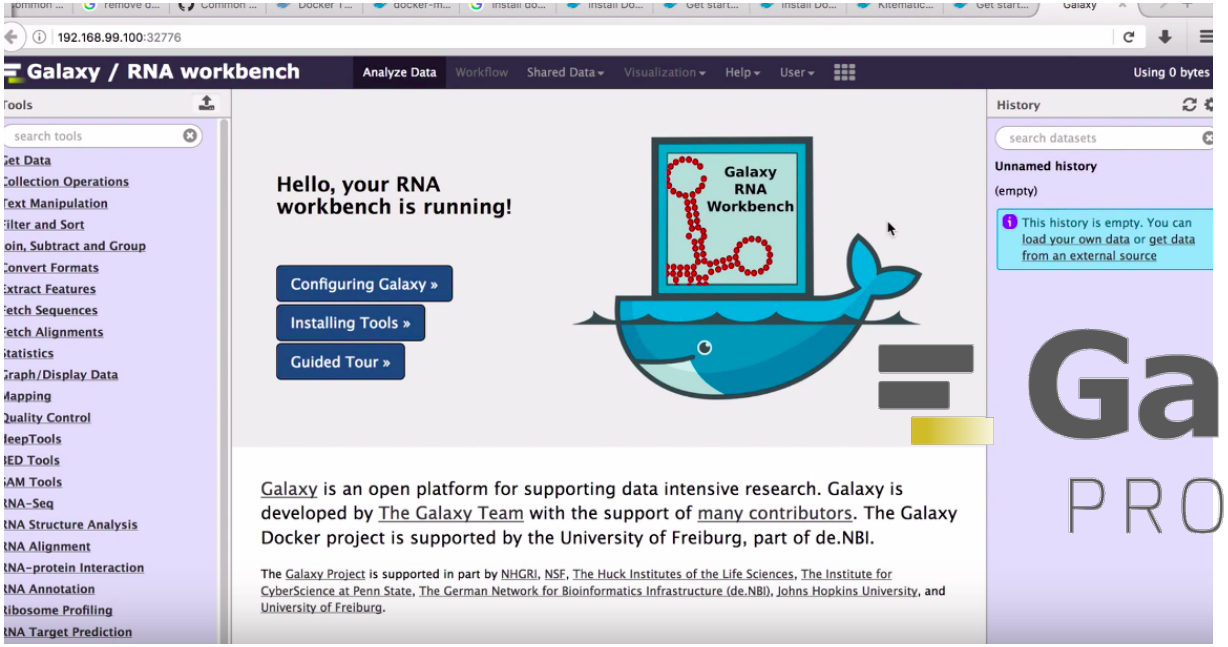






Docker

- Galaxy is web-based platform for biomedical research
- It scales from a single PC to a high performance cluster
- Lets create our own Galaxy instance!



Galaxy PROJECT

Docker & Conda → shippable Galaxy

- Workflow:
  - Use as basis a bare Linux Kernel
  - Pull e.g. Debian from Dockerhub
  - Run the container
  - Install software
  - Save (commit) Container modifications
  - Save the new image or push it back to Dockerhub

Docker & Conda → shippable Galaxy

- Check the local repository
  1. docker images
- Pull the latest Debian into local repository
  1. docker pull debian:latest
  2. docker images
- Run the Debian container interactively “-i” with a terminal “-t” and using the Bash interpreter
  1. docker run -i -t debian:latest bash

Docker & Conda → shippable Galaxy

- Install Conda
  1. apt-get update
  2. apt-get install -y curl wget bzip2
  3. wget <URL>/Miniconda2-latest-Linux-x86_64.sh
  4. bash Miniconda2-latest-Linux-x86_64.sh -b -p conda
  5. source conda/bin/activate

Docker & Conda → shippable Galaxy

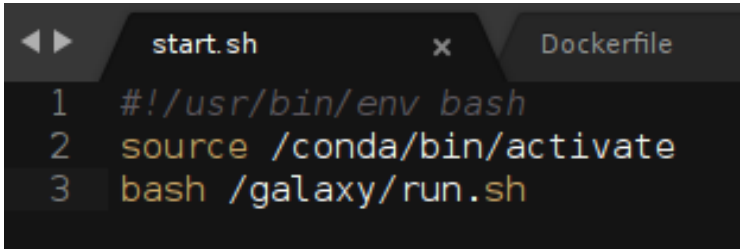
- Install tools and Galaxy using Conda
  1. conda install -c bioconda samtools
  2. conda install git
  3. git clone https://github.com/galaxyproject/galaxy.git
  4. cd galaxy
  5. git checkout v17.05 → get latest stable release
  6. bash run.sh → start the Galaxy server

Docker & Conda → shippable Galaxy

- Check running and stopped Docker processes
  1. docker ps -a → copy the running container ID

```
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
47d2466e45c5        debian:galaxy       "bash"              3 hours ago         Exited (0) 3 hours ago                          epic_snyder
b1203b184d48        debian:galaxy       "/data/start.sh"    5 hours ago         Up 5 hours                                      hungry_keller
```

- Commit recent changes as new image into repository
  1. docker stop <ID>
  2. docker commit <ID> debian-galaxy:17.05
- Create a starter script for Conda and Galaxy
  1. mkdir dockershare
  2. gedit dockershare/start.sh
  3. chmod 777 dockershare/start.sh



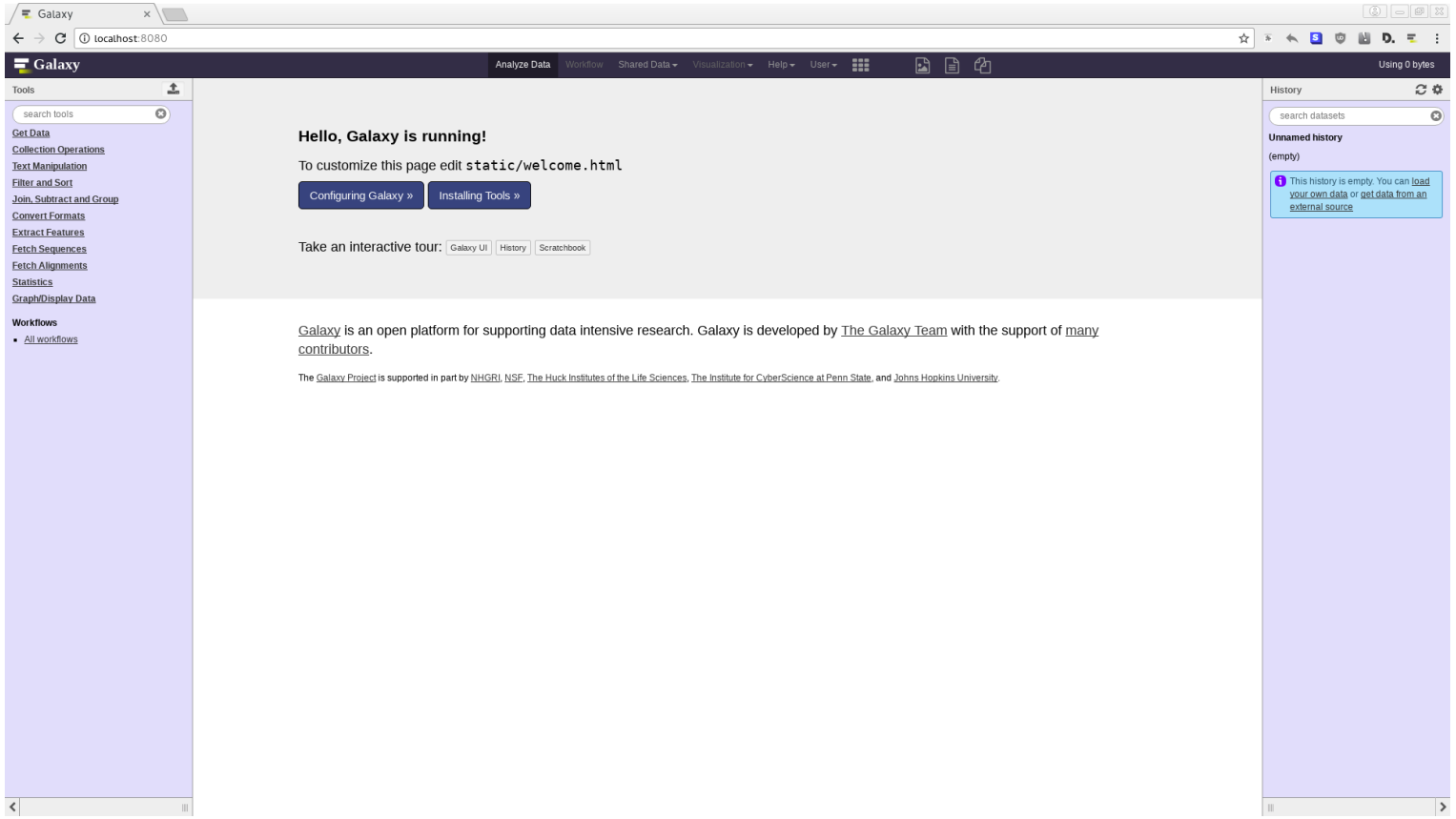
Docker & Conda → shippable Galaxy

- Run debian-galaxy:17.05 image in background / detached / daemon mode “-d”
- expose necessary IP adresses and ports “--net host”
- Share a directory with the container “-v”
- Start the server by executing the start-script

1. docker run -d --net host -v $PWD/dockershare:/data debian-galaxy:17.05 /data/start.sh

Docker & Conda → shippable Galaxy

- Open a web browser with URL “localhost:8080”
  1.firefox -new-tab “localhost:8080”



Docker & Conda → shippable Galaxy

- Work with a running container
  1. mv b_subtilis.fasta dockershare
  2. docker ps
  3. docker exec -i -t <ID> bash
  4. source conda/bin/activate
  5. samtools faidx /data/b_subtilis.fasta AL009126.3:1-10 > /data/b_subtilis.subseq.fasta
  6. exit
  7. cat dockershare/b_subtilis.subseq.fasta

Docker & Conda → shippable Galaxy

- Shortcut
  1. docker run -d -p 8080:80 -p 8021:21 -p 8022:22 bgruening/galaxy-stable

Conclusion

- Linux based OS offer many possibilities to solve a problem
- Users are able to overcome administrative restrictions
- Conda completes generic package manager
- Conda & Docker bring a basis for simple, OS independent development and distribution of bioinformatic pipelines

Pairwise Alignment

A formal alignment definition

- We assume we have two sequences u, v over the alphabet A of length m and n, respectively.

  $u = u_1 u_2 \ldots u_m$

  $v = v_1 v_2 \ldots v_n$

- Note that these sequences can be of different length.

A formal alignment definition

- First we need to model ‘biological’ or ‘chemical’ events.
- Typically we restrict ourselves to these three events
- edit operations:
  - ($u_i \rightarrow v_j$) substitution
  - ($u_i \rightarrow \varepsilon$) deletion
  - ($\varepsilon \rightarrow v_j$) insertion
- We call a substitution a match if $u_i = v_j$
- or mismatch if $u_i \neq v_j$

A formal alignment definition

- An example:

```
u = ACTCG-CCACGCG
    || || |||| ||
v = ACCCGGCCAC-CG
```

$(A \rightarrow A)(C \rightarrow C)(T \rightarrow C)(C \rightarrow C)(G \rightarrow G)(\varepsilon \rightarrow G)\ldots$

A formal alignment definition

- An example:

```
u = ACTCG-CCACGCG
    || || |||| ||
v = ACCCGGCCAC-CG
```

- This is also an alignment of u and v:

```
u = ACTCGCCACGCG
    || || |   ||
v = ACCCGGCCACCG
```

- What is the difference?

Dynamic Programming

- The DP is explained using a global alignment. It can easily be modified to a semi-global alignment.

- The unit edit distance (edist) is the number of mismatches, insertions and deletions in an optimal sequence alignment.

- Minimize the edist.
- Partial solutions are tabulated in a (m + 1) x (n + 1) matrix.

$$E(i, j) = \min \begin{cases} E(i-1, j) + 1 & \text{insertion} \\ E(i, j-1) + 1 & \text{deletion} \\ E(i-1, j-1) + (u[i] \neq v[j]) & \text{substitution} \end{cases}$$

Dynamic Programming

Initialization of the alignment matrix

	i →	T	G	A	T	A	T
j ↓	0	1	2	3	4	5	6
G	1						
C	2						
A	3						
C	4						
T	5						

← deletion E (i, 0) = j

↑ insertion E (0, j) = i

Dynamic Programming

	i →	T	G	A	T	A	T
j ↓	0	1	2	3	4	5	6
G	1	?					
C	2						
A	3						
C	4						
T	5						

Dynamic Programming

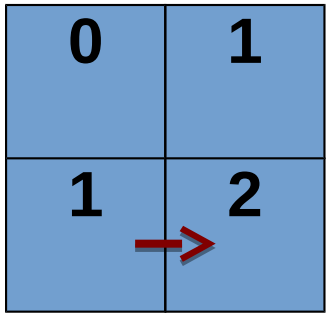
	i →	T	G	A	T	A	T
j ↓	0	1	2	3	4	5	6
G	1	?					
C	2						
A	3						
C	4						
T	5						

0	1 ↓
1	2

$$E(i,j) = \min \begin{cases} E(i-1,j) + 1 & \text{insertion} \quad 2 \\ E(i,j-1) + 1 & \text{deletion} \\ E(i-1,j-1) + (u[i] \neq v[j]) & \text{substitution} \end{cases}$$

Dynamic Programming

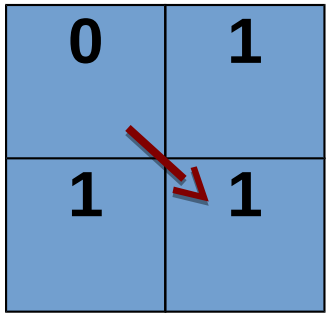
	i →	T	G	A	T	A	T
j ↓	0	1	2	3	4	5	6
G	1	?					
C	2						
A	3						
C	4						
T	5						



$$E(i,j) = \min \begin{cases} E(i-1,j) + 1 & \text{insertion} \\ \mathbf{E(i,j-1) + 1} & \mathbf{\text{deletion}} \quad \mathbf{2} \\ E(i-1,j-1) + (u[i] \neq v[j]) & \text{substitution} \end{cases}$$

Dynamic Programming

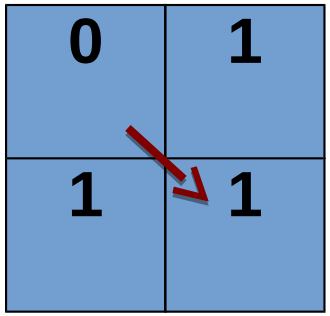
	i →	T	G	A	T	A	T
j ↓	0	1	2	3	4	5	6
G	1	?					
C	2						
A	3						
C	4						
T	5						



$$E(i,j) = \min \begin{cases} E(i-1,j) + 1 & \text{insertion} \\ E(i,j-1) + 1 & \text{deletion} \\ \mathbf{E(i-1,j-1) + (u[i] \neq v[j])} & \textbf{substitution} \quad \textbf{1} \end{cases}$$

Dynamic Programming

	i →	T	G	A	T	A	T
j ↓	0	1	2	3	4	5	6
G	1	**1**					
C	2						
A	3						
C	4						
T	5						



$$E(i,j) = \min \begin{cases} E(i-1,j) + 1 & \text{insertion} \quad 2 \\ E(i,j-1) + 1 & \text{deletion} \quad 2 \\ \mathbf{E(i-1,j-1) + (u[i] \neq v[j])} & \mathbf{\text{substitution}} \quad \mathbf{1} \end{cases}$$

Dynamic Programming

	i →	T	G	A	T	A	T
j ↓	0	1	2	3	4	5	6
G	1	1	1	2	3	4	5
C	2	2	2	2	3	4	5
A	3	3	3	2	3	4	5
C	4	4	4	3	3	4	4
T	5	4	5	4	3	4	4

$$E(i,j) = \min \begin{cases} E(i-1,j) + 1 & \text{insertion} \\ E(i,j-1) + 1 & \text{deletion} \\ E(i-1,j-1) + (u[i] \neq v[j]) & \text{substitution} \end{cases}$$

Dynamic Programming

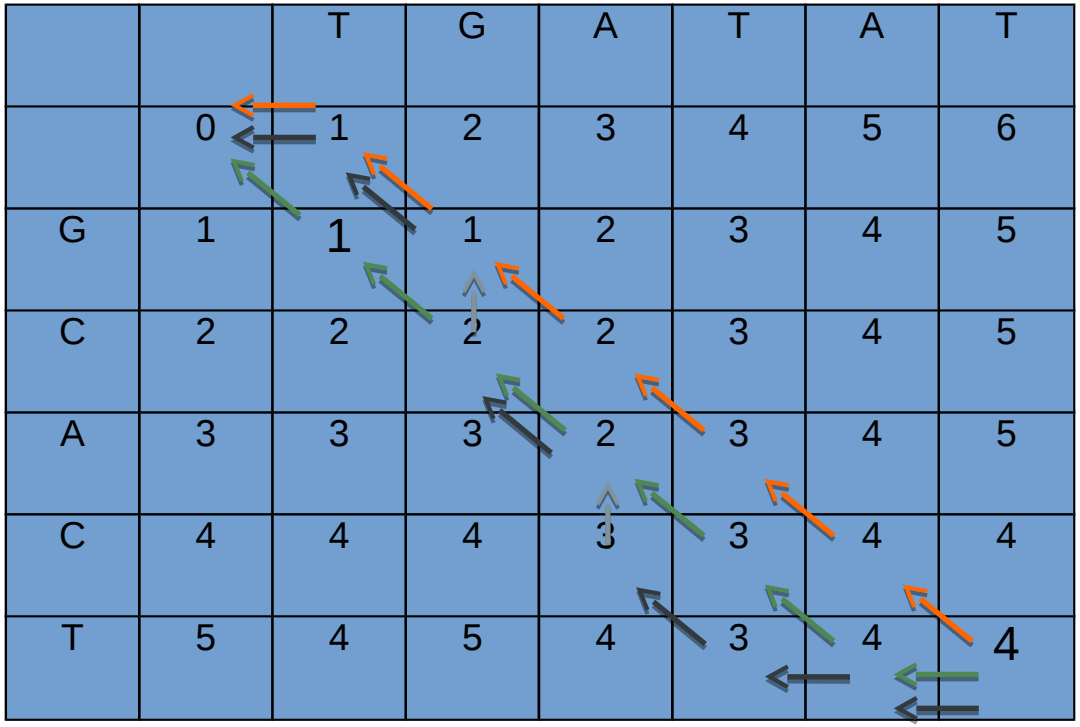
	i →	T	G	A	T	A	T
j ↓	0	1	2	3	4	5	6
G	1	1	1	2	3	4	5
C	2	2	2	2	3	4	5
A	3	3	3	2	3	4	5
C	4	4	4	3	3	4	4
T	5	4	5	4	3	4	**4**

← **edist of u and v**

$$E(i,j) = \min \begin{cases} E(i-1,j) + 1 & \text{insertion} \\ E(i,j-1) + 1 & \text{deletion} \\ E(i-1,j-1) + (u[i] \neq v[j]) & \text{substitution} \end{cases}$$

Dynamic Programming

Backtracking



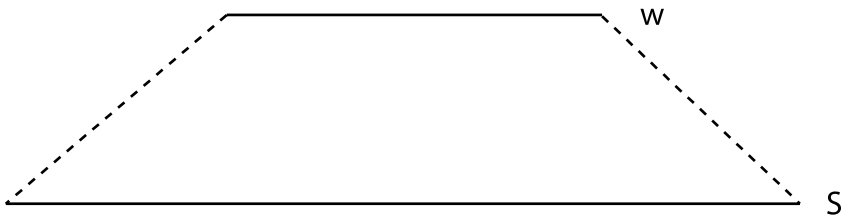

```
TGATAT
  |   |
-GCACT

TGATAT
  |   |
GCAC-T

TG-A-TAT
  | | |
-GCACT--
```

Optimal Sequence Alignment

- The previous example was a global alignment. For aligning comparably short reads to large reference genomes other alignments are more useful.

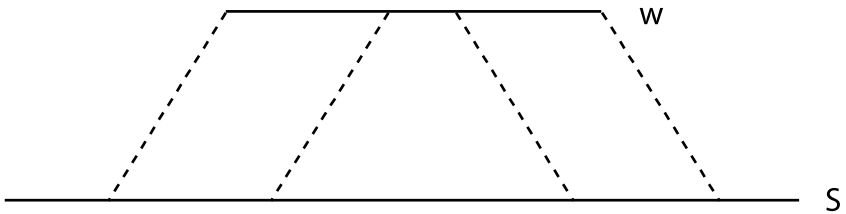
- Local alignment
  - Smith-Waterman algorithm
  - Gotoh algorithm
- Semi-global alignment
  - Modified Needleman-Wunsch algorithm
  - Myers' bit-vector algorithm (edist-bounded)

Optimal Sequence Alignment



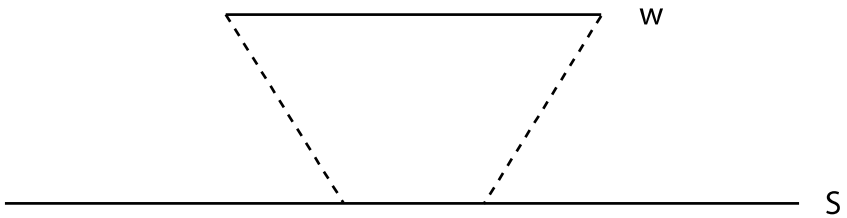
global

```
--GAAC-GTGGATTC--------------------
  ||||  |||  |   |
ACGAACAGTG-A--CACGATCACAAGGGAACGATTC
```



local

```
   GAAC-GTGGATTC
   ||||  |||
ACGAACAGTGACACGATCACAAGGGAACGATTC

                              GAACGTGGATTC
                              |||||
ACGAACAGTGACACGATCACAAGGGAACGATTC
```



semi-global

```
  GAAC-GTGGATTC
  ||||  |||  |   |
ACGAACAGTG-A--CACGATCACAAGGGAACGATTC
```

The search space

- **Complexity**
  With dynamic programming local and semi-global alignments require **O(nm)** in time

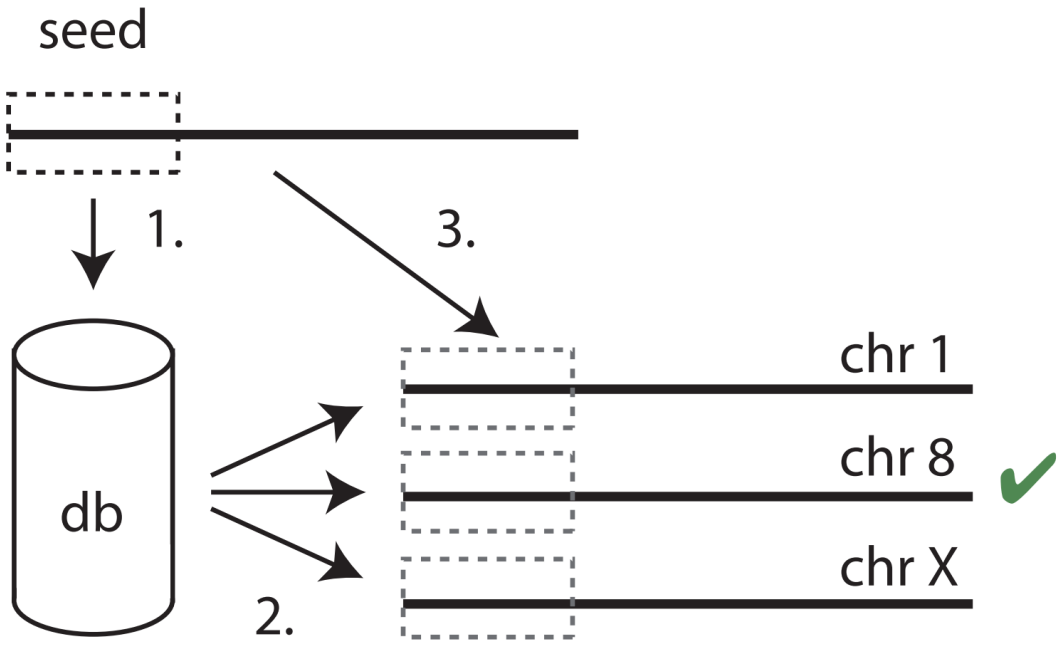
  One NGS-read vs. Human genome
  3.2G · 100 = 320, 000, 000, 000, 000 calculations!!

- **Search problem**
  Find the correct locus of origin of a 100 bp long read within a 3.0 x 10^9 bp long reference genome**.**

  If the mapping of one read takes **0.5** seconds, you would need to wait around **130 days**, until all reads are mapped (on 30 cores ~4 days).

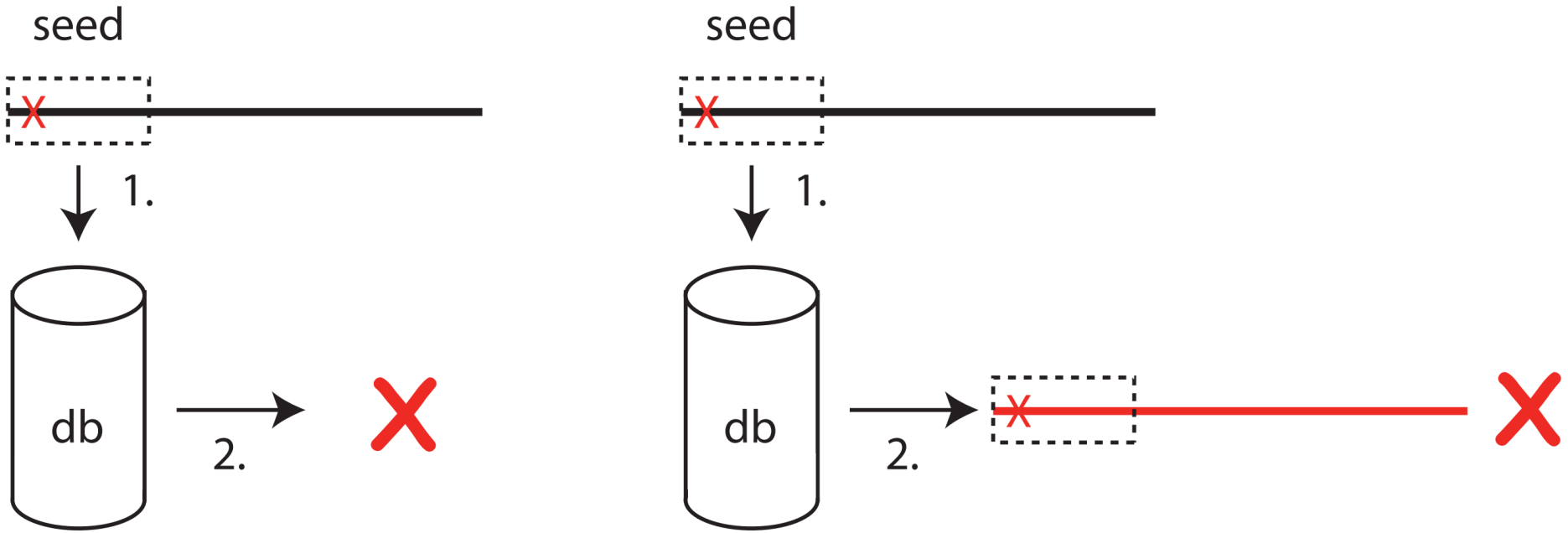
Alignment Heuristics - Seed search

- Typically, alignment heuristics aim to find the optimal alignment in large sequences in three steps:
  - Fast (near-)exact search of read substrings/seeds
  - Elongate regions with seed matches
  - Perform optimal alignments



Alignment Heuristics - Seed search

- Seed search can fail - handling of errors in seed regions is crucial
  - The seed is not in the db
  - The wrong seed (with the error) is available and the db entry leads to a wrong locus



Alignment Heuristics - Seed search

- Seed: k-mer
  - Fragment genome in unique pieces of length k
  - Store position(s) of each fragment in a hash (forward and reverse)
    - key = sequence
    - value = position(s) in the genome

k-mer	Position
GCGATGACGAGTCATC	chr8:304
CGATGACGAGTCATCA	chr3:59438;chr8:306
AGCGATGACGAGTCAT	chr10:7043
ATGACGAGTCATCATA	chr8:308;chrX:97364

Alignment Heuristics - Seed search

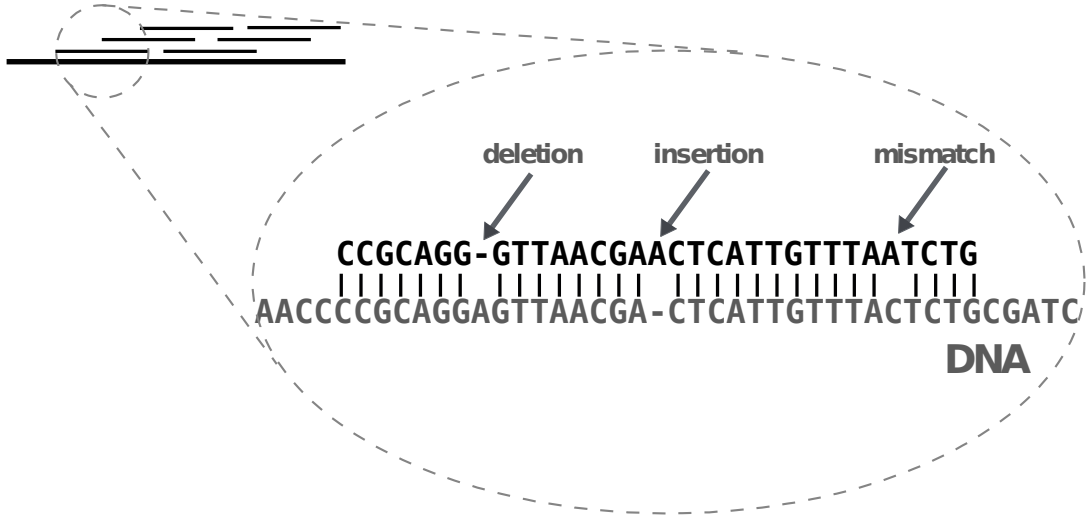
- Mapping
  - Find all seeds
  - Elongate all the seed matches on both sides
  - Compute a semi- global or local alignment
  - Discard all alignments with more than the maximum number of allowed errors
  - Compare all alignments and return these with the minimum number of errors

Alignment Heuristics - Seed search

- Problem
  - Too short seed length results in too many false positives
    - shorter seed results in more positions and longer running time
    - In most cases it will find the correct mapping position
  - Too long seed length might result in no hit at all

The reads

- The reads have errors
  - Errors from PCR amplification and sequencing machine
  - Wrong nucleotides (mismatches).
  - Missing nucleotides (deletions).
  - Inserted nucleotides (insertions).
  - Adapter sequences not correctly clipped
  - Contaminated sample



The reference genome

- The reference genome is not perfect
  - Missing region (Ns)
  - Repeats / Low complexity regions
  - SNVs
  - Genome rearrangements

- Despite enormous financial and scientific efforts to generate a high quality reference recent studies report
  - 23–29 Mb are absent from the latest build [Wang et al.]
  - 2,363 novel insertion sequences (720 loci) [Kidd et al.]
  - 5 Mb novel sequences and 104 unalignable RefSeq genes [Chen et al.]

The expectation of a sequence

The expectation of a sequence

- Assume an alphabet of 4 characters {A,C,G,T} and a text of length n

  When is a sequence uniquely occurring in the text?

The expectation of a sequence

- Assumptions
  - read of length m, reference of length n
  - both uniformly randomly composed
    → each character occurs with p = 0.25

(1) $E = p^m \cdot n$

- Solving for m

(2) $m = \log_p(E/n)$

- Setting E = 1

(3) $m = \log_p(1/n) = -\log(n)/\log(p)$

The expectation of a sequence

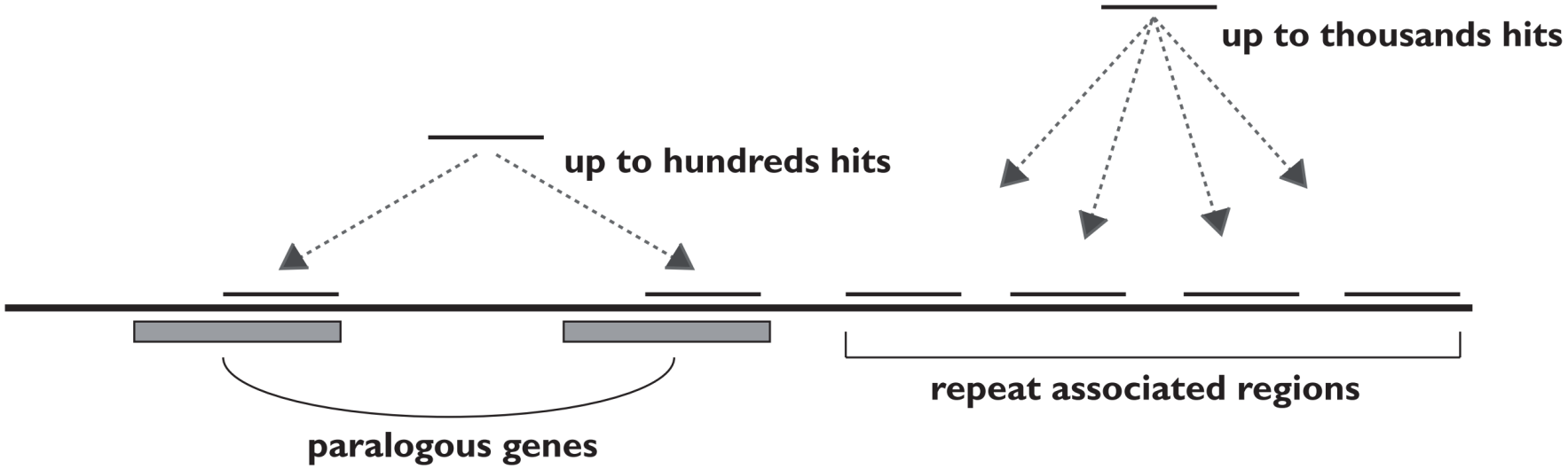
- Assumptions
  - length n is 3.2G nucleotides

(4) $-\log(3.2G)/\log(0.25) \approx 16$

The expectation of a sequence

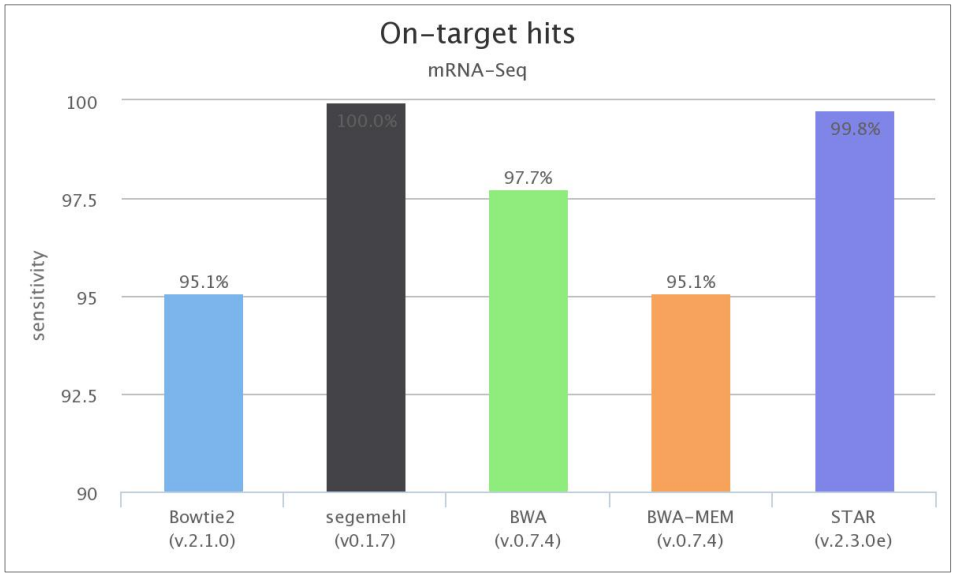
- **BUT**
  - non-randomly distributed
  - repetitive regions
  - paralogous genes
  - Transposable elements (e.g. Alu repeats)

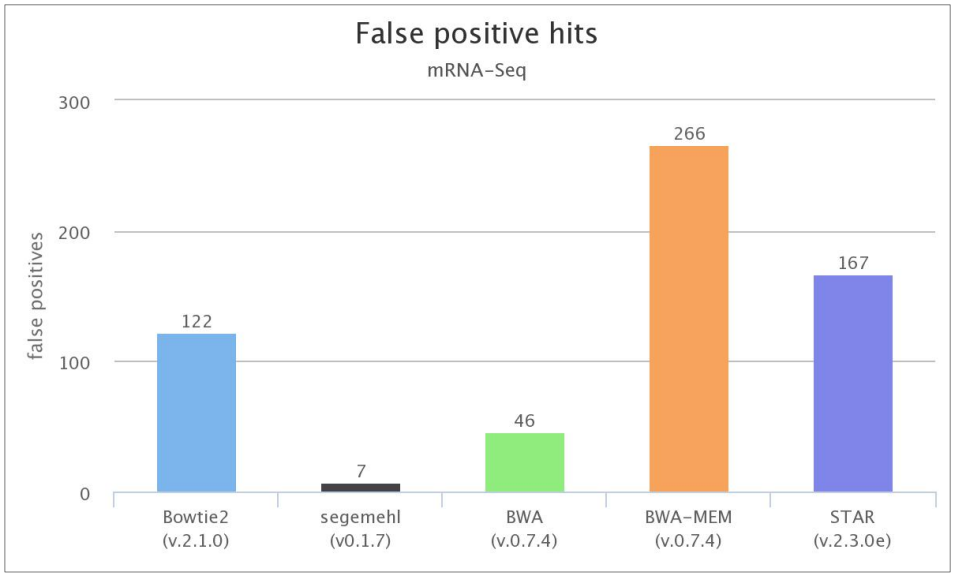
  → more than 16 nucleotides necessary to map uniquely



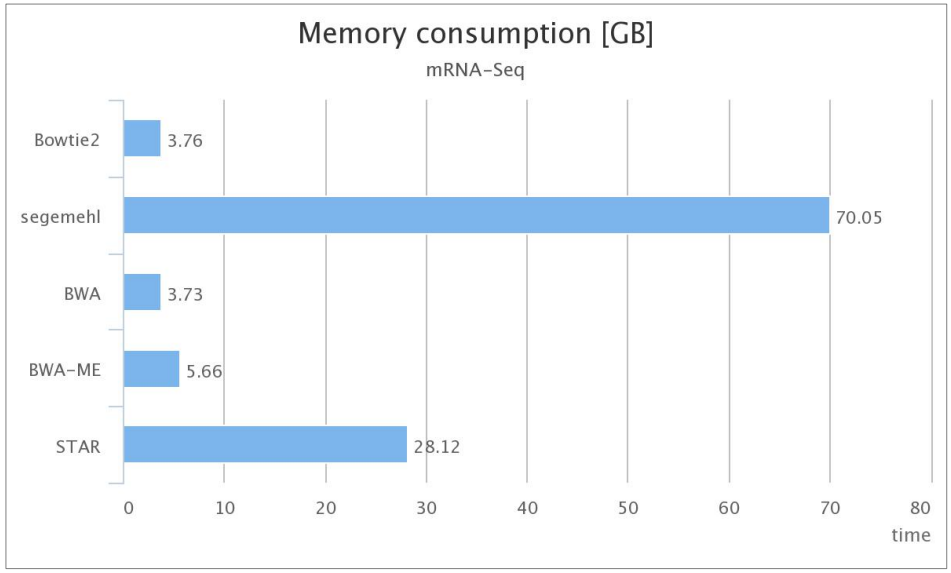
Mapping & Quantification

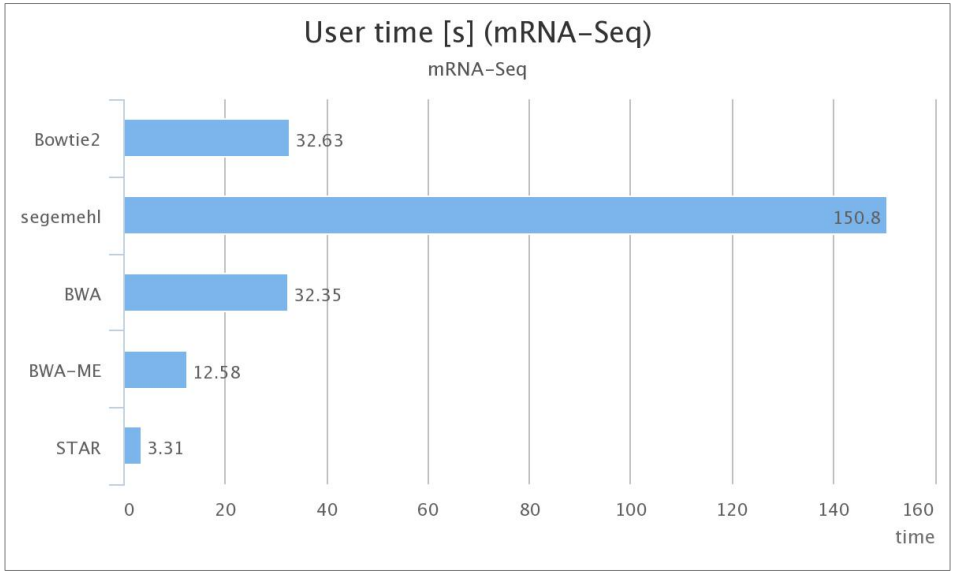
Popular Mapping Tools



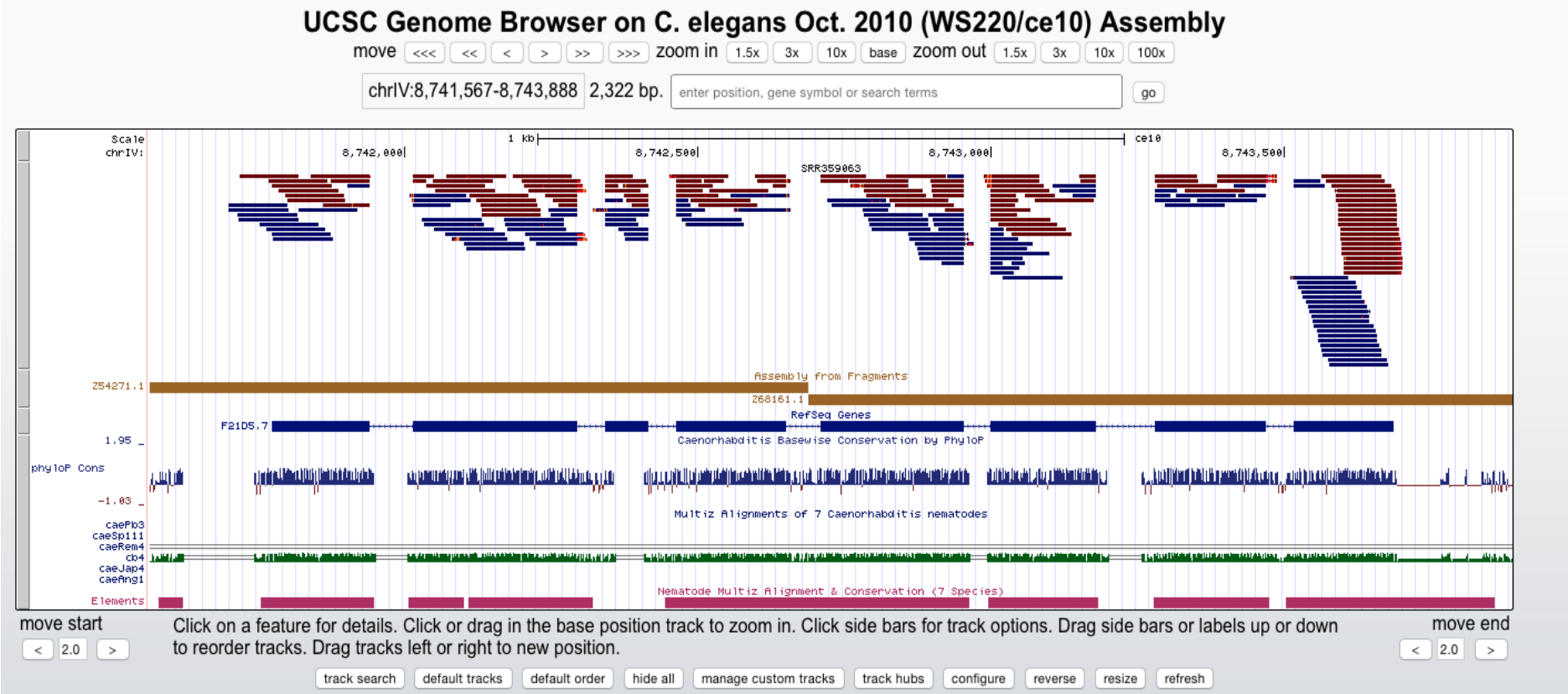


Popular Mapping Tools





Mapping Visualization - UCSC Genome Browser



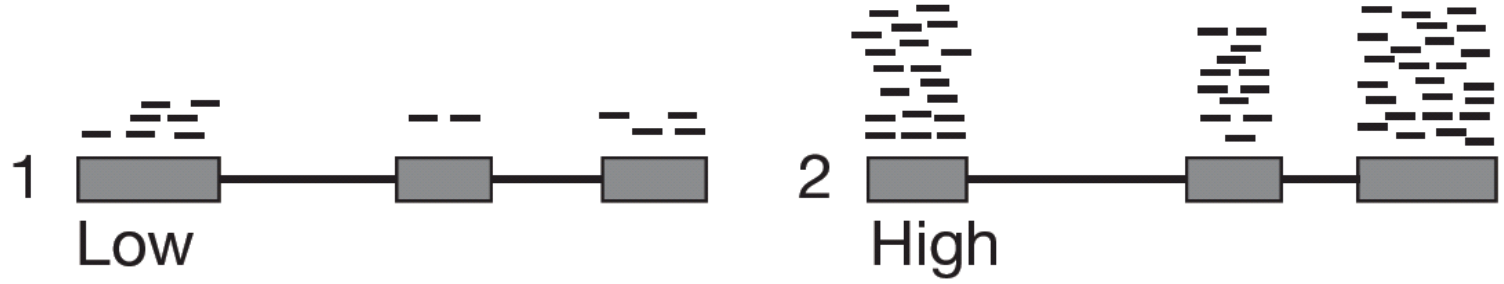
Read Quantification

- Thumb-value
  - If less than 70% of the mates are mappable, there might be a problem with the sequencing experiment
- Reads with multiple hits were counted several times
  → Count uniquely mapped reads
- What to count?
  - Fragments
  - Reads (single mates)
  - Splits of reads
- Where to count
  - Genes
  - Exons

Differential Gene Expression

- Why not use RAW counts?
  - Differences in transcript length

  - Differences in sequencing depth



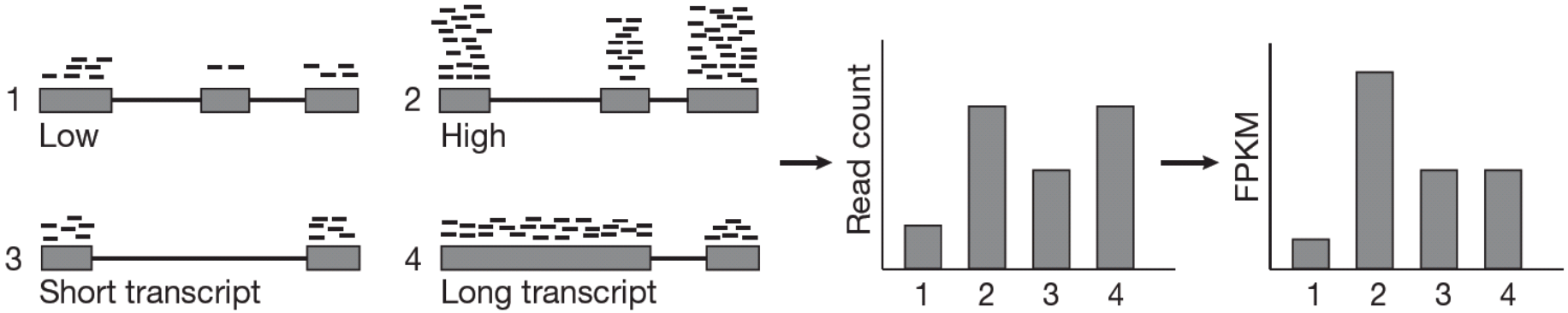
Normalization between samples

- Normalization by
  - Sample size
  - Gene length

- FPKM - Fragments Per Kilobase Million
  - Count total reads of a sample (w), divided by 1000000
  - Divide the exon read counts (x) by the “per million” scaling factor
  - Divide this values by the length of the gene (sum of exon lengths), devided by 1000 (kilobase scaling)

$$FPKM = 10^9 \times \frac{x}{wl}$$

Normalization between samples



https://www.youtube.com/watch?v=TTUrtCY2k-w

Example with scaling factor 10 instead of 100000

Gene	N 1 (35)	N 2 (45)	N 3 (106)
A (2kb)	10	12	30
B (4kb)	20	25	60
C (1kb)	5	8	15
D (10kb)	0	0	1

Gene	N1 **(3.5)**	N 2 **(4.5)**	N 3 **(10.6)**
A (2kb)	2.86	2.67	2.83
B (4kb)	5.71	5.56	5.66
C (1kb)	1.43	1.78	1.42
D (10kb)	0	0	0.09

https://www.youtube.com/watch?v=TTUrtCY2k-w

Example with scaling factor 10 instead of 100000

Gene	N1 (3.5)	N 2 (4.5)	N 3 (10.6)
A **(2kb)**	1.43	1.33	1.42
B **(4kb)**	1.43	1.39	1.42
C **(1kb)**	1.43	1.78	1.42
D **(10kb)**	0	0	0.009

https://www.youtube.com/watch?v=TTUrtCY2k-w