

A practical introduction to RNA-seq

KJ Yi, YS Ju

KAIST

0 | workshop의 목적

1. Transcriptome analysis에 대한 기본 지식을 전달
2. Analysis pipeline을 이해
3. Example data를 통한 실습

Schedule

Day 1

Introduction

Basic linux command

STAR alignment

Day 2

Quantification

RSEM

IGV

Day 3

Differential expression

Multiple test correction

DESeq2

Day 4

Pathway analysis

GSEA

Day 5

Visualization

Dimension reduction

Discussion

Transcriptome sequencing

Analysis pipeline

Basic linux commands

Will not cover:

Various sequencing methods (e.g. GRO-seq, CAGE-seq)

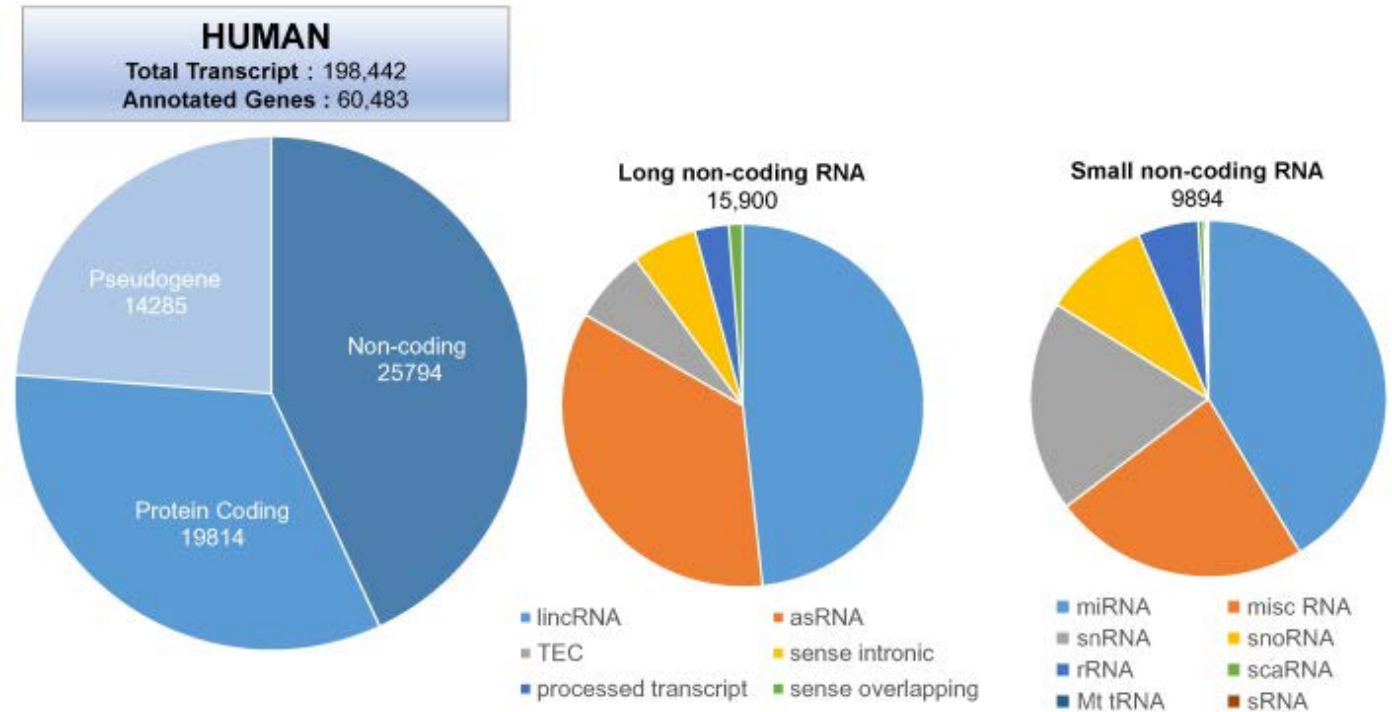
Transcriptome assembly



Human genome = 3.2×10^9 base pairs
Coding region = $\sim 2\%$
Transcribed region = $\sim 60\%$

Total discovered transcript = $\sim 200k$
Annotated = $\sim 60k$
Protein coding = $\sim 20k$

A



Kashi, Kaori, et al. "Discovery and functional analysis of lncRNAs: Methodologies to investigate an uncharacterized transcriptome." *Biochimica et Biophysica Acta (BBA)-Gene Regulatory Mechanisms* 1859.1 (2016): 3-15.

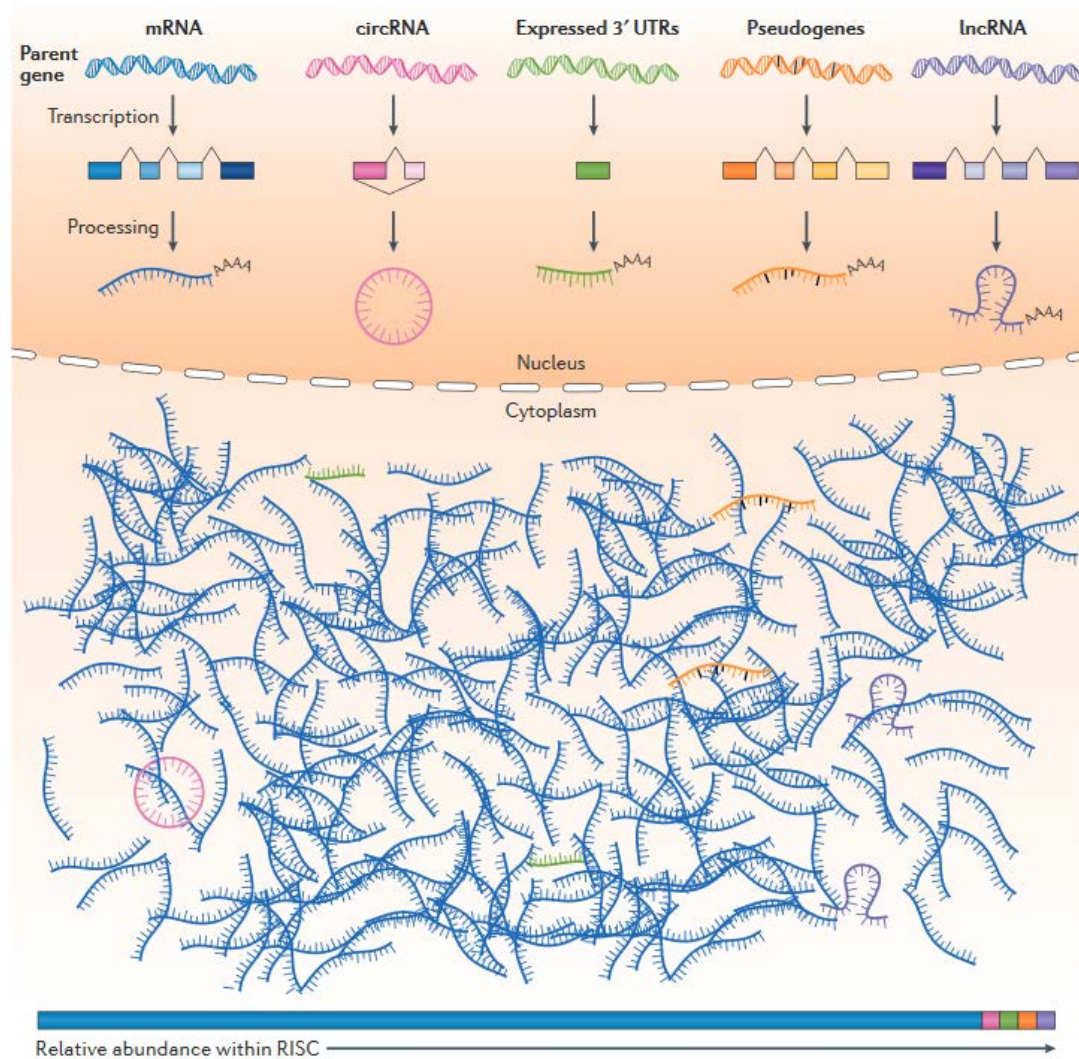


Figure 2 | **The active transcriptome available for microRNA binding competition.** The genome encodes diverse RNA classes including mRNAs, circular RNAs (circRNAs), expressed 3' untranslated regions (3' UTRs), pseudogenes and long non-coding RNAs (lncRNAs). Transcription of all RNA classes constitutes the transcriptome, which collectively can compete for miRNA targeting. The majority of active transcripts (those associated with the RNA-induced silencing complex (RISC)) available for microRNA targeting consist of coding transcripts (that is, mRNAs).

Transcription rate
Stability
Turnover rate

에 영향을 받음

Thomson, Daniel W., and Marcel E. Dinger. "Endogenous microRNA sponges: evidence and controversy." *Nature Reviews Genetics* 17.5 (2016): 272.

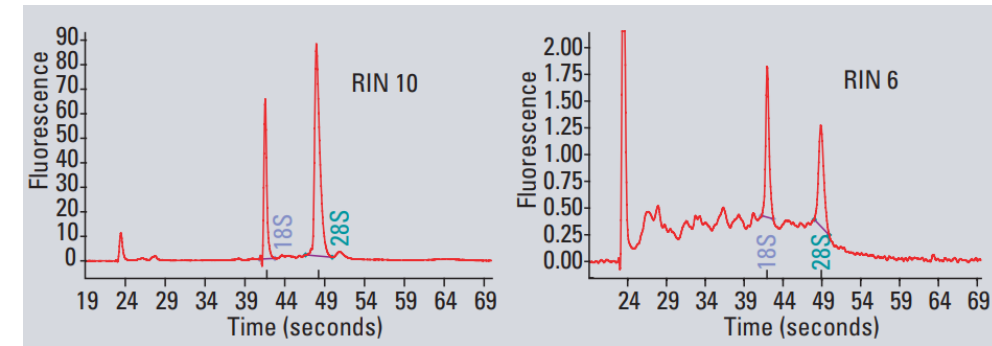
RNAs in transcriptome

- Structural RNAs (tRNA, rRNA)
- ncRNA
 - Small regulatory ncRNAs (microRNA, endogenous siRNA, piRNA)
 - snRNA, snoRNA
 - Long noncoding RNAs
 - With poly(A) tail (mRNA-like)
 - Circular RNA
 - lncRNA with triple-helix end
 - eRNA
 - Repeat-associated RNAs

- ~10–30 pg total RNA in a mammalian cell (mostly rRNA, tRNA)
- ~300,000 mRNA (1~5% of total RNA)
- ~12,000 different transcripts, average length of 2kb
- Rare mRNA ~10copies / abundant mRNA $\sim 10^4$ ~

Checklist

- Sample
- RNA isolation, RNA integrity
 - RIN (RNA integrity number 1-10) > 6~7
 - Quantity > 1ug
- Library prep
 - rRNA-depletion vs. Poly-A tail capture
 - Size selection
 - Strand specificity, paired-end, long reads ...
- Quantitative standards
 - ERCC spike-in
 - Barcodes / Unique molecular Identifier

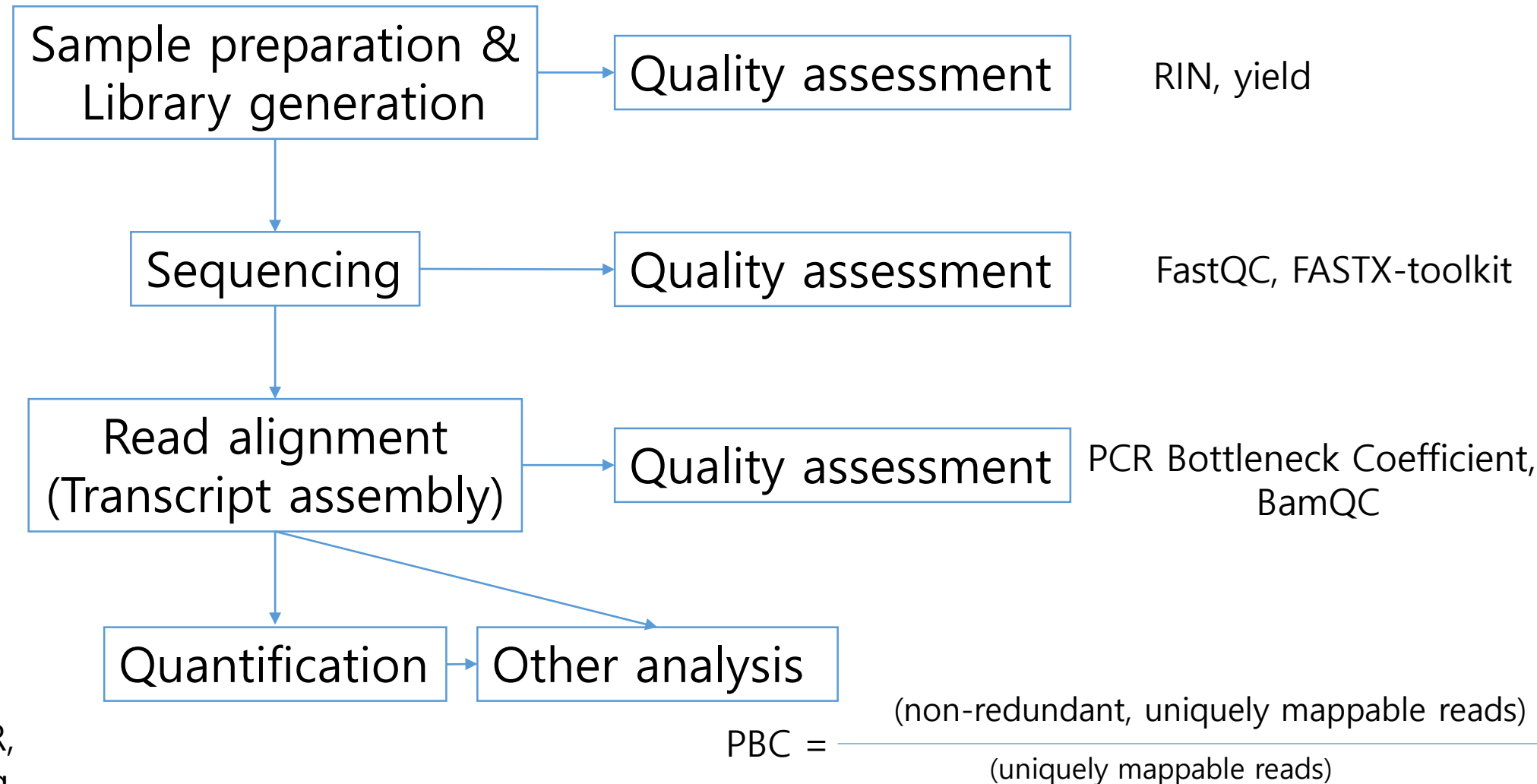




Checklist (2)

- Sequencing platform
 - Illumina platform (~150b, paired-end, 50gb/lane in HiSeq-4000)
 - PacBio SMRT (~10kb, 100,000~ reads, not proper for quantification)
- Depth
 - >4Gb, ~20,000,000 reads for quantification
 - Depend on library quality, library complexity, sample ...

Pipeline



Alignment tools

BWA, Bowtie, STAR, Tophat, GSNAP, MapSplice ...

Assembly tools

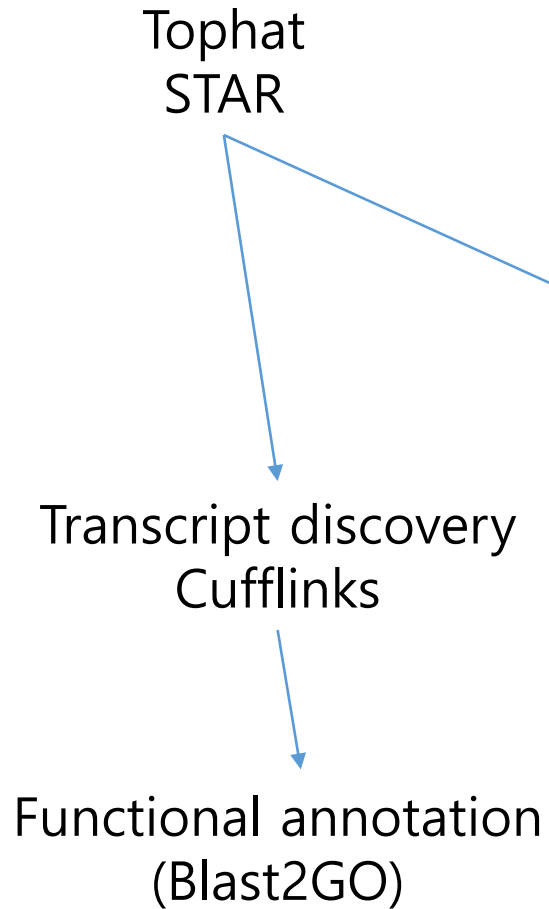
Cufflinks, StringTie, Scripture, Trinity, Velvet, CAFÉ, TACO

Quantification tools

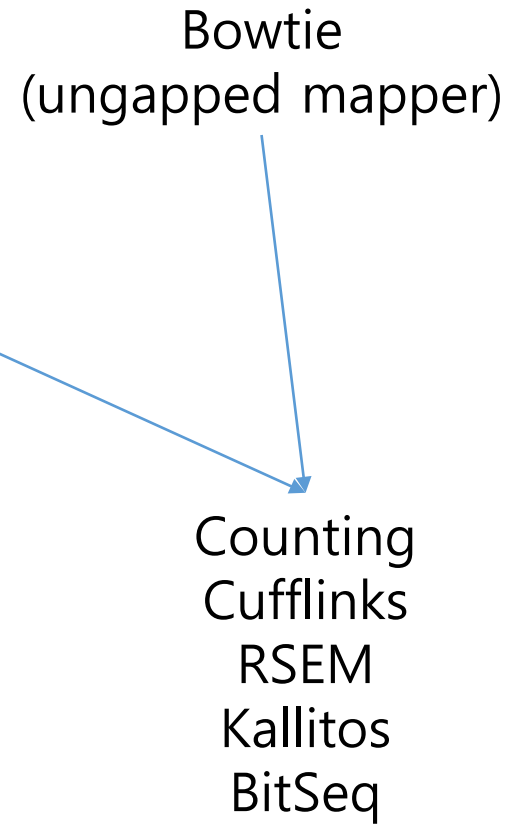
Cuffdiff, DESSeq, edgeR, DEGseq, baySeq, BitSeq, ...

~.5: severe, .5-.8: moderate, .8-.9: mild, .9-1: no bottlenecking

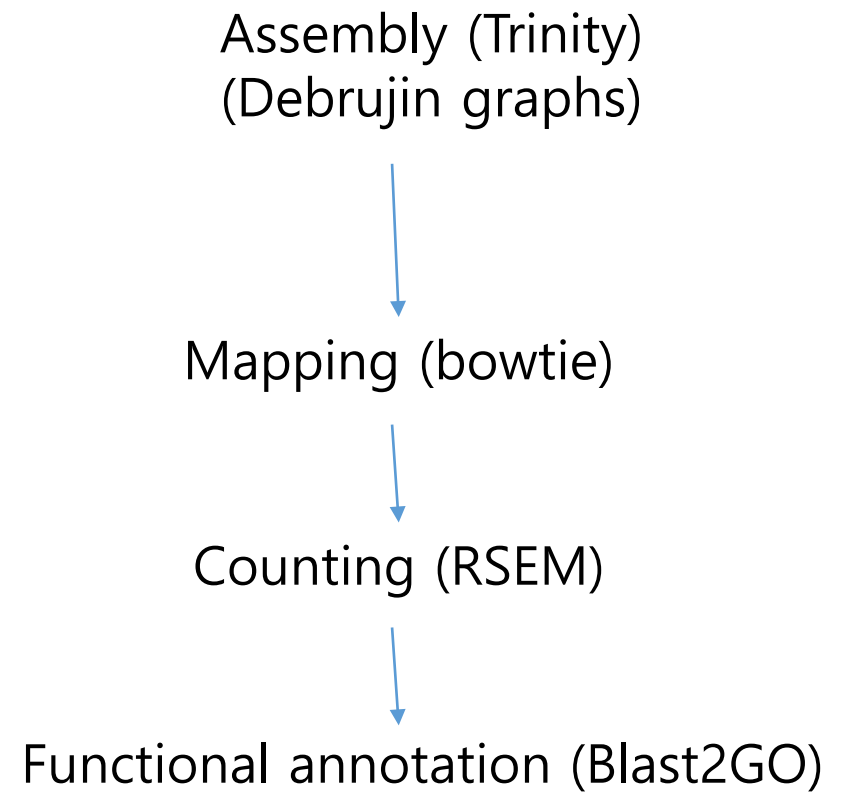
Genome mapping



Transcriptome mapping



Reference-free assembly



FASTQ File format

```
@SEQ_ID
```

```
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT  
+
```

```
! ' ' * ( ( ( * * * + ) ) % % % + + ) ( % % % % ) . 1 * * * - + * ' ' ) ) * * 5 5 C C F > > > > > C C C C C C C C 6 5
```

```
@SEQ_ID_2
```

```
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT  
+
```

```
! ' ' * ( ( ( * * * + ) ) % % % + + ) ( % % % % ) . 1 * * * - + * ' ' ) ) * * 5 5 C C F > > > > > C C C C C C C C 6 5
```

Phread quality score

$$P = 10^{-Q/10}$$

$$Q = -10 \log_{10}(P)$$

```

SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS.....
.....XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.....
.....IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII.....
.....JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ.....
..LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL.....
!"#$%&'()*+,-./0123456789;<=>?@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)

https://en.wikipedia.org/wiki/FASTQ_format

https://en.wikipedia.org/wiki/Phred_quality_score

Further reading

Strand specific library

Levin, Joshua Z., et al. "Comprehensive comparative analysis of strand-specific RNA sequencing methods." *Nature methods* 7.9 (2010): 709.

polyA selection vs rRNA depletion

Zhao, Shanrong, et al. "Evaluation of two main RNA-seq approaches for gene quantification in clinical RNA sequencing: polyA+ selection versus rRNA depletion." *Scientific reports* 8.1 (2018): 4781.

Other transcriptome sequencing methods

<https://www.illumina.com/science/sequencing-method-explorer/kits-and-arrays/cage-seq.html>

<https://www.illumina.com/science/sequencing-method-explorer/kits-and-arrays.html>

Pipeline resources

TCGA

https://docs.gdc.cancer.gov/Data/Bioinformatics_Pipelines/Expression_mRNA_Pipeline/

GTEx Consortium

<https://github.com/broadinstitute/gtex-pipeline/tree/master/rnaseq>

ENCODE project

<https://www.encodeproject.org/rna-seq/>

Transcriptome sequencing

Analysis pipeline

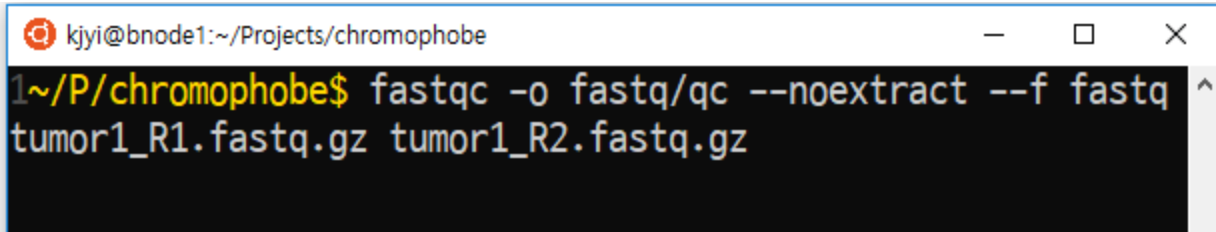
Basic linux commands


```
kjyi@DESKTOP-F50HDIL: ~  
kjyi@DESKTOP-F50HDIL:~$
```

Basic linux command = Bash

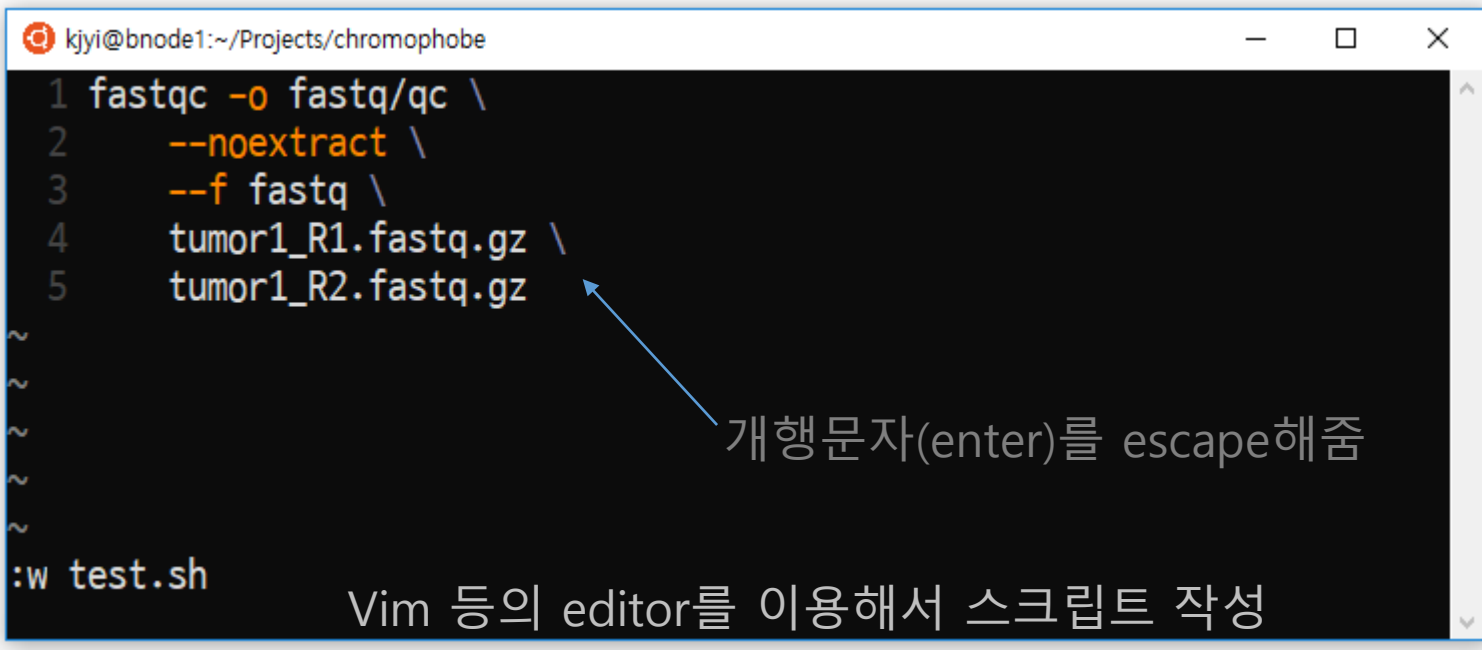
- 왜 Bash를 배워야 하는가
 - 셸 스크립트의 동작 원리를 이해하는 것은 분석에 필요한 tool들을 구동하는데 필수적임
 - 대부분의 tool들이 shell (bash)에서 분석을 구현하도록 함
 - 예외: (Bioconductor) R packages, python modules
 - 간단하고 빠른 prototype 코드의 구현
- Bash로 해결할 수 없는 경우
 - 속도가 중요한 작업을 할 때
 - 복잡한 산술 연산 작업들
 - 플랫폼간 이식성이 필요할 때 (리눅스 배포판 간 호환성이 안 좋음)

- BASH: 터미널에서 입력하는 것이 곧 script



```
kjyi@bnode1:~/Projects/chromophobe
1~/P/chromophobe$ fastqc -o fastq/qc --noextract --f fastq
tumor1_R1.fastq.gz tumor1_R2.fastq.gz
```

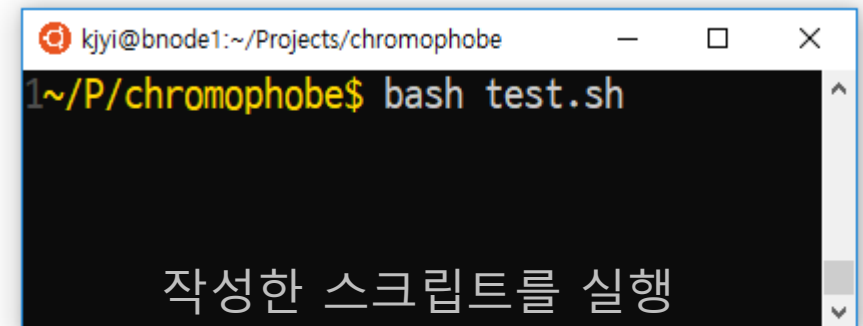
터미널에 직접 입력



```
kjyi@bnode1:~/Projects/chromophobe
1 fastqc -o fastq/qc \
2   --noextract \
3   --f fastq \
4   tumor1_R1.fastq.gz \
5   tumor1_R2.fastq.gz
~
~
~
~
:w test.sh
```

개행문자(enter)를 escape해줌

Vim 등의 editor를 이용해서 스크립트 작성



```
kjyi@bnode1:~/Projects/chromophobe
1~/P/chromophobe$ bash test.sh
```

작성한 스크립트를 실행

유념해야 할 것

- 절대경로/상대경로/현재위치
- PATH (명령어를 탐색할 위치들)
- 대소문자를 구분할 것

주요 커맨드

- cd, pwd, ls, mv, cp, mkdir
- cat (파일 내용을 출력), less (pager), vim (editor)
- Soft link (윈도우의 바로가기와 유사)
 - ln -s 상대경로 링크파일

change directory
print working directory
list
move (rename)
copy
make directory
concatenate
echo

효율적인 script의 첫걸음: redirect(>)와 pipe(|)

```
$ zcat ./log.gz | grep "error" | tail -n 20 > ./last_errors
```

현재 directory의 log.gz라는 파일의 압축을 풀어 출력해서

이중 error라는 단어가 포함된 줄만 골라서

마지막 20줄만

현재 directory의 last_errors라는 파일에 써라

- | 는 앞 명령의 standard output을 뒤 명령어의 standard input으로 넘겨줌
- > 는 앞 명령의 standard output을 파일로 저장함

오래 걸리는 작업을 수행할 때

- `$ bash long_script.sh`
- 작업 도중 Ctrl-C (windows user) - 취소됨
- 터미널을 종료함 - 취소됨
- 노트북의 네트워크를 종료함 - 취소됨

`$ bash long_script.sh &` 명령을 계속 입력할 수 있지만, shell이 종료되면 함께 종료됨

`$ nohup bash long_script.sh 1> stdout.txt 2> stderr.txt &`

`$ qsub -q day long_script.sh` (job scheduler)

- PBS/Torque
- LSF / SGE / Slurm
- Spark/Hadoop (고성능 parallel computing)

Command separators

아무것도 안쓰면 (엔터): 오류가 나든 안 나든 다음 명령어가 실행됨

;
아무것도 안 쓴 것과 동일. 여러 줄의 command를 한 줄에 쓸 때 이용됨

&
앞 명령어가 background로 시작되고, 뒤 명령어가 fg로 실행됨

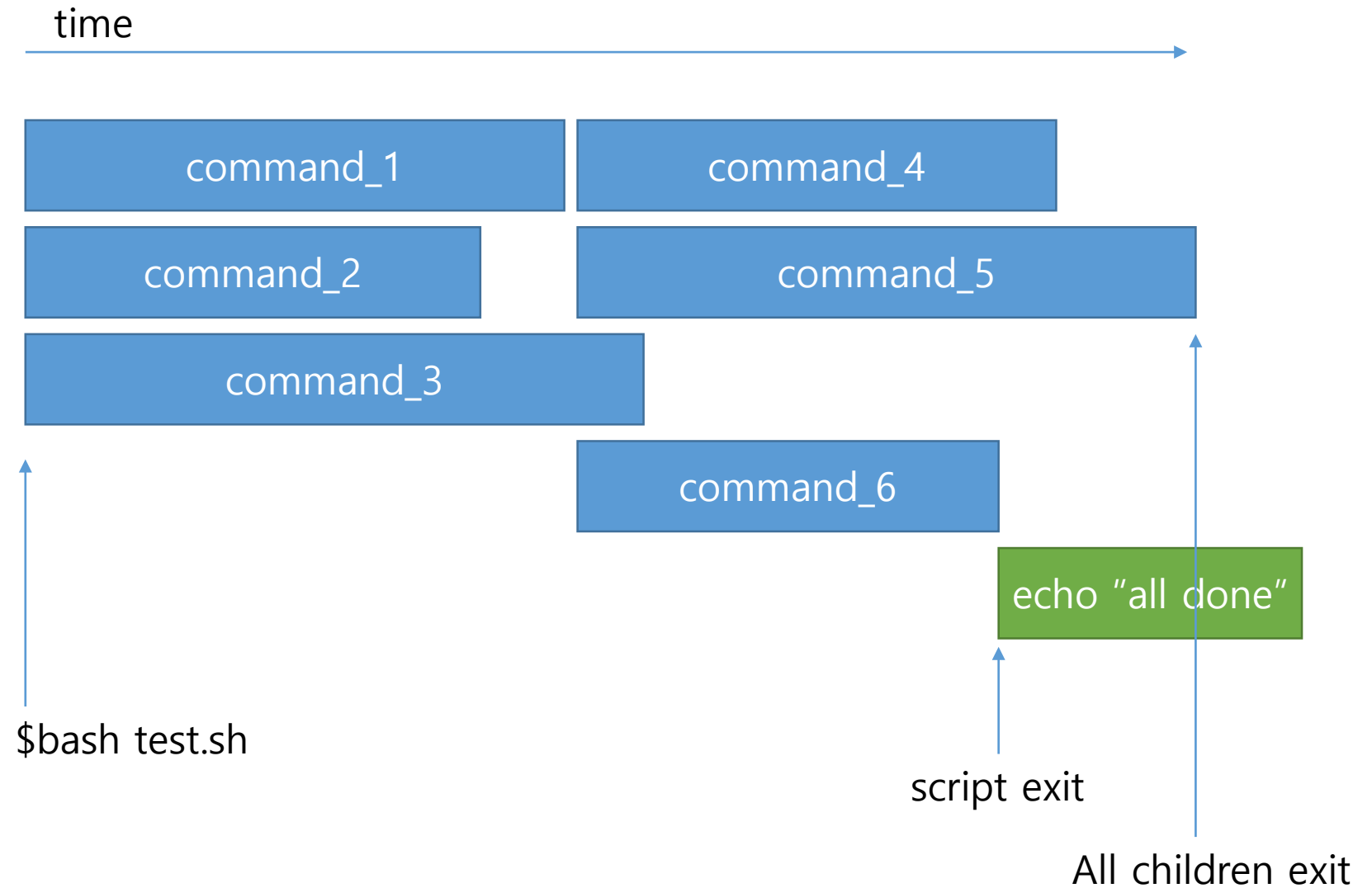
&&
앞 명령어가 정상 종료 (exit status zero)여야 뒤 명령어가 실행됨

||
앞 명령어가 오류상태(non-zero status)로 끝나야 뒤 명령어가 실행됨

```
$ cat test.sh
command_1 &
command_2 &
command_3
command_4 &
command_5 &
command_6
echo "all done"
```

```
$ bash test.sh
all done
```

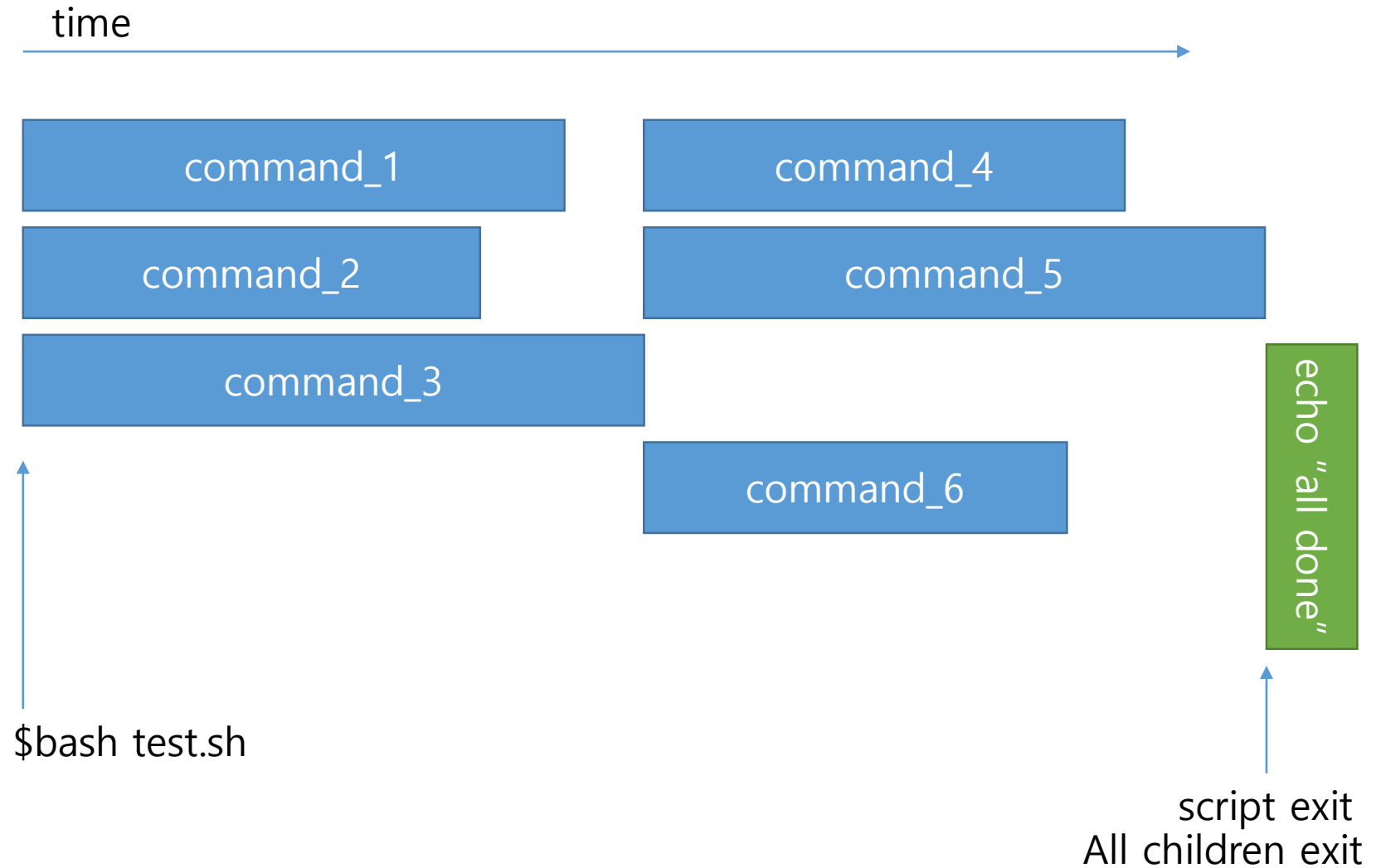
```
$ _
```




```
$ cat test.sh
command_1 &
command_2 &
command_3
wait
command_4 &
command_5 &
command_6
wait
echo "all done"

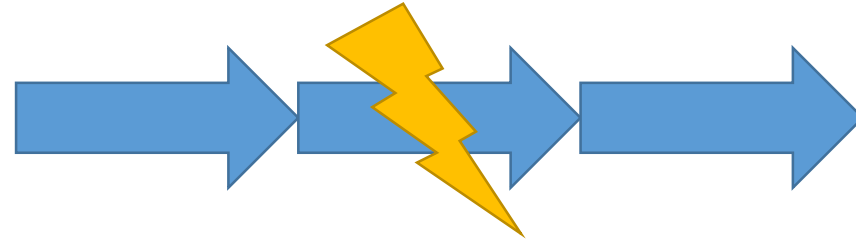
$ bash test.sh
all done

$ _
```

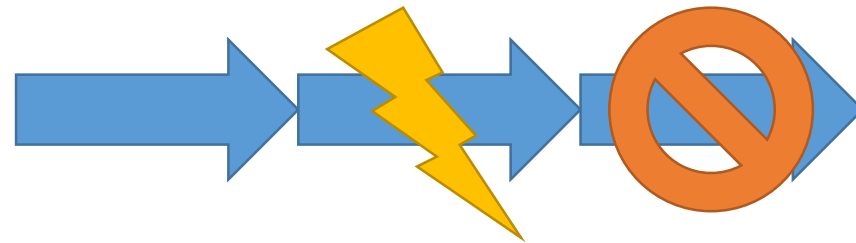


기본 SHELL SETTING에서는 에러가 나도 다음 STEP이 수행됨

```
$ cat test.sh  
command_1 &&  
command_2 &&  
command_3
```



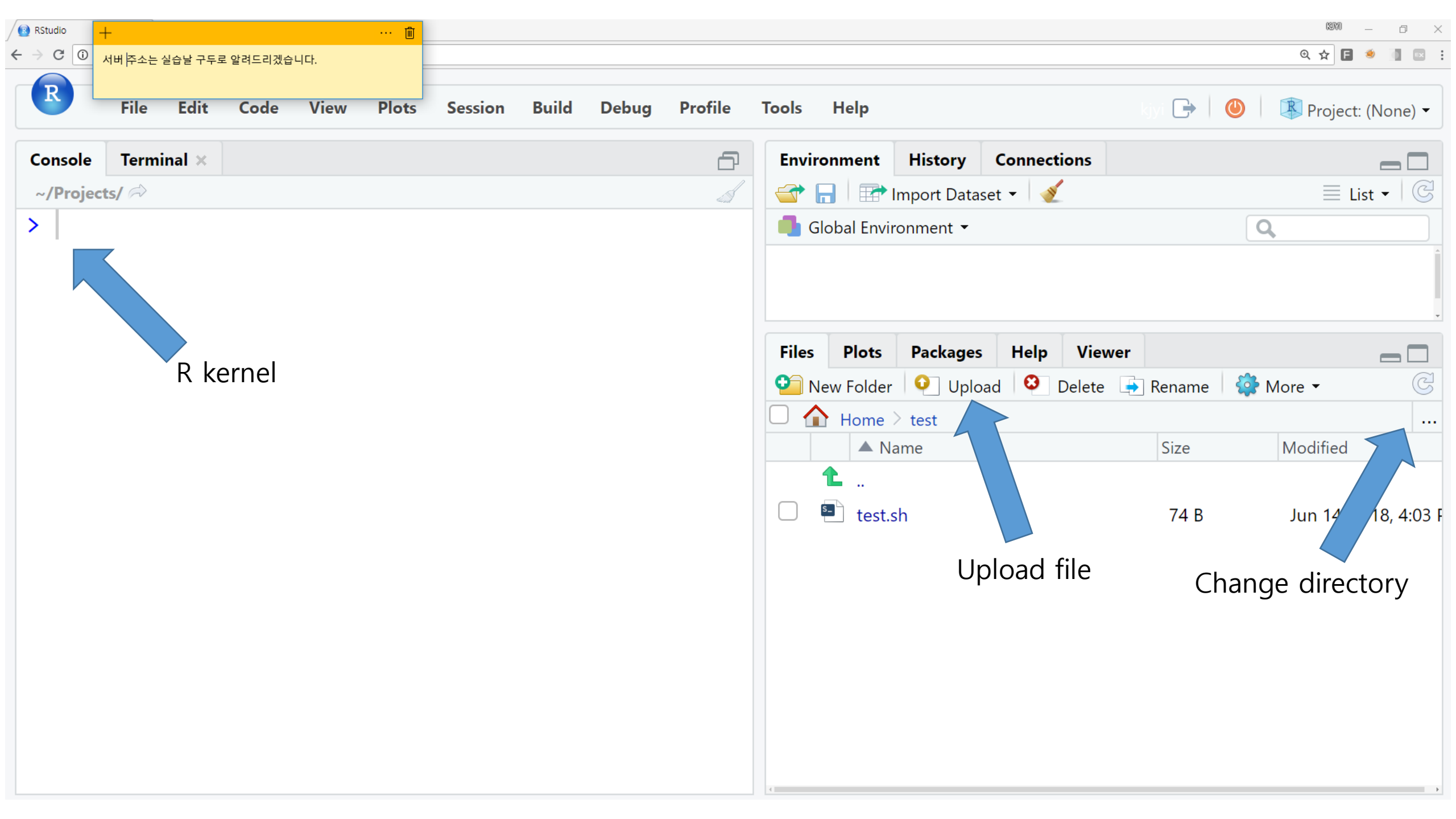
```
$ cat test2.sh  
command_1 || exit 1  
command_2 || exit 1  
command_3 || exit 1
```



에러 발생시 다음 STEP을 시행하지 않으려면
이처럼 코딩 해야 함

```
$ cat test3.sh  
set -e  
command_1  
command_2  
command_3
```

- ssh [your id@](#)000.000.000.000 -p 0000
(패스워드 입력 시 화면에 출력되지 않는 것은 정상)
- Mac and linux users: use built-in terminal
- Windows users: ①putty ②mobaXterm ③openSSH ④WSL
- In this tutorial, use Rstudio-server in our cluster!



서버 주소는 실습날 구두로 알려드리겠습니다.

R kernel

Upload file

Change directory

RStudio

+

서버 주소는 실습날 구두로 알려드리겠습니다.

R

FileEditCodeViewPlotsSessionBuildDebugProfileToolsHelp

kjy1

Project: (None)

test.sh

RunRun Script

```
1 for i in apple banana;
2   do echo $i
3 done
4
```

2:11Shell

ConsoleTerminal

Terminal 1~/test

```
[bnode1] ~/test$ sh test.sh
apple
banana
[bnode1] ~/test$
```

EnvironmentHistoryConnections

Global Environment

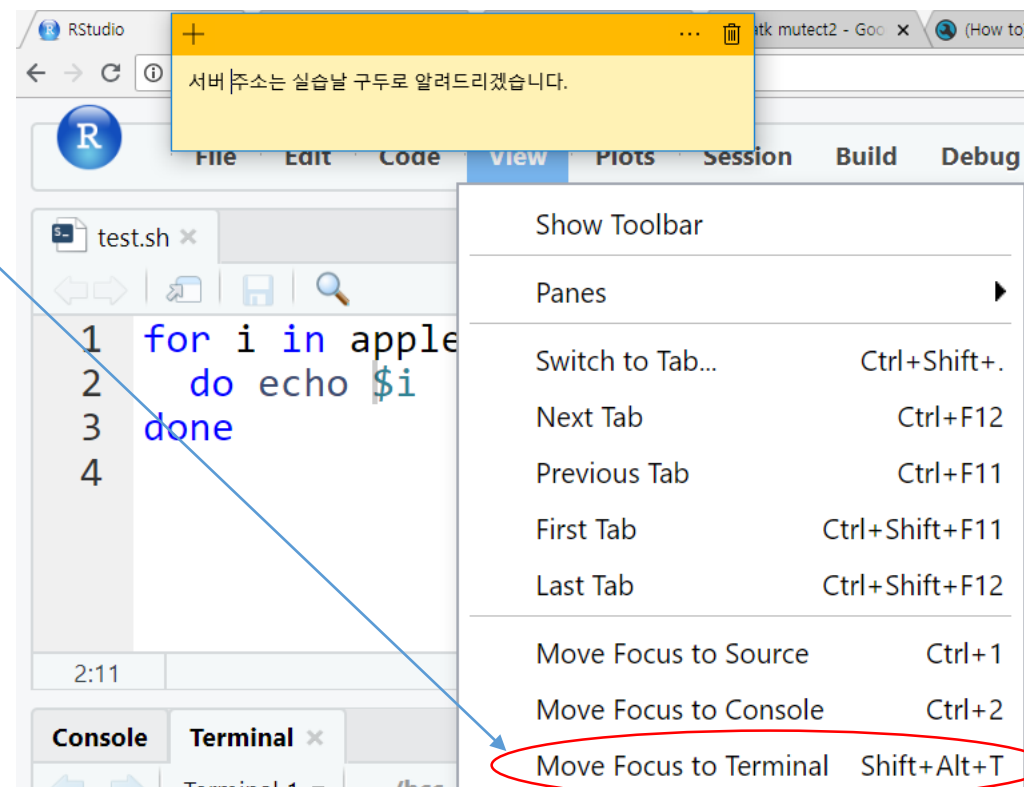
FilesPlotsPackagesHelpViewer

New FolderUploadDeleteRenameMore

Home > test

	Name	Size	Modified
..			
test.sh		41 B	Jun 18, 2018, 3:25 P

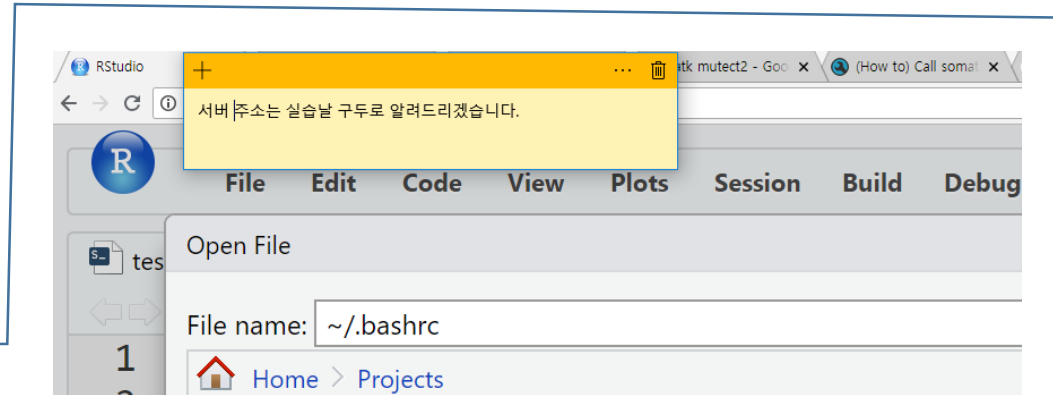
- 터미널 탭이 보이지 않으면



PATH 설정

\$ echo \$PATH 로 현재 PATH를 확인

export PATH=\$HOME/bin:\$PATH ← ~/.bashrc에 추가하고 저장



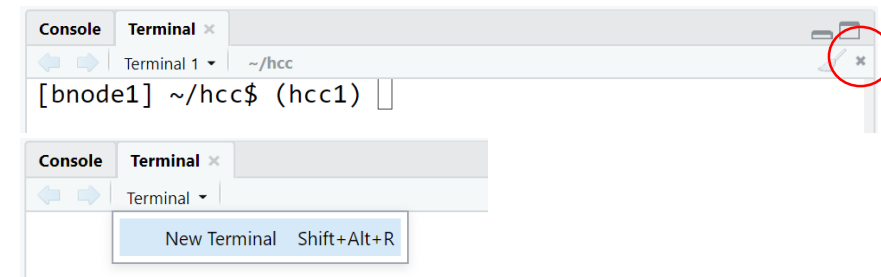
터미널을 재시작해야 \$PATH가 업데이트 됨

링크 만들기

\$ mkdir -p ~/bin

\$ cd ~/bin

\$ ln -s ~kjyi/tools/STAR/STAR-2.5.4b/bin/Linux_x86_64/STAR .



현재 directory 의미

PATH

- 잘 설정되었는지 테스트

\$ STAR -help

...long help message...

\$ _

tutorial

Rstudio-server terminal, 혹은 ssh 접속 환경에서 다음을 수행

```
$ cd ~           (home으로 이동)
$ mkdir day1     (day1이란 폴더를 만듦)
$ cd day1        (day1로 이동)
$ mkdir fastq    (day1안에 fastq라는 폴더를 만듦)
$ cd fastq       (fastq안으로 이동)
$ ln -s /home/users/kjyi/day1/fastq/* .
$ ls
$ cd ..
```

/home/users/kjyi/day1/fastq/ 에 있는
모든 파일에 대해 각 파일을 참조하는
softlink (바로가기)를 현재 위치에 생성

run_star.sh를 home director내의 day1 폴더 안에 작성

```
STAR --runMode alignReads \  
    --outSAMtype BAM Unsorted \  
    --genomeDir /home/users/kjyi/ref/hg19/star_index \  
    --outFileNamePrefix star/example1/example1 \  
    --readFilesIn ./fastq/R1.fastq ./fastq/R2.fastq
```

주의: \ 뒤에 공백이 없어야 함
OS/browser 언어/font 환경에 따라 \ 대신 ₩로 보일 수 있음
실제 사용시엔 [STAR manual](#) 참조할 것
e.g. 압축된 fastq (fastq.gz)를 읽으려면?

다음 중 한가지 방법으로 코드를 실행

\$ bash run_star.sh



\$ nohup bash run_star.sh 1> stdout.txt 2> stderr.txt &

\$ qsub -q day -l nodes=1:ppn=1 run_star.sh

(이 방법을 이용할 경우 run_star.sh의 맨 위에 다음과 같이 수정)

cd ~/day1

STAR --runMode alignReads \

...

- 결과물

star/example1/example1_STARgenome

star/example1/example1_STARtmp

star/example1/example1_STARAligned.out.bam

star/example1/example1_STARChimeric.out.junction

star/example1/example1_STARChimeric.out.sam

nohup code를 이용한 경우

stdout.txt

stderr.txt

qsub code를 이용한 경우

stdout.txt

stderr.txt

run_star.sh.o12345

run_star.sh.e12345

Further information

고급 Bash 스크립팅 가이드

<https://wiki.kldp.org/HOWTO/html/Adv-Bash-Scr-HOWTO/index.html>

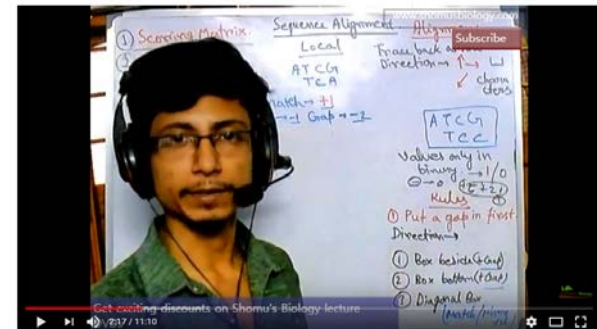
STAR github (manual/download/paper)

<https://github.com/alexdobin/STAR>

<https://www.ncbi.nlm.nih.gov/pubmed/23104886>

Smith-Waterman algorithm (basic local alignment)

<https://www.youtube.com/watch?v=latoWOsJ35Q&t=101s>



Quantification

Normalization

Advanced bash commands

(for, while, if, md5sum)

Introduction to R