Linux Fundamentals

Complete Command Reference for Bioinformatics

Companion Guide

February 2026

Section 1: Navigation & File System

Essential commands for navigating the Linux filesystem, understanding directory structure, and managing directories.

Command	Description	Syntax / Options	Example
pwd	Print current working directory path	`pwd`	`$ pwd` `/home/user/genomics`

ls	List directory contents	`ls [options] [path]`
		`-l long format`
		`-h human sizes`
		`-a show hidden`
		`-R recursive`
		`-t sort by time`
		`-S sort by size`
			`$ ls -lhS /data/fastq/`
			`$ ls -la ~/`
			`$ ls -R project/`

cd	Change directory	`cd [directory]`
		`cd .. parent dir`
		`cd ~ home dir`
		`cd - previous dir`
			`$ cd /data/genomes`
			`$ cd ..`
			`$ cd ~/projects`
			`$ cd -`

tree	Display directory structure as tree	`tree [options] [path]`
		`-L n depth limit`
		`-d dirs only`
		`-h human sizes`
		`-I ignore pattern`
			`$ tree -L 2 project/`
			`$ tree -d /data/`
			`$ tree -I 'node_modules'`

mkdir	Create directories	`mkdir [options] dir`
		`-p create parents`
		`-v verbose`
			`$ mkdir results`
			`$ mkdir -p project/{raw,aligned,vcf}`
			`$ mkdir -pv analysis/round1`

rmdir	Remove empty directories	`rmdir [dir]`	`$ rmdir empty_folder/`

Section 2: File Operations

Commands for copying, moving, removing, linking, and finding files. Includes disk usage monitoring.

Command	Description	Syntax / Options	Example

cp	Copy files and directories	cp [options] src dest
		-r recursive (dirs)
		-i interactive
		-v verbose
		-p preserve attribs
		-u update only
			$ cp sample.fq backup/
			$ cp -rv raw_data/ backup/
			$ cp -u *.vcf results/

mv	Move or rename files/dirs	mv [options] src dest
		-i interactive
		-v verbose
		-n no overwrite
			$ mv old.txt new.txt
			$ mv *.bam aligned/
			$ mv -iv data/ archive/

rm	Remove files and directories	rm [options] file
		-r recursive
		-f force
		-i interactive
		-v verbose
			$ rm temp.txt
			$ rm -rf old_results/
			$ rm -iv *.tmp

touch	Create empty file or update timestamp	touch [file]	$ touch notes.txt $ touch sample_{1..10}.fq
ln	Create links (symbolic or hard)	ln -s target link_name -s symbolic link -f force overwrite	$ ln -s /data/ref/hg38.fa ref.fa $ ln -sf new_ref.fa ref.fa

find	Search for files in directory tree	find [path] [expression]
		-name filename pattern
		-type f/d file/dir
		-size file size
		-mtime modified time
		-exec run command
			$ find . -name '*.fastq.gz'
			$ find /data -type f -size +1G
			$ find . -name '*.bam' -mtime -7
			$ find . -name '*.tmp' -exec rm {} \;

locate	Find files using indexed database (fast)	locate [pattern]
		updatedb update index
			$ locate hg38.fa
			$ locate '*.gtf'
			$ sudo updatedb

du	Estimate file/directory space usage	du [options] [path]
		-s summary only
		-h human readable
		-a all files
		--max-depth=N
			$ du -sh *
			$ du -sh /data/project/
			$ du -ah --max-depth=1 .

df	Report filesystem disk space usage	df [options] -h human readable -T show filesystem type	$ df -h $ df -hT /data
stat	Display detailed file information	stat [file]	$ stat sample.bam $ stat -c '%s %n' *.vcf

Section 3: Viewing & Inspecting Files

Tools for reading, paging through, comparing, and verifying files without modifying them.

Command	Description	Syntax / Options	Example

cat	Concatenate and print file contents	`cat [options] [file]`
		`-n number lines`
		`-b number non-blank`
		`-s squeeze blank lines`
			`$ cat sample.txt`
			`$ cat -n script.sh`
			`$ cat file1.txt file2.txt > merged.txt`

less	View file with pagination (scrollable)	`less [file]`
		`Space next page`
		`b prev page`
		`/term search`
		`q quit`
		`G end of file`
		`g start of file`
			`$ less large_file.vcf`
			`$ less -N aligned.sam`
			`$ zless sample.fastq.gz`

more	View file page by page (forward only)	`more [file]`
		`Space next page`
		`q quit`
			`$ more results.txt`

head	Display first lines of file	`head [options] [file]`
		`-n N first N lines`
		`-c N first N bytes`
			`$ head -n 20 sample.vcf`
			`$ head -n 4 reads.fastq`
			`$ head -c 1000 file.txt`

tail	Display last lines of file	`tail [options] [file]`
		`-n N last N lines`
		`-f follow (live)`
		`-c N last N bytes`
			`$ tail -n 50 alignment.log`
			`$ tail -f pipeline.log`
			`$ tail -n +2 file.tsv`

wc	Count lines, words, and bytes	`wc [options] [file]`
		`-l lines only`
		`-w words only`
		`-c bytes`
		`-m characters`
			`$ wc -l variants.vcf`
			`$ wc -l *.fastq`
			`$ cat reads.fq | echo $(wc -l)/4 | bc`

file	Determine file type	`file [file]`	`$ file sample.bam`
			`$ file mystery_file`
			`$ file *.gz`

diff	Compare files line by line	`diff [options] file1 file2`
		`-u unified format`
		`-y side by side`
		`-q brief (differ?)`
		`-r recursive (dirs)`
			`$ diff old.vcf new.vcf`
			`$ diff -u v1.txt v2.txt`
			`$ diff -rq dir1/ dir2/`

md5sum	Calculate MD5 checksum for integrity	`md5sum [file]`
		`-c check against file`
			`$ md5sum genome.fa`
			`$ md5sum -c checksums.md5`
			`$ md5sum *.fastq.gz > checksums.md5`

sha256sum	Calculate SHA-256 checksum	`sha256sum [file]`	`$ sha256sum reference.fa`
zcat	View gzipped files without decompressing	`zcat [file.gz]`	`$ zcat sample.fastq.gz | head -8` `$ zcat *.fastq.gz | wc -l`

Command	Description	Syntax / Options	Example
zless	Page through gzipped files	`zless [file.gz]`	`$ zless reads.fastq.gz`
zgrep	Search inside gzipped files	`zgrep [pattern] [file.gz]`	`$ zgrep 'ATCGATCG' reads.fq.gz` `$ zgrep -c '^@' sample.fq.gz`

Section 4: Text Processing (Critical for Bioinformatics)

The most powerful and frequently used commands in bioinformatics. Master these for efficient data manipulation.

Command	Description	Syntax / Options	Example

grep	Search for patterns in files	grep [options] pattern [file]
		-i ignore case
		-v invert match
		-c count matches
		-n line numbers
		-r recursive
		-l files with match
		-w whole word
		-E extended regex
		-P Perl regex
		-o only matching part
		-A/-B/-C context lines
			$ grep 'BRCA1' genes.gff
			$ grep -c '^>' sequences.fa
			$ grep -v '^#' file.vcf
			$ grep -rn 'ERROR' logs/
			$ grep -E 'PASS|WARN' results.vcf
			$ grep -oP 'gene=\K[^;]+' anno.gff

awk	Pattern scanning and text processing language	awk [options] 'pattern {action}' file -F field separator -v set variable Built-in variables: $0 entire line $1-N field N NR record number NF number of fields FS field separator OFS output separator BEGIN/END blocks	$ awk '{print $1, $4}' file.bed $ awk -F'\t' '$5 > 30' data.vcf $ awk 'NR>1 {sum+=$3} END{print sum/NR}' data.tsv $ awk -F'\t' '{OFS="\t"; print $1,$2,$3}' file.bed $ awk '/^>/ {print}' seqs.fa $ awk '{print NR, $0}' file.txt

sed	Stream editor for text transformation	sed [options] 'command' file
		s/old/new/g substitute
		d delete line
		p print line
		-i edit in-place
		-n suppress output
		-E extended regex
			$ sed 's/chr//' variants.vcf
			$ sed -i 's/\r$//' script.sh
			$ sed -n '10,20p' file.txt
			$ sed '/^#/d' file.vcf
			$ sed -E 's/gene_id "([^"]+)"/\1/' genes.gtf

cut	Extract columns/fields from files	cut [options] [file]
		-f N fields (tab-delim)
		-d C delimiter
		-c N characters
		--complement inverse
			$ cut -f1,4,5 data.tsv
			$ cut -f1-3 -d',' data.csv
			$ cut -c1-10 file.txt
			$ cut -f2 --complement data.tsv

sort	Sort lines of text files	sort [options] [file]
		-n numeric sort
		-r reverse
		-k sort by field
		-t delimiter
			$ sort -k2,2n data.bed
			$ sort -t'\t' -k5,5nr results.tsv
			$ sort -V chromosomes.txt

Command	Description	Syntax / Options	Example
		-u unique only -V version sort -S buffer size -T temp directory	$ sort -u names.txt $ sort -k1,1 -k2,2n file.bed

uniq	Filter/count adjacent duplicate lines	uniq [options] [file]
		-c count occurrences
		-d only duplicates
		-u only unique
		-i ignore case
		Note: input must be sorted!
			$ sort genes.txt | uniq
			$ sort genes.txt | uniq -c | sort -rn
			$ sort data.txt | uniq -d

tr	Translate or delete characters	tr [options] SET1 [SET2]
		-d delete chars
		-s squeeze repeats
		-c complement
			$ echo 'ATCG' | tr 'ATCG' 'TAGC'
			$ tr '\t' ',' < data.tsv > data.csv
			$ tr -d '\r' < win.txt > unix.txt
			$ tr -s ' ' '\t' < file.txt

paste	Merge lines of files column-wise	paste [options] file1 file2
		-d delimiter
		-s serial (transpose)
			$ paste file1.txt file2.txt
			$ paste -d',' col1.txt col2.txt
			$ paste - - - - < reads.fq

join	Join two sorted files on common field	join [options] file1 file2 -t delimiter -1/-2 join fields -a include unpairable	$ join -t'\t' -1 1 -2 1 genes.txt annotations.txt $ join -a1 file1.txt file2.txt

comm	Compare two sorted files line by line	comm [options] file1 file2
		-1 suppress col 1
		-2 suppress col 2
		-3 suppress col 3
			$ comm -12 list1.txt list2.txt
			(lines common to both)
			$ comm -23 list1.txt list2.txt
			(only in list1)

tee	Read stdin, write to stdout AND files	tee [options] file
		-a append
			$ cmd | tee output.log
			$ cmd | tee -a run.log
			$ cmd | tee file1.txt | wc -l

xargs	Build and execute commands from stdin	xargs [options] command
		-I{} placeholder
		-P N parallel jobs
		-n N args per cmd
			$ find . -name '*.bam' | xargs ls -lh
			$ cat samples.txt | xargs -I{} echo {}.fastq
			$ ls *.fq | xargs -P4 -I{} fastqc {}

column	Format text into aligned columns	column [options] -t create table -s delimiter	$ column -t -s'\t' data.tsv $ cat results.txt | column -t

Section 5: Pipes, Redirection & Chaining

The glue that connects Linux commands into powerful pipelines. These operators enable complex data workflows.

Command	Description	Syntax / Options	Example
| (pipe)	Send output of one command as input to another	`cmd1 | cmd2 | cmd3`	`$ grep -v '^#' file.vcf | wc -l` `$ cat genes.txt | sort | uniq -c | sort -rn`
> (redirect)	Redirect stdout to file (overwrite)	`cmd > file`	`$ echo 'hello' > output.txt` `$ sort data.txt > sorted.txt`
>> (append)	Redirect stdout to file (append)	`cmd >> file`	`$ echo 'new line' >> log.txt` `$ date >> timestamps.txt`
2>	Redirect stderr to file	`cmd 2> error.log`	`$ samtools sort in.bam 2> errors.log`
2>&1	Merge stderr into stdout	`cmd > all.log 2>&1` `cmd &> all.log`	`$ pipeline.sh > run.log 2>&1` `$ bwa mem ref.fa r1.fq r2.fq &> align.log`
< (input)	Redirect file as stdin	`cmd < file`	`$ tr ',' '\t' < data.csv > data.tsv`
&& (and)	Run next command only if previous succeeds	`cmd1 && cmd2`	`$ mkdir results && cd results` `$ bwa index ref.fa && echo 'Done!'`
|| (or)	Run next command only if previous fails	`cmd1 || cmd2`	`$ test -f ref.fa || echo 'File missing!'` `$ cd data || mkdir data`
cmd &	Run command in background	`cmd &`	`$ long_pipeline.sh &` `$ nohup alignment.sh &`
; (sequential)	Run commands sequentially regardless of exit status	`cmd1 ; cmd2`	`$ cd data ; ls -lh` `$ echo 'start' ; sleep 5 ; echo 'end'`

Section 6: Compression & Archives

Compression tools essential for handling large genomic data files efficiently.

Command	Description	Syntax / Options	Example

gzip	Compress files (.gz format)	gzip [options] file
		-d decompress
		-k keep original
		-1 to -9 compression level
		-c stdout
			$ gzip large_file.sam
			$ gzip -k sample.fastq
			$ gzip -c file.vcf > file.vcf.gz
			$ gzip -d file.gz

gunzip	Decompress .gz files	gunzip [file.gz]
		-k keep original
		-c stdout
			$ gunzip sample.fastq.gz
			$ gunzip -k reads.fq.gz
			$ gunzip -c file.gz > file.txt

bgzip	Block-compress for random access (bioinformatics)	bgzip [options] file
		-c stdout
		-d decompress
		-@ N threads
			$ bgzip variants.vcf
			$ bgzip -@ 4 large.vcf
			$ bgzip -c in.vcf > out.vcf.gz

tabix	Index bgzipped tab-delimited files	tabix [options] file.gz
		-p preset (vcf,bed,gff)
		-s/-b/-e seq/begin/end cols
			$ tabix -p vcf variants.vcf.gz
			$ tabix -p bed regions.bed.gz
			$ tabix file.vcf.gz chr1:1000-2000

tar	Archive and compress directories	tar [options] archive files
		-c create
		-x extract
		-z gzip
		-j bzip2
		-v verbose
		-f filename
		-t list contents
			$ tar -czvf project.tar.gz project/
			$ tar -xzvf data.tar.gz
			$ tar -tzvf archive.tar.gz
			$ tar -cjvf data.tar.bz2 data/

zip / unzip	Create/extract ZIP archives	zip [options] out.zip files
		-r recursive
		unzip archive.zip
		-l list contents
			$ zip -r results.zip results/
			$ unzip data.zip
			$ unzip -l archive.zip

bzip2 / bunzip2	Compress/decompress with bzip2 (higher compression)	bzip2 [file] bunzip2 [file.bz2] -k keep original	$ bzip2 large_file.sam $ bunzip2 -k data.bz2

Section 7: Permissions, Environment & Configuration

Managing file permissions, environment variables, and shell configuration for a productive working environment.

Command	Description	Syntax / Options	Example

chmod	Change file permissions	chmod [mode] file
		+x add execute
		-w remove write
		755 = rwxr-xr-x
		644 = rw-r--r--
		-R recursive
			$ chmod +x script.sh
			$ chmod 755 pipeline.sh
			$ chmod -R 644 data/
			$ chmod u+w,g-w file.txt

chown	Change file owner and group	chown [user]:[group] file -R recursive	$ chown user:bioinfo data/ $ chown -R user:group project/

export	Set environment variable for session	export VAR=value	$ export PATH=$PATH:/opt/tools/bin
			$ export REF=/data/ref/hg38.fa
			$ export THREADS=8

echo	Display text or variable value	echo [options] [text] -n no newline -e enable escapes	$ echo $PATH $ echo $HOME $ echo -e 'line1\nline2' $ echo "Ref is: $REF"

which	Show full path of command executable	which [command]	$ which samtools
			$ which python3
			$ which bwa

whereis	Locate binary, source, and manual pages	whereis [command]	$ whereis python $ whereis gcc

alias	Create shortcut for a command	alias name='command'	$ alias ll='ls -lh'
			$ alias gs='git status'
			$ alias samtop='samtools view -h'

source / .	Execute commands from file in current shell	source file
		. file
			$ source ~/.bashrc
			$ source activate myenv
			$. ~/.bash_profile

env	Display or modify environment	env printenv [VAR]	$ env | grep PATH $ printenv HOME
~/.bashrc	User shell configuration file (runs on each terminal)	Edit with: nano ~/.bashrc Reload: source ~/.bashrc	# Add to ~/.bashrc: export PATH=$PATH:/opt/tools alias ll='ls -lh' export REF=/data/hg38.fa

Section 8: Process & Job Management

Monitor and control running processes, manage background jobs, and use persistent terminal sessions.

Command	Description	Syntax / Options	Example

top / htop	Real-time process monitor	top
		htop (interactive, colored)
		Keys: q quit, k kill, P cpu sort, M mem sort
			$ top
			$ htop
			$ top -u username

ps	List running processes	ps [options]
		aux all processes
		-ef full format
		-u by user
			$ ps aux
			$ ps aux | grep samtools
			$ ps -u $USER
			$ ps -ef | grep bwa

kill	Send signal to terminate process	kill [signal] PID
		-9 force kill (SIGKILL)
		-15 graceful (SIGTERM)
		killall name
			$ kill 12345
			$ kill -9 12345
			$ killall bwa
			$ pkill -f 'python pipeline'

nohup	Run command immune to hangup signals	nohup cmd &	$ nohup pipeline.sh & $ nohup bwa mem ref.fa r1.fq r2.fq > out.sam &

screen	Terminal multiplexer (persistent sessions)	screen -S name new session
		screen -r name reattach
		Ctrl+A D detach
		screen -ls list
			$ screen -S alignment
			$ screen -r alignment
			$ screen -ls

tmux	Terminal multiplexer (modern alternative)	tmux new -s name
		tmux attach -t name
		tmux ls
		Ctrl+B D detach
		Ctrl+B % split vert
		Ctrl+B " split horiz
			$ tmux new -s pipeline
			$ tmux attach -t pipeline
			$ tmux ls

jobs / fg / bg	Manage background/foreground jobs	jobs list jobs
		fg %N bring to foreground
		bg %N resume in background
		Ctrl+Z suspend current job
			$ jobs
			$ fg %1
			$ bg %2

time	Measure command execution time	time command	$ time bwa mem ref.fa r1.fq > out.sam $ time sort -k1,1 huge_file.bed
nice / renice	Set process priority (-20 highest to 19 lowest)	nice -n N command renice N -p PID	$ nice -n 10 heavy_job.sh $ renice 15 -p 12345
watch	Run command repeatedly and watch output	watch [options] command -n N interval (sec)	$ watch -n 5 'ls -lh results/'

Command	Description	Syntax / Options	Example
		`-d  highlight changes`	`$ watch -n 10 'squeue -u $USER'`

Section 9: Networking & Data Transfer

Download data, transfer files between servers, and connect to remote HPC systems.

Command	Description	Syntax / Options	Example

wget	Download files from the web	wget [options] URL
		-O output filename
		-c continue partial
		-r recursive
		-q quiet mode
		-P output directory
			$ wget https://url.com/hg38.fa.gz
			$ wget -O ref.fa https://url.com/ref
			$ wget -c large_file.tar.gz
			$ wget -P /data/ref/ URL

curl	Transfer data from URLs (versatile)	curl [options] URL
		-O save with orig name
		-o output file
		-L follow redirects
		-s silent
		-I headers only
			$ curl -O https://url.com/file.gz
			$ curl -sL url | gunzip > out.fa
			$ curl -I https://url.com/file

scp	Secure copy between hosts over SSH	scp [options] src dest
		-r recursive
		-P port
		-C compress
			$ scp file.bam user@server:/data/
			$ scp user@server:/data/*.vcf ./
			$ scp -r project/ user@hpc:~/

rsync	Efficient file sync (only transfers changes)	rsync [options] src dest
		-a archive mode
		-v verbose
		-z compress transfer
		--progress show progress
		--dry-run test first
		-e ssh over SSH
			$ rsync -avz data/ user@server:/backup/
			$ rsync -avz --progress *.bam server:/data/
			$ rsync -av --dry-run src/ dest/

ssh	Secure shell remote login	ssh [options] user@host
		-p port
		-i identity file
		-X X11 forwarding
		-L port forwarding
			$ ssh user@hpc.university.edu
			$ ssh -X user@server (for GUI)
			$ ssh -p 2222 user@server

ssh-keygen	Generate SSH key pair for passwordless login	ssh-keygen [options]
		-t key type (rsa, ed25519)
		-b bits
		ssh-copy-id copy to server
			$ ssh-keygen -t ed25519
			$ ssh-copy-id user@server
			$ ssh-keygen -t rsa -b 4096

sftp	Interactive secure file transfer	sftp user@host get/put download/upload lcd/cd local/remote cd ls/lls remote/local ls	$ sftp user@server sftp> get data.bam sftp> put results.vcf sftp> mget *.fastq.gz

Section 10: Bash Scripting Essentials

Core scripting constructs for automating bioinformatics pipelines and repetitive tasks.

Command	Description	Syntax / Options	Example
#!/bin/bash	Shebang - specify script interpreter	First line of script Make executable: chmod +x	#!/bin/bash set -euo pipefail echo 'Pipeline starting...'

for loop	Iterate over list of items	for var in list; do
		  commands
		done
			$ for f in *.fastq.gz; do
			  fastqc $f
			  done
			$ for i in {1..22} X Y; do
			  echo "chr$i"
			  done

while loop	Loop while condition is true	while read line; do
		  commands
		done < file
			$ while read sample; do
			  bwa mem ref.fa ${sample}_R1.fq ${sample}_R2.fq > ${sample}.sam
			  done < samples.txt

if/else	Conditional execution	if [condition];
		then
		  cmd1
		elif [cond2]; then
		  cmd2
		else
		  cmd3
		fi
			$ if [-f ref.fa];
			then
			  echo 'Found reference'
			  else
			  echo 'Missing!'
			  fi

test / [ ]	Evaluate conditional expressions	-f file exists -d dir exists -s file not empty -z string is empty -eq numeric equal -gt greater than	$ [-f sample.bam] && echo 'exists' $ [! -d results] && mkdir results $ [$(wc -l < file) -gt 100] && echo 'big'
Variables	Store and use values	VAR=value (no spaces!) $VAR use variable ${VAR} explicit $(cmd) command substitution $() preferred over backticks	$ REF=/data/hg38.fa $ THREADS=8 $ READS=$(wc -l < file.fq) $ echo "Using $THREADS cores" $ OUTDIR="results_$(date +%Y%m%d)"
set -euo pipefail	Strict mode for robust scripts	set -e exit on error set -u error on unset var set -o pipefail pipe errors	#!/bin/bash set -euo pipefail # Script will stop on any error

functions	Define reusable code blocks	function_name() {
		  commands
		  return value
		}
			count_reads() {
			  echo $(($(wc -l < $1) / 4))
			}
			count_reads sample.fq

Section 11: Bioinformatics-Specific Tools

Essential bioinformatics tools: samtools, bcftools, bedtools, FastQC, conda, and more.

Command	Description	Syntax / Options	Example

samtools view	View/convert/filter SAM/BAM/CRAM	`samtools view [options] file`
		`-b output BAM`
		`-h include header`
		`-f/-F require/exclude flags`
		`-q min MAPQ`
		`-@ N threads`
			`$ samtools view -bS in.sam > out.bam`
			`$ samtools view -b -q 30 in.bam > filt.bam`
			`$ samtools view -F 4 in.bam > mapped.bam`
			`$ samtools view -c in.bam # count`

samtools sort	Sort BAM file by coordinate or name	`samtools sort [options] file`
		`-o output file`
		`-n sort by name`
		`-@ N threads`
		`-m memory per thread`
			`$ samtools sort -@ 8 -o sorted.bam in.bam`
			`$ samtools sort -n -o namesort.bam in.bam`

samtools index	Create index (.bai) for sorted BAM	`samtools index [options] file.bam` `-@ N threads`	`$ samtools index sorted.bam` `$ samtools index -@ 4 aligned.bam`
samtools flagstat	BAM alignment statistics summary	`samtools flagstat file.bam`	`$ samtools flagstat aligned.bam` `$ samtools flagstat aligned.bam > stats.txt`

samtools depth	Compute read depth at each position	`samtools depth [options] file`
		`-a all positions`
		`-b BED regions`
		`-q min base quality`
			`$ samtools depth -a sorted.bam > depth.txt`
			`$ samtools depth -b target.bed sorted.bam`

samtools idxstats	Per-chromosome alignment counts	`samtools idxstats file.bam`	`$ samtools idxstats sorted.bam` `$ samtools idxstats in.bam | awk '{s+=$3}END{print s}'`

bcftools view	View/filter/subset VCF/BCF files	`bcftools view [options] file`
		`-f filter expression`
		`-i include`
		`-e exclude`
		`-s samples`
		`-r regions`
			`$ bcftools view -f PASS variants.vcf`
			`$ bcftools view -i 'QUAL>30' in.vcf`
			`$ bcftools view -r chr1 in.vcf.gz`

bcftools stats	Generate VCF/BCF statistics	`bcftools stats [options] file`	`$ bcftools stats variants.vcf > stats.txt` `$ bcftools stats -s - multi.vcf`

bcftools query	Extract fields from VCF into text	`bcftools query -f 'format' file`
		`%CHROM, %POS, %REF, %ALT`
		`%QUAL, %FILTER, %INFO/tag`
			`$ bcftools query -f '%CHROM\t%POS\t%REF\t%ALT\n' in.vcf`
			`$ bcftools query -f '%INFO/AF\n' in.vcf`

Command	Description	Syntax / Options	Example

bedtools intersect	Find overlapping genomic regions	bedtools intersect [options]
		-a file A
		-b file B
		-v non-overlapping
		-wa/-wb write A/B entry
		-f min overlap fraction
			$ bedtools intersect -a peaks.bed -b genes.bed
			$ bedtools intersect -a var.vcf -b exons.bed -wa
			$ bedtools intersect -a a.bed -b b.bed -v

bedtools coverage	Compute coverage of A over B	bedtools coverage -a regions -b reads	$ bedtools coverage -a exons.bed -b aligned.bam $ bedtools coverage -a targets.bed -b sorted.bam -d
fastqc	Quality control for FASTQ files	fastqc [options] file.fq -o output directory -t threads --noextract	$ fastqc sample_R1.fq.gz sample_R2.fq.gz $ fastqc -o qc_reports/ -t 4 *.fastq.gz
multiqc	Aggregate QC results into one report	multiqc [directory] -o output dir -n report name	$ multiqc qc_reports/ $ multiqc . -o multiqc_output/

conda	Package and environment manager	conda create -n env pkg
		conda activate env
		conda deactivate
		conda install pkg
		conda env list
		conda list
			$ conda create -n ngs samtools bwa
			$ conda activate ngs
			$ conda install -c bioconda gatk4
			$ conda env list

module	Load/unload HPC software modules	module load pkg
		module unload pkg
		module list
		module avail
		module purge
			$ module load samtools/1.19
			$ module avail | grep bwa
			$ module list
			$ module purge

parallel	Run commands in parallel (GNU parallel)	parallel [options] cmd ::: args
		-j N max jobs
		--bar progress bar
		--dry-run show commands
			$ parallel -j4 fastqc {} ::: *.fq.gz
			$ parallel -j8 'samtools index {}' ::: *.bam
			$ cat cmds.txt | parallel -j10

Section 12: HPC / SLURM Job Scheduler

Commands for submitting and managing jobs on high-performance computing clusters using SLURM.

Command	Description	Syntax / Options	Example

sbatch	Submit job script to SLURM scheduler	sbatch [options] script.sh
		--job-name
		--output
		--ntasks
		--cpus-per-task
		--mem
		--time
		--partition
			$ sbatch pipeline.sh
			$ sbatch --mem=32G --cpus-per-task=8 job.sh
			$ sbatch -p gpu --gres=gpu:1 job.sh

squeue	View SLURM job queue	squeue [options]
		-u by user
		-j by job ID
		-p by partition
			$ squeue -u $USER
			$ squeue -j 123456
			$ squeue -p compute

scancel	Cancel SLURM job(s)	scancel [job_id]
		-u cancel all by user
		-n by job name
			$ scancel 123456
			$ scancel -u $USER
			$ scancel -n alignment

sinfo	View SLURM cluster partition info	sinfo [options]
		-N node list
		-p partition
			$ sinfo
			$ sinfo -N -l
			$ sinfo -p gpu

sacct	View past job accounting data	sacct [options] -j job ID --format fields -S start date	$ sacct -j 123456 --format=JobID,Elapsed,MaxRSS $ sacct -S 2024-01-01 -u $USER
srun	Run interactive job on compute node	srun [options] command --pty pseudo terminal	$ srun --pty -p interactive bash $ srun --mem=16G --cpus-per-task=4 ./script.sh

Quick Reference: Power One-Liners for Bioinformatics

 # Count reads in a FASTQ file

```
echo $(( $(wc -l < sample.fastq) / 4 ))
```

 # Count reads in gzipped FASTQ

```
zcat sample.fastq.gz | echo $(( $(wc -l) / 4 ))
```

 # Extract gene names from GFF3

```
awk -F'\t' '$3=="gene"' annotation.gff | grep -oP 'Name=\K[^;]+'
```

 # Count variants per chromosome

```
grep -v '^#' variants.vcf | cut -f1 | sort | uniq -c | sort -rn
```

 # Find the 20 largest files

```
du -ah . | sort -rh | head -20
```

 # Run FastQC on all FASTQ files in parallel

```
ls *.fastq.gz | parallel -j 4 'fastqc {}'
```

 # Convert SAM to sorted indexed BAM

```
samtools sort -@ 8 -o sorted.bam input.sam && samtools index sorted.bam
```

 # Extract PASS variants with QUAL > 30

```
bcftools view -f PASS -i 'QUAL>30' variants.vcf.gz > filtered.vcf
```

 # Get average coverage from samtools depth

```
samtools depth -a sorted.bam | awk '{sum+=$3} END{print sum/NR}'
```

 # Batch rename files (remove prefix)

```
for f in sample_*.bam; do mv "$f" "${f#sample_}"; done
```

 # Monitor disk usage while pipeline runs

```
watch -n 30 'du -sh results/'
```

 # Count unique mapped reads (exclude unmapped & duplicates)

```
samtools view -F 1028 -q 30 -c sorted.bam
```