**Script 1 — Filtro de calidad (genera los \*.filtered.vcf)**

**Guarda como filtra\_calidad\_dp25.sh:**

#!/usr/bin/env bash

set -euo pipefail

# USO:

# ./filtra\_calidad\_dp25.sh m1des.vcf m2des.vcf m3des.vcf

# (sirve también para .vcf.gz)

# --- UMBRALES DE FILTRADO ---

QUAL=30 # calidad mínima de la variante

MIN\_DP=25 # PROFUNDIDAD mínima (25× como en tu TFM)

AB\_MIN=0.25 # balance alélico mínimo en heterocigotos

AB\_MAX=0.75 # balance alélico máximo en heterocigotos

for VCF in "$@"; do

[[ -f "$VCF" ]] || { echo "No existe: $VCF" >&2; exit 1; }

OUT="${VCF%.vcf}.filtered.vcf"; OUT="${OUT%.vcf.gz}.filtered.vcf"

# lector según extensión

if [[ "$VCF" =~ \.vcf\.gz$ ]]; then READER="zcat --force"; else READER="cat"; fi

{

# Cabecera

$READER "$VCF" | awk '/^##/ {print} /^#CHROM/ {print; exit}'

# Cuerpo con filtros

$READER "$VCF" | awk -v QUAL="$QUAL" -v MIN\_DP="$MIN\_DP" -v AB\_MIN="$AB\_MIN" -v AB\_MAX="$AB\_MAX" -F'\t' '

function split\_fields(fmt,arr){return split(fmt,arr,":")}

function idx(fmt,name, a,i,n){n=split(fmt,a,":"); for(i=1;i<=n;i++) if(a[i]==name) return i; return 0}

function val(sample,i, a,n){if(i<=0) return ""; n=split(sample,a,":"); return a[i]}

function getInfo(key, info, a,i,n,b){n=split(info,a,";"); for(i=1;i<=n;i++){ if(a[i]~("^"key"=")){ split(a[i],b,"="); return b[2] } else if(a[i]==key){ return 1 } } return ""}

function getDP(fmt,sample,info, j,v){ j=idx(fmt,"DP"); if(j){ v=val(sample,j); if(v!="") return v+0 } v=getInfo("DP",info); if(v!="") return v+0; return -1 }

function getAB(fmt,sample,info, j,ad,a,ro,ao){

v=getInfo("AB",info); if(v!=""){ return v+0 }

j=idx(fmt,"AD"); if(j){ ad=val(sample,j); if(ad!=""){ split(ad,a,","); if((a[1]+a[2])>0) return a[2]/(a[1]+a[2]) } }

j1=idx(fmt,"RO"); j2=idx(fmt,"AO"); if(j1&&j2){ ro=val(sample,j1)+0; ao=val(sample,j2)+0; if(ro+ao>0) return ao/(ro+ao) }

return -1

}

/^#/ {next}

{

qual=$6+0

dp = getDP($9,$10,$8)

gt = val($10, idx($9,"GT"))

# --- filtros de calidad ---

if(qual < QUAL) next

if(dp>=0 && dp < MIN\_DP) next

if(gt=="" || gt=="./." || gt ~ /^0[\/|]0/) next

# filtro de AB solo en heterocigotos

if(gt ~ /^0[\/|]1|^1[\/|]0|^0\|1|^1\|0/){

ab = getAB($9,$10,$8)

if(ab>=0 && (ab < AB\_MIN || ab > AB\_MAX)) next

}

print

}

'

} > "$OUT"

echo "✓ Generado: $OUT"

done

ejecutamos:

chmod +x filtra\_calidad\_dp25.sh

./filtra\_calidad\_dp25.sh m1des.vcf m2des.vcf m3des.vcf

Obteniendo: m1des.filtered.vcf, m2des.filtered.vcf, m3des.filtered.vcf.

**Script 2 — Cigosidad (usa los \*.filtered.vcf)**

Guarda como cigosidad\_postfiltro.sh:

#!/usr/bin/env bash

set -euo pipefail

# USO:

# ./cigosidad\_postfiltro.sh m1des.filtered.vcf m2des.filtered.vcf m3des.filtered.vcf

OUT="tabla\_antes\_despues\_cigosidad.tsv"

echo -e "Muestra\tTotales(crudo)\tRetenidas(post-filtro)\tHet 0/1 (n)\tHet (%)\tHomAlt 1/1 (n)\tHomAlt (%)" > "$OUT"

for F in "$@"; do

[[ -f "$F" ]] || { echo "No existe: $F" >&2; exit 1; }

SAMPLE\_ORIG="${F%.filtered.vcf}.vcf"

[[ -f "$SAMPLE\_ORIG" ]] || SAMPLE\_ORIG="${F%.filtered.vcf}.vcf.gz"

if [[ "$F" =~ \.vcf\.gz$ ]]; then R1="zcat --force"; else R1="cat"; fi

if [[ "$SAMPLE\_ORIG" =~ \.vcf\.gz$ ]]; then R0="zcat --force"; else R0="cat"; fi

TOTAL\_RAW=$($R0 "$SAMPLE\_ORIG" | grep -vc '^#')

read HET HOMALT TOTAL\_FILT < <(

$R1 "$F" | awk -F'\t' '

function split\_fields(fmt,arr){return split(fmt,arr,":")}

function idx(fmt,name, a,i,n){n=split(fmt,a,":"); for(i=1;i<=n;i++) if(a[i]==name) return i; return 0}

function val(sample,i, a,n){if(i<=0) return ""; n=split(sample,a,":"); return a[i]}

/^#/ {next}

{

gt = val($10, idx($9,"GT"))

if(gt ~ /^0[\/|]1|^1[\/|]0|^0\|1|^1\|0/) het++

else if(gt ~ /^1[\/|]1/) homalt++

}

END{ printf "%d %d %d\n", (het+0), (homalt+0), (het+homalt+0) }

'

)

if [[ "${TOTAL\_FILT:-0}" -gt 0 ]]; then

PH=$(awk -v a="$HET" -v t="$TOTAL\_FILT" 'BEGIN{printf "%.2f", (a\*100.0/t)}')

PM=$(awk -v a="$HOMALT" -v t="$TOTAL\_FILT" 'BEGIN{printf "%.2f", (a\*100.0/t)}')

else

PH="0.00"; PM="0.00"

fi

echo -e "$(basename "$F")\t$TOTAL\_RAW\t$TOTAL\_FILT\t$HET\t$PH\t$HOMALT\t$PM" >> "$OUT"

done

column -t -s $'\t' "$OUT"

echo "Guardado: $OUT"

ejecutar:

chmod +x cigosidad\_postfiltro.sh

./cigosidad\_postfiltro.sh m1des.filtered.vcf m2des.filtered.vcf m3des.filtered.vcf

**Script3 — SNVs/Indels después del filtro**

Guárdalo como snvindel\_postfiltro.sh:

#!/usr/bin/env bash

set -euo pipefail

# USO:

# ./snvindel\_postfiltro.sh m1des.filtered.vcf m2des.filtered.vcf m3des.filtered.vcf

# (acepta .vcf y .vcf.gz)

#

# Clasifica por ALT (maneja multialélicos contando cada ALT):

# - SNV: len(REF)==1 && len(ALT)==1

# - MNP: len(REF)>1 && len(ALT)>1 && len(REF)==len(ALT)

# - Inserción: len(ALT) > len(REF)

# - Deleción: len(REF) > len(ALT)

# - Indel cx.: resto (p.ej., combinaciones complejas)

#

# Salida: tabla\_postfiltro\_snv\_indel.tsv

OUT="tabla\_postfiltro\_snv\_indel.tsv"

echo -e "Muestra\tTotal\_filtrado\tSNVs\tMNPs\tInserciones\tDeleciones\tIndels\_complejos\tSNVs(%)\tMNPs(%)\tIns(%)\tDel(%)\tIndelCx(%)" > "$OUT"

for F in "$@"; do

[[ -f "$F" ]] || { echo "No existe: $F" >&2; exit 1; }

# lector

if [[ "$F" =~ \.vcf\.gz$ ]]; then R="zcat --force"; else R="cat"; fi

$R "$F" | awk -F'\t' '

BEGIN{snv=mnp=ins=del=icx=0}

/^#/ {next}

{

ref=$4

n=split($5,alts,",")

for(i=1;i<=n;i++){

alt=alts[i]

lr=length(ref); la=length(alt)

if(lr==1 && la==1) snv++

else if(lr>1 && la>1 && lr==la) mnp++

else if(la>lr) ins++

else if(lr>la) del++

else icx++

}

total += n

}

END{

tf = (total>0)? total : 0

psnv = (tf? snv\*100.0/tf : 0)

pmnp = (tf? mnp\*100.0/tf : 0)

pins = (tf? ins\*100.0/tf : 0)

pdel = (tf? del\*100.0/tf : 0)

picx = (tf? icx\*100.0/tf : 0)

printf "%s\t%d\t%d\t%d\t%d\t%d\t%d\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\n",

FILENAME, tf, snv, mnp, ins, del, icx, psnv, pmnp, pins, pdel, picx

}

' FILENAME="$(basename "$F")" >> "$OUT"

done

column -t -s $'\t' "$OUT"

echo "Guardado: $OUT"

ejecutar:

chmod +x snvindel\_postfiltro.sh

./snvindel\_postfiltro.sh m1des.filtered.vcf m2des.filtered.vcf m3des.filtered.vcf

**Script4 Productos de la anotación (para resultados)**

**Lista de genes con impacto HIGH/MODERATE**

Guarda como extract\_genes\_and\_variants.sh (genera también una **tabla** de variantes):

#!/usr/bin/env bash

set -euo pipefail

# USO:

# ./extract\_genes\_and\_variants.sh m1des.filtered.vep.vcf.gz m2des.filtered.vep.vcf.gz ...

ONLY\_PC=0 # pon 1 si quieres limitar a biotipo protein\_coding

for VCF in "$@"; do

[[ -f "$VCF" ]] || { echo "No existe: $VCF" >&2; exit 1; }

if [[ "$VCF" =~ \.vcf\.gz$ ]]; then R="zcat --force"; else R="cat"; fi

base="${VCF%.vcf.gz}"; base="${base%.vcf}"

GENES\_HM="${base}.genes.symbol.HM.txt"

VARS\_TSV="${base}.variants.HM.tsv"

# Lista de genes HIGH/MODERATE

$R "$VCF" | awk -v ONLY\_PC="$ONLY\_PC" -F'\t' '

/^#/ {next}

{

n=split($8,info,";");

for(i=1;i<=n;i++){

if(info[i] ~ /^CSQ=/){

sub(/^CSQ=/,"",info[i]);

m=split(info[i],rec,",");

for(j=1;j<=m;j++){

k=split(rec[j],f,"|");

# VEP típico: f[3]=IMPACT, f[4]=SYMBOL, f[2]=Consequence, f[8]=BIOTYPE

if(k>=5 && f[4]!=""){

if(f[3]=="HIGH" || f[3]=="MODERATE"){

if(ONLY\_PC==0 || (ONLY\_PC==1 && k>=8 && f[8]=="protein\_coding")) print f[4];

}

}

}

}

}

}

' | sort -u > "$GENES\_HM"

echo "✓ Genes HM: $GENES\_HM"

# Tabla de variantes HIGH/MODERATE (campos clave)

$R "$VCF" | awk -F"\t" '

BEGIN{

OFS="\t";

print "CHROM","POS","REF","ALT","GT","SYMBOL","IMPACT","Consequence","BIOTYPE","CLIN\_SIG","gnomAD\_AF";

}

/^#/ {next}

{

n=split($8,info,";"); csq="";

for(i=1;i<=n;i++) if(info[i]~/^CSQ=/){ sub(/^CSQ=/,"",info[i]); csq=info[i]; break }

if(csq=="") next;

# Toma el primer registro CSQ (suele ser el más severo si VEP ordena)

split(csq,rec,","); split(rec[1],f,"|");

symbol=(length(f)>=4?f[4]:""); impact=(length(f)>=3?f[3]:""); cons=(length(f)>=2?f[2]:""); biotype=(length(f)>=8?f[8]:"");

clin=(length(f)>=14?f[14]:""); af=(length(f)>=20?f[20]:"");

# Genotipo

split($9,F,":"); split($10,S,":"); gt=""

for(i=1;i<=length($9);i++) if(F[i]=="GT"){ gt=S[i]; break }

if(impact=="HIGH" || impact=="MODERATE"){

print $1,$2,$4,$5,gt,symbol,impact,cons,biotype,clin,af;

}

}

' > "$VARS\_TSV"

echo "✓ Variantes HM: $VARS\_TSV"

done

ejecutar:

chmod +x extract\_genes\_and\_variants.sh

./extract\_genes\_and\_variants.sh m1des.filtered.vep.vcf m2des.filtered.vep.vcf m3des.filtered.vep.vcf

Script5 **Filtrado en dos modos (DP≥25 y DP≥10)**

Guarda como filter\_dual\_dp.sh:

#!/usr/bin/env bash

set -euo pipefail

# USO:

# ./filter\_dual\_dp.sh m1des.vcf m2des.vcf m3des.vcf

# Salidas:

# mXdes.dp25.filtered.vcf

# mXdes.dp10.filtered.vcf

QUAL=30

AB\_MIN=0.25

AB\_MAX=0.75

filter\_one() {

local VCF="$1" MIN\_DP="$2" OUT="$3"

if [[ "$VCF" =~ \.vcf\.gz$ ]]; then R="zcat --force"; else R="cat"; fi

{

# cabecera

$R "$VCF" | awk '/^##/{print} /^#CHROM/{print; exit}'

# cuerpo con filtros

$R "$VCF" | awk -v QUAL="$QUAL" -v MIN\_DP="$MIN\_DP" -v AB\_MIN="$AB\_MIN" -v AB\_MAX="$AB\_MAX" -F'\t' '

function idx(fmt,name, a,i,n){n=split(fmt,a,":"); for(i=1;i<=n;i++) if(a[i]==name) return i; return 0}

function val(sample,i, a,n){if(i<=0) return ""; n=split(sample,a,":"); return a[i]}

function getInfo(key, info, a,i,n,b){n=split(info,a,";"); for(i=1;i<=n;i++){ if(a[i]~("^"key"=")){ split(a[i],b,"="); return b[2] } else if(a[i]==key){ return 1 } } return ""}

function getDP(fmt,sample,info, j,v){ j=idx(fmt,"DP"); if(j){ v=val(sample,j); if(v!="") return v+0 } v=getInfo("DP",info); if(v!="") return v+0; return -1 }

function getAB(fmt,sample,info, v,j,ad,a,ro,ao){

v=getInfo("AB",info); if(v!=""){ return v+0 }

j=idx(fmt,"AD"); if(j){ ad=val(sample,j); if(ad!=""){ split(ad,a,","); if((a[1]+a[2])>0) return a[2]/(a[1]+a[2]) } }

j1=idx(fmt,"RO"); j2=idx(fmt,"AO"); if(j1&&j2){ ro=val(sample,j1)+0; ao=val(sample,j2)+0; if(ro+ao>0) return ao/(ro+ao) }

return -1

}

/^#/ {next}

{

qual=$6+0

dp = getDP($9,$10,$8)

gt = val($10, idx($9,"GT"))

if(qual < QUAL) next

if(dp>=0 && dp < MIN\_DP) next

if(gt=="" || gt=="./." || gt ~ /^0[\/|]0/) next

if(gt ~ /^(0[\/|]1|1[\/|]0|0\|1|1\|0)$/){

ab = getAB($9,$10,$8)

if(ab>=0 && (ab < AB\_MIN || ab > AB\_MAX)) next

}

print

}

'

} > "$OUT"

}

for VCF in "$@"; do

[[ -f "$VCF" ]] || { echo "No existe: $VCF" >&2; exit 1; }

base="${VCF%.vcf.gz}"; base="${base%.vcf}"

out25="${base}.dp25.filtered.vcf"

out10="${base}.dp10.filtered.vcf"

echo ">> Filtrando $VCF con DP>=25..."

filter\_one "$VCF" 25 "$out25"

echo "✓ $out25"

echo ">> Filtrando $VCF con DP>=10..."

filter\_one "$VCF" 10 "$out10"

echo "✓ $out10"

done

Ejecuta:

chmod +x filter\_dual\_dp.sh

./filter\_dual\_dp.sh m1des.vcf m2des.vcf m3des.vcf

**Métricas post-filtro: totales, tipos y cigosidad**

Guarda como postfilter\_metrics.sh

#!/usr/bin/env bash

set -euo pipefail

# USO:

# ./postfilter\_metrics.sh m1des.dp25.filtered.vcf m1des.dp10.filtered.vcf ...

# Produce una tabla resumen en stdout.

echo -e "Muestra\tDP\_mode\tTotal\tHet\_0/1\tHet\_%\tHomAlt\_1/1\tHomAlt\_%\tSNVs\tMNPs\tInserciones\tDeleciones\tIndels\_cx"

for VCF in "$@"; do

[[ -f "$VCF" ]] || { echo "No existe: $VCF" >&2; exit 1; }

SAMPLE="$(basename "$VCF")"

DP\_MODE="$(echo "$SAMPLE" | sed -n 's/.\*\(dp[0-9][0-9]\).\*/\1/p')"

if [[ "$VCF" =~ \.vcf\.gz$ ]]; then R="zcat --force"; else R="cat"; fi

$R "$VCF" | awk -v SAMPLE="$SAMPLE" -v DP="$DP\_MODE" -F'\t' '

function idx(fmt,name, a,i,n){n=split(fmt,a,":"); for(i=1;i<=n;i++) if(a[i]==name) return i; return 0}

function val(sample,i, a,n){if(i<=0) return ""; n=split(sample,a,":"); return a[i]}

function type\_by\_len(ref,alt, lr,la){

lr=length(ref); la=length(alt);

if(lr==1 && la==1) return "SNV";

if(lr==la && lr>1) return "MNP";

# indels

if((lr==1 && la>1) || (lr>1 && la==1)) return (la>lr)?"INS":"DEL";

# complejos

return "INDEL\_CX"

}

/^#/ {next}

{

total++;

gt="";

split($9,F,":"); split($10,S,":");

for(i=1;i<=length(F);i++) if(F[i]=="GT"){ gt=S[i]; break }

if(gt ~ /^(0[\/|]1|1[\/|]0|0\|1|1\|0)$/){ het++ } else if(gt ~ /^(1[\/|]1|1\|1)$/){ homa++ }

# ALT puede tener múltiples; contamos por primer ALT (más simple y consistente con tus tablas)

split($5,a,","); alt=a[1];

t=type\_by\_len($4,alt);

if(t=="SNV") snv++;

else if(t=="MNP") mnp++;

else if(t=="INS") ins++;

else if(t=="DEL") del++;

else indcx++;

}

END{

hetp=(total?100\*het/total:0);

homp=(total?100\*homa/total:0);

printf "%s\t%s\t%d\t%d\t%.2f\t%d\t%.2f\t%d\t%d\t%d\t%d\t%d\n",

SAMPLE,DP,total,het,hetp,homa,homp,snv,mnp,ins,del,indcx;

}

'

done

Ejecuta:

chmod +x postfilter\_metrics.sh

./postfilter\_metrics.sh m1des.dp25.filtered.vcf m1des.dp10.filtered.vcf m2des.dp25.filtered.vcf m2des.dp10.filtered.vcf m3des.dp25.filtered.vcf m3des.dp10.filtered.vcf > postfilter\_summary.tsv

**Script6 obtener nodes y edges en muestra 1**

nano string\_network\_stats.sh

Copia y pega este contenido dentro de nano:

#!/usr/bin/env bash  
NODES=m1.nodes.tsv  
EDGES=m1\_edges\_subnet.tsv  
  
# Nº total de nodos  
N\_TOTAL\_NODES=$(($(wc -l < "$NODES") - 1))  
echo "Total nodos: $N\_TOTAL\_NODES"  
  
# Nº total de edges  
N\_TOTAL\_EDGES=$(($(wc -l < "$EDGES") - 1))  
echo "Total edges: $N\_TOTAL\_EDGES"  
  
# Nº de nodos conectados  
N\_CONNECTED\_NODES=$(wc -l < m1\_connected\_nodes.txt)  
echo "Nodos conectados: $N\_CONNECTED\_NODES"  
  
# Nodos aislados (degree=0)  
awk 'NR>1{print $1}' "$NODES" | sort > all\_nodes.list  
comm -23 all\_nodes.list m1\_connected\_nodes.txt > isolated\_nodes.list  
N\_ISOLATED=$(wc -l < isolated\_nodes.list)  
echo "Nodos aislados: $N\_ISOLATED"

Para **guardar**:

* Pulsa CTRL + O → luego Enter.  
  (te confirmará que se guardó en string\_network\_stats.sh).

Para **salir**:

* Pulsa CTRL + X.

Identificacion de hubs

awk 'NR>1{deg[$1]++; deg[$2]++} END{print "node\tdegree"; for(n in deg) print n"\t"deg[n]}' m1\_edges\_subnet.tsv \  
| sort -k2,2nr | head -n 20 > m1\_top20\_hubs.tsv

enriquecimiento funcional GO y KEGG:

awk -F'\t' '  
BEGIN{IGNORECASE=1}  
NR==1{  
 for(i=1;i<=NF;i++) h[$i]=i  
 C = h["#category"]  
 ID = h["term ID"]  
 D = h["term description"]  
 OG = h["observed gene count"]  
 BG = h["background gene count"]  
 F = h["false discovery rate"]  
 ML = h["matching proteins in your network (labels)"]  
 next  
}  
{  
 fdr = $F; gsub(",", ".", fdr)  
 if (fdr=="" || fdr+0>0.05) next  
 cat = tolower($C)  
 if (cat ~ /biological process/ || cat ~ /kegg/){  
 print fdr, $C, $ID, $D, $OG, $BG, $ML  
 }  
}  
' OFS='\t' M1.enrichment.tsv \  
| sort -t $'\t' -k1,1g \  
| awk -F'\t' '  
BEGIN{  
 IGNORECASE=1; OFS="\t";  
 print "Category","Term ID","Description","ObservedGenes","BackgroundGenes","FDR","MatchingGenes"  
}  
tolower($2) ~ /biological process/ && ++bp<=10 {print $2,$3,$4,$5,$6,$1,$7}  
tolower($2) ~ /kegg/ && ++k<=10 {print $2,$3,$4,$5,$6,$1,$7}  
' > M1\_enrichment\_top.tsv

**Script 7 obtener genes especificos de cancer ovario**

Creamos un archivo con genes\_ovario.txt. Despues

# Filtrar genes de cáncer de ovario en tu lista

grep -F -f ovarian\_cancer\_genes.txt m1des.filtered.vep.genes.symbol.HM.txt > m1\_OVCA\_hits.txt

# Opcional: ver sus variantes asociadas

awk -F'\t' 'NR==1{for(i=1;i<=NF;i++){h[$i]=i} print; next} \

NR>1 && ($h["SYMBOL"]=="BRCA1" || $h["SYMBOL"]=="BRCA2" || $h["SYMBOL"]=="TP53" || $h["SYMBOL"]=="RAD51")' \

m1des.filtered.vep.variants.HM.tsv > m1\_OVCA\_variants.tsv