

TCGA-BRCA MAF Analysis

Raman Butta

2025-10-22

Contents

Dataset access	1
MAF Summary Plot	4
Oncoplot	4
Transition-Transversion Plot	5
Lollipop plot	6
Rainfall Plot	8
TCGA Compare Plot	9
Somatic Interaction Heatmap	9
Variant Allele Frequency (VAF) Plot	12
Tumor Heterogeneity Inference	13
Oncodrive	15
PFAM Domains	17
Survival Analysis	20
Prognostic survival analysis	27
Clinical Enrichment	29
Drug-gene interactions	34
Oncogenic signaling pathways	35

Dataset access

For initial reference, refer to <https://bioconductor.org/packages/release/bioc/vignettes/maftools/inst/doc/maftools.html> and <https://github.com/PoisonAlien/maftools>.

Go to <https://portal.gdc.cancer.gov/> and click on the “Cohort Builder” tab. Select the following filters to find the breast cancer (BRCA) cohort: - Program : TCGA - Project: TCGA-BRCA

and then save the cohort by giving it a name like “My-BRCA-Cohort”.

Then go to the “Analysis Center” tab and select this cohort and download the Compressed MAF file of all samples. Then work on this maf file as below.

```
# Read the MAF file to get a MAF object (not just file path)
brca_maf <- read.maf(maf = "cohortMAF.maf.gz")
```

```
## -Reading
## -Validating
## -Silent variants: 18779
## -Summarizing
## --Possible FLAGS among top ten genes:
##   TTN
##   MUC16
##   HMCN1
##   FLG
## -Processing clinical data
## --Missing clinical data
## -Finished in 11.5s elapsed (14.7s cpu)
```

```
# Now access the unique samples
sample_count <- length(unique(brca_maf@data$Tumor_Sample_Barcode))
print("The columns are:")
```

```
## [1] "The columns are:"
```

```
colnames(brca_maf@data)
```

```
##   [1] "Hugo_Symbol"           "Entrez_Gene_Id"
##   [3] "Center"                "NCBI_Build"
##   [5] "Chromosome"            "Start_Position"
##   [7] "End_Position"          "Strand"
##   [9] "Variant_Classification" "Variant_Type"
##  [11] "Reference_Allele"      "Tumor_Seq_Allele1"
##  [13] "Tumor_Seq_Allele2"    "dbSNP_RS"
##  [15] "dbSNP_Val_Status"     "Tumor_Sample_Barcode"
##  [17] "Matched_Norm_Sample_Barcode" "Match_Norm_Seq_Allele1"
##  [19] "Match_Norm_Seq_Allele2" "Tumor_Validation_Allele1"
##  [21] "Tumor_Validation_Allele2" "Match_Norm_Validation_Allele1"
##  [23] "Match_Norm_Validation_Allele2" "Verification_Status"
##  [25] "Validation_Status"     "Mutation_Status"
##  [27] "Sequencing_Phase"      "Sequence_Source"
##  [29] "Validation_Method"     "Score"
##  [31] "BAM_File"              "Sequencer"
##  [33] "Tumor_Sample_UUID"    "Matched_Norm_Sample_UUID"
##  [35] "HGVSc"                 "HGVS_p"
##  [37] "HGVS_p_Short"          "Transcript_ID"
##  [39] "Exon_Number"           "t_depth"
##  [41] "t_ref_count"           "t_alt_count"
##  [43] "n_depth"               "n_ref_count"
##  [45] "n_alt_count"           "all_effects"
##  [47] "Allele"                "Gene"
##  [49] "Feature"               "Feature_type"
##  [51] "One_Consequence"       "Consequence"
##  [53] "cDNA_position"         "CDS_position"
##  [55] "Protein_position"      "Amino_acids"
```

```
## [57] "Codons" "Existing_variation"
## [59] "DISTANCE" "TRANSCRIPT_STRAND"
## [61] "SYMBOL" "SYMBOL_SOURCE"
## [63] "HGNC_ID" "BIOTYPE"
## [65] "CANONICAL" "CCDS"
## [67] "ENSP" "SWISSPROT"
## [69] "TREMBL" "UNIPARC"
## [71] "UNIPROT_ISOFORM" "RefSeq"
## [73] "MANE" "APPRIS"
## [75] "FLAGS" "SIFT"
## [77] "PolyPhen" "EXON"
## [79] "INTRON" "DOMAINS"
## [81] "1000G_AFR" "1000G_AFR_AF"
## [83] "1000G_AMR" "1000G_EAS_AF"
## [85] "1000G_EUR" "1000G_SAS_AF"
## [87] "ESP_AA" "ESP_EA_AF"
## [89] "gnomAD_AFR" "gnomAD_AFR_AF"
## [91] "gnomAD_AMR" "gnomAD_ASJ_AF"
## [93] "gnomAD_EAS" "gnomAD_FIN_AF"
## [95] "gnomAD_NFE" "gnomAD_OTH_AF"
## [97] "gnomAD_SAS" "MAX_AF"
## [99] "MAX_AF_POPS" "gnomAD_non_cancer_AF"
## [101] "gnomAD_non_cancer_AFR_AF" "gnomAD_non_cancer_AMI_AF"
## [103] "gnomAD_non_cancer_AMR_AF" "gnomAD_non_cancer_ASJ_AF"
## [105] "gnomAD_non_cancer_EAS_AF" "gnomAD_non_cancer_FIN_AF"
## [107] "gnomAD_non_cancer_MID_AF" "gnomAD_non_cancer_NFE_AF"
## [109] "gnomAD_non_cancer_OTH_AF" "gnomAD_non_cancer_SAS_AF"
## [111] "gnomAD_non_cancer_MAX_AF_adj" "gnomAD_non_cancer_MAX_AF_POPS_adj"
## [113] "CLIN_SIG" "SOMATIC"
## [115] "PUBMED" "TRANSCRIPTION_FACTORS"
## [117] "MOTIF_NAME" "MOTIF_POS"
## [119] "HIGH_INF_POS" "MOTIF_SCORE_CHANGE"
## [121] "miRNA" "IMPACT"
## [123] "PICK" "VARIANT_CLASS"
## [125] "TSL" "HGVS_OFFSET"
## [127] "PHENO" "GENE_PHENO"
## [129] "CONTEXT" "case_id"
## [131] "GDC_FILTER" "COSMIC"
## [133] "hotspot" "tumor_bam_uuid"
## [135] "normal_bam_uuid" "RNA_Support"
## [137] "RNA_depth" "RNA_ref_count"
## [139] "RNA_alt_count" "callers"
```

```
print(paste("Sample count is", sample_count))
```

```
## [1] "Sample count is 800"
```

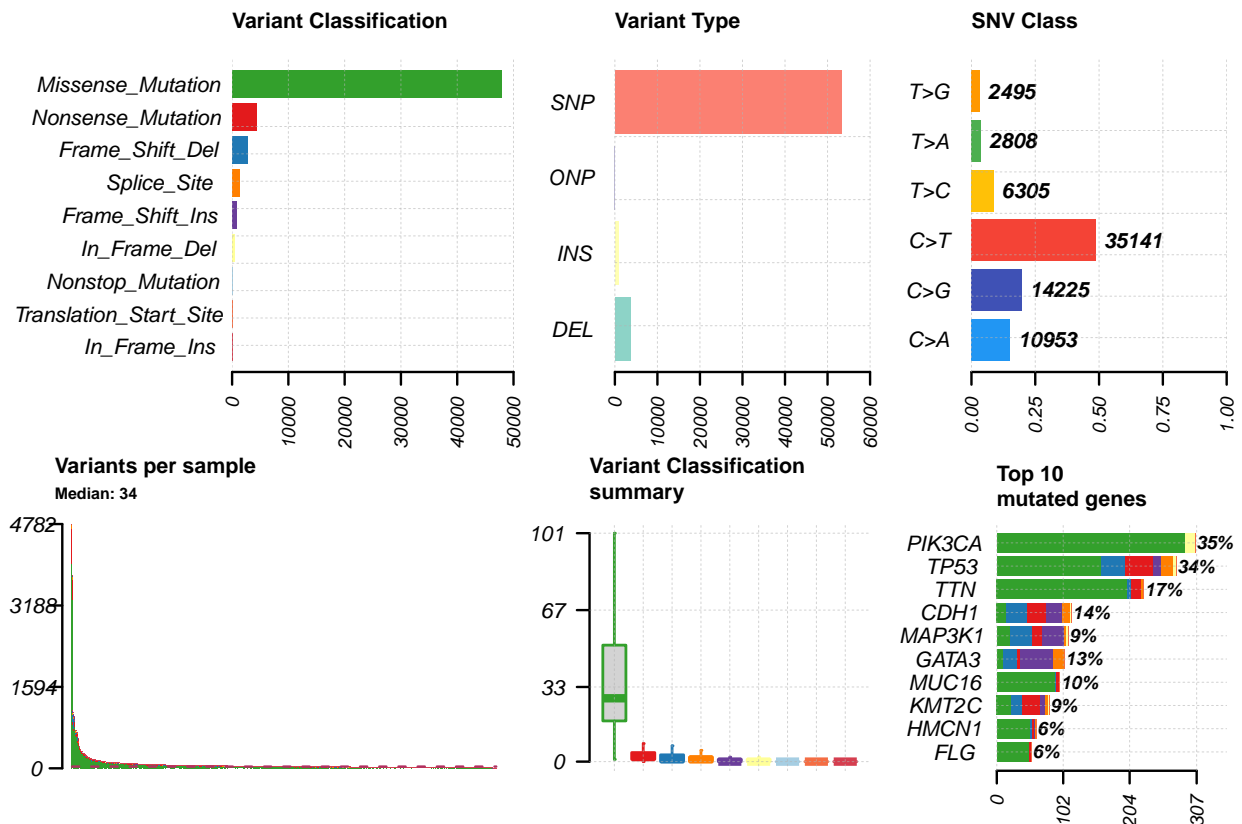
```
# Print summary of the MAF object
print(brca_maf)
```

```
## An object of class MAF
##           ID summary   Mean Median
##           <char> <char> <num>  <num>
```

## 1:	NCBI_Build	GRCh38	NA	NA
## 2:	Center	WUGSC	NA	NA
## 3:	Samples	800	NA	NA
## 4:	nGenes	14867	NA	NA
## 5:	Frame_Shift_Del	2844	3.555	1
## 6:	Frame_Shift_Ins	780	0.975	0
## 7:	In_Frame_Del	517	0.646	0
## 8:	In_Frame_Ins	35	0.044	0
## 9:	Missense_Mutation	47794	59.742	28
## 10:	Nonsense_Mutation	4315	5.394	2
## 11:	Nonstop_Mutation	68	0.085	0
## 12:	Splice_Site	1289	1.611	1
## 13:	Translation_Start_Site	61	0.076	0
## 14:	total	57703	72.129	34

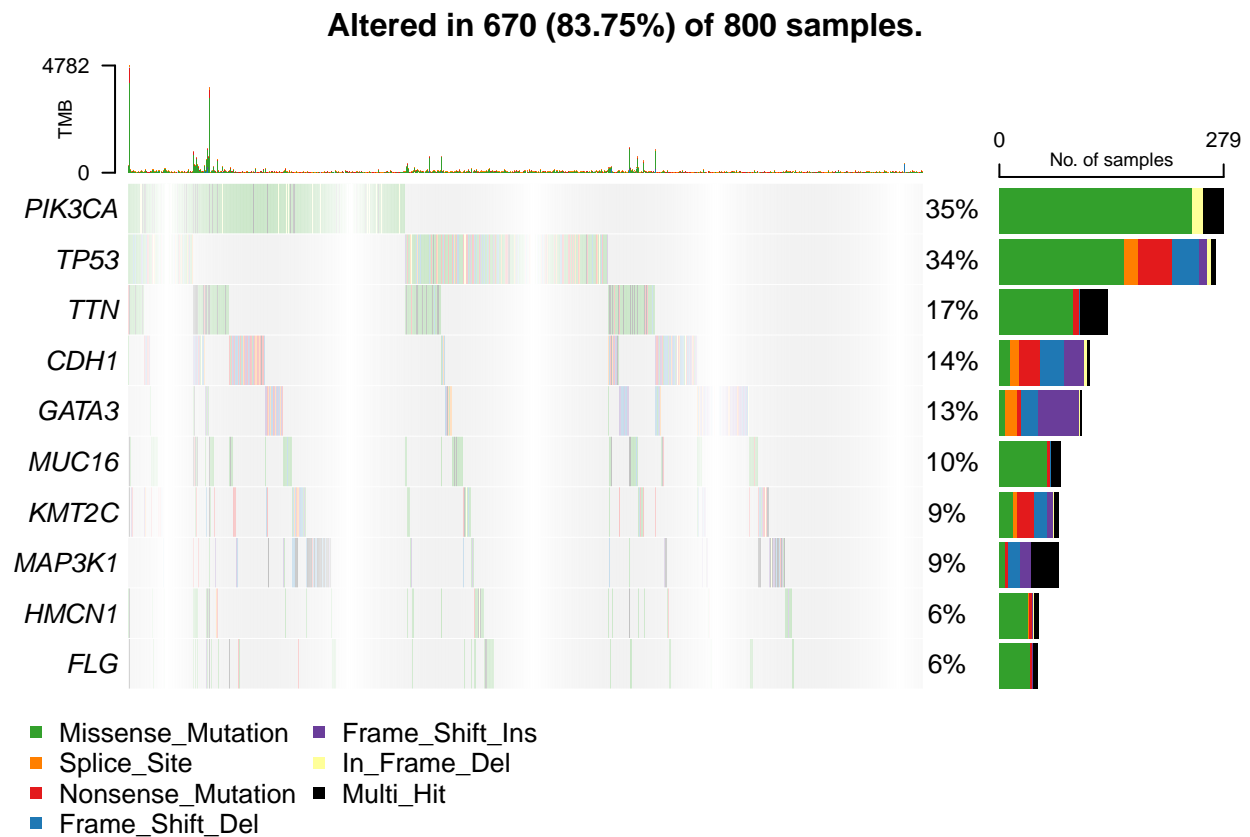
MAF Summary Plot

```
plotmafSummary(maf = brca_maf, rmOutlier = TRUE, addStat = 'median', dashboard = TRUE, titvRaw = FALSE)
```



Oncoplot

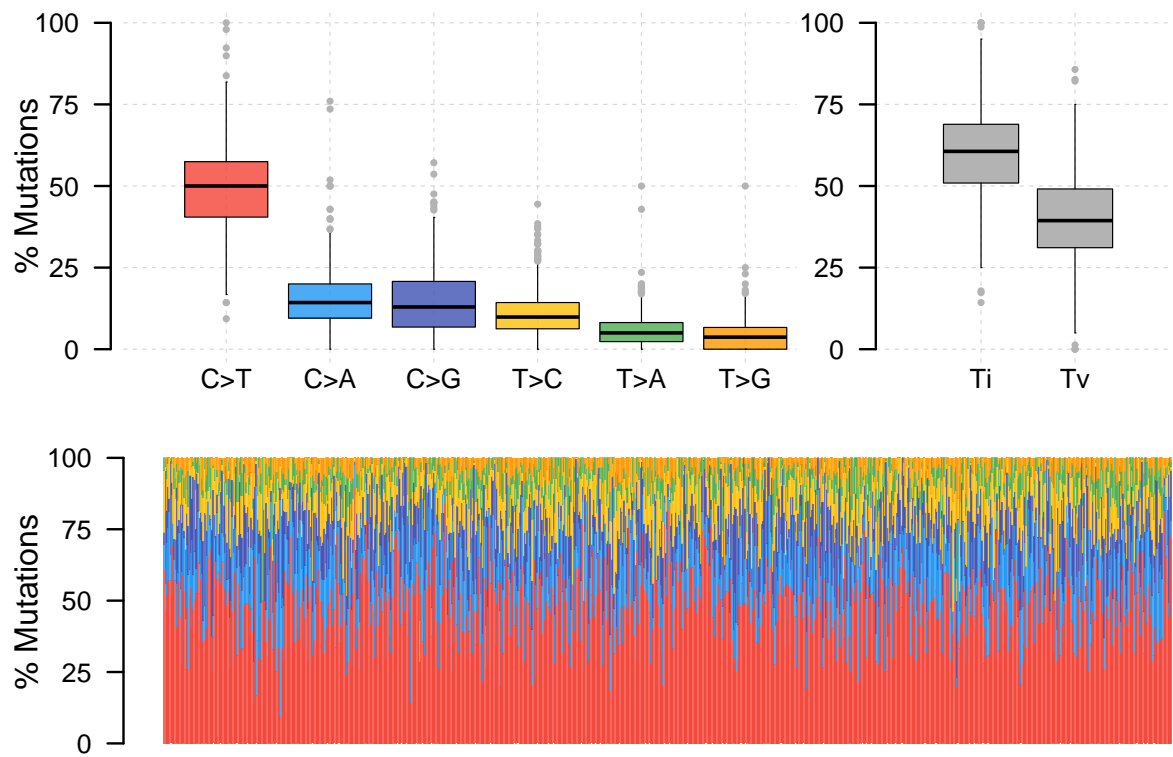
```
#oncoplot for top ten mutated genes.
oncoplot(maf = brca_maf, top = 10)
```



Transition-Transversion Plot

Classifies mutations (SNPs) into six categories based on the type of nucleotide change: - Transitions (Ti): Purine to Purine (A G) or Pyrimidine to Pyrimidine (C T) - Transversions (Tv): Purine to Pyrimidine or vice versa (A C, A T, G C, G T)

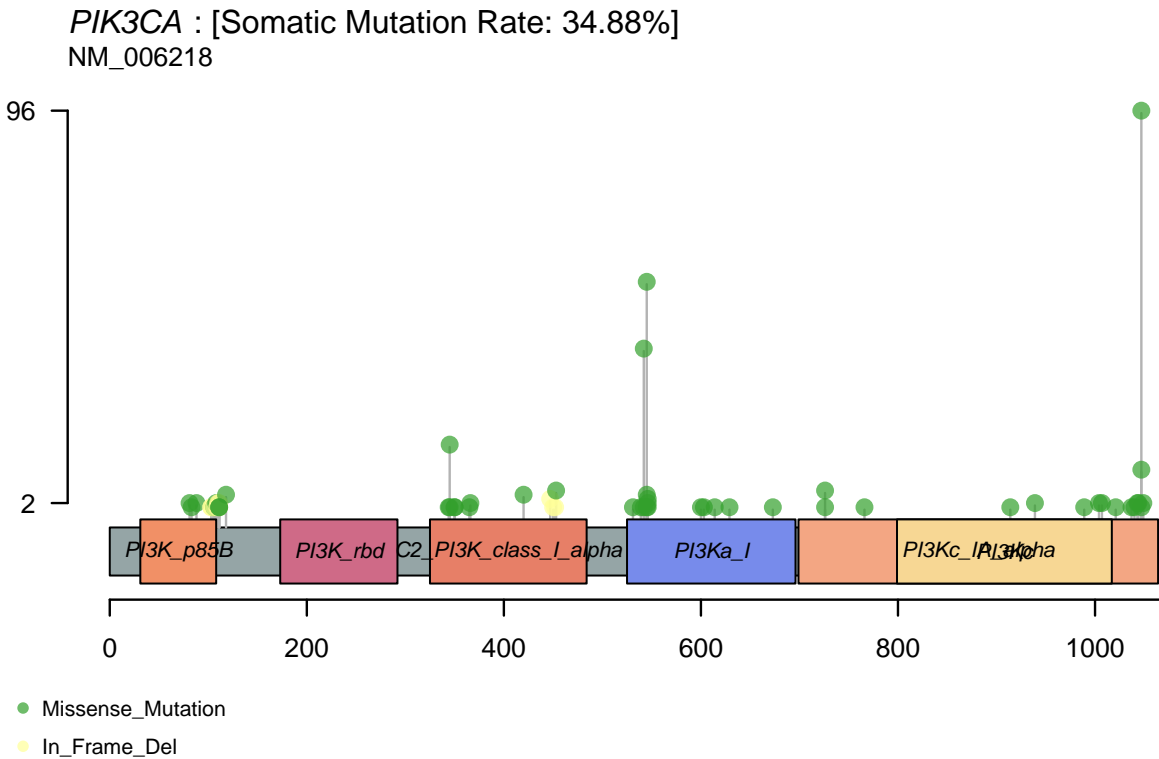
```
brca_maf.titv = titv(maf = brca_maf, plot = FALSE, useSyn = TRUE)
#plot titv summary
plotTiTv(res = brca_maf.titv)
```



Lollipop plot

Let's plot the lollipop plot of the most critical oncogene we found in breast cancer : PIK3CA

```
lollipopPlot(maf = brca_maf, gene = 'PIK3CA', AACol = 'HGVS_Short', showMutationRate = TRUE)
```



The somatic mutation rate is 34.88%, meaning nearly 35% of samples have at least one somatic mutation in this gene. NM_006218 refers to a specific mRNA isoform of PIK3CA. The X-Axis ranges from 0 to ~1000+ indicating amino acid positions in the corresponding protein. The protein is divided into distinct colored functional domains. Mutations in these domains can dysregulate PI3K pathway activity, contributing to cancer development.

The lollipops (green dots) represent missense mutations, which change a single amino acid. The rare yellow lollipops indicate in-frame deletions denoting deletion of one or more amino acids without disrupting the reading frame. The positions of lollipops along the x-axis correspond to the amino acid position in the protein sequence. The height of a lollipop represents the number of samples (or patients) within the cohort in which a mutation was observed at that specific amino acid position. The clustering of mutations in certain regions (e.g., around position 1000) suggests these are recurrent mutation “hotspots” seen across several unrelated patients. This happens because somatic cell cancers like BRCA are an extreme case of intra-generational evolution of somatic cells within an organism. For example, the same PIK3CA H1047R mutation arises independently in different patients because it’s the optimal solution to activate the P13K pathway and multiply that cell type.

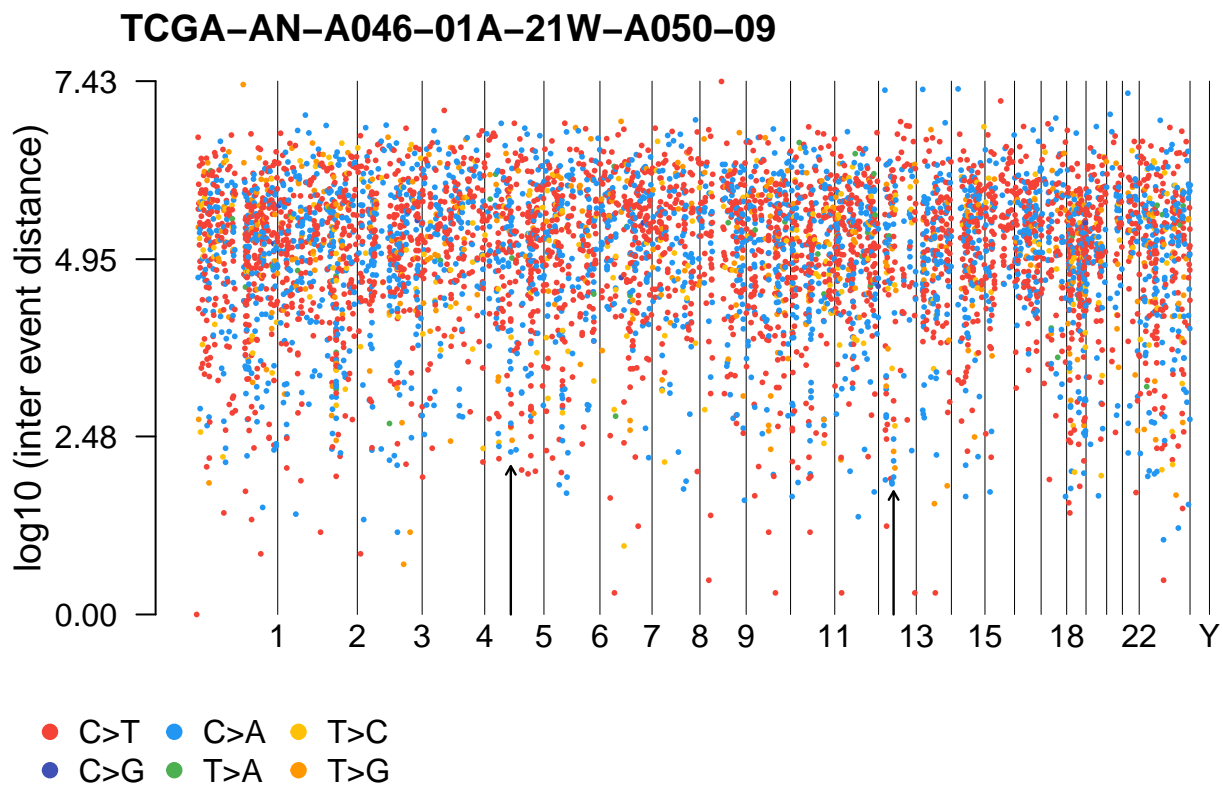
In fact, the famous H1047R hotspot mutation (histidine → arginine at position 1047) sits within the kinase domain (yellow color) of this gene. It is one of the rare Driver mutations that directly increases a cell’s fitness, i.e. its ability to survive, divide, and outcompete neighbors in the tissue ecosystem. From theory we know that mutations in the kinase domain lead to constitutive activation of the PI3K pathway, promoting uncontrolled cell growth and survival, which is great for selective pressure in evolution of such somatic cells within an organism. That’s why such mutations pop up de novo in later life in so many unrelated patients due to “convergent evolution” at the cellular level, not due to any inherited changes from germ-line cells.

In fact, >90% of oncogenic PIK3CA mutations cluster in just two hotspots: - Helical domain: E542K, E545K - Kinase domain: H1047R/L Such mutation hotspots serve as actionable drug targets. Drugs like alpelisib (a PI3K inhibitor) are approved for PIK3CA-mutant breast cancer.

Rainfall Plot

```
rainfallPlot(maf = brca_maf, detectChangePoints = TRUE, pointSize = 0.4)
```

##	Chromosome	Start_Position	End_Position	nMuts	Avg_intermutation_dist	Size
##	<num>	<num>	<num>	<int>	<num>	<num>
## 1:	5	79736501	79739125	6	524.8000	2624
## 2:	13	45967816	45969966	7	358.3333	2150
##	Tumor_Sample_Barcode	C>A	C>T	T>G		
##	<fctr>	<int>	<int>	<int>		
## 1:	TCGA-AN-A046-01A-21W-A050-09	3	2	1		
## 2:	TCGA-AN-A046-01A-21W-A050-09	3	3	1		



Rainfall plots are genomic maps of somatic mutations that display the inter-mutation distance on a log scale. Each point represents a mutation, and the y-axis shows the distance to the previous mutation in base pairs. The x-axis represents the genomic position across chromosomes.

Why do mutations cluster or spread out?

- **Clustered mutations** → suggest simultaneous damage (e.g., UV exposure causing adjacent pyrimidine dimers → C>T at dipyrimidine sites)
- **Evenly spaced mutations** → suggest random accumulation over time (e.g., aging-related replication errors)

So the pattern of mutation spacing tells us about the underlying mutational process.

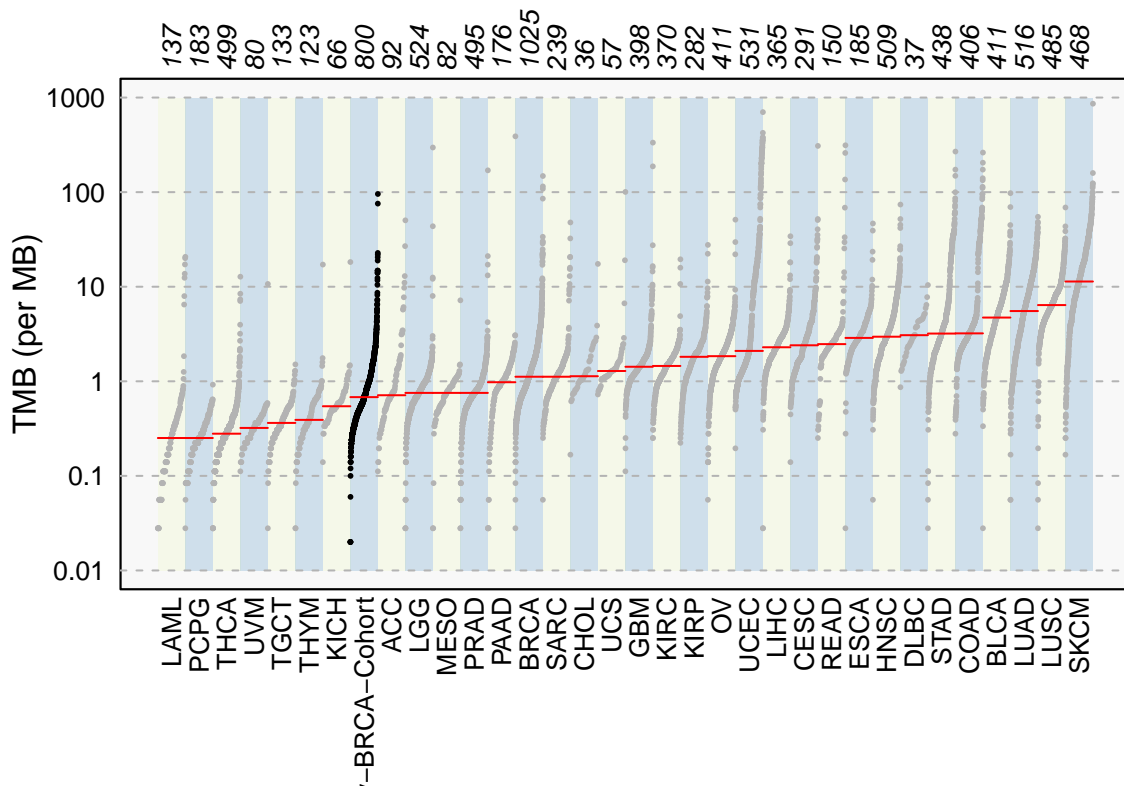
In the above rainfall plot, Chromosomes have different physical lengths (in base pairs), and the x-axis is scaled to reflect that (e.g. Chr 1 is the largest with 249 MB bases (over 4x larger than chromosome 19 (~59 Mb)).

Rainfall plots give us a visual idea of the **mutation density**, i.e. no. of mutations/length of chromosome. We observe there are 2 nos. kataegis (localized hypermutation) events on Chr 5 and Chr 13.

TCGA Compare Plot

Now let's compare the mutation load with 33 nos. TCGA cohorts representing different primary cancer types.

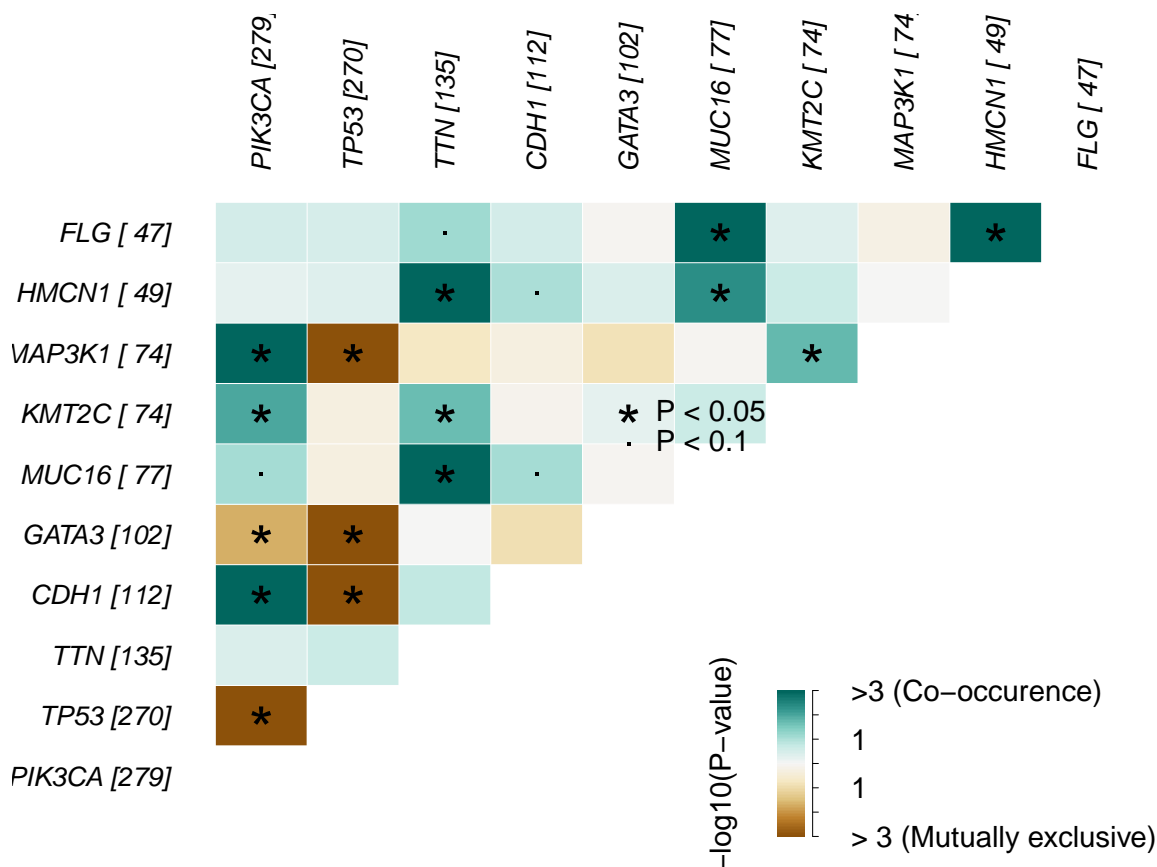
```
brca_mutload <- tcgaCompare(maf = brca_maf,
                             cohortName = "My-BRCA-Cohort",
                             logscale = TRUE,
                             capture_size = 50)
```



Our cohort is marked in black dots. The median is below 1 TMB/MB in log scale. But there are few outlier samples with surprisingly high mutation load. This could be due to conditionally hypermutated samples or samples with microsatellite instability (MSI), which is a known characteristic of BRCA-mutated cancers.

Somatic Interaction Heatmap

```
somaticInteractions(maf = brca_maf, top = 10, pvalue = c(0.05, 0.1))
```



##	gene1	gene2	pValue	oddsRatio	00	01	11	10	pAdj
##	<char>	<char>	<num>	<num>	<int>	<int>	<int>	<int>	<num>
##	1: GATA3	TP53	1.070333e-10	0.1418739	436	262	8	94	8.919439e-10
##	2: CDH1	TP53	6.274503e-10	0.1806917	429	259	11	101	4.481788e-09
##	3: MAP3K1	TP53	5.804712e-07	0.1842118	463	263	7	67	3.627945e-06
##	4: TTN	HMCN1	8.204052e-06	4.1802687	637	28	21	114	4.557807e-05
##	5: TP53	PIK3CA	1.037918e-05	0.4819216	317	213	66	204	5.189590e-05
##	6: MUC16	TTN	3.954483e-05	3.0693475	615	108	27	50	1.797492e-04
##	7: FLG	MUC16	4.738847e-05	4.6317773	690	63	14	33	1.867357e-04
##	8: HMCN1	FLG	4.855128e-05	5.7255888	715	36	11	38	1.867357e-04
##	9: PIK3CA	MAP3K1	1.721452e-04	2.5442674	488	33	41	238	6.148044e-04
##	10: PIK3CA	CDH1	8.807647e-04	1.9969411	464	57	55	224	2.935882e-03
##	11: MUC16	HMCN1	4.549189e-03	2.9982509	685	38	11	66	1.421622e-02
##	12: PIK3CA	KMT2C	1.050372e-02	1.8815171	483	38	36	243	3.089330e-02
##	13: MAP3K1	KMT2C	1.798308e-02	2.3200263	665	61	13	61	4.995300e-02
##	14: PIK3CA	GATA3	1.944301e-02	0.5679185	444	77	25	254	5.116580e-02
##	15: TTN	KMT2C	2.142333e-02	1.9658638	611	54	20	115	5.355832e-02
##	16: TTN	FLG	6.751260e-02	1.9756082	631	34	13	122	1.607443e-01
##	17: MUC16	PIK3CA	7.876493e-02	1.5418468	478	245	34	43	1.790112e-01
##	18: MUC16	CDH1	8.309911e-02	1.7117735	627	96	16	61	1.806502e-01
##	19: HMCN1	CDH1	8.855052e-02	1.8611033	650	101	11	38	1.844803e-01
##	20: GATA3	CDH1	1.263152e-01	0.5594016	595	103	9	93	2.526305e-01

## 21:	MAP3K1	GATA3	1.410219e-01	0.4702528	629	97	5	69	2.711960e-01
## 22:	TTN	CDH1	1.741340e-01	1.4170192	577	88	24	111	3.224703e-01
## 23:	TTN	MAP3K1	1.910219e-01	0.5720611	599	66	8	127	3.411105e-01
## 24:	KMT2C	HMCN1	2.037977e-01	1.7001275	684	42	7	67	3.513753e-01
## 25:	MUC16	KMT2C	2.187759e-01	1.5358921	659	64	10	67	3.646265e-01
## 26:	TTN	TP53	2.308067e-01	1.2841771	447	218	52	83	3.722688e-01
## 27:	FLG	PIK3CA	2.713198e-01	1.4122651	494	259	20	27	4.239372e-01
## 28:	FLG	CDH1	2.814472e-01	1.4937816	650	103	9	38	4.264351e-01
## 29:	TP53	FLG	3.415095e-01	1.3566598	502	28	19	251	5.022199e-01
## 30:	TTN	PIK3CA	3.727199e-01	1.2085450	438	227	52	83	5.324571e-01
## 31:	HMCN1	GATA3	3.847554e-01	1.3631916	657	94	8	41	5.343825e-01
## 32:	FLG	KMT2C	4.310410e-01	1.4733483	685	68	6	41	5.783904e-01
## 33:	HMCN1	TP53	4.395767e-01	1.2612111	500	251	19	30	5.783904e-01
## 34:	MAP3K1	CDH1	4.845691e-01	0.7252106	622	104	8	66	6.212424e-01
## 35:	KMT2C	TP53	5.191679e-01	0.8156738	478	248	22	52	6.420916e-01
## 36:	MUC16	TP53	5.265151e-01	0.8210356	476	247	23	54	6.420916e-01
## 37:	HMCN1	PIK3CA	5.405263e-01	1.1957375	491	260	19	30	6.434837e-01
## 38:	KMT2C	GATA3	5.825416e-01	1.2180653	635	91	11	63	6.773739e-01
## 39:	FLG	MAP3K1	6.116695e-01	0.6552276	682	71	3	44	6.950790e-01
## 40:	KMT2C	CDH1	7.273778e-01	0.8376679	623	103	9	65	8.081976e-01
## 41:	FLG	GATA3	8.227895e-01	0.8053092	656	97	5	42	8.893589e-01
## 42:	MAP3K1	MUC16	8.359974e-01	0.8141978	655	71	6	68	8.893589e-01
## 43:	GATA3	MUC16	8.591346e-01	0.8967031	630	68	9	93	8.949319e-01
## 44:	GATA3	TTN	1.000000e+00	0.9830676	580	118	17	85	1.000000e+00
## 45:	HMCN1	MAP3K1	1.000000e+00	0.8649371	681	70	4	45	1.000000e+00
##	gene1	gene2	pValue	oddsRatio	00	01	11	10	pAdj
##		Event		pair	event_ratio				
##		<char>		<char>	<char>				
## 1:	Mutually_Exclusive	GATA3, TP53			8/356				
## 2:	Mutually_Exclusive	CDH1, TP53			11/360				
## 3:	Mutually_Exclusive	MAP3K1, TP53			7/330				
## 4:	Co_Occurrence	HMCN1, TTN			21/142				
## 5:	Mutually_Exclusive	PIK3CA, TP53			66/417				
## 6:	Co_Occurrence	MUC16, TTN			27/158				
## 7:	Co_Occurrence	FLG, MUC16			14/96				
## 8:	Co_Occurrence	FLG, HMCN1			11/74				
## 9:	Co_Occurrence	MAP3K1, PIK3CA			41/271				
## 10:	Co_Occurrence	CDH1, PIK3CA			55/281				
## 11:	Co_Occurrence	HMCN1, MUC16			11/104				
## 12:	Co_Occurrence	KMT2C, PIK3CA			36/281				
## 13:	Co_Occurrence	KMT2C, MAP3K1			13/122				
## 14:	Mutually_Exclusive	GATA3, PIK3CA			25/331				
## 15:	Co_Occurrence	KMT2C, TTN			20/169				
## 16:	Co_Occurrence	FLG, TTN			13/156				
## 17:	Co_Occurrence	MUC16, PIK3CA			34/288				
## 18:	Co_Occurrence	CDH1, MUC16			16/157				
## 19:	Co_Occurrence	CDH1, HMCN1			11/139				
## 20:	Mutually_Exclusive	CDH1, GATA3			9/196				
## 21:	Mutually_Exclusive	GATA3, MAP3K1			5/166				
## 22:	Co_Occurrence	CDH1, TTN			24/199				
## 23:	Mutually_Exclusive	MAP3K1, TTN			8/193				
## 24:	Co_Occurrence	HMCN1, KMT2C			7/109				
## 25:	Co_Occurrence	KMT2C, MUC16			10/131				
## 26:	Co_Occurrence	TP53, TTN			52/301				

## 27:	Co_Occurence	FLG, PIK3CA	20/286
## 28:	Co_Occurence	CDH1, FLG	9/141
## 29:	Co_Occurence	FLG, TP53	19/279
## 30:	Co_Occurence	PIK3CA, TTN	52/310
## 31:	Co_Occurence	GATA3, HMCN1	8/135
## 32:	Co_Occurence	FLG, KMT2C	6/109
## 33:	Co_Occurence	HMCN1, TP53	19/281
## 34:	Mutually_Exclusive	CDH1, MAP3K1	8/170
## 35:	Mutually_Exclusive	KMT2C, TP53	22/300
## 36:	Mutually_Exclusive	MUC16, TP53	23/301
## 37:	Co_Occurence	HMCN1, PIK3CA	19/290
## 38:	Co_Occurence	GATA3, KMT2C	11/154
## 39:	Mutually_Exclusive	FLG, MAP3K1	3/115
## 40:	Mutually_Exclusive	CDH1, KMT2C	9/168
## 41:	Mutually_Exclusive	FLG, GATA3	5/139
## 42:	Mutually_Exclusive	MAP3K1, MUC16	6/139
## 43:	Mutually_Exclusive	GATA3, MUC16	9/161
## 44:	Mutually_Exclusive	GATA3, TTN	17/203
## 45:	Mutually_Exclusive	HMCN1, MAP3K1	4/115
##	Event	pair event_ratio	

This somatic interaction heatmap shows which pairs of genes tend to be mutated together (co-occurrence) or apart (mutual exclusivity) across breast cancer patients.

- Green = genes that mutate together more than expected → possible cooperation or shared pathway.
- Orange = genes that rarely mutate together → possible redundancy or synthetic lethality.
- Stars = statistically significant interactions.
- Numbers in brackets = how many patients had mutations in that gene.

This reveals the evolutionary logic of cancer: tumors don't just collect random mutations — they select combinations that enhance survival and growth. Understanding these patterns helps identify driver pathways, predict behavior, and design therapies

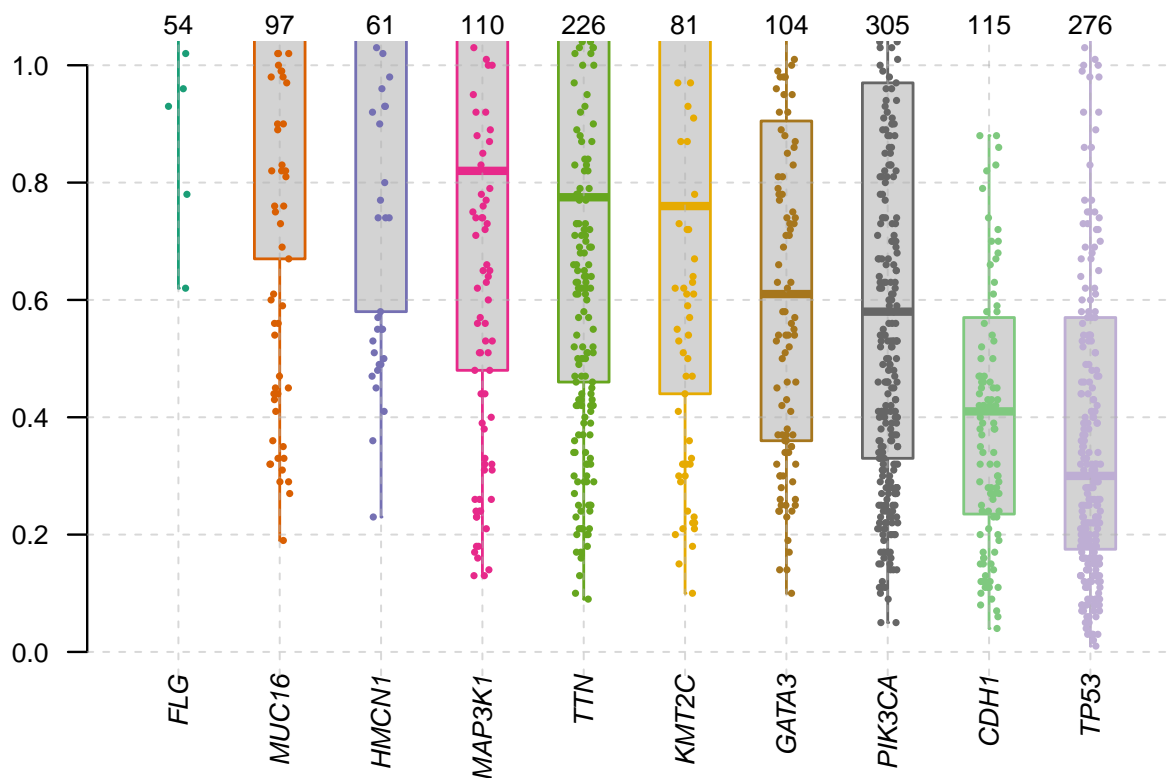
If two genes are frequently mutated together (co-occur), they likely cooperate to drive cancer — either in parallel pathways or as complementary hits in the same biological program.

If two genes are mutually exclusive (rarely mutated together), they likely act in the same essential pathway. Disrupting either one is enough to confer a fitness advantage — so tumors “choose” one or the other, but not both. E.g. TP53 and CDH1. Thus we'll never see a tumor with both mutated. Moreover we'll design a drug to target the shared pathway, not the individual gene.

Creating gene families also helps us classify cancer subtypes, e.g. PAM50 which is a 50-gene signature that classifies breast cancer into five molecular intrinsic subtypes: Luminal A, Luminal B, HER2-enriched, Basal-like, and Normal-like. It is a gene expression-based assay, such as the Prosigna test, that measures the activity of these specific genes within a tumor sample to determine its subtype, helping to guide treatment decisions and predict a patient's risk of recurrence.

Variant Allele Frequency (VAF) Plot

```
plotVaf(maf = brca_maf, vafCol = 't_ref_count')
```



The above is the VAF plot of the most significant genes in our cohort. The Variant Allele Frequency represents the proportion of reads showing mutation, i.e. what fraction of cells carry a certain mutation. We get the following information from it.

- Clonality: High VAF \rightarrow early, clonal mutation; Low VAF \rightarrow late, subclonal.
- Copy number changes: VAF $> 0.5 \rightarrow$ possible loss of wild-type allele (LOH) in many cells, and the remaining allele (mutated one) becomes the only one present at that locus.
- Driver vs. passenger: Driver genes (e.g., PIK3CA) show consistent, high VAFs; passengers show variable, often low VAFs.
- Tumor evolution: The spread of VAFs reveals the branching structure of the tumor — who came first, who came later.

Thus VAF acts like a molecular clock, helping us reconstruct the evolutionary history of the tumor and identify key driver events.

Tumor Heterogeneity Inference

Rather than analyzing the VAF of the entire cohort, we can focus on one sample to infer its tumor heterogeneity. This is done using the `inferHeterogeneity()` function.

```
# Infer tumor heterogeneity for one sample

if(!requireNamespace("mclust", quietly=TRUE))
  install.packages("mclust")
library(mclust)
```

```

# Pick one sample to analyze (replace with your sample ID)
sample_id <- "TCGA-B6-A0WW-01A-11D-A10G-09" # ← CHANGE THIS TO A SAMPLE IN YOUR DATASET from Tumor_Sam

# Infer heterogeneity
brca_sample_het <- inferHeterogeneity(
  maf = brca_maf,
  tsb = sample_id,
  vafCol = "t_ref_count"
)

# Print cluster means
print(brca_sample_het$clusterMeans)

```

```

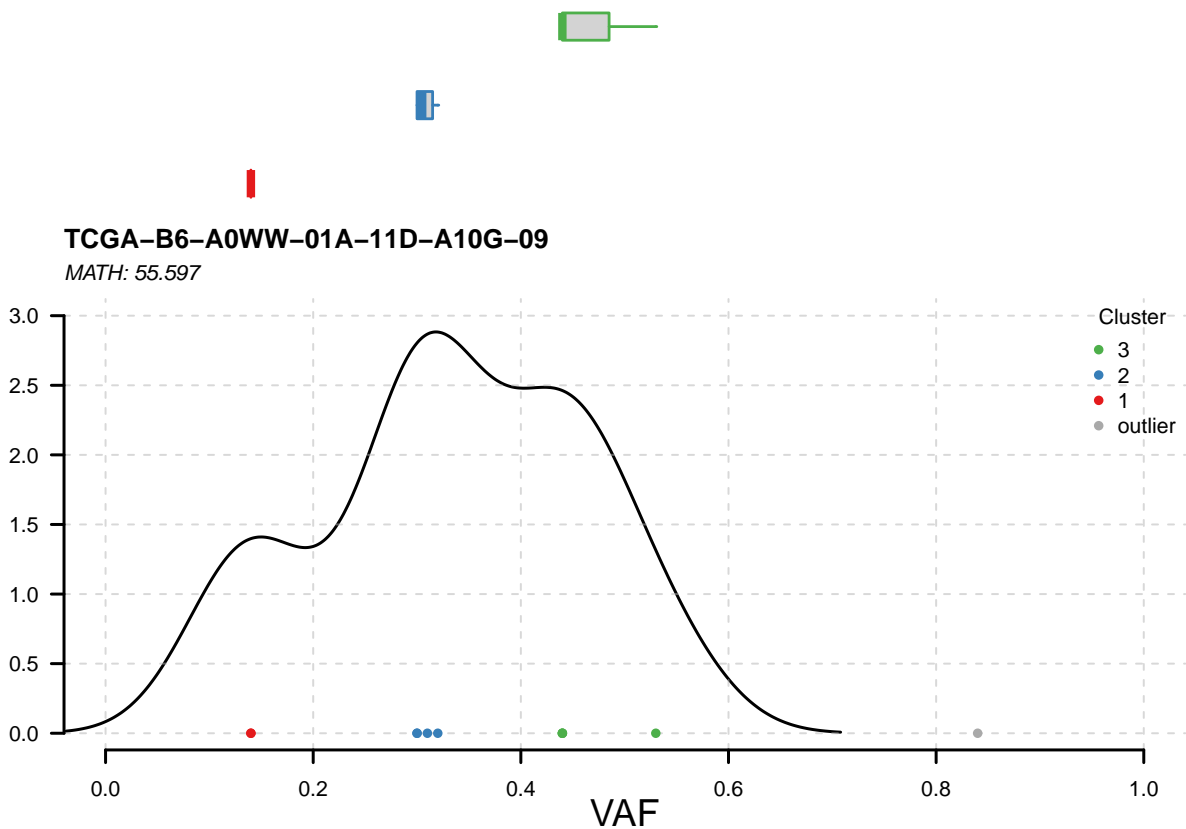
##           Tumor_Sample_Barcode cluster meanVaf
##           <fctr>   <char>   <num>
## 1: TCGA-B6-A0WW-01A-11D-A10G-09      3  0.4625
## 2: TCGA-B6-A0WW-01A-11D-A10G-09      2  0.3075
## 3: TCGA-B6-A0WW-01A-11D-A10G-09      1  0.1400
## 4: TCGA-B6-A0WW-01A-11D-A10G-09 outlier 0.8400

```

```

# Plot clusters with MATH score
plotClusters(clusters = brca_sample_het)

```



This is a density plot + scatter plot of VAFs (Variant Allele Frequencies) for one breast cancer sample TCGA-B6-A0WW-01A-11D-A10G-09.

The colored dots are the individual mutations, grouped into clusters by mclust. Each cluster represents a subpopulation of tumor cells sharing similar VAFs, indicating they likely arose from a common ancestor cell during tumor evolution.

The black line is the kernel density curve — shows where most mutations pile up. The height at any point depicts the density of clustering of the dots underneath it.

We can see 3 clusters of which the 2nd cluster (red) has the highest density peak. This means most mutations in this tumor sample belong to this subclone, which likely represents the dominant tumor cell population.

Further, - MATH > 20 = high heterogeneity

- MATH > 40 = very heterogeneous - MATH > 50 = extreme — often associated with poor prognosis, resistance, metastasis

Thus a MATH score of 55.597 here indicates a highly heterogeneous cancer. So this isn't a single cancer but more like an inter-generational family of clones of sub-cancers :

- A founder population (Cluster 3) started things — maybe with a PIK3CA or TP53 mutation.
- Then a second wave (Cluster 2) branched off — perhaps gaining growth advantage.
- Then a third, tiny group (Cluster 1) emerged — possibly resistant or metastatic.

Oncodrive

Our purpose here is to identify potential driver genes in our breast cancer cohort.

From lollipop plots we remember that some genes have mutations clustered in specific protein domains (e.g., PIK3CA). This suggests these genes are under positive selection in cancer, i.e., they are driver genes. The fraction of clustered mutations against the total no. of mutations in that gene is thus a measure of how likely it is a driver gene.

`oncodrive()` function identifies the driver genes in the MAF using such a logic. In our cohort dataset, `HGVSp_Short` is the column that contains the protein level mutations (e.g., p.V600E). We set a minimum mutation threshold of 5 to focus on genes with sufficient mutation data for statistical analysis. The p-value method is set to 'zscore' to identify genes with mutation patterns significantly deviating from random distribution, indicating potential driver genes.

```
brca_sig = oncodrive(maf = brca_maf, AACol = 'HGVSp_Short', minMut = 5, pvalMethod = 'zscore')
```

```
##      |
##      |
```

```
head(brca_sig)
```

```
##      Hugo_Symbol Frame_Shift_Del Frame_Shift_Ins In_Frame_Del In_Frame_Ins
##      <char>          <int>          <int>          <int>          <int>
## 1:      NDUFS1             0             0             0             0
## 2:      RPL22              6             0             0             0
## 3:      AKT1               0             0             0             0
## 4:      KRAS              0             0             0             0
```

```
## 5:      PIK3CA      0      0      15      0
## 6:      PSIP1      1      1      0      0
##      Missense_Mutation Nonsense_Mutation Nonstop_Mutation Splice_Site
##      <int>      <int>      <int>      <int>
## 1:      9      0      0      0
## 2:      0      0      0      0
## 3:     19      0      0      0
## 4:      6      0      0      0
## 5:    290      0      0      0
## 6:      1      0      0      2
##      Translation_Start_Site total MutatedSamples AlteredSamples clusters
##      <int> <num>      <int>      <int>      <int>
## 1:      0      9      9      9      1
## 2:      0      6      6      6      1
## 3:      0     19     19     19      1
## 4:      0      6      6      6      1
## 5:      0    305    279    279      9
## 6:      0      5      5      5      1
##      muts_in_clusters clusterScores protLen      zscore      pval      fdr
##      <int>      <num>      <int>      <num>      <num>      <num>
## 1:      9      1.0000000      741 5.546154 1.460110e-08 4.117509e-06
## 2:      6      1.0000000      128 5.546154 1.460110e-08 4.117509e-06
## 3:     17      0.8947368      480 4.736437 1.087540e-06 2.044576e-04
## 4:      5      0.8333333      189 4.264103 1.003536e-05 1.414986e-03
## 5:    278      0.7781751     1068 3.839809 6.156516e-05 6.944550e-03
## 6:      2      0.6666667      530 2.982051 1.431620e-03 1.153477e-01
##      fract_muts_in_clusters
##      <num>
## 1:      1.0000000
## 2:      1.0000000
## 3:      0.8947368
## 4:      0.8333333
## 5:      0.9114754
## 6:      0.4000000
```

We can see that genes like `NDUFS1` and `RPL22` are significant driver genes with `fract_muts_in_clusters` = 1, which means that all mutations perfectly non-randomly clustered in hotspots.

We can plot the result using the `ggplot2` function.

We set a false discovery rate (FDR) cutoff of 0.1 to highlight genes with statistically significant clustering of mutations after correcting for multiple testing.

```
library(ggplot2)

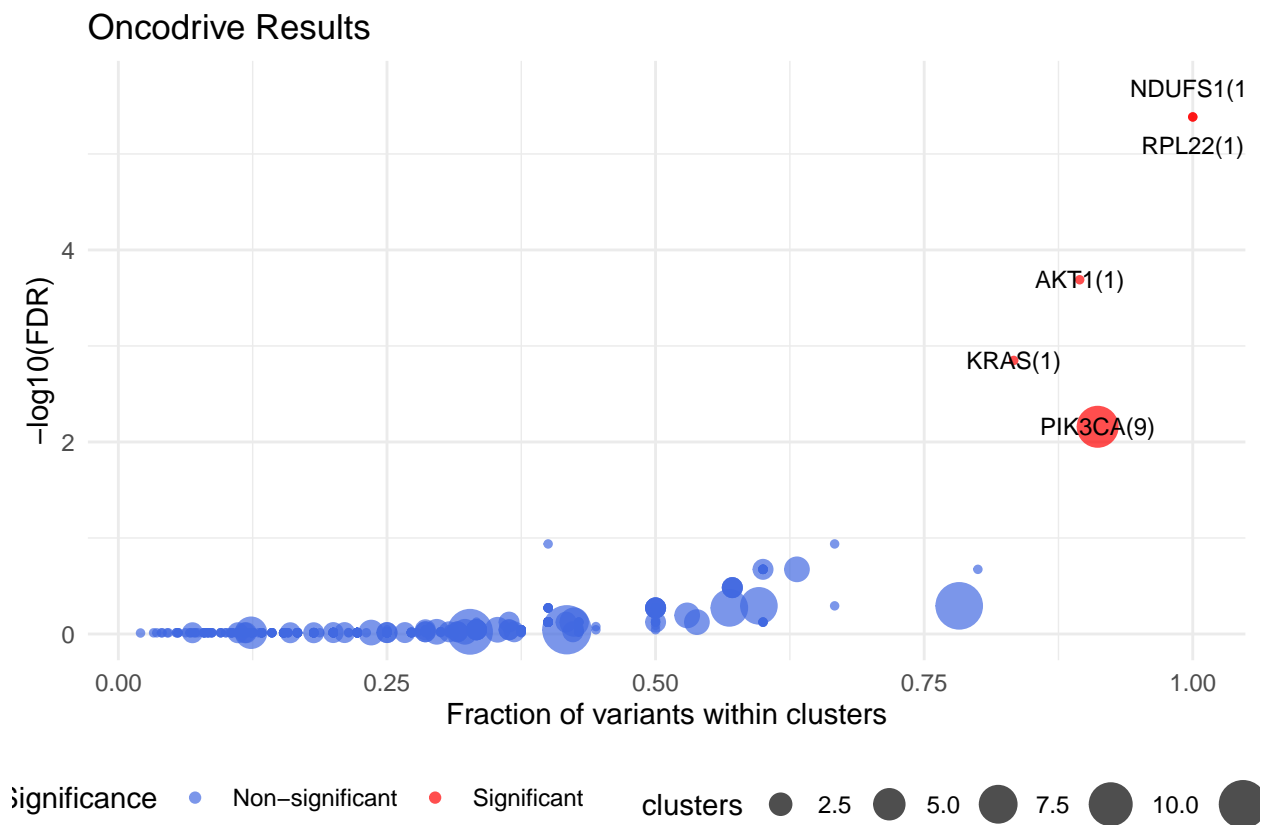
# Add labels to your results (if not already)
brca_sig$label <- paste0(brca_sig$Hugo_Symbol, "(", brca_sig$clusters, ")")

# Filter significant genes
sig_genes <- brca_sig[brca_sig$fdr < 0.1, ]

# Quick fix: Manually nudge y-position for overlapping NDUFS1 & RPL22
# (Adjust +0.3 offset if needed; assumes they're close in y ~2-3 from your plot)
sig_genes$label_y <- -log10(sig_genes$fdr)
sig_genes$label_y[sig_genes$Hugo_Symbol == "NDUFS1"] <- sig_genes$label_y[sig_genes$Hugo_Symbol == "NDUFS1"] + 0.3
```

```
sig_genes$label_y[sig_genes$Hugo_Symbol == "RPL22"] <- sig_genes$label_y[sig_genes$Hugo_Symbol == "RPL22"]

# Create the plot (points unchanged)
ggplot(brca_sig, aes(x = fract_muts_in_clusters, y = -log10(fdr))) +
  geom_point(aes(size = clusters, color = ifelse(fdr < 0.1, "Significant", "Non-significant")), alpha = 0.5) +
  scale_color_manual(values = c("Significant" = "red", "Non-significant" = "royalblue")) +
  scale_size_continuous(range = c(1, 8)) +
  # Labels with nudged y for overlaps
  geom_text(data = sig_genes, aes(x = fract_muts_in_clusters, y = label_y, label = label),
            size = 3, color = "black") +
  labs(x = "Fraction of variants within clusters", y = "-log10(FDR)",
        color = "Significance", title = "Oncodrive Results") +
  theme_minimal() +
  theme(legend.position = "bottom")
```



The x-axis represents the fraction of mutations in each gene that are clustered within specific regions of the protein, while the y-axis shows the statistical significance (negative log10 of the p-value i.e. of the fdr) of this clustering. Genes that appear higher on the y-axis have mutation patterns that are less likely to be due to random chance, indicating they may be driver genes contributing to cancer development. The size of the points is proportional to the number of clusters found in the gene.

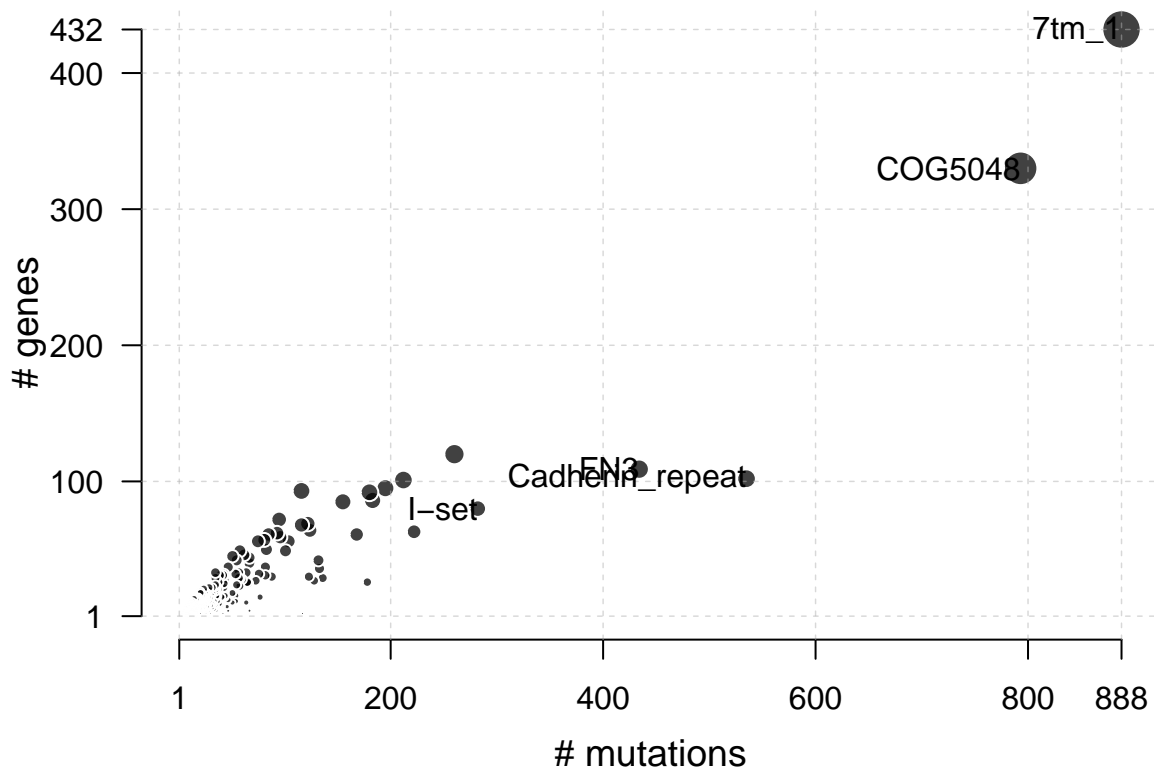
PFAM Domains

From lollipop plots, we remember that proteins have functional domains (e.g., kinase domain in PIK3CA). We studied the mutations clustering in such domains through lollipop plots & oncodrive plots. However

domains are not unique to one gene: they are reusable, conserved functional units that appear in many different genes. For example, the kinase domain appears in hundreds of human genes like EGFR, BRAF, PIK3CA, AKT1, SRC, ABL1, etc. This also has evolutionary reasons.

The PFAM database defines domain families based on sequence similarity. The `pfamDomains()` function maps mutations in the MAF to their corresponding PFAM domains. This helps us identify which domains are frequently mutated across different genes in our breast cancer cohort.

```
brca_pfam = pfamDomains(maf = brca_maf, AACol = 'HGVSp_Short')
```



```
head(brca_pfam)
```

```
## $proteinSummary
##      HGNC AAPos Variant_Classification    N total  fraction
##      <char> <num>          <fctr> <int> <num>      <num>
##    1: PIK3CA  1047      Missense_Mutation  107  305 0.3508197
##    2: PIK3CA   545      Missense_Mutation   60  305 0.1967213
##    3: PIK3CA   542      Missense_Mutation   40  305 0.1311475
##    4:  AKT1    17       Missense_Mutation   17   19 0.8947368
##    5: PIK3CA  345      Missense_Mutation   17  305 0.0557377
##    ---
## 56148: ZZEF1  1645      Nonsense_Mutation    1    9 0.1111111
## 56149: ZZEF1   962      Missense_Mutation    1    9 0.1111111
## 56150: ZZZ3   569      Missense_Mutation    1    4 0.2500000
## 56151: ZZZ3   356      Missense_Mutation    1    4 0.2500000
```

```

## 56152:   ZZZ3   493   Missense_Mutation   1   4 0.2500000
##           DomainLabel
##           <char>
##   1:       PI3Kc_IA_alpha
##   2:       PI3Ka_I
##   3:       PI3Ka_I
##   4:       PH_Akt
##   5: C2_PI3K_class_I_alpha
##   ---
## 56148:           <NA>
## 56149:           <NA>
## 56150:           <NA>
## 56151:           <NA>
## 56152:           <NA>
##
##
##   1:                               The PI3K catalytic domain family is part of a larger superfamily that i
##   2: PIK domain is conserved in all PI3 and PI4-kinases. Its role is unclear but it has been sugges
##   3: PIK domain is conserved in all PI3 and PI4-kinases. Its role is unclear but it has been sugges
##   4:
##   5:
##   ---
## 56148:
## 56149:
## 56150:
## 56151:
## 56152:
##
##                                     Description
##                                     <char>
##   1: Phosphoinositide 3-kinase (PI3K), class IA, alpha isoform, catalytic domain
##   2: Phosphoinositide 3-kinase (PI3K) class I, accessory domain
##   3: Phosphoinositide 3-kinase (PI3K) class I, accessory domain
##   4: Akt pleckstrin homology (PH) domain
##   5: C2 domain present in class I alpha phosphatidylinositol 3-kinases (PI3Ks)
##   ---
## 56148:                                     <NA>
## 56149:                                     <NA>
## 56150:                                     <NA>
## 56151:                                     <NA>
## 56152:                                     <NA>
##
## $domainSummary
##           DomainLabel nMuts nGenes
##           <char> <int> <int>
##   1:       7tm_1     888   432
##   2:       COG5048   793   330
##   3: Cadherin_repeat 535   102
##   4:       FN3     434   109
##   5:       I-set    282    80
##   ---
## 5485:       zf-UBR      1     1
## 5486:       zf-ZPR1     1     1
## 5487:       zf-rbx1     1     1
## 5488:       zf-ribbon_3  1     1

```

```

## 5489:          zf-tcix      1      1
##
##
## 1:
## 2:
## 3:
## 4: One of three types of internal repeats found in the plasma protein fibronectin. Its tenth fibr
## 5:
## ---
## 5485:
## 5486:
## 5487:
## 5488:
## 5489:
##
##          Description
##          <char>
## 1:          7 transmembrane receptor (rhodopsin family)
## 2: FOG: Zn-finger [General function prediction only]
## 3:          Cadherin tandem repeat domain
## 4:          Fibronectin type 3 domain
## 5:          Immunoglobulin I-set domain
## ---
## 5485:          Putative zinc finger in N-recognin (UBR box)
## 5486:          ZPR1 zinc-finger domain
## 5487:          RING-H2 zinc finger
## 5488:          zinc-ribbon domain
## 5489:          <NA>

```

Each dot in this scatterplot depicts a protein domain.

Most dots cluster in the bottom-left: Low # mutations and Low # genes. This is expected as most domains are not under strong selection in cancer, and thus are low-impact domains.

The domains in the top-right corner are likely driver domains that contribute to cancer development when mutated.

Now protein domains correspond to specific pathways. This shows that cancer doesn't just target a few genes, but rather selects for pathway-level disruption. E.g.:

- Cadherin_repeat : mediates cell-cell adhesion — critical in epithelial cancers like BRCA
- COG5048 : contains genes involved in chromatin/DNA binding which may support epigenetic dysregulation in BRCA
- 7tm : contains GPCR signaling genes such as ORs(Olfactory Receptors). The role of GPCR signals is still feebly known in cancer, so it's worth researching !

Survival Analysis

Survival analysis is a statistical method to study time until an event occurs — in cancer, that event is usually death or recurrence.

Since this particular cohort MAF dataset doesn't have any useful clinical data included, I couldn't get the `mafSurvival` function from `maftools` to work properly. So I tried the `survival` and `survminer` packages. Further I needed to read the clinical data from a separate TSV file for survival analysis. I went to <https://portal.gdc.cancer.gov/projects/TCGA-BRCA>, then clicked on the 'Clinical' tab and downloaded the TSV format. A .tar.gz file will be downloaded. Extract it to get the following files :

- clinical.tsv ← Core patient info (vital status, age, stage)
- exposure.tsv ← Smoking, alcohol, environmental exposures
- family_history.tsv ← Cancer history in relatives
- follow_up.tsv ← Serial follow-up records (e.g., recurrence, treatment response)
- pathology_detail.tsv ← Tumor grade, histology, ER/PR/HER2 status

For survival analysis, we need:

- Survival time : days_to_death (if deceased) or days_to_last_followup (if alive)
- Event status : vital_status (Dead or Alive)

These fields are in clinical.tsv so we don't need other files. We copy the clinical.tsv file in our working directory

```
# ---- Minimal survival analysis from clinical.tsv ----

if(!requireNamespace("readr",quietly=TRUE)) install.packages("readr")
if(!requireNamespace("survival",quietly=TRUE)) install.packages("survival")

library(readr)
library(survival)
library(tidyverse)

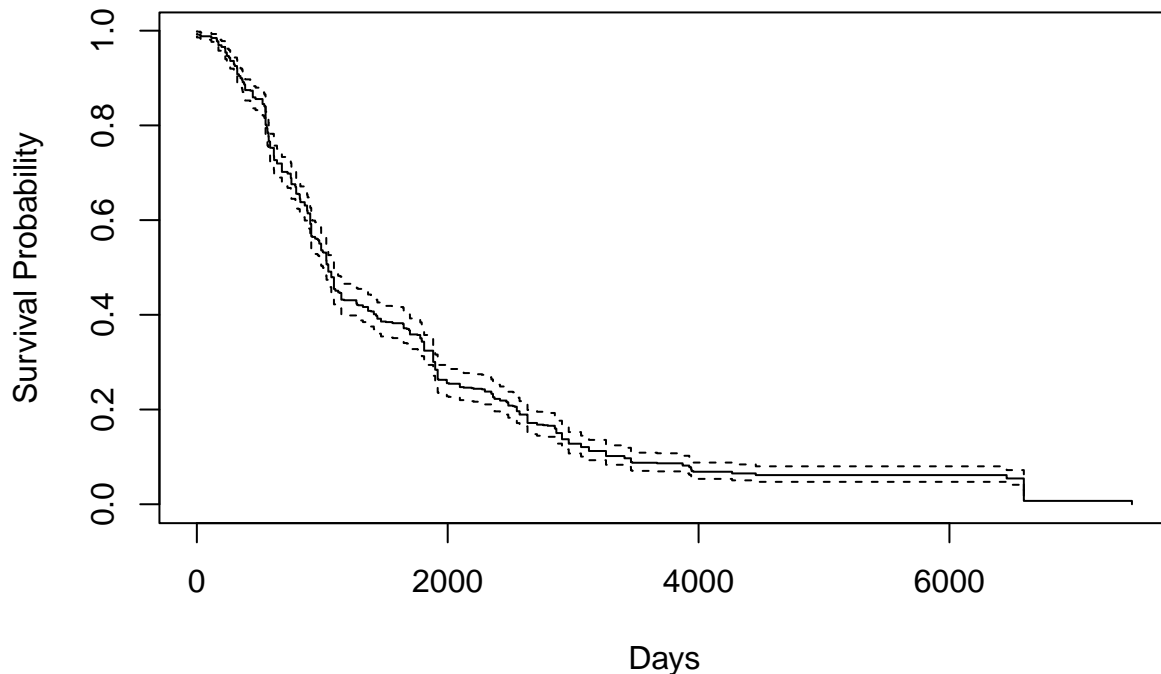
# 1) Load clinical.tsv
clin <- read_tsv("clinical.tsv", show_col_types = FALSE)

# 2) Create survival time and event
clin2 <- clin %>%
  mutate(
    patient_id = toupper(substr(cases.submitter_id,1,12)), # Creates clean uppercase patient IDs
    vital_status = tolower(demographic.vital_status),
    days_to_death = as.numeric(demographic.days_to_death),
    days_to_last_followup = as.numeric(cases.days_to_lost_to_followup),

    surv_time = ifelse(!is.na(days_to_death) & vital_status=="dead",
                      days_to_death, days_to_last_followup),
    event = ifelse(vital_status=="dead", 1, 0)
  ) %>%
  filter(!is.na(surv_time))

# 3) Kaplan-Meier
km <- survfit(Surv(surv_time, event) ~ 1, data = clin2)
plot(km, xlab="Days", ylab="Survival Probability", main="TCGA-BRCA Overall Survival")
```

TCGA-BRCA Overall Survival



This plot shows the overall survival of all breast cancer patients in our cohort. The y-axis represents the probability of survival, while the x-axis shows time in days. The curve starts at 1 (100% survival) and decreases over time as patients die. The steepness of the curve indicates how quickly patients are dying. It is like a staircase going down. Each step down is a death. The height of the staircase at any day tells you: “If you were diagnosed today, this is your chance of still being alive on that day.”

Remember we can’t just average everyone’s survival time, because some people are still alive! If we did that, we’d be pretending we know when they’ll die, which we don’t. So instead, we use a trick called the Kaplan-Meier method, where every time someone dies, we update the survival chance based only on the people we still know about at that moment. The people who vanish (“lost to follow-up”) aren’t ignored—they just stop counting after their last known day.

The way this works in my R code is as follows: 1. We load the clinical data from the `clinical.tsv` file using `read_tsv()`. 2. We create a new data frame `clin2` that contains the patient ID, survival time, and event status (1 if dead, 0 if alive). 3. We use the `survfit()` function to fit a Kaplan-Meier survival model to the data. 4. Finally, we plot the survival curve using the `plot()` function.

`nrow(clin2)` tells us how many patients have survival data. In our case, there are 845 patients with survival information, which is a little more than 800 which was the cohort size of `brca_maf`.

Next, we can perform survival analysis to see if mutations in **specific genes** impact patient outcomes. Here, I analyze the effect of TP53 mutations on overall survival in our breast cancer cohort.

```
# Survival by Mutation (Mutant vs WT)

gene_of_interest <- "TP53" # change as needed

if(!requireNamespace("survminer",quietly=TRUE)) install.packages("survminer")
```

```

library(readr); library(dplyr); library(survival); library(survminer)

# Compute per-patient mutation (non-silent) for gene_of_interest in the MAF object

maf_df <- brca_maf@data %>%
  mutate(patient_id = toupper(substr(Tumor_Sample_Barcode,1,12)))

# consider non-silent classes (basic filter)
nonsilent <- c("Missense_Mutation","Nonsense_Mutation","Frame_Shift_Del","Frame_Shift_Ins",
              "Splice_Site","Nonstop_Mutation","In_Frame_Del","In_Frame_Ins","Translation_Start_Site")

mut_patients <- maf_df %>%
  filter(Hugo_Symbol == gene_of_interest & Variant_Classification %in% nonsilent) %>%
  distinct(patient_id) %>%
  mutate(mut = 1)

# Merge clinical + mutation status (missing -> WT=0)
clin_mut <- clin2 %>%
  left_join(mut_patients, by = "patient_id") %>%
  mutate(mut = ifelse(is.na(mut), 0, mut))

# quick check counts
cat("N total:", nrow(clin_mut), " Mutant:", sum(clin_mut$mut==1), " WT:", sum(clin_mut$mut==0), "\n")

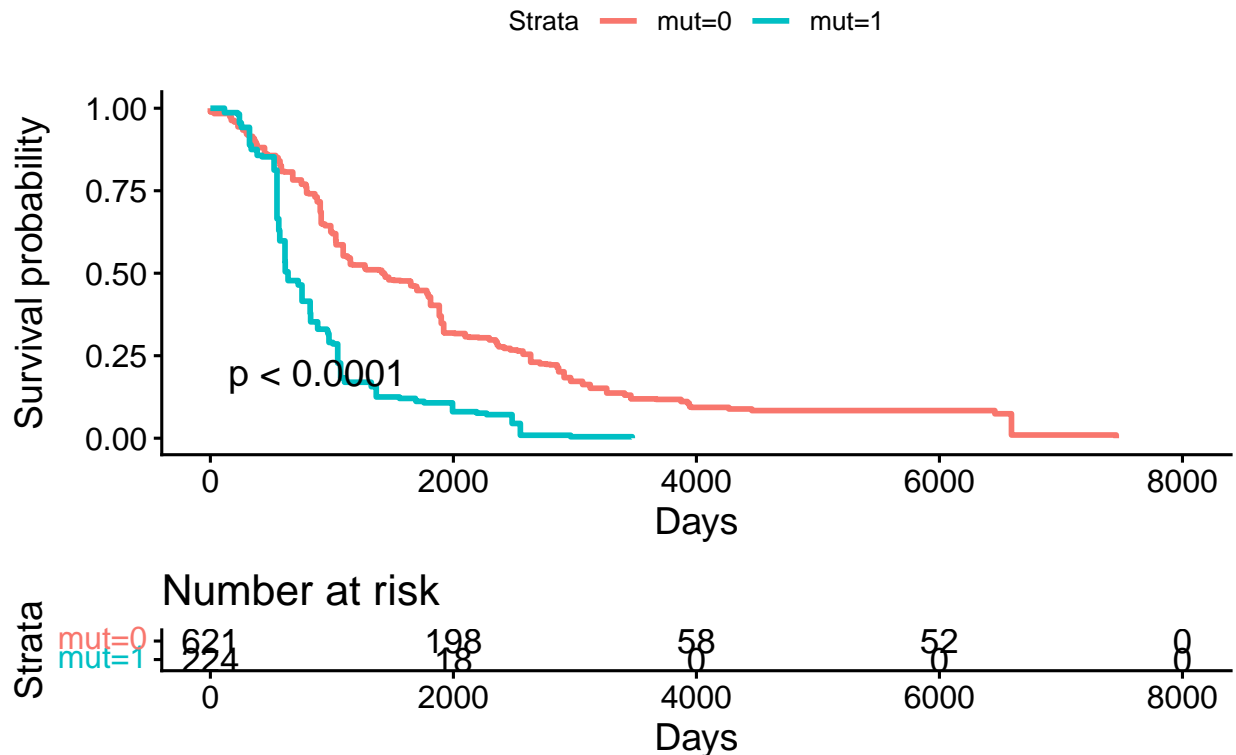
## N total: 845  Mutant: 224  WT: 621

# KM plot (Mutant vs WT) with p-value and risk table
fit <- survfit(Surv(surv_time, event) ~ mut, data = clin_mut)
p <- ggsurvplot(fit, data = clin_mut,
                pval = TRUE, risk.table = TRUE,
                title = paste(gene_of_interest, "Mutant v/s WT"),
                xlab = "Days", ylab = "Survival probability")

print(p)

```

TP53 Mutant v/s WT



```
# Cox model (mut effect) -> HR and p-value
cox <- coxph(Surv(surv_time, event) ~ mut, data = clin_mut)
s <- summary(cox)
hr <- s$coefficients["exp(coef)"]
ci_low <- s$conf.int["lower .95"]
ci_high <- s$conf.int["upper .95"]
pval <- s$coefficients["Pr(>|z|)"]
cat(sprintf("Cox HR for %s (mut vs WT): %.2f (95% CI: %.2f - %.2f), p = %.3g\n",
            gene_of_interest, hr, ci_low, ci_high, pval))
```

```
## Cox HR for TP53 (mut vs WT): 2.60 (95% CI: 2.21 - 3.06), p = 1.94e-30
```

The `(mut = ifelse(is.na(mut), 0, mut))` code ensures that any patient in `clin2` whose mutation status is missing in the MAF file is assumed to carry the wild-type (`mut=0`) in the studied gene.

The `survfit()` function from `survival` package was used to fit and compute survival (KM) curves.

The `ggsurvplot()` function from the `survminer` package takes the output of a `survfit()` object, enhances it with some features like p-value and uses `ggplot2` to display the resulting plot.

In the Cox test, our null hypothesis is that the mutation (`mut`) has no effect on the hazard of the event (e.g., death). If the hazard in the mutated group is the same as the hazard in the wild-type (non-mutated) group, their ratio is 1.0. Therefore, the null value is $HR = 1.0$.

When we examine the confidence interval for our hazard ratio, we check whether it includes the null value of 1.0. If the 95% confidence interval DOES NOT contain 1.0, the result is statistically significant at the 5%

level ($=0.05$). This means you have enough evidence to reject the null hypothesis that the mutation has no effect.

In our case, HR of 2.60 (95% CI: 2.21 - 3.06) indicates the TP53 mutation is associated with a significantly higher risk of the event (death, in this case).

```
# Survival by Mutation (Mutant vs WT)

gene_of_interest <- "RPL22" # change as needed

if(!requireNamespace("survminer",quietly=TRUE)) install.packages("survminer")

library(readr); library(dplyr); library(survival); library(survminer)

# Compute per-patient mutation (non-silent) for gene_of_interest in the MAF object

maf_df <- brca_maf@data %>%
  mutate(patient_id = toupper(substr(Tumor_Sample_Barcode,1,12)))

# consider non-silent classes (basic filter)
nonsilent <- c("Missense_Mutation","Nonsense_Mutation","Frame_Shift_Del","Frame_Shift_Ins",
              "Splice_Site","Nonstop_Mutation","In_Frame_Del","In_Frame_Ins","Translation_Start_Site")

mut_patients <- maf_df %>%
  filter(Hugo_Symbol == gene_of_interest & Variant_Classification %in% nonsilent) %>%
  distinct(patient_id) %>%
  mutate(mut = 1)

# Merge clinical + mutation status (missing -> WT=0)
clin_mut <- clin2 %>%
  left_join(mut_patients, by = "patient_id") %>%
  mutate(mut = ifelse(is.na(mut), 0, mut))

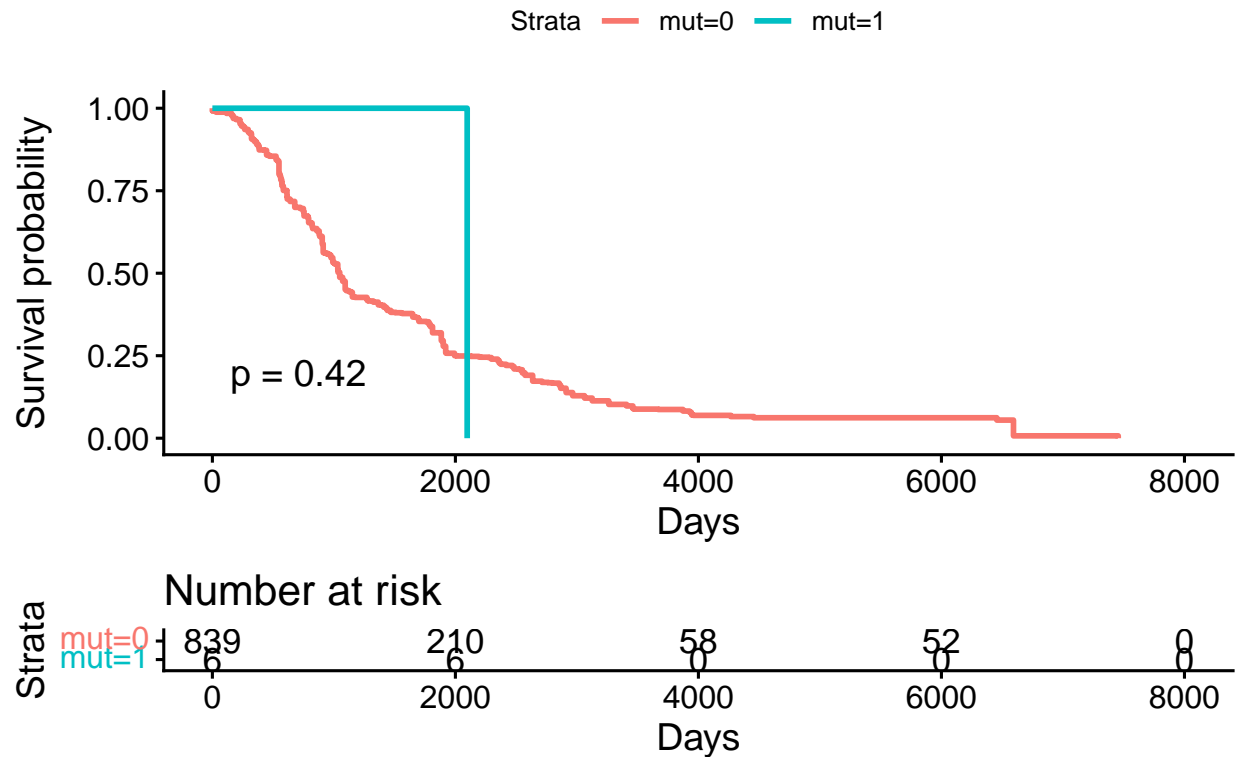
# quick check counts
cat("N total:", nrow(clin_mut), " Mutant:", sum(clin_mut$mut==1), " WT:", sum(clin_mut$mut==0), "\n")

## N total: 845 Mutant: 6 WT: 839

# KM plot (Mutant vs WT) with p-value and risk table
fit <- survfit(Surv(surv_time, event) ~ mut, data = clin_mut)
p <- ggsurvplot(fit, data = clin_mut,
                pval = TRUE, risk.table = TRUE,
                title = paste(gene_of_interest, "Mutant v/s WT"),
                xlab = "Days", ylab = "Survival probability")

print(p)
```

RPL22 Mutant v/s WT



```
# Cox model (mut effect) -> HR and p-value
cox <- coxph(Surv(surv_time, event) ~ mut, data = clin_mut)
s <- summary(cox)
hr <- s$coefficients[,"exp(coef)"]
ci_low <- s$conf.int[,"lower .95"]
ci_high <- s$conf.int[,"upper .95"]
pval <- s$coefficients[,"Pr(>|z|)"]
cat(sprintf("Cox HR for %s (mut vs WT): %.2f (95% CI: %.2f - %.2f), p = %.3g\n",
            gene_of_interest, hr, ci_low, ci_high, pval))
```

```
## Cox HR for RPL22 (mut vs WT): 0.72 (95% CI: 0.32 - 1.61), p = 0.423
```

```
# Survival at 2, 3, 5 years (where both groups have decent data)
time_points <- c(730, 1095, 1825) # days

surv_rates <- summary(fit, times = time_points, extend = TRUE)$surv

rates_df <- data.frame(
  Time_Years = time_points / 365.25,
  WT_Survival = surv_rates[1] * 100,
  Mutant_Survival = surv_rates[2] * 100
)

print(rates_df)
```

```
##   Time_Years WT_Survival Mutant_Survival
## 1   1.998631   69.60667   45.05364
## 2   2.997947   69.60667   45.05364
## 3   4.996578   69.60667   45.05364
```

Prognostic survival analysis

After individual gene survival analysis, I want to study which genes are prognostic, i.e. associated with poor survival (high HR).

```
# ---- Prognostic genes (high HR) ----
# minimal gene-wise Cox scan (uses existing clin2 and maf_df)
# minimal gene-wise Cox scan (uses existing clin2 and maf_df)

library(survival)

mut_df <- maf_df[, c("patient_id", "Hugo_Symbol")]
if(nrow(mut_df) == 0) stop("No mutations found in maf_df")

# patient x gene binary matrix (1 if mutated)
mat <- as.data.frame.matrix(table(mut_df$patient_id, mut_df$Hugo_Symbol))
mat$patient_id <- rownames(mat)

# merge with clin2 (keep all clin patients)
dat <- merge(clin2[, c("patient_id", "surv_time", "event")], mat, by="patient_id", all.x=TRUE)
dat[is.na(dat)] <- 0

# The list of genes to test
genes <- setdiff(names(dat), c("patient_id", "surv_time", "event"))

# Now we run a Cox model for each gene
res <- do.call(rbind, lapply(genes, function(g){
  mod <- try(coxph(as.formula(paste0("Surv(surv_time,event)~", g)), data = dat), silent=TRUE)
  if(inherits(mod, "try-error")) return(NULL)
  s <- summary(mod)
  data.frame(gene = g, HR = s$coef[1, "exp(coef)"], pval = s$coef[1, "Pr(>|z|)"], row.names = NULL)
}))

# Sorted by pval
res <- res[order(res$pval), ]
head(res, 20) # top genes by p-value
```

```
##           gene           HR           pval
## 9000    PCNX1    23.072600 2.758542e-34
## 9709    PPRC1     8.364968 1.106395e-32
## 5791  IL1RAPL1     6.910207 2.801652e-32
## 4413    FKBP8   119.479325 3.250433e-32
## 1363    BRCA2     8.061424 4.726706e-32
## 12574   TEX15     8.029087 1.749149e-31
## 13038    TP53     1.714419 5.281829e-30
## 11112  SETDB1     7.466662 1.355110e-28
## 3606    E2F3    16.745565 1.759780e-28
```

```
## 3674    EFCAB5    16.745565 1.759780e-28
## 8975    PCDHGB4    7.350459 4.552335e-28
## 11125    SF3B3    7.350459 4.552335e-28
## 13664    USP37    7.350459 4.552335e-28
## 3909     ERBB3    6.371841 2.180387e-27
## 300     ADCY10    44.886531 2.412807e-27
## 9166     PGGHG    44.886531 2.412807e-27
## 13259    TRPM4    44.886531 2.412807e-27
## 2338     CHD7     7.291989 4.018835e-27
## 3750     EIF4G3    7.291989 4.018835e-27
## 7677     MYO18A    7.291989 4.018835e-27
```

What happened above was that I computed the per-patient mutation status for all genes in the MAF object. Then I merged this mutation status with the clinical survival data. Finally, I looped over each gene to fit a Cox proportional hazards model and extract the hazard ratio (HR) and p-values. Then I sorted the results by p-value to identify the most prognostic genes.

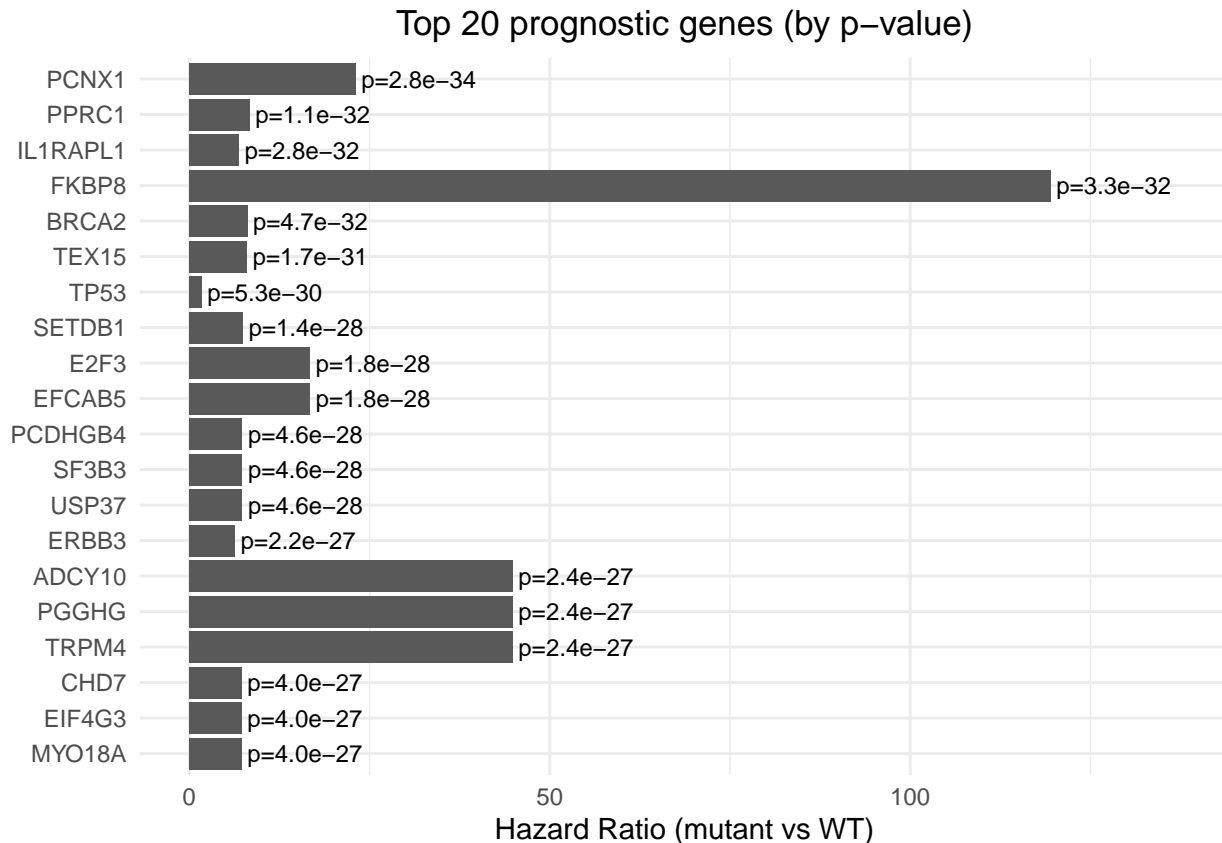
Next, to plot these most prognostic genes, I used the `ggplot2` package to create a forest plot showing the hazard ratios and their confidence intervals.

```
library(ggplot2)

# Select top 20 genes by p-value (assumes res has gene, HR, pval)
top_genes <- head(res[order(res$pval), ], 20)
top_genes <- subset(top_genes, !is.na(HR))          # drop any NA HR rows
top_genes$gene <- factor(top_genes$gene, levels = rev(top_genes$gene)) # order for plotting

# Plot: horizontal bars of HR with p-value labels
p <- ggplot(top_genes, aes(x = gene, y = HR)) +
  geom_col() +
  coord_flip() +
  geom_text(aes(label = sprintf("p=%.1e", pval)),
            hjust = -0.05, size = 3) +
  expand_limits(y = max(top_genes$HR, na.rm = TRUE) * 1.15) +
  labs(x = NULL, y = "Hazard Ratio (mutant vs WT)",
       title = "Top 20 prognostic genes (by p-value)") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))

print(p)
```



This plot shows the top 20 genes whose mutation status is most significantly associated with overall survival in our breast cancer cohort. Each horizontal bar represents a gene, with the length of the bar indicating the hazard ratio (HR) for patients with mutations in that gene compared to wild-type patients.

Clinical Enrichment

In Survival Analysis we asked the question: “Does mutation in gene X predict survival?” In “clinical enrichment” test we will ask: “Are mutations in certain genes enriched in specific clinical groups—like Stage III vs. all others?”

So, we’ll need to pick a categorical variable that defines a subgroup like tumor stage, grade, receptor status, or treatment outcome : anything that divides patients into categories.

```
# ---- Clinical Enrichment Test ----
# Requires: clin2 (deduplicated, one row per patient) and dat (with gene mutation columns)

# 1) Add stage column safely (no merge duplication)
dat$diagnoses.ajcc_pathologic_stage <- clin2$diagnoses.ajcc_pathologic_stage[
  match(dat$patient_id, clin2$patient_id)
]

# 2) Collapse stages: "Stage IIA" + "II", etc.
dat$stage <- sub("([IV]+)[ABC]$", "\\1",
  sub("^Stage\\s*", "", dat$diagnoses.ajcc_pathologic_stage))

# 3) Define clinical feature to test
```

```

clinical_feature <- "stage"
feature_vals <- na.omit(unique(dat[[clinical_feature]]))

# 4) For each stage group, test enrichment of mutations in each gene
results <- lapply(feature_vals, function(fv) {
  # Create binary grouping: this stage vs. all others
  group <- ifelse(dat[[clinical_feature]] == fv, fv, "Rest")

  # Fisher's test for each gene (0/1 mutation status)
  pvals <- sapply(setdiff(names(dat), c("patient_id", "surv_time", "event",
                                         "diagnoses.ajcc_pathologic_stage", "stage")),
                  function(g) {
                    tab <- table(group, dat[[g]])
                    if (all(dim(tab) == c(2, 2))) {
                      fisher.test(tab, workspace = 2e8)$p.value # avoid workspace errors
                    } else NA
                  })

  # Assemble results
  df <- data.frame(Gene = names(pvals), Pval = pvals, Group = fv)
  df <- df[!is.na(df$Pval), ]
  df$FDR <- p.adjust(df$Pval, method = "fdr")
  df[order(df$Pval), ][1:10, ] # top 10 genes per group
}) %>% bind_rows()

# 5) View top enriched gene-stage associations
head(results, 20)

```

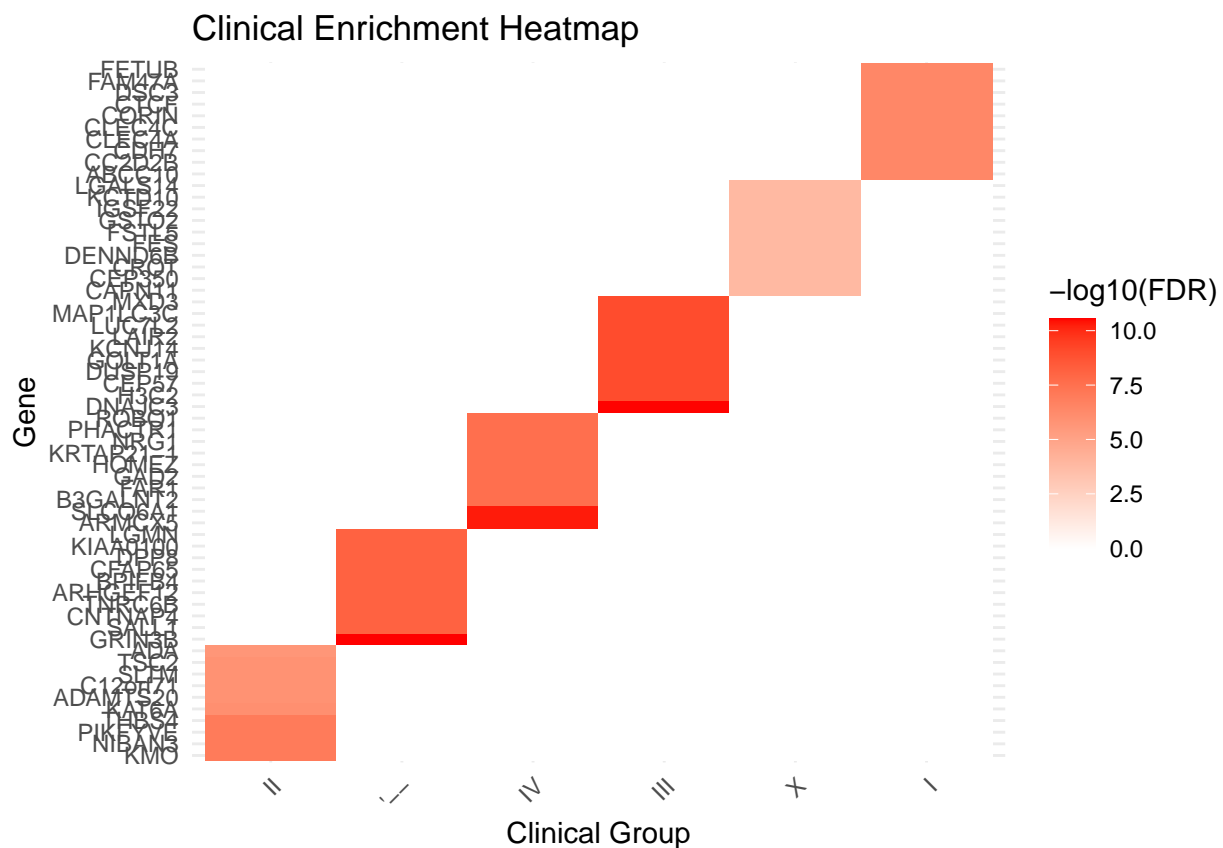
##	Gene	Pval	Group	FDR
##	KMO	KMO 6.775830e-11	II	9.103328e-08
##	NIBAN3	NIBAN3 6.775830e-11	II	9.103328e-08
##	PIKFYVE	PIKFYVE 6.775830e-11	II	9.103328e-08
##	THBS4	THBS4 6.775830e-11	II	9.103328e-08
##	KAT6A	KAT6A 8.451441e-10	II	9.083609e-07
##	ADAMTS20	ADAMTS20 2.046022e-09	II	1.221702e-06
##	C12orf71	C12orf71 2.046022e-09	II	1.221702e-06
##	SLTM	SLTM 2.046022e-09	II	1.221702e-06
##	TSC2	TSC2 2.046022e-09	II	1.221702e-06
##	ADA	ADA 1.115283e-08	II	2.066734e-06
##	GRIN3B	GRIN3B 5.195603e-15	'--	2.792117e-11
##	SALL1	SALL1 2.627964e-12	'--	6.960796e-09
##	CNTNAP4	CNTNAP4 1.080745e-11	'--	6.960796e-09
##	TNRC6B	TNRC6B 1.080745e-11	'--	6.960796e-09
##	ARHGEF12	ARHGEF12 2.201964e-11	'--	6.960796e-09
##	BPIFB4	BPIFB4 2.201964e-11	'--	6.960796e-09
##	CFAP65	CFAP65 2.201964e-11	'--	6.960796e-09
##	DPP8	DPP8 2.201964e-11	'--	6.960796e-09
##	KIAA0100	KIAA0100 2.201964e-11	'--	6.960796e-09
##	LGMN	LGMN 2.201964e-11	'--	6.960796e-09

To get a graphical representation of the above results, we can plot a heatmap of $-\log_{10}(\text{FDR})$ values for the top enriched genes across different clinical groups.

```

library(ggplot2)
library(tidyr)
# Prepare data for heatmap
heatmap_data <- results %>%
  filter(FDR < 0.1) %>% # only significant
  mutate(logFDR = -log10(FDR)) %>%
  select(Gene, Group, logFDR) %>%
  pivot_wider(names_from = Group, values_from = logFDR, values_fill = 0)
# Convert to matrix for ggplot
heatmap_matrix <- as.data.frame(heatmap_data)
rownames(heatmap_matrix) <- heatmap_matrix$Gene
heatmap_matrix$Gene <- NULL
heatmap_melt <- reshape2::melt(as.matrix(heatmap_matrix))
# Plot heatmap
ggplot(heatmap_melt, aes(Var2, Var1, fill = value)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "red", na.value = "grey") +
  labs(x = "Clinical Group", y = "Gene", fill = "-log10(FDR)",
       title = "Clinical Enrichment Heatmap") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

```



To focus only on the Group/grade 4 tumors, we can filter the `results` data frame for `Group == "4"` and then plot the top 5 significant genes.

```

heatmap_data <- results %>%
  filter(FDR < 0.1, Group == "IV") %>% # <--- keep only group IV
  mutate(logFDR = -log10(FDR)) %>%
  select(Gene, Group, logFDR) %>%
  pivot_wider(names_from = Group, values_from = logFDR, values_fill = 0)

heatmap_matrix <- as.data.frame(heatmap_data)
rownames(heatmap_matrix) <- heatmap_matrix$Gene
heatmap_matrix$Gene <- NULL
heatmap_melt <- reshape2::melt(as.matrix(heatmap_matrix))

ggplot(heatmap_melt, aes(Var2, Var1, fill = value)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "red", na.value = "grey") +
  labs(x = "Clinical Group", y = "Gene", fill = "-log10(FDR)",
       title = "Clinical Enrichment - Group IV only") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

```



To filter out only the grade 4 tumor top 5 significant genes in the graph, and represent graphically through , we can do the following:

```

# Filter for grade 4 only
grade4_results <- results %>%
  filter(Group == "4") %>%
  arrange(Pval) %>%

```

```

head(5)
# Plot bar graph for grade 4 top 5 genes
ggplot(grade4_results, aes(x = reorder(Gene, -log10(FDR)), y = -log10(FDR), fill = Gene)) +
  geom_bar(stat = "identity") +
  labs(x = "Gene", y = "-log10(FDR)", title = "Top 5 Enriched Genes in Grade 4 Tumors") +
  theme_minimal()

```

Top 5 Enriched Genes in Grade 4 Tumors

-log10(FDR)

Gene

The above graph is empty, so to resolve this issue the code is modified as below:

```

# Filter for grade 4 only
grade4_results <- results %>%
  filter(Group == "4") %>%
  arrange(Pval) %>%
  head(5)
# Plot bar graph for grade 4 top 5 genes
ggplot(grade4_results, aes(x = reorder(Gene, -log10(FDR)), y = -log10(FDR), fill = Gene)) +
  geom_bar(stat = "identity") +
  labs(x = "Gene", y = "-log10(FDR)", title = "Top 5 Enriched Genes in Grade 4 Tumors") +
  theme_minimal()

```

Top 5 Enriched Genes in Grade 4 Tumors

$-\log_{10}(\text{FDR})$

Gene

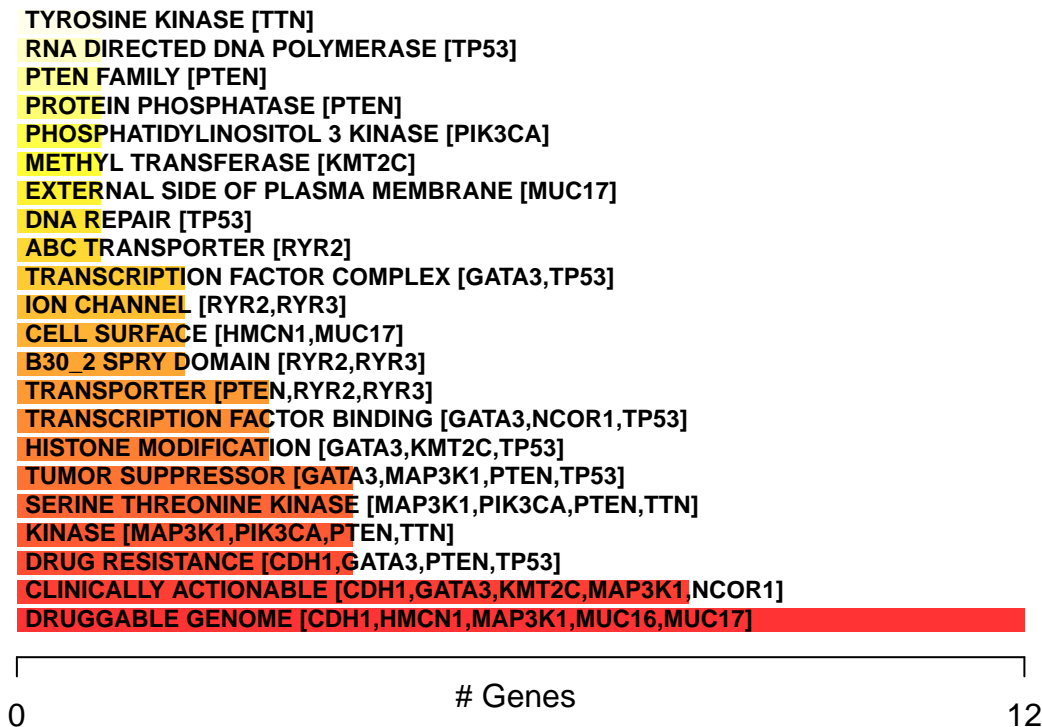
From above **results**, we get that mutated forms of genes **ARMCX5** and **SLCO6A1** are significantly enriched in high-grade tumors (grade 4) compared to lower grades. This suggests these gene mutations may play a role in tumor aggressiveness and progression in breast cancer.

Drug-gene interactions

maftools also provides the `drugInteractions()` function to fetch data from the Drug Gene Interaction database. It depicts which mutated genes in our cohort belongs to known drug-targetable pathways or functional classes.

```
# Drug-Gene Interactions  
dgi = drugInteractions(maf = brca_maf, fontSize = 0.75)
```

Druggable categories



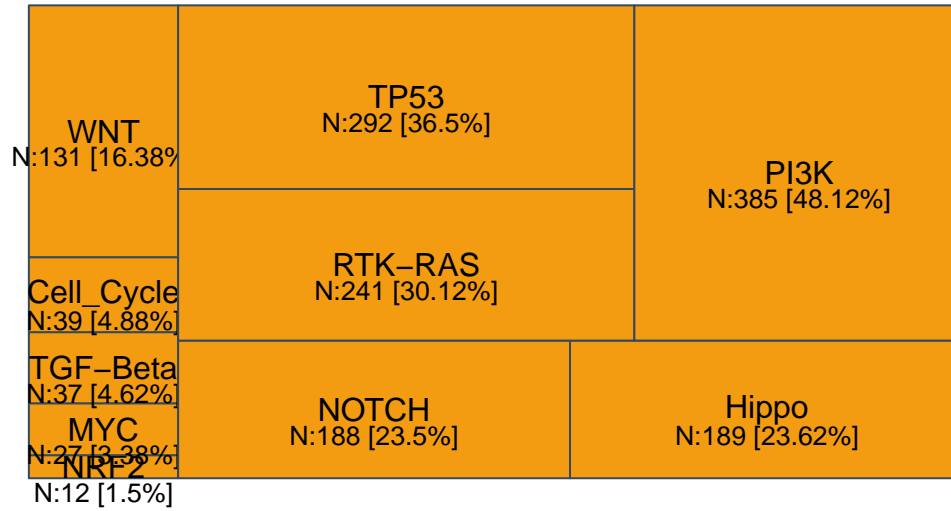
This bar plot highlights two special categories: “Clinically Actionable” and “Druggable Genome” — meaning these genes have existing drugs or clinical trials associated with them.

Connecting dots with known biology, I can link certain genes listed in this plot with known pathways or drug targets in breast cancer: - PIK3CA mutations can be targeted by PI3K inhibitors like alpelisib - PTEN loss can activate the PI3K/AKT pathway, suggesting potential benefit from PI3K/AKT/mTOR inhibitors - CDH1 mutations may impact cell adhesion and could influence response to therapies targeting the tumor microenvironment.

But to get a clearer picture of how these genes fit into broader oncogenic signaling pathways, we can next use the `pathways()` function in `maftools`.

Oncogenic signaling pathways

```
# Pathway analysis from maftools
## Summarizing signalling pathways [Sanchez-Vega et al., https://doi.org/10.1016/j.cell.2018.03.035]
pws = pathways(maf = brca_maf, plotType = 'treemap')
```



maftools also allows plotting these pathways.

```
# Plotting the pathways
plotPathways(maf = brca_maf, pathlist = pws)
```



This plot is a bar chart of pathways, where:

- Each row = a pathway (e.g., “PI3K”, “TP53”, “RTK-RAS”)
- The bar length = % of samples in your cohort with at least one alteration in that pathway
- The tiny vertical lines inside each bar = which patients have alterations in that pathway

For e.g., we can say that in 48.12% of tumors, we found at least one mutation or copy number change in a gene belonging to the PI3K pathway (e.g., PIK3CA, PTEN mutations). From theory we know that hyperactive PI3K signaling due to PIK3CA mutation or PTEN loss are targetable with existing drugs like PI3K inhibitors. So such drug interventions can be proposed.