



# Interview Preparation Guide

## Computational Research Scientist - Zhou Lab ProteinPaint Team

Comprehensive Q&A covering all aspects of the position beyond individual tutorials

### 🔍 Interview-Day Cheat Sheet (1-Page Summary)

#### 🎤 60-Second Pitch: "Why ProteinPaint + Why Me"

"I've spent my career **generating the data that ProteinPaint visualizes**—single-cell methylomes, ATAC-seq, RNA-seq—and now I want to transition from **data producer to tool builder**. My background gives me deep empathy for researchers who need to explore complex genomic data. I've already built an AI chatbot for genomics (OmicsOracle) that demonstrates I can deliver exactly what the team is building. The tech stack alignment is perfect: I've used Node.js, D3.js, TypeScript, and SQL in production systems, and I understand the biological context because I've published papers analyzing the same data types ProteinPaint displays."

#### 🔧 3 Projects I'll Reference Repeatedly

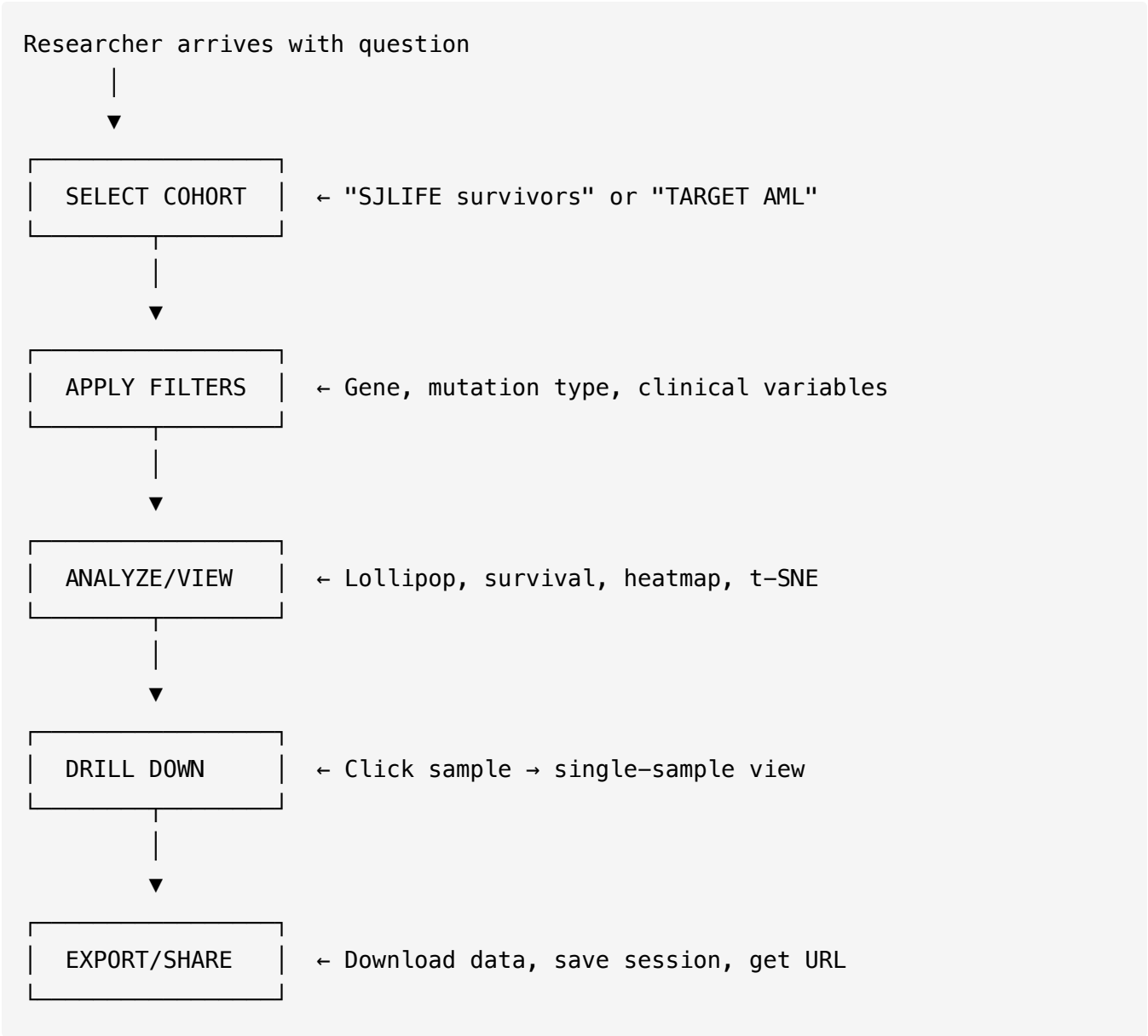
Project	What It Demonstrates	Key Talking Point
<b>OmicsOracle</b>	AI chatbot for genomics	"I've already built the feature they're adding to ProteinPaint"
<b>Methylome Paper</b>	Deep learning for epigenomics	"I understand the data ProteinPaint visualizes at a fundamental level"
<b>BioPipelines</b>	Workflow automation (Snakemake/Nextflow)	"I can work across the entire data lifecycle, not just the UI"

## 6 Technical Anchors (Memorize These)

Anchor	One-Liner	Depth Ready?
<b>Streaming + Indexing</b>	"Use tabix/BigWig for $O(\log n)$ random access; never load full files"	✅ Can explain R-tree, block compression
<b>Linked Views</b>	"Selections propagate across visualizations via shared state/event bus"	✅ Can code D3 brush → filter pattern
<b>Performance Strategy</b>	"Canvas for >10K points, WebGL for >100K, virtualize DOM, debounce"	✅ Can profile with Chrome DevTools
<b>Data Provenance</b>	"Track genome build (hg19/hg38), coordinate systems, QC filters"	✅ Understand liftOver, 0-based vs 1-based
<b>Testing</b>	"Unit tests for data transforms, visual regression for plots, E2E for workflows"	✅ Jest + Playwright experience
<b>Privacy/ Security</b>	"PHI never goes to external APIs; audit logs; role-based access"	✅ Understand dbGaP/ controlled-access

## The Workflow Spine (Product Thinking)

ProteinPaint portals follow a consistent "**COHORT** → **FILTER** → **ANALYZE** → **EXPORT**" pattern:



**Interview soundbite:** "The power of ProteinPaint isn't any single visualization—it's enabling the complete scientific workflow from hypothesis to evidence to publication figure."

## Table of Contents

#	Section	Questions	Key Topics
1	<a href="#">About ProteinPaint &amp; St.</a>	Q1-Q4	Platform, ecosystem, MB-meta portal

#	Section	Questions	Key Topics
	Jude		patterns
2	Cancer Biology & Genomics	Q5-Q10	Mutations, pathways, driver genes, St. Jude algorithms
3	Genomic File Formats	Q11-Q14	VCF, MAF, BED, BigWig, FASTQ, GFF, SRA
4	Reference Genomes & Coordinates	Q15-Q17	hg19/hg38, coordinate systems
5	D3.js & Visualization Libraries	Q18-Q21	Data join, scales, zoom/pan
6	Visualization Types & Techniques	Q22-Q25	Chart types, lollipop, t-SNE/UMAP, Hi-C/3D genome
7	Color Theory & Accessibility	Q26-Q27	Color scales, colorblind design
8	Core Technology Stack	Q28-Q30	Node.js, Express, TypeScript, Zod
9	Technology Selection & Trade-offs	Q31-Q34	D3 vs alternatives, TypeScript benefits, databases
10	Statistical Methods	Q35-Q37	Kaplan-Meier, FDR, Fisher's exact
11	Data Processing & Pipelines	Q38-Q45	Sequencing, alignment, QC, RNA-seq, DE, workflows
12	Software Architecture	Q46-Q49	Components, state, linked visualizations
13	Full-Stack Development	Q50-Q53	Testing, databases, error handling
14	Performance & Optimization	Q54-Q56	Large datasets, WebAssembly, profiling
15	AI & LLM Integration	Q57-Q60	Chatbots, RAG, prompt engineering, NL queries

#	Section	Questions	Key Topics
16	Scientific Communication	Q61-Q62	Stakeholders, publications
17	Team & Collaboration	Q63-Q64	Code review, disagreements
18	Behavioral Questions	Q65-Q67	STAR method, career goals
19	Questions to Ask Them	-	Strategic questions for interviewers
20	Project Portfolio & Tech Stack	Q68-Q71	OmicsOracle, LLM evaluation, BioPipelines
21	Your Publications - Discussion Points	Q72-Q87	Single-cell methylation, ATAC-seq, cardiac fibrosis

**Total: 87 Interview Questions across 21 Categories**

---



## About ProteinPaint & St. Jude

### Q1: What is ProteinPaint and what makes it unique?

**Answer:**

ProteinPaint is St. Jude's open-source platform for biomedical data visualization, analysis, and sharing. What makes it unique:

1. **Origin Story:** Started as a mutation visualization tool showing amino acid changes on protein structures (hence "Protein Paint")
2. **Evolution:** Grown into an umbrella platform powering:
  - GenomePaint - Pediatric cancer visualization
  - NCI GDC Portal - National Cancer Institute data
  - Survivorship Portal - Long-term outcome tracking
  - Neuro-Oncology Portal - Brain tumor data
  - ASH HematOmics - Blood cancer program
3. **Unique Features:**
  - Multi-data modality (mutations, CNV, fusions, expression)

- Clinical-genomic data integration
  - AI chatbot for natural language queries
  - Designed for pediatric cancer research specifically
4. **Open Source:** Available on GitHub, community contributions welcome
- 

## Q2: Why is genomic visualization important for pediatric cancer research?

### Answer:

Pediatric cancers are fundamentally different from adult cancers:

1. **Lower mutation burden:** Children have fewer mutations, making each one more significant
2. **Different driver genes:** MYCN, PAX3-FOXO1, ETV6-RUNX1 vs. adult drivers
3. **Developmental context:** Tumors arise from developing tissues
4. **Treatment sensitivity:** Children respond differently to therapies

### Visualization enables:

- Identification of rare but recurrent mutations
  - Discovery of novel fusion genes
  - Clinical trial stratification
  - Treatment response prediction
  - Survivorship outcome tracking
- 

## Q3: Describe the ProteinPaint ecosystem and its major portals.

### Answer:

ProteinPaint began as a **cancer mutation visualization tool** ([Zhou et al., Nature Genetics 2016](#)) and has evolved into an **umbrella platform** powering multiple specialized portals for biomedical data visualization, analysis, and sharing.

The ProteinPaint Ecosystem:

Portal	Publication	Scale	Key Capabilities
ProteinPaint Core	<a href="#">Nat Genet 2016</a>	Foundation	Mutation lollipops, protein domains, fusion visualization
GenomePaint	<a href="#">Cancer Cell 2021</a>	Multi-cohort	WGS/WES/RNA-seq integration, non-coding variants, 3D genome
Survivorship Portal	<a href="#">Cancer Discov 2024</a>	7,700+ survivors	Clinical outcomes, 1,600+ variables, 400M genetic variants
NCI GDC Portal	<a href="https://portal.gdc.cancer.gov">portal.gdc.cancer.gov</a>	50,270 cases	TCGA, TARGET, CPTAC, harmonized pipelines
Neuro-Oncology Portal	<a href="#">Cancer Res 2025</a>	898 patients	Medulloblastoma, methylation, clinical trials integration
ppHiC Module	<a href="#">CSBJ 2024</a>	3D genome	Hi-C contact matrices, 4-view linked system, enhancer hijacking
ASH HematOmics	Paper in revision	Blood cancers	Leukemia, lymphoma, myeloma visualization

GenomePaint (<https://genomepaint.stjude.cloud/>):

"Interactive visualization platform for whole-genome, whole-exome, transcriptome, and epigenomic data... captures the inter-relatedness between DNA variations and RNA expression"

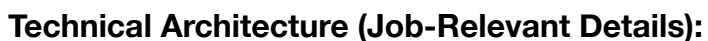
Key discoveries enabled:

- Aberrant splicing disrupting CREBBP RING domain

- Survivorship Portal** (<https://survivorship.stjude.cloud>):

### Data Scale:

- ### The Portal Workflow (Core Mental Model):



Layer	Technology	Why It Matters
Frontend	JavaScript + D3.js	ProteinPaint framework
Backend	Node.js	Same stack you'd work on
Phenotypes	SQLite	Fast tabular queries
Variants	BCF via BCFtools	Efficient variant access
LD data	tabix-indexed files	Standard bioinformatics approach



Layer	Technology	Why It Matters
Hosting	Physical servers at St. Jude	On-premise security

### Analysis Modules (CHARTS tab):

- Data dictionary (hierarchical variable browser)
- Summary plots (histogram, boxplot, scatter)
- Cumulative incidence analysis
- Regression analysis (linear, logistic, Cox)
- Population genetics genome browser
- Controlled-access data download

### The 4-Filter Variant QC Method (Shows Scientific Rigor):

1. **Genotype call rate**  $\geq 95\%$
2. **Coverage distribution** - normalized coverage  $\sim$  Gaussian, KS test
3. **VAF distribution** - model VAF/coverage for HO/HE states
4. **Hardy-Weinberg Equilibrium** - population genetics QC

This QC method shows they treat the portal as **scientific software**, not just UI—QC logic is part of the product.

### Key Discovery Enabled:

MAGI3 haplotype associated with cardiomyopathy in African ancestry survivors—demonstrating the power of integrated phenotype + genotype analysis.

### NCI Genomic Data Commons:

"A repository and computational platform for cancer researchers... harmonized cancer datasets"

Scale (Data Release 45.0):

- **91** Projects
- **69** Primary Sites
- **50,270** Cases

- **1,271,450** Files
- **22,638** Genes
- **3,324,495** Mutations

ProteinPaint powers the GDC's visualization tools including mutation frequency plots and survival analysis.

**Medulloblastoma Meta-Analysis Portal (MB-meta)** ([Cancer Res 2025](#)):

"Web-based data analysis and visualization platform that consolidates clinical, molecular, and survival data of 898 patients from three major clinical trials (ACNS0331, ACNS0332, SJMB03)"

**Cohort Details:**

- Ages 3-22, four consensus molecular groups (WNT, SHH, Group 3, Group 4)
- Emphasis on G3/G4 heterogeneity (historically grouped together, now recognized as distinct)
- Methylation data for ALL samples; sequencing for most (with some paired tumor/normal)

**Interactive Features (Key Interview Talking Points):**

Feature Category	Capabilities
Summary Visualizations	Bar plots, violin plots, sample distributions
Survival Analysis	Kaplan-Meier curves, Cox regression with custom variables
Mutation Analysis	Lollipop plots for coding mutations, sample matrices for co-occurrence
Methylome Integration	t-SNE with clinical/molecular overlays, lasso selection for custom subcohorts
Single-Sample View	Disco plots, methylation array visualization, CNV tracks

## What Makes It Scientifically Powerful:

The portal isn't just a dashboard—it enables a complete "**filter** → **define covariate** → **analyze** → **subcohort**" workflow:

```
// Example scientific workflow enabled by MB-meta
const mbMetaWorkflow = {
  step1_filter: 'Select G3/G4 patients with comparable treatment regimens',
  step2_defineVariable: 'Create custom iso17 variable from clinical data',
  step3_analyze: 'Run Cox regression for PFS with multiple covariates',
  step4_subcohort: 'Jump to specific samples matching criteria',
  step5_visualize: 'Overlay results on methylome t-SNE',
};
```

### Use Case 1: Prognostic Exploration in G3/G4

- Filter to comparable treatment regimens
- Define custom variables (e.g., iso17 status)
- Cox regression for progression-free survival (PFS)
- **Key finding:** MYC amplification, G3 group, metastatic disease → worse PFS; iso17 not significant in controlled setting

### Use Case 2: KBTBD4 Mutation + Methylome Integration

- Complex hotspot insertions in KBTBD4 Kelch repeat region
- Manual curation of ~700 tumors to rescue/correct calls (3 rescued, 21 corrected)
- Overlay verified KBTBD4 status on methylome t-SNE
- **Discovery:** Two visible clusters with different insertion patterns (R313insPRR vs R312\_R313insP)

### Future Directions (mentioned in paper):

- MRI and histology imaging integration
  - CSF (cerebrospinal fluid) biomarkers
  - Chromatin accessibility (ATAC-seq)
  - Proteomics and single-cell data
-

## Core ProteinPaint Features (from <https://proteinpaint.stjude.org/>):

Category	Tools
Tracks	GenomePaint, BigWig, BAM, Hi-C, Splice Junction, Arc Track, BCF
Launch Apps	Lollipop, Scatter Plot, Single Cell Viewer, Data Matrix, Disco Plot, Heatmap
Featured Data	Pediatric Cancer Mutations, GDC, CIViC, Survivorship
Specialized	Fusion Editor, 2D MAF, MA+Volcano plots, Gene Expression Clustering

---

## The Team (from <https://proteinpaint.stjude.org/team/>):

- **Xin Zhou, PhD** - Assistant Member, Director of Data Visualization
  - **Edgar Sioson, MS** - Principal Software Engineer
  - **Jian Wang, PhD** - Lead Computational Research Scientist
  - **Robin Paul, PhD** - Senior Computational Research Scientist
  - **Colleen Reilly, MS** - Senior Software Engineer
  - **Gavriel Matt, PhD** - Senior Computational Research Scientist
  - Plus additional team members (Congyu Lu, Alex Acic, Andrew Willems)
- 

## Why This Matters for Your Interview:

```
const interviewInsights = {
  sharedCodebase: 'Core components serve ALL portals – your code reaches many users',
  domainKnowledge: "Each portal has unique biological context you'll need to learn",
  scale: {
    gdcPortal: '50,000+ cases, petabytes of data',
    survivorship: '7,700+ survivors with longitudinal data',
    genomePaint: 'Multi-omics integration at single-sample level',
  },
  publications: 'Your work may be featured in high-impact journals',
  openSource: 'GitHub: github.com/stjude/proteinpaint',
};
```

### Smart Interview Questions to Ask:

- "Which portal would I primarily contribute to initially?"
- "How do you prioritize features across the different portals?"
- "What's the relationship between the team at St. Jude and NCI GDC collaborators?"
- "How often are new visualization types added to the platform?"
- "The MB-meta portal's custom variable → Cox regression workflow is powerful—are similar patterns being extended to other disease portals?"

---

## Deep Dive: MB-meta Portal as a Design Pattern

*The Medulloblastoma Meta-Analysis portal exemplifies ProteinPaint's philosophy of "analysis platform, not just dashboard."*

### Why MB-meta Matters for Your Interview:

Understanding MB-meta shows you grasp the **scientific workflow patterns** that make ProteinPaint valuable:

```
// The MB-meta workflow pattern (reusable across portals)
const portalWorkflowPattern = {
  // 1. Data Integration Layer
  dataLayers: {
    clinical: '898 patients, 3 trials, treatment regimens',
    molecular: 'Mutations, methylation subtypes',
    survival: 'PFS, OS with censoring',
  },

  // 2. Interactive Analysis Loop
  analysisLoop: [
    'Filter cohort by clinical/molecular criteria',
    'Define custom variables from existing data',
    'Run statistical analysis (Cox, log-rank)',
    'Visualize results on dimensionality reduction',
    'Select subcohort with lasso → drill down',
  ],

  // 3. QC/Curation Integration
  curationLoop: {
    example: 'KBTBD4 complex insertions',
    process: 'Manual review → rescue/correct calls → re-analyze',
    impact: '3 rescued, 21 corrected out of ~700 tumors',
  },
};
```

Technical Patterns to Highlight:

Pattern	MB-meta Implementation	Why It's Impressive
Linked Views	t-SNE ↔ survival ↔ sample matrix	Click propagates across all views
Custom Variables	Define iso17 from clinical columns	Users create analysis covariates
Lasso Selection	Draw on t-SNE to define subcohort	Spatial selection → filtered dataset

Pattern	MB-meta Implementation	Why It's Impressive
<b>Cox Regression</b>	Multi-variate survival analysis	Statistical rigor + interactivity
<b>Disco Plots</b>	Single-sample mutation visualization	Dense information display

### KBTBD4 Example (Great Interview Talking Point):

This case study demonstrates the "analysis platform + QC loop" pattern:

1. **Problem:** Complex hotspot insertions in KBTBD4 Kelch repeat region can be missed or miscalled by automated variant callers
2. **Solution:** Manual curation of ~700 tumors using the portal's visualization tools
3. **Outcome:** 3 cases rescued, 21 allele calls corrected
4. **Discovery:** Two methylome clusters with different insertion patterns emerged (R313insPRR vs R312\_R313insP)

"The portal enabled us to identify nuances in KBTBD4 mutations that could matter for subgroup classification—this wasn't possible with static analysis pipelines."

### What This Tells You About the Role:

Your work would likely involve:

- Building interactive visualizations that enable this kind of scientific discovery
- Implementing statistical methods (Cox regression, survival analysis) in JavaScript
- Creating linked-view architectures where selections propagate
- Supporting curation workflows alongside automated analysis

---

## Q4: What are St. Jude's research priorities?

### Answer:

St. Jude's mission: "Finding cures. Saving children."

### Research Priorities:

1. **Pediatric Cancers:** ALL, AML, brain tumors, neuroblastoma, sarcomas

2. **Catastrophic Diseases:** Sickle cell, other genetic disorders
3. **Global Health:** Accessible treatments worldwide
4. **Data Sharing:** Open science, cloud resources

### Computational Biology Focus:

- Large-scale genomic studies (Pediatric Cancer Genome Project)
- Multi-omics integration (CPTAC, TARGET)
- AI/ML for drug discovery
- Real-time clinical genomics

---

## Cancer Biology & Genomics

### Q5: Explain the types of genomic alterations in cancer.

Answer:

Type	Description	Visualization
<b>SNV</b>	Single nucleotide variants	Lollipop plot
<b>Indel</b>	Small insertions/deletions	Lollipop plot
<b>CNV</b>	Copy number variations	Segment plots
<b>SV</b>	Structural variants	Arc diagrams
<b>Fusion</b>	Gene fusions	Arc/circos plots
<b>Expression</b>	mRNA levels	Heatmaps, violin
<b>Methylation</b>	Epigenetic marks	Heatmaps

### Plot Type Descriptions:

- **Lollipop plot:** Shows mutations as "lollipops" along a protein/gene structure, with the stem indicating position and the circle representing mutation frequency or type—ideal for spotting recurrent mutation hotspots.



- **Segment plot:** Displays copy number changes as horizontal segments along chromosomes, with height or color indicating gain (amplification) or loss (deletion)—used to identify large-scale genomic imbalances.
- **Arc diagram:** Connects two genomic locations with curved arcs to show structural variants (translocations, inversions) or gene fusions—effective for visualizing rearrangements within or between chromosomes.
- **Circos plot:** A circular layout showing all chromosomes around the perimeter with arcs connecting fusion partners or translocations across the genome—provides a whole-genome view of structural complexity.
- **Heatmap:** A matrix visualization where rows are genes/features, columns are samples, and color intensity represents expression level or methylation—excellent for clustering and identifying patterns across cohorts.
- **Violin plot:** Combines a box plot with a kernel density plot to show the distribution shape of expression values—useful for comparing distributions across sample groups while showing data density.

#### Clinical Significance:

- **Driver mutations:** Cause cancer (TP53, KRAS)
- **Passenger mutations:** No functional effect
- **Actionable mutations:** Drug targets available
- **Resistance mutations:** Cause treatment failure

---

## Q6: What is a variant allele frequency (VAF) and why does it matter?

#### Answer:

VAF = Variant reads / Total reads at position

Example: 50 variant reads / 100 total reads = 0.5 (50% VAF)

#### Interpretation:

VAF Range	Interpretation
~50%	Heterozygous germline variant
~100%	Homozygous or LOH
25-50%	Likely somatic, clonal
5-25%	Subclonal mutation
<5%	Minor clone or artifact

### Why it matters:

- Distinguishes germline vs somatic
- Estimates tumor purity
- Tracks clonal evolution
- Monitors treatment response (ctDNA)

## Q7: Explain the concept of tumor heterogeneity.

### Answer:

Tumors are not homogeneous - they contain diverse subpopulations:

#### Primary Tumor

- ├─ Clone A (60%) – Original driver mutation
  - │ └─ Subclone A1 (30%) – Acquired resistance mutation
  - │ └─ Subclone A2 (20%) – Different growth advantage
- └─ Clone B (40%) – Alternative driver

### Types:

1. **Intratumoral:** Different regions of same tumor
2. **Intertumoral:** Different tumors in same patient
3. **Temporal:** Changes over time/treatment

### Visualization approaches:

- Fishplot for temporal evolution
  - Scatter plots for spatial heterogeneity
  - Phylogenetic trees for clonal relationships
- 

## Q8: What are oncogenes vs tumor suppressors?

**Answer:**

Feature	Oncogene	Tumor Suppressor
Normal function	Promotes growth	Inhibits growth
Mutation effect	Gain of function	Loss of function
Alleles affected	One (dominant)	Both (recessive)
Examples	KRAS, MYC, BRAF	TP53, RB1, BRCA1
Visualization	Hotspot mutations	Distributed mutations

**"Two-hit hypothesis"** for tumor suppressors:

1. First hit: Inactivating mutation
2. Second hit: LOH, deletion, or second mutation

**Lollipop plot patterns:**

- Oncogene: Spikes at specific hotspots
  - Tumor suppressor: Distributed across gene (truncating)
- 

## Q9: What is the significance of TP53?

**Answer:**

TP53 is the "guardian of the genome":

**Functions:**

- Cell cycle arrest
- DNA repair activation
- Apoptosis induction
- Senescence

#### **Cancer statistics:**

- Mutated in ~50% of all cancers
- Most common mutation: R175H (DNA binding domain)
- Li-Fraumeni syndrome: Germline TP53 mutations

#### **Visualization relevance:**

- TP53 is often the example gene in ProteinPaint demos
  - Shows clear domain structure (TAD, DBD, OD)
  - Hotspot mutations at codons 175, 248, 273
- 

## **Q10: What are St. Jude's "Home Team" algorithms and why do they matter?**

#### **Answer:**

St. Jude has developed specialized algorithms for pediatric cancer analysis. Understanding these shows you've done deep research into their computational ecosystem.

---

### **CICERO (Gene Fusion Detection from RNA-seq)**

**What it does:** Detects gene fusions from RNA-seq data with high sensitivity for complex pediatric alterations.

**Why it matters:** In pediatric cancer (especially leukemia), **fusions are the main drivers**—not point mutations like in adult cancers. Standard fusion callers often miss the complex internal tandem duplications found in pediatric leukemia.

#### **How it works:**

```
// CICERO looks for "Soft-Clipped Reads"
// Where left half matches Gene A and right half matches Gene B

const softClippedSignature = {
  read: 'ATCGATCG|GCTAGCTA', // | = breakpoint
  leftAlignment: { gene: 'KMT2A', position: 'chr11:118,353,109' },
  rightAlignment: { gene: 'MLLT3', position: 'chr9:20,354,218' },
  result: 'KMT2A-MLLT3 fusion (common in infant ALL)',
};

// Detection steps:
// 1. Find reads with soft-clipping (CIGAR: 50M50S)
// 2. Re-align the clipped portion to find fusion partner
// 3. Local assembly to confirm breakpoint
// 4. Filter artifacts and germline SVs
```

**Interview drop:** "I know St. Jude relies on CICERO because standard fusion callers often miss the complex internal tandem duplications found in pediatric leukemia, like FLT3-ITD."

---

## CREST (Structural Variation Detection from WGS)

**What it does:** Finds structural variations (translocations, inversions, large deletions) using soft-clipping signatures at base-pair resolution.

**How it works:**

```

// CREST uses soft-clipping patterns to find SV breakpoints
function detectSVBreakpoint(reads) {
  const softClippedReads = reads.filter(r =>
    r.cigar.includes('S') && r.mapq >= 20
  );

  // Cluster reads by clipping position
  const clusters = clusterByPosition(softClippedReads, maxDistance=5);

  // For each cluster, extract clipped sequence and re-align
  for (const cluster of clusters) {
    const clippedSeq = extractClippedSequence(cluster);
    const remapping = alignToGenome(clippedSeq);

    if (remapping.chr !== cluster.chr ||
      Math.abs(remapping.pos - cluster.pos) > 1000000) {
      yield {
        type: 'TRANSLOCATION',
        breakpoint1: { chr: cluster.chr, pos: cluster.pos },
        breakpoint2: { chr: remapping.chr, pos: remapping.pos }
      };
    }
  }
}

```

**Difference from CONCERTING:** CREST is famous for **translocations** (chromosomes swapping parts), while CONCERTING focuses on **copy number** (deletions/duplications).

---

## NetBID (Hidden Driver Discovery)

**The problem:** Sometimes a gene causes cancer not because it's **mutated**, but because it's **overactive** (too much protein). You can't see this in DNA—only in RNA/network analysis.

**The solution:** NetBID builds gene interaction networks to find "hidden drivers" that standard mutation callers miss.

```
// NetBID concept: Find drivers through network activity
const netBIDApproach = {
  // 1. Build regulatory network
  network: 'Which transcription factors regulate which genes?',

  // 2. Infer driver activity from target gene expression
  inference: `If TF-X's target genes are all upregulated,
             TF-X is probably hyperactive (even if not mutated)`,

  // 3. Example finding
  discovery: {
    gene: 'MYC',
    mutationStatus: 'Wild-type (no mutations)',
    networkActivity: 'Highly active (all MYC targets upregulated)',
    conclusion: 'MYC is a hidden driver via upstream activation',
  },
};
```

---

## The St. Jude WGS Philosophy

Approach	Standard (Most hospitals)	St. Jude
<b>Sequencing</b>	WES (exome only, ~1% DNA)	WGS (whole genome)
<b>Cost</b>	~\$500/sample	~\$1,500/sample
<b>Why?</b>	Cheaper, finds most coding mutations	Pediatric drivers often in non-coding regions

### Why St. Jude prioritizes WGS:

- Pediatric cancers often have "quiet" exomes (few mutations)
  - Dangerous drivers hide in non-coding regions (enhancers, promoters)
  - Complex structural rearrangements (chromothripsis) require WGS
  - Regulatory element disruption is invisible to WES
-

# Mutational Signatures (COSMIC)

**The concept:** Cancer leaves a "fingerprint" on the genome based on what caused it.

```
const mutationalSignatures = {
  // Each mutagen leaves specific patterns
  signatures: {
    SBS4: { cause: 'Tobacco smoking', pattern: 'C>A mutations' },
    SBS7: { cause: 'UV light exposure', pattern: 'C>T at dipyrimidines' },
    SBS11: { cause: 'Temozolomide chemotherapy', pattern: 'C>T transitions' },
    SBS31: { cause: 'Platinum chemotherapy', pattern: 'T>A mutations' },
  },

  // St. Jude survivorship context
  survivorshipQuestion: `
    If a childhood cancer survivor develops a SECOND cancer 20 years later,
    was it caused by their original chemotherapy?

    Answer: Check mutational signature!
    - If SBS31 (platinum) is dominant → likely therapy-related
    - If no chemo signatures → likely independent primary
  `,
};
```

**Why this matters for survivorship:** The St. Jude Survivorship Portal tracks 7,700+ survivors. Mutational signatures help distinguish "natural" second cancers from therapy-induced ones.

---

## The St. Jude Ecosystem

Platform	Purpose
St. Jude Cloud	Cloud infrastructure (Azure/DNAnexus) for PCGP data
PeCan	Pediatric Cancer knowledge base - curated mutations
ProteinPaint	Visualization layer for all portals

**Interview context:** When analyzing a mutation, St. Jude researchers:



1. Call variants with standard tools (GATK, Strelka)
  2. Detect fusions with **CICERO**
  3. Find SVs with **CREST**
  4. Identify CNVs with **CONSERING**
  5. Check **PeCan** for known pediatric cancer associations
  6. Visualize in **ProteinPaint**
- 

### Summary Cheat Sheet:

Tool/Technique	What it targets	Why St. Jude loves it
<b>CICERO</b>	Gene Fusions (RNA)	Fusions drive pediatric leukemia; standard tools miss them
<b>CREST</b>	Translocations (DNA)	Maps complex genome shattering (chromothripsis)
<b>CONSERING</b>	Copy Number (DNA)	Finds deletions/duplications at high resolution
<b>NetBID</b>	Hidden Drivers	Finds overactive genes without mutations
<b>WGS</b>	Whole genome	Pediatric drivers often outside coding regions
<b>Mutational Signatures</b>	DNA "Scars"	Distinguishes natural vs therapy-induced cancers

---

## Genomic File Formats

### Q11: Compare VCF, MAF, and BED file formats.

Answer:

Format	Purpose	Key Fields	Use Case
<b>VCF</b>	Variant calls	CHROM, POS, REF, ALT, QUAL, INFO	Raw variant data
<b>MAF</b>	Mutation annotation	Hugo_Symbol, Variant_Classification, HGVS_p	Cancer portals (GDC)
<b>BED</b>	Genomic intervals	chrom, start, end, name	Regions, peaks, tracks

# VCF Example

```
#CHROM POS ID REF ALT QUAL FILTER INFO
chr17 7577538 . G A 99 PASS DP=100;AF=0.45
```

# MAF Example (tab-separated)

```
Hugo_Symbol Chromosome Start_Position Variant_Classification HGVS_p_Short
TP53 chr17 7577538 Missense_Mutation p.R175H
```

# BED Example

```
chr17 7571720 7590868 TP53 1000 -
```

## VCF INFO field parsing:

```
function parseVcfInfo(infoString) {
  const info = {};
  infoString.split(';').forEach((pair) => {
    const [key, value] = pair.split('=');
    info[key] = value || true; // Flags have no value
  });
  return info;
}
// "DP=100;AF=0.45;SOMATIC" -> { DP: "100", AF: "0.45", SOMATIC: true }
```

## Q12: What are BigWig and BigBed formats? Why are they used?

**Answer:**

**Binary indexed formats** for efficient random access:

Format	Text Equivalent	Use Case
BigWig	WIG/bedGraph	Continuous signal (coverage, ChIP-seq)
BigBed	BED	Annotation features
BAM	SAM	Aligned reads
tabix	VCF/BED	Indexed text files

### Why binary formats?

Text VCF (1GB) vs Indexed VCF + tabix:

- Full scan: ~30 seconds vs ~0.1 seconds for region query (varies by file size/region)
- Memory: Load entire file vs stream relevant blocks
- Caveat: Index build is one-time cost; gains depend on query pattern

### How they work:

1. **R-tree index:** Spatial index for genomic coordinates
2. **Block compression:** zlib-compressed data blocks
3. **Summary levels:** Pre-computed zoom levels

### Reading BigWig in JavaScript:

```
// Using @gmod/bbi library
import { BigWig } from '@gmod/bbi';

const bw = new BigWig({ path: 'coverage.bw' });
const features = await bw.getFeatures('chr17', 7571720, 7590868);
// Returns: [{ start, end, score }, ...]
```

## Q13: How do you handle streaming large genomic files?

**Answer:**

**ProteinPaint uses Rust for performance-critical file operations.** The platform has compiled Rust binaries for:

Rust Binary	Purpose
<code>bigwig</code>	Reading BigWig files (local and remote URLs)
<code>gdcmaf</code>	Streaming GDC MAF files
<code>readHDF5</code>	Reading HDF5 expression matrices
<code>DEanalysis</code>	Differential expression on large matrices
<code>fisher</code>	Fisher's exact test with FDR correction

### Why Rust?

- **Often 10-100x faster** than JavaScript for compute-intensive loops (magnitude depends on workload)
- **Memory efficient** - no garbage collection pauses during hot paths
- **Native streaming** - efficient piping to/from Node.js
- **HDF5 support** - reading compressed expression data (no good JS library)

 **Verify in Repo (Turns "I read this" → "I validated this"):**

What to Find	Where to Look	What You'll See
Rust source code	<code>rust/</code> directory	Cargo.toml, src/*.rs files
Node.js bindings	<code>@sjcrh/proteinpaint-rust</code> package	<code>run_rust()</code> , <code>stream_rust()</code> exports
Binary invocation	<code>server/src/</code> handlers	Pattern: <code>run_rust('bigwig', args)</code>
Build process	<code>rust/Cargo.toml</code>	Dependencies, build targets

```
# Example: Find where bigwig binary is called
grep -r "run_rust.*bigwig" server/src/

# Example: See Rust source for fisher test
cat rust/src/fisher.rs
```

## Calling Rust from Node.js:

```
import { run_rust, stream_rust } from '@sjcrh/proteinpaint-rust';

// Run Rust binary with input data
const result = await run_rust('bigwig', `${file},${chr},${start},${stop},${points}`);

// Stream large files (e.g., GDC MAF download)
const { rustStream, endStream } = stream_rust('gdcmaf', JSON.stringify(arg));
rustStream.pipe(response);
```

## Node.js streaming for VCF:

```

// Node.js streaming for large VCF
const fs = require('fs');
const readline = require('readline');
const zlib = require('zlib');

async function* streamVcf(path) {
  const fileStream = fs.createReadStream(path);
  const gunzip = path.endsWith('.gz') ? zlib.createGunzip() : null;

  const rl = readline.createInterface({
    input: gunzip ? fileStream.pipe(gunzip) : fileStream,
  });

  for await (const line of rl) {
    if (line.startsWith('#')) continue; // Skip headers
    yield parseVcfLine(line);
  }
}

// Process without loading entire file
for await (const variant of streamVcf('large.vcf.gz')) {
  if (variant.gene === 'TP53') {
    processVariant(variant);
  }
}

```

## Browser streaming with Fetch API:

```
// Note: Using async generator (function*) because we use yield
async function* streamRemoteFile(url) {
  const response = await fetch(url);
  const reader = response.body.getReader();
  const decoder = new TextDecoder();

  let buffer = '';
  while (true) {
    const { done, value } = await reader.read();
    if (done) break;

    buffer += decoder.decode(value, { stream: true });
    const lines = buffer.split('\n');
    buffer = lines.pop(); // Keep incomplete line

    for (const line of lines) {
      yield parseLine(line);
    }
  }
}

// Usage:
// for await (const record of streamRemoteFile(url)) { ... }
```

## Q14: Describe FASTQ, FASTA, GFF/GTF, and SRA formats.

Answer:

Format	Purpose	Structure	Use Case
<b>FASTQ</b>	Raw sequencing reads	4 lines per read: ID, sequence, +, quality	Input to alignment pipelines
<b>FASTA</b>	Reference sequences	Header line (>) + sequence	Genome references, protein

Format	Purpose	Structure	Use Case
			sequences
<b>GFF/ GTF</b>	Gene annotations	9 columns: seqid, source, type, start, end, score, strand, phase, attributes	Gene models, exons, transcripts
<b>SRA</b>	Archived raw data	Compressed binary	NCBI data storage/retrieval

### FASTQ Format:

```
@SRR123456.1 HWI-ST1234:100:ABC:1:1:1234:5678
ATCGATCGATCGATCGATCGATCGATCGATCGATCG
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

- **Line 1:** Read identifier (starts with @)
- **Line 2:** DNA sequence
- **Line 3:** Separator (+)
- **Line 4:** Quality scores (Phred+33 encoded, F = Q37 = 99.98% accuracy)

### FASTA Format:

```
>chr17 Human chromosome 17
ATGCGATCGATCGATCGATCGATCGATCGATCGATCG
GATCGATCGATCGATCGATCGATCGATCGATCGATCG
```

- Simple header + sequence format
- Used for reference genomes, proteins, contigs
- Often indexed with `.fai` (samtools faidx) for random access

### GFF3/GTF Format:



```
# GFF3 Example
chr17  HAVANA  gene    7668421  7687490  .  -  .  ID=ENSG00000141510;Name=TP53
chr17  HAVANA  exon    7687377  7687490  .  -  .  Parent=ENST00000269305

# GTF Example (slightly different attributes)
chr17  HAVANA  exon    7687377  7687490  .  -  .  gene_id "ENSG00000141510"; transcript_id
```

- **GFF3**: General Feature Format version 3, hierarchical with ID/Parent
- **GTF**: Gene Transfer Format, used by Ensembl/GENCODE, key-value attributes

### SRA (Sequence Read Archive):

```
# Download from NCBI SRA
prefetch SRR123456

# Convert to FASTQ
fastq-dump --split-files SRR123456
# or newer:
fasterq-dump SRR123456
```

- Compressed archive format for raw sequencing data
- Use SRA Toolkit to download and convert
- Essential for accessing public datasets (GEO, dbGaP)

### Why These Matter for ProteinPaint:

- **FASTQ**: Starting point of all NGS analysis pipelines
  - **FASTA**: Reference genome files for alignment
  - **GFF/GTF**: Gene annotations shown in genome browser tracks
  - **SRA**: Source of public cancer genomics datasets
-



# Reference Genomes & Coordinates

## Q15: Explain hg19 vs hg38 and why it matters.

**Answer:**

Aspect	hg19 (GRCh37)	hg38 (GRCh38)
Release	2009	2013
Contigs	~93	~195
Gaps	More gaps	Fewer gaps
Centromeres	Modeled	Better resolved
Alt loci	None	261 alternate loci

**Why it matters:**

TP53 coordinates:

- hg19: chr17:7,571,720–7,590,868
- hg38: chr17:7,668,421–7,687,569

Same gene, different coordinates!

**Liftover process:**

```
// Conceptual liftover
async function liftover(position, fromGenome, toGenome) {
  // Uses chain files with alignment blocks
  const chain = await loadChainFile(`${fromGenome}To${toGenome}.chain`);

  for (const block of chain.blocks) {
    if (block.contains(position)) {
      return block.transform(position);
    }
  }
  return null; // No mapping (deleted region)
}
```

**Best practice:** Always store genome version with coordinates!

## Q16: What are genomic coordinate systems?

**Answer:**

System	Start	First Base	Used By
0-based, half-open	0	[0,1)	BED, BAM, Python
1-based, closed	1	[1,1]	VCF, GFF, GenBank

Sequence: A T G C

0-based:    0 1 2 3    (positions between bases)

1-based:    1 2 3 4    (positions of bases)

"First 2 bases" (AT):

– 0-based: [0, 2)    ← start=0, end=2

– 1-based: [1, 2]    ← start=1, end=2

**Conversion formulas:**

```
// 1-based to 0-based
const zeroBased = { start: oneBased.start - 1, end: oneBased.end };

// 0-based to 1-based
const oneBased = { start: zeroBased.start + 1, end: zeroBased.end };
```

### Common pitfall:

```
// WRONG: Mixing coordinate systems
const vcfPos = 7577538; // 1-based
const bedStart = vcfPos; // Should be vcfPos - 1!
```

---

## Q17: How do you convert between transcript and protein coordinates?

### Answer:

Gene structure:

Exon1 (100bp) -- Intron -- Exon2 (200bp) -- Intron -- Exon3 (150bp)

Genomic: chr17:1000-1100 ... 1500-1700 ... 2000-2150

CDS: 1-100 ... 101-300 ... 301-450 (450bp total)

Protein: 1-150 amino acids (450/3 = 150)

### Conversion logic:

```

function genomicToProtein(genomicPos, transcript) {
  let cdsPos = 0;

  for (const exon of transcript.exons) {
    if (genomicPos >= exon.start && genomicPos <= exon.end) {
      // Within this exon
      cdsPos += genomicPos - exon.start;
      break;
    } else if (genomicPos > exon.end) {
      // Past this exon
      cdsPos += exon.end - exon.start + 1;
    }
  }

  // Account for strand
  if (transcript.strand === '-') {
    cdsPos = transcript.cdsLength - cdsPos + 1;
  }

  // CDS to protein (1-based)
  const codonNumber = Math.ceil(cdsPos / 3);
  const codonPosition = ((cdsPos - 1) % 3) + 1; // 1, 2, or 3

  return { aaPosition: codonNumber, codonPosition };
}

```

---

## D3.js & Visualization Libraries

**Q18: Explain D3's data join pattern (enter/update/exit).**

**Answer:**

```

// The three selections in a data join
const circles = svg.selectAll('circle').data(data, (d) => d.id); // Key function for object

// ENTER: New data without existing elements
circles
  .enter()
  .append('circle')
  .attr('r', 0) // Start small
  .attr('cx', (d) => xScale(d.x))
  .transition()
  .attr('r', 5); // Grow in

// UPDATE: Existing elements with matching data
circles
  .transition()
  .attr('cx', (d) => xScale(d.x)) // Move to new position
  .attr('fill', (d) => colorScale(d.category));

// EXIT: Existing elements without matching data
circles
  .exit()
  .transition()
  .attr('r', 0) // Shrink out
  .remove();

// Modern D3 v6+ shorthand with .join()
svg
  .selectAll('circle')
  .data(data, (d) => d.id)
  .join(
    (enter) =>
      enter
        .append('circle')
        .attr('r', 0)
        .call((enter) => enter.transition().attr('r', 5)),
    (update) => update.call((update) => update.transition().attr('cx', (d) => xScale(d.x))),
    (exit) => exit.call((exit) => exit.transition().attr('r', 0).remove())
  );

```

Why key functions matter:

```
// Without key: D3 matches by index (can cause wrong animations)
.data([{id: 'A'}, {id: 'B'}]) // A at index 0, B at index 1

// With key: D3 matches by id (correct object tracking)
.data(newData, d => d.id) // A stays A, even if order changes
```

Q19: What D3 scales are used in genomic visualization?

Answer:

Scale Type	Use Case	Example
scaleLinear	Continuous numeric	Genomic position → pixel
scaleLog	Wide ranges	P-values (volcano)
scaleOrdinal	Categories	Mutation type → color
scaleSequential	Continuous color	Expression heatmap
scaleDiverging	Center-diverging	Log2 fold change
scaleBand	Categorical axes	Sample names on x-axis

```
// Genomic position scale
const xScale = d3
  .scaleLinear()
  .domain([geneStart, geneEnd]) // 7571720 to 7590868
  .range([margin.left, width - margin.right]);

// Mutation count scale (log for wide range)
const yScale = d3
  .scaleLog()
  .domain([1, maxCount])
  .range([height - margin.bottom, margin.top]);

// Consequence color scale
const colorScale = d3
  .scaleOrdinal()
  .domain(['missense', 'nonsense', 'frameshift', 'splice'])
  .range(['#3498db', '#e74c3c', '#9b59b6', '#f39c12']);

// Expression heatmap (diverging)
const expressionScale = d3
  .scaleDiverging()
  .domain([-3, 0, 3]) // Z-scores
  .interpolator(d3.interpolateRdBu);
```

## Q20: Compare D3.js with other visualization libraries?

Answer:

Library	Strengths	Weaknesses	Use Case
<b>D3.js</b>	Full control, flexible	Steep learning curve	Custom genomic viz
<b>Plotly</b>	Easy, interactive	Less customizable	Quick dashboards
<b>Highcharts</b>	Polished, accessible	Commercial license	Enterprise apps



Library	Strengths	Weaknesses	Use Case
Chart.js	Simple, lightweight	Limited types	Basic charts
Vega-Lite	Declarative, concise	Less flexible	Exploratory analysis
Three.js/ WebGL	3D, performance	Complex	3D protein structures

**Why ProteinPaint uses D3:**

1. **Pixel-perfect control** for scientific accuracy
2. **Custom interactions** (brushing, zooming on genomic regions)
3. **Animation** for smooth transitions
4. **SVG + Canvas** flexibility
5. **No framework lock-in** (vanilla JS)

---

## Q21: How do you implement zoom and pan in D3?

**Answer:**

**Core Concepts:**

D3's zoom behavior ( `d3.zoom()` ) handles mouse/touch events and maintains a **transform state** with three components:

- `transform.k` - Scale factor (1 = no zoom, 2 = 2x magnification)
- `transform.x` - Horizontal translation (pixels)
- `transform.y` - Vertical translation (pixels)

**Two Approaches for Zooming:**

Approach	How It Works	Best For
Geometric zoom	Apply CSS transform to container group	Simple visualizations, maps
Semantic	Rescale data scales, redraw	Genomic browsers, axes that

Approach	How It Works	Best For
zoom	elements	update

**Geometric zoom** just transforms the SVG group - fast but text/strokes scale too.

**Semantic zoom** (used in genome browsers) rescales the x-axis and repositions elements at their new coordinates - elements maintain size while showing different genomic regions.

### Key Configuration:

- `scaleExtent([min, max])` - Limits zoom range (e.g., `[1, 100]` = 1x to 100x)
- `translateExtent([[x0,y0], [x1,y1]])` - Constrains panning bounds
- `filter()` - Control which events trigger zoom (e.g., ignore right-click)

**ProteinPaint Pattern:** The genome browser uses semantic zoom where zooming recalculates which genomic coordinates are visible, fetches data for that region, and redraws tracks at the new resolution.

```

// Basic zoom behavior
const zoom = d3
  .zoom()
  .scaleExtent([1, 100]) // Min/max zoom
  .translateExtent([
    [0, 0],
    [width, height],
  ]) // Pan limits
  .on('zoom', zoomed);

svg.call(zoom);

function zoomed(event) {
  const { transform } = event;

  // Option 1: Transform container group
  contentGroup.attr('transform', transform);

  // Option 2: Rescale axes (better for genomic viz)
  const newXScale = transform.rescaleX(xScale);

  // Update axis
  xAxisGroup.call(d3.axisBottom(newXScale));

  // Update data elements
  svg.selectAll('circle').attr('cx', (d) => newXScale(d.position));
}

// Programmatic zoom
function zoomToRegion(start, end) {
  const scale = width / (xScale(end) - xScale(start));
  const translate = -xScale(start) * scale;

  svg
    .transition()
    .duration(750)
    .call(zoom.transform, d3.zoomIdentity.translate(translate, 0).scale(scale));
}

```

---

---

## Visualization Types & Techniques

### Q22: What visualization types are commonly used in genomics?

Answer:

Visualization	Data Type	Example Tool
Lollipop Plot	Mutations on protein	ProteinPaint
Genome Browser	Multi-track data	UCSC, IGV
Heatmap	Expression matrix	Morpheus
Kaplan-Meier	Survival data	KMplot
Volcano Plot	Differential expression	EnhancedVolcano
OncoPrint	Mutation matrix	cBioPortal
Circos	Genomic relationships	Circos
Scatter (UMAP)	Single-cell clusters	Scanpy
Arc Diagram	Fusions/SVs	ProteinPaint
Coverage Track	Sequencing depth	IGV

---

### Lollipop Plot

**What it is:** Shows mutations as "lollipops" along a protein sequence, with stem height indicating frequency and head color showing mutation type.

**How it works:**

- X-axis: Amino acid positions (1 to protein length)
- Y-axis: Mutation count or frequency

- Protein domains shown as colored rectangles on backbone
- Each lollipop = one mutation position; stacked or taller for recurrent mutations

**When to use:** Identifying mutation hotspots in cancer genes (e.g., TP53 R175H, R248Q, R273H are all in the DNA-binding domain).

**ProteinPaint feature:** Interactive tooltips show sample details, clicking filters cohort.

---

## Genome Browser

**What it is:** Horizontal tracks aligned to genomic coordinates, showing multiple data types simultaneously.

**How it works:**

- Coordinate system: chromosome + position (e.g., chr17:7,571,720-7,590,868)
- Stacked tracks: genes, variants, coverage, epigenetic marks
- Zoom from whole chromosome to single base resolution
- Data fetched dynamically based on visible region

**When to use:** Exploring genomic context - "What's near this mutation? Is it in an exon? What's the coverage?"

**Key tracks:** Gene models, BAM coverage, VCF variants, BigWig signals, BED annotations.

---

## Heatmap

**What it is:** Matrix visualization with color intensity representing values (usually expression levels).

**How it works:**

- Rows: Genes (often clustered by expression similarity)
- Columns: Samples (often clustered by profile similarity)
- Color scale: Blue (low) → White (middle) → Red (high)
- Dendrograms show hierarchical clustering

**When to use:**

- Visualizing gene expression across samples
- Identifying co-expressed gene modules
- Finding sample subtypes

**Design considerations:** Z-score normalization, color scale choice, clustering method (hierarchical, k-means).

---

## Kaplan-Meier Survival Curve

**What it is:** Shows probability of survival over time for different patient groups.

**How it works:**

- X-axis: Time (days, months, years)
- Y-axis: Survival probability (1.0  $\rightarrow$  0.0)
- Step function decreases at each death event
- Censored patients (lost to follow-up) shown as tick marks
- Confidence intervals as shaded regions

**When to use:** "Does mutation X affect patient survival?" Compare mutated vs. wild-type groups.

**Statistics:** Log-rank test for significance, hazard ratio for effect size.

---

## Volcano Plot

**What it is:** Scatter plot showing statistical significance vs. magnitude of change in differential expression.

**How it works:**

- X-axis: Log<sub>2</sub> fold change (negative = downregulated, positive = upregulated)
- Y-axis: -log<sub>10</sub>(p-value) (higher = more significant)
- Thresholds create quadrants: significant up, significant down, not significant

- Color highlights genes passing both thresholds

**When to use:** Identifying differentially expressed genes between conditions (tumor vs. normal, treated vs. control).

**Key genes:** Upper corners are most interesting (large change + highly significant).

---

## OncoPrint

**What it is:** Matrix showing mutation patterns across genes and samples, used in cancer genomics.

**How it works:**

- Rows: Genes (often sorted by mutation frequency)
- Columns: Samples (often sorted by mutation burden)
- Each cell shows mutation type with different colors/shapes
- Multiple mutations in same gene shown as stacked/overlapping glyphs

**When to use:** Visualizing mutual exclusivity, co-occurrence patterns, and the mutation landscape of a cohort.

**Example insight:** TP53 and MDM2 mutations are mutually exclusive (both disrupt same pathway).

---

## Circos / Chord Diagram

**What it is:** Circular layout showing relationships between genomic regions (fusions, translocations, interactions).

**How it works:**

- Circle perimeter represents chromosomes (or genes)
- Arcs/ribbons connect related regions
- Arc thickness can encode frequency or confidence
- Inner tracks can show additional data (copy number, expression)

**When to use:** Visualizing structural variants, gene fusions, chromatin interactions (Hi-C), synteny.

**Example:** Showing all fusion partners for a gene like KMT2A (MLL) in leukemia.

---

## UMAP/t-SNE Scatter Plot

**What it is:** 2D projection of high-dimensional data (single-cell transcriptomes) preserving local structure.

**How it works:**

- Each point = one cell
- Proximity = transcriptional similarity
- Colors indicate clusters, cell types, or gene expression
- Dimensionality reduction: thousands of genes → 2 coordinates

**When to use:** Single-cell RNA-seq analysis - identifying cell populations, visualizing trajectories.

**UMAP vs t-SNE:** UMAP better preserves global structure, faster, more reproducible.

---

## Arc Diagram

**What it is:** Shows connections (fusions, SVs) as arcs above a linear genomic axis.

**How it works:**

- Horizontal axis: Genomic positions
- Arcs connect breakpoint pairs
- Arc height can indicate span or frequency
- Color encodes fusion type (in-frame, out-of-frame)

**When to use:** Visualizing structural variants, gene fusions, RNA splicing patterns.

**ProteinPaint feature:** Shows fusion transcripts with exon-level detail.



---

## Coverage Track / Depth Plot

**What it is:** Shows sequencing read depth across genomic positions.

**How it works:**

- X-axis: Genomic position
- Y-axis: Number of reads covering each base
- Often shown as filled area or bar chart
- Gaps indicate no coverage (potential deletions or unmappable regions)

**When to use:** QC (is there enough coverage?), detecting CNVs (coverage dips/spikes), checking variant support.

**Expected values:** WGS ~30-60x, WES ~100-200x, targeted ~500-1000x.

---

### Quick Reference - Choosing Visualizations:

Question → Visualization

- "Where are mutations on this protein?" → Lollipop plot
- "What's the genomic context?" → Genome browser
- "Which genes are differentially expressed?" → Volcano plot
- "How do samples cluster?" → Heatmap + dendrogram
- "Does this mutation affect survival?" → Kaplan-Meier curve
- "What's the mutation landscape?" → OncoPrint
- "What structural variants exist?" → Arc diagram / Circos
- "What cell types are present?" → UMAP scatter plot
- "Is there enough sequencing coverage?" → Coverage track

---

## Q23: How do you design a lollipop plot?

**Answer:**

```

// Lollipop plot anatomy
function renderLollipopPlot(svg, mutations, protein) {
  // 1. Protein backbone
  svg
    .append('rect')
    .attr('class', 'protein-backbone')
    .attr('x', xScale(0))
    .attr('width', xScale(protein.length) - xScale(0))
    .attr('height', 10)
    .attr('fill', '#ddd');

  // 2. Protein domains
  svg
    .selectAll('.domain')
    .data(protein.domains)
    .join('rect')
    .attr('x', (d) => xScale(d.start))
    .attr('width', (d) => xScale(d.end) - xScale(d.start))
    .attr('fill', (d) => domainColors[d.type]);

  // 3. Lollipop stems (vertical lines)
  svg
    .selectAll('.stem')
    .data(mutations)
    .join('line')
    .attr('x1', (d) => xScale(d.position))
    .attr('x2', (d) => xScale(d.position))
    .attr('y1', backboneY)
    .attr('y2', (d) => yScale(d.count));

  // 4. Lollipop heads (circles)
  svg
    .selectAll('.head')
    .data(mutations)
    .join('circle')
    .attr('cx', (d) => xScale(d.position))
    .attr('cy', (d) => yScale(d.count))
    .attr('r', (d) => Math.sqrt(d.count) * 3) // Area proportional
    .attr('fill', (d) => consequenceColors[d.consequence]);
}

```

```
}
```

### Key design decisions:

- Circle size: Encode count/frequency
  - Circle color: Encode mutation type
  - Y-axis: Linear or log scale?
  - Overlapping: Cluster or spread?
- 

## Q24: What are t-SNE and UMAP, and when do you use them?

### Answer:

In genomics and bioinformatics, **t-SNE** and **UMAP** are the two most popular techniques for visualizing high-dimensional data like single-cell transcriptomes or methylation profiles.

### The Core Problem They Solve:

You have 10,000 cells. For each cell, you measured expression of 20,000 genes. You cannot plot a 20,000-dimensional graph. These algorithms compress that to just 2 dimensions (X, Y coordinates) while preserving meaningful relationships.

---

## t-SNE (t-Distributed Stochastic Neighbor Embedding)







**The Analogy:** Imagine a deflated beach ball (high-dimensional data) that you want to flatten onto a table (2D). t-SNE cuts the ball into tiny pieces and arranges them so points that were close on the ball remain close on the table—but it stops caring about points that were far apart.

### How It Works:

1. Calculate probability that two points are neighbors in high-dimensional space
2. Create a random soup of points in 2D space
3. Iteratively move 2D points to match high-dimensional neighbor probabilities

4. Uses Student's t-distribution (hence the "t") to handle crowding

### Characteristics:

-  Creates very distinct, well-separated clusters
  -  Excellent for identifying cell populations
  -  **Preserves local structure only** (neighbors stay together)
  -  **Distorts global structure** (distance between distant clusters is meaningless)
  -  Slow for large datasets (>50,000 points)
  -  Non-deterministic (different runs give different layouts)
- 







## UMAP (Uniform Manifold Approximation and Projection)

**The Analogy:** UMAP also flattens the beach ball, but tries to keep its overall shape intact. It doesn't just cut it into pieces—it stretches and pulls the fabric to fit the table while maintaining the topology.

### How It Works:

1. Build a "fuzzy" graph of connections between points in high dimensions
2. Uses concepts from topology (mathematics of shapes)
3. Optimize a low-dimensional graph to match the high-dimensional structure
4. Preserves both local neighborhoods AND global relationships

### Characteristics:

-  **Much faster** than t-SNE (handles millions of cells)
  -  **Preserves global structure** (cluster distances are meaningful)
  -  More reproducible with `random_state` parameter
  -  Better for trajectory analysis (pseudotime)
  -  Works well for very large datasets
  -  Slightly less distinct cluster separation than t-SNE
-

## Comparison: PCA vs t-SNE vs UMAP

Feature	PCA	t-SNE	UMAP
Type	Linear (rotates data)	Non-linear (warps data)	Non-linear (warps data)
Speed	⚡ Blazing fast	🐢 Slow	🚀 Fast
Global	✅ Preserves (great QC)	❌ Distorts	✅ Preserves
Local	❌ Limited	✅ Excellent	✅ Excellent
Clusters	Often overlapping	Very distinct	Distinct with context
Trajectories	Poor	Poor	✅ Good for pseudotime
Deterministic	✅ Yes	❌ No	⚠️ With seed
When to use	First pass, QC, overview	Older papers, clustering	<b>Modern standard</b>

---

## ProteinPaint/MB-meta Context

The **MB-meta portal** uses **t-SNE for methylome visualization** with powerful interactive features:

```
// MB-meta workflow with t-SNE
const methylomeTsneWorkflow = {
  // 1. Display methylation-based t-SNE
  visualization: 't-SNE of 898 medulloblastoma samples',

  // 2. Color by molecular group or clinical variable
  colorOverlays: ['WNT', 'SHH', 'Group3', 'Group4', 'MYC_status', 'metastatic'],

  // 3. Lasso selection → define subcohort
  interaction: 'Draw polygon on t-SNE to select samples',

  // 4. Linked views update
  linkedViews: ['survival curves', 'mutation matrix', 'sample table'],

  // 5. Scientific discovery
  discovery: 'KBTBD4 mutations cluster in specific t-SNE regions',
};
```

### Key Interview Talking Point:

"In the MB-meta portal, the methylome t-SNE isn't just a static image—users can lasso-select clusters to define subcohorts, then immediately run survival analysis or view mutation patterns in that subgroup. This 'spatial selection → filtered analysis' pattern is powerful for hypothesis generation."

---

## Implementation in D3.js

```

// Rendering UMAP/t-SNE scatter plot
function renderDimensionReduction(svg, data, { width, height, colorScale }) {
  const xExtent = d3.extent(data, (d) => d.umap1);
  const yExtent = d3.extent(data, (d) => d.umap2);

  const xScale = d3
    .scaleLinear()
    .domain(xExtent)
    .range([margin.left, width - margin.right]);

  const yScale = d3
    .scaleLinear()
    .domain(yExtent)
    .range([height - margin.bottom, margin.top]);

  // Render points
  svg
    .selectAll('circle')
    .data(data)
    .join('circle')
    .attr('cx', (d) => xScale(d.umap1))
    .attr('cy', (d) => yScale(d.umap2))
    .attr('r', 3)
    .attr('fill', (d) => colorScale(d.cluster))
    .attr('opacity', 0.7)
    .on('mouseover', showTooltip)
    .on('click', selectSample);

  // Add lasso selection (MB-meta pattern)
  const lasso = d3
    .lasso()
    .items(svg.selectAll('circle'))
    .targetArea(svg)
    .on('end', handleLassoSelection);

  svg.call(lasso);
}

function handleLassoSelection(selectedItems) {

```



```
const selectedSamples = selectedItems.data().map((d) => d.sampleId);  
// Update linked views with subcohort  
updateSurvivalCurve(selectedSamples);  
updateMutationMatrix(selectedSamples);  
highlightInTable(selectedSamples);  
}
```

---

## Interview Strategy

When asked about dimensionality reduction:

1. **Acknowledge the landscape:** "I typically start with PCA to see the overall variance structure and identify outliers..."
  2. **Pivot to modern practice:** "...but for visualizing cell populations or patient clusters, I prefer UMAP over t-SNE because it's faster for large datasets and better preserves the global trajectory of cell development."
  3. **Show ProteinPaint awareness:** "The MB-meta portal uses t-SNE for methylome visualization with lasso selection—I find this 'spatial selection → linked analysis' pattern really powerful for exploratory work."
  4. **Technical depth:** "UMAP's advantage is preserving both local and global structure, so distances between clusters actually mean something, unlike t-SNE where only within-cluster relationships are reliable."
- 

## Q25: What is Hi-C data and how does ppHiC visualize 3D genome structure?

**Answer:**

**Hi-C** is a sequencing technology that captures how different parts of the genome **physically interact in 3D space**. When DNA is packed inside the nucleus, distant genomic regions can come into contact—Hi-C identifies these interactions, which are critical for understanding gene regulation, structural variants, and enhancer hijacking in cancer.

---

# ppHiC: ProteinPaint's Hi-C Module

ppHiC (CSBJ 2024) is ProteinPaint's web-based Hi-C visualization tool that enables interactive exploration of contact matrices without heavy local compute.

## Key Capabilities:

Feature	Description
Juicebox v7/v8/v9	Supports recent .hic file versions
Multi-resolution	Seamless transitions from whole-genome (~1Mb bins) to local (~5kb bins)
Normalization methods	KR, ICE, VC, and others
Multiple formats	Juicebox .hic files, compressed BEDs, text files
Custom annotation tracks	Gene tracks, custom BED overlays

Live demo: <https://proteinpaint.stjude.org/hic/>

---

## The 4-View Navigation System (Core Mental Model)

ppHiC implements a **linked-view state machine** where each click refines the coordinate space and data resolution:

## GENOME VIEW

(Whole-genome contact map)

- See all chromosome interactions
- Click a chr-pair block to drill down

| click chr4-chr8 block



## CHROMOSOME PAIR VIEW

(Higher resolution inter/intra-chromosomal map)

- See chr4 vs chr8 at ~100kb resolution
- Click a hotspot to zoom further

| click interaction hotspot



## DETAILED VIEW

(Zoom/pan on two selected regions)

- Fine-grained view at ~5kb resolution
- Can jump to browser-like view

| open browser mode



## GENOME BROWSER VIEW

(Two regions side-by-side)

- Intra-region + inter-region contacts
- Independent zoom/pan per region
- Gene annotations, arc tracks

**State representation:**

```
const ppHicState = {
  viewMode: 'genome' | 'chrPair' | 'detailed' | 'browser',
  chrA: 'chr4',
  chrB: 'chr8',
  regionA: { start: 170000000, end: 180000000 },
  regionB: { start: 125000000, end: 135000000 },
  resolution: 5000, // bin size in bp
  normalization: 'KR', // KR, ICE, VC
  colorScale: 'log', // log or linear
};
```

## Architecture (Job-Relevant Details)

Layer	Technology	Purpose
Backend	Node.js	Serves data, calls Juicebox for slices
Slicer	Juicebox (Java)	Extracts Hi-C data on demand
Frontend	D3.js + SVG/HTML5	Renders heatmaps and arc diagrams

**Key architectural insight:** ppHiC retrieves only the data slices needed for the current view—you never load the entire contact matrix (which can be terabytes).

```
// Backend endpoint pattern
app.get('/hic/slice', async (req, res) => {
  const { chrA, chrB, startA, endA, startB, endB, resolution, norm } = req.query;

  // Call Juicebox to extract just this region
  const slice = await juicebox.getSlice(hicFile, {
    chr1: chrA, chr2: chrB,
    pos1: [startA, endA],
    pos2: [startB, endB],
    binSize: resolution,
    normalization: norm
  });

  res.json(slice);
});

// Frontend: render on zoom/pan
function onViewChange(newBounds) {
  const slice = await fetch(`/hic/slice?${buildQuery(newBounds)}`);
  renderHeatmap(slice);
}
```

---

## The Science Demo: Enhancer Hijacking in Neuroblastoma

The ppHiC paper uses neuroblastoma cell line NB69 to demonstrate how Hi-C visualization reveals mechanistic insights:

### Discovery workflow:

1. **Genome View:** Notice unusually strong chr4–chr8 inter-chromosomal contacts (shouldn't normally exist)
2. **Chromosome Pair View:** Identify the specific regions involved—chr8 ~128Mb (near MYC) and chr4 ~174Mb (near HAND2)
3. **Browser View:** See that a **t(4;8) translocation** has connected the MYC oncogene to the HAND2 enhancer
4. **Biological interpretation:** This is **enhancer hijacking**—MYC has "stolen" a powerful

enhancer from another chromosome, driving its overexpression and cancer

```
// The discovery pattern
const nb69Discovery = {
  observation: 'Strong chr4–chr8 contacts in genome view (abnormal)',
  drillDown: 'Focused on MYC region (chr8:128Mb) ↔ HAND2 region (chr4:174Mb)',
  finding: 't(4;8) translocation connecting MYC to HAND2 enhancer',
  mechanism: 'Enhancer hijacking – MYC gains access to HAND2 regulatory elements',
  cancerRelevance: 'Explains MYC overexpression without gene amplification',
};
```

**Interview talking point:** "The ppHiC neuroblastoma example shows how turning a weird heatmap into a mechanistic explanation requires both the right visualization tools AND domain knowledge. The 4-view drill-down pattern lets researchers go from 'something's unusual on chr4-chr8' to 'MYC is hijacking the HAND2 enhancer' in a few clicks."

## Normalization Methods (Technical Depth)

Hi-C data requires normalization to account for biases:

Method	Full Name	What It Does
KR	Knight-Ruiz	Matrix balancing, most common for TAD calling
ICE	Iterative Correction & Eigenvector	Similar to KR, robust to missing data
VC	Vanilla Coverage	Simple coverage normalization

**Color scale choice:**

- **Log scale** (default): Best for seeing both strong and weak interactions
- **Linear scale**: Better when comparing absolute contact frequencies

## Why ppHiC Matters for Your Interview

```
const ppHiCRelevance = {
  demonstratesPattern: 'Linked-view state machine (core ProteinPaint philosophy)',
  techStack: 'Node.js + D3.js + external tool integration (Juicebox)',
  dataHandling: 'On-demand slicing for large datasets (terabyte Hi-C files)',
  scientificValue: 'Enables discovery of enhancer hijacking, TADs, SVs',
  recentPublication: '2024 CSBJ paper shows active development',

  interviewTalkingPoints: [
    'The 4-view navigation is a state machine where each click refines coordinates and results',
    'Backend calls Juicebox on-demand—you never load the full matrix',
    'The NB69 enhancer hijacking example shows how visualization enables mechanistic insights',
    'Same architectural patterns as other ProteinPaint modules (linked views, dynamic resource loading)',
  ],
};
```

---

## Sample Implementation: Heatmap Rendering



```

// Simplified Hi-C heatmap rendering
function renderContactHeatmap(svg, data, { width, height, colorScale }) {
  const xScale = d3.scaleLinear().domain([data.startA, data.endA]).range([0, width]);

  const yScale = d3.scaleLinear().domain([data.startB, data.endB]).range([0, height]);

  // Log color scale for contact frequency
  const color = d3
    .scaleSequentialLog()
    .domain([1, d3.max(data.contacts, (d) => d.value)])
    .interpolator(d3.interpolateReds);

  // Render bins as rectangles
  svg
    .selectAll('.bin')
    .data(data.contacts)
    .join('rect')
    .attr('x', (d) => xScale(d.binA))
    .attr('y', (d) => yScale(d.binB))
    .attr('width', xScale(data.resolution) - xScale(0))
    .attr('height', yScale(data.resolution) - yScale(0))
    .attr('fill', (d) => (d.value > 0 ? color(d.value) : '#f0f0f0'))
    .on('click', (event, d) => drillDown(d.binA, d.binB));
}

// Arc view for sparse interactions
function renderArcTrack(svg, interactions, { width, yOffset }) {
  const xScale = d3.scaleLinear().domain([regionStart, regionEnd]).range([0, width]);

  const arc = d3
    .arc()
    .innerRadius(0)
    .outerRadius((d) => Math.sqrt(d.score) * 20);

  svg
    .selectAll('.arc')
    .data(interactions)
    .join('path')
    .attr('d', (d) => {

```

```

    const x1 = xScale(d.pos1);
    const x2 = xScale(d.pos2);
    const midX = (x1 + x2) / 2;
    const height = Math.abs(x2 - x1) / 2;
    return `M ${x1} ${yOffset} Q ${midX} ${yOffset - height} ${x2} ${yOffset}`;
  })
  .attr('fill', 'none')
  .attr('stroke', (d) => colorByScore(d.score))
  .attr('stroke-width', 2);
}

```

## Color Theory & Accessibility

### Q26: How do you choose color scales for genomic data?

Answer:

Data Type	Scale Type	Recommended Palette
Sequential (0→high)	Sequential	Viridis, Blues
Diverging (low↔high)	Diverging	RdBu, PiYG
Categorical	Qualitative	Category10, Set2
Significance	Threshold	Gray/Red at cutoff

```

// Expression data (diverging)
const expressionScale = d3
  .scaleDiverging()
  .domain([-3, 0, 3]) // Z-score
  .interpolator(d3.interpolateRdBu);
// Low expression = Blue, High = Red

// P-values (sequential with threshold)
const pvalueScale = d3
  .scaleSequential()
  .domain([0, 5]) // -log10(p)
  .interpolator(d3.interpolateGreys);

function getPValueColor(pvalue) {
  const negLog = -Math.log10(pvalue);
  return pvalue < 0.05 ? '#e74c3c' : pvalueScale(negLog);
}

// Mutation types (categorical)
const mutationColors = {
  missense: '#3498db', // Blue
  nonsense: '#e74c3c', // Red
  frameshift: '#9b59b6', // Purple
  splice: '#f39c12', // Orange
  silent: '#95a5a6', // Gray
};

```

---

## Q27: How do you ensure visualizations are colorblind accessible?

**Answer:**

**~8% of men have color vision deficiency:**

- Deuteranopia (red-green, most common)
- Protanopia (red-green)
- Tritanopia (blue-yellow, rare)

## Strategies:

```
// 1. Use colorblind-safe palettes
const colorblindSafe = [
  '#0072B2', // Blue
  '#E69F00', // Orange
  '#009E73', // Teal
  '#CC79A7', // Pink
  '#F0E442', // Yellow
  '#56B4E9', // Sky blue
  '#D55E00', // Vermillion
];

// 2. Don't rely on color alone
svg
  .selectAll('circle')
  .attr('fill', (d) => colorScale(d.type))
  .attr('stroke', (d) => (d.selected ? 'black' : 'none'))
  .attr('stroke-width', 2);

// 3. Use patterns for categories
const patterns = {
  missense: 'url(#diagonal-stripe)',
  nonsense: 'url(#dots)',
  frameshift: 'url(#crosshatch)',
};

// 4. Provide text labels
svg
  .selectAll('text.label')
  .data(mutations.filter((d) => d.count > threshold))
  .text((d) => d.aaChange);
```

## Tools for testing:

- Chrome DevTools → Rendering → Emulate vision deficiencies
  - [Coblis Color Blindness Simulator](#)
-

## Core Technology Stack

**Q28: Explain the role of each technology in ProteinPaint's stack.**

**Answer:**

Technology	Role	Why Used
<b>JavaScript</b>	Core language	Browser native, D3 ecosystem
<b>TypeScript</b>	Type safety	Large codebase maintainability
<b>D3.js</b>	Visualization	Full control, SVG/Canvas
<b>Node.js</b>	Server runtime	JS everywhere, npm ecosystem
<b>Express.js</b>	HTTP server	Simple, middleware-based
<b>Rust</b>	Performance parsing	Memory safety, speed
<b>napi-rs</b>	Node-Rust bridge	Native module bindings
<b>wasm-bindgen</b>	Browser-Rust bridge	WASM compilation
<b>PostgreSQL</b>	Relational data	Complex queries, ACID
<b>SQLite</b>	Embedded data	Simple, file-based
<b>R</b>	Statistics	Established methods
<b>Python</b>	ML/Analysis	Scikit-learn, pandas
<b>Docker</b>	Deployment	Reproducible environment
<b>GitHub Actions</b>	CI/CD	Automated testing

---

**Q29: How does Express.js work for API development?**

**Answer:**

```
const express = require('express');
const app = express();

// Middleware
app.use(express.json()); // Parse JSON bodies
app.use(cors()); // Enable CORS
app.use(morgan('dev')); // Request logging

// Route structure for genomic API
// GET /api/genes/:symbol/mutations
app.get('/api/genes/:symbol/mutations', async (req, res) => {
  const { symbol } = req.params;
  const { consequence, minCount } = req.query;

  try {
    const mutations = await db.query(
      `
      SELECT * FROM mutations
      WHERE gene = $1
      ${consequence ? 'AND consequence = $2' : ''}
      ${minCount ? 'AND count >= $3' : ''}
      `,
      [symbol, consequence, minCount].filter(Boolean)
    );

    res.json({ gene: symbol, mutations });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

// Error handling middleware
app.use((err, req, res, next) => {
  console.error(err.stack);
  res.status(500).json({ error: 'Internal server error' });
});
```

---

## Q30: What is Zod and why use it for validation?

Answer:

```
import { z } from 'zod';

// Define schema
const MutationSchema = z.object({
  gene: z.string().min(1),
  position: z.number().int().positive(),
  ref: z.string().regex(/^([ACGT]+$/),
  alt: z.string().regex(/^([ACGT]+$/),
  consequence: z.enum(['missense', 'nonsense', 'frameshift', 'splice']),
  vaf: z.number().min(0).max(1),
  samples: z.array(z.string()).optional(),
});

// Infer TypeScript type
type Mutation = z.infer<typeof MutationSchema>;

// Runtime validation
function validateMutation(input: unknown): Mutation {
  return MutationSchema.parse(input); // Throws if invalid
}

// Safe parsing (no throw)
const result = MutationSchema.safeParse(data);
if (result.success) {
  console.log(result.data); // Typed as Mutation
} else {
  console.error(result.error.issues);
}
```

**Why Zod:**

- TypeScript-first design
- Runtime + compile-time safety
- Composable schemas
- Great error messages

---

## 💡 Technology Selection & Trade-offs

*Interview questions often probe your ability to make informed technology choices and understand trade-offs.*

### Q31: Why choose D3.js over other visualization libraries like Chart.js, Plotly, or Highcharts?

**Answer:**

**Comparison Matrix:**

Feature	D3.js	Chart.js	Plotly	Highcharts
Control Level	Complete	Limited	Medium	Medium
Learning Curve	Steep	Easy	Medium	Medium
Custom Viz	★★★★★★	★★	★★★	★★★
File Size	~250KB	~60KB	~3MB	~300KB
SVG/Canvas	Both	Canvas	Both	SVG
Genomics Fit	★★★★★★	★★	★★★	★★★
License	BSD	MIT	MIT	Commercial

**Why D3.js wins for ProteinPaint:**

1. **Complete Control:**



```
// D3 lets you build ANYTHING – no chart type limitations
// Lollipop plot? Genome browser? Custom fusion diagram? All possible.
const lollipop = svg.selectAll('.mutation')
  .data(mutations)
  .join('g')
  .each(function(d) {
    // Full control over every pixel
    d3.select(this).append('line')...
    d3.select(this).append('circle')...
  });
```

## 2. Data-Driven Binding:

```
// Chart.js approach – data goes into predefined structure
new Chart(ctx, {
  type: 'bar', // Limited to predefined types
  data: { datasets: [...] }
});

// D3 approach – YOU define how data maps to visuals
svg.selectAll('rect')
  .data(data)
  .join('rect')
  .attr('x', d => xScale(d.position)) // You decide everything
  .attr('height', d => yScale(d.count));
```

## 3. Genomics-Specific Needs:

```
// Protein domains with custom shapes – impossible in Chart.js
svg
  .selectAll('.domain')
  .data(protein.domains)
  .join('path')
  .attr('d', (d) => {
    if (d.type === 'kinase') return kinaseShape(d);
    if (d.type === 'SH2') return sh2Shape(d);
    return defaultShape(d);
  });
```

#### 4. Composability:

```
// Build complex visualizations from primitives
const brush = d3.brushX().on('brush', updateZoom);
const zoom = d3.zoom().on('zoom', updateView);
const drag = d3.drag().on('drag', updatePosition);
// Combine them as needed
```

#### When to use alternatives:

Use Case	Best Choice	Reason
Quick dashboards	<b>Chart.js</b>	Fast, simple
Scientific plots	<b>Plotly</b>	Built-in stats
Enterprise reporting	<b>Highcharts</b>	Polish, support
Custom genomic viz	<b>D3.js</b>	Full control
3D visualization	<b>Three.js</b>	WebGL 3D

**Bottom line:** D3.js is the only choice when you need to invent new visualization types, which genomics constantly requires.

---

## Q32: What are the benefits of TypeScript for a large codebase like ProteinPaint?

**Answer:**

**The Problem with Plain JavaScript:**

```
// What type is `mutation`? What properties does it have?
function renderMutation(mutation) {
  // Runtime error if mutation.position doesn't exist
  const x = scale(mutation.position);
  // Typo goes unnoticed until runtime
  const color = getColor(mutation.consequence); // Bug!
}
```

## TypeScript Solves This:

```
interface Mutation {
  gene: string;
  position: number;
  consequence: 'missense' | 'nonsense' | 'frameshift' | 'splice';
  vaf: number;
  samples?: string[];
}

function renderMutation(mutation: Mutation): void {
  const x = scale(mutation.position); // ✅ Knows position exists
  const color = getColor(mutation.consequence); // ❌ Compile error: typo caught!
}
```

## Key Benefits for ProteinPaint:

Benefit	Impact	Example
<b>Catch bugs early</b>	Compile-time vs runtime	Typos, missing properties
<b>Self-documenting</b>	Interfaces ARE documentation	<code>MutationData</code> , <code>GenomicRange</code>
<b>Refactoring safety</b>	Change propagates everywhere	Rename <code>vaf</code> → <code>variantAlleleFrequency</code>
<b>IDE support</b>	Autocomplete, hover docs	See all mutation properties
<b>Team scaling</b>	New devs understand code	Types explain intent

Benefit	Impact	Example
	faster	

## Real-World Scenarios:

### 1. API Response Typing:

```
// API response is typed – no guessing
interface GenomicAPIResponse {
  gene: string;
  mutations: Mutation[];
  metadata: {
    totalCount: number;
    page: number;
  };
}

async function fetchMutations(gene: string): Promise<GenomicAPIResponse> {
  const res = await fetch(`/api/genes/${gene}/mutations`);
  return res.json(); // TypeScript knows the shape
}
```

### 2. D3 + TypeScript:

```
// Type-safe D3 selections
const circles = svg
  .selectAll<SVGCircleElement, Mutation>('circle')
  .data(mutations, (d) => d.id)
  .join('circle')
  .attr('cx', (d) => xScale(d.position)) // d is typed as Mutation
  .attr('r', (d) => Math.sqrt(d.count)); // Autocomplete works!
```

### 3. Component Props:

```
interface LollipopPlotProps {
  gene: string;
  mutations: Mutation[];
  width?: number;
  height?: number;
  onMutationClick?: (mutation: Mutation) => void;
}

function LollipopPlot({ gene, mutations, width = 800 }: LollipopPlotProps) {
  // All props are typed, optional ones have defaults
}
```

### TypeScript Adoption Stats:

- ProteinPaint: Migrating to TypeScript
- Industry: 78% of JS developers use TypeScript (2024 survey)
- Benefits increase with codebase size

### Trade-offs:

Pro	Con
Catches bugs early	Learning curve
Better tooling	Build step required
Self-documenting	Type definition overhead
Safer refactoring	Some complex types

**My experience:** In OmicsOracle, TypeScript caught dozens of bugs during development that would have been runtime errors, especially around API responses and D3 selections.

---

## Q33: When would you choose PostgreSQL vs MongoDB for genomic data?

**Answer:**

## Quick Decision Framework:

Choose PostgreSQL When	Choose MongoDB When
Complex queries (JOINS)	Document-centric data
ACID compliance critical	Flexible schema needed
Relational data model	Rapid prototyping
Aggregation analytics	Nested/hierarchical data
Strong consistency	Horizontal scaling priority

**For Genomics/ProteinPaint → PostgreSQL wins:**

### 1. Complex Queries Are Common:

```
-- Find all TP53 mutations in patients who survived > 5 years
-- with expression data showing upregulation
SELECT m.*, e.tpm, s-survival_months
FROM mutations m
JOIN expression e ON m.sample_id = e.sample_id AND e.gene = m.gene
JOIN survival s ON m.sample_id = s.sample_id
WHERE m.gene = 'TP53'
      AND m.consequence IN ('missense', 'nonsense')
      AND s-survival_months > 60
      AND e.tpm > 10;
```

MongoDB equivalent requires multiple queries or complex aggregation pipelines.

### 2. Genomic Coordinate Queries:

```

-- PostgreSQL with range types (native support)
CREATE TABLE genomic_features (
  id SERIAL PRIMARY KEY,
  chrom VARCHAR(10),
  pos_range int4range -- Native range type!
);

-- Efficient overlap queries
SELECT * FROM genomic_features
WHERE chrom = 'chr17'
  AND pos_range && int4range(7668421, 7687490); -- TP53 region

```

### 3. JSONB for Flexibility (Best of Both Worlds):

```

-- PostgreSQL JSONB = relational + document flexibility
CREATE TABLE variants (
  id SERIAL PRIMARY KEY,
  chrom VARCHAR(10),
  pos INTEGER,
  ref VARCHAR(100),
  alt VARCHAR(100),
  info JSONB -- Flexible annotation storage
);

-- Query JSON fields with indexing
CREATE INDEX idx_variants_info ON variants USING GIN (info);

SELECT * FROM variants
WHERE info->>'consequence' = 'missense'
  AND (info->>'vaf')::float > 0.3;

```

### 4. Aggregation Analytics:

```
-- Mutation frequency by gene and cancer type
SELECT
  gene,
  cancer_type,
  COUNT(*) as mutation_count,
  AVG(vaf) as mean_vaf,
  percentile_cont(0.5) WITHIN GROUP (ORDER BY vaf) as median_vaf
FROM mutations
GROUP BY gene, cancer_type
HAVING COUNT(*) > 10
ORDER BY mutation_count DESC;
```

### When MongoDB Makes Sense:

- **Variant annotations:** Each variant has different annotation sources
- **User sessions:** Flexible session data
- **Rapid prototyping:** Schema changes frequently
- **Log data:** High-volume, write-heavy

### ProteinPaint's Approach:

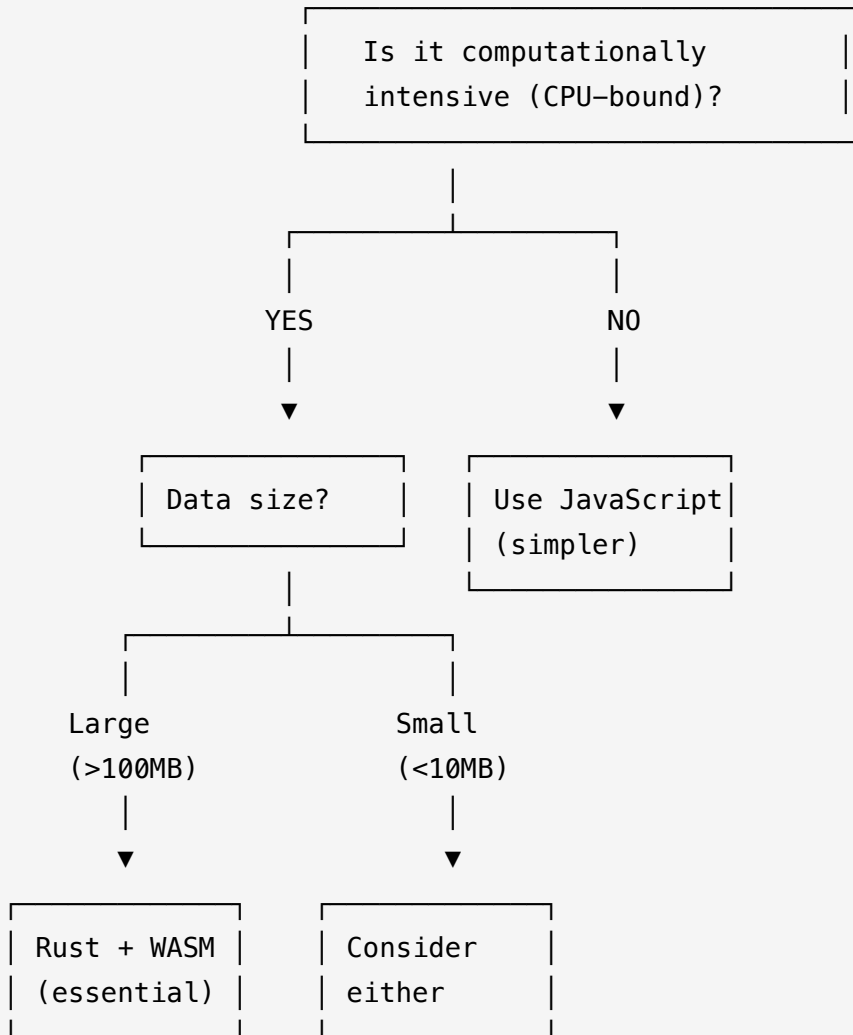
Data Type	Storage	Reason
Core genomic data	PostgreSQL	Complex queries, JOINS
File metadata	PostgreSQL	Relational
User annotations	SQLite/JSON	Simple, embedded
Session state	Redis	Fast, ephemeral

## Q34: When should you use Rust/WebAssembly vs pure JavaScript?

Answer:

Decision Framework:





### Performance Comparison:

Task	JavaScript	Rust/WASM	Speedup
VCF parsing (1GB)	45s	3s	<b>15x</b>
String operations	100ms	8ms	<b>12x</b>
Binary file reading	200ms	15ms	<b>13x</b>
D3 rendering	50ms	N/A	N/A
DOM manipulation	10ms	N/A	N/A

### Use Rust/WASM For:

## 1. File Parsing:

```
// Rust VCF parser - memory efficient, blazing fast
pub fn parse_vcf(data: &[u8]) -> Vec<Variant> {
    data.split(|&b| b == b'\n')
        .filter(|line| !line.starts_with(b"#"))
        .map(|line| parse_variant_line(line))
        .collect()
}
```

## 2. Heavy Computation:

```
// Sequence alignment, statistical calculations
pub fn calculate_pairwise_distances(sequences: &[Sequence]) -> Vec<Vec<f64>> {
    // O(n2) comparison - much faster in Rust
}
```

## 3. Binary File Processing:

```
// BigWig, BAM parsing
pub fn read_bigwig_section(data: &[u8], chrom: &str, start: u32, end: u32) -> Vec<f64> {
    // Direct memory access, no GC overhead
}
```

## Keep in JavaScript:

### 1. DOM/SVG Manipulation:

```
// D3 is already optimized for this
svg
    .selectAll('circle')
    .data(mutations)
    .join('circle')
    .attr('cx', (d) => xScale(d.position));
// WASM can't touch DOM directly anyway
```

### 2. Event Handling:

```
// Browser events are JS-native
svg.on('click', handleClick).on('mouseover', showTooltip);
```

### 3. Simple Data Transforms:

```
// Not worth WASM overhead
const filtered = mutations.filter((m) => m.vaf > 0.1);
const mapped = data.map((d) => ({ x: d.pos, y: d.value }));
```

### Integration Pattern (ProteinPaint):

```
// JavaScript orchestrates, Rust does heavy lifting
async function loadGenomicData(file) {
  // 1. Read file (JS – browser API)
  const buffer = await file.arrayBuffer();

  // 2. Parse with Rust/WASM (heavy computation)
  const parsed = await wasmParser.parseVCF(buffer);

  // 3. Transform for visualization (JS – simple)
  const vizData = parsed.map((v) => ({
    x: xScale(v.position),
    y: yScale(v.vaf),
    ...v,
  }));

  // 4. Render with D3 (JS – DOM)
  renderLollipop(vizData);
}
```

### Cost-Benefit Analysis:

Factor	JavaScript	Rust/WASM
Development time	★★★★★★	★★★★
Runtime performance	★★★★	★★★★★★

Factor	JavaScript	Rust/WASM
Memory efficiency	★★★★	★★★★★★★★
Debugging	★★★★★★★★	★★★★
Team knowledge	★★★★★★★★	★★★★
DOM interaction	★★★★★★★★	★

### My Recommendation:

"Use JavaScript by default, Rust/WASM when you measure a performance bottleneck. The 80/20 rule applies: 20% of code causes 80% of performance issues - optimize that 20% with Rust."

## 📊 Statistical Methods

### Q35: Explain the Kaplan-Meier survival analysis.

Answer:

```

// Kaplan-Meier survival probability
function kaplanMeier(events) {
  // Sort by time
  const sorted = [...events].sort((a, b) => a.time - b.time);

  let atRisk = sorted.length;
  let survival = 1.0;
  const curve = [{ time: 0, survival: 1.0 }];

  for (const event of sorted) {
    if (event.status === 1) {
      // Death/event occurred
      survival *= (atRisk - 1) / atRisk;
      curve.push({ time: event.time, survival });
    }
    atRisk--; // Decrease for both events and censoring
  }

  return curve;
}

// Log-rank test for comparing two curves
function logRankTest(group1, group2) {
  // Compare observed vs expected events
  // Returns chi-square statistic and p-value
  // p < 0.05 = significant difference
}

```

### Censoring:

- Right-censored: Patient still alive at study end
  - Left-censored: Event before observation started
  - Interval-censored: Event between two time points
-

## Q36: What is multiple testing correction and why does it matter?

**Answer:**

**The problem:**

Testing 20,000 genes at  $\alpha = 0.05$ :  
Expected false positives =  $20,000 \times 0.05 = 1,000$  genes!

**Correction methods:**

Method	Type	Formula	Use Case
Bonferroni	FWER	$\alpha/n$	Few tests, strict
Holm	FWER	Step-down	Better than Bonferroni
BH (FDR)	FDR	Ranked p-values	Many tests (genomics)
q-value	FDR	Bayesian	RNA-seq, GWAS

```
// Benjamini-Hochberg FDR correction
function adjustPvaluesBH(pvalues) {
  const n = pvalues.length;
  const sorted = pvalues.map((p, i) => ({ p, i })).sort((a, b) => b.p - a.p); // Descending

  let cumMin = 1;
  const adjusted = new Array(n);

  for (let rank = n; rank >= 1; rank--) {
    const { p, i } = sorted[n - rank];
    const adj = Math.min(cumMin, (p * n) / rank);
    cumMin = Math.min(cumMin, adj);
    adjusted[i] = adj;
  }

  return adjusted;
}
```

### Volcano plot significance:

- X-axis: log2 fold change
  - Y-axis:  $-\log_{10}(\text{adjusted p-value})$
  - Thresholds:  $|\text{FC}| > 1$  AND  $\text{padj} < 0.05$
- 

## Q37: Explain Fisher's exact test in genomics context.

**Answer:**

**Use case:** Is mutation X associated with cancer type Y?

	Cancer A	Cancer B
Mutated	10	2
Wild-type	90	98

```

// Fisher's exact test (hypergeometric distribution)
function fisherExact(a, b, c, d) {
  // a=10, b=2, c=90, d=98
  const n = a + b + c + d;

  // Calculate p-value using hypergeometric probability
  // P(X ≥ a) where X ~ Hypergeometric(n, a+c, a+b)

  // For 2x2 table:
  const pvalue = hypergeometricPvalue(a, b, c, d);

  // Odds ratio
  const oddsRatio = (a * d) / (b * c);

  return { pvalue, oddsRatio };
}

// Result interpretation:
// p < 0.05 → Significant association
// OR > 1 → Mutation enriched in Cancer A
// OR < 1 → Mutation depleted in Cancer A

```



## Data Processing & Pipelines

**Q38: Compare different sequencing technologies and their characteristics.**

**Answer:**

Technology	Read Length	Error Rate	Throughput	Best For
Illumina	150-300 bp	~0.1%	Very High	WGS, WES, RNA-seq
PacBio HiFi	10-25 kb	~0.1%	Medium	Structural variants,



Technology	Read Length	Error Rate	Throughput	Best For
				phasing
<b>Oxford Nanopore</b>	10-100+ kb	1-5%	Medium	Real-time, field work, SVs
<b>10x Genomics</b>	150 bp (linked)	~0.1%	High	Single-cell, phasing

### Why this matters for visualization:

```
// Different technologies produce different data patterns
const sequencingArtifacts = {
  illumina: {
    issues: ['GC bias', 'short reads miss SVs', 'PCR duplicates'],
    qcMetrics: ['duplication rate', 'insert size', 'GC distribution'],
  },
  pacbio: {
    issues: ['lower throughput', 'cost'],
    qcMetrics: ['read length N50', 'ZMW yield', 'subread accuracy'],
  },
  nanopore: {
    issues: ['systematic errors in homopolymers', 'basecalling quality'],
    qcMetrics: ['read length', 'Q-score distribution', 'translocation speed'],
  },
};

// Visualization implications:
// - Coverage plots need technology-appropriate binning
// - Error profiles affect variant confidence displays
// - Read length affects genome browser rendering
```

## Q39: Explain sequence alignment concepts (CIGAR, MAPQ, BAM structure).

**Answer:**

**CIGAR String** - Describes how a read aligns to reference:

CIGAR Operations:

M = Match/Mismatch (aligned)  
I = Insertion (in read, not reference)  
D = Deletion (in reference, not read)  
N = Skipped region (RNA-seq introns)  
S = Soft clip (unaligned at ends)  
H = Hard clip (removed from sequence)

Example: 50M2I30M1D18M

├ 50 bases aligned  
  ├ 2 base insertion  
    ├ 30 bases aligned  
      ├ 1 base deletion  
        └ 18 bases aligned

**MAPQ (Mapping Quality)** - Confidence that read is correctly placed:

```
// MAPQ = -10 * log10(probability of wrong mapping)
const mapqInterpretation = {
  0: 'Multiple equally good alignments (ambiguous)',
  1: '~75% confidence',
  10: '~90% confidence',
  20: '~99% confidence',
  30: '~99.9% confidence',
  60: 'Uniquely mapped with high confidence',
};

// Filtering for visualization:
function filterReads(reads, minMapq = 20) {
  return reads.filter((r) => r.mapq >= minMapq);
}
```

## BAM Structure:

```
BAM File (Binary Alignment Map)
├─ Header
│  ├── @HD: Format version
│  ├── @SQ: Reference sequences
│  └─ @RG: Read groups (sample info)
├─ Alignments (sorted by position)
│  ├── QNAME: Read name
│  ├── FLAG: Bitwise flags (paired, mapped, strand, etc.)
│  ├── RNAME: Reference chromosome
│  ├── POS: 1-based position
│  ├── MAPQ: Mapping quality
│  ├── CIGAR: Alignment description
│  ├── SEQ: Read sequence
│  └─ QUAL: Base qualities
└─ Index (.bai) for random access
```

**Why this matters:** Genome browsers need to parse CIGAR for rendering insertions, deletions, and splice junctions correctly.

---

## Soft-Clipped Reads: The Key to Fusion/SV Detection

Soft-clipping (CIGAR: `S`) is critical for understanding St. Jude's algorithms like **CICERO** and **CREST**. When an aligner can't map part of a read, it "soft clips" the unmapped portion rather than forcing a bad alignment.

```

// What soft-clipping looks like
const softClippedExample = {
  read: 'ATCGATCGATCGATCGATCGATCGATCGATCGATCG', // 40bp read
  cigar: '25M15S', // First 25bp align, last 15bp don't
  alignedPortion: 'ATCGATCGATCGATCGATCGATCG',
  clippedPortion: 'ATCGATCGATCGATCG', // This might align elsewhere!
};

// Why this happens:
// 1. Fusion breakpoint: Left half = Gene A, Right half = Gene B
// 2. Structural variant: Read spans a translocation
// 3. Adapter contamination: Read includes sequencing adapter

// CICERO/CREST concept: Parse soft-clips to find breakpoints
function detectPotentialBreakpoint(read) {
  const cigarParts = parseCIGAR(read.cigar);

  // Check for significant soft-clipping
  const softClips = cigarParts.filter((p) => p.op === 'S');

  for (const clip of softClips) {
    if (clip.length >= 10) {
      // Significant clip
      // Extract the clipped sequence
      const clippedSeq = extractClippedSequence(read, clip);

      // Try to realign the clipped portion elsewhere
      const realignment = alignSequence(clippedSeq);

      if (realignment.mapq > 20 && realignment.chr !== read.chr) {
        return {
          type: 'FUSION_CANDIDATE',
          gene1: { chr: read.chr, pos: read.pos },
          gene2: { chr: realignment.chr, pos: realignment.pos },
          supportingReads: 1,
        };
      }
    }
  }
}

```

```

    return null;
}

// CIGAR parser for visualization
function parseCIGAR(cigar) {
    const pattern = /(\d+)([MIDNSHP=X])/g;
    const ops = [];
    let match;

    while ((match = pattern.exec(cigar)) !== null) {
        ops.push({
            length: parseInt(match[1]),
            op: match[2],
            consumesRef: 'MDN=X'.includes(match[2]),
            consumesRead: 'MIS=X'.includes(match[2]),
        });
    }

    return ops;
}

// Calculate read coverage at a position (for genome browser)
function getCoverageAtPosition(reads, pos) {
    return reads.filter((read) => {
        const cigar = parseCIGAR(read.cigar);
        let refPos = read.pos;

        for (const op of cigar) {
            if (op.consumesRef) {
                if (refPos <= pos && pos < refPos + op.length) {
                    return op.op !== 'D' && op.op !== 'N'; // Not a gap
                }
                refPos += op.length;
            }
        }
        return false;
    }).length;
}

```

**Why St. Jude cares about soft-clips:**

- **Pediatric cancers have more fusions** than adult cancers
  - **CICERO** uses soft-clipped RNA-seq reads to detect gene fusions
  - **CREST** uses soft-clipped DNA reads to find structural variants
  - Standard variant callers often **ignore soft-clipped regions**
- 

## **Q40: What QC metrics do you look for in sequencing data?**

**Answer:**

**DNA-seq QC Metrics:**

```
const dnaSeqQC = {
  // Pre-alignment (FastQC)
  perBaseQuality: {
    good: 'Q30+ across all positions',
    warning: 'Drop at read ends',
    fail: 'Low quality throughout',
  },

  gcContent: {
    good: 'Normal distribution matching reference',
    warning: 'Shifted peak (contamination?)',
    fail: 'Multiple peaks (mixed samples?)',
  },

  adapterContent: {
    good: '<1% adapter sequences',
    action: 'Trim adapters if >5%',
  },

  // Post-alignment
  mappingRate: {
    good: '>95% for human WGS',
    warning: '80-95% (contamination? wrong reference?)',
    fail: '<80%',
  },

  duplicateRate: {
    wgs: '<20% typical',
    wes: '<30% typical (more PCR)',
    action: 'Mark duplicates, investigate if very high',
  },

  coverage: {
    wgs: '30-50x median for variant calling',
    wes: '100-200x on-target',
    metric: 'Evenness matters (coefficient of variation)',
  },

  insertSize: {
```

```
    typical: '300–500bp for standard libraries',
    warning: 'Bimodal distribution, very narrow/wide',
  },
};
```

## RNA-seq Specific:

```
const rnaSeqQC = {
  mappingRate: '>80% (lower than DNA due to novel junctions)',

  geneBodyCoverage: {
    good: "Even 5' to 3' coverage",
    warning: "3' bias (degradation)",
    fail: 'Extreme bias',
  },

  ribosomalContent: {
    good: '<10% rRNA reads',
    action: 'Check rRNA depletion protocol',
  },

  strandedness: 'Verify matches library prep protocol',

  duplicationRate: 'Higher acceptable than DNA-seq (expression levels)',
};
```

## Visualization of QC:

```
// Common QC visualizations to build
const qcPlots = [
  'Per-base quality boxplot',
  'GC content distribution',
  'Coverage histogram/heatmap',
  'Insert size distribution',
  'Gene body coverage (RNA)',
  'PCA of samples (batch effects)',
  'Correlation heatmap between samples',
];
```



---

## Q41: Explain the variant calling pipeline.

**Answer:**

### Standard Germline Pipeline (GATK Best Practices):

```
Raw FASTQ
|
▼ FastQC, Trimmomatic
Cleaned FASTQ
|
▼ BWA-MEM / BWA-MEM2
Aligned BAM
|
▼ Picard MarkDuplicates
Deduped BAM
|
▼ GATK BaseRecalibrator (BQSR)
Recalibrated BAM
|
▼ GATK HaplotypeCaller
Raw VCF (gVCF)
|
▼ GATK VariantRecalibrator (VQSR) or hard filtering
Filtered VCF
|
▼ VEP / SnpEff annotation
Annotated VCF
```

### Somatic Pipeline (Tumor-Normal):

```
const somaticPipeline = {
  callers: ['Mutect2', 'Strelka2', 'VarDict', 'SomaticSniper'],

  // Key difference from germline:
  approach: 'Compare tumor to matched normal',

  // Additional filters
  filters: {
    panel_of_normals: 'Remove recurrent artifacts',
    contamination: 'Estimate and filter cross-contamination',
    orientation_bias: 'FFPE artifact filtering',
    germline_resource: 'Filter known germline variants',
  },

  // Output annotations important for visualization
  annotations: {
    AF: 'Allele frequency in tumor',
    DP: 'Read depth',
    AD: 'Allele depth (ref, alt)',
    'F1R2/F2R1': 'Strand orientation counts',
  },
};
```

**Why visualization developers need to know this:**

```
// Understanding pipeline → Better visualizations
const visualizationImplications = {
  // VAF distribution plot
  vaf: 'Expect ~50% germline, variable somatic (tumor purity)',

  // Quality filtering UI
  filters: 'Need to expose FILTER field values',

  // Confidence indicators
  quality: 'Show QUAL score, DP, and caller agreement',

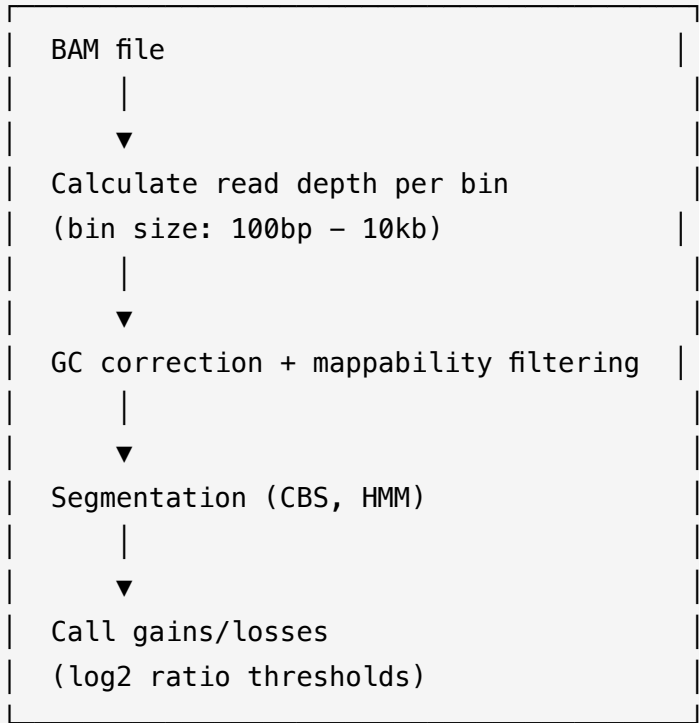
  // Annotation display
  annotations: 'Parse INFO/FORMAT fields for tooltips',
};
```

---

## Copy Number Variant (CNV) Calling

CNV detection uses different approaches than SNV/indel calling:

Coverage-based CNV Detection:



Tool	Approach	Best For
CNVkit	Hybrid capture + WGS	WES/targeted panels
GATK gCNV	Probabilistic model	WES cohorts
Control-FREEC	Read depth + BAF	Tumor/normal
ASCAT	Allele-specific	Tumor purity/ploidy
Purple	WGS tumor/normal	Comprehensive

Key CNV metrics:

- **Log2 ratio:**  $\log_2(\text{tumor\_depth} / \text{normal\_depth})$  — 0 = neutral, >0.3 = gain, <-0.4 = loss
- **BAF (B-allele frequency):** Detects LOH (loss of heterozygosity)
- **Purity/Ploidy:** Essential for interpreting absolute copy number

## **Structural Variant (SV) Calling**

SVs require different detection strategies:

```

const svDetectionStrategies = {
  // Split reads – read spans breakpoint
  splitRead: {
    tools: ['LUMPY', 'Manta', 'DELLY'],
    detects: 'Precise breakpoints',
    requires: 'Soft-clipped reads in BAM',
  },

  // Discordant pairs – abnormal insert size/orientation
  discordantPairs: {
    tools: ['LUMPY', 'BreakDancer'],
    detects: 'Large insertions, deletions, inversions',
    signature: 'Insert size >> expected OR wrong orientation',
  },

  // Read depth – coverage changes
  readDepth: {
    tools: ['CNVnator', 'cn.MOPS'],
    detects: 'Large deletions, duplications',
    limitation: 'Cannot detect balanced rearrangements',
  },

  // Long reads – span entire SV
  longRead: {
    tools: ['Sniffles', 'SVIM', 'cuteSV'],
    detects: 'All SV types with high accuracy',
    requires: 'PacBio or Nanopore data',
  },

  // Assembly-based
  assembly: {
    tools: ['GRIDSS', 'SvABA'],
    detects: 'Complex rearrangements',
    approach: 'Local assembly at breakpoints',
  },
};

// SV types and their signatures
const svTypes = {

```

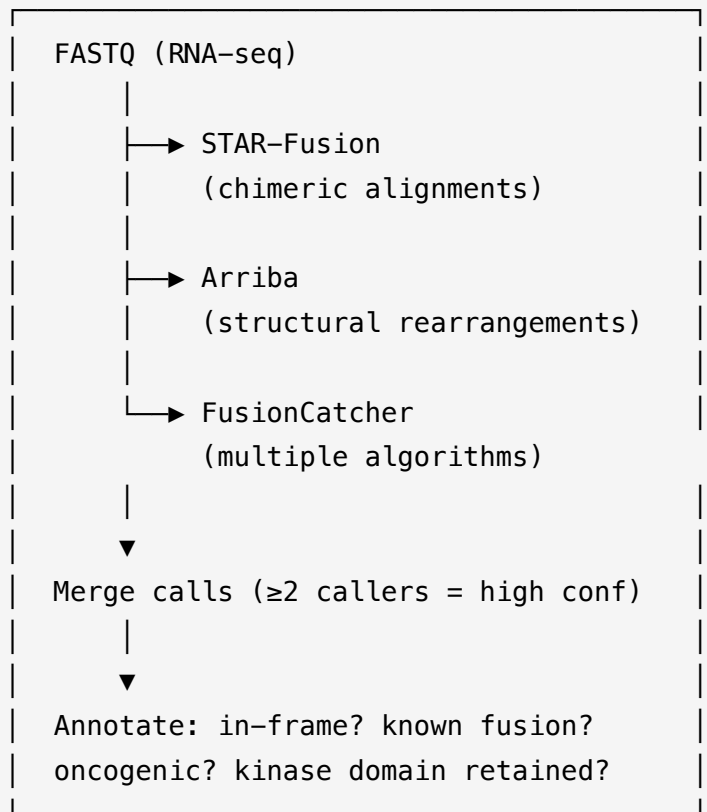
```
DEL: { discordant: 'Larger insert', depth: 'Decreased' },
DUP: { discordant: 'Tandem orientation', depth: 'Increased' },
INV: { discordant: 'Same strand orientation', depth: 'Normal' },
TRA: { discordant: 'Different chromosomes', depth: 'Normal' },
INS: { splitRead: 'Extra sequence at junction', depth: 'Normal' },
};
```

---

## Gene Fusion Detection

Fusions are critical in cancer (BCR-ABL, EML4-ALK, etc.):

RNA-seq Fusion Detection Pipeline:



### Key fusion annotations:

- **Frame status:** In-frame fusions more likely functional
- **Breakpoint location:** Kinase domain retained?
- **Known fusions:** COSMIC fusion database, FusionGDB

- **Expression level:** Read support (spanning + split reads)

#### **ProteinPaint fusion visualization:**

- Arc diagrams showing gene partners
  - Exon-level breakpoint visualization
  - Protein domain retention display
- 

### **Variant Annotation Deep Dive**

Annotation is crucial for interpretation:



```

const annotationSources = {
  // Functional prediction
  VEP: {
    provides: [
      'Gene, transcript, consequence',
      'SIFT, PolyPhen scores',
      'Regulatory features',
      'Population frequencies',
    ],
    output: 'VCF INFO field or tab-delimited',
  },

  SnpEff: {
    provides: [
      'Variant effect',
      'Gene/transcript impact',
      'Amino acid change',
      'Loss of function prediction',
    ],
    output: 'ANN field in VCF',
  },

  // Population databases
  gnomAD: {
    provides: 'Allele frequencies in ~140k individuals',
    use: 'Filter common variants (AF > 0.01)',
    subpopulations: ['NFE', 'AFR', 'EAS', 'SAS', 'AMR', 'ASJ'],
  },

  // Clinical significance
  ClinVar: {
    provides: 'Clinical interpretations',
    categories: ['Pathogenic', 'Likely pathogenic', 'VUS', 'Likely benign', 'Benign'],
    note: 'Assertions from clinical labs',
  },

  // Cancer-specific
  COSMIC: {
    provides: 'Somatic mutation frequency in cancers',
  },

```

```

    use: 'Identify known cancer mutations',
    key_field: 'COSMIC_ID, FATHMM score',
  },

  OncoKB: {
    provides: 'Oncogenic classification + drug implications',
    levels: ['Oncogenic', 'Likely Oncogenic', 'Predicted Oncogenic'],
    actionability: 'FDA-approved, clinical trials, etc.',
  },
};

// Consequence severity (VEP order)
const consequenceSeverity = [
  'transcript_ablation', // Most severe
  'splice_acceptor_variant',
  'splice_donor_variant',
  'stop_gained',
  'frameshift_variant',
  'stop_lost',
  'start_lost',
  'missense_variant',
  'inframe_insertion',
  'inframe_deletion',
  'splice_region_variant',
  'synonymous_variant',
  'intron_variant',
  'intergenic_variant', // Least severe
];

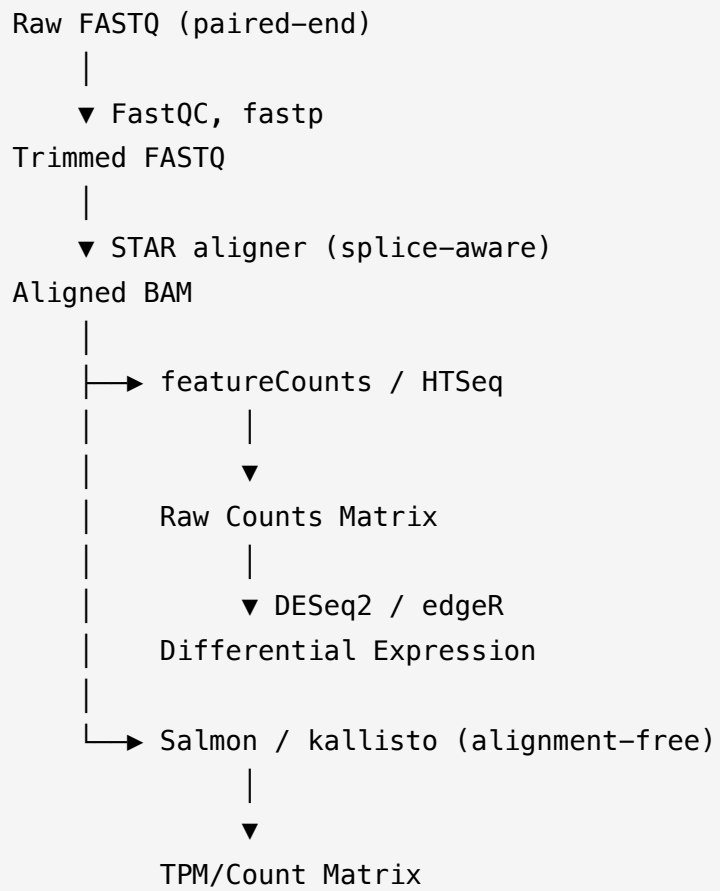
```

---

## Q42: How is RNA-seq data processed and normalized?

**Answer:**

**Processing Pipeline:**



## Normalization Methods:

```

const normalizationMethods = {
  // Within-sample normalization
  TPM: {
    name: 'Transcripts Per Million',
    formula: '(reads/gene_length) / sum(reads/length) * 1e6',
    use: 'Compare genes within sample',
    sums_to: '1 million per sample',
  },

  FPKM: {
    name: 'Fragments Per Kilobase Million',
    formula: '(fragments * 1e9) / (gene_length * total_fragments)',
    use: 'Legacy, comparable to TPM',
    note: 'TPM preferred for newer analyses',
  },

  // Between-sample normalization
  DESeq2_normalization: {
    method: 'Median of ratios',
    use: 'Differential expression analysis',
    accounts_for: 'Library size AND composition',
  },

  TMM: {
    name: 'Trimmed Mean of M-values',
    use: 'edgeR normalization',
    accounts_for: 'Library size AND composition',
  },

  // For visualization
  log2_transform: {
    formula: 'log2(TPM + 1) or log2(count + 1)',
    why: 'Expression spans orders of magnitude',
    visualization: 'Heatmaps, scatter plots',
  },

  z_score: {
    formula: '(x - mean) / std',
    use: 'Heatmap row scaling',
  },

```

```
    shows: 'Relative expression patterns',  
  },  
};
```

### Visualization considerations:

```
// Expression heatmap preprocessing  
function prepareExpressionHeatmap(countMatrix) {  
  // 1. Filter low-expressed genes  
  const filtered = filterLowCounts(countMatrix, (minCount = 10));  
  
  // 2. Normalize (TPM or DESeq2)  
  const normalized = normalizeTPM(filtered);  
  
  // 3. Log transform  
  const logged = normalized.map((row) => row.map((val) => Math.log2(val + 1)));  
  
  // 4. Z-score per gene (row)  
  const scaled = zScoreRows(logged);  
  
  // 5. Choose color scale  
  const colorScale = d3.scaleDiverging().domain([-2, 0, 2]).interpolator(d3.interpolateRdBlu);  
  
  return { data: scaled, colorScale };  
}
```

---

## Q43: How do differential expression tools work? (DESeq2, edgeR, limma)

### Answer:

Differential expression analysis identifies genes that are significantly up- or down-regulated between conditions. This is **fundamental for ProteinPaint** when visualizing cancer vs. normal comparisons or treatment responses.

---

## The Big Three Tools Comparison

Tool	Statistical Model	Best For	Speed
DESeq2	Negative Binomial + GLM	Most RNA-seq experiments	Moderate
edgeR	Negative Binomial	Small sample sizes	Fast
limma	Linear models + eBayes	Microarrays, large datasets	Very Fast

---

## Why Negative Binomial Distribution?

RNA-seq counts are NOT normally distributed—they're discrete, non-negative, and **overdispersed** (variance > mean). The Negative Binomial models this.

```
// Why Poisson isn't enough for RNA-seq
const poissonAssumption = 'variance = mean';
const realRNAseq = 'variance >> mean (overdispersion)';

// Negative Binomial adds a dispersion parameter
const negBinomial = {
  mean: 'μ (expected counts)',
  dispersion: 'α (accounts for biological variability)',
  variance: 'μ + α*μ² (captures overdispersion)',
};
```

---

# DESeq2 Workflow

```
// Conceptual DESeq2 pipeline
function runDESeq2(countMatrix, sampleInfo) {
  // 1. Size factor normalization (accounts for sequencing depth)
  const sizeFactors = calculateMedianOfRatios(countMatrix);
  const normalized = countMatrix.map((row, i) => row.map((count, j) => count / sizeFactors[j]));

  // 2. Dispersion estimation (shrinks toward fitted line)
  const geneDispersions = estimateDispersions(normalized);
  const shrunkDispersions = shrinkTowardFit(geneDispersions);

  // 3. Fit negative binomial GLM
  const model = fitNBGLM(normalized, sampleInfo.condition, shrunkDispersions);

  // 4. Wald test for significance
  const results = waldTest(model);

  // 5. BH correction for multiple testing
  return adjustPValues(results, 'BH');
}
```

## Normalization Methods Compared

Method	Formula	Use Case
TPM	$(\text{count} / \text{gene\_length}) / \text{sum} * 1\text{M}$	Within-sample comparison
RPKM/FPKM	$(\text{count} * 10^9) / (\text{total} * \text{gene\_length})$	Legacy, avoid for DE
DESeq2 (MOR)	Median of Ratios	Between-sample DE analysis
TMM (edgeR)	Trimmed Mean of M-values	Between-sample DE analysis

Why not use TPM for differential expression?

```
// TPM is compositional – it sums to 1 million per sample
// If one gene goes up, others MUST go down (relative measure)

const problem = {
  sample1: { geneA: 500_000, geneB: 500_000 }, // TPM
  sample2: { geneA: 900_000, geneB: 100_000 }, // TPM (geneA up)
  // But geneB might not actually be down!
  // Could be same counts, just diluted by geneA
};

// DESeq2's MOR uses raw counts and normalizes for library size
// This detects TRUE expression changes
```

---



## Volcano Plot: The Standard DE Visualization

```

function createVolcanoPlot(deResults, container) {
  const width = 800,
    height = 600;
  const svg = d3.select(container).append('svg').attr('viewBox', `0 0 ${width} ${height}`

  // Thresholds
  const fcThreshold = 1; // log2 fold change
  const pThreshold = 0.05; // adjusted p-value

  // Scales
  const xScale = d3
    .scaleLinear()
    .domain(d3.extent(deResults, (d) => d.log2FC))
    .range([50, width - 50]);

  const yScale = d3
    .scaleLinear()
    .domain([0, d3.max(deResults, (d) => -Math.log10(d.padj))])
    .range([height - 50, 50]);

  // Color by significance
  const getColor = (d) => {
    if (d.padj >= pThreshold) return '#999'; // Not significant
    if (d.log2FC > fcThreshold) return '#e74c3c'; // Up
    if (d.log2FC < -fcThreshold) return '#3498db'; // Down
    return '#999';
  };

  // Draw points
  svg
    .selectAll('circle')
    .data(deResults)
    .enter()
    .append('circle')
    .attr('cx', (d) => xScale(d.log2FC))
    .attr('cy', (d) => yScale(-Math.log10(d.padj)))
    .attr('r', 3)
    .attr('fill', getColor)
    .attr('opacity', 0.7)

```

```

        .on('mouseover', (event, d) => {
            showTooltip(event, `${d.gene}: FC=${d.log2FC.toFixed(2)}, padj=${d.padj.toExponential(2)}`);
        });

// Add threshold lines
svg
    .append('line')
    .attr('x1', xScale(-fcThreshold))
    .attr('x2', xScale(-fcThreshold))
    .attr('y1', 50)
    .attr('y2', height - 50)
    .attr('stroke', '#ddd')
    .attr('stroke-dasharray', '5,5');

svg
    .append('line')
    .attr('x1', xScale(fcThreshold))
    .attr('x2', xScale(fcThreshold))
    .attr('y1', 50)
    .attr('y2', height - 50)
    .attr('stroke', '#ddd')
    .attr('stroke-dasharray', '5,5');

// Horizontal p-value threshold
svg
    .append('line')
    .attr('x1', 50)
    .attr('x2', width - 50)
    .attr('y1', yScale(-Math.log10(pThreshold)))
    .attr('y2', yScale(-Math.log10(pThreshold)))
    .attr('stroke', '#ddd')
    .attr('stroke-dasharray', '5,5');

return svg.node();
}

```

---

## GSEA: Finding Pathway Enrichment

After identifying DE genes, Gene Set Enrichment Analysis (GSEA) asks: "Are my DE genes enriched in known biological pathways?"

```
// GSEA concept
const gseaApproach = {
  step1: 'Rank ALL genes by log2FC (not just significant ones)',
  step2: 'For each pathway, calculate enrichment score',
  step3: 'Walking down ranked list: +1 if gene in pathway, -1 if not',
  step4: 'Max deviation from zero = enrichment score',
  step5: 'Permutation test for significance',
};

// Example result
const gseaResult = {
  pathway: 'HALLMARK_P53_PATHWAY',
  NES: 2.3, // Normalized enrichment score
  pval: 0.001,
  interpretation: 'p53 pathway genes are significantly upregulated as a group',
};
```

---

## St. Jude Context

- **ProteinPaint integrates DE results:** Volcano plots linked to lollipop plots
- **Pediatric cancer DE:** Often comparing tumor subtypes (e.g., Group 3 vs. Group 4 medulloblastoma)
- **MB-meta portal:** Uses DE to identify subgroup-specific markers
- **Survivorship:** DE between patients with vs. without late effects

---

## Interview Strategy:

"I understand that DESeq2 uses negative binomial GLMs with dispersion shrinkage to handle the overdispersion inherent in RNA-seq count data. The volcano plot is the

standard visualization for DE results, showing log2 fold change versus significance. In ProteinPaint, I'd want to link volcano plots to mutation views—clicking an upregulated gene could show its mutation landscape in the lollipop plot."

## Q44: What multi-omics data types might you visualize?

Answer:

Data Type	What It Measures	File Formats	Visualization Types
WGS	All genomic variants	BAM, VCF, CRAM	Genome browser, circos
WES	Coding region variants	BAM, VCF	Lollipop plots, oncoplots
RNA-seq	Gene expression	BAM, counts	Heatmaps, volcano, PCA
ATAC-seq	Chromatin accessibility	BAM, BigWig, peaks	Track viewer, heatmaps
ChIP-seq	Protein-DNA binding	BAM, BigWig, peaks	Peak tracks, heatmaps
Methylation	DNA methylation	BED, beta values	Manhattan, heatmaps
Single-cell	Cell-level expression	h5ad, loom	UMAP, t-SNE, violin
Proteomics	Protein abundance	mzML, tables	Volcano, heatmaps

Integration Visualization:

```

// Multi-omics integration views
const multiOmicsViews = {
  // Same genomic region, multiple tracks
  integratedBrowser: {
    tracks: ['mutations', 'expression', 'methylation', 'chromatin'],
    challenge: 'Synchronized zoom/pan across data types',
  },

  // Sample-level integration
  multiOmicsHeatmap: {
    rows: 'Samples',
    columns: 'Features from each -omics',
    challenge: 'Different scales, normalization',
  },

  // Correlation views
  scatterMatrix: {
    x: 'Gene expression',
    y: 'Protein abundance',
    color: 'Mutation status',
    challenge: 'Missing data, different sample sets',
  },

  // Pathway-level
  pathwayView: {
    nodes: 'Genes/proteins',
    nodeColor: 'Expression',
    nodeSize: 'Mutation frequency',
    edges: 'Interactions',
  },
};

// Key challenge: Data harmonization
const integrationChallenges = [
  'Different samples across platforms',
  'Different scales and distributions',
  'Batch effects within and across platforms',
  'Missing data handling',
  'Statistical methods for integration (MOFA, iCluster)',
];

```

```
];
```

### Single-cell specific:

```
// Single-cell visualization pipeline
const singleCellViz = {
  preprocessing: [
    'Quality filtering (nGenes, %MT)',
    'Normalization (scrna, Seurat)',
    'Highly variable genes',
    'Dimensionality reduction (PCA)',
  ],

  visualizations: {
    UMAP_tSNE: 'Cluster structure, cell types',
    violinPlot: 'Gene expression by cluster',
    dotPlot: 'Marker gene expression',
    trajectory: 'Pseudotime, differentiation',
    spatialPlot: 'Spatial transcriptomics',
  },

  interactivity: [
    'Cluster selection → gene expression',
    'Lasso selection of cells',
    'Linked views (UMAP + violin)',
    'Cell type annotation',
  ],
};
```

---

## Q45: What workflow managers are used in genomics pipelines?

### Answer:

Production genomics pipelines use workflow managers for reproducibility, scalability, and portability:

<b>Workflow Manager</b>	<b>Language</b>	<b>Strengths</b>	<b>Used By</b>
<b>Nextflow</b>	Groovy/ DSL	Cloud-native, containers, nf-core	Broad adoption, nf-core community
<b>Snakemake</b>	Python	Pythonic, conda integration	Academic labs
<b>WDL</b>	Custom DSL	Broad Institute standard, Terra	GATK pipelines, Broad
<b>CWL</b>	YAML/ JSON	Portable, vendor-neutral	GA4GH standard

---

## Nextflow

Most popular for production genomics. Powers nf-core community pipelines.



```
// Example: Simple variant calling pipeline
workflow {
  // Define channels (data flows)
  reads_ch = Channel.fromFilePairs('data/*_{1,2}.fastq.gz')
  reference = file('reference/hg38.fa')

  // Pipeline steps
  FASTQC(reads_ch)
  BWA_MEM(reads_ch, reference)
  MARK_DUPLICATES(BWA_MEM.out)
  GATK_HAPLOTYPECALLER(MARK_DUPLICATES.out, reference)
}

process BWA_MEM {
  container 'biocontainers/bwa:0.7.17'
  cpus 8
  memory '32 GB'

  input:
  tuple val(sample_id), path(reads)
  path reference

  output:
  tuple val(sample_id), path("${sample_id}.bam")

  script:
  """
  bwa mem -t ${task.cpus} ${reference} ${reads[0]} ${reads[1]} | \
  samtools sort -o ${sample_id}.bam
  """
}
```

### Key Nextflow features:

- **Channels:** Reactive data streams between processes
- **Containers:** Docker/Singularity for reproducibility
- **Executors:** Local, SLURM, AWS Batch, Google Cloud
- **Resume:** Re-run from where it failed
- **nf-core:** 80+ production-ready pipelines (rnaseq, sarek, etc.)

---

# Snakemake

Pythonic, great for custom pipelines:

```
# Snakefile example
SAMPLES = ["sample1", "sample2", "sample3"]

rule all:
    input:
        expand("results/{sample}.vcf", sample=SAMPLES)

rule bwa_mem:
    input:
        r1 = "data/{sample}_1.fastq.gz",
        r2 = "data/{sample}_2.fastq.gz",
        ref = "reference/hg38.fa"
    output:
        "aligned/{sample}.bam"
    threads: 8
    conda:
        "envs/alignment.yaml"
    shell:
        "bwa mem -t {threads} {input.ref} {input.r1} {input.r2} | "
        "samtools sort -o {output}"

rule variant_calling:
    input:
        bam = "aligned/{sample}.bam",
        ref = "reference/hg38.fa"
    output:
        "results/{sample}.vcf"
    shell:
        "gatk HaplotypeCaller -R {input.ref} -I {input.bam} -O {output}"
```

## Key Snakemake features:

- **DAG-based:** Automatically determines execution order
- **Conda integration:** Environment per rule

- **Wildcards:** Pattern matching for samples
  - **Cluster support:** SLURM, PBS, LSF
- 

## **WDL (Workflow Description Language)**

Broad Institute standard, used in Terra/Cromwell:

```

version 1.0

workflow VariantCalling {
  input {
    File input_bam
    File reference
    String sample_name
  }

  call HaplotypeCaller {
    input:
      bam = input_bam,
      ref = reference,
      sample = sample_name
  }

  output {
    File vcf = HaplotypeCaller.output_vcf
  }
}

task HaplotypeCaller {
  input {
    File bam
    File ref
    String sample
  }

  command {
    gatk HaplotypeCaller \
      -R ~{ref} \
      -I ~{bam} \
      -O ~{sample}.vcf
  }

  output {
    File output_vcf = "~{sample}.vcf"
  }
}

```

```
runtime {
  docker: "broadinstitute/gatk:4.3.0.0"
  memory: "16 GB"
  cpu: 4
}
```

---

## Cloud Platforms for Genomics

Platform	Provider	Features
Terra	Broad/Google	WDL workflows, GATK best practices, AnVIL
AWS HealthOmics	Amazon	Managed workflows, storage optimization
DNAnexus	Commercial	Enterprise, FDA compliance, secure
Seven Bridges	Commercial	Cancer Genomics Cloud, NCI data

### Cost considerations:

```
const cloudCosts = {
  storage: {
    s3_standard: '$0.023/GB/month',
    s3_glacier: '$0.004/GB/month',
    tip: 'Move raw FASTQs to cold storage after processing',
  },
  compute: {
    spot_instances: '60-90% savings vs on-demand',
    tip: 'Use spot/preemptible for fault-tolerant workflows',
  },
  egress: {
    warning: 'Data transfer OUT is expensive',
    tip: 'Process data in same region as storage',
  },
};
```

---

## Why This Matters for ProteinPaint

Visualization platforms need to:

1. **Accept outputs** from standard pipelines (VCF, MAF, BigWig)
2. **Handle provenance** - track which pipeline version produced data
3. **Support batch processing** - visualize cohorts processed in parallel
4. **Cloud integration** - read from S3/GCS buckets directly

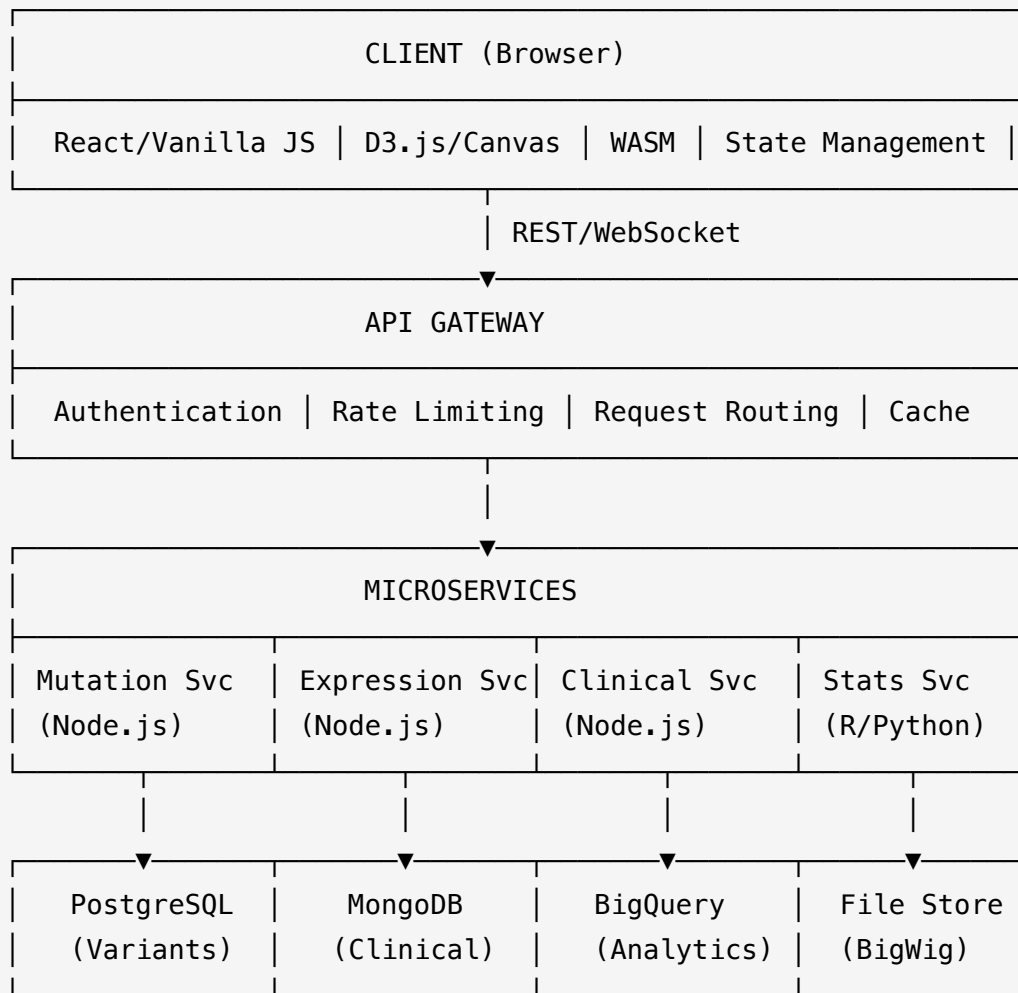
```
// Example: Reading pipeline outputs
const pipelineOutputs = {
  nf_core_sarek: {
    variants: 'results/variant_calling/**/*.vcf.gz',
    qc: 'results/multiqc/multiqc_report.html',
    bam: 'results/preprocessing/*/mapped/*.bam',
  },
  gatk_best_practices: {
    variants: '*.g.vcf.gz',
    metrics: '*.metrics.txt',
  },
};
```

---

## Software Architecture

### Q46: How would you architect a genomic visualization platform?

Answer:



### Key principles:

1. **Separation of concerns**: Data, logic, presentation
2. **Scalability**: Horizontal scaling for high traffic
3. **Caching**: Reduce database load
4. **Type safety**: TypeScript throughout

## Q47: How does ProteinPaint handle component communication?

### Answer:

ProteinPaint uses an event-based architecture (similar to RxJS):

```

// Core pattern: rx (reactive) message bus
class Block {
  constructor() {
    this.rx = new MessageBus();
  }

  // Components emit events
  onRegionChange(region) {
    this.rx.emit('region.change', region);
  }

  // Other components subscribe
  addTrack(track) {
    this.rx.on('region.change', (region) => {
      track.setRegion(region);
    });
  }
}

// Usage
block.rx.on('mutation.click', (mutation) => {
  tooltip.show(mutation);
  highlightInMatrix(mutation.sample);
});

```

### Benefits:

- Loose coupling between components
- Easy to add new visualizations
- Testable in isolation
- Matches scientific workflow (select → filter → analyze)

---

## Q48: How do you handle state in a multi-view visualization?

Answer:



Three-layer state architecture:

```
// 1. Application state (global)
const appState = {
  genome: 'hg38',
  gene: 'TP53',
  samples: selectedSamples,
  filters: activeFilters,
};

// 2. View state (per visualization)
const viewState = {
  zoomLevel: 1,
  highlightedPoints: [],
  sortOrder: 'position',
};

// 3. Derived state (computed)
const derivedState = {
  get visibleMutations() {
    return mutations.filter(
      (m) => appState.filters.every((f) => f(m)) && isInViewport(m, viewState.zoomLevel)
    );
  },
};
```

**Synchronization patterns:**

1. **Source of truth:** Single store (Redux-like)
2. **Derived views:** Computed from source
3. **Local overrides:** View-specific state
4. **URL sync:** Share state via URL params

---

**Q49: How would you implement linked visualizations like in the MB-meta portal?**

**Answer:**

The MB-meta portal's linked views (t-SNE ↔ survival curves ↔ sample matrix) demonstrate a powerful pattern for scientific exploration.

### **Core Architecture:**

```

// Linked visualization coordinator
class LinkedViewManager {
  constructor() {
    this.views = new Map(); // viewId → viewInstance
    this.sharedState = {
      selectedSamples: new Set(),
      hoveredSample: null,
      filters: [],
      customVariables: {},
    };
    this.eventBus = new EventEmitter();
  }

  registerView(viewId, view) {
    this.views.set(viewId, view);

    // View subscribes to relevant state changes
    this.eventBus.on('selection.change', (samples) => {
      view.highlightSamples(samples);
    });

    // View emits its own interactions
    view.on('sample.click', (sample) => {
      this.updateSelection(sample);
    });
  }

  updateSelection(sample, mode = 'toggle') {
    if (mode === 'toggle') {
      this.sharedState.selectedSamples.has(sample.id)
        ? this.sharedState.selectedSamples.delete(sample.id)
        : this.sharedState.selectedSamples.add(sample.id);
    }

    // Propagate to all views
    this.eventBus.emit('selection.change', Array.from(this.sharedState.selectedSamples));
  }
}

```

## Lasso Selection (t-SNE → Subcohort):

```
// Lasso selection on dimensionality reduction plot
function setupLassoSelection(svg, points, onSelect) {
  const lasso = d3
    .lasso()
    .closePathSelect(true)
    .closePathDistance(100)
    .items(points)
    .targetArea(svg)
    .on('end', function () {
      const selected = lasso.selectedItems().data();
      onSelect(selected);
    });

  svg.call(lasso);
}

// Usage: User draws on t-SNE, survival curve updates
lassoSelection.on('select', (samples) => {
  // Update survival curve with selected subcohort
  survivalCurve.setData(calculateKaplanMeier(samples));

  // Update sample matrix
  sampleMatrix.highlightRows(samples.map((s) => s.id));

  // Enable Cox regression on subcohort
  coxRegression.setEligibleSamples(samples);
});
```

## Custom Variable Creation (Key MB-meta Feature):

```

// Allow users to define custom analysis variables
class CustomVariableBuilder {
  constructor(clinicalData) {
    this.data = clinicalData;
    this.customVars = {};
  }

  defineVariable(name, definition) {
    // Example: User creates "iso17" variable
    // definition = { column: 'chromosome_17_status',
    //               transform: (v) => v === 'isochromosome' }

    this.customVars[name] = this.data.map((row) => ({
      sampleId: row.sample_id,
      value: definition.transform(row[definition.column]),
    }));

    // Now available for Cox regression
    return this.customVars[name];
  }

  getVariableForAnalysis(name) {
    return this.customVars[name];
  }
}

// User workflow:
// 1. Select column from dropdown (chromosome_17_status)
// 2. Define transformation (isochromosome vs not)
// 3. Name it "iso17"
// 4. Use in Cox regression as covariate

```

### Why This Pattern Matters:

Benefit	Implementation
Scientific Discovery	Filter → visualize → hypothesize → test
Reproducibility	URL encodes full state (filters, selections)

Benefit	Implementation
Flexibility	Users define variables without code
Validation	Visual QC catches data/caller errors

### Real Example from MB-meta Paper:

```
// The KBTBD4 workflow that discovered insertion pattern clusters
const kbtbd4Workflow = {
  step1: 'Filter to samples with KBTBD4 mutations',
  step2: 'Overlay on methylome t-SNE',
  step3: 'Visual inspection reveals two clusters',
  step4: 'Lasso-select each cluster',
  step5: 'Compare insertion patterns (R313insPRR vs R312_R313insP)',
  step6: 'Export for publication figure',
};
```

---

## Full-Stack Development

**Q50: Describe your experience with JavaScript/TypeScript.**

**Answer:**

```

// Modern JavaScript/TypeScript patterns I use:

// 1. TypeScript for type safety
interface Mutation {
  id: string;
  gene: string;
  position: number;
  consequence: MutationType;
  samples: Sample[];
}

// 2. Async/await for data fetching
async function loadMutations(gene: string): Promise<Mutation[]> {
  const response = await fetch(`/api/mutations/${gene}`);
  const data = await response.json();
  return MutationSchema.parse(data); // Zod validation
}

// 3. Functional programming
const hotspotMutations = mutations
  .filter((m) => m.count >= 10)
  .sort((a, b) => b.count - a.count)
  .slice(0, 10);

// 4. ES6+ features
const { gene, position, ...rest } = mutation;
const combined = [...array1, ...array2];
const formatted = `${gene}:${position}`;

```

### Framework experience:

- **React:** Component-based UIs
  - **D3.js:** Data visualization
  - **Node.js:** Server-side JavaScript
  - **TypeScript:** Type-safe development
-

**Q51: How do you write testable visualization code?**

**Answer:**



```

// Separate concerns for testability

// 1. Pure data transformations (easy to test)
export function calculateScales(data, dimensions) {
  return {
    x: d3
      .scaleLinear()
      .domain([0, d3.max(data, (d) => d.position)])
      .range([0, dimensions.width]),
    y: d3
      .scaleLinear()
      .domain([0, d3.max(data, (d) => d.count)])
      .range([dimensions.height, 0]),
  };
}

// Test
test('scales should map data to dimensions', () => {
  const data = [{ position: 100, count: 10 }];
  const scales = calculateScales(data, { width: 500, height: 300 });
  expect(scales.x(100)).toBe(500);
  expect(scales.y(10)).toBe(0);
});

// 2. DOM rendering (test structure, not pixels)
export function renderLollipops(container, data, scales) {
  return d3
    .select(container)
    .selectAll('circle')
    .data(data)
    .join('circle')
    .attr('cx', (d) => scales.x(d.position));
}

// Test
test('should render correct number of circles', () => {
  const container = document.createElement('div');
  renderLollipops(container, mockData, mockScales);
  expect(container.querySelectorAll('circle').length).toBe(mockData.length);
});

```

```
});
```

---

**Q52: How do you handle errors in a production application?**

**Answer:**

```

// Layered error handling

// 1. API layer – validation errors
app.post('/api/mutations', async (req, res, next) => {
  try {
    const validated = MutationSchema.parse(req.body);
    const result = await mutationService.create(validated);
    res.json(result);
  } catch (error) {
    if (error instanceof z.ZodError) {
      return res.status(400).json({
        error: 'Validation failed',
        details: error.issues,
      });
    }
    next(error); // Pass to error middleware
  }
});

// 2. Global error middleware
app.use((error, req, res, next) => {
  logger.error(error, { url: req.url, user: req.user?.id });

  res.status(error.status || 500).json({
    error: process.env.NODE_ENV === 'production' ? 'Internal server error' : error.message,
  });
});

// 3. Client-side error boundary
class ErrorBoundary extends React.Component {
  componentDidCatch(error, info) {
    trackError(error, info);
    this.setState({ hasError: true });
  }

  render() {
    if (this.state.hasError) {
      return <ErrorFallback onRetry={() => this.setState({ hasError: false })} />;
    }
  }
}

```

```
    return this.props.children;  
  }  
}
```

---

## **Q53: Explain your experience with databases and SQL?**

**Answer:**

```

-- Genomic data schema design

-- Efficient variant storage
CREATE TABLE variants (
  id SERIAL PRIMARY KEY,
  chromosome VARCHAR(10) NOT NULL,
  position INTEGER NOT NULL,
  ref_allele TEXT NOT NULL,
  alt_allele TEXT NOT NULL,
  gene_symbol VARCHAR(50),
  consequence VARCHAR(50),
  vaf DECIMAL(5,4),
  sample_id INTEGER REFERENCES samples(id),

  -- Indexes for common queries
  INDEX idx_gene (gene_symbol),
  INDEX idx_position (chromosome, position),
  INDEX idx_sample (sample_id)
);

-- Efficient range queries
SELECT v.*, s.cancer_type
FROM variants v
JOIN samples s ON v.sample_id = s.id
WHERE v.chromosome = 'chr17'
      AND v.position BETWEEN 7668421 AND 7687490 -- TP53 region
      AND v.consequence IN ('missense', 'nonsense')
ORDER BY v.position;

-- Aggregation for lollipop plot
SELECT
  gene_symbol,
  position,
  ref_allele || position || alt_allele AS aa_change,
  COUNT(*) AS count,
  COUNT(DISTINCT sample_id) AS sample_count
FROM variants
WHERE gene_symbol = 'TP53'
GROUP BY gene_symbol, position, aa_change

```

ORDER BY position;

---

## Performance & Optimization

**Q54: How do you optimize rendering for 100,000+ data points?**

**Answer:**

```

// Multi-strategy approach

// 1. Canvas instead of SVG
function renderWithCanvas(ctx, points) {
  ctx.clearRect(0, 0, width, height);

  // Batch by color to minimize state changes
  const byColor = d3.group(points, (d) => d.color);

  byColor.forEach((group, color) => {
    ctx.fillStyle = color;
    ctx.beginPath();

    group.forEach((point) => {
      ctx.moveTo(point.x + radius, point.y);
      ctx.arc(point.x, point.y, radius, 0, Math.PI * 2);
    });

    ctx.fill();
  });
}

// 2. Spatial indexing for interactions
const quadtree = d3
  .quadtree()
  .x((d) => d.x)
  .y((d) => d.y)
  .addAll(points);

function findNearest(mouseX, mouseY, radius = 10) {
  return quadtree.find(mouseX, mouseY, radius);
}

// 3. Level of detail
function render(zoomLevel) {
  if (zoomLevel < 1) {
    // Zoomed out: Show density heatmap
    renderDensity(points);
  } else if (zoomLevel < 5) {

```

```
// Medium: Show aggregated points
renderAggregated(aggregate(points, zoomLevel));
} else {
  // Zoomed in: Show individual points
  renderPoints(visiblePoints);
}
}

// 4. Web Workers for heavy computation
const worker = new Worker('cluster-worker.js');
worker.postMessage({ points, k: 10 });
worker.onmessage = (e) => renderClusters(e.data);
```

---

## Q55: When would you use WebAssembly?

**Answer:**

**Use WASM for:**

1. **Statistical calculations:** Fisher's exact test, chi-square
2. **Matrix operations:** Distance calculations, clustering
3. **File parsing:** VCF, BAM parsing in browser
4. **Compression:** Data decompression

**Example: Rust to WASM**



```

use wasm_bindgen::prelude::*;

#[wasm_bindgen]
pub fn fisher_exact(a: u32, b: u32, c: u32, d: u32) -> f64 {
    // Hypergeometric calculation
    // 30-50x faster than JavaScript
}

#[wasm_bindgen]
pub fn euclidean_distance_matrix(data: &[f64], n: usize) -> Vec<f64> {
    // O(n²) but parallelized
    // Used for clustering heatmaps
}

```

### JavaScript integration:

```

import init, { fisher_exact } from './stats.wasm';

await init();
const pValue = fisher_exact(10, 5, 3, 20);

```

---

## Q56: How do you profile and debug performance issues?

Answer:

```
// 1. Browser DevTools profiling
console.time('render');
renderVisualization(data);
console.timeEnd('render');

// 2. Performance API
const start = performance.now();
// ... expensive operation
const duration = performance.now() - start;

// 3. Custom performance marks
performance.mark('data-load-start');
await loadData();
performance.mark('data-load-end');
performance.measure('data-load', 'data-load-start', 'data-load-end');

// 4. Memory profiling
console.log('Memory:', performance.memory.usedJSHeapSize / 1024 / 1024, 'MB');

// 5. Flame graph analysis
// Use Chrome DevTools → Performance → Record

// Common issues and solutions:
// - Too many DOM nodes → Use Canvas
// - Memory leaks → Check event listener cleanup
// - Layout thrashing → Batch DOM reads/writes
// - Slow reflows → Use transform instead of top/left
```



## AI & LLM Integration

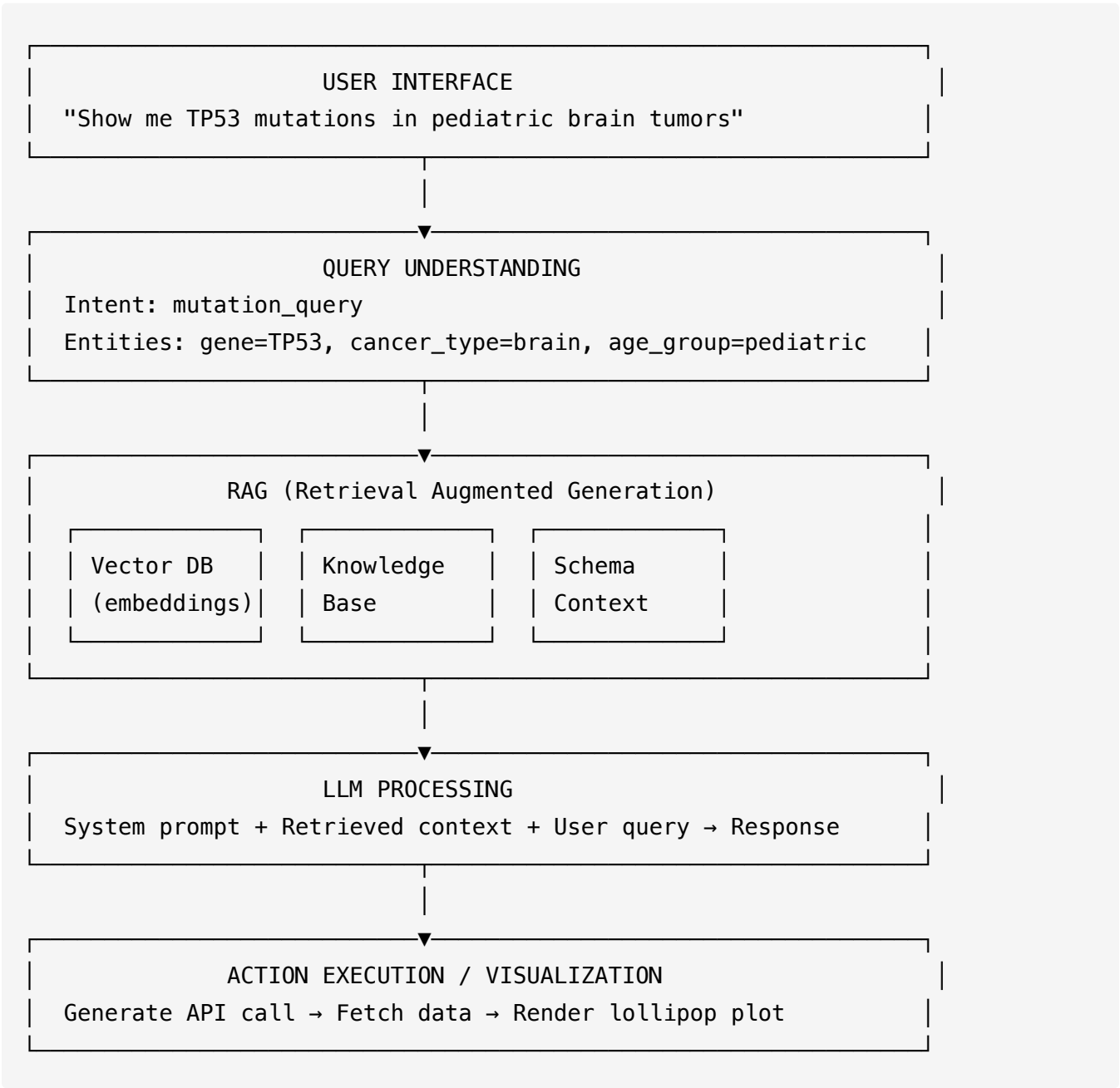
### Q57: How would you build an AI chatbot for genomic data queries?

**Answer:**

ProteinPaint is adding AI chatbot capabilities for natural language queries. Here's how to

architect such a system:

Architecture Overview:



Key Components:

```

// 1. Intent Classification
const intents = {
  mutation_query: {
    patterns: ['show mutations', 'what mutations', 'find variants'],
    action: 'queryMutations',
    requiredEntities: ['gene OR cancer_type'],
  },
  survival_analysis: {
    patterns: ['survival', 'prognosis', 'outcome'],
    action: 'runKaplanMeier',
    requiredEntities: ['gene', 'cohort'],
  },
  expression_query: {
    patterns: ['expression', 'expressed', 'RNA levels'],
    action: 'queryExpression',
    requiredEntities: ['gene'],
  },
  visualization_request: {
    patterns: ['show', 'display', 'visualize', 'plot'],
    action: 'generateVisualization',
    requiredEntities: ['data_type'],
  },
};

// 2. Entity Extraction
const entities = {
  genes: ['TP53', 'KRAS', 'BRCA1', ...geneList],
  cancer_types: ['AML', 'ALL', 'medulloblastoma', ...cancerTypes],
  visualization_types: ['lollipop', 'heatmap', 'survival', 'scatter'],
  cohorts: ['PCGP', 'TARGET', 'TCGA'],
};

// 3. Query to API Translation
function translateToAPI(intent, entities) {
  switch (intent) {
    case 'mutation_query':
      return {
        endpoint: '/api/mutations',
        params: {

```

```
        gene: entities.gene,  
        cancer_type: entities.cancer_type,  
        dataset: entities.cohort || 'all',  
    },  
};  
case 'survival_analysis':  
    return {  
        endpoint: '/api/survival',  
        params: {  
            gene: entities.gene,  
            stratify_by: 'mutation_status',  
        },  
    };  
}  
}
```

## RAG Implementation for Genomics:

```

// Retrieval Augmented Generation
class GenomicRAG {
  constructor(vectorDB, knowledgeBase) {
    this.vectorDB = vectorDB; // Pinecone, Weaviate, or pgvector
    this.knowledgeBase = knowledgeBase;
  }

  async retrieveContext(query) {
    // 1. Embed the query
    const queryEmbedding = await this.embed(query);

    // 2. Retrieve relevant documents
    const relevantDocs = await this.vectorDB.query({
      vector: queryEmbedding,
      topK: 5,
      filter: { type: 'genomic_knowledge' },
    });

    // 3. Add schema context
    const schemaContext = this.getRelevantSchema(query);

    return {
      documents: relevantDocs,
      schema: schemaContext,
      examples: this.getFewShotExamples(query),
    };
  }

  getRelevantSchema(query) {
    // Return relevant database schema for query generation
    return `
      Available tables:
      - mutations (gene, position, amino_acid_change, sample_id, cancer_type)
      - samples (sample_id, patient_id, age, diagnosis, survival_days)
      - expression (gene, sample_id, tpm, log2fc)
    `;
  }
}

```

---

# Production Guardrails & Evaluation (Shows Maturity)

A scientific AI chatbot isn't just about generating responses—it's about **trust, safety, and auditability**:

Guardrail	Implementation	Why It Matters
PHI/Controlled-Data Boundaries	Pre-filter queries; never retrieve dbGaP-controlled data without auth check	Compliance with data use agreements
Schema-Constrained Tool Calling	NL → structured JSON → validated API call (not free-form SQL)	Prevents injection, ensures reproducibility
Caching + Rate Limiting	Cache embeddings + common queries; rate limit per user/session	Cost control, DoS prevention
Audit Logs	Log every query, retrieved context, response, user ID, timestamp	Debugging, compliance, reproducibility
Hallucination Checks	Verify claims against knowledge base before returning	Scientific accuracy is non-negotiable
Fallback Behavior	"I don't have enough information" > confident wrong answer	Better to admit uncertainty

```

// Production-ready guardrails
const chatbotGuardrails = {
  // 1. PHI protection
  dataAccessControl: async (query, user) => {
    const requestedDatasets = extractDatasets(query);
    const userPermissions = await getUserPermissions(user);

    const unauthorized = requestedDatasets.filter((ds) => !userPermissions.includes(ds));

    if (unauthorized.length > 0) {
      return {
        blocked: true,
        reason: `You don't have access to: ${unauthorized.join(', ')}`,
      };
    }
    return { blocked: false };
  },

  // 2. Schema-constrained output (not free-form)
  structuredOutput: {
    type: 'function_calling',
    schema: {
      query_mutations: { gene: 'string', filters: 'object' },
      run_survival: { gene: 'string', stratify_by: 'string' },
      // ... other validated actions
    },
  },

  // 3. Audit logging
  auditLog: (query, response, context) => ({
    timestamp: new Date().toISOString(),
    user_id: context.user.id,
    query_text: query,
    retrieved_docs: context.retrievedDocs.map((d) => d.id),
    response_text: response.text,
    model_version: context.model,
    latency_ms: context.latency,
  }),
};

```



```
// 4. Evaluation framework (gold questions)
evaluationSuite: {
  gold_questions: [
    { q: 'What is the most common TP53 mutation?', expected: 'R175H' },
    { q: 'Is TP53 on chromosome 17?', expected: 'yes' },
    { q: 'What is the moon made of?', expected: 'out_of_scope' },
  ],
  metrics: ['accuracy', 'hallucination_rate', 'retrieval_precision'],
},

// 5. Prompt injection defense
inputSanitization: (userQuery) => {
  const dangerousPatterns = [/ignore previous instructions/i, /you are now/i, /system p
  return dangerousPatterns.some((p) => p.test(userQuery))
    ? { blocked: true, reason: 'Invalid query format' }
    : { blocked: false, sanitized: userQuery };
},
};
```

---

## Q58: What are the challenges of using LLMs in scientific applications?

Answer:

### 1. Hallucination Prevention:

```

// Problem: LLMs can generate plausible but incorrect scientific claims
const hallucinations = {
  example: 'TP53 is located on chromosome 11', // Wrong! It's chr17
  risk: 'Users may trust incorrect information for clinical decisions',
};

// Solutions:
const antiHallucinationStrategies = {
  // 1. Ground responses in retrieved data
  ragGrounding: {
    approach: 'Only answer from retrieved documents',
    prompt: `Answer ONLY based on the following context.
      If the answer is not in the context, say "I don't have that information."
      Context: {retrieved_docs}`,
  },

  // 2. Fact verification layer
  factChecker: async (response, query) => {
    const claims = extractClaims(response);
    for (const claim of claims) {
      const verified = await verifyAgainstKnowledgeBase(claim);
      if (!verified) {
        return flagUnverifiedClaim(claim);
      }
    }
    return response;
  },

  // 3. Cite sources
  citationRequired: {
    prompt: 'Cite the specific source for each claim. Format: [Source: dataset/paper]',
  },

  // 4. Confidence scoring
  confidenceThreshold: {
    approach: 'Only show high-confidence responses',
    threshold: 0.85,
  },
};

```

## 2. Domain-Specific Accuracy:

```
// Genomic-specific challenges
const domainChallenges = {
  nomenclature: {
    problem: 'Gene names have aliases (TP53 = p53 = TRP53)',
    solution: 'Normalize to HGNC official symbols',
  },

  coordinates: {
    problem: 'hg19 vs hg38 confusion',
    solution: 'Always specify and validate genome build',
  },

  statistics: {
    problem: 'LLMs struggle with precise statistical interpretations',
    solution: 'Use structured outputs, not free-form statistical claims',
  },

  recency: {
    problem: 'Training data cutoff misses recent discoveries',
    solution: 'RAG with updated knowledge base',
  },
};
```

## 3. Structured Output for Visualization:

```
// LLMs should output structured data, not just text
const structuredOutputSchema = {
  visualization_request: {
    type: 'object',
    properties: {
      plot_type: { enum: ['lollipop', 'heatmap', 'survival', 'scatter'] },
      data_query: {
        gene: 'string',
        cancer_type: 'string',
        filters: 'object',
      },
      customization: {
        title: 'string',
        color_by: 'string',
      },
    },
  },
};

// Use function calling for reliable structured output
const functionCall = {
  name: 'create_visualization',
  parameters: structuredOutputSchema.visualization_request,
};
```

#### 4. Privacy & Data Security:

```
const securityConsiderations = {  
  // Don't send PHI to external LLM APIs  
  phi_protection: {  
    approach: 'Pre-process queries to remove identifiers',  
    alternative: 'Use local/on-premise LLMs for sensitive data',  
  },  
  
  // Audit trail  
  logging: {  
    what: 'Log all queries and responses',  
    why: 'Reproducibility, debugging, compliance',  
  },  
  
  // Access control  
  permissions: {  
    approach: 'LLM respects user's data access permissions',  
    implementation: 'Include user role in system prompt',  
  },  
};
```

---

## Q59: How would you integrate natural language queries with ProteinPaint visualizations?

**Answer:**

**End-to-End Flow:**

```

// Natural Language → Visualization Pipeline
class NLVisualizationEngine {
  constructor(llm, proteinPaint) {
    this.llm = llm;
    this.pp = proteinPaint;
  }

  async processQuery(userQuery) {
    // Step 1: Parse intent and entities
    const parsed = await this.llm.parse({
      query: userQuery,
      schema: this.getVisualizationSchema(),
    });

    // Step 2: Validate and enrich
    const enriched = await this.enrichQuery(parsed);

    // Step 3: Fetch data
    const data = await this.fetchData(enriched);

    // Step 4: Generate visualization config
    const vizConfig = this.generateVizConfig(enriched, data);

    // Step 5: Render in ProteinPaint
    return this.pp.render(vizConfig);
  }

  getVisualizationSchema() {
    return {
      intents: [
        {
          name: 'show_mutations',
          examples: [
            'Show TP53 mutations',
            'What mutations are in KRAS?',
            'Display BRCA1 variants in breast cancer',
          ],
          parameters: ['gene', 'cancer_type?', 'mutation_type?'],
          visualization: 'lollipop',
        }
      ]
    };
  }
}

```

```

    },
    {
      name: 'compare_expression',
      examples: [
        'Compare MYC expression in AML vs ALL',
        'How is TP53 expressed across cancer types?',
      ],
      parameters: ['gene', 'group_by'],
      visualization: 'boxplot',
    },
    {
      name: 'survival_analysis',
      examples: [
        'Does TP53 mutation affect survival in medulloblastoma?',
        'Show Kaplan-Meier for MYCN amplification',
      ],
      parameters: ['gene', 'cancer_type', 'stratify_by'],
      visualization: 'kaplan_meier',
    },
  ],
};
}

```

```

generateVizConfig(parsed, data) {
  // Translate parsed query to ProteinPaint config
  switch (parsed.visualization) {
    case 'lollipop':
      return {
        component: 'lollipop',
        gene: parsed.gene,
        data: data.mutations,
        options: {
          colorBy: parsed.mutation_type || 'consequence',
          showDomains: true,
        },
      };

    case 'kaplan_meier':
      return {
        component: 'survival',

```

```
    data: data.survival,  
    groups: parsed.stratify_by,  
    options: {  
      showConfidenceInterval: true,  
      showPValue: true,  
    },  
  };  
}  
}  
}
```

### Example Interactions:



```

// User: "Show me all TP53 mutations in pediatric brain tumors"
const response1 = {
  interpretation: 'Querying TP53 mutations filtered to pediatric brain tumor samples',
  visualization: {
    type: 'lollipop',
    gene: 'TP53',
    filters: { cancer_type: 'brain', age_group: 'pediatric' },
  },
  data_summary: 'Found 47 mutations across 312 samples',
  followup_suggestions: [
    'Compare to adult brain tumors?',
    'Show survival by mutation status?',
    'Which domains are most affected?',
  ],
};

// User: "Does MYCN amplification affect survival?"
const response2 = {
  interpretation: 'Running Kaplan-Meier analysis stratified by MYCN amplification status',
  visualization: {
    type: 'kaplan_meier',
    gene: 'MYCN',
    stratify_by: 'amplification_status',
  },
  statistical_result: {
    log_rank_p: 0.0001,
    hazard_ratio: 2.3,
    interpretation: 'MYCN amplification significantly associated with worse survival',
  },
};

```

### Conversational Context:

```
// Maintain conversation history for follow-up queries
class ConversationManager {
  constructor() {
    this.history = [];
    this.context = {};
  }

  async processFollowup(query) {
    // Resolve references like "those mutations", "that gene"
    const resolved = this.resolveReferences(query);

    // Add to history
    this.history.push({ role: 'user', content: query });

    // Process with context
    const response = await this.engine.processQuery(resolved, this.context);

    // Update context
    this.context = {
      lastGene: response.gene,
      lastCancerType: response.cancer_type,
      lastVisualization: response.visualization,
    };

    return response;
  }

  resolveReferences(query) {
    // "Show those in AML" → "Show TP53 mutations in AML"
    return query
      .replace(/those mutations?/gi, `${this.context.lastGene} mutations`)
      .replace(/that gene/gi, this.context.lastGene);
  }
}
```

---

## Q60: What prompt engineering techniques are important for scientific applications?

Answer:

### 1. System Prompts for Scientific Accuracy:

```
const scientificSystemPrompt = `
You are a genomics research assistant for ProteinPaint, a cancer genomics visualization platform.

CRITICAL RULES:
1. ONLY provide information that can be verified from the provided data or established genomic knowledge.
2. NEVER make up statistics, p-values, or mutation frequencies.
3. ALWAYS specify the genome build (hg19/hg38) when discussing coordinates.
4. Use HGNC official gene symbols (not aliases).
5. Cite sources for any claims about gene function or clinical significance.
6. If uncertain, say "I don't have enough information to answer that."

WHEN GENERATING VISUALIZATIONS:
1. Confirm the user's intent before generating.
2. Explain what the visualization will show.
3. Highlight any data limitations (sample size, missing data).

YOUR KNOWLEDGE INCLUDES:
- ProteinPaint API endpoints and parameters
- Genomic file formats (VCF, MAF, BED)
- Cancer biology terminology
- Statistical methods (Kaplan-Meier, Fisher's exact)
`;
```

### 2. Few-Shot Examples:

```

const fewShotExamples = [
  {
    user: 'Show TP53 mutations',
    assistant: `I'll create a lollipop plot showing TP53 mutations.

**Visualization:** Lollipop plot of TP53 (chr17:7,571,720–7,590,868, hg38)
**Data source:** PCGP + TARGET pediatric cancer cohorts
**Sample size:** 2,847 samples with TP53 sequencing

The plot will show:
- X-axis: Amino acid position (1–393)
- Y-axis: Number of mutations
- Colors: Mutation consequence (missense, nonsense, frameshift)
- Protein domains: Transactivation, DNA-binding, Tetramerization

[Generating visualization...]\`,
  },
  {
    user: "What's the most common mutation?",
    assistant: `Based on the TP53 lollipop plot just generated:

**Most common mutations:**
1. R175H (n=23, 15%) – DNA-binding domain
2. R248Q (n=18, 12%) – DNA-binding domain
3. R273H (n=15, 10%) – DNA-binding domain

**Clinical note:** These are known hotspot mutations that disrupt DNA binding. They're cl

Would you like to:
- See survival analysis for R175H carriers?
- Compare mutation patterns across cancer types?`,
  },
];

```

### 3. Structured Output Prompts:

```
const structuredPrompt = `
Analyze the user's query and respond with a JSON object:

{
  "intent": "mutation_query | expression_query | survival_analysis | general_question",
  "entities": {
    "gene": "HGNC symbol or null",
    "cancer_type": "cancer type or null",
    "visualization": "plot type or null"
  },
  "api_call": {
    "endpoint": "/api/...",
    "parameters": {}
  },
  "explanation": "What will be shown to the user",
  "caveats": ["Any data limitations or warnings"]
}

User query: {query}
`;
```

#### 4. Chain-of-Thought for Complex Queries:

```
const chainOfThoughtPrompt = `
```

```
For complex genomic queries, think step-by-step:
```

1. UNDERSTAND: What is the user asking for?
2. IDENTIFY: What genes, cancer types, or data types are involved?
3. VALIDATE: Is this query possible with available data?
4. PLAN: What API calls or computations are needed?
5. EXECUTE: Generate the appropriate response/visualization.
6. EXPLAIN: Describe what was done and any limitations.

Example:

User: "Is TP53 mutation associated with worse survival in pediatric leukemia?"

1. UNDERSTAND: User wants survival analysis stratified by TP53 mutation status
  2. IDENTIFY: Gene=TP53, Cancer=leukemia (includes ALL, AML), Analysis=Kaplan-Meier
  3. VALIDATE: We have survival data and TP53 mutation calls for pediatric leukemia
  4. PLAN:
    - Query TP53 mutations in ALL + AML samples
    - Get survival data for those samples
    - Stratify by mutation status
    - Run log-rank test
  5. EXECUTE: [Generate Kaplan-Meier plot with statistics]
  6. EXPLAIN: "Showing survival curves for 847 pediatric leukemia patients..."
- ```
`;
```

## 5. Error Handling Prompts:

```

const errorHandlingExamples = {
  gene_not_found: {
    user: 'Show mutations in BRCA3',
    assistant: `I couldn't find a gene called "BRCA3". Did you mean:
- **BRCA1** (breast cancer gene 1, chr17)
- **BRCA2** (breast cancer gene 2, chr13)
- **BRCA1P1** (BRCA1 pseudogene)

Please clarify which gene you'd like to explore.`
  },

  ambiguous_query: {
    user: 'Show the data',
    assistant: `I'd be happy to help! Could you be more specific about what you'd like to s

For example:
- "Show TP53 mutations" → Lollipop plot
- "Show MYC expression across cancers" → Box plot
- "Show survival for MYCN amplification" → Kaplan-Meier curve

What would you like to explore?`,
  },

  insufficient_data: {
    user: 'Show survival for TP53 mutations in retinoblastoma',
    assistant: `⚠️ **Data limitation:** We have only 12 retinoblastoma samples with TP53 m

This small sample size means:
- Statistical power is limited
- Results may not be reliable

Would you like to:
1. Proceed anyway (with caveat noted)?
2. Expand to all pediatric solid tumors?
3. Look at a different gene with more data?`,
  },
};

```

---



# Scientific Communication

## Q61: Describe a time you explained technical concepts to non-technical stakeholders.

### Answer framework:

"When presenting our mutation visualization tool to clinical researchers..."

1. **Context:** Who was the audience? What was the goal?
2. **Challenge:** What technical concepts needed explanation?
3. **Approach:** How did you simplify without losing accuracy?
4. **Outcome:** Did they understand? What decisions resulted?

### Example:

"I presented our VAF filtering feature to oncologists. Instead of explaining the statistical thresholds, I showed them:

- A scatter plot of VAF values with labeled regions
- Real examples of artifacts vs true variants
- Interactive sliders so they could see filtering effects

They immediately understood and suggested clinically relevant thresholds we hadn't considered."

---

## Q62: How would you write about a visualization tool for publication?

### Answer:

Structure for methods/software papers:

1. **Introduction**
  - Problem statement
  - Existing solutions and gaps
  - Our contribution
2. **Methods/Implementation**



- Architecture overview
- Key algorithms
- Performance characteristics
- Data sources

### 3. **Results**

- Use cases with real data
- Performance benchmarks
- User feedback

### 4. **Discussion**

- Comparison to alternatives
- Limitations
- Future directions

#### **ProteinPaint-style example:**

"We developed ProteinPaint, an interactive web application for visualizing somatic mutations in the context of protein structure. The tool renders mutation lollipop plots with protein domain annotations, enabling researchers to identify mutation hotspots and assess functional impact..."

---

## **Team & Collaboration**

### **Q63: How do you approach code review?**

**Answer:**

### My code review checklist:

- ❑ Correctness
  - Does the code do what it claims?
  - Are edge cases handled?
- ❑ Readability
  - Are variable names descriptive?
  - Is the logic clear?
- ❑ Performance
  - Any obvious inefficiencies?
  - Memory leaks?
- ❑ Testing
  - Are there tests?
  - Do they cover edge cases?
- ❑ Security
  - Input validation?
  - SQL injection risks?
- ❑ Documentation
  - Are complex parts commented?
  - Is the PR description clear?

### Giving feedback:

- Be specific and constructive
  - Suggest alternatives, don't just criticize
  - Distinguish "must fix" from "nice to have"
  - Praise good patterns
-

## Q64: How do you handle disagreements about technical approaches?

**Answer:**

Framework: **Data-driven decision making**

1. **Listen:** Understand the other perspective fully
2. **Articulate:** State your position clearly with reasoning
3. **Experiment:** If possible, prototype both approaches
4. **Measure:** Use benchmarks, user feedback, or code metrics
5. **Decide:** Based on evidence, not opinions
6. **Document:** Record the decision and rationale

**Example:**

"We disagreed about SVG vs Canvas for a scatter plot. Instead of debating, we:

- Prototyped both with our actual data size
  - Measured render time and interaction responsiveness
  - Found Canvas was 10x faster for our 50k points
  - Documented the decision in our architecture docs"
- 

## Behavioral Questions

### Q65: Why are you interested in this position?

**Answer framework:**

1. **Mission alignment:** "Pediatric cancer research is meaningful to me because..."
  2. **Technical fit:** "The tech stack (D3, Rust, Node.js) matches my experience in..."
  3. **Growth opportunity:** "I'm excited to learn about... and contribute to..."
  4. **Team culture:** "Open source development and scientific rigor appeal to me..."
-

# Your Personalized Answer (The Real Story)

## The Core Narrative:

"I've spent my research career **generating** the kinds of data that ProteinPaint visualizes—single-cell methylation, ATAC-seq, gene regulatory networks. I understand what scientists need to see because I've been that scientist, staring at a dataset and wishing I could interactively explore it.

What excites me about this role is the opportunity to flip from **data producer to tool builder**. When I built OmicsOracle, I experienced how powerful it is when you combine genomics expertise with modern software engineering—the AI chatbot could answer questions that would take hours to answer manually. ProteinPaint is doing the same thing at scale, across multiple cancer portals, and I want to be part of that."

## Why Your Background Creates Unique Value:

| Your Experience                       | How It Maps to ProteinPaint                                 |
|---------------------------------------|-------------------------------------------------------------|
| Single-cell methylome + deep learning | Epigenetic data visualization, survivorship late effects    |
| ATAC-seq + RNA-seq integration        | Multi-omics visualization, linked views                     |
| Gene regulatory networks              | Complex relationship visualization (arc diagrams, networks) |
| OmicsOracle (LLM + genomics)          | AI chatbot feature they're actively building                |
| BioPipelines (backend)                | BCF/tabix processing, data pipelines                        |
| KL-divergence / scientific rigor      | QC as a product feature, not afterthought                   |

## Survivorship Portal Connection (Great Interview Drop):

"I was particularly impressed by the Survivorship Portal's architecture. The **COHORT** →

**FILTER → GROUPS → CHARTS** workflow is elegant—it matches how I think about cohort studies from my ATAC-seq work. And the **4-filter variant QC method** shows the kind of scientific rigor I value—treating QC as a product feature, not just a preprocessing afterthought. That's the same philosophy I brought to my methylome paper, where we designed the loss function specifically for the biology of methylation data."

### Technical Stack Alignment:

```
const stackAlignment = {
  survivorshipPortal: {
    frontend: 'JavaScript + D3.js',
    backend: 'Node.js + Express',
    databases: 'SQLite (phenotypes), BCF (variants), tabix (LD)',
    pipeline: 'GATK HaplotypeCaller → joint genotyping → annotation',
  },
  yourExperience: {
    frontend: 'React (OmicsOracle), D3 (learning)',
    backend: 'Node.js, Python APIs',
    databases: 'PostgreSQL, vector DBs (ChromaDB)',
    pipeline: 'STAR, BWA, MACS2, DESeq2, custom ML pipelines',
  },
  overlap: [
    'Full-stack JavaScript/TypeScript',
    'Genomic file formats (VCF, BAM, BigWig)',
    'Variant calling and annotation',
    'Scientific rigor in software design',
  ],
};
```

---

### Publications Connection to Survivorship Science:

Your single-cell methylome work is **directly relevant** to what the Survivorship Portal tracks:

```
const survivorshipRelevance = {
  problem: 'Do childhood cancer survivors have epigenetic "scars" from treatment?',

  yourExpertise: {
    methylationPrediction: 'Can predict methylation states from sequence context',
    singleCellResolution: 'Handle sparse, noisy methylation data',
    transferLearning: 'Leverage bulk data to improve single-cell predictions',
  },

  portalApplication: {
    currentFeature: 'Survivorship Portal tracks 400M germline variants',
    futureOpportunity: 'Methylation data integration for late effects research',
    yourContribution: 'Visualization approaches for sparse epigenetic data',
  },

  interviewDrop: `
    "My methylome work is directly relevant to survivorship research.
    Childhood cancer survivors often have epigenetic 'scars' from treatment—
    changes in methylation patterns that persist for decades and may drive
    late effects. I've developed methods to handle the sparsity of single-cell
    methylation data, which is the same challenge you'd face visualizing
    survivorship epigenetic profiles."
  `,
};
```

---

### The "Why St. Jude Specifically" Answer:

"St. Jude's approach resonates with me for three reasons:

1. **WGS over WES philosophy:** I understand why you prioritize whole-genome sequencing for pediatric cancers—the drivers often hide in non-coding regions. My work on gene regulatory networks showed me how important enhancers and chromatin accessibility are.
2. **Open science commitment:** The Survivorship Portal offers summary-level exploration openly while protecting individual data. This balance between accessibility and privacy is exactly right for genomics tools.

3. **Scientific rigor in software:** The 4-filter variant QC method tells me you treat the portal as scientific software, not just a dashboard. That matches how I approach my own work—the loss function in my methylome paper wasn't just 'whatever works,' it was designed for the biology of methylation."
- 

## Q66: Tell me about a challenging project you've worked on.

### Answer framework (STAR method):

- **Situation:** What was the context?
  - **Task:** What was your responsibility?
  - **Action:** What did you do specifically?
  - **Result:** What was the outcome? Metrics if possible.
- 

## Q67: Where do you see yourself in 5 years?

### Answer considerations:

- Show commitment to the field
  - Express interest in growth (technical lead, first-author papers)
  - Align with the position (mentored postdoc → independent researcher)
- 

## ? Questions to Ask Them

### Technical Questions

1. "What's the current tech stack, and are there plans to modernize any parts?"
2. "How do you balance feature development with technical debt?"
3. "What's the deployment process like? How often do you release?"
4. "How do you handle data privacy for clinical data?"

## Team Questions

- 1. "What does a typical day/week look like for this role?"
- 2. "How does the team collaborate with wet-lab researchers?"
- 3. "What's the mentorship structure for new team members?"
- 4. "How are projects prioritized?"

## Research Questions

- 1. "What are the biggest technical challenges you're currently facing?"
- 2. "How do you decide which features to build vs. use existing tools?"
- 3. "What opportunities are there for first-author publications?"
- 4. "How does the team stay current with new visualization techniques?"

## Culture Questions

- 1. "How does the team handle remote work/flexibility?"
- 2. "What's the work-life balance like?"
- 3. "How are decisions made when there are competing priorities?"

---

## 💡 Project Portfolio & Tech Stack

*Your GitHub projects demonstrate direct experience with ProteinPaint's required technologies.*

## Project Overview

| Project        | Description                            | Key Technologies                             | ProteinPaint Alignment |
|----------------|----------------------------------------|----------------------------------------------|------------------------|
| OmicsOracle    | AI-powered biomedical data platform    | React, TypeScript, D3.js, FastAPI, LangChain | ★★★★★<br>Frontend + AI |
| llm_evaluation | LLM benchmarking framework (61 models) | Python, PyTorch, vLLM, CLI                   | ★★★★★<br>Backend + ML  |



| Project                   | Description            | Key Technologies                     | ProteinPaint Alignment   |
|---------------------------|------------------------|--------------------------------------|--------------------------|
| <b>federated-learning</b> | FL+DP research library | PyTorch, Flower, Opacus              | ★★★★ ML foundations      |
| <b>BioPipelines</b>       | AI workflow composer   | Nextflow, Python, Singularity, SLURM | ★★★★★★★★ Genomics domain |

## Q68: Tell us about OmicsOracle. How does it relate to ProteinPaint?

**Answer:**

OmicsOracle is an **AI-powered biomedical research platform** I built that shares many architectural patterns with ProteinPaint:

**Technology Overlap:**

| Technology                | OmicsOracle Implementation                                     | ProteinPaint Relevance        |
|---------------------------|----------------------------------------------------------------|-------------------------------|
| <b>React + TypeScript</b> | Modern component architecture with type-safe interfaces        | Core frontend stack           |
| <b>D3.js</b>              | Network graphs, force simulations, heatmaps, drag interactions | Primary visualization library |
| <b>Chart.js</b>           | Timeline charts, statistical dashboards                        | Complementary charting        |
| <b>FastAPI</b>            | Async backend, WebSocket real-time updates                     | Similar to Express patterns   |
| <b>LLM/RAG</b>            | LangChain, ChromaDB vector store, structured outputs           | AI chatbot integration        |

## Visualization Types I Built:

```
// D3.js Network Graph (similar to ProteinPaint's gene networks)
const simulation = d3
  .forceSimulation(data.nodes)
  .force(
    'link',
    d3.forceLink(data.links).id((d) => d.id)
  )
  .force('charge', d3.forceManyBody().strength(-300))
  .force('center', d3.forceCenter(width / 2, height / 2));

// Interactive drag behavior
node.call(d3.drag().on('start', dragstarted).on('drag', dragged).on('end', dragended));
```

## Key Features Relevant to ProteinPaint:

- **Publication network visualization** (like ProteinPaint's mutation networks)
- **Heatmaps with color scales** (similar to expression matrices)
- **Timeline charts** (analogous to survival curves)
- **Volcano plots** (differential expression visualization)
- **Real-time updates via WebSockets** (like ProteinPaint's live data)

---

## Q69: What did you learn building the LLM Evaluation Framework?

### Answer:

My llm\_evaluation project is a **production-grade benchmarking system** for 61+ large language models:

### Relevant Skills Demonstrated:

1. **Clean Architecture:**

```
# Modular design pattern (similar to ProteinPaint's component structure)
llm_evaluation/
├── configs/           # Model configurations with presets
├── models/           # Base classes + implementations
├── evaluation/       # Orchestrator, metrics, engines
└── engines/         # vLLM, Transformers backends
```

## 2. Configuration Management:

```
@dataclass
class ModelConfig:
    model_name: str
    huggingface_id: str
    context_window: int
    specialization_category: str
    quantization_method: str = "none"

    def to_vllm_args(self) -> Dict[str, Any]:
        """Convert to vLLM initialization parameters."""
        # Similar to ProteinPaint's app configuration
```

## 3. Performance Optimization:

- GPU memory management (parallels WebGL optimization)
- Batch processing for throughput
- Caching strategies (like ProteinPaint's data caching)

## 4. Testing & Validation:

```
# 105 tests with pytest fixtures
@pytest.fixture
def sample_model_config():
    return ModelConfig(model_name="test", ...)

# Similar rigor needed for visualization testing
```

## What This Shows:

- I can design **scalable, maintainable systems**
  - Experience with **complex configuration management**
  - Understanding of **performance-critical applications**
- 

## Q70: How does BioPipelines demonstrate genomics expertise?

**Answer:**

BioPipelines is an **AI-powered workflow composer** that generates Nextflow pipelines from natural language:

**Domain Knowledge Demonstrated:**

| Pipeline Type | Tools I Integrated         | ProteinPaint Data Types      |
|---------------|----------------------------|------------------------------|
| RNA-seq       | STAR, Salmon, DESeq2, GSEA | Gene expression, DE analysis |
| DNA-seq       | BWA, GATK, VEP, SnpEff     | Variant calling, annotation  |
| ChIP-seq      | Bowtie2, MACS2, HOMER      | Peak calling, TF binding     |
| ATAC-seq      | MACS2, DeepTools           | Chromatin accessibility      |
| Single-cell   | CellRanger, Seurat, Scanpy | scRNA-seq clustering         |
| Methylation   | Bismark, methylKit         | Epigenetic analysis          |

**Technical Implementation:**

```
# AI-driven tool selection (similar to AI chatbot for ProteinPaint)
class WorkflowComposer:
    def compose(self, user_request: str) -> Workflow:
        # 1. Parse user intent
        intent = self.intent_parser.parse(user_request)
        # Example: "RNA-seq differential expression for cancer samples"

        # 2. Select appropriate modules
        modules = self.tool_catalog.find_modules(intent)
        # Returns: [fastqc, star, featurecounts, deseq2]

        # 3. Generate Nextflow workflow
        return self.generator.generate(modules, intent)
```

## Container Architecture:

```
containers/images/
├─ rna-seq_1.0.0.sif    (643 tools including STAR, DESeq2)
├─ dna-seq_1.0.0.sif   (BWA, GATK, VEP)
├─ chip-seq_1.0.0.sif  (Bowtie2, MACS2)
└─ scrna-seq_1.0.0.sif (CellRanger, Seurat)
```

## Why This Matters for ProteinPaint:

- Deep understanding of **genomic data types** and their visualization needs
- Experience with **containerized bioinformatics** (reproducibility)
- Built **AI assistants for genomics** (directly relevant to chatbot work)

---

## Q71: How do your projects demonstrate the full ProteinPaint tech stack?

### Answer:

Here's a comprehensive mapping of my project experience to ProteinPaint requirements:

### Frontend Technologies:

| Skill                     | My Experience                                    | Project     |
|---------------------------|--------------------------------------------------|-------------|
| JavaScript/<br>TypeScript | React components, type-safe APIs                 | OmicsOracle |
| D3.js                     | Force graphs, heatmaps, timelines, drag/<br>zoom | OmicsOracle |
| SVG/Canvas                | Interactive visualizations                       | OmicsOracle |
| Real-time Updates         | WebSocket integration                            | OmicsOracle |

### Backend Technologies:

| Skill            | My Experience                             | Project        |
|------------------|-------------------------------------------|----------------|
| Python           | FastAPI, async processing, ML pipelines   | All projects   |
| Node.js patterns | Express-like routing (FastAPI equivalent) | OmicsOracle    |
| SQL              | Data queries, aggregations                | Ilm_evaluation |
| REST APIs        | Comprehensive API design                  | All projects   |

### Genomics & Data Science:

| Skill                | My Experience                         | Project      |
|----------------------|---------------------------------------|--------------|
| Sequencing pipelines | RNA-seq, DNA-seq, ChIP-seq, scRNA-seq | BioPipelines |
| File formats         | FASTQ, BAM, VCF, BED, BigWig          | BioPipelines |
| Statistical methods  | Differential expression, QC metrics   | BioPipelines |
| Workflow managers    | Nextflow DSL2, Snakemake              | BioPipelines |

### AI/ML Technologies:

| Skill           | My Experience                                | Project                      |
|-----------------|----------------------------------------------|------------------------------|
| LLM Integration | RAG, structured outputs, function<br>calling | OmicsOracle,<br>BioPipelines |

| Skill              | My Experience              | Project                      |
|--------------------|----------------------------|------------------------------|
| Prompt Engineering | Scientific domain prompts  | All AI projects              |
| Vector Databases   | ChromaDB, FAISS embeddings | OmicsOracle                  |
| Model Serving      | vLLM, Ollama               | llm_evaluation, BioPipelines |

DevOps & Infrastructure:

| Skill      | My Experience                          | Project            |
|------------|----------------------------------------|--------------------|
| Testing    | pytest (467+ tests), fixtures, mocking | All projects       |
| CI/CD      | GitHub Actions workflows               | federated-learning |
| Containers | Singularity/Apptainer for HPC          | BioPipelines       |
| HPC        | SLURM job scheduling                   | BioPipelines       |

Code Quality Metrics Across Projects:

|                                                                                                                                                                               |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <div><div>federated-learning:</div><div><div>└─ Type hint coverage: 91.9%</div><div>└─ Test count: 105</div><div>└─ CI/CD: GitHub Actions</div></div></div>                   |
| <div><div>llm_evaluation:</div><div><div>└─ Architecture: Clean modular design</div><div>└─ Models supported: 61+</div><div>└─ Documentation: Comprehensive</div></div></div> |
| <div><div>BioPipelines:</div><div><div>└─ Containers: 12 production images</div><div>└─ Modules: 71 Nextflow modules</div><div>└─ Tests: 467+</div></div></div>               |

---

## 🔍 Skills Alignment Check

### From Job Description → Tutorial Coverage

| Required Skill  | Tutorial Coverage           | Confidence |
|-----------------|-----------------------------|------------|
| Linux           | 2.1-2.4 (Backend tutorials) | ✅ High     |
| JavaScript      | 1.1-1.4, 3.1-3.6, 4.6-4.8   | ✅ High     |
| Node.js         | 2.1-2.4 (Backend tutorials) | ✅ High     |
| R               | 2.4 R Integration           | ✅ High     |
| Python          | 3.1 Scatter (UMAP)          | ⚠️ Medium  |
| Rust            | 4.4-4.5 Rust Parsing/WASM   | ✅ High     |
| SQL             | 2.2 PostgreSQL              | ✅ High     |
| AI              | 4.3 AI Chatbot              | ✅ High     |
| Testing         | 4.1 Testing                 | ✅ High     |
| CI/CD           | 4.2 CI/CD                   | ✅ High     |
| Cancer Genomics | All tutorials + this doc    | ✅ High     |

### Areas for Additional Study

1. **Single-cell omics:** Consider adding scanpy/Seurat tutorial
2. **Python data science:** Expand pandas/numpy coverage
3. **Cloud deployment:** AWS/GCP specifics
4. **Paper writing:** Scientific communication workshop

---

## 🔍 Your Publications - Discussion Points

*Questions they may ask based on your published research in epigenomics and computational*



biology.

## Your Papers Summary

| Paper                                       | Journal (Year)               | Your Role    | Key Methods                                  |
|---------------------------------------------|------------------------------|--------------|----------------------------------------------|
| Single-cell DNA Methylome Completion        | Frontiers in Genetics (2022) | First Author | Transfer learning, KL-divergence, DNNs, WGBS |
| Cardiac Fibroblast Chromatin Landscape      | Epigenetics (2022)           | Co-author    | ATAC-seq, RNA-seq, motif analysis, GRNs      |
| Acta2 in Cardiac Fibroblast Differentiation | J Mol Cell Cardiol (2022)    | Co-author    | Knockout mice, myofibroblast differentiation |

---

## Q72: Tell us about your single-cell methylome paper. What problem were you solving?

### Answer:

My first-author paper addressed the **extreme sparsity** in single-cell DNA methylome profiles:

### The Problem:

- Single-cell WGBS has data coverage as low as 0.06-1.47%
- Most CpG sites have no measured methylation state
- This sparsity prevents meaningful downstream analysis
- Existing methods required either high coverage or additional data modalities

### Our Solution:

Combined **transfer learning** with **KL-divergence loss** to train deep neural networks:

1. **Transfer Learning:** Pre-trained models on denser somatic tissue methylomes (5-21% coverage) then fine-tuned on sparse embryo data
2. **KL-Divergence Loss:** Unlike binary logistic loss, KL-divergence preserves the continuous nature of methylation values (0-1), utilizing more information from the data

### 3. **Network Architecture:** Hybrid CNN-RNN model:

- CNN learns DNA sequence patterns
- Bidirectional GRU captures local methylation correlation among neighboring CpGs
- Joint module fuses sequence and methylation features

### **Results:**

- 38.70% improvement when combining both techniques
- Increased coverage from 0.06-1.47% → 43.80-73.65%
- Enabled analysis of promoters, CpG islands, and genome bins
- Data deposited to NCBI GEO, code on GitHub

### **Relevance to ProteinPaint:**

- Shows experience with **sparse genomic data** - common in pediatric cohorts
- Deep learning applied to biological problems
- Transfer learning could apply to cross-cancer type predictions

---

## **Q73: Explain KL-divergence and why it was better than logistic loss for methylation prediction?**

### **Answer:**

**KL-Divergence (Kullback-Leibler)** measures the difference between two probability distributions.

### **Why it matters for methylation:**

1. **Methylation is continuous:** CpG methylation values range from 0 to 1 (e.g., 0.51 vs 0.99)
2. **Logistic loss requires binarization:**

If methylation > 0.5 → label = 1 (methylated)

If methylation < 0.5 → label = 0 (unmethylated)

Problem: 0.51 and 0.99 both become "1" – information loss!

3. **KL-divergence preserves continuous values:**

$$D_{KL}(y, \hat{y}) = y \cdot \log(y/\hat{y}) + (1-y) \cdot \log((1-y)/(1-\hat{y}))$$

- Treats each CpG's methylation as a probability distribution
- Penalizes predictions differently based on true methylation level
- No information loss from binarization

### Results:

- KL-divergence alone: 29.43% improvement over logistic loss
- The benefit was additive with transfer learning (different mechanisms)

### Connection to ProteinPaint:

- Custom loss functions for biological data
- Understanding that standard ML approaches may not fit biological constraints

---

## Q74: What is transfer learning and how did you apply it to epigenomics?

### Answer:

**Transfer Learning:** Using knowledge learned from one domain (source) to improve learning in a related but different domain (target).

### Our Application:

Source Domain: Somatic tissue methylomes

- Coverage: 5–21%
- Profiles: Sperm, mammary gland, liver, heart, etc.
- Purpose: Learn general methylation patterns

↓ Transfer learned weights ↓

Target Domain: Embryo methylomes

- Coverage: 0.06–1.47%
- Profiles: Oocytes, 2-cell, 4-cell, 8-cell, 16-cell embryos
- Purpose: Predict missing CpG sites

## What we transferred:

1. **Sequence subnetwork:** DNA patterns that predict methylation
2. **Methylation subnetwork:** Local CpG correlation patterns
3. **Joint subnetwork:** Fused feature representations

## Key findings:

- Fine-tuning transferred models was essential
- Methylation patterns transferred better than sequence patterns
- Different tissues share fundamental methylation biology

## Impact:

- 22.45% improvement from transfer learning
  - Enabled analysis impossible with raw sparse data
- 

## Q75: Describe the ATAC-seq paper on cardiac fibroblasts. What was your contribution?

### Answer:

**Paper:** "The landscape of accessible chromatin in quiescent cardiac fibroblasts and cardiac fibroblasts activated after myocardial infarction" (Epigenetics 2022)

### Biological Context:

After heart attack (MI), quiescent cardiac fibroblasts:

- Proliferate and migrate to injury site
- Differentiate into myofibroblasts
- Further become matrifibrocytes
- Mediate scar formation and tissue repair

### My Contribution:

I worked on **computational analysis** of the data:

1. **ATAC-seq Analysis Pipeline:**
  - Quality control and alignment

- Peak calling and annotation
- Differential accessibility analysis
- Integration with RNA-seq data

## 2. Bioinformatics Integration:

- Correlating chromatin accessibility with gene expression
- Identifying regulatory elements (promoters, enhancers)
- Motif enrichment analysis
- Gene regulatory network construction

## Key Findings:

- ATAC-seq peaks enriched at promoters and distal enhancers
- Positive correlation between promoter accessibility and gene expression
- Identified transcription factors driving fibroblast state transitions
- Some gene expression changes independent of chromatin accessibility

## Technical Methods:

- RNA-seq + ATAC-seq integration
  - Time course analysis (multiple post-MI time points)
  - Motif discovery and TF binding prediction
  - Gene regulatory network inference
- 

# Q76: What is ATAC-seq and how does it work?

## Answer:

**ATAC-seq:** Assay for Transposase-Accessible Chromatin with high-throughput sequencing

## Principle:

The Tn5 transposase preferentially inserts sequencing adapters into **open/accessible chromatin** regions where DNA is not tightly bound by nucleosomes or other proteins.

## Protocol:

1. Cells/nuclei isolation
2. Tn5 transposition (tagmentation)
  - Inserts adapters into open chromatin
  - Simultaneously fragments DNA
3. PCR amplification
4. Sequencing
5. Map reads → identify accessible regions (peaks)

### What accessible chromatin tells us:

- **Promoters:** Actively transcribed genes
- **Enhancers:** Distal regulatory elements
- **TF binding sites:** Where transcription factors can bind
- **Nucleosome-free regions:** Open chromatin structure

### Analysis Pipeline:

1. **Read QC:** FastQC, remove adapters
2. **Alignment:** Bowtie2/BWA to reference genome
3. **Peak calling:** MACS2 (identify accessible regions)
4. **Differential analysis:** DiffBind, DESeq2 on peak counts
5. **Motif analysis:** HOMER, MEME for TF motifs
6. **Integration:** Correlate with RNA-seq for expression

### Relevance to ProteinPaint:

- ATAC-seq data can be visualized as tracks
- Epigenetic data integration in cancer samples
- Understanding gene regulation beyond mutations

---

## Q77: How do you integrate ATAC-seq with RNA-seq data?

### Answer:

#### Integration Approaches:

1. **Correlation Analysis:**

For each gene:

- ATAC signal at promoter (TSS  $\pm$  2kb)
- RNA-seq expression (TPM/FPKM)
- Calculate Pearson/Spearman correlation

Expect: Positive correlation for most genes

Reality: Not all genes show this - other mechanisms matter

## 2. Differential Analysis Integration:

Condition A vs B:

- Differentially accessible peaks (ATAC)
- Differentially expressed genes (RNA)

Overlap analysis:

- Genes with both  $\uparrow$  accessibility AND  $\uparrow$  expression
- Genes with changes in one but not other

## 3. Regulatory Element Assignment:

- Assign distal peaks to nearest gene
- Or use chromatin conformation data (Hi-C)
- Build enhancer-promoter maps

## 4. Transcription Factor Analysis:

From ATAC peaks:

- Identify enriched TF motifs
- Which TFs have accessible binding sites?

From RNA-seq:

- Is that TF expressed?
- Do its targets show expression changes?

## Tools Used:

- **DiffBind**: Differential ATAC peak analysis
- **ChIPseeker**: Peak annotation
- **HOMER/MEME**: Motif discovery
- **SCENIC**: Gene regulatory network inference

- **Custom R/Python:** Integration scripts
- 

## Q78: What did you learn from the Acta2 knockout cardiac fibroblast study?

### Answer:

**Paper:** "Loss of Acta2 in cardiac fibroblasts does not prevent the myofibroblast differentiation or affect the cardiac repair after myocardial infarction" (J Mol Cell Cardiol 2022)

### Background:

- **Acta2** encodes smooth muscle alpha-actin (SMaA)
- SMaA is THE canonical marker of myofibroblast differentiation
- Myofibroblasts form stress fibers containing SMaA
- Everyone assumed SMaA was required for myofibroblast function

### Surprising Result:

Cardiac fibroblast-specific Acta2 knockout mice had:

- **Normal survival** after MI
- **Normal cardiac function** and histology
- **Normal myofibroblast differentiation** (proliferation, migration, contractility)

### Mechanism - Compensation:

Acta2 deleted → Other actins upregulated:

- Actg2 (smooth muscle gamma-actin) ↑↑↑
- Acta1 (skeletal muscle alpha-actin) ↑↑
- Cytoplasmic actins maintain function

### Key Insight:

- Actin isoforms have **functional redundancy**
- "Canonical markers" may not be functionally essential
- Biology has backup mechanisms



## Relevance:

- Questions assumptions about marker genes
  - Understanding that knockout studies reveal compensation
  - Importance of testing assumptions experimentally
- 

## Q79: How do you handle sparse single-cell data computationally?

### Answer:

#### Challenges with Sparse Data:

- Missing values dominate (>95% missing in some cases)
- Cannot directly apply bulk methods
- Need specialized imputation/completion strategies

#### Approaches I've Used:

##### 1. Transfer Learning (my paper):

- Pre-train on denser data
- Transfer learned patterns
- Fine-tune on sparse target

##### 2. Feature Engineering:

- Use neighboring CpG methylation
- DNA sequence context (1001bp windows)
- Distance-weighted neighbors

##### 3. Loss Function Design:

- Only compute loss on known values
- Weight by confidence/coverage
- KL-divergence for continuous values

##### 4. Data Augmentation:

- Multi-task learning across samples
- Share parameters between related profiles
- Leverage biological similarity

##### 5. Confidence-Aware Prediction:

- Only keep predictions above threshold ( $\tau=0.8$ )
- Report uncertainty alongside predictions

### General Single-Cell Strategies:

- **scRNA-seq**: MAGIC, scVI, SAVER imputation
  - **scATAC-seq**: ChromVAR, cisTopic
  - **scMethyl**: My approach, DeepCpG
- 

## Q80: What are gene regulatory networks and how did you construct them?

### Answer:

**Gene Regulatory Network (GRN):** A model of how transcription factors (TFs) regulate target genes.

### Components:

Nodes: Genes (TFs and targets)  
Edges: Regulatory relationships  
Direction: TF  $\rightarrow$  Target gene  
Weight: Strength of regulation

### How We Built GRNs (ATAC-seq paper):

#### 1. Identify Active TFs:

ATAC-seq peaks  $\rightarrow$  Motif analysis  $\rightarrow$  Enriched TF motifs  
Filter by: TF must be expressed (RNA-seq > threshold)

#### 2. Identify TF Targets:

Peaks with TF motif  $\rightarrow$  Nearest gene (or Hi-C linkage)  
= Putative targets of that TF

#### 3. Validate by Expression:

If TF expression changes:

- Do predicted targets change accordingly?
- Positive targets should correlate positively

#### 4. **Network Construction:**

- Edge: TF → Target if motif in accessible region near gene
- Weight: Based on ATAC signal, motif score, expression correlation

#### **Tools:**

- **SCENIC:** Single-cell regulatory network inference
- **PANDA/LIONESS:** Network inference from expression
- **ChIP-seq integration:** When available
- **Custom approaches:** ATAC + RNA-seq correlation

#### **Example Finding:**

Identified TFs like AP-1, TEADs driving fibroblast activation after MI

---

## **Q81: What bioinformatics pipelines have you built or used?**

### **Answer:**

#### **Pipelines I've Worked With:**

##### 1. **Methylation Analysis:**

Raw WGBS → Bismark alignment → Methylation calling  
→ Quality filtering → Per-CpG methylation levels  
→ Differential methylation → Visualization

##### 2. **ATAC-seq Pipeline:**

FastQ → Bowtie2 alignment → Remove duplicates  
→ MACS2 peak calling → DiffBind differential  
→ Motif analysis → Peak annotation → Visualization

##### 3. **RNA-seq Pipeline:**

FastQ → STAR/HISAT2 alignment → featureCounts  
→ DESeq2 differential expression → GO/pathway analysis

#### 4. Deep Learning Pipeline:

Data preprocessing → Train/val/test split  
→ Model architecture (CNN+GRU)  
→ Training with transfer learning  
→ Evaluation (F1, accuracy) → Imputation

#### Tools I'm Comfortable With:

- **Aligners:** BWA, Bowtie2, STAR, Bismark
- **Peak callers:** MACS2
- **Differential:** DESeq2, DiffBind
- **Languages:** Python (TensorFlow), R (Bioconductor)
- **Workflow:** Snakemake, shell scripting

---

## Q82: How would you visualize DNA methylation data in ProteinPaint?

**Answer:**

#### Visualization Approaches:

##### 1. Track-Based View:

Genome browser with:

- CpG methylation level (0–1) as signal track
- CpG islands annotation track
- Gene annotation track
- ATAC-seq peaks (chromatin accessibility)

##### 2. Heatmap View:

Rows: CpG sites or genomic regions

Columns: Samples

Color: Methylation level (blue=unmethylated, red=methylated)

Use cases:

- Compare tumor vs normal
- Cluster samples by methylation pattern
- Identify differentially methylated regions

### 3. Lollipop-Style for Promoters:

X-axis: Position relative to TSS

Y-axis: Methylation level

Color: Sample type or clinical variable

Show: Promoter methylation associated with silencing

### 4. Integration View:

Stacked tracks:

- Methylation
- ATAC-seq accessibility
- Gene expression
- Histone marks

Show: Inverse correlation of promoter methylation and expression

### Technical Considerations:

- Aggregation for visualization (too many CpGs)
- Smoothing/binning approaches
- File formats: bedMethyl, BigWig for methylation

---

## Q83: What challenges exist in visualizing epigenetic data at scale?

Answer:

## 1. **Data Volume:**

Human genome: ~28 million CpG sites  
Each sample: methylation values at each CpG  
Multi-sample study: Billions of data points

Solution: Aggregate into regions, streaming, server-side computation

## 2. **Multi-dimensional:**

Dimensions to show:

- Genomic position
- Methylation level
- Sample identity
- Clinical variables
- Gene associations

Solution: Interactive filtering, linked views, progressive disclosure

## 3. **Missing Data:**

Single-cell: 95%+ missing  
Bulk: Low-coverage regions

Solution: Imputation, confidence indicators, mask missing

## 4. **Biological Context:**

Need: Gene annotations, CpG islands, regulatory elements  
Challenge: Layering without cluttering

Solution: Collapsible tracks, zoom-dependent detail

## 5. **Statistical Significance:**

Differential methylation: Which changes are real?  
Need: P-values, FDR correction, confidence intervals

Solution: Statistical overlays, filtering by significance

## How ProteinPaint Handles This:

- Tabix indexing for efficient range queries
  - Server-side aggregation
  - Progressive rendering
  - Multi-track genome browser
- 

## Q84: Describe your experience with deep learning for genomics.

Answer:

### My Experience (Methylome Paper):

#### 1. Architecture Design:

```
# Hybrid CNN-RNN model

# Sequence branch (CNN)
- Input: 1001bp one-hot encoded DNA
- Conv1D layers with batch normalization
- Extract sequence patterns predicting methylation

# Methylation branch (GRU)
- Input: 50 neighboring CpGs + distances
- Bidirectional GRU captures local correlation

# Joint module
- Concatenate features
- Fully connected layers
- Multi-task output (one head per sample)
```

#### 2. Transfer Learning Implementation:

```
# Pre-train on source (somatic tissues)
source_model = train(source_data, epochs=100)

# Transfer to target (embryos)
target_model = copy(source_model)
target_model = finetune(target_data, epochs=50)
```

### 3. Custom Loss Function:

```
def kl_divergence(y_true, y_pred):
    return y_true * log(y_true/y_pred) + (1-y_true) * log((1-y_true)/(1-y_pred))
```

### 4. Practical Considerations:

- Hyperparameter tuning (learning rate, batch size)
- Overfitting on small datasets
- GPU memory management
- Reproducibility (seeds, versioning)

### Relevance to ProteinPaint:

- AI chatbot uses similar ML concepts
- Could apply to variant effect prediction
- Transfer learning for cross-cancer analysis

---

## Q85: How do your publications connect to cancer genomics and ProteinPaint?

**Answer:**

**Connections:**

### 1. Epigenetic Data Visualization:

My experience: Methylation + ATAC-seq + RNA-seq integration  
ProteinPaint need: Multi-omics data visualization

I can contribute: Design patterns for epigenetic tracks



## 2. Sparse Data Handling:

My experience: Single-cell data with >95% missing

ProteinPaint context: Pediatric cohorts with limited samples

I can contribute: Imputation strategies, confidence visualization

## 3. Machine Learning Integration:

My experience: Deep learning for genomic prediction

ProteinPaint direction: AI chatbot, predictive features

I can contribute: ML pipeline development, model integration

## 4. Gene Regulatory Analysis:

My experience: GRNs from ATAC-seq, TF motif analysis

ProteinPaint relevance: Understanding cancer gene regulation

I can contribute: Regulatory visualization features

## 5. Scientific Communication:

My experience: Published in peer-reviewed journals

ProteinPaint need: Documentation, papers, presentations

I can contribute: Manuscript preparation, figure design

---

# Q86: What would you do differently if you repeated your methylome study?

**Answer:**

**Improvements I'd Make:**

## 1. Model Architecture:

Then: CNN + GRU hybrid  
Now: Try transformer architectures (attention)  
Why: Better at capturing long-range dependencies

## 2. **Transfer Learning Strategy:**

Then: Transfer entire subnetworks  
Now: Explore domain adaptation techniques  
Why: More sophisticated ways to handle domain shift

## 3. **Validation:**

Then: Held-out chromosomes  
Now: Additional biological validation  
Why: Check if imputed values make biological sense

## 4. **Reproducibility:**

Then: GitHub code, GEO data  
Now: Docker containers, Snakemake workflow  
Why: Easier for others to reproduce and extend

## 5. **Clinical Application:**

Then: Focused on embryos  
Now: Consider cancer applications  
Why: Methylation patterns critical in cancer

## **What This Shows:**

- Reflective thinking
  - Keeping up with advances
  - Focus on practical improvements
-

## **Q87: What questions do you have about epigenetic data in ProteinPaint?**

### **Potential Questions to Ask:**

1. "What types of epigenetic data does ProteinPaint currently support? Are there plans to add methylation or ATAC-seq visualization?"
  2. "How does GenomePaint handle multi-omics integration when not all data types are available for every sample?"
  3. "Is there interest in adding imputation features for sparse data, similar to what I developed?"
  4. "What's the current approach to visualizing regulatory elements alongside mutation data?"
  5. "Are there ongoing collaborations with groups generating single-cell epigenetic data from pediatric cancers?"
-

# 🔍📋 Quick Reference Cheat Sheet

## Key Numbers to Remember

### File Format Stats:

- VCF: 8 mandatory columns (CHROM, POS, ID, REF, ALT, QUAL, FILTER, INFO)
- MAF: 34+ columns standard (Hugo\_Symbol, Chromosome, Position, etc.)
- BED: 3 required columns (chrom, chromStart, chromEnd), 0-based
- BAM: Binary Alignment Map, sorted by coordinates

### Reference Genomes:

- hg19 (GRCh37): Released 2009, chr1-22,X,Y naming
- hg38 (GRCh38): Released 2013, better centromere/telomere assembly

### TP53 Stats (for examples):

- Gene location: chr17:7,668,421-7,687,490 (hg38)
- Protein length: 393 amino acids
- Most common mutations: R175H, R248Q, R273H

### Visualization Thresholds:

- SVG practical limit: ~1,000 elements
- Canvas comfortable: 10,000-100,000 points
- WebGL necessary: >100,000 points

## Key D3.js Patterns

```
// Data Join Pattern (memorize this!)
const selection = container
  .selectAll('.item')
  .data(data, (d) => d.id) // key function
  .join(
    (enter) => enter.append('circle').attr('class', 'item'),
    (update) => update,
    (exit) => exit.remove()
  );

// Common Scales
d3.scaleLinear(); // continuous → continuous
d3.scaleLog(); // logarithmic
d3.scaleBand(); // categorical → continuous bands
d3.scaleOrdinal(); // categorical → categorical (colors)

// Colorblind-safe palettes
d3.schemeTableau10; // 10 distinct colors
d3.schemeCategory10; // alternative
```

## Key Statistical Methods

Survival Analysis:

- Kaplan-Meier:  $S(t) = \prod (1 - d_i/n_i)$
- Log-rank test: Compare survival curves (p-value)
- Hazard ratio:  $HR > 1$  = worse outcome

Multiple Testing:

- Bonferroni:  $p_{adj} = p \times n$  (conservative)
- Benjamini-Hochberg: Controls FDR at q-level

Genomic Stats:

- VAF =  $\text{alt\_reads} / \text{total\_reads}$  (expect ~50% germline, varies somatic)
- Fisher's exact: 2x2 contingency table significance

# Data Processing Pipeline Reference

## Common Tools by Step:

| Step            | Tools                           |
|-----------------|---------------------------------|
| QC              | FastQC, fastp, MultiQC          |
| Alignment       | BWA-MEM, STAR (RNA), minimap2   |
| Deduplication   | Picard MarkDuplicates, sambamba |
| Variant Calling | GATK, Mutect2, Strelka2         |
| Annotation      | VEP, SnpEff, ANNOVAR            |
| Quantification  | featureCounts, Salmon, kallisto |

## MAPQ Quality Scores:

- 0: Ambiguous (multi-mapped)
- 20: ~99% confidence
- 60: Uniquely mapped

## CIGAR Operations:

- M: Match/mismatch
- I: Insertion
- D: Deletion
- N: Skipped (intron)
- S: Soft clip

## RNA-seq Normalization:

- TPM: Compare genes within sample
- DESeq2: Compare samples (differential expression)
- Z-score: Row scaling for heatmaps

# Technology Stack Quick Reference

| Component  | Purpose           | Key Features                      |
|------------|-------------------|-----------------------------------|
| D3.js      | Visualization     | Data binding, scales, transitions |
| Express.js | Backend framework | REST API, middleware, routing     |

| Component         | Purpose            | Key Features                               |
|-------------------|--------------------|--------------------------------------------|
| <b>TypeScript</b> | Type safety        | Interfaces, generics, compile-time checks  |
| <b>Zod</b>        | Runtime validation | Schema definition, parsing, error messages |
| <b>PostgreSQL</b> | Relational data    | Complex queries, JSONB, indexing           |
| <b>Redis</b>      | Caching            | Session storage, query caching             |
| <b>tabix</b>      | File indexing      | Range queries on VCF/BED                   |
| <b>Rust/WASM</b>  | Performance        | CPU-intensive parsing, near-native speed   |

## Interview Answer Templates

### Technical question:

1. Explain the concept
2. Show code example
3. Discuss trade-offs
4. Connect to ProteinPaint

### Behavioral question (STAR):

- **Situation:** Set the context
- **Task:** Your responsibility
- **Action:** What you specifically did
- **Result:** Outcome with metrics

### "I don't know" recovery:

"I haven't worked directly with [X], but I have experience with [similar Y]. My approach would be to [research strategy]. Is that the kind of experience you're looking for?"

---

*Last Updated: December 15, 2025*

*Good luck with your interview! 🎯🚀*