Advanced RNA-Seq Differential Expression: Every dataset is a snowflake, and limma is a flamethrower

Ryan C. Thompson

November 21, 2016

Why care about "advanced" RNA-seq analysis?

- "I just want to do a simple analysis, nothing fancy!"
- When your experiment is well designed and your data quality is good, you get to use simple statistics. Yay!
- When your experimental design is problematic or your data has issues, your analysis gets more complicated
- You don't get to choose the problems your data has, so you need to be prepared with the solutions
- I will cover how to identify and correct several common design and data issues in RNA-seq experiments

Common themes watch for in this presentation

All of the techniques covered have the same underlying theme:

- Taking advantage of many genes to make up for few samples ("large p, small N")
- Identifying systematic violations of model assumptions across multiple genes that could not be detected gene-by-gene
- Relaxing model assumptions to accommodate these violations
- Estimating model parameters robustly despite small sample sizes by sharing information between genes
- Avoiding "circular reasoning" issue because no single gene contributes too much to the estimation
- In short: taking advantage of "large p" to make up for the problem of "small N"

Outline

1. Review of "standard" RNA-seq analysis paradigm

- Basic statistical model
- Your role in defining the experimental design
- Do's and Don'ts

2. Leveraging the large number of genes to evaluate your model

- Using p-value distributions as model QC
- Explaining FDR, q-value, etc. in terms of p-value distributions

3. Problems your data might have and how to solve them

1. Want to use ordinary lm instead of glm? Use `voom`
2. Outlier genes? Use `robust=TRUE`
3. Outlier samples/groups? Use `arrayWeights`
4. Nested factors? Use `duplicateCorrelation` (mixed model)
5. Unobserved batch effects? Use `sva`
6. Nonlinear effects? Use `ns` for natural spline regression

Review of basic RNA-seq analysis

The Negative Binomial Generalized Linear Model

Let's unpack that, working backwards from the end:

- "Linear model": Generalization of t-test, F-test, ANOVA, linear regression, etc.
- "Generalized linear model": Like linear models, but don't assume normal distribution
- Negative binomial
- Input is a matrix of read counts: genes $\times$ samples
- Technical noise for counts is known to be binomial-distributed, but is well-approximated by the simpler Poisson distribution
- Experimental noise is assumed *a priori* to be gamma-distributed
- Gamma-Poisson mixture yields negative-binomial distribution

Negative-binomial dispersion estimation

- Dispersion parameter (biological variance) must be estimated from few samples *before* fitting GLM
- Normally requires many samples for a stable estimate: individual gene dispersions will be highly inaccurate
- So estimate a single dispersion parameter from all samples
- Use empirical Bayes to get robust estimate by taking the mean dispersion across all genes as a prior and squeezing point estimates toward the mean
- Account for heteroskedasticity (i.e. mean-variance dependence) by allowing the prior to be a smooth function of abundance instead of a constant
- Counter the overall downward bias of dispersion MLE using quasi-likelihood methods

Raw dispersion estimates: unbiased but unstable

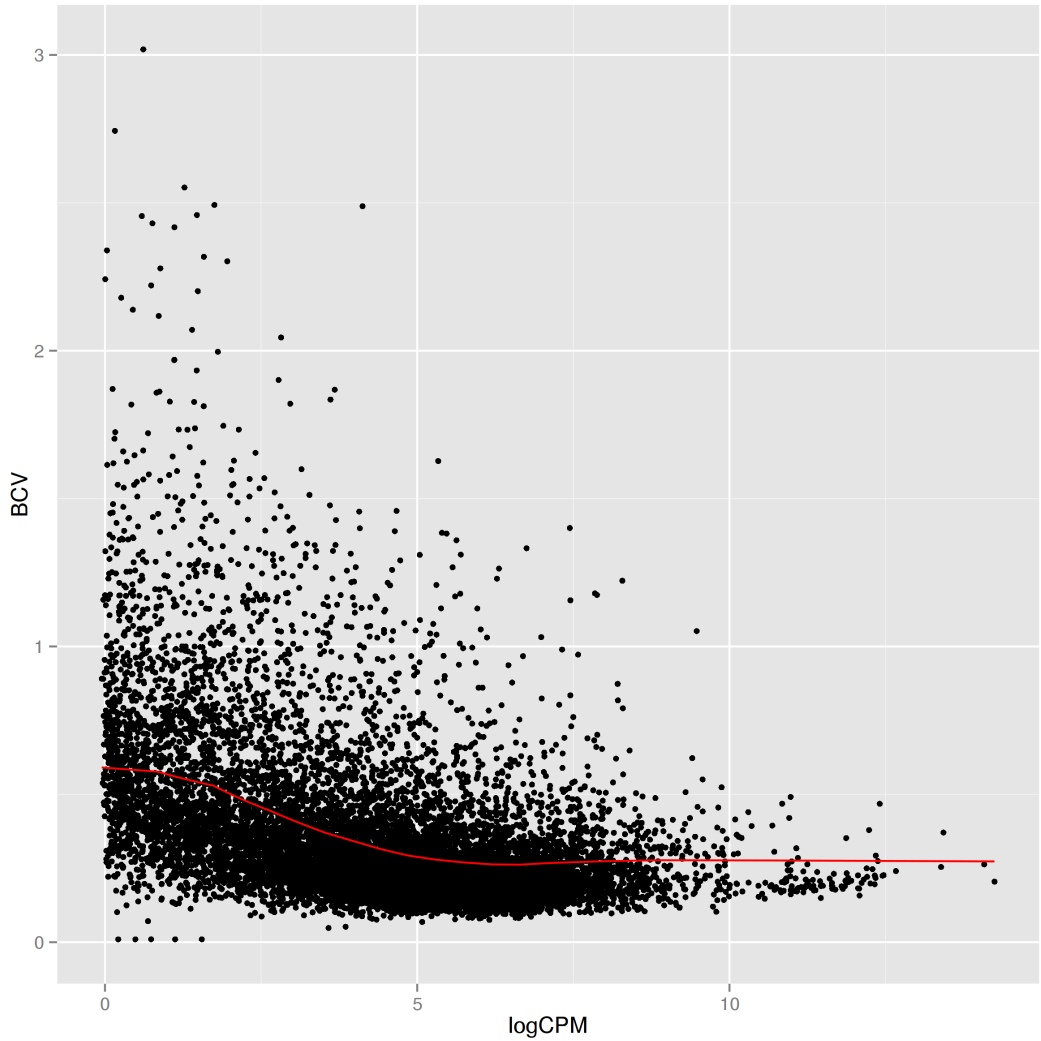


Figure 1: Raw genewise dispersions

Empirical Bayes regularizes dispersions at the cost of bias

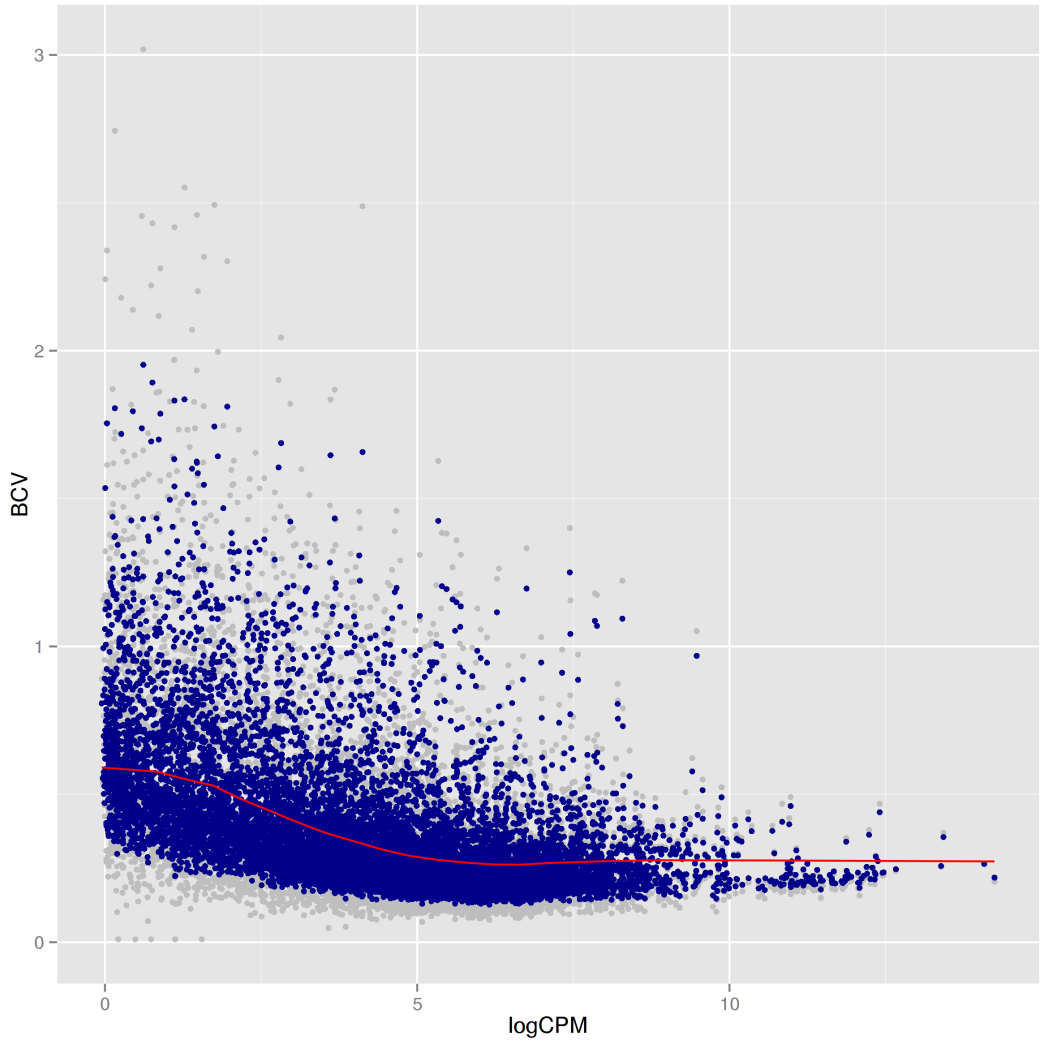


Figure 2: Empirical Bayes squeezing of dispersions

Your input as a bioinformatician

- edgeR & limma packages do all the math for you
- *Your* main job is to construct the design matrix and specify contrasts
- You are the interpreter, translating your experimental (or someone else's) design into mathematical language
- This is the only step that isn't plug-and-chug. You must carefully consider how best to represent your experiment.

Basic RNA-seq do's and don'ts

DO

- Build your design matrix and contrasts during the planning stage, *before* performing the experiment (and do power calculations while you're at it)
- Normalize for composition bias
- Analyze the raw read counts + normalization factors, not normalized counts (RSEM estimated counts are OK)
- Use PCoA plots (`plotMDS` function in edgeR) and other EDA to explore the data
- Filter low-count genes before estimating dispersions
- Fit a single model with all samples
- Use edgeR's `estimateDisp` function
- Use edgeR's quasi-likelihood F-test (`glmQLFTest`) or limma-voom

Basic RNA-seq do's and don'ts

DO NOT

- Feed normalized counts/CPM/RPKM/FPKM to edgeR/DESeq2/voom
- Rush through building your design matrix & contrasts
- Skip exploratory data analysis
- Analyze a different subset of samples for each test
- Use edgeR's `estimateGLM*Disp` functions (they're obsolete)
- Use edgeR's `glmLRT` with genewise dispersions

Why not glmLRT?

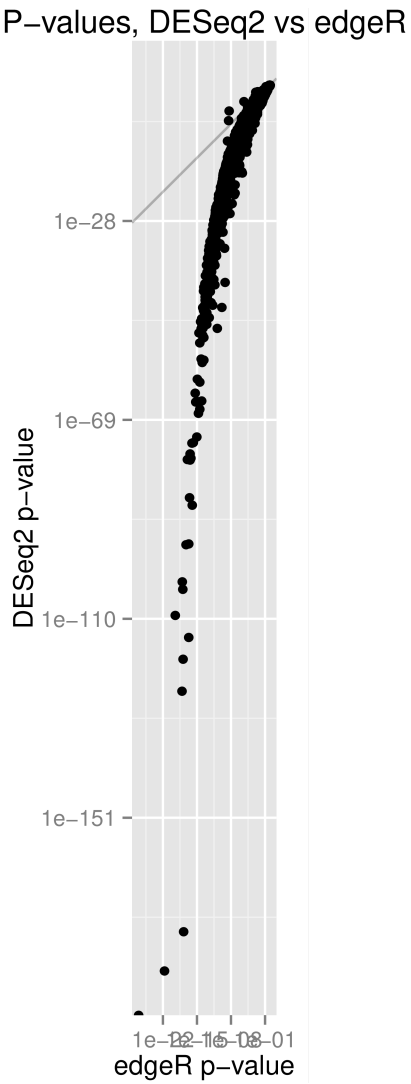


Figure 3: edgeR QL vs DESeq2 LRT

Leveraging the large number of genes to evaluate your model

High dimensionality: Curse, or Blessing in Disguise?

- Conventional wisdom says doing lots of tests is "bad"
- Lots of tests means lots of opportunities for false positives
- Thus, multiple testing correction is a requirement: can't use raw p-values (But could you *ever* really use them?)
- Effectively, significance threshold seems to get more stringent as you do more tests (But does it really?)
- How can you leverage the large number of tests we're doing to get more information about your data and more statistical power out of your model?

Frequentist statistics *likes* having lots of tests

- Frequentist statistics makes statements about average behavior across many repetitions of the same experiment
- $p = 0.05$ means "you would get this result at random 5% of the time if you repeated the test thousands of times", not "5% chance of being wrong"
- The former statement is not useful for a single test - you'd rather have the latter.
- But fitting the same model to thousands of genes means we actually *are* doing many repetitions of the "same" experiment!
- So frequentist statements that seemed pointless for a single test are suddenly highly relevant
- We can take advantage of this to actually get the *real* "chance of being wrong"

Properties of P-values

- By construction, a p-value is distributed as *Uniform(0,1)* under the null hypothesis, and biased toward zero under the alternative
- Using this knowledge, we can estimate the uniform and non-uniform components of a distribution of p-values
- Then we compute the estimated proportions of true null and alternative hypotheses at a given p-value threshold by computing ratios of areas
- "Estimated proportion of true null hypotheses" should sound familiar...
- Surprise! We invented the False Discovery Rate!

FDR: Important definitions and distinctions

- FDR: expected number of false positives in a *list* of genes – does not tell the probability of any one gene being a false positive
- Important: FDR is a general term for any false discovery rate calculation – remember to specify a specific method of computing FDR
- Benjamini-Hochberg: algorithm to put an *upper bound* on the FDR
- π_0: Estimated proportion of all null hypotheses that are true (non-DE genes), a.k.a. prior probability of non-DE
- local FDR: probability that a given gene is non-DE
- q-value: Algorithm for unbiased estimation of FDR; more liberal than BH, but q-value might overestimate significance
- Specifically q-value equals BH FDR times π_0; or equivalently BH FDR is q-value under the pessimistic assumption that $\pi_0 = 1$
- These definitions are best understood in graphical terms

Typical P-value distribution: all null

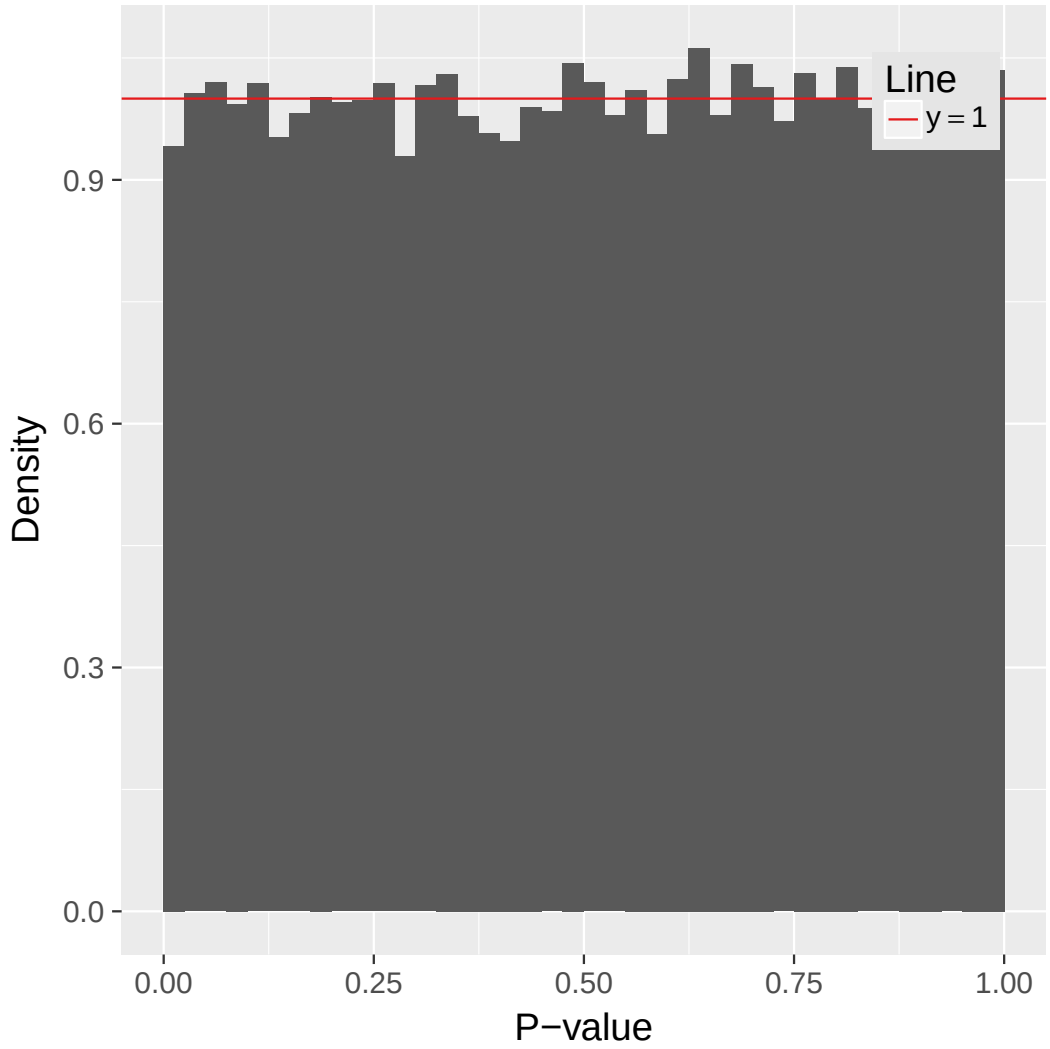


Figure 4: P-value distribution with no signal

Typical P-value distribution: moderate signal

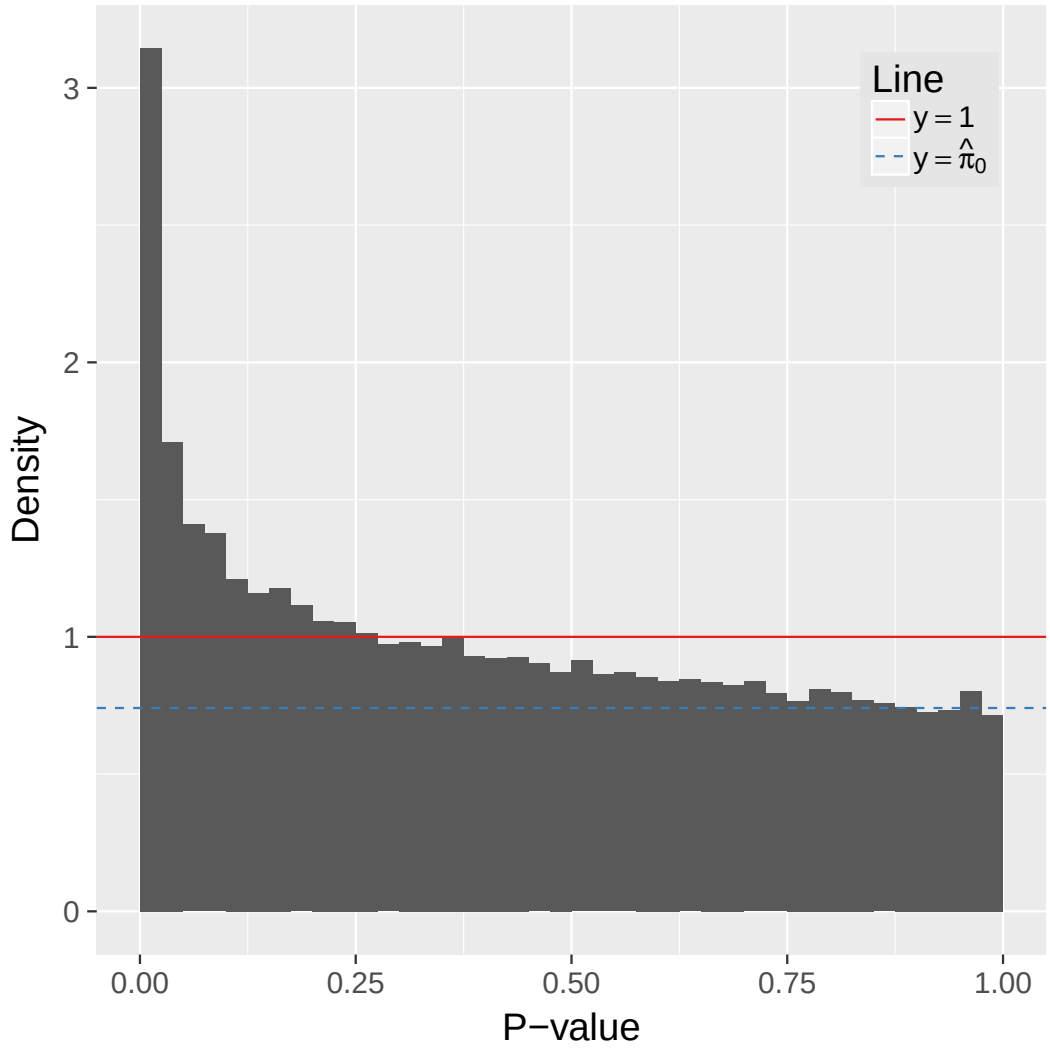


Figure 5: P-value distribution with moderate signal

Typical P-value distribution: moderate signal

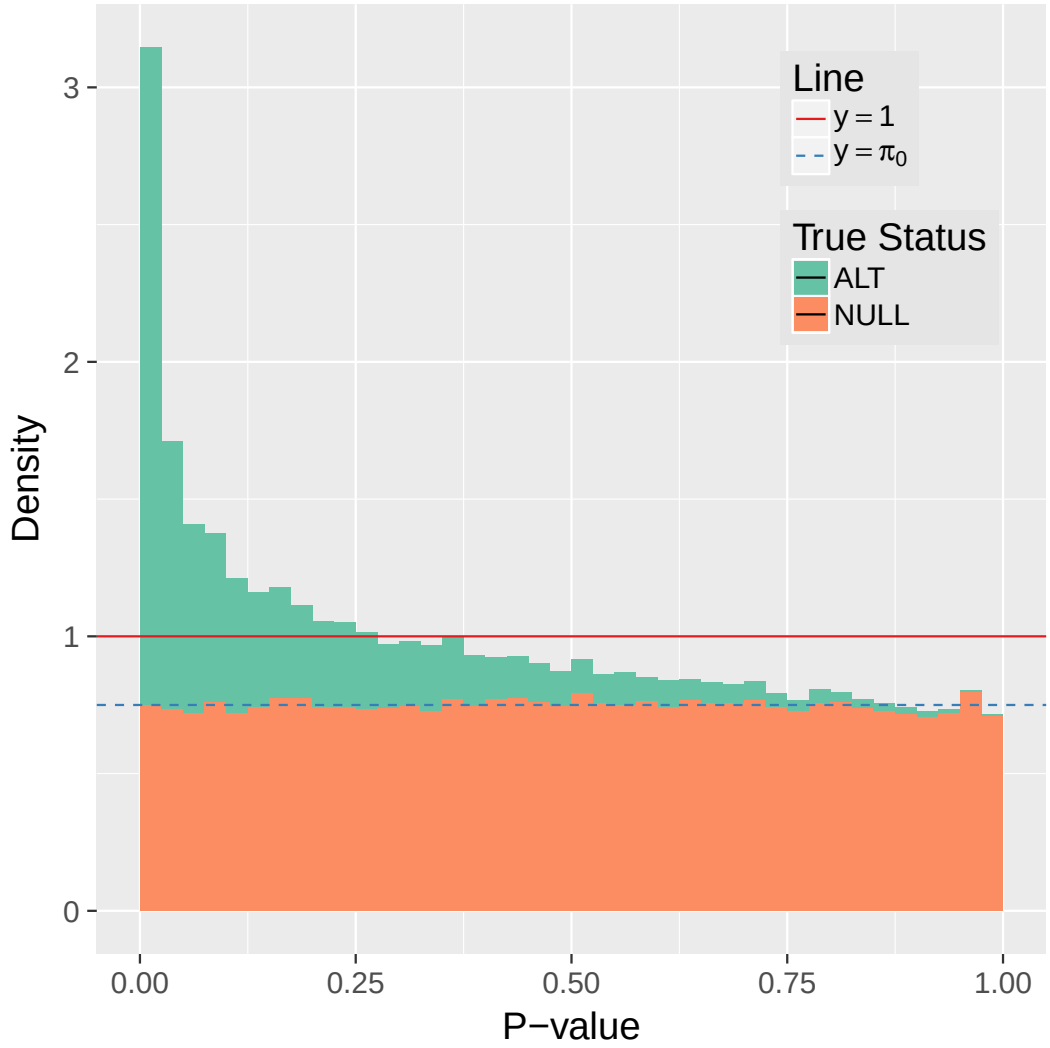


Figure 6: P-value distribution with moderate signal, colored by true status

Typical P-value distribution: moderate signal

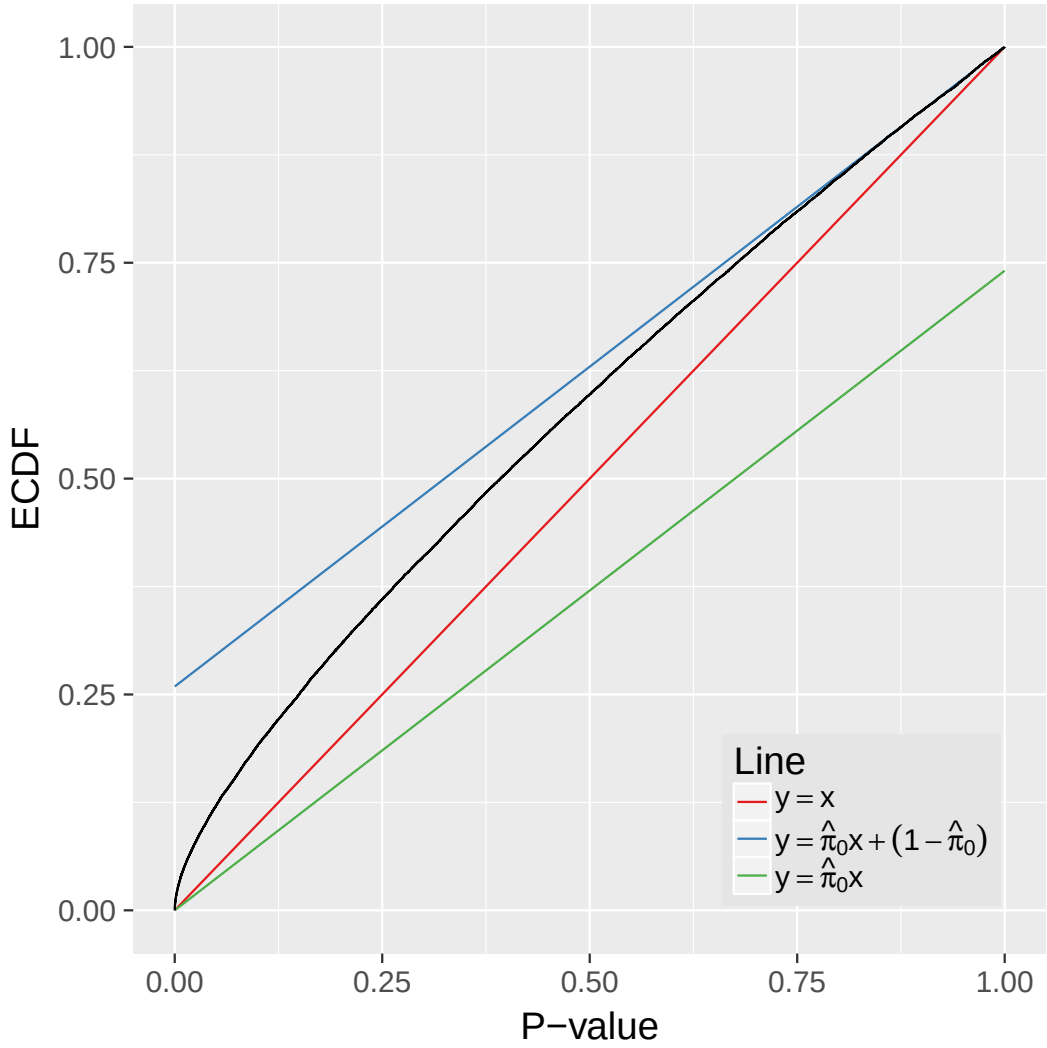


Figure 7: P-value ECDF with moderate signal

Typical P-value distribution: strong signal

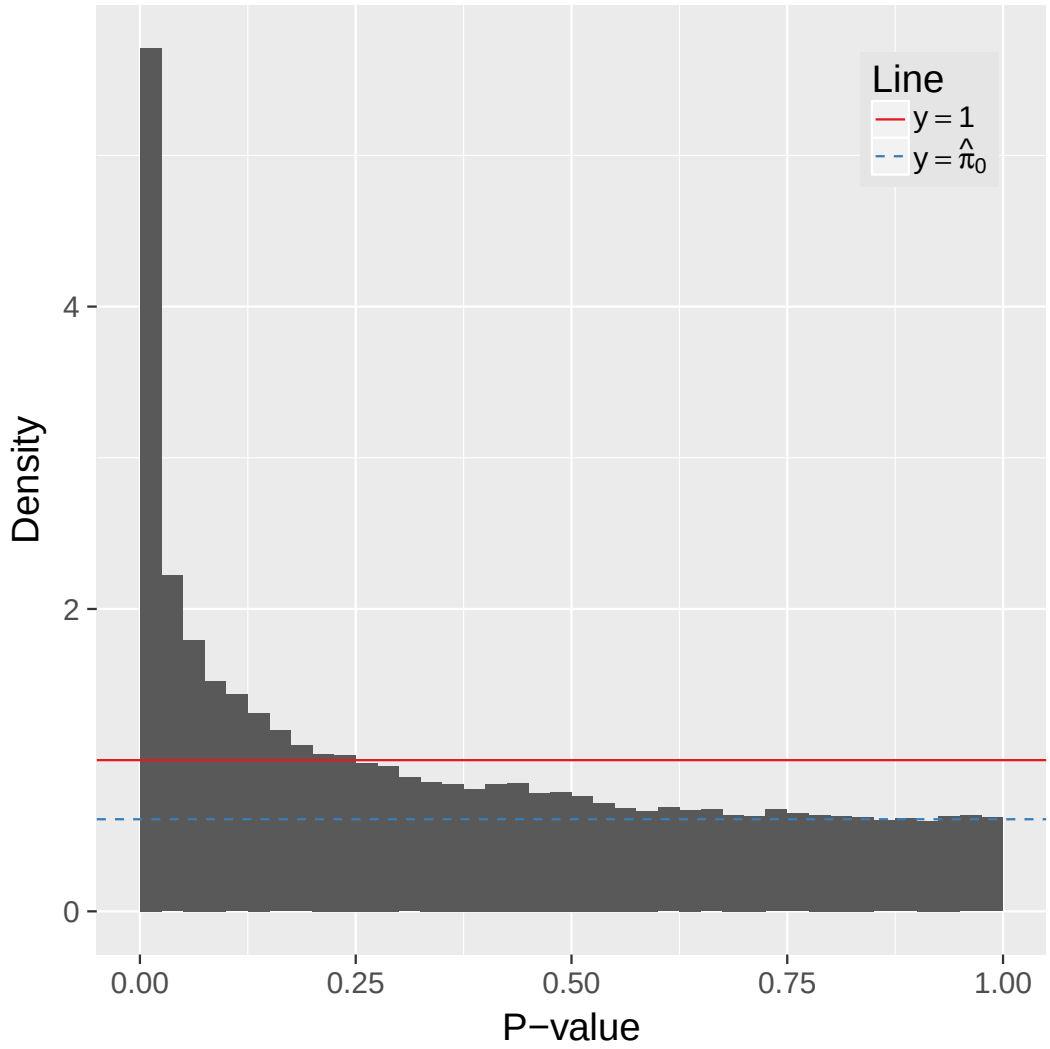


Figure 8: P-value distribution with strong signal

Typical P-value distribution: strong signal

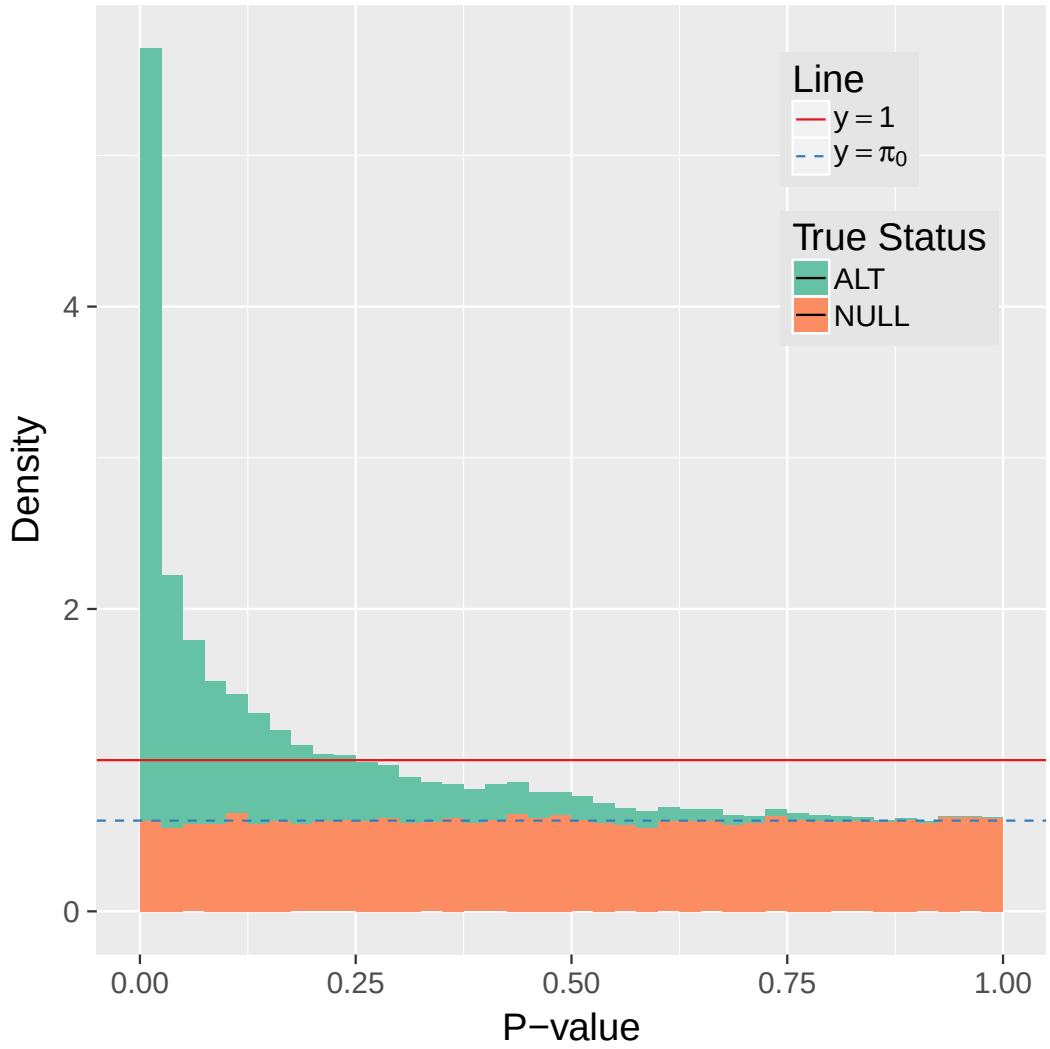


Figure 9: P-value distribution with strong signal, colored by true status

Typical P-value distribution: strong signal

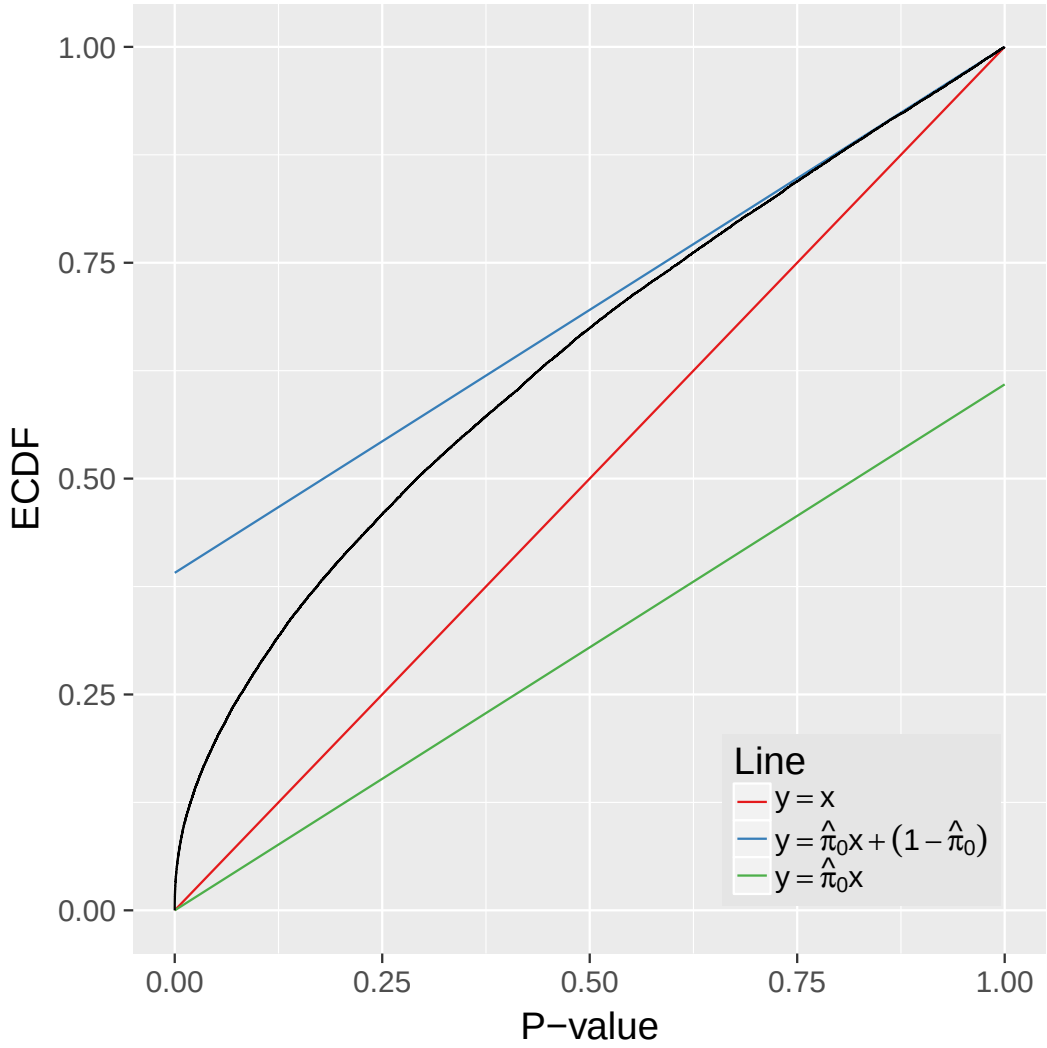


Figure 10: P-value ECDF with strong signal

Typical P-value distribution: weak signal

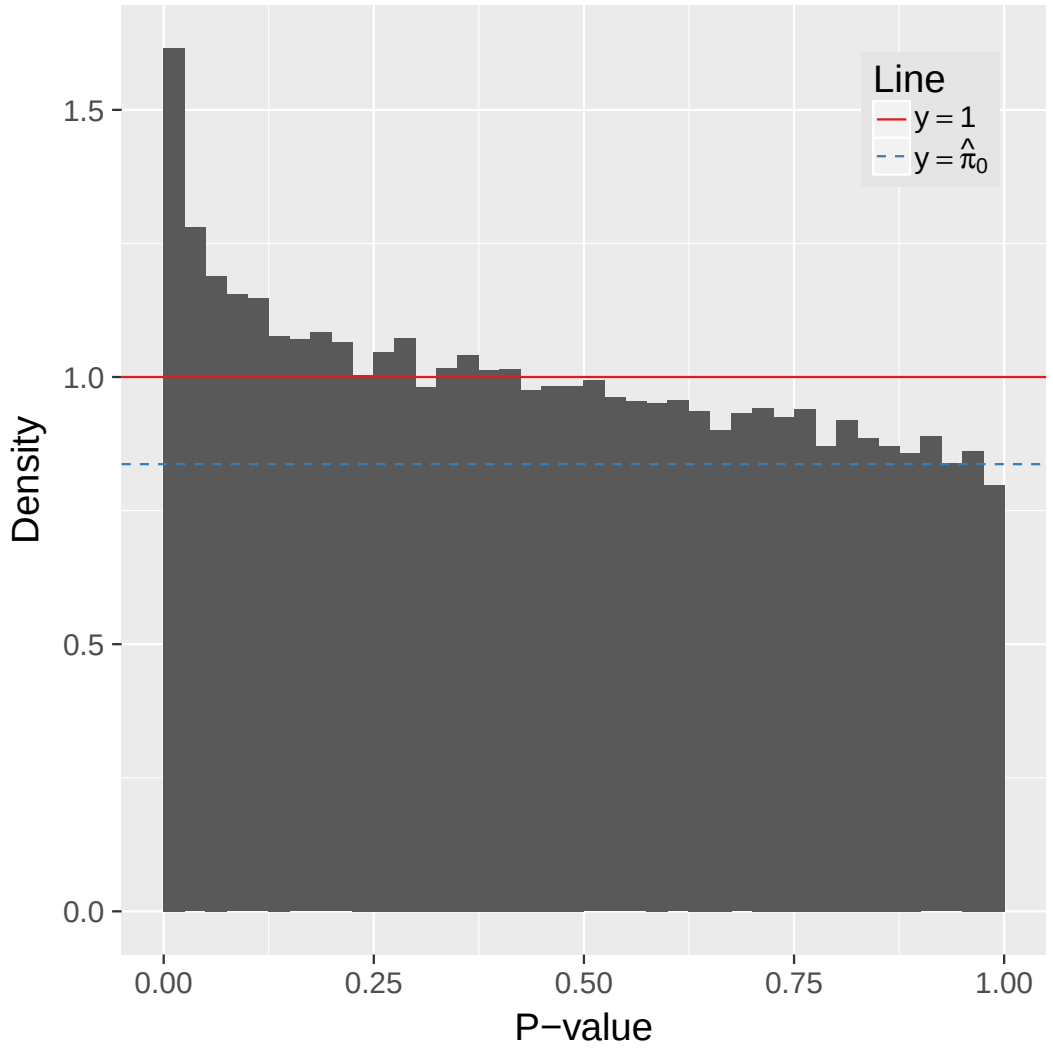


Figure 11: P-value distribution with weak signal

Typical P-value distribution: weak signal

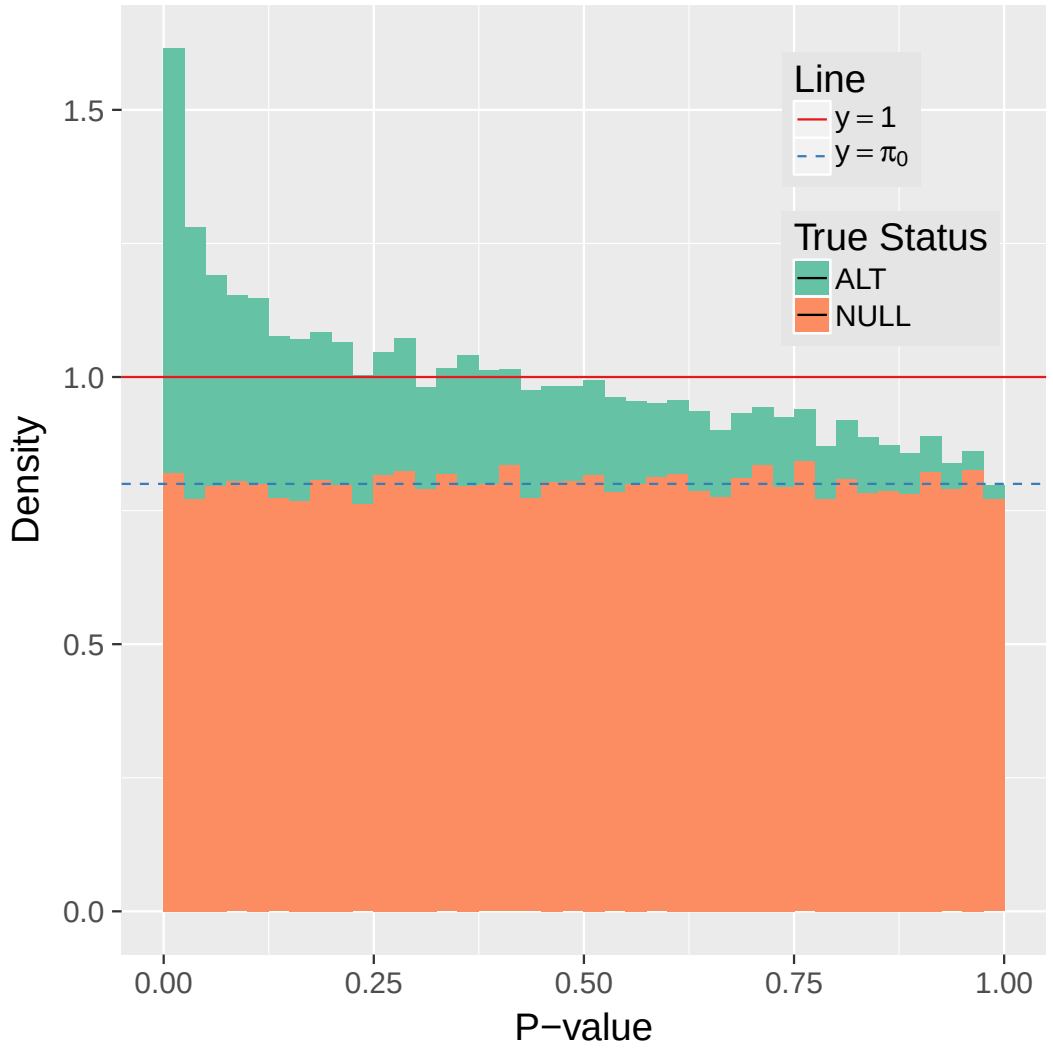


Figure 12: P-value distribution with weak signal, colored by true status

Typical P-value distribution: weak signal

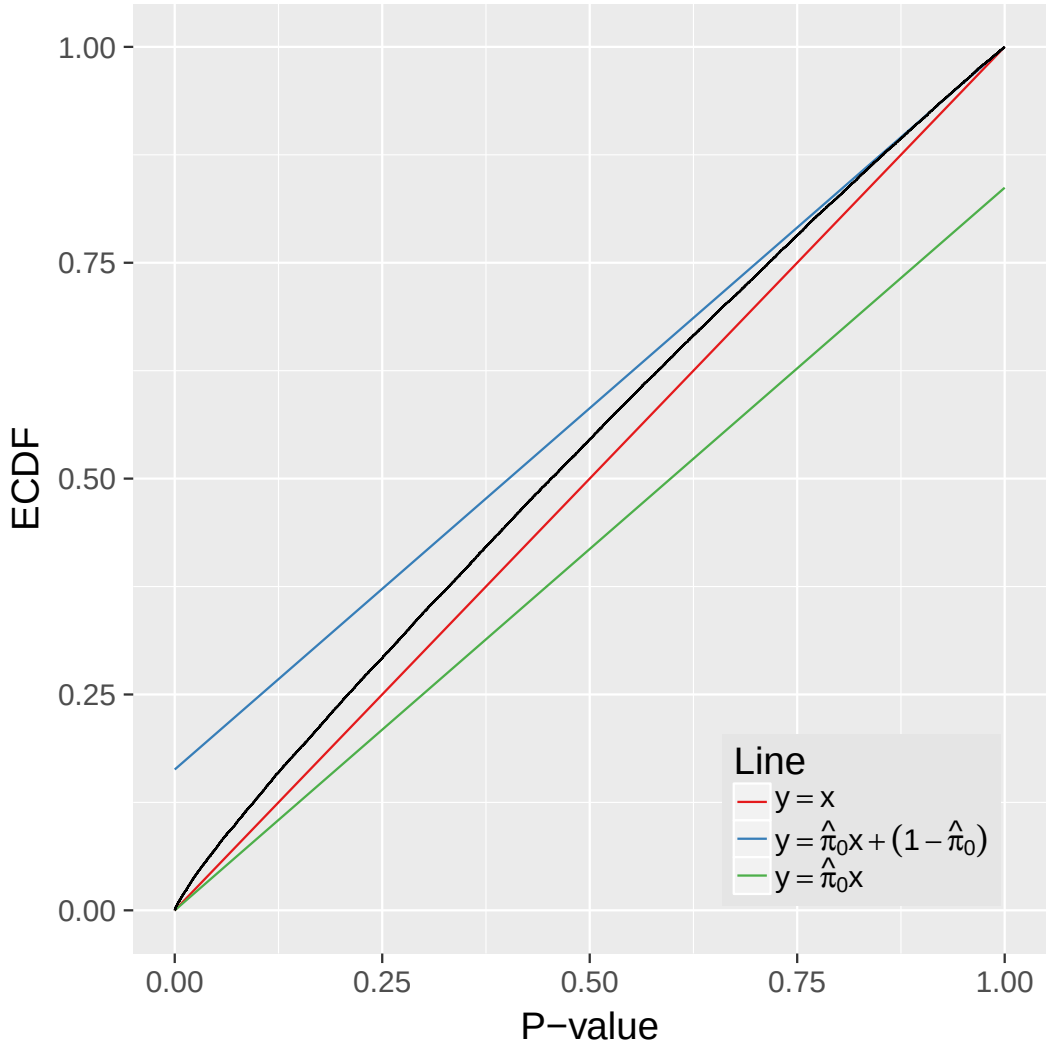


Figure 13: P-value ECDF with weak signal

Evaluating your model using the p-value distribution

- Every p-value distribution should either be uniform or zero-biased
- Any other distribution indicates that your model does not fit the data - Fix your model!
- FDR methods will not have a useful interpretation for such a p-value distribution
- Possible issues:
  - Critical assumptions of your model are severely violated (e.g. heteroskedasticity, wrong distribution, too many outliers)
  - Covariates/batch effects not included in your model
  - Highly correlated covariates are splitting the effect size
  - Unobserved batch effect or other confounding factor is obscuring the signal
  - Accidentally treated continuous variable as categorical or vice versa (common R pitfall!)
- CANNOT be explained by simple lack of signal

Atypical P-value distribution: Over-Conservative

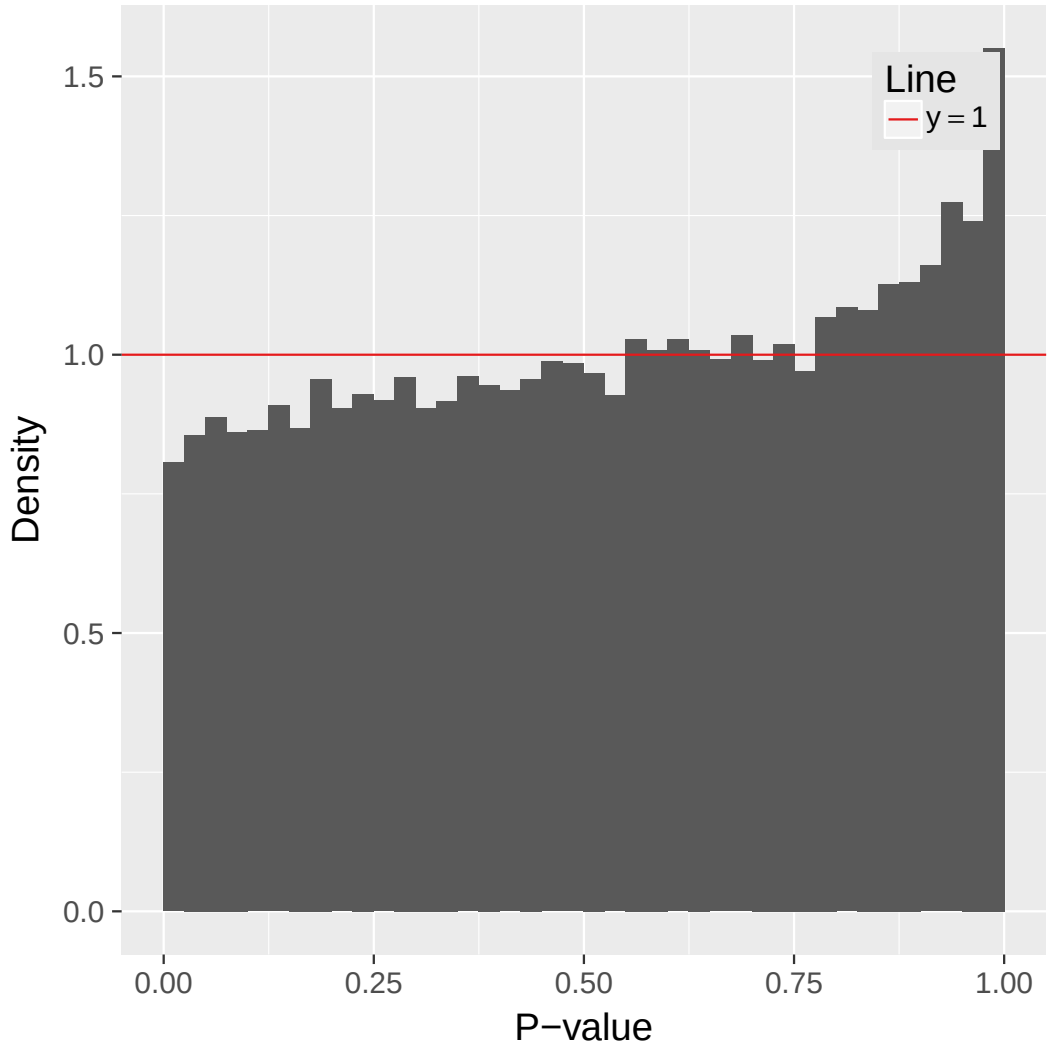


Figure 14: P-value distribution, worse than uniform

Atypical P-value distribution: Bimodal

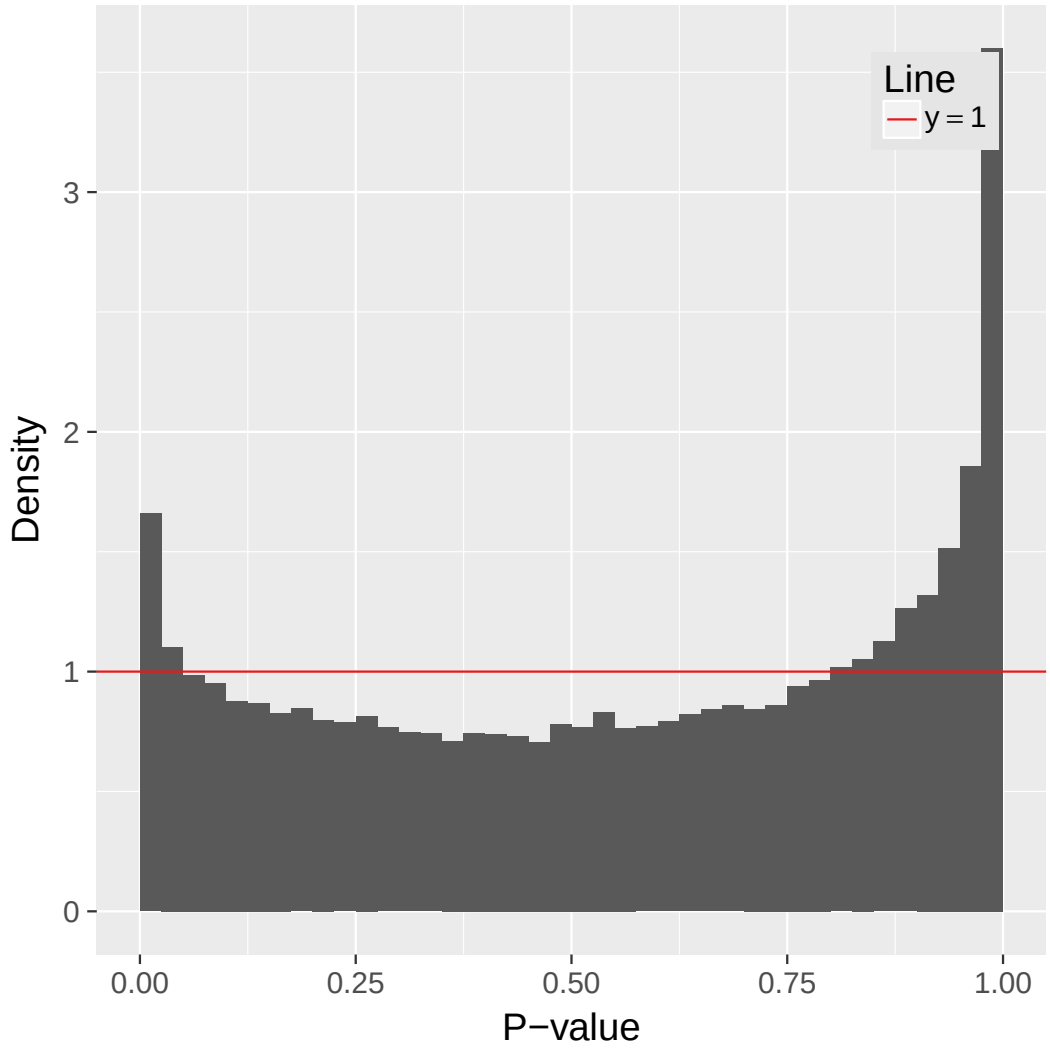


Figure 15: P-value distribution, bimodal

Atypical P-value distribution: Bump in the middle

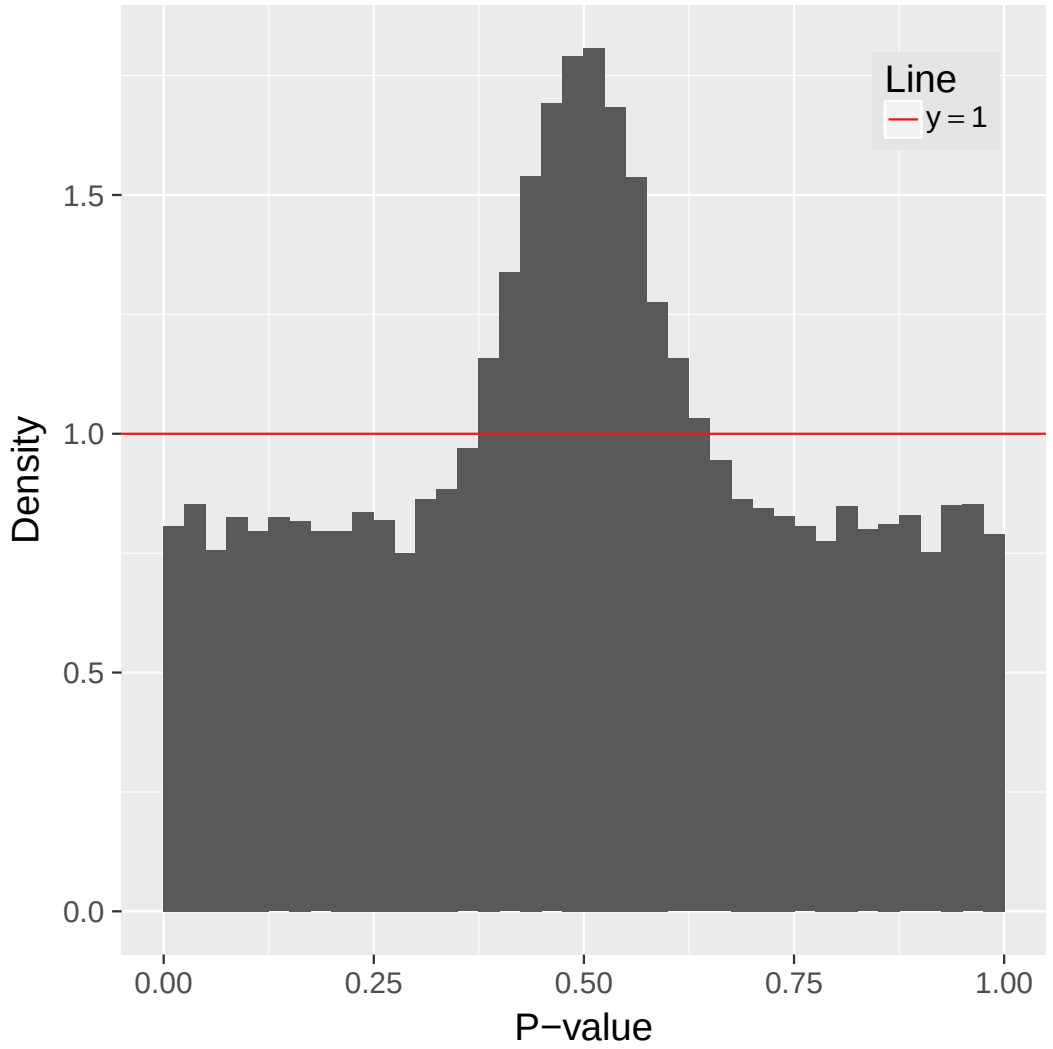


Figure 16: P-value distribution, non-monotonic

Atypical P-value distribution: Discrete values

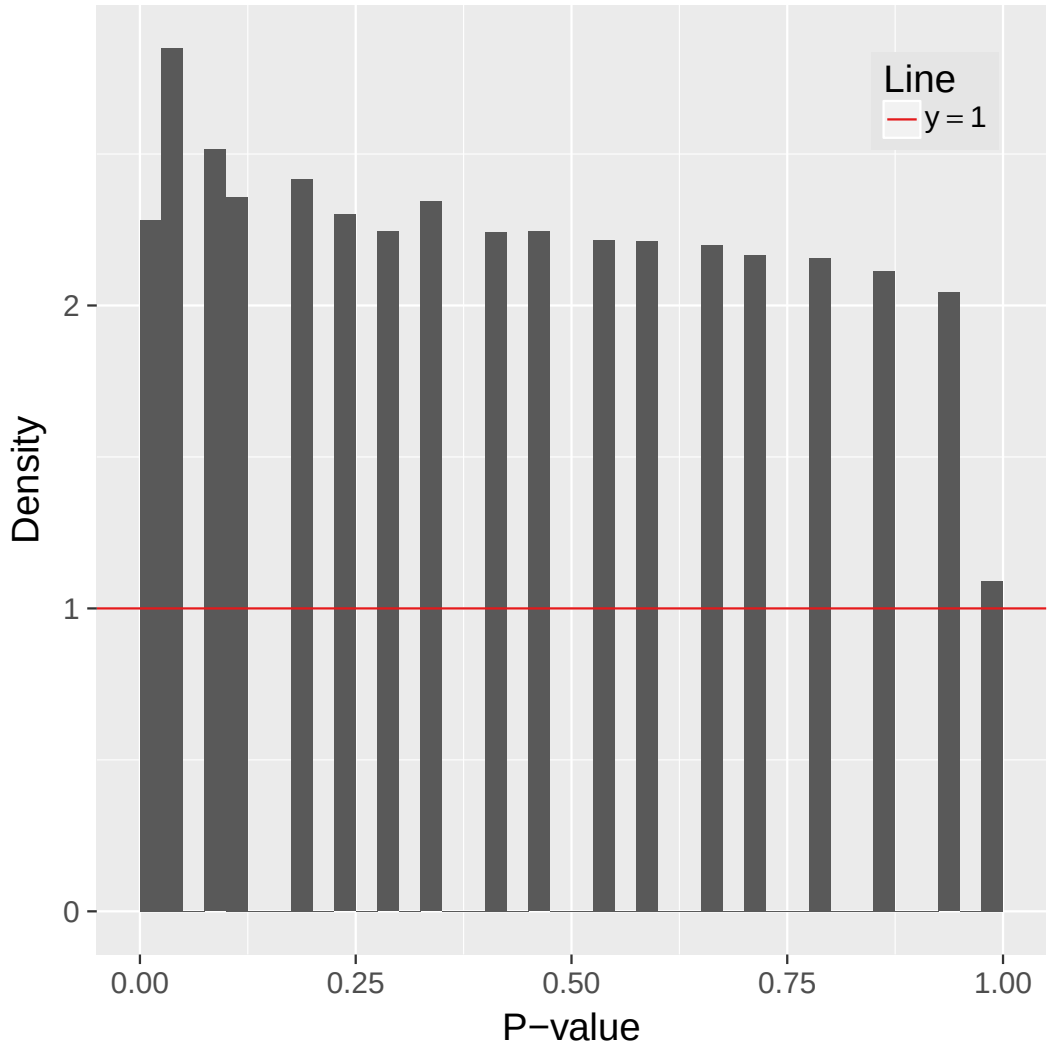


Figure 17: P-value distribution, discrete

A large number of tests is actually good!

- *"significance threshold seems to get more stringent as you do more tests"?* Not true!
- The important parameters governing significance are:
  - π_0, the estimated proportion of true null hypotheses
  - Effect strength, the fold chances of DE genes
- Example: when looking for needles in a haystack, your threshold for identifying needles is based on:
  - What fraction of your haystack is needles?
  - How similar-looking are needles and hay stalks?
  - *Not* the size of the haystack
- With many tests, FDR answers the question we really care about: What is the probability that a gene is differentially expressed?
- Don't cry over losing p-values, because they were never answering a question you cared about anyway

Data problems and solutions using limma

Problem 1: We want to use a linear model, not a GLM

- Fitting GLMs is slow (iteratively reweighted least squares); fitting linear models is much faster (least squares), especially for large datasets (100s of samples)
- Many downstream analysis methods are only implemented for linear models, with no analogous algorithms implemented for GLMs (although they could be)
- Especially true of very new methods and methods originally designed for microarrays
- However, heteroskedasticity makes linear models a poor fit for count data, hence the use of Poisson/NB distributions that account for mean-variance relationship
- "close your eyes and pretend it's microarray data" is not a good strategy

"voom: precision weights unlock linear model analysis tools for RNA-seq read counts"

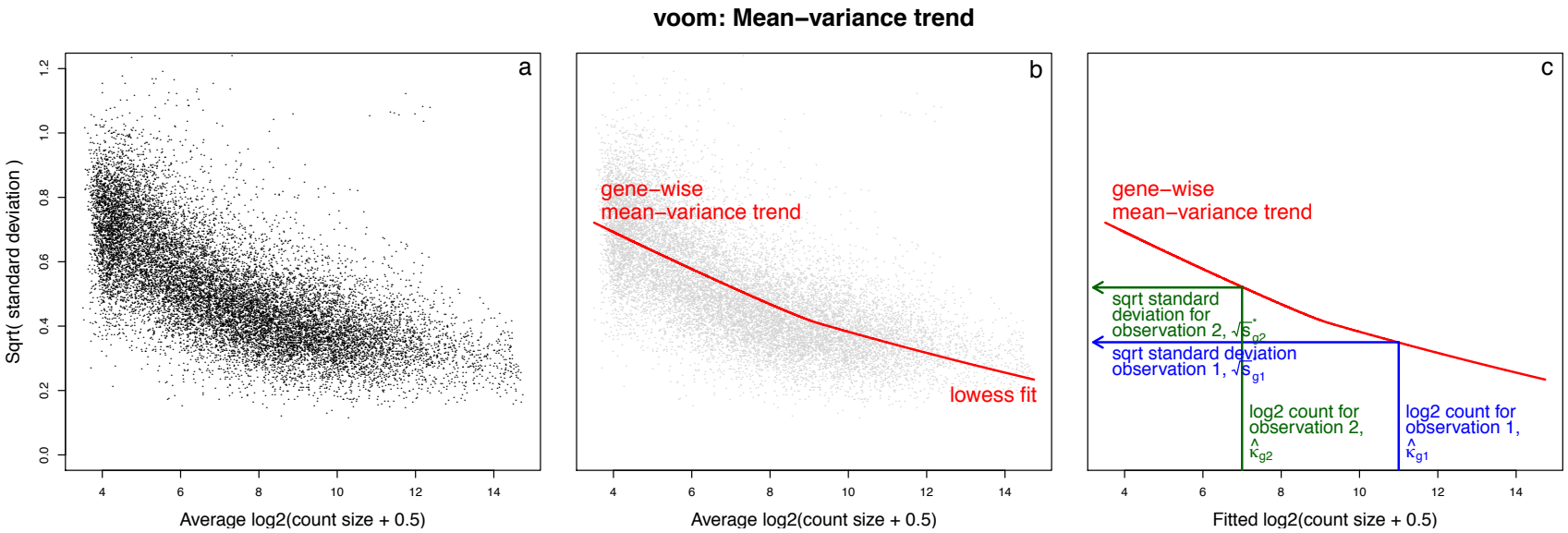


Figure 18: Diagram of voom method

"voom: precision weights unlock linear model analysis tools for RNA-seq read counts"

- voom models the mean-variance trend, and then applies that trend via precision weights to each observation
- Mean-variance trend is computed on a raw count scale, which accounts for differences in counting precision at different abundances *and* different sequencing depths
- In short: voom weights let you use limma and other linear model techniques on RNA-seq counts
- We model the variance trend, and assume the model is robust against moderate violations of the normality assumption
- In practice, limma-voom and edgeR perform very similarly on the same design
- But limma has more flexibility to model experimental designs that edgeR cannot (e.g. mixed models, sample heteroskedasticity)

limma-voom vs edgeR, p-values

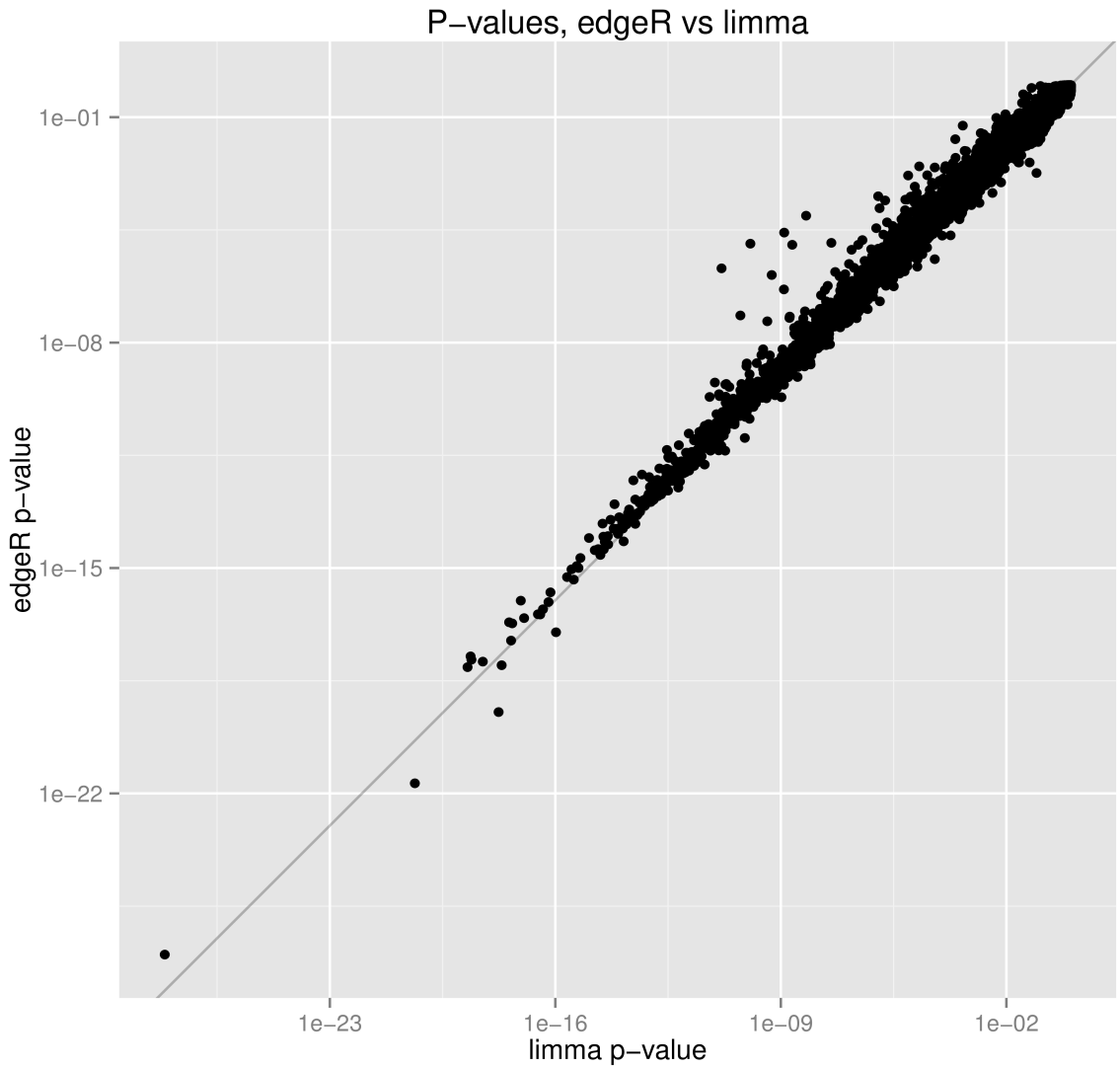


Figure 19: P-values on real data, edgeR QL vs limma-voom

Problem 2: empirical Bayes squeezing underestimates the variance of high-variance genes

- Both limma and edgeR use empirical Bayes squeezing of variances/dispersions
- This pulls everything toward the mean, so above-average variances are systematically underestimated
- This can lead to false positives in high-variance genes
- Solution: use `robust=TRUE` in `eBayes` (for limma) and `estimateDisp`/`glmQLFit` (for edgeR)
- This penalizes genes with outlier variances by reducing the strength of squeezing at the extremes
- You should pretty much always use this option

Dispersion estimation: no squeezing

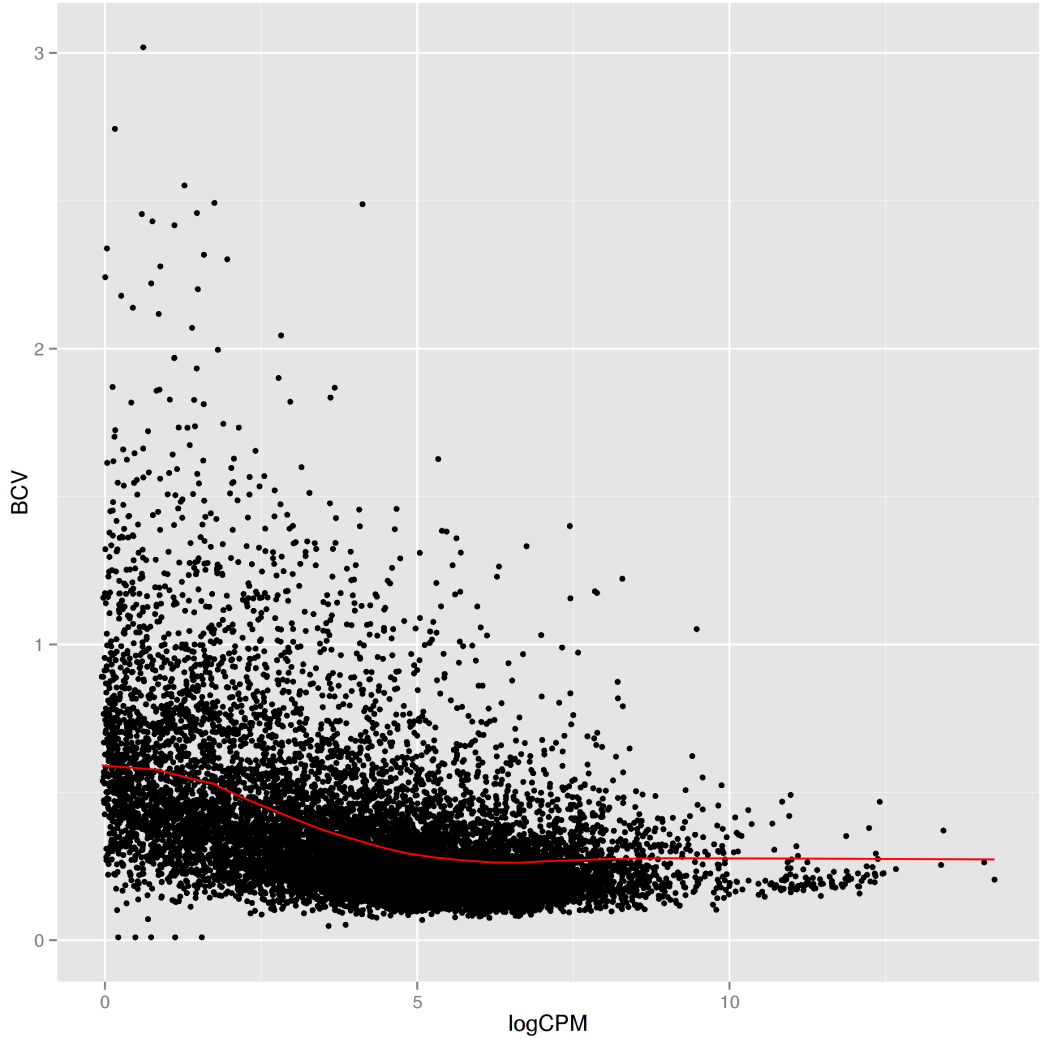


Figure 20: Raw genewise dispersions

Dispersion estimation: equal squeezing for all genes

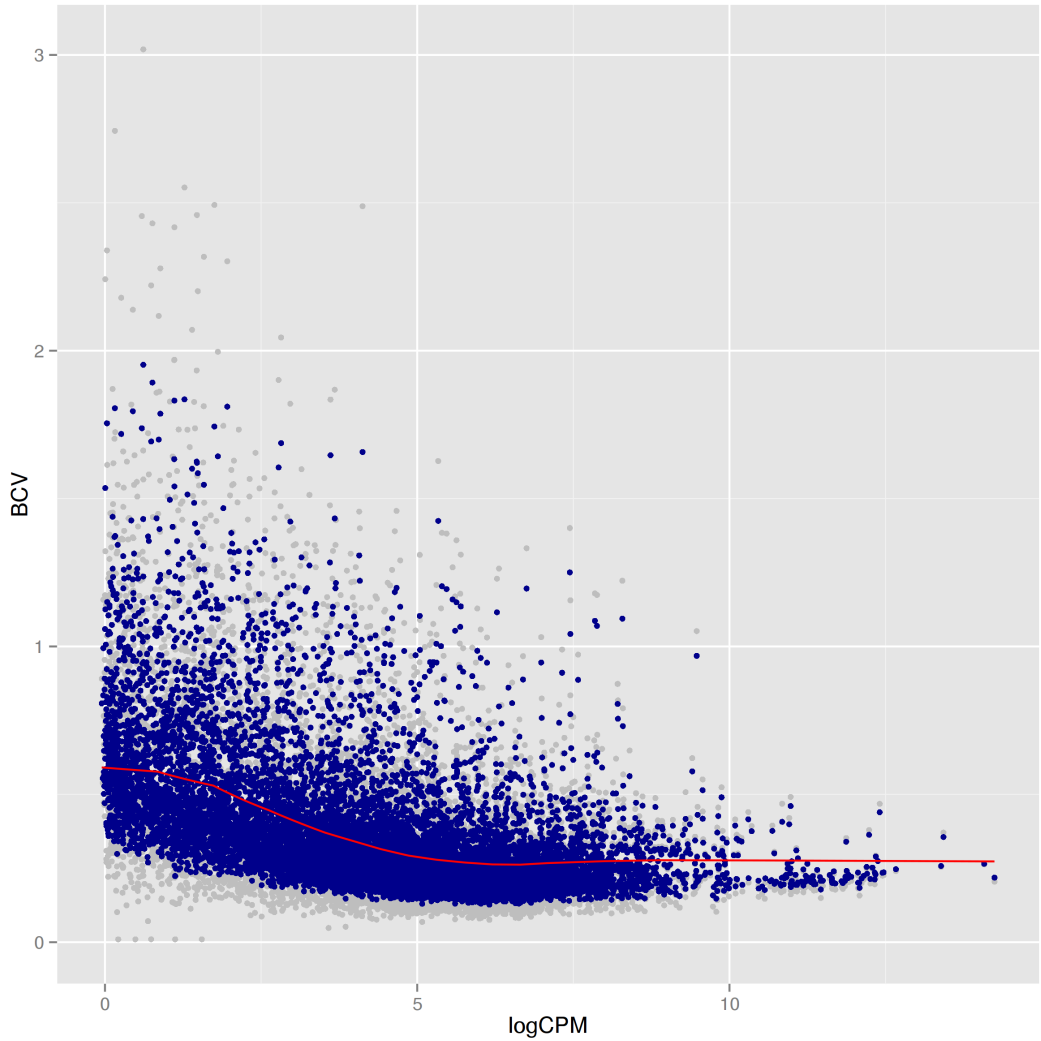


Figure 21: Empirical Bayes squeezed vs raw dispersions

Dispersion estimation: robust squeezing

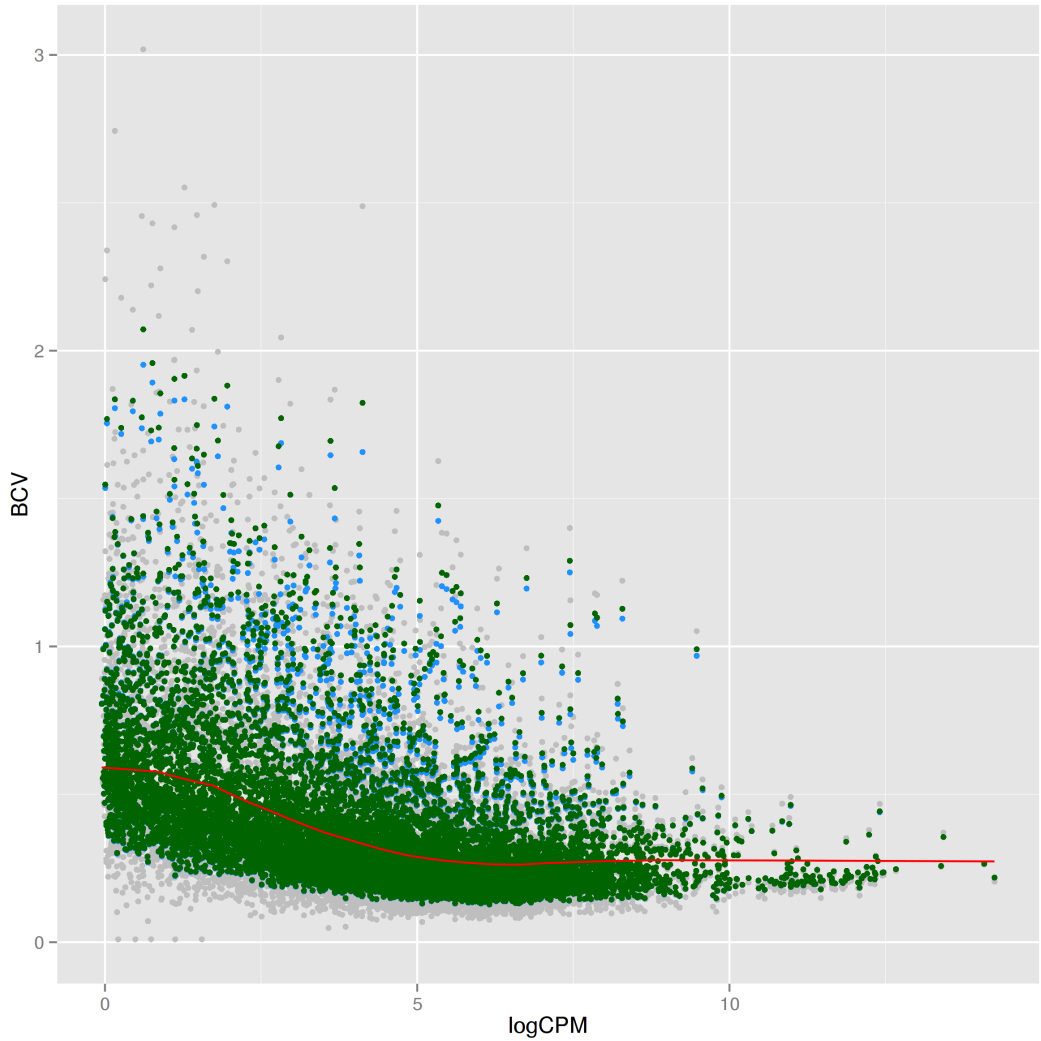


Figure 22: Empirical Bayes robust vs ordinary vs raw dispersions

Problem 3: Your samples/groups vary in quality

- Sometimes one experimental condition is hypervariable (e.g. cancer vs normal?)
- Sometimes one batch came out better or worse than others
- Sometimes samples are degraded to varying and unknown degrees
- In general: your dataset has outlier samples, or heteroskedasticity from sample to sample or group to group

Solution: `arrayWeights` to calculate sample quality weights

- Assumption: residuals should all be i.i.d.
- `arrayWeights` works by detecting violations in this assumption
- If a sample's residuals are consistently larger than average, that sample is considered low quality and downweighted
- Conversely, samples with consistently small residuals are upweighted
- `var.design` argument can be used to constrain groups of samples to have the same weight (this can be different from your main design)
- To use `arrayWeights` on RNA-seq data, use the combined function `voomWithQualityWeights` - 2x alternating iteration of `voom` and `arrayWeights`

arrayWeights example: MDS plot

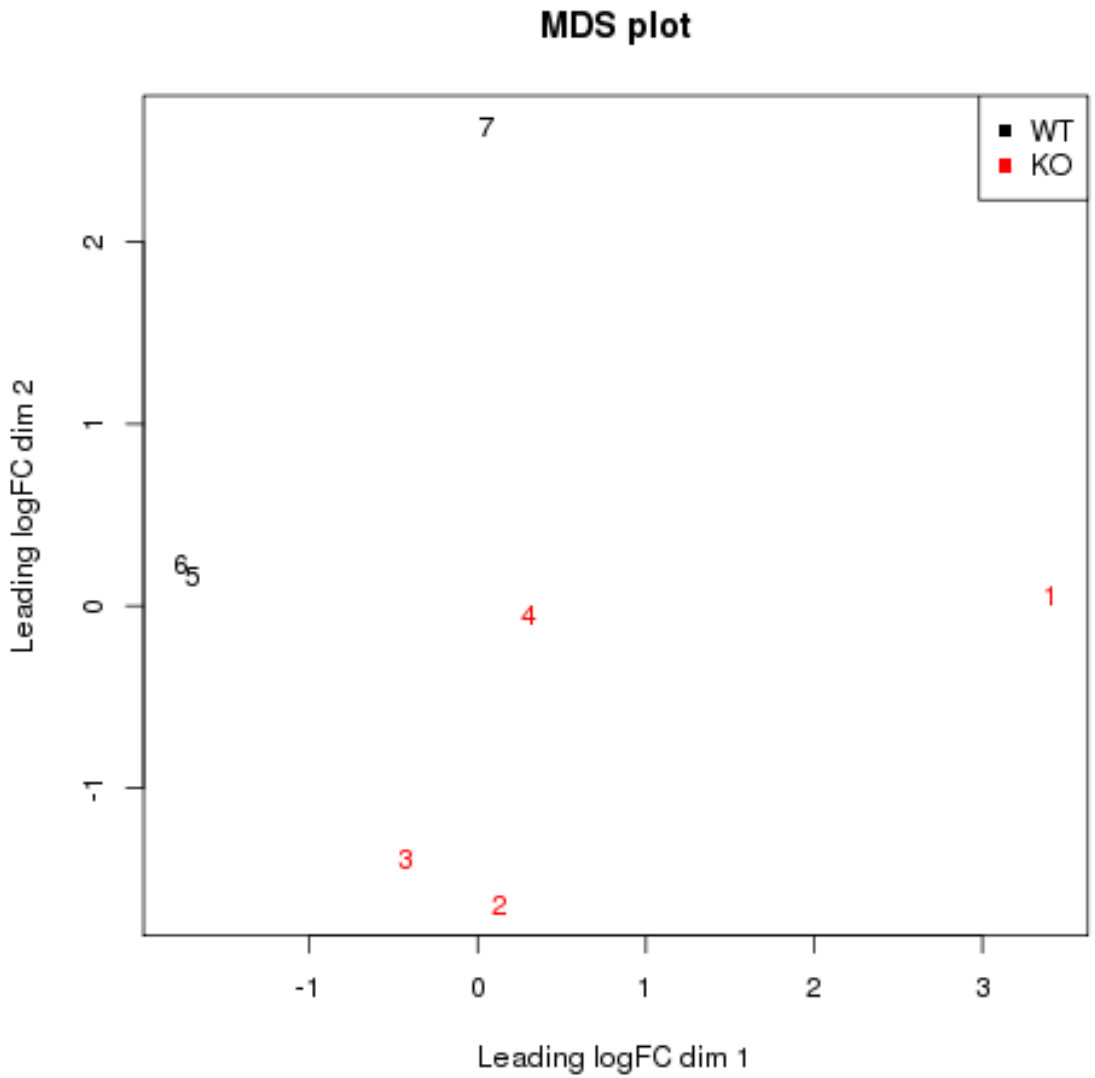


Figure 23: Example MDS plot of data for arrayWeights

arrayWeights example: Resulting weights

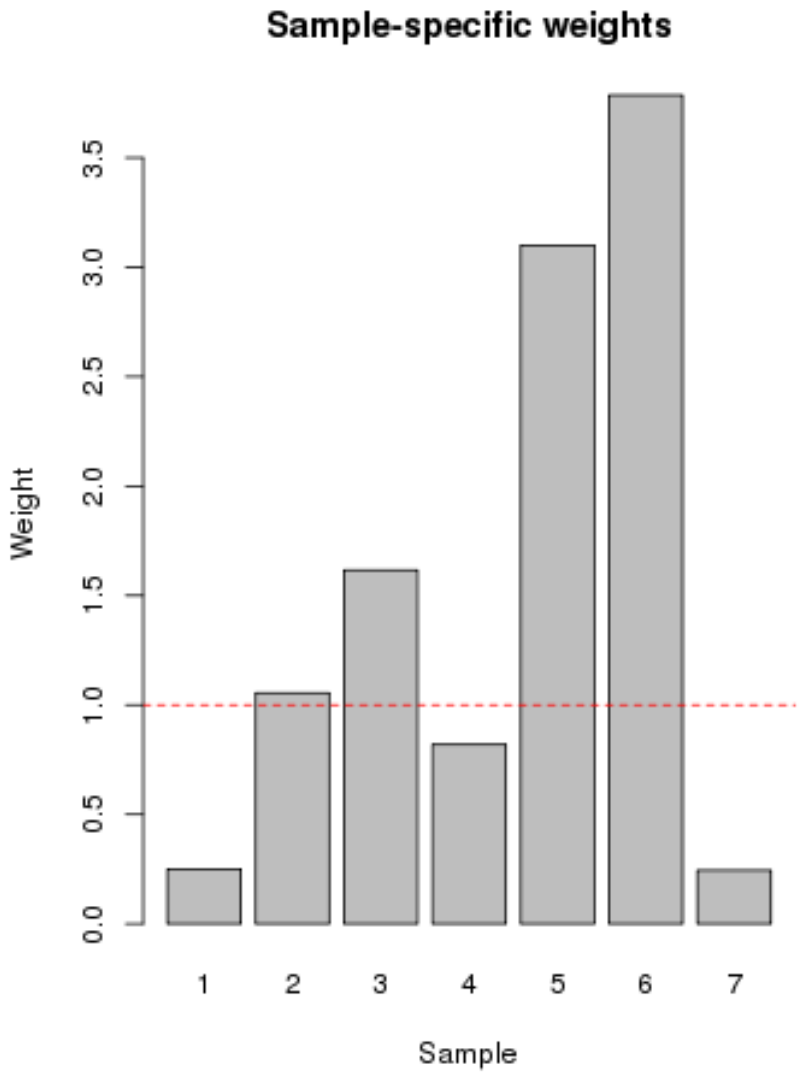


Figure 24: Sample weights determined by arrayWeights

Practical arrayWeights usage

- To check for group/batch heteroskedasticity, run `arrayWeights` without `var.design` and plot the weights vs all your covariates to check for correlations (you can also do ANOVA of weights vs covariates)
- Try plotting weights vs known quality measures (e.g. RIN) to see if they explain quality issues
- For group heteroskedasticity, choose an appropriate `var.design` based on which factors are associated with the weights and then re-run
- For sample heteroskedasticity, no `var.design` needed

Real-world example: Quality differences between cell types

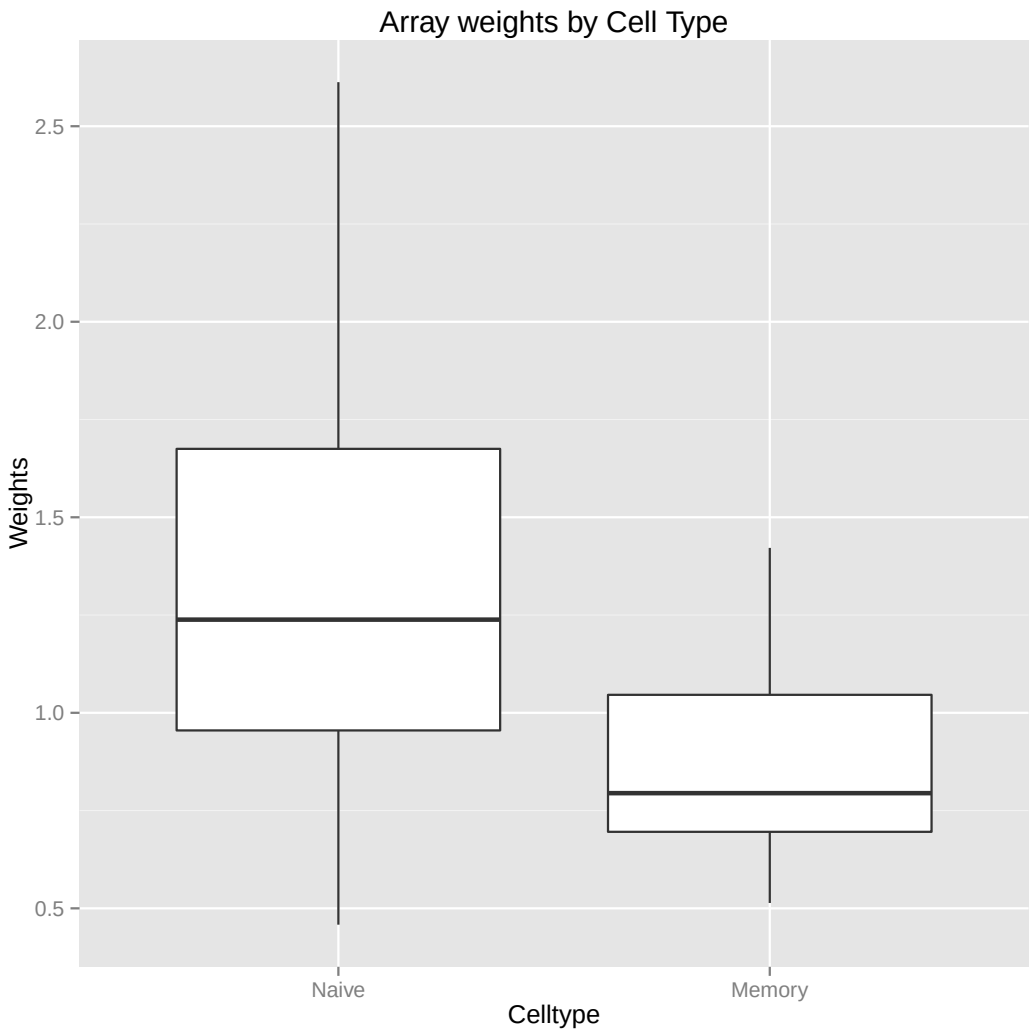


Figure 25:

Real-world example: Quality differences between time points

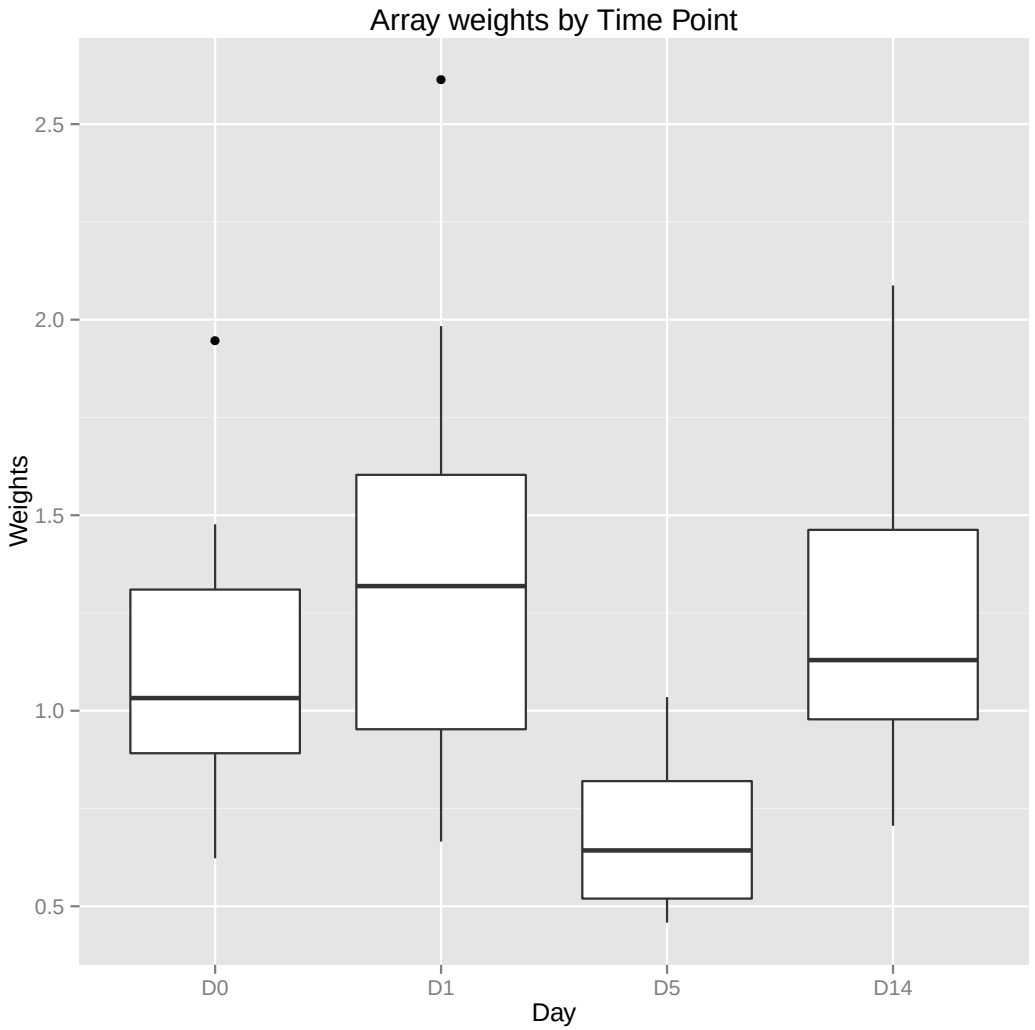


Figure 26:

Real-world example: Quality differences between donors

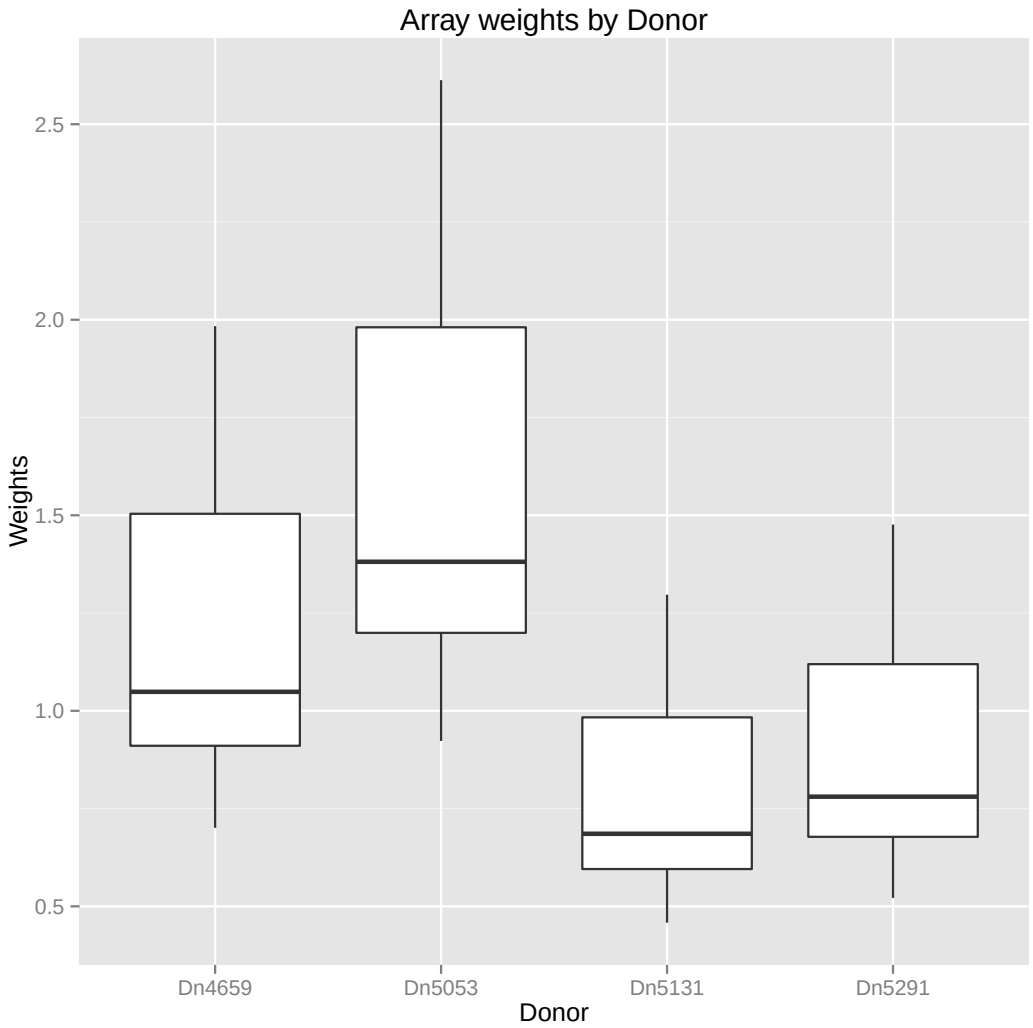


Figure 27:

Real-world example: Quality differences between batches

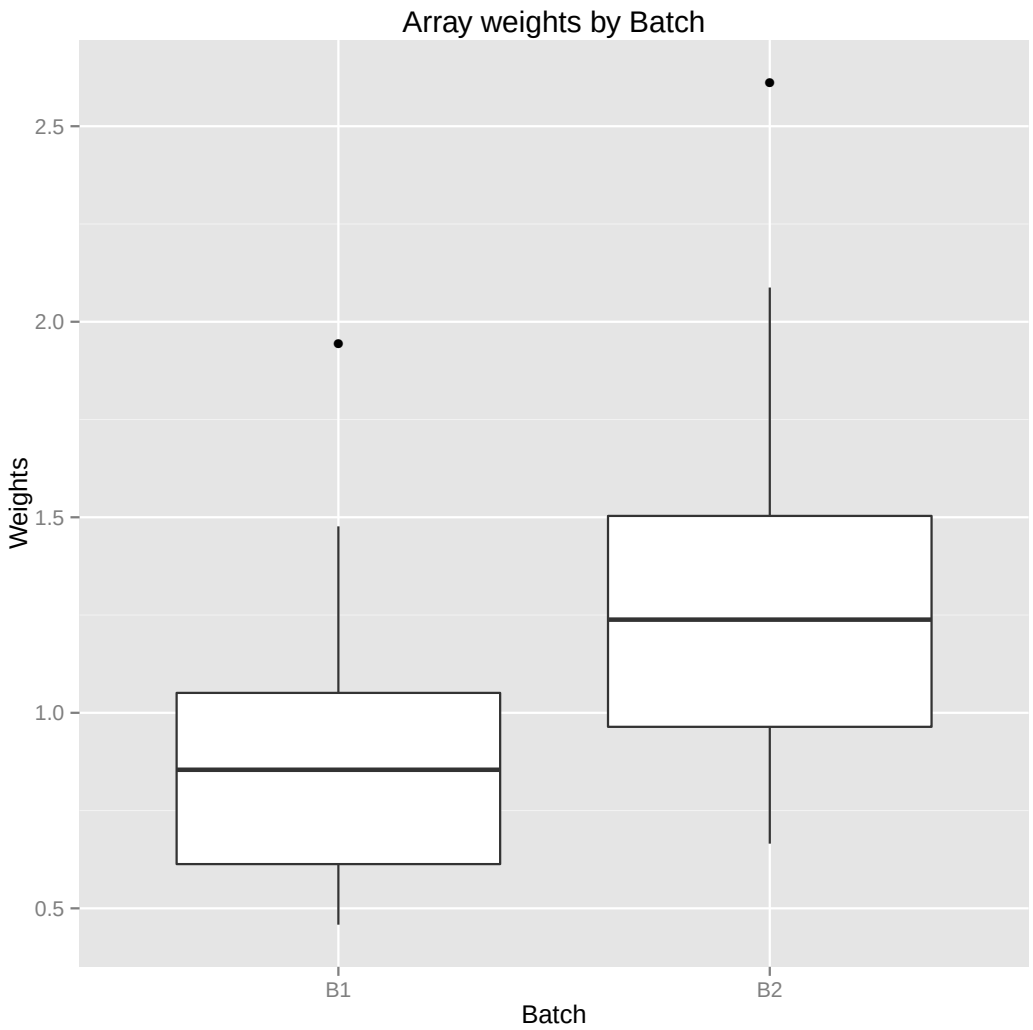


Figure 28:

Problem 4: Experimental design has nested factors

Patient	Treatment	Status
P1	Drug A	Untreated
P1	Drug A	Treated
P2	Drug A	Untreated
P2	Drug A	Treated
P3	Drug A	Untreated
P3	Drug A	Treated
P4	Drug B	Untreated
P4	Drug B	Treated
P5	Drug B	Untreated
P5	Drug B	Treated
P6	Drug B	Untreated
P6	Drug B	Treated

Figure 29: Typical nested-factor design

Problem 4: Experimental design has nested factors

- You want to fit a model with all three factors ($\sim Treatment * Status + Patient$)
- But Patient is nested within Treatment, and you can't fit a model with nested factors
- It would be like fitting a model of $y = a * x + b * x + c$; the a and b are redundant, and could vary infinitely
- Essentially, the patient effects could absorb any difference between treatments
- The same issue can arise in mircoarrays with technical replicates, which are necessarily nested within sample groups
- Including Patient is mathematically unsound, while excluding it is ignoring part of the design. Is there a compromise?

Solution: Treat nested factor as a random effect with `duplicateCorrelation`

- Samples from the same patient are more highly correlated than samples from different patients
- Residuals are not independent
- `duplicateCorrelation` estimates this correlation and adjusts statistical significance appropriately
- It is equivalent to fitting a linear mixed-effects model with Patient as a random effect, and your design matrix as fixed effects
- Mixed models normally require many samples for a robust fit, but `duplicateCorrelation` estimates a single consensus correlation value for all genes, making it robust with few samples
- In general, when you see nested factors, you should consider a mixed model

Problem 5: Experimental effects obscured by unknown confounding factors

- Your experimental effect of interest doesn't show up in the first few dimensions of a PCA
- Something else must be the primary driver of expression variability in your data
- It's possible that the data is simply hypervariable
- But it's more likely that expression of many genes follows some unobserved confounder(s)
- Can we reconstruct these confounders from the data itself?

sva visual example

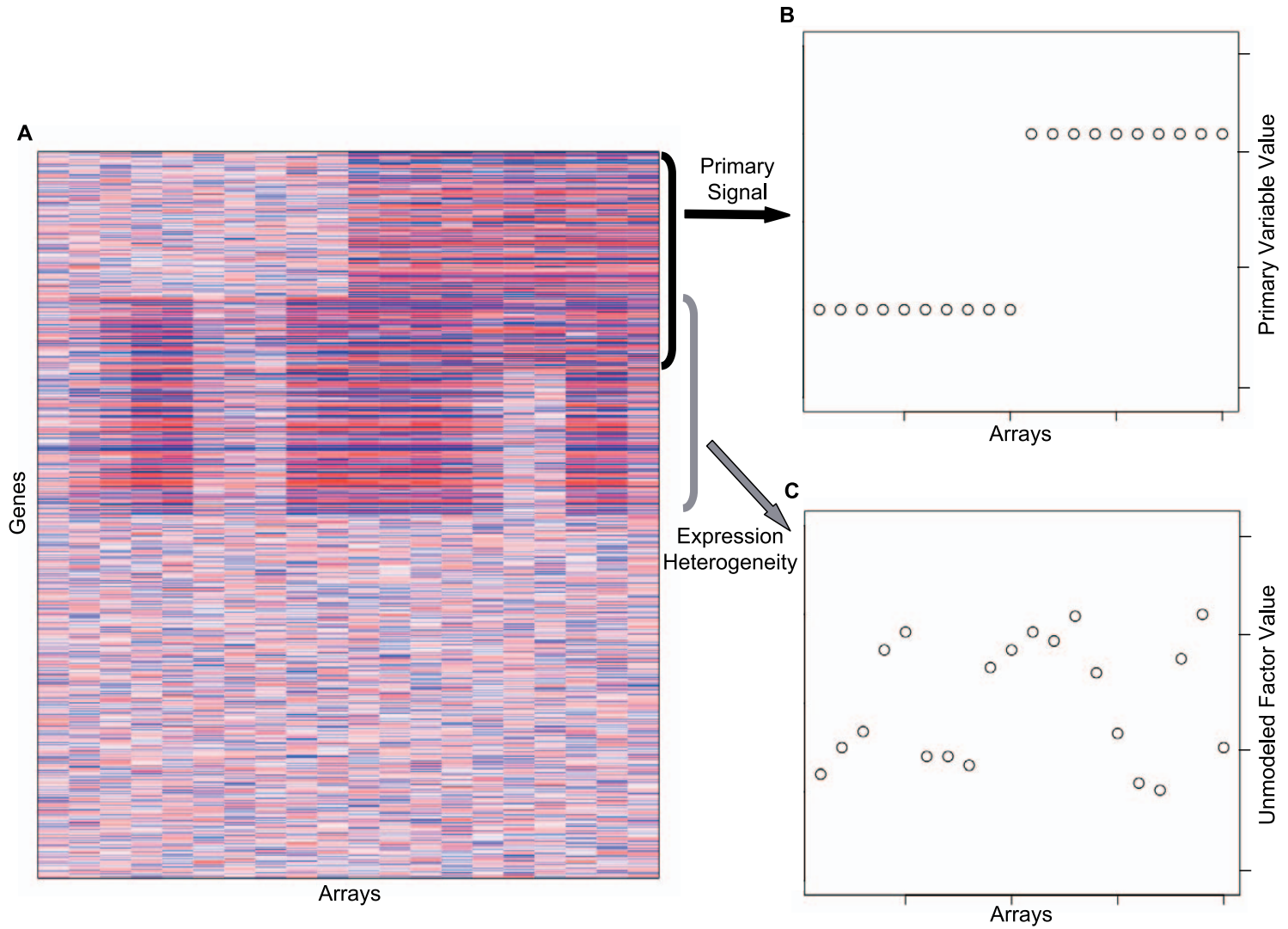
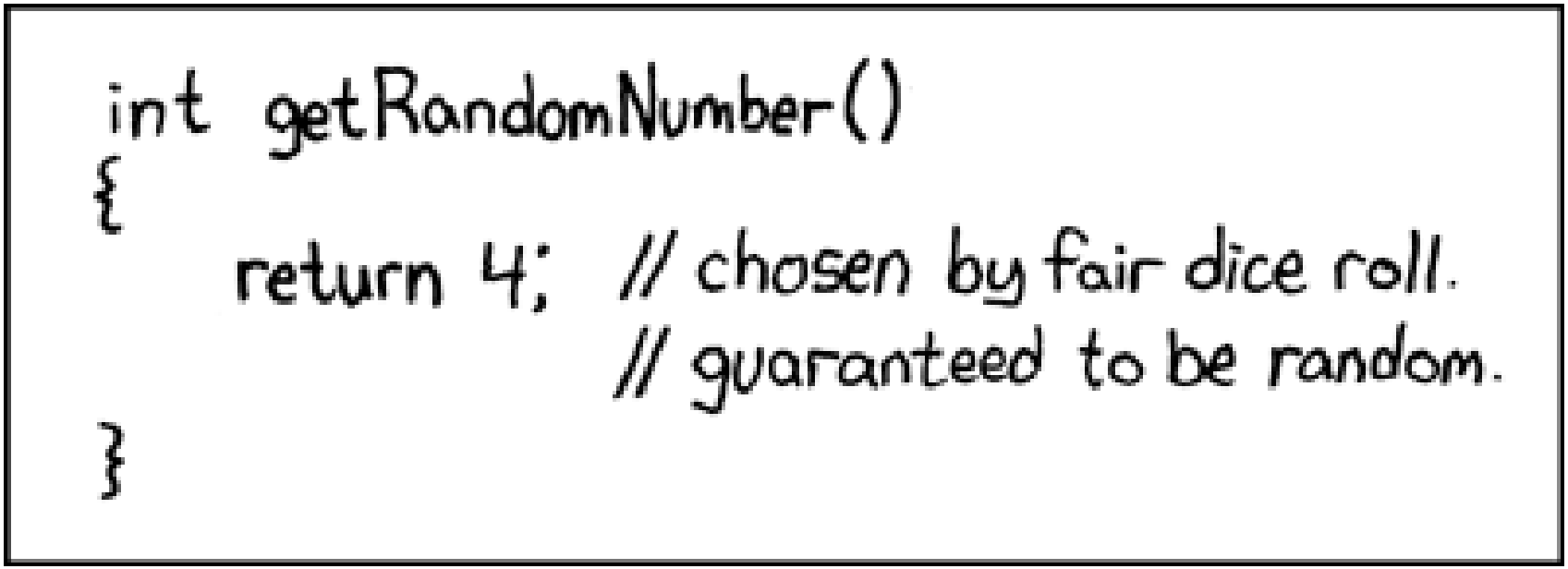


Figure 30: Visual explanation of sva

Sometimes "random" variation isn't so random after all

When many genes show the same "random" variations, maybe something else is going on.



Surrogate variable analysis: exploiting inter-gene residual dependence

- If many genes are affected by the same factors, their residuals should be correlated, violating the independence assumption
- Naive (but incorrect) solution: do PCA on model residuals and add the first few PCs to your model
- Naive idea is overly optimistic, since it doesn't allow for correlation between unobserved factors and known factors
- Better: use residuals to identify genes associated with unobserved factors, but then estimate those factors from the original data (*not* the residuals)
- sva produces a surrogate variable matrix which you `cbind` onto your design
- sva is great for degraded samples or other quality issues
- Other options with similar approaches: RUV, PEER

Problem 6: Non-linear effects of continuous variables

- Gene expression may be related to a continuous covariate in a non-linear fashion
- circadian gene expression as a periodic function of time
- Gene expression after a treatment: activation and return to resting state
- You want to model gene expression as some smooth but non-linear function of time
- You could treat time as a factor, but this allows arbitrary (biologically implausible) differences between time points - no smoothness constraint
- If you have many time points, treating time as a factor will add too many coefficients to the model
- You want a compromise between a single linear coefficient and separate coefficients for each time point
- Time points are not uniform between replicates

Solution: Use a natural spline basis

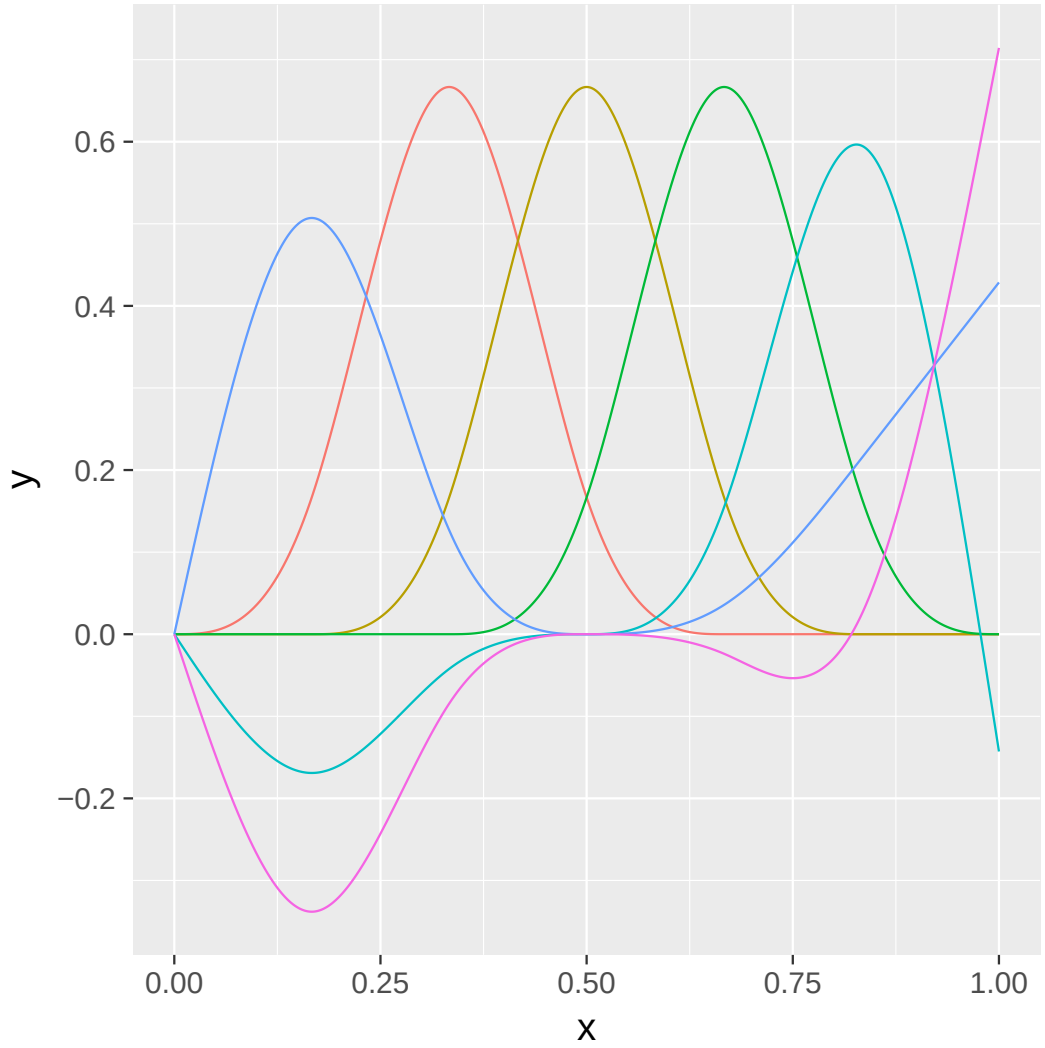


Figure 31: Natural spine basis with 6 df on unit interval

Natural spline basis can encode any smooth curve

- Any sufficiently smooth curve can be approximated as a linear combination of the basis splines
- "Sufficiently smooth" means "no more inflections than there are degrees of freedom in the basis"
- Generally 3 to 5 df is a good choice unless you have very many time points and suspect repeated fluctuations in expression
- An experiment with more than 5 inflections is probably not a well-designed/well-controlled experiment
- You can sometimes identify nonlinear effects from the PCA plot

Nonlinear time effect in post-transplant data PCA

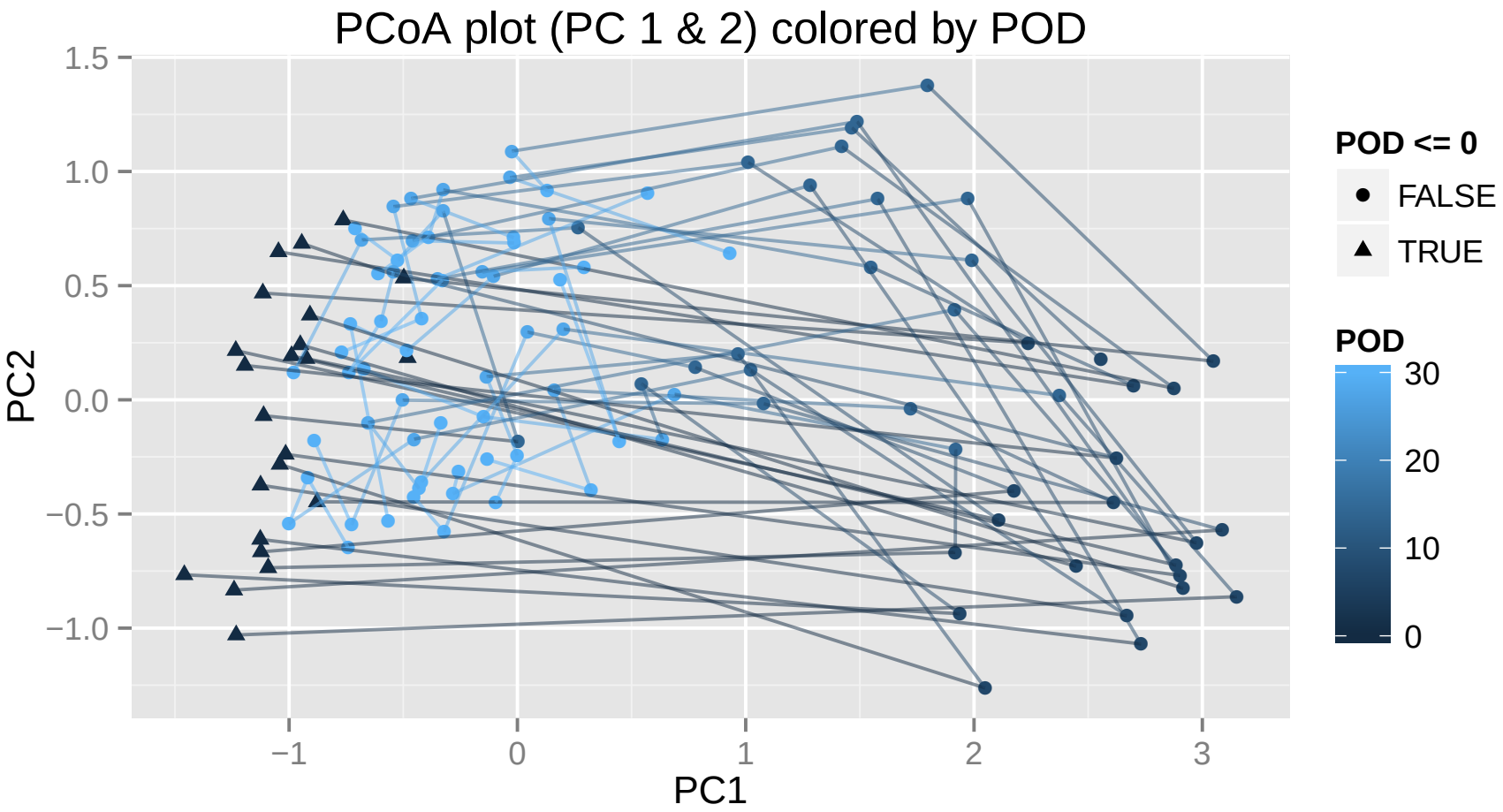


Figure 32: Expression changes non-linearly with time (POD)

Conclusions

- limma is very flexible analysis framework incorporating many different methods for handling data issues
- These methods all take advantage of the fact that you're analyzing many genes to robustly identify and correct violations of model assumptions
- You don't have to memorize all of these methods, but you should be aware that they exist, and be on the lookout for these kinds of issues in your data
- Exploratory data analysis, including looking at PCA plots and p-value histograms, is essential for catching data or model issues and should never be skipped or rushed

Cool New Stuff (which I didn't have time to cover)

- Alignment-free quantification with Salmon & Kallisto
  - Faster and more accurate than alignment + counting
  - Account for multi-mapping, alternative splicing, priming bias, GC, bias, etc.
  - Quantify both transcripts and genes
  - Bootstrap to assess quantification precision
  - Now recommended over classic align+count method by DESeq2 author
- Smooth quantile normalization: Interpolate between full quantile normalization and per-group quantile normalization for best bias-variance trade-off

Any Questions?

Links to more stuff I've done

- My RNA-seq lecture & lesson from the Scripps Bioinformatics course
- Presentation on pathway and gene set testing
- Presentation on reproducible workflows
- Lab meeting slides on ChIP-Seq analysis
- My Github, with some of my bioinformatics code released as reproducible workflows