A set of notes by Charlie **Spring 2017**

Notes on 'Statistics for Genomic Data Science'

Seventh Module of the Coursera Genomic Data Science Specialization from John Hopkins University

Week Zero: Introduction

Statistics is: 'the science of learning generalizable knowledge from data'. Study design, visualization, exploration, preprocessing/normalizing data, inference and then communicating all of these things. When finding statistical software, be sure it's trustworthy! Packages on CRAN have only passed very minimal checks! Bioconductor adds some exclusive checks on the software. Bioconductor forums and StackOverflow are useful community resources for asking questions. Data are values of qualitative of quantitative variables belonging to a set of items e.g. - created from a sequencing machine - one slice of a sequence at a time. The sequencing machine then chemically attaches A,C,G and T of a different colour one slice of a sequence at a time. Then associate a likelihood with each colour being present at each point of the image, and summarize them further into counts or reads. Central dogma of statistics: **Large population of people, and measuring all of them is expensive, so use probability to sample from this big population and make measurements of them - use inference to say something about them**. Don't forget: hats over characters represent estimates! Data points are represented by letters, subscripts for different data points e.g. H_1 represents the height of person 1.

Reproducibility is defined informally as the ability to recompute data analytic results conditional on an observed data set and knowledge of the statistical pipeline used to calculate them. Replicability of a study is the chance that a new experiment targeting the same scientific question will produce a consistent result.

Week One

Reproducibility should be front and centre of all research! It's so critical that all the data and code are available. How do you achieve reproducibility? Data sharing plan: raw data (no preprocessing), tidy data (one variable per column, one observation per column, shareable), codebook (information on variables in tidy data) and recipe used to go between these three things. In general, `R markdown - (.Rmd)` and `IPython Notebooks` are indispensible, especially for embedding 'code chunks'. To compile `.rmd` files, use 'knit' in RStudio, with a frontmatter including title, author, and output format, etc. # primary header, ## secondary, etc. * bulleted lists, 1.,2. numbered lists, just like regular `.md` files:

```
'''{ r chunk1 options}
insert code chunk here
'''
```

where some options are to set the cache, echo, etc. You can insert inline code with something like `r Sys.Date()` which will compile the code into the package directly in-line (as opposed to chunks). The processed data from genomic experiments can be stored in three tidy datasets:

1. The phenotype data: Who are the cases and controls? Information on batches, or other technical or biological information used for modeling.
2. The genomics dataset: tall and skinny - few samples, but a lot of features/genes (e.g. SNPs).
3. The features dataset - a bit about the features which have been measured - e.g. what genome did they come from? What biological pathways do they belong to?

As a check, you want to make sure there are the same number of rows in the phenotype data as there are columns in the genomics data. There should be the same number of rows in the features data as there

are in the genomics data. There should always be 'indicators' for linking. In general, there are three major sources of variation in genomic measurement: a.) phenotypic variability (between cancers and normals) b.) measurement error (random or biased - i.e. batch effects) c.) natural biological variation (any two individuals will have variation due to the fact that theyre different people).

Technical Replicates: Process the sample two separate times on the same biological sample.
Biological Replicates: Different samples, prepared the same way - to measure natural biological variation.

For GWAS - measure lots and lots of people (N between 10,000 to 1,000,000). Sequencing technology does not eliminate biological variability. A low number of observations is bad, because it means studies have low power (higher better!). **What is power? Probability of discovering a real signal if it is there**. Based on: signal size, variability of measurements and N. Typically set to 80%. However, all power calculations require a guess about variability, signal size, etc. We can calculate it with R code: `power.t.test(n=,delta=,sd=)` or `power.t.test(n=,delta=,power=)`. Or, if we know it will go in one direction: `power.t.test(n=,delta=,sd=, alternative='one.sided')`.

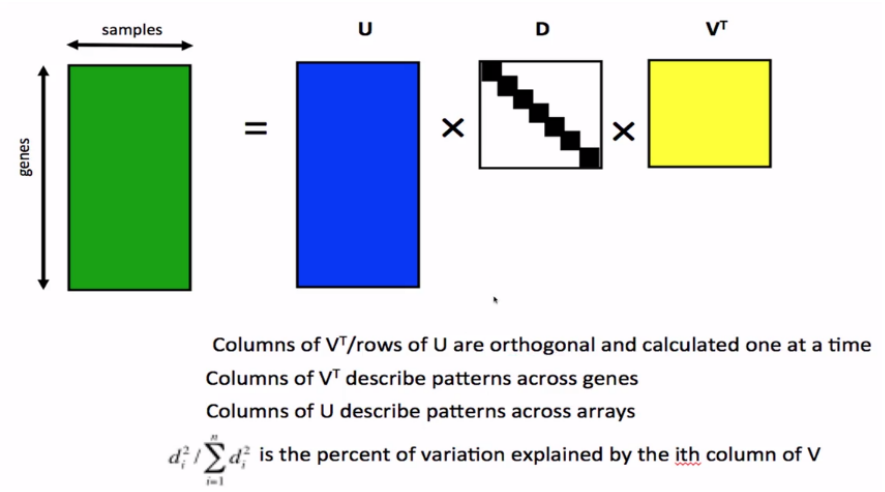
A confounder is another variable which mediates the relationship between two variables. An example is batch effects. With randomization, confounding does not differ among treatments - 'block' them to eliminate bias. Balance the experiment: equal number of treatment and controls (negative and positive), and have different methods of replication. The first thing to do in a study: exploratory analysis - which can also suggest modeling strategies (don't use pie charts!). Plots also help to uncover non-linear patterns. Rediculograms: meaningless albeit visually impressive image of a network. Log-scales can also uncover a large amount of hidden information by making the data more visible. Consider using t-tests of equal or unequal variance. In R we can use commands like `table()` and `summary()` to undertake some exploratory data analysis (don't forget to check for missing values 'useNA'). `dim()` tells us the dimensions of the data - check that they match and then begin to plot. The second thing to explore is done visually: if most of the missing values are near to zero, take log transformations. Boxplots for single values, or boxplots per matrix (gives multiple plots). Density plots allow you to write multiple histograms on top of each other. The third thing to do is: check for consistency (potentially using external data), potentially using ensemble IDs and chromozone information (chromozone labels). Check that the chromozone data matches with the expression data. Filter to just chromozone Y samples: convert dataframe and apply filtering command - to get a smaller dataset which consists of just genes on y-chromozone. Check, for example, that the males have more y-chromozone genes than for females: this is an independent check of consistency! `jitter` adds random noise so that the points dont land all on top of each of other. Heatmaps are a useful multivariate plot, and the `heatmap.2` function adds color key. If you have highly skewed count data, taking logs is essential. Add a value of one to the count to make the log transform defined: this wont affect the big counts, but will make the log transformation defined for the rest of them. A `log2` transform is a different base - allows comparison of two values - difference is equivalent to the log of the full change: an increase of 1 is equal to a doubling of the size. After we filter on the basis of the mean, dimensions should be lower after we remove genes with lower values, and transform plots should be easier to see.

Clustering: can we identify points close to each other and then cluster them together into groups somehow? Two common measures of distance between observations: (i.) Euclidean distance: $\sqrt{(X_1+X_2)^2+(Y_1-Y_2)^2}$ and (ii.) Manhatten distance: $|X_1-X_2|+|Y_1-Y_2|$. How can we use this to cluster points? Start with nearest points: merge them together, as per hierarchical clustering. K-means clustering says: what are the centers of the clusters? Should we start off by guessing centres, then assign all points to the closest centers. If you guess repeatedly, you can iteratively identify cluster centers and then re-identify centers repeatedly. This is a little different to hierarchical clustering, as you have to define the number of clusters that you want. Be careful! The scaling matters a lot, as do outliers and starting values and the number of clusters (not-trivial): better to visualize the data. These clustering methods are widely over-utilized and over-interpreted. `matplot(t(kmeans1$centers),col=1:3,type='l',lwd=3)`.

Week Two

We now move on to preprocessing. With genomic data, you often make a large number of measurements. One important tool is to be able to reduce the dimension. One way to do this is to take the average of the rows or the columns. There are two related problems in general: You have multivariate data (X): find a new set of

Dimension Reduction



multivariate variables that are uncorrelated and explain as much of the variability across rows as possible, and find the best matrix created with fewer variables that explain the original data. This is essentially a singular value decomposition (a factorization of a real or complex matrix).

An example of this is Novembre et al (2013) from the first module. There are also many other decompositions which people use: multidimensional scaling, independent components analysis and non-negative matrix factorization. Practically, we should 'preprocess' by subtracting out all the row-means which are less than a hundred (`edata=edata[rowMeans(edata) > 100,]`), and take logs so the scales are easier to work with: `edata=log2(edata+1)`. We need to 'centre' the data: to remove the (row or column) means otherwise the first singular value or vector will always be the mean level. The SVD matrix has three parts to it: **u**, **v** and **d**: `svd1=svd(edatacentered)`. To plot the percent of variance explained, you need to calculate each singular value squared divided by the sum of singular values squared. Principle components are not quite the same as SVD: subtract column means rather than row means. Outliers can really drive these decompositions - be careful to pick centering and scaling so that all of the different measurements for all the different features are on a common scale.

A preprocessing step might involve adding the reads up, or looking for patterns which arise due to GC content (which might need normalizing). Quantile normalization: order the values (within columns or rows), then average across rows and substitute value with average. Then: re-order averaged values in original order - this forces the distribution to be the same as each other - only useful when you see big bulk differences between these distributions. Be careful! Distributions shouldn't necessarily be the same. Preprocessing: the step where you take the raw data and make a set of data you can do modeling on. Normalizing: make samples have common distribution across samples. You would likely want to do this when the distributions are driven by some technical considerations (i.e. biases?): One example is `normalize.quantiles()` in R - does gene to gene variability still remain after normalization - are they still seperated by study? Both preprocessing and normalization are highly platform and problem dependent. Note!:

'First, researchers starting out in genomics must keep in mind that interesting outliers - that is, results that deviate significantly from the sample - will inevitably contain a plethora of experimental or analytical artefact.'

The linear model: a best line related to two variables - take some genomic measurement (e.g. technical artifact or phenotype and relate it to some outcome). You're trying to minimize the distance between $Y\text{-}b_0\text{-}b_1X_1$. Example: average height of children and average height of parents - explain more variability than all genomic methods . What's e in:

$$C = b_0 + b_1 P + e \tag{7.1}$$

random noise - everything that we didnt measure - sampling variability or bias or variation which we didn't measure. Fit by minimizing distance between Children (C) and line that we care about: $\sum (C - b_0 - b1P)^2$.

Table 7.1: Interpreting Coefficients

Quantity	Log Odds	Odds
Definition	log(p/(1-p))	p/(1-p)
In Logistic Regression:	b	exp(b)
No Effect'	0	1

We can always plot a line, but the line may not always make sense. In genomics, it's common to have a both continuous and categorical covariates e.g. X={0,1,2} - the interpretation is that the difference between 0 and 1 is the same between 1 and 2. Another way is to fit different means: 1's and 0's, where the 0's are subsumed into the constant. Another way is to expand to have 2 dummies equal to 1s: to fit mean levels to different values. The most important thing is to be aware of the interpretation - how many levels do you want to fit? When we have binary or categorical variables, be aware that for observations when the value is equal to zero, this gets subsumed into the constant (e.g. b_0+b_2). Interaction models allow multiple means for each of the different values - these can get tricky when dealing with multiple covariates - what exactly does each β mean? In R: `lm2 = lm(edata[1,]-pdata$gender)` - where under the hood it takes the gender variable into a 1s and 0s dummy variable and edata[1,] is our dependent - this expands naturally to categorical variables with more categories. An interactive term: `lm3=lm(edata[1,]-pdata$age*pdata$gender)`. You can easily overlay these points on a graph - and this can really help detect outliers. Another way we can diagnose is through histograms of the residuals. A unique thing in genomics is that you typically want to run multiple regressions all at once, and this will give us hundreds, thousands or millions of model fits, for which there can be structure in the estimates, structure in the noise, residuals, etc. The key is that we need to think of linear models as one tool which can be applied many times across many different samples. In R, two packages that we will need the most are `lima` and `edge` (which creates an *edge* study object which you can just fit models to directly from passing in a dataset. We can fit on a matrix: `fit=lm.fit(mod,t(edata))`. `lmfit` is much faster (vectorized?): but you'll get basically the same thing when you're running multiple regressions. We can then look at histograms of coefficients, etc.

Batch effects and con-founders: the biggest confounder in genomics is batch effects: how do you adjust for them? Batch: the time at which samples were processed: Control like this: $Y = b_0 + b_1 P + b_2 b + e$ - which is straight ahead to estimate. There are other ways (i.e. empirical Bayes - Johnson and Li), surrogate analysis, etc. To do this in `R` is as simple as: `fit=model.matrix(~as.factor(cancer)+as.factor(batch),data=pheno)`. Some R functions will return you a dataset *cleaned* for batch effects (SVA?) which you can then regress normally.

Week Three

Statistical significance and p-values: the most commonly used statistic in the world. In genomics, corrections for multiple comparisons are essential. A common occurrence is that the outcome is a binary rather than a continuous variable:

$$C_i = b_0 + b_1 + G_i + e_i \tag{7.2}$$

where C and G can take 0 or 1. We can instead formulate it as a probability $pr(C_i = 1) = b + 0 + b_1 + G_i$ - however, this probability is between zero and one. One way is to model the dependent as log(p) - a little better - values between negative infinity and zero - more like a continuous variable. Or: go even further - model the *log odds*: $log(\frac{p}{(1-p)} = b_0 + b_1 + G_i$ but we need to interpret b_i as a change in the log-odds.

A common type of data is count data: a number of reads which overlap a region or varient. There are a number of choices about how you can count each gene with this type of data. The most common type of distribution for modeling this type of data is the Poisson distribution, where it's important to remember that here the mean and the variance are the same: low mean-low variance, etc. : $f(k;\lambda) = \frac{e^{-lambda}\lambda^k}{k!}$. The regression model becomes more complicated here (a type of generalized linear model):

$$g(E[f(c_{i,j})|y_j]) = b_{i,0} + \gamma_i log(q_j) + b_{i,1} y_j \tag{7.3}$$

where the left hand side is the link function conditional (typically logged) on the group indicator as some function of the data (where we are testing b). We can go one step further and use a type of local/smoothed

regression to estimate the relationship between mean and variance, and then plug this into the negative binomial model using a pair of parameters for mean and variance:

$$K_{i,j}\tilde{N}B(\mu_{i,j},\alpha_i) \tag{7.4}$$

$$\mu_{i,j} = s_j q_{i,j} \tag{7.5}$$

$$log_2(q_{i,j}) = x_j\beta_i \tag{7.6}$$

where $K_{i,j}$ are counts of reads for gene i, sample j, $\mu_{i,j}$ is the fitted mean, α_i is gene-specific dispersion, s_j is sample-specific size factor, $q_{i,j}$ is the parametre proportional to the expected true concentration of fragments, x_j is the j-th row of the design matrix X and β_i is the log fold changes for gene i for each column of X. A lot of the work on these topics in R comes in the form of the `glm` command. Don't forget: remember to calculate the PCs first to use for adjustment (a typical confounder is the population structure). The variability is a quality that matters a lot. One way to measure it is to take the standard deviation or variance:

$$S_X^2 = \frac{1}{M-1}\sum_{i=1}^{M}(X_i - \bar{X})^2 \tag{7.7}$$

$$CI = (\bar{X} - c\frac{S_x}{\sqrt{n}}, \bar{X}, c\frac{S_x}{\sqrt{n}} \tag{7.8}$$

Note - as sample size gets bigger, variability gets smaller. The CI is defined by the probability that the real parameter covers the interval: $Pr(\theta \in CI) = f(c)$: you're looking for confidence intervals to give you some idea of some expected range of values that youre somewhat confident will cover the range of values if you repeat the study over and over again. It acts to quantify the uncertainty. The null hypothesis is that the relationship is zero. The alternative hypothesis is that there is a relationship. For example: $y = b_0 + b_1X + e$, and the null is $H_0 : b_1 = 0$ and the alternative of $H_1 : b_1 \neq 0$. In general, it's not possible to accept the null: you're going to reject it 'in favor' of the alternative. Ideally your statistic should be set up to be 'monotone': bigger/smaller is 'less null'. The null depends on the specification, and it must make intuitive sense. Example: is the average height of a child equal to 70? $\bar{X} - 70$ quantifies how far we are. We want to put it on an interpretable scale: $\frac{\bar{X}-70}{\frac{s_x}{\sqrt{n}}}$. This tells us something about how many 'variance units' we see the difference to be. This allows us to make comparisons to standard distributions and calculate probabilities of observing statistics this extreme. Imagine we have two samples:

$$S_X^2 = \frac{1}{M-1}\sum_{i}^{M}(X_i - \bar{X})^2 \tag{7.9}$$

$$S_Y^2 = \frac{1}{M-1}\sum_{i}^{M}(Y_i - \bar{Y})^2 \tag{7.10}$$

To compare whether these two groups are the same, use the t-statistic:

$$t = \frac{\bar{Y} - \bar{X}}{\sqrt{(\frac{S_y^2}{N} + \frac{S_X^2}{M}}} \tag{7.11}$$

When you divide by the estimate of the variability, the results may be tiny, and one way to get around this is to use an empirical Bayes procedure - the simplest one of this is to add a constant to add to the denominator: $\frac{\hat{b}}{s.e.(\hat{b})+c}$. We might want to compare the fit of two or more models: does one fit better than one which plots an overall average? One way we can consider this is through the residuals - the data point subtracting the model fit: $y - b_0 - b_1B$ vs $y - b_0 - b_2B$, for example. One way we can test this is through the commonly used F-statistic:

$$F = \frac{n - p_1}{p_1 - p_0}\frac{RSS_0 - RSS_1}{RSS_1} \tag{7.12}$$

$$RSS_k = \sum_i (y_i - \hat{y}_i^k)^2 \tag{7.13}$$

In R, `limma` is indispensable for calculating statistics such as this (and other empirical Bayes calculations), but again, `edge` can be used more simply if the user has no familiarity with matrix notation, and with `edge` you can use an adjust variable. **Permutation:** for example you might calculate a statistic to compare the difference in expression level between responders and non-responders standardized by some measure of their variability. One thing we can do is randomly scramble the labels. The idea is that you do this permutation and then re-calculate this statistic for the original gene: how extreme is the original (unscrambled) statistic? This concept is *extremely* common in genomics - not just for simple comparisons, but also network comparison, etc. By switching the labels we can still assume that the data come from the exact same distribution: by permuting them, we are making the assumption that the labels don't matter. When doing this computationally, don't forget to set the `seed` vale.

The p-value is so popular that if it were cited every time it were used, it would have 3m citations - and it's consistently mis-interpreted. **It is the probability of observing a statistic that extreme if the null hypothesis is true**. It is not: (i) the probability that the null or (ii) alternative are true, or (iii) a measure of statistical evidence. The p-value can be thought of as the number of permutation statistics that we observe to be larger than the number of permutations that we calculated:

$$P - value = \frac{\#|S^{permutations}| \geq |S^{obs}|}{\#of permutations} \tag{7.14}$$

A particularly important feature of this is the fact that the permutations will be uniformly distributed. The p-values almost always go to zero with sample size. The cutoff of 0.05 is a made up number: these should be reported in conjunction with estimates and variances. The concept of multiple hypothesis testing (MHT) is best emphasized by the 'jelly bean' phenomenon. For example: if you measure 10,000 genes, 500 will be false positive. Family wise error rate: Pr(# False Positives$\geq$1). False discovery rate: E[$\frac{\#FalsePositives}{\#ofDiscoveries}$] - this helps to quantify the noise level in the number of discoveries made. The way to do these corrections:

- **Bonferroni Correction** for the family wise error rate: p-values less than α/m are significant.
- **Benjamin-Hochberg Correction** for the false discovery rate: order the p-values from smallest to largest - p_1,...,p_m: if $p_i < \alpha \times \frac{i}{m}$ then it is significant: this is a linearly growing function in the number of tests performed.

Just using the raw-uncorrected p-values is only done when really analyzing one variable of interest. Type 1 errors are different to the family wise error rate and the false discovery rate. These all rely on the p-values being 'correct': but many things can go wrong - such as batch effects, bad specification, etc.

Week Four

When you get a list of SNPs or genes that you find interesting, how do you then communicate them? What do these genes or SNPs do? Do some genes have something in common with a list of differentially expressed genes? Take statistics for every gene calculated, and order from largest to smallest (or largest to smallest p-value). Then take some gene set that you care about and label them - calculate a running statistic that goes up every time you have a gene in the gene set - are they clustered in one region? This is gene set enrichment - the max deviation from zero - is it more than we would expect to see by chance? Again - permute this - recalculate the statistics and re-order them to get a reference set and again do a false discovery rate type test. This can be hard to interpret - especially if the categories are broad or vague - it is too easy to 'tell stories' if you aren't careful, and be careful to correct for the multiple testing problem. Enrichment analysis can be useful for summarizing a number of things though - not just gene sets - for example - two sets of results for SNPs labeled with one of two different analyses. For example: count eQTLs which are in a particular number of GWAS SNPs, and take another randomly selected bunch of hits to calculate eQTLs - again - a type of permutation scheme: are they independent of each other? Use Fishers exact test or chi-square test (possibly within minor allele frequency or GC content) - take into account common properties. Getting the null right in this case is extremely hard.

The process for RNA-seq: Recall that the central dogma - information passed from DNA to RNA to protein: imagine we have a fragmented RNA molecule using its poly-A tail. You can then reverse transcribe it into complimentary DNA and you can sequence that and turn it into the RNA.

1. Align the reads to the genome or transciptome, assuming it's known. You can align reads to the genome to account for splicing using software such as `HiSat`, `Rail`, `Star`, `Tophat2`, etc.

2. Then you need to count the values that correspond to a particular gene or transcript - use `HTSeq, featureCounts, kallisto (no alignment - quicker), derfinder`. You can also assemble and quantify: `StringTie, Cufflinks (where the reference genome is known), Trinity, RSEM`.

3. Then you need to normalize the data - `EDAseq, cqn, DESeq/edger, Ballgown, derfinder`. To remove batch effects: use `sva or RUVseq`.

4. Then you need to perform statistical tests/modelling - `DESeq2/edgeR, Ballgown, derfinder`.

5. Finally - gene set enrichment analysis to find gene sets or categories which are enriched: `goseq, SeqGSEA`, etc.

The process for CHip-Seq: useful for measuring the way in which proteins (regulate) interact with DNA - through binding of proteins to DNA (transcription factors which regulate the expression of particular genes) - how much are particular proteins attached to DNA - the first step is to crosslink proteins to DNA - then fragment the DNA. If you have an anti-body for the protein to which you have attached to the DNA, you can do an anti-body pulldown - enriches for a particular subset of the DNA associated with the proteins of interest. Sequence just those fragments - how do you analyze the DNA that just comes down from this pulldown experiment.

1. Align: you don't necessarily need to worry about doing anything other than a straight-ahead alignment to the genome, such as with `Bowtie2` or `BWA`.

2. Detect peaks: we've enriched particular sequences and big piles of reads corresponding to sequences : `CisGenome, MACS, PICS`.

3. Count the amount of reads which cover a particular peak - how quantitative is the CHip-Seq technology? It's useful to know how much of the reads fall into each of the peaks with `CisGenome,MACS, diffbind`.

4. A newer step is normalization – especially cross-sample normalization because of the increasing number of replicates over time. Use `diffbind, MAnorm` in Bioconductor.

5. Then you need to do statistical tests to identify differences between the cases you care about - binomial test or others: `CisGenome and MACS` focus on the two-sample case and cover the whole process, `diffbind` cover the whole spectrum.

6. What is the sequence motifs underlying this particular transcription factor? Annotate the transcription factor binding sites - `CisGenome`, `meme-suite`, `BioC Annotation Workflow`

The Process for DNA Methylation: one of the many epigenetic marks measured through NGS or microarrays. DNA methylation refers to a particular methyl group binding to CpG sites in the genome - where can we identify the places where the binding is occurring - one is bi-sulfite conversion followed by sequencing. Another way is through methylation arrays and hybridisation to a microarray to show how much methylation is happening at a specific locus.

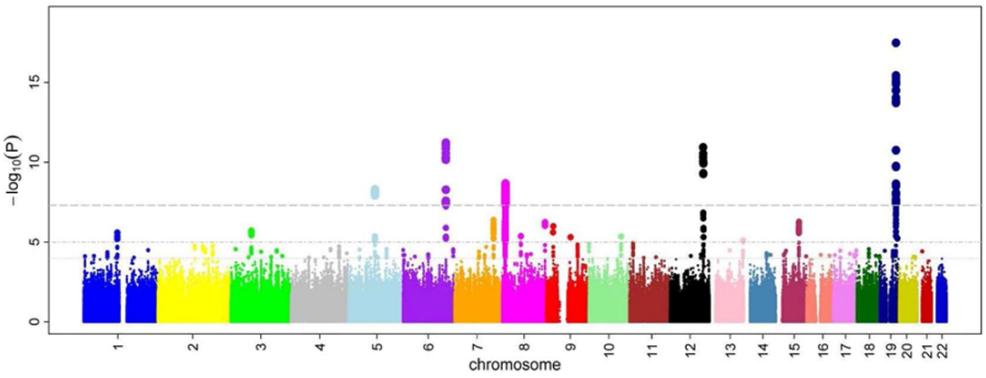
1. Normalization - detect whether there was methylation at that locus, comparing methylated dna to unmethylated dna whether that is through bi-sulfite conversion or hybridisation (`minfi` or `charm`).

2. Smoothing - find clumps of points above a particular level which shows they are highly methylated (`charm` or `bsseq`).

3. Region finding - fit a statistical model to find these regions or tissues (`charm` or `bsseq`).

4. Annotate the regions to the genome: particular categories specific to methylation - regions next to CpG islands, outside regions, etc.

Manhatten Plot Example



The Process for GWAS/WGS: One of the most common applications is WGS (previously called GWAS) - directly measuring variability in the DNA to find different types of varients - insertion or deletion in the genome. Population level inference. One way to do this is direct re-sequencing of the genome. Fragment it then sequence it - look for variations in the genome compared to the standard reference genome and identify if there is any variability. Should there be more than one reference genome? How do we define variability? How much do these variations associate with different outcomes? Once you've got the genome and your fragments, do many fragments have a C vs a G? Is this a heterozygote for a varient which doesnt necessarily feature in the reference sequence. The idea is that you can do the same sort of thing with a microarray - GWAS approach typically applied on microarrays - natural extension. You have fragmented samples and you compare them on homozygous reference alleles, etc, and identify which one it is to genotype samples.

1. Varient identification: compare the levels which you've observed for the different genotypes then make a call for each varient - (SNP-chip - `crlmm`) (sequencing - a little more complicated - `freeBayes, GATK`).

2. Population stratification - if youre looking for association between variants, the most common confounder is population structure. Try: `EIGENSOFT, snpStats`.

3. Once you have these confounders you do a set of statistical tests - SNP by SNP or varient by varient - test for association with outcome - with a logistic regression model potentially adjusted for principle components - with a p-value for every SNP. A typical way to do this is with Manhatten plots where the highest values looks like spikes above a threshold with BF corrections. Software: `PLINK and snpStats`.

4. Then analyze specific variants and particular regions to figure out which one might cause it, such as with `PLINK, Annotating Genomic Variants`.

5. Then a whole bunch of software is required to categorize the variants - `CADD, VariantAnnotation, Annotating Genomic Variants`. Are they in a splice site, etc. Ultimately you need to do downstream experiments to identify functional variation.

eQTL: expression quantitative trait losci - a common integrative analysis - an analysis where you try to identify variation in DNA which correlate with variations in RNA - what is the abundance of different RNA modules in the same samples - a whole class of problems associated with genomic types. Integrate those data together to identify cross-regulation between measurements. Identify relationship between expression levels as well as genotype levels - SNP/marker level associated with each gene in the genome (position of SNP) and information on a particular gene (how much is it expressed? where is it?) An association between all gene expression levels and all SNPs. Important to be wary of batch effects in the data which may or may not be biasing results. cis-eQTL: where SNP position are close to coding region to the gene, and trans-eQTL where the distance is far. Like GWAS, you need to adjust for population, batch effects and sequence artifacts. The basic idea is to do linear regressions relating gene expression information related to genotype information one by one.

There are a large number of steps in these analyses: this leads us to 'researcher degrees of freedom'. These undisclosed flexibilities in data collection and analysis allow us to craft significance from anything.

The question is contingent on what your model is. The whole idea is summarized by the 'garden of forking paths' - by Gelman. The key take home method - have a specific hypothesis, and pre-specify the analysis plan with training/testing sets. Only analyze your data once or report all analyses. Don't add more and more things until you find the significance you want in order to 'run into' extra false positives. Throughout this course we've focused in inference, as opposed to prediction. The central dogma of prediction: use a training set to build a prediction function (e.g. a classifier). Inference and prediction can give hugely different answers - and rightly so. Inferences may or may not be 'predictive'.