Presentation Notes

#research/harvard medical/misc#

Slide 1: Introduction

My project over this summer concerns a phenomenon called Mono Allelic Expression which is essentially when one allele is expressed and the other is "silenced" in autosomal chromosomes.

Slide 2: MAE: an autosomal analog of X inactivation

X Chromosome

You see this allelic imbalance in X chromosomes, where in females an entire X chromosome will be inactivated on a per cell basis. This is how you can get a mosaic pattern of cell expression in tissues. You can see here in this mouse's heart, where different fluorescent indicators are being expressed depending on which X chromosome was silenced.

Mono-Allelic Expression

On the other hand, MAE happens in genes that lie on autosomal chromosomes. Effectively the same thing is happening here: the allele with the red SNP is being expressed while the one with the green SNP is not transcribed. The blue here is DAPI which is a fluorescent stain that binds to A-T in DNA; the green probe binds to the gene of interest (here it is *Death Associated Protein Kinase 1* DAPK1) that has MAE; and the red probe binds to the RNA that is transcribed from that gene. Only one allele is being transcribed.

Slide 3: MAE affects many mammalian genes

MAE is interesting to study bc it happens across many genes in multiple autosomes. This schematic shows the 22 autosomes of the human genome, and by using machine learning techniques, the lab was able to identify ~4000 MAE genes and ~6000 BAE (balletic) genes in many cell types.

Slide 4: Finding perturbations that reactivate silenced alleles

Because MAE affects many mammalian genes, trying to find ways to **reactivate** silenced alleles is very **important**, as it can lead to the discovery of treatments that can reactivate an

allele that should **NOT** have been silenced.

A big hurdle to overcome in this field is the fact that allele expressions are very stably maintained throughout development. Thus, it can be very difficult to shift allele frequency in any direction.

There are efforts being made to understand the mechanisms of these allelic imbalances. Part of that effort involves trying to find perturbations that can actually induce a change in allele expression. The perturbation here, indicated by the blue arrow, can be a drug or shRNA or small molecule that can be added. When the target gene is "hit", the green allele will be reactivated, and allelic expression of this cell would shift.

Slide 5: ScreenSeq

So how do we determine which drugs work and which do not work, how well they work, and how to quantify all this? ScreenSeq is a systematic way of screening each of these perturbations across cells. **In the interest of time**, I can't go into too much detail about the whole process, but essentially what you need to take away from this, is that

1. perturbations are added onto a plate of clonal cells,
2. RNA is expressed after incubating for a specified period of time,
3. UMI's are added via RT to produce cDNA from RNA,
4. PCR is done to amplify various SNP regions of interest,
5. and finally a barcode map, where in each well there's a unique barcode identifier, is added to the plate in order to be able to differentiate the data that comes out of each well.

In the end, you're left with a plate that has information on how much of each allele was transcribed per perturbation. All the material in the wells can be pipetted into one tube with a plate identifier, which will go to the sequencer. Multiple plates can be made (testing for different perturbations, or sampling at different time points, etc.)

Slide 6: What Did I Do Over the Summer?

These are listed in the order that they contribute to the pipeline that I streamlined. I'll be demoing the ridge plot app, I'll show you the other two apps briefly, and explain to you exactly how the pipeline was streamlined.

- Barcode Helper App

- Cell Viability Input Shiny App
- STAR Alignment
- Streamlined the existing pipeline
- Ridge Plot Visualization Shiny App

Slide 7 + 8: Problems with Status Quo

But first, I'll briefly go over the problems with the status quo.

The whole objective of the pipeline is to get from *plate RNAseq data*, to *SNP counts of alleles for specified genes per perturbation*. This is an exercise in data manipulation in itself, because it requires you to aggregate all of this data that I mentioned in the ScreenSeq slide from different sources by hand.

Once you stitch together all this data into one yaml configuration file, an existing pipeline aligns and performs SNP counts for each gene and for each perturbation. Then that counts data is munged more and input into a R script that visualizes ridge plots to create this **large master ridge plot grid for each perturbation.**

Next Slide

Disadvantages are clear - *manual, time consuming, error prone*.

More specifically:

1. Excel sheets/formulas are bad bc too much copy pasting, no standardization of data input
2. There were manipulations of alignment files to make up for Bowtie's insufficiences
3. furthermore, plates were being pooled together and being processed together –> added another level of complexity –> required highly structured directories that were difficult to keep track of
4. Ridge plots were intimidating to analyze

Slide 9: Live Demo: Streamlined YAML Pipeline

One of the first things I did this summer is to streamline this tedious data collection process, so that the only manual work that an experimenter would need to do is define a yaml file for each plate.

Every plate is defined by these parameters that you can set. The yaml file also points to

several of the key files I mentioned before: **perturbation map, barcode map, cell viabilities, genes** we are interested in, and the **fastq** files.

There are some more nitty gritty parameters that you can set here, but I won't go into too much detail. Essentially, all of this data is read and fed into the existing align and SNP counting pipeline. Instead of using Bowtie where we had to manually inject paired ends into the alignment files, I use STAR aligner to include both paired ends. The counts that we get out are reformed by my pipeline and that file can be input into a shiny app to visualize and analyze.

Slide 10: Live Demo: Visualization

Instead of those master ridge plot analyses that we had before, this tool allows you to visualize the data in slices.

Before we get into the live demo, it's helpful to cartoonize what we are looking at in these ridge plots. Here we see a ridge plot for just one readout gene (or the gene that is being affected by the introduction of a perturbation).

- The **mountains** can be thought of as **1/2 of a violin plot**. It is the distribution of all of the other cells that have been sequenced in the plate for this particular readout gene. So if a mountain is shifted mostly to the left or right, it means it has favored the expression of one allele over the other.
- The **x axis** indicates allele bias from 0 to 1.
- If the mountain is in the middle, both alleles are being expressed equally ➜ **biallelic**.
- The **colored bands** here are the cells in the wells that received the treatment. The multiple colors are versions of a treatment (for example, shRNA that targets different parts of the same gene, different dosages of a drug/chemical, etc)

What are we actually looking for? Previously I mentioned that these allele expressions are very difficult to change because they are stable. By introducing these perturbations we are trying to shift these allele biases towards the other direction. As you step through the days (from day 12 ➜ 16), you'll hopefully see an allelic shift being induced for these cells that received the perturbation, away from the normal mono-allelic bias.

Open app

Screen by Perturbation

1. Upload a screen_summary.csv, which is the last output file of the pipeline.

    - contains data on biases that each perturbation induced on multiple readout genes
2. Here you can select several parameters (readout_gene, days, perturbations

Screen by Readout Gene

Just another way to visualize this by readout gene. This is helpful in seeing what effect a screen of perturbations had on one particular readout gene.

Close app

From here, **the next steps** would be picking out which perturbations are promising (or were able to move this stable allele expression) and analyzing them further with a technique like (ddPCR).

Slide 11 + 12: Barcode Helper App

I also developed several supplementary apps for the biologists in wet lab for experimental design and preparation. Barcode map preparation can be a tedious and error filled task that can get confusing.

I created this app as a visual aid for biologists. For the sake of time, I will not be demoing this app, but essentially **how a biologist would use this**, is they would upload a barcode configuration file for the plate they want to prepare. Each well's barcode identifier is made by concatenating two 6 base barcodes: red indicates the forward bc and blue indicates the reverse bc. Pipetting is done by row, and each row in the plate you're preparing is made by combining two barcode rows. If you hover over each row in the bottom plate, two barcode rows will be highlighted with numbers and gradients to show you which wells from the blue and red rows go where in the purple row.

Go to next slide

If you want to use different barcode maps, aka make different combinations of barcodes, you can simply offset the second row by an arbitrary number relative to the first row. The gradients and the numbering should help them figure out which wells in the barcode plate combine into which plate well.

Slide 13 - 15: Cell Viability Input

This is another app that I made to help make the lives of the biologists in the lab a bit easier. Cell viability is a qualitative measure of how many cells are alive after the perturbation has been introduced. Currently the experimenters would have to manually input cell viability data into an excel worksheet while using a microscope to look at each well of a plate. This reduces all of that manual labor into a couple of keystrokes.

The levels of viability you can record range from NC, NC-VL, VL, VL-L, all the way to VG, and each level is associated with keys *1234567890-*. When you press one of these keys, a row is automatically appended to this data table, where each row represents a well in the plate (defined by the row and column, columns).

This app standardizes a previously unstandardized way of inputting cell viability, and hopefully removes the tedium of this task as well. All this can be exported into a tab separated value .tsv file, which is directly used in the streamlined pipeline that I mentioned before.

Slide 16: Github and Conclusion

To recap, a big hurdle to overcome in this field is the fact that allele expressions are very stably maintained throughout development. It is difficult to find perturbations that will shift these stable allele expressions, so it is necessary to have a pipeline that is quick, clean, and easy to use ➜ in order to screen a bunch of perturbations on a bunch of readout genes at the same time.

Streamlining this pipeline and supplementing biologist with these tools will hopefully make that process easier.

For the sake of documentation, I've uploaded all my code up on Github and deployed several of these applications to public servers so that you all can go out and try to break it.