Q1:

The basic algorithmic part of this problem is fairly straightforward: We need to find some matching protein coding sequences to one part of our read and another protein coding sequences to another part of our read. The main difficulties are in the details.

There are many different strategies for matching to multiple proteins. With respect to the kinds of strategies a couple of factors:

1. Are the reads from DNA or RNA? RNA sequencing might be a bit more straightforward because we don’t have to worry about introns and such, but then we do have to worry about genomic splice patterns for alignment.
2. Is there a possibility that the second protein segment could be short and insert itself into the first, such that we could have two or more junctions in the read? One could rule out such things by assumption. That is, since those kinds of events are hard to detect with confidence with only 150bp reads, we could just accept the possibility of false negatives for that class.
3. When two proteins fuse, there are four possible orientations: 5’-A-3’+5’-B-3’; 5’-A-3’+3’-B-5’; 3’-A-5’+5’-B’-3’; and 3’-A-5’+3’-B-5’. Dependent on the kind of algorithm, one would have to take this into account (e.g., if decided on a BWT search).

Most of you did the above in a reasonable way. The most straightforward would be doing local alignments and seeing if the results implied fusion. Some of you did not allow for any possibility of mismatch, which is not reasonable for two reasons. First, there is always a possibility of sequencing error and population variations from the reference genome. Second, given that these are sequencing from cancer cells, they should be full of mutations.

Some of you took A or B protein too literally. Here, A and B could be ANY pair of proteins from the genome. Also, some of you were looking for a SINGLE fusion protein. There could be multiple in a cancer genome.

Some of you very nicely thought about how the fusion should be in coding frame. This is a good idea—although in actual fusion proteins, it could have had a frameshift, which might still have produced some early terminating aberrant protein.

Strangely, most of you did not take into account that one can have spurious alignment of either short sequences or similar sequences. For example, if the implied sequence of protein B is 5bp of the read, that is quite unlikely to be a real fusion. Thus, how much of each sequence alignment is needed to determine fusion status is a parameter and will affect sensitivity and precision.

Lastly, very few mentioned that one way to have confidence in the fusion inference is if multiple reads all pointed to the same junction.

Here are two answers that I gave 100%:

Before describing the algorithm, we must first determine the TYPE of Nextgen sequencing experiment to identify oncogenic fusion proteins, both in terms of computational search space and cost. I propose we should use a cytoplasmic fraction of poly-A selected RNA-seq from the tumor samples, which will have several advantages. While it is certainly possible to identify chromosomal translocations from WGS, since we are interested in identifying fusion PROTEINS, focusing on the expressed RNA only will greatly increase our read coverage and reduce our alignment space compared to sequencing the whole genome. This will miss potential translocation events that do not directly express RNA which could be causative (e.g. disrupting local chromatin structure to influence expression of other genes). An rRNA depleted total RNA prep will waste reads on pre-mRNAs containing introns not yet spliced out. A poly-A library prep is favorable here because it should (in theory) capture fully processed mRNAs that are polyadenylated and thus translatable. While the experiment may be technically challenging to isolate enough RNA for high depth sequencing, using poly-A RNA from a cytoplasmic or even polysomal fractionation would be ideal in reducing false positives. Sequences in these specific pools are what is translatable. Therefore we avoid false negative fusion transcripts that have defective mRNA processing/export and thus never make it to the cytoplasm/polysomes for translation. First we will reduce the read set for fusion protein detection by first doing linear alignment to the standard genome using STAR. If the read is uniquely mappable using the standard parameters, we will not consider it further. Given that a fusion protein will have separate reads that map uniquely to part(A) and part(B) without the fusion junction, we can greatly reduce are future alignment space by building a custom STAR genome containing genomic sequences of the genes identified in the first pass mapping. The basic idea behind the algorithm is to treat the 5’ and 3’ends of each read within the greatly reduced read set separately and attempt to map them to the greatly reduced genomic search space to identify split reads corresponding to fusion genes. We must consider what minimally acceptable amount of bases (X) must span the fusion junction. Requiring too many bases to map across each side of the fusion could lead to false negatives and too few to false positives. This is a tunable parameter, but say X=20, meaning we believe it is a true fusion protein if at least the 20nt start of the read maps to part(A) AND at least the 20 nt end of the read maps to part(B). Since it is known that there is a 3’end decay in read quality so we will trim the reads using standard QC protocols to increase specificity by removing low quality 3’ends. We will use an efficient alignment algorithm like STAR to try and uniquely map the first 20nt and the last 20nt of each read to the reduced genomic space. If both ends at this point uniquely map we will check if they map to the same†gene and discard them (not a fusion, likely most of our reads) or they uniquely map to two different†genes in which case this is a putative fusion read. 20nt may not be enough to uniquely map the read end so we will extend the 5’ and/or 3’ end by a interval (another parameter, say s = 5nt) and re-align one or both of the ends until each 5’ and 3’ hits its first unique mapping position. Assuming a read has 100% quality with no trimming that means at most each read in our reduced set will be mapped (((read\_len/2)-X)/s)\*2 times or (((150/2)-20)/5)\*2=22 times. For splice junction detection a standard in the field is to require at least 8nt to map across a split so even in this extreme case with a step interval of 1nt each time each read would be aligned (((150/2)-8)/1)\*2 = 134 times. In my experience, an average RNA-seq experiment main 60M reads with perhaps at most 10% of those reads (6M) being non-uniquely mapped for the human genomes. In our worst-case scenario of each of those 6M being mapped 134 times this would amount to at worst 804M alignments to be performed, which is completely feasible for an algorithm like STAR. However we are aligning to the reduced genome described above which should improve mapping time and we expect many fewer alignments to be necessary as we should get unique mapping of the 3’end and 5’end of the reads before all 134 alignments are performed. Finally to further reduce false positives further we can apply additional filters. First there should be multiple fusion junction spanning reads with multiple start positions that pass the above thresholds (Reads that have the same start and end positions should be reduced to a single count to account for potential PCR biases). They should also agree on the fusion location. Once identified we can require similar normalized expression of part(A) and part(B) of the fusion protein (assuming it does not cause allelic imbalance in expression) and ensure a valid ORF of the resulting fusion transcript (otherwise it would not be translated).

My first step would be to obtain a human reference genome with which to compare the Nextgen sequence reads. Sequence reads would then be aligned to the reference genome. Due to the nature of fusion proteins, they should have alignment matches to multiple parts of the genome. Therefore, I will keep track of the alignments corresponding to the top two best alignment scores based on the objective function. Reads from highly repetitive sections of the genome could mess up the analysis here, so perhaps any reads with >10 similarly high alignment scores or reads that consist of a motif repeated over and over could be discarded.

Reads that come from unmutated regions of the genome should have a best alignment with a much better score than the second best one (which would be an alignment to some random part of the genome, which is extremely unlikely to have significant matching given the 150-bp length of reads). However, reads that are from fusion proteins should have two top scores that are both somewhat good, corresponding to the original positions in the genome of the two fused proteins. I will define second-best alignment as the best one that is at least 1000 bp from the best alignment, so that I do not pick up alternative alignments to the best-matching gene. Therefore, the next step is to find a threshold score with which to judge second-best alignments, over which we can say that the read likely comes from a fusion protein.

I propose to find this threshold by aligning reads from known fusion proteins and looking at the resulting scores, comparing them with reads from unmutated proteins. This is the crucial step for determining both sensitivity and precision. For example, a fusion-protein read containing 10 bp from one protein and 140 bp from another is not likely to produce a good 2nd-best alignment corresponding to the position of the first protein. That 10 bp segment is likely to show up at an unrelated 3rd location in the genome, and so I might not even be able to identify what that first protein is. Since the human exome is ~3x107 bp long (~1% of genome), a sequence of ~13 bp (with probability 1/6.7x107) is expected to show up only once in the entire exome and so should not be repeated between proteins. Therefore I could align fusion protein reads with at least 13 bp from each protein and pick a threshold value for the 2nd highest alignment score from there.

This 13 bp requirement could be adjusted, of course, depending on the actual alignment results.

The higher the threshold, the higher the false negative rate as the algorithm will favor fused protein reads containing substantial parts from both proteins resulting in two similar alignment scores. Fused protein reads containing only a bit of one protein will be missed. The lower the threshold, though, the higher the false positive rate, as unfused genes with some sequences or motifs conserved throughout the genome could have high 2nd alignment scores too. Finally, the computational complexity of this algorithm will be driven by the alignment method, as it is simple to just keep track of some top alignment scores instead of a single best alignment score. Smith-Waterman has O(nm) where n would be 150 bp and m would be the size of the genome, so my algorithm could be O(nm) as well. This could get very costly depending on how many reads I have, but I am not sure of the complexities of any more efficient alignment methods out there.

Q2: In this question we have two basic facts to guide our hypothesis about candidate proteins. The first is the transcriptomes of the three differently projecting neurons and the second is the assumption that the proteins we are looking for should be a transmembrane protein.

Using the information from the first, we might reasonably think that receptors that guide axon projections would be different in these three different classes of neurons, since they each project to a different area. An immediate idea is to carryout differential gene expression analysis to identify possible genes whose expression is specific to each area. Many of you have detailed how to identify genes from the RNAseq reads and count the gene expression, which is a bonus but I didn’t require that part for correct answer. Some other possible complications to consider: (1) RNA expression might not be indicative of protein differences; (2) Individual genes might not differ between the three types but some combination of patterns might differ; (3) It might not be expression differences but that the same set of proteins might respond differently to different ligands—at least one of you had this interesting idea.

The second part is to identify transmembrane proteins as candidates in conjunction with the above step. Feature space construction is important and many of you suggested biophysics property of amino-acids for hydrophobicity, sequence length, ligand binding domains, etc. Another possibility is inference of multimer formation, which is typical of receptors. The conceptually big problem is whether one should treat this as a supervised learning problem or an unsupervised problem. The more straightforward approach is to treat it as a supervised problem where we train the learning model on other datasets where transmembrane proteins have been annotated. As an unsupervised problem, one could use prior knowledge to develop an expectation scale. For example, transmembrane proteins tend to be longer, so give longer proteins higher value; give a score to presence of hydrophobic domains, etc. Another possibility is to cluster the proteins and see if they tend to form cluster that coincide with the three regions.

These answers were given full credit:

First, I would align our RNA-Seq reads to genes using standard annotations, and filter to protein-coding genes (removing lncRNAs and other non-coding RNAs). Next, I may use a prioritization method to focus our attention to likely candidate protein-coding genes by several criteria. The gene should:

- have a likely transmembrane region (which could be identified by a series of hydrophobic amino acids, or by a hidden Markov model trained on known transmembrane amino acid sequences);

- should be relatively highly expressed; and

- should be differentially expressed between the three regions of the nose.

There are several ways to approach this prioritization. For example, if I have a large database of transmembrane amino acid sequences and high confidence in the predictions of an HMM, I may simply exclude any gene that encodes for a protein that is predicted to have no transmembrane domains. The feature space of such an HMM would be the amino acid sequences from known transmembrane domains, which produces a generative model of transmembrane sequences. If I am less confident in these predictions, the transmembrane domain criterion may be used as a prior to inform subsequent steps of the analysis.

The criterion of high expression can also be approached in several ways. We may expect a protein involved in axonal guidance to be highly expressed in neuronal tissue compared to other tissues, so we might use a database such as GTEx to exclude genes that are not highly expressed in neurons relative to other tissues. We may also use a criterion specific to our data, such as excluding any gene that is below a certain percentile of gene expression across genes.

The criterion of differential expression can be approached through standard differential expression methods such as Limma, edgeR, or DESeq2. The specific approach we use depends on the nature of our biological hypothesis. For example, we may hypothesize that neurons in more distal regions express this unknown protein in higher amounts; we may design a linear model of differential expression based on this assumption.

Overall, as is often the case in science, combining multiple criteria for identifying a novel gene depends on approaches that may end up being rational but ad hoc. Ultimately, this kind of dataset is most useful for hypothesis generation, which we would then validate through an experiment. With high-throughput biological assays, we can rapidly produce large amounts of data, and computational methods can allow us to integrate prior knowledge to make inferences. Ultimately, the use of these large datasets depends on principled and rational use of computational methods, combined with an ability to demonstrate that findings generalize to real biological scenarios.

Proteins involved in the axon guidance of olfactory sensory neurons have shown to be transmembrane proteins that have unique binding sites for the signaling molecules of three different target regions: the proximal, medial, and distal regions. Therefore, I will use a clustering learning method to predict these proteins.

To map the transcripts to genes, I will use Burrows Wheeler Transform to first generate a circular permutation table of the genes, then using the FM index to align those transcripts to genes. After obtaining those full-length transcripts, I will run a clustering learning method.

The feature space is the hydrophobicity of amino acids. From my prior knowledge, transmembrane proteins must contain nonpolar amino acids that enable them to span the entire hydrophobic region of cell membranes. Amino acids at the terminal ends of the sequence, however, are more likely to be charged or polar, including the binding site for the signaling molecule. Therefore, prior to running the algorithm, I will calculate the relative hydrophobicity of the amino acid sequences. Since most transmembrane proteins have alpha helices that are embedded in the cell membranes, I can also develop an algorithm that estimates the presence and number of alpha helices in each amino acid sequence based on the known amino acid sequences of alpha helices. For instance, nonpolar amino acids such as methionine, leucine, and lysine are commonly found in alpha helices.

When deciding on the learning method, I considered that I wanted to accurately sort the coding sequences into two groups: transmembrane proteins or non-transmembrane proteins. In addition, I can generate training and test datasets not only by using some of the given transcripts but also by incorporating some known transmembrane proteins and non-transmembrane proteins with their respective sequences. Therefore, I will implement a support vector machine (SVM). For the training data, I will randomly select 100 transcripts and classify whether those sequences as encode for transmembrane proteins, in addition to taking the sequences of other transmembrane and non-transmembrane proteins. For the test data, I will repeat the same process using a different set of sequences. I will first run the SVM on the training data, then use the test data to confirm the accuracy of the model. In addition, I will use cross validation by randomly splitting the input data, fitting the SVM on the first part of the data, and running the SVM on the second part of the data to calculate a prediction error. This cross validation can be completed many times.

If the SVM produces a low prediction error, I can conclude that the sequences that the model classifies as transmembrane proteins are likely to be the guidance proteins. To further support the predictions made by the SVM, I can develop an agglomerative clustering model that accounts for the three different binding sites of the transmembrane proteins to three different types of signaling molecules – one for each target region. The feature space could include the relative ratios of amino acids at the terminal ends of the sequences, where the binding sides are located. While this approach could help identify the three kinds of proteins, the number of clusters is not known prior to running the model so there’s no guarantee the method will classify the sequences in the same way I’m seeking.

Q3: Here we said all of the pre-processing of ChIP-seq data was done, so we already identified peaks and now we have intervals under the peaks. The main idea is that these intervals contain some motif bound by our TF, with some healthy dose of false positive intervals since ChIP-seq may generate some non-specific reads. Therefore, our goal is to find some motif family that is in a good percentage of the 2000 sequence intervals. I expected you all to know that transcription factor binding sites tends to be small, say 6-20 bp range, also typically without gaps. It may be possible that some of you were not aware of this because fair number of you decided to do standard multiple alignment of the 2000 sequences. While this is a possible approach it will most likely not work since standard affine gap alignment has a hard time finding short stretches of good alignment amongst long background sequences. It should also be noted that even with heuristic approaches like CLUSTAL, multiple alignment of 2000 sequences is difficult—although not impossible. The fall back here is then doing local alignments. The problem is that it is difficult to do local multiple alignments. This is because best aligning local sequences for different subsets of sequences might be quite different.

A better approach is to more directly look for short sequences that are found often in each of the 2000 sequences. One might start with deciding on some range of possible motif lengths, say 6-20bp. Then start hashing each k-mer found for k=6 to 20. After this, we can collect the most frequent k-mers and k-mers that are close (in terms of base pair differences) and are also relatively frequent. This set can be summarized into a position-specific weight matrix (PWM). A better approach might be to concatenate all of the sequences and build a suffix tree. Recall the suffix tree reveals all the information about k-mer occurrence frequencies. We can use the suffix tree to find all k-mers with frequency over some threshold for some minimum k. Then those can be used as a seed to extend the lengths. In sum, some kind of bottom-up k-mer search is likely to work best. At least one of you had the good observation that when estimating the PWM, the reads should be examined to make sure that they are not from the same locus so that we don’t over count the occurrence.

Some of you suggested estimating an HMM. This is, in principle, a possible idea. However, in practice this would be a very difficult to train (estimate). The HMM would have to parse 100-500bp to create a core match set for 6-20bp while treating everything else as a background—note also the location of the match set could be anywhere within 100-500bp.

In the second part, most of you said either see if the PWM from the first step fits the second set of sequences or estimate another PWM and then compare them. But, most of you did not specify how we would determine when the fit was sufficiently different or how the two PWM were sufficiently different from each other. Some of you had good ideas here like computing a probability distance between PWM, compute fit probability against a random sequence, randomize the two datasets and fit PWMs, etc. The following answer was given full credit:

The algorithm to process these 2000 sequences to return a position specific weight matrix for GATA4 motifs in addition to a list of motifs above a threshold that is over-represented in the given set of 2000 sequences. In order to to find enriched motifs the algorithm will need a background set to compare against. By default, it will simply shuffle the 2000 sequences given which will preserve the nucleotide composition of the underlying sequences. Since CpGs can be important in promoter regions we could also consider preserving the dinucleotide composition in our null. Our null set of sequences could also be drawn from random, non-GATA4 bound promoter regions whose genes are expressed in mouse embryonic heart cells. I assume that GATA4 will bind a short, continuous stretch of DNA of 8 bp. For to find the sites most likely bound by GATA4 will use a maximal complexity model where I consider all 4^8 possible (65536) 8-mers. For each 8-mer I will use an efficient string search algorithm, like Boyer-Moore, to search the 2000 GATA4 sequences and the 2000 null sequences for the presence or absence of the 8-mer. Because I am requiring an exact match to the 8-mer, Boyer-Moore could be used in this case. Using an indexing strategy, gapped p-mer approach, or an efficient regular expression strategy could also increase efficiency of this step. As soon as the 8-mer is found the search stops for that sequence and it gets assigned a value of 1. Sequences without the 8-mer are assigned a value of 0. We only score each sequence as zero or one instead of counting the all occurances of the 8-mer which could bias the results in the case of overlapping 8-mers. The algorithm will perform a hypergeometric test to assess the significance of the number of GATA4 peaks that had one or more occurance of the 8-mer in comparison to our null set of shuffled or background sequences. The top most significant 8-mers above a specified threshold significance will be aligned using an efficient multiple sequence alignment algorithm like Clustal Omega and this alignment will be the basis of the PWM for the GATA4 motif. Overall this algorithm is somewhat naive but is a straightforward way to identify enriched motifs under GATA4 binding peaks. If GATA4 binds a motif larger than 8 nucleotides another, non-maximal complexity model may be considered. Also worth noting is the large number of tests performed on all possible 8-mers. To correct for multiple hypothesis testing in this case we may want to repeat the null distribution background shuffling to increase our statistical power since we would have to multiply the hypergeometric p-values by the 65536 8-mer tests for a Bonferroni correction.

This interesting possibility could be tested by running the above described algorithm on this new dataset to get the PWM of GATA4 in kidney cells and return the top 8-mers. With this we can compare the PWMs for GATA4 from the embryonic heart and kidney data using a distance metric to see if they differ from one another. For example, we could use TOMTOM to apply a Euclidean distance metric down the columns of the two PWMs. These scores are summed and compared to an approximation of the null distribution to give a P†-value. This will tell us if the two PWM are significantly different. There are other, less quantitative ways to quickly assess this. Namely, how many of the top 8-mers are shared between the two experiments? We expect the ChIP binding motif to be enriched at the centers of ChIP peaks. We could align the centers of these 2000 and 1200 sequences and plot the frequency of occurrence at each position of the enriched 8-mers or the PWM motif. If the binding sites are different we should not see any clear positional enrichment of say the heart 8-mers over the 1200 kidney sequences when compared to the null background.

Q4: Here the main approach is to use the single cell mutation information to infer the lineage trees. This is a straightforward application of the phylogeny algorithm. A few details that need to be considered are:

1. how to treat the different mutations—many of you differentially weighted substitutions versus indels in computing distances, which is fine. If you propose a ML method, you will need to consider how to treat the probability of indels versus substitutiosn.
2. Whether to estimate a tree of all 300 cells, or each of the 100 separately. This is really dependent on biological assumptions. If we believe each distant metastatic site is colonized by a single clone, then we can estimate three different trees and then try to paste them together.
3. How to use the control normal tissue data. This should be used to root the tree.

Some of you tried to use the RNA data as expression data rather than mutation inference data. This is a possible route but for cancer clones with lots of mutations, it is hard to model the expression data in terms of lineage-dependent changes.

For the second part of the question, I wanted to see a discussion of using the independent metastatic sites to identify common mutations to both while different from the primary tumor. Ideally, one should use the inferred tree and look for mutations that mark the transition to the metastatic lineages—shared between the two different sites.

The following answers were given full credit:

Q4.1. I would use UPGMA to construct a cell lineage tree of the 300 cell sequence data obtained from the tumors. I would create pairwise distance matrix that compares each of these sequences with the healthy tissue sequence. I assume that indel mutations are more harmful and therefore more likely to cause metastasis because they can cause frameshifts which can dramatically alter the protein that is translated, while base-substitutions will likely be less harmful and cause relatively minor changes to the altered protein. Of course, base-substitutions can still cause significant change such as by altering stop and start codons, but I assume this happens infrequently. Because of this, when calculating the pairwise distances, I will make the indel mutations have a larger weight than base-substitutions. For example, a single base-pair substitution may result in a score of 1, while a single indel mutation may result in a score of 2. The distances will then be a sum of all base-substitution and indel penalties based on these scores. UPGMA typically has a computational complexity of O(t3), which may not be a huge problem given that our dataset only contains 300 leaves. However, we could construct the tree in O(t2) time if we don’t have the computational resources to handle 3003 time complexity. One big problem with this method, however, is that it assumes a constant rate of mutation, which may not be true with cancerous cells – the accumulation of some mutations may increase the rate at which a cell accumulates more mutations. Thus, it may be better to use a probabilistic model where we use maximum parsimony. Again, we make indel mutations have a higher cost than base-substitutions. However, this problem is often NP-hard.

Q4.2. I assume that it is the specific gene that is being mutated that is contributing to metastasis rather than the specifics of the mutation itself – such as if it is an indel or substitution – because it is more likely that the metastasis is caused by altered gene expression or protein activity rather than due to a single amino acid change. First, I would perform multiple sequence alignment and compare the mutations between the two lymph node sequence data sets to find common mutations between the two as well as regions in which both sets have high amounts of mutations, and then see if these mutations can also be found in the lung tumor sequence data. I would compare lymph node sequence data sets first before comparing to the lung tumor sequence data to get a more accurate sense of which mutations are driving metastasis – I don’t want to focus on similarities between the lung tumor sequence data if it’s not found in both lymph node data sets. To identify which regions are significant, I would break up the sequences into smaller segments (say with length n) and measure the amount of mutations in each segment. I would then use ANOVA on the number of mutations over n to test the significance between regions. To make sure that I’m not overlooking clusters of mutations that are separated due to the length of the regions I’ve chosen, I would repeat this with a different length of segments, such as 2n and 3n. I would then compare the different significances of the groups to identify regions that globally contain high amounts of mutations. An alternative method would be to classify DNA segments as either CpG islands or non-CpG islands, and use a hidden Markov model to identify these regions, although this may result in larger regions and be less informative. Each region would then get its own identifier, and I would count the number of common mutations found in each region and perform an ANOVA test to find regions with a significant number of shared mutations. After finding these regions, I would align them to a database such as with BLAST to see which genes are being mutated. I would then be left with knowledge of which regions in which genes are likely to be responsible for metastasis if mutated.

Q5: This question required the most creative insight since I am asking for a new way to do dimension reduction and clustering. The main thing I wanted to see is whether you’ve thought about why the PCA,tSNE, etc. might not have worked and found some creative new way to thinking about dimension reduction and gave a good rationale for the approach. For example, one could imagine different cell types have their own particularly appropriate dimension reduction—this is difficult to solve because we don’t know the cell types beforehand but one could imagine constructing some iterative process. I also allowed for creative new application of existing method, if it wasn’t something well-known as in the first example answer below. The second example answer below has some fuzzy details but it was really good answer in terms of creative approach to the data. Both answers received full points.

I propose a dimensionality reduction technique based on Latent Dirichlet Allocation (LDA) [1,2] and knowledge transfer from a compendium of previous single-cell RNA-Seq datasets. This approach borrows from highly successful methods used in text mining. The rationale for this approach is that RNASeq data, like text, has many properties that makes it natural to share statistical methods:

- RNA-Seq reads are count data, similar to word frequencies in documents;

- scRNA-Seq data is often sparse, similar to text data where even the most common ~tens of thousands of words show up zero times in most documents

- there is an increasingly large number of unlabeled datasets in scRNA-Seq, similar to text.

Briefly, Latent Dirichlet Allocation is a probabilistic graphical model that assumes that each document is produced by a mixture of “topics”, and each “topic” has a generating distribution for each word P(word=wI | topic=zj). Each document is modelled as a bag of words in which the count of each word is generated by a multinomial distribution multiplied by these conditional distributions:

**𝑃(𝑤𝑜𝑟𝑑 = 𝑤1)𝑃(𝑤𝑜𝑟𝑑 = 𝑤1|𝑡𝑜𝑝𝑖𝑐𝑠 = 𝑧1)**

A Dirichlet prior for the topic distribution, and parameter estimation is performed through EM in which the topics are treated as latent variables. I propose that we collect a compendium of publicly available scRNA-Seq data. We perform LDA, using single cell transcriptomes as “documents”, read counts per gene as “words”, and extract “topics” from this dataset. The “topics” are inferred entirely from the data, and may represent biological hidden variables that produce the transcriptional profile of each cell. Given a number of “topics” less than the number of genes, this approach acts as a dimensionality reduction technique that can then be applied to new data (e.g. our scRNA kidney data). Each cell can be represented as a mixture of topics; we can construct a matrix where each row is a cell, and each column is the extent to which each cell is assigned to each topic. This matrix can be clustered through conventional methods (k-means hierarchical clustering, etc). In practice, it can be helpful to reduce dimensionality further, e.g. with PCA, as the LDA topics may be correlated.

This approach takes a fundamentally different model of the data from t-SNE, and comparisons to t-SNE would depend largely on empirical observations of stability, biological interpretability, extensibility to new datasets, etc. LDA is conceptually more similar to PCA or autoencoders, in that it applies a latent variable model of the data to reduce dimensions. Fehrmann et al [3] performed a conceptually similar approach using PCA, in which the authors collected a compendium of microarray datasets and performed PCA and attempted to interpret each principal component. While the statistical assumptions are very different than PCA and other methods (I would argue that the statistical assumptions of LDA are nicer!), this approach would require extensive benchmarking and comparison to PCA through multiple metrics to assess whether the empirical results on real datasets are superior.

In our algorithm, we basically want to ignore the variation in the single cell data that is due to gene expression common across all cells and only look at kidney-speci\_c gene expression in order to cluster the cells. For the \_rst step of the algorithm, we would go onto a website like the UCSC to \_nd a list of "housekeeping" genes that are similar in expression across all cells and don't tell us much about di\_erent cells, and then we can eliminate these gene expression measurements in the single cell data. Then, we would like to go on another website like ENCODE and download free single cell data for other cell types. We would like to aggregate this data to get a pool of cells representative of the whole body, and when we combine the data sets we need to make sure that we normalize for the total read depth per experiment. We will then perform PCA on this data set to get what the "normal" directions of variation are across all cells. We then take these PCA vectors and do a change of basis to the new PCA vectors. Then we select the top variance explaining PCA vectors (which explained the most variance in the pooled data) and we actually eliminate those coordinates from the kidney data. We then change the kidney data back to the original coordinates, and then we perform PCA and only keep the coordinates of the top principal components. The reason that we did all of this was that the PCA vectors from the pooled data should tell us of variation that is normal across all cells and is not cell line speci\_c, and we then want to eliminate this variation from the kidney data in order to only see variation due to kidney-speci\_c cells. We then only consider the top principal components of the modi\_ed data since those are variation directions of the kidney data, and we have reduced the dimension to only the top PC's. Finally, we perform the clustering. We are going to create an aided clustering algorithm that not only takes into account the coordinates of the top principal components, but also the expression levels of the marker genes (we can \_nd these marker genes as we knew what cell clusters we were looking for and should be able to \_nd 6-12 groups (for 6-12 cell types) of marker genes for them online, possibly at UCSC or ENCODE). We are going to use this marker genes to identify the original clusters. For each marker gene in a group, we calculate a distribution over all cells (in the original data, after cell-size algorithm), and for each cell we calculate the percentile in the joint distribution that each expression value for each marker gene is for the cell. For each cell then, we look at whether the marker gene was expected to be upregulated or downregulated in the cell type, and we calculate for each group of marker genes the p norm of the di\_erence of the vector where each entry is the percentile for the marker gene in the group, and the other vector where each entry is 0 if the marker gene was supposed to be downregulated and 1 if the marker gene was supposed to be upregulated. This gives us a value for each group for each cell of how close the expression levels for each group of marker genes are to the "de\_ning" levels of those marker genes, whether they are upregulated or downregulated. We then rank these values for each cell across all cells, and for each cell we put it in the cluster where its ranking for its group of marker gene distance is the highest (as this is the best group to de\_ne the cell). We then perform k-means clustering on the dimension reduced data given these initial clusters. My algorithm compared to t-SNE is better because it takes into account biological data about marker genes, housekeeping genes, and other single-cell data that t-SNE does not use. t-SNE also reduces the data to very few dimensions (2 or 3), while my algorithm performs clustering in higher dimensions to retain more kidney-speci\_c information. However, my method is experimental and may take longer than t-SNE because of the additional steps. Also, t-SNE is good at de\_ning the distances between cells in a manner that diminishes outliers, but my algorithm uses euclidian distance in the k-means clustering between high dimensional data, so outliers stay (which may be good for rare cell types).

BONUS: Here the proposal function must propose some new tree and branching parameters over the space of trees and non-negative branch length (rate). The following answer was excellent.

In a Bayesian estimate of phylogeny, we hope to model possible ancestors and tree structures when given some sample of children (leaves). In doing so we need methods to compute the maximum likelihood of certain parameters, for instance nucleotide substitution rates, and indel mutation rates. For these we can use MCMC to model the posterior probability of these substitution rates as to develop a more accurate Jukes Cantor transition matrix. My suggestion would be to perform MCMC for each substitution rate, using a normal distribution as the proposal function. We can begin by centering our normal distribution using rates determined in literature.

If we seek to predict tree structures as a parameter for our maximum likelihood phylogeny we could additionally use maximum likelihood. First we developed a search tree for binary tree graphs which would be a graph of tree graphs, the neighbors of each node would be any other tree structure by invertin one pair of neighboring leaves or nodes. Every graph will be initialized with uniform probability of occurrence.

During our MCMC our proposal function with work as follows: For a given tree structure as a parameter x we would propose a tree structure y where the probability of proposal is determined by the weighted probability of each neighboring tree structure. We can update our proposed model after a particular burn in period to account for the increased likelihood of certain models