Mini-Project 1

Josh Popp

March 2020

1 Introductions

Genetic effects on gene expression across human tissues (Specialized)

Expression quantitative trait loci (eQTLs) have been extensively studied as candidates for the explanation of the phenotypic consequences of genetic variation [1, 2]. A significant challenge for eQTL analyses is the limitation of eQTL effects to certain biological contexts such as specific tissues [2]. Furthermore, small sample sizes and uncertainty resulting from genome imputation can render these analyses underpowered [3]. In this study we performed the deepest survey of individual- and tissue-specific gene expression to date by collecting genotype, gene expression, histological and clinical data for 449 human donors across 44 (42 distinct) tissues. This enabled us to identify *cis*-eQTL's for more than half of the known lincRNA genes, and more than 85% of protein-coding genes. We also identified tissue-specific *trans*-eQTLs in several tissues, and compared the functional characteristics and genomic contexts of both eQTL types.

Common SNPs explain a large proportion of the heritability for human height (General)

Understanding the effects of genetic variation on phenotype is essential for an effective understanding and treatment of human health and disease. Toward this goal, family studies have sought to disentangle the effects of genetics from those of the environment in order to estimate what proportion of variance in quantitative traits such as height can be explained by additive genetic differences [4]. Genome-wide association studies have taken this exploration a step further, associating phenotypic variance with specific changes in the genome by focusing on single-nucleotide polymorphisms (SNPs), variations at a single position in the genome which are common in the human population. A surprising finding of these genome-wide association studies has been that for any trait, these SNPs are only able to account for a very small portion of the heritability determined by family studies, leading to a 'missing heritability' problem in genomics [5]. Here we explore two possible explanations for this missing heritability. (1) Instead of a few significant causal SNPs, a large set of causal SNPs with small effect sizes may be responsible for phenotypic variance. Consequently, the significance thresholds that are applied to decide which SNPs to consider inadvertently filter out some causal signal. (2) Current technologies only allow for a subset of SNPs to be genotyped. If these SNPs are not the causal SNPs (or are not perfectly correlated with causal SNPs through linkage disequilibrium), further signal will be lost.

2 Review

Deep generative modeling for single-cell transcriptomics

Summary

In this paper, Lopez et al. present scVI, an all-in-one probabilistic method for the normalization and analysis of scRNA-seq data based on a variational autoencoder. scVI models observed expression levels x_{ng} as samples drawn from a zero-inflated negative binomial distribution (ZINB), conditioned on:

1. an observed batch variable s_n
2. an unobserved library size variable ℓ_n which captures variation in capture efficiency and sequencing depth, and
3. a low-dimensional representation of the individual, z_n

The low-dimensional embedding and library size variable are predicted by encoder neural networks, and a decoder maps this learned representation to the parameters of a zero-inflated negative binomial distribution. Intermediate values provide a batch-corrected, normalized estimate for percent abundance of a transcript in a cell, and this can be scaled by the library size variable for imputation. Approximate inference of the posterior distribution of the latent variables is done using variational inference and stochastic optimization. Differential expression analysis is additionally implemented in a robust and efficient manner which takes advantage of the explicit modeling of s_n to reduce the impact of batch effects.

Requested Clarifications

Further explanation of the entropy measurement used to study batch separation in clustering would be appreciated, and would be a welcome quantitative addition to the results of the comparison to PCA/ MNN clustering on the RETINA dataset.

Requested Extensions

Is this method currently adaptable to account for other sources of noise in single-cell sequencing data? While library size appears to be appropriately addressed in this report, a demonstration of the the effects of other nuisance factors in single-cell data (such as undetected doublets or multiplets, low-quality cells, and cell leakage) is absent, even though these factors would play an important role in an all-in-one scRNA-Seq pipeline. While it may be intractable to explicitly model these factors, a display that the method is robust to these factors or a mention of recommended methods for integration of scVI with other pre-processing methods would be beneficial.

The benchmarking of scVI compared to other methods such as ZIFA did not seem to identify whether the observed improvements in performance occurred as a result of the use of non-linear embeddings or the stochastic optimization procedure that was outlined here. A strong extension to this paper would involve the extension of stochastic optimization to some of the alternative methods benchmarked here in order to separate the improvement due to these two factors.

Impact

The framework outlined here is a highly efficient and robust method for the cohesive analysis of single-cell RNA sequencing data, from normalization and imputation to dimensionality reduction and differential expression measurement. The stochastic optimization procedure enables efficient and scalable learning of a biologically meaningful latent space, as is strikingly demonstrated in figure 2. Explicit modeling of two key nuisance factors improves performance and may facilitate downstream analysis.

3 Data Analysis

Q2: Principal Components Analysis

The scree plot shown in figure 1 shows that the first 4 expression PCs capture nearly half of the variance in the data, and after this point each additional PC is, individually, responsible for no more than 3% of the variance.

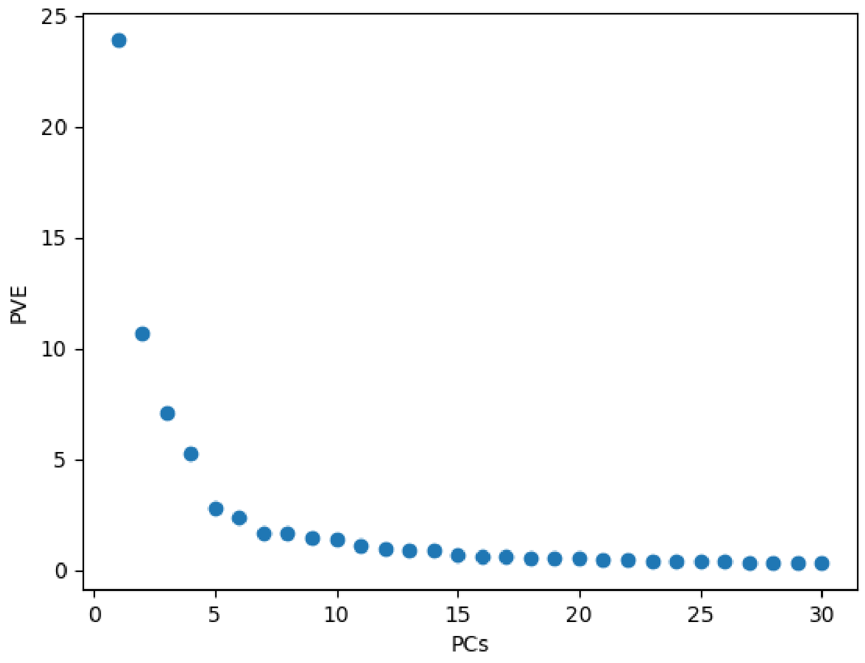

Figure 1: Percent variance explained by the first thirty expression principal components of

Q3: eQTL Calling

To detect eQTLs, variants with MAF below 0.01 were filtered out before running Matrix eQTL. Only genotype PCs and sex were regressed out as covariates.

Benjamini-Hochberg for all SNP-gene pairs

Using a window of 1Mb on either side around each gene's TSS, and filtering out SNP-gene pairs with a MAF less than 0.01, a total of 3,660,383 candidate SNP-gene pairs were identified. 40,837 statistically significant cis-eQTL's were identified at a false discovery rate of 0.05 using the Benjamini-Hochberg procedure across all 3,660,383 SNP-gene pairs.

Gene-level FDR control

Applying Bonferroni correction within each gene on chromosome 10, and taking the most significant adjusted p-value for each gene before applying Benjamini-Hochberg at FDR=0.05, 119 significant cis-eQTLs were identified. We did not explore the possibility that some genes could have multiple cis-eQTLs. This is discussed below.

Q4: Effect of Expression PCs

We found that the number of eQTLs detected with gene-level FDR 0.05 increased with the number of expression PCs regressed out.

Expression PCs	eGenes Detected
0	119
5	210
10	252
20	292
30	315

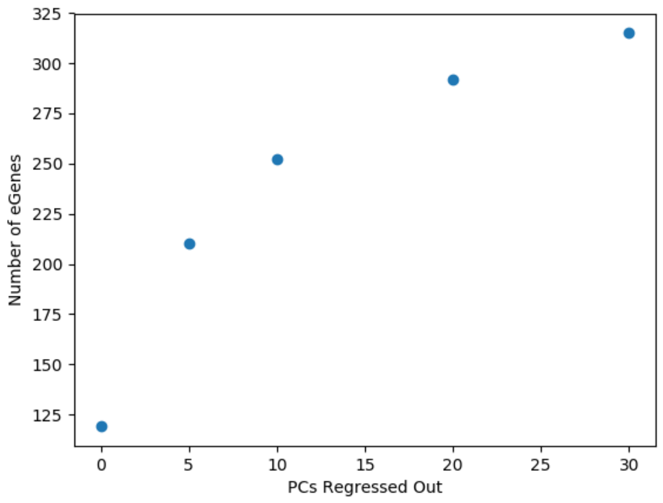

Figure 2 & Table 1: Number of eGenes detected vs number of expression PCs regressed out

Q5: Analysis

For optimal eQTL calling, I would probably regress out at least 30 expression PCs, though possibly more. I would run the following further analyses to decide:

1. Regress out more expression PCs (perhaps up to 50). If the number of eGenes detected begins to decrease, this would suggest that we had begun to regress out signal rather than noise, and the point where the most eGenes are detected could be an appropriate cutoff. I would not expect the number to increase too dramatically, as this curve appears to be “approaching saturation" (though not yet reaching it). Furthermore, considering the scree plot, additional PCs will not be regressing out a significant portion of the variance in our expression data. This suggests that there will not be much improvement in the sensitivity to eQTL detection as more PCs are regressed out.

2. For each set of eQTLs (those derived after regressing out 20, 25, 30 PCs, etc), I would run some functional enrichment tests. If the eQTLs that were detected after regressing out 25 PCs are significantly enriched for enhancer and promoter regions, but the significance of this trend notably decreases as more PCs are regressed out, this could suggest that we have begun picking up spurious signal. In this case, I would regress out 25 PCs. Similarly, if the number of eGenes detected continues to increase after 30, along with the significance of meaningful enrichments, then I would continue regressing out PCs.

3. It would be good to check how expression PCs correlate with collected metadata. If we find that even these last few PCs are correlating with variables such as sex, age, ancestry, etc. then we can be confident that regressing them out is the right decision. With eQTLs, we are looking for effects on a

single gene, which are not going to be easily captured by an entire expression PC. I would expect these broad trends to be captured by the first few PCs, but if they are still correlated with PCs 25-30, then I would be confident in regressing out 30 PCs.

With these potential caveats noted, given the information at hand, I would choose to regress out all 30 PCs.

References

[1] Battle, A. *et al.* Impact of regulatory variation from rna to protein. *Science* **347**, 664–667 (2015).

[2] Petretto, E. *et al.* Heritability and tissue specificity of expression quantitative trait loci. *PLoS genetics* **2** (2006).

[3] Hao, K., Chudin, E., McElwee, J. & Schadt, E. E. Accuracy of genome-wide imputation of untyped markers and impacts on statistical power for association studies. *BMC genetics* **10**, 27 (2009).

[4] Visscher, P. M., Hill, W. G. & Wray, N. R. Heritability in the genomics era—concepts and misconceptions. *Nature reviews genetics* **9**, 255–266 (2008).

[5] Manolio, T. A. *et al.* Finding the missing heritability of complex diseases. *Nature* **461**, 747–753 (2009).

Appendix - Code

Matrix eQTL

```
# Matrix eQTL by Andrey A. Shabalin
# http://www.bios.unc.edu/research/genomic_software/Matrix_eQTL/
#
# Be sure to use an up to date version of R and Matrix eQTL.

# source("Matrix_eQTL_R/Matrix_eQTL_engine.r");
library(MatrixEQTL)

args <- commandArgs()
cov <- args[[6]]

## Settings
data.loc = "/work-zfs/abattle4/lab_data/GTEx_v8_eqtl_practice/matrix_eqtl/"

# Linear model to use, modelANOVA, modelLINEAR, or modelLINEAR_CROSS
useModel = modelLINEAR; # modelANOVA, modelLINEAR, or modelLINEAR_CROSS

# Genotype file name
SNP_file_name = paste0(data.loc, "Whole_Blood.v8.genotype.chr10.txt");
snps_location_file_name = paste0(data.loc, "Whole_Blood.v8.snp_location.chr10.txt");

# Gene expression file name
expression_file_name = paste0(data.loc, "Whole_Blood.v8.normalized_expression.txt");
gene_location_file_name = paste0(data.loc, "Whole_Blood.v8.gene_location.txt");

# Covariates file name
covariates_file_name = paste0("data/", cov, ".txt");

# Output file name
output_file_name_cis = paste0("matrixeqtl/cis.eqtl.", cov, ".txt");
output_file_name_tra = tempfile();

# Only associations significant at this level will be saved
pvOutputThreshold_cis = 1;
pvOutputThreshold_tra = 0;

# Error covariance matrix
# Set to numeric() for identity.
errorCovariance = numeric();
# errorCovariance = read.table("Sample_Data/errorCovariance.txt");

# Distance for local gene-SNP pairs
cisDist = 1e6;

## Load genotype data

snps = SlicedData$new();
snps$fileDelimiter = "\t";      # the TAB character
snps$fileOmitCharacters = "-"; # denote missing values;
snps$fileSkipRows = 1;          # one row of column labels
```

```
snps$fileSkipColumns = 1;       # one column of row labels
snps$fileSliceSize = 2000;      # read file in slices of 2,000 rows
snps$LoadFile(SNP_file_name);

## Load gene expression data

gene = SlicedData$new();
gene$fileDelimiter = "\t";      # the TAB character
gene$fileOmitCharacters = "NA"; # denote missing values;
gene$fileSkipRows = 1;          # one row of column labels
gene$fileSkipColumns = 1;       # one column of row labels
gene$fileSliceSize = 2000;      # read file in slices of 2,000 rows
gene$LoadFile(expression_file_name);

## Load covariates

cvrt = SlicedData$new();
cvrt$fileDelimiter = "\t";      # the TAB character
cvrt$fileOmitCharacters = "NA"; # denote missing values;
cvrt$fileSkipRows = 1;          # one row of column labels
cvrt$fileSkipColumns = 1;       # one column of row labels
if(length(covariates_file_name)>0) {
  cvrt$LoadFile(covariates_file_name);
}

## Run the analysis
snpspos = read.table(snps_location_file_name, header = TRUE, stringsAsFactors = FALSE);
genepos = read.table(gene_location_file_name, header = TRUE, stringsAsFactors = FALSE);

# Filter out snps with MAF<0.01
maf.list = vector('list', length(snps))
for(sl in 1:length(snps)) {
  slice = snps[[sl]];
  maf.list[[sl]] = rowMeans(slice,na.rm=TRUE)/2;
  maf.list[[sl]] = pmin(maf.list[[sl]],1-maf.list[[sl]]);
}
maf = unlist(maf.list)

## Look at the distribution of MAF
cat('SNPs before filtering:',nrow(snps))
snps$RowReorder(maf>=0.01);
cat('SNPs after filtering:',nrow(snps))

me = Matrix_eQTL_main(
  snps = snps,
  gene = gene,
  cvrt = cvrt,
  output_file_name     = output_file_name_tra,
  pvOutputThreshold     = pvOutputThreshold_tra,
  useModel = useModel,
  errorCovariance = errorCovariance,
  verbose = TRUE,
  output_file_name.cis = output_file_name_cis,
  pvOutputThreshold.cis = pvOutputThreshold_cis,
  snpspos = snpspos,
  genepos = genepos,
  cisDist = cisDist,
  pvalue.hist = "qqplot",
  min.pv.by.genesnp = FALSE,
  noFDRsaveMemory = TRUE);

unlink(output_file_name_tra);
#unlink(output_file_name_cis);

## Results:
cat('Analysis done in: ', me$time.in.sec, ' seconds', '\n');
cat('Detected local eQTLs:', '\n');
show(me$cis$eqtls)

## Plot the Q-Q plot of local and distant p-values
png(paste0('plots/', cov, 'qq.png'))
plot(me)
dev.off()
```

Count SNPs per Gene

```
import numpy as np

# load snp file
snpfile = "/work-zfs/abattle4/lab_data/GTEx_v8_eqtl_practice/matrix_eqtl/Whole_Blood.v8.snp_location.chr
snplocs = np.loadtxt(snpfile, dtype=int, skiprows=1, usecols=2)
snplocs = np.array(snplocs)

# select from this array only the snps that passed the MAF threshold
maf_file = open("snps.passed", "r")
fl = maf_file.readlines()
maf_snps = [f.strip() for f in fl]

allsnp_file = open(snpfile, "r")
fl = allsnp_file.readlines()
all_snps = [f.strip().split()[0] for f in fl[1:]]

assert(len(all_snps) == len(snplocs))
assert(len(maf_snps) == 339919)

_, inds, _ = np.intersect1d(all_snps, maf_snps, return_indices=True)
snplocs = np.take(snplocs, sorted(inds))

# load gene file
genefile = "/home-2/jpopp4@jhu.edu/work/josh/genomic-data-science/hw1/data/chr10genelocs.txt"
genelocs = np.loadtxt(genefile, dtype=int, skiprows=0, usecols=(2,3))
genelocs = np.array(genelocs)

generanges = np.zeros(genelocs.shape)
generanges[:, 0] = genelocs[:, 0] - 1e6
generanges[:, 0] = [i if i>=1 else 1 for i in generanges[:,0]]  # i think they use one-indexing?

chr10len = 133797422
```

```
generanges[:, 1] = genelocs[:, 1] + 1e6
generanges[:, 1] = [i if i<=chr10len else chr10len for i in generanges[:,1]]

snpcount = np.zeros(len(generanges), dtype=int)
startSNP = 0
for i in range(len(generanges)):
    g = generanges[i]
    for j in range(startSNP, len(snplocs)):
        snploc = snplocs[j]
        if snploc < g[0]:
            startSNP = j
        elif g[0] <= snploc <= g[1]:
            snpcount[i] += 1
        else:
            break

np.savetxt("snpcounts.chr10.maffiltered.txt", snpcount, fmt="%d")
```

Count QTLs

```
import numpy as np
import matplotlib.pyplot as plt


def get_qtl_data(cov):
    """
    Input: indicator of which covariates to use (cov1, cov2, ..., or cov5)
    Output: snps (list of str), genes (list of str), pvals (ndarray of float)
            can assume all three are in the same order
    """

    fname = "/home-2/jpopp4@jhu.edu/work/josh/genomic-data-science/hw1/matrixeqtl/cis.eqtl." + cov + ".t
    f = open(fname, 'r')
    lines=f.readlines()
    snps = []
    genes=[]
    for l in lines[1:]:
        snps.append(l.split('\t')[0])
        genes.append(l.split('\t')[1])
    f.close()
    pvals = np.array(np.loadtxt(fname, skiprows=1, usecols=4))
    assert(len(snps) == len(genes))
    assert(len(genes) == len(pvals))
    return snps, genes, pvals


def get_SNPs_per_gene():
    """
    Input: none
    Output: dictionary of gene names (list of str) to snp counts per gene (ndarray of int, calculated by
    """
    # get list of gene names
    f = open("/home-2/jpopp4@jhu.edu/work/josh/genomic-data-science/hw1/data/chr10genelocs.txt", 'r')
    lines=f.readlines()
    genes=[]
    for l in lines:
        genes.append(l.split('\t')[0])
    f.close()

    # get list of snp counts per gene
    qtlfile="/home-2/jpopp4@jhu.edu/work/josh/genomic-data-science/hw1/snpcounts.chr10.maffiltered.txt"
    snp_counts = np.array(np.loadtxt(qtlfile), dtype=int)

    gene2snpcount = dict(zip(genes, snp_counts))
    return gene2snpcount


def count_bh_significant_snps(pvals, outfile, ntest, fdr=0.05, tied_rankings=False):
    """
    Input: pvals (ndarray of floats), outfile (str), ntest (int), [fdr (float)]
    Output: file located at outfile will contain one line saying how many eqtl's are
            identified by Benjamini-Hochberg procedure at the given false discovery
            rate
    """
    pvals = np.sort(pvals)
    if tied_rankings:
        ranks = np.zeros(len(pvals), dtype=int)
        ranks[0] = 1
        increment = 1
        for i in range(1, len(pvals)):
            if pvals[i] == pvals[i-1]:
                ranks[i] = ranks[i-1]
                increment += 1
            else:
                ranks[i] = ranks[i-1]+increment
                increment = 1
    else:
        ranks = range(1, len(pvals)+1)
    bh = [(r/ntest)*fdr for r in ranks]
    assert(len(pvals) == len(bh))
    fout = open(outfile, "w+")
    for i in range(len(bh)):
        pos=len(bh)-1-i
        if pvals[pos] < bh[pos]:
            fout.write("%d eQTL's identified\n" % (pos+1))
            fout.close()
            found=True
            return (pos+1)
    print("0 eQTL's identified\n")
    fout.close()
    return 0


def bh_all_combos():
    snps, genes, pvals = get_qtl_data("cov1")
    nsnp = count_bh_significant_snps(pvals, "qtlcount/all_combos.txt", 3660383)
    return nsnp
```

```
def bh_gene_level(cov):
    gene2count = get_SNPs_per_gene()
    snps, genes, pvals = get_qtl_data(cov)
    for i in range(len(genes)):
        pvals[i] *= gene2count[genes[i]]  # bonferroni adjust
    # take top pval for each gene
    ordered = np.argsort(pvals)
    pvals = [pvals[o] for o in ordered]
    genes = [genes[o] for o in ordered]
    snps = [snps[o] for o in ordered]

    genes, uniq = np.unique(np.array(genes), return_index=True)
    pvals = [pvals[u] for u in uniq]
    snps = [snps[u] for u in uniq]

    outfile="qtlcount/"+cov+".txt"
    nsnp =  count_bh_significant_snps(pvals, outfile, 765)
    return nsnp


def plot_egenes_vs_covs(covs, egenes):
    plt.scatter(covs, egenes)
    plt.xlabel('PCs Regressed Out')
    plt.ylabel('Number of eGenes')
    plt.savefig('./plots/egenes.png')


def main():
    all_combined = bh_all_combos()
    cov1genes = bh_gene_level("cov1")
    cov2genes = bh_gene_level("cov2")
    cov3genes = bh_gene_level("cov3")
    cov4genes = bh_gene_level("cov4")
    cov5genes = bh_gene_level("cov5")
    plot_egenes_vs_covs([0, 5, 10, 20, 30],
                        [cov1genes, cov2genes, cov3genes, cov4genes, cov5genes])


if __name__=="__main__":
    main()
```