Finding allelic imbalance: Simple approach

March 31, 2017

NiekdeKlein@gmail.com

Niek de Klein

For this tutorial we will use GATK Allele Counter to get the reference and alternative counts for a few genes that are associated with schizophrenia (nature.com/nature/journal/v511/n7510/full/nature13595.html), and use a wilcoxon-test to test for allelic imbalance. Due to compute time limit we will start with pre-aligned reads. In a typical RNAseq experiment you will have to do the alignment yourself. The first part will be done in Bash to get allelic counts from BAM files for a small subset of the SNPs. The second part, plotting the results and statistically testing for allelic imbalance will be done in R using all SNPs.

Before we start

Check if all programs are installed on your VM. Open the terminal and type below commands. Each should give instructions how to run the program

```
/opt/bcftools-1.4/bcftools
java -jar /opt/GenomeAnalysisTK.jar --help
/opt/samtools-1.4/samtools
/opt/htslib-1.4/bgzip
/opt/htslib-1.4/tabix
```

See if the data is available at /virdir/Backup/ASE/

```
ls /virdir/Backup/ASE/
```

This should show the files and directories

```
BIOS_freeze2.biallelicOnly.subset.hetSamples.vcf.gz
BIOS_freeze2.biallelicOnly.subset.hetSamples.vcf.gz.tbi
human_g1k_v37.fasta
human_g1k_v37.fasta.fai
human_g1k_v37.dict
gene_regions.interval
haplotypes.txt
bam_files/
counted_reads/
```

In the first steps we will use the bam (binary aligned reads) files and BIOS_freeze2.biallelicOnly.subset.vcf.gz to get allelic counts for a few genes. In the second step we will use many allelic counts from counted_reads/ to test for allelic imbalance and do some plotting.

Advice

Please use and text editor, such as wordpad or notepad++ to store your commands before you issue them. This makes it easy to change and fix problems, plus after you issue your first command you can easy recycle a lot of the parts used in the first command.

Inspecting VCF

First lets see which SNPs we have currently in /virdir/Backup/ASE/BIOS_freeze2.biallelicOnly.subset.hetSamples.vcf.gz. It doesn't matter if the SNPs are from whole genome sequencing, chip genotypes or inferred from RNAseq. In this case it is a subset of SNPs that fall within the regions of the eQTLs that were found in nature.com/nature/journal/v511/n7510/full/nature13595.html.
You can inspect the vcf with

```
zcat /virdir/Backup/ASE/BIOS_freeze2.biallelicOnly.subset.hetSamples.vcf.gz | less -S
```

1. How many SNPs do we have?
2. How many samples are heterozygous per SNP (Given that 0|0 is homozygous, 1|0 and 0|1 is heterozygous on on different alleles)?

The first question is in this case easy answerable by counting the number of lines, since there are not so many. For the number of heterozygotous per SNP we need to use some programming. In bash this can be done with

```
zcat /virdir/Backup/ASE/BIOS_freeze2.biallelicOnly.subset.hetSamples.vcf.gz |
  grep -v "^#" |
  awk -F"1\\|0|0\\|1" "{print NF-1}"
```

Where `zcat /virdir/Backup/ASE/BIOS_freeze2.biallelicOnly.subset.hetSamples.vcf.gz` reads the gzipped file, `grep -v "^\#"` selects the lines that do not start with #, removing the header lines, and

```
awk -F"1\\|0|0\\|1" "{print NF-1}"
```

counts how often 1|0 and 0|1 occurs. The \\escape the | character for the genotypes. Now you can answer question 2.

Getting allelic counts

We now know that there are 10 SNPs in the example vcf for which some of the individuals are heterozygotous, and can count the number of RNA-seq counts per individual, for the alternative allele and reference allele separately. There are several tools to do this with, such as samtools mpileup, phASER, and GATK AlleleCounter. In this tutorial we will use the GATK AlleleCounter. We do this for all the individuals that had at least one heterozygotous SNP in our subset. The BAM files are in /virdir/Backup/ASE/bam_files/, so we have to loop over them

```
mkdir countTables/
# loop over all bam files
for BAMFILE in /virdir/Backup/ASE/bam_files/*bam;
do
  echo $BAMFILE;
  OUTNAME=$(basename ${BAMFILE%bam}countTable.txt);
  java -jar /opt/GenomeAnalysisTK.jar \
    -T ASEReadCounter \
    -R /virdir/Backup/ASE/human_g1k_v37.fasta \
    -o countTables/$OUTNAME \
    -I $BAMFILE \
    -sites /virdir/Backup/ASE/BIOS_freeze2.biallelicOnly.subset.hetSamples.vcf.gz \
    -U ALLOW_N_CIGAR_READS \
    -minDepth 10 \
    --minMappingQuality 10 \
    --minBaseQuality 2
done
```

We use a few, not very strict, quality cut-offs. Each read has to have a mapping quality of at least 10 (minMappingQuality) and the base quality at the SNP has to be at least 2 (minBaseQuality). Per sample each SNP has to have at least 10 reads overlapping that pass mapping and base quality (minDepth), or it will not be counted for that sample.

Getting allelic counts continued on next page. . .

Per bam file in /virdir/Backup/ASE/bam_files/ you will get one output file in countTables/. These output files contain info on the SNP, counts per ref and alt allele, and some additional information about the quality of the reads overlapping the SNP.

Plotting allelic imbalance per sample

The previous part showed how to do get the allelic counts for multiple samples and one SNP. For the next part, the plotting, we will use the pre-made data with the same samples, but with additional SNPs. Due to computation time limits these have been provided already and are located in data/rtables/.
The next par will be in R. We read in all the different tables and combine them. First we get a list of all the file names.
Type R in the command line, or use RStudio.

```
##### read in multiple files #####
allelic_counts <- data.frame()
# get the paths of all the rtables
rtable_filenames <- list.files(path="/virdir/Backup/ASE/counted_reads/",
               pattern = "\\.rtable$",full.names=T)
```

Then we loop over all the files and merge the data together.

```
for (filename in rtable_filenames){
 # get the filename without the path
 filename_base <- basename(filename);
 # remove the extensions to get the sample name
 sample_name <- unlist(strsplit(filename_base,".rtable"))[1];
 print(sample_name);
 # read in the file
 allelic_counts_tmp <- read.table(filename,header=TRUE);
 # add the samplename to the table
 allelic_counts_tmp$sample <- sample_name;
 # merge the tables from differnet files together

 allelic_counts <- rbind(allelic_counts, allelic_counts_tmp);
}
```

Now we can plot the ref and alt counts for all SNPs and samples to get a first idea of the data.

```
library(ggplot2)
ggplot(allelic_counts, aes(x=refCount, y=altCount))+
geom_point(alpha=0.1)
```

This should make figure 1. You can see three distinct groups, in the left, in the middle and at the bottom. What are these different groups?

As you can see, there are quite a lot of points with (almost) 0 ref or al counts. These are samples that are homozygous for that particular SNP. Since these SNPs are not informative, we remove them from our data table.

```
allelic_counts <- allelic_counts[allelic_counts$refCount > 0 &
                              allelic_counts$altCount > 0,]
```

Plotting allelic imbalance per sample continued on next page. . .

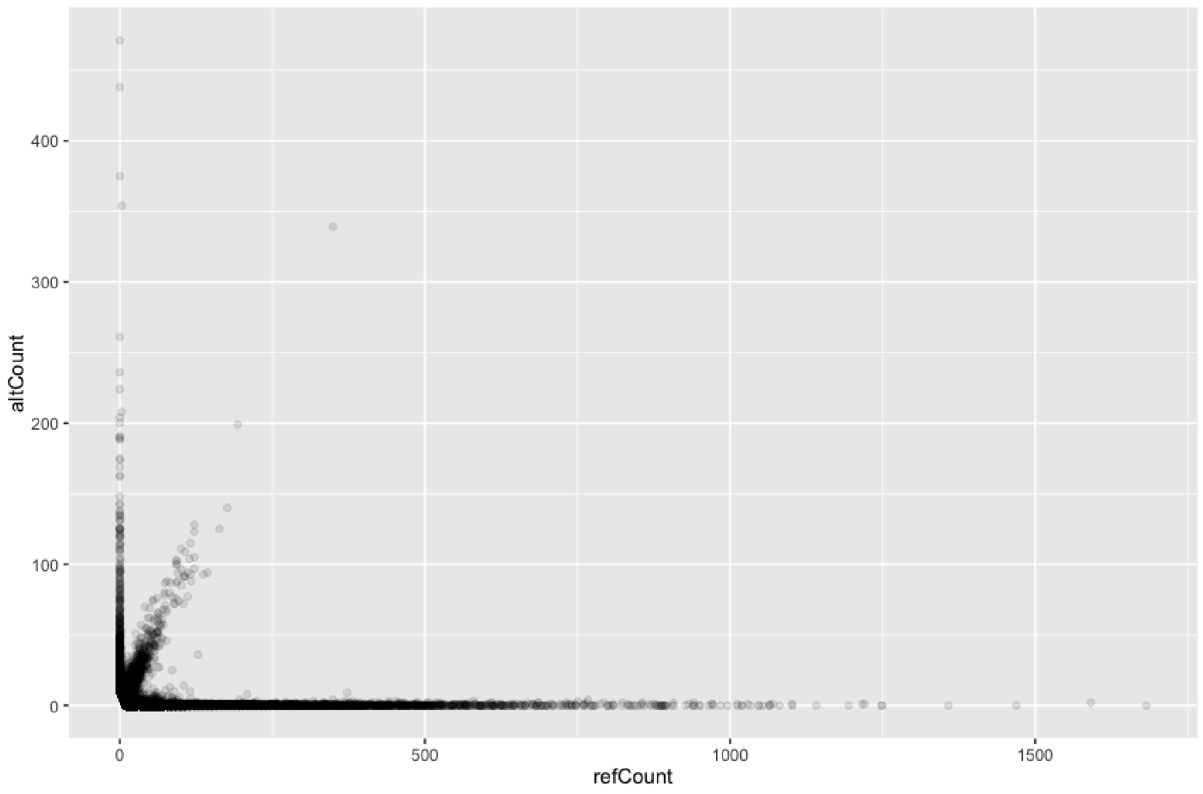

Figure 1: Allelic imbalance per sample.

How many data points did this remove?
Also, to circumvent reference bias, we only include SNPs where the ref/alt ratio is between 0.05-0.4 and 0.6-0.95.

```
# remove points not ref/alt >0.05 <0.4 >0.6 <0.95
# calculate ref/alt ratio
allelic_counts$refAltRatio <- allelic_counts$refCount /
                            (allelic_counts$refCount+allelic_counts$altCount)
allelic_counts <- allelic_counts[(allelic_counts$refAltRatio > 0.05 &
                          allelic_counts$refAltRatio < 0.4) |
                         (allelic_counts$refAltRatio > 0.6 &
                          allelic_counts$refAltRatio < 0.95), ]
```

When we now redo the same plot you can more clearly see the allelic imbalance in figure 2

```
ggplot(allelic_counts, aes(x=refCount, y=altCount))+
geom_point(alpha=0.1)
```

Because one gene can have multiple SNPs we need to add up the reads per gene for each sample. To do this we first include gene info to our allelic count table.

Plotting allelic imbalance per sample continued on next page. . .

```
# read the gene info
gene_regions <- read.table("/virdir/Backup/ASE/gene_regions.interval", header=TRUE)
# add gene info to allelic count table
allelic_counts$gene <- NA
for (index in 1:nrow(allelic_counts)){
    print(paste0(index,"/",nrow(allelic_counts)))
    allelic_counts[index,]$gene <- as.character(
    gene_regions[gene_regions$chr == allelic_counts[index,]$contig &
    gene_regions$start < allelic_counts[index,]$position &
    gene_regions$stop > allelic_counts[index,]$position,]$gene)
}
```

And then per gene we can sum the number of reads per haplotype. This is possible because the genotypes that we have are phased, e.g. we know for each SNP if the reference is . You can see this in the VCF file because the genotypes are denoted as 0|1 or 1|0. Unphased genotypes are written as 0/1, with a dash. So because the genotypes are phased, if in a gene you have 4 SNPs that are denoted like this:
SNP1 1|0
SNP2 0|1
SNP3 0|1
SNP4 1|0

you can sum the counts per haplotype by adding the alt count of SNP1 and SNP4, and the ref count of SNP2 and SNP3 for haplotype 1, and the opposite for haplotype 2. First we need to load the haplotype information. This is the same as in the VCF, but with only the SNP information loaded and split over haplotype 1 and haplype 2, with for haplotype 1 the SNP information before the | and haplotype 2 the SNP information after the |.

```
library(data.table)
# read the gene info
haplotypes <- data.frame(fread("/virdir/Backup/ASE/haplotypes.txt", header=TRUE))
head(haplotypes[1:10])
```

Then we can sum over the different haplotypes. Because we want to get the haplotypic count per gene per sample we need to use a few loops.

Plotting allelic imbalance per sample continued on next page...

```
# for every sample, sum counts for all SNPs within a gene
allelic_count_per_gene <- data.frame(sample=c(), gene=c(),
                         haplotype_1=c(), haplotype_2=c())
# loop over the unique samples
for (sample_name in unique(allelic_counts$sample)){
    # loop over the unqiue genes for which this sample had heterozygotous SNP
    for (gene in unique(allelic_counts[allelic_counts$sample==sample_name,]$gene)){
        # subset the data
        allelic_subset <- allelic_counts[allelic_counts$sample==sample_name &
                                allelic_counts$gene==gene,]
        summed_haplotype_1 <- 0;
      summed_haplotype_2 <- 0
        # loop over all the SNPs in the gene

        for(i in 1:nrow(allelic_subset)){
            # subset count data for current SNP
            snp_count <- allelic_subset[i,]
            pos <- paste0(allelic_subset[i,]$contig,"_",allelic_subset[i,]$position)
            # because the colnames that get read in replace - for .,
            # need to do same in samplename
            sample_name_fix <- gsub(x=sample_name, pattern="-", replacement ="\\.")

            # get the genotype at that position for current sample
            allele_1 <- haplotypes[haplotypes$chr_pos == pos,
                              paste0(sample_name_fix,".allele1")];
            allele_2 <- haplotypes[haplotypes$chr_pos == pos,
                              paste0(sample_name_fix,".allele2")]

            # Check which haplotype the SNP is on
            if(allele_1 == "1" & allele_2 == "0"){
                # if SNP on first haplotype, sum alt count with hap_1
                # and ref count with hap_2
                summed_haplotype_1 <- summed_haplotype_1+snp_count$altCount;
                summed_haplotype_2 <- summed_haplotype_1+snp_count$refCount;
            }
            if(allele_1 == "0" & allele_2 == "1"){
                # if the SNP is on the second haplotype, count other way

                summed_haplotype_1 <- summed_haplotype_1+snp_count$refCount;
                summed_haplotype_2 <- summed_haplotype_1+snp_count$altCount;
            }
        }
        # Put the counts in a dataframe with sample and gene info
        df <- data.frame(sample=sample_name,
                         gene=gene,
                         haplotype_1=summed_haplotype_1,
                         haplotype_2=summed_haplotype_2)
        # combine all the data
        allelic_count_per_gene <- rbind(allelic_count_per_gene,df)
    }
}
```

Plotting allelic imbalance per sample continued on next page. . .

Remove homozygous samples from the dataset (where both counts is 0)

```
    allelic_count_per_gene_filter <- allelic_count_per_gene[
                        allelic_count_per_gene$haplotype_1 > 0 |
                        allelic_count_per_gene$haplotype_2 > 0,]
```

After summing the haplotype count per gene we can now plot this as in figure 3.

```r
# per gene make a separate plot
library(gridExtra)
library(grid)

# make a plotting function
plotRatios <- function(gene){
  # subset based on gene
  allelic_count <- allelic_count_per_gene[allelic_count_per_gene$gene==gene,]

  # get the max value so we can plot with x-axis and y-axis limits the same,
  # makes it easier to see the imbalance

  maxVal <- max(allelic_count$haplotype_1, allelic_count$haplotype_2)
  ggplot(allelic_count, aes(x=haplotype_1, y=haplotype_2))+
    geom_point(alpha=0.5)+
    geom_abline(slope=1, intercept=0)+
    ggtitle(gene)+
    xlim(0,maxVal)+
    ylim(0,maxVal)
}

p1 <- plotRatios("ATXN7")
p2 <- plotRatios("LRRC48")
p3 <- plotRatios("EP300")
p4 <- plotRatios("SLC39A8")

pdf("haplotype_counts.pdf")
grid.arrange(p1,p2,p3,p4)
dev.off()
```

And finally, we can test if there is a significant difference between the two haplotypes, i.e. is there an allelic imbalance?

```r
EP300 <- allelic_count_per_gene_filter[allelic_count_per_gene_filter$gene == "EP300",]

wilcox.test(EP300$haplotype_1,EP300$haplotype_2)
```

Do all 4 genes show an allelic imbalance?

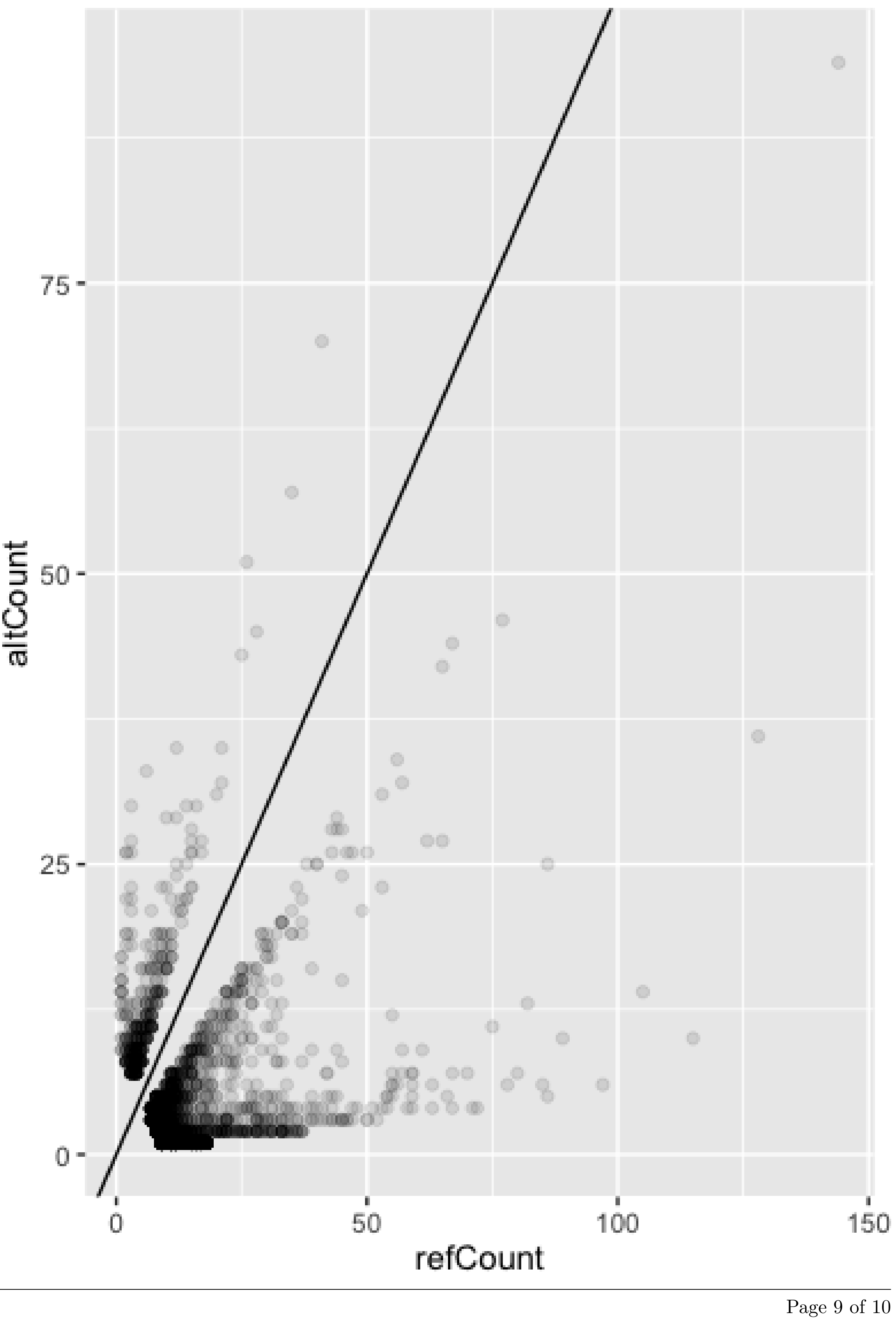

Figure 2: Allelic imbalance per sample.

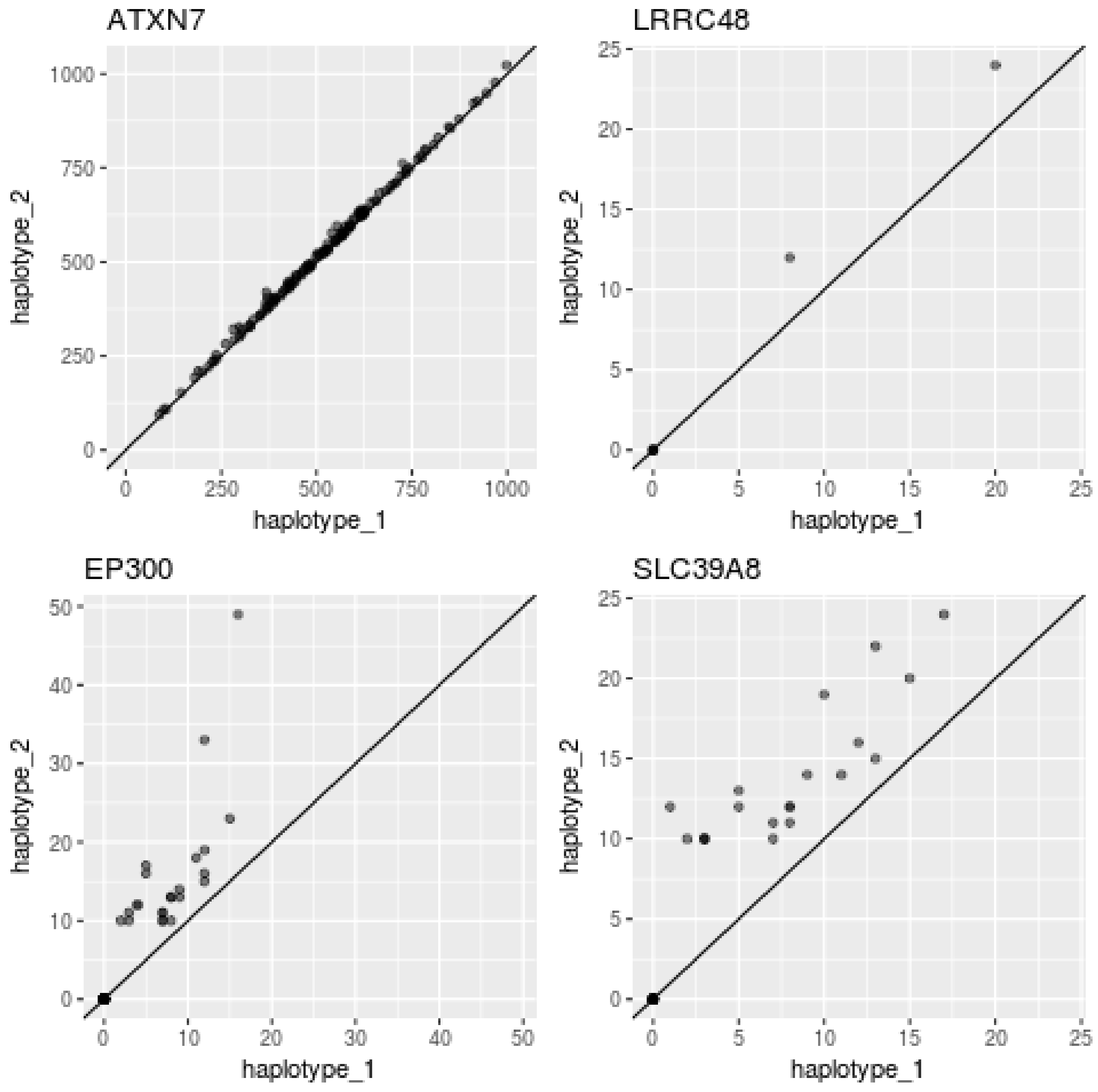

Figure 3: Allelic imbalance per gene.