Supplementary Information

Andrew Ghazi

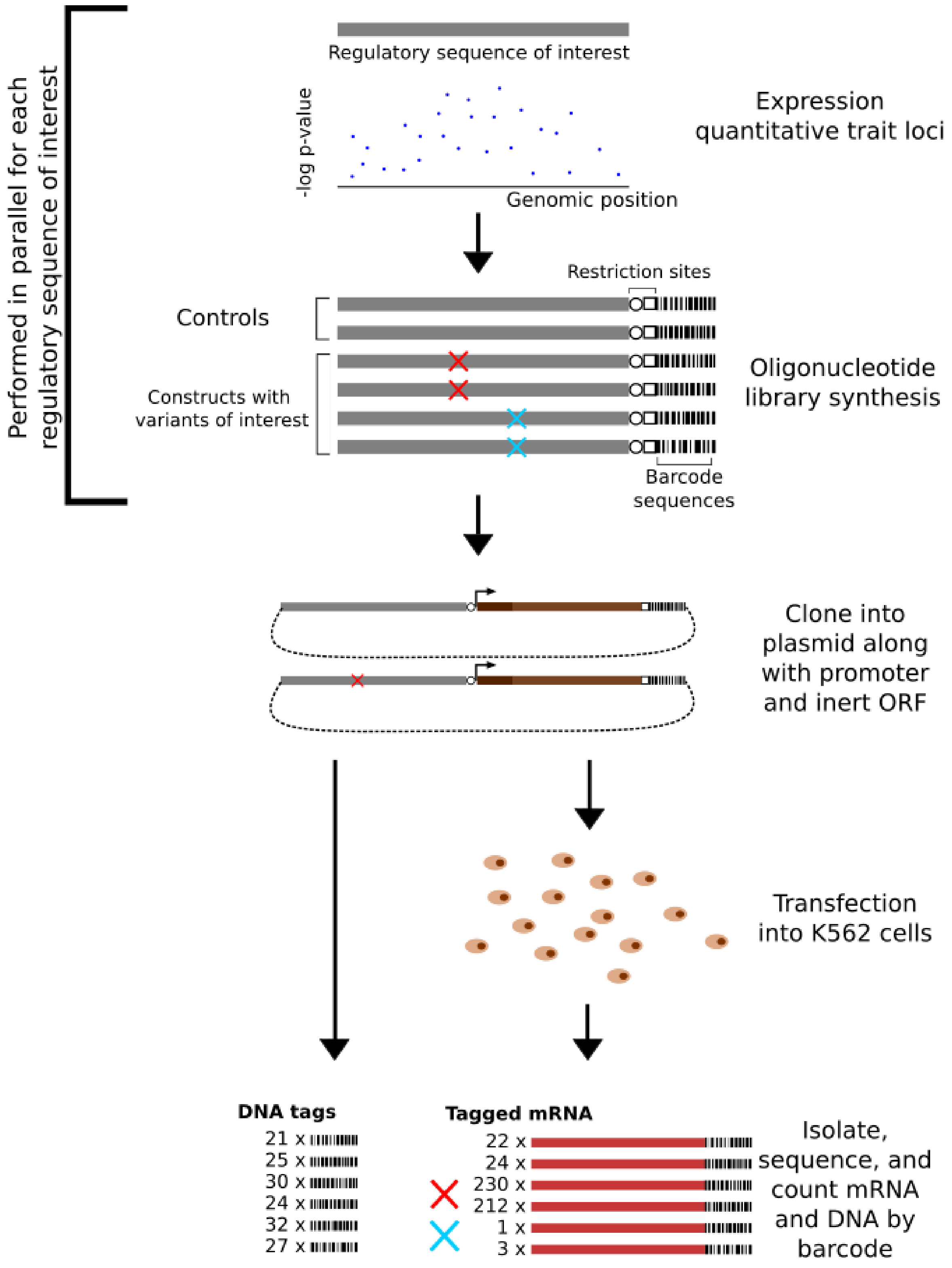
6/21/2017

Contents

The raw code for this RMarkdown document can be found at this link.

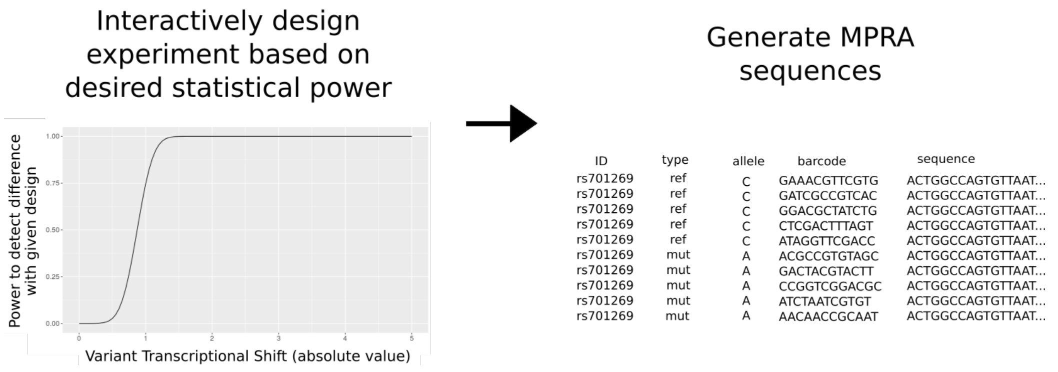
S1 - MPRA diagrams

S1.1 MPRA experimental diagram



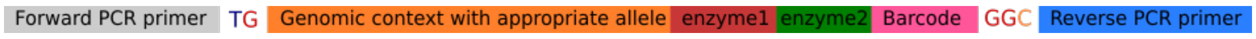
Performed in parallel for each regulatory sequence of interest
Regulatory sequence of interest
-log p-value
Genomic position
Expression quantitative trait loci
Restriction sites
Controls
Constructs with variants of interest
Oligonucleotide library synthesis
Barcode sequences
Clone into plasmid along with promoter and inert ORF
Transfection into K562 cells
DNA tags
21 x
25 x
30 x
24 x
32 x
27 x
Tagged mRNA
22 x
24 x
230 x
212 x
1 x
3 x
Isolate, sequence, and count mRNA and DNA by barcode

S1.2 MPRA Design Tools Workflow


Interactively design experiment based on desired statistical power
Power to detect difference with given design
1.00
0.75
0.50
0.25
0.00
0
1
2
3
4
5
Variant Transcriptional Shift (absolute value)
Generate MPRA sequences

ID	type	allele	barcode	sequence
rs701269	ref	C	GAAACGTTCGTG	ACTGGCCAGTGTTAAT...
rs701269	ref	C	GATCGCCGTCAC	ACTGGCCAGTGTTAAT...
rs701269	ref	C	GGACGCTATCTG	ACTGGCCAGTGTTAAT...
rs701269	ref	C	CTCGACTTTAGT	ACTGGCCAGTGTTAAT...
rs701269	ref	C	ATAGGTTCGACC	ACTGGCCAGTGTTAAT...
rs701269	mut	A	ACGCCGTGTAGC	ACTGGCCAGTGTTAAT...
rs701269	mut	A	GACTACGTACTT	ACTGGCCAGTGTTAAT...
rs701269	mut	A	CCGGTCGGACGC	ACTGGCCAGTGTTAAT...
rs701269	mut	A	ATCTAATCGTGT	ACTGGCCAGTGTTAAT...
rs701269	mut	A	AACAACCGCAAT	ACTGGCCAGTGTTAAT...

S1.3 MPRA Design Tools Sequence Element Layout



`enzyme1` and `enzyme2` default to KpnI and XbaI as in Melnikov et al., Nature Biotechnology 2012. The forward and reverse primers default to `ACTGGCCAG` and `CTCGGCGGCC` respectively.

S2 - MPRA Activity Variance

Below is an analysis and visualization of the variance of MPRA activity measurements. We do this by computing the activity of each barcode for an allele, then taking the standard deviation of those activities. Repeating this for all alleles, we can see what the distribution of standard deviations is across alleles.

S2.1 - MPRA Activity Variance in Tewhey et al., Cell 2016

The study's data were acquired through direct correspondence with the authors. We compute the activity of each barcode in each transfection. Then, for every allele in every transfection, we compute the standard deviation of the activities. We then produce a histogram of the standard deviations in each transfection.

Below is the complete code necessary to perform this analysis. First we read in the data, then we normalize the counts according to sample depths, then we compute the barcode activity levels and their standard deviations within each allele.

```r
#library(tidyverse)
library(dplyr)
library(purrr)
library(readr)
library(ggplot2)
library(tidyr)
library(magrittr)
library(parallel)
library(knitr)
library(nortest)
```

```r
dir = '/mnt/bigData2/andrew/MPRA/Tewhey/indivTags/'

file_names = list.files(dir,
                        pattern = '.expanded$')
```

```
getFileDepth = function(file){
  # function to get the total number of reads in a sample.
  # Used to normalize counts by sample depth.

  read_tsv(paste0(dir, file),
           col_names = c('allele', 'barcode', 'count')) %>%
    .$count %>%
    sum
}

#apply the above function to the sample files
fileDepths = data_frame(src = file_names,
                        fileDepth = mclapply(src,
                                             getFileDepth,
                                             mc.cores = 6) %>% unlist)
```

A few example DNA barcode counts by sample file:

```
dnaCounts = map(1:5, ~read_tsv(paste0(dir, file_names[.x]),
                               col_names = c('allele', 'barcode', 'count')) %>%
                  mutate(src = file_names[.x])) %>%
  bind_rows

dnaCounts %>% # show a random sample of the counts
  sample_n(5) %>%
  kable

dnaCounts %<>%
  left_join(fileDepths, by = 'src') %>%
  mutate(depthAdjCount = 1e6*count/fileDepth)

depthAdjDNAmeanCount = dnaCounts %>%
  group_by(barcode) %>%
  summarise(allele = allele[1],
            bcMean = mean(depthAdjCount)) %>%
  ungroup
#grouping together the huge number of barcodes takes a while,
#so this is saved and loaded

save(dnaCounts,
     file = '~/designMPRA/outputs/tewheyDNAcounts.RData')
save(depthAdjDNAmeanCount,
     file = '~/designMPRA/outputs/tewheyDepthAdjDNAcounts.RData')
```

Allele	Barcode	Count	Transfection File Name
rs116303217_altA	CATGCTTCGTTAGGGTCGCC	7	Geuv_90K_ctrl.r4.tag.ct.indiv.expanded
rs55906525_RC_B	GTCACGCTTAGCCAATGAGA	20	Geuv_90K_ctrl.r4.tag.ct.indiv.expanded
rs13228599_RC_B	ACGAATCACAACAAATGTAT	19	Geuv_90K_ctrl.r5.tag.ct.indiv.expanded
rs9358930_A	AGAACAAACGTTGTCCACCT	22	Geuv_90K_ctrl.r3.tag.ct.indiv.expanded
rs17721766_A	CCATCGTACTAGGAAGTCGA	7	Geuv_90K_ctrl.r5.tag.ct.indiv.expanded

So to get the DNA normalization factor we simply take the mean of the DNA counts across the DNA transfections. For example:

```
depthAdjDNAmeanCount %>%
  head %>%
  set_names(c('Barcode', 'Allele', 'Mean Depth Adjusted DNA Count')) %>%
  kable
```

Barcode	Allele	Mean Depth Adjusted DNA Count
AAAAAAAAAAAACCAAGCGG	rs116221068_B	0.0835755
AAAAAAAAAAAACGTACTTC	rs9977746_B	0.0245955
AAAAAAAAAAAATGGCTCAC	rs6051692_RC_A	0.0228665
AAAAAAAAAAAATTCCACGT	rs2915876_RC_A	0.0628202
AAAAAAAAAAACAAAAGTCC	rs2234161_B	0.2756719
AAAAAAAAAAACAGTGTTTT	rs59955136_A	0.2367092

So for example the first barcode for allele rs116221068_B was counted 0.0835755 times on average across the plasmid sequencing runs (after adjusting for depth). The depth adjustment is performed as follows:

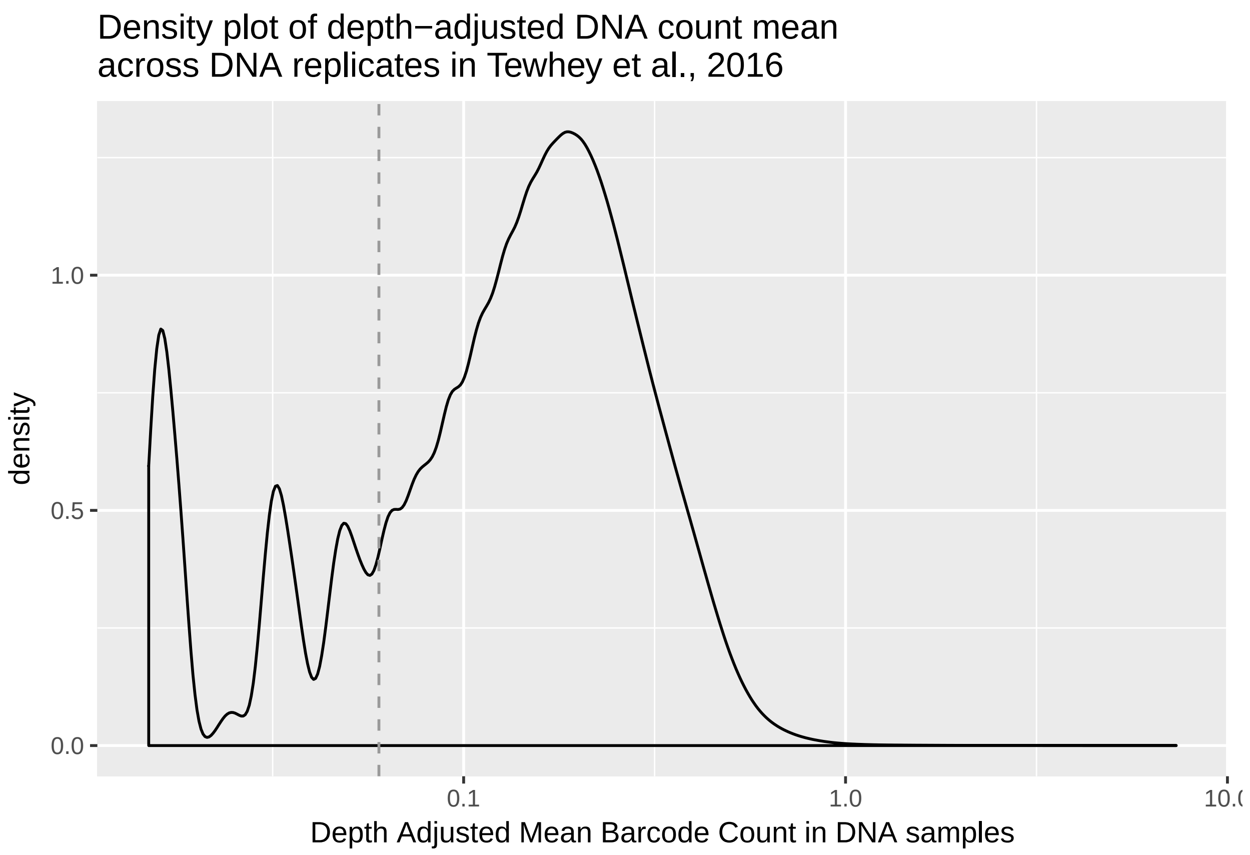
$$10^6 * \frac{count}{sum\, of\, barcode\, counts\, in\, sequencing\, run}$$

This normalizes each observed count according to how deeply the replicate was sequenced.

Tewhey et al., Cell 2016 had five plasmid replicates. Therefore the estimates of the mean DNA counts from this study will likely be more precise than the counterparts in Ulirsch et al., Cell 2016 which had only two plasmid replicates. This will in turn make the downstream activity measurements more stable and thus the activity standard deviations in this paper will likely be lower.

We exclude barcodes that were not well represented in the DNA samples. Visual inspection of a (log-scale) density plot of the mean depth-normalized count shows that .06 would be a reasonable cutoff (shown as a dotted vertical line). This cuts out 4,221,460 out of 19,611,641 barcodes. The first few modes represents failed barcodes with very low counts while the large mode represents well-performing barcodes (for further example see Ulirsch et al., Cell 2016 Figure 1B):

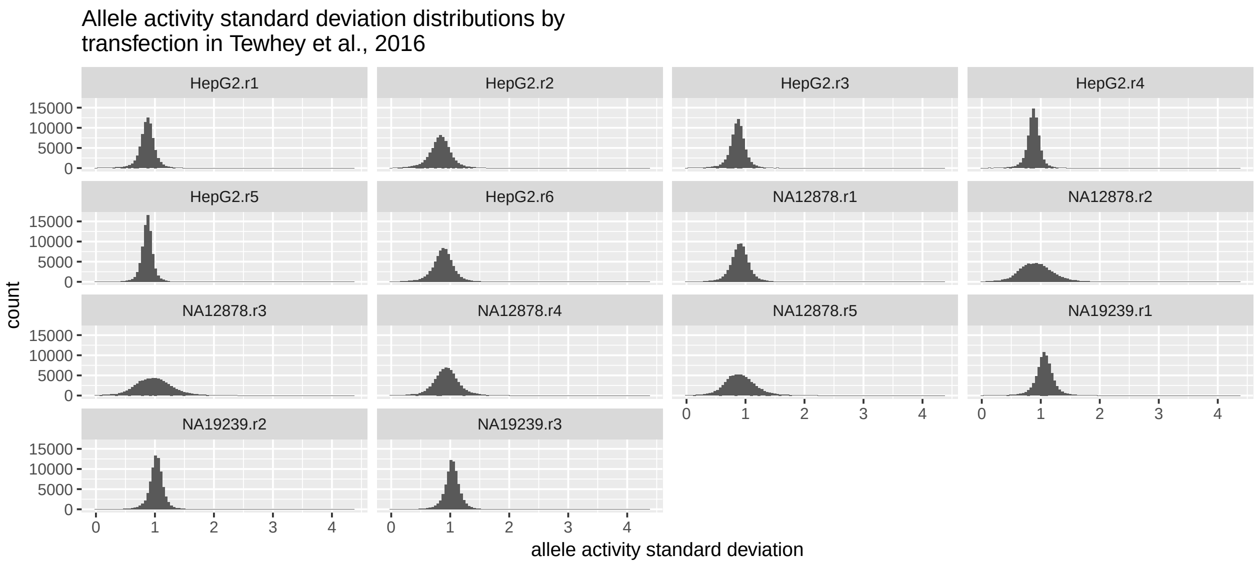
```
depthAdjDNAmeanCount %>%
  ggplot(aes(bcMean)) +
  geom_density(adjust = 2) +
  scale_x_log10() +
  geom_vline(xintercept = .06,
             lty = 2,
             color = 'grey60') +
  xlab('Depth Adjusted Mean Barcode Count in DNA samples') +
  ggtitle('Density plot of depth-adjusted DNA count mean\nacross DNA replicates in Tewhey et al., 2016'
```



Then, we compute the activity levels of each barcode by taking the depth adjusted count from an RNA sequencing run, dividing through the depth adjusted mean count from the DNA runs, then taking the log:

```r
library(parallel)
library(nortest) # for lillie.test()

# For a given data file
computeTransfectionStatistics = function(fileNum){
  # Get the file's depth normalization factor
  depthNum = fileDepths %>% filter(src == file_names[fileNum]) %>% .$fileDepth

  # Read in the counts
  rnaCounts = read_tsv(paste0(dir, file_names[fileNum]),
                       col_names = c('allele', 'barcode', 'count')) %>%
    mutate(depthAdjCount = 1e6*count/depthNum) #normalize them for depth

  # Then compute statistics for the file
  alleleStatistics = depthAdjDNAmeanCount %>%
    filter(bcMean > .06) %>% # this is where we introduce the cutoff
    left_join(rnaCounts, by = c('allele', 'barcode')) %>% #join onto DNA counts
    mutate(activity = log(depthAdjCount/bcMean)) %>% #compute activity
    group_by(allele) %>% #for each allele
    summarise(alleleMean = mean(activity, na.rm = TRUE), #compute statistics
              alleleSD = sd(activity, na.rm = TRUE),
              numBarcodes = sum(!is.na(count)),
              llliieforsP = ifelse(numBarcodes > 4,
                                   lillie.test(na.omit(activity))$p.value,
                                              NA)) %>%
     filter(numBarcodes > 1) %>% # take only alleles which had > 1 barcode
     mutate(file = file_names[fileNum]) # and add on a file identifier
    # The filter removes alleles that only had zero or one barcodes show up in
    # the RNA, for which a standard deviation is not meaningful

  return(alleleStatistics)
}

#apply the above function to the RNA data file_names
transfectionStatistics = mclapply(6:19,
                                  computeTransfectionStatistics,
                                  mc.cores = 6)

# Join the results together and clean up the file names
allTransfectionsStats = transfectionStatistics %>%
  reduce(bind_rows) %>%
  mutate(file = gsub('Geuv_90K_', '', file) %>% gsub('.tag.ct.indiv.expanded', '', .))
save(allTransfectionsStats,
     file = '~/designMPRA/outputs/tewheyAllTransfectionsStats.RData')
```

```
allTransfectionsStats %>%
  ggplot(aes(alleleSD)) +
  geom_histogram(bins = 100) +
  facet_wrap('file') +
  xlab('allele activity standard deviation') +
  ggtitle('Allele activity standard deviation distributions by\ntransfection in Tewhey et al., 2016')
```



There is variability in the distribution of allele activity standard deviations by transfection, but they are commonly above 1 (the mean of every allele in every transfection in this study is .926). Because activity is a log quantity, a standard deviation of 1 correpsonds to an average variability of $exp(1) \approx 2.7$-fold more mRNA molecules out per DNA molecule in within barcode replicates of the same allele.

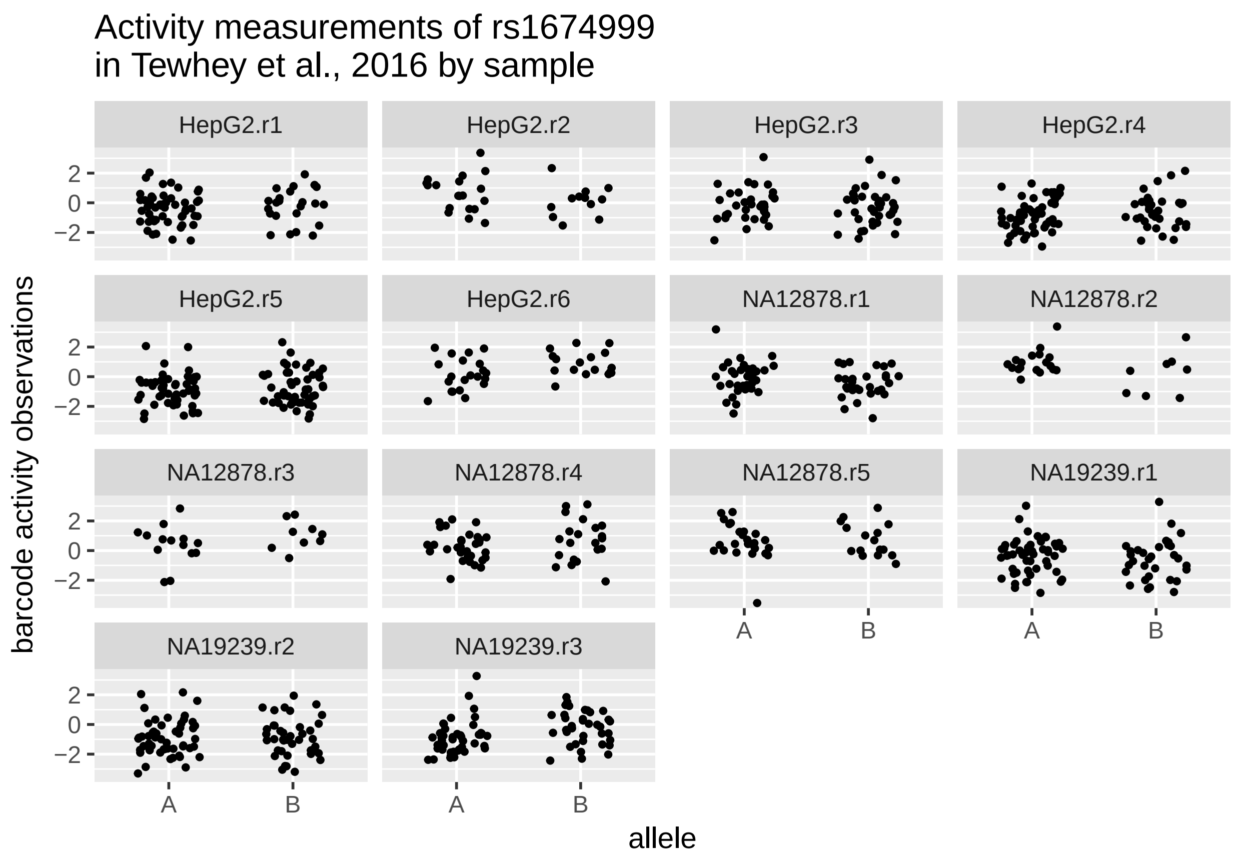
We can look at the activity distribution of the two alleles for a single SNP tested. We take `rs1674999` as an example. A random sample of the raw counts:

```r
# counts = map(1:19, ~read_tsv(paste0(dir, file_names[.x]),
#                               col_names = c('allele', 'barcode', 'count')) %>%
#                    mutate(src = file_names[.x])) %>%
#   reduce(bind_rows)
#
# rs1674999counts = counts %>% filter(grepl('rs1674999', allele))

load('~/designMPRA/outputs/tewheyrs1674999counts.RData')

rs1674999counts %>%
  set_names(c('Allele', 'Barcode', 'Count', 'Transfection File Name')) %>%
  sample_n(5) %>%
  kable
```

Allele	Barcode	Count	Transfection File Name
rs1674999_A	TGACCTTTAGTATTTGATCT	6	Geuv_90K_ctrl.r1.tag.ct.indiv.expanded
rs1674999_B	GATGAAGACATGTAGCCATA	7	Geuv_90K_ctrl.r4.tag.ct.indiv.expanded
rs1674999_B	TGTGTCGCCCATGGTCGTGC	3	Geuv_90K_NA19239.r2.tag.ct.indiv.expanded
rs1674999_B	TCGCGGTTGTACTAGGTGAC	34	Geuv_90K_NA12878.r5.tag.ct.indiv.expanded
rs1674999_A	AAAAGGGAAAGGGATAGTTG	1	Geuv_90K_HepG2.r1.tag.ct.indiv.expanded

And a plot of the activity levels of all barcodes in all samples:

```r
assureRNAandDNA = function(barcodeDat){
  # function for only including barcodes with DNA & RNA measurements
  any(grepl('ctrl',
           barcodeDat$src)) & any(grepl('HepG2|NA[0-9]{5}',
                                        barcodeDat$src))
}


rs1674999counts %>%
  group_by(barcode) %>%
  nest %>%
  filter(map_lgl(data, assureRNAandDNA)) %>%
  unnest %>%
  left_join(fileDepths, by = 'src') %>%
  mutate(depthAdjCount = count*1e6/fileDepth) %>% #adjust count for sample depth
  group_by(barcode) %>% nest %>%
  mutate(ctrlMean = map_dbl(data, # take mean count of control samples
                          ~filter(.x, grepl('ctrl', .x$src))$depthAdjCount %>% mean)) %>%
  unnest %>%
  filter(!grepl('ctrl', src)) %>%
  mutate(activity = log(depthAdjCount / ctrlMean)) %>% #compute activity
  mutate(replicate = gsub('Geuv_90K_', '', src) %>% gsub('.tag.ct.indiv.expanded', '', .),
         allele = gsub('rs1674999_', '', allele)) %>%
  ggplot(aes(allele, activity)) +
  geom_jitter(height = 0, width = .25, size = .8) +
  facet_wrap('replicate') +
  ylab('barcode activity observations') +
  ggtitle('Activity measurements of rs1674999\nin Tewhey et al., 2016 by sample')
```



This SNP does not seem to have a large transcriptional shift between the alleles (i.e. both alleles have roughly the same average activity), but one can see that this level of activity variance is large. The variance is close to 1.0 in most transfections so this SNP is a reasonably representative example of the variances observed throughout the experiment.

Detecting a low magnitude transcriptional shift (for example a 33% increase in mRNA per DNA corresponding to a TS of $log(1.33) = .285$) at 90% power at the significance level required to overcome multiple corrections would require aggregating an even larger number of data points than shown here. A power calculation with `pwr::pwr.t.test` shows this:

```
library(pwr)

pwr.t.test(d = .285,
           sig.level = .05/39479, # bonferroni correction for the number of pairs tested in the paper
           power = .9)
```

```
##
##      Two-sample t test power calculation
##
##               n = 930.0622
##               d = 0.285
##       sig.level = 1.266496e-06
##           power = 0.9
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

S2.2 - MPRA Activity Variance in Ulirsch et al., Cell 2016

Repeating the analysis with a different study. These data were acquired from the authors' website. The code differs slightly due to differences in how the data was provided by the authors, but the analysis is the same.

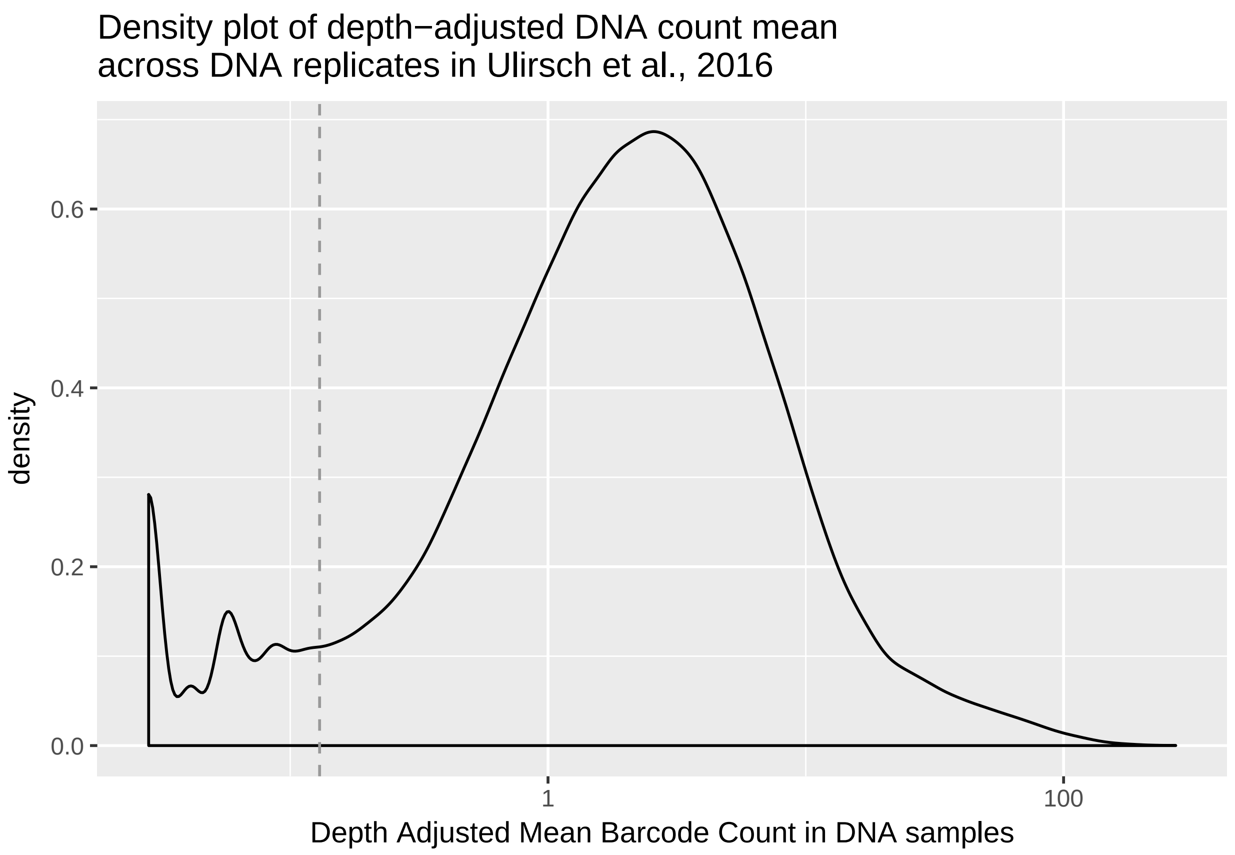
A few example barcode counts:

```r
dir = "/mnt/labhome/andrew/MPRA/paper_data/"

UMPRA = read_delim(file = paste0(dir, "Raw/", "RBC_MPRA_minP_raw.txt"),
                          delim = "\t",
                          col_names = T,
                          col_types = cols(chr = "c"))

UMPRA %>% # show a few example counts
  select(chr, pos, ref, alt, byallele, K562_minP_DNA1:K562_GATA1_minP_RNA4) %>%
  gather(key = src, value = count, K562_minP_DNA1:K562_GATA1_minP_RNA4) %>%
  sample_n(5) %>%
  set_names(c('Chrom', 'Position', 'Reference',
              'Alternate', 'Allele ID', 'Transfection', 'Count')) %>%
  kable
```

Chrom	Position	Reference	Alternate	Allele ID	Transfection	Count
12	2474661	C	T	12 2474661 1/2 Mut	K562_GATA1_minP_RNA3	251
17	42294462	A	G	17 42294462 1/2 Ref	K562_CTRL_minP_RNA6	1
1	158628014	T	G	1 158628014 1/2 Ref	K562_CTRL_minP_RNA3	38
6	32612079	T	C	6 32612079 1/3 Ref	K562_GATA1_minP_RNA3	3
1	203650945	C	T	1 203650945 1/3 Ref	K562_CTRL_minP_RNA1	1

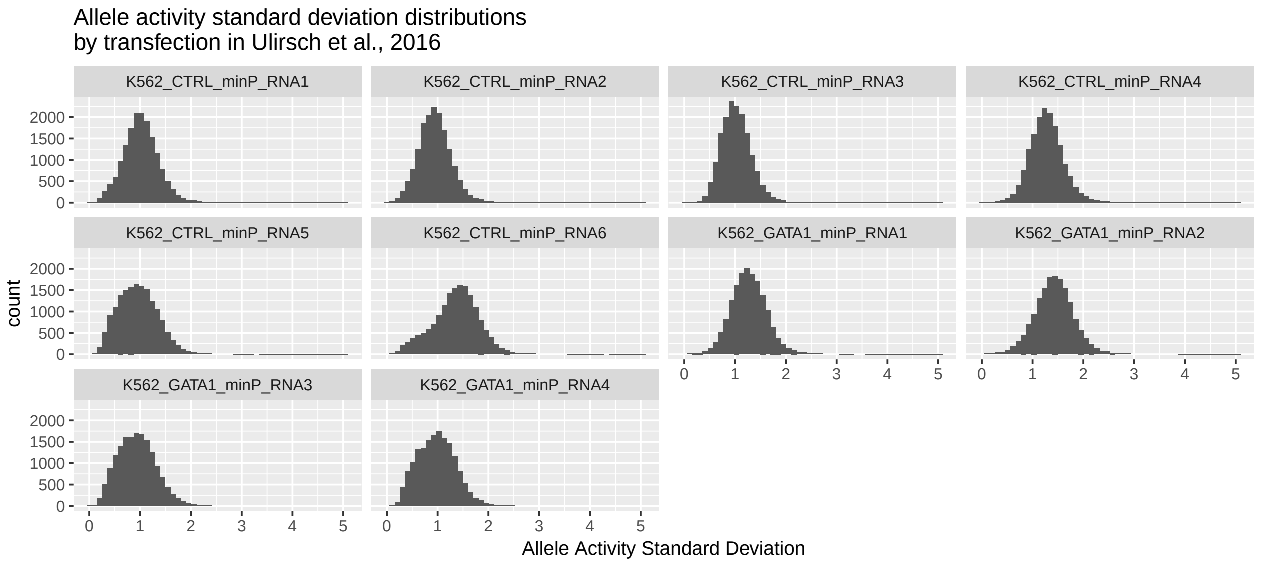
Here we'll use a mean depth-normalized DNA count of .13:

```r
depthNormalize = function(sampleCounts){
  sampleCounts*1e6/sum(sampleCounts)
}

UMPRA %>%
  mutate_at(vars(contains('K562')), depthNormalize) %>%
  mutate(dnaMean = (K562_minP_DNA1 + K562_minP_DNA2)/2) %>%
  ggplot(aes(dnaMean)) +
  geom_density() +
  scale_x_log10() +
  geom_vline(xintercept = .13,
             lty = 2,
             color = 'grey60') +
  ggtitle('Density plot of depth-adjusted DNA count mean\nacross DNA replicates in Ulirsch et al., 2016
  xlab('Depth Adjusted Mean Barcode Count in DNA samples')
```



And here are the distributions of allele activity standard deviations by sample:

```
activitySDs = UMPRA %>%
  mutate_at(vars(contains('K562')), depthNormalize) %>%
  mutate(dnaMean = (K562_minP_DNA1 + K562_minP_DNA2)/2) %>%
  filter(dnaMean > .13) %>%
  select(-K562_minP_DNA1, -K562_minP_DNA2) %>%
  gather(sample, depthAdjCount, K562_CTRL_minP_RNA1:K562_GATA1_minP_RNA4) %>%
  mutate(activity = log(depthAdjCount / dnaMean)) %>%
  group_by(byallele, sample) %>%
  summarise(alleleSD = sd(activity[is.finite(activity)], na.rm = TRUE)) %>%
  ungroup
# the use of is.finite() removes barcodes with 0 RNA counts (-Inf activity)

activitySDs %>%
  ggplot(aes(alleleSD)) +
  geom_histogram(bins = 50) +
  facet_wrap('sample') +
  xlab('Allele Activity Standard Deviation') +
  ggtitle('Allele activity standard deviation distributions\nby transfection in Ulirsch et al., 2016')
```



Again, this plot shows that for a randomly chosen single allele, the standard deviation of its activity measurements are commonly around or above 1. The mean activity standard deviation across all samples in this study was 1.1221643.

If we combine both studies, we can see the typical range of activity standard deviations. The third output exponentiates the standard deviations, showing the corresponding standard fold-change in mRNA out to DNA in.

```
bothStudies = c(allTransfectionsStats$alleleSD, #Tewhey et al.
                activitySDs$alleleSD) #Ulirsch et al.

# Mean activity standard deviation in both studies:
bothStudies %>%
  mean(na.rm = TRUE)
```

```
## [1] 0.9523771
```

```
# Exponentiated
bothStudies %>%
  mean(na.rm = TRUE) %>%
  exp
```

```
## [1] 2.591864
```

```
# Central 95% interval of activity standard deviations in both studies:
bothStudies %>%
  na.omit %>%
  quantile(c(.025, .975))
```

```
##      2.5%     97.5%
## 0.4531956 1.5874324
```

```
# Exponentiated
bothStudies %>%
  na.omit %>%
  quantile(c(.025, .975)) %>%
  exp
```

```
##     2.5%    97.5%
## 1.573332 4.891174
```

S3 - Activity Normality Assumption

The "Power" tab of the application uses a t-test to estimate the power to detect functional variants with given activity variance across a range of transcriptional shifts. Therefore the validity of the underlying normality is an important consideration. We performed a number of analyses including normal Q-Q plots, Lilliefors tests, and Monte Carlo simulations to examine the impact of the normality assumption on our power calculations. In this section we show that a t-test is a reasonable approximation for modelling MPRA data.

S3.1 - Normal Q-Q plots

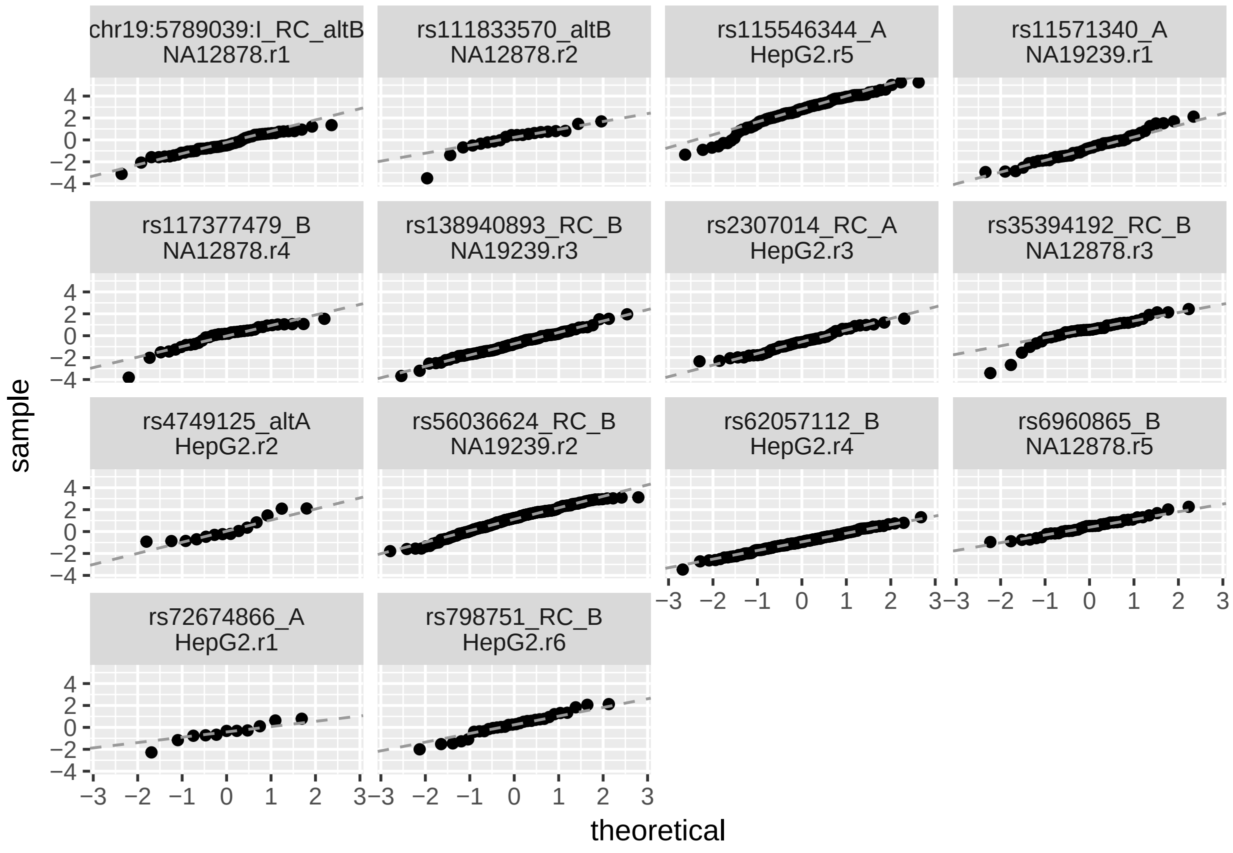
We can use a normal QQ plot to visually inspect the normality of allele activity levels.

S3.1.1 - Normal Q-Q plots of randomly chosen alleles

We will do this for a set of randomly chosen alleles from each sample in Tewhey et al., 2016.

```r
get_random_allele = function(fileNum){
  dir = '/mnt/bigData2/andrew/MPRA/Tewhey/indivTags/'

  # Get the file's depth normalization factor
  depthNum = fileDepths %>% filter(src == file_names[fileNum]) %>% .$fileDepth

  # Read in the counts
  rnaCounts = read_tsv(paste0(dir, file_names[fileNum]),
                       col_names = c('allele', 'barcode', 'count')) %>%
    mutate(depthAdjCount = 1e6*count/depthNum) #normalize them for depth

  # return a data_frame with the activity measurements for one allele
  depthAdjDNAmeanCount %>%
    filter(bcMean > .06) %>% # this is where we introduce the cutoff
    left_join(rnaCounts, by = c('allele', 'barcode')) %>% #join onto DNA counts
    na.omit %>%
    filter(allele == sample(.$allele, 1)) %>%
    mutate(activity = log(depthAdjCount/bcMean),
           file_name = file_names[fileNum]) # and add on a file identifier
}

set.seed(123456)

random_alleles = mclapply(6:19, get_random_allele, mc.cores = 6) %>%
  reduce(rbind) %>%
  as_tibble %>%
  mutate(file_name = gsub('Geuv_90K_', '', file_name) %>% gsub('.tag.ct.indiv.expanded', '', .)) %>%
  unite(allele_sample, allele, file_name, sep = '\n')

qq_lines = random_alleles %>% # http://mgimond.github.io/ES218/Week06a.html
  group_by(allele_sample) %>%
  summarise(act25 = quantile(activity,.25),
            act75 = quantile(activity, .75),
            norm25 = qnorm(.25),
            norm75 = qnorm(.75),
            qq_slope = (act25 - act75) / (norm25 - norm75),
            qq_int = act25 - qq_slope * norm25)
```

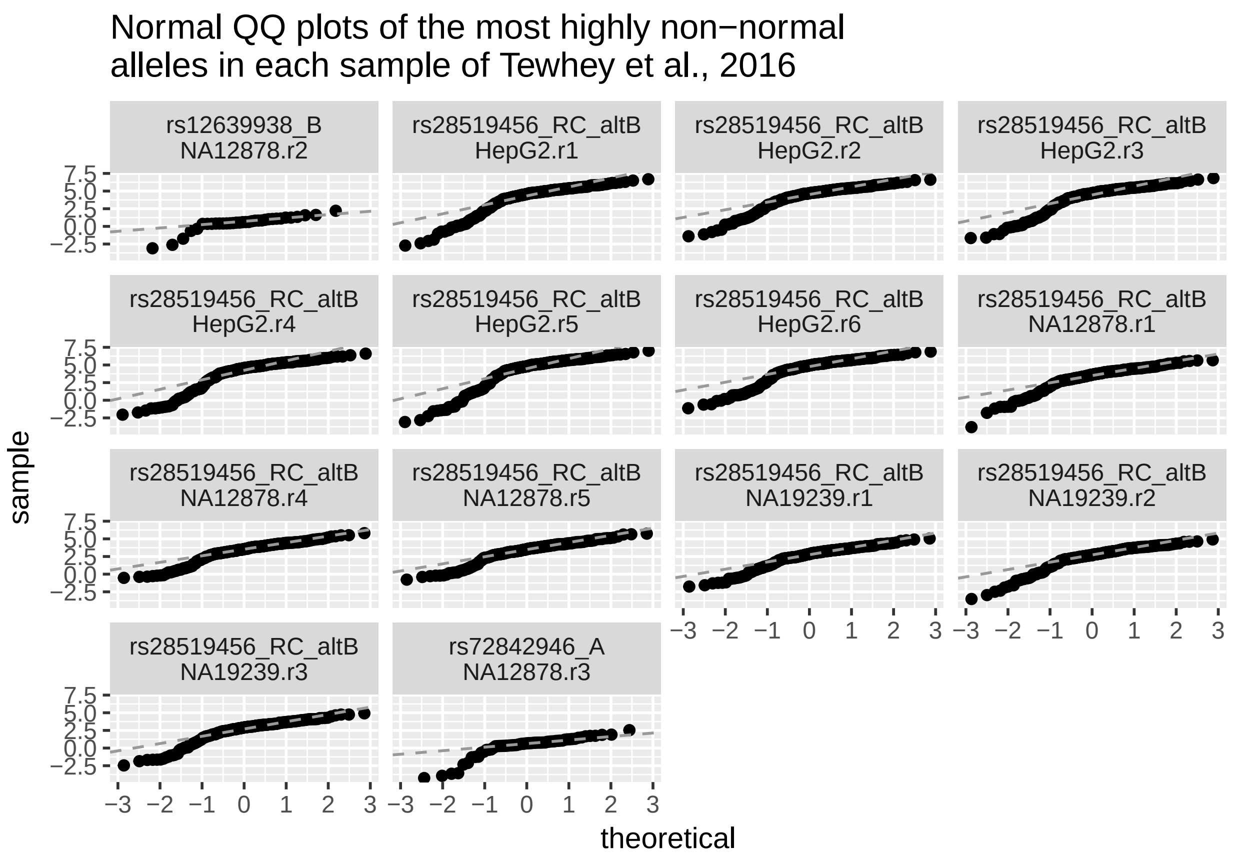
```r
random_alleles %>%
  ggplot() +
  stat_qq(aes(sample = activity)) +
  facet_wrap('allele_sample') +
  geom_abline(aes(slope = qq_slope, intercept = qq_int),
              data = qq_lines,
              color = 'grey60', lty = 2)
```



S3.1.2 Normal Q-Q plots of highly non-normal alleles

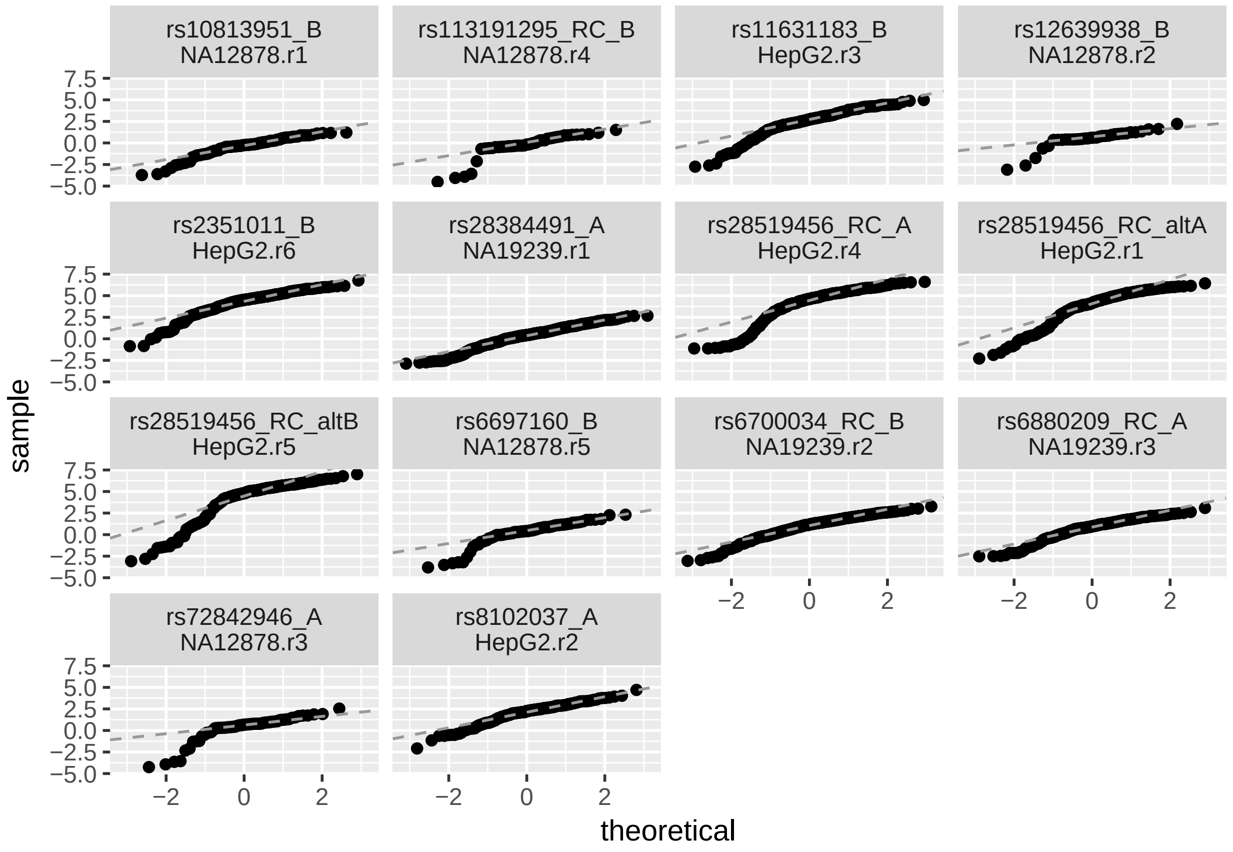
We can also do this to inspect the most non-normal alleles (see S3.2 to see how we define non-normality in terms of low Lilliefors test p-values) from each sample.

```r
get_nonnormal_allele = function(file_allele_list){
  dir = '/mnt/bigData2/andrew/MPRA/Tewhey/indivTags/'

  file_name = file_allele_list[['file_name']]
  file_name_str = file_name
  nonnormal_allele = file_allele_list[['allele']]

  depthNum = fileDepths %>% filter(src == paste0('Geuv_90K_', file_name, '.tag.ct.indiv.expanded')) %>%

  rnaCounts = read_tsv(paste0(dir, 'Geuv_90K_', file_name, '.tag.ct.indiv.expanded'),
                       col_names = c('allele', 'barcode', 'count')) %>%
    mutate(depthAdjCount = 1e6*count/depthNum) #normalize them for depth
  # return a data_frame with the activity measurements for one allele
  depthAdjDNAmeanCount %>%
    filter(bcMean > .06) %>% # this is where we introduce the cutoff
    left_join(rnaCounts, by = c('allele', 'barcode')) %>% #join onto DNA counts
    na.omit %>%
    filter(allele == nonnormal_allele) %>%
    mutate(activity = log(depthAdjCount/bcMean),
           file_name = file_name_str) # and add on a file identifier
   # The filter removes alleles that only had zero or one barcodes show up in
   # the RNA, for which a standard deviation is not meaningful
}

nonnormal_alleles = allTransfectionsStats %>%
  arrange(lillieforsP) %>%
  filter(!duplicated(file)) %$%
  map2(file, allele, ~list(file_name = .x, allele = .y)) %>%
  mclapply(get_nonnormal_allele, mc.cores = 6) %>%
  reduce(rbind) %>%
  as_tibble %>%
  mutate(file_name = gsub('Geuv_90K_', '', file_name) %>% gsub('.tag.ct.indiv.expanded', '', .)) %>%
  unite(allele_sample, allele, file_name, sep = '\n')

qq_lines = nonnormal_alleles %>% # http://mgimond.github.io/ES218/Week06a.html
  group_by(allele_sample) %>%
  summarise(act25 = quantile(activity,.25),
            act75 = quantile(activity, .75),
            norm25 = qnorm(.25),
            norm75 = qnorm(.75),
            qq_slope = (act25 - act75) / (norm25 - norm75),
            qq_int = act25 - qq_slope * norm25)

nonnormal_alleles %>%
  ggplot() +
  stat_qq(aes(sample = activity)) +
  facet_wrap('allele_sample') +
  geom_abline(aes(slope = qq_slope, intercept = qq_int), data = qq_lines, color = 'grey60', lty = 2) +
  ggtitle('Normal QQ plots of the most highly non-normal\nalleles in each sample of Tewhey et al., 2016
```



These plots show that the most highly non-normal alleles generally feature heavy tails on the left side of the distribution. We noticed that the most highly non-normal allele in each sample is usually rs28519456_RC_altB so we repeat this process while filtering out any duplicate alleles:

```
nonnormal_alleles = allTransfectionsStats %>%
  arrange(lillieforsP) %>%
  filter(!duplicated(allele)) %>%
  filter(!duplicated(file)) %$%
  map2(file, allele, ~list(file_name = .x, allele = .y)) %>%
  mclapply(get_nonnormal_allele, mc.cores = 6) %>%
  reduce(rbind) %>%
  as_tibble %>%
  mutate(file_name = gsub('Geuv_90K_', '', file_name) %>% gsub('.tag.ct.indiv.expanded', '', .)) %>%
  unite(allele_sample, allele, file_name, sep = '\n')

qq_lines = nonnormal_alleles %>% # http://mgimond.github.io/ES218/Week06a.html
  group_by(allele_sample) %>%
  summarise(act25 = quantile(activity,.25),
            act75 = quantile(activity, .75),
            norm25 = qnorm(.25),
            norm75 = qnorm(.75),
            qq_slope = (act25 - act75) / (norm25 - norm75),
            qq_int = act25 - qq_slope * norm25)

nonnormal_alleles %>%
  ggplot() +
  stat_qq(aes(sample = activity)) +
  facet_wrap('allele_sample') +
  geom_abline(aes(slope = qq_slope, intercept = qq_int), data = qq_lines, color = 'grey60', lty = 2)
```



Generally the most highly non-normal alleles show most of their data falling in a roughly normal pattern and the rest going into a heavy negative tail. This pattern is also apparent in most of the highly non-normal alleles from Ulirsch et al., 2016 (analysis not shown), however it is less obvious given that study's lower number of barcodes per allele. See section S3.3 for a monte carlo simulation that examines the effect of this pattern on the results of a t-test.

S3.2 - Lilliefors tests

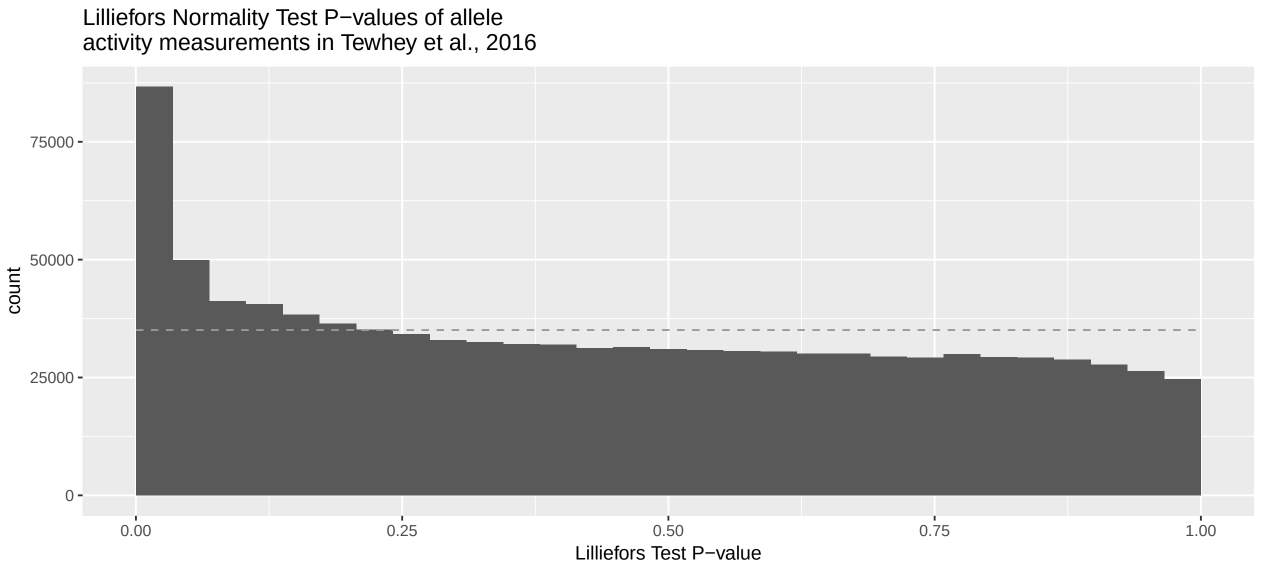
We can test every allele in a systematic, quantitative way through the use of Lilliefors tests.

In the earlier section S2.1 that describes the activity variance in MPRA assays we also calculated the p-value of a Lilliefors test for each allele. A Lilliefors test is a modified Kolmogorov-Smirnov test that tests if the data come from a normal distribution with unspecified mean and variance. A low p-value from a Lilliefors test suggests that the data come from a non-normal distribution.

S3.2.1 - Lilliefors p-value distribution

Looking at the distribution of these p-values will tell us how commonly the normality assumption of our t-test holds. If all samples come from normal distributions, the p-values should follow a $Unif(0,1)$ distribution (shown here scaled to this number of allele observations as a grey horizontal line).

```
segment = data_frame(x1 = 0,
                     x2 = 1,
                     y1 = nrow(allTransfectionsStats) / 30,
                     y2 = nrow(allTransfectionsStats) / 30)

allTransfectionsStats %>%
  na.omit() %>% #we returned NA for alleles with <4 observations
  ggplot(aes(lillieforsP)) +
  geom_histogram(breaks = seq(0,1,length.out = 30)) +
  xlab('Lilliefors Test P-value') +
  geom_segment(aes(x = x1,
                   xend = x2,
                   y = y1,
                   yend = y2),
               data = segment,
               lty = 2,
               color = 'grey60') +
  ggtitle('Lilliefors Normality Test P-values of allele\nactivity measurements in Tewhey et al., 2016')
```

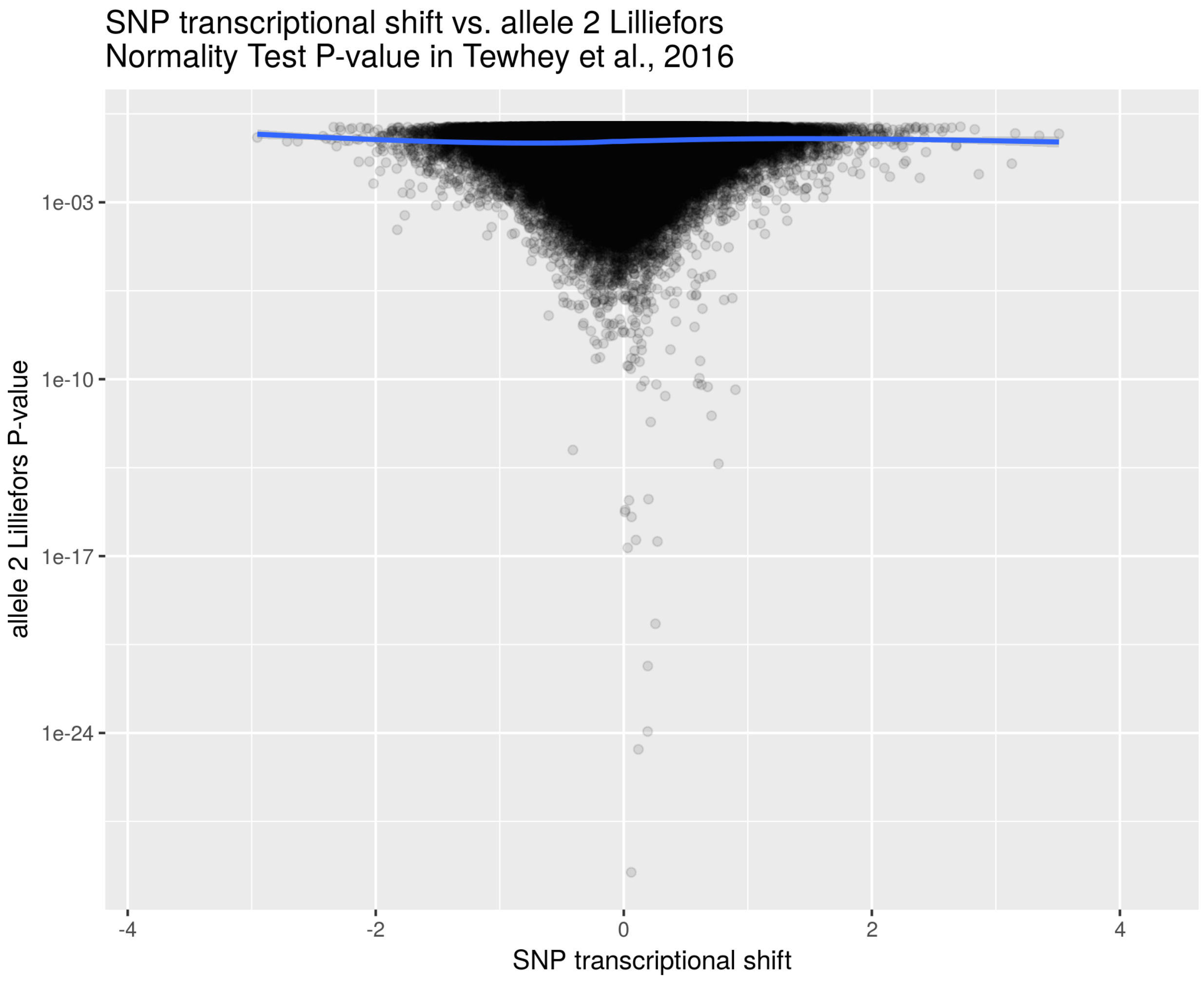


While a subset of the variants from Tewhey et al. clearly skew from the $Unif(0,1)$ distribution one would expect from normally distributed samples, the fraction of alleles that defy the expected pattern is relatively small. This suggests that a t-test should usually provide a reasonable approximation of the true power.

S3.2.2 - Lilliefors p-value and Transcriptional Shift relationship

There is no indication that SNPs with large transcriptional shifts (i.e. those one aims to detect with the assay) systematically defy the normality assumption. The following plot shows this with the observed transcriptional shift on the x-axis and p-value of a Lilliefors test on the activity measurements of the second allele on the y-axis (the plot for the first allele looks very similar). If SNPs with large transcriptional effects systematically defied the normality assumptions, the SNPs on the left and right sides would show the lowest p-values, which is not observed:

```
shiftNonNormality = allTransfectionsStats %>%
  separate(allele,
           into = c('SNP', 'allele'),
           sep = '_(?=[AB]$)|_(?=alt[AB]$)') %>%
  separate(allele,
           into = c('alt', 'allele'),
           sep = '(?=[AB]$)') %>%
  unite(SNP, SNP:alt) %>%
  mutate(SNP = gsub('_$', '', SNP)) %>%
  unite(SNPfile, SNP, file) %>%
  group_by(SNPfile) %>%
  summarise(numAlleles = n(),
            TS = ifelse(numAlleles == 2, alleleMean[2] - alleleMean[1], NA),
            allele1Lilliefors = lillieforsP[1],
            allele2lilliefors = lillieforsP[2])

shiftNonNormality %>%
  ggplot(aes(TS, allele2lilliefors)) +
  geom_point(alpha = .1) +
  scale_y_log10() +
  xlab('SNP transcriptional shift') +
  ylab('allele 2 Lilliefors P-value') +
  ggtitle('SNP transcriptional shift vs. allele 2 Lilliefors\nNormality Test P-value in Tewhey et al., 2
  geom_smooth()
```

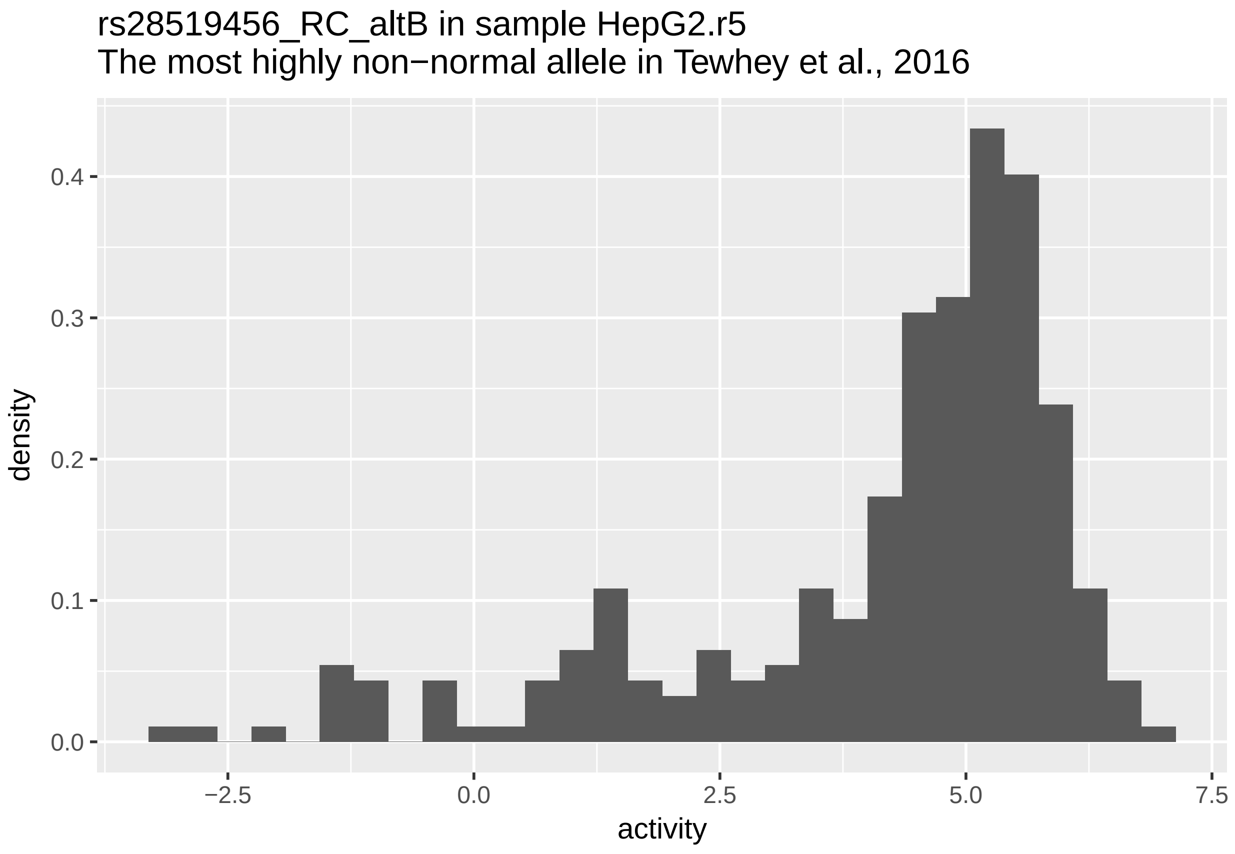


S3.3 Monte Carlo t-test simulation

S3.3.1 Non-normal allele

Given the common shape of the highly non-normal alleles seen in S3.1, we show here through Monte Carlo simulation that this non-normality does not significantly impact the outcome of a t-test. First we look at the activity measurements of the most highly non-normal allele in the entirety of Tewhey et al., 2016: allele `rs28519456_RC_altB` in the sample `HepG2.r5`.

```
library(pwr)
library(parallel)

worst = list(allele = 'rs28519456_RC_altB', file_name = 'HepG2.r5') %>%
  get_nonnormal_allele()

worst %>%
  ggplot(aes(x = activity)) +
  geom_histogram(aes(y = ..density..), bins = 30) +
  ggtitle('rs28519456_RC_altB in sample HepG2.r5\nThe most highly non-normal allele in Tewhey et al., 2
```

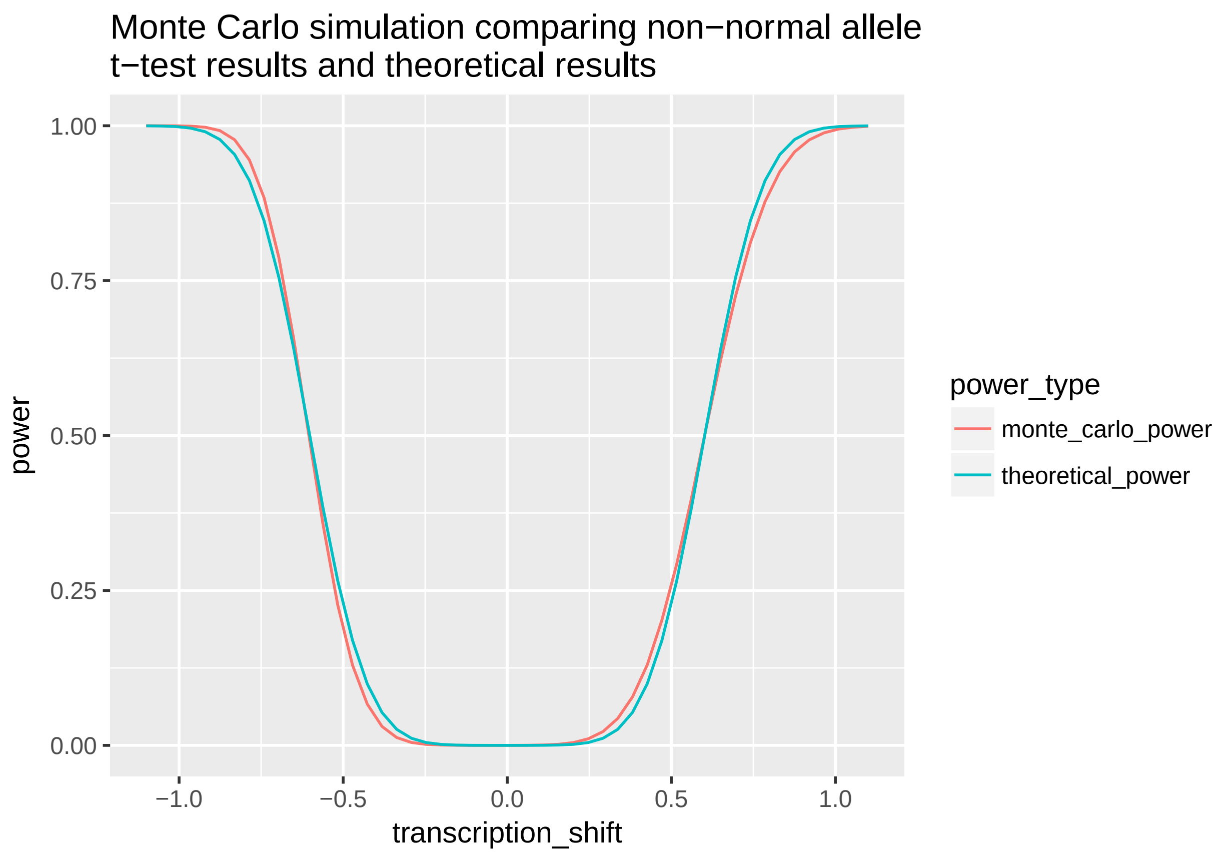


By re-sampling from this distribution, we can compute Monte Carlo simulations of the average power of a t-test to detect a difference between this distribution and a "typical" reference allele (we use samples from a normal distribution with a standard deviation of .926) at $\alpha = 10^{-5}$. We can compare the simulated power to the theoretical power of the situation where both alleles were drawn from normal distributions.

```
worst_mean = mean(worst$activity)

# used to calculate effect size
pooled_SD = sqrt(((nrow(worst) - 1)*sd(worst$activity)**2 + (nrow(worst) - 1)*.926**2) /
```

```r
                   (nrow(worst)*2 - 2))

simulate_power = function(transcription_shift, sig_level = .05, n_sim = 10000){

  #simulate a t-test many times
  monte_carlo_p_values = sapply(1:n_sim, function(x){

    # randomly sample the most non-normal allele
    non_normal_samples = sample(worst$activity, size = nrow(worst), replace = TRUE)

    # randomly sample a "typical" allele of the same size
    normal_samples = rnorm(nrow(worst),
                           mean = worst_mean - transcription_shift,
                           sd = .926) # using the average SD from Ulirsch and Tewhey together

    # compute the p-value of a t-test
    t.test(non_normal_samples, normal_samples)$p.value
  })

  # return the fraction that pass the input significance level
  sum(monte_carlo_p_values < sig_level) / n_sim
}

sim_power = data_frame(transcription_shift = seq(-1.1, 1.1, length.out = 50),
                       theoretical_power = pwr.t.test(n = nrow(worst),
                                                      d = abs(transcription_shift) / pooled_SD,
                                                      type = 'two.sample',
                                                      alternative = 'two.sided',
                                                      sig.level = 1e-5)$power,
                       monte_carlo_power = mclapply(transcription_shift,
                                                    simulate_power,
                                                    mc.cores = 6,
                                                    sig_level = 1e-5, n_sim = 3e6) %>% unlist)
```

```r
load('~/designMPRA/outputs/sim_power_p_1e-5_n_3e6.RData')

sim_power %>%
  gather(power_type, power, -transcription_shift) %>%
  ggplot(aes(transcription_shift, power, color = power_type)) +
  geom_path() +
  scale_y_continuous(breaks = seq(0, 1, by = .25),
                     limits = c(0, 1)) +
  ggtitle('Monte Carlo simulation comparing non-normal allele\nt-test results and theoretical results')
```



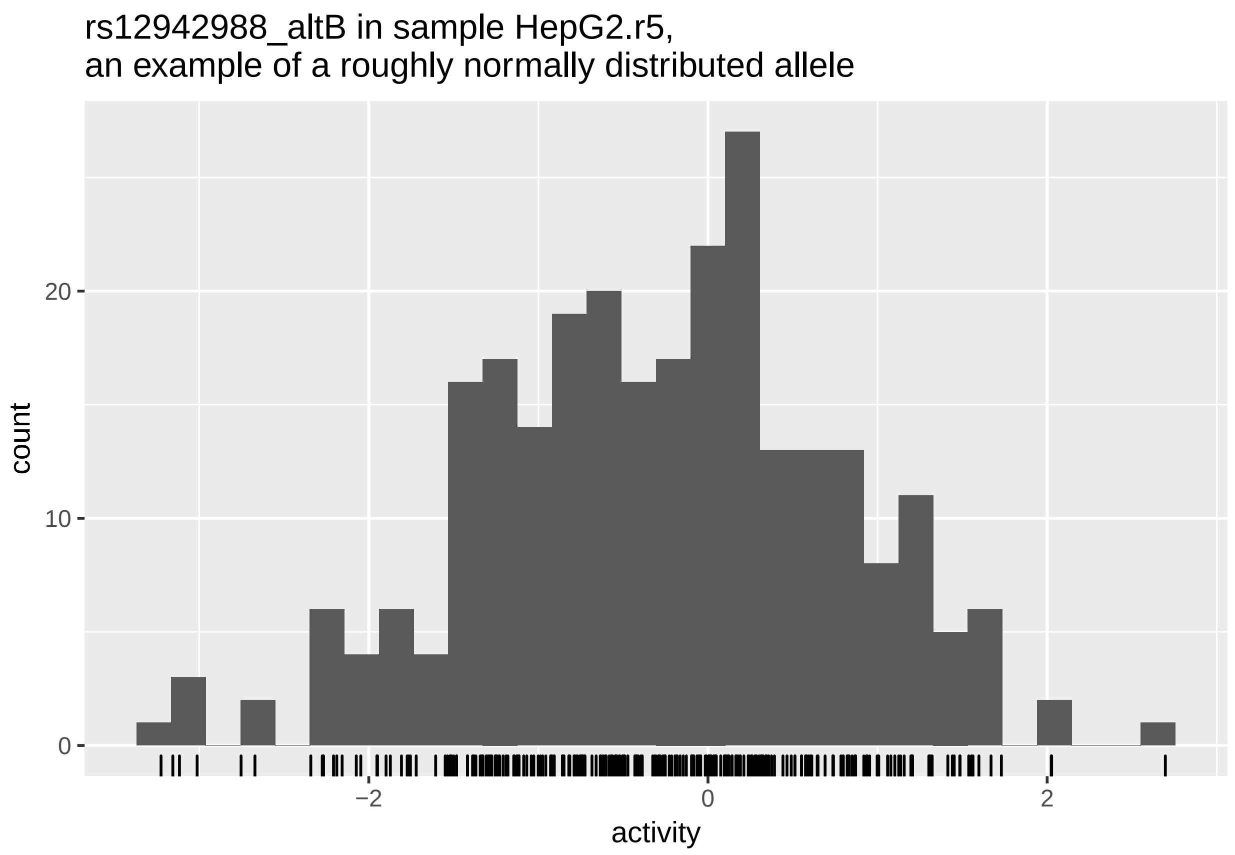
The two curves are very close, showing a maximum absolute difference between theoretical and simulated power of 0.0400.

S3.3.2 Normal allele

For the sake of comparison, we repeated the process using an allele from the same transfection whose activity levels looked convincingly normally distributed: `rs12942988_altB`. We chose this allele because it had the same number of barcode allele measurements as `rs28519456_RC_altB` and it's Lilliefors test p-value was 0.782.

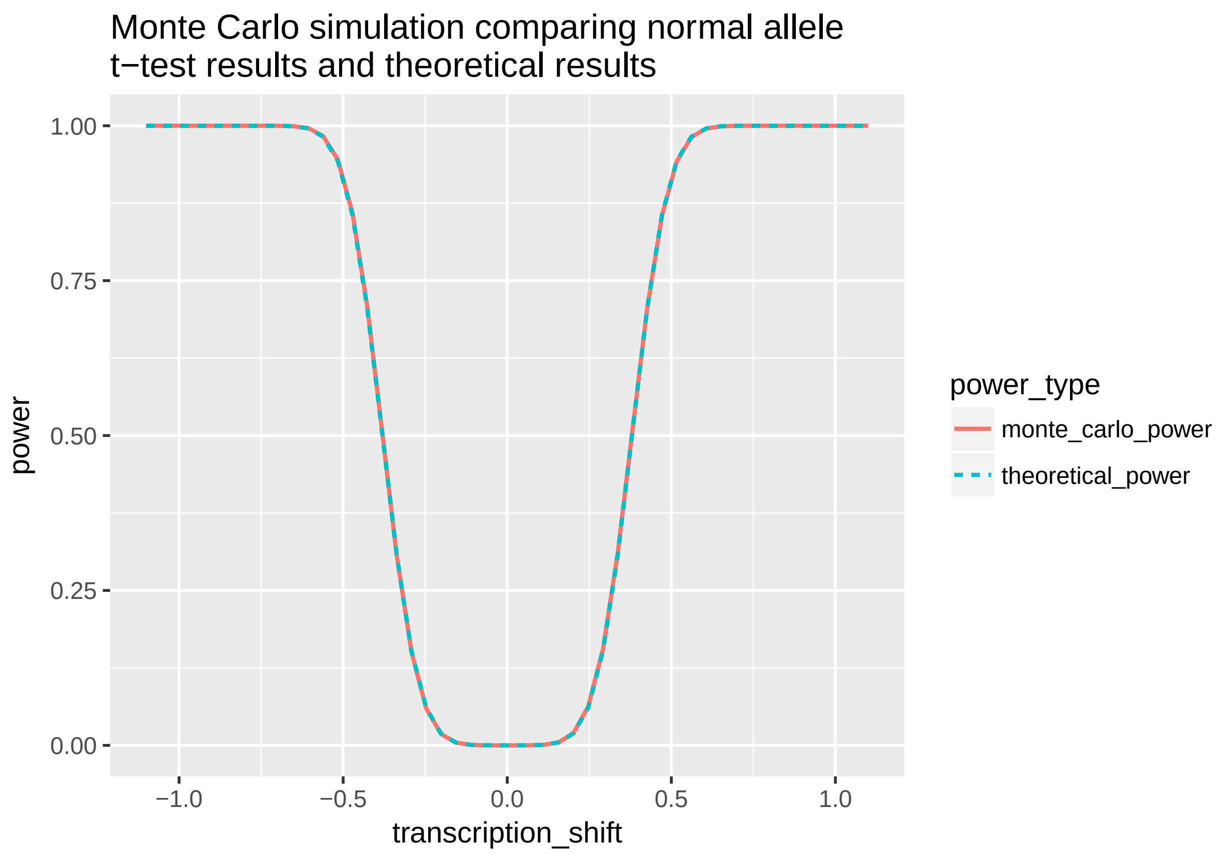
```r
good = list(allele = 'rs12942988_altB', file_name = 'HepG2.r5') %>%
  get_nonnormal_allele()
```

```r
good %>%
  ggplot(aes(activity)) +
  geom_histogram(bins = 30) +
  geom_rug() +
  labs(title = 'rs12942988_altB in sample HepG2.r5,\nan example of a roughly normally distributed allel
```



```
good_mean = mean(good$activity)

# used to calculate effect size
pooled_SD = sqrt(((nrow(good) - 1)*sd(good$activity)**2 + (nrow(good) - 1)*.926**2) /
                   (nrow(good)*2 - 2))

sim_power = data_frame(transcription_shift = seq(-1.1, 1.1, length.out = 50),
                       theoretical_power = pwr.t.test(n = nrow(good),
                                                      d = abs(transcription_shift) / pooled_SD,
                                                      type = 'two.sample',
                                                      alternative = 'two.sided',
                                                      sig.level = 1e-5)$power,
                       monte_carlo_power = mclapply(transcription_shift,
                                                    simulate_power,
                                                    mc.cores = 6,
                                                    sig_level = 1e-5, n_sim = 3e6) %>% unlist)
```

```
load('~/designMPRA/outputs/sim_power_rs12942988_altB.RData')
sim_power %>%
  gather(power_type, power, -transcription_shift) %>%
  ggplot(aes(transcription_shift, power, color = power_type)) +
  geom_path(aes(lty = power_type), lwd = .75) +
  scale_y_continuous(breaks = seq(0, 1, by = .25),
                     limits = c(0, 1)) +
  ggtitle('Monte Carlo simulation comparing normal allele\nt-test results and theoretical results')
```



As expected, the simulated power almost exactly matches the theoretical power, showing a maximum absolute difference of 0.00341.

S3.4 Activity Normality Assumption - Conclusions

- The distributions of a majority of alleles are indistinguishable from normal distributions (S3.1 and S3.2).
- Those that are highly non-normal are non-normal in a way that does not significantly impact the outcome of the t-test in terms of statistical power (S3.3).
- Non-normality does not correlate with high transcriptional shift, meaning that the rare violations of the normality assumption would only be impactful on SNPs that are uninteresting in the first place (S3.2).

Furthermore we must remember that t-tests are generally considered robust against minor violations of the normality assumption (Lehmann 1986).

All of this together suggests it would be reasonable to have a t-test underlie the power calculations. Researchers cannot know the activity variance their experimental setups will achieve nor the true transcriptional shifts of their variants *a priori* in any case. The "Power" tab of the application is meant to use a few assumptions in order to provide approximate power estimates that researchers can use as rough guidelines for their experiments.

S4 - Power calculations

The power calculations are done with `pwr.t.test` from the R `pwr` package using the following R code:

```r
tibble::data_frame(meanDiff = seq(0,5, by = .05),
                   pwr = pwr.t.test(n = input$nbarcode*input$nBlock,
                                    d = meanDiff / input$sigma,
                                    sig.level = input$alpha / input$nsnp)$power)
```

This returns the power to detect a transcriptional shift across a range from zero to five using user inputs (denoted `input$inputName`) on:
* the number of barcodes per allele
* the number of transfection replicates (blocks)
* the number of variants being tested (used for correcting the significance level to account for multiple testing)
* the desired significance level
* the assumed variance of activity measurements

This data frame is then plotted.

S5 - Barcode design

We generated the set of all possible DNA 12-mers then screened these according to the design parameters in Melnikov et al., Nature Biotechnology 2012 intended to assure that the barcodes are inert. These involve the following parameters:
* each nucleotide occurs at least once
* no runs of single nucleotides greater than length 3
* do not contain restriction sites for KpnI/XbaI
+ do not start with `TCT` (this creates a restriction site with XbaI with the preceding sequence)
* they do not match any human miR seed sequences

This was done with the `generate12mers.R` script in the package (copied below; not run in the generation of this document).

```r
library(tcR)
library(Biostrings)
nucruns = vector(mode = 'character', length = 4) %>% DNAStringSet
ni = 1
for (i in 4) {
  for (j in c('A', 'G', 'T', 'C')) {
    nucruns[ni] = rep(j, i) %>% paste(collapse = '') %>% DNAStringSet
    ni = ni + 1
  }
}

twelvemers = generate.kmers(12) %>% DNAStringSet
cat(paste0('done generating 12mers at ', Sys.time()))

#tmp = twelvemers[sample.int(length(twelvemers), size = 10)]

#Each nucleotide occurs at least one
missingone = apply(alphabetFrequency(twelvemers)[,1:4], 1, function(x){any(x == 0)})
twelvemers = twelvemers[!missingone]
cat(paste0('done removing twelvemers missing a nucleotide at ', Sys.time()))


#Cut out those with nucleotide runs of 4 or more in a row
hasnucruns = vcountPDict(nucruns, twelvemers) %>% colSums
hasnucruns = hasnucruns > 0
```

```r
twelvemers = twelvemers[!hasnucruns]
cat(paste0('done removing 12mers with runs of 4 or more at ', Sys.time()))

#Cut out those that start with TCT (creates an alternative digestion site for XbaI)
tctStart = subseq(twelvemers, 1, 3) == DNAString('TCT')
twelvemers = twelvemers[!tctStart]
cat(paste0('done removing 12mers starting with TCT at ', Sys.time()))

#Cut out those that match the miRNA seed sequences
#For now let's just use the human ones since there are fewer and it won't take as long
#isolated species names with cat mature.fa | grep '>' | cut -f 3,4 -d \  > mirBaseSpecies.txt
species = read.table('~/plateletMPRA/mirBaseSpecies.txt') %>%
  as.tbl %>%
  transmute(name = paste(V1 %>% as.character, V2 %>% as.character)) #%>%
  #filter(!duplicated(name))
human = grepl('Homo sapiens', species$name)

allSeeds = readRNAStringSet('~/plateletMPRA/mature.fa')

seedSeqs = allSeeds %>% subseq(2,7) %>% DNAStringSet #%>% unique
humanSeedSeqs = seedSeqs[human] %>% unique
seedSeqs = allSeeds %>% subseq(2,7) %>% DNAStringSet %>% unique

haveSeedlist = vwhichPDict(humanSeedSeqs, twelvemers) #this takes ~40 minutes. All seeds takes ~1h45m
save(list = c('twelvemers', 'haveSeedlist', 'humanSeedSeqs'),
     file = '~/designMPRA/outputs/haveHumanRNAiSeeds.RData')
haveSeed = sapply(haveSeedlist, function(x){length(x) > 0})

twelvemers = twelvemers[!haveSeed]
cat(paste0('done removing those with mirSeeds at ', Sys.time()))

print(length(twelvemers))
save(twelvemers, file = '~/designMPRA/outputs/inertTwelveMers.RData')
```

S6 - References

1. Melnikov, A., Murugan, A., Zhang, X., Tesileanu, T., Wang, L., Rogov, P., … Mik-kelsen, T. S. (2012). Systematic dissection and optimization of inducible enhancers in human cells using a massively parallel reporter assay. Nature Bio-technology, 30(3), 271–7.

2. Tewhey, R., Kotliar, D., Park, D. S., Liu, B., Winnicki, S., Reilly, S. K., … Sabeti, P. C. (2016). Direct Identification of Hundreds of Expression-Modulating Variants using a Multiplexed Reporter Assay. Cell, 165(6), 1519–1529.

3. Ulirsch, J. C., Nandakumar, S. K., Wang, L., Giani, F. C., Zhang, X., Rogov, P., … Sankaran, V. G. (2016). Systematic Functional Dissection of Common Genetic Variation Affecting Red Blood Cell Traits. Cell, 165(6), 1530–1545.

4. Lehmann, E. L. (1986). Unbiasedness: Applications to Normal Distributions; Confidence Intervals. In *Testing Statistical Hypotheses* (2nd ed., pp. 203-206). Wiley.