CAGExploreR Vignette

Emmanuel Dimont

October 10, 2013

Contents

1 Simple Workflow Example

1.1 Installation

To install **CAGExploreR**, you need to have R version 3.0.1 or newer (`http://www.r-project.org/`). You can download the package from `https://github.com/edimont/CAGExploreR/` into a local directory (the file is called **CAGExploreR_1.0.tar.gz**). There is no need to unpack or decompress the contents of this file. Then open R and run the following code (you only have to do this once). Since **CAGExploreR** depends on several other packages, you will need to install them first.

```
source("http://bioconductor.org/biocLite.R")
biocLite(c("GenomeGraphs", "Rsamtools", "GenomicFeatures"))
install.packages(c("gridBase", "R2HTML", "data.table"))
```

Once the dependencies are installed, we can now install the package from source:

```
install.packages("CAGExploreR_1.0.tar.gz", type = "source")
```

1.2 Loading the package

Once the package is installed, you can load it just like any other R package, simply type:

```
library(CAGExploreR)
```

This should take no more than 20 seconds.

1.3 Loading Data

CAGExploreR comes pre-loaded with tables that consist of CAGE-Seq tag counts at promoter regions from two samples: the MCF7 breast cancer and the A549 lung cancer cell lines. The example workflow presented here will use this dataset to demonstrate the capabilities of **CAGExploreR**. For convenience, you can choose to load this data directly from the package and skip the next section. However in practice when analyzing your own personal datasets, you would need to complete the next section.

1.3.1 Obtaining tag counts from raw BAM files

For this example, the CAGE-Seq BAM files can be downloaded from the ENCODE website `http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeRikenCage/`. In addition to the files below you will also need to download the corresponding index files (*.bai). Be warned that the total size of these files is 12.6 GB!

- wgEncodeRikenCageA549CellPapAlnRep1.bam
- wgEncodeRikenCageA549CellPapAlnRep2.bam
- wgEncodeRikenCageA549CytosolPapAlnRep3.bam
- wgEncodeRikenCageA549CytosolPapAlnRep4.bam
- wgEncodeRikenCageA549NucleusPapAlnRep3.bam
- wgEncodeRikenCageA549NucleusPapAlnRep4.bam
- wgEncodeRikenCageMcf7CellPapAlnRep1.bam
- wgEncodeRikenCageMcf7CellPapAlnRep2.bam
- wgEncodeRikenCageMcf7CytosolPapAlnRep3.bam
- wgEncodeRikenCageMcf7CytosolPapAlnRep4.bam
- wgEncodeRikenCageMcf7NucleusPapAlnRep3.bam
- wgEncodeRikenCageMcf7NucleusPapAlnRep4.bam

Files 1-6 come from the A549 cell line and files 7-12 come from the MCF7 cell line. You will notice that these files were generated from RNA that was extracted from the nucleus, cytosol, or whole cell, however in our example here we will be pooling them together and treating them as 6 replicates for each of the two conditions.

Once we have these files in our local folder, we can load their names into R. It is important to provide IDs to each file in the format "**sample name**.*replicate number*". Files that correspond to replicates from a single condition should have the same prefix and different numbered suffix separated by a dot as shown below.

```
files <- dir()[grep("bam", dir())]
my.bai.files <- dir()[grep("bam.bai", dir())]
my.bam.files <- setdiff(files, my.bai.files)
my.ids <- c("a545.1", "a545.2", "a545.3", "a545.4", "a545.5", "a545.6", "mcf7.1", "mcf7.2",
    "mcf7.3", "mcf7.4", "mcf7.5", "mcf7.6")
```

The next step is to load promoter region definitions. **CAGExploreR** provides both **FANTOM5** and **MPromDB** promoter regions. In this example we shall be using the **FANTOM5** regions which are more comprehensive:

```
data(F5.hg19.promoters)
my.prom.defs <- definePromoters(F5.hg19.promoters)
```

User-supplied regions can also be used as long as their structure and format corresponds to the ones shown here. Let's take a look at the promoters that we just loaded to see how the file is structured.

```
head(F5.hg19.promoters)

##    chr strand  start    end      gene
## 1 chr1      - 910579 917517 C1orf170@
## 2 chr1      - 917466 917485  C1orf170
## 3 chr1      - 917507 917517  C1orf170
## 4 chr1      - 934342 935552     HES4@
## 5 chr1      - 934985 934997      HES4
## 6 chr1      - 935277 935309      HES4
```

Each row corresponds to a promoter region that is assigned to a gene, usually by virtue of its genomic vicinity. Coordinates are based on the hg19 genome build. Gene names are HGNC-approved names only. You will notice that some gene names contain the character @ at the end. This corresponds to rows which are "full-gene" regions that are calculated by taking the union of the whole gene plus all promoters defined for that gene. These are used for calculating "coverage quality" in downstream analysis. Whole gene regions are obtained from the **txdb.hsapiens.ucsc.hg19.knowngene** Bioconductor package. NOTE: not all genes have "full-gene" regions assigned to them in this dataset because the aforementioned package is based on Entrez gene IDs, and so there may be some genes that have HGNC names but no Entrez identifier.

The **FANTOM5** promoter set contains a total of 77207 promoters across 14449 multi-promoter genes, out of which 14253 genes have full gene regions available for "coverage quality" assessment. The average number of promoters per gene is 5.3434 with an average promoter region length of 26 bases (sd=21).

For comparison, the **MPromDB** promoter set contains a total of 12200 promoters across 4436 multi-promoter genes, out of which 4424 genes have full gene regions available for "coverage quality" assessment. The average number of promoters per gene is 2.7502 with an average promoter region length of 784 bases (sd=767).

Now that we have our input files and the promoter region definitions, we need to quantify the number of CAGE-Seq tags that map to each of these regions.

```
mcf7a549.raw.counts.F5 <- countTags(my.bam.files, my.bai.files, my.ids, my.prom.defs)
# This takes 2-10 minutes per BAM file
```

1.3.2 Loading data included with CAGExploreR

For the purposes of this exercise to save time, we can load the tag counts table from within the package.

```
data(mcf7a549.raw.counts.F5)
```

Let's take a look at what this object looks like by taking the first 6 rows of each element:

```
lapply(mcf7a549.raw.counts.F5, head)

$depth
   a545.1    a545.2    a545.3    a545.4    a545.5    a545.6
 29146943  18420034 103773964 154692891  89079842 133060582

$counts
```

```
   chr strand  start    end a545.1 a545.2 a545.3 a545.4 a545.5 a545.6 mcf7.1 mcf7.2
1 chr1      - 910579 917517      7      7      5     10     10     25     10     14
2 chr1      - 917466 917485      0      0      0      0      0      0      0      0
3 chr1      - 917507 917517      0      0      0      0      0      0      0      0
4 chr1      - 934342 935552     81     47   1490   2956   1558   4954   2664    652
5 chr1      - 934985 934997      0      2      8     10     14     14     16      6
6 chr1      - 935277 935309      0      0     14     10      4     16     10      2
  mcf7.3 mcf7.4 mcf7.5 mcf7.6      gene
1     12     17     43     19 C1orf170@
2      0      0      0      0  C1orf170
3      0      0      0      0  C1orf170
4   2634   5705   8172   2969     HES4@
5     10     26     46     22      HES4
6     20     18     52     26      HES4
```

The first element of this list **depth**, shows the total number of tags mapped in each library. The second element **counts** is a data.frame showing the number of CAGE-Seq tags mapping to each library and promoter. User-supplied count tables such as this one can also be provided directly instead of counting from BAM files, as long as the format is identical to the one seen above.

1.4 Combining Replicates

As mentioned previously, sample IDs follow the "**sample name**.*replicate number*" format, where replicates coming from the same sample have common sample name and different replicate number separated by a period. In this section we shall be combining replicate libraries together via pooling. NOTE: here we use the term "replicates" loosely, allowing the user to decide which samples to pool for analysis. In this example we are interested in comparing MCF7 and A549 cell lines to one another, but another choice of pooling might involve comparing between cancer v.s. normal or early-time point v.s. late-time point, etc. The user has the flexibility to pool samples any way they wish and this is achieved by changing the sample IDs in the data.frame above accordingly.

```
data.pooled <- pool.data(mcf7a549.raw.counts.F5)
```

The object produced is a list with 3 elements, let's take a look at first few lines of each:

```
head(lapply(data.pooled, head))

## $depth
##     a545.     mcf7.
## 528174256 326228197
##
## $coverage
##                          a545. mcf7.     gene
## chr1:910579..917517,-      64   115 C1orf170
## chr1:934342..935552,-   11086 22796     HES4
## chr1:1017198..1051754,-  6315  5643 C1orf159
## chr1:1138888..1142194,-     1  2941 TNFRSF18
## chr1:1146706..1149548,-    17    18  TNFRSF4
## chr1:1152288..1167452,- 36396 21742     SDF4
##
## $data
## $data$A1BG
##                            a545. mcf7.
## chr19:58858886..58858925,-    22    70
```

```
## chr19:58858938..58859039,-  2831  3237
## chr19:58864822..58864847,-     0     2
##
## $data$`A1BG-AS1`
##                             a545. mcf7.
## chr19:58858666..58858722,+    14    10
## chr19:58859101..58859149,+   864   906
##
## $data$A1CF
##                             a545. mcf7.
## chr10:52645379..52645393,-    14     0
## chr10:52645416..52645444,-    36     2
##
## $data$A2M
##                           a545. mcf7.
## chr12:9242983..9242995,-     0     2
## chr12:9268507..9268523,-     0    76
## chr12:9268528..9268542,-     0    12
##
## $data$A2ML1
##                           a545. mcf7.
## chr12:8975101..8975108,+     0     4
## chr12:8975144..8975169,+     0    12
##
## $data$A4GALT
##                             a545. mcf7.
## chr22:43116819..43116859,-   974  2086
## chr22:43116867..43116885,-  5058  3495
```

The **depth** element contains the total number of tags mapped in the pooled samples. The **coverage** element shows the total number of tags mapped to each gene in all samples after pooling, and this is the raw gene expression. Finally the **data** element is itself a list, each element corresponding to a gene. Each such sub-element contains a table of tag counts for each promoter region for that gene in all samples after pooling.

1.5 Obtaining Promoter Composition Results

Once we have a data object that contains the pooled CAGE-Seq tag counts for all samples across all genes and promoters, we can now obtain differential promoter composition (DPC) statistics.

1.5.1 A broad view: Gene-level statistics

```
results <- diffcomp(data.pooled)
```

This step should take about 3 minutes. By default, this function call will generate gene-level statistics, let's take a look:

```
head(results)

         entropy.Reduction fdr geneHetero coverage
MIR205HG                 1   0    0.97767  0.76144
TRIM55                   1   0    0.18864  0.20223
ADRA2A                   1   0    0.38089  0.26275
```

```
AOC1                    1   0    0.26733  0.16957
C1orf167                1   0    0.67985       NA
DNAH10                  1   0    0.08181  0.08985
                                        dominant.promoter.switch
MIR205HG   chr1:209602156..209602174,+|chr1:209605592..209605601,+
TRIM55         chr8:67039329..67039377,+|chr8:67039474..67039487,+
ADRA2A   chr10:112836589..112836607,+|chr10:112836779..112836797,+
AOC1       chr7:150549595..150549614,+|chr7:150553614..150553634,+
C1orf167       chr1:11845345..11845347,+|chr1:11847842..11847849,+
DNAH10   chr12:124246842..124246883,+|chr12:124392789..124392812,+
```

This is the main table showing DPC results. Remember that the motivation behind detecting DPC is to determine if the set of promoters for a gene are being utilized differently across conditions.

- **entropy.Reduction** is the main effect measure for DPC and the table is sorted by this value, with genes with the highest DPC showing up first. A value of 0 corresponds to no DPC, i.e. the relative transcription occurring at the different promoters is the same across conditions. Conversely, a value of 1 would indicate maximal DPC and that each condition transcribes a gene from a unique promoter.
- **fdr** is the corresponding one-sided p-value testing the null hypothesis that the entropy reduction is zero. It is corrected for multiple comparisons using the Benjamini-Hochberg method by default. The user can supply any correction method supported by the **p.adjust** function in R.
- **geneHetero** is an entropy-based measure of gene expression heterogeneity across conditions. A value of 0 corresponds to no differential gene expression between conditions, and a value of 1 means that one or more particular conditions have non-zero gene expression while in another condition(s) the gene is not expressed at all.
- **coverage** compares the number of tags mapped to promoter regions to the total number of tags mapped to the entire gene region. The value reported is the mean across conditions. Since CAGE-Seq tags are strictly associated with the 5′ mRNA cap, as long as the promoter regions are defined correctly, we do not expect to see any tags aligning to other nearby non-promoter regions in theory. A value of 1 would mean that the promoter regions defined capture all of the tags found in the entire gene region. In practice however, there is noise present and we do expect some tags mapping outside our known promoters. It should be noted that a significant deviation from 1 could point to the existence of a novel promoter that is not accounted for by the current definitions. A value close to 0 would mean that for that particular gene, the promoter definitions are completely inadequate and the results suspicious. This could also occur if the gene region was not defined correctly. Values greater than 1 are possible if the promoter definitions overlap one another. In either case, a value of greater than 1 or close to 0 imply that the results for that gene should be discarded and the promoter definitions adjusted as they do not provide a complete and unequivocal view of DPC. We recommend to filter results to values of coverage between 0.5 and 1. If a gene does not have a gene region defined in the promoter definitions file (e.g. GENE@), then coverage cannot be calculated and NA will appear.
- **dominant.promoter.switch** shows whether the dominant promoter switches between conditions. This is in the form of a pipe (|) delimited list of promoters if there is a switch in dominant promoters and is empty otherwise.

Let's subset our results to only those genes that pass the FDR-adjusted p-value cutoff of 0.001 and have coverage between 0.5 and 1. Beneath the output table are some examples of useful statistics of interest.

```
significant <- subset(results, fdr < 0.001 & coverage >= 0.5 & coverage <= 1)
head(significant)

         entropy.Reduction        fdr geneHetero coverage
```

```
MIR205HG            1.0000  0.000e+00  0.9776714   0.7614
AFP                 1.0000  0.000e+00  0.1213595   0.5438
ZNF70               0.6495  5.078e-19  0.0002358   0.7354
PITX2               0.6493 3.639e-157  0.0740950   0.6792
PAX8                0.6467  6.505e-13  0.0503409   0.7130
GPNMB               0.6402  3.773e-13  0.8584443   0.6502
                                        dominant.promoter.switch
MIR205HG chr1:209602156..209602174,+|chr1:209605592..209605601,+
AFP          chr4:74301931..74301947,+|chr4:74316336..74316348,+
ZNF70      chr22:24093154..24093183,-|chr22:24093267..24093327,-
PITX2    chr4:111544254..111544270,-|chr4:111558135..111558198,-
PAX8     chr2:113977697..113977733,-|chr2:113993052..113993097,-
GPNMB        chr7:23286379..23286401,+|chr7:23312902..23312968,+

# This is how many significant, high-coverage genes we are left with:
nrow(significant)

[1] 3979

# This is the number of genes in which the dominant promoter switches between conditions:
sum(significant$dominant.promoter.switch != "")

[1] 533

# This is the number of genes that are have differential promoter composition but very
# little to no differential gene expression:
sum(significant$geneHetero < 0.01)

[1] 1102

# Differences in gene expression and differences in promoter composition do not appear to
# be correlated:
cor(significant$entropy.Reduction, significant$geneHetero)

[1] 0.1342
```

1.5.2 A detailed view: Promoter-level statistics

```
results.detailed <- diffcomp(data.pooled, detailed = TRUE)
```

Now that we've found which genes have DPC and to what extent, we may be interested in finding out exactly which promoters are involved and the magnitude of their switching. To get such promoter-level statistics we make use of the **detailed** option. By default, results for all genes are displayed. In our example we have a total of 166,696 comparisons made. Let's take a look at just the first 15:

```
head(results.detailed, 15)
```

```
                                                                comparison     gene
1    chr1:209602156..209602174,+;chr1:209605592..209605601,+;a545.;mcf7. MIR205HG
2        chr8:67039329..67039377,+;chr8:67039474..67039487,+;a545.;mcf7.   TRIM55
3  chr10:112836779..112836797,+;chr10:112836589..112836607,+;a545.;mcf7.   ADRA2A
4    chr7:150549595..150549614,+;chr7:150553614..150553634,+;a545.;mcf7.     AOC1
5        chr1:11847842..11847849,+;chr1:11845345..11845347,+;a545.;mcf7. C1orf167
6  chr12:124246842..124246883,+;chr12:124392789..124392812,+;a545.;mcf7.   DNAH10
7        chr9:18474125..18474160,+;chr9:18474163..18474202,+;mcf7.;a545. ADAMTSL1
8    chr5:160112418..160112436,-;chr5:160112386..160112405,-;a545.;mcf7.   ATP10B
9      chr12:63544882..63544895,-;chr12:63544973..63544986,-;mcf7.;a545.   AVPR1A
10     chr11:27062502..27062512,+;chr11:27077015..27077031,+;a545.;mcf7.    BBOX1
11     chr10:45496218..45496270,-;chr10:45496353..45496379,-;a545.;mcf7. C10orf25
12 chr12:121078397..121078417,+;chr12:121078423..121078446,+;a545.;mcf7.    CABP1
13     chr20:54987494..54987512,+;chr20:54987258..54987289,+;mcf7.;a545.    CASS4
14 chr10:106113515..106113527,+;chr10:106113583..106113597,+;mcf7.;a545.  CCDC147
15     chr11:86106304..86106326,+;chr11:86106370..86106377,+;a545.;mcf7.   CCDC81
           OR  log2OR    pvalue       fdr
1  8.131e-05 -13.586 5.312e-11 2.441e-10
2  2.353e-03  -8.731 3.304e-05 9.636e-05
3  4.878e-03  -7.679 2.717e-04 7.038e-04
4  3.175e-03  -8.299 6.656e-05 1.871e-04
5  1.587e-02  -5.977 6.160e-03 1.266e-02
6  1.070e-04 -13.190 5.344e-11 2.455e-10
7  2.041e-02  -5.615 8.638e-03 1.722e-02
8  1.299e-02  -6.267 3.028e-03 6.605e-03
9  1.538e-02  -6.022 4.770e-03 1.002e-02
10 2.857e-02  -5.129 1.781e-02 3.303e-02
11 1.818e-02  -5.781 6.915e-03 1.406e-02
12 1.010e-02  -6.629 1.583e-03 3.636e-03
13 1.099e-02  -6.508 2.011e-03 4.534e-03
14 4.000e-02  -4.644 3.404e-02 5.888e-02
15 5.917e-03  -7.401 3.627e-04 9.225e-04
```

- **Comparison** specifies which pair of promoters and conditions are being compared.
- **Gene** specifies the gene.
- **OR** is the odds ratio and measures the strength of switching in promoter composition between the two conditions and promoters being compared. This ratio shows how much larger (or smaller) the odds of observing tags in the first v.s. the second promoter in the first condition are as compared to the odds of observing tags in the first v.s. the second promoter in the second condition. A value of 1 means that the odds are the same in both conditions, and so promoter composition does not change for this set of promoters when comparing these two conditions.

- **log2OR** is the log-base-2 transformed odds ratio. This transformation is useful to make it clearer which direction the odds ratio is in: if the OR is less than 1, the log will have a negative sign, and if the OR is greater than 1, the log will be positive. The log odds ratio also makes the scale more easily understood, varying from 0 (no switching) to infinity (maximum switching). You will notice that all results are sorted within genes by decreasing log odds ratio (irrespective of sign) so that most significant switches show up first within gene blocks.
- **P-value** corresponds to the odds ratio and tests whether it is significantly different from 1 and conversely whether the log odds ratio is significantly different from 0. Note that this is different from the p-value that corresponds to the overall entropy reduction for a given gene.
- **FDR** is the Benjamini-Hochberg adjusted p-value. Just as before, the user can supply any method for correcting for multiple comparisons as long as it is supported by the **p.adjust** function in R.

Now let's subset these results to only those significant, high-coverage genes that we found in the previous section. In addition, let's also just look at those genes in which the dominant promoter switches.

```
significant.genes <- rownames(subset(results, fdr < 0.001 & coverage >= 0.5 & coverage <= 1 &
    dominant.promoter.switch != ""))
sig.results.detailed <- subset(results.detailed, gene %in% significant.genes)
nrow(sig.results.detailed)

[1] 13807

head(sig.results.detailed)

                                                                comparison     gene
1   chr1:209602156..209602174,+;chr1:209605592..209605601,+;a545.;mcf7. MIR205HG
64      chr4:74316336..74316348,+;chr4:74301931..74301947,+;mcf7.;a545.      AFP
109   chr22:24093267..24093327,-;chr22:24093130..24093141,-;mcf7.;a545.    ZNF70
110   chr22:24093267..24093327,-;chr22:24093154..24093183,-;mcf7.;a545.    ZNF70
111   chr22:24093154..24093183,-;chr22:24093130..24093141,-;mcf7.;a545.    ZNF70
112 chr4:111558135..111558198,-;chr4:111544254..111544270,-;mcf7.;a545.    PITX2
           OR   log2OR     pvalue        fdr
1   8.131e-05 -13.5863  5.312e-11  2.441e-10
64  1.817e-05 -15.7479  3.928e-15  2.202e-14
109 2.685e-03  -8.5411  4.759e-09  1.939e-08
110 3.126e-03  -8.3215 4.822e-145 1.409e-143
111 8.588e-01  -0.2196  8.814e-01  9.136e-01
112 4.388e-06 -17.7981  6.143e-19  3.972e-18
```

1.6 Generating Plots

1.6.1 Visualizing DPC in a gene of interest

In order to visualize differential promoter composition in a gene of interest, we need to specify the pooled data object and the gene we want to look at. Let's see what happens in the gene that in our example had the largest odds ratio, *SYT1*.

```
plotcomp(data.pooled, "SYT1")
# [1] 'chr12:79258547..79258607,+' 'chr12:79258480..79258499,+'
# [3] 'chr12:79439461..79439490,+' 'chr12:79845142..79845155,+'
# [5] 'chr12:79439444..79439455,+' 'chr12:79258463..79258477,+'
# [7] 'chr12:79258764..79258773,+' 'chr12:79371576..79371593,+'
# [9] 'chr12:79439405..79439421,+' 'chr12:79258791..79258806,+'
```

```
# [11] 'chr12:79258444..79258455,+' 'chr12:79371509..79371528,+'
# [13] 'chr12:79371495..79371508,+' 'chr12:79371636..79371672,+'
# [15] 'chr12:79844368..79844393,+' 'chr12:79357890..79357893,+'
```

The figure is composed of 3 parts: (a) promoter composition, (b) gene expression and (c) promoter location plots.

(a) is the main promoter composition plot and has the conditions in rows and promoters in columns, each having a different color. The size of each colored bar specifies the extent to which transcription occurs from that promoter (measured as a proportion of all tags mapping to all promoters) and the number of CAGE-seq tags mapping to that promoter region is printed inside the bar. The number for the promoter which has the largest expression is colored white to make it more clearly visible. The conditions are clustered together by similarity in these composition profiles. In our example, the clustering is not meaningful because we only have 2 conditions, but this can help identify conditions that share similar promoter composition profiles when a large number of conditions is being compared. We can clearly see in this example that MCF7 cells preferentially utilize the "light orange" promoter, whereas A545 cells preferentially utilize the "red" one, indicating a very strong switch in promoter composition between the two conditions.

(b) has the same row structure as in (a) and shows the gene expression for the conditions and is measured in tags per million (tpm). This is calculated by taking all of the CAGE-Seq tags mapping to the entire gene region (if it is available) relative to all tags mapped in this condition, multiplied by a million. If the gene region coordinates have not been specified in the promoter definitions, then the total number of tags are taken by summing across all promoters available. Together with (a), these plots simultaneously demonstrate the extent of DPC and differential gene expression for the gene of interest.

(c) is a genomic region plot showing where the promoters are located relative to the gene model and known transcript models from ENSEMBL. This plot is generated using data from the **GenomeGraphs** Bioconductor package. The gene model is shown in yellow on top, the transcripts in blue in the middle, and the promoter regions in various colors at the bottom, where the colors correspond to the same color-code used in panel (a) for clarity. Each promoter region also has a colored vertical bar around it, the height of which gives a sense of the relative amount of transcription occurring from the promoters when summing over all conditions. NOTE: the coordinates for this plot are chosen in such a way as to fit all of the promoter regions comfortably on the screen, and so this means that most of the time, only the 5 prime end of the gene or another portion of it is visible. The strand and direction of transcription can be seen by looking at the location of the "3'" and "5'" markers on the coordinate display, with the positive strand always starting with 5' on the left. In this example we are on the positive strand and transcription is going from left to right.

After a call to **plotcomp()**, the coordinates of the promoter regions are displayed in the R console which we can see above. The promoters are ordered in the same way as in the plot, left to right as displayed.

1.6.2 Making multiple plots in HTML

Instead of making plots for each gene one by one, we can perform batch generation into an HTML document:

```
html.report(data.pooled, rownames(results)[1:3])
```

We once again have to specify the pooled data object, followed by the names of the genes we would like the plots of. In this example we specify to make plots for the top 3 significant genes from our results table. Similarly we can specify any number of genes in any order that are of interest (e.g. a list of transcription factors, etc.) This will create an HTML file called *Switch Report.html* in the current directory, together with a folder called *Figures* that will contain individual plots inside. NOTE: if such a file/directory exists already,

all files will be overwritten. You can change the names of report files and figure directory easily however:

```
html.report(data.pooled, my.genes, fig.dir = "myFigures", report.name = "my.report")
```

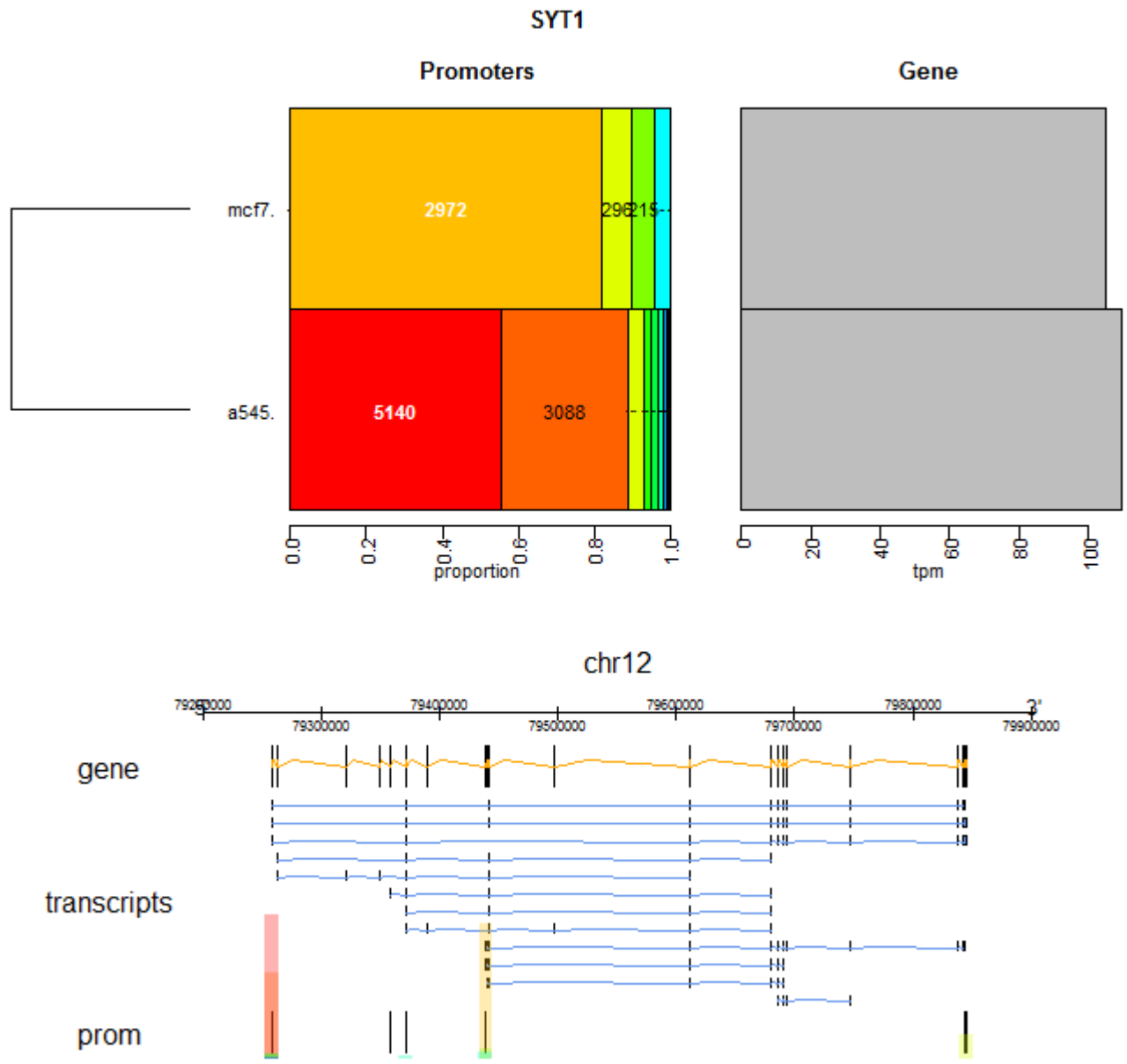

Figure 1: Differential promoter composition for SYT1 gene with 16 promoters

1.6.3 Genome-wide DPC volcano plot

One possible way to visualize DPC on a genome-wide basis is to generate a volcano plot that displays the effect measure on the x-axis and the statistical significance on the y-axis. Since the effect measured as an odds ratio is one-sided, we can make it extend across the entire x-axis by taking its logarithm. Similarly we can take the negative of the logarithm of the multiplicity corrected p-value for the y-axis. This way, more statistically significant results are higher up, and larger effect measures are away from the origin.

```
plot(y=-log(results.detailed$fdr,base=10),
x=results.detailed$log2OR,
col=rgb(0,100,0,10,maxColorValue=255),
pch=20,
ylab="-log10(q-value)",
xlab="log2(odds ratio)",
main="MCF7 v.s. A549 cell lines")
abline(v=0,h=0)
```

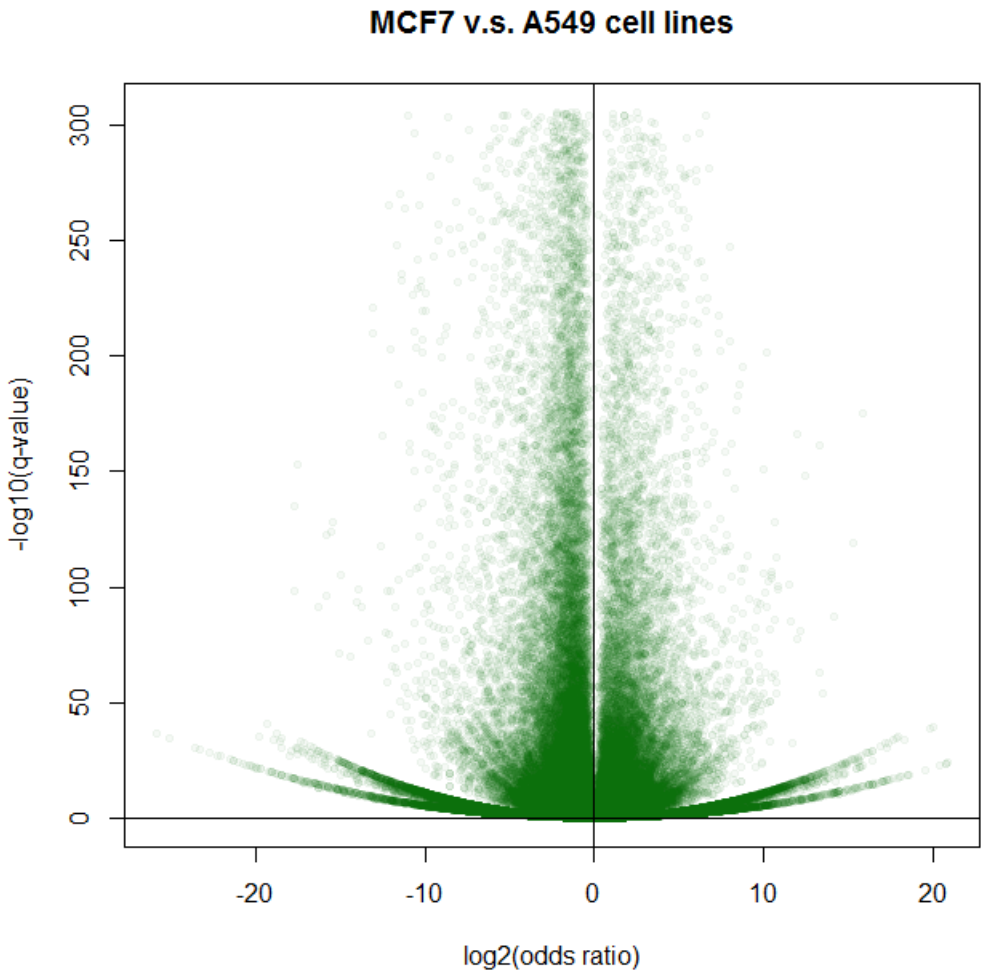

Figure 2: Genome-wide volcano plot for comparing differential promoter composition between 2 cell lines