Peak calling statisticsSolutions to the exercise

Jacques van Helden

2016-11-24

Contents

Objectives **2**

Data sets **2**

Read counts per 200bp windows . . . 2

Read counts per 50bp windows . . . 2

How was this dataset generated? **2**

Generating windows of equal size along the reference genome . . . 3

Converting bam to sorted bed . . . 3

Counting the reads per bin . . . 3

Questions **4**

References **4**

Solutions **5**

Loading the two bed files describing read counts in **R** . . . 5

Counting the total reads per library . . . 6

Normalization of the input library . . . 6

Comparison between FNR and input count distributions . . . 8

Chromosomal repartition . . . 12

6. For each bin, compute the following statistics, and collect them in a table (one row per bin, one column per statistics): . . . 13

P-value of the test reads, using a hypergeometric distribution (explain your model) . . . 14

An alternative (but equivalent) reasoning to model hypergeometric p-value . . . 16

P-value of the test reads, using a Poisson distribution with local lambda estimate (explain your model) 17

P-value of the test reads, using a binomial distribution (explain your model). . . . 18

7. Draw some dot plots to compare the different statistics (fold enrichment, log-fold, p-values with the different models). . . . 19

Distribution of measured p-values . . . 21

8. **Discuss the results.** . . . 21

The binomial and Poisson p-values are very close to each other . . . 21

Hypergeometric p-values are much higher than binomial or Poisson p-values . . . 22

Objectives

During this practical, we will develop in **R** a simplified version of the **MACS** peak calling algorithm, and test it on a relatively small dataset (ChIP-seq data for the FNR transcription factor in the genome of the Bacteria *Escherichia coli*).

Beware: this exercise only performs some basic part of the peak-calling processing. The result will certainly not be as good as the peaks detected by conventional peak calling algorithms. The essential goal of this tutorial is understand the basic statistics used for peak calling, by implementing ourselves a simplified way to test the enrichment of reads in a partition of non-overlapping windows covering the genome.

Data sets

For each sample (***FNR ChIP-seq*** and ***genomic input***) we pre-computed bed files with the counts reads mapped in non-overlapping windows of 200bp, or 50bp, respectively (two files per sample).

Read counts per 200bp windows

1. FNR ChIP-seq sample (***test***)
   FNR_ChIP-seq_Anaerobic_A_GSM1010219_reads_per_200bp.bedg
2. Genomic input (control)
   Escherichia_coli_K_12_MG1655_input_reads_per_200bp.bedg

Read counts per 50bp windows

3. FNR ChIP-seq sample (***test***)
   FNR_ChIP-seq_Anaerobic_A_GSM1010219_reads_per_50bp.bedg
4. Genomic input (control)
   Escherichia_coli_K_12_MG1655_input_reads_per_200bp.bedg

These files are in bedGraph format.

Remember: the bed convention uses zero-based coordinates, with semi-open intervals. Thus, the coordinates `0 50` correspond to

- the semi-open interval `[0:50[` in zero-based coordinates;
- i.e. the closed interval `[0:49]` in zero-based coordinates;
- i.e. the closed interval `[1:50]` in one-based (human understandable) coordinates.

How was this dataset generated?

This section is optional. It explains the tricks we used to generate a data set for this exercise.

We obtained the original datasets (mapped reads for the ChIP-seq and input samples) by following **Morgane Thomas-Chollier's** tutorial Hands-on introduction to ChIP-seq analysis.

Then, we used bedtools to count the number of reads per bins, i.e. non-overlapping windows of fixed size (200bp per window) covering the full genome of *Escherichia coli*.

The protocol to count reads per bin is the following:

1. Generate a bed file defining the bin limits (one row per bin).
2. Convert the mapped reads from sam to bed file, sorted by chromosomal position.
3. Use **bedtools** to compute the intersection between each bin and the read file (i.e. count the reads falling onto each bin).

Generating windows of equal size along the reference genome

```
bedtools makewindows \
  -g Escherichia_coli_K_12_MG1655_genome.txt -w 200 \
  > Escherichia_coli_K_12_MG1655_windows_200bp.bed
```

The file genome.txt is expected to be a 2-columns file indicating the ID and length of each chromosome. In the case of Escherichia coli, the file contains a single line:

```
gi|49175990|ref|NC_000913.2|    4639675
```

Converting bam to sorted bed

To compute the intersection between two sets of genomic regions (bedtools intersect), bedtools requires two bed files sorted by chromosome and by chromosomal position. We first need ton convert sam to bed format. For this we use an intermediate BAM format.

```
## Convert mapped reads of FNR ChIP-seq library (test)
samtools view -bS SRR576933.sam | bedtools bamtobed | \
  sort -n -k 2 \> SRR576933_sorted.bed

## Convert mapped reads of control library (input)
samtools view -bS SRR576938.sam | bedtools bamtobed \
  | sort -n -k 2 > SRR576938_sorted.bed
```

Counting the reads per bin

```
# Count the number of reads per window in the FNR ChIP-seq library
bedtools intersect -a Escherichia_coli_K_12_MG1655_windows_200bp.bed \
-b SRR576933_sorted.bed \
-c -sorted \
>  FNR_ChIP-seq_Anaerobic_A_GSM1010219_reads_per_200bp.bedg

# Count the number of reads per window in the control library
bedtools intersect -a Escherichia_coli_K_12_MG1655_windows_200bp.bed \
-b SRR576938_sorted.bed \
-c -sorted \
>  Escherichia_coli_K_12_MG1655_input_reads_per_200bp.bedg
```

Questions

1. Download the two bed files describing read maps ("test" and "input", respectively), for a given window size.
2. Load these two tables in R.
3. Count the total reads for the FNR and input libraries, respectively.
4. Normalize the input library in order to obtain the same sum as the test (FNR) library.
5. Compare the distributions of counts per reads (normalized for the input) in the ChIP-seq and input samples.
6. For each bin, compute the following statistics, and collect them in a table (one row per bin, one column per statistics):

   Note: each of these statistics can be computed with a single operation – in R, you should avoid loops whenever possible.

   1. Number of reads in the test (FNR ChIP-seq)
   2. Number of reads in the input
   3. Normalized number of reads in the input (norm.input)/
   4. Reads per base in the input
   5. Fold enrichment (ratio between test and normalized input)
   6. Log-fold enrichment ($\log_{10}$ of the fold enrichment)
   7. P-value of the test reads, using a Poisson distribution with local lambda estimate (explain your model).
   8. P-value of the test reads, using a binomial distribution (explain your model).
   9. P-value of the test reads, using a hypergeometric distribution (explain your model).
7. Draw some dot plots to compare the different statistics (fold enrichment, log-fold, p-values with the different models).
8. **Discuss the results.**

References

1. MACS method descripion:
   Zhang, Y., Liu, T., Meyer, C.A., Eeckhoute, J., Johnson, D.S., Bernstein, B.E., Nussbaum, C., Myers, R.M., Brown, M., Li, W., et al. (2008) Model-based analysis of ChIP-Seq (MACS). Genome Biol, 9, R137.
2. To generate the data, we followed **Morgane Thomas-Chollier's** tutorial Hands-on introduction to ChIP-seq analysis, and then applied some tricks to count reads in fixed-width windows over the whole genome of *Escherichia coli*.

Solutions

Loading the two bed files describing read counts in R

We can directly load the data from the web site.

```
url.course <- "~/stats_avec_RStudio_EBA/" # For JvH local test only
#url.course <- "http://jvanheld.github.io/stats_avec_RStudio_EBA/"
url.data <- file.path(url.course, "practicals", "02_peak-calling", "data")

## Choose a window size (200 or 50)
window.size <- 200

## Load counts per window in chip sample
chip.bedg.file <- paste(sep="",
                        "FNR_",
                        window.size,"bp.bedg")
chip.bedg <- read.table(file.path(url.data, chip.bedg.file))
names(chip.bedg) <- c("chrom", "start", "end","counts")

## Load counts per window in the input sample
input.bedg.file <- paste(sep="", "input_",window.size,"bp.bedg")
input.bedg <- read.table(file.path(url.data, input.bedg.file))
names(input.bedg) <- c("chrom", "start", "end","counts")
```

Let us check the content of the bed files.

```
dim(chip.bedg) ## Check number of rows and columns
```

```
## [1] 23199     4
```

```
head(chip.bedg) ## check the content of the first rows
```

```
##                           chrom start  end counts
## 1 gi|49175990|ref|NC_000913.2|     0  200   1594
## 2 gi|49175990|ref|NC_000913.2|   200  400    834
## 3 gi|49175990|ref|NC_000913.2|   400  600    222
## 4 gi|49175990|ref|NC_000913.2|   600  800    172
## 5 gi|49175990|ref|NC_000913.2|   800 1000    123
## 6 gi|49175990|ref|NC_000913.2|  1000 1200    116
```

```
dim(input.bedg) ## Check number of rows and columns (should be the same as for test file)
```

```
## [1] 23199     4
```

```
head(input.bedg) ## check the content of the first rows
```

```
##                           chrom start  end counts
## 1 gi|49175990|ref|NC_000913.2|     0  200    514
## 2 gi|49175990|ref|NC_000913.2|   200  400    490
## 3 gi|49175990|ref|NC_000913.2|   400  600    392
## 4 gi|49175990|ref|NC_000913.2|   600  800    374
## 5 gi|49175990|ref|NC_000913.2|   800 1000    316
## 6 gi|49175990|ref|NC_000913.2|  1000 1200    352
```

Counting the total reads per library

For each library (FNR and input, respectively), the total number of reads is obtained by computing the sum of all values from the fourth column, which contains the number of reads for each window.

```
## The fourth column of the bedgraph file contains the counts
chip.counts <- chip.bedg[,4]
input.counts <- input.bedg[,4]

## Count total counts
chip.total <- sum(chip.counts)
input.total <- sum(input.counts)

## Compare total counts between test and input samples
print(chip.total)
```

```
## [1] 2622163
```

```
print(input.total)
```

```
## [1] 7480400
```

```
print (input.total/chip.total)
```

```
## [1] 2.852759
```

Let us already note that the input library contains 2.8 times more reads than the test library. The input library was generated by sequencing the raw genome of *Escherichia coli*.

We can thus compute the genome coverage.

```
G <- 4641652 ## Genome size
print(genome.coverage <- sum(input.total) * 35 / G)
```

```
## [1] 56.40535
```

The input library represents a 56-fold genome coverage. This is a rather good indication, since it will be used to estimate our random expectation for each positional window. It is thus important to have a sufficient depth in order to get robust estimations.

However, we need to keep in mind that the test and input libraries are of different sizes to define our statistics for the enrichment of each window.

Note: the sum of reads differs from the actual total counts. Indeed, the protocol used to generate the data counts the number of reads overlapping each window. So, if a read overlaps two adjacent windows, it will be counted twice. We will temporarily ignore this problem, which will be fixed in the future by counting reads only for the window that overlaps their central residue.

Normalization of the input library

We will now normalize the input library in order to obtain the same sum as the test (FNR) library.

```
## Option 1: mormalize the data based on the library sum
norm.input.libsum <- input.counts * chip.total / input.total
head(norm.input.libsum)
```

```
## [1] 180.1764 171.7635 137.4108 131.1011 110.7699 123.3893
```

```
## Option 1: mormalize the data based on median counts per library.
## This is more robust to outliers, esp. in the ChIP sample,
## where the actual peaks bias the mean towards higher values
norm.input.median <- input.counts *  median(chip.counts) /  median(input.counts)
head(norm.input.median)
```

```
## [1] 162.2312 154.6562 123.7250 118.0438  99.7375 111.1000
```

```
# Check the normalizing effect:
# Libsum-scaled input counts should have the same sum as test counts.
sum(chip.counts)
```

```
## [1] 2622163
```

```
sum(norm.input.libsum)
```

```
## [1] 2622163
```

```
sum(norm.input.libsum)/sum(chip.counts)
```

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

```
# Median-scaled input counts do not have the same sum as test counts.
sum(norm.input.median)
```

```
## [1] 2361001
```

```
sum(norm.input.median)/sum(chip.counts)
```

```
## [1] 0.9004022
```

```
## Make a choice between options 1 and 2
norm.input <- norm.input.median

## Display the first elements of the normalized input counts
head(norm.input)
```

```
## [1] 162.2312 154.6562 123.7250 118.0438  99.7375 111.1000
```

Whereas the input and test datasets consisted in natural numbers (read counts), the normalized input consists of decimal numbers. It can be interpreted as the number of reads that would be expected in each window from a library of the same size as the FNR (test) library, in absence of any specific binding. This is thus a genomic position-specific background model, estimated from the input library.

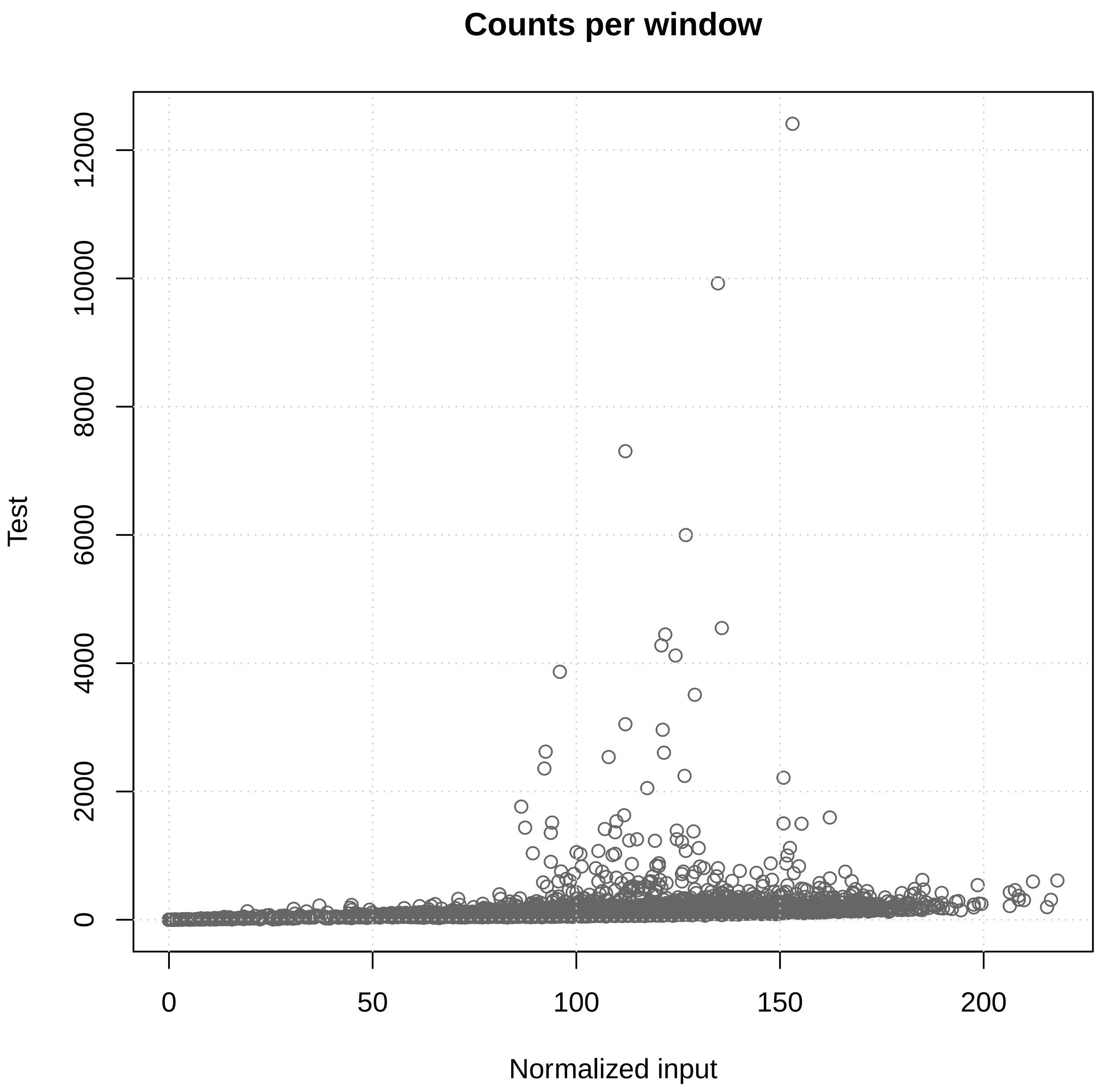
The principle of the following tests will be to define suitable statistics to compare the number of reads observed in each window of the FNR sample with this random expectation estimated from the normalized input.

Comparison between FNR and input count distributions

Compare the distributions of counts per reads (normalized for the input) in the ChIP-seq and input samples.

A simple way to compare the counts is to draw a dot plot of the test versus input reads per window.

```
par(mfrow=c(1,1))
plot(norm.input, chip.counts,
     xlab="Normalized input", ylab="Test",
     col="#666666",
     main="Counts per window")
grid()
```



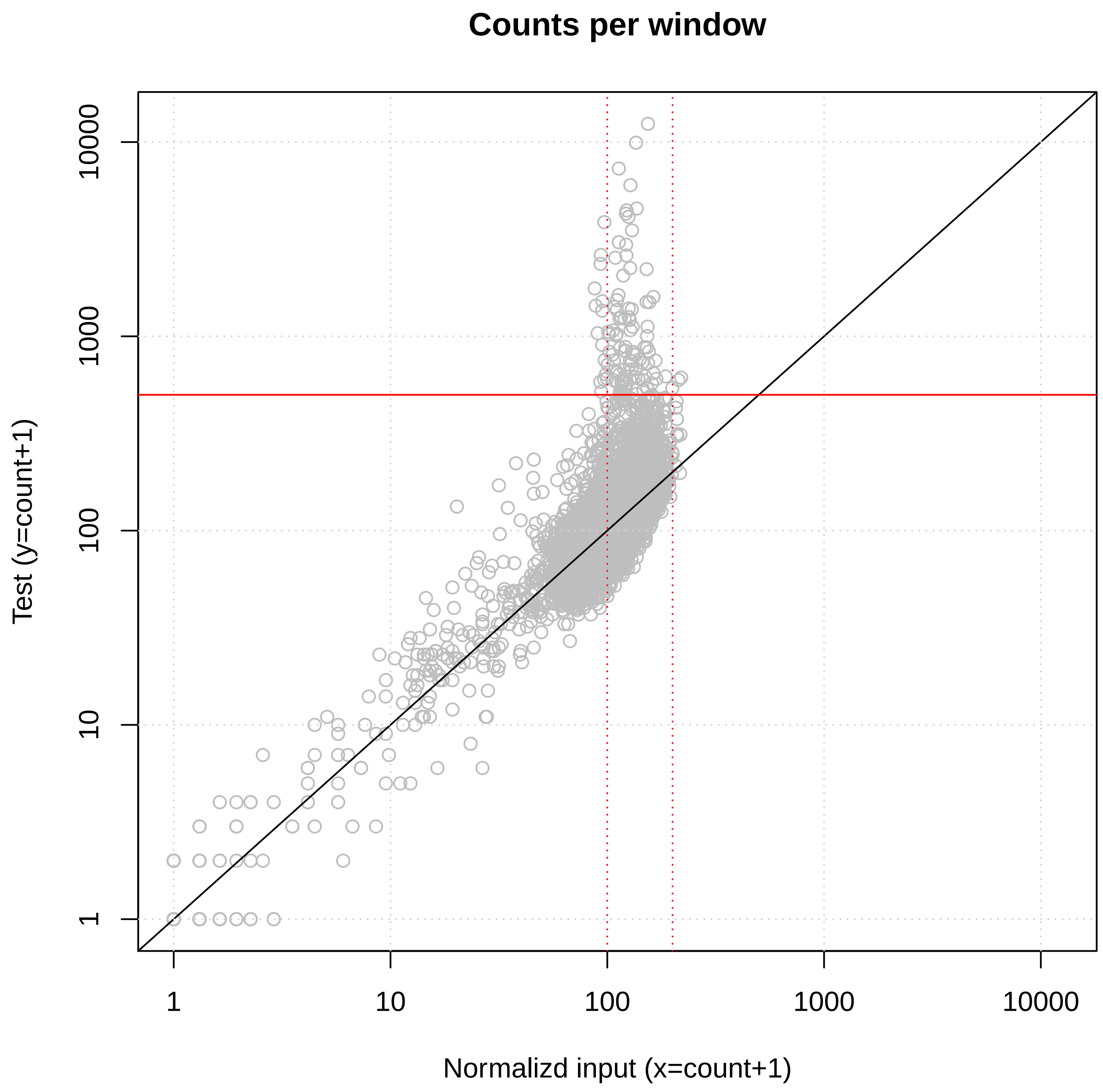
This plot highlights some windows with a very high rate of over-representation: whereas the counts per window range from 0 and 250 in the normalized input, the test library includes some windows with thousands of reads.

Notice the strong differences between the ranges of the abcissa and ordinate, which somewhat masks the relationship between test and input in the non-enriched windows. We can improve the readability by using a log scale.

```r
## Compute the range of counts for all windows (test or input) in order to use the same scale to displa
count.range <- range(c(chip.counts, norm.input))

## Note: we add 1 to the counts in order to avoid problem with the logarithmic scale
par(mfrow=c(1,1))
plot(norm.input+1, chip.counts+1,
     xlab="Normalizd input (x=count+1)",
     ylab="Test (y=count+1)",
     main="Counts per window",
     log="xy", col="gray", xlim=count.range+1, ylim=count.range+1)
abline(a=0,b=1) ## Plot the diagonal to compare counts between samples
grid()

## Roughly surround a region of interest to be discussed below
abline(h=500, col="red")
abline(v=c(100,200),col="red", lty="dotted")
```



The log plot shows a more or less linear relationship between test and input counts for the low frequencies ($x < 100$). It also highlights the fact that the windows that have a very high number of reads (e.g. $n \geq 500$) in the test library have an input count comprised between 100 and 200.

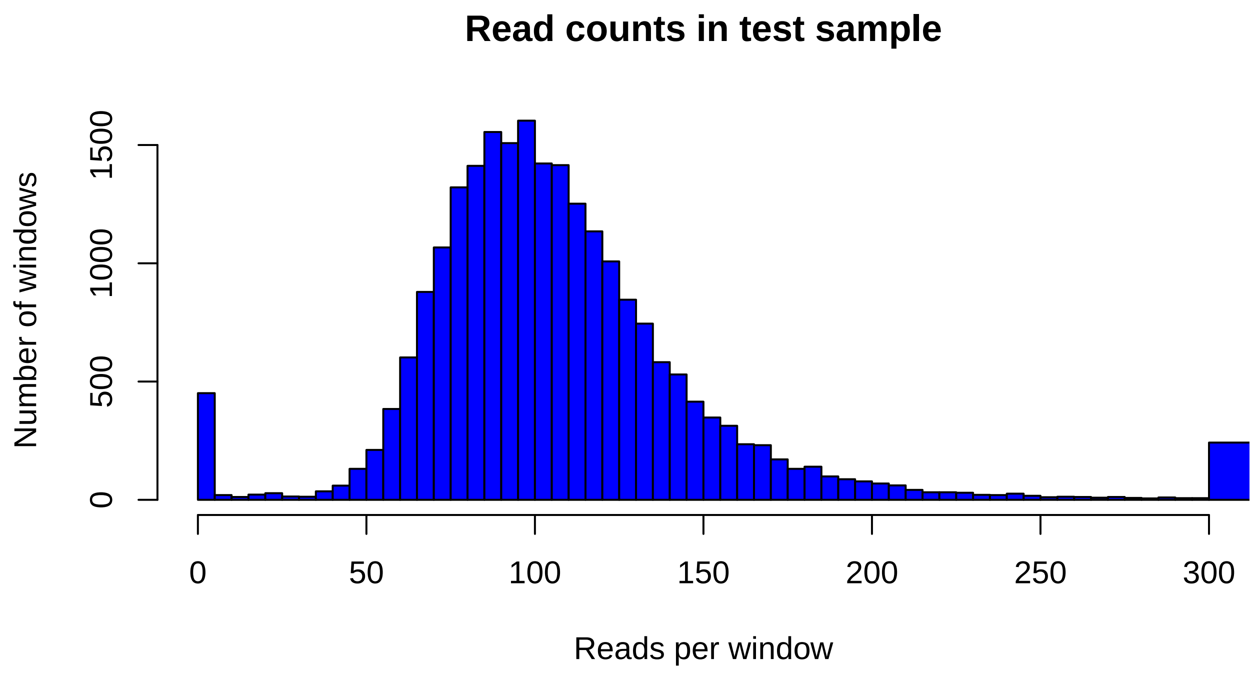
We can now plot the distribution of counts for each sample. We restrict the histograms to the range $0 \leq c \leq 300$ in order to better see the bulk of the distribution.

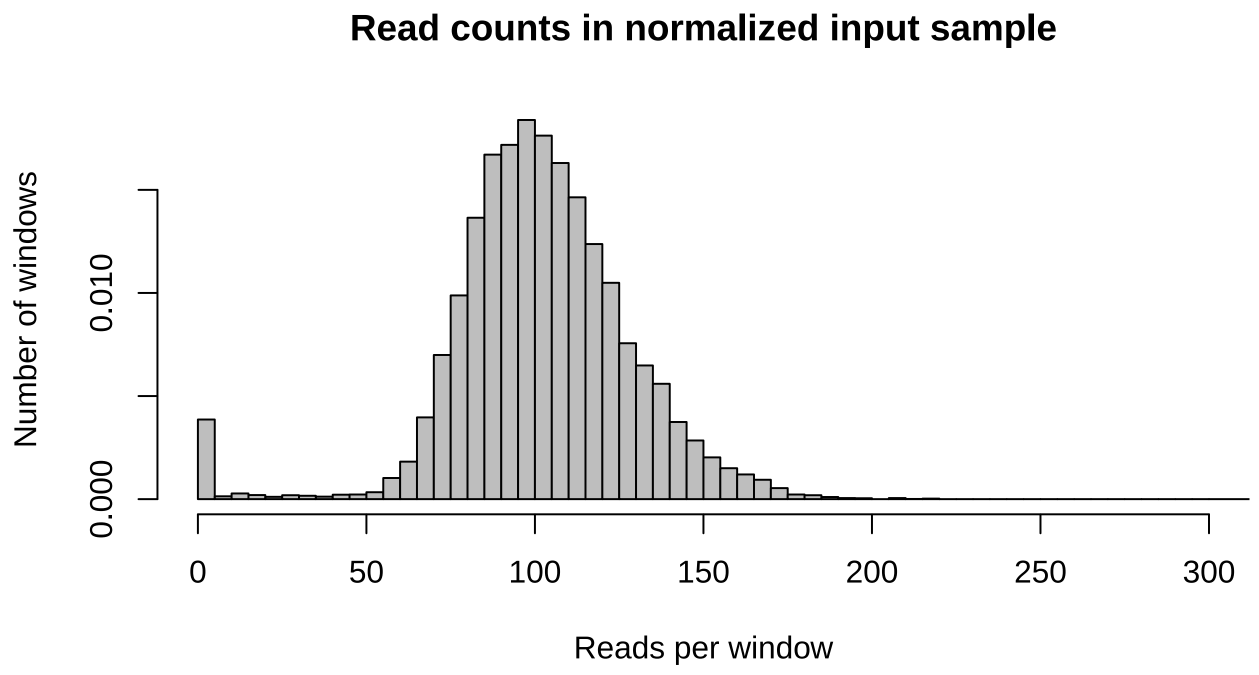
```
max.count <- count.range[2]
par(mfrow=c(2,1))

hist(chip.counts, breaks=c(seq(from=0,to=300,by=5),max.count),
     xlim=c(0,300), freq=TRUE, col="blue",
     main="Read counts in test sample",
     xlab="Reads per window", ylab="Number of windows")
```

```
## Warning in plot.histogram(r, freq = freq1, col = col, border = border,
## angle = angle, : the AREAS in the plot are wrong -- rather use 'freq =
## FALSE'
```

```r
hist(norm.input, breaks=c(seq(from=0,to=300,by=5),max.count),
     xlim=c(0,300),freq=FALSE, col="gray",
     main="Read counts in normalized input sample",
    xlab="Reads per window", ylab="Number of windows")
```





```
par(mfrow=c(1,1)) ## Restore default plot setting
```

Chromosomal repartition

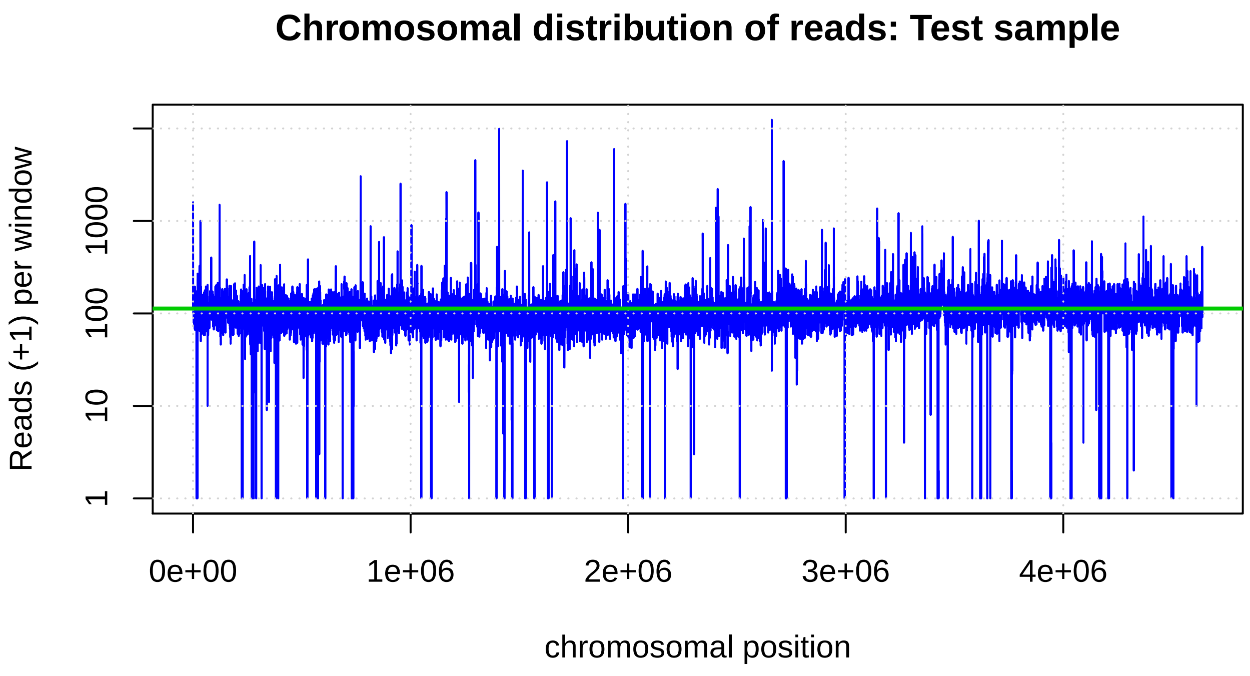
The two plots below are genome-wide density maps: the X axis represents the chromosomal position, the Y axis the number of reads per 200bp window. The test sample (top plot) clearly shows many windows with a relatively high number of reads, as compared to the normalized input (bottom plot).

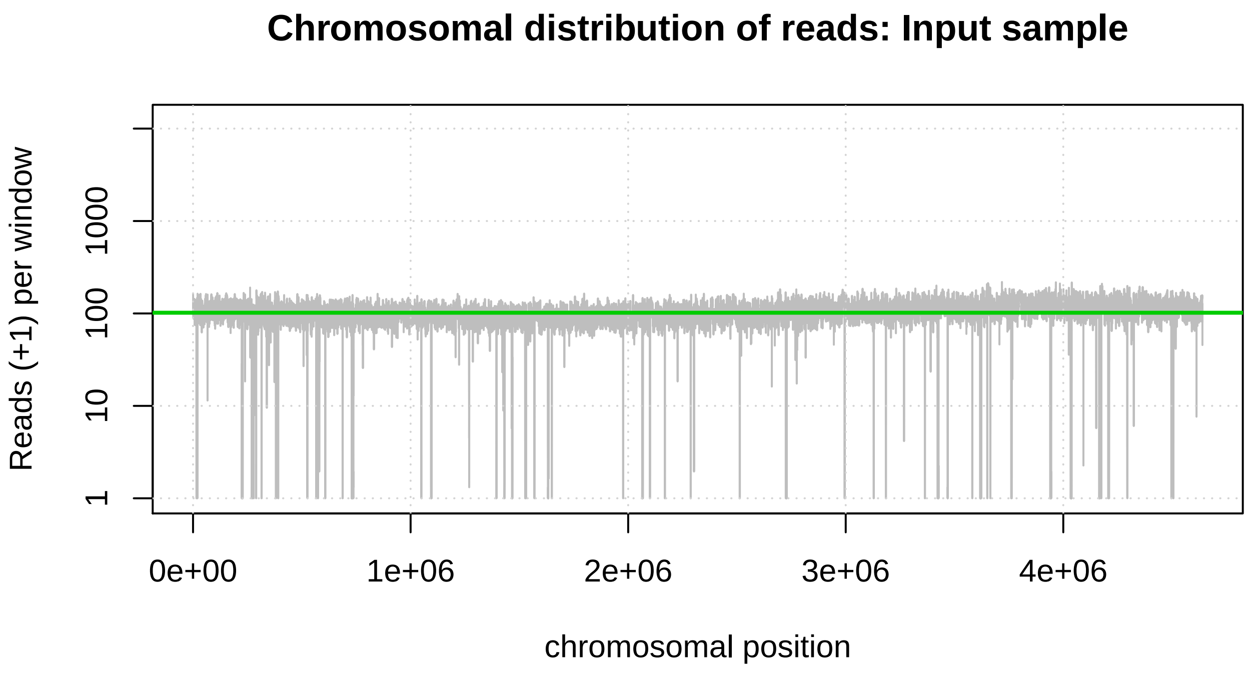
```
## Compute the middle position of each window (do use as coordinate for plots)
chip.midpos <- (chip.bedg[,3] +  chip.bedg[,2])/2
input.midpos <- (input.bedg[,3] +  input.bedg[,2])/2

par(mfrow=c(2,1)) ## Split the plot window in two vertical panels

## Note: we add 1 to the counts to avoid problems with the logarithmic scale
plot(chip.midpos,
     chip.counts + 1,
     col="blue", log="y",
     main="Chromosomal distribution of reads: Test sample",
     type="l",
     xlab="chromosomal position",
     ylab="Reads (+1) per window", ylim=count.range+1)
grid()
abline(h=mean(chip.counts), col="#00CC00", lwd=2)

plot(input.midpos,
     norm.input + 1,
     col="gray", log="y",
     main="Chromosomal distribution of reads: Input sample",
     type="l",
     xlab="chromosomal position",
     ylab="Reads (+1) per window", ylim=count.range+1)
grid()
abline(h=mean(norm.input), col="#00CC00", lwd=2)
```





```
par(mfrow=c(1,1))
```

6. For each bin, compute the following statistics, and collect them in a table (one row per bin, one column per statistics):

Note: each of these statistics can be computed with a single operation – in R, you should avoid loops whenever possible.

a. Number of reads in the test (FNR ChIP-seq)
b. Number of reads in the input
c. Normalized number of reads in the input (norm.input)
d. Reads per base in the normalized input
e. Fold enrichment (ratio between test and normalized input)
f. Log-fold enrichment ($\log_{10}$ of the fold enrichment)
g. P-value of the test reads, using a Poisson distribution with local lambda estimate hexplain your model).
h. P-value of the test reads, using a binomial distribution (explain your model).
i. P-value of the test reads, using a hypergeometric distribution (explain your model).

```
# a.  Number of reads in the test (FNR ChIP-seq)
# b.  Number of reads in the input
# c.  Normalized number of reads in the input (norm.input)
read.stats <- cbind(chip.bedg, input.counts, norm.input)

# d.  Reads per base in the normalized input
read.stats$input.norm.rpb <- read.stats$norm.input/window.size

# e.  Fold enrichment (ratio between test and normalized input)
read.stats$fold <- read.stats$counts/read.stats$norm.input

# f.  Log-fold enrichment (log(10) of the fold enrichment)
read.stats$log.fold <- log(base=10, read.stats$fold)

head(read.stats)
```

```
##                          chrom start  end counts input.counts norm.input
## 1 gi|49175990|ref|NC_000913.2|     0  200   1594          514   162.2312
## 2 gi|49175990|ref|NC_000913.2|   200  400    834          490   154.6562
## 3 gi|49175990|ref|NC_000913.2|   400  600    222          392   123.7250
## 4 gi|49175990|ref|NC_000913.2|   600  800    172          374   118.0438
## 5 gi|49175990|ref|NC_000913.2|   800 1000    123          316    99.7375
## 6 gi|49175990|ref|NC_000913.2|  1000 1200    116          352   111.1000
##   input.norm.rpb     fold   log.fold
## 1      0.8111562 9.825481 0.99235380
## 2      0.7732813 5.392605 0.73179858
## 3      0.6186250 1.794302 0.25389551
## 4      0.5902188 1.457087 0.16348545
## 5      0.4986875 1.233237 0.09104663
## 6      0.5555000 1.044104 0.01874393
```

P-value of the test reads, using a hypergeometric distribution (explain your model)

The classical model for the hypergeometric is the drawing of a set of balls in an urn containing m black ("marked") and n white ("non-marked") balls. We will apply an analogous reasoning, by considering that we draw at random a set of reads among a population combining all the reads of the two librarires, where m is the total number of reads in the test library (i.e. the FNR-chipped sample), and n the total number of reads in the control library ("input"").

$$m = n_{\text{FNR}} = \sum_{i=1}^{w} x_{\text{FNR},i} = 2,622,163$$

$$n = n_{\text{input}} = \sum_{i=1}^{w} x_{\text{input},i} = 7,480,400$$

Where w is the number of windows on the genome ($w = 23,199$ for the counts per 200bp windows), and x_i^{FNR}, x_i^{input} are respectively the read counts in the i^{th} window for the FNR and input libraries.

We will now focus on one particular window i, for which we know the read counts in the FNR and input samples, respectively. Let us define k (the size of the selection) as the number of reads falling into this window, irrespective of the fact that they belong to the FNR or to the input library.

$$k_i = x_{\text{FNR},i} + x_{\text{input},i}$$

We can then compute, for each window i, the hypergeometric p-value of the number of FNR reads (x_i):

$$pvalue_{\text{hyper}}(x_i|m, n, k_i) = \sum_{j=x_i}^{\min(m,n)} P_{\text{hyper}}(j|m, n, k_i)$$

According to the number of reads in the FNR (m) and input (n) libraries, we can also compute the prior probability for a read to belong to the FNR library:

$$p_{\text{FNR}} = \frac{n_{\text{FNR}}}{n_{\text{FNR}} + n_{\text{input}}} = \frac{2,622,163}{2,622,163 + 7,480,400} = 0.2596$$

```
## Compute the prior probabilities
n.FNR = sum(chip.counts) ## Number of reads in FNR library
print(paste("Marked = n.FNR = ", n.FNR))
```

```
## [1] "Marked = n.FNR =  2622163"
```

```
n.input = sum(input.counts) ## Number of reads in the input library
print(paste("Non-marked = n.input = ", n.input))
```

```
## [1] "Non-marked = n.input =  7480400"
```

```
p.FNR = n.FNR/(n.FNR + n.input) ## Prior probability to belong to the FNR library
print(paste("Prior probability for the FNR library=", format(p.FNR, digits=4)))
```

```
## [1] "Prior probability for the FNR library= 0.2596"
```

```
read.stats$phyper <- phyper(chip.counts-1,
                               m=n.FNR,
                               n=n.input,
                               k=chip.counts+input.counts,
                               lower=FALSE, log=FALSE)
head(read.stats)
```

```
##                          chrom start  end counts input.counts norm.input
## 1 gi|49175990|ref|NC_000913.2|     0  200   1594          514   162.2312
## 2 gi|49175990|ref|NC_000913.2|   200  400    834          490   154.6562
## 3 gi|49175990|ref|NC_000913.2|   400  600    222          392   123.7250
## 4 gi|49175990|ref|NC_000913.2|   600  800    172          374   118.0438
## 5 gi|49175990|ref|NC_000913.2|   800 1000    123          316    99.7375
## 6 gi|49175990|ref|NC_000913.2|  1000 1200    116          352   111.1000
##   input.norm.rpb     fold   log.fold        phyper
## 1      0.8111562 9.825481 0.99235380  0.000000e+00
## 2      0.7732813 5.392605 0.73179858 7.554558e-176
## 3      0.6186250 1.794302 0.25389551  1.621883e-08
## 4      0.5902188 1.457087 0.16348545  2.148692e-03
## 5      0.4986875 1.233237 0.09104663  1.754887e-01
## 6      0.5555000 1.044104 0.01874393  7.338414e-01
```

An alternative (but equivalent) reasoning to model hypergeometric p-value

An alternative way to model the probabilities of read counts in FNR and input libraries is to consider as the universe all the reads from both libraries (as above), but to consider as “marked” those reads falling into a given specific window i, irrespective of their library.

$$m_i = x_{\text{FNR},i} + x_{\text{input},i}$$

All the reads falling into other windows are considered as “non-marked”.

$$n_{\text{FNR}} = \sum_{i=1}^{w} x_{\text{FNR},i}$$

$$n_{\text{input}} = \sum_{i=1}^{w} x_{\text{input},i}$$

$$n_i = n_{\text{FNR}} + n_{\text{input}} - x_{\text{FNR},i} - x_{\text{input},i}$$

We now consider that the FNR library is a random sampling of reads among the two combined libraries (FNR + input).

$$k = n_{\text{FNR}} = \sum_{i=1}^{w} x_{\text{FNR},i} = 2,622,163$$

We can compute the probability for this library to contain at least x_i marked reads in the window i.

$$pvalue_{\text{hyper}}(x_i|m_i, n_i, k) = \sum_{j=x_i}^{\min(m,n)} P_{\text{hyper}}(j|m_i, n_i, k)$$

Although the reasoning is different, we can easily check that the numerical result is identical, by virtue of the symmetry of the hypergeometric formula.

```
n.FNR <- sum(chip.counts) ## Size of the FNR library
n.input <- sum(input.counts) ## Size of the input library
N <-n.FNR  + n.input ## Total number of read counts

## A vector indicating as "marked" the number of test+input counts in each window.
## Each window will be considered successively as the "marked" reads for the hypergeometric
m.per.window <- chip.counts + input.counts

## Similarly , we define the number of no-marked reads per window, i.e. those not belonging to the cons
n.per.window <- N - m.per.window

read.stats$phyper.bis <- phyper(chip.counts-1,
                             m=m.per.window,
                             n=n.per.window,
                             k=n.FNR,
                             lower=FALSE,log=FALSE)

head(read.stats)
```

```
##                          chrom start  end counts input.counts norm.input
## 1 gi|49175990|ref|NC_000913.2|     0  200   1594          514   162.2312
## 2 gi|49175990|ref|NC_000913.2|   200  400    834          490   154.6562
## 3 gi|49175990|ref|NC_000913.2|   400  600    222          392   123.7250
## 4 gi|49175990|ref|NC_000913.2|   600  800    172          374   118.0438
## 5 gi|49175990|ref|NC_000913.2|   800 1000    123          316    99.7375
## 6 gi|49175990|ref|NC_000913.2|  1000 1200    116          352   111.1000
##   input.norm.rpb     fold   log.fold        phyper    phyper.bis
## 1      0.8111562 9.825481 0.99235380  0.000000e+00  0.000000e+00
## 2      0.7732813 5.392605 0.73179858 7.554558e-176 7.554558e-176
## 3      0.6186250 1.794302 0.25389551  1.621883e-08  1.621883e-08
## 4      0.5902188 1.457087 0.16348545  2.148692e-03  2.148692e-03
## 5      0.4986875 1.233237 0.09104663  1.754887e-01  1.754887e-01
## 6      0.5555000 1.044104 0.01874393  7.338414e-01  7.338414e-01
```

P-value of the test reads, using a Poisson distribution with local lambda estimate (explain your model)

The simplest way to compute a P-value is to use the Poisson distribution, which requires a single parameter (λ) indicating the expected number of hits.

We already computed the normalized input, which indicates the number of reads we would expect in each window, assuming the factor would not bind anything specific. We can thus use this normalized input as a window-specific estimator of the $lambda_i$ parameter.

$$\lambda_i = x_{\text{norm.input},i} = x_{\text{input},i} \cdot \frac{n_{\text{FNR}}}{n_{\text{input}}}$$

```
# sum(read.stats$norm.input != read.stats$input.counts*m/n) == 0
## Compute the Poisson p-value
read.stats$ppois <- ppois(q=read.stats$counts-1,
                          lambda=read.stats$norm.input,
                          lower.tail=FALSE, log=FALSE)
```

P-value of the test reads, using a binomial distribution (explain your model).

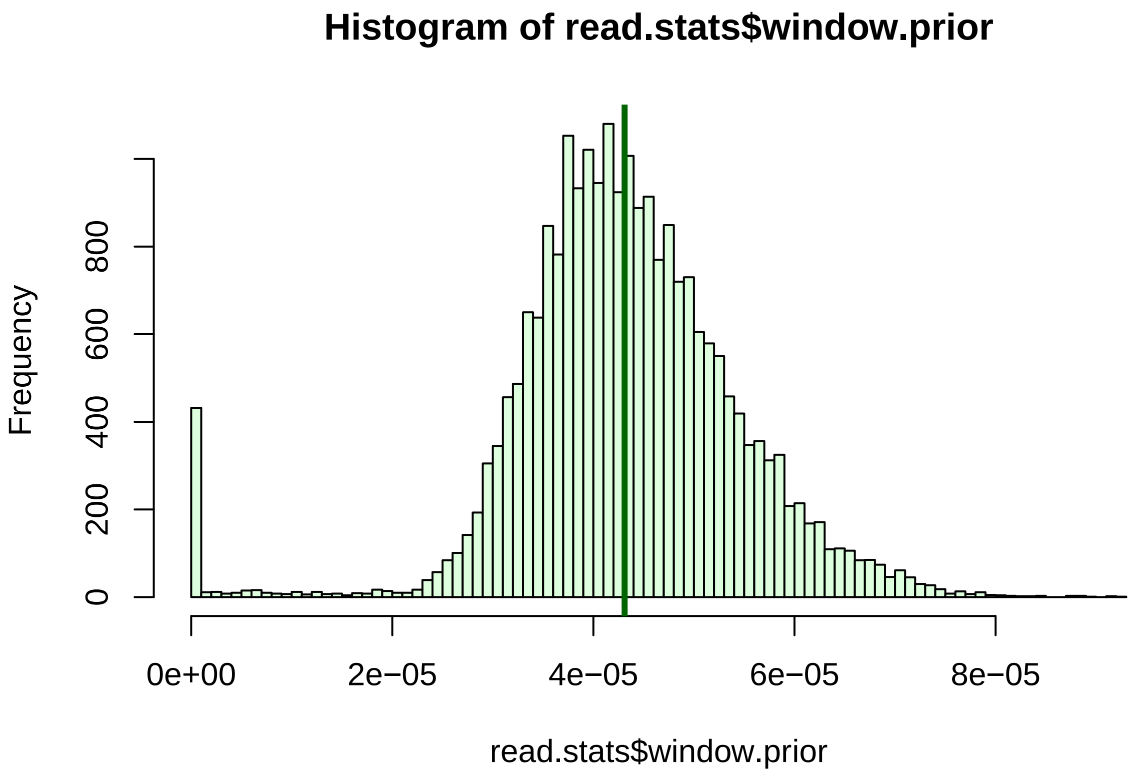
For the binomial, we can consider each read as a trial, which can either result in a "success" if the read falls within the window of interest, and a "failure" otherwise. Our input library will serve us to estimate a window-specific prior probability (p_i^{window}), indicating the probability for a read to fall within the i^{th} positional window.

$$\hat{p_i^{window}} = \frac{x_i^{input}}{\sum_{j=1}^{w} x_j^{input}}$$

```r
## Prior probability for a read to fall within a given window
read.stats$window.prior <- input.counts/sum(input.counts)
print (range(read.stats$window.prior))
```

```
## [1] 0.000000e+00 9.237474e-05
```

```r
hist(read.stats$window.prior, breaks=100, col="#DDFFDD")
abline(v=mean(read.stats$window.prior), lwd=3, col="darkgreen")
```



```r
## Prior probability per window if one would assume equiprobability (which we don't)
equi.p = (1/nrow(read.stats))
print(paste("Equiprobable window prior =", equi.p))
```

```
## [1] "Equiprobable window prior = 4.3105306263201e-05"
## Verification: this equiprobable prior should equal the mean probability of the window-specific priors
mean.p = mean(read.stats$window.prior)
print(paste("Mean window prior = ", mean.p))
```

```
## [1] "Mean window prior =  4.3105306263201e-05"
```

```
print(paste("Equality ? ", mean.p == equi.p))
```

```
## [1] "Equality ?  TRUE"
```

```
## Compute the P-value of the test reads, using a binomial distribution (explain your model).
read.stats$pbinom <- pbinom(q=chip.counts-1,
                            size=sum(chip.counts),
                            prob=read.stats$window.prior,
                            lower=FALSE, log=FALSE)

## Display the first rows of the stats table
head(read.stats)
```

```
##                          chrom start  end counts input.counts norm.input
## 1 gi|49175990|ref|NC_000913.2|     0  200   1594          514   162.2312
## 2 gi|49175990|ref|NC_000913.2|   200  400    834          490   154.6562
## 3 gi|49175990|ref|NC_000913.2|   400  600    222          392   123.7250
## 4 gi|49175990|ref|NC_000913.2|   600  800    172          374   118.0438
## 5 gi|49175990|ref|NC_000913.2|   800 1000    123          316    99.7375
## 6 gi|49175990|ref|NC_000913.2|  1000 1200    116          352   111.1000
##   input.norm.rpb     fold   log.fold        phyper    phyper.bis
## 1      0.8111562 9.825481 0.99235380  0.000000e+00  0.000000e+00
## 2      0.7732813 5.392605 0.73179858 7.554558e-176 7.554558e-176
## 3      0.6186250 1.794302 0.25389551  1.621883e-08  1.621883e-08
## 4      0.5902188 1.457087 0.16348545  2.148692e-03  2.148692e-03
## 5      0.4986875 1.233237 0.09104663  1.754887e-01  1.754887e-01
## 6      0.5555000 1.044104 0.01874393  7.338414e-01  7.338414e-01
##           ppois window.prior        pbinom
## 1  0.000000e+00 6.871290e-05  0.000000e+00
## 2 8.799240e-318 6.550452e-05 3.087108e-287
## 3  1.234871e-15 5.240361e-05  2.113602e-11
## 4  1.922915e-06 4.999733e-05  3.610691e-04
## 5  1.334553e-02 4.224373e-05  1.332241e-01
## 6  3.334319e-01 4.705631e-05  7.589231e-01
```

7. Draw some dot plots to compare the different statistics (fold enrichment, log-fold, p-values with the different models).

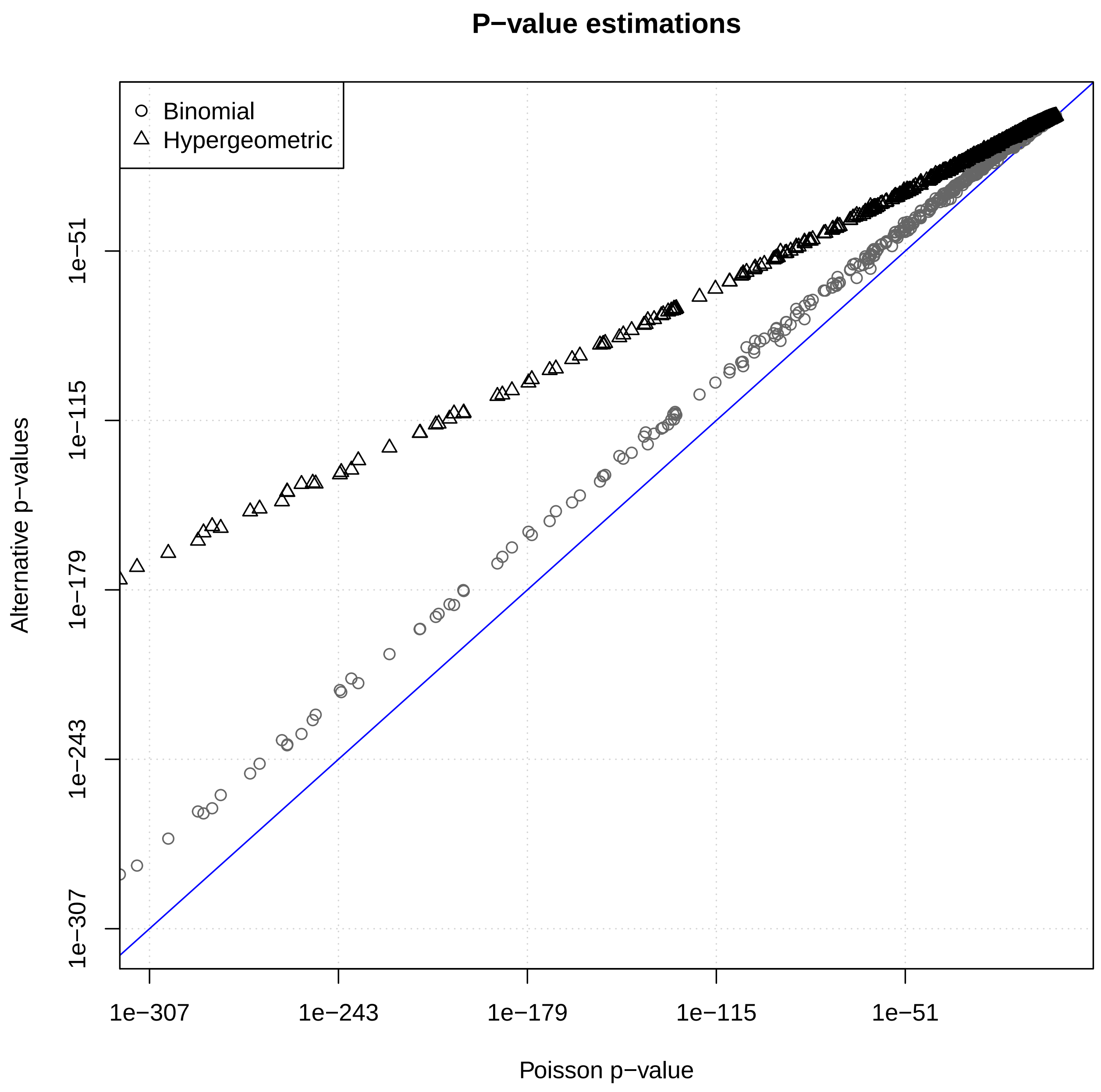
```
## Plot the Poisson versus binomial p-values
plot(read.stats[,c("ppois","pbinom")],log="xy",
     col="#666666",
     main="P-value estimations",
     xlab="Poisson p-value", ylab="Alternative p-values",
     panel.first=c(grid(),abline(a=0,b=1, col="blue")))
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 69 x values <= 0 omitted
## from logarithmic plot

## Warning in xy.coords(x, y, xlabel, ylabel, log): 65 y values <= 0 omitted
## from logarithmic plot
```

```
legend("topleft", legend=c("Binomial", "Hypergeometric"),pch=c(1,2))

## Plot the hypergeometric versus Poisson p-values
lines(read.stats[,c("ppois","phyper")],type="p", pch=2)
```

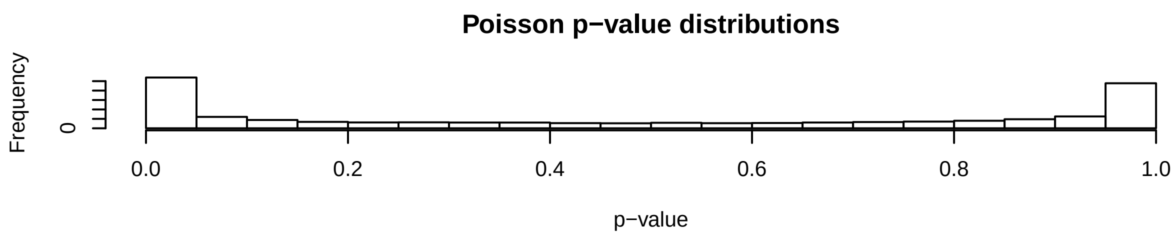


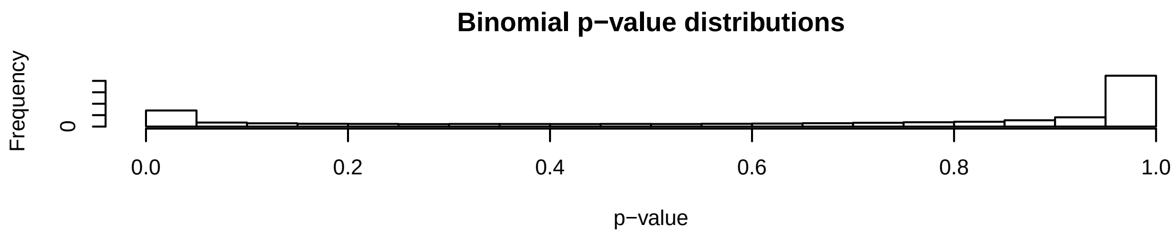
Distribution of measured p-values

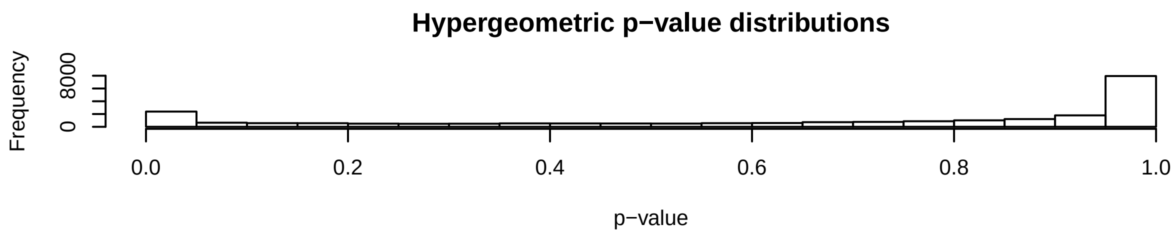
```
par(mfrow=c(3,1))
## Draw histograms of p-value distributions
hist(read.stats$ppois, breaks=20,
     main ="Poisson p-value distributions",
     xlab="p-value", ylab="Frequency")


## Draw histograms of p-value distributions
hist(read.stats$pbinom, breaks=20,
     main ="Binomial p-value distributions",
     xlab="p-value", ylab="Frequency")


## Draw histograms of p-value distributions
hist(read.stats$phyper, breaks=20,
     main ="Hypergeometric p-value distributions",
     xlab="p-value", ylab="Frequency")
```







8. Discuss the results.

The binomial and Poisson p-values are very close to each other

On the plot comparing p-values obtained with the three respective models, the Poisson and binomial distribution give remarkably close estimations of the p-value. This is not surprizing, since we are in the ideal conditons under which the binomial converges towards a Poisson :

1. The probability of success should be very small ($p \ll 1$). This is the case, since our prior probabilities range from 0 (windows with not a single read in the input) to $1e^{-4}$.
2. The number of trials should be very large ($n \gg 1$). In our binomial model, the number of trials is, for each window, the sum of reads from the FNR (test) and input libraries, this sum gives on average 435 reads per window, and the very large majority of windows have at least 200 reads.

The Poisson is thus a very good approximation of the binomial.

Hypergeometric p-values are much higher than binomial or Poisson p-values

In contrast, the hypergeometric p-values shows a strong departure from the Poisson and binomial. There is a relatively consistency between the relative p-values: although the relationship between hypergeometric and Poisson p-values is not monotonous, there seems to be a strong linear correlation on the logaritmic plot. However, for the smallest p-values, it is several tens of orders of magnitudes higer than the Poisson and binomial p-values.

The reason is that our hypergeometric model relies on a different basis than the Poisson/binomial ones. For the hypergeometric, the probability for a read to fall within a given window is estimated from both FNR and input libraries:

$$p_i^{hyper} = \frac{m_! + n_i}{m + n}$$