Stats withb R and RStudio

Practical: basic stats for peak calling

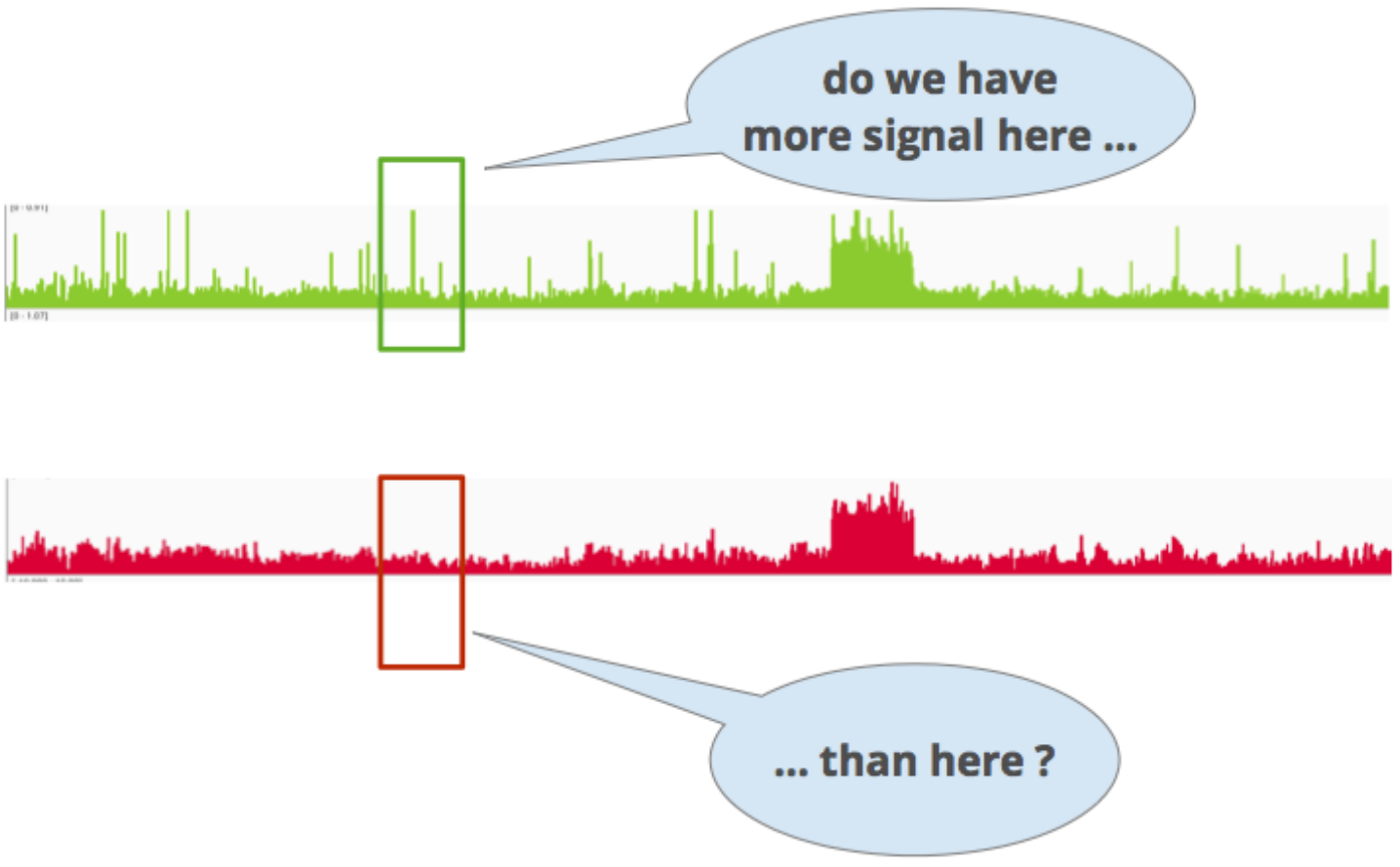
Jacques van Helden

2016-11-23

1. Introduction
2. Data loading
3. Exploring the data with basic plots
4. Step-by-step walk to the significance

Introduction

Peak-calling: question



Data loading

Defining the data directory

We will first define the URL from which the data can be downloaded, by concatenating the URL fo the course with the path to our dataset.
To concatenate paths, it is *recommended* to use the **R** command `file.path()`.

```
# url.course <- "http://jvanheld.github.io/stats_avec_RStu
url.course <- "~/stats_avec_RStudio_EBA/"
url.data <- file.path(url.course, "practicals", "02_peak-ca
```

Loading a data table

R enables to download data directly from the Web.
Load counts per window in chip sample.

```
## Define URL of the ChIP file
chip.bedg.file <- file.path(url.data, "FNR_200bp.bedg")

## Load the file content in an R data.frame
chip.bedg <- read.table(chip.bedg.file)

## Set column names
names(chip.bedg) <- c("chrom", "start", "end","counts")
```

Exploring a data frame: dim()

Before anything else, let us inspect the size of the data frame, in order to check that it was properly lodaded.

```
dim(chip.bedg)
```

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

Checking the n first rows: head()

The function `head()` displays the first rows of a table.

```
head(chip.bedg, n = 5)
```

```
##                                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
```

Checking the n last rows: tail()

The function `tail()` displays the last rows of a table.

```
tail(chip.bedg, n = 5)
```

```
##                                      chrom   start     end count
## 23195 gi|49175990|ref|NC_000913.2| 4638800 4639000    15
## 23196 gi|49175990|ref|NC_000913.2| 4639000 4639200     9
## 23197 gi|49175990|ref|NC_000913.2| 4639200 4639400     9
## 23198 gi|49175990|ref|NC_000913.2| 4639400 4639600    22
## 23199 gi|49175990|ref|NC_000913.2| 4639600 4639675    18
```

Viewing a table

The function `View()` displays the full table in a user-friendly mode.

```
View(chip.bedg)
```

Selecting arbitrary rows

```
chip.bedg[100:105,]
```

```
##                                    chrom start   end counts
## 100 gi|49175990|ref|NC_000913.2| 19800 20000    21
## 101 gi|49175990|ref|NC_000913.2| 20000 20200     0
## 102 gi|49175990|ref|NC_000913.2| 20200 20400     0
## 103 gi|49175990|ref|NC_000913.2| 20400 20600   108
## 104 gi|49175990|ref|NC_000913.2| 20600 20800   229
## 105 gi|49175990|ref|NC_000913.2| 20800 21000   245
```

Selecting arbitrary columns

```
chip.bedg[100:105, 2]
```

```
## [1] 19800 20000 20200 20400 20600 20800
```

```
chip.bedg[100:105, "start"]
```

```
## [1] 19800 20000 20200 20400 20600 20800
```

```
chip.bedg[100:105, c("start", "counts")]
```

```
##      start counts
## 100 19800     21
## 101 20000      0
## 102 20200      0
## 103 20400    108
## 104 20600    229
## 105 20800    245
```

Adding columns

We can add columns with the result of computations from other columns.

```
chip.bedg$midpos <- (chip.bedg$start + chip.bedg$end)/2
head(chip.bedg)
```

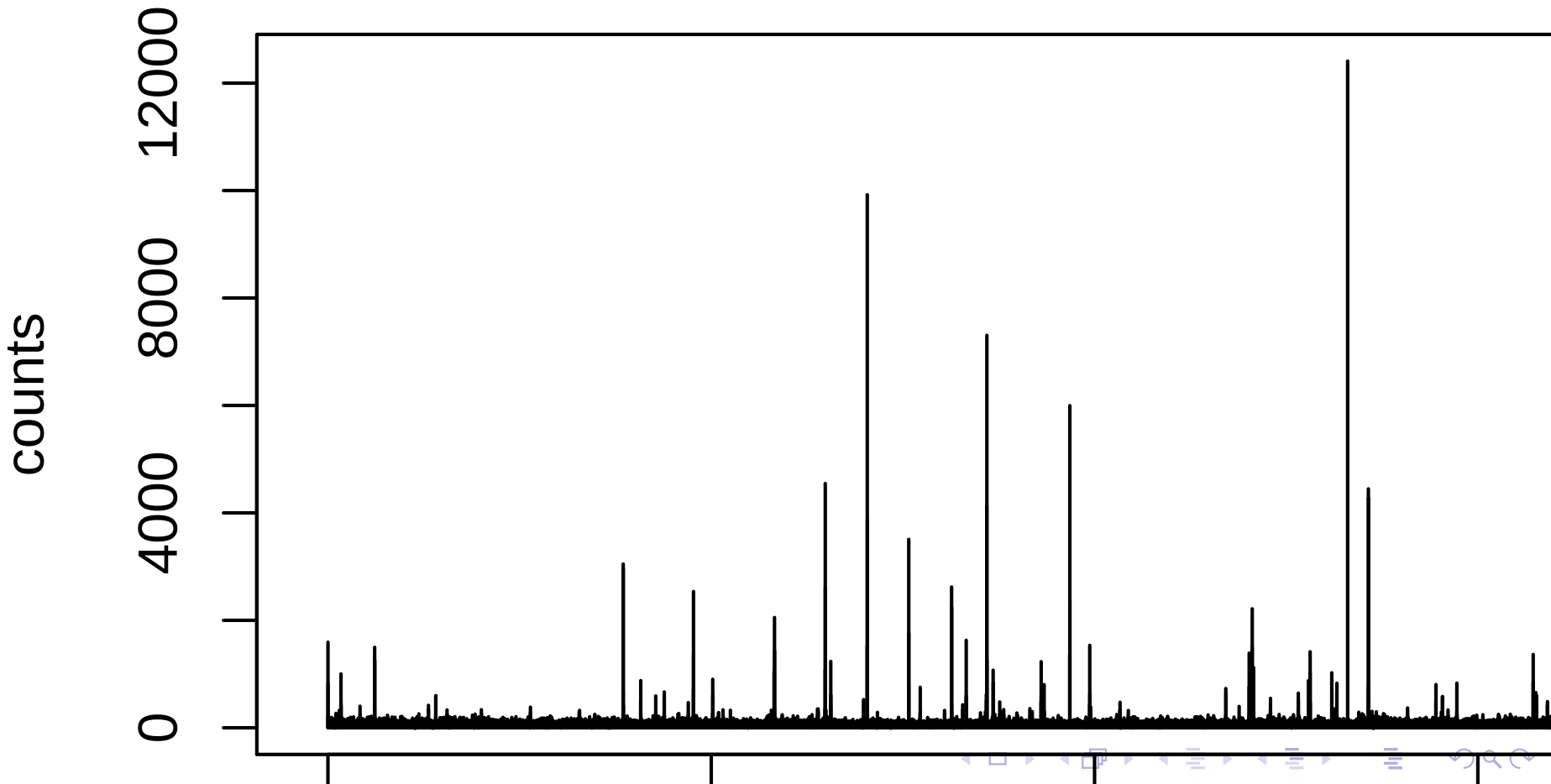
```
##                                  chrom start  end counts midpos
## 1 gi|49175990|ref|NC_000913.2|     0   200 1594    100
## 2 gi|49175990|ref|NC_000913.2|   200   400  834    300
## 3 gi|49175990|ref|NC_000913.2|   400   600  222    500
## 4 gi|49175990|ref|NC_000913.2|   600   800  172    700
## 5 gi|49175990|ref|NC_000913.2|   800  1000  123    900
## 6 gi|49175990|ref|NC_000913.2|  1000  1200  116   1100
```

Exploring the data with basic plots

Plotting a density profile

We can readily print a plot with the counts per window.

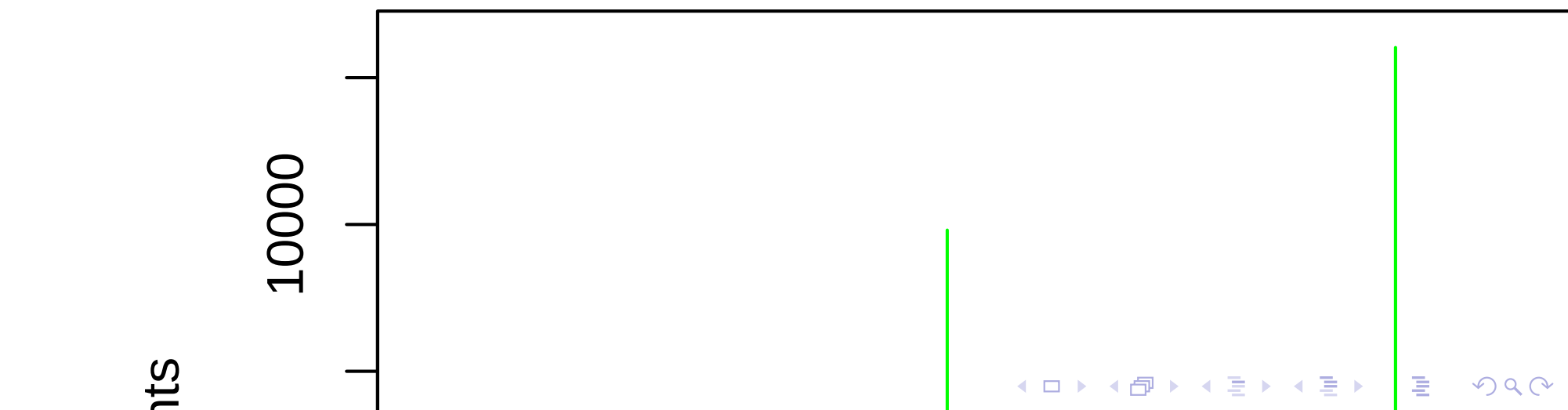
```
plot(chip.bedg[, c("midpos", "counts")], type="h")
```



Plotting a density profile

Let us improve the plot

```
plot(chip.bedg[, c("midpos", "counts")], type="h",
     col="green", xlab="Genomic position (200bp windows)",
     ylab= "Read counts",
     main="FNR ChIP-seq")
```



Exercise: exploring the background

We already loaded the count table for the FNR ChIP counts per window.
The background level will be estimated by loading counts per window in a genomic input sample. These counts are available in the same directory a file named `input_200bp.bedg`

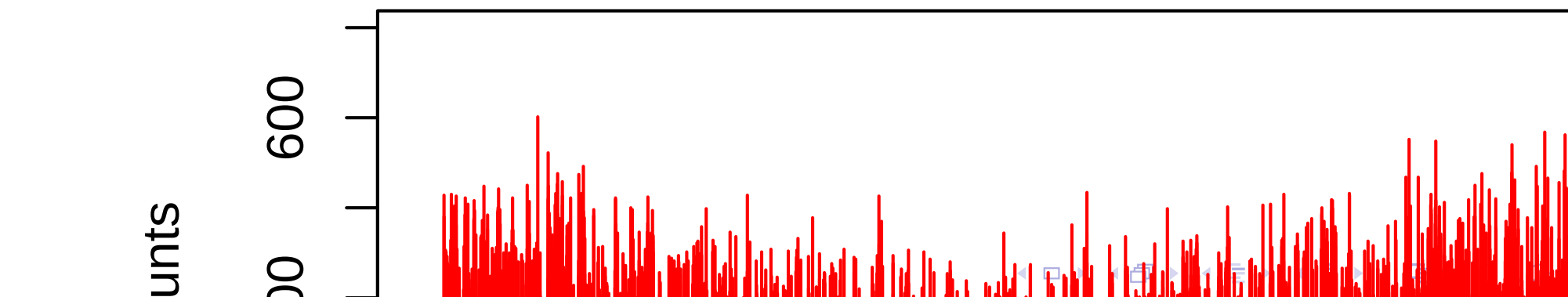
1. Load the counts per window in the input sample (genome sequencing).
2. Plot the density profile of the input
3. Compare chip-seq and input density profiles
4. Compare counts per window between chip-seq and input

Solution: loading the input counts per window

```
## Define URL of the input file
input.bedg.file <- file.path(url.data, "input_200bp.bedg")

## Load the file content in an R data.frame
input.bedg <- read.table(input.bedg.file)

## Set column names
names(input.bedg) <- c("chrom", "start", "end","counts")
```

Solution: plotting the input density profile

```
## Compute middle positions per window
input.bedg$midpos <- (input.bedg$start + input.bedg$end)/2

plot(input.bedg[, c("midpos", "counts")], type="h",
     col="red", xlab="Genomic position (200bp windows)",
     ylab= "Read counts",
     main="Background (genomic input)")
```

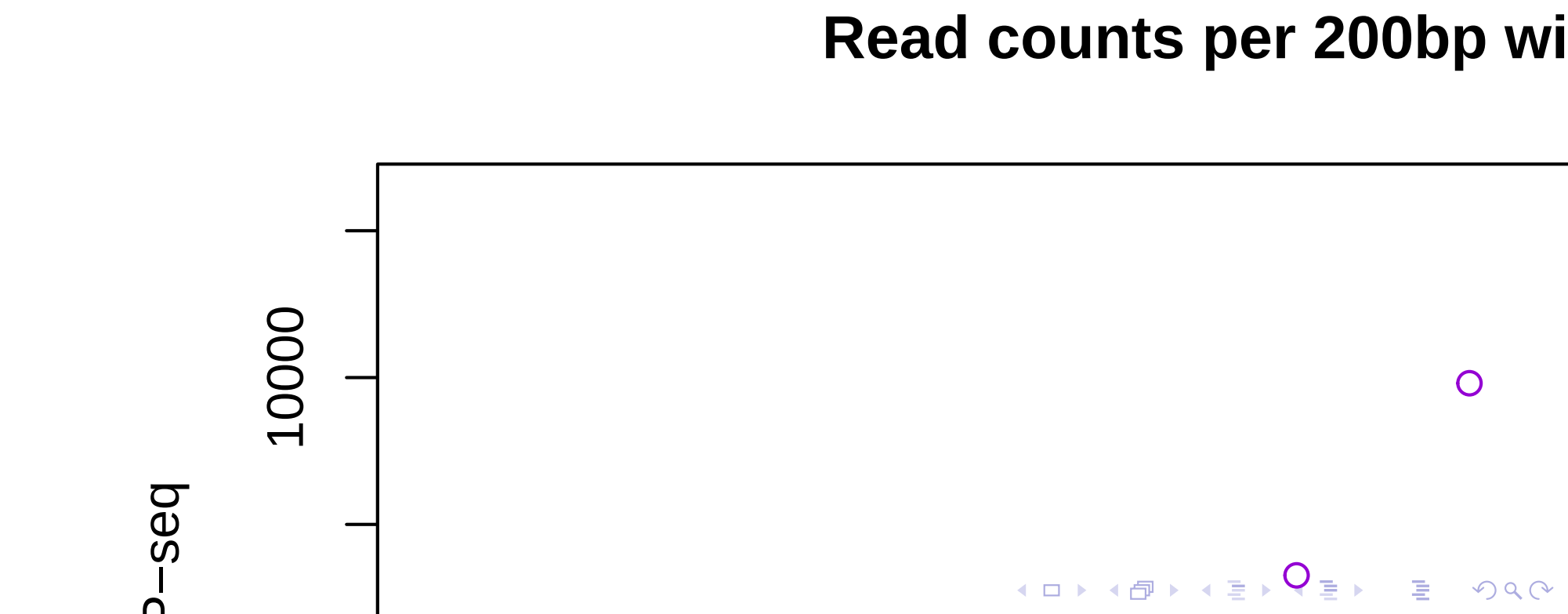


Solution: comparing chip-seq and background density profiles

```
par(mfrow=c(2,1)) ## Draw two panels on top of each other
plot(chip.bedg[, c("midpos", "counts")], type="h",
     col="green", xlab="Genomic position (200bp windows)",
     ylab= "Read counts",
     main="FNR ChIP-seq")
plot(input.bedg[, c("midpos", "counts")], type="h",
     col="red", xlab="Genomic position (200bp windows)",
     ylab= "Read counts",
     main="Background (genomic input)")
```

Solution: comparing counts per window between chip-seq and input

```
plot(input.bedg$counts, chip.bedg$counts, col="darkviolet",
     xlab="Genomic input", ylab="FNR ChIP-seq",
     main="Read counts per 200bp window")
```



Solution: comparing counts per window between chip-seq and input

In order to better highight the dynamic range, we can use a log-based representation

```
plot(input.bedg$counts, chip.bedg$counts, col="darkviolet",
     xlab="Genomic input", ylab="FNR ChIP-seq",
     main="Read counts per 200bp window",
     log="xy")
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 377 x v
## from logarithmic plot
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 403 y v
## from logarithmic plot
```

```
grid() ## add a grid
```

Questions

- On the ChIP-seq versus input plot, how would you define peaks ?
- Where would you place the limit between peaks and background fluctuations ?

Exercises

1. Think about further drawing modes to improve your perception of the differences between signal and background.
2. We will formulate (together) a reasoning path to compute a p-value for each peak.

Step-by-step walk to the significance

Merge the two tables

```
names(input.bedg)
```

```
## [1] "chrom"  "start"  "end"    "counts" "midpos"
```

```
count.table <- merge(chip.bedg, input.bedg, by=c("chrom", "
dim(count.table)
```

```
## [1] 23199     6
```

```
names(count.table)
```

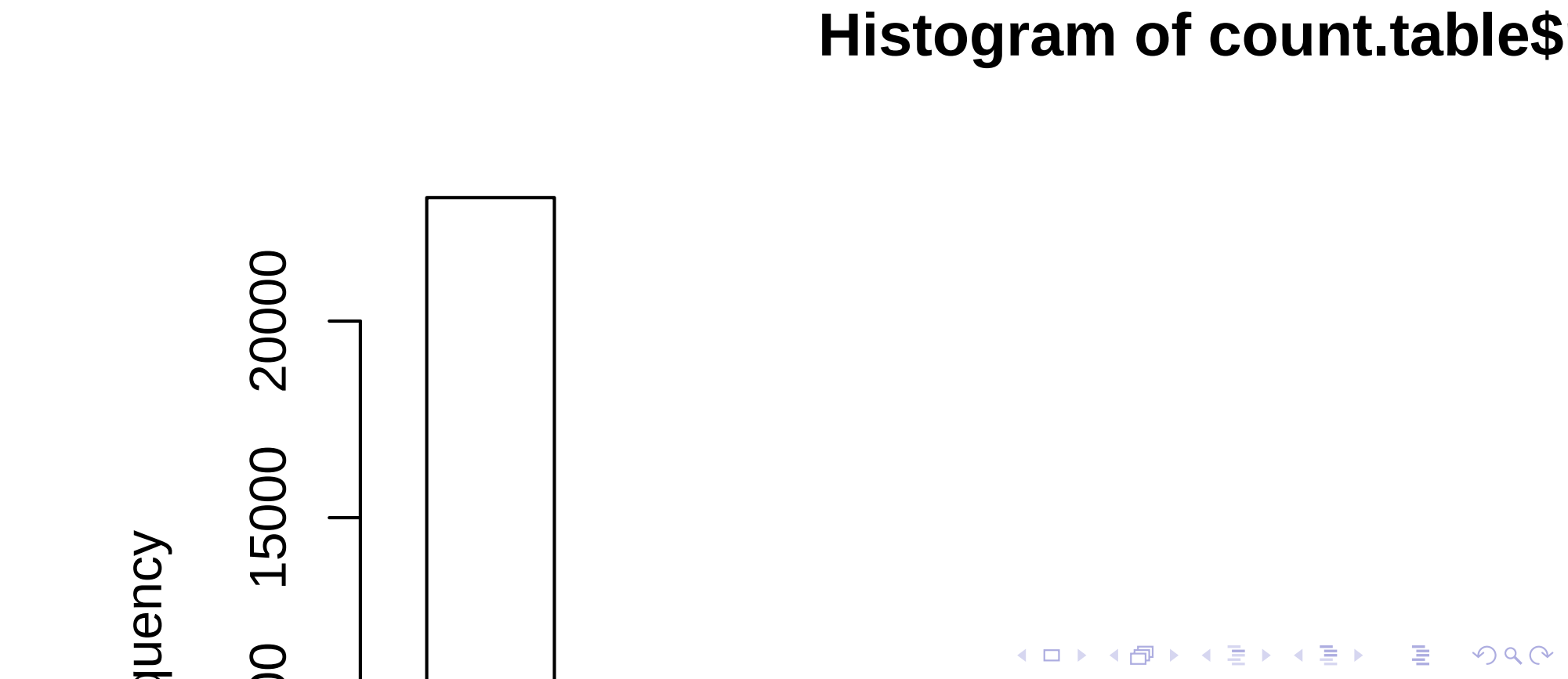
```
## [1] "chrom"         "start"         "end"           "midpos
## [5] "counts.chip"   "counts.input"
```

```
head(count.table)
```

```
##                                chrom   start     end  midpos c
## 1 gi|49175990|ref|NC_000913.2|     0     200     100
```

Ratios

```
count.table$ratio <- (count.table$counts.chip + 1) / (count

hist(count.table$ratio)
```



Input normalization

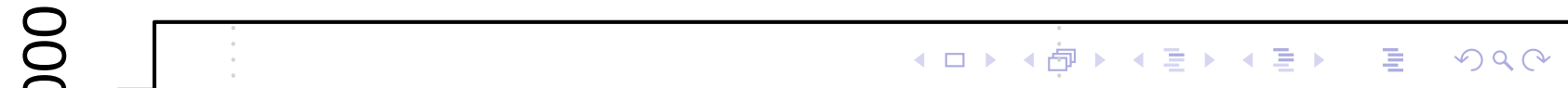
```
## Normalize input counts by library sizes
count.table$input.norm <- count.table$counts.input * sum(co

plot(x = count.table$input.norm + 1,
     y = count.table$counts.chip + 1, col="blue", log="xy",
     main="ChIP versus input",
     xlab="Scaled input counts",
     ylab="ChIP counts")
grid()
abline(a=0, b=1, col="black")
```

Input normalization by median

Why ? Robust to outliers.

```
## Normalize input counts by median count
count.table$input.norm <- count.table$counts.input * sum(co

plot(x = count.table$input.norm + 1,
     y = count.table$counts.chip + 1, col="blue", log="xy",
     main="ChIP versus input",
     xlab="Scaled input counts",
     ylab="ChIP counts")
grid()
abline(a=0, b=1, col="black")
```

ChIP versus input



Log-ratios

```
count.table$scaled.ratio <- (count.table$counts.chip +1 ) /

hist(count.table$scaled.ratio, breaks=2000, xlim=c(0, 5))
abline(v=1, col="darkgreen", lwd=3, lty="dashed")
abline(v=median(count.table$scaled.ratio), col="darkblue",
```