Seminar III: R/Bioconductor Rolexa

Luz Mayela Soto Jimenez

November 13, 2009

Installing the package I

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("Rolexa")
```

Introduction to Rolexa I

- This package provides an alternative base calling algorithm using model-based clustering (mclust) and probability theory to identify ambiguous bases and code them with IUPAC symbols
- We also select optimal sub-tags using a score based on information content to remove uncertain bases towards the ends of the reads.

Environment Variables: RolexaRun I

- The Rolexa package uses a RolexaRun object to store the various parameters of the run, and uses the ShortRead for manipulating data, in particular many Rolexa functions take a SolexaPath object as argument.

Loading the library I

```
> library("Rolexa")
> rolenv = SetModel(idsep = "_")
> GetModel(rolenv)

$MinimumTagLength
[1] 15

$SequencingLength
[1] 36

$Barcode
[1] 0

$HThresholds
```

Loading the library II

```
[1] 0.5849625 1.3219281 1.8073549

$IThresholds
 [1] 2.058894 2.115477 2.169925 2.222392
 [5] 2.273018 2.321928 2.369234 2.415037
 [9] 2.459432 2.502500 2.544321 2.584963
[13] 2.624491 2.662965 2.700440 2.736966
[17] 2.772590 2.807355 2.841302 2.874469
[21] 2.906891 2.938599 2.969626 3.000000
[25] 3.029747 3.058894 3.087463 3.115477
[29] 3.142958 3.169925 3.196397 3.222392
[33] 3.247928 3.273018 3.297681 3.321928

$PET
```

Loading the library III

```
[1] FALSE

$fit
[1] FALSE

$normal
[1] TRUE

$decorrelate
[1] "both"

$verbose
[1] 0
```

Loading the library IV

```
$colors
 [1] "black"      "green"
 [3] "blue"       "chocolate3"
 [5] "red"        "#007F7F"
 [7] "#66B20E"    "#7F7F00"
 [9] "#66338E"    "#7F007F"
[11] "#E6330E"    "#7F464E"
[13] "#7F6035"    "#6C5649"
[15] "#685F4C"    "gray"

$idsep
[1] "_"
```

Meaning of each parameter I

The meaning of these parameters is as follows:

- **MinimumTagLength** tags shorter than this will not be saved
- **SequencingLength** number of sequencing cycles, used to calculate the number of columns in files
- **Barcode** number of bases used as barcode at the beginning of the tag
- **HThresholds** entropy thresholds between 1 and 2-base ambiguities, 2 and 3-base ambiguities and 3-base ambiguity or undecided (the default is log2(c(1:5; 2:5; 3:5)))
- **IThresholds** total entropy thresholds, as a function of tag length (the default is log2(4+1 : 36=6))
- **PET** paired-end sequencing run

Meaning of each parameter II

- **fit** use full EM convergence instead of only one-step optimization if TRUE
- **normal** use tile-level normalization before base-calling if TRUE
- **decorrelate** use 'cycle'-level decorrelation procedure, 'channel'-level, 'both' or 'none'
- **idsep** character separating coordinate elds in sequence headers (default is ":")
- **verbose** print debug information if > 0

Base Calling I

```
> path = SolexaPath(system.file("extdata",
+     package = "ShortRead"))
> (seq_fastq = readFastq(path))

class: ShortReadQ
length: 256 reads; width: 36 cycles

> (int = readIntensities(path, pattern = "s_1_0001",
+     withVariability = FALSE))

class: SolexaIntensity
dim: 256 4 36
readInfo: SolexaIntensityInfo
intensity: ArrayIntensity
measurementError: not available
```

Base Calling II

```
> (seq = CombineReads(run = rolenv,
+     path = path, pattern = "s_1_0001_seq*"))

class: ShortRead
length: 256 reads; width: 36 cycles

> (theta = OptimizeAngle(int = int))[1:10,
+     ]

           [,1]     [,2]      [,3]
 [1,] 0.7767119 1.375080 0.4721182
 [2,] 0.7653824 1.377907 0.5618510
 [3,] 0.7276859 1.367992 0.5290140
 [4,] 0.7551378 1.384266 0.6453509
 [5,] 0.7349694 1.377229 0.6220983
 [6,] 0.7377151 1.383378 0.6556697
```

Base Calling III

```
 [7,] 0.7213154 1.377866 0.6412864
 [8,] 0.7685749 1.384597 0.6472642
 [9,] 0.7681729 1.387350 0.5537521
[10,] 0.7710965 1.379977 0.6961033
           [,4]
 [1,] 1.557188
 [2,] 1.570796
 [3,] 1.570796
 [4,] 1.570796
 [5,] 1.570796
 [6,] 1.564773
 [7,] 1.570796
 [8,] 1.570796
```

Base Calling IV

```
 [9,] 1.570796
[10,] 1.570796
```

```
> int = DeCorrelateChannels(int = int,
+     theta = theta)
> (rate = OptimizeRate(int = int))
```

```
[1] 0.01760222
```

```
> int = DeCorrelateCycles(int = int,
+     rate = rate)
> int2 = TileNormalize(run = rolenv,
+     int = int)
```

Base Calling II I

- The base calling algorithm its a gaussian mixture model to the four-dimensional intensity values from each cycle. Sequences from a previous base calling, if available, are used to seed the algorithm:

```
> (res = SeqScore(run = rolenv, int = int,
+     seqInit = seq, cycles = 1:36))$sread

  A DNAStringSet instance of length 256
      width seq
  [1]    36 TTGTTTTCATGTG...GTATTTGTTTGT
  [2]    36 TCCAAACTGGTAG...ATTCTCAAATCT
  [3]    36 TGCACCTGATAGG...GAGAGAGDAAGK
  [4]    36 TATGAGAGTAGCY...GWSGRKGTGKBY
  [5]    36 TAGTAGGTGTCCT...CAGCACGCCAAG
```

Base Calling II II

```
  [6]     36 GAGAGAACTGAAA...TGAGAAATAGAC
  [7]     36 GCAGAGACCCACA...CGGCTCCWGACC
  [8]     36 GAGATATTTATTG...TCTGTCATGCAA
  [9]     36 GGTGGAAAWAGGA...YTCYGCTTAYAT
  ...    ... ...
[248]     36 TGGGGAGMYGKGG...MYRTHHRWVVDK
[249]     36 GTGGAGGCTAGCA...CBTTGTGARGBA
[250]     36 GATTTTCAAAGTT...TGTTATCACCCG
[251]     36 GAAAATGAGAAAC...GACTTGAAAAAT
[252]     36 GGYATTTTCCTTT...RCTTTGKWGBDH
[253]     36 GGTAGGRAGAGCT...TTCTGCTTRRAW
[254]     36 GAAAAACGWGAAA...CACACTGTAGRA
[255]     36 GATTCCTTATGTG...TAATATTTCATC
[256]     36 GGATGAGAAGAAT...TCTCTAGCCACA
```

Filtering and Saving I

- The base calling results consist of a full-length tag with base quality entropy scores, which can then be filtered to extract the best sequence tag for each colony. This is where the parameters IThresholds comes into play:

```
> rolenv@MinimumTagLength = as.integer(1)
> (res2 = FilterResults(run = rolenv,
+     results = res))$sread
```

Filtering and Saving II

```
  A DNAStringSet instance of length 256
      width seq
  [1]    36 TTGTTTTCATGTG...GTATTTGTTTGT
  [2]    36 TCCAAACTGGTAG...ATTCTCAAATCT
  [3]    36 TGCACCTGATAGG...GAGAGAGDAAGK
  [4]    28 TATGAGAGTAGCYAATGCCACAAAGWSG
  [5]    36 TAGTAGGTGTCCT...CAGCACGCCAAG
  [6]    36 GAGAGAACTGAAA...TGAGAAATAGAC
  [7]    36 GCAGAGACCCACA...CGGCTCCWGACC
  [8]    36 GAGATATTTATTG...TCTGTCATGCAA
  [9]    21 GGTGGAAAWAGGAAGCAYCCC
  ...   ...  ...
[248]    10 TGGGGAGMYG
[249]    34 GTGGAGGCTAGCA...GGCBTTGTGARG
```

Filtering and Saving III

```
[250]     36 GATTTTCAAAGTT...TGTTATCACCCG
[251]     36 GAAAATGAGAAAC...GACTTGAAAAAT
[252]     30 GGYATTTTCCTTT...TATTTMRCTTTG
[253]     33 GGTAGGRAGAGCT...GTCTTCTGCTTR
[254]     36 GAAAAACGWGAAA...CACACTGTAGRA
[255]     36 GATTCCTTATGTG...TAATATTTCATC
[256]     36 GGATGAGAAGAAT...TCTCTAGCCACA
```

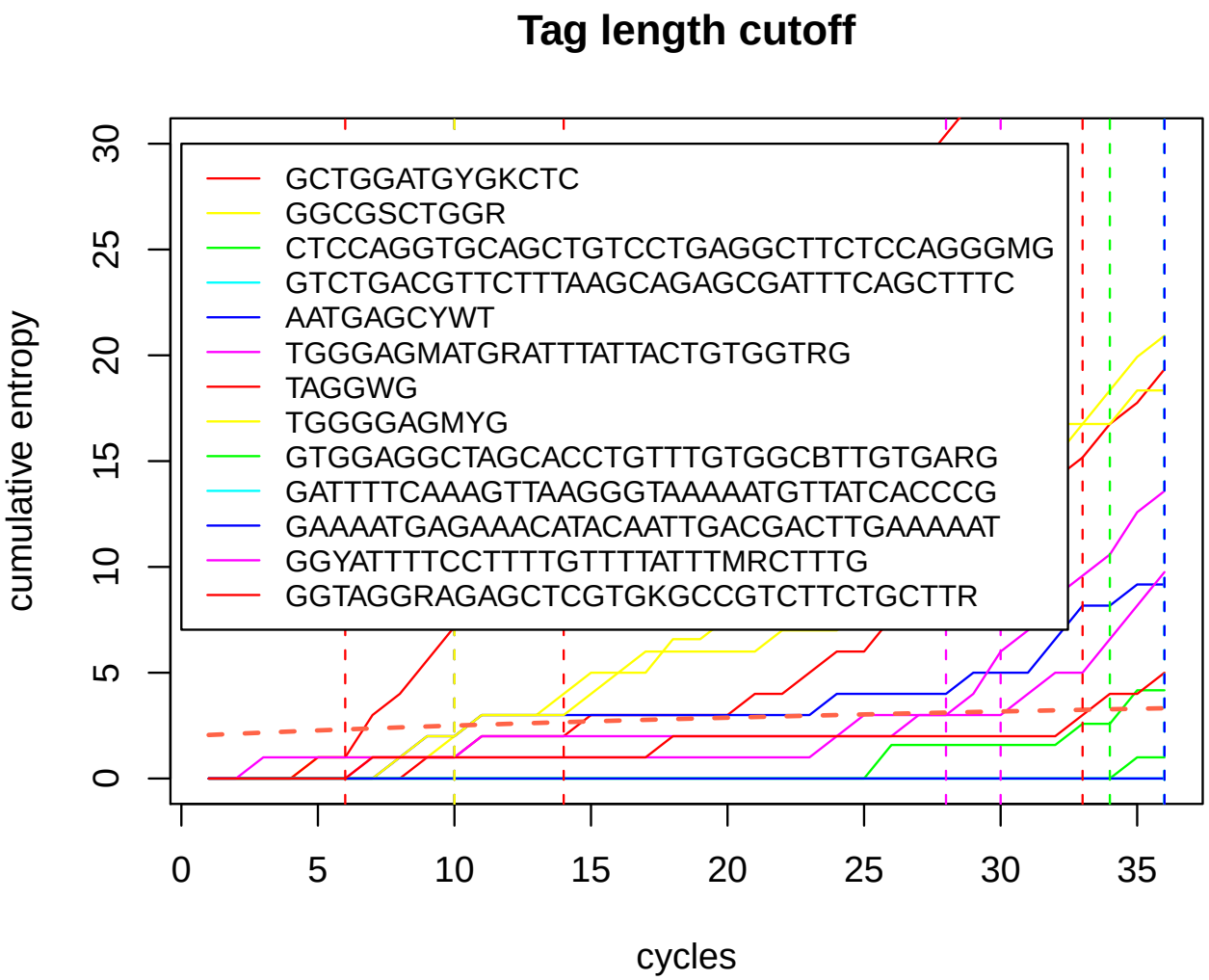
Plotting I

```
> str = as.matrix(res$sread[241:253])
> nt = DNA_ALPHABET
> post.entropy = matrix(0, nrow = nrow(str),
+     ncol = 36)
> post.entropy[which(str %in% nt[5:10])] = 1
> post.entropy[which(str %in% nt[11:14])] = log2(3)
> post.entropy[which(str == "N")] = 2
> matplot(1:36, y = apply(post.entropy,
+     1, cumsum), t = "l", lty = 1,
+     col = rainbow(6), ylim = c(0,
+         30), xlim = c(1, 36), xlab = "cycles",
+     ylab = "cumulative entropy",
+     main = "Tag length cutoff")
```

Plotting II

```
> lines(1:36, rolenv@IThresholds,
+       t = "l", lty = 2, lwd = 2,
+       col = "tomato")
> abline(v = nchar(res2$sread[241:253]),
+       col = rainbow(6), lty = 2)
> legend(x = 0, y = 30, res2$sread[241:253],
+       col = rainbow(6), lty = 1,
+       bg = "white", cex = 0.8)
```

Plotting III

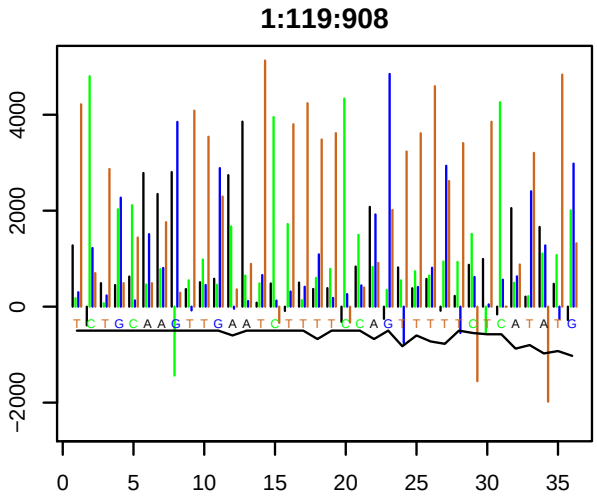


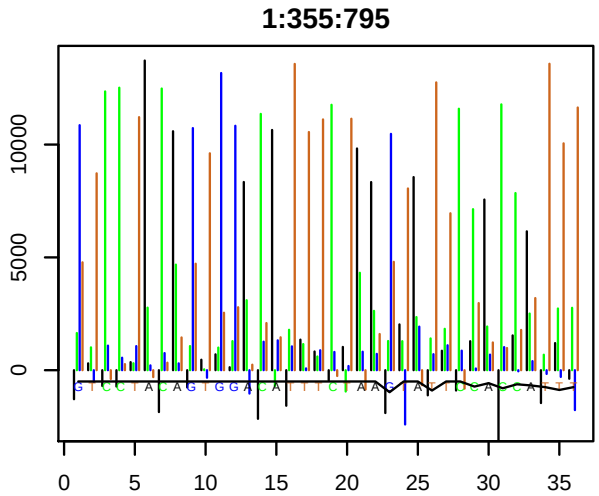
Diagnostic Plots I

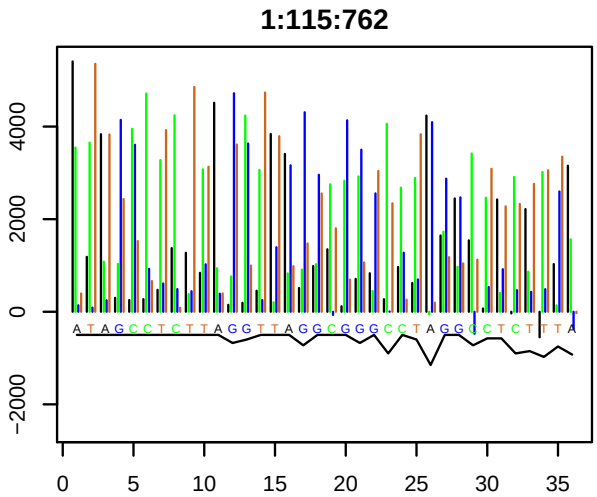
- There are multiple possibilities for evaluating the quality of the base calling, at the level of each sequence, tile or lane. Given a sequence tag, the corresponding raw intensities and a base quality score, we can use CombinedPlot:

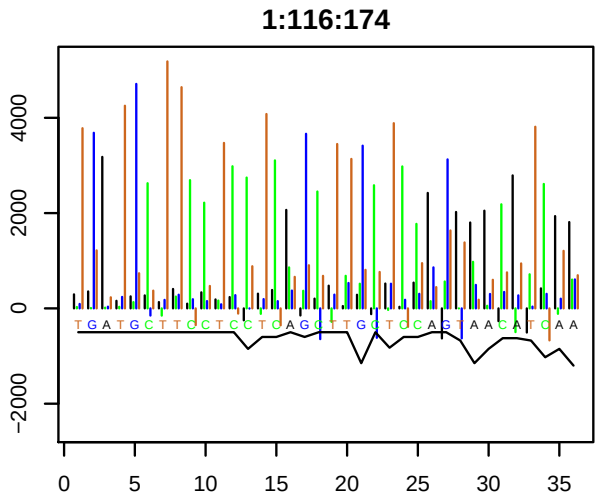
```
> CombinedPlot(run = rolenv, int = int,
+     seq = seq, scores = as(quality(seq_fastq),
+         "matrix"), colonies = sample(1:nrow(int),
+         4), par = list(mfrow = c(2,
+         2), cex = 0.6, mar = c(4,
+         4, 2, 1) + 0.1))
```

Diagnostic Plots II







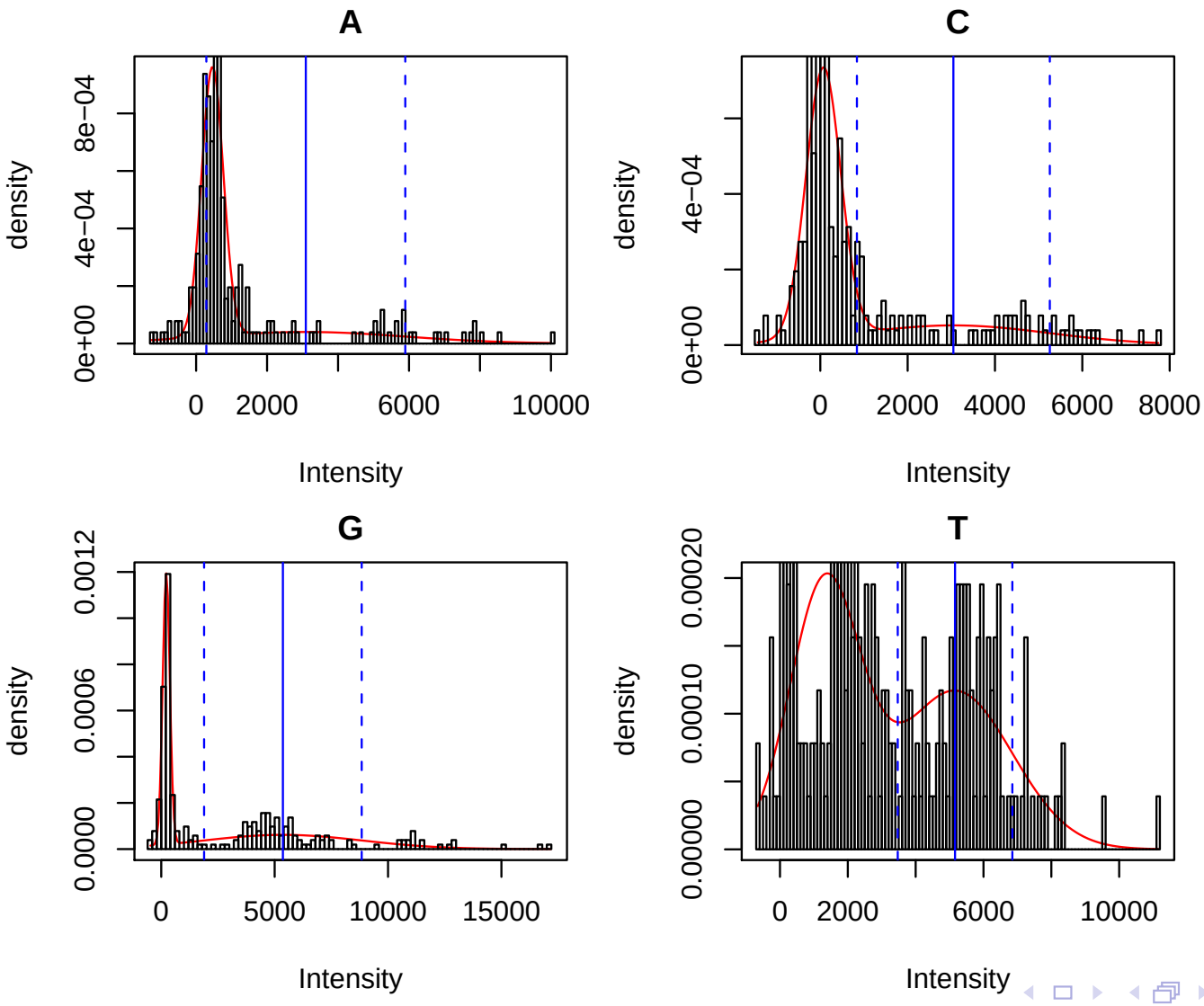


Dimensional Projections I

- We can also evaluate the distribution of intensity values at selected cycles via 1- and 2- dimensional projections:

```
> ChannelHistogram(int = int, cycles = 1,
+     par = list(mfrow = c(2, 2),
+         mar = c(4, 4, 2, 1) + 0.1))
```

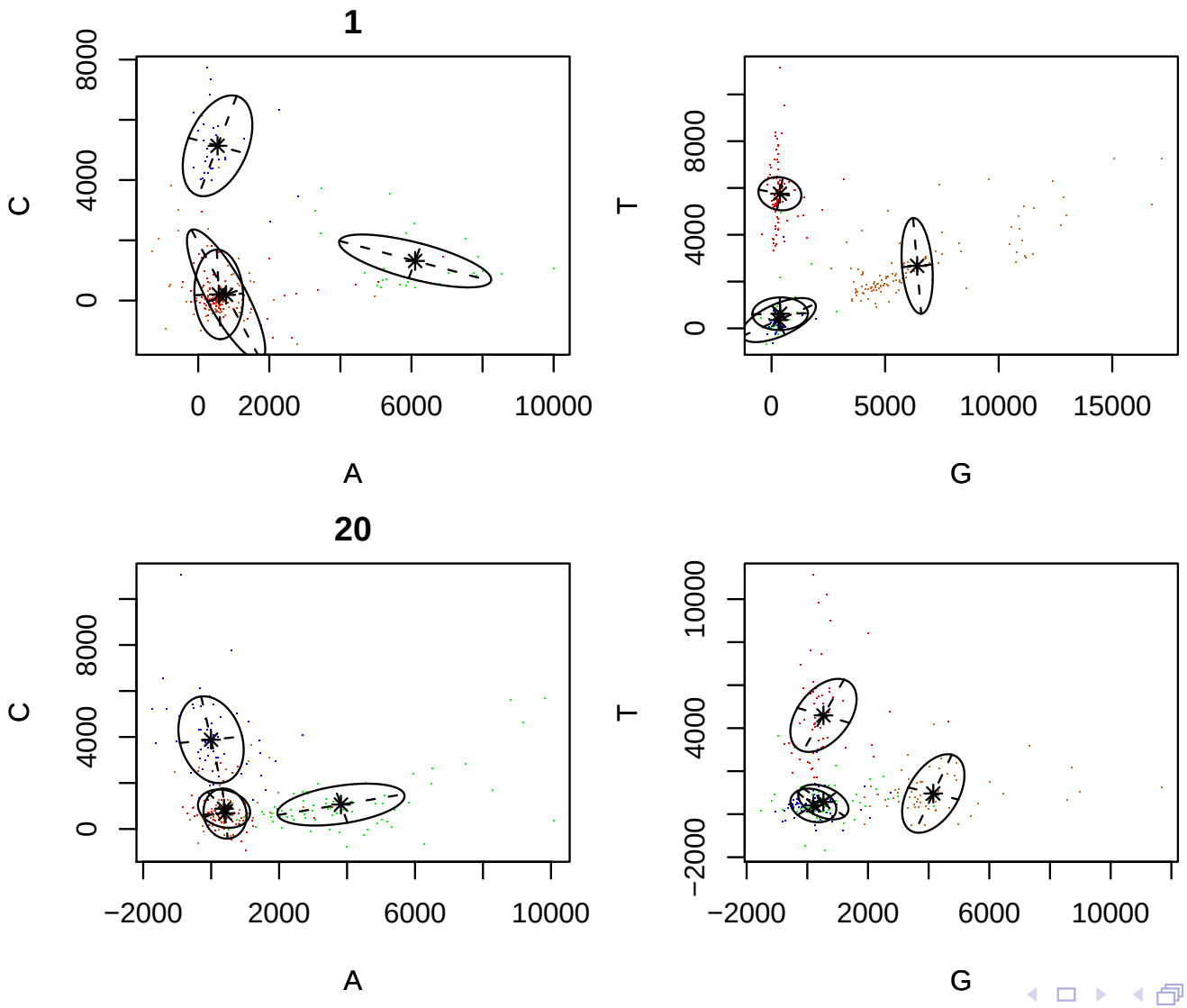
Dimensional Projections II



Dimensional Projections I

```
> par(mfrow = c(2, 2), mar = c(4,
+     4, 2, 1) + 0.1)
> PlotCycles(run = rolenv, int = int,
+     seq = seq, cycles = c(1, 20))
```

Dimensional Projections II



Global statistics plotting I

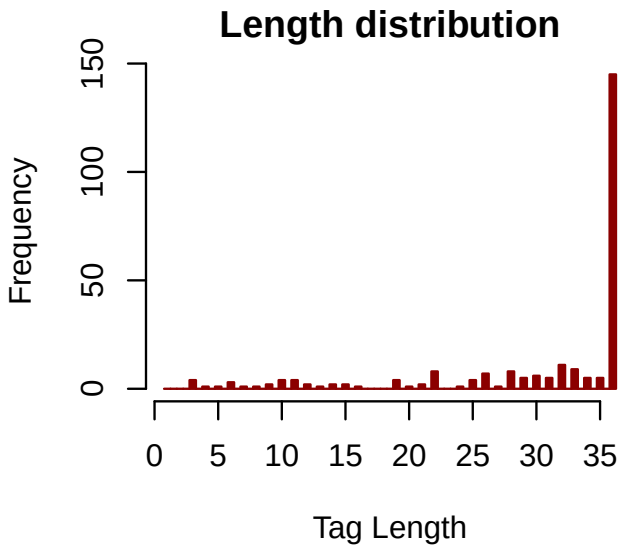
- look at global statistics of a base-calling:

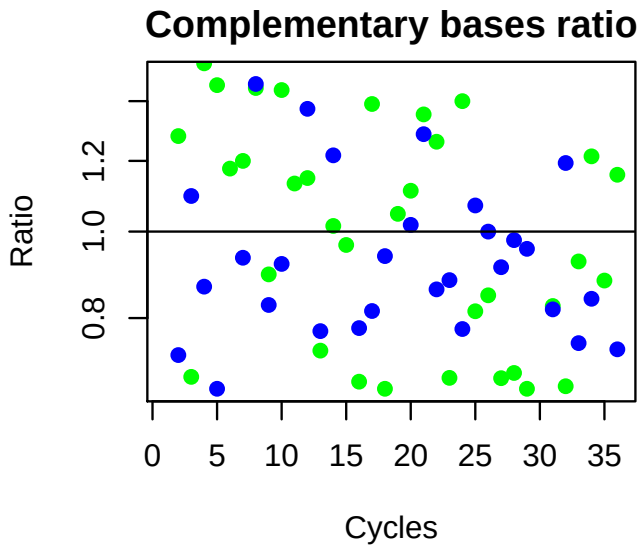
```
> par(mfrow = c(2, 2), cex = 0.8,
+     mar = c(4, 4, 2, 1) + 0.1)
> BatchAnalysis(run = rolenv, seq = res2$sread,
+     scores = res2$entropy, what = "length",
+     main = "Length distribution")
> BatchAnalysis(run = rolenv, seq = res$sread,
+     scores = res$entropy, what = "ratio",
+     main = "Complementary bases ratio")
> BatchAnalysis(run = rolenv, seq = res$sread,
+     scores = res$entropy, what = "iupac",
+     main = "IUPAC codes")
> BatchAnalysis(run = rolenv, seq = res2$sread,
```

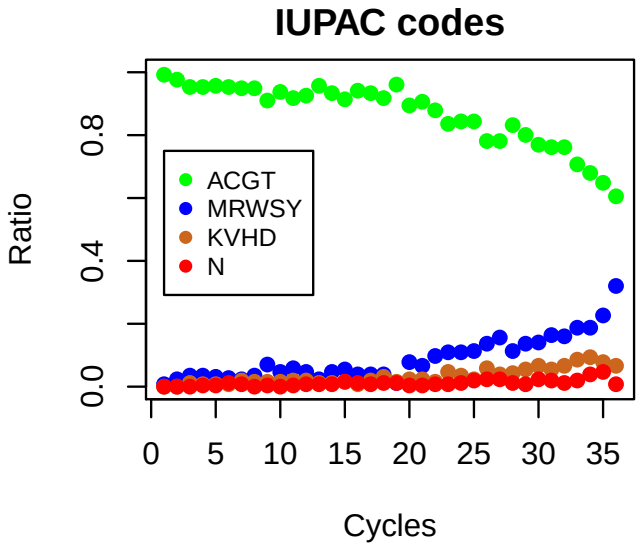
Global statistics plotting II

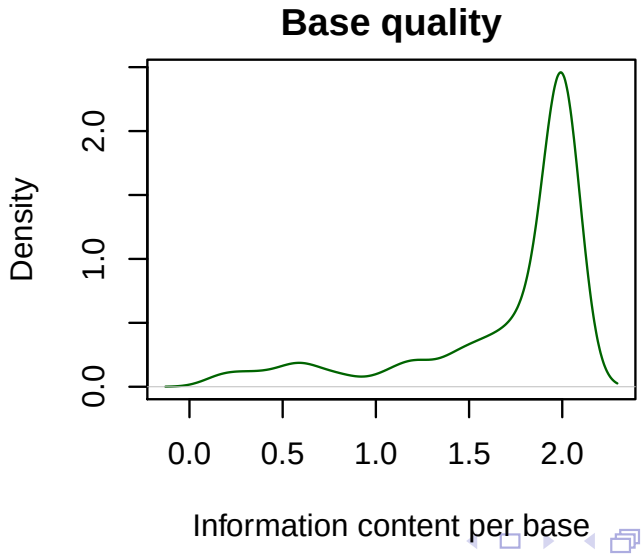
```
+      scores = res2$entropy, what = "information",
+      main = "Base quality")
```

Global statistics plotting III







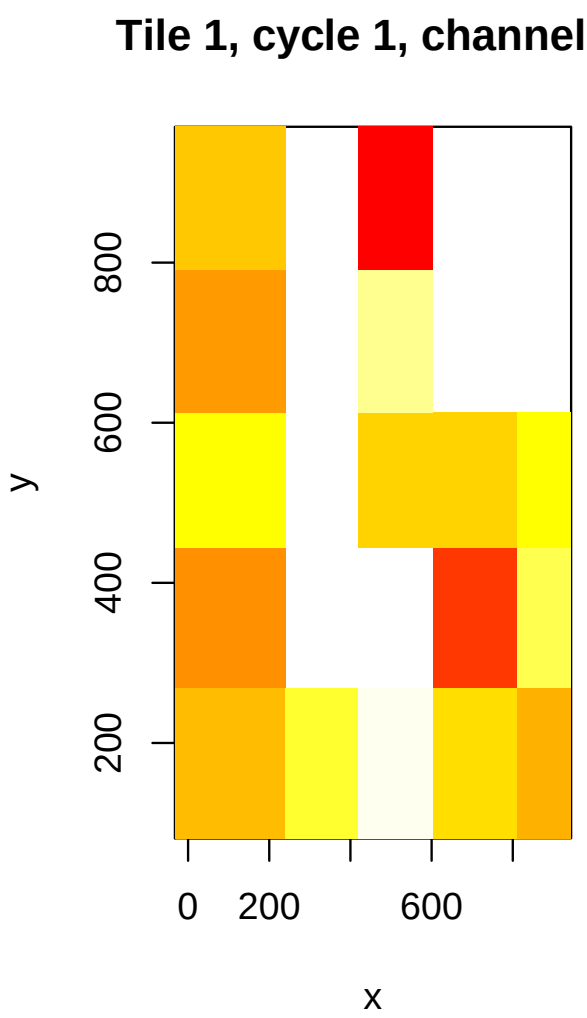


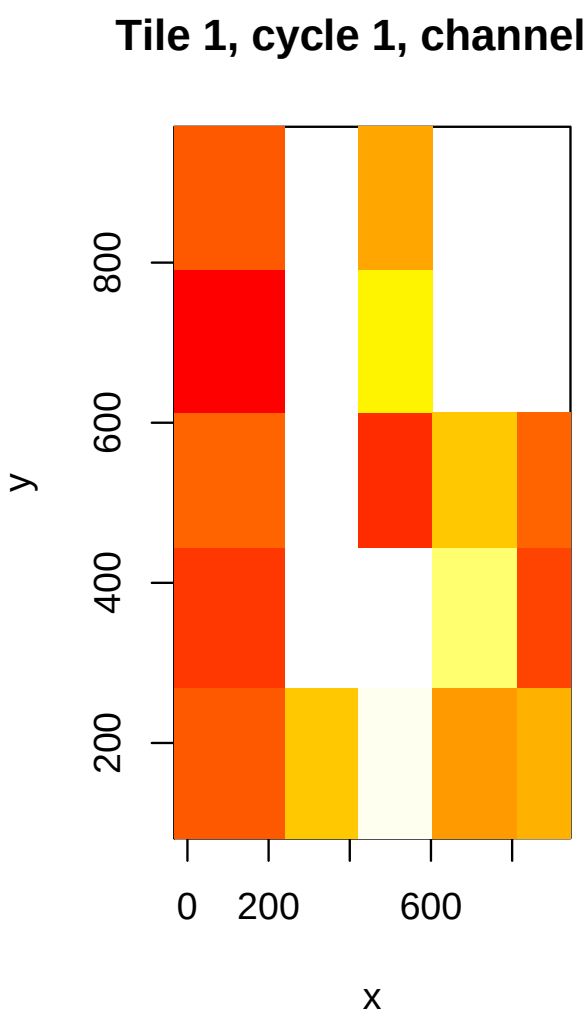
Global statistics plotting I

- And visualize the positional bias over a tile by:

```
> par(mfrow = c(1, 2))
> TileImage(int = int, cycle = 1,
+     tile = readInfo(int)$tile[1],
+     ncell = 5, channel = "A")
> TileImage(int = int, cycle = 1,
+     tile = readInfo(int)$tile[1],
+     ncell = 5, channel = "C")
```

Global statistics plotting II





Bye Bye I

Muchas gracias!!