# HDF5-supported tallying and the 'h5vc' package

EMBL


Developer Day 2013

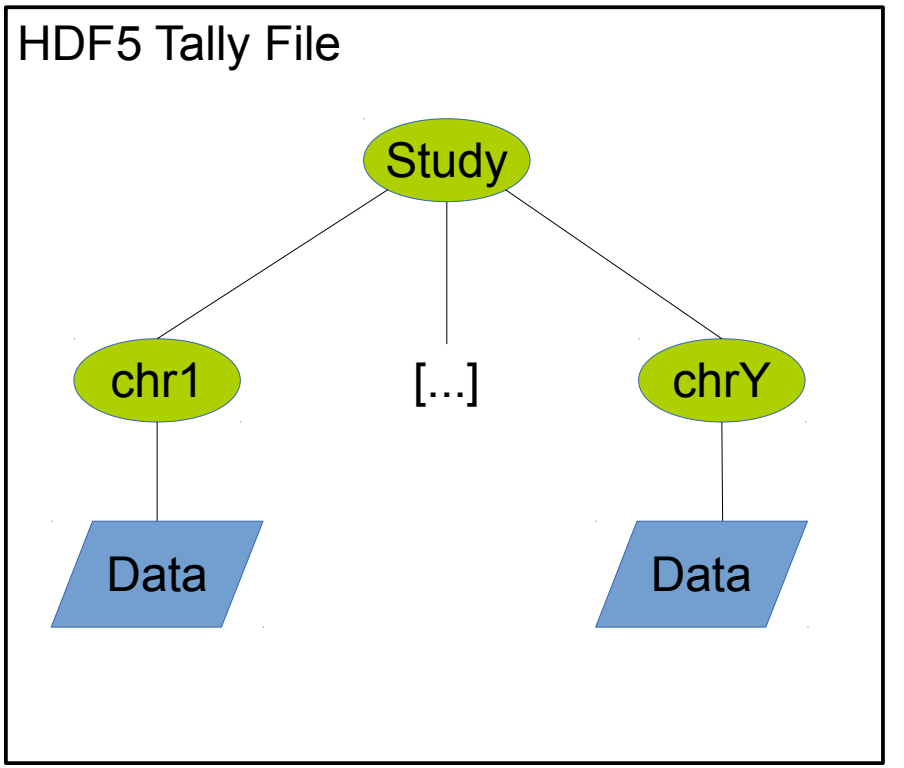
EMBL

# Outline

- HDF5
  - Data format
- Tallying
- Interacting with the Data
  - Calling Variants
  - Plots
  - Mutation Spectra

# HDF5 – very brief overview

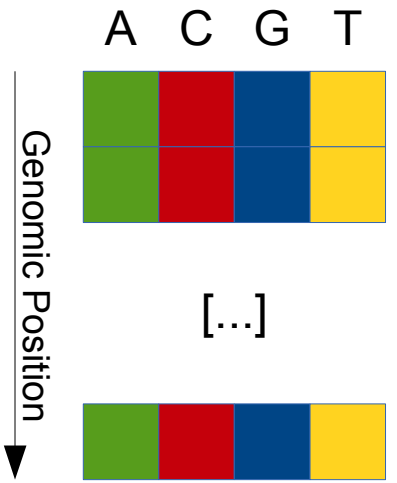
- 'file system'-like hierarchical structure of groups and datasets
- Tree structure
  - groups as nodes
  - datasets as leafs
  - ➔ efficent storage of large numerical datasets



EMBL

# Why HDF5

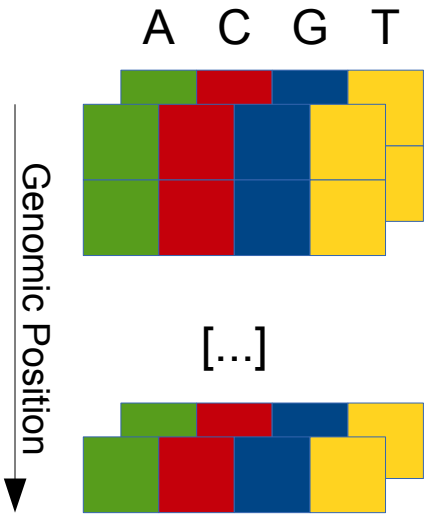
- Performance and portability (OS / Interface)
- Central object in SNV calling – the Tally:



EMBL

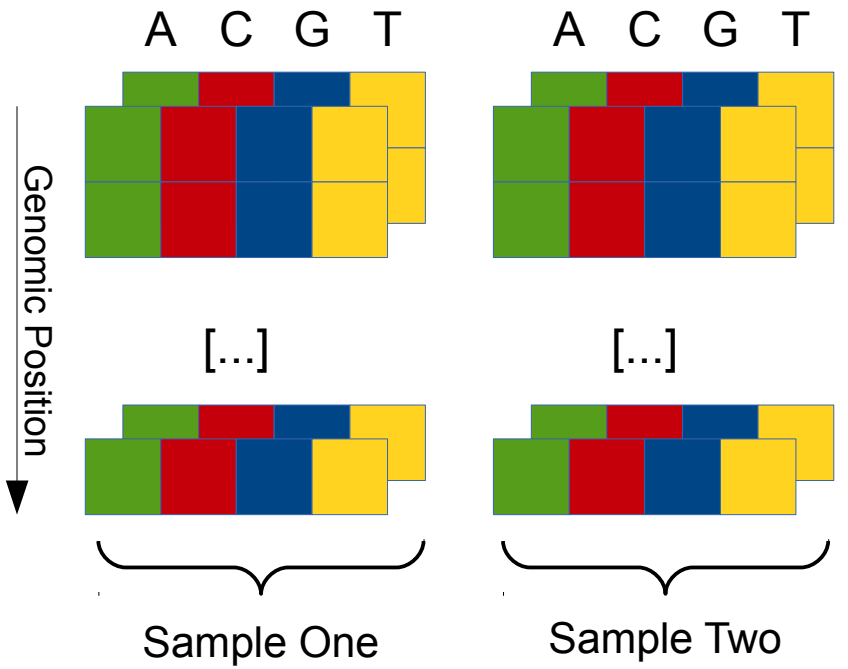
# Why HDF5

- Performance and portability (OS / Interface)
- Central object in SNV calling – the Tally:



# Why HDF5

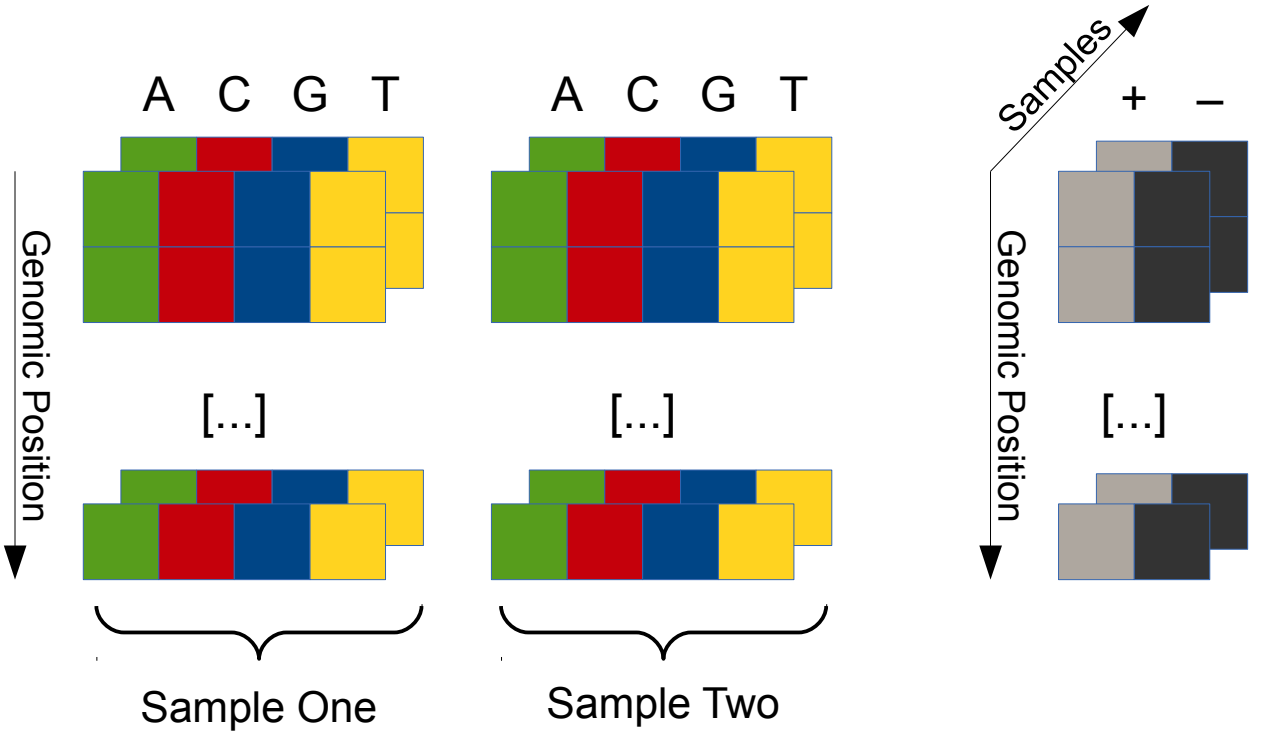
- Performance and portability (OS / Interface)
- Central object in SNV calling – the Tally:



EMBL

# Why HDF5

- Performance and portability (OS / Interface)
- Central object in SNV calling – the Tally:



EMBL

# Data format definition

```
> library(h5vc)
Loading required package: rhdf5
[...]
> h5ls("example.tally.hfs5")
              group          name       otype  dclass                  dim
0                 / ExampleStudy   H5I_GROUP
1     /ExampleStudy           16   H5I_GROUP
2  /ExampleStudy/16       Counts H5I_DATASET INTEGER 12 x 6 x 2 x 90354753
3  /ExampleStudy/16    Coverages H5I_DATASET INTEGER      6 x 2 x 90354753
4  /ExampleStudy/16    Deletions H5I_DATASET INTEGER      6 x 2 x 90354753
5  /ExampleStudy/16    Reference H5I_DATASET INTEGER              90354753
6     /ExampleStudy           22   H5I_GROUP
7  /ExampleStudy/22       Counts H5I_DATASET INTEGER 12 x 6 x 2 x 51304566
8  /ExampleStudy/22    Coverages H5I_DATASET INTEGER      6 x 2 x 51304566
9  /ExampleStudy/22    Deletions H5I_DATASET INTEGER      6 x 2 x 51304566
10 /ExampleStudy/22    Reference H5I_DATASET INTEGER              51304566

>
> getSampleData( filename = "example.tally.hfs5", name = "/ExampleStudy/16" )
            Sample Column  Patient    Type
1    PT8PrimaryDNA      6 Patient8    Case
2    PT5PrimaryDNA      2 Patient5    Case
3    PT5RelapseDNA      3 Patient5    Case
4 PT8EarlyStageDNA      5 Patient8    Case
5    PT5ControlDNA      1 Patient5 Control
6    PT8ControlDNA      4 Patient8 Control
```

# Tallying

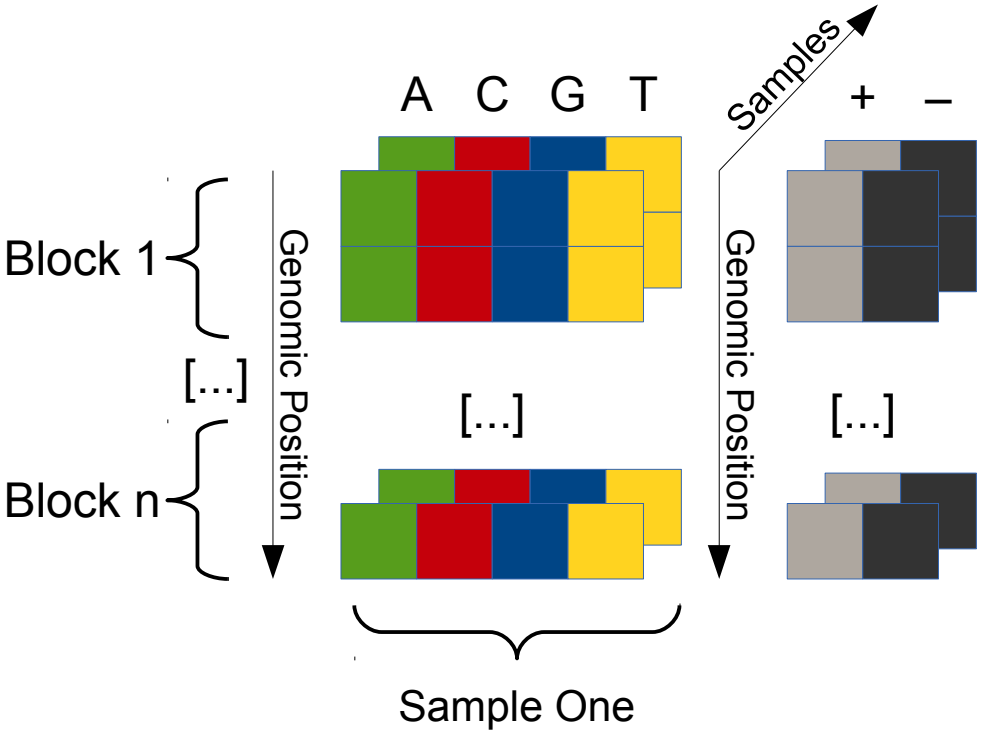
- Need for a common data format
- Explore different algorithms for tallying, calling, qc, ...
- Proof of concept implementation in Python
  - HTSeq, h5py, ...

```
python tally.bam.py --out example.tally.hfs5 \
    --reg 22:38000000-40000000 --reg 16:29000000-30000000 \
    --bam PT5ControlDNA:Patient5:Control:../Input/PT5ControlDNA.bam \
    --bam PT5PrimaryDNA:Patient5:Case:../Input/PT5PrimaryDNA.bam \
    --bam PT5RelapseDNA:Patient5:Case:../Input/PT5RelapseDNA.bam \
    --bam PT8ControlDNA:Patient8:Control:../Input/PT8ControlDNA.bam \
    --bam PT8EarlyStageDNA:Patient8:Case:../Input/PT8PreLeukemiaDNA.bam \
    --bam PT8PrimaryDNA:Patient8:Case:../Input/PT8PrimaryDNA.bam –replace --gzip
```

# Calling Variants – h5dapply

- h5dapply

```
variant_calls <- h5dapply(
     filename = "example.tally.hfs5",
     group = "/ExampleStudy/16",
     blocksize = 100000,
     names = c("Coverages", "Counts"),
     dims = c(3,4),
     range = c(29000000,30000000),
     FUN = callVariants,
     getSampleData(
          "example.tally.hfs5",
          "/ExampleStudy/16"
     )
)
```



# Calling Variants – a simple approach

```
vcConfParams <- function(
  minStrandCov = 5,
  maxStrandCov = 200,
  minStrandAltSupport = 2,
  maxStrandAltSupportControl = 0,
  bases = 5:8 ){
  return(
    list(
      minStrandCov = minStrandCov,
      maxStrandCov = maxStrandCov,
      minStrandAltSupport = minStrandAltSupport,
      maxStrandAltSupportControl =
maxStrandAltSupportControl,
      bases = bases
      )
    )
}

callVariants <- function(
  data,
  sampledata,
  cl = vcConfParams() )
```

EMBL

# Calling Variants – a simple approach

```
cov_filter <- caseCoverage[1,] >= cl$minStrandCov &
  caseCoverage[1,] <= cl$maxStrandCov &
  caseCoverage[2,] >= cl$minStrandCov &
  caseCoverage[2,] <= cl$maxStrandCov

count_filter <- caseCounts[,1,] >= cl$minStrandAltSupport &
  caseCounts[,2,] >= cl$minStrandAltSupport &
  controlCounts[,1,] <= cl$maxStrandAltSupportControl &
  controlCounts[,2,] <= cl$maxStrandAltSupportControl
```

EMBL

# Exploration – Plots

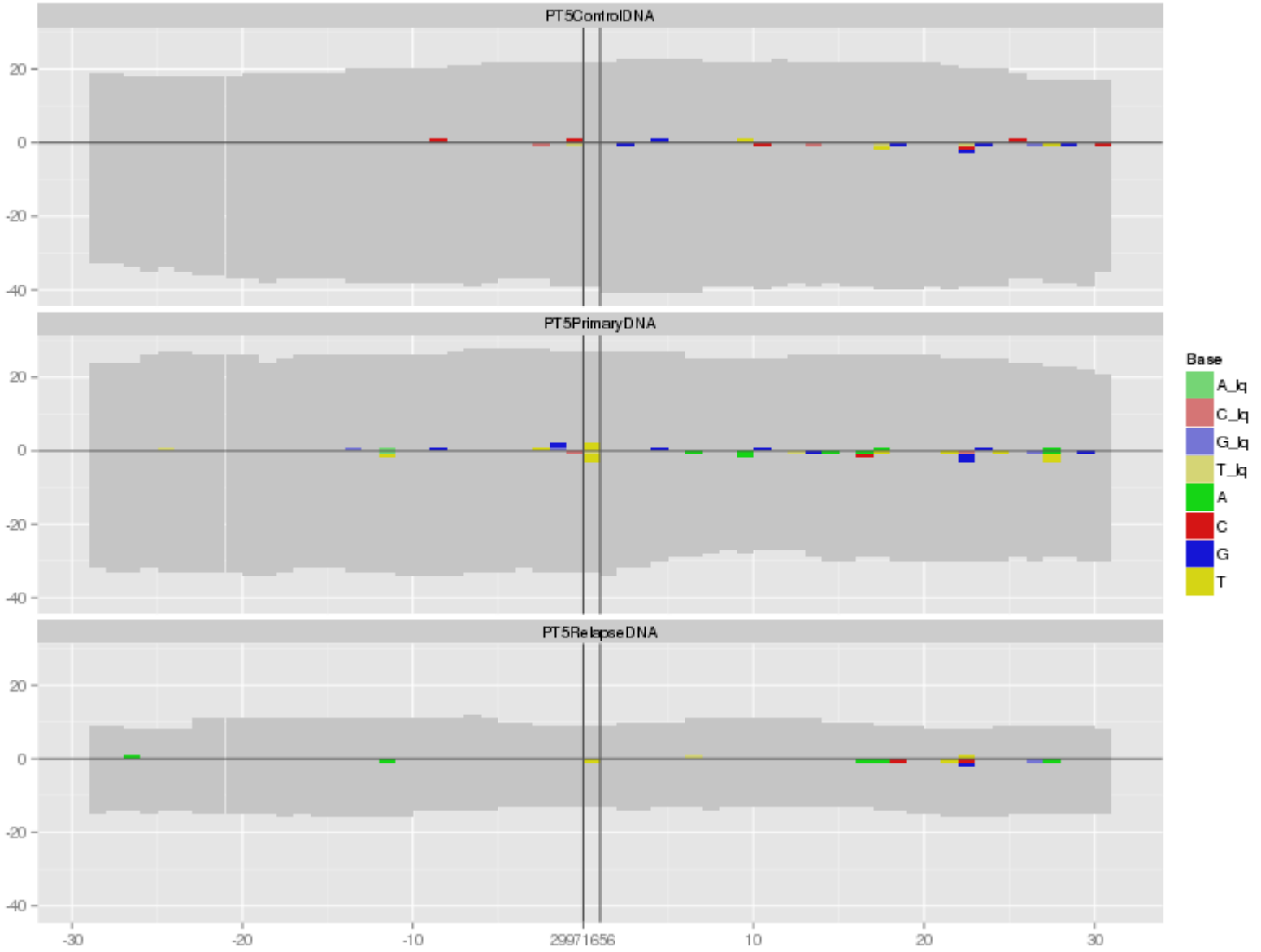
```
position = variant_positions$PT5PrimaryDNA[1]
windowsize = 30

data = h5dapply(
  filename = "example.tally.hfs5",
  name = "/ExampleStudy/16",
  blocksize = 50000,
  range = c(position - windowsize, position + windowsize)
  )

mismatchPlot(
  data = data[[1]],
  sampledata,
  samples = sampledata$Sample[ sampledata$Patient == "Patient5" ],
  windowsize,
  position
  )
```

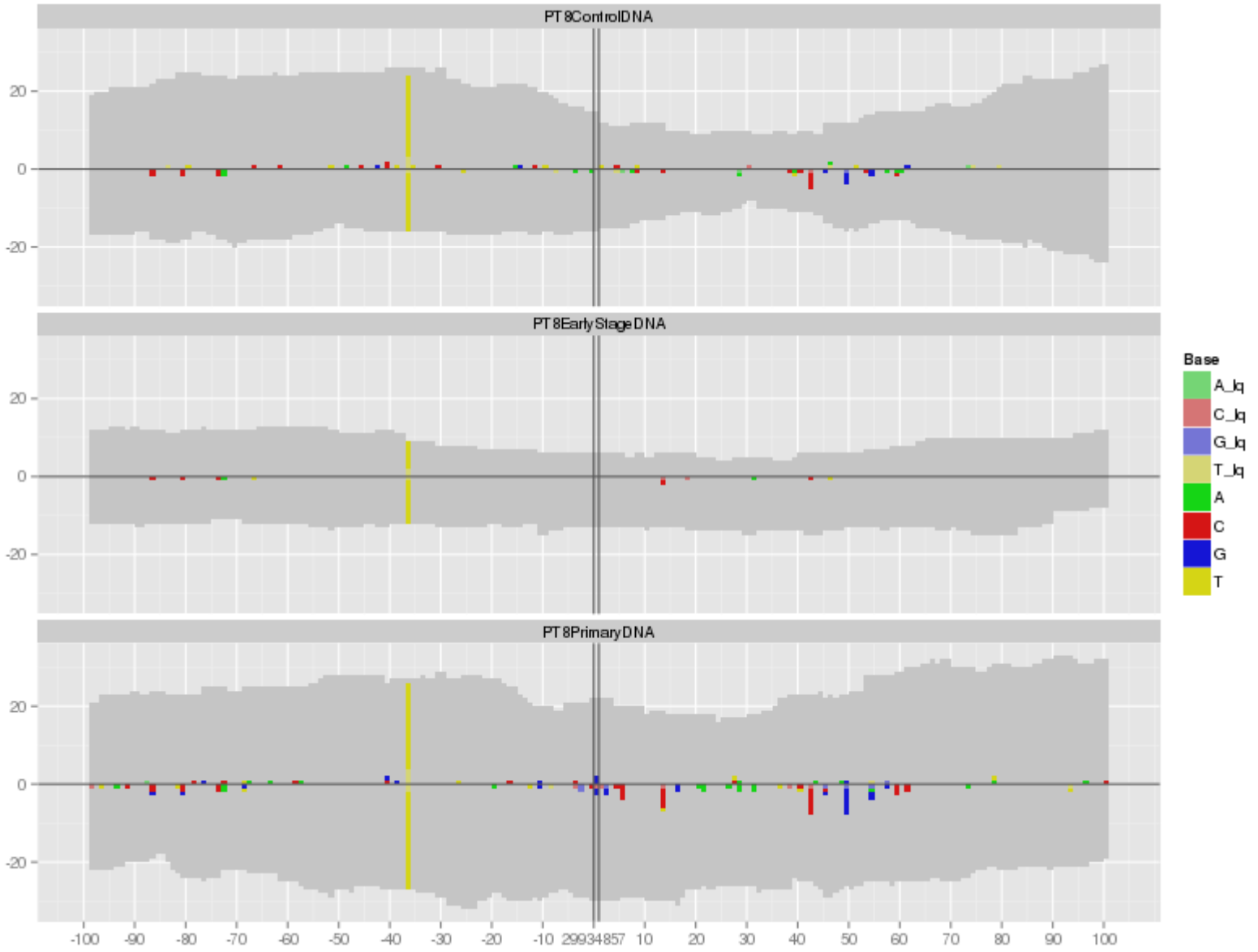
EMBL

# Exploration – Plots



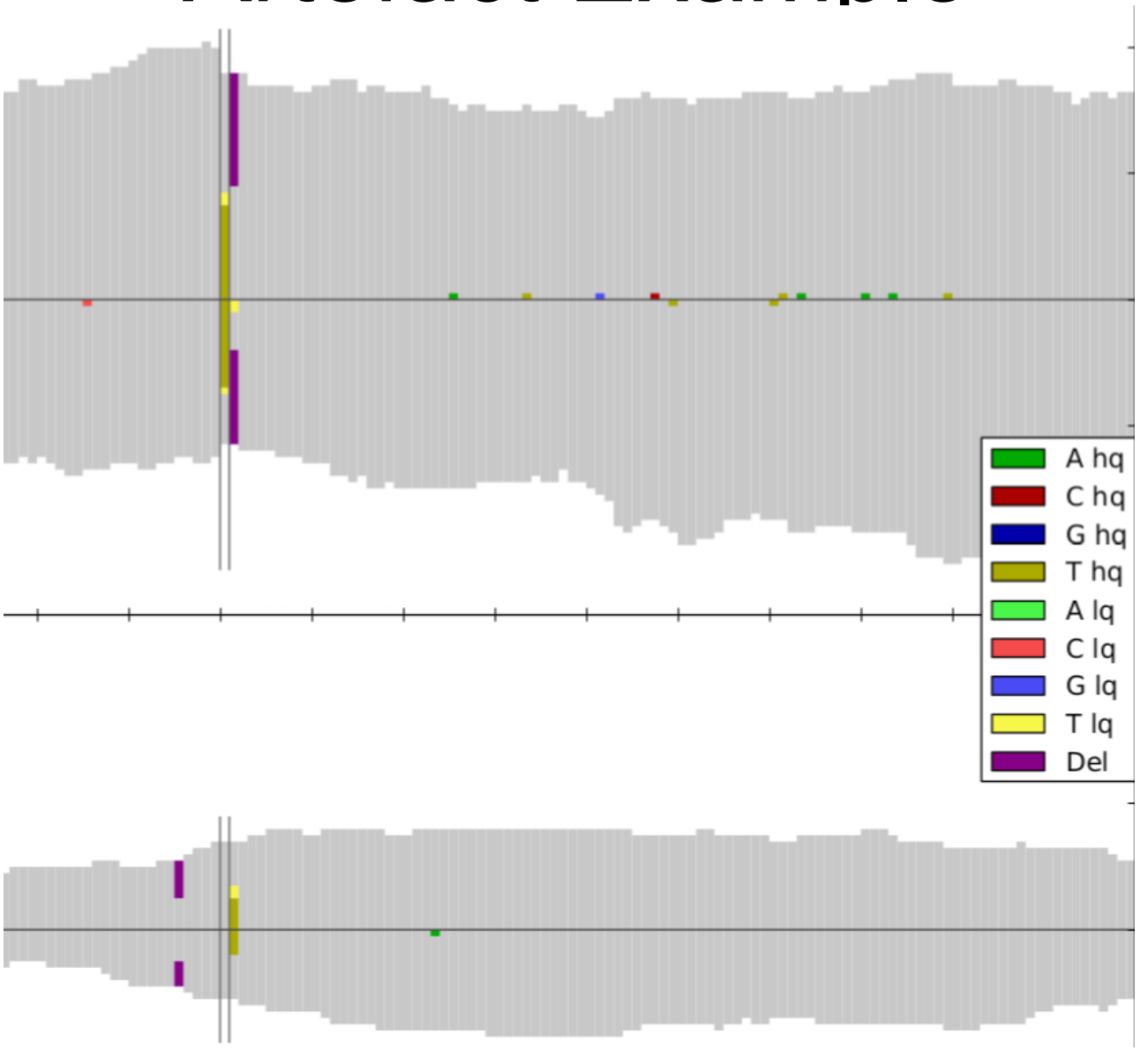
EMBL

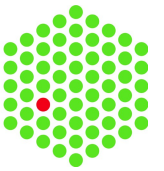
# Exploration – Plots



EMBL

# Artefact Example





# Exploration – Mutation Spectra

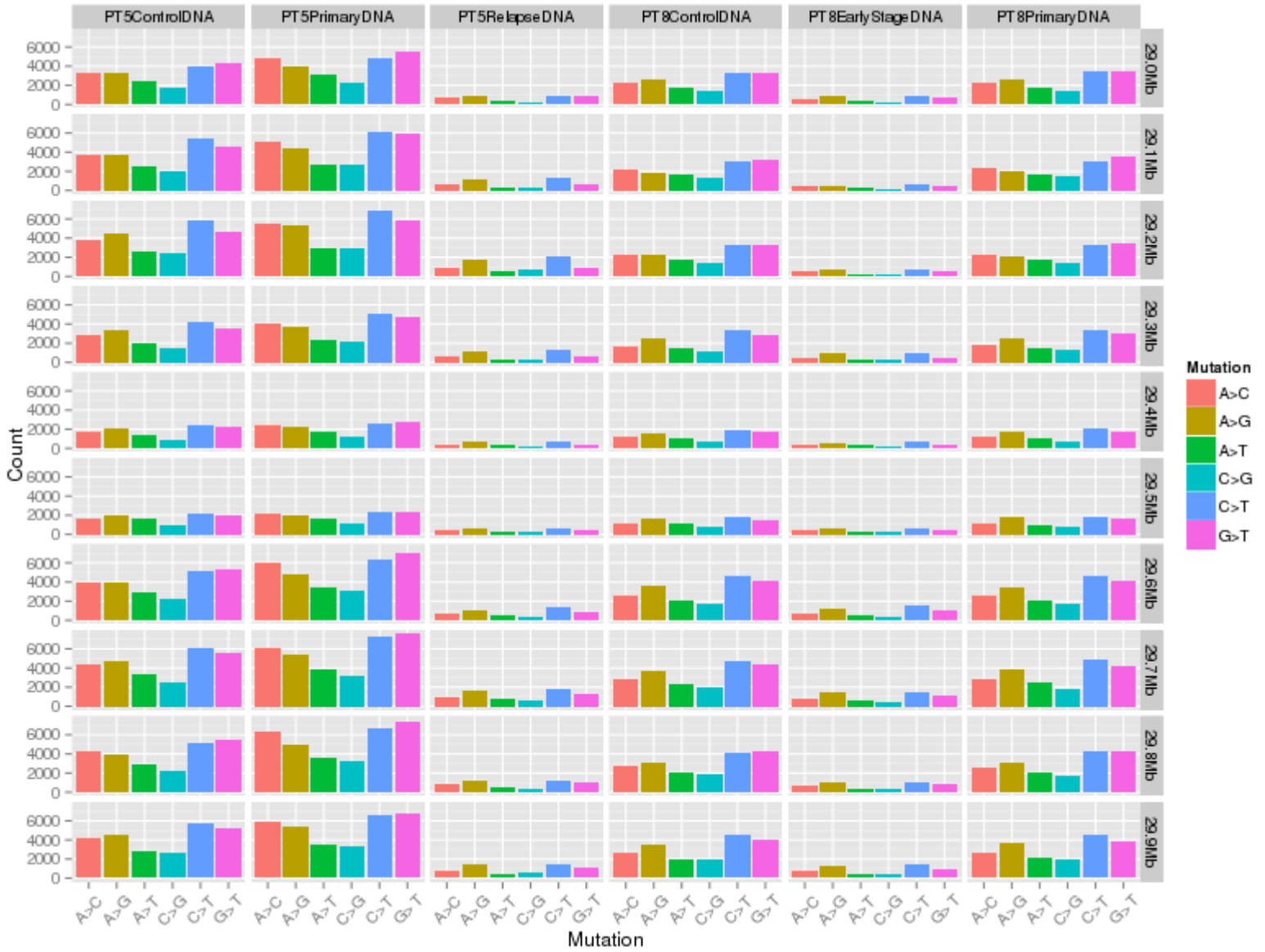
```
mutationSpectra <- h5dapply(
  filename = tallyFile,
  group = "/ExampleStudy/16",
  blocksize = 100000,
  range = c(29000000,30000000),
  names = c("Reference", "Counts"),
  dims = c(1,4),
  FUN = mutationSpectrum,
  sampledata
  )

plotData = melt( mutationSpectra )
colnames( plotData ) <- c( "Sample", "Mutation", "Count", "Bin" )
plotData$Bin = formatGenomicPosition( plotData$Bin )

ggplot(
  data = plotData,
  aes( x = Mutation, y = Count, fill = Mutation )
  ) +
  geom_bar( stat = "identity" ) + facet_grid( Bin ~ Sample ) + plotTheme
```

EMBL

# Exploration – Mutation Spectra



EMBL

# Exploration – Mutation Spectra

# Summary

- Many genomics analyses are centered around the tally
- HDF5 is very well suited for representing this data format
  - A common data format is needed
  - Creating the tally file is expensive
  - Downstream analysis is inexpensive
- http://www.ebi.ac.uk/~pyl/h5vc/

EMBL

# Thanks!

- Huber Group
  - Wolfgang
  - **Bernd**
  - Everyone!
- BioC 2013 Organisers