```
---
title: "Bioconductor для биологов"
author: "Гайк Тамазян"
date: "30 июля 2014"
output: html_document
---

Цель данного семинара --- продемонстрировать возможности пакета
**Bioconductor** для анализа геномных данных на примере задачи
анализа плотности однонуклеотидных полиморфизмов.

# Исходные данные

- FASTA-файл с последовательностью 38-й хромосомы генома собаки (сборка
CanFam3.1), длина хромосомы - 23 914 537 bp.
- GTF-файл с аннотацией генов, расположенных на заданной хромосоме.
- текстовый файл с координатами SNP собаки на заданной хромосоме.
- BED-файлы с регионами геномных повторов, найденных программами
**RepeatMasker** и **WindowMasker**.

Пакеты, необходимые для анализа:
- **GenomicRanges**,
- **GenomicFeatures**
- **VariantAnnotation**.

Упомянутые пакеты можно установить с помощью следующей команды.
```
source("http://bioconductor.org/biocLite.R")
biocLite()
biocLite(c("GenomicRanges", "GenomicFeatures", "VariantAnnotation"))
```

```{r settings, include=FALSE}
library(GenomicRanges)
library(GenomicFeatures)
library(VariantAnnotation)

fasta.file <- '~/Desktop/Bioconductor_workshop/dog_chr38.fa'

gtf.file <- '~/Desktop/Bioconductor_workshop/dog_genes_chr38.gtf'
variant.file <- '~/Desktop/Bioconductor_workshop/dog_snp_chr38.txt'
wm.file <- '~/Desktop/Bioconductor_workshop/dog_windowmasker_chr38.bed'
rm.file <- '~/Desktop/Bioconductor_workshop/dog_repeatmasker_chr38.bed'
```

Загружаем исходные данные.

```{r load_data}
chr.length <- 23914537
window.size <- 50000
# полиморфизмы
variant.df <- read.table(variant.file, as.is=TRUE, skip=1)
names(variant.df) <- c("ID", "CHROM", "POS", "ALLELES")
# повторы WindowMasker
wm.df <- read.table(wm.file, as.is=TRUE)
wm.df <- wm.df[, 1:3]
names(wm.df) <- c("CHROM", "START", "END")
wm.df$START <- wm.df$START + 1
# повторы RepeatMasker
rm.df <- read.table(rm.file, as.is=TRUE)
rm.df <- rm.df[, 1:3]
names(rm.df) <- c("CHROM", "START", "END")
rm.df$START <- rm.df$START + 1
# гены
exon.df <- read.table(gtf.file, as.is=TRUE)
names(exon.df) <- c("CHROM", "SOURCE", "TYPE", "START", "END", "SCORE",
                    "STRAND", "FRAME", "ATTRIBUTES")
```

Всего загружено:

- `r nrow(variant.df)` полиморфизмов,
- `r nrow(wm.df)` повторов, найденных **WindowMasker**,
- `r nrow(rm.df)` повторов, найденных **RepeatMasker**,
- `r nrow(exon.df)` экзонов.

Работа с однонуклеотидными полиморфизмами

Получение однонуклеотидных полиморфизмов
```

Загруженный файл содержит как однонуклеотидные полиморфизмы (SNP), так
и инсерции-делеции. Напишем функцию, которая оставит только SNP.

```{r get_snp_function}
get.snps <- function(x) {
  alleles <- strsplit(x$ALLELES, "/", fixed=TRUE)
  is.one.base <- sapply(alleles, function(x) max(sapply(x, length))) == 1
  is.indel <- sapply(alleles, function(x) any(x == "-"))
  x <- x[is.one.base & !is.indel, ]
  return(x)
}
```

Применим ее к нашему набору полиморфизмов.
```{r get_snp}
snp.df <- get.snps(variant.df)
```
Получено `r nrow(snp.df)` SNP.

В дальнейшем нам понадобится `collapsedVCF`-объект, представляющий
полученные однонуклеотидные полиморфизмы. Создадим его.

```{r snp_granges}
snp.ranges <- GRanges(seqnames=paste("chr", snp.df$CHROM, sep=""),
                      ranges=IRanges(start=snp.df$POS, end=snp.df$POS),
                      id=snp.df$ID)
snp.vcf <- VCF(rowData=snp.ranges)
alt(snp.vcf) <- DNAStringSetList(lapply(strsplit(snp.df$ALLELES, "/", fixed=TRUE), "[", 1))
ref(snp.vcf) <- DNAStringSet(sapply(strsplit(snp.df$ALLELES, "/", fixed=TRUE), "[", 2))
```

Фильтрация по геномным повторам

Полиморфизмы, которые попадают в области геномных повторов, в
действительности полиморфизмами не являются. Мы исключим из дальнейшего
анализа те SNP, которые попадают в геномные повторы, идентифицированные
программами **RepeatMasker** или **WindowMasker**.

Сначала мы создадим `GRanges`-объекты для повторов, полученных обеими
программами.

```{r repeat_granges}
rm.ranges <- GRanges(seqnames=rm.df$CHROM, ranges=IRanges(
  start=rm.df$START, end=rm.df$END))
wm.ranges <- GRanges(seqnames=wm.df$CHROM, ranges=IRanges(
  start=wm.df$START, end=wm.df$END))
```

Затем мы объединим интервалы с помощью функции `union`.

```{r repeat_union}
repeat.ranges <- union(rm.ranges, wm.ranges)
```

Наконец мы пересекаем множество интервалов геномных повторов с
множеством полиморфизмов.

```{r filter_snp_by_repeats}
snp.in.repeats <- as.logical(countOverlaps(rowData(snp.vcf),
                                           repeat.ranges))
snp.vcf <- snp.vcf[!snp.in.repeats]
```

Из `r length(snp.in.repeats)` полиморфизмов `r sum(snp.in.repeats)`
было отфильтровано по вхождениям в геномные повторы. Всего
`r sum(!snp.in.repeats)` полиморфизмов будет рассмотрено далее.

Разбиение хромосомы на окна

Мы явно построим `GRanges`-объект, который будет представлять разбиение
хромосомы на окна фиксированной длины `r window.size` bp.

```{r constructing_chromosome_windows}
chr.grid <- seq(window.size, chr.length, by=window.size)
chr.windows <- GRanges(seqnames="chr38", ranges=IRanges(
  start=c(1, chr.grid + 1), end=c(chr.grid, chr.length)))
```

Всего построено `r length(chr.windows)` окон.

Анализ плотности SNP

Сначала для каждого окна мы подсчитаем количество SNP, которые в него
попали.

```{r count_snp_in_windows}
window.snp.counts <- countOverlaps(chr.windows, rowData(snp.vcf))
```

Теперь мы получим количество окон, в которые попало фиксированное число
SNP.

```{r analyse_window_counts}
freq.table <- table(window.snp.counts)
freq.df <- data.frame(SNP.NUMBER=as.numeric(names(freq.table)),
                      WINDOW.COUNT=unname(as.vector(freq.table)))
```

Заметим, что, например, ни одно окно не содержит ровно 2 SNP. Дополним
полученный data frame, явно указав нулевые значения для отсутствующих
частот SNP в окнах.

```{r fill_frequency_data_frame}
max.snp.number <- max(freq.df$SNP.NUMBER)
full.snp.count <- numeric(max.snp.number + 1)
full.snp.count[freq.df$SNP.NUMBER + 1] <- freq.df$WINDOW.COUNT
freq.df <- data.frame(SNP.NUMBER=0:max.snp.number,
                      WINDOW.COUNT=full.snp.count)
```

Построим график количества окон с тем или иным количеством SNP.

```{r plot_snp_frequency_plot, fig.width=15}
plot(freq.df$SNP.NUMBER, freq.df$WINDOW.COUNT,
     xlab="# SNPs in window", ylab="# windows", type="o")
grid()
```

Выделение синонимичных SNP

Мы воспользуемся средствами пакета **VariantAnnotation** для определения
синонимичных и несинонимичных SNP. Сначала нужно загрузить информацию
о генах.

```{r load_gene_data}
chrom.info <- data.frame(chrom="chr38", length=chr.length,
                         is_circular=FALSE)
gene.db <- makeTranscriptDbFromGFF(gtf.file, format="gtf",
                                   chrominfo=chrom.info)
```

Теперь мы можем воспользоваться функцией `predictCoding` из пакета
`VariantAnnotation`. Также мы выведем данные по несинонимичным SNP.

```{r get_nonsynonymous_snp}
snp.effect <- predictCoding(snp.vcf, gene.db,
                            seqSource=FaFile(fasta.file))
print(snp.effect[snp.effect$CONSEQUENCE == 'nonsynonymous'])
```