geneXtendeR

Bohdan B. Khomtchouk

July 19, 2017

Introduction

This vignette describes `geneXtendeR`, an R/Bioconductor package for the annotation of genomic features (primarily peaks called from a ChIP-seq experiment, but any coverage island regions would work) with the nearest gene. "Extending" refers to performing gene-feature overlaps after adding to the gene-span a user-specified region upstream of the start of the gene model and a fixed (500 bp) region downstream of the gene, resulting in assigning to a gene the features that do not physically overlap with it but are sufficiently close. Extending is an automated iterative procedure in `geneXtendeR`, allowing the user to repeatedly align peaks to multiple GTF files to assess what global gene-spans optimize the genomewide alignment of peaks with their closest genes. This facilitates the process of deciphering which differentially enriched peaks are dysregulating which specific genes. This, in turn, aids experimental follow-up and validation, such as the design of primers for a set of genes during qPCR.

Rationale

With an abundance of Bioconductor software available for peak annotation to nearby features (e.g., ChIPpeakAnno) and even command line tools (e.g., BEDTools *closest* function), what makes `geneXtendeR` different? The simple answer is: `geneXtendeR` is designed for cis-regulatory elements and proximal-promoter region analysis.

`geneXtendeR` is designed to optimally annotate a histone modification ChIP-seq peaks file with functionally important genomic features (e.g., genes associated with peaks) based on optimization calculations. These optimization calculations automatically factor in experimental conditions such as the broadness of the histone peaks found in the specific tissue of the ChIP-seq peak file.

geneXtendeR is designed to optimally annotate a histone modification ChIP-seq peaks file with functionally important genomic features (e.g., genes associated with peaks) based on optimization calculations. These optimization calculations automatically factor in experimental conditions such as the broadness of the histone peaks found in the specific tissue of the ChIP-seq peak file.

To accomplish this level of custom-tailored data-centric analysis, geneXtendeR first optimally extends the boundaries of every gene in a genome by some genomic distance (in DNA base pairs) for the purpose of flexibly incorporating cis-regulatory elements, such as enhancers and promoters, as well as downstream elements that are important to the function of the gene relative to an epigenetic histone modification ChIP-seq dataset. This action effectively transforms genes into "gene-spheres", a new term that we coin to emphasize the 3D-nature of heterochromatin. A gene-sphere is composed of cis-regulatory elements (e.g., proximal promoters $+/- \approx 3$ kb from TSS), distal regulatory elements (e.g., enhancers), transcription start/end sites (TSS/TES), exons, introns, and downstream elements of a gene. As such, geneXtendeR maximizes the signal-to-noise ratio of locating gene regions closest to and directly under peaks. By performing a computational expansion of this nature, ChIP-seq reads that would initially not map strictly to a specific gene can now be optimally mapped to the regulatory regions of the gene, thereby implicating the gene as a potential candidate, and thereby making the ChIP-seq experiment more successful. Such an approach becomes particularly important when working with epigenetic histone modifications that have inherently broad peaks with a diffuse range of signal enrichment (e.g., H3K9me1, H3K27me3).

A series of diagnostic criteria are used to compute optimal gene extensions tailored to the tissue-specific broadness of the specific epigenetic mark in the ChIP-seq peak input file:

First, install the geneXtendeR R package via:

```
> ## try http:// if https:// URLs are not supported
> source("https://bioconductor.org/biocLite.R")
> biocLite("geneXtendeR")
> library(geneXtendeR)
```

This automatically loads the rtracklayer R package, which contains the readGFF() command used to retrieve gene transfer format files of any model organism. As such, load in a GTF file into your R environment, e.g.:

```
> rat <- readGFF("ftp://ftp.ensembl.org/pub/release-84/gtf/
+                        rattus_norvegicus/Rattus_norvegicus.Rnor_6.0.84.chr.gtf.gz")
```

URLs may be obtained as direct links from: http://useast.ensembl.org/info/data/ftp/index.html. Click on the "GTF" link under the "Gene sets" column for a particular species and then right-click (or command-click on Mac OS X) the name of the file containing the species name/version number and file extension chr.gtf.gz (e.g., Homo_sapiens.GRCh38.84.chr.gtf.gz, Mus_musculus.GRCm38.84.chr.gtf.gz, etc), and copy the link address. Then, paste it into the readGFF() as shown above. This will create an R data frame object containing the respective gene transfer format file.

Next, the user must input their peak data from some peak caller (e.g., SICER, MACS2, etc). The peak data must contain only three tab-delimited columns: chromosome number, peak start, and peak end. See ?samplepeaksinput for an example. Once the peak input data (e.g., "somepeaksfile.txt") has been assembled properly (i.e., to contain only the three tab-delimited columns above), it must be

properly formatted prior to the execution of various geneXtendeR analyses. To do this, the user must first set their working directory to point to the location of their peak data file. Then type the following command:

```
> peaksInput("somepeaksfile.txt")
```

This command properly formats the user's peak file in preparation for subsequent analyses, producing a resultant "peaks.txt" file.

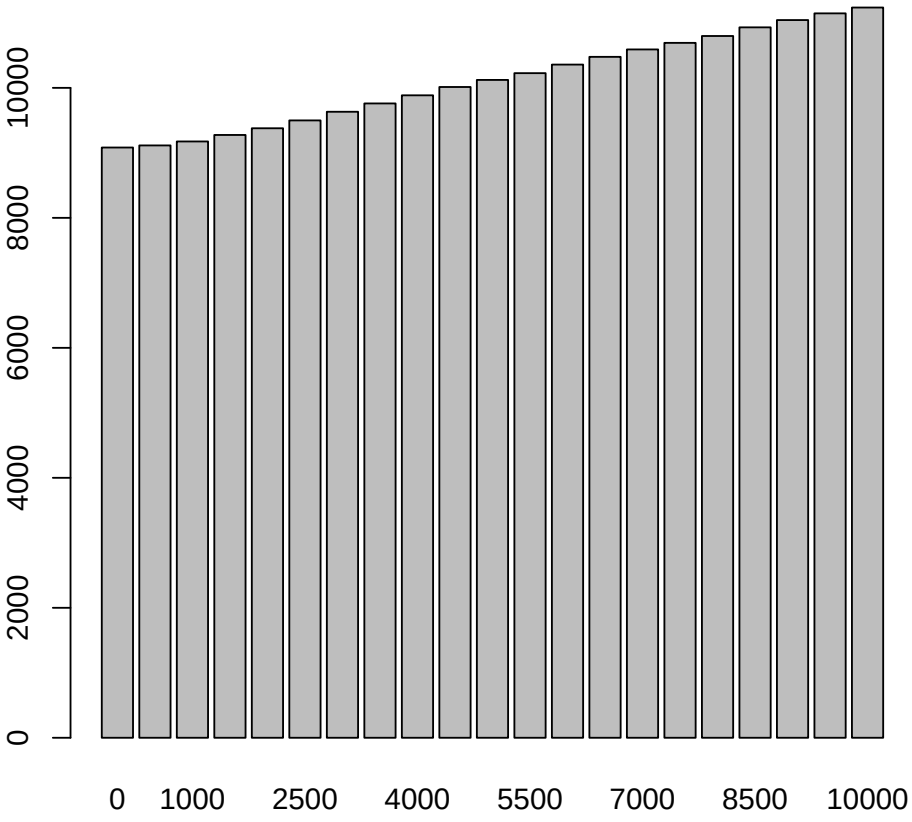
To see how this works using a built-in example, the `geneXtendeR` R package provides a peak input dataset called "somepeaksfile.txt", which can be loaded like this:

```
> fpath <- system.file("extdata", "somepeaksfile.txt", package="geneXtendeR")
> peaksInput(fpath)
```

This creates a properly formatted (i.e., properly sorted) "peaks.txt" file in the user's working directory.

Now, we may use the R object that we created with `readGFF()` above to create a bar chart visualization showing the number of peaks that are sitting directly on top of genes across a series of upstream extensions (of each gene in a genome):

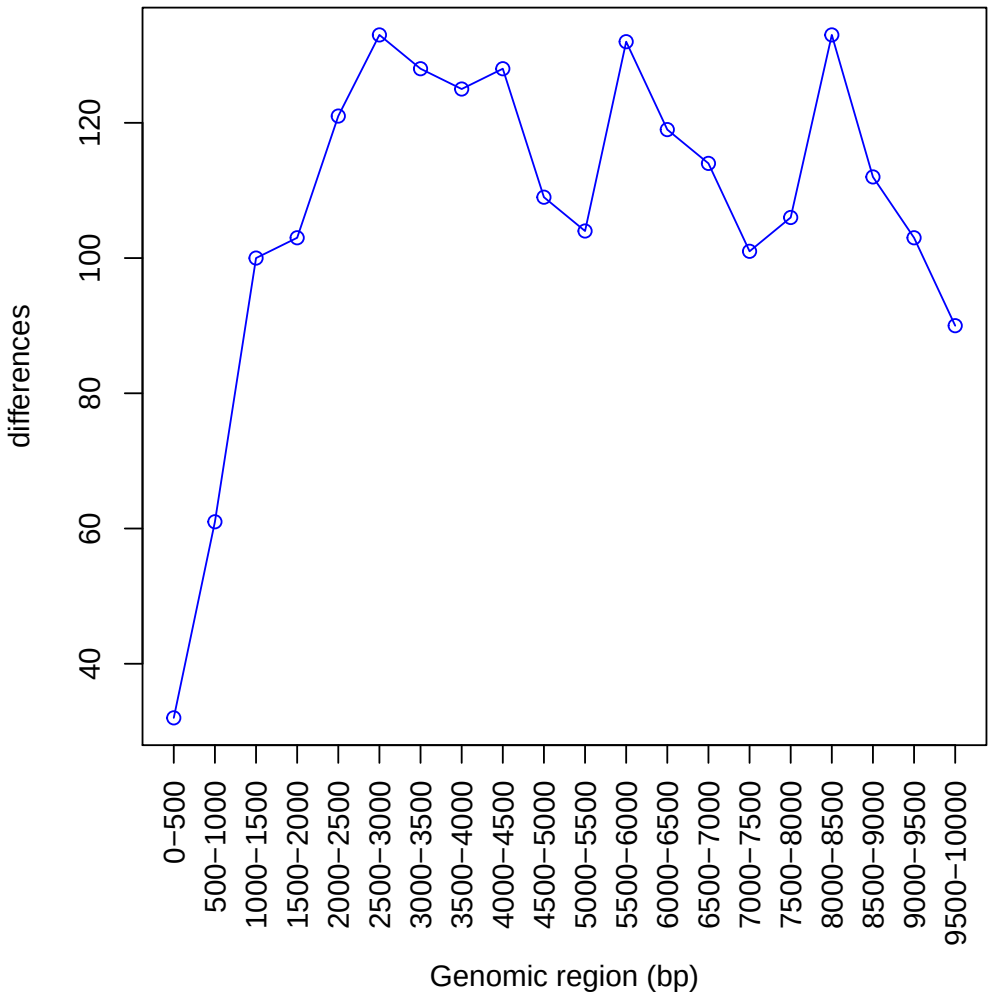
```
> barChart(rat, 0, 10000, 500)
```



This command first generates 21 individual whole-genome files: 0, 500, 1000, ..., and 10000 bp upstream extension files for the rat (*Rattus norvegicus*) genome, each having an automatic 500 bp downstream extension. In other words, each gene in the rat genome is extended upstream and downstream by some user-specified distance, thereby creating a "gene-sphere." As such, this command visualizes the raw count of the number of peaks that are sitting on top of genes at each individual upstream cutoff.

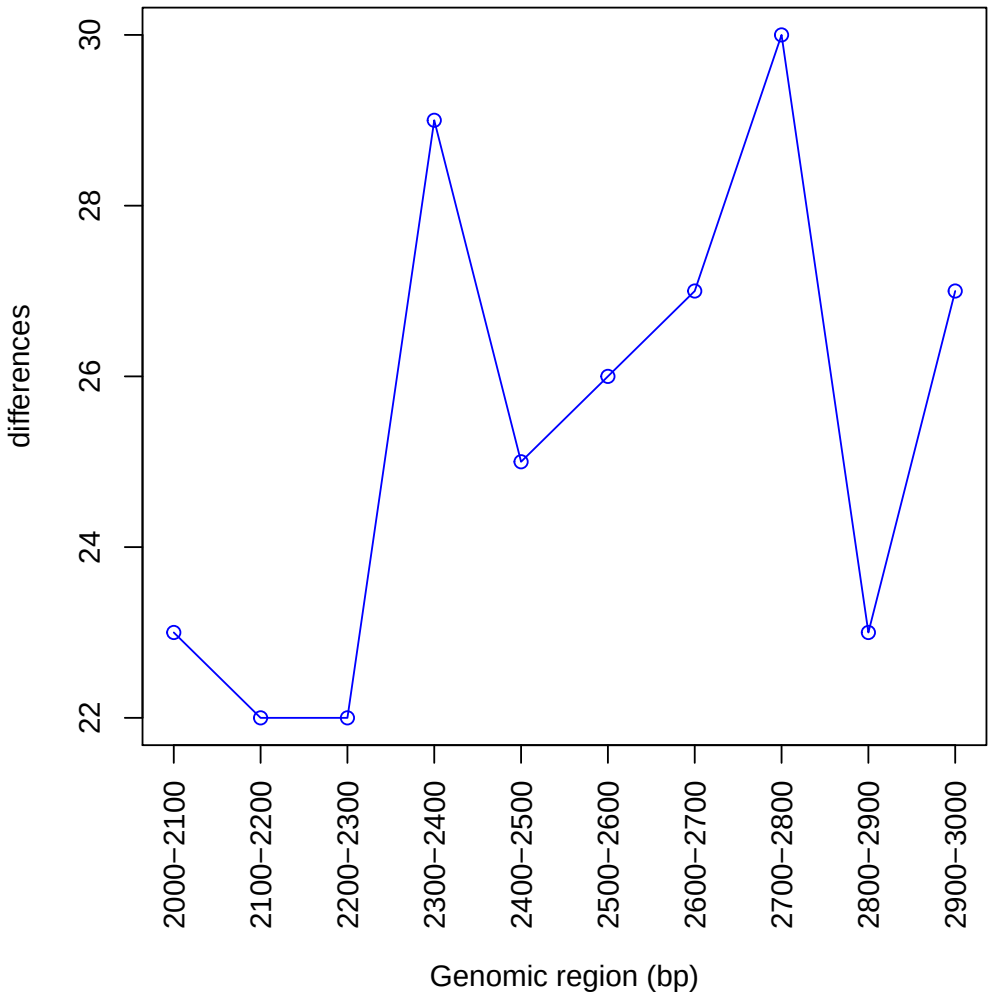
Clearly, the wider the gene-sphere, the more peaks-on-top-of-genes are found throughout the genome. However, the law of diminishing returns begins to kick in at increasing upstream extension levels (see `linePlot()` for a visual representation):

```
> linePlot(rat, 0, 10000, 500)
```



Clearly, there is a sharp rise in the number of peaks-on-top-of-genes from a 0 bp upstream extension to a 1500 bp upstream extension, and from a 2000 bp upstream extension to a 3000 bp upstream extension. This steady rise up until 3000 bp is followed by a steady decline at subsequent extension levels followed by some noisy fluctuations. It may be interesting to investigate what is going on in the interval from 2500 bp to 3000 bp:

```
> linePlot(rat, 2000, 3000, 100)
```



Clearly, there is a relatively sharp spike in the number of peaks-on-top-of-genes at the 2400 bp upstream extension (as compared to the 2300 bp extension). This spike then drops back down at subsequent extension levels and fluctuates in a noisy manner.

It is also possible to identify the genes that are unique amongst the 2300 and 2400 bp upstream extension levels:

```
> distinct(rat, 2300, 2400)
```

shows the first six entries sorted by chromosome and start position. `V1−V3` denote the chromosome/start/end positions of the peaks, `V4−V6` denote the respective values for the genes, `V7` is the gene ID (e.g., Ensembl ID), `V8` is the gene name, and `V9` is the distance of each respective peak to its nearest gene. It should be noted that the X chromosome is designated by the integer 100, the Y chromosome by the integer 200, and the mitochondrial chromosome by the integer 300. This is done for sorting purposes (see `?peaksInput` for details). In short, the `distinct()` command finds what peaks-on-top-of-genes would be missed if a 3250 bp upstream extension is used instead of a 3300 bp extension.

Of course, subsequent follow-up extensions naturally incorporate additional peaks-on-top-of-genes, since the concept of a gene is being expanded into an ever-widening gene-sphere.

However, even though these dynamics are to be expected, such extensions are unlikely to add significant value to the annotation of the peak file. Taking the example of the 0-10000 bp line plot, an upstream extension beyond 3500 bp globally across every gene in a genome would most likely not accurately reflect the biology of the peak input file (since such large global upstream extensions are likely to reach considerably beyond known proximal promoter elements, especially for relatively narrow histone marks). Such assumptions may be validated directly by the user by investigating the p-value and FDR of specific peaks using a combination of HT-seq (to count the reads) and edgeR/DESeq (to assess statistical significance). As such, geneXtendeR is designed to be used as part of a biological workflow involving subsequent statistical analysis:

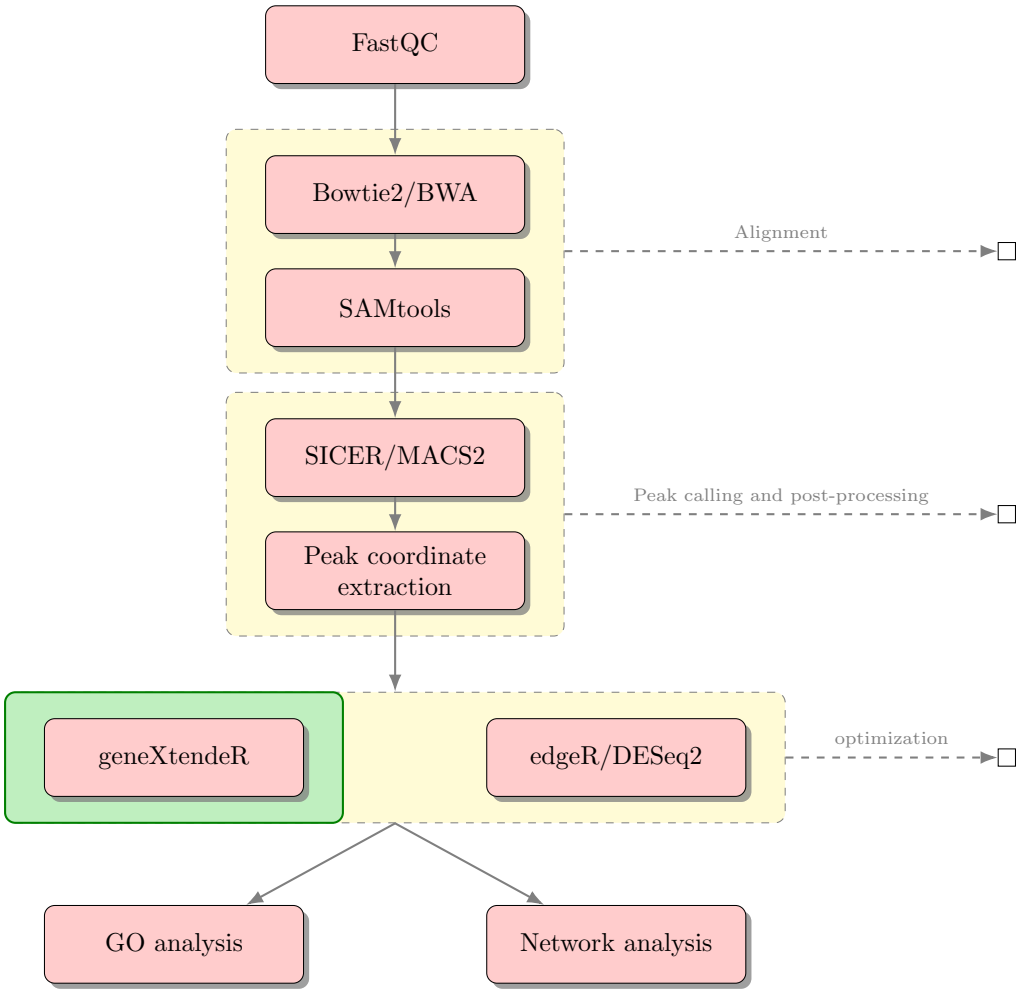
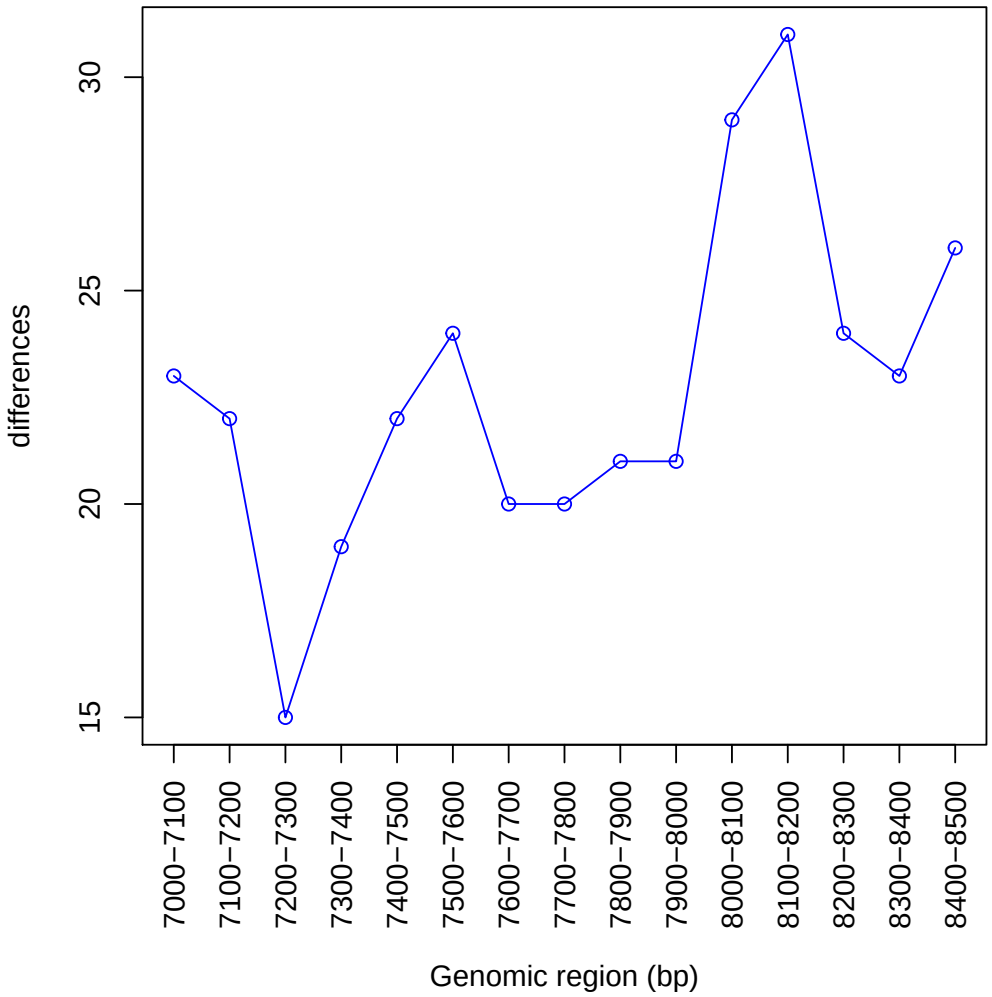


Figure 1: Sample biological workflow using geneXtendeR in combination with existing statistical software to analyze peak significance. Subsequent gene ontology or network analysis may be conducted on genes associated with statistically significant peaks.

It is entirely possible (and probable) for significant peaks to be present at relatively high upstream extension levels (i.e., large gene-spheres), albeit these significant peaks may be associated with biology not directly relevant to the study at-hand, due mainly to the sheer magnitude of the distance of the peak from traditional gene boundaries (where traditional gene boundaries are defined as $+/- \approx 3$ kb from TSS and $+/- \approx 0.5$kb from TES). Consequently, it is normal for peaks-on-top-of-genes to exhibit higher levels of noise at higher upstream extension levels. However, this does not mean that potential enhancer activity should be discounted. For instance, it is not uncommon to see a steady rise or even a surge in the number of peaks-on-top-of-genes at higher upstream extension levels:

```
> linePlot(rat, 7000, 8500, 100)
```



In far-out cases like this, it is particularly recommended to examine the statistical significance of peaks to get a sense for the possibility of potential enhancer activity. Of course, such computational findings would require experimental follow-up and/or database mining for known motifs.

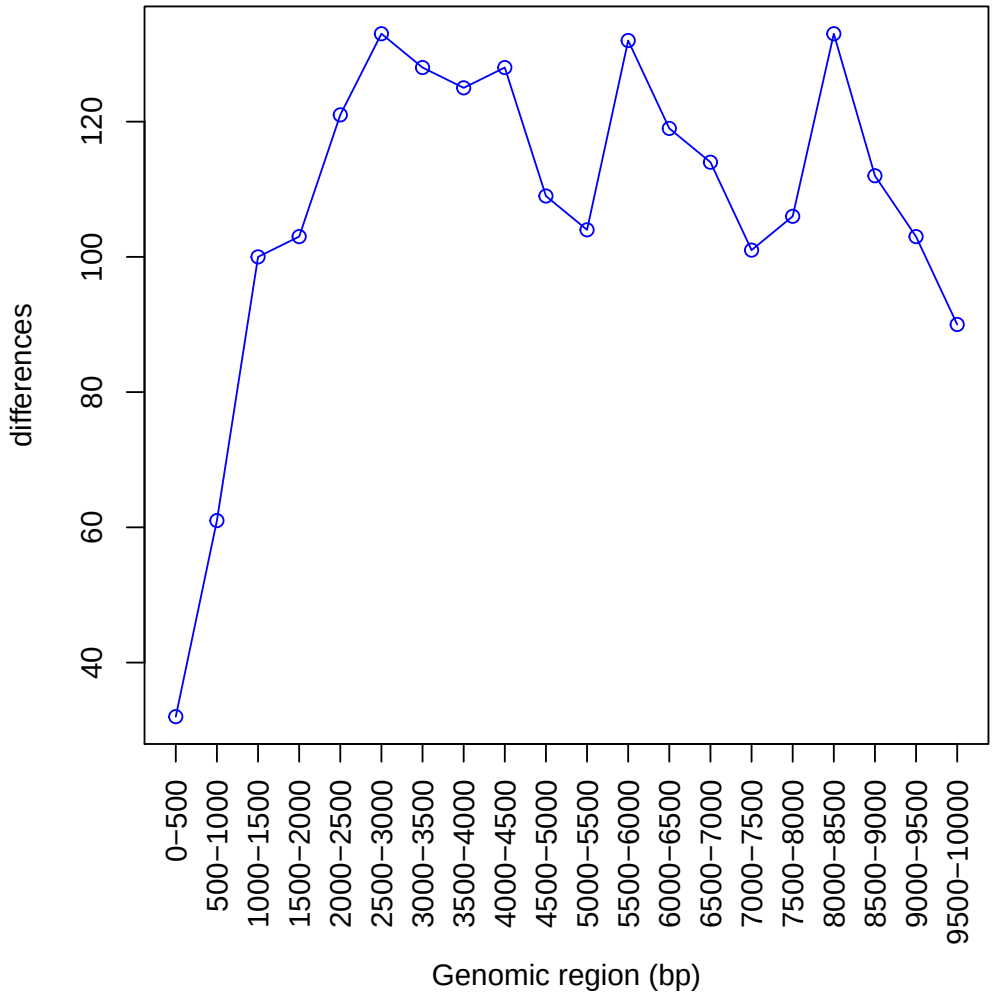
Assessment of such statistical significance values is beyond the scope of geneXtendeR, in order to allow the user freedom to choose the respective statistical package/technique. As before, first use the

`distinct()` command to create a table of unique genes located under peaks between the two upstream extension levels:

```
> distinct(rat, 7800, 7900)
```

Then, assess the statistical significance of these peaks using a combination of HT-seq and edgeR, or HT-seq and DESeq2, or some other appropriate combination of existing software tools. Genes associated with the resultant statistically significant peaks may then be further assessed with gene ontology analysis or network analysis to help answer a variety of interesting research questions. DNA sequences under peaks may be checked for the presence of known regulatory motifs (e.g., using TRANSFAC (Matys et al. 2006) or MEME/JASPAR (Sandelin et al. 2004, Bailey et al. 2009)), or for the presence of biological repeats (e.g., using RepeatMasker (Smit et al. 2015)). Pending a prospective GO/network analysis, functional validation may be followed up in the lab to test any potential regulatory sites or prospective enhancer elements, thereby bringing the computational analysis pipeline successfully back to the bench.

Even though geneXtendeR is designed to compute (and analyze/display) optimal gene extensions tailored to the characteristics of a specific peak input file, geneXtendeR will not explicitly impose on the user the optimal extension to use, since this information is highly study-dependent and, as such, is ultimately reserved to the user's discretion. For example, a user may choose a conservatively lower upstream extension (e.g., for studies investigating narrow peaks such as H3K4me3 or H3K9ac that exhibit a compact and localized enrichment pattern, where high upstream extensions may lose biological meaning). An example of such user-driven decisions would be the selection of a 1500 bp upstream extension instead of a 3500 bp extension in situations like this:



Likewise, a user may also investigate the statistical significance of specific peaks of interest at varying upstream cutoffs via the help of external software (e.g., HT-seq/edgeR, HT-seq/DESeq2, etc). Once the user has chosen the specific upstream extension to be used, the peak file is ready to be fully annotated:

```
> annotate(rat, 3300)
```

which generates a fully annotated peaks file containing various genomic features and labeled headers.