SOFTWARE TOOL ARTICLE

**REVISED Infrastructure for genomic interactions: Bioconductor classes for Hi-C, ChIA-PET and related experiments [version 2; peer review: 2 approved]**

Aaron T. L. Lun¹, Malcolm Perry², Elizabeth Ing-Simmons²

¹Cancer Research UK Cambridge Institute, University of Cambridge, Cambridge, UK
²MRC Clinical Sciences Centre, Faculty of Medicine, Imperial College London, London, UK

**Abstract**

The study of genomic interactions has been greatly facilitated by techniques such as chromatin conformation capture with high-throughput sequencing (Hi-C). These genome-wide experiments generate large amounts of data that require careful analysis to obtain useful biological conclusions. However, development of the appropriate software tools is hindered by the lack of basic infrastructure to represent and manipulate genomic interaction data. Here, we present the *InteractionSet* package that provides classes to represent genomic interactions and store their associated experimental data, along with the methods required for low-level manipulation and processing of those classes. The *InteractionSet* package exploits existing infrastructure in the open-source Bioconductor project, while in turn being used by Bioconductor packages designed for higher-level analyses. For new packages, use of the functionality in *InteractionSet* will simplify development, allow access to more features and improve interoperability between packages.

**Keywords**

Hi-C, ChIA-PET, infrastructure, data representation, genomic interactions

---

This article is included in the RPackage gateway.

This article is included in the Bioconductor gateway.
Introduction

Techniques such as chromatin conformation capture with high-throughput sequencing (Hi-C)\(^1\) and chromatin interaction analysis with paired-end tags (ChIA-PET)\(^2\) are increasingly being used to study the three-dimensional structure and organisation of the genome. Briefly, genomic DNA is fragmented and subjected to a ligation step during which DNA from interacting loci are ligated together. High-throughput paired-end sequencing of the ligation products will identify pairs of interacting genomic regions. The strength of each interaction can also be quantified from the number of read pairs connecting the two interacting regions. This information can be used to derive biological insights into the role of long-range interactions in transcriptional regulation as well as the general organization of the genome inside the nucleus.

The analysis of Hi-C and ChIA-PET data is not a trivial task, and many software packages have been developed to facilitate this process. Several of these packages like diffHic\(^3\) and GenomicInteractions\(^4\) are part of the open-source Bioconductor project, which aims to provide accessible tools for analyzing high-throughput genomic data with the R programming language. One of the strengths of the Bioconductor project is the quality and quantity of shared infrastructure available to developers\(^5\). Pre-defined S4 classes such as GenomicRanges and SummarizedExperiment can be used to represent various types of genomic data and information, easing the maintenance burden for developers while also improving interoperability between packages for users. However, this kind of common infrastructure does not yet exist for the genomic interaction field. Instead, each package contains its own custom classes, which increases code redundancy and development load while reducing interoperability.

Here, we describe the InteractionSet package that provides base S4 classes for representing and manipulating genomic interaction data. It contains the GInteractions class, to represent pairwise interactions; the InteractionSet class, to store the associated experimental data; and the ContactMatrix class, to represent interactions in a matrix format. This facilitates code reuse across Bioconductor packages involved in analyzing data from Hi-C, ChIA-PET and similar experiments.

Overview of available classes

The GInteractions class

Each object of the GInteractions class is designed to represent interactions between pairs of “anchor” regions in the genome (Figure 1A). It does so by storing pairs of anchor indices that point towards a reference set of genomic coordinates (specified as a GenomicRanges object). Each anchor index refers to a specific reference region, such that a pair of such indices represents a pairwise interaction between the corresponding regions. This design reduces memory usage as the reference coordinates need only be stored once, even if each region is involved in multiple interactions. Computational work is also reduced as calculations can be quickly applied across the small set of reference regions, and the results can be retrieved for each interaction based on the anchor indices. In addition, the GInteractions class inherits from the Vector class in Bioconductor’s S4Vectors package. This allows storage of metadata for each interaction (e.g., intensities, \(p\)-values) and for the entire object (e.g., experiment description).

The InteractionSet class

The InteractionSet class is designed to store experimental data for each feature (Figure 1B). It inherits from the SummarizedExperiment base class, where each object of the class stores any number of matrices of the same dimensions. Each row of each matrix corresponds to a pairwise genomic interaction (represented by a GInteractions object that is also stored within each InteractionSet object), while each column corresponds to an experimental sample. Each entry of the matrix then represents the observation for the corresponding interaction in the corresponding sample. Different matrices can be used to store different types of data, e.g., read counts, normalized intensities. The InteractionSet class also inherits a number of fields to store metadata for each interaction, for each sample, and for the entire object.

The ContactMatrix class

The ContactMatrix class is designed to represent pairwise interactions in a matrix format (Figure 1C). Each row and column of the matrix represents a genomic region, such that each cell of the matrix represents an interaction between the corresponding row/column regions. Experimental data for that interaction can be stored in the associated cell. This provides a direct representation of the “interaction space”, i.e., the two-dimensional space in which \((x, y)\) represents an interaction between \(x\) and \(y\). Like the GInteractions class, the genomic coordinates are not stored directly – rather, the rows/columns have indices that point towards a reference set of coordinates, which reduces memory usage and computational work. The matrix representation itself uses classes...
Figure 1. Overview of the classes in the InteractionSet package. Relevant slots of each class (i.e., data values stored in each object of the class) are labelled with a preceding “@”. (A) The GInteractions class represents pairwise interactions between genomic regions by storing pairs of anchor indices that refer to coordinates in a GenomicRanges object. (B) The InteractionSet class stores experimental data in an “assays” matrix where each row is an interaction and each column is a sample. Here, counts represent the number of read pairs mapped between each pair of interacting regions in each sample. (C) The ContactMatrix class represents the interaction space as a matrix, where each cell represents an interaction between the corresponding row/column regions.

in the Matrix package to provide support for both dense and sparse matrices. The latter may be more memory-efficient, particularly for sparse areas of the interaction space. ContactMatrix instances can also be easily converted to instances of existing matrix-based classes such as those in the HiTC package.

Overview of available methods
The InteractionSet package provides a variety of methods for manipulating objects of each class. In addition to slot accessors and modifiers, methods are available to convert objects to different classes in the same package (e.g., GInteractions to ContactMatrix) or to base Bioconductor classes (e.g., GInteractions to GRangesList). The distance between anchor regions on the linear genome can be computed for each pairwise interaction, to use in fitting a distance-dependent trend for diagnostics or normalization. The minimum bounding box in the interaction space can also be defined for a group of interactions (Figure 2A) to summarize the location of that group.

The InteractionSet package supports one- or two-dimensional overlaps for its objects (Figure 2B). A one-dimensional overlap is considered to be present between an interaction and a genomic
interval if either anchor region of the interaction overlaps the interval. This can be used to identify interactions overlapping pre-defined regions of interest. A two-dimensional overlap is considered to be present between an interaction and two genomic intervals if one anchor region overlaps one interval and the other anchor region overlaps the other interval. This can be used to identify interactions linking two specific regions of interest, e.g., a gene and its enhancer. The same framework can be used to define two-dimensional overlaps between two interactions, based on whether the corresponding anchor regions overlap – this can be used to relate similar interactions in different GInteractions objects or across different experiments. More generally, interactions can be identified that link any two regions in a set of regions of interest. For example, given a set of genes, interactions between two genes can be identified; or given a set of genes and another set of enhancers, interactions linking any gene to any enhancer can be found.

Hi-C data in an InteractionSet object can also be converted into a 4C-like format (Figure 2C). Firstly, a bait region is defined as some region of interest, e.g., a target gene or enhancer. All interactions in the InteractionSet object that have one-dimensional overlaps with the bait are identified. For each overlapping interaction, the anchor region that does not overlap with the bait is extracted and – along with the data associated with that interaction – used to construct a RangedSummarizedExperiment object. This process yields data for intervals on the linear genome, which is similar to the output of 4C experiments that measure the

---

**Figure 2. Schematic of several methods in the InteractionSet package.** (A) Minimum bounding boxes can be identified for groups of interactions using the boundingBox method. Here, u', v' and w' belong in one group while x', y' and z' belong in another. (B) One- or two-dimensional overlaps can be identified between interactions and one or two genomic intervals, respectively, using the findOverlaps method. Here, x' and y' have one-dimensional overlaps with the gene and enhancer, respectively, while z' has a two-dimensional overlap with the gene and the enhancer. (C) An InteractionSet object contains data – in this case, read pair count data – for interactions in the two-dimensional interaction space. Given a bait region, a “cross-section” of the space can be extracted and converted into a RangedSummarizedExperiment object using the linearize method. This object holds count data for intervals on the linear genome (blue lines) where the count for each interval describes the strength of the interaction between that interval and the bait. This format effectively mimics that of 4C data.
intensity of interactions between the bait and all other regions. The “linearized” format may be preferable when a specific region can be defined as the bait, as intervals on the linear genome are easier to interpret than interactions in two-dimensional space.

Implementation and operation details
All classes and methods in the InteractionSet package are implemented using the S4 object-orientated framework in R (version 3.3.0 or higher). Classes are exported to allow package developers to derive custom classes for their specific needs. Pre-existing Bioconductor classes and generics are used to provide a consistent interface for users. After loading the InteractionSet package into an R session, instances of each class can be constructed from existing data structures, either directly (e.g., GInteractions objects from GRanges via the GInteractions constructor, or from Pairs via the makeGInteractionsFromGRanges function; ContactMatrix objects from GRanges and Matrix via the ContactMatrix constructor) or in a hierarchical manner (e.g., InteractionSet objects from matrices and a Ginteractions object via the InteractionSet constructor).

The methods described above can then be applied to each instance of the class. While the InteractionSet package does not have functions to load data from file, it can be combined with the import function in the rtracklayer package to construct class instances after importing data from a range of formats including BED and BEDPE. A similar strategy can be used to export data to file.

Conclusions
The availability of common infrastructure is highly beneficial to software development by reducing redundancy and improving reliability, as more developers can check the same code; improving interoperability, as different packages use the same classes; and increasing the accessibility of useful features, which exist in a single package rather than being sequestered away in a variety of different packages. Here, we present the InteractionSet package that implements a number of classes and methods for representing, storing and manipulating genomic interaction data from Hi-C, ChIA-PET and related experiments. The package is fully integrated into the Bioconductor ecosystem, depending on a number of base packages to implement its classes (e.g., S4Vectors,

References
Open Peer Review

Current Peer Review Status: ✔ ✔

Version 1

Reviewer Report 01 June 2016

https://doi.org/10.5256/f1000research.9426.r14093

© 2016 Servant N. This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Nicolas Servant
Institut Curie, Paris, France

The authors present the InteractionSet package, that eases the manipulation of chromosome conformation data within the BioConductor/R framework. The InteractionSet package was designed to store direct interactions between two genomic loci. It also proposed a ContactMatrix class allowing to store the interaction counts as a Matrix format. One important point is its ability to be generic allowing the manipulation of any type of interaction data, such as ChIA-PET, Hi-C or 4C data. This work provides an interesting base for package development in this field and should therefore be of great use to the community.

In practice, the manuscript highlights the quality of the implementation and the optimization of this package. Dealing with Hi-C data can be challenging as the amount of data can be very large. Through this manuscript (Figure 1), it is clear that the authors propose an efficient strategy to manipulate such data.

In addition, the package is well documented with a quick start guide (vignette) and the description of each function. The new classes are based on existing S4 classes and methods and should therefore be easy to use for users familiar with the intervals manipulation in R. The package is already used as a dependency of other packages such as GenomicInteractions and diffHiC. Finally, it is compatible with other existing BioConductor packages such as the HiTC package.

Regarding the manuscript itself, it clearly describes what the InteractionSet does. It is well written and easy to read.

I only have a few minor comments that I hope will help the authors to improve the manuscript and/or the package.

1. Storing direct interaction counts looks very interesting in practice. I'm just wondering how efficient is the GInteractions class in term of scalability and memory usage ? As an example, Rao et al recently generated Hi-C contact maps at a resolution of 5kb. This very high resolution dataset implies billion of Hi-C contacts. It would be interesting the know up to which resolution (or data throughput) the InteractionSet package is efficient, and/or which amount of RAM is require to deal
1. The authors mentioned that the ContactMatrix class is compatible with matrix-based classes from the HiTC package. I therefore tried to convert a ContactMatrix object into an HTCexp object from HiTC (using the as() function) but it doesn't work. A note-example about that might be useful in the manual.

2. The package requires a recent version of R (>=3.3.0). It might be good to mention it somewhere.

**Competing Interests:** No competing interests were disclosed.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 02 Jun 2016

Aaron Lun, Cancer Research UK Cambridge Research Institute, UK

Thanks for your comments Nicolas. Our responses are below:

*Storing direct interaction counts looks very interesting in practice. I'm just wondering how efficient is the GInteractions class in term of scalability and memory usage? As an example, Rao et al recently generated Hi-C contact maps at a resolution of 5kb. This very high resolution dataset implies billion of Hi-C contacts. It would be interesting the know up to which resolution (or data throughput) the InteractionSet package is efficient, and/or which amount of RAM is require to deal with very large dataset.*

In several Hi-C analyses that we have performed (50 kbp resolution, ~1 billion reads), the size of the InteractionSet object is around 100MB to 1GB. This is well within the capacity of modern desktop machines, let alone high performance computing (HPC) facilities. For smaller bin sizes, the quadratic increase in the potential number of bin pairs is mitigated by the fact that a greater number of those bin pairs will be empty. We only store non-empty interactions in GInteractions/InteractionSet objects, which avoids a quadratic increase in memory requirements. (In practice, further savings can be made by filtering to remove low-abundance interactions.)

That said, for very large data sets with read coverage across the entire interaction space, the memory requirements will increase dramatically at higher resolutions. This can be mitigated to some extent by only operating on a single chromosome (or pair of chromosomes) at any given time. However, if this is not possible (e.g., the downstream analysis requires all interactions), then HPC resources and 64-bit R may be required to handle the resulting objects. We feel that such requirements are mostly unavoidable, as the generation of large data sets requires concomitant effort in the computational analysis.

The authors mentioned that the ContactMatrix class is compatible with matrix-based classes from the HiTC package. I therefore tried to convert a ContactMatrix object into an HTCexp object from HiTC (using the as() function) but it doesn't work. A note/example about that might be useful in the manual.

Our concept of compatibility was based more on the class implementations and concepts, rather than through any explicit conversion. Specifically, both ContactMatrix and HTCexp use GRanges...
to represent the genomic coordinates of the row/column regions, and a Matrix class to represent store the intensity values across the interaction space. Thus, information extracted from an instance of a ContactMatrix class can directly be supplied to the HTCexp constructor:

```r
coords <- GRanges("chrA", IRanges(1:10, 1:10))
x2 <- ContactMatrix(Matrix(1:100, 10, 10), coords, coords) # dummy object
colnames(x2) <- rownames(x2) <- LETTERS[1:10] # dummy names
HTCexp(as.matrix(x2), anchors(x2, "column"), anchors(x2, "row"))
```

We are reluctant to implement this directly as an "as" method in the InteractionSet package, because we have tried to maintain a distinction between the low-level base classes in our package and the high-level analysis and visualization methods in other packages like HiTC, diffHic, GenomicInteractions, etc. Our hope is that developers who would like to use or be compatible with InteractionSet would write appropriate methods in their own packages to convert to/from classes as necessary.

The package requires a recent version of R (>=3.3.0). It might be good to mention it somewhere.

Done.

**Competing Interests:** No competing interests were disclosed.
and more complex functions such as findOverlaps that allows users to overlap sets of pairwise interactions in a variety of ways.

This package is very useful and powerful and provides a valuable resource to software developers and advanced users. The GenomicRanges-style organization of the data, that InteractionSet adopts, is often too complicated for casual R users to learn. In many cases simply reading files from BED, BEDPE, or sam format into data frames is easier and faster for simple tasks. However, developers will prefer this more standardized format for improved stability of their packages. And advanced users may prefer the standardized yet flexible approach to data organization and the powerful built in tools.

Importantly, the package is accompanied by a detailed and clear online tutorial which clearly demonstrates how to use the classes and functions. In summary, this paper is succinct and clearly written and accurately describes an R package that will be of great use to the scientific community.

**Competing Interests:** No competing interests were disclosed.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

---

**Author Response 02 Jun 2016**

**Aaron Lun, Cancer Research UK Cambridge Research Institute, UK**

Thanks for your comments Douglas. We agree that the Bioconductor ecosystem of data classes can be somewhat daunting for new users. Nonetheless, we believe that the use of standard Bioconductor tools is the safest strategy for the majority of users (and obviously developers), given the number of "gotchas" in data processing, e.g., off-by-one issues in BED file loading.

**Competing Interests:** No competing interests were disclosed.

---

The benefits of publishing with F1000Research:

- Your article is published within days, with no editorial bias
- You can publish traditional articles, null/negative results, case reports, data notes and more
- The peer review process is transparent and collaborative
- Your article is indexed in PubMed after passing peer review
- Dedicated customer support at every stage

For pre-submission enquiries, contact research@f1000.com