AIMS Biophysics

AIMS Biophysics, 6(3): 83–98.
DOI: 10.3934/biophy.2019.3.83
Received: 31 May 2019
Accepted: 16 July 2019
Published: 15 August 2019

http://www.aimspress.com/journal/biophysics

Research article

Data formats for modelling the spatial structure of chromatin based on experimental positions of nucleosomes

Michael-Christian Mörl, Tilo Zülske, Robert Schöpflin and Gero Wedemann*

Competence Center Bioinformatics, Institute for Applied Computer Science, University of Applied Sciences Stralsund, Zur Schwedenschanze 15, 18435 Stralsund, Germany

*** Correspondence:** Email: gero.wedemann@hochschule-stralsund.de.

Abstract: In the nucleus of eukaryotic cells, DNA is wrapped around histone proteins, forming units termed nucleosomes. Nucleosome chains fold into chromatin. Despite extensive experimental advancement, many fundamental features of chromatin remain uncertain. Since all cell types and states cannot be profiled experimentally, especially in solution and *in vivo*, computer simulations are valuable tools for research. Most computer simulation models of chromatin are coarse-grained and describe the main characteristics of 3D chromatin packing. Newer models include experimentally derived positions of nucleosomes. While it is common practice in other disciplines, such as systems biology, to make experimental data publicly available, data from computer simulations of chromatin models are not usually published. Thus, data standard exchange formats are lacking, and we address this issue in the present work. We analysed the workflow, from experimental determination of the positions of nucleosomes through to analysis of outputs from simulated computer models. We defined standardized formats based on Extensible Markup Language (XML) for artefacts generated by steps in this workflow. We found that XML is useful since it is easy to transform XML-based-files by applying Extensible Stylesheet Language Transformations (XSLT) to other formats. We demonstrate the viability of this approach and the associated file formats using a complete example of computer simulation of chromatin domains based on experimentally determined nucleosome positions. The XML schemas and examples are published in an open source repository.

Keywords: exchange format; xml; chromatin; computer simulation

1. Introduction

The spatial structure of chromatin has an important impact on gene regulation [1]. It is closely linked to the positions of nucleosomes in the genome, which are actively regulated by cells [2]. Computer models based on Monte Carlo, Brownian Dynamics or Molecular Dynamics can be applied to investigate the connection between the positions of nucleosomes and the spatial structure of chromatin [3,4]. Computer models differ with respect to their resolution level, among other properties. For chains with many nucleosomes coarse-grained models are used, whereas molecular models with atomic resolution are more detailed and can represent single nucleosomes. In the present work, we study the usage of coarse-grained models, beginning with experimental determination of the positions of nucleosomes and ending with analysis of data from computer simulations. We focused at the level of a particular locus in the genome of a specific cell type. For all steps in this process, a multitude of bioinformatics and computational biophysics tools are applied that require different input formats and deliver various output formats, which makes the process tedious [5].

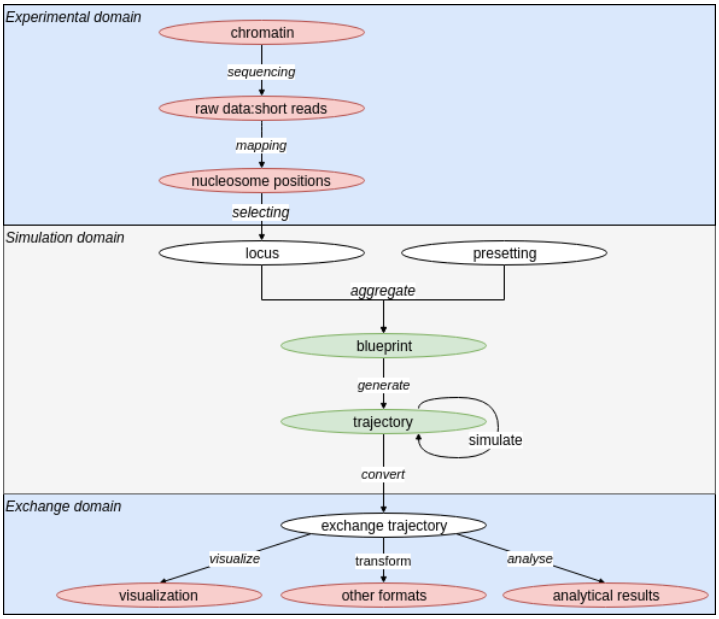

Figure 1. Experimental, simulation and exchange domain features constitute a workflow for the simulation of chromatin based on nucleosome positions. Domains are shown as shaded areas, artefacts are represented by ellipses, workflow steps are symbolized by arrows. The newly developed public exchange formats are depicted as white ellipses. Artefacts used internally by our research group are shown as green ellipses, and external formats are indicated by red ellipses.

For the development of data formats we analysed the workflow used in our research group as an example. We identified three domains; experimental, simulation and analysis (Figure 1). Each domain involves several steps generating different artefacts.

The experimental domain is where chromatin is prepared and experimentally investigated. Nucleosomes are isolated from chromatin by applying e.g., micrococcal nuclease (MNase), and DNA fragments are isolated and sequenced to generate raw data in the form of short reads.

Nucleosome positions are determined by mapping the obtained sequences to a reference genome using dedicated software [6–8] or existing nucleosome position databases [9,10].These positions usually overlap, reflecting different positions in different cells from a sample, and experimental shortcomings. For computer models, individual non-overlapping nucleosome positions at a particular locus of interest are identified using appropriate software tools [11].

The simulation domain comprises all steps required to prepare and perform a computer simulation of a locus, and files contain the positions for a genomic region of interest. Additionally, a presetting file contains the settings for the 3D computer simulation model and the simulation process, such as nucleosome geometry, and parameters for potentials and simulations. Locus and presetting parameters are aggregated into a single blueprint, and the simulation software creates a trajectory that contains the positions and orientations of the model elements for all simulation steps.

In the exchange domain, simulation results are made accessible for other research groups. This can be done by converting the trajectory into an easy to understand exchangeable format, although this requires greater disk space. Published data can then be used for visualisation and analysis.

Computer simulations are performed differently in different research groups [12,13] utilizing various data formats and different chromatin models [14–16]. In the present work, we propose common data formats for presetting and locus parameters, and an exchange trajectory based on the Extensible Markup Language XML [17]. We believe that data can be expressed in a uniform way for different applications since the information contained within is essentially of a universal structure (Figure 2). We demonstrate the viability of the approach herein using a working example. The value for other groups will be corroborated by examples of Extensible Style Language Transformations XSLT [18] for analysis of exchange trajectories and transformation into other data formats. The XML Schema files and examples of XML files are published in open source repositories.

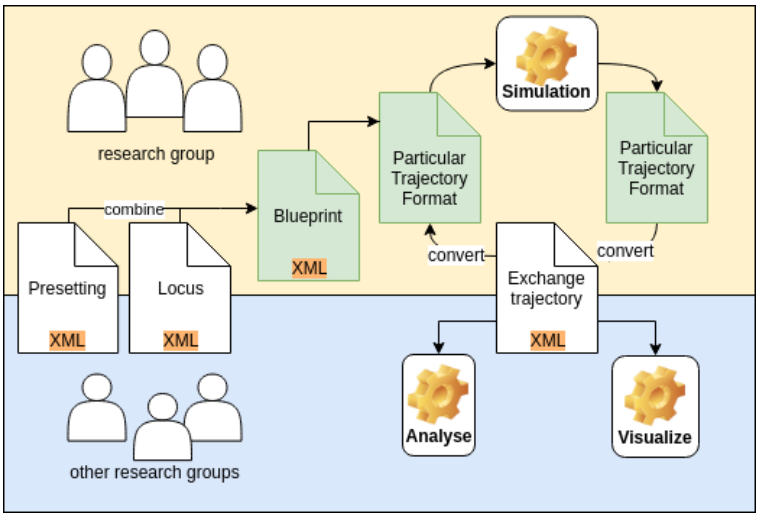

Figure 2. Communication between different research groups. The yellow area defines the scope of a research group for performing computer simulations. The blue area defines the scope of other research groups analysing data from the yellow group. Presetting, locus and exchange trajectory files are placed at the interface of the blue and yellow areas to illustrate their use for communication between different research groups.

2. Materials and methods

2.1. Extensible Markup Language (XML)

The locus, presetting and exchange trajectory data formats are freely available if (1) the format specification is unrestrictedly accessible, (2) use must not be subject to any fees or licenses, and (3) it must be interoperable (i.e., the data format must be platform-, hardware- and software-independent). A data file must be correct in terms of its structure and content. The clarity and usefulness of data can be assessed by means of a defined syntax and grammar. Data for an exchange trajectory can be validated utilising a schema describing the validity of the type and value range. In order to ensure the usability of data, data formats must not only be machine-readable, but also human-readable (i.e., text instead of a binary format). Broad and advanced software support should be available (e.g., communities, libraries, etc.) to minimise the effort for the processing an exchange trajectory. All the above criteria are met by the Extensible Markup Language (XML [17]), specified by the World Wide Web Consortium (W3C) [19]). The aforementioned data schema can be defined using an XML Schema.

2.2. UML-Class diagram

In software development, object-oriented methods are regularly used to model structure and behaviour. Unified Modelling Language (UML) is a widespread standard [20]. In this article we use UML class diagrams to model classes and their relations as foundation for the XML formats. See the UML standard [20] for details, since most software engineering textbooks and even Wikipedia use the outdated UML 2.0 standard.

2.3. Classes/Objects

A class defines common properties of objects. In this context, objects are also referred to as instances of a class. For example, Figure 3 shows the nucleosome as a class with the properties pos and mpos. The text after the colon describes the type of the property e.g., integer. Instances of a class have its properties with concrete values. A class may be labelled as <abstract> which means that it cannot be instantiated, and only derived classes can be instantiated (s. section Associations between Objects). An example for an abstract class is the class Bead in Figure 6. Here, only DNABead and NuclesomeBead can be instantiated.

2.4. Relations between Objects

In UML different relations between objects can be modelled.

An association denotes a link between two classes and is depicted by a line between the two classes. A filled dot at a class signifies that the other class of the association owns the association. As example in Figure 5 Presetting owns the association to ParameterList. If no numbers or symbols are attached to the end of the line one instance of a class is associated to the other. A star indicates that an indefinite number of instances of that class are associated. As example in Figure 4 the association

between Locus and Nucleosome symbolises, that a single instance of Locus can have an association to an indefinite number of instances of Nucleosome.

Inheritance describes a specialisation of an object A to an object B. One also says that B is derived from A. This means, that B inherits the properties and associations of A and has additional properties or associations that are special to B. A line with an open triangle as arrowhead in direction of the generalization represents graphically an inheritance. An example for inheritance in Figure 6 the class DNABead is derived from Bead.

3. Results

Herein, we propose common data formats for locus, presetting and exchange trajectory files. For illustration purposes, we demonstrate our format for the blueprint.

3.1. Locus file data

A locus is located on a chromosome and is delimited by a first and a last nucleotide (base). It contains the positions of identified nucleosomes. The position of a nucleosome is defined by the positions of the two bases that delimit the nucleosome in the genome. The positions are derived experimentally, and these relations can be modelled in a straightforward way using two classes with a one-to-many association, as shown in Figure 3. The results are presented using an XML schema, as shown in Figure 4.

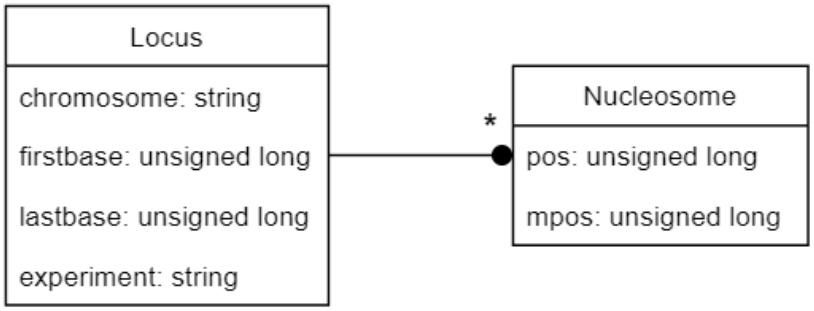

Figure 3. Schematic structure of a locus modelled as a UML class diagram [20]. The locus is defined experimentally (experiment), as are and the chromosome (chromosome) of which it is a part and the positions of the two bases (firstbase, lastbase) that delimit the locus in the genome. Furthermore, a locus includes nucleosome objects, also delimited by two bases (pos, mpos) on the mapped nucleosome of the genome. The int and string data types are built into the XML schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:locus="http://bioinformatics.hochschule-stralsund.de/locus"
targetNamespace="http://bioinformatics.hochschule-stralsund.de/locus"
elementFormDefault="qualified" >
<complexType name="nucleosome">
    <sequence>
      <element name="pos" type="unsignedLong" maxOccurs="1" minOccurs="1"/>
      <element name="mpos" type="unsignedLong" maxOccurs="1" minOccurs="1"/>
    </sequence>
</complexType>
<complexType name="locus">
    <sequence>
    <element name="nucleosome" type="locus:nucleosome"
      maxOccurs="unbounded" minOccurs="1"/>
    </sequence>
    <attribute name="chromosome" type="string"/>
    <attribute name="firstbase" type="unsignedLong"/>
    <attribute name="lastbase" type="unsignedLong"/>
    <attribute name="experiment" type="string"/>
</complexType>
<element name="locus" type="locus:locus"/>
</schema>
```

Figure 4. Locus schema (Extensible Markup Language-Schema file (XSD)).

3.2. Presetting file data

The presetting file defines the settings for the simulation model and the simulation process. These are mostly the geometry of the DNA at the nucleosomes, the parameters of the different energy potentials, and the simulation parameters [12]. The geometry of a nucleosome is defined as the location and orientation of the DNA at the nucleosome, and can be described by seven angles (α, β, γ, δ, ε, φ and ζ), the segment length in the elastically relaxed state, and the distance between the centre of gravity of the nucleosome and the segment centre (Figure 7) [21]. The nucleosome potential is determined by selecting one of the models implemented in our simulation software (e.g., Lennard-Jones or Gay-Berne). To exclude invalid angles or distances, the data schema uses its own data types for floating point values. It is also possible to define additional simulation parameters. The presetting structure is shown schematically in Figure 5.

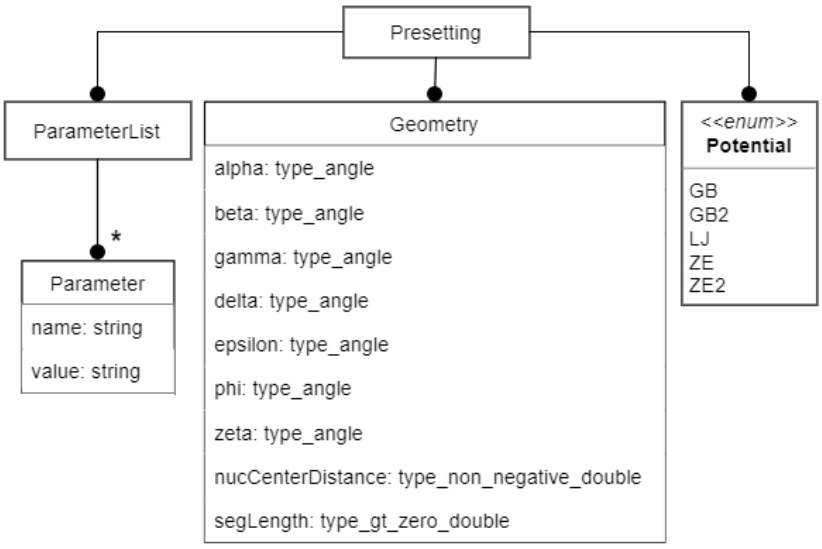

Figure 5. Schematic structure of a presetting file as a Unified Modelling Language (UML) class diagram. The presetting file consists of nucleosome geometry (Geometry), nucleosome potential (potential), and simulation parameters (type_parameter_list). The nucleosome geometry consists of the seven angles, the equilibrated segment length, and the distance between the centre of gravity of the nucleosome and the segment centre. The nucleosome potential has been designed as an enumerator comprising Gay-Berne, Lennard-Lones and Zewdie models, and their variants [22]. The simulation parameters are described as generic key-value pairs. The data types type_non_negative_double, type_gt_zero_double and type_angle limit the value ranges of floating-point values. Double and string data types are built into the XML Schema.

3.3. Exchange trajectory file data

An exchange trajectory file contains the results of a computer simulation, in particular the type, positions and orientations of all elements for different simulation steps. It includes information about the origin of the data (i.e., the conversion of a particular trajectory format, the time stamp of the conversion, the name and version of the converter, and the version of the data schema on which the exchange trajectory is based). The parameter list contains parameters for simulations as given in the presetting file. Parameters can be changed during the simulation (e.g., the temperature in simulations can be altered by applying simulated annealing) [22]. Altered parameters can be associated to configurations in simulation steps, at which point new parameter values are valid. The resulting structure of an exchange trajectory can be represented as a UML class diagram (Figure 7).

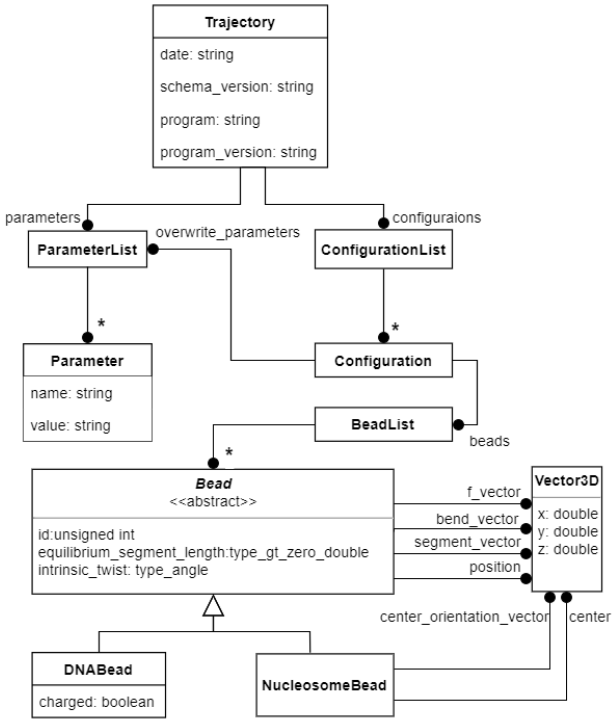

Figure 6. Schematic structure of an exchange trajectory in the form of a UML class diagram. A trajectory consists of parameters (ParameterList) and configurations (ConfigurationList), and contains information such as creation time (date), the version of the XML schema (schema_version), as well as the converter name (program) and version (program_version). Parameters are key-value pairs (name, value). Configurations include a segment chain (BeadList) after a number of performed Metropolis Monte Carlo steps (mc_steps). Segment chains consist of nucleosomes (NucleosomeBead) and DNA segments. Each segment is assigned a position in the chain (id) and contains its angle of intrinsic twist (intrinsic_twist), its length at equilibrium (equilibrium_segment_length), its position and the vector describing it orientation (f_vector), the vector to the next bead (segment_vector), and the vector describing the equilibrium position of the next bead (bend_vector) [22]. A DNA segment also has a flag stating whether it is charged (charged). Nucleosome segments also include information on the centre of gravity (centre) and the normal unit vector (center_orientation_vector). The coordinates of a spatial point or vector are represented by a type (Vector3D). The data types type_angle and type_gt_zero_double are used to limit the ranges of floating-point values. The data types Boolean, double, string and unsignedInt used here are built into the XML schema.

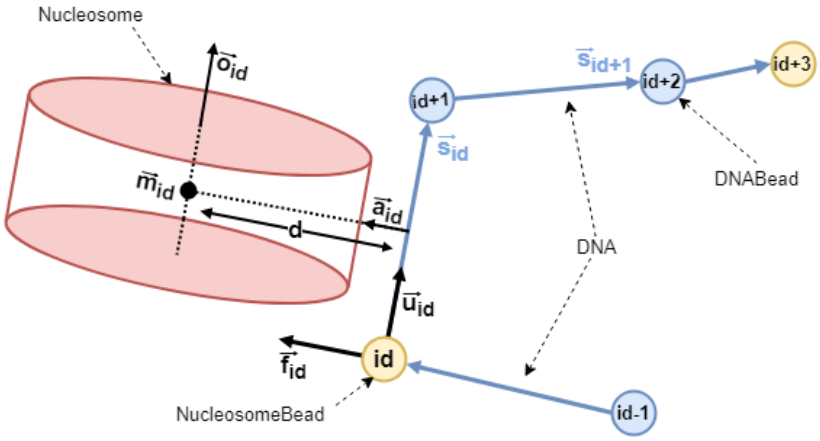

Figure 7. Model of a bead chain. 'id' represents the position of the bead in the chain, yellow circles indicate nucleosome bead positions, and blue circles indicate DNA bead positions. The nucleosome is represented by a red cylinder. The segment vector $\vec{s}$ points from one bead to the next bead. A local coordination system characterised by the normalised vectors $\vec{u}, \vec{f}$ *and* $\vec{v}$ (not shown) describe the orientation of a bead, where $\vec{u}$ is the normalized segment vector. Vector $\vec{a}$ (centre vector) describes the direction from the segment to the nucleosome centre, and d is the distance between them. Vector $\vec{m}$ determines the geometrical center of the nucleosome, and vector $\vec{o}$ describes the orientation of the nucleosome and is named the centre-orientation-vector.

3.4. Example workflow

The workflow and the suitability of the schema files are demonstrated using an example locus. Eight nucleosomes are defined, and their positions are included in a locus.xml-file (Figure 8) for which correctness was validated by the corresponding XML-schema file (Figure 4). The simulation conditions are specified in a presetting.xml file (Figure 9).

```
<?xml version="1.0" encoding="UTF-8"?>
<locus experiment="experiment"
chromosome="chr4"
firstbase="12345"
lastbase="12998"
    xmlns="http://bioinformatics.hochschule
-stralsund.de/locus" ...>
  <nucleosome>
    <pos>12384</pos>
    <mpos>12534</mpos>
  </nucleosome>
  <nucleosome>
    <pos>12574</pos>
    <mpos>12729</mpos>
  </nucleosome>
  <nucleosome>
    <pos>12767</pos>
    <mpos>12910</mpos>
  </nucleosome>
  ...
</locus>
```

Figure 8. Example of an XML file describing eight nucleosomes. Only the first three are shown.

```
<?xml version="1.0" encoding="UTF-8"?>
<presetting ...
xsi:schemaLocation=
"http://bioinformatics.hochschule-stralsund.de/presetting
../presetting.xsd ">
<geometry>
<alpha>50.0</alpha>
<beta>-140.0</beta>
<gamma>0.0</gamma>
<delta>0.0</delta>
<epsilon>-70.0</epsilon>
<phi>30.0</phi>
<zeta>10.0</zeta>
<nucCenterDistance>3.3</nucCenterDistance>
<segLength>8</segLength>
</geometry>
<parameterlist>
<parameter>
<name>MIN_SEGMENT_LENGTH</name>
<value>2</value>
</parameter>
...
</parameterlist>
<potential>LJ</potential>
</presetting>
```

Figure 9. Example of an XML file describing the presetting of the simulation. The 'geometry' defining the orientation of the DNA segments at the nucleosomes is shown with a single parameter value, while other parameters are omitted for clarity.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<configuration buildMode="straight" closed="false"
nucPotential="NUC_POT_LJ" type="chromatin">
<parameters>
<parameter name="NumberOfBeads" value="8"/>
<parameter name="Temperature" value="293"/>
...
</parameters>
<beadPrototype>
<dna id="dna1">
<segmentLength>6.8</segmentLength>
</dna>
<nuc id="nuc1">
<segmentLength>8</segmentLength>
<alpha>50</alpha><beta>-32</beta>
<gamma>0</gamma><delta>0</delta>
<epsilon>-70</epsilon><phi>0</phi><zeta>-50</zeta>
<nucCenterDistance>3.3</nucCenterDistance>
</nuc>
...
</beadPrototype>
<chain>
<dna position="0" ref_prototype="dna1"/>
<dna position="1" ref_prototype="dna1"/>
<nuc position="2" ref_prototype="nuc1"/>
<dna position="3" ref_prototype="dna2"/>
<dna position="4" ref_prototype="dna2"/>
<nuc position="5" ref_prototype="nuc2"/>
<dna position="6" ref_prototype="dna3"/>
<dna position="7" ref_prototype="dna3"/>
</chain>
</configuration>
```

Figure 10. Example of an XML file describing the blueprint.

The locus and presetting files were used to generate a blueprint.xml file (Figure 10). For the generation process, an internal tool from our simulation software was employed. The blueprint was

used to generate a trajectory format suitable as an input for our simulation software. All steps described so far were performed prior to the simulation being executed. For the example simulation, 5000 Monte-Carlo steps were performed. The results are in the form of an extended trajectory file containing the generated configurations, and this is converted to an exchange trajectory XML file (Figure 11).

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<trajectory ... xsi:schemaLocation=
"http://bioinformatics.hochschule-stralsund.de/trajectory trajectory.xsd">
...
<parameters>
   <parameter>
    <name>NumberOfBeads</name>
    <value>8</value>
  </parameter>
  ...
</parameters>
<configurations>
  <configuration>
   <mc_steps>0</mc_steps>
   <overwrite_parameters/>
   <beads>
    <dnabead>
     <id>0</id>
     <equilibrium_segment_length>6.8</equilibrium_segment_length>
     <intrinsic_twist>0</intrinsic_twist>
     <position><x>0</x><y>0</y><z>0</z></position>
     <f_vector><x>0</x><y>0</y><z>1</z></f_vector>
     <bend_vector><x>1</x><y>0</y><z>0</z></bend_vector>
     <segment_vector><x>6.8</x><y>0</y><z>0</z></segment_vector>
     <charged>true</charged>
    </dnabead>
    <nucleosomebead>
     <id>2</id>
     <equilibrium_segment_length>8</equilibrium_segment_length>
     <intrinsic_twist>-4.13643033</intrinsic_twist>
     <position><x>13.6</x><y>0</y><z>0</z></position>
     <f_vector><x>0</x><y>0</y><z>1</z></f_vector>
     <bend_vector><x>-0</x><y>0.90630779</y><z>-0.42261826</z></bend_vector>
     <segment_vector><x>8</x><y>0</y><z>0</z></segment_vector>
     <center_orientation_vector><x>0.98480775</x><y>0.17364818</y><z>0</z>
     </center_orientation_vector>
     <center><x>18.16433324</x><y>-0.97745384</y><z>3.10098565</z></center>
    </nucleosomebead>
    ...
   </beads>
  </configuration>
  ...
 </configurations>
</trajectory>
```

Figure 11. Example of an XML file containing the exchange trajectory. Only selected parameters, beads and configurations are shown.

To demonstrate the possible usage of the file format, we show two examples utilising XSLT for analysis and transformation of a trajectory. In the first example, the end-to-end distance of the nucleosome chain is computed (Figure 12). This is the Euclidian distance between the starting point of the first segment and the end point of the last segment.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet … xmlns:trj="http://bioinformatics.hochschule-stralsund.de/trajectory">
<xsl:output method="text" indent="yes"/>
  <xsl:template match="/">
    <xsl:for-each
      select="/trj:trajectory/trj:configurations/trj:configuration">
      <xsl:for-each
        select="trj:beads/trj:dnabead|trj:beads/trj:nucleosomebead">
        <xsl:if test="position() = last()">
          <xsl:param name="first_bead_x"select="../trj:*[1]/trj:position/trj:x" />
          <xsl:param name="first_bead_y"select="../trj:*[1]/trj:position/trj:y" />
          <xsl:param name="first_bead_z"select="../trj:*[1]/trj:position/trj:z" />
          <xsl:param name="last_bead_x"
            select="trj:position/trj:x + trj:segment_vector/trj:x" />
          <xsl:param name="last_bead_y"
            select="trj:position/trj:y + trj:segment_vector/trj:y" />
          <xsl:param name="last_bead_z"
            select="trj:position/trj:z + trj:segment_vector/trj:z" />
      <xsl:value-of select="math:sqrt(math:power(($first_bead_x - $last_bead_x), 2) +
          math:power(($first_bead_y - $last_bead_y), 2) +
          math:power(($first_bead_z - $last_bead_z), 2))" />
        </xsl:if>
      </xsl:for-each>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

result.dat

```
52.72
22.726362370110945
15.771168974055538
…
```

Figure 12. XSL-Transformation file for calculation of the end-to-end distance of a polynucleosome based on a trajectory file. The results are shown in the bottom right corner.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version= "3.0" xmlns:trj=
"http://bioinformatics.hochschule-stralsund.de/trajectory" >
…
  <xsl:template match="trj:trajectory">
    atom default radius 0.7 name DEF
    <xsl:for-each select="trj:configurations/trj:configuration[1]
                          /trj:beads/trj:dnabead">
      <xsl:choose>
        <xsl:when test="/trj:trajectory/trj:configurations/trj:configuration[1]
                                    /trj:beads/trj:dnabead[1] = current()">
          atom <xsl:value-of select="trj:id"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:text/>,<xsl:value-of select="trj:id"/>
        </xsl:otherwise>
      </xsl:choose>
      <xsl:if test="/trj:trajectory/trj:configurations/trj:configuration[1]
            /trj:beads/trj:dnabead[last()] = current()">
        <xsl:text> </xsl:text>radius 12 name DNA
      </xsl:if>
    </xsl:for-each>
    …
  </xsl:template>
</xsl:stylesheet>
```

```
atom default radius 0.7
atom 0,1,3,4,6,7 radius 12
atom 2,5 radius 46.7

# Bonds
bond 0::7
# set the unitcell
unitcell 480.0 260.0 90.0

timestep
0   0   0
68   0   0
136   0   0
216   0   0
280.59999999999997   0   0
345.20000000000005   0   0
425.20000000000005   0   0
476.2   0   0

timestep
0.0077324   -0.19   1.5089507
67.99   0.54   0.16
134.04   16.6781651   -0.66
…
```

Figure 13. XSL-Transformation file for transformation of a trajectory to a VTF file (left). The VTF format makes it possible to visualise trajectory data using VMD-Viewer. The start of the resulting VTF file is shown on the right.

As a second example, we transformed the exchange trajectory into another trajectory format, VTF [23], which can be used with VMD-Viewer [24]. This can be done also easily with XSLT (Figure 13).

As an example for the whole workflow, we analysed a locus in mouse cells at chromosome 14 from base pair 47589000 to 47593031 (genome mm9). We evaluated experimental data from MNase-seq experiments of mouse embryonic stem cells (ESC) (GSM1004653), Neural Precursor Cells (NPC) (GSM1004652) and mouse embryo fibroblast (MEF) (GSM1004654) cells (BED-Files, zero based coordinates, right open interval) [25,26]. Positions of nucleosome were determined applying the software NucPosSimulator using default settings [11] identifying between 22 and 24 nucleosomes (Table 1). We simulated the three cell types according to the described workflow with our simulation software [3] for 1000000 Monte Carlo steps, saved all 2000 steps and transformed each last configuration of the exchange trajectories for visualization with the raytracer POV-Ray [27] (Figure 14). Moreover, we analysed the average end-to-end distances of the chain. From analysis we excluded the first 25 values respectively 50000 steps needed for equilibration [28]. The results are presented in Table 1. Surprisingly, the chromatin model for MEF cells have longest elongation despite the largest number of nucleosomes caused by the positions of the nucleosomes.

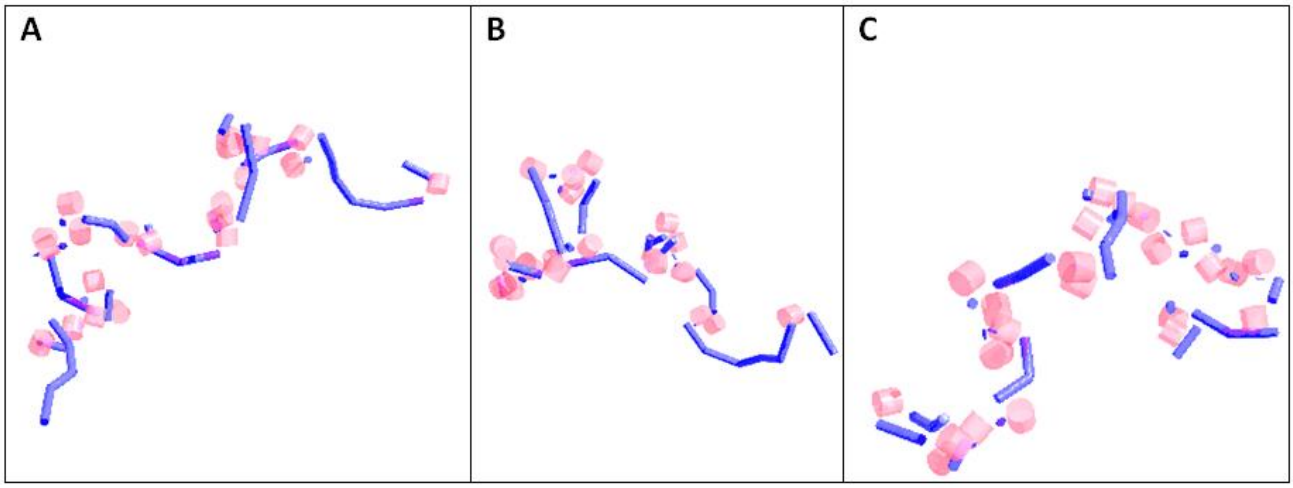

Figure 14. POV-Ray [26] visualization of snapshots from chromatin chains based on experimental data from MNase-seq experiments of mouse **(A)** ESC (GSM1004653 [25]), **(B)** NPC (GSM1004652 [25]) and **(C)** MEF (GSM1004654 [25]) cells from base pair 47589000 to 47593042) at chromosome 14 [26]. Red cylinders represent the nucleosomes and blue cylinder represent DNA.

Table 1. Mean end-to-end distances, standard deviation and number of nucleosomes from Monte Carlo-simulations for chromatin chains based on experimental data from MNase-seq experiments of mouse (A) ESC (GSM1004653 [25]), (A) NPC (GSM1004652 [25]) and (C) MEF (GSM1004654 [25]) cells from base pair 47589000 to 47593042) at chromosome 14 [26].

Type of Cell	End-To-End Distance in nm	#Nucleosomes
ESC	90,50 ± 32,31	22
NPC	89,44 ± 29,30	23
MEF	97,68 ± 28,67	24

4. Discussion and conclusions

In this article, we present data formats for use in a workflow for computer simulation of chromatin based on experimental nucleosome positions that are easy to read and process. The formats establish inputs and outputs for the simulation domain as XML schemas. We used an example workflow on a locus to demonstrate the viability of our approach. Specifically, three XSL transformations demonstrated the ease with which the XML format can be processed. XML is superior to other text formats such as JSON [29] since it has standardised schema definitions and permits transformations using XSLT. Sophisticated libraries exist for nearly all programming languages. XML is the standard file format used by the systems biology community, especially the Systems Biology Markup Language (SBML) [30]. On the downside, XML files tend to require much more disk space than terser formats such as our internal trajectory format or binary formats (Table 2). These data formats are less readable, which hinders analysis by other groups. For publication of trajectories, we recommend only to publish uncorrelated configurations, and to compress the trajectory using standard tools such as bzip2 to minimise disk space.

Table 2. Comparison of file size for Block-Trajectory, Exchange-Trajectory and Compressed Exchange Trajectory formats with an increasing number of configurations.

#Configurations	Block-Trajectory	XML-Trajectory	Compressed
100	2.55 Mb	26.5 Mb	895.6 Kb
1250	31.33 Mb	327.8 Mb	10.8 Mb
2500	63.00 Mb	655.1 Mb	21.9 Mb

The proposed formats are limited to systems containing DNA and nucleosomes. Next-generation computer models include explicit models of linker histones [31], CTCF or Cohesin [32]. The format will be advanced in the future to include these elements. The proposed file format has possible shortcomings that may inhibit use by other groups. For example, the angles describing the geometry of nucleosomes are very specific for our model. However, chromatin has been investigated for more than 20 years using computer simulations [33], and a common file format for model and simulation results has not yet been proposed. Simulation data are usually not published, and the present work is the first to attempt to establish a common format. We intend to publish all future data in the proposed format and invite other groups to participate in this endeavour.

Data Availability

All mentioned files are included in the supplemental material and are publicly available at https://github.com/HOSTbioinformatics/ExchangeFormats

Acknowledgements

We thank Antonio Tilocca for critical reading and copyediting of the manuscript.

Conflict of interest

The authors declare no conflicts of interest.

References

1. Lanctôt C, Cheutin T, Cremer M, et al. (2007) Dynamic genome architecture in the nuclear space: regulation of gene expression in three dimensions. *Nat Rev Genet* 8: 104–115.
2. Diermeier S, Kolovos P, Heizinger L, et al. (2014) TNFα signalling primes chromatin for NF-κB binding and induces rapid and widespread nucleosome repositioning. *Genome Biol* 15: 536.
3. Müller O, Kepper N, Schöpflin R, et al. (2014) Changing chromatin fiber conformation by nucleosome repositioning. *Biophys J* 107: 2141–2150.
4. Bajpai G, Padinhateeri R (2018) Irregular chromatin: packing density, fiber width and occurrence of heterogeneous clusters. *bioRxiv*: 453126.
5. Busch N, Wedemann G (2009) Modeling genomic data with type attributes, balancing stability and maintainability. *BMC Bioinf* 10: 97.
6. Teif VB (2015) Nucleosome positioning: resources and tools online. *Briefings Bioinf* 17: 745–757.
7. Bowtie: Bowtie 2: fast and sensitive read alignment. Available from: http://bowtie-bio.sourceforge.net/bowtie2/index.shtml.
8. BWA-Mapping: BWA Mapper. Available from: https://www.ridom.de/u/BWA_Mapper.html.
9. Zhao Y, Wang J, Liang F, et al. (2019) NucMap: a database of genome-wide nucleosome positioning map across species. *Nucleic Acids Res* 47: D163–D169.
10. Marti-Renom MA, Almouzni G, Bickmore WA, et al. (2018) Challenges and guidelines toward 4D nucleome data and model standards. *Nat Genet* 50: 1352.
11. Schöpflin R, Teif VB, Müller O, et al. (2013) Modeling nucleosome position distributions from experimental nucleosome positioning maps. *Bioinformatics* 29: 2380–2386.
12. Rippe K, Stehr R, Wedemann G (2012) Monte Carlo Simulations of nucleosome chains to identify factors that control DNA compaction and access. In: Schlick T, editor, *Innovations in Biomolecular Modeling and Simulations*. Cambridge: Royal Society of Chemistry, 198–235.
13. Nordenskiold L (2017) Coarse-Grained Modeling of Biomolecules. In: Papoian GA, editor, *Coarse-Grained Modeling of Biomolecules*, CRC Press, 297–340.
14. Jung J, Nishima W, Daniels M, et al. (2019) Scaling molecular dynamics beyond 100,000 processor cores for large-scale biophysical simulations. *J Comput Chem* 40: 1919–1930.
15. Perišić O, Portillo-Ledesma S, Schlick T (2019) Sensitive effect of linker histone binding mode and subtype on chromatin condensation. *Nucleic Acids Res* 47: 4948–4957.
16. Nordenskiöld L, Soman A, Korolev N, et al. (2019) Structure and Dynamics of the Telomeric Nucleosome and Chromatin. *Biophys J* 116: 71a.
17. W3C: XML Technology. Available from: https://www.w3.org/standards/xml/.
18. W3C: The Extensible Stylesheet Language Family (XSL). Available from: https://www.w3.org/Style/XSL/.
19. W3C: World Wide Web Consortium (W3C). Available from: https://www.w3.org/.
20. Group OM: About the Unified Modeling Language Specification Version 2.5.1. Available from: https://www.omg.org/spec/UML/About-UML/.

21. Kepper N, Foethke D, Stehr R, et al. (2008) Nucleosome geometry and internucleosomal interactions control the chromatin fiber conformation. *Biophys J* 95: 3692–3705.
22. Stehr R, Kepper N, Rippe K, et al. (2008) The effect of internucleosomal interaction on folding of the chromatin fiber. *Biophys J* 95: 3677–3691.
23. Lenz O (2018) The VTF Plugin is a plugin for the VMD software that reads the VTF format. Available from: https://github.com/olenz/vtfplugin/wiki.
24. Lenz O: VMD-Visual Molecular Dynamics. Available from: http://www.ks.uiuc.edu/Research/vmd/.
25. Information NCfB: Data Series GSE40896. Available from: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE40896.
26. Teif VB, Vainshtein Y, Caudron-Herger M, et al. (2012) Genome-wide nucleosome positioning during embryonic stem cell development. *Nat struct mol biol* 19: 1185–1192.
27. POV-Ray: POV-Ray-The Persistence of Vision Raytracer. Available from: http://www.povray.org/.
28. Wedemann G, Langowski J (2002) Computer simulation of the 30-nanometer chromatin fiber. *Biophy J* 82: 2847–2859.
29. ECMA-404: ECMA-404 The JSON Data Interchange Standard. Available from: https://www.json.org/.
30. Hucka M, Finney A, Sauro HM, et al. (2003) The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* 19: 524–531.
31. Bascom GD, Schlick T (2018) 5-Mesoscale Modeling of Chromatin Fibers. In: Lavelle C, Victor J-M, editors. *Nuclear Architecture and Dynamics*. Boston: Academic Press, 123–147.
32. Bascom GD, Sanbonmatsu KY, Schlick T (2016) Mesoscale modeling reveals hierarchical looping of chromatin fibers near gene regulatory elements. *J Phys Chem B* 120: 8642–8653.
33. Ehrlich L, Münckel C, Chirico G, et al. (1997) A Brownian dynamics model for the chromatin fiber. *Comput Appl Biosci* 13: 271–279.

AIMS Press