sibsim 2.1 User Documentation

Daniel Franke

sibsim 2.1 User Documentation
by Daniel Franke

User Documentation version 2.1 for sibsim Edition
Published 2003

GNU General Public License

Table of Contents

List of Examples

Preface

Getting sibsim

You can get the most up-to-date version and information about sibsim from the web page of the Institute of Medical Biometry and Statistics (IMBS): www.imbs.uni-luebeck.de/sibsim (http://www.imbs.uni-luebeck.de/sibsim) .

Request for Comments

Nobody is perfect. Please report bugs, wishes, suggestions as well as plain typographic errors, either in this manual or the program itself to <daniel.franke@imbs.uni-luebeck.de>. Thanks for your interest and help!

Acknowledgements

I would like to thank Andre Kleensang, as he did most of the testing and he never tired to explained me the simplicities of genetics, hours after hours. Thank you!

Chapter 1. Introduction

Here’s some text amiss ...

Chapter 2. Using sibsim

Running simulations using sibsim is simple and straightforward. You may either translate your available linkage files to XML input or create a new template and adjust it according to your needs. Then run

```
$> sibsim xmlfile
```

Depending on your hardware, up to 100 files per minute will be written. Memory usage is moderate, in most cases less than 10MB will be sufficient.

If sibsim terminates normally, no output will be written to `stdout`/`stderr`. Otherwise, an error description, as detailed as possible, will be written to `stderr`. Please send in your xmlfile if sibsim segfaults without a word ...

Below all currently supported options of sibsim are listed. A more detailed description to some of these may be found in the section called Basics of XML.

Available Options in sibsim 2.1

`--help`

prints a short help text to each option and exits

`--version`

prints version information and exits

`--template`

write a sibsim template file to `stdout`

`--dtd`

write sibsim document type definition to `stdout` Using this DTD you may e.g. use xmllint (part of the libxml2-package) to validate your sibsim documents.

`--dat2xml datfile`

write sibsim XML tags from linkage `datfile` to `stdout`. See also the section called From Linkage dat Format

> **Note:** This option is not implemented yet (v. 2.0.0), but coming soon!

`--pre2xml datfile`

write sibsim XML tags from linkage `prefile` to `stdout`. See also the section called From Linkage pre Format

`--xml2dat xmlfile`

write sibsim XML tags as linkage dat to `stdout`. See also the section called To Linkage dat Format

`--xml2sagedat xmlfile`

write sibsim XML tags as S.A.G.E. parameter file to `stdout`. See also the section called To S.A.G.E. dat File

`--xml2sageloc xmlfile`

write sibsim XML tags as S.A.G.E. locus description file to `stdout`. See also the section called To S.A.G.E. locus description File

`--xml2sagedat datfile`

write sibsim XML tags as S.A.G.E. genome description file to `stdout`. See also the section called To S.A.G.E. genome description File

Basics of XML

XML documents are of one specific dcument type, elements, their attributes and some other text. The following sections describe these parts subsequently.

Document Type

A valid sibsim-XML-document has to begin with the XML declaration and the sibsim root element:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<sibsim attributes>
 ...
</sibsim>
```

The first line specifies that this is a XML document and gives useful information about its encoding. The rest of the document is a text format whose structure is specified by tags between brackets. Each tag opened has to be closed! XML is pedantic about this.

A DOCTYPE definition is not required, neither the PUBLIC or SYSTEM identifiers since sibsim brings its DTD build-in.

> **Note:** Personally I would prefer a PUBLIC identifier without a specified DTD as available in SGML, but unfortunately not in XML, e.g.
>
> ```
> <?xml version="1.0" encoding="ISO-8859-1" ?>
> <DOCTYPE sibsim PUBLIC "-//Daniel Franke//DTD sibsim 2.0.0//EN" >
> <sibsim attributes>
>  ...
> </sibsim>
> ```
>
> Currently version checking of an document is impossible, so new versions of sibsim have to support old documents literally! This might be subject to change in upcoming versions of sibsim.

Elements

Elements are terms that describe a document's contents and structure. Most elements are represented by pairs of tags and mark the start and end of the construct they surround-for example, a particular pedigree begins with a <Pedigree> and ends with a </Pedigree> If an tag is empty (no content), a single tag can serve as both, the opening and the closing tag if it ends with /> instead of >. An examples for an empty tag is EqualFrequencies.

Attributes

Most elements of the sibsim doctype definition (DTD) contain attributes that determine various aspects during simulation.

An attribute to an element is given as follows:

```
 <elementname
attributename="attributevalue">
```

The "" around `attributevalue` are necessary.

Types of Attributes

`ID`

Document wide unique ID as first part of a cross reference, e.g. marker names are of type ID.

`IDREF`

Second part of a cross reference.

`Enumeration`
"`or-group`"

Group of valid values for an attribute, defined as `opt1`| `opt2`|`opt3`. The `map`-attribute of element genotype is an enumeration.

`CDATA`

Any character string of variable length. Most attributes in sibsim are of type `CDATA`. Drawback in this: an attribute as `variance` in an effect element can be set to "hi there!", it is completely valid in the sense of the document type definition. Nevertheless, an additional validation step in sibsim finds such errors.

See the reference pages of any element to see their corresponding attributes and the description of those.

Entities

Note: This section is adapted from *DocBook: The Definitive Guide*. Available from www.docbook.org (http://www.docbook.org) under the GNU Free Documentation License.

Entities are a fundamental concept in SGML and XML, and can be somewhat daunting at first. They serve a number of related, but slightly different functions, and this makes them a little bit complicated. In the most general terms, entities allow you to assign a name to some chunk of data, and use that name to refer to that data. The complexity arises because there are two different contexts in which you can use entities (in the DTD and in your documents), two types of entities (parsed and unparsed), and two or three different ways in which the entities can point to the chunk of data that they name. Entities can be divided into two broad categories, general entities and parameter entities. To sibsim, only general entities (parsed and unparsed) apply. Before you can use any type of entity, it must be formally declared. This is typically done in the document prologue.

General Entites

In use, general entities are introduced with an ampersand (&) and end with a semicolon (;). Within the category of general entities, there are two types: internal general entities and external general entities.

Internal General Entites

With internal entities, you can associate an essentially arbitrary piece of text (which may have other markup, including references to other entities) with a name. You can then include that text by referring to its name. For example, if your document frequently refers to, say, "3.14159" you might declare it as an entity:

```
<!ENTITY pi "3.14159" >
```

Then, instead of typing it out each time, you can insert it as needed in your document with the entity reference π, simply to save time. Note that this entity declaration includes another entity reference within it. That's perfectly valid as long as the reference isn't directly or indirectly recursive.

In sibsim internal general entites are of limited use.

External General Entites

With external entities, you can reference other documents from within your document. If these entities contain document text, then references to them cause the parser to insert the text of the external file directly into your document (these are called parsed entities). In this way, you can use entities to divide your single, logical document into physically distinct chunks. For example, you might break your document into three parts and store them in separate files. At the top of your document, you would include entity declarations to reference the three files:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE sibsim [
 <!ENTITY genotype  SYSTEM "human-chromosome-02.sibsim">
```

```
 <!ENTITY phenotype SYSTEM "bmi.sibsim">
 <!ENTITY pedigree  SYSTEM "nuclear-family.sibsim">
]>
```

These declarations form what is known as the internal subset. The declarations stored in the file referenced by the public or system identifier in the DOCTYPE declaration is called the external subset, which is technically optional. It is legal to put the DTD in the internal subset and to have no external subset.

> **Note:** The internal subset is parsed first in XML and, if multiple declarations for an entity occur, the first declaration is used. Declarations in the internal subset override declarations in the external subset. To have a look at the external subset, try *sibsim --dtd*.

Your document now consists simply of references to the entities defined above:

```
<sibsim& replicates="100" prefix="./">
 &genotype;
 &phenotype;
 &pedigree;
</sibsim>
```

Unparsed external entites aren't used with sibsim, we skip them here.

Parameter Entites

Parameter entities are only recognized in markup declarations (in the DTD, for example). Instead of beginning with an ampersand, they begin with a percent sign. Parameter entities are most frequently used to customize the DTD. In SGML documents they may be used to identify Marked Sections.

Parameter entites are skipped in this manual, since Marked Sections aren't available in XML documents.

Comments

Everything between "<!--" and "-->" is treated as comment and therefore ignored:

```
<!-- I'm a XML-style comment -->
```

Comments my occur anywhere after the `DOCTYPE` definition in an document, but in element definitions, therefore the following is invalid:

```
<elementname <!-- I'm a misplaced comment --> >
```

Translating To XML

For more convinience in usage of sibsim some import and export facilities, to and from XML, where added. So people who e.g. are used to linkage file format may more readily use sibsim.

Currently Supported are the `.pre` and `.dat` file formats as described from [Terwilliger,Ott, 1994].

... From Linkage pre Format

To translate a given `prefile` to XML run sibsim as follows:

```
 $> sibsim --pre2xml prefile
```

sibsim Pedigree tags will be written to `stdout`. They may be captured and written to file with:

```
 $> sibsim --pre2xml prefile
> filename
```

... From Linkage dat Format

To translate a given `datfile` to XML run sibsim as follows:

```
 $> sibsim --dat2xml datfile
```

sibsim Genotype tags will be written to `stdout`. They may be captured and written to `filename` with:

```
 $> sibsim --dat2xml datfile
> filename
```

Note: This option is not implemented yet (v. 2.0.0), but coming soon!

Translating From XML

For a more convinient use of sibsim XML, some export routines were implemented to translate the XML tags to various file formats.

Currently supported are the linkage `.dat` file format as described from [Terwilliger,Ott, 1994] as well as S.A.G.E. (http://darwin.cwru.edu/sage) parameter, locus and genome description files as described in [S.A.G.E. online doc]. See this manual also for further

On request, other export formats may be implemented. If you implement any yourself, please contribute.

... To Linkage dat Format

To translate a genotype block of a given `xmlfile` to linkage `dat` run sibsim as follows:

```
$> sibsim --xml2dat xmlfile
```

Linkage dat-file formatted output will be written to `stdout`. It may be captured and written to `filename` with:

```
$> sibsim --xml2dat xmlfile
> filename
```

... To S.A.G.E. parameter File

To translate a given `xmlfile` to S.A.G.E. parameter file format, run sibsim as follows:

```
$> sibsim --xml2sagedat xmlfile
```

Formatted output will be written to `stdout`. It may be captured and written to `filename` with:

```
$> sibsim --xml2sagedat xmlfile

> filename
```

Since S.A.G.E. programs know dozens of parameter and attributes, the output created may only be used as a skeleton for your analysis. It is not and can not be intended to be ready to use!

... To S.A.G.E. locus description File

To translate a given `xmlfile` to S.A.G.E. locus description format, run sibsim as follows:

```
$> sibsim --xml2sageloc xmlfile
```

Formatted output will be written to `stdout`. It may be captured and written to `filename` with:

```
$> sibsim --xml2sageloc xmlfile
> filename
```

This file is ready to use and it should rarely be necessary that one has to edit it.

... To S.A.G.E. genome description File

To translate a given `xmlfile` to S.A.G.E. genome description (map) format, run sibsim as follows:

```
$> sibsim --xml2sagemap xmlfile
```

Formatted output will be written to `stdout`. It may be captured and written to `filename` with:

```
$> sibsim --xml2sagemap xmlfile
> filename
```

This file is ready to use and it should rarely be necessary that one has to edit it.

Chapter 3. Step-by-Step Example

In this section, we will study the XML inputfile below in a step-by-step manner.

Example 3-1. A fully featured sibsim XML file

```
❶<?xml version="1.0" encoding="ISO-8859-1" ?>
❷<sibsim replicates="100" format="linkage" prefix="/tmp/file" seed="-1" >
 ❸<Genotype id="dinosaur-chr02" map="haldane" missingrate="0.02" missingvalue="0">
  <AllelNumbers name="mrk01" pos="0.0">
   <Frequency allel="1"> 0.25 </Frequency>
   <Frequency allel="2"> 0.25 </Frequency>
   <Frequency allel="3"> 0.25 </Frequency>
   <Frequency allel="4"> 0.25 </Frequency>
  </AllelNumbers>
  <AllelNumbers name="mrk02" pos="3.1415">
   <EqualFrequencies alleles="7" />
  </AllelNumbers>
  <AllelNumbers name="mrk03" pos="8.3749">
   <Frequency allel="1"> 0.5 </Frequency>
   <Frequency allel="2"> 0.3 </Frequency>
   <Frequency allel="3"> 0.2 </Frequency>
  </AllelNumbers>
  <Quantitative name="qtl" pos="4.21" inheritance="additive">
   <Frequency allel="1"> 0.5 </Frequency>
   <Frequency allel="2"> 0.5 </Frequency>
  </Quantitative>
 </Genotype >

 ❹<Phenotype id="bmi" missingrate="0.0" missingvalue="-99999.9">
  <FamilyEffect mean="0.0" variance="1.0" distribution="normal" />
  <GeneticEffect mean="0.0" variance="1.0" />
  <ErrorEffect mean="0.0" variance="1.0" distribution="lognormal" />
 </Phenotype >

 ❺<Pedigree id="NuclearFamily" replicates="" missingvalue="0">
  <Founder id="1" sex="male">
   <GenotypeConst locus="mrk01" paternal="missing" maternal="missing"/>
   <GenotypeConst locus="mrk03" paternal="3" maternal="random"/>
  </Founder>
  <Founder id="2" sex="female" />
  <NonFounder id="3" father="1" mother="2" sex="male" affected="true" />
  <NonFounder id="4" father="1" mother="2" sex="male" affected="false" />
 </Pedigree >
</sibsim >
```

❶ The XML header. See Header section.

❷ The root element of the document, has to be `sibsim`. See section Document Root.

❸ Block of genotype descriptions. See section Genotype Description.

❹ Optional block of phenotype description. See section Phenotype Description.

❺ Block of (multiple) pedigree descriptions. See section Pedigree Description.

Header

```
<?xml version="1.0"❶ encoding="ISO-8859-1"❷ ?>
```

❶ Version of XML used. Always set to "1.0".

❷ If you are not sure about the encoding, you may omit this attribute.

Document Root

The document root element. The root element contains any other elements of the document. Nothing but comments is allowed beyond the borders of the root element open and close tags.

```
<sibsim❶ replicates="100" format="linkage" prefix="/tmp/file" seed="-1">
 ...
</sibsim❷>
```

❶❷ Document root element, open and close tags.

See also the reference page of: sibsim

Genotype Description

```
<Genotype❶ id="dinosaur-chr02" map="haldane" missingrate="0.02" missingvalue="0">
 <AllelNumbers❷ name="mrk01" pos="0.0">
  <Frequency allel="1"> 0.25 </Frequency>
  <Frequency allel="2"> 0.25 </Frequency>
  <Frequency allel="3"> 0.25 </Frequency>
  <Frequency allel="4"> 0.25 </Frequency>
 </AllelNumbers>
 <AllelNumbers❸ name="mrk02" pos="3.1415">
  <EqualFrequencies alleles="7" />
 </AllelNumbers>
 <AllelNumbers❹ name="mrk03" pos="8.3749">
  <Frequency allel="1"> 0.5 </Frequency>
  <Frequency allel="2"> 0.3 </Frequency>
  <Frequency allel="3"> 0.2 </Frequency>
 </AllelNumbers>
 <Quantitative❺ name="qtl" pos="4.21" inheritance="additive">
  <Frequency allel="1"> 0.5 </Frequency>
  <Frequency allel="2"> 0.5 </Frequency>
 </Quantitative>
</Genotype❻>
```

❶❻ Genotype description open and close tags.

❷❸❹ Various possibilities to define a marker locus. First and third, the conservative way to define alleles and allel frequencies: value by value. As the third AllelNumbers element has to defined this way, because each marker has different frequencies, the first could have been written the same way as the second, using EqualFrequencies.

❺ Definition of a quantitative trait locus. As AllelNumbers, frequencies may be specified by using either multiple Frequency elements or one EqualFrequencies element. Due to limitations in implementation/model specification it's currently invalid to specify more (or less) than one and only one qtl. This qtl must exactly have two alleles.

Note: I hope that both of these limitations will be subject to change really soon!

See also the reference pages of: Genotype, AllelNumbers, Quantitative, Frequency, EqualFrequencies

Phenotype Description

Due to the content model, the phenotype description may be omitted.

```
<Phenotype❶ id="bmi" missingrate="0.0" missingvalue="-99999.9">
 <FamilyEffect❷ mean="0.0" variance="1.0" distribution="normal" />
 <GeneticEffect❸ mean="0.0" variance="1.0" />
 <ErrorEffect❹ mean="0.0" variance="1.0" distribution="lognormal" />
</Phenotype❺>
```

❶❺ Phenotype description open and close tags.

❷❹❸ Effects that contribute to quantitative phenotype.

See also the reference pages of: Phenotype, FamilyEffect, GeneticEffect, ErrorEffect

Pedigree Description

```
<Pedigree❶ id="NuclearFamily" replicates="" missingvalue="0">
 <Founder❷ id="1" sex="male">
  <GenotypeConst❸ locus="mrk01" paternal="missing" maternal="missing"/>
  <GenotypeConst locus="mrk03" paternal="3" maternal="random"/>
 </Founder>
 <Founder id="2" sex="female" />
 <NonFounder❹ id="3" father="1" mother="2" sex="male" affected="true" />
 <NonFounder id="4" father="1" mother="2" sex="male" affected="false" />
</Pedigree❺>
```

❶❺ Pedigree description open and close tags.

❷❹ Members of a pedigree. Founder do not have genotyped parents, but NonFounder do.

❸ Define constant genotypes for specific persons and specific loci. This element is only valid with founders. Be aware, when defining any GenotypeConst, previously defined and simulated frequencies of alleles will most likely differ!

See also the reference pages of: Pedigree, Founder, NonFounder, GenotypeConst

Runnning The Example

TOTO: Write me!

Chapter 4. sibsim Element Reference

This reference describes every element in the sibsim DTD.

In sibsim 2.1, the initial release, the following elements were introduced: AllelNumbers, EqualFrequencies, ErrorEffect, FamilyEffect, Founder, Frequency, GeneticEffect, Genotype, GenotypeConst, NonFounder, Pedigree, Phenotype, Quantitative, sibsim

Organization of Reference Pages

Synopsis

Provides a quick synopsis of the element.

Content Model or Declared Content

Describes the content model of the element in SGML/XML. See below for more information about content models.

Description

Describes the semantics of the element in detail.

Attributes

Describes the semantics of each attribute in detail.

See Also

Lists related elements.

Examples

Provides examples of proper usage for the element.

Content models are the way that document type definitions (DTDs) describe the name, number, and order of other elements that may be used inside an element.

In sibsim, there are five components to content model syntax: element names, keywords, repetitions, sequences and alternatives.

Element names

An element name in a content model indicates that an element of that type may (or must) occur at that position.

A content model of Genotype indicates that the element must contain a single Genotype definition.

Keywords

There are two keywords that occur in the content models of sibsim elements: EMPTY, and # PCDATA.

A content model that consists of the single keyword EMPTY identifes an element as an empty element. Empty elements are not allowed to have any content. In order for the word "EMPTY" to

have this special meaning, it must be the first and only word in the content model. The word "EMPTY" at any other place is treated as an element name.

The #PCDATA keyword indicates that text may occur at that position. The text may consist of entity references and any characters that are legal in the document character set.

Repetitions

An unadorned element name indicates that an element must occur exactly once at that position. A content model can also specify that an element may occur zero or more times, one or more times, or exactly zero or one time. This is accomplished by following the element name with one of the following characters: * for zero or more times, + for one or more times, or ? for exactly zero or one times.

A content model of Pedigree+ indicates that the element must contain at least one Pedigree definition and may contain many.

Sequences

If element names in a content model are separated by commas, then they must occur in sequence.

A content model of Genotype, Pedigree+, indicates that the element must contain a single Genotype definition followed by at least one Pedigree.

Alternatives

If element names in a content model are separated by vertical bars (|), then they are alternatives. These are sometimes called "or groups" because they require the selection of one or another element.

In sibsim such "or groups" are only used as attribute values.

sibsim

Name

`sibsim` — starting point of sibsim document type

Synopsis

```
The starting point of a sibsim XML document.
```

Content Model

sibsim ::= (Genotype, Phenotype?, Pedigree+)

Attributes

Name	Attribute	Default
seed	CDATA	-1
format	linkage	linkage
replicates	CDATA	*Required*
prefix	CDATA	*Required*

Description

The starting point of an sibsim XML document. Any document to be used as input with sibsim must use `sibsim` as DOCTYPE. See also the example below.

Children

The following elements occur in sibsim: Genotype, Phenotype, Pedigree

Attributes

`seed`

Initial random seed. If not defined or "-1", the current time is used. Default: "-1".

`format`

Genotype/phenotype output format. Currently supported: linkage pre-file format. Default: "linkage".

`replicates`

Number of files to simulate.

`prefix`

Output filename prefix, including relative or absolute path. Filenames are build by appending numbers from 1 to `replicates` and the format corresponding suffix, e.g. ".pre".

Example

Skeleton sibsim document definition. In this example, `/tmp/example001.pre` to `/tmp/example100.pre` would be created.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<sibsim replicates="100" format="linkage" prefix="/tmp/example">
 <!-- place genotype, phenotype and pedigree definitions here -->
</sibsim>
```

AllelNumbers

Name

`AllelNumbers` — specify a marker locus

Synopsis

```
Specify a marker locus by name and position.
```

Content Model

AllelNumbers ::= (Frequency+| EqualFrequencies)

Attributes

Name	Attribute	Default
name	ID	*Required*
position	CDATA	*Required*

Description

Specify a marker locus by name and position (in centimorgan).

Parent

These elements contain AllelNumbers: Genotype

Children

The following elements occur in AllelNumbers: Frequency, EqualFrequencies

Attributes

`name`

The name of the marker.

`position`

The position in centimorgan as distance from 0.0 .

Example

For an example usage see the example to Genotype.

EnvironmentalEffect

Name

`EnvironmentalEffect` — effect to phenotypes

Synopsis

```
Environmental Effect, an effect, shared within sibships
```

Content Model

Phenotype ::= (EMPTY)

Attributes

Name	Attribute	Default
mean	CDATA	0.0
variance	CDATA	*Required*

Name	Attribute	Default
distribution	(normallognormal)	"normal"

Description

Environmental effects are shared within sibships (contrary to the
effect within complete families/pedigrees)) Environmental effects of founders are drawn from a distribution with specified *mean* and *variance*. Effects of siblings are drawn from a distribution with mean averaged over the environmental effects of their parents and the specified *variance*.

Parent

These elements contain EnvironmentalEffect: Phenotype

Attributes

`mean`

`variance`

`distribution`

Distribution of environmental effect, either normal or lognormal, with *mean* and *variance*.

Example

For an example usage see the example to Phenotype.

EqualFrequencies

Name

`EqualFrequencies` — allel probabilities

Synopsis

```
Equally distributed probability for each allel.
```

Content Model

EqualFrequencies ::= (EMPTY)

Attributes

Name	Attribute	Default
alleles	CDATA	*Required*

Description

Equally distributed probability for each allel of the `alleles` alleles.

Parent

These elements contain EqualFrequencies: AllelNumbers, Quantitative

Attributes

`alleles`

The number of alleles of the current marker. Each allel will have the probability 1/`alleles`.

Example

For an example usage see the example to Genotype.

ErrorEffect

Name

`ErrorEffect` — effect to phenotypes

Synopsis

```
Random effect to phenotypes.
```

Content Model

ErrorEffect ::= (EMPTY)

Attributes

Name	Attribute	Default
mean	CDATA	0.0
variance	CDATA	*Required*
distribution	(normallognormal)	*Required*

Description

Random effect to phenotypes, a random number drawn from a `distribution` with `mean` and `variance`.

Parent

These elements contain ErrorEffect: Phenotype

Attributes

`mean`

`variance`

`distribution`

Distribution of error effect, either normal or lognormal, with `mean` and `variance`.

Example

For an example usage see the example to Phenotype.

FamilyEffect

Name

`FamilyEffect` — effect to phenotypes

Synopsis

```
Effect to phenotypes within families.
```

Content Model

FamilyEffect ::= (EMPTY)

Attributes

Name	Attribute	Default
mean	CDATA	0.0
variance	CDATA	*Required*
distribution	(normallognormal)	*Required*

Description

Effect to phenotypes within families. Each member of a family has the same FamilyEffect, a random number drawn from a `distribution` with `mean` and `variance`.

Parent

These elements contain FamilyEffect: Phenotype

Attributes

`mean`

`variance`

`distribution`

Distribution of family effect, either normal or lognormal, with `mean` and `variance`.

Example

For an example usage see the example to Phenotype.

Founder

Name

`Founder` — a person without known (not genotyped) parents

Synopsis

```
Specifies a person without known (not genotyped) parents
```

Content Model

Founder ::= (GenotypeConst*)

Name	Attribute	Default

Attributes

Name	Attribute	Default
id	CDATA	*Required*
sex	(male|female)	*Required*
affected	(true|false|unknown)	unknown

Description

A founder is a person without known (not genotyped) parents. The pedigree is seeded with new genotypes only by founders.

Parent

These elements contain Founder: Pedigree

Children

The following elements occur in Quantitative: GenotypeConst

Attributes

`id`

ID/name of this person, should be unique in one pedigree, may be reused in other pedigrees. Therefore `id` is of type CDATA, not ID.

`sex`

The gender of a person, either male or female.

`affected`

The affection status of a person. A person may be either affected (affected="true"), unaffected (affected="false") or unknown (affected="unknown").

Example

For an example usage see the example to Pedigree.

Frequency

Name

`Frequency` — single allel probability

Synopsis

```
Defines one specific allel by probability.
```

Content Model

Frequency ::= (#PCDATA)

Attributes

Name	Attribute	Default
allel	CDATA	*Required*

Description

Define one specific allel by probability.

Parent

These elements contain Frequency: AllelNumbers, Quantitative

Attributes

`allel`

The allel identifier, may be any character string up to 12 characters.

Example

For an example usage see the example to Genotype.

GeneticEffect

Name

`GeneticEffect` — effect to phenotypes

Synopsis

```
Genetic effect to phenotypes due to qtl genotype.
```

Content Model

GeneticEffect ::= (EMPTY)

Attributes

Name	Attribute	Default
mean	CDATA	0.0
variance	CDATA	*Required*

Description

Genetic effect to phenotypes due to qtl genotype.

Parent

These elements contain GeneticEffect: Phenotype

Attributes

`mean`

`variance`

Example

For an example usage see the example to Phenotype.

Genotype

Name

`Genotype` — a collection of marker information

Synopsis

```
A collection of marker information.
```

Content Model

Genotype ::= (AllelNumbers| Quantitative)+

Attributes

Name	Attribute	Default
id	CDATA	*Required*
map	haldane|kosambi	*Required*
missingrate	CDATA	0.0
missingvalue	CDATA	0

Description

A collection of marker information. Valid marker types are AllelNumbers and Quantitative Traits.

Parent

These elements contain Genotype: sibsim

Children

The following elements occur in Genotype: AllelNumbers, Quantitative

Attributes

`id`

Unique ID of this genotype.

`map`

The function to use to map from distances (in cM) to recombination values. Currently implemented: “haldane” and “kosambi”.

`missingrate`

Rate of randomly missing genotypes in output. Valid range: [0.0; 1.0], inclusive. Default: “0.0”.

`missingvalue`

Value to place where missing genotypes occur. This may be any character string up to 12 characters length. Default: “0”.

Example

A valid set of marker definitions: one marker at position 0.0, named “mrk”, ten equally distributed alleles. One quantitative trait locus, named qtl, two alleles, each frequency defined separatly.

```
<Genotype id="dinosaur-chr17" map="haldane">
 <AllelNumbers name="mrk" pos="0.0">
  <EqualFrequencies alleles="10" />
 </AllelNumbers>
 <Quantitative name="qtl" pos="0.0">
  <Frequency allel="1"> 0.5 </Frequency>
  <Frequency allel="2"> 0.5 </Frequency>
 </Quantitative>
</Genotype>
```

GenotypeConst

Name

`GenotypeConst` — constant genotypes definition

Synopsis

```
Constant genotype definition of a founder at a specified locus.
```

Content Model

Genotype ::= (EMPTY)

Attributes

Name	Attribute	Default
locus	IDREF	*Required*
maternal	(missing|random|CDATA)	*Required*
paternal	(missing|random|CDATA)	*Required*

Description

Constant genotype definition of a founder at a specifc founder. This includes single missing genotypes, single random genotypes or any specific allel. If undefined, both, maternal and paternal alleles, will be drawn at random.

Parent

These elements contain GenotypeConst: Founder

Attributes

`locus`

IDREF of a defined locus, either an marker or a quantitative trait locus.

`maternal`

Allelotype at specified locus, may be missing (Genotype::missingvalue will be used) or random (maternal allel will be drawn at random). Otherwise any other character string, up to 12 characters is allowed.

`paternal`

Allelotype at specified locus, may be missing (Genotype::missingvalue will be used) or random (paternal allel will be drawn at random). Otherwise any other character string, up to 12 characters is allowed.

Example

```
???
```

NonFounder

Name

`Founder` — a person with known (genotyped) parents

Synopsis

```
Specifies a person with known (genotyped) parents
```

Content Model

NonFounder ::= (EMPTY)

Attributes

Name	Attribute	Default
id	CDATA	*Required*
father	CDATA	*Required*
mother	CDATA	*Required*
sex	(male|female)	*Required*
affected	(true|false|unknown)	unknown

Description

A nonfounder is a person with known (genotyped) parents.

Parent

These elements contain NonFounder: Pedigree

Attributes

`id`

ID/name of this person, should be unique in one pedigree, may be reused in other pedigrees. Therefore `id` is of type CDATA, not ID.

`father`

A person's father ID in current pedigree. May be reused in other pedigrees. Therefore `father` is of type CDATA, not IDREF.

`mother`

A person's mother ID in current pedigree. May be reused in other pedigrees. Therefore `mother` is of type CDATA, not IDREF.

`sex`

The gender of a person, either male or female.

`affected`

The affection status of a person. A person may be either affected (affected="true"), unaffected (affected="false") or unknown (affected="unknown").

Example

For an example usage see the example to Pedigree.

Pedigree

Name

`Pedigree` — a collection of related persons

Synopsis

```
A collection of related persons.
```

Content Model

Pedigree ::= (Founder| NonFounder)+

Attributes

Name	Attribute	Default
id	ID	*Required*
replicates	CDATA	*Required*
missingvalue	CDATA	0

Description

A collection of related persons. A pedigree is seeded by founders as parents of non-founders.

Parent

These elements contain Pedigree: sibsim

Children

The following elements occur in Pedigree: Founder, NonFounder

Attributes

`id`

Unique name of this pedigree.

`replicates`

Number of times this pedigree should be simulated and output in one file.

`missingvalue`

Value to place where persons are amiss, commonly parents of founders. This may be any character string up to 12 characters length. Default: “0”.

Example

Note: Pedigree data is written to file in the exakt order as defined here.

A nuclear family, parents, two offsprings, simulated ten times:

```
<Pedigree id="nuclearfamily" replicates="10">
 <Founder id="1" sex="male">
 <Founder id="2" sex="female">
 <NonFounder id="3" father="1" mother="2" sex="male" affected="true">
```

```
 <NonFounder id="4" father="1" mother="2" sex="male" affected="false">
</Pedigree>
```

Phenotype

Name

`Phenotype` — a collection of effect types

Synopsis

```
A collection of effect types.
```

Content Model

Phenotype ::= (GeneticEffect, PolygenicEffect, (FamilyEffect | EnvironmentalEffect), ErrorEffect)

Attributes

Name	Attribute	Default
id	CDATA	*Required*
missingrate	CDATA	0.0
missingvalue	CDATA	-99999.9

Description

A collection of effect types. See model description for further information.

Parent

These elements contain Phenotype: sibsim

Children

The following elements occur in Phenotype: GeneticEffect, PolygenicEffect, FamilyEffect, EnvironmentalEffect, ErrorEffect

Attributes

`id`

Unique ID of this phenotype.

`missingrate`

Rate of randomly missing phenotypes in output. Valid range: [0.0; 1.0], inclusive. Default: “0.0”.

`missingvalue`

Value to place where phenotypes are missing. This may be any character string up to 12 characters length. Default: “-99999.9”.

Example

```
<Phenotype id="bmi">
 <GeneticEffect mean="0.0" variance="1.0" />
 <PolygenicEffect mean="0.0" variance="1.0" />
 <FamilyEffect mean="0.0" variance="1.0" distribution="normal" />
 <ErrorEffect mean="0.0" variance="1.0" distribution="lognormal" />
</Phenotype>
```

PolygenicEffect

Name

`PolygenicEffect` — effect to phenotypes

Synopsis

```
Polygenic Effect, also known as "Breeding Value"
```

Content Model

Phenotype ::= (EMPTY)

Attributes

Name	Attribute	Default
mean	CDATA	0.0
variance	CDATA	*Required*

Description

Polygenic effect to phenotypes within families. Polygenic effects of founders are drawn from a *distribution* with *mean* and *variance*. Effects of NonFounders are drawn from a *distribution* with mean averaged from the polygenic effects of the parents and *variance*.

Parent

These elements contain PolygenicEffect: Phenotype

Attributes

`mean`

`variance`

`distribution`

Distribution of polygenic effect, either normal or lognormal, with *mean* and *variance*.

Example

For an example usage see the example to Phenotype.

Quantitative

Name

`Quantitative` — specify a quantitative trait locus

Synopsis

```
Specify a quantitative trait locus by name and position.
```

Content Model

Quantitative ::= (Frequency+| EqualFrequencies)

Attributes

Name	Attribute	Default
name	ID	*Required*
position	CDATA	*Required*
inheritance	(dominant|recessive|additive)	*Required*

Description

Specify a quantitative trait locus by name and position (in centimorgan).

Parent

These elements contain Quantitative: Genotype

Children

The following elements occur in Quantitative: Frequency, EqualFrequencies

Attributes

`name`

The name of the quantitative trait locus.

`position`

The position in centimorgan as distance from 0.0 .

`inheritance`

Specifies the inheritance model of this qtl. `Inheritance` affects the calculation of phenotypes if enabled.

Example

For an example usage see the example to Genotype.

Appendix A. Installation of sibsim

Requirements

Since sibsim uses XML as input format, you will have to have *libxml2* installed. It is available for free from xmlsoft.org (http://xmlsoft.org) and is most likely already installed. If you are using a RPM distribution as SuSE or Redhat, please make sure you also have the *libxml2-devel* package installed. In development versions 2.5.5, 2.5.7 and 2.5.10 where used and testet. If you have older versions of libxml2 installed, please let me know if sibsim works for you with this version. Nevertheless, you are encouraged to update your installed copy of libxml2 to the latest one available, since libxml2 is still developed further and various bugs are fixed from version to version.

No more prerequirements are currently known.

Basic Installation

The ‘configure’ shell script attempts to guess correct values for various system-dependent variables used during compilation. It uses those values to create a ‘Makefile’ in each directory of sibsim. Finally, it creates a shell script ‘config.status’ that you can run in the future to recreate the current configuration, a file ‘config.cache’ that saves the results of its tests to speed up reconfiguring, and a file ‘config.log’ containing compiler output (useful mainly for debugging ‘configure’).

If you need to do unusual things to compile the package, please try to figure out how ‘configure’ could check whether to do them, and mail diffs or instructions to the address given in the ‘README’ so they can be considered for the next release. If at some point ‘config.cache’ contains results you don’t want to keep, you may remove or edit it.

The file ‘configure.in’ is used to create ‘configure’ by a program called ‘autoconf’. You only need ‘configure.in’ if you want to change it or regenerate ‘configure’ using a newer version of ‘autoconf’.

The simplest way to compile this package is:

1. ‘cd’ to the directory containing the package’s source code and type ‘./configure’ to configure the package for your system. If you’re using ‘csh’ on an old version of System V, you might need to type ‘sh ./configure’ instead to prevent ‘csh’ from trying to execute ‘configure’ itself.

   Running ‘configure’ takes a while. While running, it prints some messages telling which features it is checking for.

2. Type ‘make’ to compile the package.
3. Type ‘make install’ to install the programs and any data files and documentation.
4. You can remove the program binaries and object files from the source code directory by typing ‘make clean’.

Compilers and Options

Some systems require unusual options for compilation or linking that the ‘configure’ script does not know about. You can give ‘configure’ initial values for variables by setting them in the environment. Using a Bourne-compatible shell, you can do that on the command line like this:

```
CC=c89 CFLAGS=-O2 LIBS=-lposix ./configure
```

Or on systems that have the ‘env’ program, you can do it like this:

```
env CPPFLAGS=-I/usr/local/include LDFLAGS=-s ./configure
```

Compiling For Multiple Architectures

You can compile the package for more than one kind of computer at the same time, by placing the object files for each architecture in their own directory. To do this, you must use a version of ‘make’ that supports the ‘VPATH’ variable, such as GNU ‘make’. ‘cd’ to the directory where you want the object files and executables to go and run the ‘configure’ script. ‘configure’ automatically checks for the source code in the directory that ‘configure’ is in and in ‘..’.

If you have to use a ‘make’ that does not supports the ‘VPATH’ variable, you have to compile the package for one architecture at a time in the source code directory. After you have installed the package for one architecture, use ‘make distclean’ before reconfiguring for another architecture.

Installation Names

By default, **`‘make install’`** will install the package’s files in ‘/usr/local/bin’, ‘/usr/local/man’, etc. You can specify an installation prefix other than ‘/usr/local’ by giving ‘configure’ the option ‘--prefix=PATH’.

You can specify separate installation prefixes for architecture-specific files and architecture-independent files. If you give ‘configure’ the option ‘--exec-prefix=PATH’, the package will use PATH as the prefix for installing programs and libraries. Documentation and other data files will still use the regular prefix.

You can cause programs to be installed with an extra prefix or suffix on their names by giving ‘configure’ the option ‘--program-prefix=PREFIX’ or ‘--program-suffix=SUFFIX’.

Operation Controls

configure recognizes the following options to control how it operates.

`--cache-file=FILE`

‘ Use and save the results of the tests in FILE instead of ‘./config.cache’. Set FILE to ‘/dev/null’ to disable caching, for debugging ‘configure’.

`--help`

‘ Print a summary of the options to ‘configure’, and exit.

`--quiet`
`--silent`
`--q`

‘ Do not print messages saying which checks are being made.

`--srcdir=DIR`

Look for the package’s source code in directory DIR. Usually ‘configure’ can determine that directory automatically.

`--version`

‘ Print the version of Autoconf used to generate the ‘configure’ script, and exit.

configure also accepts some other, not widely useful, options.

Bibliography

Web Ressources

[autoconf] *www.gnu.org (http://www.gnu.org/manual/autoconf): GNU autoconf documentation*.

[automake] *www.gnu.org (http://www.gnu.org/manual/automake): GNU automake documentation*.

[docbook] *www.docbook.org (http://www.docbook.org): DocBook DTD and documentation*.

[doxygen] *www.doxygen.org (http://www.doxygen.org): Generating C/C++ documentation from source files*.

[libxml2] *www.xmlsoft.org (http://www.xmlsoft.org): XML parsing and validating*.

[S.A.G.E. online doc] *darwin.cwru.edu/docs/sage (http://darwin.cwru.edu/docs/sage): Statistical Analysis for Genetic Epidimiology (S.A.G.E.) User Manual*.

[STL] *www.sgi.com (http://www.sgi.com): Standard Template Library Programmer's Guide*.

Books

[Press et. al., 1999] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, 0-521-43108-5, Cambridge University Press, *Numerical Recipes in C*, 2nd. Edition.

[Ott, 1999] Jurg Ott, 0-8018-6140-3, The Johns Hopkins University Press, *Analysis Of Human Genetic Linkage*, 3rd. Edition.

[Terwilliger, Ott, 1994] Joseph Douglas Terwilliger and Jurg Ott, 0-8018-4803, The Johns Hopkins University Press, *Handbook Of Human Genetic Linkage*.