



# Moa

0.1

author: Mark Fiers

date: Tue Aug 18 14:40:57 NZST 2009

git: master 31f8365093eb2e69a62e159aa8a28f89417ec4b4

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Installation</b>	<b>5</b>
2.1	Prerequisites . . . . .	5
2.2	Getting the code . . . . .	5
2.3	Configuration . . . . .	6
<b>3</b>	<b>Using Moa</b>	<b>7</b>
3.1	Creating a pipeline . . . . .	7
3.2	Running a pipeline . . . . .	8
<b>4</b>	<b>Couchdb</b>	<b>9</b>
<b>5</b>	<b>Creating new templates</b>	<b>11</b>
5.1	Definition . . . . .	11
5.2	Implementation . . . . .	11
<b>6</b>	<b>Appendix A: Template reference</b>	<b>12</b>
6.1	blast . . . . .	12
6.2	blastSingle . . . . .	13
6.3	blastdb . . . . .	14
6.4	blat . . . . .	15
6.5	bowtie . . . . .	16
6.6	bowtiedb . . . . .	16
6.7	cleanFasta . . . . .	17
6.8	clustalgroup . . . . .	17
6.9	clustalpair . . . . .	18
6.10	clustalw . . . . .	19
6.11	concatenate . . . . .	19
6.12	create.gbrowse.db . . . . .	20
6.13	dottup . . . . .	20
6.14	empty . . . . .	21
6.15	fasta2gff . . . . .	21

6.16	gap4export . . . . .	22
6.17	gather . . . . .	22
6.18	getFromNcbi . . . . .	23
6.19	gmap . . . . .	24
6.20	gmapdb . . . . .	24
6.21	lftp . . . . .	25
6.22	mummer . . . . .	26
6.23	nstretch . . . . .	26
6.24	pregap . . . . .	27
6.25	traverse . . . . .	28
6.26	upload2gbrowse . . . . .	28
6.27	varscan . . . . .	29

<b>Bibliography</b>	<b>29</b>
---------------------	-----------

# Chapter 1

## Introduction

Moa is a piece of software build around GNU Make<sup>7</sup> that allows you to use Gnu Make run bioinformatics pipelines.

GNU Make is an excellent tool to automate the compilation of software. Gnu make determines how a file is created, what it's dependencies are, and what needs to be executed. Gnu Make uses so called Makefiles to describe a project. A bioinformatics project is often of the same form as compiling software.

Moa wraps a set of common bioinformatics tools as Makefiles. Features of Moa are:

- A uniform interface; all Moa makefiles use a central library that provides a uniform, command line, interface to configuring and executing jobs.
- Interaction; all templates are designed to interact with each other.
- Parallel execution; Gnu make bla bla bla

Moa consists of several parts:

- moaBase; a central library describing a number of central routines used by all Makefiles
- Template Makefiles; each application is wrapped in a template Makefile.
- The “moa” helper script; a number of tools that cannot be caught in Makefiles are implemented in a central helper script called “moa”.
- Additional helper scripts; a number of diverse utilities are part of the moa packages. These are a part of an embedded application.
- Couchdb interface; Moa is able to store information on each job in a couchdb. See chapter XX.

### 1.0.1 Example session

To better understand how Moa works, please read this sample session:

```
mkdir test  
cd test  
moa new lftp
```

## Chapter 2

# Installation

### 2.1 Prerequisites

Moa is developed on Ubuntu<sup>10</sup> and RHEL<sup>9</sup> Linux and is expected to operate without much problems on most modern Linux distributions. Moa is depends on the following list of software. The version numbers are an indication, not strict prerequisites. Other, even older, versions might work.

- Gnu Make 3.81<sup>7</sup>
- Git 1.6<sup>6</sup>. To download the Moa software. Alternatively it is possible to download a tarball.
- Python 2.6<sup>8</sup>. Python 2.5 will not work, several supporting scripts use 2.6 specific functionality
- Biopython 1.49<sup>3</sup>. Only used by the blast warpper.
- Apache Couchdb 0.9.0<sup>2</sup> (Only when using couchdb functionality)
- Couchdb-python<sup>4</sup>

Furthermore, the required bioinformatics analysis tools need to be installed. All Moa templates that wrap an application expect that application to be installed and present in the PATH.

### 2.2 Getting the code

Moa is hosted at github:

<http://github.com/mfiers/Moa>

Currently there are no formal releases so the only option is to download the latest version of the software, this can be done using git<sup>6</sup>:

```
git clone git://github.com/mfiers/Moa.git
```

It is also possible to download an (automatically generated) archive of the trunk. Using Git, however, makes it very easy to stay in sync with the latest bugfixes and is thus strongly recommended until there are formal releases. An archive can be found here:

```
http://github.com/mfiers/Moa/tarball/master
```

After downloading, and possibly unpacking, the source code must be moved to a suitable location of your choice. For example /opt/moa. The resulting tree should contain the following directories: /opt/moa/bin and /opt/moa/template. Remember to set the file attributes, depending on who is going to use the software.

## 2.3 Configuration

Configuration of Moa is simple: The Moa /bin/ directory must be included in the PATH and a environment variable must be set pointing to the Moa directory. The easiest way to do this is by adding the following lines to your .bashrc:

```
export PATH=/opt/moa/bin:$PATH
export MOABASE=/opt/moa
```

and run `source .bashrc`.

..done..

## Chapter 3

# Using Moa

### 3.1 Creating a pipeline

#### 3.1.1 Guiding principles

Most (bioinformatics?) projects start small, and grow over time. From that perspective it is advisable to give the organization of your project some thought on forehand.

When using Moa the separate analysis steps of a pipeline each reside in a directory. The output data of each analysis usually resides in the same directory or a subdirectory thereof. Moa has templates that assist in downloading and organizing data. This has as result that all project data in a Moa project will be organized in a directory tree on your filesystem. Such a tree must represent both the data in logical way as well as the analysis pipeline organization.

Although there are likely multiple ways of achieving a healthy organization of a Moa project, this manual proposes the following organization:

- On the highest levels organize your project according to fundamental divisions in the project or data source. For example, if you work with data from multiple organisms, that might be a good top level division.
- On lower levels start organizing your annotation pipeline. Since most



### **3.1.2 Setting up analysis steps**

## **3.2 Running a pipeline**

### **3.2.1 Running one job**

### **3.2.2 Running a series of jobs**

## Chapter 4

# Couchdb

Couchdb<sup>2</sup> is a novel type of database that is almost completely unlike a SQL database. In its simplest form it is a high performance key-value datastore. Moa uses Couchdb to store information on all analyses performed by Moa. This means that for each job that Moa performs, a record is created in the Couchdb database. This record has a unique identifier, called `jid`. Moa creates `jids` on the fly by combining the template name, the directory name and a unique identifier (to prevent collisions). These names are not always very descriptive, so it is advisable to set a `jid` manually. This is possible using the following command (do not use spaces!):

```
make set jid=SensibleName
```

Each time Moa executes, the analysis record in Couchdb is updated. The record contains all parameters used, the type of analysis done and the location (current directory) of the analysis. It is possible to update the couchdb record without running the analysis using:

```
make register
```

Moa/Couchdb records are a set of key/value pairs, that look like this:

The most important application of Couchdb in Moa is to refer to other jobs using Couchdb identifiers. In a Moa project without couchdb references to the output of other jobs is done by defining the path to that analysis. If, at a certain moment, the project structure needs to be rearranged, it can be hard to discover which path references need to be updated. Use of couchdb solves this, instead of referring to a path, it is now possible to refer to a `jid` / value combination.

allows a user to refer to another Moa job by the identifier, as opposed to using (relative) directories. The biggest advantage is that it is now possible to shuffle your directories around without breaking the pipeline structure.

### 4.0.3 Configuration

Please follow the couchdb documentation to set up a local server. Moa has been developed with the latest version of Couchdb (currently 0.9.1). It might be possible to use an older version, but that has not been tested.

All Moa configuration for couchdb is done in `$MOABASE/etc/moa.conf.mk`.

The default setting of Moa is to not use couchdb. This can be overridden by setting: `usecouchdb=T`

Moa expects a Couchdb server on `localhost:5984`. This can be overridden using: `couchserver=other.server:portnumber`

All information

### 4.0.4 Using couchdb with Moa

Instead of using `make set key=value`, couchdb variables are set using `make cset jid\^{}key`

LocalWords: jid SensibleName

## **Chapter 5**

# **Creating new templates**

This chapter describes how to create new templates for use with Moa. Creating a template is not very difficult, the hardest part is ensuring that templates are able to interact with each other.

Each template exists of two main parts; definition and implementation.

### **5.1 Definition**

### **5.2 Implementation**

## Chapter 6

# Appendix A: Template reference

This chapter contains, as a reference, all help documentation of all templates currently in the Moa repository. It is possible to get the (latest) version of the help for each template by running:

```
make help
```

in a directory with a Moa analysis. Moreover, if an Makefile links to multiple templates, on the fly generated help will detail all targets that can be used and all parameters that can be defined.

### 6.1 blast

Wraps BLAST<sup>1</sup>, the most popular similarity search tool in bioinformatics

#### 6.1.1 Targets

**blast** Running BLAST takes an input directory (*blast\_input\_dir*), determines what sequence files are present (with the parameter *blast\_input\_extension*) and executes BLAST on each of these. Moa BLAST is configured to create XML output (as opposed to the standard text based output) in the *./out* directory. The output XML is subsequently converted to GFF3<sup>5</sup> by the custom *blast2gff* script (build around biopython<sup>3</sup>). Additionally, a simple text report is created.

**blast\_report** Generate a text BLAST report.

## 6.1.2 Parameters

### Required parameters

**blast.db** Location of the blast database

**blast.gff\_source** source field to use in the gff

### Optional parameters

**blast.input\_dir** directory containing the input sequences

**input\_extension** Extension of the input files

**blast.program** blast program to use (default: blastn)

**blast.eval** e value cutoff

**blast.nohits** number of hits to report

**blast.threads** threads to run blast with (note the overlap with the Make -j parameter)

**jid** Unique identifier for this analysis job. Autogenerated unless defined.

**project** • undefined

## 6.2 blastSingle

Wraps BLAST<sup>1</sup>, the most popular similarity search tool in bioinformatics

### 6.2.1 Targets

**blast** Running BLAST takes an input directory (*blast.input\_dir*), determines what sequence files are present (with the parameter *blast.input\_extension*) and executes BLAST on each of these. Moa BLAST is configured to create XML output (as opposed to the standard text based output) in the *./out* directory. The output XML is subsequently converted to GFF3<sup>5</sup> by the custom *blast2gff* script (built around biopython<sup>3</sup>). Additionally, a simple text report is created.

**blast.report** Generate a text BLAST report.

## 6.2.2 Parameters

### Required parameters

**blast\_input\_file** Input fasta file to BLAST

**blast\_db** Location of the blast database

**blast\_gff\_source** source field to use in the gff

### Optional parameters

**blast\_input\_dir** directory containing the input sequences

**input\_extension** Extension of the input files

**blast\_program** blast program to use (default: blastn)

**blast\_eval** e value cutoff

**blast\_nohits** number of hits to report

**blast\_nothreads** threads to run blast with (note the overlap with the Make -j parameter)

**jid** Unique identifier for this analysis job. Autogenerated unless defined.

**project** • undefined

## 6.3 blastdb

### 6.3.1 Targets

blastdb :

### 6.3.2 Parameters

#### Required parameters

**bdb\_name** Database name to create

### Optional parameters

**bdb\_input\_dir** Dir with the input fasta files, defaults to ./fasta

**bdb\_input\_extension** extension of the input sequence files, defaults to fasta

**bdb\_protein** Protein database? (T)rue) or not (F)alse (default: F)

**jid** Unique identifier for this analysis job. Autogenerated unless defined.

**project** • undefined

## 6.4 blat

### 6.4.1 Targets

blat :

### 6.4.2 Parameters

#### Required parameters

**blat\_db** Blat db file (multifasta)

**blat\_gff\_source** • undefined

#### Optional parameters

**blat\_input\_file** input query file. If this variable is not defined, the combination of blat\_input\_dir and blat\_input\_extension is used to find a list of input files

**blat\_input\_dir** source field in the generated gff

**blat\_input\_extension** extension of the input files

**blat\_eval** evaluate cutoff to select the reported hits on (defaults to 1e-15)

**blat\_db\_id\_list** a sorted list of db ids and descriptions, enhances the report generated

**blat\_db\_type** type of the database (dna, prot or dnax)

**blat\_query\_type** type of the query (dna, rna, prot, dnax or rnax)

**jid** Unique identifier for this analysis job. Autogenerated unless defined.



**project**     • undefined

## 6.5 bowtie

### 6.5.1 Targets

bowtie :

### 6.5.2 Parameters

#### Required parameters

**bowtie\_db** Bowtie db

**bowtie\_input\_dir** input dir with the query files

#### Optional parameters

**bowtie\_input\_extension** Extension of the input files, defaults to fastq

**bowtie\_input\_format** Format of the input files, defaults to fastq

**bowtie\_extra\_params** extra parameters to feed bowtie

**bowtie\_output\_name**     • undefined

**jid** Unique identifier for this analysis job. Autogenerated unless defined.

**project**     • undefined

## 6.6 bowtiedb

### 6.6.1 Targets

bowtiedb :

### 6.6.2 Parameters

#### Required parameters

**bowtiedb\_input\_dir** The reference sequence to build a bowtie database with.

**bowtiedb\_name** Name of the bowtie index to create

### Optional parameters

**bowtiedb\_input\_extension** Extension of the input files, defaults to 'fasta'

**jid** Unique identifier for this analysis job. Autogenerated unless defined.

**project** • undefined

## 6.7 cleanFasta

### 6.7.1 Targets

clean\_fasta :

### 6.7.2 Parameters

#### Required parameters

#### Optional parameters

**cf\_input\_dir** • undefined

**cf\_input\_extension** • undefined

**sed\_command** The sed command cleaning the code, defaults to '/>/!s/[ACGTNacgtn]/N/g'

**jid** Unique identifier for this analysis job. Autogenerated unless defined.

**project** • undefined

## 6.8 clustalgroup

### 6.8.1 Targets

clustalgroup :

## 6.8.2 Parameters

### Required parameters

**cwg\_input\_dir** This set of sequences to run clustalw on

### Optional parameters

**cwg\_input\_extension** • undefined

**jid** Unique identifier for this analysis job. Autogenerated unless defined.

**project** • undefined

## 6.9 clustalpair

### 6.9.1 Targets

clustalpair :

### 6.9.2 Parameters

#### Required parameters

**input\_dir\_a** This set is compared to the sequences in input\_dir\_b. only a forward comparison is made (a against b, not the other way round )

**input\_dir\_b** The set to compare against

#### Optional parameters

**input\_extension** Extension of the input files

**jid** Unique identifier for this analysis job. Autogenerated unless defined.

**project** • undefined

## 6.10 clustalw

### 6.10.1 Targets

clustalw :

### 6.10.2 Parameters

#### Required parameters

**input\_dir\_a** This set is compared to the sequences in input\_dir\_b. only a forward comparison is made (a against b, not the other way round )

**input\_dir\_b** The set to compare against

#### Optional parameters

**input\_extension** Extension of the input files

**jid** Unique identifier for this analysis job. Autogenerated unless defined.

**project** • undefined

## 6.11 concatenate

### 6.11.1 Targets

concatenate :

### 6.11.2 Parameters

#### Required parameters

**input\_dir** Directory with the input data

**name** A unique project name defining this job. Cannot have spaces.

### Optional parameters

**input\_extension** Extension of the input files

**jid** Unique identifier for this analysis job. Autogenerated unless defined.

**project**     • undefined

## 6.12 create.gbrowse.db

### 6.12.1 Targets

create.gbrowse.db :

### 6.12.2 Parameters

#### Required parameters

**gbrowse\_user** gbrowse db user

**gbrowse\_db** gbrowse db

### Optional parameters

**jid** Unique identifier for this analysis job. Autogenerated unless defined.

**project**     • undefined

## 6.13 dottup

### 6.13.1 Targets

dottup :

### 6.13.2 Parameters

#### Required parameters

**dottup\_input\_dir\_a** This set is compared to the sequences in input\_dir\_b. only a forward comparison is made (a against b, not the other way round )

**dottup\_input\_dir\_b** The set to compare against

#### Optional parameters

**dottup\_input\_extension** • undefined

**dottup\_wordsize** • undefined

**jid** Unique identifier for this analysis job. Autogenerated unless defined.

**project** • undefined

## 6.14 empty

### 6.14.1 Targets

empty :

### 6.14.2 Parameters

#### Required parameters

#### Optional parameters

**jid** Unique identifier for this analysis job. Autogenerated unless defined.

**project** • undefined

## 6.15 fasta2gff

### 6.15.1 Targets

fasta2gff :

### 6.15.2 Parameters

#### Required parameters

**f2g\_gffsource** Source to be used in the gff

### Optional parameters

**f2g\_input\_dir** Directory with the input fasta (default: ./fasta)

**f2g\_output\_dir** Directory with the output gff (default: ./gff)

**f2g\_input\_extension** glob pattern of the fasta files (default: \*.fasta)

**f2g\_options** options to be passed to the fasta2gff script

**jid** Unique identifier for this analysis job. Autogenerated unless defined.

**project**     • undefined

## 6.16 gap4export

### 6.16.1 Targets

gap4export :

### 6.16.2 Parameters

#### Required parameters

**ge\_input\_dir** Directory with the input data

**ge\_input\_pattern** file name pattern

#### Optional parameters

**jid** Unique identifier for this analysis job. Autogenerated unless defined.

**project**     • undefined

## 6.17 gather

### 6.17.1 Targets

gather :

## 6.17.2 Parameters

### Required parameters

**g\_input\_dir** list of directories with the input files

**g\_input\_pattern** glob pattern to download

### Optional parameters

**g\_name\_sed** • undefined

**g\_output\_dir** Output subdirectory, defaults to '.'

**g\_process** Command to process the files. If undefined, hardlink the files.

**g\_limit** limit the number of files gathered (with the most recent files first, defaults to 1mln)

**g\_powerclean** Do brute force cleaning (T/F). Remove all files, except moa.mk & Makefile when calling make clean. Defaults to F.

**jid** Unique identifier for this analysis job. Autogenerated unless defined.

**project** • undefined

## 6.18 getFromNcbi

### 6.18.1 Targets

getFromNcbi :

### 6.18.2 Parameters

#### Required parameters

**ncbi\_db** NCBI database (for example nucest)

**ncbi\_query** NCBI query (for example txid9397[Organism%3Aexp])

#### Optional parameters

**jid** Unique identifier for this analysis job. Autogenerated unless defined.



**project**     • undefined

## 6.19 gmap

### 6.19.1 Targets

gmap :

### 6.19.2 Parameters

#### Required parameters

**gmap\_db** Gmap db

**gmap\_input\_file** input file with the sequences to map

#### Optional parameters

**gmap\_extra\_parameters** extra parameters to feed to gmap

**gmap\_invert\_gff** Invert the GFF (T/F)

**jid** Unique identifier for this analysis job. Autogenerated unless defined.

**project**     • undefined

## 6.20 gmapdb

### 6.20.1 Targets

gmapdb :

### 6.20.2 Parameters

#### Required parameters

**gmapdb\_input\_dir** The reference sequence to build a gmap database with.

**gmapdb\_name** Name of the gmap index to create

### Optional parameters

**gmapdb\_input\_extension** Extension of the input files, defaults to 'fasta'

**jid** Unique identifier for this analysis job. Autogenerated unless defined.

**project** • undefined

## 6.21 lftp

### 6.21.1 Targets

lftp :

### 6.21.2 Parameters

#### Required parameters

**lftp\_url** The base url to download from

**lftp\_pattern** glob pattern to download

#### Optional parameters

**lftp\_timestamp** Depend on lftp to decide if a file needs updating, else a touch-file is created that you need to delete or touch before updating (T/F)

**lftp\_powerclean** Do brute force cleaning (T/F). Remove all files, except moa.mk & Makefile when calling make clean. Defaults to F.

**lftp\_noclean** set of files not to be deleted by the powerclean

**lftp\_user** username for the remote site

**lftp\_pass** password for the remote site, note that this can be defined on the commandline using: 'make lftp\_pass=PASSWORD'

**lftp\_output\_dir** subdir to create & write all output to. If not defined, data will be downloaded to directory containing the Makefile

**lftp\_dos2unix** (T/F) Run dos2unix to prevent problems with possible dos text files (default=F).

**lftp\_mode** Mode of operation - mirror or get. Mirror enables timestamping. Get just gets a single file. If using get, consider setting depend\_lftp\_timestamp

to F. When using get, the full url should be in lftp\_url. lftp\_pattern is ignored. Defaults to mirror.

**jid** Unique identifier for this analysis job. Autogenerated unless defined.

**project**     • undefined

## 6.22 mummer

### 6.22.1 Targets

mummer :

### 6.22.2 Parameters

#### Required parameters

**input\_dir\_a** This set is compared to the sequences in input\_dir\_b. only a forward comparison is made (a against b, not the other way round )

**input\_dir\_b** The set to compare against

#### Optional parameters

**input\_extension** Extension of the input files

**jid** Unique identifier for this analysis job. Autogenerated unless defined.

**project**     • undefined

## 6.23 nstretch

### 6.23.1 Targets

nstretch :

## 6.23.2 Parameters

### Required parameters

### Optional parameters

**nstretch\_input\_dir** input dir with the fasta files

**nstretch\_input\_extension** extension of the input files

**nstretch\_len** minimal number of Ns before its reported (default 10)

**jid** Unique identifier for this analysis job. Autogenerated unless defined.

**project**     • undefined

## 6.24 pregap

### 6.24.1 Targets

pregap :

### 6.24.2 Parameters

#### Required parameters

**input\_dir** Directory with the input data

**input\_pattern** file name pattern

**cloning\_vector** File containing the cloning vector

**sequencing\_vector** File containing the sequencing vector

**ecoli\_screenseq** File containing ecoli screen sequences

**repeat\_masker\_lib** File with a repeatmasker library

**vector\_primerfile** File with the vector primers

#### Optional parameters

**quality\_value\_clip** quality cutoff (default=10)

**pregap\_template** the template pregap config file to use. if not defined, Moa tries ./files/pregap.config.

**jid** Unique identifier for this analysis job. Autogenerated unless defined.

**project** • undefined

## 6.25 **traverse**

### 6.25.1 **Targets**

traverse :

### 6.25.2 **Parameters**

**Required parameters**

**Optional parameters**

**jid** Unique identifier for this analysis job. Autogenerated unless defined.

**project** • undefined

## 6.26 **upload2gbrowse**

### 6.26.1 **Targets**

upload2gbrowse :

### 6.26.2 **Parameters**

**Required parameters**

**gup\_user** gbrowse db user. If not defined, this defaults to 'moa'.

**gup\_db** gbrowse database. If not defined, this defaults to 'moa'.

**gup\_gffsource** the gff source field, used in batch operations

**Optional parameters**

**gup\_fasta\_dir** input directory with fasta files to upload to gbrowse

**gup\_gff\_dir** input directory with gff files to upload to gbrowse

**gffsource** gff source of the data to upload  
**gup\_upload\_fasta** Perform fasta upload (T/F)  
**gup\_upload\_gff** Perform gff upload (T/F)  
**gbrowse\_do\_upload** Deprecated: use gup\_upload\_gff or gup\_upload\_fasta  
**jid** Unique identifier for this analysis job. Autogenerated unless defined.  
**project** • undefined

## 6.27 varscan

### 6.27.1 Targets

varscan :

### 6.27.2 Parameters

#### Required parameters

**varscan\_input\_file** Varscan input alignments file

#### Optional parameters

**varscan\_extra\_params** location of varscan.pl, defaults to '/usr/lib/perl5/site\_perl/5.8.8/varscan.pl'

**varscan\_output\_name** Base name of the output files

**varscan\_perl\_file** • undefined

**jid** Unique identifier for this analysis job. Autogenerated unless defined.

**project** • undefined

# Bibliography

- [1] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J Mol Biol*, 215(3):403–410, Oct 1990.
- [2] Apache couchdb. <http://couchdb.apache.org/>.
- [3] Biopython. <http://biopython.org/>.
- [4] Couchdb python. <http://code.google.com/p/couchdb-python/>.
- [5] Generic Feature Format (v3). <http://song.sourceforge.net/gff3.shtml>.
- [6] Git. <http://git-scm.com/>.
- [7] Gnu Make. <http://www.gnu.org/software/make/>.
- [8] Python. <http://python.org>.
- [9] Red Hat Enterprise Linux. <http://www.redhat.com/rhel/>.
- [10] Ubuntu. <http://www.ubuntu.com/>.