THE NEUBIRO SOFTWARE

stefano gualdi

stefano.gualdi@gmail.com

Nicosia, CY - 21/23 september 2017

ABOUT ME

software developer
worked on EUBIROD/BiroBox
developer of NeuBiro

stefano.gualdi@gmail.com

@stefanogualdi

stefano.gualdi

AGENDA

- what is NeuBiro?
- how it works
- how to configure
- how to install

THE ORIGIN

The initial data collection used the Biro/Birobox software, but:

- it was complicated to install
- it needed a virtual machine to run the software and all the required dependencies
- all the logic was hardcoded into the core of the sofware

and we learned from experience!

WHAT IS NEUBIRO

The NeuBiro’s goal is to process an input data stream, run a sequence of statistical operation an eventually produce a structured report.

NeuBiro is the evolution of a software developed under the **MATRICE** project of **AGE.NA.S.** *(AGEnzia NAzionale per i Servizi sanitari regionali)* that used the BIRO approach.

NeuBiro improves the original code base in several aspects.

KEY ASPECTS

- fast
- modular
- have flexible configuration
- have few external dependencies (only R-stat)
- developed with modern languages and techniques
- multiplatform (runs on the JVM)

TECHNOLOGY STACK

Groovy

GRIFFON

H2

DocBook

The Source for Documentation

HOW IT WORKS

NeuBiro is not bound to a specific field of analysis.

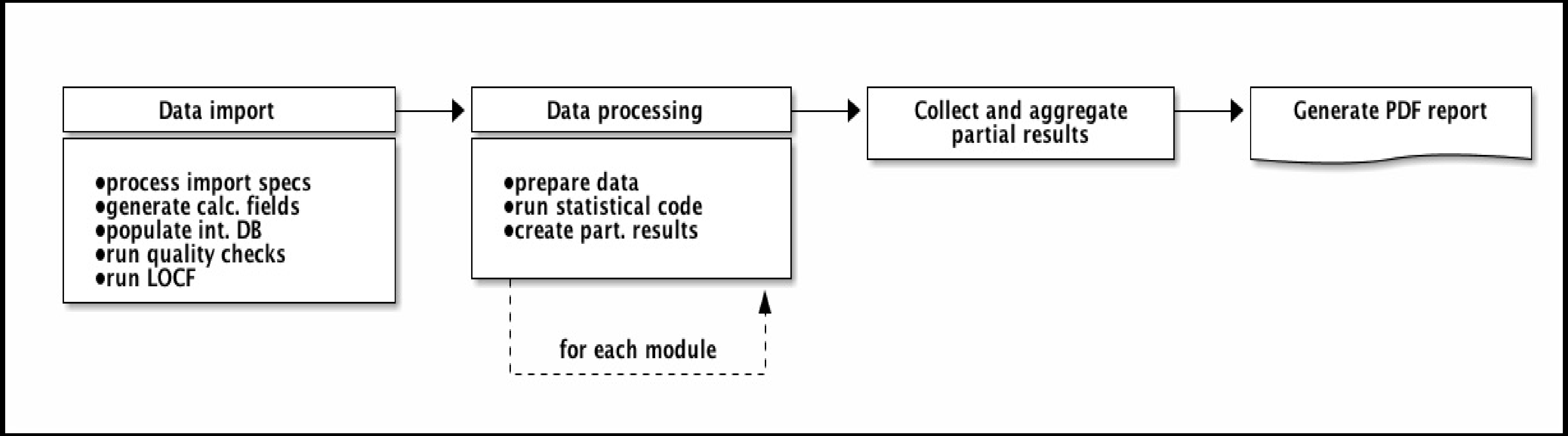
Every aspect of the software can be defined with configuration modules.

COMPONENTS

The main components that implements the analysis are:

- **Import specification**
  a config file which describes the input data stream (master table, lookup tables, calculated fields, etc.)
- **Statistical package**
  the bundle of all the statistical modules (specs and implementation) and the report template
- **Statistical module** (AKA indicator)
  the single indicator composed by the specs and the R code

WORKFLOW



overview of the main workflow

PROCESSING STEPS

- `data import`: data are imported and preprocessed, ready to be used for the next steps
- `data processing`: data are processed executing each selected statistical module. Each module creates partial data
- `report generation`: partial data created in the data processing phase are collected and merged together to generate the PDF report

DATA IMPORT

To optimise the analysis the input data are stored in an internal database.

The data import process read the input stream *(in CSV format)* and populate the internal storage.

At the end of the import step the internal database is populated with the data and NeuBiro is ready to start the analysis.

DATA IMPORT

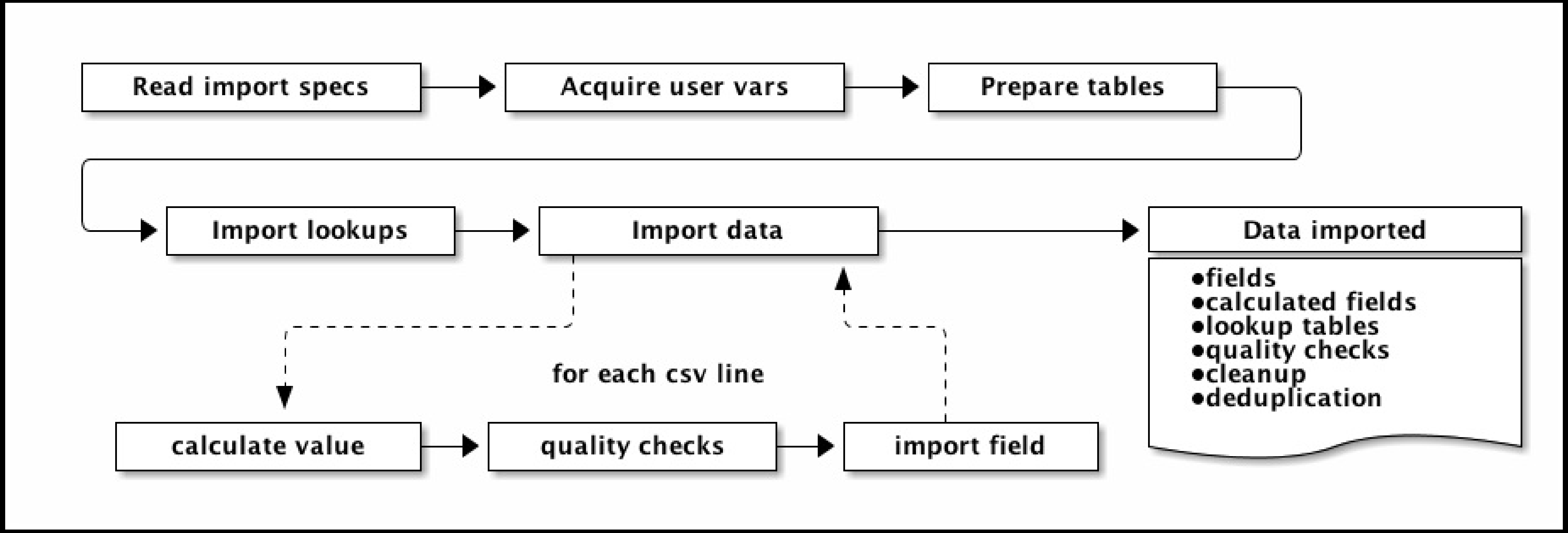
The data are imported using the import specifications, and only the defined fields are imported.

The import specifications can also define new calculated fields based on the value of the existing ones.

This kind of fields are useful to augment the data structure and/or to normalise the data.

The specifications can also define the import of accessory *(lookup)* tables.

DATA IMPORT



overview of the import process

QUALITY CHECKS

Quality checks can be defined at two levels:

- field level
- record level

the field level check can set the value to missing if it is invalid.

The record level check has access to the entire record and can modify or nullify every field; it can also signal to the import engine to discard the record if necesary.

CLEANUP

The last check performed on the imported record is the one for the mandatory fields.

In this case, if a mandatory field is missing, the entire record will be discarded.

RECORD DEDUPLICATION

NeuBiro implements a simple, and optional, LOCF (Last Observation Carried Forward) algorithm.

The task, if configured, will run at the end of the import.

As a result a new table will be created into the internal database, such a table will contain just one record per unique key.

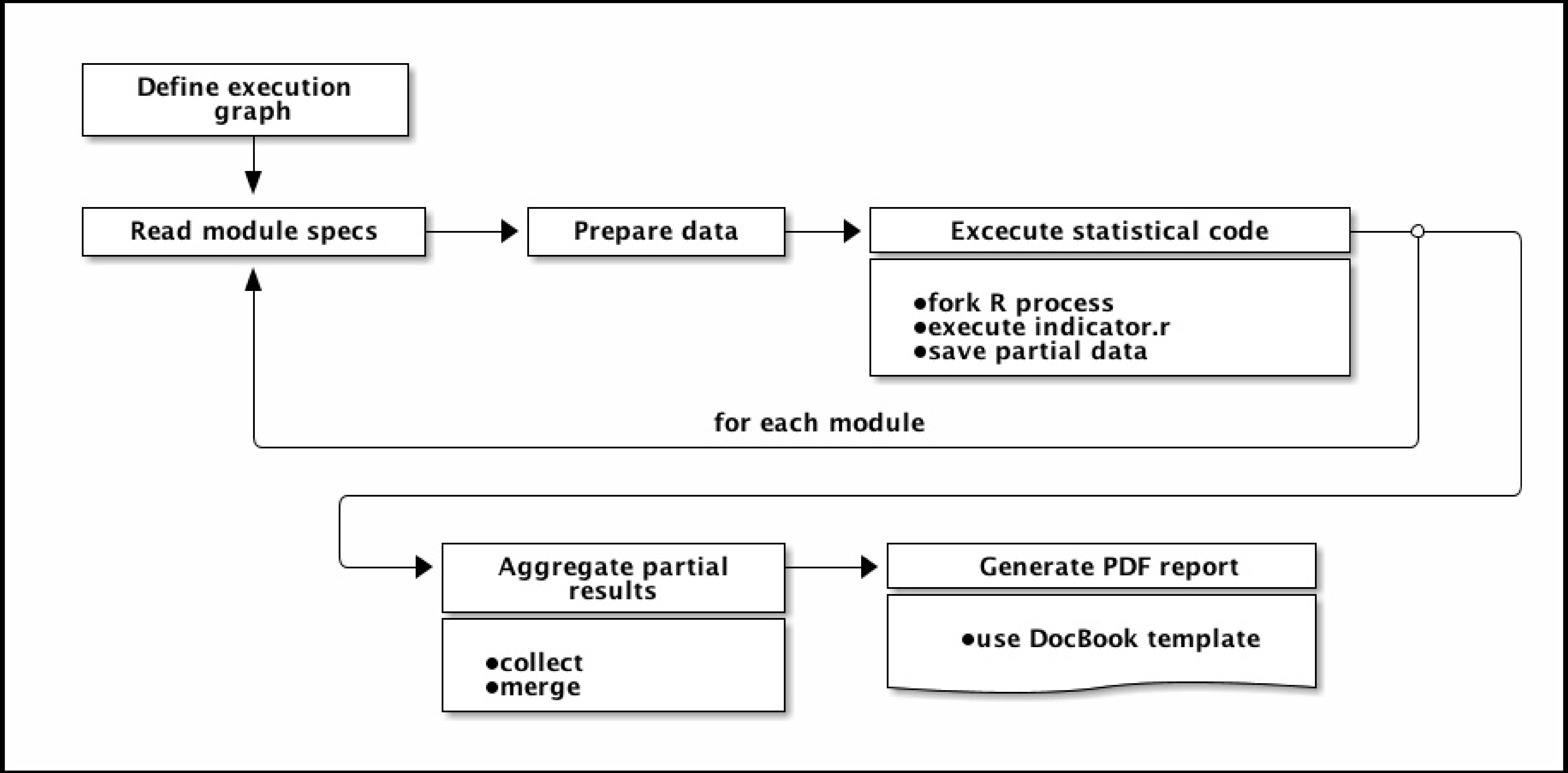
DATA PROCESSING

The data processing step is interactive, the user selects the statistical modules to run and starts the analysis.

At this point NeuBiro creates the execution graph *(taking care of the dependencies between modules)* and for each module:

- the module’s data are prepared (as defined in the module’s spec)
- the R process is invoked and writes on disk the partial data
- the R process returns and NeuBiro performs a check to verify the data have been created

DATA PROCESSING



data processing overview

DATA PROCESSING

Each module creates a part of the final report.

The module can also create other artifacts which can be used by other dependant modules during the processing phase.

With this technique it is possible to optimise the computations, modularise the code and avoid duplications.

REPORT GENERATION

After a successful completion of the analysis phase the report generation is automatically started.

All the partial outputs from the modules are collected and aggregated into a PDF file.

REPORT GENERATION

The PDF file is generated through a DocBook template.

The template's layout can be different and customised for each statistical package.

CONFIGURATION

Every aspect of the analysis can be customised.

The main configurable components are:

- the import specifications
- the statistical package
- the report template

IMPORT SPECIFICATIONS

Import can be configured with a complete DSL (Domain Specific Language) based on Groovy language.

- tables to import
- fields to import
- lookup tables
- calculated fields for main and lookup tables
- quality checks
- deduplication task

IMPORT SPECIFICATIONS

```
master {
  'THETABLE' {
    label = "Master table file"
    mandatory = true

    fields {
      // tag::field[]
      'PAT_ID' { (1)
        type = "varchar" (2)
        size = 10
        mandatory = true
      }
      // end::field[]

      // tag::field_format[]
      'BIRTH_DATE' {
```

FIELD DEFINITION

```
'PAT_ID' { (1)
  type = "varchar" (2)
  size = 10
  mandatory = true
}
```

1. field name
2. field type

FIELD FORMAT DEFINITION

```
'BIRTH_DATE' {
  type = "date" (1)
  format = "dd/MM/yyyy" (2)
}
```

1. field of type date
2. the date format

⚠ If the field cannot be parsed with the provided format it will be set to missing

CALCULATED FIELDS DEFINITION

```
'RECORD_DATE' {
  persist = false 1
  type = "date"
  value = { record, context -> 2
    try {
      Date.parse("yyyy-MM-dd", record['BIRTH_DATE']) 3
    }
    catch(Exception) { 4
      null
    }
  }
}
```

1. do not create a column (temporary field not persisted)
2. code to calculate the field’s value
3. every java/groovy statement can be used
4. error handling

QUALITY CHECKS

The quality check can be defined at field and at record level.

The field level check can set the value of the field to null (missing).

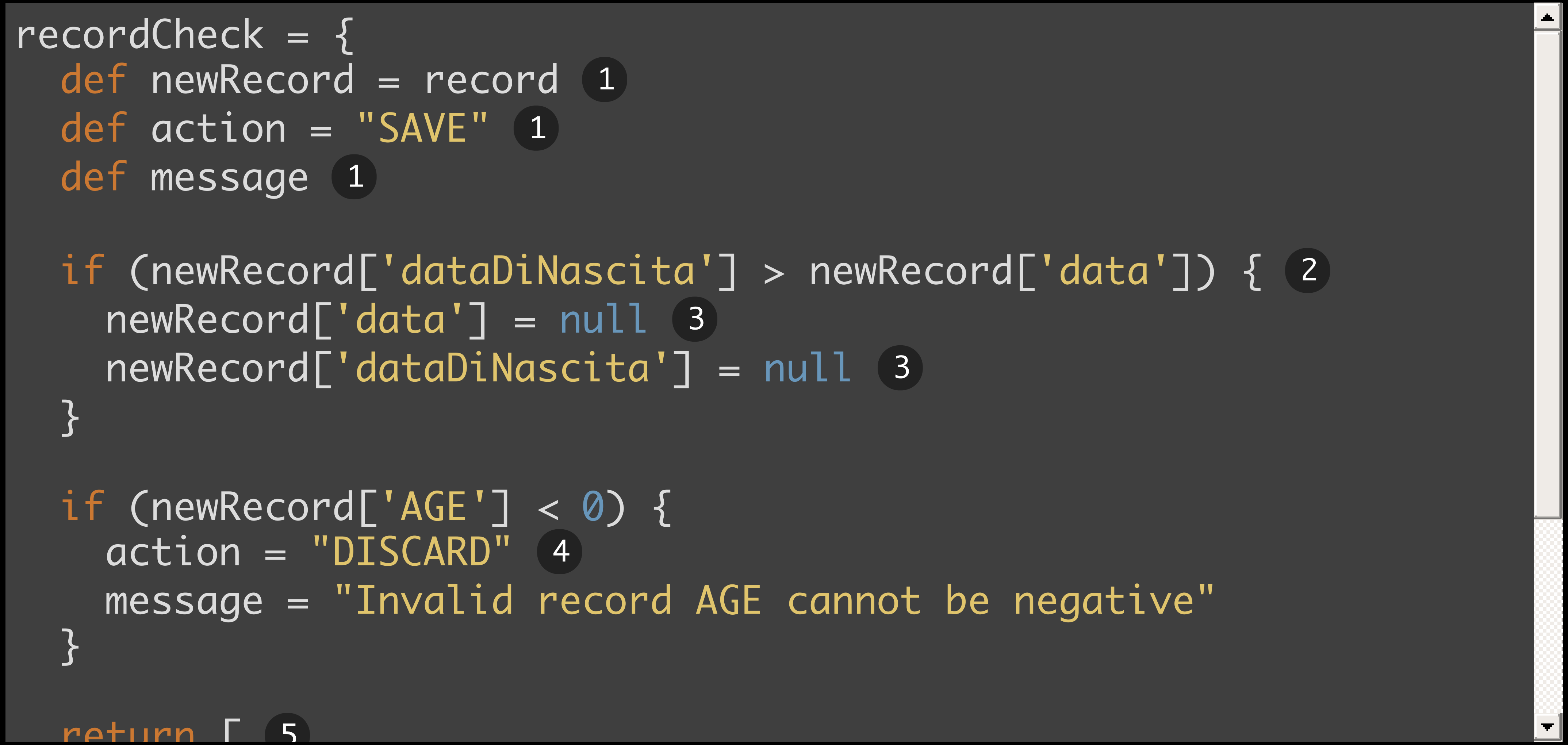
The record level check, has visibility on the entire record; it can modify every field and even discard the entire record if desired.

FIELD LEVEL QUALITY CHECKS

```
fields {
  'TYPE_DM' {
    type = "int"
    valid = { value ->
      if (!value in [1, 2, 3]) { (1)
        false
      }
    }
  }
}
```

(1) if not in range return false. eg set value to MISSING

RECORD LEVEL QUALITY CHECKS



```
recordCheck = {
  def newRecord = record 1
  def action = "SAVE" 1
  def message 1

  if (newRecord['dataDiNascita'] > newRecord['data']) { 2
    newRecord['data'] = null 3
    newRecord['dataDiNascita'] = null 3
  }

  if (newRecord['AGE'] < 0) {
    action = "DISCARD" 4
    message = "Invalid record AGE cannot be negative"
  }

  return [ 5
```

1. support variables
2. check the value of the field
3. set the fields to missing
4. discard the entire record
5. return structure for NeuBiro core

LOCF CONFIGURATION

```
locf {
  table = "MASTER_LOCF" 1
  keys  = ['PAT_ID'] 2
  order = ['PAT_ID', 'EPI_DATE'] 3
  exclude = ['CHOL', 'HDL', 'LDL'] 4
}
```

1. name of the resultig table after the LOCF task
2. list of fields representing the unique key
3. order of the table
4. optional list of fields excluded from the LOCF task

LOOKUP TABLES DEFINITION

```
lookups {
  'LOOKUP_TABLE_ONE' {
    label = "Simple lookup table"
    mandatory = true
    skipAutoId = true
    fields {
      'CODE' {
            type = "varchar"
            size = 16
      }
      'NAME' {
            type = "varchar"
            size = 6
      }
    }
```

FINE TUNING IMPORT

define indexes

```
indexes {
  'codeidx' {
    primary = true
    unique = true
    fields = ['CODE']
  }
}
```

STATISTICAL PACKAGE

The statistical package provides the logic of the analysis.

A statistical package can contain one or more modules (AKA indicators).

Each module is composed by:

- the specification file
- one or more implementation files (R source code)

THE SPECIFICATION FILE

The `indicator.specs` file contains the description of the module:

- the ID for the indicator
- optional dependencies from other modules
- a human readable description (i18n available)
- the data preparation query
- a declaration of the expected output (used for error checking)

THE SPECIFICATION FILE

Its main purpose is to describe all the objects needed for the R code to operate correctly.

Each module can optionally depend on the execution of other modules.

At the start of each analysis NeuBiro creates an execution plan for the modules and run each of them in the correct order.

This lead to code modularization in observance of the DRY (Don't Repeat Yourself) principle.

THE SPECIFICATION FILE

```
indicator {
  id = 'module_1_1'

  description = "Create data file example"

  dependsOn = ['setup'] 1

  hidden = true
  excludeReport = true

  input { 2
    table = "MAIN"

    fields = [ 3
      'DIST_MMG', 'MMG',
      'SEX', 'AGE_RANGE',
```

1. dependant modules (to be executed before the current one)
2. input definition for R code
3. field’s selection
4. filename for the output of the `input` block
5. expected output

STATISTICAL CODE

The statistical operations are implemented with the R language.

The R process receives all the inputs defined in `indicator.specs` greatly simplifying the code's structure.

The R code can concentrate its efforts on the single task to perform and using only the needed data.

Thanks to the dependency system one module can also access data prepared from other modules.

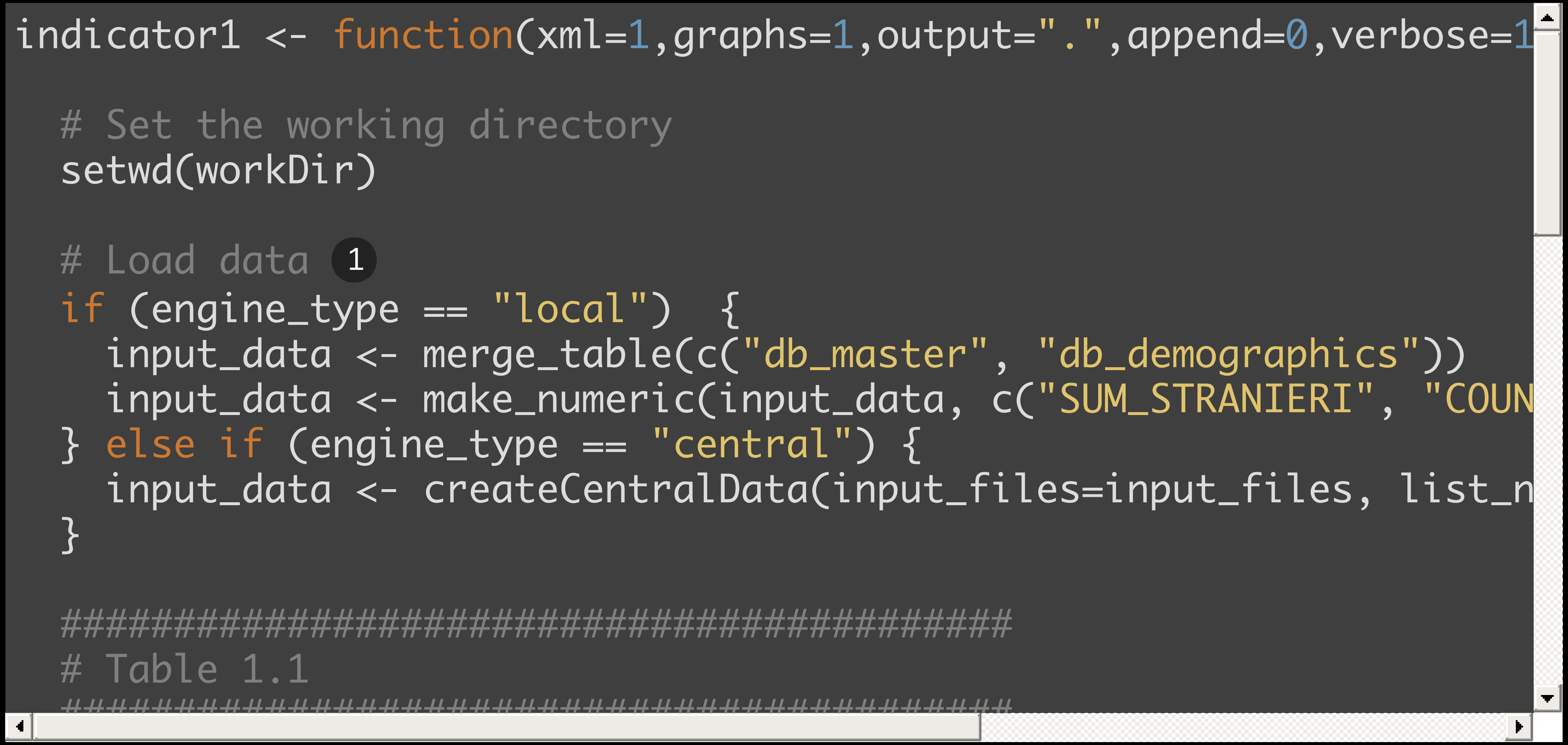
IMPLEMENTATION CODE

sample code for a module

```
source(paste(baseDir, "/commons/tools.r", sep="")) 1

source(paste(baseDir,"/module/implementation.r", sep="")) 2

indicator1() 3

rm(list=ls(all=TRUE)) 4
```

1 load external modules from common path (eg. common functions, libraries, etc.)
2 load implementation from external file in same module
3 execute the main function
4 cleanup

IMPLEMENTATION CODE



```
indicator1 <- function(xml=1,graphs=1,output=".",append=0,verbose=1

  # Set the working directory
  setwd(workDir)

  # Load data 1
  if (engine_type == "local")  {
    input_data <- merge_table(c("db_master", "db_demographics"))
    input_data <- make_numeric(input_data, c("SUM_STRANIERI", "COUN
  } else if (engine_type == "central") {
    input_data <- createCentralData(input_files=input_files, list_n
  }

  ##########################################
  # Table 1.1
  ##########################################
```

1. load the prepared data
2. processing
3. write partial data (will be aggregated during report generation)

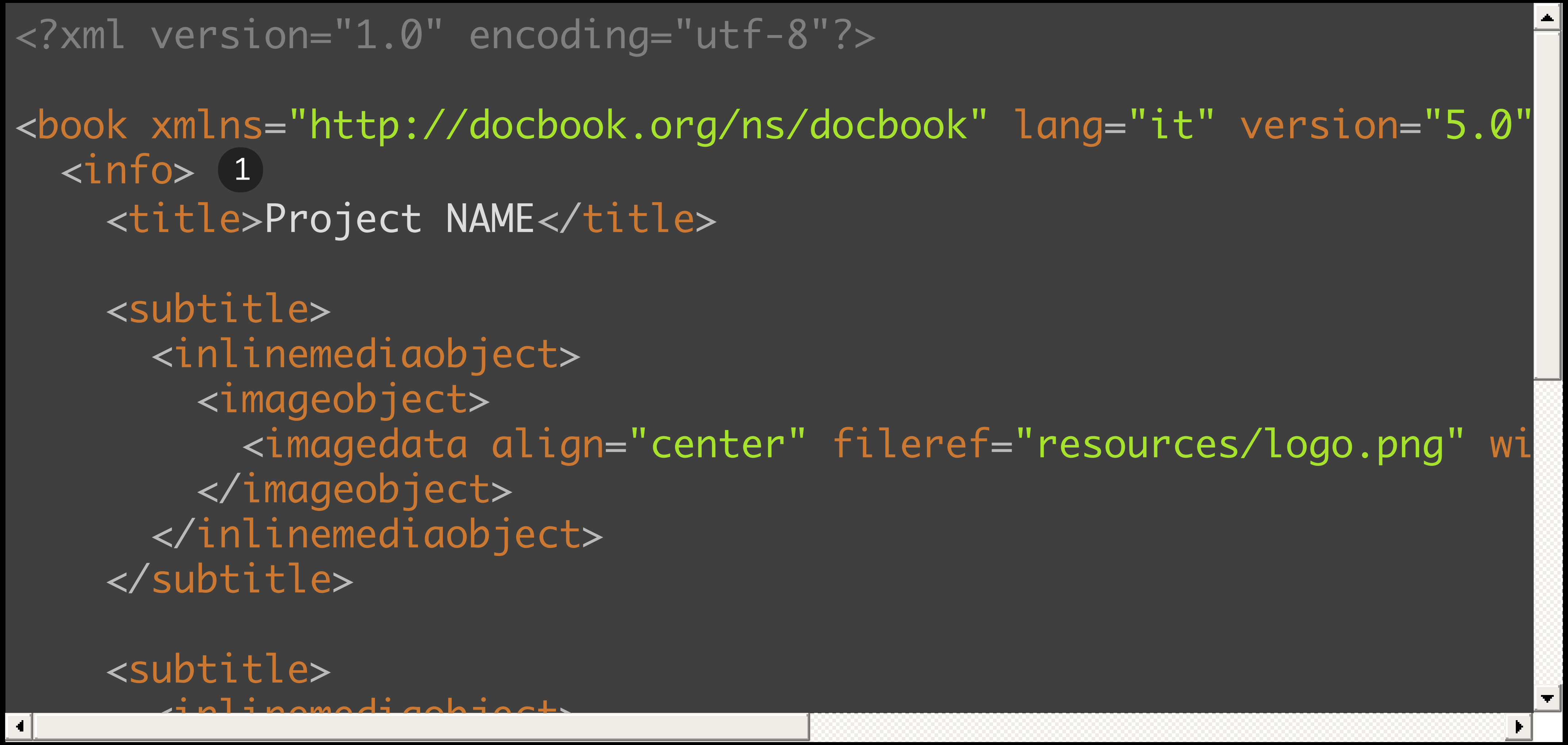
FINAL REPORT

The final report is rendered with the DocBook processor.

The entire process uses a configurable template.

The output is a PDF file.

MASTER TEMPLATE



```
<?xml version="1.0" encoding="utf-8"?>

<book xmlns="http://docbook.org/ns/docbook" lang="it" version="5.0"
  <info> 1
    <title>Project NAME</title>

    <subtitle>
      <inlinemediaobject>
        <imageobject>
          <imagedata align="center" fileref="resources/logo.png" wi
        </imageobject>
      </inlinemediaobject>
    </subtitle>

    <subtitle>
```

1. cover page
2. table of contents (from the list of executed modules)
3. content of the report (collected data from each module)

CHAPTER TEMPLATE

```
<chapter xml:id="${id}">
  ${body} (1)
</chapter>
```

(1) content of the single module output

HOW TO INSTALL NEUBIRO

NeuBiro can be easily installed with the provided installer.

The installer creates on the target computer everything needed to start working with the software.

MINIMAL REQUIREMENTS

NeuBiro requires the following mandatory dependencies:

Component	**Minimium version**
Java Virtual Machine	1.7
R	3.3.x

⚠ The dependencies must be installed **before** NeuBiro.

RUNNING THE INSTALLER

After the download the installer can be run with a simple double-click on the file:

`neubiro-installer-0.6.jar`

or from the command line with the following command:

```
java -jar neubiro-installer-0.6.jar
```

ℹ The installer program is multi platform and can be launched on Microsoft Windows and Linux.

INSTALLED RESOURCES

The installer is organized in packages, some of which are optional.

The provided packages are:

1. the main executables for NeuBiro *(mandatory)*
2. the BIRO System statistical package
3. the sample statistical package
4. the sample data
5. the documentation for users and programmers

INSTALLATION LAYOUT

```
C:\NeuBiro-0.6 1
├──docs 2
├──packages 3
|  ├──neubirod 4
|  |  ├──import 5
|  |  └──indicators 6
|  └──sample-package 7
|     ├──import
|     └──indicators
└──sample-data 8
```

1. Main installation folder
2. Documentation in PDF e HTML format
3. Statistical package main folder
4. BIRO System statistical package
5. Import specifications
6. Statistical modules
7. Sample statistical package
8. Sample data set

OPEN SOURCE

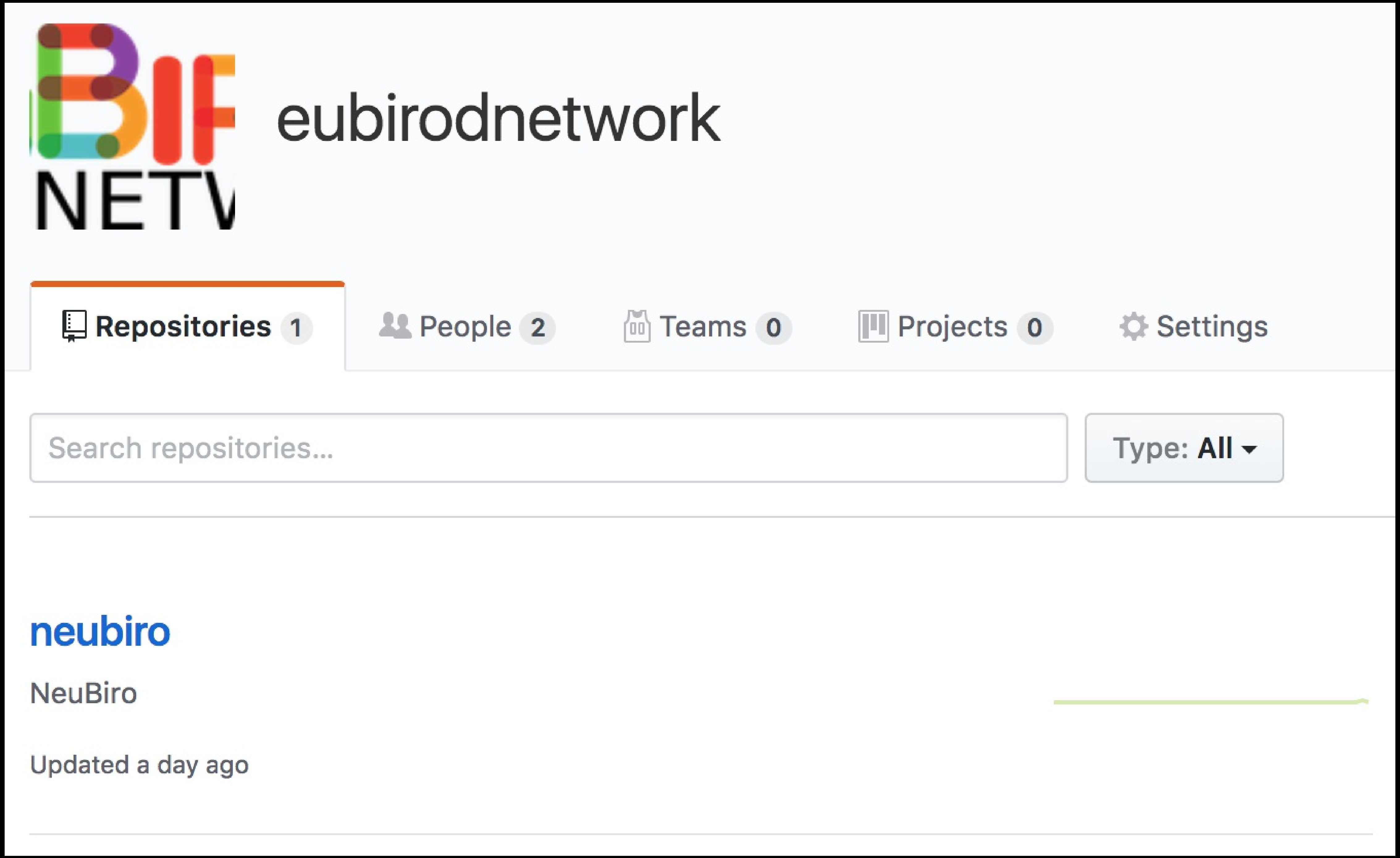
NeuBiro is open source software released under the EUPL v1.1 license.

The source code will be published on GitHub soon.

The EUBIROD network account on GitHub can be reached at the following address:

`https://github.com/eubirodnetwork`

GITHUB: SOURCE CODE



eubirodnetwork

neubiro

NeuBiro

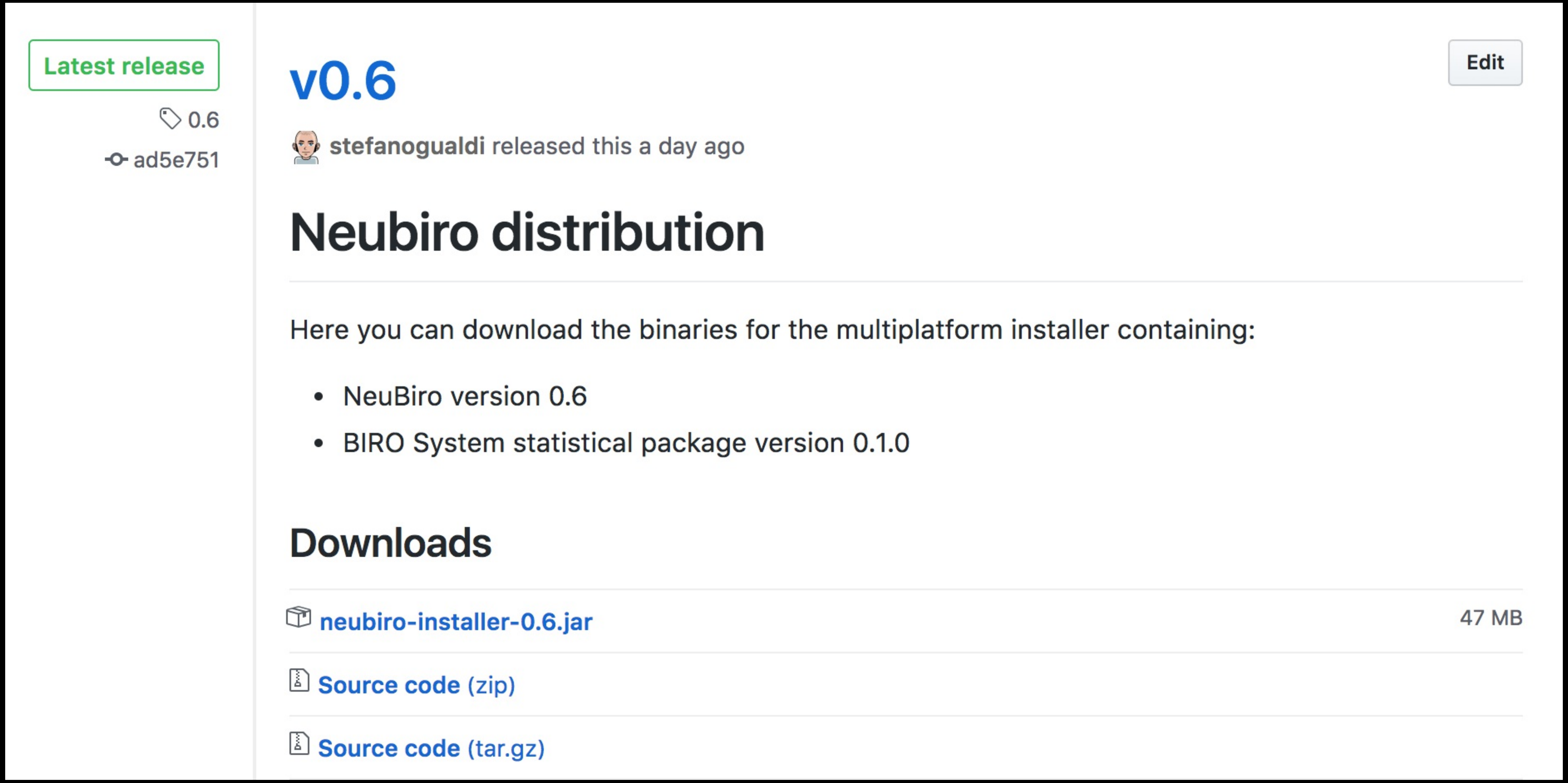
Updated a day ago

BINARY FILES

The binary components can be downloaded at the following page on GitHub:

`https://github.com/eubirodnetwork/neubiro/releases`

GITHUB: BINARY DOWNLOAD


Latest release

0.6

ad5e751

v0.6

Edit

stefanogualdi released this a day ago

Neubiro distribution

Here you can download the binaries for the multiplatform installer containing:

- NeuBiro version 0.6
- BIRO System statistical package version 0.1.0

Downloads

neubiro-installer-0.6.jar	47 MB
Source code (zip)	
Source code (tar.gz)	

THANK YOU