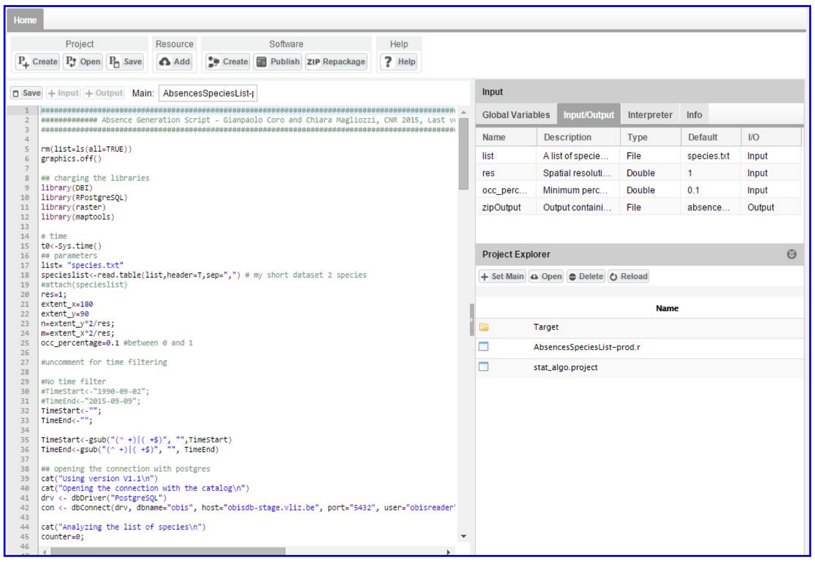
A Web application to publish R scripts as-a-Service on a Cloud computing platform

Gianpaolo Coro, ISTI-CNR (Italy), gianpaolo.coro@isti.cnr.it
Giancarlo Panichi, ISTI-CNR (Italy), giancarlo.panichi@isti.cnr.it
Pasquale Pagano, ISTI-CNR (Italy), pasquale.pagano@isti.cnr.it

Prototype scripting is the base of most models in computational biology and environmental sciences. Scientists making prototype scripts (e.g. using R and Matlab) often need to share results and make their models used also by other scientists on new data. To this aim, one way is to publish scripts as-a-Service, possibly under a recognized standard (e.g. the Web Processing Service of the Open Geospatial Consortium). Unfortunately, prototype scripts are not generally meant to be transformed into Web services, which require managing multi-tenancy, concurrency etc. Often, porting prototype scripts to more efficient programming languages is not affordable, because this operation demands for time, competencies and money. For this reason, Web services are becoming smart enough to integrate prototype scripts directly and possibly make them run efficiently (e.g. WPS4R[1]).



```
############################################################################
############# Absence Generation Script - Gianpaolo Coro and Chiara Magliozzi, CNR 2015, Last v
############################################################################

rm(list=ls(all=TRUE))
graphics.off()

## charging the libraries
library(DBI)
library(RPostgreSQL)
library(raster)
library(maptools)

# time
t0<-Sys.time()
## parameters
list= "species.txt"
specieslist<-read.table(list,header=T,sep=",") # my short dataset 2 species
#attach(specieslist)
res=1;
extent_x=180
extent_y=90
n=extent_y*2/res;
m=extent_x*2/res;
occ_percentage=0.1 #between 0 and 1

#uncomment for time filtering

#No time filter
#TimeStart<-"1990-09-02";
#TimeEnd<-"2015-09-09";
TimeStart<-"";
TimeEnd<-"";

TimeStart<-gsub("(^ +)|( +$)", "",TimeStart)
TimeEnd<-gsub("(^ +)|( +$)", "", TimeEnd)

## opening the connection with postgres
cat("Using version V1.1\n")
cat("Opening the connection with the catalog\n")
drv <- dbDriver("PostgreSQL")
con <- dbConnect(drv, dbname="obis", host="obisdb-stage.vliz.be", port="5432", user="obisreader'

cat("Analyzing the list of species\n")
counter=0;
```

Name	Description	Type	Default	I/O
list	A list of specie...	File	species.txt	Input
res	Spatial resoluti...	Double	1	Input
occ_perc...	Minimum perc...	Double	0.1	Input
zipOutput	Output containi...	File	absence...	Output

Name
Target
AbsencesSpeciesList-prod.r
stat_algo.project

Fig. 1. Interface of the D4Science Statistical Algorithms Importer.

In this paper, we present an interface (Statistical Algorithms Importer, SAI) that allows scientists to easily and quickly import R scripts onto a distributed e-Infrastructure, which publishes the scripts as-a-Service and manages multi-tenancy and concurrency. Additionally, it allows scientists to update their scripts without following long software re-deploying procedures each time. SAI relies on the D4Science e-Infrastructure[2], a distributed computer system supporting large-scale resource sharing and Cloud computing, via the definition of Virtual Research Environments (VREs). VREs define groups of scientists working together in the same domain and are endowed with social networking and collaborative facilities. VREs are the main collaboration and sharing facilities offered by D4Science. SAI produces algorithms that run on the D4Science Cloud computing platform and are accessible via the WPS standard[3]. This platform can manage multiple versions of the R interpreter by selecting the most appropriate execution environment given the script's requirements. SAI has been developed in the context of European projects having fisheries, biodiversity and environmental science experts as target users (BlueBRIDGE, i-Marine, D4Science).

The SAI interface (Fig.1) resembles the R Studio environment, a popular IDE for R scripts. Our aim was in fact to make SAI friendly to our script providers. The *Project* button allows creating, opening

[1]52North, WPS4R for creating WPS processes based on annotated R-scripts, https://wiki.52north.org/bin/view/Geostatistics/WPS4R
[2]Candela, L.; Castelli, D.; Manzi, A.; Pagano, P. Realising Virtual Research Environments by Hybrid Data Infrastructures: the D4Science Experience. International Symposium on Grids and Clouds (ISGC) 2014, Proceedings of Science PoS (ISGC2014) 022
[3]Coro, G., Candela, L., Pagano, P., Italiano, A., & Liccardo, L. (2015). Parallelizing the execution of native data mining algorithms for computational biology. Concurrency and Computation: Practice and Experience, 27(17), 4630-4644.

and saving a working session. A user uploads a set of files and data on the workspace area (lower-right panel). Upload can be done by dragging and dropping local desktop files. As next step, the user indicates the “main script”, i.e. the script that will be executed on D4Science and that will use the other scripts and files. After selecting the main script, the left-side editor panel visualises it with R syntax highlighting and allows modifying it. Afterwards, the user indicates the input and output of the script by highlighting variable definitions in the script and pressing the *+Input* (or *+Output*) button: behind the scenes the application parses the script strings and guesses the name, description, default value and type of the variable. This information is visualised in the top-right side *Input/Output* panel, where the user can modify the guessed information. Alternatively, SAI can automatically fulfill the same information based on WPS4R annotations in the script. Other tabs in this interface area allow setting global variables and adding metadata to the process. In particular, the *Interpreter* tab allows indicating the R interpreter version and the packages required by the script and the *Info* tab allows indicating the name of the algorithm and its description. In the *Info* tab, the user can also specify the D4Science VRE the algorithm should be available to.

Once the metadata and the variables information has been fulfilled, the user can create a D4Science as-a-Service version of the script by pressing the *Create* button in the *Software* panel. With the term "software", in this case we mean a Java program that implements an as-a-Service version of the user provided scripts. The Java software contains instructions to automatically download the scripts and the other required resources on the server that will executed it, configure the environment, execute the main script and return the result to the user. The computations are orchestrated by the D4Science Cloud computing platform that ensures the program has one istance for each request and user. The servers constitute a distributed services system that manages concurrent requests by several users and executes code in a closed sandbox folder, to avoid damage caused by malicious code. Based on the SAI Input/Output definitions written in the generated Java program, D4Science automatically creates a Web GUI[4]. In the creation phase, SAI checks the presence of at least one input and one output, the algorithm name, its description and the interpreter’s version. A package is created under the *Target* folder that includes a Java archive (JAR), metadata information and a zip file containing the script and the accessory resources. By pressing the *Publish* button, the application notifies D4Science system administrators that a new process should be deployed. Upon approval, the D4Science staff, assisted by appropriate tools, puts the JAR on the computational platform, but not the zip file. The obfuscated-compiled Java code inside the JAR contains an HTTP link to the zip file, which remains in the user’s private project area. Thus, D4Science will not own the source code, which is downloaded on-the-fly by the computing machines and deleted after the execution. This approach meets the policy requirements of those users who do not want to share their code. The *Repackage* button re-creates only the zip package, which will substitute the previous file and inherit the same HTTP link. Thus, the JAR file on the computational platform will be using the new version of the script. The repackaging function allows the user to modify the script and to immediately have the new code running on the computing system. This approach separates the script updating and deployment phases, making the script producer completely independent on e-Infrastructure deployment and maintenance issues. However, deployment is necessary again whenever Input/Output or algorithm’s metadata are changed.

To summarise, the SAI Web application relies on the D4Science e-Infrastructure and enables an R script, provided by a community of practice working in a VRE, with as-a-Service features. SAI reduces integration time with respect to direct Java code writing. Additionally, important functions that D4Science offers[2] include (i) multi-tenancy and concurrent access, (ii) scope and access management through Virtual Research Environments, (iii) output storage on a distributed, high-availability file system, (iv) graphical user interface, (v) WPS interface, (vi) data sharing and publication of results, (vii) provenance management and (viii) accounting facilities.

[4]Coro, G., Gioia, A., Pagano, P., & Candela, L. (2013). A service for statistical analysis of marine data in a distributed e-infrastructure. *Bollettino di Geofisica Teorica e Applicata*, *54*(1), 68-70.