Future Generation Computer Systems 25 (2009) 481–488

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

GridR: An R-based tool for scientific data analysis in grid environments

Dennis Wegener [a,1], Thierry Sengstag [b,2], Stelios Sfakianakis [c,3], Stefan Rüping [a,4], Anthony Assi [d,*]

[a] Fraunhofer Institute Intelligent Analysis and Information Systems IAIS, Schloss Birlinghoven, 53754 St. Augustin, Germany
[b] Swiss Institute of Bioinformatics, Bâtiment Génopode, CH-1015 Lausanne, Switzerland
[c] Biomedical Informatics Laboratory, Institute of Computer Science, FORTH, Greece
[d] Ouest-genopole Bio-Informatics Platform, Symbiose Team, INRIA/IRISA, Rennes, France

ARTICLE INFO

Article history:
Received 16 June 2008
Received in revised form
25 August 2008
Accepted 1 September 2008
Available online 20 September 2008

Keywords:
R statistical computing
Grid
Globus toolkit
ACGT
Genomics data mining

ABSTRACT

In this paper, we describe an analysis tool based on the statistical environment R, GridR, which allows using the collection of methodologies available as R packages in a grid environment. It provides the user with transparent and seamless access to large-scale distributed computational services and data repositories within the secure and reliable framework of a grid system. The aim of GridR, which was initiated in the context of the EU project Advancing Clinico-Genomics Trials on Cancer (ACGT), is to provide a powerful framework for the analysis of clinico-genomic trials involving large amount of data (e.g. microarray-based clinical trials). As a proof of the concept, an example of microarray-based analysis taken from the literature was reproduced using GridR.

1. Introduction

Scientific research today, and eScience in particular, is often characterized by an iterative process, switching between human hypothesis generation and experimentation, and computationally intensive data checking and statistical analysis. In cancer research for instance, the identification of novel biomarkers with the recent use of high-throughput technologies such as gene expression arrays (microarrays) has suggested new approaches to patient treatment. For instance, in [7], microarray information allows to identify a subtype of breast cancer, in which the gene pattern suggests that it might be treated with drugs known to be efficient with prostate cancer, which is why we selected this work as illustration for the use of our grid-based data analysis environment. This new hypothesis, currently tested in the context of a clinical trial [6], will generate new data which can help further refinements in the characterization of the disease. An issue with recent clinical trials is the increasing difficulty of the statistical treatment of data, both because of the increasing size of the data sets, and of the complexity of the statistical design of the experiments. The R statistical toolkit [19] has proven to be the tool of choice to address this issue in a wide variety of applications because of its flexibility and interactivity, and the large number of available software packages. However, in state of the art scientific research, single tools usually do not suffice, as important steps such as data gathering and data integration, and important requirements such as data privacy and reliability have to be taken care of. Grid technologies promise to effectively deal with these requirements, however often at the cost of high additional effort and less interactivity. In this paper, GridR [26], a software to transparently embed the R environment into a grid framework is presented and evaluated in the context of ACGT [1], a large-scale project in the area of medical informatics.

1.1. Overview of ACGT

In recent years, the rapid development of high-throughput genomics and post-genomics technologies has provided clinicians fighting cancer with new discovery paths and has opened the possibility to develop patient-specific treatment strategies. However, the amount of information now available for each patient (e.g. in microarray context from 10'000s to 100'000s of variables summarizing up-to millions of array features) has rendered difficult the isolation of the clinically relevant information from all available data. Considering the current size of clinical trials

* Corresponding author. Tel.: +33 2 99 84 71 58.
E-mail addresses: dennis.wegener@iais.fraunhofer.de (D. Wegener), thierry.sengstag@isb-sib.ch (T. Sengstag), ssfak@ics.forth.gr (S. Sfakianakis), stefan.rueping@iais.fraunhofer.de (S. Rüping), anthony.assi@inria.fr (A. Assi).
1 Tel.: +49 2241 14 2261.
2 Tel.: +41 21 692 4096.
3 Tel.: +30 2810 391650.
4 Tel.: +49 2241 14 3512.

0167-739X/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2008.09.004

(hundreds to thousands of patients), there is a clear need, both from the viewpoint of the fundamental research and from that of the treatment of individual patients, for a data analysis environment that allows the exploitation of this enormous pool of data [21]. This is the aim of the Advancing Clinico-Genomics Trials on Cancer (ACGT) project [1].

ACGT aims at developing an open-source IT infrastructure to provide the biomedical community with the tools needed to integrate complex clinical information and make a concrete step towards the tailorization of treatment to the patient.

On the data side, the ACGT environment is designed to be versatile and will allow the integration of databases with data both from existing (e.g. microarrays, imaging) and future (e.g. sequencing and proteomics) high-throughput technologies. The design of the platform considers the integration of private (i.e. trial-specific) databases with public ones, thus making publicly available datasets potentially immediately available for hypothesis validation and meta-analyses.

On the methodology side, the ACGT platform is designed to be modular, allowing to integrate additional data analysis tools (software, both open-source and commercial, web services) as plug-ins, as they become available.

Considering that the amount of data generated is expected to rise to several gigabytes of data per patient in a close future access to high-performance computing resources will be unavoidable. Hence, grid computing [9] appears as a promising technology. Access and use of grid-based resources is thus an integral part of the design of the infrastructure.

There are a number of other projects that aim at developing grid-based infrastructure for post-genomic cancer clinical trials, the most advanced of which are NCI's caBIG [4] (Cancer Biomedical Informatics Grid) in the USA and CancerGrid [3] in the UK. The overall approach in those projects is somewhat different from the one in ACGT. In caBIG and CancerGrid, the bottom-up, technology-oriented, approach was chosen, in which the focus was put on the integration of a large number of analysis tools but with a relatively weak concern on data privacy issues and a looser approach to semantic integration of data. Somewhat differently the ACGT data and security architecture emphasizes the concern to provide an ethically and legally sound treatment of patient data in a grid environment. In addition, ACGT uses an "uniform ontology"-based approach to data annotation which greatly facilitates the integration of heterogeneous data sources.

Security is of paramount importance to have such an architecture accepted in the clinical community. Thus, the ACGT architecture guarantees that the data issued from clinical trials are properly anonymized before any processing, transport or temporary storage on publicly accessible infrastructures occurs. Access to the various components of the system is granted only after a proper identification of the user and on a case-by-case basis. Anonymized data can be re-identified upon request of stakeholders entitled to have access to non-anonymized data through the services of a trusted third-party. The Center for Data Protection [5] was created for this purpose as a spin-off of the project ACGT.

Despite their apparent complexity, these security mechanisms are fully transparent to the user, once s/he has undergone a one-time registration procedure at an ACGT-recognized certification body.

Two on-going international clinical trials are actually conducted in the framework of the ACGT project, thus ensuring that clinicians' and biomedical data analysts' needs remain at the heart of the development of the platform.

1.2. R in clinico-genomic context

The present article describes the initial implementation of GridR [26] one of the core analysis tools to be used in the ACGT environment. GridR is based on the open-source statistical package R [19]. The R environment provides a broad range of state of the art statistical, graphical techniques and advanced data mining methods (including comprehensive packages for linear and non-linear modelling, cluster analysis, prediction, hypothesis tests, resampling, survival analysis, time-series analysis, etc.), it is easily extensible and turned out as the de facto standard for statistical research in many applied statistics project. In particular the associated project BioConductor [11] addresses the needs of the biomedical and biostatisticians community by providing very quickly R packages to analyze data issued from new technologies appearing in the biology laboratory.

Numerous methods available as R/BioConductor packages and considered experimental a few years ago have been stabilized and became accepted standard in the analysis of high-throughput genomic data. Integrating R/BioConductor in an open-source clinical data-analysis environment is thus completely meaningful.

In the design of the ACGT analysis environment, R was meant to be used interactively as well as wrapped in analysis components embedded in workflows. R as used interactively allows a programmatic access to the resources of the ACGT environment, e.g., running R code for analysis on the grid from within a local R session. Used as an analysis component in workflows, it allows a seamless integration of the functionality of R with the ACGT semantic data services, hiding the implementation details of the R scripts from the workflow developer.. These two approaches in the use of R form the core of GridR, which aims at hiding the complexity of the grid environment from the user. The biggest novelty introduced by GridR is the exploitation of state of the art grid infrastructures for the efficient and secure execution of R code. GridR is integrated with the rest of the ACGT environment but can also be used separately provided that a compliant grid platform is in place.

The rest of the paper is organized as follows: In Section 2 we sketch the ACGT architecture and give an overview over the key components. Section 3 describes the main features, GridR, as implemented in ACGT. In Section 4 the clinico-genomics analysis scenario ("Farmer scenario") used for the present illustration is described, which is evaluated in Section 5. In Section 6 we give an overview of related work. Finally, in Section 7 we discuss the results of the paper and give a future outlook.

2. ACGT architecture

The ACGT requirements in terms of data management, efficient utilization of computational resources, and security can be matched by the adoption of a grid infrastructure. The adopted architecture builds upon the grid fabric and it is further enhanced by the deployment of Web Services and Semantic Web technologies. These technologies, although initially separated, are currently converging in a complementary way and the ACGT platform is a case which demonstrates the feasibility and the added-value of such a convergence and integration.

The architecture that was adopted for ACGT is shown in greater detail in Fig. 1. A layered approach has been followed to providing different levels of abstraction and a classification of functionality into groups of homologous software entities [24]. In this approach the security services and components are pervasive throughout ACGT in order to facilitate user management, trust bindings and access-rights management and enforcement. The Grid and domain-specific security mechanism used ensure the proper implementation of security requirements like pseudonymization

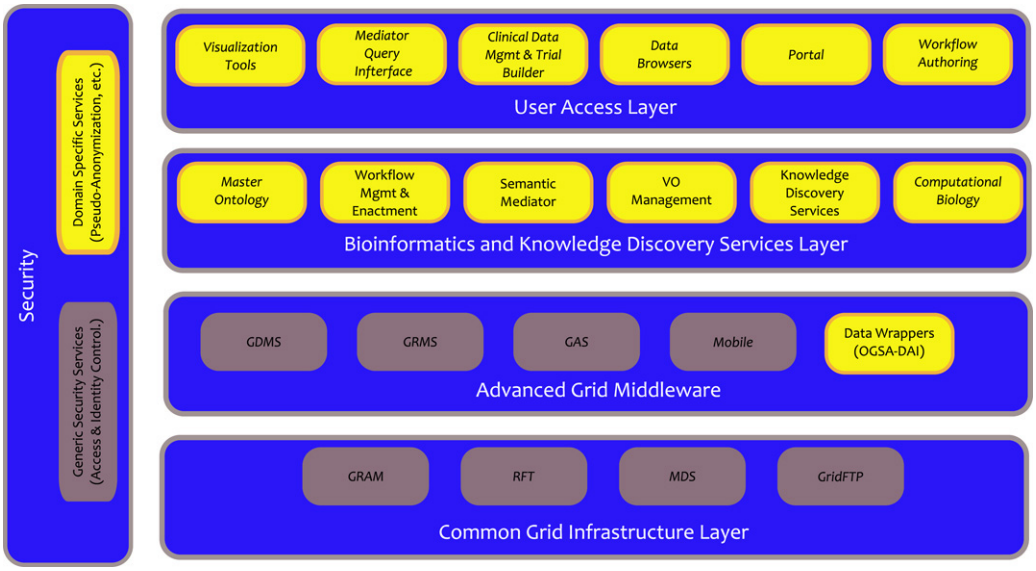

Fig. 1. The ACGT layered initial architecture.

and anonymization. Apart from the security requirements, the grid infrastructure and other services are located in the first two layers (bottom of Fig. 1). In particular the grid Authorization Service (GAS) is the central entity for managing access authorization rules in the context of a Virtual Organization and Myproxy services greatly facilitate the credentials management tasks by storing the user certificates through a familiar user name and password authentication mechanism.

The middle layer consists of the Bioinformatics and Knowledge Discovery Services. These services are the "workhorse" of ACGT and the corresponding layer is where the majority of ACGT specific services lie. Some characteristic examples of these services are:

- Mediator Services that offer uniform access to distributed and heterogeneous clinical and biomedical databases.
- Ontology Services that provide a conceptualization of the domain through the Master Ontology for Cancer and the "domain of discourse" for constructing complex queries for the mediator services.
- Workflow Enactment Services that support the efficient management and execution of complex biomedical workflows.
- Metadata Repositories environment which ensures the persistence and proper management of the metadata description of the services available to the users.
- A number of data mining and knowledge discovery tools and services that fulfill the data analysis requirements of ACGT, with GridR as one of the most prominent tools.

The upper layer is where the user-accessed services, such as the portal and the visualization tools, reside. The portal is the main entry point for ACGT. The majority of the tasks a user needs to do in the context of a clinical trial, be it analysis of biological data, execution of services, enactment of published workflows, training and learning activities, and so on, are supported by the ACGT portal and its "portlets". Though this portal the users are able to also design new scientific workflows that combine data retrieval tasks and different data analysis tools in order to either test a hypothesis, or implement an experiment or a scenario. These activities are supported by the ACGT workflow editor (shown in Fig. 2), a web based drawing and workflow designing tool. The workflow editor is integrated with the rest of the architecture and provides, in particular, a graphical browser which facilitates the discovery of the existing ACGT services and their composition. In comparison to existing workflow editing environments the ACGT workflow editor is an example of the "Software as a Service" approach, eliminating the need to install and run the application on the users' desktop machines, and also featuring better integration with the grid and Service Oriented Architecture of the platform (e.g. server side execution of the workflows, central repository of all the workflows to better support sharing, etc.), Users can thus develop complex analysis pipelines, by interconnecting services registered in the metadata repository, and store them as new services for later reuse by the same or other users.

R scripts can be wrapped into such workflow elements by using GridR, hence giving the possibility to perform highly complex and flexible analyses. At the workflow level GridR is considered as a simple service that accepts an R script as input. Upon enactment of such workflow component, the details and the complexity of the mechanisms involved in the execution are hidden from the user. The R scripts are considered as another kind of reusable entities that are shared between the ACGT users and that are discoverable on the basis of their metadata descriptions, e.g. the algorithm they implement, the data types of their inputs, their quality properties (e.g. performance), etc.

The workflow depicted on Fig. 2 is an example of a specialized workflow, developed in another context, which interfaces two GridR scripts with a number of data sources (specifically: a set of mediator queries to and a set of static files) to produce the various types outputs, e.g. plots and matrices.

In summary, in terms of the architecture GridR is one of several ACGT services, which provides the use of R as a workflow component, that can be seamlessly integrated with the other data access and data analysis services. This integration is realized in practice by exposing a Web Service interface that makes it compliant with the platform.

As already mentioned, GridR can be also used in the traditional sense, i.e., in the context of a separate interactive R environment, through the call of a number of user functions (see Table 1). Such an interface is adequate for more advanced users or users that want to create new scripts for GridR that can subsequently shared in the context of ACGT. With respect to security both approaches are based on working with Myproxy credentials. For using R as interactive client, the information on the Myproxy certificate to be used can be pre-configured in a configuration file or passed when calling the respective functions interactively, whereas in the workflow context credential delegation is used. In the next section, however, we provide details about the internal mechanisms underlying the implementation of GridR without looking in detail at the different interfaces it provides to the users.

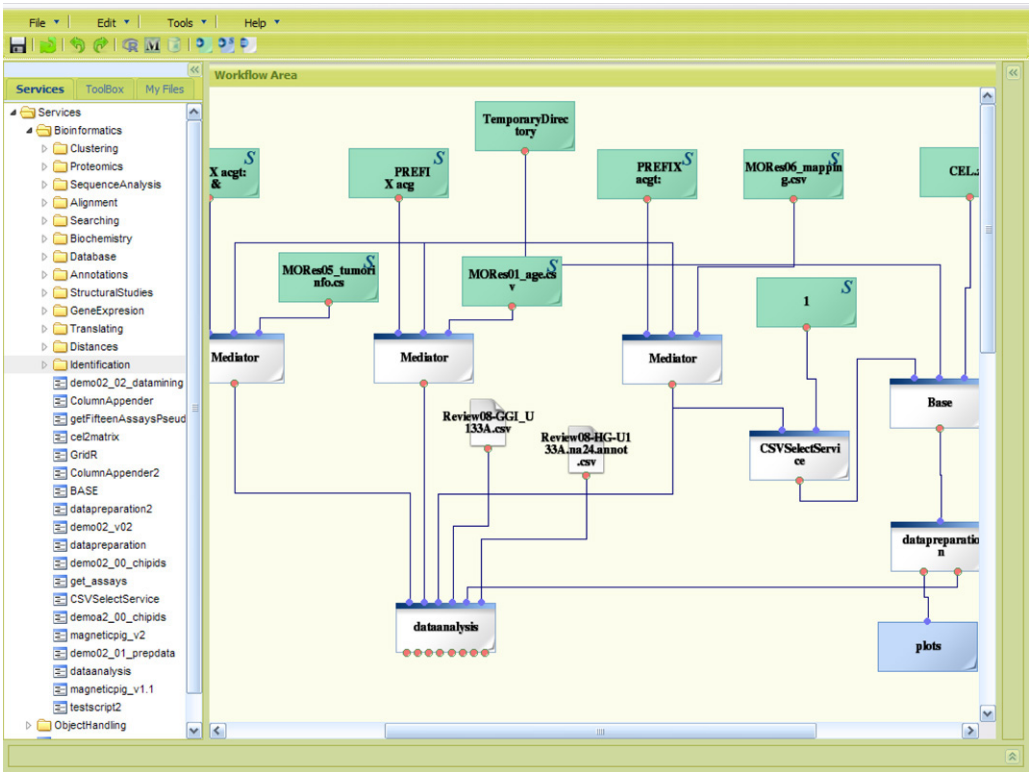

Fig. 2. A complex genomics data analysis workflow, represented in side the workflow editor portlet. (Two of the elements of the workflow, "datapreparation" and "dataanalysis", are GridR scripts.) The tree explorer on the left of the picture allows the user to select services registered in the metadata repository to add them to the workflow.

Table 1
List of functions for the users

grid.apply	Performs a remote execution of R functions; waits (callback) or sets a lock (grid.lock)
grid.check	Checks if varlist contains all variables and functions to execute function f remotely. If not the missing variables are returned
grid.myproxyGet	Gets a myproxy certificate from a myproxy server
grid.proxyInit	Creates new proxy cetificate with the use of globus
grid.consistency	Checks if the internal structure has errors or if there are local files without a running job
grid.isLocked	Checks if a variable has a lock

3. GridR

The GridR toolkit consists of the components GridR R environment, GridR services and GridR client. The GridR R environment is basically a standard R environment with some additional packages installed. This environment has to be installed on the machines which will execute the R functions remotely. The GridR services consist of several grid or web services wrapping the middleware components. One of these services is the ACGT GridR service, which is strongly connected and integrated into the ACGT architecture and can be used as component in ACGT workflows. This integration is achieved by offering a Web Service interface that exposes the execution of R scripts functionality to the ACGT legitimate users according to the security, quality of service, and other restrictions imposed by the platform. Interoperability with the rest ACGT services is realized and attested by the inclusion of GridR into non trivial scientific workflows. The GridR client is used as an interactive tool and programming language interface for accessing grid resources and in particular the remote execution of R code in the grid. More specifically, the task of the execution of the R code is submitted as a job to a remote machine in a distributed environment by interactively calling the respective functions from an R session (see Table 1 for an overview).

However, GridR is not limited to only access service layer components and submit the jobs through them. In general, the GridR client, which is implemented in form of an R package, provides the functionality of remote method invocation in a distributed environment. In detail, the GridR client contains components for submitting jobs to the GridR services or directly to grid resource management systems as the Globus Toolkit 4 grid middleware [8] using the Cog-kit [14], to the Gridge toolkit via a GRMS-Client [17], to the Condor system [15] which can also interface with grid middleware as well as to simple pools of machines contacted via SSH. Moreover, GridR provides support for parameter sweep on the client side which means that a number of jobs can be generated by performing a single submission from the GridR client.

The client side part is structured around the components "RemoteExecution" (JobSubmission and JobDescription Generator) and "Locking". The RemoteExecution component is responsible for the execution of R code as a job in the distributed environment by transforming the R code to execute into a set of files, creating a job description file in the respective job description language and submitting the job to the resource management system. During this process, the locking component takes care of the consistency of files/variables.

In practice, grid access is performed through the call of predefined R functions loaded from a package. The client side components are based on usual R functions (see Tables 1 and 2), so no

Table 2
List of internal used functions

Name	Action
grid.callback	Performs a callback
grid.waitForAll	Performs grid.callback for all jobs
grid.readLock	Read lock a variable
grid.writeLock	Write lock a variable
grid.unLock	Removes the lock from grid variable
grid.unlockAll	Unlocking of all grid variables
grid.catchObjectNotFoundError	Error handling by parsing of error messages

changes in the core R implementation are necessary. The functions are based, among other, on the following R functionalities:

- callbacks — functions that are executed automatically by R after the user has issued a command (this is used when checking for results).
- active bindings — a variable is replaced by a function call which is handling the locking system and allows working interactively with that variable. When the variable is read, the predefined function is called and returns the value associated to the variable (or an error code if the variable is locked). When a value is assigned to the variable the function is called with the value as parameter for storage in an internal structure.
- parsing of script and error code — check for missing values, variables and functions in the code which is executed remotely as well as for errors in the result objects.

R users can make use of the grid technology in a transparent way by passing the functions to be executed in the grid as input to one of those predefined functions (grid.apply) in their local code. Fig. 3 shows the execution of a simple sum function with the GridR client side components. Details on the steps of execution are described in Section 5.

In an extension of the concept currently under development, the GridR client will be provided as an R programming language interface that supports the access to all services of the ACGT environment. This means that R users and developers will have access to distributed resources in a transparent fashion, the complexity of the grid being, again, hidden from the user. And, again, accessing the ACGT grid environment requires no changes in the core R implementation.

4. Scenario (use case)

In order to illustrate the working principles of GridR, we have selected the article by Farmer et al. [7], a simple clinical research project available from the literature and for which all data were available online [10]. This also provides a validation of GridR in a realistic usage.

In the "Farmer scenario", microarray data (Affymetrix U133A gene expression microarrays) obtained from breast cancer tumor samples of 49 patients are used to associate subtypes of breast cancer to patterns of gene expression and molecular signatures. R and BioConductor packages are used to load, normalize and analyze the data. The present work was validated by showing that the results of the original paper can be reproduced using GridR.

The validation of GridR actually implements only a subset of the analysis steps presented in the original article, namely the principal component analysis (PCA). On the other hand steps related to the quality control of the arrays are shown, which were not presented in the original paper. Fig. 4 shows the plots related to those steps; in particular the plot illustrating the PCA is seen to be identical to that in [7].

The given scenario can be described from the technical point of view as follows: A researcher wants to perform interactive, grid-enabled data mining in the R environment. On his local machine (client) he develops algorithms using R as user interface, which he wishes to use on a clinical data set.

In clinical context, the data sets to be analyzed are usually so large (∼800 MB in the present, limited-scope, scenario) that a transfer to the client might be ineffective or not possible. Besides, execution machines in the grid environment might have bigger computational power as client machines. It is thus often more efficient to just ship the algorithm to the execution machine (the best being if the execution machine is the machine where the data is located, in order to minimize transfer time), execute it remotely on this machine and transfer the results back to the client. This is the strategy implicitly implemented in GridR.

5. Evaluation

The implementation of GridR was validated on the basis of the "Farmer scenario" [7] by implementing some typical analysis steps of a microarray experiment. In the present case, R was used in conjunction with the BioConductor packages affy, affyPLM and marray, which are specialized packages for microarray analysis, to build the individual modules of validation. Besides loading the expression data matrix and associated clinical data, those modules contribute in:

- Producing some figures required for the quality control of the chips (e.g. RNA degradation plots).
- Producing "MvA plots" to obtain an overall view of the fraction of differentially expressed genes.
- Using a variance filter to pick unique probeset per gene and performing a principal component analysis, to verify that samples with similar subtypes group together.
- Extracting symbols of genes most correlated to molecular markers relevant to the analysis (androgen receptor, AR, and estrogen receptor, ESR1).

The analysis steps are wrapped into functions for remote execution with GridR. Technically, the evaluation for computing the scenario with GridR in the ACGT testbed involves the GridR client at client side, a GRMS-Server installation from the Gridge toolkit [17] on a central machine in the grid environment that is responsible for resource management and orchestration of the execution machines, and the GridR R environment as well as the packages needed for the specific scenario installed on the execution machines. In the following the process of executing a single R function in such a grid environment is described. The different steps of execution are briefly shown in Fig. 4 (see also Fig. 3 for a view from the R client). (See Fig. 5.)

- *Function loading.* The GridR functions are loaded from the GridR package into the workspace of the R client.
- *Grid initialization.* The grid environment is initialized by calling the function grid.init. This function sets all needed settings for the client side components of the resource management systems to contact (e.g., including the information on the Myproxy certificate in case GT4 or Gridge are used as grid middleware) as well as some temporary directories to use on the remote hosts that will be taken as execution machine. To avoid the specification of all setting on each submission once again a configuration file can be used.
- *Code writing.* The R code which is to be executed in the grid is written and wrapped as single R function in the local R environment.
- *Grid submission.* The grid.apply function is called, which launches the process of submission. At first, the function to be executed in the grid and the needed parameters are written into a file (uniqueID-fx). Then the R script which is actually executed on the remote machine is generated (uniqueID-RemoteScript.R).Next an R file is created, which specifies the

```
>
> library(GridR)
> grid.init()
> grid.apply("y",sum,1:100,check=FALSE)
> y
Error in function (...)  : Object y has write lock from grid function
> y
Grid job finished, result written to variable y
> y
[1] 5050
>
```

Fig. 3. Simple GridR example.

Fig. 4. Farmer scenario plots.

"workflow" which is performed on the client side (uniqueID-LocalScript.R). After that, the R client executes this file, which results in connecting to the ACGT services including the Gridge components responsible for data and resource management. During the GridR stage-in phase, all files needed for the job submission (uniqueID-fx and uniqueID-RemoteScript.R) are uploaded to the grid and a GRMS job is generated. The GRMS system then takes care of staging-in files to the execution machine, launching the processing and managing the handling of the results. During the remote execution the created R script (uniqueID-RemoteScript.R) is executed on the remote machine, which reads the parameters and the function from uniqueID-fx, executes $y = f.x/$ and writes the result or any errors into a file (uniqueID-y.dat).

- *Waiting for result.* There are two ways of waiting for the result, a blocking one and a non-blocking one (specified with wait=TRUE or FALSE). While the remote execution is active and the R client waits for result (by checking if the file y.dat is created) the variable y is locked or if the blocking mode is used, R is blocked until the result is available.
- *Result processing.* When the result file (uniqueID-y.dat) was created on the remote machine it is, together with the other result files, transferred back to the client during the stage-out phase. In the GridR client, this file is loaded. The exit status is checked and – if the job was successful – the value is assigned to y and the variable is unlocked.

The present scenario is a proof of the concept by performing a shipping of the algorithm. With GridR, users can vastly improve their efficiency because they can easily create multiple jobs that are executed in parallel. In addition, the parallelization of calculation steps within a script can improve the scalability of an application even more. Even if a calculation does not run faster in GridR environment, the infrastructure accessible to GridR may render executable R scripts that may not be executable on typical workstations. For instance, the normalization of microarray sets with a few hundreds of chips may require up to several tens of