Introduction to Web Development with R

moving to the cloud...

Jeroen Ooms
http://www.stat.ucla.edu/~jeroen

UCLA Dept. of Statistics
Revolution Analytics

useR 2010, Gaithersburg, Maryland, USA
An example: stockplot

- Plots live data from Yahoo Finance.
- Uses a local MySQL database, some PHP.
- Intuitive Drag-n-drop interface.
What is a Web Application

Web Application Design

Phoenix Server

Extra slides
What is a client? A Browser
What is a client? Desktop software
What is a client? Your production process
Why Web Applications?

Convenient for the user:

- Making new tools available to a wide audience.
- Make applications that only require a browser.
- Cross-platform.

Server-based by design:

- Efficient use of resources.
- Easier to maintain.
- Integration with existing services/databases/etc.
Web application Setup

Server

- Calling R

Request/Response

Client

- Workstation
- Interface
Stateful or Stateless

Stateful R session:
- Efficient for multiple operations on the same workspace.
- Essential for big data.
- What to do with parallel requests?
- When to timeout sessions?

Stateless R sessions:
- Every request gets a new empty R session.
- Session is killed after operation finishes.
- Parallel requests no problem.
- Simulate statefulness by saving/loading workspaces.
Resources and Scalability

How to prevent overloading your servers?

- Limit memory per R session?
- Limit CPU time per R session?
- Limits per user or per request?
- Prevent DoS (1000 requests/sec).
- Load balancing, CPU distribution.
Security

How to prevent users from abusing your servers?

- R interacts freely with the system shell, which can be abused.
- What is the trust relationship with the user?
- Allow free code execution or only run predefined parameterized scripts?
- If only scripts, still watch out for code injection.
- Public (anonymous) privileges vs authorized privileges.
- Sandboxing users?
Error Catching

What to do with R errors?

- It is often useful to feed errors back to the client.
- Catch the errors in your R scripts.
- Always return a 'success' property in your response.

```xml
<response>
  <success>true</success>
  <results>// here the results</results>
</response>

OR:

```xml
<response>
  <success>false</success>
  <error>line 1 did not have 8 elements</error>
</response>
```
Graphics rendering

At the server

- use R’s graphics device
- plot to PNG, PDF, SVG, etc
- Easy, fast, pretty plots
- Limited by R’s graphics, no interaction, etc
- [example: sales]

At the client

- use R only for ’numbercrunching’
- return just data, no figure.
- render a plot on the client.
- more work, more eye candy.
- [example: BA]
General Advice

- Use R for calculations and plots. Do not generate HTML in R.
- Separate statistical/R layer from data layer, presentation layer, etc.
- Use a CMS or Web Development framework for UI stuff.
- Use R semantically. Think about Input/Output of your R scripts in terms of the statistical model.
- Standardize your R services. They should be client-independent.
- Make XML or JSON interfaces to your R services.
Connecting R

Low level tools:

- RScript - execute R scripts from the shell (stateless)
- RApache - execute R scripts from Apache httpd (stateless)
- Rserve - stateful R session with socket access.
- py2R - call R locally from python.
- JRI - call R locally from Java.

Problems:

- You need to be both web developer and R programmer to use any of these.
- A lot of work is required to get some simple functionality.
- Little control over resources, security, scalability.
Phoenix Server: a high level solution

Phoenix (codename) server:

- Framework for R web development by Revolution Analytics.
- Commercial and Academic (free) licences.
- Can be hosted locally or ‘in the cloud’.
- Scalability \{1..n\} R Servers with load balancing.
- Runs on Linux and Windows.

Our Goals:

- Keywords: Scalable, Standardized, Secure.
- Easily add R functionality to any application.
- Separates the R/statistical programming from the web development.
Phoenix Server: features

- RESTful API
- Management Console
- Standardized XML/JSON Interfaces.
- JSON/XML object encoding.
- Stateful and Stateless code execution.
- Deploy R scripts without any programming.
- Open source clients available for Java, .NET, PHP, Javascript, Excel, etc.
**Management Console**

R Scripts are blocks of R code designed to perform a specific function that can be exposed through the Phoenix R Web services API. Share the details of a script with developers by copying the URL to the developer summary page. Simply click Shareable Link to see the details of the developer summary page.

<table>
<thead>
<tr>
<th>Name</th>
<th>Annotation</th>
<th>Developer Summary</th>
<th>Enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Assignment</td>
<td>Test simple assignment only.</td>
<td>Shareable Link</td>
<td>True</td>
</tr>
<tr>
<td>Simple Busy Execute</td>
<td>Test long lived script.</td>
<td>Shareable Link</td>
<td>True</td>
</tr>
<tr>
<td>stockplot-png</td>
<td>Stockplot script with PNG output.</td>
<td>Shareable Link</td>
<td>True</td>
</tr>
<tr>
<td>stockplot-pdf</td>
<td>Stockplot script with pdf output.</td>
<td>Shareable Link</td>
<td>True</td>
</tr>
<tr>
<td>stockplot-ws</td>
<td>Stockplot script to export multiple plots to a single PDF.</td>
<td>Shareable Link</td>
<td>True</td>
</tr>
<tr>
<td>testscript</td>
<td>jeroens testscript.</td>
<td>Shareable Link</td>
<td>True</td>
</tr>
<tr>
<td>gg-guessdata</td>
<td>ggplot2 guessscript.</td>
<td>Shareable Link</td>
<td>True</td>
</tr>
<tr>
<td>Im - david</td>
<td>df = dataframe x = dependent variable y = independent variable, summary = regression, string.</td>
<td>Shareable Link</td>
<td>True</td>
</tr>
<tr>
<td>summary - david</td>
<td>Test</td>
<td>Shareable Link</td>
<td>True</td>
</tr>
<tr>
<td>gg_plotsvg</td>
<td>plot svg plots.</td>
<td>Shareable Link</td>
<td>True</td>
</tr>
</tbody>
</table>
RESTful API

Every command is called from a http url
E.g. http://calc.company.org/r/session/login

- /r/session/login
- /r/session/create
- /r/session/file/upload
- /r/session/execute/script
- /r/session/object/list
- /r/session/object/save
- /r/project/save
- /r/session/close
- etc
XML/JSON interfaces

Example call:

POST /r/session/save
format=json&
session=LIVE-92d9c643-5620-40a1-8626-47ded19970cc&
descr=My Workspace 1

Example response:

"phoenix": {
   "response": {
      "success": true,
      "call": "/r/session/saveworkspace",
      "pobjects": {
         "My Workspace 1": {
            "value": "CBNRY-7f8fa254-cc5a-4f77-a9d9-3b873be1bad3"
         }
      },
      "session": "LIVE-92d9c643-5620-40a1-8626-47ded19970cc"
   }
}"
XML/JSON Data-object Encoding

"myDataFrame" : {
    "type": "dataframe",
    "value": {
        "age": {
            "type": "vector",
            "value": [13, 15, 16]
        },
        "gender": {
            "type": "factor",
            "value": ["female", "male", "female"]
        }
    }
}
Stateful and Stateless code execution

Execute an R script that exists on the server.

Interfaces:
- /r/script/execute
- /r/session/execute/script

Features:
- Execute an R script that exists on the server.
- Either stateless or stateful.
- Parameterize by pushing R objects before execution.
- Retrieve encoded objects or files after execution.
Privilege Roles

Some built-in standard roles (customizable):

- Administrator: Deploy R-Scripts, Manage users, etc
- Unrestricted user: Use entire API.
- Restricted user: Use entire API, except for custom code execution.
- Public: permits the execution of stateless scripts anonymously.
Open Source Client Libraries

jPhoeni (Java) client Lib

```java
String phoenixUrl = "http://www.thecloud.com/phoenix/";

PClient pClient = PClientFactory.createClient(phoenixUrl);
pClient.login(new PBasicAuthentication("testuser", "password"));

PSession pSession = pClient.createSession();
pSession.executeCode("x <- rnorm(1000);");
pSession.executeScript("predict-stocks", ...);

pSession.closeSession();
pClient.release();
```
Open Source Client Libraries

phpPhoenix client lib:

```php
$webSession = WebSession::getInstance();

$client = new PhoenixClient::createHttpClient(PHOENIX_URL, $webSession);

$client->login(new PhoenixBasicAuthentication(USERNAME, PASSWORD));

$session = $client->createSession('calculate_average_session');

$phoenixExecution = $session->executeCode('myvar <- rnorm(100)', 'myvar');

$robjects = $phoenixExecution->getRObjects();
```
phoenixJS client library (sales example)

```javascript
button.onClick = function(){
    pExecuteScript({
        scriptname: 'plotsales-png',
        inputs: plotparams,
        files: ['salesplot.png'],
        mask: Ext.getCmp('plotpanel').getLayoutTarget(),
        success: function(robjects,files){
            insertPlot(files['salesplot.png'].value);
        }
    });
}
```
Example: sales forecasting
Finally, dinner!

Thank you for your attention.

- http://www.stat.ucla.edu/~jeroen
- http://www.revolutionanalytics.com
Object Encoding: XML

<myModel>
  <family>Gaussian</family>
  <deviance>3569.23</deviance>
  <coefficients>
    <coef>
      <name>Intercept</name>
      <value>5.69</value>
    </coef>
    <coef>
      <name>Age</name>
      <value>0.36</value>
    </coef>
    <coef>
      <name>Gender</name>
      <value>2.54</value>
    </coef>
  </coefficients>
</myModel>
Object Encoding: JSON

{"myModel": {
    "family": "Gaussian",
    "deviance": 3569.23,
    "coefficients":
        [ {"Intercept": 5.69}, {"Age": 0.36}, {"Gender": 2.54} ]
}

Or for example a dataframe:

{"myData": {
    "Age": [9,8,12,6,7,8,9,8,10,11,9,6,8],
    "Gender": ["M","F","F","M","F","M","F","M","F","M","F","F"],
    "Treatment": [1,0,0,1,1,1,0,0,0,1,1,1,0]
}
}
Examples: sales forecasting
Examples: yeroon.net/ggplot2

- Exploratory graphical analysis and education of stats and R.
- Upload data or import spreadsheet from Google Docs.
- Add graphical layers and map/set any aesthetics.
- Export to PNG, PDF, SVG
Examples: yeroon.net/lme4

- Online random effects / multilevel modeling.
- Upload data, maintain several models.
- Export PDF report (Latex)
Examples: stockplot

- Plots live data from Yahoo Finance.
- Uses a local MySQL database, some PHP.
- Intuitive Drag-n-drop interface.