

R在应用领域的扩展

——JAVA开发和最优化

李舰

目录

- 简介
- R与JAVA的整合
- R在最优化中的应用

R的优势

- 语言
 - 矩阵计算
 - 科学计算环境
 - 面向对象
- 强大的社区支持
 - 丰富的Package
 - 广大的支持者

以科学计算为中心的IT工具需求

- 工具的终极目标: Thinking in R
- 一个小例子: Excel
 - Office自动化程序, VBA;
 - 文件操作, FSO;
 - 系统操作, Shell;
 - 数据库操作, ODBC;
 - 复杂计算, R;
 - SAP, ABAP接口。。。

R的扩展

不足	扩展
解释型语言效率较低	调用外部函数（C/Fortran等）
处理海量数据困难	数据库的支持
缺少某些先进语言的特性	与其他语言的集成（C++，JAVA等）
不适合大型系统的开发	与JAVA的整合
最优化计算偏弱	GLPK

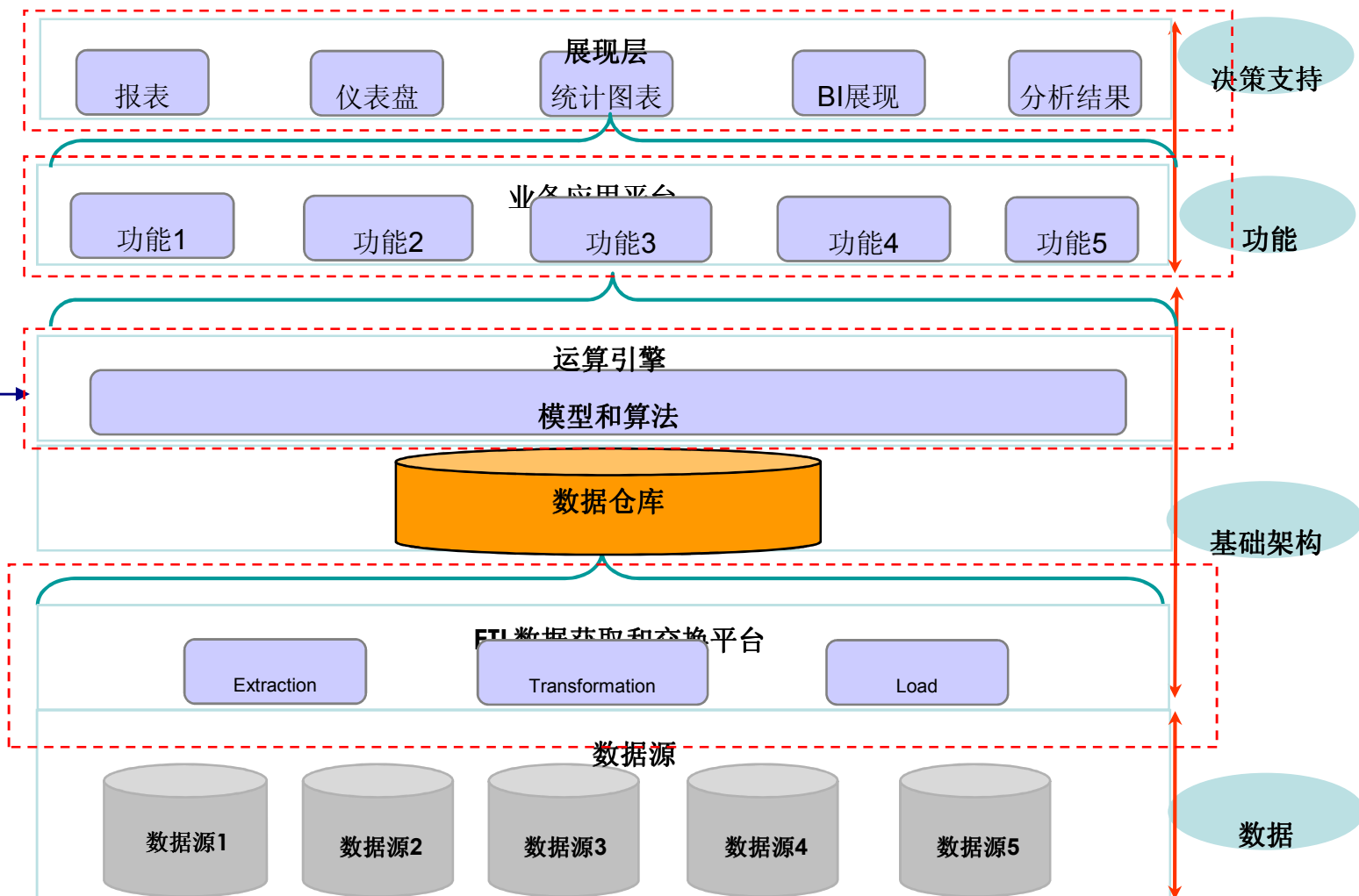
目录

- 简介
- R与JAVA的整合
- R在最优化中的应用

面向对象的开发

- 面向过程与面向对象
- 面向对象的优点
 - 符合人类正常思维
 - 封装、易维护，适合大型系统
 - 多态

R在分析型系统中的位置



R与JAVA的连接方式

- Rserve
 - <http://www.rforge.net/Rserve/>
 - TCP/IP 服务器
- rJava
 - <http://www.rforge.net/rJava/>
 - 通过JNI 的方式从底层实现互联
- JRI
 - 目前已成为rJava的子项目

Rserve的安装和配置

- 安装Server
 - 下载并安装最新版的Rserve 0.6-0
 - 可以直接使用`install.packages("Rserve")`
- 配置客户端
 - 下载REngine.jar和RserveEngine.jar
 - 在JAVA中导入Jar包
- 加载
 - `library(Rserve)`
 - `Rserve()`

JRI的安装和配置

- 安装rJava
 - 最新版本的JRI包含在rJava中
 - `install.packages("rJava")`
- 导入相关jar包
 - 下载JRIEngine.jar、REngine.jar和JRI.jar
 - 导入
- 设置环境变量
 - 在PATH中编辑JRI.jar和r.dll所在文件夹

应用实例 (Rserve)

```
import org.rosuda.REngine.REXP;  
import org.rosuda.REngine.REXPMismatchException;  
import org.rosuda.REngine.REngineException;  
import org.rosuda.REngine.Rserve.RConnection;  
import org.rosuda.REngine.Rserve.RserveException;  
  
public class rtest3 {  
  
    public static void main(String[] args) throws REXPMismatchE  
        // TODO Auto-generated method stub  
        RConnection c = new RConnection();  
        REXP x = c.eval("R.version.string");  
        System.out.println(x.asString());  
        REXP z = c.eval("library(survival)");  
        REXP y = c.eval("length(levels(strata(5)))");  
        System.out.println(y.asString());  
    }  
}
```

应用实例 (JRI)

```
-
7 import org.rosuda.JRI.Rengine;
8 import org.rosuda.JRI.REXP;
9 import org.rosuda.JRI.RList;
10 import org.rosuda.JRI.RVector;
11 import org.rosuda.JRI.RMainLoopCallbacks;
12
13 public class rtest {
14     public static void main(String[] args) {
15         Rengine re=new Rengine(args, false, new TextConsole());
16         System.out.println("Rengine created, waiting for R");
17         try {
18             REXP x;
19             re.eval("data(iris)", false);
20             System.out.println(x=re.eval("iris"));
21             // generic vectors are RVector to accomodate name
22             RVector v = x.asVector();
23             if (v.getNames() != null) {
24                 System.out.println("has names:");
25                 for (Enumeration e = v.getNames().elements())
26                     System.out.println(e.nextElement());
27             }
28         }
29     }
30 }
```

目录

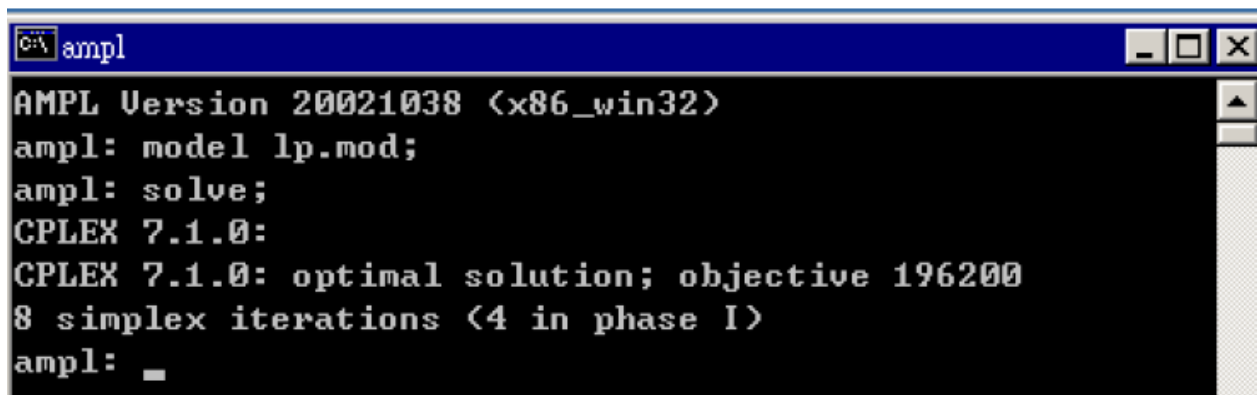
- 简介
- R与JAVA的整合
- R在最优化中的应用

最优化简介

- 运筹学
- 优化方法
 - 线性规划、整数规划、二次规划
 - 无约束最优化（函数法、一维搜索……）
 - 约束规划（拉格朗日乘子法、罚函数法）
 - 动态规划
 - 遗传算法
 - ……
- 优化工具
 - Lingo/Lindo
 - CPLEX
 - Matlab
 - ……

优化编程语言 AMPL

- AMPL
 - A Mathematical Programming Language
 - 能够快速而灵活地描述复杂的优化模型
 - 贝尔实验室开发，目前由Ilog经销维护
- Ilog CPLEX
 - Ilog将AMPL建模语言以及ILOG CPLEX进行了结合
 - AMPL经过解释后由Ilog维护的Solver进行求解
- 一个例子



```
C:\> ampl
AMPL Version 20021038 (x86_win32)
ampl: model lp.mod;
ampl: solve;
CPLEX 7.1.0:
CPLEX 7.1.0: optimal solution; objective 196200
8 simplex iterations (4 in phase I)
ampl: _
```


GLPK

- GNU Linear Programming Kit
 - GNU的一个项目，处理规划问题的求解
 - 支持GMPL语言，是AMPL的一个子集
- RGLPK
 - 用C编写的一个API
 - 作为R中的Package发布
 - 为GLPK的Solver提供接口
- 安装
 - `install.packages("glpk")`

导入GMPL模型求解

- 导入模型文件.mod和数据文件.dat
 - `lpx_read_model(modelfile, datafile, outputfile)`
- 对模型求解
 - `lpx_simplex(lp)`

```
> lp <- lpx_read_model("D:/Projects/2009-10-12 Optimization/R Data/GLPK/transport.mod")
Reading model section from D:/Projects/2009-10-12 Optimization/R Data/GLPK/transport.mod...
Reading data section from D:/Projects/2009-10-12 Optimization/R Data/GLPK/transport.mod...
62 lines were read
Generating cost...
Generating supply...
Generating demand...
Model has been successfully generated
> lpx_simplex(lp)
      0:  objval =  0.000000000e+000   infeas =  1.000000000e+000 (0)
      4:  objval =  1.561500000e+002   infeas =  0.000000000e+000 (0)
*      4:  objval =  1.561500000e+002   infeas =  0.000000000e+000 (0)
*      5:  objval =  1.536750000e+002   infeas =  0.000000000e+000 (0)
OPTIMAL SOLUTION FOUND
[1] 200
> |
```

利用API求解

```
70 print ("USING API")
71 canneries <- c("Seattle", "San-Diego")
72 capacity <- c(350, 600)
73 markets <- c("New-York", "Chicago", "Topeka")
74 demand <- c(325, 300, 275)
75 distance <- c(2.5, 2.5, 1.7, 1.8, 1.8, 1.4)
76 dim(distance) <- c(2, 3)
77 freight <- 90
83 lpi <- lpx_create_prob()
84 lpx_set_prob_name(lpi, "cannery API")
85 lpx_set_obj_name(lpi, "Total Cost")
86 lpx_set_obj_dir(lpi, LPX_MIN)
87
88
.....

140
141 for (i in 1:numcanneries){
142   #supply constraints
143   cannerysupplyrow = numcols + (i-1)*nummarkets
144   for (j in 1:nummarkets){
145     ia[cannerysupplyrow+j] <- (i+1);
146     ja[cannerysupplyrow+j] <- (i-1)+numcanneries *(j-1)+1;
147     ar[cannerysupplyrow+j] <- 1;
148   }
149   #demand constraints
150   marketdemandrow = numcols+numcanneries * nummarkets
151   for (j in 1:nummarkets){
152     colnum <- (i-1)*nummarkets+j
153     ia[marketdemandrow + colnum] <- numcanneries+j+1;
154     ja[marketdemandrow + colnum] <- colnum;
155     ar[marketdemandrow + colnum] <- 1;
156   }
157 }
158 lpx_load_matrix(lpi, length(ia), ia, ja, ar);
```

为什么使用R做优化

- 复杂问题求解
 - 标准的优化模型只是分解后的子问题
- 随机类问题的处理
 - 统计计算语言具有先天的优势
- 编写复杂算法
 - 矩阵运算，调用C / Fortran
- 用作运算引擎
 - 完整的科学计算环境