R语言与金融大数据应用 - 张丹

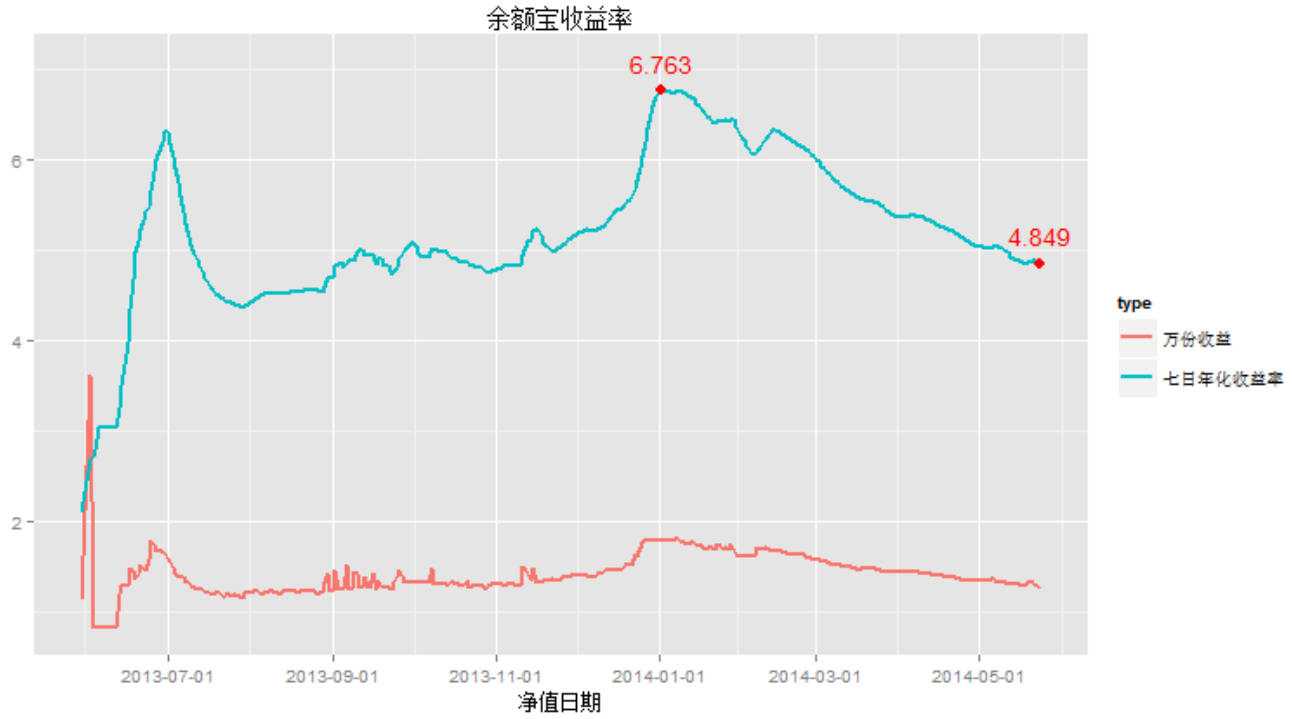
2014.05.07

目录

- 项目背景：金融知识介绍
- 需求分析：逆回购套利
- 算法模型：R语法算法
- 架构设计：RHive系统架构
- 程序开发：Hive数据处理，R语言算法

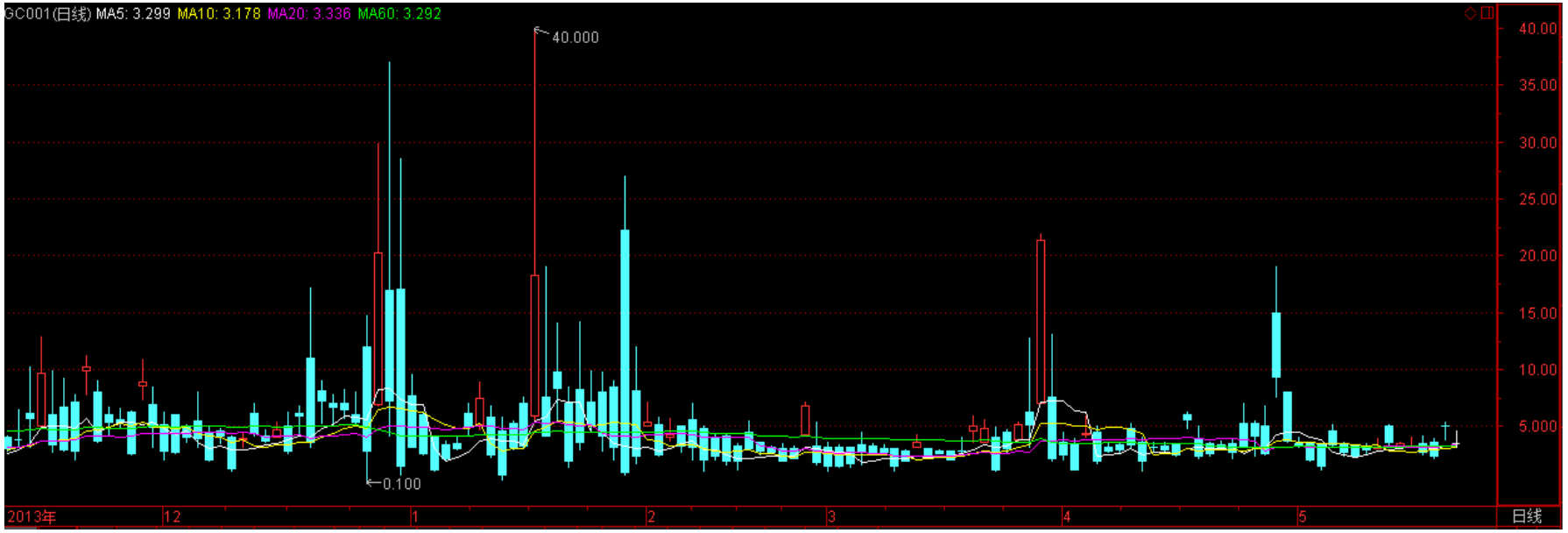
项目背景：金融知识介绍

- 支付宝推出的“余额宝”赚尽无数人的眼球，同时也吸引的大量的小额资金进入。余额宝把用户的散钱利息提高到了年化收益率5.0%左右，最高时7日年化收益率达到6.76%，比起银行活期存储存款0.3%左右高出太多了，正撼动着银行躺着赚钱的地位。



项目背景：逆回购 GC001

- 在金融市场，如果想获得年化收益率4%-5%左右也并非难事，通过“逆回购”一样可以。一旦遇到货币紧张时(银行缺钱)，更可达到30%一天隔夜回购利率。
- 我们就可以美美地在家里数钱了！！



项目背景：GC001 2014-01-17

- 100W * 30% / 365 - 10= 811.9178

项目背景：逆回购交易

- **国债回购交易(深)**，也称国债的现货交易，是指进行国债回购交易的市场，即卖出债券，并附加条件，于一定期间后，以预定的价格和收益，由最初出售者买回债券。还包括买入债券，于一定期间后，以预定的条件和价格，再卖给最初出售者的反回购。

- **新质押式回购(沪)，**是交易双方以债券为权利质押所进行的短期资金融通业务。在质押式回购交易中，资金融入方（正回购方）在将债券出质给资金融出方（逆回购方）融入资金的同时，双方约定在将来某一日期由正回购方向逆回购方返还本金和按约定回购利率计算的利息，逆回购方向正回购方返还原出质债券。

- 通俗来讲，就是你(A)把钱借给别人(B)，到期时，B按照约定利息，还给你(A)本资+利息。逆回购本身是无风险的。(操作与银行储蓄存款类似)

项目背景：逆回购种类

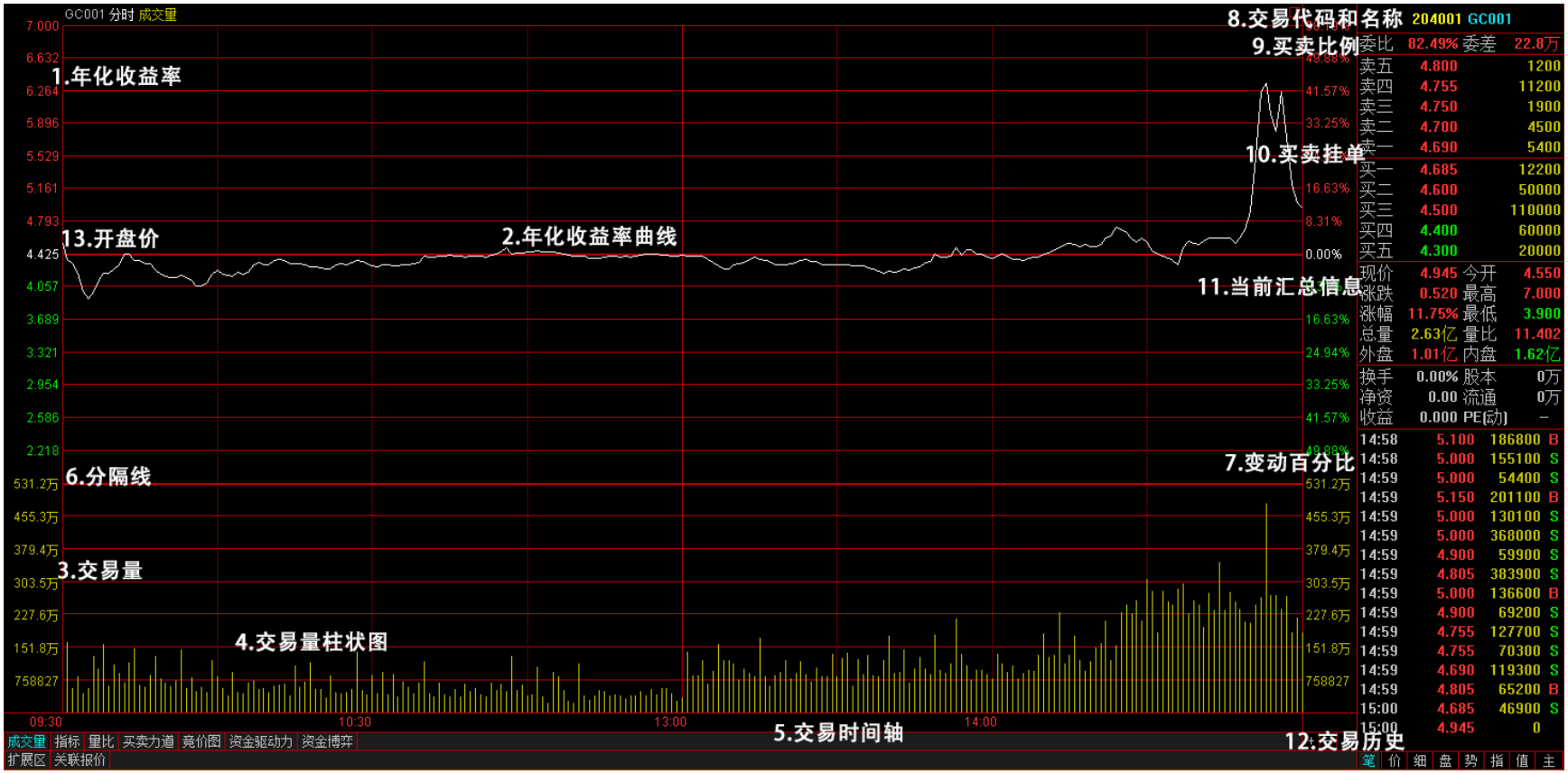
债券回购类型	回购天数	回购代码	当前可卖出年化回购利率
上海新质押式回购1天	1	204001	4.685
上海新质押式回购2天	2	204002	1.95
上海新质押式回购3天	3	204003	1.3
上海新质押式回购4天	4	204004	1.02
上海新质押式回购7天	7	204007	3.805
上海新质押式回购14天	14	204014	4.07
上海新质押式回购28天	28	204028	4.2
上海新质押式回购91天	91	204091	4.62
上海新质押式回购182天	182	204182	4.51
深圳国债回购1天	1	131810	3.2
深圳国债回购2天	2	131811	0.2
深圳国债回购3天	3	131800	0
深圳国债回购4天	4	131809	2
深圳国债回购7天	7	131801	3.5
深圳国债回购14天	14	131802	3.6
深圳国债回购28天	28	131803	2.85
深圳国债回购91天	91	131805	4.22
深圳国债回购192天	182	131806	4.101

项目背景：逆回购交易规则

- 上交所9支，质押式回购，资金门槛10万元，每笔为10万的整数倍，分别对应该1天，2天，3天，4天，7天，14天，28天，91天，182天
- 深交所9支，国债回购，资金门槛1千元，每笔为1000的整数倍，分别对应该1天，2天，3天，4天，7天，14天，28天，91天，182天

- 此操作为无风险操作，完全就是吃利息。

项目背景：逆回购分时图


GC001 分时 成交量
1.年化收益率
2.年化收益率曲线
3.交易量
4.交易量柱状图
5.交易时间轴
6.分隔线
7.变动百分比
8.交易代码和名称 204001 GC001
9.买卖比例
10.买卖挂单
11.当前汇总信息
12.交易历史
13.开盘价
委比 82.49% 委差 22.8万
卖五 4.800 1200
卖四 4.755 11200
卖三 4.750 1900
卖二 4.700 4500
卖一 4.690 5400
买一 4.685 12200
买二 4.600 50000
买三 4.500 110000
买四 4.400 60000
买五 4.300 20000
现价 4.945 今开 4.550
涨跌 0.520 最高 7.000
涨幅 11.75% 最低 3.900
总量 2.63亿 量比 11.402
外盘 1.01亿 内盘 1.62亿
换手 0.00% 股本 0万
净资 0.00 流通 0万
收益 0.000 PE(动) -
14:58 5.100 186800 B
14:58 5.000 155100 S
14:59 5.000 54400 S
14:59 5.150 201100 B
14:59 5.000 130100 S
14:59 5.000 368000 S
14:59 4.900 59900 S
14:59 4.805 383900 S
14:59 5.000 136600 B
14:59 4.900 69200 S
14:59 4.755 127700 S
14:59 4.755 70300 S
14:59 4.690 119300 S
14:59 4.805 65200 B
15:00 4.685 46900 S
15:00 4.945 0
09:30
10:30
13:00
14:00
成交量 指标 量比 买卖力道 竞价图 资金驱动力 资金博弈
扩展区 关联报价
笔 价 细 盘 势 指 值 主

项目背景：逆回购举例

- 进行操作：上交所发行的”上海新质抵押式回购1天”，借款时间”1天”，回购代码”204001”，资金”10万”元，
  - 1个自然天后利息=100000*4.685%/365=12.83562。
  - 回款本金+利息=100000+12.83562=100012.83562

- 对“上海新质抵押式回购7天”操作，7个自然天后利息=100000*3.85%*5/365 – 1 =51.73973

- 一般我们最常操作的，就是1天和7天的逆回购。

需求分析：逆回购套利案例

- 出始条件：
  - 5天日内历史数据， 1万元本金，投放逆回购(深圳国债回购1天，代码131810)

- 业务需求：
  - 通过历史数据分析套利策略，发现当日高点，进行逆回购，赚取最高利息。

需求分析：逆回购套利KPI

- KPI指标：
  - 1天利息：最大

- 需求程序化：
  - 通过历史数据用程序设计套利策略，发现每日高点，建立算法模型。
  - 以算法模型为基础，时时计算当日交易数据，并投放逆回购。
  - 赚取最高利息

算法模型: 数据格式

- tradedate：交易日期
- tradetime：交易时间
- securityid：金融产品ID
- bidpx1：买一价
- bidsize1：买一量
- offerpx1：卖一价
- bidsize1：卖一量

```
> head(bidpx1)
  tradedate tradetime securityid bidpx1 bidsize1 offerpx1 offersize1
1  20130724    145004     131810  2.620     6960    2.630      13000
2  20130724    145101     131810  2.860    13880    2.890       6270
3  20130724    145128     131810  2.850   327400    2.851       1500
4  20130724    145143     131810  2.603    44630    2.800      10650
5  20130724    144831     131810  2.890    11400    3.000      77990
6  20130724    145222     131810  2.600  1071370    2.601      35750
```

算法模型: 数据描述

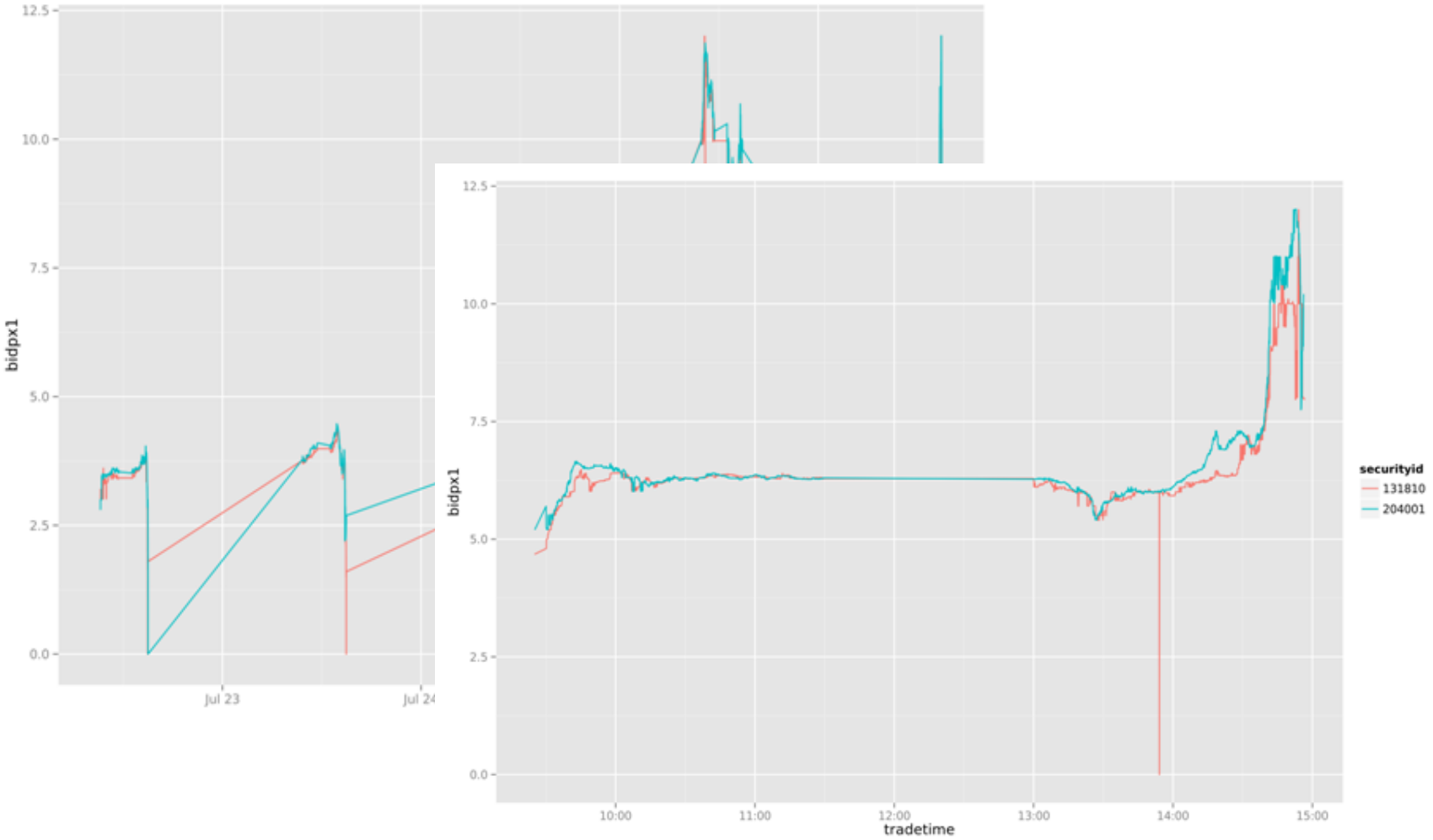
- 包括2支回购数据：代码131810，代码204001
- 包括5个交易日数据：20130722， 20130723， 20130724， 20130725， 20130726

```
> table(bidpx1$tradedate)

20130722 20130723 20130724 20130725 20130726
    6074     5991     5899     5346     6192
> table(bidpx1$securityid)

131810 204001
 17061  12441
```

算法模型: 发现131810与204001关系



算法模型:发现131810与204001关系

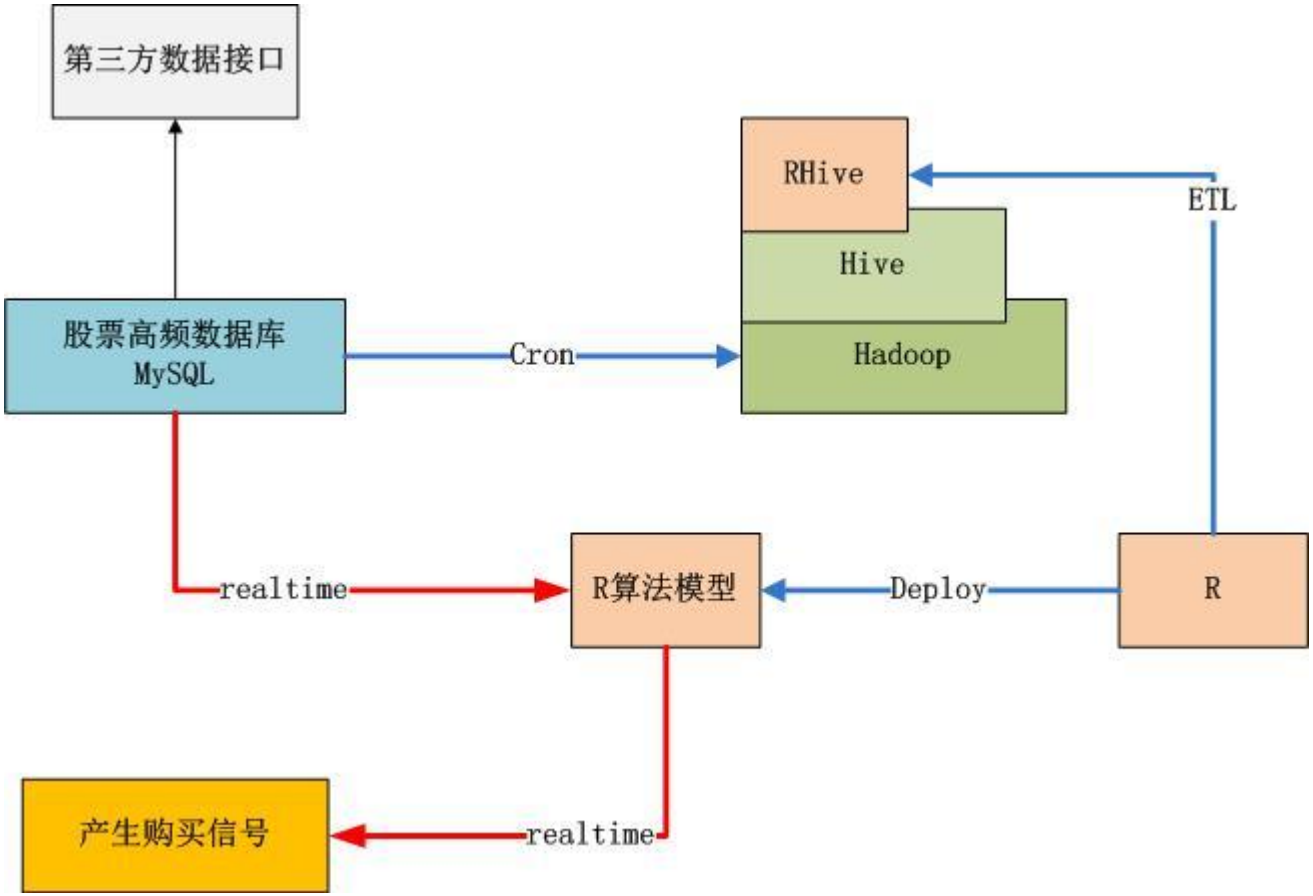
- 发现规律：
  - 131810会跟随204001的变化
  - 131810通常都比204001要低

算法模型: 建立算法模型

- 下面做一个简单的策略分析：
  - 通过204001变化，判断131810的卖点。

- 算法设计
  - 把131810和204001按每分钟标准化
  - 设置当131810和204001有交点的时候，提取卖出信号
  - 当后一个交点的卖一价格大于前一个交点的卖一价格10%以上，做为局部最优的卖出信号点

架构设计：R+Hive系统架构



架构设计：R+Hive系统架构

1. 交易时间，时时从第三方数据接口获得数据，存储在MySQL中
2. 每日收盘后，从MySQL导入到Hadoop系统
3. 通过Hive管理Hadoop的数据存储
4. 通过RHive实现R与Hive的通信接口
5. 在RHive中完成ETL
6. 通过R语言建立，算法模型，并部署
7. 交易时间，R算法模型时时处理交易数据，时时产生买卖信号
8. 根据买卖信号，通过交易接口完成逆回购的下单。

程序开发： RHive操作

```
#装载RHive
library(RHive)

#初始化
rhive.init()

#连接到Hive集群
> rhive.connect("c1.wtmart.com")
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/cos/toolkit/hive-0.9.0/lib/slf4j-log4j12-1.6.1.jar!/
SLF4J: Found binding in [jar:file:/home/cos/toolkit/hadoop-1.0.3/lib/slf4j-log4j12-1.4.3.jar
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.

#查看当前的表
> rhive.list.tables()
                tab_name
1             t_hft_day    //历史数据表
2             t_hft_tmp    //临时表
4 t_reverse_repurchase     //逆回购表
```

```
rhive.desc.table('t_reverse_repurchase')
     col_name data_type
1  tradedate    string
2  tradetime    string
3 securityid    string
4     bidpx1    double
5   bidsize1    double
6   offerpx1    double
7 offersize1    double
```

程序开发：查看所有股票历史数据分片

- 测试数据从20130627–20130726。

```
> rhive.query("SHOW PARTITIONS t_hft_day");
            partition
1  tradedate=20130627
2  tradedate=20130628
3  tradedate=20130701
4  tradedate=20130702
5  tradedate=20130703
6  tradedate=20130704
7  tradedate=20130705
8  tradedate=20130708
9  tradedate=20130709
10 tradedate=20130710
11 tradedate=20130712
12 tradedate=20130715
13 tradedate=20130716
14 tradedate=20130719
15 tradedate=20130722
16 tradedate=20130723
17 tradedate=20130724
18 tradedate=20130725
19 tradedate=20130726
```

程序开发：ETL数据提取

- 分为提取”上交所一天逆回购”(204001)，和”深交所一天逆回购”(131810)，从7月22日至7月26日的一周数据。

```
> rhive.drop.table("t_reverse_repurchase")
> rhive.query("CREATE TABLE t_reverse_repurchase AS SELECT tradedate,tradetime,securityid,bi
```

- 查看数据结果集

```
> rhive.query("SELECT securityid,count(1) FROM t_reverse_repurchase group by securityid");
  securityid  X_c1
1     131810 17061
2     204001 12441
```

程序开发：加载到R的内存中

```
> bidpx1<-rhive.query("SELECT securityid,concat(tradedate,tradetime) as tradetime,bidpx1 FR
[1] 29502

#查看数据
> head(bidpx1)
  securityid      tradetime bidpx1
1     131810 20130724145004  2.620
2     131810 20130724145101  2.860
3     131810 20130724145128  2.850
4     131810 20130724145143  2.603
5     131810 20130724144831  2.890
6     131810 20130724145222  2.600
```

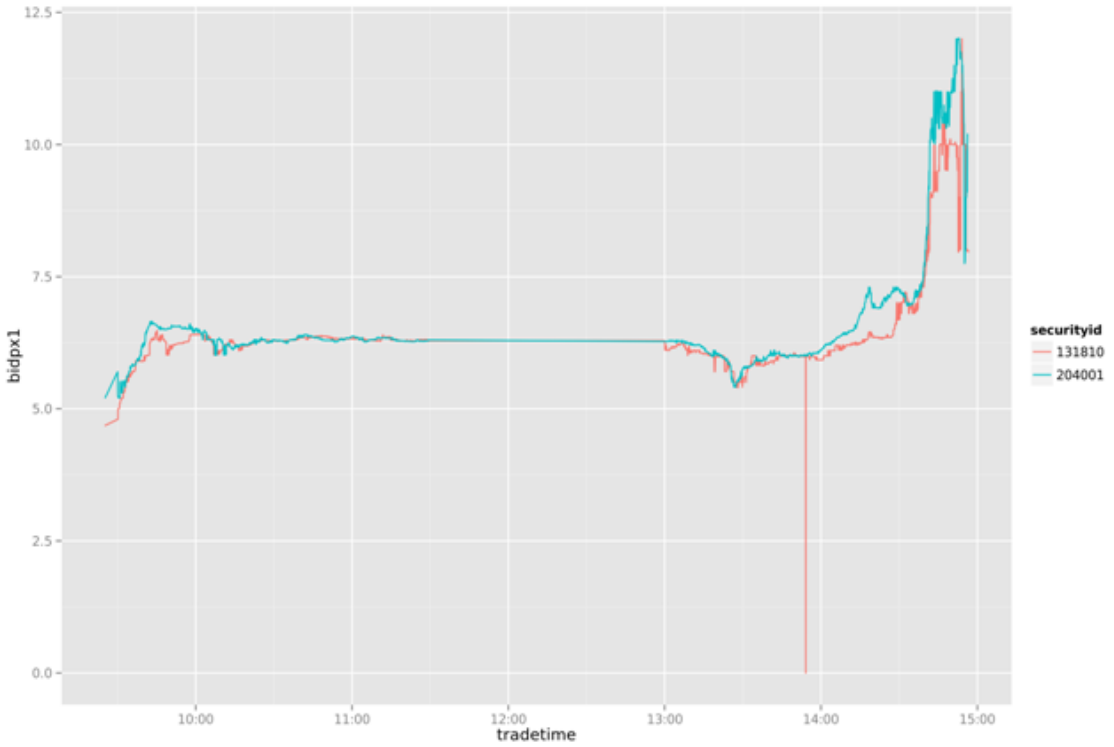
```
> summary(bidpx1)
   tradedate          tradetime       securityid       bidpx1          bidsize1          offerpx1        offersize1
 Min.   :20130722   Min.   : 91501   131810:17061   Min.   : 0.000   Min.   :      0   Min.   : 0.000   Min.   :     0.0
 1st Qu.:20130723   1st Qu.:102709   204001:12441   1st Qu.: 3.745   1st Qu.:  20472   1st Qu.: 3.750   1st Qu.:   642.5
 Median :20130724   Median :112208                  Median : 4.370   Median :  76500   Median : 4.380   Median :  3600.0
 Mean   :20130724   Mean   :119165                  Mean   : 5.516   Mean   : 177252   Mean   : 5.533   Mean   : 27220.5
 3rd Qu.:20130725   3rd Qu.:135222                  3rd Qu.: 6.328   3rd Qu.: 200000   3rd Qu.: 6.337   3rd Qu.: 19380.0
 Max.   :20130726   Max.   :150148                  Max.   :12.010   Max.   :4828080   Max.   :12.300   Max.   :2493970.0
```

程序开发：一周数据的走势

```
library(ggplot2)
g<-ggplot(data=bidpx1, aes(x=as.POSIXct(tradetime,format="%Y%m%d%H%M%S"), y=bidpx1))
g<-g+geom_line(aes(group=securityid,colour=securityid))
g<-g+xlab('tradetime')+ylab('bidpx1')
ggsave(g,file="01.png",width=12,height=8)
```

程序开发：一天数据的走势

```
bidpx1<-rhive.query("SELECT securityid,concat(tradedate,tradetime) as tradetime,bidpx1 FROM
g<-ggplot(data=bidpx1, aes(x=as.POSIXct(tradetime,format="%Y%m%d%H%M%S"), y=bidpx1))
g<-g+geom_line(aes(group=securityid,colour=securityid))
g<-g+xlab('tradetime')+ylab('bidpx1')
ggsave(g,file="02.png",width=12,height=8)
```



程序开发：建立算法模型 1

```
#把一周的数据加载到内存
bidpx1<-rhive.query(paste("SELECT securityid,tradedate,tradetime,bidpx1 FROM t_reverse_repur

#加载一天的数据并做ETL
oneDay<-function(date){
  d1<-bidpx1[which(bidpx1$tradedate==date),]
  d1$tradetime2<-round( as.numeric(as.character(d1$tradetime))/100)*100
  d1$tradetime2[which(d1$tradetime2<100000)]<-paste(0,d1$tradetime2[which(d1$tradetime2<1000
  d1$tradetime2[which(d1$tradetime2=='1e+05')]='100000'
  d1$tradetime2[which(d1$tradetime2=='096000')]='100000'
  d1$tradetime2[which(d1$tradetime2=='106000')]='110000'
  d1$tradetime2[which(d1$tradetime2=='126000')]='130000'
  d1$tradetime2[which(d1$tradetime2=='136000')]='140000'
  d1$tradetime2[which(d1$tradetime2=='146000')]='150000'
  d1
}

#通过均值标准化
meanScale<-function(d1){
  ddply(d1, .(securityid,tradetime2), summarize, bidpx1=mean(bidpx1))
}
```

程序开发：建立算法模型 2

```
#通过均值标准化
meanScale<-function(d1){
  ddply(d1, .(securityid,tradetime2), summarize, bidpx1=mean(bidpx1))
}

#找到要分析的点
findPoint<-function(a1,a2){
  #找到所有a1大于a2的点
  bigger_point<-function(a1,a2){
    idx<-c()
    for(i in intersect(a1$tradetime2,a2$tradetime2)){
      i1<-which(a1$tradetime2==i)
      i2<-which(a2$tradetime2==i)
      if(a1$bidpx1[i1]-a2$bidpx1[i2]>=-0.02){
        idx<-c(idx,i1)
      }
    }
    idx
  }

  #去掉连续的索引值
  remove_continuous_point<-function(idx){
    idx[-which(idx-c(NA,rev(rev(idx)[-1]))==1)]
  }
```

程序开发：建立算法模型 3

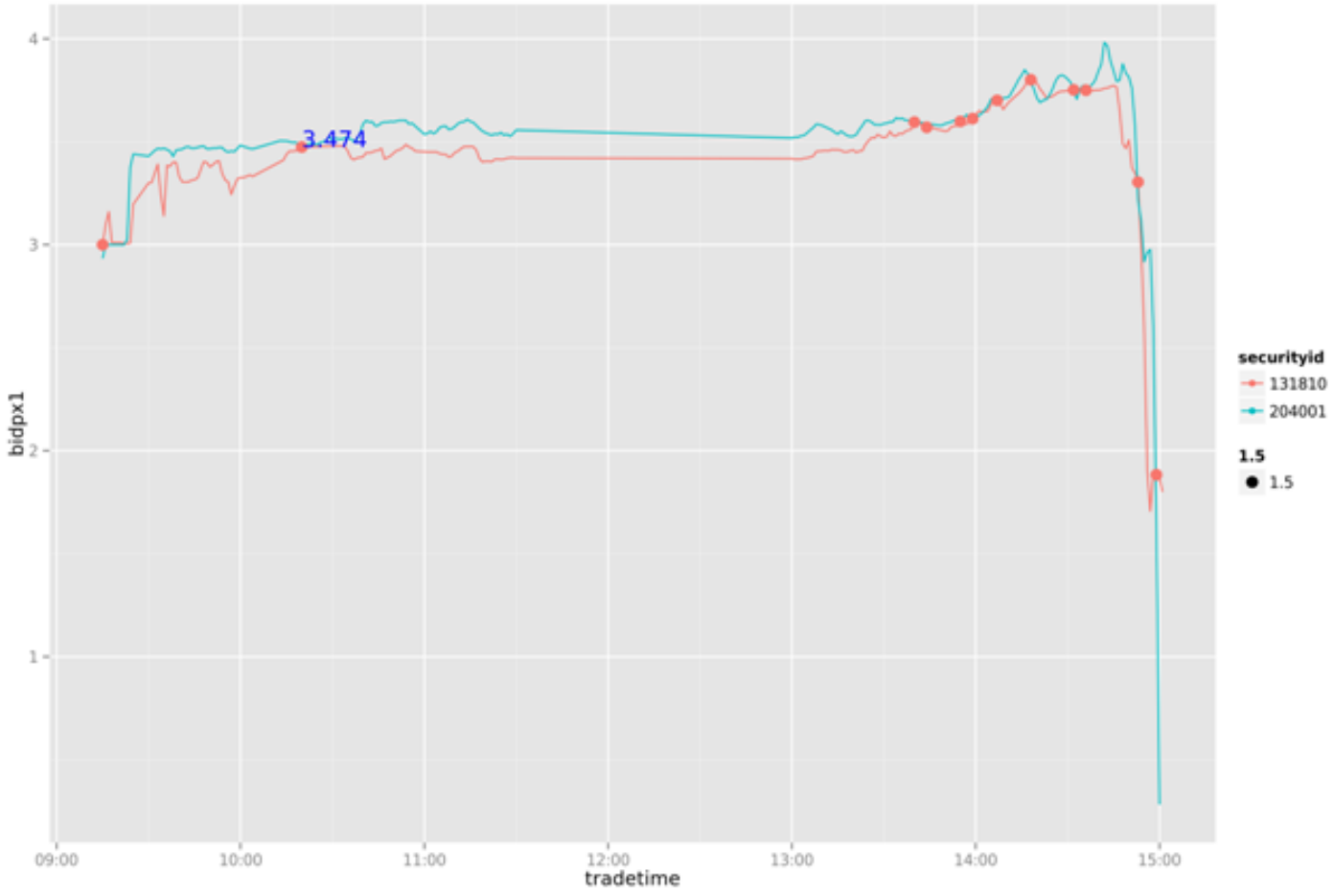
```
#发现局部最优卖出点
findOptimize<-function(d3){
  idx2<-which((d3$bidpx1-c(NA,rev(rev(d3$bidpx1)[-1])))/d3$bidpx1>0.1)
  if(length(idx2)<1)
    print("No Optimize point")
  d3[idx2,]
}

#画图查看结果
draw<-function(d2, d3, d4, date, png=FALSE){
  g<-ggplot(data=d2,  aes(x=strptime(paste(date, tradetime2, sep=""), format="%Y%m%d%H%M%S"),  y
  g<-g+geom_line(aes(group=securityid, colour=securityid))
  g<-g+geom_point(data=d3, aes(size=1.5, colour=securityid))   if(nrow(d4)>0){
    g<-g+geom_text(data=d4, aes(label= format(d4$bidpx1, digits=4)), colour="blue", hjust=0,  vj
  }
  g<-g+xlab('tradetime')+ylab('bidpx1')
  if(png){
    ggsave(g, file=paste(date, ".png", sep=""), width=12, height=8)
  }else{
    g
  }
}
```

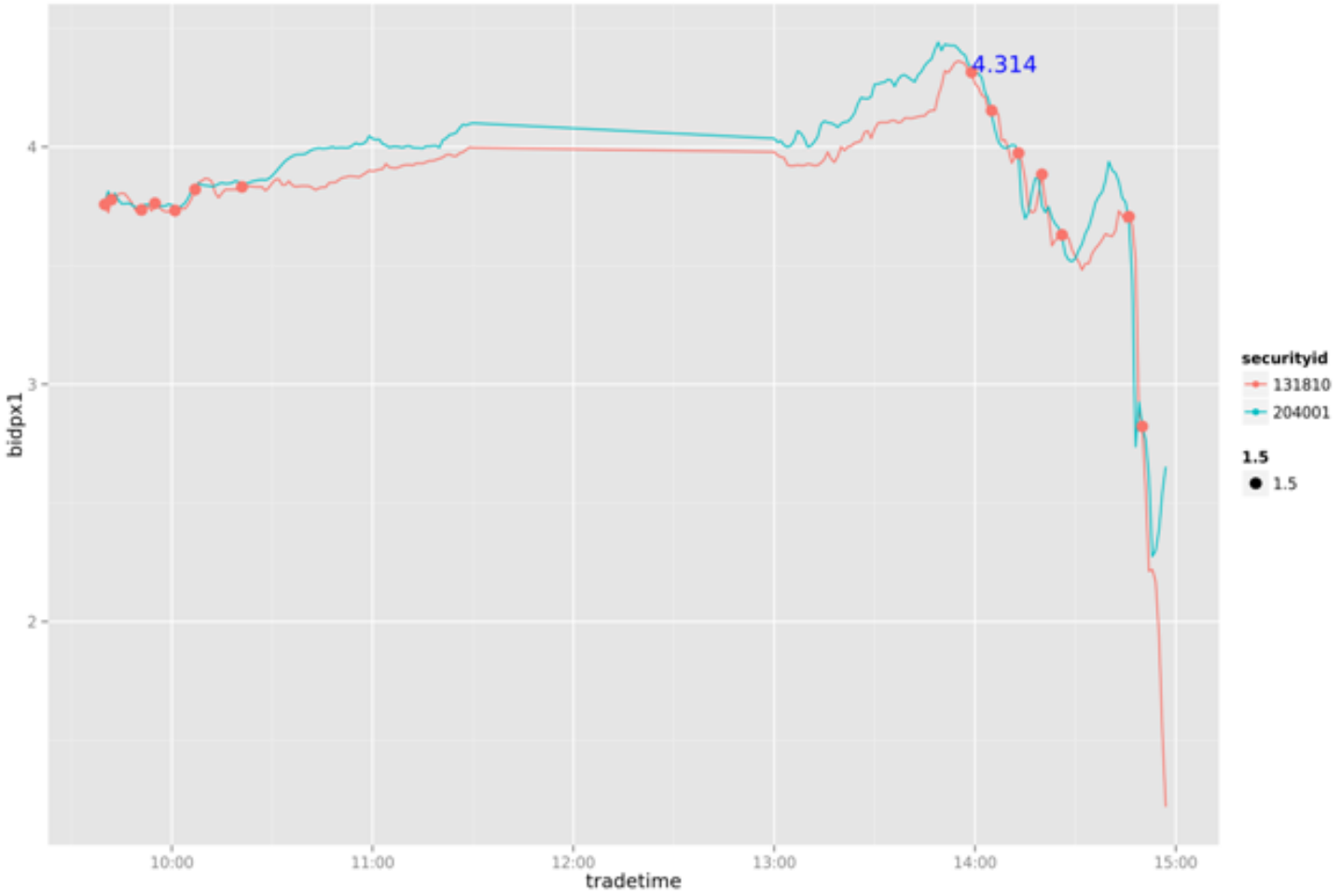
程序开发：运行算法

```
#执行策略封装
date<-20130722
d1<-oneDay(date)
d2<-meanScale(d1)
a1<-d2[which(d2$securityid==131810),]
a2<-d2[which(d2$securityid==204001),]
d3<-d2[findPoint(a1,a2),]
d4<-findOptimize(d3)
draw(d2,d3,d4,as.character(date),TRUE)
```

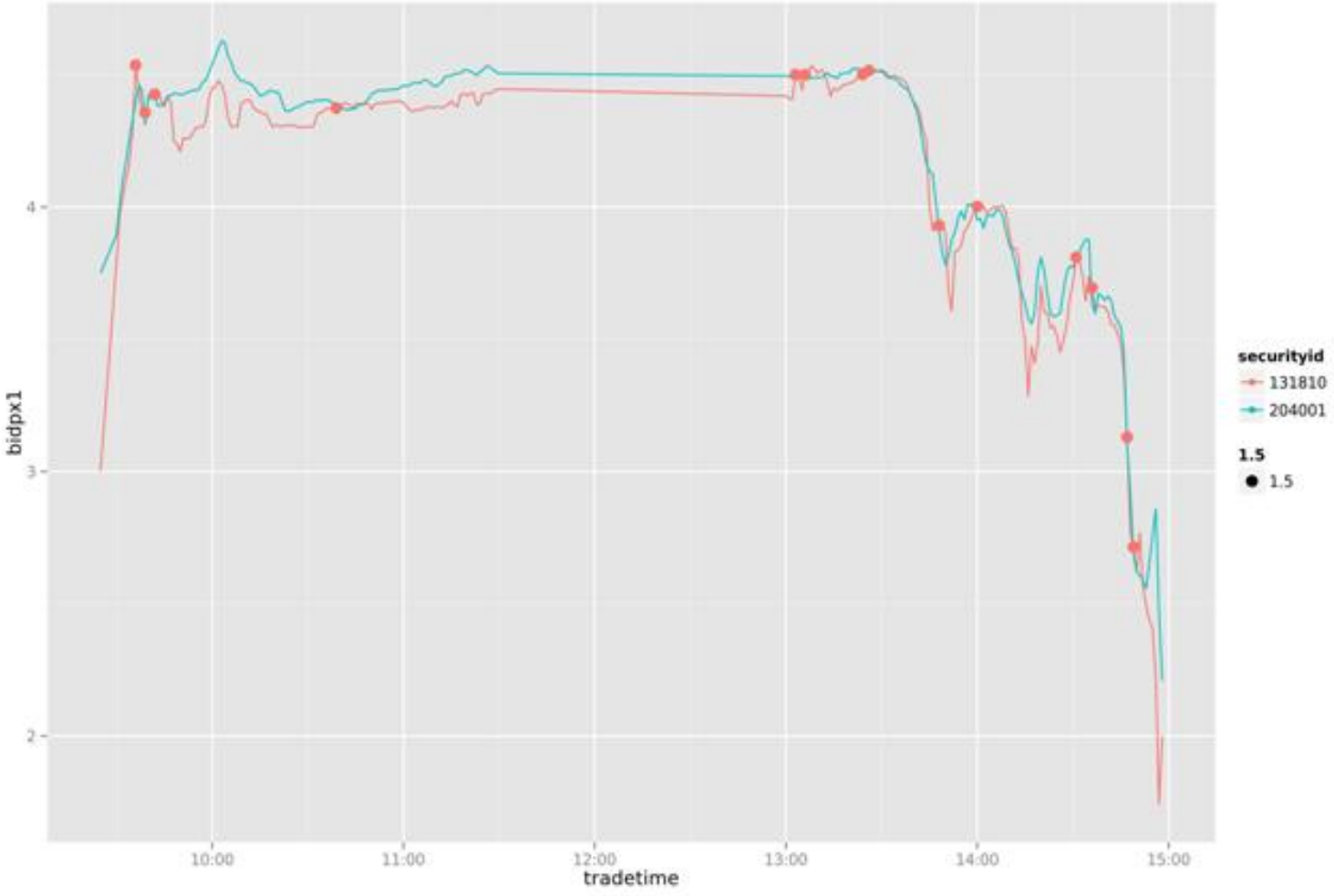
程序开发：策略可视化2013-07-22



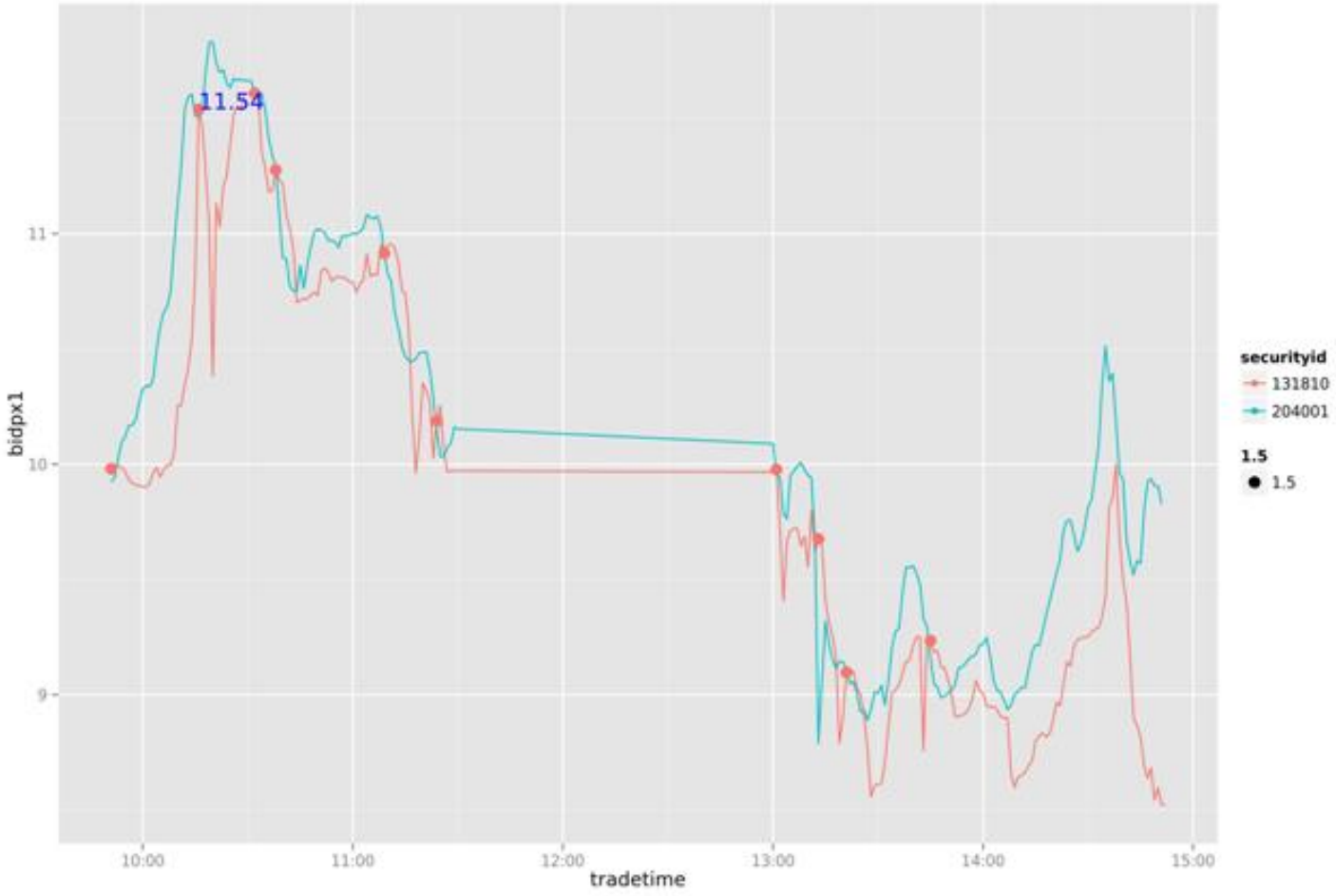
程序开发：策略可视化2013-07-23



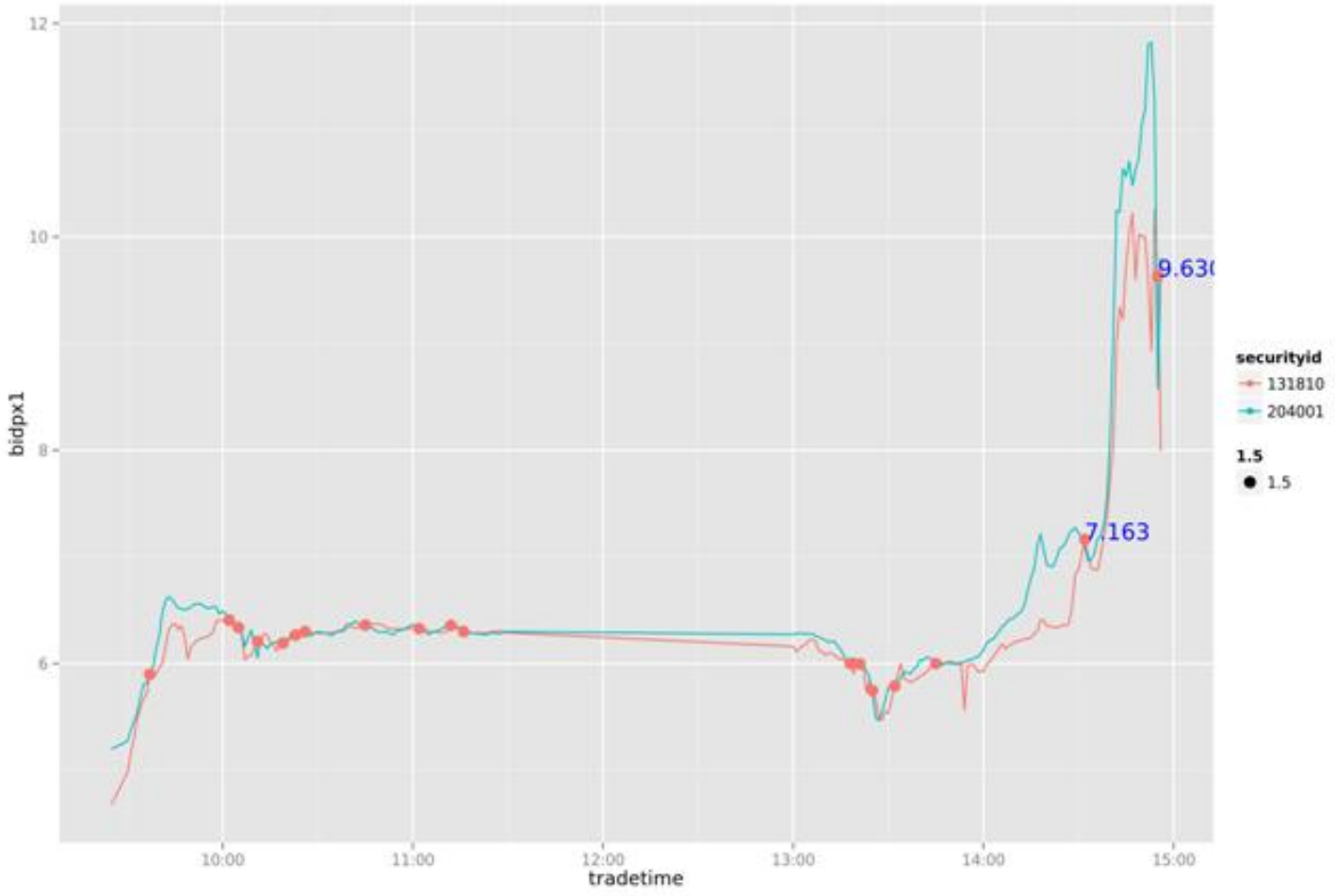
程序开发：策略可视化2013-07-24



程序开发：策略可视化2013-07-25



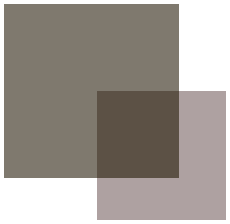
程序开发：策略可视化2013-07-26



联系作者

- 张丹， R语言深度用户
- Weibo: @Conan_Z
- Blog：**http://blog.fens.me**
- Email: bsspirit@gmail.com

- 《R的极客理想》系列图书，即将出版。



Thanks

FAQ时间