Table of Contents

安装

注意：我们目前的 RQAlpha 和 RQDatac 使用的都是 Python3.5 的版本。

MacOS

安装anaconda环境

如果您已经安装好了Python3+的anaconda环境，就可以跳过这个环节

Anaconda 是一个用于科学计算的 Python 发行版，支持 Linux, Mac, Windows, 包含了众多流行的科学计算、数据分析的 Python 包。

安装

anaconda环境，包含了整个Python的科学计算库，而rqalpha有很多模块是依赖于这些科学计算库的，因此下载anaconda会搭建起来一个强大的Python量化研发的基础环境：

1. 官方下载：https://www.continuum.io/downloads （我们不是特别建议，因为官方下载的速度较慢，但是如果您非常认可官方的可信度可以考虑...）
2. 清华的官方镜像下载：https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/ ，找最新的anaconda版本下载即可

更改Anconda下载源，提高下载速度

安装好anaconda的环境以后，就可以使用conda install 【包名】安装需要的Python库，非常方便。但是官方的服务器在国外，因此下载速度很慢，国内清华大学提供了Anaconda的仓库镜像，我们只需要配置Anaconda的配置文件，添加清华的镜像源，然后将其设置为第一搜索去到即可：

运行以下命令行：

```
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/
conda config --set show_channel_urls yes
```

然后运行 conda install numpy 试下吧！

CentOS

安装Anaconda

首先从清华的官方镜像下载anaconda Linux64的版本：

```
wget https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/Anaconda3-4.2.0-Linux-x86_64.sh
```

修改权限让这个脚本可运行：

```
chmod +x Anaconda3-4.2.0-Linux-x86_64.sh
```

接着运行这个安装Anaconda的脚本：

```
./Anaconda3-4.2.0-Linux-x86_64.sh
```

剩下就是一路Yes或者Enter好了...

```
Welcome to Anaconda3 4.2.0 (by Continuum Analytics, Inc.)


In order to continue the installation process, please review the license

agreement.

Please, press ENTER to continue
>>>
```

接着可以重新加载一下bash就可以看到新的conda命令行了：

```
source ~/.bashrc
```

然后尝试一下运行conda命令行看是否已经安装成功：

```
conda
```

设置matplotlib的backend（没有图形化界面的情况下）：

```
echo "backend: Agg" > ~/.config/matplotlib/matplotlibrc
```

安装中文字体

将 WenQuanYi Micro Hei.ttf 放到 /usr/share/fonts/chinese

```
mkdir /usr/share/fonts/chinese
cd /usr/share/fonts/chinese
wget https://static.ricequant.com/data/WenQuanYi%20Micro%20Hei.ttf
fc-cache -fv
fc-list
rm -rf ~/.cache/matplotlib
rm -rf ~/.fontconfig
```

Windows

安装anaconda环境

```
如果您已经安装好了Python3+的anaconda环境，就可以跳过这个环节
```

Anaconda 是一个用于科学计算的 Python 发行版，支持 Linux, Mac, Windows, 包含了众多流行的科学计算、数据分析的 Python 包。

安装

anaconda环境，包含了整个Python的科学计算库，而rqalpha有很多模块是依赖于这些科学计算库的，因此下载anaconda会搭建起来一个强大的Python量化研发的基础环境：

1. 官方下载：https://www.continuum.io/downloads （我们不是特别建议，因为官方下载的速度较慢，但是如果您非常认可官方的可信度可以考虑...）
2. 清华的官方镜像下载：https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/ ，找最新的anaconda版本下载即可

构建Conda虚拟环境

我们**强烈建议**您去创建并使用Python虚拟环境，因为这样才能让您的开发环境更加独立，不会因为安装不同的包而出现问题，造成运行失败等。目前流行的Python虚拟环境有两种：conda和pyenv, 由于大部分的量化开发都是基于anaconda的python技术栈，因此我们建议您使用conda作为默认的虚拟环境开发。

以下有几个常用的conda的命令行可以使用：

创建Conda虚拟环境

```
conda create --name env_name python=3.5
```

使用Conda虚拟环境

```
source activate env_name
```

退出Conda虚拟环境

```
source deactivate env_name
```

删除Conda虚拟环境

```
conda-env remove --name env_name
```

注意：Windows下无需加上source

安装机构版RQAlpha和RQDatac

以下的安装并不需要区分操作系统，pip install已经会自动区分安装对应的版本了。

注意！！以下所有的安装都应该使用Conda的虚拟环境！！

比如您的虚拟环境叫做:rqalpha-plus, 那么应该先运行 “source activate rqalpha-plus” 之后再做以下的安装，这样才可以保证您的所有包都成功安装在正确的虚拟环境之下！

Windows版本如果没有安装Visual C++ Compiler的话，

要先去http://www.lfd.uci.edu/~gohlke/pythonlibs/#bcolz下载相应版本的boclz wheel包

并且去http://www.lfd.uci.edu/~gohlke/pythonlibs/#line_profiler下载相应版本的line_profiler 包

我们强烈建议您先安装1和2的cython和bcolz依赖，因为比较容易失败，单独安装好然后继续安装ricequant的产品会比较好。

1. 安装cython, bcolz依赖cython：pip install -U pip cython
2. 安装bcolz, 有时候会卡一会儿，需要耐心等待...： pip install bcolz
3. 安装RQAlpha的机构版rqalpha-plus: (视乎您的网络状况可以选择不同的源改善加载速度)
    a. 豆瓣源：pip install -i http://pypi.douban.com/simple/ --trusted-host pypi.douban.com --extra-index-url https://rquser:Ricequant8@pypi2.ricequant.com/simple/ -U rqalpha-plus
    b. 阿里云源：pip install -i https://pypi.tuna.tsinghua.edu.cn/simple --extra-index-url https://rquser:Ricequant8@pypi2.ricequant.com/simple/ rqalpha-plus
    c. 清华源：pip install -i https://pypi.tuna.tsinghua.edu.cn/simple --extra-index-url https://rquser:Ricequant8@pypi2.ricequant.com/simple/ rqalpha-plus
4. 安装RQDatac：
    a. 豆瓣源：pip install -i http://pypi.douban.com/simple/ --trusted-host pypi.douban.com --extra-index-url https://rquser:Ricequant8@pypi2.ricequant.com/simple/ -U rqdatac
    b. 阿里云源：pip install -i https://pypi.tuna.tsinghua.edu.cn/simple --extra-index-url https://rquser:Ricequant8@pypi2.ricequant.com/simple/ rqdatac
    c. 清华源：pip install -i https://pypi.tuna.tsinghua.edu.cn/simple --extra-index-url https://rquser:Ricequant8@pypi2.ricequant.com/simple/ rqdatac

安装好了以后您可以尝试在命令行打入"rqalpha-plus"看下是否有如下的help出来，如果有的话就说明rqalpha-plus已经安装成功啦！

```
Usage: rqalpha-plus [OPTIONS] COMMAND [ARGS]...
Options:

  -v, --verbose

  --help          Show this message and exit.


Commands:

  examples        generate example strategies to target folder

  report          generate report from backtest output file
  run

  update_bundle   update data bundle, download if not found

  version         Output Version Info
```

也可以打rqalpha-plus run --help看到更多的run方面的帮助：

```
>> rqalpha-plus run --help

Usage: rqalpha-plus run [OPTIONS]

Options:
```

```
  -d, --data-bundle-path PATH
  -f, --strategy-file PATH
  -s, --start-date DATE
  -e, --end-date DATE

 ...

  --help                          Show this message and exit.
```

更新rqalpha-plus和rqdatac

我们会快速迭代和解决bug、加入新的功能，因此您如果收到我们的销售、产品经理的通知的时候，可以使用上面一样的安装的命令行来更新:

1. 更新rqalpha-plus(多源，视乎您的不同的下载速度):
    a. 豆瓣源：pip install -i http://pypi.douban.com/simple/ --trusted-host pypi.douban.com --extra-index-url https://rquser:Ricequant8@pypi2.ricequant.com/simple/ -U rqalpha-plus
    b. 阿里云源：pip install -i https://pypi.tuna.tsinghua.edu.cn/simple --extra-index-url https://rquser:Ricequant8@pypi2.ricequant.com/simple/ rqalpha-plus
    c. 清华源：pip install -i https://pypi.tuna.tsinghua.edu.cn/simple --extra-index-url https://rquser:Ricequant8@pypi2.ricequant.com/simple/ rqalpha-plus
2. 更新rqdatac(多源，视乎您的不同的下载速度):
    a. 豆瓣源：pip install -i http://pypi.douban.com/simple/ --trusted-host pypi.douban.com --extra-index-url https://rquser:Ricequant8@pypi2.ricequant.com/simple/ -U rqdatac
    b. 阿里云源：pip install -i https://pypi.tuna.tsinghua.edu.cn/simple --extra-index-url https://rquser:Ricequant8@pypi2.ricequant.com/simple/ rqdatac
    c. 清华源：pip install -i https://pypi.tuna.tsinghua.edu.cn/simple --extra-index-url https://rquser:Ricequant8@pypi2.ricequant.com/simple/ rqdatac

更新数据bundle

注意！！以下所有的安装都应该使用Conda的虚拟环境！！

比如您的虚拟环境叫做: rqalpha-plus, 那么应该先运行 “source activate **rqalpha-plus**” 之后再做以下的安装，这样才可以保证您的所有包都成功安装在正确的虚拟环境之下!

Windows 系统因为缺少 rsync ，我们需要手动安装。我们可以从 https://www.itefix.net/content/cwrsync-free-edition 下载 rsync ，然后解压缩到一个目录，例如 C:\cwrsync 。然后我们需要从计算机高级设置中，把 C:\cwrsync\bin 加到 PATH 中。这样就完成了 rsync 安装。我们在 cmd 中运行 rsync ，如果能够输出 rsync 的信息，那么就安装成功了。

rqalpha-plus需要有本地的我们米筐的历史数据bundle才可以运行回测，但是由于包含了分钟级别的10年+所有品种的股票和期货历史数据，所以数据量比较大，需要**花费数小时下载**（视乎您的网络情况）。bundle 默认目录在 ~/.rqalpha

下载rqalpha-plus需要的数据bundle的命令行:

```
rqalpha-plus update_bundle
```

密码为 di18SDkqi18sisfASDiqj

如果是 Windows 用户，情况稍微有些特殊，需要手动指定 bundle 的相对路径，例如我们把 bundle 放在 D:\ 根目录，或者放在用户 home 目录，我们需要执行下面的命令

```
c:\ cd d:
d:\ rqalpha-plus update_bundle -d .rqalpha-plus
```

然后，进行回测的时候，在同样的目录，执行

```
d:\ rqalpha-plus run -d .rqalpha-plus -f ......
```

如何自动输入密码更新数据Bundle?

sshpass -p di18SDkqi18sisfASDiqj rqalpha update_bundle

注意！！

Windows下不支持sshpass，rqalpha-plus update_bundle需要加上-d指定相应的相对路径，不支持绝对路径。

rqalpha-plus中的rqdatac数据对接配置

由于RQAlpha-Plus里面只有日和分钟的交易所市场数据，因此需要额外指向我们的rqdata服务来获得更多的数据来源（指数构成、板块分类、财务数据等等），运行的时候需要加入额外的参数:

1. rqdatad-addr: rqdata服务的服务器地址
2. rqdatad-port：rqdata服务的端口
3. rqdatad-username：rqdata服务的用户名（找米筐科技的销售获得）
4. rqdatad-password：rqdata服务的密码（找米筐科技的销售获得）

一个范例如下:

```
rqalpha-plus run -f examples/buy_and_hold.py --kind stock --log-level verbose --stock-starting-cash 10000 -s 2015-01-01 -e 2015-04-01 --frequency 1d --progress --rqdatad-addr rqdatad.ricequant.com --rqdatad-port 16003 -o /tmp/a.pkl --rqdatad-username rqdata_username --rqdatad-password rqdata_password --plot --report ./
```

rqdatac支持的数据种类

您可以在https://www.ricequant.com/doc/rqdata-institutional找到rqdatac最新的数据字典，我们不断在丰富我们支持的数据品种，希望能给您的策略带来更多的可能!

基本使用

生成策略范例

运行以下的命令行可以生成策略范例代码，便于一开始的测试和上手:

```
rqalpha-plus examples
```

上面的命令行会在当前文件夹生成一个examples的范例文件夹，里面会有不少的一些经典测试范例。

运行第一个范例策略

接着我们可以尝试运行一个最简单的买入并持有的策略吧!

```
rqalpha-plus run -f examples/buy_and_hold.py --kind stock --log-level verbose --stock-starting-cash 10000 -s 2015-01-01 -e 2015-04-01 --frequency 1d --progress --rqdatad-addr rqdatad.ricequant.com --rqdatad-port 16003 -o buy_and_hold.pkl --rqdatad-username rqdata_username --rqdatad-password rqdata_password --benchmark 000300.XSHG --plot --report ./
```

上面有几个常见的运行参数都可以通过rqalpha-plus run --help来了解到他们的用途。

注意！！

Windows下的路径分隔符为\，而非/

编写策略

您可以随便创建一个python文件，只需要里面必须包含我们的几个要求的回调函数:

1. def init(context)
2. def before_trading(context, bar_dict):
3. def handle_bar(context, bar_dict):

最简单的方法还是您可以copy paste一份范例策略，然后基于上面修改编辑您的策略代码。

保留和分析回测结果

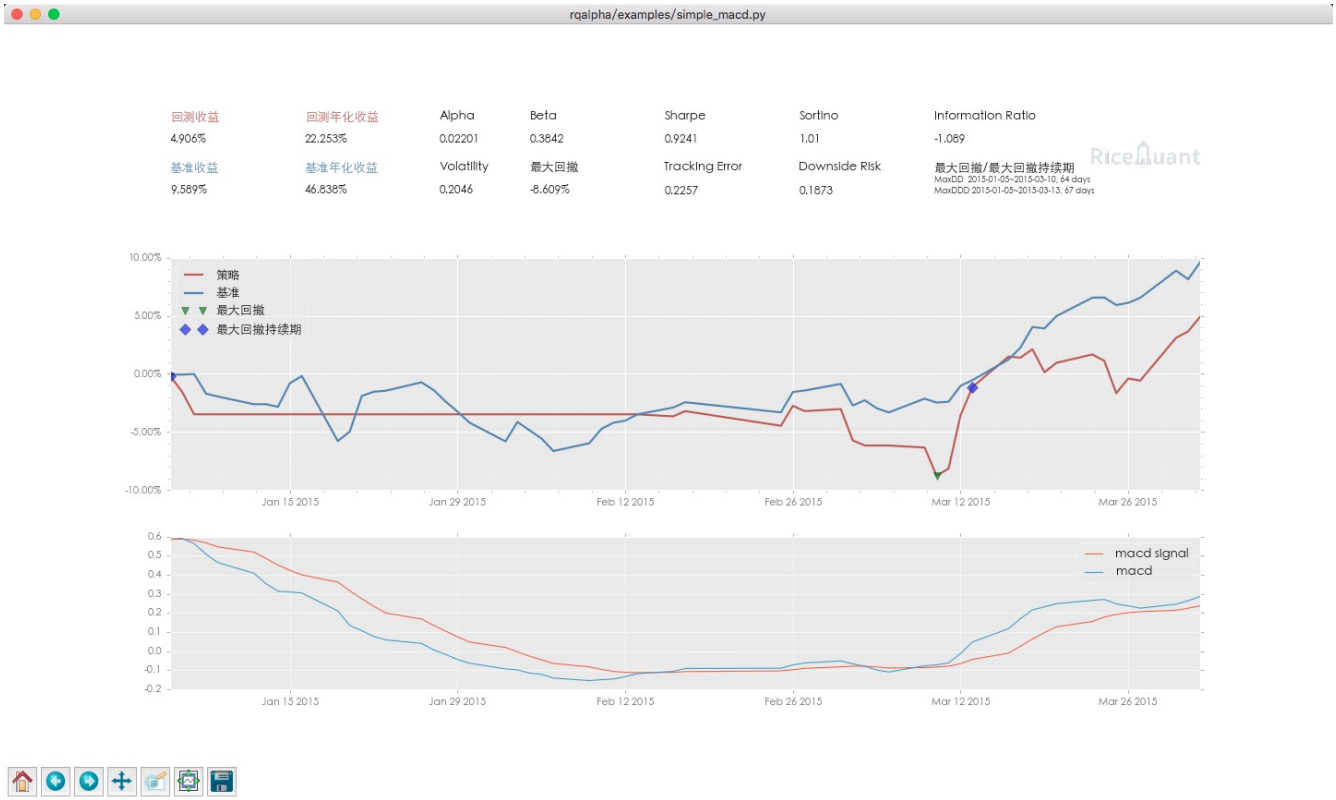
您可以通过几种方式来分析回测结果:

1. 收益曲线图报告
    a. 回测结束弹出收益曲线报告
    b. 保留曲线报告
    c. 从pickle文件生成收益曲线图
2. 详实的excel和csv策略报告
    a. 回测结束自动保存csv报告文件
    b. 从pickle文件生成csv报告文件
3. 结果pickle文件

收益曲线图报告

回测结束弹出收益曲线报告

通过–plot来展示回测的图形化结果，这样可以一目了然您的回测表现以及各项收益和风险：



保留收益曲线图报告

如果您在服务器上运行rqalpha-plus得到的结果报告不方便打开查看的话，也可以保存下来，在rqalpha-plus run后面加入--plot-save的命令行，比如下面的范例，是指定当前的文件夹保存，由于没有指定文件名，因此保存的图片文件名就是策略名.png:

```
rqalpha-plus run -f examples/buy_and_hold.py --kind stock --log-level verbose --stock-starting-cash 10000 -s 2015-01-01 -e 2015-04-01 --frequency 1d --progress --rqdatad-addr rqdatad.ricequant.com --rqdatad-port 16003 -o buy_and_hold.pkl --rqdatad-username rqdata_username --rqdatad-password rqdata_password  --plot-save .
```

运行完以后会发现有一个叫做"buy_and_hold.csv"的图形化报告保存在当前文件夹

您也可以指定保存的图形化报告的文件名：

```
rqalpha-plus run -f examples/buy_and_hold.py --kind stock --log-level verbose --stock-starting-cash 10000 -s 2015-01-01 -e 2015-04-01 --frequency 1d --progress --rqdatad-addr rqdatad.ricequant.com --rqdatad-port 16003 -o buy_and_hold.pkl --rqdatad-username rqdata_username --rqdatad-password rqdata_password  --plot-save ./another_buy_hold.png
```

接着您会发现有一个叫做“another_buy_hold.png”的图形化报告保存在当前文件夹了。

从pickle文件生成收益曲线图报告

如果您只使用了-o 保存了pickle文件，依然可以从pickle文件生成收益曲线图报告的:

```
rqalpha-plus report [pickle文件] [生成收益曲线图报告的文件夹路径]
```

详细的csv策略结果报告

回测结束自动保存csv报告文件

如果想回测结束的时候就自动生成详细的excel和csv策略报告，可以在运行命令后面加入"--report [文件夹]"，比如："--report ./"就是指定生成报告在当前的文件夹下，文件夹的名字默认就是策略的名字，注意结果可能会被覆盖，最好自己创建不同的文件夹来保存不同运行次数的回测结果报告，生成的报告包括以下几种：

文件名	用途
report.xlsx	所有以下文件的汇总excel表
summary.csv	运行参数和回测结果总结
total_portfolios.csv	股票、期货等投资组合的汇总

stock_portfolios.csv	股票投资子组合的汇总
future_portfolios.csv	期货投资子组合的汇总
stock_positions.csv	股票持仓的汇总
future_positions.csv	期货持仓的汇总
trades.csv	所有交易记录的汇总

详细的每个报告里面的字段可以在附录里的策略结果报告数据字典里找到

从pickle文件生成csv报告文件

如果您只是喜欢保存结果到pickle文件，您也可以从pickle文件结果生成详尽的可阅读的excel、csv格式的报告文件，使用“rqalpha-plus report”命令行：

```
rqalpha-plus report

Usage: rqalpha-plus report [OPTIONS] RESULT_PICKLE_FILE_PATH

                           TARGET_REPORT_CSV_FILE
```

比如将buy_and_hold.pkl转换成一个文件夹“buy_and_hold”的excel和csv结果报告，保存在当前文件夹目录下：

```
rqalpha-plus report buy_and_hold.pkl ./
```

RQAlpha-Plus的各项参数

运行策略的各项参数

使用rqalpha-plus run即可开始运行回测，运行 “rqalpha-plus run --help” 即可看到所有运行策略的各项参数解释：

参数名缩写	参数名全称	说明	
-d	--data-bundle-path PATH	指定rqalpha-plus使用的历史数据bundle的位置	
-f	--strategy-file PATH	指定rqalpha-plus运行的策略文件的位置	
-s	--start-date DATE	回测的开始日期，格式："YYYY-mm-dd"	
-e	--end-date DATE	回测的结束日期，格式：“"YYYY-mm-dd"”	
-r	--rid TEXT	制定一个运行回测的run id, 基本不会用到	
-sc	--stock-starting-cash FLOAT	股票账户初始资金	
-fc	--future-starting-cash FLOAT	期货账户初始资金	
-bm	--benchmark TEXT	设定对标的基准合约	
-sp	--slippage FLOAT	滑点水平	
-cm	--commission-multiplier FLOAT	佣金倍率	
-mm	--margin-multiplier FLOAT	保证金倍率	
-k	--kind TEXT stock/future	策略的类型 - • stock: 纯股票策略 • future: 纯期货策略 • stock_future: 股票期货混合策略	

-fq	--frequency [1d|1m] 1d/1m	使用日/分钟回测， • 日回测: --frequency 1d • 分钟回测: --frequency 1m	
-me	--match-engine [current_bar|next_bar]	• 使用当前的bar做订单成交撮合：--match-engine current_bar • 使用下一分钟的bar做订单成交撮合：--macth-engine next_bar 默认的是current_bar	
-rt	--run-type [b|p]	• b: 回测 • p: 实盘模拟交易	
	--risk-grid / --no-risk-grid	是否计算risk grid	
-l	--log-level [verbose|debug|info|error]	日志的级别： • verbose - 最详细的日志级别 • debug - 较为详细的debug日志级别 • info - • error	
-p	--plot / --no-plot plot result	是否画报告图	
	--plot-save PATH save plot result to file	保存报告图到制定的文件夹/文件名	
-o	--output-file PATH	制定报告pickle文件的存储位置	
	--fast-match	是否卖出以后立刻成交，资金到账，而不是跟随真实情况卖出以后等待一段时间成交了才会资金到账	
	--progress / --no-progress show progress bar	是否打印运行回测的进度	
	--report	生成excel、csv等结果报告文件	
-rdu	--rqdatad-username TEXT	指定RQData数据源的用户名	
-rdpw	--rqdatad-password TEXT	指定RQData数据源的密码	
-rda	--rqdatad-addr TEXT	指定RQData数据源的地址	
-rdpt	--rqdatad-port TEXT	指定RQData数据源的端口	
-h	--help	打印该帮助文档	

更多的策略范例

您可以访问 www.ricequant.com 寻找更多的策略范例和思路，以及教学：https://www.ricequant.com/courses ，我们支持的数据字典可以在 https://www.ricequant.com/doc/rqdata-institutional 找到。

您运行过"rqalpha-plus examples"之后，也可以在本地的examples文件夹下找到不少的策略范例，从策略范例熟悉如何编写策略可能是最快的方法之一。

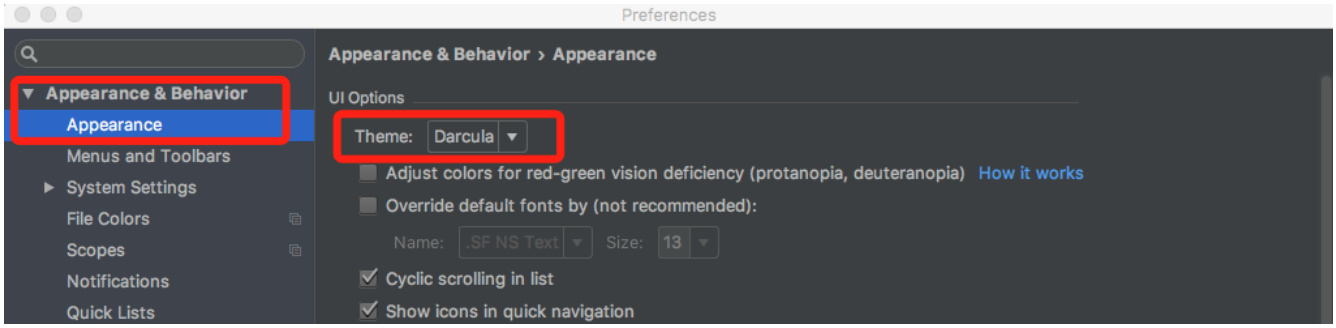
IDE下写策略和DEBUG

下载并且搭建PyCharm环境

有不少的Python IDE（集成开发环境可以选取），不过我们非常建议您使用PyCharm，一方面PyCharm的功能极为强大、易用，另外一方面米筐团队也比较擅长使用PyCharm，能更好的提供支持服务。

搜索一下就可以找到PyCharm的下载，community版本就可以了，如果有需要也可以购买他们的专业license：https://www.jetbrains.com/pycharm/download/

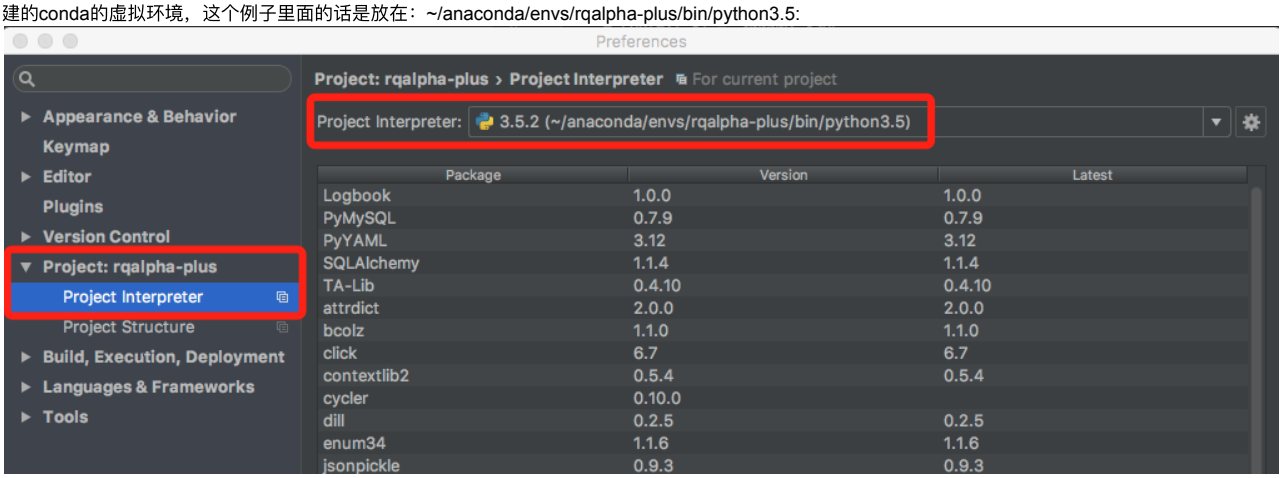
安装好以后我们首先非常建议您更改主题为【Darcula】,黑黑的背景写策略代码更爽：



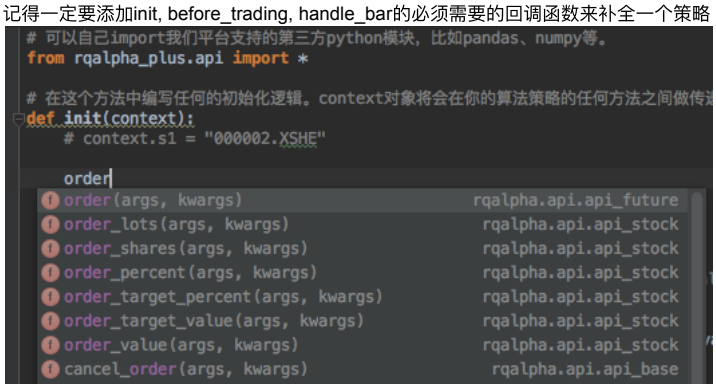
在Py Charm IDE中搭建开发环境

以下都假设我们使用的conda虚拟环境叫做【rqalpha-plus】，并且您已经使用conda已经建立好了对应的【rqalpha-plus】的虚拟环境!

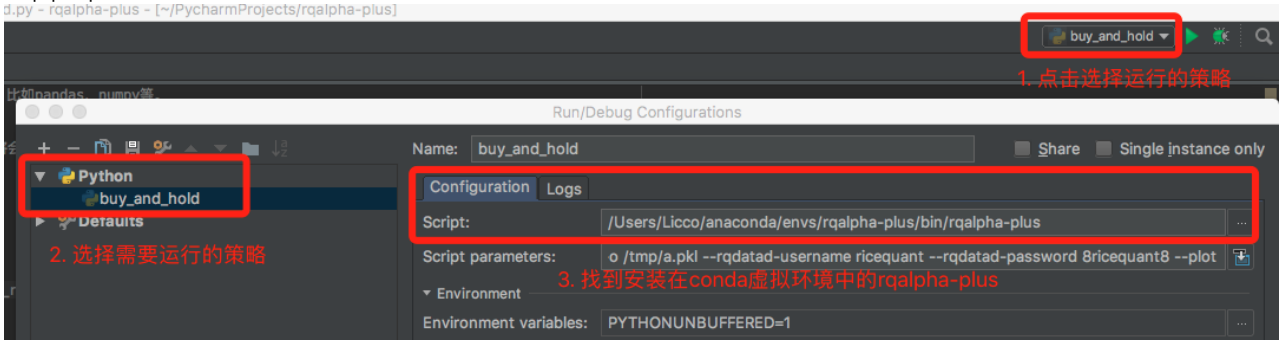
1. 新建一个项目: File→ New Project, 比如项目叫做rqalpha-plus
2. 配置对应的conda虚拟环境的Python解释器给他：PyCharm Community Edition → Preferences → Project: rqalpha-plus → Project Interpreter, 然后选择您创建的conda的虚拟环境，这个例子里面的话是放在：~/anaconda/envs/rqalpha-plus/bin/python3.5:



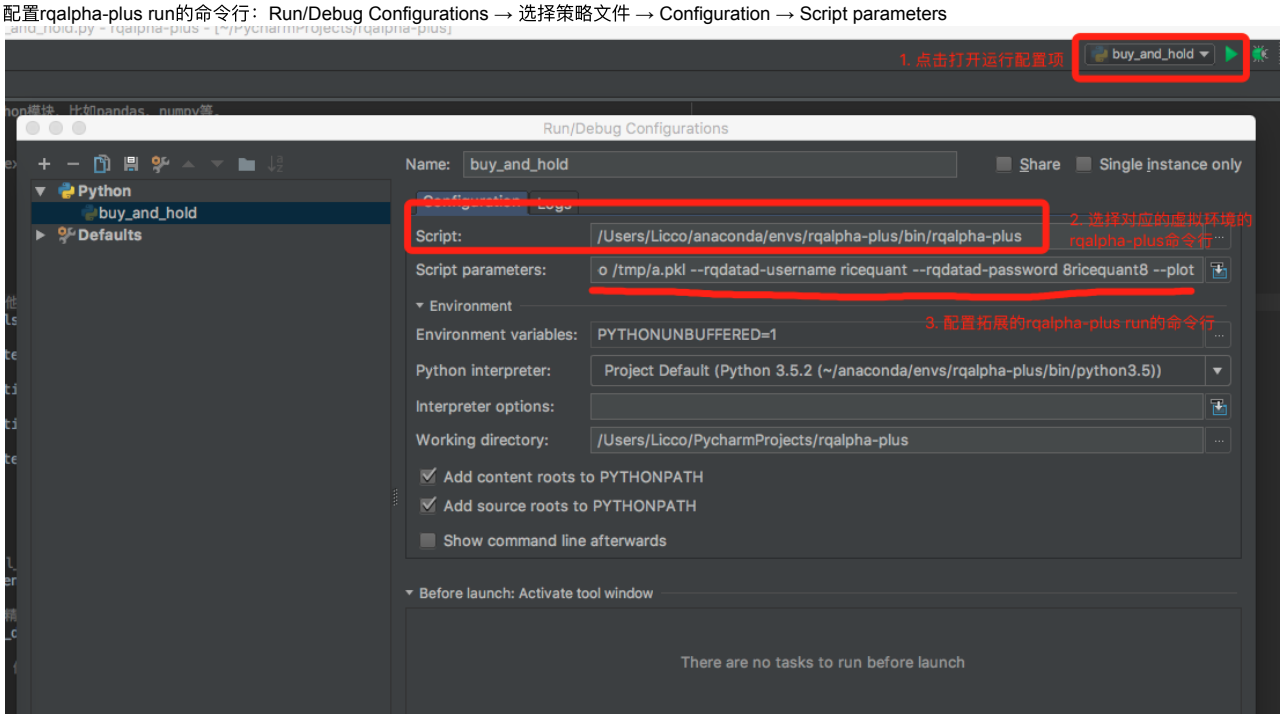
3. 项目创建好以后，新建立一个简单的rqalpha策略吧，比如叫做buy_and_hold.py: 右键点击rqalpha-plus项目→ New→ File→ buy_and_hold.py:
4. 在buy_and_hold.py策略里面import所有的rqalpha-plus支持的API: "from rqalpha_plus.api import *"，那么可以享受到代码的自动补全了:
5. 记得一定要添加init, before_trading, handle_bar的必须需要的回调函数来补全一个策略



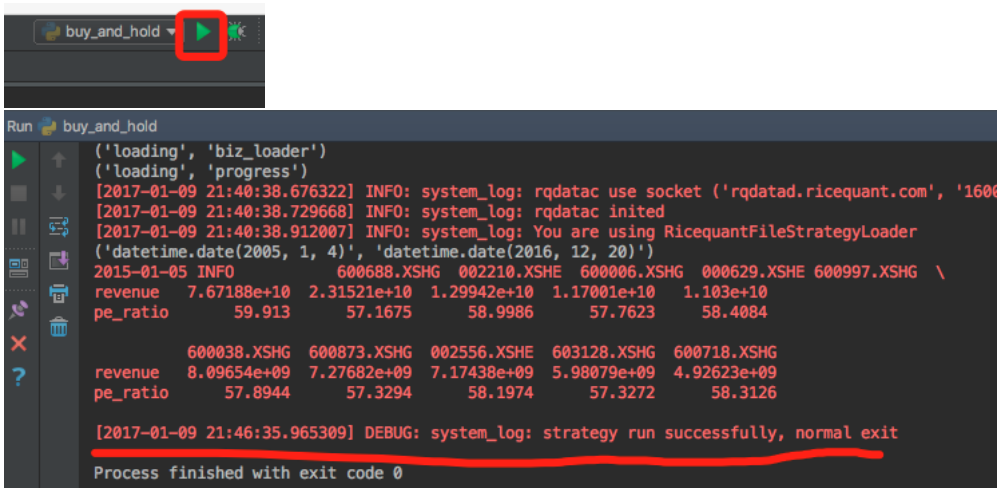
6. 配置运行的命令行rqalpha-plus的在conda环境中的位置：Run/Debug Configurations → 选择策略文件 → Configuration → Script → 找到对应的conda环境的rqalpha-plus



7. 配置rqalpha-plus run的命令行：Run/Debug Configurations → 选择策略文件 → Configuration → Script parameters



8. 点击运行！可以看到运行结果：



Q&A

Q: 为什么我的策略无法用–plot画matplotlib的图？（Mac操作系统）

A: 这是因为matplotlib的后端渲染问题，matplotlib的后端渲染引擎有Qt4Agg和GTKAgg，但是没有默认的。我们可以这样在MacOS中解决问题：

1. 我们假设您通过pip安装matplotlib，有一个文件夹叫做 ~/.matplotlib
2. 在上面的文件夹创建一个文件：~/.matplotlib/matplotlibrc，然后在该文件中加入这样的一行：backend: TkAgg，然后保存
3. 重新加入 --plot 运行您的策略，就可以看到matplotlib的画图出来了！

Q: 为什么我无法运行一些使用到talib的策略，比如MACD,etc...

A: 单独安装TA-Lib就可以了:

```
pip install TA-Lib
```

Q: 如何选取不同的基准（benchmark）？

A: rqalpha-plus run的时候加多一个参数： --benchmark order_book_id , 比如如果想使用csi300作为基准： --benchmark 000300.XSHG , 一个完整的运行范例命令行是:

```
rqalpha-plus run -f examples/buy_and_hold.py --kind stock --log-level verbose --stock-starting-cash 10000 -s 2015-01-01 -e 2015-04-01 --frequency 1d --progress --rqdatad-addr rqdatad.ricequant.com --
```

```
rqdatad-port 16003 -o buy_and_hold.pkl --rqdatad-username rqdata_username --rqdatad-password rqdata_password --
benchmark 000300.XSHG --plot
```

Q: 我们支持什么样的策略?

A: 由于我们支持股票和期货的数据和策略引擎，并且支持混合品种的策略的实现。因此从简单的股票买入并持有，定期调仓，统计套利到期货的CTA策略和股票、期货混搭的Alpha对冲策略都可以支持。

Q: 怎么进行参数调优?

A: 对于当前版本的RQAlpha，我们提供了 --extra-vars 传入参数功能，通过覆盖 context 中变量的方式进行参数调优。我们提供的样例策略 golden_cross.py 中 context.SHORTPERIOD 和 context.LONGPERIOD 为用来计算两条均线的变量，下面的示例流程就是针对这两个变量通过简单嵌套循环方式进行参数调优，最后分别以夏普率以及策略年化收益率对策略表现进行分析：

1. 运行 mkdir results ，在当前目录下创建 results 文件夹，用于存放每次回测结果的 pickle 文件
2. 嵌套循环进行参数调优。创建一个 run_optimize.py ，填入下面的代码，通过命令行运行 python run_optimize.py

```
import os
import json
import base64
import concurrent.futures
import multiprocessing


tasks = []
for short_period in range(3, 10, 2):
    for long_period in range(30, 90, 5):
        extra_vars = {
            "SHORTPERIOD": short_period,
            "LONGPERIOD": long_period,
        }
        vars_params = base64.b64encode(json.dumps(extra_vars).encode("utf-8")).decode("utf-8")

        cmd = ("rqalpha-plus run -fq 1d -f rqalpha/examples/golden_cross.py --start-date 2015-01-01 --end-date
2016-01-01 "
               "-o results/out-{short_period}-{long_period}.pkl -sc 100000 --progress -bm 000001.XSHE --extra-
vars {params} ").format(
                   short_period=short_period,
                   long_period=long_period,
                   params=vars_params)

        tasks.append(cmd)


def run_bt(cmd):
    print(cmd)
    os.system(cmd)


with concurrent.futures.ProcessPoolExecutor(max_workers=multiprocessing.cpu_count()) as executor:
    for task in tasks:
        executor.submit(run_bt, task)
```

3. 分析批量回测结果。我们创建一个 analyser.py ，填入下面代码，通过命令行运行 python analyser.py

```
import glob
import pandas as pd


results = []

for name in glob.glob("results/*.pkl"):
    result_dict = pd.read_pickle(name)
    summary = result_dict["summary"]
    results.append({
        "name": name,
        "annualized_returns": summary["annualized_returns"],
        "sharpe": summary["sharpe"],
        "max_drawdown": summary["max_drawdown"],
    })

results_df = pd.DataFrame(results)
```

```
print("-" * 50)
print("Sort by sharpe")
print(results_df.sort_values("sharpe", ascending=False)[:10])

print("-" * 50)
print("Sort by annualized_returns")
print(results_df.sort_values("annualized_returns", ascending=False)[:10])
```

4. 最后我们就得到，按照「Sharpe」排序的参数组合，以及按照「年化收益 」排序的参数组合。

```
% python analyser.py
--------------------------------------------------
Sort by sharpe
    annualized_returns  max_drawdown                 name  sharpe
3                0.350         0.119  results/out-3-45.pkl   1.231
4                0.347         0.119  results/out-3-50.pkl   1.218
15               0.313         0.119  results/out-5-45.pkl   1.095
16               0.290         0.125  results/out-5-50.pkl   1.009
28               0.263         0.144  results/out-7-50.pkl   0.895
9                0.261         0.182  results/out-3-75.pkl   0.885
5                0.246         0.131  results/out-3-55.pkl   0.848
41               0.239         0.152  results/out-9-55.pkl   0.793
27               0.212         0.163  results/out-7-45.pkl   0.718
14               0.206         0.166  results/out-5-40.pkl   0.704
--------------------------------------------------
Sort by annualized_returns
    annualized_returns  max_drawdown                 name  sharpe
3                0.350         0.119  results/out-3-45.pkl   1.231
4                0.347         0.119  results/out-3-50.pkl   1.218
15               0.313         0.119  results/out-5-45.pkl   1.095
16               0.290         0.125  results/out-5-50.pkl   1.009
28               0.263         0.144  results/out-7-50.pkl   0.895
9                0.261         0.182  results/out-3-75.pkl   0.885
5                0.246         0.131  results/out-3-55.pkl   0.848
41               0.239         0.152  results/out-9-55.pkl   0.793
39               0.216         0.175  results/out-9-45.pkl   0.699
27               0.212         0.163  results/out-7-45.pkl   0.718
```

附录

策略结果报告数据字典

report.xlsx就是所有的csv报告文件的汇总excel表格，下面会分别介绍不同的报告文件的字段

summary.csv - 运行参数和回测结果总结

属性	类型	注释	备注
alpha	*float*	策略最终的alpha	
annualized_returns	*float*	策略的年化收益率	
benchmark	*str*	基准合约	经常会选取指数，比如csi300: 000300.XSHG
benchmark_annualized_returns	*float*	基准合约的年化收益率	
benchmark_total_returns	*float*	基准合约的总收益	
beta	*float*	策略最终的beta	
cash	*float*	策略当前剩余现金	
commission_multiplier	*float*	佣金倍率	详细 见：https://www.ricequant.com/api/python/chn#backtests-transaction_cost
downside_risk	*float*	策略收益下行波动率(年化)。和普通收益波动率相比，下行标准差区分了好的和坏的波动。	具体的公式解释 见：https://www.ricequant.com/api/python/chn#risk-metrics
end_date	*date*	回测结束日期	
frequency	*str*	日/分钟回测	
frozen_cash	*float*	冻结资金	Optional
future_starting_cash	*float*	期货账户初始资金	
information_ratio	*float*	信息比率。衡量超额风险带来的超额收益。	具体的公式解释 见：https://www.ricequant.com/api/python/chn#risk-metrics

margin_multiplier	*float*	保证金倍率	
market_value	*float*	投资组合当前的市场价值，为子组合市场价值的加总	
matching_type	*MATCHING_TYPE*	撮合方式，MATCHING_TYPE.NEXT_BAR_OPEN代表以下一bar开盘价撮合，MATCHING_TYPE.CURRENT_BAR_CLOSE代表以当前bar收盘价撮合	
max_drawdown	*float*	最大回撤。描述策略可能出现的最糟糕的情况。和Sortino Ratio一样适合对资产下跌敏感的投资者。	具体的公式解释见：https://www.ricequant.com/api/python/chn#risk-metrics
pnl	*float*	当前投资组合的累计盈亏	
portfolio_value	*float*	总权益，为子组合总权益加总	
run_id	*long*	策略运行的ID	
run_type	*str*	BACKTEST(回测)	
sharpe	*float*	夏普比率。表示每承受一单位总风险，会产生多少的超额回报酬。	具体的公式解释见：https://www.ricequant.com/api/python/chn#risk-metrics
slippage	*float*	滑点水平	
sortino	*float*	索提诺比率。表示每承担一单位的下行风险，将会获得多少超额回报(*excess return*). 具体计算方法为(策略年化收益率－回测起始交易日的无风险利率）/ 策略收益下行波动率 。	具体的公式解释见：https://www.ricequant.com/api/python/chn#risk-metrics
start_date	date	策略投资组合的回测/实时模拟交易的开始日期	
starting_cash	*float*	回测或实盘交易给算法策略设置的初始资金	
stock_starting_cash	*float*	股票账户初始资金	
strategy_file	*str*	运行的策略文件	
strategy_name	*str*	运行的策略名称	
strategy_type	*str*	stock: 纯股票策略 future: 纯期货策略 stock_future: 股票期货混合策略	
total_returns	*float*	投资组合至今的累积收益率。计算方法是现在的投资组合价值/投资组合的初始资金	
tracking_error	*float*	跟踪误差波动率(年化)。数值越大表示投资组合和参考标准距离越大。	具体的公式解释见：https://www.ricequant.com/api/python/chn#risk-metrics
transaction_cost	*float*	总费用	
volatility	*float*	策略收益波动率（年化）。用来测量资产的风险性，波动越大代表投资组合风险越高。	具体的公式解释见：https://www.ricequant.com/api/python/chn#risk-metrics

total_portfolios.csv - 股票、期货等投资组合的汇总

属性	类型	注释	备注
date	*date*	该Portfolio记录的日期	
cash	*float*	可用资金，为子组合可用资金的加总	
total_returns	*float*	投资组合至今的累积收益率。计算方法是现在的投资组合价值/投资组合的初始资金	
daily_returns	*float*	投资组合每日收益率	
daily_pnl	*float*	当日盈亏，子组合当日盈亏的加总	
market_value	*float*	投资组合当前的市场价值，为子组合市场价值的加总	
portfolio_value	*float*	总权益，为子组合总权益加总	
pnl	*float*	当前投资组合的累计盈亏	

属性	类型	注释	备注
annualized_returns	*float*	投资组合的年化收益率	
frozen_cash	*float*	冻结资金	
transaction_cost	*float*	总费用	

stock_portfolios.csv - 股票投资子组合的汇总

属性	类型	注释	备注
date	*date*	该Portfolio记录的日期	
cash	*float*	可用资金	
total_returns	*float*	投资组合至今的累积收益率。计算方法是现在的投资组合价值/投资组合的初始资金	
daily_returns	*float*	当前最新一天的每日收益	
daily_pnl	*float*	当日盈亏，当日投资组合总权益-昨日投资组合总权益	
market_value	*float*	投资组合当前所有证券仓位的市值的加总	
portfolio_value	*float*	总权益，包含市场价值和剩余现金	
pnl	*float*	当前投资组合的累计盈亏	
annualized_returns	*float*	投资组合的年化收益率	
frozen_cash	*float*	冻结资金	Optional
dividend_receivable	*float*	投资组合在分红现金收到账面之前的应收分红部分。具体细节在分红部分	
transaction_cost	*float*	总费用	

future_portfolios.csv - 期货投资子组合的汇总

属性	类型	注释	备注
date	*date*	该Portfolio记录的日期	
cash	*float*	可用资金	
frozen_cash	*float*	冻结资金	Optional
total_returns	*float*	投资组合至今的累积收益率，当前总权益/初始资金	
daily_returns	*float*	当日收益率 = 当日收益 / 昨日总权益	
market_value	*float*	投资组合当前所有期货仓位的名义市值的加总	
pnl	*float*	累计盈亏，当前投资组合总权益-初始资金	
daily_pnl	*float*	当日盈亏，当日浮动盈亏 + 当日平仓盈亏 - 当日费用	
daily_holding_pnl	*float*	当日浮动盈亏	
daily_realized_pnl	*float*	当日平仓盈亏	
portfolio_value	*float*	总权益，昨日总权益+当日盈亏	
transaction_cost	*float*	总费用	
annualized_returns	*float*	投资组合的年化收益率	
margin	*float*	已占用保证金	
buy_margin	*float*	多头保证金	
sell_margin	*float*	空头保证金	

stock_positions.csv - 股票持仓的汇总

属性	类型	注释	备注
date	*date*	该仓位数据的日期	
quantity	*int*	当前持仓股数	
pnl	*float*	持仓累计盈亏	

属性	类型	注释	备注
date	*date*	该仓位数据的日期	
bought_quantity	*int*	该证券的总买入股数，例如：如果你的投资组合并没有任何平安银行的成交，那么平安银行这个股票的仓位就是0	
sold_quantity	*int*	该证券的总卖出股数，例如：如果你的投资组合曾经买入过平安银行股票200股并且卖出过100股，那么这个属性会返回100	
bought_value	*float*	该证券的总买入的价值，等于每一个该证券的 买入成交价 * 买入股数 总和	
sold_value	*float*	该证券的总卖出价值，等于每一个该证券的 卖出成交价 * 卖出股数 总和	
total_orders	*int*	该仓位的总订单的次数	待定，有可能将会去掉支持
total_trades	*int*	该仓位的总成交的次数	待定，有可能将会去掉支持
sellable	*int*	该仓位可卖出股数。T＋1的市场中sellable = 所有持仓-今日买入的仓位	
market_value	*float*	获得该持仓的实时市场价值	
value_percent	*float*	获得该持仓的实时市场价值在总投资组合价值中所占比例，取值范围[0, 1]	
transaction_cost	*float*	仓位交易费用	
symbol	str	合约名称	
avg_price	float	平均建仓价格	

future_positions.csv - 期货持仓的汇总

属性	类型	注释
date	*date*	该仓位数据的日期
order_book_id	*str*	合约代码
pnl	*float*	累计盈亏
daily_pnl	*float*	当日盈亏，当日浮动盈亏+当日平仓盈亏
daily_holding_pnl	*float*	当日持仓盈亏
daily_realized_pnl	*float*	当日平仓盈亏
transaction_cost	*float*	仓位交易费用
margin	*float*	仓位总保证金
market_value	*float*	当前仓位的名义价值。如果当前净持仓为空方向持仓，则名义价值为负
buy_daily_pnl	*float*	多头仓位当日盈亏
buy_pnl	*float*	多头仓位累计盈亏
buy_transaction_cost	*float*	多头费用
closable_buy_quantity	*float*	可平多头持仓
buy_margin	*float*	多头持仓占用保证金
buy_today_quantity	*int*	多头今仓
buy_quantity	*int*	多头持仓
buy_avg_open_price	*float*	多头开仓均价
buy_avg_holding_price	*float*	多头持仓均价
sell_daily_pnl	*float*	空头仓位当日盈亏
sell_pnl	*float*	空头仓位累计盈亏
sell_transaction_cost	*float*	空头费用
closable_sell_quantity	*int*	可平空头持仓
sell_margin	*float*	空头持仓占用保证金
sell_today_quantity	*int*	空头今仓

属性	类型	注释
date	*date*	该仓位数据的日期
sell_quantity	*int*	空头持仓
sell_avg_open_price	*float*	空头开仓均价
sell_avg_holding_price	*float*	空头持仓均价

trades.csv - 所有交易记录的汇总

属性	类型	注释	品种
datetime	datetime	trade发生的时间	
order_book_id	str	合约代码	
symbol	str	合约名称	Optional, 依据不同产品来决定
side	*SIDE*	买/卖	
last_price	float	该Trade成交的价格	
last_quantity	float	该Trade成交的交易量	
transaction_cost	float	tax + commission = transaction_cost	
exec_id	long	trade的primary key，用来identify交易所给回的trade的ID	
position_effect	POSITION_EFFECT	开/平	
commission	float	该Trade产生的交易佣金	
tax	float	该Trade产生的税费	
order_id	long	产生该Trade的order的ID	

Enum类型汇总

ORDER_STATUS - 订单状态

枚举值	说明
PENDING_NEW	待报
ACTIVE	可撤
FILLED	全成
CANCELLED	已撤
REJECTED	拒单

SIDE - 订单方向

枚举值	说明
BUY	买
SELL	卖

ORDER_TYPE - 订单类型

枚举值	说明
MARKET	市价单
LIMIT	限价单

RUN_TYPE - 策略运行类型

枚举值	说明
BACKTEST	回测
PAPER_TRADING	实盘模拟

MATCHING_TYPE - 撮合方式

枚举值	说明
CURRENT_BAR_CLOSE	以当前bar收盘价撮合
NEXT_BAR_OPEN	以下一bar数据开盘价撮合

推荐的第三方Python模块

以下的Python模块我们都在www.ricequant.com 上支持，都经过了一段时间的测试，如果您想更好的研发策略，可以尝试引入到您的本地环境，我们依然建议您在conda虚拟环境中安装这些第三方的Python模块：

模块名	简介	文档链接
talib	TA-Lib 是一个被交易员/程序员常用的金融数据技术分析库。包含了超过150+的技术指标比如ADX,MACD,RSI,Stochastic,Bollinger Bands等	TA-Lib官网
pandas	最流行的Python数据分析库	pandas文档
numpy	numpy是一个Python的科学计算基础库。	numpy文档
scipy	SciPy是一个Python的数学、科学和工程计算的生态系统库。	scipy文档
statsmodels	Statsmodels是一个Python的模块可以让您研究数据，构架统计模型和进行统计测试。功能包括：线性回归模型（Linear regression models）等	statsmodels文档
bisect	Python的排序模块	bisect文档
cmath	提供可以对复数计算的数学模块	cmath文档
collections	提供除了Python内嵌的容器之外的容器种类选择 - dict, list, set 和 tuple	collections文档
sklearn	Python的机器学习模块（machine learning）	sklearn文档
hmmlearn	Python的隐马尔可夫模型（Hidden Markov Models）模块，类似scikit-learn的API	hmmlearn文档
pykalman	超级简单的卡尔曼滤波（Kalman Filter）, Kalman Smoother和EM模块	pykalman文档
cvxopt	cvxopt提供了凸优化（convex optimization）的解的python库。	cvxopt文档
arch	arch提供了Univariate volatility模型，Bootstrapping和Multiple comparison procedures	arch文档
dateutil	dateutil模块提供了对标准的datetime模块的强大的拓展	dateutil文档
datetime		datetime文档
functools		functools文档
heapq		heapq文档
pywt	PyWavelets是一个Python的小波变换的库	pywt文档
tensorflow	Tensor flow is an open source software library for machine intelligence.	tensorflow文档
tushare	国内流行的开源数据库，燥起来吧，各种数据。	tushare网站
pybrain	pybrain是一个流行的机器学习库。PyBrain is a modular Machine Learning Library for Python.	pybrain文档
nltk	一个流行的人类语言分析库。	nltk文档
keras	Theano和Tensorflow的深度学习库。	keras文档
requests	易用的HTTP库	requests文档
bs4	beautifulsoup是网页爬取数据的利器！	beautifulsoupd文档
lxml	处理XML和HTML的最好用的python库	lxml中文文档
urllib	python自带的url处理库	urllib文档
xgboost	速度快效果好的boosting模型	xgboost文档
itertools		itertools文档
math		math文档
pytz		pytz文档
queue		queue文档
random		random文档
re		re文档
time		time文档

模块名	简介	文档链接
array		array文档
copy		copy文档
json		json文档
operator		operator文档
xml		xml文档