# 项目背景

数据统计分析对数据进行收集、整理、及分析的过程。信息化过程中，各个行业将建立起各自的数据中心，未来将基于大数据进行行业分析。

从行业角度，在证券行业通过对大数据建立数学模型发掘股票价格走势；从企业管理角度，产品或项目市场销售情况和各部门成本投入制定相应报表，从中发掘各个企业管理中的短板；从岗位角度，产品经理通过对用户行为建立起数据中心，通过对该数据中心进行分析以判断用户对产品功能的粘粘度。

大数据分析未来将在各行各业崭露头角。为对一个行业或一组数据进行数据分析，不仅仅需要行业专业及管理知识，同时需要相关的工具来帮助收集数据、整理数据。最终将数据以图形化方式展现。

Python作为广泛使用的大数据分析工具，其高效、快捷、易学深受大众的欢迎。

本项目主旨，将以股票数据为基础，学习数据分析过程，通过Python工具实现数据模型，实现机器分析将结果展示。

本项目涉及范围：

1. Python通过tushare获取股票相关数据；
2. 实现数据清理、分析、建模和图形展示。

# 需求描述

本项目最终目标：

1. 使用tushare获取相关股票数据；
2. 存储相关股票数据；
3. 通过图形化方式展现数据。

## 阶段一：实现Python获取tushare数据

1、系统通过股票代码查询获取A股行情数据

2、获取信息有：

股票代码、股票名称、日期、开票价、最高价、收盘价、最低价、成交量、涨跌幅、换手率

3、系统自动每隔3秒获取新数据

4、在终端输入命令显示交易数据信息

## 阶段二：实现程序可配置化

配置化内容包括：股票代码、刷新频率等

1. 系统读取股票代码list文件(stock.txt)

## 阶段三：数据保存至本地

1、将数据保存至mysql或excel等数据库

## 阶段四：数据图形化展现

1. 趋势图

# 功能描述

# 详细设计

## 获取交易数据信息

### 历史行情数据

获取个股历史交易数据（包括均线数据），可以通过参数设置获取日k线、周k线、月k线，以及5分钟、15分钟、30分钟和60分钟k线数据。本接口只能获取近3年的日线数据，适合搭配均线数据进行选股和分析，如果需要全部历史数据，请调用下一个接get\_h\_data()

import tushare as ts

ts.get\_hist\_data('600848') #一次性获取全部日k线数据

* **设定历史数据的时间**

ts.get\_hist\_data('600848',start='2015-01-05',end='2015-01-09')

* **其它**

ts.get\_hist\_data('600848'，ktype='W') #获取周k线数据

ts.get\_hist\_data('600848'，ktype='M') #获取月k线数据

ts.get\_hist\_data('600848'，ktype='5') #获取5分钟k线数据

ts.get\_hist\_data('600848'，ktype='15') #获取15分钟k线数据

ts.get\_hist\_data('600848'，ktype='30') #获取30分钟k线数据

ts.get\_hist\_data('600848'，ktype='60') #获取60分钟k线数据

ts.get\_hist\_data('sh'）#获取上证指数k线数据，其它参数与个股一致，下同

ts.get\_hist\_data('sz'）#获取深圳成指k线数据

ts.get\_hist\_data('hs300'）#获取沪深300指数k线数据

ts.get\_hist\_data('sz50'）#获取上证50指数k线数据

ts.get\_hist\_data('zxb'）#获取中小板指数k线数据

ts.get\_hist\_data('cyb'）#获取创业板指数k线数据

### 复权历史数据

df = ts.get\_stock\_basics()

date = df.ix['600848']['timeToMarket'] #上市日期YYYYMMDD

### 实时行情数据（所有股票）

import tushare as ts

ts.get\_today\_all()

### 获取大盘指数行情

### 大单交易数据

### 获取实时分笔

* **获取单子股票实时数据**

import tushare as ts

df = ts.get\_realtime\_quotes('002410') #Single stock symbol

df[['code','name','price','bid','ask','volume','amount','time']]

* **获取多个股票代码数据**

#symbols from a list

ts.get\_realtime\_quotes(['600848','000980','000981'])

#from a Series

ts.get\_realtime\_quotes(df['code'].tail(10)) #一次获取10个股票的实时分笔数据

### 个股历史分笔数据

import tushare as ts

df = ts.get\_tick\_data('600848',date='2014-01-09')

df.head(10)

### 当日历史分笔

## 图形化显示cmd

import os

os.system('cls') #清除屏幕数据

## 系统每隔3秒自动刷新数据

import time

time.sleep(3)

# 系统功能清单（checklist)

1）系统读取配置文件

配置文件包括内容：股票代码、刷新数据间隔

2）

# 知识点

## 数据分析

### 《深入浅出-数据分析》

#### 分解数据

**确定->分解->评估->决策**

1. ***确定问题及目标***

心智模型？大家对问题及事物的认知。

你对外界的假设和你确信的观点就是你的心智模型；

心智模型应当包括你不了解的因素；

1. ***对所收集的问题进行分类、分组***

比较因子、基准假设

对大问题进行分解及量化

1. ***评估数据***

通过“比较”对数据及假设进行评估总结，回答之前提出的问题。

1. ***决策***

将评估结果形成方案。

## 风险指标

无论是回测还是模型，所有风险指标都只会每天更新一次，也只根据每天收盘后的收益计算，并不考虑每天盘中的收益情况。

### 策略收益（Total Returns）

### 基准收益（Benchmark Returns）

### Alpha（阿尔法）

投资中面临系统风险（即Beta）和非系统性风险（即Alpha），Alpha是投资者获得与市场波动无关的回报。

Alpha=α=Rp−[Rf+βp(Rm−Rf)]Alpha=α=Rp−[Rf+βp(Rm−Rf)]

Rp=策略年化收益率

Rm=基准年化收益率

Rf=无风险利率（默认0.04）

βp=策略beta值

|  |  |
| --- | --- |
| **Alpha值** | **解释** |
| α>0 | 策略相对于风险，获得了超额收益 |
| α=0 | 策略相对于风险，获得了适当收益 |
| α<0 | 策略相对于风险，获得了较少收益 |

### Beta（贝塔）

表示投资的系统性风险，反映了策略对大盘变化的敏感性。

Beta=βp=Cov(Dp,Dm)Var(Dm)

Dp=策略每日收益

Dm=基准每日收益

Cov(Dp,Dm)=策略每日收益与基准每日收益的协方差

Var(Dm)=基准每日收益的方差

|  |  |
| --- | --- |
| **Beta值** | **解释** |
| β<0 | 投资组合和基准的走向通常反方向，如空头头寸类 |
| β=0 | 投资组合和基准的走向没有相关性，如固定收益类 |
| 0<β<1 | 投资组合和基准的走向相同，但是比基准的移动幅度更小 |
| β=1 | 投资组合和基准的走向相同，并且和基准的移动幅度贴近 |
| β>1 | 投资组合和基准的走向相同，但是比基准的移动幅度更大 |

### Sharpe（夏普比率）

表示每承受一单位总风险，会产生多少的超额报酬，可以同时对策略的收益与风险进行综合考虑。

### Sortino（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。

### Information Ratio （信息比率）

衡量单位超额风险带来的超额收益。信息比率越大，说明该策略单位跟踪误差所获得的超额收益越高，因此，信息比率较大的策略的表现要优于信息比率较低的基准。合理的投资目标应该是在承担适度风险下，尽可能追求高信息比率。

### Algorithm Volatility（策略波动率）

用来测量策略的风险性，波动越大代表策略风险越高

### Benchmark Volatility（基准波动率）

用来测量基准的风险性，波动越大代表基准风险越高。

### Max Drawdown（最大回撤）

描述策略可能出现的最糟糕的情况，最极端可能的亏损情况。

### Downside Risk（下行波动率）

策略收益下行波动率。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。

# 程序化交易平台分析

## 竞品

<http://vnpy.org/>