

# 第 7 周 数据可视化与交互 (初级)

## 任务目标

编程的目标是实现 **自动化**, 但是就像我们需要经过 **调试**一步一步写出代码那样, 我们也需要 **可视化与交互** 工具来查看、检查、监控、探索、分析、理解数据, 在“手动”的基础上实现“自动”。终端 (TUI) 是自动化的利器, 但在可视化与交互方面, 终端确实不太擅长。图形用户界面 (GUI) 比较擅长可视化与交互, 但不够跨平台 (指在不同操作系统上获得相似的体验), 也不够开放 (指不容易定制和扩展功能)。基于浏览器 (Browser, 比如 Chrome、Edge、Firefox、Safari 等等) 的 **Web 技术** (比如网页版 App), 不但擅长可视化与交互, 而且跨平台和足够开放, 是特别理想的选择, 但缺点是技术栈 (HTML + CSS + Javascript) 门槛较高, 不适合初学者。

本周将通过案例 (新股 IPO) 介绍两个 Python 生态 (PyPI) 下适合初学者使用的可视化与交互工具 (全部是基于 Web 技术), 分别是 [JupyterLab](#) 和 [Perspective](#)。案例使用的数据将通过 [Tushare](#) 获取。

Fork 第 07 周打卡 仓库至你的名下, 然后将你名下的这个仓库 Clone 到你的本地计算机

用 VS Code 打开项目目录, 新建一个 environment.yml 文件, 指定安装 Python 3.12 和 jupyterlab, 然后运行 conda env create 命令创建 Conda 环境

在项目目录下, 运行 jupyter lab 命令, 启动 后端 (Backend) 服务, 在浏览器里粘贴地址访问 前端 (Frontend) 页面

在 JupyterLab 页面里, 新建一个 Notebook, 改名为 trial-jupyterlab.ipynb, 在里面实践掌握以下功能:

在单元格 (Cell) 里编写 Python 代码, 按 Shift+Enter 运行 Cell 并下移

在单元格 (Cell) 上按 ESC 切换到 命令模式 (command mode), 按 Enter 切换到 编写模式 (edit mode)

在单元格 (Cell) 的命令模式下, 按 j 选择下一个, 按 k 选择上一个, 按 a 在上方添加, 按 b 在下方添加, 按 dd 删除, 按住 Shift 多选, 按 x 剪切, 按 c 复制, 按 v 粘贴, 按 Shift+M 合并, 按 z 撤销, 按 Shift+Z 重做, 按 Shift+L 显示/隐藏代码行号

在单元格 (Cell) 的编写模式下, 按 Ctrl+Shift+- 切分单元格

按按钮显示/隐藏 Minimap

运行单元格 (Cell) 注意序号单调递增

单元格最后一行如果是 表达式 (expression) 且运行后返回的对象不是 None, 则计输出 (Out), 否则只计输入 (In), 序号为 i 的输出, 可以用 \_i 变量来引用

单元格 (Cell) 序号为 \* 表示代码运行中, 尚未返回, 按 ii 可以打断 (KeyboardInterrupt) (类似于终端的 Ctrl+C)

在单元格 (Cell) 的命令模式下, 按 00 重启后端 Python 解释器 (被 Jupyter 称为 Kernel), 重启后需要从上至下重新运行一遍代码 (Shift+Enter), 运行前建议先在菜单里选择 “Edit / Clear Outputs of All Cells” 清空全部页面显示的输出

在单元格 (Cell) 的命令模式下, 按 m 切换至 Markdown 模式, 按 y 切换至 Python 模式 用豆包 (或 DeepSeek 等任何大模型) 生成一段示例 Markdown 代码, 复制粘贴进 Markdown 单元格, 运行以呈现 (Render)

用豆包 (或 DeepSeek 等任何大模型) 生成一段示例 HTML 代码, 复制粘贴进 Markdown 单元格, 运行以呈现 (Render); 注意不支持 CSS

用豆包 (或 DeepSeek 等任何大模型) 生成一段示例 LaTeX 数学公式代码, 复制粘贴进 Markdown 单元格, 运行以呈现 (Render); 注意要用 \$(行内模式) 或 \$\$ (整行模式) 包围  
关闭前端页面, 在后端按 Ctrl+C 打断运行中的服务, 回到 Bash 提示符

## 1. 启动 JupyterLab

在终端运行:

bash

深色版本

jupyter lab

这将打开 JupyterLab 的网页界面 (通常会自动跳转到浏览器)。

---

## 2. 新建一个 Notebook 并重命名

- 点击 "Python 3 (ipykernel)" 或其他内核创建一个新的 Notebook。
  - 右键点击当前标签页顶部的文件名 (默认是 Untitled.ipynb), 选择 "Rename File", 将其改名为 trial-jupyterlab.ipynb。
- 

## 3. 单元格的基本操作

### 编写和运行代码

- 在任意单元格中输入 Python 代码, 例如:

python

深色版本

a = 5

b = 10

a + b

- 按下 Shift+Enter: 执行该单元格并跳到下一个单元格。
- 

## 4. 切换命令模式与编辑模式

- **Esc**: 进入命令模式 (左侧边框变蓝)
  - **Enter**: 回到编辑模式 (可以修改内容)
- 

## 5. 命令模式下的快捷键

快捷键	功能说明
j / k	上/下选择单元格
a	在上方插入新单元格
b	在下方插入新单元格
dd	删除当前单元格
Shift + ↑ / ↓	多选多个单元格
x	剪切所选单元格
c	复制所选单元格
v	粘贴剪贴板中的单元格
Shift+M	合并所选单元格
z	撤销操作

快捷键	功能说明
Shift+Z	重做
Shift+L	显示/隐藏行号

## ✓ 6. 编写模式下的快捷键

- **Ctrl+Shift+-**: 在光标位置将单元格一分为二

## ✓ 7. Minimap 设置

- 点击右上角的按钮或使用菜单栏:
  - View > Show Minimap 切换显示/隐藏 MiniMap

## ✓ 8. 理解 In 和 Out 序号

- 单元格最后一行为表达式且返回非 None, 则输出结果会被记录为 Out[i], 否则只显示 In[i]
- 示例:

python

深色版本

# In[1]

2 + 3

输出:

深色版本

Out[1]: 5

你可以引用这个值:

python

深色版本

result = \_

print(result)

## ✓ 9. 中断正在运行的单元格

- 如果某个单元格长时间未结束 (如死循环), 按下 ii (两次 i) 发送中断信号 (相当于 Ctrl+C)

## ✓ 10. 重启 Kernel

- 按 00 (两个零) 重启内核, 所有变量都会被清除
- 重启后建议:
  - 菜单栏点击 Edit > Clear Outputs of All Cells
  - 然后重新从头运行整个 Notebook

## ✓ 11. Markdown 单元格操作

- 在命令模式下按 m 将单元格切换为 Markdown 模式
- 按 y 切换回 Python 模式

## ✓ 12. 生成并粘贴 Markdown 示例

通过 tushare 软件包下载保存一些数据：

在 Tushare 网站上 注册 并登陆，完善修改个人资料，浏览阅读 平台介绍 和 数据接口

通过定制的推荐链接 <https://tushare.pro/register?reg=gl> 完成注册，将可获赠 2000 平台积分（有效期一年）。积分达到门槛才有数据接口的使用权限，否则需要 付费购买积分（约 200~1000 元/年）才有权限使用数据接口。本课程的量化投资实战案例，将主要将通过 Tushare 平台获取数据，请确保拥有足够积分进行实践。

修改 environment.yml 文件，添加 pip:tushare (注意，conda-forge 没有收录 tushare，只能从 PyPI 安装，参考) 依赖项，运行 conda env update 更新 Conda 环境

在终端 (Terminal) 激活 week07 Conda 环境，运行 ipython 命令启动 IPython 交互界面 (IPython 是 Jupyter 项目的一部份，ipython 是 jupyterlab 的依赖项之一)

在 IPython 提示符下，运行下面的 Python 代码设置 Tushare Token

```
import tushare as ts
```

```
ts.set_token("****") # 将 **** 修改成你自己的 Token 字符串
```

其中 \*\*\* 要替换成你在 Tushare 平台上的 接口 TOKEN —— 复制粘贴即可。运行 set\_token 函数会把 Token 字符串保存在 ~/tk.csv 文件里，今后每次使用 tushare 软件包请求数据时都会自动读取并发送 Token，不需要反复设置。

按 Ctrl+D 结束前面的 IPython 进程，再重新启动一个新的 IPython 进程，运行下面的 Python 代码向 Tushare 服务器请求 IPO 新股列表 数据，并保存在本地

```
import tushare as ts
```

```
pro = ts.pro_api()  
df = pro.new_share()  
df.to_parquet("new_share.parquet")
```

其中请求数据函数返回的对象 df 是 pandas.DataFrame 类型，调用其 to\_parquet 方法能够将内存 (memory) 中的 DataFrame 数据按照 Parquet 格式 (Parquet 是大数据领域的首选格式，已经成为业界标准) 序列化 (serialize) 为字节串 (bytes) 保存到磁盘 (disk)。

询问豆包 (或 DeepSeek 等任何大模型)，初步了解 Parquet 格式和 CSV 格式的特点和适用领域

new\_share 接口只需要 120 积分，如果你有 2000 积分，可以采用与上面类似的方法访问 stock\_basic 接口，并将数据保存为 stock\_basic.parquet 文件 (注意，需要指定 fields 参数获取全部字段)。如果积分暂时不够，可以在终端运行下面的命令，从我们开源的 课程仓库 下载数据文件到你的本地

```
curl -O https://raw.gitcode.com/cuebfintech/courses/blobs/8fc08f7bc4dbbf17d356234472795e59c7b9ce2f/stock\_basic.parquet
```

通过 perspective-python 软件包查看 polars.DataFrame 数据，实践交互式可视化：

修改 environment.yml 文件，添加 perspective-python 和 polars 依赖项，运行 conda env update 更新 Conda 环境

启动 JupyterLab，新建一个 Notebook，改名为 trial-perspective.ipynb

调用 polars.read\_parquet 函数，分别读取磁盘 (disk) 中的 new\_share.parquet 文件和 stock\_basic.parquet 文件，得到内存 (memory) 中的 polars.DataFrame 对象，命名为 d1 和 d2

进行适当的列变换，尤其是要把实际上是日期类型的列，从 polars.String() 类型转换为 polars.Date() 类型

把 d1 或 d2 作为参数传递给 perspective.widget.PerspectiveWidget 类型进行初始化，返回的对象会呈现在 NoteBook 的 Output 里

### 第一步：更新 Conda 环境

#### 修改 environment.yml

在你的项目目录中找到或新建一个 environment.yml 文件，并添加以下内容：

yaml

深色版本

```
name: perspective-env
```

channels:

```
- conda-forge
```

dependencies:

```
- python=3.10
```

```
- jupyterlab
```

```
- polars
```

```
- pyarrow # 可选，用于 parquet 支持（如果未默认安装）
```

```
- perspective-python
```

如果你已经有一个环境文件，请确保 perspective-python 和 polars 被加入其中。

#### 更新 Conda 环境

在终端执行：

bash

深色版本

```
conda env update -f environment.yml --prune
```

激活环境：

bash

深色版本

```
conda activate perspective-env
```

---

### ✓ 第二步：启动 JupyterLab 并创建 Notebook

bash

深色版本

```
jupyter lab
```

- 在 JupyterLab 中点击 "Python 3 (ipykernel)" 创建一个新的 Notebook。
  - 右键点击顶部标签页，重命名为 trial-perspective.ipynb。
- 

### ✓ 第三步：读取 Parquet 文件并进行类型转换

在 Notebook 的第一个 Cell 中输入以下代码：

python

深色版本

```
import polars as pl
from datetime import date
```

```
# 读取 parquet 文件
d1 = pl.read_parquet("new_share.parquet")
d2 = pl.read_parquet("stock_basic.parquet")
```

```
# 查看前几行数据（可选）
print("d1 schema:")
print(d1.schema)
print("\nd2 schema:")
print(d2.schema)
```

运行该单元格以加载数据。假设输出显示某些列是字符串但实际上是日期类型（如 "2024-01-01" 格式），我们可以进行类型转换。

---

### ✓ 第四步：将字符串列转换为日期类型

假设 d1 中有列名 ann\_date 是日期字符串，我们将其转为 pl.Date() 类型：

python

深色版本

```
# 示例：将 d1 中 ann_date 列从字符串转为日期
```

```
d1 = d1.with_columns(
    pl.col("ann_date").str.strptime(pl.Date, fmt="%Y-%m-%d").alias("ann_date")
)
```

```
# 同样处理 d2 中可能存在的日期列
```

```
# d2 = d2.with_columns(
#     pl.col("list_date").str.strptime(pl.Date, fmt="%Y-%m-%d").alias("list_date")
# )
```

```
# 查看转换后的 schema
```

```
print("d1 schema after conversion:")
```

```
print(d1.schema)
```

---

#### ✓ 第五步：使用 PerspectiveWidget 进行交互式可视化

现在我们将 d1 或 d2 数据传入 PerspectiveWidget 来展示交互式表格。

python

深色版本

```
from perspective import PerspectiveWidget
```

```
# 使用 PerspectiveWidget 展示 d1 数据
```

```
widget = PerspectiveWidget(d1)
```

```
widget
```

运行这个单元格后，你应该在 JupyterLab 输出区域看到一个交互式的数据表格，可以排序、筛选、分组等。

在 PerspectiveWidget 默认的 Datagrid 视图下，尝试实践：

修改各种列数据类型（文本、数值、日期）的显示风格（style）

设置 Group By 选项，选择某些列作为分组依据（纵向排列），选择其他某些列进行汇总（注意汇总方式有多种函数选项）

设置 Split By 选项，选择某些列作为拆分依据（横向排列）

设置 Order By 选项，选择某些列作为排序依据（注意可以切换 升序/降序）

设置 Where 选项，选择某些列，进一步设置条件，进行数据行（观测）方向的过滤

设置 Columns 选项，选择要显示的数据列（变量），及其显示的先后顺序

在 All Columns 部分，是能够显示但没有显示的数据列（变量），可以点击 NEW COLUMN 添加衍生计算出的新列，需要用 ExprTK 语言书写表达式代码，变量名用双引号（"）包围，字符串用单引号（'）包围

在 PerspectiveWidget 图形界面依靠鼠标（手动）所做的设置（configure），可以导出代码，根据导出的代码，可以修改我们的代码，使得我们运行代码直接就能得到我们所需要的视图（自动化）

在 PerspectiveWidget 的右上方有按钮，可以把图形界面的数据或设置（configure）导出（export）为文件，或复制（copy）到剪贴板

把设置（config.json）复制到剪贴板，粘贴进 Notebook Cell，保存成字符串（str）

也可以把设置（config.json）导出为文件，用 pathlib.Path.read\_text 方法从文件读取出字符串（str）

可以用 json.loads 函数将无结构的（unstructured）字符串（str）解析为有结构的（structured）Python 字典（dict），这样就容易在 Notebook 里美化呈现，也容易进一步通过 Python 代码访问内部的具体设置

也可以把复制到剪贴板的 JSON 字符串，粘贴进某个在线的 JSON 工具网站（比如 链接）进行美化

根据导出的设置代码，在初始化（init）PerspectiveWidget 类型时，传入适当的参数进行设置，运行代码，观察是否符合我们的期望

把 PerspectiveWidget 切换为 Treemap 视图，尝试设置各种选项（configure），观察数据可视化的实际效果

Treemap (树形结构图) 用不同大小的矩形来体现数据的分类占比构成情况，还可以用矩形的颜色来体现第二个维度的数据（文本或数值都可以）

点击 [链接 1](#) 或 [链接 2](#) 学习了解更多 Treemap 的概念、适用情形以及实现代码

把 PerspectiveWidget 切换为 Y Bar 视图，尝试设置各种选项 (configure)，观察数据可视化的实际效果

Y Bar (条形图/柱状图) 的横轴 (不同的条形) 是第一个维度，用 Group By 控制，纵轴 (条形的高度) 是第二个维度，用 Y Axis 控制 (支持多变量并列显示)，还可以把每个条形进一步拆分为多个颜色，用 Split By 控制

点击 [链接 1](#) 或 [链接 2](#) 学习了解更多 Bar Chart 的概念、适用情形以及实现代码

Y Bar 视图还可以用来实现一类很重要的统计制图 —— 直方图 (histogram)。对于数据表中的某一列连续型数值变量（比如新股发行的市盈率 pe），我们经常希望观察其 分布 (distribution)。可以用 bucket 函数对连续变量进行“分桶”（比如表达式 `bucket("pe", 10)`），生成一个新的离散变量（比如命名为 `bucket_pe`），然后把离散变量设置为 Y Bar 的横轴 (Group By)，把任意其他一列变量用 count (计数) 函数汇总，设置为纵轴 (Y Axis)。这样看到的就是直方图。“分桶”在有的地方也叫“分箱”(bin)，其粒度大小需要根据数据适当调节。把 PerspectiveWidget 切换为 Y Line 视图，尝试设置各种选项 (configure)，观察数据可视化的实际效果

Y Line (折线图) 常用来绘制时间序列，横轴通常是时间，用 Group By 控制，纵轴 (折线的 Y 坐标) 通常是连续型数值变量（经过汇总），用 Y Axis 控制 (支持多序列同时显示)，还可以进一步拆分为多条序列，用 Split By 控制

点击 [链接 1](#) 或 [链接 2](#) 学习了解更多 Line Chart 的概念、适用情形以及实现代码

使用我们的示例数据，可以尝试观察最近几年 A 股 IPO 市场的 融资额 (funds) 与 市盈率 (pe) 变化情况。为了加深对数据的 理解 和 验证，可以询问豆包（或 DeepSeek 等任何大模型），在某个时间段内发生了哪些影响 A 股（或 IPO）的重大国内外财经事件，由此加强我们对现实背景的理解

也可以使用示例数据，观察对比最近几年不同交易所 (exchange) 或市场 (market) 的平均中签率 (ballot) 情况

把 PerspectiveWidget 切换为 X/Y Scatter 视图，尝试设置各种选项 (configure)，观察数据可视化的实际效果

X/Y Scatter (散点图) 常用来观察两个数值型连续变量之间的相关关系 (correlation)。数据首先可以进行分组汇总，每一个组对应一个散点，用 Group By 控制。然后把两个连续型数值变量分别设置为 X Axis 和 Y Axis，其汇总数值将作为每个散点的坐标

点击 [链接 1](#) 或 [链接 2](#) 学习了解更多 Scatter Plot 的概念、适用情形以及实现代码

散点的分布如果特别不均匀，则意味着变量单位可能有问题，或者需要经过变换（比如取对数）

散点的分布如果杂乱无规律，则意味着 X 与 Y 没有相关性

散点的分布如果看起来能够拟合成一条直线（即回归线，regression），则意味着 X 与 Y 具有正的或负的相关性，意味着可能存在某些规律

散点图上可以进一步体现更多的变量维度，比如可以把更多变量映射为散点的不同颜色 (Color)、大小 (Size)、符号 (Symbol)、标签 (Label)、提示框 (Tooltip) 等

我们经常还可以把用于分类的类别变量 (类别不宜太多) 设置为 Split By, 从而把一个散点图拆分为多个小散点图 (small multiple), 从而更细致地观察是否存在规律

在我们的示例数据中, 融资额 (funds)、市盈率 (pe)、中签率 (ballot) 是数值型连续变量, 适合用散点图观察他们的规律, 散点可以以个股为单位 (不汇总), 也可以按 行业 (industry) 汇总, 或者按 上市时间 (ipo\_date) 汇总 (每月分桶), 都可以大胆尝试探索