# 指南

### 1. 面向的读者

《城市空间数据分析方法》可用于的研究专业方向包括：城市规划、风景园林、建筑、生态、地理信息等专业。人群为高校在校学生和研究者、企事业的规划设计师等。

### 2. 电脑的配置与库的安装

#### 电脑的配置

本书涉及到深度学习，为了增加运算速度，用到GPU。关于选择哪种GPU可以查看相关信息。作者所使用的配置为（仅作参考）：

OMEN by HP Laptop 15  
Processor - Intel(R) Core(TM) i9-9880H CPU @2.30GHz   
Installed RAM - 64.0 GB (63.9 GB usable)  
System type - 64-bit operating system, x64-based processor  
  
Display adapter - NVIDA GeForce RTX 2080 with Max-Q Design

建议内存最低32.0GB，最好64.0GB以上，或者更高。如果内存较低，则考虑分批读取数据处理后，再合并等方法。善用内存管理，数据读写方式，例如用[HDF5](https://docs.h5py.org/en/stable/)①等格式。

城市空间数据分析通常涉及到海量数据处理，例如芝加哥城三维激光雷达数据约有1.41TB。如果使用的是笔记本电脑，最好配置容量大的外置硬盘。虽然会影响到读写速度，但是避免了对笔记本自身存储空间的占用。

#### 库的安装

[Anaconda](https://www.anaconda.com/)集成环境②，使得Python库的安装，尤其处理库之间的依赖关系变得轻松。虽然Anaconda提供了非常友好的库管理环境，但还是会有个别库之间存在冲突，有可能造成库安装失败，甚至已经安装的库也可能出现问题无法调入，无形之中增加了学习的时间成本。因此，在开始本书代码之前，不要在一个环境之下一气安装所有需要的库，而应是跟随代码的编写，由提示再行逐个安装。

同时，对于某一项目工程，最好仅建立支持本项目的新环境，不安装未使用到的库，使集成环境轻便，在网络环境部署时也不会冗余，减少出错几率。对于自行练习或者不是很繁重的项目任务，只要环境能够支持，代码运行不崩溃就可以。

### 3. 如何学习

#### 善用print()

print()是Python语言中使用最为频繁的语句。在代码编写、调试过程中，不断的用该语句来查看数据的结构、值和变化，变量所代表的内容，监测程序进程及显示计算结果等。通过print()实时查看数据反馈，才可知道是否实现了代码编写的目的，并做出反馈。善用print()是用Python写代码的基础。

#### 充分利用搜索引擎

代码的世界摸不到边际，不可能记住所有方法函数；而且代码库在不断的更新完善，记住的不一定是对的。学习代码关键是养成写代码的习惯，训练写代码的思维，最终借助这个工具来解决实际的问题。那些无以计数的库、涉及到的函数、方法、属性等内容，通常是在写代码过程中，充分利用搜索引擎和对应库的官网在线文档来查找，其中一个不错的问答平台是[StackOverflow](https://stackoverflow.com/)③。经常用到的方法函数会不知不觉的被记住，或有意识的来记忆经常用到的方法函数。即使对使用过的方法仅存有蛛丝马迹的印象，通常也可以依据这些线索快速的再行搜索寻找答案。

#### 由案例来学习

对方法函数再多的解释不如一个简单的案例示范来的清晰明了。通常查看某一个方法函数及属性的功用，最好的途径是直接搜索案例来查看数据结构的变化，往往不需要阅读相关解释文字就可以从数据的前后变化中获悉该方法的作用，再辅助阅读文字说明进一步深入理解或者对比查看仍旧不明晰地方的解释。

#### 库的学习

城市空间数据分析方法在各类实验中使用有大量的相关库，例如④:

* 数据结构：[NumPy](https://numpy.org/)，[pandas](https://pandas.pydata.org/)；
* 科学计算：[SciPy](https://www.scipy.org/)，[SymPy](https://www.sympy.org/en/index.html);
* 统计推断：[statsmodels](https://www.statsmodels.org/stable/index.html)；
* 机器学习：[scikit-learn](https://scikit-learn.org/stable/)；
* 深度学习：[PyTorch](https://pytorch.org/)，[TensorFLow](https://www.tensorflow.org/?gclid=EAIaIQobChMI8_XU1qrS-gIVz6iWCh1pxAGSEAAYASAAEgIKYfD_BwE)，[TorchVision](https://pytorch.org/vision/stable/index.html)，[segmentation-models-pytorch](https://smp.readthedocs.io/en/stable/index.html)，[Kornia](https://github.com/kornia/kornia)，[TorchGeo](https://torchgeo.readthedocs.io/en/stable/user/installation.html)，[Transformers](https://github.com/huggingface/transformers)，[pytorch-lightning](https://pytorch-lightning.readthedocs.io/en/latest/);
* GIS工具：[GDAL](https://gdal.org/)，[GeoPandas](https://geopandas.org/)，[rasterstats](https://pythonhosted.org/rasterstats/)，[rasterio](https://rasterio.readthedocs.io/en/latest/)，[EarthPy](https://earthpy.readthedocs.io/en/latest/)，[PySAL](https://pysal.org/)，[Shapely](https://shapely.readthedocs.io/en/latest/manual.html)，[Fiona](https://github.com/Toblerity/Fiona)，[rio-tiler](https://cogeotiff.github.io/rio-tiler/)，[rioxarray](https://github.com/corteva/rioxarray)；[Xarray](https://docs.xarray.dev/en/stable/index.html);
* 复杂网络：[NetworkX](https://networkx.github.io/)，[igraph](https://igraph.readthedocs.io/en/stable/index.html)；
* 点云处理：[PDAL](https://pdal.io/)；[PCL](https://pointclouds.org/)；[open3D](http://www.open3d.org/docs/release/introduction.html)
* 影像视觉：[scikit-image](https://scikit-image.org/)，[OpenCV](https://opencv.org/)，[Pillow](https://python-pillow.org/);
* 数据可视化：[matplotlib](https://matplotlib.org/)，[Plotly](https://plotly.com/)，[seaborn](https://seaborn.pydata.org/)，[bokeh](https://docs.bokeh.org/en/latest/index.html)，[VTK](https://vtk.org/)，[gradio](https://gradio.app/)；
* Web应用：[flask](https://flask.palletsprojects.com/en/1.1.x/);
* 数据库：[sqlite3](https://docs.python.org/3/library/sqlite3.html)；
* GUI：[tkinter](https://docs.python.org/3/library/tkinter.html)，[PyQt](https://riverbankcomputing.com/software/pyqt/)，[Kivy](https://kivy.org/)，[Pygame](https://www.pygame.org/news)；
* 其它类：[re](https://docs.python.org/3/library/re.html)，[pathlib](https://docs.python.org/3/library/pathlib.html)，[itertools](https://docs.python.org/3/library/itertools.html)，[urllib](https://docs.python.org/3/library/urllib.html)。

很多都为大体量库，单独学习每个库都会花费一定的时间。在阅读该书时，如果应用到的库之前没有接触过，一种方式是，提前快速的学习各个库官方提供的教程(tutorial)，配合手册（manual）快速掌握库的主要结构，再继续阅读本书实验；另一种方式是，为了不打断阅读的连贯性，可以只查找用到该库相关方法部分的内容。

一般每个库的内容都会非常多，通常不需要一一学习，当用到时再根据需要有针对性和有目的性的查阅。但是也有些库是需要系统的学习，例如scikit-learn、PyTorch、PyQt等重量型的库。

#### 用代码学习数学公式

在科学中，公式是用符号表达信息的一种简明方式。数学公式通常使用特定逻辑语言的符号和形成规则构建事物间的联系，以更好的理解事物的本质和内涵。当用公式代替文字阐述方法时，可以更直观的表达内在逻辑和变化规律。然而很多时候，公式及文字阐述让人费解，读不懂作者所要表达的含义和目的，尤其包含有大量公式，而给出的案例也都只是“白纸黑字”的文字论述时。这是因为很难去调整参数再实验作者的案例。但是用Python等语言进行数据分析，可以用print()查看每一步数据的变化，尤其对于不易理解的地方，通过比较前后的数据变化，容易发现内在的逻辑，理解用公式和文字描述不易阐明的地方。这也是为什么本书的所有阐述都基于代码，当公式和文字阐释不易理解的时候，只要逐行的运行代码，往往能明了。同时尽可能的将数据及其变化以图表数据可视化的方式表达，显现事物中隐藏的联系，让问题更容易浮出水面。

#### 避免重复造轮子

“避免重复造轮子”是程序员的一个“座右铭”。当然每个人所持的观点不尽相同，但是可以肯定的是，没有必要从零开始搭建所有项目内容，这如同再写Python的所有库，甚至Python核心内容的东西。例如scikit-learn、PyTorch集成了大量的模型算法，通常直接调用，不会再重复写这些代码。本书阐述过程中，会有意识地对最为基本的部分按照计算公式重写代码，目的是认识方法的内在逻辑，而实际的项目则无需如此。

#### 对于那些绕弯的代码和理解算法

读代码不是读代码本身，而是读代码解决问题的逻辑策略。有些代码比较简单，例如数组的加减乘除；但是有些代码往往“不知其所以然”，却解决了某一问题。那么这里面可能就有一些非常巧妙的处理手段，例如梯度下降法、主成分分析、生成对抗网络，及各类算法等。很多时候解决问题的方法越巧妙，逻辑方法越不同于常规，因此阅读起来可能不那么容易。对于理解这些绕弯的代码，除了要不断查看数据变化，甚至要结合可交互图表动态分析，也可以网络搜索寻找更清晰易懂的解释方式，或者自行寻找到易于理解的途径，找到作者所发现的方法的内在逻辑。

库中的方法通常是解决某一问题算法逻辑的实现，一般而言直接调用该方法，避免重复造轮子，这毋庸置疑。但是，有时很多算法能够启发我们，从而对某些问题提出新的解决策略，因此有意识的去理解一些算法是必要的。

#### 对于“千行+”复杂代码的学习

能够一下子看懂百千行代码的可能性不大，尤其这些代码以类的形式出现时，各类参数、变量，甚至函数自身在各个函数下不断的跳跃，乃至分支、合并，不断产生新的变量。第1阶段，捋清楚初始化的条件，及初始化参数传输的路径是首先需要明确的事情，这个过程的难易视代码的复杂程度决定。一旦清楚了代码运行的流程，即语句执行的顺序（有顺序，也可能并行），可以将其记录下来，方便进一步的分析；进而开始第2阶段的代码分析，因为已经清楚了流程，那么就从开始逐行的运行代码，打印数据，理解每一行代码所要解决的是哪些问题。这个过程与第1阶段的捋清代码运行流程可以同时进行，边梳理，边运行行查看数据，边分析理解代码解决问题的逻辑。在逐行分析代码时，往往会遇到那些绕弯的代码（算法），如果一时无法理解，在不影响后续代码运行前提下，可以先绕过，直接获取用于运行后续代码行的数据，最后再重点分析这些绕弯的代码。如果无法绕过的话，就只能花费精力先来解决这个问题。

#### 参考文献的推荐

城市空间数据分析方法是多学科的融合，这包括作为分析工具的Python编程语言知识，及城乡规划、建筑和风景园林等的专业知识，和数学（统计学、微积分、线性代数），数据库，地理信息系统，生态学等大类和其下更为繁杂的小类细分等知识。虽然涉猎内容众多，但实际上只是在解决某一问题时用到所需要的相关知识。当阅读该书时，如果某处的知识点不是很明白，可以查找相关文献补充阅读。在每一部分，也提供了所参考的文献，必要时可以翻看理解或印证。

注释（Notes）：

① Hierarchical Data Format，HDF是一组文件格式（HDF4，HDF5），用于大数据的存储和组织。其中h5py包（<https://docs.h5py.org/en/stable/>）是HDF5二进制数据格式的一个Python接口。

② Anaconda，用于科学计算（数据科学、机器学习应用、大数据处理、预测分析等）的Python和R编程语言的环境管理器，旨在简化包管理和部署（<https://www.anaconda.com/>）。

③ Stack Overflow，是一个面向专业程序员和爱好者，关于计算机编程广泛主题的问答网站（<https://stackoverflow.com/>）。

④ 例举的相关库，印刷版本未给出网页链接时，可以通过搜索引擎检索。