2025~2026 学年第一学期水利遥感结课作业

一、基于 R 语言的 Landsat 8 遥感影像处理与分析

二、论文《2001-2019 年长江中下游农业干旱遥感监测及植被敏感性分析》阅读报告

三、对水利遥感课程的感想、意见、建议等

姓　　名：	杨 丹 阳
学　　号：	2023302131259
学　　校：	武汉大学
学　　院：	遥感信息工程学院
专　　业：	遥感科学与技术
课　　程：	水利遥感
指导教师：	查 元 源

二〇二五 年 十一 月

武漢大學
WUHAN UNIVERSITY

武漢大學遥感信息工程学院
School of Remote Sensing and Information Engineering,Wuhan University
笃志 敦行 和协 拓新

本科课程论文（设计）

——2025~2026 学年第一学期水利遥感实践课程报告

基于 R 语言的 Landsat 8 遥感影像处理与分析

姓　　名：	杨 丹 阳
学　　号：	2023302131259
学　　校：	武汉大学
学　　院：	遥感信息工程学院
专　　业：	遥感科学与技术
课　　程：	水利遥感
指导教师：	查 元 源

二〇二五 年 十一 月

原创性声明

本人郑重声明：所呈交的论文（设计），是本人在指导教师的指导下，严格按照学校和学院有关规定完成的。除文中已经标明引用的内容外，本论文（设计）不包含任何其他个人或集体已发表及撰写的研究成果。对本论文（设计）做出贡献的个人和集体，均已在文中以明确方式标明。本人承诺在论文（设计）工作过程中没有伪造数据等行为。若在本论文（设计）中有侵犯任何方面知识产权的行为，由本人承担相应的法律责任。

作者签名：杨丹阳

日　　期：2025 年 11 月 1 日

版权使用授权书

本人完全了解武汉大学有权保留并向有关部门或机构送交本论文（设计）的复印件和电子版，允许本论文（设计）被查阅和借阅。本人授权武汉大学将本论文的全部或部分内容编入有关数据进行检索和传播，可以采用影印、缩印或扫描等复制手段保存和汇编本论文（设计）。

作者签名：杨丹阳

日　　期：2025 年 11 月 1 日

摘　要

本报告围绕湖北省孝感市孝南区杨店镇，构建了一个以 R 语言为主、结合 Google Earth Engine（GEE）与 GeoScene Pro 的 Landsat 8 L2（C2/T1）遥感影像处理与分析流程。方法上，首先利用 GEE 基于 ROI 统计云量与 QA_PIXEL 掩膜，自动筛选并导出低云量影像；在本地以 R 完成波段解析、真/假彩渲染、通道重命名、规则裁剪与矢量掩膜。随后计算 NDVI、NDWI、NDBI、SAVI 等指数，并通过 Otsu 法实现水体、建成区与植被的阈值掩膜与多级分级。在地物分类方面，先以 SR_B2~SR_B7 构建特征栈，开展 K-means 无监督聚类与 3×3 众数滤波后处理，再以人工样区训练随机森林（Random Forest）监督分类，获得>99% 的总体精度和极低 OOB 误差。时序分析方面，选取 2015–2023 年夏/冬代表影像，构建指数时间序列，采用闭式解像元级线性趋势计算斜率、R^2 与均值，并输出首末期差值、相对变化率及基于四分位的变化类别图。结果显示：研究区植被（NDVI/SAVI）年际总体稳定，季节性信号显著；水分状况（NDWI）整体平稳、局地轻微干化；建成区强度（NDBI）未呈现持续、成带式扩张，以零散小斑块变化为主。本文流程自洽、复现性强，可在县/镇尺度推广至农业监测、国土巡查与水利管理等应用场景。

关键词：Landsat 8；R 语言；Google Earth Engine；Otsu 阈值；K-means；随机森林；变化检测

ABSTRACT

This report develops an end-to-end workflow for Landsat 8 Level-2 (Collection 2, Tier 1) remote sensing over Yangdian Town (Xiaonan District, Xiaogan City, Hubei, China), integrating R as the primary environment with Google Earth Engine (GEE) and GeoScene Pro. First, low-cloud scenes are automatically screened and exported from GEE using ROI-based cloud fraction and QA_PIXEL masking. In R, we perform band parsing, true/false color rendering, standardized band renaming, rectangular cropping, and vector masking. Core spectral indices (NDVI, NDWI, NDBI, SAVI) are computed, followed by Otsu-based thresholding for water, built-up areas, and vegetation, including multi-level NDVI stratification. For land-cover mapping, we use SR_B2 – SR_B7 as features to conduct K-means clustering with 3×3 modal filtering, and train a Random Forest classifier with manually delineated training polygons, achieving >99% overall accuracy and very low OOB error. For change analysis, we build 2015 – 2023 summer/winter index time series and compute per-pixel linear-trend slope, R^2, and mean using a closed-form solution, along with first – last differences, relative change, and quartile-based change categories. Results indicate overall stability of vegetation (NDVI/SAVI) with strong seasonality, generally steady moisture conditions (NDWI) with slight localized drying, and no persistent corridor-like expansion in built-up intensity (NDBI), which changes mainly as scattered small patches. The workflow is coherent, reproducible, and transferable to county/town scales for agricultural monitoring, land stewardship, and water-resources applications.

Keywords: Landsat 8; R language; Google Earth Engine; Otsu thresholding; K-means; Random Forest; Change detection

目　录

1 实验背景

1.1 实验背景与目的

卫星遥感作为地球系统观测的重要技术手段，凭借其跨尺度、连续化和可重复获取的优势，在生态环境监测、土地利用/覆被变化分析、水资源评估以及城乡时空扩张研究等领域发挥着关键作用[1]。近年来，USGS/ESA 数据开放政策与云计算生态（如 Google Earth Engine、AWS 公共数据集等）的成熟，使得大范围、多时相遥感数据的检索、获取与预处理成本显著降低[2]。Landsat 计划自上世纪以来稳定的时间序列积累为多时相变化检测与趋势诊断提供了长期、同质且可比的观测基础；以 Landsat 8 为代表的中分辨率多光谱数据具备可见光、近红外与短波红外等关键波段配置（图 1-1），能够用于在光谱空间上区分植被、水体、裸地与建设用地等典型地物类型[3]。

Landsat 8	
Band description (30 m native resolution unless otherwise denoted)	Wavelength (μm)
Band 1 – blue	0.43–0.45
Band 2 – blue	0.45–0.51
Band 3 – green	0.53–0.59
Band 4 – red	0.64–0.67
Band 5 – near infrared	0.85–0.88
Band 6 – shortwave infrared	1.57–1.65
Band 7 – shortwave infrared	2.11–2.29
Band 8 – panchromatic (15 m)	0.50–0.68
Band 9 – cirrus	1.36–1.38
Band 10 – thermal Infrared (100 m)	10.60–11.19
Band 11 – thermal Infrared (100 m)	11.50–12.51

图 1-1 Landsat 8 的部分波段信息

小城镇作为城乡过渡带，其季节性耕作、建设用地扩张与水体时变特征并存。例如，在长江中下游平原等亚热带季风气候区，河流湖泊在季风期因降水增多而水域扩大，造成耕地与水体边界动态变化[4]；随着城镇的人口增长、商业发展、企业厂房或旅游观光区的投资建设，建筑区域可能有所拓展[5]；等等。这些特征要求在城镇地区的遥感影像分析既需要面向单景影像的精细增强与识别，也需要跨年份的定量趋势评估。

在上述背景下，本实验以 R 语言为核心实现环境，综合使用 Google Earth Engine、GeoScene Pro 等多种资源与工具，实现 Landsat 8 影像的下载与读取、真彩/假彩合成显示、常用遥感指数计算、基于遥感指数的阈值分割、影像的空间裁剪与掩膜、基于影像多波段信息的地物分类、基于多时相指数的像元级趋势估计与首末期差异度量等，实现从原始观测到专题产品的端到端加工，并为土地利用格局演变诊断、农业生产监测与城乡建设管理提供定量依据，从而为资源环境治理与区域可持续发展评估提供可靠支撑。

1.2 研究区域介绍

1.2.1

本次实习的研究区域是湖北省孝感市孝南区杨店镇，也是作者的故乡，地处孝感市东北方向大别山的南麓、孝感市与武汉市接壤的东部边陲，东面濒临黄孝河（界河），与武汉市黄陂区一衣带水；西边紧靠孙家河（浐川河），与孝南区西河镇隔河相望；南部紧连京广线，与孝南区三汉、祝站两镇接壤；北部遥望双峰山，与孝昌县丰山镇、邹岗镇两镇相邻，介于东经114°02′—114°10′，北纬 30°58′—31°06′之间。这里被历史长河浸润、被桃花春风点染千年，127平方公里的土地上，万亩桃林与青砖黛瓦相映，古驿道与现代产业交织，绘就了一幅“花海与古韵共生、生态与人文交融”的春日画卷。

杨店镇古称“斗山铺”，始建于唐代，因驿站蜚声荆楚。北宋文豪苏轼途经此地，留下“花发颜如醉，风吹面不寒”的诗句，斗山铺由此得名“桃花驿”。明朝中期（公元 1436 年），富绅杨廷松在此地施行“六义”善举，扩建市场，将“斗山铺”改名为“杨家店（铺）”，简称“杨店”，沿用至今。1992 年 12 月，时任中共中央总书记江泽民与中共中央政治局候补委员、书记处书记温家宝和国务院委员陈俊生在湖北省委书记关广富、省长郭树言、副书记回良玉的陪同下到访杨店镇，并视察了杨店园艺总场万亩早蜜桃生产基地。漫步镇内，仿佛能听见马蹄声声穿过时空。而每年三月，当万亩桃林沿古驿道次第绽放，粉白花瓣如云似霞，与青瓦白墙的民居相映成趣，便成了“人面桃花相映红”的现实写照。你可在此参与“桃花诗社”笔会，或提笔在漆扇上浸染一朵属于自己的桃花，感受唐宋文人的风雅。

图 1.2-1 杨店镇“飞马传书”镇标

杨店镇以“四季桃乡”为品牌，构建了“春赏花、夏采果、秋研学、冬年庆”的全季旅游体系。春日里，卓尔·桃花驿小镇的智慧桃园中，游客可体验“桃花认养”计划，通过手机APP实时查看果树生长；夏季，水蜜桃采摘节热闹非凡，2024年“杨店水蜜桃”获省级地理标志认证，果肉细腻、汁多味甜；秋季，桃胶制品入选国家“三新食品”目录，游客可亲手熬制一锅养生桃胶羹；冬季，古驿年货市集上，非遗麻糖、米酒与桃木工艺品琳琅满目，年味十足。此外，镇内还推出“1小时桃花生活圈”，联动武汉都市圈，让都市人轻松实现“周末桃乡游”。

图 1.2-2 杨店镇桃林

此外，杨店镇是还“龙灯之乡”，现存传统龙灯127条，每年正月十五的“高龙庙会”震撼人心。巨龙身长百米，需百人协作舞动，配合鼓乐、烟火，场面蔚为壮观。2025年桃花文化旅游节期间，游客可参与非遗漆扇制作、植物拓印等20余项特色项目，或在桃园笔会中与楚剧大师、书画名家共绘春光。

图 1.2-3 央视新闻报道杨店高龙

杨店镇的蜕变，是乡村振兴的生动实践。近年来，全域绿化工程让孙家河至北外环口沿线绿树成荫，污水处理厂与管网覆盖全镇，守护“杨店蓝”；全镇积极开展农村人居环境整治、生活垃圾与污水治理、村庄亮化等工程，对村庄建设及流域治理包括和美乡村建设 7 个自然湾、农村道路提档升级 25.61 公里、刘寨水库清淤工程、河道综合治理工程、灌渠修复改造工程等 5 个子项目；卓尔集团携“桃花驿”IP 深耕农旅融合，刚强农业智慧农田项目落地生根，桃茶品牌化链条与三产融合蓝图渐次展开，沉睡的土地正迸发集约化、现代化的新生机……

杨店欢迎你！不妨暂别都市喧嚣，来杨店镇赴一场千年之约——在桃花树下读一首东坡的诗，在龙灯舞中感受非遗的脉动，在田园间品尝春天的味道。杨店镇，正以万紫千红的样貌与千年文化的底蕴，等待您的到来。

2 实验数据与实验环境

本节是对本次实验所采用的实验数据来源及实验环境配置进行说明。本节内容仅作为实验过程的事实性记录，用于保证实验的可复现性与可验证性。本文所述的算法、模型与代码实现并不限于本节所述的数据与硬件软件环境。

2.1 实验数据

本次实习围绕湖北省孝感市孝南区杨店镇区域，选用多时相的中分辨率卫星遥感数据及配套矢量边界数据，构建从原始下载、预处理到分析应用（真/假彩渲染、指数计算与变化检测）的完整数据链。核心影像源为 USGS 提供的 Landsat 8 Level-2（C2/T1）表面反射率产品，辅以杨店镇范围矢量数据用于裁剪与掩膜。

2.1.1 遥感影像数据

本实习所用的数据集 USGS Landsat 8 Level 2, Collection 2, Tier 1 包含由 Landsat 8 OLI/TIRS 传感器生成的数据得出的经过大气校正的地表反射率和地表温度[6]所构成的遥感影像，这些影像的波段信息为（图 2.1.1-1）：包含 5 个可见光、近红外 (NIR) 波段和 2 个短波红外 (SWIR) 波段，这些波段经过了处理，可生成正射校正的地表反射率；此外还包含 1 个热红外 (TIR) 波段，该波段经过了处理，可生成正射校正的地表温度。它们还包含用于计算 ST 产品和 QA 波段的中间波段。

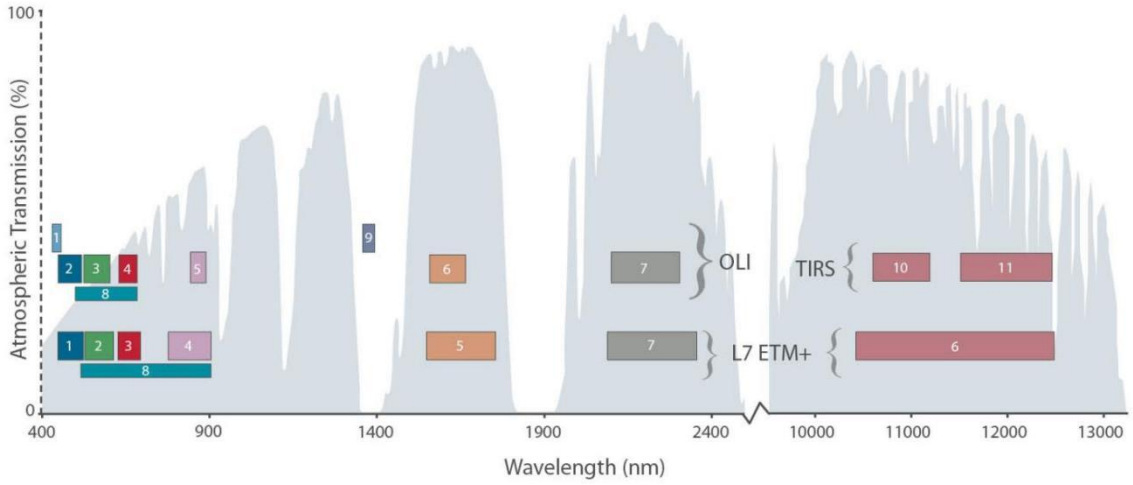


图 2.1.1-1 Landsat-8 波段示意图[7]

Landsat 8 卫星拍摄所得的数据条带会打包成有一定航向和旁向重叠度的“场景”（称为

一景影像），每景影像的标准化空间参考网格覆盖大约 170 公里 × 183 公里的地表区域。

2.1.2 矢量边界数据

所用的湖北省孝感市孝南区杨店镇矢量边界数据来源为 GtiCode 仓库“开源工具包/全国乡镇级行政区划 SHP 数据”[8]，用于在 GEE 中下载影像时进行裁剪（crop）与掩膜（mask），限制影像下载或数据统计范围到行政边界（即研究区轮廓）。矢量与栅格的 CRS 不一致时需进行投影统一。

2.2 软件环境

项目	说明
系统环境	Windows 11（64位）
编程语言	R，版本 4.5.1 (2025-06-13 ucrt)
集成开发环境（IDE）	RStudio 2025.09.1+401"Cucumberleaf Sunflower"(20de356561bd58a6d88927cce948bd076d06e4ca)，发布于 2025-09-23
主要使用的 R 包	imager、grDevices、sp、RColorBrewer、randomForest、sf、utils等
使用的其他软件或平台	Google Earth Engine、GeoScene Pro 3.1等

表 2.2-1 软件环境

建议使用的目录结构：

```
Remote_Sensing_Image_Processing_Using_R_project_root/
│
├── R 脚本文件
│   ├── 4.1_Change_Detection.R                    # 变化检测脚本
│   ├── Remote_Sensing_Image_Processing_Using_R.R  # 主处理脚本
│   ├── tool_visualize_folder_img.R               # 批量可视化工具
│   └── tool_visualize_single_img.R               # 单图可视化工具
│
├── 其他文件
│   ├── memo.txt                                  # 备忘录
│   ├── 报告.lnk                                   # 报告链接
│   └── RS-实践课程材料(OCR).pdf                    # 课程资料
│
├── data/                                         # 原始数据目录
│   ├── YangDian.*                                # 杨店区域矢量数据
│   ├── Yangdian_Zhiniangyuanzi.*                 # 织娘院子区域矢量数据
│   └── supervision_classification/               # 监督分类数据
│       ├── Yangdian_building/                    # 建筑样本
│       ├── Yangdian_vegetation/                  # 植被样本
│       └── Yangdian_water/                       # 水体样本
```

```
│
├── GEE_L8_Yangdian_image/                          # Landsat 8 原始影像
│   └── 多个遥感影像 .tif 文件 (2015-2025 年时间序列)
│
├── GEE_L8_Yangdian_images_vis/                     # 影像可视化结果
│
├── res/                                            # 处理结果
│
└── res_change_detection/                           # 变化检测结果
```

2.3 硬件环境

项目	说明
计算机型号	Legion Y9000P IRX9
处理器（CPU）	Intel® Core™ i9-14900HX（2.20 GHz）
内存（RAM）	32.0 GB (31.7 GB 可用)
显卡（GPU）	Intel® UHD Graphics（集成显卡）、NVIDIA® GeForce RTX 4060 Laptop GPU（独立显卡）
硬盘（磁盘）	SSD（NVMe）1 TB × 2（C: D: 共 1 TB，E: 1 TB）

表 2.3-1 硬件环境

3 实验原理、过程与结果分析

本节将对本次实验中所实施的各项图像处理步骤分别介绍：**①算法的基本原理；②展示实现所用的代码；③展示处理所得的结果；④对实验结果进行分析解释。**为帮助读者更好地理解实验中“原理——代码——结果——分析”的整体逻辑，并加深对各个图像处理功能的认识，本文并未将上述四个部分分别独立成章，而是将相关图像处理的代码组织整理成为具有一定功能意义的模块（因此每个模块中的代码会完成多个不同的图像处理操作，是多个图像处理操作的集合），并在本报告文档中按照这些具体功能模块的顺序，在同一章中逐一展开阐述上述①~④部分的内容。

本节的二级标题（如“3.1 影像数据的获取与筛选”）通常对应一个图像处理的功能模块；三级标题（如“3.1.1 下载全国乡镇级行政区划的矢量边界数据”）通常对应一个包含上述①~④部分的图像处理的功能单元或步骤，若没有再分三级标题，说明二级标题已是一个单独的、无需继续细分的图像处理的功能单元；三级标题下通常不再设子标题。虽然**在正文中为行文简洁，未显式标注“原理”“代码”“结果”“分析”等小节标题**，但读者可在每各图像处理的功能单元获步骤对应的内容中清晰辨识出这四个部分的对应内容。

为了使报告的叙述逻辑更加清晰，第 3 章的分节并未完全按照实习指南中的编号，而是根据代码实现的功能重新组织，且**本报告所实现的功能包括且不限于本次实习要求实现的功能**。

具体展开各图像处理功能模块的实验内容之前，先介绍本次实验中所编写并反复调用的一组通用工具函数。这些函数在整个实验脚本中承担了基础支撑作用，用于简化图像保存等操作，确保实验过程高效、结果管理规范。

为便于后续各图像处理步骤的自动化执行与结果比对，实验在脚本开始部分定义了若干通用函数：

```
##  ----    0.2  工具函数  ----
#  用于给输出图像/图表自动编号并写入  res/  目录，便于后续比对
.counter  <-  new.env(parent  =  emptyenv());  .counter$i  <-  1
next_name  <-  function(label,  ext  =  "png"){
       fn  <-  sprintf("res/%02d_%s.%s",  .counter$i,  label,  ext)
       .counter$i  <-  .counter$i  +  1
       fn
}
save_seq_img  <-  function(im,  label){
       fn  <-  next_name(label,  "png")
       save.image(im,  fn)
       return(fn)
}
save_seq_plot  <-  function(expr_plot,  label,  width=960,  height=720,  res=120){
       #  以  png  设备保存任意绘图表达式（传入大括号块），并自动关闭设备
```

```
        fn <- next_name(label, "png")
        png(fn, width=width, height=height, res=res)
        on.exit(dev.off(), add = TRUE)
        force(expr_plot)
        message("已输出：", fn) # 添加消息提示
        return(fn)
}
```

这段代码中的三个函数分别的功能为：统一生成类似 res/01_xxxl.png 的文件名，并且结果图前面的序号逐次递增；把已有的图像对象保存为 PNG 文件；把任意绘图代码块以 PNG 设备绘制并保存。这三个函数把“命名规范 + 自动编号 + 可靠保存”打包起来，既能保存现成图像对象，也能一键捕获任何绘图代码生成的图，适合批量产出和结果复现的工作流。

此外，为了更好地保存输出结果，便于进行结果分析，本脚本实现了日志记录功能：通过 sink() 将控制台输出重定向到文件 res/log.txt，同时开启 split=TRUE 让信息既在控制台显示，又写入日志文件。脚本中到处使用 message()、print()、warning() 与自定义的 save_seq_plot() 内部 message() 调用，形成统一的日志流水。最后在结尾用 sink() 关闭重定向，完成一次“全流程作业”的完整日志记录。

接下来的各节将基于按照“原理——代码——结果——分析”的逻辑，对图像读取、通道分解、灰度化、滤波、频域分析及几何校正等典型图像处理功能进行系统阐述。

3.1 影像数据的获取与筛选

本次实验主要在 Google Earth Engine（GEE）代码编辑器平台上对实习所需遥感影像进行下载与筛选[9]。其整体流程如下图所示：

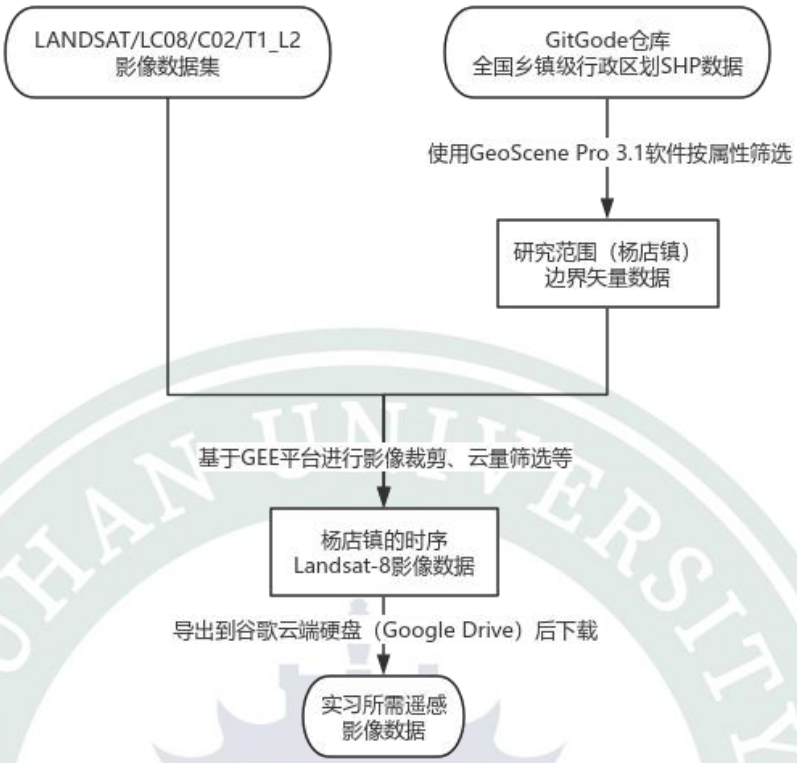


图 3.1-1 影像数据的获取与筛选

3.1.1 下载全国乡镇级行政区划的矢量边界数据

首先需要获取研究区域（湖北省孝感市孝南区杨店镇）的矢量边界，用于在 GEE 中下载影像时进行裁剪（crop）与掩膜（mask）。

首先，在 GtiCode 仓库“开源工具包/全国乡镇级行政区划 SHP 数据”中下载全国乡镇级行政区划的矢量数据：

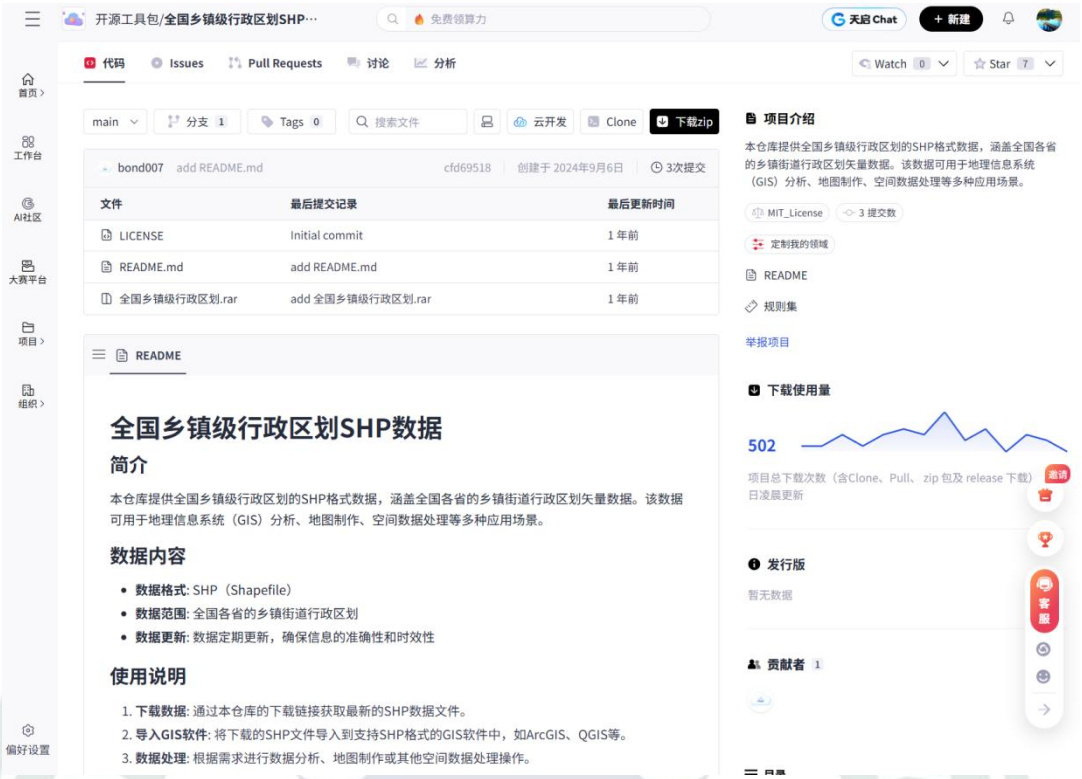


图 3.1.1-1 在GtiCode仓库“全国乡镇级行政区划SHP数据”中下载全国乡镇级行政区划的矢量数据

3.1.2 在 GeoScene Pro 3.1 软件中得到研究区域杨店镇的矢量边界数据

解压缩后，在 GeoScene Pro 3.1 软件中新建工程，打开所得矢量数据 Town.shp：

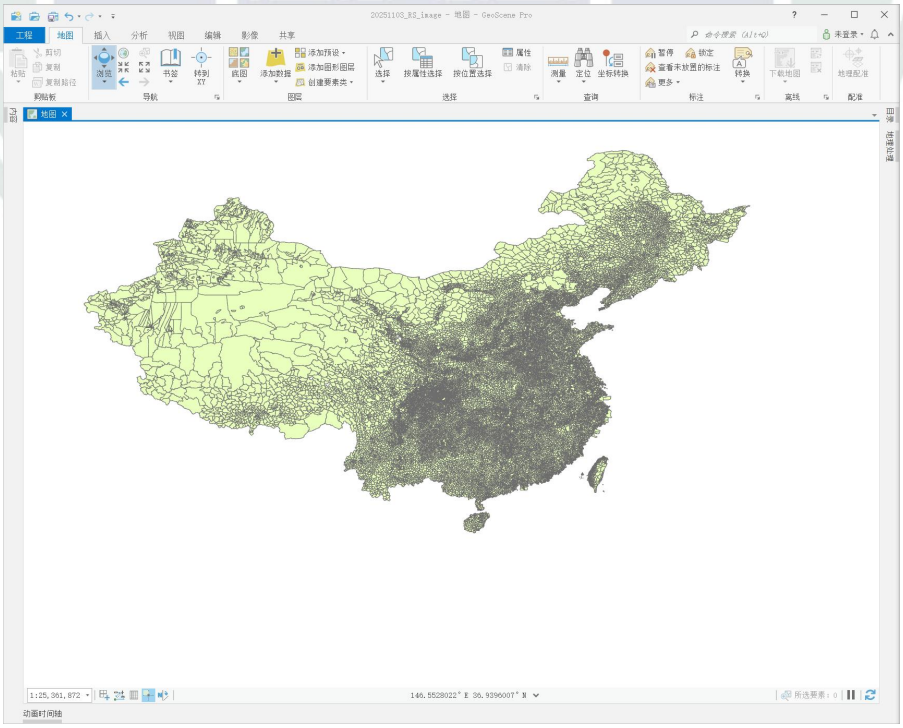


图 3.1.2-1 在GeoScene Pro 3.1软件中打开所得全国乡镇级行政区划的矢量数据

打开左侧的“内容”，右键点击 Town 对象，打开其属性表：

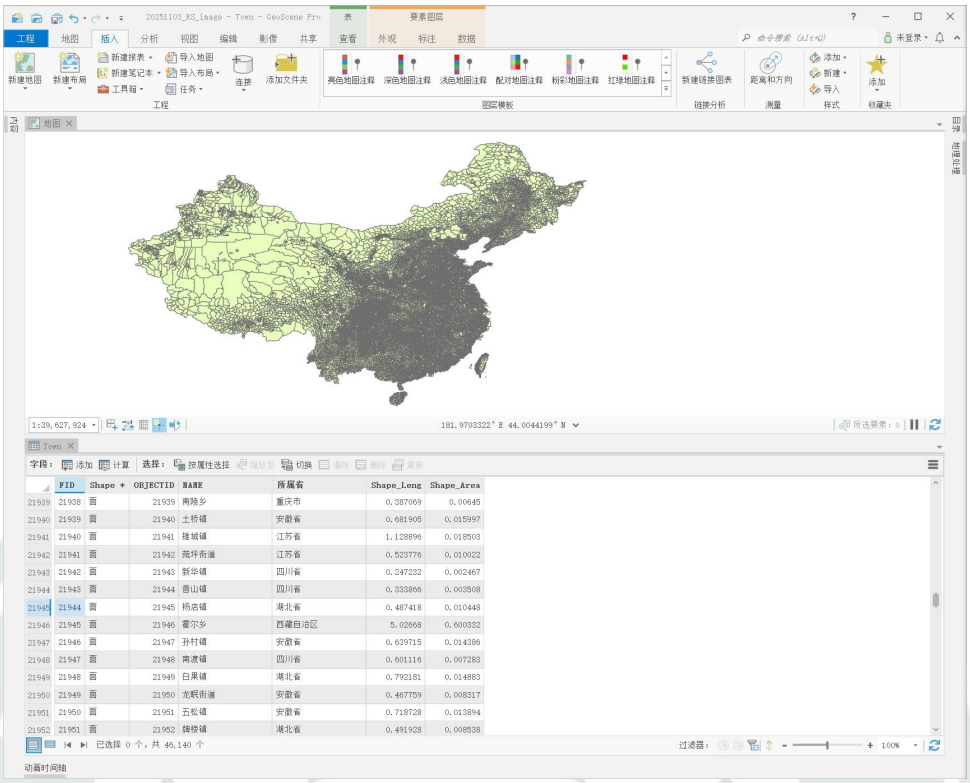


图 3.1.2-2 打开Town对象的属性表

可以发现，其中的字段项“NAME”表示该乡镇的名称，字段项“所属省”表示该乡镇所述省的名称。

使用“按属性选择”工具，确认是否可以正常筛选出孝感市孝南区杨店镇的数据：

图 3.1.2-3 使用“按属性选择”工具筛选出杨店镇的数据项

该查询对应的 SQL 语句为：

```
NAME LIKE '%杨店%' And 所属省 LIKE '%湖北%'
```

成功得到查询结果如下：

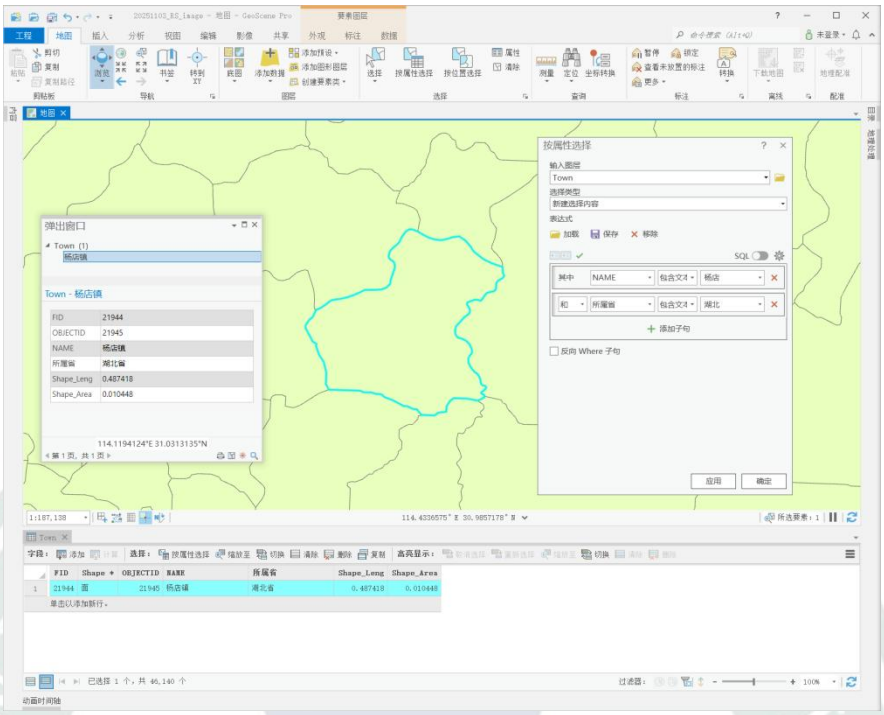


图 3.1.2-4 成功查询到杨店镇的矢量数据的字段项

可以在上方的“地图”-“底图”上选择矢量底图（如天地图的矢量注记，若地图坐标系不一致可先进行不同坐标系之间的投影变换）并加载，调整杨店镇对应矢量的透明度与图层叠置关系后可以更直观地检查按属性筛选结果是否合理：

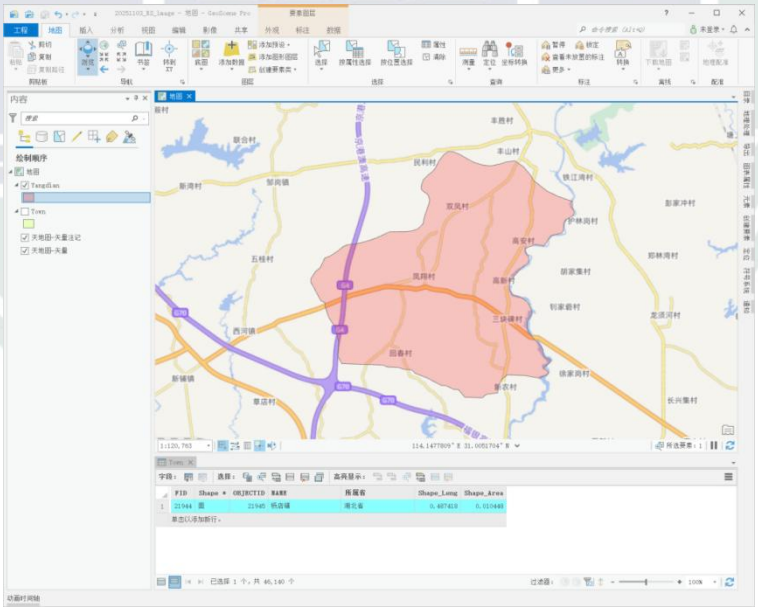


图 3.1.2-5 加载矢量底图以检验筛选结果是否合理

由图可见，筛选结果位于杨店镇地区，南北向的 G4 高速（京港澳高速，在杨店镇有收费站）、东西向的 G316 国道（即横穿杨店镇集中居民区的桃花驿大道）和王杨线（一般认为是杨店镇区的外环）等位置正确，东边的界河即孝感市和武汉市的分界线（杨店镇位于孝感市东边界上，孝感市孝南区杨店镇距离武汉市黄陂区界河仅数公里）。

打开左侧的“内容”，右键点击 Town 对象，在打开的菜单栏中点击“数据”-“导出要素”，使用相同的方法筛选出杨店镇的矢量数据导出为新的要素：

图 3.1.2-6 使用“导出要素”工具将杨店镇的矢量数据导出为新的要素

在右侧的“目录”中，打开数据库，点击项目数据库，可以找到刚刚导出的“Yangdian”要素：

图 3.1.2-7 新导出的“Yangdian”要素

右键点击该要素，选择“导出”-“要素类至 Shapefile...”：

图 3.1.2-8 将得到的要素导出为 Shapefile

即可得到研究区域杨店镇的矢量边界数据：

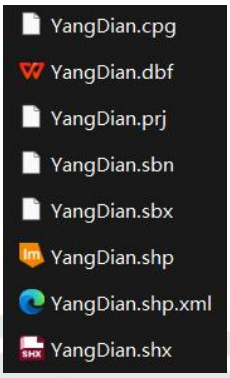


图 3.1.2-9 得到的矢量数据

3.1.3 在 GEE 平台获取遥感影像

打开 GEE 平台的代码编辑器（https://code.earthengine.google.com/），选择左侧的“Assets”，点击“NEW”-“Table Upload”-“Shape files”：

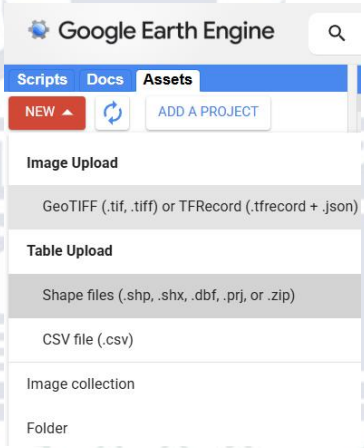


图 3.1.3-1 向GEE平台中导入所得矢量边界数据

在弹出的窗口中点击“SELECT”上传所得的矢量文件，设置名称为“Yangdian”：

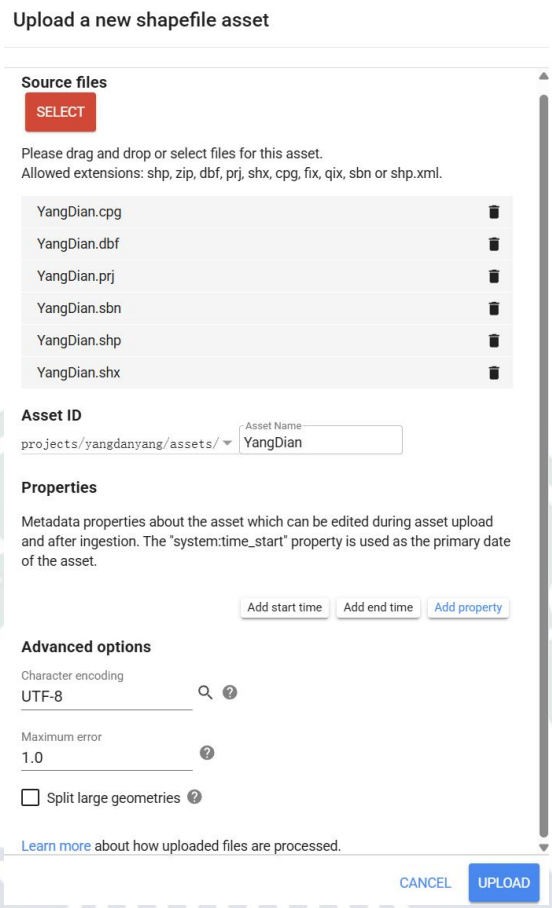


图 3.1.3-2 向GEE平台中导入所得矢量边界数据

等待上传完成后，在左侧的“Assets”区域即可看见上传后的数据：

图 3.1.3-3 在左侧的“Assets”区域即可看见上传的数据

先在 Scripts 打开脚本，然后鼠标停在 Assets 中该数据的右侧，会显示导入脚本按钮：

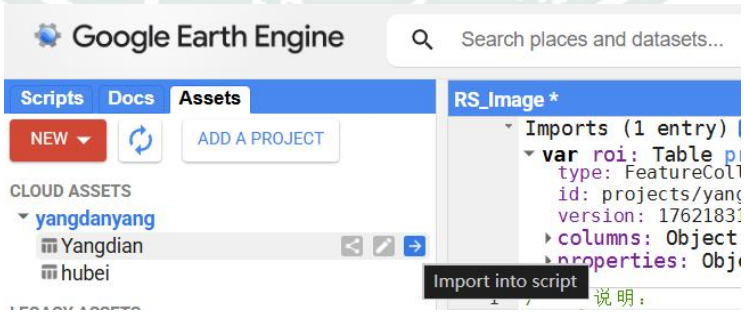


图 3.1.3-4 将Assets中的数据导入脚本

导入后，在脚本中使用下列代码：

```
/**** 说明:
 *  - 需预先定义 roi（ee.Geometry / ee.Feature / ee.FeatureCollection）
```

```
 *  - 针对 Landsat 8 C2 L2：缩放→ROI 裁剪→ROI 内云比例统计（roi_cloud_frac）
 *  - 依据阈值下载“与 ROI 相交（或可选：完整覆盖 ROI）且 ROI 云量低于阈值”的所有影像（保留全部波段，统一为 Float）
 *  - 合并统计流程、统一命名与导出，仅一次 evaluate 即完成预览与导出任务创建
 ****/
var roi = ee.FeatureCollection("projects/yangdanyang/assets/Yangdian");
// ====================== 可调参数 ======================
var START_YEAR = 2015;                  // 起始年份（含）
var CLOUD_FRAC_MAX = 0.001;               // ROI 内云比例上限阈值（0~1），如 0.01 表示 1%
var REQUIRE_FULL_ROI_COVERAGE = false;  // true：要求影像足迹完整包含 ROI；false：只要与 ROI 相交即可
var TODAY        = ee.Date(Date.now());
var FILTER_START = ee.Date.fromYMD(START_YEAR, 1, 1);
var FILTER_END   = TODAY.advance(1, 'day'); // 过滤结束时间（开区间）
// 预览参数
var VIS_RGB = {bands: ['SR_B4', 'SR_B3', 'SR_B2'], min: 0, max: 0.3};
// ====================== 影像集合（L8 C2 L2） ======================
var l8 = ee.ImageCollection('LANDSAT/LC08/C02/T1_L2')
  .filterBounds(roi)
  .filterDate(FILTER_START, FILTER_END);
// ====================== 工具函数 ======================
// 选择更稳健的场景 ID（优先 LANDSAT_SCENE_ID，不存在则用 LANDSAT_PRODUCT_ID）
function getSceneIdSafe(image) {
  return ee.String(ee.Algorithms.If(
    image.propertyNames().contains('LANDSAT_SCENE_ID'),
    image.get('LANDSAT_SCENE_ID'),
    image.get('LANDSAT_PRODUCT_ID')
  ));
}
// 缩放光学与热红外，裁剪 ROI，并写入时间/场景 ID
function withScaledClipAndMeta(image) {
  // USGS C2 L2 标准缩放
  var optical = image.select('SR_B.*').multiply(0.0000275).add(-0.2);
  var thermal = image.select('ST_B.*').multiply(0.00341802).add(149.0);
  var scaled = image.addBands(optical, null, true)
                    .addBands(thermal, null, true)
                    .clip(roi);
  return scaled
    .set('system:time_start', image.get('system:time_start'))
    .set('SCENE_ID_SAFE', getSceneIdSafe(image));
}
// 由 QA_PIXEL 生成云相关掩膜（膨胀云/卷云/云/云影）
function qaToCloudMask(qaImage) {
  var bit = function(n) { return qaImage.bitwiseAnd(1 << n).neq(0); };
```

```
  // Landsat C2 L2 QA_PIXEL 位：1=膨胀云, 2=卷云, 3=云, 4=云阴影
  return bit(1).or(bit(2)).or(bit(3)).or(bit(4)).rename('cloudmask');
}
// 在 ROI 内统计云比例与“是否完整覆盖 ROI”
function withRoiStats(image) {
  var qa        = image.select('QA_PIXEL');
  var cloudAny  = qaToCloudMask(qa);  // 已命名为 cloudmask
  var valid     = ee.Image(1)
                    .updateMask(image.select('SR_B2').mask())
                    .rename('valid');
  // 合并两波段后分别求和：键名仍为 cloudmask / valid
  var stats = ee.Image.cat([cloudAny, valid]).reduceRegion({
    reducer: ee.Reducer.sum(),
    geometry: roi,
    scale: 30,
    maxPixels: 1e13,
    bestEffort: true
  });
  var cloudPix = ee.Number(ee.Algorithms.If(stats.get('cloudmask'),
ee.Number(stats.get('cloudmask')), 0));
  var totalPix = ee.Number(ee.Algorithms.If(stats.get('valid'),
ee.Number(stats.get('valid')),     0));
  var cloudFrac = ee.Number(ee.Algorithms.If(totalPix.gt(0), cloudPix.divide(totalPix),
1));
  var contains = image.geometry(ee.ErrorMargin(1)).contains(roi, ee.ErrorMargin(1));
  return image
    .set('roi_cloud_pixels', cloudPix)
    .set('roi_total_pixels', totalPix)
    .set('roi_cloud_frac', cloudFrac)
    .set('contains_roi', contains);
}
// 组装导出命名与统一数据类型（Float）
function prepareForExport(image) {
  var dateStr = ee.Date(image.get('system:time_start')).format('YYYYMMdd');
  var sceneId = ee.String(image.get('SCENE_ID_SAFE'));
  var cfStr   = ee.Number(image.get('roi_cloud_frac')).format('%.3f');
  // var exportDesc = ee.String('L8_ROIcf').cat('_').cat(cfStr).cat('_').cat(dateStr);
  var exportDesc = ee.String('L8_ROIcf').cat('_').cat(dateStr);
  var filePrefix = exportDesc.cat('_').cat(sceneId);
  return ee.Image(image).toFloat().set({
    exportDescription: exportDesc,
    exportPrefix:      filePrefix
  });
}
```

```
// ==================== 预处理（缩放 + 统计 + 覆盖标记） ====================
var processed = l8
  .map(withScaledClipAndMeta)
  .map(withRoiStats);
// ==================== 按阈值筛选（云量 + 可选完整覆盖） ====================
var filtered = processed
  .filter(ee.Filter.gt('roi_total_pixels', 0))                    // ROI 内必须有有效像元
  .filter(ee.Filter.lte('roi_cloud_frac', CLOUD_FRAC_MAX));        // ROI 内云比例 ≤ 阈值
if (REQUIRE_FULL_ROI_COVERAGE) {
  filtered = filtered.filter(ee.Filter.eq('contains_roi', true));   // 影像足迹完整包含 ROI
}
// ==================== 构建导出集合、命名、统一类型 ====================
var exportCollection = filtered.map(prepareForExport);
// ==================== 预览 + 批量导出（单次 evaluate 完成） ====================
var exportList = exportCollection.toList(exportCollection.size());
exportList.evaluate(function(list) {
  if (!list || list.length === 0) {
    print('提示：当前条件下无可导出影像。请放宽云量阈值或检查 ROI/时间范围。');
    return;
  }
  // 预览首景
  Map.centerObject(roi, 8);
  Map.addLayer(ee.Image(exportList.get(0)), VIS_RGB, 'ROI 低云量影像（首景预览）');
  print('可导出影像。');
  // 逐景创建导出任务
  list.forEach(function(info, index) {
    var image = ee.Image(exportList.get(index));  // 保持服务端引用
    Export.image.toDrive({
      image: image,                                  // 导出全部波段（统一 Float）
      description: info.properties.exportDescription, // 任务名
      folder: 'GEE_L8_ROI_LowCloud',
      fileNamePrefix: info.properties.exportPrefix,   // 文件名：含云量与场景 ID
      scale: 30,
      region: roi,
      crs: 'EPSG:4326',
      maxPixels: 1e13,
      fileFormat: 'GeoTIFF'
    });
  });
  print('已创建导出任务。请在 Tasks 面板中启动。');
});
```

上述代码是用 Google Earth Engine (GEE) 下载研究区域的 Landsat 8 卫星影像，并筛选出云量较低的影像，将其导出到谷歌云端硬盘中。具体步骤为：基于杨店镇的矢量边界定义感兴趣区域（ROI），设定云量的上限阈值、时间过滤的开始和结束日期等参数，使用 QA_PIXEL 波段检测影像中的云并统计 ROI 区域内的云比例，将影像的光学波段和热红外波段的像素值进行缩放以将其转换成标准化值，剪裁影像到 ROI 区域并附加元数据（时间、场景 ID 等），根据云量阈值筛选影像（云量小于 CLOUD_FRAC_MAX），在下方的地图中预览符合条件的第一张影像，并为每个符合条件的影像创建批量导出任务。导出的影像包含所有波段，且数据类型统一为 float。

3.1.4 从谷歌云端硬盘（Google Drive）上下载实习所需的遥感影像

所得数据均在谷歌云端硬盘（Google Drive）中，其所在文件夹与 GEE 中对应的项目同名：

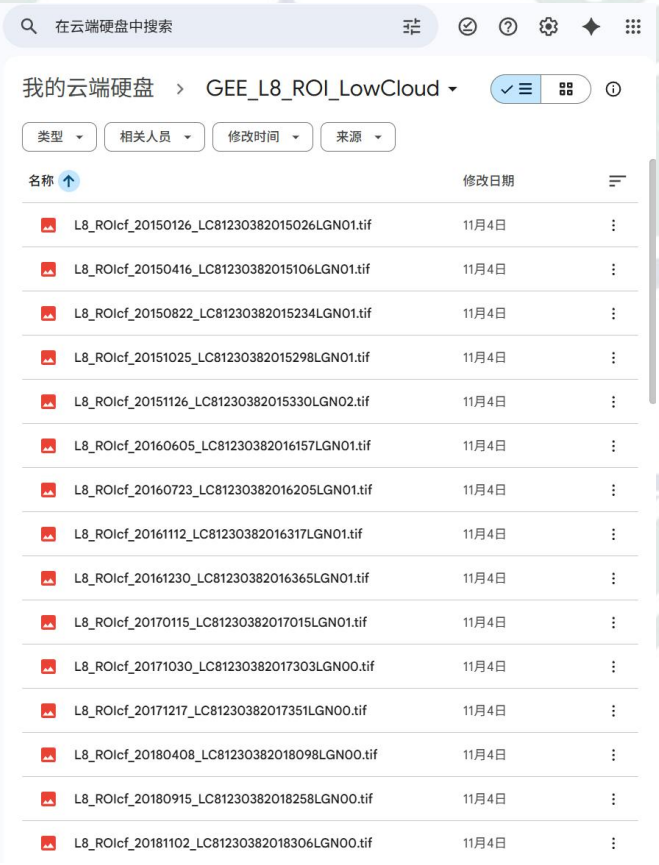


图 3.1.4-1 从GEE中导出的遥感影像数据导出到了谷歌云端硬盘上

将该文件夹中的内容下载即可。

3.2 影像的基本信息

下载的影像可以使用 R 中的 raster 包分析其性质，如数据的波段信息、空间参考、分辨率、

行列规模、以及各波段可叠加性等。除时序分析外，其余处理均使用 L8_ROIcf_20230727_LC81230382023208LGN00.tif 作为处理对象，因为这一天是夏天，且影像云量较少，适合进行较为理想的遥感影像分析。

3.2.1 影像的波段信息

影像的波段信息是使用影像时最基本、最常用的信息之一。

```
## ----  1.2 影像的性质 ----
## 读取影像栅格数据
img <- stack(tif_path)
names(img)
## 波段提取函数
pick_band <- function(x, key = "B2") {
    nm <- tolower(names(x))
    key <- tolower(key)
    i <- grep(paste0("(^|[_-])", key, "0?2?($|[_-])"), nm)
    if (length(i) == 0) i <- grep("b2", nm)
    if (length(i) == 0) stop("找不到波段：", key)
    x[[i[1]]]
}
b2 <- pick_band(img, "B2") # Blue
b3 <- pick_band(img, "B3") # Green
b4 <- pick_band(img, "B4") # Red
b5 <- pick_band(img, "B5") # NIR
```

其中输出的波段信息如下：

```
> names(img)
 [1] "SR_B1"         "SR_B2"         "SR_B3"         "SR_B4"         "SR_B5"
"SR_B6"         "SR_B7"         "SR_QA_AEROSOL"
 [9] "ST_B10"        "ST_ATRAN"      "ST_CDIST"      "ST_DRAD"       "ST_EMIS"
"ST_EMSD"       "ST_QA"         "ST_TRAD"
 [17] "ST_URAD"       "QA_PIXEL"      "QA_RADSAT"
```

上述波段中，SR_*表示地表反射率（Surface Reflectance），主要针对 OLI 可见/近红外/短波红外波段；ST_*表示地表温度（Surface Temperature）相关层，主要基于 TIRS 热红外数据推导；QA_*表示质量保证/像元标记层，用于云、积雪、阴影、辐射饱和等检测与屏蔽。

波段代码	波段内容	波段范围	常见用途
SR_B1	海岸/气溶胶带（Coastal/Aerosol）	约 0.43 - 0.45 μm	大气校正、沿海/内陆水体、薄雾识别
SR_B2	蓝光（Blue）	约 0.45 - 0.51 μm	水体制图、气溶胶敏感、可见光合成
SR_B3	绿光（Green）	约 0.53 - 0.59 μm	植被活力、叶绿素响应、真彩/伪彩合成
SR_B4	红光（Red）	约 0.64 - 0.67 μm	植被指数（如 NDVI 的红波段）、土壤/岩石分辨
SR_B5	近红外 NIR	约 0.85 - 0.88 μm	植被冠层结构、含水状况、NDVI/NDBI 等

波段代码	波段内容	波段范围	常见用途
SR_B6	短波红外 SWIR1	约 1.57 – 1.65 μm	指数 估算土壤/植被含水量、烧毁面积、地质分辨、计算NBR/NDMI 等
SR_B7	短波红外 SWIR2	约 2.11 – 2.29 μm	矿物/岩性判别、烧毁严重度、积雪/冰特征补充
SR_QA_AEROSOL	气溶胶质量层	—	像元级气溶胶质量/类别标记（低/中/高等），帮助评估 SR 可靠性与屏蔽受影响像元
QA_PIXEL	像元级质量标记	—	以位标志编码云、云阴影、雪/冰、水体、稀薄云等，可用于生成云/阴影/雪掩膜，提升时序分析与分类准确性
QA_RADSAT	辐射饱和标记（Radiometric Saturation）	—	指出了哪些波段在该像元发生饱和，以提示高反射或高亮度区域（如云顶、雪、城市亮屋顶）应剔除或谨慎解读

表 3.2.1-1 Landsat-8的部分常用波段信息

在计算各类遥感指数（如 NDVI/NDMI/NBR 等）时，应优先用 SR_* 波段的信息来计算，并结合 SR_QA_AEROSOL 和 QA_PIXEL 屏蔽云/阴影，以提升结果稳健性。一般的影像使用流程未：先用 QA_PIXEL 与 QA_RADSAT 掩膜，然后结合 SR_QA_AEROSOL/Cloud Distance 做增强筛选，在清洁像元上使用 SR_* 计算指数或使用 ST_* 反演温度。

3.2.2 影像的其他基本信息

```
## ---- 1.3 影像信息和统计指标 ----
## 坐标参考系（CRS）
crs(b2)
## 像元/行列/维度等基础信息
ncell(b2)              # 像元总数
nrow(b2); ncol(b2)     # 行列数
dim(b2)                # 若为 RasterStack/Brick，会返回 (nrow, ncol, nlayers)
## 空间分辨率
res(b2)
## 通道（波段）数
nlayers(b2)
## 比较两个栅格是否具有相同的空间属性（范围、分辨率、投影等）
compareRaster(b2, b3)
## 除了第 8 波段（全色，分辨率 15 m）以外，其余波段分辨率为 30 m，可自由叠加。
## 例如：叠加 5-4-3 波段（近红外-红-绿）
s <- stack(b5, b4, b3)
```

输出结果如下：

```
> ## ---- 1.3 影像信息和统计指标 ----
> ## 坐标参考系（CRS）
> crs(b2)
Coordinate Reference System:
```

```
Deprecated Proj.4 representation: +proj=longlat +datum=WGS84 +no_defs
WKT2 2019 representation:
GEOGCRS["unknown",
    DATUM["World Geodetic System 1984",
        ELLIPSOID["WGS 84",6378137,298.257223563,
            LENGTHUNIT["metre",1]],
        ID["EPSG",6326]],
    PRIMEM["Greenwich",0,
        ANGLEUNIT["degree",0.0174532925199433],
        ID["EPSG",8901]],
    CS[ellipsoidal,2],
        AXIS["longitude",east,
            ORDER[1],
            ANGLEUNIT["degree",0.0174532925199433,
                ID["EPSG",9122]]],
        AXIS["latitude",north,
            ORDER[2],
            ANGLEUNIT["degree",0.0174532925199433,
                ID["EPSG",9122]]]]
> ## 像元/行列/维度等基础信息
> ncell(b2)              # 像元总数
[1] 247234
> nrow(b2); ncol(b2)     # 行列数
[1] 481
[1] 514
> dim(b2)                # 若为 RasterStack/Brick，会返回 (nrow, ncol, nlayers)
[1] 481 514   1
> ## 空间分辨率
> res(b2)
[1] 0.0002694946 0.0002694946
> ## 通道（波段）数
> nlayers(b2)
[1] 1
> ## 比较两个栅格是否具有相同的空间属性（范围、分辨率、投影等）
> compareRaster(b2, b3)
[1] TRUE
```

上述影像的坐标系信息可以根据需要进行投影转换，对影像栅格的信息的了解有助于后续代码的编写。

3.3 从影像中提取单通道和多通道图像

3.3.1 从影像中提取单通道和多通道图像

单通道显示用于观察单一波段的灰度纹理与亮度差异，有助于辨识云、阴影、水体、亮屋顶等高/低反射特征[10]；多通道合成通过选择不同波段映射至 RGB 通道，形成真彩（B4/B3/B2）与常用假彩（B5/B4/B3）[11]。真彩利于直观解译地物色彩，假彩能突出植被（NIR 高反射）与水体（NIR 强吸收）。此外，常用线性拉伸（分位裁剪）操作来压制极端值影响，增强可视对比度。

```
## ----  1.4 单通道和多通道图像 ----
## 单通道灰度图（B2/B3/B4/B5）
## 使用灰度色带，并在一个 2x2 画布中展示四个波段
save_seq_plot({
    op <- par(mfrow = c(2,2), mar = c(3,3,2,1))
    on.exit(par(op), add = TRUE)
    plot(b2, main = "Blue (B2)",  col = gray(0:100/100))
    plot(b3, main = "Green (B3)", col = gray(0:100/100))
    plot(b4, main = "Red (B4)",   col = gray(0:100/100))
    plot(b5, main = "NIR (B5)",   col = gray(0:100/100))
}, "单通道_灰度")
```

上述代码对 B2/B3/B4/B5 分别进行灰度绘制，并采用 2%–98% 分位裁剪的线性拉伸，组装真彩（依次为 B4,B3,B2 波段）与假彩（依次为 B5,B4,B3 波段）RGB 并绘图。若检测到原始 DN 为 16-bit，将按 USGS C2 L2 缩放系数将 DN 转换为地表反射率后再绘制一版真/假彩色影像。

输出的结果图如下：

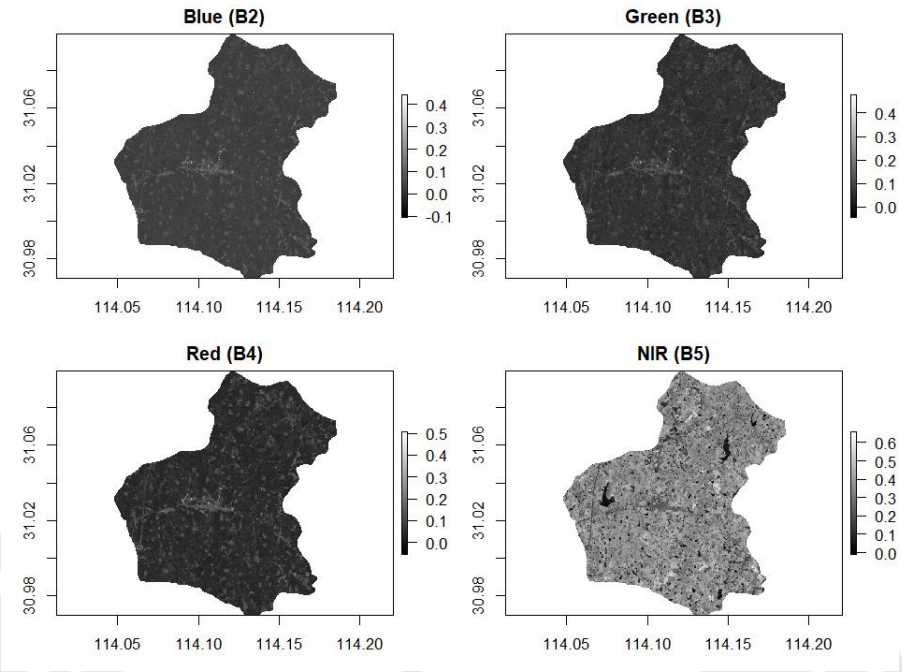


图 3.3.1-1 单通道灰度图像

由图可见，单通道灰度图提取后较为清晰，四个单通道的图的黑白色彩有显著不同，这是因为不同波段对不同地物的反射性质不一样，如健康植被的近红外（NIR）波段反射率高、水体的近红外（NIR）波段反射率低，这些特点从影像上可以得到验证，从而验证了波段提取的正确性。

为了增强对比度，可以进行灰度的拉伸：

```
## 简单线性拉伸以增强对比（2%-98% 分位裁剪）
lin_stretch <- function(x, probs = c(0.02, 0.98)){
    q <- quantile(values(x), probs = probs, na.rm = TRUE)
    xx <- clamp(x, lower = q[1], upper = q[2], useValues = TRUE)
    (xx - q[1]) / (q[2] - q[1])
}
b2s <- lin_stretch(b2); b3s <- lin_stretch(b3)
b4s <- lin_stretch(b4); b5s <- lin_stretch(b5)
save_seq_plot({
    op <- par(mfrow = c(2,2), mar = c(3,3,2,1))
    on.exit(par(op), add = TRUE)
    plot(b2s, main = "Blue (拉伸)",  col = gray(0:100/100))
    plot(b3s, main = "Green (拉伸)", col = gray(0:100/100))
    plot(b4s, main = "Red (拉伸)",   col = gray(0:100/100))
    plot(b5s, main = "NIR (拉伸)",   col = gray(0:100/100))
}, "单通道_拉伸")
```

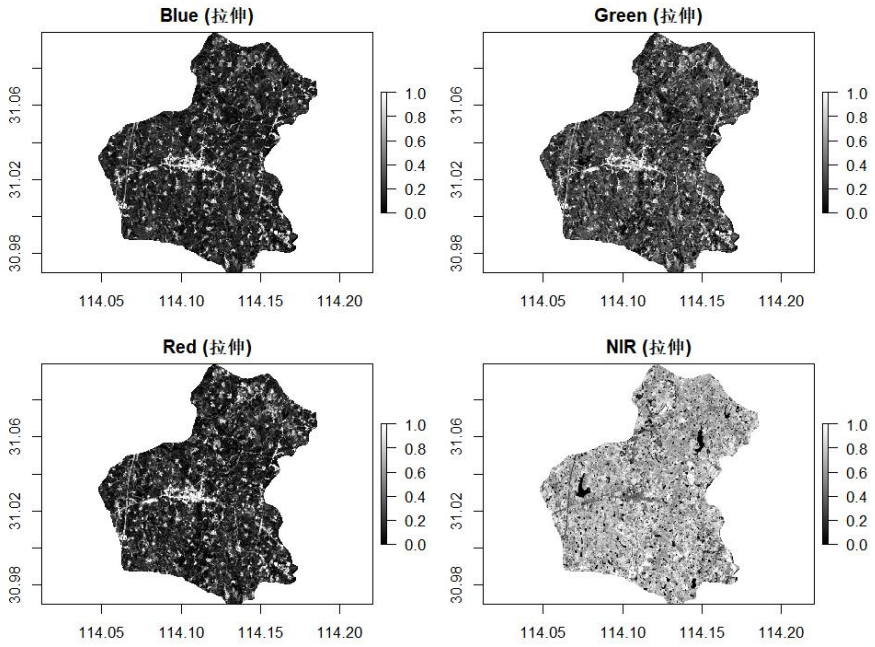


图 3.3.1-2 线性拉伸后的单通道灰度图像

灰度拉伸后，城镇建成区、道路、水体与耕地的灰度差异更加清晰。

然而，单通道灰度图所含信息单一，可以通过多波段组合的方式形成彩色图片。为构成真彩色图片。真彩色映射规则为 Red=SR_B4，Green=SR_B3，Blue=SR_B2；假彩色映射规则为 NIR=SR_B5，Red=SR_B4，Green=SR_B3：

```
## 真彩与假彩合成
s_true  <- stack(b4, b3, b2) # R, G, B
s_false <- stack(b5, b4, b3) # NIR, R, G
## 自动判定 scale
maxDN <- max(c(maxValue(b2), maxValue(b3), maxValue(b4), maxValue(b5)), na.rm = TRUE)
## Landsat-8 原始数据是 16-bit
plot_scale <- if (is.finite(maxDN) && maxDN > 1) maxDN else 1
## 反射率转换
to_reflectance <- function(x) {
    calc(x, fun = function(v) 0.0000275 * v - 0.2)
}
if (plot_scale > 1) {
    # 堆叠顺序为 B, G, R, NIR (索引 1, 2, 3, 4)
    img_ref <- to_reflectance(stack(b2, b3, b4, b5))
}
## 绘制真彩 / 假彩并保存
save_seq_plot({
    plotRGB(s_true,  scale = plot_scale, stretch = "hist",
            main = "真彩色 (B4,B3,B2)")
}, "真彩色")
save_seq_plot({
    plotRGB(s_false, scale = plot_scale, stretch = "hist",
```

```
            main = "假彩色 (B5,B4,B3)")
}, "假彩色")
if (exists("img_ref")) {
    # 由于 img_ref 尚未重命名，我们使用索引访问
    # 索引：1=Blue, 2=Green, 3=Red, 4=NIR
    save_seq_plot({
        plotRGB(stack(img_ref[[3]],  img_ref[[2]], img_ref[[1]]), # 真彩 R,G,B
                scale = 1, stretch = "hist", main = "真彩色(反射率)")
    }, "真彩色_反射率")
    save_seq_plot({
        plotRGB(stack(img_ref[[4]], img_ref[[3]], img_ref[[2]]), # 假彩 NIR,R,G
                scale = 1, stretch = "hist", main = "假彩色(反射率)")
    }, "假彩色_反射率")
}
```

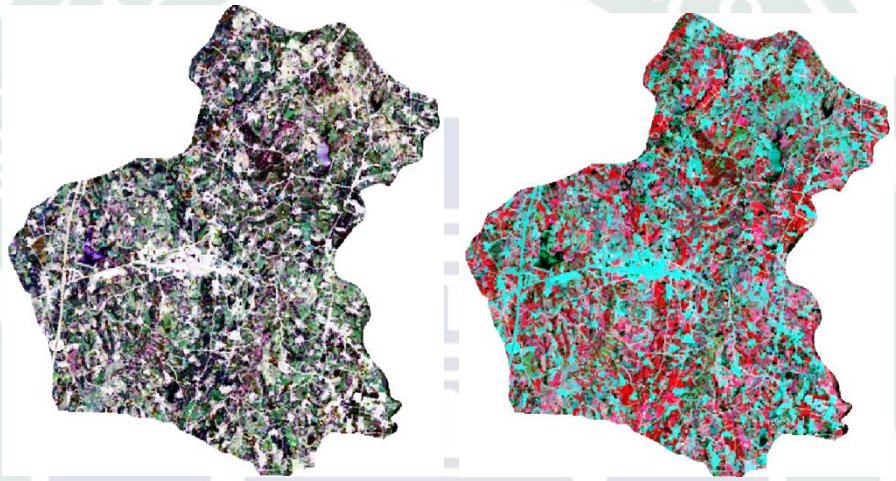


图 3.3.1-3 真彩色（左）与假彩色（右）合成图

在真彩色合成图中，所呈现的影像与人眼感受基本一致，建筑区、道路、植被等显示清晰；假彩合成中植被呈亮红至暗红，水体偏暗（近红外吸收），居民区在红光/近红外呈中高亮度且纹理规则。可以提取 NDVI 与假彩色图像对比验证：

```
## NDVI 计算并输出
## 使用索引 (4=NIR, 3=Red) 或原始变量
nir <- if (exists("img_ref")) img_ref[[4]] else b5
red <- if (exists("img_ref")) img_ref[[3]] else b4
## NDVI = (NIR - Red) / (NIR + Red)
ndvi <- (nir - red) / (nir + red)
## 输出 NDVI 栅格 (使用 next_name 自动编号)
ndvi_tif <- next_name("NDVI", "tif")
writeRaster(ndvi, ndvi_tif, overwrite = TRUE)
message("已输出：", ndvi_tif)
## 绘图保存 NDVI
save_seq_plot({
    plot(ndvi, col = rev(terrain.colors(10)), main = "NDVI (Landsat-8 L2 C2)")
```

```
}, "NDVI")
message("全部结果已保存在：", normalizePath(out_dir))
```

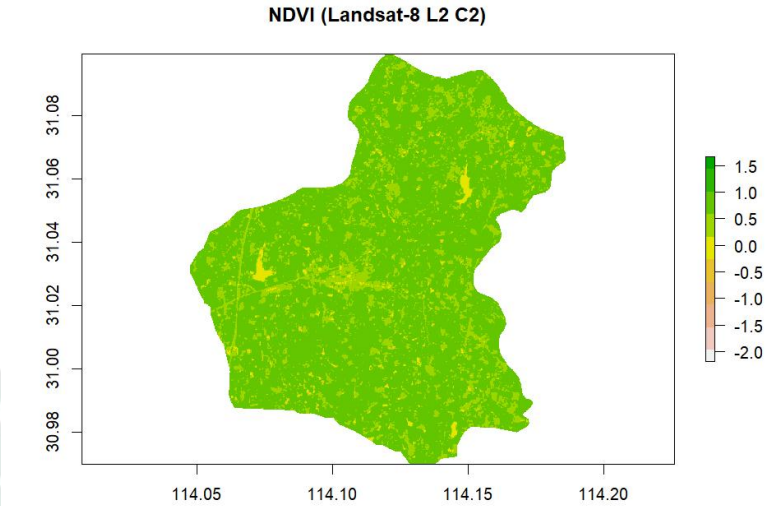


图 3.3.1-4 提取的NDVI指数的图像

对比图 3.3.1-3（右）与图 3.3.1-4 可见：在假彩色图（B5,B4,B3 波段）上的“亮红色”区域基本对应 NDVI 较高的区域，进一步验证了这些区域为植被这种地物类型。

3.3.2 遥感影像可视化渲染工具

为了更加便捷地进行遥感影像的可视化，我基于 R 的 terra 包等编写了对 Landsat 等多波段遥感影像做真彩色渲染的代码。

单图可视化脚本可以选一张图进行可视化：

```
setwd("E:/code/r_code/20251031_Remote_Sensing_Image_Processing_Using_R") # 设置工作路径，便于使用相对路径（带有盘符的是绝对路径）
library(terra)
# 1) 指定你的影像路径（把文件名改成你那一张）
#       例：Windows 用反斜杠或双反斜杠；Mac/Linux 用斜杠
#       假设你的文件夹叫 "GEE_L8_Yangdian_images"
image_path <-
"GEE_L8_Yangdian_images_cloudfree/L8_Yangdian_20250902_LateSeason_LC81230392025245LGN00_masked_cloudfree.tif"
# 2) 读取影像
x <- rast(image_path)
# 3) 选择真彩色波段：红=SR_B4，绿=SR_B3，蓝=SR_B2
bn <- names(x)
if (all(c("SR_B4","SR_B3","SR_B2") %in% bn)) {
      rgb <- x[[c("SR_B4","SR_B3","SR_B2")]]
} else if (all(c("B4","B3","B2") %in% bn)) {
```

```
        #  兼容部分导出或重命名场景
        rgb  <-  x[[c("B4","B3","B2")]]
} else  {
        stop("未找到用于真彩色的  3  个波段：请检查文件是否包含  SR_B4  /  SR_B3  /  SR_B2。
")
}
#  4)  可视化前处理
#        你的  GEE  里已按  0.0000275  和  -0.2  做了缩放，可能出现<0 或>1 的值，显示前裁剪
到  [0,1]
rgb  <-  clamp(rgb,  lower=0,  upper=1)
#  5)  绘图—线性拉伸  +  轻微  gamma（让观感更自然）
#        q=c(0.02,  0.98)  表示按  2%~98%  分位做拉伸；可以根据观感改成  c(0.01,0.99)  或
c(0.03,0.97)
plotRGB(
        rgb,
        stretch  =  "lin",
        q  =  c(0.02,  0.98),
        gamma  =  1.1,
        axes  =  TRUE,
        main  =  "Landsat  8  真彩色  (SR_B4-B3-B2)"
)
#  6)  如需保存到图片（可选）
#  png("L8_truecolor.png",  width=1600,  height=1200)
#  plotRGB(rgb,  stretch="lin",  q=c(0.02,0.98),  gamma=1.1,  axes=TRUE,
#                main="Landsat  8  真彩色  (SR_B4-B3-B2)")
#  dev.off()
#  —额外：如想看近红外假彩色（可选），解注释以下几行—
#  if  (all(c("SR_B5","SR_B4","SR_B3")  %in%  bn))  {
#      nir_rgb  <-  x[[c("SR_B5","SR_B4","SR_B3")]]  |>  clamp(0,1)
#      plotRGB(nir_rgb,  stretch="lin",  q=c(0.02,0.98),  gamma=1.1,  axes=TRUE,
#                    main="Landsat  8  假彩色  (NIR-Red-Green  =  SR_B5-B4-B3)")
#  }
```

批处理脚可视化本可以遍历整个文件夹批量出 PNG：

```
#  批处理整文件夹：将遥感影像渲染为真彩色  PNG
#  依赖：terra
#  使用方法：
#  1)  修改【参数区】中的路径和可视化参数；
#  2)  直接运行本脚本。
#  ====================  基础设置  ====================
setwd("E:/code/r_code/20251031_Remote_Sensing_Image_Processing_Using_R")    #  工作路
径
suppressPackageStartupMessages({
        library(terra)
})
```

```
# ====================  参数区（按需修改）  ====================
in_dir          <-  "GEE_L8_Yangdian_image"    #  输入影像文件夹（相对或绝对路径均可）
out_dir         <-  "GEE_L8_Yangdian_images_vis"                                    #
输出图片文件夹
pattern         <-  "\\.(tif|tiff|img)$"                                  #  处理的文件扩展
名
recursive     <-  TRUE                                                    #  是
否递归遍历子文件夹
#  可视化参数（线性拉伸与  gamma）
q_low            <-  0.02
#  下分位（建议  0.01~0.03  之间调）
q_high          <-  0.98                                                    #
上分位
gamma_v         <-  1.1                                                     #
gamma（>1  稍亮中低调；=1  关闭）
add_axes       <-  FALSE                                                  #
是否绘制坐标轴
bg_col          <-  "white"                                                #
画布背景色
#  输出图像尺寸与质量
base_width   <-  1800                                                   #  基
准宽度（像素）
max_width     <-  2400                                                  #
最大宽度（像素，上限避免超大图）
dpi               <-  150
#  分辨率（像素/英寸）
overwrite     <-  TRUE                                                    #
已存在是否覆盖
#  可选：同时导出近红外假彩色（NIR-Red-Green  =  SR_B5-SR_B4-SR_B3）
export_falsecolor  <-  FALSE                                            #  需要就
设为  TRUE
#  ====================  工具函数  ====================
#  根据常见命名自动抓取  RGB  三个波段
get_rgb  <-  function(x)  {
       bn  <-  names(x)
       bn_lower  <-  tolower(bn)
       #  1)  Landsat  SR  命名
       if  (all(c("sr_b4","sr_b3","sr_b2")  %in%  bn_lower))  {
             return(x[[bn[match(c("sr_b4","sr_b3","sr_b2"),  bn_lower)]]])
       }
       #  2)  简写  B4/B3/B2
       if  (all(c("b4","b3","b2")  %in%  bn_lower))  {
             return(x[[bn[match(c("b4","b3","b2"),  bn_lower)]]])
       }
```

```
        #  3)  red/green/blue
        if  (all(c("red","green","blue")  %in%  bn_lower))  {
                return(x[[bn[match(c("red","green","blue"),  bn_lower)]]])
        }
        stop("未找到真彩色所需三波段（SR_B4/B3/B2  或  B4/B3/B2  或  red/green/blue）")
}
#  假彩色（NIR-Red-Green）三波段
get_nir_rgb  <-  function(x)  {
        bn  <-  names(x);  bn_lower  <-  tolower(bn)
        if  (all(c("sr_b5","sr_b4","sr_b3")  %in%  bn_lower))  {
                return(x[[bn[match(c("sr_b5","sr_b4","sr_b3"),  bn_lower)]]])
        }
        if  (all(c("b5","b4","b3")  %in%  bn_lower))  {
                return(x[[bn[match(c("b5","b4","b3"),  bn_lower)]]])
        }
        return(NULL)    #  没有就返回空
}
#  按图幅比例计算输出尺寸
calc_size  <-  function(r,  base_w=1800,  max_w=2400)  {
        nc  <-  ncol(r);  nr  <-  nrow(r)
        ar  <-  nr  /  nc
        width_px    <-  min(max(base_w,  800),  max_w)
        height_px  <-  max(  round(width_px  *  ar),  600  )
        list(w=width_px,  h=height_px)
}
#  ====================  目录与文件收集  ====================
if  (!dir.exists(out_dir))  dir.create(out_dir,  recursive  =  TRUE)
files  <-  list.files(in_dir,  pattern  =  pattern,  full.names  =  TRUE,  recursive
=  recursive)
if  (length(files)  ==  0)  stop("在输入文件夹中未找到影像（检查路径与扩展名  pattern）。
")
#  可选：控制临时与内存占用
terraOptions(memfrac  =  0.8)
#  ====================  处理与导出  ====================
log  <-  data.frame(
        file        =  character(),
        out_png  =  character(),
        ncol        =  integer(),
        nrow        =  integer(),
        success  =  logical(),
        message  =  character(),
        stringsAsFactors  =  FALSE
)
if  (export_falsecolor)  {
```

```
        log_false <- log[0,]
}
for (f in files) {
        cat("\n>>> 处理：", f, "\n")
        base <- tools::file_path_sans_ext(basename(f))
        # -------- 真彩色 --------
        out_png <- file.path(out_dir, paste0(base, "_RGB.png"))
        if (!overwrite && file.exists(out_png)) {
                cat("跳过（已存在）：", out_png, "\n")
                log <- rbind(log, data.frame(file=f, out_png=out_png, ncol=NA,
nrow=NA, success=TRUE, message="skip exists"))
        } else {
                tryCatch({
                        r     <- rast(f)
                        rgb <- get_rgb(r)
                        # 若 GEE 已做 0.0000275/-0.2 缩放，这里只需裁剪到 [0,1]
                        rgb <- clamp(rgb, lower=0, upper=1, values=TRUE)
                        sz    <- calc_size(rgb, base_w = base_width, max_w =
max_width)
                        png(filename = out_png, width = sz$w, height = sz$h,
res = dpi, bg = bg_col)
                        op <- par(mar=c(0,0,2,0))
                        plotRGB(
                                rgb,
                                stretch = "lin",
                                q = c(q_low, q_high),
                                gamma = gamma_v,
                                axes = add_axes,
                                main = paste0(base, " – 真彩色 (SR_B4-B3-B2)")
                        )
                        par(op); dev.off()
                        cat("完成：", out_png, "\n")
                        log <- rbind(log, data.frame(file=f, out_png=out_png,
ncol=ncol(rgb), nrow=nrow(rgb), success=TRUE, message="ok"))
                }, error = function(e) {
                        cat("失败：", conditionMessage(e), "\n")
                        if (file.exists(out_png)) try(unlink(out_png), silent=TRUE)
                        log <- rbind(log, data.frame(file=f, out_png=out_png,
ncol=NA, nrow=NA, success=FALSE, message=conditionMessage(e)))
                })
        }
        # -------- 假彩色（可选）--------
        if (export_falsecolor) {
```

```
            out_png_fc <- file.path(out_dir, paste0(base,
"_FalseColor_NIR-R-G.png"))
            if (!overwrite && file.exists(out_png_fc)) {
                cat("假彩色跳过（已存在）：", out_png_fc, "\n")
                log_false <- rbind(log_false, data.frame(file=f,
out_png=out_png_fc, ncol=NA, nrow=NA, success=TRUE, message="skip exists"))
            } else {
                tryCatch({
                    r <- if (exists("r")) r else rast(f)
                    nir <- get_nir_rgb(r)
                    if (!is.null(nir)) {
                        nir <- clamp(nir, lower=0, upper=1,
values=TRUE)
                        sz <- calc_size(nir, base_w = base_width,
max_w = max_width)
                        png(filename = out_png_fc, width = sz$w,
height = sz$h, res = dpi, bg = bg_col)
                        op <- par(mar=c(0,0,2,0))
                        plotRGB(
                            nir,
                            stretch = "lin",
                            q = c(q_low, q_high),
                            gamma = gamma_v,
                            axes = add_axes,
                            main = paste0(base, " – 假彩色
(NIR-Red-Green = SR_B5-B4-B3)")
                        )
                        par(op); dev.off()
                        cat("假彩色完成：", out_png_fc, "\n")
                        log_false <- rbind(log_false,
data.frame(file=f, out_png=out_png_fc, ncol=ncol(nir), nrow=nrow(nir),
success=TRUE, message="ok"))
                    } else {
                        cat("未找到生成假彩色所需波段（SR_B5/B4/B3 或
B5/B4/B3），跳过。\n")
                        log_false <- rbind(log_false,
data.frame(file=f, out_png=out_png_fc, ncol=NA, nrow=NA, success=FALSE,
message="no NIR set"))
                    }
                }, error = function(e) {
                    cat("假彩色失败：", conditionMessage(e), "\n")
                    if (file.exists(out_png_fc)) try(unlink(out_png_fc),
silent=TRUE)
```

```
                                log_false <- rbind(log_false, data.frame(file=f,
out_png=out_png_fc, ncol=NA, nrow=NA, success=FALSE,
message=conditionMessage(e)))
                        })
                }
        }
}
# ==================== 输出日志 ====================
write.csv(log, file.path(out_dir, "rgb_export_log.csv"), row.names = FALSE)
if (export_falsecolor) {
        write.csv(log_false, file.path(out_dir, "falsecolor_export_log.csv"),
row.names = FALSE)
}
cat("\n—— 全部完成 ——\n",
        "真彩色 成功：", sum(log$success), "；失败：", sum(!log$success),
        "\n 输出目录：", normalizePath(out_dir), "\n")
if (export_falsecolor) {
        cat("假彩色 成功：", sum(log_false$success), "；失败：",
sum(!log_false$success), "\n")
}
```

3.3.3 通道提取和重命名

对波段的提取可按名称或索引抽取目标波段，便于模块化调用。标准化命名（如“Blue/Green/Red/NIR”）能降低后续合成、指数计算与脚本复用的出错率。

```
## ---- 1.5 通道提取和重命名 ----
if (exists("img_ref")) {
    message("1.5 节：使用在 1.4 节中创建的 'img_ref'。")
    # 如果 img_ref 存在，我们就用它
    img_demo_stack <- img_ref
} else {
    message("1.5 节：'img_ref' 未创建。将使用 b2, b3, b4, b5 手动创建演示堆栈。")
    # 如果 img_ref 不存在，就用 1.2 节的波段手动堆叠一个
    # 堆叠顺序与 1.4 节中 'to_reflectance' 的输入顺序一致
    img_demo_stack <- stack(b2, b3, b4, b5)
}
message("正在对演示堆栈 'img_demo_stack' 执行重命名...")
## 对 'img_demo_stack' 操作
names(img_demo_stack) <- c("Blue", "Green", "Red", "NIR")
## 重命名 (Renaming)
message("原始 'img' 堆栈的波段名称：")
print(names(img)) # 打印原始名称
message("已重命名的 'img_demo_stack' 堆栈名称：")
```

```
print(names(img_demo_stack)) # 打印重命名后的名称
## 提取 (Subsetting)
message("从 'img_demo_stack' 中使用 [[...]] 提取 'Red' 波段 (演示): ")
# 现在 'img_demo_stack' 已被重命名，可以按名称提取
red_band_demo <- img_demo_stack[['Red']]
message("从 'img_demo_stack' 中使用 subset() 提取 'Green' 和 'Blue' (演示): ")
gb_bands_demo <- subset(img_demo_stack, c("Green", "Blue"))
# 检查层数
message("提取 'Red' 后的层数: ", nlayers(red_band_demo))
message("提取 'Green' 和 'Blue' 后的层数: ", nlayers(gb_bands_demo))
save_seq_plot({
    plot(red_band_demo, main = "1.5 提取演示: Red 波段", col = gray(0:100/100))
}, "提取_Red")
```

重命名结果如下：

```
> print(names(img)) # 打印原始名称
 [1] "SR_B1"         "SR_B2"         "SR_B3"         "SR_B4"         "SR_B5"         "SR_B6"
"SR_B7"         "SR_QA_AEROSOL"
 [9] "ST_B10"        "ST_ATRAN"      "ST_CDIST"      "ST_DRAD"       "ST_EMIS"
"ST_EMSD"       "ST_QA"         "ST_TRAD"
[17] "ST_URAD"       "QA_PIXEL"      "QA_RADSAT"
> message("已重命名的 'img_demo_stack' 堆栈名称: ")
> print(names(img_demo_stack)) # 打印重命名后的名称
[1] "Blue"  "Green" "Red"   "NIR"
```

重命名后，plotRGB、指数计算、裁剪叠加等调用更直观，降低了误配风险。

波段提取结果如下：

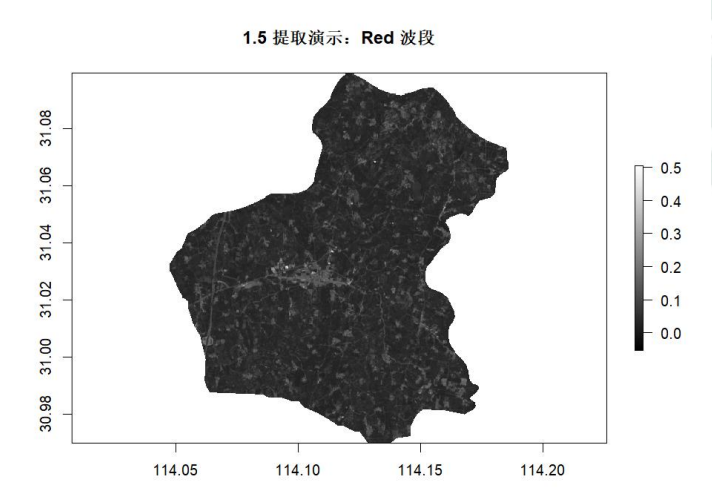


图 3.3.3-1 红波段提取结果图

如图所示，红波段提取结果与图 3.3.1-1 所示的红波段提取结果图一致，验证了波段提取结果的正确性。

3.4 空间裁剪或取子集

在实际研究过程中，研究的区域往往不一定与下载的影像的范围相同，因此通常需要进行图像空间的裁剪或者拼接，即在整景影像基础上完成规则矩形裁剪与矢量掩膜，为后续指标与分类提供聚焦区域等。

3.4.1 空间裁剪

在代码中，选用已重命名的多波段栈 img_demo_stack 作为裁剪基准，调用 crop() 完成矩形规则裁剪：

```
## ---- 1.6 空间裁剪或取子集 ----
## 本节演示两种常见操作：
## 1) 用 Extent 进行规则矩形裁剪 crop()
## 2) 用矢量多边形进行掩膜 mask()
message("1.6 节：开始空间裁剪与掩膜演示...")
## 选定用于裁剪的多波段栈（优先使用已重命名的 img_demo_stack；作为兜底用 s 或
stack(b5,b4,b3)）
if (exists("img_demo_stack")) {
    rs_base <- img_demo_stack
    message("使用 img_demo_stack 作为裁剪基础数据（层名：", paste(names(rs_base), collapse
= ","), "）")
} else if (exists("s")) {
    rs_base <- s
    message("未发现 img_demo_stack，使用 s <- stack(b5,b4,b3) 作为基础数据。")
} else {
    rs_base <- stack(b5, b4, b3)
    names(rs_base) <- c("NIR","Red","Green")
    message("未发现 s，临时创建 stack(b5,b4,b3) 作为基础数据。")
}
## 1) Extent 规则裁剪
message("创建 Extent 进行规则矩形裁剪 ...")
## 如果你知道目标范围，可在此替换为自己的坐标（需与数据 CRS 一致）
## 这里用 rs_base 的范围中间缩一圈作为示例（保证示例可运行）
ext_full <- extent(rs_base)
dx <- (xmax(ext_full) - xmin(ext_full)) * 0.1
dy <- (ymax(ext_full) - ymin(ext_full)) * 0.1
e <- extent(xmin(ext_full) + dx, xmax(ext_full) - dx,
            ymin(ext_full) + dy, ymax(ext_full) - dy)
message("矩形裁剪范围：", toString(c(xmin(e), xmax(e), ymin(e), ymax(e))))
rs_crop <- crop(rs_base, e)
message("矩形裁剪完成。尺寸(行,列,层)：", paste(dim(rs_crop), collapse = " x "))
```

```
## 可视化：裁剪后的真彩/假彩（根据可用波段选择）
save_seq_plot({
    if (all(c("Red","Green","Blue") %in% names(rs_crop))) {
        plotRGB(rs_crop[[c("Red","Green","Blue")]], scale = 1, stretch = "hist",
                main = "矩形裁剪-真彩（若为反射率则 scale=1）")
    } else if (all(c("NIR","Red","Green") %in% names(rs_crop))) {
        plotRGB(rs_crop[[c("NIR","Red","Green")]], scale =
ifelse(maxValue(rs_crop[[1]]) > 1, maxValue(rs_crop[[1]]), 1),
                stretch = "hist", main = "矩形裁剪-假彩(NIR,Red,Green)")
    } else {
        plot(rs_crop[[1]], main = "矩形裁剪-首层示意", col = gray(0:100/100))
    }
}, "矩形裁剪_预览")
```

裁剪结果如下：

图 3.4.1-1 矩形裁剪结果

3.4.2 空间掩膜

掩膜环节读取 shapefile（Yangdian_Zhiniangyuanzi.shp），以 plotRGB 真彩或假彩渲染裁剪结果。

其中，Yangdian_Zhiniangyuanzi.shp 是我在 GeoScene Pro 3.1 软件中绘制的，由于我的老家（祖宅与田产所在地）——织娘院子（这个名字得名于传说这里曾经有一户杨姓人家有一位老妇善于织布而闻名）——是江汉地区以血缘氏族聚居而形成的“湾”而不是行政村[12]，因此无法获取其矢量编辑，只能依据底图绘制大致范围：

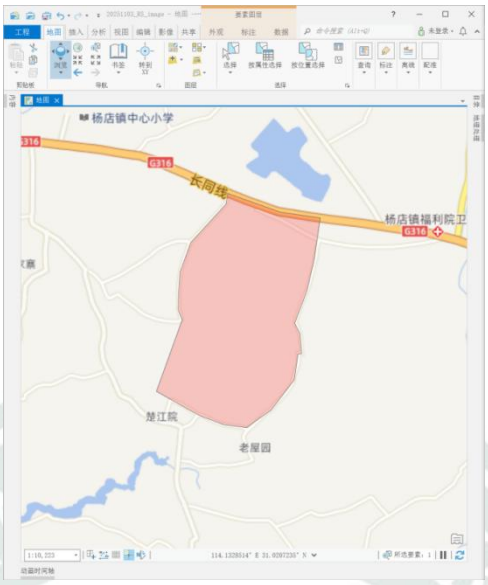


图 3.4.2-1 在GeoScene Pro 3.1 软件中绘制的织娘院子的大致范围矢量边界

```
## 2) 多边形掩膜（mask）
## 优先使用已有 shapefile；若没有，可选交互绘制或用示例矩形转多边形
## 请将下方 shp_path 替换为你的矢量边界文件路径（同一坐标系）
shp_path <- "data/Yangdian_Zhiniangyuanzi.shp"  # 例：shp_path <-
"vector/hengsha_draw.shp"
rs_masked <- NULL
if (!is.null(shp_path) && file.exists(shp_path)) {
    message("检测到 shapefile：", shp_path, "，将用于掩膜 ...")
    ## 建议使用 rgdal 或 terra 读取；为尽量不新增依赖，这里用 raster::shapefile
    suppressWarnings({
        ss <- shapefile(shp_path)
    })
    ## 若坐标系不一致，需投影变换（此处仅在不一致时提示，不自动变换）
    if (!is.na(crs(ss)) && !is.na(crs(rs_crop)) && crs(ss)@projargs !=
crs(rs_crop)@projargs) {
        warning("矢量与栅格 CRS 不一致。请先使用 spTransform() 进行投影统一后再运行 mask()。
")
    }
    rs_masked <- mask(rs_crop, ss)
    message("掩膜完成（基于 shapefile）。")
} else {
    message("未提供 shapefile，使用矩形 extent 生成的示例多边形进行掩膜演示。")
    ## 用矩形 e 转为 SpatialPolygons 作为示例掩膜
    p <- as(e, "SpatialPolygons")
    crs(p) <- crs(rs_crop)
    rs_masked <- mask(rs_crop, p)
    message("掩膜完成（基于示例多边形）。")
}
## 掩膜结果可视化
```

```
save_seq_plot({
    if (!is.null(rs_masked)) {
        if (all(c("NIR","Red","Green") %in% names(rs_masked))) {
            plotRGB(rs_masked[[c("NIR","Red","Green")]],
                    scale = ifelse(maxValue(rs_masked[[1]]) > 1,
maxValue(rs_masked[[1]]), 1),
                    stretch = "hist", main = "掩膜结果-假彩(NIR,Red,Green)")
        } else if (all(c("Red","Green","Blue") %in% names(rs_masked))) {
            plotRGB(rs_masked[[c("Red","Green","Blue")]], scale = 1, stretch = "hist",
                    main = "掩膜结果-真彩(R,G,B)")
        } else {
            plot(rs_masked[[1]], main = "掩膜结果-首层示意", col = gray(0:100/100))
        }
    } else {
        plot(rs_crop[[1]], main = "掩膜失败，显示矩形裁剪首层", col = gray(0:100/100))
    }
}, "掩膜_预览")
## 将裁剪与掩膜结果另存为 GeoTIFF（便于后续复用）
crop_tif  <- next_name("裁剪_栅格", "tif")
mask_tif  <- next_name("掩膜_栅格", "tif")
writeRaster(rs_crop, crop_tif,  overwrite = TRUE)
message("已输出：", crop_tif)
if (!is.null(rs_masked)) {
    writeRaster(rs_masked, mask_tif, overwrite = TRUE)
    message("已输出：", mask_tif)
}
message("1.6 节：空间裁剪与掩膜演示完成。")
```

使用织娘院子的矢量边界的掩膜结果如下：

图 3.4.2-2 织娘院子的矢量边界的掩膜结果（假彩色显示）

与 GeoScene Rro 3.1 软件中绘制的矢量边界（图 3.4.2-1）对比，可以确认确实是织娘院子区域；将其导入 GeoScene Rro 3.1 软件中，掩膜的栅格也可以与矢量边界重合，说明掩膜结果正确：

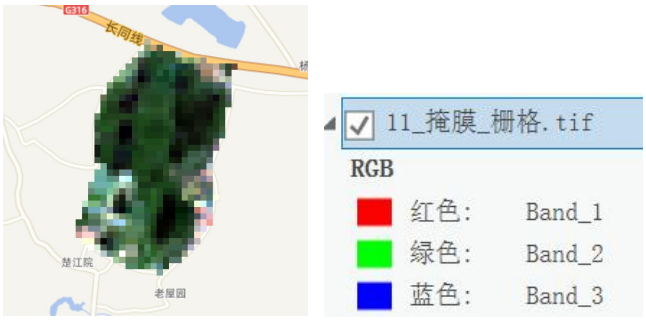


图 3.4.2-3 织娘院子的栅格影像在 GeoScene Rro 3.1 软件中加载的结果（真彩色显示，波段已经过抽取）

3.5 文件保存

为了便于进行后续操作操作，常常需要将读取的 raster 对象存储在本地磁盘上。

3.5.1 储存为 GeoTiff 格式

遥感影像栅格数据的存储格式常为 GeoTiff 格式（以 .tif 为后缀），但由于 GeoTiff 格式无法保存不同通道的名称，所以该属性会丢失。

```
## ---- 1.7 文件保存 ----
## 本节将演示将处理后的 Raster 对象保存为 GeoTIFF（以及可选的 .grd），
## 并说明 GeoTIFF 不保留层名称的注意事项。
message("1.7 节：开始文件保存演示...")
## 选择一个要保存的多层对象，优先使用掩膜结果，其次裁剪结果，再次演示栈
rs_to_save <- if (exists("rs_masked") && !is.null(rs_masked)) {
    message("将使用 rs_masked 作为保存对象。")
    rs_masked
} else if (exists("rs_crop")) {
    message("未发现 rs_masked，改用 rs_crop 作为保存对象。")
    rs_crop
} else if (exists("img_demo_stack")) {
    message("未发现 rs_crop，改用 img_demo_stack 作为保存对象。")
    img_demo_stack
} else {
    message("未发现可用对象，临时使用 stack(b5,b4,b3) 作为保存对象。")
    stack(b5, b4, b3)
}
## 保存为 GeoTIFF（注意：GeoTIFF 不保存层名）
tif_out <- next_name("多层保存_GeoTIFF", "tif")
x <- writeRaster(rs_to_save, filename = tif_out, overwrite = TRUE)
message("已输出（GeoTIFF）：", tif_out)
print(x)
```

在 GeoScene Rro 3.1 软件中打开：

图 3.5.1-1 储存的 GeoTiff 格式的栅格数据（波段名称保留）

可见影像能正确显示，但是波段名称已经丢失。

3.5.2 储存为 raster 的 .grd 格式

此外，还可以保存为 raster 的 .grd 格式，这种格式可以保存影像的通道名，但它不够普适，因为 R 语言的 raster 包在其他语言或者软件中不常用。

```
## 保存为 raster 的 .grd 格式（可保留层名，但通用性较差）
grd_out <- next_name("多层保存_grd", "grd")
xg <- writeRaster(rs_to_save, filename = grd_out, overwrite = TRUE)
message("已输出（GRD）：", grd_out)
print(xg)
message("1.7 节：文件保存演示完成。")
```

在 GeoScene Rro 3.1 软件中打开：

图 3.5.1-1 储存的 .grd 格式的栅格数据（波段名称丢失）

影像能正确显示，且波段名称保留下来了。

3.6 遥感影像不同波段间的关系

对于具有多个波段的遥感影像而言，不同波段直接往往具有一定的相关关系（称为“波段间信息冗余”），通常波长范围相邻的波段的相关性较高[13]。探究遥感影像不同波段间的相关关系有助于更加深入地理解影像信息，从而应用于影像的波段提取、影像压缩等任务中。可以通过画出不同通道的反射率的相关性图，判断不同的通道组合是否包含重复信息。

本节实习中将对两组关键组合绘制 pairs 图：Coastal(B1) vs Blue(B2)，用于检视超蓝与蓝

段的相关性与差异性（对薄雾/水体边界的敏感）；Red(B4) vs NIR(B5)，与植被指数 NDVI 的分子/分母来源直接相关，能直观看到植被与非植被在散点空间的分离程度。

```
## ---- 1.8 不同通道关系 ----
## 使用 pairs() 可视化多波段之间的相关性，确保 Landsat-8 的超蓝(B1)与蓝(B2)必定被选中
message("1.8 节：开始不同通道关系分析...")
## 明确从原始 img 中按名称提取 L8 波段（包含超蓝 B1）
## 说明：前文已定义 pick_band(img, key)，此处直接复用，避免名称不一致导致的匹配失败
b1 <- pick_band(img, "B1") # Coastal/Aerosol (Ultra-blue)
b2 <- pick_band(img, "B2") # Blue
b3 <- pick_band(img, "B3") # Green
b4 <- pick_band(img, "B4") # Red
b5 <- pick_band(img, "B5") # NIR
## 统一命名，便于 pairs 图标题清晰
names(b1) <- "Coastal(B1)"
names(b2) <- "Blue(B2)"
names(b3) <- "Green(B3)"
names(b4) <- "Red(B4)"
names(b5) <- "NIR(B5)"
## 组 1：超蓝 vs 蓝
rs_pairs1 <- stack(b1, b2)
title1 <- paste0(names(rs_pairs1)[1], " vs ", names(rs_pairs1)[2])
## 组 2：红 vs 近红外（NDVI 相关常用组合）
rs_pairs2 <- stack(b4, b5)
title2 <- paste0(names(rs_pairs2)[1], " vs ", names(rs_pairs2)[2])
## pairs 图保存（两张）
save_seq_plot({
    pairs(rs_pairs1, main = paste0("通道关系：", title1))
}, "通道关系_pairs_组 1")
save_seq_plot({
    pairs(rs_pairs2, main = paste0("通道关系：", title2))
}, "通道关系_pairs_组 2")
```

得到的结果图如下：

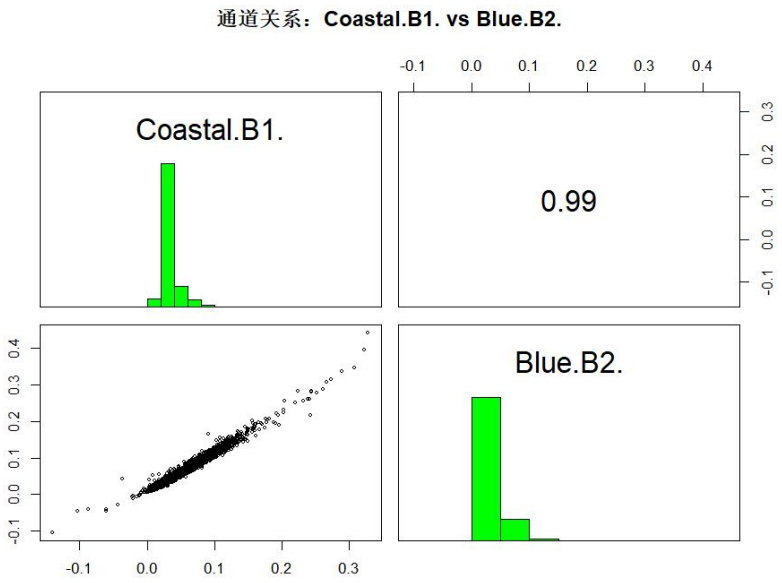


图 3.6-1 超蓝与蓝波段的相关性分析

图中绘制的左上和右下分别为两个波段的频率分布，是检查反射率分布是否合理的依据。左下为两通道对应点反射率的散点图,可见其分布接近一条直线；右上角的线性相关系数为 0.99，可见超蓝和蓝波段的相关性很强，说明采用超蓝和蓝波段的组合一般不能带来更多信息。

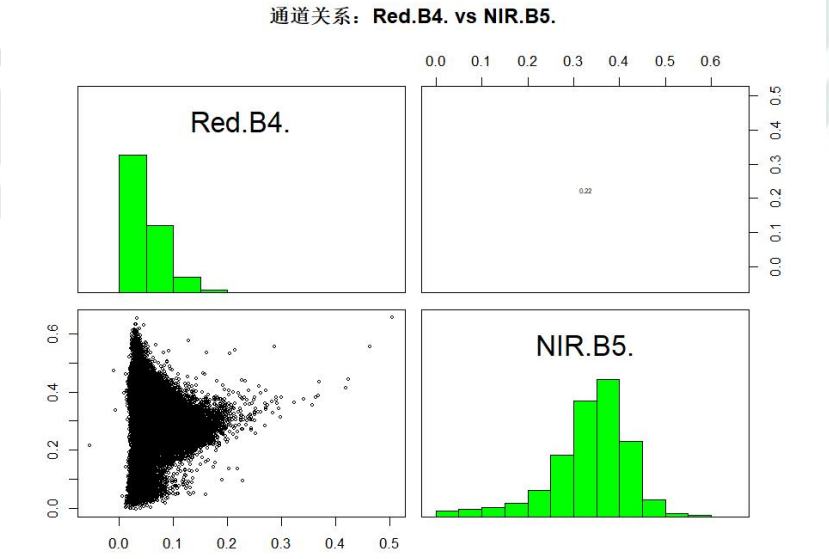


图 3.6-2 近红外与红波段的相关性分析

而近红外与红波段的线性相关程度较低，在散点图中存在明细的三角区。这是因为植被有较高的 NIR 波段的反射率，但是其在 R 波段的反射率较低，因此导致散点集中于 NIR 的轴。

这说明近红外与红波段的组合能够包含更加丰富的信息，这在植被的识别（如 NDVI 指数的计算）等任务中具有重要意义。

3.7 像元级光谱提取

在对地物特征的研究过程中，有时会对个别像素（landmark）的反射率感兴趣。地表的标志物、参照物等经常用来校准遥感影像的辐射值，因此需要把对应像元的光谱信息提取出来。本节将在影像中选取四类典型地物点（Residential/Farmland/Water/Road），提取其 SR_B1~SR_B7 的像元级反射率谱，并以曲线形式展示差异，验证不同地物在可见-近红外-短波红外的典型响应特征，指导指数选择与分类特征构建。

```
## ---- 1.9 提取像素（居民区/农田/水体/道路） ----
## 直接使用预设点；在原始影像范围内可视化并显著标注；仅基于 SR_B1~SR_B7 绘制光谱
message("1.9 节：开始像素提取（居民区/农田/水体/道路，使用预设点）...")
## 1) 明确从原始 img 中按名称提取 L8 必要波段（用于底图与光谱）
b1 <- pick_band(img, "B1") # Ultra-blue
b2 <- pick_band(img, "B2") # Blue
b3 <- pick_band(img, "B3") # Green
b4 <- pick_band(img, "B4") # Red
b5 <- pick_band(img, "B5") # NIR
## 2) 构造用于底图显示的 RGB 组合（原始 img）
s_true_img  <- stack(b4, b3, b2) # R,G,B
s_false_img <- stack(b5, b4, b3) # NIR,R,G
## 自动判定显示 scale（按原始 DN）
maxDN_img <- max(c(maxValue(b2), maxValue(b3), maxValue(b4), maxValue(b5)), na.rm =
TRUE)
plot_scale_img <- if (is.finite(maxDN_img) && maxDN_img > 1) maxDN_img else 1
## 3) 直接构造三个预设点（确保位于原始 img 范围内）
ext_img <- extent(img)  # 原始文件（img）的空间范围
cx <- (xmin(ext_img) + xmax(ext_img)) / 2
cy <- (ymin(ext_img) + ymax(ext_img)) / 2
dx <- (xmax(ext_img) - xmin(ext_img)) * 0.03
dy <- (ymax(ext_img) - ymin(ext_img)) * 0.03
a_df <- data.frame(
    x = c(114.107493, 114.123371,  114.073097, 114.100777),  # 居民区、农田、水体、道路
    y = c(31.027838,  31.021896,   31.030039,  31.032147)
)
labels <- c("Residential", "Farmland","Water","Road")
## 4) 底图可视化并显著标注三个点
save_seq_plot({
    ## 优先真彩显示；若失败则退回假彩；再退回单层
    ok <- FALSE
    try({
```

```
        plotRGB(s_true_img, scale = plot_scale_img, stretch = "hist",
                main = "1.9 预设点（原始影像）：Residential / Farmland / Water / Road（真彩）")
        ok <- TRUE
    }, silent = TRUE)
    if (!ok) {
        try({
            plotRGB(s_false_img, scale = plot_scale_img, stretch = "hist",
                    main = "1.9 预设点（原始影像）：Residential / Farmland / Water / Road（假彩）")
            ok <- TRUE
        }, silent = TRUE)
    }
    if (!ok) {
        plot(b4, main = "1.9 预设点（原始影像）：Residential / Farmland / Water / Road（单层示意）",
             col = gray(0:100/100))
    }
    ## 更显著的点与标注
    cols <- c("#e7298a", "#1b9e77", "#2550b9", "#d95f02")
    points(a_df$x, a_df$y, pch = 21, bg = cols, col = "black", cex = 2.2, lwd = 1.5)
    ## 为每个点添加文字与引线
    text_offset <- max(res(b4)) * 5  # 根据分辨率拉开标注距离
    for (i in 1:3) {
        segments(a_df$x[i], a_df$y[i],
                 a_df$x[i] + text_offset, a_df$y[i] + text_offset,
                 col = cols[i], lwd = 2)
        text(a_df$x[i] + text_offset, a_df$y[i] + text_offset,
             labels[i], col = cols[i], cex = 1.2, font = 2)
    }
}, "预设点_底图预览_原始影像")
## 5) 仅选择光谱相关波段进行提取与绘图（SR_B1 ~ SR_B7）
## 根据你提供的层名，筛选 SR_B1~SR_B7
all_names <- names(img)
spec_idx <- grep("^SR_B[1-7]$", all_names, ignore.case = FALSE)
if (length(spec_idx) == 0L) {
    stop("在影像中未发现光谱波段 SR_B1~SR_B7，请检查图层名称。")
}
img_spec <- subset(img, spec_idx)
## 统一命名为 SR_B1..SR_B7 的显式顺序（如存在）
names(img_spec) <- all_names[spec_idx]
## 6) 提取像素的光谱（仅 SR_B1~SR_B7）
spec_mat <- raster::extract(img_spec, a_df)
## 7) 保存数值到 CSV（输出顺序与 img_spec 的层顺序一致）
```

```
csv_out <- next_name("像素光谱_SR_B1_B7", "csv")
utils::write.csv(cbind(class = labels, as.data.frame(spec_mat)),
                 file = csv_out, row.names = FALSE, fileEncoding = "UTF-8")
message("已输出像素光谱 CSV（SR_B1~SR_B7）：", csv_out)
## 8) 绘制四类地物的光谱曲线（仅 SR_B1~SR_B7，横轴使用物理含义命名）
save_seq_plot({
    mat <- as.matrix(spec_mat)
    ## 构建层名 -> 物理含义 的映射表（仅对 SR_B1..SR_B7）
    band_map <- c(
        SR_B1 = "Coastal\\Aerosol\n",
        SR_B2 = "Blue\n",
        SR_B3 = "Green\n",
        SR_B4 = "Red\n",
        SR_B5 = "NIR\n",
        SR_B6 = "SWIR1\n",
        SR_B7 = "SWIR2\n"
    )
    orig_names <- names(img_spec)          # e.g., "SR_B1" ... "SR_B7"
    pretty_names <- ifelse(orig_names %in% names(band_map),
                           paste0(band_map[orig_names], " (", orig_names, ")"),
                           # unname(band_map[orig_names]),
                           orig_names)      # 兜底：未匹配就用原名
    colnames(mat) <- pretty_names
    y_min <- suppressWarnings(min(mat, na.rm = TRUE))
    y_max <- suppressWarnings(max(mat, na.rm = TRUE))
    if (!is.finite(y_min) || !is.finite(y_max)) {
        y_min <- 0; y_max <- 1
    }
    plot(0, 0, type = "n",
         xlim = c(1, ncol(mat)), ylim = c(y_min, y_max),
         xaxt = "n", xlab = "Spectral Bands", ylab = "Reflectance (SR)",
         main = "四类地物光谱（SR_B1~SR_B7）")
    axis(1, at = seq_len(ncol(mat)), labels = colnames(mat), las = 1, cex.axis = 0.9)
    cols <- c("#e7298a", "#1b9e77", "#2550b9", "#d95f02")
    for (i in 1:nrow(mat)) {
        lines(seq_len(ncol(mat)), mat[i,], type = "l", lwd = 3, col = cols[i])
        points(seq_len(ncol(mat)), mat[i,], pch = 16, col = cols[i], cex = 1.2)
    }
    legend("topleft", legend = labels, col = cols, lwd = 3, pch = 16, bty = "n")
}, "四类地物光谱_SR_B1_B7")
## 9) 输出点坐标
pts_csv <- next_name("点坐标_预设", "csv")
utils::write.csv(data.frame(class = labels, a_df), file = pts_csv,
                 row.names = FALSE, fileEncoding = "UTF-8")
```

```
message("已输出点坐标 CSV：", pts_csv)
message("1.9 节：像素提取与光谱绘制完成（使用预设点，SR_B1~SR_B7）。")
```

预设点的选择可视化如下：

图 3.7-1 选取的四类典型地物点，用于进行像元级光谱提取

对应的四种地物像元光谱信息如下：

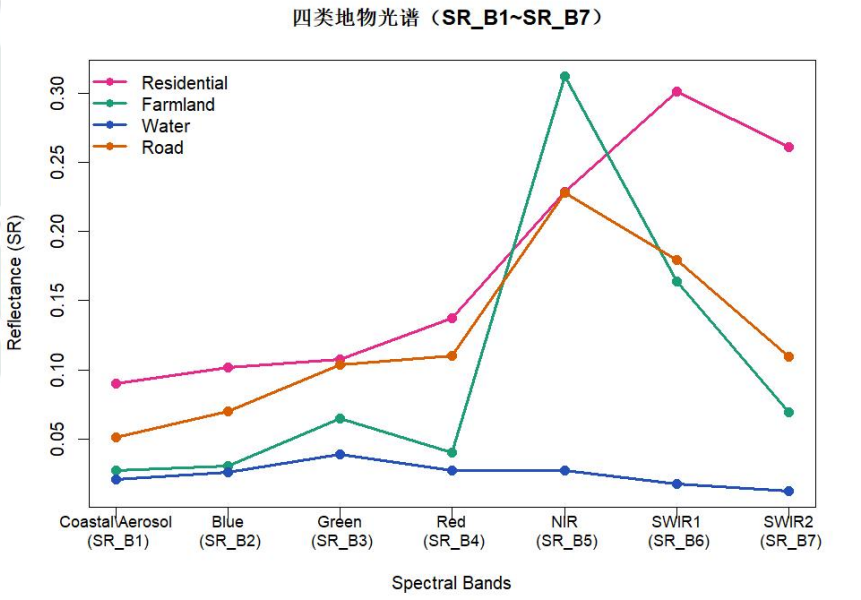


图 3.7-2 选取的四类典型地物点的光谱曲线

选择的四类地物分别为居民区、农田、水体和道路，由图可见其光谱特征非常典型：植被（Farmland）在 B5（NIR）应显著抬升，Red（B4）较低，曲线呈“红低-近红外高”的经典“红边效应”；水体（Water）在 NIR/SWIR 段反射最低，全谱偏暗；建成区/道路（Residential/Road）在可见光相对较亮，NIR 升幅不如植被明显，SWIR 上行常见，利于与植被分离。

上述地物的光谱曲线及其特征差异与后续基于 NDVI/NDWI/NDBI/SAVI 等遥感指数的

地物识别、无监督与监督的地物分类等任务提供了理论支撑。

3.8 遥感指数（NDVI/NDWI/NDBI/SAVI）的计算与阈值掩膜

本节利用 Landsat 8 表面反射率波段计算常用指数：归一化植被指数 NDVI[14]、归一化水体指数 NDWI[15]、归一化建筑指数 NDBI[16] 以及考虑土壤背景的土壤调整植被指数 SAVI[17]；在此基础上使用 Otsu 法自动选阈值，对水体与建筑区域进行二元掩膜。

遥感指数	计算公式	原理
归一化植被指数 NDVI	$NDVI = \frac{NIR - Red}{NIR + Red}$	植被在 NIR 高反射、在 Red 强吸收，因此 NDVI 对植被敏感
归一化水体指数 NDWI	$NDWI = \frac{Green - NIR}{Green + NIR}$	水体在可见绿较亮、在 NIR 吸收强，指数对水体增强
归一化建筑指数 NDBI	$NDBI = \frac{SWIR1 - NIR}{SWIR1 + NIR}$	建设用地在 SWIR1 反射通常高于 NIR，指数对城建区敏感
土壤调整植被指数 SAVI	$SAVI = 1.5 \times \frac{NIR - Red}{NIR + Red + 0.5}$	通过 L 因子抑制土壤背景，适合稀疏植被的监测

表 3.8-1 遥感指数

上述遥感指数的依据也可以从图 3.7-2 的光谱曲线中看出。

Otsu 法又称最大类间方差法、大津法，基于图像的灰度直方图双峰分布特性将图像像素分为前景和背景两类，通过计算类间方差σ^2的最大化来确定最佳分割阈值，当类间方差最大时，背景与目标之间的错分概率达到最小[18]。该方法通过阈值T将图像分割后的两类概率分别记为W_A和W_B，平均灰度记为μ_A和μ_B。实际计算中，可以简化使得为$\sigma^2 = W_A W_B (\mu_A - \mu_B)^2$最大。

```
# ---- 2 遥感影像的数学计算 ----
## ---- 2.1 遥感指数 ----
## 本节将实现包括植被指数（NDVI）在内的各种遥感指数的计算
## 本节将 NDVI 定义为 (NIR - Red)/(NIR + Red)
message("2.1 节：开始计算遥感指数...")
## 载入 RColorBrewer 以改善调色板
if (!requireNamespace("RColorBrewer", quietly = TRUE)) {
    warning("请先安装 RColorBrewer 包：install.packages('RColorBrewer')")
    # 如果包不存在，提供一个兜底的调色板函数
    brewer.pal <- function(n, name) {
        if (name == "RdYlBu") return(colorRampPalette(c("#A50026", "#D73027", "#F46D43",
"#FDAE61", "#FEE090", "#FFFFBF", "#E0F3F8", "#ABD9E9", "#74ADD1", "#4575B4",
"#313695"))(n))
```

```
        if (name == "RdYlGn") return(colorRampPalette(c("#A50026", "#D73027", "#F46D43",
"#FDAE61", "#FEE08B", "#FFFFBF", "#D9EF8B", "#A6D96A", "#66C2A5", "#1A9850",
"#006837"))(n))
        return(rev(terrain.colors(n))) # 默认回退
    }
}
## --- 自动阈值函数 (Otsu) ---
## 从 2.3 节中提取，用于自动确定 NDWI/NDBI 阈值
otsu_threshold <- function(x, nbins = 256, xmin = -1, xmax = 1){
    v <- values(x); v <- v[is.finite(v)]
    # 钳制数据到理论范围
    v <- pmin(xmax, pmax(xmin, v))
    if (length(v) < nbins) {
        warning("Otsu: 样本点过少，可能返回不可靠阈值。")
        return(as.numeric(quantile(v, 0.5, na.rm=TRUE))) # 返回中位数
    }
    h <- hist(v, breaks = seq(xmin, xmax, length.out = nbins+1), plot = FALSE)
    p <- h$counts / sum(h$counts)
    omega <- cumsum(p)
    mu <- cumsum(p * (h$mids))
    mu_t <- mu[length(mu)]
    # 避免除以 0
    sigma_b2 <- (mu_t*omega - mu)^2 / (omega*(1-omega) + 1e-12)
    # 查找最大方差对应的阈值
    if (all(!is.finite(sigma_b2)) || !any(sigma_b2 > 0)) {
        warning("Otsu: 未能计算有效类间方差，返回中位数。")
        return(as.numeric(quantile(v, 0.5, na.rm=TRUE)))
    }
    t_idx <- which.max(sigma_b2)
    h$mids[t_idx]
}
## --- NDVI (植被指数) ---
## 通用 VI 函数：从多层 Raster 对象中选取第 k 与 i 层计算 (bk - bi) / (bk + bi)
vi <- function(img_stack, k, i) {
    bk <- img_stack[[k]]
    bi <- img_stack[[i]]
    (bk - bi) / (bk + bi)
}
## 选择用于计算 NDVI 的输入数据源与层索引
## - 若存在 img_ref（顺序为: 1=Blue, 2=Green, 3=Red, 4=NIR），则用 4 与 3
## - 否则使用 b5(NIR) 与 b4(Red) 直接计算
if (exists("img_ref")) {
    ndvi <- vi(img_ref, k = 4, i = 3)  # NIR, Red（反射率）
    ndvi_source <- "反射率(img_ref)"
```

```
} else {
    ## 究底确保 b4/b5 存在（此前已定义），若未定义则从 img 中提取
    if (!exists("b4") || !exists("b5")) {
        b4 <- pick_band(img, "B4")
        b5 <- pick_band(img, "B5")
    }
    ndvi <- (b5 - b4) / (b5 + b4)      # 以 DN 近似
    ndvi_source <- "原始 DN(b5,b4)"
}
## 输出 NDVI GeoTIFF
ndvi_tif_21 <- next_name("NDVI", "tif")
writeRaster(ndvi, ndvi_tif_21, overwrite = TRUE)
message("2.1 节：已输出 NDVI GeoTIFF：", ndvi_tif_21, "（数据来源：", ndvi_source, "）
")
## 绘制 NDVI（修改为 RdYlGn 调色板，绿色代表高植被）
save_seq_plot({
    plot(ndvi,
         col = brewer.pal(11, "RdYlGn"), # 红(低)-黄-绿(高)
         zlim = c(-1, 1),
         main = "Landsat NDVI（2.1 植被指数）\n(高值为绿色)")
}, "NDVI_渲染图")
## --- 其他遥感指数：NDWI/NDBI/SAVI ---
g  <- pick_band(img, "B3")
sw1<- pick_band(img, "B6")
nir<- pick_band(img, "B5")
red<- pick_band(img, "B4")
ndwi <- (g - nir) / (g + nir)            # McFeeters（绿-近红外）
ndbi <- (sw1 - nir) / (sw1 + nir)        # 建筑指数
savi <- (1.5*(nir - red)) / (nir + red + 0.5)  # L=0.5
writeRaster(ndwi, next_name("NDWI","tif"), overwrite=TRUE)
writeRaster(ndbi, next_name("NDBI","tif"), overwrite=TRUE)
writeRaster(savi, next_name("SAVI","tif"), overwrite=TRUE)
## --- 指数可视化（使用新调色板） ---
save_seq_plot({
    # 添加 zlim = c(-1, 1)，将颜色固定在理论范围，使水体(高值/蓝色)更突出。
    plot(ndwi,
         col = brewer.pal(11, "RdYlBu"), # 红(低)-黄-蓝(高)
         zlim = c(-1, 1),
         main = "NDWI（水体敏感）\n(高值为蓝色)")
}, "NDWI")
save_seq_plot({
    # 添加 zlim = c(-1, 1)，将颜色固定在理论范围，使建筑(高值/红色)更突出。
    # 注意：rev() 在此是正确的，因为 "RdYlBu" 中 Bu 是高值，反转后 Rd 是高值。
    plot(ndbi,
```

```
        col = rev(brewer.pal(11, "RdYlBu")), # 蓝(低)-黄-红(高)
        zlim = c(-1, 1),
        main = "NDBI（建筑敏感）\n(高值为红色)")
}, "NDBI")
save_seq_plot({
    # 添加 zlim = c(-1, 1)，将颜色固定在理论范围。
    plot(savi,
        col = brewer.pal(11, "RdYlGn"), # 红(低)-黄-绿(高)
        zlim = c(-1, 1),
        main = "SAVI（土壤背景抑制）\n(高值为绿色)")
}, "SAVI")
## --- 掩膜阈值（使用 Otsu 自动阈值） ---
message("正在使用 Otsu 自动计算 NDWI/NDBI 阈值...")
# NDWI 阈值（理论范围 -1 到 1）
ndwi_thr_auto <- otsu_threshold(ndwi, nbins=256, xmin=-1, xmax=1)
message(sprintf("Otsu 自适应 NDWI 阈值（水体 > ...): %.3f", ndwi_thr_auto))
# NDBI 阈值（理论范围 -1 到 1）
ndbi_thr_auto <- otsu_threshold(ndbi, nbins=256, xmin=-1, xmax=1)
message(sprintf("Otsu 自适应 NDBI 阈值（建筑 > ...): %.3f", ndbi_thr_auto))
## 简单水体/建筑掩膜（使用自动阈值）
water_mask <- ndwi > ndwi_thr_auto
urban_mask <- ndbi > ndbi_thr_auto
# 增加掩膜的预览图，确保图例为 0 和 1
save_seq_plot({
    plot(water_mask, main=sprintf("掩膜_水体 (NDWI > %.3f)", ndwi_thr_auto),
        col=c("#FFFFFF", "#0000FF"), zlim=c(0,1), legend=TRUE,
        legend.args=list(text=c('非水体','水体'), side=4, line=2.5))
}, "掩膜_水体_预览")
save_seq_plot({
    plot(urban_mask, main=sprintf("掩膜_建筑 (NDBI > %.3f)", ndbi_thr_auto),
        col=c("#FFFFFF", "#FF0000"), zlim=c(0,1), legend=TRUE,
        legend.args=list(text=c('非建筑','建筑'), side=4, line=2.5))
}, "掩膜_建筑_预览")
# 保存 TIF 文件
writeRaster(water_mask, next_name("掩膜_水体","tif"), overwrite = TRUE)
writeRaster(urban_mask, next_name("掩膜_建筑","tif"), overwrite = TRUE)
message("2.1 节：指数计算、可视化与自动掩膜完成。")
```

得到的结果图如下：

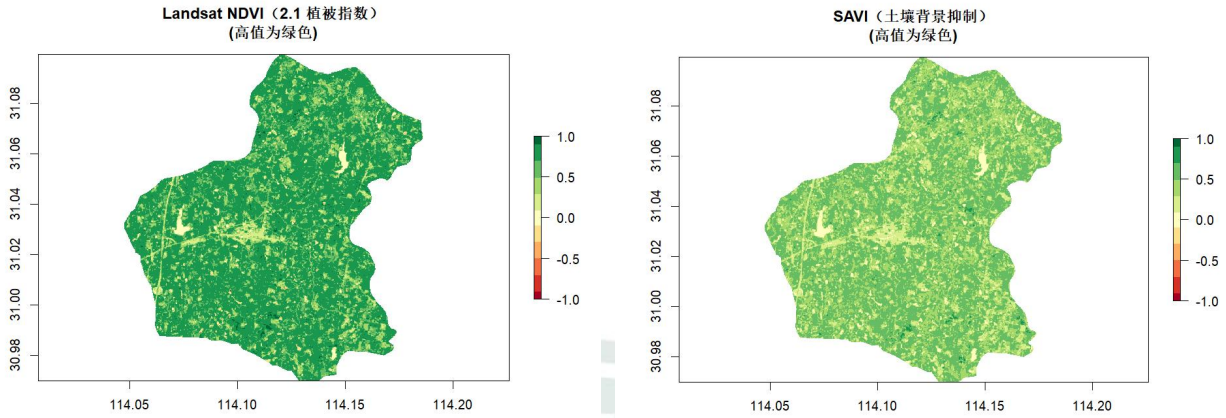


图 3.8-1 归一化植被指数 NDVI（左）和土壤调整植被指数 SAVI（右）

图中，NDVI 空间分布高值主要对应耕地与林地斑块，与前文假彩色（B5/B4/B3）图（图 3.3.1-3 右）中的亮红区域一致，二者在植被识别上形成互证。其中的植被指数低值区域为 G4 京港澳高速、杨店镇镇区、水库等（可以在下文中的与已有底图对比局部示例图中看出）。SAVI 的结果与 NDVI 的结果基本一致，且抑制了一部分土壤区域的干扰。

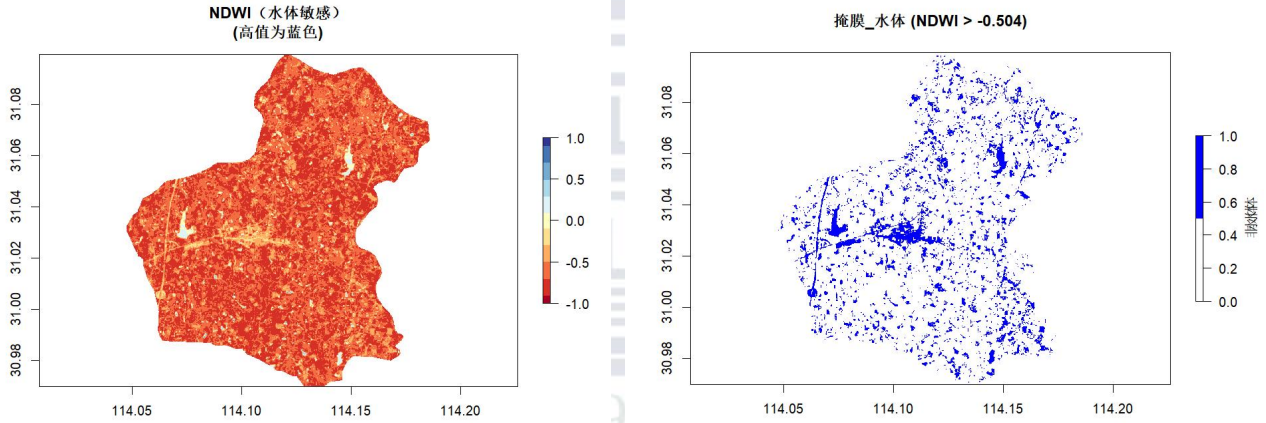


图 3.8-2 归一化水体指数 NDWI（左）及其阈值掩膜结果（右）

图 3.8-3 与已有底图对比局部示例

图中，NDWI 高值沿河道、塘库等水域呈连续带状，局部低洼积水地亦有零散响应，由由图可知其结果符合长江中下游平原江汉平原河湖密布、湖北“千湖之省”的地理特点；但在道路区阴影处出现弱假阳性，后续可结合 QA/阴影掩膜或多时相进行抑制，或在地物的无监督/监督分类时通过考虑更多波段的综合信息来加以区分。

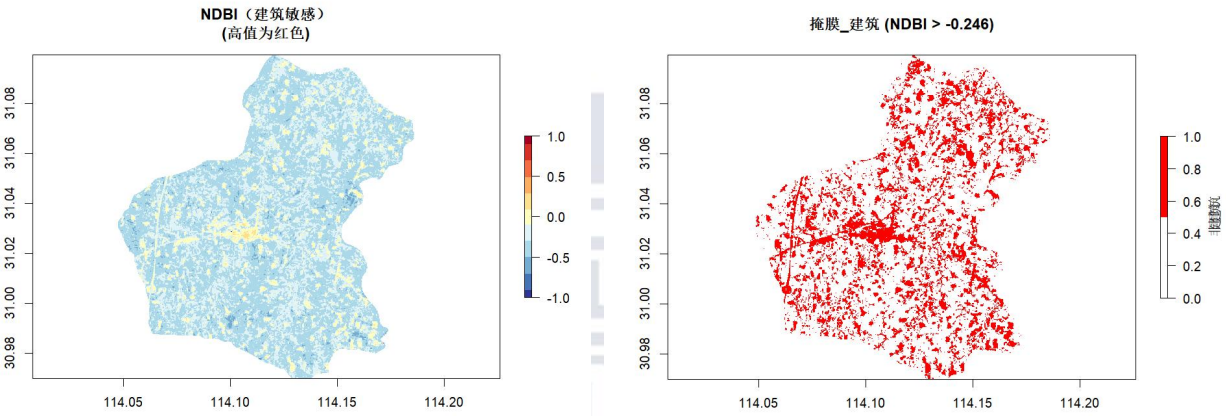


图 3.8-4 归一化建筑指数 NDBI（左）及其阈值掩膜结果（右）

图 3.8-5 与已有底图对比局部示例

图中，NDBI 在主干道和居民点、高反射工业厂房等区域显著抬升，能较好指示建设用地骨架，如 G4 京港澳高速和杨店镇区都能够从右图中识别出来；部分裸土地块在 SWIR 反射较高，会产生一定混淆。

3.9 遥感指数值分布的频率直方图

为理解指数的整体分布形态与峰值位置，本节对 NDVI/NDWI/NDBI/SAVI 绘制 2×2 直方图面板，为阈值选取与分类特征构建提供依据。

```
## ---- 2.2 频率直方图（NDVI/NDWI/NDBI/SAVI 分布） ----
## 说明：本节基于 PDF 2.2，将 NDVI 的分布绘制为直方图
## 使用 2x2 画布，显示所有四个主要指数的直方图
message("2.2 节：开始绘制 NDVI, NDWI, NDBI, SAVI 频率直方图...")
save_seq_plot({
    # 设置 2x2 绘图布局
    op <- par(mfrow = c(2, 2), mar = c(5, 4, 4, 2) + 0.1) # 恢复默认 mar 并设置 mfrow
    on.exit(par(op), add = TRUE) # 确保在函数退出时恢复布局
    # 增加 breaks = 50 以提高精细度。
    # 1. NDVI 直方图
    hist(ndvi,
         main  = "Distribution of NDVI values",
         xlab  = "NDVI",
         ylab  = "Frequency",
         col   = "wheat",
         breaks = 50)
    # 2. NDWI 直方图
    hist(ndwi,
         main  = "Distribution of NDWI values",
         xlab  = "NDWI (Water)",
         ylab  = "Frequency",
         col   = "lightblue",
         breaks = 50)
    # 3. NDBI 直方图
    hist(ndbi,
         main  = "Distribution of NDBI values",
         xlab  = "NDBI (Built-up)",
         ylab  = "Frequency",
         col   = "lightcoral",
         breaks = 50)
    # 4. SAVI 直方图
    hist(savi,
         main  = "Distribution of SAVI values",
         xlab  = "SAVI (Vegetation)",
         ylab  = "Frequency",
         col   = "lightgreen",
         breaks = 50)
}, "Indices_Histograms_2x2") # 修改了保存标签
message("2.2 节：四合一指数直方图已保存。")
```

得到的结果图如下：

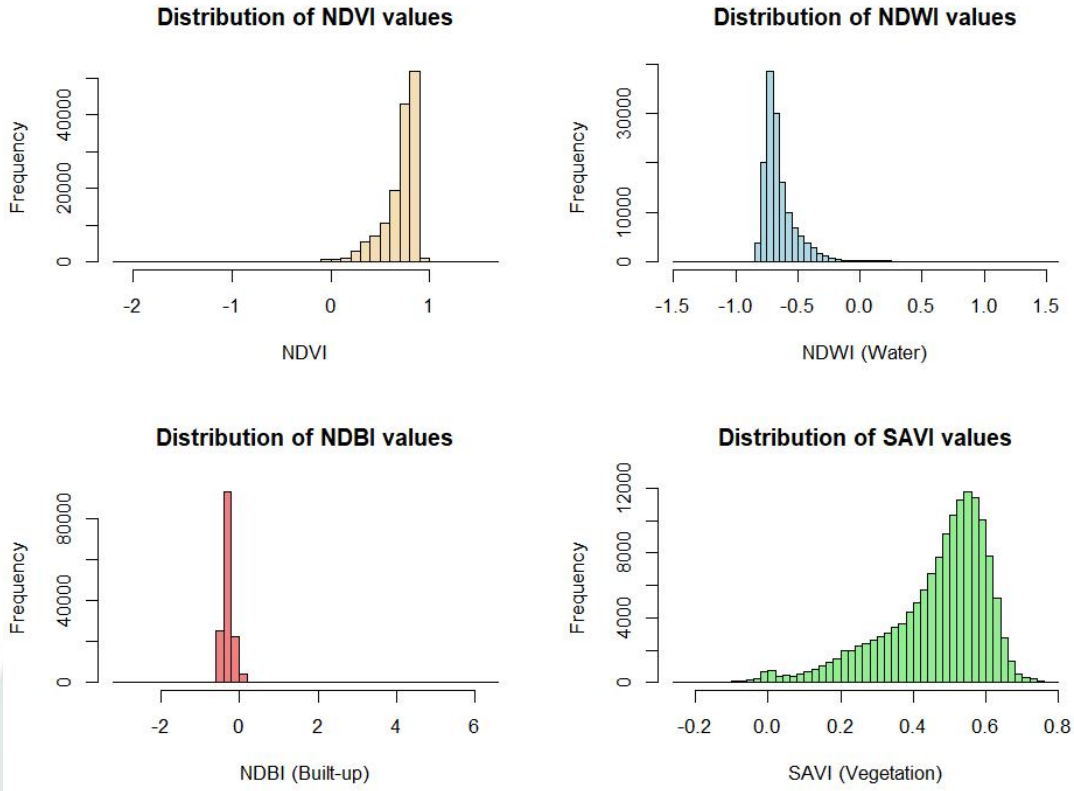


图 3.9-1 四种遥感指数的频率分布直方图

由图可见：NDVI 分布在该夏季遥感影像样本中右偏（0.5–0.8 段概率较高），符合夏季作物及林地高反射特性。NDVI 呈现单峰分布，这是由于在农村地区，大部分土地均为农田或林地，因此房屋等 NDVI 低值像元由于数量较少未形成峰值。若曲线呈现出多峰分布，则往往是城市等居民区集中的区域；由于水体分布面积并不广，NDWI 呈现出偏向低值的单峰分布，但由于河湖广布，因此曲线在高值有长尾；NDBI 在非城市化区域多集中于低值，但在道路/村镇密集处向高值拉伸；SAVI 相比 NDVI 在低植被覆盖区的分布更“抬升”一些，体现土壤背景抑制的效果，便于区分稀疏植被与裸地。

3.10 基于 NDVI 的阈值分割与多级分级

在研究区域夏季（2023-07-27）的遥感影像上，使用 Otsu 法对 NDVI 进行自适应阈值化，得到二元植被掩膜，并按夏季典型分布对 NDVI 进行四级分级，支持在真彩底图上的半透明叠加检视。

3.10.1 基于 NDVI 的阈值分割

通过地物光谱反射率以及它们的组合（如 NDVI，代表了植被覆盖的水平）可以进行地物分类，这是基于像素光谱特征的分类方法。还有阴影、纹理、大小、形状等特征，这些都需要将待分地物作为一个整体进行考虑，因此称之为面向对象的分割和面向像素的分割。仍使用 Otsu 法自动确定植被阈值 t_veg，使类间方差最大。

```
## ---- 2.3 分类临界值（基于 NDVI，2023-07-27 夏季） ----
## 目标：
## 1) 通过阈值将 NDVI 分为“植被/非植被”（NaN 掩膜非植被，便于突出植被）
## 2) 进一步细分多个 NDVI 等级（更贴合夏季 NDVI 较高的分布）
## 3) 输出分类结果 GeoTIFF 与配图
message("2.3 节：开始基于 NDVI 的分类与阈值分割（2023-07-27 夏季）...")
## 保障：若 2.1 节未定义 ndvi，这里回退计算一次
if (!exists("ndvi")) {
    message("未发现 ndvi 对象，按 b5/b4 兜底即时计算 NDVI。")
    if (!exists("b4") || !exists("b5")) {
        b4 <- pick_band(img, "B4")
        b5 <- pick_band(img, "B5")
    }
    ndvi <- (b5 - b4) / (b5 + b4)
}
## Otsu 阈值（针对NDVI）
otsu_threshold <- function(x, nbins = 256, xmin = -0.5, xmax = 1){
    v <- values(x); v <- v[is.finite(v)]
    v <- pmin(xmax, pmax(xmin, v))
    h <- hist(v, breaks = seq(xmin, xmax, length.out = nbins+1), plot = FALSE)
    p <- h$counts / sum(h$counts)
    omega <- cumsum(p)
    mu <- cumsum(p * (h$mids))
    mu_t <- mu[length(mu)]
    sigma_b2 <- (mu_t*omega - mu)^2 / (omega*(1-omega) + 1e-12)
    t_idx <- which.max(sigma_b2)
    h$mids[t_idx]
}
ndvi_thr_auto <- otsu_threshold(ndvi)
message(sprintf("Otsu 自适应 NDVI 阈值：%.3f", ndvi_thr_auto))
# ndvi_thr <- 0.20 # 简单阈值：植被掩膜（夏季设定 NDVI > 0.20）
ndvi_thr <- if (is.finite(ndvi_thr_auto)) ndvi_thr_auto else 0.20 # Otsu 自适应阈值选择
veg_mask <- reclassify(ndvi, cbind(-Inf, ndvi_thr, NA))  # <=阈值 置为 NA，>阈值 保留
save_seq_plot({
    plot(veg_mask,
         main = sprintf("2023-07-27 夏季 植被分布（NDVI > %.2f）", ndvi_thr),
```

```
        col = rev(terrain.colors(10)))
}, "NDVI_阈值_植被掩膜_20230727")
veg_mask_tif <- next_name("NDVI_阈值_植被掩膜_20230727", "tif")
writeRaster(veg_mask, veg_mask_tif, overwrite = TRUE)
message("已输出 NDVI 植被掩膜 GeoTIFF：", veg_mask_tif)
```

阈值分割结果如下：

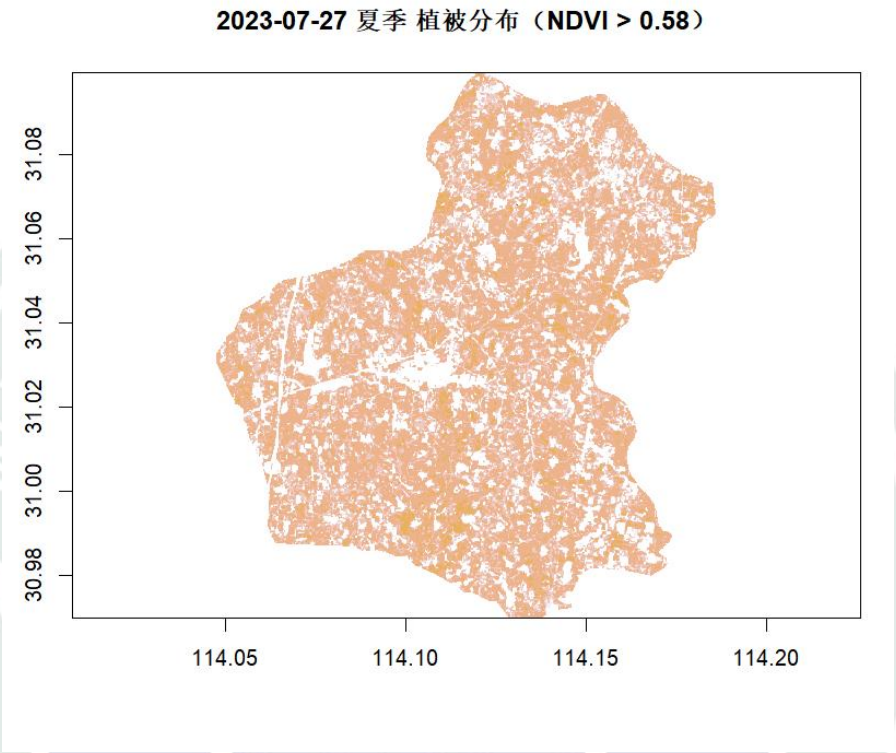


图 3.10.1-1 阈值分割结果

上图中偏黄色的部分即为所得植被掩膜，对比上述植被掩膜与真彩底图（图 3.3.1-3 左），可以发现其所描述的植被区域高度一致，耕地块状分布连续，林带边界清晰，居民点内部与道路区域基本被正确剔除。

3.10.2 基于 NDVI 的多级分级

进一步地，可以结合直方图峰值位置与夏季作物/林木反射特点，将 NDVI 细分为低、中、高、极高四档：

(0.20, 0.40] 低度植被

(0.40, 0.60] 中等植被

(0.60, 0.80] 高植被

(0.80, +∞) 极高植被

```
## 2.3.2 多级分类（夏季区间，更贴合直方图峰值在 0.6~0.85）
## 区间（左开右闭）：
##  (-Inf,0.20] -> 0（非植被，置 NA 以便可视化透明）
```

```
##  (0.20,0.40] -> 1（低度植被/稀疏）
##  (0.40,0.60] -> 2（中等植被）
##  (0.60,0.80] -> 3（高植被）
##  (0.80, Inf) -> 4（极高植被/茂盛冠层）
## 可视化时 0 作为 NA，不参与渲染
class_mat <- matrix(c(
    -Inf, 0.20, NA,
    0.20, 0.40, 1,
    0.40, 0.60, 2,
    0.60, 0.80, 3,
    0.80,  Inf, 4
), ncol = 3, byrow = TRUE)
ndvi_class <- reclassify(ndvi, class_mat)
## 自定义调色板（由低到高：黄-淡绿-绿-深绿）
pal_class <- c("#fee08b", "#a6d96a", "#1a9850", "#006837")
save_seq_plot({
    plot(ndvi_class, col = pal_class, zlim = c(1,4), legend = TRUE,
         main = "NDVI 多级分类（夏季 2023-07-27）\n 0.20–0.40:低 | 0.40–0.60:中 | 0.60–0.80:
高 | >0.80:极高")
}, "NDVI_多级分类_20230727")
ndvi_class_tif <- next_name("NDVI_多级分类_20230727", "tif")
writeRaster(ndvi_class, ndvi_class_tif, overwrite = TRUE)
message("已输出 NDVI 多级分类 GeoTIFF：", ndvi_class_tif)
## 2.3.3 在真彩底图上叠加分类结果，直观核对
## 优先使用 1.4 节的真彩 s_true（B4,B3,B2），否则临时组装
if (!exists("s_true")) {
    if (!exists("b2") || !exists("b3") || !exists("b4")) {
        b2 <- pick_band(img, "B2"); b3 <- pick_band(img, "B3"); b4 <- pick_band(img, "B4")
    }
    s_true <- stack(b4, b3, b2)
}
## 自动 scale 判定：若是 16bit DN 则取其最大值，否则 1（反射率）
maxDN_overlay <- max(c(maxValue(s_true[[1]]), maxValue(s_true[[2]]),
maxValue(s_true[[3]])), na.rm = TRUE)
scale_overlay <- if (is.finite(maxDN_overlay) && maxDN_overlay > 1) maxDN_overlay else 1
## 叠加预览 1：植被掩膜（半透明）
save_seq_plot({
    plotRGB(s_true, scale = scale_overlay, stretch = "hist",
            main = "叠加预览：真彩 + NDVI 植被掩膜(> 0.20) | 2023-07-27")
    plot(veg_mask, add = TRUE, legend = FALSE, alpha = 0.45)
}, "叠加预览_真彩_植被掩膜_20230727")
## 叠加预览 2：多级分类（隐藏 0 类；半透明渲染 1~4）
save_seq_plot({
    plotRGB(s_true, scale = scale_overlay, stretch = "hist",
```

```
        main = "叠加预览：真彩 + NDVI 多级分类（2023-07-27）")
    ## 仅渲染 1:4，0/NA 不显示
    plot(ndvi_class, add = TRUE, legend = FALSE, col = pal_class, alpha = 0.45, zlim = c(1,4))
}, "叠加预览_真彩_多级分类_20230727")
message("2.3 节：NDVI 阈值分类与叠加预览完成（2023-07-27 夏季）。")
```

多级分类的结果图如下：

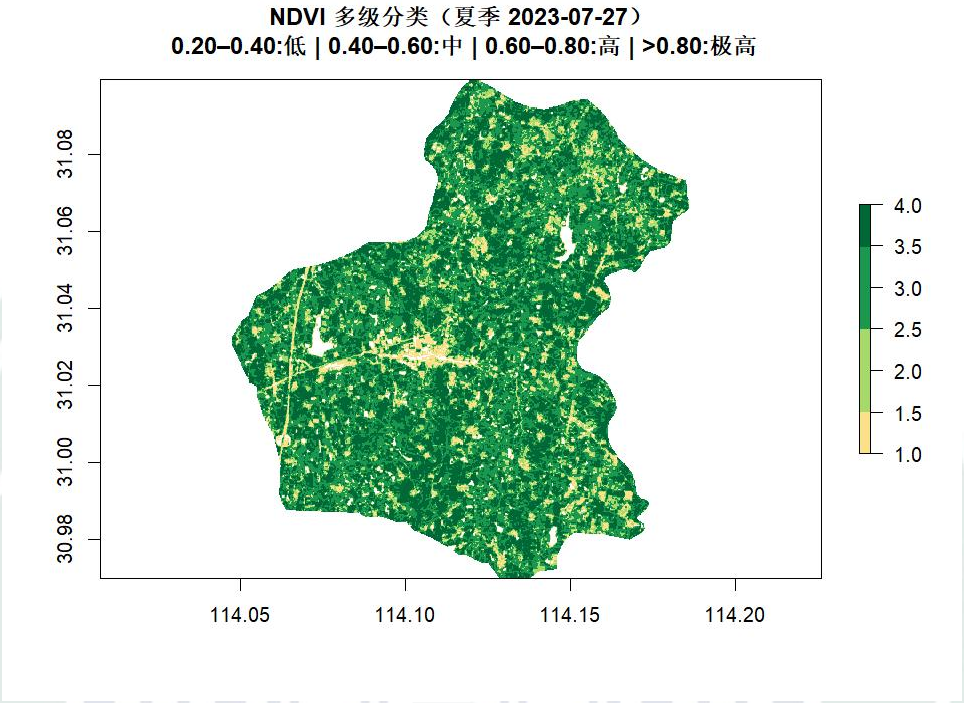


图 3.10.2-1 多级分类的结果图

如图所示，多级分级能够刻画作物熟性与冠层郁闭度差异：高与极高等级多分布于成片林地与旺长作物区，低/中等级散布于田埂、撂荒或采收期地块。对于水稻等水田作物，若处在灌溉期或生长初期，NDVI 会较林地偏低。

3.11 基于 K-means 算法的无监督分类

3.11.1 用于进行无监督分类的波段选择

本节面向整景 Landsat 8 影像进行无监督聚类，利用影像中多个光谱的特征将地表像元划分为若干内部相似、彼此差异的类别。该方法不依赖人工标注，便于在先验知识不足时快速生成场景结构草图，为后续监督分类与样本设计提供参考。

首先进行所需波段的提取：在波段选择上，使用表面反射率的 SR_B2~SR_B7（蓝、绿、红、近红外、短波红外 1/2），覆盖可见光—近红外—短波红外的主要判别信息。

```
# ---- 3 地物分类 ----
## ---- 3.1 无监督分类（K-means，整景影像） ----
```

```
## 目标：
## - 对整景“原始 img”进行无监督聚类（K-means），而不是仅对裁剪/掩膜区域
## - 自动优先选择用于做类的关键波段（优先 SR_B2~SR_B7，其次 B2~B7/Blue~SWIR2 等）
## - 标准化各波段，样本用于 K 值经验法评估，随后对整景影像聚类
## - 对聚类结果执行后处理（多数滤波），以消除碎斑
## - 输出分类 GeoTIFF、分类图、真彩图叠加与类别统计
message("3.1 提示：开始无监督遥感分类（K-means）...")
suppressPackageStartupMessages({
    library(raster)
    library(sp)
})
## 3.1.1 固定输入数据源为整景原始影像 img
## 说明：不再使用 rs_masked 或 rs_crop，确保针对整景影像进行分类
if (!exists("img") || !inherits(img, c("RasterStack","RasterBrick"))) {
    stop("3.1：未发现整景影像对象 'img'，请确保上文已成功读取 stack(tif_path)。")
}
rs_in <- img
message("3.1：已确定使用整景原始影像 'img' 作为分类输入。")
## 3.1.2 自动优先选择用于分类的关键波段（增强稳健性）
nm <- names(rs_in)
pick_by_pattern <- function(x, patterns) {
    idx_all <- integer(0)
    nms <- names(x)
    for (pt in patterns) {
        idx <- grep(pt, nms, ignore.case = TRUE)
        idx_all <- c(idx_all, idx)
    }
    idx_all <- unique(idx_all)
    if (length(idx_all) == 0) return(NULL)
    subset(x, idx_all)
}
## A) 优先：严格匹配 SR_B2~SR_B7
idx_sr <- grep("^SR_B([2-7])$", nm, ignore.case = FALSE)
if (length(idx_sr) >= 3) {
    rs_cls <- subset(rs_in, idx_sr)
    message("3.1：使用 SR_B2~SR_B7 作为特征（匹配到：", paste(names(rs_cls), collapse =
",",), "）。")
} else {
    message("3.1：未充分匹配 SR_B2~SR_B7，尝试宽松命名匹配。")
    ## B) 宽松命名集合（避开 B1/B8），意向顺序：B2..B7
    patterns_ordered <- c(
        "(^|[_-])b2($|[_-])|(^blue$)|(^blue\\b)",
        "(^|[_-])b3($|[_-])|(^green$)|(^green\\b)",
        "(^|[_-])b4($|[_-])|(^red$)|(^red\\b)",
```

```
        "(^|[_-])b5($|[_-])|(^nir$)|(^nir\\b)",
        "(^|[_-])b6($|[_-])|(^swir1$)|(^swir_?1\\b)",
        "(^|[_-])b7($|[_-])|(^swir2$)|(^swir_?2\\b)"
    )
    rs_try <- pick_by_pattern(rs_in, patterns_ordered)
    if (!is.null(rs_try) && nlayers(rs_try) >= 3) {
        rs_cls <- rs_try
        message("3.1：使用宽松匹配到的波段：", paste(names(rs_cls), collapse = ","))
    } else {
        message("3.1：宽松匹配仍不足，尝试从上文变量收集候选波段或底限选取。")
        ## C) 从上文变量收集（若用户在前文已定义 b2..b7 等）
        cand_list <- list()
        add_if_ok <- function(obj) {
            if (exists(obj) && inherits(get(obj),
c("RasterLayer","RasterBrick","RasterStack"))) {
                r <- get(obj)
                if (inherits(r, "RasterLayer")) {
                    cand_list[[obj]] <<- r
                } else {
                    for (i in 1:nlayers(r)) {
                        cand_list[[ paste0(obj, "_", names(r)[i]) ]] <<- r[[i]]
                    }
                }
            }
        }
        for (nmv in
c("b2","b3","b4","b5","b6","b7","blue","green","red","nir","swir1","swir2")) {
            add_if_ok(nmv)
        }
        # 同时尝试自 img 中按名称提取 B2..B7
        try({
            for (key in c("B2","B3","B4","B5","B6","B7")) {
                i <- grep(paste0("(^|[_-])", tolower(key), "($|[_-])"),
tolower(names(rs_in)))
                if (length(i) > 0) cand_list[[key]] <- rs_in[[i[1]]]
            }
        }, silent = TRUE)
        if (length(cand_list) >= 3) {
            rs_cls <- stack(cand_list)
            unq <- !duplicated(names(rs_cls))
            rs_cls <- subset(rs_cls, which(unq))
            if (nlayers(rs_cls) > 6) rs_cls <- rs_cls[[1:6]]
            if (nlayers(rs_cls) >= 3) {
```

```
            message("3.1：使用上文收集的候选波段：", paste(names(rs_cls), collapse =
","))
        } else {
            rs_cls <- NULL
        }
    } else {
        rs_cls <- NULL
    }
    ## D) 最后底限：自 img 中筛选数值型波段（剔除质量波段），取前 3~6 层
    if (is.null(rs_cls) || nlayers(rs_cls) < 3) {
        message("3.1：继续底限策略—从整景影像中自动筛选数值波段。")
        qa_pat <- "(qa|quality|cloud|pixel_qa|radsat|saturation|mask)"
        keep_idx <- which(!grepl(qa_pat, tolower(names(rs_in))))
        rs_tmp <- if (length(keep_idx) >= 3) subset(rs_in, keep_idx) else rs_in
        good <- logical(nlayers(rs_tmp))
        for (i in 1:nlayers(rs_tmp)) {
            v <- values(rs_tmp[[i]])
            v <- v[is.finite(v)]
            good[i] <- length(unique(v)) > 1
        }
        if (any(good)) rs_tmp <- subset(rs_tmp, which(good))
        if (nlayers(rs_tmp) >= 3) {
            take <- min(6, nlayers(rs_tmp))
            rs_cls <- rs_tmp[[1:take]]
            message("3.1：使用底限筛选的数值波段：", paste(names(rs_cls), collapse =
","))
        }
    }
}
if (is.null(rs_cls) || nlayers(rs_cls) < 3) {
    stop("3.1：未能找到足够的关键波段用于无监督分类（至少需要 3 层）。请检查输入影像波段命名。
")
}
## 3.1.3 对有效像元对齐掩膜（避免 NA 不一致）
rs_cls <- mask(rs_cls, rs_cls[[1]])
## 3.1.4 波段标准化（逐层 z-score）
scale_stack <- function(stk) {
    b <- vector("list", nlayers(stk))
    for (i in seq_len(nlayers(stk))) {
        vi <- values(stk[[i]])
        m <- mean(vi, na.rm = TRUE)
        s <- stats::sd(vi, na.rm = TRUE)
        if (!is.finite(s) || s == 0) s <- 1
```

```
        b[[i]] <- (stk[[i]] - m) / s
    }
    stack(b)
}
rs_std <- scale_stack(rs_cls)
names(rs_std) <- paste0(names(rs_cls), "_z")
```

3.11.2 K-means 算法的 K 值评估

K-means 算法的超参数 K 值具有敏感性[19]，K 过小会“合并”细类，K 过大则易过分割并引入噪声，因此 K 值的选择可以采用“肘部法”：通过观察聚类目标函数随簇数 K 的下降趋势，寻找“边际收益显著递减”的转折点，把该转折点作为合理的 K。对 K-means 而言，典型目标函数是类内平方和总和（Total Within-Cluster Sum of Squares，WSS，也称 SSE）。K 越大，WSS 必然不增（一般会下降），但下降幅度并非线性：起初增大 K 能快速吸收大规模异质性，后续继续加 K 只是在细枝末节上做“微调”，收益迅速变小。

```
## 3.1.5 抽样用于 K 值评估
set.seed(20251031)
n_total <- ncell(rs_std)
n_sample <- min(50000, max(2000, round(n_total * 0.02)))
smp <- sampleRandom(rs_std, size = n_sample, na.rm = TRUE, as.data.frame = TRUE, sp =
FALSE)
## 3.1.6 K 值快速评估
k_candidates <- 2:8
wss <- numeric(length(k_candidates))
message("3.1：评估 K 值（经验法）...")
for (i in seq_along(k_candidates)) {
    k <- k_candidates[i]
    km <- kmeans(smp, centers = k, nstart = 10, iter.max = 100)
    wss[i] <- km$tot.withinss
    message(sprintf(" - K=%d, tot.withinss=%.3e", k, wss[i]))
}
save_seq_plot({
    plot(k_candidates, wss, type = "b", pch = 19,
         xlab = "K (clusters)", ylab = "Total within-cluster sum of squares",
         main = "K-means 经验法评估（肘部法）")
}, "Kmeans_经验评估")
```

绘制的 K 值选择折线图如下：

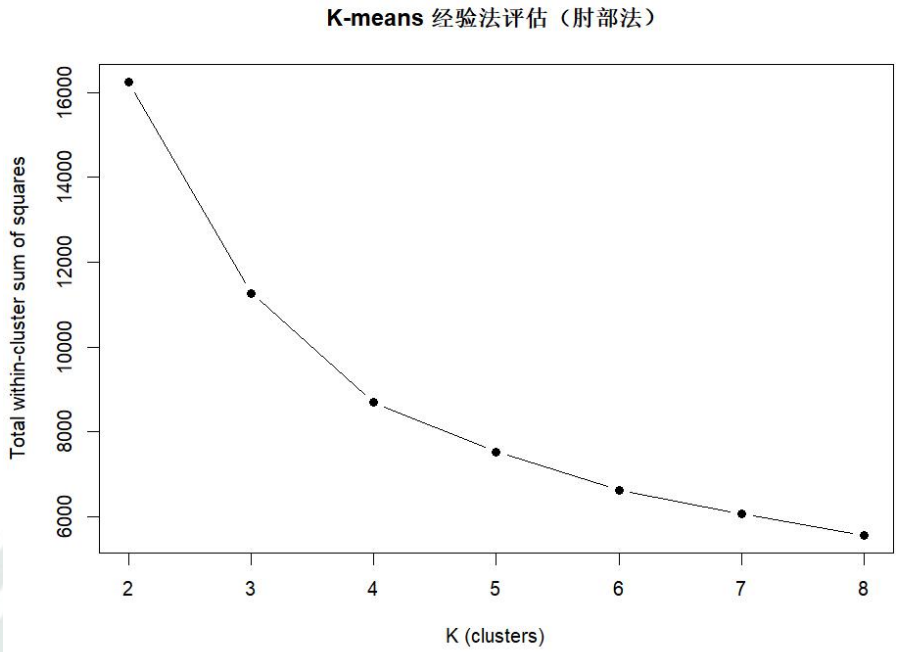


图 3.11.2-1 K值的评估折线图

由上图可知，最合适的 K 值为 4。

```
## 3.1.7 确定最终 K（根据肘部法得到的折线图选择并修改下面这一行代码）
K_final <- 4
```

3.11.3 光谱空间中基于 K-means 算法的无监督分类

K-means 聚类假设同类像元在特征空间上呈紧凑分布。算法通过迭代更新“簇中心—像元归属”的过程，最小化类内平方和（Within-Cluster Sum of Squares, WSS），从而得到 K 个谱类[20]。考虑到不同波段量纲与方差差异显著，聚类前对各波段做 z-score 标准化。

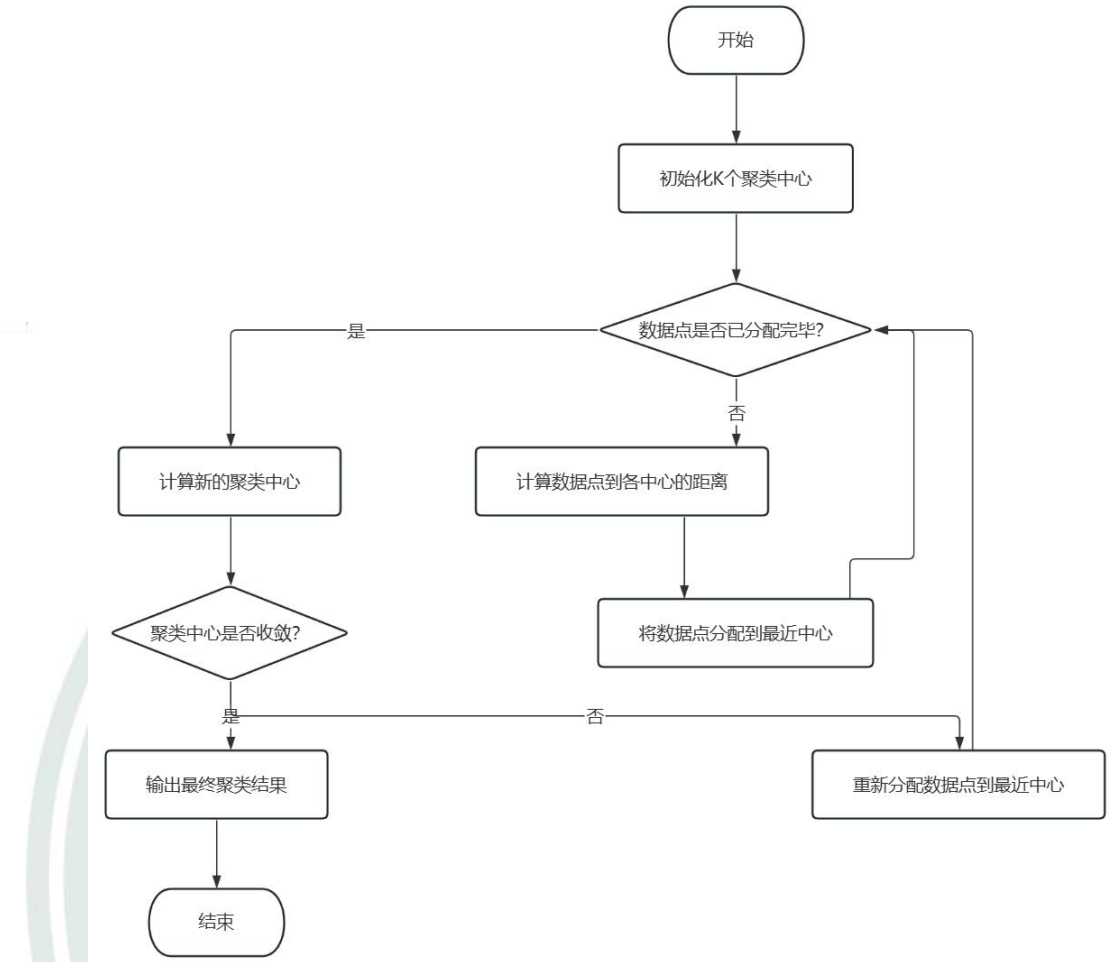


图 3.11.3-1 K-means 聚类算法的流程图

```
## 3.1.8 使用样本初始化中心
km_init <- kmeans(smp, centers = K_final, nstart = 10, iter.max = 200)
centers_init <- km_init$centers
## 3.1.9 对整景影像执行 K-means（使用标准化后的全图）
message(sprintf("3.1：对整景影像执行 K-means（K=%d）...", K_final))
vals <- getValues(rs_std)
ok <- complete.cases(vals)
cls_all <- rep(NA_integer_, nrow(vals))
if (sum(ok) == 0) stop("3.1：有效像元为 0，无法分类。")
km_full <- kmeans(vals[ok, , drop = FALSE], centers = centers_init, iter.max = 200)
cls_all[ok] <- km_full$cluster
## 3.1.10 写出分类栅格（原始结果）
cls_r_raw <- rs_std[[1]]
values(cls_r_raw) <- cls_all
levels_df <- data.frame(ID = 1:K_final,
                        Class = paste0("Class_", 1:K_final),
                        stringsAsFactors = FALSE)
# 保存原始 K-means 结果
writeRaster(cls_r_raw, next_name("Kmeans_分类_原始", "tif"), overwrite = TRUE)
```

```
if (!exists("pal_k")) {
    pal_k <-
c("#e31a1c","#33a02c","#1f78b4","#ff7f00","#6a3d9a","#a6cee3","#b2df8a","#fb9a99")[s
eq_len(K_final)]
}
# 使用 save_seq_plot 函数来保存 PNG
save_seq_plot({
    plot(cls_r_raw, col = pal_k[1:K_final], legend = FALSE,
         main = sprintf("K-means 原始分类结果（K = %d）", K_final))
    legend("topright", legend = levels_df$Class, fill = pal_k[1:K_final], bty = "n", cex
= 0.9)
}, "Kmeans_分类图_原始") # 标签名
message("3.1：已输出 Kmeans 原始分类 PNG 预览图。")
```

K-means 算法得到的分类结果图如下：

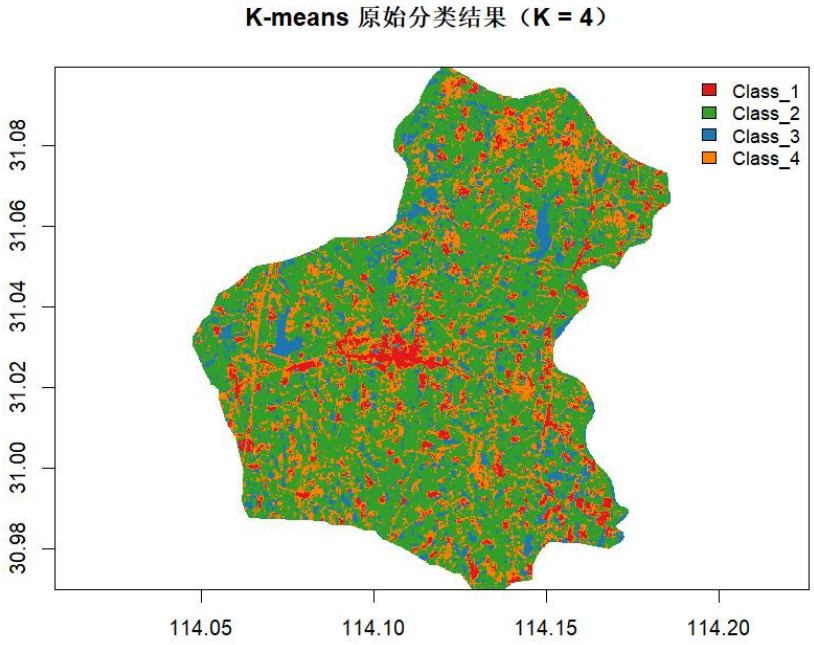


图 3.11.3-2 K-means 算法的原始分类结果图

由上图可见，Class_1 至 Class_4 这四种地物类型大致对应居民区、植被、水体、道路，且与 3.8 节、3.10 节中的相关结果能够大致符合：连片农田/林地多被归为植被类，边界清晰；镇区的人工地面聚为建设类；河流、库塘、渠系连续分布的像元归为水体类；虽然由于道路类和居民区的地表反射的光谱信息太过相近，二者的区分效果不如它们与其余地物类型的区分效果，例如部分人工地表等更适合归为居民区的区域被归为道路类，但也有不少包括 G4 京港澳告诉和 G316 国道等较为宽阔的道路也大致被成功归为道路类。

3.11.4 众数滤波后处理

3.11.3 中 K-means 算法的原始分类结果图中有很多细碎的地物斑块，但是根据地理学第一定律（又称空间自相关性定律），地理事物或属性在空间分布上互为相关，即邻近事物间的相关性高于距离较远的事物，那么真实地标的地物类型应当具有连续性。但是上述聚类算法仅仅在光谱空间进行聚类，并未考虑影像空间域的关系，因此容易出现许多细碎的地物区域，而这与前述地球表面的实际特征不相符。因此聚类后还需要进行后处理，对分类结果进行 3×3 窗口的多数（众数）滤波，抑制“椒盐”碎斑，提升地物类型的空间一致性。

```
## 3.1.11 K-means 后处理（多数/众数滤波）
message("3.1：K-means 原始结果已生成，开始执行后处理（3x3 多数/众数滤波）...")
# 定义 3x3 邻域窗口
w <- matrix(1, nrow=3, ncol=3)
# 使用 focal() 和 modal()（众数）函数进行滤波
# modal 是 raster 包中用于 focal 的内置函数
# na.rm = TRUE 保证邻域内的 NA 值被忽略
# pad = FALSE 避免在图像边缘填充（结果边缘会是 NA）
cls_r_post <- focal(cls_r_raw, w = w, fun = modal, na.rm = TRUE, pad = FALSE)
# 恢复掩膜：
# 确保后处理后的数据覆盖范围与原始一致，避免滤波导致数据区域“增长”到 NA 区域。
# 同时也保证了原始的 NA 边界不会产生“缝隙”。
cls_r_post <- mask(cls_r_post, cls_r_raw)
message("3.1：后处理完成。")
```

得到的后处理后的结果图如下：

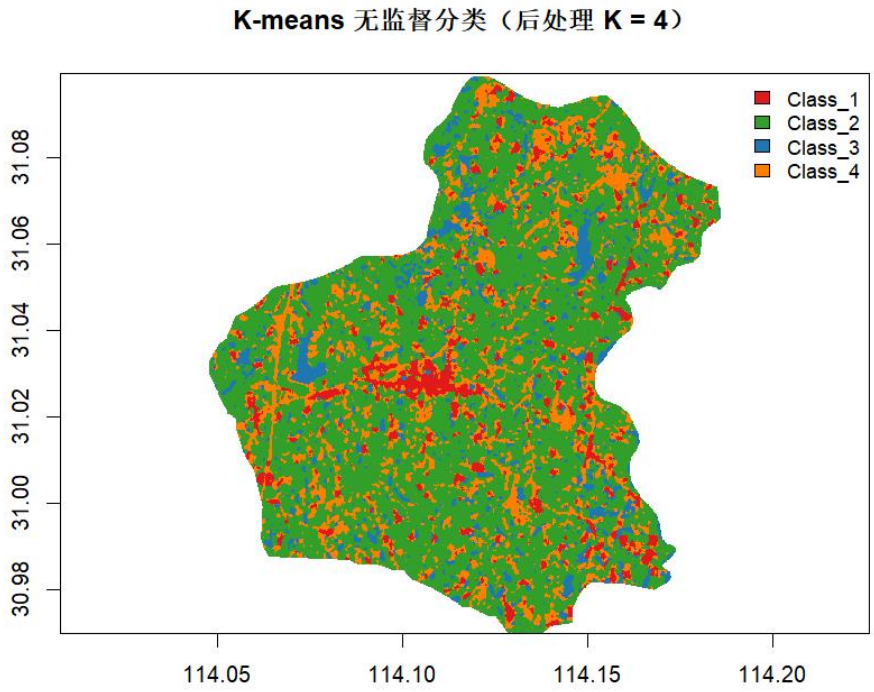


图 3.11.4-1 K-means 算法的分类结果的后处理图

由图可以看出后处理的效果，3×3 众数滤波有效消除了零星杂点，提高了地块内部一致性，例如使得部分道路分类的连通性更好，使输出图像的视觉解释性更好。

3.12 基于随机森林算法的监督分类

本节利用人工标注的训练样区，通过随机森林（Random Forest, RF）构建地物分类模型，并在整景影像上进行像元级预测。相比无监督聚类（3.11 节），监督分类能将光谱相似但语义不同的目标分开。

3.12.1 用于随机森林算法的预测模型的训练的样区数据的构建

如图 3.12.1-1 所示，首先使用 GeoScene Pro 3.1 软件绘制建筑（图中红色）、植被（图中绿色）、水体（图中蓝色）的样区矢量边界。由于道路与居民区光谱差异过小，基于光谱信息进行地物分类较为困难，所以在本节不作为一个单独的地物类别，可以视作与居民区在同一类。

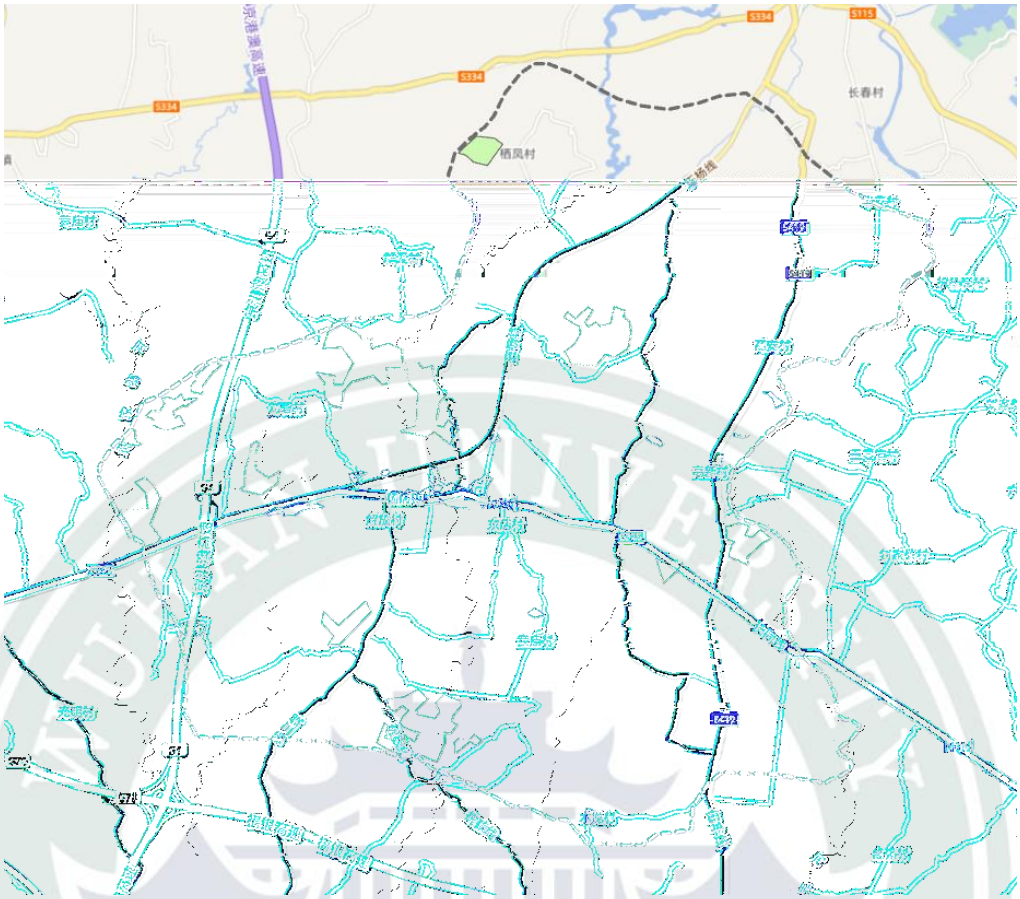


图 3.12.1-1 使用 GeoScene Pro 3.1 软件绘制建筑、植被、水体的样区矢量边界

将绘制的样区矢量导出到 data/supervision_classification 目录下，每个子文件夹对应一个类别，每个子文件夹中包含一个 Shapefile（.shp），代码根据这些矢量文件对影像栅格进行裁剪后，将作为随机森林模型的训练数据。栅格特征栈使用前文分类准备的 rs_cls（SR_B2~SR_B7 波段）。

```
## ---- 3.2 监督分类 ----
message("3.2 节：开始监督分类（基于 data/supervision_classification 中的 shp 训练样区）...")
## 1. 选择特征栅格
rs_train_base <- NULL
if (exists("img")) {
    rs_train_base <- img
    message("使用 img 作为底图。")
} else stop("未找到可用栅格。")
if (exists("rs_cls") && inherits(rs_cls, c("RasterStack","RasterBrick"))) {
    rs_feat <- rs_cls
    message("3.2：使用 rs_cls 作为特征栈。")
} else {
    rs_feat <- rs_train_base
}
## 2. 训练样区路径
```

```
train_dir <- "data/supervision_classification"
if (!dir.exists(train_dir)) stop("未找到训练矢量文件夹
data/supervision_classification")
subdirs <- list.dirs(train_dir, full.names = TRUE, recursive = FALSE)
shp_files <- list()
for (sd in subdirs) {
    shp_candidates <- list.files(sd, pattern = "\\.shp$", full.names = TRUE, ignore.case
= TRUE)
    if (length(shp_candidates) > 0) shp_files[[basename(sd)]] <- shp_candidates[1]
}
if (length(shp_files) == 0) stop("未找到任何 .shp 文件。")
## 3. 提取训练样本
training_list <- list()
for (cls_name in names(shp_files)) {
    shp_path <- shp_files[[cls_name]]
    message("读取类别：", cls_name, " --> ", basename(shp_path))
    # 使用 sf 读取（更鲁棒）
    shp_sf <- try(sf::st_read(shp_path, quiet = TRUE), silent = TRUE)
    if (inherits(shp_sf, "try-error") || nrow(shp_sf) == 0) {
        warning("跳过空或错误文件：", shp_path)
        next
    }
    # 若 CRS 不同则重投影
    if (!is.na(st_crs(shp_sf)) && !is.na(crs(rs_feat))) {
        shp_sf <- st_transform(shp_sf, crs = st_crs(rs_feat))
    }
    # 转为 Spatial 以兼容 raster::extract
    shp_sp <- as(shp_sf, "Spatial")
    # 确保几何合法
    if (is.null(shp_sp@polygons) && is.null(shp_sp@coords)) {
        warning("矢量无几何数据：", shp_path)
        next
    }
    ex <- try(raster::extract(rs_feat, shp_sp), silent = TRUE)
    if (inherits(ex, "try-error") || is.null(ex)) {
        warning("提取失败：", shp_path)
        next
    }
    if (is.list(ex)) ex <- do.call(rbind, ex)
    if (is.null(ex) || nrow(as.data.frame(ex)) == 0) {
        warning("类别 ", cls_name, " 未提取到任何像元（可能未与影像重叠）。")
        next
    }
    message("类别 ", cls_name, " 提取像元：", nrow(ex))
```

```
    if (is.list(ex)) ex <- do.call(rbind, ex)
    ex <- as.data.frame(ex)
    ex <- ex[complete.cases(ex), ]
    if (nrow(ex) == 0) next
    ex$class <- cls_name
    training_list[[cls_name]] <- ex
    message(" -> 提取像元：", nrow(ex))
}
if (length(training_list) == 0) stop("未能提取任何样本，请检查矢量。")
## 汇总每个类别像元数
sample_summary <- data.frame(
    类别 = names(training_list),
    像元数 = sapply(training_list, nrow)
)
message("各类别样本统计：")
print(sample_summary)
## 合并训练数据
training_df <- do.call(rbind, training_list)
message("训练样本总计：", nrow(training_df))
```

各地物类别的样区像元个数统计如下：

地物类别	样区像元个数
building	1135
vegetation	5945
water	325

表 3.12.1-1 各地物类别的样区像元个数统计

3.12.2 使用随机森林算法对样区进行学习与决策

分类器使用随机森林（ntree=500，输出变量重要性），验证方案为训练集/验证集 7:3 随机划分，训练完成后报告 OOB 错误率、混淆矩阵与验证集总体精度。

随机森林算法的主要过程是：先从带标签的数据中反复进行自助法抽样（对样本做有放回的随机抽取），为每一次抽样训练一棵决策树，并在每个树的节点分裂时再随机抽取一小部分特征候选来选择最优划分，这样既引入了样本层面的随机性（bootstrap），也引入了特征层面的随机性（feature subsampling），从而让各棵树彼此去相关[21]。训练完成后，森林中的每棵树都学到了从光谱/指数等特征到类别标签（如植被、水体、建成区）的映射关系。在预测阶段，未被模型训练过的像元的特征会被送入所有树，各树各自给出一个类别投票，最终以多数表决（分类任务）作为像元的预测类别。由于集成了大量弱相关的树，随机森林能在不需复杂参数调优的前提下获得稳健的泛化能力，并通过袋外样本（未被某棵树抽到的样本）自然估计泛化误差，同时还能提供特征重要性度量，帮助解释哪些波段或指数对分类贡献最大。

代码中还设计输出了精度评价的指标。监督分类的 OOB 错误率与验证集总体精度能够提供模型泛化能力的量化指标，若二者差距过大，可能存在过拟合或样本分布漂移；混淆矩阵能够定位主要混淆对，可用于在样本再平衡与特征扩展时优先优化这些类别。

```
# 4. 随机森林分类
set.seed(20251101)
idx <- sample(nrow(training_df))
train_df <- training_df[idx, ]
n_train <- round(0.7 * nrow(train_df))
train_x <- train_df[1:n_train, setdiff(names(train_df), "class")]
train_y <- factor(train_df[1:n_train, "class"])
test_x <- train_df[(n_train+1):nrow(train_df), setdiff(names(train_df), "class")]
test_y <- factor(train_df[(n_train+1):nrow(train_df), "class"])
rf_model <- randomForest(x = train_x, y = train_y, ntree = 500, importance = TRUE)
message("OOB 错误率：", round(100 * tail(rf_model$err.rate[, "OOB"], 1), 2), "%")
print(rf_model$confusion)
pred_test <- predict(rf_model, test_x)
acc <- sum(pred_test == test_y) / length(test_y)
message(sprintf("验证集总体精度：%.2f%%", 100 * acc))
## 5. 整图预测
pred_tif <- next_name("Supervised_RF_分类_原始", "tif")
pred_raster <- raster::predict(rs_feat, rf_model, progress = "text", filename = pred_tif,
                               overwrite = TRUE, type = "response")
message("输出原始分类栅格：", pred_tif)
## 6. 颜色映射
classes <- sort(unique(training_df$class))
pred_raster <- as.factor(pred_raster)
levels(pred_raster)[[1]] <- data.frame(ID = seq_along(classes), class = classes)
pal_sup <- setNames(colorRampPalette(
    c("#e31a1c","#33a02c","#1f78b4"))(length(classes)), classes)
save_seq_plot({
    plot(pred_raster, col = pal_sup, main = sprintf("监督分类结果（RF 原始，%d 类）",
length(classes)),
         axes = FALSE, box = FALSE, legend = FALSE)
    legend("topleft", legend = classes, fill = pal_sup, bty = "n", cex = 0.9)
```

```
}, "监督分类_RF_原始")
```

监督分类的结果图如下：

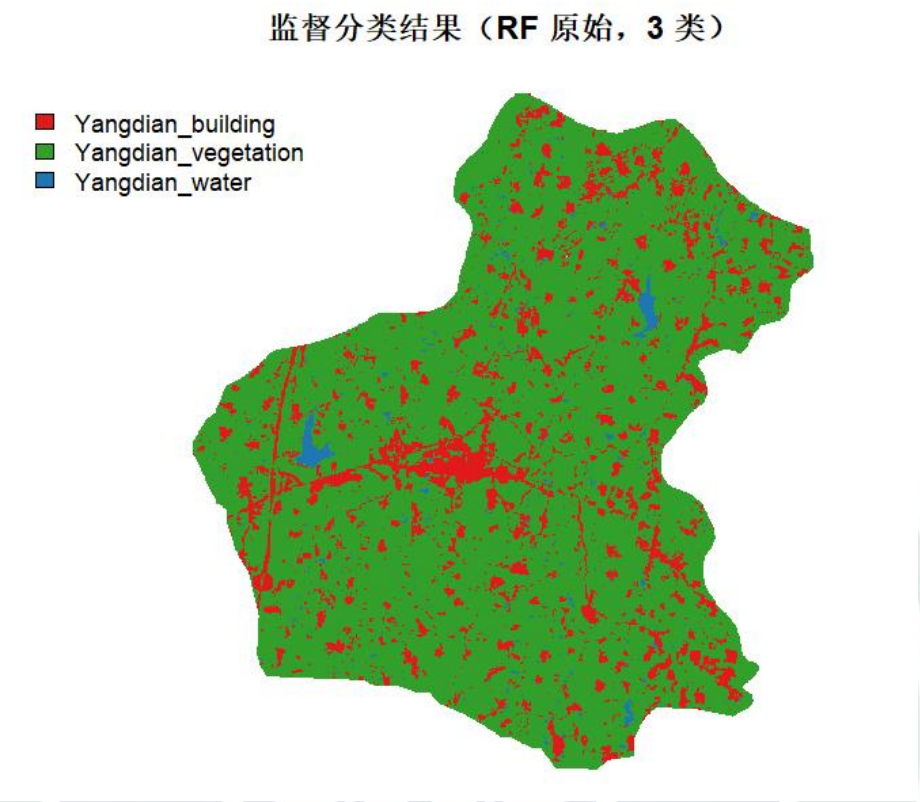


图 3.12.2-1 监督分类的原始结果图

由于有明确的样区来监督，监督分类的结果图比无监督分类的结果（图 3.11.3-2）的语义更清晰，例如对水域的分类更符合真实情况，避免了许多假阳性结果的出现。

输出的精度指标如下：

```
> message("OOB 错误率：", round(100 * tail(rf_model$err.rate[, "OOB"], 1), 2), "%")
OOB 错误率：0.39%
> print(rf_model$confusion)
                  Yangdian_building Yangdian_vegetation Yangdian_water class.error
Yangdian_building               788                   6              0 0.007556675
Yangdian_vegetation               6                4155              6 0.002879770
Yangdian_water                    0                   2            221 0.008968610
> pred_test <- predict(rf_model, test_x)
> acc <- sum(pred_test == test_y) / length(test_y)
> message(sprintf("验证集总体精度：%.2f%%", 100 * acc))
验证集总体精度：99.55%
```

OOB 错误率 0.39%，表示随机森林对每棵树未被抽中的样本（袋外样本）做即时评估得到的内置泛化误差估计极低，说明特征对目标类别区分度很高，模型对训练数据的泛化良好；混淆矩阵（训练集的 OOB/内部统计口径）中，building 正确 788，误分到 vegetation 6，错误率 0.76%；vegetation 正确 4155，误分到 building 6、water 6，错误率 0.29%；water 正确 221，

误分到 vegetation 2，错误率 0.90%，极少量混淆可能出现在低矮植被/裸土地与建设用地交界、村镇边缘耕地或庭院绿化等由于影像的空间分辨率不够高而产生的同一像元内不同地物的光谱信息的混淆，但总体来说各类精度都极高；将样本按 7:3 划分出的独立验证子集上得到的总体正确率为 99.55%，与 OOB 一致性高（0.45 个百分点内），提示数据划分与 OOB 评估相互印证，过拟合风险较低。

3.12.3 众数滤波后处理

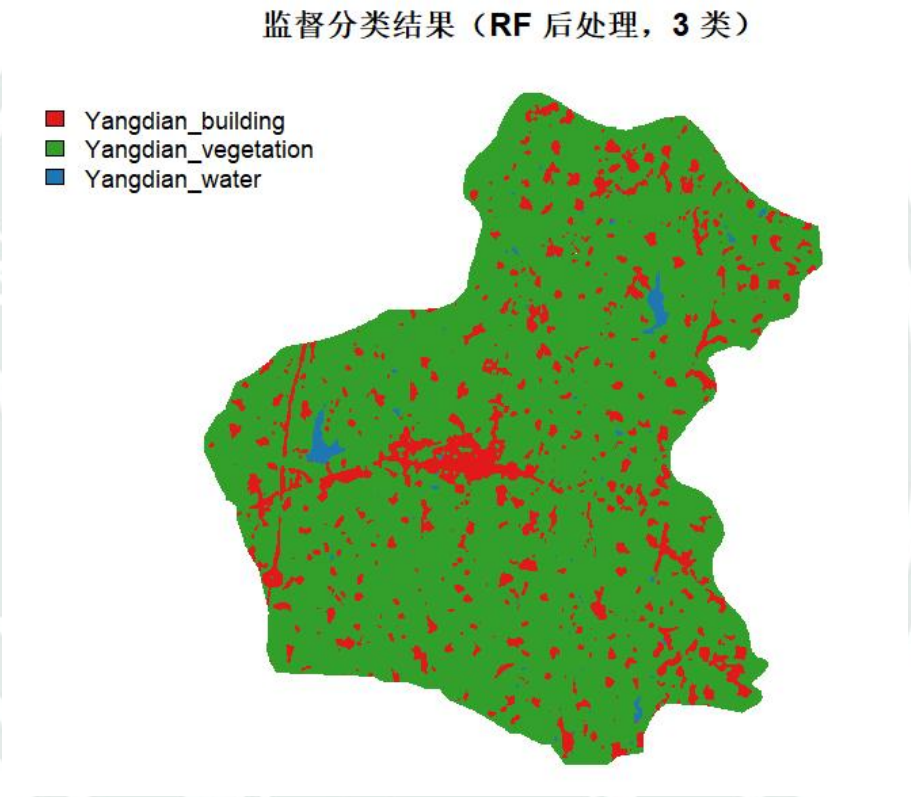


图 3.12.2-2 监督分类后处理后的结果图

与 3.11.4 节进行后处理的原因类似，后处理图的碎斑显著减少，居民区、植被、水体的分类结果在空间上更连贯，更符合地表的实际情况。

3.13 基于遥感指数的变化监测

为刻画杨店镇近十年间植被、水体与建成区的长期变化趋势，本节基于 Landsat 8 多时相影像构建 NDVI/NDWI/NDBI/SAVI 的时间序列，按像元级进行线性趋势拟合，并输出首末期差值、相对变化率、趋势斜率空间分布、斜率直方图与四分位变化类别图，将宏观趋势与局地细节同时呈现出来。

由于对时序数据的分析对算力有一定要求，当计算密集时 R 语言的运行不是很稳定，因此本节单独写了一个脚本来运行，本节的代码也有很多计算优化的设计。

3.13.1 抽样构建时序影像数据

在目录 GEE_L8_Yangdian_image 中搜索所有 GeoTIFF，按文件名中的 YYYYMMDD 提取日期并排序，在 2015–2023 年间每年各选一张冬季（离 1 月 15 日最近）与一张夏季（与 7 月 1 日最近）的影像，形成代表时相序列，统计输出被选中的文件名与对应日期，便于审阅与复现。

```
# ---- 4 变化检测 ----
## ---- 4.1 基于遥感指数的变化检测 ----
## 1. 输出目录准备
out_dir <- "res_change_detection"
if (!dir.exists(out_dir)) {
  message(sprintf("创建结果目录：%s", out_dir))
  dir.create(out_dir, recursive = TRUE)
} else {
  message(sprintf("结果目录 '%s' 已存在，将尝试跳过已存在文件的计算。", out_dir))
}
message(sprintf("结果输出目录已准备好：%s", normalizePath(out_dir)))
## 2. 辅助函数
message_line <- function(...) cat(paste0(format(Sys.time(), "%H:%M:%S "), ... , "\n"))
next_name <- function(prefix = "Plot", ext = "png", dir = out_dir) {
  i <- 1
  repeat {
    fn <- file.path(dir, sprintf("%s_%03d.%s", prefix, i, ext))
    if (!file.exists(fn)) return(fn)
    i <- i + 1
  }
}
save_seq_plot <- function(expr, prefix = "Plot", width = 10, height = 8, res = 200) {
  fn <- next_name(prefix, "png")
  png(fn, width = width, height = height, units = "in", res = res)
  on.exit(dev.off())
  force(expr)
  message_line("保存图像: ", fn)
}
extract_date_from_name <- function(fn){
  b <- basename(fn)
  m <- regexpr("\\d{8}", b)
  if (m[1] == -1) return(NA)
  ds <- regmatches(b, m)
```

```
  as.Date(ds, "%Y%m%d")
}
safe_pick <- function(x, key){
  nm <- tolower(names(x))
  idx <- grep(key, nm, ignore.case = TRUE)
  if (length(idx) == 0 && key == "B6") {
    idx <- grep("swir1", nm, ignore.case = TRUE)
  }
  if (length(idx) > 0) return(x[[idx[1]]])
  stop(sprintf("找不到波段 %s", key))
}
## 3. 输入文件（查找所有 tif）
img_dir <- "GEE_L8_Yangdian_image"
tif_files_all <- list.files(img_dir, pattern = "\\.tif$", full.names = TRUE, ignore.case
= TRUE)
if (length(tif_files_all) == 0) stop("未找到任何 tif 文件，请检查 GEE_L8_Yangdian_image
文件夹。")
dates_all <- sapply(tif_files_all, extract_date_from_name)
valid_idx <- which(!is.na(dates_all))
if (length(valid_idx) == 0) stop("未能从任何文件名中提取到日期（YYYYMMDD）。")
tif_files_all <- tif_files_all[valid_idx]
dates_all <- as.Date(dates_all[valid_idx])
ord_all <- order(dates_all)
tif_files_all <- tif_files_all[ord_all]
dates_all <- dates_all[ord_all]
message_line(sprintf("发现 %d 份可解析日期的影像（总计），时间范围：%s → %s",
                     length(tif_files_all), min(dates_all), max(dates_all)))
## 4. 在每年（2015-2023）选择夏季与冬季代表影像
years <- 2015:2023
selected_idx <- integer(0)
for (y in years) {
  target_summer <- as.Date(sprintf("%d-07-01", y))
  target_winter <- as.Date(sprintf("%d-01-15", y))

  diffs_summer <- abs(as.numeric(dates_all - target_summer))
  diffs_winter <- abs(as.numeric(dates_all - target_winter))

  pick_s <- which.min(diffs_summer)
  pick_w <- which.min(diffs_winter)

  selected_idx <- unique(c(selected_idx, pick_s, pick_w))
}
if (length(selected_idx) < 2) stop("选取到的代表影像不足 2 张，无法进行变化检测。请检查原
始影像或年份范围设置。")
```

```
tif_files <- tif_files_all[selected_idx]
dates <- dates_all[selected_idx]
ord <- order(dates)
tif_files <- tif_files[ord]
dates <- dates[ord]
date_strs <- format(dates, "%Y%m%d")
message_line(sprintf("选定 %d 张代表影像用于变化检测，时间范围：%s → %s",
                     length(tif_files), min(dates), max(dates)))
message_line("选定的文件（按时间排序）：")
for (i in seq_along(tif_files)) message_line(sprintf("  %s  (%s)",
basename(tif_files[i]), date_strs[i]))
```

3.13.2 遥感指数的计算

对每景栅格按需提取 B3（绿）、B4（红）、B5（近红外）、B6（SWIR1），计算：

$$NDVI = \frac{NIR - Red}{NIR + Red}$$

$$NDWI = \frac{Green - NIR}{Green + NIR}$$

$$NDBI = \frac{SWIR1 - NIR}{SWIR1 + NIR}$$

$$SAVI = 1.5 \times \frac{NIR - Red}{NIR + Red + 0.5}$$

结果按日期命名写入 res_change_detection/ 作为 .tif 文件，后续运行该代码时会直接加载以节省计算时间。

```
## 5. 指数计算函数
compute_indices_from_stack <- function(rs_or_path){
    if (is.character(rs_or_path)) {
        s <- try(stack(rs_or_path), silent = TRUE)
        if (inherits(s, "try-error")) stop("无法读取: ", rs_or_path)
    } else {
        s <- rs_or_path
    }
    red <- safe_pick(s, "B4")
    nir <- safe_pick(s, "B5")
    g   <- safe_pick(s, "B3")

    sw1 <- tryCatch(safe_pick(s, "SR_B6"), error = function(e) NULL)

    ndvi <- (nir - red) / (nir + red)
    ndwi <- (g - nir) / (g + nir)
    if (!is.null(sw1)) {
        ndbi <- (sw1 - nir) / (sw1 + nir)
```

```
    } else {
        ndbi <- nir; ndbi[] <- NA
        warning("未找到 SWIR1 波段，NDBI 将被设置为 NA（全 NA 图层）。")
    }
    savi <- (1.5 * (nir - red)) / (nir + red + 0.5)

    list(NDVI = ndvi, NDWI = ndwi, NDBI = ndbi, SAVI = savi)
}
## 6. 批量计算或加载 指数影像
# 检查所有期望的输出文件是否存在
all_indices_exist <- TRUE
expected_files_list <- list() # 用来存储期望的文件名
indices_to_check <- c("NDVI", "NDWI", "NDBI", "SAVI")
# 构建期望文件路径列表
for (ds in date_strs) {
    for (idx_name in indices_to_check) {
        fn <- file.path(out_dir, sprintf("%s_%s.tif", idx_name, ds))
        expected_files_list[[idx_name]][[ds]] <- fn
        # 检查文件是否存在
        if (!file.exists(fn)) {
            all_indices_exist <- FALSE
        }
    }
}
# 初始化列表
ndvi_list <- list(); ndwi_list <- list(); ndbi_list <- list(); savi_list <-
list()
if (all_indices_exist) {
    message_line("检测到所有时相的指数 TIF 文件均已存在，跳过计算，开始加载...")

    for (ds in date_strs) {
        tryCatch({
            # 从 expected_files_list 加载
            ndvi_list[[ds]] <- raster(expected_files_list[["NDVI"]][[ds]])
            ndwi_list[[ds]] <- raster(expected_files_list[["NDWI"]][[ds]])
            ndbi_list[[ds]] <- raster(expected_files_list[["NDBI"]][[ds]])
            savi_list[[ds]] <- raster(expected_files_list[["SAVI"]][[ds]])
            message_line(sprintf("  已加载 %s 的指数。", ds))
        }, error = function(e) {
            stop(sprintf("加载 %s 的 TIF 文件时出错: %s。请删除
res_change_detection 目录后重试。", ds, e$message))
        })
    }
    message_line("所有指数 TIF 文件加载完毕。")
```

```
} else {
    message_line("部分或全部指数 TIF 文件缺失，开始批量计算...")

    for (i in seq_along(tif_files)) {
        tf <- tif_files[i]; ds <- date_strs[i]
        message_line(sprintf("处理 %s (%s) ...", basename(tf), ds))
        idxs <- compute_indices_from_stack(tf)

        # 使用 expected_files_list 中的路径写入文件
        writeRaster(idxs$NDVI, expected_files_list[["NDVI"]][[ds]], overwrite = TRUE)
        writeRaster(idxs$NDWI, expected_files_list[["NDWI"]][[ds]], overwrite = TRUE)
        writeRaster(idxs$NDBI, expected_files_list[["NDBI"]][[ds]], overwrite = TRUE)
        writeRaster(idxs$SAVI, expected_files_list[["SAVI"]][[ds]], overwrite = TRUE)

        # 存入列表
        ndvi_list[[ds]] <- idxs$NDVI
        ndwi_list[[ds]] <- idxs$NDWI
        ndbi_list[[ds]] <- idxs$NDBI
        savi_list[[ds]] <- idxs$SAVI
    }
    message_line("指数 TIF 文件计算并保存完毕。")
}
```

3.13.3 分析变化趋势

将前面计算好的各期指数影像（按日期字符串命名）堆叠成时间序列 RasterStack，将日期转换为从首个日期起算的相对年数，便于对同一像元在不同日期的值进行时间运算。这样做是因为：线性趋势的斜率单位变为“每年变化量”（1/yr），更易解读与跨区比较，且能够使数值尺度稳定，避免用天数为单位造成斜率量级过小。

然后进行像素级趋势计算，为了减轻计算压力，使用闭式解，且将结果分块写盘。分别对四个指数分别计算趋势，每个输出的 trend 栅格都是三波段的 GTiff。以时间变量 t 为“相对年数”（从首张影像算起），对每个像元的时间序列 v 计算：

$$slope = \frac{cov(t, x)}{var(t)}$$

$$R^2 = \frac{[cov(t,x)]^2}{var(t) \times var(x)}$$

$$mean = mean(x)$$

同时，还计算首末时相差值与百分比变化：取序列首/末图层，计算首末两期的绝对变化量 $diff = 末 - 首$ 与相对变化率 $pctchg = \frac{末-首}{|首|+10^{-6}}$（$10^{-6}$ 是防止除以 0 的稳定项）。diff 能够直观展示“净增/净减”的大小，pctchg 更适合低基线像元的相对幅度的刻画。

```
# 7. 构建时间栈
ndvi_stack <- stack(ndvi_list); names(ndvi_stack) <- date_strs
ndwi_stack <- stack(ndwi_list); names(ndwi_stack) <- date_strs
ndbi_stack <- stack(ndbi_list); names(ndbi_stack) <- date_strs
savi_stack <- stack(savi_list); names(savi_stack) <- date_strs
# 将时间转换为以第一个选定日期为基准的年数（便于线性拟合）
time_years <- as.numeric(dates - min(dates)) / 365.25
# 8. 像素级趋势计算（使用闭式公式，避免 lm；分块写入磁盘）
# 对每个像元 v（向量），我们在有有效值的位置上用以下闭式公式计算：
# slope = cov(t,x) / var(t)
# R2 = (cov(t,x)^2) / (var(t) * var(x))
# mean = mean(x)
# calc 函数会把每个像元的时间序列作为向量传入函数。
# 在开始计算前，调整 raster 的临时目录与内存选项（降低内存压力）
rasterOptions(tmpdir = out_dir,   # 临时文件写到输出文件夹（避免系统 /tmp 空间不足）
              progress = "text",
              chunksize = 1e+07,  # 可调整；默认为 5e+05 等。这里稍大以减少 IO 次数，但不至于太大。
              maxmemory = 1e+08)   # 根据机器内存可适当调小
# 闭式计算函数（传入一个像元的时间序列向量 v）
pixel_trend_closedform <- function(v) {
  # v: 长度 = nlayers(stk)，包含 NA 或数值
  ok <- is.finite(v)
  n <- sum(ok)
  if (n == 0) return(c(NA_real_, NA_real_, NA_real_))
  if (n < 2) return(c(NA_real_, NA_real_, mean(v[ok], na.rm = TRUE)))
  t <- time_years[ok]
  x <- v[ok]

  # sums
  sum_t <- sum(t)
  sum_x <- sum(x)
  sum_tx <- sum(t * x)
  sum_t2 <- sum(t * t)
  sum_x2 <- sum(x * x)
```

```
  # means
  mean_t <- sum_t / n
  mean_x <- sum_x / n

  # denominator for slope (n * var(t))
  denom_t <- (sum_t2 - n * mean_t * mean_t)  # = n * var(t)

  # compute slope
  num_cov <- (sum_tx - n * mean_t * mean_x)  # = n * cov(t,x)
  if (abs(denom_t) < 1e-12) {
    slope <- NA_real_
  } else {
    slope <- num_cov / denom_t
  }

  # compute R^2 using closed form:
  denom_x <- (sum_x2 - n * mean_x * mean_x)  # = n * var(x)
  denom_r <- denom_t * denom_x
  if (denom_r <= 0) {
    r2 <- NA_real_
  } else {
    r2 <- (num_cov * num_cov) / denom_r
    # numeric guard
    if (!is.finite(r2)) r2 <- NA_real_
  }

  c(slope, r2, mean_x)
}
# 为了确保 calc 输出的三个波段名合适，写个包装器：
calc_trend_safe <- function(stk, name, out_dir, filename = NULL) {
  message_line(paste("计算", name, "趋势（闭式解 + 分块写磁盘）..."))
  if (is.null(filename)) filename <- file.path(out_dir, paste0(name, "_trend.tif"))
  # 使用 raster::calc，fun 返回长度为 3 的 numeric 向量 -> 会生成 3 个波段
  # 指定 datatype 为 FLT4S（32-bit float），以节约磁盘与内存
  tr <- calc(stk, fun = pixel_trend_closedform, filename = filename,
             format = "GTiff", overwrite = TRUE, datatype = "FLT4S", progress = "text")
  names(tr) <- c(paste0(name, "_slope"), paste0(name, "_R2"), paste0(name, "_mean"))
  message_line("写出：", filename)
  return(tr)
}
# 计算（对每个指数）
ndvi_trend <- calc_trend_safe(ndvi_stack, "NDVI", out_dir)
ndwi_trend <- calc_trend_safe(ndwi_stack, "NDWI", out_dir)
```

```
ndbi_trend <- calc_trend_safe(ndbi_stack, "NDBI", out_dir)
savi_trend <- calc_trend_safe(savi_stack, "SAVI", out_dir)
# 若启用了并行，请在结束后关闭集群
# endCluster()
## 9. 差值与百分比变化（首末时相）：保持原来实现，但在写出时指定数据类型
first_idx <- 1; last_idx <- nlayers(ndvi_stack)
diff_fun <- function(stk){
  stk[[last_idx]] - stk[[first_idx]]
}
pct_fun <- function(stk){
  overlay(stk[[first_idx]], stk[[last_idx]], fun = function(a,b){
    ifelse(is.na(a)|is.na(b), NA, (b - a)/(abs(a)+1e-6))
  })
}
indices <- list(NDVI=ndvi_stack, NDWI=ndwi_stack, NDBI=ndbi_stack, SAVI=savi_stack)
for (nm in names(indices)) {
  d <- diff_fun(indices[[nm]])
  p <- pct_fun(indices[[nm]])
  writeRaster(d, file.path(out_dir, paste0(nm, "_diff.tif")), overwrite = TRUE, datatype
= "FLT4S")
  writeRaster(p, file.path(out_dir, paste0(nm, "_pctchg.tif")), overwrite = TRUE,
datatype = "FLT4S")
}
```

3.13.4 可视化结果输出

可视化输出以下三类结果图：

（1）时间序列图：按时相计算全图均值，绘制 NDVI/NDWI/NDBI/SAVI 的 mean 曲线；

（2）斜率空间分布图与直方图：斜率图直观显示空间格局，直方图展示总体分布；

（3）变化类别图：以斜率四分位数分三类：① NDVI 与 SAVI：红=下降，白=稳定，绿=上升（植被语义直观配色）；② NDWI：红=下降，白=稳定，蓝=上升（水体语义直观配色）；③ NDBI：绿=下降，白=稳定，红=上升（建设用地语义直观配色）。

```
## 10. 可视化（时间序列与趋势图）
mean_over_time <- function(stk){
  sapply(1:nlayers(stk), function(i) {
    cellStats(stk[[i]], stat = "mean", na.rm = TRUE)
  })
}
ndvi_means <- mean_over_time(ndvi_stack)
ndwi_means <- mean_over_time(ndwi_stack)
ndbi_means <- mean_over_time(ndbi_stack)
savi_means <- mean_over_time(savi_stack)
```

```
### 10.1 时间序列图
message_line("生成 NDVI 时间序列图...")
save_seq_plot({
  op <- par(mar = c(5, 4.5, 3, 1)) # 调整单图的边距
  on.exit(par(op), add=TRUE)
  plot(dates, ndvi_means, type="b", pch=19, main="NDVI 均值变化",
       xlab="", ylab="Mean NDVI", xaxt="n")
  # 添加自定义 X 轴, las=2 使标签垂直, cex.axis 缩小字体
  axis.Date(1, at = dates, format = "%Y-%m", las = 2, cex.axis = 0.7)
}, "ChangeDetection_TimeSeries_NDVI")
message_line("生成 NDWI 时间序列图...")
save_seq_plot({
  op <- par(mar = c(5, 4.5, 3, 1))
  on.exit(par(op), add=TRUE)
  plot(dates, ndwi_means, type="b", pch=19, main="NDWI 均值变化",
       xlab="", ylab="Mean NDWI", xaxt="n")
  axis.Date(1, at = dates, format = "%Y-%m", las = 2, cex.axis = 0.7)
}, "ChangeDetection_TimeSeries_NDWI")
message_line("生成 NDBI 时间序列图...")
save_seq_plot({
  op <- par(mar = c(5, 4.5, 3, 1))
  on.exit(par(op), add=TRUE)
  plot(dates, ndbi_means, type="b", pch=19, main="NDBI 均值变化",
       xlab="", ylab="Mean NDBI", xaxt="n")
  axis.Date(1, at = dates, format = "%Y-%m", las = 2, cex.axis = 0.7)
}, "ChangeDetection_TimeSeries_NDBI")
message_line("生成 SAVI 时间序列图...")
save_seq_plot({
  op <- par(mar = c(5, 4.5, 3, 1))
  on.exit(par(op), add=TRUE)
  plot(dates, savi_means, type="b", pch=19, main="SAVI 均值变化",
       xlab="", ylab="Mean SAVI", xaxt="n")
  axis.Date(1, at = dates, format = "%Y-%m", las = 2, cex.axis = 0.7)
}, "ChangeDetection_TimeSeries_SAVI")
### 10.2 各指数斜率空间分布图（分图输出 - 保持不变）
# 定义调色板
pal_veg <- brewer.pal(11,"RdYlGn")  # 植被: 绿增红减
pal_water <- brewer.pal(11,"RdBu") # 水体: 蓝增红减
pal_built <- brewer.pal(9,"YlOrRd")   # 建筑: 红增黄减
message_line("生成 NDVI 斜率空间分布图...")
save_seq_plot({
  # 单图不再需要设置 par(mfrow) 和 mtext
  plot(ndvi_trend[[1]], main="NDVI 斜率 (植被)", col=pal_veg, legend.width=1)
}, "NDVI_Slope_Map", # 使用新前缀, 如 NDVI_Slope_Map_001.png
```

```
width = 8, height = 7, res = 200 # 调整为适合单图的尺寸
)
message_line("生成 NDWI 斜率空间分布图...")
save_seq_plot({
  plot(ndwi_trend[[1]], main="NDWI 斜率 (水体)", col=pal_water, legend.width=1)
}, "NDWI_Slope_Map", # 如 NDWI_Slope_Map_001.png
width = 8, height = 7, res = 200
)
# (修改后)
message_line("生成 NDBI 斜率空间分布图...")
save_seq_plot({
  v <- values(ndbi_trend[[1]])
  q_robust <- quantile(v, c(0.01, 0.99), na.rm = TRUE)
  # 确保范围有效（防止所有值都一样）
  if (is.na(q_robust[1]) || is.na(q_robust[2]) || diff(q_robust) == 0) {
    q_robust <- c(-0.1, 0.1) # 设置一个合理的默认值
  }

  plot(ndbi_trend[[1]],
       main="NDBI 斜率 (建筑)",
       col=pal_built,
       legend.width=1,
       zlim=q_robust) # --- 修改：添加 zlim 参数 ---

}, "NDBI_Slope_Map", # 如 NDBI_Slope_Map_001.png
width = 8, height = 7, res = 200
)
message_line("生成 SAVI 斜率空间分布图...")
save_seq_plot({
  plot(savi_trend[[1]], main="SAVI 斜率 (土壤)", col=pal_veg, legend.width=1)
}, "SAVI_Slope_Map", # 如 SAVI_Slope_Map_001.png
width = 8, height = 7, res = 200
)
## 10.3 斜率直方图
message_line("生成 NDVI 斜率直方图...")
save_seq_plot({
  op <- par(mar = c(4, 4, 3, 1)) # 调整单图边距
  on.exit(par(op), add=TRUE)
  hist(values(ndvi_trend[[1]]), breaks=60, main="NDVI 斜率直方图", xlab="Slope (1/yr)",
col="darkgreen")
}, "NDVI_Slope_Histogram")
message_line("生成 NDWI 斜率直方图...")
save_seq_plot({
  op <- par(mar = c(4, 4, 3, 1))
```

```
  on.exit(par(op), add=TRUE)
  hist(values(ndwi_trend[[1]]), breaks=60, main="NDWI 斜率直方图", xlab="Slope (1/yr)",
col="darkblue")
}, "NDWI_Slope_Histogram")
message_line("生成 NDBI 斜率直方图...")
save_seq_plot({
  op <- par(mar = c(4, 4, 3, 1))
  on.exit(par(op), add=TRUE)
    v <- values(ndbi_trend[[1]])
  q_robust <- quantile(v, c(0.01, 0.99), na.rm = TRUE)
  if (is.na(q_robust[1]) || is.na(q_robust[2]) || diff(q_robust) == 0) {
    q_robust <- c(-0.1, 0.1) # 设置一个合理的默认值
  }
  # 根据这个范围生成 60 个 breaks
  b <- seq(q_robust[1], q_robust[2], length.out = 61)
  hist(v, # 仍然使用原始 v
       breaks=b,
       xlim=q_robust,
       main="NDBI 斜率直方图", xlab="Slope (1/yr)", col="darkred")
}, "NDBI_Slope_Histogram")
message_line("生成 SAVI 斜率直方图...")
save_seq_plot({
  op <- par(mar = c(4, 4, 3, 1))
  on.exit(par(op), add=TRUE)
  hist(values(savi_trend[[1]]), breaks=60, main="SAVI 斜率直方图", xlab="Slope (1/yr)",
col="darkolivegreen")
}, "SAVI_Slope_Histogram")
## 11. 所有指数的分类变化图（基于各自斜率的四分位）
message_line("开始生成所有指数的分类变化图...")
# 定义一个辅助函数来处理分类和绘图
process_and_plot_change_category <- function(
    trend_raster,   # 包含 slope, R2, mean 的 RasterStack
    index_name,     # "NDVI", "NDWI" 等
    plot_colors     # c(下降颜色, 稳定颜色, 上升颜色)
) {

  message_line(sprintf("  正在处理 %s...", index_name))

  # 1. 提取斜率图层
  slope_raster <- trend_raster[[1]] # 第一个波段是 slope
  names(slope_raster) <- sprintf("%s_slope", index_name)

  # 2. 计算分位数
  slope_vals <- values(slope_raster)
```

```
  q_hi <- quantile(slope_vals, 0.75, na.rm=TRUE)
  q_lo <- quantile(slope_vals, 0.25, na.rm=TRUE)

  message_line(sprintf("    %s 分类阈值: Q25 (下降) < %.4f, Q75 (上升) > %.4f",
                       index_name, q_lo, q_hi))

  # 3. 执行分类
  # -1 = 显著下降 (Bottom 25%)
  #  0 = 相对稳定
  #  1 = 显著上升 (Top 25%)
  change_cat_raster <- calc(slope_raster, fun=function(v){
    if (is.na(v)) return(NA)
    if (v > q_hi) return(1)
    if (v < q_lo) return(-1)
    return(0)
  })

  # 4. 保存 Tif (使用 INT1S 节省空间)
  out_tif_fn <- file.path(out_dir, sprintf("%s_change_cat.tif", index_name))
  writeRaster(change_cat_raster, out_tif_fn, overwrite=TRUE, datatype="INT1S")

  # 5. 保存 PNG 可视化
  plot_prefix <- sprintf("%s_Change_Category", index_name)
  plot_title <- sprintf("%s 变化类别 (基于四分位)", index_name)

  save_seq_plot({
    plot(change_cat_raster, col=plot_colors, legend=FALSE,
         main=plot_title)

    # 添加分类图例
    legend("topleft",
           legend=c("显著上升 (Top 25%)", "相对稳定", "显著下降 (Bottom 25%)"),
           fill=rev(plot_colors), # 反转图例顺序以匹配 1, 0, -1
           bg="white")
  }, plot_prefix)

  message_line(sprintf("    %s 分类图已保存。", index_name))
}
# 批量调用
# 定义颜色方案 (下降, 稳定, 上升)
cols_veg <- c("#d73027","#ffffff","#1a9850")    # 植被 (NDVI, SAVI): 降(红), 稳(白), 升(绿)
cols_water <- c("#d73027","#ffffff","#4575b4")  # 水体 (NDWI): 降(红), 稳(白), 升(蓝)
cols_built <- c("#1a9850","#ffffff","#d73027")  # 建筑 (NDBI): 降(绿), 稳(白), 升(红)
```

```
# 依次处理每个指数
process_and_plot_change_category(ndvi_trend, "NDVI", cols_veg)
process_and_plot_change_category(ndwi_trend, "NDWI", cols_water)
process_and_plot_change_category(ndbi_trend, "NDBI", cols_built)
process_and_plot_change_category(savi_trend, "SAVI", cols_veg)
message_line("所有指数的分类变化图已全部生成。")
## 12. 汇总输出（统计 & 时间序列表）
message_line("正在为 NDVI 汇总统计提取斜率值...")
slope_vals <- values(ndvi_trend[[1]]) # ndvi_trend 的第1个波段是斜率
slope_vals_clean <- slope_vals[is.finite(slope_vals)]
stat_df <- data.frame(
  metric=c("count","mean","median","sd","min","max","q25","q75"),
  value=c(length(slope_vals_clean), mean(slope_vals_clean), median(slope_vals_clean),
          sd(slope_vals_clean), min(slope_vals_clean), max(slope_vals_clean),
          quantile(slope_vals_clean,0.25), quantile(slope_vals_clean,0.75))
)
write.csv(stat_df, file=file.path(out_dir,"NDVI_slope_stats.csv"), row.names=FALSE)
df_ts <- data.frame(date=dates, NDVI_mean=ndvi_means, NDWI_mean=ndwi_means,
                    NDBI_mean=ndbi_means, SAVI_mean=savi_means)
write.csv(df_ts, file=file.path(out_dir,"Indices_Mean_TimeSeries.csv"),
row.names=FALSE)
message_line("变化检测完成！所有结果已保存到 ", normalizePath(out_dir), "/")
```

输出的结果图如下：

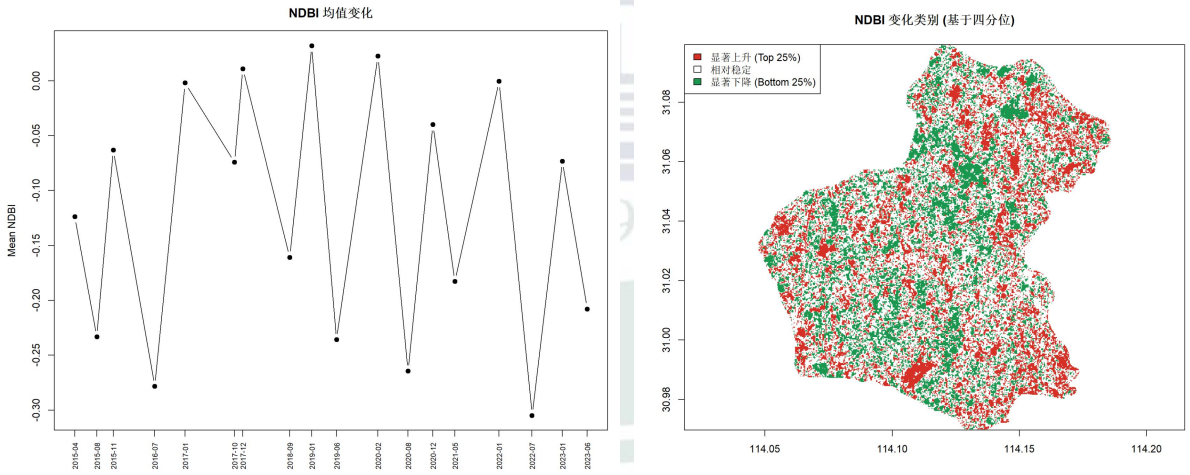


图 3.13.4-1 NBVI 时序分析结果图

NDBI 指数常用于示意不透水地/建筑强度。

时间序列折线图中，全区均值长期为负（-0.30 至 0 附近），说明 NIR 反射普遍强于 SWIR1，区域整体以非强建成面为主。时序曲线在-0.30 至 0.02 之间波动，无持续单向抬升；个别时间点接近 0 或略正，可能对应局部干燥地表或施工裸地阶段的瞬时增强。

四分位分类图（红=上升、绿=下降）呈点状或小斑块分布，西南与东北角均有零散上升或下降混布，不构成连续城镇扩张带。

总的来看，2015–2023 期间，杨店镇区域未出现大尺度、持续化的建成区扩张信号；建成强度变化以小斑块、零散施工变化为主。

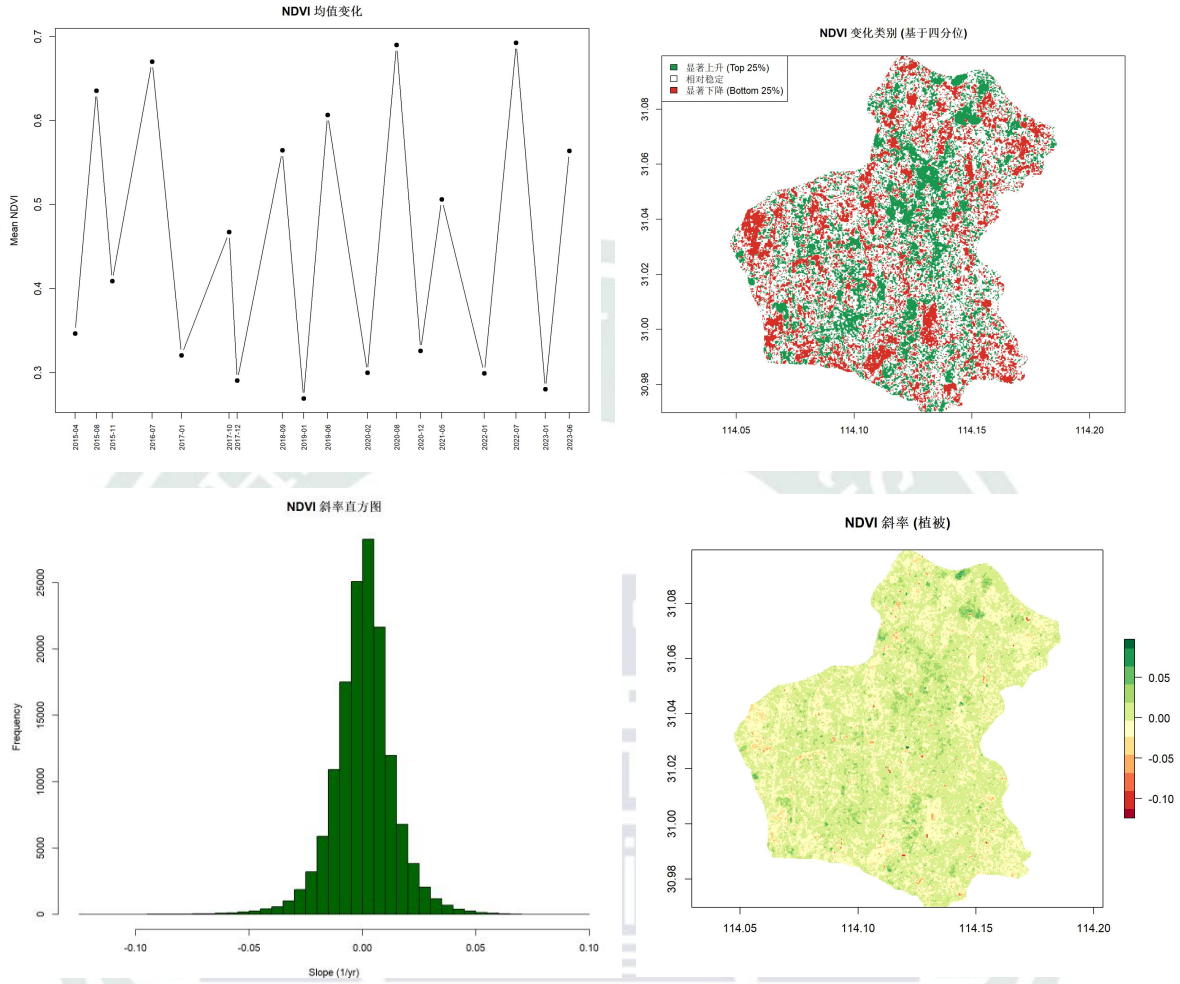


图 3.13.4-2 NDVI 时序分析结果图

NDVI（植被）的变化反映植被覆盖与活力的年际变化。

四分位分类图：显著上升与显著下降“棋盘式”分布，整体均衡；中东部和北部零散上升斑块稍集中，西南侧有下降斑块带。

NDVI 时序图在 0.29–0.69 之间振荡，夏季高、冬季低的季节性明显（图中 2016-07、2020-08、2022-07 等为高峰；2019-01、2020-12、2023-01 等为低谷），从 2015 到 2023 的“峰值—谷值”振幅基本稳定，没有出现持续上扬或下挫的单向走势。末期（2023-06）的均值约 0.56，高于样本中的多次冬季值，也处在历史中上水平。这说明植被总体稳定，年景/降水及耕作节律主导的季节性信号强于多年趋势。

NDVI 斜率直方图以 0 附近对称，峰值在-0.01 到+0.01 1/yr 之间，表示微弱变化占多数。

斜率空间图呈浅黄、浅绿色为主，局部斑块（多在北部与中部）出现浅绿到深绿，指向轻微增长。

总体而言，杨店镇植被状况近 9 年总体稳定，小范围受土地利用微调（复耕/退耕、绿化、小型建设）或季节更替影响出现正负交替变化。

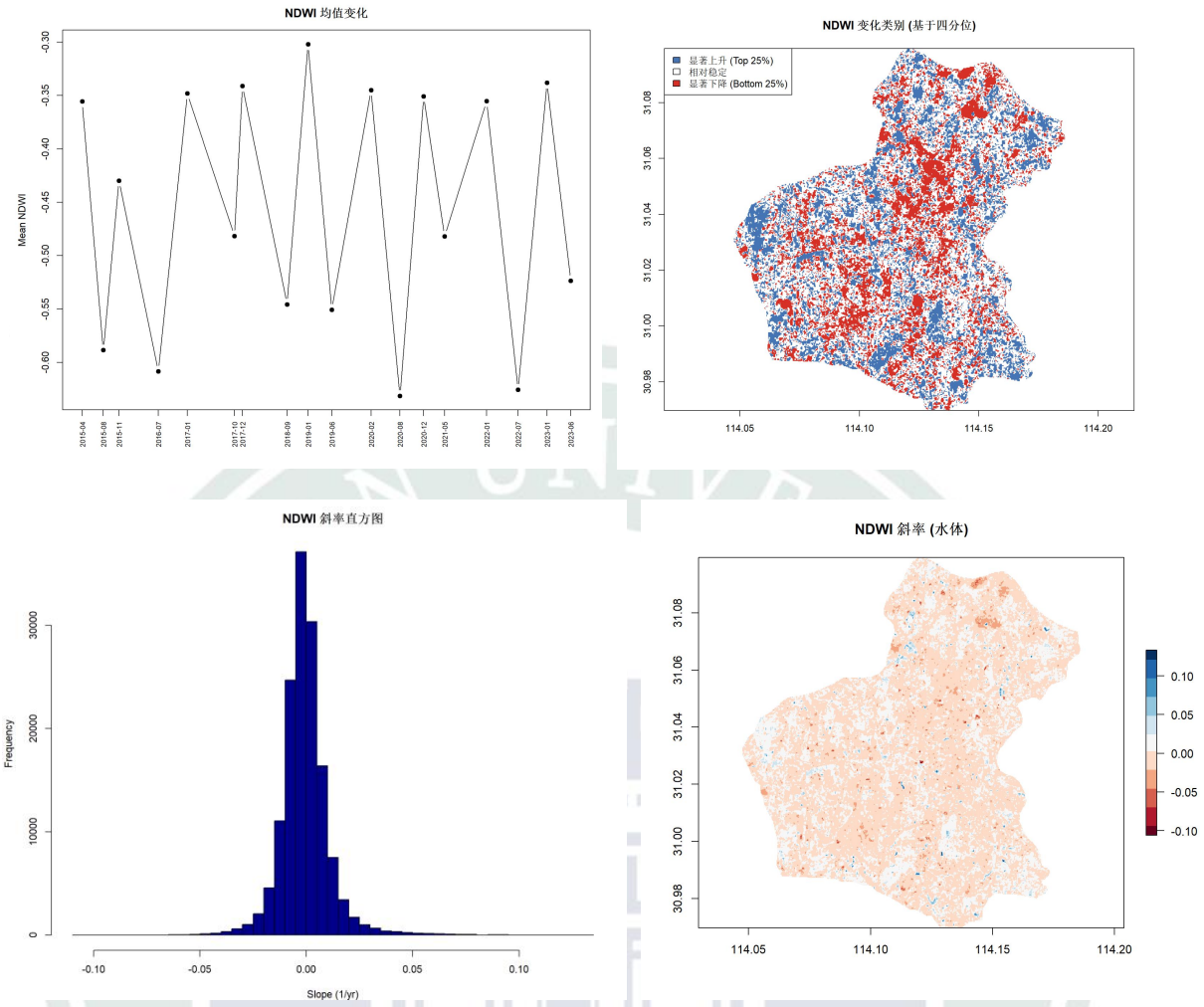


图 3.13.4-3 NDVI 时序分析结果图

NDWI 对地表含水与浅水体敏感。

时间序列折线图中，全区均值为负（-0.60 至-0.30），典型的非大面积开阔水体区域特征。曲线呈“谷—峰—谷”快速起伏，2020-08 与 2022-07 附近相对偏湿，但 2020-12 与 2023-01 为干态极值。这种起伏多与年内降水、蒸散与耕地水分管理相关。

四分位变化分类图（蓝=上升、红=下降）显示上下四分位交错，无单侧占优的区域级趋势。

斜率直方图中心略偏负，小幅负值像元略多，表示随着杨店镇的居民区建设有轻微干化倾向但幅度很小。

斜率空间分布图以浅橙为主，零散蓝色条带沿河网或沟渠分布，代表局部湿度上升；靠西部与南缘有更多浅橙红斑，提示微弱干化。

总的来看，水分状况整体稳定，局部微弱干化可能是耕作制度、作物轮作与季节性水分管理所致，不指向显著水体收缩或扩展。

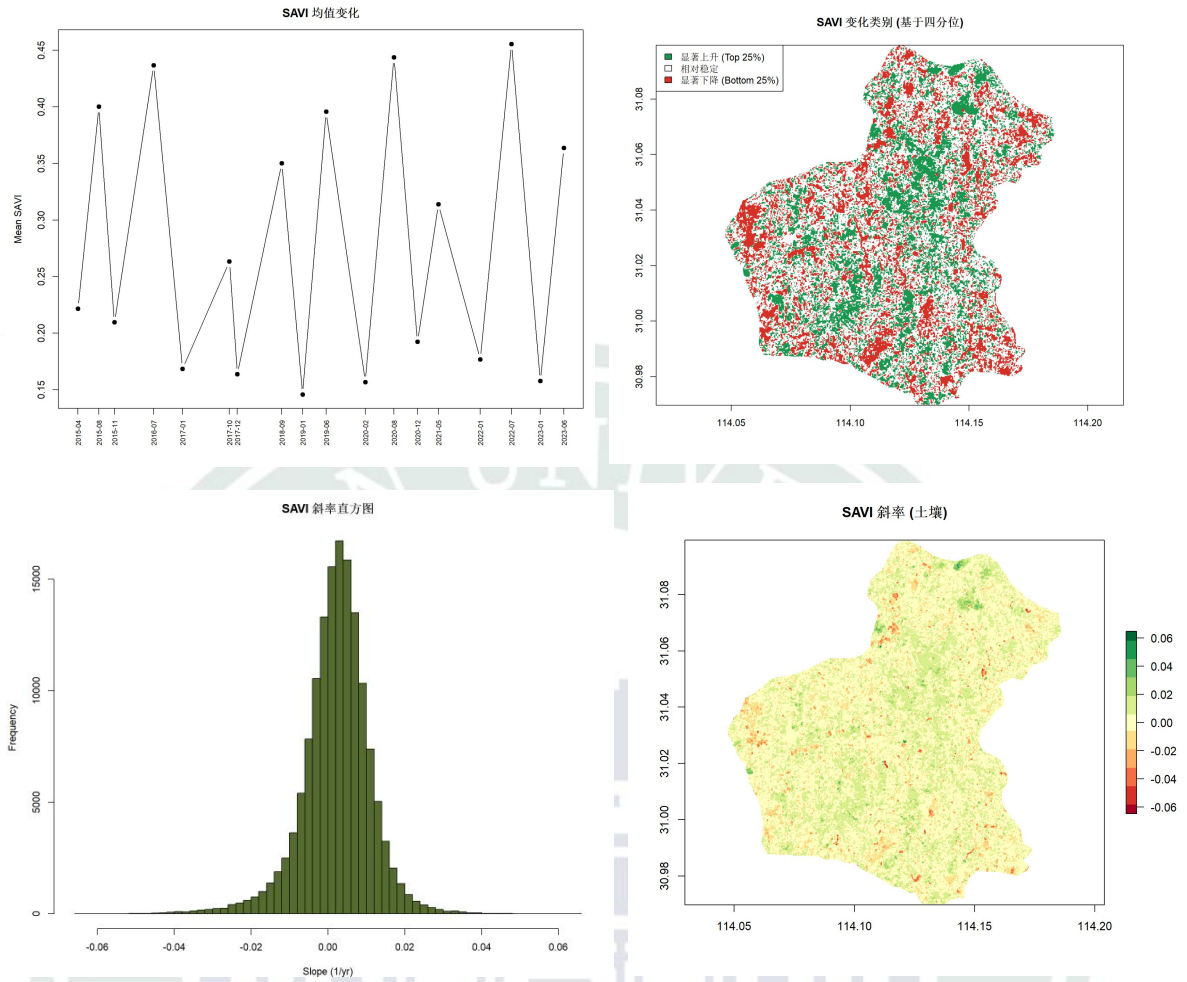


图 3.13.4-4 SAVI 时序分析结果图

SAVI 在低中植被覆盖时比 NDVI 更稳健。

时间序列折线图中，均值 0.15–0.45 之间波动，夏季偏高、冬季偏低，与 NDVI 同步，但峰谷幅度相对小一些。末期（2023-06）回到中高位，表明近年末稀疏植被与裸土混合区未出现持续退化。

四分位变化分类图与 NDVI 类似，呈镶嵌状分布，上升与下降面积总体相当。

直方图对称且尖峰在 0 附近，绝大多数像元年变化率接近零。

斜率地图以浅黄为主，夹杂零散绿或红斑；北部与中部有少量转绿（增植被或土壤变湿或变暗），西南侧有少量转红（更裸更亮或干旱）。

总的来说，稀疏植被与裸土条件长期稳定，小范围受耕作强度、覆被物候随季节变换、土壤湿度波动影响而正负交替。

4 结语

4.1 总结与展望

4.1.1 实习过程总结

本次实践围绕从原始观测到专题产品的完整链路展开，以 GEE 高效筛选与导出 Landsat 8 L2 影像为起点，在 R 环境中完成波段解析、真/假彩渲染、几何子集与矢量掩膜、指数计算与阈值分割、无监督与监督分类、以及多时相趋势诊断。整个实习过程中，我注重规范的数据获取与管理、稳健的处理策略与可复现的代码组织，最终在杨店镇这一小城镇-农田复合景观中较为系统地刻画了植被、水体与建成区的空间格局与年际变化特征。

在数据获取与质量控制上，利用 GEE 按 ROI 统计云量、自动筛选低云影像，并统一完成反射率/温度缩放与裁剪，显著降低本地预处理成本，并结合 QA_PIXEL 掩膜思路与气溶胶/饱和标记层，为后续指数与时序分析提供“干净像元”。

在可视化方法上，基于真彩（B4-B3-B2）与假彩（B5-B4-B3）双线索校验地物解译，结合 NDVI/NDWI/NDBI/SAVI 构建植被-水-建成三角谱系的快速判读框架。Otsu 自适应阈值用于水体、建成区二元掩膜与 NDVI 阈值分割，减少人工阈值主观性。

在分类与后处理方法上，K-means 无监督分类可用于快速获取场景草图，随机森林（RF）监督分类在少量高质量样区驱动下获得高精度结果。3×3 众数滤波显著降低椒盐噪声，改善斑块连续性，提升制图表达质量。

在时序变化分析上，构建 2015–2023 年多时相指数序列，采用闭式解像元级线性趋势（斜率、R2、均值），计算首末期差值与相对变化率，形成趋势强度、趋势方向、趋势不确定性的组合判读。以四分位法将斜率分为上升/稳定/下降三类，便于宏观对比与局部锁定。

4.1.2 实习发现

本次实习获得的主要发现与认知为：

1. 植被（NDVI/SAVI）

杨店镇夏高冬低的季节性信号显著，符合亚热带季风气候雨热同期的气候特征[22]，且多年趋势总体稳定；北部与中部散见轻微上升斑块，西南侧局部下降，但不构成区域性衰退带。多级 NDVI 分级有效区分高郁闭林地与农田成熟期，适合农业物候对比与绿化评估。

2. 水体（NDWI）

全域均值偏负，符合河网、塘库点带状分布、主要地表以农田为主的农村区域特征[23]；年

内起伏与降水情况和耕作管理相关，未见持续收缩或扩展的强信号。阴影可能导致局部假阳性，可通过结合阴影/云影掩膜或多时相一致性抑制。

3. 建成区（NDBI）

建成区的变化以小斑块施工/裸地阶段为主，未观察到大尺度、持续的扩张通道。道路/裸地与建成区在 SWIR/NIR 上的混淆在单时相上较常见，若加入多时相与纹理特征可能有助缓解此类混淆[24]。

4. 模型表现

RF 监督分类 OOB 误差与独立验证精度高度一致（>99%总体精度），说明选得准的特征与干净的样区能够在中分辨率场景下获得鲁棒结果。无监督结果对谱系相近的类别（例如道路与居民区）区分力有限，验证了监督样本对语义解耦的必要性[25]。

4.1.3 未来工作与改进空间

本次实习还存在许多值得开展进一步工作与改进之处：如单景或小样本时序在多云季节易受遮蔽影响，可以引入多景合成（如月-季中位数合成）与严格云/阴影掩膜，提升时间序列的稳健性；现用 Otsu 阈值对单峰分布敏感，面对多峰/偏态时可能偏移，可结合形态学约束或面向对象后处理提高空间一致性；仅使用多光谱反射率与简单指数进行分类等，尚未引入纹理（GLCM）、光谱角/分解、地形校正等特征；后续还可探索谱-空联合聚类（SLIC 超像元 + K-means/谱聚类）或半监督自训练，兼顾边界与语义；时序分析过程中可引入 Theil–Sen/季节-趋势分解（STL）[26]、断点检测（BFAST）[27]以及异常年剔除策略等；仅用 Landsat 8 影像则 10 米级以下的变化难以观测，若与更高空间分辨率的影像融合则有助于道路/小尺度建设识别。

4.1.4 应用前景与扩展

本次实习的工作可应用于区域治理与国土监测、农业与水利、多个领域，如可为小城镇生态本底评估、耕地保护、水系连通性巡查、城市化状态评估等提供量化依据[24]；NDVI/SAVI 多级分级适合耕地长势分区，NDWI 时序可用于农田灌溉管理与旱涝监测的区域预警线索；且本实习的影像处理流程可迁移至其他乡镇，在多景合成与并行计算加持下具备市、县域尺度推广潜力。总体而言，本次实习的结果具有解释力与稳健性，后续在特征增强、多景合成与非线性时序上有继续精进的空间，且有望将本流程升级为面向县域的轻量级长时序遥感监测工具链。

4.2 感想

在本次基于 R 语言的 Landsat 8 遥感影像处理与分析实习中，我不仅掌握了一套完整的技术流程，更以一种独特的视角重新认识了我所研究的这片土地——我的家乡杨店镇。遥感技

术让我得以从空中凝视这片土地，看见它的植被覆盖、水体分布、建成区变迁，看见那些我自幼熟悉的道路、田埂、河流与老宅，对我而言，她们不仅仅是遥感意义上的一种种“地物”，而是在影像中呈现出另一种意义。

影像中的每一个像元，不仅是地物的反射光谱，更是一段段被时光浸润的记忆。当我提取NDVI识别植被时，我想起童年奔跑的桃林；当我用NDWI勾勒水体时，我想起夏日游泳的池塘；当我用监督分类区分建成区时，我想起祖辈在此建房立业的艰辛。遥感不仅是科学工具，更是一面镜子，映照出土地与人之间千丝万缕的联系。

这种技术与情感的共振，促使我在实习的最后，写下一篇跨越千年的家族叙事。它是对我的故乡杨店镇的历史、乃至“杨”这一姓氏源流的追溯，对家族在这片土地上生根、繁衍、奋斗的想象与重构。从晋水之畔的受土立氏，到乌江之侧的因功封侯、昌邑驿馆的暮夜却金、龙兴关的开皇之治，再到江汉平原的“杨家店”，我的家族史与这片土地交织在一起。而今天，我以遥感为笔，以代码为墨，继续书写着这片土地的故事。

这份“感想”，是我在技术背后的人文思考，也是我对故乡最深情的回望。

木昜为杨

楊，蒲柳也[①]。从木，昜[②]聲。——《说文解字》

——题记

图 4.2-1 杨姓图腾

一 晋水杨枝（周厘王四年，公元前678年）

晋水畔的黄昏泛着青铜光泽，周武王第三子唐叔虞[③]第十二世孙姬伯侨站在新封的杨国城垣上。风中飘来熟粟的气息，他的手指抚过夯土墙缝里一株萌发的杨树苗——这顽强的植物已在此地生长了三代人。根系深扎，枝叶蔓发，如同他们这一支姬姓子孙，终于在这片土地上扎下了根。

“大夫，蒙僖王恩准，晋侯赐的祭肉已安置宗庙，胙土命氏之仪，吉时将至。”司仪的声音打断沉思。此刻城下正在举行“胙土命氏”仪式。

① 赤茎柳，即杨树。
② 音 yáng，古“陽”字。
③ 《史记·卷三十九·晋世家第九》：晋唐叔虞者，周武王子而成王弟。……成王与叔虞戏，削桐叶为珪以与叔虞，曰：“以此封若。”史佚因请择日立叔虞。成王曰：“吾与之戏耳。”史佚曰：“天子无戏言。言则史书之，礼成之，乐歌之。”於是遂封叔虞於唐。唐在河、汾之东，方百里，故曰唐叔虞。姓姬氏，字子于。国学唐叔子燮，是为晋侯。

巫祝将新折的杨树枝浸入晋水，清冽的水珠顺着翠绿的叶片滴落，轻轻洒在姬伯侨的额间。那冰凉触感让他微微一颤，他仿佛听到了来自叔虞始祖、乃至周武王先祖的无声嘱托。

“自今尔为杨氏，当如白杨立晋土！”身着玄衣的巫祝声音苍劲有力，在黄昏的空气中回荡。

他缓缓屈膝，朝宗庙方向叩首：“姬姓杨氏伯侨，谨承天命，受此土，命此氏[①]！”

图 4.2-2 杨（篆书）

二 赤泉星火（西汉高帝五年，公元前 202 年）

乌江畔的腥风卷着残夜的余温，杨喜的铁戟深深卡进了那个伟岸身躯的肋骨间。他看见西楚霸王项羽披散着红袍自刎时，溅起的血珠如同破碎的玛瑙，洒在江边摇曳的荻花上。

那一刻，天地寂静，唯有江水呜咽。

数月后，长安未央宫内，杨喜跪接汉高祖赐下的赤泉侯金印时[②]，掌心满是冷汗。鎏金印纽在透过殿门照射进来的日光下灼热如炭，他几乎要握不住这用鲜血换来的荣耀。

深夜，他独对杨氏先祖牌位，忽然懂得了先祖食我[③]当年拼死护住的，不是冰冷的宗庙木主，而是某种比青铜更坚韧的东西。其子杨殷在续修族谱时，在昂贵的丝帛上饱蘸浓墨，用工整的隶书郑重添注：“喜公得爵，复杨姓于汉庭。”笔锋在“杨”字上重重一顿，墨迹深透纸背，仿佛要将这个姓氏永远镌刻进历史的长卷。

图 4.2-3 杨（隶书）

三 暮夜清流（东汉永初二年，公元 108 年）

昌邑官驿的夜晚，万籁俱寂，唯有房中一盏孤灯，灯芯偶尔噼啪炸响，结出一朵小小的、焦黑的蕊花。灯光如豆，将杨震高大的身影投在斑驳的墙壁上，摇曳不定。他的面前，案几上

① 《新唐书·宰相世系表》：杨氏出自姬姓。周宣王子尚父封为杨侯。一云，晋武公子伯侨生文，文生突，羊舌大夫也。又云晋之公族食邑于羊舌，凡三县：一曰铜鞮，二曰杨氏，三曰平阳。突生职，职五子：赤、肸、鲋、虎、季夙。赤字伯华，为铜鞮大夫，生子容。肸字叔向，亦曰叔誉。鲋字叔鱼。虎字叔羆，号“羊舌四族”。叔向，晋太傅，食采杨氏，其地平阳杨氏县是也。

② 《史记·卷七·项羽本纪第七》：……项王乃曰：“吾闻汉购我头千金，邑万户，吾为若德。”乃自刎而死。……最其后，郎中骑杨喜……得其一体。……故分其地为五：……封杨喜为赤泉侯……。

③ 《新唐书·宰相世系表》：……叔向生伯石，字食我，以邑为氏，号曰杨石，党于祁盈，盈得罪于晋，并灭羊舌氏，叔向子孙逃于华山仙谷，遂居华阴。

堆放着的黄金，在昏黄的光线下闪烁着诱人而冰冷的光泽。这光芒，让他不由自主地想起了弘农华阴老宅院墙上，那些历经风霜雨雪的杨树疤痕——每一道扭曲的纹路，都是岁月与磨难刻下的印记，沉默而坚定。

王密，这位他昔日举荐的荆州茂才，如今已是一身绯色官服，在灯下那颜色如在流淌的朱砂十分刺目，映得他面色愈发惶惑不安。

“恩师，”王密的声音带着一丝不易察觉的颤抖，他将身体凑近些，压低嗓音，“学生深知恩师清贫，此不过聊表寸心，感念昔日提携之恩……况且，”他目光扫过紧闭的门窗，语气更轻，“此刻夜深人静，绝无人知晓。”

杨震缓缓推开木窗，春夜潮湿的泥土气息扑面而来。他望着黑暗中一株摇曳的杨树，新发的枝条在月光下泛着青白的光泽。

“天知，神知，我知，子知。何谓无知？”[①]

一字一顿，声音不高，却重若千钧。

“四知”出口，王密脸色苍白，踉跄退走，遗落的金饼滚到杨震脚边，被他用手中的《尚书》竹简轻轻拨开，仿佛拂去尘埃。

二十年后，官至太尉的杨秉跪在父亲墓前。白玉碑碣映出他官袍上的银鱼绣纹，风中传来稚子诵读《杨氏诫谕》的清音：“暮夜却金，清白传家……”他想起那夜父亲在驿馆种下的不是后人传颂的道德丰碑，而是深扎在杨氏血脉里的根脉，比晋水的杨枝更坚韧，比赤泉的金印更珍贵。

四 龙兴关中（北周建德七年，公元578年）

那一天，长安的宫墙在暮色中泛着冷硬的光。你听闻伽罗独自入宫，诣阁陈谢，直至叩头流血，才终于保住了女儿的性命[②]。你急忙赶去，看见她走出宫门时苍白的脸上那道刺目的血痕。

回家的车上，你们相对无言。车厢随着碾过石板路的节奏微微摇晃，斜阳从竹帘的缝隙漏进来，在她染血的额头上投下跳跃的光斑。你想起这些年在北周朝廷的步步惊心，宇文护专权时，独孤一门被屠戮殆尽，只剩伽罗一人；弘农杨氏虽贵为关陇门阀，在宣帝宇文赟这般荒淫暴戾的君主治下亦是如履薄冰。

回到家中，你谨慎地关上房门，木门发出沉重的声响。

“还疼吗？”

你在伽罗的额前给她小心地包扎伤口。

① 《后汉书·卷五十四·杨震列传第四十四》：大将军邓骘闻其贤而辟之，举茂才，四迁荆州刺史、东莱太守。当之郡，道经昌邑，故所举荆州茂才王密为昌邑令，谒见，至夜怀金十斤以遗震。震曰：“故人知君，君不知故人，何也？”密曰：“暮夜无知者。”震曰：“天知，神知，我知，子知。何谓无知！”密愧而出。后转涿郡太守。性公廉，不受私谒。子孙常蔬食步行，故旧长者或欲令为开产业，震不肯，曰：“使后世称为清白吏子孙，以此遗之，不亦厚乎！”

② 《周书·卷九·列传第一·宣帝杨皇后》：帝后昏暴滋甚，喜怒乖度。尝谴后，欲加之罪，后进止详闲，辞色不挠。帝大怒，遂赐后死，逼令引诀。后母独孤氏闻之，诣阁陈谢，叩头流血，然后得免。

“嗯。”

你们相拥而泣。

那一天，你发誓此生定要保护好自己的家人；

那一天，不知是出于妻女受辱的屈辱，还是对九五之尊的渴望，抑或是这二者的交织，你开始布局夺权；

那一天，你终于下定决心：天命，有德者当执之！

……

后来，周宣帝宇文赟骤然驾崩，留下幼主临朝。你作为皇太后杨丽华的父亲，以外戚身份入主朝堂，官拜大丞相、假黄钺、都督中外诸军事，总揽朝政。一时间，你权倾天下，却也置身于风口浪尖。相州总管尉迟迥、郧州总管司马消难、益州总管王谦先后举兵，反对你的势力暗流涌动。

一天夜里，你辗转难眠，朦胧间做了一个梦，梦见童年时华阴宅邸庭院中，那株历经三代、枝繁叶茂的老杨树，在狂风暴雨、电闪雷鸣之中，虬结的根系拔地而起，粗壮的树干扭曲、膨胀，最终在一声震彻天地的龙吟里，化作一条鳞甲森然的赤色巨龙，腾空而去，直上九霄云外。

翌日，你入宫代幼帝批阅奏章，心神仍被那梦境萦绕。你一边反复回想这个奇异的梦，一边在关于均田制的奏章里用朱笔勾画出弘农杨氏应得的永业田数额。就在你凝神间，目光扫过案几，发现砚台下压着一张素笺。取过展开，你再熟悉不过的伽罗的字迹映入眼中：

“大事已然，骑兽之势，必不得下，勉之！”①

你攥紧了手中的虎符。

……

后来，三方之乱既平，开皇元年二月甲子日，你受禅即皇帝位于临光殿，册封独孤氏为皇后，改元开皇，定国号大隋②。

登基大典过后，你和伽罗携手来到宫阙高处，俯瞰着在晨曦中苏醒的帝都。你想起了晋水之畔的受封、赤泉侯的搏杀、太守的“四知”，以及无数杨氏先人在历史长河中留下的足迹。你深知，自己已将这个氏族带到了前所未有的高度。当然，这并非终点。你立志要建立一个超越前代的强大、统一、富庶的帝国，要如同那梦中化龙的杨树，扎根深厚，枝繁叶茂，庇佑万民。

……

后世的史书中这样评价你：

“……虽晋武之克平吴会，汉宣之推亡固存，比义论功，不能尚也。七德既敷，九歌已洽，要荒咸暨，尉候无警。于是躬节俭，平徭赋，仓廪实，法令行，君子咸乐其生，小人各安其业，

① 《隋书·卷三十六·列传第一·文献独孤皇后》：及周宣帝崩，高祖居禁中，总百揆。后使人谓高祖曰：“大事已然，骑兽之势，必不得下，勉之！”

② 《隋书·卷一·帝纪第一·高祖上》：开皇元年二月甲子，上自相府常服入宫，备礼即皇帝位于临光殿。设坛于南郊，遣使柴燎告天。是日告庙，大赦，改元。

强无陵弱，众不暴寡，人物殷阜，朝野欢娱。二十年间，天下无事，区宇之内晏如也。考之前王，足以参踪盛烈。……”

他们说，你和伽罗创造的盛世，名为“开皇之治”。

图 4.2-4 杨（颜真卿楷书）

五 杨店晨曦（公元 1436 年，明正统元年）

江汉平原的晨雾尚未完全散尽，杨廷松就来到镇口新落成的青石牌坊之下。他手中的账册写着他设立的“六义”：义仓以备荒，义学以兴教，义馆以待远，义桥以济涉，义塍以安亡，义井以济渴[①]。朱砂批注的款项数字，在熹微的晨光中闪烁。

“东家，湖广来的客商已至驿馆，正等着验看我们今春新出窑的漆器花样。”老管家轻声提醒着。

杨廷松微微颔首，目光却越过禀报的管家，望向那条贯穿镇子、南来北往的驿道。此时，道旁悬挂的灯笼正被家仆们次第点亮，橘黄色的光芒在薄雾中晕染开来，最终清晰地勾勒出“杨家店”三个遒劲的鎏金大字。更夫敲响了五更的梆子，清脆的声音在镇子上空回荡。与此同时，镇东头的义学堂里，传来了蒙童们稚嫩而清朗的早诵声。他们诵读的，正是关于杨姓始祖叔虞封唐、汉之清白太守杨震“暮夜却金”的故事。那些跨越千年的古老训诫，与窗外杨树叶片的沙沙声响交织在一起，飘过青石板路，落进镇西头榨油坊里刚刚滤出的、清亮醇香的桐油之中，仿佛将文化的种子，浸润到了这里的每一个角落。

又是一年春风度，杨店镇外的桃花林，已是烂漫如霞，绵延数里，与镇中青瓦白墙的民居相映成趣。杨廷松手抚长须，目光深邃地眺望着这片土地。他的身后，是刚刚落成的杨氏宗祠，青砖灰瓦，飞檐斗拱，气势恢宏，门前石狮肃立，象征着家族的秩序与荣光。

“廷松公，”一位负责谱牒的族中子弟手捧厚厚的族谱稿本，“族谱已核对完毕。自宋末元初，我支先祖为避战乱迁居至此澴川之地，垦殖建业，形成聚落。考其源流，我杨店杨氏一脉系出弘农华阴望族，上溯可至隋高祖文皇帝杨坚，再往上乃至汉赤泉侯杨喜、‘关西孔子’太尉杨震，直至周室分封之杨侯伯侨。脉络清晰，世系分明。您所设义仓一事，也是隋高祖文皇帝于开皇五年始设的[②]。”

杨廷松闻言，缓缓转过身，目光扫过巍峨的宗祠，掠过繁花似锦的桃林，最终落在驿道上那些开始忙碌的杨氏商行的车队上。他的脸上露出了欣慰而沉稳的笑容。从晋水的封侯立姓，

① 《乾隆湖广通志·卷六十四》：杨廷松，字汝节，孝感人。父沔，饶于财，析箸时，兄弟各择腴产，松欣然受其瘠者。性好客，恤灾，所居城市，今名杨家店。岁饥，出粟贷人，或不能偿，焚其券。更置义仓、义学、义馆、义桥、义塍、义井，远近便之。翰林王廷陈、御史毛凤诏皆有六义记，茅坤为之传。

② 《隋书·卷一·帝纪第一·高祖上》：戊申，车驾至自洛阳。五月甲申，诏置义仓。

到赤泉的搏杀复起，从太尉的暮夜清白，到隋帝的龙兴开皇，千年的家族史诗，最终在这江汉平原的晨曦中，凝结成了“杨家店”这个踏实而充满烟火气的名字。历史的风云变幻，化作了义学里的书声、榨油坊的香气、驿道上的车马、以及宗祠里袅袅的香火。杨氏的根脉，在此地深扎于泥土，繁茂于市井，将继续在这片土地上生生不息。

六 根脉新绿

“今夜月明人尽望，不知秋思落谁家？”

我望着图书馆窗外的明月，思绪回到了上半年清明节回老家的那一天——

……

又是一年清明，清明时节雨纷纷。

烟雨中，我推开杨店镇织娘院子老宅的木门，陈旧的木香混合着雨后泥土的清新扑面而来。阳光透过天井上方的玻璃亮瓦，在布满岁月痕迹的青石板上投下斑驳的光晕，无数微尘在光柱中缓缓浮动，像是时光的碎屑。

我的目光落在堂屋正中最显眼的位置，新修撰的《杨店杨氏族谱》被郑重地安放在香案之上。深蓝色的封面上，“弘农杨氏”四个烫金篆字在略显昏暗的堂屋内，依然散发着幽微而持重的光芒。我走上前，指尖拂过冰凉的缎面，一种难以言喻的厚重感顺着指尖蔓延开来。

翻开扉页，是始祖叔虞封唐的古朴绘像。再往下，是工笔细描的“晋水杨枝”场景，姬伯侨站在城垣上，巫祝手持浸水的杨枝，历史的瞬间被定格在泛黄的纸页间。我的手指轻轻划过那些密密麻麻的姓名与谱系，仿佛能触摸到流淌在血脉深处的记忆。

“周厘王四年，伯侨公受封杨侯，胙土命氏，肇基杨姓之始……”

“汉高祖五年，喜公从龙佐命，以战功封赤泉侯，杨氏由是显于朝……”

“汉永初二年，震公任东莱太守，暮夜却金，恪守‘四知’，清白之风，垂范后世……”

“隋开皇元年，坚公受禅登极，肇基大隋……隋开皇九年，渡江灭陈，南北归一……隋高祖文皇帝受天明命，奄有区夏，拯群飞于四海，革凋散于百王，恤狱缓刑，生灵皆遂其性，轻徭薄赋，比屋各安其业……恢夷宇宙，混壹车书，开皇之治，泽被天下……”

“明正统元年，廷松公于澴川斗山铺立杨家店，立‘六义’以惠桑梓，德泽绵长，宗风永续……”

每一个名讳，每一段简短的记载，背后都是一段波澜壮阔的人生，都连接着一段宏大的历史。我此刻站在这本千年族谱前思考：我是谁？是那个在晋水畔接受杨枝洒额的姬伯侨？是那个在乌江畔搏杀建功的杨喜？是那个在昌邑驿馆秉烛拒金的杨震？还是那个在晨曦中规划“六义”的杨廷松？……

或许，都是。又或许，我只是这漫长根脉上，又萌发出的一片新绿。

窗外，一阵风吹过，院中那棵老杨树新发的叶片沙沙作响，声音清脆，带着勃勃生机。那声音，与族谱纸页的翻动声，与义学堂里隐约传来的童谣声——原来如今镇上小学的孩子们仍

在传颂杨震的故事——奇妙地交织在一起。

我合上族谱，走出老宅。镇上的青石牌坊依旧矗立，但周围已是楼房林立，车水马龙。“杨家店”这个名字，已从一个小镇，变成了这片土地上一个深刻的文化印记。上下班的年轻人骑着电瓶车从刻有“义渡”典故的石碑旁疾驰而过；曾经的“义仓”旧址旁，建起了现代化的粮食储备库；而“义学”的精神，则延续在镇上的“桃花驿小学”那朗朗的读书声中。

我漫步到镇郊的桃花林，这里依旧是春日里最美的景致。游人在花间拍照留念，笑语欢声。我站在一株繁茂的桃树下，眺望着这片祖先筚路蓝缕、开拓生息的土地。千年的风起云涌，帝业的辉煌，士族的清誉，商贾的诚信，最终都沉淀为这寻常村镇里的烟火人间，化作了每一个杨姓子孙血脉里流淌的基因密码——那是如同白杨般扎根大地的坚韧，是“四知”般的清廉自守，是“六义”般的仁心担当。

……

思绪回到现实。此刻，我正细细端详着电脑前自己亲自获取、处理、分析的杨店镇的遥感影像，每一个像元背后都是我的故土，更是每一位杨店人赖以为生、辛勤建设的成果。是的，历史将由生活在这片土地上的每一个平凡的我们，用每一天的生活，继续书写下去。从晋水的青铜光泽到杨店桃林的晨曦，这条名为“杨氏”的根脉穿越了近三千年的时光，依然鲜活，正在这片古老而又崭新的土地上孕育着下一个春天。

图 4.2-5 诗经名物图解·杨树枝

杨丹阳

2025 年 11 月 5 日

月是故乡明[①]

① 杜甫《月夜忆舍弟》：“露从今夜白，月是故乡明。”公元 2025 年 11 月 6 日 6 时 27 分，月球过近地点，距离地球约 35.7 万公里，这是 2025 年所有满月中与地球最近的一次。因此，公元 2025 年 11 月 5 日 21 时 19 分前后，中国大部分地区的将月亮呈现出最完满的圆形，这轮月亮将是公元 2025 年的最大满月，俗称“超级月亮”。本文完成于公元 2025 年 11 月 5 日当晚。

致　谢

在本次实践与论文撰写过程中，谨向所有帮助过我的人致以诚挚的感谢。

首先感谢武汉大学水利水电学院的查元源老师在上课过程中的细致指导与平时的耐心指点，他在上课过程中详细地指导我 R 语言的及其包的用法，拓展讲解了 Google Earth Engine 等的使用方法，并经常耐心地回答我的提问，查老师的帮助使本报告的完成更加顺利。

感谢同学们的交流讨论，他们的分享与经验让我在本次实习过程中的许多细节得以完善。

感谢各个开源社区、软件、平台等提供的卓越工具链，让本项目能够完成端到端的完整的遥感处理与分析流程。

最后，还要感谢我的家人与宗族观念，以及我的研究区域——家乡杨店镇与老家织娘院子的自然与人文风情，为本次实践注入了独特的意义。

杨丹阳

2025 年 11 月 5 日

附　录

附录 A 查重检测报告

在撰写本论文的全程中，我始终坚守严谨求实的学术原则，致力于对基于 R 语言的 Landsat 8 遥感影像处理与分析这一实践主题进行深入的剖析与探究。为了进一步培养学术诚信的科研品格，弘扬求真务实的学术风气，作者在附录部分附上本论文的查重检测报告。此报告详细列出了论文与既有数据库中其他文献之间的相似度对比结果，旨在充分验证本论文的原创性与学术价值。

PaperYY® 检测报告单–打印版

检测文献：基于R语言的 Landsat 8 遥感影像处理与分析（免费版）
文献作者：杨丹阳
报告时间：2025-11-16 14:55:21
段落个数：23
报告编号：YY202511161455003919
检测范围：中国期刊库 中国图书库 硕士论文库 博士论文库 会议论文库 报纸库 网友专利库 网友标准库 网友共享库 个人对比库 网页库 百科库

相似度 0.7%

总文字复制比：0.7%
去除引用文献复制比：0.7%　　去除本人已发表文献复制比：0.7%
单篇最大文字复制比：0.2%

重复字数：973　　总字数：137,922（不含参考文献）
总段落数：23（不含参考文献）　　前部重合字数：555
疑似段落数：7　　后部重合字数：418
单篇最大重复字数：262　　疑似段落最小重合字数：18

图 A-1 查重检测报告

附录 B 完整源代码

本节附录包含本实验所使用的全部源代码。为便于查阅与复现，本实验完整源代码已上传至作者的 GitHub 仓库，并在以下链接中公开：

https://github.com/yangdanyang2005/Remote_Sensing_Image_Processing_Using_R

使用与引用说明：

（1）本附录中所有代码均为作者独立编写或在开源基础上改进而成，版权归作者本人所有。

（2）源代码遵循 MIT License 开源协议，允许学习、引用与修改，但禁止未经许可的商业用途。若在科研论文、教学或软件开发中使用或引用，请注明引用信息。以下引用信息可供参考：

Yang, D. (2025). Remote sensing image processing using R: Complete experiment source code [Computer software]. GitHub. https://github.com/yangdanyang2005/Remote_Sensing_Image_Processing_Using_R

（3）**本附录代码仅供本实验与学术研究使用，任何非学术性传播、商业化行为或未授权再分发均属侵权。**

本实验所使用的完整源代码如下：

① Remote_Sensing_Image_Processing_Using_R.R

```
# ---- 0 环境设置与工具函数 ----
## ----  0.1 工作路径与包加载 ----
setwd("E:/code/r_code/20251031_Remote_Sensing_Image_Processing_Using_R") # 设置工作路径
library(raster)
library(grDevices)
library(sp)
library(RColorBrewer)
library(randomForest)
library(sf)
message("环境设置与包加载完成。")
## ----  0.2 工具函数 ----
# 用于给输出图像/图表自动编号并写入 res/ 目录，便于后续比对
.counter <- new.env(parent = emptyenv()); .counter$i <- 1
next_name <- function(label, ext = "png"){
    fn <- sprintf("res/%02d_%s.%s", .counter$i, label, ext)
    .counter$i <- .counter$i + 1
    fn
}
```

```r
save_seq_img <- function(im, label){
    fn <- next_name(label, "png")
    save.image(im, fn)
    return(fn)
}
save_seq_plot <- function(expr_plot, label, width=960, height=720, res=120){
    # 以 png 设备保存任意绘图表达式（传入大括号块），并自动关闭设备
    fn <- next_name(label, "png")
    png(fn, width=width, height=height, res=res)
    on.exit(dev.off(), add = TRUE)
    force(expr_plot)
    message("已输出：", fn) # 添加消息提示
    return(fn)
}
# ---- 1 影像的获取与基本信息 ----
## ----  1.1 影像数据 ----
tif_path <- "GEE_L8_Yangdian_image/L8_ROIcf_20230727_LC81230382023208LGN00.tif"
out_dir <- "res"
## 检查 "res" 文件夹是否存在
if (dir.exists(out_dir)) {
    message("检测到 'res' 文件夹，正在清空...")
    # 强制递归删除 "res" 文件夹及其所有内容
    unlink(out_dir, recursive = TRUE, force = TRUE)
}
## 重新创建空的 "res" 文件夹
dir.create(out_dir)
message("'res' 文件夹已清空并准备就绪。")
## 开启日志记录
# 将所有控制台输出保存到 log.txt
log_file <- file.path(out_dir, "log.txt")
message("正在将所有控制台输出重定向到: ", log_file)
sink(log_file, append = FALSE, split = TRUE) # split=TRUE 保证控制台和文件同时输出
## ----  1.2 影像的性质 ----
## 读取影像栅格数据
img <- stack(tif_path)
names(img)
## 波段提取函数
pick_band <- function(x, key = "B2") {
    nm <- tolower(names(x))
    key <- tolower(key)
    i <- grep(paste0("(^|[_-])", key, "0?2?($|[_-])"), nm)
    if (length(i) == 0) i <- grep("b2", nm)
    if (length(i) == 0) stop("找不到波段：", key)
```

```
    x[[i[1]]]
}
b2 <- pick_band(img, "B2") # Blue
b3 <- pick_band(img, "B3") # Green
b4 <- pick_band(img, "B4") # Red
b5 <- pick_band(img, "B5") # NIR
## ---- 1.3 影像信息和统计指标 ----
## 坐标参考系（CRS）
crs(b2)
## 像元/行列/维度等基础信息
ncell(b2)              # 像元总数
nrow(b2); ncol(b2)     # 行列数
dim(b2)                # 若为 RasterStack/Brick，会返回 (nrow, ncol, nlayers)
## 空间分辨率
res(b2)
## 通道（波段）数
nlayers(b2)
## 比较两个栅格是否具有相同的空间属性（范围、分辨率、投影等）
compareRaster(b2, b3)
## 除了第 8 波段（全色，分辨率 15 m）以外，其余波段分辨率为 30 m，可自由叠加。
## 例如：叠加 5-4-3 波段（近红外-红-绿）
s <- stack(b5, b4, b3)
## ----  1.4 单通道和多通道图像 ----
## 单通道灰度图（B2/B3/B4/B5）
## 使用灰度色带，并在一个 2x2 画布中展示四个波段
save_seq_plot({
    op <- par(mfrow = c(2,2), mar = c(3,3,2,1))
    on.exit(par(op), add = TRUE)
    plot(b2, main = "Blue (B2)",  col = gray(0:100/100))
    plot(b3, main = "Green (B3)", col = gray(0:100/100))
    plot(b4, main = "Red (B4)",   col = gray(0:100/100))
    plot(b5, main = "NIR (B5)",   col = gray(0:100/100))
}, "单通道_灰度")
## 简单线性拉伸以增强对比（2%-98% 分位裁剪）
lin_stretch <- function(x, probs = c(0.02, 0.98)){
    q <- quantile(values(x), probs = probs, na.rm = TRUE)
    xx <- clamp(x, lower = q[1], upper = q[2], useValues = TRUE)
    (xx - q[1]) / (q[2] - q[1])
}
b2s <- lin_stretch(b2); b3s <- lin_stretch(b3)
b4s <- lin_stretch(b4); b5s <- lin_stretch(b5)
save_seq_plot({
    op <- par(mfrow = c(2,2), mar = c(3,3,2,1))
    on.exit(par(op), add = TRUE)
```

```
    plot(b2s, main = "Blue (拉伸)",  col = gray(0:100/100))
    plot(b3s, main = "Green (拉伸)", col = gray(0:100/100))
    plot(b4s, main = "Red (拉伸)",   col = gray(0:100/100))
    plot(b5s, main = "NIR (拉伸)",   col = gray(0:100/100))
}, "单通道_拉伸")
## 真彩与假彩合成
s_true  <- stack(b4, b3, b2) # R, G, B
s_false <- stack(b5, b4, b3) # NIR, R, G
## 自动判定 scale
maxDN <- max(c(maxValue(b2), maxValue(b3), maxValue(b4), maxValue(b5)), na.rm =
TRUE)
## Landsat-8 原始数据是 16-bit
plot_scale <- if (is.finite(maxDN) && maxDN > 1) maxDN else 1
## 反射率转换
to_reflectance <- function(x) {
    calc(x, fun = function(v) 0.0000275 * v - 0.2)
}
if (plot_scale > 1) {
    # 堆叠顺序为 B, G, R, NIR (索引 1, 2, 3, 4)
    img_ref <- to_reflectance(stack(b2, b3, b4, b5))
}
## 绘制真彩 / 假彩并保存
save_seq_plot({
    plotRGB(s_true,  scale = plot_scale, stretch = "hist",
            main = "真彩色 (B4,B3,B2)")
}, "真彩色")
save_seq_plot({
    plotRGB(s_false, scale = plot_scale, stretch = "hist",
            main = "假彩色 (B5,B4,B3)")
}, "假彩色")
if (exists("img_ref")) {
    # 由于 img_ref 尚未重命名，我们使用索引访问
    # 索引：1=Blue, 2=Green, 3=Red, 4=NIR
    save_seq_plot({
        plotRGB(stack(img_ref[[3]],  img_ref[[2]], img_ref[[1]]), # 真彩 R,G,B
                scale = 1, stretch = "hist", main = "真彩色(反射率)")
    }, "真彩色_反射率")
    save_seq_plot({
        plotRGB(stack(img_ref[[4]], img_ref[[3]], img_ref[[2]]), # 假彩 NIR,R,G
                scale = 1, stretch = "hist", main = "假彩色(反射率)")
    }, "假彩色_反射率")
}
## NDVI 计算并输出
## 使用索引 (4=NIR, 3=Red) 或原始变量
```

```r
nir <- if (exists("img_ref")) img_ref[[4]] else b5
red <- if (exists("img_ref")) img_ref[[3]] else b4
## NDVI = (NIR - Red) / (NIR + Red)
ndvi <- (nir - red) / (nir + red)
## 输出 NDVI 栅格（使用 next_name 自动编号）
ndvi_tif <- next_name("NDVI", "tif")
writeRaster(ndvi, ndvi_tif, overwrite = TRUE)
message("已输出：", ndvi_tif)
## 绘图保存 NDVI
save_seq_plot({
    plot(ndvi, col = rev(terrain.colors(10)), main = "NDVI (Landsat-8 L2 C2)")
}, "NDVI")
message("全部结果已保存在：", normalizePath(out_dir))
## ---- 1.5 通道提取和重命名 ----
if (exists("img_ref")) {
    message("1.5 节：使用在 1.4 节中创建的 'img_ref'。")
    # 如果 img_ref 存在，我们就用它
    img_demo_stack <- img_ref
} else {
    message("1.5 节：'img_ref' 未创建。将使用 b2, b3, b4, b5 手动创建演示堆栈。")
    # 如果 img_ref 不存在，就用 1.2 节的波段手动堆叠一个
    # 堆叠顺序与 1.4 节中 'to_reflectance' 的输入顺序一致
    img_demo_stack <- stack(b2, b3, b4, b5)
}
message("正在对演示堆栈 'img_demo_stack' 执行重命名...")
## 对 'img_demo_stack' 操作
names(img_demo_stack) <- c("Blue", "Green", "Red", "NIR")
## 重命名 (Renaming)
message("原始 'img' 堆栈的波段名称：")
print(names(img)) # 打印原始名称
message("已重命名的 'img_demo_stack' 堆栈名称：")
print(names(img_demo_stack)) # 打印重命名后的名称
## 提取 (Subsetting)
message("从 'img_demo_stack' 中使用 [[...]] 提取 'Red' 波段（演示）：")
# 现在 'img_demo_stack' 已被重命名，可以按名称提取
red_band_demo <- img_demo_stack[['Red']]
message("从 'img_demo_stack' 中使用 subset() 提取 'Green' 和 'Blue'（演示）：")
gb_bands_demo <- subset(img_demo_stack, c("Green", "Blue"))
# 检查层数
message("提取 'Red' 后的层数：", nlayers(red_band_demo))
message("提取 'Green' 和 'Blue' 后的层数：", nlayers(gb_bands_demo))
save_seq_plot({
    plot(red_band_demo, main = "1.5 提取演示：Red 波段", col = gray(0:100/100))
}, "提取_Red")
```

```r
## ---- 1.6 空间裁剪或取子集 ----
## 本节演示两种常见操作：
## 1) 用 Extent 进行规则矩形裁剪 crop()
## 2) 用矢量多边形进行掩膜 mask()
message("1.6 节：开始空间裁剪与掩膜演示...")
## 选定用于裁剪的多波段栈（优先使用已重命名的 img_demo_stack；作为兜底用 s 或 stack(b5,b4,b3)）
if (exists("img_demo_stack")) {
    rs_base <- img_demo_stack
    message("使用 img_demo_stack 作为裁剪基础数据（层名：", paste(names(rs_base), collapse = ","), "）")
} else if (exists("s")) {
    rs_base <- s
    message("未发现 img_demo_stack，使用 s <- stack(b5,b4,b3) 作为基础数据。")
} else {
    rs_base <- stack(b5, b4, b3)
    names(rs_base) <- c("NIR","Red","Green")
    message("未发现 s，临时创建 stack(b5,b4,b3) 作为基础数据。")
}
## 1) Extent 规则裁剪
message("创建 Extent 进行规则矩形裁剪 ...")
## 如果你知道目标范围，可在此替换为自己的坐标（需与数据 CRS 一致）
## 这里用 rs_base 的范围中间缩一圈作为示例（保证示例可运行）
ext_full <- extent(rs_base)
dx <- (xmax(ext_full) - xmin(ext_full)) * 0.1
dy <- (ymax(ext_full) - ymin(ext_full)) * 0.1
e <- extent(xmin(ext_full) + dx, xmax(ext_full) - dx,
            ymin(ext_full) + dy, ymax(ext_full) - dy)
message("矩形裁剪范围：", toString(c(xmin(e), xmax(e), ymin(e), ymax(e))))
rs_crop <- crop(rs_base, e)
message("矩形裁剪完成。尺寸(行,列,层)：", paste(dim(rs_crop), collapse = " x "))
## 可视化：裁剪后的真彩/假彩（根据可用波段选择）
save_seq_plot({
    if (all(c("Red","Green","Blue") %in% names(rs_crop))) {
        plotRGB(rs_crop[[c("Red","Green","Blue")]], scale = 1, stretch = "hist",
                main = "矩形裁剪-真彩（若为反射率则 scale=1）")
    } else if (all(c("NIR","Red","Green") %in% names(rs_crop))) {
        plotRGB(rs_crop[[c("NIR","Red","Green")]], scale =
ifelse(maxValue(rs_crop[[1]]) > 1, maxValue(rs_crop[[1]]), 1),
                stretch = "hist", main = "矩形裁剪-假彩(NIR,Red,Green)")
    } else {
        plot(rs_crop[[1]], main = "矩形裁剪-首层示意", col = gray(0:100/100))
    }
}, "矩形裁剪_预览")
```

```r
## 2) 多边形掩膜（mask）
## 优先使用已有 shapefile；若没有，可选交互绘制或用示例矩形转多边形
## 请将下方 shp_path 替换为你的矢量边界文件路径（同一坐标系）
shp_path <- "data/Yangdian_Zhiniangyuanzi.shp"  # 例：shp_path <-
"vector/hengsha_draw.shp"
rs_masked <- NULL
if (!is.null(shp_path) && file.exists(shp_path)) {
    message("检测到 shapefile：", shp_path, "，将用于掩膜 ...")
    ## 建议使用 rgdal 或 terra 读取；为尽量不新增依赖，这里用 raster::shapefile
    suppressWarnings({
        ss <- shapefile(shp_path)
    })
    ## 若坐标系不一致，需投影变换（此处仅在不一致时提示，不自动变换）
    if (!is.na(crs(ss)) && !is.na(crs(rs_crop)) && crs(ss)@projargs !=
crs(rs_crop)@projargs) {
        warning("矢量与栅格 CRS 不一致。请先使用 spTransform() 进行投影统一后再运行
mask()。")
    }
    rs_masked <- mask(rs_crop, ss)
    message("掩膜完成（基于 shapefile）。")
} else {
    message("未提供 shapefile，使用矩形 extent 生成的示例多边形进行掩膜演示。")
    ## 用矩形 e 转为 SpatialPolygons 作为示例掩膜
    p <- as(e, "SpatialPolygons")
    crs(p) <- crs(rs_crop)
    rs_masked <- mask(rs_crop, p)
    message("掩膜完成（基于示例多边形）。")
}
## 掩膜结果可视化
save_seq_plot({
    if (!is.null(rs_masked)) {
        if (all(c("NIR","Red","Green") %in% names(rs_masked))) {
            plotRGB(rs_masked[[c("NIR","Red","Green")]],
                    scale = ifelse(maxValue(rs_masked[[1]]) > 1,
maxValue(rs_masked[[1]]), 1),
                    stretch = "hist", main = "掩膜结果-假彩(NIR,Red,Green)")
        } else if (all(c("Red","Green","Blue") %in% names(rs_masked))) {
            plotRGB(rs_masked[[c("Red","Green","Blue")]], scale = 1, stretch =
"hist",
                    main = "掩膜结果-真彩(R,G,B)")
        } else {
            plot(rs_masked[[1]], main = "掩膜结果-首层示意", col = gray(0:100/100))
        }
    } else {
```

```
        plot(rs_crop[[1]], main = "掩膜失败，显示矩形裁剪首层", col =
gray(0:100/100))
    }
}, "掩膜_预览")
## 将裁剪与掩膜结果另存为 GeoTIFF（便于后续复用）
crop_tif  <- next_name("裁剪_栅格", "tif")
mask_tif  <- next_name("掩膜_栅格", "tif")
writeRaster(rs_crop, crop_tif,  overwrite = TRUE)
message("已输出：", crop_tif)
if (!is.null(rs_masked)) {
    writeRaster(rs_masked, mask_tif, overwrite = TRUE)
    message("已输出：", mask_tif)
}
message("1.6 节：空间裁剪与掩膜演示完成。")
## ---- 1.7 文件保存 ----
## 本节将演示将处理后的 Raster 对象保存为 GeoTIFF（以及可选的 .grd），
## 并说明 GeoTIFF 不保留层名称的注意事项。
message("1.7 节：开始文件保存演示...")
## 选择一个要保存的多层对象，优先使用掩膜结果，其次裁剪结果，再次演示栈
rs_to_save <- if (exists("rs_masked") && !is.null(rs_masked)) {
    message("将使用 rs_masked 作为保存对象。")
    rs_masked
} else if (exists("rs_crop")) {
    message("未发现 rs_masked，改用 rs_crop 作为保存对象。")
    rs_crop
} else if (exists("img_demo_stack")) {
    message("未发现 rs_crop，改用 img_demo_stack 作为保存对象。")
    img_demo_stack
} else {
    message("未发现可用对象，临时使用 stack(b5,b4,b3) 作为保存对象。")
    stack(b5, b4, b3)
}
## 保存为 GeoTIFF（注意：GeoTIFF 不保存层名）
tif_out <- next_name("多层保存_GeoTIFF", "tif")
x <- writeRaster(rs_to_save, filename = tif_out, overwrite = TRUE)
message("已输出（GeoTIFF）：", tif_out)
print(x)
## 保存为 raster 的 .grd 格式（可保留层名，但通用性较差）
grd_out <- next_name("多层保存_grd", "grd")
xg <- writeRaster(rs_to_save, filename = grd_out, overwrite = TRUE)
message("已输出（GRD）：", grd_out)
print(xg)
message("1.7 节：文件保存演示完成。")
## ---- 1.8 不同通道关系 ----
```

```r
## 使用 pairs() 可视化多波段之间的相关性，确保 Landsat-8 的超蓝(B1)与蓝(B2)必定被选中
message("1.8 节：开始不同通道关系分析...")
## 明确从原始 img 中按名称提取 L8 波段（包含超蓝 B1）
## 说明：前文已定义 pick_band(img, key)，此处直接复用，避免名称不一致导致的匹配失败
b1 <- pick_band(img, "B1") # Coastal/Aerosol (Ultra-blue)
b2 <- pick_band(img, "B2") # Blue
b3 <- pick_band(img, "B3") # Green
b4 <- pick_band(img, "B4") # Red
b5 <- pick_band(img, "B5") # NIR
## 统一命名，便于 pairs 图标题清晰
names(b1) <- "Coastal(B1)"
names(b2) <- "Blue(B2)"
names(b3) <- "Green(B3)"
names(b4) <- "Red(B4)"
names(b5) <- "NIR(B5)"
## 组 1：超蓝 vs 蓝
rs_pairs1 <- stack(b1, b2)
title1 <- paste0(names(rs_pairs1)[1], " vs ", names(rs_pairs1)[2])
## 组 2：红 vs 近红外（NDVI 相关常用组合）
rs_pairs2 <- stack(b4, b5)
title2 <- paste0(names(rs_pairs2)[1], " vs ", names(rs_pairs2)[2])
## pairs 图保存（两张）
save_seq_plot({
    pairs(rs_pairs1, main = paste0("通道关系：", title1))
}, "通道关系_pairs_组 1")
save_seq_plot({
    pairs(rs_pairs2, main = paste0("通道关系：", title2))
}, "通道关系_pairs_组 2")
message("1.8 节：不同通道关系分析完成。")
## ---- 1.9 提取像素（居民区/农田/水体/道路） ----
## 直接使用预设点；在原始影像范围内可视化并显著标注；仅基于 SR_B1~SR_B7 绘制光谱
message("1.9 节：开始像素提取（居民区/农田/水体/道路，使用预设点）...")
## 1) 明确从原始 img 中按名称提取 L8 必要波段（用于底图与光谱）
b1 <- pick_band(img, "B1") # Ultra-blue
b2 <- pick_band(img, "B2") # Blue
b3 <- pick_band(img, "B3") # Green
b4 <- pick_band(img, "B4") # Red
b5 <- pick_band(img, "B5") # NIR
## 2) 构造用于底图显示的 RGB 组合（原始 img）
s_true_img  <- stack(b4, b3, b2) # R,G,B
s_false_img <- stack(b5, b4, b3) # NIR,R,G
## 自动判定显示 scale（按原始 DN）
```

```
maxDN_img <- max(c(maxValue(b2), maxValue(b3), maxValue(b4), maxValue(b5)), na.rm
= TRUE)
plot_scale_img <- if (is.finite(maxDN_img) && maxDN_img > 1) maxDN_img else 1
## 3) 直接构造三个预设点（确保位于原始 img 范围内）
ext_img <- extent(img)  # 原始文件（img）的空间范围
cx <- (xmin(ext_img) + xmax(ext_img)) / 2
cy <- (ymin(ext_img) + ymax(ext_img)) / 2
dx <- (xmax(ext_img) - xmin(ext_img)) * 0.03
dy <- (ymax(ext_img) - ymin(ext_img)) * 0.03
a_df <- data.frame(
    x = c(114.107493, 114.123371,  114.073097, 114.100777),  # 居民区、农田、水体、
道路
    y = c(31.027838,  31.021896,   31.030039,  31.032147)
)
labels <- c("Residential", "Farmland","Water","Road")
## 4) 底图可视化并显著标注三个点
save_seq_plot({
    ## 优先真彩显示；若失败则退回假彩；再退回单层
    ok <- FALSE
    try({
        plotRGB(s_true_img, scale = plot_scale_img, stretch = "hist",
                main = "1.9 预设点（原始影像）：Residential / Farmland / Water / Road
（真彩）")
        ok <- TRUE
    }, silent = TRUE)
    if (!ok) {
        try({
            plotRGB(s_false_img, scale = plot_scale_img, stretch = "hist",
                    main = "1.9 预设点（原始影像）：Residential / Farmland / Water /
Road（假彩）")
            ok <- TRUE
        }, silent = TRUE)
    }
    if (!ok) {
        plot(b4, main = "1.9 预设点（原始影像）：Residential / Farmland / Water / Road
（单层示意）",
             col = gray(0:100/100))
    }
    ## 更显著的点与标注
    cols <- c("#e7298a", "#1b9e77", "#2550b9", "#d95f02")
    points(a_df$x, a_df$y, pch = 21, bg = cols, col = "black", cex = 2.2, lwd =
1.5)
    ## 为每个点添加文字与引线
    text_offset <- max(res(b4)) * 5  # 根据分辨率拉开标注距离
```

```
    for (i in 1:3) {
        segments(a_df$x[i], a_df$y[i],
                 a_df$x[i] + text_offset, a_df$y[i] + text_offset,
                 col = cols[i], lwd = 2)
        text(a_df$x[i] + text_offset, a_df$y[i] + text_offset,
             labels[i], col = cols[i], cex = 1.2, font = 2)
    }
}, "预设点_底图预览_原始影像")
## 5) 仅选择光谱相关波段进行提取与绘图（SR_B1 ~ SR_B7）
## 根据你提供的层名，筛选 SR_B1~SR_B7
all_names <- names(img)
spec_idx <- grep("^SR_B[1-7]$", all_names, ignore.case = FALSE)
if (length(spec_idx) == 0L) {
    stop("在影像中未发现光谱波段 SR_B1~SR_B7，请检查图层名称。")
}
img_spec <- subset(img, spec_idx)
## 统一命名为 SR_B1..SR_B7 的显式顺序（如存在）
names(img_spec) <- all_names[spec_idx]
## 6) 提取像素的光谱（仅 SR_B1~SR_B7）
spec_mat <- raster::extract(img_spec, a_df)
## 7) 保存数值到 CSV（输出顺序与 img_spec 的层顺序一致）
csv_out <- next_name("像素光谱_SR_B1_B7", "csv")
utils::write.csv(cbind(class = labels, as.data.frame(spec_mat)),
                 file = csv_out, row.names = FALSE, fileEncoding = "UTF-8")
message("已输出像素光谱 CSV（SR_B1~SR_B7）：", csv_out)
## 8) 绘制四类地物的光谱曲线（仅 SR_B1~SR_B7，横轴使用物理含义命名）
save_seq_plot({
    mat <- as.matrix(spec_mat)
    ## 构建层名 -> 物理含义 的映射表（仅对 SR_B1..SR_B7）
    band_map <- c(
        SR_B1 = "Coastal\\Aerosol\n",
        SR_B2 = "Blue\n",
        SR_B3 = "Green\n",
        SR_B4 = "Red\n",
        SR_B5 = "NIR\n",
        SR_B6 = "SWIR1\n",
        SR_B7 = "SWIR2\n"
    )
    orig_names <- names(img_spec)          # e.g., "SR_B1" ... "SR_B7"
    pretty_names <- ifelse(orig_names %in% names(band_map),
                           paste0(band_map[orig_names], " (", orig_names, ")"),
                           # unname(band_map[orig_names]),
                           orig_names)      # 兜底：未匹配就用原名
    colnames(mat) <- pretty_names
```

```
    y_min <- suppressWarnings(min(mat, na.rm = TRUE))
    y_max <- suppressWarnings(max(mat, na.rm = TRUE))
    if (!is.finite(y_min) || !is.finite(y_max)) {
        y_min <- 0; y_max <- 1
    }
    plot(0, 0, type = "n",
         xlim = c(1, ncol(mat)), ylim = c(y_min, y_max),
         xaxt = "n", xlab = "Spectral Bands", ylab = "Reflectance (SR)",
         main = "四类地物光谱（SR_B1~SR_B7）")
    axis(1, at = seq_len(ncol(mat)), labels = colnames(mat), las = 1, cex.axis =
0.9)
    cols <- c("#e7298a", "#1b9e77", "#2550b9", "#d95f02")
    for (i in 1:nrow(mat)) {
        lines(seq_len(ncol(mat)), mat[i,], type = "l", lwd = 3, col = cols[i])
        points(seq_len(ncol(mat)), mat[i,], pch = 16, col = cols[i], cex = 1.2)
    }
    legend("topleft", legend = labels, col = cols, lwd = 3, pch = 16, bty = "n")
}, "四类地物光谱_SR_B1_B7")
## 9) 输出点坐标
pts_csv <- next_name("点坐标_预设", "csv")
utils::write.csv(data.frame(class = labels, a_df), file = pts_csv,
                 row.names = FALSE, fileEncoding = "UTF-8")
message("已输出点坐标 CSV：", pts_csv)
message("1.9 节：像素提取与光谱绘制完成（使用预设点，SR_B1~SR_B7）。")
# ---- 2 遥感影像的数学计算 ----
## ---- 2.1 遥感指数 ----
## 本节将实现包括植被指数（NDVI）在内的各种遥感指数的计算
## 本节将 NDVI 定义为 (NIR - Red)/(NIR + Red)
message("2.1 节：开始计算遥感指数...")
## 载入 RColorBrewer 以改善调色板
if (!requireNamespace("RColorBrewer", quietly = TRUE)) {
    warning("请先安装 RColorBrewer 包：install.packages('RColorBrewer')")
    # 如果包不存在，提供一个兜底的调色板函数
    brewer.pal <- function(n, name) {
        if (name == "RdYlBu") return(colorRampPalette(c("#A50026", "#D73027",
"#F46D43", "#FDAE61", "#FEE090", "#FFFFBF", "#E0F3F8", "#ABD9E9", "#74ADD1",
"#4575B4", "#313695"))(n))
        if (name == "RdYlGn") return(colorRampPalette(c("#A50026", "#D73027",
"#F46D43", "#FDAE61", "#FEE08B", "#FFFFBF", "#D9EF8B", "#A6D96A", "#66C2A5",
"#1A9850", "#006837"))(n))
        return(rev(terrain.colors(n))) # 默认回退
    }
}
## --- 自动阈值函数 (Otsu) ---
```

```r
## 从 2.3 节中提取，用于自动确定 NDWI/NDBI 阈值
otsu_threshold <- function(x, nbins = 256, xmin = -1, xmax = 1){
    v <- values(x); v <- v[is.finite(v)]
    # 钳制数据到理论范围
    v <- pmin(xmax, pmax(xmin, v))
    if (length(v) < nbins) {
        warning("Otsu: 样本点过少，可能返回不可靠阈值。")
        return(as.numeric(quantile(v, 0.5, na.rm=TRUE))) # 返回中位数
    }
    h <- hist(v, breaks = seq(xmin, xmax, length.out = nbins+1), plot = FALSE)
    p <- h$counts / sum(h$counts)
    omega <- cumsum(p)
    mu <- cumsum(p * (h$mids))
    mu_t <- mu[length(mu)]
    # 避免除以 0
    sigma_b2 <- (mu_t*omega - mu)^2 / (omega*(1-omega) + 1e-12)
    # 查找最大方差对应的阈值
    if (all(!is.finite(sigma_b2)) || !any(sigma_b2 > 0)) {
        warning("Otsu: 未能计算有效类间方差，返回中位数。")
        return(as.numeric(quantile(v, 0.5, na.rm=TRUE)))
    }
    t_idx <- which.max(sigma_b2)
    h$mids[t_idx]
}
## --- NDVI (植被指数) ---
## 通用 VI 函数：从多层 Raster 对象中选取第 k 与 i 层计算 (bk - bi) / (bk + bi)
vi <- function(img_stack, k, i) {
    bk <- img_stack[[k]]
    bi <- img_stack[[i]]
    (bk - bi) / (bk + bi)
}
## 选择用于计算 NDVI 的输入数据源与层索引
## - 若存在 img_ref（顺序为: 1=Blue, 2=Green, 3=Red, 4=NIR），则用 4 与 3
## - 否则使用 b5(NIR) 与 b4(Red) 直接计算
if (exists("img_ref")) {
    ndvi <- vi(img_ref, k = 4, i = 3)  # NIR, Red（反射率）
    ndvi_source <- "反射率(img_ref)"
} else {
    ## 兜底确保 b4/b5 存在（此前已定义），若未定义则从 img 中提取
    if (!exists("b4") || !exists("b5")) {
        b4 <- pick_band(img, "B4")
        b5 <- pick_band(img, "B5")
    }
    ndvi <- (b5 - b4) / (b5 + b4)      # 以 DN 近似
```

```
    ndvi_source <- "原始 DN(b5,b4)"
}
## 输出 NDVI GeoTIFF
ndvi_tif_21 <- next_name("NDVI", "tif")
writeRaster(ndvi, ndvi_tif_21, overwrite = TRUE)
message("2.1 节：已输出 NDVI GeoTIFF：", ndvi_tif_21, "（数据来源：", ndvi_source,
"）")
## 绘制 NDVI（修改为 RdYlGn 调色板，绿色代表高植被）
save_seq_plot({
    plot(ndvi,
         col = brewer.pal(11, "RdYlGn"), # 红(低)-黄-绿(高)
         zlim = c(-1, 1),
         main = "Landsat NDVI（2.1 植被指数）\n(高值为绿色)")
}, "NDVI_渲染图")
## --- 其他遥感指数：NDWI/NDBI/SAVI ---
g  <- pick_band(img, "B3")
sw1<- pick_band(img, "B6")
nir<- pick_band(img, "B5")
red<- pick_band(img, "B4")
ndwi <- (g - nir) / (g + nir)            # McFeeters（绿-近红外）
ndbi <- (sw1 - nir) / (sw1 + nir)        # 建筑指数
savi <- (1.5*(nir - red)) / (nir + red + 0.5)  # L=0.5
writeRaster(ndwi, next_name("NDWI","tif"), overwrite=TRUE)
writeRaster(ndbi, next_name("NDBI","tif"), overwrite=TRUE)
writeRaster(savi, next_name("SAVI","tif"), overwrite=TRUE)
## --- 指数可视化（使用新调色板）---
save_seq_plot({
    # 添加 zlim = c(-1, 1)，将颜色固定在理论范围，使水体(高值/蓝色)更突出。
    plot(ndwi,
         col = brewer.pal(11, "RdYlBu"), # 红(低)-黄-蓝(高)
         zlim = c(-1, 1),
         main = "NDWI（水体敏感）\n(高值为蓝色)")
}, "NDWI")
save_seq_plot({
    # 添加 zlim = c(-1, 1)，将颜色固定在理论范围，使建筑(高值/红色)更突出。
    # 注意：rev() 在此是正确的，因为 "RdYlBu" 中 Bu 是高值，反转后 Rd 是高值。
    plot(ndbi,
         col = rev(brewer.pal(11, "RdYlBu")), # 蓝(低)-黄-红(高)
         zlim = c(-1, 1),
         main = "NDBI（建筑敏感）\n(高值为红色)")
}, "NDBI")
save_seq_plot({
    # 添加 zlim = c(-1, 1)，将颜色固定在理论范围。
    plot(savi,
```

```
        col = brewer.pal(11, "RdYlGn"), # 红(低)-黄-绿(高)
        zlim = c(-1, 1),
        main = "SAVI（土壤背景抑制）\n(高值为绿色)")
}, "SAVI")
## --- 掩膜阈值（使用 Otsu 自动阈值） ---
message("正在使用 Otsu 自动计算 NDWI/NDBI 阈值...")
# NDWI 阈值（理论范围 -1 到 1）
ndwi_thr_auto <- otsu_threshold(ndwi, nbins=256, xmin=-1, xmax=1)
message(sprintf("Otsu 自适应 NDWI 阈值（水体 > ...): %.3f", ndwi_thr_auto))
# NDBI 阈值（理论范围 -1 到 1）
ndbi_thr_auto <- otsu_threshold(ndbi, nbins=256, xmin=-1, xmax=1)
message(sprintf("Otsu 自适应 NDBI 阈值（建筑 > ...): %.3f", ndbi_thr_auto))
## 简单水体/建筑掩膜（使用自动阈值）
water_mask <- ndwi > ndwi_thr_auto
urban_mask <- ndbi > ndbi_thr_auto
# 增加掩膜的预览图，确保图例为 0 和 1
save_seq_plot({
    plot(water_mask, main=sprintf("掩膜_水体 (NDWI > %.3f)", ndwi_thr_auto),
        col=c("#FFFFFF", "#0000FF"), zlim=c(0,1), legend=TRUE,
        legend.args=list(text=c('非水体','水体'), side=4, line=2.5))
}, "掩膜_水体_预览")
save_seq_plot({
    plot(urban_mask, main=sprintf("掩膜_建筑 (NDBI > %.3f)", ndbi_thr_auto),
        col=c("#FFFFFF", "#FF0000"), zlim=c(0,1), legend=TRUE,
        legend.args=list(text=c('非建筑','建筑'), side=4, line=2.5))
}, "掩膜_建筑_预览")
# 保存 TIF 文件
writeRaster(water_mask, next_name("掩膜_水体","tif"), overwrite = TRUE)
writeRaster(urban_mask, next_name("掩膜_建筑","tif"), overwrite = TRUE)
message("2.1 节：指数计算、可视化与自动掩膜完成。")
## ---- 2.2 频率直方图（NDVI/NDWI/NDBI/SAVI 分布） ----
## 说明：本节基于 PDF 2.2，将 NDVI 的分布绘制为直方图
## 使用 2x2 画布，显示所有四个主要指数的直方图
message("2.2 节：开始绘制 NDVI, NDWI, NDBI, SAVI 频率直方图...")
save_seq_plot({
    # 设置 2x2 绘图布局
    op <- par(mfrow = c(2, 2), mar = c(5, 4, 4, 2) + 0.1) # 恢复默认 mar 并设置 mfrow
    on.exit(par(op), add = TRUE) # 确保在函数退出时恢复布局
    # 增加 breaks = 50 以提高精细度。
    # 1. NDVI 直方图
    hist(ndvi,
        main  = "Distribution of NDVI values",
        xlab  = "NDVI",
        ylab  = "Frequency",
```

```
        col   = "wheat",
        breaks = 50)
    # 2. NDWI 直方图
    hist(ndwi,
        main  = "Distribution of NDWI values",
        xlab  = "NDWI (Water)",
        ylab  = "Frequency",
        col   = "lightblue",
        breaks = 50)
    # 3. NDBI 直方图
    hist(ndbi,
        main  = "Distribution of NDBI values",
        xlab  = "NDBI (Built-up)",
        ylab  = "Frequency",
        col   = "lightcoral",
        breaks = 50)
    # 4. SAVI 直方图
    hist(savi,
        main  = "Distribution of SAVI values",
        xlab  = "SAVI (Vegetation)",
        ylab  = "Frequency",
        col   = "lightgreen",
        breaks = 50)
}, "Indices_Histograms_2x2") # 修改了保存标签
message("2.2 节：四合一指数直方图已保存。")
## ---- 2.3 分类临界值（基于 NDVI，2023-07-27 夏季） ----
## 目标：
## 1) 通过阈值将 NDVI 分为“植被/非植被”（NaN 掩膜非植被，便于突出植被）
## 2) 进一步细分多个 NDVI 等级（更贴合夏季 NDVI 较高的分布）
## 3) 输出分类结果 GeoTIFF 与配图
message("2.3 节：开始基于 NDVI 的分类与阈值分割（2023-07-27 夏季）...")
## 保障：若 2.1 节未定义 ndvi，这里回退计算一次
if (!exists("ndvi")) {
    message("未发现 ndvi 对象，按 b5/b4 兜底即时计算 NDVI。")
    if (!exists("b4") || !exists("b5")) {
        b4 <- pick_band(img, "B4")
        b5 <- pick_band(img, "B5")
    }
    ndvi <- (b5 - b4) / (b5 + b4)
}
## Otsu 阈值（针对NDVI）
otsu_threshold <- function(x, nbins = 256, xmin = -0.5, xmax = 1){
    v <- values(x); v <- v[is.finite(v)]
    v <- pmin(xmax, pmax(xmin, v))
```

```
    h <- hist(v, breaks = seq(xmin, xmax, length.out = nbins+1), plot = FALSE)
    p <- h$counts / sum(h$counts)
    omega <- cumsum(p)
    mu <- cumsum(p * (h$mids))
    mu_t <- mu[length(mu)]
    sigma_b2 <- (mu_t*omega - mu)^2 / (omega*(1-omega) + 1e-12)
    t_idx <- which.max(sigma_b2)
    h$mids[t_idx]
}
ndvi_thr_auto <- otsu_threshold(ndvi)
message(sprintf("Otsu 自适应 NDVI 阈值：%.3f", ndvi_thr_auto))
# ndvi_thr <- 0.20 # 简单阈值：植被掩膜（夏季设定 NDVI > 0.20）
ndvi_thr <- if (is.finite(ndvi_thr_auto)) ndvi_thr_auto else 0.20 # Otsu 自适应阈值选择
veg_mask <- reclassify(ndvi, cbind(-Inf, ndvi_thr, NA))  # <=阈值 置为 NA，>阈值 保留
save_seq_plot({
    plot(veg_mask,
         main = sprintf("2023-07-27 夏季 植被分布（NDVI > %.2f）", ndvi_thr),
         col = rev(terrain.colors(10)))
}, "NDVI_阈值_植被掩膜_20230727")
veg_mask_tif <- next_name("NDVI_阈值_植被掩膜_20230727", "tif")
writeRaster(veg_mask, veg_mask_tif, overwrite = TRUE)
message("已输出 NDVI 植被掩膜 GeoTIFF：", veg_mask_tif)
## 2.3.2 多级分类（夏季区间，更贴合直方图峰值在 0.6~0.85）
## 区间（左开右闭）：
##  (-Inf,0.20] -> 0（非植被，置 NA 以便可视化透明）
##  (0.20,0.40] -> 1（低度植被/稀疏）
##  (0.40,0.60] -> 2（中等植被）
##  (0.60,0.80] -> 3（高植被）
##  (0.80, Inf) -> 4（极高植被/茂盛冠层）
## 可视化时 0 作为 NA，不参与渲染
class_mat <- matrix(c(
    -Inf, 0.20, NA,
    0.20, 0.40, 1,
    0.40, 0.60, 2,
    0.60, 0.80, 3,
    0.80,  Inf, 4
), ncol = 3, byrow = TRUE)
ndvi_class <- reclassify(ndvi, class_mat)
## 自定义调色板（由低到高：黄-淡绿-绿-深绿）
pal_class <- c("#fee08b", "#a6d96a", "#1a9850", "#006837")
save_seq_plot({
    plot(ndvi_class, col = pal_class, zlim = c(1,4), legend = TRUE,
```

```
        main = "NDVI 多级分类（夏季 2023-07-27）\n 0.20–0.40:低 | 0.40–0.60:中 |
0.60–0.80:高 | >0.80:极高")
}, "NDVI_多级分类_20230727")
ndvi_class_tif <- next_name("NDVI_多级分类_20230727", "tif")
writeRaster(ndvi_class, ndvi_class_tif, overwrite = TRUE)
message("已输出 NDVI 多级分类 GeoTIFF：", ndvi_class_tif)
## 2.3.3 在真彩底图上叠加分类结果，直观核对
## 优先使用 1.4 节的真彩 s_true（B4,B3,B2），否则临时组装
if (!exists("s_true")) {
    if (!exists("b2") || !exists("b3") || !exists("b4")) {
        b2 <- pick_band(img, "B2"); b3 <- pick_band(img, "B3"); b4 <- pick_band(img,
"B4")
    }
    s_true <- stack(b4, b3, b2)
}
## 自动 scale 判定：若是 16bit DN 则取其最大值，否则 1（反射率）
maxDN_overlay <- max(c(maxValue(s_true[[1]]), maxValue(s_true[[2]]),
maxValue(s_true[[3]])), na.rm = TRUE)
scale_overlay <- if (is.finite(maxDN_overlay) && maxDN_overlay > 1) maxDN_overlay
else 1
## 叠加预览 1：植被掩膜（半透明）
save_seq_plot({
    plotRGB(s_true, scale = scale_overlay, stretch = "hist",
            main = "叠加预览：真彩 + NDVI 植被掩膜(> 0.20) | 2023-07-27")
    plot(veg_mask, add = TRUE, legend = FALSE, alpha = 0.45)
}, "叠加预览_真彩_植被掩膜_20230727")
## 叠加预览 2：多级分类（隐藏 0 类；半透明渲染 1~4）
save_seq_plot({
    plotRGB(s_true, scale = scale_overlay, stretch = "hist",
            main = "叠加预览：真彩 + NDVI 多级分类（2023-07-27）")
    ## 仅渲染 1:4，0/NA 不显示
    plot(ndvi_class, add = TRUE, legend = FALSE, col = pal_class, alpha = 0.45,
zlim = c(1,4))
}, "叠加预览_真彩_多级分类_20230727")
message("2.3 节：NDVI 阈值分类与叠加预览完成（2023-07-27 夏季）。")
# ---- 3 地物分类 ----
## ---- 3.1 无监督分类（K-means，整景影像） ----
## 目标：
## - 对整景“原始 img”进行无监督聚类（K-means），而不是仅对裁剪/掩膜区域
## - 自动优先选择用于做类的关键波段（优先 SR_B2~SR_B7，其次 B2~B7/Blue~SWIR2 等）
## - 标准化各波段，样本用于 K 值经验法评估，随后对整景影像聚类
## - 对聚类结果执行后处理（多数滤波），以消除碎斑
## - 输出分类 GeoTIFF、分类图、真彩图叠加与类别统计
message("3.1 提示：开始无监督遥感分类（K-means）...")
```

```
suppressPackageStartupMessages({
    library(raster)
    library(sp)
})
## 3.1.1 固定输入数据源为整景原始影像 img
## 说明：不再使用 rs_masked 或 rs_crop，确保针对整景影像进行分类
if (!exists("img") || !inherits(img, c("RasterStack","RasterBrick"))) {
    stop("3.1：未发现整景影像对象 'img'，请确保上文已成功读取 stack(tif_path)。")
}
rs_in <- img
message("3.1：已确定使用整景原始影像 'img' 作为分类输入。")
## 3.1.2 自动优先选择用于分类的关键波段（增强稳健性）
nm <- names(rs_in)
pick_by_pattern <- function(x, patterns) {
    idx_all <- integer(0)
    nms <- names(x)
    for (pt in patterns) {
        idx <- grep(pt, nms, ignore.case = TRUE)
        idx_all <- c(idx_all, idx)
    }
    idx_all <- unique(idx_all)
    if (length(idx_all) == 0) return(NULL)
    subset(x, idx_all)
}
## A) 优先：严格匹配 SR_B2~SR_B7
idx_sr <- grep("^SR_B([2-7])$", nm, ignore.case = FALSE)
if (length(idx_sr) >= 3) {
    rs_cls <- subset(rs_in, idx_sr)
    message("3.1:使用 SR_B2~SR_B7 作为特征(匹配到:",paste(names(rs_cls),collapse
= ","), "）。")
} else {
    message("3.1：未充分匹配 SR_B2~SR_B7，尝试宽松命名匹配。")
    ## B) 宽松命名集合（避开 B1/B8），意向顺序：B2..B7
    patterns_ordered <- c(
        "(^|[_-])b2($|[_-])|(^blue$)|(^blue\\b)",
        "(^|[_-])b3($|[_-])|(^green$)|(^green\\b)",
        "(^|[_-])b4($|[_-])|(^red$)|(^red\\b)",
        "(^|[_-])b5($|[_-])|(^nir$)|(^nir\\b)",
        "(^|[_-])b6($|[_-])|(^swir1$)|(^swir_?1\\b)",
        "(^|[_-])b7($|[_-])|(^swir2$)|(^swir_?2\\b)"
    )
    rs_try <- pick_by_pattern(rs_in, patterns_ordered)
    if (!is.null(rs_try) && nlayers(rs_try) >= 3) {
        rs_cls <- rs_try
```

```
        message("3.1：使用宽松匹配到的波段：", paste(names(rs_cls), collapse = ","))
    } else {
        message("3.1：宽松匹配仍不足，尝试从上文变量收集候选波段或底限选取。")
        ## C) 从上文变量收集（若用户在前文已定义 b2..b7 等）
        cand_list <- list()
        add_if_ok <- function(obj) {
            if (exists(obj) && inherits(get(obj),
c("RasterLayer","RasterBrick","RasterStack"))) {
                r <- get(obj)
                if (inherits(r, "RasterLayer")) {
                    cand_list[[obj]] <<- r
                } else {
                    for (i in 1:nlayers(r)) {
                        cand_list[[ paste0(obj, "_", names(r)[i]) ]] <<- r[[i]]
                    }
                }
            }
        }
        for (nmv in
c("b2","b3","b4","b5","b6","b7","blue","green","red","nir","swir1","swir2")) {
            add_if_ok(nmv)
        }
        # 同时尝试自 img 中按名称提取 B2..B7
        try({
            for (key in c("B2","B3","B4","B5","B6","B7")) {
                i <- grep(paste0("(^|[_-])", tolower(key), "($|[_-])"),
tolower(names(rs_in)))
                if (length(i) > 0) cand_list[[key]] <- rs_in[[i[1]]]
            }
        }, silent = TRUE)
        if (length(cand_list) >= 3) {
            rs_cls <- stack(cand_list)
            unq <- !duplicated(names(rs_cls))
            rs_cls <- subset(rs_cls, which(unq))
            if (nlayers(rs_cls) > 6) rs_cls <- rs_cls[[1:6]]
            if (nlayers(rs_cls) >= 3) {
                message("3.1：使用上文收集的候选波段：", paste(names(rs_cls),
collapse = ","))
            } else {
                rs_cls <- NULL
            }
        } else {
            rs_cls <- NULL
        }
```

```
        ## D) 最后底限：自 img 中筛选数值型波段（剔除质量波段），取前 3~6 层
        if (is.null(rs_cls) || nlayers(rs_cls) < 3) {
            message("3.1：继续底限策略—从整景影像中自动筛选数值波段。")
            qa_pat <- "(qa|quality|cloud|pixel_qa|radsat|saturation|mask)"
            keep_idx <- which(!grepl(qa_pat, tolower(names(rs_in))))
            rs_tmp <- if (length(keep_idx) >= 3) subset(rs_in, keep_idx) else rs_in
            good <- logical(nlayers(rs_tmp))
            for (i in 1:nlayers(rs_tmp)) {
                v <- values(rs_tmp[[i]])
                v <- v[is.finite(v)]
                good[i] <- length(unique(v)) > 1
            }
            if (any(good)) rs_tmp <- subset(rs_tmp, which(good))
            if (nlayers(rs_tmp) >= 3) {
                take <- min(6, nlayers(rs_tmp))
                rs_cls <- rs_tmp[[1:take]]
                message("3.1：使用底限筛选的数值波段：", paste(names(rs_cls),
collapse = ","))
            }
        }
    }
}
if (is.null(rs_cls) || nlayers(rs_cls) < 3) {
    stop("3.1：未能找到足够的关键波段用于无监督分类（至少需要 3 层）。请检查输入影像波
段命名。")
}
## 3.1.3 对有效像元对齐掩膜（避免 NA 不一致）
rs_cls <- mask(rs_cls, rs_cls[[1]])
## 3.1.4 波段标准化（逐层 z-score）
scale_stack <- function(stk) {
    b <- vector("list", nlayers(stk))
    for (i in seq_len(nlayers(stk))) {
        vi <- values(stk[[i]])
        m <- mean(vi, na.rm = TRUE)
        s <- stats::sd(vi, na.rm = TRUE)
        if (!is.finite(s) || s == 0) s <- 1
        b[[i]] <- (stk[[i]] - m) / s
    }
    stack(b)
}
rs_std <- scale_stack(rs_cls)
names(rs_std) <- paste0(names(rs_cls), "_z")
## 3.1.5 抽样用于 K 值评估
set.seed(20251031)
```

```
n_total <- ncell(rs_std)
n_sample <- min(50000, max(2000, round(n_total * 0.02)))
smp <- sampleRandom(rs_std, size = n_sample, na.rm = TRUE, as.data.frame = TRUE,
sp = FALSE)
## 3.1.6 K 值快速评估
k_candidates <- 2:8
wss <- numeric(length(k_candidates))
message("3.1：评估 K 值（经验法）...")
for (i in seq_along(k_candidates)) {
    k <- k_candidates[i]
    km <- kmeans(smp, centers = k, nstart = 10, iter.max = 100)
    wss[i] <- km$tot.withinss
    message(sprintf(" - K=%d, tot.withinss=%.3e", k, wss[i]))
}
save_seq_plot({
    plot(k_candidates, wss, type = "b", pch = 19,
         xlab = "K (clusters)", ylab = "Total within-cluster sum of squares",
         main = "K-means 经验法评估（肘部法）")
}, "Kmeans_经验评估")
## 3.1.7 确定最终 K（根据肘部法得到的折线图选择并修改下面这一行代码）
K_final <- 4
## 3.1.8 使用样本初始化中心
km_init <- kmeans(smp, centers = K_final, nstart = 10, iter.max = 200)
centers_init <- km_init$centers
## 3.1.9 对整景影像执行 K-means（使用标准化后的全图）
message(sprintf("3.1：对整景影像执行 K-means（K=%d）...", K_final))
vals <- getValues(rs_std)
ok <- complete.cases(vals)
cls_all <- rep(NA_integer_, nrow(vals))
if (sum(ok) == 0) stop("3.1：有效像元为 0，无法分类。")
km_full <- kmeans(vals[ok, , drop = FALSE], centers = centers_init, iter.max = 200)
cls_all[ok] <- km_full$cluster
## 3.1.10 写出分类栅格（原始结果）
cls_r_raw <- rs_std[[1]]
values(cls_r_raw) <- cls_all
levels_df <- data.frame(ID = 1:K_final,
                        Class = paste0("Class_", 1:K_final),
                        stringsAsFactors = FALSE)
# 保存原始 K-means 结果
writeRaster(cls_r_raw, next_name("Kmeans_分类_原始", "tif"), overwrite = TRUE)
if (!exists("pal_k")) {
    pal_k <-
c("#e31a1c","#33a02c","#1f78b4","#ff7f00","#6a3d9a","#a6cee3","#b2df8a","#fb9a
99")[seq_len(K_final)]
```

```
}
# 使用 save_seq_plot 函数来保存 PNG
save_seq_plot({
    plot(cls_r_raw, col = pal_k[1:K_final], legend = FALSE,
         main = sprintf("K-means 原始分类结果（K = %d）", K_final))
    legend("topright", legend = levels_df$Class, fill = pal_k[1:K_final], bty =
"n", cex = 0.9)
}, "Kmeans_分类图_原始") # 标签名
message("3.1：已输出 Kmeans 原始分类 PNG 预览图。")
## 3.1.11 K-means 后处理（多数/众数滤波）
message("3.1：K-means 原始结果已生成，开始执行后处理（3x3 多数/众数滤波）...")
# 定义 3x3 邻域窗口
w <- matrix(1, nrow=3, ncol=3)
# 使用 focal() 和 modal()（众数）函数进行滤波
# modal 是 raster 包中用于 focal 的内置函数
# na.rm = TRUE 保证邻域内的 NA 值被忽略
# pad = FALSE 避免在图像边缘填充（结果边缘会是 NA）
cls_r_post <- focal(cls_r_raw, w = w, fun = modal, na.rm = TRUE, pad = FALSE)
# 恢复掩膜：
# 确保后处理后的数据覆盖范围与原始一致，避免滤波导致数据区域“增长”到 NA 区域。
# 同时也保证了原始的 NA 边界不会产生“缝隙”。
cls_r_post <- mask(cls_r_post, cls_r_raw)
message("3.1：后处理完成。")
## 3.1.12 写出（后处理后的）分类栅格
cls_tif <- next_name("Kmeans_分类_后处理", "tif") # 更新文件名
writeRaster(cls_r_post, cls_tif, overwrite = TRUE) # 保存 post 结果
message("3.1：已输出 Kmeans 分类 GeoTIFF（后处理）：", cls_tif)
## 3.1.13 分类结果可视化（后处理）
pal_k <-
c("#e31a1c","#33a02c","#1f78b4","#ff7f00","#6a3d9a","#a6cee3","#b2df8a","#fb9a
99")[seq_len(K_final)]
save_seq_plot({
    # 绘制后处理结果
    plot(cls_r_post, col = pal_k[1:K_final], legend = FALSE,
         main = sprintf("K-means 无监督分类（后处理 K = %d）", K_final)) # 更新标题
    legend("topright", legend = levels_df$Class, fill = pal_k[1:K_final], bty =
"n", cex = 0.9)
}, "Kmeans_分类图_后处理") # 更新保存名
## 3.1.14 真彩图叠加（后处理结果）
if (!exists("s_true")) {
    if (!exists("b2") || !exists("b3") || !exists("b4")) {
        b2 <- try(pick_band(img, "B2"), silent = TRUE)
        b3 <- try(pick_band(img, "B3"), silent = TRUE)
        b4 <- try(pick_band(img, "B4"), silent = TRUE)
```

```
    }
    if (inherits(b2, "RasterLayer") && inherits(b3, "RasterLayer") && inherits(b4,
"RasterLayer")) {
        s_true <- stack(b4, b3, b2)
    }
}
if (exists("s_true")) {
    maxDN_overlay2 <- max(c(maxValue(s_true[[1]]), maxValue(s_true[[2]]),
maxValue(s_true[[3]])), na.rm = TRUE)
    scale_overlay2 <- if (is.finite(maxDN_overlay2) && maxDN_overlay2 > 1)
maxDN_overlay2 else 1
    save_seq_plot({
        plotRGB(s_true, scale = scale_overlay2, stretch = "hist",
                main = sprintf("叠加预览：真彩 + K-means 分类（后处理，K=%d）", K_final))
# 更新标题
        # 叠加后处理结果
        plot(cls_r_post, add = TRUE, legend = FALSE, col =
adjustcolor(pal_k[1:K_final], alpha.f = 0.45))
    }, "Kmeans_叠加_真彩_后处理") # 更新保存名
} else {
    message("3.1：未能构建真彩底图，跳过叠加预览。")
}
## 3.1.15 简要统计（后处理结果）：各类别像元计数与占比
# 统计后处理结果
tab <- table(values(cls_r_post))
prop <- round(100 * tab / sum(tab, na.rm = TRUE), 2)
message("3.1：类别像元统计（后处理结果，像元数 / 占比%）：") # 更新消息
for (i in 1:K_final) {
    # 必须按名称（类别 ID）索引，而不是按位置索引
    k_name <- as.character(i)
    cnt <- if (k_name %in% names(tab)) as.integer(tab[k_name]) else 0L
    pr  <- if (k_name %in% names(prop)) as.numeric(prop[k_name]) else 0
    message(sprintf(" - Class_%d: %d / %.2f%%", i, cnt, pr))
}
## 3.1.16 输出类别统计到 CSV（后处理结果）
csv_kmeans_stat <- next_name("Kmeans_类别统计_后处理", "csv") # 更新文件名
# 使用 sapply 确保安全地从 table 中提取 1:K_final 的值，即使某些类在后处理中被完全消
除
df_stat <- data.frame(
    class = paste0("Class_", 1:K_final),
    count = sapply(1:K_final, function(k) {
        c <- tab[as.character(k)]; if(is.na(c)) 0 else as.integer(c)
    }),
    percent = sapply(1:K_final, function(k) {
```

```
        p <- prop[as.character(k)]; if(is.na(p)) 0 else as.numeric(p)
    })
)
utils::write.csv(df_stat, file = csv_kmeans_stat, row.names = FALSE, fileEncoding
= "UTF-8")
message("3.1：已输出 Kmeans 类别统计（后处理）：", csv_kmeans_stat)
message("3.1 提示：无监督地物分类（含后处理）完成。")
## ---- 3.2 监督分类 ----
message("3.2 节：开始监督分类（基于 data/supervision_classification 中的 shp 训练
样区）...")
## 1. 选择特征栅格
rs_train_base <- NULL
if (exists("img")) {
    rs_train_base <- img
    message("使用 img 作为底图。")
} else stop("未找到可用栅格。")
if (exists("rs_cls") && inherits(rs_cls, c("RasterStack","RasterBrick"))) {
    rs_feat <- rs_cls
    message("3.2：使用 rs_cls 作为特征栈。")
} else {
    rs_feat <- rs_train_base
}
## 2. 训练样区路径
train_dir <- "data/supervision_classification"
if (!dir.exists(train_dir)) stop("未找到训练矢量文件夹
data/supervision_classification")
subdirs <- list.dirs(train_dir, full.names = TRUE, recursive = FALSE)
shp_files <- list()
for (sd in subdirs) {
    shp_candidates <- list.files(sd, pattern = "\\.shp$", full.names = TRUE,
ignore.case = TRUE)
    if (length(shp_candidates) > 0) shp_files[[basename(sd)]] <- shp_candidates[1]
}
if (length(shp_files) == 0) stop("未找到任何 .shp 文件。")
## 3. 提取训练样本
training_list <- list()
for (cls_name in names(shp_files)) {
    shp_path <- shp_files[[cls_name]]
    message("读取类别：", cls_name, " --> ", basename(shp_path))
    # 使用 sf 读取（更鲁棒）
    shp_sf <- try(sf::st_read(shp_path, quiet = TRUE), silent = TRUE)
    if (inherits(shp_sf, "try-error") || nrow(shp_sf) == 0) {
        warning("跳过空或错误文件：", shp_path)
        next
```

```
    }
    # 若 CRS 不同则重投影
    if (!is.na(st_crs(shp_sf)) && !is.na(crs(rs_feat))) {
        shp_sf <- st_transform(shp_sf, crs = st_crs(rs_feat))
    }
    # 转为 Spatial 以兼容 raster::extract
    shp_sp <- as(shp_sf, "Spatial")
    # 确保几何合法
    if (is.null(shp_sp@polygons) && is.null(shp_sp@coords)) {
        warning("矢量无几何数据：", shp_path)
        next
    }
    ex <- try(raster::extract(rs_feat, shp_sp), silent = TRUE)
    if (inherits(ex, "try-error") || is.null(ex)) {
        warning("提取失败：", shp_path)
        next
    }
    if (is.list(ex)) ex <- do.call(rbind, ex)
    if (is.null(ex) || nrow(as.data.frame(ex)) == 0) {
        warning("类别 ", cls_name, " 未提取到任何像元（可能未与影像重叠）。")
        next
    }
    message("类别 ", cls_name, " 提取像元：", nrow(ex))
    if (is.list(ex)) ex <- do.call(rbind, ex)
    ex <- as.data.frame(ex)
    ex <- ex[complete.cases(ex), ]
    if (nrow(ex) == 0) next
    ex$class <- cls_name
    training_list[[cls_name]] <- ex
    message(" -> 提取像元：", nrow(ex))
}
if (length(training_list) == 0) stop("未能提取任何样本，请检查矢量。")
## 汇总每个类别像元数
sample_summary <- data.frame(
    类别 = names(training_list),
    像元数 = sapply(training_list, nrow)
)
message("各类别样本统计：")
print(sample_summary)
## 合并训练数据
training_df <- do.call(rbind, training_list)
message("训练样本总计：", nrow(training_df))
# 4. 随机森林分类
set.seed(20251101)
```

```r
idx <- sample(nrow(training_df))
train_df <- training_df[idx, ]
n_train <- round(0.7 * nrow(train_df))
train_x <- train_df[1:n_train, setdiff(names(train_df), "class")]
train_y <- factor(train_df[1:n_train, "class"])
test_x <- train_df[(n_train+1):nrow(train_df), setdiff(names(train_df),
"class")]
test_y <- factor(train_df[(n_train+1):nrow(train_df), "class"])
rf_model <- randomForest(x = train_x, y = train_y, ntree = 500, importance = TRUE)
message("OOB 错误率：", round(100 * tail(rf_model$err.rate[, "OOB"], 1), 2), "%")
print(rf_model$confusion)
pred_test <- predict(rf_model, test_x)
acc <- sum(pred_test == test_y) / length(test_y)
message(sprintf("验证集总体精度：%.2f%%", 100 * acc))
## 5. 整图预测
pred_tif <- next_name("Supervised_RF_分类_原始", "tif")
pred_raster <- raster::predict(rs_feat, rf_model, progress = "text", filename =
pred_tif,
                               overwrite = TRUE, type = "response")
message("输出原始分类栅格：", pred_tif)
## 6. 颜色映射
classes <- sort(unique(training_df$class))
pred_raster <- as.factor(pred_raster)
levels(pred_raster)[[1]] <- data.frame(ID = seq_along(classes), class = classes)
pal_sup <- setNames(colorRampPalette(
    c("#e31a1c","#33a02c","#1f78b4"))(length(classes)), classes)
save_seq_plot({
    plot(pred_raster, col = pal_sup, main = sprintf("监督分类结果（RF 原始，%d 类）
", length(classes)),
         axes = FALSE, box = FALSE, legend = FALSE)
    legend("topleft", legend = classes, fill = pal_sup, bty = "n", cex = 0.9)
}, "监督分类_RF_原始")
## 7. 后处理（3×3 众数滤波）
w <- matrix(1, 3, 3)
pred_r_post <- focal(pred_raster, w = w, fun = modal, na.rm = TRUE)
pred_r_post <- mask(pred_r_post, pred_raster)
pred_tif_post <- next_name("Supervised_RF_分类_后处理", "tif")
writeRaster(pred_r_post, pred_tif_post, overwrite = TRUE)
message("输出后处理分类栅格：", pred_tif_post)
save_seq_plot({
    plot(pred_r_post, col = pal_sup, main = sprintf("监督分类结果（RF 后处理，%d 类）
", length(classes)),
         axes = FALSE, box = FALSE, legend = FALSE)
    legend("topleft", legend = classes, fill = pal_sup, bty = "n", cex = 0.9)
```

```
}, "监督分类_RF_后处理")
## 8. 保存模型与样本
write.csv(training_df, next_name("监督分类_训练样本", "csv"), row.names = FALSE)
saveRDS(rf_model, next_name("监督分类_RF_model", "rds"))
message("3.2 节：监督分类全部完成。")
## ---- 结尾部分 ----
## 关闭日志
sink()
message("日志已关闭。")
```

② 4.1_Change_Detection.R

```
# ---- 0 环境设置与包加载 ----
setwd("E:/code/r_code/20251031_Remote_Sensing_Image_Processing_Using_R") # 设置工作路径
library(raster)
library(RColorBrewer)
library(utils)
message("环境设置与包加载完成。")
# ---- 4 变化检测 ----
## ---- 4.1 基于遥感指数的变化检测 ----
## 1. 输出目录准备
out_dir <- "res_change_detection"
if (!dir.exists(out_dir)) {
  message(sprintf("创建结果目录：%s", out_dir))
  dir.create(out_dir, recursive = TRUE)
} else {
  message(sprintf("结果目录 '%s' 已存在，将尝试跳过已存在文件的计算。", out_dir))
}
message(sprintf("结果输出目录已准备好：%s", normalizePath(out_dir)))
## 2. 辅助函数
message_line <- function(...) cat(paste0(format(Sys.time(), "%H:%M:%S "), ... , "\n"))
next_name <- function(prefix = "Plot", ext = "png", dir = out_dir) {
  i <- 1
  repeat {
    fn <- file.path(dir, sprintf("%s_%03d.%s", prefix, i, ext))
    if (!file.exists(fn)) return(fn)
    i <- i + 1
  }
}
save_seq_plot <- function(expr, prefix = "Plot", width = 10, height = 8, res = 200)
{
  fn <- next_name(prefix, "png")
  png(fn, width = width, height = height, units = "in", res = res)
```

```
  on.exit(dev.off())
  force(expr)
  message_line("保存图像: ", fn)
}
extract_date_from_name <- function(fn){
  b <- basename(fn)
  m <- regexpr("\\d{8}", b)
  if (m[1] == -1) return(NA)
  ds <- regmatches(b, m)
  as.Date(ds, "%Y%m%d")
}
safe_pick <- function(x, key){
  nm <- tolower(names(x))
  idx <- grep(key, nm, ignore.case = TRUE)
  if (length(idx) == 0 && key == "B6") {
    idx <- grep("swir1", nm, ignore.case = TRUE)
  }
  if (length(idx) > 0) return(x[[idx[1]]])
  stop(sprintf("找不到波段 %s", key))
}
## 3. 输入文件（查找所有 tif）
img_dir <- "GEE_L8_Yangdian_image"
tif_files_all <- list.files(img_dir, pattern = "\\.tif$", full.names = TRUE,
ignore.case = TRUE)
if (length(tif_files_all) == 0) stop("未找到任何 tif 文件，请检查
GEE_L8_Yangdian_image 文件夹。")
dates_all <- sapply(tif_files_all, extract_date_from_name)
valid_idx <- which(!is.na(dates_all))
if (length(valid_idx) == 0) stop("未能从任何文件名中提取到日期（YYYYMMDD）。")
tif_files_all <- tif_files_all[valid_idx]
dates_all <- as.Date(dates_all[valid_idx])
ord_all <- order(dates_all)
tif_files_all <- tif_files_all[ord_all]
dates_all <- dates_all[ord_all]
message_line(sprintf("发现 %d 份可解析日期的影像（总计），时间范围：%s → %s",
                     length(tif_files_all), min(dates_all), max(dates_all)))
## 4. 在每年（2015-2023）选择夏季与冬季代表影像
years <- 2015:2023
selected_idx <- integer(0)
for (y in years) {
  target_summer <- as.Date(sprintf("%d-07-01", y))
  target_winter <- as.Date(sprintf("%d-01-15", y))

  diffs_summer <- abs(as.numeric(dates_all - target_summer))
```

```
  diffs_winter <- abs(as.numeric(dates_all - target_winter))

  pick_s <- which.min(diffs_summer)
  pick_w <- which.min(diffs_winter)

  selected_idx <- unique(c(selected_idx, pick_s, pick_w))
}
if (length(selected_idx) < 2) stop("选取到的代表影像不足 2 张，无法进行变化检测。请检查原始影像或年份范围设置。")
tif_files <- tif_files_all[selected_idx]
dates <- dates_all[selected_idx]
ord <- order(dates)
tif_files <- tif_files[ord]
dates <- dates[ord]
date_strs <- format(dates, "%Y%m%d")
message_line(sprintf("选定 %d 张代表影像用于变化检测，时间范围：%s → %s",
                     length(tif_files), min(dates), max(dates)))
message_line("选定的文件（按时间排序）：")
for (i in seq_along(tif_files)) message_line(sprintf("  %s  (%s)",
basename(tif_files[i]), date_strs[i]))
## 5. 指数计算函数（保持原有实现）
compute_indices_from_stack <- function(rs_or_path){
  if (is.character(rs_or_path)) {
    s <- try(stack(rs_or_path), silent = TRUE)
    if (inherits(s, "try-error")) stop("无法读取: ", rs_or_path)
  } else {
    s <- rs_or_path
  }
  red <- safe_pick(s, "B4")
  nir <- safe_pick(s, "B5")
  g   <- safe_pick(s, "B3")

  sw1 <- tryCatch(safe_pick(s, "SR_B6"), error = function(e) NULL)

  ndvi <- (nir - red) / (nir + red)
  ndwi <- (g - nir) / (g + nir)
  if (!is.null(sw1)) {
    ndbi <- (sw1 - nir) / (sw1 + nir)
  } else {
    ndbi <- nir; ndbi[] <- NA
    warning("未找到 SWIR1 波段，NDBI 将被设置为 NA（全 NA 图层）。")
  }
  savi <- (1.5 * (nir - red)) / (nir + red + 0.5)
```

```
  list(NDVI = ndvi, NDWI = ndwi, NDBI = ndbi, SAVI = savi)
}
## 6. 批量计算或加载 指数影像
# 检查所有期望的输出文件是否存在
all_indices_exist <- TRUE
expected_files_list <- list() # 用来存储期望的文件名
indices_to_check <- c("NDVI", "NDWI", "NDBI", "SAVI")
# 构建期望文件路径列表
for (ds in date_strs) {
  for (idx_name in indices_to_check) {
    fn <- file.path(out_dir, sprintf("%s_%s.tif", idx_name, ds))
    expected_files_list[[idx_name]][[ds]] <- fn
    # 检查文件是否存在
    if (!file.exists(fn)) {
      all_indices_exist <- FALSE
    }
  }
}
# 初始化列表
ndvi_list <- list(); ndwi_list <- list(); ndbi_list <- list(); savi_list <- list()
if (all_indices_exist) {
  message_line("检测到所有时相的指数 TIF 文件均已存在，跳过计算，开始加载...")

  for (ds in date_strs) {
    tryCatch({
      # 从 expected_files_list 加载
      ndvi_list[[ds]] <- raster(expected_files_list[["NDVI"]][[ds]])
      ndwi_list[[ds]] <- raster(expected_files_list[["NDWI"]][[ds]])
      ndbi_list[[ds]] <- raster(expected_files_list[["NDBI"]][[ds]])
      savi_list[[ds]] <- raster(expected_files_list[["SAVI"]][[ds]])
      message_line(sprintf("  已加载 %s 的指数。", ds))
    }, error = function(e) {
      stop(sprintf("加载 %s 的 TIF 文件时出错: %s。请删除 res_change_detection 目录
后重试。", ds, e$message))
    })
  }
  message_line("所有指数 TIF 文件加载完毕。")

} else {
  message_line("部分或全部指数 TIF 文件缺失，开始批量计算...")

  for (i in seq_along(tif_files)) {
    tf <- tif_files[i]; ds <- date_strs[i]
    message_line(sprintf("处理 %s (%s) ...", basename(tf), ds))
```

```
    idxs <- compute_indices_from_stack(tf)

    # 使用 expected_files_list 中的路径写入文件
    writeRaster(idxs$NDVI, expected_files_list[["NDVI"]][[ds]], overwrite = TRUE)
    writeRaster(idxs$NDWI, expected_files_list[["NDWI"]][[ds]], overwrite = TRUE)
    writeRaster(idxs$NDBI, expected_files_list[["NDBI"]][[ds]], overwrite = TRUE)
    writeRaster(idxs$SAVI, expected_files_list[["SAVI"]][[ds]], overwrite = TRUE)

    # 存入列表
    ndvi_list[[ds]] <- idxs$NDVI
    ndwi_list[[ds]] <- idxs$NDWI
    ndbi_list[[ds]] <- idxs$NDBI
    savi_list[[ds]] <- idxs$SAVI
  }
  message_line("指数 TIF 文件计算并保存完毕。")
}
# 7. 构建时间栈
ndvi_stack <- stack(ndvi_list); names(ndvi_stack) <- date_strs
ndwi_stack <- stack(ndwi_list); names(ndwi_stack) <- date_strs
ndbi_stack <- stack(ndbi_list); names(ndbi_stack) <- date_strs
savi_stack <- stack(savi_list); names(savi_stack) <- date_strs
# 将时间转换为以第一个选定日期为基准的年数（便于线性拟合）
time_years <- as.numeric(dates - min(dates)) / 365.25
# 8. 像素级趋势计算（使用闭式公式，避免 lm；分块写入磁盘）
# 对每个像元 v（向量），我们在有有效值的位置上用以下闭式公式计算：
# slope = cov(t,x) / var(t)
# R2 = (cov(t,x)^2) / (var(t) * var(x))
# mean = mean(x)
# calc 函数会把每个像元的时间序列作为向量传入函数。
# 在开始计算前，调整 raster 的临时目录与内存选项（降低内存压力）
rasterOptions(tmpdir = out_dir,   # 临时文件写到输出文件夹（避免系统 /tmp 空间不足）
              progress = "text",
              chunksize = 1e+07,  # 可调整；默认为 5e+05 等。这里稍大以减少 IO 次数，
但不至于太大。
              maxmemory = 1e+08)   # 根据机器内存可适当调小
# 闭式计算函数（传入一个像元的时间序列向量 v）
pixel_trend_closedform <- function(v) {
  # v: 长度 = nlayers(stk)，包含 NA 或数值
  ok <- is.finite(v)
  n <- sum(ok)
  if (n == 0) return(c(NA_real_, NA_real_, NA_real_))
  if (n < 2) return(c(NA_real_, NA_real_, mean(v[ok], na.rm = TRUE)))
  t <- time_years[ok]
  x <- v[ok]
```

```
  # sums
  sum_t <- sum(t)
  sum_x <- sum(x)
  sum_tx <- sum(t * x)
  sum_t2 <- sum(t * t)
  sum_x2 <- sum(x * x)

  # means
  mean_t <- sum_t / n
  mean_x <- sum_x / n

  # denominator for slope (n * var(t))
  denom_t <- (sum_t2 - n * mean_t * mean_t)  # = n * var(t)

  # compute slope
  num_cov <- (sum_tx - n * mean_t * mean_x)  # = n * cov(t,x)
  if (abs(denom_t) < 1e-12) {
    slope <- NA_real_
  } else {
    slope <- num_cov / denom_t
  }

  # compute R^2 using closed form:
  denom_x <- (sum_x2 - n * mean_x * mean_x)  # = n * var(x)
  denom_r <- denom_t * denom_x
  if (denom_r <= 0) {
    r2 <- NA_real_
  } else {
    r2 <- (num_cov * num_cov) / denom_r
    # numeric guard
    if (!is.finite(r2)) r2 <- NA_real_
  }

  c(slope, r2, mean_x)
}
# 为了确保 calc 输出的三个波段名合适，写个包装器：
calc_trend_safe <- function(stk, name, out_dir, filename = NULL) {
  message_line(paste("计算", name, "趋势（闭式解 + 分块写磁盘）..."))
  if (is.null(filename)) filename <- file.path(out_dir, paste0(name,
"_trend.tif"))
  # 使用 raster::calc，fun 返回长度为 3 的 numeric 向量 -> 会生成 3 个波段
  # 指定 datatype 为 FLT4S（32-bit float），以节约磁盘与内存
  tr <- calc(stk, fun = pixel_trend_closedform, filename = filename,
```

```
            format = "GTiff", overwrite = TRUE, datatype = "FLT4S", progress =
"text")
  names(tr) <- c(paste0(name, "_slope"), paste0(name, "_R2"), paste0(name,
"_mean"))
  message_line("写出：", filename)
  return(tr)
}
# 计算（对每个指数）
ndvi_trend <- calc_trend_safe(ndvi_stack, "NDVI", out_dir)
ndwi_trend <- calc_trend_safe(ndwi_stack, "NDWI", out_dir)
ndbi_trend <- calc_trend_safe(ndbi_stack, "NDBI", out_dir)
savi_trend <- calc_trend_safe(savi_stack, "SAVI", out_dir)
# 若启用了并行，请在结束后关闭集群
# endCluster()
## 9. 差值与百分比变化（首末时相）：保持原来实现，但在写出时指定数据类型
first_idx <- 1; last_idx <- nlayers(ndvi_stack)
diff_fun <- function(stk){
  stk[[last_idx]] - stk[[first_idx]]
}
pct_fun <- function(stk){
  overlay(stk[[first_idx]], stk[[last_idx]], fun = function(a,b){
    ifelse(is.na(a)|is.na(b), NA, (b - a)/(abs(a)+1e-6))
  })
}
indices <- list(NDVI=ndvi_stack, NDWI=ndwi_stack, NDBI=ndbi_stack,
SAVI=savi_stack)
for (nm in names(indices)) {
  d <- diff_fun(indices[[nm]])
  p <- pct_fun(indices[[nm]])
  writeRaster(d, file.path(out_dir, paste0(nm, "_diff.tif")), overwrite = TRUE,
datatype = "FLT4S")
  writeRaster(p, file.path(out_dir, paste0(nm, "_pctchg.tif")), overwrite = TRUE,
datatype = "FLT4S")
}
## 10. 可视化（时间序列与趋势图）
mean_over_time <- function(stk){
  sapply(1:nlayers(stk), function(i) {
    cellStats(stk[[i]], stat = "mean", na.rm = TRUE)
  })
}
ndvi_means <- mean_over_time(ndvi_stack)
ndwi_means <- mean_over_time(ndwi_stack)
ndbi_means <- mean_over_time(ndbi_stack)
savi_means <- mean_over_time(savi_stack)
```

```
### 10.1 时间序列图
message_line("生成 NDVI 时间序列图...")
save_seq_plot({
  op <- par(mar = c(5, 4.5, 3, 1)) # 调整单图的边距
  on.exit(par(op), add=TRUE)
  plot(dates, ndvi_means, type="b", pch=19, main="NDVI 均值变化",
       xlab="", ylab="Mean NDVI", xaxt="n")
  # 添加自定义 X 轴, las=2 使标签垂直, cex.axis 缩小字体
  axis.Date(1, at = dates, format = "%Y-%m", las = 2, cex.axis = 0.7)
}, "ChangeDetection_TimeSeries_NDVI")
message_line("生成 NDWI 时间序列图...")
save_seq_plot({
  op <- par(mar = c(5, 4.5, 3, 1))
  on.exit(par(op), add=TRUE)
  plot(dates, ndwi_means, type="b", pch=19, main="NDWI 均值变化",
       xlab="", ylab="Mean NDWI", xaxt="n")
  axis.Date(1, at = dates, format = "%Y-%m", las = 2, cex.axis = 0.7)
}, "ChangeDetection_TimeSeries_NDWI")
message_line("生成 NDBI 时间序列图...")
save_seq_plot({
  op <- par(mar = c(5, 4.5, 3, 1))
  on.exit(par(op), add=TRUE)
  plot(dates, ndbi_means, type="b", pch=19, main="NDBI 均值变化",
       xlab="", ylab="Mean NDBI", xaxt="n")
  axis.Date(1, at = dates, format = "%Y-%m", las = 2, cex.axis = 0.7)
}, "ChangeDetection_TimeSeries_NDBI")
message_line("生成 SAVI 时间序列图...")
save_seq_plot({
  op <- par(mar = c(5, 4.5, 3, 1))
  on.exit(par(op), add=TRUE)
  plot(dates, savi_means, type="b", pch=19, main="SAVI 均值变化",
       xlab="", ylab="Mean SAVI", xaxt="n")
  axis.Date(1, at = dates, format = "%Y-%m", las = 2, cex.axis = 0.7)
}, "ChangeDetection_TimeSeries_SAVI")
### 10.2 各指数斜率空间分布图（分图输出 - 保持不变）
# 定义调色板
pal_veg <- brewer.pal(11,"RdYlGn")  # 植被: 绿增红减
pal_water <- brewer.pal(11,"RdBu") # 水体: 蓝增红减
pal_built <- brewer.pal(9,"YlOrRd")   # 建筑: 红增黄减
message_line("生成 NDVI 斜率空间分布图...")
save_seq_plot({
  # 单图不再需要设置 par(mfrow) 和 mtext
  plot(ndvi_trend[[1]], main="NDVI 斜率 (植被)", col=pal_veg, legend.width=1)
}, "NDVI_Slope_Map", # 使用新前缀, 如 NDVI_Slope_Map_001.png
```

```
width = 8, height = 7, res = 200 # 调整为适合单图的尺寸
)
message_line("生成 NDWI 斜率空间分布图...")
save_seq_plot({
  plot(ndwi_trend[[1]], main="NDWI 斜率 (水体)", col=pal_water, legend.width=1)
}, "NDWI_Slope_Map", # 如 NDWI_Slope_Map_001.png
width = 8, height = 7, res = 200
)
# (修改后)
message_line("生成 NDBI 斜率空间分布图...")
save_seq_plot({
  v <- values(ndbi_trend[[1]])
  q_robust <- quantile(v, c(0.01, 0.99), na.rm = TRUE)
  # 确保范围有效（防止所有值都一样）
  if (is.na(q_robust[1]) || is.na(q_robust[2]) || diff(q_robust) == 0) {
    q_robust <- c(-0.1, 0.1) # 设置一个合理的默认值
  }

  plot(ndbi_trend[[1]],
       main="NDBI 斜率 (建筑)",
       col=pal_built,
       legend.width=1,
       zlim=q_robust) # --- 修改：添加 zlim 参数 ---

}, "NDBI_Slope_Map", # 如 NDBI_Slope_Map_001.png
width = 8, height = 7, res = 200
)
message_line("生成 SAVI 斜率空间分布图...")
save_seq_plot({
  plot(savi_trend[[1]], main="SAVI 斜率 (土壤)", col=pal_veg, legend.width=1)
}, "SAVI_Slope_Map", # 如 SAVI_Slope_Map_001.png
width = 8, height = 7, res = 200
)
## 10.3 斜率直方图
message_line("生成 NDVI 斜率直方图...")
save_seq_plot({
  op <- par(mar = c(4, 4, 3, 1)) # 调整单图边距
  on.exit(par(op), add=TRUE)
  hist(values(ndvi_trend[[1]]), breaks=60, main="NDVI 斜率直方图", xlab="Slope
(1/yr)", col="darkgreen")
}, "NDVI_Slope_Histogram")
message_line("生成 NDWI 斜率直方图...")
save_seq_plot({
  op <- par(mar = c(4, 4, 3, 1))
```

```
  on.exit(par(op), add=TRUE)
  hist(values(ndwi_trend[[1]]), breaks=60, main="NDWI 斜率直方图", xlab="Slope
(1/yr)", col="darkblue")
}, "NDWI_Slope_Histogram")
message_line("生成 NDBI 斜率直方图...")
save_seq_plot({
  op <- par(mar = c(4, 4, 3, 1))
  on.exit(par(op), add=TRUE)
    v <- values(ndbi_trend[[1]])
  q_robust <- quantile(v, c(0.01, 0.99), na.rm = TRUE)
  if (is.na(q_robust[1]) || is.na(q_robust[2]) || diff(q_robust) == 0) {
    q_robust <- c(-0.1, 0.1) # 设置一个合理的默认值
  }
  # 根据这个范围生成 60 个 breaks
  b <- seq(q_robust[1], q_robust[2], length.out = 61)
  hist(v, # 仍然使用原始 v
       breaks=b,
       xlim=q_robust,
       main="NDBI 斜率直方图", xlab="Slope (1/yr)", col="darkred")
}, "NDBI_Slope_Histogram")
message_line("生成 SAVI 斜率直方图...")
save_seq_plot({
  op <- par(mar = c(4, 4, 3, 1))
  on.exit(par(op), add=TRUE)
  hist(values(savi_trend[[1]]), breaks=60, main="SAVI 斜率直方图", xlab="Slope
(1/yr)", col="darkolivegreen")
}, "SAVI_Slope_Histogram")
## 11. 所有指数的分类变化图（基于各自斜率的四分位）
message_line("开始生成所有指数的分类变化图...")
# 定义一个辅助函数来处理分类和绘图
process_and_plot_change_category <- function(
    trend_raster,   # 包含 slope, R2, mean 的 RasterStack
    index_name,     # "NDVI", "NDWI" 等
    plot_colors     # c(下降颜色, 稳定颜色, 上升颜色)
) {

  message_line(sprintf("  正在处理 %s...", index_name))

  # 1. 提取斜率图层
  slope_raster <- trend_raster[[1]] # 第一个波段是 slope
  names(slope_raster) <- sprintf("%s_slope", index_name)

  # 2. 计算分位数
  slope_vals <- values(slope_raster)
```

```
  q_hi <- quantile(slope_vals, 0.75, na.rm=TRUE)
  q_lo <- quantile(slope_vals, 0.25, na.rm=TRUE)

  message_line(sprintf("    %s 分类阈值: Q25 (下降) < %.4f, Q75 (上升) > %.4f",
                       index_name, q_lo, q_hi))

  # 3. 执行分类
  # -1 = 显著下降 (Bottom 25%)
  #  0 = 相对稳定
  #  1 = 显著上升 (Top 25%)
  change_cat_raster <- calc(slope_raster, fun=function(v){
    if (is.na(v)) return(NA)
    if (v > q_hi) return(1)
    if (v < q_lo) return(-1)
    return(0)
  })

  # 4. 保存 Tif (使用 INT1S 节省空间)
  out_tif_fn <- file.path(out_dir, sprintf("%s_change_cat.tif", index_name))
  writeRaster(change_cat_raster, out_tif_fn, overwrite=TRUE, datatype="INT1S")

  # 5. 保存 PNG 可视化
  plot_prefix <- sprintf("%s_Change_Category", index_name)
  plot_title <- sprintf("%s 变化类别 (基于四分位)", index_name)

  save_seq_plot({
    plot(change_cat_raster, col=plot_colors, legend=FALSE,
         main=plot_title)

    # 添加分类图例
    legend("topleft",
           legend=c("显著上升 (Top 25%)", "相对稳定", "显著下降 (Bottom 25%)"),
           fill=rev(plot_colors), # 反转图例顺序以匹配 1, 0, -1
           bg="white")
  }, plot_prefix)

  message_line(sprintf("    %s 分类图已保存。", index_name))
}
# 批量调用
# 定义颜色方案 (下降, 稳定, 上升)
cols_veg <- c("#d73027","#ffffff","#1a9850")    # 植被 (NDVI, SAVI): 降(红), 稳(白), 升(绿)
cols_water <- c("#d73027","#ffffff","#4575b4")  # 水体 (NDWI): 降(红), 稳(白), 升(蓝)
```

```
cols_built <- c("#1a9850","#ffffff","#d73027")  # 建筑 (NDBI): 降(绿), 稳(白), 升(红)
# 依次处理每个指数
process_and_plot_change_category(ndvi_trend, "NDVI", cols_veg)
process_and_plot_change_category(ndwi_trend, "NDWI", cols_water)
process_and_plot_change_category(ndbi_trend, "NDBI", cols_built)
process_and_plot_change_category(savi_trend, "SAVI", cols_veg)
message_line("所有指数的分类变化图已全部生成。")
## 12. 汇总输出（统计 & 时间序列表）
message_line("正在为 NDVI 汇总统计提取斜率值...")
slope_vals <- values(ndvi_trend[[1]]) # ndvi_trend 的第 1 个波段是斜率
slope_vals_clean <- slope_vals[is.finite(slope_vals)]
stat_df <- data.frame(
  metric=c("count","mean","median","sd","min","max","q25","q75"),
  value=c(length(slope_vals_clean), mean(slope_vals_clean),
median(slope_vals_clean),
          sd(slope_vals_clean), min(slope_vals_clean), max(slope_vals_clean),
          quantile(slope_vals_clean,0.25), quantile(slope_vals_clean,0.75))
)
write.csv(stat_df, file=file.path(out_dir,"NDVI_slope_stats.csv"),
row.names=FALSE)
df_ts <- data.frame(date=dates, NDVI_mean=ndvi_means, NDWI_mean=ndwi_means,
                    NDBI_mean=ndbi_means, SAVI_mean=savi_means)
write.csv(df_ts, file=file.path(out_dir,"Indices_Mean_TimeSeries.csv"),
row.names=FALSE)
message_line("变化检测完成！所有结果已保存到 ", normalizePath(out_dir), "/")
```

③ tool_visualize_single_img.R

```
setwd("E:/code/r_code/20251031_Remote_Sensing_Image_Processing_Using_R") # 设置工作路径，便于使用相对路径（带有盘符的是绝对路径）
library(terra)
# 1) 指定你的影像路径（把文件名改成你那一张）
#    例：Windows 用反斜杠或双反斜杠；Mac/Linux 用斜杠
#    假设你的文件夹叫 "GEE_L8_Yangdian_images"
image_path <-
"GEE_L8_Yangdian_images_cloudfree/L8_Yangdian_20250902_LateSeason_LC8123039202
5245LGN00_masked_cloudfree.tif"
# 2) 读取影像
x <- rast(image_path)
# 3) 选择真彩色波段：红=SR_B4，绿=SR_B3，蓝=SR_B2
bn <- names(x)
if (all(c("SR_B4","SR_B3","SR_B2") %in% bn)) {
    rgb <- x[[c("SR_B4","SR_B3","SR_B2")]]
} else if (all(c("B4","B3","B2") %in% bn)) {
```

```
    # 兼容部分导出或重命名场景
    rgb <- x[[c("B4","B3","B2")]]
} else {
    stop("未找到用于真彩色的 3 个波段：请检查文件是否包含 SR_B4 / SR_B3 / SR_B2。")
}
# 4) 可视化前处理
#    你的 GEE 里已按 0.0000275 和 -0.2 做了缩放，可能出现<0 或>1 的值，显示前裁剪到
[0,1]
rgb <- clamp(rgb, lower=0, upper=1)
# 5) 绘图—线性拉伸 + 轻微 gamma（让观感更自然）
#    q=c(0.02, 0.98) 表示按 2%~98% 分位做拉伸；可以根据观感改成 c(0.01,0.99) 或
c(0.03,0.97)
plotRGB(
    rgb,
    stretch = "lin",
    q = c(0.02, 0.98),
    gamma = 1.1,
    axes = TRUE,
    main = "Landsat 8 真彩色 (SR_B4-B3-B2)"
)
# 6) 如需保存到图片（可选）
# png("L8_truecolor.png", width=1600, height=1200)
# plotRGB(rgb, stretch="lin", q=c(0.02,0.98), gamma=1.1, axes=TRUE,
#         main="Landsat 8 真彩色 (SR_B4-B3-B2)")
# dev.off()
# —额外：如想看近红外假彩色（可选），解注释以下几行—
# if (all(c("SR_B5","SR_B4","SR_B3") %in% bn)) {
#   nir_rgb <- x[[c("SR_B5","SR_B4","SR_B3")]] |> clamp(0,1)
#   plotRGB(nir_rgb, stretch="lin", q=c(0.02,0.98), gamma=1.1, axes=TRUE,
#           main="Landsat 8 假彩色 (NIR-Red-Green = SR_B5-B4-B3)")
# }
```

④ tool_visualize_folder_img.R

```
# 批处理整文件夹：将遥感影像渲染为真彩色 PNG
# 依赖：terra
# 使用方法：
# 1) 修改【参数区】中的路径和可视化参数；
# 2) 直接运行本脚本。
# ====================== 基础设置 ======================
setwd("E:/code/r_code/20251031_Remote_Sensing_Image_Processing_Using_R")  # 工
作路径
suppressPackageStartupMessages({
    library(terra)
})
# ====================== 参数区（按需修改） ======================
in_dir     <- "GEE_L8_Yangdian_image"  # 输入影像文件夹（相对或绝对路径均可）
```

```r
out_dir    <- "GEE_L8_Yangdian_images_vis"                    # 输出图片文件夹
pattern    <- "\\.(tif|tiff|img)$"                            # 处理的文件扩展名
recursive  <- TRUE                                            # 是否递归遍历子文件夹
# 可视化参数（线性拉伸与 gamma）
q_low      <- 0.02                                            # 下分位（建议 0.01~0.03 之间调）
q_high     <- 0.98                                            # 上分位
gamma_v    <- 1.1                                             # gamma（>1 稍亮中低调；=1 关闭）
add_axes   <- FALSE                                           # 是否绘制坐标轴
bg_col     <- "white"                                         # 画布背景色
# 输出图像尺寸与质量
base_width <- 1800                                            # 基准宽度（像素）
max_width  <- 2400                                            # 最大宽度（像素，上限避免超大图）
dpi        <- 150                                             # 分辨率（像素/英寸）
overwrite  <- TRUE                                            # 已存在是否覆盖
# 可选：同时导出近红外假彩色（NIR-Red-Green = SR_B5-SR_B4-SR_B3）
export_falsecolor <- FALSE                                    # 需要就设为 TRUE
# ====================== 工具函数 ======================
# 根据常见命名自动抓取 RGB 三个波段
get_rgb <- function(x) {
    bn <- names(x)
    bn_lower <- tolower(bn)
    # 1) Landsat SR 命名
    if (all(c("sr_b4","sr_b3","sr_b2") %in% bn_lower)) {
        return(x[[bn[match(c("sr_b4","sr_b3","sr_b2"), bn_lower)]]])
    }
    # 2) 简写 B4/B3/B2
    if (all(c("b4","b3","b2") %in% bn_lower)) {
        return(x[[bn[match(c("b4","b3","b2"), bn_lower)]]])
    }
    # 3) red/green/blue
    if (all(c("red","green","blue") %in% bn_lower)) {
        return(x[[bn[match(c("red","green","blue"), bn_lower)]]])
    }
    stop("未找到真彩色所需三波段（SR_B4/B3/B2 或 B4/B3/B2 或 red/green/blue）")
}
# 假彩色（NIR-Red-Green）三波段
get_nir_rgb <- function(x) {
    bn <- names(x); bn_lower <- tolower(bn)
    if (all(c("sr_b5","sr_b4","sr_b3") %in% bn_lower)) {
        return(x[[bn[match(c("sr_b5","sr_b4","sr_b3"), bn_lower)]]])
    }
    if (all(c("b5","b4","b3") %in% bn_lower)) {
        return(x[[bn[match(c("b5","b4","b3"), bn_lower)]]])
    }
    return(NULL)  # 没有就返回空
}
# 按图幅比例计算输出尺寸
calc_size <- function(r, base_w=1800, max_w=2400) {
    nc <- ncol(r); nr <- nrow(r)
    ar <- nr / nc
    width_px  <- min(max(base_w, 800), max_w)
    height_px <- max( round(width_px * ar), 600 )
    list(w=width_px, h=height_px)
}
```

```r
# ====================== 目录与文件收集 ======================
if (!dir.exists(out_dir)) dir.create(out_dir, recursive = TRUE)
files <- list.files(in_dir, pattern = pattern, full.names = TRUE, recursive =
recursive)
if (length(files) == 0) stop("在输入文件夹中未找到影像（检查路径与扩展名 pattern）。
")
# 可选：控制临时与内存占用
terraOptions(memfrac = 0.8)
# ====================== 处理与导出 ======================
log <- data.frame(
    file    = character(),
    out_png = character(),
    ncol    = integer(),
    nrow    = integer(),
    success = logical(),
    message = character(),
    stringsAsFactors = FALSE
)
if (export_falsecolor) {
    log_false <- log[0,]
}
for (f in files) {
    cat("\n>>> 处理：", f, "\n")
    base <- tools::file_path_sans_ext(basename(f))
    # -------- 真彩色 --------
    out_png <- file.path(out_dir, paste0(base, "_RGB.png"))
    if (!overwrite && file.exists(out_png)) {
        cat("跳过（已存在）：", out_png, "\n")
        log <- rbind(log, data.frame(file=f, out_png=out_png, ncol=NA, nrow=NA,
success=TRUE, message="skip exists"))
    } else {
        tryCatch({
            r   <- rast(f)
            rgb <- get_rgb(r)
            # 若 GEE 已做 0.0000275/-0.2 缩放，这里只需裁剪到 [0,1]
            rgb <- clamp(rgb, lower=0, upper=1, values=TRUE)
            sz  <- calc_size(rgb, base_w = base_width, max_w = max_width)
            png(filename = out_png, width = sz$w, height = sz$h, res = dpi, bg =
bg_col)
            op <- par(mar=c(0,0,2,0))
            plotRGB(
                rgb,
                stretch = "lin",
                q = c(q_low, q_high),
                gamma = gamma_v,
                axes = add_axes,
                main = paste0(base, " — 真彩色 (SR_B4-B3-B2)")
            )
            par(op); dev.off()
            cat("完成：", out_png, "\n")
            log <- rbind(log, data.frame(file=f, out_png=out_png, ncol=ncol(rgb),
nrow=nrow(rgb), success=TRUE, message="ok"))
        }, error = function(e) {
            cat("失败：", conditionMessage(e), "\n")
            if (file.exists(out_png)) try(unlink(out_png), silent=TRUE)
```

```
            log <- rbind(log, data.frame(file=f, out_png=out_png, ncol=NA, nrow=NA,
success=FALSE, message=conditionMessage(e)))
        })
    }
    # -------- 假彩色（可选）--------
    if (export_falsecolor) {
        out_png_fc <- file.path(out_dir, paste0(base, "_FalseColor_NIR-R-G.png"))
        if (!overwrite && file.exists(out_png_fc)) {
            cat("假彩色跳过（已存在）：", out_png_fc, "\n")
            log_false <- rbind(log_false, data.frame(file=f, out_png=out_png_fc,
ncol=NA, nrow=NA, success=TRUE, message="skip exists"))
        } else {
            tryCatch({
                r   <- if (exists("r")) r else rast(f)
                nir <- get_nir_rgb(r)
                if (!is.null(nir)) {
                    nir <- clamp(nir, lower=0, upper=1, values=TRUE)
                    sz  <- calc_size(nir, base_w = base_width, max_w = max_width)
                    png(filename = out_png_fc, width = sz$w, height = sz$h, res =
dpi, bg = bg_col)
                    op <- par(mar=c(0,0,2,0))
                    plotRGB(
                        nir,
                        stretch = "lin",
                        q = c(q_low, q_high),
                        gamma = gamma_v,
                        axes = add_axes,
                        main = paste0(base, " — 假彩色 (NIR-Red-Green =
SR_B5-B4-B3)")
                    )
                    par(op); dev.off()
                    cat("假彩色完成：", out_png_fc, "\n")
                    log_false <- rbind(log_false, data.frame(file=f,
out_png=out_png_fc, ncol=ncol(nir), nrow=nrow(nir), success=TRUE, message="ok"))
                } else {
                    cat("未找到生成假彩色所需波段（SR_B5/B4/B3 或 B5/B4/B3），跳过。
\n")
                    log_false <- rbind(log_false, data.frame(file=f,
out_png=out_png_fc, ncol=NA, nrow=NA, success=FALSE, message="no NIR set"))
                }
            }, error = function(e) {
                cat("假彩色失败：", conditionMessage(e), "\n")
                if (file.exists(out_png_fc)) try(unlink(out_png_fc), silent=TRUE)
                log_false <- rbind(log_false, data.frame(file=f,
out_png=out_png_fc, ncol=NA, nrow=NA, success=FALSE,
message=conditionMessage(e)))
            })
        }
    }
}
# ===================== 输出日志 =====================
write.csv(log, file.path(out_dir, "rgb_export_log.csv"), row.names = FALSE)
if (export_falsecolor) {
    write.csv(log_false, file.path(out_dir, "falsecolor_export_log.csv"),
row.names = FALSE)
}
```

```
cat("\n—— 全部完成 ——\n",
    "真彩色 成功：", sum(log$success), "；失败：", sum(!log$success),
    "\n 输出目录：", normalizePath(out_dir), "\n")
if (export_falsecolor) {
    cat("假彩色 成功:", sum(log_false$success), ";失败:", sum(!log_false$success),
"\n")
}
```

参考文献

[1] Jiao, Z. (2024). The Application of Remote Sensing Techniques in Ecological Environment Monitoring. Highlights in Science, Engineering and Technology, 81, 449-455. https://doi.org/10.54097/7dqegz64

[2] Latysh, N., Kirk, K.G., and Faundeen, J., 2020, Purpose and benefits of U.S. Geological Survey Trusted Digital Repositories: U.S. Geological Survey Fact Sheet 2020–3032, 4 p., https://doi.org/10.3133/fs20203032.

[3] Roy, D. P., Wulder, M. A., Loveland, T. R., Woodcock, C. E., Allen, R. G., Anderson, M. C., Helder, D., Irons, J. R., Johnson, D. M., Kennedy, R., Scambos, T. A., Schaaf, C. B., Schott, J. R., Sheng, Y., Vermote, E. F., Belward, A. S., Bindschadler, R., Cohen, W. B., Gao, F., Hipple, J. D., Hostert, P., Huntington, J., Justice, C. O., Kilic, A., Kovalskyy, V., Lee, Z. P., Lymburner, L., Masek, J. G., McCorkel, J., Shuai, Y., Trezza, R., Vogelmann, J., Wynne, R. H., & Zhu, Z. (2014). Landsat-8: Science and product vision for terrestrial global change research. Remote Sensing of Environment, 145, 154 – 172. https://doi.org/10.1016/j.rse.2014.02.001

[4] Rimal, B., Zhang, L., Keshtkar, H., Wang, N., & Lin, Y. (2017). Monitoring and Modeling of Spatiotemporal Urban Expansion and Land-Use/Land-Cover Change Using Integrated Markov Chain Cellular Automata Model. ISPRS International Journal of Geo-Information, 6(9), 288. https://doi.org/10.3390/ijgi6090288

[5] 付悦,刘玉亭. 小城镇建设用地扩张特征与影响因素研究[J]. 大众标准化,2022(9):68-70. DOI:10.3969/j.issn.1007-1350.2022.09.025.

[6] Google Earth Engine. (n.d.). LANDSAT/LC08/C02/T1_L2 — USGS Landsat 8 Collection 2, Tier 1, Level-2. Retrieved November 5, 2025, from https://developers.google.com/earth-engine/datasets/catalog/LANDSAT_LC08_C02_T1_L2

[7] NASA Goddard Space Flight Center. (n.d.). Landsat 8. NASA Landsat Science. Retrieved November 5, 2025, from https://landsat.gsfc.nasa.gov/satellites/landsat-8/

[8] GitCode. (n.d.). open-source-toolkit/5d06e (repository page). Retrieved November 10, 2025, from https://gitcode.com/open-source-toolkit/5d06e

[9] Google Earth Engine. (n.d.). Earth Engine Code Editor. Retrieved November 5, 2025, from https://code.earthengine.google.com/

[10] 徐涵秋,唐菲.新一代 Landsat 系列卫星：Landsat8 遥感影像新增特征及其生态环境意义.生态学报,2013,33(11):3249~3257

[11] Sena, I., Casagrande, P., Rocha, N., Fonseca, B., & Moura, A. (2018). METHODOLOGY FOR GREEN AND BUILT VOLUME ANALYSIS. Mercator, 17. doi:10.4215/rm2018.e17021

[12] 桂华, 余彪. 散射格局：地缘村落的构成与性质——基于一个移民湾子的考察[J]. 青年研究, 2011, (1): 44-54.

[13] ZHANG Rong, LIU Zheng kai, ZAN Shu. Wavelet-based Compression for Multispectral Imagery[J]. Journal of Remote Sensing, 2000,(2):100-105. DOI： 10.11834/jrs.20000204.

[14] Rouse, J. W., Jr., Haas, R. H., Schell, J. A., & Deering, D. W. (1973, December 10-14). Monitoring vegetation systems in the Great Plains with ERTS (Earth Resources Technology Satellite). In Proceedings of the Third ERTS-1 Symposium (Vol. 1, pp. 309-317). NASA SP-351.

[15] Gao, B. C. (1996). NDWI—a normalized difference water index for remote sensing of vegetation liquid water from space. Remote Sensing of Environment, 58(3), 257-266. https://doi.org/10.1016/S0034-4257(96)00067-3

[16] Zha, Y., Gao, J., & Ni, S. (2003). Use of normalized difference built-up index in automatically mapping urban areas from TM imagery. International Journal of Remote Sensing, 24(3), 583-594. https://doi.org/10.1080/01431160304987

[17] Huete, A. R. (1988). A soil-adjusted vegetation index (SAVI). Remote Sensing of Environment, 25(3), 295-309. https://doi.org/10.1016/0034-4257(88)90106-X

[18] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," in IEEE Transactions on Systems, Man, and Cybernetics, vol. 9, no. 1, pp. 62-66, Jan. 1979, doi: 10.1109/TSMC.1979.4310076.

[19] Shachar, A. (2024). Introduction to algogens. arXiv. https://arxiv.org/abs/2403.01426

[20] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations.

[21] Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324

[22] 马超,崔珍珍,李婷婷,彭杨钊.中国“平原-山地”地形过渡带 NDVI 时空变异与气候响应.生态学报,2023,43(5):2141~2157

[23] Singh, S., Kanhaiya, S., Kumar, S., & Yadav, S. K. (2022). Spatial and temporal variation in NDVI and NDWI of the Ukhma River Basin, Central India. Journal of Scientific Research, 66(3), 62 – 65. https://doi.org/10.37398/JSR.2022.660309

[24] Zaabar, N., Niculescu, S., & Mihoubi, M. K. (2021). Assessment of combining convolutional neural networks and object-based image analysis to land cover classification using Sentinel-2 satellite imagery (Tenes region, Algeria). The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLIII-B3-2021, 383 – 389. https://doi.org/10.5194/isprs-archives-XLIII-B3-2021-383-2021

[25] Thanh Noi, P., & Kappas, M. (2018). Comparison of Random Forest, k-Nearest Neighbor, and Support Vector Machine Classifiers for Land Cover Classification Using Sentinel-2 Imagery. Sensors, 18(1), 18. https://doi.org/10.3390/s18010018

[26] Awty-Carroll, K., Bunting, P., Hardy, A., & Bell, G. (2019). An Evaluation and Comparison of Four Dense Time Series Change Detection Methods Using Simulated Data. Remote Sensing, 11(23), 2779. https://doi.org/10.3390/rs11232779

[27] Ghaderpour, E., Pagiatakis, S. D., & Hassan, Q. K. (2021). A Survey on Change Detection and Time Series Analysis with Applications. Applied Sciences, 11(13), 6141. https://doi.org/10.3390/app11136141

武漢大學
WUHAN UNIVERSITY

武漢大學遥感信息工程学院
School of Remote Sensing and Information Engineering,Wuhan University
笃志 敦行 和协 拓新

本科课程论文（设计）

——2025~2026 学年第一学期水利遥感论文阅读报告

论文《2001-2019 年长江中下游农业干旱遥感监测及植被敏感性分析》阅读报告

姓　　名：	杨 丹 阳
学　　号：	2023302131259
学　　校：	武汉大学
学　　院：	遥感信息工程学院
专　　业：	遥感科学与技术
课　　程：	水利遥感
指导教师：	查 元 源

二〇二五 年 十一 月

目 录

1 综述

1.1 研究目的

长江中下游是中国最重要的粮食主产区之一，但近年来极端气象事件频发，农业干旱监测与评估需求愈发迫切。现有研究虽使用了 VCI、TCI 与 VHI 等遥感指数开展干旱监测，但这些遥感指数针对整个长江中下游六省一市的系统性分析较少，缺乏适应性分析及干旱演变的趋势分析；此外，干旱作为大气过程与地表特性相互作用的结果，其影响与不同地表覆盖类型密切相关，但目前缺少针对长江中下游地区各类地表类型干旱敏感性的研究。

鉴于此，研究旨在构建适用于长江中下游地区的农业干旱遥感监测体系，系统分析 2001—2019 年干旱的时空变化规律，揭示不同植被类型的干旱敏感性特征，并评估区域在气候变化背景下的干旱趋势，为区域农业管理与抗旱决策提供科学依据。

1.2 研究内容

围绕上述目标，研究主要从以下几个方面展开：

（1）以 MODIS 的 NDVI 和地表温度为基础构建 VCI、TCI 和 VHI 指数体系，并与多时间尺度 SPI 对比，分析其在长江中下游地区的适应性及最优权重配置；

（2）选取 2001 年、2011 年这两个典型的干旱年份，通过 VCI、TCI 与 VHI 多指数综合分析揭示干旱事件在全年尺度上的时序演化和空间扩散过程；

（3）利用土地覆盖数据，将区域地表类型归类为林地、草地和农田三类，通过统计不同地类在各干旱等级下的受旱面积比例，量化不同植被类型对干旱的敏感性差异；

（4）构建各省市逐年 VHI 序列，并结合降水距平与气温变化率分析近二十年来区域农业干旱的整体趋势，尝试作出其背后潜在的气候背景解释。

1.3 研究方法

首先对 MODIS 的 NDVI 与 LST 数据进行基于质量控制指标的云污染剔除、逐月时间合成、尺度统一和缺失数据补偿等预处理，确保指数计算的连续性与可靠性。

在此基础上，根据 VCI 和 TCI 的定义分别对植被状态和地表温度进行历史同期归一化处理，并在权重因子α的多组实验中与 SPI-1、SPI-3 和 SPI-6 的进行相关性评估，来确定长江中下游地区 VHI 指数的最优权重配置。

为分析植被敏感性，研究利用 MCD12Q1 数据对土地覆盖类型进行再分类，随后结合月尺度 VHI 干旱等级统计不同地类受旱面积比例，以定量表征敏感性差异。

在干旱趋势分析中，研究构建了 2001~2019 年各省市年度 VHI 序列，计算其线性趋势斜率，并结合区域降水距平序列和年际温度变化率，对干旱缓解或加剧的区域特征进行综合判断。

1.4 研究结论

研究结果表明：

（1）VCI 与中长期尺度的 SPI（SPI-3、SPI-6）具有更高相关性，能够有效反映植被生长异常导致的农业干旱，而 TCI 与短时间尺度的 SPI-1 更为一致，适合刻画温度异常引发的短期干旱。综合两者优势后可知，VHI 在 VCI 与 TCI 的权重比为 0.7∶0.3 时，在本区域表现最优。

（2）典型干旱年份的分析显示，2001 年干旱具有持续时间长、覆盖面广的特征，而 2011 年的干旱主要集中在上半年，随后在夏季降水增加的作用下逐渐缓解。VHI 与降水距平的变化一致性良好，能够准确刻画干旱时空演化。

（3）不同植被对干旱的敏感性存在显著差异，林地因根系深厚表现出最强的抗旱性，草地处于中间，而农田根系浅、需水量大，受旱程度最为明显，显示出最高的干旱敏感性。

（4）在长期演变方面，2001~2019 年间长江中下游整体呈现湿润化趋势，其中江西、湖南、湖北、安徽和浙江的湿润趋势较为显著；江苏与上海也呈现改善但趋势脆弱，仍需关注潜在的干旱风险。这说明区域干旱变化不仅受降水增加影响，同时也受温度升高导致蒸散发加剧的影响。**（论文原文首页见下一页）**

附：《2001-2019年长江中下游农业干旱遥感监测及植被敏感性分析》原文首页

第47卷第8期 武汉大学学报·信息科学版 Vol.47 No.8
2022年8月 Geomatics and Information Science of Wuhan University Aug. 2022

DOI:10.13203/j.whugis20210172 文章编号:1671-8860(2022)08-1245-12

2001—2019年长江中下游农业干旱遥感监测及植被敏感性分析

尹国应[1] 张洪艳[1] 张良培[1,2]

1 武汉大学测绘遥感信息工程国家重点实验室,湖北 武汉,430079
2 地球空间信息技术协同创新中心,湖北 武汉,430079

摘 要:长江中下游地区是中国最重要的粮食产区之一,近年来,由于极端天气影响,长江中下游地区的农业生产时常受到干旱灾害威胁。利用植被条件指数(vegetation condition index, VCI)、温度条件指数(temperature condition index, TCI)及植被健康指数(vegetation health index, VHI)对2001—2019年长江中下游地区农业干旱的时空演变情况进行监测,探究长江中下游地区VCI、TCI在VHI指数中的最优权重比例,挖掘不同植被对干旱的敏感性差异,同时基于气候变化背景分析长江中下游六省一市的干旱趋势。结果表明,VCI和TCI指数能够分别反映地区植被生长异常和热量异常;当VCI和TCI的权重分配比为7:3时,VHI指数能够结合两种指数的特点,在长江中下游地区农业干旱监测上更有优势;不同植被对干旱的敏感性不同,在长江中下游地区,农作物对干旱的敏感性最高,森林最低,草地介于二者之间;在气候变化背景下,近20年来,长江中下游地区呈现逐渐湿润的趋势,干旱风险逐步降低,其中湖北、湖南、安徽、江西和浙江等地湿润趋势明显,而江苏和上海地区湿润趋势较弱,在极端气候下仍存在一定的干旱风险。相关结果能够为长江中下游地区各省市旱情预警及抗旱措施制定、区域农业生产管理提供参考。

关键词:农业干旱监测;植被条件指数;温度条件指数;植被健康指数;植被敏感性;气候变化;长江中下游地区
中图分类号:P237;TP79 **文献标志码**:A

干旱是一种形成机理复杂、持续时间长、影响范围广的自然灾害,对农业生产、人类生活、经济发展造成了严重影响[1-2]。例如,2011年美国德克萨斯州发生的干旱灾害造成农业损失达76.2亿美元[3]。干旱还会诱发一系列次生灾害的发生,如森林火灾、大气污染等[4-5],进一步威胁人类健康和生态环境稳定[6]。根据联合国政府间气候变化专门委员会(The Intergovernmental Panel on Climate Change, IPCC)的相关报告,到2030—2052年左右,全球平均温度将比第二次工业革命前上升1.5℃,气候变化等因素将大幅增加未来极端干旱发生的可能性[7]。干旱主要分为气象干旱、农业干旱、水文干旱和社会经济干旱4种干旱类型[8]。气象干旱一般指降水量缺少引起的水分亏缺现象,可以用降水量相关指标进行衡量。随着降水量的长期亏缺,土壤水分含量减少,无法满足植被对水分需求,干旱演变为农业干旱。农业干旱的严重程度可以利用土壤水分含量、植被生长状况和地表温度反映。水文干旱是河流湖泊水资源减少或地下水位低于正常值的现象,一般由气象干旱或农业干旱的旱情条件进一步恶化引起。社会经济干旱则强调自然气候干旱对人类社会经济活动造成的影响[9]。农业干旱密切影响粮食产量、关系农业生产,因此本文主要关注农业干旱的监测研究。

早期农业干旱监测主要使用气象监测方法,通过测量站点获取的降水量和土壤墒情等数据,计算得到帕尔默干旱严重指数(palmer drought severity index, PDSI)、标准化降水指数(standardized precipitation index, SPI)、Z指数(Z-index)等对干旱程度进行等级量化,最终达到监测的目的[10]。这些方法虽然能够真实反映地面干旱情况,但站点分布稀疏,日常维护及数据测量成本高,不利于进行大范围的农业干旱监测[11]。

收稿日期:2021-04-09
项目资助:国家自然科学基金(61871298,42071322);湖北省自然科学基金(2020CFA053)。
第一作者:尹国应,博士生,研究方向为农业遥感。yin_gy@whu.edu.cn
通讯作者:张洪艳,博士,教授。zhanghongyan@whu.edu.cn

2 与水利遥感课程知识点的关系

2.1 电磁辐射与地物光谱特征

论文中使用的遥感数据（如 MODIS 的红光与近红外波段）直接体现了电磁波在不同地物上的反射特性差异。NDVI（归一化植被指数）就是基于红光吸收与近红外反射差异计算的；植被水分胁迫、温度异常等遥感特征的识别依赖于地物光谱特征。这与课程**第二章关于“电磁波谱特征与地物识别”的理论基础**相关。

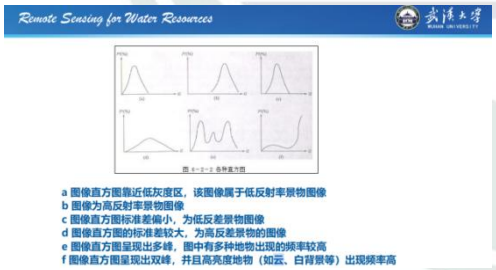
2.2 遥感成像原理和成像特征

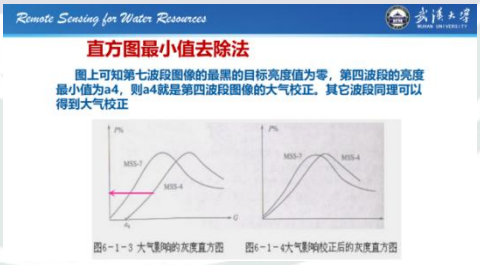
研究采用了 MODIS 多光谱成像仪的影像产品（MOD09GA、MOD11A1、MCD12Q1）来监测干旱和植被类型，论文中提到的“地表反射率”、“地表温度”、“空间分辨率重采样”等也是成像原理与传感器特征的常用术语。因此，该论文体现了课程**第三章中“遥感影像的成像机理”“分辨率与遥感影像特征分析”**等知识点。

此外，为了分析不同植被对干旱的敏感性，本文还使用了中分辨率成像光谱仪（moderate resolution imaging spectroradiometer，MODIS）的MCD12Q1产品中的土地覆盖类型数据。该产品包括5种全球土地覆盖分类子产品，分别是国际地圈-生物圈计划（international geosphere-biosphere programme，IGBP）分类、马里兰大学分类、叶面积指数分类、生物地球化学循环（biome-biogeochemical cycles，BGC）分类和植物功能类型分类。本文选取IGBP分类作为分类依据，共包含17个土地覆盖类型，分辨率为1 km。考虑到长江中下游地区区域独特性，本文将研究区域内不包含的类别剔除，并按照剩余类别的类间相似性对其他13类进行重新归类。原始分类和重新归类结果如表1所示。

2.3 遥感图像处理

论文的数据处理过程非常典型，包括大气校正、利用质量评估波段 QA 掩膜云区进行云检测与去除、将掩膜后的无云影像通过逐月平均得到每月的合成影像、空间重采样以统一 NDVI 与 LST 分辨率至 1 km）、利用相邻时相平均值插值将缺失值补全等，这些步骤都是遥感图像预处理与校正的具体应用，体现了课程**第四章中遥感图像处理**相关知识点。





1.3 研究方法

在计算干旱监测指数前，需要对NDVI和LST数据进行预处理，包括云去除、月尺度合成、统一分辨率和缺失数据补全。其中云去除是利用MODIS数据的质量评估（quality assessment，QA）波段，将每日的受云污染影响的影像掩膜，留下无云区域的影像；月尺度合成是将掩膜后的无云影像通过逐月平均得到每月的合成影像；将500 m分辨率的NDVI采样到1 000 m，统一NDVI和LST的分辨率；由于月尺度合成不能够完全保证云去除之后的无云数据在研究区域上月尺度合成后的完整性，因此在当前月份存在空间数据空洞时，利用该月份的相邻月份和历史同

2.4 基于遥感指数的遥感图像解译与典型土地利用类型判别

论文在干旱敏感性研究中使用了 MCD12Q1 土地覆盖产品，并将 IGBP 分类体系重新归类为林地、草地、农田等类别。这种处理反映了遥感数据在土地利用/土地覆盖分类中的应用，与课程中**第六章典型土地利用类型**的“**土地利用类型遥感识别与分类**”等直接相关。此外，论文在干旱监测与植被敏感性分析时，通过计算和对 VCI、TCI、VHI 等遥感指数进行分级阈值划分（这个操作在我的 Landsat-8 影像实践作业中也进行了）实现了遥感影像的定量解译，比如根据 VHI 值将影像所对应的地表划分为极端干旱、严重干旱、中度干旱等等级，这部分对应了课程**第五章中遥感图像解译**的相关内容。

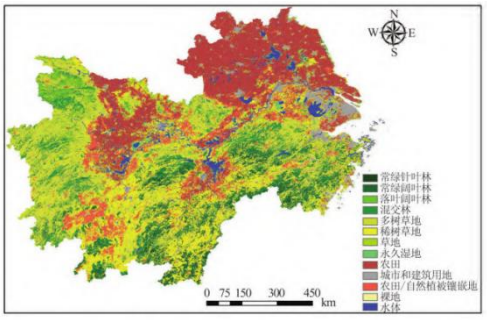


图1 研究区域范围及土地覆盖类型

Fig.1 Location and Land Cover Types of the Middle and Lower Reaches of the Yangtze River

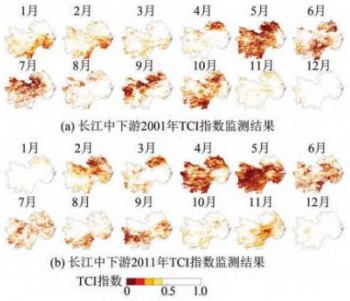


图6 长江中下游地区2001年与2011年TCI指数的监测结果

Fig. 6 Monitoring Results of 2001 and 2011 in the Middle and Lower Reaches of the Yangtze River Using TCI

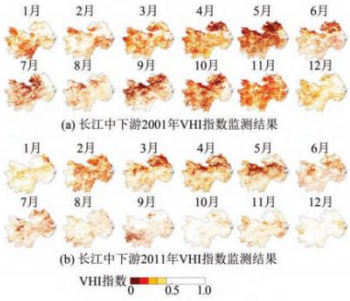
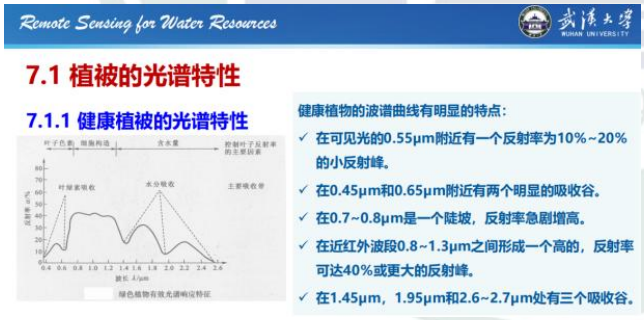


图7 长江中下游地区2001年与2011年VHI指数的监测结果

Fig. 7 Monitoring Results of 2001 and 2011 in the Middle and Lower Reaches of the Yangtze River Using VHI

2.5 植被遥感

这是论文的核心对应部分，这篇论文的主要内容即通过NDVI、VCI、TCI、VHI等遥感指数监测农业干旱和植被状态变化，分析不同植被（林地、草地、农田）对干旱的响应差异等。这些恰好和**第七章植被遥感**中的内容高度相关，包括**各种植被指数原理与应用、植被覆盖变化监测、植被生态参数与其他环境因子（温度、水分）的关系（比如哪些因素会影响植被生长或者对植被指数的计算产生干扰）**等等。



1.2 研究数据

本文使用的数据包括NDVI、地表温度(land surface temperature，LST)、降水量数据和地表覆盖类型数据。其中，NDVI由地表反射率数据产品MOD09GA中的红波段和近红外波段计算得到[16]，可以很好地反映植被的生长状况，能够作为监测植被水分胁迫及农业干旱发展状况的有效指标；LST选取MOD11A1产品的日间陆地表面温度数据，当植被受到水分胁迫时，叶片气孔关闭，蒸散发活动减少，局部温度升高[20]。NDVI的原始分辨率为500 m，LST的原始分辨率为1 km，为了统一分辨率，同时一定程度上消除单位像元上因土地覆盖/土地利用变化带来的误差，本文对NDVI栅格数据进行重采样聚合，将分辨率采样到1 km。两类数据的时间范围均选择2001-01—2019-12，共计19 a。

3 本课程中没有讲到的内容

3.1 多时间尺度气象干旱指数 SPI 与降水距平概念

（1）标准化降水指数 SPI 的定义与多时间尺度含义

论文中用 SPI-1、SPI-3、SPI-6 这三种时间尺度的标准化降水指数来评估 VCI、TCI、VHI 这些遥感指数的适应性，但是 SPI 的概念在我之前所学的课程中基本没有涉及。

我先从原文中看到 SPI 的引用，顺着参考文献中关于干旱指数和 SPI 的文献（Li Bin et al., 2011[1]、Wang Xianwei et al., 2014[2]）了解 SPI 的数学定义和计算流程。又查阅了国内关于农业干旱监测方法的综述性文章（钱莉莉等，2019[3]、黄友昕等，2015[4]），这些文献系统解释了 SPI 与 PDSI、Z 指数等气象干旱指标的区别，让我对“多时间尺度干旱指数”的定位更加清楚。

SPI 的计算公式为：

$$SPI = \frac{P - \mu}{\sigma}$$

其中，P是当前时间段的降水量，μ是长期平均降水量，σ是降水量的标准差。这个公式的意义是：SPI 值表示当前降水量与长期平均降水量的偏差程度，以标准差为单位。

根据我的理解，SPI 的核心思想是对长时间序列降水进行概率分布拟合，再把某一时段的降水量转化为标准化的距平值，从而量化干旱强弱，而 SPI-1、SPI-3、SPI-6 不同时间尺度（这些 SPI 的后缀的单位为月）对应不同类型干旱，包括短期气象旱、中期农业旱、长期水文旱等。对于 SPI-1、SPI-3、SPI-6 这三种时间尺度的标准化降水指数，SPI-1 更敏感于短时降水异常，适合气象干旱；SPI-3、SPI-6 则平滑了短期波动，更适合反映作物生长季的水分累积状况，因此更贴近农业干旱。

3.2 VCI、TCI、VHI 等遥感指数及其定义、意义等相关信息

（1）VCI、TCI、VHI 等遥感指数

课程中讲到了 NDVI 等指标，但没有讲过条件植被指数 VCI、条件温度指数 TCI 和植被健康指数 VHI 及其计算方式。

通过查看原文中的公式（1）~（3）：

$$\mathrm{VCI}_{ijk} = \frac{\mathrm{NDVI}_{ijk} - \mathrm{NDVI}_{\min}}{\mathrm{NDVI}_{ij}^{\max} - \mathrm{NDVI}_{ij}^{\min}} \quad (1)$$

$$\mathrm{TCI}_{ijk} = \frac{\mathrm{LST}_{ij}^{\max} - \mathrm{LST}_{ijk}}{\mathrm{LST}_{ij}^{\max} - \mathrm{LST}_{ij}^{\min}} \quad (2)$$

$$\mathrm{VHI}_{ijk} = \alpha \times \mathrm{VCI}_{ijk} + (1-\alpha) \times \mathrm{TCI}_{ijk} \quad (3)$$

以及追溯论文参考文献中 Kogan(1995)[5]的工作（提出了条件温度指数（temperature condition index，TCI）监测地表温度异常，并综合 VCI 和 TCI 进一步提出植被健康指数（vegetation health index，VHI）），我了解到：

①VCI 是用当前 NDVI 与历史同期 NDVI 的最小、最大值归一化，反映植被“相对历史正常状态”的好坏：

$$VCI = \frac{NDVI_{cur} - NDVI_{min}}{NDVI_{max} - NDVI_{min}}$$

其中，$NDVI_{cur}$为当前时段的$NDVI$，$NDVI_{min}$、$NDVI_{max}$为该像元在同历时段的多年历史最小/最大$NDVI$。

②TCI 是用当前 LST 与历史同期 LST 的最小/最大值归一化，反映温度相对异常：

$$TCI = \frac{BT_{max} - BT_{cur}}{BT_{max} - BT_{min}}$$

其中，BT_{cur}为当前地表温度（LST），BT_{min}、BT_{max}为该像元在同历时段的多年历史最小/最大 LST。

③VHI 是对 VCI 和 TCI 按一定权重线性组合，从而综合植被和温度双重信息：

$$VHI = \alpha \times VCI + (1 - \alpha) \times TCI$$

其中，α为 VCI 与 TCI 的权重比。

（2）VHI 计算时 VCI 与 TCI 的权重α的设置

论文中通过改变（1）③中 VCI 与 TCI 的权重α来计算 VHI 与 SPI-1/3/6 的像元尺度相关系数，并选择相关性最高的权重（VCI:TCI=0.7:0.3）作为长江中下游的最优方案。课程中并未讲过通过与 SPI 相关性来优化（1）③中权重α的思路。

在我理解（1）中 VCI、TCI、VHI 等指标的定义后，再去查阅利用 VCI、TCI、VHI 监测干旱的综述或具体应用的论文（Yang Shao’e et al., 2010[6]、Bokusheva et al., 2016[7]、Pei et al., 2018[8]等），看到很多研究都提到“VHI 权重需因地制宜”的观点，我理解到权重优化是一个需要用独立气象指标（如 SPI 等）来校准的问题。因此，VHI 的计算不是完全固定的公式，而是一个可以根据本地区气候与植被特征调整的综合指数，其中的超参数权重α设置的本质是用独立观测（气象指标 SPI 等）来检验遥感指数是否真正反映了干旱。

3.3 基于土地覆盖产品的植被干旱敏感性定量分析

（1）MCD12Q1 IGBP 的分类与本论文的再分类

课程中介绍了典型土地利用类型识别的相关主题，但主要是关于如何从影像上区分林地、农田、草地等，并未涉及论文中所用的 MCD12Q1 这样的全球土地覆盖产品及其分类体系。

为了了解相关信息，我先查看论文中的表 1：

表 1 土地覆盖原始分类(IGBP 分类)和再归类结果

Tab. 1 Original Classification (IGBP Classification) and the Reclassification of Land Cover Types

原始分类序号	IGBP 分类情况	再归类序号	再归类结果
1	常绿针叶林	1	林地
2	常绿阔叶林	1	林地
3	落叶阔叶林	1	林地
4	混交林	1	林地
5	多树草地	2	草地
6	稀树草地	2	草地
7	草地	2	草地
8	农田	3	农田
9	农田/自然植被镶嵌地	3	农田
10	城市和建筑用地	4	其他
11	永久湿地	4	其他
12	裸地	4	其他
13	水体	4	其他

再查阅 MODIS 土地覆盖产品 MCD12Q1 中的 IGBP 分类的相关信息（在 GEE 上找到的[9]）：

名称	单位	最小值	最大值	像素尺寸	说明
LC_Type1				米	土地覆盖类型 1：年度国际地圈-生物圈计划 (IGBP) 分类
LC_Type2				米	土地覆盖类型 2：马里兰大学 (UMD) 年度分类
LC_Type3				米	土地覆盖类型 3：年叶面积指数 (LAI) 分类
LC_Type4				米	土地覆盖类型 4：年度 BIOME-Biogeochemical Cycles (BGC) 分类
LC_Type5				米	土地覆盖类型 5：一年生植物功能类型分类
LC_Prop1_Assessment	%	0	100	米	LCCS1 土地覆盖层置信度
LC_Prop2_Assessment	%	0	100	米	LCCS2 土地利用层置信度
LC_Prop3_Assessment	%	0	100	米	LCCS3 地表水文层置信度
LC_Prop1				米	FAO-土地覆盖分类系统 1 (LCCS1) 土地覆盖层
LC_Prop2				米	FAO-LCCS2 土地用途层
LC_Prop3				米	FAO-LCCS3 地表水文层
QC				米	产品质量标志
LW				米	根据 MOD44W 得出的二元陆地（类别 2）/水体（类别 1）掩码

LC_Type1 类表

值	颜色	说明
1	#05450a	常绿针叶林：以常绿针叶乔木为主（树冠 >2 米）。树木覆盖率 >60%。
2	#086a10	常绿阔叶林：以常绿阔叶树和掌状树为主（树冠 >2 米）。树木覆盖率 >60%。
3	#54a708	落叶针叶林：以落叶针叶树（落叶松）为主（树冠 >2 米）。树木覆盖率 >60%。
4	#78d203	落叶阔叶林：以落叶阔叶树为主（树冠 >2 米）。树木覆盖率 >60%。
5	#009900	混交林：以落叶树和常绿树为主（各占 40-60%）的树种（树冠 >2 米）。树木覆盖率 >60%。
6	#c6b044	密灌丛：以木本多年生植物（高度 1-2 米）为主，覆盖率 >60%。
7	#dcd159	开阔的灌木丛：以木本多年生植物（高度 1-2 米）为主，覆盖率 10-60%。
8	#dade48	多树稀树草原：树木覆盖率 30-60%（树冠 >2 米）。
9	#fbff13	热带草原：树木覆盖率 10-30%（树冠 >2 米）。
10	#b6ff05	草原：以草本一年生植物（高度 <2 米）为主。
11	#27ff87	永久性湿地：永久性淹没的土地，水面覆盖率 30-60%，植被覆盖率 >10%。
12	#c24f44	耕地。
13	#a5a5a5	城市和建成区：至少 30% 的不透水表面积，包括建筑材料、沥青和车辆。
14	#ff6d4c	农田/天然植被镶嵌：小规模耕作占 40-60% 的镶嵌，其余为天然树木、灌木或草本植被。
15	#69fff8	永久性冰雪：至少 60% 的区域全年至少有 10 个月被冰雪覆盖。
16	#f9ffa4	植被覆盖率低于 10% 的区域（沙地、岩石地、土壤地）。
17	#1c0dff	水体：至少 60% 的面积被永久性水体覆盖。

由此我弄清楚了 IGBP 17 类地物的含义，并理解到：这篇论文为了便于进行干旱敏感性统计，把这 17 类地物它们重新归类为“林地”“草地”“农田”“其他”四大类。

（2）植被对干旱的敏感性量化思路

课程中没有专门讲植被对干旱的敏感性如何用遥感定量表示。这篇论文中采用的方法是：先用土地覆盖数据锁定不同植被类型的空间分布，再在每个月、每个干旱等级下统计各类植被受旱面积占该类总面积的比例，通过全年、不同年份的对比，判断哪类植被在干旱事件中受旱频率和严重程度更高，以此刻画敏感性。

为理解这一思路，我进一步查阅了关于农业干旱传播和植被响应的研究（Bai et al., 2025[10]），并结合植物生理学文献（Berry et al., 2010[11]）对根系深度、蒸散发调节等差异的解释，弄清楚了为什么森林、草地、农田会表现出不同的抗旱能力和敏感性。

3.4 多源数据融合的干旱趋势分析与气候变化背景解读

（1）VHI 年序列与线性趋势分析

课程中虽然提到过时间序列分析，但没有具体讲以省为单位构建年尺度 VHI 序列并求线性趋势斜率的做法。论文中通过拟合 2001~2019 年各省的 VHI 序列的斜率来判断湿润化/干旱化趋势，这需要一定的气候统计学基础。

我参考了研究区域干旱趋势的文献（Wang & Kong, 2021[12]、Zeng et al., 2020[13]），学习了如何用线性趋势和显著性检验来定量描述趋势的强弱，以及如何判断一个趋势脆弱但为正的区域仍然存在干旱风险。

（2）降水距平、温度年际变化率与蒸散发

论文将 VHI 趋势与降水距平、温度年际变化率结合讨论，指出江苏和上海虽然降水增加，但温度升高带来的蒸散发增强会抵消部分湿润化效果，这部分属于气候变化影响机理，课程中没有展开讲过。

我顺着论文中引用的 IPCC 报告（2018）[14]和 Dai(2013)[15]等文献，理解了在全球变暖背景下某些区域可能出现更湿但仍有干旱风险的情况，即由于可能会有降水增加与蒸散发增强并存的情况，所以其水分收支不一定显著改善。

（注：相关参考文献在下一页的“参考文献”下，点击方括号内的数字可以跳转）

参考文献

[1] Li Bin, Li Lijuan, Li Haibin, et al. Spatial and Temporal Variability of Droughts in the Lancang River Basin[J]. Transactions of the Chinese Society of Agricultural Engineering, 2011, 27（5）: 87-92

[2] Wang Xianwei, Liu Mei, Liu Lin. Responses of MODIS Spectral Indices to Typical Drought Events from 2000 to 2012 in Southwest China [J]. Journal of Remote Sensing, 2014, 18（2）: 432-452

[3] Qian Lili, He Zhonghua, Liang Hong, et al. Research Progress of Agricultural Drought Monitoring Methods and Indicator [J]. Journal of Green Science and Technology, 2019（6）: 5-8

[4] Huang Youxin, Liu Xiuguo, Shen Yonglin, et al. Advances in Remote Sensing Derived Agricultural Drought Monitoring Indices and Adaptability Evaluation Methods [J]. Transactions of the Chinese Society of Agricultural Engineering, 2015, 31（16）: 186-195

[5] Kogan F N. Application of Vegetation Index and Brightness Temperature for Drought Detection [J].Advances in Space Research, 1995, 15（11）: 91-100

[6] Yang Shao’e, Yan Nana, Wu Binfang, et al. Advances in Agricultural Drought Monitoring by Remote Sensing [J]. Remote Sensing Information, 2010, 25（1）: 103-109

[7] Bokusheva R, Kogan F, Vitkovskaya I, et al. Satellite Based Vegetation Health Indices as a Criteria for Insuring Against Drought-Related Yield Losses [J].Agricultural and Forest Meteorology, 2016, 220: 200-206

[8] Pei F S,Wu C J,Liu X P,et al. Monitoring the Vegetation Activity in China Using Vegetation Health Indices [J]. Agricultural and Forest Meteorology, 2018, 248: 215-227

[9] Google Earth Engine. MCD12Q1.061 MODIS Land Cover Type Yearly Global 500m. Google Developers. Retrieved November 18, 2025, from https://developers.google.cn/earth-engine/datasets/catalog/MODIS_061_MCD12Q1

[10] Bai, Y.-H., Chen, J., Zhang, Y.-W., & Tang, Z. (2025). Response modes of global vegetation to extreme drought. Global Change Biology. Advance online publication. https://doi.org/10.1111/gcb.70488

[11] Berry J A, Beerling D J, Franks P J. Stomata: Key Players in the Earth System, Past and Present [J].Current Opinion in Plant Biology, 2010, 13（3）: 232-239

[12] Wang A H, Kong X H. Regional Climate Model Simulation of Soil Moisture and Its Application in Drought Reconstruction Across China from 1911 to 2010 [J]. International Journal of Climatology, 2021, DOI: 10. 1002/joc. 6748

[13] Zeng Z Q, Wu W X, Li Y M, et al. Spatiotemporal Variations in Drought and Wetness from 1965 to 2017 in China [J]. Water, 2020, 12（8）: 2097

[14] The Intergovernmental Panel on Climate Change（IPCC）. Global Warming of 1.5℃: An IPCC Special Report [M]. Cambridge, UK: Cambridge University Press, 2018

[15] Dai A. Increasing Drought Under Global Warming in Observations and Models [J]. Nature Climate Change, 2013, 3（1）: 52-58

对水利遥感课程的感想、意见、建议等

武汉大学 遥感信息工程学院 杨丹阳

学号：2023302131259

一、课程感想

水利遥感课程系统地将遥感的物理基础、数据获取、分析方法与实际应用场景紧密结合，从电磁辐射与光谱特性出发，逐步讲解成像机理、图像处理与解译，再到植被与水体遥感等典型专题，课程结构清晰、内容连贯，使我进一步深化了对遥感原理及其应用的理解。

令我印象最深刻、也受益最大的是贯穿全课程的 R 语言实践环节。从基础语法到水利遥感中常用工具包的调用，查老师都细致指导我们操作，并耐心解答我提出的问题。此外，老师还拓展介绍了 GEE、Google Colab 等平台的使用，这些内容极大开阔了我的视野，也让我掌握了更多解决实际问题的实用技能，有效提升了我的动手能力与实践素养。

在授课过程中，老师经常结合自身科研项目案例辅助知识点讲解，生动展示了遥感技术在水利领域的实际应用价值，也让我了解到相关技术如何在真实场景中落地。作为一名遥感专业的学生，这样的教学方式对我专业认知和实践能力的提升帮助显著。

每次上课我都坐在第一排，深切体会到这门课对我学习的帮助。由衷感谢查老师细致的课堂讲解、精心的课程设计以及在实践中给予的耐心指导，您所传授的知识与经验将使我受益终身。

二、意见与建议

结合课程学习中的实际体验，我谨提出以下几点建议，供老师参考：

1. 关于遥感数据获取的指导

遥感数据的获取是实践过程中的重要环节。在我完成“基于 R 语言的 Landsat 8 遥感影像处理与分析”结课实习任务时，使用 GEE 平台下载影像的代码调试耗费了较多时间。老师在课堂上曾演示过 GEE 下载流程，建议未来可适当增加该部分的讲解比例，并在课程群中分享一个实际工程中较为标准的 GEE 影像下载脚本，供同学们参考，以减少大家在刚刚接触使用 GEE 平台下载影像时可能遇到的障碍。

此外，课程中虽介绍了多种遥感影像获取渠道（如部分 R 包及专门的影像下载网站），但目前相关内容较为分散，实践手册中提供的链接也不够完整。建议系统整理一份完整的遥感影像获取途径清单，并对一些较为“隐蔽”的方法附上简要下载教程，方便同学们在实习及相关任务中查阅使用。

2. 关于拓展介绍的内容

老师在课程中拓展讲解的 GEE、Google Colab 等内容，有效拓宽了我们的技术视野。建议未来可适当增加相关工具与技能的授课课时，引入更多水利遥感中可能用到的平台或方法。这对于水利水电学院及其他相关学院的同学都将具有重要的实用价值。

以上建议若有考虑不周之处，敬请谅解。师恩难忘，有幸受教，再次衷心感谢查老师一直以来的悉心指导与无私帮助！

杨丹阳

2025 年 11 月 5 日