教程: CHAP 数据库空气污染数据提取及使用： 从.nc 到变量

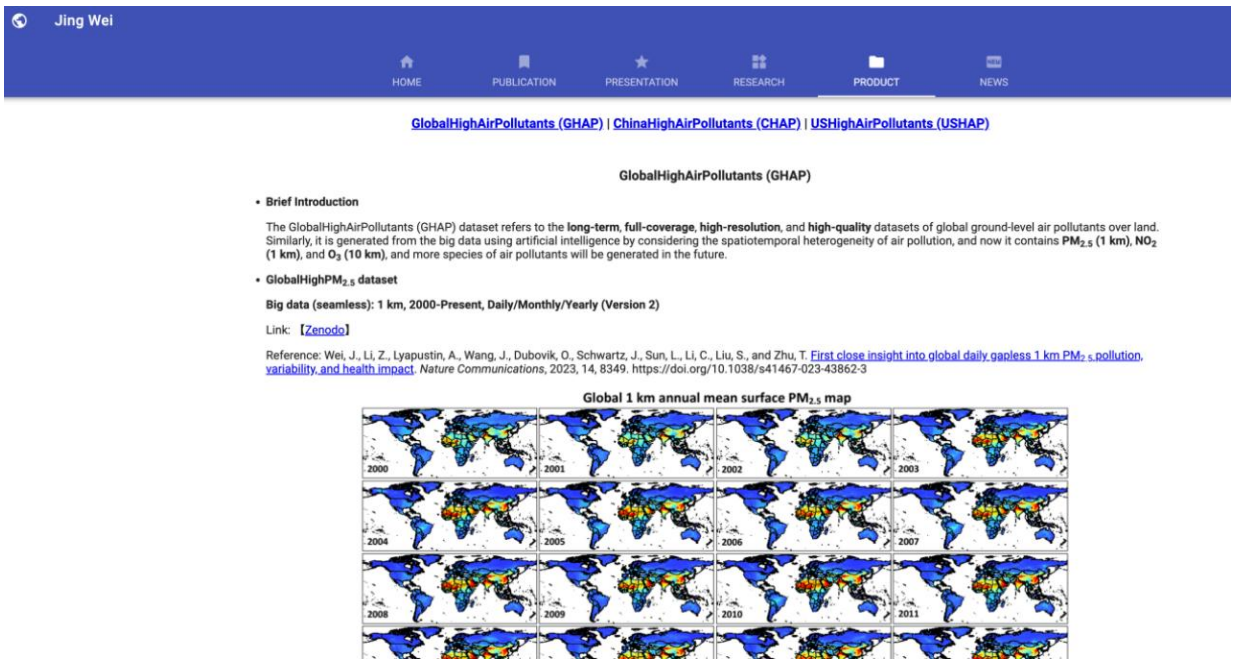
思路概览

1. 从 CHAP 数据库下载.nc 格式数据
2. 用 Python 将.nc 格式数据批量转为.tif 格式
3. 将.tif 格式文件用 ENVI 软件批量缩放&剪裁
4. Python 调用 ArcGIS 批量转换栅格数据为 Excel 表格
5. R 合并数据 + 动态提取构造特定时间窗的变量

1. 从 CHAP 数据库下载.nc 格式数据

①从以下地址访问 CHAP 数据库

https://weijing-rs.github.io/product.html



②选取目标空气污染物： 空间颗粒度 × 时间颗粒度 × 时间范围， 点击数据库存储链接

- **ChinaHigh$PM_{2.5}$ dataset**

[1] Big data (seamless): 1 km, 2000-Present, Daily/Monthly/Yearly (Version 4)

Link: 【Zenodo】, 【国家地球系统科学数据中心】, 【国家青藏高原科学数据中心】

Reference:

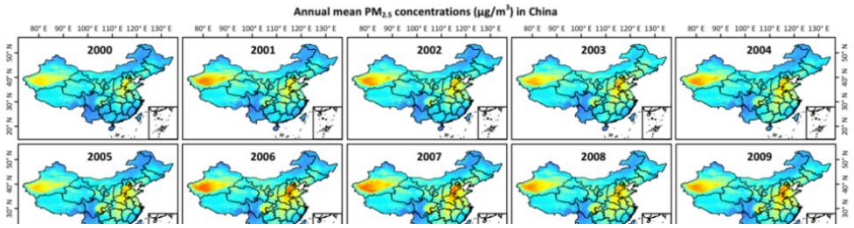
Wei, J., Li, Z., Lyapustin, A., Sun, L., Peng, Y., Xue, W., Su, T., and Cribb, M. Reconstructing 1-km-resolution high-quality $PM_{2.5}$ data records from 2000 to 2018 in China: spatiotemporal variations and policy implications. *Remote Sensing of Environment*, 2021, 252, 112136. https://doi.org/10.1016/j.rse.2020.112136 **(ESI Hot and Highly Cited Paper, Journal Most Cited Articles since 2019/2020, Top 100 Most Cited Chinese Papers Published in International Journals, ESSIC 2022 Best Paper Award)**

Wei, J., Li, Z., Cribb, M., Huang, W., Xue, W., Sun, L., Guo, J., Peng, Y., Li, J., Lyapustin, A., Liu, L., Wu, H., and Song, Y. Improved 1 km resolution $PM_{2.5}$ estimates across China using enhanced space-time extremely randomized trees. *Atmospheric Chemistry and Physics*, 2020, 20, 3273–3289. https://doi.org/10.5194/acp-20-3273-2020 **(ESI Hot and Highly Cited Paper)**

[2] Himawari-8: Eastern China, 5 km, 2018, Hourly (Version 1)

Link: 【Zenodo】

Reference: Wei, J., Li, Z., Pinker, R., Wang, J., Sun, L., Xue, W., Li, R., and Cribb, M. Himawari-8-derived diurnal variations of ground-level $PM_{2.5}$ pollution across China using a fast space-time Light Gradient Boosting Machine (LightGBM). *Atmospheric Chemistry and Physics*, 2021, 21, 7863–7880. https://doi.org/10.5194/acp-21-7863-2021 **(ESI Highly Cited Paper)**



③数据说明：

ATBD_ChinaHighPM2.5.pdf	今天 11:10	今天 11:10	3.3 MB	PDF文稿	今天 11:10
CHAP_PM2.5_D1K_2000_V4	2024/9/30 17:08	2024/9/30 17:08	--	文件夹	今天 11:04
CHAP_PM2.5_M1K_2000_V4	今天 11:03	今天 11:03	--	文件夹	今天 11:04
CHAP_PM2.5_Y1K_2000_V4.nc	今天 11:04	今天 11:04	7.3 MB	文稿	今天 11:04
nc2geotiff codes	今天 11:09	今天 11:09	--	文件夹	今天 11:09

D1K：Day_1Km：日均 × 1km

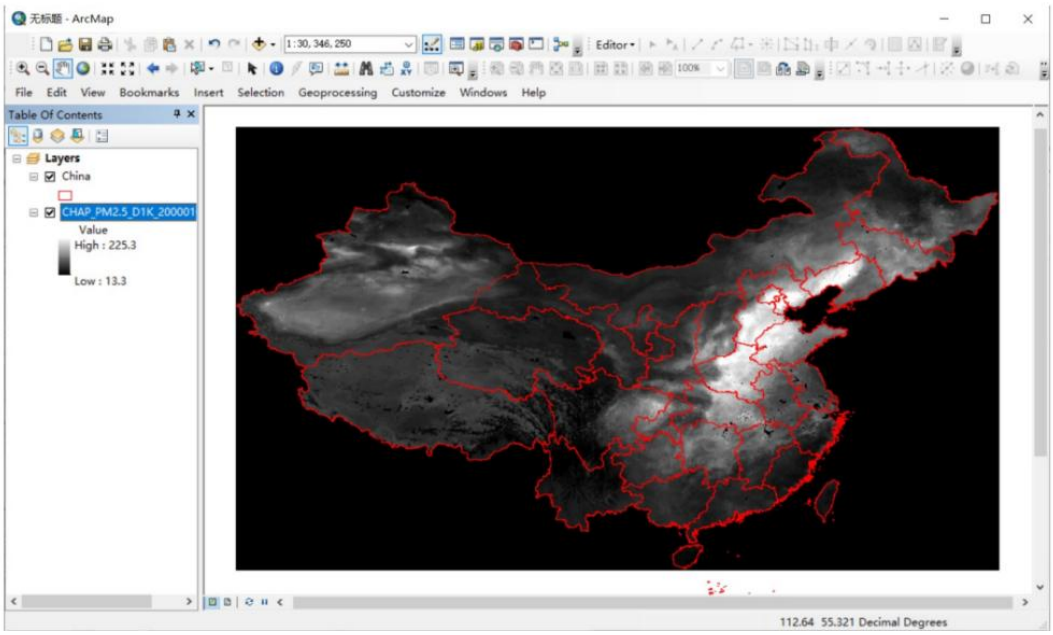
M1K：Month_1Km：月均 × 1km

Y1K：Year_1Km： 年均 × 1km

ATBD_ChinaHighPM2.5.pdf 中是数据库说明，其中提到如何将.nc 转为.tif：

How to read?

Use Python, Matlab, and IDL codes (nc2geotiff codes.rar) to batch convert NetCDF (.nc) to GeoTIFF (.tif).



nc2geotiff codes 文件夹中是配套的将.nc 文件批量转为.tif 文件的三种格式：

选取自己熟悉的软件代码即可，eg: IDL 和 Python 我都会，matlab 比较生。

2. 用 Python 将.nc 格式数据批量转为.tif 格式

Python 代码如下：

```
# 导入所需库
import os  # 用于文件路径操作
import netCDF4 as nc  # 用于处理 NetCDF 格式的数据
import numpy as np  # 用于处理数值计算和数组操作
from glob import glob  # 用于匹配文件路径模式
from osgeo import osr  # 用于设置投影信息
from osgeo import gdal # gdal 的正确导入
```

```
# 定义工作路径和输出路径
# 这里原始用的 F 盘
WorkPath = r'D:\2024.10.07 Convert nc to tif\CHAP_PM2.5_1KM\YEAR_2000_2021_nc'  # 工作路径，存储 .nc 文件

OutPath = r'D:\2024.10.07 Convert nc to tif\CHAP_PM2.5_1KM\YEAR_2000_2021_tif'  # 输出路径，存储 .tif 文件

# 定义空气污染物类型（例如 PM1, PM2.5, PM10, O3, NO2, SO2, 和 CO 等）
AP = 'PM2.5'  # 在 NetCDF 文件中提取 PM2.5 数据

# 定义空间分辨率（例如，1 km ≈ 0.01 度）
SP = 0.01  # 定义空间分辨率，单位为度（Degree）

# 如果输出路径不存在，则创建该目录
if not os.path.exists(OutPath):
    os.makedirs(OutPath)

# 搜索工作路径下所有 .nc 文件（NetCDF 文件）
path = glob(os.path.join(WorkPath, '*.nc'))

# 遍历每个 .nc 文件进行处理
for file in path:
    f = nc.Dataset(file)  # 打开 NetCDF 文件
    # 读取指定污染物（AP，PM2.5）的数据，并转换为 NumPy 数组
    data = np.array(f[AP][:])
    # 将数据中的无效值（65535）替换为 NaN
    data[data == 65535] = np.nan
```

```
    # 读取经度和纬度信息
    lon = np.array(f['lon'][:])
    lat = np.array(f['lat'][:])
    # 获取经度和纬度的最小和最大值
    LonMin, LatMax, LonMax, LatMin = lon.min(), lat.max(), lon.max(), lat.min()
    # 获取纬度和经度的数量
    N_Lat = len(lat)
    N_Lon = len(lon)
    # 定义经度和纬度的分辨率
    Lon_Res = SP
    Lat_Res = SP

    # 定义输出文件名，去掉 .nc 扩展名，添加 .tif 扩展名
    fname = os.path.basename(file).split('.nc')[0]
    outfile = OutPath + '/{}.tif'.format(fname)

    # 创建 GeoTIFF 文件，用来存储处理后的栅格数据
    driver = gdal.GetDriverByName('GTiff')
    outRaster = driver.Create(outfile, N_Lon, N_Lat, 1, gdal.GDT_Float32)  # 创建一个浮点数
32 位的单波段图像
    # 设置 GeoTIFF 的地理变换参数：左上角坐标、分辨率等
    outRaster.SetGeoTransform([LonMin - Lon_Res / 2, Lon_Res, 0, LatMax + Lat_Res / 2, 0, -
Lat_Res])
    # 定义投影为 WGS84
    sr = osr.SpatialReference()
    sr.SetWellKnownGeogCS('WGS84')
    outRaster.SetProjection(sr.ExportToWkt())
    # 将数据写入 GeoTIFF 文件
    outRaster.GetRasterBand(1).WriteArray(data)
    print(fname + '.tif', ' Finished')
```

```python
# 释放内存，关闭文件
del outRaster
f.close()
```

Python

3. 将.tif 格式文件用 ENVI 软件批量缩放&剪裁

3.1 下载目标地区 shp 文件

现成的 shp 文件从 CSDN 的这个网址下载：

https://blog.csdn.net/weixin_61129400/article/details/142142844

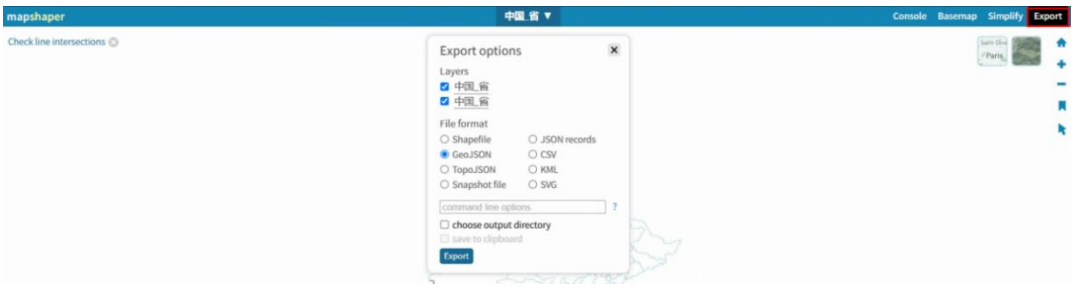
具体原理：

①从天地网网站服务中心-行政区域可视化板块，下载中国行政区划 GeoJSON 文件：

https://cloudcenter.tianditu.gov.cn/administrativeDivision

②使用 GeoJSON 转 shp 工具，将 GeoJSON 文件转为 shp 文件：

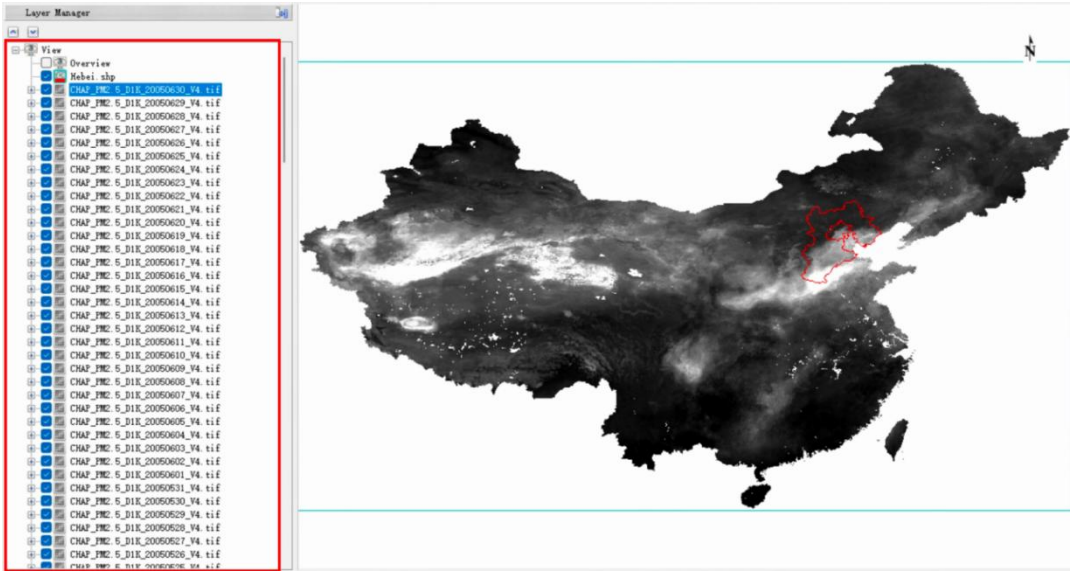
https://mapshaper.org/



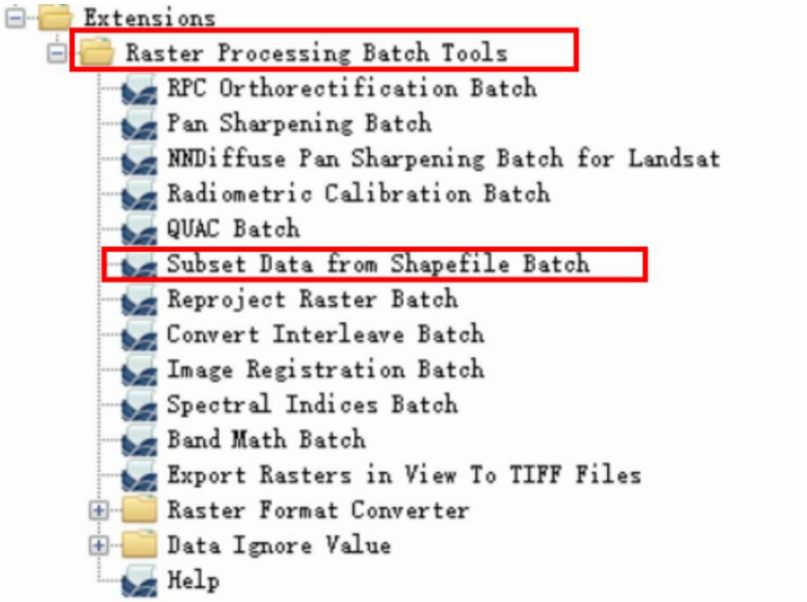
3.2 ENVI 缩放栅格值&以缓冲区为边界剪裁(.dat)

3.2.2 批量处理方法

①将所有 tif 数据以及目标区域 shp 数据导入左侧栏：外加目标区域.shp



②首先需要安装 ENVI 软件的 App Store（软件商店），然后安装其中的 ENVI 栅格图像批处理工具包插件：Raster Processing Batch Tools 工具包，用到其中的：Subset Data from Shapefile Batch；

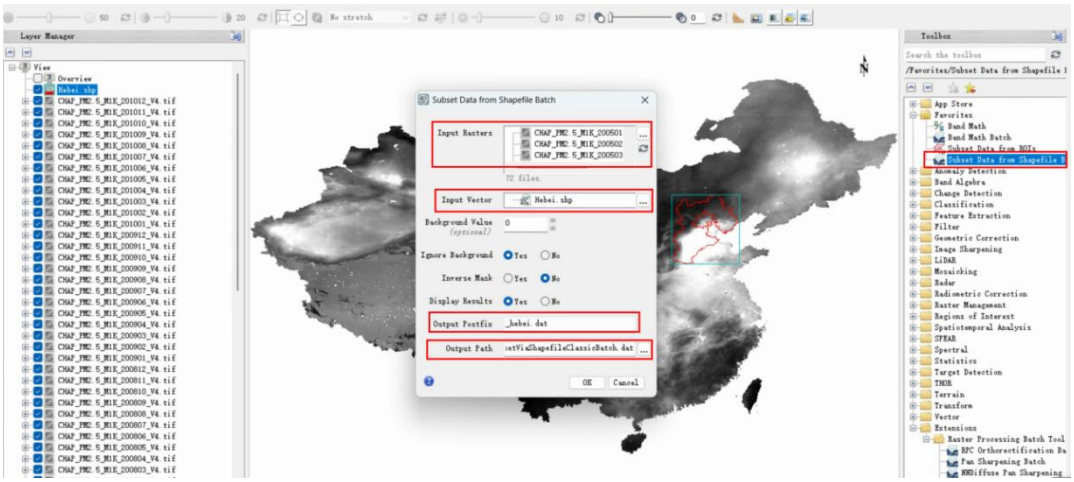


③使用工具包中的 Subset Data from Shapefile Batch（根据 Shapefile 批处理提取数据子集）插件进行批量栅格剪裁。

- 选取待处理的所有栅格数据，
- 输入目标区域行政区域的 Shapefile 文件作为剪裁掩膜，

- 设置输出后缀，默认后缀为.dat，可以将其手动改为.tif，这样可以省去用 IDL 转换.dta 为.tif 这一步！ 【这一步千万不能动.dat 格式，应先顺着生成.dazt 格式，然后用 IDL 转成.tif；若强行改后缀为.tif，会得到.tif 文件，但是像素为 0！！！】

- 设置输出路径。



Notes:

1. ENVI 内部存储空间位 32000M，所以太大的数据只能分批导入然后处理，无法做到一次性全部导入。也无法做到一次性全部处理。只能切成小片小片，一点一点跑。

2. 6 年的 DAY 数据，nc 状态文件大小约 15G，转换为 tif 格式后约为 170G，此外进行剪裁会产生.enp 后缀的工程文件，大小约为 20G；如果这些内存超过了磁盘内

存，则 ENVI 软件会不停闪退，无法导入数据。

（https://wenku.csdn.net/answer/3q6b580tih）

3.3 IDL 批量转换 dat 为 tif 文件

这种方式是安全的，不会破坏 tif 文件使其像素为 0。

IDL 代码：

```
;+
; :Description:
;    Batch Convert Dat to Tif
;
; :Author: YangZeping
; :Date: 2024.05.07
;-

pro dat_to_tif
  envi = ENVI()                          ; 初始化 ENVI 对象

  dat_dir =
'C:\Users\22671\Desktop\YangZepings_IDL_Files\20241101_Batch_Convert_Dat_to_Tif_CHAP_Hebei\CHAP_PM25_1KM\MONTH_2005_2010_hebei_dat\'          ; 设置.dat 文件的目录
  tif_dir =
'C:\Users\22671\Desktop\YangZepings_IDL_Files\20241101_Batch_Convert_Dat_to_Tif_CHAP_Hebei\CHAP_PM25_1KM\MONTH_2005_2010_hebei_tif\'     ; 设置输出.tif 文件的目录
  file=file_search(dat_dir,'*dat',count = num); 在 dat_dir 中搜索所有.dat 文件，并返回文件列表及文件个数

  for i=0, num-1 do begin                  ; 对于每个找到的.dat 文件执行循环
    raster = envi.OpenRaster(file[i])        ; 打开当前.dat 文件作为 raster 对象
    dat_filename = strsplit(file[i], '\' ,/EXTRACT)  ; 从文件路径中提取文件名
```

```
    dat_filename = dat_filename[-1]          ; 获取文件路径中的最后一个元素（即文件名）

    ; 改进部分：移除文件名中的".dat"扩展名
    dat_filename = strmid(dat_filename, 0, strlen(dat_filename) - 4) ; 移除文件名最后的四个字符(".dat")

    tif_filename = dat_filename + '.tif'      ; 将文件扩展名改为.tif
    tif_path = tif_dir + tif_filename         ; 构造输出的完整.tif 文件路径
    raster.Export, tif_path, 'TIFF'           ; 导出 raster 为 TIFF 格式到指定路径
  endfor                                      ; 结束循环
  print,'Process Over'                        ; 打印处理完成的消息
end
```

Stata Ado

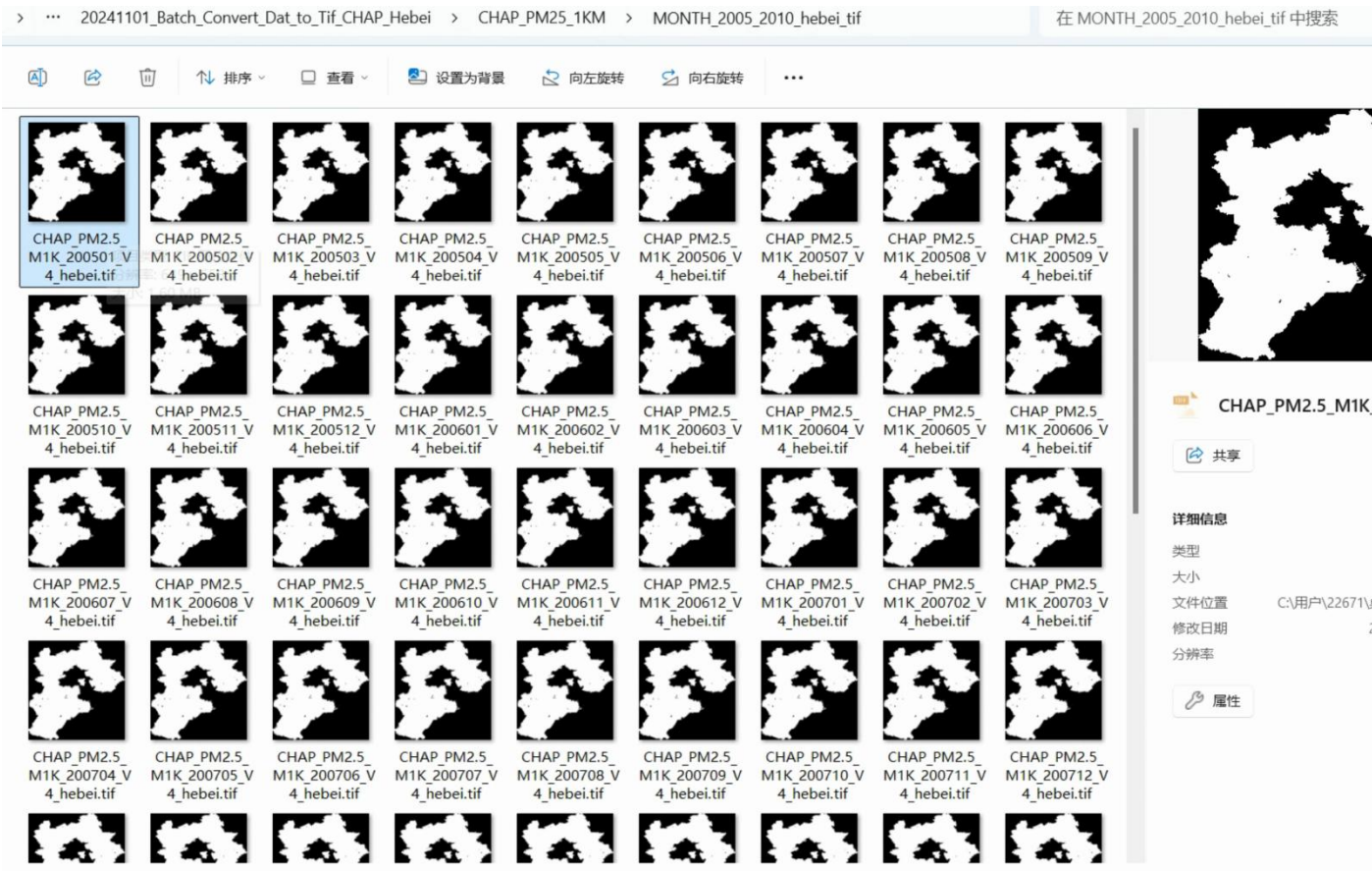
在命令行中输入↓命令以运行脚本：

```
.compile 'C:\Users\22671\Desktop\YangZepings_IDL_Files\20241101_Batch_Convert_Dat_to_Tif_CHAP_Hebei\20241101_Batch_Convert_Dat_to_Tif_CHAP_Hebei_PM25_Month.pro'
dat_to_tif
```

Stata Ado

运行结果：

```
IDL> .compile 'C:\Users\22671\Desktop\YangZepings_IDL_Files\20241101_Batch_Convert_Dat_to_Tif_CHAP_Hebei\20241101_Batch_Convert_Dat_to_Tif_CHAP_Hebei_PM
% Compiled module: DAT_TO_TIF.
IDL> dat_to_tif
% Restored file: ENVI.
% Loaded DLM: HPGRAPHICS.
% Loaded DLM: XML.
% Loaded DLM: PNG.
% Loaded DLM: URL.
% Loaded DLM: NATIVE.
% Loaded DLM: MAP_PE.
Process Over
ENVI>
```



4. Python 调用 ArcGIS 批量转换栅格数据为 Excel 表格

Optional: 处理掉 tif 文件名中的 . 将其替换为: _或者不要。

例如 PM2.5.tif 应转换为 PM25.tif; PM10 则不需要转。

Python 代码如下:

```
# coding=utf-8

import os

# 设置包含 tif 文件的目录
directory = r"C:\Users\22671\Desktop\YangZepings Python Files\20241031 convert tif to
```

```
xlsx\CHAP_PM2.5_1KM\MONTH_2005_2010_hebei_tif"

# 遍历目录中的所有文件
for filename in os.listdir(directory):
    if filename.endswith(".tif"):  # 确认文件是 tif 文件
        # 将文件名和扩展名分离
        name_part, extension = os.path.splitext(filename)

        # 替换文件名中的所有点为下划线，除了扩展名前的最后一个点
        new_name_part = name_part.replace('.', '')

        # 重新组合新的文件名和扩展名
        new_filename = new_name_part + extension

        # 构建完整的旧文件路径和新文件路径
        old_file = os.path.join(directory, filename)
        new_file = os.path.join(directory, new_filename)

        # 重命名文件
        os.rename(old_file, new_file)
        print("Renamed '{filename}' to '{new_filename}'")
```

Python

4.1 使用 Mask 功能去除黑色区域

Python 代码如下：

```
# coding=utf-8
```

```
import arcpy # 导入 ArcPy 库，它是与 ArcGIS Desktop 交互的 Python 库

arcpy.CheckOutExtension('spatial') # 检查并借出空间分析工具的许可证

arcpy.gp.overwriteOutput = 1 # 设置工具执行的输出可以覆盖现有文件

arcpy.env.workspace = "C:\\Users\\22671\\Desktop\YangZepings Python Files\\20241031 convert tif to xlsx\\CHAP_PM25_1KM\\MONTH_2005_2010_hebei_tif" # 设置 ArcPy 环境的工作空间，所有文件操作都在这个目录下

# 裁剪栅格
rasters = arcpy.ListRasters('*', 'tif') # 列出工作空间中所有的 tif 格式栅格文件
arcpy.AddMessage(rasters) # 将栅格文件列表输出到 ArcGIS 的消息窗口

# 裁剪 shp
mask = "C:\\Users\\22671\\Desktop\\YangZepings Python Files\\20241031 convert tif to xlsx\\Hebei\\Hebei.shp" # 设置用于裁剪的形状文件（shp）的路径

for raster in rasters: # 遍历所有栅格文件
    print(raster) # 在控制台打印当前处理的栅格文件名
    out = "C:\\Users\\22671\\Desktop\YangZepings Python Files\\20241031 convert tif to xlsx\\CHAP_PM25_1KM\\MONTH_2005_2010_hebei_tif_Mask\\" + raster[:-4] + "_MASK.tif" # 设置输出文件的路径及名称

    arcpy.gp.ExtractByMask_sa(raster, mask, out) # 使用指定的 shp 文件对栅格文件进行裁剪，并保存到指定的输出路径

print("OK") # 当所有栅格文件处理完成后在控制台输出"OK"
```

Python

4.2 将栅格 tif 转为矢量点 shp

Python 代码如下：

```
# coding=utf-8

import arcpy # 导入 ArcPy 库，用于执行地理空间数据分析、数据转换等
from arcpy import env # 从 arcpy 导入 env 模块，用于环境设置
from arcpy.sa import * # 从 arcpy 的空间分析模块导入所有功能
import os # 导入 os 模块，用于处理文件和目录
import os.path # 导入 os.path 模块，用于路径相关操作
import sys, string # 导入 sys 和 string 模块，sys 用于访问与 Python 解释器紧密相关的变量和函数，string 用于常见的字符串操作

arcpy.gp.overwriteOutput = 1 # 设置工具执行的输出可以覆盖现有文件

dir = r"C:\Users\22671\Desktop\YangZepings Python Files\20241031 convert tif to xlsx\CHAP_PM25_1KM\MONTH_2005_2010_hebei_tif_Mask/" # 设置要处理的栅格数据所在的目录；此处 r 是声明\保留原意而非作为转义符；结尾正斜杠是所以系统都接受的路径分隔符

filenames = os.listdir(dir) # 列出指定目录下的所有文件名
for filename in filenames: # 遍历文件名列表
    if os.path.splitext(filename)[1] == '.tif': # 检查文件扩展名是否为.tif，由于文件名中自带 2 个点，总计 3 个点，因此后缀位于 0123 的 3
        inRaster = dir + filename # 构建完整的栅格文件路径
        Output_Workspace = r"C:\Users\22671\Desktop\YangZepings Python Files\20241031
```

```
convert tif to xlsx\CHAP_PM25_1KM\MONTH_2005_2010_hebei_tif_Mask_shp/" # 设置输出点文件的工作空间
    basename = filename[:-4] + '_Point' # 从文件名移除扩展名，并添加'_Point'作为新的基础名
    outPoint = Output_Workspace + basename + '.shp' # 构建输出点矢量文件的完整路径
    field = "VALUE" # 设置用于点矢量数据的字段名
    print(inRaster, outPoint, field)
    arcpy.RasterToPoint_conversion(inRaster, outPoint, field) # 执行栅格到点的转换
print("ok") # 处理完成后在控制台输出"ok"
```

Python

4.3 将 shp 文件的 Attribute Table 批量转换为 dbf 表格

以方便后续的批量处理，此文件类型可被 R 的 rio 包读入为 dataframe

Python 代码如下：

```
# coding=utf-8

import arcpy
import os

# 设置工作空间目录（包含待转换的地理数据文件）
input_folder = r"C:\Users\22671\Desktop\YangZepings Python Files\20241031 convert tif to xlsx\CHAP_PM25_1KM\MONTH_2005_2010_hebei_tif_Mask_shp"
output_folder = r"C:\Users\22671\Desktop\YangZepings Python Files\20241031 convert tif to xlsx\CHAP_PM25_1KM\MONTH_2005_2010_hebei_tif_Mask_shp_dbf"

# 创建输出文件夹（如果不存在）
if not os.path.exists(output_folder):
```

```python
    os.makedirs(output_folder)

# 获取输入文件夹中的所有要素类（例如 shapefile）
arcpy.env.workspace = input_folder
feature_classes = arcpy.ListFeatureClasses()

# 遍历每个要素类并导出属性表为 .dbf 文件
for fc in feature_classes:
    # 获取要素类名称并设置输出 .dbf 文件路径
    fc_name = os.path.splitext(os.path.basename(fc))[0]
    dbf_output_path = os.path.join(output_folder, "{}.dbf".format(fc_name))

    # 将属性表导出为 .dbf 文件
    arcpy.TableToTable_conversion(fc, output_folder, "{}.dbf".format(fc_name))

    print("Exported {} to {}".format(fc, dbf_output_path))

print("属性表批量转换为 dbf 完成！")
```

Python

不过需要注意，此得到的 dbf 文件需要经过以下 R 代码处理才能使用：

```r
# import
dt <- rio::import("CHAP_PM25_M1K_200501_V4_hebei_MASK_Point.dbf")

# pointid to FID
dt3$pointid <- dt3$pointid - 1
names(dt3)[1] <- "FID"
 ## 做完这两步其实已经是 R 中的 Dataframe 格式，并且数据准确，可以直接使用!
```

```R
# export
rio::export(dt, "CHAP_PM25_M1K_200501_V4_hebei_MASK_Point.xlsx")
```

4.4 匹配住址 Address 中的 FID 与属性表/dbf 中的 FID

方便后续将住址批量匹配空气污染数据，方便进行数据合并 (所有时点空气污染数据合并到一个文件且带 ID)

Python 代码如下：

```python
# coding=utf-8

# 导入所需的 Python 库
import arcpy
import os

# 允许覆盖现有输出文件
arcpy.env.overwriteOutput = True

# 设置工作空间
arcpy.env.workspace = r"C:\Users\22671\Desktop\YangZepings Python Files\20241031 convert tif to xlsx\CHAP_PM25_1KM\MONTH_2005_2010_hebei_Match_ID_to_FID"

# 复制并重命名原始的 shp 文件以进行分析
original_shp = "Address_18369.shp"
new_shp = "Address_18369_ID_Match_CHAP_PM25_M1K_195307_FID.shp"
arcpy.CopyFeatures_management(original_shp, new_shp)

# 进行近邻分析
in_features = new_shp
near_features = "CHAP_PM25_M1K_200501_V4_hebei_MASK_Point.shp"
```

```
search_radius = "1000 Meters"


arcpy.Near_analysis(in_features, near_features, search_radius, "", "", "PLANAR")
print("Near analysis completed successfully.")


# 将近邻分析得到的结果 shp 文件导出为 xls
xls = "Address_18369_ID_Match_CHAP_PM25_M1K_195307_FID.xls"
arcpy.TableToExcel_conversion(new_shp, xls)
print("Shapefile successfully exported to Excel.")
```

Python

5 使用 R 语言合并数据 + 动态提取构造特定时间窗的变量

5.1 R 语言合并数据

使用以下 R 语言合并多个时间点的变量数据:

```
# 04 合并 2005-2010 月均 PM 数据为一整体 ----
## 04-1 PM25 ----
# 加载所需的包
{
  library(rio)
  library(dplyr)
}


# 设置数据路径和结果路径
Data_dir <- "/Users/yangzeping/Desktop/YangZeping's R Files/2024.09.09 毕业设计-河北RCT/Data/CHAP_PM25_1KM_MONTH_2005_2010_hebei_dbf"
Data1_dir <- "/Users/yangzeping/Desktop/YangZeping's R Files/2024.09.09 毕业设计-河北RCT/Data"
Results_dir <- "/Users/yangzeping/Desktop/YangZeping's R Files/2024.09.09 毕业设计-河北
```

```
RCT/Results"

setwd(Data_dir)



# 读取主数据集
main_df <- rio::import("ID_CHAP_PM25_M1K_FID.xls")

# 定义月份列表
months <- format(seq(as.Date("2005-01-01"), as.Date("2010-12-01"), by = "month"),
"%Y%m")

# 循环处理每个月份的数据
for (ym in months) {
  dbf_filename <- sprintf("CHAP_PM25_M1K_%s_V4_hebei_MASK_Point.dbf", ym)

  # 读取 dbf 文件并进行预处理
  month_df <- import(file.path(Data_dir, dbf_filename))
  month_df$pointid <- month_df$pointid - 1
  names(month_df) <- c("FID", paste0("PM25_1km_m_", ym))  # 此处 【PM25 需要根据变量
进行修改】

  # 合并数据集，使用 dplyr 的 left_join 进行匹配
  main_df <- main_df %>%
    left_join(month_df, by = "FID")
}

# 剔除 FID 这列 (已无用)
library(dplyr)
main_df <- main_df %>%
```

```
  select(-FID)

# 结果保存
setwd(Data1_dir)
rio::export(main_df, "PM25_200501_201012_1km_monthly.rds")

setwd(Results_dir)
rio::export(main_df, "PM25_200501_201012_1km_monthly.xlsx")
rio::export(main_df, "PM25_200501_201012_1km_monthly.rds")

## 04-2 PM10 ----
# 加载所需的包
{
  library(rio)
  library(dplyr)
}

# 设置数据路径和结果路径
Data_dir <- "/Users/yangzeping/Desktop/YangZeping's R Files/2024.09.09 毕业设计-河北RCT/Data/CHAP_PM10_1KM_MONTH_2005_2010_hebei_dbf"
Data1_dir <- "/Users/yangzeping/Desktop/YangZeping's R Files/2024.09.09 毕业设计-河北RCT/Data"
Results_dir <- "/Users/yangzeping/Desktop/YangZeping's R Files/2024.09.09 毕业设计-河北RCT/Results"

setwd(Data_dir)



# 读取主数据集
main_df <- rio::import("ID_CHAP_PM10_M1K_FID.xls")
```

```r
# 定义月份列表
months <- format(seq(as.Date("2005-01-01"), as.Date("2010-12-01"), by = "month"),
"%Y%m")

# 循环处理每个月份的数据
for (ym in months) {
  dbf_filename <- sprintf("CHAP_PM10_M1K_%s_V4_hebei_MASK_Point.dbf", ym)

  # 读取 dbf 文件并进行预处理
  month_df <- import(file.path(Data_dir, dbf_filename))
  month_df$pointid <- month_df$pointid - 1
  names(month_df) <- c("FID", paste0("PM10_1km_m_", ym)) # 此处 【PM10 需要根据变量
进行修改】

  # 合并数据集，使用 dplyr 的 left_join 进行匹配
  main_df <- main_df %>%
    left_join(month_df, by = "FID")
}

# 剔除 FID 这列 (已无用)
library(dplyr)
main_df <- main_df %>%
  select(-FID)

# 结果保存
setwd(Data1_dir)
rio::export(main_df, "PM10_200501_201012_1km_monthly.rds")

setwd(Results_dir)
```

```R
rio::export(main_df, "PM10_200501_201012_1km_monthly.xlsx")
rio::export(main_df, "PM10_200501_201012_1km_monthly.rds")
```

5.2 使用 R 语言动态提取并构造特定时间窗的变量

使用以下 R 代码动态提取并构造特定时间窗的变量:

```R
# 06-1 构造 Monthly 1st_trimester 2nd_trimester 3rd_trimester ave_first_3_trimester PM25 PM10_规避 3rd trimester 溢出问题
 ## Author: YangZeping
 ## Date: 2024-11-09

# 01 根据 LMP 计算时间窗所需变量 ----
## 加载包 ----
{
 library(rio)
 library(tidyverse)
 library(lubridate)
}

## 设置路径 ----
Data_dir <- "/Users/yangzeping/Desktop/YangZeping's R Files/2024.09.09 毕业设计-河北 RCT/Data"
Results_dir <- "/Users/yangzeping/Desktop/YangZeping's R Files/2024.09.09 毕业设计-河北 RCT/Results"
setwd(Data_dir)

### PM25_200501_201012_1km_monthly ----
# 数据导入
dt <- rio::import("Id_LMP_Birth_date_Gest_day_Gest_wk_Gest_mth_18089_20241107.rds")
```

```r
# 空气污染数据导入
PM25 <- rio::import("PM25_200501_201012_1km_monthly.rds")
dt_PM25 <- left_join(dt, PM25, by = "ID")

# 转换日期格式并生成年份和月份变量
dt_PM25 <- dt_PM25 %>%
  mutate(
    LMP_date = ymd(LMP),
    lmp_year = year(LMP_date),
    lmp_month = month(LMP_date),

    # 孕前 9 个月的年份和月份变量
    y_pre_9m = lmp_year, y_pre_8m = lmp_year, y_pre_7m = lmp_year,
    y_pre_6m = lmp_year, y_pre_5m = lmp_year, y_pre_4m = lmp_year,
    y_pre_3m = lmp_year, y_pre_2m = lmp_year, y_pre_1m = lmp_year,

    m_pre_9m = lmp_month - 9, m_pre_8m = lmp_month - 8, m_pre_7m = lmp_month - 7,
    m_pre_6m = lmp_month - 6, m_pre_5m = lmp_month - 5, m_pre_4m = lmp_month - 4,
    m_pre_3m = lmp_month - 3, m_pre_2m = lmp_month - 2, m_pre_1m = lmp_month - 1,

    # 孕后 12 个月的年份和月份变量
    y_post_1m = lmp_year, y_post_2m = lmp_year, y_post_3m = lmp_year,
    y_post_4m = lmp_year, y_post_5m = lmp_year, y_post_6m = lmp_year,
    y_post_7m = lmp_year, y_post_8m = lmp_year, y_post_9m = lmp_year,
    y_post_10m = lmp_year, y_post_11m = lmp_year, y_post_12m = lmp_year,

    m_post_1m = lmp_month, m_post_2m = lmp_month + 1, m_post_3m = lmp_month + 2,
    m_post_4m = lmp_month + 3, m_post_5m = lmp_month + 4, m_post_6m = lmp_month + 5,
    m_post_7m = lmp_month + 6, m_post_8m = lmp_month + 7, m_post_9m = lmp_month + 8,
```

```
    m_post_10m = lmp_month + 9, m_post_11m = lmp_month + 10, m_post_12m = lmp_month
+ 11
  )

# 处理跨年数据
dt_PM25 <- dt_PM25 %>%
  mutate(
    across(
      .cols = starts_with("y_"),
      .fns = ~ ifelse(get(sub("y_", "m_", cur_column())) <= 0, . - 1,
                      ifelse(get(sub("y_", "m_", cur_column())) > 12, . + 1, .))
    ),
    across(
      .cols = starts_with("m_"),
      .fns = ~ ifelse(. <= 0, . + 12, ifelse(. > 12, . - 12, .))
    )
  )

# 生成动态变量名
dt_PM25 <- dt_PM25 %>%
  mutate(
    PM25_pre_9m = paste0("PM25_1km_m_", y_pre_9m, sprintf("%02d", m_pre_9m)),
    PM25_pre_8m = paste0("PM25_1km_m_", y_pre_8m, sprintf("%02d", m_pre_8m)),
    PM25_pre_7m = paste0("PM25_1km_m_", y_pre_7m, sprintf("%02d", m_pre_7m)),
    PM25_pre_6m = paste0("PM25_1km_m_", y_pre_6m, sprintf("%02d", m_pre_6m)),
    PM25_pre_5m = paste0("PM25_1km_m_", y_pre_5m, sprintf("%02d", m_pre_5m)),
    PM25_pre_4m = paste0("PM25_1km_m_", y_pre_4m, sprintf("%02d", m_pre_4m)),
    PM25_pre_3m = paste0("PM25_1km_m_", y_pre_3m, sprintf("%02d", m_pre_3m)),
    PM25_pre_2m = paste0("PM25_1km_m_", y_pre_2m, sprintf("%02d", m_pre_2m)),
    PM25_pre_1m = paste0("PM25_1km_m_", y_pre_1m, sprintf("%02d", m_pre_1m)),
```

```
    PM25_post_1m = paste0("PM25_1km_m_", y_post_1m, sprintf("%02d", m_post_1m)),
    PM25_post_2m = paste0("PM25_1km_m_", y_post_2m, sprintf("%02d", m_post_2m)),
    PM25_post_3m = paste0("PM25_1km_m_", y_post_3m, sprintf("%02d", m_post_3m)),
    PM25_post_4m = paste0("PM25_1km_m_", y_post_4m, sprintf("%02d", m_post_4m)),
    PM25_post_5m = paste0("PM25_1km_m_", y_post_5m, sprintf("%02d", m_post_5m)),
    PM25_post_6m = paste0("PM25_1km_m_", y_post_6m, sprintf("%02d", m_post_6m)),
    PM25_post_7m = paste0("PM25_1km_m_", y_post_7m, sprintf("%02d", m_post_7m)),
    PM25_post_8m = paste0("PM25_1km_m_", y_post_8m, sprintf("%02d", m_post_8m)),
    PM25_post_9m = paste0("PM25_1km_m_", y_post_9m, sprintf("%02d", m_post_9m)),
    PM25_post_10m = paste0("PM25_1km_m_", y_post_10m, sprintf("%02d", m_post_10m)),
    PM25_post_11m = paste0("PM25_1km_m_", y_post_11m, sprintf("%02d", m_post_11m)),
    PM25_post_12m = paste0("PM25_1km_m_", y_post_12m, sprintf("%02d", m_post_12m))
  )

# 存储数据
rio::export(dt_PM25, "PM25_200501_201012_1km_monthly_Pre_9mth_Post_12mth.rds")
setwd(Results_dir)
rio::export(dt_PM25, "PM25_200501_201012_1km_monthly_Pre_9mth_Post_12mth.rds")

### PM10_200501_201012_1km_monthly ----
# 数据导入
setwd(Data_dir)
dt <- rio::import("Id_LMP_Birth_date_Gest_day_Gest_wk_Gest_mth_18089_20241107.rds")

# 空气污染数据导入
PM10 <- rio::import("PM10_200501_201012_1km_monthly.rds")
dt_PM10 <- left_join(dt, PM10, by = "ID")

# 转换日期格式并生成年份和月份变量
```

```r
dt_PM10 <- dt_PM10 %>%
  mutate(
    LMP_date = ymd(LMP),
    lmp_year = year(LMP_date),
    lmp_month = month(LMP_date),

    # 孕前 9 个月的年份和月份变量
    y_pre_9m = lmp_year, y_pre_8m = lmp_year, y_pre_7m = lmp_year,
    y_pre_6m = lmp_year, y_pre_5m = lmp_year, y_pre_4m = lmp_year,
    y_pre_3m = lmp_year, y_pre_2m = lmp_year, y_pre_1m = lmp_year,

    m_pre_9m = lmp_month - 9, m_pre_8m = lmp_month - 8, m_pre_7m = lmp_month - 7,
    m_pre_6m = lmp_month - 6, m_pre_5m = lmp_month - 5, m_pre_4m = lmp_month - 4,
    m_pre_3m = lmp_month - 3, m_pre_2m = lmp_month - 2, m_pre_1m = lmp_month - 1,

    # 孕后 12 个月的年份和月份变量
    y_post_1m = lmp_year, y_post_2m = lmp_year, y_post_3m = lmp_year,
    y_post_4m = lmp_year, y_post_5m = lmp_year, y_post_6m = lmp_year,
    y_post_7m = lmp_year, y_post_8m = lmp_year, y_post_9m = lmp_year,
    y_post_10m = lmp_year, y_post_11m = lmp_year, y_post_12m = lmp_year,

    m_post_1m = lmp_month, m_post_2m = lmp_month + 1, m_post_3m = lmp_month + 2,
    m_post_4m = lmp_month + 3, m_post_5m = lmp_month + 4, m_post_6m = lmp_month + 5,
    m_post_7m = lmp_month + 6, m_post_8m = lmp_month + 7, m_post_9m = lmp_month + 8,
    m_post_10m = lmp_month + 9, m_post_11m = lmp_month + 10, m_post_12m = lmp_month
+ 11
  )

# 处理跨年数据
dt_PM10 <- dt_PM10 %>%
```

```
  mutate(
    across(
      .cols = starts_with("y_"),
      .fns = ~ ifelse(get(sub("y_", "m_", cur_column())) <= 0, . - 1,
                      ifelse(get(sub("y_", "m_", cur_column())) > 12, . + 1, .))
    ),
    across(
      .cols = starts_with("m_"),
      .fns = ~ ifelse(. <= 0, . + 12, ifelse(. > 12, . - 12, .))
    )
  )

# 生成动态变量名
dt_PM10 <- dt_PM10 %>%
  mutate(
    PM10_pre_9m = paste0("PM10_1km_m_", y_pre_9m, sprintf("%02d", m_pre_9m)),
    PM10_pre_8m = paste0("PM10_1km_m_", y_pre_8m, sprintf("%02d", m_pre_8m)),
    PM10_pre_7m = paste0("PM10_1km_m_", y_pre_7m, sprintf("%02d", m_pre_7m)),
    PM10_pre_6m = paste0("PM10_1km_m_", y_pre_6m, sprintf("%02d", m_pre_6m)),
    PM10_pre_5m = paste0("PM10_1km_m_", y_pre_5m, sprintf("%02d", m_pre_5m)),
    PM10_pre_4m = paste0("PM10_1km_m_", y_pre_4m, sprintf("%02d", m_pre_4m)),
    PM10_pre_3m = paste0("PM10_1km_m_", y_pre_3m, sprintf("%02d", m_pre_3m)),
    PM10_pre_2m = paste0("PM10_1km_m_", y_pre_2m, sprintf("%02d", m_pre_2m)),
    PM10_pre_1m = paste0("PM10_1km_m_", y_pre_1m, sprintf("%02d", m_pre_1m)),

    PM10_post_1m = paste0("PM10_1km_m_", y_post_1m, sprintf("%02d", m_post_1m)),
    PM10_post_2m = paste0("PM10_1km_m_", y_post_2m, sprintf("%02d", m_post_2m)),
    PM10_post_3m = paste0("PM10_1km_m_", y_post_3m, sprintf("%02d", m_post_3m)),
    PM10_post_4m = paste0("PM10_1km_m_", y_post_4m, sprintf("%02d", m_post_4m)),
    PM10_post_5m = paste0("PM10_1km_m_", y_post_5m, sprintf("%02d", m_post_5m)),
```

```
    PM10_post_6m = paste0("PM10_1km_m_", y_post_6m, sprintf("%02d", m_post_6m)),
    PM10_post_7m = paste0("PM10_1km_m_", y_post_7m, sprintf("%02d", m_post_7m)),
    PM10_post_8m = paste0("PM10_1km_m_", y_post_8m, sprintf("%02d", m_post_8m)),
    PM10_post_9m = paste0("PM10_1km_m_", y_post_9m, sprintf("%02d", m_post_9m)),
    PM10_post_10m = paste0("PM10_1km_m_", y_post_10m, sprintf("%02d", m_post_10m)),
    PM10_post_11m = paste0("PM10_1km_m_", y_post_11m, sprintf("%02d", m_post_11m)),
    PM10_post_12m = paste0("PM10_1km_m_", y_post_12m, sprintf("%02d", m_post_12m))
  )

# 存储数据
rio::export(dt_PM10, "PM10_200501_201012_1km_monthly_Pre_9mth_Post_12mth.rds")
setwd(Results_dir)
rio::export(dt_PM10, "PM10_200501_201012_1km_monthly_Pre_9mth_Post_12mth.rds")

# 02 动态提取计算得到时间窗内环境变量浓度+计算各月浓度+各时间窗浓度 ----
## 加载包 ----
{
  library(rio)
  library(tidyverse)
}

## 设置路径 ----
setwd(Data_dir)

## 导入数据 ----
dt_PM25 <- rio::import("PM25_200501_201012_1km_monthly__Pre_9mth_post12mth.rds")
dt_PM10 <- rio::import("PM10_200501_201012_1km_monthly__Pre_9mth_post12mth.rds")

## 动态检索 ----
### 按行操作数据，动态计算 PM25 暴露值，过滤超出 Birth_date 的月份 ----
```

```
  # 运行时间会有点久
dt_PM25 <- dt_PM25 %>%
  rowwise() %>% # 按行进行操作，确保每一行独立处理
  mutate(
    # 使用 `if_else` 判断每个月份的暴露值是否超过分娩日期，超出则赋值为 0

    # 孕前 9 到 1 个月的暴露值
    PM25_pre_9m_value = if_else(ymd(paste0(y_pre_9m, sprintf("-%02d", m_pre_9m), "-01"))
<= ymd(Birth_date), get(PM25_pre_9m), 0),
    PM25_pre_8m_value = if_else(ymd(paste0(y_pre_8m, sprintf("-%02d", m_pre_8m), "-01"))
<= ymd(Birth_date), get(PM25_pre_8m), 0),
    PM25_pre_7m_value = if_else(ymd(paste0(y_pre_7m, sprintf("-%02d", m_pre_7m), "-01"))
<= ymd(Birth_date), get(PM25_pre_7m), 0),
    PM25_pre_6m_value = if_else(ymd(paste0(y_pre_6m, sprintf("-%02d", m_pre_6m), "-01"))
<= ymd(Birth_date), get(PM25_pre_6m), 0),
    PM25_pre_5m_value = if_else(ymd(paste0(y_pre_5m, sprintf("-%02d", m_pre_5m), "-01"))
<= ymd(Birth_date), get(PM25_pre_5m), 0),
    PM25_pre_4m_value = if_else(ymd(paste0(y_pre_4m, sprintf("-%02d", m_pre_4m), "-01"))
<= ymd(Birth_date), get(PM25_pre_4m), 0),
    PM25_pre_3m_value = if_else(ymd(paste0(y_pre_3m, sprintf("-%02d", m_pre_3m), "-01"))
<= ymd(Birth_date), get(PM25_pre_3m), 0),
    PM25_pre_2m_value = if_else(ymd(paste0(y_pre_2m, sprintf("-%02d", m_pre_2m), "-01"))
<= ymd(Birth_date), get(PM25_pre_2m), 0),
    PM25_pre_1m_value = if_else(ymd(paste0(y_pre_1m, sprintf("-%02d", m_pre_1m), "-01"))
<= ymd(Birth_date), get(PM25_pre_1m), 0),

    # 孕后 1 到 12 个月的暴露值
    PM25_post_1m_value = if_else(ymd(paste0(y_post_1m, sprintf("-%02d", m_post_1m), "-
01")) <= ymd(Birth_date), get(PM25_post_1m), 0),
    PM25_post_2m_value = if_else(ymd(paste0(y_post_2m, sprintf("-%02d", m_post_2m), "-
```

```
01")) <= ymd(Birth_date), get(PM25_post_2m), 0),
    PM25_post_3m_value = if_else(ymd(paste0(y_post_3m, sprintf("-%02d", m_post_3m), "-
01")) <= ymd(Birth_date), get(PM25_post_3m), 0),
    PM25_post_4m_value = if_else(ymd(paste0(y_post_4m, sprintf("-%02d", m_post_4m), "-
01")) <= ymd(Birth_date), get(PM25_post_4m), 0),
    PM25_post_5m_value = if_else(ymd(paste0(y_post_5m, sprintf("-%02d", m_post_5m), "-
01")) <= ymd(Birth_date), get(PM25_post_5m), 0),
    PM25_post_6m_value = if_else(ymd(paste0(y_post_6m, sprintf("-%02d", m_post_6m), "-
01")) <= ymd(Birth_date), get(PM25_post_6m), 0),
    PM25_post_7m_value = if_else(ymd(paste0(y_post_7m, sprintf("-%02d", m_post_7m), "-
01")) <= ymd(Birth_date), get(PM25_post_7m), 0),
    PM25_post_8m_value = if_else(ymd(paste0(y_post_8m, sprintf("-%02d", m_post_8m), "-
01")) <= ymd(Birth_date), get(PM25_post_8m), 0),
    PM25_post_9m_value = if_else(ymd(paste0(y_post_9m, sprintf("-%02d", m_post_9m), "-
01")) <= ymd(Birth_date), get(PM25_post_9m), 0),
    PM25_post_10m_value = if_else(ymd(paste0(y_post_10m, sprintf("-%02d", m_post_10m),
"-01")) <= ymd(Birth_date), get(PM25_post_10m), 0),
    PM25_post_11m_value = if_else(ymd(paste0(y_post_11m, sprintf("-%02d", m_post_11m),
"-01")) <= ymd(Birth_date), get(PM25_post_11m), 0),
    PM25_post_12m_value = if_else(ymd(paste0(y_post_12m, sprintf("-%02d", m_post_12m),
"-01")) <= ymd(Birth_date), get(PM25_post_12m), 0)
  ) %>%
  ungroup()  # 取消按行分组，恢复数据框到标准列操作

### 按行操作数据，动态计算 PM10 暴露值，过滤超出 Birth_date 的月份 ----
dt_PM10 <- dt_PM10 %>%
  rowwise() %>%  # 按行进行操作，确保每一行独立处理
  mutate(
    # 使用 `if_else` 判断每个月份的暴露值是否超过分娩日期，超出则赋值为 0
```

```
  # 孕前 9 到 1 个月的暴露值
  PM10_pre_9m_value = if_else(ymd(paste0(y_pre_9m, sprintf("-%02d", m_pre_9m), "-01"))
<= ymd(Birth_date), get(PM10_pre_9m), 0),
  PM10_pre_8m_value = if_else(ymd(paste0(y_pre_8m, sprintf("-%02d", m_pre_8m), "-01"))
<= ymd(Birth_date), get(PM10_pre_8m), 0),
  PM10_pre_7m_value = if_else(ymd(paste0(y_pre_7m, sprintf("-%02d", m_pre_7m), "-01"))
<= ymd(Birth_date), get(PM10_pre_7m), 0),
  PM10_pre_6m_value = if_else(ymd(paste0(y_pre_6m, sprintf("-%02d", m_pre_6m), "-01"))
<= ymd(Birth_date), get(PM10_pre_6m), 0),
  PM10_pre_5m_value = if_else(ymd(paste0(y_pre_5m, sprintf("-%02d", m_pre_5m), "-01"))
<= ymd(Birth_date), get(PM10_pre_5m), 0),
  PM10_pre_4m_value = if_else(ymd(paste0(y_pre_4m, sprintf("-%02d", m_pre_4m), "-01"))
<= ymd(Birth_date), get(PM10_pre_4m), 0),
  PM10_pre_3m_value = if_else(ymd(paste0(y_pre_3m, sprintf("-%02d", m_pre_3m), "-01"))
<= ymd(Birth_date), get(PM10_pre_3m), 0),
  PM10_pre_2m_value = if_else(ymd(paste0(y_pre_2m, sprintf("-%02d", m_pre_2m), "-01"))
<= ymd(Birth_date), get(PM10_pre_2m), 0),
  PM10_pre_1m_value = if_else(ymd(paste0(y_pre_1m, sprintf("-%02d", m_pre_1m), "-01"))
<= ymd(Birth_date), get(PM10_pre_1m), 0),

  # 孕后 1 到 12 个月的暴露值
  PM10_post_1m_value = if_else(ymd(paste0(y_post_1m, sprintf("-%02d", m_post_1m), "-
01")) <= ymd(Birth_date), get(PM10_post_1m), 0),
  PM10_post_2m_value = if_else(ymd(paste0(y_post_2m, sprintf("-%02d", m_post_2m), "-
01")) <= ymd(Birth_date), get(PM10_post_2m), 0),
  PM10_post_3m_value = if_else(ymd(paste0(y_post_3m, sprintf("-%02d", m_post_3m), "-
01")) <= ymd(Birth_date), get(PM10_post_3m), 0),
  PM10_post_4m_value = if_else(ymd(paste0(y_post_4m, sprintf("-%02d", m_post_4m), "-
01")) <= ymd(Birth_date), get(PM10_post_4m), 0),
  PM10_post_5m_value = if_else(ymd(paste0(y_post_5m, sprintf("-%02d", m_post_5m), "-
```

```
01")) <= ymd(Birth_date), get(PM10_post_5m), 0),
    PM10_post_6m_value = if_else(ymd(paste0(y_post_6m, sprintf("-%02d", m_post_6m), "-
01")) <= ymd(Birth_date), get(PM10_post_6m), 0),
    PM10_post_7m_value = if_else(ymd(paste0(y_post_7m, sprintf("-%02d", m_post_7m), "-
01")) <= ymd(Birth_date), get(PM10_post_7m), 0),
    PM10_post_8m_value = if_else(ymd(paste0(y_post_8m, sprintf("-%02d", m_post_8m), "-
01")) <= ymd(Birth_date), get(PM10_post_8m), 0),
    PM10_post_9m_value = if_else(ymd(paste0(y_post_9m, sprintf("-%02d", m_post_9m), "-
01")) <= ymd(Birth_date), get(PM10_post_9m), 0),
    PM10_post_10m_value = if_else(ymd(paste0(y_post_10m, sprintf("-%02d", m_post_10m),
"-01")) <= ymd(Birth_date), get(PM10_post_10m), 0),
    PM10_post_11m_value = if_else(ymd(paste0(y_post_11m, sprintf("-%02d", m_post_11m),
"-01")) <= ymd(Birth_date), get(PM10_post_11m), 0),
    PM10_post_12m_value = if_else(ymd(paste0(y_post_12m, sprintf("-%02d", m_post_12m),
"-01")) <= ymd(Birth_date), get(PM10_post_12m), 0)
  ) %>%
  ungroup()  # 取消按行分组，恢复数据框到标准列操作

## 计算窗口期浓度平均值以得到最终变量 ----
### PM25 ----
#### 计算孕前和孕后的 3 个 Trimester 的平均浓度 ----
dt_PM25 <- dt_PM25 %>%
  rowwise() %>%
  mutate(
    # 孕前 Trimester 平均浓度，仅包含非零数值
    PM25_pre_3rd_trimester = mean(c(PM25_pre_9m_value, PM25_pre_8m_value,
PM25_pre_7m_value)[c(PM25_pre_9m_value, PM25_pre_8m_value, PM25_pre_7m_value) !=
0], na.rm = TRUE),
    PM25_pre_2nd_trimester = mean(c(PM25_pre_6m_value, PM25_pre_5m_value,
PM25_pre_4m_value)[c(PM25_pre_6m_value, PM25_pre_5m_value, PM25_pre_4m_value) !=
```

```
0], na.rm = TRUE),
    PM25_pre_1st_trimester = mean(c(PM25_pre_3m_value, PM25_pre_2m_value,
PM25_pre_1m_value)[c(PM25_pre_3m_value, PM25_pre_2m_value, PM25_pre_1m_value) !=
0], na.rm = TRUE),

    # 孕后 Trimester 平均浓度，仅包含非零数值
    PM25_post_1st_trimester = mean(c(PM25_post_1m_value, PM25_post_2m_value,
PM25_post_3m_value)[c(PM25_post_1m_value, PM25_post_2m_value,
PM25_post_3m_value) != 0], na.rm = TRUE),
    PM25_post_2nd_trimester = mean(c(PM25_post_4m_value, PM25_post_5m_value,
PM25_post_6m_value)[c(PM25_post_4m_value, PM25_post_5m_value,
PM25_post_6m_value) != 0], na.rm = TRUE),
    PM25_post_3rd_trimester = mean(c(PM25_post_7m_value, PM25_post_8m_value,
PM25_post_9m_value)[c(PM25_post_7m_value, PM25_post_8m_value,
PM25_post_9m_value) != 0], na.rm = TRUE)
  ) %>%
  ungroup()

#### 计算 whole pregnancy 的月均浓度 ----
dt_PM25 <- dt_PM25 %>%
  rowwise() %>%  # 按行操作数据，以确保每一行的 `pregnancy_months` 列生成的值与其
所在行独立计算
  mutate(
    # 计算孕期的起始日期
    start_date = ymd(LMP),  # 将末次月经日期 `LMP_date` 作为孕期的开始日期
    end_date = ymd(Birth_date),  # 将出生日期 `Birth_date` 转换为日期格式并作为孕期的
结束日期

    # 创建一个日期序列，涵盖整个孕期的月份
    pregnancy_months = list(seq.Date(
```

```
    from = floor_date(start_date, "month"), # 将 `start_date` 向下取整到月份起点
    to = floor_date(end_date, "month"), # 将 `end_date` 向下取整到月份起点
    by = "month" # 生成从 `start_date` 到 `end_date` 之间每月的日期序列
  )),

  # 动态提取每个月份的浓度值，并计算平均浓度
  PM25_whole_pregnancy = mean( # 计算孕期各月份 PM2.5 浓度的平均值
    sapply(pregnancy_months, function(d) { # 使用 `sapply` 遍历每个月份的日期 `d`
      var_name <- paste0("PM25_1km_m_", year(d), sprintf("%02d", month(d)))
      # 动态构造列名 `var_name`，例如 `PM25_1km_m_2007_05`，以匹配每个月份的数据列

      if (var_name %in% names(dt_PM25)) get(var_name) else NA
      # 检查 `var_name` 是否存在于数据框的列名中，存在则提取该列值，否则返回 `NA`
    }), na.rm = TRUE # 忽略 NA 值计算平均浓度
  )
 ) %>%
 ungroup() # 取消 `rowwise` 分组，以便后续操作按列进行

dt_PM25 <- dt_PM25 %>%
 select(
  ID,
  PM25_pre_9m_value,
  PM25_pre_8m_value,
  PM25_pre_7m_value,
  PM25_pre_6m_value,
  PM25_pre_5m_value,
  PM25_pre_4m_value,
  PM25_pre_3m_value,
```

```
    PM25_pre_2m_value,
    PM25_pre_1m_value,
    PM25_post_1m_value,
    PM25_post_2m_value,
    PM25_post_3m_value,
    PM25_post_4m_value,
    PM25_post_5m_value,
    PM25_post_6m_value,
    PM25_post_7m_value,
    PM25_post_8m_value,
    PM25_post_9m_value,
    PM25_post_10m_value,
    PM25_post_11m_value,
    PM25_post_12m_value,
    PM25_pre_3rd_trimester,
    PM25_pre_2nd_trimester,
    PM25_pre_1st_trimester,
    PM25_post_1st_trimester,
    PM25_post_2nd_trimester,
    PM25_post_3rd_trimester,
    PM25_whole_pregnancy
  )

### PM10 ----
#### 计算孕前和孕后的 3 个 Trimester 的平均浓度 ----
dt_PM10 <- dt_PM10 %>%
  rowwise() %>%
  mutate(
    # 孕前 Trimester 平均浓度，仅包含非零数值
    PM10_pre_3rd_trimester = mean(c(PM10_pre_9m_value, PM10_pre_8m_value,
```

```
PM10_pre_7m_value)[c(PM10_pre_9m_value, PM10_pre_8m_value, PM10_pre_7m_value) !=
0], na.rm = TRUE),
    PM10_pre_2nd_trimester = mean(c(PM10_pre_6m_value, PM10_pre_5m_value,
PM10_pre_4m_value)[c(PM10_pre_6m_value, PM10_pre_5m_value, PM10_pre_4m_value) !=
0], na.rm = TRUE),
    PM10_pre_1st_trimester = mean(c(PM10_pre_3m_value, PM10_pre_2m_value,
PM10_pre_1m_value)[c(PM10_pre_3m_value, PM10_pre_2m_value, PM10_pre_1m_value) !=
0], na.rm = TRUE),

    # 孕后 Trimester 平均浓度，仅包含非零数值
    PM10_post_1st_trimester = mean(c(PM10_post_1m_value, PM10_post_2m_value,
PM10_post_3m_value)[c(PM10_post_1m_value, PM10_post_2m_value,
PM10_post_3m_value) != 0], na.rm = TRUE),
    PM10_post_2nd_trimester = mean(c(PM10_post_4m_value, PM10_post_5m_value,
PM10_post_6m_value)[c(PM10_post_4m_value, PM10_post_5m_value,
PM10_post_6m_value) != 0], na.rm = TRUE),
    PM10_post_3rd_trimester = mean(c(PM10_post_7m_value, PM10_post_8m_value,
PM10_post_9m_value)[c(PM10_post_7m_value, PM10_post_8m_value,
PM10_post_9m_value) != 0], na.rm = TRUE)
  ) %>%
  ungroup()

#### 计算 whole pregnancy 的月均浓度 ----
dt_PM10 <- dt_PM10 %>%
  rowwise() %>%  # 按行操作数据，以确保每一行的 `pregnancy_months` 列生成的值与其
所在行独立计算
  mutate(
    # 计算孕期的起始日期
    start_date = ymd(LMP),  # 将末次月经日期 `LMP_date` 作为孕期的开始日期
    end_date = ymd(Birth_date),  # 将出生日期 `Birth_date` 转换为日期格式并作为孕期的
```

```
结束日期

    # 创建一个日期序列，涵盖整个孕期的月份
    pregnancy_months = list(seq.Date(
      from = floor_date(start_date, "month"), # 将 `start_date` 向下取整到月份起点
      to = floor_date(end_date, "month"), # 将 `end_date` 向下取整到月份起点
      by = "month" # 生成从 `start_date` 到 `end_date` 之间每月的日期序列
    )),

    # 动态提取每个月份的浓度值，并计算平均浓度
    PM10_whole_pregnancy = mean( # 计算孕期各月份 PM2.5 浓度的平均值
      sapply(pregnancy_months, function(d) { # 使用 `sapply` 遍历每个月份的日期 `d`
        var_name <- paste0("PM10_1km_m_", year(d), sprintf("%02d", month(d)))
        # 动态构造列名 `var_name`，例如 `PM10_1km_m_2007_05`，以匹配每个月份的数据列

        if (var_name %in% names(dt_PM10)) get(var_name) else NA
        # 检查 `var_name` 是否存在于数据框的列名中，存在则提取该列值，否则返回 `NA`
      }), na.rm = TRUE # 忽略 NA 值计算平均浓度
    )
  ) %>%
  ungroup() # 取消 `rowwise` 分组，以便后续操作按列进行

dt_PM10 <- dt_PM10 %>%
  select(
    ID,
    PM10_pre_9m_value,
    PM10_pre_8m_value,
    PM10_pre_7m_value,
```

```
    PM10_pre_6m_value,
    PM10_pre_5m_value,
    PM10_pre_4m_value,
    PM10_pre_3m_value,
    PM10_pre_2m_value,
    PM10_pre_1m_value,
    PM10_post_1m_value,
    PM10_post_2m_value,
    PM10_post_3m_value,
    PM10_post_4m_value,
    PM10_post_5m_value,
    PM10_post_6m_value,
    PM10_post_7m_value,
    PM10_post_8m_value,
    PM10_post_9m_value,
    PM10_post_10m_value,
    PM10_post_11m_value,
    PM10_post_12m_value,
    PM10_pre_3rd_trimester,
    PM10_pre_2nd_trimester,
    PM10_pre_1st_trimester,
    PM10_post_1st_trimester,
    PM10_post_2nd_trimester,
    PM10_post_3rd_trimester,
    PM10_whole_pregnancy
  )

## 保存结果 ----
rio::export(dt_PM25,
"PM25_pre_post_trimesters_whole_pregnancy_18089_rev_3rd_trimester.rds")
```

```R
rio::export(dt_PM10,
"PM10_pre_post_trimesters_whole_pregnancy_18089_rev_3rd_trimester.rds")
rio::export(dt_PM25,
"PM25_pre_post_trimesters_whole_pregnancy_18089_rev_3rd_trimester.xlsx")
rio::export(dt_PM10,
"PM10_pre_post_trimesters_whole_pregnancy_18089_rev_3rd_trimester.xlsx")

setwd(Results_dir)
rio::export(dt_PM25,
"PM25_pre_post_trimesters_whole_pregnancy_18089_rev_3rd_trimester.rds")
rio::export(dt_PM10,
"PM10_pre_post_trimesters_whole_pregnancy_18089_rev_3rd_trimester.rds")
rio::export(dt_PM25,
"PM25_pre_post_trimesters_whole_pregnancy_18089_rev_3rd_trimester.xlsx")
rio::export(dt_PM10,
"PM10_pre_post_trimesters_whole_pregnancy_18089_rev_3rd_trimester.xlsx")
```
R