

113 學年度第 2 學期 GIS 特論作業 2

資料簡介

資料為 1960~2020 的日平均雨量資料共 61 個 csv 檔案。每年皆有一個檔案，檔案內有當年度每日的雨量計算資料。而其中-99.9 的資料為無資料。如圖所示：

LON	LAT	20000101	20000102	20000103
121.5875	24.3125	0	0	0
121.5958	24.3125	0	0	0
121.6042	24.3125	0	0	0
121.7542	24.3125	0	0	0
121.7625	24.3125	0	0	0
121.7708	24.3125	-99.9	-99.9	-99.9
121.5625	24.32083	0	0	0
121.5708	24.32083	0	0	0
121.5792	24.32083	0	0	0
121.5875	24.32083	0	0	0
121.5958	24.32083	0	0	0

圖一、2000 年 1 月日雨量內容示意圖

資料前處理

首先先來看檔案的結構，如圖二所示：

1	LON, LAT, 19600101,19600102,19600103,19600104,19600105,19600106,19600107,19600108,19600109,19600110,19600111,19600112
2	121.58750, 24.31250, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
3	121.59580, 24.31250, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
4	121.60420, 24.31250, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
5	121.75420, 24.31250, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
6	121.76250, 24.31250, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
7	121.77080, 24.31250, -99.9000, -99.9000, -99.9000, -99.9000, -99.9000, -99.9000, -99.9000, -99.9000, -99.9000, -99.9000,
8	121.56250, 24.32083, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
9	121.57080, 24.32083, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
10	121.57920, 24.32083, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
11	121.58750, 24.32083, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
12	121.59580, 24.32083, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
13	121.60420, 24.32083, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
14	121.72920, 24.32083, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
15	121.73750, 24.32083, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
16	121.74580, 24.32083, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
17	121.75420, 24.32083, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
18	121.76250, 24.32083, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
19	121.77080, 24.32083, -99.9000, -99.9000, -99.9000, -99.9000, -99.9000, -99.9000, -99.9000, -99.9000, -99.9000, -99.9000,
20	121.55420, 24.32916, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
21	121.56250, 24.32916, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
22	121.57080, 24.32916, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,

圖二、1960 年 1 月部分資料示意圖

可以發現第一欄位的 LON 和第二欄位的 LAT，以及許多資料的前面都有空白。為了以免空白所導致的解算錯誤，因此要先利用程式碼將其整理好將空格移除。再來計算每個點位之餘每個年月的平均雨量。

首先是先移除資料的空白之處，程式碼關鍵部分如下：

```
# 處理每一行，移除所有空格
processed_lines = []
for line in lines:
    # 分割每一行的欄位
    fields = line.split(',')

    # 移除每個欄位中的空格
    cleaned_fields = [field.replace(' ', '') for field in fields]

    # 重新組合行
    processed_lines.append(','.join(cleaned_fields))

# 寫回檔案
with open(file, 'w', encoding='utf-8', newline='') as f:
    f.writelines(processed_lines)

print(f"已完成處理: {os.path.basename(file)}")
```

由此便可針對每一年份的資料進行清理。

再來是計算每一年份內的月平均雨量，關鍵程式碼如下：

```
def convert_daily_to_monthly(input_dir, output_dir):
    # 獲取所有年份的 CSV 檔案
    csv_files = [f for f in os.listdir(input_dir) if f.endswith('.csv')]

    for csv_file in csv_files:
        print(f"處理檔案: {csv_file}")

        # 讀取 CSV 檔案
        file_path = os.path.join(input_dir, csv_file)
        df = pd.read_csv(file_path, index_col=False)

        # 獲取年份
        year_match = re.search(r'_(\d{4})\.csv$', csv_file)
        if not year_match:
            print(f"無法從檔名獲取年份: {csv_file}")
            continue

        year = year_match.group(1)
```

```

# 創建新的 DataFrame 來存儲月資料
monthly_df = pd.DataFrame()
monthly_df['LON'] = df['LON']
monthly_df['LAT'] = df['LAT']

# 處理每個月
for month in range(1, 13):
    month_str = f"{year}{month:02d}"

    # 獲取該月的所有日期列
    days_in_month = calendar.monthrange(int(year), month)[1]
    date_cols = [f"{year}{month:02d}{day:02d}" for day in range(1,
days_in_month + 1)]

    # 過濾出存在於 DataFrame 中的日期列
    existing_cols = [col for col in date_cols if col in df.columns]

    if existing_cols:
        # 創建一個臨時 DataFrame，將-99.9 替換為 NaN
        temp_df = df[existing_cols].replace(-99.9, np.nan)

        # 檢查每一行是否所有值都是 NaN（原始值都是-99.9）
        all_missing = temp_df.isna().all(axis=1)

        # 計算月累積降雨量（忽略 NaN 值）
        monthly_df[month_str] = temp_df.sum(axis=1, skipna=True)

        # 如果某行全部都是-99.9（無資料），將結果設為-99.9
        monthly_df.loc[all_missing, month_str] = -99.9
    else:
        monthly_df[month_str] = -99.9

# 保存月資料為 CSV
output_file = os.path.join(output_dir, f"月降雨量_{year}.csv")
monthly_df.to_csv(output_file, index=False)
print(f"已保存: {output_file}")

```

可以看到除了將數值計算完畢以外，也將無資料的欄位設定成-99.9。

```
def merge_monthly_data(monthly_dir, output_dir):
```

```

# 獲取所有月資料 CSV 檔案
csv_files = [f for f in os.listdir(monthly_dir) if f.startswith('月降雨量_') and
f.endswith('.csv')]
csv_files.sort() # 確保按年份排序

if not csv_files:
    print("找不到月降雨量資料檔案")
    return

# 讀取第一個檔案以獲取坐標資訊
first_file = os.path.join(monthly_dir, csv_files[0])
result_df = pd.read_csv(first_file)[['LON', 'LAT']]

# 處理每個年份的檔案
for csv_file in csv_files:
    print(f"合併: {csv_file}")
    file_path = os.path.join(monthly_dir, csv_file)
    df = pd.read_csv(file_path)

    # 獲取月份列 (排除 LON 和 LAT)
    month_cols = [col for col in df.columns if col not in ['LON', 'LAT']]

    # 將月份資料加入結果 DataFrame
    for col in month_cols:
        result_df[col] = df[col]

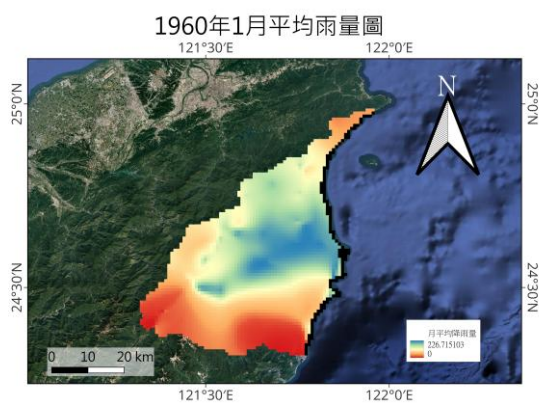
# 保存最終結果
result_file = os.path.join(output_dir, "result.csv")
result_df.to_csv(result_file, index=False)
print(f"已完成合併，結果保存為: {result_file}")

```

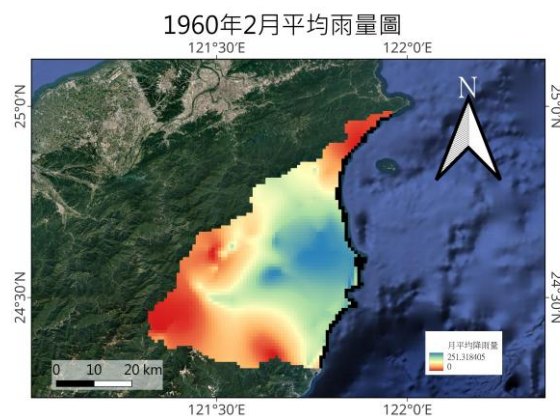
最後便可以將處理完成的 csv 檔案全部合併，獲得最終的 result.csv。

1960~2020 各月分宜蘭縣雨量圖及平均月雨量圖

以 1961 年 1、2 月份的雨量成果圖為展示(總共共有 732 張圖)。

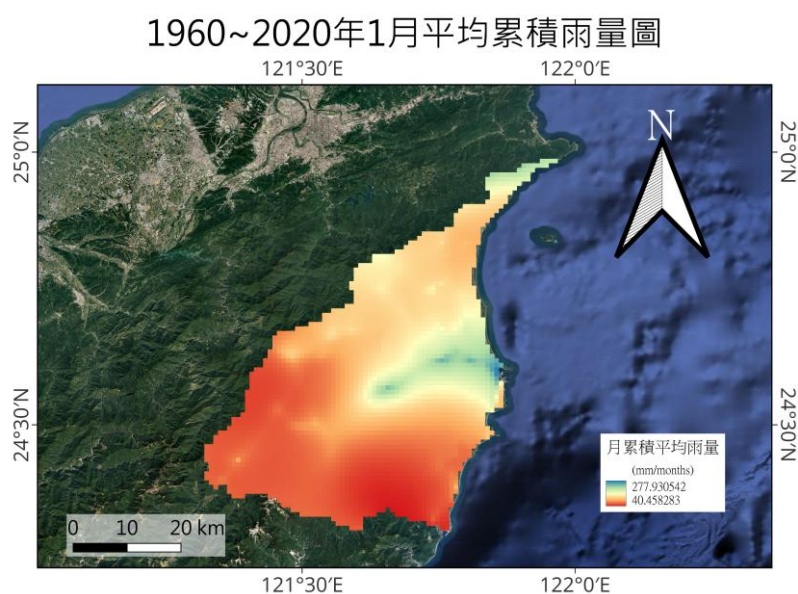


圖三 1960 年 1 月累積雨量圖



圖四 1960 年 2 月累積雨量圖

1960~2020 年平均月累積雨量成果圖(僅展示 1 月，總共共有 12 張圖):



圖五 1960~2020 年 1 月平均累積月雨量圖

程式碼簡介

首先鑒於解算的效率，兩組程式碼我皆是利用 OSGeo4W shell 作為執行環境，主要使用裡面所提供的 GIS 套件進行解算。

圖 X [OSGeo4W](#)

以下為各年份每月累計雨量計算出圖程式碼之重點部分：

```
def csv_to_raster(csv_path, output_folder, x_field='LON', y_field='LAT',
delimeter=',', nodata_value=-99.9):
    """將 CSV 檔案轉換為柵格檔案，將特定值設為 NoData"""
    try:
        # 確保輸出資料夾存在
        if not os.path.exists(output_folder):
            os.makedirs(output_folder)
            print(f"已創建輸出資料夾：{output_folder}")

        # 檢查 CSV 檔案是否存在
        if not os.path.exists(csv_path):
            print(f"錯誤：找不到 CSV 檔案 '{csv_path}'")
            return False

        print(f"處理 CSV 檔案：{csv_path}")

        # 讀取 CSV 檔案的標頭
        with open(csv_path, 'r') as f:
            header = f.readline().strip().split(delimeter)

        print(f"檢測到的欄位：{header}")

        # 確認經緯度欄位是否存在
        if x_field not in header or y_field not in header:
            print(f"錯誤：找不到經緯度欄位 '{x_field}' 或 '{y_field}'")
            return False

        # 獲取所有可能的數值欄位（排除經緯度欄位）
        value_fields = [field for field in header if field not in [x_field,
y_field]]
        print(f"找到 {len(value_fields)} 個可能的數值欄位")

        # 創建臨時的 Shapefile
        driver = ogr.GetDriverByName('MEMORY')
        data_source = driver.CreateDataSource('memory')

        # 創建空間參考
        srs = ogr.osr.SpatialReference()
```

```

srs.ImportFromEPSG(4326) # WGS84

# 創建圖層
layer = data_source.CreateLayer('points', srs, ogr.wkbPoint)

# 添加欄位
for field in value_fields:
    field_defn = ogr.FieldDefn(field, ogr.OFTReal)
    layer.CreateField(field_defn)

# 讀取 CSV 資料
with open(csv_path, 'r') as f:
    next(f) # 跳過標頭
    for line in f:
        parts = line.strip().split(delimiter)
        if len(parts) != len(header):
            continue # 跳過格式不正確的行

        try:
            x = float(parts[header.index(x_field)])
            y = float(parts[header.index(y_field)])

            # 創建點幾何
            point = ogr.Geometry(ogr.wkbPoint)
            point.AddPoint(x, y)

            # 創建要素
            feature = ogr.Feature(layer.GetLayerDefn())
            feature.SetGeometry(point)

            # 設定欄位值，將 nodata_value 視為 NULL
            for field in value_fields:
                try:
                    value = float(parts[header.index(field)])
                    # 如果值等於 nodata_value，則不設置該欄位，讓它保持為 NULL
                    if abs(value - nodata_value) > 0.0001: # 使用近似比較避免
                        # 浮點誤差

                        feature.SetField(field, value)
                except (ValueError, IndexError):
                    pass # 不設置欄位，讓它保持為 NULL

```

```

        # 添加要素到圖層
        layer.CreateFeature(feature)
        feature = None
    except (ValueError, IndexError) as e:
        print(f"跳過無效行: {line.strip()} - {str(e)}")

print(f"成功載入 {layer.GetFeatureCount()} 個點")

# 獲取圖層的範圍
extent = layer.GetExtent()
x_min, x_max, y_min, y_max = extent[0], extent[1], extent[2], extent[3]

# 設定網格分辨率
res_x = 0.0083
res_y = 0.0083

# 為每個數值欄位創建柵格
for field in value_fields:
    print(f"處理欄位: {field}")
    output_file = os.path.join(output_folder, f"{field}.tif")

    # 設定柵格化參數
    width = int((x_max - x_min) / res_x)
    height = int((y_max - y_min) / res_y)

    # 創建目標柵格
    driver = gdal.GetDriverByName('GTiff')
    raster = driver.Create(output_file, width, height, 1, gdal.GDT_Float32)

    # 設定地理參考
    raster.SetGeoTransform((x_min, res_x, 0, y_max, 0, -res_y))
    raster.SetProjection(srs.ExportToWkt())

    # 設定無資料值
    band = raster.GetRasterBand(1)
    band.SetNoDataValue(nodata_value)
    band.Fill(nodata_value)

# 執行柵格化

```



```

gdal.RasterizeLayer(raster, [1], layer, options=[f"ATTRIBUTE={field}"])

# 處理柵格資料，確保 -99.9 被設為 NoData
data = band.ReadAsArray()

# 將資料寫回柵格
band.WriteArray(data)

# 清理
band = None
raster = None

print(f"已生成柵格檔案: {output_file}")

return True

except Exception as e:
    print(f"處理 CSV 檔案時發生錯誤: {str(e)}")
    import traceback
    traceback.print_exc()
    return False

```

關鍵在於記得設定正確的網格分辨率

➤ `res_x = 0.0083`

➤ `res_y = 0.0083`

而設定完後就利用 gdal 所提供的套件將它轉換成網格的 GeoTIFF 值

➤ `driver = gdal.GetDriverByName('GTiff')`

➤ `raster = driver.Create(output_file, width, height, 1, gdal.GDT_Float32)`

而 1960~2020 的每月平均累計雨量圖的產出差異只在於計算上面的差異。