# 土地改良事業計画（排水）における将来の降雨予測に基づく確率降雨量算定マニュアル

令和７年４月

令和７年 10 月改定

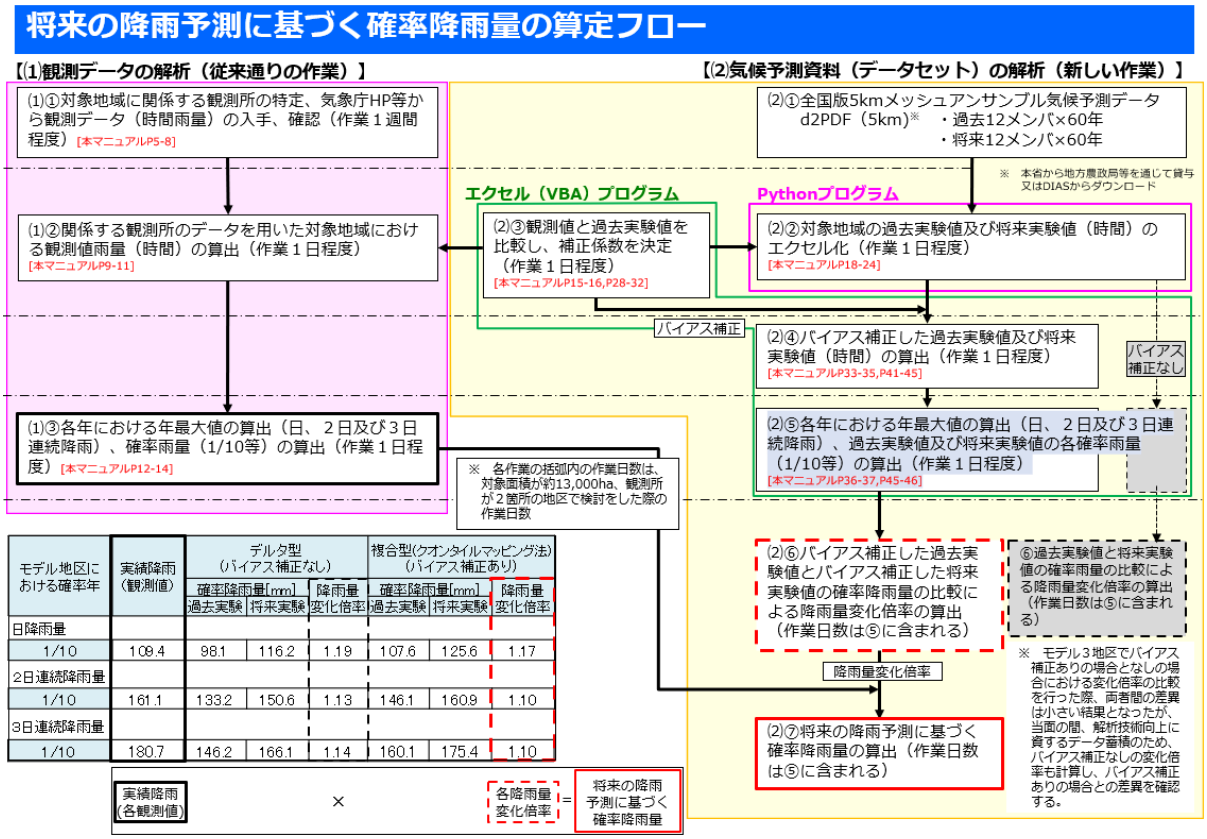
農村振興局整備部設計課計画調整室計画基準班

目次

# 第１章　作業フロー

将来の降雨予測に基づく確率降雨量の算定は、気象庁の観測降雨量データ、気候予測資料（データセット）の d2PDF（５km）の過去実験データ及び将来実験データの３つのデータを使用して行う。なお、d2PDF（５km）は「全国５km メッシュアンサンブル気候予測データ」における RCP8.5 シナリオに基づく２℃上昇時点（2040 年頃）の予測結果であり、正式名称は「d4PDF（２℃上昇実験、５km)」である。

その３つのデータを利用した確率降雨量の算定フローを下記に示す。




| モデル地区における確率年 | 実績降雨（観測値） | デルタ型（バイアス補正なし） | | | 複合型(クオンタイルマッピング法)（バイアス補正あり） | | |
|---|---|---|---|---|---|---|---|
| | | 確率降雨量[mm] | | 降雨量 | 確率降雨量[mm] | | 降雨量 |
| | | 過去実験 | 将来実験 | 変化倍率 | 過去実験 | 将来実験 | 変化倍率 |
| 日降雨量 | | | | | | | |
| 1/10 | 109.4 | 98.1 | 116.2 | 1.19 | 107.6 | 125.6 | 1.17 |
| 2日連続降雨量 | | | | | | | |
| 1/10 | 161.1 | 133.2 | 150.6 | 1.13 | 146.1 | 160.9 | 1.10 |
| 3日連続降雨量 | | | | | | | |
| 1/10 | 180.7 | 146.2 | 166.1 | 1.14 | 160.1 | 175.4 | 1.10 |

# 第２章　作業環境の設定（Python）

本作業で実行する Python プログラムコードは、Linux 上で動作するため、Linux 環境が整っていない場合は、以下第１節の環境構築が必要である。

## 第１節　Windows における環境構築例（Linux 環境未構築の場合）

WSL（Windows Subsystem for Linux）をインストールすることにより Windows に Linux 環境（Linux 系の OS である Ubuntu）を構築することが可能である（前提条件として Windows のバージョンは、Windows 10 の 2004 以上であること）。

また、WSL のインストールにおいて、同時に Python もインストールすることが可能であり一度に作業環境が構築出来るため、下記のインストール方法を推奨する。

（インストールの詳細 https://learn.microsoft.com/ja-jp/windows/wsl/install）

### １　WSL のインストール

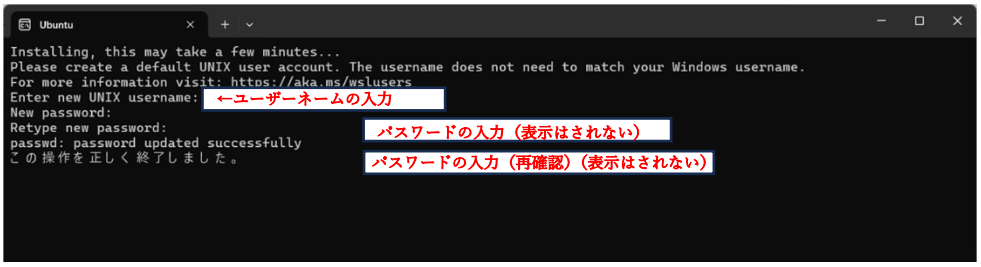
Windows PowerShell 上で、「wsl --install」と入力し、WSL をインストールする。

（Windows PowerShell 画面）


### ２　ユーザー情報の登録

Microsoft Store から Linux で検索して Ubuntu アプリをクリックし、設定画面で、ユーザーネームとパスワードを設定しユーザー情報を登録する。

（Ubuntu 設定画面）



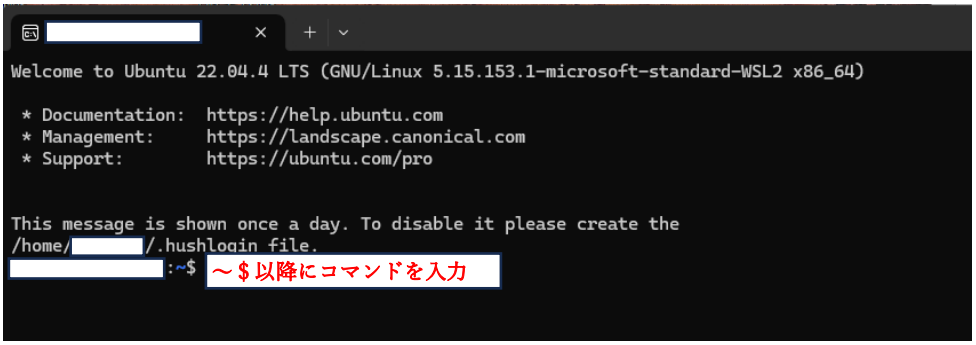
# 第２節　Python のライブラリのインストール（Linux 環境構築後）

## 1　PIP（ライブラリをインストールするための仕組み）のインストール

Python のプログラムコードを実行する際に必要なライブラリ（Python 上で簡単に数値計算等の機能を利用出来るようにするためのプログラムコードの集合体）をインストールするために、PIP をインストールする。

Ubuntu のターミナル画面に下記のコマンドを入力し、PIP のインストールを行う。

（Ubuntu のターミナル画面の例）




（コマンドの入力：PIP のインストール状況の確認）

```
$ pip3 -V
```

（未インストールの場合の表示例）

```
$ pip3 -V
pip3: command not found
```

（インストール済みの場合の表示例）

```
$ pip3 -V
pip 22.0.2 from /usr/lib/python3/dist-pa
ckages/pip (python 3.10)
```

（コマンドの入力：apt のアップデート）

```
$ sudo apt update
```

（コマンドの入力：apt のアップグレード）

```
$ sudo apt upgrade
```

（コマンドの入力：PIP のインストール）

```
$ sudo apt install python3-pip
```

※PIP のインストール状況は、再度「$ pip3 -V」を入力し確認する。

## 2　ライブラリのインストール

必要なライブラリ（numpy,Pillow, netCDF4, pyshp）をインストールする。

（コマンドの入力：ライブラリのインストール）

```
$ pip3 install numpy Pillow netCDF4 pyshp
```

※ライブラリのインストール結果は、下記のコマンドを入力し確認する。

（コマンドの入力：ライブラリのインストール結果の確認）

```
$ pip list
```

（Ubuntu のターミナル画面の表示例）

```
[redacted]:~$ pip list
Package              Version
-------------------- -------------
blinker              1.4
certifi              2024.6.2
cftime               1.6.4
command-not-found    0.3
cryptography         3.4.8
dbus-python          1.2.18
distro               1.7.0
distro-info          1.1+ubuntu0.2
httplib2             0.20.2
importlib-metadata   4.6.4
jeepney              0.7.1
keyring              23.5.0
launchpadlib         1.10.16
lazr.restfulclient   0.14.4
lazr.uri             1.0.6
more-itertools       8.10.0
netCDF4              1.7.1
netifaces            0.11.0
numpy                2.0.0
oauthlib             3.2.0
pillow               10.3.0
pip                  22.0.2
PyGObject            3.42.1
PyJWT                2.3.0
pyparsing            2.4.7
pyshp                2.3.1
python-apt           2.4.0+ubuntu3
PyYAML               5.4.1
SecretStorage        3.3.1
setuptools           59.6.0
six                  1.16.0
systemd-python       234
ubuntu-pro-client    8001
ufw                  0.36.1
unattended-upgrades  0.1
wadllib              1.3.6
wheel                0.37.1
zipp                 1.0.0
```

４つのライブラリが
表示されていること

# 第３章　観測値の処理

## 第１節　データの入手

気象庁 HP（気象庁|過去の気象データ・ダウンロード（jma.go.jp））等から観測値（時間降雨量）に関するデータをダウンロードする。

また、ダウンロードする観測値（将来降雨量を算出する際に用いる観測値）は、実験データが９月１日～翌年８月 31 日を１年間として計算されていることから、９月１日～翌年８月 31 年を１年間として扱う。

■ダウンロードするデータ

1951 年９月１日～2011 年８月 31 日（過去実験値の計算期間）の内、2011 年８月 31 日から遡った 30 年～50 年程度のデータ

（例：気象庁 HP からのダウンロード）

（地点の選択）

（項目の選択）データの種類：時別値　　項目：降水量（前１時間）


（期間の選択）１回毎のダウンロード制限があるため、１地点１年毎にダウンロード

（例：気象庁 HP からダウンロードした 1 観測所の降雨量データ）

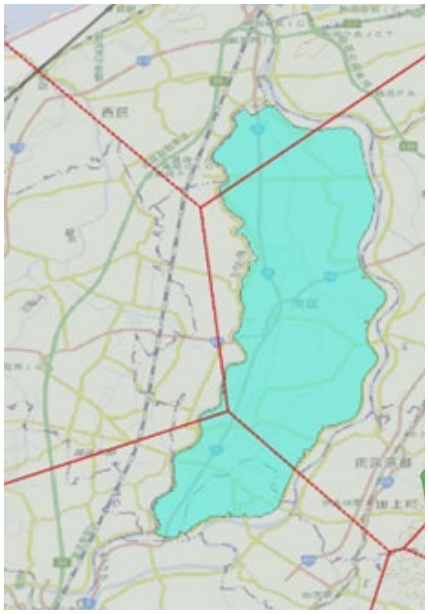
| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | ダウンロードした時刻：2024/09/17 09:11:27 | | | | | | |
| 2 | | | | | | | |
| 3 | | 新潟 | 新潟 | 新潟 | 新潟 | | |
| 4 | 年月日時 | 降水量(mm) | 降水量(mm) | 降水量(mm) | 降水量(mm) | | |
| 5 | | | 現象なし情報 | 品質情報 | 均質番号 | | |
| 6 | 1981/9/1 1:00 | 0 | 1 | 8 | 1 | | |
| 7 | 1981/9/1 2:00 | 0 | 1 | 8 | 1 | | |
| 8 | 1981/9/1 3:00 | 0 | 1 | 8 | 1 | | |
| 9 | 1981/9/1 4:00 | 2.5 | 0 | 8 | 1 | | |
| 10 | 1981/9/1 5:00 | 0.5 | 0 | 8 | 1 | | |
| 11 | 1981/9/1 6:00 | 2 | 0 | 8 | 1 | | |
| 12 | 1981/9/1 7:00 | 0.5 | 0 | 8 | 1 | | |
| 13 | 1981/9/1 8:00 | 0.5 | 0 | 8 | 1 | | |
| 14 | 1981/9/1 9:00 | 0.5 | 0 | 8 | 1 | | |
| 15 | 1981/9/1 10:00 | 0 | 0 | 8 | 1 | | |
| 16 | 1981/9/1 11:00 | 0 | 1 | 8 | 1 | | |
| 17 | 1981/9/1 12:00 | 0 | 1 | 8 | 1 | | |
| 18 | 1981/9/1 13:00 | 0 | 1 | 8 | 1 | | |
| 19 | 1981/9/1 14:00 | 0 | 1 | 8 | 1 | | |
| 20 | 1981/9/1 15:00 | 0 | 1 | 8 | 1 | | |
| 21 | 1981/9/1 16:00 | 0 | 1 | 8 | 1 | | |
| 22 | 1981/9/1 17:00 | 0 | 1 | 8 | 1 | | |
| 23 | 1981/9/1 18:00 | 0 | 1 | 8 | 1 | | |

降水量（時間）データ

data（1）

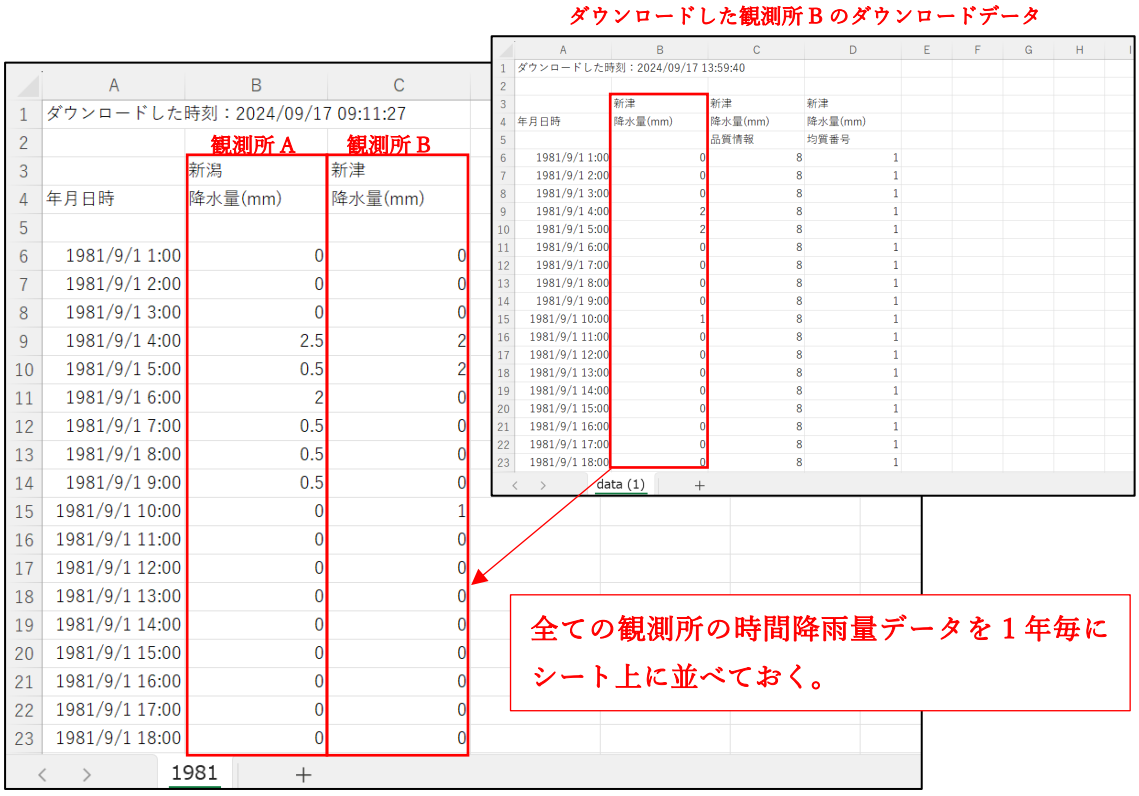
また、各対象地区に必要な観測地点、面積割合の特定は、地図上からティーセン法等を用いて行う。

（例：GIS 上におけるティーセン法の使用）



複数の観測所を使用する場合は、全ての観測所の降雨量を１年毎にエクセルシートに整理する。

（例：１年毎の観測値の整理）

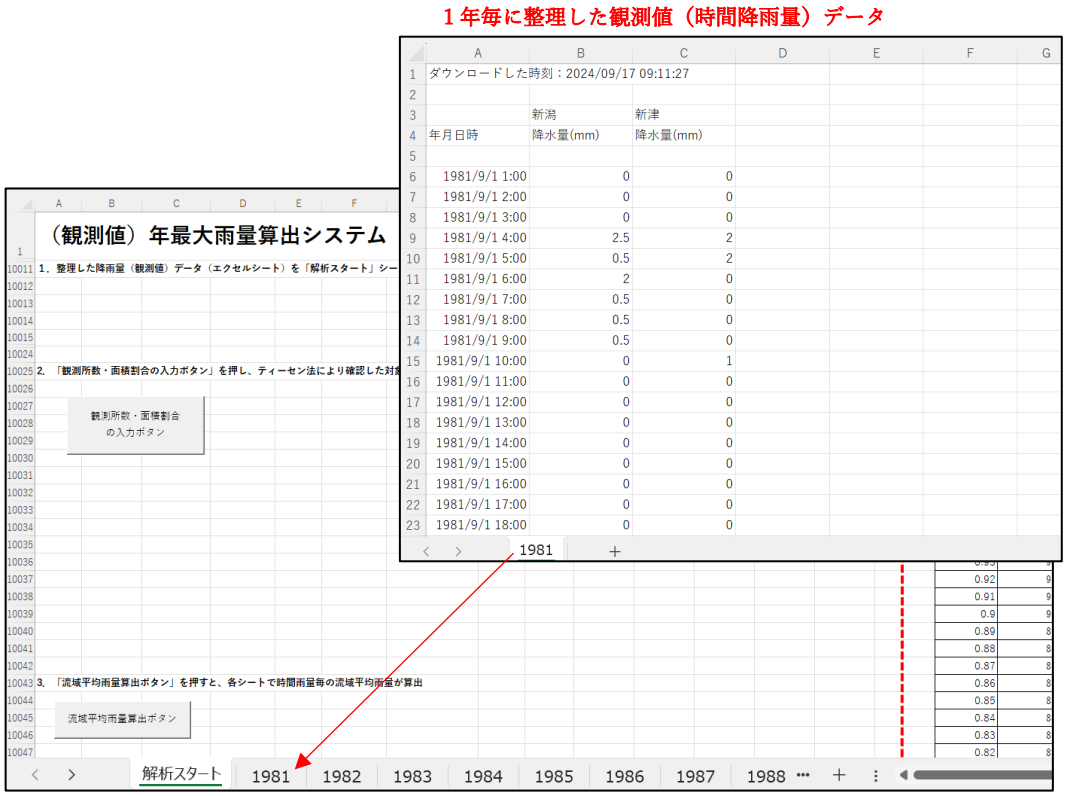


# 第２節　データの処理

## １　確率降雨量の算出

### （１）　システムへのデータの追加

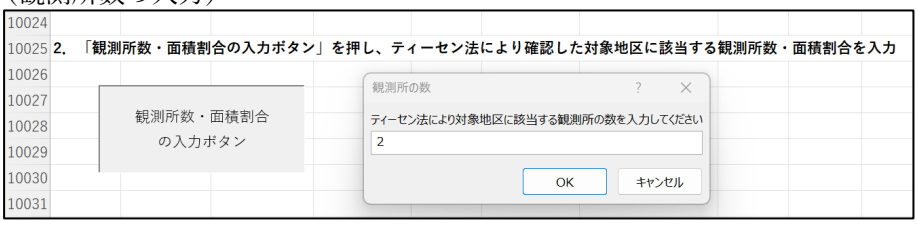
『(観測値）年最大雨量算出システム』を開き、「解析スタート」シートの後ろに整理した観測値のシートを１年毎に追加（例：1981 年～2010 年を選定した場合、「解析スタート」シートの後ろ（右側）に 1981 年～2010 年のデータを追加）



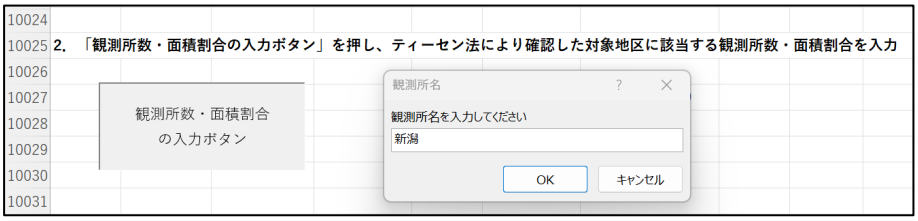
（２）　対象流域の条件入力

「解析スタートシート」内の「観測所数・面積割合の入力ボタン」を押し、観測所数、面積割合等を入力（例は、対象地区に関係する観測所が２箇所の場合）

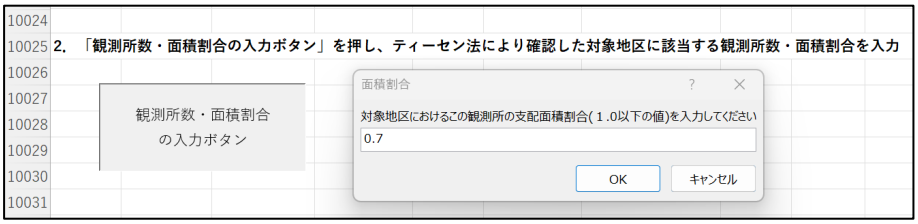

（観測所数の入力）




（観測所名の入力）




（面積割合の入力）

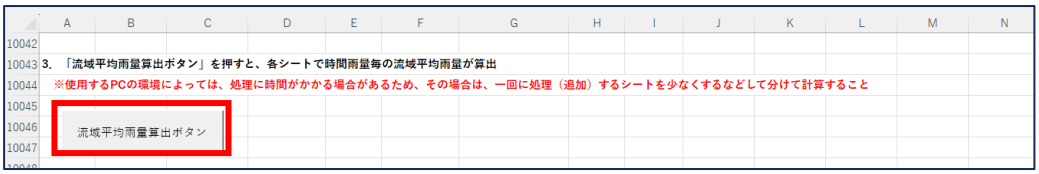



（３）　時間毎の流域平均雨量の算出

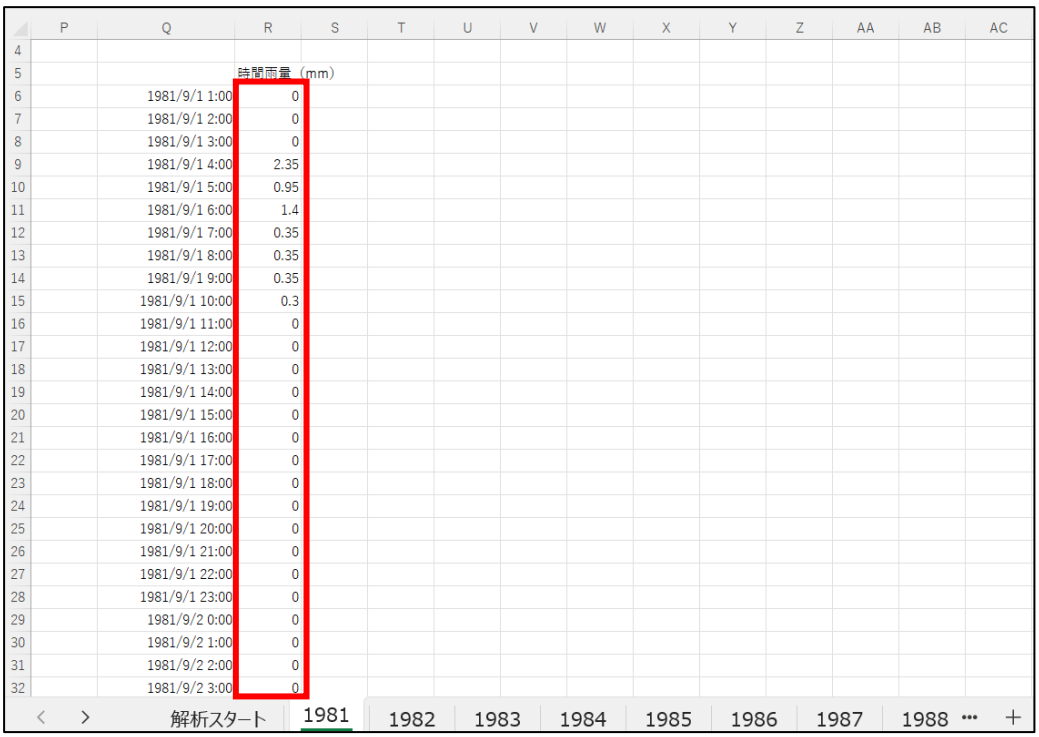
「解析スタートシート」内の「流域平均雨量算出ボタン」を押し、各年の「降雨量シート」の R 列に各地区の流域平均雨量（時間）を算出

※　使用する PC の環境によっては、処理に時間がかかる場合があるため、その場合は、一回に処理（追加）するシートを少なくするなどして分けて作業すること

（解析スタートシート）




（降雨量シート）

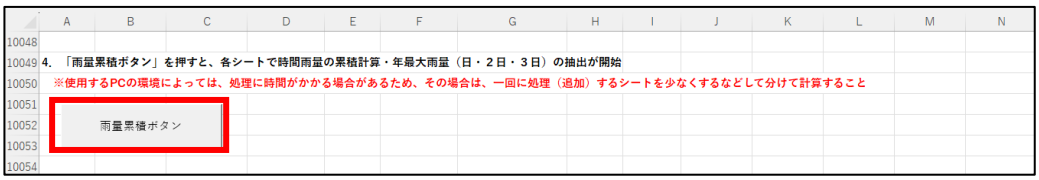


（４）　年最大雨量の算出

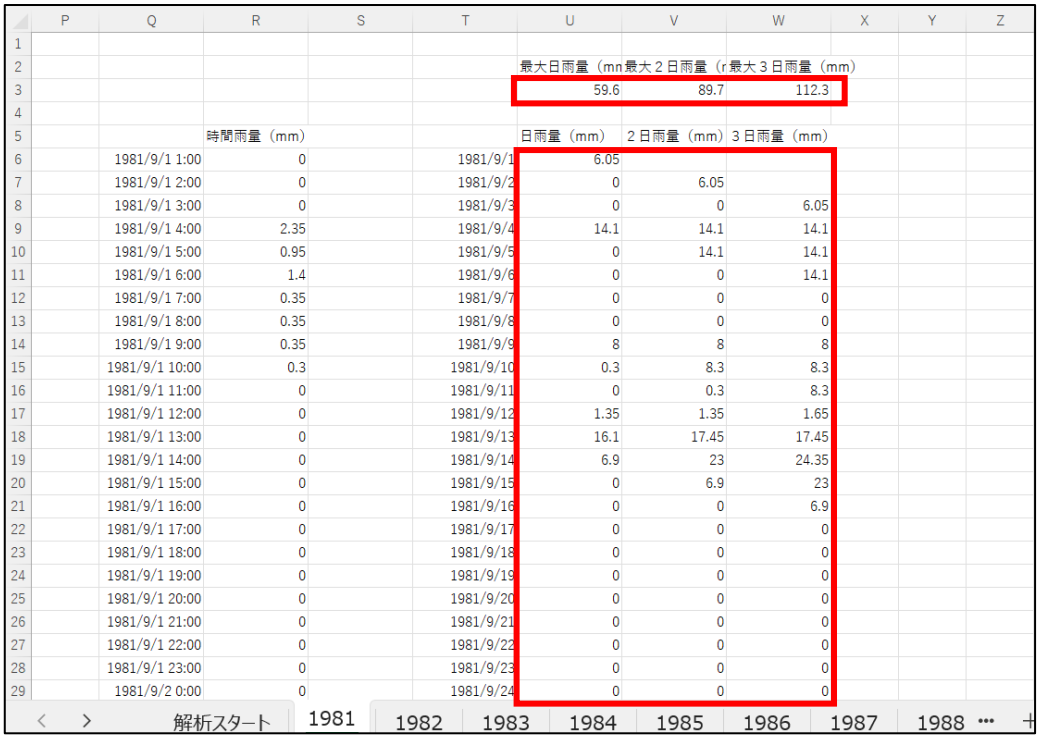
「解析スタートシート」内の「雨量累積ボタン」を押すと、各降雨量シートの U～W 列に時間雨量の累積計算（日・２日・３日）・年最大雨量（日・２日・３日）が算出

※　使用する PC の環境によっては、処理に時間がかかる場合があるため、その場合は、一回に処理（追加）するシートを少なくするなどして分けて作業すること

（解析スタートシート）




（降雨量シート）



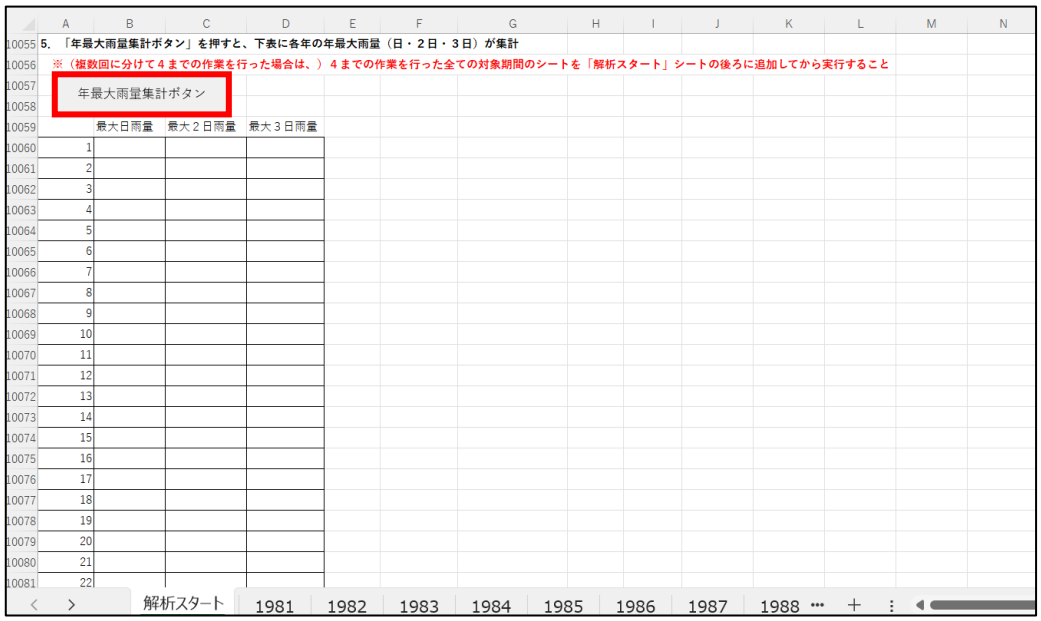
（５）　年最大雨量の整理

「解析スタートシート」内の「年最大雨量集計ボタン」を押し、下表に各年の年最大雨量（日・２日・３日）を抽出

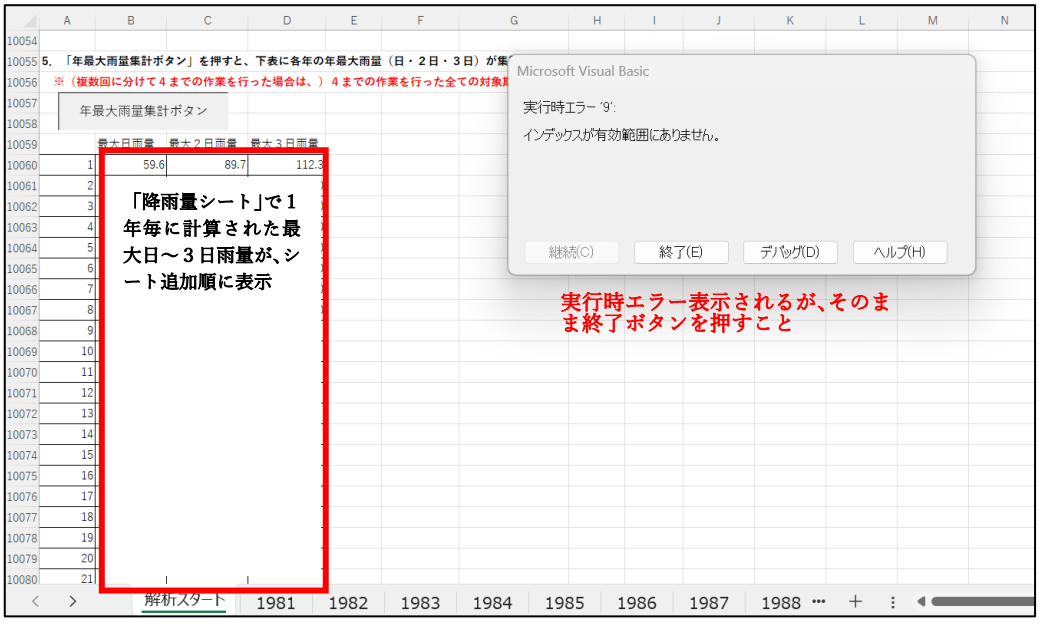
※　ボタンを押した際に実行時エラー表示されるが、そのまま終了ボタンを押すこと

※　複数回に分けて（４）までの作業を行った場合は、（４）までの作業を行った全ての対象期間のシートを「解析スタート」シートの後ろ（右側）に追加してから実行すること

（解析スタートシート）




（解析スタートシート）



（6）　確率降雨量の算出

（5）で集計した値を用いて、日～3日連続雨量における 1/10 等の確率降雨量を算出

※　確率降雨量の算出方法は、土地改良事業計画設計基準　計画「排水」技術書「6．実績降雨に基づく計画基準降雨」を参照

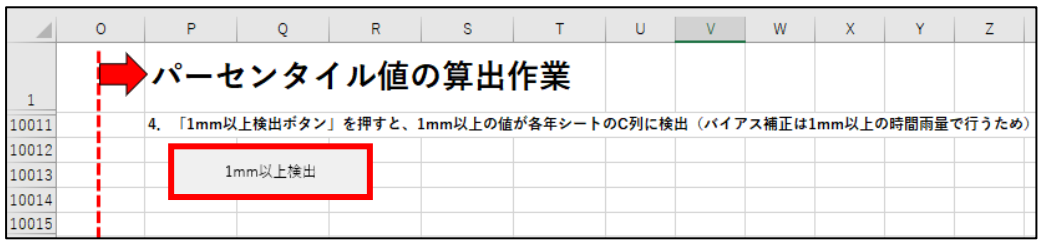
## 2　パーセンタイル値の算出

以降の（1）及び（2）の作業の結果は、第5章第2節において実験値のバイアス補正をする上で必要となる。

### （1）　時間1mm以上の降雨値の抽出

「解析スタートシート」内の「1mm以上検出ボタン」を押し、各年「降雨量シート」のS列に1mm以上の値を抽出

（解析スタートシート）




（降雨量シート）

| | 時間雨量（mm） | 1mm以上の数値 | | 日雨量（mm） | 2日雨量（mm） | 3日雨量（mm） |
|---|---|---|---|---|---|---|
| | | | | 最大日雨量（mm | 最大2日雨量（r | 最大3日雨量（mm） |
| | | | | 59.6 | 89.7 | 112.3 |
| 1981/9/1 1:00 | 0 | | 1981/9/1 | 6.05 | | |
| 1981/9/1 2:00 | 0 | | 1981/9/2 | 0 | 6.05 | |
| 1981/9/1 3:00 | 0 | | 1981/9/3 | 0 | 0 | 6.05 |
| 1981/9/1 4:00 | 2.35 | 2.35 | 1981/9/4 | 14.1 | 14.1 | 14.1 |
| 1981/9/1 5:00 | 0.95 | | 1981/9/5 | 0 | 14.1 | 14.1 |
| 1981/9/1 6:00 | 1.4 | 1.4 | 1981/9/6 | 0 | 0 | 14.1 |
| 1981/9/1 7:00 | 0.35 | | 1981/9/7 | 0 | 0 | 0 |
| 1981/9/1 8:00 | 0.35 | | 1981/9/8 | 0 | 0 | 0 |
| 1981/9/1 9:00 | 0.35 | | 1981/9/9 | 8 | 8 | 8 |
| 1981/9/1 10:00 | 0.3 | | 1981/9/10 | 0.3 | 8.3 | 8.3 |
| 1981/9/1 11:00 | 0 | | 1981/9/11 | 0 | 0.3 | 8.3 |
| 1981/9/1 12:00 | 0 | | 1981/9/12 | 1.35 | 1.35 | 1.65 |
| 1981/9/1 13:00 | 0 | | 1981/9/13 | 16.1 | 17.45 | 17.45 |
| 1981/9/1 14:00 | 0 | | 1981/9/14 | 6.9 | 23 | 24.35 |
| 1981/9/1 15:00 | 0 | | 1981/9/15 | 0 | 6.9 | 23 |
| 1981/9/1 16:00 | 0 | | 1981/9/16 | 0 | 0 | 6.9 |
| 1981/9/1 17:00 | 0 | | 1981/9/17 | 0 | 0 | 0 |
| 1981/9/1 18:00 | 0 | | 1981/9/18 | 0 | 0 | 0 |
| 1981/9/1 19:00 | 0 | | 1981/9/19 | 0 | 0 | 0 |
| 1981/9/1 20:00 | 0 | | 1981/9/20 | 0 | 0 | 0 |
| 1981/9/1 21:00 | 0 | | 1981/9/21 | 0 | 0 | 0 |
| 1981/9/1 22:00 | 0 | | 1981/9/22 | 0 | 0 | 0 |
| 1981/9/1 23:00 | 0 | | 1981/9/23 | 0 | 0 | 0 |
| 1981/9/2 0:00 | 0 | | 1981/9/24 | 0 | 0 | 0 |

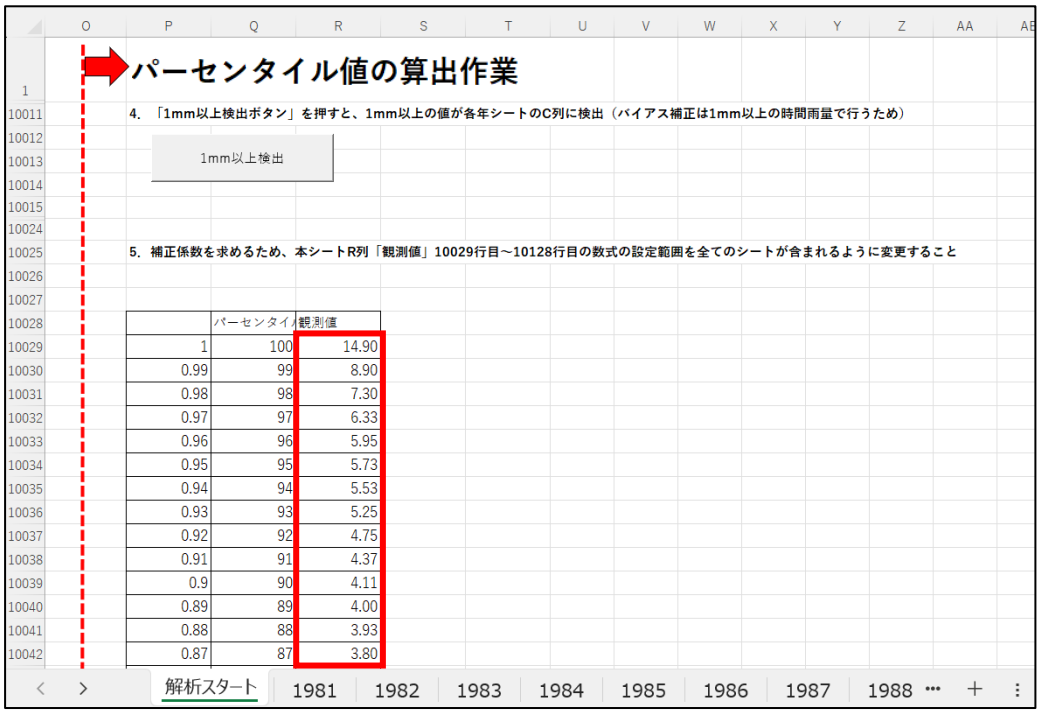
解析スタート　1981　1982　1983　1984　1985　1986　1987　1988

（２）パーセンタイル値の算出

「PERCENTILE.INC」関数を使用し、1mm 以上の観測値における１％毎のパーセンタイル値を算出　※　必ず、全ての期間のシートを追加した上で作業すること

「解析スタートシート」内の R 列 10029 行目から 10128 行目の数式を全てのシートが計算対象範囲に含まれるように変更すること（例：計算するシートの１番目が「1981」で最後が「2010」であれば、R 列 10029 行目のセルの関数を「=PERCENTILE.INC(1981:2010!$S$6:$S$9000,P10029)」と修正し、数式を R 列 10128 行目までドラッグすること）

（解析スタートシート）




数式を変更すること

=PERCENTILE.INC(**最初のシート名**:**最後のシート名**!$S$6:$S$9000,P10029)

# 第４章　実験データの処理

気候変動を踏まえた排水計画を策定するにあたり、気候予測資料（データセット）は d2PDF（５km）を使用する。d2PDF（５km）データは、文部科学省の委託事業として開発が進められてきたデータ統合・解析システム（以降、DIAS）（https://diasjp.net/）上で「全国 5km メッシュアンサンブル気候予測データ」という名称で公開されているが、DIAS 上では、netCDF 形式のファイルで公開されており、事前に Python 等を使用したプログラミングにより CSV 形式のファイルに変換する必要があるため（降雨量数値の見える化）、その方法について説明する。

## 第１節　実験データの入手

気候予測資料（データセット）の d2PDF（５km）データは、農林水産省農村振興局整備部設計課から地方農政局等を通じて貸与する。なお、貸与データは、本マニュアルにより土地改良事業計画（排水）における将来の降雨予測に基づく確率降雨量算定にのみ使用するものとし、配布先は各貸与元において管理するものとする。

また、DIAS より d2PDF（５km）データを入手する場合は、巻末に付記している「附属資料１　実験データの入手」を参照する。

## 第２節　対象地域の過去実験値及び将来実験値の csv 化

第１節で入手した「rain.nc」及び「rain.nc_pdef.ctl」ファイルを用いて実験データを csv 化する必要があるため、本項では、Python を用いた netCDF ファイルの csv 化の例を示す。

### 1　Python 実行プログラム及びシェイプファイルの準備

巻末に付記している「附属資料２　Python プログラムコード」は、複数のディレクトリにある rain.nc ファイルから降雨量データを抽出し、対象地域（シェイプファイルで指定）に対応するデータを csv 形式で出力するもの。なお、csv 形式で出力される対象地域の降雨量は、d2PDF（５km）のメッシュ（以降、グリッド）に含まれるシェイプファイルの面積に応じて加重平均されたものである。

「附属資料２　Python プログラムコード」をコマンドにより実行するため、事前に拡張子 py ファイル（ファイル名：netcdf2csv_arealmean_cnst.py、別途配布）を PC 上に保存しておく。d2PDF（５km）の緯度経度情報を得るため、DIAS より「cnst.nc」をダウンロードして PC 上に保存しておく。また、GIS（地理情報システム）ソフトにより対象地域のシェイプファイルを作成し、PC 上に保存しておく。

### 2　csv 化コマンドの実行

コマンドを実行すると出力ディレクトリの直下に入力ディレクトリと同名のサブディレクトリが作成され、csv ファイルが格納される。

（１）　コマンドの記載方法

「＄」→「netcdf2csv_arealmean_cnst.py」→「-o 出力ディレクトリパス」→「-s シェイプファイルの保存ディレクトリパス」→「--shift-hours -48」→「--min-coverage 0」→「--export-cells-shp 出力シェイプファイル名」→「入力ディレクトリパス」の順番で記載する。

（ターミナル画面：コマンド入力例）

```
$ python3 netcdf2csv_arealmean_cnst.py -o ./csv -s 排水ブロック N_EPSG4326.shp --shift-hours -48 --min-coverage 0 --export-cells-shp ./csv/HFB_2K_CC_m101/selected_cells.shp ../HFB_2K_CC_m101/
```

上記コマンド入力例では、相対パスを例示しているが、絶対パスを使用できる。

（パス設定を行う際の参考）

現在の作業ディレクトリは下記のコマンドで確認できる。

（ターミナル画面：コマンド入力例）

```
$ pwd
```

（２）　コマンドの説明

-o ［csv ファイルの出力パス］

・・　output、出力ディレクトリパスを設定する。d2PDF（５km）から抽出した降雨が出力される。

-s ［対象地域のシェイプファイル］

・・　shapefile、抽出対象の地点を指定するシェイプファイルを設定する。

--shift-hours -48

・・　時刻データを時間単位で補正（-48 を指定）する。

--min-coverage 0

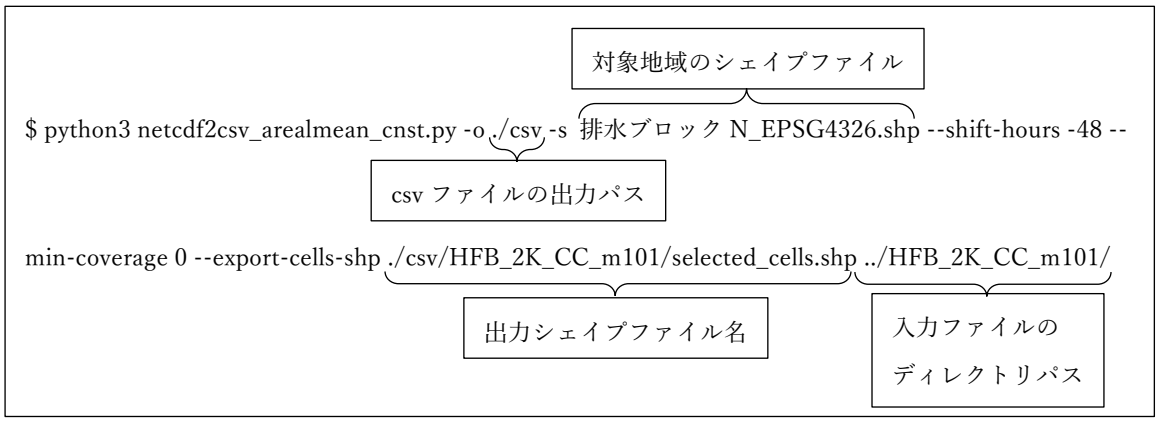
・・　採用する最小の被覆率を指定（0 を指定）する（被覆率については P21 に記載）。

--export-cells-shp ［出力シェイプファイル名］

・・　採用グリッドの出力シェイプファイル名（拡張子は shp、gpkg、geojson から選択可能）を設定する。

［入力ファイルのディレクトリパス］

・・　入力ディレクトリパス(複数指定可)を設定する。



（参考）

usage: netcdf2csv_arealmean_cnst.py [-h] -o OUTPUT -s SHAPEFILE --shift-hours --min-coverage --export-cells-shp inputs [inputs ...]
②③⑤⑥ ⑦ ①

positional arguments:

① inputs
入力ディレクトリパス(複数指定可)
入力ディレクトリ下に存在するすべての rain.nc ファイルを探索し，入力ファイルとする
出力サブディレクトリは入力ディレクトリと同じ名前で作成される
ex) ../HFB_2K_*/

optional arguments:
-h, --help show this help message and exit

② -o OUTPUT, --output OUTPUT
出力ディレクトリパス
ファイルの出力先ディレクトリ
出力ディレクトリの直下に入力ディレクトリと同名のサブディレクトリが作成される
ex) -o ../csv/

③ -s SHAPEFILE, --shapefile SHAPEFILE
対象地点 shapefile
抽出対象の地点を指定する shapefile
CRS は EPSG:4326(緯度経度)であること
ex) -s ../排水ブロック N_EPSG4326.shp

④今回の作業では必要ないが、③シェイプファイルの情報の代わりに、緯度経度を指定すれば、１地点の降雨量データも確認することが可能

④ -p POINT, --point POINT
対象地点の緯度経度
抽出対象地点の緯度経度
緯度,経度 の順にカンマ区切りで指定する
(カンマの前後にスペースを入れないこと)
(-p と-s の両方を指定した場合，こちらは無視される)
ex) -p 36.0329092313968,140.09731122949984

⑤ --shift-hours SHIFT_HOURS
読み込んだ日時を指定時間だけシフトする
ex) -48

⑥ --min-coverage MIN_COVERAGE
採用する最小被覆率（default=1e-12）
0 で『少しでも重なれば』採用する

⑦ --export-cells-shp EXPORT_CELLS_SHP
採用セルの範囲ポリゴンを出力(shp/gpkg/geojson)

（３）　出力される csv ファイル

１）　対象グリッドの全雨量データ：rain_allgrids_[yyyy].csv

d2PDF（５km）のうち、対象地域が該当するグリッドの全雨量データが年毎に出力される。対象地域のデータが抽出されているか、確認するための参考データとして使用する。

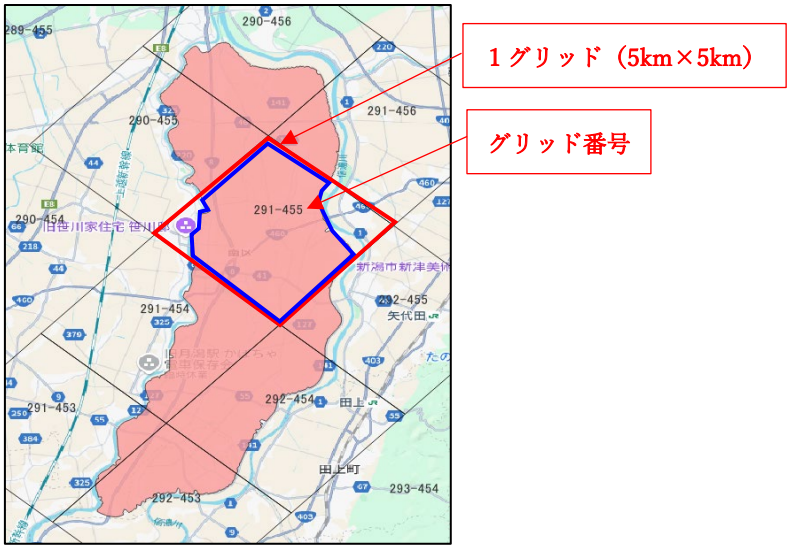
２）　地域平均雨量：rain_arealmean_[yyyy].csv

地域平均雨量は、d2PDF（５km）のグリッドに含まれるシェイプファイルの面積に応じて加重平均※されたものが年毎に出力される。本マニュアル第５章、第６章では、このデータを用いる。

※　面積加重平均雨量＝総和（各グリッドの雨量×被覆率）／総和（被覆率）

被覆率=グリッドに占めるポリゴンの面積（青枠部分）／１グリッドの面積（25km$^2$）（赤枠部分）

（例：GIS 上における d2PDF（５km）グリッドとの関係）



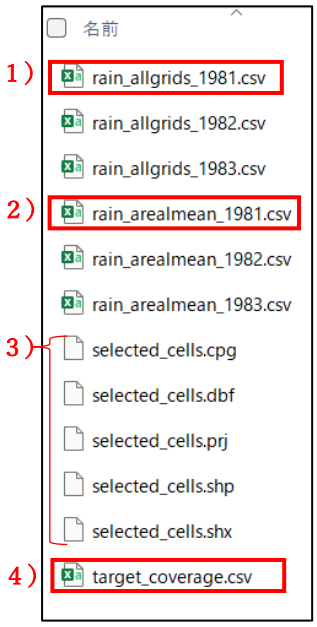

３）　対象グリッドのシェイプファイル出力：selected_cells.shp

対象グリッドのシェイプファイルが出力される。このシェイプファイルを GIS 上にインポートすると、グリッド番号を確認できる。

４）　対象グリッドの面積率：target_coverage.csv

対象グリッド毎に被覆率が出力される。面積加重平均雨量を確認するための参考データとして使用する。

（出力イメージ：過去実験値　1981 年～1983 年を出力した場合）




1）対象グリッドの全雨量データ

グリッド番号（地図上の数字）と整合しているか確認

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | date-hour | 291_453 | 292_453 | 291_454 | 292_454 | 290_455 | 291_455 | 292_455 | 290_456 | 291_456 |
| 2 | 1981/9/1 0:00 | 1.515076 | 1.461891 | 1.541069 | 1.47187 | 1.587837 | 1.521133 | 1.43457 | 1.51091 | 1.46273 |
| 3 | 1981/9/1 1:00 | 3.300064 | 2.142395 | 3.209442 | 2.212921 | 4.044678 | 3.147827 | 2.279976 | 3.52034 | 3.075714 |
| 4 | 1981/9/1 2:00 | 9.239891 | 7.811279 | 9.347359 | 7.524849 | 10.11089 | 9.178932 | 7.02491 | 10.02097 | 8.478424 |
| 5 | 1981/9/1 3:00 | 0.308846 | 0.445488 | 0.449409 | 0.774719 | 0.429657 | 0.710632 | 1.442802 | 0.53714 | 1.226959 |
| 6 | 1981/9/1 4:00 | 0.081345 | 0.155083 | 0.088089 | 0.148926 | 0.058586 | 0.091949 | 0.153473 | 0.056458 | 0.096382 |
| 7 | 1981/9/1 5:00 | 0.056946 | 0.069977 | 0.025322 | 0.038353 | 0.012741 | 0.011307 | 0.024208 | 0.005562 | 0.004524 |
| 8 | 1981/9/1 6:00 | 0.681519 | 0.415558 | 0.532249 | 0.371147 | 0.399765 | 0.411789 | 0.315865 | 0.229027 | 0.284233 |
| 9 | 1981/9/1 7:00 | 0.162132 | 0.126938 | 0.215546 | 0.179398 | 0.461838 | 0.308846 | 0.246277 | 0.569038 | 0.40699 |
| 10 | 1981/9/1 8:00 | 0.229507 | 0.088768 | 0.167572 | 0.086792 | 0.034431 | 0.128464 | 0.087898 | 0.039749 | 0.116241 |
| 11 | 1981/9/1 9:00 | 0.020699 | 0.006615 | 0.012444 | 0.004082 | 0.01487 | 0.00753 | 0.003624 | 0.009277 | 0.006226 |
| 12 | 1981/9/1 10:00 | 0.034523 | 0.004555 | 0.041939 | 0.006195 | 0.184258 | 0.050453 | 0.00869 | 0.181076 | 0.059433 |
| 13 | 1981/9/1 11:00 | 3.05E-05 | 0 | 6.87E-05 | 0 | 0.001503 | 0.000237 | 2.29E-05 | 0.002403 | 0.000732 |
| 14 | 1981/9/1 12:00 | 0.000465 | 0 | 0.000259 | 0 | 0.00647 | 0.000145 | 0 | 0.005188 | 9.16E-05 |
| 15 | 1981/9/1 13:00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

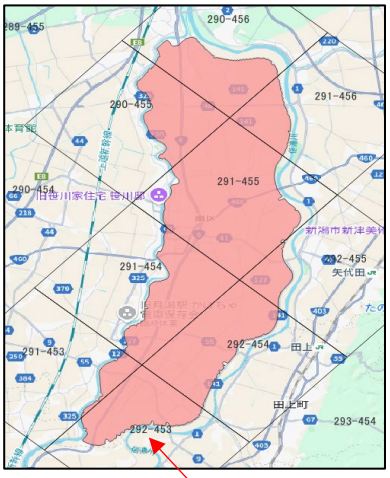
各グリッドの時間雨量

2）地域平均雨量

地域平均雨量

| | A | B |
|---|---|---|
| 1 | date-hour | rainfall |
| 2 | 1981/9/1 0:00 | 1.509467 |
| 3 | 1981/9/1 1:00 | 3.003704 |
| 4 | 1981/9/1 2:00 | 8.874269 |
| 5 | 1981/9/1 3:00 | 0.652825 |
| 6 | 1981/9/1 4:00 | 0.101944 |
| 7 | 1981/9/1 5:00 | 0.024269 |
| 8 | 1981/9/1 6:00 | 0.398258 |
| 9 | 1981/9/1 7:00 | 0.298949 |
| 10 | 1981/9/1 8:00 | 0.104757 |
| 11 | 1981/9/1 9:00 | 0.008587 |
| 12 | 1981/9/1 10:00 | 0.063337 |
| 13 | 1981/9/1 11:00 | 0.00052 |
| 14 | 1981/9/1 12:00 | 0.001284 |
| 15 | 1981/9/1 13:00 | 0 |

3）対象グリッドのシェイプファイル出力




グリッド番号

4）対象グリッドの面積率

グリッドの被覆率

| | A | B | C |
|---|---|---|---|
| 1 | i | j | fraction_area |
| 2 | 291 | 453 | 0.062625 |
| 3 | 292 | 453 | 0.294764 |
| 4 | 291 | 454 | 0.391916 |
| 5 | 292 | 454 | 0.439995 |
| 6 | 290 | 455 | 0.279855 |
| 7 | 291 | 455 | 0.807092 |
| 8 | 292 | 455 | 0.095453 |
| 9 | 290 | 456 | 0.297298 |
| 10 | 291 | 456 | 0.14549 |

グリッド番号

（４）　資料の整理

フォルダに格納された csv ファイルを整理する。

フォルダ名、ファイル名は以下のように整理しておくと後の作業がしやすくなる。

## 3　抽出データの確認方法

csv 化され、抽出されたデータが対象地域のデータとなっているか確認する方法は、以下のとおり。

（1）　GIS（地理情報システム）ソフトによる確認

2の（3）の3）で出力されたシェイプファイルを GIS 上にインポートすると、d2PDF（5 km）の採用グリッドとグリッド番号が表示できる。

GIS 上に表示されたグリッド番号と対象地域のシェイプファイルの位置関係から、抽出したグリッドが間違いないかを確認する。

（2）　検算による確認

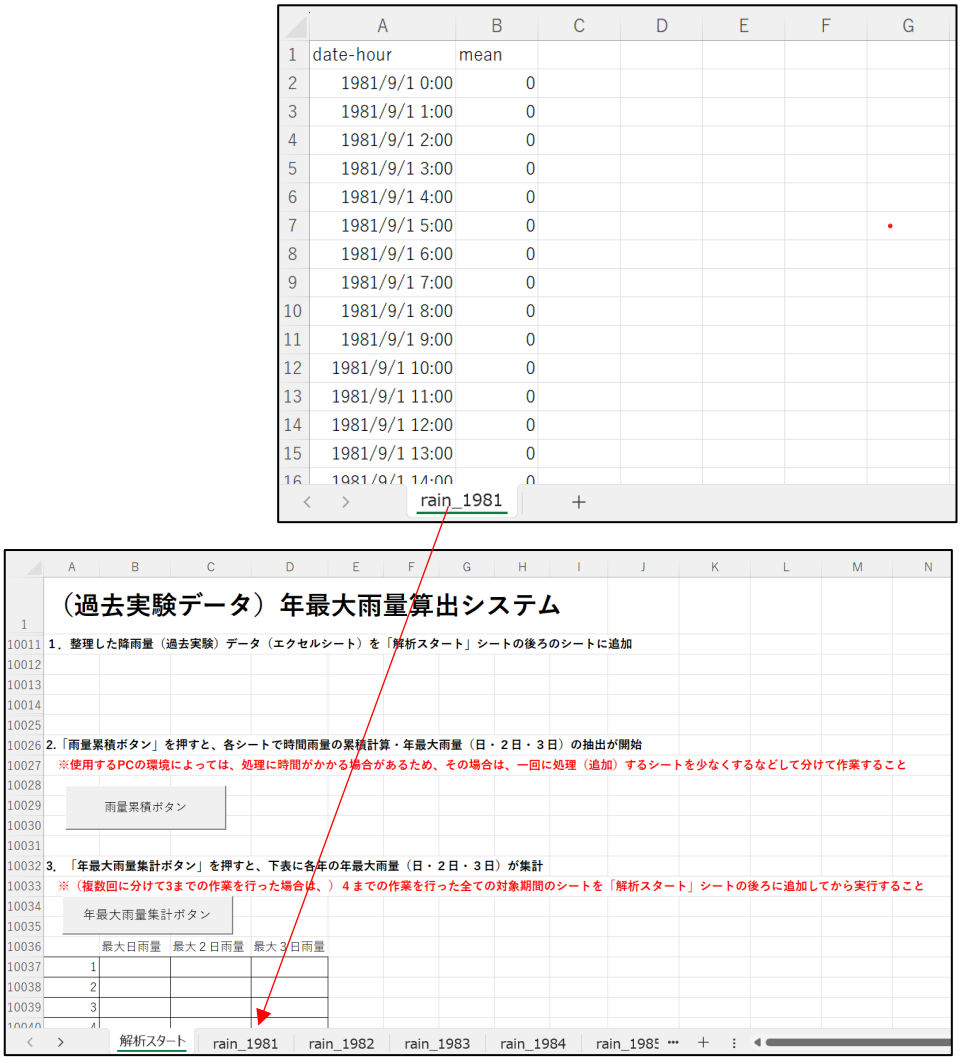
対象グリッドの全雨量データ（rain_allgrids_[yyyy].csv）及び対象グリッドの面積率（target_coverage.csv）を用いて地域平均雨量を検算し、地域平均雨量（rain_arealmean_[yyyy].csv）との整合を確認する。

# 第５章　過去実験データの処理

## 第１節　確率降雨量の算出

### １　データの追加

『(過去）年最大雨量算出システム』を開き、「解析スタート」シートの後ろに d2PDF の降雨量シートを追加（例：1981 年～2010 年の期間のデータを選定した場合、「解析スタート」シートの後ろ（右側）に 12 メンバ×30 年（1981 年～2010 年）のデータを追加）




※シート名は、どのメンバか分かるようにしたうえで追加する。

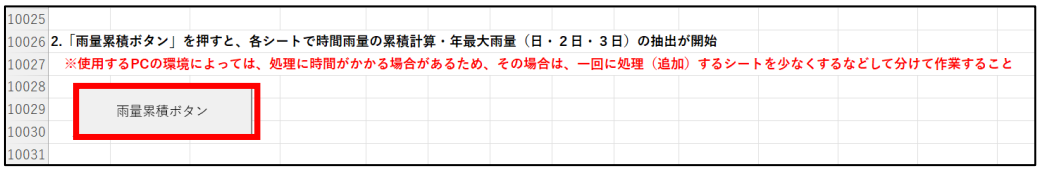
（例：HPB_m002→rain_1981(2)、HPB_m012→rain_1981(12)など）

## 2　年最大雨量の算出

「解析スタートシート」内の「雨量累積ボタン」を押すと、各降雨量シートで時間雨量の累積計算（日・２日・３日）・年最大雨量（日・２日・３日）が算出

※　使用する PC の環境によっては、処理に時間がかかる場合があるため、その場合は、一回に処理（追加）するシートを少なくするなどして分けて作業すること

（解析スタートシート）




（降雨量シート）

年最大雨量
（日～３日）

累積雨量
（日～３日）

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | date-hour | mean | | | 最大日雨量（mm） | 最大２日雨量（mm） | 最大３日雨量（mm） | |
| 2 | 1981/9/1 0:00 | 0 | | | 53.93763856 | 56.98011744 | 60.25484833 | |
| 3 | 1981/9/1 1:00 | 0 | | | | | | |
| 4 | 1981/9/1 2:00 | 0 | | | 日雨量（mm） | ２日雨量（mm） | ３日雨量（mm） | |
| 5 | 1981/9/1 3:00 | 0 | | 1981/9/1 | 0 | | | |
| 6 | 1981/9/1 4:00 | 0 | | 1981/9/2 | 0 | 0 | | |
| 7 | 1981/9/1 5:00 | 0 | | 1981/9/3 | 0 | 0 | 0 | |
| 8 | 1981/9/1 6:00 | 0 | | 1981/9/4 | 50.66648333 | 50.66648333 | 50.66648333 | |
| 9 | 1981/9/1 7:00 | 0 | | 1981/9/5 | 3.762165556 | 54.42864889 | 54.42864889 | |
| 10 | 1981/9/1 8:00 | 0 | | 1981/9/6 | 0.020168556 | 3.782334111 | 54.44881744 | |
| 11 | 1981/9/1 9:00 | 0 | | 1981/9/7 | 8.230930667 | 8.251099222 | 12.01326478 | |
| 12 | 1981/9/1 10:00 | 0 | | 1981/9/8 | 3.427639667 | 11.65857033 | 11.67873889 | |
| 13 | 1981/9/1 11:00 | 0 | | 1981/9/9 | 0.480409556 | 3.908049222 | 12.13897989 | |
| 14 | 1981/9/1 12:00 | 0 | | 1981/9/10 | | | 3.908049222 | |
| 15 | 1981/9/1 13:00 | 0 | | 1981/9/11 | | 0 | 0.480409556 | |
| 16 | 1981/9/1 14:00 | 0 | | 1981/9/12 | | 0 | 0 | |
| 17 | 1981/9/1 15:00 | 0 | | 1981/9/13 | | 0 | 0 | |
| 18 | 1981/9/1 16:00 | 0 | | 1981/9/14 | | 0 | 0 | |
| 19 | 1981/9/1 17:00 | 0 | | 1981/9/15 | 0 | 0 | 0 | |
| 20 | 1981/9/1 18:00 | 0 | | 1981/9/16 | 5.964225556 | 5.964225556 | 5.964225556 | |

解析スタート　rain_1981　rain_1982　rain_1983　rain_1984　rain_198…

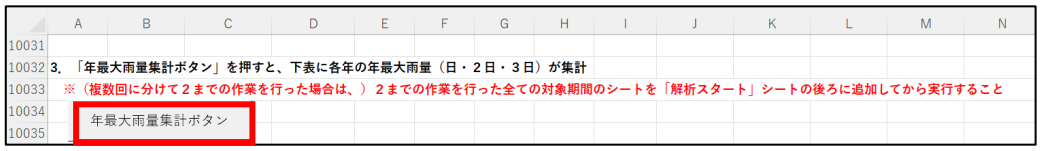
## 3　年最大雨量の整理

「解析スタートシート」内の「年最大雨量集計ボタン」を押し、表に各年の年最大雨量（日・２日・３日）を抽出

※　ボタンを押した際に実行時エラー表示されるが、そのまま終了ボタンを押すこと

※　複数回に分けて２までの作業を行った場合は、２までの作業を行った全ての対象期間のシートを「解析スタート」シートの後ろに追加してから実行すること

（解析スタートシート）




（解析スタートシート）


## 4　確率降雨量の算出

３で集計した値を用いて、日～３日連続雨量における 1/10 等の確率降雨量を算出

※　確率降雨量の算出方法は、土地改良事業計画設計基準　計画「排水」技術書「６．実績降雨に基づく計画基準降雨」を参照

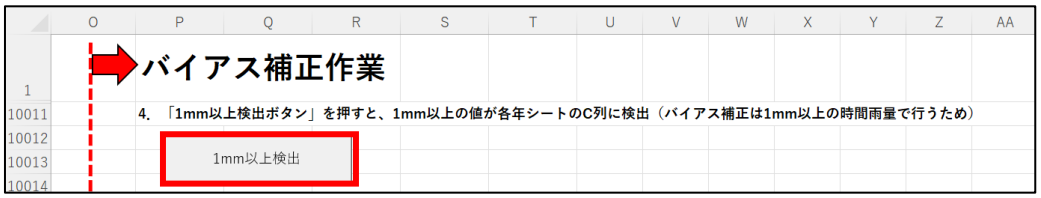
## 第２節　バイアス補正

過去実験値と観測値それぞれの時間降雨量のパーセンタイル値を比較し、1%毎に補正係数を求めて、過去実験値の時間降雨量に乗ずることにより、過去実験値のバイアス補正を行う。

### １　時間 1mm 以上の降雨値の抽出

「解析スタートシート」内の「1mm 以上検出ボタン」を押し、各年降雨量シートの C 列に 1mm 以上の値を抽出

（解析スタートシート）




（降雨量シート）

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | date-hour | mean | 1mm以上の数値 | | 最大日雨量（mm） | 最大２日雨量（mm） | 最大３日雨量（mm） | |
| 2 | 1981/9/1 0:00 | 0 | | | 53.93763856 | 56.98011744 | 60.25484833 | |
| 3 | 1981/9/1 1:00 | 0 | | | | | | |
| 4 | 1981/9/1 2:00 | 0 | | | 日雨量（mm） | ２日雨量（mm） | ３日雨量（mm） | |
| 5 | 1981/9/1 3:00 | 0 | | 1981/9/1 | 0 | | | |
| 6 | 1981/9/1 4:00 | 0 | 1 | 1981/9/2 | 0 | 0 | | |
| 7 | 1981/9/1 5:00 | 0 | m | 1981/9/3 | 0 | 0 | 0 | |
| 8 | 1981/9/1 6:00 | 0 | m | 1981/9/4 | 50.66648333 | 50.66648333 | 50.66648333 | |
| 9 | 1981/9/1 7:00 | 0 | 以 | 1981/9/5 | 3.762165556 | 54.42864889 | 54.42864889 | |
| 10 | 1981/9/1 8:00 | 0 | 上 | 1981/9/6 | 0.020168556 | 3.782334111 | 54.44881744 | |
| 11 | 1981/9/1 9:00 | 0 | の | 1981/9/7 | 8.230930667 | 8.251099222 | 12.01326478 | |
| 12 | 1981/9/1 10:00 | 0 | 値 | 1981/9/8 | 3.427639667 | 11.65857033 | 11.67873889 | |
| 13 | 1981/9/1 11:00 | 0 | が | 1981/9/9 | 0.480409556 | 3.908049222 | 12.13897989 | |
| 14 | 1981/9/1 12:00 | 0 | 抽 | 1981/9/10 | 0 | 0.480409556 | 3.908049222 | |
| 15 | 1981/9/1 13:00 | 0 | 出 | 1981/9/11 | 0 | 0 | 0.480409556 | |
| 16 | 1981/9/1 14:00 | 0 | | 1981/9/12 | 0 | 0 | 0 | |
| 17 | 1981/9/1 15:00 | 0 | | 1981/9/13 | 0 | 0 | 0 | |
| 18 | 1981/9/1 16:00 | 0 | | 1981/9/14 | 0 | 0 | 0 | |
| 19 | 1981/9/1 17:00 | 0 | | 1981/9/15 | 0 | 0 | 0 | |
| 20 | 1981/9/1 18:00 | 0 | | 1981/9/16 | 5.964225556 | 5.964225556 | 5.964225556 | |

解析スタート　rain_1981　rain_1982　rain_1983　rain_1984　rain_198…

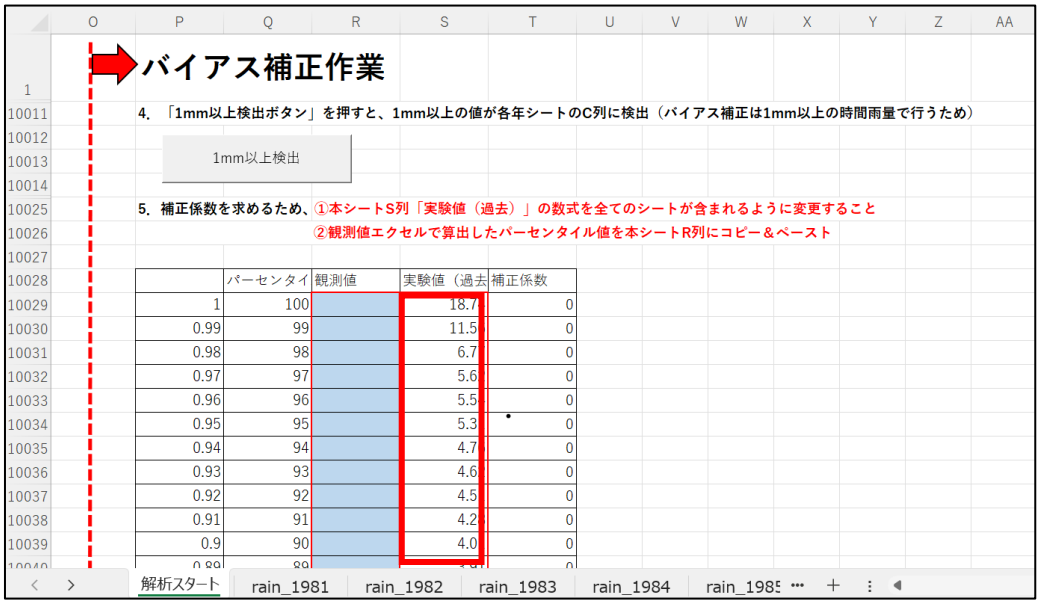
## 2　補正係数の算出

補正係数を求めるため、「PERCENTILE.INC」関数を使用し、１％毎にパーセンタイル値を算出　※　必ず、全ての期間のシートを追加した上で作業すること

（１）パーセンタイル値の算出

「解析スタートシート」内の S 列 10029 行目から 10128 行目の数式を全てのシートが対象範囲に含まれるように変更すること（例：計算するシートの 1 番目が rain_1981 で最後が rain_2010(12)であれば、S 列 10029 行目のセルの関数を「=PERCENTILE.INC(rain_1981:rain_2010(12)!$C$2:$C$9000,P10029)」と修正し、数式を S 列 10128 行目までドラッグすること）

（解析スタートシート）




数式を変更すること

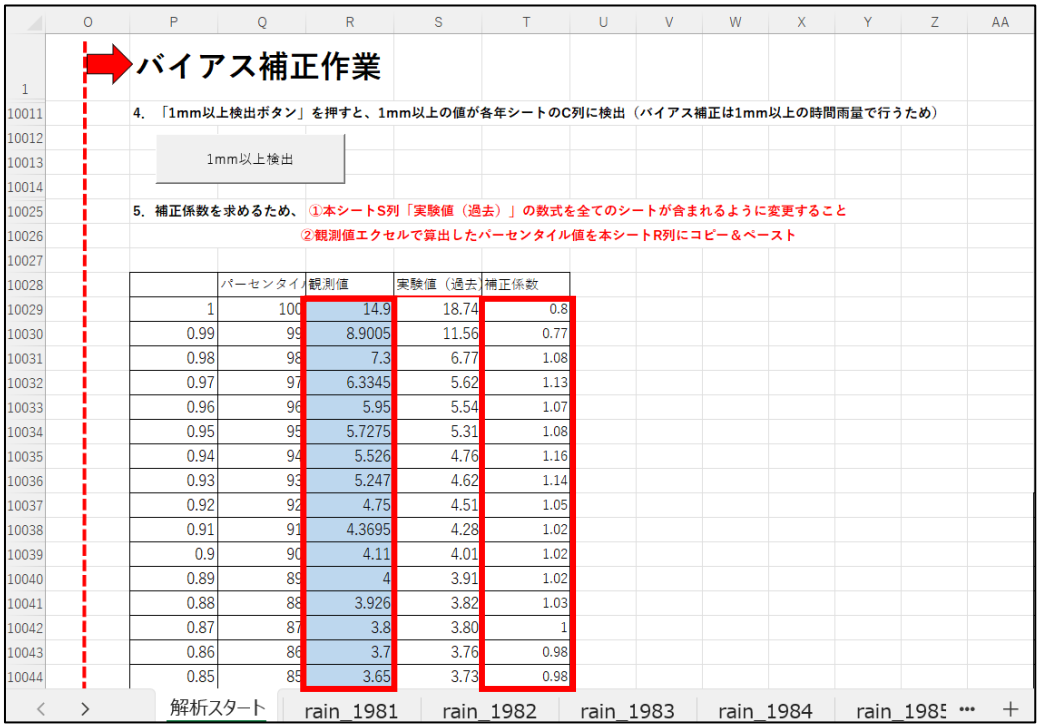
S10029=PERCENTILE.INC(**最初のシート名:最後のシート名**!$C$2:$C$9000,P10029)

（２） 補正係数の算出

別途観測値エクセルで算出したパーセンタイル値を「解析スタートシート」内の R 列 10029 行目～10128 行目にコピー＆ペースト

実験値（過去）と観測値を比較した結果が、補正係数として T 列に表示

（解析スタートシート）



バイアス補正作業

4. 「1mm以上検出ボタン」を押すと、1mm以上の値が各年シートのC列に検出（バイアス補正は1mm以上の時間雨量で行うため）

1mm以上検出

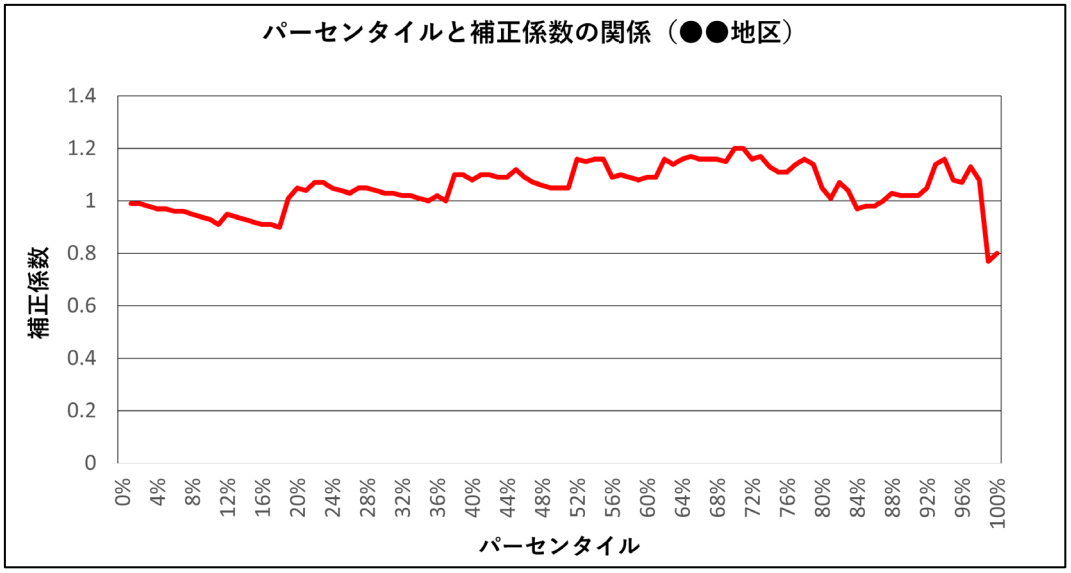
5. 補正係数を求めるため、 ①本シートS列「実験値（過去）」の数式を全てのシートが含まれるように変更すること
②観測値エクセルで算出したパーセンタイル値を本シートR列にコピー＆ペースト

| | パーセンタイ | 観測値 | 実験値（過去 | 補正係数 |
|---|---|---|---|---|
| 1 | 100 | 14.9 | 18.74 | 0.8 |
| 0.99 | 99 | 8.9005 | 11.56 | 0.77 |
| 0.98 | 98 | 7.3 | 6.77 | 1.08 |
| 0.97 | 97 | 6.3345 | 5.62 | 1.13 |
| 0.96 | 96 | 5.95 | 5.54 | 1.07 |
| 0.95 | 95 | 5.7275 | 5.31 | 1.08 |
| 0.94 | 94 | 5.526 | 4.76 | 1.16 |
| 0.93 | 93 | 5.247 | 4.62 | 1.14 |
| 0.92 | 92 | 4.75 | 4.51 | 1.05 |
| 0.91 | 91 | 4.3695 | 4.28 | 1.02 |
| 0.9 | 90 | 4.11 | 4.01 | 1.02 |
| 0.89 | 89 | 4 | 3.91 | 1.02 |
| 0.88 | 88 | 3.926 | 3.82 | 1.03 |
| 0.87 | 87 | 3.8 | 3.80 | 1 |
| 0.86 | 86 | 3.7 | 3.76 | 0.98 |
| 0.85 | 85 | 3.65 | 3.73 | 0.98 |

解析スタート　rain_1981　rain_1982　rain_1983　rain_1984　rain_198

（参考）「解析スタートシート」内にパーセンタイルと補正係数の関係を示すグラフが作成される

（表示例）

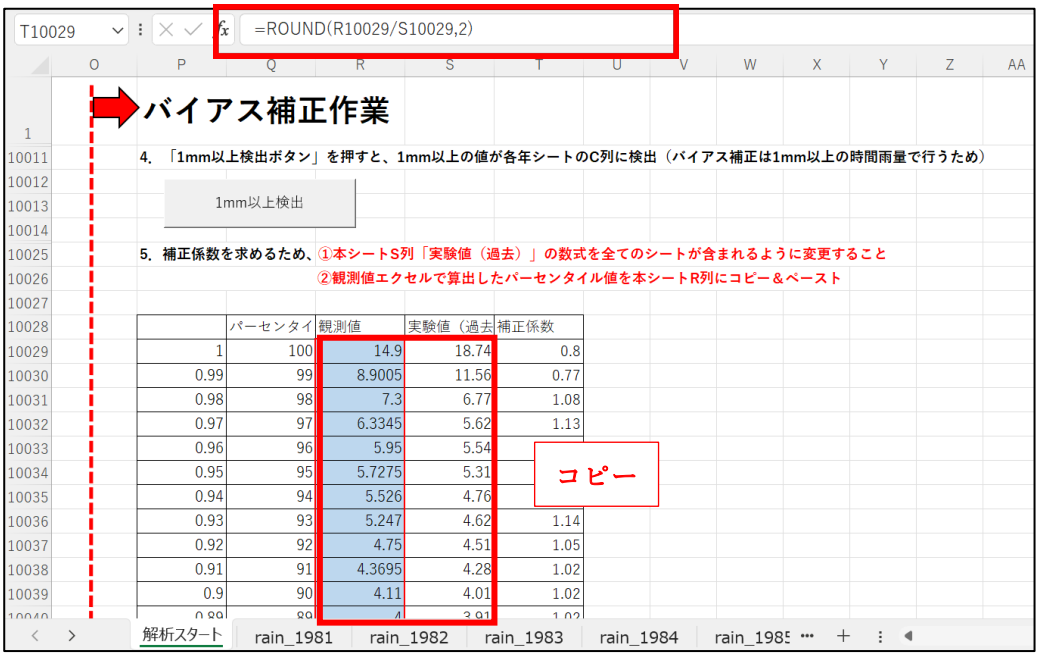


（３） 数値の貼り付け

「解析スタートシート」内の R 列「観測値」10029 行目～10128 行目と S 列「実験値（過去)」10029 行目～10128 行目をコピーし、同じ場所に数値をペースト（数式をペーストしない）

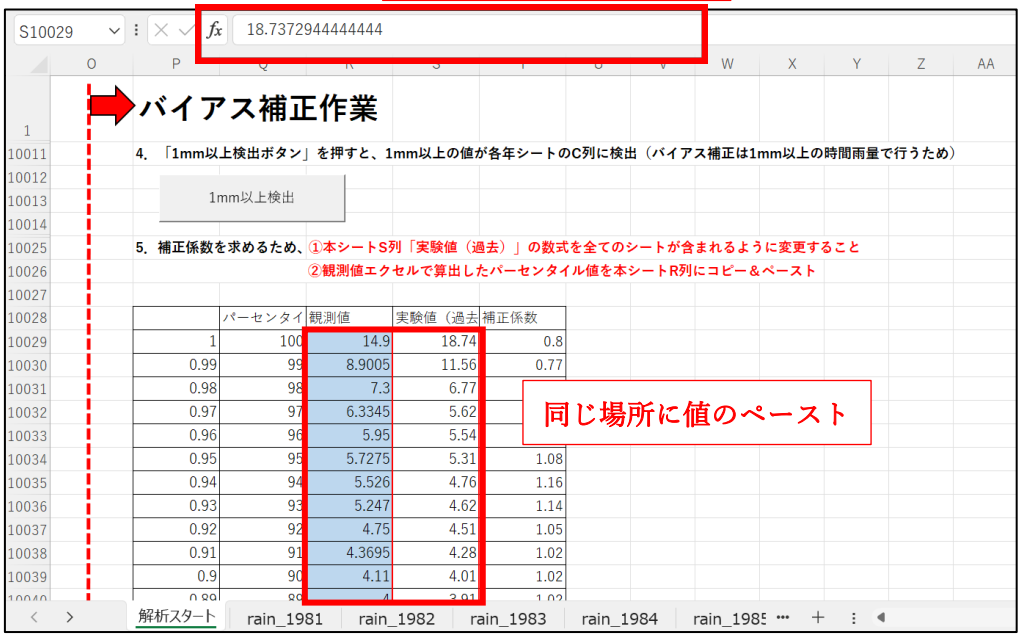
（解析スタートシート）

関数が入力されている状態




（解析スタートシート）

値が入力されている状態



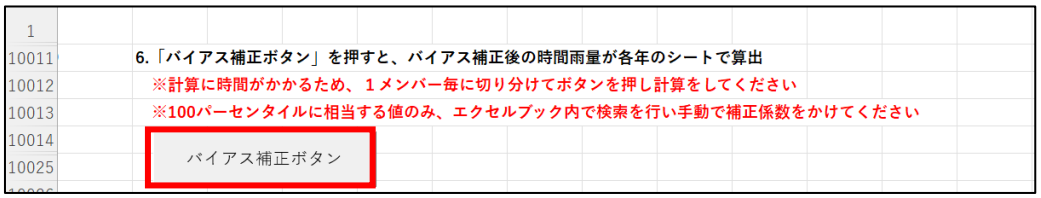
## 3　時間雨量の補正

「解析スタートシート」内の「バイアス補正ボタン」を押すと、各年降雨量シートのＪ列にバイアス補正後の時間雨量が算出

※　計算に時間がかかるため、１メンバ毎に切り分けた計算（ボタンを押す）を推奨

※　100 パーセンタイルに相当する値のみ、エクセルブック内で検索を行い手動で補正係数をかけること

（解析スタートシート）




（降雨量シート）

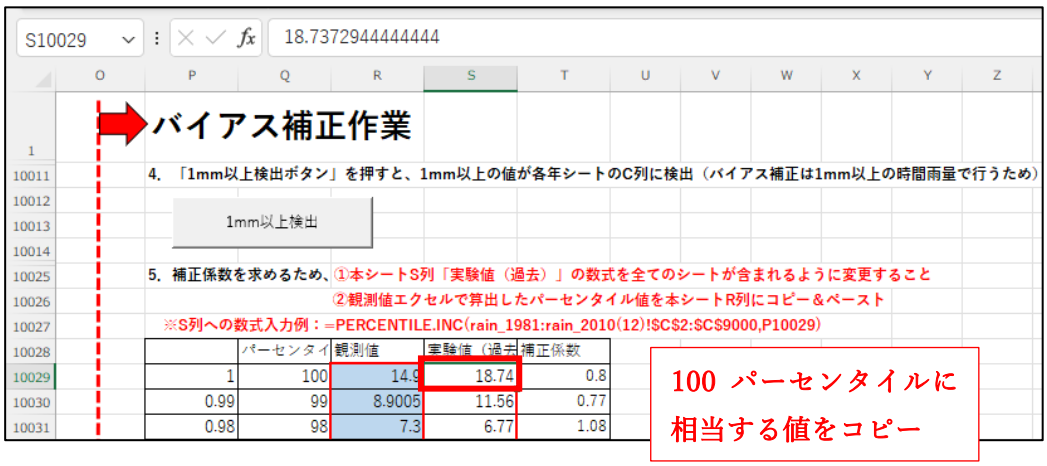
**バイアス補正後の数値**

| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | date-hour | mean | 1mm以上の数値 | | 最大日雨量（mm | 最大２日雨量（m | 最大３日雨量（mm） | | date-hour | バイアス補正後 | |
| 2 | 2031/9/1 0:00 | 0 | | | 90.04054067 | 93.69310133 | 101.3212879 | | 2031/9/1 0:00 | 0 | |
| 3 | 2031/9/1 1:00 | 0 | | | | | | | 2031/9/1 1:00 | 0 | |
| 4 | 2031/9/1 2:00 | 0 | | | 日雨量（mm） | ２日雨量（mm） | ３日雨量（mm） | | 2031/9/1 2:00 | 0 | |
| 5 | 2031/9/1 3:00 | 0 | | 2031/9/1 | 0.706500556 | | | | 2031/9/1 3:00 | 0 | |
| 6 | 2031/9/1 4:00 | 0 | | 2031/9/2 | 0.123322222 | 0.829822778 | | | 2031/9/1 4:00 | 0 | |
| 7 | 2031/9/1 5:00 | 0 | | 2031/9/3 | 0.567161556 | 0.690483778 | 1.396984333 | | 2031/9/1 5:00 | 0 | |
| 8 | 2031/9/1 6:00 | 0.005284667 | | 2031/9/4 | 0 | 0.567161556 | 0.690483778 | | 2031/9/1 6:00 | 0.0052847 | |
| 9 | 2031/9/1 7:00 | 0.108332333 | | 2031/9/5 | 0.063595222 | 0.063595222 | 0.630756778 | | 2031/9/1 7:00 | 0.1083323 | |
| 10 | 2031/9/1 8:00 | 0.462775333 | | 2031/9/6 | 2.190212222 | 2.253807444 | 2.253807444 | | 2031/9/1 8:00 | 0.4627753 | |
| 11 | 2031/9/1 9:00 | 0.130108222 | | 2031/9/7 | 0.539259444 | 2.729471667 | 2.793066889 | | 2031/9/1 9:00 | 0.1301082 | |
| 12 | 2031/9/1 10:00 | 0 | | 2031/9/8 | 2.368858556 | 2.908118 | 5.098330222 | | 2031/9/1 10:00 | 0 | |
| 13 | 2031/9/1 11:00 | 0 | | 2031/9/9 | 15.22911033 | 17.59796889 | 18.13722833 | | 2031/9/1 11:00 | 0 | |
| 14 | 2031/9/1 12:00 | 0 | | 2031/9/10 | 25.80585611 | 41.03496644 | 43.403825 | | 2031/9/1 12:00 | 0 | |
| 15 | 2031/9/1 13:00 | 0 | | 2031/9/11 | 60.28632144 | 86.09217756 | 101.3212879 | | 2031/9/1 13:00 | 0 | |
| 16 | 2031/9/1 14:00 | 0 | | 2031/9/12 | 0.012042444 | 60.29836389 | 86.10422 | | 2031/9/1 14:00 | 0 | |
| 17 | 2031/9/1 15:00 | 0 | | 2031/9/13 | 0 | 0.012042444 | 60.29836389 | | 2031/9/1 15:00 | 0 | |
| 18 | 2031/9/1 16:00 | 0 | | 2031/9/14 | 0 | 0 | 0.012042444 | | 2031/9/1 16:00 | 0 | |
| 19 | 2031/9/1 17:00 | 0 | | 2031/9/15 | 0 | 0 | 0 | | 2031/9/1 17:00 | 0 | |

解析スタート　rain_2031　rain_2032　rain_2033　rain_2034　rain_2035

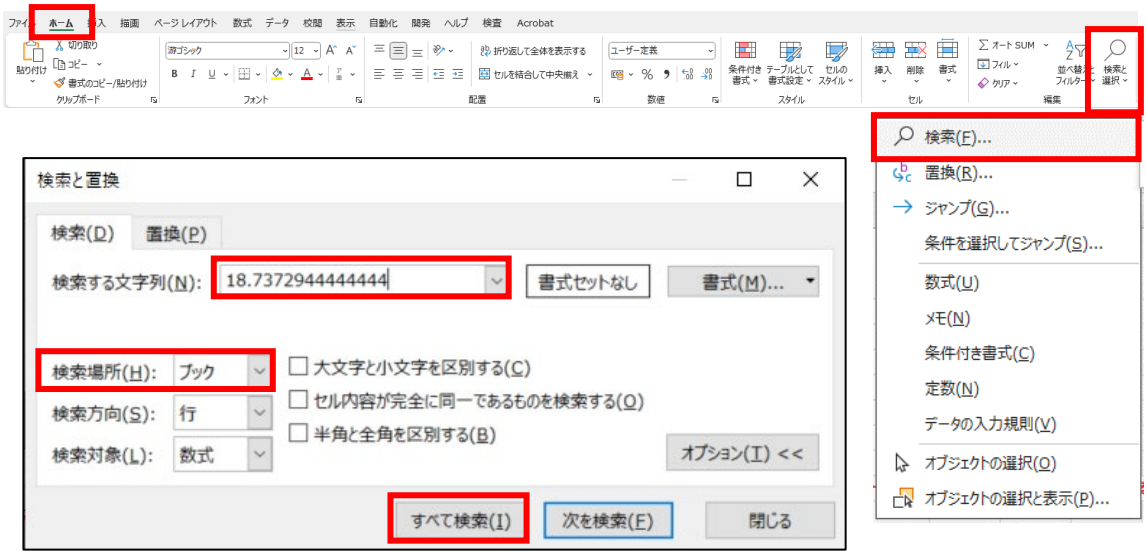
※　100 パーセンタイルに相当する値の補正方法

（解析スタートシート）

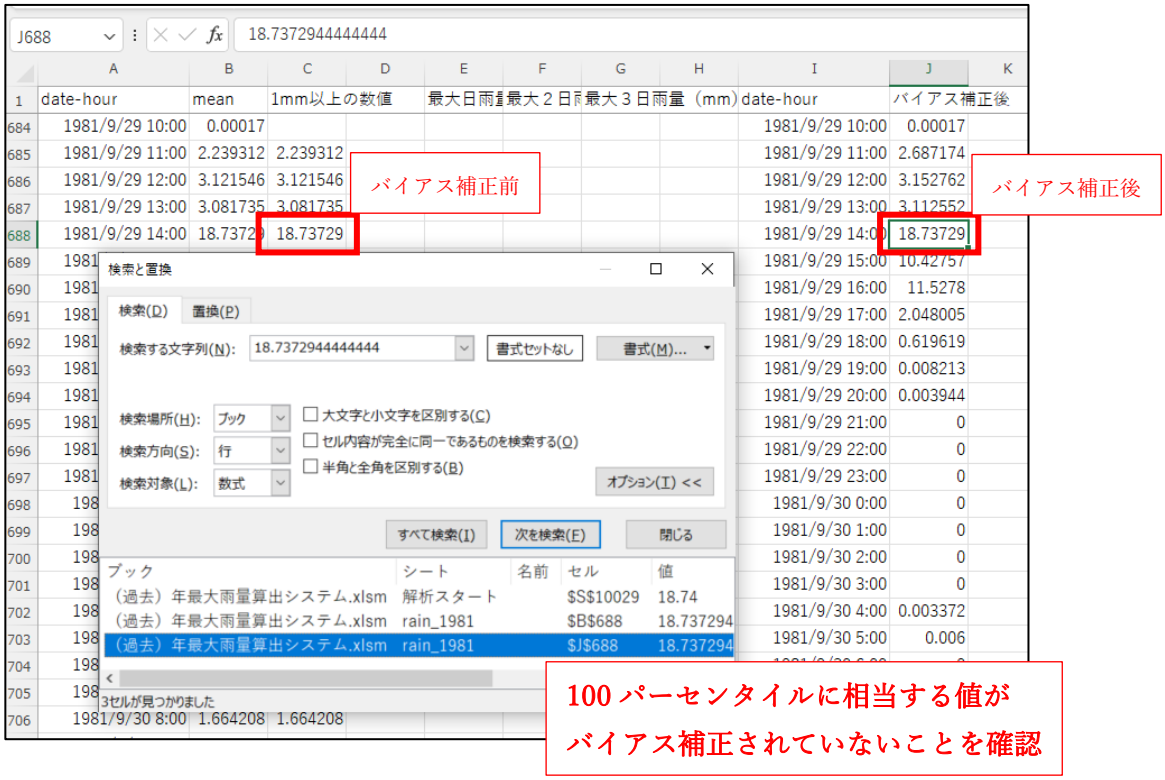



ホームタブのツールバー右側にある「検索と選択」より、検索をクリック

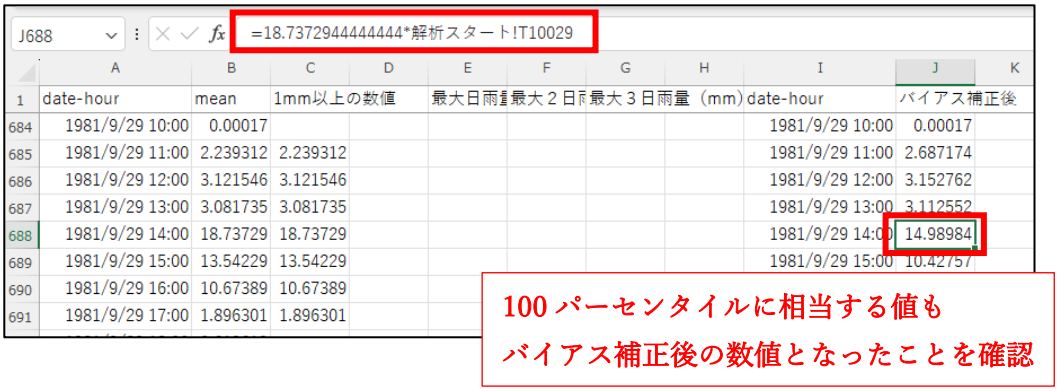
「検索と置換」の検索する文字列にコピーした値をペーストし、検索場所を「ブック」にして検索



（降雨量シート）




「バイアス補正後」欄の 100 パーセンタイルに相当する値のセルに、手動で計算式を入力して（「解析スタートシート」の補正係数を乗じて）補正する

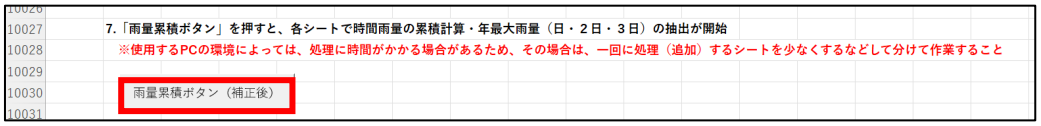


## 4　年最大雨量の算出

「解析スタートシート」内の「雨量累積ボタン（補正後）」を押すと、各降雨量シートで時間雨量の累積計算（日・２日・３日）・年最大雨量（日・２日・３日）が算出

※　使用する PC の環境によっては、処理に時間がかかる場合があるため、その場合は、一回に処理（追加）するシートを少なくするなどして分けて作業すること

（解析スタートシート）




（降雨量シート）

最大雨量
（日～３日）

| | date-hour | バイアス補正後 | | 最大日雨量（mm） | 最大２日雨量（mm | 最大３日雨量（mm） |
|---|---|---|---|---|---|---|
| 2 | 1981/9/1 0:00 | 0 | | 53.82963171 | 57.73729153 | 59.00581157 |
| 3 | 1981/9/1 1:00 | 0 | | | | |
| 4 | 1981/9/1 2:00 | 0 | | 日雨量（mm） | ２日雨量（mm） | ３日雨量（mm） |
| 5 | 1981/9/1 3:00 | 0 | 1981/9/1 | 0 | | |
| 6 | 1981/9/1 4:00 | 0 | 1981/9/2 | 0 | 0 | |
| 7 | 1981/9/1 5:00 | 0 | 1981/9/3 | 0 | 0 | 0 |
| 8 | 1981/9/1 6:00 | 0 | 1981/9/4 | 53.82963171 | 53.82963171 | 53.82963171 |
| 9 | 1981/9/1 7:00 | 0 | 1981/9/5 | 3.907659811 | 57.73729153 | 57.73729153 |
| 10 | 1981/9/1 8:00 | 0 | 1981/9/6 | 0.020168556 | 3.927828367 | 57.75746008 |
| 11 | 1981/9/1 9:00 | 0 | 1981/9/7 | 8.918071687 | 8.938240242 | 12.84590005 |
| 12 | 1981/9/1 10:00 | 0 | 1981/9/8 | 3.427639667 | 12.34571135 | 12.36587991 |
| 13 | 1981/9/1 11:00 | 0 | 1981/9/9 | 0.480409556 | 3.908049222 | 12.82612091 |
| 14 | 1981/9/1 12:00 | 0 | 981/9/10 | 0 | 0.480409556 | 3.908049222 |
| 15 | 1981/9/1 13:00 | 0 | 981/9/11 | | | 0.480409556 |
| 16 | 1981/9/1 14:00 | 0 | 981/9/12 | | | 0 |
| 17 | 1981/9/1 15:00 | 0 | 981/9/13 | | | 0 |
| 18 | 1981/9/1 16:00 | 0 | 981/9/14 | | | 0 |
| 19 | 1981/9/1 17:00 | 0 | 981/9/15 | 0 | 0 | 0 |
| 20 | 1981/9/1 18:00 | 0 | 981/9/16 | 5.922406298 | 5.922406298 | 5.922406298 |

累積雨量
（日～３日）

解析スタート　rain_1981　rain_1982　rain_1983　rain_1984　rain_198…

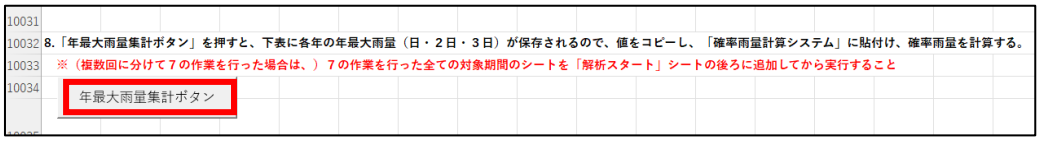
## 5　年最大雨量の整理

「解析スタートシート」内の「年最大雨量集計ボタン」を押すと、下表に各年の年最大雨量（日・２日・３日）が抽出

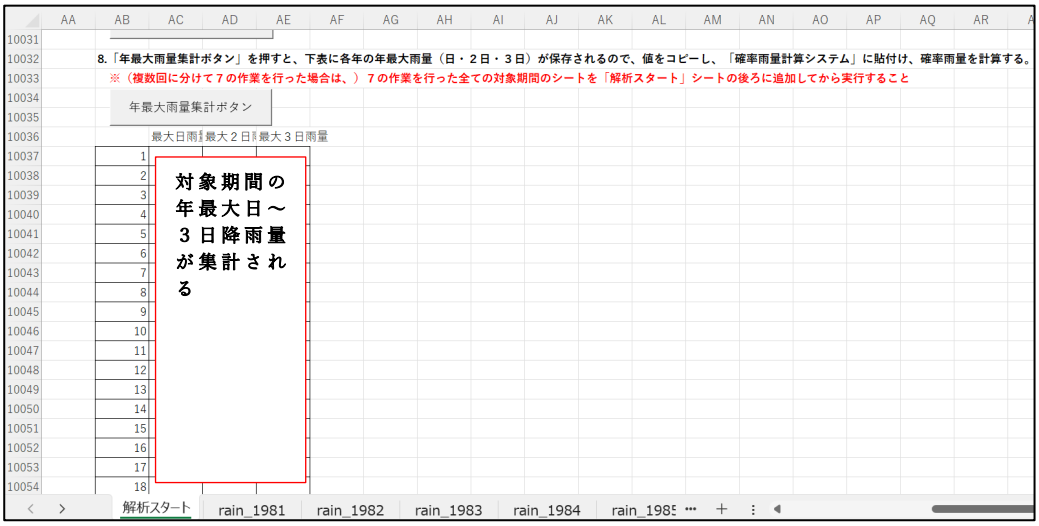
※　ボタンを押した際に実行時エラー表示されるが、そのまま終了ボタンを押すこと

※　複数回に分けて４の作業を行った場合は、４の作業を行った全ての対象期間のシートを「解析スタート」シートの後ろに追加してから実行すること

（解析スタートシート）




（解析スタートシート）




## 6　確率降雨量の算出

5 で集計した値を用いて、日～３日連続雨量における 1/10 等の確率降雨量を算出

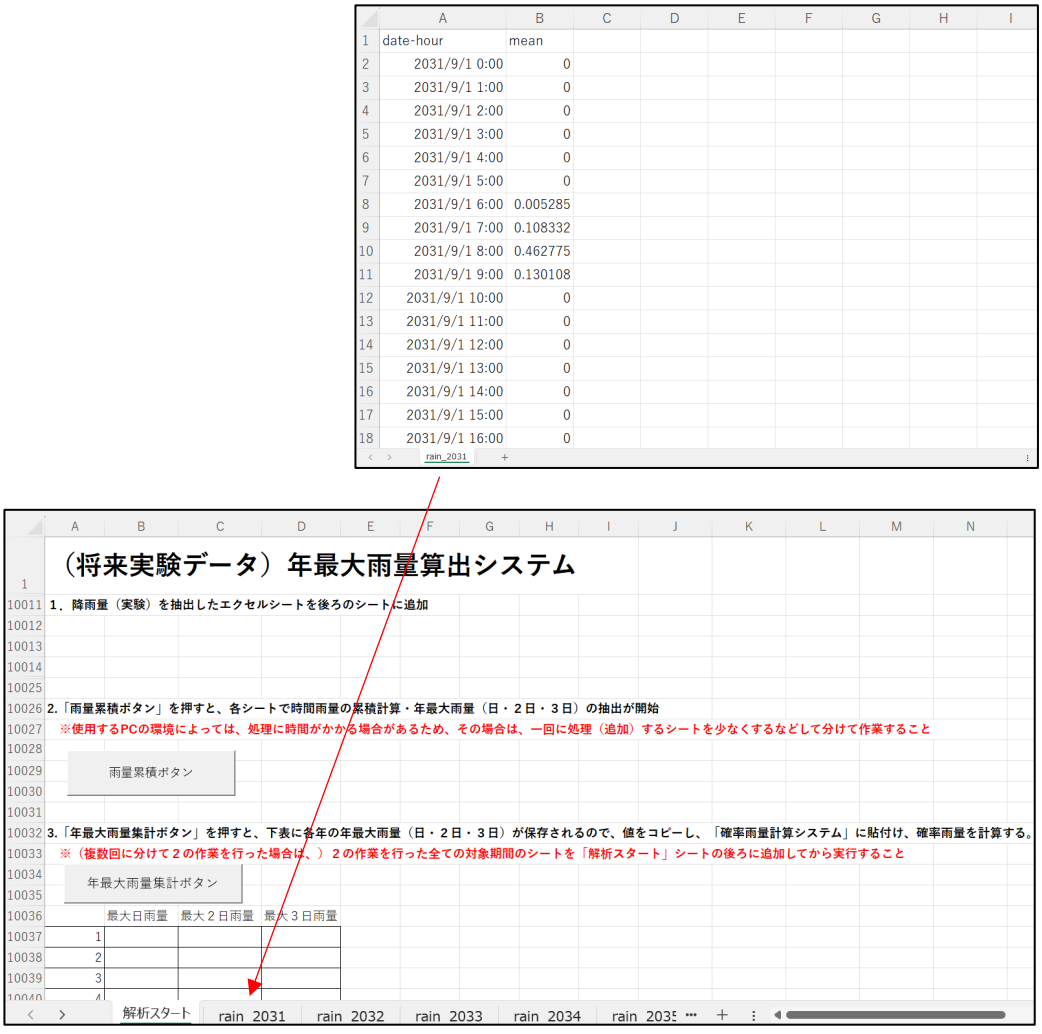
※　確率降雨量の算出方法は、土地改良事業計画設計基準　計画「排水」技術書「６．実績降雨に基づく計画基準降雨」を参照

# 第６章　将来実験データの処理

## 第１節　確率降雨量の算出

### １　データの追加

『(将来）年最大雨量算出システム』を開き、「解析スタート」シートの後ろ（右側）にd2PDF の降雨量（実験）エクセルシートを追加（12 メンバ×60 年分（2031 年～2090 年）




※シート名は、どのメンバか分かるようにしたうえで追加する。

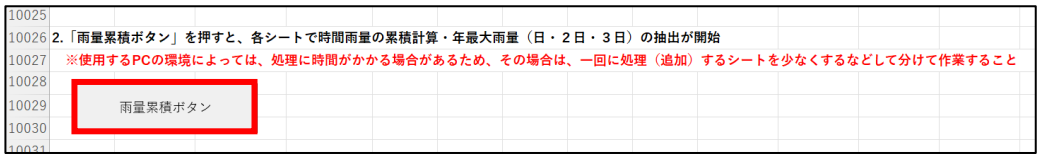
（例：HFB_2K_CC_m102→rain_2031(2)、HFB_2K_MR_m102→rain_2031(12)など）

## 2　年最大雨量の算出

「解析スタートシート」内の「雨量累積ボタン」を押すと、各降雨量シートで時間雨量の累積計算（日・２日・３日）・年最大雨量（日・２日・３日）が算出

※　使用する PC の環境によっては、処理に時間がかかる場合があるため、その場合は、一回に処理（追加）するシートを少なくするなどして分けて作業すること

（解析スタートシート）




（降雨量シート）

最大雨量（日～３日）

累積雨量（日～３日）

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | date-hour | mean | | | 最大日雨量（mm | 最大２日雨量（m | 最大３日雨量（mm） | |
| 2 | 2031/9/1 0:00 | 0 | | | 90.04054067 | 93.69310133 | 101.3212879 | |
| 3 | 2031/9/1 1:00 | 0 | | | | | | |
| 4 | 2031/9/1 2:00 | 0 | | | 日雨量（mm） | ２日雨量（mm） | ３日雨量（mm） | |
| 5 | 2031/9/1 3:00 | 0 | | 2031/9/1 | 0.706500556 | | | |
| 6 | 2031/9/1 4:00 | 0 | | 2031/9/2 | 0.123322222 | 0.829822778 | | |
| 7 | 2031/9/1 5:00 | 0 | | 2031/9/3 | 0.567161556 | 0.690483778 | 1.396984333 | |
| 8 | 2031/9/1 6:00 | 0.005284667 | | 2031/9/4 | 0 | 0.567161556 | 0.690483778 | |
| 9 | 2031/9/1 7:00 | 0.108332333 | | 2031/9/5 | 0.063595222 | 0.063595222 | 0.630756778 | |
| 10 | 2031/9/1 8:00 | 0.462775333 | | 2031/9/6 | 2.190212222 | 2.253807444 | 2.253807444 | |
| 11 | 2031/9/1 9:00 | 0.130108222 | | 2031/9/7 | 0.539259444 | 2.729471667 | 2.793066889 | |
| 12 | 2031/9/1 10:00 | 0 | | 2031/9/8 | 2.368858556 | 2.908118 | 5.098330222 | |
| 13 | 2031/9/1 11:00 | 0 | | 2031/9/9 | 15.22911033 | 17.59706889 | 18.13722833 | |
| 14 | 2031/9/1 12:00 | 0 | | 2031/9/10 | 25.805 | | 43.403825 | |
| 15 | 2031/9/1 13:00 | 0 | | 2031/9/11 | 60.286 | | 01.3212879 | |
| 16 | 2031/9/1 14:00 | 0 | | 2031/9/12 | 0.0120 | | 86.10422 | |
| 17 | 2031/9/1 15:00 | 0 | | 2031/9/13 | | | 0.29836389 | |
| 18 | 2031/9/1 16:00 | 0 | | 2031/9/14 | 0 | 0 | 0.012042444 | |
| 19 | 2031/9/1 17:00 | 0 | | 2031/9/15 | 0 | 0 | 0 | |

解析スタート　rain_2031　rain_2032　rain_2033　rain_2034　rain_2035

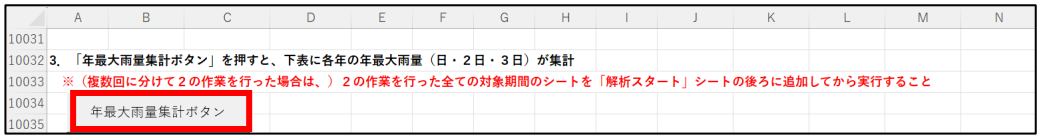
## 3　年最大雨量の整理

「解析スタートシート」内の「年最大雨量集計ボタン」を押すと、下表に各年の年最大雨量（日・２日・３日）が抽出

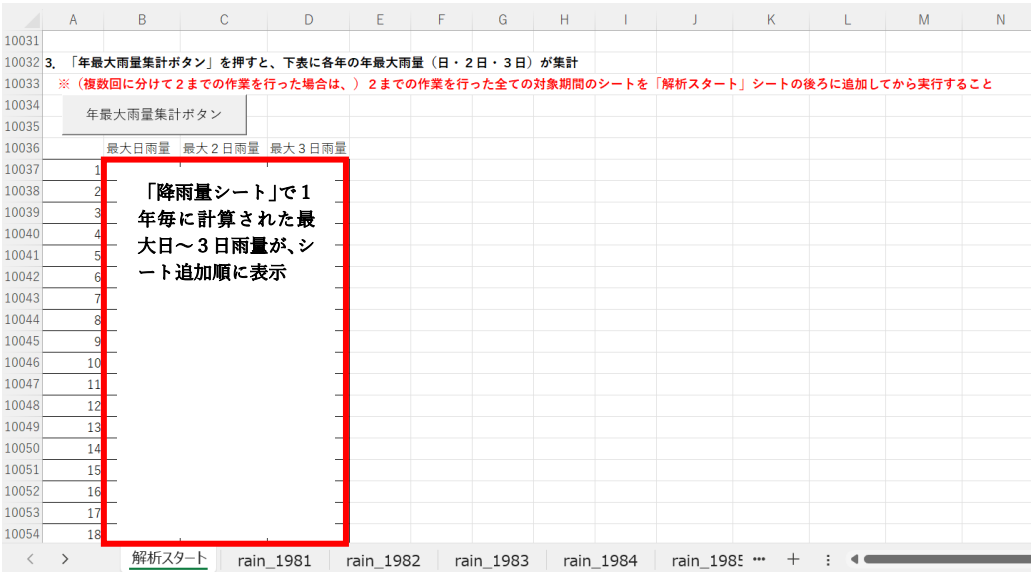
※　ボタンを押した際に実行時エラー表示されるが、そのまま終了ボタンを押すこと

※　複数回に分けて２の作業を行った場合は、２の作業を行った全ての対象期間のシートを「解析スタート」シートの後ろに追加してから実行すること

（解析スタートシート）




（解析スタートシート）




## 4　確率降雨量の算出

３で集計した値を用いて、日～３日連続雨量における 1/10 等の確率降雨量を算出

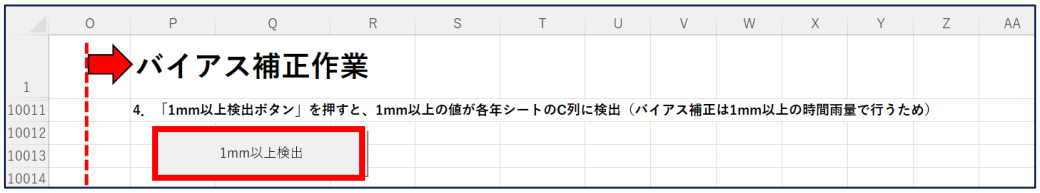
※　確率降雨量の算出方法は、土地改良事業計画設計基準　計画「排水」技術書「６．実績降雨に基づく計画基準降雨」を参照

# 第２節　バイアス補正

## 1　時間 1mm 以上の降雨値の抽出

「解析スタートシート」内の「1mm 以上検出ボタン」を押し、各年降雨量シートの C 列に 1mm 以上の値を抽出

（解析スタートシート）




（降雨量シート）

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | date-hour | mean | 1mm以上の数値 | | 最大日雨量（mm | 最大２日雨量（m | 最大３日雨量（mm） | |
| 2 | 2031/9/1 0:00 | 0 | | | 90.04054067 | 93.69310133 | 101.3212879 | |
| 3 | 2031/9/1 1:00 | 0 | | | | | | |
| 4 | 2031/9/1 2:00 | 0 | | | 日雨量（mm） | ２日雨量（mm） | ３日雨量（mm） | |
| 5 | 2031/9/1 3:00 | 0 | | 2031/9/1 | 0.706500556 | | | |
| 6 | 2031/9/1 4:00 | 0 | | 2031/9/2 | 0.123322222 | 0.829822778 | | |
| 7 | 2031/9/1 5:00 | 0 | | 2031/9/3 | 0.567161556 | 0.690483778 | 1.396984333 | |
| 8 | 2031/9/1 6:00 | 0.005284667 | | 2031/9/4 | 0 | 0.567161556 | 0.690483778 | |
| 9 | 2031/9/1 7:00 | 0.108332333 | | 2031/9/5 | 0.063595222 | 0.063595222 | 0.630756778 | |
| 10 | 2031/9/1 8:00 | 0.462775333 | | 2031/9/6 | 2.190212222 | 2.253807444 | 2.253807444 | |
| 11 | 2031/9/1 9:00 | 0.130108222 | | 2031/9/7 | 0.539259444 | 2.729471667 | 2.793066889 | |
| 12 | 2031/9/1 10:00 | 0 | | 2031/9/8 | 2.368858556 | 2.908118 | 5.098330222 | |
| 13 | 2031/9/1 11:00 | 0 | | 2031/9/9 | 15.22911033 | 17.59796889 | 18.13722833 | |
| 14 | 2031/9/1 12:00 | 0 | | 2031/9/10 | 25.80585611 | 41.03496644 | 43.403825 | |
| 15 | 2031/9/1 13:00 | 0 | | 2031/9/11 | 60.28632144 | 86.09217756 | 101.3212879 | |
| 16 | 2031/9/1 14:00 | 0 | | 2031/9/12 | 0.012042444 | 60.29836389 | 86.10422 | |
| 17 | 2031/9/1 15:00 | 0 | | 2031/9/13 | 0 | 0.012042444 | 60.29836389 | |
| 18 | 2031/9/1 16:00 | 0 | | 2031/9/14 | 0 | 0 | 0.012042444 | |
| 19 | 2031/9/1 17:00 | 0 | | 2031/9/15 | 0 | 0 | 0 | |

1mm以上の値が抽出

解析スタート　rain_2031　rain_2032　rain_2033　rain_2034　rain_2035

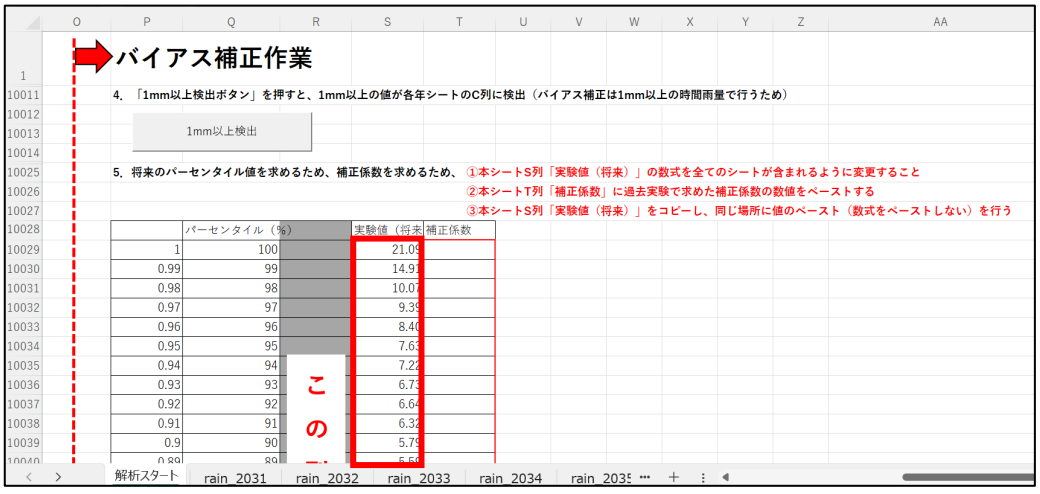
## 2　パーセンタイル値の算出

補正係数を求めるため、「PERCENTILE.INC」関数を使用し、１％毎にパーセンタイル値を算出　※　必ず、全ての期間のシートを追加した上で作業すること

（１）　「PERCENTILE.INC」関数の適用

「解析スタートシート」内の S 列 10029 行目から 10128 行目の数式を全てのシートが対象範囲に含まれるように変更すること（例：計算するシートの 1 番目が rain_2031 で最後が rain_2090(12)であれば、S 列 10029 行目のセルの関数を「=PERCENTILE.INC(rain_2031:rain_2090(12)!$C$2:$C$9000,P10029)」と修正し、数式を S 列 10128 行目までドラッグすること）

（解析スタートシート）

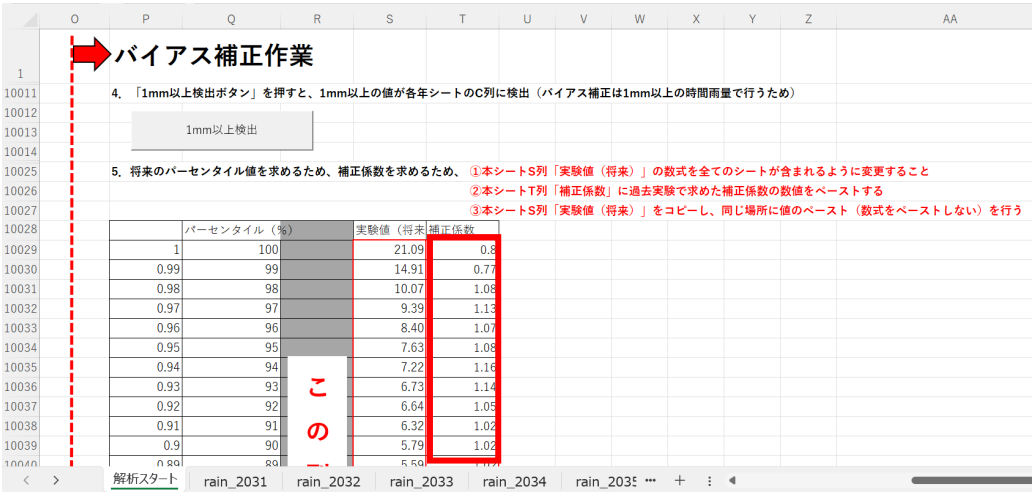



S10029=PERCENTILE.INC(**最初のシート名:最後のシート名**!$C$2:$C$9000,P10029)

（２）　補正係数の引用

本シート T 列「補正係数」に別途観測値と過去実験により求めた補正係数の数値をペーストする

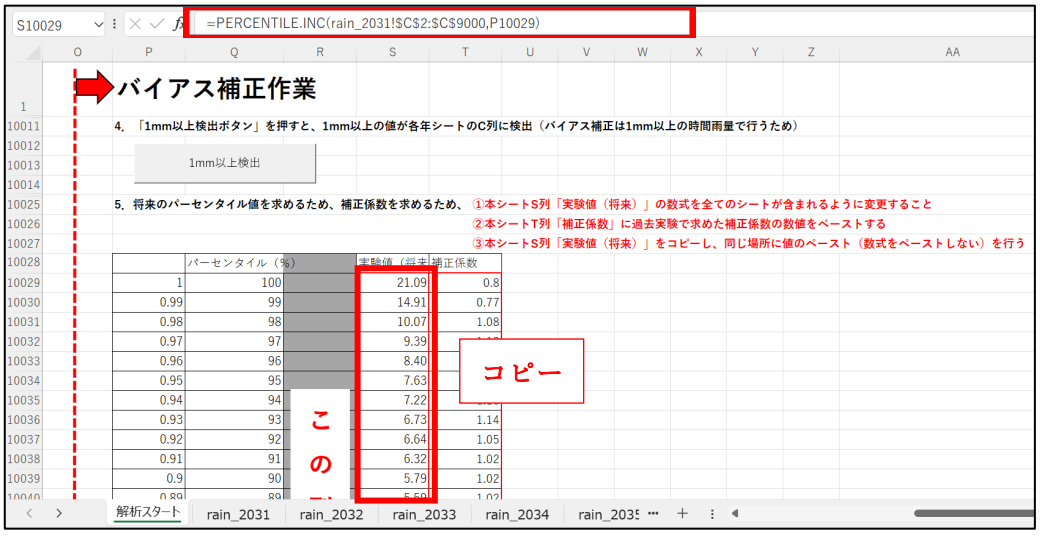
（解析スタートシート）



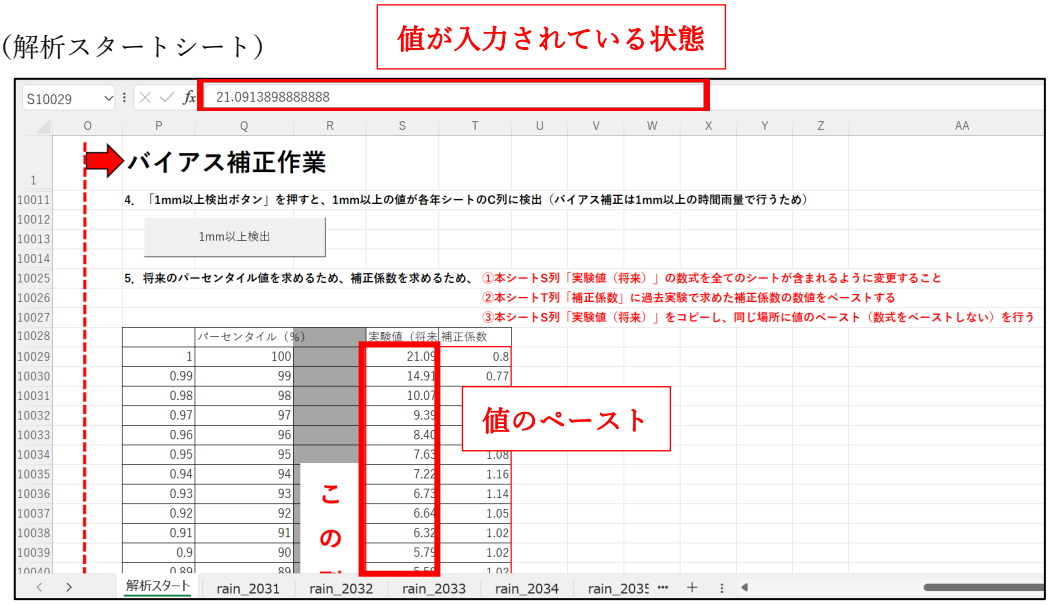

（３）　数値の貼り付け

本シート S 列「実験値（将来)」をコピーし、同じ場所に値のペースト（数式をペーストしない）を行う

（解析スタートシート）　**関数が入力されている状態**



（解析スタートシート）



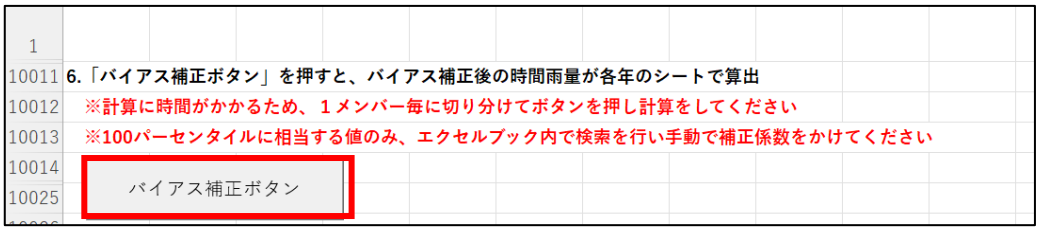

## 3　時間雨量の補正

「解析スタートシート」内の「バイアス補正ボタン」を押すと、各年降雨量シートの J 列にバイアス補正後の時間雨量が算出

※　計算に時間がかかるため、1 メンバ毎に切り分けた計算（ボタンを押す）を推奨

※　100 パーセンタイルに相当する値のみ、エクセルブック内で検索を行い手動で補正係数をかけること（補正方法は P34 を参照）

（解析スタートシート）



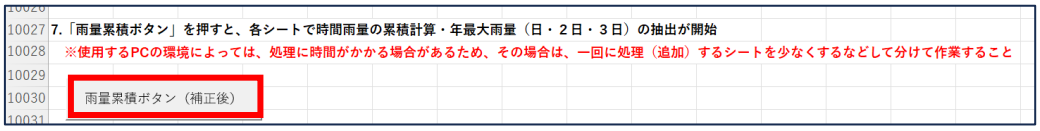
（降雨量シート）


## 4　年最大雨量の算出

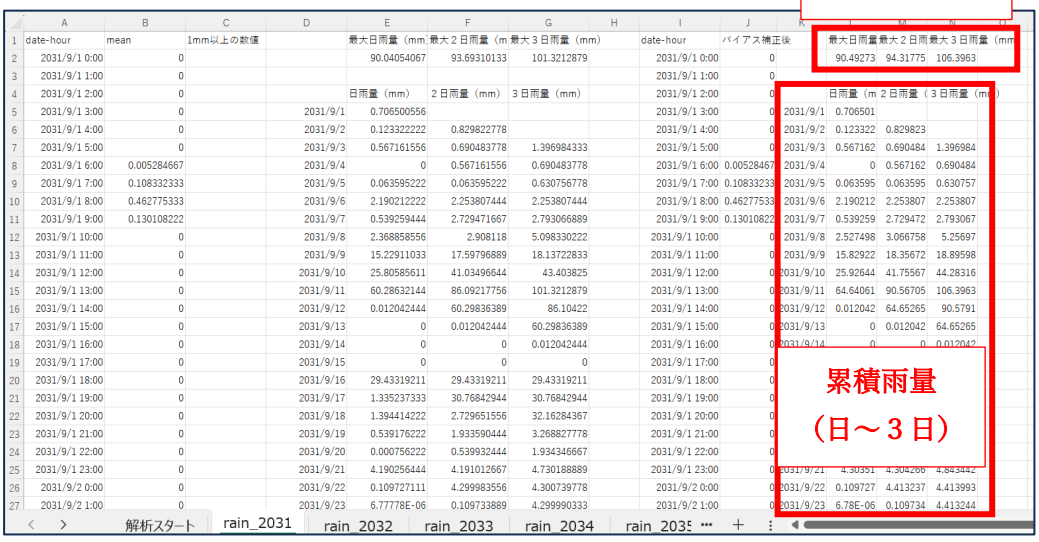
「解析スタートシート」内の「雨量累積ボタン（補正後)」を押すと、各降雨量シートで時間雨量の累積計算（日・２日・３日)・年最大雨量（日・２日・３日）が算出

※　使用する PC の環境によっては、処理に時間がかかる場合があるため、その場合は、一回に処理（追加）するシートを少なくするなどして分けて作業すること

（解析スタートシート）




（降雨量シート）



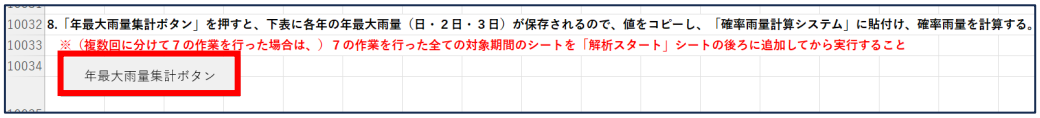
## 5　年最大雨量の整理

「解析スタートシート」内の「年最大雨量集計ボタン」を押すと、下表に各年の年最大雨量（日・２日・３日）が抽出

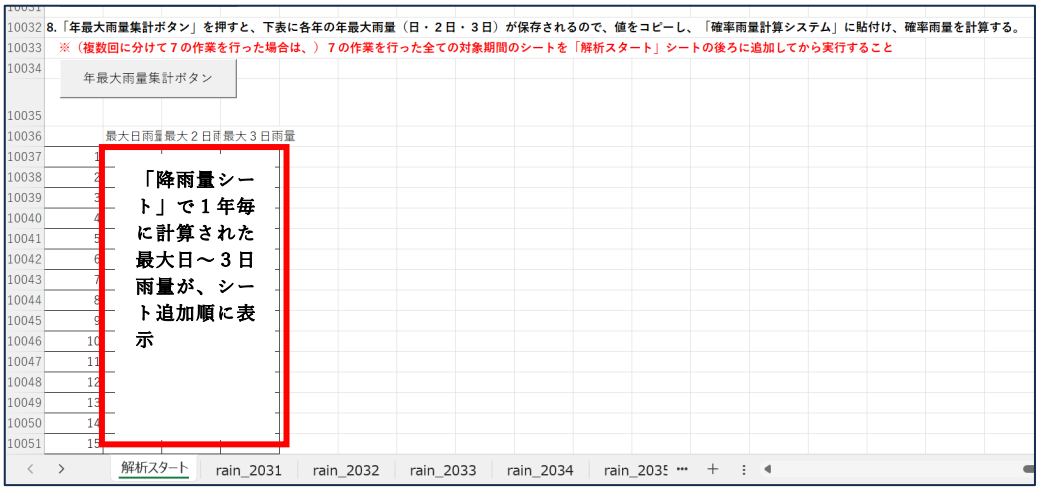
※　ボタンを押した際に実行時エラー表示されるが、そのまま終了ボタンを押すこと

※　複数回に分けて４の作業を行った場合は、４の作業を行った全ての対象期間のシートを「解析スタート」シートの後ろに追加してから実行すること

（解析スタートシート）




（解析スタートシート）




## 6　確率降雨量の算出

５で集計した値を用いて、日～３日連続雨量における 1/10 等の確率降雨量を算出

※　確率降雨量の算出方法は、土地改良事業計画設計基準　計画「排水」技術書「６．実績降雨に基づく計画基準降雨」を参照

# （附属資料１）

# 実験データの入手

気候予測資料（データセット）の d2PDF（５km）データは、本省から地方農政局等を通じて貸与するが、DIAS より入手する場合は、以下の手順で行う。

（DIAS HP　https://diasjp.net/）


（DIAS 「公開データセット一覧」）

（DIAS 「全国 5km メッシュアンサンブル気候予測データの諸元、規約画面」）


（DIAS 「ログイン画面」）

DIAS サービスを利用する際にアカウント登録が必要

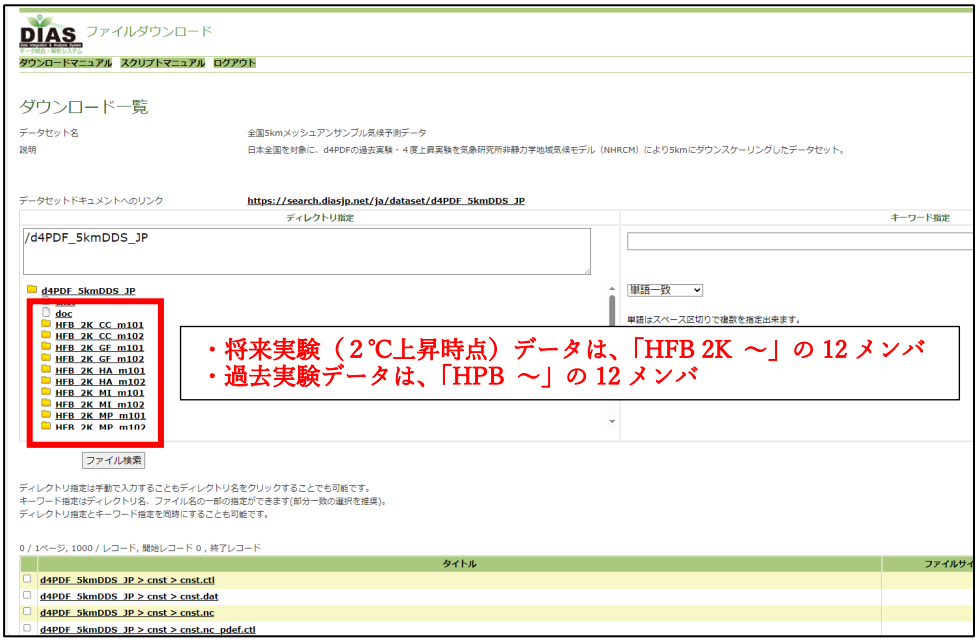
■ダウンロードするデータ

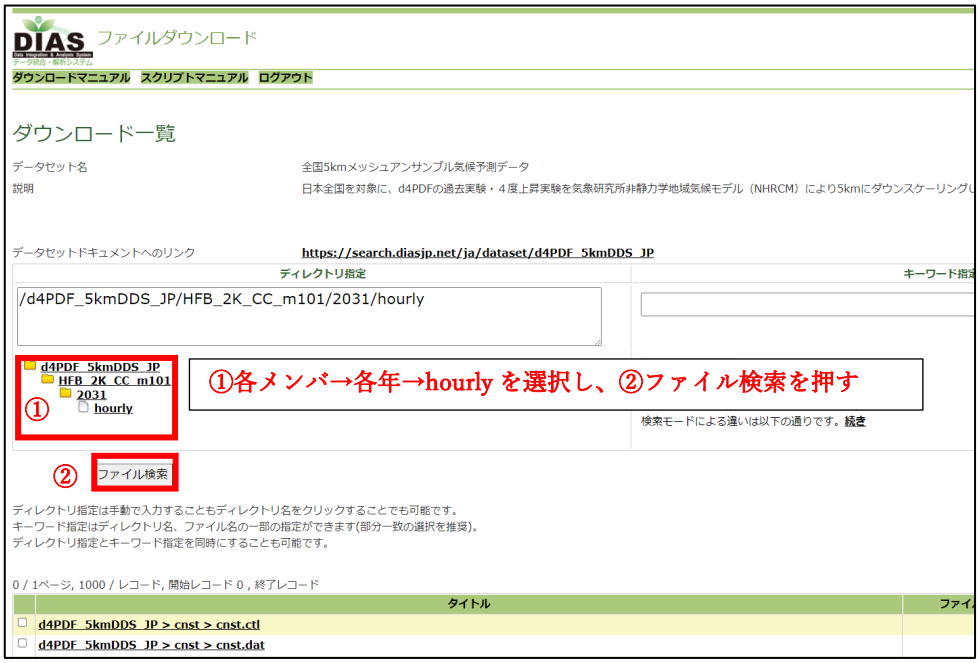
・過去実験：12 メンバ×60 年分（1951 年 9 月 1 日～2011 年 8 月 31 日）の内、観測値で選定した期間の時間降雨量データ

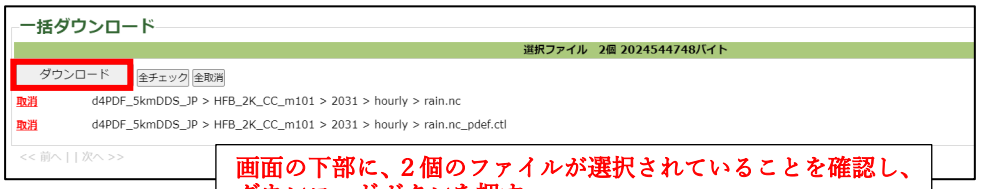
・将来実験（2℃上昇時点）：12 メンバ×60 年分（2031 年 9 月 1 日～2091 年 8 月 31 日）の時間降雨量データ

※　将来実験（2℃上昇時点）では、産業革命以降 2℃上昇時点（2040 年頃）の予測を行っており、データの年月日は便宜上設定されている。

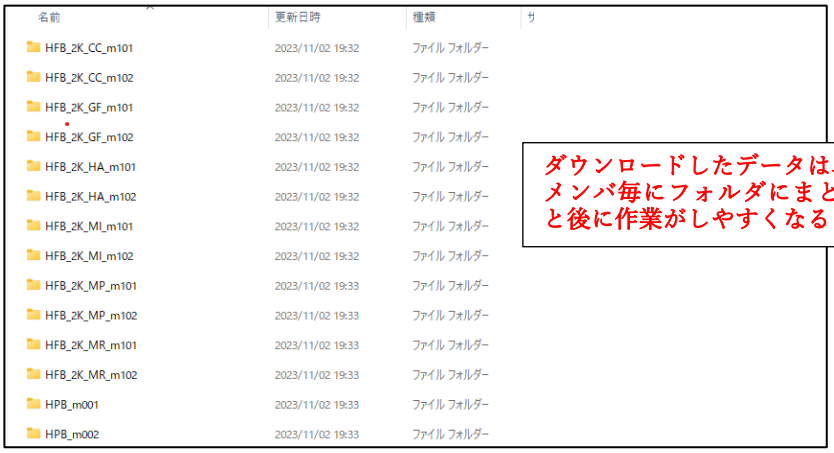
（DIAS 「全国 5km メッシュアンサンブル気候予測データ ダウンロード画面」）




上記を 60 年分×24 メンバ（過去実験 12 メンバ＋将来実験 12 メンバ）を繰り返し行い、データのダウンロードを行う。



# （附属資料２）

# d2PDF データを csv 化する際の

# Python プログラムコード

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
netcdf2csv_arealmean_cnst.py — cnst.nc の flon/flat を唯一の位置情報として用い、
セル定義を選択（voronoi）して shapefile と重なり率を計算し、
対象セルの coverage と降水 CSV（全格子・面積加重平均）を出力する。

主な仕様:
  - 位置情報: cnst.nc の flon/flat を使用。
  - セル定義: --cell-def voronoi|dual|primal（default=voronoi）
      * voronoi: 各格子点をサイトとする Voronoi 多角形（等距離直線の分割）。
                 投影空間(EPSG:3857)で有限化＆クリップ＆有効化し、WGS84 に逆変換。
      * dual   : 各格子点中心の東西南北“中点”四角形（FCT=499 サンプリング近似）。
      * primal : 通常セル（(j,i)-(j,i+1)-(j+1,i+1)-(j+1,i）の四角形）。
  - 抽出: coverage > --min-coverage のセルを採用。
  - 時刻: netCDF4.num2date で cftime 対応。--shift-hours で補正可能。
  - rain 形状: (time, member, j, i）または（time, j, i）に対応（member=0 固定）。
  - 出力: target_coverage.csv は「中心点（i,j)」をセル名で統一（--index-base で 0/1 始まり）。
  - 追加出力: --export-cells-shp で採用セルのポリゴン出力（shp/gpkg/geojson)。

使い方（例）:
  python netcdf2csv_arealmean_cnst.py ../../hourly/HPB_m009/ -o ./csv_cnst ¥
    -s ./murayama_hokubu_wgs84.shp --min-coverage 0 ¥
    --export-cells-shp ./csv_cnst/HPB_m009/selected_cells.shp
"""

import time
import os
import sys
import glob
import argparse
import numpy as np
import pandas as pd
import datetime as dt
from datetime import timedelta
from netCDF4 import Dataset, num2date
```

```
import shapefile  # pyshp
from matplotlib.path import Path

# 依存（ある場合のみ使用）
try:
    from scipy.spatial import Voronoi
    _HAVE_SCIPY = True
except Exception:
    _HAVE_SCIPY = False

try:
    import geopandas as gpd
    from shapely.geometry import Polygon, LineString, Point as ShpPoint, MultiPoint
    from shapely.ops import unary_union, transform
    from shapely.errors import TopologicalError
    try:
        from shapely.validation import make_valid  # shapely>=2
        _HAVE_MAKE_VALID = True
    except Exception:
        _HAVE_MAKE_VALID = False
    _HAVE_GPD = True
except Exception:
    _HAVE_GPD = False
    _HAVE_MAKE_VALID = False

try:
    import pyproj
    from pyproj import Transformer
    _HAVE_PYPROJ = True
except Exception:
    _HAVE_PYPROJ = False

from functools import partial

# ------------------------------------------------------------
# ユーティリティ
# ------------------------------------------------------------

def ensure_2d(a, name):
    a = np.asarray(a)
    orig = a.shape
    a = np.squeeze(a)
    while a.ndim > 2:
        a = a[0]
    if a.ndim != 2:
        raise ValueError(f"{name} must be 2D, got {orig} -> {a.shape}")
```

```
    return a


def find_rain_files(input_dir: str):
    return sorted(glob.glob(os.path.join(input_dir, "**", "rain.nc"), recursive=True))


def find_cnst_for(rain_nc_path: str, max_up: int = 2) -> str:
    d = os.path.dirname(rain_nc_path)
    for _ in range(max_up + 1):   # 0=同階層, 1=1 つ上, ...
        cand = os.path.join(d, "cnst.nc")
        if os.path.exists(cand):
            return cand
        parent = os.path.dirname(d)
        if parent == d:
            break
        d = parent
    raise FileNotFoundError(f"cnst.nc not found within {max_up} levels up from: {rain_nc_path}")


def load_lonlat_from_cnst(cnst_path: str):
    ds = Dataset(cnst_path, "r")
    lon_candidates = ["flon", "lon", "g0_lon", "xx", "x", "longitude"]
    lat_candidates = ["flat", "lat", "g0_lat", "yy", "y", "latitude"]
    lon_name = next((v for v in lon_candidates if v in ds.variables), None)
    lat_name = next((v for v in lat_candidates if v in ds.variables), None)
    if lon_name is None or lat_name is None:
        keys = list(ds.variables.keys())
        ds.close()
        raise RuntimeError(f"cnst.nc に lon/lat 変数が見つかりません: {keys}")
    lon2d = ensure_2d(ds.variables[lon_name][:], lon_name)
    lat2d = ensure_2d(ds.variables[lat_name][:], lat_name)
    ds.close()
    if lon2d.shape != lat2d.shape:
        raise ValueError(f"lon/lat shape mismatch: {lon2d.shape} vs {lat2d.shape}")
    return lon2d, lat2d   # (J, I)


def read_polygon_paths(shp_path: str):
    sf = shapefile.Reader(shp_path, encoding="utf-8", encodingErrors="ignore")
    paths = []
    shapely_polys = []
    for shp in sf.shapes():
        parts = list(shp.parts) + [len(shp.points)]
        for a, b in zip(parts[:-1], parts[1:]):
```

```
                pts = np.array(shp.points[a:b])
                if pts.size == 0:
                    continue
                if np.max(np.abs(pts[:, 0])) > 360 or np.max(np.abs(pts[:, 1])) > 90:
                    raise ValueError("Shapefile の CRS は EPSG:4326 (lon/lat) にしてく
ださい。")
                paths.append(Path(pts))
                if _HAVE_GPD:
                    try:
                        shapely_polys.append(Polygon(pts))
                    except Exception:
                        pass
    if not paths:
        raise ValueError("Shapefile 内に有効なポリゴンが見つかりません。")
    return paths, shapely_polys


def bbox_from_paths(paths):
    lon_min = +1e30; lon_max = -1e30
    lat_min = +1e30; lat_max = -1e30
    for p in paths:
        v = p.vertices
        lon_min = min(lon_min, float(np.min(v[:, 0]))); lon_max = max(lon_max,
float(np.max(v[:, 0])))
        lat_min = min(lat_min, float(np.min(v[:, 1]))); lat_max = max(lat_max,
float(np.max(v[:, 1])))
    return lon_min, lon_max, lat_min, lat_max


def median_grid_step(a2d, axis):
    a2d = np.asarray(a2d)
    if axis == 1 and a2d.shape[1] > 1:
        dif = np.abs(a2d[:, 1:] - a2d[:, :-1]).ravel()
    elif axis == 0 and a2d.shape[0] > 1:
        dif = np.abs(a2d[1:, :] - a2d[:-1, :]).ravel()
    else:
        return 0.05
    dif = dif[~np.isnan(dif)]
    return float(np.median(dif)) if dif.size else 0.05


def nearest_cell_by_point(center_lon2d, center_lat2d, lat, lon):
    d2 = (center_lat2d - lat) ** 2 + (center_lon2d - lon) ** 2
    j, i = np.unravel_index(np.argmin(d2), d2.shape)
    return np.array([[j, i]])
```

```
# ------------------------------------------------------------
# dual coverage（中点四角形 × ポリゴン）
# ------------------------------------------------------------

def coverage_fraction_dual(lon2d, lat2d, sel_jis, paths, FCT=499):
    n = len(sel_jis)
    if n == 0:
        return np.array([], dtype=float)
    if FCT <= 1:
        return np.ones(n, dtype=float)

    J, I = lon2d.shape
    step = 1.0 / FCT
    offs = np.linspace(-0.5 + step / 2, 0.5 - step / 2, FCT)
    uu, vv = np.meshgrid(offs, offs, indexing="xy")
    uv = np.column_stack([uu.ravel(), vv.ravel()])

    fracs = np.empty(n, dtype=float)

    for k, (j, i) in enumerate(sel_jis):
        iW = max(i - 1, 0);    iE = min(i + 1, I - 1)
        jN = max(j - 1, 0);    jS = min(j + 1, J - 1)
        lonE = 0.5 * (lon2d[j, i] + lon2d[j, iE]);    latE = 0.5 * (lat2d[j, i] + lat2d[j, iE])
        lonW = 0.5 * (lon2d[j, iW] + lon2d[j, i]);    latW = 0.5 * (lat2d[j, iW] + lat2d[j, i])
        lonS = 0.5 * (lon2d[jS, i] + lon2d[j, i]);    latS = 0.5 * (lat2d[jS, i] + lat2d[j, i])
        lonN = 0.5 * (lon2d[j, i] + lon2d[jN, i]);    latN = 0.5 * (lat2d[j, i] + lat2d[jN, i])
        C = np.array([[lonE, latE], [lonS, latS], [lonW, latW], [lonN, latN]], dtype=float)
        u = (uv[:, 0] + 0.5); v = (uv[:, 1] + 0.5)
        w0 = (1 - u) * (1 - v); w1 = u * (1 - v); w2 = u * v; w3 = (1 - u) * v
        lon_sub = w0 * C[0, 0] + w1 * C[1, 0] + w2 * C[2, 0] + w3 * C[3, 0]
        lat_sub = w0 * C[0, 1] + w1 * C[1, 1] + w2 * C[2, 1] + w3 * C[3, 1]
        pts = np.column_stack([lon_sub, lat_sub])
        inside = np.zeros(pts.shape[0], dtype=bool)
        for p in paths:
            inside |= p.contains_points(pts)
        fracs[k] = inside.mean()

    return fracs

# ------------------------------------------------------------
# Voronoi（等距離直線の分割セル）ヘルパ
# ------------------------------------------------------------

def _project_xy(lon, lat, lat0=None):
    lon = np.asarray(lon); lat = np.asarray(lat)
    if _HAVE_PYPROJ:
```

```
            tf = Transformer.from_crs("EPSG:4326", "EPSG:3857", always_xy=True)
            x, y = tf.transform(lon, lat)
            return np.asarray(x), np.asarray(y)
        if lat0 is None:
            lat0 = float(np.nanmean(lat))
        R = 6371000.0
        x = np.deg2rad(lon) * R * np.cos(np.deg2rad(lat0))
        y = np.deg2rad(lat) * R
        return x, y


    def _voronoi_finite_polygons_2d(vor, radius=None):
        if vor.points.shape[1] != 2:
            raise ValueError("Requires 2D input")
        new_regions = []
        new_vertices = vor.vertices.tolist()
        center = vor.points.mean(axis=0)
        if radius is None:
            radius = np.ptp(vor.points, axis=0).max()*2
        all_ridges = {}
        for (p1, p2), (v1, v2) in zip(vor.ridge_points, vor.ridge_vertices):
            all_ridges.setdefault(p1, []).append((p2, v1, v2))
            all_ridges.setdefault(p2, []).append((p1, v1, v2))
        for p1, region_idx in enumerate(vor.point_region):
            vertices = vor.regions[region_idx]
            if all(v >= 0 for v in vertices):
                new_regions.append(vertices); continue
            ridges = all_ridges[p1]
            new_region = [v for v in vertices if v >= 0]
            for p2, v1, v2 in ridges:
                if v2 < 0: v1, v2 = v2, v1
                if v1 >= 0 and v2 >= 0: continue
                t = vor.points[p2] - vor.points[p1]
                t /= np.linalg.norm(t)
                n = np.array([-t[1], t[0]])
                midpoint = vor.points[[p1, p2]].mean(axis=0)
                direction = np.sign(np.dot(midpoint - center, n)) * n
                far_point = vor.vertices[v2] + direction * (radius if radius is not None else 1.0)
                new_vertices.append(far_point.tolist())
                new_region.append(len(new_vertices)-1)
            vs = np.array([new_vertices[v] for v in new_region])
            c = vs.mean(axis=0)
            ang = np.arctan2(vs[:,1]-c[1], vs[:,0]-c[0])
            new_region = [v for _, v in sorted(zip(ang, new_region))]
            new_regions.append(new_region)
        return new_regions, np.asarray(new_vertices)
```

```
def build_voronoi_cells(lon2d, lat2d, sel_jis):
    if not _HAVE_SCIPY or not _HAVE_GPD:
        raise RuntimeError("Voronoi requires scipy + shapely/geopandas")
    J, I = lon2d.shape
    lon_flat = lon2d.ravel(); lat_flat = lat2d.ravel()
    x, y = _project_xy(lon_flat, lat_flat)
    vor = Voronoi(np.column_stack([x, y]))
    regions, vertices = _voronoi_finite_polygons_2d(vor)

    # 投影空間で外枠（凸包+余白）を作ってクリップ
    mp_proj = MultiPoint([(xx, yy) for xx, yy in zip(x, y)])
    hull_proj = mp_proj.convex_hull.buffer(20000.0)
    minx, miny, maxx, maxy = hull_proj.bounds
    pad = 50000.0
    bbox_proj = Polygon([(minx-pad, miny-pad), (maxx+pad, miny-pad), (maxx+pad,
maxy+pad), (minx-pad, maxy+pad)])
    try:
        hull_proj = hull_proj.intersection(bbox_proj)
    except TopologicalError:
        hull_proj = hull_proj.buffer(0).intersection(bbox_proj)

    if _HAVE_PYPROJ:
        tf_inv = pyproj.Transformer.from_crs("EPSG:3857", "EPSG:4326",
always_xy=True)
        to_wgs84 = partial(tf_inv.transform)
    else:
        to_wgs84 = None

    polys = [None]*(J*I)
    for p_idx, reg in enumerate(regions):
        ring_xy = vertices[reg]
        poly_proj = Polygon(ring_xy)
        try:
            poly_proj = poly_proj.intersection(hull_proj)
        except TopologicalError:
            poly_proj = poly_proj.buffer(0).intersection(hull_proj)
        if not poly_proj.is_valid:
            poly_proj = make_valid(poly_proj) if _HAVE_MAKE_VALID else
poly_proj.buffer(0)
        poly_ll = transform(to_wgs84, poly_proj) if to_wgs84 is not None else poly_proj
        polys[p_idx] = poly_ll

    out = []
    for j, i in sel_jis:
```

```
            out.append(polys[j*I + i])
        return out

# ------------------------------------------------------------
# セルポリゴン出力（任意）
# ------------------------------------------------------------

def export_selected_cell_polys(out_path, cell_def, lon2d, lat2d, sel_jis, cov, index_base=0):
        if not _HAVE_GPD:
            raise RuntimeError("--export-cells-shp requires geopandas/shapely")
        J, I = lon2d.shape
        geoms = []
        recs = []
        if cell_def == "voronoi":
            polys = build_voronoi_cells(lon2d, lat2d, sel_jis)
            geoms = polys
        elif cell_def == "primal":
            for (j, i) in sel_jis:
                i1 = min(i+1, I-1); j1 = min(j+1, J-1)
                ring = [(lon2d[j, i],     lat2d[j, i]),
                        (lon2d[j, i1],    lat2d[j, i1]),
                        (lon2d[j1, i1], lat2d[j1, i1]),
                        (lon2d[j1, i],    lat2d[j1, i]),
                        (lon2d[j, i],     lat2d[j, i])]
                geoms.append(Polygon(ring))
        else:   # dual
            for (j, i) in sel_jis:
                iW = max(i - 1, 0);     iE = min(i + 1, I - 1)
                jN = max(j - 1, 0);     jS = min(j + 1, J - 1)
                E = (0.5 * (lon2d[j, i] + lon2d[j, iE]), 0.5 * (lat2d[j, i] + lat2d[j, iE]))
                S = (0.5 * (lon2d[jS, i] + lon2d[j, i]), 0.5 * (lat2d[jS, i] + lat2d[j, i]))
                W = (0.5 * (lon2d[j, iW] + lon2d[j, i]), 0.5 * (lat2d[j, iW] + lat2d[j, i]))
                N = (0.5 * (lon2d[j, i] + lon2d[jN, i]), 0.5 * (lat2d[j, i] + lat2d[jN, i]))
                ring = [E, S, W, N, E]
                geoms.append(Polygon(ring))
        for (j, i), fr in zip(sel_jis, cov):
            recs.append({
                "i": int(i + (1 if index_base == 1 else 0)),
                "j": int(j + (1 if index_base == 1 else 0)),
                "fraction_area": float(fr)
            })
        gdf = gpd.GeoDataFrame(recs, geometry=geoms, crs="EPSG:4326")
        ext = os.path.splitext(out_path)[1].lower()
        if ext == ".gpkg":
            gdf.to_file(out_path, layer="cells", driver="GPKG")
        elif ext in (".geojson", ".json"):
```

```
            gdf.to_file(out_path, driver="GeoJSON")
        else:
            gdf.to_file(out_path)

# ------------------------------------------------------------
# メイン処理
# ------------------------------------------------------------

def main():
    ap = argparse.ArgumentParser(
        formatter_class=argparse.RawTextHelpFormatter,
        description=(
            "cnst.nc の flon/flat + 指定セル定義で格子選択し、"
            "rain.nc から時系列 CSV 出力（全格子・面積加重平均)。"
        ),
    )
    ap.add_argument("inputs", nargs="+", help="入力ディレクトリ（再帰で **/rain.nc を探索）")
    ap.add_argument("-o", "--output", required=True, help="出力ディレクトリ")
    ap.add_argument("-s", "--shapefile", default=None, help="対象ポリゴン shapefile (EPSG:4326)。-p と排他")
    ap.add_argument("-p", "--point", default=None, help="単一点 'lat,lon'（例: 36.03,140.09)。-s と排他")
    ap.add_argument("--shift-hours", type=int, default=0, help="読み込んだ日時を指定時間だけシフト（例: -48)")
    ap.add_argument("--index-base", type=int, choices=[0, 1], default=0, help="target_coverage の i,j を 0/1 始まりで出力（default=0)")
    ap.add_argument("--cell-def", choices=["voronoi", "dual", "primal"], default="voronoi", help="セル定義を選択")
    ap.add_argument("--min-coverage", type=float, default=1e-12, help="採用する最小被覆率（default=1e-12). 0 で『少しでも重なれば』")
    ap.add_argument("--export-cells-shp", default=None, help="採用セルの範囲ポリゴンを出力（shp/gpkg/geojson)")

    args = ap.parse_args()

    if (args.shapefile is None) == (args.point is None):
        print("[ERROR] --shapefile か --point のどちらか一方を指定してください。", file=sys.stderr)
        sys.exit(1)

    input_dirs = sorted(set(args.inputs))
    os.makedirs(args.output, exist_ok=True)

    for in_dir in input_dirs:
        dname = os.path.basename(os.path.normpath(in_dir)) or "out"
```

```
        out_dir = os.path.join(args.output, dname)
        os.makedirs(out_dir, exist_ok=True)
        print(f"¥ninput: {in_dir}¥n   -> output: {out_dir}", flush=True)

        rain_paths = find_rain_files(in_dir)
        if not rain_paths:
            print("  (skip) rain.nc が見つかりません。", flush=True)
            continue

        # 初回: cnst.nc から lon/lat を取得
        try:
            cnst_path = find_cnst_for(rain_paths[0], max_up=2)
        except Exception as e:
            print(f"[ERROR] {e}", file=sys.stderr)
            continue
        lon2d, lat2d = load_lonlat_from_cnst(cnst_path)   # (J,I)
        J, I = lon2d.shape

        # ------------------------ 対象セル抽出 ------------------------
        if args.shapefile:
            print(f"   shapefile: {args.shapefile}")
            shp_paths, shapely_polys = read_polygon_paths(args.shapefile)

            # bbox で広めに候補抽出
            lon_min, lon_max, lat_min, lat_max = bbox_from_paths(shp_paths)
            lon_pad = max(median_grid_step(lon2d, axis=1), 1e-3)
            lat_pad = max(median_grid_step(lat2d, axis=0), 1e-3)
            lon_min -= lon_pad; lon_max += lon_pad
            lat_min -= lat_pad; lat_max += lat_pad

            bbox_mask = ((lon2d >= lon_min) & (lon2d <= lon_max) &
                         (lat2d >= lat_min) & (lat2d <= lat_max))
            jj, ii = np.where(bbox_mask)
            sel_candidates = np.column_stack([jj, ii])   # (m,2) (j,i)

            if sel_candidates.size == 0:
                print("  (skip) bbox 内候補セルが見つかりませんでした。", flush=True)
                continue

            # coverage 計算
            if args.cell_def == "dual":
                cov_all = coverage_fraction_dual(lon2d, lat2d, sel_candidates, shp_paths,
FCT=499)
            else:
                if not _HAVE_GPD:
                    raise RuntimeError("cell-def voronoi/primal requires
```

```
shapely/geopandas installed.")
                    cell_polys = []
                    if args.cell_def == "primal":
                        for (j, i) in sel_candidates:
                            i1 = min(i+1, I-1); j1 = min(j+1, J-1)
                            ring = [(lon2d[j, i],     lat2d[j, i]),
                                    (lon2d[j, i1],    lat2d[j, i1]),
                                    (lon2d[j1, i1], lat2d[j1, i1]),
                                    (lon2d[j1, i],    lat2d[j1, i]),
                                    (lon2d[j, i],     lat2d[j, i])]
                            cell_polys.append(Polygon(ring))
                    elif args.cell_def == "voronoi":
                        cell_polys = build_voronoi_cells(lon2d, lat2d, sel_candidates)
                    # 被覆率（厳密面積）
                    shp_u = unary_union(shapely_polys) if shapely_polys else None
                    cov_all = np.zeros(len(sel_candidates), dtype=float)
                    for idx, poly in enumerate(cell_polys):
                        try:
                            p = make_valid(poly) if (_HAVE_MAKE_VALID and not
poly.is_valid) else (poly if poly.is_valid else poly.buffer(0))
                            if shp_u is not None:
                                inter = p.intersection(shp_u)
                                a = inter.area; A = p.area
                                cov_all[idx] = (a / A) if A > 0 else 0.0
                            else:
                                # フォールバック: 中心点内外
                                j,i = sel_candidates[idx]
                                inside = False
                                for pp in shp_paths:
                                    if pp.contains_point((lon2d[j,i], lat2d[j,i])):
                                        inside = True; break
                                cov_all[idx] = 1.0 if inside else 0.0
                        except Exception:
                            cov_all[idx] = 0.0

                mask = cov_all > float(args.min_coverage)
                sel_jis = sel_candidates[mask]
                cov = cov_all[mask]

                if sel_jis.size == 0:
                    print("  (skip) coverage>min_coverage のセルがありませんでした。",
flush=True)
                    continue
            else:
                # 単一点: 最寄りセル 1 点、被覆率は 1.0
                lat_str, lon_str = args.point.split(",")
```

```
            sel_jis = nearest_cell_by_point(lon2d, lat2d, float(lat_str), float(lon_str))
            cov = np.ones(len(sel_jis), dtype=float)

        # coverage CSV（中心点の（i,j）をセル名として出力）
        i_out = sel_jis[:, 1] + (1 if args.index_base == 1 else 0)
        j_out = sel_jis[:, 0] + (1 if args.index_base == 1 else 0)
        cov_df = pd.DataFrame({"i": i_out, "j": j_out, "fraction_area": cov})
        cov_df.to_csv(os.path.join(out_dir, "target_coverage.csv"), index=False)
        print((" [INFO] target_coverage rows: %d (index_base=%d, min_coverage=%g,
cell_def=%s)"
               % (sel_jis.shape[0], args.index_base, args.min_coverage, args.cell_def)))
        print(f" [INFO] mean fraction: {cov.mean():.6f}, min/max:
{cov.min():.6f}/{cov.max():.6f}")

        # セル範囲のポリゴン出力（任意）
        if args.export_cells_shp:
            export_selected_cell_polys(args.export_cells_shp, args.cell_def, lon2d, lat2d,
sel_jis, cov, index_base=args.index_base)
            print(f" [INFO] exported selected cell polygons -> {args.export_cells_shp}")



        # break



        # ------------------------ rain 読み出し & 出力 ------------------------
        jmin, imin = sel_jis.min(axis=0)
        jmax, imax = sel_jis.max(axis=0)
        sel_local = sel_jis.copy(); sel_local[:, 0] -= jmin; sel_local[:, 1] -= imin
        col_names = [f"{i}_{j}" for (j, i) in sel_jis]

        all_dates = []
        all_rows = []
        w_sum_all = []
        w_tot_all = []

        for k, f in enumerate(rain_paths, 1):
            print(f" [{k:3d}/{len(rain_paths)}] {f}", flush=True)
            ds = Dataset(f, "r")

            # 時刻
            tvar = ds.variables["time"]
            cal = getattr(tvar, "calendar", "standard")
```

```
            t = num2date(tvar[:], units=tvar.units, calendar=cal)
            if args.shift_hours:
                td = timedelta(hours=args.shift_hours)
                t = [tt - td for tt in t]
            t_str = [pd.Timestamp(str(tt)).strftime("%Y-%m-%d %H:%M:%S") for tt in
t]

            # rain 切り出し
            r = ds.variables["rain"]
            if r.ndim == 4:
                sub = r[:, 0, jmin:jmax + 1, imin:imax + 1]   # member=0
            elif r.ndim == 3:
                sub = r[:, jmin:jmax + 1, imin:imax + 1]
            else:
                ds.close()
                raise ValueError(f"想定外の rain 次元: {r.shape}")
            year = int(t_str[0][0:4])
            start_term = dt.datetime(year, 9, 1, 0)
            end_term = dt.datetime(year + 1, 8, 31, 23)

            for ti, ts in enumerate(t_str):
                ts_dt = pd.to_datetime(ts)
                if  (start_term <= ts_dt <= end_term):
                    vals = [float(sub[ti, jloc, iloc]) for (jloc, iloc) in sel_local]
                    all_rows.append(vals)

                    wsum = float(np.dot(vals, cov))
                    wtot = float(cov.sum())
                    w_sum_all.append(wsum)
                    w_tot_all.append(wtot)

                    all_dates.append(ts)

            ds.close()

        if not all_rows:
            print("  (skip) 有効な時刻がありませんでした。", flush=True)
            continue

        df_all = pd.DataFrame(all_rows, columns=col_names)
        df_all.insert(0, "date-hour", all_dates)
        df_all.set_index("date-hour", inplace=True)
        df_all.index = pd.to_datetime(df_all.index)

        arr_mean = np.array(w_sum_all) / np.array(w_tot_all)
        df_mean = pd.DataFrame({"rainfall": np.round(arr_mean, 6)}, index=df_all.index)
```

```
        for yr in sorted(df_all.index.year.unique()):
            start = dt.datetime(yr, 9, 1, 0)
            end = dt.datetime(yr + 1, 8, 31, 23)
            d_all = df_all[(df_all.index >= start) & (df_all.index <= end)]
            d_mean = df_mean[(df_mean.index >= start) & (df_mean.index <= end)]
            if not d_all.empty:
                d_all.to_csv(os.path.join(out_dir, f"rain_allgrids_{yr:04d}.csv"))
            if not d_mean.empty:
                d_mean.to_csv(os.path.join(out_dir, f"rain_arealmean_{yr:04d}.csv"))

        print("   -> done.", flush=True)


if __name__ == "__main__":
    try:
        t0 = time.time()
        main()
        t1 = time.time()
        print(f"¥nall done. elapsed time: {(t1 - t0)/60:.1f} min", flush=True)
    except KeyboardInterrupt:
        print("Interrupted.", file=sys.stderr)
        sys.exit(130)
```