農研機構　メッシュ農業気象データシステム開発チーム

# メッシュ農業気象データ再現キット(Version 2.0)利用説明書

## はじめに

「メッシュ農業気象データ再現キット」は、メッシュ農業気象データシステムが過去に作成してインターネット経由で配信したデータを、日を指定して USB ハードディスク上に再現するものです。これを利用すると、メッシュ農業気象データを使用するプログラムが過去の特定の日にどのような結果を出したかを再現できるので、プログラムやアルゴリズムの実効性を検証することができます。

過去の気象データを再現するにあたり、このキットは三種類の中から再現方法を選択することができます。第一の再現方法では、再現指定日当時にメッシュ農業気象データシステムが配信したデータをそのまま再現します（ｆモード)。第二の方法では、再現指定日の前日までは確定値(観測値)が与えられ指定日以降には平年値が与えられたデータを再現します（ｎモード)。そして、第三の方法では、再現指定日に関わらず、このキットを提供する時点における最新のデータを再現します(ｏモード)。これにより、多彩な検証をすることができます。

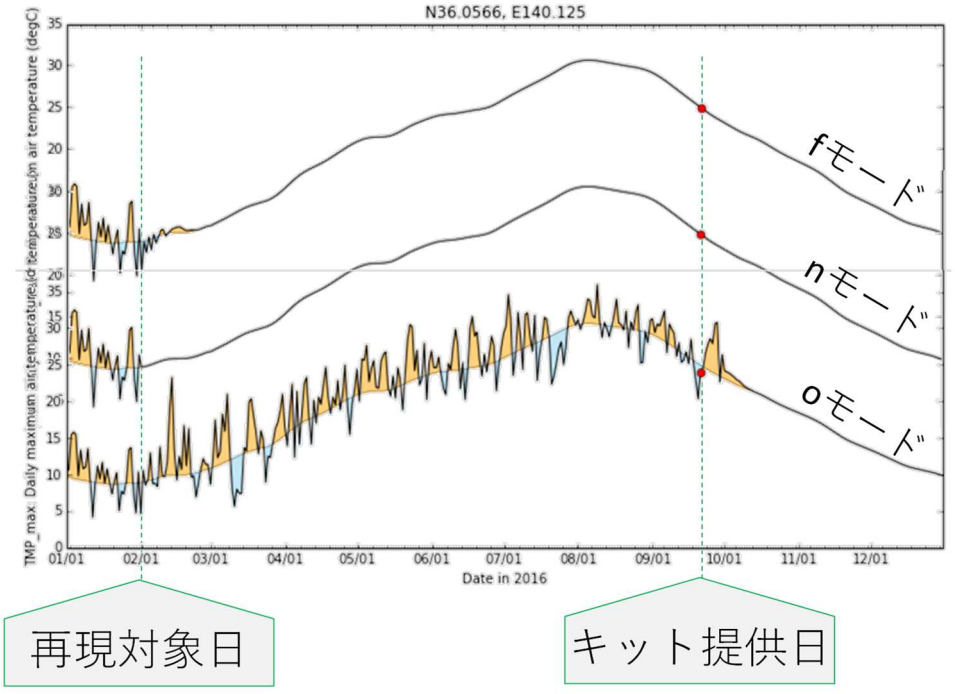


図１．2016 年 9 月 16 日に提供されたキットを用いて、2016 年 2 月 1 日の日平均気温データを再現しグラフ化したもの。ｆモード：再現対象日にメッシュ農業気象データシステムが配信したデータがそのまま再現される。ｎモード：再現対象日の前日までが確定値(観測値)、再現対象日以降に平年値が与えられたデータが再現される。ｏモード：キット提供時点においてメッシュ農業気象データシステムがもつ最新のデータが再現される。

### 利用に必要な環境

USB ハードディスクが接続でき、プログラミング言語 Python バージョン3が実行できる計算機が必要です。なお、本利用説明書は、Anaconda を使用して Python がインストールされていることを前提として記載しています。

### 利用の準備

まず、再現データを使用する計算機の USB ソケットにキットを接続し、認識された HDD のドライブレターをメモします(ここでは仮に X:¥とします)。

### データの再現

○Windows のスタートメニューから[Anaconda3]→[Spyder]と進んで Spyder を起動します。

○ツールバーの左側にある黒いフォルダのアイコンをクリックして USB ハードディスクを開き、X:¥AMGSDS¥programs その中から backto.py を選択して、エディタペインに開きます。

○プログラムのスクリプトが表示されたら、ツールバーの緑三角アイコンを押して実行します。

■すると：このプログラムはコマンドライン引数が必要なので、それを注意する表示が出て終了してしまいます。これでかまいません。

> この作業は、Spyder の作業ディレクトリを手っ取り早く X:¥AMGSDS¥programs に変更する目的で行うものです。Spyder の作業ディレクトリを変更する正規の方法は、IPython コンソールのプロンプトに「cd X:¥AMGSDS¥programs」と入力します。なお、作業ディレクトリが現時点でどこに設定されているかを知るには、「ls」と入力します。入力の最後にエンターキーを押すのを忘れないでください。

○次に、Spyder の右下、IPython コンソールに表示されているプロンプト「In[#]:」に続けて「run backto yyyymmdd -m」と入力し、エンターキーを押します。ここで、yyyymmdd は日付で、m は再現モードです。例えば、2016年10月1日のデータを f モードで再現する場合には、IPython コンソールのプロンプトに以下のように入力します。

```
In[#]: run backto 20161001 -f ＜エンター＞
```

■すると：その下に以下のような表示が現れます。

```
In[#]: run backto 20161001 -f
       forecast mode
=> AMD_Area3_TMP_mea_20161001.tar.gz
```

### データの確認

プログラム「chk.py」を使用すると、再現されたデータをグラフで確認することができます。先ほどと同様にフォルダアイコンをクリックして chk.py を選択し、開きます。ただし、このプログラムはそのままでは実行できません。実行する前に、プログラムの

14行目から16行目をご自分の再現環境に合うように修正する必要があります(図２)。

○14行目には、再現するメッシュ領域を指定します。データ再現キットを依頼するときに指定した領域の記号(Area1～Area6のいずれか)に修正してください。

○15行目には、再現を確認する年次を指定します。先ほどの backto.py プログラムの後ろに書いた日付の年の部分がそれにあたるので、その数字に修正してください。

○16行目には再現を確認する気象要素を指定します。あらかじめ日平均気温(TMP_mea)が設定されていますが、他の要素を確認したい場合はそれに変更してください。変更が終了したら、変更を保存(Ctrl+[S])して、実行ボタンをクリックして実行します。

```
11 import matplotlib.dates as mdates
12 import AMD_Tools3 as AMD
13
14 area = "AreaX"       # 再現する領域
15 yyyy = "2017"        # 再現する期間内の任意の年
16 elem = "TMP_mea"     # 再現する気象要素の中のどれか
17 assert area != "AreaX", "16-18行目を再現環境に合わせて設定してください。"
18
```

図２．プログラム chk.py の一部。再現データの領域、グラフ化する年次、気象要素を 14 行目から 16 行目で指定する。

■すると：再現されたデータに基づく折れ線グラフが IPython コンソールに描画されます(図3)。

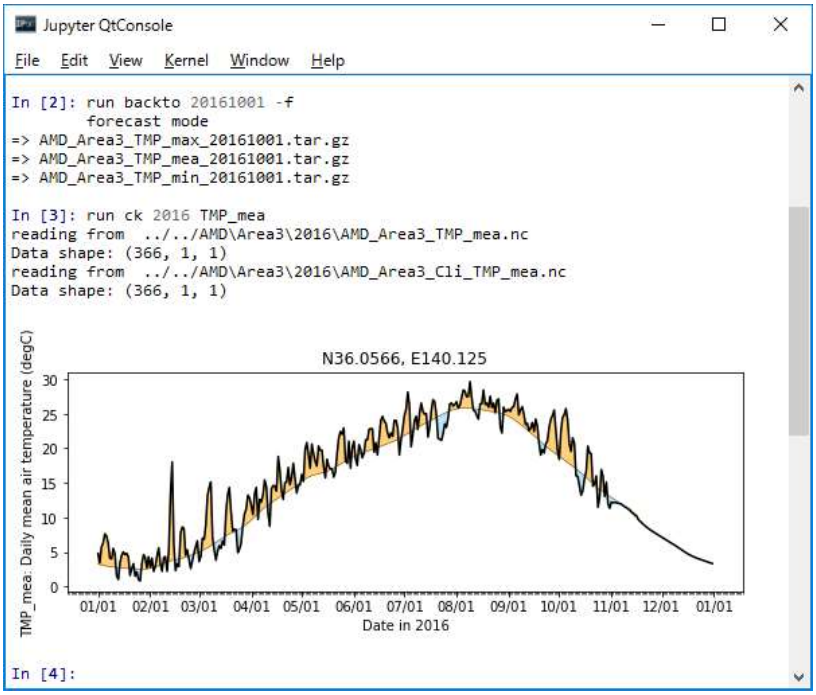


図３．Spyder の右下部分に配置される IPython コンソールのスナップショット。プログラム chk.py を実行すると、再現されたメッシュ気象データに基づくグラフが描画される。

期待通りの気象データが再現されているか確認してください。

次に、別な日のメッシュデータを再現してみます。先ほどと同じように、「run backto 20161015 -f」などと入力するのですが、この入力を楽にすることができます。

○マウスカーソルを IPython コンソールの領域に移動して、一回クリックします。
○上矢印キー([↑])を2回押します。

■すると：さっき入力した「run backto 20161001 -f」がプロンプトに表示されます。

○この表示を適当に修正して、最後にエンターキーを押します。

これで、再現が開始されます。再現処理の終了は、IPython コンソールに新しいプロンプト「In[#]:」が表示されることでわかります。グラフで確認したい場合は、先ほどのように上矢印キーを2回押して今度はそのままエンターキーを押します。これで、chk.py が実行されます。

**再現データの利用**

再現したメッシュデータを利用するプログラムの実行手順は、プログラム chk.py の実行手順と同じです。実行したいプログラムのファイルをデータ再現キットの X:¥AMGSDS¥programs に配置した後、Spyder のフォルダアイコンから読み込み、実行ボタンをクリックします。

ただし、再現データで実行するプログラムは、ほとんどの場合、メッシュ農業気象データ配信サーバーから気象データを取得して動作するように作成されているので、以下の要領でプログラムを修正して入手先をデータ再現キットに切り替える必要があります。

○再現データを利用するプログラムの中で、GetMetData 関数ならびに GetGeoData 関数が使用されている場所を確認します。
○これらの関数の括弧の中に、以下の二つのオプション引数を追加します。

```
area='AreaX'   ただし、X は利用対象地域のエリア番号(1～6)
url='../AMD'
```

たとえば、キットが東北地方のメッシュデータを再現するものであるときは、以下のように修正します。

```
Ta,tim,lat,lon = AMD.GetMetData('TMP_mea',kikan,hanni)
```

この右辺を次のように修正します。

```
Ta,tim,lat,lon = AMD.GetMetData('TMP_mea',kikan,hanni
                                ,area='Area2',url='../AMD')
```

○すべての GetMetData 関数、GetGeoData 関数を修正したら保存して実行ボタンをクリックして実行します。

## メッシュデータ再現と処理の連続実行（上級者向け）

### はじめに

農業情報が最終的な確定値となるまでに、気象情報とともにどのようは経過を辿ったかを検証するようなケースでは、メッシュデータの再現とそれに基づく処理が何度も繰り返されることになります。そこで、メッシュ農業気象データ再現キットには、これら二つの作業をあらかじめ決めた日付範囲について連続して実行するツール `grugru.py` が用意されています。

たとえば、特定年の1年分の気象値の折れ線グラフを画像ファイルとして出力するプログラムに対してこのツールを利用すると、2017年1月1日、11日、21日、31日時点の4枚の日平均気温の図を一度に得ることができます(図４)。

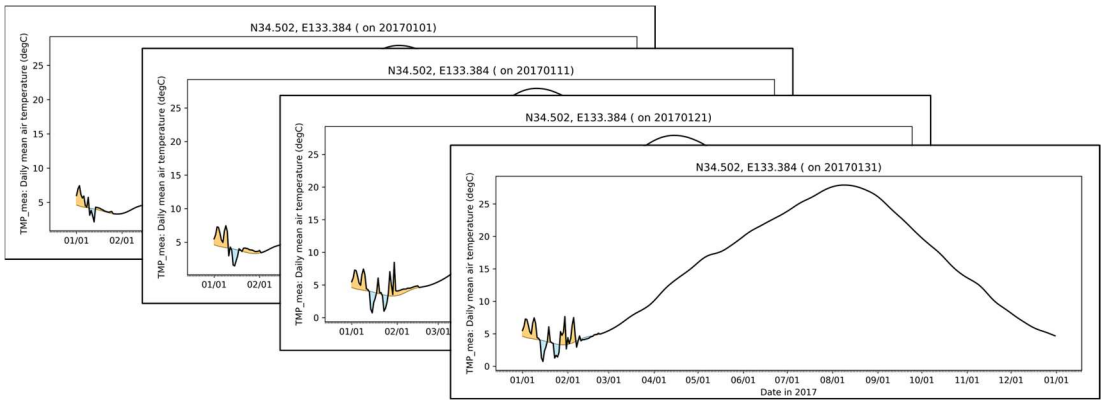


図４．ツール grugru.py を用いて、プログラム chk-grugru.py を 4 回実行して作成されたグラフの画像。

### 連続処理ツール grugru.py の設定

連続処理ツール grugru.py は､他のプログラムを実行させる Python プログラムで、データ再現プログラム backto.py と再現したデータを処理するプログラム(以降「ゲストプログラム」と呼びます)を交互に実行させるように作られています(図５)。利用者は、メッシュデータの再現モード、再現期間、再現間隔、ゲストプログラムのファイル名を設定して使用してください。

○27行目の変数 Mode に、再現モードの記号を文字列で設定してください。
○30行目の変数 Period に、再現日の開始日と終了日を文字列のリストで指定してください。
○33行目の変数 Step に、再現日の間隔を整数で指定してください。
○36行目の変数 GuestProg に､ゲストプログラムのファイル名を文字列で指定してください。

```
20
21      2018.02.07, OHNO, Hiroyuki
22  """
23  import subprocess as sp
24  import AMD_Tools3 as AMD
25  #-----------------------------------------------------
26  #データの再現モード
27  Mode  = "f"
28
29  #期間の開始日と終了日 リストで与えます
30  Period = ["2017-01-01", "2017-01-31"]
31
32  #日数間隔 期間のすべての日を繰り返すときは「1」､10日おきなら「10」。
33  Step = 10
34
35  # ゲストプログラムのファイル名
36  GuestProg = "chk-grugru.py"
37
38  #-----------------------------------------------------
39  tim = AMD.timrange(Period[0],Period[1])
40  yyyymmdds = [oo.strftime("%Y%m%d") for oo in tim]
41  yyyymmdds = yyyymmdds[0::Step]
42  for (i,yyyymmdd) in enumerate(yyyymmdds):
43      igrugru = str(i + 1)
44      print(igrugru, yyyymmdd)
45      proc = sp.run(["python", "backto.py", yyyymmdd,"-"+Mode], stdout = sp.PIPE).stdout
46      print(proc.decode('sjis'))
47
48      proc = sp.run(["python", GuestProg,igrugru,yyyymmdd], stdout = sp.PIPE).stdout
49      print(proc.decode('sjis'))
50  #--
```

図５． プログラム grugru.py のスクリプトの主要部分。

### ゲストプログラムの設定

ゲストプログラムは、二つの要件を満たしている必要があります。第一の要件は、メッシュデータをデータ再現キットから取得するようになっていることです。第二の要件は、コマンドライン引数を二つ受け取るようになっていることです。ツール grugru.py は、繰り返しの連番と yyyymmdd 形式の再現日をこの順にゲストプログラムに渡します(これらを'ぐるぐる文字変数'と呼びます')。

ゲストプログラムがコマンドライン引数を取り込むようにするには、グログラム冒頭に「import sys」を追加して、モジュール sys をインポートしたうえで、さらに「argvs = sys.argv」という文を追加します(図５)。すると、リスト argvs には、grugru.py から渡された連番と日付が第２要素(argvs[1])と第３要素(argvs[2])にそれぞれ代入されるようになるので、これらを取り出して'ぐるぐる文字変数'として使用します。'ぐるぐる文字変数'の使い方については、サンプルプログラム chk-grugru.py のスクリプトを参照してください。

```
 8
 9  実行前に、14行目と15行目をお使いの再現環境に応じて設定してください。
10
11 #   OHNO, Hiroyuki 2018.02.19
12 """
13 import sys # コマンドライン引数を読み込むにはこのモジュールが必要です
14 import datetime
15 import matplotlib.pyplot as plt
16 import matplotlib.dates as mdates
17 import AMD_Tools3 as AMD
18
19 area = "AreaX"       # 再現する領域
20 elem = "TMP_mea"     # 再現する気象要素の中のどれか
21 assert area != "AreaX", "14-15行目を再現環境に合わせて設定してください。"
22
23 argvs = sys.argv     # コマンドライン引数はこのようにしてプログラムに取り込みます
24 igrugru = argvs[1]   # grugru.pyから渡された連番
25 yyyymmdd = argvs[2] # grugru.pyから渡された年月日文字列
26
```

図６．プログラム chk-grugru.py のスクリプトの一部。sys モジュールをインポート(13 行目)したあと、変数 argvs で引数を受け取る(23 行目)。引数は文字列のリストとして代入されるので、それを分解して' ぐるぐる文字変数' を作る(24-25 行目)。