



P MAT

Precipitable-water Model Analysis Tool Official Manual

Spencer Riley

Jun 12, 2022

CONTENTS:

1	Getting Started	3
1.1	Introduction	3
1.2	Installation and Deployment Tutorial	4
1.2.1	Github	4
1.2.2	Amazon Web Service (AWS)	5
1.2.3	Google Cloud Console (GCloud)	5
1.2.4	Local	5
1.2.5	Development	5
2	Working with data and PMAT	7
2.1	Input Data Formatting	7
2.1.1	Configuration Input	7
2.1.2	Raw Data File	12
2.2	Output Data Formatting	14
2.2.1	General data files	14
2.2.2	Machine learning	14
2.2.3	Analytic results	15
3	Reporting Issues with PMAT	17
3.1	Bug reporting	17
3.1.1	Opening a Ticket	17

3.2	Feature requests	17
4	Templates	19
4.1	Input Data File	19
4.1.1	_pmat.yml	19
5	PMAT Reference	21
5.1	pmat_analysis.r	21
5.2	pmat_processing.r	24
5.3	pmat_products.r	28
5.4	pmat_run.r	34
5.5	pmat_utility.r	34

Precipitable-water Model Analysis Tool Official Manual

The Precipitable-Water Model Analysis Tool is an open-source suite for analyzing the relationship between atmospheric brightness temperature and precipitable water.

Warning: This documentation is under active development.

Precipitable-water Model Analysis Tool Official Manual

GETTING STARTED

1.1 Introduction

The Precipitable-water Model Analysis Tool (PMAT) is a computational utility that is used to analyze the data collected from this project to understand the relationship between the zenith sky temperature and precipitable water in the atmosphere. PMAT has three different modules that work together to present data.

The first is the Deployment Module. This module acts as the user interface for the software suite, whether it be locally or through cloud services.

The second is the Pre-processing Module, this module imports data from University of Utah's MesoWest and the University of Wyoming UpperAir Databases.

The third module is the main program to run the analysis, the DAnalysis Module. Here the all of the data is presented and the regression analysis between precipitable water and zenith sky temperature is conducted.

1.2 Installation and Deployment Tutorial

We also require two data files. One that contains the raw data collected by the temperature sensors, that also includes date and time information (`cool_data.csv`). The second should contain sensor definitions with additional parameters for the preprocessing and analysis phases (`_pmat.yml`). A template and detailed breakdown of the configuration file is provided in Chapter 2.1, followed by a detailed breakdown on the data file format.

1.2.1 Github

This version of the Deployment module is, for the most part, automated and recommended. Follow the steps in this section to successfully deploy PMAT through GitHub with GitHub Actions.

1. Create a GitHub repository from the [template repository](https://template.pmat.app)¹.
2. Edit the README.md page based on your location and username
3. Update all files that are contained in the `data/` directory, and utilize the documentation on data formatting that is provided
4. Upon finalizing updates on `cool_data.csv`, the workflow will run automatically and the visual and data products will be generated

¹ <https://template.pmat.app>

1.2.2 Amazon Web Service (AWS)

For Amazon Web Services, PMAT can be configured through the EC2 virtual machines. Once they have been configured, connect to the virtual machine. Once connected, enter the following commands

```
sudo yum update -y
```

```
sudo amazon-linux-extras install docker
```

```
sudo docker pull ghcr.io/physicsgoddess1972/  
↪pmat:latest
```

From here, data files can be added and utilizing the `local_deploy.sh` script, PMAT can be executed.

1.2.3 Google Cloud Console (GCloud)

1.2.4 Local

We fully support Ubuntu and Debian systems. We do have minimal Windows support through the usage of Windows Subsystem for Linux (WSL) and Virtualization.

1.2.5 Development

WORKING WITH DATA AND PMAT

This chapter will discuss the data components associated with PMAT. The first section will detail the formatting guidelines of the input files, followed by another discussion regarding the various output files generated by the software suite.

2.1 Input Data Formatting

The two input files discussed in this section include a YAML configuration file and a Comma Separated Value data file.

2.1.1 Configuration Input

The role of the configuration file is to store a series of parameters that includes sensor information, analytic parameters, logging options, and the site identifiers for the University of Wyoming's Upper-Air database and the University of Utah's MesoWest database.

See also:

Chapter 5.1.1 shows the structure of the data fields. The file-name of this file must be `_pmat.yml`.

Sensor fields

sensor.name (*string*):

Detail

The name of the sensor.

Note

If there are multiple of the same sensor use the notation `_N` with `N` being the index of the sensor.

Example

Sensor 09, Sensor 10_1, Sensor 10_2

sensor.error (*float*):

OPTIONAL

Detail

The manufacturer reported error on the sensor

Example

2.5, 5.0

sensor.color (*string*):

Detail

A hexadecimal color code that will be used to identify the sensor on visualizations.

Example

FF0000, 0000FF

sensor.ratio (*string*):

OPTIONAL

Detail

The distance to spot ratio that is reported by the manufacturer.

Example

12 to 1, 21 to 1

sensor.emissivity (*float*):

OPTIONAL

Detail

The emissivity of the sensor as reported by the manufacturer.

Example

0.95

sensor.poster (*boolean*):

Detail

A boolean that will decide whether the sensor will be shown in the poster-specific plots

Example

true, false

sensor.active (*boolean*):

Detail

A boolean that will decide whether the sensor will be used in the analysis.

Example

true, false

Analysis

train_fraction (*float*):

Detail

The fraction of data being used to create the training set.

Note

A value between 0 and 1.

Example

0.8

rel_difference (*float*):

Detail

Example

2

iteration.step (*integer*):

Detail

The number of steps the analysis will run

Example

1, 100, 1000

Logging

verbose (*string*):

Detail

An identifier for the level of logging

Example

DEBUG, WARN, ERROR, INFO

Import

For information regarding the usage of external files for PWV or RH measurements, refer to ...

mesowest.id (*string*):

Detail

The measurement site identifier for the MesoWest database

Example

KONM, KRAP

wyoming.id (*string*):

Detail

The measurement site identifier for the Wyoming Upper-Air database

Example

ABQ, EPZ

wyoming.weight (*string*):

Detail

The weighting used on the PWV measurements for analysis.

Note

If there is multiple sites, these values should add to 0.5.

Example

0.4, 0.2, 0.5

2.1.2 Raw Data File

The raw data file is processed, through pattern identification, allowing for a flexible format with few strict requirements. One of these requirements is that the sky and ground temperature should be separated into groups and ordered the same way as the configuration file. Here are three examples of data files:

- Dataset Example 1²
- Dataset Example 2³
- Dataset Example 3⁴

It should be noted that the columns do not have to be in any set order, with one small caveat, the model pulls the data from columns with headers containing specific words or phrases. The caveat is with regards to Ground and Sky temperature readings. The temperature measurements must go in consecutive order by sensor as determined by `_pmat.yml`.

² <https://github.com/physicsgoddess1972/Precipitable-Water-Model/blob/master/data/example/example1.csv>

³ <https://github.com/physicsgoddess1972/Precipitable-Water-Model/blob/master/data/example/example2.csv>

⁴ <https://github.com/physicsgoddess1972/Precipitable-Water-Model/blob/master/data/example/example1.csv>

For example, if the order of the sensors in `_pmat.yml` is 1610 TE, FLIR i3, and then AMES 1. Then the order of the ground and sky temperature measurements in the dataset should be: 1610 TE, FLIR i3, and then AMES 1. (As seen in Dataset 2).

Date (*datetime*, ``YYYY-MM-DD``):

Detail

The date of the measurements.

Time (*datetime*, ``HH:MM``):

Detail

The local time of the measurements

Sky temperature (*float*):

Detail

The sky temperature measurements. The header of this column should be Sensor Name (Sky), where Sensor Name is the name of the sensor used in the configuration file.

Ground temperature (*float*):

Detail

The ground temperature measurements. The header of this column should be Sensor Name (Ground), where Sensor Name is the name of the sensor in the configuration file.

2.2 Output Data Formatting

There are a variety of data files generated by the software suite. The data files are stored as CSV files, with each row presenting data for a single day.

2.2.1 General data files

The primary data file [`master_data.csv`] generated is the full dataset that includes:

- Date
- time
- sky condition (clear sky/overcast)
- ground temperature
- sky temperature
- Radiosonde PWV
- Relative Humidity
- Dewpoint
- User comments

2.2.2 Machine learning

The machine learning data file includes five columns:

- Date
- Average brightness temperature

- Average PWV
- Relative Humidity
- Sky Condition

This data set supports the classification of data by the sky condition label.

2.2.3 Analytic results

The main analytical results are stored as YAML configuration files. The results of each step in the iterative analysis process are saved to a file with the name `_output.yml`. An example of this file is presented below. [sample of `_output.yml`] [table of the fields in `_output.yml`]

The averaged results of the steps are also stored in a YAML file. [sample of `_results.yml`] [table of the fields in `_results.yml`]

REPORTING ISSUES WITH PMAT

3.1 Bug reporting

The developers of this project are human and can make mistakes.

3.1.1 Opening a Ticket

3.2 Feature requests

CHAPTER FOUR

TEMPLATES

4.1 Input Data File

4.1.1 `_pmat.yml`

```
- instruments:
  - sensor:
      name:
      error:
      color:
      ratio:
      range:
      emissivity:
      poster:
      active:
- analysis:
  - train_fraction:
  - rel_difference:
  - iteration:
      step:
```

(continues on next page)

(continued from previous page)

```
    seed:
- logging:
  - verbose:
- import:
  - mesowest:
    - id:
  - wyoming:
    - id:
  weight:
```


PMAT REFERENCE

5.1 `pmat_analysis.r`

module

Precipitable Water Model Analysis
Tool: Analysis

synopsis

This module contains analysis
functions

`exp.regression(t=NULL, mean.out)`

Detail

Function includes all of the stuff to generate the exponential regression model with intervals

Parameters

- **t** (*double*) – training fraction
- **mean.out** (*list*) – the output of `mean.filter`

Returns

returns the data series and model statistics

Return type

list

`lin.regression(x, y)`

Detail

Linear regression function

Parameters

- **x** (*double*) – the domain of the dataset
- **y** (*double*) – the range of the dataset

Returns

returns the data series and model statistics

Return type

list

`data.partition(x, y, tr.sz=0.7)`

Detail

splits the data into a training/testing set

Parameters

- **x** (*double*) – domain of the data
- **y** (*double*) – range of the data
- **tr.sz** (*double*) – fraction of the data in the testing set

Returns

a list containing the training and testing

sets

Return type

list

`iterative.analysis`(*obool*, *mean.out*,
overcast=args\$overcast)

Detail

computes regression statistics and outputs to a yaml file

Parameters

- **obool** (*logical*) – determine whether to generate new `_output.yaml`
- **mean.out** (*list*) – output of `mean.filter`

Returns

iterative stats and `_output.yaml`

Return type

list

`lsvm`(*x*, *y*, *l*, *tr.sz*=0.7, *seed*=`sample(1:2^15, 1)`)

Detail

Generates a Linear Support Vector Machine and draws the decision hyperplane and support vectors

Parameters

- **x** (*double*) – domain of dataset
- **y** (*double*) – range of dataset
- **l** (*double*) – labels of the dataset

- **tr.sz** (*double*) – fraction of data to be used for model training
- **seed** (*integer*) – the random seed

Returns

list of data, labels, and the coefficients

Return type

list

5.2 pmat_processing.r

module

Precipitable Water Model Analysis
Tool: Pre-processing

synopsis

functions for preprocessing

colscheme(*range*)

Detail

a function that generates an array of colors
based on the number of elements

Parameters

range (*list*) – a list of data series

Returns

a list of colors

Return type

list

`mean.filter(nan.out, n)`

Detail

filters the data based on the comparison of the daily std and the average std of the dataset

Parameters

- **nan.out** (*list*) – the output of `nan.filter`
- **n** (*integer*) – threshold

Returns

an array of indices for PWV values to be analyzed

Return type

list

`dna.filter(fover)`

Detail

removes data labels as Do Not Analyze

Parameters

fover (*list*) – overcast.filter results

Returns

overcast.filter results with DNA points removed

Return type

list

`nan.filter(stuff)`

Detail

removes nan values from a set of lists

Parameters

stuff (*list*) – list of arrays

Returns

returns list with filtered data and the indices with nans

Return type

list

`inf.counter(bool, snsr_data, label)`

Detail

identifies the -Inf values

Parameters

- **bool** (*logical*) – decides if -Inf is not replaced with NaN
- **snsr_data** (*list*) – the dataset
- **label** (*character*) – the identifier for the dataset (e.g. sky, gro, skyo, groo)

Returns

data set that replaces all -Infs for NaN (If `bool == FALSE`).

Return type

list

`index.norm(x)`

Detail

calculates the normalized index of the dataset

Parameters

x (*double*) – data range

Returns

an array of values between 0 and 1

Return type

double

`overcast.filter(col_con, col_date, col_com, pw_name, snsr_name, cloud_bool)`

Detail

Filters our data with overcast condition

Parameters

- **col_con** (*integer*) – column index for condition labels
- **col_date** (*integer*) – column index for date stamp
- **col_com** (*integer*) – column index for comments
- **pw_name** (*list*) – pw measurement labels
- **snsr_name** (*list*) – sensor labels
- **cloud_bool** (*logical*) –

Returns

A list of lists containing either clear-sky/overcast data

Return type

list

`sky.processing(overcast)`

Detail

Computes average values and weighted averages

Parameters

overcast (*list*) – results of the `overcast.filter` function

Returns

series of arrays including average PWV, RH, etc.

Return type

list

5.3 `pmat_products.r`

module

Precipitable Water Model Analysis
Tool: Products

synopsis

plotting functions for PMAT

`time.pwindex(datetime)`

Detail

Normalized PWV index for both clear sky and overcast data

Parameters

date – the datestamp of the data

`time.nth_range`(*range, title, color, leg.lab, ylab, datetime, overcast*)

Detail

Multirange Time Series plot series

Parameters

- **date** – the datestamp of the data
- **overcast** (*bool*) – the condition of data (clear sky/overcast)

`time.composite`(*range, title, color, ylab, datetime, overcast*)

Detail

Time Series composite plot series

Parameters

- **date** – the datestamp of the data
- **overcast** (*bool*) – the condition of data (clear sky/overcast)

Returns

A sky temperature time series plot

`time.mono_composite`(*range, title, ylab, datetime, overcast*)

Detail

Time Series composite plot series

Parameters

- **date** – the datestamp of the data
- **overcast** (*bool*) – the condition of data (clear sky/overcast)

Returns

A sky temperature time series plot

`time.multiyear(range, title, color, datetime, ylab, overcast)`

`analysis.nth_range(overcast, x, y, title, label, color, leg.lab)`

Detail

Super Average Plot with Exponential Fit

Parameters

overcast (*bool*) – the condition of data (clear sky/overcast)

Returns

A sky temperature time series plot

`analysis.regression(overcast, x, y, des, label, iter)`

Detail

Super Average Plot with Exponential Fit

Parameters

overcast (*bool*) – the condition of data (clear sky/overcast)

Returns

A sky temperature time series plot

`analysis.svm(model)`

`pac.compare`(*overcast, des, x, y, angular, radial*)

Detail

Pac-Man plot of Super Average Plot

Parameters

overcast (*bool*) – the condition of data (clear sky/overcast)

Returns

A sky temperature time series plot

`pac.regression`(*overcast*)

Detail

Pac-Man residual plot

Parameters

overcast (*bool*) – the condition of data (clear sky/overcast)

Returns

A sky temperature time series plot

`chart.histogram`(*range, xlabel, title*)

Detail

Histograms of defined quantities

Parameters

- **range** – a data range
- **xlabel** – the xaxis label
- **title** – the title of the histogram

`poster.plots`(*overcast, iter, mean.out*)

Detail

The set of all poster

Parameters

overcast (*bool*) – the condition of data
(clear sky/overcast)

Returns

All available poster plots

poster1(...)

poster2(*overcast, iter, mean.out*)

Detail

The analytics poster plot

Parameters

overcast (*bool*) – the condition of data
(clear sky/overcast)

sensor.chart(...)

Detail

overcast distribution charts

sensor.time(*overcast*)

Detail

Instrumentation time series plots

data.gen(*overcast, dir*)

Detail

creates a datafile containing the date, avg temp, and avg pwv for a defined condition

Parameters

- **overcast** (*bool*) – the condition of the data (clear sky/overcast)
- **dir** – directory path

`data.ml(dir)`

Detail

creates a datafile containing the machine learning relevant information

Parameters

dir – directory path

`data.step(seed, i, coef, r, S)`

`data.final(dir, clear.len, over.len, train.len, nan.len, frac.kept, coef, std, rmse, overcast=args$overcast)`

`visual.products(set, mean.out, datetime=datetime, overcast=args$overcast)`

Detail

saves plot sets

Parameters

- **set** (*character*) – the set identifier
- **overcast** (*logical*) – overcast boolean

5.4 pmat_run.r

module

Precipitable-Water Model Analysis
Tool

synopsis

The main file for PMAT. Documentation available at <https://docs.pmat.app>.

5.5 pmat_utility.r

module

Precipitable Water Model Analysis
Tool: Utility

synopsis

general functions for PMAT

logg(*msglevel*, *msg*, *dir*=*out.dir*, *lev*='INFO')

Detail

creates log entries for _log.txt

Parameters

- **msglevel** (*character*) –
- **msg** (*character*) –
- **dir** (*character*) –
- **lev** (*character*) –

`aloha.first()`

Detail

shows first time user information

`aloha.startup()`

Detail

shows title banner for program

`aloha.closing()`

Detail

cleans up files and ends the program

`reset_time(datetime)`

Detail

A function that sets the time to 00:00:00

Parameters

datetime (*character*) – a Date or date-time object

Returns

A datetime object with time 00:00:00

Return type

double

`time_axis_init(date)`

Detail

A function that calculates the min, max, and position of the tick marks for

Parameters

date (*double*) – A date or datetime object

Returns

The max, min, and tick mark positions

Return type

list

time_axis(*datetime*)

Detail

A function that sets the x-axis format for time series plots

Parameters

datetime (*double*) – A date or datetime object

std_title(*des, overcast*)

Detail

A function that generates the title based on

Parameters

- **des** (*character*) – the description of the plot
- **overcast** (*logical*) – the sky condition

Returns

a title string

Return type

character