The main point of data management in this context is deriving season-long NDVI charts from the raw irradiance data.

[example graphic]

We'll cover the complexities of this, both in general principles, and the nuts-and-bolts.

Users have also wanted charts of other data, such as the temperature trends. We call these collectively the "microclimate" datasets, though these are generalized and can include the spectral data used for NDVI.

[example graphic]

Microclimate data is a separate "branch" of the data management, which has details in its own right.

First, we'll discuss the steps to derive the NDVI charts. We'll cover the principles first, without reference to any particular operating system or software. Then we'll delve into the details of how you can perform each of the steps using specific tools. Along the way, we'll need to introduce certain abstractions, such as the concept of "Series" of data.

We'll follow a similar path for the microclimate data.

In the data principles, we will need to be specific about the data sources. We originally developed the equipment using instrumentation from Onset Computer Corp.

<http://www.onsetcomp.com/>

These are off-the shelf weather station parts, which output data in specific formats, and have certain limitations of available sensors. Being general purpose, these instruments cannot be optimized for NDVI work, especially in regards to size and bulk of the equipment.

In the course of our research, one of us began experimenting with customized hardware, specific to NDVI work. He realized that most of the functionality of a thirty-five pound mantis, with its heavy frame, large instrument boxes, and bulky cables, could be done by a small instrument the size of one's fist. This is called the "Greenlogger" project. At the time of this writing, prototypes are available for testing and loan. Current status is available at:

<http://www.rickshory.com/greenlogger.htm>

The data from these two sources is in different formats, though our data processing handles most of the details transparently.

# Data pre-processing:

To use data from the Onset data loggers, you must first convert files from native format to text format. We refer to this native format as "Hobo" format, after Onset's line of Hobo data loggers. Hobo format is proprietary and cannot be directly read by any other software.

To covert the Hobo files, use the HOBOware software "HOBOware pro" from Onset Computer Corporation.

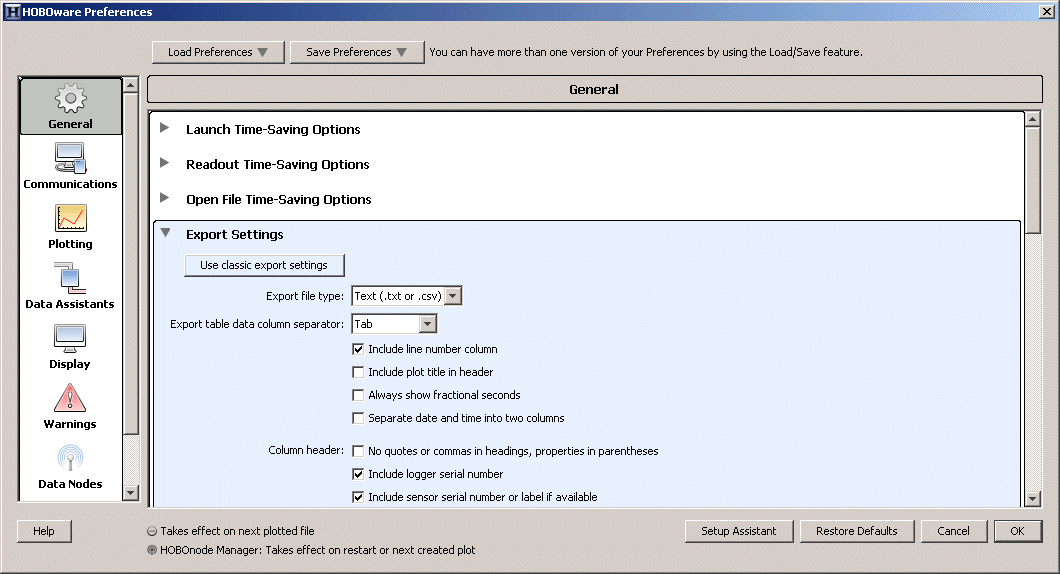
http://www.onsetcomp.com/products/software

In early versions of HOBOware, the file conversion tool was a plug-in you had to manually install, but in later versions it is included.

Before actually converting, there are some Hoboware issues.

The latest HOBOware version 3.5.0, seems to be fussy. It sometimes will not run, giving a Java error. This seems to happen when there are too many other things running on your computer. Rebooting seems to fix it.

Hoboware needs to be set up with the correct options for export, so the text files will import into our database systems (see next page for details).



Make these settings in HOBOware:

File > Preferences... > General > Export Settings:

Export file type: Text (.txt or .csv)

Export table data column separator: Tab

Include line number in column: Yes

Include plot title in header: No

Always show fractional seconds: No

Separate date and time into two columns: No

Column header:

No quotes or commas in headings, properties in parentheses: No

Include logger serial number: Yes

Include sensor serial number, or label if available: Yes

Date format: Y M D

Date separator: Dash (-)

Time format: 24-Hour

Positive number format: 1,234.56

Negative number format: -123

File > Preferences... > Display > Default Unit System

Unit System: SI

File > Preferences... > Display > Date/Time

Date format: Y M D

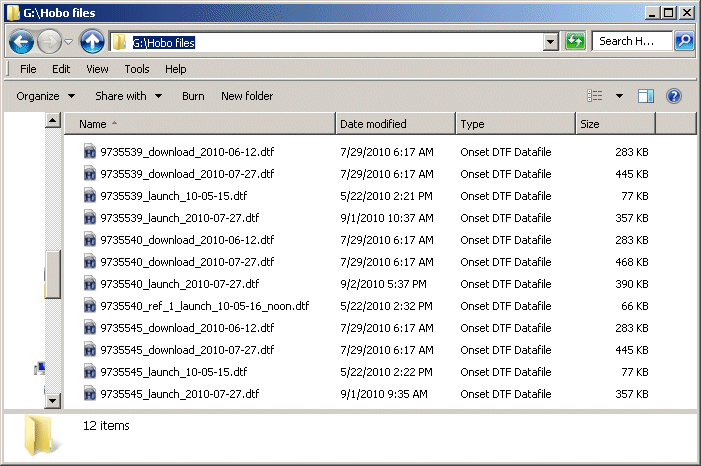
Date separator: Dash(-)

Year format: 4-digit Year

Time format: 24-hour

Then, do the actual text export. It's easiest if you have all the Hobo (DTF) files you want to convert in one folder, and create a separate empty folder where the converted text files will go.

You may have Hobo files downloaded on various dates, and not too consistently named, as shown here. In this example, we have files from three different loggers, with serial numbers ending in 39, 40, and 45. You can deal with any inconsistencies later in the database, as long as you have the complete data record for the time period of your study. You do not even need to have the serial number in the file name; it is stored internally in the file.

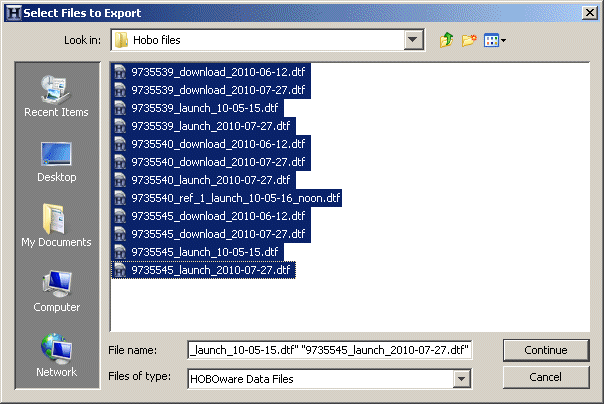


In HOBOware use the Bulk Export Tool:

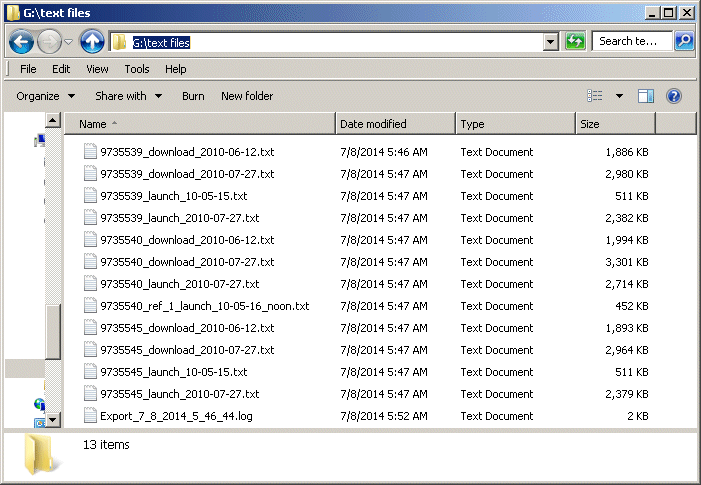
Tools > Bulk File Export > Select Files to Export ...

Rather than browsing to the file source and destination, it may be easier to copy and paste the source and destination folders into the corresponding screens of the export tool.

In the Hoboware file export tool, select the Hobo files to convert:



The Hobo export tool creates text files corresponding to the original Hobo data files, plus a log file.



Every version of HOBOware creates a slightly different text file format. Most of them have bugs, such as mis-aligned columns and random junk characters. We have written the database import routines to catch all the bugs we know about and transparently fix them. We have fixed all the bugs we know of, up to Hoboware version 3.5.0, but if your imported data looks odd contact us for further bug fixes [where for long term support?].

# Deriving season-long NDVI charts from Solar Energy (SE) and Photosynthetically Active Radiation (PAR) sensors.

This is a platform-independent "recipe" for deriving NDVI from solar energy (SE) and photosynthetically active radiation (PAR) sensors. Such sensors are available from Onset Computer Corp. The SE sensor is a silicon pyranometer, part number S-LIB-M003. It has units of watts per square meter. The PAR sensor is part number S-LIA-M003. It has units of microeinsteins (µE), which is µMoles of photons/square meter/second.

Derivations from "Greenlogger" data, mentioned above, would somewhat different. For those calculations, contact the developer via:

<http://www.rickshory.com/greenlogger.htm>

This derivation, from Onset sensor data, assumes you have four different season-long irradiance logs from SE and PAR sensors. Two of these would be from down-looking sensors to detect reflectance from the ground surface, and two from up-looking sensors for a sky reference to compensate for weather changes. All readings should be accurately timestamped in Universal Time (UT, same as Greenwich Mean Time) or derivable as that from local time.

If you have, or wish to use, only reflectance readings (from down-looking sensors) and no up-looking reference readings, there are notes on how to vary the calculations.

Mask out or otherwise eliminate all data readings known not to be valid, such as before/after sensors were properly emplaced, when sensors might have been shaded by work nearby, when spectrally anomalous instrumentation was in their field of view, when unseasonal snow was present, etc.

Format the readings into "records" consisting of the following five fields:

1.) TUT: timestamp, in Universal Time.

2.) SEU: "solar energy up", from the reference solar radiation sensor looking up at the sky.

3.) PARU: "PAR up", reference photosynthetically active radiation sensor looking up at the sky.

4.) SED: "solar energy down", the solar radiation sensor pointing down to read reflectance from the ground surface.

5.) PARD: "PAR down", the PAR sensor looking down to read ground reflectance.

The Up and Down readings may not have been recorded by the same instrument, and may not contain synchronized timestamps. In this case, choose one data series as the source of TUT. If, for example, if one Up sensor served as the reference for a number of Down sensors, the simplest choice would be to use the Up (reference) series and match the closest Down (reflectance) readings to create each record.

Sky irradiance can change rapidly with the movement of clouds, so the closer the time match the better. Ideally, any time difference should be a few seconds at most. A suggested (consistent, programmable) rule of thumb is to eliminate any records where the time difference between Up and Down readings is more than half the interval between. For example, if the recording interval is two minutes, consider any Down reading not matchable if it is one minute or more from an Up reading. You could also use an absolute time difference cutoff, such as ten seconds.

If your Up readings are very noisy, or you do not have such reference data, use these fields instead.

1.) TUT: timestamp, in Universal Time, form Down readings.

2.) SE: "solar energy", the solar radiation sensor pointing down to read reflectance from the ground surface.

3.) PAR: "PAR", the PAR sensor looking down to read ground reflectance.

In this case, you may not be able to derive absolute NDVI, but only relative values. In the following steps, adapt the calculations.

Derive local solar time (ST) from UT using site longitude (LON) and Equation of Time (EOT).

ST = UT + (LON / 15) + EOT

One hour of time difference is equivalent to fifteen degrees of longitude. This is the source of the "(LON / 15)" term. Choose the sign based on your convention, positive or negative longitude. This term will be in hours. Multiply by 60 for minutes, 3600 for seconds.

EOT ("Equation of Time") is the difference between solar time and clock time due to astronomical factors (the earth's orbit is elliptic rather than circular, and the apparent path of the sun is along the ecliptic rather than the celestial equator). It is possible to calculate EOT, however a value looked-up by ordinal day of the year is accurate enough. Tables are available online. Take care to get the sign correct as EOT is nearly a quarter of an hour at some times of the year.

Derive DT, the difference of ST from solar noon. Eliminate all records where the absolute value of DT is greater than 2 hours. This leaves only mid-day values from the four-hour time window plus and minus two hours of solar noon.

Season-long spectral recordings are noisy. Noise sources include weather changes, digitization errors, and random shading of sensors. We experimented with various ways to eliminate erratic data. These included:

- The degree of change between successive readings, both of the original spectral bands, and the derived NDVI.

- The consistency of a moving subset of the readings. For example eliminate a record if the cumulative change in the previous and following (e.g two or three) readings were too great.

We concluded most of these techniques were overly complex. We found that a cutoff based on thresholds relative to normal solar maximum was simple and consistent. This is the method:

Find an example day when the sky was clear in the four-hour time window around solar noon. The best way to do this is to observe daily traces of these bands and look for smooth bell-shaped curves. A partly cloudy day gives a jagged curve. Times when the sun is obscured give low values. Times when the sun is out can have peaks much higher than on clear-sky days because of reflections off clouds.

The following is a heuristic method for finding cloudy days; that is, narrowing the search for clear days:

Find or guess at the maximum reading (IMAX) for a spectral band. You can start with the maximum for the whole season.

Set a cutoff (ICUT) at a test percentage (PCT\_CUT) of that value, such as 90%.

ICUT = IMAX \* PCT\_CUT

Query the data by the criterion of readings (for that spectral band) below ICUT, during the four-hour solar noon time window. Days having even occasional clouds will virtually always show pronounced dips in irradiance. These are the days to eliminate as "cloudy".

Experiment with different values of IMAX and PCT\_CUT. Except in data from the driest deserts, this method will rapidly identify 80% to 90% of cloudy days, and leave a relatively small sample to manually look through for a representative clear day.

You may find it practical to allow a few irradiance "dips" per day, for example if the shadow of a cable fell across the sensor every day, even on clear days. Something like this, of course, does not indicate cloud conditions.

After you have found a representative clear day in your data, derive maximum values of the four spectral bands (SEU, PARU, SED, and PARD) during the mid-day clear sky time windows. If you have multiple instruments, the clear-day maxima will be different for each.

Eliminate all records from your data where spectral values are less than 75% of these maxima. This removes cloudy periods, and most random shading events. If this leaves too little data, you can try a cutoff of 50% instead, or even 25%.

Convert PAR to solar energy equivalent by multiplying by a conversion factor as below. This converted approximation is "VIS". ("VISU" = up-looking, "VISD" = down-looking).

VISU = 0.21\*PARU

VISD = 0.21\*PARD

This approximation converts photon count to energy.

Alternatively, you can use the reciprocal, as below, to convert energy to equivalent photon count. The final NDVI result is normalized, so only the ratio between the bands matters.

VISU = PARU

VISD = PARD

SEU = 4.76 \* SEU

SED = 4.76 \* SED

Now that all bands are in the same units, derive a surrogate "near infrared" (NIR) band by subtracting out the visible from the total solar energy.

NIRU = SEU - VISU

NIRD = SED - VISD

Derive the reflectance values for visible and near infrared by dividing the observed reflected by the incident irradiance from the sky.

VIS = VISD/VISU

NIR = NIRD/NIRU

If your sky irradiance bands are very noisy, or you do not have these bands, use the observed reflectance bands from the ground alone.

VIS = VISD

NIR = NIRD

Finally, derive NDVI, by the familiar formula, from VIS and NIR.

NDVI = (NIR - VIS) / (NIR + VIS)

Depending on daily weather, you may have different numbers of NDVI values for different days. For example, recording at two-minute intervals, on a clear day, if all readings were "good", the four-hour solar noon time window would give you about 120 readings (30 per hour, for 4 hours). On a dark, cloudy day, you would have fewer.

The obvious way to aggregate the NDVI values is as an average. However, the average will be "better" the greater number of values you have. To get a measure of how "good" it is, use the Standard Error of the Mean (SEM).

STDEV = standard deviation of set of NDVI values for a day

COUNT = number of NDVI values for a day

SEM = STDEV / (sqrt(COUNT))

If charting NDVI values by date, SEM provides good error bars.

If your original spectral readings are not well calibrated, or if you do not have sky reference readings, your NDVI values may only be able to show relative changes through the season rather than absolute NDVI levels. In this case, it may be more useful to chart by relative NDVI (REL\_NDVI), normalized to have values from zero to one.

MAX\_NDVI = maximum NDVI for the whole season

MIN\_NDVI = minimum NDVI for the whole season

REL\_NDVI = (NDVI - MIN\_NDVI) / (MAX\_NDVI - MIN\_NDVI)

If you derive MAX\_NDVI and MIN\_NDVI from the daily averages and then apply the REL\_NDVI formula to original values, some results will be out of the zero-to-one range. This is because some of the values that went into the maximum average are greater than that average, and similarly for the minimum. (The numbers are valid, but may give odd statistical results.) On the other hand, if you derive MAX\_NDVI and MIN\_NDVI from individual (pre-average) values, the daily averages will take up less than the full zero-to-one range.

# "Channels" and "Series"

For configuring data, we introduce the concepts of "channels" and "series"

By analogy with TV programming, when you tune to a channel, you get everything that's streaming in. But say you want a certain show, a certain series. It would be handy if you could just ask for that series, and get all the episodes of the show, even if some of them came in on different channels. You'd also want to leave out all the dead time, reruns, and commercials.

Data loggers are something like TV stations. Instead of video, a logger provides a stream of numbers, and they get recorded into a file rather than broadcast. Most kinds of loggers provide multiple channels, but the concept is the same.

Think of a spreadsheet where the first column is timestamps. The other columns are sensor readings taken at these timestamps. Each column would be a channel. An example might be the data from, say, sensor serial number 9721335, representing temperatures, in units of °C, and so on. The channel specifies exactly what the numbers *are*, but not what they *mean*.

A series on the other hand might be "Air temperature at 1m height at the outlet of Clear Lake". This is what the numbers *mean*. You can think of it as the salient "story" of the data.

We will cover the mechanics of how you work with channels and series in the data management instructions. The main point is that data processing can automatically extract channels; however you have to manually tag data to specify a series. By analogy, a tuner could automatically lock on to a TV channel and record it, but you'd have to label the shows.

Why isn't a series always the same as a channel? There are several reasons, including:

• Like in the television analogy, different "episodes" may come in on different channels. For example if a sensor dies and you replace it, it's a new channel but it continues the same "story".

• There are various sources of bad data. Like TV commercials, they come in on a channel but they're not part of the story line.

• Successive downloads from a logger may have redundant, overlapping data. Like TV reruns, they're no reason to have them in your data twice.

What can you do with series, after you've got them?

Much of data processing is relationships between data series. For example, an NDVI trend in its simplest form is the relationship between the near-infrared data series and that of visible light. How you do this in the data processing is covered in the documentation, but it's far easier to use a meaningful series like "visible" or "infrared" than a channel specification such as "sensor serial number 9700890" (or was that sensor number 9700809?).

Setting up the various data series in your database may seem tedious, but it allows you to use these meaningful names in analysis. And, for each database, you only have to do it once.

# Software:

In our data management system we have two options. The more mature option uses Microsoft Access databases. The other option uses cross-platform GUI modules written in Python.

We are developing this second option because:

* Many users don't have Microsoft Office (which MS Access is part of).
* Many researchers are committed to non-Windows operating systems like Mac OS or Linux, which won't run MS Access.
* Microsoft has a poor track record for maintaining backwards compatibility. There are even rumors Microsoft may discontinue support for Access.

In both options, the processes for data management are much the same, though implemented differently. If you can use MS Access, you will have the advantage of more mature methods.

We give complete instructions below for each system. If you read through them both, you will find the discussions somewhat redundant. The material is arranged this way so you can follow either path without reference to the other.

# Data processing option 1

# Microsoft Access databases

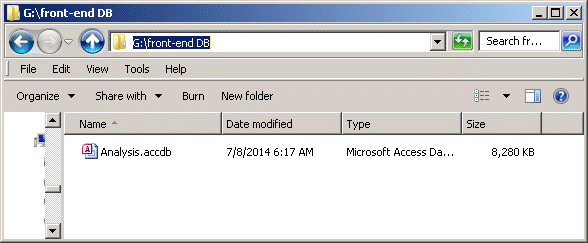
[Where will users be able to get the databases, long term?]

Setup:

These instructions assume you have Microsoft Access installed, so you can work on MS Access database files.

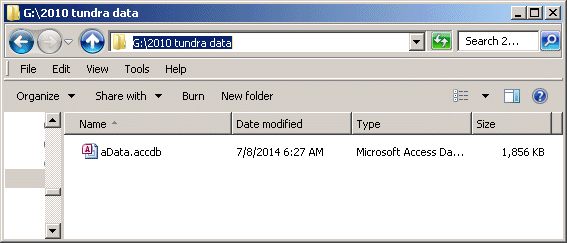
The ZIP file [will it be a Zip file?] will contain a database file named "Analysis.accdb" and a folder containing other files.

"Analysis.accdb" is the "front-end" which is like a control panel that you can use to operate on various data sets. The other folder is the "back-end", which represents one particular data set.

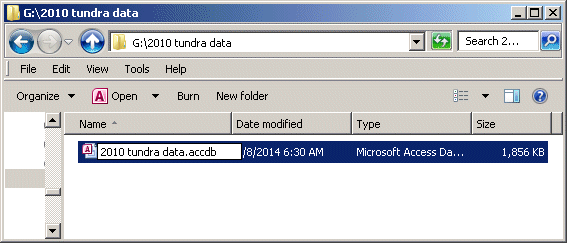


For ease of use, put "Analysis.accdb" (the front-end) in one place and the other folder (back-end) somewhere else. Think in terms of the front-end being where you will usually work. You might eventually have a number of different back-ends, like books on a shelf, each for a different project/year, or however you organize your data sets. You can plug into each different back-end when you want to use it. Perhaps a better analogy than "book" is "disk" or "tape", because you will usually use the front-end to read it, rather than opening it directly.

The back-end folder originally contains only one file, though the data processing will soon create more. The original file is named "aData.accdb". This is the main back-end file. If you want to keep an empty back-end, make a copy of this file and folder.



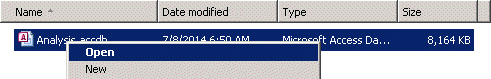
You can rename the back-end file "aData.accdb" to whatever you want, for example "2010 tundra data.accdb", as a reminder that it is your data from your tundra study in year 2010.



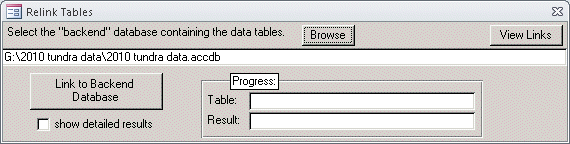
The data processing will create additional files in this same folder that have names beginning with "Data...". It can be handy to name your main file so it alphabetizes first, before all the "Data" files. This makes it easier to find if you ever need to work in it directly, or examine the filename to know which dataset this is.

The data processing will create a separate database file, in the back-end folder, for each calendar day of data. It was necessary to break the data down like this to come in under the 2 Gig size limit of Windows files, for some large project data sets. These daily files will have names like "Data\_For\_2010-05-16.accdb". Do not change any of these names.

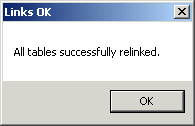
Open the front-end database, either by double-clicking it if you have Access files associated with MS Access, or by opening MS Access first and then browsing to the file with the Open dialog.



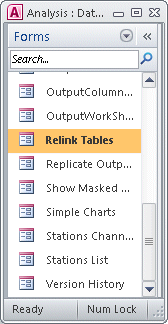
You will probably get a message that some link is broken. This is because the links are to specific file/folder locations on the source computer. Fixing the links is largely automatic. A form named "Relink Tables" should open.



Browse to "aData.accdb" (whatever you renamed it) and click "Link to Backend Database". The links should be refreshed to the new location on your computer. If you rename or move the back-end file or folder, you will need to re-link again.

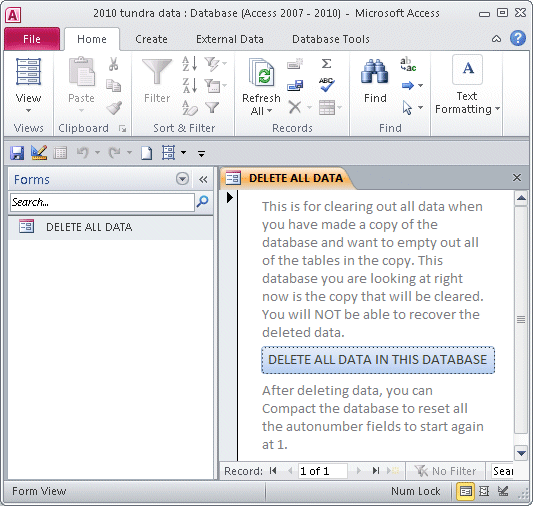


The front-end database "Analysis.accdb" can only look at one back-end at a time. If you want to look at more than one back-end at the same time, you can simply create copies of the front-end, and link each one to a different back-end.

 You link to the back-end you want to work on using the same "Relink Tables" form that automatically appeared the first time you opened the front-end. You find forms in an Access database by looking in the object bar to the left. The selector at the top can show "All Access Object" or only a certain subset of objects, here "Forms". Then, the objects are listed in a column that you can scroll. Double-click the object to manually open it.

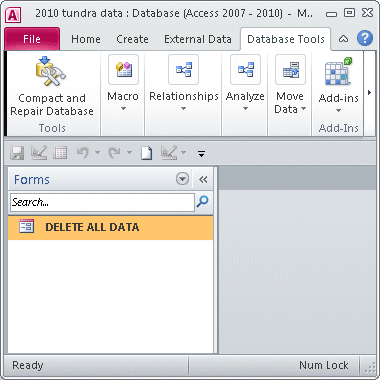
It may get confusing which back-end the front-end is linked to. You can check the link using the "View Links" button in the "Relink Tables" form.

If you want to set up a new data set, and you did not save an original empty copy of the back-end, this is how to create one: Find the main back-end database file for an existing back-end data set, for example "2009 tundra data.accdb". Copy only this one file to a new empty folder. Rename the file to something meaningful for what the dataset is going to be, for example "2010 tundra data.accdb". Open this file in Access. This is one of the few times you work on a back-end file directly.



In the main back-end database file, there is a form named "DELETE ALL DATA". Open this form and click the "DELETE ALL DATA IN THIS DATABASE" button.

Then, compact and repair the database to reset all the table counters to 1. In Access version 2010, this is Database Tools > Compact and Repair Database.



You can close this now-empty back-end database file, as there is no more need to work in it directly. From the front-end, use the "RelinkTables" form to link to the new empty back-end file.

There are 3 general parts to working with a project's data:

1. Import
2. Setup
3. Output.

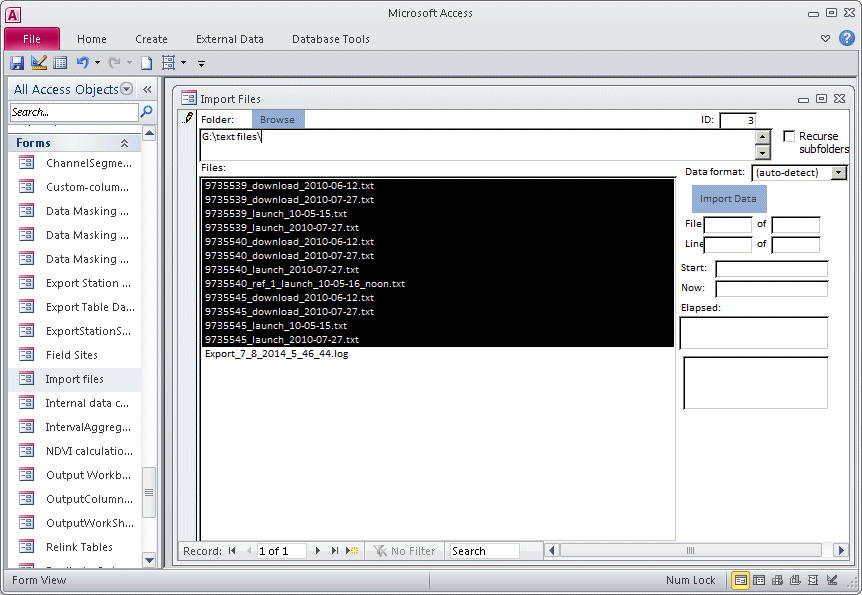
# 1. Data Import

For import, there are two stages. First, converting the native Hobo format files (downloaded from data loggers) to text format. Then, importing these text files into the database.

File conversion is covered in the "Data pre-processing" section above. It is the same for both our processing systems.

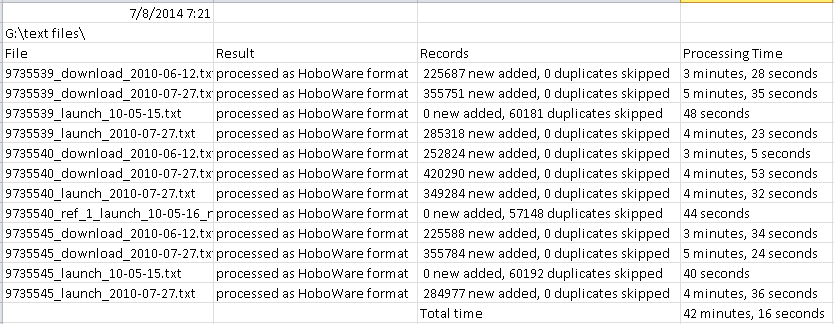
From the front-end database, use the form "Import files" to import the text files converted by HOBOware. Browse to the location of the text files and select the ones you want using Shift and/or Ctrl selection (you can skip the log file).

It may take a while to process all the data, but it should be automatic. This is the point to watch out for new HOBOware bugs.



The process gives extensive diagnostics, including creating a job log spreadsheet when done. If something looks wrong, save this job log and send it [where for long term support?].

Each database import creates a job log spreadsheet.

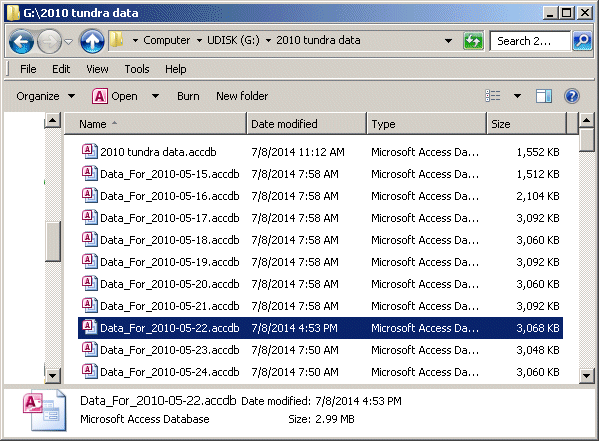


This gives a list of all the files processed, with details including the time each one took. As shown in this example, there were duplicate records in some of the files, but the import process simply ignored the duplicates. So, if you are not sure whether data files have been imported yet, it's safe to try re-importing them.

The data import process automatically converts local time to Universal Time (UT) using the time zone information in the text files. After import, data are stored in the database as UT time. Local time is recoverable, if needed, by stored "time zone offset" numbers.

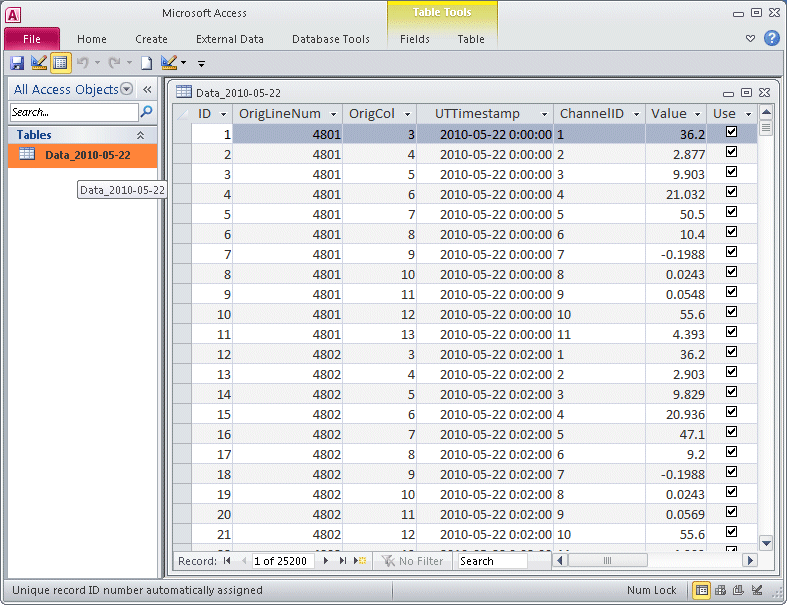
# 2. Data Setup

If you look in the folder of your back-end database, you will see that the import process has created a series of new files.



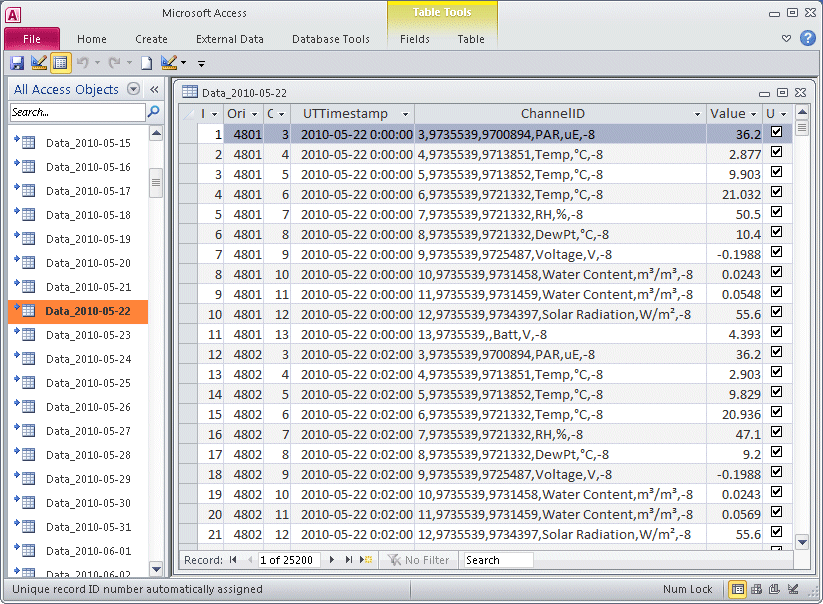
These are the data files, one for each day, mentioned earlier. (If you have database files open in MS Access when you view a folder, you may see files with the extension "laccdb". These are small "link" files MS Access temporarily creates when it uses a database file.)

If you open one of these daily files, you will see that it contains one table, also named for the calendar date.



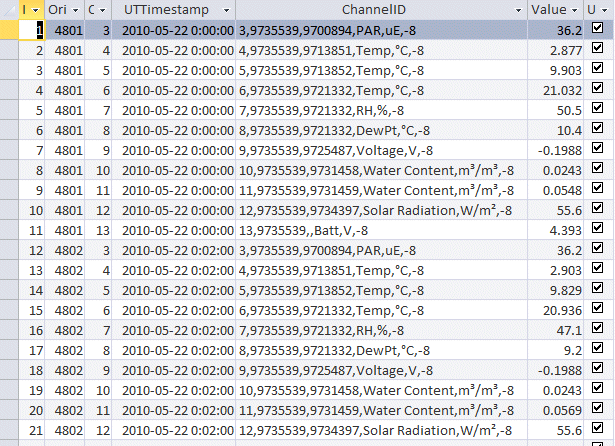
Here you can see that the import process has organized the data into a generalized and consistent format.

If you look in the front-end DB, you will see the same tables with the same data.



Here, the table icons (in the left column) appear with small arrows by them. This indicates these are "linked tables". They are not "in" this database file, but are somewhere else, accessed by their "link". If you float over one of these, you can view the link address. These links are how the front-end connects to the data in the back-end folder.

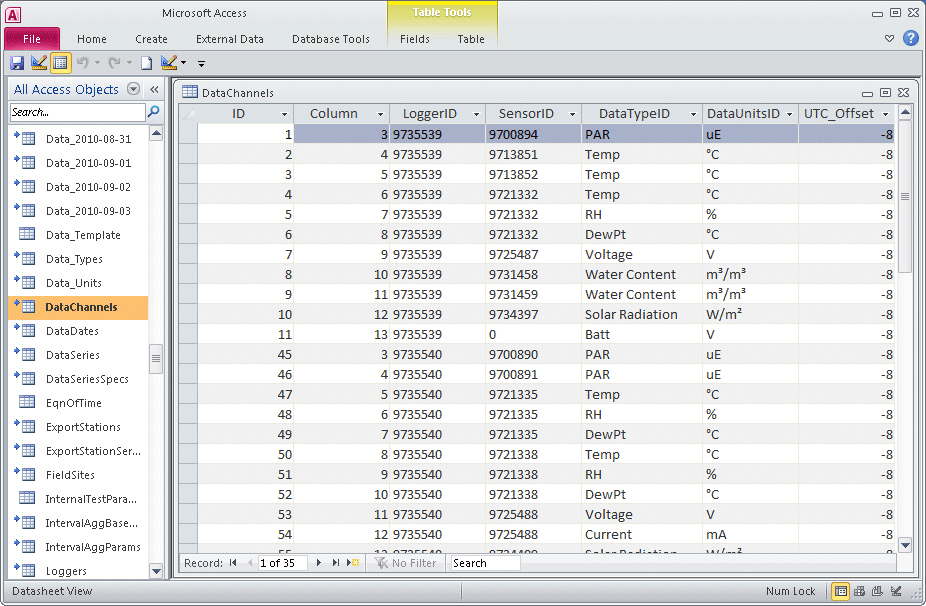
The main difference in appearance between the local and linked view is the "ChannelID" field.



When viewed locally in its own database, the ChannelID appears as simply a number. As a linked table, this field appears as compactly coded text.

In both cases, the ChannelID is "really" a number. However the linked table is configured to display a "lookup" instead.

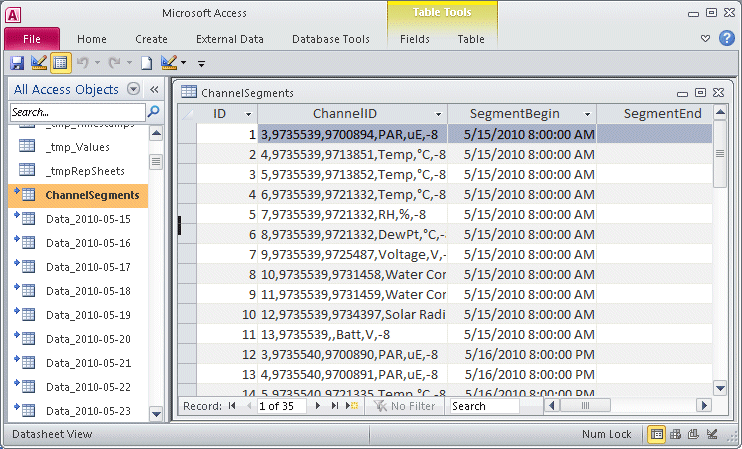
The lookup refers to the table "DataChannels". This is also a linked table, but in the main back-end database file. (Most tables in the front-end are linked. This is what allows the front-end to work on various back-ends. It's a matter or re-directing the links.)



Without going into too much detail, this mechanism of using a lookup allows the system to store data very efficiently. The DataChannels table has a record for every existing pattern of Data Type, Data Units, UTC hour offset, etc. This is coded by the ID number in the first column of DataChannels. Then, each individual item of data only has to store a ChannelID, which matches that ID number.

The ChannelID is also a convenient handle for data. For example, in this case, ChannelID = 1 is the PAR data for a certain sensor on a certain logger. That same number will be in the data tables as the ChannelID for all those data records, and only those data records.

The final abstraction that allows completely handling the data is of "Channel Segments".



Ideally, the data from a sensor would all be relevant. In practice, the pertinent data set has a certain extent through time. The sensor is deployed at a one beginning timestamp, and then turned off or repurposed at a second ending timestamp, some time later.

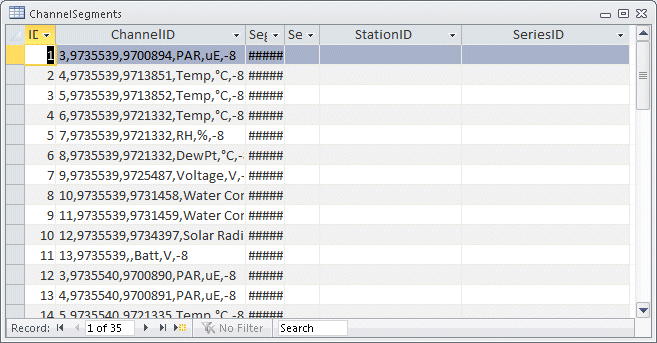
This, then, is the data "story". Or, rather, the "story" is one or more ChannelSegments. In most cases, it is only one channel segment. However, if a sensor fails and is replaced, the "story" continues in an additional channel. The second segment begins after the first one ends.

The data import process automatically parses data into Channels. Then, it assigns default ChannelSegments. The default "SegmentBegin" is set to the earliest timestamp that exists for a given Channel. Since more data typically come in during the course of a deployment, the convention is to leave "SegmentEnd" blank. In all the analysis, a non-existent SegmentEnd is interpreted as "all the data up to Now". You would manually override this is in the case of sensor failure mentioned above, and explicitly set a SegmentEnd. You would also override this if a sensor were re-purposed. This conceptually separates its data into two segments, one for each purpose.

Successive data imports handle things automatically in most cases. If data come in with an earlier timestamp for a Channel, the import process adjusts SegmentBegin to that new earliest date for the default ChannelSegment. If you explicitly set a SegmentEnd, any new data after that become a new ChannelSegment.

How do we work with ChannelSegments as the data "story"? This is the part you must manually assign.

This is the ChannelSegments table with the timestamp columns narrowed, and the view scrolled over to show two additional fields.



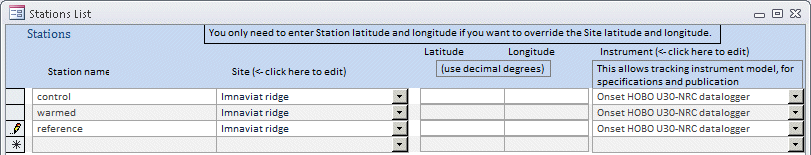
These two fields, "Station" and "Series" are still blank. Generally, Station corresponds to the overall logger while Series corresponds to the individual sensor.

These data are sampled from a study of tundra climate change. There were several replicates each of two treatments (early snow removal and supplemental warming). The various Stations were named by their replicate site, and whether they represented treatments or controls.

In this trimmed-down sample, we will call the Stations simply "warmed" and "control", with an additional "reference" Station, which we will discuss later. Thus, the Station is a meaningful description of the "context" or "role" of a particular logger.

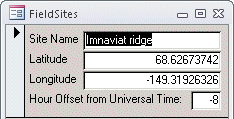
First, we'll set up the information about Stations. Then we'll assign ChannelSegments of data to these stations.

To set up Station information, use the database form "Stations List".



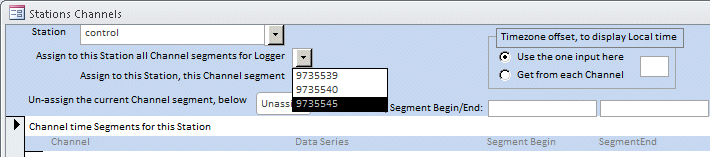
In the first column you enter a meaningful name for the Station. The Station names can be up to 25 characters. Don't use these characters (:\/?\*[]) because they are disallowed in sheet names of Excel workbooks, which will be part of the data output.

The main additional information we need for a Station is its longitude. The "Site" column is a shorthand way of entering it.

 The analysis uses longitude to calculate solar time. Longitude to within a degree is usually good enough. In many cases, Stations are close enough together to share coordinates. So, setting up a Site, and then assigning it to various Stations is more efficient. You can also explicitly enter a Station's latitude and longitude, which will then override any from the Site.

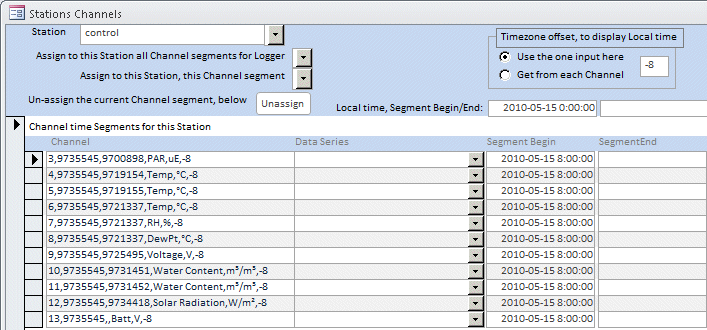
The Instrument column is optional, but available for storing metadata about the loggers.

After Station records are created, you could manually assign them in the ChannelSegments table, but the "Stations Channels" form makes it easier.

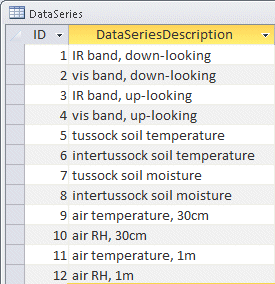


Presumably, you have kept a record of which Station had which logger. In the graphic above, the station named "control" is about to have all the channel segments assigned to it that came from logger serial number "9735545". Later, the Series will distinguish the individual sensors.

After the above task, the Channel segments are assigned.



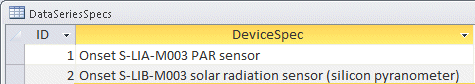
The timezone offset makes it easier to verify things are correct. In this case, the logger was set up so it started at midnight (0:00:00) local time. Converted to Universal Time (UT), this start time might not be so intuitively clear. The timestamps within the records are from the database, and so they are in UT. However, the "Local time..." bar above the columns offers a convenient translation back to the local time, for whichever record is currently selected.

 In a similar way, we then select the other Stations, assigning logger "9735539" to "warmed", and "9735540" to "reference". Later, in setting up analysis, we will be able to use these meaningful Station names instead of cryptic logger numbers.

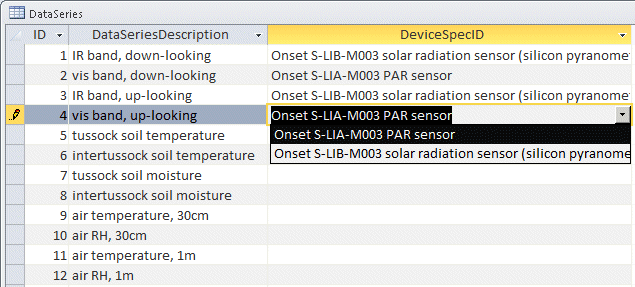
Next, we set up Series. You work directly in the tables for this, as a form would give little advantage. Open the table "DataSeries" and type in the names (examples shown). If you want to store metadata (examples below) about the Series, first put these in the table "DataSeriesSpecs". Then, they will be available to pick from in the "DeviceSpecID" field of table "DataSeries".

A Series typically corresponds to a sensor. It is a way of giving a meaningful name to a sensor, similar to how a Station gives a meaningful name to a logger. In this study, there were spectral sensors, to collect the irradiance band data needed for NDVI calculations, and additional "microclimate" sensors for soil and air parameters.

If you want to store metadata about the sensors, put these descriptions in the table "DataSeriesSpecs".

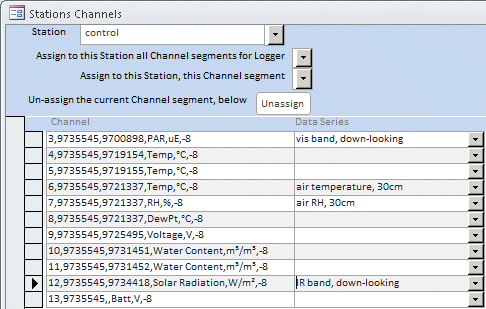


Then, these will be available to pick from in the "DataSeries" table.



These descriptions are not used anywhere in analysis, but might be useful for documentation. When newly added to "DataSeriesSpecs", they might not be available to pick from until you re-open the table "DataSeries".

We return to the form "Station Channels". Now, the Series items you added to "DataSeries" are available to pick from, to assign ChannelSegments to.

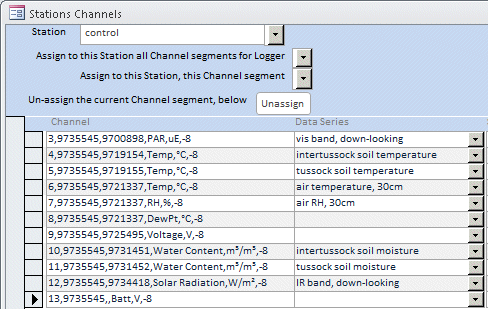


Presumably, you know what sensors you have plugged into your logger. Some are easy to assign because they are unique. For example, Photosynthetically Active Radiation (PAR) is visible light, so it is the "vis band". Every station, except reference stations, have only down-looking spectral sensors, so the "down-looking" choice is made. Similarly for assigning "IR band, down-looking" to the Solar Radiation channel (it *contains* the infrared, though also visible; see the general derivations above, and the further setup instructions later).

Three "sensors" all have the same sensor serial number (the 2nd 7-digit number in the Channel text), 9721337. This device is a composite sensor for both air temperature and relative humidity (RH). The Channels share this same serial number because they are physically one device (the 3rd Channel, "DewPt" is a derived Channel). Since, in this design, the temp/RH sensor is mounted at 30cm, these two channels are known and assigned.

Other channels, which differ only in a cryptic serial number, take some work to assign. In this study, there are pairs of temperature and water content sensors. They monitor the heterogeneous tundra, one of each pair in a "tussock" of vegetation, and the other of each pair in the space between tussocks. In other studies, multiple sensors of the same type may be plugged in to the same logger, so this is a generalized problem.

You may be able to determine which is which by using HOBOware. That software allows you to view any or all of the channels in a log file. You can "spike" a sensor, for example making a temperature sensor anomalously warm for a brief period. You would keep a record of exactly when you did this, and then later look for that spike in the log file, to see what channel it was on.



In any case, once you assign Station and Series to a ChannelSegment, it's done. You don't have to do it again. Then, you can make use of that ChannelSegment by the meaningful name of the Data Series.

Note that you don't need to fill in a Series for Channels you won't use: For example "Batt" (the battery voltage) in the example above.

Data Masking

The other important step in data setup is Data Masking. This means ignoring invalid data. For example, when the loggers in this sample were started, the tundra was still frozen so the soil temperature sensors could not be got in. Therefore, the sensors were really recording random air temperature, not valid soil temperature.

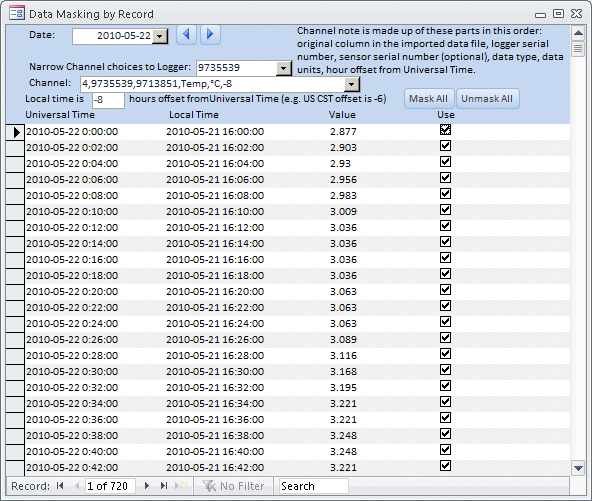
We don't usually delete bad data, but just "mask" it out, ignore it. It's always possible you may want the data back again. If you open one of the data tables, the rightmost column is true/false field named "Use". The default, when a record is created is that Use = True, which means to use that data record in analysis. If you mark any data records to Use = False, those records are ignored in all analysis.

It's not practical to manually mark and un-mark the Use field directly in Data tables, so there are various shortcuts (described next). However, this is how all these methods work, behind the scenes. They just mark and un-mark the Use field.

You can use the shortcut options for data masking via three different forms:

* Data Masking by Record
* Data Masking by Interval
* Data Masking by Specs

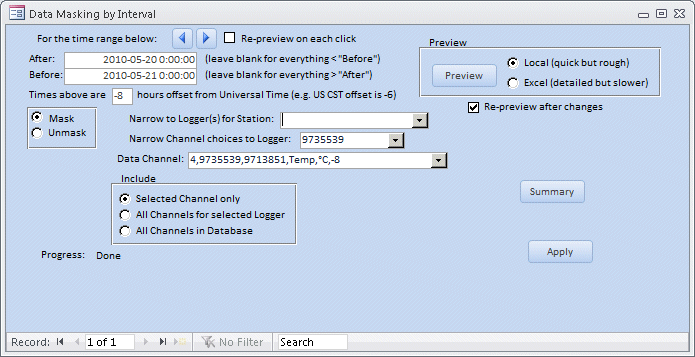
Data Masking by Record



This form pulls out all the records for a certain Channel for a certain date, and lets you mark and un-mark them manually. This is most useful for an obvious glitch, such as if you "spiked" a sensor to see what Channel it was. You would just un-check the spike values and they would be ignored in analysis.

You can "walk" back and forth through the dates using the arrow buttons next to the Date selector. You can translate the UT timestamps back to local time, to more easily correlate the time with any field notes you may have.

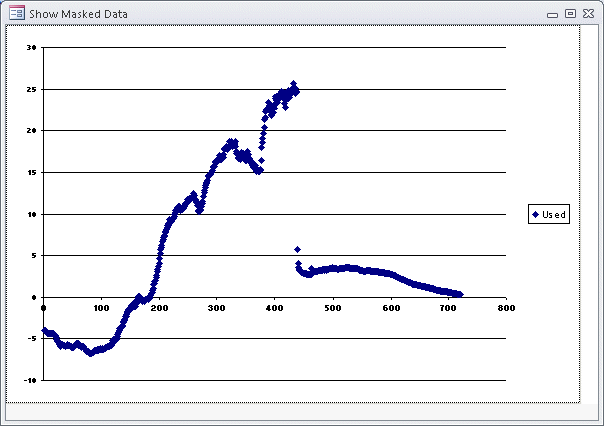
Data Masking by Interval



This form has a lot more capability. It allows you to visualize the data and home in on a section you want to mask.

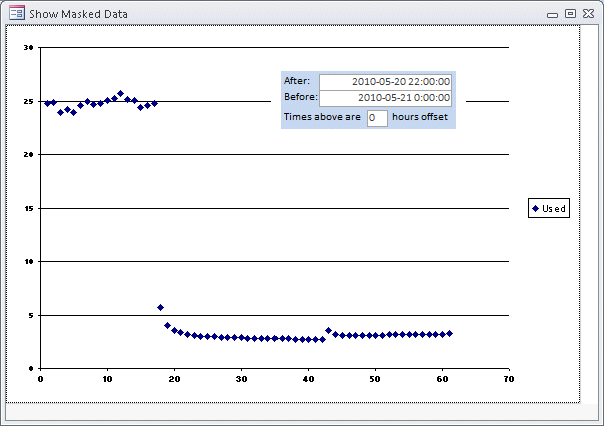
For example, the sensor for this Channel (soil temperature) was not correctly emplaced at first because the tundra was still frozen when the logger was deployed on May 15. The sensor was just sitting on the surface for the first several days. Ideally, you would take note of when the sensor did get correctly emplaced. However, field work is not always ideal.

"Walking" through the data day by day, and looking at previews, we see that the readings were going through wild daytime excursions, as you would expect from a temperature sensor sitting on top of the ground, exposed to the sun.

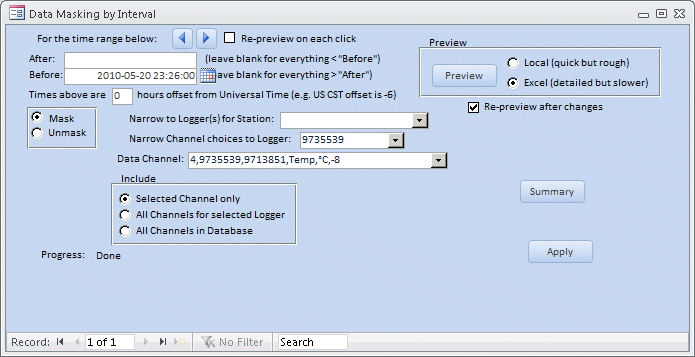


On May 20, the wild excursions abruptly stop, and the readings settle down as you would expect for valid soil temperatures. If this corresponds to your field notes as to when the tundra was finally thawed enough to insert the temperature sensor, that's great. But if such notes were never taken, at least the data give a clue.

Now, using the form, you can look at narrower and narrower intervals until you determine the exact time when the data became valid.

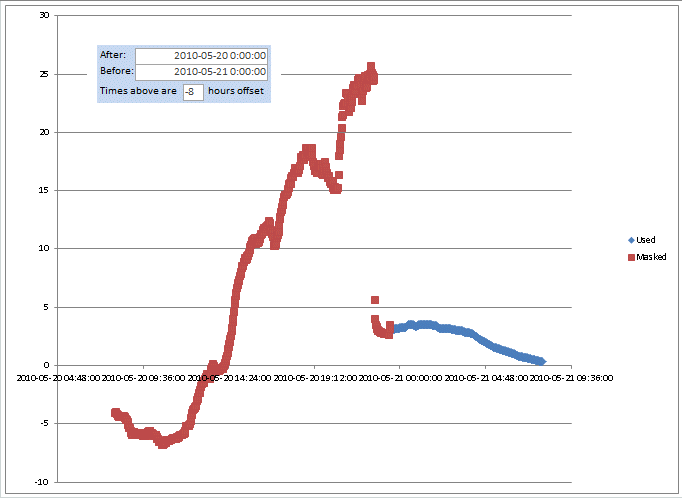


When you have your masking interval set up, click the "Apply" button.



You can use duration intervals, but this example is an open-ended interval that will mask everything before the given time point.

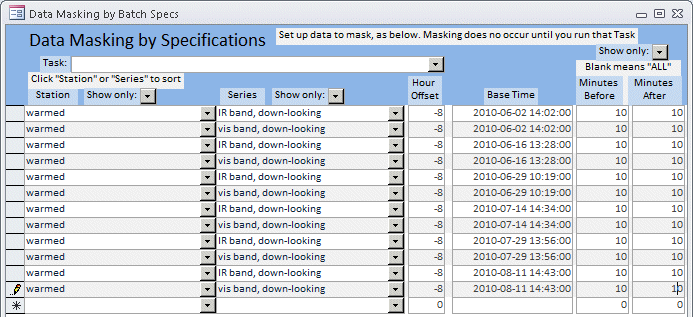
Returning to a broader preview, you can verify that the invalid data are now masked.



Data Masking by Specifications

(Form name: "Data Masking by Spec")

This is the most general and versatile form of masking. You specify a set of intervals to mask, then click a button and they are masked.



An example of usage was a case where gas flux instruments were periodically put in the field of view of the mantises. This, of course, interfered with the spectral readings. Nobody kept track of the exact times when this was done, but we generated the list of base times from the timestamps of the gas flux files. We allowed a generous interval of ten minutes before and after, for setup and takedown of the gas flux equipment.

You can also use this form to mask out invalid data at the beginning and end of seasonal deployments, or for various other scenarios.

# 3. Data output:

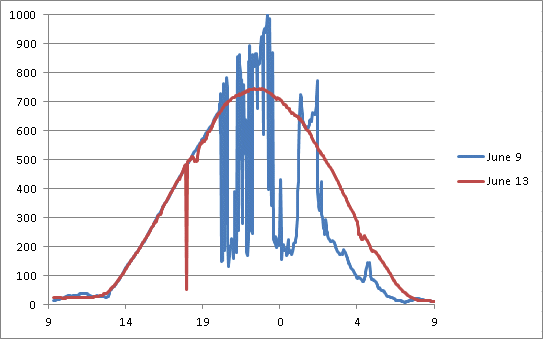
* NDVI datasets
* Other datasets

NDVI datasets

You generate NDVI datasets using the form "NDVI calculations".

Users have typically wanted multiple NDVI dataset; for example, separate datasets for different subsets of plots, or datasets run using different criteria. Therefore, the "NDVI calculations" form uses "panels". You set up all your preferences for a dataset, and these settings constitute a "panel". You can easily copy and vary a panel to create a new different panel. All the panels are saved. You can come back to any existing panel and run it again. This is especially useful because there are several different data tasks you can run on any given panel.

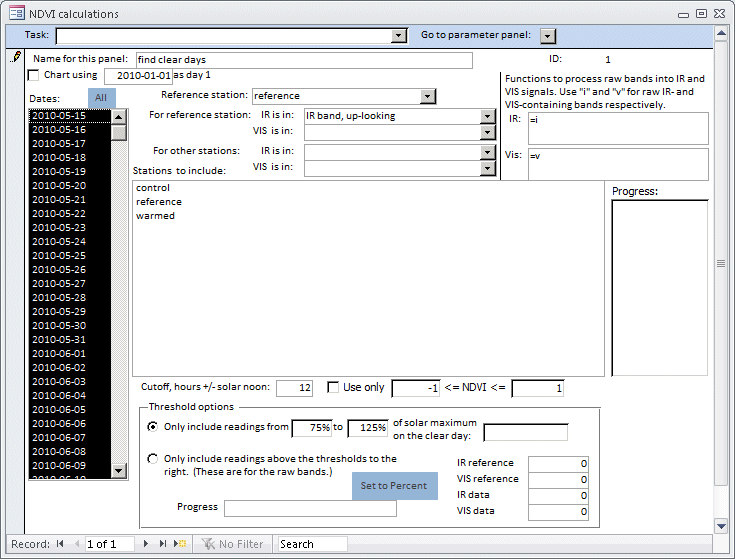
The first step in developing an NDVI dataset is limiting the source data. Ideally, we want clear-sky readings. In practice, the irradiance levels have a lot of "noise"; highs and lows due to changing cloud cover.



In this comparison, June 13 was a clear nearly all day. June 9 started clear, but was cloudy from mid-day on. Notice that, even though clouds cause marked drops in irradiance, there were also peaks much higher than the clear sky levels. This is due to reflections off clouds.

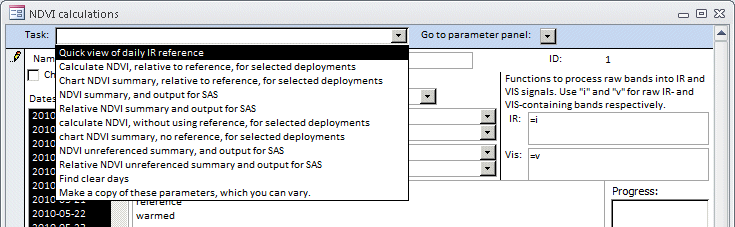
We have concluded that, the more irradiance levels differ from typical clear-sky, the more spectrally anomalous they are likely to be, and the less suitable for NDVI. So our main filtering of data is by cutoff thresholds, above and below typical clear-sky levels.

The first task is determining what the typical clear-sky levels are. This amounts to finding a clear day in the data. The "NDVI calculations" form has a process for doing this.



We start by setting up a panel, named "find clear days". For "Reference station" we could use any Station at a site because they all experienced the same weather, but we have one named "reference". This was a mantis with up-looking irradiance sensors as well as down-looking ones. The task we are about to describe could look at any irradiance Series, but it is arbitrarily defined to use the one selected at "For reference station: IR is in:". Here, we choose "IR band, up-looking". We change the default "Cutoff, hours +/- solar noon:" from 2 to 12, so as to take in the entire 24-hour day. The other parameters in the panel will be ignored.

We choose the task "Quick view of daily IR reference"



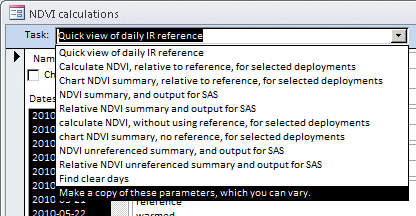
The process creates an Excel workbook with one sheet for each day. It charts the irradiance of the specified Series.

The chart traces look much as in the comparison above, of June 9 vs June 13. By examining the charted sheets, we conclude that June 13 is the best day. The thing to look for is a smooth bell-shaped curve. If there are several clear days to choose from, we pick the one closest to summer solstice.

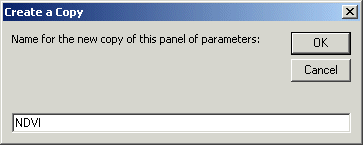
Even for this rudimentary task, the data processing does a lot behind the scenes. From the longitude you put in in the Station setup, the process calculates local solar time and centers the irradiance charts on solar noon.

Now, we configure a panel to output NDVI charts. We could start a new panel from scratch by advancing the record selector at the bottom of the form, but we will illustrate copying a panel.

Select the task "Make a copy of these parameters, which you can vary."



The form prompts you for a name for the new panel. We will call this one simply "NDVI".



The new panel looks exactly the same as the previous one, except for the name.

Begin setting up the "NDVI" panel by entering the date of the clear day we chose in the last step, June 13.



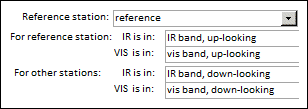
The 75% to 125% default thresholds will usually be suitable, but you can vary them if you don't get enough valid data points. Each of the four Series that go into the NDVI calculations will have its own maximum level. The "clear day" parameter on the form instructs the processing to look up the correct level for each of those channels, and apply the relative thresholds.

The other option under "Threshold options" allows you to set absolute lower thresholds for the spectral bands. We retain this for some back compatibility, but most users find the "clear day" thresholds easier to use.

Change the Cutoff hours from 12 back to the default of 2.

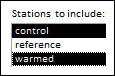


This limits the readings to a 4-hour time window centered around solar noon. This is when daylight is spectrally best for deriving NDVI. If the NDVI output still ends up with outliers that are obvious glitches, you can try cutting them out using the post-calculation thresholds here.



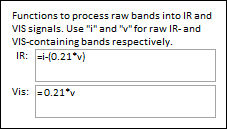
For NDVI processing, this form uses four bands for each chart (optionally only two, detailed later). The standard NDVI formula uses visible light (Vis) and near-infrared (IR) bands reflected up from vegetation (i.e. down-looking sensors). To better compensate for changes in weather, this form's processing by default pre-compensates by scaling these bands relative to two corresponding Vis and IR bands that are recorded from the sky (up-looking sensors). In this example, these bands come from a reference Station named "reference". The two relevant Series for that Station are named "IR band, up-looking" and "vis band, up-looking".

You may recognize these names from setting up the Stations and Series. We defined these names and then assigned them to the correct ChannelSegments. This points up the value of using Stations and Series. Now, you can pick from these meaningful names, rather than looking up cryptic logger and sensor serial numbers.

 Below the Series selectors on the form, you choose the Stations you want charts for. In this trimmed-down sample, there are only two. In a real study, there could be many more. You select from the list in the usual way, using Shift and Control.

Notice that we do not include the Station named "reference". It serves as the reference for the charts, but we do not need a chart from it. That station did happen to have down-looking sensors, but they were over a random patch of tundra, not any experimental site.

A logger on an experimental site could be set up with both up- and down-looking sensors. In that case, that Station could serve as the reference for charts, and so its up-looking sensors would be the two reference Series. You would also select it in this lower list, to get a chart from.

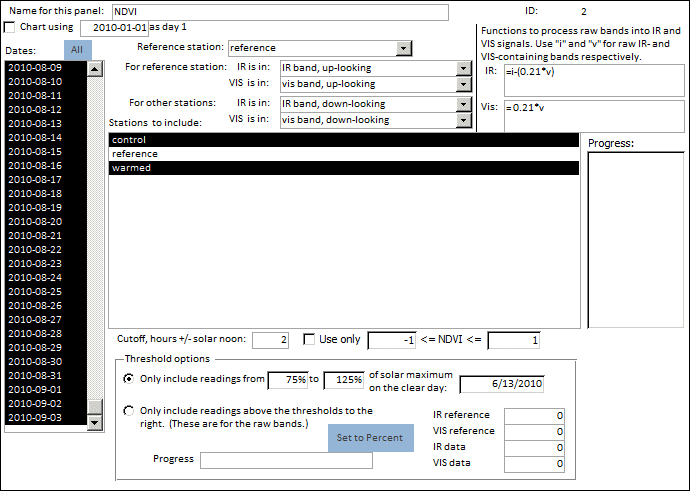
 As explained in the general principles, Onset sensors are not inherently optimized for NDVI. While the PAR sensor is suitable for a "visible light" band, the Solar Radiation sensor used for the "infrared" band reads both visible and infrared. In addition, the two sensors' data are not in the same units.

Beginning with the "Vis" formula: The "0.21" coefficient is an approximation to convert PAR (symbol lowercase "v", which has units of photon count) to the same energy units as Solar Radiation (watts per square meter).

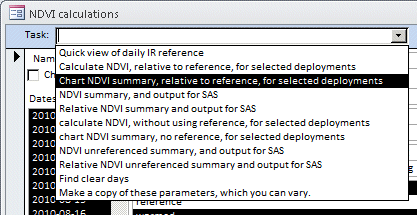
The "IR" formula expresses that the raw Solar Radiation band (symbol lowercase "i") includes both the infrared signal we want, and also the same visible component delivered by the PAR sensor. Subtracting out that visible component, including its units conversion, approximates a "near infrared" band.

The form provides places to put these parameters so you can use generalized formulas. Other sensors, such as photodiodes, would have very different formulas. Write the formulas in Excel format. This is fairly standard across several computer languages. The data processing copies these formulas directly into spreadsheet cells, only substituting cell references for "i" and "v".

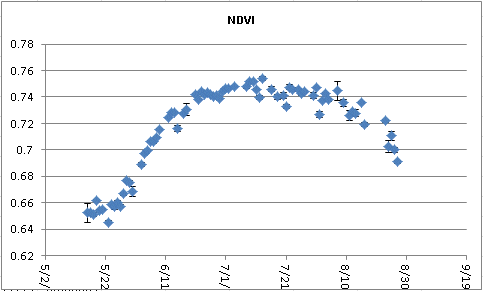
At this point, the "NDVI" panel looks like this.



To create, the NDVI charts, select the task "Chart NDVI summary, relative to reference, for selected deployments".



This option creates an Excel workbook. The workbook has two worksheets for each Station that was selected to chart. The first in each pair of worksheets is a table of NDVI calculations that includes each data record. The second worksheet is a summary, by date, of the first sheet, including a chart. The example below is from the station named "control".



The other task selections have these differences:

1.) "Quick view of daily IR reference": (already described, creates daily irradiance charts)

2.) "Calculate NDVI, relative to reference, for selected deployments": Same as the example workbook (3), except without the daily summary sheets.

3,) "Chart NDVI summary, relative to reference, for selected deployments": (The example just given).

4.) "NDVI summary, and output for SAS": Same as the example (3), except with an additional workbook of values tabulated for easy statistical analysis using SAS.

5.) "Relative NDVI summary and output for SAS": Same as (4) except NDVI is normalized to the range zero-to-one. Makes it easier to see trends, though loses explicit values.

6.) "calculate NDVI, without using reference, for selected deployments": Same as (2) except NDVI formula uses reflectance bands directly, does not pre-calibrate relative to sky reference bands. Useful if reference bands are very noisy or not available.

7.) "chart NDVI summary, no reference, for selected deployments": Same as (3) except without using reference bands.

8.) "NDVI unreferenced summary, and output for SAS": Same as (4) except without using reference bands.

9.) "Relative NDVI unreferenced summary and output for SAS": Same as (5) except without using reference bands.

10.) "Find clear days": Heuristic process that can make it easier to identify clear days.

11.) "Make a copy of these parameters, which you can vary.": Copies the current panel.

The processing tasks take care of these details behind the scenes:

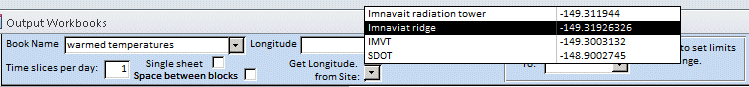
* Calculate solar time from Universal Time and longitude, and divide the data into days at solar midnight
* Make the closest time match between Series datapoints, if the timestamps are not exactly the same in the various Series.
* Automatically combine any Series that are made up of multiple ChannelSegments
* Automatically adapt the calculations, whether using sky reference Series or not

"Microclimate" datasets:

Users have wanted datasets from the other data Series as well as the irradiance Series used for NDVI. We call these colloquially the "microclimate" data because they are often from weather station sensors specific to a small site. However, they really could be any data. You can generate these generalized datasets, as Excel workbooks, using the form "Output Workbooks".

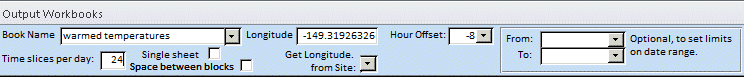
You conceive of the dataset you want as a three-tiered structure. You're going to create a workbook, even if you only have one sheet in it. Within a workbook, you can specify multiple sheets. Within a sheet, you specify each column.

The following will take you through an example that illustrates most everything the "Output Workbooks" form can do.



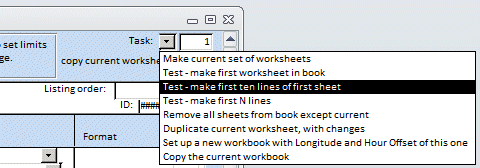
First, you need a name for a dataset, here "warmed temperatures".

There are various shortcuts for filling in parameters, such as selecting the longitude from an existing site. You can also select the hour offset. The calculations use these to get local solar time from Universal Time.



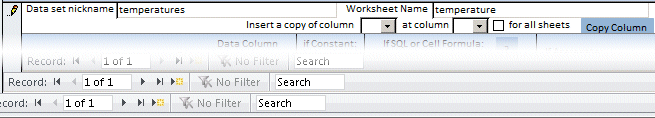
The parameters "Single sheet" and "Space between blocks" are rarely used, but available if you need the data for the "sheets", which you will set up next, to appear all in one worksheet rather than separate worksheets.

"Time slices per day" is a generalized way of dividing the data into the time blocks you need. For example 1 time slice per day gives you daily data. The example here, 24 time slices per day, gives you hourly data.



As you work, you can use Test tasks such as "... make first ten lines ..." to preview your dataset, to see if you're getting what you want.

The next tier after setting up a workbook is setting up the sheets for it. You navigate back and forth to different sheets, and create new sheets, using the second-from-outer level of navigation buttons along the bottom of the form.



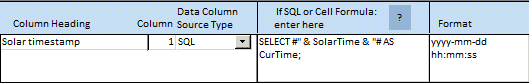
Since a sheet is what you usually think of as a "data set", you can optionally make a nickname for it. The "Worksheet Name" is the text that will appear on that worksheet's tab, which is limited to 30 characters, and some characters (:\/?\*[]) are disallowed.

There are a few other options for copying columns, which will be more apparent after we cover data columns.



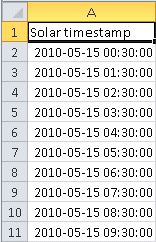
The final parameter for a sheet is its "Listing order", which is simply the order you want it to appear in the workbook.

Usually, you want one or more columns to show some aspect of the time the data were recorded. Here are some examples:



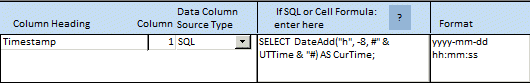
For specifying time, you can use "SolarTime" or "UTTime". This first example illustrates "SolarTime", which is seldom used but is shown for completeness.

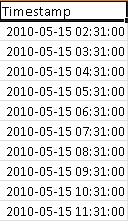
In creating a column record, you enter the "Column Heading". This is simply text that will be inserted in the top row of the spreadsheet. The "Column" parameter is the spreadsheet column where you want these data to appear; here column 1, the first column.

 For time data, the "Data Column Source Type" is "SQL". This means doing some operations directly on the database. The format is elaborate, so we give some exact examples. This example will fetch time points according to the "Time slices per day" parameter of the overall workbook dataset. For example, if "... time slices ..." = 1, it will fetch one per day, at noon. If "... time slices ..." = 24 (as it is currently set), it will fetch 24 per day, centered on the half-hours of the hours. The related data are always fetched by solar time because environmental data go by the planetary cycle, but unless you explicitly use "SolarTime" here in the formula, the corresponding Universal Time, is delivered as the label. Both are available, but in practice users have preferred "UTTime" to "SolarTime", to correspond with what the local clock says.

To complete this example, the "Format" parameter allows you to explicitly format the spreadsheet cells. To the right is the first ten lines of output.

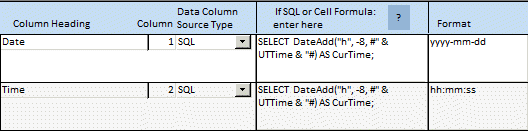
Users have tended to prefer "clock" time. This shows the changes from the previous, to use local clock time.

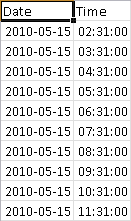


 "DateAdd" is an MS Access function which, in this case adds -8 hours to the Universal Time timestamp, to compensate for the timezone (here, Alaska Daylight Time). This is a rather extreme example, where local clock time really is more than two hours different from solar time.

Without the "DateAdd" function, this statement would show the Universal Time (Greenwich Mean Time) that all database records are converted into, for consistency. The convention is to explicitly write this into the SQL statement, rather than have the processing pull it from the logger's metadata, because in many cases individual loggers are set to different time offsets, and the offset changes with daylight saving time.

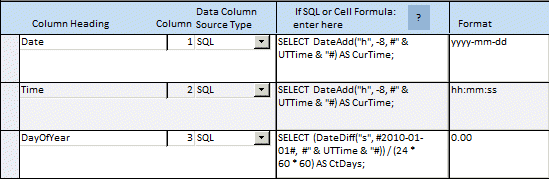
You may prefer Date and Time in separate columns.

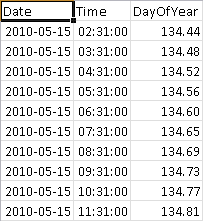


 The best practice is to set up two separate columns that both generate the complete timestamp. Then, format them differently, one for date and one for time.

This will assure your data sort correctly, regardless of which date/time column you use.

As you can see, the choice between "SolarTime" and "UTTime" becomes less important for larger time slices. In this case, the Date would "round" to the same for either.



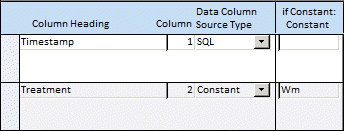
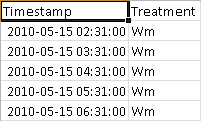


You can use more elaborate date/time calculations.

The "DayOfYear" expression extracts the difference, in seconds, since midnight of January 1. It divides this by the number of seconds in a day to get the fractional day, and formats the result as a decimal number.

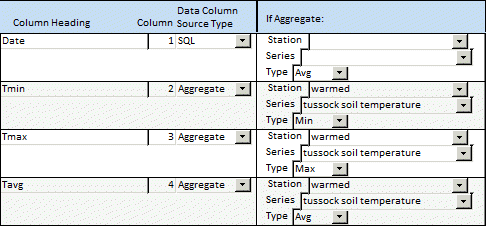
You could look at time difference from an environmental event, such as when the ground is first snow-free; or from a phenology stage such as the first flowering of an indicator species. You can create as many columns as you want. Note that the calculated field names (CurTime, CtDays) are "dummy" variables that do not appear in the results.

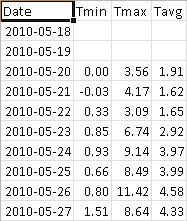
The usage of the "Constant" option of the "Data Column Source Type" parameter is simply to insert the same content in the specified cell of every row.



This is often a single character to identify the block, treatment, etc., to distinguish the content of different sheets.

The "Aggregate" option is the workhorse for pulling out the data you want.



 "Aggregation" is a database term for reducing multiple values to one. The Minimum, Maximum, and Average are illustrated here. In addition, you can choose Count, Sum, and Standard Deviation.

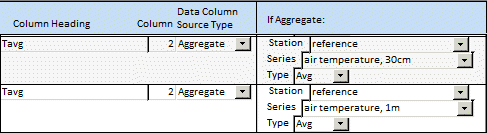
The processing pulls out values to aggregate based on time slices. In this example, the time slice is a whole day, so the processing pulls values timestamped *greater than or equal to* midnight and *less than* the following midnight. We built in this slight asymmetry to prevent ambiguous overlap.

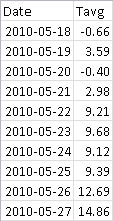
For hourly time slices, the aggregation is from *greater than or equal to* the current hour and *less than* the following hour. The nominal timestamp for the time slice is the midpoint, in this case the half-hour. This allows the most realistic charting. If nominal times were "reduced" to the beginning of the interval (e.g. the whole-number hour) the values would be time-shifted by half the interval.

As mentioned before, the timestamp criteria are always by solar time. The break between days is solar midnight. However, unless you use "SolarTime" in the SQL expression, this will be *labeled* with the clock time. This is usually what users find most meaningful.

Note that in the example results grid, there are no numbers until May 20. The logger was deployed May 15, but the tundra was not thawed enough to insert the soil temperature sensors until May 20. We "masked" out the sensor data before that time. If we had not done so, the numbers here would reflect the wild swings of a temperature sensor exposed to the sun. This also illustrates the importance of masking out any artificial spikes you put into the data to identify Channels. A brief 20 degree warming in your hand, to mark which Channel that sensor was recording, would show up as the (incorrect) maximum temperature for that day.

The processing is very strict about forcing you to specify a distinct number in the "Column" parameter, for every row in this form, with one exception.

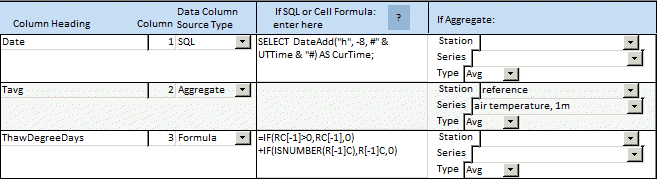


 You can aggregate multiple Series into the same column, if you use the same column heading, aggregation type, and format. The form strictly enforces this, to assure you have not set up two different data sources as the same column by mistake. In this example, we are combining the average of air temperature at 30cm and at 1m.

You can even combine Series from different Stations. For example, if you have five replicates of the same a treatment, at five different stations, this allows you to average over all of them.

In all cases, aggregations are done in "database style". For example, averages are over all existing values, not weighted by source. If there are 100 values from the first Series and 50 from the second Series, the average is over the total of 150. This is usually what you want, but it's good to be aware of it.

If aggregation will not give you what you want, you can insert Excel formulas into cells, to derive values from adjacent cells.

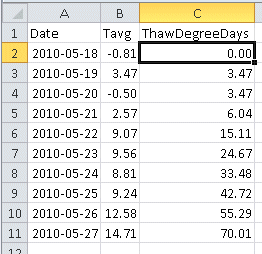


For example, here is a simplified formula for calculating cumulative thaw degree days. You write the formula in "RC" notation, which means relative to adjacent Rows ("R") and Columns ("C").

The formula is:

=IF(RC[-1]>0,RC[-1],0) +IF(ISNUMBER(R[-1]C),R[-1]C,0)

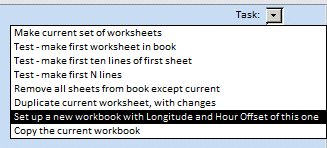
It may be easier to see what's going on from the format Excel translates it into:





This is the formula that ends up in the cell selected in the illustration to the right. Essentially, the formula sums the cell above it (C[-1] or "column minus 1") and the cell to its left (R[-1] or "row minus 1"). The "IF B2 > 0" term is needed to ignore any negative temperatures. The "IF ISNUMBER ..." term takes care of the first row, which will have only the column heading above it, instead of a valid number.

Often, you want new workbooks or worksheets similar to existing ones.



You can use options in the Task selector in the upper right corner to save time.

The task "Set up a new workbook with Longitude and Hour Offset of this one" creates a new empty skeleton workbook. Then, you can go to the original and copy sheets to the new one using the "copy current worksheet to" selector.

The "Copy the current workbook" task does what it says.

Occasionally, you may need to trim down a new workbook using the "Remove all sheets from this book except current" task.

The "Duplicate current worksheet, with changes" task is an experimental feature beyond the present scope.



The "Copy Columns" controls above the column details make it easy to duplicate columns. This also "bumps up" any existing columns by automatically incrementing their column numbers.

# Data processing option 2

# Python modules

The modules are available from:

https://github.com/rickshory/NDVI-modules

Setup:

These modules use the Python programming language, and a cross-platform Graphical User Interface (GUI) package named "wxPython". Both of these are available free online.

The modules are written for Python version 2.7. First, install Python 2.7. When you install the wxPython package, download the installer that matches Python 2.7.

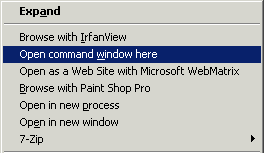
To install the Python language, search online for the version for your operating system.

To install wxPython, go to:

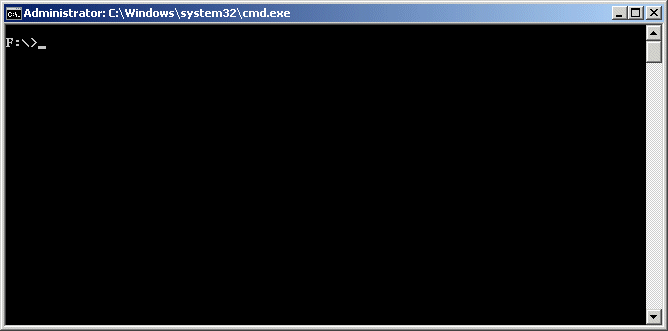
http://www.wxpython.org/

Follow the "Download" link in the menu bar to the left, for your operating system. Download the version of wxPython for your OS, and for Python 2.7.

You run the Python modules from the command line.



In Windows OS, you can open a command window by holding the shift key down, and left-clicking on a folder in a file view window. This would be the folder where you have saved copies of the Python modules. In the pop-up menu, click on "Open command window here". Then, click in that window.



On many Linux systems, like Ubuntu, you can open the analogous "Terminal window" using Ctrl-Alt-T.

The Python modules operate as stand-alone chunks. That is, they will open and run independently. You would not normally use more than one of them at once. For example, you should let the module that imports data complete its work before running modules to output analysis, or the analysis will not have all the data.

The modules' jobs are:

- Put data in: "add\_data.py"

- Set up Stations and Series: "stations.py"

- Mask out bad data: "masking.py"

- Get microclimate analysis out: "datasets.py"

- Get NDVI analysis out: "NDVI.py"

- "scidb.py" is the shared module that contains functions etc. the other modules can use.

The modules will automatically set up a database in the folder where you put the modules. The database is of a format called "SQLite". This type of database is embodied as a single file. In this system, there are actually two SQLite database files, one for the data storage, and one for temporary processing. The storage database is named "sci\_data.db". The temporary database is named "tmp.db". You would not normally rename these files. If you delete, rename, or move "sci\_data.db", the next use of a module with re-create the file, but the new file will not have any of the pre-existing data.

If you want to use the modules for a new data set, the cleanest way to do this is to copy the module files (\*.py) to a new folder and start using those copies. The modules will create a new "sci\_data.db" file for your new data set, in the new folder.

You could also swap in and out of different datasets by renaming "sci\_data.db", or moving this file in and out of the module folder. Whichever file you have named as "sci\_data.db", in the module folder, at the time you start any module, is the one the module will use.

The modules connect to "sci\_data.db" and set up most things automatically. If you have need of looking at the tables in "sci\_data.db" directly, you can use a plug-in written for the Firefox web browser named "SQLite manager". The process for setting up SQLite manager varies with version, so we do not give it here.

You can also look at an SQLite database by connecting to it using MS Access. You can easily find the instructions to do this online, so we do not give them here.

There are 3 general parts to working with a project's data:

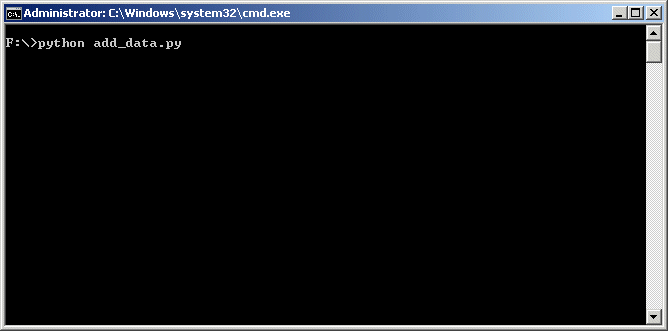
1. Import
2. Setup
3. Output.

# 1. Data Import

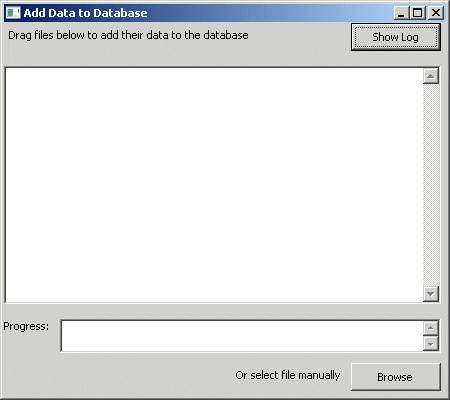
For import, there are two stages. First, converting the native Hobo format files (downloaded from data loggers) to text format. Then, importing these text files into the database.

File conversion is covered in the "Data pre-processing" section above. It is the same for both our processing systems.

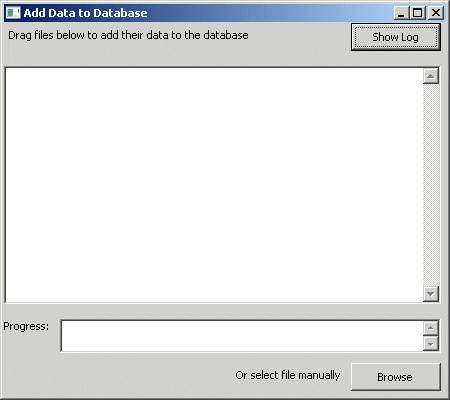
From the command line, run the Python module "add\_data.py".

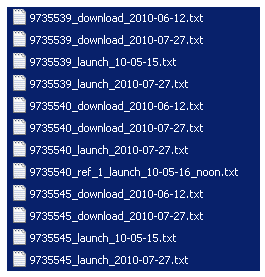


This creates a Graphical User Interface (GUI) window, which embodies this module

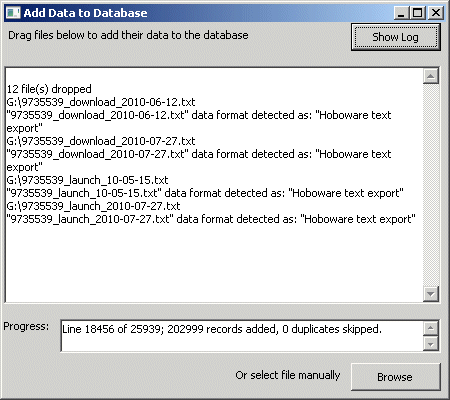


Simply drag the converted text files from a file view window and drop them into the "Add Data" window (you can skip the "log" file).





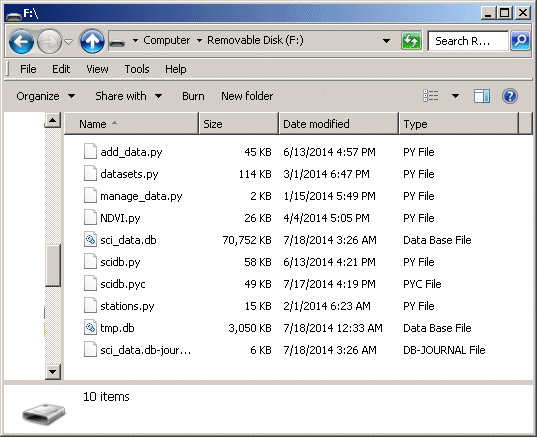
The data processing gives diagnostics as it parses the files and creates records in the database.



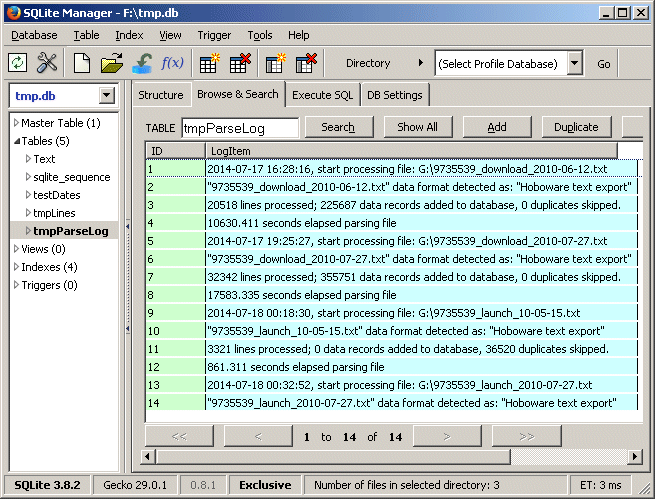
This can take quite a long time, but is automatic.

This is the point to watch out for new HOBOware bugs.

If you look at the module folder, you can see that the processing has created the database files "sci\_data.db" and "tmp.db".



If you use the Firefox SQLite Manager (or another method) to view the database files, you can see the diagnostics from data file import.



These are in the temporary processing database file "tmp.db", in the table "tmpParseLog".

The table records all the files processed, with details including the time each one took. As shown in this example, there were duplicate records in some of the files, but the import process simply ignored the duplicates. So, if you are not sure whether data files have been imported yet, it's safe to try re-importing them.

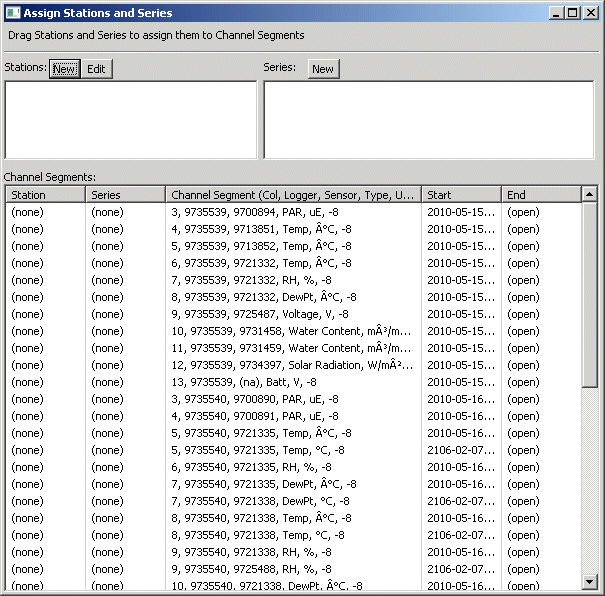
The data import process automatically converts local time to Universal Time (UT) using the time zone information in the text files. After import, data are stored in the database as UT time. Local time is recoverable, if needed, by stored "time zone offset" numbers.

# 2. Data setup:

You set up Stations and Series using the module "stations.py". You run this from the command line the same as the previous module, except using the command:

python stations.py

The GUI window for this module shows the "Channel Segments" (explained below) created by the import process.



The import process organizes the data by "Channels", which correspond to all the different sensors. For example, the top item here was from column 3 in the data files, from logger serial number 9735539, with sensor serial number 9700894, data type Photsynthetically Active Radiation (PAR), having units of microEinsteins, and the timezone offset was 8 hours negative relative to UT.

This organization by Channels allows the system to store data very efficiently. Each item of data is keyed to its Channel, so you can query out all the data for a Channel by this key.

A "Channel Segment" is the time extent of data that we are interested in. Typically, this is all the data for the Channel, the exceptions described below. Notice that the time extent is given only by a start time. An "open" end time means "all the data up till now". This is because the import can easily determine the start time from the earliest data reading from a sensor, but then data readings are added as your experiment continues. The "open" end time automatically includes all the subsequent data.

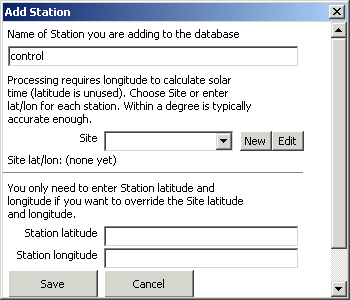
The exceptions are:

* If a sensor died partway through the data collection. Then you could use a specific end time to say data after this did not apply. Often, when sensors fail, the logger keeps recording spurious numbers.
* If you repurposed a sensor. For example if you deployed a temperature probe to track early snow melt, then moved the logger to a new location for the remainder of the season. Then you would set an end time for the first segment, with a new start time following that.

If you have periods of bad data for a Channel, you do not need to chop the Channel up into a lot of segments to cut that out. There is an easier way, called "data masking", covered later. The purpose of Channel Segments is to distinguish general ranges of data you want to treat the same.

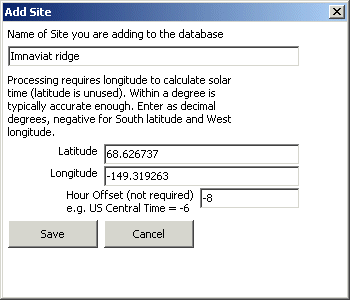
This "stations" module allows you to assign meaningful names to these data ranges, rather than dealing with cryptic serial numbers.

You describe a Channel Segment by a "Station" and a "Series". The Station generally corresponds to the logger while the Series is associated with a sensor.

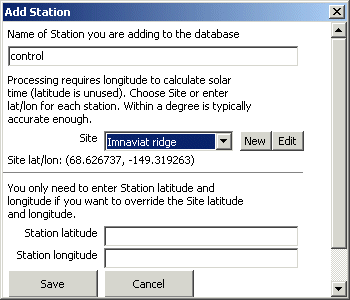
 Create names for Stations by clicking the "New" button next to the Stations label.

A dialog pops up where you enter the station name. In this example, the name is "control".

A Station requires a longitude, to correctly adjust readings for solar time.

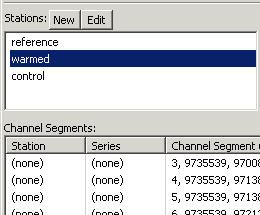


Often, you have many stations in close enough proximity that their longitude is essentially the same. In this case, you can define a "Site".



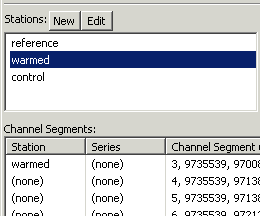
After defining a Site, you can simply select it to use its longitude for a Station.

Latitude is not used in any of the analysis, but fields for its entry are included for completeness.



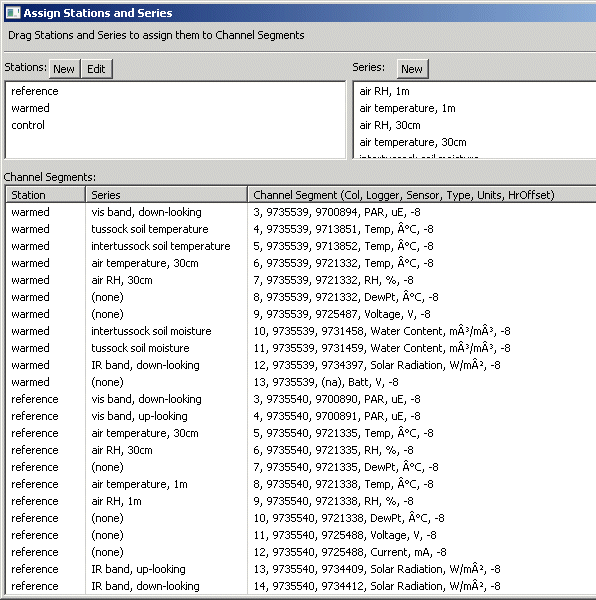
As you add stations, the names go into a list at the top of the GUI window

Select a Station and drag it to the appropriate Channel Segment.



The Channel Segment is assigned to that Station.

Repeat for the other Channel Segments, and assign Series in a similar way. If you make a mistake, drag the correct Station or Series to the Channel Segment and it will be re-assigned.

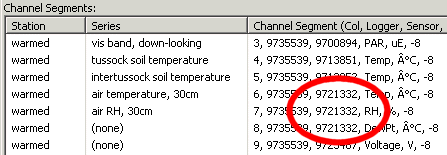


Notice that you do not need to assign anything to Channel Segments you are not going to use, such as "Batt" (the battery voltage) in this example.

You can close the GUI window and run the module again to re-open it. Notice that all changes are saved. This is the advantage of using persistent storage in the form of a database.

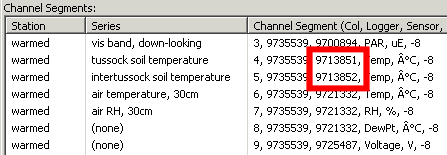
Assigning Series can be a chore, but you only have to do it this once. Some are easy because they are unique, however Onset sensors have their quirks.

For example, three "sensors" here all have the same serial number, 9721332. This is the second 7-digit number in rows 4 to 7 in the graphic above.



This device is a composite sensor for both air temperature and relative humidity (RH). The Channels share this same serial number because they are physically one device. The 3rd Channel, "DewPt" is a derived Channel).

Other channels may differ only by a cryptic serial number. In this study, there are pairs of temperature and water content sensors. They monitor the heterogeneous tundra, one of each pair in a "tussock" of vegetation, the other in the space between tussocks. This kind of thing is a generalized problem, as you often have multiple sensors of the same type plugged into the same logger.



You may be able to determine which is which by using HOBOware, or by the "Data Masking" module described next. HOBOware allows you to view any or all of the channels in a log file, and the Data Masking module has similar visualization.

You can "spike" a sensor, for example making a temperature probe anomalously warm for a brief period. You would keep a record of exactly when you did this, and then later look for that spike in the data, to see what Channel it was on.

Some kinds of sensors do not have a serial number. In this case, you may only be able to distinguish them by the Column number, the first numeral in the Channel text. This is the tabular column in the text log file.

 You seldom need to explicitly add an "End" timestamp, but if you do, right click on the row and a menu will pop up. It opens a dialog to enter or edit the ending timestamp. The dialog will automatically create a new Channel Segment for that Channel that begins when the one you marked ends.

The process of manually setting start and end is not well error checked, because there are so many possible ambiguous conditions (gaps, overlaps, end-before-start, etc.). You must be responsible for any changes you make. If you really need to manually change things, you can do so by directly accessing the database, such as through Firefox SQLite Manager or an MS Access link.

Data Masking

The other important step in data setup is Data Masking. This means ignoring invalid data. We don't usually delete bad data, but just "mask" it out, to ignore it. It's always possible you may want the data back again.

When the loggers in this sample were started, the tundra was still frozen so the soil temperature probes could not be got in. Therefore, they were really recording random air temperature, not valid soil temperature.

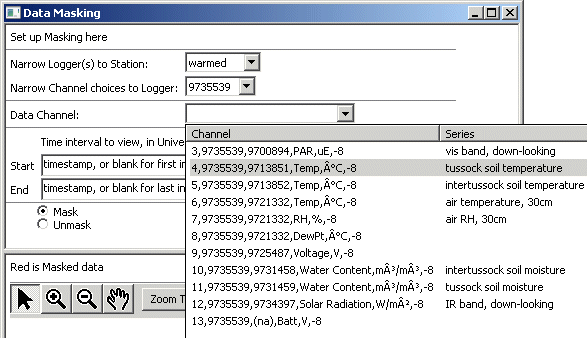
We'll use this case as an example, but the principle would be the same for removing any "spikes" you induced to identify a sensor. Mask out the spike data points and they will be ignored in all analysis.

Each data record has an internal mark to "Use" it or not. If marked False, the record is ignored in all analysis. It's not practical to mark and un-mark these manually, so you use a graphical interface.

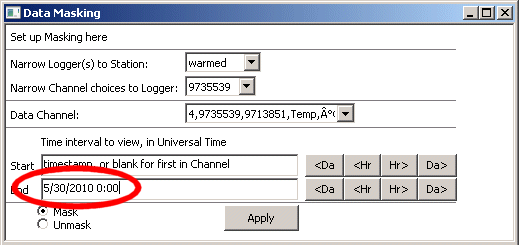
You start this module from the command line using:

python masking.py

Although you must work on masking at the Channel level, you can view the Stations and Series you previously set up to narrow the selections.

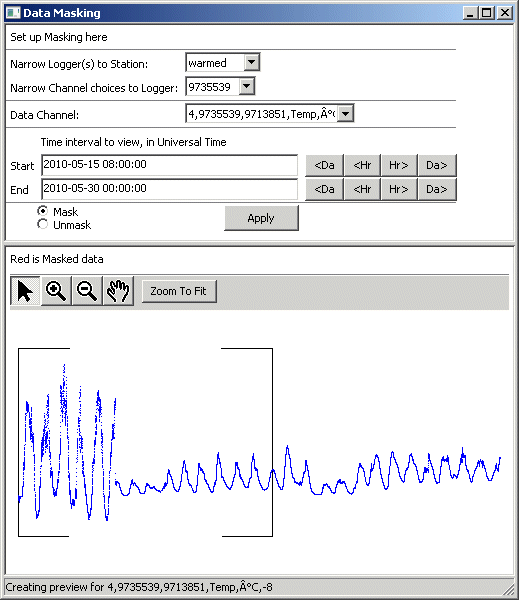


For Start and End, you need to enter at least a portion of the time as well as the date because the date alone is ambiguous. You can use most date/time formats.



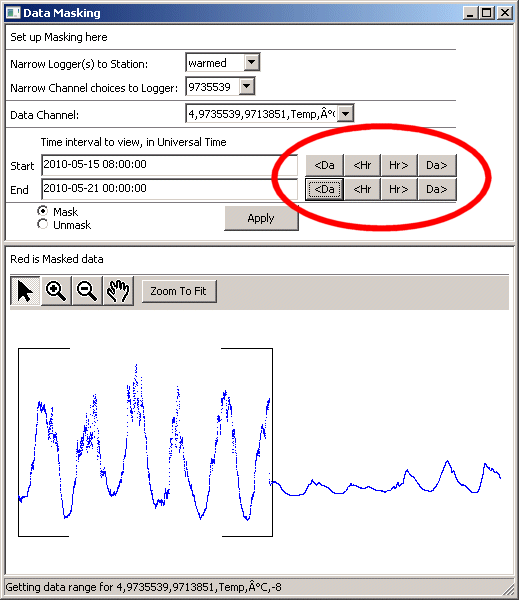
The processing will translate the timestamps into standard format. Remember that the timestamps are in Universal Time, if you are comparing them with your field notes.

When you have valid timestamps, the window shows a preview of the data. The brackets mark the area that would be masked or unmasked.

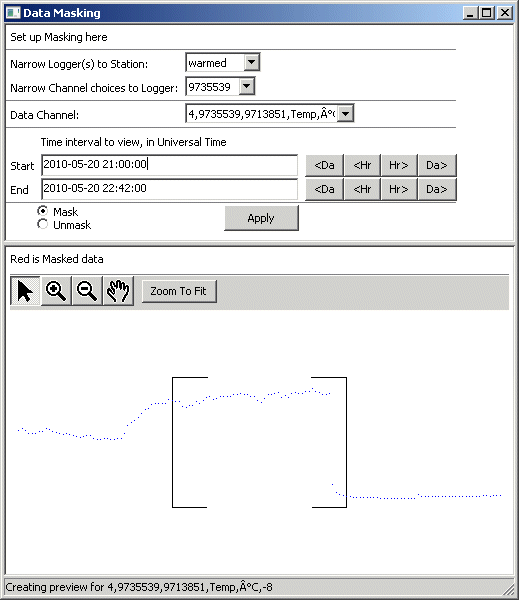


Ideally, you would have some field notes specifying bad data, because it is not always as obvious as this example. Here, the irrelevant data shows the wild swings you would expect from a temperature probe lying on the ground surface exposed to the sun. When the tundra was thawed enough, the probe was inserted and the temperature excursions became much more moderate.

To home in on the time period of interest, you can use the Hour and Day up and down buttons to adjust the active area, without having to explicitly type in timestamps.

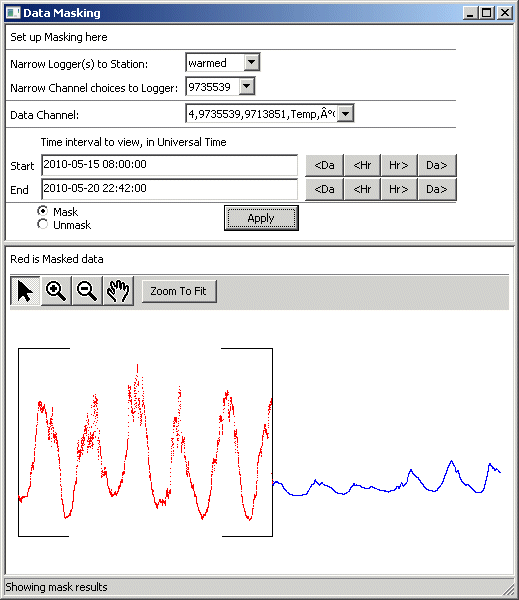


You can zoom in to extreme detail, to define one end of a mask range, then change the other timestamp to zoom out for the other end.



You can go to the limits of the data by blanking one timestamp or the other. For example, if you erase the Start, as soon as you leave that text box, the form will fill in the earliest value.

When you have the correct range selected, click the "Apply" button. The display updates to show masked data in red.



 If you make a mistake, you can change the option button to "Unmask" and Apply again. The points will turn blue to show they are no longer masked.

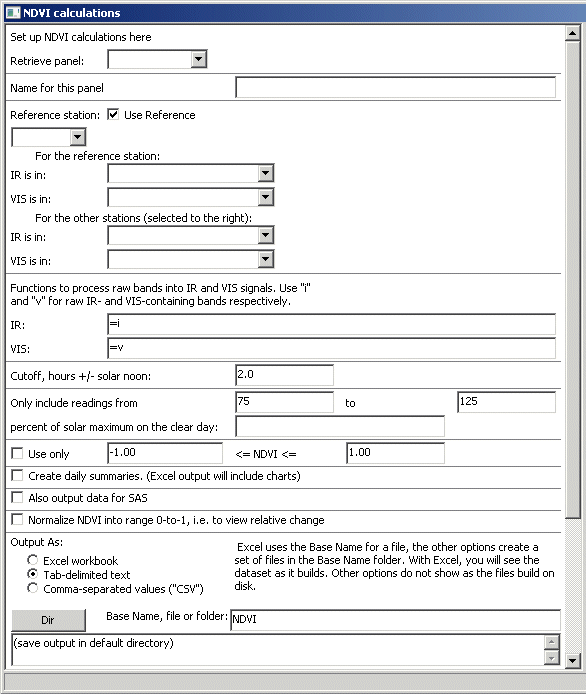
You can apply masking and unmasking in pieces because all it does is toggle markers in the underlying data records.

# 3. Data output:

* NDVI datasets
* Other datasets

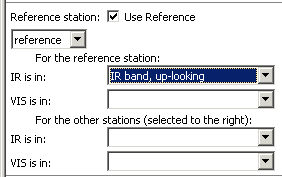
To get NDVI datasets, start the module "NDVI.py" from the command line:

python NDVI.py

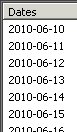


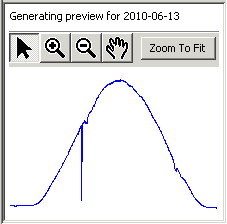
The first step of setting up an NDVI dataset is finding a clear day to use for a reference.

In the upper part of the form, select a Station and a Series.

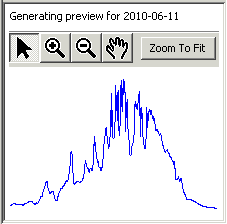
 The "Use Reference" checkbox is unimportant at this time.

The Station you choose should be one that recorded irradiance, and the Series should be one of those. An up-looking sensor is preferable since the higher numbers will have more resolution, but any irradiance sensor will work.

 With this Series selected, float the cursor over the list of dates on the right side of the form.

 As you float over a date, a preview will appear below. This is a chart of the irradiance Series for that date.

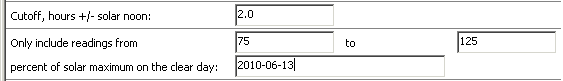
A clear day has a characteristic bell-shaped curve, as in the example for 2010-06-13 to the left. There was only one brief interruption in late morning, perhaps from a shadow briefly falling over the sensor.

 By contrast, a cloudy day has a jagged curve, as for 2010-06-11. Sometimes, portions of the bell shape may be evident, but there are broad excursions.

On partly cloudy days, there will be peaks much higher than the base level. When the sun is shining on a sensor, and there are bright clouds in the sky, the total irradiance will be higher than from the sun alone.

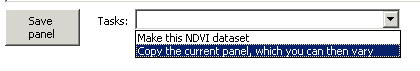
The purpose of finding a clear day is to establish a typical clear-sky solar maximum for the site. Our algorithm for extracting the "best" data is to use only irradiance levels close to this. The more the irradiance levels differ from this, both by time of day and by atmospheric factors, the more spectrally anomalous they tend to be.

The details of specifying "close to" are covered in other parts of the form, but for now enter the best clear day you find in the field labeled "percent of solar maximum on the clear day:". If you have a choice of good days, ones closest to summer solstice are preferred.



If the bell curves of irradiance traces are not well centered, make sure you have the correct longitude entered using the "stations" module.

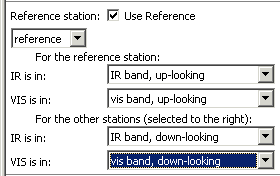
Before proceeding further, let's look at some form mechanics. As you enter these parameters, you are building up a "panel" of settings.



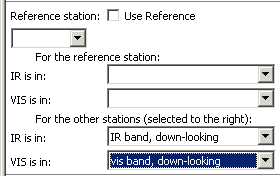
At any point, you can click the "Save panel" button at the bottom of the form. If some parameters are invalid, the form will substitute reasonable defaults as placeholders. You can then retrieve the panel using the selector at the top of the form.

Users often wish to create panels similar to an existing one. The form provides a one-step mechanism to copy a panel, without having to enter all the parameters from scratch.

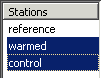
Returning to the parameters:

 If you are using references, select the infrared and visible bands to use for these, as well as the infrared and visible bands that contain the station data.

If using references, the processing pre-scales the station data by the reference bands, which better compensates for changing weather conditions.

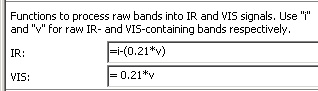
 If you are not using references, you can leave the selectors for the Reference station blank, both for the station itself and the bands. If the "Use Reference" checkbox is un-checked, the processing will ignore these whether or not they are filled in.

You would skip using references if you did not have reference data, or if your reference data were very bad.

 These "other" stations refer to the list on the far right side of the form. Here you choose the Stations you want charts for. Notice that we do not include the Station named "reference". It serves as the reference for the charts, but we do not need a chart from it. That station did happen to have down-looking sensors, but they were over a random patch of tundra, not any experimental site.

A Station could include both reference sensors and data sensors. In this case you would select it in the Stations list, as well as selecting it as the Reference Station. It would then serve as the reference for itself, and potentially other Stations as well.

The bands you select are the raw bands, which usually require adjustment. For example, if using Onset sensors, the "visible" band measures Photosynthetically Active Radiation in units of microEinsteins (photon count) while the "infrared" band measure Solar Radiation in units of watts per square meter (energy).



To put these two sources into approximately equal units, multiply the raw "visible" band by 0.21. The Solar Radiation signal includes both visible and infrared. To derive a surrogate infrared-only, subtract out the visible component.

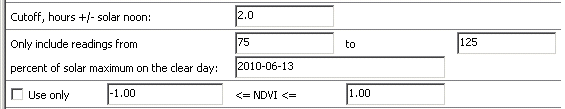
The adjustment formulas for other types of sensors, such as photodiodes, would be very different, so the module allows any generalized functions. You would use the mathematical syntax that corresponded to the output format, either Excel or text file. These are often the same, but they may not be. For example "two raised to the third power": in Excel this would be "=2^3". For text file output, use the Python syntax "=2\*\*3".

If you have different kinds of loggers in your study, such as some with Onset sensors and some with photodiode sensors, you should set up separate panels for each because the band adjustment functions would be different.

The next part of the form allows you to use only the best data.

The "Cutoff hours..." section limits data to a time window before and after solar noon. The default is two hours plus-and-minus, for a window of four hours total. You must have the longitude entered correctly, using the "stations" module, for the processing to determine solar noon.

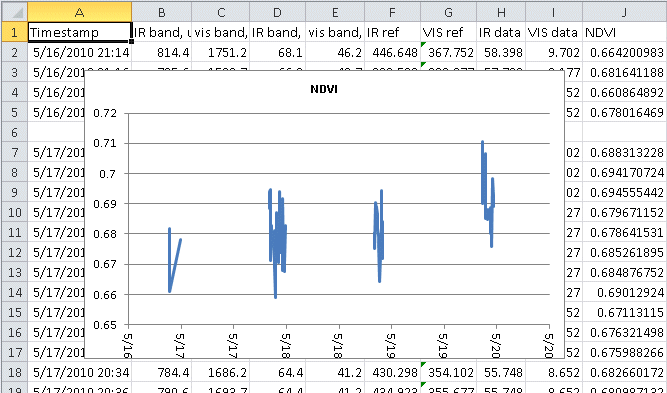
The "Only include..." section limits data to those values within the given thresholds above and below clear sky maximum. The default is from 75% to 125%. The processing applies these limits to each of the raw bands individually, based on that band's clear sky maximum. The processing applies this before any calculations, so out of range values in any band will veto all calculations for those time points.



The "Use only" section lets you limit the resulting dataset to only those with NDVI in a certain range. The processing makes these decisions after calculating NDVI for a time point.

These limits apply to the default data set. The processing always creates a default dataset, but there are options to create other datasets as well, as detailed below.

The default dataset is a tabular format that includes all the NDVI time points. If your system has Excel, you can output Excel workbooks, otherwise you can use text format as explained in the next section.



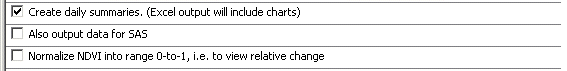
After the timestamp, the first four columns are the raw bands, the next four columns are the bands adjusted by the calibration functions, and the last column is the NDVI. If you are not using references, some of these columns will be empty, and the calculations will be varied as appropriate. If using Excel, the processing will create a chart, for a visual look at the results.

If using Excel, the cells, where possible, will contain formulas. Text output cannot, of course, use formulas so the tables will be all numeric.

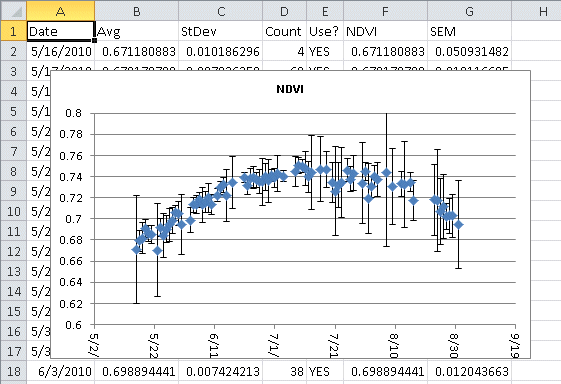
This example is only a small sample; a few days for one station. We suggest you start with a small sample, to make sure your adjustment formulas are working correctly.

A line-by-line dataset like this is what you get if you do not mark any of the other output options. Often, users have wanted further variations.

The "Create daily summaries" option will compile the values by date. In Excel, these will be in an additional worksheet. As text output, these will be additional files.

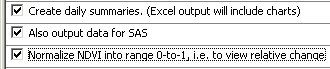


The summaries contain a column of the average NDVI for the date, the Standard Deviation, the Count of valid values, and a decision column ("Use?") based on whether there are enough values, more than 1 (if only one value, Standard Deviation is undefined). If valid, the NDVI average is given again, and the Standard Error of the Mean (SEM).



If using Excel, the processing creates a chart with error bars, which are SEM.

The "Also output data for SAS" option creates an additional dataset in a simple format for statistical analysis. In Excel, this will be a separate workbook. As text, these will be additional files.



The "Normalize NDVI..." option adjusts the NDVI into the range of zero-to-one, to make it easier to see trends. In some cases, such as if you have no reference bands, this may be your most useful output. If your data is such you cannot calculate true NDVI values very well, at least you can see the variations.

The "Normalize..." option interacts with the other two. The basic line-by-line output is never normalized. If you use the "...summaries..." option, the processing will insert new columns in the dataset and (in Excel) chart the normalized values.

It's easiest to see how this works by running a few samples. However, there are two general ways of doing normalization.

The charts below are from the SAS data, which will be normalized if that option is marked. On the left, normalization is applied based on the line-by-line values. After normalization, all these values fall in the zero-to-one range.

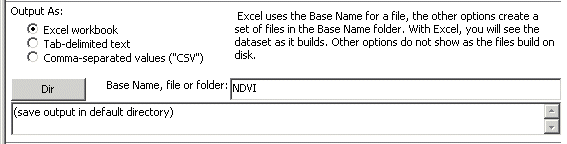
The chart on the right is normalized based on the daily averages. This means the daily averages are normalized to fall in the zero-to-one range. After that, the line-by-line values are back-calibrated based on the ratios needed to make the averages 0-to-1. In this case, the line-by-line values can fall outside the 0-to-1 range, because they may have been above and below the resultant averages derived from them.

|  |  |
| --- | --- |
|  |  |
| Normalized ignoring daily averages | Normalized based on daily averages. |

Again, running a few samples will make this more clear. The setting of the "...summaries..." checkbox determines the type of normalization. If "...summaries..." is un-checked, there are no averages, and the normalization is as on the left. If "...summaries..." is checked, the normalization is as on the right.

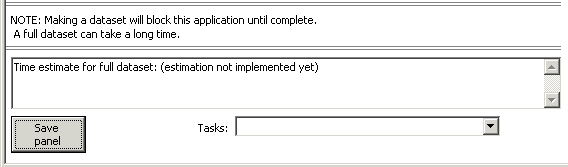
You have a choice of output file formats. Under Windows, if you have Excel installed, you can create Excel workbooks. On operating systems (Mac, Linux) that do not support this, you can get your data as text files. The two options for text files are tab delimited and comma-separated.

Where table cell contents would be formulas in Excel, they are numerical values in the text output. Excel output creates workbooks, and the individual data sets are worksheets in these workbooks. In text output, the individual data sets are distinct files, created in a folder which has the Base Name.



If you do not explicitly select an overall folder (directory) for output, the processing will save files in a default folder based on your operating system and settings. This may be cryptic, so we suggest you browse to a known location.

As you run a data set, the text box that originally says "Time estimate..." will change to show progress. The modules are all open source, so if you wish to vary them, you can do so, including developing an algorithm for estimating how long a data set will take.



As mentioned before, we suggest starting with a small sample dataset before running one for all dates and all stations.

"Microclimate" datasets:

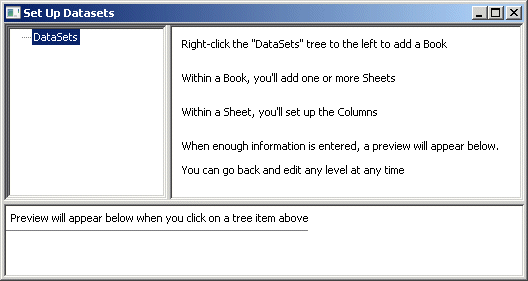
Users have wanted datasets from the other data Series as well as the irradiance Series used for NDVI. We call these colloquially the "microclimate" data because they are often from weather station sensors specific to a small site. However, they really could be any data. You generate these generalized datasets using the "datasets" module. At the command line enter:

python datasets.py

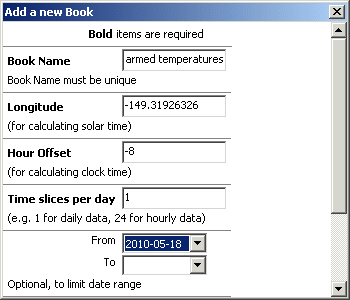
You use a three-tiered conceptual structure of "workbooks" that contain "sheets" that contain "columns". If you will be outputting them as Excel, they will be actual workbooks. If you are using text format, they will be a collection of files, but the concept is the same.

The following will take you through examples that illustrates most everything the "datasets" module can do.

You build your datasets as a tree structure. When you have entered enough information, the module window will begin showing previews, to help you make sure you're getting what you want.



 Begin by right-clicking on the "DataSets" root tree item. A menu item "Add a Book" pops up.

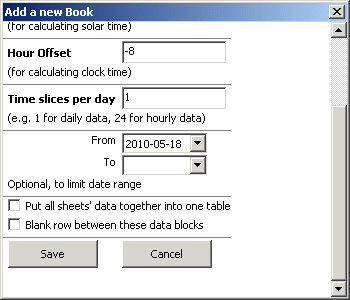
 When you click that, a panel pops up where you put in the overall parameters for your "workbook".

The example name, here "warmed temperatures", will scroll within the text input box as you type.

You manually enter the Longitude and the Hour Offset. Longitude adjusts the data's Universal Time timestamps to local solar time. Environmental readings most reasonably vary by the solar day. Hour Offset adjusts the timestamps to local clock time for a more intuitive display.

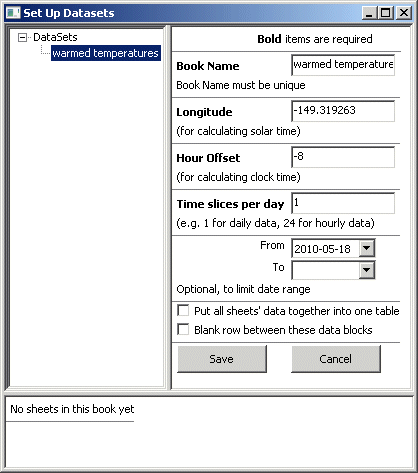
"Time slices per day" is a generalized way of dividing the data into the time blocks you need. For example 1 time slice per day gives you daily data. Another popular value, 24 time slices per day, gives you hourly data.

You can use the "From" and "To" selectors to can set any limits on the temporal range of the data. In these selectors, only dates appear that have some data. Leaving "From" blank means "from the first, and "To" blank means "to the last".

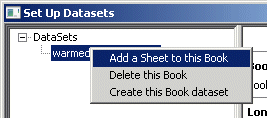


In the lower part of the panel are the seldom-used options.

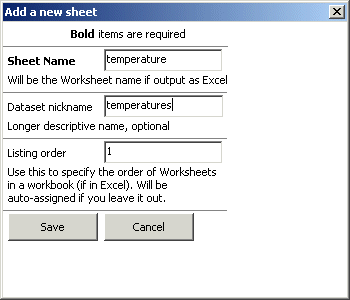
When done, click "Save". You will be able to edit the panel later if you wish.



After you have saved a "Book" its parameters are available to edit when you select that Book in the tree.

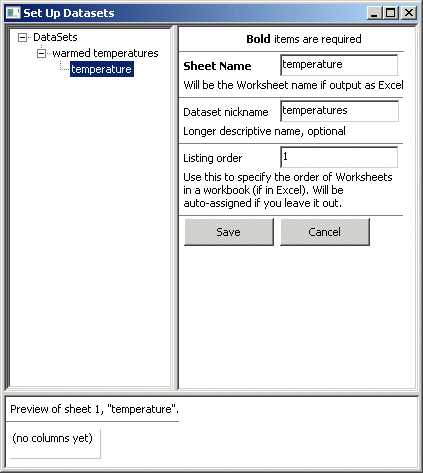
 To add a Sheet to this book, right-click on the Book item in the tree, similar to what you did to create the Book.

Pop-up menus always show the options that apply to the item you right-clicked.



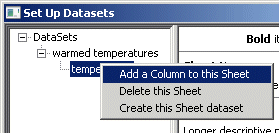
The Sheet parameters are simple, essentially only a name and a listing order.

Listing order mainly applies to Excel workbooks, where it controls the order of the worksheets in the workbook. In text output, the Sheets will be discreet files in a folder, where listing order depends on how you sort them.

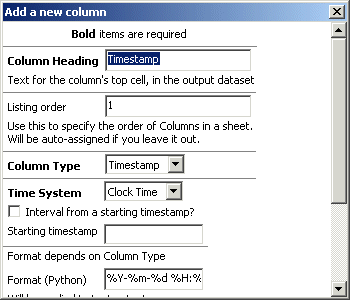


A saved sheet panel appears in the overall module GUI when you have that item selected in the tree. As for the overall book, you can edit it here and save changes.

As you add columns, a preview will appear in the lower part of the GUI.

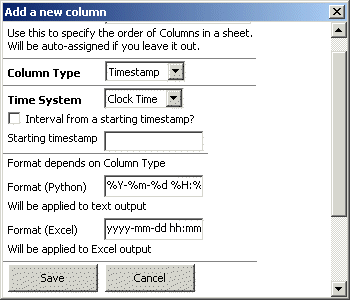


Again, the right-click pop-up menu is available. Use it to build columns for the sheet.



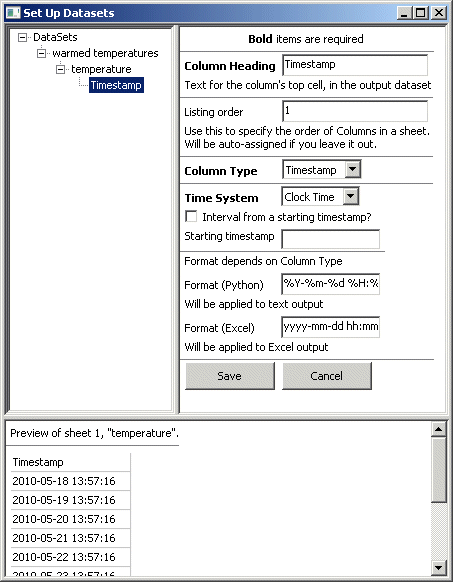
The column panel tries to "guess" some things. For example, at the first empty column you add, the panel offers "Timestamp" as both the Column Heading and the Column Type.

This is based on the assumption that most datasets will have some sort of time structure. You can edit these, or override the defaults, and you can add additional timestamp columns.



In the lower part of the panel, there are two different format fields. "Excel" format will be used if you have Excel and create the dataset as an Excel workbook. "Python" format is a generalized text format that will be used for output other than Excel.

Since both of these format types can be intricate, the panel offers one of the most elaborate for each. You can then "edit it down" to simplify it if you wish.

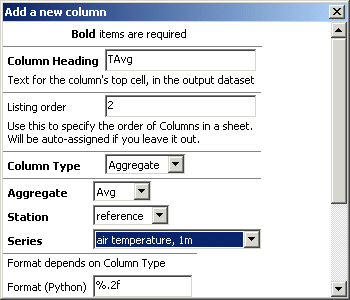


After you have at least one column Saved, the GUI shows a preview in the lower part.

The preview shows the first ten lines, if there are that many.

The times shown in this example are clock times adjusted from Universal Time to the time zone specified by the Hour Offset of the Book, which was -8. If you wish to adjust to a different time zone, change this number. If you want the times to be direct Universal Time, set the Hour Offset to zero.

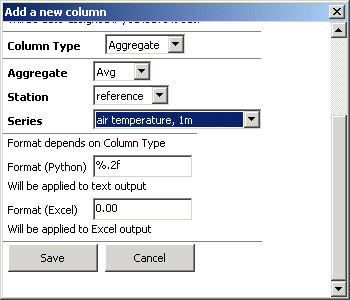
You may prefer Date and Time in separate columns. The best practice is to set up two separate columns that are both of Column Type "Timestamp". Then, format them differently, one for date and one for time. This will assure your data sort correctly, regardless of which date/time column you use.

 After the first column, the panel offers "Aggregate" as the default for Column Type. This is easier to "edit down" to Constant, for example, than to write from scratch.

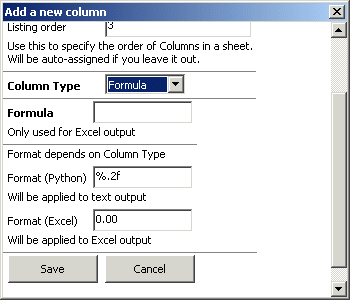
(A "Constant" Column Type is just a value you want to appear in each row of the column. You can use it to label datasets.)

You can select Station and Series, using their meaningful names.

You can also combine multiple series into one column by assigning them the same Column Heading, Listing order, Aggregate, and Format. All aggregation, including this combined aggregation, is "database style". This means the aggregation is over all available values. For example, if you combine a Series that has 40 values with another that has 10 values, and the aggregation is an Average, the average will be over the 50 total values, and will be strongly weighted towards the first series.



The lower part of the panel offers the two options for formatting. This is similar to the Timestamp formatting. You can try different formats. The preview will show the Python format. To try the Excel format, make a sample of the dataset.



You can use the Column Type "Formula", but at this time it is only supported for Excel output format.

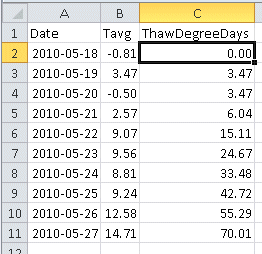
This inserts a formula into a spreadsheet cell. The formula can refer to cells in other rows and columns.

For example, here is a simplified formula for calculating cumulative thaw degree days. You write the formula in "RC" notation, which means relative to adjacent Rows ("R") and Columns ("C").

The formula is:

=IF(RC[-1]>0,RC[-1],0) +IF(ISNUMBER(R[-1]C),R[-1]C,0)

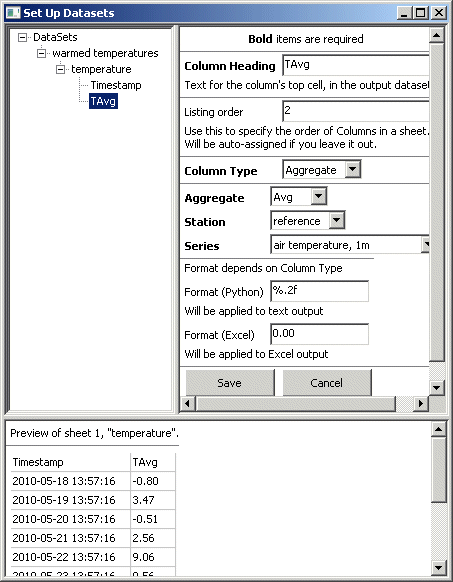
It may be easier to see what's going on from the format Excel translates it into:



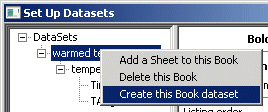


This is the formula that ends up in the cell selected in the illustration to the right. Essentially, the formula sums the cell above it (C[-1] or "column minus 1") and the cell to its left (R[-1] or "row minus 1"). The "IF B2 > 0" term is needed to ignore any negative temperatures. The "IF ISNUMBER ..." term takes care of the first row, which will have only the column heading above it, instead of a valid number.

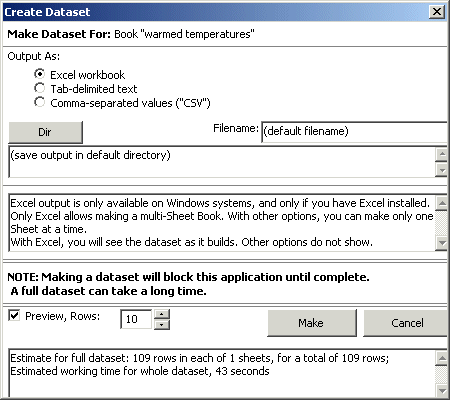
The module GUI can't evaluate Excel formulas, so the preview will only show the literal formula as text.



As you add columns, they preview at the bottom of the GUI.

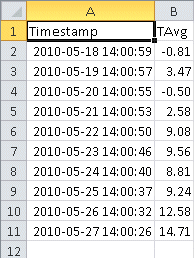


To create the actual dataset, use the pop-up menu items of the "book" item in the tree.

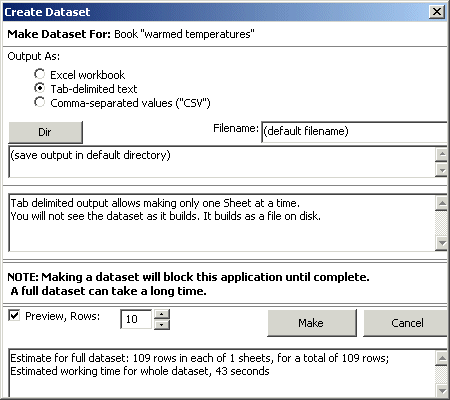


The panel tries to help you out by offering some notes on what to expect. Here, with "Excel workbook" output selected, there are messages about what this option entails.

By default, the output dataset will only contain a preview of the first ten rows, until you explicitly un-check this.

 If you do not give a filename the processing will create the dataset using the book name as the file name. If you do not select a folder ("Dir"), the file(s) will go into the "user" folder, wherever that is in your operating system. This may be cryptic, so we suggest choosing a specific location.

For Excel output, the dataset will be a single file, an Excel workbook. It will have one worksheet for each "sheet" in the book.



For the output formats other than Excel, the dataset will be a *set* of files, created in a folder having the book name.

Each file in the set will be what's specified by the corresponding "sheet" in the "book".