Atmos. Meas. Tech. Discuss., 7, 3645–3679, 2014
www.atmos-meas-tech-discuss.net/7/3645/2014/
doi:10.5194/amtd-7-3645-2014

Atmospheric Measurement Techniques Discussions
Open Access

Software and database structure to analyze the relationship between aerosol, clouds and precipitation: SAMAC

S. Gagné[1,2], L. P. MacDonald[1,*], W. R. Leaitch[2], and J. R. Pierce[1,3]

[1]Department of Physics and Atmospheric Science, Dalhousie University, Halifax, B3H 3J5, Canada
[2]Environment Canada, Downsview, Toronto, M3H 5T4, Canada
[3]Department of Atmospheric Science, Colorado State University, Fort Collins, CO 80523, USA
[*]now at: Department of Atmospheric Science, Colorado State University, Fort Collins, CO 80523, USA

Received: 30 January 2014 – Accepted: 17 March 2014 – Published: 10 April 2014

Correspondence to: S. Gagné (stephanie.gagne@dal.ca)

Published by Copernicus Publications on behalf of the European Geosciences Union.

Abstract

The analysis of aircraft-based measurements of clouds are critical for studies of aerosol and clouds. Many such measurements have been taken, but it is difficult to compare such data across instruments, flights and campaigns. We present a new open-source software program, SAMAC (Software for Airborne Measurements of Aerosol and Clouds), that may enable a more generic approach to the analysis of aerosol-cloud-precipitation data. The software offers a faster and standardized approach to the analysis of aircraft measurements of clouds across campaigns. SAMAC is an object-oriented software program in which a cloud is an object. The cloud objects come with built-in methods and properties that allow for the quick generation of basic plots and calculations, which provide a quick view of the data set and may be used to compare clouds and to filter for specific characteristics. Other researchers are encouraged to apply the software to their data and to contribute to its improvement. SAMAC can be downloaded at https://github.com/StephGagne/SAMAC/releases.

1 Introduction

The links between aerosol, clouds and precipitation particles is often referred to as one of the least understood, and one of the most important processes by which humans affect the Earth's climate (IPCC, 2007). Aerosols, clouds and precipitation are described in Global Climate Models (GCMs) using parametrizations (e.g Nenes and Seinfeld, 2003; Morrison et al., 2005). The parametrizations are validated and constrained using measured data, remote sensing or direct airborne measurements (see e.g. Zhang et al., 2013) and can even be developed from observations (e.g. Boucher and Lohmann, 1995). Measurements of aerosols, clouds and precipitation in concert are crucial to better constrain global climate models. Satellite observations are very helpful in the study of aerosol-cloud-precipitation interactions because they cover large and diverse geographical areas, but they are not without some biases (e.g. they generally cannot sample co-located aerosols and clouds). Therefore, airborne in-situ measurements are a necessary part of the improvement of the aerosol and cloud processes in GCMs.

Airborne measurements of aerosol-cloud-precipitation interactions have been performed on different cloud types, and in different areas around the world (see e.g. campaigns such as RICO (Rauber et al., 2013), VAMOS-VOCALS (Wood et al., 2011), or MASE (Lu et al., 2007)). In many cases, these measurements were performed using an aircraft tracing a single path through clouds (e.g. Leaitch et al., 1996, Johnson et al., 2003, Schmid et al., 2003; Sollazzo et al., 2003; Kleinmann et al., 2012). This measurement method makes data analysis difficult because the relevant data are measured at different time intervals and at different places in 3-D space and subject to variability in the conditions at the time (e.g. clouds may be in different stages of their life cycle during sampling). Further, the aerosol effects, while important, may be subtle relative to other aspects of the cloud life cycle.

Another difficulty of airborne measurement is the lack of standardized instrumentation, data processing and analysis. While, during a given measurement campaign, all the clouds are likely to be measured with the same set of instruments, it may not always be possible to perform the same calculations for all of them. For example, there may be a vertical profile available for one cloud, but not for the other. From one measurement campaign to another, the data format and set of instruments used differ widely, especially across research groups. While we cannot standardize instrumentation across research groups and time, we can try to standardize data format and improve the ease of processing.

The creation and use of an open source standardized database structure and software will allow researchers from different institutions to compare their measurements with those of others more easily. Moreover, a basic quantity (e.g. the liquid water path, LWP) could be calculated using exactly the same technique, making the quantity more comparable across various clouds, campaigns, and in the literature. Such software could also improve analysis speed by producing basic plots and calculations so that

the analyst can visualize multiple aspects of the measured clouds rapidly and then decide on the next analysis steps to take.

For this kind of software and data structure to be efficient, it should be freely available for all researchers and easy to modify to fit their needs. The software should hence be written in a free, cross-platform programming language and be open-source. The advantages of the open-source release of software under version control are:

a. it can be modified by the users in order to better fit their needs;

b. when a user makes a modification that could be useful to others or that makes the software more flexible, the improvement can be included in the next public release;

c. when improvements are made to methods (a predefined routine), version control ensures that the old method can still be retrieved and used if needed. This way, calculations are reproducible (as long as the software version is mentioned in publications);

d. although there is a need for a main programmer overseeing the quality and release of new versions of the software, participative programming under version control means that improvements can be made gradually by users who need these improvements, sharing the effort between the main programmer and the users.

In this paper, we present a data structure for cloud measurement data and analysis software, SAMAC (Software for Airborne Measurements of Aerosol and Clouds), that attempts to answer the concerns and fulfil the specifications mentioned above. A first version of the software has been released for download on GitHub. We first give an overview of SAMAC's capabilities after which we cover the thought process and philosophy behind this database and software. We then give instructions and specify the requirements to use and modify the software. Next, we describe the structure and functions in the software. The data structure and associated software were built to be flexible, allow for new subroutines or functions, accommodate many types of instruments, and be easily modified in order to add and handle data from new instruments. Finally, we illustrate the uses of SAMAC with IPython, an interactive python shell, and through data from the Canadian SOLAS 2003 campaign (Gagné et al., 2014).

2 Overview of software capabilities

SAMAC has a number of methods (routines) that allows users to rapidly visualize their data, execute standard operations, and add practical information to their data. In this section, we present a brief overview of the software's features. (More detailed information can be found in the subsequent sections, along with the description of more methods.) To use this software, the measurements must be filled in the cloud object structure of SAMAC. Guiding software is available with SAMAC but may need to be modified to fit different data/file formats. Once the data is placed in the cloud structure of SAMAC, many time-saving methods become available to users.

SAMAC comes with a set of figure-generating methods that include time series, vertical profiles and adiabatic liquid water content. Some examples from these plots can be found in Fig. 1. A figure, or many figures in certain cases, can be generated using only one line of code that calls the plotting method. The figure-generating methods that are available in this version of the software include a time series, maps of the aircraft trajectory, various size-distribution figures, and different types of vertical profiles (with altitude as the *y* axis). More figure-generating methods may be added to the next versions of the software, depending on the users' needs.

This software also comes equipped with a suite of calculation methods. For example, with a single-line command, a user can get the liquid water path, the average size distribution within a chosen aircraft manoeuvre or flight leg, an average or integrated precipitation amount, an average cloud droplet number concentration, and more. As we show in Sect. 5.2, it becomes very easy to make plots that aggregate results from many clouds, controlling for certain characteristics.

There are also a few interactive methods. One category of method is based on the user clicking on a figure to provide information to the software. To define the time period of a manoeuvre (we define a "manoeuvre" as a part of the flight that traces a horizontal line in cloud (horizontal manoeuvre), a vertical profile, a flight below or above clouds), the user can either determine the times and insert them manually into the structure of the cloud instance, or use a specially designed function and click on the beginning and end time of the manoeuvre on a time series plot (top panel in Fig. 1). The same principle can be used to define the base and top of a cloud. The clicking principle is also used in a method created to help mask unphysical, erroneous or dummy data (Fig. 2). In this method, the points within a rectangle defined by the user become masked after receiving confirmation. Another category of interactive methods require the user to answer to questions on the prompt. For example, there is a method to assess the quality of liquid water content profiles that asks the users to enter quality flags for the profile.

Overall, we believe that SAMAC will save time for the users, who will be able to use the ready methods. We also believe that, by establishing an open-source software, users will be able to increase the transparency and reproducibility of their work. Moreover, using the same methods across the research community will help make results more comparable and give more consistent results to modellers.

3 Programming techniques

We created and developed SAMAC in order to process measurements of clouds spanning over 30 years. The measurements had taken place on several different measurement campaigns with different cloud types and instrumentation. In order to compare the different clouds with each other, a standard structure and standard calculation techniques had to be implemented. The structure had to be flexible enough to allow adding new data in case some instruments' data had not been foreseen or were not in the original data set. Another important feature, was that if the original data had to be corrected or re-calibrated, all the related calculations would be corrected automatically: the user wouldn't need to track down the repercussions of this change, instead it would be done by the software itself.

It was decided that object-oriented programming would be the best way to satisfy all these conditions. We define *cloud object* as the overall class, or type of object, we used to represent measured clouds, and a *cloud instance* as a particular individual cloud of the cloud object class. Each cloud instance of a cloud object is a measured cloud in which the data (e.g. time, altitude and cloud droplet concentration), and a few basic subjective cloud characteristics (e.g. environment type and cloud type) would be stored in a structure. Object-oriented programming also allows for methods associated with an object. All cloud instances share the data structure and methods associated with cloud objects. Methods are procedures that can be performed on an object (or more precisely, on the data in each instance, Beazley, 2009). The method can either change the instance itself or simply use the data in the instance to return an output.

We have chosen Python as a programming language for SAMAC mainly for these reasons:

a. Python is a free, open-source, cross-platform, high-level programming language;

b. Python is already used among researchers in the Earth sciences (Lin, 2012);

c. Python has a very clear, readable syntax (Python Programming Language, 2013);

d. New scientific programming packages in Python are released on a regular basis (Yarkoni, 2013)

e. Python's NumPy modules (Numpy, 2013) provides Masked Arrays (Masked Arrays, 2013), which allow for the masking of unphysical or missing/dummy data without having to delete or overwrite them with flag values. Masks are preserved through calculations (masked inputs give masked outputs);

f. Python's object-oriented programming allows for many types of methods and properties bound to the object.

g. Python's SciPy modules (SciPy, org, 2013) particularly NumPy, Matplotlib and Pandas are similar to Matlab and R which are two additional programming languages that are widely used in the Earth sciences. Python also has an interface module with R: RPy (RPy: A simple and efficient access to R from Python, 2013).

The software as released in its Beta version was tested using Python 2.7.1 with Matplotlib 0.99.3, NumPy 1.5.1, SciPy 0.8.0, and Basemap 1.0.1 on Linux Mint 11; with Python 2.7.3 with Matplotlib 1.1.1, NumPy 1.6.2, SciPy 0.10.1, Basemap 1.0.5 and h5py 2.0.1-2 on Linux Mint 14; and with Python 2.7.3 with Matplotlib 1.3.1, NumPy 1.8.0, SciPy 0.13.0, Basemap 1.0.7 and h5py 2.2.0 on OSX 10.9.1 (Build 13B42).

Because the software was designed to handle clouds flexibly (with different instrument types and number of instruments, different aircraft trajectories and series of manoeuvres, etc.), its use by other research groups is possible and encouraged. Once the data is placed inside the cloud instance, the methods can be applied. The use of SAMAC by other research groups should save them time on data analysis and make different quantities more comparable between research groups. Moreover, shared open-source software makes data analysis more transparent and possible mistakes are more likely to be spotted and corrected (Barnes, 2010; Merali, 2010).

SAMAC 0.9.2 (Beta) can be downloaded at https://github.com/StephGagne/SAMAC/releases by clicking on the *source code* button in your preferred data compression format. Upon download, make sure to read through the Documentation material in the package or in the wiki pages, as it includes a description of the SAMAC cloud object structure, a description of the methods associated with cloud objects, and a guide on how to create and populate a cloud object. Git is used for version control. To participate in improving the software, a user needs to have an account on GitHub, pull the latest version from the repository (https://github.com/StephGagne/SAMAC) by forking the repository. The users can make changes to their copy of the repository without affecting the main branch. When the contribution to the code are tested and ready, the users' new code can be merged into the main branch by sending the original author a pull request. For more options and detail, a GitHub tutorial is available at https://help.github.com/. Use the *Issues* section to report any bugs you may discover, or to suggest any improvements. SAMAC falls under the protection of the GNU General Public License version 3.

4 Object structure and associated methods

4.1 Structure of a cloud instance

As discussed in the methods section, to compare large amounts of clouds with different characteristics, we need a standard but flexible data structure. In this section, the structure of the cloud object, ways to add to the structure, as well as current methods associated to cloud objects are discussed. The basic structure of a cloud object is described in Fig. 3.

The structure's core section is the *Basic Aircraft Data*. This section includes the time series from all instruments sharing the same time stamp as the aircraft data (with exception of the size distributions). The *Basic Aircraft Data* section could contain, for example, time, altitude, latitude, longitude, and also other quantities like liquid water content that happen to share the same time stamp. The titles and units of the data are also stored in the structure so that it is possible to scan the data for specific quantities and keep track of the units. The methods are programmed to recognize a number of titles and units combinations when looking for certain quantities. If users have data that do not share the same time stamp as the *Basic Aircraft Data*, these data can either be interpolated to a common time stamp or be placed in a section called *Extra Data*. When the data needed is not found in the *Basic Aircraft Data* section, the software scans the *Extra Data* section for that data, and uses it if found. If the same data title is found in both sections, the data found in the *Basic Aircraft Data* is used.

Another important section is the *Size Distribution* section. This section is designed especially to handle size distribution formats and representations, and is separate from the *Basic Aircraft Data* section. It stores the aerosol, droplet, or hydrometeor

concentrations as a function of time and size. The time and the average bin sizes are placed in their own vectors, similarly to the total concentration, mean diameter and median diameter. Each size distribution has a record of the instrument's name or distribution's name, the concentration's units and the distribution type (aerosol, cloud droplet, etc.).

For the aforementioned sections of the structure, it is strongly recommended to save the name and path of the original files from which the data originated in the cloud instance itself. Although it can be added in a section created by users, there are plans to design a section dedicated to this purpose in a later version of SAMAC.

The *Times of Manoeuvres* section keeps track of the times when the aircraft is executing certain manoeuvres in relation to the cloud. In this section, the beginning and end time of each manoeuvre is stored: the aircraft being in the area of the cloud (as opposed to being on its way to or from the airport), being above-, below-, horizontally in-cloud, and vertical profiles. Because only the start and end times are stored, two manoeuvre types can take place at the same time without having to keep two copies of the same data. For example, the last few seconds of a vertical manoeuvre can also belong to the beginning of an above cloud manoeuvre. Because of this section, it is easy to isolate a particular manoeuvre to make a calculation, such as calculating the average aerosol size distribution below-cloud only. This section can be populated manually or using a graphical interface and clicking at the beginning and end time of each manoeuvre.

The *Descriptive Data* section is where all the keywords relevant to data analysis are stored. Such descriptive keywords include the name of the measurement campaign or project, the type of cloud, the cloud's environment, the place where the cloud is situated, etc. This allows for automatically searching and selecting only the relevant cloud instances for a given analysis project.

Finally, the *Other Attributes* section contains attributes of the cloud instance that cannot be calculated without human intervention. Because human intervention requires time, these attributes are stored in this section to be retrieved whenever needed. This can, for example, include quality flags or classification results. The users should always review the *Other Attributes* section when they make a change to the original data and re-calculate them if needed.

After creating cloud instances by populating the structure, it is recommended to store the cloud instances in a list of instances and to save that list of cloud instances (we will henceforth call this list a *cloud database*). Users can then load the database and modify the data further or start the analysis of the data set. It is also possible to scan the cloud database for a selection of clouds (based on some criteria) and compare them easily.

4.2 Format and methods of a Cloud object

The cloud object methods will only work if the data is placed in the right sections of the structure and in the correct format. The exact format of the data in the structure is described in Table 1. The complete format description and an example of software guiding the creation of a cloud instance can be found with the supporting material in the software download and the wiki pages. Flexibility and the possibility to add new sections or new data types to the structure is accommodated: a user can create a new section at any point without disturbing the built-in methods, as long as the existing sections of the structure are not affected by the modifications.

To speed up the analysis, the cloud object comes with built-in methods and properties. These methods and properties are executed in a single-line command and return standard calculations or plots. By using these methods and properties, users can have a quick overview of their data set. A list of methods and properties included in this version of SAMAC along with a short description can be found in Table 2. More detailed description is available in the software documentation, online.

A method manipulates the cloud instance either to change its content or to give an output related to this instance. In the case of SAMAC, non-output methods are rather used to add new data or filter the data that were downloaded from the source files. We try to avoid deleting or modifying permanently any of the original data using methods to prevent accidental changes to take place. The users are, however, still capable of

modifying the original data by directly manipulating the cloud instance's data. There is also a method meant to filter erroneous or dummy data using NumPy's masked arrays. By using masked arrays, the erroneous data is masked (and thus not used in calculations or plots) yet the data is still stored and can be retrieved at any time if needed. The output methods in SAMAC are mostly designed to generate standard plots (e.g. profile plots) and return calculated quantities (e.g. average amount of drizzle drops). Some methods are also designed to add information to the cloud instance (e.g. quality flags for adiabaticity).

A property is a special attribute of a cloud instance that is computed only when accessed. Cloud properties are also given in Table 2. A property is very close to a method, but generally less elaborate and used for simpler calculations that do not require input. An example of a property in SAMAC is the liquid water path. The advantage of having such quantities computed only when they are accessed is that it gets updated when the original data involved is changed (for example, if the liquid water content was filtered or re-calibrated).

5 Software use

5.1 Basic use of Python commands with SAMAC

In this section, we give example lines of code to create a cloud instance and use it. Some samples of code we used to create our own cloud instances from a data file are available in the download package. SAMAC has so far only been used with IPython, an interactive shell for Python. We therefore recommend that users install IPython to run the software, so better support can be provided.

The first step for using SAMAC is to create a cloud instance. The instance is created using the __init__ method. For example:

```
import Cloud
```

```
MyFirstCloud=Cloud.Cloud(data, title, units, times, \
sizedists)
```
where data, title, units, times and sizedists are input variables containing the cloud's data in the specified cloud object format (see Table 1 for a list of formats). The first line imports the cloud class or cloud object, and the second line creates a cloud instance called MyFirstCloud. These cloud instances should be saved so they can be loaded in a later session. We have been saving our instances using the *pickle* module in a single list of cloud instances that we call a *cloud database*.

There are several ways to create a cloud instance. All involve the command lines above, but the way the data is put into it can vary. Along with SAMAC, we provide our own code that guides the user into finding the relevant data, shaping the data, and creating the cloud. This piece of code is tailored for our own data format and modifications will very likely be necessary. Another way would be to prepare the *data*, *title*, *units*, *times* and *sizedists* variables into the specified format and use the "Cloud.Cloud" command passing those variables as input. Yet another way to create an instance would be to create an empty cloud instance, and populate it manually afterwards. It is also possible to use a mix of passing the data directly and populating it afterwards. Post-populating the cloud can also be done using some of the built-in methods, especially the method *timechange* in the case of the variable *times,* the methods *addsizedist* and *addsizedistxl* for the variable *sizedists*, and finally the method add2basic for variables *data*, *title*, and *units.*

The data in the cloud object can be accessed using the name of the cloud and its address (replacing "c" in Table 1 by the name of the cloud instance). For example, the manoeuvre times are accessed like this:
```
MyFirstCloud.times
```
To calculate, for example, the average value of the total concentration of the first size distribution in the list of distributions, the data's address is used as if it was a variable:
```
Average_value=numpy.mean(MyFirstCloud.sd[0]["total"])
```
where numpy.mean is an averaging function available in the NumPy extension, and

MyFirstCloud.sd[0][″total″] is the address of the total concentration array for the first (indicated by 0, because python counts from zero) size distribution.

Properties are special built-in calculations done on data existing in the cloud instance's structure. They perform tasks very similar to the previous example, with the difference that the procedure is already coded and can be performed with a one-line command. For example, the liquid water path in a cloud can be calculated from a vertical profile, integrating the liquid water content over altitude. This could be performed by executing a few lines of code each time such quantity needs to be calculated. The calculation is complex enough and the quantity is required often enough to justify the coding and use of a property. One can obtain the liquid water path using the property *lwp* for each vertical profile in a cloud instance with this line:
```
LWP=MyFirstCloud.lwp
```
The liquid water path is stored in the variable *LWP*.

Methods are usually more complex and may include input variables or options. Methods are always called using brackets after the method's name. For example, to create or modify a cloud's description section, a user would type:
```
MyFirstCloud.describe()
```
and then create or modify the section through questions in the prompt. If input variables are required, these variables are defined inside the parentheses. For example, the method *plotavsd* plots the average size distribution for a given manoeuvre and instrument. The method that generates this plot must then be called like this:
```
MyFirstCloud.plotavsd(prof='verticloud',scan=2,inst='FSSP96')
```
to get the average size distribution of the 3rd (indicated with scan=2) vertical profile (verticloud), for the FSSP96 (Forward Scattering Spectrometer Probe).

In the case where input variables are required but default values are already defined in the method, the user can choose to define those variables (and change them) or not. For example, the default for *plotavsd* is already defined in the method and is prof=′belowcloud′, scan=0, inst=′PCASP′. If a user wants to see the average size distribution for the first below cloud manoeuvre from the PCASP (Passive Cavity Aerosol Spectrometer Probe), it would be enough to type this line:

```
  MyFirstCloud.plotavsd()
```

but this line would also work:

```
  MyFirstCloud.plotavsd(prof='belowcloud',scan=0,inst='PCASP')
```

Input variable definitions and default inputs are explained in the help section of each method. To read the description for method *plotavsd*, a user can request help using this line:

```
  help(MyFirstCloud.plotavsd)
```

5.2 Example use from the Canadian SOLAS 2003 campaign

The first measurement campaign on which the software was tested was the Canadian SOLAS 2003 campaign (Leaitch et al., 2010). In this section, we will use examples from that campaign to illustrate how SAMAC can speed up the analysis of cloud-related data. In this version of the software, methods and properties, as well as the structure were designed to study the aerosol-cloud-precipitation interaction, but we would like to emphasize that it would cost little effort to add sections to the structure or add methods and properties to fit other types of studies.

Time series and map plots are available from methods *overview* and *mapcloud* respectively (Fig. 4). The time series can be used once the cloud instance has been created to quickly review the measurements. The map shows the path of the measurements performed at any moment, superposed to the ocean and land map. A colour-coded time tracker is implemented in both methods for easy comparison.

For optimal use, some structure modifying methods should be used early after the creation of a cloud instance. For example, the methods *defheight and defBGheight* guide the user into defining the cloud base and cloud top by simply clicking on vertical profile plots. It is recommended to use methods *describe*, *defheight and defBGheight, lwcflag,* and *MaskData* to create a description section, define the cloud base and top, evaluate the adiabaticity of the vertical profiles and remove erroneous or unphysical

data, respectively. The method *timechange* can be used after the creation of the cloud instance to correct or add times of manoeuvres. See Table 2 for more methods that modify the cloud instance.

One of the advantages provided by SAMAC is that standard plots can be quickly generated using methods, leaving more time for users to work on science. For example, users may want to see the vertical profiles of cloud instances to guide their analysis. Such profiles can be generated with a single-line command using the method *vprof*, which plots each vertical profile in a separate figure, or *wholeprof*, which plots the data from the entire measurement period of the cloud instance, with altitude on the *y* axis (Fig. 5). Many other plot-making methods are provided (see Table 2 for a full list of the 9 methods with a figure output).

Standard calculations are also built in the analysis software. These are especially useful when a user wants to compare clouds to each other. Adding calculating methods is relatively easy. To make the most of community use of SAMAC, once new methods are coded by users, they should be committed and added to future versions of the software. There are already 18 methods that calculate new quantities from the cloud instance's available data. One useful calculating method is the property *lwp* which returns an array with the liquid water path for every vertical profile in a cloud. Examples of other useful calculating methods are *avsizedist,* which calculates the average size distribution (average concentration as a function of particle size) for a given manoeuvre, or *avCDNC,* which calculates the average cloud droplet number concentration weighted according to the liquid water content in a vertical profile. Other intermediate methods calculate values that are later used by other methods. Examples of such intermediate methods are *belowprof* (which finds measurements below cloud, in or under a vertical profile) and *adlwc* (which returns the adiabatic liquid water content).

Using these methods, a user could easily plot, for example, the liquid water path as a function of the in-cloud cloud droplet number concentration for all vertical profiles of stratiform, marine clouds using only a few lines of code. Assuming the user has a number of cloud instances stored in a cloud database called *CloudList*, these few lines would generate a figure like Fig. 6.

```
index=[i for i, c in enumerate(CloudList) if \
c.desc["cloudtype"]=="stratiform" and\
c.desc["environment"]=="marine"]
LWPs=list(); CDNCs=list();
  for i in index:
  for j in range(len(CloudList[i].times["verticloud"])):
    LWPs.append(CloudList[i].lwp[j][0])
    CDNCs.append(CloudList[i].avCDNC(scan=j))
plot(CDNCs,LWPs,'bo')
xlabel('Cloud Droplet Number Concentration (cm$^{-3}$)')
ylabel('Liquid Water Path (g m$^{-3}$)')
```

This code could be refined, for example by getting the units automatically from the cloud instances or by verifying if all cloud instances in the list have the same units. Plotting of complex quantities is simplified by the use of methods and properties, and the list of clouds can easily be scanned for cloud instances with selected characteristics.

6 Conclusions

We present a new software package, SAMAC, designed for the analysis of cloud measurement data using the programming language Python. The use of SAMAC by the aerosol-cloud-precipitation research community can speed-up the analysis of cloud data and facilitate intercomparisons among datasets and within and among research groups. Through the release of this software as an open-source project, the quality of the software may be improved from the evaluation by other researchers, and the existing methods and the diversity of methods may be improved by user coding according to their needs. Moreover, by sharing this software, all users save on coding time. In addition to describing the structure of the cloud object and the methods and

properties associated with it, a few examples of potential usage of this software shows how complex quantities can be accessed with only a few lines of code.

Although we recognize the need for further improvements, at its current stage, SAMAC is advanced enough for other groups to profit from its use. Reciprocally, SAMAC will benefit from a larger base of users with different needs capable of contributing to the software. Contribution to the software should cost little effort other than harmonizing the user's code to the existing code and will centralize aerosol-cloud-precipitation measurement handling, and make results more comparable to other users' results.

Detailed instructions on how to use and/or contribute to this software can be found on SAMAC's wiki pages at https://github.com/StephGagne/SAMAC/wiki. There is a discussion board (Issues) to discuss needed improvements and direction of future development, ask for help, or report bugs.

Acknowledgements. The authors would like to thank David Delene from University of North Dakota for his constructive comments on an early version of this software. NSERC's Visiting Fellowships in Canadian Government Laboratories Program is acknowledged for Stéphanie Gagné's funding. The NSERC USRA program is acknowledged for Landan MacDonald's funding.

References

Barnes, N.: Publish your computer code: it is good enough, Nature, 467, p. 753, 2010.
Beazley, D. M.: Python Essential Reference, 4th Edn., Developper's Library, 2009.
Boucher, O. and Lohmann, U.: The sulfate-CCN-cloud albedo effect: a sensitivity study with two general circulation models, Tellus B, 47, 281–300, 1995.
Delene, D. J.: Airborne data processing and analysis software package, Earth Science Informatics, 4, 29–44, 2011.
Gagné, S., MacDonald, L., Earle, M., Leaitch, W. R., and Pierce, J. R.: Effect of the aerosol size distribution on clouds and precipitation – results from the Canadian SOLAS 2003 and NARE 1993 aircraft campaigns, in preparation, 2014.

GitHub Help: available at: https://help.github.com/, last access: January 2014.
Hunter, J. D.: Matplotlib: a 2-D graphics environment, IEEE Computer Society, 9, 90–95, 2007.
Kleinman, L. I., Daum, P. H., Lee, Y.-N., Lewis, E. R., Sedlacek III, A. J., Senum, G. I., Springston, S. R., Wang, J., Hubbe, J., Jayne, J., Min, Q., Yum, S. S., and Allen, G.: Aerosol concentration and size distribution measured below, in, and above cloud from the DOE G-1 during VOCALS-REx, Atmos. Chem. Phys., 12, 207–223, doi:10.5194/acp-12-207-2012, 2012.
Leaitch, W. R., Banic, C. M., Isaac, G. A., Couture, M. D., Liu, P. S. K., Gultepe, I., Li, S.-M., Kleinman, L., Daum, P. H., and MacPherson, J. I.: Physical and chemical observations in marine stratus during the 1993 North Atlantic Regional Experiment: factors controlling cloud droplet number concentrations, J. Geophys. Res., 101, 29123–29135, 1996.
Leaitch, W. R., Lohmann, U., Russell, L. M., Garrett, T., Shantz, N. C., Toom-Sauntry, D., Strapp, J. W., Hayden, K. L., Marshall, J., Wolde, M., Worsnop, D. R., and Jayne, J. T.: Cloud albedo increase from carbonaceous aerosol, Atmos. Chem. Phys., 10, 7669–7684, doi:10.5194/acp-10-7669-2010, 2010.
Lin, J. W.-B.: Why Python is the next wave in Earth Sciences Computing, B. Am. Meteorol. Soc., 93, 1823–1824, doi:10.1175/BAMS-D-12-00148.1, 2012.
Lu, M.-L., Conant, W. C., Jonsson, H. H., Varutbangkul, V., Flagan, R. C., and Seinfeld, J. H.: The Marine Stratus/Stratocumulus Experiment (MASE): aerosol–cloud relationships in marine stratocumulus, J. Geophys. Res., 112, D10209, doi:10.1029/2006JD007985, 2007.
Masked Arrays – NumPy v1.8 Manual: available at: http://docs.scipy.org/doc/numpy/reference/maskedarray.html, last access: December 2013.
Matplotlib Python Plotting: Matplotlib 1.3.1 documentation, available at: http://matplotlib.org/, last access: December 2013.
Merali, Z.: Why scientific programming does not compute, Nature, 467, 775–777, 2010.
Morrison, H., Curry, J. A., and Khvorostyanov, V. I.: A new double-moment microphysics parameterization for application in cloud and climate models. Part I: Description, J. Atmos. Sci., 62, 1665–1677, 2005.
Nenes, A. and Seinfeld, J. H.: Parameterization of cloud droplet formation in global climate models, J. Geophys. Res., 108, 4415, doi:10.1029/2002JD002911, 2003.
NumPy: available at: http://www.numpy.org/, last access: December 2013.
Python Programming Language: Official Website, available at: http://www.python.org, last access: December 2013.

Rauber, R. M., Zhao, G., Di Girolamo, L., and Colón-Robles, M.: Aerosol size distribution, particle concentration, and optical property variability near Caribbean trade cumulus clouds: Isolating effects of vertical tranport and cloud processing from humidification using aircraft measurements, J. Atmos. Sci., 70, 3063–3083, doi:10.1175/JAS-D-12-0105.1, 2013.
RPy: A simple and efficient access to R from Python, available at: http://rpy.sourceforge.net/, last access December 2013.
SciPy.org: SciPy.org, available at: http://www.scipy.org/, last access: December 2013.
Solomon, S., Qin, D., Manning, M., Chen, Z., Marquis, M., Averyt, K. B., Tignor, M., and Miller, H. L.: IPCC, 2007: Climate Change 2007: the Physical Science Basis, Contribution of Working Group I to the Fourth Assessment Report of IPCC, Cambridge University Press, Cambridge, UK and New York, NY, USA, 996 pp., 2007.
Wood, R., Mechoso, C. R., Bretherton, C. S., Weller, R. A., Huebert, B., Straneo, F., Albrecht, B. A., Coe, H., Allen, G., Vaughan, G., Daum, P., Fairall, C., Chand, D., Gallardo Klenner, L., Garreaud, R., Grados, C., Covert, D. S., Bates, T. S., Krejci, R., Russell, L. M., de Szoeke, S., Brewer, A., Yuter, S. E., Springston, S. R., Chaigneau, A., Toniazzo, T., Minnis, P., Palikonda, R., Abel, S. J., Brown, W. O. J., Williams, S., Fochesatto, J., Brioude, J., and Bower, K. N.: The VAMOS Ocean-Cloud-Atmosphere-Land Study Regional Experiment (VOCALS-REx): goals, platforms, and field operations, Atmos. Chem. Phys., 11, 627–654, doi:10.5194/acp-11-627-2011, 2011.
Yarkoni, T.: The homogenization of scientific computing, or why Python is steadily eating other languages' lunch, available at: http://www.r-bloggers.com/the-homogenization-of-scientific-computing-or-why-python-is-steadily-eating-other-languages-lunch/, last access: 2 December 2013.
Zhang, K., Liu, X., Wang, M., Comstock, J. M., Mitchell, D. L., Mishra, S., and Mace, G. G.: Evaluating and constraining ice cloud parameterizations in CAM5 using aircraft measurements from the SPARTICUS campaign, Atmos. Chem. Phys., 13, 4963–4982, doi:10.5194/acp-13-4963-2013, 2013.

Table 1. Cloud object structure description. The first column shows the address of a certain data, the second its name, the third a more detailed description and the fourth the format.

Address	Name	Description	format
c.data	Main (basic) data	Table including time series of a selection of quantities. All the data should share the same time column. These data would usually include time, aircraft positioning and cloud-related data.	2-D numpy array or masked array [numpy.ma.core.MaskedArray or numpy.array]. Each quantity is in a row vector.
c.dttl	Titles to the main data	Titles or quantity description associated with the main data	1-D list. The number of items should correspond to the number of rows in c.data.
c.dunit	Units to the main data	Units associated with the main data	1-D list. The number of items should correspond to the number of rows in c.data.
c.orient	Orientation of the main data	Tells whether the main data is organized in rows or columns.	string
c.extradatanum	Number of extra data	Number of extra data added after initialization in the extradata section of the cloud structure (i.e. data that was not the same length as the main data or added later).	integer
c.extradata	Extra data	Extra time series data that has a different time stamp from the main data.	list of items containing extra data in the same format as the main data
c.extrattl	Titles to the extra data	Titles associated with the extra data	list of lists of items containing titles for extra data in the same format as the titles to the main data
c.extraunit	Units to the extra data	Units associated with the extra data	list of lists of items containing units for extra data in the same format as the units to the main data
c.sd	Size distribution section	Section in which all the size distributions are stored	list of size distributions. Each size distribution unit can be accessed using c.sd[number] where number goes from 0 to the number of size distributions in the cloud instance.
c.sd[n]	Size distribution unit	Size distribution structure for one instrument	dictionary. *n* is the position of the size distribution in the list of size distributions sd.
c.sd[n][″data″]	Size distribution concentration	Size distribution concentration as a function of time and size. The concentration must be in dN/dlogDp format	2-D numpy array. The shape is number of size bins (as rows) per number of time entries (as columns)
c.sd[n][″bins″]	Size distribution sizes	Mean particle diameter sizes for the distribution	1-D numpy array. The array should be as long as the number of rows in c.sd[n][″data″]
c.sd[n][″time″]	Size distribution times	Times at which the size distribution was recorded	1-D numpy array. The array should be as long as the number of columns in c.sd[n][″data″]

Table 1. Continued.

Address	Name	Description	format
c.sd[n][″Distname″]	Size distribution name	Name of the size distribution, either an instrument name or a more descriptive name such as ″aerosol″ of ″cloud droplets″	string
c.sd[n][″units″]	Size distribution units	units of the size distribution's concentration	string
c.sd[n][″medp″]	Median diameter	Median particle diameter for this size distribution	1-D numpy array. The array should be as long as the number of columns in c.sd[n][″data″]
c.sd[n][″mdp″]	Mean diameter	Mean particle diameter for this size distribution	1-D numpy array. The array should be as long as the number of columns in c.sd[n][″data″]
c.sd[n][″total″]	Total particle number concentration	Total particle number concentration for this size distribution	1-D numpy array. The array should be as long as the number of columns in c.sd[n][″data″]
c.sd[n][″sourcefile″]	Size distribution source file	Source file for the size distribution's data	string
c.sd[n][″sdtype″]	Size distribution type	Type of particles measured in this size distribution (Aerosol, Cloud droplet, or Precipitation)	string
c.times	Manoeuvre times	Times at which the cloud measurement and manoeuvres start and end	dictionary. All times in the entire structure should be in days since 1 Jan of year 0 (1 Jan of year 0 at noon would be written 1.5)
c.times[″cloud″]	Period measuring a cloud	Times at which the measurements of the cloud started and ended	numpy array. Shape is 1 per 2. Start time at [0,0] and end time at [0,1].
c.times[″abovecloud″]	Periods above cloud	Times at which the measurements above the cloud started and ended	numpy array. Shape is *N* per 2. Start time at [*N*, 0] and end time at [*N*, 1] where *N* is the (*N* − 1)th measurement leg above cloud.
c.times[″belowcloud″]	Periods below cloud	Times at which the measurements below the cloud started and ended	numpy array. Shape is *N* per 2. Start time at [*N*, 0] and end time at [*N*, 1] where *N* is the (*N* − 1)th measurement leg below cloud.
c.times[″horicloud″]	Periods horizontally in cloud	Times at which the measurements horizontally in the cloud started and ended	numpy array. Shape is *N* per 2. Start time at [*N*, 0] and end time at [*N*, 1] where *N* is the (*N* − 1)th measurement leg horizontally in cloud.
c.times[″verticloud″]	Periods vertically in cloud	Times at which the measurements vertically in the cloud started and ended	numpy array. Shape is *N* per 2. Start time at [*N*, 0] and end time at [*N*, 1] where *N* is the (*N* − 1)th measurement leg vertically in cloud.
c.descedit	Number of descriptions	Number of times the method describe has been used to either create or edit the Descriptive Data section.	integer

Table 1. Continued.

Address	Name	Description	format
c.desc	Descriptive Data	Section where qualitative or descriptive data is stored	dictionary
c.desc[″campaign″]	Campaign	Name of the campaign or project where the data was measured or created	string
c.desc[″date″]	Date	Date on which the cloud was measured	string, YYYYMMDD
c.desc[″time″]	Time of cloud	The time at which the cloud measurement started.	string, HH:MM
c.desc[″cloudtype″]	Type of Cloud	Type of cloud observed (e.g. Stratiform, convective, etc.)	string
c.desc[″environment″]	Environment	Environment in which the cloud was situated (e.g. Maritime, desert, tropical forest, etc.)	string
c.desc[″flight#″]	Flight number	Flight or simulation number or name	string
c.desc[″humanplace″]	Place	Place above which the cloud was measured. The place should be recognizable by humans (e.g., a city name)	string
c.desc[″warmcold″]	Cloud temperature	Warm, cold (ice), or mixed clouds	string
c.desc[″landwater″]	Surface	Surface type below the cloud: land, water or both.	string
c.desc[″layered″]	Layered	Describes if there is a knowledge of other layers of cloud above or below the measured cloud	string
c.props	Other Attributes	Section in which properties needing human input are stored after initialization	dictionary
c.props[″height″]	Height	Cloud base and top height for all vertical profiles (lower and upper estimates)	list of numpy arrays. For each vertical profile there is a height entry (in the same order). Each entry is an array of shape 4 by 1 that includes the lowest and highest estimate for cloud base and lowest and highest estimate for cloud top.
c.props[″BGheight″]	Best guess height	Cloud base and top height for all vertical profiles (best guess)	list of numpy arrays. For each vertical profile there is a best guess height entry (in the same order). Each entry is an array of shape 2 by 1 with the best guess for cloud base and cloud top.
c.props[″lwcflags″]	LWC flags	Flags reflecting the quality of the LWC profile compared to the adiabatic LWC	list of numpy arrays. For each vertical profile there is a LWC profile quality entry (in the same order). Each entry is an array containing all the flags pertinent to the profile.

Table 2. List of methods and properties available in SAMAC 0.9.2. The name of the method or property is found in the first column, followed by a short description, a list of inputs and, in the last column, a list of outputs.

Name	Description	Inputs	Outputs
Methods	Methods are called using parentheses at the end of the method's name. E.g. c.cloudplot() or c.cloudplot(Rtime=1)		
overview	Plots the time series of altitude, particle concentration, droplet concentration, liquid water content and drizzle concentration on one subplot with possibility of offsetting and magnifying each data. Plots the latitude and longitude on a second subplot. The plot includes a manoeuvre tracker and a time tracker.	interact: interactive mode, Rtime: time format option	Figure
mapcloud	Plots the trajectory of the probe over a map, one subplot zoomed out and one zoomed in includes a time tracker. The size of the dots indicates the liquid water content.	interact: interactive mode	Figure
add2basic	Adds data to the basic aircraft data, if the new data shares the same time vector, or adds it to c.extradata if they have a different time vector.	newdata: the new data to be added, newttl: new titles, newunit: new units	None. The method modifies the cloud instance.
describe	Prompts the user to create a description section or modify the current description.	None	None. The method modifies the cloud instance.
plotsd	Plots the size distribution (colour plot) for a selection of instruments, and scans, as a function of time.	Rtime: time format option	Figure(s)
addsizedistxl	Adds a size distribution from an excel file to the size distribution section.	None	None. The method modifies the cloud instance.
addsizedist	Adds a size distribution from a text file to the size distribution section.	None	None. The method modifies the cloud instance.
vprof	Plots the vertical profile(s) of a cloud instance: altitude on the *y* axis and liquid water content, total concentration from various size distribution instruments, relative humidity, drizzle concentration, temperature, theta *Q*, effective radii from various instruments.	interact: interactive mode	Figure(s)
wholeprof	Plots the liquid water content, total concentration from various size distribution instruments, relative humidity, drizzle concentration, temperature, theta *Q*, effective radii from various instruments of a cloud instance on the *x* axis and altitude on the *y* axis.	interact: interactive mode	Figure

Table 2. Continued.

Name	Description	Inputs	Outputs
defheight	Prompts the user to define a range for the cloud base and top for every vertical profile by clicking on a figure.	None	None. The method modifies the cloud instance.
defBGheight	Prompts the user to enter a best guess cloud base and top for every vertical profile by clicking on a figure.	None	None. The method modifies the cloud instance.
lwcflag	Prompts the user to assign quality flags for liquid water content profiles in each vertical profiles by comparing to the adiabatic liquid water content.	None	None. The method modifies the cloud instance.
timechange	Prompts and guides the user to redefine or add manoeuvre times.	Rtime: time format option	None. The method modifies the cloud instance.
MaskData	Masks data that the user selects by drawing a square on a plot or filtering for outliers.	None	None. The method modifies the cloud instance.
avsizedist	Returns the average size distribution concentration and sizes.	prof: chosen manoeuvre, num=(n-1)th manoeuvre of the chosen type, Inst: instrument	2-D NumPy Array
plotavsd	Plots an average size distribution.	prof: chosen manoeuvre, num=(n-1)th manoeuvre of the chosen type, Inst: instrument	Figure
plotallavsd	Plots the average size distribution for all instruments, with possible exceptions.	prof: chosen manoeuvre, num: (n-1)th manoeuvre of the chosen type, Interact: interactive mode, Exc: excepted instruments	Figure
belowprof	Finds the probe passages below cloud, below vertical profiles. Also find points where the plane was in the same area as the vertical profile (below-, above-, and in-cloud).	None	List of dictionaries

Table 2. Continued.

Name	Description	Inputs	Outputs
hrzplot	Plots the altitude, liquid water content, droplet and aerosol total concentration, theta Q, temperature, updraft velocity and its running standard deviation as a function of time during a horizontal manoeuvre. One plot is made for each horizontal manoeuvre.	interact: interactive mode, Rtime: time format option	Figure(s)
precipincloudTF	Tells whether there is precipitation in-cloud (horizontal or vertical manoeuvres).	abovesize: drizzle bigger than this size	Boolean
totalprecip	Returns the integrated precipitation number concentration for a vertical profile.	abovesize: drizzle bigger than this size, scan: (n-1)th vertical profile	Float
totalvolprecip	Returns the integrated precipitation volume concentration for a vertical profile.	abovesize: drizzle bigger than this size, scan: (n-1)th vertical profile	Float
precipconc	Returns the integrated precipitation number concentration in-cloud for a vertical profile.	abovesize: drizzle bigger than this size, Scan: (n-1)th vertical profile, minlwc: minimum liquid water content to be considered in-cloud	Float
avCDNC	Returns the weighted average cloud droplet number concentration (weighted according to the liquid water content) in a vertical profile.	abovesize: droplets above this size, uppersize: droplets below this size, Inst: instrument, Scan: (n-1)th vertical profile	Float
vpinfo	Computes statistical information about a quantity from the main data for vertical profiles.	param: name of the quantity in the main data	Dictionary

Table 2. Continued.

Name	Description	Inputs	Outputs
effrad	Computes the effective radius for a given instrument during the entire cloud period.	inst: instrument, bindist: are the size bins distributed linearly or logarythmically	1-D NumPy Array
path3d	Plots a 3-D view of the probe's path	None	Figure
writeouthdf5	Creates an hdf5 programming language-independent backup file for the cloud where all the data is saved.	fName: name of the file to be saved fdir: path of the directory in which the file should be saved	Hdf5 file
Properties	Properties are called directly. E.g. c.lwp		
lwp	Calculates the liquid water path for each vertical profile.	x	Numpy Array
adlwc	Returns the altitude and corresponding adiabatic liquid water content.	x	List of 2-D NumPy Arrays
thickness	Computes the maximum and minimum estimated cloud depth for each vertical profile.	x	Dictionary of lists of NumPy Arrays
tempinfo	Special case of vpinfo over the whole cloud. (Does not handle the extradata module.)	x	Dictionary
presinfo	Special case of vpinfo use over the whole cloud. (Does not handle the extradata module.)	x	Dictionary
precipblwTF	Tells whether there is precipitation below cloud.	x	Dictionary of booleans
precipblwvpTF	Tells whether there is precipitation below cloud, below a vertical profile.	x	Dictionary of lists of booleans
turbhrz	Returns the average updraft velocity standard deviation and the distance over which the average was calculated. Also returns the corresponding indexes for the used stretches of measurements.	x	List of lists of lists
angles	Returns the time, vertical angle and horizontal angle of the aircraft or probe.	x	List of 1-D NumPy Arrays

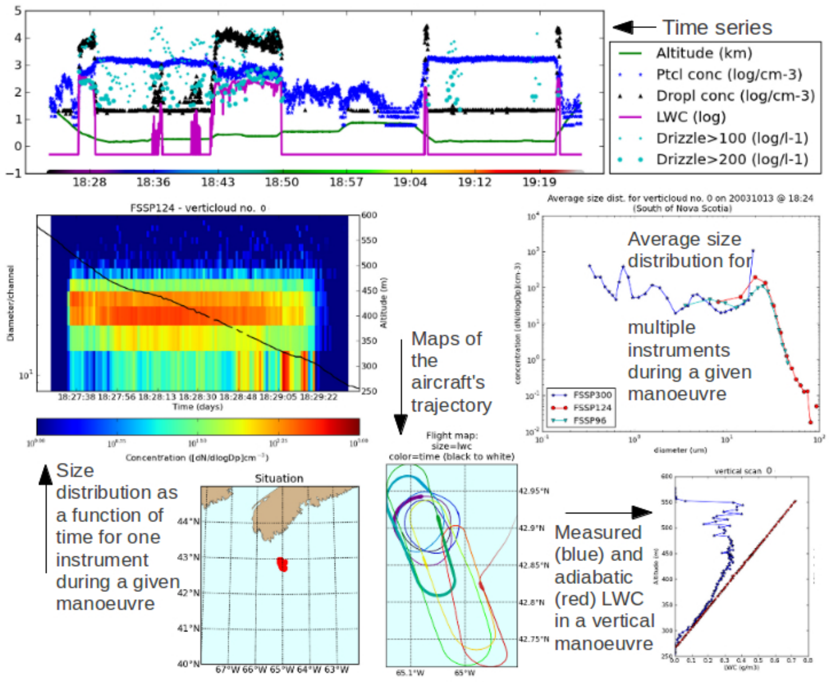

Fig. 1. Figures generated by a selection of methods available in SAMAC. On the first row, the time series shows a selection of data as a function of time (more detail on the time series in Fig. 4a). On the left-hand side figure of the middle row is a size distribution as a function of time during a vertical profile manoeuvre. The colour represents the concentrationand the black line the altitude. On the righthand side, the average size distribution is plotted for a selection of instruments for one manoeuvre. The maps on the bottom row show a general situation map and a zoomed-in trajectory of the aircraft in the cloud (more detail on the maps in Fig. 4b). Finally, the figure in the lower right corner displays a profile of the measured and adiabatic liquid water contents.

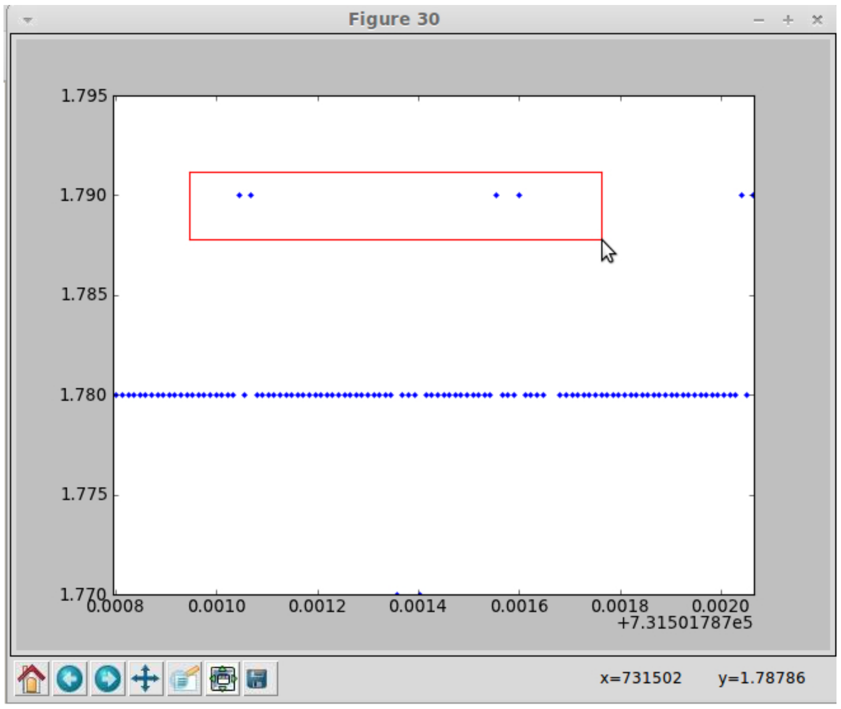

Fig. 2. This is an example of an interactive method. In this case, the method asks the user to choose a data type to display. The user can then choose to use an automatic outliers removal tool or to zoom in and select the points that need to be masked with a few clicks of the mouse. Here, the points in the red rectangle will be masked when the user confirms it.

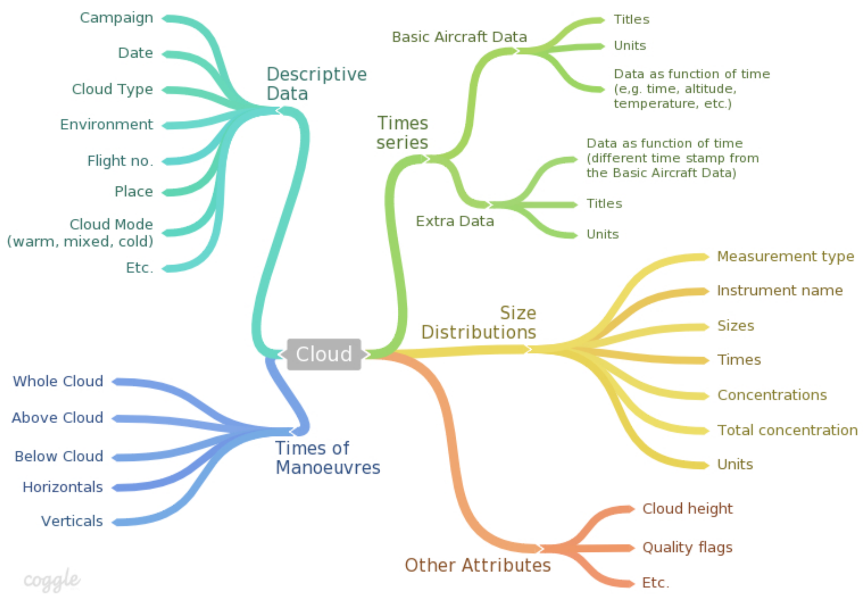

Fig. 3. Basic structure of the cloud object. All main sections are created upon initialization of a cloud instance (except for the Descriptive Data section, which is created using the method *describe*).

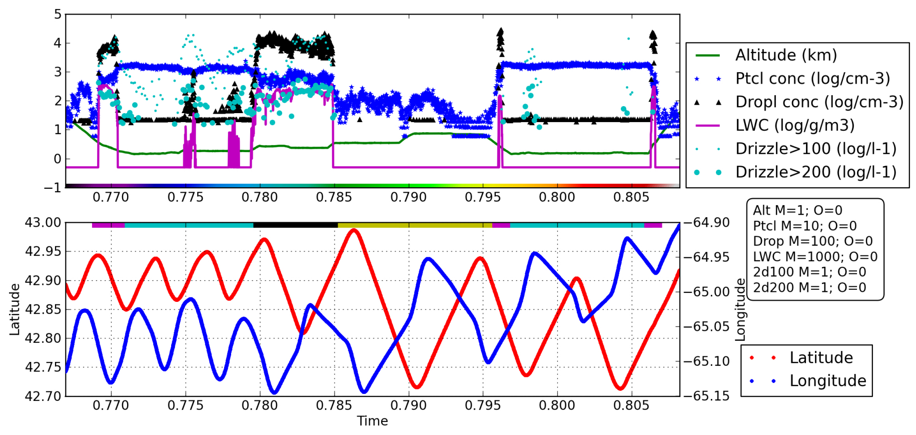

Fig. 4a. Qualitative view of the aircraft's altitude, the measured particle and cloud droplet total concentrations, the liquid water content and drizzle drop concentration in the upper subplot, and the aircraft's latitude and longitude on the lower subplot. All the data in the upper subplot can be magnified and/or offset (values of which are indicated in the rounded box on the right hand side of the plot). At its bottom, a colour-coded time tracker is inserted. The lower subplot shows the latitude (left axis) and longitude (right axis) of the aircraft or probe. At its top, the colours indicate which manoeuvre the aircraft is making (magenta= vertical, cyan = below cloud, black= horizontal in cloud and yellow= above cloud).

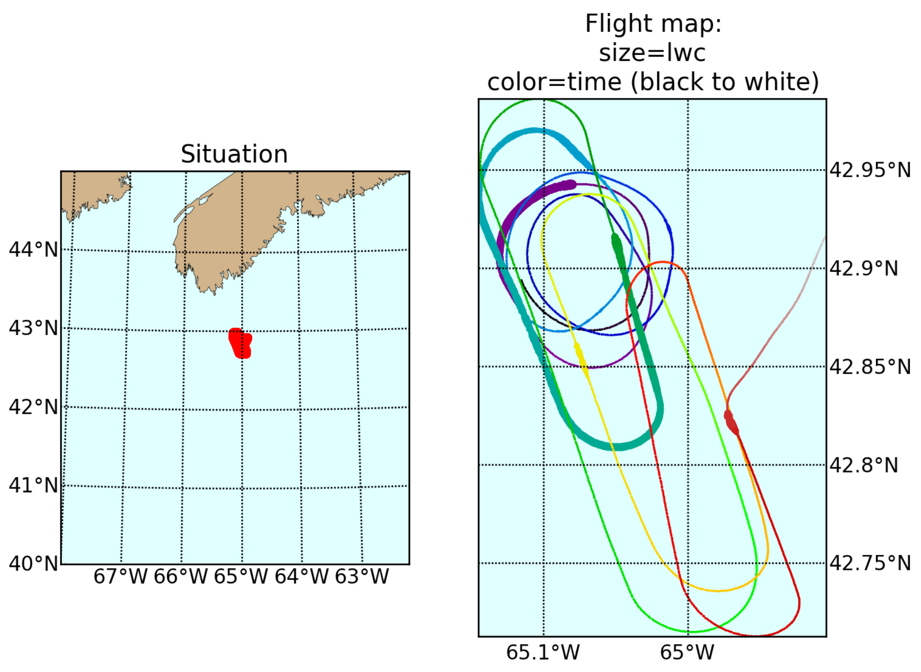

Fig. 4b. Path of the aircraft or probe while the cloud instance was being measured. The subplot on the left shows a zoomed-out map and the subplot on the right shows a zoomed-in situation, with the size of the points corresponding to the liquid water content and the colour corresponding to the time tracker in the time series **(a)**.

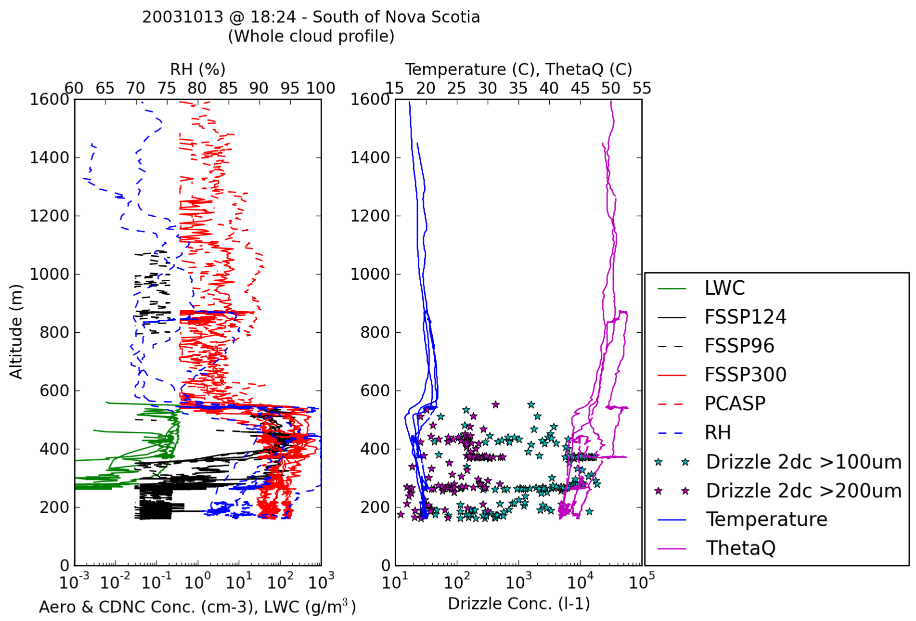

Fig. 5a. Two methods that display the vertical profile of a cloud. In this cloud instance, the aircraft made three vertical profiles (in addition to two below-cloud manoeuvres, two above-cloud manoeuvres and one in-cloud horizontal manoeuvre). **(a)** was created using the method *wholeprof* and displays the liquid water content, FSSP total concentrations, PCASP total concentration, relative humidity, drizzle concentrations above 100 and 200 microns, as well as temperature and theta *Q* for all the manoeuvres as a function of altitude on the *y* axis.

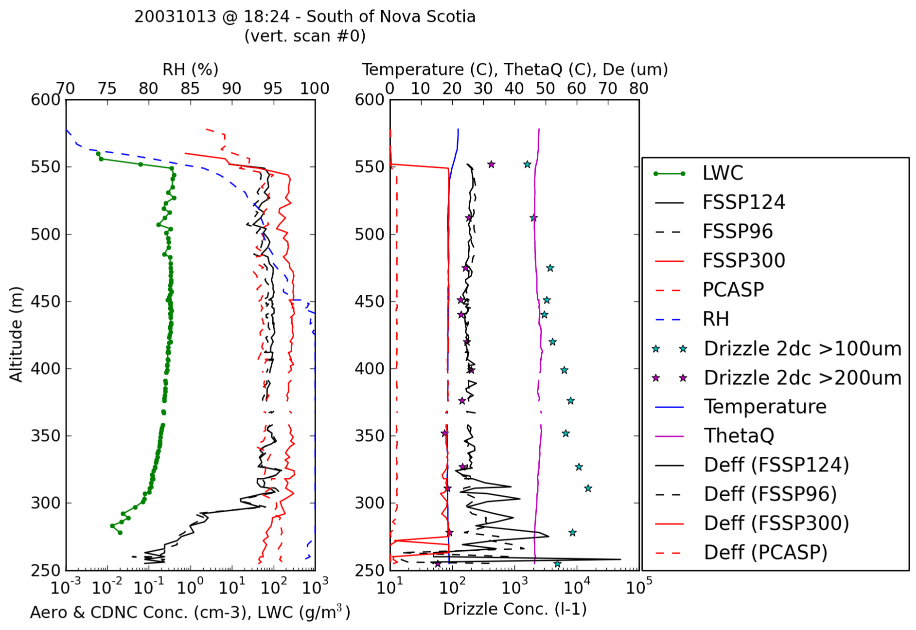

Fig. 5b. was created using the method *vprof* and displays the same quantities as **(a)** as well as the eff ective radii of the FSSPs and PCASP for the first vertical profile only, as a function of altitude on the *y* axis.

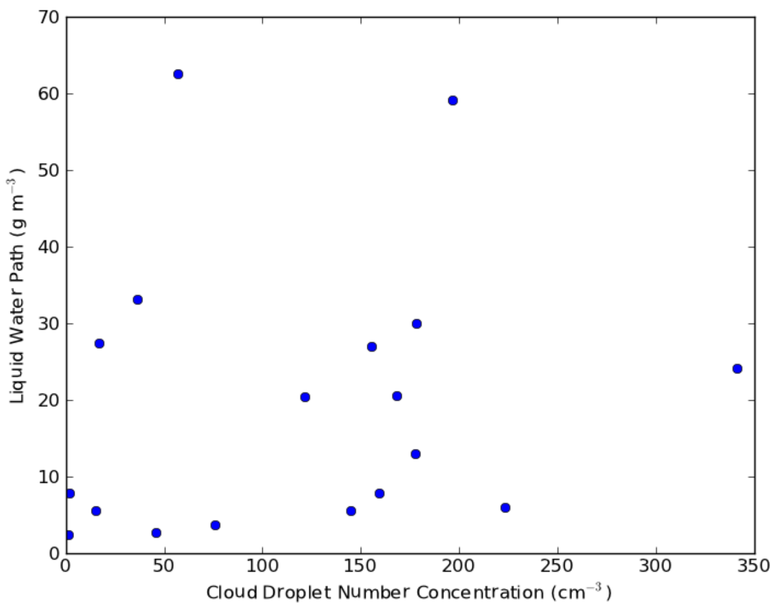

Fig. 6. Figure resulting from the example code at the end of Sect. 5.2. Here, we plotted the liquid water path as a function of the cloud droplet number concentration. Many other useful figures can be plotted using all the available methods and properties.