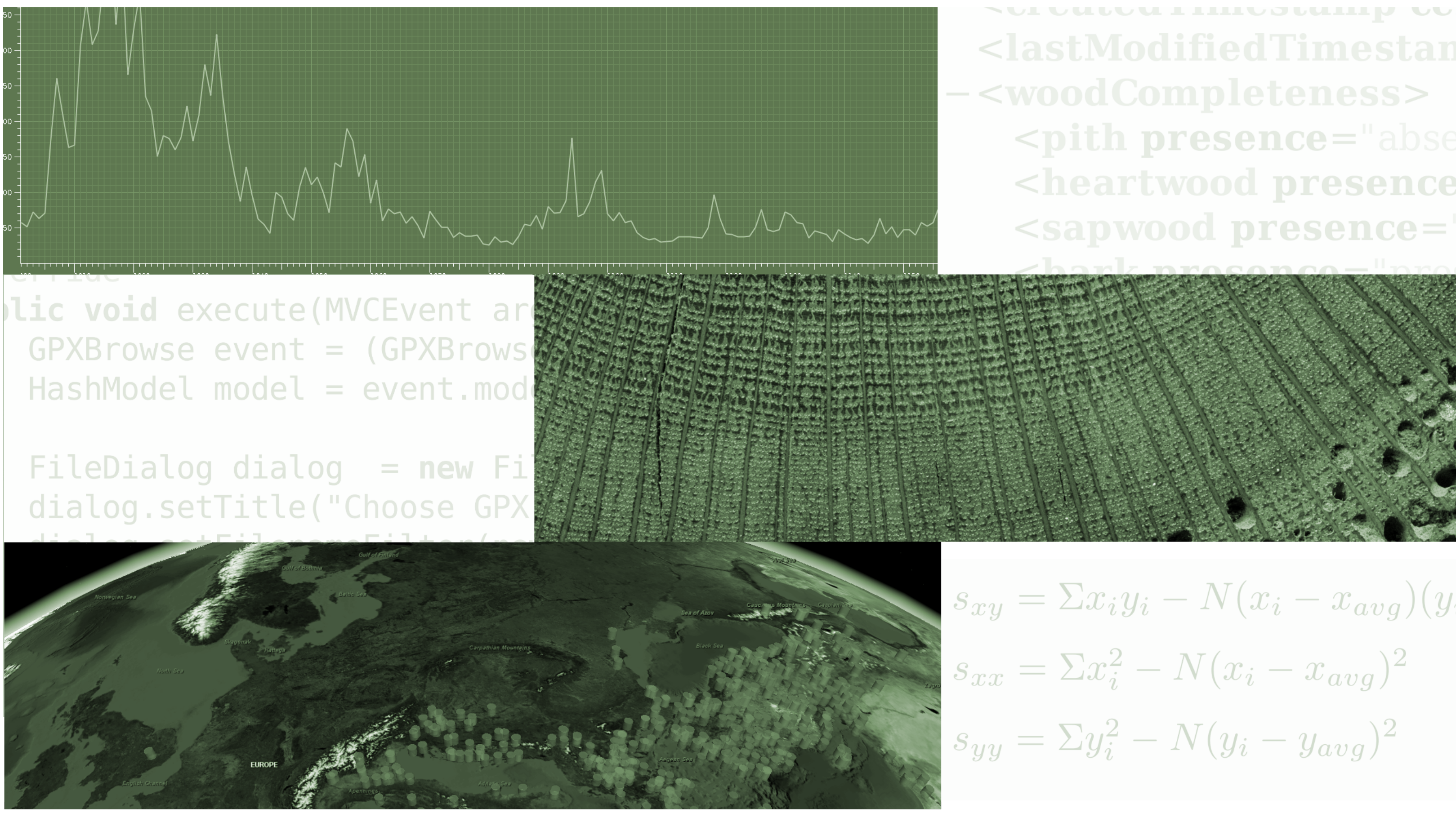
Corina

A guide for users and developers



By Peter W. Brewer and Ken Harris

For Corina server and desktop
version 2.12

Malcolm and Carolyn Wiener Laboratory for Aegean
and Near Eastern Dendrochronology
B48 Goldwin Smith Hall
Cornell University,
Ithaca, New York 14853. USA.

☎ +1 607 255 8650
✉ p.brewer@cornell.edu

CORINA

A guide for users and developers

By Peter W. Brewer and Ken Harris

Contents

Preface

Corina is the tree ring measuring and analysis program developed at the Cornell Tree-Ring Laboratory. It is focused primarily on the measurement of tree ring widths and the organization and curation of the data, metadata and physical samples. It is cross-platform (running on all Java 6 enabled operating systems including Windows, MacOSX and Linux) and open-source. It includes support for Velmex, Lintab and Hensen measuring platforms.

Corina has been developed since 2000 as a desktop Java application, following an earlier DOS-based version which itself was derived from a collection of FORTRAN and C utilities. Earlier iterations of Corina (version 0.x and 1.x) were built around a standard file-based data management system. In 2007, work began on a major rewrite of the software whereby this file-based data management was replaced with an object-relational database management system (ORDBMS) and server/client webservice infrastructure. This series of releases (versions 2.x) are what are described in this manual.

This manual is divided into two main sections, the first for users, the second for developers. Corina is open source software (see the details of the license on pages 119–123) so you are welcome to inspect and edit the code. The second part of this manual will help you do that.

Over the years Corina has been developed by a number of people: Peter Brewer, Chris Dunham, Aaron Hamid, Ken Harris, Drew Kalina, Lucas Madar, Daniel Murphy, Robert 'Mecki' Pohl and Kit Sturgeon. We hope that you find Corina useful and look forward to hearing your feedback.

Part I

User Guide

Chapter 1

Installation

Corina is made up of two packages; the Corina desktop application and the Corina database server. Corina was designed primarily for laboratories with multiple users, each running the Corina desktop application on their own computer connecting to a single central server containing the lab's data. In this situation the Corina server would be run on a separate computer to those running the desktop client, but this need not necessarily be the case. It is perfectly possible to run both the server and the client on the same computer. This is likely to be the situation if you simply want to try out Corina, if you don't have a separate server, or if you do not work in a multi-user laboratory.

1.1 Desktop application

Installation packages for the Corina desktop application are available for Windows, MacOSX and Ubuntu Linux. Corina can also be run on other operating systems as long as they support Java 6.

To install Corina, download the installation file for your operating system from *http://dendro.cornell.edu/corina/download.php*. The website should provide you with a link to the installer for your current operating system:

- **Windows** – Run the setup.exe and follow the instructions. If you do not have Java installed the installer will direct you to the Java website where you can get the latest version. Once installed, Corina can be launched via the Start menu.
- **Mac OS X** – As mentioned above, Corina requires Java 6. Although MacOSX ships with Java installed, unfortunately Apple have been very slow to provide Java 6. Although it was released in 2006, it was not until August 2009 that Apple made Java 6 available as part of v10.6 (Snow Leopard). Corina can therefore only be run on Snow Leopard or later. To do so, download the dmg disk image file and mount it by double clicking on it. Drag the Corina.app into your applications folder and copy the manual and license files to somewhere convenient in your documents folder. For the more adventurous there is the possibility that you could run Corina using SoyLatte instead of the standard Java installation that comes with the operating system. This could be a possible method for running Corina even on earlier versions of MacOSX but is unsupported and largely untested.
- **Ubuntu Linux** – A deb file is available which was designed for use on Ubuntu distributions but should work on any Debian based system. Install using your favorite package management system e.g.

  ```
  sudo dpkg --install corina_2.xx-1_all.deb
  ```

  On Ubuntu and similar distributions, the package should add a Corina shortcut to your applications menu. Alternatively you can start Corina from the command line by typing corina.
- **Other operating systems** – Make sure you have Java 6 installed, then download the Corina jar file to your hard disk. You can run Corina from the command line by typing:

```
java -jar corina.jar
```

1.1.1 Mapping support

Corina includes 3D mapping for visualization of sampling locations. Although this is not necessary for most tasks, to make use of the mapping functions you will require a OpenGL 3D capable graphics card. To check whether your computer already supports 3D mapping, open Corina, go to Admin, then Site map. Corina will warn you if your graphics card is not supported.

All MacOSX computers should automatically support OpenGL. Most Windows and Linux computers made since 2006 should also support OpenGL, however, this does require proper drivers to be installed. In some cases Windows computers may include a compatible graphics card, but may only have the default Windows video drivers installed. If you are having trouble with the mapping in Corina make sure you have installed the most recent drivers for your graphics card. Linux users may be required to install proprietary graphics drivers.

The mapping component of Corina makes use of NASA's open source World Wind Java. NASA's website *http://worldwind.arc.nasa.gov/* contains further information and instructions that you may find helpful if you are having problems getting the mapping to work.

1.2 Server installation

For the Corina desktop application to be useful you will also require access to a Corina server. If you are running Corina in a lab where the Corina server has already been set up by your systems administrator, you can skip this section.

The Corina server is made up of a number of components, which unlike the desktop client, can not be easily combined together into cross-platform packages. Although all the constituent components are open-source and available for all major platforms, building and maintaining separate packages for each platform is too large a task for a small development team. To conserve resources, we therefore made the decision to utilize Virtual Machine technology to ensure that the Corina server could still be run on all major operating systems. This means that we can package the Corina server for a single operating system (Ubuntu Linux) and then distribute it as a Virtual Appliance that can be run as a program on your normal operating system.

The Corina server is therefore available via two main methods. The first is as a VirtualBox Virtual Appliance which can be run on any major operating system, the second is as an Ubuntu package for running natively on a Linux server. The source code for the server is also available so it is perfectly possible for more experienced users to set up the Corina server to run natively on other platforms. But to do this you will require a good knowledge of Apache 2, PHP and PostgreSQL.

1.2.1 Virtual Appliance - all operating systems

To run the Corina server Virtual Appliance, you will first need to download and install VirtualBox from *http://www.virtualbox.org*. Installation packages are available for Windows, MacOSX, OpenSolaris and many Linux distributions.

Once you have VirtualBox installed, you will then need to download the Corina server from the Cornell website *http://dendro.cornell.edu/corina*. This package contains a bare-bones Ubuntu Linux server with everything required to run the Corina server installed and ready to use. As VirtualBox, the entire Ubuntu operating system and Corina server components are all open source there are no license fees to pay.

Open VirtualBox and go to File, Import Appliance, then follow the wizard selecting the Corina server appliance file when prompted. Once installed you can run your server by highlighting it in the list and pressing start. The server will boot up in a window alongside your normal operating system and eventually reach a login prompt. To save on download size and disk space only the essential packages to make the server run have been installed. This means there is no graphical interface just a command line. Hopefully this should not be a problem as once set up, the only interaction needed with the Virtual Appliance will be through the normal

Corina desktop application. If you would prefer to use a graphical interface to the server this can be easily installed. See chapter 10 for further details.latex define environment

Before you can use your server you will need to know the IP address that the server has been assigned by your network. To do this login at the prompt with the default admin credentials: user – corina; password – w3l0v3tr33s. Once logged in, type `corina --test` and a basic configuration test will be performed. If all is well, all tests will be passed and it will tell you the URL of your new server. You will need to set your Corina client to point at this webservice to use your server.

1.2.2 Ubuntu native installation

If you are fortunate enough to be running Ubuntu then the native Ubuntu deb package is the best and easiest method for installing the Corina server, otherwise see section 1.2.1 to install the server as a Virtual Appliance.

To install the Corina server in Ubuntu simply download the deb package from the Cornell server *http://dendro.cornell.edu/corina* and install with your favourite package manager. For instance, to install from the command line simply type:

```
sudo dpkg --install corina-server.deb
```

The package will automatically run a configuration script to assist with creating a database user, building the Corina PostgreSQL database, setting database permissions and setting up the Apache webservice. The configuration ends with a test routine to check all services are set up correctly and if so, will provide you with the URL of the newly configured Corina webservice.

1.2.3 Advanced install on other operating systems

As mentioned previously, the limited resources available for Corina development means that we have been unable to produce native installers for platforms other that Ubuntu. If you are an experience systems administrator though, it should not be too difficult to set up the Corina server manually.

The Corina server is essentially a PostgreSQL database accessed via a PHP webservice running on Apache 2. The following dependencies are therefore required: postgresql-8.4; postgis; postgresql-contrib-8.4; postgresql-8.4-pljava; sun-java6-jre; apache2; php5; php5-pgsql; php5-curl; php5-mhash.

The basic procedure for installation is as follows:

- Install all dependencies
- Create PostgreSQL database from Corina template SQL file
- Set up a database user and provide access to the server in the pg_hba.conf file
- Give this user read and write permissions to the database
- Copy the webservice code into a web accessible folder
- Set up Apache to see this folder by creating an entry in the sites-enabled folder
- Restart PostgreSQL and Apache and check you can access the webservice from a web browser

Chapter 2

Measuring samples

Although it is possible to manually enter the ring widths of your samples into Corina, it is normal to automate this process using a measuring platform. Corina supports the most common measuring platforms including Velmex and Lintab.

2.1 Configuring measuring platforms

Measuring platforms typically use serial ports to communicate to computers. In recent years computer manufacturers have been phasing out serial ports so you may need to purchase a serial-USB converter. Modern MacOSX, Linux as well as Windows 7 should support most serial-USB adapters out of the box, otherwise you must install the relevant drivers before continuing. Recent Lintab USB platforms use internal serial-USB converters so are treated in exactly the same way by Corina.

To begin, shut down your computer, attach your platform, then reboot and launch Corina. Next, go to the preferences window and open the hardware tab and you should see an interface that looks like figure 2.1.

In the type pull down menu, select the type of measuring equipment you are using. Note that this refers to the equipment that the computer is attached to, and not necessarily the measuring platform itself. For instance, Velmex platforms are typically connected through a Metronics digital readout device. Included in this list is the EveIO device which is an open-source device designed for the Cornell Tree-Ring Laboratory. Circuit drawings for this device can be obtained from the Cornell lab to enable Hensen measuring platforms to be used with Corina (and other software).

> Standard Lintab platforms use a proprietary communications protocol. Rinntech–the manufacturers of Lintab platforms–claim intellectual property rights over this protocol. During discussions between the Corina development team and Rinntech an agreement was reached whereby the Corina developers agreed not to release details of the protocol. In turn Rinntech has agreed to produce an adapter that can be attached to Lintab platforms so that they communicate with an open ASCII protocol. Users wishing to use Lintab platforms with Corina (or any software not developed by Rinntech) must therefore contact Rinntech and purchase an adapter.

Next you must choose the port that your platform is connected to from the pull down menu. In Windows this will be a COM port, in Linux and Mac this will be a /dev/xxx port. Depending on the type of platform you choose, you may also need to set various communication parameters. If these boxes are enabled, please check the documentation that came with your measuring platform to ensure these values are set correctly.

To check whether your platform is working, click the 'Start measuring test' button and attempt to measure a few rings. Once you are satisfied, click OK on the preferences window and you will be returned to the Corina home screen.

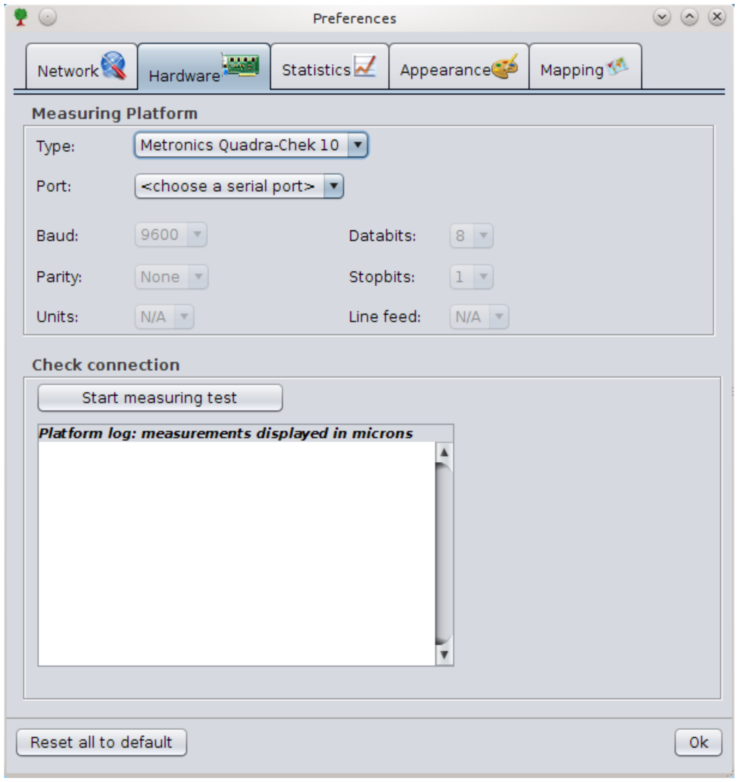


Figure 2.1: The hardware preferences dialog.

2.2 Differences in measuring hardware

Different measuring platforms have different capabilities. For instance, some include a physical switch for firing measurement events, others also include switches for resetting measurements to zero. Some platforms (e.g. Lintab) also continuously report the measurement values to the computer. So depending on the hardware you use, Corina will present the you with slightly different options.

Please note that the architecture of Corina is such that providing support for additional measuring platforms is a relatively simple task. If you have equipment that is not currently supported, please contact the developers to see if it can be included.

2.3 Measuring a sample

Once your measuring platform has been configured, measuring your first sample is simple. To start a new measurement go to *File* ➔ *New* or click the 'new' icon on the home screen. A dialog will appear where you can scan your sample's barcode, or press the button to enter metadata for your sample later. Barcodes minimize data entry errors and also speed up the process of measuring your samples. See section **??** for more information. Once you have scanned your barcode or pressed the button, you will then be presented with an empty Corina data screen (figure 2.2).

The next step is to fill out the metadata tab. If you have used a barcode, nearly all of this metadata will be filled in for you, otherwise you will need to fill this out yourself. Details about metadata can be found in chapter 3, page 13.

To begin measuring your sample you can now go to *Edit* ➔ *Start measuring* or you can press F5. While measuring you should be provided with audible feedback for each ring measured with a more pronounced sound made every 10th ring. If there is a problem communicating with your measuring hardware, check your settings

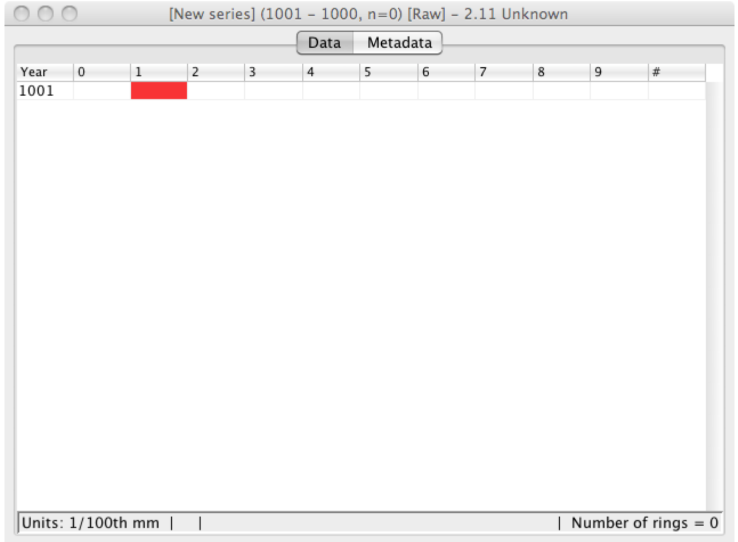


Figure 2.2: An empty data window ready to receive measurements. Note the status bar at the bottom includes buttons for changing the display units and cumulative statistics.

in the preferences dialog. If you still have problems contact the Corina developers by going to *Help* ➔ *Report bug on last transaction*, making sure you include your email address and any further information.

While you measure your sample you can flag features in a ring by right clicking on any cell in the table and selecting one or more of the standard notes (see figure 2.3).

Corina supports all standard TRiDaS remarks including: fire damage; frost damage; crack; false ring(s); compression wood; tension wood; traumatic ducts; single pinned; double pinned; triple pinned and many others. Rings that include remarks are indicated by the relevant icon in the data screen. Depending on your method of work, this can be useful for keeping track of sample pin holes. For instance, if a missing or false ring is discovered after a sample has been pinholed, the offset in pinholes can be easily seen without resurfacing the sample. In the future Corina will also include support for user defined ring remarks.

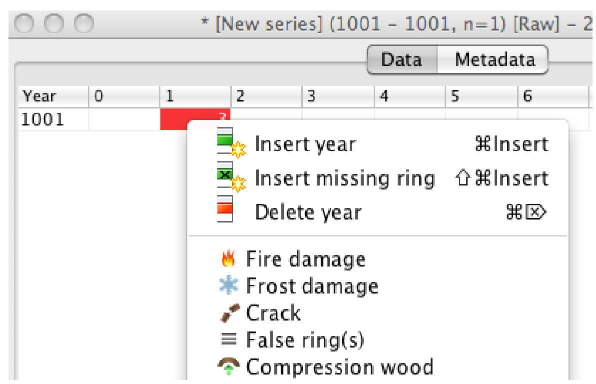


Figure 2.3: Right click context menu showing some of the options for adding remarks to rings.

The data screen also contains a status bar at the bottom. By click on the units section, you can switch between micron and 1/100th mm units. Corina understands the units being supplied by the measuring platform, therefore changes here are purely for display purposes only. If you have a platform that measures in microns, but prefer to see the values in 1/100th mm then you can use this feature. At the bottom ring of the status bar you can choose one of a variety of summary information about your series.

Once you have finished measuring your sample, you should then go to *File* ➔ *Save* to save your series to the database.

2.4 Reconciling data

Corina has been developed not only for experience dendrochronologists, but as a tool for teaching students. It therefore includes a comprehensive 'reconciling' tool for supervisors to check the quality of measurements made by students. The reconcile dialog does a comparison of a measurement series made by a student with a references series of the same radius measured by the supervisor. The same dialog can also prove useful for comparing measurements from two experienced dendrochronologists when handling particularly difficult samples.

Chapter 3

Metadata

Metadata is 'data about data'. In Corina this means all the information associated with your physical samples and measurement series e.g. species, location, who measured it, dimensions, slope, soil type etc.

The metadata in Corina, and in fact the entire Corina data model, is based on the Tree Ring Data Standard (TRiDaS). Before you use Corina you may find it useful to read **?** so that you get a better understanding of the principles of TRiDaS, but a summary is also provided here.

3.1 Tree Ring Data Standard - TRiDaS

TRiDaS is an XML-based data standard for recording dendrochronological data and metadata. More than 80 dendrochronologists, computer scientists and specialists from research disciplines that rely on dendrochronology have so far contributed to its development, including dendroarchaeologists, art and architecture historians, ecologists, geologists and climatologists. The standard is therefore capable of recording the wide variety of metadata required by these different fields. TRiDaS builds upon other established standards, such as GML for the recording of locality information. The extensible nature of XML also means that TRiDaS can evolve to accommodate the changing needs of dendrochronologists over time.

TRiDaS includes a total of eight data entities: project; object; element; sample; radius; measurementSeries; derivedSeries; and value. Detailed descriptions of each of these entities are given below and their relationships are illustrated in figure 3.1.

A project is defined by a laboratory and encompasses dendrochronological research of a particular object or group of objects. Examples include: the dating of a building; the research of forest dynamics in a stand of living trees; the dating of all Rembrandt paintings in a museum. What is considered a "project" is up to the laboratory performing the research. It could be the dating of a group of objects, but the laboratory can also decide to define a separate project for each object. Therefore, a project can have one or more objects associated with it. Due to the way research is conducted at the Cornell Tree-Ring Lab, TRiDaS projects are not currently supported within Corina, although future plans include adding project support.

An object is the item to be investigated. Examples include: violin; excavation site; painting on a wooden panel; water well; church; carving; ship; forest. An object could also be more specific, for example: mast of a ship; roof of a church. Depending on the object type various descriptions are made possible. An object can have one or more elements and can also refer to another (sub) object. For instance a single file may contain three objects: an archaeological site object, within which there is a building object, within which there is a beam object. The list of possible object types is extensible and is thus flexible enough to incorporate the diversity of data required by the dendro community. Only information that is essential for dendrochronological research is recorded here. Other related data may be provided in the form of a link to an external database such as a museum catalogue.

An element is a piece of wood originating from a single tree. Examples include: one plank of a water well; a single wooden panel in a painting; the left-hand back plate of a violin; one beam in a roof; a tree trunk preserved in the soil; a living tree. The element is a specific part of exactly one object or sub object. An

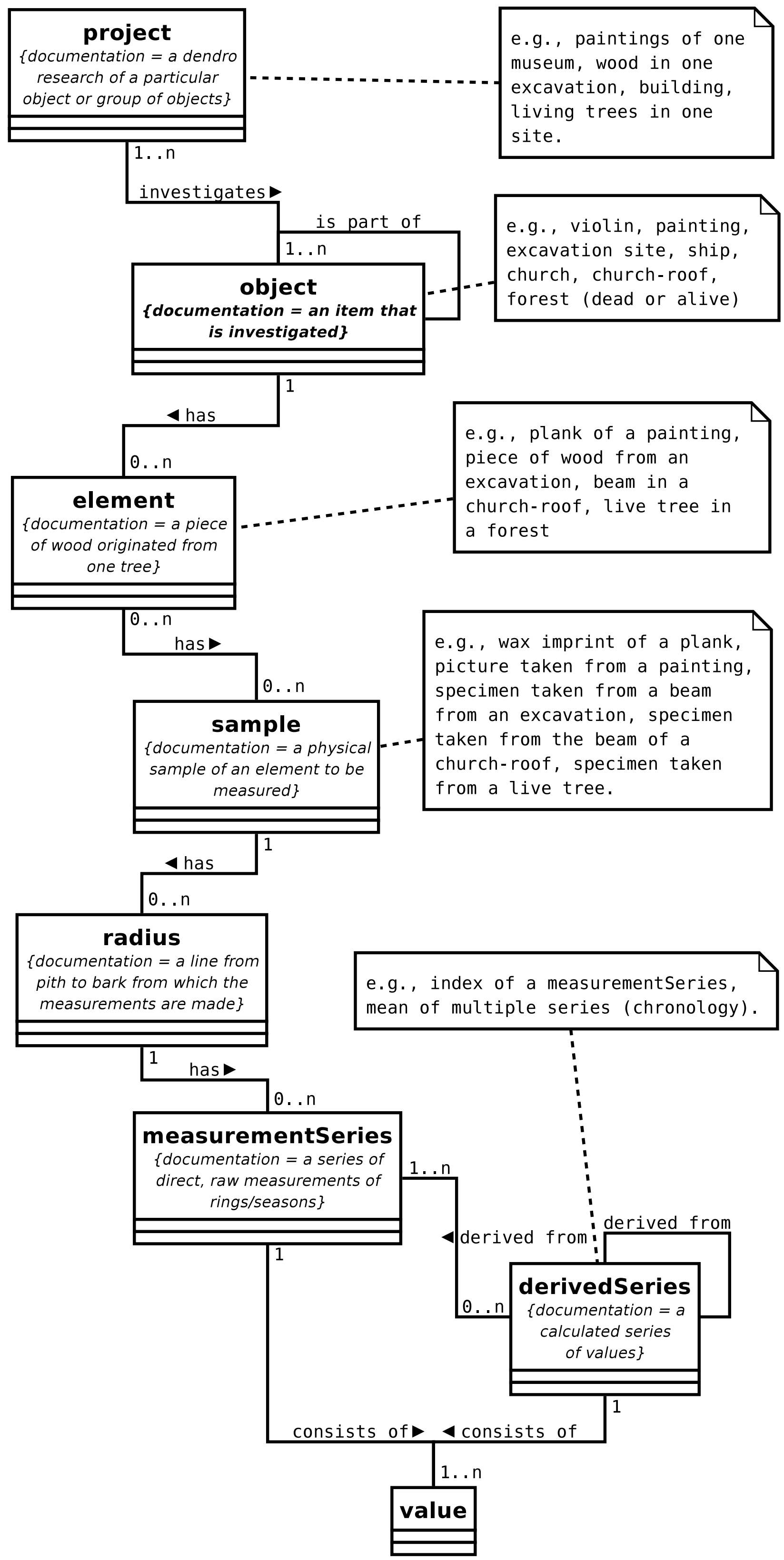


Figure 3.1: TRiDaS data model showing the relationships between data entities.

object will often consist of more than one element, e.g., when dealing with the staves (elements) of a barrel (object). One or more samples can be taken from an element and an element may be dated using one or more derivedSeries.

A sample is a physical specimen or non-physical representation of an element. Examples include: core from a living tree; core from a rafter in a church roof; piece of charcoal from an archaeological trench; slice from a pile used in a pile foundation; wax imprint of the outer end of a plank; photo of a back plate of a string instrument. Note that a sample always exists and that it can either be physical (e.g. a core) or representative (e.g. a picture). A sample is taken from exactly one element and can be represented by one or more radii.

A radius is a line from pith to bark along which the measurements are taken. A radius is derived from exactly one sample. It can be measured more than once resulting in multiple measurementSeries.

A measurementSeries is a series of direct, raw measurements along a radius. A single measurementSeries can be standardised or a collection of measurementSeries can be combined into a derivedSeries. The measurements themselves are stored separately as values.

A derivedSeries is a calculated series of values and is a minor modification of the "v-series" concept proposed by **?**. Examples include: index; average of a collection of measurementSeries such as a chronology. A derivedSeries is derived from one or more measurementSeries and has multiple values associated with it.

A value is the result of a single ring measurement. Examples include: total ring width; earlywood width; latewood width. The values are related to a measurementSeries or a derivedSeries. In case of a measurementSeries the variable and its measurement unit (e.g. microns, 1/100th mm etc) are recorded as well.

Working top to bottom, the TRiDaS entities are nested within each other. For instance a project contains one or more objects, which in turn contains one or more elements, and so on. The benefit of this is that you record data once and once only. In standard file-based dendrochronological software, when creating measurement series you are typically required to type the name of the site, the species of tree etc over and over again. This is not only time consuming, but very error prone.

Keeping data consistent is also difficult. For instance, if it was determined that a tree species was identified incorrectly, in existing file-based software, the user would need to locate all data series from this tree and manally update the metdata. This is not the case in Corina. A tree is represented just once in Corina and samples of this tree, and the subsequent measurement series reference this one entry. If metadata for this tree needs to be changed, the tree record is updated in just this one place. Because the measurement series obtain this information by reference, then all associated series are automatically kept up to date.

3.2 Entering sample metadata

The metadata for a series is viewed and edited on the 'Metadata' tab of the main window such as that shown in figure 3.2. You can see the interface is organized according to the TRiDaS data model with separate screens for object, through to series.

When creating a new series, the metadata screens must be populated in order. This is necessary because of the nesting of entities described above. For instance, an element is associated with an object, so an object must be chosen because and element can be defined. Likewise, an element must be chosen before any samples of this element can be defined.

Much of the time the entities that you need will already be stored within the database. Instead of re-entering data, you simply need to select the existing entry from the database, saving a great deal of time. Depending on the situation buttons will appear at the top of the dialog to let you 'choose' an entry from the database, 'revert' to the previously chosen entry, 'change' the existing entry to a different one from the database, or create a 'new' record.

Please note that the content of these metadata screens is kept read-only by default. To edit the values, you must first click the padlock icon to unlock the fields. When you have finished making changes you need to press the save button to write the changes to the database before moving to another metadata screen.

Very few of the metadata fields in the TRiDaS data model are mandatory, but a few are. In this case, these fields are highlighted with a red background. Note that whether a field is mandatory or not can depend on

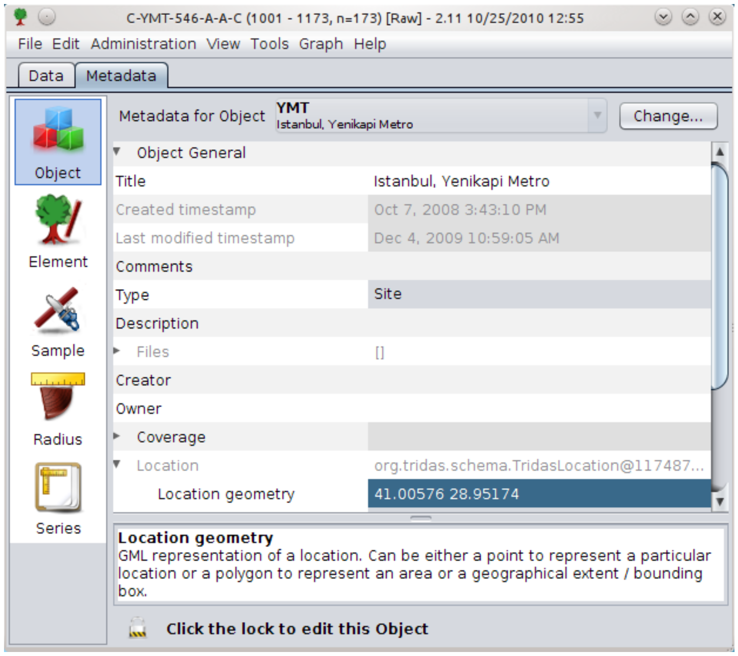


Figure 3.2: Example of the metadata dialog. The screen is showing the details of a TRiDaS object. Note that the location geometry field is highlighted and so a description of what is expected in this field is given below.

the other fields that have been filled in. For instance, the dimensions of an element are not required, but if dimensions are given then the units for these measurements must also be provided.

A number of the metadata fields are restricted with regards the values that you can enter. These are known as 'controlled vocabularies' in TRiDaS terms. Controlled vocabulary fields are represented by drop down menus. Similarly fields that expect numerical values (such as element dimensions) will only allow numbers. The final method data entry method is through custom dialogs. The only custom dialog currently implemented is for locations. This accepts coordinates in either decimal degrees or degrees minutes and seconds. Alternatively you can use data from a GPS handset by providing a GPS Exchange (GPX) format file containing the waypoints. The GPX format is the most common interchange format for GPS data. You can pick the relevant waypoint from the drop down menu. You can also preview the defined coordinates on a map using the 'view on map' button.

> 💡 A popular open source GPS communication tool is GPS Babel. It is an easy to use application which can download data from the majority of GPS handsets. See *http://www.gpsbabel.org* for more information.

3.3 Entering bulk metadata

Entering metadata on a sample-by-sample basis works perfectly well, but does not necessarily fit best with the typical workflow of a laboratory. Samples do not typically arrive in a lab in ones and twos, rather in large quantities following a field excursion. In this case it is most efficient to enter all the metadata for the samples as they arrive. This is often best in terms of data accuracy as the metadata can be entered while the field notes are still fresh in the mind.

To enable the efficient entry of lots of metadata Corina includes the bulk data entry interface. This can be access from the file menu and is illustrated in figure 3.3. There are three pages, one each for objects, elements and samples.

The interface is designed like a spreadsheet so as to be as familiar to users as possible. Each row of the table represents a new entry in the Corina database. Which columns are shown to the user is determined by the 'show/hide columns' button on the top right of the screen.

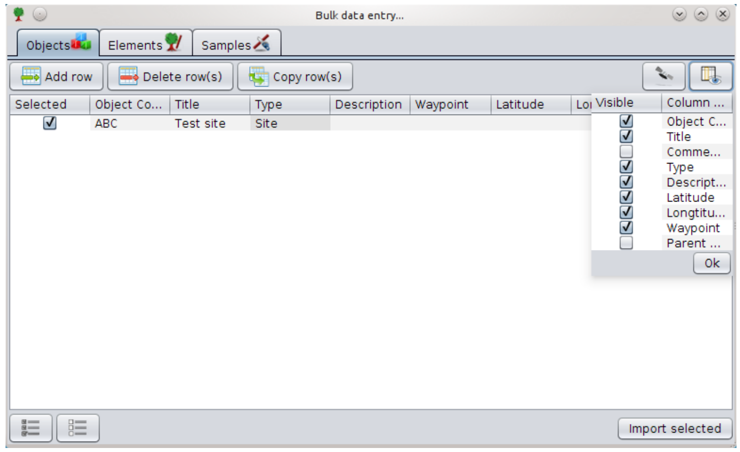


Figure 3.3: The bulk metadata entry screen. The 'show/hide columns' button has been pressed showing how the user can turn on and off particular columns.

The bulk entry interface also includes support for reading GPS units. By pressing the satellite button on the toolbar, the user can provide a GPS Exchange (GPX) format file containing the waypoint locations recorded in the field. Corina will add a waypoint column to the spreadsheet with a drop down menu which will automatically populate the latitude and longitude fields for the record.

It is common for many of the metadata fields to be same in a single field collection. For instance, when coring trees in a forest, they are often of the same species. Rather than requiring the user to repeatedly type the same data over and over, the 'copy row' button can be used to duplicate a record, and then the user can change the few fields that are different.

When you have entered all the data you want, you can press the 'Import selected' button to write the records to the database.

3.4 Metadata browser

The metadata browser interface provides a convenient way to view all the metadata within your Corina database. It can be accessed through the 'Administration' menu.

The metadata browser contains two parts: a hierarchical representation of all TRiDaS entities in your database on the left; and a metadata viewer for the selected entry on the right. This interface is also the best method for fixing mistakes in your database.

Although Corina's database architecture maintains integrity within your data, it does come at the price of being a little more complicated to fix mislabelled series. For instance, what if you were to measure a series 'B' and assign it to sample ABC-138-A only later to realize you misread the label and it was in fact ABC-188-A. In a traditional file-based system, you would probably just need to rename the file you'd just created. In Corina however, you need to redefine the relationship of the series within the database and reassign it to the create sample. This is best understood when looking at the hierarchical tree in the metadata browser. Hopefully you will see that you what you need to do is to move the series from its current position in the database to the correct one.

The reorganization of data in this way is achieved by right clicking on items in the hierarchical tree and choosing with 'merge' or 'reassign'.

Chapter 4

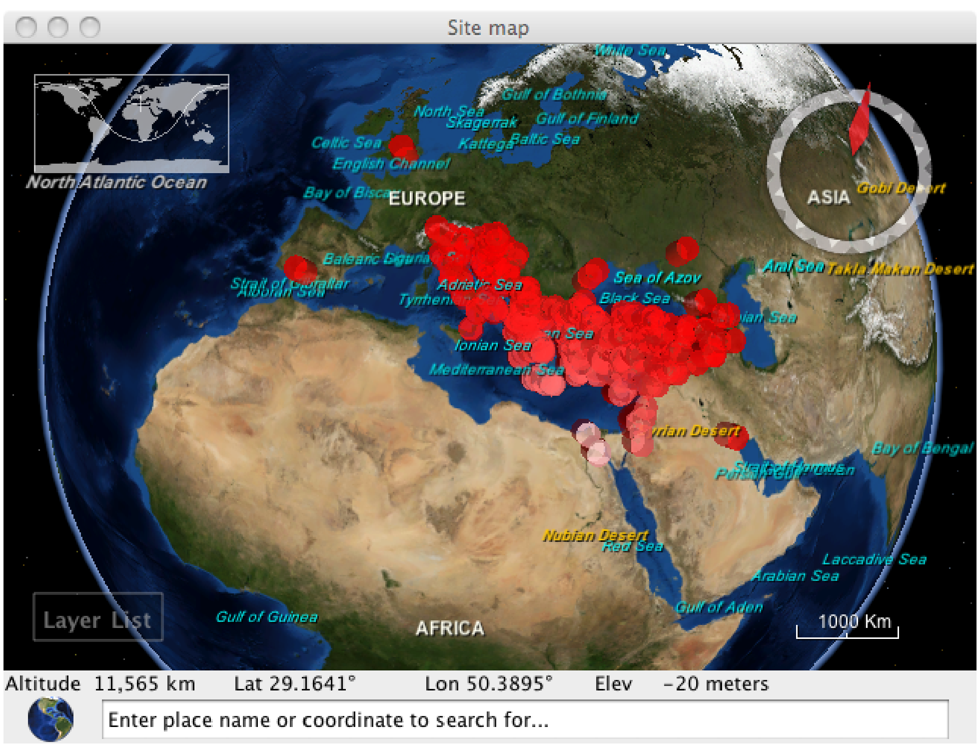
Mapping

Corina includes an integrated open source 3D mapping system (based on NASA's award winning World Wind Java SDK) similar to the program Google Earth which you're no doubt familiar with. As mentioned in the installation chapter, this mapping system requires an OpenGL 3D capable graphics card.

Before you can use the mapping in Corina, it must have something to map! See the chapter on Metadata (page 13) for information about adding coordinates to your system.

There are two ways to map data from your database. First of all, you can see a map of all the sites (i.e. TRiDaS objects) by going to Administration, then Site map. This will give you a screen like this:

Figure 4.1: Screenshot showing an example of a site map.



You can also see a map of your current series if you have latitude/longitude metadata by clicking on the map tab on the main data screen.

4.1 Navigation

You can navigate around your maps using both your mouse and keyboard.

4.1.1 Mouse with scroll wheel

Pan Left mouse button click and drag – all directions
Zoom Use the scroll wheel on the mouse or Left and Right mouse (both buttons) click and drag up and down
Tilt Right mouse button click and drag – up and down or use 'Page Up' and 'Page Down' on the keyboard.
Rotate Right mouse button click and drag – left and right Note: Crossing the top and bottom half of the screen while rotating will change direction.
Stop Spacebar
Reset Heading N
Reset all R

4.1.2 Single button mouse

Pan Left mouse button click and drag - all directions. L left mouse button click once to center view.
Zoom Hold 'Ctrl' on the keyboard and Left mouse button click and drag - up and down
Tilt Hold 'Shift' on the keyboard and Left mouse button click and drag - up and down or use "Page Up" and "Page Down" on the keyboard.
Rotate Hold 'Shift' on the keyboard and Left mouse button click and drag - left and right
Stop Spacebar
Reset Heading N
Reset all R

These controls enable you to explore your location information in 3D such as the example of Mount Vesuvius in figure 4.2.

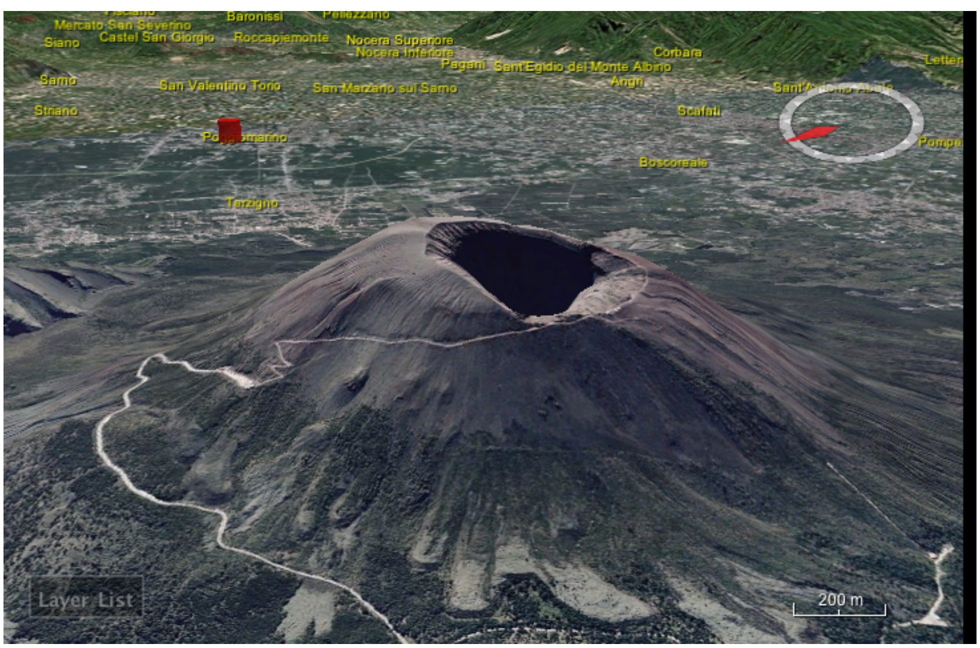


Figure 4.2: Example of 3D mapping in Corina.

Another method of navigating around the map is by using the built in gazetteer. You can enter and place name or coordinate information into the box at the bottom of the screen and you will fly to the requested location.

4.2 Interacting with data

Each marker on the map represents either a TRiDaS object or element in your Corina database. By clicking on these pins you can get more information from the database (see figure **??**).

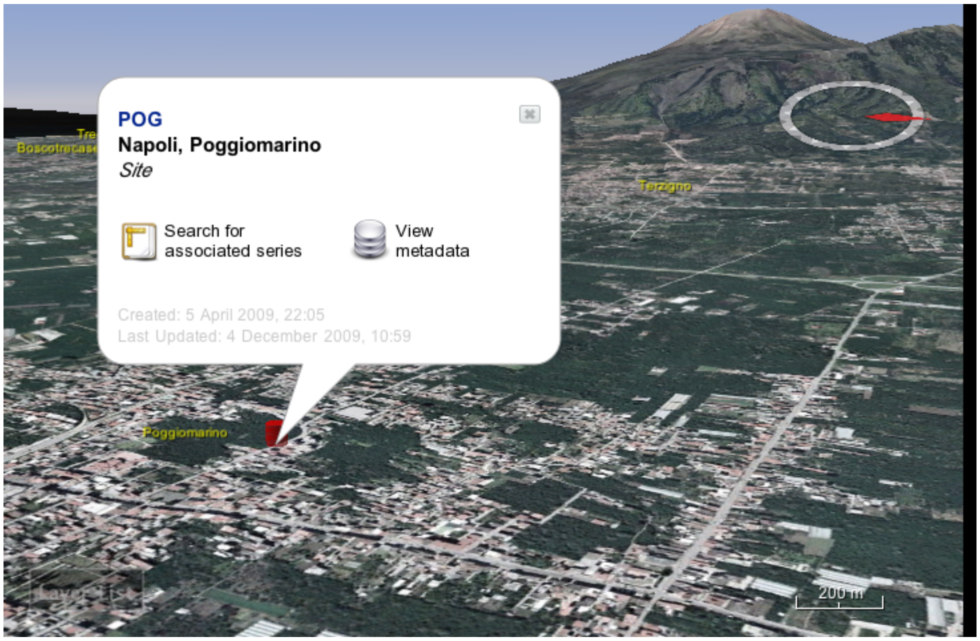


Figure 4.3: Screenshot of a map with information pin expanded

The example above shows the ring marker is of a site in Napoli called Poggiomarino (code name POG). You can see the option for searching for all series in the database associated with this site, and also the option for viewing all the metadata.

4.3 Background map layers

Corina comes ready configured with basic map layers, including high resolution satellite imagery and basic political features. You can turn background layers on and off by going to View ➔Layers.

Map layers are downloaded on-the-fly so there is likely to be a delay when you initially visit to a new region. However, up to 2Gb of map data can be cache locally to your hard disk, so on future visits, maps should load quickly.

The mapping system in Corina includes support for remote map servers that use the OGC Web Mapping Service (WMS) standard. If you go to View ➔Layers ➔Add remote layers, you will get a dialog with a tab for each WMS server configured for your system. By default this includes the NASA Earth Observation and Jet Propulsion Lab servers. By ticking layers in this list you can add data layers to your map.

You can add map data from other WMS servers by clicking the '+' tab and entering the URL of the server you would like to use. This will give an additional tab with all the available map layers. This server will only be available for the duration of your current session so will need to be added each time you start Corina. If you would like a particular WMS server to be made permanently available, your Corina administrator can do this (see 'Managing map services', on page 41 for further details). Additional WMS servers added in this way will be available to all users the next time they connect to your Corina server.

Your system administrator may host a map server specifically for your lab, for instance, containing high resolution plans of an archaeological site that you are working on, or environmental data for your study region. Figure **??** shows an example overlay of sea surfaces temperatures loaded dynamically from the NASA EO server.

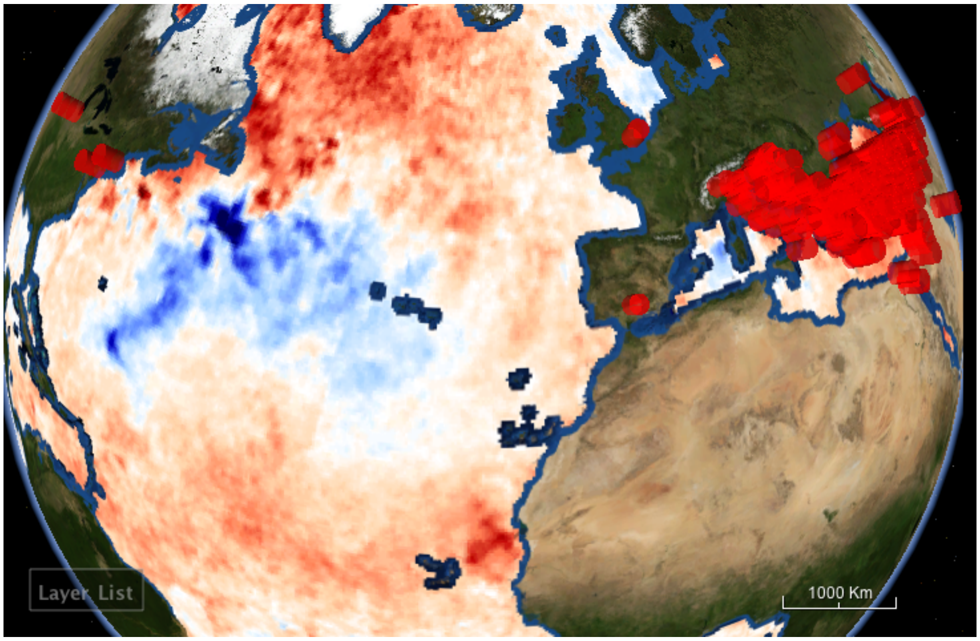


Figure 4.4: Map screenshot with a NASA sea surface temperature overlay dynamically loaded from the NASA WMS server.

4.4 Data map layers

Data map layers (i.e. site and sample locations) are controlled with the layer list in the bottom left of the screen. When viewing series, you will have the option of adding layers containing points for all the other series at the current site, and showing all the sites in the database.

4.5 Exporting maps

You can export maps by going to File ¿ Export map as image. For best results, maximize your map window first. You may also like to turn off various map widgets by going to the View menu. The exported image will include everything you can see on your map screen.

Chapter 5

Graphing

The graphing component is reused in many places throughout the Corina desktop application. The following description although based on the main graphing screen in Corina is largely applicable to all dialogs that include graphs (e.g. crossdating, indexing and reconciliation).

The main method for graphing your tree-ring data is by choosing an option from the Graph menu. Depending on the type of series you have open, the options available to you will be different. For raw measurement series, you will just have the option to 'Graph active series'. This will give you a simple graph of the current series that you have open. If you have a derived series open, then you may also choose 'Graph component series' which will plot all the series that go to create this series, or 'Graph all series' which graphs all the component series as well as the current series.

5.1 Controlling graphs

When newly created graphs are plotted according to the scale on the axes. A feature of Corina graphs though is that they can be manipulated directly on the screen. Both dendrochronology was computerized, dendrochronologists would plot rings manually on to graph paper. These paper graphs were then placed on lightboxes and moved around to enable comparisons. The graph function in Corina emulates this behaviour allowing users to click and drag graphs around to test for visual matches.

Figure 5.1 shows an example graph dialog. The mouse is hovering of the blue measurement series at relative year 1040 illustrating Corina's highlighting and guide line capabilities. A feature not shown in this screenshot is the illustration of sapwood rings. When sapwood rings are present the corresponding years on the chart are denoted via a heavier line.

The layout of graphs can be changed using both the toolbar buttons and menu options. The type of graph can be changed between a standard line graph, a semi-log graph and a toothed graph using the radio buttons. The remaining buttons are as follows:

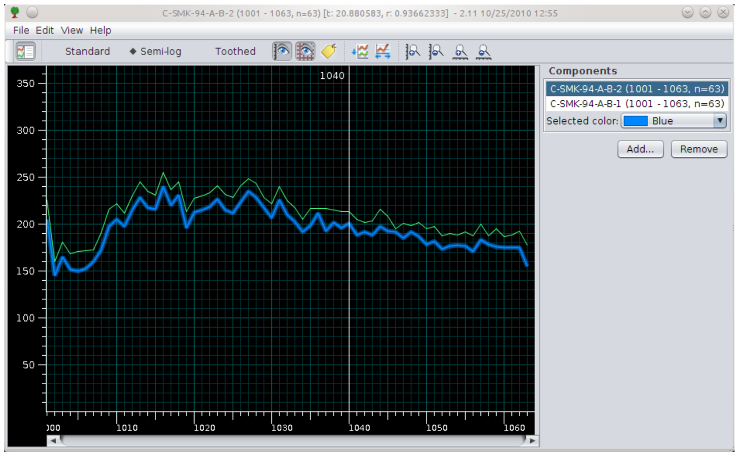


Figure 5.1: An example graph window containing two undated series of the same sample on a semi-log graph. Note the legend is visible with the options for adding or removing series.

- Zoom in on the horizontal axis
- Zoom out on the horizontal axis
- Zoom in on the vertical axis
- Zoom out on the vertical axis
- Toggle show/hide the grid lines
- Toggle show/hide the series labels
- Toggle show/hide the vertical axis
- Spread the series evening up the vertical axis
- Set the baselines of all the series to zero
- Resize graph to fit horizontally
- Toggle show/hide the legend

There are also a number of keyboard shortcuts that you might find useful:

Tab : Cycles through each graph component
Ctrl+W : Increase vertical scale
Ctrl+S : Decrease vertical scale
Ctrl+A : Increase horizontal scale
Ctrl+D : Decrease horizontal scale
Up arrow : Moves selected graph up by 10 units
Down arrow : Moves selected graph down by 10 units
+ : Moves selected graph up by 1 unit
- : Moves selected graph down by 1 unit
HOME : Scroll to first year of series
END : Scroll to last year of series
PAGE UP : Scroll left by one page width
PAGE DOWN : Scroll right by one page width
SPACE : Sets horizontal origin of all graphs to the same value

5.2 Exporting graphs

To export your graphs for use in reports you can go to File ➔Export plot as PDF file, or Export plot as PNG file. This presents you with a dialog for setting the colors, labels and size of the exported image. This functionality is due for an overhaul in the future to provide more flexible support for publication quality graphics.

Chapter 6

Importing and exporting

Importing and exporting of dendro data in Corina is provided through the TRiCYCLE libraries. TRiCYCLE is a universal dendro data conversion application for converting back and forth between over 20 supported data formats (**?**). The open source libraries that provide the functionality to TRiCYCLE are incorporated directly into Corina providing support for all these formats.

Belfast Apple	Oxford
Belfast Archive	PAST4
Besancon (including SYLPHE variants)	Sheffield D-Format (Dendro for Windows)
CATRAS	Topham
Comma delimited text files (CSV)	TRiDaS
Corina Legacy	TRIMS
DendroDB	Tucson (RWL and CRN)
Heidelberg (TSAP-Win)	Tucson Compact
Microsoft Excel 97/2000/XP	VFormat
Microsoft Excel 2007	WinDENDRO
Nottingham	

Table 6.1: List of the twenty-one formats supported by Corina. See appendices A – V (pages 55 – 117) for full descriptions.

6.1 Exporting data

Exporting data is initiated by the File ➔Export data menu. If this is called from the main Corina data window, it will export the current series. If it is called from the Corina home screen, then it will present you with the database browser and allow you to pick one or more series to export.

6.2 Exporting graphs

6.3 Exporting maps

6.4 Importing data

Chapter 7

Curation

7.1 Laboratory workflow

Corina includes a number of functions to assist you with the curation of your physical sample collection. To understand how these are designed to assist users, we must first consider the workflow within a laboratory.

In research laboratories, samples generally come to the lab in large batches following field collection. In this case the typical workflow may be as follows:

1. Collect samples and record field notes as accurately as possible
2. On returning to the lab enter field notes as soon as possible into the 'bulk data entry' interface
3. Print sample barcode labels
4. Prepare physical samples and label with barcodes
5. Assign samples to storage boxes
6. Measure samples, using barcodes to recall metadata from database
7. Crossdate samples / build chronologies
8. When all samples from a box are completed register box as archived and then store

For commercial labs offering dendrochronological dating as a service, samples more likely to arrive in smaller batches. In this case, the bulk data entry interface may not be the most efficient method for entering metadata. In this case the user may simply prefer to use the *File* ➔ *New* method for each sample.

Either way, the concept behind the curation of a collection in Corina revolves around the accurately recording as much metadata about a sample as possible, then labelling the physical sample with a label containing a barcode for Corina and sample code for the user. By entering a sample into the database as soon as it enters the lab, it can be traced throughout the workflow. When a chronology is built, it is easily to quickly and efficient locate all samples that have been used. By assigning samples to boxes, groups of similar samples (e.g. from the same site) can also be easily stored together and located quickly and efficiently.

7.2 Barcodes

Barcodes allow you to keep track of what samples you have and where they are stored. Although it is not essential to use the barcode functions, we strongly suggest you do because they save time and money, but most importantly they greatly reduce the scope for erroneous data entry. For instance, when measuring a sample a user simply scans its barcode and all the relevant metadata is retrieved from the database, rather than relying on them to enter data manually. Barcodes have been routinely used in the retail industry since the 1980s. They can be equally as useful in dendrochronology laboratories.

Corina creates and reads barcodes for samples, measurement series and boxes. Each barcode encodes the unique identification code stored in the Corina database for each of these entities. Due to Corina use of universally unique identifiers (UUIDs), these codes a garanteed to be unique opening the opportunity of labs to loan samples, much like libraries do with books. There are many styles (or 'symbologies') of barcodes in use today, but Corina uses one of the most common (Code 128) which is supported by the vast majority of barcode readers. For a detailed discussion on the specifications of the Corina barcode see section 13.4.

Basic barcode readers are now cheap and widely available, with basic devices retailing for a few tens of dollars. Most are characterised as 'keyboard interface devices' and work like an automated keyboard, typing in a string of characters when a label is scanned.

Within the Corina application, whenever the user is required to specify a box, sample or series, they have the option of typing the human readable lab code or scanning the barcode. By using the barcode, the user can be sure they are not entering typographic errors so we recommend using barcodes whenever possible.

The most important barcode is the label for the physical wood sample. These are easily generated through the *Administration* ➔ *Labels* ➔ *Sample labels* menu entry. Currently the layout of these labels is fixed, but in the future we aim to provide different styles.

7.2.1 Sample labels

Before labels can be generated, metadata entries the sample level must have been made in the database. This is typically done using the 'bulk data entry' interface (see page 16). If samples are already in the database, the user needs to select the object of interest in the label creation dialog to see all the available samples. It is then just a matter of selecting the samples of interest and moving them into the 'selected' column. Once the list is populated (samples from multiple objects can be included), then you can either click 'Preview' to see a PDF of the labels, or 'Print' to print directly.

The current label style is designed to fit on standard core mounts and most samples. There are no widely available die-cut labels that fulfill this need, so the labels are intended to be printed on archival grade full page sheet labels (e.g. Avery® layout 6575), and then manually guillotined.

7.2.2 Box labels

The procedure for printing box labels is the same as for samples. Samples must have already been assigned to boxes before the label is printed (see section 7.3 for details). To print (or preview) box labels go to *Administration* ➔ *Labels* ➔ *Box labels*. The label style is designed to be printed on $5'' \times 8\frac{1}{8}''$ labels, two per sheet such as the Avery® 6579 layout.

> Until dynamic label styles have been implemented, box labels will print one per page. To make use of the second label on the page, the same sheet should be fed through the printer a second time.

7.2.3 Series barcodes

Series barcodes are printed at the top of a standard series report. These are produced through the *File* ➔ *Print*, or *File* ➔ *Print preview*, menus.

7.3 Storage boxes

Corina uses the term 'box' to refer to the collection of samples you archive. Many labs (including Cornell) use cardboard bankers boxes to store samples once they are completed, but the same box concept could refer to draws or shelves in your collection.

7.3.1 Creating and editing boxes

Records for boxes in the system are created and edited through the *Administration* ➔ *Curation* ➔ *Box details* menu. To editing an existing box, you can scan the barcode label on the box, or select from the list. To create a new box, click the 'Create new box' button and enter its details. There is no restriction on what boxes should be called, but it is probably easiest if you use some sort of numerical sequence to assist with organizing the boxes in your store. At Cornell, we use a two part name for each, the first being the year of collection, the second being a sequential number (e.g. 2009-11).

The contents tab lists all the samples that have been assigned to this box. To add new samples, simply click the 'Add sample to box' button and scan the sample's barcode.

7.3.2 Inventory

An important feature of any collection management system is the ability to perform an inventory on the collection. Even with the most robust system, samples will always go astray so its important to be able to periodically check that the boxes contain what you expect.

The 'Contents' tab of the Box details dialog contains a feature to assist with this. Next to the list of samples that are recorded as present, there is a temporary checklist column. By checking the boxes for each sample actually stored in the box it is easy to see which samples have been mislaid. If the 'Mark unchecked as missing from box' button is then pressed, the date and time the discrepancy was noted is then recorded in the comments field for the box.

7.3.3 Checking boxes in and out

Corina includes function for checking boxes in and out of a store, much like when a book is borrowed from a library. The *Administration* ➔ *Curation* ➔ *Check out box from store* and *Administration* ➔ *Curation* ➔ *Return box to store* menus do just this. You can either scan the box barcode or select the box from the drop down menu. These options record when a box is checked out/in and by whom. These details can be seen by users in the box details dialog.

7.3.4 Locating samples

As you might expect, Corina also includes a function for locating your physical samples. This is available in the *Administration* ➔ *Curation* ➔ *Find a sample* menu. There are three methods for locating a sample: via barcode; via lab code; and manually by object/element/sample.

If you have the sample in your hand and you simply want to know which box it should be returned to you can scan the barcode. If you are looking for a sample and you know its lab code then you can enter this instead. Alternatively, you can use the drop down menus to search for one or more samples at once. For instance, you can locate all the samples for a particular object and element.

Chapter 8

Indexing

Trees tend to put on big rings when they're young, and smaller rings when they get older. Some trees put on very large rings, while others put on very small rings. These variations in growth can make it difficult to crossdate samples. Some dendrochronologists therefore prefer to index or normalize their ring width data before combining into chronologies.

Indexing is a manipulation you can perform on your data to make it easier to crossdate.

The procedure for indexing is as follows:

1. You open a series (raw data)
2. You ask Corina to index it
3. Corina shows you some possible curves
4. You pick a curve (based on its graph, statistical scores, and your expectation of how the tree is growing)
5. Corina converts each year's ring width to a ratio of actual growth to expected growth for that year
6. You save the series (indexed data)

Indexing changes the units of a dataset. A raw sample has units of hundredths of a millimeter (0.01 mm) or microns. An indexed sample has units of parts per thousand (0.1%, or ‰).

This doesn't cause a problem with crossdating. The t-score normalizes all samples as part of its test, and the trend only cares if the values are increasing or decreasing. For more information on crossdating and chronology building, see chapter 9. It does, however, cause a problem with 'summing' since summing needs to take the average (what's the average of 1mm and 75%?). Therefore, the samples in a sum must be either all raw, or all indexed.

8.1 Types of index

There are a total of six different indexing methods available in Corina:

8.1.1 Exponential Index

This is the most commonly used index as it matches the way trees typically grow. Quickly when young and then gradually slower. An exponential index is therefore by far the most common index you'll use as 9 times out of 10 this will be the best choice.

This index tries to fit an equation of the following form to your data, searching for the best values of a, b and p.

- $y = a + be - px$

This is sometimes called a negative exponential index, because the exponent is negative. Corina doesn't require that the exponent is negative, but if it's not, using this index probably isn't such a good idea; it means the tree is generally getting bigger, not smaller.

The least-squares algorithm used comes from **?**; the matrix solving function comes from **?**.

Sometimes the exponential index does a lousy job. If a tree is living in a crowded area and the trees around it get cut down, suddenly it has much better growing conditions, so it might grow faster as it gets older, instead of slower. If you tried to use an exponential curve on a tree like this, it would exaggerate this growth, and useful data would get flattened out.

The result is you're looking at the growing conditions of this one tree, so it's not going to crossdate as well.

Alternatively, imagine you are working on a tree with a fire scar that has a few very large rings. An exponential index wouldn't take much notice of this, because most of the sample is still shaped like an exponential curve, but when you applied it they would be grossly out of proportion. For these types of samples, there are other indexing algorithms available.

8.1.2 Polynomial Index

When you ask Corina to perform a Polynomical Index it tries to fit a polynomial curve to your data using the following equation:

- $y = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_{1x} + a_0$

You decide what degree polynomial, n, to use and Corina automatically finds the best values of a_0, $a_1 \ldots a_n$, to fit your data.

8.1.3 Horizontal Line Index

This only changes the magnitude not shape of the curve and is used when you would link to combine raw and indexed data together. It is a special case of polynomial where the horizontal line is equal to the average value.

- $y = x_{avg}$

This index is not used for crossdataing because dividing each value by the same value doesn't change the shape of the curve, only its magnitude. A horizontal line index is, however, useful because every element in a sum must use the same units, either raw or indexed. Therefore if you want to include a raw sample with an indexed sample then a horizontal line index can be used to convert the raw sample without otherwise altering the shape of the curve.

8.1.4 Floating Index

This is a running average of the 11 surrounding years. The adaptive index is generally used as a "last resort" when both exponential and a high-degree polynomial have failed. It is simply the average of the eleven surrounding years:

- $ind_i = 1/11(data - i - 5 + data_{i-4} + \ldots + data_{i+4} + data_{i+5}$

This index was originally called floating average, probably in reference to the fact that the index curve "floats" around, not following any explicit $y = f(x)$-type formula. But people tended to call it floating, and then floating-point, which means something very different. You might still hear people calling this index by these other names.

8.1.5 High-Pass Filter Index

The high-pass index is a more general case of the adaptive index. Instead of simply taking the average of 11 values, it takes a weighted average. It's an example of a "high-pass" filter because high-frequency signals can pass through, but low-frequency signals are filtered out.

The default is "1-2-4-2-1", meaning:

- $ind_i = 1/10(data_{i-2} + 2{\cdot}data_{i-1} + 4{\cdot}data_i + 2{\cdot}data_{i+1} + data_{i+2})$

This comes from **?** who used it as a discrete filter before moving to a cubic spline. Note that almost half $(4/10)$ of the computed value is simply its old value. The high-pass index is nearly the same as the input, so the χ^2 values are usually the lowest, therefore do not choose this index solely on a low χ^2 value.

8.1.6 Cubic Spline Index

Cubic splines are a very specific type of high-pass filter. A cubic spline curve is created by combining a collection of cubic (3rd degree polynomial) functions.

There are many methods for constructing cubic splines through a dataset. The algorithm used by Corina has a parameter, s, which controls how tightly the spline fits the data. A lower value fits the data more tightly, a higher value fits the data more loosely. Therefore, s=0 fits the data exactly while s=1 is a simple line. A good starting point for dendro data seems to be around $s = 1x1016$.

Cubic splines were first used for dendro by **?** using an algorithm from **?**.

You can change the s-value used for the subic spline in the preferences. You might use a cubic spline in the same cases you would use a high-pass filter e.g. when the sample doesn't generally follow an exponential or polynomial curve very well, perhaps due to a fire scar.

8.2 Indexing data

To index your data, first you need to open the series you would like to index. Next choose *Tools* → *Index* to display the indexing dialog (figure 8.1).

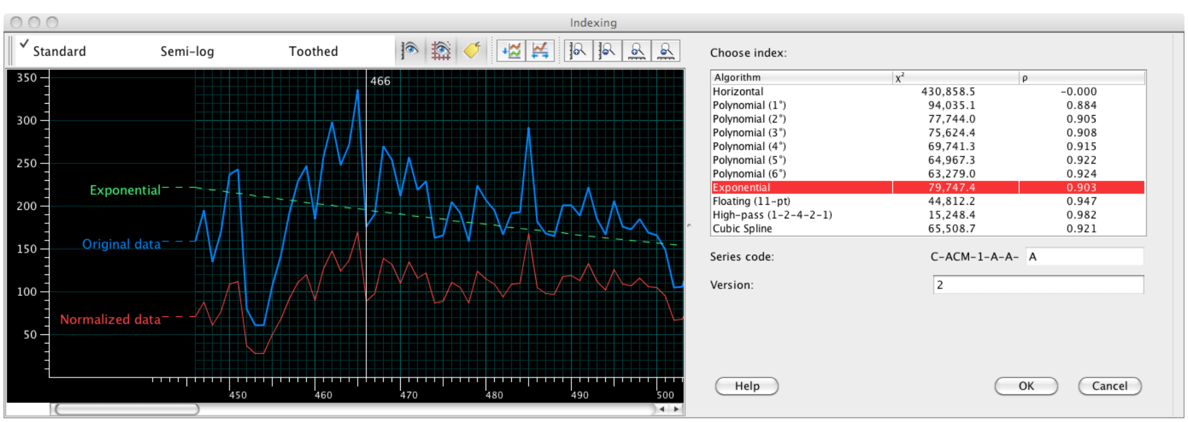


Figure 8.1: Indexing dialog showing the original data in blue, the exponential index of this data in green, and the normalized data in red.

From the indexing dialog you can then choose which type of index to apply to your data. The table on the right show the available options along with the χ^2 and p values to help you choose the correct index to use. The graph shows your original data, the index line and the result of applying the index to the data and changes dynamically as you pick between different indexing methods. Once you have decided which index you want to use, select it, and click OK ensuring that you have given your data series a new version number.

Chapter 9

Crossdating and chronology building

All algorithms work in pretty much the same way. There's a "fixed" sample, and there's a "moving" sample. Imagine you have printouts of their graphs on translucent paper. The fixed graph is taped to a table, and you can slide the moving sample left and right. This is actually how it was originally done, on graph paper, with one inch per decade. Start with the moving sample to the left of the fixed sample, overlapping it by 10 years. Look at how well the graphs match: this is the first score that's computed. Slide the moving sample to the right one year and so on until you reach the end.

You could do it all simply by moving graphs and eyeballing the crossdates like this but there are hundreds of sites and millennia of chronologies you'll want to crossdate your samples against, so that would take a while. Corina has a few algorithms to find likely crossdates almost instantaneously. They aren't perfect, though, and all crossdates should be inspected visually to ensure they are a good fit.

9.1 Algorithms

Corina includes a total of five different algorithms for crossdating:

9.1.1 T-Score

The t-score is the classic crossdate. Everybody quotes t-scores: if you want to brag about how good a cross is, you tell them your t-score. Unfortunately, every dendro program seems to have a slightly different implementation of tscore, so the numbers you get from Corina might not be exactly comparable to the numbers from other programs.

The version Corina uses is based on the algorithms given in **?**, though with some apparent bugs corrected. In the following equations, $x_0, x_1, x_2, \ldots$ are the data of the fixed sample in the overlap, $y_0, y_1, y_2, \ldots$ are the data of the moving sample in the overlap, and N is the length of the overlap.

The first step is to make each dataset bivariate normal by replacing each value with the mean of the values around it, and then taking its natural logarithm. The preparation for the T-Score is therefore done as follows and is done to both the fixed and moving series:

- $x_i \leftarrow \frac{x_{i-2}+x_i+x_{i+1}+x_{i+2}}{5}$
- $x_i \leftarrow ln(_{xi})$

The student's T computation is then done as follows:

- $s_{xy} = \Sigma x_i y_i - N(x_i - x_{avg})(y_i - y_{avg})$
- $s_{xx} = \Sigma x_i^2 - N(x_i - x_{avg})^2$
- $s_{yy} = \Sigma y_i^2 - N(y_i - y_{avg})^2$

- $r = \frac{s_{xy}}{\sqrt{(s_{xx}s_{yy})}}$
- $t = r\sqrt{\frac{N-2}{1-r^2}}$

The t-score is called a "parametric" algorithm, because it takes into account the magnitudes of the samples.

A t-score is considered statistically significant if it's greater than a certain value. Just what this value is varies with the length of the overlap between the samples: a 500 year overlap can have a t-score of 2.6 and pass, but an overlap of only 15 years would have to be higher, like 3.0. In reality most dendrochronologists require t-scores to be much higher than this, and would also insist on overlaps of many decades.

9.1.2 Trend

Trend is another popular crossdate statistic. It computes the percentage of years with the same trend (going-up- or going-down-ness). Scores greater than 60%-70% are good. Trend is also referred to as ufigkeitsko-Gleichläeffizient, Gleichläufigkeit and Eckstein's W.

The trend is the simplest crossdate. For each sample, it computes the trend of each 2-year interval (1001-1002, 1002-1003, and so on). The trend of a 2-year interval is simply whether the next ring is larger, smaller, or the same. The trend score is the percentage of intervals in the overlap which are the same. For example, a 75% trend (a very good score, by the way) means that for 75% of the intervals in the overlap, both samples went up in the same years and down in the same years.

If one sample stays the same, and the other increases or decreases, Corina considers that to be halfway between a same-trend and different-trend, and gives it half a point. Trend is a "non-parametric" algorithm, because it only takes into account if a given ring is bigger or smaller than the previous one, not by how much. To the trend, a drop of "100 1" looks exactly the same as a drop of "100 99". Two completely random samples will have a trend of 50%, on average. So you'd expect a trend must be greater than 50% to be significant.

According to **?**, a trend is significant if:

1. $tr > 50\% + \frac{50}{\sqrt{N}}$ - For example a pair of samples with a 50-year overlap needs a $50 + 50\sqrt{50} = 57.1\%$ trend to be significant, but at a 400-year overlap need only a $50 + 50\sqrt{400} = 52.5\%$ trend. In practice, however, this doesn't tend to work terribly well. Using this scheme, there are typically about three times as many "significant" trend scores as t-scores, and users want this narrowed down a bit more. So take $\sigma = 3$ and use:
2. $tr > 50\% + \frac{50\sigma}{\sqrt{N}}$ - This gives about the same number of significant trend scores as t-scores.

Trends are also used in reconciliation. After they've been reconciled, both readings of a sample should have 100% trend.

9.1.3 Weiserjahre

The Weiserjahre algorithm is used for crossdating summed samples (chronologies) against single samples. All of the algorithms that have been mentioned so far only compare the ring widths. This works fine for raw samples, but when crossdating summed samples, there's a lot more information available, namely, the Weiserjahre data. Wouldn't it make sense to count a [20] 19×1 ring more heavily than a [1] $1 \div 0$ ring? 19 out of 20 samples think it's an increasing year, not just 1.

This is what the Weiserjahre cross does: for each possible overlap, it starts by counting the number of significant intervals of the master for that overlap. A significant interval is one with at least 3 samples, where at least 75% of them have the same trend. Then it computes the percent agreement (like the trend) between the master and the raw sample for only those significant years of the overlap. Of course, for the trend of the master, it doesn't use the trend of the master; it uses the trend of the majority of its elements. They're usually the same, but not necessarily.

Another way to think about the Weiserjahre crossdate is: it's like a trend, but ignoring years where the sum has only 1 or 2 samples, or where there isn't an overwhelming trend in the sum. Also like the trend, the results are given as a percentage.

9.1.4 R-Value

The R-value, or correlation coefficient, is a crossdate which you'll almost never use. It's not terribly useful to dendrochronologists, but statisticians might want to know its value, so Corina makes it available.

The R-value is used in the T-Score, the T-score being defined in terms of the r-value and the overlap, N. If you look at the equations for calculating a T-Score you will see on the penultimate line:

- $r = \frac{s_{xy}}{\sqrt{(s_{xx} s_{yy})}}$

An r-value can range from 0.0 (no correlation) to 1.0 (perfect correlation).

9.1.5 D-Score

The D-score (AXE), is a combination of the T-Score and Trend.

- $D = (tr - 50\%) \times t$

Corina considers 40 to be the threshold for significant AXE.

9.2 Crossdating series

9.3 Managing chronologies

Chapter 10

The Corina server

For basic day-to-day running of the Corina server, you simply need to make sure that the server is running. All other interaction and managment (creating users, granting permissions, accessing data) is done through the Corina desktop application. This section, however, outlines a number of aspects of the server that advanced users may find useful.

10.1 Extending the Virtual Appliance

For those of you that are unfamiliar with Linux, the basic command line prompt is not likely to be very comfortable. If you are interesting in looking at the server in more detail you may therefore prefer to install a full graphical interface. Unlike Windows, there are a number of different graphical interfaces (or desktops) to choose from in Linux, the most popular being Gnome and KDE. To install one of these you need to type one commands listed below. The first line installs Gnome and the second KDE. Windows users that are new to Linux may find KDE more familiar than Gnome.

```
sudo apt-get install ubuntu-desktop
```

```
sudo apt-get install kubuntu-desktop
```

10.2 Security

The basic installation of the Corina server includes the standard configuration for Apache, PHP and PostgreSQL. Although these products are considered secure by default, there are a number of measures that can be taken to make them more so. If your server is only accessible within your local intranet (e.g. behind a robust firewall) then you may not feel it necessary to modify the standard setup. Precautions may be deemed more important if you server is accessible from the internet. In this case it would be wise to contact your local network administrator for further information.

10.2.1 Usernames and passwords

There are a number of default usernames and passwords setup on your server. If your server is accessible for the internet we strongly advise you to change these defaults and anyone with knowledge of the Corina server could access and compromise your machine.

System user - these are the credentials you use to log in to the command prompt in your Corina Virtual Appliance. By default the user is 'corina' and the password is 'w3l0v3tr33s'. To change this log in to the command prompt and type `passwd` and follow the instructions.

Database user - these are the credentials used by the webservice to read and write to the database.

Corina admin user - these are the admin credentials that you use to log in with in your Corina desktop application. Be default the user is '???' and the password is '???'. To change these open the Corina desktop application, then go to Admin then Change password.

10.2.2 Authentication and encryption

Corina uses a relatively sophisticated method to ensure that unauthorised users cannot access the Corina database through the webservice. It is loosely based around http digest authentication and uses a challenge and response scheme. This makes use of cryptographic hashes (a relatively short digital fingerprint of some data but which cannot be decompiled to retrieve the original data) and nonces (a pseudo-random string used just once). All hashes used in the Corina webservice use the MD5 algorithm. This decision will be periodically reviewed to ensure that MD5 is the most appropriate and secure algorithm to use. Whilst an MD5 hash of a short phrase can be compromised, the length and randomness of the original data means with current cracking techniques this is essentially impossible. For a complete description of Corina's authentication procedure see section 13.1.

The default Corina server setup, however, uses standard HTTP protocol to communicate between the server and the desktop application. This is the same protocol used for the majority of web pages on the internet and a determined hacker could eavesdrop on this communication. Depending on how important and private you perceive your data you may choose to use Secure Socket Layer (SSL) to encrypt this communication. This is the same technology used by websites such as online banking. To make full use of this upgrade in security you will however also require a SSL certificate from an official licensing authority. These certificates typically cost several hundred dollars per year.

10.3 Directly accessing the database

Although the Corina database is designed to only be accessed by the Corina desktop application via the Corina server's webservice, you may decide that you'd like to directly access the database yourself. For instance, you may like to write complicated SQL queries to probe your database in ways not currently supported by the Corina desktop client.

> ⚠ *Any changes made to the database may have drastic consequences. We strongly recommend that you never write changes directly to the database as this can cause loss of data and corrupt future upgrades to Corina.*

10.3.1 PGAdminIII

One of the easiest ways to access the PostgreSQL database is through the application PGAdmin3. This is a cross-platform open source application for communicating with PostgreSQL databases. You can install PGAdmin3 on your desktop computer and access the remotely running database using your database user credentials. By default, PostgreSQL runs on port 5432.

10.3.2 ODBC

It is also possible to connect to your Corina database via an ODBC connection. This allows limited access to the database from a variety of database applications including programs like Microsoft Access for which further details are given here. To use ODBC you will need to install the PostgreSQL ODBC driver (*http://www.postgresql.org/ftp/odbc/*) on your desktop computer.

Once you've installed the driver you can then open a blank database in Access and go to Files, Get external data then Link tables. In the file dialog box change the file type to ODBC Databases(). Next, select the PostgreSQL Unicode driver, then fill out the server details. You should then be able to open the tables and views from the Corina server database directly from within Access as if they were local tables. Be warned though that Access and ODBC have many limitations compared to PostgreSQL, especially with regards data types. For this reason we *strongly* recommend using this for read only purposes. Using the ODBC connection to write changes to your PostgreSQL database is quite likely to cause serious issues.

10.3.3 PSQL

The final, and most advanced method is to use the psql client on your server. This is a command line client which can be used to interrogate the database. If you're not already familiar with psql it is unlikely that this is a good method for you to use!

10.4 Managing map services

Chapter 11

Help

Part II

Developers guide

Chapter 12

Development environment

Corina is open source software and we actively encourage collaboration and assistance from others in the community. There is always lots to do, even for people with little or no programming experience. Please get in touch with the development team as we'd love to hear from you.

12.1 Developing Corina Desktop

The IDE of choice of the main Corina developers is Eclipse (*http://www.eclipse.org*. There are many other IDEs around and there is no reason you can't use them instead. Either way, the following instructions will hopefully be of use.

We have successfully developed Corina on Mac, Windows and Linux computers over the years. The methods for setting up are almost identical.

The first step is to install eclipse, sun-java6-jdk and subversion. These are all readily available from their respective websites. On Ubuntu they can be install from the command line easily as follows:

```
sudo apt-get install eclipse subversion sun-java6-jdk
```

Once installed, you can then launch Eclipse. To access the Corina source code you will need to install the Subversive plugin to Eclipse. As of Eclipse v3.5 this can be done by going to *Help* ➔ *Install new software*. Select the main Update site in the 'Work with' box, then locate the 'Subversive SVN Team Provider' plugin under 'Collaboration'. If you are using an earlier version of Eclipse you may need to add a specific Subversive update site. See the Subversive website (*http://www.eclipse.org/subversive/*) for more details. Once installed you will need to restart Eclipse.

Next you need to get the Corina source code. Go to File *New* ➔ *Project*, then in the dialog select *SVN* ➔ *Project from SVN*. There are two methods of accessing the Corina repository: anonymously, in which case you will have read only access; or with a username provided by the Corina development team. Anonymous users will need to add a repository in the form: *http://dendro.cornell.edu/svn/corina/* and full users will need to use *svn+ssh://dendro.cornell.edu/home/svn/corina/*.

Once the project has downloaded to your workspace, you may need to set the compliance level. This can be done by going to *Project* ➔ *Properties* ➔ *Java compiler* and choosing compliance level of 6.0. Corina uses a handful of Java 6 specific functions, particularly with regards JAXB, so will not run successfully with Java 5.

To launch Corina, you will need to *Run* ➔ *Run Java application*. Create a new run configuration with the main class 'set to edu.cornell.dendro.corina.gui.Startup'.

12.2 Developing the webservice

The Corina webservice is written entirely in PHP. Eclipse is used for this development too so most of the setup steps are identical to setting up Eclipse for developing the Corina desktop client. You will, however, probably want to install the PHP development plugin so that you get syntax highlighting etc. See the Eclipse PDT website (*http://www.eclipse.org/pdt/*) for further information.

12.3 Dependencies

Apache commons lang Apache 2.0 TridasJLib Apache 2.0 Batik Apache 2.0 RXTXcomm LGPL JDOM Apache 2.0 Swing layout LGPL Log4J Apache 2.0 JNA LGPL Apache mime 4J Apache 2.0 Commons codec Apache 2.0 http Client LGPL Http core Apache 2.0 Http mime Apache 2.0 Jsyntaxpane Apache 2.0 L2prod-common-shared Apache 2.0 L2prod-common-sheet Apache 2.0 l2fprod common buttonbar Apache 2.0 itext GAPL PDFRenderer LGPL DendroFileio Apache 2.0 java simple mvc MIT JgoogleAnalyticsTracker MIT gluegen BSD jogl BSD+ nuclear clause worldwind NASA slf4j MIT jfontchooser LGPL miglayout BSD pljava BSD postgresql PostgreSQL License (BSD/MIT) forms BSD simplelog GPLv3 jxl LGPL wizard GPLv2 netbeans swing outline GPLv2

Chapter 13

Systems architecture

The centralised nature of the Cornell Tree-Ring Lab data required a server-client architecture of some type. In the original Corina v1 this was achieved simply by having users save their data in a network folder saved on a central server. Whilst this method was adequate, it has many data storage issues that can be largely solved by moving the data storage infrastructure to a relational database management system.

Although it would be possible (and arguably simpler) to refactor Corina to talk directly to one central database server it was decided to go a step further and implement a Web Services orientated server-client architecture for Corina v2.

A web services approach decouples the desktop client from the server so that the server can work on its

13.1 Authentication design

The authentication mechanism is loosely based around http digest authentication and uses a challenge and response scheme. This makes use of cryptographic hashes (a relatively short digital fingerprint of some data but which cannot be decompiled to retrieve the original data) and nonces (a pseudo-random string used just once). All hashes used in the Corina webservice use the MD5 algorithm. This decision will be periodically reviewed to ensure that MD5 is the most appropriate and secure algorithm to use. Whilst an MD5 hash of a short phrase can be compromised, the length and randomness of the original data means with current cracking techniques this is essentially impossible.

The first time a client attempts to retrieve data from the webservice (or when the client's credentials are incorrect or have expired) the following events occur:

- Server returns an message requesting authentication. This message includes a nonce (a hash of the current date and time to the nearest minute) which we will call 'server nonce'.
- The client creates a second nonce (client nonce) which is a random hash of it's choosing, and a response which is a hash of "username:hashofpassword:servernonce:clientnonc". It sends this response, along with the username and client nonce back to the server but does not send the original server nonce.
- The server computes the same "username:hashofpassword:servernonce:clientnonce" hash using the information it has stored in the database. As the server nonce is constant for a minute the two response should match. If not the server recomputes the server nonce for one minute ago and tries again. This ensures that the server nonce sent to the client is valid for between 1 and 2 minutes.
- Once the server authenticates the user a session cookie is sent to the client. On subsequent requests the server recognises the session id and doesn't request authentication again.

As the user's password is hashed at all points, even if the communication is hijacked the attacker will not be able to derive the users password. The user's password is also stored in hash form within the database. This also means that system administrators do not have access to the passwords either.

The use of the server nonce within the response means that it will only be valid for a maximum of two minutes. This removes the possibility of a replay attack.

13.2 Database permissions design

The database has a user and group based security scheme at three object levels: site, tree and measurement. A user can be a member of one or more groups, and groups can be members of zero or more other groups. The current implementation allows for one nested level of groups within groups however this could be extended if required. Security is set on a group-by-group basis rather than on a single user to ensure ease of management.

There are five types of permissions granted: create, read, update, delete and no permission. Each permission is independent of each other with the exception of 'no permission' which overrides all other permissions.

A group can be assigned one or more of the permissions types to any of the sites, trees or measurements in the database. Intermediate objects such as subsites, specimens and radii inherit permissions from their parent object. For instance if a group has permission to read a site then it will have permission to read all subsites from that site.

It is envisaged that most of the time, permissions will be set on a site-by-site basis. It will not be necessary to explicitly assign permissions to trees and measurements as all permissions will be inherited. So assuming that no permissions are set on a tree for a particular group, the permissions for the tree will be derived from the site from which the tree was found. If, however, permissions are assigned to the tree, then these will override those of the site. In this way it will be possible to allow a group to read the data from one particular tree from a site in which there otherwise do not have permission to access.

Privileges are cumulative. This means that if a user is a member of multiple groups then they will gain all the privileges assigned to those groups. If one of the groups that the user is a member of has 'no privileges' set on an object it will however override all other privileges. Therefore if a user is a member of groups A and B, and group A has read privilege and group B has 'no privilege' then the user will not be able to access the record.

A special 'admin group' has been created into which only the most trusted users are placed. Members of the admin group automatically gain full privileges on all data within the database. They also have permission to perform a number of administrative tasks that standard users are insulated from.

13.3 Universally Unique Identifiers

All entities in the Corina database have a primary key based on the Universally Unique Identifier (UUID) concept. This is a randomly created 128-bit number which due to the astronomically large number of possibilities (3×1038) means that it is guaranteed to be unique across all installations of Corina. This code is typically represented by 32 hexadecimal digits and 4 hyphens like this: 550e8400-e29b-41d4-a716-446655440000.

13.4 Barcode specifications

Barcodes in Corina are based on the UUID primary keys of database entities. Because they are used for different entities in Corina (boxes, samples and series) it was also necessary to incorporate a method for determining what type of entity a barcode represents. This is done by appending a single character and a colon to the beginning of the UUID: 'B:' for box; 'S:' for sample; 'Z:' for series.

The barcodes in Corina use the Code 128 scheme. This symbology was chosen as it allows the encoding of alphanumeric characters in a high-density label and can be read by all popular barcode scanners. While it would have been possible to create a barcode of plain UUIDs, the 36 (or even 32) characters would result in a barcode wider than many scanners could read. Most scanners on the market have a maximum scan width of at least 80mm, so this was used as the baseline to work to.

To make the barcodes less than 80mm, the UUID (with prepended entity type character code) are Base64 encoded. For example the series with UUID 3a8f4336-d17d-11df-abde-c75e325aebae would be encoded from Z:3a8f4336-d17d-11df-abde-c75e325aebae to become: Wjo6j0M20X0R36vex14yWuuu

Part III

Appendices

Appendix A

Belfast Apple

Format name	Belfast Apple
Other name(s)	None known
Type	Text file
Extension(s)	Various (typically txt and dat)
Read/write support	Read and write
Reference implementation	No original software is known to exist so TRiCYCLE is proposed as the reference implementation
Data / metadata	Data only with comment
Calendar type	n/a
Absolute dating support	No
Undated series support	Yes
Relative dating support	No
Multi series support	No
Original designer	John Pilcher

A.1 Description

Belfast Apple is a simple text file format (see also Belfast Archive) originating from the Queens University Belfast lab and originally designed for use on an Apple II computer. This format is not known to be actively used but a large amount of data (especially at Belfast) is archived in this format.

- Line 1 - name of the site or object the data refers to.
- Line 2 - identifier for the sample the data refers to.
- Line 3 - number of data values in the file
- Lines 4+ - line feed delimited data values as integers in 1/100th mm
- Final line contains a comment typically starting with 'COMMENT -'

A.2 Example file

```
EXAMPLE SITE
A1805
106
188
165
184
112
103
111
239
226
132
143
146
140
100
176
139
124
115
78
80
156
75
110
80
130
83
157
99
115
102
110
108
87
135
107
96
70
128
119
86
101
106
129
88
101
151
106
97
110
97
91
93
100
124
99
134
125
105
96
107
142
100
COMMENT -     PB 15-NOV-99
```

Appendix B

Belfast Archive

Format name	Belfast Archive
Other name(s)	None known
Type	Text file
Extension(s)	Various (typically arx, txt and dat)
Read/write support	Read only
Reference implementation	No original software is known to exist so TRiCYCLE is proposed as the reference implementation
Data / metadata	Data with limited metadata
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	No
Relative dating support	No
Multi series support	Yes
Original designer	Martin Munro

B.1 Description

Belfast Archive is a simple text file format based on the original Belfast Apple format at the Queens University Belfast lab. It shares the same features as Belfast Apple but with the addition of a number of metadata fields at the end of the file.

- Line 1 - name of the site or object the data refers to.
- Line 2 - identifier for the sample the data refers to.
- Line 3 - number of data values in the file
- Lines 4+ - line feed delimited data values as integers in 1/100th mm
- The lines `"[[ARCHIVE]]"` and `"[[ END OF TEXT ]]"` denote the start and finish of the metadata section

The metadata section contains the following lines:

- Line 1 - start year as an integer.
- Line 2 - unknown
- Line 3 - Double representing the resolution of data values e.g. .1= 1/10ths mm, .01 = 1/100th mm, .001 = microns etc
- Line 4 - unknown
- Line 5 - unknown
- Line 6 - unknown
- Line 7 - title of the data series
- Line 8 - unknown

- Line 9 - unknown

B.2 Example file

```
1 EXAMPLE SITE
2 1
3 176
4 342
5 338
6 334
7 409
8 362
9 308
10 360
11 264
12 325
13 318
14 51
15 48
16 47
17 60
18 49
19 48
20 "[[ARCHIVE]]"
21 1277
22 9177
23 .01
24 1.035795
25 0.212144
26 BOB 25/03/95
27 EXAMPLE SITE #01
28 Pith F Sap 32
29 ""
30 "[[ END OF TEXT ]]"
```

Appendix C

Besançon

Format name	Besançon
Other name(s)	SYLPHE
Type	Text file
Extension(s)	txt
Read/write support	Read and write
Reference implementation	
Data / metadata	Data and some structured metadata
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	Yes
Relative dating support	No
Multi series support	Yes
Original designer	Georges Lambert

C.1 Description

The Besançon format is most commonly used in a number of French laboratories. The format allows for multiple series in the same file. Each series (or element block in Lambert's notation) is made up of a header line, optional metadata and a data block each of which are delimited by a line feed.

The header line begins with a dot character, then one or more spaces, then an element name (without spaces) followed by a space and any number of ignored characters.

The metadata fields are space or line feed delimited. Each field is recorded using a key of three letters. The format allows for the full spelling out of the field if preferred, but it is the first three letters that are read by software so LON is the same as LONGEUR. Some fields are 'unimodal' in that their presence is all that is required e.g. CAM means that cambium was observed. Other fields are 'bimodal' which means they require a value to be associated with them. In this case the field key is followed by a space and then an integer or string value e.g. POS 1950. The accepted metadata fields are as follows:

LON Number of data values

POS The temporary first ring date given relatively to a group

ORI The year for the first ring

TER The year for the last ring. Should be the same as ORI + LON

MOE Pith present

CAM Cambium present

AUB Number of the first sapwood ring

All other information in the metadata block should be ignored. This feature is often used to allow the inclusion of multi-line comments.

The data block begins with the marker line VAL (like metadata keys, subsequent characters are ignored so sometimes the rest of this line is used for comments). Subsequent lines contain integer values delimited by a space or line feed. Missing rings are marked with a comma character and the end of the data is marked with a semicolon.

C.2 Additional information

- There is nothing in the specification to say what precision the data values should be in. Following conversations with users it appears that Besançon files are mostly 1/100th mm but this is not always the case. Some files include a Précision field, but this is not documented or standardised.
- There are a number of additional fields that are commonly used but which do not appear in the format specification. These are also supported by the DendroFileIOLib
  ESP Species
  ECO Bark present

C.3 Example file

```
 . abc22/43
 Lon 129
 Esp quercus sp  Nat lambris
 Precision 1/100
 Moelle non presente
 Aub 0
valeurs
  149  119  156  146  170  187  197  146  191  177
  137  108  160  108  120  177  136  174  190  109
  189  176  170  162  114  126  133  152  146  127
  119  131  146  133  147   82   57   77   77   82
   96   49   97   76   88   82   72   83   81   90
   85   87   78  104  111  132  141  105  104  120
  111  121  115   89   94   88   90  115  111  106
  107  120   80   92   98   84   97   82  100   86
   99   65   85  113   90   82   57   57   99   94
   95  105  120  110   93   96  131  133  123  122
  113  119   95  127   88  104    ,    ,    ,    ,
    ,    ,    ,    ,    ,    ,    ,    ,    ;
```

Appendix D

CATRAS

Format name	CATRAS
Other name(s)	None known
Type	Binary
Extension(s)	cat
Read/write support	Read only
Reference implementation	CATRAS
Data / metadata	Data and some structured metadata
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	Yes
Relative dating support	No
Multi series support	No
Original designer	Roland Aniol

D.1 Background

The CATRAS format (**?**) is the only known binary dendro data format. As such it can't be read by a simple text editor, and can't be imported by spreadsheet or database programs. The format was designed by Roland Aniol for use in his program of the same name. The binary nature of the format means the files are typically much smaller than text files containing similar data. The closed nature of the format originally meant that users were tied to the application. The fact that users can't manually edit the file means that the validity of files is not a problem like it is with most other dendro formats.

CATRAS is a closed format with no documentation. The format was originally decoded in the early 1990's and permission was granted by Aniol for a converter to be included in Henri Grissino-Mayer's CONVERT5 application on the condition that the format remained closed source. Subsequently others have independently released application and code that can read ring-width data from CATRAS files to a greater or lesser extent.

D.2 Reading byte code

Reading byte code is more complicated than reading text files. Each byte is 8-bits and therefore can represent up to 256 values. Depending on the type of information each byte contains, the bytes are interpreted in one of four ways:

D.2.1 Strings

Some of the bytes in CATRAS files contain character information. In this case each byte represents a letter. In java an array of bytes can be directly decoded into a string.

D.2.2 Integers

As a byte can only represent 256 values, whenever an integer is required, CATRAS stores them as byte pairs. Each byte pair consists of a least significant byte (LSB) and a most significant byte (MSB). The order that they appear in files typically varies between platforms and is known as 'endianness'. As CATRAS solely runs of Microsoft (x86) processors we can safely assume that all CATRAS files will be using little-endian (i.e. LSB MSB). The counting in a byte pair therefore works as follows:

Value	LSB	MSB
0	0	0
1	1	0
...	...	...
255	255	0
256	0	1
257	1	1
258	2	1
...	...	...

A byte pair can therefore store 256x256=65536 values (more than enough for most number fields). Matters are complicated though by the need to store negative numbers. In CATRAS pairs with an MSB¡=128 are positive, while pairs with an MSB ranging from 255 to 128 (counting backwards) represent negative values:

Value	LSB	MSB
-1	255	255
-2	254	255
-3	253	255
-4	252	255
...	...	...

D.2.3 Categories

Categories are typically recorded as single bytes as most categories have just a few possible values. They can therefore be conceptualised as being integers where 0=first option, 1=second option etc. The exception to this is for species because there are more than 256 species. In this case, a byte pair is used in exactly the same way as described for integers above. The only problem for species is that the codes are unique to each laboratory and refer to values enumerated in a separate '.wnm' file. Without this dictionary the species code is of little use.

D.2.4 Dates

Dates are stored as three single bytes, one for day, one for month, one for year. With only 256 values available for 'year', all dates are stored with 2 digit years e.g. 25/12/84. When reading CATRAS files all years ¿70 are therefore treated as 20th century, whereas years ¡70 are treated as 21st century. This is an arbitrary decision for use in this library as CATRAS does not care either way.

D.3 Metadata

The first 128 bytes contain the file header information and the remainder of the file contains the ring-width data and sample depth data. Our current understanding of the header bytes is as follows but I'm not convinced that these are all correct. Deciphering these requires painstaking work because we must try to ascertain how each byte is being used (e.g. as a byte pair, single byte or as a string):

- 1-32 - Series name
- 33-40 - Series code
- 41-44 - File extension
- 45-46 - Series length
- 47-48 - Sapwood length
- 49-50 - Start year
- 51-52 - End year
- 53 - 1=pith 2=waldkante 3=pith to waldkante
- 54 - 1 = ew only last ring
- 55-56 - Start year
- 59-60 species also needs a catras.wnm file
- 61-63 - Creation date
- 64-66 - Amended date
- 67 - Sapwood
- 68 - 1=valid stats
- 69-75 - dated?
- 84 - 0=raw 1=treecurve 2=chronology
- 85-86 - User id
- 89-92 - Average width
- 93-95 - Standard deviation
- 96-100 - Autocorrelation
- 101-104 - Sensitivity

D.4 Data

The remaining bytes in the file contain the actual data values stored as integer byte pairs. It appears that older version of CATRAS included one or more padding values of -1. These values should be ignored. The end of the data values are indicated by a stop value of 999.

Following the ring-width data values there are 42 bytes of unknown meaning. These are then followed by byte pairs representing the counts/sample depth for each ring if the series is a chronology.

D.5 Unknown bytes

There are a number of bytes in both the header and data sections that are are unaccounted for and are therefore likely to contain data that we are ignoring. For this reason although we could attempt to create CATRAS files from what we know we can't be sure they would be valid:

- Header
  - 57-58
  - 69-82
  - 105-128
- Data
  - 0-42 following end of data marker

Appendix E

Comma Separated Values

Format name	Comma Separated Values
Other name(s)	CSV
Type	Text file
Extension(s)	Various (typically txt or csv)
Read/write support	Read and write
Reference implementation	n/a
Data / metadata	Data only
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	No
Relative dating support	No
Multi series support	No
Original designer	n/a

E.1 Description

Comma separated values format is a simple text format for representing tabular data. It is not specific to dendrochronology data and is supported by most spreadsheet and database applications. Data is delimited into columns using a comma character to indicate cell boundaries.

Support for CSV files in TRiCYCLE is limited to a particular layout of data. The expected layout is the same as for Excel and ODF spreadsheet files:

- Row 1 - Header names for each column
- Column A - Year values
- Column B+ - One column for each series containing values in millimetres. Cells are left empty if no data is available for a series because it does not extend to a particular year. Data must be continuous for each series, so missing/unmeasured rings should be included as zero.

E.2 Example file

```
Year , MySample1 , MySample2
500 ,0.33 ,
501 ,0.26 ,0.26
502 ,0.2 ,0.2
503 ,0.14 ,0.14
504 ,0.08 ,0.08
505 ,0.02 ,0.02
506 ,0.2 ,0.2
507 ,0.14 ,0.14
508 ,0.08 ,0.08
509 ,0.2 ,
510 ,0.33 ,
511 ,0.08 ,
512 ,0.33 ,
513 ,0.22 ,
```

Appendix F

Corina Legacy

Format name	Corina Legacy
Other name(s)	Corina
Type	Text file
Extension(s)	Various including raw, rec, ind, cln, sum)
Read/write support	Read and write
Reference implementation	Corina
Data / metadata	Data and some structured metadata
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	No
Relative dating support	Yes
Multi series support	No
Original designer	Robert 'Mecki' Pohl

F.1 Description

The Corina Legacy format is the file format used by the Corina software prior to version 2, when it transferred to using TRiDaS. The format was originally designed for use with the MS-DOS version of Corina but was also used as the native file format in the later Java versions (up to and including v1.1).

A Corina file contains yearly data (ring-width and number of samples for that year), some fixed metadata, and optionally weiserjahre data and a listing of element samples (for summed samples).

The title comes first, on a line by itself, followed by a blank line. The title is repeated later, so this is only to make it easier for people or external programs to read the title.

The *metadata section* comes next. The syntax is `;TAG` value. Tags are all uppercase. Their order is fixed. Some values are terminated by a newline, others by the next semicolon. Valid tags, and their internal names are:

- ID - 8 character ID used when exporting to Tucson format
- NAME - Name of the series
- DATING - Either R (relative) or A (absolute)
- UNMEAS_PRE - Number of unmeasured rings towards the pith
- UNMEAS_POST - Number of unmeasured rings towards the bark
- FILENAME
- COMMENTS, COMMENTS2 etc - Free text comments
- TYPE - either C (core), H (charcoal) or S (section)
- SPECIES

- SAPWOOD - Count of sapwood rings
- PITH - either P (present), * (present but undateable), or N (absent)
- TERMINAL - either B (bark), W (waney edge), v (near edge), vv (unknown)
- CONTINUOUS - referring to the outer ring, either C (continuous), R (partially continuous) or N (not continuous)
- QUALITY - either + (one unmeasured ring), ++ (more than one unmeasured ring)
- FORMAT - either R (raw) or I (indexed)
- INDEX_TYPE - type of index used
- RECONCILED - Y or N indicating whether the series has been reconciled against another series

The *data section* comes next and this always starts with the line ;DATA and for reasons lost in time there are nine spaces afterwards.

Data lines come in pairs, the first line containing the year and data values, the second containing the sample depth/count for each value. For reasons unknown, the first and last data line pair have a slightly different syntax to the others.

- First data line begins with a space and an integer for the first year in the line. There then follows 9 spaces followed by the integer data value for the first ring. The remaining data values (often less than a full decades worth) on that line follow as integers left padded by spaces to take up 6 characters.
- The sample depth line that pairs with this follows next starting with 16 spaces, followed by the sample depth value enclosed in square brackets. The remaining sample depth values follow in square brackets left padding with spaces to take up 6 characters.
- Next comes the first normal data line. This begins with a space, followed by an integer year value. The data values follow as integers left padded by spaces to take up 6 characters. A data line has a decades worth of data values.
- Next comes the normal sample depth line. It begins with 7 spaces followed by each of the sample depth values enclosed in square brackets and left padded with spaces up to 6 characters.
- Data lines continue in pairs until the last line is reached. This is the same as a normal data line except it includes an extra data value 9990 as a stop marker. This data line may have less than a full decade of values.
- The final sample depth line is the same as normal except it is shifted left by 4 characters. A sample depth value is also included for the dummy 9990 stop marker year.

Following the data block there is a blank line and two option blocks of data that are only included if the file is a chronology file.

The next block of information in a chronology file is denoted by a line ;ELEMENTS. The following lines contain the file names of the data files that have contributed to the creation of the chronology.

Following this is an optional block denoted by the line ;weiserjahre followed by the weiserjahre data. Each weiserjahre data line begins with a space followed by a integer year value for the first year in the line. The weiserjahre value is left padded with spaces to fill 6 characters and the value itself is written as X/Y where X is the number of samples that show an upward trend in width; and Y is the number of samples that show a downward trend in width. The weiserjahre value is forward facing so the value for ring 1001 shows the trend between ring 1001 and 1002. There is therefore one less weiserjahre value in the final row than there are ring-widths.

The final line of Corina data files contains the author's name preceded by a tilde.

F.2 Example file

```
Trebenna, Byzantine Fortress, NW tower 1AB

;ID 907010;NAME Trebenna, Byzantine Fortress, NW tower 1AB;DATING R;UNMEAS_PRE 1;
    UNMEAS_POST 1
;FILENAME G:\DATA\TRB\TRB1AB.SUM


;TYPE S;SPECIES Juniperus sp.;FORMAT R;PITH +
;TERMINAL vv;CONTINUOUS N;QUALITY +
;RECONCILED Y
;DATA
 1001        125   219   207   139    62   107    29    91    65
               [1]   [1]   [1]   [1]   [1]   [1]   [1]   [1]   [1]
 1010     71   132    74   150    75   156   122    81    46    57
         [1]   [1]   [1]   [1]   [1]   [1]   [1]   [1]   [1]   [1]
 1020    147    78    89   126    73   121    67    71    64   129
         [1]   [1]   [1]   [1]   [1]   [1]   [1]   [1]   [1]   [1]
 1030    149   155   122   126    53   136    90    65   100    67
         [1]   [1]   [1]   [1]   [1]   [1]   [1]   [1]   [1]   [2]
 1040     67   101   132   102    40    67    42    36    62    29
         [2]   [2]   [2]   [2]   [2]   [2]   [2]   [2]   [2]   [2]
 1050     30    44    46    40    34    61    55    29    44    63
         [2]   [2]   [2]   [2]   [2]   [2]   [2]   [2]   [2]   [2]
 1060     62    38    22    26    26    28    37    21    21    27
         [2]   [2]   [2]   [2]   [2]   [2]   [2]   [2]   [2]   [2]
 1070     17    18    50    21    33    12    17    16    27    20
         [2]   [2]   [2]   [2]   [2]   [2]   [2]   [2]   [1]   [1]
 1080     18    11     9     8  9990
     [1]   [1]   [1]   [1]   [1]

;ELEMENTS
G:\DATA\TRB\TRB1A.REC
G:\DATA\TRB\TRB1B.REC
;weiserjahre
 1001  1/0      0/1      0/1      0/1      1/0      0/1      1/0      0/1      1/0
 1010  1/0      0/1      1/0      0/1      1/0      0/1      0/1      0/1      1/0
    1/0
 1020  0/1      1/0      1/0      0/1      1/0      0/1      1/0      0/1      1/0
    1/0
 1030  1/0      0/1      1/0      0/1      1/0      0/1      0/1      1/0      0/1
    1/1
 1040  2/0      2/0      0/2      0/2      2/0      0/2      0/2      2/0      0/2
    2/0
 1050  2/0      1/1      0/2      0/2      2/0      0/2      0/2      2/0      2/0
    1/1
 1060  0/2      0/2      2/0      1/1      2/0      2/0      0/2      1/1      2/0
    0/2
 1070  1/1      2/0      0/2      2/0      0/2      2/0      1/1      1/0      0/1
    0/1
 1080  0/1      0/1      0/1
~ Unknown User
```

Appendix G

DendroDB

Format name	DendroDB
Other name(s)	
Type	Text file
Extension(s)	dat
Read/write support	Read only
Reference implementation	DendroDB website
Data / metadata	Data and some structured metadata
Calendar type	Astronomical
Absolute dating support	Yes
Undated series support	No
Relative dating support	No
Multi series support	Yes
Original designer	Simon Brewer

G.1 Description

The DendroDB format is an export file format produced by the DendroDB website/database. There is no known software that can natively read DendroDB files so a 'writer' for this format has not been developed.

The format is self-explanatory, beginning with a copyright line, followed by 7 metadata lines, then the data itself. There are eight possible data variables: Total width; Earlywood width; Latewood width; Min. Density; Max. Density; Earlywood density; Latewood density; Average density. Ring width data is provided in microns but the units for density measurements are not document.

As of Feb 2011, the DendroDB database does not contain data prior to 441AD so handling of BC/AD transition has not been tested. The DendroDB web interface suggests that BC dates should be entered as negative integers, but it also allows request for data from year 0. This suggests the database uses an Astronomical calendar and this is how the DendroIOLib treats it.

G.2 Example file

```
Data downloaded from DendroDB. Please acknowledge authors
Site: Example site
Contact: A N Other
Species: Larix sibirica
Parameter: Latewood width
Latitude: 53.25
Longitude: 57.35
Elevation: 1670
Tree Core Year Latewood width
1 1 1648 16
1 1 1649 21
1 1 1650 8
1 1 1651 10
1 1 1652 6
1 1 1653 8
1 1 1654 11
1 1 1655 13
1 1 1656 9
1 1 1657 10
1 1 1658 10
1 1 1659 4
1 1 1660 5
1 1 1661 7
1 1 1662 4
1 1 1663 8
...
```

Appendix H

Heidelberg

Format name	Heidelberg
Other name(s)	TSAP, FH
Type	Text file
Extension(s)	.fh
Read/write support	Read and write
Reference implementation	TSAP-Win
Data / metadata	Data and extensible metadata
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	Yes
Relative dating support	Yes
Multi series support	Yes
Original designer	Frank Rinn

H.1 Description

The Heidelberg format (**?**) is the native file format for Rinntech's TSAP-Win software. It supports metadata in the form of keyword-value pairs. There are more than 140 standard keywords specified in the documentation, but users can extend these with their own. This makes the format extremely flexible, but the absence of any checking of data types (strings, numbers categories etc) and no method of validation means that there can be problems interpreting metadata entries.

Heidelberg files can store one or more series in a single file. Each series is represented by a header and a data block.

The header block begins with a line HEADER:. This is followed by lines of metadata, with one field on each line, in the format keywords=value much like a standard Windows INI file. As mentioned previously there are a number of predefined keywords, all of which are outlined here:

- AcceptDate
- Age
- AutoCorrelation
- Bark
- BHD
- Bibliography
- Bibliography[n]
- BibliographyCount
- Bundle
- CardinalPoint
- ChronologyType
- ChronoMemberCount
- ChronoMemberKeycodes
- Circumference
- Client
- ClientNo
- Collector
- Comment

- Comment[n]
- CommentCount
- Continent
- CoreNo
- Country
- CreationDate
- DataFormat
- DataType
- DateBegin
- Dated
- DateEnd
- DateEndRel
- DateOfSampling
- DateRelBegin[n]
- DateRelEnd[n]
- DateRelReferenceKey[n]
- DateRelCount
- DeltaMissingRingsAfter
- DeltaMissingRingsBefore
- DeltaRingsFromSeedToPith
- Disk
- District
- EdgeInformation
- EffectiveAutoCorrelation
- EffectiveMean
- EffectiveMeanSensitivity
- EffectiveNORFAC
- Key
- EffectiveNORFM
- EffectiveStandardDeviation
- Eigenvalue
- Elevation
- EstimatedTimePeriod
- Exposition
- FieldNo
- FilmNo
- FirstMeasurementDate
- FirstMeasurementPersID
- FromSeedToDateBegin
- GlobalMathComment[n]
- GlobalMathCommentCount
- GraphParam
- Group
- HouseName
- HouseNo
- ImageCellRow
- ImageComment[n]
- ImageFile[n]
- ImageCount
- ImageFile
- Interpretation
- InvalidRingsAfter
- InvalidRingsBefore
- JuvenileWood
- KeyCode
- KeyNo
- LabotaryCode
- LastRevisionDate
- LastRevisionPersID
- Latitude
- LeaveLoss
- Length
- Location
- LocationCharacteristics
- Longitude
- MajorDimension
- MathComment
- MathComment[n]
- MathCommentCount
- MeanSensitivity
- MinorDimension
- MissingRingsAfter
- MissingRingsBefore
- NumberOfSamplesInChrono
- NumberOfTreesInChrono
- PersId
- Pith
- Project
- ProtectionCode
- Province
- QualityCode
- Radius
- RadiusNo
- RelGroundWaterLevel
- RingsFromSeedToPith
- SampleType
- SamplingHeight
- SamplingPoint
- SapWoodRings
- Sequence
- SeriesEnd
- SeriesStart
- SeriesType
- ShapeOfSample
- Site
- SiteCode
- SocialStand
- SoilType
- Species
- SpeciesName
- StandardDeviation
- State
- StemDiskNo
- Street
- Timber
- TimberHeight
- TimberType
- TimberWidth
- TotalAutoCorrelation
- TotalMean
- TotalMeanSensitivity
- TotalNORFAC

- TotalNORFM
- TotalStandardDeviation
- Town
- TownZipCode
- Tree
- TreeHeight
- TreeNo
- Unit
- UnmeasuredInnerRings
- UnmeasuredOuterRings
- WaldKante
- WoodMaterialType
- WorkTraces

The meaning of many of these keywords is fairly self-explanatory but others are a little more obscure. As there is no data typing or validation the format of the contents of these fields cannot be predicted. This is particularly a problem when trying to compare fields such as Latitude, Longitude and FirstMeasurementDate, but is especially a problem when comparing files produced in different labs.

The header section is followed by a data section denoted by a line containing the keyword DATA: followed by the type of data present which can be one of Tree; HalfChrono; Chrono; Single; Double; Quad. Tree, HalfChrono and Chrono are the original keywords supported by early versions of TSAP but these are now deprecated in preferences of the more generic Single, Double and Quad terms. The terms Single, Double and Quad are largely interchangeable with Tree, HalfChrono and Chrono respectively, but not completely. Double can refer to both Tree and HalfChrono format data. When the newer terms are used, the header keyword DataFormat is used to record whether the data is equivalent to Tree, HalfChrono or Chrono.

Single format - data is typically used for storing raw measurement series. Each data line contains 10 data values each being a left space padded integer taking up 6 characters. Any spare data values in the final data line are filled with zeros. Alternatively it appears that TSAP-Win also accepts this data section as single integer values one per line.

Double format - data is for storing data with sample depth information - typically chronologies. Like the single format section, data is stored as 10 integer values, each taking up 6 characters and left padded with spaces. The values are in pairs of ring-widths and sample depths, therefore five rings are stored per line.

Quad format - data is for storing chronologies with sample depth as well as data on how many of the constituent series increase and decrease. This format therefore requires four numbers for each data point: ring-width; sample depth; increasing series; decreasing series. Numbers are stored as integers, left space padded as before, but this time only using 5 characters not 6. Four data points are included on each line, therefore this means there are 16 numbers per row and each row is 80 characters long.

H.2 Example file - raw series

```
HEADER:
DateEnd=-66
KeyNo=27
Project=Growth studies
Length=103
Location=Example site
Species=PISY
SapWoodRings=14
WaldKante=WKF
State=Colorado
PersId=FR
KeyCode=271017
Country=USA
DateOfSampling=19950506
TreeNo=5
CoreNo=1
Exposition=North-West
CreationDate=19970526
SoilType=Sand
DATA:Tree
     125     130      99     120     115     145     151     130     135     151
     200     190     151     170     170     174     170     200     210     130
     180     197     210     160     180     155     180     199     140     150
     146     140     145     150     155     110     115     113     120     130
     110     120     150     120     120     110     115     160     160     145
     135     145     125     115     145     149     120     150     160      99
     110      75      70      82      96      90     120     151     155     130
     132     133     149     110     130     120     128     118     125     115
      95      90     110      98      80      85      97      88      70     100
      90      70      80      90      85      78      95      84      70      90
      80      75      70       0       0       0       0       0       0       0
```

H.3 Example file - chronology

```
HEADER:
KeyCode=ABCK0530
DataFormat=HalfChrono
SeriesType=Mean curve
Length=60
DateBegin=987
DateEnd=1046
Dated=Dated
Location=Example site
Species=QUSP
GlobalMathCommentCount=0
ImageCount=0
CommentCount=0
BibliographyCount=0
DATA:Double
     125       1     125       2     264       2     206       2     115       2
     111       2     188       2     308       2     197       2     419       2
     238       2     227       2     279       2     293       2     271       2
     309       2     170       2     204       2     163       2     175       2
     164       2     211       2     134       2     141       2     107       2
      72       2      74       2      91       2     110       2      47       2
      87       2      87       2      35       2      47       2      80       2
      66       2      38       2      82       2      78       2      65       2
      63       2      76       2      67       2      91       2      73       3
      39       3      41       3      78       3      57       3      54       3
      41       3      39       3      52       3      53       3      43       3
      48       3      32       3      32       3      48       3      59       3
```

Appendix I

Microsoft Excel 97/2000/XP

Format name	Microsoft Excel 97/2000/XP
Other name(s)	Binary Interchange File Format, BIFF
Type	Binary file
Extension(s)	xls
Read/write support	Read and write
Reference implementation	Microsoft Excel
Data / metadata	Data only
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	No
Relative dating support	No
Multi series support	Yes
Original designer	Microsoft

I.1 Description

The Excel file format is a widely used format for storing spreadsheet data. It is a proprietary binary format created by Microsoft but suppported by many spreadsheet and statistical applications. It is not to be confused with the Office Open XML format which was introduced by Microsoft with MS Office 2007 and typically has the file extension xlsx.

Although Excel files can contain multiple sheets in a workbook, only the first sheet is considered. Like the CSV and ODF Spreadsheet formats, support for Excel files is limited to a particular layout or style of spreadsheet. The layout of the data sheet should be as follows:

- Row 1 - Header names for each column
- Column A - Year values
- Column B+ - One column for each series containing values in millimetres. Cells are left empty if no data is available for a series because it does not extend to a particular year. Data must be continuous for each series, so missing/unmeasured rings should be included as zero.

I.2 Example file

	A	B	C
1	Year	MySample1	MySample2
2	1954	0.33	
3	1955	0.26	0.26
4	1956	0.2	0.2
5	1957	0.14	0.14
6	1958	0.08	0.08
7	1959	0.02	0.02
8	1960	0.2	0.2
9	1961	0.14	0.14
10	1962	0.08	0.08
11	1963	0.2	
12	1964	0.33	
13	1965	0.08	
14	1966	0.33	
15	1967	0.22	
16			

Appendix J

Microsoft Excel 2007

Format name	Microsoft Excel 2007
Other name(s)	Office Open XML Spreadsheet, OOXML, OpenXML
Type	XML file
Extension(s)	xlsx
Read/write support	Read and write
Reference implementation	ISO 29500
Data / metadata	Data only
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	No
Relative dating support	No
Multi series support	Yes
Original designer	Microsoft

J.1 Description

This is the new XML file format introduced by Microsoft with Excel 2007. Unlike the binary format used by the previous version of Excel, this format is an open standard. However, it should not be confused with the OpenDocument Format standard that was developed by the OASIS consortium.

The layout of the data sheet should be just as for the Excel 97/2000/XP format:

- Row 1 - Header names for each column
- Column A - Year values
- Column B+ - One column for each series containing values in millimetres. Cells are left empty if no data is available for a series because it does not extend to a particular year. Data must be continuous for each series, so missing/unmeasured rings should be included as zero.

See the screenshot in the Microsoft Excel 97/2000/XP format to see how an example of how the spreadsheet should look.

Appendix K

Nottingham

Format name	Nottingham
Other name(s)	Nottingham Laboratory format
Type	Text file
Extension(s)	txt
Read/write support	Read and write
Reference implementation	Unknown
Data / metadata	Data only
Calendar type	n/a
Absolute dating support	No
Undated series support	Yes
Relative dating support	No
Multi series support	Yes
Original designer	Cliff Litton

K.1 Description

The Nottingham format was designed by Cliff Litton. It is a simple text format with no support for metadata.

Line 1 contains a series name and an integer indicating how many data values there are in the file. Subsequent lines contain the data represented as 1/100th mm integers in twenty columns seemingly in either 4 characters or 3 characters + 1 space.

There is no known reference implementation for this format and few known examples of data so little is known about how it should handle unusual situations such as negative values, values ¿999 etc.

K.2 Example file

```
ABCD01      176
 342 338 334 409 362 308 360 264 325 318 134 151 219 268 290 222 278 258 173 198
 294 202 170 176 172 121  87 130 114 108 170 135 131 126  87 100  86 104 103 127
 112  94  96 120 168 149 119 124  79  67  88  90  93  77  49  42  53  38  57  43
  50  41  56  66  62  55  55  45  47  63  58  60  44  45  49  50  62  61  43  54
  91  60  56  43  52  51  65  68  55  44  41  75  94  78  63  69  58  75  55  47
  58  46  62  45  52  50  77  50  63  75  77  64  66  57  80  57  78  65  68  75
  65  98  85  82 119  89  85  87  83 108 129 123 160 117 129 121  88  69  97  77
  96 106  71  89  50  65 133  89  88  50  60  95  95  91 102 158  83  55  98  70
  45  46  40  36  64  58  52  58  56  94  51  48  47  60  49  48
```

Appendix L

ODF Spreadsheet

Format name	ODF Spreadsheet
Other name(s)	ODF, ODS, OpenDocument Spreadsheet, OpenOffice.org Spreadsheet,
Type	XML file
Extension(s)	ods
Read/write support	Read and write
Reference implementation	ISO/IEC 26300:2006
Data / metadata	Data only
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	No
Relative dating support	No
Multi series support	Yes
Original designer	OASIS consortium

L.1 Description

The OpenDocument Format (ODF) spreadsheet format is an XML-based specification developed by the Organization for the Advancement of Structured Information Standards (OASIS) consortium. It should not be confused with the similarly named Office Open XML format developed by Microsoft. The ODF spreadsheet format is an open standard which can be read by most modern spreadsheet applications including MS Excel, OpenOffice.org and Google Docs.

Support for ODF spreadsheets in TRiCYCLE is necessarily limited to a particular layout of spreadsheet:

- Row 1 - Header names for each column
- Column A - Year values
- Column B+ - One column for each series containing values in millimetres. Cells are left empty if no data is available for a series because it does not extend to a particular year. Data must be continuous for each series, so missing/unmeasured rings should be included as zero.

Please see the Excel section for a screenshot of how an ODF spreadsheet should look.

Appendix M

Oxford

Format name	Oxford
Other name(s)	Dan Miles Format, English Heritage Format
Type	Text file
Extension(s)	Various including dan, ddf but often none
Read/write support	Read and write
Reference implementation	Various English Heritage applications
Data / metadata	Data only
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	Yes
Relative dating support	Yes
Multi series support	No
Original designer	Ancient Monuments Laboratory of English Heritage

M.1 Description

The Oxford format seems to be only currently used in the Oxford Dendrochronology Laboratory. It was designed in the 1980s for use with a number of DOS based applications for the English Heritage Ancient Monuments Laboratory. It is still actively used by the Oxford Lab with these programs and a number of newer Windows applications.

The file is a text file format containing two header lines following by a block of data values and an optional block of count/sample depth values. Some files also contain a number of comment lines at the end of the file.

Line 1 contains the following fields:

- Char 1 - Apostrophe
- Chars 2-8 - Series name
- Char 9-10 - spaces
- Char 11 - ¡
- Chars 12-15 - First year in sequence (when series is securely dated). Year should be left padded with spaces if less than 4 characters.
- Char 16 - hyphen
- Chars 17-20 - Last year in sequence (when series is securely dated). Year should be left padded with spaces if less than 4 characters.
- Char 21 - space
- Char 22+ - Description - typically name of site/building etc
- Final char - optional apostrophe

Line 2 contains:

- Integer number of years
- Comma
- Integer start year

The start year on line 2 and the first year on line 1 will be the same for securely dated series. When the series is tentatively or relatively dated the first year (and/or) the last year on line 1 will be left blank. For undated series the start year is set to 1001.

The data lines follow the two header lines. These typically contain 10 data values per line, but there can be more (if rings have been added) or less e.g. last line. The values are in 1/100th mm integers and can only contain three digits (e.g. max 999 1/100th mm). Data values are space delimited. Some example files contain values that are left padded with zeros if the value is on 1 or 2 characters wide (e.g. '025' rather than ' 25').

Following the data values there should be an empty line followed by an optional sample count/depth block. The count block is formatted in largely the same way as the data values block. The values are stored in columns 2 characters (rather than 3 characters) wide. Like the data values, the count values are space delimited integers, typically (but not always) 10 per line.

The file is terminated with 0, 1 or 2 free-text comment lines. A number of Oxford data files have been seen that terminate with the ASCII control character referred to variably as 'SUB', 'SUBSTITUTE' or 'CTRL+Z' (represented in Unicode as character dec 26 - hex 1A). It is not clear whether this is necessary for any particular programs to function.

M.2 Limitations

- Only holds whole ring-width data
- Does not cope with data values ¿999 1/100th mm
- Does not cope with chronologies of ¿99 samples
- Does not allow dates before 1AD

M.3 Example file

```
'ABCD       <1850-1925> A Fictious site - abcd1 abcd2'
75,1850
422 582 355 266 225 271 361 235 387 395
794 611 446 248 277 359 111 226 189 711
464 172 190 239 128 153 234 828 207 157
768 180 178 168 204 163 160 255 166 136
182 201 142 188 223 186 150 135 134 666
191 122 223 555 123 126 108 133 137 134
161 222  93 100 132 104  86 277 101 141
185 151 261 110 145

  1  2  2  2  2  2  2  2  2  2
  2  2  2  2  2  2  2  2  2  2
  2  2  2  2  2  2  2  2  2  2
  2  2  2  2  2  2  2  2  2  2
  2  2  2  2  2  2  2  2  2  2
  2  2  2  2  2  2  2  2  2  2
  2  2  2  2  2  2  2  2  2  2
  2  2  2  2  1
```

Appendix N

PAST4

Format name	PAST4
Other name(s)	P4P PAST4 Project File
Type	Text file
Extension(s)	p4p
Read/write support	Read and write
Reference implementation	PAST4
Data / metadata	Data and some structured metadata
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	
Relative dating support	
Multi series support	Yes
Original designer	Bernhard Knibbe

The PAST4 format (**?**) is the native file format for SCIEM's PAST4 software. It is a hybrid XML file, containing most metadata in structured XML but some metadata and all data as plain text. It is unique amongst dendro data formats in that it contains not only data and metadata but also settings information for the PAST4 software such as details on what colours to use in graphs, which series should be displayed on screen etc. The general structure of a P4P file is as follows:

- Project header (required)
- Settings (optional)
- Groups (required, repeatable)
- Records (required, repeatable)

The root XML tag for the file is <PAST_4_PROJECT_FILE>. Inside this is the <PROJECT> tag which contains the following attributes:

- ActiveGroup - Zero based index specifying which group is active
- EditDate - Date the file was last edited
- Groups - Number of groups within this project
- Locked - Either TRUE or FALSE indicating whether a password is required to open the file
- Name - Name of the project
- Password - Password used to lock the project
- PersID - Abbreviation of the authors name
- Records - Number of records in the project
- Reference - Zero based index indicated which is the reference series (-1 if none selected)
- Sample - Zero based index indicating which is the selected sample (-1 if none selected)
- Version - Version number for this PAST4 format. At the time of writing only one version exists (400).

Of these fields only Name, Groups and Records are mandatory. The project tag can also contain a `<![CDATA[` tag which allows the storing of a project description in plain text.

Next comes the `<SETTINGS>` tag. This is one very large XML tag with many attributes controlling the what PAST4 should display the data. The contents of this tag are optional and are therefore irrelevant for the transfer of dendro data.

Next comes one or more `<GROUPS>` tags. A group is an arbitrary collection of series, perhaps representing a number of measurements of a single object, or perhaps an administrative collection of series. Groups can be nested in a hierarchy, but rather than use the hierarchical nature of XML files, the format instead lists all groups side-by-side and maintains the relationships through the use of an 'owner' attribute containing the index of the parent group. This arrangement means than any changes to the hierarchy, or the deletion of a group requires all indices to be carefully updated to avoid corrupting the file. The group tag has the following attributes:

- Name - Name of the group
- Visible - Either TRUE or FALSE indicating whether the group should be shown in graphs
- Fixed - Either TRUE or FALSE indicating whether the group can be moved
- Locked - Either TRUE or FALSE. If locked the group can be used in the calculation of further mean values.
- Changed - Internal TRUE or FALSE value for keeping track of changes
- Expanded - TRUE or FALSE value indicating whether the group should be expanding in the project navigator window
- UseColor - TRUE or FALSE value for is content should be displayed in color
- HasMeanValue - TRUE or FALSE indicating if the group has a dynamic mean value
- IsChrono - TRUE or FALSE indicating if the group mean is calculated with sample depth information
- Checked - TRUE or FALSE indicating if the group is locked and checked
- Selected - TRUE or FALSE indicated in the group is selected in the project navigation window
- Color - 24bit integer indicating the RGB volor value for the group using Borland format
- Quality - Integer value describing the quality of the group mean
- MVKeycode - String code for the group. If empty the Name field is used
- Owner - Integer pointing containing the index of the parent group if this group is in a hierarchy. If its a top level group it should be -1.

As with the project tag, the group tag can also contain a `<![CDATA[` section for storing a plain text description of the group.

The final tag type in the file is the `<RECORDS>` tag. These contain the actual data series and most of the metadata. Like group tags, records tags are placed side-by-side in the file and are placed into the group hierarchy by the use of the 'owner' attribute. In addition, the tag also has the following attributes:

- Keycode - Name of the series
- Length - Integer for the number of rings
- Owner - Integer index to the group to which this record belongs
- Chrono - TRUE or FALSE indicating whether this record has density information
- Locked - TRUE or FALSE indicating in the record can be moved
- Filter - TRUE or FALSE indicating if an indexing function is appled to the data
- FilterIndex - Integer index for the filter used
- FilterS1 - Parameter 1 for the filter
- FilterS2 - Parameter 2 for the filter
- FilterB1 - Additional filter parameter
- FilterWeight - Additional filter parameter
- Offset - Position of the first ring
- Color - 24bit RGB color for record in Borland format
- Checked - TRUE or FALSE indicating is the record is selected for use in the dynamic group mean
- !VShift - Temporary integer value added to data value to shift vertically in graphs
- IsMeanValue - TRUE or FALSE indicating if this is a dynamic mean value
- Pith - TRUE or FALSE
- SapWood - Integer storing the number of sapwood rings
- Location - String location information

- Waldkante - String description of presence of waney edge
- FirstValidRing - Integer indicating which ring is the first valid ring. If ¿0 then some rings are discarded
- LastValidRing - Integer indicating which ring is the last valid ring. If ¿0 then some rings are discarded
- UseValidRingsOnly - TRUE or FALSE - internal use only
- Quality - Integer indicating the quality of the record

The record tag then contains a `<HEADER>` tag with a `<![CDATA[` section which includes additional free-text header information. There are no requirements as to how information should be laid out in this field however many users seem to adopt the Heidelberg style of keyword=value.

Next comes the `<DATA>` tag which is empty except another `<![CDATA[` section. This is where the actual ring-width data is stored. Each data value is recorded on a separate line (using CR LR line breaks). Each line contains the following six tab delimited fields:

- Ring width as a floating point number
- Sample depth
- Number of sample increasing
- Latewood percentage as a floating point value 0-1 (0 if not known)
- Duplicate/backup ring-width value to store the original ring-width value. If an index is applied the ring-width value in column 1 is altered.
- Comment string about this particular ring

N.1 Dating

PAST4 contains an option for enabling/disabling the year 0 but it does not record within the data file whether the option was set when the file was created. By default the year 0 is disabled therefore the library treats PAST4 files as if they use the Gregorian calendar but it is possible that files were in fact created with the Astronomical calendar in mind.

N.2 Example file

```
<?xml version="1.0"?>
<PAST_4_PROJECT_FILE>
    <PROJECT Name="title0" Version="400" Locked="FALSE" Password=""
        CreationDate="04/05/2006 2:13:51 PM" EditDate="09/01/2010 13:02" ActiveGroup="0"
        Reference="-1" Sample="-1" PersID="investigator0" Groups="2" Records="3">
<![CDATA[description0
]]></PROJECT>
    <SETTINGS/>
    <GROUP Name="title1" Visible="TRUE" Fixed="FALSE" Locked="FALSE" Changed="FALSE"
        Expanded="TRUE" UseColor="TRUE" HasMeanValue="FALSE" IsChrono="FALSE"
        Checked="FALSE" Selected="FALSE" Color="0" MVKeycode="" Owner="-1">
        <![CDATA[]]></GROUP>
    <GROUP Name="Unnamed Group" Visible="TRUE" Fixed="FALSE" Locked="FALSE" Changed="FALSE"
        Expanded="TRUE" UseColor="TRUE" HasMeanValue="FALSE" IsChrono="FALSE" Checked="FALSE"
        Selected="FALSE" Color="0" MVKeycode="" Owner="-1"><![CDATA[]]></GROUP>
    <RECORD Keycode="title6" Length="4" Owner="0" Chrono="FALSE" Locked="FALSE" Filter="FALSE"
        FilterIndex="-1" FilterS1="100" FilterS2="100" FilterB1="FALSE" FilterWeight="" Offset="0"
        Color="0" Checked="FALSE" VShift="0" IsMeanValue="0" Pith="FALSE" SapWood="0"
        Location="locationComment1" Species="Quercus" Waldkante="" FirstValidRing="0"
        LastValidRing="0" UseValidRingsOnly="FALSE">
        <HEADER><![CDATA[Unit=1/100th millimetres
]]></HEADER>
        <DATA><![CDATA[123    1    1    0    123
123    1    1    0    123
123    1    1    0    123
125    1    1    0    125
]]></DATA>
    </RECORD>
    <RECORD Keycode="title6" Length="4" Owner="0" Chrono="FALSE" Locked="FALSE" Filter="FALSE"
        FilterIndex="-1" FilterS1="100" FilterS2="100" FilterB1="FALSE" FilterWeight="" Offset="0"
        Color="0" Checked="FALSE" VShift="0" IsMeanValue="0" Pith="FALSE" SapWood="0"
        Location="locationComment1" Species="QUSP" Waldkante="" FirstValidRing="0"
        LastValidRing="0" UseValidRingsOnly="FALSE">
        <HEADER><![CDATA[Unit=1/100th millimetres
]]></HEADER>
        <DATA><![CDATA[123    1    1    0    123
123    1    1    0    123
123    1    1    0    123
125    1    1    0    125
]]></DATA>
    </RECORD>
    <RECORD Keycode="Unnamed series" Length="2" Owner="1" Chrono="FALSE" Locked="FALSE"
        Filter="FALSE" FilterIndex="-1" FilterS1="100" FilterS2="100" FilterB1="FALSE"
        FilterWeight="" Offset="0" Color="0" Checked="FALSE" VShift="0" IsMeanValue="0"
        Pith="FALSE" SapWood="0" Location="" Species="" Waldkante="" FirstValidRing="0"
        LastValidRing="0" UseValidRingsOnly="FALSE">
        <HEADER><![CDATA[Unit=Wierd units
]]></HEADER>
        <DATA><![CDATA[96    1    1    0    96    fire_damage; fire_damage;
34    1    1    0    34    fire_damage; fire_damage;
]]></DATA>
    </RECORD>
</PAST_4_PROJECT_FILE>
```

Appendix O

Sheffield

Format name	Sheffield
Other name(s)	D Format
Type	Text file
Extension(s)	.d
Read/write support	Read and write
Reference implementation	Dendro for Windows
Data / metadata	Data and some structured metadata
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	No
Relative dating support	Yes
Multi series support	No
Original designer	Ian Tyers

O.1 Description

Sheffield format (**?**) is a dendro specific text file designed by Ian Tyers for his Dendro for Windows application. It is probably most widely used in the UK but is also used in continental Europe as well as New Zealand.

The format contains both data and some structured metadata with each field/value stored one per line. The order of fields is fixed so missing data must be indicated by the use of a question mark. The data present on each line is as follows:

1. Site name/sample number - Free form text not including `,"()` up to 64 characters
2. Number of rings - Whole positive number
3. Date type - Single character; A = absolute date, R = relative date
4. Start date - Whole number (can be negative). If absolute year then add 10000 to value so 1AD = 10001
5. Raw data type *or* Mean data type
    - Single character; R = annual raw ring-width data (NB earlier versions used some other codes here for species e.g. ABEFPSU these are all interpreted as equivalent to R)
    - Single character; W=timber mean with signatures, X=chron mean with signatures, T = timber mean, C = chron mean, M = un-weighted master sequence
6. Raw sapwood number *or* mean number of timbers/chronologies
    - Whole positive number or 0
    - Whole positive number
7. Raw edges inf. *or* Mean chronology type
    - Single character; Y = has bark, ! = has ?bark, W = terminal ring probably complete (i.e. possibly Winter Felled), S = terminal ring probably incomplete (i.e. possibly Summer Felled), B = has h/s

  boundary, ? = has ?h/s boundary, N = has no specific edge, (NB but may have sap), U = sap/bark unknown, C = charred outer edge, P = possibly charred outer edge
  - Single character; R = raw unfiltered data, 5 = 5 year running mean, I = indexed data, U = unknown mean type

8. Author and comment - Free form text not including `,"()` up to 64 characters
9. UK National grid reference - 2 characters +even no of digits up to 14 characters in all, ? = not known e.g. TQ67848675
10. Latitude and longitude - Either decimal format e.g. 53.382457;-1.513623 or previously N51^30 W1^20
11. Pith - single character; C = centre of tree, V = within 5 years of centre, F = 5-10 years of centre, G = greater than 10, ? = unknown
12. Cross-section code - Two character code; first character, A = whole roundwood, B = half round, C quartered, D radial/split plank, E tangential/sawn plank. second character, 1 untrimmed, 2 trimmed, X irregularly trimmed. or, X = core /unclassifiable, ? unknown/unrecorded
13. Major dimension - whole number in mm, 0 if unrecorded or mean
14. Minor dimension - whole number in mm, 0 if unrecorded or mean
15. Unmeasured inner rings - single character+whole number; use pith codes + number of rings or, H = heartwood, N = none
16. Unmeasured outer rings - single character+whole number; use edges code + number of rings except that S = sapwood with no edge and V is the spring felling equivalent other codes are, H = heartwood with no edge, N = none
17. Group/Phase - free form text not including , " ( ) up to 14 characters
18. Short title - free form text not including , " ( ) up to 8 characters
19. Period - single character; C = modern, P = post medieval, M = medieval, S = Saxon, R = Roman, A = pre Roman, 2 = duplicate e.g. repeat measure, B = multiperiod e.g. long master, ? = unknown
20. ITRDB species code - 4 character code - refer to ITRDB species codes
21. Interpretation and anatomical notes - ? =no interpretation/notes. The interpretation and the anatomical notes can be in any order but each must consist of three parts, a single character A or I for anatomy or interpretation, a separator , for interpretations the date of the start, for anatomy the ringno, a separator , for anatomy the anatomical code for interpretations P for plus, 0 for felled and a number for the length of the range, where more than one record is present these are separated by , there must not be a terminal separator and each record must consist of the tree parts. The anatomical codings can be anything of a single character but supported usage is based on Hans-Hubert Leuschners anatomical codes; D = Density Band, R = Reaction Wood, L = Light Latewood, H = Dense Latewood, F = Frost Ring, K = Small Earlywood Vessels - oak, G = Great Latewood Vessels - oak, T = Wound Tissue, N = Narrow Latewood, A = Light Latewood End, P = Narrow and Light Latewood, Q = Narrow and Dense Latewood
22. Data type - single character; D = ring widths, E = early-wood widths only, L = late-wood widths only, R = late+early wood widths (i.e. reverse of normal rings), I = minimum density, A = maximum density, S = early, late; (i.e. sequentially and separately), M = mixed (?means of others)

The remaining lines contain the data:

- For each width (equivalent to the value of length) the individual increments etc. if a C X T or W type mean. No negatives or zeros
- Check field - Single character H
- For each width the individual weightings of the mean sequences. If an X or W type mean. No negatives or zeros.
- Check field - Single character R
- For each width the number of individual series with rising values. No negatives or zeros.
- Check field - Single character F
- For each width the number of individual series with falling values. No negatives.

O.2 Dating

The format copes with the problem of the non-existent year 0AD/BC by adding 10000 to all year values. Therefore:

Year	Value in file
1AD	10001
1BC	10000
9999BC	2
10000BC	1

O.3 Example file

```
Ship wreck 4 timber mean
170
A
10784
W
4
R
made PB 22/6/2004
?
?
?
?
0
0
N
N
A
Example
M
QUSP
?
D
391
454
309
314
270
273
229
319
267
276
128
163
221
269
214
201
218
199
198
209
156
177
...
```

Appendix P

Topham

Format name	Topham
Other name(s)	Instrument format
Type	Text file
Extension(s)	txt
Read/write support	Read and write
Reference implementation	
Data / metadata	Data only
Calendar type	n/a
Absolute dating support	No
Undated series support	Yes
Relative dating support	No
Multi series support	No
Original designer	John Topham

P.1 Description

The Topham format is probably the most simplistic of formats consisting of just a column of decimal data values and no metadata whatsoever. Each data value is a decimal ring width in millimetres.

P.2 Example file

```
3.42
3.38
3.34
4.09
3.62
3.08
3.60
2.64
3.25
3.18
3.42
3.38
...
```

Appendix Q

TRiDaS

Format name	TRiDaS
Other name(s)	Tree-Ring Data Standard, TRiDaS XML
Type	Text file
Extension(s)	xml
Read/write support	Read and write
Reference implementation	TRiCYCLE
Data / metadata	Data and structured metadata
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	Yes
Relative dating support	Yes
Multi series support	Yes
Original designer	Esther Jansma, Peter Brewer and Ivo Zandhuis

Q.1 Description

TRiDaS (Tree-Ring Data Standard see *http://www.tridas.org*) is a data format designed by over 80 dendrochronologists, computer scientists and users of dendrochronological data from a variety of associated fields as part of the DCCD project and the Dendro Data Standard forum. It is designed to accurately represent any dendro data and metadata and it is hoped over time the dendro community will accept TRiDaS as the de facto standard for all dendro data.

The format uses extensible markup language (XML) which means the standard can be extended and evolve as future needs change. The format is structured around the eight data entities described below:

A project is defined by a laboratory and encompasses dendrochronological research of a particular object or group of objects. Examples include: the dating of a building; the research of forest dynamics in a stand of living trees; the dating of all Rembrandt paintings in a museum. What is considered a "project" is up to the laboratory performing the research. It could be the dating of a group of objects, but the laboratory can also decide to define a separate project for each object. Therefore, a project can have one or more objects associated with it.

An object is the item to be investigated. Examples include: violin; excavation site; painting on a wooden panel; water well; church; carving; ship; forest. An object could also be more specific, for example: mast of a ship; roof of a church. Depending on the object type various descriptions are made possible. An object can have one or more elements and can also refer to another (sub) object. For instance a single file may contain three objects: an archaeological site object, within which there is a building object, within which there is a beam object. The list of possible object types is extensible and is thus flexible enough to incorporate the diversity of data required by the dendro community. Only information that is essential for dendrochronological research is recorded here. Other related data may be provided in the form of a link to an external database such as a museum catalogue.

An element is a piece of wood originating from a single tree. Examples include: one plank of a water well; a single wooden panel in a painting; the left-hand back plate of a violin; one beam in a roof; a tree trunk preserved in the soil; a living tree. The element is a specific part of exactly one object or sub object. An object will often consist of more than one element, e.g., when dealing with the staves (elements) of a barrel (object). One or more samples can be taken from an element and an element may be dated using one or more derivedSeries.

A sample is a physical specimen or non-physical representation of an element. Examples include: core from a living tree; core from a rafter in a church roof; piece of charcoal from an archaeological trench; slice from a pile used in a pile foundation; wax imprint of the outer end of a plank; photo of a back plate of a string instrument. Note that a sample always exists and that it can either be physical (e.g. a core) or representative (e.g. a picture). A sample is taken from exactly one element and can be represented by one or more radii.

A radius is a line from pith to bark along which the measurements are taken. A radius is derived from exactly one sample. It can be measured more than once resulting in multiple measurementSeries.

A measurementSeries is a series of direct, raw measurements along a radius. A single measurementSeries can be standardised or a collection of measurementSeries can be combined into a derived- Series. The measurements themselves are stored separately as values.

A derivedSeries is a calculated series of values and is a minor modification of the "v-series" concept proposed by **?**. Examples include: index; average of a collection of measurementSeries such as a chronology. A derivedSeries is derived from one or more measurementSeries and has multiple values associated with it.

A value is the result of a single ring measurement. Examples include: total ring width; earlywood width; latewood width. The values are related to a measurementSeries or a derivedSeries. In case of a measurementSeries the variable and its measurement unit (e.g. microns, 1/100th mm etc) are recorded as well.

For a full description of the standard see **?**.

Q.2 Example file

```xml
<?xml version="1.0" encoding="UTF-8"?>
<tridas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.tridas.org/1.2.1 ../dev/sourceforge/tridas/XMLSchema
        /1.2.1/tridas-1.2.1.xsd"
    xmlns="http://www.tridas.org/1.2.1" xmlns:xlink="http://www.w3.org/1999/xlink">
    <project>
        <title>Aegean Dendrochronology Project</title>
        <identifier domain="dendro.cornell.edu">C</identifier>
        <createdTimestamp certainty="exact">1997-02-01T14:13:51.0Z</createdTimestamp>
        <lastModifiedTimestamp certainty="exact">1997-02-01T14:13:51.0Z</
            lastModifiedTimestamp>
        <type>Dating</type>
        <description>Our key long-range goal is to build long multi-millennial scale tree
            -ring
            chronologies in the Aegean and Near East that will extend from the present to
                the
            early Holocene to cover, broadly speaking, the last 10,000 years of human and
            environmental history. Our raison d'etre is to provide a dating method for
                the study
            of history and prehistory in the Aegean that is accurate to the year. This
                kind of
            precision has, up to now, been lacking in ancient studies of this area.
                Indeed, few
            archaeological problems stimulate as much rancor as chronology, especially
                that of
            the Eastern Mediterranean. The work of the Aegean and Near Eastern
                Dendrochronology
            Project aims to help to bring some kind of rational and neutral order to
                Aegean and
            Near Eastern chronology from the Neolithic to the present. </description>
        <laboratory>
```

```
            <name>Malcolm and Carolyn Weiner Laboratory for Aegean and Near Eastern Dendrochronology</name>
            <address>
                <addressLine1>B48 Goldwin Smith Hall</addressLine1>
                <addressLine2>Cornell University</addressLine2>
                <cityOrTown>Ithaca</cityOrTown>
                <stateProvinceRegion>NY</stateProvinceRegion>
                <postalCode>14853</postalCode>
                <country>USA</country>
            </address>
        </laboratory>
        <category>Archaeology</category>
        <investigator>Peter I Kuniholm</investigator>
        <period>1976-present</period>
        <reference>reference1</reference>
        <object>
            <title>White Tower, Thessaloniki</title>
            <identifier domain="dendro.cornell.edu"
                >28acb483-f337-412f-a063-59d911c37594</identifier>
            <createdTimestamp certainty="exact">1997-02-01T14:13:51.0Z</createdTimestamp>
            <lastModifiedTimestamp certainty="exact">1997-02-01T14:13:51.0Z</lastModifiedTimestamp>
            <type normalStd="Corina Dictionary" normalId="4" normal="Building">Building</type>
            <description>The White Tower of Thessaloniki was originally constructed by the Ottomans
                to fortify the city's harbour.</description>
            <coverage>
                <coverageTemporal>Ottoman</coverageTemporal>
                <coverageTemporalFoundation>Stylistic</coverageTemporalFoundation>
            </coverage>
            <location>
                <locationGeometry xmlns:gml="http://www.opengis.net/gml">
                    <gml:Point srsName="urn:ogc:def:crs:EPSG:6.6:4326">
                        <gml:pos>40.6263 22.9485</gml:pos>
                    </gml:Point>
                </locationGeometry>
                <locationPrecision>20</locationPrecision>
                <locationComment>Thessaloniki, Greece</locationComment>
            </location>
            <object>
                <title>Fourth floor</title>
                <type>Floor</type>
                <element>
                    <title>C-TWT-65</title>
                    <identifier domain="dendro.cornell.edu"
                        >89dbd409-03a3-42a0-9391-62c6be7009ad</identifier>
                    <createdTimestamp certainty="exact">1997-02-01T14:13:51.0Z</createdTimestamp>
                    <lastModifiedTimestamp certainty="exact"
                        >1997-02-01T14:13:51.0Z</lastModifiedTimestamp>
                    <type normalStd="Corina Dictionary" normalId="3" normal="Rafter">Rafter</type>
                    <description>15th Rafter from the south</description>
                    <taxon normalStd="Catalogue of Life Annual Checklist 2008" normal="Quercus"
                        normalId="49139">Quercus sp.</taxon>
                    <dimensions>
                        <unit normalTridas="metres"/>
                        <height>1</height>
                        <width>1</width>
                        <depth>1</depth>
                    </dimensions>
                    <authenticity>Original</authenticity>
                    <sample>
                        <title>C-TWT-65-A</title>
                        <identifier domain="dendro.cornell.edu"
                            >ff688357-b2d4-4394-a21a-90696cd4558c</identifier>
                        <createdTimestamp certainty="exact"
                            >1997-02-01T14:13:51.0Z</createdTimestamp>
                        <lastModifiedTimestamp certainty="exact"
```

```
                            >1997-02-01T14:13:51.0Z</lastModifiedTimestamp>
                        <type normal="Corina Dictionary" normalId="1" normalStd="Section"
                            >Section</type>
                        <samplingDate certainty="exact">1981-07-25</samplingDate>
                        <state>Dry</state>
                        <radius>
                            <title>C-TWT-65-A-B</title>
                            <identifier domain="dendro.cornell.edu"
                                >5b7baa8b-cd4e-4b3b-88fa-82939420e544</identifier>
                            <createdTimestamp certainty="exact"
                                >2006-05-04T18:13:51.0Z</createdTimestamp>
                            <lastModifiedTimestamp certainty="exact"
                                >2006-05-04T18:13:51.0Z</lastModifiedTimestamp>
                            <woodCompleteness>
                                <pith presence="absent"/>
                                <heartwood presence="incomplete"/>
                                <sapwood presence="complete"/>
                                <bark presence="present"/>
                            </woodCompleteness>
                            <measurementSeries>
                                <title>C-TWT-65-A-B-A</title>
                                <identifier domain="dendro.cornell.edu"
                                    >8c50234e-8eda-41bb-b578-01cc881d1ea1</identifier>
                                <createdTimestamp certainty="exact"
                                    >1997-02-01T14:13:51.0Z</createdTimestamp>
                                <lastModifiedTimestamp certainty="exact"
                                    >1997-02-01T14:13:51.0Z</lastModifiedTimestamp>
                                <analyst>Laura Steele</analyst>
                                <dendrochronologist>Peter I Kuniholm</dendrochronologist>
                                <measuringMethod normalStd="Corina Dictionary" normalId="1"
                                    >Measuring platform</measuringMethod>
                                <interpretation>
                                    <firstYear suffix="AD">1254</firstYear>
                                    <statFoundation>
                                        <statValue>8.3</statValue>
                                        <type>t-score</type>
                                        <usedSoftware>Corina 2.10</usedSoftware>
                                    </statFoundation>
                                    <deathYear suffix="AD">1535</deathYear>
                                    <provenance>Possibly from the region of Serres</provenance>
                                </interpretation>
                                <values>
                                    <variable normalTridas="Ring width"/>
                                    <unit normalTridas="1/100th millimetres"/>
                                    <value value="54"/>
                                    <value value="111"/>
                                    <value value="71"/>
                                    <value value="40"/>
                                    <value value="56"/>
                                </values>
                            </measurementSeries>
                        </radius>
                    </sample>
                </element>
            </object>
        </object>
    </project>
</tridas>
```

Appendix R

TRIMS

Format name	TRIMS
Other name(s)	None known
Type	Text file
Extension(s)	.rw
Read/write support	Read and write
Reference implementation	
Data / metadata	Data only
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	Yes
Relative dating support	No
Multi series support	No
Original designer	Unknown

This is a simple data only text file format. These files were originally produced using the Henson rotary micrometer measuring stages but have largely been phased out.

- Line 1 - Initials of user that created the series
- Line 2 - Date the file was created in dd/MM/YY format
- Line 3 - Year of first data value (0 treated as undated series)
- Line 4+ - Space character followed by an integer data value in 1/100th mm
- Final line - Space character + 999 denoting end of series.

R.1 Example file

```
pb
05/10/94
 1816
 169
 96
 165
 85
 139
 87
 112
 ...
 999
```

Appendix S

Tucson

Format name	Tucson
Other name(s)	Decadal, RWL, CRN, ITRDB, Time series format, TSF
Type	Text file
Extension(s)	Various including tuc, rwl, dec, crn
Read/write support	Read and write
Reference implementation	COFECHA
Data / metadata	Data with some structured metadata, however, standardisation of metadata is very poor resulting in metadata often being little more than free text comments
Calendar type	Astronomical
Absolute dating support	Yes
Undated series support	No
Relative dating support	No
Multi series support	Yes
Original designer	Richard Holmes

S.1 Description

The Tucson format is perhaps the most widely used dendro data format. Unfortunately it seems there was never definitive documentation. Support for the format has been incorporated into a number of dendro applications but without format documentation there are variations in these implementations resulting in quite a lot of subtle differences in files. The often tight association between the Dendro Program Library (DPL) and the ITRDB means that perhaps the most definitive documentation for the format is the ITRDB website.

The Tucson format is best considered as covering two different sub-formats which are often referred to by their file extensions (RWL and CRN). RWL files are used for storing ring-width data, whereas CRN files are used for storing chronologies.

The ITRDB website includes detailed information on how to include structured metadata in Tucson format files. Unfortunately there are no tools for creating and/or validating Tucson files so the vast majority of files circulating in the community today (including those in the ITRDB) do not adhere to these standards.

S.2 RWL files

Tucson RWL files begin with three lines of metadata. Strictly these lines should contain structured metadata, but with no software to assist in this, users either only partially stick to these rules, or reject them entirely instead using the three lines as free-text comment lines. The metadata should be set out as follows:

- Line 1 - Chars 1-6 Site ID
- Line 1 - Chars 10-61 Site Name
- Line 1 - Chars 62-65 Species Code followed by optional ID number
- Line 2 - Chars 1-6 Site ID
- Line 2 - Chars 10-22 State/Country
- Line 2 - Chars 23-30 Species
- Line 2 - Chars 41-45 Elevation
- Line 2 - Chars 48-57 Lat-Long in degrees and minutes, ddmm or dddmm
- Line 2 - Chars 68-76 1st and last Year
- Line 3 - Chars 1-6 Site ID
- Line 3 - Chars 10-72 Lead Investigator
- Line 3 - Chars 73-80 comp. date

Then follows the data lines which are set out as follows:

- Chars 1-8 - Series ID - the series ID should be unique in the file so that it is clear where one series ends and another begins when multiple series are present in the same file.
- Next 4 chars - Year of first value in this row.
- Ten data values consisting of a space character and 5 integers. The file and last data line for a series may have less than 10 data values so that the majority of lines begin at the start of a decade.

The final data value should be followed by a a stop marker which is either 999 or -9999. When a stop marker of 999 is used this indicates that the integer values in the file are measured in 0.01mm (1/100th mm) units, whereas if a -9999 stop marker is used the units are 0.001mm (microns). The stop marker is therefore used to indicate the end of the data series and the units the data are stored in.

There appears to be no official specification as to how missing rings should be encoded, but the standard notation seems to be to use -999 or 0.

S.3 CRN files

Tucson CRN files are used to store chronology data. In addition to each data values they also have space for a sample depth or count value to record how many values were combined to give each data value. CRN files should strictly begin with the same 3 header lines that are described above for RWL. Like RWL files the specification is often partially adhered to and at times ignored completely.

The data lines for CRN files are quite different to RWL:

- Chars 1-6 - Series ID
- Next 4 chars - Year of first value in this row.
- Ten data value blocks consisting of four integer characters for the data value, then a space, then two integer characters for sample depth.

The stop marker in a CRN file should be 9990.

S.4 Workarounds and quirks

- No information was given as to how to handle the non-existent year 0AD/BC. For data files with years all in the AD period, this is not a problem. Most dendro software seem to treat year numbers in Tucson files as using the 'Astronomical Calendar' whereby 1 = 1AD, 0=1BC, -1=2BC etc. This goes against what most dendrochronologists assume (and do) when using Tucson files. For instance most people that work entirely in the BC period use negative integers to represent BC years e.g. -5 as 5BC. With no clear specification and different people interpreting the format in different ways, there is no way of being certain what data negative year numbers in Tucson files mean.
- Tucson format places a restriction of just four characters to the year values. This means that strictly the earliest value a Tucson file can represent is -999. Some users work around this by steeling the last character of the series ID to give them five characters for the year. For example: ABCDEFG-9999. This

conversely limits the series ID to 7 characters. To add to the confusion, other users have been known to add an arbitrary number (e.g. 5000) to all year numbers to overcome this problem.

- The fact that 999 is used as the stop marker for series in 1/100th mm means that Tucson files cannot store a ring value of 9.99mm. In the unlikely event that a sample should have this large a ring, it should be rounded up or down to 998 or 1000.
- Some programs appears to add padding values after the stop marker to fill the rest of the 10 data values in the row.
- Some data files seem to use 9990 as a stop marker
- Some files appears to use a full-stop character to indicate empty data values after the stop marker.
- Data values in RWL files are space delimited, however some programs use tabs instead.
- When reading Tucson files, COFECHA and ARSTAN ignore all lines that do not match the standard data line format. As such, some users have used this to enable them to include multiple comment lines in their files.
- The ITRDB documentation says they should be recorded as DDMM or DDDMM, but this along with sign (N,S,E,W,+ or -) would require 11 characters, when the Tucson specification only allows for 10. Perhaps this was due to an assumption that all places would be in the northern hemisphere? This has resulted in a large amount of variation in the way that coordinates are recorded making it extremely difficult to parse them without error. Here are some examples (including some that use 11 chars not 10):
  - 4652N01101E
  - +4652-01101
  - N4652E01101
  - 4652-01101
  - 465201101
  - 4652 01101

S.5 Example file - raw series

```
107     1 OBERGURGL
107     2 AUSTRIA       NORWAY SPRUCE       6726   4652N01101E      1911  1959
107     3 GIERTZ                                                      08    76
107011   1911    78    93    43   100    93   110   135   115   102
107011   1920    92   125   110   135    98    80    75   125   102   110
107011   1930   105   105    95   120   135   140   110   120   130   135
107011   1940   120   130   130   165   135   145   155   160    88   135
107011   1950   140   150   140   130   115   130   130   110   110   135
107011   1960   125   120   135   160   15    102   105   135   105   140
107011   1970   120   115   100   110   110   999
107012   1862   450   580   550   480   620   420   390   420
107012   1870   360   370   300   360   470   460   410   430   510   500
107012   1880   500   510   500   410   380   430   340   380   350   400
107012   1890   290   260   270   320   340   370   330   310   240   170
107012   1900   280   300   300   310   350   400   300   280   280   180
107012   1910   190   290   270   210   230   300   220   360   240   260
107012   1920   200   270   250   230   270   210   160   210   220   200
107012   1930   170   250   200   130   140   210   210   180   190   180
107012   1940   170   180   190   190   190   200   190   180   110   180
107012   1950   220   230   180   220   200   240   220   210   240   999
```

S.6 Example file - chronology

```
107089 1 Antalya, Elmali Isletmesi                             CDLI
107089 2 Turkey        Cedar              1800M  3640 02955          1370 1988
107089 3 Peter I. Kuniholm
1070001370 567  11115  1 798  11105  11407  1 398  1 436  1 543  1 490  1 225  1
1070001380 127  1  39  1  29  1  69  1 178  1 445  1 227  1 510  11020  11120  1
10700013901390  11310  1 979  11585  11111  1 444  1 214  1 520  1 275  1 224  1
1070001400 153  1 371  1 567  1 711  1 835  1 687  1 322  1 291  1 291  1 218  1
1070001410 168  1 378  1 557  1 410  1 315  1 202  1 531  1 765  1 797  1 840  1
1070001420 440  1 774  1 946  1 838  1 397  1 380  1 206  1 510  1 695  1 521  1
1070001430 461  1 978  1 967  1 857  1 978  1 733  1 522  1 333  1 577  1 477  1
1070001440 730  1 752  1 932  1 955  1 898  1 629  11170  1 738  1 920  1 363  1
```

```
1070001450 863  1 896  1 965  1 390  1 172  1 126  1  69  1 209  1 313  1 883  1
10700014601255  11220  11364  11035  11364  11282  11364  11611  11369  11273  1
10700014701797  12035  11821  11927  11819  11807  11464  21421  21009  21089  2
10700014801042  21040  21404  2 955  21291  2 982  21186  21042  2 728  2 781  2
1070001490 800  21040  2 503  2 869  21387  21365  21574  21591  22178  21594  3
10700015001629  31282  31126  31409  31433  31406  31239  31479  3 990  31063  3
10700015101026  31035  31175  31217  31500  31358  31171  31140  31005  31340  3
10700015201225  31164  31283  31496  31439  31603  31335  3 982  3 973  31147  3
10700015301086  31146  41403  41454  41209  41451  41292  4 964  41003  41289  4
1070001540 895  4 951  4 745  4 835  4 800  41182  4 952  41097  4 973  4 973  4
10700015501158  41370  41245  41392  41215  41047  51133  5 847  5 961  51295  5
10700015601287  51082  5 899  51012  51195  51409  51107  5 962  5 970  51031  5
1070001570 990  51028  51206  51092  51414  51209  51090  51265  51261  51019  5
1070001580 791  5 995  5 956  5 933  61144  61022  61001  61007  61097  61290  6
10700015901263  6 902  71002  71151  71032  8 968  8 592  8 940  8 936  81131  8
10700016001098  81128  81334  81255  91136  91097 101273 101075 10 952 10 897 10
1070001610 915 10 991 10 735 10 708 10 627 10 848 101010 10 872 10 959 101138 10
10700016201173 101122 101191 101146 10 928 10 820 10 935 10 741 10 812 101126 10
10700016301123 10 781 101111 101054 101275 101052 101068 101049 101016 10 970 10
10700016401093 101159 101023 101159 101060 101117 101314 10 843 101057 101040 10
10700016501030 101268 10 971 101059 101078 101170 101159 101388 101194 101260 10
1070001660 917 101222 101052 101165 101325 101608 101161 121181 12 931 12 992 12
1070001670 750 12 675 12 614 12 638 12 624 12 600 12 506 12 681 12 887 12 708 12
1070001680 797 12 940 12 955 12 886 12 878 12 970 12 916 12 861 12 861 121021 13
1070001690 928 13 961 131043 13 936 13 939 131003 13 619 13 846 13 838 13 822 13
1070001700 717 13 699 14 746 14 900 141022 14 781 14 968 141028 141051 141341 14
1070001710 980 14 817 14 718 14 642 14 554 14 589 14 637 14 677 16 710 16 877 16
1070001720 930 16 931 16 718 16 721 16 616 16 576 16 519 16 790 161046 161067 16
10700017301047 171141 181080 181128 181144 181112 191066 191252 19 971 191076 19
10700017401284 191242 191001 191145 191219 191162 19 576 20 979 231148 231062 23
10700017501119 231255 231267 231352 231397 231487 231116 231092 231150 23 938 23
10700017601118 241240 241258 241023 24 971 241071 241124 241225 241135 241114 24
10700017701072 241171 24 853 24 964 241075 24 820 241154 241059 241270 241022 24
10700017801098 24 903 241038 241147 241141 241162 24 782 241221 241424 241208 24
1070001790 974 241265 241256 241281 241166 241580 24 889 24 955 241158 241101 24
1070001800 949 24 990 24 813 24 758 24 821 24 914 24 889 24 999 24 991 241163 24
10700018101068 241184 24 852 24 870 241037 241070 241132 241047 24 978 24 852 24
1070001820 839 241063 241045 24 957 24 958 24 997 24 841 241209 241053 241013 24
1070001830 920 241103 241151 241166 24 850 24 962 24 944 24 871 24 989 24 906 24
1070001840 697 24 973 24 779 24 647 24 689 24 731 24 981 24 709 24 949 24 580 24
1070001850 619 24 345 24 545 24 688 24 723 241046 24 738 24 785 24 742 24 815 24
1070001860 842 241015 24 888 24 884 24 792 24 594 24 902 24 885 24 841 24 770 24
1070001870 822 24 710 24 838 24 783 24 697 24 768 24 515 24 670 24 855 24 793 24
10700018801021 25 932 25 799 25 902 251038 251017 25 739 25 750 25 963 251149 25
1070001890 798 24 871 24 870 24 625 24 772 24 827 241046 241182 24 701 24 704 24
1070001900 977 241237 241249 241162 241118 241007 241271 241123 241116 241045 24
10700019101167 24 827 24 482 24 952 241370 241260 24 783 241169 241096 241108 24
10700019201387 241484 241293 241182 241282 241527 241261 241146 24 920 24 859 24
10700019301235 241335 24 799 24 819 241000 24 763 241111 241019 24 916 241252 24
10700019401537 241387 231217 23 929 23 685 23 894 231106 231123 231089 23 896 23
10700019501384 231172 231151 231130 231244 231187 231118 231144 231268 231245 23
10700019601547 231015 231208 231203 231109 23 602 23 690 23 684 23 901 23 968 23
1070001970 963 231095 231368 231069 231084 21 936 191146 191362 191288 191572 19
10700019801034 19 870 191186 191047 19 968 191089 191272 191231 191083 189990  0
```

Appendix T

Tucson Compact

Format name	Tucson Compact
Other name(s)	Compact
Type	Text file
Extension(s)	rwm
Read/write support	Read and write
Reference implementation	Various DPL programs including FMT
Data / metadata	Data only
Calendar type	Astronomical
Absolute dating support	Yes
Undated series support	No
Relative dating support	No
Multi series support	Yes
Original designer	Richard Holmes

T.1 Description

The Tucson Compact format was design by Richard Holmes for use with a number of the applications in the Dendro Program Library (DPL). Holmes designed it as a space saving alternative to the standard Tucson format at a time when disk space was expensive. The format never really caught on, perhaps due to the complexity and variability of the format.

The key feature of Tucson Compact format is the inclusion of a code that describes the layout of the data within the series. This code means that only the required amount of space is allocated to each data value in the text file with little wastage. No space is provided for metadata.

Tucson Compact files can contain one or more series of data so the description of a data series below can be repeated multiple times in a single file. All lines should be 80 characters long and the first line of a series is denoted by a tilde ( ) in the final column. This meta line contains four fields:

- Chars 0-9 = number of data values terminated with =N
- Chars 11-19 = start year terminated with =I
- Chars 21-68 = series title
- Chars 69-79 = fortran format descriptor
- Char 80 = Tilde marker

The Fortran format descriptor in the example below is -2(26F3.0). The constituent parts are as follows:

- -2 = this is the scaling factor for the data values. In this case -2 = 10-2 = 0.01. Please note that in the Convert5 program this scaling factor is only read once in the first header line so files with multiple series each with different scaling factors will read incorrectly.

- 26F = means there are 26 values in each line
- 3.0 = means that each data value should be read as 3 integer values

The example below therefore means there are 26 data values per line each consisting of 3 digits which should be interpreted by multiplying by 0.01 (i.e. values are in 1/10ths mm).

T.2 Example file

```
     176=N       1277=I  ABCD01                                                  -2(26F3.0)~
142338334409362308360264325318134151219268290222278258173198294202170176172121
 87130114108170135131126 87100 86104103127112 94 96120168149119124 79 67 88 90
 93 77 49 42 53 38 57 43 50 41 56 66 62 55 55 45 47 63 58 60 44 45 49 50 62 61
 43 54 91 60 56 43 52 51 65 68 55 44 41 75 94 78 63 69 58 75 55 47 58 46 62 45
 52 50 77 50 63 75 77 64 66 57 80 57 78 65 68 75 65 98 85 82119 89 85 87 83108
129123160117129121 88 69 97 77 96106 71 89 50 65133 89 88 50 60 95 95 91102158
 83 55 98 70 45 46 40 36 64 58 52 58 56 94 51 48 47 60 49 48
```

Appendix U

VFormat

Format name	VFormat
Other name(s)	OJ Format
Type	Text file
Extension(s)	Various depending on data type but commonly .!oj
Read/write support	Read and write
Reference implementation	VFormat
Data / metadata	Data with some structure metadata
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	No
Relative dating support	No
Multi series support	Yes
Original designer	Thomas Reimer and Hans-Hubert Leuschner

U.1 Description

A relatively extensive format which includes highly encoded header lines for metadata. VFormat files have an array of file extensions depending on the type of data the files contain.

VFormat files can contain mutliple data series. Each series contains 2-4 header lines followed by a number of data lines. The metadata fields are encoded into the header lines in specific character positions. In line 1 the character positions are as follows:

- 1-12 = Series identifier. The series identifier also determines the filename. If there is just one series in the file then the series identifier will be the same as the filename. For files with multiple series, the filename will use characters 1-7 of the series identifiers that are the same throughout the file with the remaining (different) characters replaced by an underscore. The 8th character of the filename would contain a running number for files that would otherwise be named the same. The series identifier is made up of the following characters:
  - 1 = Code representing the project or country
  - 2 = Code representing the region of ecological area
  - 3-4 = Code number for sample site (optionally encoded using hexadecimal or hexatresimal to enable values greater than 99).
  - 5-6 = Series/tree number (optionally encoded using hexadecimal or hexatresimal to enable values greater than 99).
  - 7 = Height code encoded as follows: 1 = 1m, 2=2m, 9=9m, A=10m, B=11m, S = Lumber height 30cm, T = breast height =130cm.
  - 8 = Running number if several series have the same values in columns 1-7.

  - 9 = Fixed as a dot character
  - 10 = Either ! (single), % (partial), # (mean curves or chronologies)
  - 11 = Code for statistical treatment. One of F (frequency filtered series); I (index); M (mean); O (original); P (pointer-year stat); Q (cluster-pointer-year stat); R (residual); S (moving deviation or variance); T (trend, fitted curve, model); W (wuchswert); X (series with standardized running mean and variance); Z (central moment, deviation or variance between several series).
  - 12 = Code for the measured parameter. One of D (mean density); F (earlywood width); G (maximum density); J (ring width); K (minimum density); P (percentage latewood); S (width of latewood).
- 13-15 Measurement units
- 16-20 Length of series
- 21-24 Species either encoded using ITRDB taxon codes or by using the first two letters of the genus and species.
- 25-30 Year of the last ring
- 31-50 Description
- 51-58 Measurement date (ddMMyy or ddMMyyyy)
- 59-60 Initials of author
- 61-68 Last modified date (ddMMyy or ddMMyyyy)
- 69-70 VFormat version identifier (00,01 etc)
- 71-73 Estimated number of missing rings as the start of the series
- 74-75 Standard error of this estimate (. if unknown)
- 76-78 Estimated number of missing rings at the end of the series
- 79-80 Standard error of this estimate (. if unknown)

The second data line is a free text comment up to 80 characters.

VFormat files from version 10 onwards then contain a third header line. This contains 8 floating point numbers of 10 digits each. These represent:

- Longitude
- Latitude
- Altitude
- Height of the tree's measurement
- Four other user definable numbers

VFormat files from version 20 onwards contain a forth header line. This is of the same format as line 3 but each of the values is user definable.

Following the 2-4 header lines come the data lines. These lines are made up of 10 data fields each containing 8 characters. Each data field is made up as follows:

- Two character code for validity and importance:
  - space = full validity
  - ! = not yet used
  - " = not yet used
  - # = not yet used
  - $ = no validity for long-term evaluations
  - % = no validity for single-value evaluations
  - & = no validity except for cumulative stats
  - ' = no validity at all, unknown value

The second character is a pseudo-binary character used to define a weighting factor. For full details of the complex method for calculating this weighting factor see the VFormat documentation.

- One character user definable code for recording information about the data value
- Five digit floating point data value which is divided by 100 for interpretation

U.2 Example file

```
G1101020.!OJmm     81Qusp  1510FLA-02  32 /572             HL01.04.9002  810 10 .
G1101020.!OJ/S20102_0.!OJ/_
     281     221     225     169     178     197     126     103     112     130
     132     207     176     175     126     150      99     131     187     204
     218     172     202     115     135     130     196     135     142     129
     144     116      92      71     109     120     137      98      86     117
      64      79      72      61      62      82      75      81      83      69
      83      66      84      95      85      94      87      99      92     109
     150     108      70     113     119     120     122     107     111     114
     123     145     112     145     164     158     122     177     155     182
     153
G1101050.!OJmm    121Qusp  1516FLA-05  13 /586             HL01.04.9002 1510 13 .
G1101050.!OJ/S20105_0.!OJ/_
     448     286     341     213     346     371     745     719     580     466
     487     353     279     323     422     436     351     238     135     172
     179     210     277     145     165     261     263     190     194     183
     127     110     144     189     135     154     217     110     115      99
     106     101     106     198     191     185     185     160     112     152
      93      95      83     176     165     193     139     101      93     113
      85     145     174     157     132     130      74      52     114     138
     174     132     144     125      83     124     118     127     150     189
     152     133     117      91     104      96      56      90     130     126
     103     163      92     103     174      99     117      85     123     116
     147     127     145     133     155     144     114     115     121     111
     174     113     112      89      99     130     111     104     164     110
     139
```

Appendix V

WinDENDRO

Format name	WinDENDRO
Other name(s)	
Type	Text file
Extension(s)	txt
Read/write support	Read only
Reference implementation	WinDENDRO
Data / metadata	Data with some structured metadata
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	Yes
Relative dating support	No
Multi series support	Yes
Original designer	Regent Instruments

V.1 Description

WinDENDRO format is a dendro text file format designed by Regent Instruments for their WinDENDRO software. Regent Instruments claims the format is proprietary. Although it is unclear whether such a claim is legally binding for a plain text file, the authors of DendroFileIOLib have decided to comply by not implementing a WinDENDRO format writer. However, in the interests of the dendro community and to ensure users can gain access to their data, DendroFileIOLib does include support for reading WinDENDRO format files.

WinDENDRO files differ from most other formats in that they contain a great deal of information specific to the image used to measure the sample. The WinDENDRO software allows users to measure ring widths from scans or photographs of samples rather than by using a traditional measuring platform.

WinDENDRO files are really just tab-delimited text files with data in columns in a specific order with a few additional header lines.

Line 1 should contain 8 tab-delimited fields

- Field 1 = WINDENDRO
- Field 2 = WinDENDRO file format version number, either 3 or 4
- Field 3 = Orientation of the data: R = in rows; C = in columns. All WinDENDRO files are in rows
- Field 4 = The column number where the data values begin. For version 3 files this is 13 and version 4 files this is 36
- Field 5 = The direction the data is recording in: P = pith to bark; B = bark to pith
- Field 6 = Whether the data is recorded incrementally (I) or cumulatively (C). WinDENDRO files are always incremental.

- Field 7 = Whether the bark width has been measured (Y or N). If yes, then there will be one more data value than there are rings
- Field 8 = RING

Line 2 contains the field names. For version 3 files these are:

- TreeName - The name of the tree being measured
- Path identification - ID of the path along which the series is measured
- Site identification - Name of the site from which the tree was taken
- YearLastRing - Year of the last ring in the series
- Sapwood - Distance (in mm) from the start of the sample to the start of the sapwood.
- Tree height - Height of tree in metres
- Tree age - Age of the tree. If unknown this should be 0, then it is assumed to be equal to the number of rings
- SectionHeight - Height up the tree in metres at which the sample was taken
- User variable - User defined variable - must be numerical
- RingCount - Number of rings the series contains
- DataType - Keyword indicating the type of data measured. This can be: RINGWIDTH; EARLYWIDTH; LATEWIDTH; EARLYWIDTH%; LATEWIDTH%; DENSITY; EARLYDENSITY; LATEDENSITY; MAX-DENSITY; MINDENSITY; RINGANGLE.
- OffsetToNext - The number of lines to skip to go to the next data line of the same type. For instance a file can contain earlywood and latewood data for multiple samples. If this is the case then each sample will have two rows, one for each variable, and the OffsetToNext field will be 1.

In additional to these fields, version 4 files also include the following:

- ImageName - The filename for the image used to do this analysis. If the image was taken directly from the scanner or camera then this field will be SCANNER
- Analysis Date Time - Date and time the measurements were initially saved to disk in format dd/m-m/YYYY HH:mm
- Acquisition Date Time - Date and time the image file was acquired in format dd/mm/YYYY HH:mm
- Modified Date Time - Date and time the file was last modified in format dd/mm/YYYY HH:mm
- ImageSize H V NBits Channel - The image size in pixels followed by bits per pixel per channel (8 or 16), channel used for analysis (Grey, RGB, R G or B)
- CalibMethod XCal YCal EditedDendro - Method of calibration: Intr (Intrinsic); Obj (ObjKnownDiam). This is followed by the size of a pixel and Y or N indicating if the image has been edited in WinDENDRO
- ScannerCamera Make Model Software - Details about the imaging hardware
- LensFocLength [35mm] - The 35mm equivalent focal length of the imaging lens
- PathBegX BegY EndX EndY Width - The coordinates for the start of the path/radius followed by the path width
- RingBoundary AutoMan Meth Precise - Details about the path taken. Ring boundary - Tg (tangent to ring) or Perp (perpendicular to path); Detection method - A (automatic) or M (manual); Ring detection method - Int (intensity differences) or T&S (teach and show); whether the 'more precise detection' method is active (Y) or not (N)
- EarlywoodDef - Earlywood-latewood transition criteria
- DensActive Media Calib - Density Analysis active (Y or N); Density Media setting (F - negative file or photo, W wood direct xray, positive film or photo); Light calibration setting (Acq - after image acquisition, Man - manual; No - none)
- DensNSteps MatDens Interpol - Number of steps and the density of the step wedge used for calibration followed by the interpolation method used between steps: Lin (Linear) Spl (Spline)
- DensStepsThick - The thickness of each step of the wedge used for density calibration
- DensStepsLightInt - The light intensity of each step of the wedge determined during the light intensity calibration
- DensStepsWoodDens - Equivalent wood density of each step of the wedge determined during light intensity calibration
- DiskArea - Area of the sample
- DiskPerim - Perimeter of the sample
- DiskAvgDiam - Average diameter of the sample

- DiskFormCoef - Sample area form coefficient
- CompWoodArea - Total area occupied by the compression areas
- VoidArea - Total area occupied by the void areas
- PathLength - Length of radius measured

Lines 3+ contain the actual data and metadata, one line for each series. Following the 13 or 36 columns of metadata (depending on file version) there are x number of columns containing ring values. The values are recorded as floating point data. The units for these data values are: mm for widths; % for percentages; g/cm^3 for densities; radians for angles.

Appendix W

License

The Corina server and desktop client are released under the GNU General Public License (GPL) version 3.

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

W.1 Terms and Conditions

0. Definitions.

   "This License" refers to version 3 of the GNU General Public License.

   "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

   "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

   To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

   A "covered work" means either the unmodified Program or a work based on the Program.

   To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

   To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

   An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

   The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

   A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

   The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

   You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

   Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

   When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

   You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

   (a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
   (b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
   (c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
   (d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

   A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

   You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

   (a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
   (b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for

software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

(c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

(d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

(e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

(a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

(b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

(c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

(d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

(e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

(f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

   You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

   You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

    Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

    An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

    You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

    A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

    A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

    Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

    In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

    If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

    If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

    A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

    If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

    If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

    Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

    THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

References

Aniol, R., 1983. Tree-ring analysis using CATRAS. Dendrochronologia 1, 45–53.

Baillie, M., Pilcher, J., 1973. A simple crossdating program for tree-ring research. Tree-ring Bulletin 33, 7–14.

Brewer, P., Murphy, D., Jansma, E., 2011. TRiCYCLE: a universal conversion tool for digital tree-ring data. Tree-Ring Research 67.

Brewer, P., Sturgeon, K., Madar, L., Manning, S., 2010. A new approach to dendrochronological data management. Dendrochronologia 28, 131–134.

Cook, E., Peters, K., 1981. The smoothing spline: A new approach to standardizing forest interior tree-ring width series for dendroclimatic studies. Tree-Ring Bulletin 41, 45–53.

Cormen, T., Leiserson, C., RL, R., C, S., 2001. Introduction to Algorithms. The MIT Press.

Huber, B., Fletcher, J., 1970. Dendrochronology in Europe.

Jansma, E., Brewer, P., Zandhuis, I., 2010. TRiDaS 1.1: The tree-ring data standard. Dendrochronologia 28, 99–130.

Knibbe, B., 2008. PAST4 - Personal Analysis System for Treering Research software version 4.5. SCIEM.

Reinsch, C., 1967. Smoothing by spline functions. Numerische Mathematik 10, 177–183.

Rinn, F., 2008. TSAP-Win software. Rinntech, Heidelberg.

Tyers, I., 1999. Dendro For Windows program guide. ARCUS Report 500.

Van Loan, C., 1999. Introduction to Scientific Computing: A Matrix-Vector Approach Using MATLAB. Prentice Hall.

Index