How to store and plot climate data

Tim Padfield, November 2008

Climate data should be stored as plain ascii text, with columns separated by spaces, tabs or commas. The native data format of data loggers is not to be trusted to endure for more than a year or two, and it prevents you sending useful data to a colleague who doesn't have that particular software. You should always export the data to plain ascii format. The date format is often taken from the set up of the exact computer the data logger is connected to. If someone else collects the data while you are on holiday, the date format can change in the middle of a sequence of climate data. So take care that the entire data format is consistent, and usefully labelled when seen from ten years in the future.

The first lines of the file should start with a # sign, to signify a comment. It should identify the place and time and significance of the data set.

Here is a sample data set. There are three comment lines, one with a general heading, the next two with identification of each column of data.

```
# RCAHMS Edinburgh, simulation run 14, Lars Aasbjerg 20080416
#Year   Month   Day     Hour    Outdoor°C       RelHumid% Corridor°C
#RelHumid-Corridor%
2005    1       1       1       1.00    83.96   6.46    65.39
2005    1       1       2       -0.20   88.03   6.30    65.29
2005    1       1       3       -1.00   92.90   6.19    65.13
2005    1       1       4       -1.30   94.07   6.13    64.73
2005    1       1       5       -1.40   94.27   6.09    64.31
2005    1       1       6       -1.40   94.27   6.05    63.93
```

Often, a data set will include surplus information, not useful in a historic record. Data can be manipulated in the text formatting program language 'awk'. This is one of the ancient utilities spawned in the infancy of computing, when unix reigned as the only serious operating system and Windows with mouse control was not yet born.

The following unix command isolates the date-time and the outside climate, reformatting the columns to space separated. It is a very simple instruction in the awk programming language which copies the first 6 data columns to a separate file:

```
awk '{print $1,$2,$3,$4,$5,$6}' edinb-climate.txt > edinb-outsideclimate.txt
```

The result of this conversion is:

```
#RCAHMS Edinburgh, simulation run 14, Lars
#Year Month Day Hour Outdoor°C
#RelHumid-Corridor%
2005 1 1 1 1.00 83.96
2005 1 1 2 -0.20 88.03
2005 1 1 3 -1.00 92.90
2005 1 1 4 -1.30 94.07
2005 1 1 5 -1.40 94.27
2005 1 1 6 -1.40 94.27
```

The data may need to be concatenated to assemble separate files from each visit to the data logger into a record for a whole year. Depending on the exact way the files are labelled, say edinb01.txt, edinb02.txt,... then this simple command will do the job:

```
cat ./edinb*.txt > edinb2008.txt
```

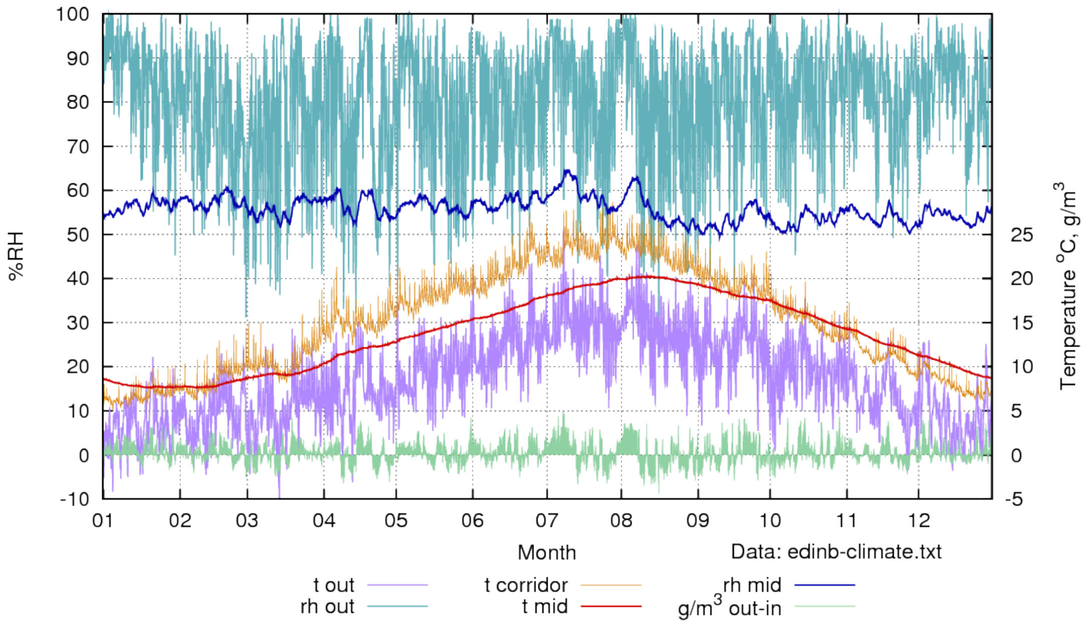
You may also want to combine data from several dataloggers for the same time period. For this there is a utility called 'join'.

So far you have not had to use a mouse or bring data into MS excel and you have typed maybe five lines of text instruction to the computer screen.

All these programs, and all the programs I mention below, will run on Windows, but they do not come installed as standard. I recommend running linux if you frequently manipulate climate data. These programs come as standard with linux distributions.

Plotting the data

All the graphs made by Tim Padfield for this course are constructed from the data files using 'gnuplot'. This is a non-graphical program for making graphs. First I show the result of running the script listed later.



This graph was generated by a gnuplot script and saved as an encapsulated postscript file. This gives the best control of appearance. To show this vector diagram as a bitmap for internet display, I have used the utility 'convert' (from the free 'imagemagick' program package) to make a .png file.

Here is the gnuplot script. It describes the layout so exactly that one could superimpose separate graphs from different loggers with perfect registration. Since the png graph is transparent, one can assemble elaborate overlays, using 'convert' or a bitmap editing program. Annotations on the graph can be made in gnuplot, or later on the eps or png image.

The script is carefully annotated so that its cryptic syntax should be intelligible. Most of the length of the script is comment. Note that at the very end of the script it plots a derived value, obtained by using data from the file, manipulated by a series of functions to give the difference between inside and outside water vapour concentration. In excel this would involve making a separate column, or several, to generate the calculated value. In gnuplot, the calculation is made on the fly as the data file is scanned. On an ordinary PC the graph building process takes less than half a second for a year's hourly data.

```
#Gnuplot script for Edinburg simulation nr 14
#data format: year month day hour 5:t_out 6:rh_out
# 7:t_corridor 8:rh_ditto
# 9:t_groundfloor 10:rh_ditto
# 11:t_mid 12:rh_ditto
# 13:t_top 14:rh_ditto
#tab separated
#example line: 2002 5 23 10 7.4 86.9 12.5 56.3...
#usage:
#command:>gnuplot
# > call 'thisfile.gnuplot' 'datafile' 'outfile.eps'
# q to exit gnuplot
#then convert eps graph to png:
# >convert -density 216 outfile.eps outfile.png
# (needs imagemagick installed)
#
#functions for dew point,vapour pressure and g/m3
svp(t) = 610*exp(t/(t+238.3)*17.2694)
vp(t,rh) = svp(t)*rh/100
w(t,rh) = log(vp(t,rh)/610.78)
dpt(t,rh) = (w(t,rh)*238.3)/(17.294-w(t,rh))
gm3(t,rh) = vp(t,rh)*2.166/(t + 273.16)
#
# postscript output gives richest layout possibilites
set term postscript eps enhanced color solid
# set output file to third item on command line
#after this gnuplot instruction file
set output "$1"
# adjust plot size to a wide rectangle
set size 5/5., 2.5/3.
set xlabel "Month"
set key below center
# indicate that the x axis is time
set xdata time
# set the time format as it is in the input data file
# default spacer is a space, so tabs need to be put
#explicitly in the format string as \t
set timefmt "%Y\t%m\t%d\t%H"
#x range must be in the same format, though hours can be omitted
set xrange ["2005\t01\t01":"2005\t12\t31"]
#The display format can be different, here it is just the month number
set format x "%m"
# no ticks between months, because they would be meaningless
set nomxtics
set yrange [-10:100]
set ytics -10,10,100
set nomytics
set ylabel "%RH"
# the right y axis is used for temperature and other things
set y2label "Temperature ^oC, g/m^3          "
# take care with y2 range, so its ticks coincide with the y1 ticks
set y2range [-5:50]
set y2tics -10,5,25
set grid
# to avoid certain confusion, name the data file on the graph
# the coordinates of the label are in graph coordinate format
# but can be off the graph
set label "Data: $0" at "2005\t09\t15", -22
# plot the data file listed on the command line immediately after
# the instruction file.
```

```
plot "$0"\
 using 1:5 axes x1y2 t 't out' with lines lt rgb "#b088ff" lw 1.5\
,'' using 1:6 t 'rh out' with lines lt rgb "#60b0ba" lw 1.5\
,'' using 1:7 axes x1y2 t 't corridor' with lines lt rgb "#d88000" lw 0.5\
,'' using 1:11 axes x1y2 t 't mid' with lines lt rgb "#d00000" lw 2\
,'' using 1:12 t 'rh mid' with lines lt rgb "#0000b0" lw 2\
,'' using 1:(gm3($$5,$$6) - gm3($$11,$$12)) axes x1y2 t 'g/m^3 out-in'\
with impulses lt rgb "#90d0a0"

# notes on the plot instruction:
# the \ at the end of a line continues the single line instruction.
# the repeated ,'' indicates that the same input data file is used
# for all traces.
# the double $$5 distinguishes the fifth data column from the
# fifth item on the command line.
# the line with the $$ calls the functions listed at the top,
#to generate a derived trace, not in the data file.
```

This script is undeniably complicated at first sight, but one soon builds up a set of scripts which can quickly be adapted to new projects. There is much tutorial help on the internet. Such scripts are well suited to automation, with many files being subjected in turn to the same script, by a super script. It is also possible to put into the script tests for deviation from a standard, for example. Gnuplot is a versatile, well established program. Like all the other programs mentioned, it is excellent, bug free, free to use and has no advertising budget.

All the programs mentioned can be downloaded free from the internet. Type 'download' plus the program name into google and follow instructions. For Linux, I recommend the Ubuntu and Suse distributions, both free and open source. Unix programs which are not available directly for MSWindows can be run using 'cygwin' for Windows. Some institutional IT administrators are unwilling to mix Windows and Linux computers on the network. It is a pity that Microsoft dominance of the desktop computer has blinded people to the merits of alternative programs.