Exercises for weather generator lectures, November 2014

Before you start

These exercises are all designed to be run from within R. You should have installed R and RStudio on your computers already, along with the necessary R packages — if not, follow the instructions in file `Rglimclim_Preparatory.pdf` at http://www.value-cost.eu/TS3. Having done this, everything you need for the weather generator sessions can be found in the `zip` archive `WeatherGenerators.zip` at the same web address. Download this archive, and unpack it somewhere sensible on your computer. It creates the following subdirectories:

`HANDOUTS` : contains copies of the handouts and slides for the weather generator training.

`PRACTICALS` : contains all of the material for the Rglimclim practical sessions, and for these exercises.

`PREPARATORY` : contains the preparatory material that you were sent before the workshop, updated to include the latest version of Rglimclim.

In the `PRACTICALS` directory, you will find a script called `WGExercises.r`. This script contains all the commands used in these exercises. Start up RStudio therefore, and use the `Session` menu to change the working directory to the `PRACTICALS/` directory you have just created. In the bottom right-hand subwindow, click on the file `WGExercises.r`. The script will open in the upper left-hand subwindow.

Exercise 1: simulation and properties of two-state Markov chains

The first exercise is designed to help you understand the properties of Markov chains, and in particular the notion of 'convergence to equilibrium'. The first few commands in the script define a function that simulates n values from a first-order, two-state Markov chain with transition probabilities p_{01} and p_{11}, starting from an initial value y_0. The states here are labelled '0' and '1', and can be thought of as representing 'dry' and 'wet' days respectively.

As supplied, the next few lines of code use this function to simulate 50 values from each of two separate Markov chains, both with transition probabilities $p_{01} = 0.2$ and $p_{11} = 0.6$. Informally you can interpret this as '20% of dry days are followed by a wet day; but 60% of wet days are followed by a wet day'. The only difference between the two simulations is that one starts with 0 and the other starts with 1. The code then calculates the cumulative proportion of 'wet days' in each simulation (i.e. for each time point $t = 1, 2, \ldots, 50$, the proportion of 'wet days' in the first t days is calculated. Finally, these cumulative proportions are plotted for the two chains starting from different points, along with the theoretical equilibrium proportion.

- Do the cumulative proportions of wet days appear to converge to the theoretical value as anticipated? If not, what happens if you increase n? Try 500, 5000, ... How long does it take to achieve convergence? (you could reasonably argue that it would be better to calculate the proportion of wet days for each chain using a moving 'window', so that you can see the convergence more clearly — feel free to adapt the code to do this!)
- Experiment with different values of the transition probabilities p_{01} and p_{11}, and look at the effect of varying them. How would you assign these transition probabilities in order to ensure that there was no temporal dependence in the Markov chain? How quickly do the chains converge in this case?

Exercise 2: properties of a simple weather generator

This exercise is designed to demonstrate that the properties of even a fairly simple weather generator can be extremely hard to deduce from its specification. The generator produces time series using the equation

$$Y_t = 3 + 3\cos\left[\frac{2\pi \times \text{day of year}}{365}\right] + \frac{1}{2}\sin\left[\frac{2\pi \times \text{day of year}}{365}\right] + \frac{3}{4}Y_{t-1} + \varepsilon_t \ ,$$

where ε_t has a standard normal distribution. This structure might provide a very crude approximation to the behaviour of a variable such as temperature, which has a strong seasonality and also day-to-day correlation (induced here by the term $3Y_{t-1}/4$ in the right-hand side of the equation).

Before you run the code, study the equation above and ask yourself:

- If you were to run a long simulation from this weather generator, roughly what would you expect to be the mean of your simulated values? (OK, you know from the lecture slide that the answer is 12: now, you have to figure out *why*!).
- Roughly what do you think will be the amplitude of the seasonal cycle in this weather generator?

Now read the code, make sure you understand it, and run it. It produces a 30-year simulation of daily values, plots the resulting time series and prints the overall mean. The mean should be very close to 12. How well did you guess the amplitude of the annual cycle?

As indicated on the lecture slide, for this particular model it *is* in fact possible to derive an expression for the amplitude of the annual cycle. The exercise is provided, however, to show that even simple-looking models can have surprising properties. For weather generators of realistic complexity, simulation is the only feasible way to proceed.

Exercise 3: extremes of dependent sequences

This final exercise is designed: first, to illustrate the fitting of Generalised Extreme Value distributions and use of return level plots; and second to show how dependence in a

sequence can change the extremal (and non-extremal!) properties. We will carry out two separate sets of simulations, for sequences based on different distributions.

Sequences based on normal distributions

The first set of simulations uses sequences of Gaussian variables, generated according to the equation

$$Y_t = \phi Y_{t-1} + \varepsilon_t$$

where $\varepsilon_t \sim N(0, 1)$ (the process (Y_t) is a first-order autoregressive process). The script contains a function to simulate realisations from this process, of arbitrary length, and with arbitrary values of ϕ. Run the lines:

```
sim.normalsequence <- function(n,phi,nstart) {
  if (abs(phi) >= 1) stop("phi must be between -1 and 1")
  n.all <- n+nstart
  e <- rnorm(n.all)
  y <- rep(NA,n.all)
  y[1] <- e[1]
  for (i in 2:n.all) {
    y[i] <- (phi*y[i-1]) + e[i]
  }
  y <- y[-(1:nstart)]
  y
}
```

R will not execute anything as a result of this: it just defines a new *function* called `sim.normalsequence()`, which you can use to simulate sequences from the model.

The next few lines show how to use this function to simulate '100 years of daily values' from the model when $\phi = 0$; to see the mean and variance of these simulated values; and to produce a quantile-quantile plot to check whether or not the simulated values have a normal distribution (obviously they *have*, because when $\phi = 0$ we have $Y_t = \varepsilon_t$ — so there is nothing surprising here). Next, a GEV distribution is fitted to 'annual maxima' and the estimated shape parameter is output, along with its standard error. The fit of the GEV is assessed using a return level plot. There are some 'coding tricks' here — ask if you don't understand them.

Run the code, and answer the following questions:

- For maxima of normally distributed variables, the GEV shape parameter should be zero in theory. Are your results consistent with this? How well does the GEV distribution fit the data?
- Next, change the value of ϕ and repeat the exercise (you can use any value between -1 and $+1$). Do your qualitative conclusions change? If so, how?

Sequences based on gamma distributions

When you have finished exploring the normal distribution, move on to the second set of simulations which uses gamma distributions instead. To see the connection between the two sets of simulations, note that under the normal model above the *conditional* distribution of Y_t given Y_{t-1} is itself normal, with mean $\mathbb{E}(Y)_t = \phi Y_{t-1}$ and variance 1. Applying a similar idea using gamma distributions, we might specify that:

- Conditional on Y_{t-1}, Y_t has a gamma distribution with mean $\mu_t = a + \phi Y_{t-1}$ and shape parameter 1.[1]
- a is equal to $1 - \phi$. Note that in this case, we have $\mu_t = 1 + \phi(Y_{t-1} - 1)$. So, given Y_{t-1}, we have $\mathbb{E}(Y_t - 1) = \phi(Y_{t-1} - 1)$; and if we define $Y_t^* = Y_t - 1$ then we have $\mathbb{E}(Y_t^*) = \phi Y_{t-1}^*$. In a sense therefore, this process is directly analogous to the Gaussian process above. It can also be shown that *unconditionally*, $\mathbb{E}(Y_t) = 1$ for each t in the new process (this is verified in the simulations below, by displaying the mean of the simulated values).

The remaining set of commands are almost identical to those for the Gaussian process: the only differences are (i) the use of the `fitdistr()` command (from the `MASS` library) to fit a gamma distribution to the simulated values, and the need to write our own code to produce a quantile-quantile plot for the gamma distribution.[2] Once again, for maxima of independent gamma variables the GEV shape parameter should be zero. Run the code with $\phi = 0$, and check (a) the quantile-quantile plot of the original observations against the fitted gamma distribution (b) the estimated shape parameter and its standard error (c) the fit of the GEV distribution to the simulated maxima according to the return level plot. Then repeat the exercise for different values of ϕ (which can take values between 0 and 1 in this case). Are the simulated values still gamma distributed? Does the GEV still fit the annual maxima? Is the estimated shape parameter still zero to within sampling error?

[1] You might think that we should set the mean to ϕY_t as before, but for the gamma distribution — which can only take non-negative values — this produces sequences that all converge to zero.

[2] The `fitdistr()` command will probably return some warning messages — these are not serious and can be ignored.