GLIMCLIM

Generalized Linear Modelling for daily climate time series

User Guide

Richard Chandler
Department of Statistical Science
University College London
Gower Street
London WC1E 6BT
ENGLAND

May 23, 2012

Contents

Licence Agreement

The copyright in this user manual and the accompanying GLIMCLIM software (together 'the Software') is owned by University College London of Gower Street, London, WC1E 6BT ('UCL'). By proceeding to use the Software you (an individual or any other legal entity) agree to be bound by the terms of this Agreement which will govern your use of the Software.

1. **Licence**

   1.1 You are permitted:

   (a) to load the Software into and use it on a single computer which is under your control;

   (b) to transfer the Software from one computer to another provided it is used on only one computer at any one time; and

   (c) to transfer the Software (complete with all its associated documentation) and the benefit of this Agreement to another person provided they have agreed to accept the terms of this Agreement and you contemporaneously transfer all copies of the Software you have made to that person or destroy all copies not transferred. If any transferee does not accept such terms then this Agreement shall automatically terminate. The transferor does not retain any rights under this Agreement in respect of the transferred Software.

   (d) to use the software for academic use only. By academic use it is meant that you can only use the software within an recognised academic institution and then only for the purposes of research and study. Any use of the software not in accordance with the previous sentence and also if used, whether directly or indirectly, for any commercial activity shall automatically terminate this Licence.

   1.2 You are not permitted:

   (a) to use or copy the Software other than as permitted by this Licence;

   (b) to load the Software on to a network server for the purposes of distribution to one or more other computer(s) on that network or to effect such distribution (such use requiring a separate licence);

   (c) except as expressly permitted by this Agreement and save to the extent and in the circumstances expressly required to be permitted by law, to rent, lease, sub-license, loan, copy, modify, adapt, merge, translate, reverse engineer, decompile, disassemble or create derivative works based on the whole or any part of the Software or its associated documentation or use, reproduce or deal in the Software or any part thereof in any way.

2. **Duration**

   This Agreement is effective until you terminate it by destroying the Software and its documentation together with all copies. It will also terminate if you fail to abide by its terms. Upon termination you agree to destroy all copies of the Software and its documentation including any Software stored on the hard disk of any computer or floppy disk or other removable media under your control.

3. **Exclusion of Warranties**

   3.1 You accept and acknowledge that this License does not set out any warranty in respect of the Software other than that save as expressly provided for in this Agreement and any condition or warranty implied by law as to the quality or fitness for purpose of the Software or as to any services provided hereunder in relation to the Software is hereby excluded to the fullest extent permitted by law. For the avoidance of doubt, UCL gives no warranty, in respect of:

   (a) Any failure of the Software to operate due to changes in the operating environment or in any operating system; or

   (b) Any failure of the functions provided by the Software to meet your requirements or to operate in combination with any hardware or other software which you may select for its use.

   3.2 You acknowledge and accept:

   (a) That the Software is still under development and will be for test and evaluation purposes only

   (b) That UCL has not produced the Software to meet your own specification;

   (c) That the Software cannot be tested in every possible combination and operating environment and that it is not possible to produce economically (if at all) computer programs known to be error free or which operate in an uninterrupted manner and that not all errors are necessarily capable of rectification.

   3.3 UCL shall not be liable to you for any indirect or consequential loss, damage or expense of any kind whatsoever arising out of or in connection with the Software whether arising in contract, tort, negligence, breach of statutory duty or otherwise.

   3.4 Subject always to clause 3.3, UCL's liability in contract, tort, negligence, breach of statutory duty or otherwise with respect to any claim arising in respect of its acts or omissions under or in connection with this Agreement shall be limited to the sums received by UCL at the date of the claim relating to such act or omission or UK £1,000,000 whichever is the lesser.

4. **Intellectual Property**

   All copyright, trade marks and other intellectual property rights subsisting in or used in connection with the Software (including but not limited to all images, animations, audio and other identifiable material relating to the Software) are and remain the sole property of UCL.

5. **Law**

   This Agreement shall be governed by English law.

1 Introduction

This manual accompanies a suite of FORTRAN programs that were originally written for the modelling and simulation of daily rainfall sequences, but may also be used for the modelling of other climatological variables. At present (September 2002), the programs fit

logistic and gamma regression models to sequences — the idea is that logistic regression is used to model zero/non-zero series, and gamma distributions are used to model skew distributions on non-zero days. At some point in the future, it's planned to make this a bit more general. However, this is a project that has grown beyond all original expectations, and it takes time to make changes, so please be patient!

For references describing the methods used in this software, see Wheater et al. (2000), Chandler and Wheater (2002) and Yan et al. (2002) for example. The Appendix to this manual gives some technical details, for the interested user.

A basic level of IT competence is assumed throughout — in particular, the ability to edit text files, create and move between directories and run commands from a prompt.

2 Obtaining the software

This software is regularly updated, in response to personal research requirements and to user requests. The latest version is always available via Internet from

http://www.homepages.ucl.ac.uk/∼ucakarc/work/rain_glm.html.

The software is provided as a zip archive which will unpack on both Unix and Windows systems. To install the software:

1. Create a directory for the software to live in (e.g. in Unix, type `mkdir GLM`).
2. Download the archive from the web address above (e.g. using Netscape), and save it to the directory just created. The archive is named rain_glm.zip. The web page contains hints on downloading, in case of problems at this stage.
3. Working in the directory containing the archive file, unzip the archive (Unix users: type `unzip rain_glm`, Windows users: you're on your own, but double-clicking the zip file in Windows Explorer is probably a good start!).
4. Examine the contents of the directory (Unix users type `ls -l`, Windows users via Windows Explorer). All being well, the directory now contains files README and rain_glm.zip, and subdirectories EXAMPLES, FITTING, SIMULATION, SOURCE and WINEXE. A brief description of each is now given.

   File README: Gives overview of the package.

   File rain_glm.zip: archive containing the distribution.

Subdirectory EXAMPLES/: Contains sample definition files and an artifical data file. See Section 7 below for more details.

Subdirectory FITTING/: Contains 2 subdirectories of its own: LOGISTIC/ and GAMMA/. LOGISTIC/ contains blank definition files necessary to fit a logistic regression model to zero/non-zero sequences, GAMMA/ contains files necessary to fit gamma distributions to positive values (zeroes will be discarded by this program if gamma distributions are fitted to rainfall sequences, for example).

Subdirectory SIMULATION/: Contains blank definition files required for simulation of fitted models.

Subdirectory SOURCE/: Contains source code (FORTRAN 77) which can be used to generate executables for fitting and simulation; also some simple Unix scripts which will compile the code on most Unix machines, and move the resulting executable to the appropriate place. There is an additional README file in this directory containing further details. Instructions on how to generate the executables may be found in the next section.

Subdirectory WINEXE/: Contains precompiled executables for use on machines running Windows 95/98/2000/NT, or any other version that has MS-DOS emulation.

As provided, the programs allow fitting of models with up to 80 parameters at up to 200 sites; there is no restriction on the number of cases to which models may be fitted (apart from disk space used to create scratch files). These limits may be changed by editing PARAMETER statements within the various source files.

The programs have been tested using the f77 compilers on the following systems:

- Silicon Graphics workstation under IRIX 6.3
- Sun workstation running SunOS 5.5.1
- IBM workstation running AIX 2.3
- PC running Linux (several versions)
- PC (MS/DOS with both Salford and GNU Fortran 77 compilers)

A list of known bugs and problems, with suggested workarounds, is given at the end of this manual.

3 Generating the executables

To use the software, it is necessary to generate the appropriate executables from the source code. For Windows users, precompiled executables are provided in the subdirectory WINEXE/. If the default storage is adequate, simply move these executables as required. To compile on a Unix system, proceed as follows:

1. Type `cd SOURCE/` to change to the SOURCE/ subdirectory. The README file in this subdirectory gives details of all the files which are present; these details are therefore omitted here.

2. There are two source files which may need editing before compilation, in order for the programs to run on your system. These are files fit_logi.f and fit_gamm.f. Both contain a parameter named BYTELN. This is used to define the length of records in temporary files used during model fitting. If the value is too small, the fitting programs will terminate with an error message; if too large, the programs will run but temporary files will be unnecessarily large, and disk space problems may result. The value should be set to the length of record required to store 1 byte of information on your system, and is compiler-dependent. Here are some values for installations which have been tried:

System	**Value of** BYTELN
Silicon Graphics (Irix 6.3)	2
Sun (SunOS 5.5.1)	8
IBM (Aix 2.3)	8
PC (Linux 2.0.35)	8
PC (MSDOS/Salford F77)	8

   The distribution comes with BYTELN set to 2. For systems which require a larger value, locate the following statement towards the top of files fit_logi.f and fit_gamm.f:

```
PARAMETER (BYTELN=2,MXNSITES=200,MXPARAMS=80)
```

   and change the value accordingly. For systems that are not listed above, it is recommended that the default value is kept, and increased in multiples of 2 until the program stops complaining when run (if the value is too small, a very clear error message is produced).

3. As provided, the programs allow fitting of models with up to 80 parameters at up to 200 sites; there is no restriction on the number of cases to which models may be fitted (apart from disk space used to create scratch files). These limits may be changed by editing PARAMETER statements within the various source files (eg. the one quoted above at the top of fit_logi.f and fit_gamm.f). To find all of the PARAMETER statements in the source files, type `grep PARAMETER *.f`. It is likely, however, that the defaults that come with the distribution should be adequate, at least to begin with[1].

4. Type `chmod 755 *compile`, to ensure that all compilation scripts are executable. These three scripts (fit_logi_compile, fit_gamm_compile and simrain_compile) compile the

[1] If the defaults are increased, note that internal storage requirements increase quadratically, both with number of sites and number of parameters. Keep an eye on swap space!

source code with various checks, and move the resulting executables into the appropriate directories. The only purpose of the compilation scripts is to obviate the need for typing a long list of source files at each compilation. The flags in the scripts work for most compilers: however, they may not be recognised by all systems, in which case some editing will be necessary — consult the f77 man pages, or better still a system administrator.

5. Run the compilation scripts (type `fit_logi_compile`, then `fit_gamm_compile` and finally `simrain_compile`). On successful completion, the directory ../FITTING/LOGISTIC/ contains an executable called fit_logi, ../FITTING/GAMMA/ contains fit_gamm, and ../SIMULATION/ contains simrain.

4 Using the package

When compiled successfully, the package consists of three executable programs: fit_logi, fit_gamm and simrain (with .exe suffices under Windows). Respectively, these are used to fit logistic and gamma regression models (which we may refer to as 'occurrence' and 'amounts' models), and to simulate a joint occurrence/amounts model. Windows users, note that the programs are designed to be run from within a command (previously MS-DOS) window[2] — for newer versions of Windows such as NT, one way to open such a window is to select Run the `Start Menu`, type `cmd` in the resulting dialogue box and click `OK`.

To run any of the programs under either Windows or Unix, change to the directory containing the executable and type its name. To fit a logistic regression model, for example, type

```
fit_logi
```

in the appropriate directory.

The package is designed to allow flexibility in fitting fairly complicated models to daily climate/weather data, and to avoid some of the tedious manipulation of data files which would be necessary if standard software packages were used to perform this kind of analysis. The package is written to take advantage of the fact that all exercises involving fitting, and simulating, GLMs for daily weather series must inevitably share common features, as follows:

- Models are to be fitted over a network of sites, for which various attributes (e.g location) are known. Flexibility in the placement and known attributes of sites can be achieved via the use of a database containing site information — this can be referenced as required by the fitting and simulation programs.
- Typically, possible covariates fall into a small number of categories, as follows:

[2]Double-clicking the executable from within Windows Explorer will work providing all required definition files (see below) are present in the directory containing the executable — however, if there is a problem, the window has a nasty habit of shutting down before you can read the error message!

- A constant term.
- Site effects.
- 'Year' effects (e.g. long-term trends).
- 'Month' effects (e.g. seasonality).
- 'Day' effects (e.g. day-to-day temporal autocorrelation).
- Interactions.

The software exploits this small number of categories, and treats each separately. In each category a variety of choices can be made regarding parametrisation by selecting from a 'menu' of choices. The software is, hopefully, written in a sufficiently modular way that users can customise these menus if they so desire.

'External' effects (such as ENSO or the North Atlantic Oscillation) are dealt with under the appropriate timescale — for example, if you wanted to use a monthly ENSO index as a covariate in your model, this would be counted as a 'monthly' effect; if you wanted to use a 'winter NAO' series (one value per year), it would count as a 'yearly' effect.

Climate datasets often have other unusual features relating to measurement methods. For example, in daily raingauge data any non-zero amount that is less than some small threshold may be recorded as a 'trace' amount, because it is too small to be measured accurately. Such features pose potential problems for statistical analysis. The software aims to provide methods for dealing with them.

To use the package, it is necessary to set up definition files which the system uses as input to define sites and models. Some of these files, especially those used for defining models, make extensive use of coding (to be detailed below). All of them must be prepared manually (templates are provided with the distribution), with a consequent risk of error. The first thing which each of the programs does is to read the definition files, flag any problems which are detected and output a meaningful summary of what has been defined in order that the user can check the input coding. Users are advised to check carefully at this stage to avoid wasted work! It is worth noting that one of the outputs from each of the model fitting programs is an updated model definition file which can subsequently be edited to make minor changes to a model and refit, or can be read directly by the simulation program. This feature takes most of the pain out of model definition.

The remainder of the section summarises the inputs to all of the programs. New users should study this section, in conjunction with the examples given below in section 7.

4.1 Input files

Each of the programs requires the relevant input files to be present, or it will terminate with an appropriate error message. The required inputs for each of the programs are summarised in Table 1.

Program	Purpose	Required input files
fit_logi	Fit logistic regression model for zero/non-zero	siteinfo.def, gaugvals.dat, logistic.def
fit_gamm	Fit gamma distributions to positive amounts	siteinfo.def, gaugvals.dat, gammamdl.def
simrain	Simulate daily values using combined occurrence and amounts models	regions.def, siteinfo.def, gaugvals.dat, logistic.def, gammamdl.def

Table 1: Input files required by each of the GLM fitting and simulation programs

All files suffixed .def are definition files; file gaugvals.dat is a data file. All of the definition files contain headers which explain the file structure. **These headers should not be altered!** Use the file headers, in conjunction with the tables in this section and the examples below in Section 7, as a guide when preparing definition files.

If you wish to define 'external' covariates to your models, data for these must be provided in separate input files. File yr_preds.dat is used to provide annual data, mn_preds.dat to provide monthly data and dy_preds.dat to provide daily data. Since models do not necessarily have to include external covariates, the presence of these files is not required for the programs to run. These files also contain self-explanatory headers which should be read but not altered.

There are two other files which may be required as input to the simulation program simrain. These are files cor_logi.dat and cor_gamm.dat. The former contains inter-site correlations for a latent Gaussian field that may be used to generate correlated rainfall occurrences at a network of sites; the latter contains observed correlations between Anscombe residuals from a gamma GLM. The files are required if the user wishes to simulate in such a way as to preserve the original correlation structure exactly. They can be generated automatically by the fitting programs fit_logi[3] and fit_gamm, so no further details need be given here.

The functions of the various mandatory input files in Table 1 are as follows:

regions.def: This file is used to define names of subareas in the area being studied. It is used by simrain when calculating monthly and annual summary statistics over different groups of sites. The intention is that simulation summaries over different subareas can be obtained.

[3]Note, however, that as output by fit_logi, the correlations in file cor_logi.dat are not guaranteed to produce a positive definite correlation structure: in this case it may be necessary to replace them with the fitted values from a plausible spatial correlation model fitted through them.

siteinfo.def: This file is used to define sites for model fitting and simulation. Any number of site attributes (e.g. Eastings, Northings, altitude) may be defined, so long as it does not exceed the maximum allowable number of parameters in a model (see discussion above in Section 2). Sites may be named, for easy identification in output files. In addition, a 4-character site identifier is defined for each site, along with the values of the various attributes which have been set.

logistic.def: Used to define a logistic regression model for zero/non-zero amounts. See tables below for more details.

gammamdl.def: Used to define a gamma regression model for positive amounts. The format is exactly the same as that for logistic.def. See tables below for more details.

gaugvals.dat: This must be an ASCII file containing daily values, and must be structured in the following way:

- Each line represents one day's value at one location.
- The file is sorted by date, and within that by site code.
- Each record takes the form

  `YYYYMMDD$$$$###.##`

  where

  `YYYY` is the year (e.g. `1978`).

  `MM` is the month (`1` – `12`).

  `DD` is the day.

  `$$$$` is the four-character site identifier, as defined in siteinfo.def.

  `###.##` is the value, to two decimal places.

  The FORTRAN format for reading each record is `I4,I2,I2,A4,F6.2`.
- Missing values are excluded from the file altogether.
- For sites which record values as 'trace' below a certain threshold, trace values should be entered as any strictly positive number less than the trace threshold (the trace threshold itself is defined to the system via the *model* definition files logistic.def and gammamdl.def).

Figure 1 gives a specimen extract from a valid gaugvals.dat file.

4.2 Defining subareas using file regions.def (for program simrain only)

The rainfall simulation program, simrain, gives the user the option to output monthly and annual summary information for each simulation. In some applications it may be useful to

```
198911 1  G3  2.30
198911 1  G5  8.50
198911 1 G10  9.70
198911 2  G3  3.10
198911 2  G5  7.10
198911 2 G10  5.30
198911 3  G3  3.50
198911 3  G5  5.40
198911 3 G10  5.70
198911 4  G3 11.20
198911 4 G10  9.00
198911 5  G3   .80
198911 5 G10   .00
198911 6  G3  2.90
198911 6  G5  3.00
198911 6 G10  3.70
```

Figure 1: Specimen extract from a valid gaugvals.dat file.

define particular subareas of interest (e.g. subcatchments within a large study region), and to calculate summaries for some or all of these subareas. The subareas can be defined to the system using the file regions.def.

The format of this file is explained in the file header, which occupies the first 19 lines of the file. This header is self-explanatory, and should not be altered. An example of a valid regions.def file is given in Figure 2.

4.3 Defining sites using file siteinfo.def (for all programs)

Sites may be defined, along with their attributes, using file siteinfo.def. This file is in three sections: the first is a header which occupies the first 37 lines of the file, and gives details of the required format (this header should not be altered). The second section is for defining site attributes, and is separated from the third section (which is for defining individual sites) by a row of asterisks containing the text END OF ATTRIBUTE DEFINITION. The formats for the second and third sections are discussed separately:

Attribute definition: Any number of site attributes (e.g. Eastings, Northings and altitude) may be defined to the system, up to a limit which, for programming convenience, has been set to the maximum number of parameters allowed in a model (the software comes with this limit set at 80). The attributes are defined in the second section of file siteinfo.def. The first line of this second section (line 38 of the file) must contain

```
REGION DEFINITION FILE
======================

        This file is used to define subareas to the rainfall simulation
program. Subareas should be numbered consecutively. After this header, each line
of the file defines one subarea, and looks something like:

NUMBER          TEXT

where NUMBER is a 2-digit subarea code, and TEXT is a name (up to 60 characters)
for the subarea being defined. These definitions will be used on program
output, and to check the specification of sites in the file siteinfo.def.
The rows are read using the FORTRAN format I5,A60.

The first subarea - number 0 - always represents the entire area and *must* be
present.

This header is 19 lines long.
------------------------------------------------------------------------
0    Ashdown Forest
1    Pooh and Piglet's side of the forest
2    Christopher Robin's side of the forest
```

Figure 2: Example of a valid regions.def file. The main study area is named, along with two subareas.

a single integer, which is the number of attributes being defined. For example, if Eastings, Northings and altitude are available for each site, this would be set to 3. There then follows a sequence of lines, one for each attribute, used to define text labels (maximum 70 characters each) for the attributes. These lines should be inserted *before* the END OF ATTRIBUTE DEFINITION line. **Note: it is recommended that you always define site $x-$ and $y-$ co-ordinates as the first two attributes**. There are good reasons for this — see Section 4.4 below.

Site definition: Sites are individually defined to the system in the third section of siteinfo.def, after the END OF ATTRIBUTE DEFINITION line. Each site is defined on two lines: the first contains a text string (maximum 70 characters) describing the site, and the second is in the form:

```
CODE  REGION  ATTRIB(1) ATTRIB(2)  ...  ATTRIB(N)
```

where

```
SITE DEFINITION FILE
====================

        This file is used to define site information. Excluding this header
there are 2 sections to the file: the first is used to define and label
site attributes (e.g. elevation, Eastings etc.), and the second is to
specify sites individually.

.
.       (header rows deleted for inclusion in manual, for space reasons)
.


This header is 37 lines long.
------------------------------------------------------------------------
2
Eastings (inches from left of 11" wide map)
Northings (inches from bottom of 8" high map)
**********************END OF ATTRIBUTE DEFINITION*************************
Pooh Bear's House
  G1    1       1.0       4.0
Where the Woozle Wasn't
  G2    1       2.5       1.5
Sandy pit where Roo plays
  G3    1       4.0       6.0
Rabbit's frends and raletions
  G4    2       6.5       5.0
Eeyore's Gloomy Place
  G5    2       9.5       1.0
Bee Tree
  G6    2       7.5       6.0
```

Figure 3: Example of a valid siteinfo.def file. Two attributes are defined, and their values specified for six named sites. The regions in this siteinfo.def file can be identified by referring to the subarea definition in the example regions.def file in Figure 2. For example, the site named 'Rabbit's frends and raletions' (code ' G4') has co-ordinates $(6.5, 5.0)$ and is in region 2. From Figure 2, this is 'Christopher Robin's side of the forest'. Some of the header rows are omitted here for reasons of space — when editing a siteinfo.def file, the header *must* be left intact, as supplied with the software distribution.

`CODE` is a 4-character string used to identify the site.

`REGION` is the number of the subarea (as defined in file regions.def) in which the site lies. It occupies positions 5–9 of the line, and is an optional field — a value of 0 or blank associates the site with no region.

The `ATTRIB`s are the values of the various attributes for the site, appearing in the same order as in the 'attribute definition' section of the file. Each value occupies the first 10 free positions of the line, starting with position 10 (i.e. the first attribute is in positions 10-19, the second in positions 20-29 and so on).

`N` is the total number of attributes defined.

The second line of each site's definition is read using the FORTRAN format A4,I5,N(F10.0). Note that this does *not* restrict you to integer-valued attributes — decimal values are read correctly using this format (at least, on all of the FORTRAN implementations that have been tested).

The structure of siteinfo.def is intended to allow completely flexible treatment of sites. Figure 3 is an example of a valid siteinfo.def file, which illustrates this flexibility.

4.4 Defining models using files logistic.def and gammamdl.def (for all programs)

The model definition files logistic.def and gammamdl.def have identical structures. They are used to define covariates for a model (along with their associated parameter values), and to select spatial dependence structures. For the model fitting programs, the parameter values in these files are treated as initial estimates for the numerical estimation algorithm.

The model definition files each contain a header which is 46 lines long, and contains details of the file structure. This header should not be altered. The line after this is reserved for future use. The next line is used to define a title, of up to 70 characters, for the model being defined. This will be printed at appropriate points in model output files, for subsequent identification of printouts.

The remainder of the file is used to define the model structure proper. Each row corresponds to a single parameter in the model, and contains five entries:

`COMPONENT VALUE CODE1 CODE2 CODE3 TEXT`.

The rows are read using the FORTRAN format `I5,F10.6,3I5,A40`. An explanation of the entries is as follows:

`COMPONENT` occupies the first 5 positions in the record, and is used to identify the type of quantity being defined. Valid entries are:

0 if the record relates to the constant term in the regression part of the model.

1 if the record relates to a site effect in the regression part of the model.

2 if the record relates to a 'year' effect in the regression part of the model.

3 if the record relates to a 'month' effect in the regression part of the model.

4 if the record relates to an 'day' effect, in the regression part of the model. This includes previous days' values, as well as any other covariate that varies on a daily timescale.

5 if the record relates to a 2-way interaction in the regression part of the model.

6 if the record relates to a 3-way interaction in the regression part of the model.

7 if the record relates to the nonlinear transformation of one of the covariates in the regression part of the model.

8 if the record relates to a global quantity such as a 'trace' threshold for raingauges. Such quantities are not strictly part of the model, but must be defined to the system somehow.

9 if the record relates to a dispersion parameter for the model (not used for the logistic model).

10 if the record relates to the spatial structure of the model.

The rows of a model definition file must be ordered according to the value of `COMPONENT`; if the rows are out of order, an error message will result.

`VALUE` is the value of the parameter being defined, occupying positions 6–15 of the record.

`CODE1`, `CODE2` and `CODE3` are used to define the precise details of the model to the system. Their interpretation depends on the value of `COMPONENT`. In general, it is not necesary to define all three codes. If they are all defined, `CODE1` occupies positions 16–20 of the record, `CODE2` occupies positions 21–25 and `CODE3` occupies positions 26–30. See Tables 2–8 for full details on coding.

`TEXT` contains descriptive text for this record, and appears after position 31. It is intended to help make the definition file readable by the user, and is not required or used by the program.

See Section 7 below for examples of model definition files.

4.4.1 Notes on Tables 2–8

1. In Table 3, covariates corresponding to a 'daily seasonal cycle' are calculated as though every year is a leap year. There is a tiny discontinuity between February 28th and March 1st in non-leap years.

Component	Code 1	Code 2	Code 3
0 (constant)	No codes used		
1 (site effects)	Number of attribute, according to order of definition in file siteinfo.def	If present, label of nonlinear transformation (see Table 4)	Not used
2 (year effects)	**Up to 50**: Label of trend function (see Table 4) **51 upwards**: $(x-50)$th variable defined in file yr_preds.dat (x being the code entered).	Optional selection of lagged values of external covariate (if Code 1 > 50).	Not used.
3 (month effects)	**1**: $\cos(2\pi \times \text{month}/12)$ **2**: $\sin(2\pi \times \text{month}/12)$ **3**: $\cos(2\pi \times \text{month}/6)$ **4**: $\sin(2\pi \times \text{month}/6)$ **11–22**: Individual month indicators (11 = Jan, 12 = Feb etc.) **51 upwards**: $(x-50)$th variable defined in file mn_preds.dat.	E.g. to use covariate value 2 years/months ago, set this field to 2. To use next year's/month's value set to -1.	Not used
4 (day effects)	**See Table 3, page 17**		
5 (2-way interactions)	Indices of interacting main effects (first site effect is 1)		Not used
6 (3-way interactions)	Indices of interacting main effects (first site effect is 1)		
7 (parameters in nonlinear transformations)	Index of covariate for which transformation is being defined. See note 3, page 17.	Parameter being defined (1, 2 or 3 — see Tables 4 and 6)	**0**: treat parameter as known **1**: find ML estimate of parameter
8 (global quantities)	**1**: Threshold for defining 'small' positive values. See note 4, page 17.	Method for dealing with such values (See Table 7)	Not used
9 (dispersion parameter)	No codes required. NB logistic models have no dispersion parameter. This field is ignored by model fitting programs.		
10 (spatial structure)	Label of spatial dependence structure used (see Table 8)	Number of parameter (see Table 8)	Not used

Table 2: Codes for specification of models in files logistic.def and gammamdl.def. To be used in conjunction with Tables 3–8.

Code 1	Code 2	Code 3
1–10: value x days ago **21**: $\cos(2\pi \times$ day of year$/366)$ (see note 1, page 15) **22**: $\sin(2\pi \times$ day of year$/366)$ **23**: $\cos(2\pi \times$ day of year$/183)$ **24**: $\sin(2\pi \times$ day of year$/183)$ **31–42**: Smooth month adjustments (31 = Jan, 32 = Feb etc.). See note 2 on page 17. **51 upwards**: $(x-50)$th variable defined in file dy_preds.dat.	Optional. If present and Code 1 $\leq$ 10, selects a transformation of previous days' values (see Tables 5 and 6). If Code 1 $>$ 50, selects lagged covariate values as in rows 2 and 3 of Table 2.	If present and equal to k, and Code 1 $\leq$ 10, cases with missing values at the same site for any of the previous k days are discarded by the fitting programs. If two records contain different values here, the highest is taken. The maximum allowable value is 10.

Table 3: Codes for specifying 'daily' effects in files logistic.def and gammamdl.def. This is row 4 of Table 2. See Tables 5 and 6 for further details on transformations and averaging.

2. In Table 3, the 'smooth month adjustments' are intended to allow departures from an overall seasonal cycle to be modelled smoothly, rather than via an indicator variable for a particular month (which results in a discontinuity in the fitted cycle). The adjustments are calculated from a shifted and scaled bisquare function

$$f(x) = \begin{cases} (1-x^2)^2 & |x| < 1 \\ 0 & \text{otherwise.} \end{cases}$$

   The scaling is chosen so that the adjustment takes its maximum value in the middle of the month, and is zero on the last day of the preceding month and on the first day of the following month.

3. In cases where the user wishes to define a covariate as a weighted average of previous values at all sites, any necessary parameters in the weighting schemes should be specified in the 'nonlinear parameters' section of the model definition file (see Tables 2, 5 and 6). Such parameters can be defined *only once* for each weighting scheme. Where more than one covariate is subject to the same weighting scheme (for example, if we wish to use weighted averages of values both one and two days ago), enter the index number of *any* of these covariates when defining the parameters. The software will automatically attach the correct weights to all other relevant covariates.

4. In row 8 of Table 2, there is an option to define a threshold below which positive-valued variables are simply regarded as 'small'. For non-negative variables such as rainfall and windspeed, the measurement of small values can be problematic. For data analysis

Component	Label	Function	Parameter 1	Parameter 2
1 (site effects)	1	Box-Cox power transform: $f(x) = \begin{cases} \ln x & \lambda = 0 \\ \frac{x^\lambda - 1}{\lambda} & \text{otherwise} \end{cases}$	λ	Not used
	2	Exponential transform: $f(x) = e^{ax}$	a	Not used
	3	Arctan transform: $f(x) = \arctan\left(\frac{x-a}{b}\right)$	a	b
	11–30	Fourier series representation of effect over the range (a, b). 11 and 12 are sine and cosine terms at the first Fourier frequency, 13 and 14 at second etc. *Odd* numbers correspond to *sine* terms (i.e. *odd* part of function). a and b can be specified *once only* for each site attribute. All sites must lie within the range $[a, b]$. Both a and b must always be treated as known.	a	b
	31–40	Legendre polynomial representation of effect over the range (a, b). 31 is linear, 32 is quadratic etc. a and b can be specified *once only* for each site attribute. All sites must lie within the range $[a, b]$. Both a and b must always be treated as known.	a	b
2 (year effects)	1	Linear: $f(t) = (t - 1950)/10$ (t is year)	No parameters required	
	2	Piecewise linear: $f(t) = \begin{cases} (t-a)/10 & \text{if } t > a \\ 0 & \text{otherwise} \end{cases}$	a	Not used
	3	Cyclical: $f(t) = -\cos\left(2\pi\frac{t-b}{a}\right)$	a	b

Table 4: Labels for nonlinear transformations of covariates (excluding previous days' values) in files logistic.def and gammamdl.def. This table should be used in conjunction with Table 2.

Label	Transformation
1	$\ln\left(Y_{t-k}^{(s)}\right)$
2	$\ln\left(1+Y_{t-k}^{(s)}\right)$
3	$I\left(Y_{t-k}^{(s)}>0\right)$ (i.e. indicator taking the value 1 if $Y_{t-k}^{(s)}$ was non-zero, 0 otherwise).
4	$I\left(0<Y_{t-k}^{(s)}<\tau\right)$, where τ is a 'trace' threshold (defined in row 8 of Table 2).
5	'Persistence' indicator: 1 if $Y_{t-1}^{(s)},\ldots,Y_{t-k}^{(s)}$ were all >0, 0 otherwise.
10–15	Transformations as above, but averaged over all sites with available data. Covariate is $S^{-1}\sum_r f\left(Y_{t-k}^{(r)}\right)$, where S is the number of contributing sites and $f(.)$ is the transformation. Code 10 is an average of untransformed values: $S^{-1}\sum_r Y_{t-k}^{(r)}$.
20–25	Transformations as above, but averaged over all sites with available data using weights that decrease exponentially with distance from the current site s. Covariate is $\sum_r w_{r,s} f\left(Y_{t-k}^{(r)}\right)$, where the weights $\{w_{r,s}\}$ sum to 1 and are proportional to $\exp\left[-ad_{r,s}\right]$. The value of a must be specified in the 'nonlinear parameters' section of the definition file — see Table 6.
30–35	Weighted averages of transformed values; weights proportional to $$\exp\left\{-a\left[(u_r-u_s-ku_0)^2+(v_r-v_s-kv_0)^2\right]^{1/2}\right\}.$$ See note 5, page 20 for an interpretation of this scheme.The values of a, u_0 and v_0 must be specified in the 'nonlinear parameters' section of the definition file — see Table 6.
110–115, 120–125 and 130–135	As 11–15, 21–25 and 31–35 but with the order of transformation and averaging reversed. Covariates are $f\left(\sum_r w_{r,s}Y_{t-k}^{(r)}\right)$ i.e. transformations of averages rather than averages of transformations.

Table 5: Labels for specifying nonlinear transformations of previous days' values, in files logistic.def and gammamdl.def. $Y_t^{(s)}$ denotes the value at site s on day t; u_s and v_s are the geographical co-ordinates of site s (in terms of the **first two** site attributes defined in file siteinfo.def); and d_{s_1,s_2} is the distance between sites s_1 and s_2, again calculated from these two site attributes. The expressions in the table relate to prediction of the value at site s on day t; k is the lag (in days), defined as described in Table 3.

purposes, perhaps the best way to deal with this problem is for the observer to set a threshold below which non-zero values cannot be measured accurately, and to record any non-zero values below this as 'trace' values. The analyst (and the GLIMCLIM software) can then treat these observations as censored data, and can adjust for the resulting uncertainty when fitting models.

Unfortunately, observational practice tends to be inconsistent. At one site, the observer may be extremely diligent with regard to the reporting of trace values, while at another the observer may simply record them as zeroes[4]. This can (and does) make fitted models appear to perform poorly on a site-by-site basis, because in its current form the GLIMCLIM software is not able to model these 'human effects'. A pragmatic solution to this problem is to set any small values to zero (effectively reducing all of the data to the level of the least diligent observer). The question then arises: what should be done with the values *above* the threshold? The software allows 2 possibilities: 'soft' and 'hard' thresholding (the terminology is borrowed from the literature on wavelets). See Table 7 for details.

Note that the software will allow *only one* of these methods of dealing with small values. This restriction has been deliberately imposed to avoid the temptation of building nonsensical models.

When simulating models for thresholded data (codes 2 and 3 in Table 7), the software converts the simulations back to the scale of the original variable where necessary. Specifically:

- For 'soft' thresholding, after simulation the threshold is added back to any non-zero values. Therefore the simulated output will contain no values between zero and the threshold.
- For 'hard' thresholding, no correction is made. In particular, positive values below the threshold are *not* set to zero, so that the simulated output may contain values between zero and the threshold. This may be seen as a problem. However, from a modelling perspective it is certainly the correct thing to do because, in this case, non-zero values are modelled on the scale of the original data. If the fitted model is reasonable, the problem will be negligible. Conversely, if the problem is non-negligible then the fitted model is unreasonable. If this is the case, 'soft' thresholding should be used instead.

5. In Table 6, weighting scheme 3:

$$w_{r,s} \propto \exp\left\{-a\left[(u_r - u_s - ku_0)^2 + (v_r - v_s - kv_0)^2\right]^{1/2}\right\}$$

allocates the greatest weight to a location displaced from the current site by a vector (ku_0, kv_0). It may be appropriate when movement of weather systems (at an average velocity of $(-u_0, -v_0)$ units per day) can be identified from the available data.

[4]In the UK this doesn't matter, because of the general policy of not making data available for analysis.

Weighting scheme	Parameters		
	1	**2**	**3**
1: Equal weights at all sites	No parameters required		
2: Distance-based exponential decay: $w_{r,s} \propto \exp[-ad_{r,s}]$	a	Not required	
3: Distance-based exponential decay with shift in origin: $w_{r,s} \propto \exp\left\{-a\left[(u_r - u_s - ku_0)^2 + (v_r - v_s - kv_0)^2\right]^{1/2}\right\}$	u_0	v_0	a

Table 6: Parameters in schemes for computing weighted averages of previous days' values. This table should be used in conjunction with Tables 3 and 5. $w_{r,s}$ is the weight associated with site r when predicting for site s. All other notation is the same as for Table 5.

Value of Code 2 (Table 2, row 8)	Treatment of 'small' values
1	Treat values as 'trace amounts'. This option is designed with rainfall in mind. Any 'small' value will be regarded as non-zero (and hence will count as a 'wet' day in a logistic regression model for rainfall, for example), but will be treated as a left-censored observation in any models for non-zero amounts.
2	'Soft' thresholding. If the original variable of interest is Y and the threshold is τ then models are fitted to Y^*, where $Y^* = 0$ if $Y < \tau$, $Y - \tau$ otherwise.
3	'Hard' thresholding. If the original variable of interest is Y and the threshold is τ then models are fitted to Y^*, where $Y^* = 0$ if $Y < \tau$, Y otherwise.

Table 7: Methods for dealing with 'small' positive values. The threshold below which a value is regarded as 'small' is defined as a 'global' quantity in the main model definition file (see row 8 of Table 2).

Table 8: Labels for specifying spatial structures in files logistic.def and gammamdl.def. For the occurrence model, the correlations in structures 1 through 20 are between latent Gaussian variables. For the amounts model, correlations are between Anscombe residuals at each site. For the correlation-based structures, d_{ij} denotes the Euclidean distance between sites i and j in terms of the first two site attributes defined in file siteinfo.def. This table should be used in conjunction with Table 2.

Label	**Model description**	**Parameter**			
		1	**2**	**3**	**4**
0 (default)	Independence	No parameters required			
1	Empirical correlations between each pair of sites	No parameters required (correlations read from file cor_logi.dat or cor_gamm.dat)			
2	Constant correlation between each pair of sites: $\rho(i,j) = \rho$	ρ	Not used		
3	Exponential correlation function: $\rho(i,j) = \exp[-\phi d_{ij}]$	ϕ	Not used		
4	Correlation function $\rho(i,j) = \alpha + (1-\alpha)\exp[-\phi d_{ij}]$	ϕ	α	Not used	
5	Powered exponential correlation function: $\rho(i,j) = \exp\left[-\phi d_{ij}^{\kappa}\right]$	ϕ	κ	Not used	
6	Correlation function $\rho(i,j) = \alpha + (1-\alpha)\exp\left[-\phi d_{ij}^{\kappa}\right]$	ϕ	κ	α	—
21	Conditional independence given weather state X, which is 0 for a 'dry' day and 1 for a 'wet' day. $P(X=1) = 1 - P(X=0) = \alpha$, the mean of the site probabilities predicted by the occurrence model. When $X = x$, the log odds for a non-zero value at site i is $\ln\left(\frac{p_i}{1-p_i}\right) + x\ln a - \ln b(\alpha, p_i)$, where p_i is the probability of rain at site i according to a logistic regression model, and $b(\alpha, p_i)$ is chosen to make the unconditional probability at the site equal to p_i. This structure is valid for the logistic model only.	$\ln a$	Not used		

Table 8: (continued).

Label	Model description	Parameter			
		1	2	3	4
22	Dependence induced by specifying a Beta-Binomial distribution for the number of wet sites on any day. The mean of the distribution, θ, is fixed at the mean of the individual site probabilities, and the shape parameter ϕ is estimated from data. This structure is valid for the occurrence model only.	ϕ	Parameters 2 and 3 are optional, and control program behaviour for days when the specified Beta-Binomial distribution is incompatible with the probabilities of rain at the sites. When this occurs, the probabilities are shrunk towards θ: p_i becomes $p_i - \lambda(p_i - \theta)$, for $\lambda \in (0, 1)$. The default value of λ is 0.01: to change this, enter it as parameter 2 for this model. By default, an error message will be printed to screen whenever such shrinking occurs. To suppress this, set parameter 3 equal to zero.		

6. In Table 8, inter-site dependence for both occurrence and amounts can be modelled via correlations. However, these correlations are defined indirectly. For the amounts models, they are the correlations between Anscombe residuals: these are defined (see Appendix D) in such a way as to have a distribution that is very close to normal. The rationale for this is that the multivariate normal distribution is the only multivariate distribution whose dependence structure is completely characterised by correlations.

   For rainfall occurrence, the correlation-based dependence structures are defined in terms of latent Gaussian variables: specifically, a standard normal random variable Z_i is associated with site i, and the occurrence of rain at that site corresponds to the event $Z_i > -\Phi^{-1}(p_i)$ where $\Phi(\cdot)$ is the standard normal distribution function and p_i is the modelled probability that site i is wet. Correlation between these latent variables therefore induces association between rainfall occurrence at the different sites. Unfortunately, estimation of these "latent" correlations is quite slow, as is imputation (i.e. simulation of 'missing' values conditioned on data that are available) using this particular dependence structure. Therefore, where possible it may be preferable to use either of dependence models 21 or 22 in Table 8. These are, however, designed for use in situations where there is not much variation of dependence with inter-site distance — in particular, model 22 is designed for use in catchments that are small relative to the typical scale of weather systems so that typically sites are either mostly wet or mostly dry. For full details of these schemes, see Appendix E.3.

7. In Table 8, many of the correlation-based structures (codes 3 through 20) are fitted by minimising a weighted least-squares criterion using an iterative numerical minimisation scheme (see Appendix E.2). The success of such fits can be dependent on a good choice

of starting values in the model definition files. However, if the user enters a starting value of zero for any of the parameters in these correlation-based structures, then the software will set its own starting value using some potentially sensible heuristics.

4.5 Defining external effects using files yr_preds.dat, mn_preds.dat and dy_preds.dat (for all programs)

In many applications, it is of interest to examine the effects of 'external' covariates upon climate/weather variables. By 'external', we mean non-deterministic time-varying quantities (as distinct from trends, which can be regarded as deterministic) other than the variable under consideration. Examples include ENSO, sea surface temperature series and the North Atlantic Oscillation.

We classify such external covariates according to the timescale of available data (i.e. whether we have annual, monthly or daily time series). To simplify the structure of the input files, a separate file is used for each timescale. These files are only required to be present if the user requests external covariates at the corresponding timescales.

The general procedure for defining external covariates using these files is very similar to that for defining sites in file siteinfo.def (see Section 4.3 above). The files have a similar structure of a header followed by 2 sections. The first section defines the variables for which data are being provided, and the second gives the data values themselves.

An example of a valid dy_preds.dat file is given in Figure 4. The structure of the other two files is very similar, so examples are not given. As in siteinfo.def, the first line after the header contains a single integer giving the number of variables whose data may be found in the file. The names of these variables are given on subsequent lines (these names will be used for labelling model output). Following this, each line of the 'data input' section of the file contains data at a single time point for all of the variables. The values of the first variable occupy positions 10–19 of a record; those of the second occupy positions 20–29 and so on. The formats of the dates are slightly different for the 3 different timescales, as follows:

- File dy_preds.dat: enter dates as `YYYYMMDD`, in the first 8 positions of each record. Each record is read using the FORTRAN format `I4,2I2,1X,50(F10.0)`. As in siteinfo.def, decimal values appear to be read correctly using this format.
- File mn_preds.dat: enter dates as `YYYY DD` ('Year' occupies positions 1–4, and 'Month' occupies positions 6–7). Each record is read using the format `I4,1X,I2,T10,50(F10.0)`.
- File yr_preds.dat: enter dates as `YYYY` in positions 1–4. Each record is read using the format `I4,T10,50(F10.0)`.

In each of these files, data should be provided in chronological order (otherwise, the programs may not be able to find all of the required covariates). 'Missing' values should be coded as -9999.9. In model fitting, cases with missing values of any covariates will be

```
EXTERNAL COVARIATES - DAILY DATA FILE
=====================================

        This file is used to define 'external' covariates, for which
daily data are available, to a Generalized Linear Model for some
climate/weather variable. By 'external', we mean non-deterministic
time-varying quantities (i.e. not trends) other than the variable under
consideration. Excluding this header, there are 2 sections to the file.
The first is used to define and label the covariates, and the second to
provide the data.

.
.       (header rows deleted for inclusion in manual, for space reasons)
.

This header is 41 lines long.
------------------------------------------------------------------------
2
Central England Temperature (degrees C)
Monday indicator
************************END OF COVARIATE DEFINITION*******************
20000701    -9999.9         0
20000702    -9999.9         0
20000703    -9999.9         1
20000704    -9999.9         0

   .           .           .
   .           .           .
   .           .           .
```

Figure 4: Example of a valid dy_pred.dat file. Two variables are defined, and their data are entered for the time period over which they are available. Missing values are coded as -9999.9. Note that some of the header rows are omitted here for reasons of space — when editing external data files, the headers *must* be left intact, as supplied with the software distribution.

Program	Outputs always produced	Optional outputs
fit_logi	logistic.res, logistic.de2	res_logi.dat, cor_logi.dat
fit_gamm	gammamdl.res, gammamdl.de2	res_gamm.dat, cor_gamm.dat
simrain	mdlspec.txt, plus monthly.dat and/or daily.dat	daily.dat, monthly.dat

Table 9: Output files generated by each of the GLM fitting and simulation programs

discarded from an analysis (so, if your daily time series of the climatological response variable covers a different period to that of your external covariates, models will be fitted only to that period for which the records overlap). **NB:** Simulation programs will halt with an error message if they are required to simulate over any period for which external covariate data are missing.

5 Summary of outputs

Output files: When each of the programs is run, it produces various output files depending on options selected by the user. If the files already exist, the user will be prompted for confirmation that they can be overwritten. The outputs are summarised in Table 9. The purpose of each of the output files is as follows:

logistic.res, gammamdl.res: these are the main results files produced by the model fitting programs fit_logi and fit_gamm respectively. They each give the specification of the fitted model, together with initial and final parameter estimates, number of cases used in the fitting, maximised log-likelihood[5], deviance (equivalent to the residual sum of squares in a linear regression model), largest standardised score (which should be close to zero near a maximum), number of parameters estimated, residual degrees of freedom,

[5]**Note:** For the gamma distribution, the software uses the maximum likelihood estimate of the dispersion parameter to calculate the likelihood (in order that different models can be compared using likelihood ratio tests — see Appendix B.1). However, for subsequent application it is normally recommended that a method of moments estimator is used instead, since the ML estimate is sensitive to rounding errors in small observations (McCullagh and Nelder, 1989). The software reports both estimates, and normally writes the moment estimate to the `.de2` file for subsequent use. The exception to this is when trace values are being replaced by their approximate conditional expectations (see Table 7), in which case the ML estimate is written to file. The reason is that here, the fitted values depend on the estimated dispersion parameter, and it is necessary to use the ML estimate to compare nested models.

parameter standard errors, and likelihood ratio statistic for comparing the initial and final fits. Results of residual analyses also appear in these files, if requested.

logistic.de2, gammamdl.de2: these files are produced by the model fitting programs, and are updated definition files for the fitted models. The idea is that the user should not have to retype all of the parameters into a new definition file every time a new model is fitted. These files can be edited very quickly to expand a model by adding an extra covariate, or can be renamed and used directly as input to the simulation program. To assist the user, the number of each main effect is appended to the descriptive text in this files — this helps when defining interactions for example.

res_logi.dat,res_gamm.dat: these files are optionally produced by the model fitting programs, and contain residual information for each case in the fitting database so that the user can carry out further residual analyses if desired. Both files have the same format, with 7 columns and a header row. The columns are SITE, YEAR, MONTH, DAY, OBSERVED, PREDICTED and ETA. Most of these are self-explanatory. In file res_logi.dat, the OBSERVED column contains 1 for a wet day and 0 for a dry day; the PREDICTED column is the fitted probability of a non-zero amount for the day. In file fit_gamm.dat, the OBSERVED column contains the observed value in mm (with trace values replaced by their approximate conditional expectation under the fitted model, as described by Chandler and Wheater (1998)), and the PREDICTED column is the mean of the forecast gamma distribution for the day. In both files, the ETA column contains the value of the fitted linear predictor (i.e. $\ln[p/(1-p)]$ for the occurrence model and $\ln \mu$ for the amounts model). Records from these files can be read using the FORTRAN format `A4,1X,I4,T14,I2,T17,I3,T21,F8.4,T31,F8.4,T41,F8.4`.

cor_logi.dat, cor_gamm.dat: these files are produced by the model fitting programs when the user chooses to incorporate inter-site dependence via correlations (options 1–20 in Table 8). For rainfall occurrence (cor_logi.dat), the correlations are between latent Gaussian variables at each site; for rainfall amounts, they are between Anscombe residuals. The files can be used directly as input to the simulation program simrain, if required (although for a caveat, see the footnote on page 9). They contain a single header row, and then a record for each pair of sites. Each record has six entries: 4-character site codes (as defined in siteinfo.def) for each site, then the observed correlation, the number of pairs of residuals used in the calculation and finally the site separation in terms of the first two attributes from file siteinfo.def (the assumption is that these attributes represent geographical coordinates; if they are not defined, the site separations are simply output as zero). The FORTRAN format for reading each record is `A4,4X,A4,4X,F7.4,1X,I5`. Correlations that could not be computed due to a lack of available observations are coded as -9.999.

mdlspec.txt: this file is produced by the simulation program simrain, and contains summary information about a simulation run, including: specification of models used in the run, details of all sites and subareas defined for simulation, a summary of simulation options chosen, and the seed used for random number generation. The information in this file

can be used to reproduce any simulation exactly, providing a non-zero random number seed was used (see the example in Section 7.3).

monthly.dat: this file is optionally produced by the simulation program simrain and contains monthly and annual summaries of simulation runs. Summaries may be present for the whole area and additionally for any or all of the subareas defined in file regions.def. Each row of the file contains 16 entries: simulation number, year, region code (from regions.def), 12 monthly totals and 1 annual total. These totals are averaged over all sites in a region. The FORTRAN format for reading records from this file is `I3,1X,I4,1X,I3,1X,12(F6.1,1X),F6.1`.

daily.dat: this file is optionally produced by the simulation program simrain and contains detailed daily results of simulation runs. Each record of the file contains $2S+4$ entries, where S is the number of sites. The first four entries are simulation number, year, month and day: the remainder are pairs for each site (sites appear in the order listed in file siteinfo.def). The first of every pair is the daily amount, the second is a flag taking the value 0 if the value was observed at the site, 1 if the value is sampled randomly from the model. The FORTRAN format for reading records from this file is `I3,1X,I4,1X,2(I2,1X),S(F6.2,1X,I1,1X)`, where `S` is the number of sites.

6 Hints on use

This section contains a few guidelines which may be useful for avoiding error messages and meaningless models.

1. When fitting models, start with a very basic model (e.g. no covariates except a constant) and gradually increase its complexity, adding a couple of covariates at a time. At each stage, use the .de2 file from the previous fitted model as a basis for the new .def file. When adding new covariates whose coefficients are unknown, set them to zero in the new .def file.

   There are two reasons for this recommendation:

   (a) The chances of error in defining a model are reduced — the user only has to make small changes to a model definition file at each stage.

   (b) The approach should ensure computational stability, by providing reasonable starting values for fitting each model.

2. When building up a model, add 'obvious' covariates (for example those representing seasonality and temporal dependence) first. There is little doubt that such covariates should appear in a model (although it may be necessary to compare different representations of temporal dependence, for example), and it makes sense to start by getting close to a reasonable model quickly. Moreover, such a strategy makes it unlikely that any covariates added early on will subsequently need to be dropped.

3. For computational stability, covariate values should not be too large. For example, it might be necessary to express site altitude in hundreds of metres rather than in metres. As a rule of thumb, choose units of measurement so that covariate values tend to be between 0.1 and 10 in magnitude, if possible.

4. Keep track of the number of covariates involving regional effects (including interactions). If this number approaches the number of sites from which data are available, there is a risk that the model may be overfitted to these individual sites and may not be reproducable at other locations. One warning sign is the presence of extremely large and uninterpretable coefficients (usually, but not necessarily, relating to site effects) in a fitted model involving large numbers of orthogonal series components to represent site effects. This is likely to be the result of one or two sites which do not fit the general pattern, and may be an indication that data from these sites are suspect.

   As a general strategy for defining site effects, it may be useful to fit a model containing *no* site effects, and to plot a map showing the magnitudes of mean residuals at each site. A bubblemap is recommended (see Section 7.2.6 below) since, in contrast to other techniques such as contour mapping, this does not artifically smooth the residuals. If there is clear regional structure in this map that can be related to, say, site altitude, then clearly altitude should be included as a covariate. If there is other clear systematic structure, then the map can be used to guide the choice of orthogonal series representation. For example, if residuals are positive in the north of a study area and negative in the south, it will probably be approriate to include Legendre polynomials for site northings. If there is *no* systematic structure in the map, there is little point trying to include regional effects in a model! In this case, if many sites have large mean residuals there may be some data quality problems; it may be worth working with thresholded data (see row 8 of Table 2) to try and overcome these.

5. When estimating parameters in nonlinear transformations of covariates, it is worth proceeding in several stages. First, fix the unknown parameters at some 'plausible' level and estimate the optimal regression coefficients (βs) for that level. Next, free up a single parameter and estimate that together with the βs, holding the remaining nonlinear parameters fixed; carry on freeing up more parameters gradually. It may be necessary at some stage to fix some parameters which had previously been freed because, despite the sophistication of the algorithm used, it can sometimes be extremely slow. In some cases it may be clear that, for all practical purposes, the algorithm has converged (keep an eye on the log-likelihood at each iteration, to determine this) but many iterations may be spent making rather small changes. If this occurs, it may be worth fixing all nonlinear parameters at their approximately optimal values, and running a final fit to obtain the correct βs for these values. If standard errors are required, they can be obtained by freeing up the parameters and running a further 'fit' with zero iterations.

   Convergence difficulties may also indicate a silly model. Please try and resist the temptation to shoot the programmer until you are sure your model is reasonable! Avoid models that are too complex to be supported by the available data. For example, it is

unlikely that weather system movement can be detected in daily data over small areas, in which case weighting scheme 3 in Table 6 should not be used.

6. Log-likelihoods (or deviances) can only be used to compare models that have been fitted to the same dataset. This is particularly relevant when comparing models that have different numbers of 'autoregressive' terms. Typically, missing observations will mean that a model involving, say, 2 previous days' values can be fitted to a larger subset of data than a model involving 3 previous days' values. The solution, in this case, is to fit the 2-day model using just those observations for which a 3-day model can be fitted (using Code 3 in Table 3), and compare log-likelihoods based on this common subset of observations.

7. In an ideal world, likelihood ratio tests or deviance comparisons are preferable to t-tests (i.e. the comparison of an estimate with its standard error) for determining the statistical significance of terms in a model. The reason is that likelihood ratios automatically adjust for correlations among the covariates. However, any model comparison can only be made if the underlying calculations are correct. In particular, if models are fitted to a network of sites then inter-site dependence will usually invalidate the 'naive' versions. The software will adjust both standard errors and likelihood ratio statistics, providing a spatial dependence structure is defined to the system (code 10 in Table 2). Note, however, that no attempt is made to adjust the log-likelihoods and deviances themselves. The theory underlying the adjustments is given in Appendix C.2 — note that the adjustments are independent of the particular spatial structure used in the model.

8. As far as 'autocorrelation' effects are concerned, averaging previous days' values over several sites is likely to lead to better performance than considering each site's history separately[6]. However, the computational load is dramatically increased by averaging over sites, particularly if parameters in weighting schemes have to be estimated. Also, there is a possibility of bias when averaging over previous days for which some sites have missing data (the software computes averages over all sites with available data, and only designates the resulting covariate as 'missing' if there is missing data at the single site of interest). This is particularly true of averages computed using the 'distance-based exponential decay with shift in origin' weighting scheme (Table 6) where edge effects may be present near the boundary of a study region. Such effects may manifest themselves via a change in the variance of residuals at boundary sites.

9. When simulating models that involve averages of previous days' values, choose the sites for simulation carefully. If simulation is carried out for a sparse subset of the sites originally used for model fitting, averages computed over this subset may have rather different properties from weighted averages computed over the entire network. In such cases, simulation results will be biased.

[6]From a physical point of view, day-to-day dependence is dictated by the movement of weather systems, which affect whole areas rather than single sites. From a mathematical perspective, averaging over previous days can go some way towards alleviating the problems caused by inter-site dependence.

10. When choosing a model for the inter-site dependence in rainfall occurrence, bear in mind that each of the available options is designed primarily for use in a specific situation. For example, the beta-binomial coverage scheme (option 22 in Table 8) is designed for use in regions that are small relative to the synoptic scale so that sites tend to be mostly wet or mostly dry: this scheme does not explicitly represent the decay of correlation with inter-site separation, because in small regions this decay is rather small and it is hydrologically more relevant to capture the high frequency of simultaneous occurrence. Such a scheme is not appropriate for use in larger catchments, however, where the distance dependence is more obvious: in this case, one of the correlation-based schemes (which are slow and inaccurate for small regions where the correlations are very high) should be used instead.

11. For the correlation-based inter-site dependence structures (options 3–20 in Table 8), the software provides no goodness-of-fit measures. To check the results therefore, it is always worth making a plot of the empirical correlations against inter-site distance (this can be done using the output in files cor_logi.dat or cor_gamm.dat), and overlaying a graph of the fitted correlation function. For example, suppose that a powered exponential correlation function (option 5 in Table 8) has been fitted to the Anscombe residuals from a rainfall amounts model and that the software has returned estimates $\hat{\phi} = 0.4321$, $\hat{\kappa} = 0.5678$, along with the corresponding cor_gamm.dat file containing the individual inter-site correlations from which these estimates were derived. In the R software environment (R Development Core Team, 2008), the required plot can be generated using the following code:

```
obs.corr <- read.table("cor_gamm.dat",header=TRUE) # Individual correlations
d <- sqrt(obs.corr$Xsep^2 + obs.corr$Ysep^2)       # Inter-site distances
plot(d,obs.corr$Corr,pch=20,xlab="Distance",       # } Make plot
                ylab="Correlation",ylim=c(0,1))    # }
phi <- 0.4321                                      # } Parameter estimates
kappa <- 0.5678                                    # }
d.grid <- seq(0,max(d),length.out=100)             # Grid of distances
rho <- exp(-phi*(d.grid^kappa))                    # Fitted correlations
lines(d.grid,rho,col="blue",lwd=2)                 # Add to plot
```

A plot such as this can also be useful to check the adequacy of the parameter estimates produced by the iterative minimisation scheme. As discussed previously, for many of the correlation structures in Table 8 the parameters are estimated via numerical minimisation of a weighted least-squares objective function. If the minimisation scheme fails to converge to within the required tolerance, the software issues a warning message[7]: the extent of the problem can be assessed by examining the largest element of the objective function standardised gradient vector (which is included in the warning message). This should be close to zero (the criterion for convergence in the software is that it should be less than 10^{-5}): it can be interpreted as an indication of how far

[7]**Note:** the warning message is printed to screen *before* the final parameter estimates: scroll up to check for warnings therefore.

the parameter estimates are from the true minimum of the objective function. Even if the standardised gradient is relatively large however, the most important test is whether the fitted function provides a reasonable summary of the observed structure: a graphical assessment may reveal that the fit is adequate for practical purposes.

7 Examples

In this section, we work through a simple example to illustrate the use of the software. The data, and necessary definition files, can be found in the EXAMPLES/ subdirectory of the distribution. This is an artificial example relating to daily rainfall over part of Ashdown Forest in Sussex, England. A map of the area can be found in Milne (1958). The data were actually generated by simulating a GLM fitted elsewhere, with appropriate modifications. These simulated data exhibit many typical features of rainfall sequences in northwestern Europe. To make things more realistic, rainfall amounts less than 0.1mm have been set to 'trace' values and appear in the files as values of 0.05mm. Moreover, approximately 20% of the values are missing (at random).

It is recommended that you create a separate directory to work through this example — this way, if something goes wrong then you'll still have a clean version of everything in the EXAMPLES/ subdirectory of the software distribution. You need to copy the files `gammamdl.def`, `logistic.def`, `siteinfo.def`, `gaugvals.dat` and `regions.def` from EXAMPLES/ to your new directory. You also need to compile the executables for your system (see Section 3 above) and copy/move these to your new directory (Windows users: just copy the precompiled .exe files from the WINEXE/ subdirectory).

Next: change to the new directory (Windows users: you should work within a command window, *not* by clicking on the executables — see Section 4, page 7), and you're ready to start.

7.1 Setting up definition files

The first stage in any analysis is to generate a data file (`gaugvals.dat`) in the correct format, and to define your sites to the system via file `siteinfo.def`. This has already been done for you here — open both of these files in a text editor (Windows users — *not* Notepad, it's incapable of reading large files!), and compare with the descriptions in Section 4.1 above. Note the following in particular:

- The link between these two files is provided via the 4-character site identification codes '` G1`',' `G2`' etc.. Justification is important — the code for site G1 is 'space space G 1', and the software requires an exact match. So for example '` G1`' in one file and '` G1 `' in the other would be interpreted as different sites[8].

[8] A useful trick, if at any stage you want to fit or simulate a model at a subset of sites, is to make small

- The first (and, in fact, only) two site attributes defined are the site X and Y co-ordinates. This follows the recommendations given in Section 4.4 above, and relates to the fact that the software will use these first two site attributes to compute inter-site distances if necessary.
- The data file `gaugvals.dat` is ordered by date and site, with missing observations excluded. For example, at the beginning of the file all 6 sites have data for 1st January 1970, but site 2 is missing for 2nd January.

7.2 Model fitting

7.2.1 Trivial logistic regression model

Now that the database has been defined, we can start to fit models. Let's start with the simplest possible logistic regression model for rainfall occurrence. Let p_i be the probability of non-zero rainfall for the ith case in the dataset. The model we will fit is

$$\ln\left(\frac{p_i}{1-p_i}\right) = \beta_0 \qquad \text{or equivalently} \qquad p_i = \frac{\exp\left[\beta_0\right]}{1+\exp\left[\beta_0\right]} \ . \tag{1}$$

This is a GLM in which the single covariate is a constant term. To fit it, carry out the following steps:

1. Use a text editor to edit the file `logistic.def`. You may like to consult Table 2 as a check, while doing this. Make the following changes to the template supplied with the distribution:
   - Locate the model definition section at the bottom of the file. In the template, various covariates have been defined. We don't need them at the moment, so delete everything exept the 'Constant' row (i.e. delete the last 5 rows of the file). Notice that this is line 49 of the file (if your editor does not display line numbers, throw it away and use a decent one!). **Note:** the file must *not* contain any empty lines after the model definition. In most editors (Emacs is the only exception I'm aware of), you can check this by trying to scroll as far down the file as possible — if the cursor will move below the 'Constant' row, then there are empty lines that should be deleted (use the `Backspace` key).
   - To ensure that you can identify the output from this fit subsequently, give your model a meaningful title. This goes in what is now the penultimate line of the file (in the template, this line contains the text 'EXAMPLE DEFINITION FILE FOR LOGISTIC REGRESSION MODEL'). Let's call the model 'TRIVIAL LOGISTIC REGRESSION EXAMPLE' (it doesn't have to be in upper case letters, but they make the title easier to spot in the output).

changes to the codes in `siteinfo.def`. For example, if you want to omit site `G3` from the analysis, change its code in `siteinfo.def` to ' `?G3`'. The software will no longer recognise the data from this site in file `gaugvals.dat`, so it will be omitted from the analysis.

Once you've made these changes, save the definition file and quit the text editor.

2. Run the fitting program, by typing `fit_logi` at the prompt. All being well, you should see the following:

```
 Opening files ...
 Reading site information ...
          Data from  6 sites will be used in the fitting.
 Reading model specification ...
TRIVIAL LOGISTIC REGRESSION EXAMPLE
==================================
Initial parameter estimates:

Main effect:                                              Coefficient
------------                                              -----------
Constant                                                    0.000000

No dispersion parameters defined

Spatial dependence structure:
-----------------------------
Structure used is Independence

**** NOTE: the following global quantities have not been defined:

          Threshold for small positive values

OK to continue (Y/N, default Y)?
```

If you do not see this (or the fitting program terminates with an incomprehensible error message!) then something is wrong. In particular, if you see the error message

```
****ERROR**** Input error while reading line  50 of file logistic.def.
This *may* (but no promises!) be because you have at least one empty
line at the end of the file. If this is the case, delete the
offending line(s) and try again.
```

then there is probably at least one empty line at the end of the model definition file, that should be deleted as described in step 1 above. The clue is in the first line of the error message — the software is trying to read line 50 of the file but, as we noted earlier, the last line of model definition is line 49.

Assuming that the definition file has been read correctly, note the following:

- The line 'Data from 6 sites will be used in the fitting' indicates that the software has correctly identified 6 sites in file siteinfo.def. (**NB:** clearly, the software can

only fit to a site if the corresponding data are present in file gaugvals.dat — the software does *not* cross-check this).

- The model title appears in the output.
- The software prints out a meaningful interpretation of the codes found in file logistic.def (including some default values for codes that we didn't put in, relating to dispersion parameters and spatial structure).
- We did not define a threshold below which positive values should be considered 'small', and the software has warned us of this. Recall from the beginning of this section (page 32) that any value less than 0.1mm should be regarded as a trace, and appears in the data files as 0.05mm. Hence, in our modelling of these data, we should define this information to the system at the outset (in fact, it makes no difference in this particular example, but it is good modelling practice).

3. Having been reminded that we might want to define a trace threshold, let's go back and do it. We don't want to continue with the analysis at present, so enter `N` or `n` in response to the on-screen prompt. The fitting program terminates. Before proceeding, have a look at the contents of the directory (Unix users type `ls -lt`, Windows users type `dir`). Notice that files logistic.res, logistic.de2 and res_logi.dat have all been created. The latter two are empty, but logistic.res contains a copy of the screen output above.

   Open logistic.def for editing again. Tables 2 (row 8) and 7 tell us how to define a trace threshold to the system: add an extra line

   ```
   8    0.1000    1    1
   ```

   to the end of the file. Take this opportunity to correct any other input errors, if there are any. Save the file and quit the editor.

4. Now let's start the fitting program again, by typing `fit_logi.` and pressing `Enter/Return`. This time the result is slightly different:

   ```
   ****WARNING**** Output file logistic.res already exists.
   Overwrite it (Y/N, default N)?
   ```

   This is intended to protect against accidentally overwriting the results of a previous fit. Any response other than `Y` or `y` will cause the program to terminate, giving you a chance to rename file logistic.res so that it doesn't get overwritten. At present, the file only contains results from the aborted step 2 above so it doesn't matter if we overwrite it. Enter `Y` or `y` to proceed, and again in response to

   ```
   ****WARNING**** Output file res_logi.dat already exists.
   Overwrite it (Y/N, default N)?
   ```

and

```
****WARNING**** Output file logistic.de2 already exists.
Overwrite it (Y/N, default N)?
```

We end up with similar output to that in step 2 above:

```
TRIVIAL LOGISTIC REGRESSION EXAMPLE
===================================
Initial parameter estimates:

Main effect:                                          Coefficient
------------                                          -----------
Constant                                                0.000000

Global quantities:
------------------
          Trace threshold               :    0.1000

No dispersion parameters defined

Spatial dependence structure:
-----------------------------
Structure used is Independence

OK to continue (Y/N, default Y)?
```

Now, however, the trace threshold has been defined to the system and the reminder has disappeared. To continue at this point, press the `Enter/Return` key (the default response is `Y`, so anything other than `N` or `n` will be interpreted as 'Yes').

5. The next prompt is

```
Input maximum number of iterations for iterative weighted
least squares (default unlimited):
```

Press `Enter/Return` here for an unlimited number of iterations (in practice this means 'iterate to convergence' — you are only likely to want a limited number of iterations when estimating parameters in nonlinear transformations, as discussed in Section 6 above). The software reads the data from gaugvals.dat and fits the model[9]. In this case, convergence occurs after 3 iterations and the following output is written to screen:

[9] Windows users: if the program terminates at this point with an error message, it may be because `gaugvals.dat` was created on a Unix system. The easiest way to rectify the problem is to open `gaugvals.dat` using WordPad, and then to resave it without making any changes.

```
     Results after   3 iterations:
     Log-likelihood -                -24025.881
     Deviance -                       48051.762
     Largest standardised score -        0.0000  (parameter   1)
     Number of observations -             35192
     No. of parameters estimated -            1
     Residual degrees of freedom -        35191
     Likelihood ratio statistic (initial vs final) -    734.7087

Computing covariance matrix of estimates ...
Final parameter estimates:

Main effect:                                          Coefficient  Std Err
------------                                          -----------  -------
Constant                                                0.290501   0.0108

Global quantities:
------------------
          Trace threshold               :    0.1000

Spatial dependence structure:
-----------------------------
Structure used is Independence

     Do you want a basic residual analysis (Y/N, default=Y)?
```

For the moment, we just draw attention to those features of the output that relate to the fitted model. These are as follows:

- The 'Coefficient' value of 0.290501 is the estimate of β_0 in **(1)**. The corresponding probability of rainfall on any day is $\exp(0.290501)/[1 + \exp(0.290501)] = 0.572$. This is perhaps a long-winded way to discover that 57.2% of the values in the database are non-zero, but it does at least provide some insight into the model structure, and serves as a simple check on the software output!
- The value of 0.0108 in the 'Std Err' column is the nominal standard error of this coefficient. It has been calculated under the assumption that data from all sites are independent, and hence is probably too small. We will see later how the software can be used to adjust these standard errors for inter-site dependence. In the meantime however, it may be convenient (and not too inaccurate) to work on the assumption that for typical climate datasets, 'independence' standard errors may be underestimated by a factor of up to two[10]. This allows us at least to make informal assessments of parameter uncertainty. Here, the ratio of the coefficient to its nominal standard error is $0.290501/0.0108 = 26.7$. This is vastly in excess

[10] After adjusting for inter-site dependence, the correct standard error in this case turns out to be 0.0214 — roughly double the nominal value.

of 1.96 (the critical value for a 5% test of the null hypothesis $H_0 : \beta_0 = 0$ against the alternative $H_1 : \beta_1 \neq 0$ under the assumption that sites are independent) and hence, even after allowing for inter-site dependence, it is unlikely that the true value of the parameter is zero. Although this is not a particularly interesting (or reasonable) hypothesis to test at this stage, it does illustrate the interpretation of the output.

6. At this stage, we could finish the current fit by typing `N` or `n`. However, to illustrate some more features of the software it will be useful to carry out a basic residual analysis, so press `Enter/Return`. We will look at the results of this analysis in the results file rather than on screen so, when the analysis is completed and the prompt

   ```
   Generate file for further residual analysis (Y/N, default=Y)?
   ```

   appears, type `N` or `n`.

7. To look at the results of this analysis, open the file logistic.res in a text editor. This file shows the initial parameter estimates, log-likelihood at each iteration of the maximisation algorithm, final parameter estimates and residual analysis. For the moment, we will focus on some aspects of the residual analysis, as follows:

   - The first item is a table:

   ```
                |    Observed    ||     % correct
                |   Dry     Wet  || Observed  Expected
   -------------------------------------------------------
   Forecast dry |      0       0 ||    0.0       0.0
   Forecast wet |  15058   20134 ||   57.2      57.2
   -------------------------------------------------------
                OVERALL % CORRECT :   57.2      57.2
   ```

   This provides a very simplistic check of the probability structure of a model. It measures the performance of a naive forecaster who issues a forecast of 'Wet' whenever the GLM gives a rainfall probability greater than 0.5, and 'Dry' otherwise. The aim is *not* to maximise the proportion of correct forecasts — rather, it is to obtain good agreement between the performance you observe and the performance you expect. In this case the modelled probability of rain is 0.572 for every case in the database — so our forecaster would always forecast 'Wet', and would expect to be right 57.2% of the time. Unsurprisingly for this model, there is an exact match between observed and expected performance here — $100 \times 20134/(15058 + 20134) = 57.2\%$ of days *were* wet and hence resulted in correct forecasts.

   - The next item in the residual analysis is an extended version of the same table. Here, the aim is to check the forecast probabilities by grouping them. The basic idea is that if we collect together all days for which the forecast probability

of rain is 0.1, then 10% of these days should have experienced rain. In practice we collect together all days for which forecast probabilities lie in the ranges $(0, 0.1), [0.1, 0.2), \ldots, [0.9, 1.0)$ and calculated the observed and expected numbers of wet days. A lack of agreement between in any cell of the table indicates a problem with the model. Clearly however, for such a simple model as this the information from these tables is not particularly useful.

- The remaining items of residual analysis relate to Pearson residuals. If the model is correct, they all come from distributions with mean zero and the same standard deviation (usually 1). To illustrate how they may guide us in model-building, locate the following section of the results file:

```
MODEL PERFORMANCE BY MONTH
--------------------------

       |        ||     Pearson residuals
 Month | N days ||   Mean    Std Dev   S.E. mean
-------------------------------------------------
     1 |   2986 ||  0.1582    0.9639     0.0355
     2 |   2677 ||  0.1393    0.9697     0.0372
     3 |   3023 || -0.1120    1.0102     0.0353
     4 |   2908 || -0.1875    1.0099     0.0360
     5 |   2951 || -0.2180    1.0082     0.0354
     6 |   2904 || -0.2084    1.0088     0.0360
     7 |   2980 || -0.0786    1.0085     0.0354
     8 |   2990 || -0.0741    1.0082     0.0355
     9 |   2897 ||  0.0304    0.9953     0.0361
    10 |   2986 ||  0.0925    0.9822     0.0355
    11 |   2871 ||  0.2122    0.9452     0.0361
    12 |   3019 ||  0.2549    0.9279     0.0355
-------------------------------------------------
```

  This gives the mean Pearson residual for each month of the year, together with standard errors. These standard errors *have* been corrected for inter-site dependence, as indicated at the bottom of the results file — details of the correction are given in Appendix D. Notice that the means in months 3–8 are all negative (indicating overprediction by the model), whereas the remainder are positive. The clear systematic structure tells us that the model is inadequate. An obvious way to improve things is to add some seasonal structure to the model.

Close the results file before proceeding.

7.2.2 Logistic regression with seasonality

We have now fitted a trivial logistic regression model, and established that it fails to capture the seasonality in the data. We therefore wish to extend this model. We could do this by

writing a new model definition file; however, we can speed things up (and minimise the risk of errors) by making slight modifications to the model we've just fitted. The file logistic.de2 now contains an updated version of the model definition file, with parameter values corresponding to the fitted model. So, to add seasonal structure to the model:

1. Move file logistic.de2 to logistic.def (this will overwrite the existing logistic.def), and open it for editing. Notice that the value of the 'Constant' term in the model is now 0.2905, as fitted previously.

2. Decide on a plausible representation of seasonality, from the options available in Tables 2 and 3. A good starting point is usually a Fourier representation of the annual cycle at a daily timescale. This requires both cosine and sine coefficients to be defined (since the phase of the cycle is unknown). Since the resulting covariates vary daily, we need to look at Table 3, and find that the required cosine term corresponds to a 'Code 1' value of 21. The sine term corresponds to a value of 22. So to define a simple annual cycle, insert the following two lines between the 'Constant' and 'Trace threshold' rows (recall from page 15 that rows must be ordered according to the value of `COMPONENT` which is 4 for daily effects and 8 for the trace threshold):

   ```
   4    0.0000   21
   4    0.0000   22
   ```

   In the absence of prior information, a value of zero is often a good starting point for estimation of βs in the fitting programs.

   Finally, remember to update the title of the model — for example 'LOGISTIC REGRESSION WITH SEASONALITY'. Save the modified definition file and quit the text editor. If for some reason you want to keep the results file from the previous model, rename it at this stage.

3. Run the fitting program again by typing `fit_logi`, and proceed until asked to check the model definition:

   ```
   LOGISTIC REGRESSION WITH SEASONALITY
   ====================================
   Initial parameter estimates:

   Main effect:                                          Coefficient
   ------------                                          -----------
   Constant                                                0.290500
   Daily seasonal effect, cosine component                 0.000000
   Daily seasonal effect, sine component                   0.000000

   Global quantities:
   ------------------
   ```

```
          Trace threshold              :    0.1000

No dispersion parameters defined

Spatial dependence structure:
-----------------------------
Structure used is Independence

OK to continue (Y/N, default Y)?
```

The software has interpreted the codes in logistic.def as seasonal cosine and sine components (if you do *not* see the output above, you have made a mistake with your coding and should correct logistic.def before proceeding).

4. Press `Enter/Return` to continue, and then again to request iteration to convergence. The results are as follows:

```
     Results after   3 iterations:
     Log-likelihood -                -23612.764
     Deviance -                       47225.528
     Largest standardised score -        0.0000  (parameter   1)
     Number of observations -             35192
     No. of parameters estimated -            3
     Residual degrees of freedom -        35189
     Likelihood ratio statistic (initial vs final) -    826.2342
Computing covariance matrix of estimates ...
Final parameter estimates:

Main effect:                                          Coefficient  Std Err
------------                                          -----------  -------
Constant                                                0.297388   0.0109
Daily seasonal effect, cosine component                 0.389860   0.0155
Daily seasonal effect, sine component                  -0.207276   0.0154

Global quantities:
------------------
          Trace threshold              :    0.1000

Spatial dependence structure:
-----------------------------
Structure used is Independence

     Do you want a basic residual analysis (Y/N, default=Y)?
```

Note the following:

- The nominal standard errors indicate that all three coefficients differ significantly from zero at any reasonable level, even after allowing for the effect of inter-site dependence.
- The maximised log-likelihood for this model is -23612.764. That for the previous model was -24025.881. The addition of two terms to the model has therefore increased the log-likelihood by 413.117. If this increase is doubled we obtain the LIKELIHOOD RATIO STATISTIC 826.234, which is also reported in the output. If all sites were independent we could compare this value with the appropriate upper percentage point of a chi-squared distribution with 2 degrees of freedom, to determine whether the data support the more complicated model (see Appendix B.1). This is an alternative to the comparison of estimates with their standard errors. In general, it is to be preferred since it automatically adjusts for correlation among the covariates. As before, inter-site dependence renders the procedure strictly invalid. The software *does* allow the option of adjusting the statistic to correct for this. However, since the upper 0.1% point of the χ^2_2 distribution is 15.20 this is hardly necessary: the old model is overwhelmingly rejected in favour of the model including seasonality.
- Inference can be based on the deviance rather than the log-likelihood. In fact, for the logistic models considered here the deviance is just $-2 \log L$ and so we can directly compare deviance reductions with tables of the appropriate chi-squared distributions. However, this interpretation of deviance is *not* valid for all models. From now on we will concentrate exclusively on the log-likelihood (whose interpretation *is* the same for all models). See the Appendix for further details.
- We may wish to check that the numerical algorithm has converged to a maximum of the log-likelihood surface. One way to do this is to compute the log-likelihood derivatives with respect to each of the parameters estimated — these should all be close to zero. It is useful to standardise the derivatives (using the *second* derivatives) so that they should all of the same order of magnitude, to aid interpretation. The software outputs the largest of these standardised derivatives, under the heading `Largest standardised score`. Here it is zero to 4 decimal places, so the algorithm has found a maximum. If the value is not close to zero, the algorithm may be having difficulty with the estimation of the corresponding parameter; here it is parameter 1, which is the constant term in the model.

5. For a residual analysis, press `Enter/Return`, then type `N` or `n` as before in response to the prompt

   ```
   Generate file for further residual analysis (Y/N, default=Y)?
   ```

   Examine the results of the residual analysis by opening file logistic.res in a text editor as before. Notice the following:

   - The tables of observed versus expected performance are now more useful, responding to the seasonal variation in wet day probabilities.

- There is still some seasonal structure in the Pearson residuals (negative means in March–May and August–October, positive means elsewhere; the values of 0.0736 for February and 0.0782 for July are significantly different from zero). However, the magnitude of the structure is much reduced. The pattern may be due to some small misspecification of the cycle, or to some other covariate that has not been included in the model. Previous days' rainfalls are obvious candidates here, since rainfall sequences are generally autocorrelated in time. This autocorrelation will affect the calculation of standard errors and likelihoods, so it is useful to account for it early on in a model-fitting exercise.

Close the results file before proceeding.

7.2.3 Logistic regression — accounting for autocorrelation

The modelling of autocorrelation is achieved, within the GLM framework, by including previous days' values as covariates in a model. This poses a number of questions, for example: how many previous days' values should be included? Can we benefit by transforming them? If so, what transformation should we use? Should we consider previous days' values at each site individually, or can we benefit by averaging over neighbouring sites as well?

In this tutorial, we will indicate how to go about answering the first two of these. Once users are familiar with the software, they will be able to answer the third as well. Proceed as follows:

1. Move logistic.de2 to logistic.def, to take advantage of the model just fitted.

2. Open logistic.def for editing. We'll start by adding a single previous day's rainfall, without any attempt at transformation. This covariate varies on a daily timescale, so `COMPONENT` has a value of 4. From Table 3, we need to put a '1' in the 'Code 1' field. So insert the following before the 'Trace threshold' row:

   ```
   4    0.0000    1
   ```

   Finally, give the new model a title before saving the definition file and quitting the text editor. For example, 'LOGISTIC REGRESSION WITH SEASONALITY & Y[t-1]'.

3. Run the fitting program until convergence. The fitting results are as follows:

   ```
   Results after   5 iterations:
   Log-likelihood -                -17954.621
   Deviance -                       35909.243
   Largest standardised score -         0.0000  (parameter   4)
   Number of observations -              28236
   No. of parameters estimated -             4
   ```

```
     Residual degrees of freedom -         28232
     Likelihood ratio statistic (initial vs final) -   1990.2690

Computing covariance matrix of estimates ...
Final parameter estimates:

Main effect:                                          Coefficient  Std Err
------------                                          -----------  -------
Constant                                               -0.034282   0.0145
Daily seasonal effect, cosine component                 0.340119   0.0179
Daily seasonal effect, sine component                  -0.178708   0.0177
Y[t-1]                                                  0.170495   0.0047

Global quantities:
------------------
          Trace threshold               :    0.1000

Spatial dependence structure:
-----------------------------
Structure used is Independence

     Do you want a basic residual analysis (Y/N, default=Y)?
```

Note the following:

- From the nominal standard errors, all four coefficients differ significantly from zero at any reasonable level of significance, even after allowing for inter-site dependence.
- The number of observations has decreased — from 35192 to 28236. This is because of missing values in the dataset — any case for which the previous day's value is missing cannot be used to fit the current model.
- The log-likelihood is 17954.621. However, this cannot be compared directly with the value of -23612.764 for the previous model (which would suggest a likelihood ratio statistic of over 11,000) because such comparisons can only be carried out for models fitted to the same data. Instead, we should use the reported likelihood ratio statistic of 1990.2690 — this has been computed using just those cases used in fitting the current model. Again, the increase is hugely significant.

Although these results support the inclusion of the previous day's rainfall into the model, before proceeding we may want to compare some transformations of this quantity. In particular, we may wonder whether knowledge of the amount of rain yesterday is more useful than just knowing *whether* it rained. We will defer any further residual analyses until we have finished modelling autocorrelation structure. So type `N` or `n` to terminate the fitting program at this point.

4. Edit logistic.def again. At this stage, the penultimate line contains

```
4    0.0000    1
```

and defines the previous day's value. We now wish to define a transformation of this value. From Table 3, this can be achieved by inserting a value in the 'Code 2' field. Now from Table 5, the value 3 defines a transformation that takes the value 1 for a non-zero amount and 0 otherwise. So change the penultimate line to:

```
4    0.0000    1    3
```

Save the file and rerun the fitting program. The maximised log-likelihood is now −15527.415. This *is* directly comparable with the value of -17954.621 obtained above, since the models are fitted to the same dataset. Clearly, the new model is vastly superior to the old one, so the transformation is worthwhile. Of course, we could experiment with other transformations in Table 5 to find the one with the highest log-likelihood, but there is no further need of illustration here. We will proceed to try and establish how many previous days' values are needed in the model. Type `N` or `n` to terminate the fitting program.

5. Move file logistic.de2 to logistic.def, and open it for editing.

At this point we could simply expand the model to include an indicator for rainfall occurrence 2 days ago, then 3 days ago and so on. However, the effect of this is successively to add covariates that may be highly correlated. As a result, inference based on nominal standard errors can be misleading, and inference based on nominal log-likelihoods is to be preferred. But since there is a lot of missing data, if we do this then each model will be fitted to a different dataset. It may be useful, before proceeding, to establish a dataset containing just those cases for which the required covariates are present for all models that we might reasonably contemplate fitting. For the sake of argument, let's restrict our search to models containing at most 4 previous days' values. According to Table 3, a value of 4 in the 'Code 3' field of a daily covariate will cause the software to discard any cases with missing values at the same site for any of the previous 4 days. Models with different numbers (up to 4) of 'previous day' covariates can therefore be compared directly using nominal log-likelihoods.

We start by refitting the last model to just those cases with at least 4 previous days' values available. The penultimate line of the definition file currently contains

```
4    2.1661    1    3    1I(Y[t-1]>0)                                    3
```

After the last fit, the software automatically inserted a value of 1 in the 'Code 3' field. Change this to a 4:

```
4    2.1661    1    3    4I(Y[t-1]>0)                                    3
```

Save the definition file and quit the editor.

6. Run the fitting program to convergence, without obtaining a residual analysis. The final log-likelihood is -8001.956, based on only 14575 observations (a consequence of the high percentage of missing values in the database). Notice that the likelihood ratio statistic is reported as 0.9486, even though the model has not changed. The increase is due to the fact that the original parameter estimates are not quite optimal for the reduced data set.

7. Move file logistic.de2 to logistic.def, add an extra line corresponding to an indicator for rainfall 2 days ago:

```
4     0.0000     2     3
```

   and change the model title ('LOGISTIC REGRESSION WITH SEASONALITY, Y[t-1] & Y[t-2]', say). Run the fitting program — the nominal log-likelihood, based on the same 14575 observations as previously, is -7845.682. The likelihood ratio statistic of 312.55, for the addition of a single parameter, is still hugely significant even after allowing for spatial dependence.

8. Fit models including both 3 and 4 previous days' rainfall indicators, in a similar manner. You should obtain log-likelihoods of -7804.427 and -7803.805 respectively. This suggests that we should consider a model that incorporates just three previous days' indicators, since the fourth does not increase the log-likelihood significantly.

   Having established that just 3 previous days are sufficient for our model, we may want to expand the dataset to include all cases for which 3 previous days' values are available (the fits above are limited to cases for which 4 previous days' values are available). It may also be a good idea to look at the model residuals again. So: refit the '3 day' model using all available data (change 'Code 3' from 4 to 3, in the line corresponding to the previous day's indicator — as a check, you should obtain a log-likelihood of -9734.604 based on 18158 observations), and obtain a residual analysis. In the residuals, notice that the tables of observed versus expected performance now show good agreement over a wide range of forecast probabilities; also that the systematic seasonal structure in Pearson residuals has decreased (although there is still a block of negative means from August to November).

   Obviously, we could experiment with the addition of other covariates representing temporal dependence (in particular, in rainfall sequences it is natural to consider the effects of 'persistence', which can be modelled via transformation 5 in Table 5). However, for the purposes of illustration we will now move on to consider interactions.

7.2.4 Logistic regression — interactions

In northwestern Europe, winter rainfall tends to be produced by frontal weather systems that may last for several days. In summer however, there are more short-lived convective

events. As a result, autocorrelation in rainfall sequences tends to be weaker in summer than in winter. Therefore, in a realistic model for rainfall occurrence, any parameters associated with previous days' rainfalls should themselves vary seasonally. Within a GLM, this can be achieved by defining interactions between previous days' rainfalls and seasonal covariates. We will use this example to demonstrate the software's capability for handling interactions.

1. Move file logistic.de2 to logistic.def, and open it for editing. All being well, the last 8 lines of the file should now read as follows:

```
LOGISTIC REGRESSION WITH SEASONALITY & 3 PREVIOUS DAYS
    0   -1.2387                Constant
    4    0.2143   21           Daily seasonal effect, cosine component    1
    4   -0.1128   22           Daily seasonal effect, sine component      2
    4    1.8042    1    3     3I(Y[t-1]>0)                               3
    4    0.5957    2    3      I(Y[t-2]>0)                               4
    4    0.3908    3    3      I(Y[t-3]>0)                               5
    8    0.1000    1    1      Trace threshold
```

   Note the following points:

   - When writing the updated definition files, the software has provided descriptive text in each row. This makes the definition files more readable.
   - At the right-hand end of each row is the covariate number. For example, the indicator for rainfall occurrence yesterday is covariate number 3. These numbers are required for defining both interactions and nonlinear transformations.

   We wish to define interactions between the 'seasonal cycle' and 'previous days' covariates. This will result in the addition of 6 additional terms to the model — one for each seasonal/previous day combination. These are 2-way interactions (each term involves 2 covariates). Referring to row 5 of Table 2, these are defined by entering the numbers of the interacting predictors in the 'Code 1' and 'Code 2' fields. So add the following lines before the trace threshold:

```
5    0.0000    1    3
5    0.0000    2    3
5    0.0000    1    4
5    0.0000    2    4
5    0.0000    1    5
5    0.0000    2    5
```

   These define interactions between covariates 1 and 3, 2 and 3, ..., 2 and 5. Give your model a title, save and quit the editor.

2. Run the fitting program. At the 'OK to continue (Y/N, default Y)?' prompt, check the model definition carefully. The software splits the output into 'main effects' and '2-way interactions'. If you have defined the model correctly, the 'interactions' section should read as follows:

```
2-way interactions:                                  Coefficient
-------------------                                  -----------
Daily seasonal effect, cosine component                0.000000
with I(Y[t-1]>0)
Daily seasonal effect, sine component                  0.000000
with I(Y[t-1]>0)
Daily seasonal effect, cosine component                0.000000
with I(Y[t-2]>0)
Daily seasonal effect, sine component                  0.000000
with I(Y[t-2]>0)
Daily seasonal effect, cosine component                0.000000
with I(Y[t-3]>0)
Daily seasonal effect, sine component                  0.000000
with I(Y[t-3]>0)
```

   Assuming your model definition is correct, run the fitting program to convergence, but do not ask for a residual analysis. The maximised log-likelihood should be -9726.781, and the likelihood ratio is 15.6460. If all sites were independent we could compare this with the upper percentiles of a χ^2_6 distribution (since we have added 6 extra parameters to the model), to find that it falls between the upper 97.5% and 99% points of χ^2_6 (which are 14.45 and 16.81 respectively). This would indicate that the additional terms are significant at the 5% level but not at the 1% level. However, the effect of inter-site dependence is generally to make things appear more statistically significant than they actually are. It is therefore probably worth computing an adjusted likelihood ratio statistic at this point.

7.2.5 Adjustment for inter-site dependence

So far, all of our model fitting has assumed that sites are independent (you may have noticed, in the software output, a reminder of this under the heading `Spatial dependence structure:`). As well as invalidating standard errors and likelihood ratio tests, this will cause problems if we try and simulate the fitted model in its current form, since the simulated sequences from each of the 6 sites will be independent. This is clearly unrealistic.

When fitting models, if any spatial dependence structure other than 'Independence' is selected, the software will compute adjusted standard errors, and will report a 'raw' and 'adjusted' likelihood ratio statistic. Therefore, to obtain the adjusted version we just need to specify a dependence structure and refit the previous model.

The software offers several alternative ways to model inter-site dependence in binary sequences. These are summarised in Table 8; more details are given in Appendix E.3. The precise structure chosen does not matter as far as the fitting programs are concerned: it only makes a difference if the models are subsequently used for simulation. For illustrative purposes we will consider the 'binary weather state' model (label 21 in Table 8). This will be discussed in more detail later, when we consider simulation.

Proceed as follows:

1. Make a note of the final parameter estimates and standard errors for the model you have just fitted. For the main effects, these are as follows:

```
Main effect:                                          Coefficient  Std Err
------------                                          -----------  -------
Constant                                               -1.234264   0.0315
Daily seasonal effect, cosine component                 0.236399   0.0452
Daily seasonal effect, sine component                  -0.194840   0.0436
I(Y[t-1]>0)                                             1.807814   0.0382
I(Y[t-2]>0)                                             0.588904   0.0416
I(Y[t-3]>0)                                             0.392094   0.0400
```

   The interactions are omitted here to save space.

2. Open logistic.def for editing. Currently this contains zero coefficients for all the interactions, and therefore represents the initial model with no interaction terms included. Row 10 of Table 2 indicates how to define spatial structure. We wish to use structure 21, which involves a single parameter (Table 8). So append the following line at the end of the file:

```
10    0.0000   21    1
```

   Give the model a title, and quit the editor.

3. Run the fitting program, but stop at the `OK to continue (Y/N, default Y)?` prompt and check that the spatial dependence structure has been read correctly — you should see

```
Spatial dependence structure:
-----------------------------
Structure used is Conditional independence given 'wet/dry' weather s
          Increase in logit on a 'wet' d:    0.0000
```

   If there are any errors at this stage, go back and correct them; otherwise run the fitting program to convergence. Do not ask for a residual analysis. Upon completion, look at the summary:

```
     Results after   3 iterations:
     Log-likelihood -                  -9726.781
     Deviance -                        19453.563
     Largest standardised score -         0.0000  (parameter   4)
     Number of observations -              18158
     No. of parameters estimated -            12
     Residual degrees of freedom -         18146
     Likelihood ratio statistic (initial vs final) -     15.6460
Computing covariance matrix of estimates ...
Computing dependence-adjusted likelihood ratio ...
     Dependence-adjusted LR statistic -                   7.0448
        NB incorporates secondary adjustment for
        hypothesis testing, since initial coefficient
        vector contained zeroes: scale factor was        0.99867
```

Everything is exactly as it was before, except for the addition of the last five lines which relate to the dependence-adjusted likelihood ratio. The parameter estimates are also the same as previously, but the standard errors have been corrected to allow for the dependence.

The output above refers to a "secondary adjustment", which has been applied to improve the accuracy of the model comparison: the details need not concern us here (they can be found in Appendix C.3). The most important part of the output is the value of the statistic itself. At only 7.0448, this is less than half of the original and falls in the centre of the χ^2_6 distribution. The increase is therefore not large enough to justify the extended model over the simpler one. This appears counter-intuitive, since there are strong *a priori* grounds for believing that at least some of these terms should appear in the model. At this point it is useful to look at the table of estimates and their standard errors (the main effects are omitted here to save space):

```
2-way interactions:                                  Coefficient  Std Err
-------------------                                  -----------  -------
Daily seasonal effect, cosine component                0.070191   0.0809
with I(Y[t-1]>0)
Daily seasonal effect, sine component                  0.163483   0.0811
with I(Y[t-1]>0)
Daily seasonal effect, cosine component               -0.074374   0.0854
with I(Y[t-2]>0)
Daily seasonal effect, sine component                 -0.087872   0.0847
with I(Y[t-2]>0)
Daily seasonal effect, cosine component               -0.034958   0.0812
with I(Y[t-3]>0)
Daily seasonal effect, sine component                  0.080595   0.0808
with I(Y[t-3]>0)
```

The interactions involving Y_{t-2} and Y_{t-3} all appear insignificant at the 5% level. This suggests that we may try dropping these from the model. Of the two remaining terms involving Y_{t-1}, only one appears significantly different from zero. However, a seasonal cycle involves 2 parameters (phase and amplitude) and neither of these are known *a priori*, so from a modelling perspective it is good practice always to add seasonal components in pairs. We will keep both of these terms in the model.

4. Open logistic.def for editing: delete the 4 rows corresponding to the insignificant interactions, save and quit the editor. Rerun the fitting program to convergence. The adjusted likelihood ratio statistic for this model compared to the 'main effects only' model is 4.1411. This is around the 87% point of a χ^2_2 distribution, which normally would not be sufficient to justify the inclusion of the interactions. However, our understanding of European climate strongly suggests that these interactions *should* be present, so we will keep them in the model.

   The residual analysis for this model does not give any cause for concern, with the possible exception of the block of negative monthly residuals between August and November (which could be accounted for by adding 'half-year cycles' to the 'seasonal' component of the model — Table 3, Code 1, values 23 and 24). Mean residuals are close to zero at all sites and for all years. There is possibly some disagreement between observed and expected rainday frequencies in the second and sixth deciles (probabilities in the ranges $[0.1, 0.2)$ and $[0.5, 0.6)$ respectively), but in practical terms this discrepancy is of little consequence.

   In applications, we would probably be satisfied with this model. However, for illustrative purposes we will now expand it, to see if there is any significant inter-site variation.

7.2.6 Logistic regression — site effects

As a preliminary step in modelling site effects, it is useful to produce a bubble map showing the mean residuals at each site, for the current model. This is presented in Figure 5. The means have been divided by their standard errors, to adjust for the fact that some sites may have longer records than others.

With only 6 sites in Figure 5, it is clearly not possible to identify complex patterns. In general, such a limited network may enable broad trends to be quantified, but little else. The map indicates a possible northwest-southeast gradient in the residuals, that could be approximated by a planar surface. By adding such a structure to the model, we can determine whether this is a genuine effect, or merely a 'chance' configuration[11].

[11] Strictly speaking, it is bad scientific practice to both derive and test a hypothesis using the same data. However, standard procedures can at least give some informal basis for judging the importance of a perceived effect. In particular, non-significant results (i.e. those indicating that an apparent pattern is merely due to chance) are probably reliable.

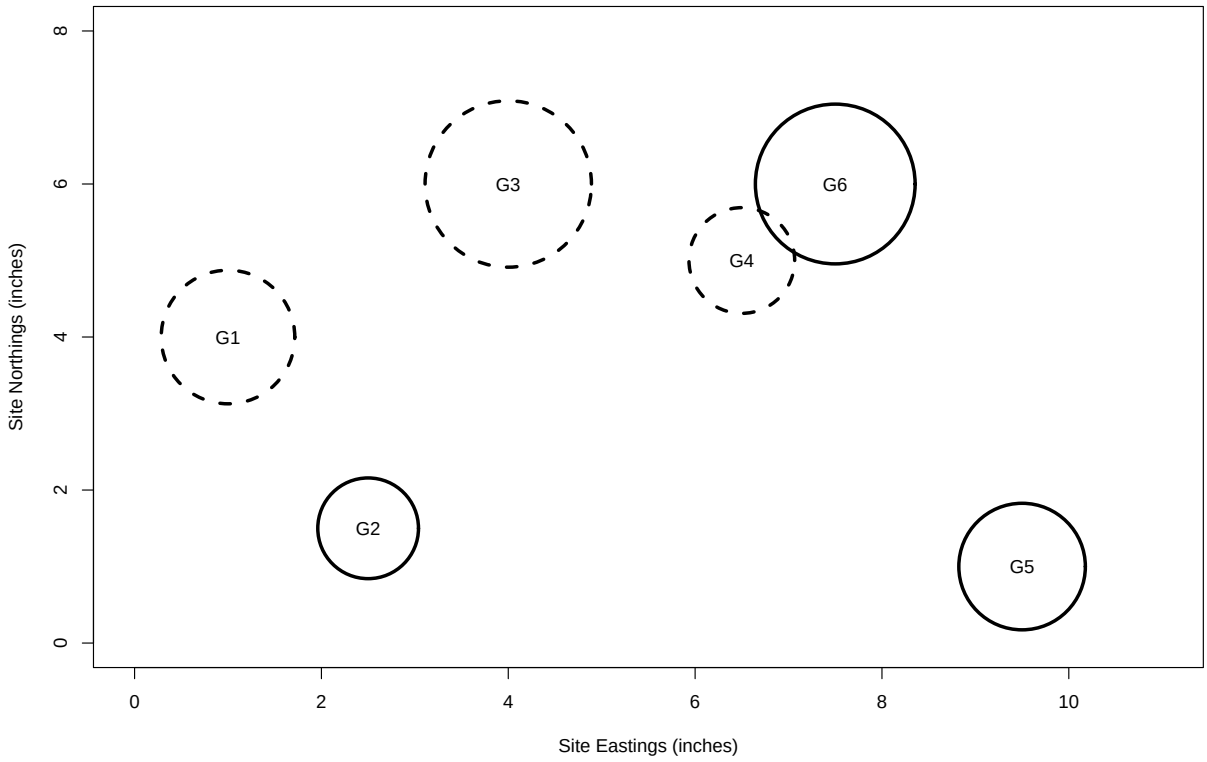

Figure 5: Bubble map showing mean residuals from fitted logistic regression model at each site. Circle areas are proportional to standardised mean residuals. Positive values are denoted by solid circles, negative values by dashed circles.

A planar surface may be represented as a linear combination of Eastings and Northings co-ordinates. For the present example, we could enter these directly into the model. However, we will instead take the opportunity to illustrate the use of Legendre polynomials of degree 1 in each direction (the model formulation is equivalent, since a degree 1 polynomial is a linear transformation of the underlying quantity). Proceed as follows:

1. Move logistic.de2 to logistic.def, and open it for editing. Check that the last 11 lines of the file are as follows:

```
LOGISTIC REGRESSION WITH SEASONALITY, PREVIOUS & INTERACTIONS
    0   -1.2380                Constant
    4    0.1950   21           Daily seasonal effect, cosine component    1
    4   -0.1914   22           Daily seasonal effect, sine component      2
    4    1.8051    1    3     3I(Y[t-1]>0)                                3
    4    0.5938    2    3      I(Y[t-2]>0)                                4
    4    0.3919    3    3      I(Y[t-3]>0)                                5
```

```
 5    0.0334    1    3     2-way interaction: covariates  1 and  3
 5    0.1500    2    3     2-way interaction: covariates  2 and  3
 8    0.1000    1    1     Trace threshold
10    6.4872   21    1     Parameter  1  in spatial dependence model
```

2. Defining site effects now requires some care. Note the following:

   - Site effects (for which `COMPONENT` is 1 — see Table 2) must be defined *after* the constant term and *before* any other covariates. As a result, existing covariate numbers will change (for example, if we insert two rows corresponding to site effects then the existing covariate 1 will become covariate 3, and so on).
   - If there are interactions involving covariates whose numbers have changed, the corresponding interaction rows will need to be updated to reflect this change (forgetting to do this is a frequent source of errors!).
   - From row 1 of Table 2, site effect data are taken from file siteinfo.def. Our siteinfo.def defines 2 attributes (Eastings and Northings) — either of these can be selected via the 'Code 1' field. Transformations (in our case Legendre polynomials) can be selected via an appropriate entry in the 'Code 2' field. Table 4 indicates that for a degree 1 polynomial we need a value of 31 in this field.
   - The Legendre polynomial representation requires, in addition to the degree of the polynomial, specification of the range (a, b) over which the representation holds. a and b are essentially parameters involved in the transformation of an underlying covariate, and hence can be defined via row 7 of Table 2.

   The desired model can therefore be specified by making the following changes to the definition file:

   (a) Insert two rows after the 'Constant' row:

   ```
   1    0.0000    1   31
   1    0.0000    2   31
   ```

   Each of these defines a site effect: the first is transformation 31 of site attribute 1, the second is transformation 31 of site attribute 2.

   (b) Edit the rows corresponding to interactions, so that they point to the correct main effects:

   ```
   5    0.0334    3    5     2-way interaction: covariates  1 and  3
   5    0.1500    4    5     2-way interaction: covariates  2 and  3
   ```

   The software ignores the descriptive text when reading the file, so there is no need to update this unless you particularly want to.

   (c) Add 2 rows after the interactions, to define the Eastings limits over which the Legendre polynomial representation is to hold. These should be chosen so that all sites fall within the limits. Based on the information in siteinfo.def, we will set the limits to 0 and 11 respectively:

```
7    0.0000    1    1
7   11.0000    1    2
```

These rows define the first and second parameters, respectively, in the transformation of covariate 1 (see Table 4).

(d) Add two more rows to define the Northings limits (0 and 8):

```
7    0.0000    2    1
7    8.0000    2    2
```

(e) Change the title of the model.

The model definition section of the file should now look something like this:

```
LOGISTIC MODEL: SEASONALITY, PREVIOUS DAYS, SITE EFFECTS & SEAS/PREV INTERACTION
    0   -1.2380               Constant
    1    0.0000    1   31
    1    0.0000    2   31
    4    0.1950   21          Daily seasonal effect, cosine component    1
    4   -0.1914   22          Daily seasonal effect, sine component      2
    4    1.8051    1    3    3I(Y[t-1]>0)                                3
    4    0.5938    2    3     I(Y[t-2]>0)                                4
    4    0.3919    3    3     I(Y[t-3]>0)                                5
    5    0.0334    3    5     2-way interaction: covariates  1 and  3
    5    0.1500    4    5     2-way interaction: covariates  2 and  3
    7    0.0000    1    1
    7   11.0000    1    2
    7    0.0000    2    1
    7    8.0000    2    2
    8    0.1000    1    1     Trace threshold
   10    6.4872   21    1     Parameter  1  in spatial dependence model
```

Save the file, and quit the editor.

3. Run the fitting program. Check carefully that the software has interpreted the definition file as you intended! In particular, check the interactions and the section under 'Parameters in nonlinear transformations'. If there are any errors here (or if the program terminates with an error message), go back and correct the definition file. Otherwise, run the program to convergence, obtain a residual analysis and also press `Enter/Return` in response to the prompt

```
Generate file for further residual analysis (Y/N, default=Y)?
```

A successful fit of this model leads to a log-likelihood of -9728.521, with an adjusted likelihood ratio statistic of 3.7239. For an additional 2 parameters, this clearly is not a significant increase, which confirms our previous conjecture that no systematic regional

patterns of rainfall occurrence are detectable in these data. Nonetheless, for present purposes it suffices to take this as our 'final' model. A residual analysis is, to all intents and purposes, indistinguishable from that of the previous model.

4. Move the file logistic.de2 to logistic.def. This definition file is now ready for use in generating simulated rainfall occurrence sequences.

5. Before going on to examine the simulation program, take a quick look at the file res_logi.dat (generated in response to the request for further residual analysis, above). The first few rows of the file are as follows:

```
SITE YEAR MONTH DAY OBSERVED PREDICTED    ETA
  G1 1970     1   4   0.0000    0.4742 -0.1034
  G4 1970     1   4   0.0000    0.8558  1.7812
  G5 1970     1   4   0.0000    0.4968 -0.0128
  G6 1970     1   4   0.0000    0.7664  1.1879
  G1 1970     1   5   1.0000    0.3316 -0.7010
  G5 1970     1   5   1.0000    0.3520 -0.6103
  G1 1970     1   6   1.0000    0.6806  0.7563
  G5 1970     1   6   1.0000    0.6999  0.8470
```

Note the following:

- The first 4 columns are self-explanatory. The 'OBSERVED' column contains a 1 if the corresponding value in gaugvals.dat is non-zero, 0 otherwise. 'PREDICTED' gives the probability of obtaining a 1 under the model; 'ETA' is the linear predictor (in this case, the log odds for rainfall occurrence).
- The first case corresponds to 4th January 1970. This is because the earliest data in gaugvals.dat are from 1st January 1970, and the model contains 3 previous days' values as covariates. Hence 4th January is the first day for which all required covariates are

  present (although not at sites 2 and 3, apparently).

This example should give the general feel of a model-building exercise using this software. We have not illustrated all of its features, but hopefully the user should now be reasonably familiar with the tables of codes in Section 4.4 above, and should be able to work out how to define more complicated models. Although we have concentrated on logistic regression, the basic model-building process is the same for any GLM. The output varies slightly for different models (for example, residual analyses for binary data are not appropriate for continuous variables), but the basic framework is the same as that presented here. For full explanations of the analysis methods, see the references given at the start of this manual.

7.3 Simulation

In this section we give a brief introduction to the rainfall simulation program simrain. We will simulate rainfall sequences using the logistic regression model just fitted to generate sequences of wet and dry days, and using a gamma GLM defined in file gammamdl.def to generate rainfall amounts on wet days. This file has been generated by building up a model for rainfall amounts using the fitting program fit_gamm, in much the same way as the logistic regression model above. The only major difference between the two classes of model is that the gamma distribution has a shape (or dispersion) parameter which must also be specified. This has been estimated from the available data, and is defined in the penultimate row of the definition file as specified in Table 2.

We have so far not paid too much attention to the spatial dependence structures in the models, beyond the use of the 'binary weather state' model in order to obtain corrected standard errors and likelihood ratios. The idea behind this dependence model is that, on any given day, the area is in one of two states, 'wet' or 'dry'. On a 'wet' day, the probability of rain at all sites is increased whereas on a 'dry' day it is correspondingly reduced. The parameter in this scheme — estimated as 6.5120 by the fitting software — controls the amount by which the probabilities are increased on 'wet' days.

For rainfall amounts, inter-site dependence is most naturally modelled using correlations. In fact, the software considers correlations between Anscombe residuals (see page 23). The modelling of inter-site dependence in amounts is therefore achieved by specifying a spatial structure for the Anscombe residual correlation. The options available are summarised in Table 8: the supplied gammamdl.def uses the same value of 0.6891 for each pair of sites.

If you followed the instructions in Section 7.1 above, all necessary definition files should be present in your current directory. To investigate the rainfall simulation program, proceed as follows:

1. Type `simrain` and press `Enter/Return`. You should see the following:

```
 Opening files ...

SUBAREA DEFINITIONS
===================

REGION: Ashdown Forest

Subarea  1: Pooh and Piglet's side of the forest
Subarea  2: Christopher Robin's side of the forest
```

   The software has picked up the area definitions from file regions.def.

2. Press `Enter/Return` to continue. The software displays the list of sites that will be used for simulation, and prompts for confirmation again. Continue here, and again

at the subsequent prompts to accept both occurrence and amounts models. The next prompt is as follows:

```
SIMULATION OPTIONS
==================

 Input start year and month (YYYYMM):
```

At the risk of mixing our literary metaphors, or something, let's do some simulations of the year 1984. In this case the first month we want to simulate is January 1984. Enter `198401` and press `Enter/Return`. Then `198412` for the end year and month.

You now have to choose the number of realisations. Input `10`.

3. The next step, in response to the prompt

```
Now please select an output option:

     1.  Output regional monthly and annual summary information
     2.  Output daily values for each site
     3.  Output both detail and summary information
```

is to decide on an output format. Option 1 will produce, for the whole area and for selected regions defined in regions.def, monthly and annual totals. Options 2 and 3 are self-explanatory. Note that the storage of large quantities of daily data can result in large output files under options 2 and 3 (although, as we will see shortly, there is some control over this). For the moment we will obtain both detail and summary information, so input `3`.

4. Whenever we request regional summary information, and subareas have been defined in regions.def, the software prompts us:

```
Choose subareas for inclusion in output file (enter a subarea
number to toggle on/off, zero to finish and any other number to
include all subareas)

NUMBER  NAME                                                  INCLUDE?
------  ----                                                  --------
   1    Pooh and Piglet's side of the forest                     N
   2    Christopher Robin's side of the forest                   N
```

By default, statistics will be produced for the whole area but not for either of the subareas. To change this default, type `1` and press `Enter/Return`. The same prompt reappears, but this time with a `Y` in the `INCLUDE?` column for subarea 1. Entering `1` again will reset the value to `N`. Entering any number other than 0, 1 or 2 will set all the `INCLUDE?` flags to `Y`.

In this example we will not produce output for subareas, so enter `1` and `2` as often as necessary to reset the `INCLUDE?` flags to `N`. Then enter `0` to proceed to the next step.

5. Whenever (as here) we request daily output the next prompt appears:

```
You have requested daily output for each site defined. You may
restrict this output to a portion of one realisation if required.
Please enter 0 now to produce complete output for every realisation.
Otherwise, enter the number of the single realisation for which
output is required:
```

   The idea is that, if desired, we can reduce the size of daily output files by examining a portion of a realisation. Of course, we don't know yet which portions will be of interest! However, by setting the random number generator to a repeatable initial state (below), we will subsequently be able to reproduce any simulation exactly. Hence, in general, a sensible strategy may be to run the simulation program and obtain monthly and annual summary information to identify periods of interest; then do the simulation again under identical conditions, this time extracting the appropriate daily data.

   For the moment we will output a single month of daily data, to see the output format. Enter `5`, to produce daily output for the fifth realisation only. You get

```
Input start year and month for daily output (YYYYMM):
```

   We may as well output just the values for January 1984, so enter `198401` here, and again for the end year and month.

6. The final step is to input a seed for the pseudo-random number generator. Any integer less than 424967291 is acceptable, with the exception (on non-Unix systems) of zero. Even on a Unix system, a value of zero is *not* recommended since this initialises the generator from the system clock and you will never be able to reproduce the resulting sequence again! The results below were obtained using the seed 123456. If you use the same seed, you will get the same results.

   After this, no further input is required. Progress is written to screen, so you know whether you've got time for a cup of coffee (probably not, in this case!).

7. Now let's have a look at the results of the simulation. The output files, and their formats, are described in Section 5 above. First of all, open file mdlspec.txt. This contains all the information necessary to reproduce this simulation exactly in the future, if required. Note in particular that the random number seed is recorded at the bottom of the file. This allows you subsequently to go back and extract daily data for a small part of this simulation. It also means that you don't have to store large daily data files for long periods of time — once you have analysed the results, you can delete the daily data file in the knowledge that you can regenerate it in the future if necessary.

8. Now close mdlspec.txt and open daily.dat in its place. If you seeded the random number generator with 123456, the first 3 lines will be as follows (otherwise, some — not all — of the figures will differ):

```
 5 1984  1  1   3.89 1   6.24 0   8.19 0  12.70 0  21.84 0  18.84 0
 5 1984  1  2   0.12 0   1.01 0   0.88 0   3.85 0   2.74 1   1.15 0
 5 1984  1  3   6.91 1   6.87 1   2.88 0   3.34 0   4.35 0   5.71 0
```

The first column here is the realisation number (recall that we performed 10 simulations, but only requested daily output for the 5th). The next three columns are year, month and day.

The remaining columns appear in pairs. Each pair corresponds to a site, in the order defined in siteinfo.def. The first (decimal) value is a rainfall amount, and the second is a flag indicating whether it has been generated from the model or read from gaugvals.dat. The first value of 3.89 is an amount at site 1 for 1st January 1984. The '1' next to it indicates that this value has been simulated from the model — in fact, the simulation is conditioned on observed values at other sites as well (see Appendix E for details). The other sites all have zero flags for this day, indicating that the values are taken from gaugvals.dat. You may care to open gaugvals.dat and check that site G1 does indeed have missing data for 1st January 1984, and that the other sites have values as above. Our simulation has therefore produced an imputation of the missing values in the database. We will see how to produce a 'true' simulation, unconstrained by observations, below. For the moment, let's explore the output files further.

9. Close daily.dat, and open monthly.dat. If you seeded the random number generator with 123456, its contents are as follows:

```
 1 1984 0 62.8 78.8 38.2 107.2 37.7 84.9 52.1 47.8 51.3 69.9 75.9 86.1 792.8
 2 1984 0 62.7 79.4 39.0 109.1 31.6 85.5 49.1 52.6 51.4 73.9 80.2 90.5 805.0
 3 1984 0 59.8 79.0 37.6 108.3 31.3 80.1 56.2 48.3 52.2 72.0 72.1 85.0 782.0
 4 1984 0 59.5 89.9 35.0 102.6 34.0 87.8 55.8 48.0 49.5 71.8 82.1 90.4 806.5
 5 1984 0 59.9 81.8 42.7 109.6 33.7 82.1 49.1 45.3 48.9 75.5 77.8 94.3 800.9
 6 1984 0 59.6 86.3 38.4 107.7 29.7 83.7 47.0 47.0 50.6 81.0 76.0 92.5 799.6
 7 1984 0 58.9 82.4 36.9 106.2 30.7 87.2 49.6 48.7 50.4 72.5 88.9 92.6 804.9
 8 1984 0 59.5 87.9 42.0 107.0 32.3 82.5 52.8 52.0 50.5 72.4 83.2 81.1 803.2
 9 1984 0 58.7 88.9 37.5 110.5 33.6 85.5 51.1 47.0 52.1 70.5 74.4 79.9 789.7
10 1984 0 60.6 83.4 36.2 112.3 33.9 83.3 55.0 49.1 52.4 70.3 77.0 77.3 790.9
```

The first three columns are simulation number, year and region code (recall that we asked for summaries only for region 0, which is the whole area). There follow 12 monthly totals and an annual total. Looking at the final column we see that each of the 10 simulations gave a different annual total. These differences are purely due to different imputations of the data missing from gaugvals.dat. This exercise therefore gives us some idea of the uncertainty due to incomplete observational records. For example, the annual totals have mean 797.6mm and standard deviation 8.2mm, so we may be reasonably confident that, if all records had been complete, the true mean annual rainfall for 1984 would have been somewhere between 782mm and 814mm.

Now close monthly.dat, and rename it to monthly.obs. This will prevent these results from being overwritten by our next simulation exercise.

10. We will now run a second simulation, this time without constraining the models using observed data. To do this, we just need to delete all rows in gaugvals.dat that correspond to dates on or after 1st January 1984. In this way, the software will not find any data with which to constrain the simulation, but each realisation will be initialised using actual data from December 1983 (the distributions for 1st January 1984 will be calculated on the basis of previous days' values that remain in the data file).

    So: make sure you have a backup copy of gaugvals.dat and then edit it, delete the unwanted rows and save. The file should now be 24645 lines long; the last 6 rows are

    ```
    19831231  G1  2.45
    19831231  G2  1.24
    19831231  G3  1.96
    19831231  G4  2.13
    19831231  G5  3.09
    19831231  G6  2.01
    ```

11. Close gaugvals.dat and run simrain again. This time there are some prompts at the beginning, asking whether you want to overwrite daily.dat and mdlspec.txt (but not monthly.dat because you renamed it a few moments ago). The rest of the program runs as before. Enter the same start and end dates, and number of realisations. This time, however, when you reach the prompt

    ```
    Now please select an output option:

        1.  Output regional monthly and annual summary information
        2.  Output daily values for each site
        3.  Output both detail and summary information
    ```

    select option 1. Then proceed as previously — the software bypasses the options relating to daily output. Again, use any seed for the random number generator. I used 654321 this time (in general, you should vary the seed between simulations to avoid using the same basic pseudo-random number sequence every time).

    When the simulation has finished (more quickly than before, because there is no need to condition on observations), have a look at the monthly.dat file. The format is exactly the same as before, but now there is much more variation between realisations. For example, with a seed of 654321 the annual totals now have a mean of 890.7mm and a standard deviation of 89.8mm. On this basis, we would expect approximately 95% of annual totals in this region to fall in the range $890.7 \pm (2 \times 89.8) = (711.1, 1070.3)$mm. An analysis of the data in the original gaugvals.dat shows that this reasonable — of the 20 annual totals, 19 fall within these limits[12]. This provides additional confirmation

[12] An easy way to establish this is to run a single simulation of the entire period (January 1970 to December 1989) in 'imputation' mode, and then read the resulting annual totals from monthly.dat. Your results may differ slightly from those quoted, depending on the random number seed used (which will affect the imputed values). The quoted results are based on a seed of 124421. For the record, there has been no attempt to massage the results here — this was the first seed I tried!

that the models defined in logistic.def and gammamdl.def are reasonable, since annual statistics were not considered during the model-building process.

This completes the tour of the software. Any bug reports, suggestions for improvements and general comments will be most welcome — email me at `richard@stats.ucl.ac.uk`. In the meantime, good luck, and happy modelling!

Richard Chandler

Acknowledgements

The development, testing and documentation of this software has been partially supported by: the Office of Public Works, Dublin; the TSUNAMI consortium (grant A2P03); the Ministry of Agriculture, Fisheries and Food, London; and the Department for the Environment, Food and Rural Affairs (R & D projects FD2103 and FD2105). I am grateful to a number of colleagues for extremely helpful discussions over the years, in particular Valerie Isham, Howard Wheater, Vern Farewell, Steven Bate and Zhongwei Yan. Thanks must also go to the numerous people who have grappled with the software in the past and have found errors or made constructive suggestions for improvement: Claudia Annoni, Liz Baitson, Nadja Leith, Amit McCann, Nick Price, Neeraj Teeluck, Chi Yang, Yoko Yoneyama and possibly others who are omitted due to inept-, rather than ingrat-, -itude (on my part, that is).

Technical Appendix

This appendix gives, for the interested user, technical details of the models and algorithms used in this software. It is necessarily brief and collates material that may be found in Cox and Hinkley (1974); Jørgensen (1983); Liang and Zeger (1986); McCullagh and Nelder (1989); Press et al. (1992); Fahrmeir and Tutz (1994); Wei (1997) and Dobson (2001), among others. For an overview and more comprehensive reference list, see Chandler and Wheater (2002), Yan et al. (2002) and Wheater et al. (2000, Chapter 4).

A Generalised Linear Models and the exponential family of distributions

A GLM, for a $n\times 1$ vector of random variables $\mathbf{Y} = (Y_1, \ldots, Y_n)'$, is a model for the probability distribution generating $\mathbf{Y}$. Each of the Ys is considered to depend on p covariates, whose values can be assembled into a $n \times p$ matrix $\mathbf{X}$ (the (i, j)th element of $\mathbf{X}$ is the value of the jth covariate for Y_i). The distribution of $\mathbf{Y}$ has vector mean $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)'$, which is related to $\mathbf{X}$ via the relationship

$$g(\boldsymbol{\mu}) = \mathbf{X}\boldsymbol{\beta} = \boldsymbol{\eta} \ , \text{ say.} \tag{2}$$

Here, $g(.)$ is a monotonic function (the LINK FUNCTION) and $\boldsymbol{\beta}$ is a $p\times 1$ vector of coefficients (by $g(\boldsymbol{\mu})$ we mean the $n \times 1$ vector whose ith element is given by $g(\mu_i)$). The elements of $\boldsymbol{\eta}$ are called LINEAR PREDICTORS.

For computational and inferential reasons, the distribution of each Y_i is restricted to belong to the EXPONENTIAL FAMILY. For current purposes, this may be defined as the family of all distributions with densities of the form

$$f(y; \psi, \phi) = \exp\left[\frac{y\psi - b(\psi)}{a(\phi)} + c(y, \phi)\right] \ , \tag{3}$$

for some parameters ψ and ϕ, and functions $a(.)$, $b(.)$ and $c(., .)$. Many standard distributions are in this family; some examples are given in Table 10.

For a distribution expressed in this way, the mean is $\partial b/\partial\psi$ and the variance is $a(\phi)\partial^2 b/\partial\psi^2$. These expressions may be verified readily for the examples given in Table 10. For the gamma distribution, for example, we have $b(\psi) = \ln\psi$, so that $\partial b/\partial\psi = \psi^{-1} = \mu$. For the variance, we get $a(\phi)\partial^2 b/\partial\psi^2 = (-\phi^{-1})\,(-\psi^{-2}) = \mu^2/\nu$.

These results suggest that the parameter ψ determines the mean of the distribution and that, given ψ, the parameter ϕ determines the variance. For this reason, ϕ is known as a DISPERSION PARAMETER. Equation **(2)** indicates that, in a GLM, the primary interest is in the relationship between the mean and the covariates. Hence, from the perspective of exponential families, ψ is a function of the covariates and of the coefficient vector $\boldsymbol{\beta}$. The

Distribution	**Density**	$\boldsymbol{\psi}$	$\boldsymbol{\phi}$	$\mathbf{a}(\boldsymbol{\phi})$	$\mathbf{b}(\boldsymbol{\psi})$	$\mathbf{c}(\mathbf{y},\boldsymbol{\phi})$
Bernoulli, parameter p	$p^y(1-p)^{1-y}$ $(y=0,1)$	$\ln\left(\frac{p}{1-p}\right)$	1	1	$\ln(1+e^{\psi})$	0
Poisson, parameter μ	$\left(e^{-\mu}\mu^y\right)/y!$ $(y\in\mathbb{N})$	$\ln\mu$	1	1	e^{ψ}	$-\ln y!$
Normal, parameters μ and σ^2	$\frac{1}{\sigma\sqrt{2\pi}}e^{\left[-(y-\mu)^2/2\sigma^2\right]}$ $(y\in\mathbb{R})$	μ	σ^2	ϕ	$\frac{\psi^2}{2}$	$-\left(\frac{y^2}{2\phi}+\frac{1}{2}\ln 2\pi\phi\right)$
Gamma, mean μ and shape parameter ν	$\left(\frac{y\nu}{\mu}\right)^{\nu}e^{-\nu y/\mu}/y\Gamma(\nu)$ $(y>0)$	μ^{-1}	ν	$-\phi^{-1}$	$\ln(\psi)$	$\phi\ln\phi y$ $-\ln y$ $-\ln\Gamma(\phi)$

Table 10: Some distributions in the exponential family.

dispersion parameter ϕ is usually assumed constant in a GLM (for example, for a normal distribution the dispersion parameter is the variance σ^2, which is assumed constant in a classical linear regression).

A.1 Interactions

It is not uncommon for covariates to interact with each other, by which we mean that the effect of one covariate may depend on the values of others. In North-Western Europe, for example, the impact of the North Atlantic Oscillation upon rainfall is confined mainly to the winter months. Hence the coefficient associated with the NAO in a GLM should vary seasonally. This can be achieved by representing the coefficient itself as a linear combination of covariates representing seasonality. Mathematically, this is equivalent to adding an extra covariate to the model, whose value is the product of the interacting covariates. Hence interactions can be incorporated straightforwardly within the overall framework of model **(2)**.

B Maximum likelihood estimation

Given a data vector $\mathbf{y} = (y_1, \ldots, y_n)'$, assumed to be drawn from some family of distributions indexed by a parameter vector $\boldsymbol{\theta}$, we may wish to estimate this parameter vector. A standard way to do this is via maximum likelihood. The basic idea is to choose the value of $\boldsymbol{\theta}$ which allocates highest probability to the observations $\mathbf{y}$. Specifically, denote the joint density of $\mathbf{y}$ by $f(\mathbf{y}; \boldsymbol{\theta})$. Then the LIKELIHOOD for $\boldsymbol{\theta}$ given $\mathbf{y}$ is defined as

$$L(\boldsymbol{\theta}|\mathbf{y}) = f(\mathbf{y}; \boldsymbol{\theta}) \ , \tag{4}$$

and the MAXIMUM LIKELIHOOD ESTIMATE (MLE) of $\boldsymbol{\theta}$ is the value maximising this expression. Equivalently, it is the value that maximises the LOG-LIKELIHOOD $\ln L(\boldsymbol{\theta}|\mathbf{y})$, which is usually easier to compute. The MLE is usually denoted by $\hat{\boldsymbol{\theta}}$. If the observations are all independent and such that the density of the distribution generating y_i is $f_i(.; \boldsymbol{\theta})$, then their joint density is just $f(\mathbf{y}; \boldsymbol{\theta}) = \prod_i f_i(y_i; \boldsymbol{\theta})$. In this case the log-likelihood is

$$\ln L(\boldsymbol{\theta}|\mathbf{y}) = \sum_{i=1}^{n} \ln f_i(y_i; \boldsymbol{\theta}) \ . \tag{5}$$

The likelihood function can be used to form confidence intervals, by finding the set of all values of $\boldsymbol{\theta}$ for which the likelihood (or, equivalently, the log-likelihood) exceeds some threshold.

Hypothesis testing can also be carried out in a likelihood-based framework. To test whether the data are consistent with an underlying value of $\boldsymbol{\theta}_0$, we can examine the LIKELIHOOD RATIO $\Lambda = L(\hat{\boldsymbol{\theta}}|\mathbf{y})/L(\boldsymbol{\theta}_0|\mathbf{y})$, or its logarithm. By definition of $\hat{\boldsymbol{\theta}}$, $L(\hat{\boldsymbol{\theta}}|\mathbf{y}) \geq L(\boldsymbol{\theta}_0|\mathbf{y})$. Values of Λ close to 1 (i.e. values of $\ln \Lambda$ close to zero) are consistent with the null hypothesis; larger values are not.

Likelihood-based procedures have a number of appealing properties. A precise statement is lengthy and theoretical — see, for example, Cox and Hinkley (1974) for a full discussion. For practical purposes however, the most important ones can be summarised as follows:

1. For many models based on the exponential family, MLEs have the smallest mean squared error of *any* estimator.

2. For such models, likelihood-based confidence intervals are generally the shortest that can be found, at a specified confidence level.

3. The most powerful test for distinguishing between two hypotheses is based on the likelihood ratio (the NEYMAN-PEARSON LEMMA). This means that if a weak signal is present in a noisy record, a likelihood ratio test may be able to detect it when other procedures cannot.

The only disadvantage to likelihood-based inference is that it requires the probability model $f(\mathbf{y}; \boldsymbol{\theta})$ to be completely (and correctly!) specified. Results and conclusions will

depend on this specification, so we need to ensure that the model structure is realistic. For example, if the observations arise as a time series then they are likely to be dependent so that **(5)** is incorrect. However, a standard factorisation of the joint density allows us to write the log-likelihood as

$$\ln L(\boldsymbol{\theta}|\mathbf{y}) = \sum_{i=1}^{n} \ln f_i\left(y_i|\mathcal{H}_i;\boldsymbol{\theta}_i\right) \ ,$$

where here $f_i\left(y_i|\mathcal{H}_i;\boldsymbol{\theta}_i\right)$ denotes the density of the ith observation given its history $\mathcal{H}_i$ say. This has the same form as **(5)**, and hence we can proceed as usual providing the history is adequately accounted for within the model. It is this result that motivates the inclusion of previous days' values into the GLMs for daily climate time series.

In GLMs, where the observations all come from distributions within the exponential family with a common dispersion parameter ϕ, the log-likelihood for $\boldsymbol{\beta}$ and ϕ given n independent observations is, from **(3)** and **(5)**,

$$\ln L\left(\boldsymbol{\beta},\phi|\mathbf{y}\right) = \sum_{i=1}^{n} \frac{y_i\psi_i - b(\psi_i)}{a(\phi)} + c(y,\phi) \ . \tag{6}$$

The coefficient vector $\boldsymbol{\beta}$ enters the right-hand side here through the ψ_i terms, as described in the previous section.

In well-behaved problems, the value of $\boldsymbol{\beta}$ maximising the log-likelihood satisfies the p SCORE EQUATIONS

$$\frac{\partial \ln L}{\partial \beta_j} = \frac{1}{a(\phi)} \sum_{i=1}^{n} \frac{\partial}{\partial \beta_j}\left[y_i\psi_i - b(\psi_i)\right] \ (= U_j, \text{ say}) \ = 0 \qquad (j = 1,\ldots,p) \ , \tag{7}$$

whose solution clearly does not depend on ϕ.

B.1 Likelihood ratio tests

In the previous section, we noted that hypothesis testing can be carried out using likelihood ratios. We now summarise the procedure. Specifically, we suppose that the linear predictors in **(2)** have the form

$$\eta_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip} \ ,$$

and we wish to test the null hypothesis $H_0 : \beta_{q+1} = \beta_{q+2} = \ldots = \beta_p = 0$, for some $q < p$. The likelihood ratio test procedure in this case is:

1. Fit the REDUCED MODEL (i.e. the model containing the first q predictors) using Maximum Likelihood; denote the resulting log-likelihood by $\ln L_0$.

2. Fit the model containing all of the xs, and denote the resulting log-likelihood by $\ln L_1$. This will never be less than $\ln L_0$.

3. Calculate the likelihood ratio test statistic $2 \ln \Lambda = 2 (\ln L_1 - \ln L_0)$. If this is larger than the appropriate percentage point of a χ^2 distribution with $(p - q)$ degrees of freedom, reject the null hypothesis; otherwise accept it.

There is a potential complication here for GLMs that involve an unknown dispersion parameter ϕ. This is because the log-likelihood **(6)** depends on ϕ, whence the likelihood ratio does also (although the $c(y, \phi)$ term cancels in the ratio). If ϕ is unknown, we can of course estimate it. However, in general we will obtain different estimates from each of the models under consideration and this may affect our inference. Most standard software packages work around this problem by reporting the DEVIANCE (see below) rather than the log-likelihood. This software reports both. For models involving a dispersion parameter ϕ, the log-likelihoods are maximised with respect to both $\boldsymbol{\beta}$ and ϕ — thus the theory above is directly applicable.

B.2 Deviance

For the reasons given above, many software packages output deviances for GLMs rather than log-likelihoods. The deviance for a model is defined as

$$D = 2a(\phi) (\ln L_F - \ln L) \ ,$$

where L_F is the likelihood for a FULL MODEL in which we set $\mu_i = y_i$ for each i, and L is the likelihood for the model under consideration. Comparing this with **(6)**, we see that the deviance does not depend on ϕ.

The deviance is equivalent to the residual sum of squares in a linear regression (and is never negative) — in fact, for a GLM based on the normal distribution with constant variances, the deviance *is* the residual sum of squares. For this reason, procedures such as analysis of variance (which describes how different predictors in a linear regression account for the variability in the Ys) are generalised to 'analysis of deviance' in GLMs. F tests can be used to compare models, as in the standard regression case.

C Numerical algorithms for GLMs

C.1 Iterative weighted least squares

We now address the problem of calculating the maximum likelihood estimate for a GLM. In practice, the score equations **(7)** must be solved numerically in all but the simplest cases. Assembling all p equations into vector form, we seek the solution of

$$\mathbf{U} (\boldsymbol{\beta}) = \mathbf{0} \tag{8}$$

where $\mathbf{U} (\boldsymbol{\beta}) = (U_1, \ldots, U_p)'$ is the SCORE VECTOR of log-likelihood derivatives. $\mathbf{U}(.)$ is typically a nonlinear function of $\boldsymbol{\beta}$.

To solve equations of the form **(8)**, the Newton-Raphson algorithm may be used: start with an initial guess at the solution, $\boldsymbol{\beta}^{(0)}$ say, and then successively calculate

$$\boldsymbol{\beta}^{(t)} = \boldsymbol{\beta}^{(t-1)} - \left[\frac{\partial \mathbf{U}}{\partial \boldsymbol{\beta}}_{|\boldsymbol{\beta}^{(t-1)}}\right]^{-1} \mathbf{U}\left(\boldsymbol{\beta}^{(t-1)}\right) \tag{9}$$

until convergence is achieved. $\partial \mathbf{U}/\partial \boldsymbol{\beta}$ here is the $p \times p$ matrix of second derivatives of the log-likelihood with respect to $\boldsymbol{\beta}$. Note that, since the likelihood for a given $\boldsymbol{\beta}$ depends on the data, it should be regarded as the realised value of a random variable, as should its derivatives. Hence it is meaningful to consider the expected value of likelihood derivatives. In particular, the quantity $\mathbf{I}(\boldsymbol{\beta}) = -E_{\boldsymbol{\beta}}[\partial \mathbf{U}/\partial \boldsymbol{\beta}]$ is called the INFORMATION MATRIX for $\boldsymbol{\beta}$.

The reason for introducing this concept is that, when finding the maximum likelihood estimate of $\boldsymbol{\beta}$ in a GLM, it is common to replace the matrix $\partial \mathbf{U}/\partial \boldsymbol{\beta}$ in **(9)** by its expected value. The resulting maximisation algorithm is called the METHOD OF SCORING: the iterative scheme is

$$\boldsymbol{\beta}^{(t)} = \boldsymbol{\beta}^{(t-1)} + \left[\mathbf{I}\left(\boldsymbol{\beta}^{(t-1)}\right)\right]^{-1} \mathbf{U}\left(\boldsymbol{\beta}^{(t-1)}\right) \ . \tag{10}$$

With a log-likelihood of the form **(6)**, the derivative with respect to β_j is

$$U_j = \frac{\partial \ln L}{\partial \beta_j} = \frac{1}{a(\phi)} \sum_{i=1}^{n} \frac{\partial \ell_i}{\partial \beta_j} \ ,$$

where ℓ_i is a contribution from the ith observation. The chain rule gives

$$\frac{\partial \ell_i}{\partial \beta_j} = \frac{\partial \ell_i}{\partial \psi_i} \frac{\partial \psi_i}{\partial \mu_i} \frac{\partial \mu_i}{\partial \eta_i} \frac{\partial \eta_i}{\partial \beta_j} \ ,$$

where μ_i and η_i are the mean and linear predictor for the ith case (recall equation **(2)**). Dealing with each of these in turn, and referring to the properties of the exponential family on page 62:

1. $\partial \ell_i/\partial \psi_i = y_i - \partial b/\partial \psi_i = y_i - \mu_i$.
2. $\partial \psi_i/\partial \mu_i = [\partial \mu_i/\partial \psi_i]^{-1}$. Now $\mu_i = \partial b/\partial \psi_i$, so $\partial \mu_i/\partial \psi_i = \partial^2 b/\partial \psi_i^2$. But $\mathrm{Var}(Y_i) = a(\phi)\partial^2 b/\partial \psi_i^2$. Hence $\partial^2 b/\partial \psi_i^2 = \mathrm{Var}(Y_i)/a(\phi)$, and $\partial \psi_i/\partial \mu_i = a(\phi)/\mathrm{Var}(Y_i) = a(\phi)/V_i$, say.
3. $\partial \mu_i/\partial \eta_i$ depends on the particular link function used in equation **(2)**.
4. $\partial \eta_i/\partial \beta_j = x_{i,j}$ (the value of the jth covariate for the ith case).

Putting these results together and summing over all cases in the dataset, we find that the jth element of the score vector is given by

$$U_j = \sum_{i=1}^{n} \left[\frac{y_i - \mu_i}{V_i}\left(\frac{\partial \mu_i}{\partial \eta_i}\right)\left(\frac{\partial \eta_i}{\partial \beta_j}\right)\right] = \sum_{i=1}^{n} \left[\frac{y_i - \mu_i}{V_i}\left(\frac{\partial \mu_i}{\partial \eta_i}\right) x_{i,j}\right] \ . \tag{11}$$

Calculation of the information matrix $\mathbf{I}(\boldsymbol{\beta})$ is simplified by the fact that its (j,k)th element is equal to $E[U_j U_k]$ providing the underlying model is correct — in particular, providing the independence assumptions of the model hold. It can also be shown that the scores all have mean zero, whence $\mathbf{I}(\boldsymbol{\beta})$ has an alternative representation as the covariance matrix of the score vector. These are standard results and will not be proved here (the former can be derived using straightforward algebra from representation **(11)** of the scores). However they enable us to deduce, after a little manipulation involving **(11)**, that if the model is correct the (j,k)th element of the information matrix is

$$\sum_{i=1}^{n}\left[\frac{1}{V_i}\left(\frac{\partial\eta_i}{\partial\beta_j}\right)\left(\frac{\partial\eta_i}{\partial\beta_k}\right)\left(\frac{\partial\mu_i}{\partial\eta_i}\right)^2\right]=\sum_{i=1}^{n}\left[\frac{x_{i,j}x_{i,k}}{V_i}\left(\frac{\partial\mu_i}{\partial\eta_i}\right)^2\right] \tag{12}$$

In matrix form then, we can write

$$\mathbf{I}(\boldsymbol{\beta})=\mathbf{X}'\mathbf{W}\mathbf{X}\ , \tag{13}$$

where $\mathbf{W}$ is a diagonal $n\times n$ matrix with elements $w_{ii}=(\partial\mu_i/\partial\eta_i)^2/V_i$ (for computational purposes, in fact it is convenient to use $a(\phi)\partial\mu_i/\partial\psi_i$ in place of V_i here, since then $a(\phi)$ appears as a constant that can be omitted from the iterative scheme **(15)** below). All of these quantities are functions of $\boldsymbol{\beta}$.

This matrix representation can be used in the iterative scoring algorithm **(10)**: multiplying both sides of that equation by $\mathbf{I}\left(\boldsymbol{\beta}^{(t-1)}\right)$ and substituting **(13)** for $\mathbf{I}\left(\boldsymbol{\beta}^{(t-1)}\right)$, we find

$$\mathbf{X}'\mathbf{W}^{(t-1)}\mathbf{X}\boldsymbol{\beta}^{(t)}=\mathbf{X}'\mathbf{W}^{(t-1)}\mathbf{X}\boldsymbol{\beta}^{(t-1)}+\mathbf{U}\left(\boldsymbol{\beta}^{(t-1)}\right)\ . \tag{14}$$

Noting that $\mathbf{X}\boldsymbol{\beta}^{(t-1)}=\boldsymbol{\eta}^{(t-1)}$, the vector of linear predictors at iteration $t-1$, and that the score vector can itself be written as a vector product involving the matrix $\mathbf{X}'\mathbf{W}$ (from **(11)**), the scoring algorithm can finally be written in matrix form as

$$\left[\mathbf{X}'\mathbf{W}^{(t-1)}\mathbf{X}\right]\boldsymbol{\beta}^{(t)}=\mathbf{X}'\mathbf{W}^{(t-1)}\mathbf{z}^{(t-1)}\ , \tag{15}$$

where $\mathbf{z}^{(\mathbf{t-1})}$ is an $n\times 1$ vector whose ith element is

$$z_i^{(t-1)}=\eta_i^{(t-1)}+\left(y_i-\mu_i^{(t-1)}\right)\left(\frac{\partial\eta}{\partial\mu}_{|\mu_i^{(t-1)}}\right)\ .$$

Equation **(15)** in fact gives the solution of the weighted least-squares regression of $\mathbf{z}^{(t-1)}$ upon $\mathbf{X}$, with weights contained in the diagonal elements of $\mathbf{W}^{(t-1)}$. The need for iteration arises because $\mathbf{z}$ and $\mathbf{W}$ both depend, in general, upon $\boldsymbol{\beta}$. Expressed in this form, the algorithm for fitting GLMs is referred to as ITERATIVE WEIGHTED LEAST SQUARES (IWLS). An additional advantage is that for large samples, the covariance matrix of the final parameter estimates is $[\mathbf{X}'\mathbf{W}\mathbf{X}]^{-1}$, which emerges as a by-product of the fitting procedure. This can be used, for example, to derive standard errors for the parameter estimates. See Appendix C.2 below for more details on this.

C.1.1 The information matrix versus the Hessian

In what follows, it will be useful to understand the implications of replacing the Hessian matrix of the log-likelihood by its expected value in the algorithm above. Differentiating **(11)** with respect to β_k, we find that the (j,k)th element of the Hessian matrix is

$$\begin{aligned}\frac{\partial U_j}{\partial \beta_k} &= \sum_{i=1}^{n}\left\{(y_i-\mu_i)\frac{\partial}{\partial\beta_k}\left[\frac{x_{i,j}}{V_i}\left(\frac{\partial\mu_i}{\partial\eta_i}\right)\right]-\frac{\partial\mu_i}{\partial\beta_k}\left[\frac{x_{i,j}}{V_i}\left(\frac{\partial\mu_i}{\partial\eta_i}\right)\right]\right\}\\ &= \sum_{i=1}^{n}\left\{(y_i-\mu_i)\frac{\partial}{\partial\beta_k}\left[\frac{x_{i,j}}{V_i}\left(\frac{\partial\mu_i}{\partial\eta_i}\right)\right]-\frac{x_{i,j}x_{i,k}}{V_i}\left(\frac{\partial\mu_i}{\partial\eta_i}\right)^2\right\}\ . \end{aligned} \tag{16}$$

Note that, if we take expectations, the first term here vanishes and the expression reduces (with a change of sign), to the (j,k)th element of the information matrix at **(12)**. The first term will also vanish if the term in square brackets is invariant with respect to β_k, since then the derivative will be identically zero. If this is the case then the Hessian matrix will always be exactly equal to its expected value. Such invariance will occur when $\partial\mu_i/\partial\eta_i \propto V_i$ i.e. for a particular choice of the link function $g(.)$ in **(2)**. This choice is called the CANONICAL LINK. From item 2 in the list preceding **(11)** (page 67), we see that V_i may be defined as $a(\phi)\times\partial\mu_i/\partial\psi_i$. Hence the canonical link function for any GLM is $g(\mu_i)=\psi(\mu_i)$. Refer to Table 10 for the canonical links $\psi(.)$ in some standard distributions.

If the information is *not* equal to the negative Hessian matrix, it is natural to question the applicability of the scoring algorithm. We now give a heuristic justification for its use. From **(9)** and **(10)**, it is clear that the algorithm will provide a good approximation to the Newton-Raphson iterative scheme (and hence should work in well-behaved problems) providing $[\partial\mathbf{U}/\partial\boldsymbol{\beta}]^{-1}+\mathbf{I}^{-1}(\boldsymbol{\beta})$ is small. Now, given n independent observations, the log-likelihood and its derivatives are sums of n terms and therefore deviate from their expectations by quantities that are $O_p\left(n^{1/2}\right)$. Hence, if the model is correct then in some neighbourhood of the true parameter vector we can write

$$-\frac{\partial\mathbf{U}}{\partial\boldsymbol{\beta}}=\mathbf{I}(\boldsymbol{\beta})+\mathbf{E}\ , \tag{17}$$

say, where $\mathbf{E}$ is a matrix whose elements are $O_p\left(n^{1/2}\right)$. Writing $\mathbf{1}$ for an identity matrix, we therefore have

$$-\left[\frac{\partial\mathbf{U}}{\partial\boldsymbol{\beta}}\right]^{-1}=\mathbf{I}^{-1}(\boldsymbol{\beta})\left[\mathbf{1}+\mathbf{I}^{-1}(\boldsymbol{\beta})\mathbf{E}\right]^{-1}\ .$$

Now the elements of $\mathbf{I}^{-1}(\boldsymbol{\beta})$ are $O_p\left(n^{-1}\right)$, so those of $\mathbf{I}^{-1}(\boldsymbol{\beta})\mathbf{E}$ are $O_p\left(n^{-1/2}\right)$. Hence, for large n we have

$$-\left[\frac{\partial\mathbf{U}}{\partial\boldsymbol{\beta}}\right]^{-1}\approx\mathbf{I}^{-1}(\boldsymbol{\beta})\left[\mathbf{1}-\mathbf{I}^{-1}(\boldsymbol{\beta})\mathbf{E}\right]=\mathbf{I}^{-1}(\boldsymbol{\beta})+\mathbf{M}\ ,$$

where now $\mathbf{M}$ is a matrix whose elements are $O_p\left(n^{-3/2}\right)$ (in contrast with those of $\mathbf{I}^{-1}(\boldsymbol{\beta})$, which are $O_p\left(n^{-1}\right)$). Hence, for sufficiently large samples, replacing $\partial\mathbf{U}/\partial\boldsymbol{\beta}$ with its expected

value should not cause numerical problems *providing* the search is restricted to an appropriate neighbourhood of the true parameter vector. This proviso arises because in **(17)**, $\mathbf{I}(\boldsymbol{\beta})$ is an expected value computed as though $\boldsymbol{\beta}$ is the true parameter vector. If $\mathbf{I}(\boldsymbol{\beta})$ varies substantially with $\boldsymbol{\beta}$ then we may expect problems away from the true value (or equivalently, away from the MLE, since this also deviates from the true value by $O_p(n^{1/2})$).

C.2 Covariance matrix of the estimates

Having demonstrated how maximum likelihood parameter estimates may be obtained, we now consider their covariance matrix. From **(15)**, it is clear that the MLE satisfies

$$\hat{\boldsymbol{\beta}} = [\mathbf{X}'\mathbf{W}\mathbf{X}]^{-1}\,\mathbf{X}'\mathbf{W}\mathbf{z}\ , \tag{18}$$

where all quantities are evaluated at $\hat{\boldsymbol{\beta}}$. The required covariance matrix may be estimated by considering the covariance of the right hand side here as a function of $\boldsymbol{\beta}$, and evaluating it at $\hat{\boldsymbol{\beta}}$. Note that as a function of $\boldsymbol{\beta}$, $[\mathbf{X}'\mathbf{W}\mathbf{X}]^{-1}$ is non-random. Note also the standard result that if $\mathbf{Y}$ is a vector of random variables and $\mathbf{A}$ a matrix such that $\mathbf{A}\mathbf{Y}$ is defined, $\mathrm{Var}(\mathbf{A}\mathbf{Y}) = \mathbf{A}\mathrm{Var}(Y)\mathbf{A}'$. Hence the covariance matrix of the right hand side of **(18)** is

$$[\mathbf{X}'\mathbf{W}\mathbf{X}]^{-1}\,\mathrm{Var}\,(\mathbf{X}'\mathbf{W}\mathbf{z})\,[\mathbf{X}'\mathbf{W}\mathbf{X}]^{-1}\ , \tag{19}$$

since $[\mathbf{X}'\mathbf{W}\mathbf{X}]^{-1}$ is a symmetric matrix.

The next step is to note that the covariance matrix of $\mathbf{X}'\mathbf{W}\mathbf{z}$ is, for fixed $\boldsymbol{\beta}$, the same as that of the score vector $\mathbf{U}$. To see this, equate the right hand sides of equations **(14)** and **(15)** to obtain $\mathbf{X}'\mathbf{W}\mathbf{X}\boldsymbol{\beta} + \mathbf{U} = \mathbf{X}'\mathbf{W}\mathbf{z}$ — this relationship defines $\mathbf{z}$ for any $\boldsymbol{\beta}$. But for fixed $\boldsymbol{\beta}$, $\mathbf{X}'\mathbf{W}\mathbf{X}\boldsymbol{\beta}$ is non-random, and hence can be ignored in variance calculations. The estimated covariance matrix of $\hat{\boldsymbol{\beta}}$ is therefore

$$\hat{\mathrm{Var}}\left(\hat{\boldsymbol{\beta}}\right) = [\mathbf{X}'\mathbf{W}\mathbf{X}]^{-1}\,\mathrm{Var}\,(\mathbf{U})\,[\mathbf{X}'\mathbf{W}\mathbf{X}]^{-1}\ . \tag{20}$$

If observations are independent, we have seen (page 68) that $\mathrm{Var}\,(\mathbf{U}) = \mathbf{X}'\mathbf{W}\mathbf{X}$. In this case, **(20)** reduces to $[\mathbf{X}'\mathbf{W}\mathbf{X}]^{-1}$. This is the 'default' formula used by the software to calculate standard errors, if no spatial dependence structure is specified for a given model.

More frequently however, in climatological applications observations are obtained from a network of sites. In this case, inter-site dependence means that observations from different sites on the same day cannot be regarded as independent; however, in general there is nothing to stop us from estimating $\boldsymbol{\beta}$ by maximising the 'independence' log-likelihood. In this case, tests based on likelihood ratios and deviance must be modified to account for the inter-site dependence — these modifications can be complex. However, tests based on the covariance matrix **(20)** can be modified straightforwardly, providing we can find an easily computable estimate of $\mathrm{Var}\,(\mathbf{U})$. Since $\mathbf{U}$ is a sum over all observations in the database, we can write

$$\mathbf{U} = \sum_{t=1}^{T}\sum_{s=1}^{S_t}\mathbf{U}_{st}\ ,$$

where $\mathbf{U}_{st}$ denotes the contribution from site s on day t. Hence

$$\mathrm{Var}\,(\mathbf{U}) = \mathrm{Var}\left(\sum_{t=1}^{T}\sum_{s=1}^{S_t}\mathbf{U}_{st}\right) = \sum_{t=1}^{T}\mathrm{Var}\left(\sum_{s=1}^{S_t}\mathbf{U}_{st}\right) \ ,$$

assuming that score contributions from different days are uncorrelated (which will be the case so long as the model contains an adequate representation of temporal dependence — see Appendix B above, and also note that from equation **(11)**, contributions to the score vector are essentially weighted residuals from the fitted model). Now, since each contribution to the score vector has expected value zero (page 68), we have

$$\sum_{t=1}^{T}\mathrm{Var}\left(\sum_{s=1}^{S_t}\mathbf{U}_{st}\right) = \sum_{t=1}^{T}\mathrm{E}\left[\left(\sum_{s=1}^{S_t}\mathbf{U}_{st}\right)\left(\sum_{s=1}^{S_t}\mathbf{U}_{st}\right)'\right] \ ,$$

where $\mathrm{E}\,[.]$ denotes expectation. For large T, we can use the observed values of the square-bracketed terms to estimate their expectations, and obtain the estimator

$$\hat{\mathrm{Var}}\,(\mathbf{U}) = \sum_{t=1}^{T}\left(\sum_{s=1}^{S_t}\mathbf{U}_{st}\right)\left(\sum_{s=1}^{S_t}\mathbf{U}_{st}\right)' \ , \tag{21}$$

which is easily computable at little extra cost, once $\boldsymbol{\beta}$ has been found. The software uses this estimator of $\mathrm{Var}\,(\mathbf{U})$, in conjunction with **(20)**, to compute standard errors whenever the user specifies a spatial dependence structure in a model.

C.3 Dependence-adjusted likelihood ratio

Appendix B.1 above presented the likelihood ratio test procedure for discriminating between two nested models. The theory underlying this procedure assumes that the observations are conditionally independent given the covariates. Specifically, it relies upon the standard identity

$$\mathrm{Var}\,[\mathbf{U}\,(\boldsymbol{\beta}_0)] = -\mathrm{E}\,[\mathbf{H}\,(\boldsymbol{\beta}_0)] \ ,$$

where $\boldsymbol{\beta}_0$ is the underlying true value of $\boldsymbol{\beta}$ (this assumes, of course, that the model is correctly specified so that it is meaningful to speak of such a 'true' value) and $\mathbf{H}$ is the Hessian matrix of second derivatives of the 'independence' log-likelihood function **(6)**. Inter-site dependence affects the covariance matrix of $\mathbf{U}\,(\boldsymbol{\beta}_0)$, as detailed above; however, the Hessian of the independence log-likelihood is unchanged.

This observation has been used by Bate (2004) to develop an adjustment to the likelihood ratio test in the presence of inter-site dependence. The idea is to construct a modified inference function that is maximised at the 'independence' MLE $\hat{\boldsymbol{\beta}}$, but with Hessian $-\mathcal{R}^{-1}$, where $\mathcal{R}$ is the 'robust' covariance estimate obtained by combining **(21)** with **(20)**. Denoting by ℓ_{IND} the 'independence' log-likelihood **(6)**, this modified inference function is defined as

$$\ell_{ADJ}(\boldsymbol{\beta}) = \ell_{IND}\,(\boldsymbol{\beta}^*) \tag{22}$$

for a linear transformation

$$\boldsymbol{\beta}^* = \hat{\boldsymbol{\beta}} + \mathbf{A}\left(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}\right). \tag{23}$$

By equating Taylor series expansions about $\hat{\boldsymbol{\beta}}$, it can be shown that the matrix $\mathbf{A}$ satisfies

$$\mathbf{A} = \left\{\left[\mathcal{N}^{-1}\right]^{1/2}\right\}^{-1}\left[\mathcal{R}^{-1}\right]^{1/2}, \tag{24}$$

where $\mathcal{N}$ is the 'naive' covariance matrix $[\mathbf{X}'\mathbf{W}\mathbf{X}]^{-1}$. The matrix square roots in **(24)** are not uniquely defined; however, the second-order Taylor expansion underlying the adjustment *is* unique. The software uses Cholesky square roots.

Notice that the adjustment here is model-dependent, and hence cannot be used for 'global' correction of the log-likelihood. It can be used, however, to compare two models when one is a special case of the other. In this case, the missing terms in the simpler model can be treated as having zero coefficients and the adjusted log-likelihood $\ell_{ADJ}(\boldsymbol{\beta})$ can be treated as though it is the true log-likelihood function. For example, a likelihood ratio test can be performed by maximising $\ell_{ADJ}(\boldsymbol{\beta})$ under both the reduced and full models, to obtain estimates $\hat{\boldsymbol{\beta}}_0$ and $\hat{\boldsymbol{\beta}}_1$, say: then the adjusted likelihood ratio statistic

$$2\left[\ell_{ADJ}(\hat{\boldsymbol{\beta}}_1) - \ell_{ADJ}(\hat{\boldsymbol{\beta}}_0)\right] \tag{25}$$

can be compared with percentage points of the appropriate χ^2 distribution in the usual way. To maximise $\ell_{ADJ}(\boldsymbol{\beta})$ under the full model is straightforward: it is achieved by maximising the 'independence' log-likelihood using the usual algorithms. Maximisation under the reduced model is more difficult; moreover, in typical applications the reduced model will already have been fitted using the 'independence' log-likelihood (to yield an estimate $\tilde{\boldsymbol{\beta}}_0$, say), and the user may wish to use this existing estimate rather than carry out an extra, computationally expensive maximisation. To compute **(25)** it is therefore tempting to consider using $\tilde{\boldsymbol{\beta}}_0$ in place of $\hat{\boldsymbol{\beta}}_0$. However, since by definition $\ell_{ADJ}\left(\hat{\boldsymbol{\beta}}_0\right) \geq \ell_{ADJ}\left(\tilde{\boldsymbol{\beta}}_0\right)$, such a procedure will inflate the values of the test statistic. This inflation can be corrected using a secondary adjustment: specifically, an alternative to **(25)** is

$$2c\left[\ell_{ADJ}(\hat{\boldsymbol{\beta}}_1) - \ell_{ADJ}(\tilde{\boldsymbol{\beta}}_0)\right], \tag{26}$$

for some scaling factor $c \in (0, 1)$. The value of c can be computed explicitly if (a) $\ell_{ADJ}(\cdot)$ is quadratic in the neighbourhood of interest (b) the reduced model is defined in terms of a linear constraint on the parameters of the full model **REFS!!!** . In the applications for which this software is designed, both criteria are likely to be satisfied.

When the software fits a model, it checks to see whether the starting values for any coefficients in the model definition file are zero. If so, it assumes that the corresponding covariates are being added to a previous-fitted, reduced model. The reported 'Dependence-adjusted LR statistic' is then calculated according to **(26)**; the value of the scaling factor c is specific to this particular choice of reduced model, and is reported in the output. If, however, none of the initial values are zero, the software merely computes the difference between initial and final dependence-adjusted log-likelihoods, using **(25)**.

C.4 Nonlinear transformations of the covariates

The above theory provides an algorithm for finding the MLE of the parameter vector $\boldsymbol{\beta}$ in a model of the form **(2)**. In some situations, however, we may wish to represent one or more covariates themselves as nonlinear transformations of some underlying quantity. For example, some of the 'year' effects in Table 4 involve parameters that are, in general unknown. Similarly, there are unknown parameters involved in some of the weighting schemes in Table 6. The general setup here is that $x_{i,j}$ can be written as $f(x^*_{i,j};\theta)$ where $x^*_{i,j}$ is the value of the underlying quantity for the ith case in the dataset, and $f(.;.)$ is a function of known parametric form. This scenario is non-standard within GLMs. For simplicity, to start with we assume that $\boldsymbol{\theta}$ is a scalar and write θ.

Although the estimation of nonlinear transformations is nonstandard, the MLE of θ can in principle be found via a straightforward extension of the IWLS algorithm above. The key point to note is the presence, before simplification in equations **(11)** and **(12)**, of the quantities $\partial\eta_i/\partial\beta_j$ and $\partial\eta_i/\partial\beta_k$. In the standard GLM case, these partial derivatives are equal to $x_{i,j}$ and $x_{j,k}$ respectively — it is the fact that these are elements of the matrix $\mathbf{X}$ that enables us to write the scoring algorithm in the IWLS form **(15)**. Following the previous argument up to **(11)**, the score for θ is just

$$U_\theta = \sum_{i=1}^{n} \left[\frac{y_i - \mu_i}{V_i} \left(\frac{\partial \mu_i}{\partial \eta_i} \right) \left(\frac{\partial \eta_i}{\partial \theta} \right) \right] , \tag{27}$$

and the information matrix for the augmented parameter vector $(\boldsymbol{\beta}\ \theta)'$ can be constructed as before via covariances of the scores. A matrix representation of the resulting scoring algorithm can therefore be obtained as

$$\mathbf{X}^{*\prime}\mathbf{W}^{(t-1)}\mathbf{X}^* \begin{pmatrix} \boldsymbol{\beta} \\ \theta \end{pmatrix}^{(t)} = \mathbf{X}^{*\prime}\mathbf{W}^{(t-1)}\mathbf{z}^{(t-1)} , \tag{28}$$

where $\mathbf{X}^*$ is now the $n \times (p+1)$ matrix obtained by adding an extra column to $\mathbf{X}$, whose elements are the values of $\partial\eta/\partial\theta$ for each case in the dataset. By analogy with **(14)**, the definition of $\mathbf{z}^{(t-1)}$ must also change slightly — its ith element is now

$$z_i^{(t-1)} = \eta_i^{(t-1)} + \theta^{(t-1)} \frac{\partial \eta_i^{(t-1)}}{\partial \theta^{(t-1)}} + \left(y_i - \mu_i^{(t-1)} \right) \left(\frac{\partial \eta}{\partial \mu}_{|\mu_i^{(t-1)}} \right) .$$

Conceptually at least then, the extension of the usual IWLS algorithm to deal with nonlinear transformations of covariates is straightforward: simply augment the original p covariates with a set of 'dummy' covariates corresponding to derivatives of the linear predictor with respect the parameters of interest, and proceed as normal. Of course, since some elements of $\mathbf{X}$ now depend on θ, they will need to be updated after each iteration — this increases the computational load somewhat, particularly for large datasets. A further consequence is that the information matrix $\mathbf{X}^{*\prime}\mathbf{W}^{(t-1)}\mathbf{X}^*$ may vary rapidly for some nonlinear

parameterisations. In this case, from the discussion in the previous section, we may expect convergence problems with the scoring algorithm.

It turns out that a small modification of the algorithm will guarantee convergence to a maximum. To introduce this, let $\ell(.)$ be an arbitrary log-likelihood function for a parameter vector $\boldsymbol{\theta}$, and let $\mathbf{U}(.)$ be the corresponding score vector. Then, providing $\ell(.)$ is continuous and differentiable in the neighbourhood of $\boldsymbol{\theta}$, there exists a radius δ (which may be small) such that, for all vectors $\boldsymbol{\epsilon}$ with $|\boldsymbol{\epsilon}| < \delta$,

$$\ell(\boldsymbol{\theta} + \boldsymbol{\epsilon}) = \ell(\boldsymbol{\theta}) + \lambda \boldsymbol{\epsilon}' \mathbf{U}(\boldsymbol{\theta})$$

for some $\lambda > 0$. In particular, this holds if we set $\boldsymbol{\epsilon} = \mathbf{D}\mathbf{U}(\boldsymbol{\theta})$ where $\mathbf{D}$ is a sufficiently small, symmetric, positive definite matrix. In this case,

$$\ell(\boldsymbol{\theta} + \boldsymbol{\epsilon}) = \ell(\boldsymbol{\theta}) + \lambda \mathbf{U}'(\boldsymbol{\theta}) \mathbf{D}\mathbf{U}(\boldsymbol{\theta}) \ ,$$

which cannot be less than $\ell(\boldsymbol{\theta})$ since $\lambda > 0$ and $\mathbf{D}$ is positive definite.

The relevance of this result is as follows: a step of the unmodified scoring algorithm changes the current value of the parameter vector by $[\mathbf{X}^{*\prime}\mathbf{W}\mathbf{X}^{*}]^{-1}\mathbf{U}(\boldsymbol{\theta})$, in an obvious notation (compare with **(14)** to verify this). This is of the form $\mathbf{D}\mathbf{U}(\boldsymbol{\theta})$, where $\mathbf{D}$ is symmetric and positive definite. Therefore, from the result above there exists a $\rho > 0$ such that changing the current value of the parameter vector by $\rho[\mathbf{X}^{*\prime}\mathbf{W}\mathbf{X}^{*}]^{-1}\mathbf{U}(\boldsymbol{\theta})$ is guaranteed to increase the log-likelihood. The implication is that, if the log-likelihood decreases during any iteration of the unmodified algorithm, the step size was too large and we should reduce it.

To stabilise the algorithm therefore, if any step reduces the log-likelihood we successively reduce the step size until an increase is obtained. The literature recommends the use of rather simple reduction schemes (e.g. successively halving the step size). However, experience shows that in particularly ill-behaved problems such schemes can require many successive reductions before an increase is found. We therefore halve the step size once and then try and locate the optimal step size by fitting a quadratic to the log-likelihood surface along the search direction. This modification guarantees eventual convergence to a maximum of the likelihood surface, although when nonlinear parametrisations are involved this maximum may be local rather than global (a trivial example of this, which can be overcome by suitable reparametrisation, arises when estimating the phase of a long-term cycle with a fixed frequency — the likelihood is then periodic with respect to the phase).

C.4.1 Miscellaneous details

A couple of points are worth noting with respect to this algorithm, and its use in the software provided here:

1. $\partial\eta/\partial\theta$ contains contributions not just from the underlying covariate itself, but also from interactions with other covariates. In most cases, this is straightforward. Care needs to be taken, however, if two interacting covariates (the jth and kth, say) are both

transformations involving the *same* parameter θ. In this case (which may not always correspond to a sensible model!) the contribution to $\partial\eta/\partial\theta$ from the interaction term $x_{i,j}x_{i,k}$ is

$$x_{i,j}\frac{\partial x_{i,k}}{\partial\theta} + x_{i,k}\frac{\partial x_{i,j}}{\partial\theta} \ .$$

The software considers contributions $\partial\eta/\partial\theta$ involving the derivatives of each covariate in turn. In the example above, the contribution involving $\partial x_{i,j}/\partial\theta$ will pick up the second term, while that involving $\partial x_{i,k}/\partial\theta$ will pick up the first. One of those magical mathematical fortuities that brightens up a programmer's life.

2. When estimating parameters in schemes for computing weighted averages (Table 6), the weight attached to site r when computing an averages for site s is of the form

$$w_{r,s} = \left(w^*_{r,s}\right) / \left(\sum_j w^*_{j,s}\right) \ , \tag{29}$$

where the sum is over all sites contributing to the average. Here, $w^*_{j,s}$ is the non-normalised weight, whose functional form is known. The resulting covariate then takes the form

$$\sum_r w_{r,s} f\left(Y^{(r)}_{t-k}\right)$$

for the 'average of transformed values' case (codes 11–99 in Table 5), and

$$f\left(\sum_r w_{r,s} Y^{(r)}_{t-k}\right)$$

for the 'transformation of averages' case (codes 111–199). For the two different cases, the contributions to $\partial\eta/\partial\eta$ are

$$\sum_r \frac{\partial w_{r,s}}{\partial\theta} f\left(Y^{(r)}_{t-k}\right) \qquad \text{and} \qquad f'\left(\sum_r \frac{\partial w_{r,s}}{\partial\theta} Y^{(r)}_{t-k}\right) \sum_r \frac{\partial w_{r,s}}{\partial\theta} Y^{(r)}_{t-k} \ ,$$

respectively. Both expressions require $\partial w_{r,s}/\partial\theta$ which, from **(29)**, is given by

$$\frac{\partial w_{r,s}}{\partial\theta} = \left[\sum_j w^*_{j,s}\right]^{-2} \left(\frac{\partial w^*_{r,s}}{\partial\theta}\sum_j w^*_{j,s} - w^*_{r,s}\sum_j \frac{\partial w^*_{j,s}}{\partial\theta}\right) \ .$$

Note, incidentally, that if $f(.)$ here is an indicator function (e.g. zero if its argument is 0, 1 otherwise), $f'(.)$ is zero almost everywhere so that parameters in weighting schemes for the 'transformation of averages' case cannot be estimated. This makes perfect sense — the covariate value will be 1 or 0 regardless of the weighting scheme used, so in this case there is no information in the data regarding the parameters.

D Residuals

The model fitting software generates a variety of residual analyses by default. Residuals can be used to check both the systematic component of a model, and its distributional assumptions. They can also be used in simulation (see Appendix E below).

D.1 Types of residual

There is no unique definition of a 'residual' for a GLM. The main residual measures used in this software are as follows:

PEARSON RESIDUALS: these are defined in such a way that, if the model is correct, they all come from distributions with zero mean and the same variance. They can be defined as

$$r_i^{(P)} = K\frac{Y_i - \mu_i}{\sigma_i} \ , \tag{30}$$

where μ_i and σ_i are the modelled mean and standard deviation for the ith case in the dataset and K is a constant that may be chosen to make the residuals as interpretable as possible. Often it will be sensible to set $K = 1$, so that the residuals all come from distributions with zero mean and unit variance. For a gamma GLM, however, we have $\sigma_i = \mu_i/\sqrt{\nu}$, where ν is the common shape parameter of the distributions (see Appendix A). In this case, putting $K = 1/\sqrt{\nu}$ gives the residual measure $(Y_i - \mu_i)/\mu_i$, which is just the proportional error in prediction. This might be preferred as being more directly interpretable.

ANSCOMBE RESIDUALS: These are model residuals defined, in some sense, to have a distribution that is as close to Gaussian as possible. This is extremely useful for simulation of dependent sequences at several sites (see Appendix E below), particularly when the variable of interest is continuous. Anscombe residuals can also be used to check the distributional assumptions of a GLM (see Appendix D.2 below). For a gamma GLM, the Anscombe residual for the ith case is defined to be $(y_i/\mu_i)^{1/3}$.

D.2 Checking the distributional assumptions

For continuous response variables, the easiest way to check the form of the forecast distribution is via quantile-quantile plots of suitably-defined residuals. 'Suitably-defined' here means that if the model is correct, all residuals come from identical distributions. For the gamma family of distributions, Anscombe residuals provide such a measure whose distribution is approximately normal; hence a normal probability plot of Anscombe residuals from a gamma GLM can be used to check the gamma assumption. This can be done very easily using a reliable statistical software package such as S-Plus or R (R Development Core Team, 2008) —

for a gamma GLM, the Anscombe residuals can be defined as the cube root of (OBSERVED ÷ EXPECTED) in the file res_gamm.dat. produced by `fit_gamm`.

Checks for discrete variables are more difficult. Here we focus on the binary case, and consider specifically the modelling of rainfall occurrence. If we collect together all of the days when the forecast probability of rain is close to some preassigned value p^*, then the overall proportion of these days upon which rainfall was actually observed should be close to p^*. An overview of the ideas is given by Dawid (1986). For practical implementation, we collect together groups of days for which forecast probabilities are in the intervals $(0.0, 0.1), (0.1, 0.2), \ldots, (0.9, 1.0)$ and compute observed and expected proportions of rainy days within each of these groups. Unless there is agreement across the whole range of forecast probabilities, there is something wrong with the probability structure of the model.

D.3 Checking systematic structure

Traditionally in regression modelling, the systematic structure of a model is checked by plotting residuals against predicted values, and against the covariates themselves. Such plots may be produced both for predictors which appear in the model, and for potential covariates that may need to be accounted for. Any apparent structure in these plots indicates a problem with the model. However, in many climate datasets such plots contain too many data points to distinguish any structure. For this reason, rather than plotting individual residuals we focus on summary statistics for residual measures over subgroups of observations. The software computes mean Pearson residuals for each month, year and site. This allows the modeller to check very quickly that the model captures the seasonal and regional structure in the data, along with any trends.

To aid the interpretation of such mean residuals, it is helpful to calculate their standard errors, which can be converted into confidence bands if desired. Suppose that all residuals are expected to have mean μ_ε and variance σ^2_ε under the model, and a mean residual ($\overline{r}$, say) is computed over a large subset of M cases. Then 95% limits for this mean are at $\mu_\epsilon \pm 1.96$s.e.$(\overline{r})$, where s.e.$(\overline{r})$ is the standard error of the mean residual under the model. This standard error can be calculated as follows: the mean residual is defined as

$$\overline{r} = \frac{1}{M} \sum_{t=1}^{T} \sum_{s=1}^{S} \chi_{ts} r_{ts} \ , \tag{31}$$

where T is the number of days in the subset under consideration; S the number of sites; χ_{ts} is an indicator taking the value 1 if a residual is available for site s on day t, 0 otherwise; and r_{ts} is the residual at site s on day t. Since M is the total number of observations in the subset, we have $M = \sum_{t=1}^{T} \sum_{s=1}^{S} \chi_{ts}$. When there is dependence between residuals at different sites on the same day (but independence between days), denote by $c_{s_1 s_2}$ the correlation between residuals at sites s_1 and s_2 on the same day. Then we have

$$\mathrm{Var}(\overline{r}) = \frac{1}{M^2} \sum_{t=1}^{T} \sum_{s_1=1}^{S} \sum_{s_2=1}^{S} \chi_{ts_1} \chi_{ts_2} \mathrm{cov}(r_{ts_1}, r_{ts_2}) \quad = \quad \frac{\sigma^2_\varepsilon}{M^2} \sum_{s_1=1}^{S} \sum_{s_2=1}^{S} \sum_{t=1}^{T} \chi_{ts_1} \chi_{ts_2} c_{s_1 s_2}$$

$$= \frac{\sigma_\varepsilon^2}{M^2} \sum_{s_1=1}^{S} \sum_{s_2=1}^{S} n_{s_1 s_2} c_{s_1 s_2} \ , \qquad (32)$$

where $n_{s_1 s_2}$ is the number of days for which sites s_1 and s_2 both have data. Taking the square root of **(32)** now yields the required standard error.

In implementing **(32)**, the software calculates a single set of inter-site correlations $\{c_{s_1 s_2}\}$ from the entire dataset, and applies these to all subsets.

E Simulation

This section gives an overview of the algorithms used in the software for simulation of daily sequences using the fitted GLMs.

E.1 Random number generator

The simulation program makes extensive use of pseudo-random numbers. Rather rely on the adequacy of generators 'built-in' to FORTRAN compilers (which are generally inadequately documented, and occasionally use rather poor algorithms), the routines used here are based on a uniform random number generator with excellent properties (Marsaglia and Zaman, 1991). These routines were written by myself together with Paul Northrop, and are suitable for extensive simulation excercises. The random number generation code is included with the software distribution. Full details, including documentation and references, are available from

`http://www.homepages.ucl.ac.uk/~ucakarc/work/randgen.html`.

E.2 Spatial dependence — continuous variables

To generate realistic sequences of simulated climate data at several sites simultaneously, it is necessary to account adequately for the dependence between sites. The literature offers a plethora of possible methods for generating such dependent sequences. This software offers a limited, but hopefully adequate, range of options. All of these options are specified in such a way that, given observations at some sites on a particular day, the conditional distribution for missing observations at other sites can be calculated. This allows missing data to be imputed.

For continuous random variables, a convenient way of generating dependent data is to generate a multivariate normal random vector using a standard algorithm, and then transform the resulting values so that they can be regarded as coming from the correct marginal distributions. In this way, spatial dependence is completely characterised by the correlation structure of the multivariate normal distribution. Furthermore, if some elements of a

multivariate normal random vector have been observed then a standard result enables us to compute the conditional distribution of the remaining elements (which is still multivariate normal) — this enables us to sample missing observations using the information in the available data.

The transformation linking the normal distribution to that specified in a GLM can be derived by considering the Anscombe residuals (see Appendix D). To generate a vector of dependent random variables in the GLM, we can draw a vector of correlated Anscombe residuals from the appropriate multivariate normal distribution and invert the residual transformation (which will depend on the means under the model). In general, Anscombe residual transformations are approximate rather than exact, but the approximation is extremely good in many cases.

Providing each pair of sites has an overlapping period of record, the Anscombe residual correlation for that pair can be estimated straightforwardly. Unfortunately however, the resulting set of correlations for all pairs of sites is not guaranteed to be internally consistent (see footnote on page 9). Moreover, correlations cannot be estimated for pairs of sites whose periods of records do not overlap, or for ungauged locations. To get round this, one might choose to adopt a valid spatial model for the Anscombe residual correlations. For small spatial scales, inter-site correlations tend to be so similar that they can be regarded as effectively constant: in this case, the constant correlation can be estimated as a weighted average of the available inter-site correlations with weights proportional to the numbers of contributing observations for each pair of sites.

At larger spatial scales, however, it becomes necessary to account for the likely decay of correlation with inter-site distance (and potentially direction). There are several commonly-used spatial correlation models that can capture this behaviour. Currently the software offers a choice of exponential and powered exponential correlation models (see Table 8); also the option for the correlation to decay to a non-zero threshold rather than to zero at large distances. This phenomenon has been observed in several data sets, and probably arises from the relatively small scale of many study regions relative to the synoptic scales of the weather systems affecting them: the overall characteristics of an individual system are likely to affect all sites simultaneously on any given day, with enhanced inter-site dependence at local scales within the system.

The correlation models currently handled by the software can all be written in the general form

$$\rho\left(d;\boldsymbol{\theta},\alpha\right)=\alpha+\left(1-\alpha\right)\rho^{*}\left(d;\boldsymbol{\theta}\right)\ , \tag{33}$$

where d is the inter-site distance, $\boldsymbol{\theta}$ is a parameter vector, α is the limiting correlation at large distances and $\rho^{*}\left(\cdot;\cdot\right)$ is a 'standard' correlation model that decays to zero at large distances. The parameters $(\boldsymbol{\theta},\alpha)$ are estimated by minimising a weighted least-squares objective function:

$$S\left(\boldsymbol{\theta},\alpha\right)=\sum_{i}n_{i}\left[r_{i}-\rho\left(d_{i};\boldsymbol{\theta},\alpha\right)\right]^{2}\ , \tag{34}$$

where the sum is over all pairs of sites with overlapping records; r_i is the Anscombe residual

correlation computed for the ith pair; n_i is the number of residuals contributing to this correlation; and d_i is the inter-site distance for this site pair.

Minimising **(34)** with respect to $\boldsymbol{\theta}$ and α can be achieved by equating the gradient vector to zero. Writing $\rho_i = \rho\left(d_i; \boldsymbol{\theta}, \alpha\right)$ and $\rho_i^* = \rho^*\left(d_i; \boldsymbol{\theta}, \alpha\right)$, we have

$$\frac{\partial S}{\partial \boldsymbol{\theta}} = -2(1-\alpha)\sum_i n_i \left(r_i - \rho_i\right)\frac{\partial \rho_i^*}{\partial \boldsymbol{\theta}} \tag{35}$$

$$\text{and} \qquad \frac{\partial S}{\partial \alpha} = -2\sum_i n_i \left(r_i - \rho_i\right)\left(1 - \rho_i^*\right) \ . \tag{36}$$

For a given value of $\boldsymbol{\theta}$, **(36)** can be solved analytically to yield the corresponding optimal value of α as

$$\hat{\alpha}\left(\boldsymbol{\theta}\right) = \left[\sum_i n_i \left(r_i - \rho_i^*\right)\left(1 - \rho_i^*\right)\right] / \left[\sum_i n_i \left(1 - \rho_i^*\right)^2\right] \ . \tag{37}$$

For most correlation models, however, **(35)** does not have an analytical solution. For a given value of α, the solution can in principle be found iteratively by specifying an initial estimate $\boldsymbol{\theta}^{(0)}$ and then iterating to convergence the Newton-Raphson scheme

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \left[\left.\frac{\partial^2 S}{\partial \boldsymbol{\theta}\partial \boldsymbol{\theta}'}\right|_{\boldsymbol{\theta}^{(t-1)}}\right]^{-1} \left.\frac{\partial S}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}^{(t-1)}} \ . \tag{38}$$

The Hessian $\partial^2 S / \partial \boldsymbol{\theta} \partial \boldsymbol{\theta}'$ can be derived from **(35)** as

$$\frac{\partial^2 S}{\partial \boldsymbol{\theta}\partial \boldsymbol{\theta}'} = -2\left(1-\alpha\right)\sum_i n_i \left[\left(r_i - \rho_i\right)\frac{\partial^2 \rho_i^*}{\partial \boldsymbol{\theta}\partial \boldsymbol{\theta}'} - \left(1-\alpha\right)\left(\frac{\partial \rho_i^*}{\partial \boldsymbol{\theta}}\right)\left(\frac{\partial \rho_i^*}{\partial \boldsymbol{\theta}}\right)'\right] \ .$$

The software implements a slightly modified version of this scheme in which, at each iteration, the value of α is also updated using **(37)**. A further refinement is that the adjustments in each iteration of **(38)** are scaled if necessary to ensure that the values of $\boldsymbol{\theta}$ always satisfy any necessary constraints for the underlying correlation model. The resulting algorithm is probably suboptimal, but it seems to be fairly stable and to produce reasonable results.

E.3 Spatial dependence — binary variables

Incorporating spatial dependence into binary sequences is much more difficult than for continuous variables. Several options are available within this software. They are documented fairly completely here since not all details have been published elsewhere.

Throughout this section we denote by S_t the number of sites we are studying on day t, and by $\mathbf{Y}_t = \left(Y_{1t} \ \ldots \ Y_{S_t t}\right)'$ a vector of binary variables corresponding to rainfall occurrence at those sites. A logistic regression model allows us to calculate $E\left(Y_{st}\right) = p_{st}$, say.

E.3.1 Latent Gaussian variables

A conceptually simple approach to generating correlated binary variables is to start by generating a set of correlated Gaussian variables $\mathbf{Z} = (Z_1, \ldots, Z_{S_t})$ and then to define

$$Y_{st} = \begin{cases} 1 & \text{if } Z_s > \tau_{st} \\ 0 & \text{otherwise,} \end{cases} \tag{39}$$

where the thresholds $\tau_{st}, \ldots, \tau_{S_t t}$ are chosen to ensure that $P(Y_{st} = 1) = p_{st}$ as required by the logistic regression model. Specifically, if the $\{Z_s\}$ are all standard normal variables (i.e. with zero mean and unit variance), we need to set $\tau_{st} = -\Phi^{-1}(p_{st})$ where $\Phi(\cdot)$ is the distribution function of the standard normal distribution: standard algorithms are available for computing this.

In this setup, dependence between sites s_1 and s_2 can be induced by specifying a correlation, $\rho_{s_1 s_2}$ say, between Z_{s_1} and Z_{s_2}. As this correlation varies between -1 and $+1$, so the strength of dependence between $Y_{s_1 t}$ and $Y_{s_2 t}$ varies over its entire range. The idea seems first to have been used by Pearson (1901). In the present context, the 'latent correlations' between each pair of sites are chosen so as to match the proportion of days for which both sites experience rain. Specifically, suppose there are n days for which observations are available at both sites s_1 and s_2. Then the observed proportion of days for which both sites experience rain is $n^{-1} \sum_t Y_{s_1 t} Y_{s_2 t}$ and the expected proportion is $n^{-1} \sum_t P\left(Z_{s_1} > \tau_{s_1 t}, Z_{s_1} > \tau_{s_2 t}\right)$ (the sums here are over days for which both sites have data). Thus we choose the correlation $\rho_{s_1 s_2}$ to solve the equation

$$\sum_t P\left(Z_{s_1} > \tau_{s_1 t}, Z_{s_1} > \tau_{s_2 t}\right) = \sum_t Y_{s_1 t} Y_{s_2 t} \ . \tag{40}$$

For a given value of $\rho_{s_1 s_2}$, the probability on the left-hand side of **(40)** can be calculated using algorithms for the bivariate normal distribution: the software uses that of Donnelly (1973). A numerical search is then carried out to solve the equation for $\rho_{s_1 s_2}$.

The approach as just outlined has two drawbacks. First, by comparison with the other binary dependence models discussed below it is relatively slow: latent correlations have to be estimated separately for each pair of sites, and each of these requires repeated evaluation of bivariate normal probabilities (the calculation of which is nontrivial despite the use of a fast algorithm) in the search for a root of **(40)**. For a moderately large data set (say 40 or 50 years of daily data at 40 sites), it might take between 5 and 10 minutes to estimate these correlations on a relatively fast modern machine — obviously the computing time increases quadratically with the number of sites.

A second drawback, which is arguably more serious, is that the matrix obtained by solving **(40)** for each pair of sites is not guaranteed to be positive definite, and is therefore not necessarily a valid correlation matrix. This can cause problems in simulation, since simulation algorithms for the multivariate normal distribution (required to generate a realisation of $\mathbf{Z}$ on each day of simulation here) require that the correlation matrix is positive definite. In practice, to overcome this problem it is necessary to postprocess the individual correlations

by fitting an appropriate spatial correlation model and then using the *modelled* correlations as inputs to the simulation program. The software uses the same algorithm for fitting spatial correlation models here as for the Anscombe residuals; see Appendix E.2 for details.

A final problem, in some way related to the second drawback above, is that a solution to **(40)** is not guaranteed. This is because as $\rho_{s_1 s_2}$ varies from its minimum value of -1 to its maximum value of $+1$, it can be shown that $P(Y_{s_1 t} = Y_{s_2 t} = 1)$ varies correspondingly from $\max(p_{s_1 t} + p_{s_2 t} - 1, 0)$ to $\min(p_{s_1 t}, p_{s_2 t})$. Thus the right-hand side of **(40)** is constrained to lie in the range

$$\sum_t \left[\max(p_{s_1 t} + p_{s_2 t} - 1, 0), \min(p_{s_1 t}, p_{s_2 t})\right] .$$

However, the left-hand side is not so constrained and — either due to sampling variation or to slight mis-specification of the modelled marginal probabilities by the GLM — can occasionally produce values outside this range. If this occurs for a pair of sites, the software estimates the corresponding latent correlation as -1 or $+1$ as appropriate, and issues a warning message.

Given a positive definite correlation matrix, simulation of a dependent vector of rainfall occurrence indicators is straightforward using this dependence structure: for each day of simulation, calculate the marginal probabilities as predicted by the GLM, simulate a vector $\mathbf{Z}$ from a multivariate normal distribution with the specified correlation structure, and then threshold the simulated Zs according to equation **(39)**.

Imputation is more difficult, however. Essentially, what is required is to simulate from the conditional distribution of $\mathbf{Z}$ given the observed elements of $\mathbf{Y}_t$ and then to generate the missing elements by thresholding the generated Z_s as before. The problem is in simulating from the conditional distribution of $\mathbf{Z}$. A naïve algorithm is as follows:

1. Sample a value of $\mathbf{Z}$ from its unconditional distribution
2. If all of the observed elements of $\mathbf{Y}_t$ are consistent with the corresponding elements of $\mathbf{Z}$, continue; otherwise reject the sampled $\mathbf{Z}$ and return to step 1.

The problem with this algorithm is that if observations are available at many sites, the probability of simultaneously generating a $\mathbf{Z}$ that is consistent with *all* of the available observations will typically be rather small. Therefore, many attempts will be required before an acceptable $\mathbf{Z}$ is generated in step 1; this makes the algorithm very slow.

As an alternative therefore, one might consider speeding up the algorithm by retaining those elements of $\mathbf{Z}$ that are consistent with the corresponding observations and then re-sampling the remainder from their distribution conditional on the retained elements; this procedure can then be iterated until all of the generated values are consistent with the observations. Unfortunately, it can be shown that this procedure is incorrect and does *not* lead to a sample from the distribution of $\mathbf{Z}$ given $\mathbf{Y}_t$.

The solution adopted in the software is to deal with the problem in two parts. Specifically, denote by $\tilde{\mathbf{Y}}_t$ the vector of *observed* components of $\mathbf{Y}_t$ and by $\tilde{\mathbf{Z}}$ the corresponding elements

of $\mathbf{Z}$. Then the first step is to sample from the joint distribution of $\tilde{\mathbf{Z}}$ conditional on $\tilde{\mathbf{Y}}_t$; the second step is to sample the remaining elements of $\mathbf{Z}$ from their distribution given $\tilde{\mathbf{Z}}$ (which is straightforward since the joint distribution is multivariate normal) and to threshold these remaining sampled elements to impute the missing values of $\mathbf{Y}_t$. Efficient sampling from the distribution of $\tilde{\mathbf{Z}}|\tilde{\mathbf{Y}}_t$ itself is nontrivial: the algorithm implemented here is a Gibbs sampler in which the elements of $\tilde{\mathbf{Z}}$ are initialised by sampling each one independently from a normal distribution truncated at the corresponding threshold τ; and then repeatedly visiting each element of $\tilde{\mathbf{Z}}$ in turn and sampling from its distribution conditional on the current configuration of the remaining elements and of $\tilde{\mathbf{Y}}$. Specifically, according to the correlation model outlined above, the unconditional distribution of $\tilde{\mathbf{Z}}$ is multivariate normal with mean $\mathbf{0}$ and covariance matrix $\mathbf{\Sigma}$, say. Let $\tilde{Z}_i$ denote the ith element of $\tilde{\mathbf{Z}}$ and let $\tilde{Z}_{(-i)}$ denote the vector of the remaining elements. Moreover, let $\sigma^{(ii)}$ denote the ith diagonal element of $\mathbf{\Sigma}^{-1}$ and let $\mathbf{\Sigma}^{-1}_{(i,-i)}$ denote the ith row of $\mathbf{\Sigma}^{-1}$ with the ith element removed. Then, using standard results for conditioning in the multivariate normal distribution, as well as the formula for the inverse of a partitioned matrix **REFS!!!** , it may be shown that the distribution of $\tilde{Z}_i$ given $\tilde{\mathbf{Z}}_{(-i)}$ is normal with mean $-\mathbf{\Sigma}^{-1}_{(i,-i)}\tilde{\mathbf{Z}}_{(-i)}/\sigma^{(ii)}$ and variance $1/\sigma^{(ii)}$. To update the value of $\tilde{Z}_i$ therefore requires sampling from this normal distribution, truncated as appropriate for consistency with the corresponding observation $\tilde{Y}_i$ say.

In the present context, a single iteration of the Gibbs sampler consists of a sequence of updates in which every element of $\tilde{\mathbf{Z}}$ is visited once. If the procedure is repeated for a large number of iterations, the resulting $\tilde{\mathbf{Z}}$ will be sampled approximately from the required distribution. Obviously, there is a tradeoff here between accuracy and computation time: the higher the number of iterations, the closer will be the distribution of the sampled $\tilde{\mathbf{Z}}$ to the required target but the overall simulation time will increase. For the purposes of imputing binary wet/dry indicators however, excessive accuracy is probably unnecessary and speed of execution is a priority. The software therefore uses just 10 Gibbs iterations when imputing rainfall occurrence indicators using this spatial dependence model. This choice was made on the basis of plots such as Figures 6 and 7. These have been produced for a hypothetical network of five sites of which site 2 is known to be dry and the remainder are wet, and with latent inter-site correlations ranging from 0.61 to 0.96. The top panel of Figure 6 shows a random sample of 50 initial configurations for the Gibbs sampler in this situation, with each element of $\tilde{\mathbf{Z}}$ sampled independently as described above. The independent sampling shows up clearly: the sampled points are not aligned along the contours of the correlated joint distributions of the $\tilde{\mathbf{Z}}$-pairs. However, by the tenth iteration of the sampler (bottom panel) the points look much more plausibly like samples from these joint distributions. Figure 7 is another way of exploring the convergence of the sampler: here, at each iteration, the quantity $\log \pi\left(\tilde{\mathbf{Z}}\right)$ is plotted for every one of the 50 initial configurations, where $\pi(\cdot)$ is the joint density of a multivariate normal distribution with mean $\mathbf{0}$ and covariance matix $\mathbf{\Sigma}$. The figure shows that the initial configurations typically have a very low joint density, but that this rapidly increases within a few iterations to fluctuate around an equilibrium level indicating convergence to the required distribution. According to this plot, all but one of the 50 traces have reached the equilbrium level by around the tenth iteration.

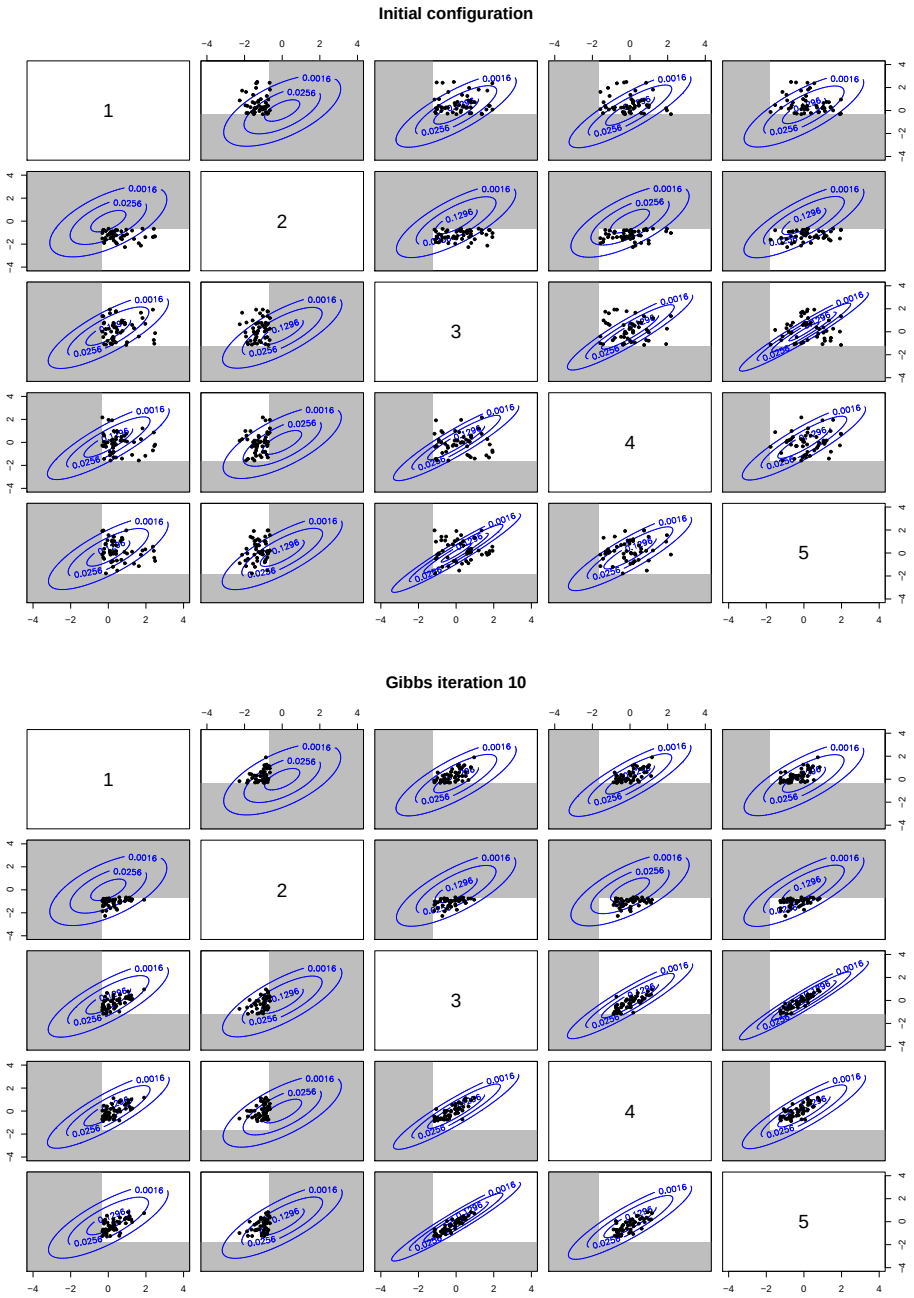

Figure 6: Gibbs sampling to draw from distribution of $\tilde{\mathbf{Z}}|\tilde{\mathbf{Y}}_t$ at five sites, with intersite correlations in $\tilde{\mathbf{Z}}$ ranging from 0.61 to 0.96. Site 2 is known to be dry and the remaining sites are wet; the grey region in each plot shows the values of $\tilde{\mathbf{Z}}$ that are inconsistent with these observations. Blue lines in each plot are contours of the bivariate densities from which the elements of $\tilde{\mathbf{Z}}$ would be drawn in the absence of observations. 50 separate realisations have been produced: each has been initialised by sampling the elements of $\tilde{\mathbf{Z}}$ independently from their marginal distributions (top plot). Bottom plot shows the sampled configurations after 10 iterations of the Gibbs sampler.

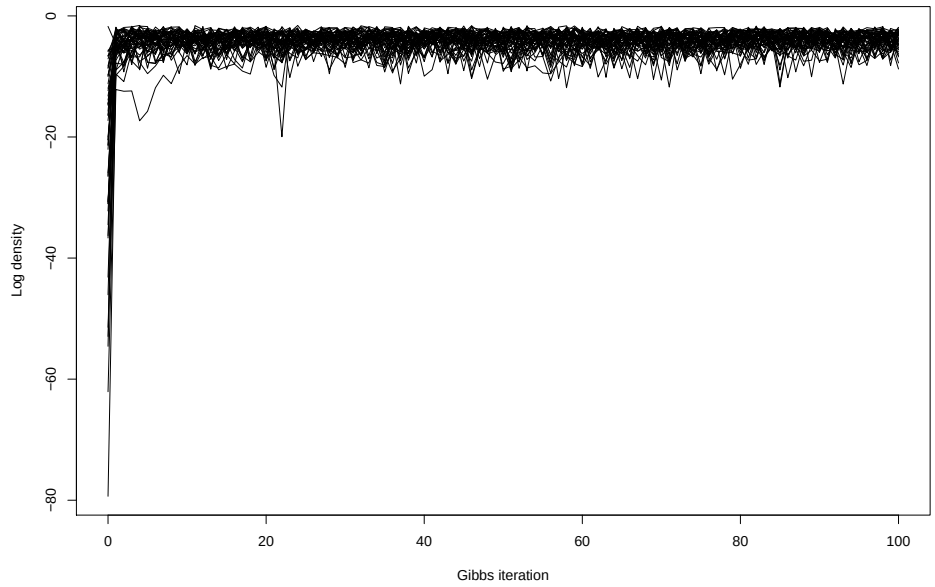

Figure 7: Convergence of the Gibbs sampler for $\tilde{\mathbf{Z}}$: logarithm of joint density of sampled observations over 100 Gibbs iterations, for each of 50 samples.

E.3.2 Hidden binary weather state

Over a wide range of spatial scales, spatial dependence in binary climate sequences at a daily timescale (e.g. absence/occurrence of rainfall) is mainly due to the fact that all sites tend to be influenced by the same weather systems on particular days. This process can be modelled by including a hidden weather state which categorizes each day as 'on' or 'off' over the entire area. For incorporation into a logistic regression model, it is convenient to model the effect of such a hidden weather state on the log odds scale.

In this context, we consider that there is a random variable X_t associated with day t that represents the 'weather state' on that day. $X_t = 1$ with probability α_t (representing a 'wet day') and 0 otherwise (a 'dry day'). X_t is not directly observable.

The $\{Y_{st}\}$ are modelled as conditionally independent given X_t. The probability of rain at each site on day t is modelled using

$$\ln\left(\frac{P\left(Y_{st}=1|X_t=x\right)}{1-P\left(Y_{st}=1|X_t=x\right)}\right)=\ln\left(\frac{p_{st}}{1-p_{st}}\right)+x\ln a+\ln b_{st}\left(\alpha_t,p_{st},a\right)\ . \tag{41}$$

Here $\ln a$ is constant for all sites and days, and is free over the range $(-\infty,\infty)$. $\ln b_{st}(.)$ is a function of α_{st}, p_{st} and a, chosen to ensure that the unconditional probability of rain at site s is p_{st}. We abbreviate it to b_{st} for convenience. Some straightforward manipulation enables us to calculate b_{st} from the remaining parameters.

It can be shown that no matter how large a is, we cannot make $P(Y_{st} = 1|X_t = 1)$ arbitrarily close to 1 if $p_{st} < \alpha_t$. Similarly, we cannot make $P(Y_{st} = 1|X_t = 0)$ arbitrarily close to zero if $p_{st} > \alpha_t$. This indicates a limitation of this particular model.

The covariance structure of model **(41)** can be derived. The covariance between Y_{s_1t} and Y_{s_2t} $(s_1 \neq s_2)$ is defined as $P(Y_{s_1t} = Y_{s_2t} = 1) - p_{s_1t}p_{s_2t}$, and is equal to

$$\operatorname{cov}(Y_{s_1t}, Y_{s_2t}) = \frac{(1-\alpha)p_{s_1t}p_{s_2t}(1-p_{s_1t})(1-p_{s_2t})(1-b_{s_1t})(1-b_{s_2t})}{\alpha_t[1-p_{s_1t}(1-b_{s_2t})][1-p_{s_1t}(1-b_{s_2t})]} . \tag{42}$$

Note that this covariance is not explicitly dependent on inter-site distance: thus this particular dependence structure is probably unsuitable for use in catchments that are large relative to the typical scale of weather systems that affect them so that inter-site correlations vary appreciably in magnitude for different pairs of sites.

This spatial dependence model involves two unknown parameters on any given day: α_t and a. An obvious choice for α_t is the mean of the $\{p_{st}\}$ on day t. In this case the marginal predictions of the GLM at all sites are used to determine whether a day will be 'wet' or 'dry'. Thus features such as seasonality are automatically incorporated into the weather states, since these should be reflected in the $\{p_{st}\}$.

How to choose a is less obvious because the sequence of Xs is not observed. Note, however, that given the ps predicted by the GLM, the corresponding α can be calculated. In this case, given the observed Y values it is possible, numerically, to obtain a maximum likelihood estimate of a. However, this may be undesirable since the model structure, while plausible in some applications, is probably a fairly crude approximation of reality. An alternative estimation procedure is a method of moments. In applications, it is often important to reproduce the spatial coverage of a binary random field (i.e. the proportion of 1s among the sites). The software therefore chooses a so as equate the observed and theoretical variances of the coverage distribution. The theoretical variance can be derived from the expression for covariances at **(42)**. The observed variance can be calculated straightforwardly from the time series of coverages, weighting each day by the number of active sites. Equating the two can be achieved using straightforward numerical methods.

Imputation of missing data is straightforward under this spatial dependence structure. Suppose on a particular day we observe $Y_{1t} = y_1, \ldots, Y_{kt} = y_k$ $(s < S_t)$, and wish to fill in the remaining values $Y_{(k+1)t}, \ldots, Y_{S_t t}$. Rearranging **(41)**, we find that

$$P(Y_{1t} = y_1, \ldots, Y_{kt} = y_k|X_t = x) = \prod_{y_s=1} \frac{a^x b_{st} p_{st}}{1 - p_{st}(1 - a^x b_{st})} \prod_{y_s=0} \left(1 - \frac{a^x b_{st} p_{st}}{1 - p_{st}(1 - a^x b_{st})}\right) .$$

Now we can use Bayes' Theorem to find the conditional probability distribution of X_t on the basis of the observed Ys:

$$P(X_t = 1|Y_{1t} = y_1, \ldots, Y_{kt} = y_k) = \frac{P(Y_{1t} = y_1, \ldots, Y_{kt} = y_k|X_t = 1)}{\sum_{x=0}^{1} P(Y_{1t} = y_1, \ldots, Y_{kt} = y_k|X_t = x)P(X_t = x)} . \tag{43}$$

Recalling that $P(X_t = 1) = \alpha = 1 - P(X_t = 0)$, we can therefore impute missing data by simulating $X_t = 1$ with probability given by **(43)**, and then sampling the remaining missing sites independently from **(41)** as before.

E.3.3 Modelling the coverage distribution

One feature of the hidden weather state model above is that the probability of all sites being in the same state (0 or 1) cannot be made arbitrarily close to 1. This may not be a problem for some applications; however, in some problems (for example rainfall modelling at small spatial scales) it is common for the distribution of coverage to be 'U-shaped' with a high concentration at both 0 and 1. An alternative approach to the generation of dependent binary sequences is to address the dependence directly through the coverage distribution, thus guaranteeing that simulations will reproduce this important feature.

In this alternative model therefore, a (non-unique) dependence structure can be specified for $\mathbf{Y}_t$ through the distribution of $Z_t = \sum_{s=1}^{S_t} Y_{st}$. Since the marginal occurrence probabilities (i.e. the p_{st}s) vary from day to day, so does the distribution of Z_t — in particular, we have $\mathrm{E}(Z_t) = \sum_{s=1}^{S_t} p_{st}$.

A flexible family of distributions for discrete random variables taking values in $\{0, 1, \ldots, S_t\}$ is the Beta-Binomial family:

$$P(Z_t = z) = \binom{S_t}{z} \frac{\Gamma(\alpha_t + z)\,\Gamma(S_t + \beta_t - z)\,\Gamma(\alpha_t + \beta_t)}{\Gamma(\alpha_t + \beta_t + S_t)\,\Gamma(\alpha_t)\,\Gamma(\beta_t)} \qquad (z = 0, 1, \ldots, S_t)\ , \tag{44}$$

for some parameters $\alpha_t, \beta_t \in \mathbb{R}^+$. For small values of S_t, these probabilities can be evaluated cheaply using a recurrence relation. The mean and variance of the distribution are

$$\frac{S_t \alpha_t}{\alpha_t + \beta_t} \qquad \text{and} \qquad \frac{S_t \alpha_t \beta_t (\alpha_t + \beta_t + S_t)}{(\alpha_t + \beta_t)^2 (\alpha_t + \beta_t + 1)}\ , \tag{45}$$

respectively. The uniform distribution corresponds to the special case $\alpha_t = \beta_t = 1$. If $\alpha_t = 0, \beta_t \neq 0$ then $P(Z_t = 0) = 1$, and if $\beta_t = 0, \alpha_t \neq 0$ then $P(Z_t = S_t) = 1$. The case when $\alpha_t = \beta_t = 0$ is discussed below.

It is convenient to reparametrise the Beta-Binomial distribution here: set

$$\theta_t = \frac{\alpha_t}{\alpha_t + \beta_t} \qquad \text{and} \qquad \phi_t = \alpha_t + \beta_t\ , \tag{46}$$

so that

$$\alpha_t = \theta_t \phi_t\ , \qquad \beta_t = \phi_t (1 - \theta_t)\ , \qquad \mathrm{E}(Z_t) = S_t \theta_t \text{ and } \mathrm{Var}(Z_t) = \frac{S_t \theta_t (1 - \theta_t)(\phi_t + S_t)}{\phi_t + 1}\ . \tag{47}$$

We can think of θ_t as a mean value parameter, and ϕ_t as a dispersion parameter (in fact, ϕ_t essentially controls the tendency of the distribution to be concentrated either at its extremities or around its mean). It is convenient, and not implausible, to assume that $\phi_t = \phi$

is constant for all t, so that θ_t (which is known, since it can be calculated from the logistic regression model) is the only time-varying parameter of the distribution. As θ_t varies then, so we hope to reproduce typical 'summer' and 'winter' coverage distributions for example.

The reparametrisation also allows us to investigate the shape of the distribution when $\alpha_t = \beta_t = 0$. In this case, we have $\phi = 0$. If we consider $\lim_{\phi \to 0} P(Z_t = 0)$ with θ_t fixed, we find that in the limit Z_t takes the values 0 and S_t with probabilities $1 - \theta_t$ and θ_t respectively. At the other extreme, it can be shown that the limiting distribution as $\phi \to \infty$ is Binomial with parameters S_t and θ_t. Since the Binomial arises if all the Y_{st}s are independent and identically distributed, we see that ϕ can be regarded as an overall summary of the dependence among the Y_{st}s — small values of ϕ correspond to strong dependence.

Given data $\{(S_t, Z_t, \theta_t) : t = 1, \ldots, T\}$, a natural way to estimate the dispersion parameter ϕ is via a method of moments. Note that

$$\mathrm{E}\left(\frac{(Z_t - S_t\theta_t)^2}{S_t\theta_t(1-\theta_t)}\right) = \frac{\phi + S_t}{\phi + 1} = E\left(R_t^2\right) \quad \text{say.}$$

Hence

$$\mathrm{E}\left(\sum_{t=1}^{T} R_t^2\right) = T + \frac{1}{\phi+1}\sum_{t=1}^{T}(S_t - 1) \ ,$$

so that a natural estimator of ϕ is

$$\hat{\phi} \ = \ \frac{\sum_{t=1}^{T}(S_t - 1)}{\sum_{t=1}^{T}(R_t^2 - 1)} - 1 \ . \tag{48}$$

In the simple case where all the p_{st}s are equal and the Ys are independent, we have $Z_t \sim Bin(S_t, \theta_t)$, and $\mathrm{Var}(Z_t) = S_t\theta_t(1-\theta_t)$, whence $\mathrm{E}(R_t^2) = 1$. If all the R_t^2s were equal to 1, **(48)** would give $\hat{\phi} = \infty$ (corresponding to independence): overdispersion results in lower values of $\hat{\phi}$ as expected.

To simulate from this model, a natural strategy is to sample the number of wet sites from the distribution of Z_t, and then to allocate the positions of these wet sites. However, this will only yield sequences with the correct properties if the conditional probabilities of rain at each site, given the overall number of wet sites, are correctly specified. This can be achieved providing a valid joint distribution can be found for $\mathbf{Y}_t$ and Z_t. We now describe the algorithm used to find such a joint distribution (which may not be unique). The subscript t is now unnecessary, so we drop it and write $\mathbf{Y}, Z$. We assume initially that the given distribution of Z is compatible with the individual probabilities of the Ys (this is not guaranteed, as we will see below).

From our earlier notation, we have $P(Y_s = 1) = p_s$. We also define $\pi_z = P(Z = z)$, and $w_{s,z} = P(Y_s = 1 \text{ and } Z = z)$. A first step in determining a joint distribution of $\mathbf{Y}$ and Z is to find $\{w_{s,z} : s = 1, \ldots, S; z = 0, \ldots, S\}$, from which we can calculate the conditional probabilities at each site:

$$P(Y_s = 1 | Z = z) = \frac{w_{s,z}}{\pi_z} \ . \tag{49}$$

		s				
		1	2	$\cdots$	S	**TOTAL**
z	0	0	0	$\cdots$	0	0
	1	$w_{1,1}$	$w_{2,1}$	$\cdots$	$w_{S,1}$	π_1
	2	$w_{1,2}$	$w_{1,2}$	$\cdots$	$w_{S,2}$	$2\pi_2$
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
	S	π_S	π_S	$\cdots$	π_S	$S\pi_S$
TOTAL		p_1	p_2	$\cdots$	p_S	$\mathrm{E}(Z)$

Figure 8: Contingency table illustrating restrictions on the weights $\{w_{s,z} : s = 1, \ldots, S; z = 0, \ldots, S\}$.

The following relationships must hold:

$$0 \leq w_{s,z} \leq \pi_z \; ; \tag{50}$$

$$\sum_{z=0}^{S} w_{s,z} = p_s \; ; \quad \text{and} \tag{51}$$

$$\sum_{s=1}^{S} w_{s,z} = z\pi_z \; . \tag{52}$$

The second of these is the Law of Total Probability; the third can be seen by noting that $\mathrm{E}\left[\left(\sum_s Y_s\right) \middle| Z = z\right] = z$. But

$$\mathrm{E}\left[\left(\sum_s Y_s\right) \middle| Z = z\right] = \sum_s P\left(Y_s = 1 | Z = z\right) = \sum_s \frac{w_{s,z}}{\pi_z} \; ,$$

and the condition follows.

To visualise the problem, it is helpful to lay the ws out in the form of a contingency table, as in Figure 8. The only constraint which is not apparent from this is **(50)**.

The algorithm developed here is based on linear programming ideas, but takes advantage of the rather tight constraints to speed up the search for the ws. Additionally, it is insensitive to the order in which sites are considered . The basic procedure is to allocate a row of the

table at a time, starting with $w_{s,0} = 0$ $(s = 1, \ldots, S)$ and noting that $w_{s,S} = \pi_S$ $(s = 1, \ldots, S)$ — both of which follow from constraints **(50)** and **(52)** above. As each row is allocated, the constraints on the remaining entities will change. Specifically, suppose we have allocated rows $0, 1, \ldots, z-1$ and are currently considering row $z \leq S-1$. Constraints **(50)** and **(52)** above are unchanged. If we define $p_{s|z} = P(Y_s = 1 \text{ and } Z \geq z)$, then constraint **(51)** becomes

$$\sum_{j=z}^{S} w_{s,z} = p_{s|z} = p_s - \sum_{j=0}^{z-1} w_{s,z} \ . \tag{53}$$

Since we know that $w_{s,S} = \pi_S$, we must have $w_{s,z} \leq p_{s|z} - \pi_S$ for $z \leq S-1$. A further inequality can be deduced from constraint **(50)** applied to the subsequent rows of the table: $p_{s|z+1} = \sum_{j=z+1}^{S} w_{s,j} \leq \sum_{j=z+1}^{S} \pi_j \Rightarrow p_{s,z} - w_{s,z} \leq \sum_{j=z+1}^{S} \pi_j$, so that $w_{s,z} \geq p_{s|z} - \sum_{j=z+1}^{S} \pi_j$. Putting these inequalities together we must have, for $z \leq S-1$,

$$\max\left(0, p_{s|z} - \sum_{j=z+1}^{S} \pi_j\right) \leq w_{s,z} \leq \min\left(\pi_z, p_{s|z} - \pi_S\right) \ . \tag{54}$$

Writing the lower and upper bounds as $LB_{s,z}$ and $UB_{s,z}$ respectively, and summing, gives

$$\sum_{s=1}^{S} LB_{s,z} \leq \sum_{s=1}^{S} w_{s,z} \leq \sum_{s=1}^{S} UB_{s,z} \ . \tag{55}$$

Now from **(52)** above, we require $\sum_{s=1}^{S} w_{s,z} = z\pi_z$. Hence, providing $\sum_{s=1}^{S} LB_{s,z} \leq z\pi_z \leq \sum_{s=1}^{S} UB_{s,z}$, we can set

$$w_{s,z} = LB_{s,z} + \frac{z\pi_z - \sum_{s=1}^{S} LB_{s,z}}{\sum_{s=1}^{S} UB_{s,z} - \sum_{s=1}^{S} LB_{s,z}} \tag{56}$$

for each s, and proceed to the next row of the table. If the desired row total $z\pi_z$ is outside the interval $\left[\sum_{s=1}^{S} LB_{s,z}, \sum_{s=1}^{S} UB_{s,z}\right]$ then we must return to a previous row and re-allocate some of the joint probabilities. The following result is useful:

Result: providing the entries in rows 0 to $z-1$ of the table all satisfy inequalities of the form **(54)**, the inequality $\sum_{s=1}^{S} UB_{s,z} \geq z\pi_z$ is automatically satisfied.

The proof is omitted, since it is not particularly instructive and the margin is too small to contain it. ■

The result tells us that in our algorithm, the only problem that can arise is when $\sum_{s=1}^{S} LB_{s,z} > z\pi_z$. Since $LB_{s,z} = \max\left(0, p_{s|z} - \sum_{j=z+1}^{S} \pi_j\right)$, the only way around this is to re-allocate some probabilities so as to reduce $p_{s|z}$ at sites where $p_{s|z} - \sum_{j=z+1}^{S} \pi_j > 0$ (because the πs are fixed), with a corresponding increase at sites where $p_{s|z} - \sum_{j=z+1}^{S} \pi_j < 0$.

If this cannot be achieved, the given distribution of Z is incompatible with the marginal probabilities of the Ys.

We are now in a position to summarise the algorithm for calculating the ws. For each z:

1. Compute $p_{s|z} = p_s - \sum_{j=1}^{z-1} w_{s,j}$.
2. Compute $LB_{s,z} = \max\left(0, p_{s|z} - \sum_{j=z+1}^{S} \pi_j\right)$ and $UB_{s,z} = \min\left(\pi_z, p_{s|z} - \pi_S\right)$.
3. Compute $\sum_{s=1}^{S} LB_{s,z}$ and $\sum_{s=1}^{S} UB_{s,z}$.
4. If $\sum_{s=1}^{S} LB_{s,z} \leq z\pi_z$, calculate the ws for the current row according to **(56)**. Otherwise, re-allocate some probabilities in previous rows of the table, if possible. For practical purposes, when re-allocating probabilities we try to avoid setting any value of $w_{s,z}$ to either 0 or π_z (except when $z = 0$ or S), since this can lead to problems in imputation (see below) and is unrealistic.

The joint distribution of $\mathbf{Y}$ and Z is only partially specified by the ws in Figure 8, since the dependencies among the Ys are not represented. However, for simulation purposes it is not necessary to specify the distribution completely: all that is required is to sample one site at a time and then update the probabilites at the remaining sites to condition upon the sampled value. It can be shown that this updating can be done using the algorithm of the previous section.

The discussion so far has assumed that the distribution of Z is compatible with the marginal probabilities of the Ys. This is not guaranteed — obvious examples of incompatibility arise when $P(Z = S) > \min_s P(Y_s = 1)$ and when $P(Z = 0) > \min_s P(Y_s = 0)$. When simulating long sequences, it is almost inevitable that at some stage we will encounter a situation where the specified probabilities are incompatible with the distribution of Z. If this occurs, to continue simulation we must either adjust the probabilities or the distribution of Z.

In practice, incompatibility usually occurs when one or two of the individual ps are very different from the majority — in many situations, this is unrealistic (it is usually related to dependence upon previous days' values, and the problem can be reduced by including averages of previous days' values as covariates). Since the objective of this spatial dependence model is to reproduce a plausible distribution for the number of wet sites, the software deals with incompatibility by modifying the ps rather than the distribution of Z. Specifically, we shrink the ps towards their mean i.e. for each s replace p_s with

$$p_s - \lambda(p_s - \theta) \ , \tag{57}$$

where $\lambda \in (0, 1)$, and θ is the mean of the ps (and the mean-value parameter of the beta-binomial distribution — see equation **(47)**). The expected number of wet sites is unchanged by this adjustment. For small values of λ, the adjustment is a small one, in which case it may need to be repeated until a compatible set of probabilities is found. Repeated adjustments are

guaranteed to find a compatible set of probabilities, since in the limit all of the probabilities are equal and, in this limit, it can be shown that at least one joint distribution exists.

Imputation using this model is, again, straightforward. Without loss of generality, we assume that the values of $Y_1, \ldots, Y_k$ are observed, and that $Y_{k+1}, \ldots, Y_S$ are missing. The starting point is the table of joint probabilities $\{w_{s,z}\}$. We work with the 'observed' sites one at a time, at each stage updating both the distribution of Z, and the probabilities of rain at the remaining sites, to condition on the observations. The procedure differs from the 'unconditional' case only insofar as we take the observed values of $Y_1, \ldots, Y_k$ rather than simulating them.

Known bugs and problems

The following problems are known to occur within the software. Please provide further bug reports to me (`richard@stats.ucl.ac.uk`) in the unlikely event of there being any ...

- If a definition file created under Dos or Windows is used on a Unix system, character strings may read incorrectly. The reason is the extra line feed character (`^M`) used by Dos/Windows: the software reads this as part of the input. This problem may manifest itself via site names or attributes appearing incorrectly (if at all) in output files. The fix is to run a utility such as `dos2unix` / `fromdos` / whatever-the-name-is-on-your-system, on the definition file first.
- Conversely, if a definition or data file created under Unix is used on a Windows system, the absence of line feed characters can cause problems and produce 'end of file' error messages. In Windows, the easiest way to fix this is to open the offending definition / data file in WordPad and save it without making any changes: this will automatically append the required line feed characters.
- If the fitting programs terminate abnormally, they can leave large temporary files lying around. These need to be manually deleted.
- With the Salford FORTRAN compiler for MS/DOS, it is necessary to use the `/INTL` flag when compiling e.g.

  ```
  C:\>FTN77 FIT_GAMM.F /INTL
  ```

- In files `fit_logi.f` and `fit_gamm.f`, the built-in error trap for an incorrect value of the parameter `BYTELN` doesn't work under the AIX compiler, because it produces a warning condition rather than an error. I don't know how to work around this. The symptom is a string of warning messages to do with 'record length too short' or something - if this occurs, you need to increase the value of `BYTELN` (set in a `PARAMETER` statement at the top of the source files) and recompile.

Revision history

October 2011: fixed some bugs in the checking of spatial dependence model specifications, arising from the changes made in August 2011. Sorry everyone!

August 2011:

- Changed the format of file cor_gamm.dat to include inter-site separation information. This is useful for exploring the decay of correlations with distance outside of GLIMCLIM.

- Added the option to model spatial dependence in rainfall occurrence via a latent Gaussian field, with correlation structure chosen to match the observed joint rainfall occurrence probabilities at each pair of sites. This makes the software potentially suitable for use over much larger spatial regions than was previously possible.
- Added the facility to fit distance-dependence spatial correlation models, again for use when dealing with larger spatial regions.
- Removed code for producing a normal probability plot of Anscombe residuals, since this code was inaccurate. The required probability plots can be produced easily using a standard statistical software package in any case.
- Made some changes to the `SCRFNO` routine in file `glm_base.f`, for compatibility with the gfortran compiler; also fixed some nonstandard code in routine `IWLS` (file `fit_both.f`) that gfortran picked up (what about all those other compilers over the years?!)

March 2006:

- Fixed a small bug in `glm_base.f`, which caused runtime errors with some compilers.
- Removed code for producing a normal probability plot of Anscombe residuals, since this code was inaccurate. The required probability plots can be produced easily using a standard statistical software package in any case.

August 2005: Increased the accuracy of the dependence-adjusted likelihood ratio test, by adding a secondary adjustment (see Appendix C.3 for details).

June 2005: Made a small change to the FORTRAN output format for daily records from the simulation program, so that the format statement itself does not impose a restriction on the number of sites.

August 2004:

- Added code to calculate increase in log-likelihood between initial and final parameter estimates; also to adjust this increase for inter-site dependence, following the methodology in Bate (2004).
- Added a check that the number of external predictors defined does not exceed the allocated storage.
- Added a check that the input data file `gaugvals.dat` is in the right order.

April 2004: Revised calculation of standard errors for mean monthly and annual Pearson residuals. Previously this calculation was approximate and assumed that each subset of residuals contained roughly equal contributions from each pair of sites. The new calculation is a bit more expensive, but exact (see Appendix D).

November 2002:

- Added code for the modelling and simulation of thresholded data, to deal with unreliable small values in positive variables.
- Fixed a minor bug which caused fitting programs to crash under any spatial dependence structure except 'independence', when the last day in the fitting period contained only 1 valid case for fitting.

October 2002: Small bug fixed in `glm_base.f` — when simulating, check on definition of dispersion parameter was programmed incorrectly.

September 2002:

- Standard errors of parameter estimates can now be adjusted for inter-site dependence if the model definition file specifies spatial structure. The adjustment uses a robust sandwich variance estimator, and is independent of the particular spatial structure used.
- Reduced the computational cost of fitting models (e.g. by using a Cholesky, rather than LU, decomposition to solve the score equations at each iteration, and reducing the amount of disk access required).

August 2002:

- Added facility to included weighted averages of previous days' values as covariates in models, and to estimate parameters in weighting schemes.
- Added facility to include lagged values of external time series as covariates in models.
- Changed the way in which the threshold for 'trace' values is defined to the system.
- Changed the number of header rows in model definition files. **Users of earlier versions, please note!**
- Improved algorithm for computing the digamma function (required when calculating the MLE of the dispersion parameter in a gamma distribution).
- Corrected a slightly embarrassing, but minor, error in the calculation of Fourier frequencies for a half-year cycle - should be $\pi/91.5$, was $\pi/93$ (NB 366-day year!). Oops, sorry folks ...

July 2001:

- Moved mathematical routines to `rec_math.f`.
- Added new spatial dependence structure for logistic regression models, based on a beta-binomial distribution for the coverage.
- Removed option, in simulation program, to do 'deterministic' simulation (i.e. output means of forecast distributions), since this was a really silly idea in the first place.

January 2001:

- Added facility to include 'external' covariates such as ENSO, NAO etc. into models.
- Redefined some covariate categories.

August 1999:

- Stabilised estimation of parameters in nonlinear transformations by adding ability to 'backtrack' if an iteration of IWLS reduces the log-likelihood. This guarantees eventual convergence, allegedly.
- Estimation now performed by a separate routine, IWLS, in file `fit_both.f`.
- Error traps added to ensure that all PARAMETER statements are set correctly.

July 1999: Added facility to represent nonlinear transformations of site attributes via orthogonal functions.

June 1999:

- Improved model definition structure to include nuisance parameters etc., as well as allowing different spatial models.
- Residual spatial correlations are now estimated in the usual way to guarantee values in the right range (supercedes 9/1998 note below).
- Added some spatial dependence structures for occurrence model.
- Fixed a few holes to do with things like dividing by zero when calculating a standard deviation for 1 observation.
- Added facility to compare models with different numbers of lags.

January 1999: Updated model specification to include spatial correlation structure definition in model definition file.

September 1998:

- Changed method of site identification in residual output file - previously the site number was written, now 4-character site code.
- Added estimation of spatial correlation structure as part of residual analysis. NB correlations are estimated as mean sums of cross-products rather than in the usual way (i.e. they're not mean-centred), since the model is assumed to be nearly right by the time we're interested in correlation! There is a possibility of getting estimates > 1 though.

July 1998: First attempt to create a general-purpose package, by pulling together bits and pieces of old code to reflect common structure.

References

Bate, S. (2004). *Generalized Linear Models for Large Dependent Data Sets*. PhD thesis, Department of Statistical Science, University College London.

Chandler, R. E. and Wheater, H. S. (1998). Climate change detection using Generalized Linear Models for rainfall — a case study from the West of Ireland. II. Modelling of rainfall amounts on wet days. Technical report, no. 195, Department of Statistical Science, University College London. `http://www.ucl.ac.uk/Stats/research/Resrpts/abstracts.html`.

Chandler, R. E. and Wheater, H. S. (2002). Analysis of rainfall variability using Generalized Linear Models — a case study from the West of Ireland. *Water Resources Research*, 38, No.10:doi:10.1029/2001WR000906.

Cox, D. R. and Hinkley, D. (1974). *Theoretical Statistics*. Chapman & Hall, London.

Dawid, A. P. (1986). Probability forecasting. In Kotz, S. and Johnson, N., editors, *Encyclopedia of Statistical Sciences*, pages 210–218. Wiley, New York.

Dobson, A. J. (2001). *An Introduction to Generalized Linear Models (second edition)*. Chapman and Hall, London.

Donnelly, T. G. (1973). Algorithm 462: bivariate normal distribution. *Communications of the ACM*, 16(10):638.

Fahrmeir, L. and Tutz, G. (1994). *Multivariate Statistical Modelling Based on Generalized Linear Models*. Springer-Verlag, New York.

Jørgensen, B. (1983). Maximum likelihood estimation and large-sample inference for generalized linear and nonlinear regression models. *Biometrika*, 70:19–28.

Liang, K.-Y. and Zeger, S. (1986). Longitudinal data analysis using generalized linear models. *Biometrika*, 73, no.1:13–22.

Marsaglia, G. and Zaman, A. (1991). A new class of random number generators. *Ann. Appl. Prob.*, 1:462–480.

McCullagh, P. and Nelder, J. (1989). *Generalized Linear Models (second edition)*. Chapman and Hall, London.

Milne, A. A. (1958). *The World of Pooh (the complete Winnie-the-Pooh and The House at Pooh Corner)*. Methuen, London.

Pearson, K. (1901). Mathematical contributions to the theory of evolution — VII. On the correlation of characters not quantitatively measurable. *Phil. Trans. Roy. Soc. Series A*, 195:1–47.

Press, W., Teukolsky, S., Vetterling, W., and Flannery, B. (1992). *Numerical Recipes in FORTRAN (second edition)*. Cambridge University Press.

R Development Core Team (2008). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Wei, B.-C. (1997). *Exponential Family Nonlinear Models*, volume 130 of *Lecture Notes in Statistics*. Springer, Singapore.

Wheater, H. S., Isham, V. S., Onof, C., Chandler, R. E., Northrop, P. J., Guiblin, P., Bate, S. M., Cox, D. R., and Koutsoyiannis, D. (2000). Generation of spatially consistent rainfall data. Report to the Ministry of Agriculture, Fisheries and Food (2 volumes). Also available as Research Report no. 204, Department of Statistical Science, University College London (`http://www.ucl.ac.uk/Stats/research/Resrpts/abstracts.html`).

Yan, Z., Bate, S., Chandler, R. E., Isham, V. S., and Wheater, H. S. (2002). An analysis of daily maximum windspeed in northwestern Europe using Generalized Linear Models. *J. Climate*, 15, No.15:2073–2088.