Practical Project: Hydrological modelling with real rainfall and flow data

INTRODUCTION

Due to the complexities of the different processes involved and widely varying landscape conditions, representative hydrological modelling can often be difficult to achieve. Consequently, where real data is available, this can be invaluable in selecting, calibrating and validating suitable models for a given situation. Geographical Information Systems (GIS) are ideally suited for spatial analysis aspects of hydrology such as identifying river basin catchment boundaries from Digital Elevation Models (DEM). However, they are typically quite poor at directly generating, analysing or presenting temporal data and are thus often loosely coupled with other software. This project will develop Python scripts for ArcGIS to automate extraction of recorded time-series rainfall data that specifically relates to geographic locations selected (typically catchment areas). This will be demonstrated through a simple hydrological model in order to compare against actual measured stream flow data.

DATA SOURCES

Although there are many sources of mean or peak rainfall measurements available, these are typically specified over a long time period and/or a vague geographical location. In order to model dynamic response of a river system (particularly for flood estimation), more finely-grained time/location data is required – which was made available under special licence by The Met Office.

Table 1: Data sources and sets used

Source	Data Set	Description
Meteorological Office (UKMO) (BADC, 2014)	Rain Radar data from the NIMROD System	1km gridded composite C-Band radar images of precipitation that have been calibrated with rain gauges. Data is spaced at 5 minute intervals and available from 2004 to the present day.
Centre for Ecology & Hydrology (CEH)	Plynlimon spatial datasets	Flow rate data and further descriptive data from these experimental/research catchments in Wales.
Ordnance Survey (via Digimap)	1:50k and 1:250k raster maps	For context within GIS map output
Data.gov.uk	Pontbren automatic weather station (AWS) dataset	Historic temperature readings. Used to put historic river discharge rates into context of likelihood of snow or ice.

METHODOLOGY

Unfortunately the Met Office rainfall data is stored in a proprietary ("NIMROD") binary format for which there are no officially supported conversion programs available. The community-supplied programmes and methodologies for converting data (BADC, 2014) have either limited functionality or are awkward to integrate with ArcGIS (Armstrong, 2013). There are generally two requirements for such data:

- Spatial raster data is required (typically for a *distributed* model), preferably for a smaller area than the default whole of the UK. The python "nimrod.py" module has been developed for this.
- Time series of single values of mean rainfall for specifically defined areas (typically river basin catchment or sub-catchment areas). The "gen_rainfall_time_series.py" ArcGIS python script has been developed for this.

Python ArcGIS Script: gen_rainfall_time_series.py

This script (listed in Appendix 1) calculates the mean rainfall rates across an arbitrarily shaped area for a specified time series. An input raster dataset is used to define the precise area perimeter, with the time series defined by a list of dates, a start time, a fixed time-step between images and a total image count. The script will automatically locate the correct data files (and will check all required dataset files are accessible before processing commences as this can be a slow process).

It is easiest to describe the script operations by illustrating the equivalent steps that would be performed if done interactively within ArcMap:

1. Get required raster overlay (of river basin catchment) of interest and extract the horizontal and vertical limits of the area extent of this overlay.

Note that such an estimate for catchment watersheds can be directly generated within ArcGIS from a Digital Elevation Model (DEM), using the standard hydrology tools sequence: Flow_Dir (to calculate the probable water flow directions), Fill (to make the DEM depressionless), Flow_Acc (accumulating flow direction vectors to give an indication of catchment areas, Watershed (after defining the pour point outlet(s) of catchment area(s)).

Figure 1: Raster overlay defined for (catchment) area of interest

2. (For each time-step) get required rainfall raster image for the whole of the UK:

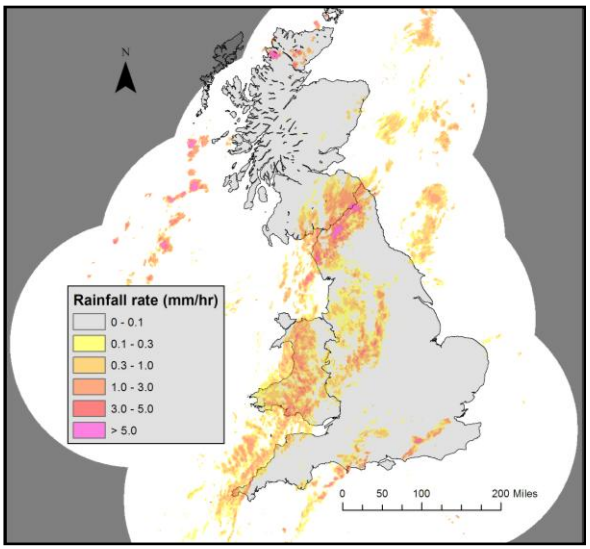

Figure 2: Full UK Rainfall raster image

3. Clip image to a rectangle a little larger (to avoid filtering artefacts at the resampling stage) than the area of interest. The newly defined python function ***nimrod2asc*** *(in nimrod.py) is used for this.*

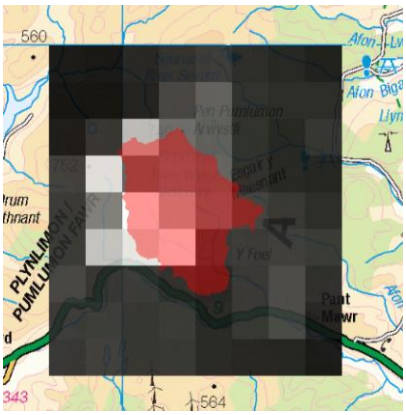

Figure 3: Rainfall raster image clipped to rectangle of interest

4. Resample the rainfall image to give it the same resolution as the (catchment) area raster to maximize precision of area included. Cubic interpolation filtering is used to make best use of adjacent (not included) pixels to smooth the rainfall image.

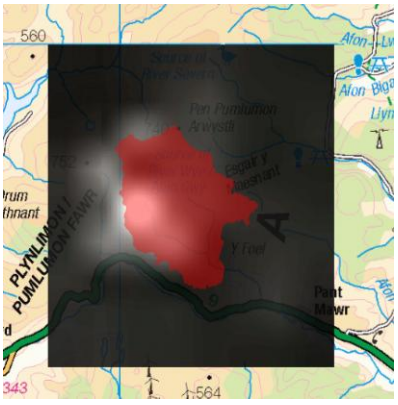

Figure 4: Rainfall image resampled to cell size of (catchment) area raster

5. Clip the resampled rainfall image to the shape required.

Figure 5: Resampled rainfall image clipped to (catchment) area

From here the "Raster Properties" function can be used to extract the mean rainfall within the clipped region, which is written out to a comma-separated variable (CSV) text file together with a timestamp before the whole sequence is repeated for the next time-step. The CSV file can later be read into a spreadsheet or other software to process the time-series results.

Python Module nimrod.py (function nimrod2asc)

This is listed in Appendix 2. The core of it is based on a community-supplied script "read_nimrod.py" (BADC, 2014) and the NIMROD format specification (Stone et al, 2008), but it has been rewritten and extended to perform the following tasks:

- Read key data from header: size of image, raster cell size, British National Grid location.
- Extract a rectangular subset of the image around specified bounds and recalculate BNG location.
- Generate an ArcGIS ASCII GRID (*.asc) format raster file, taking care to use "xllcenter" rather than the standard "xllcorner" header as this would introduce a 500m position error.

Figure 3 (earlier) illustrates how the ArcGIS script would use function nimrod2asc to extract just the rectangular section of the rainfall image required.

Test Methodology: comparison with flow data

A catchment area was selected for which detailed flow data (discharge Q) could be obtained – in this case the upper reaches of the River Wye above the gauging station at Cefyn Brwyn in Wales. This is a relatively small catchment (10.6 km^2) of unforested moorland which has high rainfall and steep gradients. As such, runoff into the extensive stream network (figure 6) and thus out of the catchment should be fairly fast, with relatively limited groundwater flow or individual storm events. The slightly smaller (8.7 km^2) adjacent upper Severn catchment was also considered, although this is largely forested.

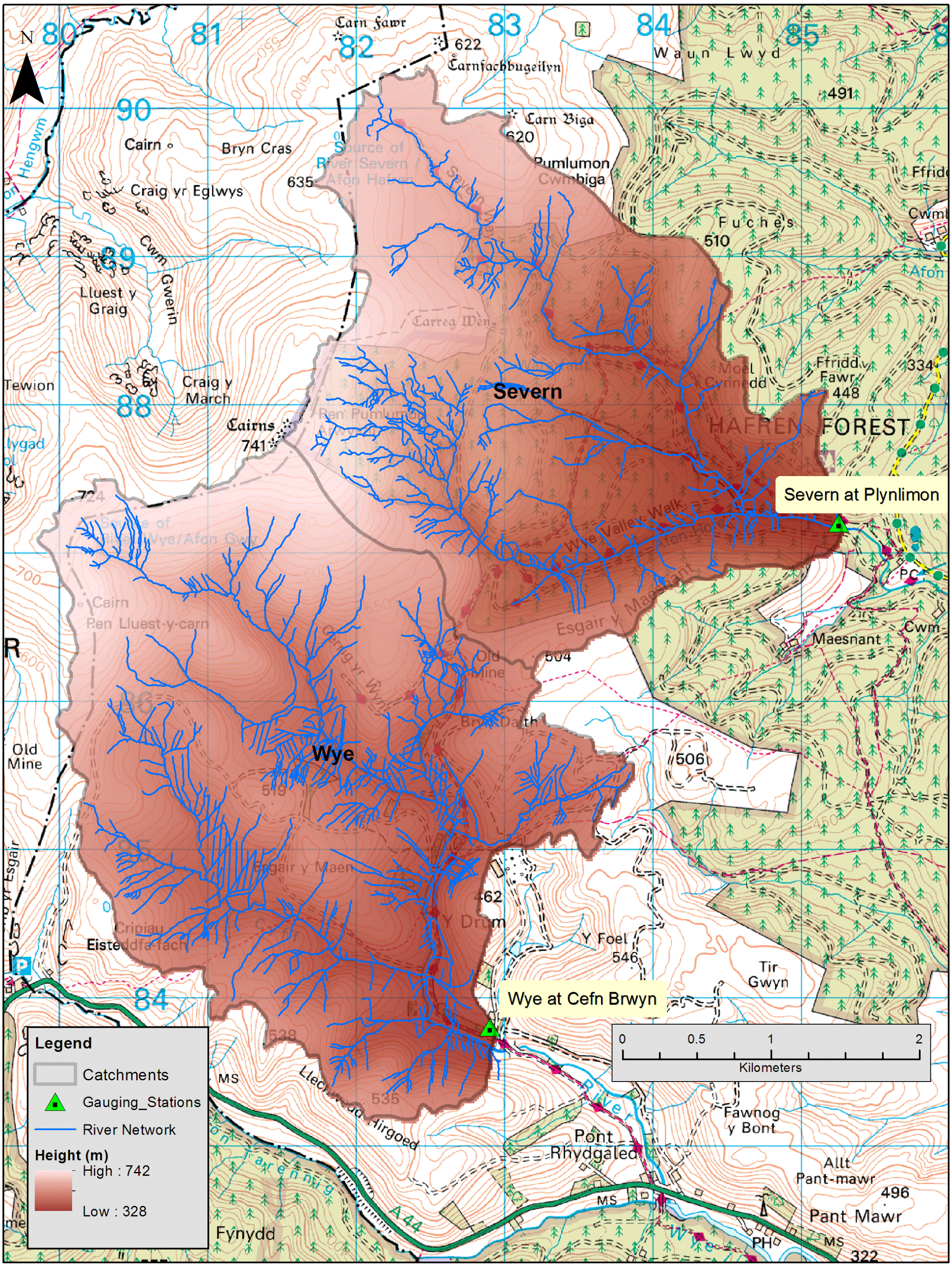

Figure 6: Catchments limits of the upper Wye and Severn rivers in Wales

Using the gen_rainfall_time_series.py script, time-series rainfall was extracted for this watershed at 30 minute intervals over a period of 10 days from 25 February 2008. By multiplying the mean catchment rainfall values by the catchment area, values of rainfall water volume per second could be derived and compared with the catchment river discharge rate Q. Clearly rainfall doesn't immediately appear at the output of the catchment, but by continuously summing both rainfall and discharge over the 10 days comparisons could be made of total volumes of water entering (via rainfall) and leaving (by river flow) the catchment.

RESULTS AND ANALYSIS

Each of the 2 catchment rainfall extractions parsed 488 UK rainfall maps (3.5GB) to generate 30 minute time steps over a 10 day period. This took approximately 45 minutes on a (2007-vintage) Dell D630 Laptop. Figure 7 illustrates that although the catchment areas are small and immediately adjacent, there are at times significant differences in mean rainfall across each catchment.

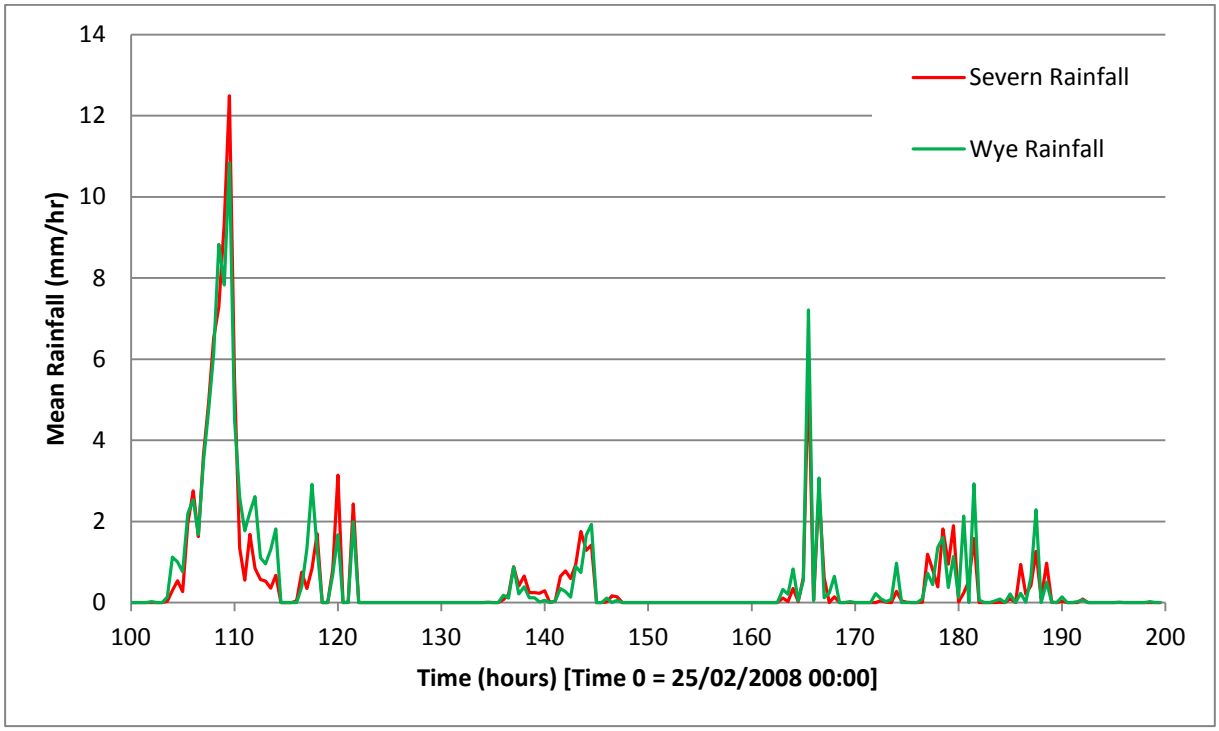

Figure 7: Comparison of mean rainfall in adjacent catchment areas

In figure 8, the direct causal effect of peaks in rainfall (Rf) monitored by weather radar leading to (more spread out) peaks in river discharge (Q) measured at the gauging station clearly confirms that the 2 independent data sets are aligned. The cumulative sums of rainfall (Accum Rf) and discharge (Accum Q) give a less promising result. It would be expected that the sum of rainfall would always be higher than discharge due to water returned to the atmosphere through evapotranspiration – but we are seeing the opposite. The discharge Q can be separated into 2 components: storm flow (in relatively quick response to rainstorm events) and groundflow (relatively slow and slowly changing flows of water that have infiltrated the soil and descended to the saturated region, i.e. below the water table). The groundflow is usually indicated by the asymptotes that Q settles to – it can be seen that in the last few days of the chart that Accum Q diverges from Accum Rf entirely due to this groundwater flow. It is plausible that there might be a significant amount of snow or ice in the system (from before the start of the trace) that is melting. Snowfall or ice formation is probably the reason why the heavy precipitation from the 168 hour mark does not translate to river discharge as temperatures can be seen to be dropping sharply after that point (figure 9). The dashed traces show that a simple exponential delay model derived purely from Rainfall Rf can give reasonable storm flow predictions (particularly for well-separated storm events). This equation is of the form:

Exp Model Q -> (Exp Model Q) + Rf *(Input Gain) – ((Exp Model Q) – (Throughflow))*(Decay rate)

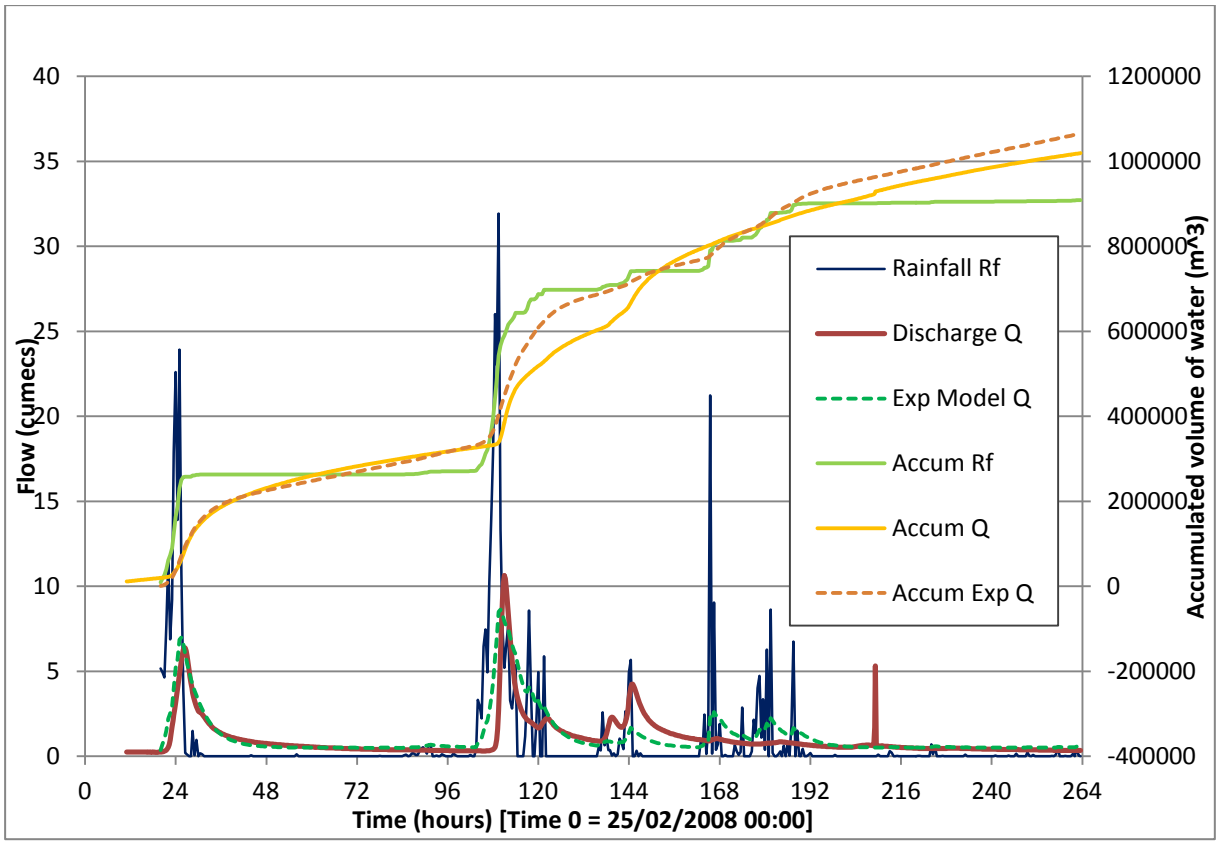

Figure 8: River Wye precipitation vs. river discharge Q

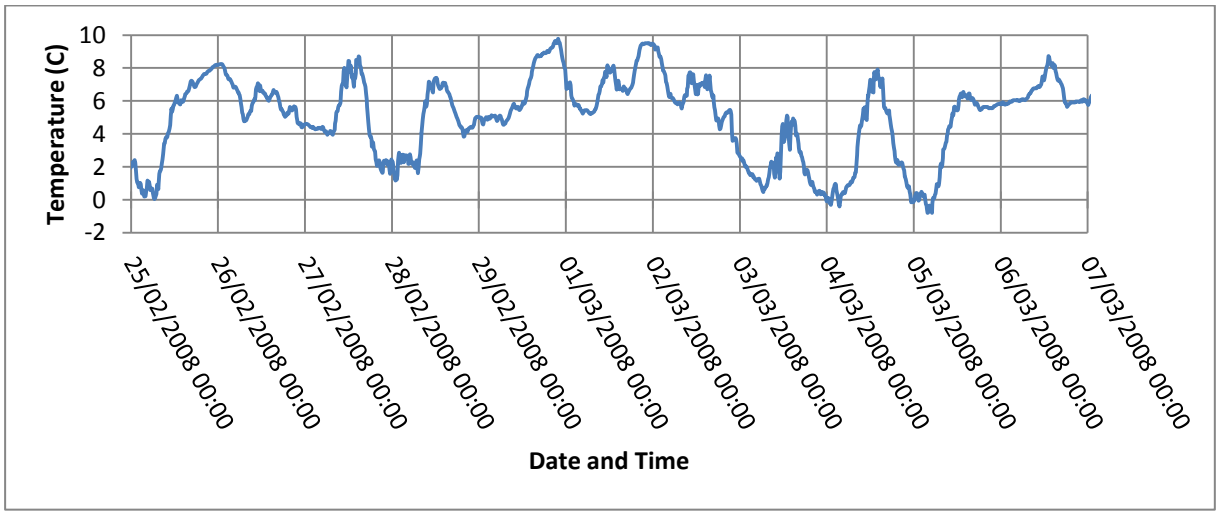

Figure 9: Pontbren (close to Plynlimon) historic temperature measurements

CONCLUSIONS

The rainfall/discharge comparisons in this report demonstrate that the weather radar maps can be used as a plausible source for time-series hydrology modelling. The time taken to extract data for each area does limit the method to lumped parameter models rather than distributed ones. However, it is believed that ArcGIS fetching/storage of raster data consumes most of the time and the script would be significantly faster if functions were implemented directly (without ArcGIS). The ArcGIS environment theoretically makes it easy to automate processes by using the "Copy python snippet" menu options on the results window of previous (interactive) operations. However, the inefficiency (and often unreliability) of the results does question this approach. For the scripts that have been developed, a useful next stage would be to automatically retrieve the required NIMROD data files via FTP rather than having them reside on the local machine.

ACKNOWLEDGEMENTS

This work used public sector information licensed under the Open Government Licence v2.0 obtained via the data.gov.uk website. Digital boundary and raster map data was supplied by Ordnance Survey via the Digimap (digimap.edina.ac.uk) website. These data are Crown copyright and are reproduced with permission of the Office of Public Sector Information (OPSI). Meteorological Office (UKMO) data was supplied under licence through the NERC BADC Data Centre. This report also contains data supplied by Natural Environment Research Council (Plynlimon datasets) which are ©NERC (Centre for Ecology & Hydrology).

REFERENCES

Armstrong, A. 2013**.** Rainfall Radar. [Online]. [Accessed 14 May 2014]. Available from: http://alonaarmstrong.wordpress.com/radar/

British Atmospheric Data Centre (BADC). 2014. BADC Met Office Rain Radar Data from the NIMROD System. [Online]. [Accessed 14 May 2014]. Available from: http://badc.nerc.ac.uk/view/badc.nerc.ac.uk__ATOM__dataent_nimrod

Met Office. 2014. Monthly rainfall measurements. [Online]. [Accessed 31 April 2014]. Available from: http://www.metoffice.gov.uk/pub/data/weather/uk/climate/stationdata/yeoviltondata.txt

Stone, D, Harrison, D, Standing, R. 2008. Nimrod System Documentation Paper No.2: Nimrod format for image and model field files. [Online]. [Accessed 14 May 2014]. Available from: http://badc.nerc.ac.uk/cgi-bin/data_browser/data_browser/badc/ukmo-nimrod/doc/

APPENDIX 1: ArcGIS Python Script for Extracting Catchment-Specific Rainfall (gen_rainfall_time_series.py)

```
#!/usr/bin/python
#---------------------------------------------------------------------------
# Generate Catchment-specific Rainfall Time Series Data
# Inputs:
#     MetOffice NIMROD UK rainfall data (1km grid)
#     Catchment Areas (in ArcGIS Raster format)
#
# Richard Thomas May 2014
#---------------------------------------------------------------------------

import os      # Standard package for operating system dependent functionality
import arcpy   # ArcGIS's Python site package
import nimrod  # Convert NIMROD format rainfall files to .asc files

# ---- Define operating parameters ----

# Name and location of catchment raster (must be in a personal geodatabase)
#catchmentRaster = "Wye_Catchment_Raster"
catchmentRaster = "Severn_Catchment_Raster"
sourceMdbPath = "work.mdb"

# Dates in order, plus hour/minute to start on first date
dateList = ["20080225", "20080226", "20080227", "20080228", "20080229",
            "20080301", "20080302", "20080303", "20080304", "20080305",
            "20080306"]
hour     = 20
minute   = 0

# Step between each reading (in minutes)
minStep  = 30

# Total number of steps
totalCount = 488

# Location / file naming of NIMROD rainfall binary files
nimrodPath = "NIMROD_data/"
nimrodSuffix = "_nimrod_ng_radar_rainrate_composite_1km_merged_UK_zip"


# ---- Perform initial calculations & checks ----

# Initialise variables
dateIx = 0
errors = 0
timeList = []

print "Creating and checking list of NIMROD files.."
while totalCount > 0:
    date = dateList[dateIx]
    timeList.append("%s%02d%02d" % (date, hour, minute))
    totalCount -= 1
    minute += minStep
    while minute >= 60:
        minute -= 60
        hour += 1
        while hour >= 24:
            hour -= 24
            dateIx += 1
            if dateIx > len(dateList):
                raise "Error: ran off end of date list"

# Run through quickly to check files all exist and are readable
for snapshot in timeList:
    nimrodFile = nimrodPath + snapshot + nimrodSuffix
    try:
        nimrodFileId = open(nimrodFile, "rb")
        nimrodFileId.close()
    except:
        print "File not found: " + nimrodFile
        errors += 1

if errors > 0:
    raise "Bailing out due to errors"


# ---- Start ArcMap interaction ----

class LicenseError(Exception):
    pass

# Check out Spatial Analyst Licence (if available)
print "Checking out Spatial Analyst licence.."
```

```
try:
    if arcpy.CheckExtension("Spatial") == "Available":
        arcpy.CheckOutExtension("Spatial")
    else:
        raise LicenseError

    # Throw an exception if an ArcGIS tool produces a warning
    arcpy.SetSeverityLevel(1)

    # Sets current folder as workspace
    arcpy.env.workspace = os.getcwd()

    # Allow over-writing of files
    arcpy.env.overwriteOutput = True

    # Get catchment raster from file
    catchRas = arcpy.sa.Raster(sourceMdbPath + "/" + catchmentRaster)

    # Get bounds of catchment
    extent = catchRas.extent

    # Sets cell size of output raster layers to be the same as catchment
    cellSize = catchRas.meanCellWidth

# Catch licence being unavailable
except LicenseError:
    print "Spatial Analyst license is unavailable"

# Define column headings in output CSV file
csvOutText = "Year, Month, Date, Hour, Minute, Rainfall (mm/hr)\n"

# ---- Loop through NIMROD files to extract local rainfall data ----

for snapshot in timeList:
    nimrodFile = nimrodPath + snapshot + nimrodSuffix
    rainRasFile = catchmentRaster + "_" + snapshot + ".asc"

    try:
        # Generate .asc raster file from NIMROD data, but only make it a
        # little bit bigger than catchment as we are about to upsample it
        nimrod.nimrod2asc(nimrodFile, rainRasFile, extent.XMin, extent.XMax,
                          extent.YMin, extent.YMax)

        # Get catchment-specific rainfall raster from ASC file
        rainRas = arcpy.sa.Raster(rainRasFile)

        # perform (Cubic/Nearest) interpolation of 1km grid to 15m grid
        resampRainPath = sourceMdbPath + "/rainResampled"
        arcpy.Resample_management(rainRas, resampRainPath, cellSize, "CUBIC")
        #arcpy.Resample_management(rainRas, resampRainPath, cellSize, "NEAREST")
        resampRainRas = arcpy.sa.Raster(resampRainPath)

        # Select only rain that falls within catchment
        catchResampRainPath = sourceMdbPath + "/catchRainResampled"
        arcpy.gp.Con_sa(catchRas, resampRainRas, catchResampRainPath, "#", "#")
        catchResampRainRas = arcpy.sa.Raster(catchResampRainPath)

        # Get mean rainfall that falls within catchment for this timeslot
        # Also correct for fact that NIMROD data is mm/hr * 32
        rpResult = arcpy.GetRasterProperties_management(catchResampRainRas,
                                                        "MEAN","#")
        meanRainfallRate = float(rpResult.getOutput(0)) / 32.0

        # Our job for this raster is done: write mean rainfall to CSV file
        csvOutText += snapshot[0:4] + ", " + snapshot[4:6] + ", " \
                    + snapshot[6:8] + ", " + snapshot[8:10] + ", " \
                    + snapshot[10:12] + ", %f\n" % meanRainfallRate

        # Remove ASC file
        del rainRas
        os.remove(rainRasFile)

    # Catch warnings from geoprocessing tools (if severity level = 1)
    except arcpy.ExecuteWarning:
        print arcpy.GetMessages()

    # Catch errors from geoprocessing tools
    except arcpy.ExecuteError:
        print arcpy.GetMessages()

# Return Spatial Analyst Licence (this would happen anyway when python quits)
arcpy.CheckInExtension("Spatial")

# Write results to CSV file
outf = open(catchmentRaster + ".csv", 'w')
outf.write(csvOutText)
outf.close()
```

```
# Delete intermediate calculation rasters
# (Note: temporary raster files would be deleted when Python window or
#  ArcGIS quits anyway)
arcpy.Delete_management(resampRainRas)
arcpy.Delete_management(catchResampRainRas)

print "Done."
```

APPENDIX 2: Python Module for converting NIMROD binary files to ".asc" files

```
#!/usr/bin/python
#-------------------------------------------------------------------------------
# Module: nimrod
#
# Convert NIMROD format rainfall files to .asc files,
# extracting only rainfall data subset required (1km buffer round catchment).
#
# Richard Thomas May 2014
# (based on read_nimrod.py by Charles Kilburn Aug 2008)
#-------------------------------------------------------------------------------

import os, stat, re, sys, time, glob
import struct, array

def nimrod2asc (nimrodFilename, ascFilename, xmin, xmax, ymin, ymax):

    print "> " + nimrodFilename

    file_id = open(nimrodFilename,"rb")
    record_length, = struct.unpack(">l", file_id.read(4))
    if record_length != 512:
        raise "Unexpected record length", record_length

    # Read first 31 2-byte integers (header fields 1-31)
    gen_ints = array.array("h")
    gen_ints.read(file_id, 31)
    gen_ints.byteswap()

    date_time_str = "%4.4d%2.2d%2.2d_%2.2d%2.2d" %(
        gen_ints[0], gen_ints[1], gen_ints[2], gen_ints[3], gen_ints[4])
    ncols = gen_ints[16]
    nrows = gen_ints[15]

    # Read next 28 4-byte floats (header fields 32-59)
    gen_reals = array.array("f")
    gen_reals.read(file_id, 28)
    gen_reals.byteswap()

    cellsize = gen_reals[3]
    xllcenter = gen_reals[4]
    y_top_center = gen_reals[2]
    yllcenter = (y_top_center - gen_reals[5] * gen_ints[15])
    nodata_value = gen_reals[6]

    # Read next 45 4-byte floats (header fields 60-104)
    spec_reals = array.array("f")
    spec_reals.read(file_id, 45)
    spec_reals.byteswap()

    # Read next 56 characters (header fields 105-107)
    characters = array.array("c")
    characters.read(file_id, 56)

    # Read next 51 2-byte integers (header fields 108-)
    spec_ints = array.array("h")
    spec_ints.read(file_id, 51)
    spec_ints.byteswap()

    record_length, = struct.unpack(">l", file_id.read(4))
    if record_length != 512:
        raise "Unexpected record length", record_length

    # Calculate min and max pixel number in each row to use
    # (Add an extra pixel each side to help smoothing)
    xMinPixelId = int((xmin - xllcenter) / cellsize) - 1
    xMaxPixelId = int((xmax - xllcenter) / cellsize) + 3
    xllcenter += xMinPixelId * cellsize
    ncols_out = xMaxPixelId - xMinPixelId

    # Calculate min and max rows of pixels to use
    # (Add an extra pixel each side to help smoothing)
    yMinPixelId = int((y_top_center - ymax) / cellsize) - 2
    yMaxPixelId = int((y_top_center - ymin) / cellsize) + 2
    yllcenter += (nrows - yMaxPixelId) * cellsize
    nrows_out = yMaxPixelId - yMinPixelId
```

```
# Read the Data
array_size = ncols * nrows

record_length, = struct.unpack(">l", file_id.read(4))
assert record_length == array_size * 2, "Unexpected data record length (1)"

data = array.array("h")
try:
    data.read(file_id, array_size)
    record_length, = struct.unpack(">l", file_id.read(4))
    assert record_length == array_size * 2, "Unexpected data record length (2)"
    data.byteswap()
except:
    print "Data Read failed"

file_id.close()

# Write to output file
print "Writing data array (%d x %d) to file: %s\n" % (nrows_out, ncols_out,
                                                     ascFilename)
outf = open(ascFilename, 'w')

outf.write("ncols          %d\n" % ncols_out)
outf.write("nrows          %d\n" % nrows_out)
outf.write("xllcenter     %d\n" % xllcenter)
outf.write("yllcenter     %d\n" % yllcenter)
outf.write("cellsize       %.1f\n" % cellsize)
outf.write("nodata_value  %.1f\n" % nodata_value)

for i in range(yMinPixelId, yMaxPixelId):
    for j in range(xMinPixelId, xMaxPixelId - 1):
        outf.write("%d, " % data[i*ncols + j])
    outf.write("%d\n" % data[i*ncols + xMaxPixelId - 1])
outf.close()
return
```